République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes

Département d'Informatique
École Doctorale Réseaux et Systèmes Distribués

Mémoire de Magistère

En Informatique

Option : Réseaux et Systèmes Distribués

Thème :

Annotation et interrogation de données non structurées : Application aux Services Web

Présenté par :
Amel BENNA

Devant le jury composé de :

Président	Djamil AISSANI	Professeur	U. Béjaïa, Algérie
Rapporteur	Nacer BOUDJLIDA	Professeur	U. Nancy, France
Examinateur	Mahmoud BOUFAIDA	Professeur	U. Constantine, Algérie
Examinateur	Nadjib BADACHE	Professeur	USTHB, Algérie
Invitée	Hassina TALANTIKITE	Chargée de cours	U. Béjaïa, Algérie

Bejaïa, Mai 2008

REMERCIEMENTS

Je remercie avant tout, le bon Dieu tout puissant de m'avoir donné la patience d'effectuer ce modeste travail.

Je tiens à remercier tout particulièrement mon promoteur, Pr. Nacer Boudjlida, pour m'avoir permis de travailler dans un domaine si passionnant, pour ses conseils judicieux, ses relectures attentives et critiques, pour sa rigueur et son souci de clarté qui m'ont aidé à améliorer la qualité de ce travail.

Je le remercie pour sa disponibilité, malgré son planning toujours très serré, il n'a pas hésité à me consacrer des séances de travail lors de ses déplacements en Algérie pour améliorer et corriger un article, une présentation, établir ou organiser le plan du mémoire.

J'ai particulièrement apprécié l'intérêt qu'il a porté à mon travail et de m'avoir encouragé à soumettre des articles.

Merci à ma co-promotrice, Mme Hassina Talantikite, pour avoir co-encadré ce travail et d'avoir insisté pour le choix du sujet.

Merci au président et aux membres du jury pour m'avoir honoré en consentant à juger mon travail.

Merci à Mme Farida Admane, qui est à la source de ma démarche de magister, et à Mr Abdelkrim Tari responsable du département informatique de m'avoir accueilli en 2ème année magistère à l'université de Béjaïa.

Merci à Assia Arab, étudiante en master Béjaïa, pour tous ses courriels sur les procédures administratives.

Finalement j'adresse un grand merci du fond du cœur à toute ma chère famille, à tous mes collègues et amis qui ont toujours été présents lorsque j'en ai eu besoin et pour leur soutien moral.

RESUME :

Les services Web ont été adoptés par les plus grandes organisations industrielles et commerciales pour l'interopérabilité des systèmes distribués. L'interaction entre leurs composants s'effectue à travers la publication, la recherche et la découverte puis l'invocation. Cependant, leur publication dans l'UDDI ne propose pas une description du service sur la base de ce qu'il offre. L'idée rapportée dans ce document, est d'effectuer une description sémantique des services Web (en SAWSDL) et un stockage des fonctionnalités des services Web, lors de la phase de publication, dans un registre que nous avons appelé base de liens sémantiques. La découverte des services revient à l'interrogation de cette base, qui représente le médiateur entre demandeur et fournisseur de services, et du registre UDDI. Le résultat retourné peut servir de point d'entrée pour une autre interrogation.

MOTS-CLÉS : annotation sémantique, découverte des services Web, SAWSDL, services Web sémantiques, OWL, XML, XQUERY.

ABSTRACT:

Web services have been adopted by a large number of industrial and commercial organizations for the purpose of interoperability between distributed systems. The interaction between Web services components is done thanks to publication, research, discovery and then invocation. However, the publication of the services in an UDDI registry doesn't offer enough description of the service offering. The main idea described in this paper is about having a semantic description of Web services (using SAWSDL) and then storing their functionality, during the publication phase, in a register named 'semantic links base'. This base represents the mediator between the service requestor and the service provider. The discovery step queries this base in addition to the UDDI registry. The result returned can then be used as entry point for a further query.

KEYWORDS: semantic annotation, Web services discovery, SAWSDL, semantic Web services, OWL, XML, XQUERY.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

L'évolution du marché et la concurrence favorisent l'innovation. Elles incitent les entreprises à développer leurs relations avec leurs clients et partenaires et à améliorer leurs productivités pour un gain en efficacité.

Une architecture orientée services (Service Oriented Architecture ou SOA) est vue par les analystes de l'industrie mais aussi chez les spécialistes de la question, les plus grands éditeurs (Microsoft, IBM, Oracle, SAP,…) et constructeurs, comme la solution « miracle» pour réduire les coûts, réagir plus vite et mieux répondre aux besoins des clients.

SOA préconise avant tout une nouvelle façon de penser un système d'information. Le Gartner Group a estimé qu'en 2008, 80 % des projets informatiques se construiraient sur la base de SOA.

Actuellement, les services Web sont les technologies les plus prometteuses qui reposent sur SOA [44]. Ils proposent une façon de développer des applications/systèmes distribués en les rendant interopérables quels que soient leurs plateformes ou leurs langages.

L'architecture de référence des services Web se base sur l'interaction entre trois services : *l'Annuaire de services, le Fournisseur de services et le Demandeur de services*. Cette interaction s'effectue à travers trois opérations : (i) la publication de la description de services, (ii) la recherche et la découverte de la description du service et (iii) l'invocation du service basée sur sa description.

Pour aider les entreprises, l'annuaire de services référence les services pour les exposer au « reste du monde ». Chaque *demandeur de services* dispose ainsi d'une vue sur l'ensemble des services publiés. Comme l'ensemble des services Web publiés augmente, il devient de plus en plus important d'avoir des outils automatisés pour aider à découvrir les services recherchés par le *demandeur de services*.

Découvrir automatiquement des services Web repose sur les facilités offertes aux fournisseurs de services pour décrire les fonctionnalités de leurs services et aux demandeurs de services à décrire leurs besoins sans ambiguïté.

Cependant, l'aspect sémantique indispensable pour automatiser certaines tâches y est absent. L'ajout de la sémantique pour représenter les besoins et les fonctionnalités des services Web devient essentiel.

Le Gartner signale que d'ici à 2012, à propos de la taxonomie et de la connaissance hiérarchique, l'indexation et les mises en correspondance seront répandues dans presque toutes les applications riches en information. Les technologies Web, les communications unifiées, la gestion des processus métiers, et la gestion des métadonnées[1] sont

[1] Les méta-données sont des données au sujet des données : elles sont compréhensibles et traitables automatiquement par des machines.

considérées, également par le Gartner, parmi les dix technologies qui devraient dans les trois ans à venir, avoir un impact significatif sur l'activité et l'organisation des entreprises et pour lesquels le cabinet américain conseille d'investir massivement.

L'annotation est l'une des formes les plus communes de *métadonnées* dans le contexte du Web [12]. L'annotation sémantique revient à assigner à des entités dans le texte un lien à leur description sémantique. Les modèles sémantiques fournissent un accord sur les termes et l'usage des termes : il y aura moins d'ambiguïté dans l'intention sémantique du fournisseur et du client. L'homme peut associer un mot à un contexte pour lui donner un sens mais ceci reste toujours difficile pour une machine.

L'objectif de ce travail et de présenter une manière d'associer des annotations sémantiques à des services Web par le *fournisseur de services* pour permettre une description sémantique du service, et une approche d'interrogation des services par le *demandeur de services* pour la découverte automatique des services Web.

Pour réaliser cet objectif, on doit tout d'abord déterminer les objets à annoter dans la description du service, pour connaître les fonctionnalités de service Web, et le modèle sémantique de référence pour représenter la cible des annotations, mais il faudra également déterminer où et comment stocker ces annotations. La découverte des services revient à une exploitation de sources de données semi-structurées pour trouver une mise en correspondance entre les termes utilisés par le *demandeur de services* et les annotations des fonctionnalités du service décrites par le *fournisseur de services*. Pour cela, nous nous appuyons sur un langage d'interrogation de données semi-structurées.

Organisation du document

Notre document est réparti en deux parties : dans la première partie, nous présentons le cadre dans lequel s'inscrit notre travail, en décrivant tout d'abord le paradigme des services Web dans le chapitre 1, l'annotation sémantique des services Web dans le chapitre 2 et l'exploration des services Web dans le chapitre 3.

La seconde partie de ce document est réservée à la description de notre approche d'annotation, de publication et d'interrogation des services Web dans le chapitre 4 et sur un scénario de mise en œuvre de la solution proposée dans le chapitre 5. Une conclusion et quelques perspectives clôturent ce document.

Chapitre 1 : Le paradigme service Web

1. Introduction

Les services Web sont un paradigme qui propose une façon de développer des applications distribuées en les rendant interopérables quels que soient leur plate-forme ou langage.

Leur principale spécificité est qu'ils s'appuient sur les technologies standards d'Internet en les utilisant comme infrastructure pour leur description et communication.

Ce chapitre constitue la première partie de l'état de l'art de notre travail dans lequel nous tâcherons de présenter la définition, l'architecture et l'infrastructure de base des services Web. Nous citerons ensuite quelques outils et plates-formes de leur développement.

2. Définition Des Services Web

Plusieurs définitions ont été avancées pour les services Web. Nous présentons celle du consortium W3C[2] qui définit un service Web comme suit :

- Un Web service est un système logiciel identifié par une URI[3] dont les interfaces publiques et les liens sont définis et décrits en XML[4].
- Sa définition peut être découverte par d'autres systèmes logiciels.
- Ces systèmes peuvent interagir avec le service Web d'une manière prescrite par sa définition en utilisant des messages XML portés par les protocoles Internet.

En d'autres termes, les services Web sont des programmes qui permettent aux applications de communiquer entre elles et d'utiliser les fonctionnalités proposées en échangeant des messages et ceci indépendamment des langages de programmation et des plates-formes d'exécution.

3. Fonctionnement des services Web

Les services Web sont une instance de l'Architecture Orientée Service (SOA).

SOA admet de gérer l'hétérogénéité des applications en permettant à des applications ou services de communiquer et de travailler ensemble, quelles que soient leur plates-forme ou leur localisation. Ces principaux composants sont les messages échangés, les agents[5] qui agissent comme des demandeurs ou des fournisseurs de services et les mécanismes de transport qui permettent le flux de messages.

L'architecture des services Web est une architecture qui définit les éléments globaux qui assurent l'interopérabilité des services Web. Son but est de décrire un ensemble minimum de caractéristiques qui doivent être communes à tous les services.

Nous allons commencer par présenter l'architecture de base utilisée pour le fonctionnement de services Web isolés pour passer ensuite à l'architecture étendue souvent utilisée lors de la composition de services Web.

---

[2] W3C : World Wide Web Consortium voir http://www.3w.org
[3] URI : Uniform Resource Identifier est une courte chaine de caractères qui identifie les ressources sur le Web:
[4] XML : eXtended Markup Language
[5] Agents : entités concrètes qui implémentent un service (abstrait) et communiquent par l'échange de messages

3.1 Fonctionnement des services Web

Comme illustré dans la figure1. Le fonctionnement des services Web se base sur l'interaction entre trois services : l'*Annuaire de services* (Services Registry), le *Fournisseur de services* (Services Provider) et le *Demandeur de services* (Services Requester). Cette interaction s'effectue à travers les trois opérations suivantes:

- La publication de la description de services,
- La recherche et la découverte de la description du service,
- L'invocation du service basée sur la description.

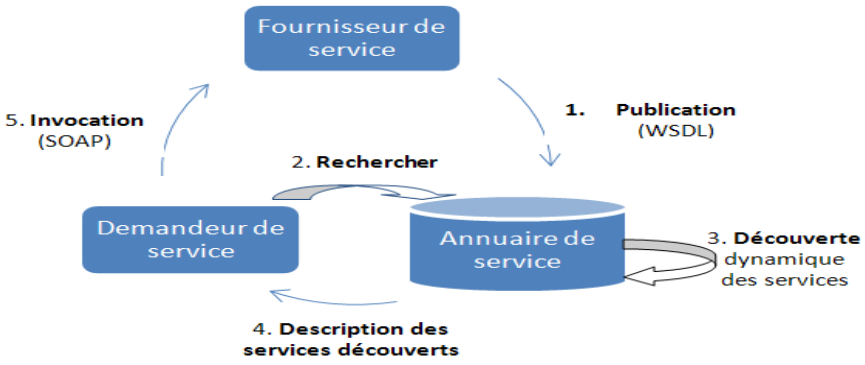


Figure 1 : Fonctionnement de services Web

Dans un scenario classique, la mise en place et l'invocation d'un service Web se déroulent de la manière suivante:

- *Le fournisseur de services* publie et déploie la description du service Web dans un *Annuaire de services* afin qu'il puisse être localisé par le *demandeur de services*.
- *Le demandeur de services* effectue des recherches dans l'*Annuaire de services* pour trouver la description de service qui répond à ses besoins.
- *Le demandeur de services* utilise la description de services découverts pour établir une connexion avec le *fournisseur de services* et interagir avec lui.

L'architecture de référence a été étendue par le W3C pour prendre en considération les protocoles de composition et pour intégrer les nouveaux standards émergeants.

3.2 Architecture étendue

L'architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile de services Web.

Nous pouvons distinguer dans cette architecture trois types de couches [28] :

- **L'infrastructure de base** (*Discovery, Description, Exchange*) définit les fondements techniques présents dans l'architecture de base. Dans un souci d'interopérabilité, les différentes couches de la pile des services Web s'interfacent avec des standards SOAP[6] pour l'échange de messages, WSDL[7] pour la description de services et UDDI[8] ou EbXML[9] pour la publication (*cf chapitre1. §4*).

[6] SOAP: Simple Objet Access Protocol
[7] WSDL: Web Service Desription Language
[8] UDDI: Universal Description, Discovery and Integration
[9] EbXML: Electronic Business using eXtensible Markup Language voir (http://www.ebXML.org)

- **La couche Business Processus** (*Business Process*) permet l'intégration de services Web. Elle présente un processus métier comme un ensemble de services Web.
- **Les couches transversales** (*Security*, *Transactions*, *Administration*, *QoS*[10]): Ces couches rendent viable l'utilisation effective des services dans le monde industriel.

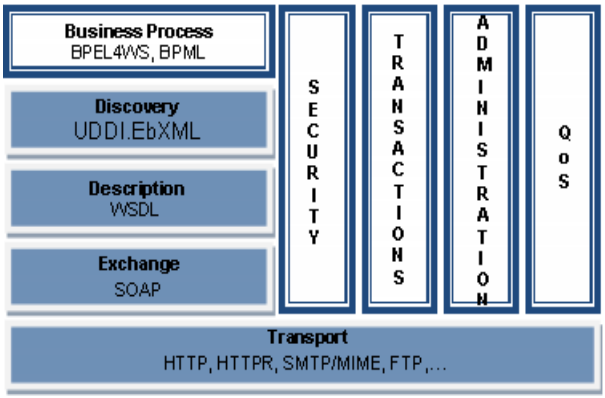


Figure 2 : Architecture en pile.

4. Infrastructure de bases des services Web

Dans l'architecture des services Web, à chaque couche du modèle correspond des technologies standards.

Les technologies de services Web sont basées sur le standard XML qui leur permet une indépendance vis-à-vis des plateformes et des langages de développement et donc leur interopérabilité. Ces technologies facilitent la description, la découverte et la communication entre services.

La définition des services Web ne présuppose ni l'utilisation du protocole SOAP pour assurer le transport et la communication entre les entités, ni une description du service Web par le langage de description WSDL. Mais l'architecture de référence d'un service Web suppose que le service possède un niveau d'abstraction dans lequel il est décrit par le langage de description WSDL et qu'il emploie le protocole de communication SOAP [22].

Les sections suivantes décrivent les technologies standards d'échange, de description, de publication et de découverte des services Web.

4.1 Echanges avec SOAP

SOAP (Simple Object Access Protocol) a été crée par Microsoft puis développé en collaboration avec IBM, Lotus et UserLand[11].

Le protocole SOAP permet l'invocation de services Web. Il s'appuie sur les protocoles Internet et assure la communication entre services par l'échange de messages au format XML. La spécification SOAP définit un modèle d'échange de messages qui repose sur trois concepts : les messages sont des documents XML, ils voyagent d'un émetteur vers un récepteur, les récepteurs peuvent former une chaîne. Un message SOAP est constitué des trois parties suivantes :

[10] Quality of Service
[11] http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

- **Enveloppe du message** (*SOAP Envelope*): cette partie est obligatoire dans tout message SOAP, elle représente la racine du document XML. L'enveloppe SOAP sert de conteneur aux autres éléments du message qui sont l'en-tête et le corps.
- **En-tête du message** (*SOAP Header*): l'entête du message est optionnelle et permet de transmettre des données supplémentaires aux services extérieurs telles que des informations sur l'authentification, la session, etc.
- **Corps du message** (*SOAP Body*): le corps du message est obligatoire et contient les informations d'appels et de réponses de procédures et l'espace de nom correspondant au nom du service ou les rapports d'erreurs.

4.2 Description de services avec WSDL 2.0

WSDL (Web Service Description Language), résulte d'un effort commun entre Microsoft, IBM et Ariba. Il décrit les composants des services Web nécessaires au protocole SOAP et à l'interaction avec les autres services.

Les versions de WSDL ont évolué. WSDL 1.x est basée exclusivement sur RPC, interdisant complètement l'utilisation de REST[12][42]. Le W3C a recommandé une nouvelle version : WSDL2.0 [62] (juin 2007). Cette version est basée sur WSDL 1.1[13]avec un modèle simplifié et clarifié, une plus grande extensibilité, un héritage d'interface, une description complète de l'utilisation de HTTP/1.1 et une description de service utilisant SOAP 1.2 [59]. Selon Jonathan Marsh [43] cette version a simplifié la tâche des développeurs qui n'ont pas toujours besoin de couches supplémentaires comme WS-Addressing[14], WS-Security[15] ou WS-ReliableMessaging. Elle permet ainsi d'utiliser très simplement REST comme architecture.

La figure 3 décrit dans un schéma XML la structure WSDL2.0.

Conceptuellement, WSDL 2.0 comprend les composants suivants: *typeDefinitions , elementDeclarations, interfaces, bindings* et *services.*

- ***typeDefinitions*** : Un *typeDefinition* fournit les définitions de types de données utilisés pour décrire les messages échangés.
- ***elementDeclarations*** : Un *elementDeclaration* inclut toutes les déclarations d'éléments définis par le schéma XML et définis dans *include* et *import.*
- ***interfaces*** : Représente un ensemble de composants Interface. Une *Interface* regroupe une ou plusieurs opérations. Elle représente *Portypes* dans WSDL1.0 et comprend les attributs : *Name* (qui est unique et obligatoire), *Extend, StyleDefault, operation* et *fault* qui sont optionnels.
    - *Extend* représente un ensemble de *elementDeclaration* qui étend l'interface.
    - *StyleDefault* indique le style par défaut utilisé pour construire la propriété *message* de *Interface message référence* dans l'ensemble des opérations.
    - *Operation* décrit les opérations du service. Elle comprend les propriétés*: Name,* et *Pattern* qui sont obligatoires et *Style, Interface message references, Interface fault References*, qui sont o*ptionnels.*

        - *Name* : identificateur unique de l'opération
        - *Pattern :* identifie le modèle d'échange du message utilisé dans l'opération. Ce modèle peut être de type *In-Only, In-Out, ...*

[12] Representational State Transfer
[13] http://www.w3.org/TR/wsdl
[14] http:// ww.w3.org/2006/04/wsaddressing-pressrelease.html.fr
[15] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

- *Style :* Ensemble d'éléments identifiant les règles qui sont utilisées pour construire les propriétés *elementDeclaration* de *Interface message reference*. Il permet de déterminer si l'opération utilise le mode RPC.
- *Interface message references :* Décrit la direction et le format du message dans une opération. Il comprend un ou plusieurs éléments d'information de type *input, output, infault* et *outfault.*
- *Interface fault References :* Décrit les erreurs envoyées ou reçues par une opération.

▪ *fault :* Décrit les erreurs qui peuvent se produire par un service.

- ***bindings:*** Représente un ensemble de composants *binding*. Cet élément spécifie les propriétés du protocole utilisé et le format de liaison pour une ou plusieurs interfaces.
- ***services*** : Représente un ensemble de composants *Service.* Un service regroupe des *Endpoints* qui implémentent une interface commune. Il comprend les attributs *name, interface* et *Endpoint* qui sont obligatoires. *Endpoint* ou *Ports* dans la version WSDL1.0 peut être utilisé pour l'accès au service en utilisant certains *Bindings.*

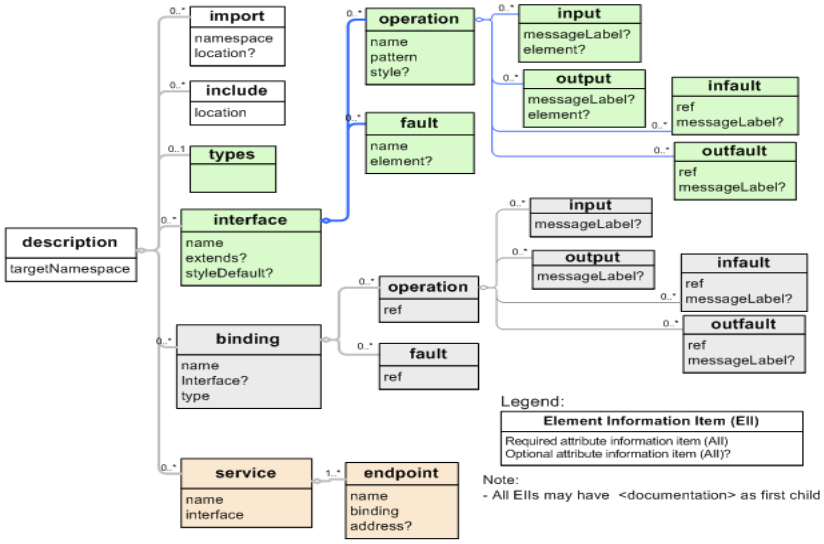


Figure 3 : Structure de WSDL2.0[16]

WSDL décrit un service Web en deux étapes fondamentales : une abstraite et une concrète. Les éléments *elementDeclaration, typeDefinition,* et *Interface* décrivent la partie abstraite d'un service tandis que *Bindings* et *services* décrivent la partie concrète du service.

La prise en charge de la spécification WSDL 2.0 est intégrée à d'importantes nouvelles recommandations de services Web telles que *Semantic Annotations for WSDL*[17] (SAWSDL) et *WS-Policy* 1.5[18].

4.3 Découverte et publication avec UDDI 3.0

La découverte des services Web peut s'effectuer sur des registres UDDI[19] ou ebRIm[20]. Ce dernier est l'annuaire utilisé dans ebXML qui est un standard du B2B[21]. Nous nous concentrons dans notre travail sur le registre UDDI.

[16] http://www.w3.org/TR/2007/PR-wsdl20-primer-20070523/images/WSDL20InfosetModel.png
[17] http://www.w3.org/TR/sawsdl/.
[18] http://www.w3.org/TR/2007/REC-ws-policy-20070904/

UDDI (Universal Description Discovery and Integration) est un standard issu de la collaboration des plus grands noms de l'industrie informatique tels que Microsoft, IBM, Sun, Oracle, SAP, HP, Intel, etc. Le cœur du projet UDDI est l'annuaire contenant des données techniques et administratives sur les entreprises et les services qu'elles publient. Cet annuaire peut être considéré lui-même comme un service car on y accède en SOAP.

La version d'UDDI a évolué Dans la version 2 ont été incluses les notions de lien entre entreprises, de taxonomie, d'internationalisation, de recherche évoluée et de réplication souple entre nœuds UDDI. La version 3, adoptée par OASIS en 2003, permet de construire des annuaires de services publiques et privés. Il apporte de nombreuses améliorations et de nouvelles fonctionnalités telles que le support de la signature électronique, le support des environnements à répertoires multiples, un support WSDL amélioré, une nouvelle interface d'abonnement et des améliorations dans la gestion du répertoire [47].

L'architecture d'UDDI3.0 se compose des structures de données suivantes [47] (la figure 4 donne un aperçu du model de données en UML de l'UDDI1.0).

- ***BusinessEntity :*** Contient des informations sur l'entreprise ou les fournisseurs et des informations sur les services qu'ils offrent. Cette structure correspond aux pages blanches dans UDDI1.0. Elle est associée à un ou plusieurs *BusinessService.*
- ***BusinessService*** : Permet de distinguer les services en identifiant leurs noms, leur description, leur code, etc. Les informations contenues dans cet élément correspondent aux pages jaunes. Il est associé à un ou plusieurs *BindingTemplate.*

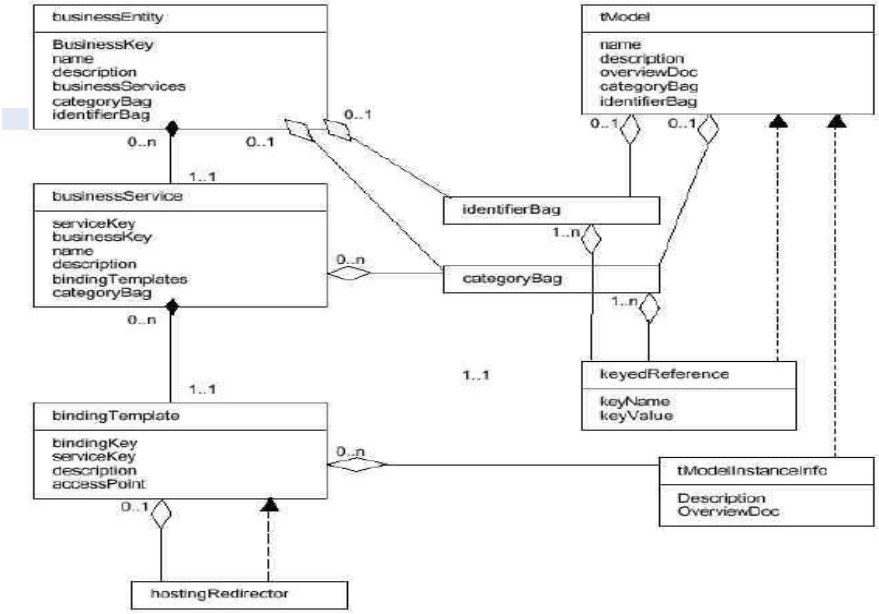


Figure 4 : Le modèle de données de l'UDDI 1.0

- ***BindingTemplate*** : Contient des informations techniques permettant aux clients de se connecter et d'invoquer le service Web. Il correspond aux pages vertes.

[19] http://www.uddi.org
[20] http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebRIM.pdf
[21] Business to Business: Echange inter- entreprise

- ***Technical Models* (*tModel)*** : Contient des informations sur les normes respectées par le service Web pour en avoir une description plus précise. Il permet d'associer un service à sa description WSDL. A la version 3.0, a été inclus l'élément *keyedReferenceGroupe* qui offre la possibilité d'utiliser une catégorisation complexe dans les entités et d'étendre les systèmes de catégorisation par dérivation [67].
- ***PublisherAssertion*** : Ajoutée dans UDDI 2.0, cette structure permet d'élaborer des relations commerciales entre les partenaires quant à la qualité d'un service.
- ***Operationalinfo :*** Introduit dans la version3.0, cet élément inclut la date et l'heure de création et de modification de la structure de données, l'identifiant du nœud UDDI auquel l'opération de publication a eu lieu et l'identité de l'éditeur.

Un client UDDI est utilisé pour effectuer une requête ou pour publier un service Web, dans un serveur UDDI.

Exemples de clients UDDI : uddi4j: UDDI pour Java, UDDI.NET SDK: UDDI pour Microsoft.NET, uddi4r: UDDI pour Ruby, uddi4py: UDDI pour Python,…
Exemples de serveurs UDDI : Apache jUDDI, Novell nSure UDDI Server: un serveur UDDI open source, BEA Aqualogic Service Registry, Microsoft Enterprise UDDI Server, Systinet Registry, OracleAS Service Registry,…

5. Quelques outils et plateformes de développement des services Web

Les éditeurs de logiciels de déploiement des services Web sont nombreux. Nous ne citons que quelques produits permettant le déploiement de l'intégralité des technologies de base des services Web à savoir :

- Web Services ToolkiT d'IBM,
- Visual Studio.NET de Microsoft,
- Sun ONE developer Studio et Java WSDP (Web Service developer Pack) de Sun,
- APACHE SOAP, Axis et Extensible Interaction System d'APACHE.

6. Conclusion

Les services Web sont une technologie admettant l'interopérabilité entre applications hétérogènes en s'appuyant sur XML. Ils permettent d'envisager une autre façon de concevoir et de déployer les applications à travers le Web.

Ce chapitre s'est porté sur la description des services Web et sur les technologies de base sur lesquelles les services Web reposent à savoir SOAP, UDDI et WSDL.

Le protocole SOAP permet l'invocation de services Web. UDDI fournit la définition d'un ensemble de services qui permettent la description et la découverte. Cependant, il ne propose aucune description du service sur la base de ce qu'il offre. Ses capacités de recherche sont limitées et reposent sur une recherche par mots clés sur la base du nom de service, de sa localisation, de son entreprise, de ses liens ou par *tModels*. Il ne supporte aucune inférence basée sur la taxonomie référencée par le *tModels*.

WSDL spécifie une méthode standardisée pour décrire les interfaces d'un service Web à un niveau syntaxique. Il ne prévoit pas explicitement de mécanismes pour spécifier la sémantique d'un service Web.

Le prochain chapitre va donc porter sur l'apport attendu des annotations sémantiques pour la description sémantique des services Web.

Chapitre 2 : Annotation sémantique des services Web

1. Introduction

Le Web sémantique suivant la vision de Tim Berners-Lee [39], vise à rendre le contenu sémantique des ressources Web interprétables non seulement par l'homme mais aussi par des programmes, pour une meilleure coopération entre humains et machines.

Comme le souligne le W3C, le but commun des services Web et du Web sémantique est de rendre l'information sur le Web plus accessible aux machines.

Tout comme l'annotation sémantique des données est la première étape critique pour améliorer la recherche, l'annotation des services Web est également la première étape pour atteindre les objectifs attendus [20].

Cette partie aborde l'annotation sémantique de façon générale (définition, classification et structure) et a pour objectif de décrire le langage de description sémantique SAWSDL, puis de présenter un état comparatif de quelques approches de description sémantique de service Web et des exemples d'outils d'annotation sémantique.

2. Définitions

Nous retrouvons dans la littérature plusieurs définitions de l'annotation.

Selon E. Desmontlis [1] une annotation est une information graphique ou textuelle attachée à un document et le plus souvent placée dans ce document. Cette place est donnée par une ancre.

L'annotation est une des formes les plus communes de méta données dans le contexte du Web [12].

Selon W3C une annotation désigne toute explication, scolie ou critique associée à une page Web.

Une annotation peut être soit privée, c'est-à-dire que le rédacteur la destine à lui-même, soit publique, c'est-à-dire destinée à un groupe de travail, institution, etc.

Les annotations peuvent être destinées à être traitées uniquement par l'humain, uniquement par l'agent logiciel ou par les deux. Les annotations destinées à être traitées uniquement par l'humain doivent avoir une forme visuelle visible, perceptible et distinguable du document qui la porte. Elles nécessitent un effort cognitif[22] et intellectuel de la part de l'humain pour l'interpréter. Les annotations destinées à être traitées par les agents logiciels décrivent l'annotation avec un langage formel.

3. Classification de l'utilisation des annotations

Bechhofer et al. [3] présentent une classification d'utilisation des annotations qui peut être considérée comme une sémantique des relations entre annotations. Ils classifient les annotations comme suit :

[22] L'annotation est interprétable par l'utilisateur

- Décoration : Les annotations sont vues comme des commentaires sur des ressources. (*cf. Annotea dans l'Annexe A*).
- Lien : Les annotations sont simplement le mécanisme qui fournit les ancres des liens. *(cf. COHSE dans l'Annexe A)*
- Identification de l'instance: Ce type d'annotation apporte une importante assertion sur une ressource qui est instance d'une classe particulière. Il existe un objet dans le monde qui est une l'instance d'un concept 'C' et qui est identifié par une URI donnée. (*cf. OntoMa dans l'Annexe A*).
- Référence de l'instance: Ce type d'annotation nécessite des pré requis et une connaissance du monde. Il existe un objet dans le monde qui est une instance d'un concept 'C' et qui est référencé par une URI donnée. *(cf. CREAM dans l'Annexe A)*
- A propos : Donne une idée assez vague de l'annotation. Il n'y a aucune assertion de l'existence d'une instance d'un concept 'C'. Le fragment de la ressource est « à propos » du concept 'C'.
- Pertinence : Retourne une faible extension ontologique. Il permet de mettre des assertions sur les classes et concepts dans l'ontologie sans explicitement incorporer cette information au sein de l'ontologie. Gruber [14] définit l'ontologie comme une spécification explicite d'une conceptualisation. L'annotation de telles ressources peut être considérée dans la classe Pertinence.

Deux types de systèmes d'annotation préexistent, l'une repose sur des annotations sémantiques et l'autre sur des annotations libres [9].

- Les annotations libres permettent d'associer des notes de lecture aux documents, de partager de l'information, d'effectuer des tâches rédactionnelles en groupe, etc. Grâce aux systèmes d'annotation, le lecteur devient aussi rédacteur.
- Les annotations sémantiques sont des méta données basées sur des ontologies et ajoutées au document [1].

Nous nous concentrons dans ce travail sur les annotations sémantiques.

4. Structure des annotations sémantiques

En bref, l'annotation sémantique revient à assigner à des entités dans le texte un lien vers leur description sémantique [4].

Les annotations sémantiques sont par nature formelles et permanentes [23][63]. Elles sont implicites[24], étant donné qu'elles font référence à une ontologie.

Elles sont utilisées dans le cadre de la recherche d'information, la classification, le résumé automatique et plus généralement pour favoriser l'interopérabilité [4]. Leur objectif majeur est de désambiguïser le document pour un traitement automatique [1].

Les annotations sémantiques sont applicables pour tout type de texte : pages Web, document régulier (non-Web), fichiers textes, bases de données, etc.

Annoter des ressources avec des méta données sémantiques fournit une certaine indication sur le contenu des ressources. Cette indication doit être accessible non

[23] Durée de vie de l'annotation, dimension temporelle proposée par C.Marshall, qui peut être permanente ou éphémère.
[24] Annotation explicite se suffit à elle-même (destinée à une autre personne que le rédacteur ne connaît pas forcément). Une annotation implicite demande une connaissance complémentaire et est destinée à un lecteur instruit des conventions adoptées (souvent le rédacteur lui même).

seulement au lecteur humain mais aussi aux agents logiciels. Pour cela, nous avons besoin de langages qui supporte la représentation des méta données sémantiques.

4.1 Représentation et modèle d'annotations sémantiques

Selon [4], les pré requis pour la représentation des annotations sémantiques sont: une ontologie et des langages qui supportent la représentation sémantique des méta données, des identificateurs d'entités, et une base de connaissance.

- Une ontologie qui définit les classes d'entités et qui offre la possibilité de se référer à ces classes. Elle représente le modèle sémantique de référence.

  Pour que les ressources du Web puissent être interprétées par des agents logiciels, il faut avoir des services capables d'exploiter et de traiter ces ressources et leurs annotations. Cela nécessite de disposer d'un langage qui permet d'exprimer les données et méta données, les ontologies et de décrire les services [12]. Le Web Sémantique propose des langages d'assertion[25] (RDF[26] et Topicmaps[27]), des langages de définition d'ontologies (DAML+OIL[28], OWL[29],..) et des langages de description et de composition des services (WSDL, UDDI, BPEL[30]...).

- Des identificateurs d'entités qui permettent de les distinguer et de les lier à leur description sémantique. Les annotations peuvent être directement insérées dans le code source du document, mais ceci a un impact négatif sur le volume du contenu et peut compliquer sa maintenance [4].
  Firyakov et al [4] proposent de découpler la représentation et la gestion de documents, des annotations et la connaissance formelle (ontologie et instance de données).
  Dans ce sens, différentes parties peuvent développer et maintenir séparément le contenu, les annotations, et la connaissance.
- Une base de connaissance avec les descriptions des entités. Lorsque les types d'entités, les relations et les entités sont codées dans l'ontologie, l'autre aspect de la représentation de l'annotation sémantique est la description de ces entités. Il doit être possible en général de décrire et d'interconnecter des entités. Le coeur de la connaissance formelle des entités est appelé « base de connaissance ».

4.2 Processus d'annotation sémantique automatique

Derrière le processus général d'annotation sémantique de documents par des ontologies se cachent trois phases [12] :

- **Repérer**: processus manuel qui consiste à placer dans le document des références aux concepts de l'ontologie qu'il contient.
- **Instancier**: processus manuel ou automatique permettant d'instancier les attributs des concepts à l'aide des informations présentes dans le document.
- **Enrichir**: processus manuel visant à ajouter des informations par l'intermédiaire des attributs de concepts qui n'ont pas pu être instanciés lors de la phase précédente.

[25] Les assertions affirment l'existence de relation entre les objets.
[26] Ressource Description Framework (http://www.w3.org/RDF/)
[27] Standard ISO issu de HyTime dont le but était d'annoter les documents multimédia
[28] DARPA Agent Markup Language et Ontology Inference Layer
[29] Ontology Web Language : standard recommandé par le W3C pour la description d'ontologie
[30] Business Process Execution Language , appelé aussi BPEL4WS

Dans le cadre d'une annotation sémantique automatique, le processus d'annotation automatique comprend l'extraction, l'indexation et la recherche [8].

5. SAWSDL : un modèle d'annotation de services Web

SAWSDL (Semantic Annotations for Web Services Description Language) est une extension des langages de définition WSDL et XML schema. Devenu depuis Août 2007 une recommandation du W3C pour la description sémantique des services Web. Il est basé sur les membres ayant soumis WSDL-S [27]. Il emploie les mécanismes d'extension WSDL2.0 et supporte WSDL1.1.

SAWSDL ne désigne pas un langage particulier pour représenter les modèles sémantiques. Le mécanisme sémantique d'annotation exige que les concepts sémantiques définis dans le document WSDL soient identifiables par l'intermédiaire des références d'URI. L'URIs se réfère typiquement à des concepts dans un modèle sémantique qui est externe au document WSDL. Cependant, l'URIs peut également se rapporter à des éléments dans le document WSDL si l'information sémantique est incluse dans le document par l'intermédiaire des éléments étendus du WSDL.

5.1 Structure des d'annotation

Les spécifications SAWSDL se concentrent sur l'annotation sémantique de la définition abstraite d'un service pour permettre la découverte, la composition et l'invocation dynamique des services. Ces spécifications ne concernent pas l'annotation de l'implémentation du service.

SAWSDL définit les attributs étendus *modelReference*, *liftingSchemaMapping* et *loweringSchemaMapping.*

- *modelReference* **:** représente l'association entre un composant WSDL ou schéma XML et un concept dans un certain modèle sémantique. Il peut être employé avec chaque élément dans WSDL et schéma XML.
- L'attribut étendu *liftingSchemaMapping,* peut être utilisé pour spécifier le mapping entre le type définition et les données sémantiques,
- L'attribut *loweringSchemaMapping* peut être utilisé pour spécifier le mapping entre les données sémantiques et type de définition.

Les attributs *liftingSchemaMapping* et *loweringSchemaMapping* permettent d'indiquer des correspondances entre les données sémantiques définies par un modèle sémantique et XML. De telles correspondances sont utiles en général quand la structure de l'instance de données ne correspond pas trivialement à l'organisation sémantique des données. Ils sont employés pour supporter l'invocation d'un service Web et pour la médiation entre données XML à travers une ontologie [5].

5.2 Annotation de documents WSDL2.0

En termes de modèle de composant WSDL 2.0, *modelreference* est une nouvelle propriété qui peut être utilisée sur un élément *wsdl:interface, wsdl:operation, xs:faults* etc.

5.2.1 Annotation des interfaces avec Model Reference

Un *modelReference* sur un élément d'interface WSDL fournit une référence à un concept ou à des concepts dans un modèle sémantique qui décrit l'interface. L'exemple ci-dessous [34] illustre une annotation de catégorisation sur un élément Interface.

```
<wsdl:interface name="Order"
sawsdl:modelReference="http://example.org/categorization/products/electronics">
… </wsdl:interface>
```

Dans cet exemple, le *modelReference* « pointe vers » un concept de la catégorie « électronique » dans un certain modèle sémantique.

5.2.2 Annotation des opérations avec ModelReference

L'annotation de l'élément *operation* ou *fault* porte une référence à un concept dans un modèle sémantique qui fournit une description de niveau élevé de l'opération, et indique ses aspects comportementaux ou inclut d'autres définitions sémantiques.

L'annotation de l'opération, employant l'attribut *modelReference,* est présentée dans l'exemple ci-dessous [34] :

```
<wsdl:operation name="order" pattern="http://www.w3.org/ns/wsdl/in-out"
sawsdl:modelReference=http://www.w3.org/sawsdl/purchasorder#RequestPurchasOrder>
…</wsdl:operation>
```

5.2.3 Annotation des erreurs avec ModelReference

L'annotation de *fault* ne décrit pas l'erreur du message, qui doit être annoté dans le schéma XML. L'exemple suivant illustre comment utiliser l'attribut *modelReference* pour annoter l'élément *fault.*

```
<wsdl:interface name="Order">
<wsdl:fault name="ItemUnavailableFault"…
sawsdl:modelReference="http://www.w3.org/sawsdl/purchaseorder#ItemUnavailable"/>…
</wsdl:interface>
```

Cet exemple identifie le concept "ItemUnavailable" dans le model sémantique de référence comme une description de l'erreur "itemUnavailableFault".

5.3 Le support du WSDL 1.1

Le mécanisme pour l'annotation sémantique décrite dans ces spécifications peut également être appliqué à WSDL 1.1.

- Annotation des opérations : WSDL1.1 ne permettant pas l'extension d'attributs au niveau de l'élément, la spécification SAWSDL a introduit une extension d'un nouvel élément *attrExtensions* pour supporter l'annotation sémantique des opérations WSDL 1.1. Ce nouvel élément sera considéré comme enfant de l'élément opération.
- Annotation des *portTypes* : Le *portType* correspond à *Interface* dans WSDL2.0. Il est donc annoté de la même façon.
- Annotation des entrées et sorties : Les attributs *liftingSchemaMapping*, *loweringSchemaMapping* ou *modelReference* peuvent être ajoutés à un type d'élément pour spécifier l'annotation des entrées ou sorties dans un type message. Les types de message sont référencés dans la structure *portType* du WSDL 1.1.
- Annotation des erreurs : Dans WSDL 1.1, les erreurs sont spécifiées comme des messages dans les opérations. L'annotation de l'erreur nécessite d'être définie au niveau de chaque opération où *fault* apparait.

6. Etude comparative de quelques approches de description sémantique des services Web

Plusieurs travaux pour la description sémantique des services Web ont été effectués [16, 17, 29, 31,45, 68], certains ont été soumis au W3C tels que : OWL-S [45], WSMO [31], WSDL-S [16, 17] et SAWSDL. Cette section présente une synthèse de l'étude de

ces approches de description sémantique qui sont présentées en Annexe D. Ces approches sont étudiées suivant les critères suivants :

- *Type de ressources décrites sémantiquement* : précise les ressources décrites sémantiquement (schema XML, WSDL, UDDI,...).
- *Propriétés* : précise ce qui est décrit sémantiquement dans le document. Il peut s'agir de propriétés fonctionnelles (*Input, Output, pre-condition*…), non fonctionnelles (implémentation) ou comportementales.
- *Langage de représentation* : décrit le langage de représentation du model sémantique (OWL, WSML[31],…)
- *Stockage des annotations :* spécifie si les annotations sont stockées dans le document (interne) ou séparées du document (externe).
- *Type de modèle :* spécifie si le modèle du domaine sémantique est interne ou externe ;
- *Technique de matching*[32] *:* indique l'approche de mise en correspondance entre la requête et les propriétés.

Approches	OWL-S	WSMO - DERI	WSDL-S METEOR-S	Spécification SAWSDL
Type de ressources décrites sémantiquement	WSDL1.0, WSDL1.1		WSDL1.2, UDDI3.0	WSDL1.1 WSDL2.0 XML schema
Propriétés	Fonctionnelles et non fonctionnelles	Fonctionnelles, non fonctionnelles et comportementales	Fonctionnelles et non fonctionnelles	Fonctionnelle (Interface, operation, faults, type description)
Langage de représentation	OWL	WSML	OWL	Indépendant d'un model particulier (OWL, RDF,..)
Stockage des annotations	/		Interne	Externe (Référence URI)
Type de modèle	Interne	Externe		Externe ou interne
Technique de matching	Algorithme matchmaker [46]	Matching de la description abstraite des buts du client	Plusieurs algorithmes de matching	/
Autres perspectives d'utilisation des Annotations	Découverte et Composition			BPEL, WS-policy Découverte et invocation

Tableau 1 : Synthèse des approches de description sémantique des services Web

Comme illustré dans le tableau ci-dessus, OWL-S, WSMO et WSDL-S offrent la possibilité de décrire sémantiquement les propriétés fonctionnelles et non fonctionnelles selon un modèle sémantique particulier.

OWL-S emploie l'algorithme Matchmaker [46], pour rechercher les descriptions de services Web qui ont une correspondance sémantique entre les paramètres d'une requête et le service Web. WSMO emploie le matching de la description abstraite des buts du

[31] Web Service Modeling Language (http://www.wsmo.org/wsml/)
[32] Un match est l'opération qui consiste à prendre en entrée deux schémas, chacun composé d'un ensemble d'entités, et renvoie en sortie les mappings (mises en correspondance) entre entités.

client. WSDL-S par son approche d'annotation du WSDL1.0 utilise plusieurs algorithmes de matching. Le stockage des annotations s'effectue de façon interne.

SAWSDL permet d'annoter les schémas XML, WSDL2.0 et supporte WSDL 1.0. Cependant, il se limite uniquement à une description sémantique des propriétés fonctionnelles. Il offre la possibilité de stocker les annotations de façon externe ou à l'intérieur même du document. Ce langage qui est une recommandation du W3C pour la description sémantique des services Web a la particularité de ne pas dépendre d'un langage de représentation du modèle sémantique particulier.

En gardant le mécanisme sémantique séparé de la représentation des descriptions sémantiques, ceci offrira la flexibilité à la communauté de réalisateurs de choisir leur langage de représentation sémantique préféré [27].

7. Exemples d'outils d'annotations sémantiques

Les outils d'annotations sémantiques visent à améliorer l'appréhension des documents ainsi que la communication et l'interopérabilité sur le Web.

Une étude comparative des outils d'annotations sémantique est présentée en *Annexe A*. Cette étude (voir le tableau de synthèse Table 2 et voir l'annexe A pour plus de détails) montre que les outils d'annotations sémantiques ont été le plus étudiés dans le cadre du Web sémantique et n'ont pas inclus le cadre des services Web. Elle nous a permis de conclure que les systèmes d'annotation sémantique varient dans leur architecture, dans les outils d'extraction de l'information et dans les méthodes et les langages d'annotations. Ils dépendent aussi du cadre de leur utilisation (destiné à une collaboration, une recherche, intégration,…). Ces outils ne regroupent pas toutes les performances et les propriétés attendues pour un système d'annotations sur le Web [14] et qui sont : la simplicité, l'efficacité, la transparence, l'indépendance vis à vis de la plate-forme support, le passage à l'échelle et le support du travail collaboratif.

Pour l'annotation des services Web, de nombreux plug-ins ont été développés. Ces derniers fournissent une interface utilisateur pour l'annotation des documents WSDL à partir d'une ontologie décrite en OWL.

La liste des caractéristiques, prise en charge par chacun, des plug-ins suivants est décrite dans un appel à candidature pour une recommandation par le W3C [34].

- Woden4SAWSDL [64] et Radiant[33] du Laboratoire Lumina de l'université de Georgia.
- WSMO4J [65] Grounding, WSMO Studio[34], WSMO Grounding et WSMO-Lite de DRI.
- SAWSDL4J [66] n'appartenant pas à une organisation particulière.
- OWL-S de SAWSDL perspective.
- Semantic Tools for Web Services d'IBM Alpha Works[35]
- RDF Mapping implementation[36].

[33] http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1
[34] http://www.wsmostudio.org/
[35] http://www.alphaworks.ibm.com/tech/wssem
[36] http://lists.w3.org/Archives/Public/public-ws-semann/2007Jun/0004

Outils d'annotations	Interprétation	Type d'annotation	Type de ressources annotées et localisation	Langage	Stockage des annotations	Langage d'ontologie	Technologies	Extraction d'information et apprentissage
Annotea	Humain	Manuelle	Fragment de page HTML et XML (svg, XHML)	RDF	Séparer dans un ou plusieurs serveurs	RDF	Serveurs HTTP Dédié; Interface d'annotation	n/a
Yawas			HTML, TEXT, XML	Texte	Insérer dans le document	Base de données	Implanté en Java et java script utilise DOM niveau2	
SHOE Knowledge Annotator	Agent logiciel		Page HTML	HTML		SHOE	Application java autonome : Interface graphique d'annotation et de construction de requête et un robot pour récupérer les assertions	
Aero-Daml		Automatique	TEXT	DAML+OIL	Séparer dans un serveur	DAML+OIL	Application autonome	Extraction par AeroText et apprentissage
SMORE	Humain et Agent logiciel	Manuelle	Fragment de pages HTML, courriers, image.	RDF et OWL		RDF et OWL	serveur dédié	n/a
KIM		Automatique	Document structuré et non structuré	RDF ET OWL LITE		Propriétaires : PROTON, KIMLO et KIMSO	Utilise les technologies GATE, Sesame, and Lucene. La plateforme comprend un serveur, une interface utilisateur Web, Internet Explorer plug-in	KIM KB + LUCENE
COHSE		Manuelle et automatique	Fragment de pages HTML	RDF, DAML+OIL. Génère des hyperliens		DAML+OIL	Architecture complexe [14] : outil pour annotation; serveurs d'annotation dédiés	Agent COHSE DLRS extraction et apprentissage
MnM		Manuelle, Semi-automatique et automatique		RDF		KMI et ontologie OCML et DAML+OIL ou schéma RDF	Architecture complexe [14]: navigateur Web, outil de navigation d'ontologie et de création d'instance dans une base de connaissance.	Moteur d'extraction Amiclare (induction) et peuplement d'ontologie
OntoMat Annotizer	Agent logiciel	Manuelle et semi-automatique	Fragment de pages HTML	RDF	Insérer dans le document ou séparer dans un serveur	OWL	Application java autonome : outil de création de page Web et annotation de robot	Moteur d'extraction Amiclare
METEOR-S			Fichier XML (WSDL et UDDI)	JAVA	Insérer dans un fichier texte	DAML, RDF-S et OWL		Plusieurs Algorithmes de mise en correspondance

Tableau 2 : Synthèse des outils d'annotation sémantique

8. Conclusion

Dans ce chapitre, après avoir passé en revue les définitions des annotations, leur classification et leurs structures, nous nous sommes focalisés sur l'annotation sémantique du langage de description des services SAWSDL. Cette nouvelle approche permet de garder le mécanisme sémantique séparé de la représentation des descriptions sémantiques. Elle ne dépend donc pas d'un langage de représentation sémantique particulier.

Aussi, des critères spécifiques ont été définis pour présenter un état comparatif entre quelques approches de description sémantique de service Web et un état comparatif sur quelques outils d'annotation sémantique.

Une fois un service Web décrit et publié, il doit pouvoir être retrouvé par le demandeur de service. Ceci fait l'objet du prochain chapitre.

Chapitre 3 : Exploration des services Web

1. Introduction

Après une étude des approches de description sémantique des données. Nous nous intéressons dans ce chapitre à trouver un moyen pour leur interrogation. Nous commencerons par une présentation générale du langage d'interrogation des données semi structurées XQUERY, puis nous détaillerons l'exploration des services Web.

2. Interrogation des données semi-structurées

Les données non structurées ont des structures irrégulières mais disposent d'une structure minimale. Elles sont sans schéma fixe. Elles sont auto descriptives (données et méta données) et il n'existe pas une séparation entre le schéma et les données.

Plusieurs syntaxes et modèles ont été proposées pour représenter des données semi-structurées (OEM [57], ACeDB [56] et XML). Le langage XML est un format textuel qui permet de créer des documents contenant des données semi-structurées. Il représente la technologie de base des services Web.

Evaluer une requête sur les données semi-structurées suppose de naviguer à travers la structure en examinant à la fois les valeurs des éléments et le nom auto-descriptif de l'élément tout au long du parcours.

Plusieurs langages de requêtes sur les données semi-structurées ont été proposés au W3C tel que : XML-QL[37], XQL[38], XPath[39], XQUERY [61].

XQUERY est un langage de requête proposé par IBM et les auteurs de XML-QL et soutenu par Microsoft. Il est depuis janvier 2007 une recommandation du W3C. Il est à base d'expression de chemin, de boucles de répétition, de tests de prédicats et d'éléments de reconstruction de document XML. Il est issu du langage Quilt[40] qui inclut XPath 1.0, XQL, XML-QL, SQL[41], et OQL[42].

XQUERY permet la manipulation et la génération de documents XML à partir des sources XML, de bases de données relationnelles ou objets. Il est un sur-ensemble de SQL : les opérations sur les relations sont étendues aux forêts qui sont sélectionnées par une expression XPath.

Une requête exprimée en XQUERY retourne une collection de documents XML.

3. Interrogation et exploration des services web

Nous présentons dans cette section les langages d'interrogation des services Web rencontrés dans la littérature à savoir XSRL, LARKS et SwellQL et quelques approches de découverte des services Web.

[37] http://www.w3.org/TR/1998/NOTE-xml-ql-19980819
[38] http://www.w3.org/Style/XSL/Group/1998/09/XQL-proposal.html
[39] http://www.w3.org/TR/xpath
[40] http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html
[41] Structured Query Language
[42] Object Query Language

3.1 Langages d'interrogation des services Web

3.1.1 XSRL

XSRL (XML Service Request Language) [51] est un langage d'interrogation des services Web basé sur le langage d'interrogation XQUERY et les techniques de planification en intelligence artificielle. Ce langage contient un ensemble de constructeurs appropriés pour l'expression des requêtes et des contraintes, mais aussi pour l'expression des opérateurs d'ordonnancement.

XSRL permet d'exprimer la succession des activités pour satisfaire la requête et l'information sur les paramètres des actions planifiées. Dans l'expression d'une requête XSRL il est important que l'utilisateur soit capable de spécifier le chemin par lequel la requête doit être planifiée et exécutée. Le document réponse peut être vu comme une série de plans qui satisfait la requête potentiellement.

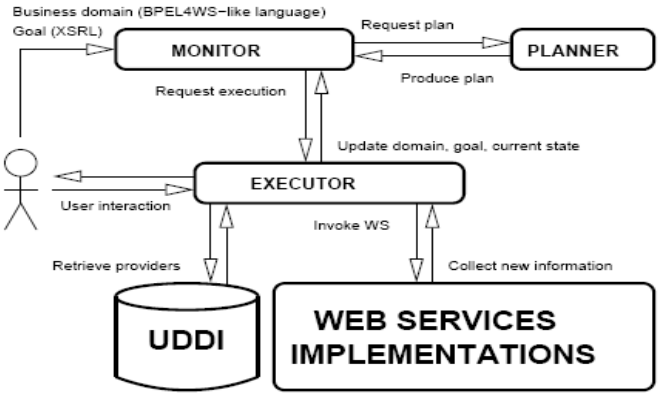


Figure 5 : Architecture de haut niveau de XSRL [52]

Comme illustré dans la figure 5, une requête exprimée en XSRL est une entrée à un planificateur automatique. Après réception de la requête, le planificateur génère un ordre pour l'interaction avec les services existants dans l'UDDI. Le langage de requêtes permet au planificateur de sélectionner les actions, initiales et finales, appropriées pour l'orchestration des services de la requête et de les planifier.

```
<XSRL>
   <REQUEST> {
      FOR $a in document(PkgTravelSegment.xml)//AirSegment
          [CarrierName =  "Alitlaia"| "United Airlines" AND
           DepartureAirport = "NewYork" AND
           ArrivalAirport = "Rome" | "Venice" AND
           (Price <= 800 AND Price >=500) AND
           SeatQty = 3 AND
           ArrivalDate = "1 June, 2002" AND
           DepartureDate = "10 June, 2002"]
      RETURN
          <ArrivalAirport>{ $a/ArrivalAirport}</ArrivalAirport>
          <price>{ $a/price}</price>
          <ArrivalDate>{ $a/ArrivalDate}</ArrivalDate>
          <DepartureDate>{ $a/DepartureDate}</DepartureDate>
          <HotelList> {
             FOR $h in document (hotelReference.xml)//HotelReference
                 [ChainHotel = "Hilton"]
             WHERE ($h/Area =$a/ArrivalAirport AND
                    $h/HotelArrivalDate = $a/ArrivalDate + 1 AND
                    $h/HotelDepartureDate = $a/DepartureDate 1)
             RETURN
                <HotelName>{ $h/HotelName }</HotelName>
                <HotelAddress>{ $h/HotelAddress }</HotelAddress>
          }</HotelList>
   }</REQUEST>
   <GOAL>
      <Then><Vital>receive_confirmation($a)</Vital>
      <Optional> receive_confirmation ($h)</Optional></Then>
   </GOAL>
</XSRL>
```

Figure 6 : Extrait de code d'une requête XSRL [60]

La figure 6 illustre un extrait d'une requête exprimée en XSRL qui permet de rechercher une réservation hôtelière.

3.1.2 Approche basée sur LARKS

LARKS (Language for Advertisement and Request for Knowledge Sharing) [49] est un langage qui peut être utilisé pour la description, la publication et l'interrogation de services Web. Il se base sur le calcul de frames.

Dans LARKS, la spécification des demandes et des offres de services sont décrites par des classes d'attributs. La signification de chaque classe d'attribut est décrite dans ce qui suit :

- *Context* : Contexte de la spécification d'une demande ou d'une offre de services, il représente un mot-clé décrivant ce que fait le service.
- *Type* : Définition des types de données abstraits/variables utilisés dans la spécification.
- *Input/output :* Déclaration des variables d'entrée/sortie d'une spécification de services.
- *InConstraints/OutConstraints* : Contraintes logiques sur les entrées/sorties décrites par des clauses de Horn.
- *ConcDescripton*: Description formelle des concepts servant à l'annotation sémantique des noms et des entrées/sorties des demandes et des offres de services. Le rattachement d'un concept *C* à un mot *w* se fait sous la forme *w*C*, et qui signifie que le concept *C* est la description formelle du mot *w*.
- *TextDescription* : Description textuelle de ce que recherche un *demandeur de services* ou de ce que peut offrir un *fournisseur de services*.

Les attributs *Context, Input* et *Output* peuvent être annotés par des concepts sémantiques stockés dans l'attribut *ConcDescriptions* afin d'être interprétables par une machine. LARKS utilise des ontologies pour décrire sémantiquement les services Web. Ces ontologies peuvent être décrites dans des langages de concepts comme ITL[43], LOOM[44], CLASSIC, ou KIF[45].

LARKS réalise à la fois un match syntaxique, un match sémantique et une spécification de concepts (via ITL). Il a été étendu, par Y. Sam et al [50], pour permettre la prise en compte de l'expression des unités de mesure des entrées et sorties dans les demandes et les offres de services.

L'appariement de services s'effectue en premier à la découverte d'un ensemble de services susceptibles de répondre à la requête du client, puis à la sélection du service qui puisse effectivement y répondre. Dans le cas où ce dernier n'existe pas, un mécanisme est offert pour la transformation d'un des services existants dans l'annuaire (sélectionné pendant le premier niveau) vers le service exigé par le client de services [50].

3.1.3 SwellQL

SWELL [35] est un outil logiciel pour la recherche sémantique des services Web. Swell comprend, entre autres, une ontologie *SwellOnt* et un langage d'interrogation *SwellQL*.

SwellOnt décrit le service comme un ensemble de caractéristiques qui peuvent être soit fonctionnelles soit non fonctionnelles. Cette ontologie est basée sur OWL-S, sur

[43] ITL: Information Terminological Language
[44] http://www.isi.edu/isd/LOOM/
[45] KIF : Knowledge Interchange Format

WSMO et étend leurs concepts. Il aide à la découverte des services dont les caractéristiques ont été décrites en utilisant l'ontologie SwellOnt. Ceci revient à l'interrogation de l'ontologie SwellOnt.

D'une façon générale les requêtes dans SwellQL ressemblent à un appel de fonction où les paramètres sont des instances de classes SwellOnt. La figure 7 illustre un extrait de code du langage *SwellQL* qui recherche des services Web ayant comme entrée le concept « Airport » [35].

```
SwellQL: findServices (input <=
http://www.daml.org/services/owl-
s/1.0/Concepts.owl#Airport)
```

Figure 7 : Extrait du code d'interrogation en SwellQL

3.2 Synthèse des langages d'interrogation des services Web

Comme illustré dans le tableau 3. Le principe des langages destinés à l'interrogation des services Web varient. Ces langages sont étudiés suivant les critères suivants :

- *Principe du langage* : précise le fondement du langage,
- *Propriétés à interroger* : décrit les propriétés (fonctionnelles, non fonctionnelles,..) qui peuvent être interrogées par le langage,
- *Description sémantique du service* : Indique si le service est décrit sémantiquement ou non,
- *Description du service:* spécifie le langage du modèle sémantique,
- *Composition des services:* spécifie si le service peut être composé ou non,
- *Résultat* : Résultat de la requête.

Langages	SwellQL	Approche basée sur LARKS	XSRL
Principe du langage	Interrogation d'une ontologie	Description, publication et interrogation de services Web. LARKS se base sur le calcul de frames	XQuery & Techniques de planification en IA
Propriétés à interroger	Fonctionnelles et non fonctionnelles		élément ou propriété du document XML
Description sémantique du service	Oui		Non
Description du service	Ontologie SwellOnt et dans d'autre formalisme	Classes d'attributs utilisant des ontologies décrites en ITL, LOOM, CLASSIC, ou KIF	WSDL
Composition des services	Non	Oui	Oui (basée sur BPEL4WS)
Résultat	Concepts de SwellOnt	/	Document XML

Tableau 3 : Synthèse des langages d'interrogation des services Web

LARKS se base sur le calcul de frames et peut être utilisé non seulement pour l'interrogation de services Web mais aussi pour la description et la publication. Les langages SWellQL et LARKS permettent une description sémantique des propriétés fonctionnelles et non fonctionnelles des services Web par des langages de description d'ontologies bien spécifiques.

Le langage XSRL permet une interrogation des services Web, décrits en WSDL et composés en BPEL4WS, et ce en se basant sur le standard d'interrogation des données semi-structurées XQUERY. Cependant, il ne prend pas en considération l'aspect sémantique.

3.3 Autres approches de découverte sémantique des services Web

Plusieurs autres approches de découverte des services Web se basant sur OWL-S et WSMO ont été proposées. Ces approches ne s'appuient pas sur un langage d'interrogation des services Web. Nous présentons brièvement dans ce qui suit quelques unes de ces approches.

Ye et al. [54] ont proposé une approche de découverte des services Web qui unifie la façon de décrire sémantiquement le service Web par le demandeur et le fournisseur de service. Leur approche annote le service Web et la requête par une même ontologie et exploite le temps de publication pour le calcul des mises en correspondance. La mise en correspondance entre service et requête Web est définie dans les concepts de l'ontologie.

Luo et al. [53] ont utilisé l'objet KeyedReferenceGroup, particularité de la dernière version de l'UDDI, pour permettre la sauvegarde de la description OWL-S du service dans UDDI3.0. Contrairement à l'approche de Colgarve et al. [55], cette solution à la spécificité de ne pas effectuer des modifications sur le registre UDDI existant. Pour supporter les requêtes sur le registre UDDI, des API[46]-UDDI sont utilisées.

Aggarwal et al. [48] ont proposé une approche qui se distingue dans le type de réponse retourné. Cette dernière indique non seulement si le service répond ou non mais aussi sous quelle condition le service peut répondre à la requête utilisateur. Ils modélisent la sémantique, les contraintes de sécurité et les contraintes temporelles de façon unifiée, sur la base des logiques de description. La requête est exprimée sous forme de condition sur les sorties. Un algorithme est défini pour mettre en correspondance les concepts du demandeur et du fournisseur de services lors de la phase de découverte ; il utilise le raisonneur KAON2[47] sur des ontologies OWL-DL.

4. Conclusion

Plusieurs travaux sur la découverte sémantique des services Web ont été proposés. Nous avons présenté dans le chapitre 2 une synthèse de quelques uns de ces travaux qui se sont concentrés sur la description sémantique des services Web tels que : OWL-S, WSMO, WSDL-S et SAWSDL. Nous avons conclu que la dernière approche d'annotation sémantique SAWSDL est la seule approche qui permet de ne pas dépendre d'un langage de représentation sémantique particulier.

Par la suite, nous nous sommes intéressés dans ce chapitre à leur interrogation. Nous avons présenté le langage d'interrogation des données semi-structurées XQUERY, puis nous nous sommes focalisés sur les langages d'interrogation des services Web ou encore sur les approches d'intégration de la description sémantique à l'UDDI [16, 48, 53, 54, 55]. Toutes ces approches de découverte des services Web soit ne prennent pas en considération l'aspect sémantique tel que XSRL soit dépendent d'un langage de représentation sémantique particulier et ne permettent pas toujours une composition des services. Ainsi le prochain chapitre présente notre approche pour pallier ces limites.

[46] Application Programming Interface
[47] http://kaon2.semanticWeb.org/

Chapitre 4 : Une base de liens sémantiques pour l'interrogation et la découverte de services

1. Introduction

Notre travail consiste à annoter les services Web et à proposer un langage d'expression de requêtes et d'exploration des services pour répondre au mieux à la requête.

L'approche que nous proposons est basée sur une architecture SOA. Toutefois, elle impose que le *fournisseur de service*s **décrive sémantiquement** les fonctionnalités de son service pour pouvoir le **publier** dans *l'annuaire de services* et que le *demandeur de service* puisse **formuler sa requête** et **retrouver** une description des fonctionnalités des services Web **découverts** avec la possibilité par la suite d'invoquer les services.

Afin de décrire sémantiquement les fonctionnalités des services Web, nous avons opté pour une annotation sémantique utilisant la spécification SAWSDL. Contrairement à d'autres approches, cette spécification offre l'avantage d'annoter les fonctionnalités des services Web sans dépendre d'un langage de représentation sémantique particulier. Pour cela, nous avons étendu l'utilisation de l'attribut *model reference* du SAWSDL aux entrées et sorties des services Web.

La publication des services Web dans un registre UDDI et peu sélective et ne propose aucune description du service sur la base de ce qu'il offre, ce qui restreint considérablement la recherche. Pour pallier ces limites, nous avons opté pour un stockage secondaire des fonctionnalités des services Web, lors de la phase de publication, dans un registre que nous avons appelé *base de liens sémantiques*.
Cette base permettra non seulement de déterminer les fonctionnalités des services Web mais aussi de restreindre l'interrogation uniquement aux services susceptibles de répondre à la requête, de minimiser donc le temps de recherche et d'optimiser le traitement des requête lors de la phase de découverte, du fait que les correspondances entre les concepts de la requête et ceux des fonctionnalités des services Web sont connues à l'avance.

Dans notre approche [71], tout comme dans Ye et al. [54] et Aggarwal et al. [48] la requête est formulée de façon unifiée en se référençant à une ontologie de domaine décrite en OWL. Ce qui permet d'une part, de formuler les requêtes sur la base d'une sémantique commune, et d'autre part de traiter les requêtes et de les diriger vers les services Web pertinents.

La découverte des services Web consiste à trouver les mises en correspondance entre une requête exprimée avec des termes de l'ontologie et les fonctionnalités des services Web. En s'appuyant sur les langages d'interrogation des données semi structurées, tout comme dans le langage XSRL mais avec l'avantage d'une description sémantique des services, la requête sera traduite en une interrogation XQUERY de document XML dont les schémas XML sont prédéfinis : le schéma de la *base de liens sémantiques* et le schéma du registre UDDI.

Dans ce chapitre, nous présentons l'architecture générale et le fonctionnement du système (paragraphe 2), puis nous détaillons (paragraphe 3) l'annotation sémantique en SAWSDL, la publication des services Web dans le registre UDDI et dans la *base de liens sémantique*s, la formulation de la requête, puis la découverte des services répondant à la requête.

2. Architecture du système

L'infrastructure de base des services Web ne permet pas encore d'effectuer une gestion largement automatisée. L'aspect sémantique indispensable pour permettre une recherche efficace y est absent. Nous présentons dans ce qui suit le modèle en couche du système que nous proposons et son fonctionnement.

2.1 Le modèle en couche proposé

Comme illustré par la figure ci-dessous, l'architecture que nous proposons est basée sur l'architecture de référence des services Web. Pour prendre en considération l'aspect sémantique, nous l'avons étendue par l'ajout d'une couche verticale supplémentaire basée sur XML et appelée liens sémantiques

L'architecture ainsi proposée est composée des couches suivantes :

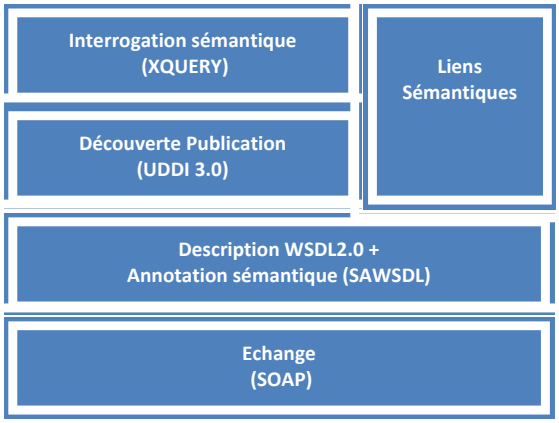


Figure 8 : Architecture générale

- **Couche Echange :** identique à la couche d'échange de l'architecture de référence des services Web. Elle utilise donc le protocole SOAP pour l'invocation de services Web.
- **Couche description sémantique :** Dans cette couche le service Web n'est pas décrit en WSDL uniquement mais s'appuie sur la spécification SAWSDL. Les services Web sont donc décrits en WSDL2.0 et annotés par un modèle sémantique de référence.
- **Couche publication** : consiste en la publication du service Web décrit en SAWSDL dans l'annuaire de service.
- **Couche liens sémantiques** : Cette couche verticale est définie en XML et exploite les autres couches horizontales. Elle décrit sémantiquement les fonctionnalités des services Web et les mises en correspondance entre les concepts du modèle sémantique de référence et les fonctionnalités des services Web. Elle permet ainsi d'optimiser le temps de recherche lors de la phase découverte.
- **Couche interrogation :** permet de retrouver les services Web répondant à la requête. La réponse à une requête, utilise le standard d'interrogation XQUERY sur le schéma de la *base de liens sémantiques* et sur le schéma *UDDI*.

Excepté la couche d'échange, les autres couches sont décrites en détail dans les autres sections.

2.2 Fonctionnement du système

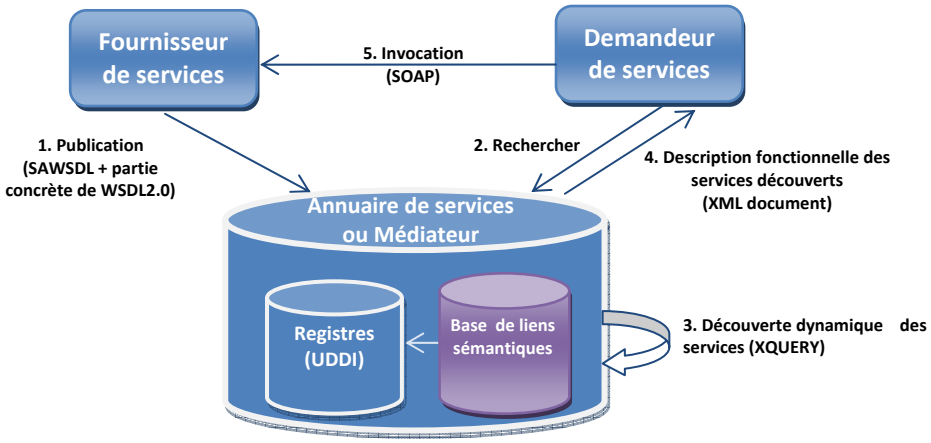


Figure 9 : Fonctionnement du système

Comme illustré dans la figure 9, l'interaction entre les différents services s'effectue à travers les opérations suivantes:

1. *Publication : Le fournisseur de services* annote le service Web par un modèle sémantique de référence selon la spécification SAWSDL et le publie dans un *Annuaire de service* afin qu'il puisse être localisé par le *demandeur de services*.
2. *Recherche : Le demandeur de services* effectue des recherches dans l'*Annuaire de service* pour retrouver les services Web qui répondent à sa requête.
3. *Découverte : L'annuaire de services ou médiateur* conserve des données techniques et administratives sur les fournisseurs de services, *les liens vers* la description du service Web, ses fonctionnalités dans des annuaires pour une découverte dynamique des services. Ces annuaires comprennent les registres UDDI et la *base de liens sémantiques.*
4. *Description fonctionnelle des services découverts :* L'interrogation de l'*annuaire de service* retourne *au demandeur de service* les services Web appropriés et leurs fonctionnalités sémantiques.
5. *L'invocation* : *Le demandeur de services* utilise la description de services découverts pour établir une connexion avec le *fournisseur de services* et interagir avec lui.

3. Description détaillée

3.1 Annotation de services avec SAWSDL

Pour décrire sémantiquement les services Web, nous étendons sa description en annotant les fonctionnalités des services par SAWSDL (*cf chapitre 2. §4.2*) qui est une recommandation du W3C pour l'annotation sémantique des services Web. SAWSDL emploie les mécanismes d'extension WSDL2.0 et ne désigne pas un langage particulier pour représenter les modèles sémantiques. Aussi l'annotation d'un objet WSDL2.0 par un concept dans un certain modèle peut être employée avec chaque élément dans WSDL2.0. Ce qui nous permet d'annoter toutes les fonctionnalités des services Web.

3.1.1 Description des propriétés annotées

Nous considérons que toutes les fonctionnalités : noms *de service, interfaces, opérations, entrées* et *sorties* des services Web sont annotés par un modèle sémantique de référence.

Les objets annotés dans notre approche sont de type décoration, identification de l'instance et de la référence (*cf Chapitre2. §3*).

Nous utilisons l'attribut *modelReference* qui peut être employé avec chaque élément dans WSDL et XML. Nous étendons sa définition dans SAWSDL à *wsdl:input* et *wsdl:output* afin de faciliter l'automatisation de la découverte et la composition des services Web.

Un *modelReference* sur une opération ou sur une erreur WSDL fournit des informations sémantiques sur cette opération, alors qu'un *modelReference* sur une interface WSDL fournit une classification ou d'autres descriptions sémantiques de l'interface.

3.1.2 Processus d'annotation

Le processus d'annotation consiste à prendre chaque fonctionnalité d'un service Web décrit en WSDL2.0 et à l'associer à l'élément correspondant dans un modèle sémantique de référence afin de produire un document SAWSDL (voir figure 10).

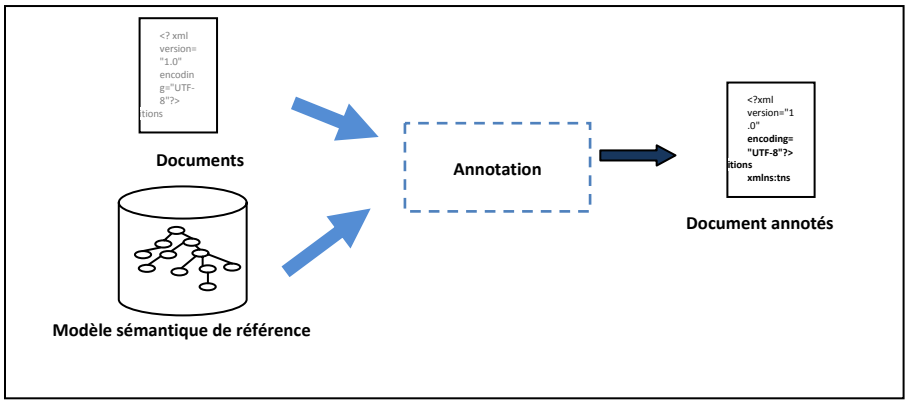


Figure 10 : Annotation des services Web

Pour faciliter l'annotation, plusieurs outils ont été développés (*cf. Annexe A*). Ces derniers ne sont pas adaptés aux annotations que nous souhaitons effectuer. Nos annotations sont donc effectuées manuellement par le fournisseur de chaque service Web. L'exemple suivant décrit un service en WSDL2.0 et le résultat de son annotation.

Un fragment de l'ontologie utilisée tout au long des exemples de cet article est présenté en figure 11.

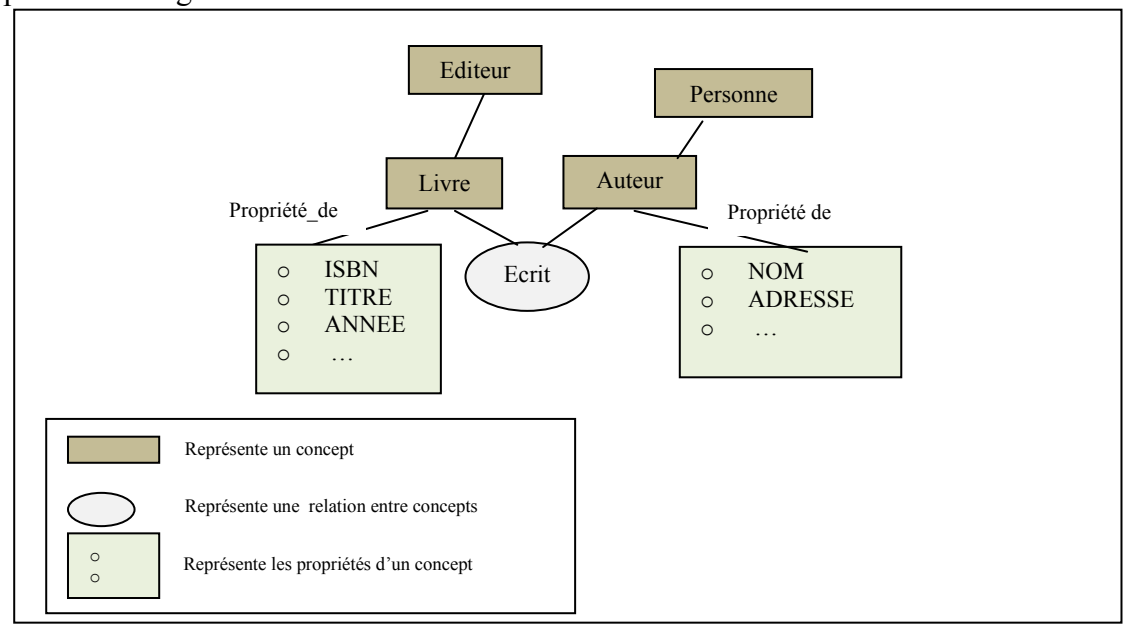


Figure 11: Fragment d'une ontologie « édition »

Exemple 1 :

Soit «Livreseditеurs» un service Web comprenant l'opération «op_GetLivreseditеurs» de type « in-out » permettant de retrouver les éditeurs des livres et ayant en entrée l'ISBN du livre défini par «GetInputcode» et en sortie le nom de son éditeur défini par «GetOutputnom».

Sémantiquement ce service se compose d'une opération qui permet de retourner le nom de l'éditeur des livres à partir de leur ISBN. Ce service est décrit en WSDL2.0 dans la figure12.

```
<wsdl:description …>
<definitions …>
  …
  <interface name="LivreseditеursInterface">
  <operation name="op_GetLivreseditеurs" pattern="http://www.w3.org/ns/wsdl/in-out">
                    <input message="tns:GetInputcode"/>
                    <output message="tns:GetOutputnom"/>
  </operation>
  </interface>
</definitions>
</wsdl:description>
```

Figure 12 : Exemple de service décrit en WSDL2.0

En faisant l'hypothèse que : « http://example.com » est l'URL de l'ontologie décrite dans la figure 11, le résultat de l'annotation sémantique du service Web décrit dans la figure 12 par l'ontologie est illustré dans la figure 13.

L'opération «op_Get Livreseditеur » fait référence au concept *Livrediteur*, l'entrée de l'opération « GetInputCode» fait référence au concept *ISBN* et la sortie «GetOutputnom» fait référence au concept *nom*.

```
<wsdl:description …>
<definitions …>
  …
  <interface name="LivreseditеursInterface"
sawsdl:modelReference="http://example.com/categorization/livres">
        <operation name="op_GetLivreseditеurs" pattern="http://www.w3.org/ns/wsdl/in-out"
sawsdl:modelReference=" http://example.com//Livresditeur#Livreseditеur">
              <input message="tns:GetInputCode"
sawsdl:modelReference=" http://example.com/Livresditeur#ISBN"/>
              <output message="tns:GetOutputnom"
sawsdl:modelReference=" http://example.com/Livresditeur#nom"/>
        </operation>
  </interface>
</definitions>
</wsdl:description>
```

Figure 13 : Exemple de service annoté en SAWSDL

3.2 Publication du service Web

Une fois la description du service Web annotée, le service Web va être publié pour pouvoir effectuer sa découverte. Comme illustré dans la figure 15, la publication dans notre approche consiste à effectuer les deux taches suivantes :

1. Un enregistrement dans le registre UDDI du document WSDL2.0 annoté et de l'URL du modèle sémantique de référence utilisé pour l'annotation, afin qu'il puisse être localisé par le *demandeur de service*.
2. Un enregistrement des fonctionnalités des services dans un registre appelé « *base de liens sémantiques* ».

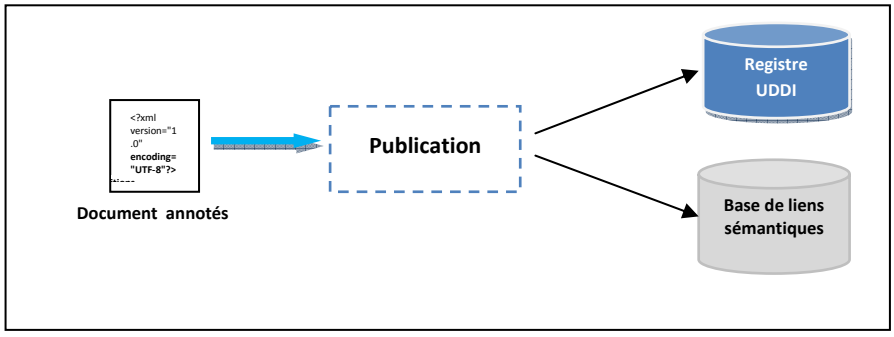


Figure 15 : Publication du service annoté

3.2.1 Publication du service dans le registre UDDI

La publication dans le registre UDDI ne stocke pas les descriptions WSDL et n'oblige pas à utiliser WSDL. Elle fait uniquement référence à WSDL.
Par conséquent, la publication du document WSDL2.0 annoté peut donc s'effectuer sur le registre UDDI3.0 de la même façon que la publication d'un document WSDL. Elle consiste à suivre les étapes suivantes :

1. Séparer la partie abstraite (partie comprenant *elementDeclaration, typeDefinition,* et *Interface)* du service représentant le fichier SAWSDL du WSDL2.0 de la partie concrète du service (partie comprenant *Bindings* et *services).*

2. Enregistrer la partie abstraite comme un *technical model* (tModel). Ce dernier permet de pointer vers une description externe. *tModel* va référencer le document SAWSDL**.**

   En considérant que la partie abstraite du service annoté est décrite dans « http://localhost/exemplesawsdl.wsdl », le *tModel* correspondant sera décrit comme suit :

```
<uddi:tModel …>
…
<uddi:overviewURL> http://localhost/exemplesawsdl.wsdl
</uddi:overviewURL>
…
</uddi:tModel>
```

3. Enregistrer le fournisseur de service comme une *Business Entity*. Ce dernier contient la description du fournisseur (nom de l'entreprise, liste de contacts, etc).
   En considérant que « CERIST TEST » est le fournisseur du service dont « user test » représente un contact, le *Business Entity* correspondant est décrit comme suit :

```
<uddi:businessEntity …>
…
<uddi:name> CERIST TEST </uddi:name>
<uddi:description xml:lang="en"> Exmple d'organisme </uddi:description>
<uddi:contacts>
<uddi:contact useType="Technical contact">
<uddi:personName> user test </uddi:personName>
<uddi:phone useType="voice">+213121916205</uddi:phone>
<uddi:phone useType="fax">+213121916205</uddi:phone>
<uddi:phone useType="mobile">+213610000000</uddi:phone>
<uddi:email>abenna@cerist.dz</uddi:email>
…
</uddi:businessEntity>
```

4. Enregistrer un *Business service* associant la *Business Entity* avec le *technical model. Business service* réalise le lien entre une interface de service (un *tModel*) et son implémentation, par l'intermédiaire de sous-éléments *bindindTemplates***.**

   En considérant que la partie concrète du service est décrite dans « http://localhost/exempleimplementationsawsdl.wsdl », le *Business service* correspondant sera décrit comme suit :

```
<uddi:businessService …">
…
<uddi:bindingTemplate…>
<uddi:description xml:lang="fr"> ceci est l'implémentation du fichier WSDL
</uddi:description>
<uddi:accessPoint useType="http">
http://localhost/exempleimplementationsawsdl.wsdl
</uddi:accessPoint> …
<uddi:tModelInstanceInfo …>
…
</uddi:bindingTemplate>
 …
</uddi:businessService >
```

5. La structure du *bindingTemplate* du service va être publiée tel que son *tModelInstanceInfo* référence le *tModel* de l'interface SAWSDL du service Web comme suit :

```
<uddi:businessService…>…
<uddi:tModelInstanceInfo …>
…
<uddi:overviewURL> http://localhost/exemplesawsdl.wsdl
...
</uddi:tModelInstanceInfo>…
</uddi:bindingTemplate>
…
</uddi:businessService>
```

6. Référencer le modèle sémantique de référence utilisé pour l'annotation dans le *tModel*. Nous utilisons *keyedreferenceGroup* dans le cas d'utilisation de différents modèles sémantiques.

3.2.2 Publication dans la Base de liens sémantiques

Insérer les annotations dans le document a un impact négatif sur le volume du contenu et peut compliquer sa maintenance. SAWSDL ne fournit aucune sémantique : il pointe par l'intermédiaire de références d'URI vers des concepts sémantiques.

Nos annotations sont de ce fait stockées dans la *base de liens sémantiques* qui représente le médiateur entre les annotations des fonctionnalités de services Web et les composants de la requête.

La base de liens sémantiques a pour objectif de fournir un accès rapide aux informations sémantiques relatives aux fonctionnalités de services et des correspondances inter services.

Elle permet ainsi d'optimiser le traitement des requêtes, et de restreindre l'interrogation du registre UDDI uniquement aux services susceptibles de répondre à la requête. Elle nécessite cependant, d'être mise à jour à chaque publication d'un nouveau service ou lors de la modification de la description du service.

Notre choix s'est porté sur une description de cette base en standard XML car la réponse à la requête inclura l'interrogation de la base de liens sémantiques et les registres UDDI. L'utilisation de XML offre la possibilité d'interopérabilité avec les autres couches. Le résultat retourné est un document XML qui peut servir comme entrée pour une autre interrogation.

La structure de notre *base de liens sémantiques,* définie en schéma XML, et représentée dans la figure ci-dessous en diagramme des classes UML, comprend les composants suivants :

- *Concept* : Représente un concept du modèle sémantique. Un concept peut être associé à la classe *Entree, Sortie* ou *Opération*.
- *Correspond* : Cette classe représente les mises en correspondance entre concepts. Ces mises en correspondance peuvent être de types différents (Exact, Subsume, Disjoint).
  - Exact : lorsque deux concepts C_1 et C_2 sont équivalents.
  - Subsume: si un concept C_2 englobe un concept C_1.
  - Disjoint : si les deux concepts sont disjoints.

  L'association du type de correspondance entre concepts permettra d'augmenter la qualité de la réponse
- *Operation :* Représente l'annotation de l'opération d'un service et comprend sa référence au modèle sémantique de référence et son type dans WSDL2.0 (in-only, in-out,…)
- *Entree :* Cette classe représente les entrées d'un service.
- *Sortie :* Cette classe représente les sorties d'un service.
- *Service :* La classe service comprend l'identificateur du registre UDDI auquel le service appartient, la référence à l'annotation sémantique de l'interface du service et de ses opérations.
- *Compose :* Cette classe représente les services pouvant être composés avec le service courant avec la condition de composition entre deux services. Nous supposons que les compositions entre services sont prédéfinies et disposons d'une fonction qui pour un service donné, retourne l'ensemble des services pouvant être composés avec lui [69,70].

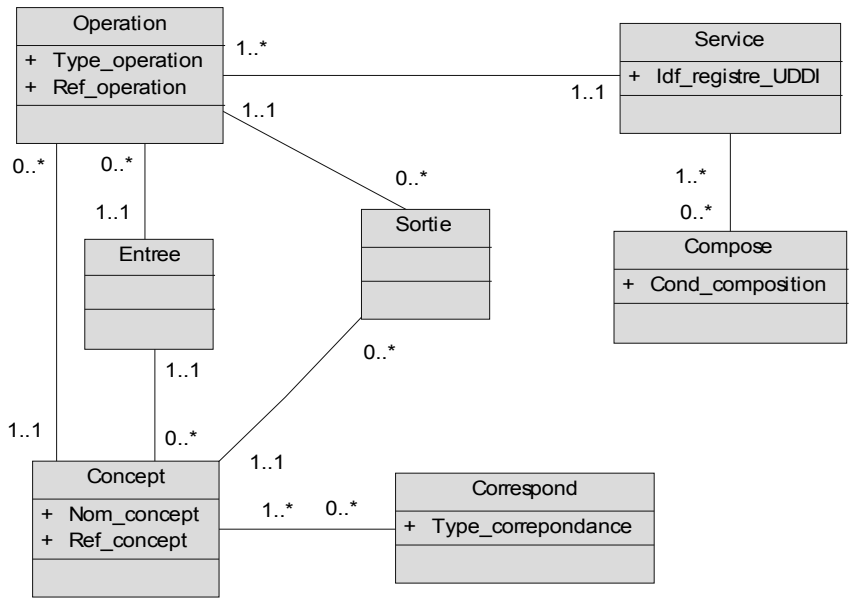


Figure 16 : Description de la base de liens sémantiques

L'enregistrement dans la *base de liens sémantiques* consiste à effectuer les tâches suivantes :

1. Associer la fonctionnalité du service Web au concept correspondant et enregistrer la fonctionnalité selon son type (Entrée, Sortie ou Opération) dans la classe appropriée et les informations relatives à son service dans la classe *Service*. Si le concept n'existe pas, effectuer la tâche 2 sinon effectuer la tâche 3.

   Exemple :
   Soit un service « SW_1 » comprenant une opération « Op_1» sur les livres et qui pour une entrée ISBN donnée identifiée «E_1_ISBN » retourne en sortie le titre du livre identifié par « S_1_titre ».
   Le service « SW1 » sera associé à la classe *Service*, l'opération « Op_1 » à la classe *Opération*, l'entrée de l'opération « E_1_ISBN » à la classe *Entree*, et la sortie de l'opération « S_1_titre » à la classe *Sortie.*

2. Rechercher le ou les concepts sémantiques correspondants à chaque annotation d'une fonctionnalité de service Web, avec le type de correspondance, l'insérer dans la classe *Correspond* et effectuer la tâche 1.

   Exemple :
   En considérant les annotations entre les composants du service SW1 et le model de référence comme suit :
   «E_1_ISBN» est annoté par *ISBN*, «S_1_titre » est annoté par *Titre*, et «Op1 » est annoté par *Livre_Edition*.
   Les concepts *ISBN, Titre, Livre_Edition* et leur concepts correspondant seront insérer dans les classes *Concept et Correspond.*

3. Enregistrer dans la classe *Compose*, les services Web pouvant être composé avec le service courant avec la condition de composition; ceci revient à parcourir la *base de liens sémantiques* et à rechercher les services qui peuvent sémantiquement être composés.

   Exemple :
   Si les services SW_2, SW_3 peuvent être composés avec le service SW_1. Ils seront donc enregistrer dans la classe *Compose*.

Le schéma XML de la *base de liens sémantiques* et un exemple de cette base sont décrits en Annexe B.

3.3 Formulation de la requête

Après annotation et publication du service, le *demandeur* de s*ervices* peut interroger un *annuaire de services* pour rechercher les services Web appropriés. La requête est formulée en se référant à une ontologie de domaine décrite en OWL.

L'utilisation d'une ontologie permet d'une part de formuler les requêtes sur la base d'une sémantique commune, et d'autre part de traiter les requêtes et de les diriger vers les services Web pertinents.

Une description en OWL contient trois catégories d'information:

- Une *Classe* du domaine représentant un prédicat sur les individus ;
- Une *relation* entre deux classes ;
- Une *relation* entre une classe et une propriété.

Une requête comprend ainsi deux parties : une partie propriétés recherchées, définie par la clause <propriete_concept>, et une partie condition définie par la clause <condition>. Elle est de la forme :

```
Requete::= <propriete_concept>('WHERE'<condition>)?
<propriete_concept>::= <propriete>(','<propriete_concept>)*
<condition>::=<propriete> <operateur> <valeur>
                    ('AND'<condition>)* | <relation> ('AND'<condition>)*
```

Chacun des composants de la requête est décrit dans le tableau suivant:

Composants de la requête	Description
Propriété	Relation entre une classe et une propriété. Soit C_i une classe de l'ontologie *O,* une relation entre une propriété P_i et la classe C_j est dénotée par $C_j.P_i$.
Relation	Une relation entre deux classes. Soient C_i, C_j deux classes de l'ontologie, une relation entre ces deux classes est dénotée par $R(C_i, C_j)$ tels que: les classes C_i, C_j appartiennent aux concepts des propriétés recherchées.
Opérateur	Opérateur logique pouvant prendre les valeurs suivantes : '<', '≤', '=', '>', '≥', '≠', 'CONTIENT'
Valeur	Représente une valeur atomique.
::=	L'élément à gauche du symbole est défini par les éléments de droites
|	Le symbole « | » définit une alternative
?	Le symbole « ? » signifie que l'élément est optionnel
*****	Le symbole « * » signifie que l'élément est optionnel ou répétitif.

Tableau 4: Description des composants de la requête

Exemple 2 :
Soient «ISBN», «Titre», «Annee» des propriétés de la classe « Livre », et soient « Nom », « Adresse » des propriétés de la classe « Auteur ».
La relation «Ecrit (Livre, Auteur) » est une relation entre le concept «Livre» et le concept «Auteur». (Voir figure 11)
Si la requête consiste à retrouver « ISBN », « Annee », « Titre » du livre et le « Nom » de son auteur pour les livres édités avant l'année 2005.
La requête est formulée comme suit :

```
REQUETE ::= Livre.ISBN, Livre.Annee, Livre.titre, Auteur.nom
                WHERE  Ecrit (Livre, Auteur)  AND Livre. Annee <2005
```

3.4 Découverte des services Web

La découverte des services Web revient à trouver les mises en correspondance entre les fonctionnalités des services Web et la requête, au traitement de cette dernière et puis à son interprétation en langage XQUERY.

3.4.1 Mise en correspondance entre les services Web et la requête

Les fonctionnalités des services Web peuvent être annotées par des représentations sémantiques différentes et pas toujours les mêmes que celles sur lesquelles l'utilisateur formule sa requête (voir figure ci dessus). Une mise en correspondance donc est nécessaire entre les annotations des services Web, entre les schémas, et entre les annotations de services Web et les concepts utilisés dans la requête. Ces mises en correspondance sont décrites dans les prochaines sections.

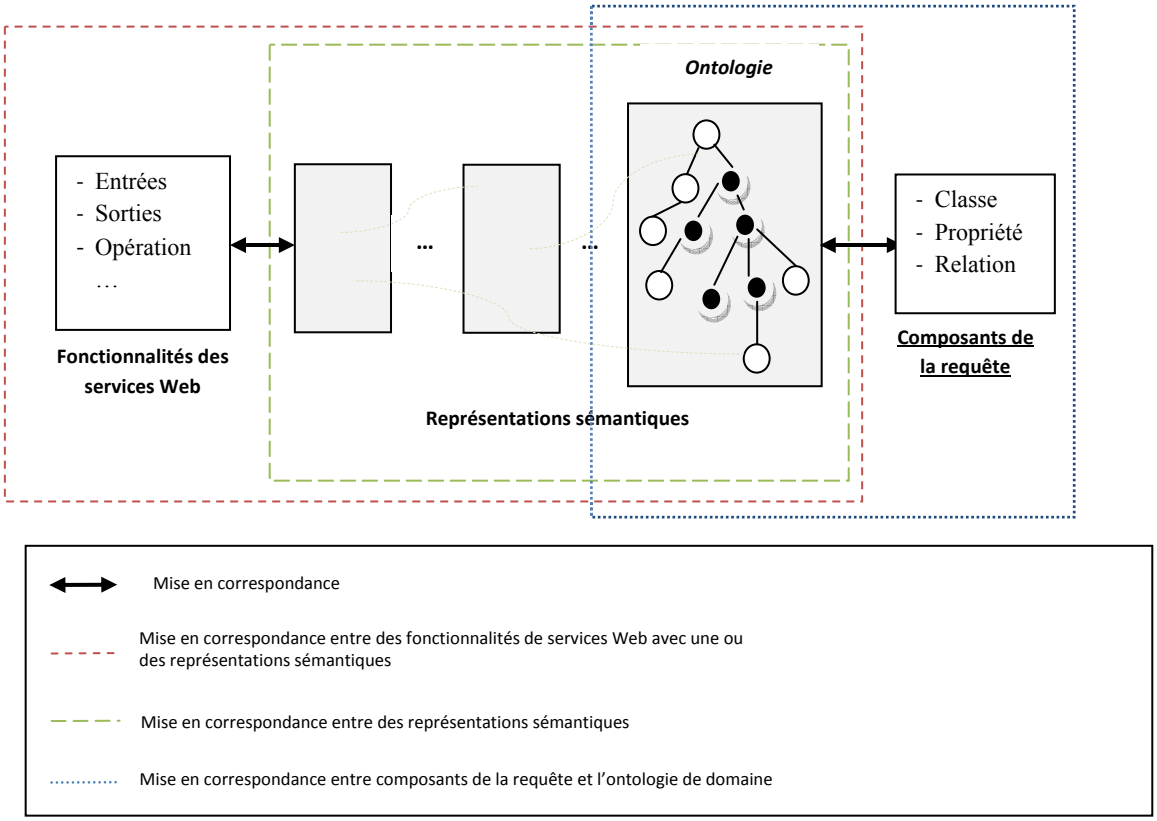


Figure 17 : Mise en correspondance entres services Web et requête

3.4.1.1 Mise en correspondance entre fonctionnalités de services Web

Les réponses à une requête, peuvent se trouver dans un seul service Web, comme ils peuvent faire l'objet d'une composition d'un ensemble de résultats partagés par différents services.

Pour cela, nous avons défini un ensemble de règles à appliquer pour effectuer des compositions entre services en se basant sur leurs fonctionnalités.

Règle 1 : Transitivité des fonctionnalités (entrées, sorties, opérations)

Deux fonctionnalités de même type (entrée, sortie, opération) sont considérées équivalentes si elles font référence à un même concept ou à un concept similaire.

Soient :
« $SW_i : E_k$ » l'entrée k du service Web SW_i
« $SW_j : E_l$ » l'entrée l du service Web SW_j
« $O : E_p$ » composant p du modèle sémantique de référence O.

Les préfixes SW et O sont utilisés pour distinguer entre les prédicats du service Web et le modèle sémantique de référence.

Le symbole ⇔ désigne une mise en correspondance (exact, subsume) entre deux éléments de type de données identiques.

Si *($SW_i : E_k \Leftrightarrow O : E_p$* ***et*** *$SW_j : E_l \Leftrightarrow O : E_p$)* ***alors*** *$SW_i : E_k \Leftrightarrow SW_j : E_l$*

Exemple 3 :

Soient les services SW_1 et SW_2 définis comme suit :
- L'opération « op_Livre » du SW_1 a comme entrée « in_titre » et comme sorties «out_date_edition », «out_nom_auteur ».
- L'opération « op_Auteur » du SW_2 a comme entrée « in_nom » et comme sortie « out_date_premiere_edition ».

Si *(SW_1 : out_nom_auteur ⇔ O : Nom)* ***et*** *(SW_2 : in_nom ⇔ O : Nom)*
alors *SW_1 : out_nom_auteur ⇔ SW_2 : in_nom*

Règle 2 : Principe de composition séquentielle entre services

Consiste à rechercher le service Web dont la sortie d'une opération est similaire à l'entrée de l'opération d'un autre service. Soit
- Un service « SW_i » qui a dans une opération « OP_k » comme sortie « $SW_i : S_m$»
- et un service « SW_j» qui a dans une opération « OP_l » comme entrée « $SW_j : E_n$».

Si *($SW_i : S_m \Leftrightarrow O : E_p$* ***Et*** *$SW_j : E_n \Leftrightarrow O : E_p$)*
alors nous pouvons effectuer une composition séquentielle entre *SW_i et SW_j.*

Exemple 4 :
Soient les services SW_1 et SW_2 définis comme suit :
- L'opération « op_Livre » du SW_1 a comme entrée : « in_titre » et comme sortie : « out_ Reference ».
- L'opération « op_Livre_editer » du service Web SW_2 a comme entrée : « in_identificateur » et comme sorties : « out_coût », « out_nom_auteur ».

Si (*SW_1 : out_*Reference⇔ *O : ISBN* **Et** *SW_2 : in_ identificateur ⇔ O : ISBN)*
alors *les services SW_1 et SW_2 peuvent être composés*

Règle 3 : Principe de composition parallèle entre services

Consiste à rechercher les services Web ayant des entrées d'opérations sémantiquement identiques pour une exécution parallèle.
Soit un service « SW_i » qui a dans une opération « OP_k » comme entrée « $SW_i : E_m$».
Et un service « SW_j» qui a dans une opération « OP_l » comme entrée « $SW_j : E_n$».

Si *(SW_i : $E_m \Leftrightarrow O : E_p$* ***Et*** *$SW_j$: $E_n \Leftrightarrow O : E_p$)*
alors *nous pouvons effectuer une composition parallèle entre SW_i et SW_j*

Dans le reste du document, le terme composition de services est utilisé pour indiquer une composition entre deux opérations de services.

Exemple 5 :
Soient les services SW_1 et SW_2 définis comme suit :

L'opération « op_Livre » du SW_1 a comme entrée : « in_ISBN » et comme sorties : «out_ titre». L'opération « op_Livre_editer » du service Web SW_2 a comme entrée : « in_identificateur » et comme sorties : « out_coût », « out_nom_auteur ».

Si (*SW_1 : in_ISBN$\Leftrightarrow O$: ISBN* **et** *SW_2 : in_ identificateur $\Leftrightarrow O$: ISBN)*

alors les services SW_1 et SW_2 peuvent être composés.

3.4.1.2 Mise en correspondance entre modèles sémantiques

Pour répondre à une requête, exprimée avec les termes de l'ontologie, nous avons besoin de mise en correspondance entre les composants de la requête et les annotations des fonctionnalités des services Web et donc d'une mise en correspondance sémantique entre leurs modèles.

La mise en correspondance entre schéma a motivé de nombreux travaux. Des synthèses des techniques de mise en correspondance de schémas ou d'ontologies ont été présentées [32, 40, 41]. Nous donnons un aperçu de ces techniques en Annexe C.

Ce point ne fait pas l'objet de notre travail, où nous considérons que le service Web et la requête font référence à un même modèle sémantique décrit en OWL et que les mises en correspondance entre concepts sont stockés dans la classe *Correspond* de la *base des liens sémantiques* avec leur type de correspondance qui peut être : Exact, subsume ou disjoint.

3.4.1.3 Mise en correspondance entre composants de la requête et les services Web

Le principe de mise en correspondance entre les fonctionnalités des services Web et composants de la requête sont décrits dans le tableau 5.

Composant de la requête	Interprétation au niveau des services Web
Une classe C_i de l'ontologie O	Peut correspondre à une opération «Op_n» du service Web «SW_k» *Notation : $SW_k : Op_n \Leftrightarrow C_i$*
Une propriété P_j d'une classe C_i	Peut correspondre à une entrée « E_l » ou à une sortie « S_l » d'un service Web «SW_k». *Notation : SW_k:$E_l \Leftrightarrow C_i.P_j$ ou $SW:S_l \Leftrightarrow C_i.P_j$*
Deux propriétés $C_i.P_k$, $C_i.P_j$ appartenant à une même classe C_i	- Peut être un service Web «SW_k» ayant une opération «Op_n» qui correspond à la classe C_i et ses sorties S_l, S_n correspondent aux propriétés « $C_i.P_k$» et «$C_i.P_j$». *Notation :* $SW_k:S_l \Leftrightarrow C_i.P_k$ Ou $SW_k:S_n \Leftrightarrow C_i.P_j$ *et* $SW_k:Op_n \Leftrightarrow C_i$

	- Peut être une composition parallèle, en appliquant la règle 3 de la section §3.4.1.1, entre deux services Web SW_k, SW_l dont les sorties $SW_k:S_l$, $SW_l:S_n$ correspondent aux propriétés $C_i.P_k$, $C_i.P_j$ et les entrées $SW_k:E_m$, $SW_l:E_n$ sont identiques. *Notation :* $SW_k:S_l \Leftrightarrow C_i.P_k$, $SW_l:S_n \Leftrightarrow C_i.P_j$, $SW_k:E_m \Leftrightarrow SW_l:E_n$ - Peut être une composition séquentielle, en appliquant la règle 2 de la section §3.4.1.1, entre deux services Web dont les sorties correspondent aux propriétés *« $C_i.P_k$» et «$C_i.P_j$»*. *Notation :* $SW_k:S_l \Leftrightarrow C_i.P_k$ et $SW_l:S_n \Leftrightarrow C_i.P_j$ et $SW_k:E_m \Leftrightarrow W_l:E_n$
Une relation R_m (C_i, Cj) entre les classes C_i, C_j	Peut être un service Web «SW_k» ayant une opération «Op_n» correspondant à « R_m » telle que : l'opération «Op_n» comprend des sorties S_l, S_n correspondant aux classes C_i ou C_j *Notation :* $SW_k : Op_n \Leftrightarrow R_m (C_i,C_j)$ et $\exists SW_k : S_l \Leftrightarrow C_i.P_m$ et $\exists SW_k : S_n \Leftrightarrow C_j.P_q$

Tableau 5 : Correspondance entre composants d'une requête et service Web

3.4.2 Principes de traitement de la requête

Le traitement de la requête consiste à analyser et décomposer la requête en une liste de propriétés à rechercher, une liste de relations entre classe et une liste de conditions sur les valeurs de propriétés, puis à effectuer les tâches suivantes :

1. Interroger la *base de liens sémantiques* pour rechercher les services Web ayant au moins une des propriétés recherchées.
2. Effectuer les traitements relatifs aux clauses de la requête. Ces traitements sont répartis en trois cas :
   Cas 1 : Les propriétés recherchées appartiennent à la même classe,
   Cas 2 : La partie condition comprend des relations entre classes,
   Cas 3 : La partie condition comprend des restrictions de valeurs sur des propriétés.
3. Interroger le registre UDDI pour déterminer le fournisseur de service et le localiser.

L'algorithme général de traitement de la requête est présenté en section §3.4.3.

Nous considérons les différents cas de traitement de la requête avec des exemples les illustrant :

- *Cas 1 : Les propriétés recherchées appartiennent à une même classe.*

Lorsque les propriétés recherchées appartiennent à une même classe, les services Web pouvant y répondre sont :

- **Cas 1.1** : Des services Web dont les sorties correspondent (exact, subsume) aux propriétés recherchées et sont produites par la même opération.

Exemple 6 :

Si la requête consiste à retrouver « ISBN» et «Titre» du livre. Les services Web pouvant répondre à la requête sont ceux dont au moins les sorties correspondent aux propriétés « ISBN », «Titre» et produites par la même opération.
Soit le service SW_1 défini comme suit : L'opération «op_list_Livre » du SW_1 a comme sorties : «out_titre_livre», «out_identicateur_livre», «out_nom_auteur». Tel que : SW_1:«out_titre_livre»⇔ O:Titre et SW_1:«out_identicateur_livre»⇔ O:ISBN.

Dans ce cas SW_1 est considéré comme un service Web répondant à la requête.

- **Cas 1.2 :** Des services Web dont les sorties correspondent aux propriétés recherchées et sont produites par des opérations différentes ou à des services Web différents mais ont des entrées identiques. Une composition parallèle peut être effectuée (cf. §3.4.1.1 règle 3).

Exemple 7 ***:***

Soit le service SW_2 défini par les opérations «op_coût_Livre» et «op_editeur_livre » comme suit :
L'opération «op_cout_Livre » du SW_2 a comme sorties : «out_ISBN», «out_coût» et l'opération «op_editeur_livre » du SW_2 a comme entrée : «in_ISBN» et comme sorties : «out_titre », «out_année_edition», «out_nom_auteur». Tel que :
SW_2 :« out_titre »⇔O: Titre, SW_2: « out_ISBN »⇔O: ISBN, SW_2: « in_ISBN »⇔ O:ISBN.
Et soit le service SW_3 défini par l'opération «op_editeur_Livre» qui a comme entrée : «in_identificateur_livre» et comme sorties : «out_date_edition», «out_auteur».
SW_3 : «op_editeur_Livre» ⇔ O: éditeur
SW_3: « in_identificateur_livre » ⇔ O: ISBN
SW_3: «out_date_edition»⇔ O: Edition
Si la requête consiste à retrouver «Titre» et « Edition» du livre. L'entrée « ISBN» est identique aux deux services SW_2 et SW_3. Ces derniers peuvent être composés pour retourner le résultat souhaité (cf. §3.4.1.1 règle 3).

- **Cas 1.3 :** Des services Web dont les sorties correspondent aux propriétés recherchées et sont produites par une même ou différentes opérations et à des services Web différents mais où la sortie d'un service Web représente une entrée de l'autre service. Une composition séquentielle peut être effectuée (cf. §3.4.1.1 règle 2).

Exemple 8 :

Si la requête consiste à retrouver «Titre», «Edition» de la classe « livre » et le « nom » de la classe «auteur». L'entrée « ISBN» du SW_3 constitue une sortie du service Web SW_1. Une composition séquentielle entre les services SW_1 et SW_3 peut retourner le résultat souhaité.

- *Cas 2 : La partie condition de la requête comprend des relations entre classes.*

Soit R une relation entre deux classes C_1 et C_2, les services Web pouvant y répondre sont :

- **Cas 2.1** : Les services Web similaires à ceux du cas 1.1 mais où l'opération correspond à R.

Exemple 9 :

Soit le rôle «Ecrit (Livre, Auteur) » une relation entre la classe «Livre» et la classe «Auteur».

Soit le service Web SW_1 définit dans l'exemple 6 telle que l'opération «op_list_Livre» est mise en correspondance avec le rôle Ecrit (Livre, Auteur).
SW_1 : «op_list_Livre» ⇔ O: Ecrit
SW_1 : «out_nom_auteur» ⇔ O: nom

- Si la requête consiste à retrouver « ISBN », « Titre » de la classe « livre » et le « Nom » de la classe « Auteur ». Alors le service Web SW1 peut répondre à la requête.

- **Cas 2.2** : Les services Web similaires à ceux du cas 1.2 mais où l'opération correspond à R.

Exemple 10 :

Soit SW_3 : «out_auteur»⇔ O: nom
Soit SW_2 : «op_editeur_livre»⇔ O: Ecrit
Si la requête consiste à retrouver «Edition», « Titre » de la classe « livre » et le « Nom » de la classe « auteur ». Alors le service Web SW_2, dont l'une des opérations est « op_editeur_Livre » et retournant : «out_titre_livre», et «out_nom_auteur», sera composé avec le service Web SW_3, dont l'une des opérations « op_editeur_Livre» et retournant : «out_date_edition» et «out_auteur», pour répondre à la requête.

- **Cas 2.3 :** Les services Web similaires à ceux du cas 1.3 mais où l'opération correspond à R.

Exemple 11:

Soit SW_3: «out_auteur»⇔ O: nom
Soit SW_1: «op_list_livre»⇔ O: Ecrit
Soit SW_1: «op_editeur_livre»⇔ O: Ecrit
Si la requête consiste à retrouver «Titre», «Edition» de la classe « livre » et le « nom » du classe « auteur ». L'entrée « ISBN» du SW_3 constitue une sortie du service Web SW_1. Une composition donc entre les deux services SW_1 et SW_3 peut retourner le résultat escompté.

- *Cas 3 : La partie condition de la requête comprend des restrictions de valeurs sur les propriétés.*

Dans ce cas, nous supposons que les valeurs sont de même type de données et nous ne traitons pas les conflits pouvant surgir d'un problème lié aux types de données. Les services Web pouvant y répondre sont :

- **Cas 3.1 :** Les services Web ayant des sorties correspondant à au moins une des propriétés recherchées et dont l'entrée correspond à une propriété sur laquelle une restriction de valeur est exigée.

Exemple 12 :

Soit le service SW_4 défini par les opérations «op_coût_Livre» et «op_editeur_livre » comme suit :
L'opération «op_editeur_livre » du SW_4 a comme entrée : «in_date_edition » et comme sorties : «out_ISBN», «out_nom_aut». Tel que : SW_4:«out_ISBN»⇔ O:ISBN SW_4:«in_date_edition»⇔ O:Edition et SW_4: «in_nom_aut »⇔ O:nom.

Si la requête consiste à retrouver « ISBN» du livre et le « nom » de son auteur pour les livres édités avant 2005. Le service Web SW_4 qui a en entrée «in_date_edition » peut répondre à la requête utilisateur.

- **Cas 3.2 :** Les services Web ayant des sorties correspondant à au moins une des propriétés recherchées et à une propriété sur laquelle une restriction de valeur est exigée.

Exemple 13 :

Si la requête consiste à retrouver « ISBN» du livre et le « nom » de son auteur pour les livres édités avant 2005. Le service Web SW_3 qui a comme sortie «out_date_edition» sera inclus dans les réponses à la requête.

- **Cas 3.3 :** Les services Web ayant en entrée la propriété sur laquelle une restriction de valeur est exigée et pouvant être composés avec des services Web ayant des sorties correspondant à au moins une des propriétés recherchées. Une composition séquentielle doit être effectuée (cf. §3.4.1.1 règle 2).

Exemple 14 :

Si la requête consiste à retrouver «titre» du livre édités avant 2005.
Le service Web SW_4 qui a en entrée : «in_date_edition » sera composé avec SW_2 qui a comme sorties : «out_ISBN», «out_coût», «out_titre», «out_année_edition», «out_nom_auteur» pour répondre à la requête.

3.4.3 Algorithmes de traitement de la requête

Soit Q (LPR, LD, LR) une requête où LPR représente la liste des propriétés recherchées, LD est l'ensemble des restrictions de valeur sur des propriétés et LR l'ensemble des relations entre classes.

- LPR = $\{LP_{1,...}, LPi_{,...} LP_k\}$ où $LP_i = \{C_i.P_1, C_i.P_{2,...}, C_i.P_j\}$ tel que : $C_i.P_l$ est une propriété P_l recherchée de la classe C_i.

- LD = $\{(D_1, ...D_i...D_n)\}$ où D_i est une restriction sur une valeur d'une propriété P_i d'une classe C_v. tel que : C_v ϵ aux classes de LPR et $D_i = C_v.P_i$ ® Z où Z est une valeur atomique et ® est un opérateur de type : $<, \leq, =, >, \geq, \neq$, CONTIENT.

- LR = $\{R_1, R_2,...R_l\}$ où R_i est une relation entre deux classes C_k, C_n tel que : $k \neq n$ et C_k, C_n ϵ aux classes de LP_i.

- La fonction TRAITER_CAS_i.j représente le traitement du cas i.j cités dans la section précédente. i, j = 1..3.

- La fonction TROUVE_SERVICE_SORTIE (LPR) retourne les services ayant en sortie au moins une des propriétés de LPR. Avec pour chaque service ses opérations, ses entrées et sorties et les services avec lesquels il peut être composé.

- La fonction TROUVE_SERVICE_ENTREE (LD) retourne la liste des services ayant en entrée au moins une des propriétés de *LD*.

- La fonction TROUVE_CONCEPT_CORRESPONDANT (prop) interroge la *base de liens sémantiques* pour retourner la liste des concepts correspondant à *prop* avec le type de correspondance.

- La fonction PRESENTATION_RESULTAT(LS_PR) présente les services répondant à la requête sous format d'un document XML. Avec pour chaque service ses fonctionnalités sémantiques, les conditions de son invocation et sa localisation.

- LS_PR retourne, sous la forme d'un document XML, la liste des services, susceptibles de répondre à la requête, avec pour chaque service aussi bien ses fonctionnalités et le type de correspondance, ses entrées, ses sorties et le registre UDDI auquel il appartient que les services avec lesquels il peut être composé.

L'algorithme de traitement de la requête est présenté dans la Table 6.

Algorithme Traitement_requete
Entrée Q (LPR, LD, LR)
Sortie : resultat_requete.XML
Debut
LS_PR := Ø ; /* LS_PR : les services ayant au moins une des propriétés recherchées*/
LS_PR :=TROUVE_SERVICE_SORTIE (LPR)
Si LS_PR≠Ø /***Traitement du cas1** */
Alors LS_PR :=VERIFIER_CONCEPT (LS_PR, LPR)
Fin si
Si (LR≠Ø ∧ LS_PR≠Ø**)** /***Traitement du cas 2** */
Alors LS_PR :=VERIFIER_ROLE (LS_PR, LR)
Fin si
Si (LD≠Ø ∧ LS_PR≠Ø) /***Traitement du cas3** */
Alors LS_PR :=VERIFIER_CONDITION (LS_PR, LD)
Fin si
Resultat_requete := PRESENTATION_RESULTAT (LS_PR)
Fin

Tableau 6 : Algorithme général de traitement de la requête

Les principales fonctions sont décrites dans les tables 7, 8, et 9.

```
Fonction VERIFIER_CONCEPT /*Traitement cas 1 cf. §3.4.2*/
Entrée : LS_PR, LPR, Sortie : LS_PR
Debut
Res_cas1 := Ø ; res_cas1.1 := Ø ; res_cas1.2 := Ø; res_cas1.3:= Ø ;
Pour chaque  LP ∈ LPR
Faire
        res_cas1.1 :=TRAITER_CAS_1.1 (LS_PR, LP)
        res_cas1.2 :=TRAITER_CAS_1.2 (LS_PR, LP)
        res_cas1.3 :=TRAITER_CAS_1.3 (LS_PR, LP)
Fin pour
Res_cas1 := res_cas1.1 ∪ res_cas1.2 ∪ res_cas1.3
Si Res_cas1 ≠ Ø
        Alors Pour chaque sw ∈ LS_PR
                Faire
                Si  sw ∉  Res_cas1 alors
                LS_PR := LS_PR \ {sw} /*supprimer sw de la liste des services*/ Fin Si
                Fin pour
        Sinon LS_PR := Ø
Fin Si
Return LS_PR
Fin
```

Tableau 7: Fonction VERIFIER_Concept

```
Fonction VERIFIER_ROLE (LS_PR, LR) /*Traitement cas 2 cf. §3.4.2*/
Entrée : LS_PR, LR, Sortie : LS_PR
Debut
Res_cas2 := Ø ; res_cas2.1 := Ø ; res_cas2.2 := Ø ; res_cas2.3:= Ø ;

        Pour chaque R ∈ LR
        Faire
                res_cas2.1 :=traiter_cas_2.1 (LS_PR, R)
                res_cas2.2 :=traiter_cas_2.2 (LS_PR, R)
                res_cas2.3 :=traiter_cas_2.3 (LS_PR, R)
        Fait
        Res_cas2 := res_cas2.1 ∪ res_cas2.2 ∪ res_cas2.3
Si Res_cas2 ≠ Ø
        Alors Pour chaque sw ∈ LS_PR
                Faire
                Si sw ∉  Res_cas2  alors
                LS_PR := LS_PR \ {sw} /*supprimer sw de la liste des services*/ Fin Si
                Fin pour
        Sinon LS_PR := Ø
Fin Si
Return LS_PR
Fin
```

Tableau 8 : Fonction VERIFIER_ROLE

```
Fonction VERIFIER_CONDITION (LS_PR, LD) /*Traitement cas 3 §3.4.2*/
Entrée : LS_PR, LD, sortie : LS_PR
Debut
Res_cas3:=Ø ; res_cas3.1:=Ø ; res_cas3.2 :=Ø ; res_cas3.3:=Ø ; LS_LD:= Ø
/* LS_LD: les services ayant au moins une des conditions recherchée comme entrée*/
Pour chaque D ∈ LD
Faire
      LS_LD :=TROUVE_SERVICE_ENTREE (D)
      Si LS_LD = Ø Alors
         res_cas3.2 :=traiter_cas_3.2 (LS_PR, D)
         Sinon
         res_cas3.1 :=traiter_cas_3.1 (LS_PR, D)
         res_cas3.3 :=traiter_cas_3.3 (LS_PR,D, LS_LD)
      Fin si
Fait
Res_cas3 := res_cas3.1 ∪ res_cas3.2 ∪ res_cas3.3
Si Res_cas3 ≠ Ø alors
                              Pour chaque sw ∈ LS_PR Faire
                              Si sw ∉  Res_cas3 alors
                              LS_PR := LS_PR \{sw}     /*supprimer sw de LS_PR*/ Fin Si
                              Fin pour
                               Sinon LS_PR := Ø
Fin Si
Return LS_PR
Fin
```

Tableau 9 : Fonction VERIFIER_Condition

```
<Resultat>
{ let $LCE:=local:concept_correspondant($p)  (:LCE est la liste des concepts equivalents:)
  for $i in ($p, $LCE//concept_idf)
  let $LS_PR:=local:trouve_service_sortie_correspondant($i)
  return
                <concept>
                {$i}
                        {if (not (empty($LS_PR))) then
                        let $LS_PR:=local:VERIFIER_CONCEPT($LS_PR)
                        return <VerifClasse>{$LS_PR}</VerifClasse> else 'Concept null'
                        }
                        {if (not((empty($LD)) and (empty($LS_PR)))) then
                        let $LS_PR:=local:VERIFIER_CONDITION ($LD, $LS_PR)
                        return <Verifcond>{$LS_PR} </Verifcond>else 'Condition null'
                        }
                        {
                        if (not((empty($LR)) and (empty($LS_PR))))  then
                        let $LS_PR:=local:VERIFIER_ROLE ($LR, $LS_PR)
                        return <VerifRole>{$LS_PR}</VerifRole> else 'Role null'
                        }
                         {
                         let $Resultat:=local:PRESENTATION_RESULTAT($LS_PR)
                         return
                         <resultat_final>{$Resultat} </resultat_final>
                         }
                        </concept>
</Resultat >
```

Tableau 10 : Algorithme général décrit en XQUERY

3.4.4 Transformation de la requête en XQUERY

La transformation de la requête consiste à utiliser XQUERY sur des documents XML dont le schéma est prédéfini : schéma de la *base de liens sémantiques* et des registres UDDI.
Elle comprend deux phases :

- La première phase consiste à récupérer les services Web appropriés et leur spécifications fonctionnelles et ceux en interrogeant la *base de liens sémantiques.*
- La deuxième phase permet de déterminer à partir du registre UDDI les informations relatives à l'invocation du service : son fournisseur, son adresse,…

Un aperçu de l'algorithme général présenté en section §3.4.3 décrit en XQUERY est illustré dans le tableau 10.

3.5 Présentation des résultats

Le résultat retourné après interrogation sera présenté dans un document XML et peut ainsi servir de point d'entrée pour une autre interrogation. La description du schéma du document résultat est la suivante :

Le nom du service, son fournisseur, la localisation de sa description, ses opérations et pour chaque opération ses entrées, ses sorties et les noms de services avec lesquels une composition est possible pour répondre à la requête.

4 Conclusion

Nous avons décrit dans ce chapitre notre approche d'annotation et d'interrogation des services Web. L'approche ainsi proposée est subdivisée en les étapes suivantes :

1. Annotations sémantiques de la description des services Web en s'appuyant sur la spécification SAWSDL.
2. Définition de la structure de la requête
3. Définition d'une base appelée « base de liens sémantiques » permettant de retrouver les mises en correspondance entre composants de la requête et services et donc de minimiser le temps de recherche lors de la phase de découverte.
4. Publication des services Web dans le registre UDDI, afin qu'il puisse être localisé par le *demandeur de service,* et enregistrement dans la *base de liens sémantiques* des annotations de services Web, afin de restreindre l'interrogation du registre UDDI uniquement aux services susceptibles de répondre à la requête, de fournir un accès rapide aux fonctionnalités des services Web et d'optimiser le temps de traitement de la requête.
5. Définition d'un ensemble de règles à appliquer pour la composition des services.
6. Traitement de la requête en XQUERY par l'interrogation de *la base de liens sémantiques* et du registre UDDI.

Le prochain chapitre est réservé à la présentation de quelques scénarii sur la solution proposée.

Chapitre 5 : Mise en œuvre

1. Introduction

Pour mettre en œuvre les idées présentées dans le chapitre précédent, nous avons implémenté un prototype. Nous entamons ce chapitre par une description des outils utilisés pour le développement du prototype, nous détaillerons un exemple de traitement d'une requête et présentons l'architecture fonctionnelle et les interfaces du prototype.

2. Outils de développement utilisés

Le prototype a été développé sous le système d'exploitation windows XP avec des outils souvent gratuits, open source, graphiques et développés en Java. Nous détaillons, dans ce qui suit, chacun des outils et langages utilisés pour la création et la validation de la *base de liens sémantique*s, ainsi que pour l'exécution des programmes.

- ALTOVA XMLSpy

  ALTOVA XMLSpy fait partie d'une suite de produits de Altova[48] qui ont pour but d'aider à la modélisation, l'édition, la transformation et le débogage des applications basées sur XML, XML Schema, XSL/XSLT, SOAP, WSDL et sur les technologies des services Web. Il est le choix de la plupart des sociétés Fortune 500 et Global 1000. Il comprend des fonctionnalités de débogage XSLT, d'édition WSDL, de conversion HTML vers XML, de génération de code C++/Java,...
  XMLSpy inclut entre autres l'outil XML Validator pour faciliter la correction de documents, un Analyseur XPath 1.0/2.0 aide à l'écriture d'expressions XPath et les outils XQuery Processor, XQuery Debugger et XQuery Profiler pour l'interrogation de données XML.
  Dans notre cas XMLSpy a été utilisé pour la conception et la validation du schéma XML de la *base de liens sémantique*s et les exemples de documents XML de *base de liens sémantique* et du registre UDDI3.0. Il a également été utilisé pour faciliter l'écriture et la validation des différents algorithmes exploitant XQUERY.

- Protégé 2000

  L'éditeur d'ontologies Protégé2000 permet la création et la visualisation d'ontologies décrite en OWL. Il est graphique et open-source et possède une riche bibliothèque de plugins.

[48] ALTOVA membre actif du World Wide Web Consortium (W3C) et Web Services Interoperability Consortium (WS-IC)

- API Jena

  L'accès et l'interrogation de l'ontologie sont effectués par l'utilisation de l'API Jena2.2 qui offre des méthodes pour la manipulation des ontologies [38].
  Jena est un kit Java pour le Web sémantique, issu d'un projet Open-source de «HP Labs Semantic Web Program ».

- API Nux

  L'accès aux sources de données de type XML s'effectue par l'interrogation des schémas source en langage XQUERY. Cet accès est rendu possible grâce à l'utilisation de l'API Nux1.6[49]. L'API Nux1.6 est développée en Java, elle est gratuite et open source. Nux étend les librairies XOM[50] et Saxon[51]. Elle utilise Saxon-B comme moteur de requêtes.

  Les fonctionnalités offertes sont nombreuses : interrogation en XQUERY et en XPath2.0 et les validations complexes.

- Java

  Le test a été implémenté en Java. Le choix de ce langage de programmation est la conséquence des choix des autres environnements de développement écrits en Java à savoir la libraire d'exploitation de l'ontologie Jena et l'Api d'interrogation des schémas XML en XQUERY Nux1.6.

  Ce langage offre aussi l'avantage d'être multi-plateforme, orienté objet. Il est doté d'une riche bibliothèque de classes comprenant : la programmation multitâches, la gestion des exceptions et la gestion des interfaces graphiques.

  Le programme d'utilisation de l'API Nux pour l'interrogation de la *base de liens sémantique*s et du registre UDDI en XQUERY est présenté en Annexe E.

3. Expérimentation

Notre prototype inclut l'utilisation des composants suivants :

- Un fragment d'ontologie de domaine « édition ».
- Le schéma de la Base de liens sémantiques. *(cf Annexe B)*
- Un exemple de Base de liens sémantiques. *(cf Annexe B)*
- Le schéma UDDI3.0 et des extraits de documents UDDI.

- L'ontologie

  Notre système inclut une ontologie de domaine décrite en OWL qui permet au demandeur de services d'exprimer sa requête. Chacune des fonctionnalités des services Web est annotée par cette ontologie.
  L'ontologie que nous avons utilisée est une ontologie du domaine « édition » conçue avec l'éditeur d'ontologie *protégé 2000*.
  Un fragment de l'ontologie utilisée a été présenté dans le chapitre 4, §3.1.2

[49] Nux1.6 : http://www.w3.org/XML/Query/
[50] XOM : http://www.xom.com
[51] Saxon : http://saxon.sourceforge.net

- ## Schéma de la base de liens sémantiques

Notre base de liens sémantiques, représentée dans la figure 16 en diagramme de classes UML, est décrite en schéma XML en Annexe B.

- ## Exemple de Base de liens sémantiques

Pour évaluer notre application, nous avons initialisé la base et effectué des expérimentations selon différents cas de figure (voir Tableau 11). Nous les présentons brièvement ci-dessous en fonction du nombre d'informations contenues dans chaque service.

- Un service satisfaisant toutes les clauses de la requête (Exemple : SW_6).
- Un service contenant certains concepts recherchés et pouvant être composé avec d'autres services mais où les conditions ne sont pas satisfaites (Exemple : SW_4).
- Un service ne répondant à aucune clause de la requête (Exemple : SW_7).
- Des services, au nombre de trois, contenant une partie des informations de la requête et pouvant être composés entre eux (Exemple : SW_1, SW_2, SW_3, SW_5).

Le résultat doit être une combinaison de tous les cas cités précédemment.

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 : S_1, SW_2:E_1)	UDDI1
				(SW_1 : S_1, SW_3:E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2:E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3:E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2, SW_5 :E_2)	
			S_8 :Livre.Specialite		
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		
SW_7	Op_3 : Domaine		S_8 :Livre.Specialite		UDDI2

Tableau 11: Description de liste des services

Notation :

SW_i : Désigne le Service Web i
S_j : Signifie la sortie j du service
E_k : Signifie l'entrée K du service
OP_j : Opération j du service
(SW_1 : S_1, SW_2: E_1) sur une sortie signifie que les deux services peuvent être composés séquentiellement sur la sortie indiquée.
(SW_2 :E_1//SW_3 :E_1) sur une entrée signifie que les deux services peuvent être exécutés parallèlement sur l'entrée indiquée.

La liste des concepts par service est présentée dans le tableau 12.

Service	ISBN (C1)	TITRE (C2)	Nom_auteur (C5, C4, C3)	Annee (C6)	Coût_livre (C10)	Domaine (C8, C9)	Maison d'Edition (C7)
SW_1	S_1		S_5				
SW_2	E_1			S_6			
SW_3	E_1	S_2			S_{10}		
SW_4		S_2	S_5			S_8	
SW_5		E_2	S_5		S_{10}		
SW_6	S_1	S_2	S_5	S_6			
SW_7						S_8	

Tableau 12 : Liste des concepts par service

4. Exemple d'exécution d'une requête

Nous avons présenté dans le chapitre précédent un exemple de requête visant la recherche de : « ISBN », « Annee », « Titre » du livre et le « Nom » de son auteur pour les livres édités avant l'année 2005

REQUETE ::= Livre.ISBN, Livre.Annee, Livre.titre, Auteur.nom
WHERE Ecrit (Livre, Auteur) **AND** Livre. Annee >2005

Après décomposition de la requête en liste de propriétés à rechercher, de relations entre concepts et de conditions à satisfaire, la requête citée dans l'exemple se présente comme suit :

- Liste des concepts : LPR = {Livre (ISBN, Annee, titre), Auteur (Nom)}
- Liste des rôles : LR = {Ecrit (Livre, Auteur)}
- Liste des conditions : LD = {Livre.Annee>2005}

Le traitement de cette requête est effectué comme suit :

4.1 Recherche des services ayant en sortie au moins une des propriétés recherchées.

La première étape de l'invocation de l'algorithme avec les paramètres initiaux (LPR, LD, LR) sur la liste des propriétés recherchées, la liste des relations entre concepts et la liste des conditions à satisfaire.

Si des services existent l'algorithme retourne la liste des services Web ayant au moins une des propriétés recherchées avec pour chaque service ses opérations, ses entrées et sorties et les services avec lesquels il peut être composé.

La fonction TROUVE_SERVICE_SORTIE (LPR) va retourner sous la forme d'un document XML tous les services sauf SW_7. Pour une meilleure lisibilité nous présentons le résultat sous forme de tableau, tel que décrit dans le tableau 13.

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1:Livre.ISBN	(SW_1 : S_1, SW_2:E_1)	UDDI1
				(SW_1 : S_1, SW_3:E_1)	
			S_5:Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6:Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2:E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2:Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}:Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3:E_1)	
SW_4	Op_2 : Livre		S_2:Livre. Titre	(SW_4 :S_2,SW_5 :E_2)	
	Op_1 : Auteur		S_5:Auteur.Nom		
SW_5	Op_2 : Livre		S_5:Auteur.Nom		UDDI3
			S_{10}:Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5:Auteur.Nom		
			S_6:Livre.Annee		

Tableau 13 : Exemple de services comprenant au moins un des concepts recherchés

4.2 Traitement des différents cas

Après récupération des services ayant au moins une des propriétés recherchés. Le traitement s'effectue selon les cas décrits dans le chapitre précédent. Des exemples de ces différents cas sont présentés dans les sections suivantes.

4.2.1 Exemple du traitement du Cas 1

Ce traitement fait appel à l'algorithme VERIFIER_CONCEPT (LS_PR, LPR) qui traite le cas où les propriétés recherchées appartiennent à la même classe. Pour notre exemple « ISBN », « Annee » et « Titre » appartienne à la même classe.

LPR = {Livre (ISBN, Annee, titre), Auteur (Nom)}

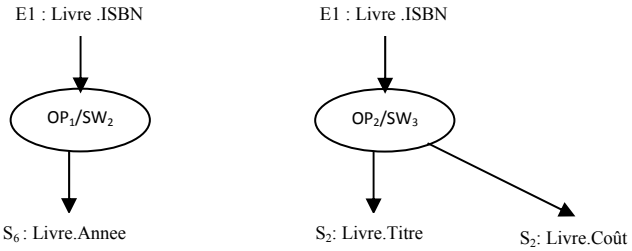
Les services susceptibles de répondre à la requête sont :

- **Cas 1.1** : Des services Web dont les sorties correspondent à « ISBN », « Titre » et « Annee » et sont produites par la même opération.
  Le résultat de ce traitement retourne le SW_6 où les propriétés recherchées sont produites par la même opération « Ecrit ».

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6:S_1, SW_2:E_1)	UDDI2
				(SW_6:S_1, SW_3:E_1)	
			S_2:Livre.Titre	(SW_6:S_2, SW_5:E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 14 : Exemple de traitement du cas 1.1

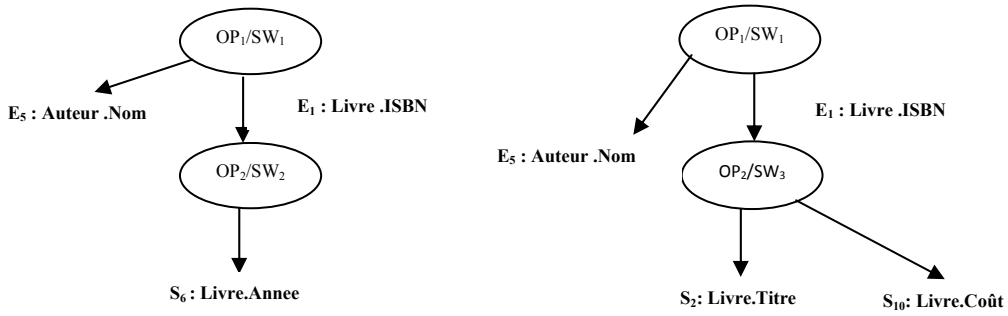
- **Cas 1.2 :** Les services Web dont les sorties correspondant aux propriétés « ISBN », « Titre » et « Annee » produites par des opérations différentes ou à des services Web différents mais ont des entrées identiques. Comme illustré dans le tableau 15 une composition parallèle peut être effectuée (cf. §3.4.1.1 règle 3) entre le SW_3 et SW_2.

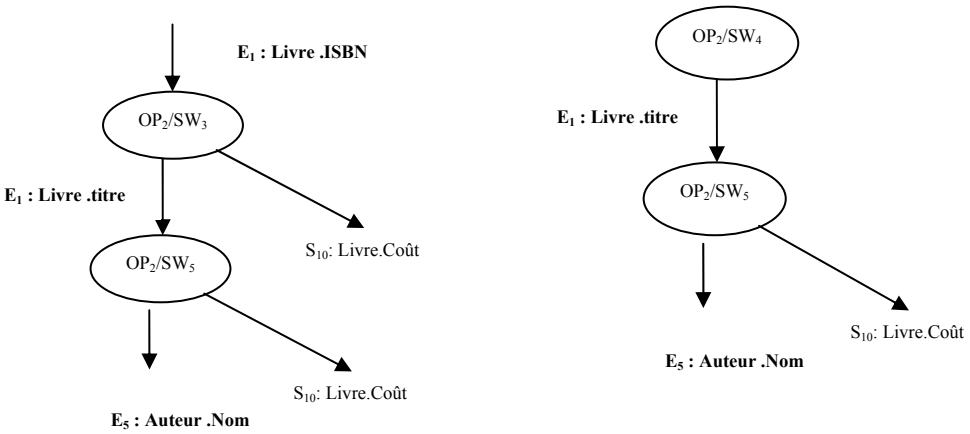


Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_2	Op_2 :Edition_Livre		S_6 :Livre.Annee		UDDI1
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2:E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3:E_1)	

Tableau 15 : Exemple de traitement du cas 1.2

- **Cas 1.3 :** Des services Web dont les sorties correspondent aux propriétés « ISBN », « Titre » et « Annee» sont produites par une même ou différentes opérations et à des services Web différents mais où la sortie d'un service Web représente une entrée de l'autre service. Comme illustré dans les figures ci après une composition séquentielle peut être effectuée (cf. §3.4.1.1 règle 2) entre les services : SW_1 et SW_2, SW_3 et SW_1, SW_3 et SW_5, SW_4 et SW_6.





Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1: S_1, SW_2: E_1)	UDDI1
				(SW_1: S_1, SW_3: E_1)	
			S_5 :Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 :Livre.Annee		
		E_1 : Livre.ISBN		(SW_1: S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1:S1, SW_3:E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2, SW_5 :E_2)	
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	

Tableau 16 : Exemple de traitement du cas 1.3

Ainsi le résultat final du traitement du cas 1 est le suivant :

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 : S_1, SW_2: E_1)	UDDI1
				(SW_1 : S_1, SW_3: E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3: E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2,SW_5 :E_2)	

	Op1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 :Livre		S_5 :Auteur.Nom		UDDI3
			S_{10}:Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 17 : Exemple de traitement du cas 1

La figure 19 illustre le résultat d'une composition antre service pour retourner le résultat escompté.

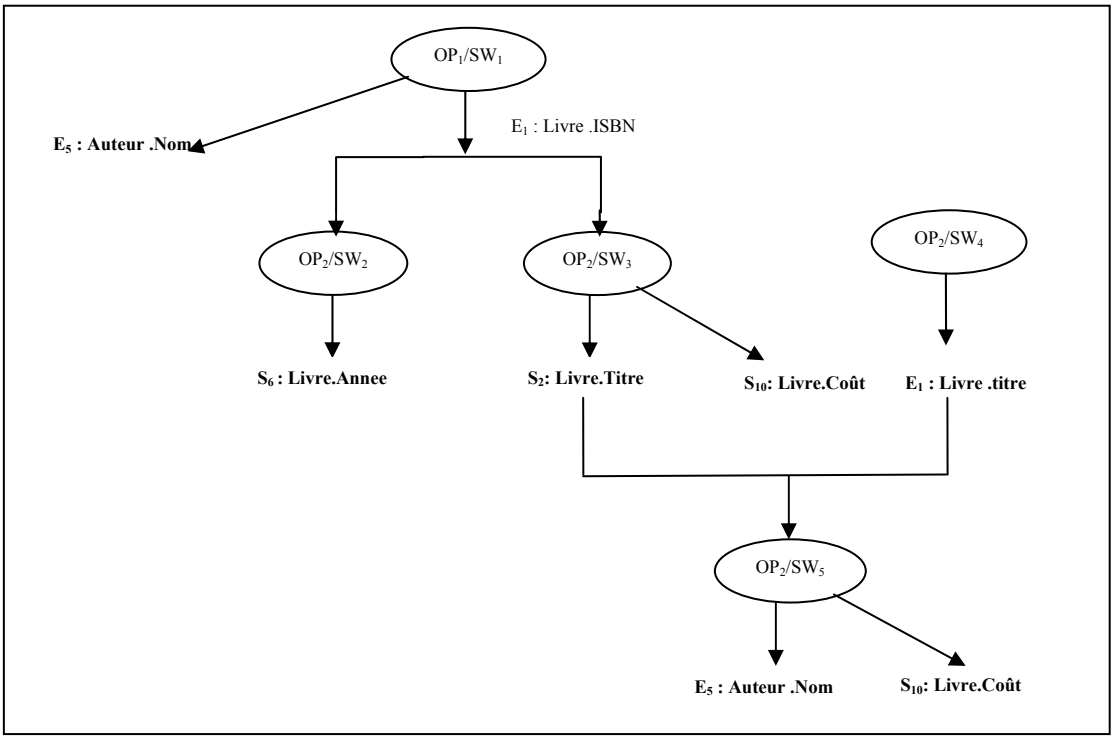


Figure 18: Exemple d'une composition pour satisfaire le cas 1

4.2.2 Exemple de traitement du Cas 2

Ce traitement fait appel à l'algorithme VERIFIER_ROLE (LS_PR, LR) où LR={Ecrit (Livre, Auteur)} et LS_PR représente le résultat du traitement du CAS1. Pour cet exemple, on considère que toutes les opérations en une correspondance avec le rôle « Ecrit ».

La partie condition de la requête comprend des relations entre la classe « Livre » et Auteur. Les services Web pouvant y répondre sont les suivants :

- **Cas 2.1 :** Le service SW_6 retourné dans le traitement du cas 1.1 mais où l'opération correspond à la relation « Ecrit ».

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 18 : Exemple de traitement du cas 2.1

- **Cas 2.2 :** Les services Web (SW_2, SW_3) retournés dans le cas 1.2 mais où l'opération doit correspondre à « Ecrit ».

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		Registre UDDI
		E_1 : Livre.ISBN		(SW_1:S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1:S_1, SW_3: E_1)	

Tableau 19 : Exemple de traitement du cas 2.2

- **Cas 2.3** : Les services Web (SW_1, SW_2, SW_3 SW_4, SW_5) retournés dans le traitement du cas 1.3 mais où l'opération doit correspondre à « Ecrit ».

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 :S_1, SW_2: E_1)	UDDI1
				(SW_1 :S_1, SW_3: E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 :S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3: E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2, SW_5:E_2)	
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	

Tableau 19 : Exemple de traitement du cas 2.3

Ainsi le résultat final du traitement du cas 2 est présenté dans le tableau 20.

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 :S_1, SW_2: E_1)	UDDI1
				(SW_1 :S_1, SW_3: E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 :S_1, SW_3: E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2, SW_5 :E_2)	
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 20 : Exemple de traitement du cas 2

4.2.3 Exemple du traitement du Cas 3

Ce traitement fait appel à l'algorithme VERIFIER_CONDITION (LS_PR, LD) oùla partie condition de la requête comprend des restrictions de valeurs sur la propriété « Annee». LD= {Livre.Annee>2005}

Dans ce cas, nous supposons que les valeurs sont de même type de données et nous ne traitons pas les conflits pouvant surgir d'un problème lié aux types de données. Les services Web pouvant y répondre sont :

- **Cas 3.1 :** Dans notre cas aucun service ne comprend une des propriétés recherchées et dont l'entrée correspond à « Annee ».

- **Cas 3.2 :** Le traitement de ce cas fait appel à la fonction TROUVE_SERVICE_ENTREE (Livre.Annee>2005) pour retrouver les services Web ayant des sorties correspondant à au moins une des propriétés recherchées et à la propriété « Annee ». L'exécution de la fonction retourne les services SW_2 et SW_6 (voir tableau 22).

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		UDDI1
		E_1 : Livre.ISBN		(SW_1: S_1, SW_2:E_1)	
				(SW_6:S_1, SW_2:E_1)	
				(SW_2:E_1//SW_3:E_1)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6:S_1, SW_2:E_1)	UDDI2
				(SW_6:S_1, SW_3:E_1)	
			S_2:Livre.Titre	(SW_6:S_2, SW_5:E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 21 : Exemple de traitement du cas 3.1

Le traitement du cas 3.2 retourne le résultat suivant

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 : S_1, SW_2: E_1)	UDDI1
				(SW_1 : S_1, SW_3: E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 : S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 : S_1, SW_3: E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2, SW_5 :E_2)	
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 22 : Exemple de traitement du cas 3.2

- **Cas 3.3 :** Les services Web ayant en entrée la propriété sur laquelle une restriction de valeur est exigée et pouvant être composé avec des services Web ayant des sorties correspondant à au moins une des propriétés recherchées. Une composition séquentielle doit être effectuée (cf. §3.4.1.1 règle 2).

Ainsi le résultat final du traitement du cas 3 est le suivant :

Services Web	Opérations			Composition	Registre UDDI
	Nom	Entrées	Sorties		
SW_1	Op_1 : Ecrit		S_1 : Livre.ISBN	(SW_1 :S_1, SW_2: E_1)	UDDI1
				(SW_1 :S_1, SW_3: E_1)	
			S_5 : Auteur.Nom		
SW_2	Op_2 :Edition_Livre		S_6 : Livre.Annee		
		E_1 : Livre.ISBN		(SW_1 :S_1, SW_2: E_1)	
				(SW_6 :S_1, SW_2 :E_1)	
				(SW_2 :E_1//SW_3 :E_1)	
SW_3	Op_2 : Livre		S_2: Livre.Titre	(SW_3 :S_2, SW_5 :E_2)	
			S_{10}: Livre_coût		
		E_1 : Livre.ISBN		(SW_2 :E_1//SW_3 :E_1)	
				(SW_6 :S_1, SW_3 :E_1)	
				(SW_1 :S_1, SW_3: E_1)	
SW_4	Op_2 : Livre		S_2: Livre. Titre	(SW_4 :S_2,SW_5 :E_2)	
	Op_1 : Auteur		S_5 :Auteur.Nom		
SW_5	Op_2 : Livre		S_5 :Auteur.Nom		UDDI3
			S_{10} :Livre.Coût		
		E_2: Livre.Titre		(SW_4 :S_2, SW_5 :E_2)	
				(SW_6 :S_2, SW_5 :E_2)	
				(SW_3 :S_2, SW_5 :E_2)	
SW_6	Op_2 : Ecrit		S_1 :Livre.ISBN	(SW_6 :S_1, SW_2 :E_1)	UDDI2
				(SW_6 :S_1, SW_3 :E_1)	
			S_2:Livre.Titre	(SW_6 :S_2, SW_5 :E_2)	
			S_5 :Auteur.Nom		
			S_6 :Livre.Annee		

Tableau 23 : Exemple de traitement du cas 3

5. Présentation des résultats

Le résultat final va interroger le registre UDDI pour pouvoir invoquer le service. Il comprend : Le nom du service, sa localisation, son fournisseur, les annotations des fonctionnalités de la liste de ses opérations, ses entrées et sorties et les noms de services avec lesquels une composition est nécessaire pour répondre à la requête.

6. Architecture fonctionnelle

Nous décrivons dans cette partie les diagrammes de cas d'utilisation UML représentant les fonctions du système.

Nous identifions les acteurs et les cas d'utilisations suivants :
Acteurs :

- Le demandeur de services.
- Le fournisseur de services.

Cas d'utilisation :

- Recherche de service.
- Traiter requête.
- Annoter un service.
- Publier un service.
- Enregistrer un service.

La figure 19 illusttre le diagramme global de cas d'utilisation

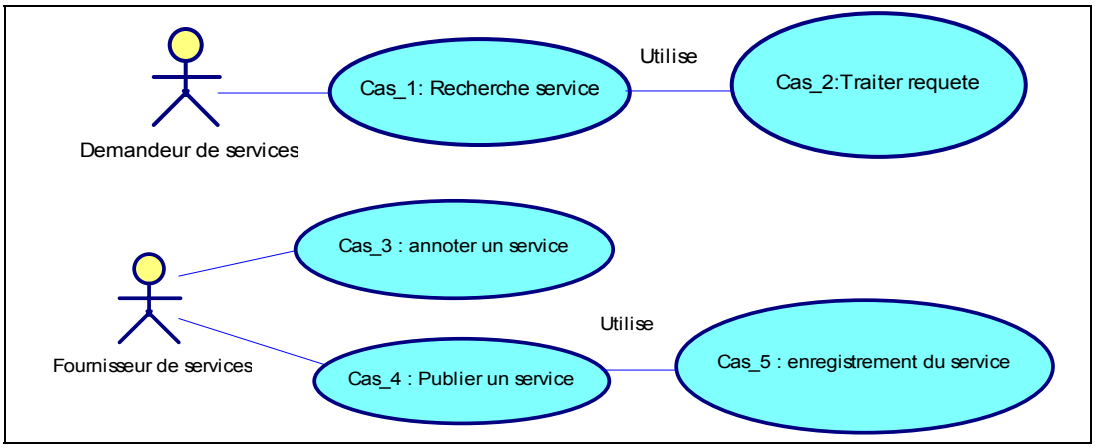


Figure 19 : Diagramme de cas d'utilisation

Le diagramme d'activité pour représenter le détail des cas d'utilisation pour une demande de services est prénsenté dans la figure 20.

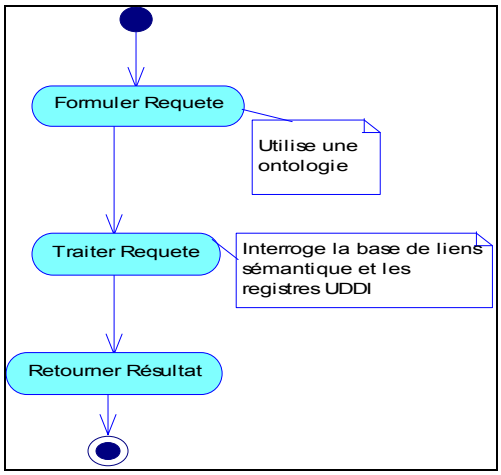


Figure 20 : Diagramme d'activité pour une demande de service

Le diagramme d'activité pour représenter le détail des cas d'utilisation pour fournir un service est prénseté en figure 21.

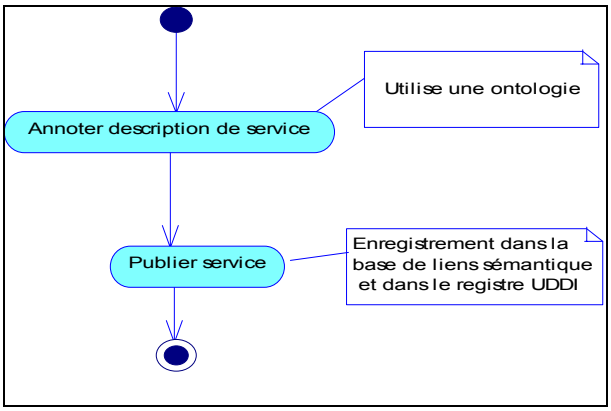


Figure 21 : Diagramme d'activité pour fournir un service

7. Interfaces

Le prototype que nous avons réalisé, implémente les cas d'utilisations relatifs au demandeur de service. Il permet à l'utilisateur de lancer une recherche simple à partir d'une requête formulée dans le vocabulaire de l'ontologie du domaine de l'édition.

Comme illustré dans la figure 22, l'utilisateur peut également indiquer les contraintes sur les champs choisis, en spécifiant le contenu et le critère de recherche. Une fois la requête formulée, elle peut être exécutée.

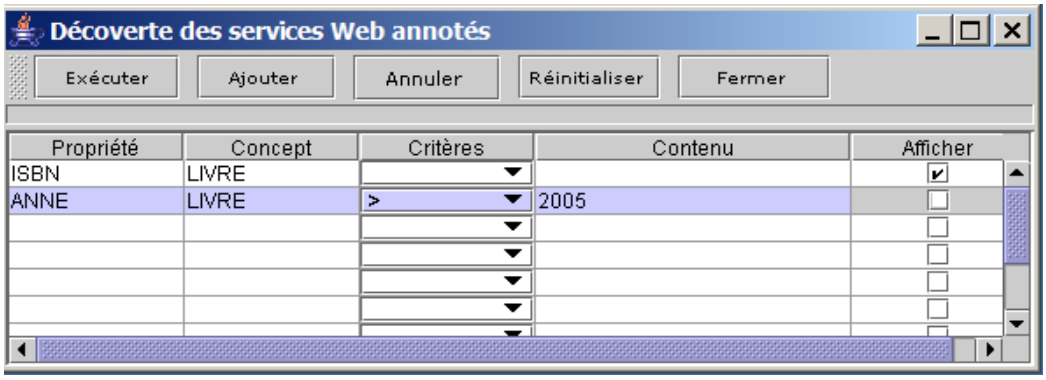


Figure 22 : Interface d'interrogation

Les résultats de la requête, sous forme XML, sont présentés dans la figure 23.

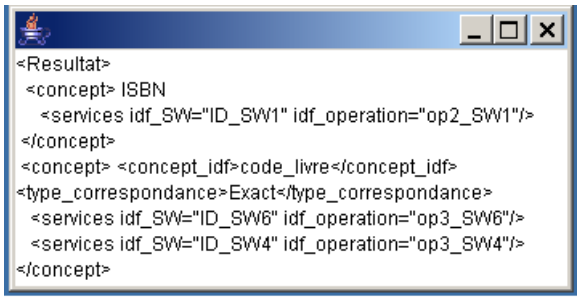


Figure 23 : Interface Résulat

Le programme d'interrogation de la base de liens sémantiques et du registre UDDI en XQUERY et un exemple de requête en XQUERY sont présentés en annexe E.

8. Conclusion

Ce chapitre nous a permis de présenter les outils utilisés pour l'implémentation de notre prototype, d'illustrer un exemple de traitement d'une requête qui prend en considération différents scénarii et de présenter l'architecture fonctionnelle et les interfaces du prototype.

CONCLUSION GENERALE

Le Web a ouvert de nouveaux horizons quant à la manière d'exposer des services aux clients. Les services Web proposent une façon de développer des applications/systèmes distribuées en les rendant interopérables quels que soient leurs plates-formes ou leurs langages. L'interaction entre composants des services Web s'effectue à travers trois opérations : (i) l*a publication* de la description de services, (ii) la *recherche et la découverte* de la description du service et (iii) l'*invocation* du service basée sur sa description.

La découverte sémantique des services Web a motivé de nombreux travaux. Un premier standard, SAWSDL, pour la description sémantique de services Web, apparu récemment, permet d'annoter sémantiquement la description des services Web et les schémas XML.

Notre travail a consisté à annoter les services Web et à proposer un langage d'expression des requêtes et d'exploration des services pour répondre aux mieux aux besoins du client. Pour atteindre cet objectif, nous avons étudié quelques approches de description sémantiques des services Web, l'annotation sémantique des données de façon générale et celle des services Web de façon particulière puis nous nous sommes intéressés aux langages d'interrogation et aux approches de découverte des services Web.

Le résultat de cette étude nous a permis de proposer une approche de découverte sémantique des services Web [71] qui diffère des solutions existantes. Elle s'appuie sur la spécification SAWSDL pour l'annotation sémantique des fonctionnalités des services Web. Nous avons effectué un stockage de ces annotations dans un registre appelé *base de liens sémantiques*. Cette base représente le médiateur entre composants de la requête et fonctionnalités de services Web et permet de minimiser le temps de recherche lors de la phase de découverte. Aussi, nous avons défini des règles de composition entre services et présenté l'algorithme de mise en correspondance entre les composants de la requête, définie de façon unifiée et selon un vocabulaire commun en utilisant le principe d'une ontologie exprimée en OWL, et les fonctionnalités de services Web. L'utilisation du langage d'interrogation des données semi-structurées XQUERY, sur la *base de liens sémantique*s et sur le registre UDDI, nous a permis de retourner aussi bien les fonctionnalités du service Web que les informations relatives à sa localisation et à son invocation sous la forme d'un document XML. Ce résultat peut servir comme point d'entrée pour une autre interrogation.

Pour tester notre solution nous avons sélectionné quelques scénarii regroupant l'ensemble des cas définis. L'accès à la *base de liens sémantique* et aux exemples de registres UDDI est rendu possible grâce à l'utilisation de l'API Nux dans des classes Java.

Il reste, cependant, à développer plusieurs points, en particulier, l'utilisation d'autres approches pour la composition des services, l'intégration des techniques de mise en correspondance entre schémas, la maintenance de la *base de liens sémantique*s et l'assistance à la découverte des annotations des services.

Nous avons suivi une approche pragmatique pour montrer la faisabilité de nos propositions mais nous sommes conscients que d'un point de vue plus fondamental, il faudrait aussi valider les algorithmes proposés et leur implantation en étudiant plus précisément leurs propriétés dont la correction et la complexité.

BIBLIOGRAPHIE

[1] E. Desmontlis et C. Jacquin**, «Annotations sur le Web: notes de lecture »**, *Actes des journées scientifiques CNRS Web Sémantique*, Paris, 10-11 octobre 2002.

[2] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, et M. Goranov, «**KIM – Semantic Annotation Platform»**, *2nd International Semantic Web Conférence (ISWC2003),* Springer-Verlag Berlin Heidelberg LNAI Vol. 2870, Florida USA, *20-23 Octobre* 2003, P. 484-499.

[3] S. Bechhofer, L. Carr, C.A. Goble, S. Kampa et T. Miles-Board «**The Semantics of Semantic Annotation»,** *First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, LNCS Vol. 2519 Springer-Verlag, Irvine-California USA, 2002, P. 1151-1167.

[4] A. kiryakov, B. Popov, D. Ognyanoff, D. Manov, A. Kirilov, et M. Goranov, «**Semantic annotation, indexing and Retrieval»**, *J. Web Sem: Science, Services and Agents on the World Wide Web ,* Elsevier, 2004, P. 49-79.

[5] J. Kopecký, T. Vitvar, C. Bournez et J. Farrell **«SAWSDL: Semantic Annotations for WSDL and XML Schema»** IEEE Internet Computing, 2007, P.60-67.

[6] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, et F. Ciravegna, «**MnM: Ontology Driven, Semi-automatic and Automatic Support for Semantic Markup».** In Proceedings of *EKAW 2002*, Springer LNCS 2473, ISBN 3-540-44268-5,Siguenza, Spain, October 1-4, 2002, P. 379–391.

[7] M. R. Koivunen, R. Swick, et E.Prud'hommeaux, «**Annotea Shared Bookmarks »**, *Proc of the KCAP 2003 workshop on knowledge markup & semantic annotation*, Sanibel Florida - USA, 25-26 octobre 2003.

[8] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, et R. Swick, «**Annotea: an open RDF infrastructure for shared Web annotations»**, *World Wide Web 10th Internationnal conference*, Hong Kong, 2001, P. 623-632.

[9] C. C. MARSHALL. «**Toward an ecology of hypertext annotation»**, *Proceedings of the Ninth ACM Conference on Hypertext*, ACM, Pittsburgh - USA, 20-24 juin 1998, P. 40-49.

[10] L. Carr, W. Hall, S. Bechhofer et C. A. Goble, **«Conceptual linking: ontology-based open hypermedia»**, *WWW 2001,* ACM, ISBN 1-58113-348-0, Hong Kong China, 1-5 Mai 2001, P. 334-342.

[11] R. M. Heck, S. M. Luebke, et C. H. Obermark, « **A Survey of Web Annotation Systems»,** Rapport technique Dep. Of Mathematics and Computer Sience, Grinnell College, USA, 1999. Disponible sur «http://www.math.grin.edu/~rebelsky/Blazers/Annotations/Summer1999/Papers/survey_paper.html».

[12] F. Azouaou. « **Modèles et outils d'annotations pour une mémoire personnelle de l'enseignant»**. Thèse de doctorat en informatique, Université Joseph Fourier de Grenoble, soutenu en octobre 2006 (255 pages).

[13] Y . Prié, Y. et S. Garlatti, « **Méta-données et annotations dans le Web sémantique** ». Hors Série 2004 - Web Sémantique. *Revue en Sciences du Traitement de l'Information, I3 (Information -*

Interaction – Intelligence), Cépaduès, Toulouse France, 2004, P. 45-68.

[14] G. Pérez, M. F. Lopez, et O. Corch. « **Ontological Engineering** », chapitre1, chapitre 5: ontology tools, Springer. London: Britich library, ISBN 1-85233-551-3, 2004, (403 pages).

[15] Hai-tao Zheng, Bo-Yeong Kang, Sang-Ok Koo, Hee-Chul Choi, Kwang-Sub Kim, et Hong-Gee Kim, « **A Semantic Annotation Tool to Extract Instances from Korean Web Documents** », *Proceedings of 1st Semantic Authoring and Annotation Workshop (SAAW2006) of 5th International Semantic Web Conference (ISWC)* ISSN 1613-0073, CEUR-WS Vol 209, Athens GA- USA, 5 - 09 novembre 2006.

[16] P. Rajasekaran, J. A. Miller, K. Verma, et A. P. Sheth: « **Enhancing Web Services Description and Discovery to Facilitate Composition** ». *SWSWPC2004*, Springer-Verlag Heidelberg, SAN Diego CA USA, 2004, P. 55-68.

[17] **LSDIS, METEOR-S: Semantic Web Services and Processes.**
Disponible sur : « http://swp.semanticWeb.org ».

[18] M. Erdmann, A. Maedche, H.-P. Schnurr, et Steffen Staab. « **From manual to semi-automatic semantic annotation: About ontology-based text annotation tools**». *Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content,* In P. Buitelaar & K. Hasida (eds), Luxembourg, Août 2000. P. 79-86.

[19] Semantic Web Annotation and Authoring. Annotation Tools [en ligne].
Disponible sur : <http://annotation.semanticWeb.org/annotationtool_view> (dernière consultation le 8 avril 2007).

[20] A.Patil, S. Oundhakar, A. Sheth, et K. Verma. « **METEOR-S Web service Annotation Framework ».** In Proceeding of the 13th International World Wide Web Conference (WWW2004), New York NY USA, May 17-22, 2004, P. 553–562.

[21] M. Paolucci, T. Kawamura, T. R. Payne, et K. Sycara**.** « **Semantic matching of Web services capabilities ».** *First International antic Web Conference (ISWSemC2002),* Springer-Verlag volume 2342 / 2002 of LNCS, Sardinia - Italy, June 9-12, 2002.

[22] S. Rampacek, « **Sémantique, interactions et langages de description des services Web complexes**», Thèse de doctorat en informatique, Université de Reims Champagne-Ardenne, soutenue le 10 novembre 2006.

[23] A. Ankolekar, M. H. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, K. P. Sycara, et H. Zeng, "**DAML-S: Semantic Markup for Web Services**", *SWWS*, Stanford University, California USA, July 30 - August 1, 2001, P. 411-430.

[24] L. H. Reeve, et H. Han, « **Survey of semantic annotation ».** *SAC 2005*, ACM Press, ISBN 1-58113-964-0, Santa-Fe NY USA, March 13-17, 2005, P. 1634-1638

[25] L. Denoue, « **De la création à la capitalisation des annotations dans un espace personnel d'informations** », thèse de doctorat en informatique, Université de Savoie, France, octobre 2000.

[26] S. Handschuh, et S. Staab, «**CREAM: CREAting Metadata for the Semantic Web»,** *Computer Networks 42*(5), Elsevier ISSN: 1389-1286, New York NY USA, 5 August 2003, P. 579-598.

[27] R. Akkiraju , J. Farell , J.A. Miller, M. Nagarajan, A. Sheth et K. Verma, «**Web Service**

Semantics - WSDL-S», UGA-IBM Technical Note, Avril 2005.
Disponible sur <http://www.w3.org/2005/04/FSWS/Submissions/17/WSDL-S.htm#SVSM03>. (Dernière consultation mai 2007).

[28] P. Kellert et F. Toumani, « **Les Web services sémantiques»**, Hors Série 2004 *Revue en Sciences du Traitement de l'Information I3 (Information - Interaction – Intelligence*), Cépaduès, Toulouse France, 13 octobre 2003, P. 93-110.

[29] J. Domingue, L. Cabral, H. Hakimpour, D. Sell, et E. Motta. « **IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services»**, *Proceedings of the Workshop on WSMO Implementations (WIW 2004), Frankfurt Germany* Vol. 113, CEUR Workshop Proceedings ISSN 1613-0073, 29-30 September 2004.
Disponible sur : « http://CEUR-WS.org/Vol-113/paper3.pdf »

[30] K. Sivashanmugam, K. Verna, A.P Sheth, et J. A. Miller, « **Adding semantics to Web services Satandards»**, *ICWS 2003,* CSREA Press, Las Vegas USA, 2003, P. 395-401.

[31] D. D. Roman, U. Keller, H. Lausen, J. D. Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, et D. Fensel, « **Web Service Modeling Ontology»**, *Applied Ontology* 1(1), IOS Press, Amsterdam Pays Bas, 2005, P. 77-106.

[32] P. Shvaiko, et J. Euzenat, « **A survey of schema-based matching approaches»** , *Journal on data semantics 4,* LNCS 3730 Springer-Verlag ISBN 3-540-31001-0, INIST-CNRS, 2005, P. 146-171.

[33] E. Rahm, et P. A. Bernstein, «**A survey of approaches to automatic schema matching** ». *VLDB J. 10(4)*, Springer-Verlag New York, Secaucus NJ USA, 2001, P. 334-350.

[34] W3C. Semantic Annotations for WSDL and XML Schema, W3c recommendation 28 August 2007.
Disponible sur **:** « http://www.w3.org/TR/2007/REC-sawsdl-20070828/**».** (Dernière consultation novembre 2007).

[35] J. Condack, D. Schwabe, « **Swell - Annotating and Searching Semantic Web Services** », *LA-WEB'05*, IEEE Computer Society, Buenos Aires – Argentina, 2005, P. 102-105.

[36] J. Condac, et D. Schwabe, « **Swell, SwellOnt and SwellQL -** a **Software Engineering Environment for Searching Semantic Web Services»**, Rapport technique, ISSN: 0103-9741, Août 2005.

[37] W3C. The Extensible Stylesheet Language Family (XSL)
Disponible sur : « http://www.w3.org/Style/XSL/ » (Dernière consultation novembre 2007).

[38] S. Bechhofer, R.Volz, et P. Lord, **« Cooking the Semantic Web with the OWL API»,** *ISWC 2003*, LNCS ISSN 0302-9743, Sanibel Island Florida, 21-23 october 2003, P. 659-674.

[39] T. Berners-Lee, J. Hendler, et O. Lassila, « **The semantic Web »,** *Scientific American,* May 2001, P. 28 -37.

[40] **Ontologymatching**.
Disponible sur « http://www.ontologymatching.org/» (Dernière consultation juillet 2007).

[41] J. Madhavan, P.A Berrnstein, et E. Rahm, « **Generic schema Matching with Cupid»***, VLDB 2001*, Morgan Kaufmann, Roma - Italy, 2001, P.49-58.

[42] R. Thomas Fielding « **Architectural Styles and the Design of Network-based Software Architectures »,** thèse de doctorat en Philosophie, Université de Californie Irvine USA, soutenu en 2000.
Disponible sur : « http://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm »

[43] W3c. Le W3C finalise un standard essentiel pour les services Web. Disponible sur http://www.w3.org/2007/06/wsdl20-pressrelease.html.fr.

[44] M. P. Papazoglou, P. Traverso, S. Dustdar, et F. Leymann, « **Service-Oriented Computing: State of the Art and Research Challenges»**, *IEEE Computer* ISSN:0018-9162 *Vol 40(11)*, IEEE Computer Society Press, Los Alamitos CA, USA, Nov 2007, P. 38-45.

[45] The OWL Services Coalition, **OWL-S: Semantic Markup for Web Services»**, 2003.
Disponible sur : « http://www.w3.org/Submission/OWL-S »

[46] N. Srinivasan, M. Paolucci, et K. P. Sycara, « **An Efficient Algorithm for OWL-S Based Semantic Search in UDDI »**, *SWSWPC 2004*, LNCS 3387 Springer, ISBN 3-540-24328-3 ,San Diego CA USA, 6 juillet 2004, P. 96-110.

[47] OASIS, UDDI Version 3.0, UDDI Spec Technical Committee Specification, 19 juillet 2002.
Disponible sur : « http://uddi.org/pubs/uddi_v3_features.htm »

[48] S. Aggarwal, et R. Studer, « **Automatic Matchmaking of Web Services»,** *ICWS 2006,* IEEE Computer Society 2006, ISBN 0-7695-2669-1, Chicago, Illinois, USA, 18-22 September 2006, P. 45-54.

[49] K. P SYCAR, S.WIDOFF, M. KLUSCH, J. LU, « **Larks : Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace»**, Autonomous Agents and Multi- Agent Systems, vol. 5, no 2, Kluwer Academic, Bologna- Italy, 2002, P. 173-203.

[50] Y. Sam, O. Boucelma, et M. S Hacid, « **Dynamic Web Services Personalization»**, *CEC/EEE,* IEEE Computer Society, San Francisco - USA, 2006, P. 564-570.

[51] M. Papazoglou, M. Aiello, M. Pistore, et J. Yang, « **XSRL: An XML Web-service request language»**. *Technical Report*, DIT-02-0079, Université de Trento, 2002.

[52] A. Lazovik, M. Aiello, et M. P. Papazoglou, «**Planning and monitoring the execution of Web service requests»,** *Int. J. on Digital Libraries 6(3*), Springer-Verlag, INIST-CNRS France, Juin 2006, P 235-246.

[53] J. Luo, B. Montrose, A. Kim, A. khashobih, et M. Kang, «**Adding OWL-S support to the Existing UDDI Infrastructure»**, *ICWS'06*, IEEE Computer Society, Washington - USA, 2006, P. 153-162.

[54] Lei Ye, et B. Zhang, «**Web Service Discovery Based on Functional Semantics »,** *SKG 2006,* IEEE Computer Society, Guilin – Guangxi - China, 2006*,* P. 57 - 57.

[55] J. Colgrave, R. Akkiraju, et R. Goodwin, « **External Matching in UDDI »**, *ICWS*'04, IEEE Computer Society 2004, San Diego – CA USA, 6-9 juin 2004, P. 226-233.

[56] J. Thierry-Mieg et R. Durbin, «**Syntactic definitions for the ACeDB data base manager »,** *Technical report*, MRC-LMB xx.92, MRC Laboratory for Molecular Biology, Cambridge,1992.

[57] S. S. Chawathe, et al «**The TSIMMIS Project: Integration of Heterogeneous Information Sources»**. *IPSJ 1994,* In Proceedings of the 10th Meeting of the Information Processing Society of

Japan , Tokyo Japan, Octobre 1994, P. 7-18.

[58] N. Shadbolt, E. Motta, et A.Rouge, « **Constructing Knowledge-Based Systems»,** IEEE Software ISSN: 0740-7459, IEEE Computer Society Press , Novembre 1993, los Alamitos CA USA, P. 34-38.

[59] W3C. **WSDL 2.0: what's new?** XML Conference 2004 , Washington, DC, USA, 2004, Disponible sur : « http://www.w3.org/2004/Talks/1117-hh-wsdl20/ »

[60] A. Lazovik, « **Interacting With Service Compositions»**, thèse de doctorat : International Doctorate School in Information and Communication Technologies DIT, Université de Trento Italy, soutenu en décembre 2006.

[61] W3c. *XQUERY 1.0 : An XML Query Language*, W3c Recommendation du 23 janvier 2007.
Disponible sur : « http://www.w3.org/TR/2007/REC-xquery-20070123/».

[62] W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation du 26 Juin 2007.
Disponible sur : « http://www.w3.org/TR/2007/REC-wsdl20-20070626/».

[63] C. C. Marshall, «**Toward an ecology of hypertext annotation»**. *ACM Hypertext*, ACM, Pittsburgh - PA USA, 20-24 juin 1998, P. 40-49.

[64] Woden4SAWSDL Home. 15 mai 2007
Disponible sur «http://lsdis.cs.uga.edu/projects/meteor-s/opensource/woden4sawsdl/» (dernière consultation mars 2008)

[65] Wsmo4j Home.
Disponible sur «http://wsmo4j.sourceforge.net/» (dernière consultation septembre 2007)

[66] SAWSDL4J home. 26 juin 2007
Disponible sur « http://knoesis.wright.edu/opensource/sawsdl4j/ » (dernière consultation mars 2008).

[67] Semantic Tools for Web Services.
Disponible sur «http://www.alphaworks.ibm.com/tech/wssem ».

[68] A. Boudries, N. Boudjlida, et H. Talantikite, «**Annotation Sémantique et Web services »**, *SIIE 2008*, IHE éditions vol 2*,* Hammamet Tunisie, 14-16 février 2008, P.589-601, ISBN 9978-9973-868-20-6.

[69] S. V. Hashemian, et F. Mavaddat: « **A Graph-Based Framework for Composition of Stateless Web Services** ». ECOWS 2006, IEEE Computer Society 2006, ISBN 0-7695-2737-X , Zürich, Switzerland, 4-6 Décembre 2006, P.75-86.

[70] R. Bova, H.Y. Paik, S. Hassas, S. Benbernou et B. Benatallah**, « WS-Advisor: A Task Memory for Service Composition Frameworks »**, *ICCCN 2007*, IEEE, , ISBN 978-1-4244-1251, Hawaii USA, 13-16 août 2007, P. 535-540.

[71] A. Benna, N. Boudjlida et H. Talantikite, **"SAWSDL, MEDIATION and XQUERY for web services"** , In proceedings of the 8th international conference on New Technologies in Distributed Systems: Notere 2008, volume 1, ACM edition, ISBN 978-1-59593-937-1, Lyon France, 23-27 juin , 2008, P. 12-21.

Annexe A : Exemples d'outils d'annotation sémantique

L'annexe 1 présente les outils d'annotation sémantique les plus référencés et une étude comparative entre ces outils.

1. Exemple d'outils d'annotation sémantique

Les outils d'annotations sémantiques basés sur les ontologies [18,19] sont apparus dans le contexte des applications du Web sémantique. Ils sont souvent intégrés à d'autres suites d'outils ou interagissent avec d'autres outils de développement d'ontologie. Ils visent à améliorer l'appréhension des documents HTML ainsi que la communication et l'interopérabilité sur le Web.

Nous nous concentrerons dans cette partie à décrire l'architecture et le fonctionnement de quelques outils d'annotations sémantiques, les plus référencés actuellement. Ces systèmes sont étudiés suivant les critères suivants :

- *Interprétation* : les annotations peuvent être destinées à être traitée uniquement par l'humain ou l'agent logiciel ou au lecteur humain et à l'agent logiciel.
- *Type d'annotation* : indique si l'annotation est effectuée de façon manuelle, semi-automatique ou automatique.
  - Annotation manuelle : les annotations sont effectuées manuellement par un utilisateur/expert, cette technique étant une source d'erreurs et d'un coût élevé.
  - Annotation semi-automatique : elle repose sur un système supervisé d'extraction d'information qui génère des règles d'extraction à partir d'un corpus de documents fourni en entrée. Elle nécessite l'intervention de l'utilisateur dans le processus d'annotation. Cette technique permet de générer des instances de concept mais n'exploite pas la structure globale du document ; les relations entre les instances ne sont donc pas extraites.
  - Annotation automatique : des systèmes non-supervisés visent à sortir totalement l'humain de la boucle d'annotation en exploitant la redondance de l'information sur le Web.
- *Type de ressources annotées :* Précise le type de document à annoter (HTML, courrier électronique, XML, ...).
- *Langage de description des annotations :* Indique le langage de description des annotations (RDF, Text, …)
- *Localisation de l'annotation :* Précise ce qui est annoté dans le document (tout le document, une région sélectionnée, une position,…)
- *Stockage des annotations :* Spécifie si les annotations sont stockées dans le document ou séparé du document.
- *Langage d'ontologie* : Représente le langage de description de l'ontologie (DAML, OIL, OWL…)
- *Destination des annotations* : Peut être privée ou publique. Certaines annotations sont destinées à une utilisation privée, d'autres sont destinées à un nombre plus large d'utilisateurs.
- *Architecture et Technologies* : Décrivent l'architecture et les technologies et utilisées dans les systèmes d'annotation.

- *Extraction et apprentissage :* Indique l'outil d'extraction, le format de la base de connaissance ainsi que les méthodes utilisées pour l'apprentissage.

1.1 COHSE - Conceptual Open Hypermedia Services Environment

COHSE[52] [3, 10] est à l'origine un projet commun entre Information Management Group (IMG) de Manchester et le groupe Intelligence Agents and Multimédia (IAM) de Southampton. C'est un système complexe reposant sur un enchaînement de fonctionnalités issues de différents outils. Il lie différentes pages annotées avec le même concept et permet le passage de l'une à l'autre. Un lien hypertexte est créé grâce à l'index de pages HTML. Les meta données sont utilisées comme un mecanisme d'indexation

Il permet aussi la visualisation de l'ontologie utilisée et les pages annotées reliées parfois entre elles par leur index.

L'objectif de COHSE est de rassembler une architecture hypermédia ouverte avec des services ontologiques afin de fournir une architecture pour le Web sémantique [3].

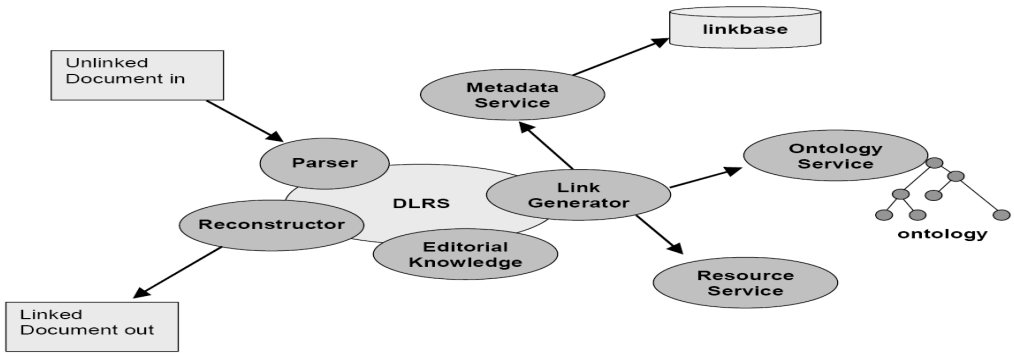


Figure 24 : Architecture de base du système COHSE [10]

Comme illustré dans la figure 24, COHSE consiste en un ensemble de quatre composants interconnectés [10].

- Le *Service d'Annotation* permet d'annoter le document : Tout le document ou une partie est traitée avec une ontologie spécifique, où une région particulière du document correspond à un terme particulier de l'ontologie. Ce service stocke les annotations de page Web en DAML+ OIL.
- Le *Service Ontologie* stocke les ontologies DAML+ OIL et met en correspondance les termes en langage naturel avec ceux de l'ontologie. L'application interagit avec ce service en utilisant une API bien définie : OWL-API.[53]
- Le *Service Ressource* agit comme un proxy et met en correspondance les concepts de pages Web et les ressources. Il est questionné pour rechercher les destinations possibles pour le lien : les ressources qui contiennent des exemples de ce concept.

[52] http://cohse.semanticWeb.org
[53] http://owl.man.ac.uk/api.shtml

- Le *Service résolution de liens distribué* est composé d'un agent DLRS (Distributed Link Resolution Service) qui enrichi des documents avec des liens basés sur le contenu sémantique de ces documents. Pour cela il s'appuie sur le S*ervice Ontologie* et le S*ervice Annotations*. Le *Générateur de liens* analyse le document en recherchant les liens dans ce document et en examinant les annotations associées au document.

Le système utilise un navigateur basé sur l'outil Mozilla Annotateur appelé aussi COHSE Concept Annotator ou un proxy par lesquels toutes les demandes HTTP sont conduites.

Lorsqu'une page Web est ouverte dans le navigateur, l'utilisateur sélectionne le texte et le concept de l'ontologie qui lui est approprié et ajoute un commentaire en langage naturel à l'annotation. Cette dernière est envoyée au *Service Annotation* pour la stocker.

COHSE permet de créer des annotations sur les concepts qui sont décrits dans l'ontologie, mais pas sur les valeurs des attributs ou les relations d'instances.

1.2 Annotea

Annotea [7,8], un projet LEAD[54] (Live Early Adoption and Demonstration) du W3C, a développé un environnement d'annotation collaborative qui supporte des commentaires textuels sur des pages Web. Il a utilisé un modèle d'annotation basé sur le schéma RDF, et accessible par un serveur HTTP, pour décrire les annotations. Un des buts de ce projet a été de réutiliser autant que possible les technologies existantes du W3C.

L'infrastructure d'Annotea offre une flexibilité, un cadre et un environnement facile pour l'extension des capacités des annotations à d'autres applications [7]. Annotea s'appuie au niveau serveur sur Apache, MySQL et Perl Script et au niveau client sur Amaya[55], Zannot[56], etc. XPointer et XLink sont utilisés pour associer les annotations avec différentes parties des documents.

Comme illustré dans la figure 25, les annotations sont externes aux documents et peuvent être stockées dans un ou plusieurs serveurs d'annotations.

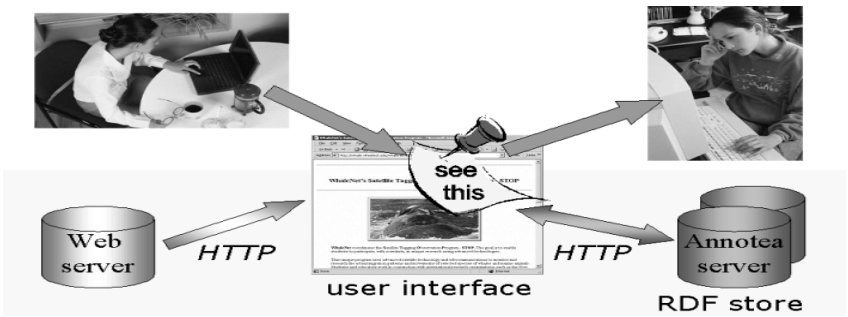


Figure 25 : Architecture d'Annotea [7]

Des utilisateurs, connectés à un serveur d'annotations peuvent ajouter, modifier et consulter des annotations.

[54] http://www.w3.org/2001/Annotea/
[55] Amaya est un éditeur Web voir (http://www.w3.org/Amaya/)
[56] *ZAnnot est un client Open-Source, plug-in du navigateur Mozilla-Firefox. Il permet la saisie et l'indexation de l'annotation d'une page Web selon les recommandations Annotea.*

L'utilisateur peut choisir d'annoter tout le document, une position ou la courante sélection.

Les annotations sont écrites en RDF/XML et peuvent être de type commentaire, requête ou correction et sont composées de méta données (Date de création, Nom auteur, URI,..) tel que Dublin Core[57] et du corps d'annotations représentant leur contenu.

Lorsqu'une annotation est publiée, elle devient partagée.

Quand l'utilisateur consulte un document, le navigateur interroge chaque serveur d'annotation, via la méthode HTTP GET, pour récupérer les métadonnées des annotations associées aux URI du document. Pour chaque liste d'annotation reçue, le navigateur analyse les métadonnées de chaque annotation et résout XPOINTER. Le corps de l'annotation est récupéré uniquement si l'utilisateur le souhaite. Ceci permet de réduire les données à transmettre et les ressources consommées

Lorsqu'un document est édité certaines de ces annotations peuvent devenir orphelines. Dans ce cas l'outil Amaya avertit l'utilisateur et rend l'annotation orpheline visible à l'utilisateur.

Dans Annotea, un langage nommé ALGAE[58] a été défini pour permettre la personnalisation de la requête utilisateur. Ce langage permet à l'utilisateur de définir sa propre requête.

1.3 KIM

KIM[59][2, 4], développé par le laboratoire Ontotext[60] du Groupe Sirma, fournit des services et une infrastructure de gestion de l'information et de la connaissance pour l'annotation sémantique automatique, l'indexation et la recherche du contenu (structuré et non-structuré). KIM permet aussi un peuplement de l'ontologie. Il se base sur une analyse de texte et identifie les références aux entités (comme une personne, organisation, date,…), puis met en correspondance la référence avec une entité connue ayant une URI unique et une description dans la base de connaissance. Une nouvelle URI et une description de l'entité sont automatiquement générées. La référence au document est annotée avec l'URI de l'entité. Ces annotations sont utilisées par la suite pour l'indexation, la recherche, la visualisation et les hyperliens de document.

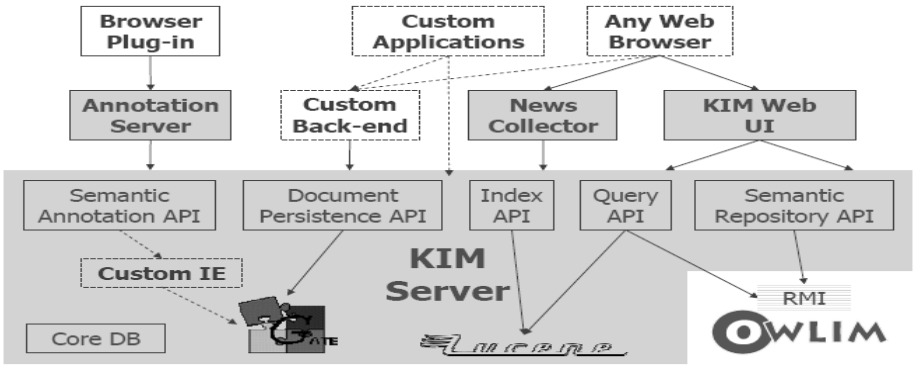


Figure 26 : Architecture de KIM [2]

Comme illustré dans la figure 26, la plateforme de KIM est composée des éléments suivants :

[57] http://dublincore.org
[58] ALGAE est un langage de requête pour RDF voir (http://www.w3.org/2004/05/06-Algae/)
[59] http://www.ontotext.com/kim/
[60] Laboratory for scientific research and development of technologies for knowledge processing and representation (KR) and linguistics (CL/NLP).

- Le Serveur KIM utilise les technologies GATE[61], Lucene[62] et Sesame[63]. Il comprend les API d'accès à distance suivantes :
  - *Semantic annotation API* permet l'annotation des documents en respectant l'ontologie KIM et la base de connaissance. Elle fournit également une infrastructure pour la gestion du contenu.
  - *Document Persistence API* permet la sauvegarde et la récupération des documents et les annotations qui leurs sont associées.
  - *Index API* basée sur Lucene, elle permet l'indexation en respectant des Entités nommées.
  - *Query API* permet la recherche par mot clé, entités combinées, chemin entité et d'autres méthodes d'accès à l'ontologie.
  - *Semantic Repository API* permet la gestion et l'accès à la base des connaissances à travers les méthodes d'accès et infrastructure RDF.
- Les ontologies PROTON[64], KIMLO (KIM Lexical Ontology) et KIMSO (KIM System Ontology)
- Une base de connaissance, KIM World KB, pré-peuplée avec plus 200000 entités.
- Des FRONT-END comprenant un serveur Web et une interface utilisateur Web
  - Plug- in navigateur pour Internet explorer qui fournit l'annotation sémantique des entités nommées (Named Entities[65]), pour des contenus du Web en ligne structurés et non structurés. Les annotations font référence au niveau supérieur de l'ontologie et à la base de connaissance, contenant les instances du monde réel. Récemment, le Plug-in KIM supporte OWL lite et RDF.
  - Interface utilisateur Web de KIM : Offre plusieurs méthodes d'accès Web aux caractéristiques de KIM.

1.4 Mnm

MnM [6], développé par l'institut knowledge Description Media Open University, de United Kingdom, dans le contexte de l'AKT[66](Collaboration de Recherche Interdisciplinaire), est une application Java autonome qui intègre un navigateur Web et un outil de visualisation d'ontologie. Il permet une annotation manuelle, semi-automatique et automatique des documents. Les ontologies peuvent être décrites au format RDF, DAML+OIL ou OCML [58].

MnM utilise des moteurs d'extraction d'information (Amiclare[67],..) pour détecter les instances du concept apparaissant dans le document. Ces moteurs exploitent du texte et des documents HTML annotés.

Les annotations crées avec MnM peuvent être utilisées pour peupler les ontologies existantes ou être attachées aux documents originaux. Elles peuvent être utilisées pour différents environnements.

MnM stocke les instances, dans une base de connaissance, selon différents format : OCML, RDF, DAML+OIL et XML.

L'étiquetage sémantique des documents se fait selon une base de connaissance à peupler ou à charger. Le peuplement de la base de connaissance se fait semi-

[61] Général Architecture for Text Engineering, plateforme d'extraction de l'information http://gate.ac.uc
[62] Lucene : Moteur de recherche d'information textuelle Open source
[63] Sesame : registre RDF
[64] http://proton.semanticWeb.org
[65] Comme une personne, organisations, locations, monaie, etc
[66] http://kmi.open.ac.uk/projects/akt/MnM/
[67] Amiclare : Outil d'extraction textuelle de l'information http://nlp.shef.ac.uk/amilcare approche par induction algorithme LP

automatiquement au début, puis selon l'apprentissage effectué il est possible d'aller jusqu'à un peuplement complet.

L'utilisateur peut choisir d'éditer ou non, les annotations générées par le module d'extraction de l'information.

1.5 Ontomat Annotizer

OntoMat-Annotizer [68] , a été développé par l'Institut AIFB à l'université de Karlsruhe, est un outil interactif d'annotation de page Web. Il permet la création manuelle d'annotations et d'ontologie OWL. Il permet aussi de télécharger des ontologies OWL.

OntoMat-Annotizer est une application Java autonome avec une interface plug-in pour l'extension. Elle inclut un browser d'ontologie pour le téléchargement et l'exploration des concepts de l'ontologie et un browser HTML pour afficher les documents et leurs annotations.

L'utilisateur identifie des instances des concepts des ontologies dans le texte et annote manuellement les parties de texte identifiées. Les annotations créées sont stockées en OWL dans des fichiers séparés ou dans les documents HTML annotés

OntoMat-annotizer est l'implémentation de S-CREAM [26]. Il intègre un extracteur d'information Amilcare pour associer automatiquement les concepts aux fragments du texte.

1.6 SHOE Knowledge Annotator

SHOE Knowledge Annotator[69], a été développé par le groupe Parallel Understanding Systems au department informatique de l'université de Maryland. C'est un outil de création d'annotation manuel, avec le langage SHOE, dans des pages HTML.

Il est disponible en application Java autonome et en applet Java. Il permet de créer les instances de concepts, les valeurs des attributs et les instances des relations.

Les annotations sont construites dans le document HTML original et les documents HTML ne sont pas visualisés comme dans les navigateurs du Web.

SHOE Knowledge Annotator est à la base de création de SMORE[70], un outil d'annotation, exploitant des ontologies OWL.

1.7 YAWAS

YAWAS [25] (Yet Another Web Annotation System), permet l'annotation de page Web et les messages du dans Outlook. YAWAS est implanté en Java et JavaScript et exploite les technologies DOM[71] niveau 2 et le HTML Dynamique (voir Figure 27).

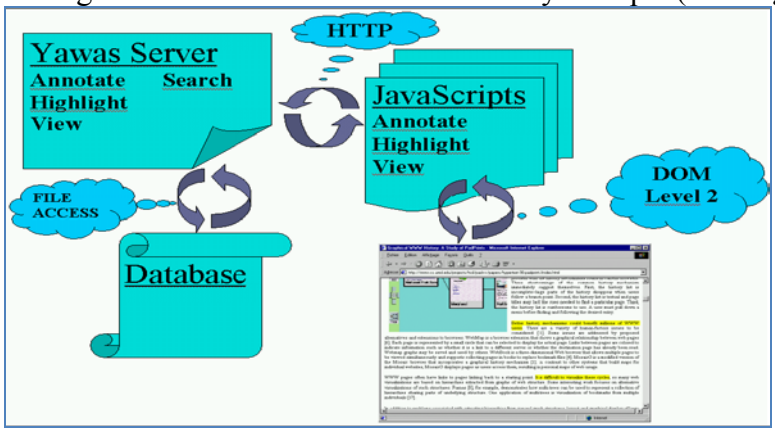


Figure 27 : Architecture générale de Yawas [25]

[68] http://annotation.semanticWeb.org/ontomat/index.html
[69] http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html
[70] SMORE http://annotation.semanticWeb.org/tools/
[71] Document Object Model http://www.w3.org/DOM/

Les principales fonctions de Yawas sont : l'ajout d'une nouvelle annotation, la visualisation des annotations, la recherche d'annotations et enfin le partage des annotations.

Pour créer une annotation, l'utilisateur surligne à la souris le texte à annoter et sélectionne l'option "Annoter" du menu contextuel. Yawas propose alors à l'utilisateur de remplir un formulaire où tous les attributs sont optionnels. Le titre du document, son URL et le texte surligné sont remplis automatiquement. Chaque attribut est personnalisable grâce à un fichier de configuration.

Une annotation est identifiée par trois éléments : l'URL du document annoté, le texte surligné et l'occurrence du texte dans le document complet. D'autres attributs appelés métadonnées permettent d'enrichir chaque annotation. Les annotations sont stockées dans un fichier texte. Chaque valeur est séparée des autres par le symbole #

Dans YAWAS, les documents ne sont modifiés qu'une fois chargés par le navigateur avec toutes les pages que l'utilisateur peut charger, y compris les documents locaux, les documents HTML, TEXT et XML. Il inclut un moteur de recherche pour rechercher les annotations.

1.8 METEOR-S

METEOR-S [16, 17], développé au laboratoire LSDIS (Large Scale Distributed Information Systems) de Géorgie, dont l'objectif est d'intégrer les standards des services Web (BPEL4WS, WSDL…) avec les technologies du Web sémantique. Il comprend trois concepts importants :

- *Web Service Discovery Infrastructure* (MWSDI) une infrastructure de découverte sémantique. Partie *Front-end* dans la figure 28.
- WSDL-S un outil d'annotation sémantique son rôle et d'ajouter un niveau sémantique aux fichiers WSDL mais également à l'UDDI. Ces fichiers sont générés par Semantic Description Generator (voir figure 28).
- *Semantic Web Process Composition Framework* (MWSCF) un outil de composition de services. Partie *Back end* dans la figure 28.

Contrairement aux autres outils d'annotations sémantiques vus précédemment qui annotent généralement des pages Web, METEOR-S propose un système basé sur l'annotation sémantique du WSDL, de manière à enrichir des services existants avec des informations sémantiques. En utilisant des descriptions sémantiques, il devient possible de sélectionner des fonctionnalités de manière plus flexible.

Les trois principaux composants de l'outil d'annotation de METEOR-S [20] sont : *Ontology-Store*, *Matcher Library* et *Translator Library*.

- *Ontology store*, comme son nom l'indique, stocke les ontologies. Ces ontologies vont être utilisées par le système pour annoter la description du service Web en WSDL. Les ontologies sont regroupées par domaines. Le système permet à l'utilisateur d'ajouter de nouvelles ontologies aux ontologies existantes. Ces ontologies sont stockées en fichier .DAML ou .RDF dans des répertoires différents.
- *Translator Library* est un programme qui est utilisé pour générer les représentations du *SchemaGraph*. Il fournit deux transformations : *WSDL2graph et Ontology2graph*. *WSDL2graph* prend en entrée un fichier WSDL pour l'annoter et génère une représentation dans un graphe *SchemaGraph*, qui alimente l'algorithme de mise en correspondance. De la même manière l'*Ontology2Graph* génère le *SchemaGraph* pour l'ontologie.

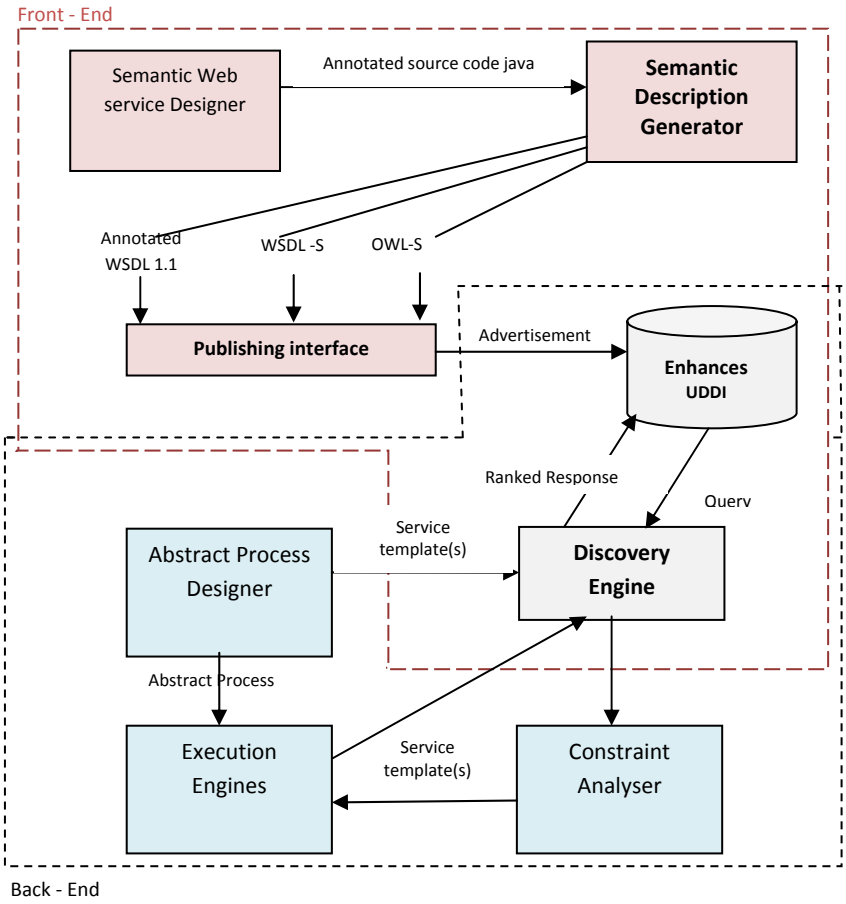


Figure 28 : Architecture de METEOR-S [17]

- Matcher Library fournit deux types d'algorithmes de mise en correspondance : la mise en correspondance au niveau élément et la mise en correspondance au niveau schéma. *findGraphMatch* est l'algorithme de mise en correspondance utilisé au niveau schéma (*cf Annexe C.* §3). Les mises en correspondance au niveau élément (*cf Annexe C.* §2) incluent les algorithmes *NGram*[72], *TokenMatcher*[73], *CheckSysnonyms*[74] et *CheckAbbreviations*.
  Cette librairie offre aussi à l'utilisateur la possibilité d'ajouter de nouveaux algorithmes de mise en correspondance en utilisant une API.

La fonction *getBestMapping* retourne un ensemble des meilleures mises en correspondance pour le schéma WSDL. Ces mises en correspondance peuvent être affichées par l'interface utilisateur. L'utilisateur peut accepter ou rejeter les mises en correspondance. Les mises en correspondance peuvent également s'effectuer manuellement. Lorsque les mises en correspondance sont acceptées elles sont insérées dans le fichier WSDL ou dans le registre UDDI.

2. Etude comparative et synthèse d'outils d'annotations sémantiques

Les outils d'annotations sémantiques ont été le plus étudiées dans le cadre du Web sémantique. Des études et synthèses, de quelques outils d'annotation ont été présentées dans [11, 12, 13, 14, 15, 24].

[72] nGram : Prend deux chaînes de caractère en entrée et calcul une séquence de n caractères communs entre elles.
[73] TokenMatcher **:** utilise l'algorithme Porter Stemmer (supprime les suffixes des mots en anglais) et d'autres techniques
[74] CheckSynonyms : utilise WordNet pour retrouver les synonymes

M. Rachel et al [11] présentent un comparatif, de quelques systèmes d'annotation, basé sur la localisation de l'annotation, de sa destination à un groupe privé ou public et sur la possibilité ou non de la consulter et de la rechercher.

Une autre étude [12], des outils d'annotation pour une mémoire personnelle dans le cadre d'une thèse, s'est focalisée sur le type d'objet d'annotation (annotation sémantique ou non) et le type d'activité (manuelle, automatique, cognitif[75], computationnelle[76]).

D'autres, A. Prié et al [13], se basent sur le type de ressources annotées, le langage d'annotation, l'architecture et l'utilisation des annotations

Dans [15] les auteurs décrivent le système SARM d'annotation de document Web Korean et compare son domaine d'utilisation aux systèmes KIM et MnM.

L. Reeve et al [24] présentent une synthèse et une classification, des plateformes d'annotations semi-automatiques (Armalido, kim, MnM, OntoMAT, SenTag), basée sur l'extraction de l'information (Machine d'apprentissage ou le schéma). Ils concluent que MnM présente la meilleure plate forme, d'annotation automatique, selon l'approche machine d'apprentissage et que OntoMat -Pankow est la meilleure plate forme selon l'approche basée sur le schéma.

Ces travaux n'ont pas inclus le cadre des services Web. Nous les avons consolidés donc et confortés à notre étude, selon les outils étudiés, pour les présenter dans le tableau 2.

Notre étude nous a permis de conclure que les systèmes d'annotation sémantique varient dans leur architecture, les outils d'extraction de l'information et dans les méthodes et les langages d'annotations. Ils dépendent aussi du cadre de leur utilisation (destiné à une collaboration, une recherche, une intégration,…). Ils peuvent être destinés à être traités par l'humain, l'agent logiciel ou par les deux.

Les annotations sémantiques peuvent être générées manuellement (Annotea, Yawas, SHOE, OntoMat Annotizer, SMORE) ou automatiquement (KIM, AERODAML). Certains outils combinent plusieurs types d'annotation : manuelle et automatique (COHSE et METEOR-S) ou aussi manuelle, automatique et semi-automatique (MnM).

Les outils d'annotations manuelles ne prévoient pas de fonctionnalités pour l'indexation ou la récupération d'annotation, ils sont plus destinés à une communication humaine au travers de commentaires, dans le cadre d'une collaboration. Les autres se concentrent sur l'indexation des documents favorisant leur rappel.

Les annotations sont le plus souvent séparées du document (KIM, COHSE, Annotea, OntoMat Annotizer) et stockées dans un ou plusieurs serveurs (Annotea), et sont parfois insérées dans le document (Yawas, SHOE, OntoMat Annotizer, METEOR-S) et rarement partagées (Annotea et Yawas) : modifiable par un autre utilisateur. Les outils d'annotation offrent souvent la possibilité de les rechercher et de les consulter.

Les documents annotés sont textuels : HTML (COHSE, Annotea, Shoe, OntoMat Annotizer, AERODAML, MnM, SMORE), structurés et non structurés (KIM), XML (METEOR-S, Annotea, Mnm et Yawas) et text (AERODAML). M- OntoMat Annotizer offre la possibilité d'annoter aussi des images.

La génération des annotations s'effectue sur des documents statiques. Dans notre étude YAWAS est le seul système qui génère l'annotation sur des documents dynamiques.

Les langages de description des annotations varient et peuvent être multiples: RDF (Annotea), DAML+OIL (COHSE, SMORE, AERODAML,), OWL (SMORE, OntoMat

[75] L'annotation est interprétable par l'utilisateur
[76] L'annotation peut être traitée par des agents logiciels

Annotizer, KIM, METEOR-S), XML (MnM et METEOR-S), textuels (Yawas), HTML (SHOE Knowledge Annotator) et en Java (METEOR-S).

Les outils d'annotation sémantique automatique ou semi automatique adoptent des approches différentes pour l'extraction de l'information. Certains se basent sur une approche basée sur les schémas (AERODAML, KIM) d'autres sur des inductions (MnM, Onto-mat annotizer). Dans METEOR-S, c'est une mise en correspondance au niveau élément (algorithmes Ngram, synonym,…) et au niveau schéma entre deux graphes (mesure de similarité entre concepts) qui est utilisée.

Certains outils exploitent des ontologies prédéfinies tels que : KIM (KIMLO, KIMSO), AERODAML (WordNet), MnM (Kmi), SOHE (SHOE). D'autres offrent la possibilité à l'utilisateur de créer ou d'ajouter des ontologies définie selon différents format : DAML+OIL (COHSE, METEOR-S), OWL (OntoMat Annotizer, SMORE, METEOR-S), ...

Ces outils ne regroupent pas toutes les performances et les propriétés attendues pour un système d'annotations sur le Web [14] et qui sont : la simplicité, l'efficacité, la transparence, l'indépendance vis à vis de la plate-forme support, le passage à l'échelle et le support du travail collaboratif.

Annexe B: Description de la base de liens sémantiques

Cette annexe décrit le schéma XML de la base de liens sémantiques ainsi qu'un exemple de base de liens sémantiques utilisée dans notre travail.

1. Description de la base de liens sémantiques

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSpy v2008 (http://www.altova.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:simpleType name="AT_8">
		<xs:restriction base="xs:string">
			<xs:enumeration value=""/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="AT_6">
		<xs:restriction base="xs:string">
			<xs:enumeration value="HTTP:\\"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="AT_5">
		<xs:restriction base="xs:string">
			<xs:enumeration value="UDD3"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="T_sortie">
		<xs:sequence>
			<xs:element ref="service" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="T_service">
		<xs:sequence>
			<xs:element ref="op_concept"/>
		</xs:sequence>
		<xs:attribute ref="ref_service" use="required"/>
		<xs:attribute ref="idf_registre" use="required"/>
		<xs:attribute ref="idf_SW" use="required"/>
	</xs:complexType>
	<xs:complexType name="T_op_concept">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute ref="type_op" use="required"/>
				<xs:attribute ref="ref_operation" use="required"/>
				<xs:attribute ref="idf_operation" use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="T_entree">
		<xs:sequence>
			<xs:element ref="service" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="T_correspondance">
		<xs:simpleContent>
```

```
                        <xs:extension base="xs:string">
                                <xs:attribute ref="type_correspondance" use="required"/>
                                <xs:attribute ref="ref_concept" use="required"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="T_concept">
                <xs:choice>
                        <xs:element ref="entree"/>
                        <xs:element ref="sortie"/>
                        <xs:sequence>
                                <xs:element ref="correspondance" maxOccurs="unbounded"/>
                                <xs:element ref="entree" minOccurs="0"/>
                                <xs:element ref="sortie"/>
                        </xs:sequence>
                </xs:choice>
                <xs:attribute ref="ref_concept" use="required"/>
                <xs:attribute ref="nom_concept" use="required"/>
                <xs:attribute ref="idf_concept" use="required"/>
        </xs:complexType>
        <xs:complexType name="T_baset">
                <xs:sequence>
                        <xs:element ref="concept" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:attribute name="type_op" type="xs:string"/>
        <xs:attribute name="type_correspondance" type="xs:string"/>
        <xs:attribute name="ref_service" type="AT_6"/>
        <xs:attribute name="ref_operation" type="AT_8"/>
        <xs:attribute name="ref_concept" type="xs:string"/>
        <xs:attribute name="nom_concept" type="xs:string"/>
        <xs:attribute name="idf_registre" type="AT_5"/>
        <xs:attribute name="idf_operation" type="xs:string"/>
        <xs:attribute name="idf_concept" type="xs:string"/>
        <xs:attribute name="idf_SW" type="xs:string"/>
        <xs:element name="sortie" type="T_sortie"/>
        <xs:element name="service" type="T_service"/>
        <xs:element name="op_concept" type="T_op_concept"/>
        <xs:element name="entree" type="T_entree"/>
        <xs:element name="correspondance" type="T_correspondance"/>
        <xs:element name="concept" type="T_concept"/>
        <xs:element name="baset" type="T_baset"/>
</xs:schema>
```

2. Exemple de la base de liens sémantiques

```
<?xml version="1.0" encoding="UTF-8"?>
<baset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="e:\magister\03112007\basetestlink.xsd">
        <concept ref_concept="c:\master\C1" nom_concept="livre.ISBN" idf_concept="C1">
                <correspondance type_correspondance="Exact" ref_concept="C11">
</correspondance>
                <entree>
                        <service idf_SW="ID_SW3" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW3" ref_operation="" type_op="in-
out">
                </op_concept>
```

```
                        </service>
                        <service idf_SW="ID_SW2" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW2" ref_operation="" type_op="in-
out">
                </op_concept>
                        </service>
                </entree>
                <sortie>
                        <service idf_SW="ID_SW1" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW1" ref_operation=""
type_op="out">
                </op_concept>
                        </service>
                        <service idf_SW="ID_SW6" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW6" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C2" nom_concept="livre.titre" idf_concept="C2">
                <sortie>
                        <service idf_SW="ID_SW3" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW3" ref_operation="" type_op="in-
out">
                        </op_concept>
                        </service>
                        <service idf_SW="ID_SW4" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW4" ref_operation=""
type_op="out">
                </op_concept>
                        </service>
                        <service idf_SW="ID_SW6" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op2_SW6" ref_operation="" type_op="in-
out">
                </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C3" nom_concept="Auteur.nom" idf_concept="C3">
                <correspondance type_correspondance="Subsume" ref_concept="C5">
</correspondance>
                <sortie>
                        <service idf_SW="ID_SW5" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op2_SW5" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C4" nom_concept="Auteur.nom" idf_concept="C4">
                <correspondance type_correspondance="Subsume" ref_concept="C5">
</correspondance>
                <sortie>
                        <service idf_SW="ID_SW1" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op2_SW1" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </sortie>
```

```
        </concept>
        <concept ref_concept="c:\master\C5" nom_concept="Auteur.prenom" idf_concept="C5">
                <correspondance type_correspondance="PlugIn" ref_concept="C3">
</correspondance>
                <correspondance type_correspondance="PlugIn" ref_concept="C4">
</correspondance>
                <sortie>
                        <service idf_SW="ID_SW6" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op3_SW6" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                        <service idf_SW="ID_SW4" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op3_SW4" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C6" nom_concept="livre.date_edition" idf_concept="C6">
                <sortie>
                        <service idf_SW="ID_SW2" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op1_SW2" ref_operation="" type_op="in-
out">
                        </op_concept>
                        </service>
                        <service idf_SW="ID_SW6" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op4_SW6" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C7" nom_concept="livre.editeur" idf_concept="C7">
                <entree>
                        <service idf_SW="ID_SW10" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op4_SW10" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                        <service idf_SW="ID_SW11" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op3_SW11" ref_operation=""
type_op="out">
                        </op_concept>
                        </service>
                </entree>
        </concept>
        <concept ref_concept="c:\master\C10" nom_concept="livre.price" idf_concept="C10">
                <sortie>
                        <service idf_SW="ID_SW3" idf_registre="UDD3" ref_service="HTTP:\\">
                                <op_concept idf_operation="op2_SW3" ref_operation="" type_op="in-
out">
                        </op_concept>
                        </service>
                </sortie>
        </concept>
        <concept ref_concept="c:\master\C11" nom_concept="livre.idf" idf_concept="C11">
                <sortie>
                        <service idf_SW="ID_SW5" idf_registre="UDD3" ref_service="HTTP:\\">
```

```
                                    <op_concept idf_operation="op2_SW5" ref_operation=""
type_op="out">
                                </op_concept>
                                </service>
                        </sortie>
                </concept>
</baset>
```

- Aperçu de l'ontologie utilisée en OWL

La figure 29 illustre un frgament de l'ontologie bibliotheque sous l'editeur protégé 2000.

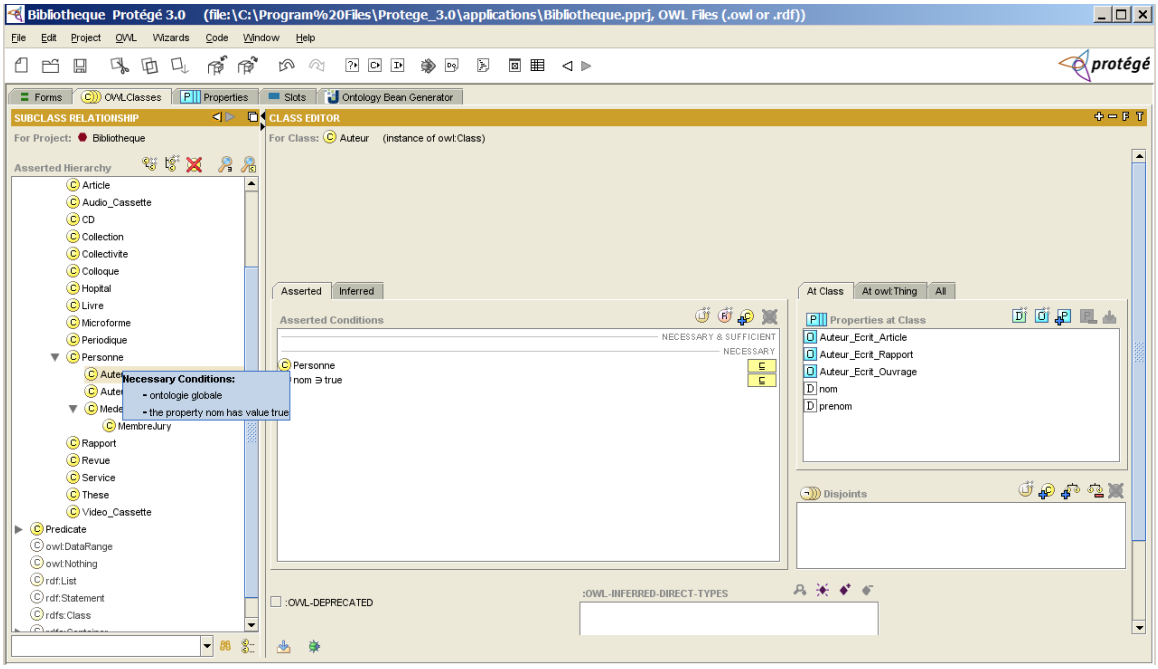


Figure 29: Aperçu de l'ontologie édition

Annexe C : Aperçu des techniques de mise en correspondance entre schémas

1. Introduction

Il existe plusieurs travaux sur les mises en correspondance automatique entre schémas ou entre ontologies. Des synthèses, des approches de matching automatique de schéma et une classification des systèmes les exploitant ont été présentées [32, 40, 41].

Nous distinguons principalement deux approches :

- *Les approches de matching individuelles* qui exploitent un seul algorithme pour générer les mises en correspondance et qui sont basées sur les schémas.
- *Les approches de matching combinées* qui utilisent plusieurs algorithmes indépendamment et combinent les résultats (Matcher composite) [41] ou plusieurs critères pour réaliser le matching.

Les approches de matching individuelles basées sur le schéma considèrent seulement l'information sur le schéma. Cette dernière inclut des noms d'éléments, des descriptions, des types de données, des types de relations (is a, part-of), des contraintes et des structures de schéma.

2. Techniques au niveau element

Les méthodes utilisées peuvent être :

- L'analyse de chaînes de caractères [32]. Elle inclut la comparaison de préfixes, de suffixes, le calcul de la distance *Levensthein*,[77]et les techniques des N-grammes[78].
- Le traitement du langage naturel. Il comprend la décomposition (caractères blancs,..), la lemmatisation et l'élimination (conjonction, préposition,…)
- L'utilisation des ressources linguistiques ou auxiliaires. Ces ressources peuvent être des thésaurus génériques [41] ou des dictionnaires ce qui permet de trouver la similarité ou de comparer des synonymes et des hypernymes[79]. L'ontologie linguistique (voir §3.3) peut également être utilisée comme ressource externe.
- La réutilisation de l'information d'un match généré précédemment pour le calcul de la composition du mappings.
- L'utilisation des techniques de recherche d'information pour comparer les descriptions qui annotent certains éléments du schéma.
- L'utilisation de contraintes d'un schéma, telles que les contraintes des types de données, les contraintes de valeurs, d'unicité, de cardinalités (1:1 ; 1:n ; n:n ;…), et également des relations inter schéma telle que l'intégrité référentielle.

[77] Levensthein : mesure de similarité entre deux chaînes : nombre minimal de caractères pour passer d'une chaîne à l'autre.
[78] N-gramme: prend deux chaînes de caractère en entrée et calcul une séquence de n caractères communs entre elles.
[79] X est hypernyme de Y si Y est un type de X

3. Techniques au niveau structure

Le matcher au niveau structure [32, 41] compare des combinaisons des éléments qui apparaissent ensembles dans un schéma, par exemple les colonnes dans les tables. Il peut y avoir un matching complet où tout le schéma est mappé ou uniquement une partie du schéma. Les algorithmes implémentent diverses heuristiques.

Nous distinguons les techniques suivantes [32] :

- *Les techniques basées sur les graphes*. Elles représentent les algorithmes sur les graphes. Le matching de graphes peut être vu comme un problème combinatoire résolu par les méthodes d'optimisation en utilisant l'algorithme du point fixe.
- *Les techniques basées sur les taxonomies.* Ce sont également des algorithmes de graphe qui considèrent uniquement la relation de spécialisation. Il peut s'agir de chemin matching ou des règles d'intuition (par exemple : si les super concepts sont similaires alors les concepts sont similaires).
- *Les techniques de stockage de structure*. Elles stockent le schéma/ontologie et son fragment ensemble avec une similarité pair à pair entre eux et un coefficient dans l'intervalle [0,1]. Ces techniques utilisent des indicateurs (racine, nombre de nœuds,..) de similarité qui sont représentés comme un seul coefficient.
- *Les techniques basées sur les modèles* telles que les techniques de raisonnement de logique de description ou de satisfiabilité propositionnelle (SAT) [32,41]. Les graphes sont décomposés en un problème de correspondances de nœuds qui sont traduits en formules propositionnelles. Ces correspondances considèrent non seulement le nom mais aussi la position des concepts dans le graphe.

Annexe D : Les services Web sémantiques

1. Introduction

L'infrastructure de base des services Web ne permet pas encore d'effectuer une gestion largement automatisée. La description des services, la découverte et l'invocation et Web, prennent en considération uniquement la syntaxe. L'aspect sémantique indispensable pour automatiser certaines de ces tâches y est absent.

Dans WSDL le nom des interfaces, le nom des opérations ou le type d'entrées et sorties ainsi définis ne permettent pas de déterminer leurs significations.

Pour remédier à ce problème plusieurs travaux sur les services Web sémantique ont été proposés [16, 17, 31, 45, 68], et dont certains ont été soumis au W3C tels que : OWL-S[45], WSMO[31], SWSF[80], METEOR-S[16,17]. Ces travaux proposent une description du service Web par l'utilisation des annotations et/ou des ontologies.

Cette annexe décrit quelques approches de description sémantique.

2. Description de quelques approches de description sémantique

2.1 OWL-S

OWL-S[81] (Ontology Web Language for Services) [45] est un langage qui définit une ontologie de Services Web, basé sur le langage OWL. Il est le successeur de DAML-S (DARPA Agent Markup Language Service Ontology) [23].

OWL propose trois sous-langages [14]: OWL Lite, OWL DL, OWL Full, classés selon les besoins d'expressivité et de complexité.

OWL-S permet la découverte et la sélection automatique d'un service, l'invocation automatique de service, l'interopération et la composition automatique de services et la surveillance automatique de l'évolution des services [45].

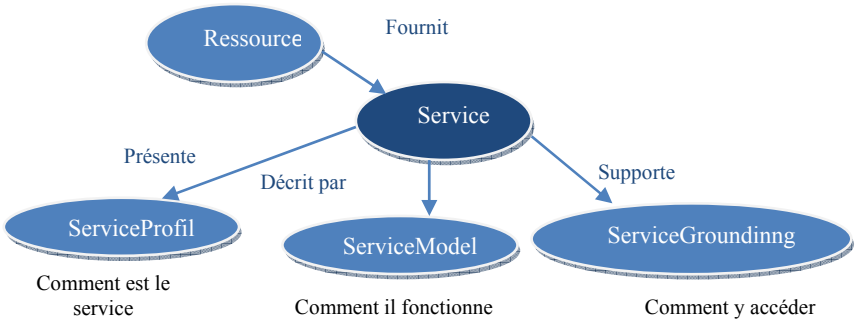


Figure 30 : Principales classes d'OWL-S [45]

Comme illustré dans la figure 30, OWL-S définit une ontologie de service Web avec quatre éléments principaux *: Service, Serviceprofil, Servicemodel et Servicegrounding*.

[80] http://www.w3.org/Submission/SWSF
[81] http://www.daml.org/services/owl-s/

- La classe *Service* fournit le point d'entrée pour la description d'un service Web. Chaque description d'un service Web décrit une seule instance de cette classe avec des valeurs appropriées : *Presented, Described by* et *Support*.
- Le *Serviceprofil* décrit ce que le service Web fait à un niveau élevé. Il est utilisé lors de la publication et de la recherche d'un service. Il contient les informations sur le fournisseur du service, sur la description fonctionnelle du service (*Input, Output, Effects* et *Precondition*) et sur des attributs additionnels incluant la garantie de la qualité que fournit le service.
- Le *Servicemodel* décrit le fonctionnement et la définition des différents processus du service Web.
- Le *ServiceGrounding* spécifie les détails sur l'accès au service. Ces détails concernent les protocoles et le format du message, la sérialisation, le transport, et l'adressage.

2.1.1 Interaction entre OWL-S et WSDL

L'OWL-S Grounding est basé sur trois correspondances entre OWL-S et WSDL (voir figure 31).

1. Un processus atomique OWL-S correspond à une opération dans WSDL
2. Dans le cas où message part utilise le type OWL, Chacun des ensembles d'entrées et de sorties d'un processus atomique OWL-S correspond au concept du message de WSDL
3. Les types des entrées et des sorties d'un processus atomique de OWL-S correspondent à la notion extensible de WSDL du type abstrait.

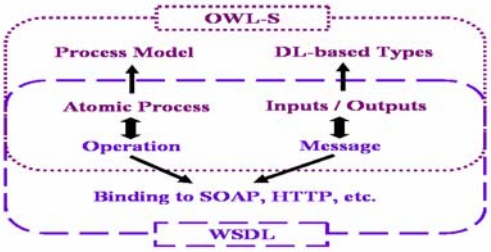


Figure 31 : Mapping entre OWL-S et WSDL[45]

2.2.2 Interaction entre OWL-S et UDDI

Comme illustré dans la Figure 32, l'interaction entre OWL-S et UDDI s'effectue à travers un *Moteur de matching*. Lors de la publication d'un service par un fournisseur, le *Moteur de matching* reçoit une annonce du service en OWL-S (*ServiceProfile*) et fait alors appel à un *Traducteur*. Ce dernier construit une description du service, l'enregistre en UDDI et renvoie au *Moteur de matching*, un identifiant. Il mémorise l'identifiant avec l'annonce du service en OWL-S.

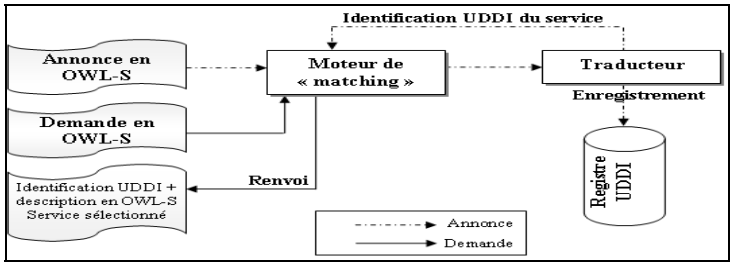


Figure 32 : Interaction entre OWL-S et UDDI

Lors d'une requête, le *Moteur de matching* emploie l'algorithme Matchmaker [46] pour sélectionner le service approprié et renvoie alors l'identification UDDI du service et sa description OWL-S.

2.2 WSMO

WSMO[82] (Web Service Modeling Ontologie) décrit un model pour la description sémantique des services Web. Son point de départ est la plateforme WSMF (Web Service Modeling Framework) et s'appui sur la plateforme et l'infrastructure IRS-III (Internet Reasoning Service) [29].

WSMO consiste en quatre principaux éléments pour la description des services Web sémantiques [31]: Les Ontologies, les objectifs, la description des Services Web et les médiateurs.

1. Les ontologies représentent l'élément clé car elles fournissent la terminologie utilisée par les autres éléments.
2. Les objectifs qui définissent les problèmes qui devrait être résolus par les servies Web.
3. Les descriptions des services Web définissent les aspects variés des services Web, utilisent les protocoles standards basés sur le Web pour l'échange et l'extension des services Web. Elles sont décrites selon trois perspectives différentes : les propriétés non fonctionnelles, fonctionnelle et comportale.
4. Les médiateurs permettent de relier des ressources hétérogènes et peuvent être de quatre types :
    - ggMediator: médiateur qui permet de lier des objectifs.
    - ooMediator: médiateur qui importe les ontologies et résout les mises en correspondance possibles entre ontologies.
    - wgMediator: médiateur qui lie les services Web aux objectifs.
    - wwMediators: médiateur qui relie deux services Web.

Aussi, deux autres groupes de travail sont liés à l'initiative WSMO: le WSML qui se focalise sur le développement de langages de modélisation de service Web et le WSMX[83] (Web Service Modeling eXecution environment) qui construit une référence pour l'implémentation d'un environnement d'exécution du WSMO.

WSML définit les langages : *WSML userlanguage* qui est destiné à l'utilisateur et *WSML/XML* une syntaxe XML pour WSML qui s'appui sur *F-logic/XML*.

2.3 METEOR-S

METEOR-S [16, 17], développé au laboratoire LSDIS de Géorgie, dont l'objectif est d'intégrer les standards des services Web (BPEL4WS, WSDL…) avec les technologies du Web sémantique.

Contrairement aux autres outils d'annotations sémantiques (*cf Annexe A*) qui annotent généralement des pages Web, METEOR-S propose un système basé sur l'annotation

[82] http://www.wsmo.org/
[83] http://www.wsmx.org/

sémantique de descriptions WSDL, de manière à enrichir des services existants avec des informations sémantiques.

2.3.1 Annotation du WSDL

Dans METEOR-S, le principe d'annotation du document WSDL consiste à étendre les éléments et attributs supportés dans la spécification WSDL1.2. Les opérations du document WSDL sont mises en correspondance avec des concepts appropriés d'une ontologie DAML+OIL ou OWL. Ce qui permet à l'utilisateur de rechercher des opérations basées sur les concepts de l'ontologie et de déterminer les fonctionnalités du service.

L'approche de stockage des annotations dans le registre UDDI est décrite dans la prochaine section.

En utilisant l'outil d'annotation, une fonction appelée getBestMapping retourne un ensemble des meilleures mises en correspondance pour le schéma WSDL. Ces mises en correspondance peuvent être affichées par l'interface utilisateur. L'utilisateur peut accepter ou rejeter les mises en correspondance. Les mises en correspondance peuvent également s'effectuer manuellement.

2.3.2 Représentation de la sémantique dans l'UDDI

Dans METEOR-S, la découverte sémantique des services Web est fournit en effectuant les tâches suivantes :

- Stocker les annotations sémantiques des services Web définies dans WSDL dans la structure existante de l'UDDI.
- Fournir une interface qui construit les requêtes et qui utilise les annotations sémantiques.

Cette approche [30] est similaire à celle qui fait correspondre DAML-S à La structure de l'UDDI, mais elle est plus consistante car elle utilise le standard WSDL.

L'UDDI supporte des formes limitées de la sémantique en utilisant le *tModel* ; Dans [30] les auteurs proposent d'utiliser *KeyReferenceGroup* avec *tModel* pour regrouper les opérations avec leurs entrées et sorties.

Pour représenter la sémantique dans UDDI, quatre tModel ont été créée [30] :

1. Le premier représente l'ontologie de concepts représentant les fonctionnalités des opérations dans un domaine précis.
2. Le deuxième représente l'ontologie des concepts d'entrées
3. Le troisième représente l'ontologie des concepts de sorties
4. Le quatrième représente le regroupement de chaque opération avec ses entrées et sorties.

Ces *tModels* sont liés et doivent aussi être reliés à une seule ontologie. Les pré-conditions et effets ont besoin de techniques similaires.

2.3.3 Découverte sémantique des services Web dans METEOR-S

Les annotations sémantiques ajoutées dans les fichiers WSDL et UDDI ont pour but de faciliter la découverte et la composition des services Web.

L'algorithme de découverte de service Web comprend trois phases [30] :

- 1ère phase : Mise en correspondance des services Web (opérations dans les différents fichiers WSDL) basée sur les fonctionnalités qu'ils fournissent.
- 2ème phase : L'ensemble résultant de la première phase est classifié sur la base de similarité sémantique entre les concepts d'entrées et de sorties de l'opération sélectionnée et les concepts d'entrées et sorties de template.

- 3ème phase : Cette phase est optionnelle, elle évoque la classification basée sur la similarité sémantique entre les concepts pré-conditions et effets du template en utilisant les nœuds ontologiques pour la découverte sémantique du service Web.

Le template crée par l'utilisateur est converti en une requête UDDI qui recherche toutes les catégories des services Web en utilisant *KeyReferenceGroup* ; l'ensemble des résultats est par la suite classifié, en se basant sur les similarités sémantiques entre les concepts en entrées du service Web retourné et les concepts en entrées du template et entre les concepts en sorties du service Web retourné et les concepts en sorties du template.

Annexe E : Exemple de programme d'interrogation en XQUERY

Cette annexe présente le programme d'utilisation de l'API nux pour l'interrogation de la base de liens sémantiques et du registre UDDI en XQUERY et un exemple de requête en XQUERY.

```
**
 * <p>Titre : Magister_Web_service_discovery</p>
 * <p>Description : </p>
 * <p>Copyright : Copyright (c) 2007</p>
 * <p>Société : </p>
 * @author BENNA AMEL
 * @version 1.0
 */
import nux.xom.xquery.XQuery;
import java.io.File;
import nu.xom.Builder;
import nu.xom.Document;
import nux.xom.xquery.XQuery;
import nu.xom.Node;
import nu.xom.Nodes;
import nux.xom.xquery.XQueryUtil;
import nux.xom.xquery.*;
import nux.xom.pool.XOMUtil;
import java.io.*;
import java.lang.String;
import java.util.Vector;
import java.io.*;
import java.lang.String;
import org.xml.sax.*;
public class Querying_link_base {
   public Querying_link_base() {
   }
   /**     * requete_xml     */
   public static void requete_xml() {
      try
```

```
{
System.out.println("**************************DEBUT**************************
*");
  Document doc = new Builder().build(new
File("C:\\localhost\base.xml"));
  String requete="for $a in //sortie/@idf_registre return $a";
  Nodes results = XQueryUtil.xquery(doc,requete);
  System.out.println ("resultat : " + results.size());
for (int i=0; i < results.size(); i++) {
  System.out.println("node " + i + ": " + XOMUtil.toPrettyXML(results.get(i)));
}
 System.out.println("**************************FIN**************************");
}
catch (Exception ex) {
          ex.printStackTrace();
         }
    }
 public static void main(String[] args) {
        try
  {
     /*initialisation du vecteur de condition*/
     requete_xml();
       Querying_link_base querying_link_base = new Querying_link_base();
  }
  catch (Exception ex) {
            ex.printStackTrace();
           }
  }
}
```

Exemple de programme en XQUERY

- Pour retrouver les services correspondant à la recherche d'un concept 'ISBN' ou d'un concept correspondant le programme est le suivant :

```
<Resultat>
{
 let $p:='ISBN'
 let $LCE:=local:concept_correspondant($p)(:LCE est la liste des concepts correspondant à $p:)
 for $i in ($p, $LCE//concept_idf)
 return
          <concept>
          {$i}{local:trouve_service_sortie_correspondant($i)}
           </concept>
  }
</Resultat>
```

- **Fonction de recherche de concept correspondant** : TROUVE_CONCEPT_CORRESPONDANT ()

Cette fonction interroge la *base de liens sémantiques* pour retourner les concepts correspondants à un concept donné avec le degré de mise en correspondance. Sa description en Xquery est la suivante :

```
declare function local:concept_correspondant($p as xs:string) as element()*
(: déclaration de fonction de récupération des identificateurs de concepts mis en correspondance
avec le degré de mise en correspondance :)
{
   for $a in doc("http:\\localhost\\base.xml ")//concept
   where $a/@idf_concept = $p
   return
   for $c in distinct-values($a/correspondance/@ref_concept),
       $d in distinct-values($a//@type_correspondance)
    return  <correspond>
              <concept_idf>{$c}</concept_idf>
                <type_correpondance>{$d} </type_correpondance>
             </correspond>
};
```

Exemple

Soit idf_concept='ISBN'

Le résultat retourné après exécution de la fonction TROUVE_CONCEPT_CORRESPONDANT (ISBN) est :

```
<Resultat>
          <correspond>
          <concept_idf>code_livre</concept_idf>
          <type_correspondance>Exact</type_correspondance>
          </correspond>
</Resultat>
```

- **Fonction de récupération des services Web** : ayant un concept 'p' donné somme sortie **:** TROUVE_SERVICE_SORTIE(p)

```
declare function local:trouve_service_sortie_correspondant($p as xs:string) as element()*
(:Fonction de récupération des services Web ayant un concept 'p' donné somme sortie:)
{
    for $a in doc("http:\\localhost\\base.xml")//concept
    for $e in $a/sortie/service
    where $a/@idf_concept=$p
    return
    let $c:=$e/@idf_SW, $d:=$e/op_concept/@idf_operation
    return <services>{$c} {$d} </services>
};
```

Résultat de l'interrogation des services correspondant au concept « ISBN » est le suivant :

```
<Resultat>
  <concept> ISBN
     <services idf_SW="ID_SW1" idf_operation="op2_SW1"/>
 </concept>
 <concept> <concept_idf>code_livre</concept_idf>
<type_correspondance>Exact</type_correspondance>
   <services idf_SW="ID_SW6" idf_operation="op3_SW6"/>
   <services idf_SW="ID_SW4" idf_operation="op3_SW4"/>
</concept>
</Resultat>
```