الجمهورية الجزائرية الديموقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العلي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf :……

Centre Universitaire
Abd elhafid boussouf Mila

Institut des Sciences et de la Technologi **Département de Mathématiques et Informatique**

Mémoire préparé En vue de l'obtention du diplôme de

Master

En :Filier informatique

Spécialité Sciences et Technologies de l'Information et de la Communication (STIC)

Construction d'une ontologie gérée par un Web service

Préparé par :

- Djoudi Abdelhakim
- Yacoub Chouaib

Soutenue devant le jury :

Président : Abderazek samira
Examinateur: Zekiouk mounira
Encadrer : Deffas Zineb

Année universitaire : 2014/2015

Remerciements

هَـٰذَا مِن فَضْلِ رَبِّي لِيَبْلُوَنِي ءَأَشْكُرُ أَمْ أَكْفُرُ ۖ
وَمَن شَكَرَ فَإِنَّمَا يَشْكُرُ لِنَفْسِهِۦ ۖ
وَمَن كَفَرَ فَإِنَّ رَبِّي غَنِيٌّ كَرِيمٌ ﴿٤٠﴾

Tout d'abord nous remercions Dieu pour nous avoir aidé, donné le savoir, la force et la patience pour réaliser ce travail.

Aussi nous tenons à remercier nos parents, nos frères, sœurs pour leurs soutiens, leurs encouragements et leurs sacrifices.

Nous exprimons notre profonde gratitude à nos encadreurs :

Madame Deffas. Z

Nous les remercions pour leur gentillesse, leur disponibilité et leur patience.

Nous remercions aussi nos enseignants qui nous ont apporté du plus et qui n'ont pas lésiné à nous transmettre leur savoir.

Nous adressons également nos remerciements aux membres de jury, dont leur présence est un grand honneur pour nous.

Dédicace

En signe d'amour de gratitude et de respect, je dédie ce modeste travail a :

A mes très chers parents qui m'ont soutenu le long de mon cursus.

A ma très chère mère qui a toujours su être à mes côtés.

Ma gratitude envers elle ne sera jamais assez.

A la mémoire de mon père (Paix a son âme) qui a voulu me voir réussir.

A mon frère (Zineeddine) et Mes sœurs (Koki, Fatmta, Abla, Badra, Wassila) ainsi que leurs petits anges (Nour, Siraj, Lolo, Ramossa,…..).

A mon ami et binôme « chouaib »

*Spécial dédicace à mon ami et colégue et frére « **SALAH** » qui ma bien aidé le long de mon parcours universitaire*
*À mes amis :(**Guigosse**, Faissal, Batrot, Bachir, Damoh, Dido, Zino, Ossama, Badro, Bemzima…,Djaber21,Kamal28,Sami12, Samir43, Fouad43, Wassim43, Mojib34, Kamal34,Yazzid43, Amin43, Ishak43…)*

A toutes mes amies, en particulier :(Khayra, Saida, … Assya , Affaf, Smahan, Lhajja43, Iman43, Zina43, Romaissa43, Nawal43,..)

A toute ma famille.

A mes proches grands et petits.
A l'ensemble des amis que j'ai connu pendant mes études.
Que Dieu les protège tous …

…Djoudi Abdel hakim…

Dédicace

Je tiens en tout premier lieu à remercier le dieu, je voudrais dédie ce modeste travail:

À mes très chers parents qui m'ont tant soutenu et encouragé dans tous les domaines et surtout pour réaliser ce mémoire :

mon père AHMAD qui m'a donné tous ces pouvoir dans mon parcours d'étude, il m'a donné le courage, la volonté, la confiance en moi-même, que dieu le protège.

Ma très chère mère KHADOUJA la plus chère personne pour moi, la meilleure femme dans le monde, que dieu le protège.

À mes beaux-frères

À mes chères sœurs

À Tous mes amis

et mes collègues de centre universitaire de Mila, surtout filière Informatique «Stic2».

A tous mes enseignants.

À Tous ceux qui me sont chers.

A tout Person qui me connaît "CHOUAIB"

Que Dieu les protège tous …

…yacoub Chouaib…

Résumé

L'intégration des ontologies et de la technologie des Web services dans le domaine médicale, peut apporter une contribution primordiale dans la réutilisation des ressources médicales ou dans l'échange d'informations. Dans ce mémoire nous nous sommes intéressées à créer une application qui offre aux médecins la possibilité de communiquer entre eux à l'aide des services web élémentaires ou composés, tout en se basant sur une ontologie du domaine.

Mots clés : Ontologie, Web service, web Service sémantique, domaine médicale

Abstract

The use of web service technology in the medical field can make a major contribution, especially when we add the semantic dimension (the ontology). These two factors allow the reuse of medical resources and permit a high level of interoperability. In this work we are interested in creating an application which offers to the doctors the possibility of communication; these processes are based on domain ontology and web services.

Key words: Ontology, Web service, Semantic web service, domanne Médicale

ملخص

الدمج بين الأنطولوجيا وخدمات الويب في مجال الطب يسمح بتحقيق مساهمات فعالة سواء من خلال اعادة استعمال الموارد الطبية او من خلال تبادل المعلومات .

لقد تطرقنا من خلال هذا العمل الى انجاز برنامج يمنح امكانية التواصل بواسطة خدمات الويب وذلك بواسطة الاعتماد على الانطولوجيا .

الكلمات الرئيسية الأنطولوجيا خدمات الويب خدمات الويب ذات الدلالة ..

Table des matières

Sommaire

Chapitre 01 : Les web services

Table des matières

Table des matières

Table des matières

Table des matières

Chapitre 05 : Implémentation

Conclusion et perspectives

Liste des figures

Liste des figures :

Liste des figures

Liste des Tableaux

Liste des tableaux

Introduction

Générale

Introduction générale

Les besoins d'accéder de façon uniforme à des sources de données multiples et réparties sont chaque jour de plus en plus forts dans les systèmes administratifs, éducatifs, et particulièrement, dans nos hôpitaux, là où l'accès à l'information pertinente dans un temps limité est primordial.

Les applications réparties sont apparues pour répondre à ces besoins, qui grâce à eux les informations ne sont plus stockées sur un seul et même serveur mais peuvent être dispersées aux quatre coins de la planète et ceci sur un nombre infini de serveurs, mais malgré cela elles ont été beaucoup critiquées sur le fait qu'elles n'autorisaient guerre l'incompatibilité entre les langages de programmation, un inconvénient parmi plein d'autres qui a suscité une nouvelle technologie multi langages, multiplateformes, facile à implémenter sur différents plateformes qu'est les services web.

Par ailleurs, les services Web sont devenus le nouveau point de convergence de l'ensemble des acteurs du marché de l'informatique, grâce à la disponibilité d'outils et de standards permettant la découverte et l'invocation automatisées des fonctions métiers via un échange de messages informatisés (SOAP, UDDI, WSDL). Cependant ces mêmes standards, largement utilisés actuellement, ne permettent pas de décrire les fonctionnalités des services Web que de manière syntaxique ce qui rend leur découverte moins efficace.

L'utilisation d'une ontologie a pour ambition de lever cette difficulté, elle permet un meilleur partage de connaissances par les différents intervenants de la plateforme (aussi bien par l'homme que par la machine), ainsi que la représentation sémantique de leurs contenus.

Dans ce contexte, nous essayerons de construire une ontologie concernant le domaine médicale et plus précisément qui spécifié à la maladie de cancer. Par la suit, nous proposons une architecture à base d'un Web service pour gérer l'ontologie construite et dont l'objectif est de faciliter la recherche des informations et la mise à jour de l'ontologie par l'expert de domaine. Notre cadre d'étude est donc celui des Web services sémantique. L'origine de notre approche repose sur le fait que les deux aspects ontologie et web service sont bien adaptés pour servir le domaine de la médecine.

Pour présenter le travail, nous avons élaboré le plan de lecture suivant:

- Le premier chapitre est dédié à la présentation du domaine des Web service : caractéristiques, architectures, technologies et langages.
- Le deuxième chapitre est consacré à la présentation des ontologies : notion des ontologies, leurs différents types et composants, et quelques outils de manipulation.
- Le troisième chapitre décrit les différentes étapes de construction de l'ontologie. Le résultat est une ontologie conceptuelle valide par des experts.
- Le quatrième chapitre présente notre architecture proposé suivi par la conception de l'application.
- Le cinquième chapitre est composé de deux parties, la premier présente la démarche d'implémentation de l'ontologie par l'éditeur protégé, le résultat est une ontologie opérationnelle. La deuxième partie est consacrée à l'implémentation de l'application web qui utilise le web service.

Chapitre 01

Les Web services

Introduction

Le web est en perpétuelle évolution, du coup nous sommes passés d'une vision, dans Laquelle, il était constitué d'un ensemble de pages accessibles par mots clés à une vision du Web, où celui-ci devient un fournisseur de ressources accessibles par leurs contenus. Ces ressources sont constituées par des machines, des programmes et des contenus accessibles en ligne. Cela dit, leur diversité rend l'échange d'informations entre eux très complexe, d'où la nécessité des standards permettant une formalisation de cet échange et ainsi la notion de service web a vu le jour.

Dans ce chapitre, nous allons nous focaliser sur les principaux concepts à savoir, les services web, ainsi que leur architecture et les différents standards utilisés.

I.1. Un peu d'histoire

Le World Wide Web on s'accroit de manière exponentielle depuis les années suivantes [1]:

- En 1990 : Ce phénomène est dû en partie aux entreprise qui ont vu dans l'Internet l'outil par excellence pour ce faire connaitre. Mais également aux organismes ayant vocation à fournir un service publique d'information, de proximité ou non.
- En 1993 : les entreprises et les états ont exploité l'Internet et en particulier les serveurs Web pour faire le commerce électronique ou bien les gouvernements électroniques.
- En 1998 : les services Web prennent leur origine dans l'informatique distribuée et dans l'événement de Web.

I.2. Définition de web service

Dans le cadre de la programmation orientée service, un service peut être défini comme une entité fonctionnelle auto-contenue, auto-décrite, indépendante des plateformes, et pouvant être décrite, publiée, découverte, invoquée, composée à l'aide de protocoles standards. La technologie des services Web constitue à une approche pour la concrétisation du paradigme de service sur le Web. Par le biais de cette technologie, il devient possible de délivrer et de consommer des services logiciels sur le Web.

Le W3C (World Wide Web Consortium) définit le web service de la manière suivante Un Web service est un système logiciel identifié par un identificateur uniforme de ressources URI, dont les interfaces publiques et les liens sont définis et décrits en XML. Un web service peut être découvert et sélectionné dynamiquement par d'autres systèmes logiciels. Ces derniers, peuvent ensuite interagir avec le service sélectionné en utilisant des messages XML transportés par des protocoles Internet. [2]

Les services web se basent sur un modèle de trois couches :

- Un protocole de communication permettant de structurer les messages échangés entre les composants logiciels.
- Une spécification de description des interfaces des services et enfin une spécification de publication et de localisation de service.
- Les services web sont le fondement de base des architectures orientées services. [2]

I.3. Fonctionnement des Web services

Dans sa première génération, le fonctionnement des Web Services repose sur trois couches fondamentales présentées comme suit:

- **Invocation** : visant à établir la communication entre le client et le fournisseur en décrivant la structure des messages échangés.
- **Découverte :** permettant de localiser un Web service particulier dans un annuaire de services décrivant les fournisseurs ainsi les services fournis.
- **Description:** dont l'objectif est la description des interfaces des Web services (paramètres des fonctions, types de données). [44]

I.4. L'Architecture Orientée Service (SOA)

Actuellement, sous le vocable de SOA, se développe un style d'architecture orientée service permettant de construire des systèmes informatiques évolutifs et adaptables, en améliorant leur qualité et en simplifiant leur intégration dans l'infrastructure informatique de l'entreprise, par recours à des composants réutilisables appelés services.

Dans la suite, nous présentons les concepts sur lesquels repose l'architecture SOA. [43]

I.4.1. Service

Un service est un programme autonome, réutilisable, indépendant des langages de programmation et qui peut s'exécuter sur n'importe quelle plateforme.

Cette composante clef du SOA expose un ensemble d'opérations (fonctionnalités) mises à disposition par un fournisseur à l'attention d'un client selon un ou des contrats prédéfinis.

L'interaction entre le client et le fournisseur est faite par le biais d'un bus de services (médiateur) qui peut être dédié ou être entièrement prise en charge par internet. Dans ce dernier cas, il s'agit des services web [43].

I.4.2. Les concepts de SOA

Les concepts de SOA sont à la mode dans le monde des systèmes d'information. Ces concepts énoncent diverses promesses:

- ***La réutilisation*** : qui signifie la possibilité de réutiliser des services par plusieurs consommateurs.

- ***La composition des services*** **:** qui est liée à la réutilisation. La composition est possible grâce à la modularité des services et permet d'assurer une fonctionnalité plus importante, en s'appuyant sur la composition de services

- ***L'autonomie des services*** : peut être définie par la capacité des services à manipuler ses ressources.

- ***L'interopérabilité des services*** : c'est-à-dire leur interconnexion sans programmation spécifique [43].

I.5. Caractéristiques des web services :

I.5. 1. Web based :

Les Web services sont basés sur les protocoles et les langages du Web, en particulier HTTP et XML.

I.5.2. Self-described, self-contained :

Le cadre des Web services contient en lui-même toutes les informations nécessaires à l'utilisation des applications, sous la forme de trois fonctions : trouver, décrire et exécuter.

I.5. 3. Modular:

Les Web services fonctionnent de manière modulaire et non pas intégrée. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée (ou on récupère) plusieurs applications spécifiques qu'on fait intéropérer entre elles, et qui remplissent chacune une de ces fonctionnalités…

Web Service = HTTP + SOAP + WSDL + Composantes

I.6. Cycle de vie d'un web service

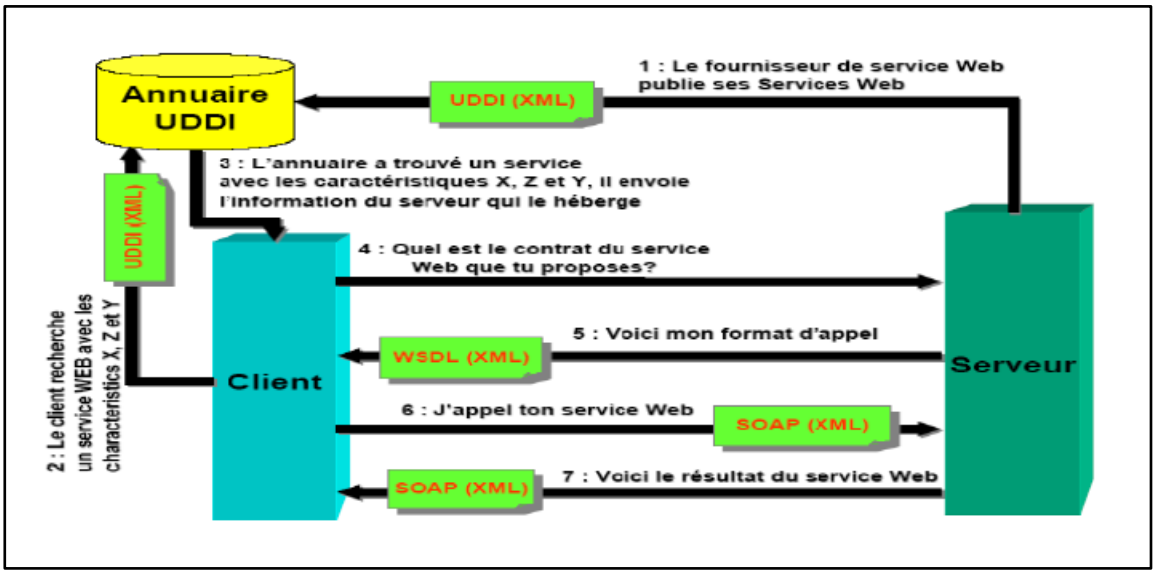

Figure 1.1 : Cycle de vie d'un web service

L'architecture de référence des Web services (cf. Figure1.1) s'articule autour des trois acteurs suivants :

I.6.1. Le fournisseur de service :

Correspond au propriétaire du service. D'un point de vue technique, il est représenté par la plate-forme d'accueil du service. [3]

I.6.2. Le client :

Correspond au demandeur de service. D'un point de vue technique, il est représenté par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un Web service. [3]

I.6.3. L'annuaire des services :

Correspond à un registre de descriptions de services offrant des facilités de publication de services à l'égard des fournisseurs ainsi que des facilités de recherche de services à l'égard des clients. [3]

Les interactions de base entre ces trois acteurs incluent les opérations de publication, de recherche et de liens d'opérations. Nous décrivons ci-dessous un scénario type d'utilisation de cette architecture. Le fournisseur de services définit la description de son service et la publie dans un annuaire de service. Le client utilise les facilités de recherche disponibles au niveau de l'annuaire pour retrouver et sélectionner un service donné. Il examine ensuite la description du service sélectionné pour récupérer les informations nécessaires lui permettant de se connecter au fournisseur du service et d'interagir avec l'implémentation du service considéré. [3]

I.7. Principes de base

Les Web Services reposent sur une architecture orientée services. Comme tout SOA, ils reposent donc sur les trois entités suivantes :

- les consommateurs de services.
- Les fournisseurs de services.
- Les annuaires de services.

Les Web services reposent sur l'articulation de trois standards XML que nous détaillerons davantage par la suite :

- SOAP, un protocole permettant d'invoquer à distance les opérations offertes par un Web Service en utilisant des messages XML.
- WSDL, un standard permettant de d´écrire l'interface d'un Web Service sous la forme d'un fichier de description en XML,
- UDDI, un protocole d'annuaire permettant à la fois de publier et de retrouver un Web Service.

II.8. Architecture des Web services.

L'exposition et l'utilisation des services se fait dans un contexte particulier qui définit clairement les interactions entre le service et ses utilisateurs. Les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, XML-RPC, SOAP et UDDI. L'architecture standard d'un Web service est organisée en plusieurs couches. Chacune d'elles répond à des préoccupations fonctionnelles différentes telles que la publication, la description, la messagerie et le transport. Comme illustré sur la Figure 1.2 [4].

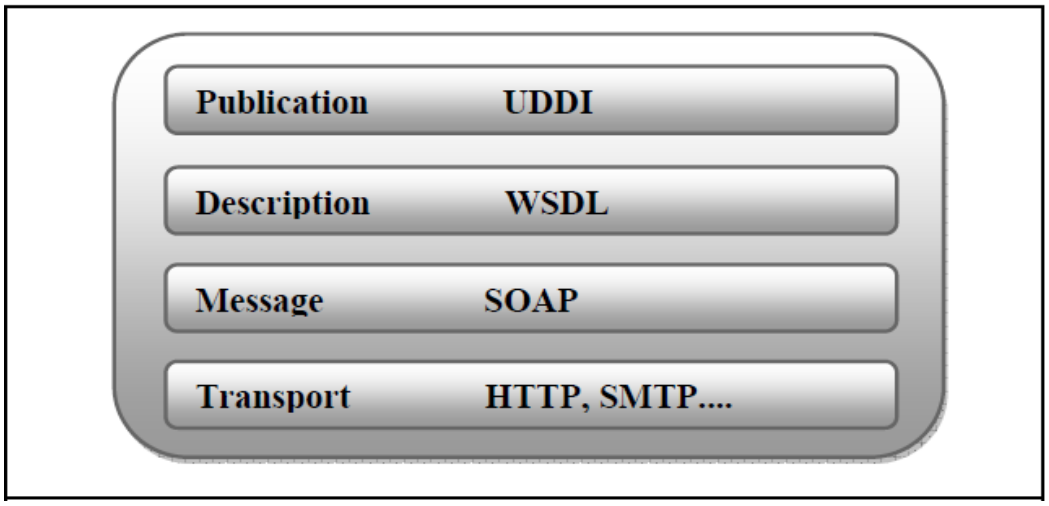

Figure 1. 2 : Cadre architectural des Web Services.

I.9. Intérêt des Web services

Les technologies des services Web peuvent être appliquées à toute sorte d'applications auxquelles elles offrent de considérables avantages en comparaison aux ancienne API propriétaires, aux implémentations spécifiques à une plate-forme et à quelques autres restrictions classique que l'on peut rencontrer (multi plate-forme, multi langage, disponible sur l'internet avec une information actualisée disponible en temps réel, …).

Les entreprises qui mettent à disposition leurs services Web permettent aux développeurs intéressés par ses fonctionnalités de les réutiliser sans avoir à les recoder. Le principe des Web services permet d'avoir un partage des fonctionnalités et facilite grandement le développement, leur internet majeur et l'instauration :

- L'interopérabilité des applications.
- L'intégration des applications sur le Web.

I.10. Composition de Web services

La composition ou l'agrégation de services Web est une opération qui consiste à construire de nouvelles applications ou services appelés services composites ou agrégats par assemblage de services déjà existants nommés services basiques ou élémentaires.

La composition spécifie quels services doivent être invoqués, dans quel ordre et sous quelles pré-conditions.

Les services basiques peuvent être soient des services atomiques soient des services composites.

La composition de services Web vise essentiellement quatre objectifs [5] :

- Créer de nouvelles fonctionnalités en combinant des services déjà existants.
- Résoudre des problèmes complexes auxquels aucune solution n'a été trouvée.
- Faire collaborer plusieurs entreprises ensemble.
- Optimiser et améliorer une fonctionnalité existante.

I.10. 1. Définition de composition de Web services

Une composition de services Web est constituée de plusieurs services qui interagissent les uns avec les autres afin d'offrir de nouvelles fonctionnalités qu'un seul service ne pourrait pas les offrir.

La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités. Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers. [6]

I.10. 2. Les types de composition de Web services

La composition des Web services peut être soit une composition statique soit une composition dynamique:

I.10. 2. 1. La composition statique :

Est appelée aussi composition off-line, précompilée ou encore proactive. C'est une composition qui utilise des services basiques qui sont au préalablement définis d'une façon figée et qui ne peuvent pas changer en fonction du contexte du client.

Ce type d'application est celui qui est aujourd'hui le plus utilisé, en particulier pour les industriels. Il existe deux visions de la composition statique qui sont l'orchestration et la chorégraphie.

- ✓ **L'orchestration :** aborde le problème de façon centralisée, ou les collaborations de Web Service statiques sont contrôlées par le service composé, tel un chef d'orchestre qui se charge d'ordonner les appels aux services Web et de rattraper les erreurs.
- ✓ **La chorégraphie** : aborde le problème de façon distribuée, chaque partenaire d'une composition, i.e. chaque fournisseur de Web service, peut réaliser une ou plusieurs tâches, chacun d'eux communicants à l'aide de Web service.

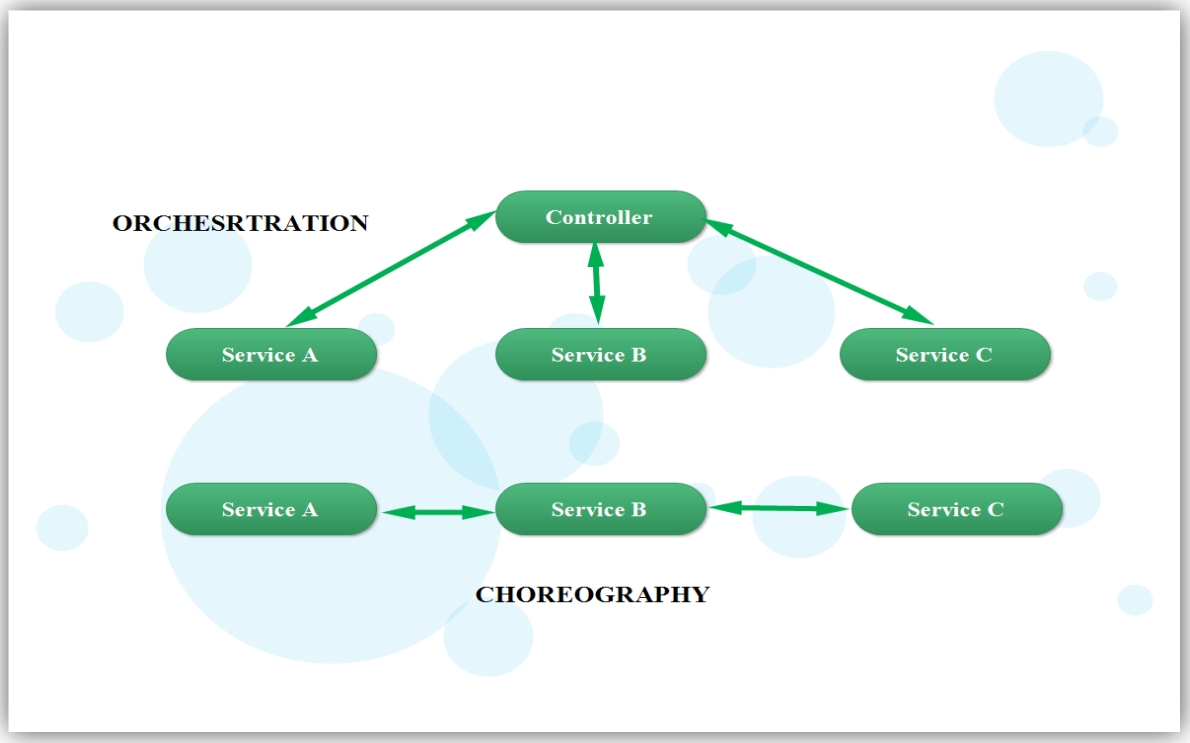

Figure.1. 3 : Orchestration et chorégraphie des services.

I.10. 2.2. La composition dynamique :

Appelée aussi composition on-line. La sélection des services basiques ne peut pas être prédéfinie à l'avance mais elle sera faite au moment de l'exécution en fonction des contraintes imposées par le client. Ceci permet d'élaborer des différents scénarios de composition qui offrent les mêmes fonctionnalités et qui tiennent compte de la dynamité de la situation du client. [5]

I.11. Les technologies des Web services

La majorité des grands sites web (Amazon, eBay…) qui proposent des Web services aux développeurs, offrent simultanément deux catégories de Web services.

- ***Les services web REST :*** exposent entièrement ces fonctionnalités comme un ensemble de ressources (URI) identifiables et accessibles par la syntaxe et la *sémantique* du protocole HTTP,
- ***Les services web SOAP :*** exposent ces mêmes fonctionnalités sous la forme de services exécutables à distance. Un web service est composé de quatre grandes parties comme la figure1.4, la description de l'interface du web service grâce au langage WSDL la sérialisation des messages transmis via le protocole SOAP, l'indexation des services Web dans des registres UDDI et la sécurité des web services, obtenue essentiellement grâce à des protocoles d'authentification et de cryptage XML.

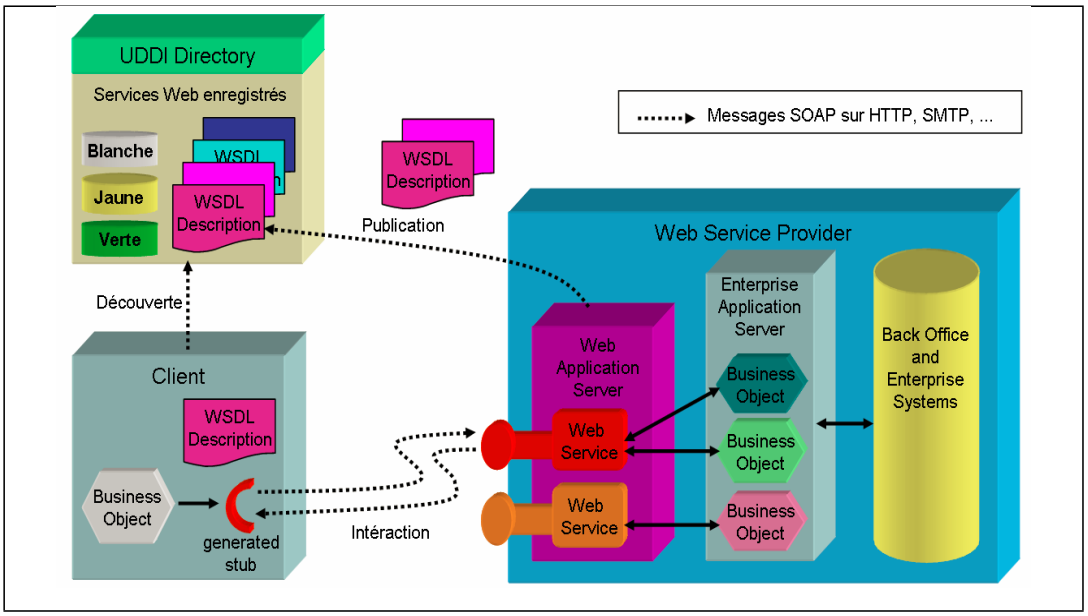

Figure .1. 4 : Les technologies des Web services

I.11.1. SOAP (Simple Object Access Protocol) [7].

Le protocole SOAP (Simple Object Access Protocol) est un protocole de communication base sur XML qui permet aux services Web d'échanger des informations dans un environnement décentralise et distribue tout en s'affranchissant des plates-formes et des langages de programmation utilises. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux *services web*. C'est le rôle du langage WSDL.

Un message SOAP est un document XML ordinaire qui contient les éléments suivants comme la figure I.5.

- **L'élément Enveloppe :** qui identifie le document XML comme étant un message SOAP
- **L'élément Header :** qui est optionnel et qui contient des informations d'entête
- **L'élément Body :** qui contient l'appel ainsi que la réponse retournée
- **L'élément Fault** : qui est optionnel et qui fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message .

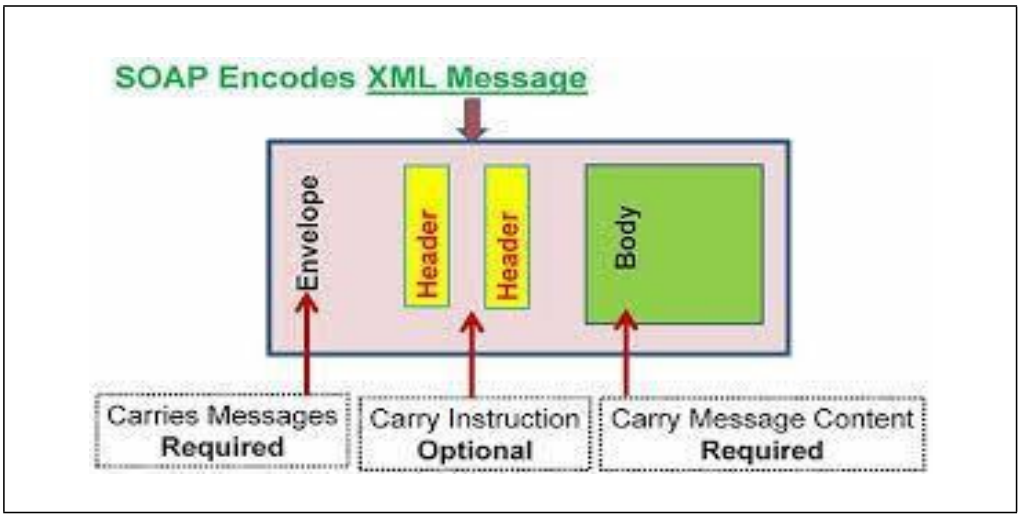

Figure .1.5: structure d'une enveloppe SOAP

I.11. 2. WSDL (Web Service Description Language) [8]

WSDL, acronyme de Web Services Description Langage est un standard XML proposé par le W3C, utilisé pour décrire les Web Services et pour permettre aux clients de savoir comment accéder au service. Le fichier de description WSDL d'un Web Service contient un ensemble de définitions décrivant :

- l'interface du service c'est à dire les opérations que le service fournit, les formats des données et les protocoles utilisés,
- l'implémentation du service c'est `à dire l'adresse applicative (URL ou URI) pour accéder au service.

Il permet de générer des documents structurés en deux parties : une partie abstraite (le Quoi) décrivant l'interface fonctionnelle du service en termes d'opérations et de messages, ainsi qu'une partie concrète (le Comment) qui contient les détails des protocoles à utiliser et de l'adresse physique des opérations. En particulier, un port type désigne une collection d'opérations, un binding consiste en une association entre un port type ainsi qu'un protocole de transport et de format de données, un port définit l'adresse physique d'un binding, et un service constitue une collection de ports. [9]

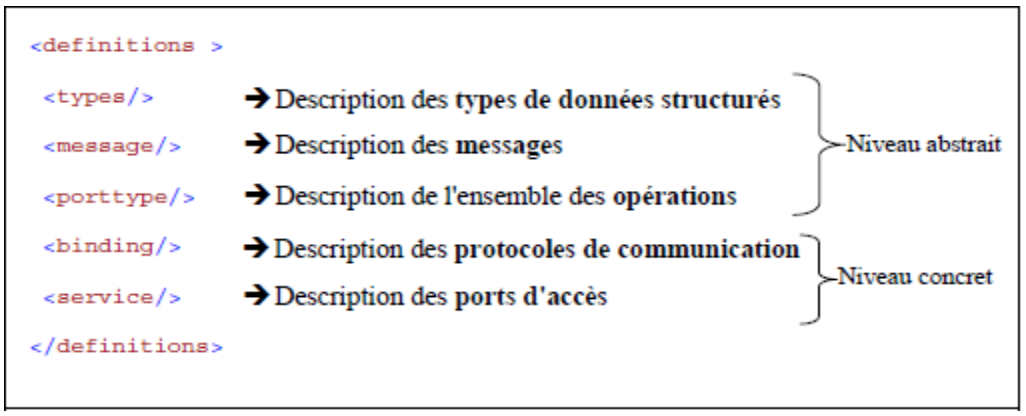

Figure .1. 6**:** Langage WSDL

I.11.3. UDDI (Universal Description, Discovery and Integration) [10].

L'UDDI définit les mécanismes permettant de répertorier des services web. Ce standard gère donc, l'information relative à la publication, la découverte et l'utilisation d'un web service. En clair, l'UDDI définit un registre des services web sous un format XML. Les organisations publient les informations décrivant leurs services web dans l'annuaire, et l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le web service qui fournit le service désiré ; ainsi l'UDDI a été créé pour faciliter la découverte de services web en plus de leurs publications.

L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP .L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte.

La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement.

La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP. L'UDDI peut être vu comme un annuaire contenant :

- ***Les pages blanches*** **:** noms, adresses, identifiants et contacts des entreprises enregistrées.
- ***Les pages jaunes*** **:** donnent les détails sur le métier des entreprises et les services qu'elles proposent.
- ***Les pages vertes*****:** contiennent des informations techniques du service offert, la manière d'interagir avec le service, et aussi un pointeur vers le fichier WSDL et une clé unique identifiant le service.

I.11.4. Langage XML (eXtensible Markup Language) est une famille de technologies développées au sein du W3C.

XML est largement utilisé par les entreprises et supporté par les manufacturiers informatiques. Il est indépendant des plates-formes informatiques. Il est lisible par l'humain mais est destiné à être lu par la machine.

Les web services communiquent via XML dans le but de décrire leurs interfaces et encoder leurs messages. Des documents sortant bruts de fonderie des traitements de texte ou des tableurs, le format XML a été conçu pour permettre le déchiffrement direct par l'utilisateur comme par les programmes. XML est un texte contenant des balises ouvrantes et fermantes qui décrivent la nature des données qu'elles encadrent. En plus, les protocoles de communication dans les Web services utilisent les interfaces et les messages d'XML, que n'importe quelle application peut interpréter, pour définir les interactions. [11]

I.12. Avantages et inconvénients des Web services

I.12. 1.Avantage des Web services [11].

L'idée fondamentale derrière des services web est de morceler les applications et les processus en segments réutilisables appelés service de telle sorte que chacun de ces segments effectuent une tache distincte. Ces services peuvent alors servir à l'intérieur et à l'extérieur de l'entreprise, facilitant l'interopérabilité entre tous ces services.

Par leur nature, les Web services :

- ✓ Permettent à des portions de logiciels écrits dans différents langages, ou évoluant sur différents systèmes d'exploitation, de communiquer entre elles facilement et à peu de frais.
- ✓ Permettent à des applications supportant différents processus d'une organisation ou de différentes organisations, de communiquer entre elles et /ou d'échanger des données facilement et à peu de frais.

Plus spécifiquement, les Web services devraient permettre aux entreprises de :

- ✓ Donner aux clients un accès direct à l'information, aux données et aux Fonctionnalités dont ils ont besoin pour interagir avec une entreprise;
- ✓ Donner aux partenaires d'une entreprise un accès direct à la fonctionnalité dont ils besoin pour mieux servir les clients qu'ils ont en commun avec cette entreprise;
- ✓ Donner aux fournisseurs d'une entreprise un accès direct à l'information et à la fonctionnalité dont ils ont besoin pour leur permettre d'ajuster les inventaires

I.12. 2. Inconvénients

La technologie des services web comporte plusieurs inconvénients dont:

- ✓ Problèmes de sécurité: Il est facile de contourner les mesures de sécurité mises en place par les pare-feu où l'utilisation du protocole HTTP.
- ✓ Confiance: Les relations de confiance entre différentes composantes d'un web service sont difficiles à bâtir, puisque parfois ces mêmes composantes ne se connaissent même pas.
- ✓ Syntaxe et sémantique: On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (la sémantique).
- ✓ Disponibilité: Les services web peuvent bien satisfaire un ou plusieurs besoins du client. Seront-ils pour autant toujours disponibles et utilisables ? Ça reste un défi pour les services web.

I.13. web Service sémantique

I.13.1. Définition de Web services sémantiques

Les services Web sémantiques sont des services Web décrits de telle sorte qu'un client (applications ou autres services Web) peut interpréter leurs fonctionnalités offertes. Pour permettre cela, la description syntaxique du Web service doit être augmentée en information sémantique et exploitable par machine (application ou service) et cela fait par la technologie d'ontologique. [9]

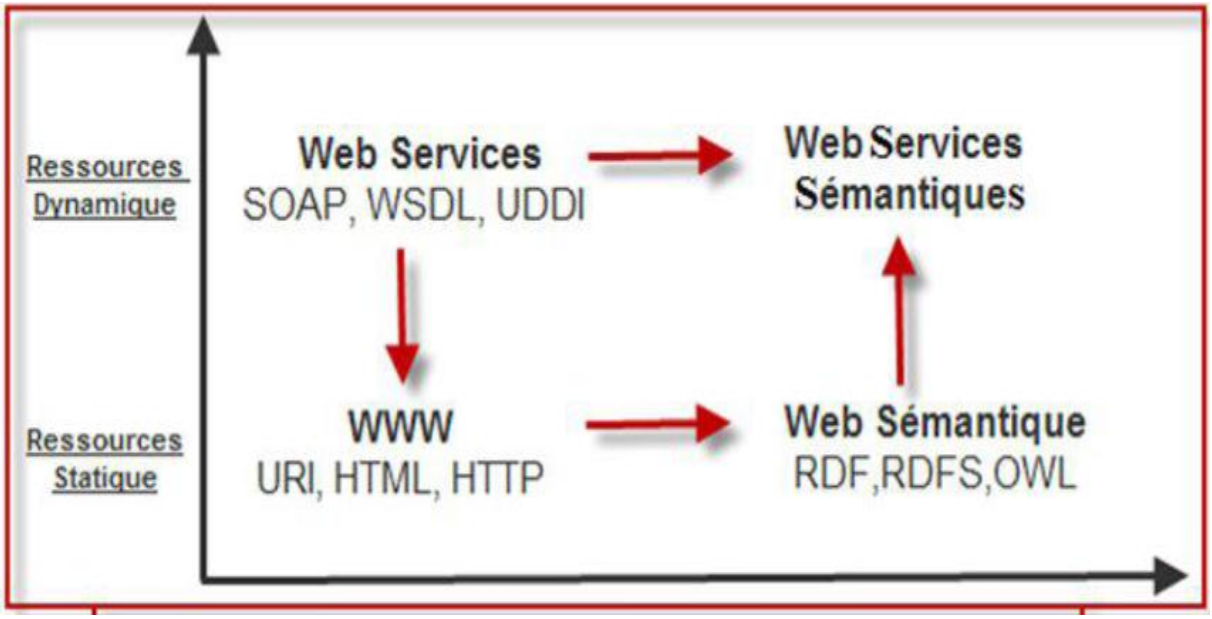

Figure 1.7: Origine des Web services sémantiques.

I.13.2. Objectifs des services Web sémantique

L'objectif visé par la notion de Web services sémantiques est de créer un Web sémantique de services dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines et ce en utilisant les couches techniques sans pour autant en être conceptuellement dépendants. La sémantique ainsi exprimée permettra l'automatisation des fonctionnalités suivantes:

- Processus de description et de publication des services
- Découverte des services
- Sélection des services
- Composition des services
- Fourniture et administration des services

I.13.3. Langages de description de services Web sémantiques

Ils existent des travaux qui proposent une manière de représenter de façon sémantique des services web (autrement dit proposent de décrire des services web sémantiques). Malgré l'abondance de ce type de travaux, aucun ne s'est imposé comme une solution de description de services web sémantiques. OWL-S est l'un des travaux issus du W3C qui tentent d'apporter une solution standard en termes de description de services web sémantiques [12]

I.13.3. 1. OWL-S [12].

Le but initial du langage OWL-S est de mettre en œuvre des services web sémantiques. Cette mise en œuvre inclut un grand nombre d'objectifs, rendus possibles par le biais de l'expressivité héritée d'OWL et de l'utilisation de la logique de description. Ces objectifs sont :

- la description de services Web sémantiques,
- l'invocation automatique de ces services, par le biais de la détection et de l'interprétation automatique de la localisation et des paramètres d'entrée/sortie.
- la composition automatique de services (description et invocation) et la surveillance de l'exécution de la composition.

I.13.3.2. WSMO

L'architecture WSMO (Web Services Modeling Ontology) est une architecture conceptuelle visant à expliciter la sémantique des services web. [12].

Conclusion

Un " Web Service " est une application logicielle à laquelle on peut accéder à distance à partir de différents langages basés sur XML.

De nombreuses normes sous-tendent cette architecture : "SOAP" pour l'échange de messages, "XML" langage de base pour décrire tous les documents sur lesquels les messages sont construits, "HTTP" pour transporter les messages, "WSDL" pour décrire les services et enfin "UDDI" pour les publier, Nous avons vu aussi les avantages et les inconvénients des Web services.

Chapitre 02

Les ontologies

Introduction

La modélisation et la représentation du monde réel constituent un sujet très riche à explorer. Le besoin de représenter les connaissances d'un domaine spécifique d'une manière à travers laquelle les machines puissent manipuler la sémantique des informations a donné naissance aux ontologies. L'ontologie est un élément-clé de la résolution du problème de l'hétérogénéité sémantique, permettant ainsi l'interopérabilité sémantique entre les différentes applications Web et des services.

Dans ce chapitre nous allons présenter les définitions d'une ontologie, leurs différents types et composantes d'une ontologie, et utilisation des ontologies, et Quelques méthodologies de construction d'ontologies...

II.1. Un peu d'histoire

Le mot ontologie est construit à partir des racines grecques : ontos qui veut dire « ce qui existe », « l'existant », et logos pour « le discours », « l'étude ». En d'autres termes, Ontologie signifie l'étude de ce qui existe, la science de l'être. Pour tenter de définir cette notion d'ontologie, il n'est pas inutile de revenir brièvement sur les origines purement philosophiques de la notion et sur son histoire afin de comprendre pourquoi l'ingénierie des connaissances a choisi de réutiliser un mot préexistant pour désigner l'un de ses objets.

II.2. Notion d'ontologie

Afin de faciliter le partage et la réutilisation de connaissances formellement représentées dans des systèmes d'intelligence artificielle, il est très utile de définir un vocabulaire commun dans lequel la connaissance partagée est représentée. La spécification de ce vocabulaire est communément appelée une ontologie. De ce fait, les ontologies définissent actuellement des vocabulaires structurés, regroupant des concepts utiles d'un domaine et de leurs relations et qui servent à organiser et échanger des informations de façon non ambiguë. Leur développement progressif en (IA) Intelligence Artificielle parvient de leur intérêt, pour associer la sémantique à des ressources ou bien à des entités textuelles, pour faciliter la localisation et la gestion des connaissances dans diverses applications [13].

II.3. Définitions d'une ontologie

Le terme « ontologie » est employé dans des contextes très différents touchant la philosophie, la linguistique ou l'IA. De nombreuses définitions ont été offertes pour donner un éclaircissement sur ce terme, mais aucune de ces définitions ne s'est explicitement imposée. Les définitions de ce terme ne sont pas toujours consistantes et cela dépend des domaines spécifiques [14]

Pour ne pas dévier de notre propos, nous avons recensé les définitions suivantes :

Définition1 « *une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire* » [15]

Définition2 « *une ontologie est une spécification explicite d'une conceptualisation*».[16]

Définition3 *« une ontologie est une spécification explicite et formelle d'un conceptualisation partagée »* [17]

Le terme « conceptualisation » réfère dans cette définition à une abstraction d'un phénomène du monde, obtenue en identifiant les concepts appropriés à ce phénomène. Le terme « formelle » indique que les ontologies sont interprétables par la machine. Cependant, « spécification explicite »

Signifie que les concepts de l'ontologie et les contraintes liées à leur usage sont définis de façon déclarative. Enfin, le terme «partagé» signifie que l'ontologie capture la connaissance consensuelle. Mais cette définition laisse la porte ouverte à de nombreuses définitions.

Définition4 *« une ontologie est une description formelle d'entités et leurs propriétés, relations, contraintes, comportement »* [18]

Définition5 *« Les ontologies sont des spécifications partielles et formelles d'une conceptualisation commune »* [19]

En 1997, Guarino accentue l'ambigüité du terme conceptualisation qui doit être pris dans son sens intuitif. La spécification des ontologies est partielle, car une conceptualisation ne peut pas toujours être entièrement formalisée dans un cadre logique, du fait d'ambiguïtés ou du fait qu'aucune représentation de leur sémantique n'existe dans le langage de représentation d'ontologies choisi. « Commune » renvoie à l'idée qu'une ontologie rend compte d'un savoir consensuel, c'est-à-dire qu'elle n'est pas l'objet d'un individu, mais qu'elle est reconnue par un groupe [20]

II.4. Composantes d'une ontologie

Comme tout formalisme de représentation, les ontologies sont basées sur l'utilisation d'un certain nombre de composantes (dites aussi briques ou constituants) de base, véhiculant avec eux les connaissances traduites par ces dernières et qui sont principalement : Concept, Relation, Fonction, Axiomes, Instance.

II.4.1. Concepts :

Aussi appelés termes ou classes de l'ontologie, constituent les objets de base manipulés par les ontologies. Ils correspondent aux abstractions pertinentes du domaine du problème, retenues en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie, par exemple, la description d'un ensemble d'objets, d'une tâche, d'une fonction, d'une stratégie, d'un processus de raisonnement, etc. [22]

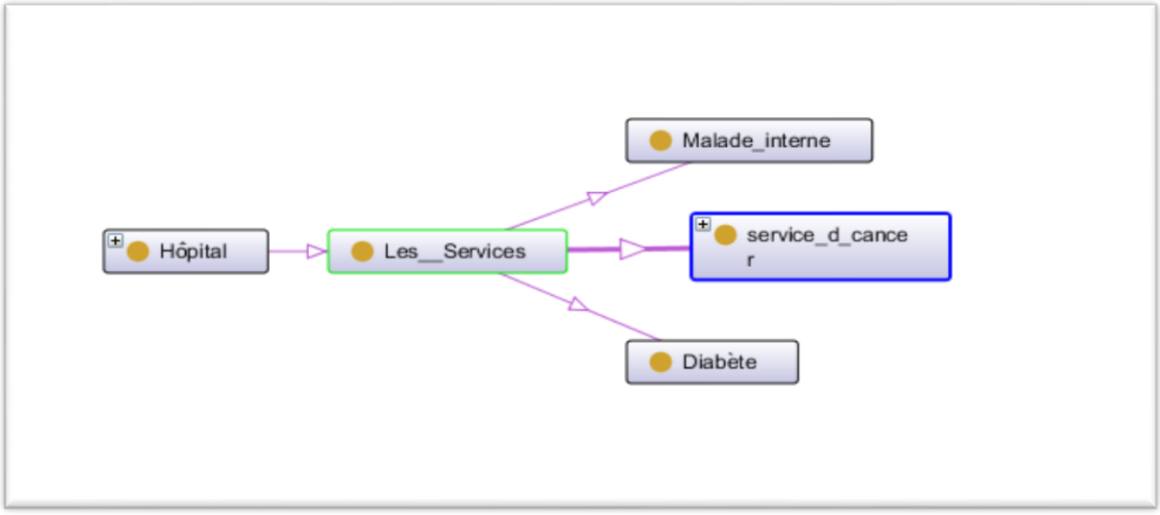

Figure 2.8: exemple de concept : super-classe et sous-classe

II.4.2. Relations :

Traduisent les interactions existant entre les concepts présents dans le domaine ciblé. Ces relations sont formellement définies comme tout sous ensemble d'un produit cartésien de n ensembles, c'est à dire R : C1 x C2x … x Cn et incluent 1- la relation de spécialisation (subsomption), 2- la relation de composition (méronymie), 3- la relation d'instanciation, etc. Ces relations nous permettent de capturer, la structuration ainsi que l'interaction entre les concepts, ce qui permet de représenter une grande partie de la sémantique de l'ontologie**.** [22]

II.4.3. Fonctions :

Ce sont des cas particuliers de relations dans lesquelles le Nième élément de la relation est défini de manière unique à partir des n-1 premiers. Formellement, les fonctions sont définies ainsi : F: C1× C2× … ×Cn-1à Cn. Comme exemple de fonctions binaires, nous avons la fonction mère de et le carré, et comme exemple de 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. [22]

II.4.4. Axiomes :

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. [22]

Leur inclusion dans une ontologie peut avoir plusieurs objectifs :

- Définir la signification des composants.
- Définir les arguments d'une relation.

II.4.5. Les instances (ou individus)

Constituent la définition extensionnelle de l'ontologie. Ils représentent des éléments singuliers véhiculant les connaissances (statiques, factuelles) à propos du domaine du problème. [22]

II.5. Typologie des ontologies

Nous listons ci-dessous les différents types d'ontologies les plus utilisées:

II.5.1. Les ontologies de représentation

N'appartiennent à aucun domaine, mais définissent et organisent les primitives de la théorie logique pour permettre la représentation des ontologies. L'exemple le plus représentatif de ce genre d'ontologie est la Frame Ontologie, qui définit d'une manière formelle, les primitives de représentation (classes, sous classes, attributs, valeurs, relations et axiomes) dans un environnement implémentant les langages de Frame. [23]

II.5.2. Les ontologies génériques

Sont aussi appelée Ontologie de haut niveau ou ontologie Top, elles décrivent des concepts généraux, indépendants d'un domaine ou d'un problème particulier. Elles permettent par exemple de formaliser les aspects temporels ou spatiaux des objets du monde réel. est un exemple d'une ontologie générique portant sur des concepts de haut niveau. Ces dernières décrivent des notions générales comme les notions d'objet, de propriété, d'état, de valeur, de moment, d'évènement, d'action, de cause et d'effet. [24] [25] [26]

II.5.3. Les ontologies de domaine

Elles sont construites sur un domaine particulier de la connaissance. Les ontologies de domaine fournissent des vocabulaires au sujet des concepts dans un domaine et leurs relations au sujet des activités qui ont lieu dans ce domaine, et au sujet des théories et des principes élémentaires régissant ce domaine. Plusieurs ontologies de domaines existent déjà, telle que MENELAS dans le domaine médical. Entreprise est un autre exemple décrivant le domaine de l'entreprise. [23]

II.5.4. Les ontologies de tâches

L'ontologie de tâche décrit les connaissances portant sur taches et/ou des activités particulières. Ces ontologies fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles incluent des noms génériques (objectif, contrainte…), des verbes génériques (classer, sélectionner,…) des adjectifs génériques (assigné,..) et autres dans les descriptions de tâches. [25]

II.5.5. Les ontologies d'application

Aussi appelée ontologie de domaine-tache : Ce sont les ontologies les plus spécifiques, elles contiennent les connaissances requises pour une application particulière permettant ainsi de modéliser une activité spécifique dans un domaine donné. [23]

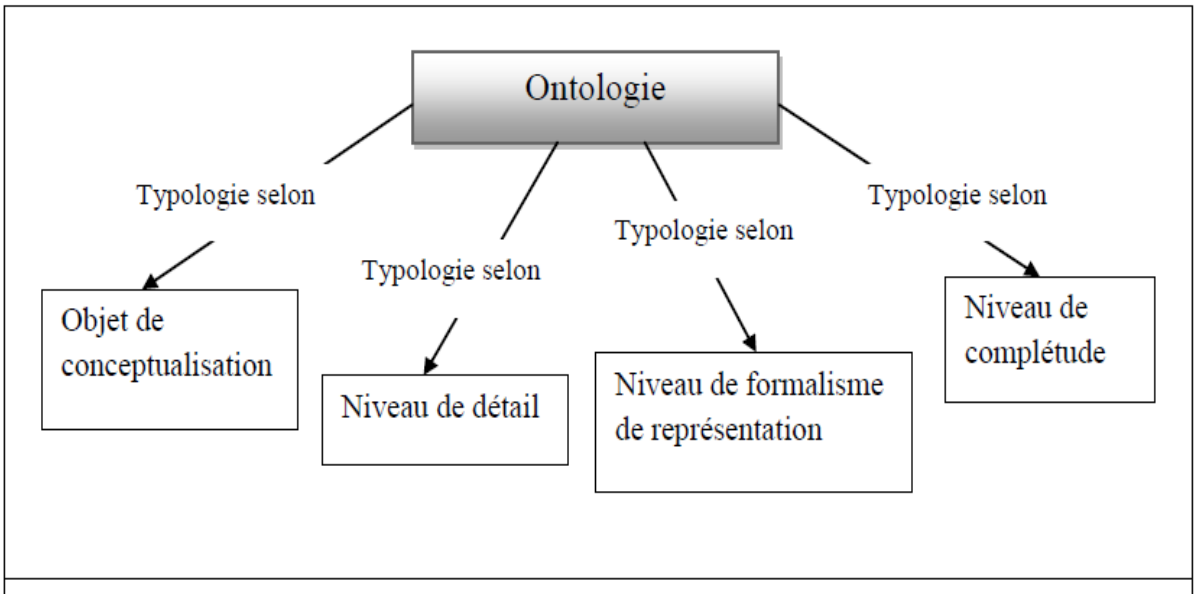

Figure 2. 9: Typologies d'ontologies

II.6. Utilisation des ontologies

Même si le besoin de développer une ontologie est très varié et dépend du domaine d'application, on peut facilement énumérer un certain nombre d'utilités, notamment:

II.6.1. La connaissance du domaine :

Les ontologies permettent la modélisation des connaissances dans un domaine particulier, dans lequel opère le système à développer. [26]

II.6.2. La communication:

Les ontologies assurent une communication fiable et hétérogène entre personnes et machines (agents logiciels ou organisations) du fait qu'elle permet de mettre en place un langage ou un vocabulaire conceptuel commun. [26]

II.6.3. L'interopérabilité :

La représentation explicite des connaissances dans un domaine donné sous forme d'une ontologie, permet à son tour une plus grande réutilisation, un partage plus large et une interopérabilité plus étendue. [26]

II.6.4. L'aide à la spécification des systèmes:

La représentation conceptuelle des éléments du domaine, permet aux systèmes de réaliser des raisonnements logiques qu'on appelle inférences, et de sortir avec des conclusions capables d'aider l'utilisateur ou le gestionnaire dans ses décisions. [26]

II.6.5. L'indexation et la recherche d'information:

Dans le web sémantique, d'une façon générale, dans certains cas en particulier, les ontologies sont utilisées pour indexer et décrire les ressources utilisées. Cela permet une plus grande précision dans les résultats des recherches ou d'assignation des ressources. [26]

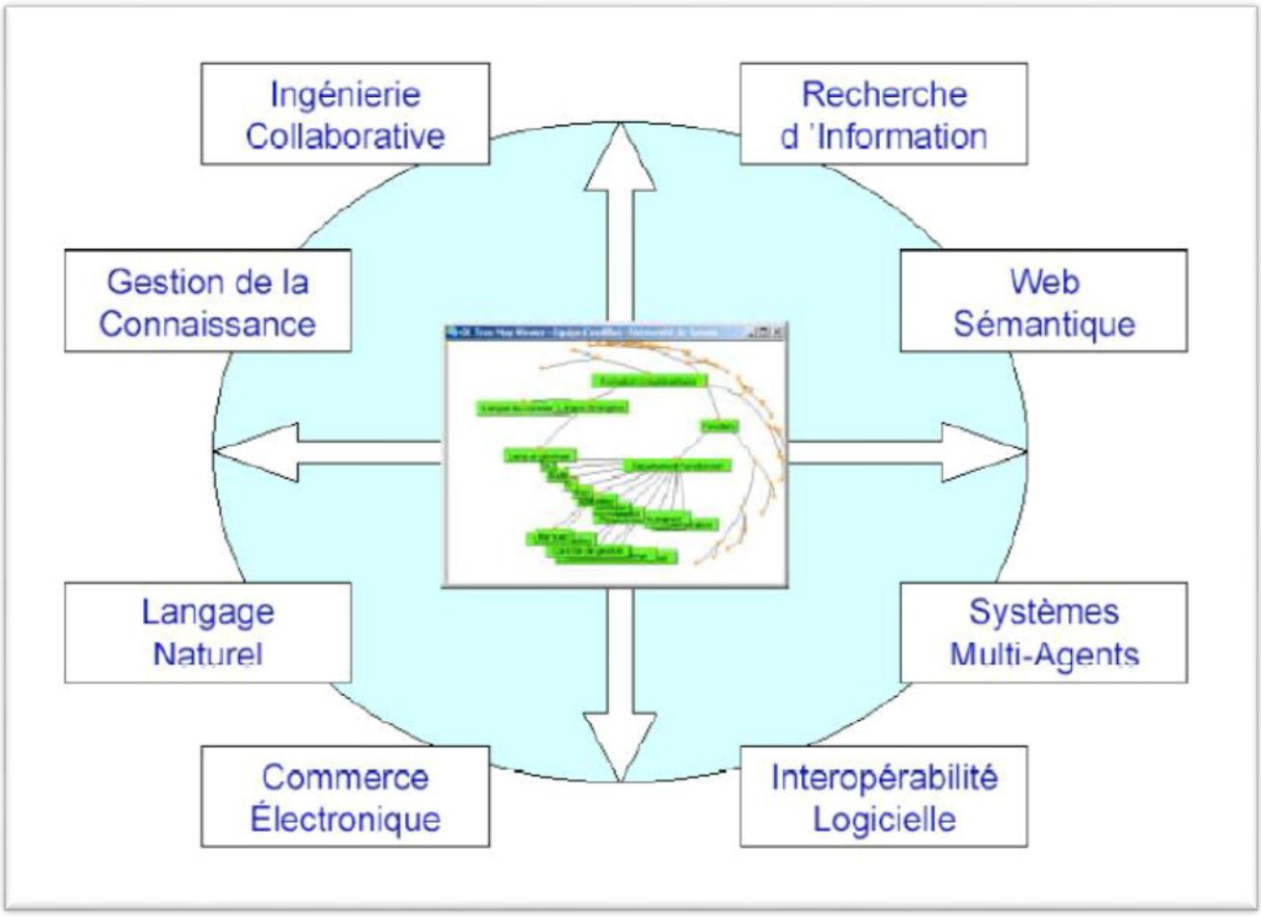

Figure 2.10: Quelque utilisation des ontologies.

II.7. Construction d'une ontologie

II.7.1. Un squelette de méthodologie pour construire des ontologies

Le processus de construction d'une ontologie est une collaboration qui réunit des experts du domaine de connaissance, des ingénieurs de la connaissance, voire les futurs utilisateurs de l'ontologie. Cette collaboration ne peut être fructueuse que si les objectifs du processus ont été clairement définis, ainsi que les besoins qui en découlent. La figure ci-dessous représente le processus de construction d'ontologie.

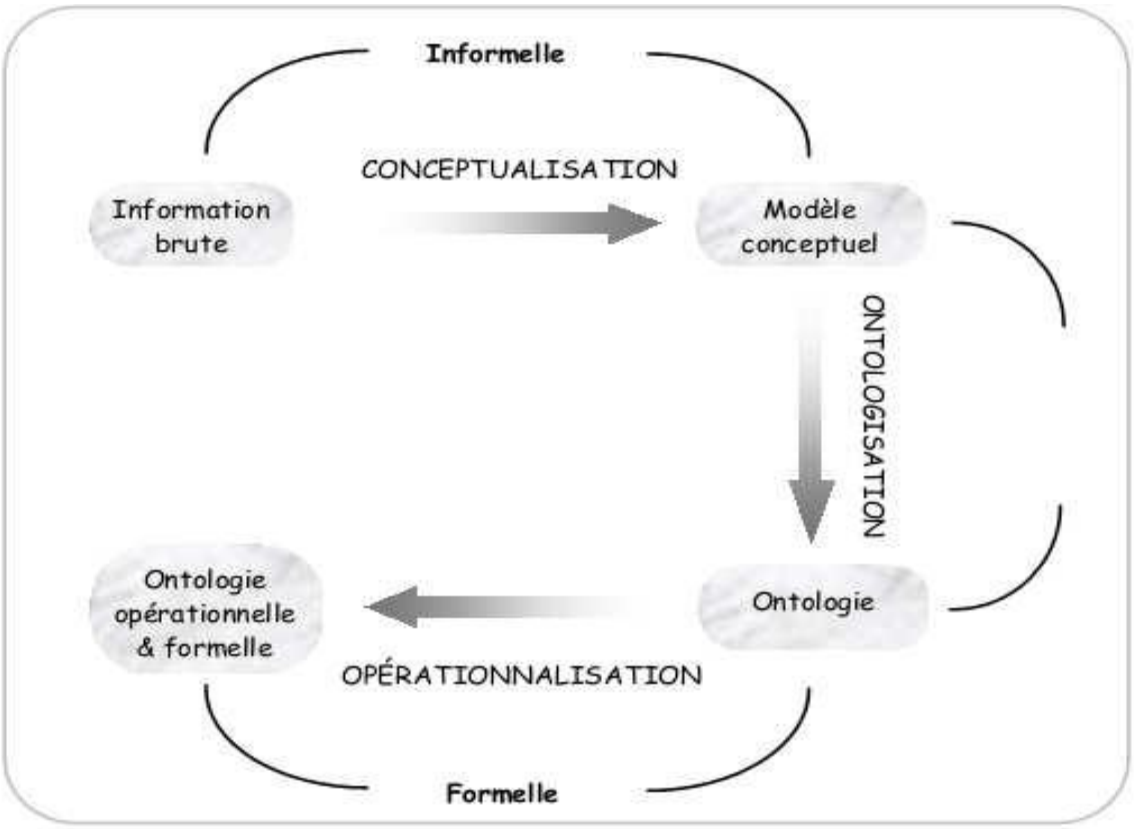

Figure 2.3: Processus de construction d'ontologie [28].

II.7.1.1.Evaluation des besoins

Le but visé par la construction d'une ontologie se décline en 3 aspects:

- L'objectif opérationnel : il est indispensable de bien préciser l'objectif opérationnel de L'ontologie, en particulier à travers des scénarios d'usage.
- Le domaine de connaissance : il doit être délimité aussi précisément que possible.
- Les utilisateurs : ils doivent être identifiés autant que faire se peut, ce qui permet de choisir, en accord avec l'objectif opérationnel, le degré de formalisme de l'ontologie, et sa granule Une fois le but défini, le processus de construction de l'ontologie peut démarrer, en commençant par la phase de conceptualisation.

II.7.1.2.Conceptualisation

Cette étape permet d'aboutir à un modèle informel, donc sémantiquement ambiguë et généralement exprimé en langage naturel. Elle consiste, à partir des données brutes, à dégager les concepts et les relations entre ces concepts permettant de décrire de manière informelle les entités cognitives du domaine.

L'objectif est d'aboutir à un modèle conceptuel : le modèle obtenu consiste en un ensemble de termes désignant les entités du domaine de connaissances (concepts, relations, propriétés des concepts et des relations, …), assortis d'informations exprimant leur sémantique. La découverte des connaissances d'un domaine peut s'appuyer à la fois sur l'analyse de documents et sur l'interview d'experts du domaine. Ces activités doivent être raffinées au fur et à mesure que la conceptualisation émerge.

II.7.1.3.Ontologisation.

L'Ontologisation consiste en une formalisation partielle, sans perte d'information, du modèle conceptuel obtenu dans l'étape précédente. Ce qui permet de faciliter sa représentation ultérieure dans un langage complètement formel et opérationnel.

Elle effectue une transcription des connaissances dans un certain formalisme de connaissances, ce formalisme devant être aussi générique que possible, mais sémantiquement clair. S'imposer de conserver toutes les connaissances conduit à intégrer, à l'ontologie du domaine, des connaissances qui ne peuvent être formalisées, ou dont la sémantique est ambiguë.

Le modèle obtenu est souvent qualifié de semi-formel (car certaines connaissances ne peuvent pas être totalement formalisées). Le caractère semi-formel d'une ontologie lui

interdit d'être utilisée telle quelle dans un SBC. En revanche, une ontologie, contenant toutes les connaissances d'un domaine, constitue le support idéal de communication et de partage des connaissances de ce domaine.

II.7.1.4.Opérationnalisation

Cette étape consiste à formaliser complètement l'ontologie obtenue dans un langage de représentation de connaissances formel (i.e. possédant une syntaxe et une sémantique) et opérationnel (i.e. doté de services différentiels permettant de mettre en œuvre des raisonnements), par exemple, le modèle des Graphes Conceptuels ou la Logique de Descriptions.

On obtient alors une représentation formelle des connaissances du domaine. Ainsi, le caractère formel de l'ontologie permet à une machine, via cette ontologie, de manipuler des connaissances du domaine. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie.

II.7.2. Quelques méthodologies de construction d'ontologies

Les méthodologies peuvent porter sur l'ensemble du processus et guider l'otologiste dans toutes les étapes de la construction. Bien qu'aucune méthodologie générale n'ait pour l'instant réussi à s'imposer, de nombreux critères de construction d'ontologies ont été proposés pour des méthodologies. ENTERPRISE, TOVE et METHONTOLOGY sont les méthodologies les plus représentatives pour construire des ontologies.

II.7.2.1.Tove

TOVE (Toronto Virtual Enterprise) développé par l'université de Toronto, cette méthodologie repose sur les expériences de développement d'une entreprise [29] [30].

Elle s'appuie également, pour le développement d'une ontologie, sur les principales étapes suivantes :

- ✓ Capturer des scénarios de motivations : Cette étape consiste à identifier des scénarios qui clarifient le domaine que l'on investit et les différentes applications dans lesquelles l'ontologie sera employée.
- ✓ Formuler des questions de compétences informelles : Cette étape consiste à formuler un ensemble de questions (basées sur les scénarios), exprimées en langage naturel, afin de déterminer la portée de l'ontologie. Ces questions et leurs

réponses sont utilisées pour extraire les concepts principaux, leurs propriétés et les relations qui existent entre ces concepts.

- ✓ Spécifier la terminologie de l'ontologie : Cette étape consiste à représenter les termes (Concepts, propriétés et relations), identifier dans l'étape précédente, en utilisant le formalisme de la logique du premier ordre. Les concepts seront représentés sous forme de constantes ou bien des variables. Par ailleurs, les propriétés et les relations seront représentées par des prédicats.
- ✓ Evaluer la complétude de l'ontologie.

II.7.2.2 Enterprise [31]

Proposent le squelette d'une méthode basé sur l'expérience de construction d'ontologies dans le domaine de la gestion des entreprises. La méthode ENTERPRISE repose sur les trois étapes suivantes :

- ✓ Identifier le rôle et la portée de l'ontologie, Dans cette étape, l'ontologie est réellement construite. Les activités suivantes sont distinguées : identifier les concepts et relations fondamentaux et des définitions provisoires de ces éléments, coder l'ontologie dans un langage adapté, intégrer des ontologies existantes,
- ✓ Evaluer l'ontologie,
- ✓ Rédiger une documentation et une trace des actions réalisées lors des différentes phases. Les étapes et sous-tâches de la méthode ENTERPRISE, sont décrites de façon abstraite. Les techniques utilisées pour les sous-tâches ne sont pas précisées (par exemple : Comment identifier les concepts fondamentaux ? Quel langage utiliser pour représenter l'onto

II.7.2.3. Methontology

La méthodologie de construction d'ontologies «METHONTOLOGY» se situe entre le GL (Génie Logiciel) 3et l'IC (Ingénierie des Connaissances). Elle identifie une séquence d'activités techniques à appliquer pour le développement de l'ontologie. Cette méthodologie a été motivée par le constat suivant : l'absence de méthodes ou de guides structurés est un obstacle à la construction d'ontologies partagées et consensuelles.

Il est également un obstacle logiez ? à l'extension d'une ontologie existante ou à sa réutilisation dans d'autres ontologies.

L'approche METHONTOLOGY distingue les étapes suivantes:

a. Spécification

Le développement d'une ontologie commence par la définition du domaine et du porté de celle-ci. Cela est basé sur la réponse à certaines questions : Quel est le domaine que l'ontologie va couvrir ? À quoi cette ontologie va servir ? À quels types de questions les informations de l'ontologie doivent fournir des réponses ? Qui va utiliser et maintenir l'ontologie ?, etc. Les réponses à ces questions peuvent changer durant le processus de développement de l'ontologie, mais à chaque étape, elles permettent de limiter la portée du modèle. L'une des solutions qui permet de déterminer la portée d'une ontologie consiste à définir ou planifier une liste de questions auxquelles une base de connaissance, basée sur l'ontologie, doit être capable de répondre («competency questions ») [29].

b. Conceptualisation

Elle consiste à identifier et à structurer les connaissances du domaine, à partir des sources d'informations. L'acquisition de ces connaissances peut s'appuyer à la fois sur l'analyse de documents et sur l'interview des experts du domaine. Une fois que les concepts sont identifiés par leurs termes, leur sémantique est décrite dans un langage semi -formel (tables et graphes) à travers leurs propriétés, leurs instances connues et les relations qui les lient entre eux..

c. Implémentation

Cette étape consiste à formaliser le modèle conceptuel obtenu dans l'étape précédente par un formalisme de représentation d'ontologie telles que les logiques de description. Puis, à coder l'ontologie dans un langage d'ontologie formel.

d. Maintenance

Cela peut s'agir d'une maintenance corrective ou évolutive de l'ontologie (nouveaux besoins de l'utilisateur), ce qui permet la validation et l'évolution de celle-ci. Cette activité est généralement faite par le constructeur et des experts du domaine. La validation se base sur l'exploitation des services d'inférences associés aux LDs, et qui sont offerts par des raisonneurs.

Pour conclure, nous avons constaté que la démarche METHONTOLOGY présente un certain nombre de phases spécifiées de manières très détaillées, notamment la phase de conceptualisation. De ce fait, nous allons adopter cette méthodologie et l'adapter pour les besoins de notre travail.

II.8. Langages de description d'ontologie

II.8.1 SHOE ("Simple HTML Ontology Extension"), basé sur du web il combine les "frames" et les règles. Il a été construit comme une extension de HTML en 1996. Il utilisait des étiquettes différentes de celles des spécifications HTML, de ce fait permettant l'insertion des ontologies dans des documents HTML. Plus tard sa syntaxe a été adaptée à XML.

II.8.2. XOL ("XML-based Ontology exchange Language") développé comme une transformation en XML d'un petit sous ensemble des primitives du protocole OKCB, appelé OKBC-Lite. [23]

II.8.3. RDF Le besoin d'un modèle conceptuel devient nécessaire pour la description de chaque ressource .D'où l'introduction de RDF pour Resource Description Framework qui est un modèle conceptuel, abstrait et formel, fondé sur un modèle de graphe de ressources, permettant de décrire les éléments simplement et sans ambiguïté selon un mécanisme basé sur des déclarations RDF. [33]

Une déclaration RDF est une phrase composée d'un triplé <Sujet, Prédicat, Objet> qui peut être traitée par la machine pour permettre à celle-ci de le faire tout en comprenant la signification de ce triplet. Chaque ressource, et plus précisément, chaque sujet du triplet est identifié par un URI (Uniform Resource identifier). Cette identification se fait de manière unique à l'aide d'un nom sans avoir à localiser la ressource (un bon exemple d'URI est l'URL). Le prédicat exprime la propriété, comme par exemple « est créateur » dans le triplet « Monsieur X – est créateur – du Web Sémantique » où Monsieur X est le sujet et enfin « Web Sémantique » est l'objet.

Par exemple, le fait que le document van-gogh.xml « parle de » peintures à l'huile sur toile de la période néo-impressionniste peut être représenté par le graphe suivant fondé sur une ontologie provenant du domaine de l'art. [34]

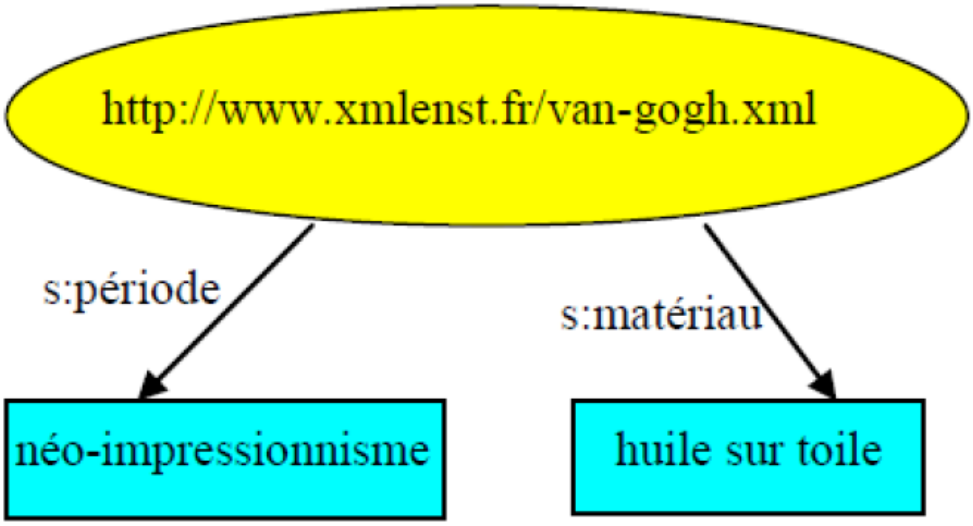

Figure 2.12: Exemple d'un graphe de ressource RDF

La recommandation RDF propose l'utilisation de XML pour la représentation des graphes de ressources. Voici une représentation XML du graphe de la Figure 2.6 :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://schemas.xmlenst.fr/peintres_rdf/">
<rdf:Description about="http://www.xmlenst.fr/van-gogh.xml">
        <s:période>néo-impressionnisme</s:période>
        <s:matériau>huile sur toile</s:matériau>
</rdf:Description>
</rdf:RDF>
```

Figure 2.13 : Exemple d'une représentation XML d'un graphe de ressource

Pour utiliser les termes d'une ontologie, il faut pouvoir indiquer les différents vocabulaires utilisés. Ainsi, au début de chaque ontologie, on intègre un ensemble de déclarations d'espaces de noms dans une balise *<rdf:RDF>*. Ces déclarations permettent l'interprétation sans ambiguïté des URIs et la lecture facile du reste de l'ontologie. On y trouve l'identification de l'espace de noms implicite de l'ontologie, l'identification de l'espace de noms relative aux préfixes utilisés. Enfin, on y trouve les différentes descriptions RDF ou, comme on le verra dans les sections suivantes, des éléments RDFS ou OWL.

Ainsi, RDF permet de définir aisément une ontologie, son inconvénient est qu'il ne supporte pas la vérification de la cohérence des données (vérification que le champ « date de naissance» est vraiment une date par exemple).

II.8.4. RDF Schéma

Une évolution de RDF est introduite dans RDFS (pour RDF Schema). Ce langage est simple et permet l'implémentation du modèle RDF pour la définition des ontologies avec une approche cette fois-ci orientée objet. Les trois notions principales permettant cela sont: la ressource (rdfs:Resource), la classe (rdfs:Class) et la propriété. On a avec RDFS l'avantage de pouvoir créer une hiérarchie de classes et de propriétés, comme dans les langages orientés objet, grâce aux notions de subClassOf et subPropertyOf. On peut bien entendu instancier une classe à l'aide de rdf:type. L'autre avantage vient du fait que RDFS restreint le domaine d'application des propriétés (avec un domaine qui précise la classe du sujet du triplet utilisant la propriété et un intervalle précisant la classe de l'objet du triplet utilisant la propriété).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <rdfs:Class rdf:ID="Objet_Iconographique"/>
    <rdfs:Class rdf:ID="Peinture">
        <rdfs:subclassOf rdf:resource="#Objet_Iconographique"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Période"/>
    <rdf:Property rdf:ID="date">
        <rdfs:domain rdf:resource="#Objet_Iconographique"/>
        <rdfs:range rdf:resource="Période"/>
    </rdf:Property>
    <rdf:Property rdf:ID="matériau">
        <rdfs:domain rdf:resource="Peinture"/>
        <rdfs:range rdf:resource="rdfs:Literal"/>
    </rdf:Property>
    <Peinture about="van-gogh.xml">
        <date>
            <Période id="#néo-impressionnisme"/>
        </date>
        <matériau>huile sur toile</matériau>
    </Peinture>
</rdf:RDF>
```

Figure. 2. 14: exemple de RDF Schéma

Dans ce document, l'ontologie est composée de trois concepts ou classes (Littéral, Peinture, Période) et de deux propriétés (matériau et date). Peinture est une description RDF, elle est appelée la ressource sujet des deux propriétés qui ont comme ressources objets, respectivement, Matériau et Période.

L'intérêt de définir un schéma RDFS n'est pas seulement de pouvoir contrôler la terminologie et la structure des descriptions RDF, mais également d'introduire la

possibilité de raisonner sur les liens isA (est-un) qui existent entre les concepts et les propriétés. Ceci est surtout utile quand on veut interroger des métadonnées pour découvrir des ressources. Voici un exemple de requête RQL, qui cherche toutes les ressources sur des objets iconographiques de la période néo-impressionniste : [35]

II.8.5. OIL ("Ontology Interchange Language and Ontology Inference Layer"), développé en début 2000 dans le cadre du projet européen On-To-Knowledge . Il ajoute des primitives de RC basées sur des "frames" à RDF(S) et sa sémantique formelle est basée sur les logiques de description.

II.8.6. DAML+OIL ("DARPA Agent Markup Language"), créé plus tard entre 2000 et 2001 par un comité mixte des USA et de l'Union Européenne dans le contexte du projet DAML sur la spécification précédente de DALM-ONT, qui a été construit en fin 2000, et sur OIL. DAML+OIL ajoute des primitives de RC basées sur les logiques de description à RDF(S). [32]

II.8.7. OWL OWL, recommandé par le W3C en février 2004, est le plus expressif des langages ontologiques pour le Web. La conception d'OWL a bénéficié de plusieurs générations de langages de représentation des connaissances, d'une base théorique solide en logique et d'une volonté de la part de ses concepteurs pour créer un langage approprié à une utilisation dans le cadre du Web Sémantique. En fait, OWL est issu des travaux autour du langage DAML+OIL, lui-même fusion de deux projets l'un européen, OIL, et l'autre américain, DAML. La plupart des chercheurs ayant participé à l'élaboration du langage DAML+OIL ont ensuite travaillé à OWL.

Le langage OWL fournit des mécanismes pour créer tous les composants d'une ontologie : classes, instances, propriétés et axiomes. OWL repose également sur la syntaxe des triplets

RDF et réutilise certaines des constructions RDFS. Comme en RDFS, les classes peuvent avoir des sous-classes, fournissant ainsi un mécanisme pour le raisonnement et l'héritage des propriétés. Par contre, en OWL, on distingue :

- ✓ les propriétés objet (object property), i.e. les relations, qui relient des instances de classes à d'autres instances de classes. C'est l'équivalent des triplets RDF dont l'objet est une ressource.

- ✓ les propriétés type de données (datatype property), i.e. les attributs, qui relient des instances de classes à des valeurs de types de données (nombres, chaînes de caractères,…). C'est l'équivalent des triplets RDF dont l'objet est une valeur littérale.

Les axiomes fournissent de l'information au sujet des classes et des propriétés, spécifiant par exemple l'équivalence entre deux classes. OWL se compose de trois sous-langages : OWL Lite, OWL DL et OWL Full. [36] [37]

II.9. Outils pour le Web Sémantique

II.9.1. Les éditeurs d'ontologies

De nombreux outils informatiques permettent aujourd'hui d'éditer des ontologies.

II. 9.1.1. Protégé [38]

L'environnement Protégé, Créé par les chercheurs de l'université de Stanford, est un éditeur d'ontologies développé en Java, gratuit et open source. Il s'agit d'une plateforme d'aide à la création, la visualisation et la manipulation d'ontologies dans divers formats de représentation (RDF, RDFS, OWL, etc.). Ce logiciel peut également être utilisé en combinaison avec un moteur d'inférence (tel que Racer ou Pellet) afin d'effectuer des raisonnements et d'obtenir de nouvelles assertions.

La plateforme Protégé permet la création et l'édition d'ontologies grâce à deux outils distincts :

- Protégé Frame permet de créer facilement une interface graphique afin de gérer. Cet outil ne demande aucune notion de programmation. Il gêner automatiquement les formulaires nécessaires en se basant sur le schéma d'ontologie l'utilisateur.
- Protégé OWL est une extension de Protégé qui supporte le langage OWL. Il permet de décrire plus précisément les classes, les propriétés et les instances grâce aux nombreuses propriétés offertes par OWL. Il est également possible d'interroger un raisonneur via une interface DIG afin de contrôler l'intégrité du modèle et de créer modèle d'inférences.

II.9.1.2. OILEd

OILEd développé sous la responsabilité de l'université de Manchester, a été conçu pour éditer des ontologies dans le langage de représentation OIL, un des précurseurs du langage DAML+OIL. Officiellement, il n'a pas d'autre ambition que de construire des exemples montrant les vertus du langage pour lequel il a été créé.

A ce titre, OILEd est souvent considéré comme une simple interface de la logique de description SHIQ. On peut créer des hiérarchies de classes et spécialiser les rôles, et utiliser avec l'interface les types d'axiomes les plus courants. Cet éditeur offre également les services d'un raisonneur, FaCT (Fast Classification of Terminologies), afin entre autres de déceler les inconsistances dans les ontologies. [39]

II.4.1.3. ONTOEdit

Contrairement aux deux outils précédents, ONTOEdit n'est pas disponible gratuitement dans sa version complète. Il présente les fonctionnalités essentielles communes aux autres éditeurs (hiérarchie de concepts, expression d'axiomes, export de l'ontologie dans des langages divers) et a le mérite de s'appuyer sur une réflexion méthodologique significative. La modélisation des axiomes a fait l'objet de soins particuliers pour pouvoir être effectuée- en tous cas pour les types les plus répandus- indépendamment d'un formalisme privilégie, et cela pour faciliter la traduction d'un langage de représentation à un autre. [39]

II.4.2. Les moteurs d'inférence

La sémantique formelle du langage OWL permet l'application des techniques de Raisonnement pour effectuer des dérivations logiques. Ces dérivations sont effectuées par des moteurs d'inférence (également nommés moteurs de raisonnement, raisonneurs sémantiques, ou tout simplement des raisonneurs), ce sont des programmes qui peuvent lire des ontologies à partir de fichiers OWL ou des serveurs web distants, ce sont donc des systèmes capables de gérer et d'utiliser la sémantique du langage de l'ontologie.

Une procédure générale de raisonnement est illustrée dans la figure suivante (Figure 2.6)

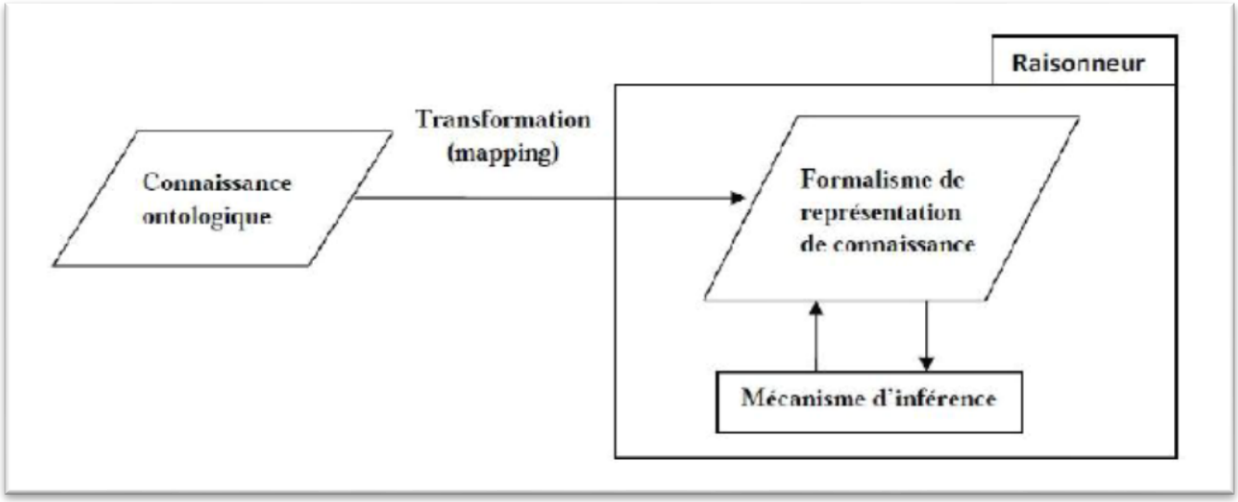

Figure 2.15 : Architecture abstraite d'un raisonneur OWL [40]

Comportant deux phases

1. Phase de transformation (mapping) des connaissances vers un formalisme de représentation de connaissance.
2. Application d'un mécanisme d'inférence pour calculer les inférences.

A l'heure actuelle, il existe de nombreux raisonneurs LD (moteurs d'inférence) capables de calculer les inférences. Une fois l'ontologie chargée, ces moteurs effectuent les inférences sur la TBox et la ABox. Pellet et Racer sont à l'heure actuelle les deux seuls moteurs d'inférence, permettant le raisonnement sur la ABox et la TBox. Ils exploitent des ontologies possédant un niveau d'expressivité en logique de descriptions.

II.4.2.1. Racer

Le système Racer (Renamed ABox and Concept Expression Reasoner ou raisonneur d'expression de concept et de ABox renommées) est un système de représentation de connaissance pour le calcul DL. Racer est le moteur d'inférence sans doute le plus connu et l'un des plus utilisés pour ces performances et sa stabilité. Racer travaille sur les ontologies modélisées par son langage, mais il accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer. [41]

II.4.2.2. Pellet

Le moteur Pellet est beaucoup plus récent. Pellet est un des projets du MINDSWAP Group, un groupe de recherche sur le web sémantique de l'université du Maryland. Il est disponible en Open Source et offre des évolutions fréquentes. Pellet travaille sur des ontologies décrites en RDF ou OWL et permet les requêtes avec RDQL et SPARQL sur la ABox et la TBox. [42]

Conclusion

Les ontologies sont des descriptions formelles d'un domaine particulier, elles ont un rôle primordial dans l'interopérabilité et l'intégration des applications distribuées. Elles facilitent aussi la médiation entre les sources de données hétérogènes

Dans ce chapitre, nous avons rappelé la notion des ontologies, leurs différents types et composants, et quelques outils de manipulation.

A cet effet, nous allons présenter dans le chapitre suivant le défèrent étapes pour la construction d'une ontologie de cancer

Chapitre 03

Construction de l'ontologie de cancer

Introduction

Ce chapitre présente la première partie de notre contribution au problème posé par ce mémoire qui est la construction d'une ontologie pour le domaine de la médecine dédiée au service de cancer. Pour ce faire, nous nous somme basé sur la méthodologie<<METHONTOLOGY>> qui est le support de base pour la conceptualisation de l'ontologie à créer, à travers un ensemble de représentations intermédiaires semi-formelles. La logique de descriptions, est le formalisme adopté pour l'expression de l'ontologie semi-formelle.

III.1. Processus construction d'une ontologie OWL

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. Différentes méthodes de construction ont été proposées (METHONTOLOGY, TOVE, ….)

Nous avons proposé un processus de construction d'une ontologie d'application partant de connaissances brutes et arrivant à une ontologie d'application opérationnelle représentée par le langage OWL. Ce processus est composé de cinq étapes [45] :

- Spécification des besoins.
- Conceptualisation.
- Formalisation.
- Implémentation.
- Test & évolution de l'ontologie.

III.1.1. L'étape de spécification (le cadrage)

Le but de cette étape est de cerner l'étendue de l'ontologie et le domaine à prendre en compte, dans notre cas, il s'agit du domaine médicale. Dans cette étape on va identifier l'ensemble des termes qui sera représentés avec leurs caractéristiques, cette spécification doit être le plus possible complète et concise.

La figure représentée ci-dessous est un document d'une spécification des besoins d'une ontologie dans le domaine médicale nous avons arrivé à ce document après l'analyse de l'énoncé de mini projet et la récolte des connaissances du domaine.

III.1.2.L'étape de Conceptualisation

L'étape de conceptualisation est l'étape la plus importante lors de développement des ontologies, car

- c'est à ce niveau-là qu'on va définir et structurer les connaissances spécifiques au domaine de l'ontologie c.à.d le contenu de l'ontologie à développer donc une bonne conceptualisation donne à la fin une ontologie efficace.
- elle facilite le passage d'une ontologie informelle à une ontologie formelle.

Dans cette étape on distingue les principales activités suivantes :

- la construction d'un glossaire de termes.
- Classification des termes dans des hiérarchies de concepts.
- La construction d'un diagramme de relations binaires.
- La construction d'un dictionnaire de concepts.
- La construction d'une table de relations binaires.
- La construction d'une table des attributs.
- La construction d'une table des axiomes logiques.
- La construction d'une table des instances.
- La construction de la table des assertions.

III.1.2.1.Construction du glossaire de termes :

Ce glossaire recueille et décrit tous les termes qui sont utiles et potentiellement utilisables dans l'ontologie finale. Le tableau ci-dessous (Tableau 3.1) fournit une liste détaillée des différents termes utilisés dans l'ontologie

Terme	Description
Hôpital	Etablissement ou l'on prodigue des soins médicaux
Maladie	La maladie est une altération des fonctions ou de la santé d'un organisme. se traduisant par des symptômes et des signes. Elle concerne un ou plusieurs patients.
Personne	Un humain qui peut être un médecin ou un malade
Médicaments	Substance ou préparation administrée en vue de modifier, corrigé des fonctions organique.

Malade	est une personne souffrant d'une maladie
Médecin	Un médecin est un professionnel de la santé titulaire d'un diplôme de docteur en médecine. Il est chargé de soigner les maladies, pathologies
Infirmier	est un professionnel de santé paramédical dont le rôle est de délivrer des soins infirmiers au patient sur prescription médicale ainsi que des soins d'hygiène et de prévention..
Aide-soignant	L'aide-soignante travaille en étroite collaboration et sous la responsabilité et l'encadrement de l'infirmière. Elle dispense aussi bien des soins d'hygiène et de prévention et contribue à la réalisation des actes de la vie quotidienne.
Agent-santé	Est une personne affiliée dans le domaine de la santé. Il peut être : Un médecin, infirmier, biologiste, ….
Service	Organisme qui fait partie d'un ensemble administratif ou économique
Préventif	Un type de traitement, tel que la vaccination, qui cherche à empêcher l'apparition d'une maladie ou à supprimer un facteur de risque.
Facteurs risques	Ce sont des éléments qui augmentent le risque d'être atteint par une maladie.
Traitement	Manière de soigner un patient qui a une ou plusieurs anomalies.
Consultation	L'examen de la patiente par un médecin pour découvrir une ou plusieurs anomalies.
Interrogatoire	L'interrogatoire médical, c'est le moment où l'on interroge le patient au début de la consultation pour recueillir certaines informations
Examen-clinique	L'examen physique, ou examen clinique, fait partie de l'examen médical, qui permet au médecin ou au clinicien (infirmière, infirmière clinicienne, infirmière praticienne) d'aboutir à un diagnostic, ou à une impression clinique.
Psychologue	Il accompagne et soutient le patient et son entourage selon les besoins.
Staff administratif	Personnes qui se préoccupent de la partie administrative du laboratoire.
Agent d'accueil	Personne chargée de recevoir et d'orienter les patients.
Agent de saisie	Personne chargée des activités administratives
…….	…….

Tableau3. 1: Table du glossaire de termes

III.1.2.2. Hiérarchies de concepts

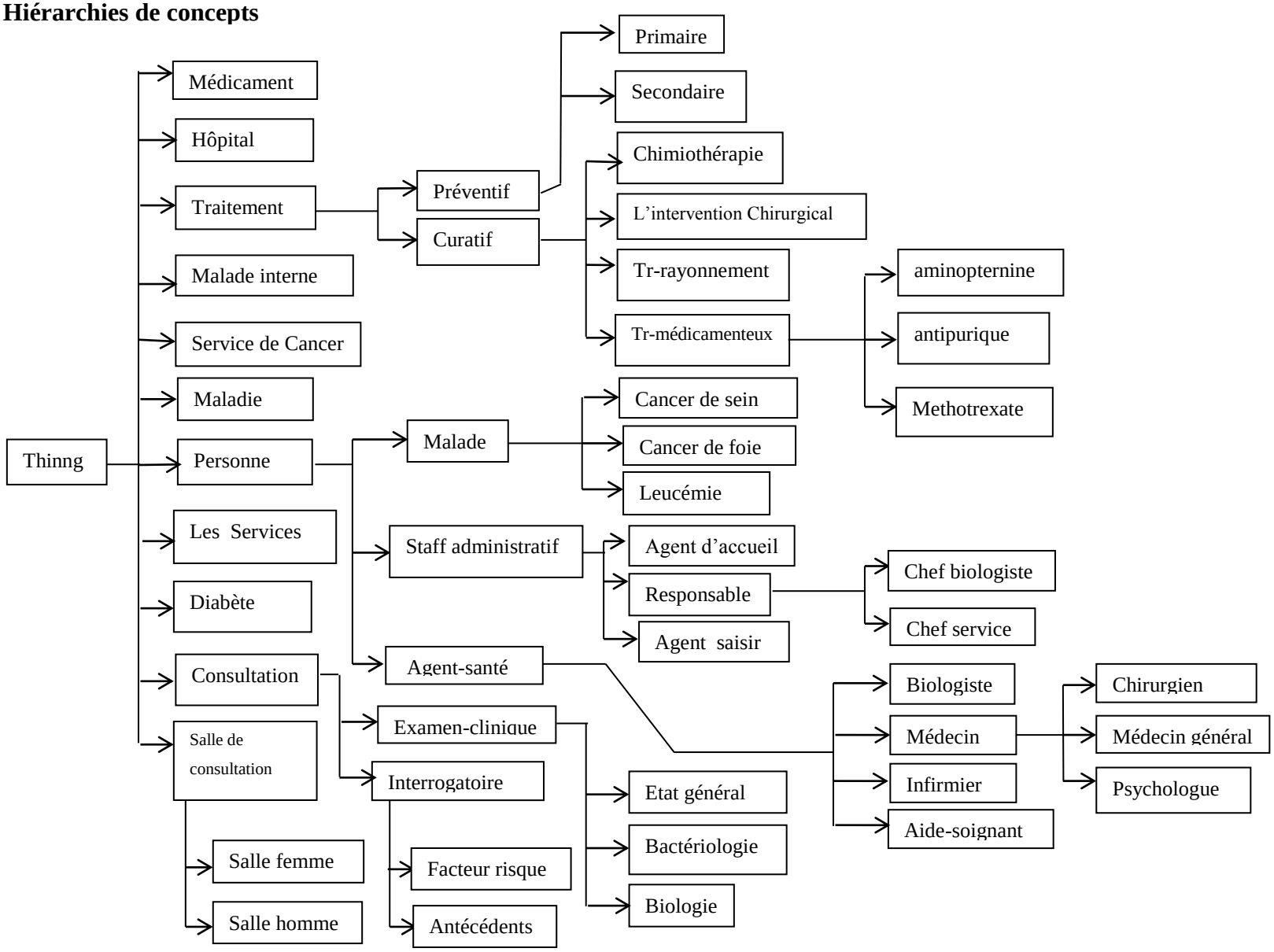

Figure 3. 16 : Hiérarchies de concepts.

III.1.2.3.Diagramme de relation binaire

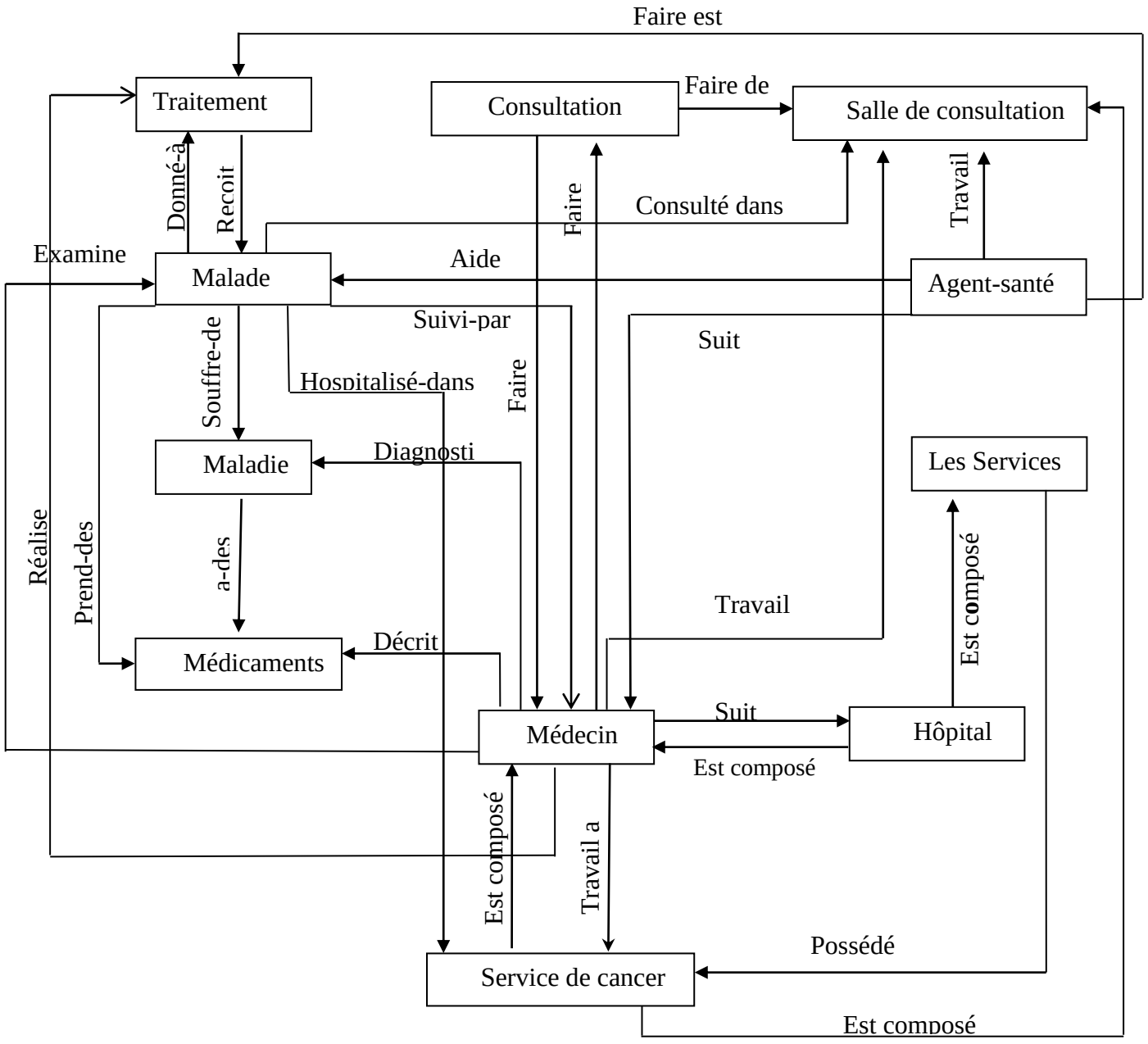

Figure 3.17**:** Diagramme de relation binaire

III.1.2.4.Dictionnaire de concepts

Dans cette étape nous allons donner une description formelle des concepts qui ont été présentés dans la hiérarchie des classes. Ce processus correspond à la création du dictionnaire de concepts accordé à METHONTOLOGY. Dans ce dictionnaire, nous définissons pour chaque concept : les instances, les attributs, les relations dont la source est ce concept, les synonymes et les acronymes de ce concept; le tableau 3.2 représente le dictionnaire des concepts pour l'ontologie définie par la hiérarchie précédente.

Nom du concept	Concepts synonymes	attributs	Instances
Personne	Humain	-Nom -Prénom -Adresse personnel -Tel -Email	-
Médecin	Docteur	-Grade_M -spécialité_M	Yacoub chouaib
Malade	Patient	-sexe -Age -Type de malade	Kamal ; djaber ; sami
Traitement	Cure, remède, soins	-Durée-Traitement	-
Consultation	Examen, visite, Réception	-Date-consultation	Nassima, fatima, zahwaniya
Infirmier	Soignant	-	Nabti imane
Aide-soignant	Aide-infirmier	-	- Latreche Farid
Staff admiratif	-	-type de staff_adm	-
Chef de service	Responsable	-type_Res	Djoudi hakimo
Préventif	-	-Descrip-prev	-
Agent-santé	-	-type d'agent	Bouguerra Amar
Hôpital	CHU Ibn Badiss	-Dénomination_h -Adresse hôpital -Tele-hôpital -Fax-hôpital -Type-hôpital	CHU mila -
Interrogatoire	Questionnaire	-Descrip-inter	-
Examen-clinique	Examen direct, physique	-Diagnostic	-
Curatif	-	-Début-trait	-
Biologiste	-	-sexe_biol	- Amamra Adlen
Service	-	-Dénomination_ser -Tel_ser -Fax_ser	-Service de cancer

Agent d'accueil	-	-	-amar edawla
Agent de saisie	-	-	- Nawi sofaine
Médicament	-	Type-trait Forme Dosage Posologie Durée	-
……	……..	……..	……

Tableau 3.2: table de Dictionnaire de concepts

III.1.2.5. Tableau de relations binaires

Les relations binaires sont représentées sous forme de propriétés ou attributs qui lient un concept à un autre, ce sont des 'attributs de type instance' : c'est-à-dire les attributs ayant pour type de valeur Instance [48].

Pour chaque relation dont la source est dans l'arbre de classification de concepts, nous définissons : son nom, le nom du concept source, le nom du concept cible, la cardinalité et le nom de la relation inverse; le tableau III.3 illustre la spécification des relations binaires entre les différentes hiérarchies pour notre ontologie.

Nom de la relation	***Concept source***	***Cardinalité source***	***Concept cible***	***Cardinalité cible***	***Relation inverse***
Travaille dans	Agent-santé	(1, n)	Services de cancer	(1, n)	-
Diagnostique	Médecin	(1, n)	Maladie	(1, n)	-
Suit	Médecin	(1, n)	Services de cancer	(1, n)	-
Réalise	Médecin	(1, n)	Consultation	(1, n)	Réalise par
Examiné	Médecin	(0, n)	Malade	(1, n)	Examiné-par
Travail	Médecin	(1, n)	Salle de consultation	(1,1)	-
Décrit	Médecin	(0, n)	Médicament	(1, 1)	Décrit a
Reçoit	Malade	(0, n)	Traitement	(0, n)	Donné-à
Subit	Malade	(0, n)	Consultation	(0, n)	-
Faire est	Agent-santé	(1, n)	Traitement	(1, n)	Traité par
Souffre	Malade	(0, 1)	Maladie	(0, n)	-
Consulté dans	Malade	(1, n)	Salle de consultation	(1, n)	-

Hospitalisé-dans	Malade	(0, n)	Services de cancer	(1, n)	-
Apaise	Médicament	(0, n)	Maladie	(0, n)	Apaise par
Traité	Médicament	(0, n)	Malade	(0, n)	Traité par
Faire de	Consultation	(0, n)	Salle de consultation	(0, n)	-
Est composer	Services de cancer	(1, n)	Salle de consultation	(1, 1)	-
Travail a	Agent-santé	(1, n)	Salle de consultation	(1, n)	-
Est composer	Hôpital	(1, n)	Les services	(1, n)	-
Est composer	Les services	(1, n)	Services de cancer	(1, n)	-
Suit a	Agent-santé	(1, n)	Services de cancer	(1, n)	-
Réalise	Médecin	(1, 1)	traitement	(1, 1)	-
Suivi-par	Malade	(1,1)	Médecin	(1, n)	-
A des	maladie	(1, n)	médicament	(1, n)	-
…….	……	…….	…….	……	……

Tableau 3.3 : tableau des relations binaires

III.1.2.6.Construction de la table des attributs

Les attributs sont des propriétés qui prennent leurs valeurs dans les types prédéfinis (String, Integer, Boolean, Date…). Par exemple le concept femme a comme attributs : Nom, prénom, âge, DDR et pariété. Pour chaque attribut apparaissant dans le dictionnaire de concepts nous spécifions: son nom, type et intervalle de ses valeurs possibles, et sa cardinalité (pour spécifier qu'une instance possède une ou plusieurs valeurs). Le tableau 3.4 spécifie ces informations pour chaque attribut.

Nom de l'attribut	***Type***	***Cardinalité (Min/Max)***	***Valeur par défaut***	***Domaine des valeurs***
Adresse personnel	Littéral	(1,1)	-	-
Nom	Littéral	(1,1)	-	-
Prénom	Littéral	(1, n)	-	-
Adresse personnel	Littéral	(1,1)	-	-
Grade-M	Littéral	(1,1)	-	(professeur, généraliste)

Tél	Littéral	(1, n)	-	-
Email	Littéral	(0, n)	-	-
Fax	Littéral	(0, n)	-	-
Type-hôpital	String	(1, n)	-	{Général, Spécialisé, militaire, universitaire}
Date-consultation	Date	(1,1)	-	-
Dénomination	Littéral	(1,1)	-	CHU
Début-trait	Date	(1,1)	-	-
Descrip-prev	Littéral	(1, n)	-	(urgent, normal)
spécialité	Littéral	(1, n)	-	General, Chirurgien…
sexe	Littéral	(1, 1)	-	(masculin, féminin)
age	Int	(1, 1)	-	-
Adresse serv	Littéral	(1, 1)	-	-
Adresse hôpital	Littéral	(1, 1)	-	-
……	……	……..	……	……

Tableau 3.4**:** Table de Construction des attributs

III.1.2.7.Construction de la table des axiomes logiques

Ces tableaux contiennent des définitions de concepts à l'aide des expressions logiques qui sont toujours vraies. Dans ce tableau nous définissons pour chaque axiome sa description en langage naturel, le nom du concept auquel l'axiome se réfère, les attributs utilisés dans l'axiome et l'expression logique. Pour notre ontologie nous spécifions quelques axiomes comme il est représenté dans le Tableau 3.5

Nom du concept	Description	Expression logique
Personne	Chaque personne est soit un Biologiste, Infirmier, Aide-soignant, Malade, Chef de service, Médecin,	∀ (X), Personne(X) ⇒Biologiste(X) ∪ Infirmier(X) ∪Aide-soignant (X) ∪ Malade(X) ∪ Chef de service(X) ∪ Médecin(X)
Biologiste	Est un spécialiste des sciences du vivant.	∀ (X), Biologiste(X) ⇒ Spécialiste sciences vivant(X)
Malade	Personne qui souffre d'une maladie ne doit au moins se rendre dans une structure de santé, et être examiné par un médecin et être a des médicaments.	∃ (X), Personne (X) ⇒Maladie(X) ∩ ⇒∃ Y sale de consultation(Y) ∩ Consulté dans (X,Y) ∩ ∃ Z Médecin (Z) ∩ Examiné-par (Z, X) ∩ ∃ N medicaments(N) ∩ ades (X, N)
Sal de consultation	Est l'endroit à être un examen d'un patient	∃ (X), sale de consultation(X) ⇒ Infirmier(X) ∩Aide-soignant(X) ∩ malade(X) ∩ médecin(X) ∩ médicament(X)
Consultation	Est un processus effectué par le médecin au patient	∃ (X), consultation (X) ⇒ malade(X) ∩ médecin(X)
Traitement	Traitement peut être préventif ou curatif.	∀ (X), Traitement (X) ⇒Curatif (X) ∪ préventif (X)
Agent-santé	Personnes ne faisant pas partie des professions médicales, mais qui ont trait sur le plan technique ou administratif aux activités relatives à la santé. Peut-être soit : infirmier soit aide-soignant.	∀ (X), agent-santé(X) ⇒ Infirmier(X) ∩Aide-soignant(X) ∩ Médecin

Staff administratif	Chaque personne est soit un : agent d'accueil, agent saisir, responsable	∀ (X), Staff administratif(X) ⇒agent d'accueil(X) ∩ agent saisir(X) ∩ responsable
Médecin	Un médecin travaille dans un hôpital, examine des malades, prescrit des traitements et dépiste des maladies.	∀ (X), Médecin (X) ⇒∃ (Y), Structure-santé (Y) ∩ Travaille (X, Y) ∩ ∃ (Z), malade(Z) ∩ Examine (X, Z) ∩ ∃ (W), Traitement (W)
Examen clinique	Un examen clinique est effectué par le médecin	∀ (X), Examen-Clinique(X) ⇒∃y Medecin (y) ∩ effectue (y,x)
Médicament	Chaque médicament est pris par des malade	∀ (X), Médicament(X) ⇒Malade(X) ∩ ∃ Y Médecin (Y) ∩ decrit (Y, X)
Service de cancer	Chaque service de cancer constitué de : Maladie, salle de consultation, consultation, traitement, personne	∀ (X),service de cancer⇒Maladie (X) ∩ personne(X) ∩ traitement(X) ∩ consultation(X) ∩ sale de consultation(X)
……	……	.....

Tableau 3.5: Table de Construction des axiomes logiques

III.1.2.8.Construction de la table des instances

Dans cette section nous donnons une description de quelques instances de l'ontologie. Pour cela, nous spécifierons les noms des individus et les valeurs des attributs pour chacun d'eux; le Tableau 3.6 illustre quelques instances pour chaque classe.

Concept	***Nom de l'instance***	***Attributs***	***Valeurs***
Médecin	salah-dj	-Nom -Prénom -Grade_M -spécialité_M -Adresse -Email -Tel	Djoudi salah Professeur cancer de foie m'sila dz.@gmail.com 0660925875
Hôpital	CHU-Mila	Dénomination Adresse_hopital Tel_hôpital Fax_hôpital Type-hôpital	CHU Mila Rue snawa Mila 31 64 16 07 / 031 64 29 72 / 031 64 29 73 / 031 64 17 00 Universitaire
Malade	Chaki kamal	-Nom -Prénom -sexe -Age -Tel	Chaki Kamal Masculin 25 0664025850
Biologiste	Botira sami	-Nom -Prénom -Adresse -Email -Tel	sami botira Constantine sami@gmail.com 0771345609

Infirmier	Bodiba djaber	-Nom -Prénom -Adresse -Email -Tel -sexe	Bodiba Djaber kikda djeber21@gmail .com 07825892875 Masculin
Traitement	-	Nom traitement	Paracétamol
Services	Etablissement sanitaire	Dénomination Tel Fax	Service de cancer 031585858 031585847
Agent -santé	-	-	-
Médicament	Methotrexate	Type-trait Forme Dosage Posologie	antibiotique injectable 100 ml (500 mg) 1 à 1,50 g par jour
Aide-soignant	Nabti imane	-Nom -Prénom -Adresse -Email -Tel -sexe	Nabti Iman Ain tine xxx @gmail.com Xxxxxxxxxx Femme
Psychologue	Bodiba djaber	-Nom -Prénom -sexe - Email -Adresse -Tel	Bodiba Djaber Masculin yyyy@gmail.com DNC xxxxxxxx
Médecin général	Botira sami	-Nom -Prénom -sexe - Email -Adresse	Botira Sami Masculin sss@gmail.com tbessa

		-Tel	xxxxxxxx
Chirurgien	Chaki kamal	-Nom -Prénom -sexe - Email -Adresse -Tel	Chaki Kamal Masculin qqqqq@gmail.com msila xxxxxxxxxx
……	……..	…….	……

Tableau 3.6: Table de Construction des instances

III.1.2.9.Construction de la table des assertions

Relation	***Instance Sources***	***Instances Cibles***
Est composé	Hôpital	Les Services
Possédé	Les Services	Service de cancer
Travail	Médecin	Sale de consultation
Travail a	Médecin	Service de cancer
Est composé	Service de cancer	Sale de consultation
Faire de	Consultation	Sale de consultation
Aide	Agent-santé	Malade
Travail	Agent-santé	Sale de consultation
Suit	Agent-santé	Service de cancer
Donné-à	Malade	Traitement
Reçoit	Traitement	Malade
Souffre	Malade	Maladie
Suit	Médecin	Hôpital
Faire	Médecin	Consultation
Faire par	Consultation	Médecin
Examine	Médecin	Malade
Diagnostique	Médecin	Maladie
Décrit	Médecin	Médicaments
Prend	Malade	Médicaments

Se- rend	Malade	Service de cancer
Apaise	Maladie	Médicaments
Consultation dans	Malade	Sale de consultation
Contient	Les services	Agent-santé
est Faire	Agent-santé	Traitement
……	……..	…….

Tableau 3.7 : Table de Construction des assertions

III.1.3. Formalisation

Cette étape consiste à formaliser l'ontologie conceptuelle obtenue dans la phase précédente afin de faciliter sa représentation ultérieure dans un langage complètement formel et opérationnel. Notre choix est porté sur le formalisme de représentation de la logique de description en s'appuyant sur sa syntaxe de type SHIQ qui présente une logique de description très expressive et qui offre un certain nombre de constructeurs pour décrire les concepts

Syntaxe	**Interprétation**
T	Le concept le plus général (TOP)
⊥	Le concept le plus spécifique (Buttom)
C	Le nom d'un concept
R	Le nom d'un rôle (une relation binaire ou bien un attribut)
A	Le nom d'un individu
C1 ∩ C2	Conjonction de concepts pour définir de nouveaux concepts
C1 ∪ C2	Disjonction de concepts pour définir de nouveaux concepts
¬ C	Utilisé pour définir le complément d'un concept
∀ R.C	Définit le Co-domaine du rôle R
∃R.C	Il y a au moins un objet relié par le rôle R au concept C
(≥ n R) (≤ n R)	Cardinalité Minimum/Maximum (n est un nombre entier non négatif).
R^{-}	Le role inverse

Tableau 3.8**:** Table de Syntaxe du langage SHIQ

III.1.3.1 Construction de T-BOX :

Nous construisons la TBox en définissant des concepts et les rôles et en utilisant les constructeurs fournis par les logiques de descriptions.

Concept	Définition	Relation de subsomption
Personne	Personne := médecin ∪ infirmier ∪ malade ∪ Staff admiratif	Personne ⊆ Thing
Hôpital	Hôpital := les services	Hôpital⊆ Thing
Les Services	Les services := service de cancer services ∩ malade interne ∩ Diabète	Les services⊆ hôpital
Service de cancer	Service de cancer := Maladie∩ Sale de consultation ∩ Traitement ∩ Consultation ∩ Personne ∩ (∃ est compose. Salle consultation) ∩ (∃ est compose. Médecin)	Service de cancer ⊆ Thing
Médecin	Médecin :=Personne∩(Généraliste∪ Chirurgien∪Psychologue)∩ (∃ suit.Hôpital) ∩ (∃ Réalise. consultation)∩(∃ Travaille dans.Service_de_cancer)∩ (≥ 1travail de salle de consultation ∩ ≤ 1 tarvail de salle de consultation) ∩(∀ examiné. Malade) ∩ (≥ 1décrit.médicament ∩≤ 1décrit.médi cament)	Médecin ⊆ Personne
Médecin général	Médecin général := personne ∩Médecin	Médecin général ⊆ Médecin
Agent santé	Agent santé := personne ∩ (∀ aide .Malade) ∩ (∃ travail .salle de consultation) ∩ (∃ faire est .traitement) ∩ (∃ suit. Médecin)	Agent santé ⊆ personne
Traitement	Traitement := Curatif ∪ Préventif ∩ (∃ Donné-à . Malade)	Traitement ⊆ Thing

Consultation	Consultation := Interrogatoire ∪ Examen-clinique ∩ (≥ 1 Faire de .salle de consultation ∩ ≤ 1faire de .consultation)	Consultation ⊆ Thing
psychologue	Psychologue := personne ∩Médecin	psychologue⊆ Médecin
Malade	malade :=personne ∩ (∃ Reçoit .traitement) ∩(∃examine par .médecin)∩ (∃ souffre-de. Maladie)∩ (≥ 1 Hospitalisé-dans. Service de cancer ∩ ≤ 1 Hospitalisé-dans. Service de cancer) ∩ (∃ Suivi-par. Médecin) ∩ (∃ Prend-des. Médicament)) ∩ (≥ 1 consulté dans. Salle de consultation ∩ ≤ 1 consulté dans. Salle de consultation)	malade ⊆ Personne
Maladie	Maladie:=∀ a-des. Médicament	Maladie ⊆ Thing
Examen clinique	Examen-clinique := Etat-général ∪ Bactériologie ∪ Biologie	Examen clinique ⊆ Consultation
Salle de consultation	Salle de consultation :=salle homme ∩ salle femme	Salle de consultation⊆Thing
Curatif	Curatif := Tr-rayonnement ∩ L'intervention Chirurgical ∩ Chimiothérapie ∩ Tr-médicamenteux	Curatif ⊆ traitement
……	…….	……

Tableau 3.9**:** Définition des concepts et subsomption.

Rôle	**Couple (domaine, co-domaine)**	**Rôle inverse**
Soufre	(malade, Maladie)	-
Travaille dans	(Agent-santé, Services de cancer)	-
Diagnostique	(Médecin, Maladie)	-
Suit	(Médecin, Services de cancer)	-
Réalise	(Médecin, consultation)	Réalise par
Travail	(Médecin, Sale de consultation)	-
Décrit	(Médecin, médicament)	Décrit par
Reçoit	(Malade, traitement)	Donné-à

Subit	(Malade, consultation)	-
Faire est	(Agent-santé, traitement)	Traité par
Examiné-par	(Malade, Médecin)	Examiné
Consulté dans	(Malade, Salle de consultation)	-
Se-rend	(Malade, Services de cancer)	-
Apaise	(médicament, Maladie)	Apaise par
Traité	(médicament, Malade)	Traité par
Réalise par	(consultation, Médecin)	Réalise
Faire de	(consultation, Salle de consultation)	-
Est composer	(Services de cancer, Salle de consultation)	-
Travail	(Agent-santé, Sale de consultation)	-
Est composer	(Hôpital, Les services)	-
Est composer	(Les services, Services de cancer)	-
Suit	(Agent-santé, Services de cancer)	-
……	……	……..

Tableau 3.10 : Table de Définition des rôles

III.1.3.2 Construction de ABox

Le langage assertionnel est dédié à la description des faits, en spécifiant leurs classes et les relations entre eux.

Concept	Description
Psychologue	Médecin : chaki kamal
Agent d'accueil	Agent d'accueil : amar dawla
	Agent d'accueil : dahkal sofyane
Agent saisir	Agent saisir : tali brahim
Infirmier	Infirmier :djmiyat assya
	Infirmier : djmiyat affaf
Aide-soignant	Aide-soignant : loula smahan
	Aide-soignant : amra adel

Biologiste	Biologiste : aitimen rokiya
	Biologiste : baouche zina
Médicament	Médicament : MTX
	Médicament :zimor 20
	Médicament :indocollyre
Salle femme	Sale femme :salle 1
	Sale femme : salle 2
Salle homme	Salle homme : salle 3
	Salle homme :salle 4
Chef service	Chef service : djoudi salah
Chef biologiste	Chef service : belhaj faissal
Médecin général	Médecin general :bodiba djaber
	Médecin general: botira sami
Chirurgien	Chirurgien : nabti iman
	Chirurgien : belfoule ishak
M.Cancer de sein	M.Cancer de sein : azizi ilyas
	M.Cancer de sein : zaimach morad
M.ancer de foie	M.ancer de foie : bojnana yazid
	M.ancer de foie :bofineche amine
	M.ancer de foie : khadrawi zino
M.ancer cerveau	M.ancer cerveau: borahra zabana
	M.ancer cerveau: negriwi nawal
Leucémie	Leucémie :aimayrach sara
	Leucémie : zahi aziza
	Leucémie : hafsi samir

Tableau 3.11 : Table de Description des assertions de concepts.

III.1.4. Implémentation :

Après avoir conçu l'ontologie, il nous reste maintenant que son implémentation. Pour cela, nous disposons de nombreux langages et d'outils qui permettent d'implémenter l'ontologie proposée.

Notre choix porte sur OWL qui représente un langage de codification utilisé pour implémenter l'ontologie en OWL (un langage de définition d'ontologie pour le Web), et cela pour toutes les fonctionnalités sémantiques que permet OWL et qui sont plus riches que celles des langages RDFS & DAML+OIL. OWL fait partie du paradigme des logiques de descriptions. Sa sémantique peut être définie via une translation (transcription) vers la logique de descriptions SHIQ.

Nous allons examiner cet élément explication dans le dernier chapitre «application »

III.1.5. Test & évolution de l'ontologie

Nous avons utilisé le système pellet pour tester l'ontologie d'application, nous distinguons deux types de test: test de consistance et de satisfiabilité; le premier type de test consiste à enlever l'inconsistance entre les concepts et cela en utilisant la test de subsumption, par contre le test de satisfiabilité permet de vérifier pour chaque concept l'existence des instances; un concept C est satisfiable si et seulement si, il existe au moins une interprétation I (instance) pour le concept C.

D'après les tests que nous avons appliqués à l'ontologie d'application, aucune erreur n'est produite lors du test.

Conclusion

Dans ce chapitre nous avons présenté les différentes phases du processus de développement d'ontologies qui est basé principalement sur la méthode METHONTOLGY ainsi que la conception de notre ontologie. Comme il fallait choisir un domaine de connaissance, nous avons opté pour le domaine médical comme domaine applicatif. Nous avons montré ainsi comment l'ontologie a été conceptualisée puis formalisée et finalement opérationnalisée.

Chapitre 04

Conception du Web Service

Introduction

Ce chapitre présente la deuxième partie de notre contribution au problème posé dans ce mémoire, à savoir la construction d'un Web service qui va exploiter l'ontologie construite.

En premier lieu, nous proposons une architecture de l'application réaliser, ensuite nous présentons la conception de notre web service, en mettant l'accent sur sa mise en œuvre dans un site web en vue de simuler son utilisation dans une application web. La conception est réalisée par des diagrammes UML.

VI.1.Proposition d'une Architecture à base web service pour L'exploitation de l'ontologie :

Après la construction de l'ontologie, nous allons s'intéresser à son exploitation dans le but d'être plus utile tant pour les utilisateurs experts du domaine que pour les utilisateurs simples. De ce fait, nous avons proposé une architecture à base Web service dont les composants sont les suivants :

VI.1.1 Application :

Notre application est un site web, qui pour son développent on a construit un Web service pour l'exploitation de notre ontologie. Donc, dans l'application on a deux composants :

- Interface utilisateur : permet des échanges et des interactions entre les utilisateurs et le Web service
- Web service : c'est le composant essentiel dans notre application dans lequel s'effectuent tous les traitements.

VI.1.2 Utilisateur :

C'est la personne qui accède à notre application Web. Il peut être un utilisateur simple ou expert.

- Utilisateur simple : toute personne qui accède à notre application pour faire une consultation ou une recherche des informations.

- Utilisateur expert : c'est une personne spécialiste du domaine de la médecine, elle dispose un chemin pour faire les mises à jour de l'ontologie.

VI.1.3 Ontologie :

Est une base de connaissance contient les informations concernant un domaine spécifique, et exploiter par un Web service.

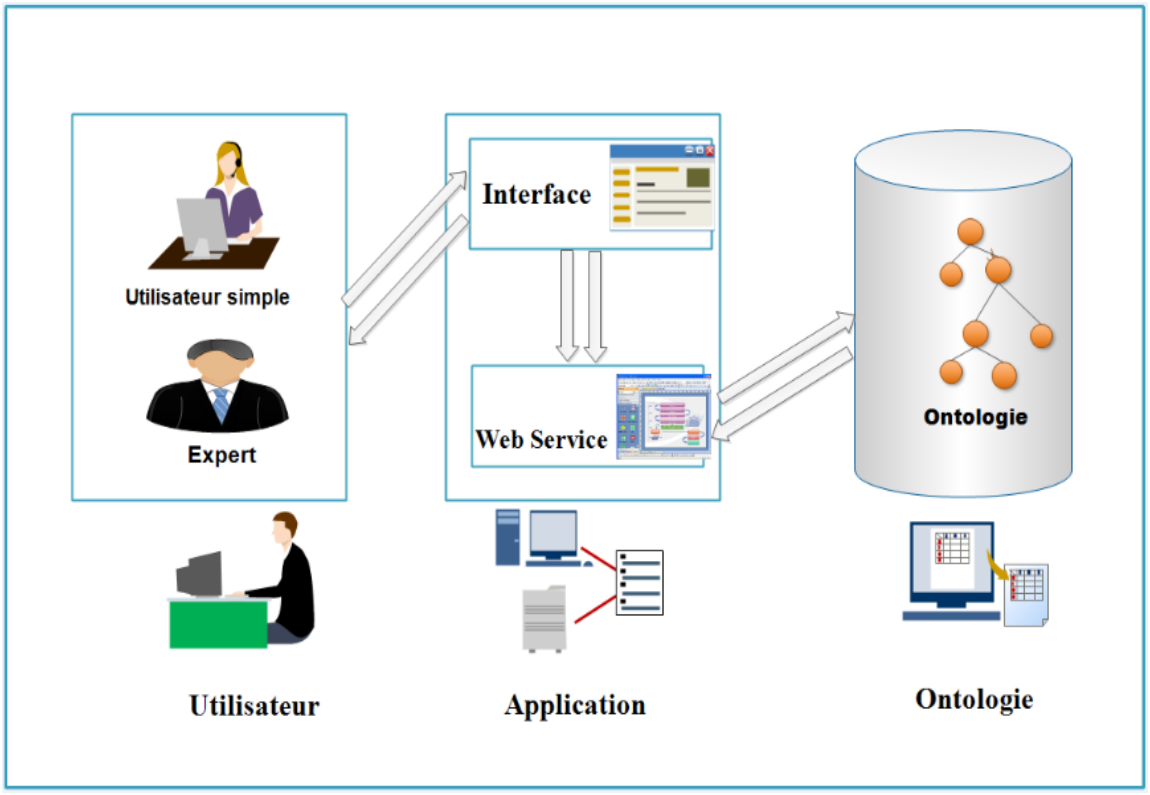

Figure 4.18: Architecture générale de l'application

VI.2. Conception de l'application Web :

La conception de notre application s'est avéré une tâche complexe d'où la nécessité de suivre une démarche d'analyse. Pour cela, nous avons utilisé le langage de modélisation UML (Unified Modeling Language) et le processus UP (Unified Process) d'une manière moins détaillée, où nous avons présenté quelques diagrammes qui sont jugée les plus importants dans notre application.

.

VI.2.1 Diagramme de cas d'utilisation

Ces diagrammes regroupent les différents acteurs ainsi que les cas d'utilisation. Ils décrivent aussi, sous forme d'actions et de réactions, le comportement d'un système du point de vue d'un utilisateur qui peut être soit un médecin généraliste ou spécialiste

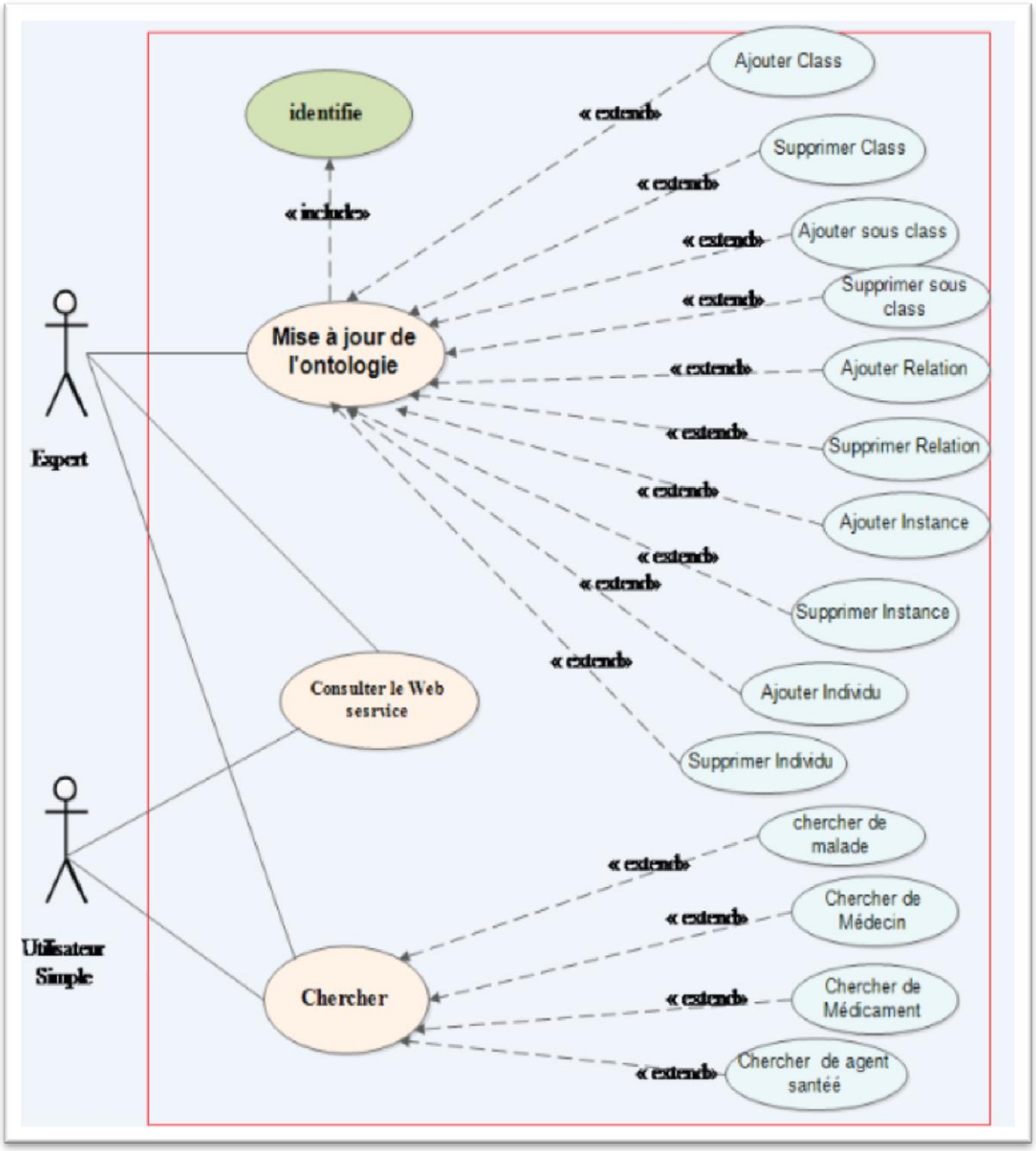

Figure 4.19 : Diagramme de cas d'utilisation

VI.2.2. Diagramme de séquence

Les diagrammes de séquences montrent les interactions existantes entre les différents objets d'un cas d'utilisation, selon un point de vue temporel.

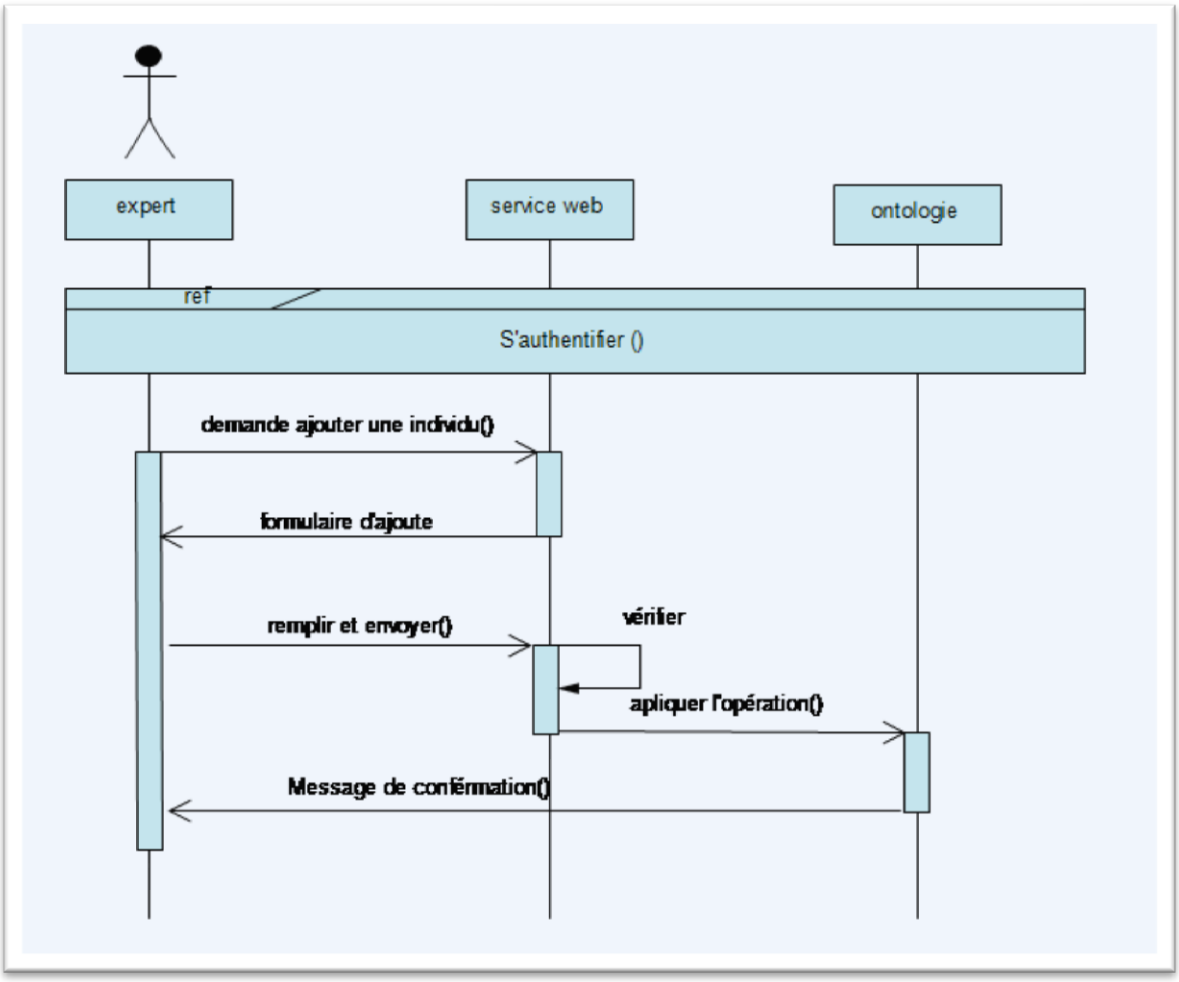

Figure 4.20: Diagramme de séquence (ajouter une instance)

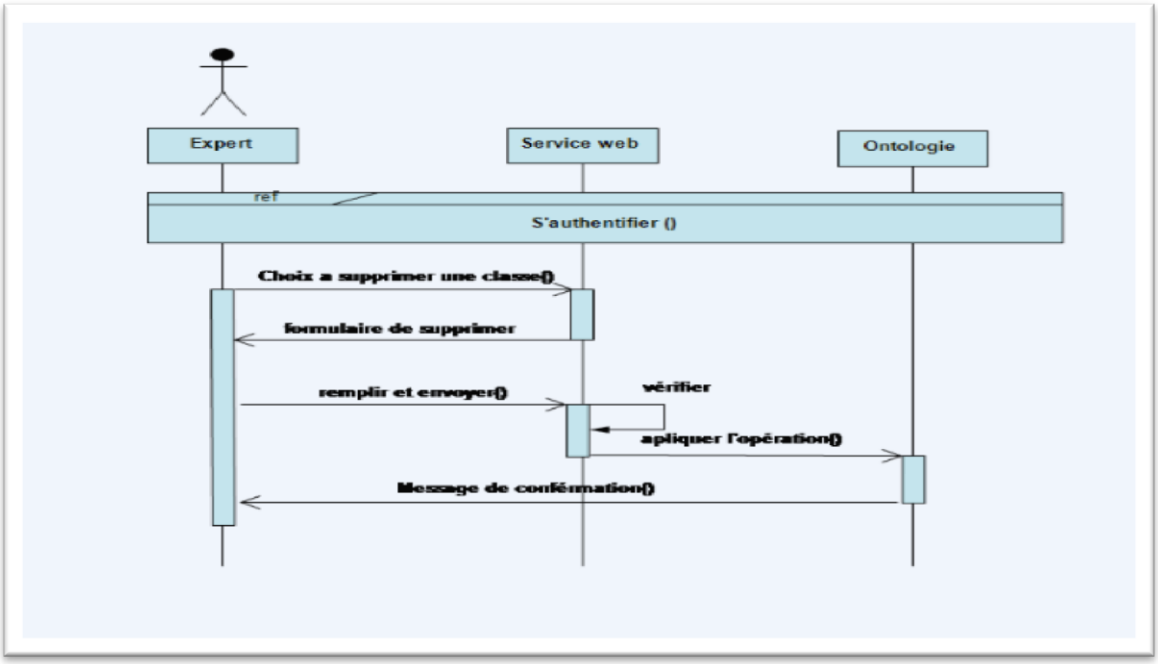

Figure 4.21**:** Diagramme de séquence (supprimer une classe)

VI.2.3. Diagramme de classes

Les diagrammes de classes expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes .

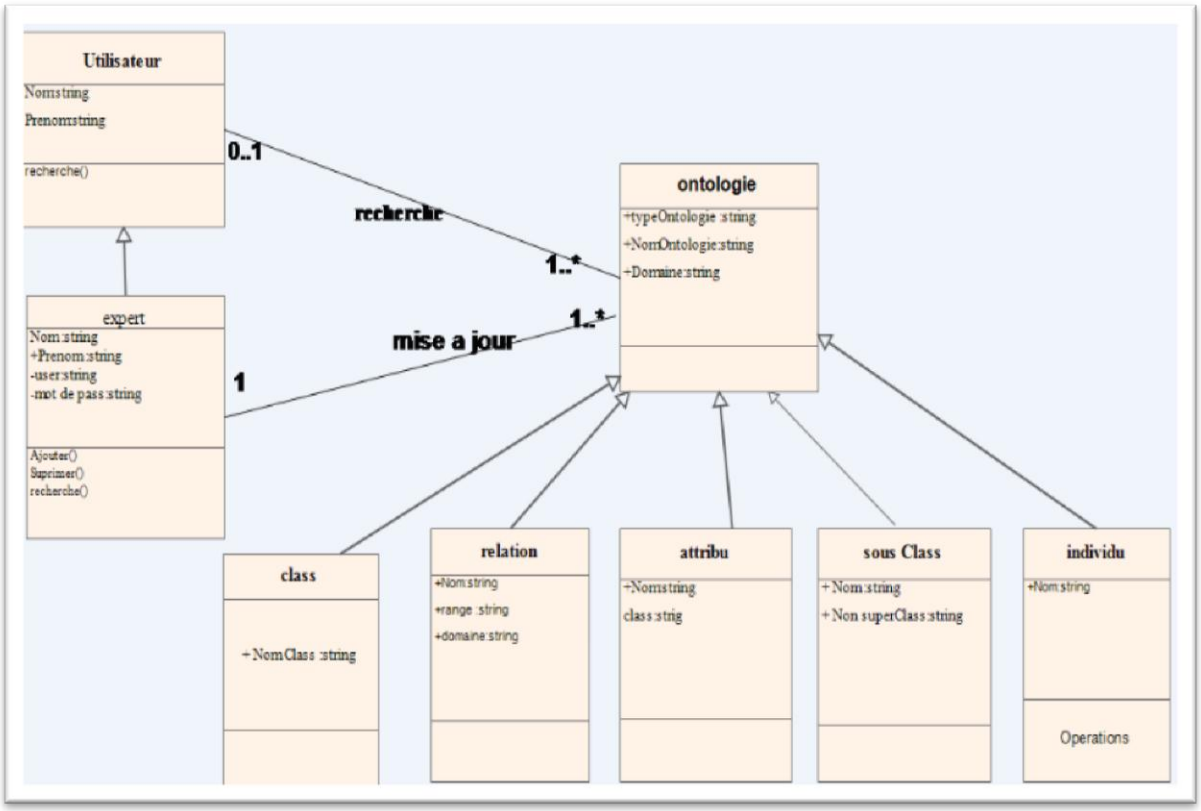

Figure 4.22 : Diagramme de class

Conclusion

Nous avons proposé dans cette phase un système futur que peux devenir notre application en respectant les besoins extraits de l'analyse pour mieux atteindre ainsi les buts fixés.

Dans le chapitre suivant, nous allons présenter la phase de d'implémentation où nous exposerons nos choix concernant la technologie qui a servi à la réalisation de notre application.

Chapitre 05

Implémentation

Introduction

Après avoir établi une étude complète sur les ontologies ce chapitre est consacré à l'implémentation de l'application de ce dernier tout en présentant les langages et les outils utilisés avec une représentation des principales étapes de notre application réalisée ainsi les étapes de création de l'ontologie.

V.1.Inmplémentation de l'ontologie

V.1.1. Les outils et environnement de développement

Avant de commencer l'implémentation de notre application, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblé être un bon choix vu les avantages qu'ils offrent.

V.1.1.1. Plateforme Protégé

PROTEGE OWL est une interface modulaire, développée au StanfordMedicalInformatics de l'Université de Stanford7, permettant l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies [49].

PROTEGE OWL autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie. De nombreux plug-ins sont disponibles ou peuvent être ajoutés par l'utilisateur. L'interface, très bien conçue, et l'architecture logicielle permettant l'insertion de plug-ins pouvant apporter de nouvelles fonctionnalités (par exemple, la possibilité d'importer et d'exporter les ontologies construites dans divers langages opérationnels de représentation tels que OWL ou encore la spécification d'axiomes) ont participé au succès de PROTEGE OWL, qui regroupe une communauté d'utilisateurs très importantes et constitue une référence pour beaucoup d'autres outils [49].

La figure (figure 5.1) présente l'interface de PROTEGE OWL.

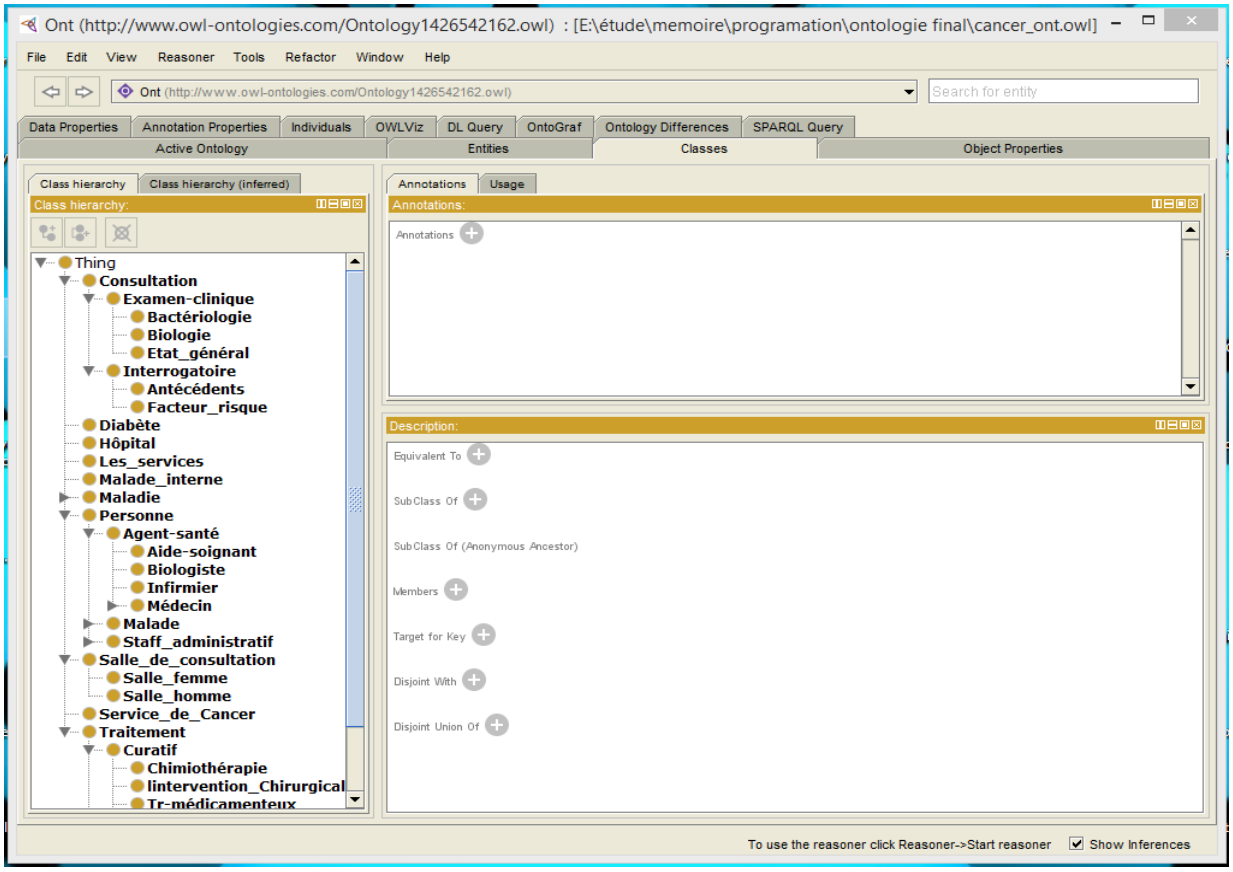

Figure 5.23: l'interface principale de Protégé.

V.1.1.2. Le classifieur Pellet

Le raisonneur Pellet disponible directement depuis Protégé 3.4.4, Pellet est open-source, développer en Java, adapte au raisonnement sur OWL.

Nous exploitons le raisonneur Pellet 1.5.2 disponible directement depuis Protégé 3.4.4.

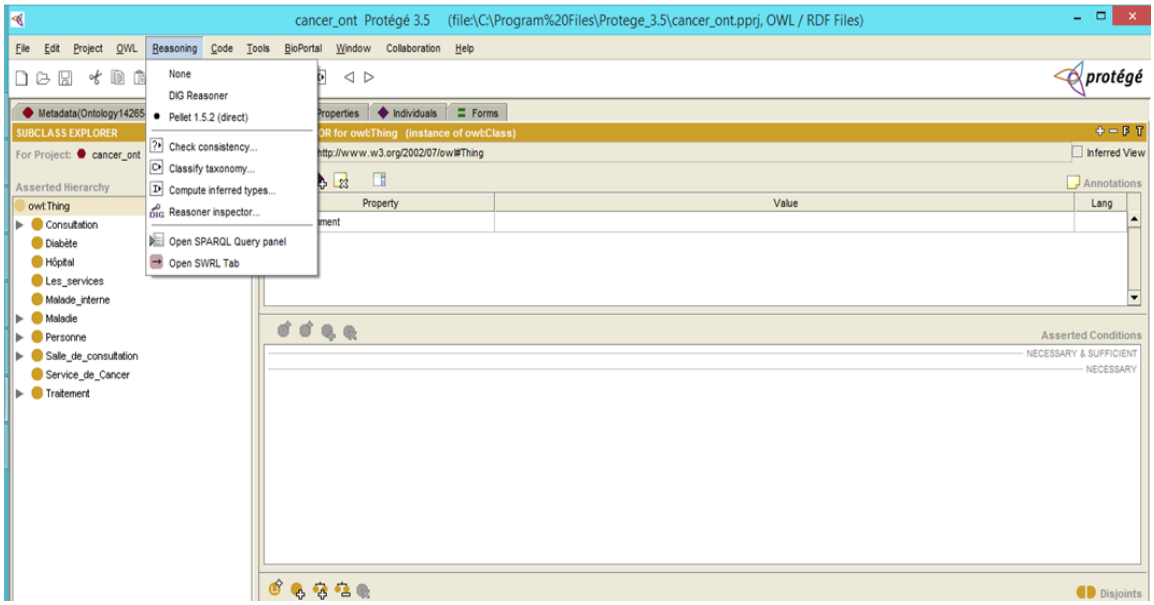

Figure 5.24 : *Le classifieur Pellet*

V.1.1.3. Création de l'ontologie avec Protégé OWL

Nous commençons tout d'abord par la création des concepts spécifiés dans l'étape de con-ceptualisation. Protégé OWL nous offre le moyen de construire la hiérarchie des concepts.

La figure 5.3 présente la procédure de création d'un concept sous Protégé OWL.

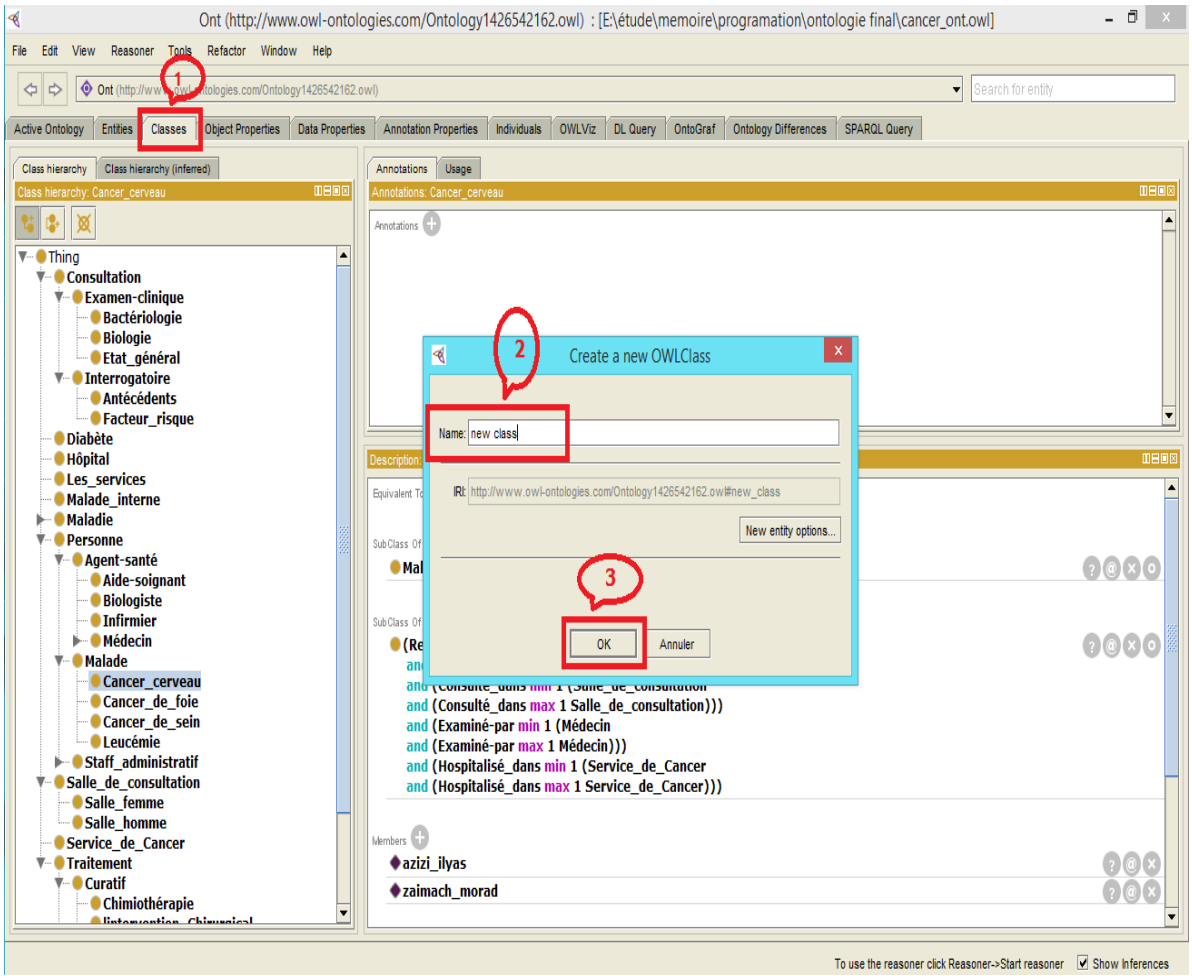

Figure 5.25: Création de classes

V.1.1.4. Création des relations (ObjectProperty)

Pour la création de relations, nous devons spécifier le nom de la relation, le domaine, le co-domaine et la relation inverse si elle existe. Nous pouvons aussi associer à cette définition le type de la relation (symétrique, transitive,…)

La figure 4.8 qui suit montre une définition de la relation "consulter dans".

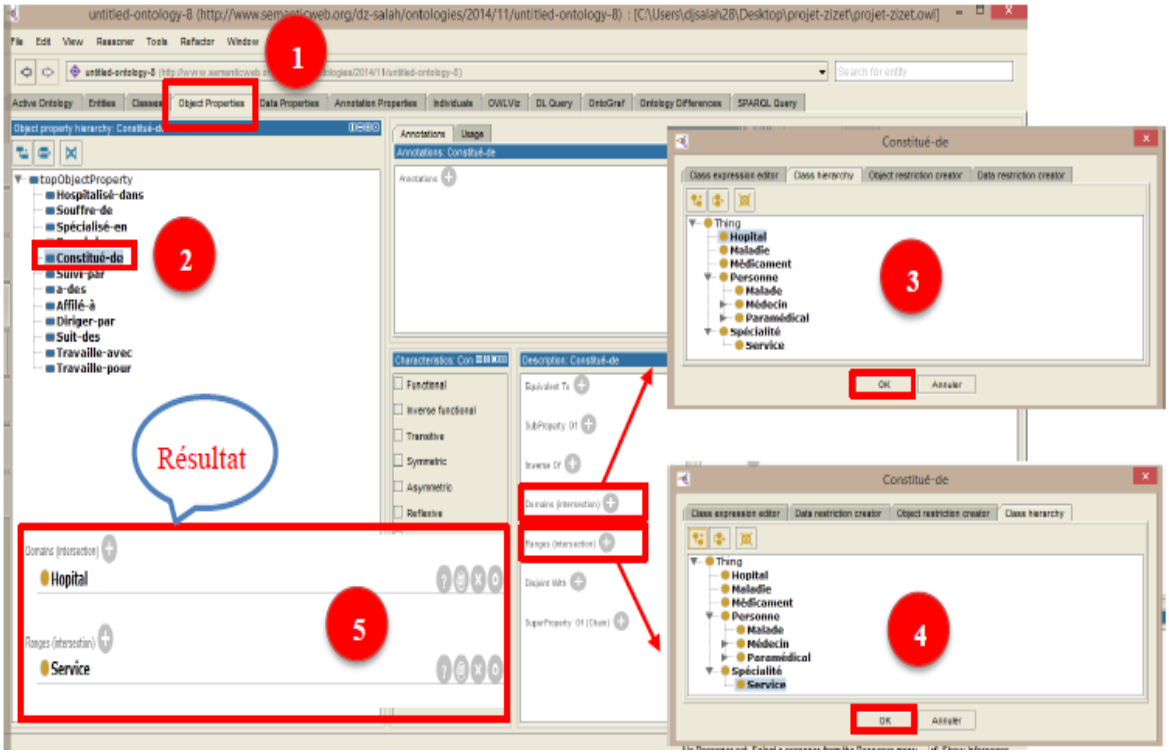

Figure 5.26 : Création de propriétés pour une classe.

V.1.1.5. Création d'attributs (DataTypeProperty)

La création des attributs se fait en cliquant sur le bouton 'dataTypeProperty', ce qui génère la fenêtre présentée par la figure 5.6. Cette fenêtre permet de fournir les champs pour remplir le nom, le domaine, et le type de l'attribut.

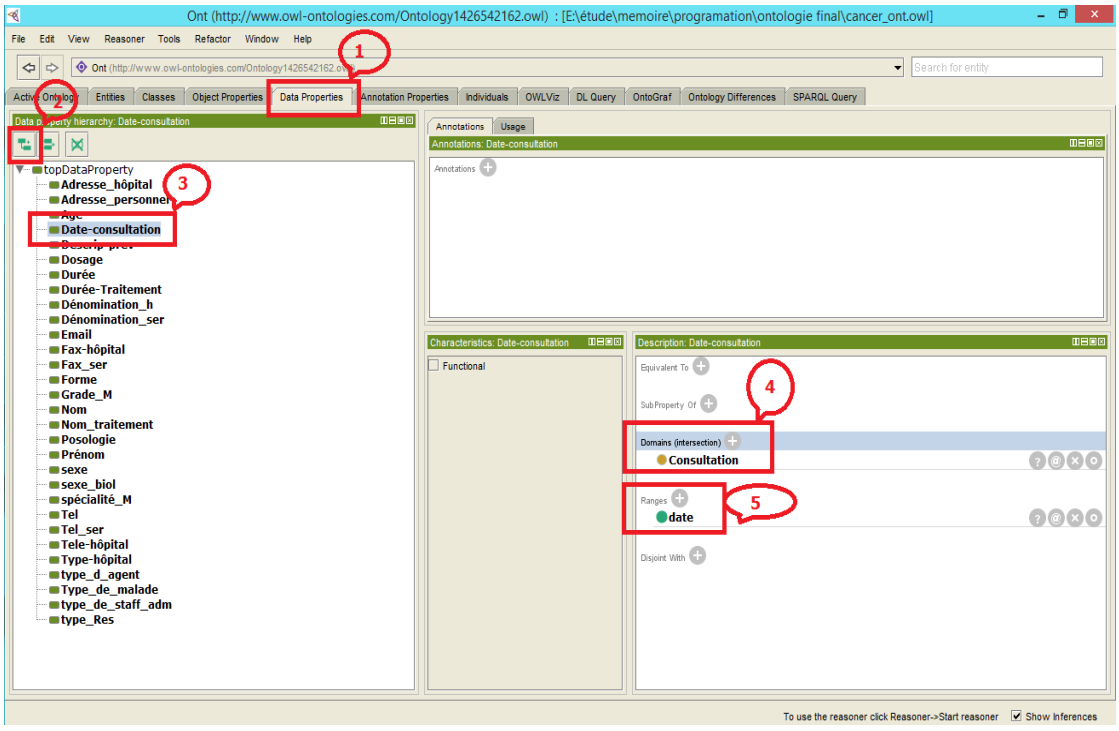

Figure 5.27**:** Création d'un attribut.

V.1.1.6. Création des expressions logiques :

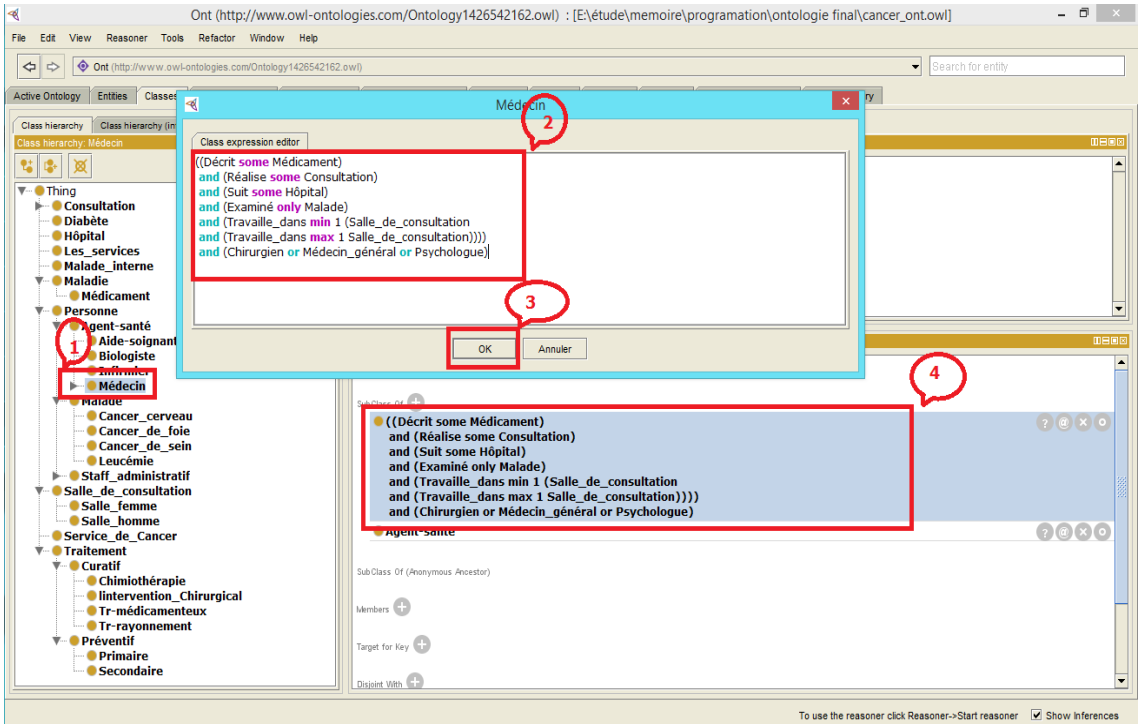

Figure 5.28: Création des expressions logiques pour le concept

V.1.1.7. Instanciation des concepts

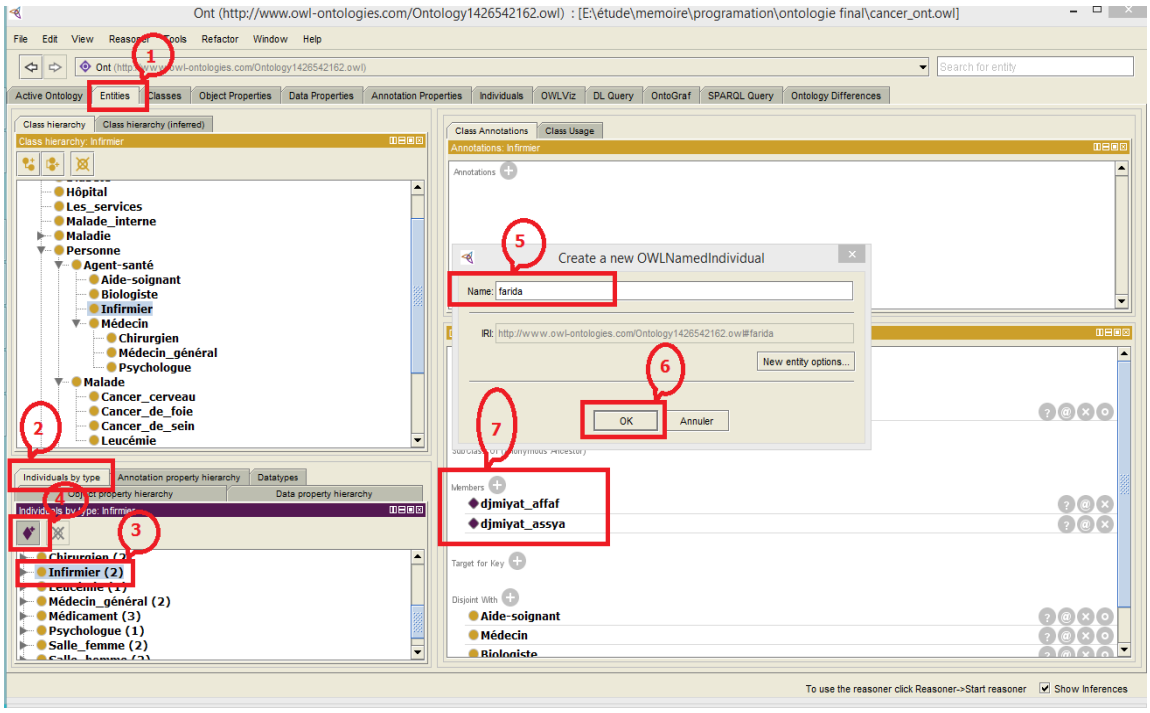

Figure 5.29: Création des instances

V.1.2. Test& évolution de l'ontologie

Nous avons utilisé le système pellet pour tester l'ontologie d'application, nous distinguons deux types de test: test de consistance et de satisfiabilité; le premier type de test consiste à enlever l'inconsistance entre les concepts et cela en utilisant la test de subsumption, par contre le test de satisfiabilité permet de vérifier pour chaque concept l'existence des instances; un concept C est satisfiable si et seulement si, il existe au moins une interprétation I (instance) pour le concept C.

D'après les tests que nous avons appliqués à l'ontologie d'application, aucune erreur n'est produite lors du test.

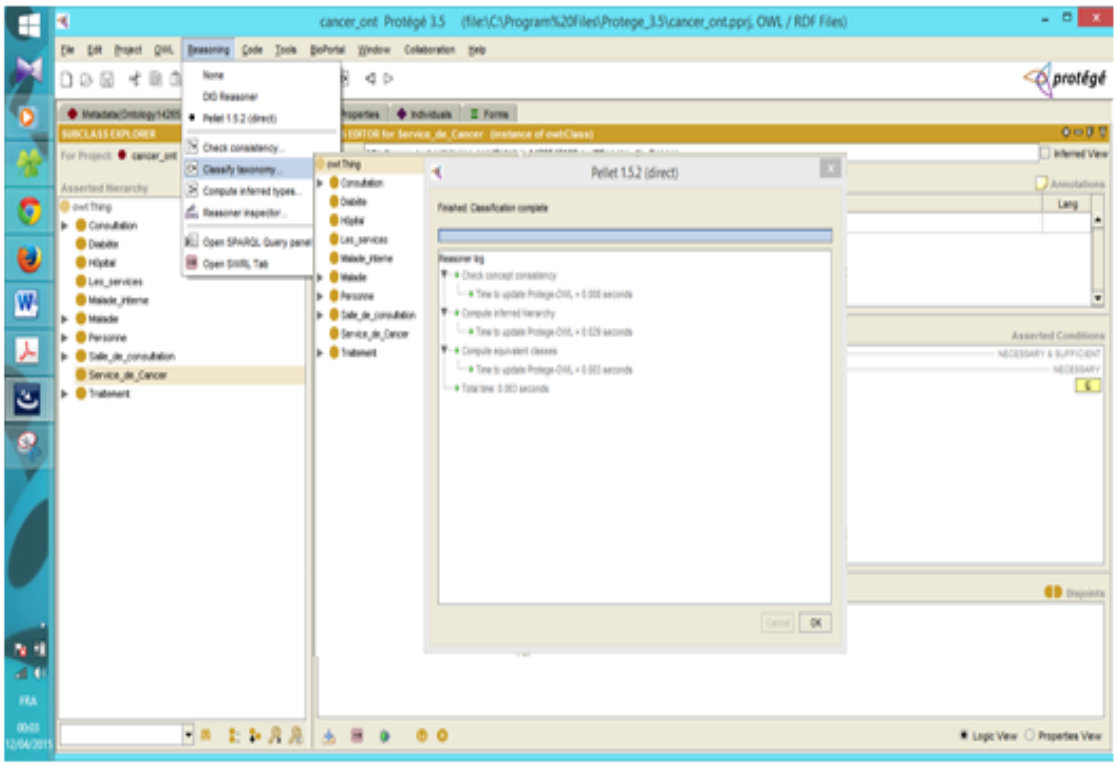

Figure 5.30: Le test de consistance

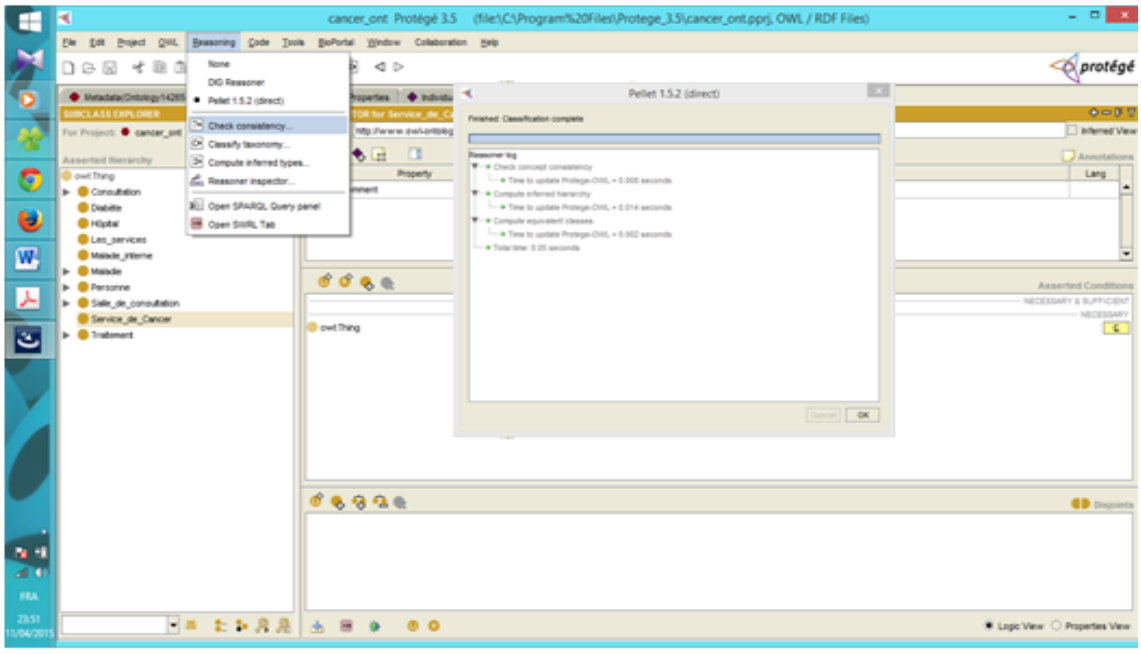

Figure 5.31**:** Le test de consistance

V.2. Implémentation du web service

V.2.1. Outils pour construire de l'application.

Autres que les outils d'implémentation et d'édition d'ontologies, nous pouvons trouver aussi les outils permettant de construire des applications basées sur les ontologies, ils fournissent également un environnement de programmation pour RDF, RDFS et OWL.Parmi ces outils nous pouvons citer langage java et plateforme Jena.

V.2.1.1. Java

Apparu fin 1995 début 1996 et développé par Sun Microsystems Java s'est très rapide ment taillé une place importante en particulier dans le domaine de constructions des applications.

Les objectifs de java sont d'être multi-plateformes et d'assurer la sécurité aussi bien Pendant le développement que pendant l'utilisation d'un programme java.

Java est un langage orienté objet, il permet de construire des applications (bureautiques, graphiques, multimédias, bases de données, environnement de développement, etc...).

Son point fort est la portabilité) à ses bibliothèques de classes indépendantes de la plateforme, ce qui est le point essentiel de la programmation sur internet ou plusieurs machines différents sont interconnectées.

La réalisation multi-plateformes dépend en fait du système d'exploitation et de sa capacité à posséder des outils de compilation et d'interprétation de la machine virtuelle Java.

V.2.1.2. Jena

Jena est une API (application programming interface)Java open source permettant de construire des applications de Web sémantique. Elle contient des classes et des interfaces pour l'interaction avec les modèles RDF et OWL.

Le développement des applications basées sur les ontologies est une tâche très lourde et qui nécessite, préalablement, des infrastructures ou bien des interfaces de programmation déjà construites et qui sont prêtes à l'emploi tel que Jena. De ce fait, nous allons opter la plateforme Jena pour développer notre applications pour visualiser les informations de l'ontologie.

V.2.1.3.La technologie jsp

Nous avons opté pour Les pages Web JSP car c'est une technologie développée par Sun basée sur Java qui simplifie le processus de développement de sites Web dynamiques.

Le mélange d'HTML et de code Java dans les pages JSP permet de séparer la présentation (en HTML) des aspects procéduraux contenus dans le code. On a ainsi une grande souplesse dans le développement de sites W

V.2.1.4.Apache Tomcat

Pour le serveur d'application on a choisi Tomcat qui est un serveur Web qui gère les servlets et les JSP. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur HTTP.

Tomcat est un outil open source qui a été écrit en langage Java, il peut donc s'exécuter via la JVM (machine virtuelle java) sur n'importe quel système d'exploitation la supportant.

V.2.1.5. Eclipse

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate−formemodulaire pour permettre de réaliser des développements informatiques

V.2.2.Les interfaces de l'application

L'interface homme/machine représente l'élément clé dans l'utilisation de tout système informatique. Les interfaces de notre système de recherche sont conçues de manière à être simples, naturelles, compréhensible et d'utilisation faciles.

Pour accéder à notre application, l'utilisateur doit d'abord lancer le serveur web TOMCAT pour se connecter avec l'application du Web service.

L'écran ci-dessous présente l'interface principale de notre application qui contient des informations de notre application et et botton de recherche…etc.

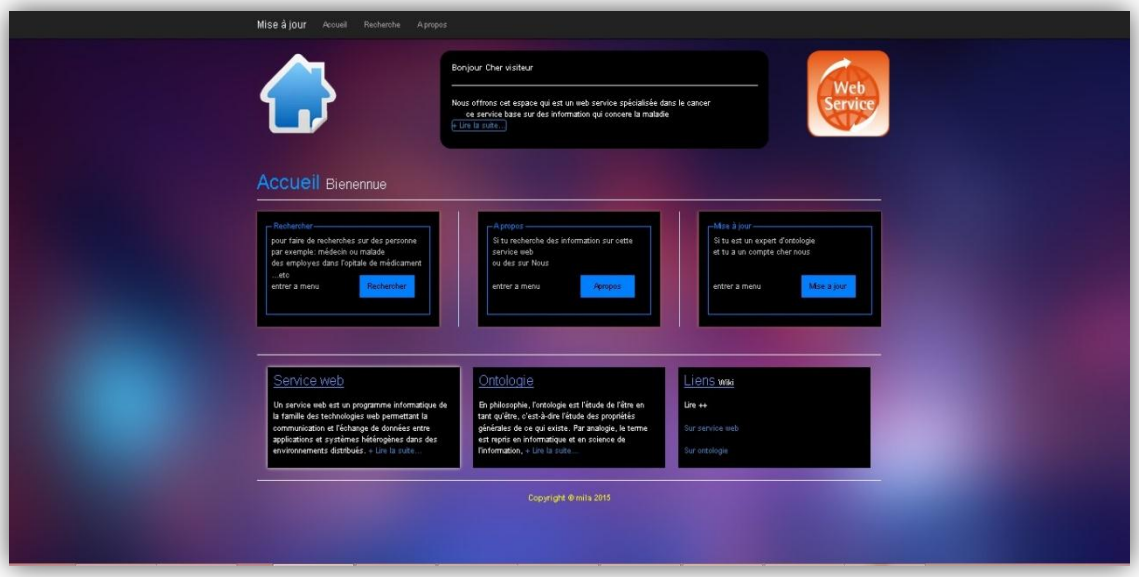

Figure .5.32 : Copie d'écran de l'interface principale.

Après la fenêtre principale, c'est la fenêtre d'authentification qui s'affiche, via cette fenêtre l'expert doit s'authentifier pour bénéficier des fonctionnalités de notre système.

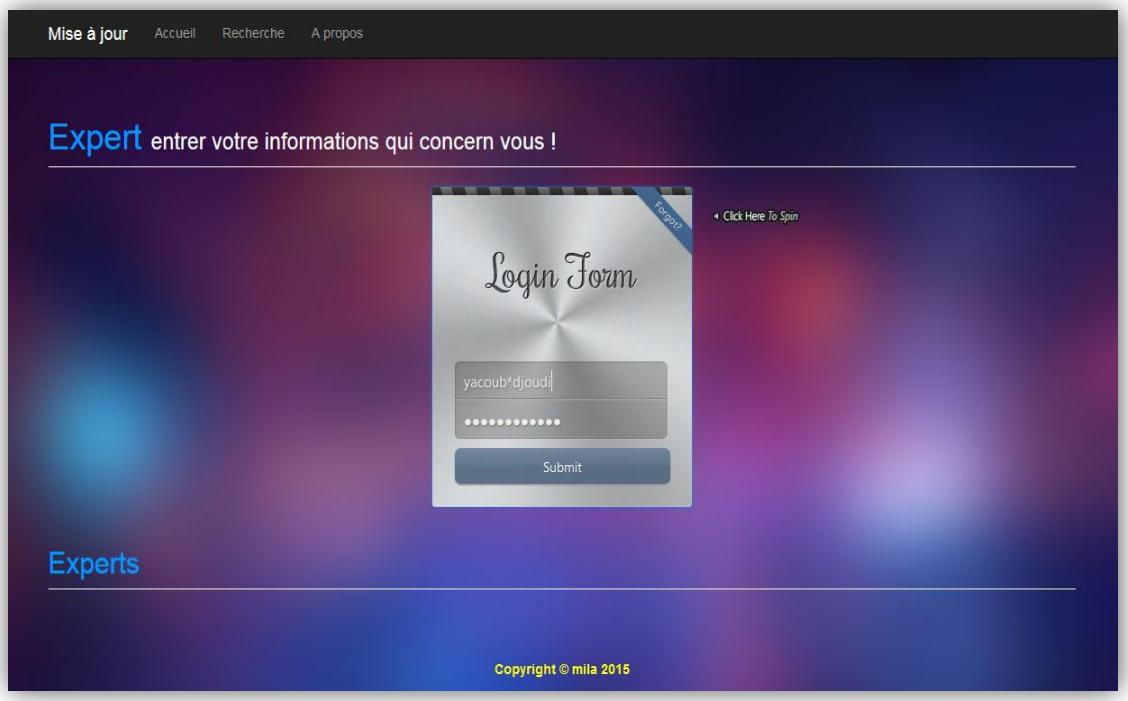

Figure 5.33 : Copie d'écran de d'authentification

L'écran suivant représentent l'ajout des individus par exemple pour aide-soignant on ajoute l'individu Mohamed

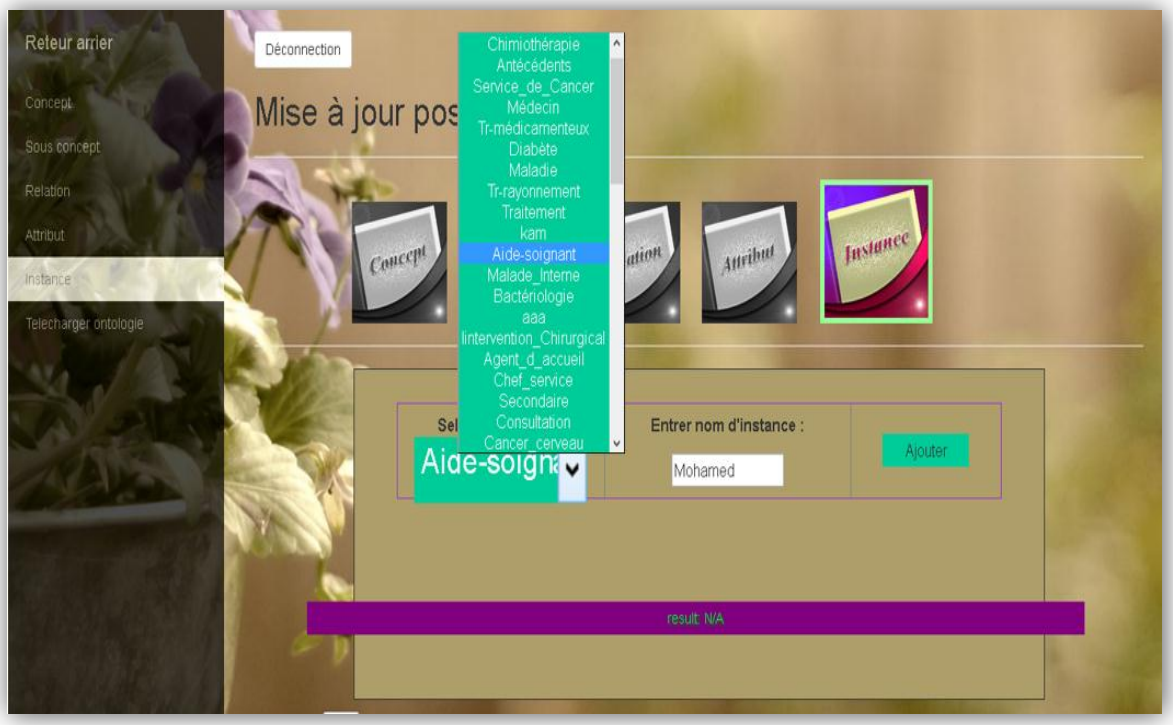

Figure .5. 34 : Copie d'écran d'ajout d'un aide-soignant.

L'ajout des individus permet d'ajouter des relations entre les concepts par exemple on insère une relation « écrit » qui existe entre Médecin et Médicament. Ceci est présenté dans la figure suivante.

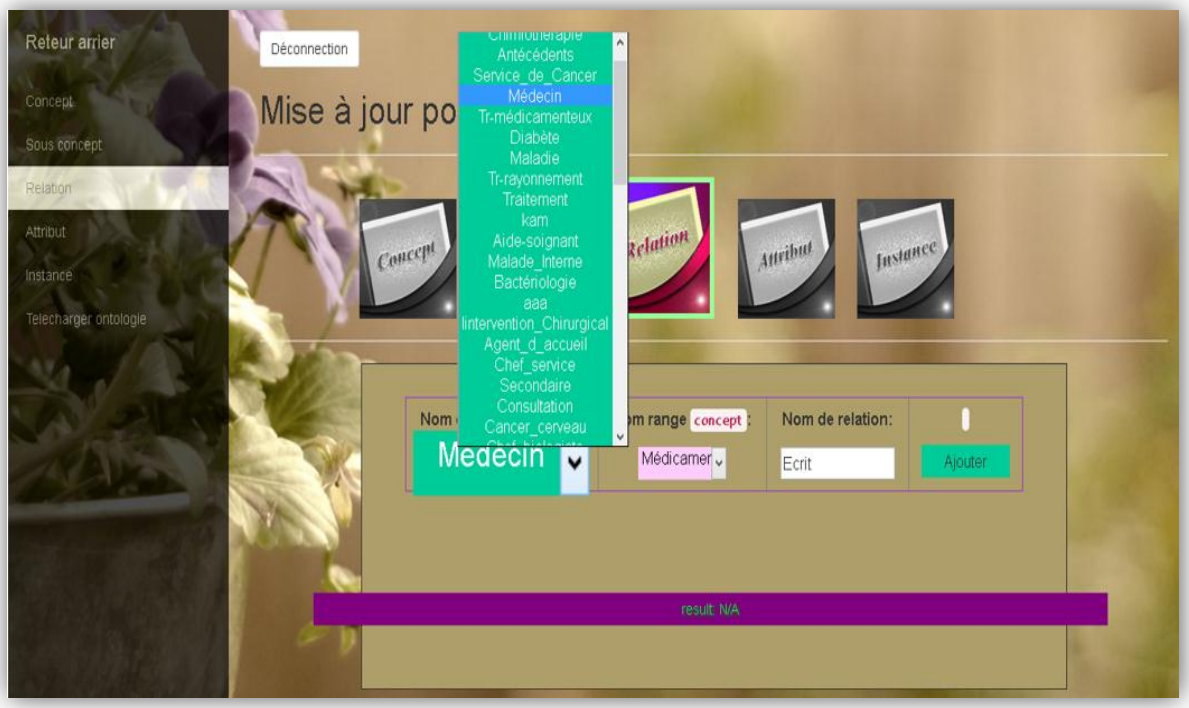

Figure .5.35 : Copie d'écran de création de relation

La figure suivante présente l'ajout des concepts, dans cette étape on ajoute un nouveau concept ou une nouvelle classe « Asperinne »

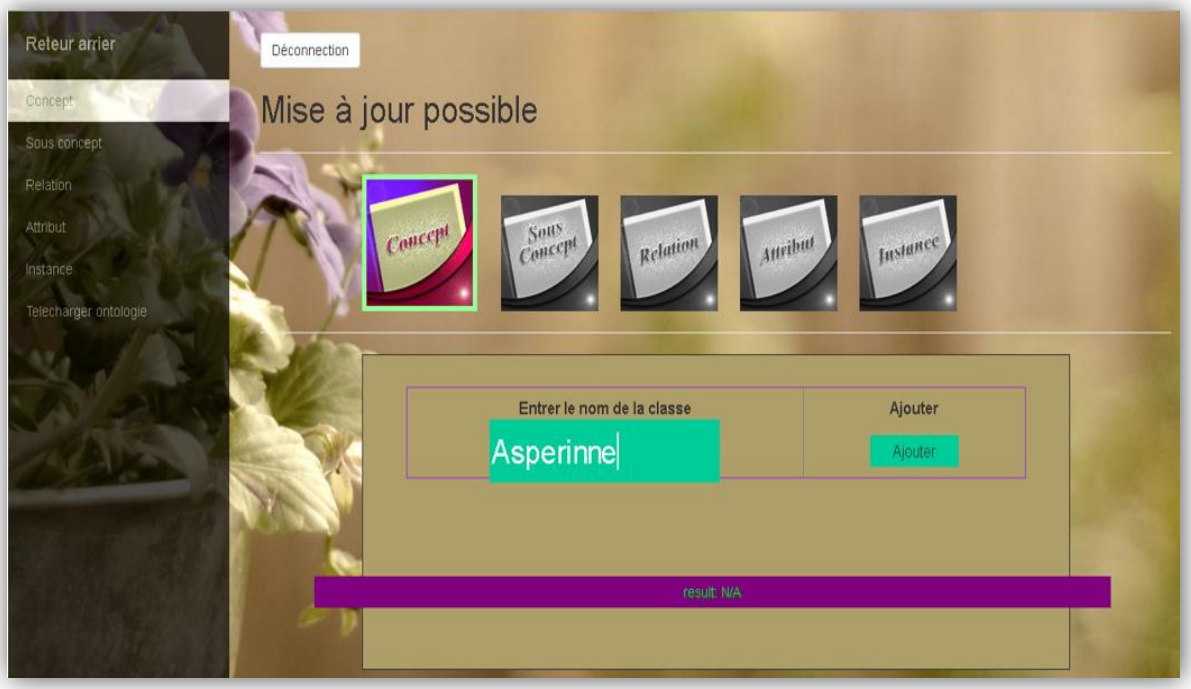

Figure 5.36 : Copie d'écran d'ajout de concept « Asperinne »

Tout comme on peut ajouter des nouveaux concepts on peut aussi en supprimer. La figure suivant présente la suppression du concept « Diabète »

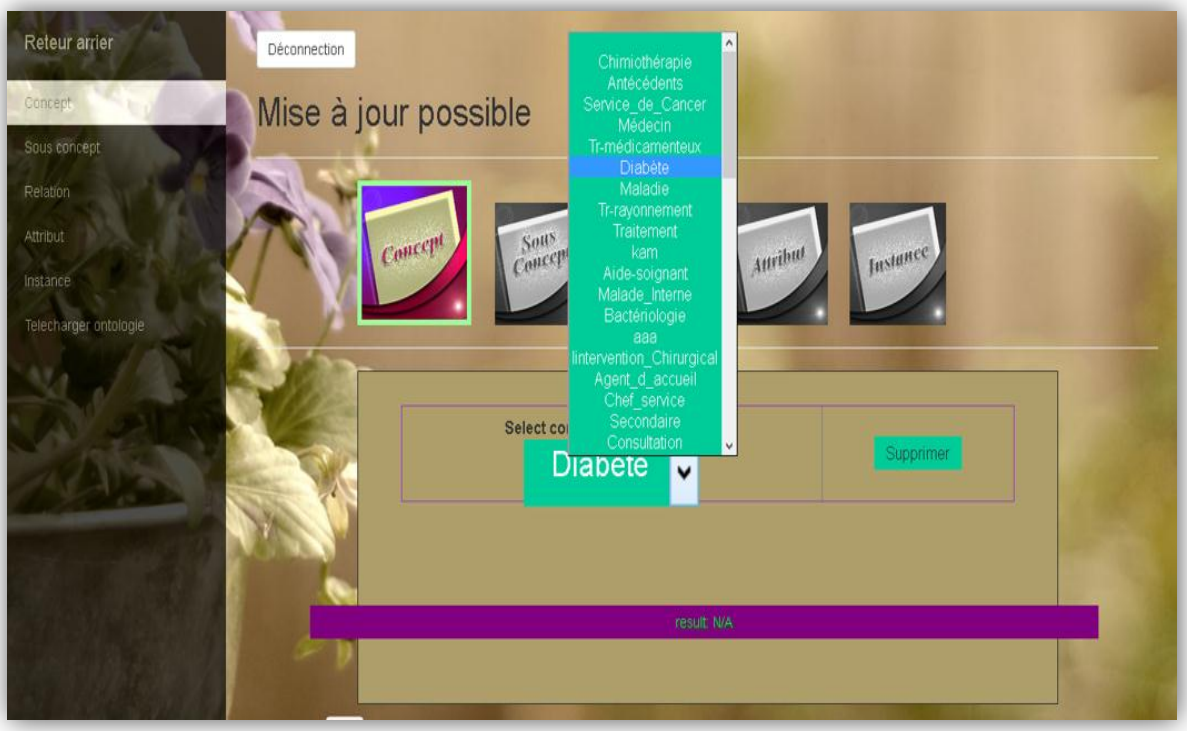

Figure 5.37 : Copie d'écran de suppression de concept « Diabète »

Dans l'écran suivante on représenter la recherche.

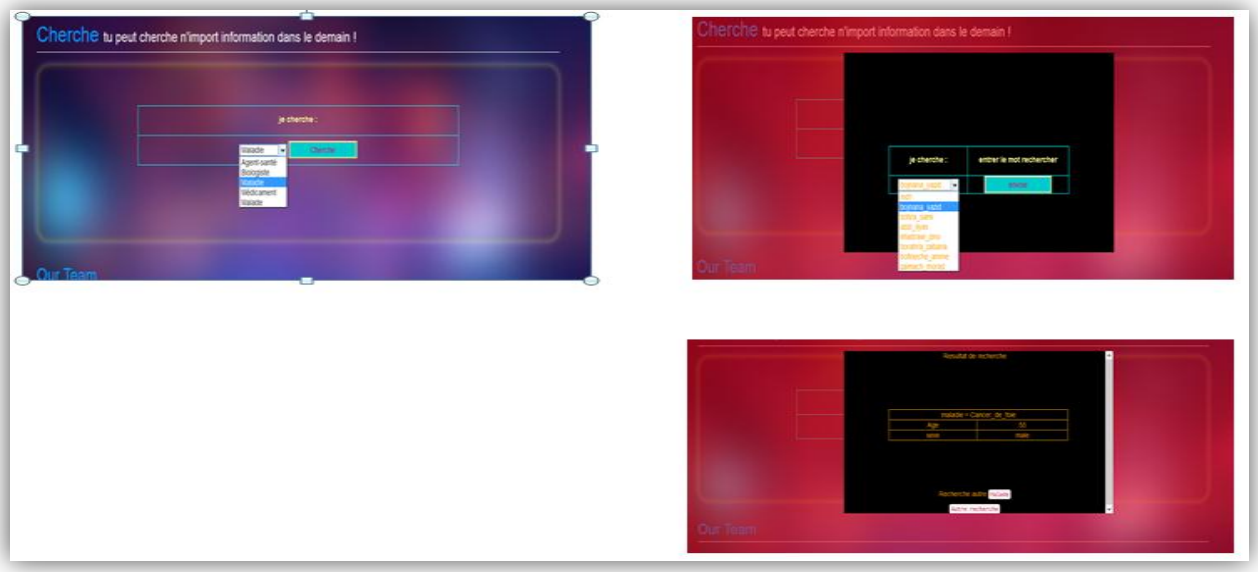

Figure 5.38 : Copie d'écran de recherche

Conclusion :

Nous avons vu - tout au long de ce chapitre – les différents outils et technologies nécessaires pour la réalisation de notre système tels que l'environnement de développement, les langages de programmations, les bibliothèques open source utilisées…etc.

Nous avons montré l'implémentation de chaque composant de notre système proposé, afin d'assurer une présentation claire et détaillée de notre outil. De plus, on a cité et expliqué les caractéristiques de notre application. Finalement, nous avons donné quelques résultats obtenus on utilisant des captures d'écran.

Conclusion

Et

Perspectives

Conclusion générale

Le principe fondateur des Web Services est d'offrir les éléments d'une infrastructure applicative supportant des interactions automatisées entre systèmes hétérogènes.

Dans la première partie de notre travail, nous présenterons la technologie des Web Services (définitions, architectures Orientée Service, Caractéristiques des services web, standards...).

Dans la deuxième partie, nous avons commencé par présenter les ontologies avec leurs caractéristiques, leurs constructions et leurs méthodes d'application, ainsi que les domaines d'application de ces ontologies. Nous avons fait un appariement entre les ontologies afin de sortir avec les meilleures correspondances.

Dans la troisième partie nous avons présenté les processus de construction d'une ontologie, inspiré des différentes phases proposées par la méthode METHONTOLOGY et également de l'exploitation d'autres travaux.

Afin d'atteindre un ensemble de représentations intermédiaires qui facilite sa formalisation ultérieure et cela en adoptant l'approche basée sur la logique de descriptions.

Dans la quatrième partie, Nous avons présenté dans ce chapitre la conception de notre système à travers les différents diagrammes UML

Dans la cinquième partie, nous avons présenté les outils utilisés et les étapes suivies pour l'implémentation de notre système qui est basé sur une ontologie. Nous avons aussi présenté les différentes captures d'écrans montrant les différentes fenêtres de notre système

Enfin, nous ne prétendons guère avoir contourné tout le sujet, plusieurs améliorations et ajouts peuvent être apportés tels que choses que nous espérons réaliser au futur et qui reste également à la portée de certains qui pourraient être intéressés dans le cadre d'un projet de fin d'études ou autre comme l'ajout de l'option de recherche à l'aide de web sémentique, également la modificationdans les critère de l'ontologies on note fait les options d'ajout et de suppression. Toutefois, on peut dire qu'on est satisfaits de nos résultats et souhaitons qu'il soit bien valorisé.

Références

Bibliographiques

Bibliographie

[1] MONFORT .V, & KADIMA.H, les services Web: techniques, démarches et outils, Dunod, 2003

[2] *Mme. Kaouthar FAKHFAKH.*Approche sémantique basée sur les intentiopour la modélisation, la négociation et la surveillance des contrats de qualité de service [*21 Mai 2011*]

[3] Maha Driss, Approche multi-perspective centrée exigences de composition de services Web, thèse de doctorat, Université de Rennes,2011.

[4] K. Hubert , M. Valérie , LES WEB SERVICES, Edition DUNOD, 2003.

[5] Maha Driss, Approche multi-perspective centrée exigences de composition de services Web, thèse de doctorat, Université de Rennes,2011.

[6] A. Aît-Bachir, Archi Med, un canevas pour la détection et la résolution des incompatibilités des conversations entre services web, thèse de doctorat, Université Joseph Fourier-Grenoble 1, 2008.

[7] Julien Guitton. Planification multi-agent pour la composition dynamique de services Web [12 Juin 2006

[8] Mohamed Gharzouli. Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer[25/09/2011]

[9] Céline LOPEZ-VELASCO. Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation [18 novembre 2008]

[10] Amina BEKKOUCHE. Composition des Services Web Sémantiques a base d'Algorithmes Génétiques [2011-2012]

[11] <Razika DRIOUCHE> Proposition d'une architecture d'intégration des applications d'entreprise basée sur l'interopérabilité sémantique de l'EbXML et la mobilité des agents

[12] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.,editors. The Description Logic Handbook: Theory, Implementation and Applications.Cambridge University Press, Cambridge, UK, 2003, 555p.

[13] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer and H. Tompits. ''Reasoning with rules and Ontologies''. In Reasoning Web 2006, volume 4126 of Lecture Notes in Computer Science, pages 93–127. Springer, September 2006.

[14] Natalya F. Noy and Deborah L. McGuinness, "Ontology Development 101: Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[15] R. Neches, R.E. Fikes, T. Finin, T. Gruber, T. Senator, W.R. Swartout, *"Enabling technology for knowledge sharing"*, AI Magazine, 1991.

[16] T. Gruber, "A translation approach to portable ontology specification", 1993.

[17] W. N. Borst, *"Construction of engineering ontologies"*. University of Tweenty, Enschede, Centre for Telematica and Information Technology, 1997.

[18] M. Gruninger and M.S. Fox, "Methodology for the Design and Evaluation of Ontologies". In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, 1995.

[19] N. Guarino, *"Understanding, building, and using ontologies"*. International Journal of Human- Computer Studies, 46: 293-310. 1997.

[20] F. Furst, "L'ingénierie ontologique". Rapport de recherche N°02-07. 2002.

[21] Mohammad Mustafa Taye. Understanding Semantic Web and Ontologies:Theory and Applications [6, June 2010]

[22] Said Izza, Intégration des Systèmes d'Information Industriels Une approche flexible basée sur les services sémantiques, thèse de doctorat, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2006.

[23] G. Van Heijst, A. Schreiber and B. Wielinga. ''*Using explicit ontologies in kbsdevelopment*''. International Journal of Human and Computer Studies – Knowledge Acquisition, 46(2/3), 183–292, 1997.

[24] R. Mizoguchi, M. Ikeda. ''*Towards ontology engineering. In The Joint 1997 Pacific Asian Conference on Expert systems*'' - International Conference on Intelligent Systems, p. 259– 266, Singapore, 1995.

[25] N**.** Guarino. ''*Some organizing principles for a unified top-level ontology*''. In National Conference of the American Association on Artificial Intelligence (AAAI), p. 57–63, Stanford, United-States.

[26] N. Guarino, A. Gangemi, C. Masolo and A. Oltrari. ‘‘*Understanding top-level ontological distinctions*. *In Workshop on Basic Ontological Issues in Knowledge Sharing*’’. The International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[27] N**.** Guarino. ‘‘*Formal ontology in information systems*’’. In the 1st International Conference on Formal Ontology in Information Systems (FOIS), p. 3–15, 1998. [26] BOUFRIDA AMINA et KHELFALLAH YASMINA (Projet de fin d’études 2012) (Université Mentouri de Constantine)

[28] G. Meditskos, N. Bassiliades, “ Combining a DL Reasoner and a Rule Engine for Improving Entailment-based OWL Reasoning”, Proc. 7th International Semantic Web Conference (ISWC-2008), 26-30 Oct 2008, Karlsruhe, Germany, Springer, 2008.

[29] T-R. Gruber. ‘‘Towards Principles for the Design of Ontologies Used for Knowledge Sharing ’’. International Journal Human-Computer Studies, 1995.

[30] M.Uschold&M.Grüninger, “ONTOLOGIES: Principles, Methods andApplications”. Knowledge Engineering Review. 1996

[31] M. Uschold, M. King. ‘‘Towards a methodology for building ontology, in Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing’’, IJCAI’95, 1995.

[32] Abdel kader Keita, Conception Coopérative d’Ontologies Pré-Consensuelles :Application au domaine de l’Urbanisme, thèse de doctorat, Institut national des sciences appliquées Lyon,2007.

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

[34] http://www.xmlenst.fr/van-gogh.xml

[35] RQL: The RDF query language. Site Web: http://139.91.183.30:9090/RDF/RQL/

[36] Florence Amardeilh, Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d’une plateforme logicielle, thès de doctorat, Université Paris X – Nanterre,2007.

[37] Baget J.-F., Canaud E., Euzenat J. & Hacid M.-S., Les langages du Web Sémantique, in Le Web sémantique, , Hors série de la Revue Information Interaction - Intelligence (I3), 4(1), Cépaduès, Toulouse, pp. 21-43,2004.

[38] F.Frédéric, Travail de diplôme 2007, ‘‘*Filière informatique de gestion, Web 3.0 : Interrogation intelligente*’’, 2007.

[39] S. Bechhofer, I. Horrocks, and C. Goble “ *OilEd: a Reason-able Ontology Editor for the Semantic Web*”. In Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, 2001**.**

[40] G. Meditskos, N. Bassiliades, “ *Combining a DL Reasoner and a Rule Engine for Improving Entailment-based OWL Reasoning*”, Proc. 7th International Semantic Web Conference (ISWC-2008), 26-30 Oct 2008, Karlsruhe, Germany, Springer, 2008.

[41] V. Haarslev, R. Moller and M. Wessel, ‘‘*RACER User’s Guide and Reference Manual, version 1.6. Technical report, University of Hamburg*’’, Computer Science Department, 2001.

[42] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, ‘‘ *Pellet: A Practical OWL-DL Reasoner*’’ , University of Maryland, 2005.

[43] < Issam RABHI>Testabilité des services Web[09 janvier 2012]

[44] Julien Guitton. Planification multi-agent pour la composition dynamique des services Web

Glossaire des termes

Liste d'abréviations :

www	World Wide Web
W3C	World Wide Web Consortium
API	Application Programming Interface
HTTP	Hyper Text Transfert Protocol
OIL	Ontology Inference Layer
OWL	Web Ontology Language
Racer	Renamed ABox and Concept Expression Reasoner
OWL-S	Web Ontology Language for Services Web
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SAWSDL	Semantic Annotations for WSDL and XML Schema
SHOE	Simple HTML Ontology Extension
XOL	XML-based Ontology exchange Language
DAML	DARPA Agent Markup Language
FaCT	Fast Classification of Terminologies
XMLS	eXtensible Markup Langage Schema
WSDL	Web Services Description Langage
TOVE	TOronto Virtual Enterprise
DAML+OIL	DARPA Agent Markup Langage + Ontology Inference Layer
SOA	Service Oriented architecture
SOAP	Simple Object Access Protocol
RPC	Remote Procedure call
SW	Service Web
IA	Intelligence Artificielle

Glossaire des termes

UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web consortium
WSMO	Web Service Modeling Ontology
WSDL	Web Service Description Language
WSDL-S	Web Description Language-Semantic
XML	eXtansible Markup Language
GL	Génie Logiciel
IC	Ingénierie des Connaissances
SBC	Système base de connaissance
TOVE	Toronto Virtual Enterprise
GL	Génie Logiciel

Annexe

Dans cette annexe, on va présenter le fichier « SevicePublie.wsdl » qui est le contrat WSDL du web service.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://publier.webs"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://publier.webs" xmlns:intf="http://publier.webs"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://publier.webs"
xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="ajouterSousClass">
    <complexType>
     <sequence>
      <element name="nomHautClasse" type="xsd:string"/>
      <element name="nomSousClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterSousClassResponse">
    <complexType>
     <sequence>
      <element name="ajouterSousClassReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listSousClasses">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listSousClassesResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="listSousClassesReturn"
type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listAttribut">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listAttributResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="listAttributReturn"
type="xsd:string"/>
     </sequence>
```

```
    </complexType>
   </element>
   <element name="listIndividu">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listIndividuResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="listIndividuReturn"
type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="cherInd">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
      <element name="nom" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="cherIndResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="cherIndReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listRelation">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="listRelationResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="listRelationReturn"
type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterClass">
    <complexType>
     <sequence>
      <element name="classe" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterClassResponse">
    <complexType>
     <sequence>
      <element name="ajouterClassReturn" type="xsd:string"/>
     </sequence>
```

```
    </complexType>
   </element>
   <element name="ajouterData">
    <complexType>
     <sequence>
      <element name="nomIndividuData" type="xsd:string"/>
      <element name="att" type="xsd:string"/>
      <element name="val" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterDataResponse">
    <complexType>
     <sequence>
      <element name="ajouterDataReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="getClasseRoot">
    <complexType/>
   </element>
   <element name="getClasseRootResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getClasseRootReturn"
type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="supprimerUneClasse">
    <complexType>
     <sequence>
      <element name="nomClasse" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="supprimerUneClasseResponse">
    <complexType>
     <sequence>
      <element name="supprimerUneClasseReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterUneProprieteObjet">
    <complexType>
     <sequence>
      <element name="nomDomain" type="xsd:string"/>
      <element name="nomRang" type="xsd:string"/>
      <element name="nomPropriete" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="ajouterUneProprieteObjetResponse">
    <complexType>
     <sequence>
      <element name="ajouterUneProprieteObjetReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
```

```
<element name="supprimerUneProprieteObjet">
 <complexType>
  <sequence>
   <element name="nomPropriete" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="supprimerUneProprieteObjetResponse">
 <complexType>
  <sequence>
   <element name="supprimerUneProprieteObjetReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterUneProprieteData">
 <complexType>
  <sequence>
   <element name="nomDomain" type="xsd:string"/>
   <element name="type" type="xsd:string"/>
   <element name="nomPropriete" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterUneProprieteDataResponse">
 <complexType>
  <sequence>
   <element name="ajouterUneProprieteDataReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="supprimerUneProprieteData">
 <complexType>
  <sequence>
   <element name="nomPropriete" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="supprimerUneProprieteDataResponse">
 <complexType>
  <sequence>
   <element name="supprimerUneProprieteDataReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterUnIndividu">
 <complexType>
  <sequence>
   <element name="dansClasse" type="xsd:string"/>
   <element name="nomIndividu" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterUnIndividuResponse">
 <complexType>
  <sequence>
   <element name="ajouterUnIndividuReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
```

```
<element name="ajouterUneProprieteObjetCard">
 <complexType>
  <sequence>
   <element name="nomClasse" type="xsd:string"/>
   <element name="nomPropriete" type="xsd:string"/>
   <element name="op" type="xsd:int"/>
   <element name="num" type="xsd:int"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterUneProprieteObjetCardResponse">
 <complexType>
  <sequence>
   <element name="ajouterUneProprieteObjetCardReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterSousProprieteObjet">
 <complexType>
  <sequence>
   <element name="superPropriete" type="xsd:string"/>
   <element name="nomPropriete" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterSousProprieteObjetResponse">
 <complexType>
  <sequence>
   <element name="ajouterSousProprieteObjetReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterDataUnIndividu">
 <complexType>
  <sequence>
   <element name="nomIndividu" type="xsd:string"/>
   <element name="nomPropriete" type="xsd:string"/>
   <element name="type" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="ajouterDataUnIndividuResponse">
 <complexType>
  <sequence>
   <element name="ajouterDataUnIndividuReturn" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="supprimerUnIndividu">
 <complexType>
  <sequence>
   <element name="dansClasse" type="xsd:string"/>
   <element name="nomIndividu" type="xsd:string"/>
  </sequence>
 </complexType>
</element>
<element name="supprimerUnIndividuResponse">
 <complexType>
  <sequence>
```

```
      <element name="supprimerUnIndividuReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="getTab">
    <complexType/>
   </element>
   <element name="getTabResponse">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="getTabReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>

   <wsdl:message name="getClasseRootResponse">

      <wsdl:part element="impl:getClasseRootResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterUneProprieteObjetResponse">

      <wsdl:part element="impl:ajouterUneProprieteObjetResponse"
name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="getClasseRootRequest">

      <wsdl:part element="impl:getClasseRoot" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterUnIndividuResponse">

      <wsdl:part element="impl:ajouterUnIndividuResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterSousClassResponse">

      <wsdl:part element="impl:ajouterSousClassResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="supprimerUneClasseRequest">
```

```
      <wsdl:part element="impl:supprimerUneClasse" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterSousClassRequest">
      <wsdl:part element="impl:ajouterSousClass" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="cherIndResponse">
      <wsdl:part element="impl:cherIndResponse" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="listRelationResponse">
      <wsdl:part element="impl:listRelationResponse" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterDataUnIndividuResponse">
      <wsdl:part element="impl:ajouterDataUnIndividuResponse"
name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterClassRequest">
      <wsdl:part element="impl:ajouterClass" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterUneProprieteObjetCardResponse">
      <wsdl:part element="impl:ajouterUneProprieteObjetCardResponse"
name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="listSousClassesResponse">
```

```
    <wsdl:part element="impl:listSousClassesResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getTabRequest">
    <wsdl:part element="impl:getTab" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listRelationRequest">
    <wsdl:part element="impl:listRelation" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="supprimerUneProprieteDataRequest">
    <wsdl:part element="impl:supprimerUneProprieteData" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="ajouterSousProprieteObjetRequest">
    <wsdl:part element="impl:ajouterSousProprieteObjet" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="ajouterDataUnIndividuRequest">
    <wsdl:part element="impl:ajouterDataUnIndividu" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getTabResponse">
    <wsdl:part element="impl:getTabResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listAttributResponse">
    <wsdl:part element="impl:listAttributResponse" name="parameters">
    </wsdl:part>
```

```
   </wsdl:message>

   <wsdl:message name="supprimerUneProprieteDataResponse">

      <wsdl:part element="impl:supprimerUneProprieteDataResponse"
name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterDataResponse">

      <wsdl:part element="impl:ajouterDataResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="supprimerUneClasseResponse">

      <wsdl:part element="impl:supprimerUneClasseResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterUneProprieteDataResponse">

      <wsdl:part element="impl:ajouterUneProprieteDataResponse"
name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="ajouterSousProprieteObjetResponse">

      <wsdl:part element="impl:ajouterSousProprieteObjetResponse"
name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="supprimerUnIndividuResponse">

      <wsdl:part element="impl:supprimerUnIndividuResponse" name="parameters">

      </wsdl:part>

   </wsdl:message>

   <wsdl:message name="supprimerUneProprieteObjetResponse">

      <wsdl:part element="impl:supprimerUneProprieteObjetResponse"
name="parameters">
```

```
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listIndividuRequest">
    <wsdl:part element="impl:listIndividu" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="ajouterUneProprieteObjetRequest">
    <wsdl:part element="impl:ajouterUneProprieteObjet" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="supprimerUneProprieteObjetRequest">
    <wsdl:part element="impl:supprimerUneProprieteObjet" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="ajouterUneProprieteObjetCardRequest">
    <wsdl:part element="impl:ajouterUneProprieteObjetCard" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="cherIndRequest">
    <wsdl:part element="impl:cherInd" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="listAttributRequest">
    <wsdl:part element="impl:listAttribut" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="ajouterClassResponse">
    <wsdl:part element="impl:ajouterClassResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
```

```
   <wsdl:message name="ajouterUnIndividuRequest">
      <wsdl:part element="impl:ajouterUnIndividu" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="listSousClassesRequest">
      <wsdl:part element="impl:listSousClasses" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="supprimerUnIndividuRequest">
      <wsdl:part element="impl:supprimerUnIndividu" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterUneProprieteDataRequest">
      <wsdl:part element="impl:ajouterUneProprieteData" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="listIndividuResponse">
      <wsdl:part element="impl:listIndividuResponse" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:message name="ajouterDataRequest">
      <wsdl:part element="impl:ajouterData" name="parameters">
      </wsdl:part>
   </wsdl:message>
   <wsdl:portType name="ServicesPublier">
      <wsdl:operation name="ajouterSousClass">
         <wsdl:input message="impl:ajouterSousClassRequest"
name="ajouterSousClassRequest">
       </wsdl:input>
```

```
         <wsdl:output message="impl:ajouterSousClassResponse"
name="ajouterSousClassResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="listSousClasses">

         <wsdl:input message="impl:listSousClassesRequest"
name="listSousClassesRequest">

       </wsdl:input>

         <wsdl:output message="impl:listSousClassesResponse"
name="listSousClassesResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="listAttribut">

         <wsdl:input message="impl:listAttributRequest"
name="listAttributRequest">

       </wsdl:input>

         <wsdl:output message="impl:listAttributResponse"
name="listAttributResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="listIndividu">

         <wsdl:input message="impl:listIndividuRequest"
name="listIndividuRequest">

       </wsdl:input>

         <wsdl:output message="impl:listIndividuResponse"
name="listIndividuResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="cherInd">

         <wsdl:input message="impl:cherIndRequest" name="cherIndRequest">

       </wsdl:input>

         <wsdl:output message="impl:cherIndResponse" name="cherIndResponse">

       </wsdl:output>
```

```
      </wsdl:operation>

      <wsdl:operation name="listRelation">

         <wsdl:input message="impl:listRelationRequest"
name="listRelationRequest">

       </wsdl:input>

         <wsdl:output message="impl:listRelationResponse"
name="listRelationResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterClass">

         <wsdl:input message="impl:ajouterClassRequest"
name="ajouterClassRequest">

       </wsdl:input>

         <wsdl:output message="impl:ajouterClassResponse"
name="ajouterClassResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterData">

         <wsdl:input message="impl:ajouterDataRequest"
name="ajouterDataRequest">

       </wsdl:input>

         <wsdl:output message="impl:ajouterDataResponse"
name="ajouterDataResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getClasseRoot">

         <wsdl:input message="impl:getClasseRootRequest"
name="getClasseRootRequest">

       </wsdl:input>

         <wsdl:output message="impl:getClasseRootResponse"
name="getClasseRootResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="supprimerUneClasse">
```

```
        <wsdl:input message="impl:supprimerUneClasseRequest"
name="supprimerUneClasseRequest">

       </wsdl:input>

        <wsdl:output message="impl:supprimerUneClasseResponse"
name="supprimerUneClasseResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterUneProprieteObjet">

        <wsdl:input message="impl:ajouterUneProprieteObjetRequest"
name="ajouterUneProprieteObjetRequest">

       </wsdl:input>

        <wsdl:output message="impl:ajouterUneProprieteObjetResponse"
name="ajouterUneProprieteObjetResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="supprimerUneProprieteObjet">

        <wsdl:input message="impl:supprimerUneProprieteObjetRequest"
name="supprimerUneProprieteObjetRequest">

       </wsdl:input>

        <wsdl:output message="impl:supprimerUneProprieteObjetResponse"
name="supprimerUneProprieteObjetResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterUneProprieteData">

        <wsdl:input message="impl:ajouterUneProprieteDataRequest"
name="ajouterUneProprieteDataRequest">

       </wsdl:input>

        <wsdl:output message="impl:ajouterUneProprieteDataResponse"
name="ajouterUneProprieteDataResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="supprimerUneProprieteData">

        <wsdl:input message="impl:supprimerUneProprieteDataRequest"
name="supprimerUneProprieteDataRequest">
```

```
       </wsdl:input>

         <wsdl:output message="impl:supprimerUneProprieteDataResponse"
name="supprimerUneProprieteDataResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterUnIndividu">

         <wsdl:input message="impl:ajouterUnIndividuRequest"
name="ajouterUnIndividuRequest">

       </wsdl:input>

         <wsdl:output message="impl:ajouterUnIndividuResponse"
name="ajouterUnIndividuResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterUneProprieteObjetCard">

         <wsdl:input message="impl:ajouterUneProprieteObjetCardRequest"
name="ajouterUneProprieteObjetCardRequest">

       </wsdl:input>

         <wsdl:output message="impl:ajouterUneProprieteObjetCardResponse"
name="ajouterUneProprieteObjetCardResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterSousProprieteObjet">

         <wsdl:input message="impl:ajouterSousProprieteObjetRequest"
name="ajouterSousProprieteObjetRequest">

       </wsdl:input>

         <wsdl:output message="impl:ajouterSousProprieteObjetResponse"
name="ajouterSousProprieteObjetResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="ajouterDataUnIndividu">

         <wsdl:input message="impl:ajouterDataUnIndividuRequest"
name="ajouterDataUnIndividuRequest">

       </wsdl:input>
```

```
         <wsdl:output message="impl:ajouterDataUnIndividuResponse"
name="ajouterDataUnIndividuResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="supprimerUnIndividu">

         <wsdl:input message="impl:supprimerUnIndividuRequest"
name="supprimerUnIndividuRequest">

       </wsdl:input>

         <wsdl:output message="impl:supprimerUnIndividuResponse"
name="supprimerUnIndividuResponse">

       </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getTab">

         <wsdl:input message="impl:getTabRequest" name="getTabRequest">

       </wsdl:input>

         <wsdl:output message="impl:getTabResponse" name="getTabResponse">

       </wsdl:output>

      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="ServicesPublierSoapBinding" type="impl:ServicesPublier">

      <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="ajouterSousClass">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="ajouterSousClassRequest">

            <wsdlsoap:body use="literal"/>

         </wsdl:input>

         <wsdl:output name="ajouterSousClassResponse">

            <wsdlsoap:body use="literal"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="listSousClasses">
```

```
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="listSousClassesRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="listSousClassesResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="listAttribut">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="listAttributRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="listAttributResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="listIndividu">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="listIndividuRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="listIndividuResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="cherInd">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="cherIndRequest">
```

```
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="cherIndResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="listRelation">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="listRelationRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="listRelationResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ajouterClass">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="ajouterClassRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="ajouterClassResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ajouterData">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="ajouterDataRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="ajouterDataResponse">
```

```
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getClasseRoot">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="getClasseRootRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="getClasseRootResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="supprimerUneClasse">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="supprimerUneClasseRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="supprimerUneClasseResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterUneProprieteObjet">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="ajouterUneProprieteObjetRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="ajouterUneProprieteObjetResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
```

```
    </wsdl:operation>
    <wsdl:operation name="supprimerUneProprieteObjet">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="supprimerUneProprieteObjetRequest">
          <wsdlsoap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="supprimerUneProprieteObjetResponse">
          <wsdlsoap:body use="literal"/>
       </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterUneProprieteData">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="ajouterUneProprieteDataRequest">
          <wsdlsoap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="ajouterUneProprieteDataResponse">
          <wsdlsoap:body use="literal"/>
       </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="supprimerUneProprieteData">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="supprimerUneProprieteDataRequest">
          <wsdlsoap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="supprimerUneProprieteDataResponse">
          <wsdlsoap:body use="literal"/>
       </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterUnIndividu">
       <wsdlsoap:operation soapAction=""/>
```

```
        <wsdl:input name="ajouterUnIndividuRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="ajouterUnIndividuResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterUneProprieteObjetCard">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="ajouterUneProprieteObjetCardRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="ajouterUneProprieteObjetCardResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterSousProprieteObjet">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="ajouterSousProprieteObjetRequest">
            <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="ajouterSousProprieteObjetResponse">
            <wsdlsoap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ajouterDataUnIndividu">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="ajouterDataUnIndividuRequest">
            <wsdlsoap:body use="literal"/>
```

```
         </wsdl:input>
         <wsdl:output name="ajouterDataUnIndividuResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="supprimerUnIndividu">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="supprimerUnIndividuRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="supprimerUnIndividuResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getTab">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="getTabRequest">
            <wsdlsoap:body use="literal"/>
         <wsdl:output name="getTabResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="ServicesPublierService">
      <wsdl:port binding="impl:ServicesPublierSoapBinding"
name="ServicesPublier">
         <wsdlsoap:address
location="http://localhost:8080/FinalCOGWS/services/ServicesPublier"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```