République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abou-bekr Belkaid Tlemcen

Faculté des Sciences

Département d'Informatique

Mémoire

Pour l'obtention du diplôme de :

Magistère en Informatique

Ecole doctorale: **S**cience et **T**echnologie de l'**I**nformation et de la **C**ommunication (**STIC)**

Option : **S**ystèmes d'**I**nformation et de **C**onnaissance (**SIC)**

Présenté et soutenu publiquement par :

Amina BEKKOUCHE

Thème

Composition des Services Web Sémantiques À base d'Algorithmes Génétiques

Devant le jury composé de:

Président :	Mr Chikh Amine	Professeur	à l'université Abou-bekr Belkaid Tlemcen
Examinatrice :	Mme DIDI Fedoua	MC-A	à l'université Abou-bekr Belkaid Tlemcen
Directeur de mémoire :	Mr BENSLIMANE Sidi Mohammed	MC-A	à l'université Djillali Liabes Sidi Bel Abbés
Invité :	Mr Benmamar Badr	MC-B	à l'université Abou-bekr Belkaid Tlemcen

2011-2012

Remerciements

Je remercie le bon DIEU, qui m'a donné la force, la volonté et le courage pour terminer ce modeste travail;

Je tiens à exprimer d'abord toute ma gratitude à mon encadreur Dr BENSLIMANE Sidi Mohammed de l'université Djillali Liabes –Sidi Bel Abbés pour m'avoir proposé ce travail, pour son encadrement, son écoute, ses élucidations, ses conseils, ses directives et encouragements qu'il m'a afflués.

Je remercie Mr CHIKH Amine, professeur à l'université Abou Bekr Belkaid –Tlemcen, de m'avoir fait l'honneur de présider le jury de ma soutenance.

Mes remerciements s'adressent à Mme DIDI Fedoua maître de conférences à l'université Abou Bekr Belkaid-Tlemcen, ainsi Mr Benmamar Badr maître de conférences à l'université Abou Bekr Belkaid-Tlemcen pour l'intérêt qu'ils ont porté à ce travail en acceptant d'être examinateurs.

Je tiens aussi à remercier mon fiancé Mr MERZOUG Mohamed maître assistant à l'université Abou Bekr Belkaid-Tlemcen et Mr HADJILA Fethallah maître assistant à l'université Abou Bekr Belkaid-Tlemcen qu'ils ont été les plus bénéfiques pour moi.

C'est un énorme remerciement que j'adresse à ma Mère et mon Père ainsi que mes sœurs, sans oublier mon frère. Vous avez su à votre manière, par vos paroles et vos gestes, m'encourager et m'accompagner dans tous les moments de ce mémoire.

Tout simplement à tous ceux et celles qui méritent mes remerciements.

Dédicace

A ma mère, à ma mère, à ma mère et à mon père

A mon frère, mes sœurs et leurs familles

Spécialement à mon fiancé Mohamed pour son encouragement, sa patience et son aide.

A mes collègues et mes amies

Résumé

Le processus de composition des services web s'est avéré essentiellement un processus qui prenait beaucoup de temps. En effet, une composition n'est pas simplement un regroupement quelconque de services web, mais un ensemble dont les tâches sont ordonnées en fonction des relations reliant ses services web. Ces derniers sont généralement fournis par des organisations différentes et indépendamment de tout contexte d'exécution. Puisque, chaque organisation possède ses propres règles de travail, ils doivent être traités comme des unités strictement autonomes. A cause de cette hétérogénéité et de cette autonomie inhérente aux services web sémantiques, leur composition devient plus complexe. De ce fait le besoin d'une composition dynamique et automatique se fait sentir. Dans ce mémoire, nous proposons une approche pour la composition dynamique des services web sémantiques basée sur un algorithme génétique, en prenant en compte à la fois les propriétés fonctionnelles des services Web -(spécifiées par les paramètres d'E/S)- et les propriétés non fonctionnelles -(spécifiées par les paramètres de QoS)-afin de satisfaire les exigences de l'utilisateur. Les résultats obtenus sont encourageants et méritent d'être poursuivis.

Mots-clés: web sémantique, service web, composition des services, appariement sémantique, qualité de service, algorithme génétique.

Abstract

The web service composition is a time-consuming process. In fact, a composition is not simply a collection of Web services, but a set, whose tasks are ordered according to the relations between these Web services .These are usually provided by different organizations and consequently an independent execution context. Since each organization has its own rules of work, they must be treated as strictly autonomous units. Because of this heterogeneity and inherent autonomy, the web service composition becomes more complex. Hence the need for a dynamic and automatic composition arises. In this work we propose an approach for the dynamic composition of semantic web services based on genetic algorithm. This later takes into account both functional properties of web services -specified by their I/O parameters- and non-functional properties -specified by their QoS parameters- in order to fulfill the user requirements. The obtained results are encouraging and merit to be continued.

Keywords : semantic web, web service, service composition, semantic matching, quality of service, genetic algorithm.

ملخص :

تعتبر عملية تركيب خدمات الواب جد مكلفة من ناحية الوقت و الجهد، هاته الصعوبات راجعة لعدة عوامل أهمها وجود الاختلافات ذات الطابع الدلالي (Sémantique) ،النحوي (Syntaxique)، ومنطق عمل المسار المركب .(Logic métier)

كما تجدر الإشارة أن استقلالية خدمات الواب ووجود مقاييس جودة الخدمة تزيد من صعوبة التركيب الآلي لخدمات الواب.

لمقاربة هذه الإشكالية، نقترح في هاته الرسالة حلا يعتمد على الخوارزميات الجينية من أجل تحسين أو استمثال دالة موضوعية تمثل جميع متطلبات المستخدم (الطابع الدلالي، جودة الخدمة، الاستلزامات الشاملة).

النتائج المتحصل عليها تعتبر جد مشجعة و تفتح المجال من أجل مقاربات أخرى.

الكلمات المفتاحية: الواب الدلالي،خدمات الواب، تركيب الخدمات، جودة الخدمة، الخوارزمية الجينية.

Table des matières

Table des illustrations

Figures

Tableaux

Introduction Générale

Contexte

Avec une évolution exponentielle dans le temps, les SI (Systèmes d'Informations) sont devenus plus complexes et plus hétérogènes en raison de la diversité des besoins, des exigences des clients, et de la grande masse d'information stockée et manipulée par les internautes. Les infrastructures publiques et privées du secteur informatique sont de plus en plus multiplateformes, multifournisseurs et distribuées à grande échelle. Dans une telle situation, nombreux sont les professionnels de l'informatique qui considèrent que l'interopérabilité est un aspect aussi important que la sécurité et la fiabilité pour la gestion de leurs SI et leurs environnements de fonctionnement.

Assurer l'interopérabilité est l'une des clés de succès de développement et d'intégration des grands systèmes d'information. Cet objectif présente le noyau des travaux de recherches dédiés à l'amélioration des architectures, des plateformes et des technologies de développement des systèmes d'information depuis les méthodes cartésiennes jusqu'aux architectures orientées services, plus connue avec l'abréviation SOA(Service Oriented Architecture).Ces dernières présentent actuellement la vision architecturelle des SI modernes.

SOA est un style architectural qui permet de construire des solutions d'entreprises basées sur les services. Au début, sa mise en place a été basée sur les technologies et les middlewares conçus pour le modèle « objets distribués » comme CORBA, RMI et DCOM. Cependant, chacune de ces technologies proposent sa propre infrastructure, ce qui impose une forte liaison entre les services fournis et leurs consommateurs.

En plus de l'urbanisation de son SI interne, l'entreprise d'aujourd'hui a besoin d'une ouverture vers son environnement économique pour accomplir des échanges interentreprises avec ses partenaires par le biais d'Internet (B2B : Business to Business), et répondre le plus rapidement possible aux besoins de ses clients (B2C : Business to Consumer).

Actuellement, les solutions EAI (pour Enterprise Applications Integration) sont orientées vers les processus métiers et les échanges interentreprises. Elles prennent en compte les nouveaux modèles économiques créés et promus par Internet et ses technologies (TCP/IP, SMTP, HTTP, FTP, XML, etc.). De plus, les progiciels de gestion intégrés de la deuxième génération apportent une quantité de nouveaux modules organisés autour de la

SOA.C'est dans ce contexte que les services web sont apparus pour mettre en place cette architecture.

A la différence des technologies précédentes, les services web proposent une architecture par composants qui permet à une application de faire l'usage d'une fonctionnalité située dans une autre application. Cependant, cette solution repose sur l'ubiquité de l'infrastructure d'Internet alors que les autres architectures reposent chacune sur sa propre infrastructure. L'interopérabilité est donc une caractéristique intrinsèque aux services web parce qu'ils sont basés sur des technologies Web dérivées du fameux standard XML.

Depuis la naissance du paradigme des services web en l'an 2000, ils sont été considérés comme une révolution pour le Web. En effet, avec les trois premières technologies de base :SOAP (Simple Object Access Protocol),UDDI (Universal Description Directory and Integration) et WSDL (Web services Description Language), les services web ont connu un grand succès grâce au haut degré d'interopérabilité fourni par les standards. Les services web sont présentés comme le meilleur moyen pour concrétiser la SOA (Service Oriented Architecture)et pour mettre en place le paradigme SOC (Services Oriented Computing).

Cependant, les services web possèdent aussi des inconvénients. En effet, il se peut qu'un client ait des besoins spécifiques qui ne sont rendus par aucun service web. A cause de l'élargissement d'internet, il est rare pour un consommateur de découvrir un service web qui réponde à ses besoins. Pour cette raison, plusieurs services peuvent interagir et échanger dynamiquement des informations. L'objectif de cette interaction est l'accomplissement d'un but complexe non concevable par un seul service web. Cette agrégation des compétences pour réaliser un but commun est connue par « composition des services web ».

Comme nous l'avons déjà précisé, les services web sont conçus pour être composés afin d'accomplir des buts complexes non réalisables par un seul service web. Pour aboutir à ces objectifs, les services web doivent interagir dynamiquement avec le minimum d'intervention humaine. Ceci nécessite la prestation d'une description plus intelligible et plus compréhensible par la machine. C'est dans ce contexte que « le web sémantique » peut intervenir pour améliorer la description des services web en utilisant des langages

telqu'OWL.La combinaison entre les services web et les technologies du web sémantique ont donné lieu à la naissance des « services web sémantiques ».

Après que le web sémantique ait résolu le problème des standards de descriptions classiques comme WSDL et UDDI, qui ne décrivent pas suffisamment les connaissances d'un service web, il s'est avéré que la composition dynamique des services web nécessite un moyen plus efficace qu'UDDI qui ne permet pas d'agréger des services web d'une manière adaptative et correcte.

Afin de résoudre ce problème, plusieurs solutions ont été proposées dans la littérature pour procéder à la composition dynamique des services web, on trouve des approches qui sont basées sur les techniques de workflow, de planification, des automates, des réseaux de pétri et des algorithmes évolutionnaires.

Problématique

Le processus de composition s'est avéré essentiellement un processus qui prenait beaucoup de temps. En effet, une composition n'est pas simplement un regroupement quelconque de services web, mais un ensemble dont les tâches sont ordonnées en fonction des relations reliant ses services web. Ces derniers sont généralement fournis par des organisations déférentes et indépendamment de tout contexte d'exécution. Puisque, chaque organisation possède ses propres règles de travail, ils doivent être traités comme des unités strictement autonomes. A cause de cette hétérogénéité et de cette autonomie inhérente aux services web sémantiques, leur composition devient plus complexe.

En outre, une composition manuelle des services web génère beaucoup d'erreurs et comporte plusieurs tâches fastidieuses et répétitives. De ce fait le besoin d'une composition dynamique et automatique se fait sentir, un tel système de composition doit prendre en compte l'optimisation de la QoS (Quality of Service), la satisfaction des contraintes globales de l'utilisateur et la préservation de la cohérence sémantique du flux de données échangé entre les services d'une composition.

Contribution

Plusieurs travaux dans la littérature ont adopté les techniques des AG's pour résoudre le problème de la composition au niveau non fonctionnel c.-à-d. en se basant sur les caractéristiques non fonctionnelles des services web. Ce problème est connu sous le nom « QoS aware service composition »[Canfora et al.,2005][Cardoso, 2002][Huan et al., 2010].D'autres efforts se sont consacrés au problème de la composition au niveau fonctionnel en exploitant les descriptions syntaxiques des services web [Chouchani, 2011] ou les descriptions sémantiques comme dans [Weise et al.,2007] et [Lécué et al.,2006].Peu d'approches ont abordé le problème de la composition des services web en prenant en compte les deux paramètres fonctionnels et non fonctionnels des services web.

Les principales contributions de ce travail de magister sont les suivantes :

- L'élaboration d'un état de l'art détaillé survolant les différentes approches de composition des services web. Une classification synthétisé des différentes contributions selon des critères d'évaluation que nous avons défini nous a permis d'une part de dégager les atouts et les limites de chaque approche, et d'autre part de mieux positionner notre approche pour voir comment il serait possible d'améliorer ce domaine de recherche.

- Proposition d'une approche de composition dynamique des services web sémantiques basée sur les algorithmes génétiques. L'approche proposée considère à la fois les propriétés fonctionnelles (c.-à-d. le flux de données échangé entre les services web en terme d'entrées et de sorties) et non fonctionnelles (les paramètres de QoS des services web tels que le temps de réponse, le coût, la réputation, la disponibilité, la fiabilité, etc.).

- Proposition d'un mécanisme de Matching sémantique basé sur les I/O des services web. Pour explorer l'espace de recherche, l'AG proposé adopte deux fonctions de mutation: la première « mutate 1 » permet de retirer aléatoirement un service web, la deuxième fonction « mutate 2 » permet de remplacer un service web par un autre qui possède une compatibilité sémantique non nulle avec les voisinages.

- Implémentation d'un prototype baptisé « *Genetic Composer* » pour montrer l'intérêt et la faisabilité de notre approche. Etant donnés une requête d'utilisateur et un ensemble de services web sémantisés, notre outil implémente un algorithme génétique visant à trouver une solution optimale (une combinaison de services web) de telle façon que :
  1. La similarité sémantique avec la requête est maximisée.
  2. La QoS de la composition (service composite) est maximisée.
  3. Les contraintes globales de l'utilisateur sont satisfaites.

Organisation du mémoire

Le manuscrit est structuré comme suit :

Le premier chapitre est consacré aux algorithmes génétiques en décrivant leurs concepts de base ainsi que leurs avantages et leurs inconvénients. Dans le deuxième chapitre nous présentons l'architecture SOA, les services web et les technologies adjacentes. Le chapitre 3 résume les objectifs visés par les web services sémantiques ainsi que les différents langages qui sont utilisés pour expliciter la sémantique dans les descriptions des services web. Le chapitre 4 représente un état de l'art sur la composition des services web, dans un premier lieu nous citons les différents langages et types de composition et puis dans un deuxième lieu, une synthèse des travaux de recherche proposant des approches de composition est décrite, suivie d'une étude comparative selon des critères d'évaluation que nous avons définit. Le chapitre 5 est lié à notre contribution, nous présentons l'approche de composition proposée tout en détaillant notre mécanisme de matching sémantique et le modèle QoS et puis nous détaillerons l'algorithme génétique(codage, fitness,opérateurs génétiques,…..).Dans le dernier chapitre nous présentons l'outil expérimental développé ainsi que les résultats obtenus. Pour finir, une conclusion générale reprendra notre contribution dans ce mémoire, et en analysera ses limites. Nous proposerons enfin quelques perspectives de recherche de ce travail.

Chapitre1 : Les Algorithmes Génétiques

1.1 Introduction

Les algorithmes génétiques sont des algorithmes d'optimisation qui utilisent la notion de sélection naturelle développée au XIX^{e} siècle par le scientifique Darwin et l'appliquent à une population de solutions potentielles d'un problème donné. C'est au début des années 1960 que John Holland de l'Université du Michigan a commencé à s'intéresser à ce qui allait devenir les algorithmes génétiques. Ses travaux ont trouvé un premier aboutissement en 1975 avec la publication de *'Adaptation in Natural and Artificial System'* [Holland,1975].

Les algorithmes génétiques appartiennent à la famille des algorithmes évolutionnistes (un sous-ensemble des métaheuristiques). Leur but est d'obtenir une solution approchée, en un temps correct, à un problème d'optimisation lorsqu'il n'existe pas (ou qu'on ne connaît pas) de méthode exacte pour le résoudre en un temps raisonnable.

Dans cette partie nous allons présenter le principe de fonctionnement des algorithmes génétiques tout en introduisant les principaux éléments qui les caractérisent.

1.2 Algorithmes génétiques : définition

Les techniques de recherche et d'optimisation sont en général classées en trois catégories [Coello et Carlos, 2000]: énumératives, déterministes et stochastiques. Les AG font partie de la troisième catégorie et quatre caractéristiques les distinguent des autres techniques d'optimisation [Goldberg, 1989]:

- ils utilisent un codage des paramètres et non les paramètres eux-mêmes;
- ils travaillent sur une population d'individus (ou de solutions);
- ils n'utilisent que les valeurs de la fonction à optimiser, pas sa dérivée, ou une autre connaissance auxiliaire;
- ils utilisent des règles de transition probabilistes et non déterministes.

1.3 Principe de base d'un AG standard

Un AG standard nécessite en premier le codage de l'ensemble des paramètres du problème d'optimisation en une chaîne de longueur finie. Le principe d'un AG est simple, il s'agit de simuler l'évolution d'une population d'individus jusqu'à un critère d'arrêt. On commence

par générer une population initiale d'individus (solutions) de façon aléatoire. Puis, à chaque génération, des individus sont sélectionnés, cette sélection est effectuée à partir d'une fonction objectif appelée fonction d'adaptation. Puis, les opérateurs de croisement et de mutation sont appliqués et une nouvelle population est créée. Ce processus est itéré jusqu'à un critère d'arrêt. Le critère le plus couramment utilisé est le nombre maximal de générations que l'on désire effectuer. La Figure 1.1 présente le principe de l'AG standard.

L'AG débute par la génération d'une population initiale et l'évaluation de la fonction d'adaptation de tous les individus qui composent cette première population. Puis, des individus sont sélectionnés aléatoirement pour la reproduction selon le principe de la survie du plus adapté. Ensuite, des individus « enfants » (ou les descendants) sont générés en appliquant les deux opérateurs génétiques suivants : le croisement et la mutation. Ces enfants sont placés dans une nouvelle population P(t) et vont se substituer, en tout ou en partie, à la population de la génération précédente. De nouvelles populations d'individus vont ensuite se succéder, d'une génération (t) à la génération (t+1), chaque génération représentant une itération jusqu'à l'atteinte du critère d'arrêt. L'AG présenté ci-dessus est dit générationnel car tous les individus enfants générés sont placés dans une population et vont remplacer entièrement la population des individus parents.

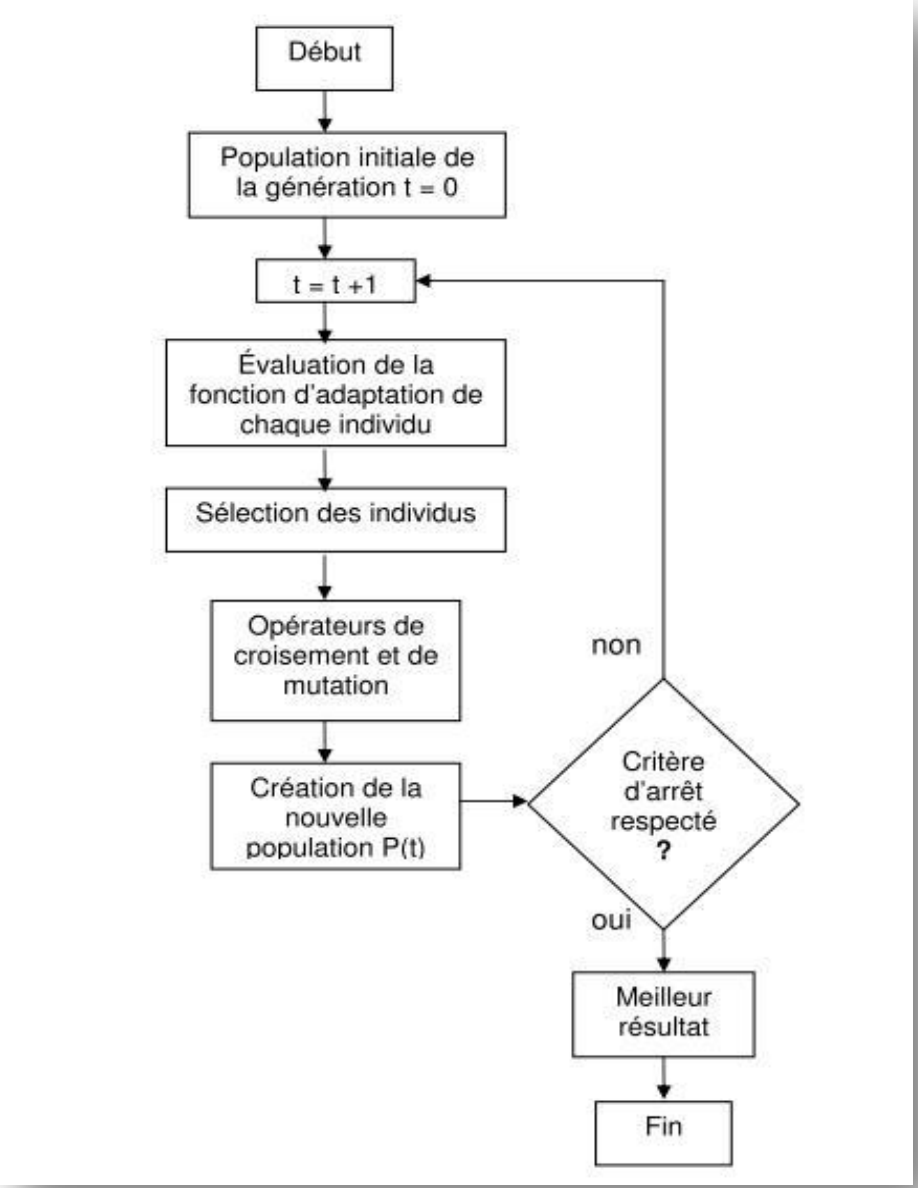

Figure 1.1 : Organigramme d'un AG standard

1.4 Les concepts de base

1.4.1 Les opérateurs génétiques

1.4.1.1 Sélection

La sélection a pour objectif d'identifier les individus qui doivent se reproduire. Cet opérateur ne crée pas de nouveaux individus mais identifie les individus sur la base de leur fonction d'adaptation, les individus les mieux adaptés sont sélectionnés alors que les moins bien adaptés sont écartés [Deb, 2000].La sélection doit favoriser les meilleurs éléments selon

le critère à optimiser (minimiser ou maximiser). Ceci permet de donner aux individus dont la valeur est plus grande une probabilité plus élevée de contribuer à la génération suivante (Figure 1.2). Il existe plusieurs méthodes de sélection, les plus connues étant la « roue de la fortune » et la « sélection par tournoi » :

I. La **« roue de la fortune »** : est la plus ancienne, où chaque individu, de la population de taille maximale Jmax , occupe une section de la roue proportionnellement à sa fonction d'adaptation Fitness (j), la probabilité de sélection d'un individu (j) s'écrit :

$$\mathbf{Prob(j)} = \frac{\mathbf{Fitness(j)}}{\sum_{\mathbf{j=1}}^{\mathbf{Jmax}} \mathbf{Fitness(j)}}$$

A chaque fois qu'un individu doit être sélectionné, un tirage à la loterie s'effectue et propose un candidat, les individus possédant une plus grande fonction d'adaptation ayant plus de chance d'être sélectionnés.

II. A chaque fois qu'il faut sélectionner un individu, la **« sélection par tournoi »** consiste à tirer aléatoirement (k) individus de la population, sans tenir compte de la valeur de leur fonction d'adaptation, et de choisir le meilleur individu parmi les k individus. Le nombre d'individus sélectionnés a une influence sur la pression de sélection, lorsque k = 2, la sélection est dite par « tournoi binaire ».

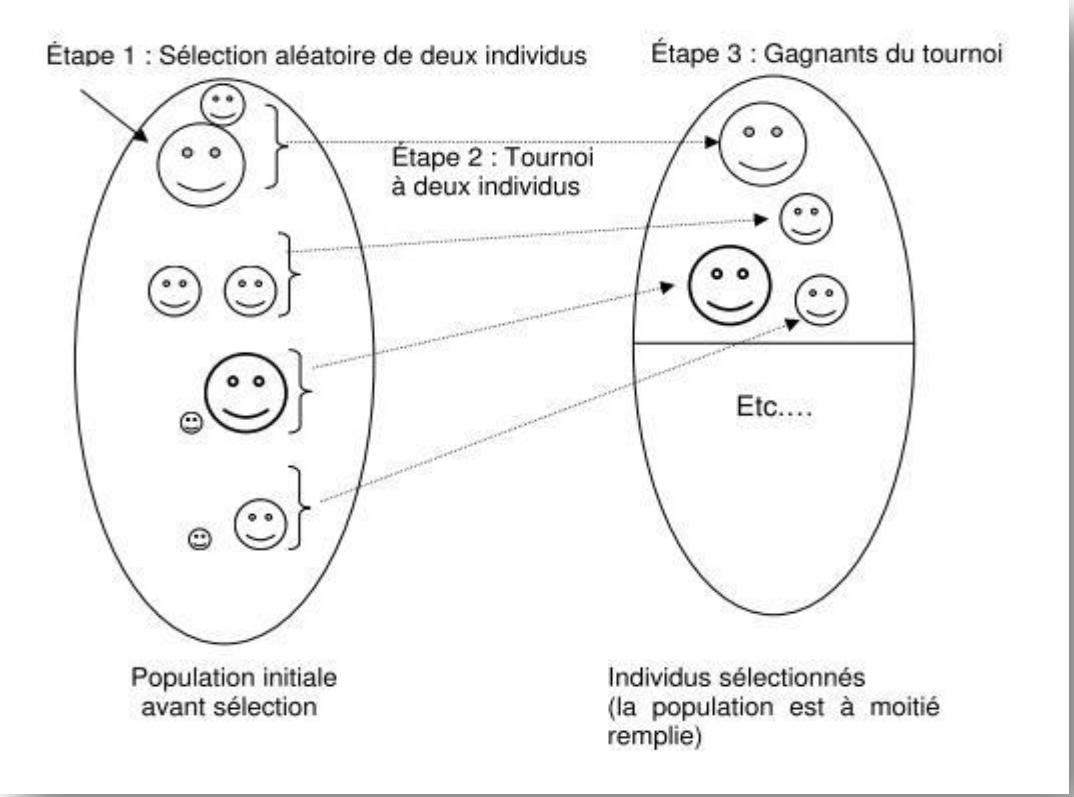

Figure 1.2 : Représentation d'une sélection par tournoi d'individus

1.4.1.2 Croisement

Le croisement permet de créer de nouvelles chaînes en échangeant de l'information entre deux chaînes (Figure 1.3). Le croisement s'effectue en deux étapes. D'abord les nouveaux éléments produits par la reproduction sont appariés, ensuite chaque paire de chaînes subit un croisement comme suit : un entier **k** représentant une position sur la chaîne est choisi aléatoirement entre **1** et la longueur de chaîne (**L**) moins un (**1− L**). Deux nouvelles chaînes sont créées en échangeant tous les caractères compris entre les positions **k +1** et **L**inclusivement. L'exemple suivant (Figure 1.3) montre deux chaînes (A1 et A2) de longueur L= 5 appartenant à la population initiale. Les deux nouvelles chaînes (A3 et A4) appartenant à la nouvelle population sont obtenues par croisement à la position k = 4 :

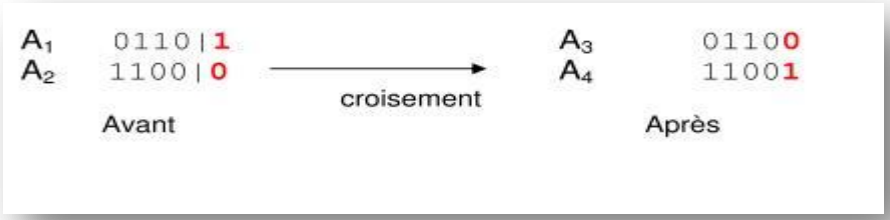

Figure 1.3 : Représentation d'un croisement en un point de deux chaînes

1.4.1.3 Mutation

La mutation est exécutée seulement sur une seule chaîne. Elle représente la modification aléatoire et occasionnelle de faible probabilité de la valeur d'un caractère de la chaîne, pour un codage binaire cela revient à changer un 1 en 0 et vice versa (Figure 1.4). Cet opérateur introduit de la diversité dans le processus de recherche des solutions et peut aider l'AG à ne pas stagner dans un optimum local.

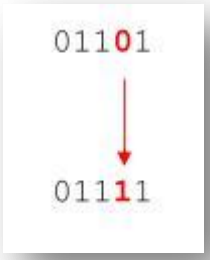

Figure 1.4 : Représentation d'une mutation de bits dans une chaîne

1.4.2 Codage

La manière de représenter un point dans l'espace d'état (l'espace de chromosomes) est très importante. De ce choix découle directement l'espace de recherche, qui selon les circonstances peut être petit ou de très grande taille, ce qui va jouer directement sur le type de solutions trouvées ainsi que sur le temps de convergence.

- Un individu est représenté par son génome (ou génotype).
- Le génotype est le ‘’programme’’ capable de produire un individu.
- Un phénotype est l'ensemble des caractères apparent d'un individu.
- Le choix d'une représentation dépend du:

- Problème à résoudre.
- Comment les individus seront évalués.
- Choix des opérateurs génétiques.

- Il y a souvent plusieurs représentations possibles.

Pour les Algorithmes Génétiques de Holland [Holland,1975],un individu est représenté par une chaîne d'éléments binaires contenant toute l'information nécessaire à la description d'un point (individu) dans l'espace d'état (voir Figure 1.5)

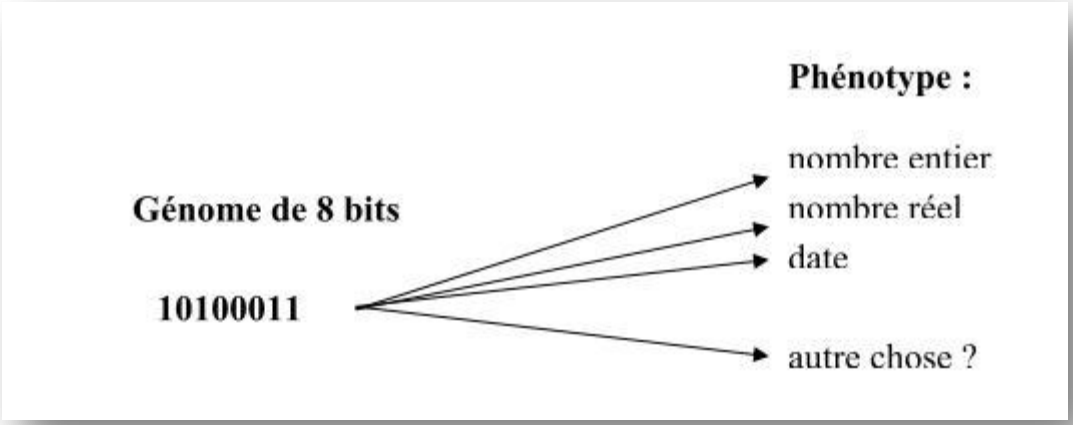

Figure 1.5 : Le codage(exemple de représentation binaire)

1.5 Paramètres d'un AG

Pour appliquer un AG à un problème réel, on doit posséder les éléments suivants :

- un codage des éléments appartenant à la population, le codage des solutions du problème à résoudre doit être choisi avec soin;
- une fonction d'évaluation ou d'adéquation ou d'adaptation de l'individu qui mesure la qualité de l'individu;
- un processus d'évolution des générations;
- des opérateurs pour modifier les individus d'une population de la génération (t) à la génération (t+1) comme le croisement et la mutation;

- des paramètres de l'AG : les opérateurs précédents dépendent de plusieurs paramètres qui sont fixés à l'avance et dont dépend fortement la convergence de l'algorithme :

1. taille de la population : c'est-à-dire le nombre d'individus dans la population. Si la taille est trop petite, l'AG peut ne pas converger, par contre si elle est trop grande, l'évaluation des individus peut être très longue;
2. probabilité de croisement et de mutation. Les valeurs de ces probabilités peuvent varier d'une application à l'autre. Par exemple, dans l'étude des AG's pour l'optimisation de cinq fonctions mathématiques, De Jong (1975) a suggéré de choisir une probabilité de croisement élevée, une probabilité de mutation faible (inversement proportionnelle à la taille de la population), et une population de taille modérée [Goldberg, 1994]. La probabilité de mutation est en général très faible, inférieure à 0,1, une probabilité trop grande, peut modifier les meilleurs individus;
3. critère d'arrêt : différents critères d'arrêt de l'algorithme peuvent être choisis : le nombre maximal de générations à effectuer, l'optimum est atteint, etc.

1.6 Avantages et limites des AG's

Un des grands avantages des algorithmes génétiques est qu'ils autorisent la prise en compte de plusieurs critères simultanément, et qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes. L'avantage principal des algorithmes génétiques par rapport aux autres techniques d'optimisation combinatoire consiste en une combinaison de l'exploration de l'espace de recherche, et de l'exploitation des meilleures solutions disponibles à un moment donné. Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections. Leur utilisation se développe dans des domaines aussi divers que l'économie, la bioinformatique ou la vérification formelle.

L'inconvénient majeur des algorithmes génétiques est le coût d'exécution important par rapport à d'autres méta-heuristiques. Les algorithmes génétiques nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation. Mais avec les capacités calculatoires des ordinateurs récents, ce problème a perdu de son importance. D'un autre côté, l'ajustement d'un algorithme génétique est délicat : des paramètres comme la taille de la population ou le taux de mutation sont parfois difficiles à déterminer.

Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. Néanmoins, la limite la plus importante des algorithmes génétiques est celle des optima locaux. En effet, lorsqu'une population évolue, il se peut que certains individus, qui à un instant occupent une place importante au sein de cette population, deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Il faut mentionner également le caractère indéterministe des algorithmes génétiques. Comme les opérateurs génétiques utilisent des facteurs aléatoires, un algorithme génétique peut se comporter différemment pour des paramètres et populations identiques. Afin d'évaluer correctement l'algorithme, il faut l'exécuter plusieurs fois et analyser statistiquement les résultats.

1.7 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base des algorithmes génétiques .Ils ont prouvé leur efficacité dans plusieurs domaines de recherche tels que la recherche d'informations, le traitement d'images, la classification, etc.

Dans ce travail, nous présentons une approche basée sur ces techniques pour la composition dynamique des services web sémantiques. Dans le chapitre suivant nous allons présenter les concepts liés aux services web ainsi que l'architecture orientée services.

Chapitre2 : Architecture Orientée Services et Services Web

2.1 Introduction

L'évolution des réseaux informatiques combinée à celle des systèmes informatiques a permis la mise en œuvre de systèmes distribués de plus en plus performants et le développement d'applications de toute sorte s'appuyant sur ces infrastructures. Les systèmes informatiques distribués recouvrent un champ très vaste dont font partie les grilles de calcul, le stockage d'informations dans des réseaux de pairs, le déploiement d'applications web, etc

L'évolution de plateformes à composants distribués (.NET de Microsoft) a donné naissance à un nouveau mode de développement logiciel appelé SOC.

SOC, acronyme de Service Oriented Computing, désigne ce paradigme de programmation émergeant, pour le développement de systèmes d'informations distribués, dans lequel les concepts de distribution, d'ouverture, de messagerie asynchrone et de faible couplage tiennent un rôle primordial [Papazoglou et al., 2003],[Singh et al., 2005].

L'avantage de SOC est qu'il permet de créer des systèmes d'information en agrégeant des services individuels. En effet, il permet de créer des systèmes d'information inter- et intra-entreprises de manière relativement aisée en "serviçant" les systèmes existants, d'accomplir interopérabilité et transparence entre ces systèmes au fil de l'eau, de sélectionner dynamiquement des services sur la base de critères de qualité de services personnalisés par chacun et d'utiliser efficacement les ressources de grilles.

Les besoins de SOC pour la satisfaction des précédents cas d'usage peuvent être plus facilement satisfaits à travers une architecture répondant aux propriétés requises. Une telle architecture est appelée SOA ou architecture orientée services. L'objectif de ce chapitre est d'introduire le concept de "Service Web". Nous commencerons par étudier l'architecture SOA, ensuite nous présenterons le modèle des web services au travers des différents standards sous-jacents : SOAP, WSDL et UDDI.

2.2 Architecture Orientées Services

Les architectures orientées services sont le fruit d'une lente maturation technologique s'appuyant sur les travaux précurseurs à la base des notions d'objets et de composants. Une compréhension préalable de ces concepts est nécessaire à l'obtention d'une plus fine appréhension des SOAs.

2.2.1 Concepts de base

2.2.1.1 Objet

La complexification des systèmes logiciels a entraîné dans son sillage l'ingénierie logicielle vers un objectif de rationalisation de la production. Ainsi, de par leur évolution constante et l'augmentation de leur distribution sur le réseau, ces systèmes nécessitent la mise en place de paradigmes permettant d'assurer un certain niveau de qualité et fiabilité, tout en offrant une plus grande flexibilité et réutilisabilité.

L'approche à objets [Jacobson,1991] s'est indubitablement révélée comme un standard de facto parmi les différents paradigmes de développement logiciel. Cette approche a été rendue possible par l'adoption intrinsèque du paradigme objet au cœur de langages tels Smalltalk, Java ou, dans une moindre mesure, C++. L'innovation portée par les objets a consisté à regrouper sous un même concept un ensemble de principes préexistants tels que :

- l'encapsulation, l'interface d'un objet est bien distincte de son implantation, et son comportement exprimée via les méthodes de cette interface ;
- l'héritage, c'est-à-dire de la capacité à définir les caractéristiques d'un objet par extension ;
- le polymorphisme, il est possible d'interagir de la même façon avec des objets de formes différentes.

2.2.1.2 Composant

Par rapport aux objets, le paradigme de composant représente une augmentation du niveau d'abstraction lors de la conception logicielle. En ce sens, le composant ne remplace pas l'objet, il se construit même bien souvent autour, dans les langages de programmation courants (par exemple Java). Il répond d'un besoin concret car, si l'utilisation exclusive d'objets lors de la programmation de systèmes simples semble raisonnable, la programmation d'applications réparties de plus grande ampleur nécessite une augmentation du niveau d'abstraction de leurs briques de base, afin de maintenir leur complexité relativement appréhendable.

Un composant dispose d'interfaces spécifiées à l'aide de contrats, dans l'optique d'une composition ultérieure à sa conception, par des entités tierces. L'implantation de ce

paradigme par de nombreuses technologies telles que les Enterprise Java Beans (EJB) de Sun, COM de Microsoft, ou FRACTAL du consortium OW2, a permis leur large diffusion. Ces technologies déchargent le plus souvent le développeur de la gestion de certaines problématiques de mise en œuvre telles que le support de la sécurité, de la persistance et des transactions.

Ces mécanismes de support à l'exécution des composants seraient difficiles à mettre en place dans des environnements plus faiblement couplés (tels les services, détaillés à la sous-section suivante).La contre-partie est que les composants sont souvent bien trop liés à leurs plateformes de support respectives pour être efficacement interopérables.

2.2.1.3 Service

Dans la lignée de l'approche à objets ou à composants, le paradigme de service tend à fournir un niveau d'abstraction encore plus élevé lors du développement de systèmes informatiques en général, et des applications réparties en particulier. Cette augmentation du niveau d'abstraction passe par une adoption poussée du concept d'encapsulation des fonctionnalités et de celui de la réutilisabilité des services existants. Les services répondent ainsi directement aux problématiques d'intégration logicielle dans les applications réparties modernes.

Tout comme un composant, un service est défini comme étant une unité logicielle autonome. Ce qui est d'autant plus vrai pour les services Web qui sont définis comme étant sans état ("stateless").Les services se distinguent cependant des composants dans la mesure où leur définition met en avant une indépendance poussée par rapport aux plateformes, et une interopérabilité élevée. Voyons maintenant comment faire interagir ces services au sein d'un environnement. On parle alors d'architecture orientée service.

2.2.2 Architecture Orientées Services : définition

Une architecture orientée service est un paradigme fondée sur la description de services et sur la description de leurs interactions [Schulte et Natis, 1996].Les caractéristiques principales d'une architecture orientée service sont le couplage faible entre les services, l'indépendance par rapport aux aspects technologiques et la mise à l'échelle. La propriété de couplage faible implique qu'un service n'appelle pas directement un autre service. En effet, les interactions

sont gérées par une fonction d'orchestration [Collet,2006]. La réutilisation d'un service est alors plus facile, du fait qu'il n'est pas directement lié aux autres services de l'architecture dans laquelle il évolue. L'indépendance par rapport aux aspects technologiques est quant à elle, obtenue grâce aux contrats d'utilisation associés à chaque service. En effet, ces contrats sont indépendants de la plate-forme technique utilisée par le fournisseur du service. Enfin, la mise à l'échelle est rendue possible grâce à la découverte et à l'invocation de nouveaux services lors de l'exécution. Ici les définitions sont nombreuses et nous retiendrons les suivantes :

2.2.2.1 Définition Métier

« L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d'un environnement métier agile pour un intérêt concurrentiel. »[Ben et Joseph, 2007].

2.2.2.2 Définition technique la plus large

« L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoir faires distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs faires pour produire le résultat désiré avec des pré-conditions et des buts mesurables. »[Josuttis, 2007].

2.2.2.3 Définition technique modérément complexe

« L'architecture SOA permet l'intégration d'applications et de ressources de manière flexible, en représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant à un service d'échanger des informations structurées (messages, documents, objets métier), coordonnant et en organisant les services, afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement. »[Hack et al., 2007].

2.2.2.4 Définition Technique Médiane

« Une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant des composants et services

sécurisés, standardisés et qui peuvent être combinés pour s'adresser aux priorités de changements métiers. »[Jennings, 2007].

2.2.3 Pile architecturale des SOA

Cette architecture est composée d'éléments pouvant être classés par catégorie fonctionnelle et catégorie relative à la qualité du service. La Figure 2.1 explicite la pile architecturale et les éléments qui pourraient être observés dans une architecture orientée services.

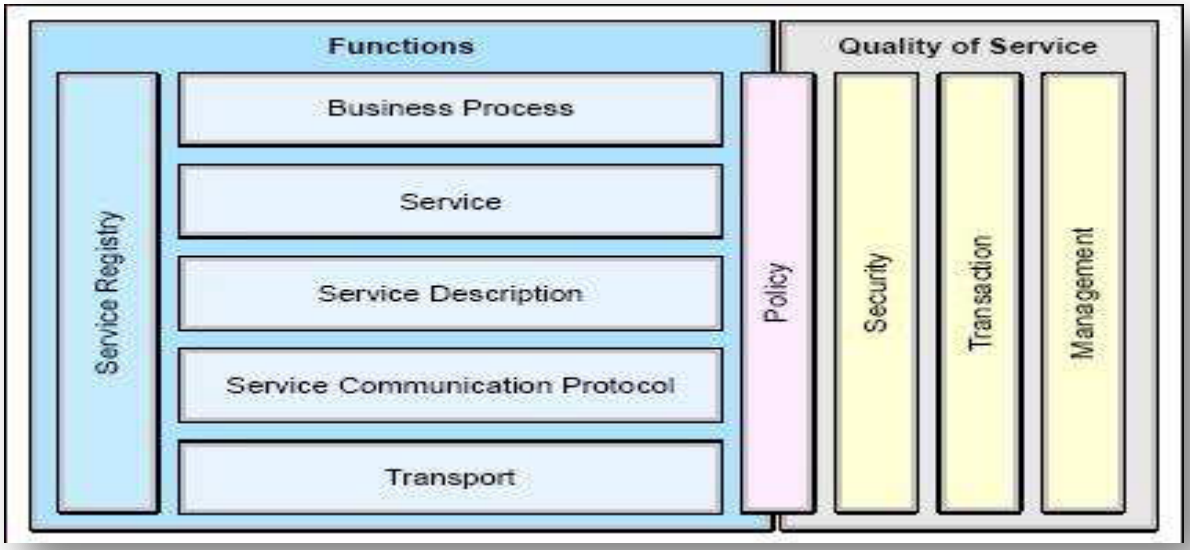

Figure 2.1 : Les différents éléments d'une architecture orientée services

La pile architecturale est donc divisée en deux parties distinctes, la partie de gauche de la Figure 2.1 adressant les aspects fonctionnels de l'architecture et la partie de droite de la figure adressant la qualité portant sur certains aspects du service. Ces éléments sont décrits en détail comme suit :

Les aspects fonctionnels incluent :

- La couche Transport : elle représente le mécanisme employé pour déplacer les demandes de services du consommateur au fournisseur, et les réponses du fournisseur de services au consommateur de services.

- Le Protocole de communication de services : il s'identifie comme un mécanisme convenu par le fournisseur de services et le consommateur de services pour communiquer ce qui est demandé et ce qui sera retourné.
- La couche de Description de Services est un schéma convenu pour décrire ce qu'est le service, comment il devrait être appelé, et quels sont les besoins requis permettant d'appeler le service avec succès.
- La couche Service décrit un service réel qui est rendu disponible pour l'usage.
- La couche faisant référence au Processus d'Affaires est une collection de services, appelée dans un ordre particulier avec un ensemble de règles particulières, dans le but de répondre à une exigence précise. Notons qu'un processus d'affaires pourrait être considéré comme un service unique, ce qui nous pousse à penser que les processus d'affaires peuvent être composés de services de différentes granularités.
- L'Annuaire de services est un entrepôt de descriptions de services et de données qui peuvent être employées par des fournisseurs de services permettant ainsi d'éditer leurs services, et par des consommateurs de services afin de découvrir ou trouver des services disponibles.

Les aspects de qualité de services incluent :

- Une Politique, qui est un ensemble de conditions ou de règles dans lesquelles un fournisseur de services rend le service disponible aux consommateurs. Il y a des aspects de la politique qui sont fonctionnels, et des aspects liés à la qualité du service ;
- La Sécurité, qui est l'ensemble des règles pouvant être appliquées à l'identification, à l'autorisation, et au contrôle d'accès des consommateurs de services.
- La Transaction qui est l'ensemble d'attributs pouvant être appliqués à un groupe de services pour fournir un résultat cohérent. Par exemple, si un groupe de trois services doivent être employés pour accomplir une fonction précise, tous doivent s'exécuter.
- La couche Gestion est l'ensemble d'attributs qui pourraient être appliqués pour contrôler les services fournis ou consommés.

2.2.4 Collaboration dans une architecture orientée services

La Figure 2.2 décrit les collaborations présentes dans une architecture orientée services. Les collaborations suivent les paradigmes de “Recherche”, de “Liaison/Communication” et d’ “Invocation” ou un consommateur de services retrouve dynamiquement la localisation du service en questionnant l’annuaire de services afin de retrouver un service correspondant à ses critères. Si le service existe, l’annuaire de services fournit au consommateur le contrat d’interface et l’adresse du service.Le diagramme suivant illustre les différentes entités présentes dans une architecture orientée services collaborant dans le but de permettre les paradigmes de “Recherche”, “Liaison/Communication” et “Invocation”.

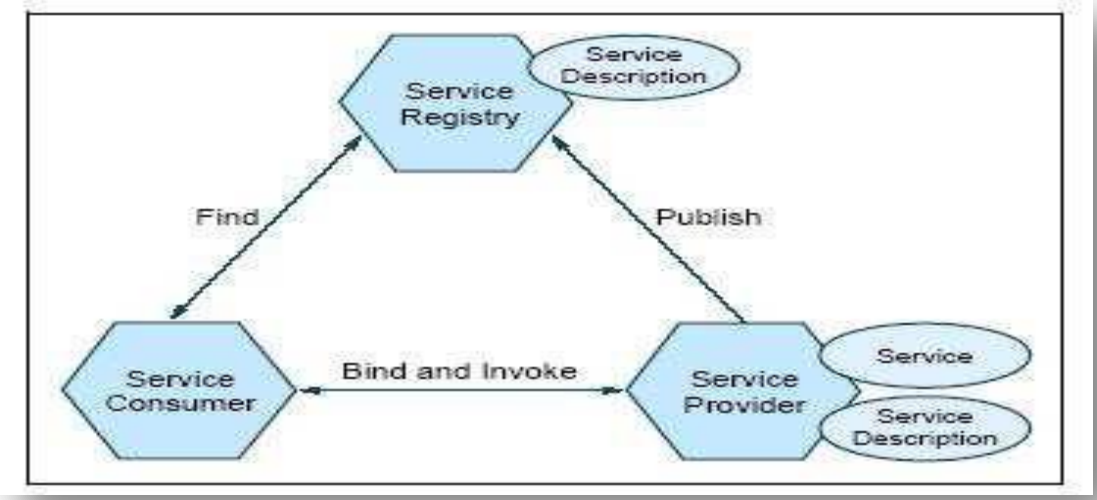

Figure 2.2 : Collaborations dans une architecture orientée services

2.2.5 Les rôles dans une architecture orientée services

Les différents rôles dans une architecture orientée services sont les suivants :

- Le consommateur de services est une application, un module logiciel ou un autre service exigeant un service. Il poste une requête à l’annuaire de services, et communique avec le service au moyen de la couche transport, puis exécute la fonction du service. Le consommateur de services exécute le service selon le contrat d’interface prédéfini.

- Le fournisseur de services est une entité accessible via le réseau acceptant et exécutant des demandes de consommateurs. Il édite ses services et contrats d'interface à l'annuaire de services de sorte que le consommateur de services puisse découvrir et accéder au service.
- Un annuaire de services est un élément essentiel pour la découverte de services. Il contient un répertoire de services disponibles et permet la consultation des interfaces de services par les consommateurs intéressés.

Chaque entité présente dans une architecture orientée services peut jouer un (ou plusieurs) des trois rôles décrits ci-dessus.

Les opérations possibles dans une architecture orientée services sont :

- **Edition/Publication :** pour être accessible, une description de service doit être publiée de sorte qu'elle puisse être découverte et appelée par un consommateur de services.
- **Découverte :** un demandeur de services localise un service via une requête à l'annuaire de services portant sur les critères du service recherché.
- **Liaison/Invocation :** après recherche de la description de service, le consommateur de services procède par appel du service selon les informations relatives à la description du service.

Les objets clefs dans une architecture orientée services sont :

- **Les services :** un service rendu disponible via la publication de la description de son interface lui permet d'être appelé par un consommateur de services.
- **La description des services :** une description de service indique la manière dont le consommateur de services agira avec le fournisseur de services. Elle spécifie le format de la demande et de la réponse du service. Cette description peut indiquer un ensemble de conditions préalables, de post-conditions et/ou de qualité de service (QoS).

2.3 Les services web

L'adoption des SOA dans le monde informatique a été renforcée par plusieurs implantations. Dans le cadre de ce mémoire nous nous intéressons aux services Web, qui représentent l'implantation la plus largement répandue.

2.3.1 Définition

Un service web est un composant logiciel indépendant, qui rassemble un ensemble de standards web et de langages dérivés du langage XML, et qui permettent sa publication, sa découverte et enfin son invocation à distance. Il expose des fonctionnalités via une interface publique et permet de communiquer avec les autres applications et services web en utilisant des messages XML transportés par des protocoles internet comme le HTTP.

Définition d'IBM

« Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer »[Ponge, 2004].

Définition de W3C

« Un service web est un système logiciel identifié par un URI dont les interfaces publiques et les incarnations sont définies et décrites en XML.Sa définition peut être découverte dynamiquement par d'autres systèmes logiciels. Ces derniers peuvent ensuite interagir avec le service web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet » [Ricardo, 2004].

2.3.2 Caractéristiques des services web

Selon [Cerami,2002], plusieurs acteurs définissent les services web par des caractéristiques technologiques distinctives, les plus importantes sont :

- Un service web est une application logicielle qui est reconnue par un URI : URI est la façon d'identifier un point de contenu sur le web comme un document tel qu'un texte, audio ou vidéo. L'URI le plus connu est l'adresse d'une page web, le service web est donc accessible en spécifiant son URI, c'est-à-dire que le service web est caractérisé par un seul objet et une seule fonctionnalité, c'est un prolongement de la programmation orientée objet et à partir de cela, on peut faire la construction d'une application logicielle très large comportant plusieurs fonctionnalités, afin de sélectionner les fonctionnalités qui sont recherchées par les URI spécifiques.
- Capacité des interfaces et liaisons (bindings) d'être publiées, localisées et invoquées via le langage XML : les principales tâches d'un service web sont : la publication dans un registre, la localisation en interrogeant ce registre qui l'héberge et l'invocation par un ou plusieurs web services après sa localisation. Ces tâches sont réalisées en utilisant le langage XML.
- Capacité d'interagir avec les composantes des logiciels via des éléments XML avec l'utilisation des protocoles Internet standards : un service web est créé pour être interrogé par d'autres logiciels contrairement à une page web, ou à une autre application qui n'utilise pas les services web. L'interopérabilité est basée sur l'utilisation du XML et des protocoles Internet standards, tels que, le HTTP qui est le protocole du web, le SMTP qui est le protocole du courrier électronique,…etc.
- Composante logicielle légèrement couplée à interaction dynamique : un service web avec un programme qui l'invoque est appelé le consommateur de service web, et qui sont indépendants l'un de l'autre. Si une modification est à faire sur le consommateur, on n'a pas besoin de connaître la machine, le langage de programmation, le système d'exploitation ou autres paramètres, afin d'établir à nouveau une communication entre le service web et son consommateur. Le consommateur possède une fonctionnalité qui consiste à faire une localisation et une invocation du service web.

2.3.3 Architecture des services Web

L'interopérabilité –aptitude de deux ou plusieurs systèmes à fonctionner ensemble en utilisant des standards communs– est l'objectif premier des services web. Pour permettre cet échange d'information entre des applications distantes, indépendamment des systèmes d'exploitation et des langages de programmation, les services web sont composés de couches standards .Nous mettons en relief dans cette section deux types d'architectures. La première est

l'architecture dite de référence et traditionnellement utilisée pour les services web isolés. La seconde architecture est plus complète et est fréquemment utilisé lors de composition de services web. Elle est appelée architecture étendue ou encore en pile.

2.3.3.1 Architecture de référence

Cette architecture vise trois objectifs importants [kreger, 2001] :

1) identification des composants fonctionnels,
2) définition des relations entre ces composants et
3) établissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

L'architecture de référence des services web (Figure 2.3) s'articule autour des trois rôles suivants :

- **Le fournisseur de service :** correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.
- **Le client :** correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un service web.
- **L'annuaire des services :** correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de recherche et de liens (Bind) d'opérations. Pour garantir l'interopérabilité de ces trois opérations, des propositions de standards ont été élaborées pour chaque type d'interaction. Nous citons, notamment les standards émergents suivants :

• **SOAP** définit un protocole de transmission de messages basé sur XML.
• **WSDL** introduit une grammaire commune pour la description des services.
• **UDDI** fournit l'infrastructure de base pour la publication et la découverte des services.

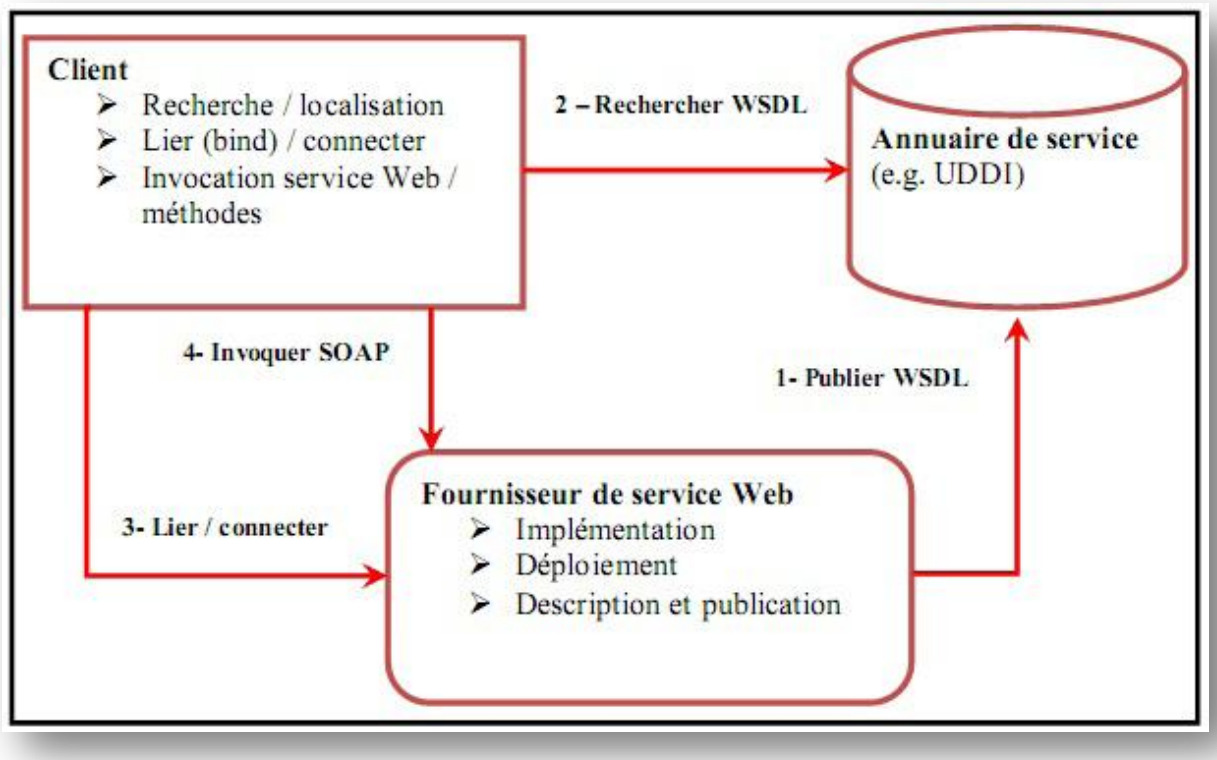

Figure 2.3 : Architecture de référence des services Web

Cependant, cette infrastructure n'est pas suffisante pour permettre une utilisation effective des services web dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards, par exemple, dans le domaine du e-business, d'ou la nécessité d'introduire une nouvelle architecture suffisante et complète.

2.3.3.2 Architecture étendue

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des services web. La figure 2.4 décrit un exemple d'une telle pile. La pile est constituée de plusieurs couches, chaque couche s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment.

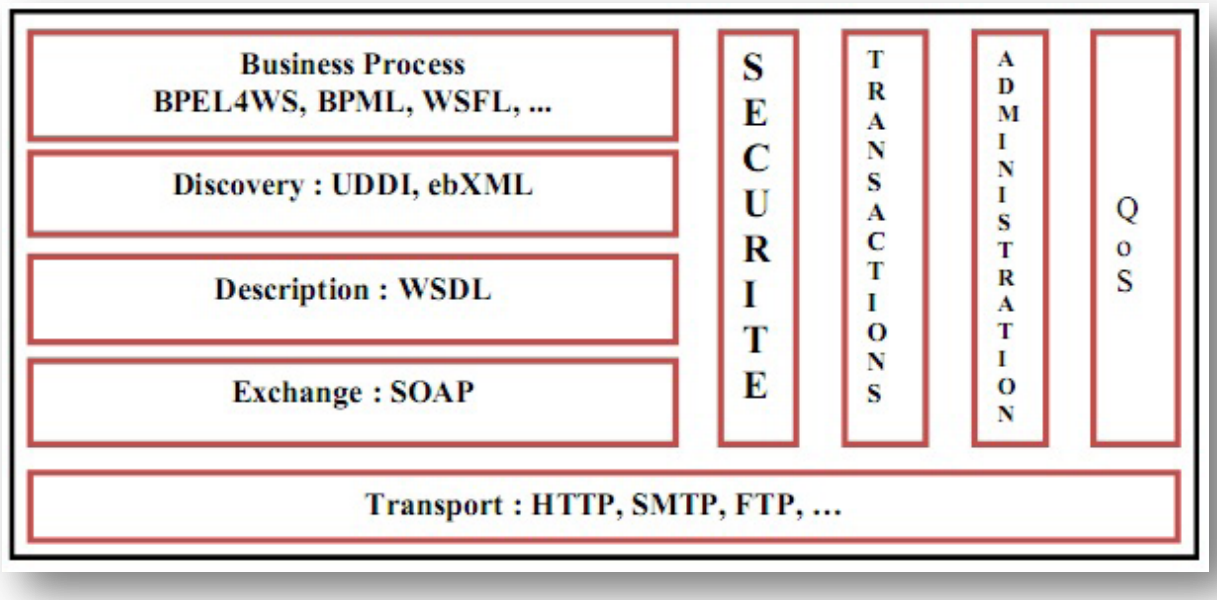

Figure 2.4 : Architecture en Pile des services Web

Nous apportons une explication de la mise en relief des trois types de couches (Voir la Figure 2.4) :

L'infrastructure de base (Discovery,Discription, Exchange) : ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les échanges des messages établis par SOAP, la description de service par WSDL et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI ;

Couches transversales (Security, Transactions, Administration, QoS) : Ce sont ces couches qui rendent viable l'utilisation effective des services Web dans le monde industriel ;

La couche Business Processus (BusinessProcess) : Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un « BusinessProcess » comme un ensemble de services Web. De plus, la description de l'utilisation de différents services composant ce service est disponible par l'intermédiaire de cette couche.

2.3.4 Cycle de vie d'un service web

L'emploi d'un service web connaît le cycle de vie suivant [Garcia et Henriet, 2004] :

- D'abord, on effectue le déploiement du service web, en fonction de la plateforme.

- Ensuite, on enregistre le service web à l'aide de WSDL (Web Services Description Language), dans l'annuaire UDDI.
- L'étape suivante est la découverte du service web par le client, par l'intermédiaire d'UDDI qui lui donne accès à tous les services web inscrits. Pour ce, on utilise SOAP.
- Enfin, Le client invoque le service web voulu, ce qui termine le cycle de vie de ce service web.

2.3.5 Les langages et protocoles utilisés par les services web

Nous avons choisi de décrire les différentes technologies des services web en nous basant sur leur architecture. Toutes ces technologies sont basées sur XML.

2.3.5.1 Echange avec SOAP-Simple Object Access Protocol [Rampacek, 2006]

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol). Ce protocole est normalisé par le W3C[Mitra, 2002], [Gudgin et al., 2002a], [Gudgin et al., 2002a].

Le protocole SOAP est une surcouche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP, la norme n'impose pas de choix.

Le choix de l'utilisation de protocoles applicatifs, comme par exemple HTTP, est lié aux problèmes d'interconnexion connus des réseaux. Le but des services web étant d'être réutilisables, après publication, à travers tout le réseau Internet, il faut alors donner les moyens de passer les protections telles que les pare-feux (firewalls). Ces derniers autorisent généralement sans aucune restriction le trafic sur les ports liés aux protocoles tels que HTTP, permettant ainsi le passage sans problème à travers les différents réseaux des messages générés par l'utilisation du protocole SOAP.

2.3.5.1.1 Structure d'un message SOAP

La technologie des services web repose principalement sur le protocole SOAP qui est indépendant des langages de programmation ou des systèmes d'exploitation. Le standard SOAP définit trois éléments composants un message : [Gardien, 2002]

L'enveloppe (Enveloppe), l'en-tête du message (Header) et le corps du message (Body) (voir Figure 2.5)

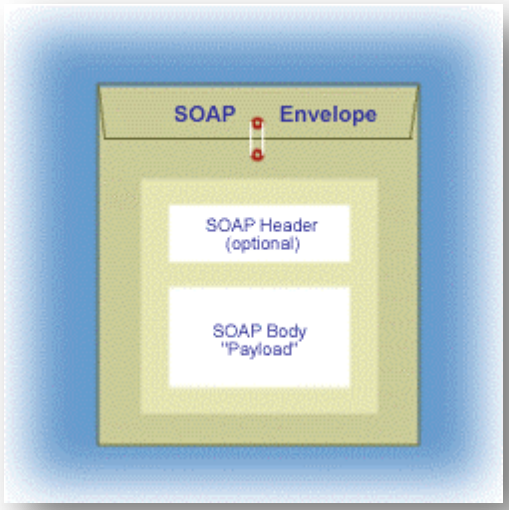

Figure 2.5 : Structure d'un message SOAP

- **Enveloppe SOAP**

L'enveloppe SOAP est l'élément obligatoire d'un message SOAP qui englobe les deux autres parties de ce message (Header et Body). Elle contient le nom du message et le nom du domaine, et permet de préciser la version de SOAP utilisée.

- **L'En-tête SOAP**

L'en-tête est un élément facultatif dans un message SOAP, marqué par la balise <Header>. Il peut avoir plusieurs fils. Ces fils sont utilisés pour ajouter des fonctionnalités au message SOAP comme l'authentification et la gestion des transactions [Ricardo, 2004].

L'en-tête peut utiliser les attributs **mustUnderstand** et/ou **SOAP actor** pour déterminer comment le destinataire d'un message doit traiter ce dernier.

- L'attribut **acteur de SOAP** peut être utilisé pour indiquer le destinataire d'un élément d'en-tête. La valeur de l'attribut d'acteur de SOAP est un URI.
- L'attribut **mustUnderstand** peut être utilisé pour indiquer si une entrée d'en-tête est obligatoire ou facultative pour être traitée par le destinataire. La valeur de l'attribut de mustUnderstand est "1" ou "0". L'absence de cet attribut est sémantiquement équivalente à sa présence avec la valeur "0".

➢ **Le corps SOAP : [Gardien, 2002]**

Le corps du message SOAP est obligatoire, marqué par la balise <Body>. Il permet de transmettre les requêtes et les réponses entre les systèmes, il est composé d'un ou plusieurs sous éléments, qui sont :

- **L'élément FAULT** : il permet d'indiquer les défaillances de transmission des messages SOAP. Il renvoie des informations sur le type d'erreur, une description de l'erreur et l'adresse du serveur SOAP qui a généré l'erreur.
- **L'élément MESSAGE** : il contient les données à transmettre via le protocole SOAP.

2.3.5.1.2 Modèle d'échange de messages en SOAP

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmission d'un message commence (ex. invocation d'un service web), un message SOAP (ou document XML) est généré. Ce message est envoyé à partir d'une entité appelé le SOAP Sender, localisé dans un SOAP nœud. Le message est transmis à zéro ou plusieurs nœuds intermédiaires (SOAP intermédiaires) et le processus fini lorsque le message arrive au SOAP receiver. Le chemin suivi par un message SOAP est nommé message path, [Ricardo, 2004].La Figure 2.6 montre les éléments qui participent au processus.

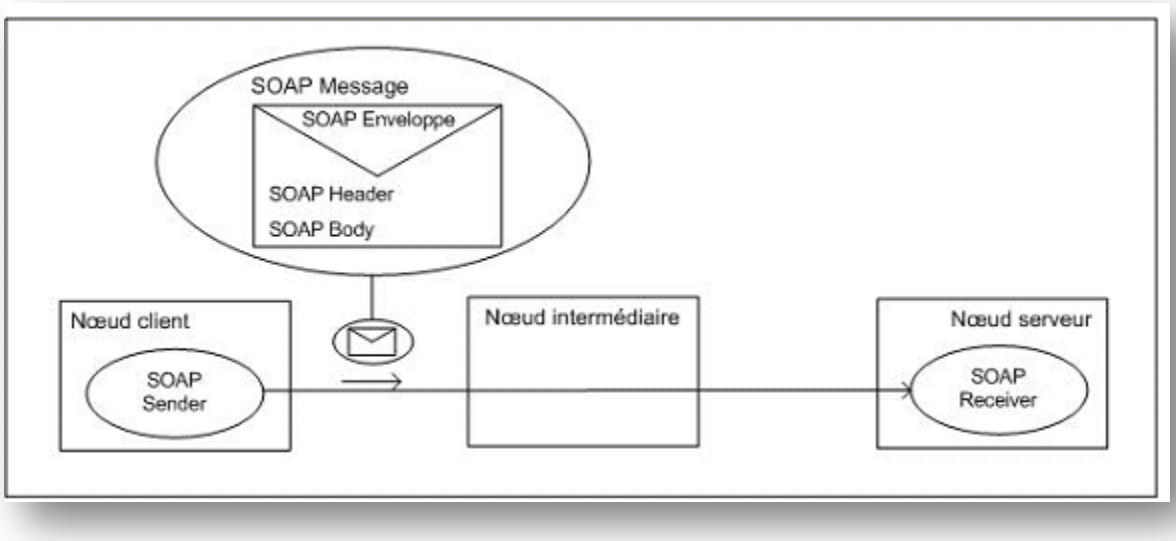

Figure 2.6 : Modèle d'échange de message en SOAP

En résumé, le SOAP est un protocole de communication entre les applications fondé principalement sur le protocole HTTP et sur XML, visant à satisfaire un double objectif : servir de protocole de communication sur Internet, dans une optique d'intégration d'application, et permettre la communication entre les applications et les services web. Il définit un ensemble de règles pour structurer les messages envoyés. Mais SOAP ne fournit aucune instruction sur la façon d'accéder aux services web. C'est le rôle du langage WSDL.

2.3.5.2 Description avec WSDL-Web Services Description Language [Rampacek, 2006]

Le WSDL (Web Services Description Language) est un langage basé sur XML, crée dans le but de fournir une description unifiée des services web. Il se présente comme un standard actuel dans ce domaine, et il est, de plus, normalisé par le W3C. Il est issu d'une fusion des langages de descriptions NASSL (Network Accessibility Service Specification Language - IBM) et SCL (Service Conversation Langage - Microsoft).Son objectif principal est de séparer la description abstraite du service de son implémentation même.

2.3.5.2.1 La structure d'un document WSDL [Ricardo, 2004]

Le langage de description WSDL se comporte donc comme un langage permettant de décrire l'interface visible (ou publiée) du service web. Il décrit, à l'aide du langage de balises XML,

les différents éléments du service,[Rampacek, 2006]. Une description WSDL d'un service web est faite sur deux niveaux, un niveau abstrait et un niveau concret.

Au niveau abstrait, la description du service web consiste à définir les éléments de l'interface du service web tel que les types de données (Data Types), les messages (Message), les types de ports (Port Type). Ces parties décrivent des informations abstraites indépendantes au contexte de mise en œuvre. On y trouve les types de données envoyées et reçues, les opérations utilisables et le protocole qui sera utilisé.

- ✓ **Data types** : est l'élément qui définit les types de données utilisées dans les messages échangés par le service web.
- ✓ **Message** : spécifie les types d'opérations supportées par le service web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.
- ✓ **Port Type** : est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogique à une définition d'un objet contenant un ensemble de méthodes.

Au niveau concret, le service web est défini grâce aux éléments Bindings et Service & Port. Ces deux derniers décrivent des informations liées à un usage contextuel du service web. On y trouve l'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

- ✓ **Bindings** : Décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (SOAP par exemple). Cette description est faite par un ensemble donné d'opérations abstraites. Pour un type de port, on peut avoir plusieurs liaisons, pour différencier le mode d'invocation ou de transport de différentes opérations.
- ✓ **Port** : Spécifie une adresse URL qui correspond à l'implémentation du service web par un fournisseur et identifie une ou plusieurs liaisons aux protocoles de transport pour un Port Type donné.
- ✓ **Service** : Spécifie l'adresse complète du service web, et permet à un point d'accès d'une application distante de choisir à exposer de multiples catégories d'opérations pour divers types d'interactions.

En résumé le document WSDL est indispensable au déploiement de services web et décrit deux documents essentiels : un document pour l'interface du service web et un autre pour son implémentation. La publication et la recherche d'un service web au sein de l'annuaire UDDI se font via ces deux types de documents WSDL.

2.3.5.3 Recherche avec UDDI- Universal Discovery Description and Integration

Un service web doit être référencé afin de pouvoir être retrouvé et utilisé par une autre organisation. Pour cela, il existe des annuaires pouvant être soit internes à l'organisation, soit universels. Il existe de nombreux annuaires, mais nous choisissons de décrire seulement le registre UDDI, annuaire dit universel.

L'UDDI définit les mécanismes permettant de répertorier des services Web. Ce standard gère donc, l'information relative à la publication, la découverte et l'utilisation d'un service Web. En clair, l'UDDI définit un registre des services Web sous un format XML. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le service Web qui fourni le service désiré, pour une éventuelle interaction ; ainsi l'UDDI a été créé pour faciliter la découverte de services Web en plus de leurs publications. Par une API SOAP, on peut interagir avec l'UDDI au moment de la conception et l'exécution des applications afin de découvrir des données techniques, et administratives sur les entreprises et leurs services Web.

L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP [Scott, 2002] [Gardien, 2002]. L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte.

La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP. L'UDDI peut être vu comme un annuaire contenant [Newcomere, 2004] :

Les pages blanches : noms, adresses, identifiants et contacts des entreprises enregistrées. Ces informations sont décrites dans des entités de type « BusinessEntity ». Cette description inclut des informations de catégorisation permettant de faire des recherches spécifiques dépendant du métier de l'entreprise.

Les pages jaunes : donnent les détails sur le métier des entreprises et les services qu'elles proposent.Ces informations sont décrites dans des entités de type « BusinessService ».

Les pages vertes: contiennent des informations techniques du service offert, la manière d'interagiravec le service, une définition du « BusinessProcess »et aussi un pointeur vers le fichier WSDL etune clé unique identifiant le service.

La structure de données du registre UDDI contient cinq types de données :

BusinessEntity : Les informations concernant l'entreprise qui publie ses services Web dans l'annuaire et le type de services qu'elle offre sont contenues dans la structure« BusinessEntity »et correspondent notamment aux pages blanches concernant l'entreprise.

BusinesService : L'élément « BusinessService »contient des informations sur les services métiers. Il s'agit des informations de description des services web référencés : nom du service, sa description, son code. Une entreprise peut enregistrer plusieurs services. Le type d'informations contenu dans l'élément « BusinessService »correspond aux informations pages jaunes d'une entreprise.

BindingTemplete (informations de liaison) : Les informations de liaison contiennent des informations techniques sur un service Web. Elles servent également de pages vertes de l'annuaire UDDI. Il s'agit des informations indiquant les références aux « tModels »désignant les spécifications de l'interface pour un service Web, ainsi le point de terminaison (adresse Internet) de ce dernier. Ces informations aident le client à se connecter puis à invoquer le service désiré. Un service peut avoir plus d'un modèle de liaison.

tModel (informations techniques) : La structure « tModèle »a pour rôle de décrire les spécifications des services Web à enregistrer. Elle comporte les informations permettant de connaître les normes auxquelles le service est conforme afin d'avoir une description

suffisamment précise du service. Le nom doit présenter suivant le format URI et doit pointer vers l'emplacement du fichier correspondant, généralement au format WSDL.

PublishAssertion (Assertion contractuelles entre partenaires) : met en correspondance deux ou plusieurs structures « BusinessEntity »selon le type de relation (filiale de, département de).Cette structure comporte les assertions contractuelles entre organismes pour les services publiés. Ces assertions représentent un ensemble de règles contractuelles d'invocation de services représentées sous la forme de protocoles entre deux partenaires métiers. Chaque rôle entre partenaires est défini à travers ces assertions.

En résumé, UDDI est un standard pour faciliter la collaboration entre partenaires dans le cadre d'échanges commerciaux, le cœur d'UDDI est un annuaire qui contient des informations techniques et administratives sur les entreprises et les services Web qu'elles publient. Donc l'annuaire UDDI permet de publier et découvrir des informations qui concernent une entreprise et ses services Web.

2.3.6 Avantages et inconvénients des services web

2.3.6.1 Avantages

La technologie des services web est populaire et couramment utilisée car elle offre des avantages intéressants pour les utilisateurs des systèmes distribués:

- Les services web réduisent le temps de mise en marché des services offerts par les diverses entreprises.
- Les services web permettent à des programmes écrits en des langages différents et sur des plateformes différentes de communiquer entre eux par le biais de certaines normes. En d'autres termes, les services web permettent une meilleure interopérabilité entre les logiciels.
- Les services web utilisent des normes et protocoles ouverts.
- Grâce au protocole HTTP, les services web peuvent fonctionner malgré les pare-feu sans pour autant nécessiter des changements sur les critères de filtrage.
- Les protocoles et les formats de données sont offerts, le plus possible, en format texte pour que la compréhension du fonctionnement des échanges soit plus intuitive.
- Grâce aux services web, les coûts sont réduits par l'automatisation interne et externe des processus commerciaux.

2.3.6.2 Inconvénients

La technologie des services web comporte plusieurs inconvénients dont:

- Problèmes de sécurité: Il est facile de contourner les mesures de sécurité mises en place par les pare-feu -l'utilisation du protocole HTTP (tel que mentionné ci-haut) n'a pas que des avantages -car les normes de sécurité des services web laissent encore à désirer. CORBA, par exemple, qui est une technologie plus mûre, est plus sécuritaire.
- Problèmes de performance: Les services web sont encore relativement faibles par rapport à d'autres approches de l'informatique répartie telles que CORBA ou RMI.
- Confiance: Les relations de confiance entre différentes composantes d'un service web sont difficiles à bâtir, puisque parfois ces mêmes composantes ne se connaissent même pas.
- Syntaxe et sémantique: On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (sémantique).
- Fiabilité: Il est difficile de s'assurer de la fiabilité d'un service car on ne peut garantir que ses fournisseurs ainsi que les personnes qui l'invoquent travaillent d'une façon fiable.
- Disponibilité: Les services web peuvent bien satisfaire un ou plusieurs besoins du client. Seront-ils pour autant toujours disponibles et utilisables? Ça reste un défi pour les services web.

2.4 Conclusion

La technologie des services Web offre de fortes potentialités pour surmonter les problèmes d'interopérabilité des systèmes. Elle constitue un cadre prometteur pour l'intégration des applications, et pour la gestion des interactions entre divers partenaires dans un environnement distribué, hétérogène, ouvert et versatile qui est le Web.

La plupart des travaux existants qui s'intéressent à l'intégration fonctionnelle évitent le problème fondamental de l'automatisation des différentes étapes liées à la fourniture d'un service web (par exemples, découverte et composition) puisqu'ils limitent l'usage des services web aux utilisateurs humains plutôt qu'aux machines. Il semble donc nécessaire de tendre vers des services intelligibles pour des machines : c'est le concept de service web sémantique qui fera l'objet du chapitre suivant.

Chapitre3 : Les Services Web Sémantiques

3.1 Introduction

L'infrastructure de base autour des standards SOAP, WSDL, UDDI est suffisante pour mettre en place des composants interopérables et intégrables mais insuffisante pour rendre automatique et efficace plusieurs tâches liées au cycle de vie des services web, par exemple : la composition et aussi la découverte des services requis. En particulier, WSDL fournit une description concrète mais de bas niveau d'un service Web, en termes de sa localisation, des opérations disponibles et des messages associés ainsi que des types de données et formats de leurs paramètres d'E/S. Ces descriptions sont insuffisantes pour qu'un agent logiciel puisse interpréter la signification réelle des opérations WSDL.

Le besoin d'automatisation du processus de conception et de mise en œuvre des services web rejoint les préoccupations à l'origine du Web sémantique, à savoir comment décrire formellement les connaissances de manière à les rendre exploitables par des machines. En conséquence, les technologies et les outils développés dans le contexte du Web sémantique peuvent certainement compléter la technologie des Web services en vue d'apporter des réponses crédibles au problème de l'automatisation.

Par exemple, la notion d'ontologie peut jouer un rôle prépondérant pour permettre d'expliciter la sémantique des services facilitant ainsi les communications hommes-machines, d'une part, et les communications machines-machines, d'autre part.

Dans ce chapitre nous présentons la notion de « service web sémantique » ainsi que les objectifs visés par ce dernier, ensuite nous abordons les différentes approches qui sont utilisées pour expliciter la sémantique dans les descriptions des services web.

3.2 Présentation et objectifs des services web sémantiques

Le Web sémantique constitue une prolongation, et une sorte de révolution de fond du Web actuel qui permet une définition non ambiguë de l'information, pour favoriser une meilleure coopération entre humain et machine. Il permet de s'ouvrir à de nouvelles possibilités d'automatisation d'une grande quantité d'information sur le Web.

Proclamé technologie du futur, en 2001, par son créateur Tim Berners-Lee, le Web sémantique propose une nouvelle plateforme permettant une gestion plus intelligente du contenu, à travers sa capacité de manipuler les ressources sur la base de leurs sémantiques. En réalité, l'intégration de la sémantique au Web n'est pas une nouvelle idée mais au contraire, elle est née avec le Web [Falquet et Mottaz, 2001].

Le Web sémantique constitue le point de départ pour le développement des services Web intelligents.

En effet, non seulement l'humain pourra partager, échanger et réutiliser la connaissance et l'information qui est disponible sur le Web mais en plus, il pourra le faire plus vite et avec l'aide des machines. HTML a été le langage du Web jusqu'à présent. Il permettait de présenter l'information aux humains. Désormais, il est nécessaire de présenter cette information de façon à ce qu'un programme puisse s'en servir. Le Web sémantique a promit de permettre aux machines de tirer parti du contenu statique du Web, en utilisant les annotations. En effet, la sémantique et la structure des données requièrent une représentation de la sémantique compréhensible et échangeable par les machines [Bruijn et al., 2007].

Le terme services Web sémantique se trouvent à la convergence de deux domaines de recherche importants concernant les technologies de l'Internet; le Web sémantique et les services Web (Voir la Figure 3.1) [Cardoso, 2007].

Cette tâche de convergence est accomplie en rendant les services Web auto-exploitables par machines, et de réaliser l'interopérabilité entre les applications via le Web en vue de rendre le Web plus dynamique.

De la même façon, que le Web sémantique a promis de considérer comme un vaste espace d'échange de ressources entre humains et machines, permettant une meilleure exploitation de masses de données disponibles sur le Web. L'objectif est non pas de permettre aux machines de se comporter comme des êtres humains, mais de développer des langages pour représenter les informations d'une manière traitable, représentable et intelligible par les machines, afin, d'améliorer les rapports des utilisateurs avec le Web. Il semble donc nécessaire de tendre vers des services intelligibles pour des machines, c'est le concept de service Web sémantique [Kadima et Monfort, 2003].

Les avantages potentiels des services Web sémantiques ont mené à l'établissement d'un domaine de recherche important, dans le milieu industriel et académique. Plusieurs, initiatives sont apparues pour faire ce qu'on appelle l'annotation sémantique des services Web, ce qui a produit une variété de descriptions des services Web et leurs aspects relatifs, ce qui en retour a abouti à de divers genres de support pour la découverte et la composition.

Le concept fondamental du Web sémantique et des services Web sémantique est l'ontologie, qui produit une signification bien définie des informations contenu dans le Web. Une ontologie représente donc un schéma conceptuel, qui tente de désigner une description rigoureuse et exhaustive d'un domaine. Habituellement, une ontologie est une structure de données hiérarchique qui comprend toutes les entités du domaine que l'on tente de décrire, ainsi que, les relations sémantiques qui existent entre ces différentes entités. Mais attention, une ontologie se doit d'être plus qu'une simple taxonomie [Bruijn et al., 2007].

De manière générale, l'objectif visé par la notion de services Web sémantiques est de créer un Web sémantique de services dont les propriétés, les capacités, les interfaces et les effets sont décrits de manière non ambiguë et exploitable par des machines, et ce en utilisant les couches techniques sans pour autant en être conceptuellement dépendants[Daconta et al., 2003]. La sémantique ainsi exprimée permettra l'automatisation des fonctionnalités suivantes qui sont nécessaires pour une collaboration inter-entreprises efficace :

- Processus de description et de publication des services ;
- Découverte des services ;
- Sélection des services ;
- Invocation des services ;
- Composition des services ;
- Surveillance automatique de l'exécution des services web (monitoring) ;

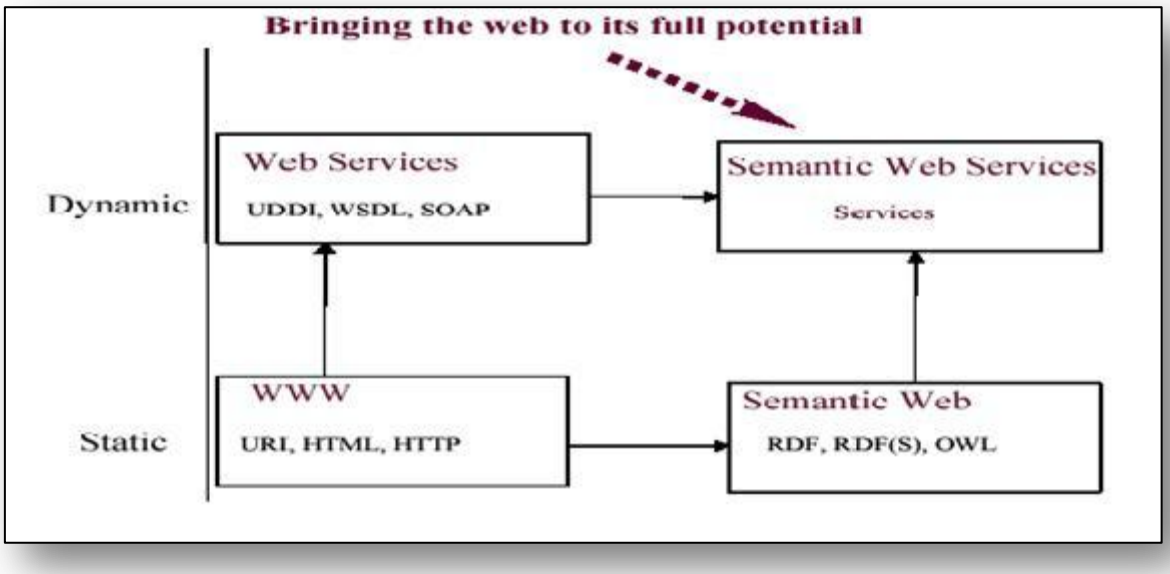

Figure 3.1 : L'évolution du web

3.3 Classification et présentation des approches

La réalisation des conditions qui élèvent les services Web au rang de services Web sémantiques peut suivre deux approches :

La première approche consiste à développer un langage complet qui décrit les services Web ainsi que leur sémantique d'un seul bloc.

La deuxième approche consiste à annoter les langages existants avec de l'information sémantique. L'avantage principal de ce genre de solutions réside dans la facilité pour les fournisseurs de services d'adapter leurs descriptions existantes aux annotations proposées.

Nous classions donc ces approches de la manière suivante : dans un premier temps, nous étudions les langages de description sémantique, puis dans un second temps nous détaillons les annotations de langages existants.

3.3.1 Langages de description sémantique

3.3.1.1 OWL-S Ontology Web Language for Services

OWL-S désigné par DAML-S dans les versions antérieures [Martin et al., 2004], est une ontologie de description des services web, dont les objectifs sont de résoudre les ambiguïtés et de rendre la description d'un service compréhensible par une machine. Dans [Ankolekar et al., 2002] les auteurs présentent une ontologie pour les services web dans le but d'automatiser

la découverte, l'invocation, la composition et la surveillance de l'exécution des services. Les auteurs reprennent la notion de classes d'OWL et proposent l'ontologie OWL-S.

OWL-S fournit trois connaissances essentielles sur les services Web, qui répondent aux questions suivantes :

- que fournit le service Web pour l'utilisateur potentiel ? La réponse à cette question est fournie par le **ServiceProfile**,
- comment fonctionne le service Web ? La réponse à cette question est fournie par le **ServiceModel**,
- comment le client peut-il interagir avec le service Web ? La réponse à cette question est fournie par le **ServiceGrounding**.

OWL-S [Solanki et al., 2004][OWL-S, 2005]décrit donc les services Web suivant trois points de vue différents qui sont le ServiceProfile, le ServiceModel et le ServiceGrounding représentés dans la Figure 3.2.

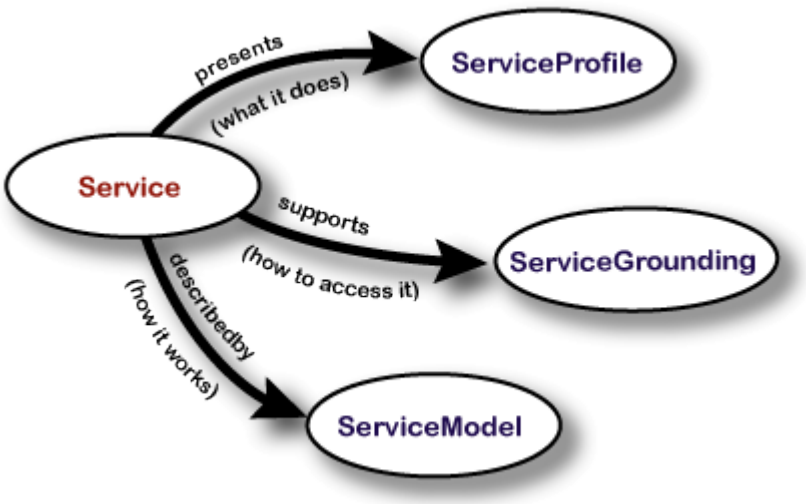

Figure 3.2 : Niveau supérieur de l'ontologie de OWL-S

- **ServiceProfile**

Pour d'écrire un service OWL-S définit la classe ServiceProfile. La classe ServiceProfile spécifie trois informations :

- Nom du service, contacts et description textuelle du service : le nom du service est utilisé comme identificateur du service, tandis que les informations contacts et la description textuelle sont destinées aux utilisateurs humains.
- Description fonctionnelle du service : Elle spécifie ce que le service exige en terme d'entrées (inputs) attendues et de résultats produits en sortie (outputs).Elle indique également les pré-conditions et les effets du service.
- Classification taxinomique.

➢ **ServiceModel**

La classe ServiceModel décrit le fonctionnement du service Web. Ceci est fait en exprimant les transformations faites par le service Web sur les données (input à output), et transformation d'état (pré-conditions et effets).

Les services Web peuvent être modélisés avec OWL-S en tant que processus grâce à la classe Process. La classe ainsi définie est une sous-classe de ServiceModel. Pour décrire un processus, on spécifie ces Entrées, Sorties et ses états. Les transitions d'un état à un autre sont décrites par les pré-conditions et les effets de chaque processus.

Il existe trois types de processus

1. Les processus atomiques (AtomicProcess) : exécutable en une seule étape, un processus atomique ne peut pas être décomposé de façon plus profonde. Son exécution correspond à une unique avancée dans l'exécution du service, il est directement invoqué par l'utilisateur du service.
2. Les processus composites (CompositeProcess) : un processus composite est constitué par l'assemblage d'autres processus (composite ou non composite). Les processus composites associent des processus à l'aide de structures de contrôle permettant de décrire leur logique d'exécution. Les structures de contrôles sont les suivantes :

- séquence (Séquence) représente une suite ordonnée de processus.
- exécutions concurrentes de processus sont décrites par Split.
- synchronisation peut être décrite par Split+Join.
- exécution de processus sans ordre particulier avec Unordered.
- le choix est décrit par Choice.
- les branchements conditionnels du type si/alors/sinon sont décrits par If -Then-Else.
- Repeat, Iterate et Repeat-Until permettent d'effectuer des itérations.

3. Les processus simples (SimpleProcess) : Les processus simples ne sont pas invocables mais comme les processus atomiques, leurs exécutions s'effectuent en une seule étape. Les processus simples sont employés comme éléments d'abstraction, un processus simple peut être employé pour fournir une vue d'un certain processus atomique, ou une représentation simplifiée d'un certain processus composé.

- **ServiceGrounding**

La classe OWL-S ServiceGrounding définit les détails techniques permettant d'accéder au service Web. Les deux premières classes ServiceProfile et ServiceModel d'une description OWL-S s'attachent à abstraire la représentation d'un service Web. ServiceGrounding est la forme concrète d'une représentation abstraite, elle fournit les détails concrets d'accès au service Web, tels les protocoles, les URIs, les messages envoyés, etc.

Les concepteurs de OWL-S ont choisi de combiner OWL-S avec WSDL comme le montre la Figure 3.3. L'objectif de cette combinaison est de tirer profit de chacun d'entre eux. Les types abstraits des messages employés dans les messages WSDL sont définis sémantiquement à l'aide de classes OWL. Les bindings WSDL servent ensuite à définir comment ces messages sont formatés. Les relations entre OWL-S et WSDL sont les suivantes :

- ✓ Un processus atomique OWL-S correspond à une opération WSDL.
- ✓ Les entrées et sorties d'un processus atomique OWL-S correspondent chacune à un message WSDL (d'entrée ou de sortie).
- ✓ Les types des messages d'un processus atomique OWL-S correspondent à l'élément Types d'une description WSDL.

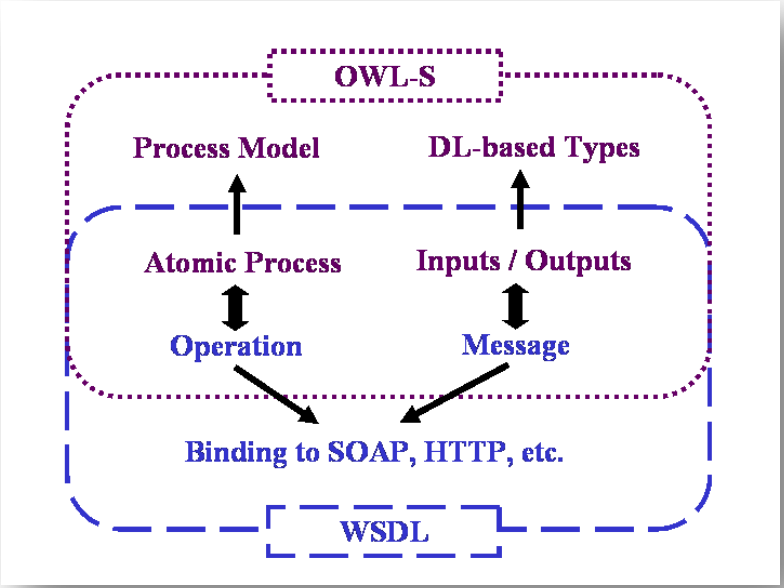

Figure 3.3: Lien entre OWL-S et WSDL

3.3.1.2 IRS-II (Internet Reasoning Service)

IRS-II est une architecture pour les services Web sémantiques. IRS-II est basée sur la structure UPML (Unified Problem Solving Method Development Language), où diverses ontologies sont définies [Gesnu, 2003] [Lopes, 2005] :

- Ontologie du domaine (Domain Model) : Permet de décrire le domaine d'une application;
- Ontologie de tâche à résoudre (Task Models) : Fournit une description générique de la tâche à résoudre, spécifie les types d'entrées et sortie, le but à atteindre et les pré-conditions à satisfaire;
- Ontologie des méthodes de résolution d'un problème (Problem Solving Methods) : sépare la description de ce qu'un service fait des paramètres et des contraintes d'une mise en œuvre particulière;
- Liens (bridges) : Permettent la correspondance entre les différents modèles d'une application.

Les principaux composants de l'architecture IRS-II sont : le serveur IRS-II (IRS-II server), éditeur de services (IRS-II Publisher) et la partie client (IRS-II Client). Ces trois composants interagissent entre eux via le protocole SOAP.

Le serveur IRS-II contient les descriptions des services Web sémantiques. Ces descriptions sont faites sur deux niveaux. Au niveau connaissance, une description est sauvegardée selon la structure UPML des tâches, PSM et l'ontologie du domaine. De plus, deux types de mise en correspondances sont utilisés pour lier les descriptions aux niveaux connaissances à un service Web spécifique.

Le composant éditeur (IRS-II Publisher) a deux fonctions. Premièrement, il permet de lier les services Web à leurs descriptions sémantiques respectives dans le serveur. Chaque PSM est associé à exactement un service Web. Un service Web peut être associé à plusieurs PSM, puisque un service Web peut avoir plus qu'une fonction. Deuxièmement, il génère automatiquement un programme qui enveloppe le code LISP ou Java du service Web, afin de l'invoquer, comme c'est le cas d'un service Web dans sa description WSDL.

Un client IRS-II invoque un service Web en envoyant une requête de la tâche à traiter. Sur la base de cette tâche le serveur sélectionne le PSM approprié, et invoque le service Web avec lequel est lié ce PSM.

3.3.1.3 WSMF (Web Service Modeling Framework)

WSMF [Fensel et al., 2000] est un modèle de représentation des divers aspects relatifs aux services Web. L'objectif principal de cette approche est de permettre le développement du e-commerce par application des technologies du web sémantique aux services Web.
WSMF est basé sur quatre éléments :

- les ontologies qui fournissent la terminologie utilisée par les autres éléments ;
- les répertoires d'objectifs qui définissent les problèmes qui doivent être résolus par les services Web ;
- les descriptions des services Web et un ensemble de
- médiateurs contribuant à outrepasser les problèmes d'interopérabilités.

L'implémentation de WSMF est partagée en deux projets : le projet SWWS (Semantic Web enabled Web Services) [Newcomere, 2004] ; et le projet WSMO (Web Service Modelling Ontology) [Paul et Sandeep, 2004].L'objectif de SWWS est de définir une structure de description et de découverte de services Web, ainsi qu'une plate-forme de médiation pour services selon une architecture conceptuelle. Le projet WSMO permettra de raffiner WSMF en plus du développement d'ontologie formelle de service et un langage pour les services Web sémantiques.

3.3.1.4 WSMO (Web Service Modeling Ontology)

L'ontologie WSMO est présentée par [Roman et al., 2005]. Initiée par l'ESSI WSMO workgroup, elle est basée sur le Web Service Modeling Framework WSMF proposé par [Fensel et Bussler, 2002].

WSMO partage avec OWL-S l'objectif d'automatiser les taches liées aux services.La séparation entre la description des services web sémantiques et les technologies exécutables est un aspect essentiel de WSMO. En effet WSMO fournit un modèle de description d'ontologies indépendant du langage utilisé pour les décrire. Le diagramme de classes UML, illustré par la Figure 3.4, est extrait de [Bruijn et al, 2005] .Il représente les éléments principaux d'une ontologie WSMO et leurs interactions. Le concept central est le **WSMO element** qui se spécialise en une **ontology** ou un **webService** ou un **goal** ou un **mediator**.

- Un élément **ontology** désigne une ontologie de référence importée par un élément de l'ontologie WSMO afin de décrire ses propriétés.
- Un élément **webService** est "une unité de calcul capable de répondre à une requête utilisateur" [Roman et al., 2005].Les webService WSMO sont définis d'une manière uniforme comprenant les éléments suivants : la fonctionnalité du service, les interfaces de description, les propriétés non fonctionnelles, les ontologies importées et les médiateurs utilisés.
    - La fonctionnalité du service, désignée par capability, est décrite en termes d'opérations associées à des pré-conditions, hypothèses (assumptions), post-conditions et effets. Les pré-conditions et les post-conditions décrivent les exigences sur les données avant l'exécution du service et les changements qu'elles subissent après exécution ; les hypothèses et effets décrivent les exigences et les changements relatifs aux services en interaction avec le service décrit.

- Les interfaces de description décrivent le comportement du service, en d'autres termes l'ordonnancement des opérations.
- Les propriétés non fonctionnelles d'un webService décrivent les attributs qui ne se rapportent pas directement aux données traitées, comme par exemple la durée d'exécution d'une opération.
- Les ontologies importées sont utilisées pour référencer les éléments de la description du service.
- Les médiateurs sont utilisés si deux webServices en interaction importent deux ontologies différentes.

- Un élément **goal** sert à décrire les souhaits des utilisateurs (trices) en terme de fonctionnalités requises. Les objectifs sont une vue orientée utilisateur (trice) du processus d'utilisation des services Web, ils sont une entité à part entière dans le modèle WSMO. Un objectif décrit la fonctionnalité, les entrées/sorties, les pré-conditions et post-conditions d'un service Web.
- Un élément **mediator** est utilisé pour résoudre de nombreux problèmes d'incompatibilité, telles que les incompatibilités de données dans le cas ou les services Web utilisent différentes terminologies, les incompatibilités de processus dans le cas de la combinaison de services Web, et les incompatibilités de protocoles lors de l'établissement des communications.

Contrairement à OWL-S, WSMO inclut les médiateurs comme des composants centraux de son architecture. Les hétérogénéités rencontrées lors de l'utilisation de services Web sont gérés par les types de médiation suivants :

- ❖ La médiation de données résout les incompatibilités de représentation des données.
- ❖ La médiation de processus est relative à la logique applicative de la composition.
- ❖ La médiation de protocoles adapte les différents protocoles de communication utilisés.

Ces types de médiation sont pris en charge par quatre familles de médiateurs :

- ❖ Les GG-médiateurs permettent d'effectuer la médiation entre deux objectifs, GG signifiant « goal-goal ». Cela signifie qu'ils permettent d'établir des correspondances entre objectifs, en se servant des ontologies d'objectifs disponibles dans le cadre de WSMO.

- Les WG-médiateurs, avec WG pour « web service-goal », établissent les correspondances entre les fonctionnalités offertes par les services Web et les requêtes des utilisateurs(trices), qui sont toutes les deux définies comme des objectifs. L'intérêt de ce type de médiateurs est d'aider la découverte et la sélection de services Web.

- Les WW-médiateurs, avec WW pour « web service-web service », établissent les correspondances entre services Web. Leur tâche est de résoudre les conflits au niveau des données, du protocole, et du processus de composition. Ils sont mis en œuvre lors de l'orchestration des services Web au sein d'une composition.

- Les OO-médiateurs, avec OO pour « ontology-ontology », sont destinés à résoudre les conflits entre ontologies. Les autres types de médiateurs énoncés ci-dessus, mais aussi n'importe quel élément de l'architecture WSMO qui pourrait utiliser les ontologies, peut utiliser un OO-médiateur pour résoudre un conflit sémantique. Le travail des OO-médiateurs consiste à établir des correspondances entre les terminologies contenues dans les différentes ontologies pour les intégrer en une représentation homogène des données. Cette représentation homogène permet de résoudre les hétérogénéités sémantiques et de répondre aux requêtes soumises par les composants de l'architecture WSMO.

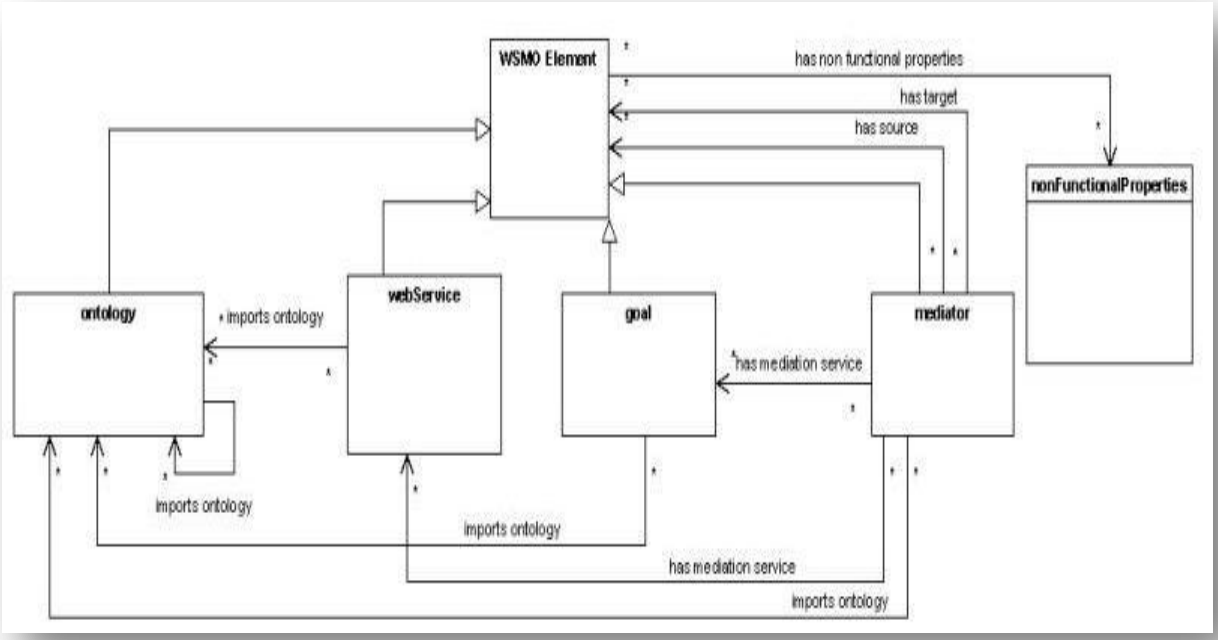

Figure 3.4 : Eléments d'une ontologie WSMO

3.3.2 Annotation des langages existants

L'annotation sémantique consiste à enrichir et à compléter la description d'un service. Elle établit des correspondances entre des éléments de la description et des concepts d'un ensemble d'ontologies de référence. Une ontologie de référence permet de représenter un domaine par des structures interprétables par une machine. Trois modèles principaux suivent l'approche d'annotation sémantique, à savoir **WSDL-S, SAWSDL** et **METEOR-S**. Les deux premiers modèles permettent d'annoter manuellement une description WSDL avec des éléments faisant référence à des ontologies tandis que le dernier permet de générer les annotations de l'interface d'un service à partir des annotations du code source de son implémentation.

3.3.2.1 WSDL-S

C'est une spécification commune à IBM et au laboratoire LSDIS. Elle a été soumise au W3C en 2005 [Akkiraju et al., 2005]. Son objectif principal est de fournir un processus d'annotation sémantique compatible avec les technologies existantes. A cette fin, [Akkiraju et al., 2005] proposent le WSDL-S méta-model [Miller et al., 2004].Concrètement, ce méta-modèle étend le WSDL en rajoutant trois éléments majeurs **<category>,**<**precondition>, <effect>** et deux attributs **modelReference** et **schemaMapping**. Les éléments introduits permettent de rajouter des informations qui n'étaient pas prises en compte dans WSDL comme les pré-conditions et les effets d'une opération. Tandis que les attributs permettent de référencer des concepts dans des ontologies de référence.

Nous illustrons dans la suite les annotations WSDL-S à travers l'exemple de l'opération **getPaymentOrder** présentée dans la Figure 3.5.Elle représente une demande de facture adressée à un service de facturation. Elle prend en entrée le message **paymentOrderRequest** qui contient la demande de facture et retourne en sortie le message **paymentOrderDispatch** contenant la facture.

Les extensions WSDL-S indiquée en gras sur la Figure 3.5 sont les suivantes :

- L'élément **<category>** est un sous-élément de **<portType>**. Il précise la catégorie d'un service lors de la publication dans un annuaire ou registre. Par exemple, pour l'opération **getPaymentOrder** de la Figure 3.5, **<category>** établit la

relation avec payment (code 1414) de la taxonomie disponible à l'adresse "http ://myTaxonomy.edu".

- L'´élément**<precondition>**, sous-élément de <operation>, indique les pré-conditions à vérifier pour que l'opération s'exécute comme prévu. Par exemple, pour l'opération **getPaymentOrder** de la Figure 3.5, **<precondition>** indique que pour qu'un ordre de paiement puisse être sollicité, il faut que l'achat associé existe.
- L'´élément**<effect>,** sous-élément de **<operation>,** indique les effets de l'exécution de l'opération. Par exemple, pour l'opération **getPaymentOrder** de la Figure 3.5, **<effect>** indique que l'effet de l'envoi d'un ordre de paiement est l'attente d'un paiement référencée par wait4payment.
- L'attribut **modelReference** peut être ajouté à un **<xs :element>** dans une grammaire XML et aux éléments <operation>, <precondition> et <effect>. Il indique une association avec un élément correspondant dans une ontologie donnée. Par exemple, pour l'opération **getPaymentOrder** de la Figure 3.5, l'attribut **modelReference** établit le lien avec le concept RequestPaymentOrder de l'ontologie myOntology.
- L'attribut **schemaMapping** peut être ajouté à un **<xs :element>** dans une grammaire XML. Il permet de décrire les correspondances entre la grammaire annotée et les ontologies de référence.

<**wssem:category** name=”payment" taxonomyURI="http://myTaxonomy.edu" taxonomyCode=“1414” />
<operation name = "getpaymentOrder” **wssem:modelReference**="myOntology#RequestPaymentOrder">

<input message = "tns:paymentOrderRequest"/>

<output message = "tns:paymentOrderDispatch"/>
<**wssem:precondition** name="ExistingPurchase" **wssem:modelReference**="myOntology#PurchaseExists"/>
<**wssem:effect** name="paymentOrderDispatchEffect" **wssem:modelReference**="myOntology#wait4payment"/>

</operation>

Figure 3.5 : Interface WSDL-S

3.3.2.2 SAWSDL

Le Semantic Annotations for Web Services Description Language and XML Schema, recommandé par W3C en avril 2007, est une des approches les plus récentes initiée par la communauté des services web sémantiques [Farrell et Lausen, 2007].Il présente un mécanisme permettant d'annoter sémantiquement les services décrits avec WSDL et leurs schémas XML associés. Les auteurs affirment que SAWSDL ne spécifie pas un langage pour représenter les modèles sémantiques, mais plutôt il fournit un mécanisme à travers lequel des concepts appartenant à des modèles sémantiques existants peuvent être référés à partir d'un document WSDL 2.0".

SAWSDL propose des extensions à WSDL 2.0 similaires à celles proposées par WSDL-S. Sa particularité réside dans l'annotation supplémentaire des schémas XML. Les principales extensions permettant d'annoter un document WSDL 2.0 sont les attributs suivants :**modelReference**, **liftingSchemaMapping** et **loweringSchemaMapping**.

L'attribut **modelReference** permet d'annoter tous les éléments WSDL 2.0. En particulier, il figure comme attribut de <interface>, <operation> et <fault>. Il pointe vers le concept équivalent en rajoutant son adresse. Nous reprenons par exemple, l'opération getPaymentOrder qui prend en entrée un message de demande de facture paymentOrderRequest et retourne en sortie un message paymentOrderDispatch contenant la facture. Pour associer cette opération avec le concept RequestPaymentOrder de l'ontologie myOntology, il suffit d'ajouter à l'élément définissant l'opération l'attribut suivant **sawsdl :modelReference="myOntology#RequestPaymentOrder".**

Les attributs **liftingSchemaMapping** et **loweringSchemaMapping** permettent d'associer à un schema type ou à un élément un concept dans une ontologie de référence adoptée. Des prototypes d'environnement de travail implémentant SAWSDL, pour décrire et manipuler les services, émergent [Klusch et Kapahnke,2008].

3.3.2.3 METEOR-S

Le projet METEOR-S [Amit et al., 2005] initié par le laboratoire LSDIS vise l'intégration des technologies de services avec celles du web sémantique. Il ne fournit pas un nouveau modèle de représentation des services mais propose de générer des annotations sémantiques pour les fichiers de type WSDL.

METEOR-S est basé sur le projet METEOR qui signifie “Managing End To End OpeRations”. Ce dernier traite de la gestion des workflows de processus à grande échelle dans des environnements hétérogènes. L’idée principale de METEOR-S est de générer une annotation sémantique de l’interface d’un service à travers des références à une ontologie ajoutée manuellement dans le code source de l’implémentation du service [Amit et al., 2005][Abhijit et al., 2004][Rajasekaran et al., 2004][Sivashanmugam et al., 2003].

Les auteurs présentent un prototype permettant de générer des interfaces de services en

1) WSDL 1.1 ou les liens avec les concepts d’une ontologie de référence sont décrits avec les éléments XML d’extensibilité autorisés par la spécification du langage ou en
2) WSDL-S à travers les attributs modelReference, schemaMapping et l’élément <category>.

3.4 Conclusion

Les services Web sémantique sont un domaine de recherche émergent. A l’état actuel il existe des technologies pour le développement des services Web. Cependant, ces technologies exigent que l’utilisateur humain interagisse avec le système tout au long du processus de découverte de services Web. Les technologies du web sémantique ont été utilisées pour palier à ce problème en enrichissant les services Web de sémantique, ce qui permet l’automatisation des divers aspects relatifs aux services Web.

Les avantages de l’utilisation du Web sémantique pour la description des services Web sont nombreux. En plus, de rendre l’interface du service Web accessible automatiquement par des machines, ils permettent également, la description de propriétés non fonctionnelles telles que la qualité de services, les contraintes de sécurité, et l’intégration effective des services Web dans des applications industrielles, d’une manière uniforme compréhensible par tous. Concrètement, ce qui nous intéresse est l’automatisation, autant que possible, des divers aspects relatifs aux services Web, en particulier la composition des services Web sémantiques, qui fera l’intérêt du prochain chapitre.

Chapitre4 : Etat de l’art de la Composition des Services Web

4.1 Introduction

Un des défis des SOA est l'intégration de services ou de systèmes distribués pour l'approvisionnement de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour des applications, d'autres services ou plus communément pour des utilisateurs humains.

En effet, si une application ou un client requièrent des fonctionnalités, et qu'aucun service n'est seul apte à les fournir, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client [Matskin et al., 2005]. C'est ce que l'on appelle la composition de services [Gustavo et al., 2004].

La composition de services vise à faire inter-opérer, interagir et coordonner plusieurs services pour la réalisation d'un but. Malgré les efforts de recherche et de développement autour de la problématique de la composition des services, elle reste une tâche hautement complexe et pose un certain nombre de défis. Sa complexité provient généralement des sources suivantes :

- L'augmentation dramatique du nombre des services web sur le web rend très difficile la recherche et la sélection des services web pouvant répondre à un besoin donné.
- Les services sont créés et mis à jour de façon hautement dynamique.
- Les services web sont d'habitude développés par différentes organisations qui utilisent différents modèles conceptuels pour décrire les caractéristiques des services web.
- La modularité constitue une caractéristique importante des services Web, par conséquent, les services Web composites doivent garder récursivement les mêmes caractéristiques que les services Web basiques à savoir auto-descriptifs, interopérables, et facilement intégrables [Melliti, 2004].

Dans cette partie nous présentons les définitions et les types de composition de services web présents dans la littérature .Ensuite nous étudions les langages de composition .Enfin un état de l'art des travaux qui proposent des approches de composition/découverte est décrit suivi d'une étude comparative des différentes solutions.

4.2 Définitions et types de composition de services Web

4.2.1 Définitions

La composition des services Web est le processus de construction de nouveaux services Web à valeur ajoutée, à partir de deux ou plusieurs services Web déjà présents et publiés sur le Web. Un service Web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres services Web, et des changements des messages entre eux afin de faire appel à leurs fonctionnalités. La composition de services Web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interaction [Arenaza, 2006].

En d'autres termes, la composition des services Web est la combinaison des services Web existants pour former de nouveaux services. L'objectif principal de la composition de service Web est la possibilité de créer de nouvelles fonctionnalités d'un nouveau service Web, en combinant des fonctionnalités offertes par d'autres services Web existants. Elle implique la capacité de sélectionner, de coordonner, d'interagir, et de faire inter-opérer des services Web existants.

Selon [Benatallah et al., 2002] le cycle de vie d'une composition de services Web est définit à partir de six activités (Figure 4.1):

- **L'encapsulation de services natifs (Wrapping services).** Cette première activité permet de s'assurer que tout service peut être appelé lors d'une composition, indépendamment de son modèle de données, de son format de message, et de son protocole d'interaction.
- **L'établissement d'accord d'externalisation (Setting outsourcing agreements).** Cette seconde activité consiste à négocier, établir, et appliquer des obligations contractuelles entre les services.
- **L'assemblage de services composants (Assembling composite services).** Cette activité permet de spécifier, à un haut niveau d'abstraction, l'ensemble des services à composer afin d'atteindre l'objectif attendu. Cet assemblage comporte une phase d'identification des services et de spécification de leurs interactions conformément aux descriptions et aux accords entre services.

- **L'exécution de services composants (Executing services).** Cette activité consiste en l'exécution des spécifications de la composition précédemment définies.
- **Le contrôle de l'exécution de services composites (Monitoring services).** La phase de contrôle permet de superviser l'exécution de la composition en vérifiant, par exemple, l'accès aux services, les changements de statut, les échanges de messages. Ce contrôle permet de détecter des violations de contrats, de mesurer les performances des services appelés et de prédire des exceptions.
- **L'évolutivité des services (Evolving services).** Cette dernière phase permet de faire évoluer la composition en modifiant les altérations de l'organisation de services, en utilisant de nouveaux services, ou en prenant en compte les retours de la phase de contrôle.

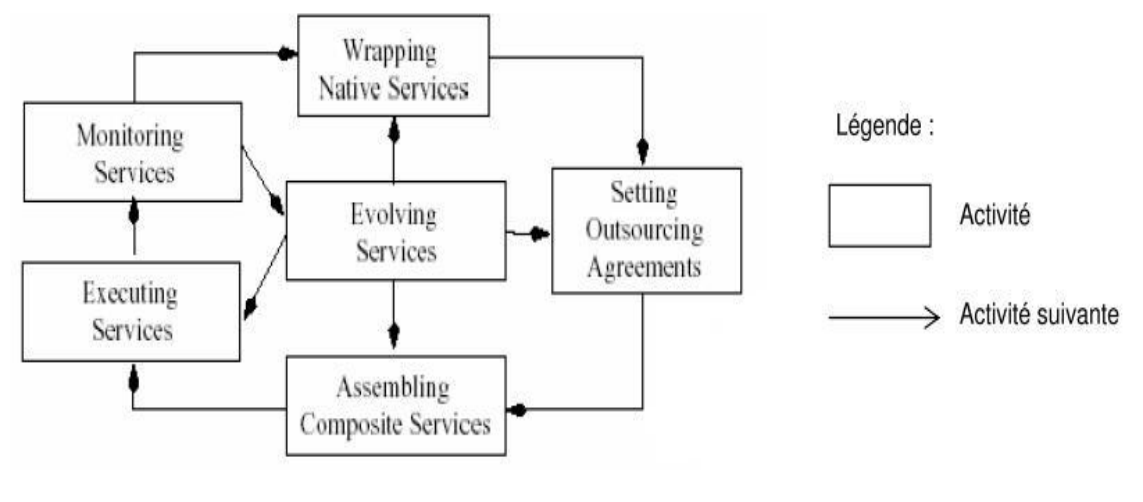

Figure 4.1: Illustration du cycle de vie de d'une composition de services Web par [Benatallah et al., 2002]

Dans le domaine de la composition de services Web, il existe deux manières différentes pour spécifier ce domaine, qui sont : l'orchestration et la chorégraphie des services.

[Barros et al.,2005b] définissent l'orchestration comme un ensemble de processus exécutés dans un ordre prédéfini afin de répondre à un but. Ce type de composition permet de centraliser l'invocation des services Web composants. Chaque service est décrit en termes d'actions internes. Les contrats entre deux services sont constitués selon le processus à exécuter.

À l'instar de [Barros et al. ,2005b],[Benatallah et al.,2005] définissent l'orchestration comme un processus exécutable.[Benatallah et al., 2005] ajoutent que l'orchestration est un ensemble d'actions à réaliser par l'intermédiaire de services Web. Un moteur d'exécution, un service Web jouant le rôle de chef d'orchestre, gère l'enchaînement des services Web par une logique de contrôle. Pour concevoir une orchestration de services Web, il faut décrire les interactions entre le moteur d'exécution et les services Web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les services Web composants.

D'après [Peltz,2003], l'orchestration de services Web consiste en la programmation d'un moteur qui appelle un ensemble de services Web selon un processus prédéfini. Ce moteur définit le processus dans son ensemble et appelle les services Web (tant internes qu'externes à l'organisation) selon l'ordre des tâches d'exécution. La Figure 4.2 illustre l'exécution du moteur (lui-même un service Web – Service Web Moteur) permise par l'enchaînement de l'exécution de deux autres services Web (le Service Web 1 puis le Service Web 2). Cet enchaînement est possible via un opérateur d'ordonnancement (représenté par le losange dans la figure). L'exécution de la composition repose sur l'appel du Service Web 1, puis sur l'appel du Service Web 2, réalisés tous deux par le Service Web Moteur.

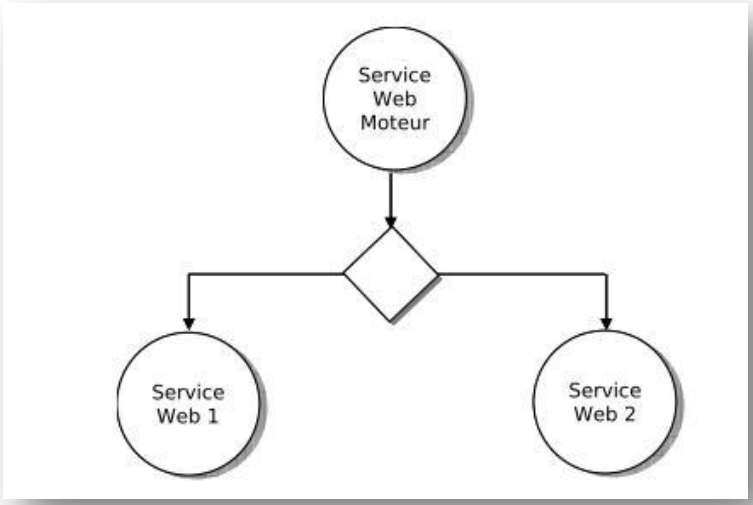

Figure 4.2 : Illustration de l'orchestration, d'après [Peltz, 2003]

En d'autres termes, l'orchestration de services Web exige de définir l'enchaînement des services Web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelants suivant le script d'orchestration.

D'après [Barros et al., 2005b], la chorégraphie permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de services Web. La collaboration entre chaque service Web de la collection (faisant partie de la composition) est décrite par des flots de contrôle. Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée.

D'après [Peltz, 2003], la description de chaque service Web intervenant dans la chorégraphie inclut la description de sa participation dans le processus. De ce fait, ces services peuvent collaborer à l'aide de messages échangés afin de savoir si tel ou tel service peut aider dans l'exécution de la requête. Chaque service Web peut communiquer avec un autre service Web par l'intermédiaire d'échange de messages. La Figure 4.3 représente un protocole d'initiation de collaboration entre deux services dans le cadre d'une chorégraphie. Dans cet exemple, le Service Web 1 demande l'exécution d'une méthode du Service Web 2 par l'intermédiaire d'un envoi de message (Requête de service). Cette requête est acceptée par le service Web 2. Ce dernier envoi un message d'acceptation au Service Web 1(Acceptation). Le Service Web 1 accepte le service proposé par le Service Web 2 en lui envoyant un message (Service admis) accordé (Acceptation) par ce second service. Une fois ces messages échangés le Service Web 1 peut invoquer le Service Web 2 dans le cadre de la composition.

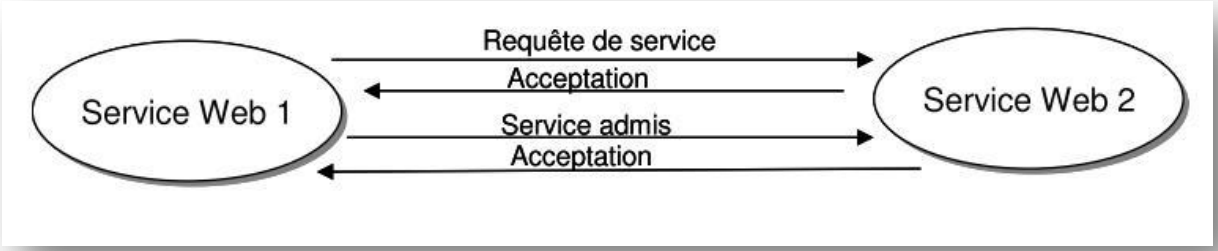

Figure 4.3 : L'illustration de la chorégraphie, d'après [Peltz, 2003]

La chorégraphie est aussi appelée composition dynamique [Peltz,2003]. En effet, l'exécution n'est pas régie de manière statique comme dans une composition de type orchestration. Dans une chorégraphie, à chaque pas de l'exécution (i.e. à chaque étape de la composition), un service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance.

4.2.2 Types de composition de services Web

L'idée de la composition de services Web consiste à définit comment les services Web vont être rassemblés selon certains règles, pour atteindre le but demandé par un utilisateur. Une fois que, la description de la composition est réalisée, il est possible de savoir facilement quels services web appartiendront à cette composition. Les solutions proposées peuvent être classifiées selon deux axes :

A. En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions sont manuelles, semi-automatiques ou automatiques [Charif, 2007] :

- La composition manuelle : la composition manuelle des services Web suppose que l'utilisateur gère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés.
- La composition semi-automatique : les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'elles font des suggestions sémantiques pour aider à la sélection des services Web dans le processus de composition.

- La composition automatique : la composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise.

B. Selon que, la sélection des services Web et la gestion du flot soient faites ou non a priori, une autre approche sera dite statique ou dynamique :

- La composition statique des services web : dans cette approche, les services web à composer sont choisis à l'heure de faire l'architecture et le design. Les composants sont choisis et reliés ensemble, avant d'être compilés et déployés [Dustdar et Schreiner,2005].Ceci peut marcher en autant que l'environnement des services web, les partenaires d'affaires, ainsi que les dits composants changent peu ou pas du tout. Microsoft Biztalk et Bea Weblogic sont deux exemples de moteurs de composition statique de services web [Sun et al., 2003]. Si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voire de la définition du processus et créerait l'obligation de faire une nouvelle conception du système.Dans ce cas, la composition statique des services web est considérée trop restrictive: les composants doivent s'adapter automatiquement aux changements imprévisibles [Sun et al., 2003].
- La composition dynamique des services web : Dans cette approche, les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur [Osman et al., 2005].Cette approche offre le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles, qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services Web est propice dans un environnement, tel que le Web et l'informatique pervasive où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées.

4.3 Langages de composition de services web

De nombreux industriels (tels qu'IBM ou Microsoft)et consortium (tel que le W3C) travaillent afin de mettre en œuvre un langage de composition de services Web standard (tel que WSCI –Web Service Choreography Integration)[Arkin et al.,2002]. Dans cette section, nous étudions les langages qui sont soit largement utilisés dans l'industrie (BPEL4WS [Andrews et al., 2003]), soit en cours de standardisation (WS-CDL [Kavantzas et al., 2005]) .

4.3.1 BPEL4WS (Business Process Execution Language for Web Services)

BEA, IBM, SAP, Siebel Systems et Microsoft ont uni leurs efforts afin de produire un langage de composition de services Web, conçu pour supporter les processus métier à travers les services Web. Ce langage, BPEL4WS (Business Process Execution Language for Web Services), est issu de la fusion de deux langages : WSFL – Web Service Flow Language [Leymann, 2001] d'IBM et XLANG [Thatte,2001] de Microsoft. BPEL4WS combine les caractéristiques d'un langage de processus structuré par bloc (XLANG) avec ceux d'un langage de processus basé sur les processus métier (WSFL).

BPEL4WS [Andrews et al.,2003] est basé sur XML et sur les Workflows. Ce langage distingue les processus abstraits des processus exécutables :

Le processus abstrait :ce processus spécifie les messages échangés entre les différentes parties (services Web composants) sans indiquer le comportement de chacune d'elles. [Wynen, 2003] parle de Business Protocol, c'est-à-dire la spécification du comportement des partenaires par rapport aux messages échangés, sans rendre public le comportement interne. Ce processus abstrait peut être relié à une composition de type chorégraphie. Les services Web communiquent alors à l'aide d'échanges de messages (partie gauche de la Figure 4.4).

Le processus exécutable : ce processus permet de spécifier l'ordre d'exécution des activités, le partenaire concerné, les messages échangés entre ces partenaires, et les mécanismes des erreurs et des exceptions. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les partenaires.

La Figure 4.4 illustre la mise en œuvre des deux types de processus (exécutable et abstrait) par BPEL4WS. La définition du processus métier est donnée par le processus exécutable. Les processus abstraits gèrent les invocations entre les différents services Web permettant l'exécution de la composition de services définie dans le processus exécutable.

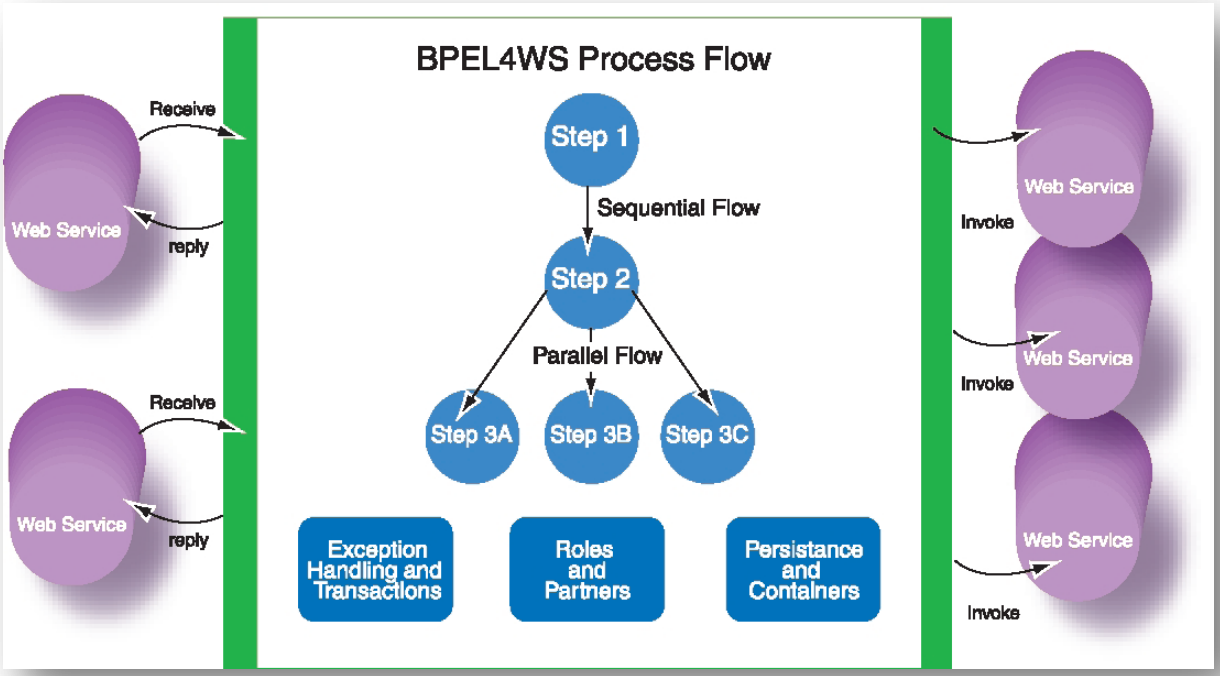

Figure 4.4 : Le flot de processus avec BPEL4WS, d'après [Peltz, 2003]

Trois éléments permettent à BPEL4WS de gérer le flot de processus dans le processus exécutable : les transactions (Exception handling and transactions), les partenaires (Roles and Partners) et les espaces de stockage (Persistance and Containers) (voir Figure 4.4) :

Les transactions. Les transactions sont utilisées dans BPEL4WS pour gérer les erreurs et les appels d'autres services si le service appelé est indisponible ou défaillant.

Les partenaires. Les partenaires sont différents services Web invoqués dans le processus. Ils ont chacun un rôle spécifique dans un processus donné. Chaque partenaire est décrit par son nom, son rôle (en tant que service indépendant), et son rôle dans le processus.

Les espaces de stockage. Les espaces de stockage permettent la transmission des données. Le flot de processus BPEL4WS permet que ces données soient cohérentes à travers les messages échangés entre les services Web. Un message peut être un message d'appel (invoke), de réponse (reply) ou d'attente (receive).

BPEL contient deux ensembles de constructeurs : constructeurs primitifs et constructeurs structurels. Ces constructeurs permettent la description d'un processus d'affaire exécutable.

Les constructeurs primitifs permettent des fonctionnalités. Parmi ces constructeurs on trouve :

- **<Invoque> :** invoque un service Web.
- **<Receive> :** attends qu'un client ou un service invoque le business process par envoie de requête.
- **<Reply> :** génère une réponse synchrone.
- **<Assign> :** permet de manipuler les données des variables.
- **<Throw> :** indique les erreurs et les exceptions.
- **<Wait> :** bloque l'exécution pour un temps donné.

Les constructeurs primitifs peuvent être combinés pour définir un algorithme complexe spécifiant les étapes par lesquelles passe le business process en utilisant les constructeurs structurels tel que :

- **<Sequence> :** définit un ensemble de constructeurs invoqués par ordre d'apparition dans la séquence
- **<Flow>** : définit un ensemble de constructeurs invoqués en parallèle
- **<Switch> :** pour le choix d'une branche à exécuter
- **<while> :** définit une boucle.

BPEL4WS est le premier langage de composition de services Web adopté par la communauté des services Web. Ceci est principalement dû au fait que ce langage possède une grande expressivité dans la définition du processus exécutable [Wohed et al.,2003]. L'inconvénient principal de BPEL4WS est que la définition du processus est rigide. Si une activité du processus échoue, le processus dans son intégralité échoue. Aucun retour en arrière et aucune alternative au processus ne sont possibles. De même, lors de la description du processus exécutable, la définition des flots de données ne permet pas de connecter des

services dont les entrées/sorties ne correspondent pas exactement. BPEL4WS ne prévoit pas de mécanisme de transformation de données.

4.3.2 WS-CDL (Web Services Choreography Description Language)

Le WS-CDL(Web Services Choreography Description Language) est un langage qui permet de décrire une vision globale des collaborations entre les services,[Kavantzas et al., 2005] insistent sur le fait que WS-CDL n'est pas un langage de description de processus exécutables, ni un langage d'implémentation. Il décrit les séquences ordonnées de messages impliquant plusieurs entités visant à accomplir un objectif commun.WS-CDL reprend et développe la spécification Web Service Choreography Interface WSCI.

WS-CDL permet :

1. de désigner les variables et les types de données échangées,
2. de décrire les activités impliquées et
3. de décrire les structures illustrant les interactions entre les activités.

WS-CDL consiste à définir un fichier XML décrivant une chorégraphie.Les éléments d'un document WS-CDL sont illustrés en Figure 4.5.

Nous détaillons l'élément <choreography> décrivant des règles générales d'échange de messages. Il permet de définir trois aspects d'une chorégraphie:

1. Choreography Life-line décrite une collaboration et indique son état d'avancement,
2. Choreography Exception Blocks exprime les actions à entreprendre en cas d'erreur,
3. Choreography Finalizer Blocks indique les méthodes permettant de valider les résultats d'une chorégraphie ou de les annuler.

En résumé WS-CDL permet de décrire les règles selon lesquelles une collaboration doit avoir lieu. Il fournit une structuration globale de l'interaction en fonction de laquelle chaque participant décrit son processus métier et par la suite ses services.

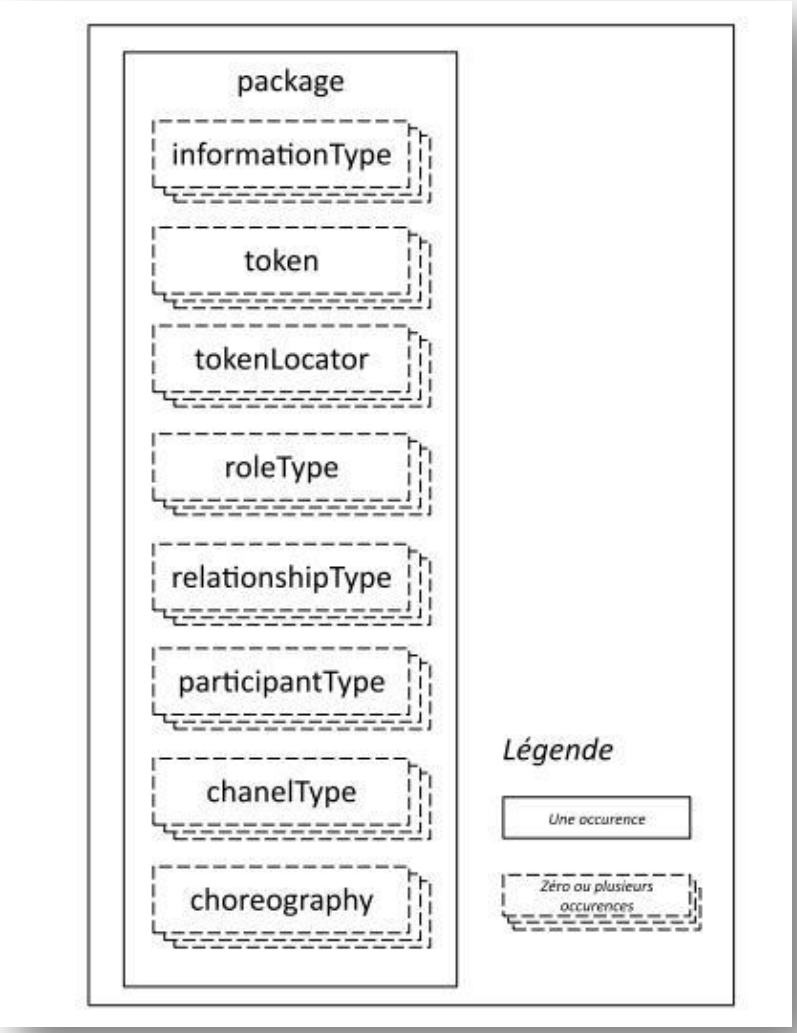

Figure 4.5: Eléments de description WS-CDL

4.4 Les différentes approches de la composition automatique

Différentes approches ont été proposées pour composer automatiquement les services web. Selon [Rao and Su,2004] , la plupart des recherches dans ce domaine peuvent se regrouper selon les axes suivants :

4.4.1 Le Workflow

Plusieurs auteurs traitent la composition de services en utilisant les workflows [Rao et Su, 2004],[Cardoso et Sheth, 2002]. La définition de la composition de services correspond alors à un ensemble de services atomiques (ou composés) sur lesquels on effectue un contrôle du flux. D'autre part, un workflow a également besoin de définir un flux d'activités.

Un workflow est une abstraction d'un processus de type business. Il est composé d'un nombre d'échelons logiques (aussi appelés tâches ou activités), de dépendances entre tâches, de règles et de participants. Dans un workflow, une tâche peut représenter une activité humaine ou un système (logiciel). Il est nécessaire d'associer une tâche à un service quand le workflow est appliqué aux services web. Une composition d'un workflow implique la sélection de tâches appropriées à des fonctionnalités désirées, devant prendre en compte les connections entre ces tâches (flux de contrôle et de données). Les workflows qui gèrent les services web (aussi appelé de e-Services) sont appelés des e-workflows [Cardoso et Sheth, 2002].

Les compositions de services qui utilisent cette approche sont normalement basées sur les workflows manuels, aussi appelé statiques. Dans le workflow manuel, l'utilisateur doit définir l'ensemble des tâches et des dépendances parmi les données. Chaque tâche contient des requêtes qui permettent de chercher des services concrets pour la satisfaire, puis l'exécuter. Dans ce cas, seules la sélection et la liaison du service sont faites automatiquement.

Du coté des compositions dynamiques, le modèle du processus ainsi que la sélection du service sont faits automatiquement [Rao and Su, 2004].A cet égard, une composition automatique demande des workflows capables de reconnaitre les services web correspondants à chaque tâche, mais aussi de trouver d'autres services au cas ou ceux-ci soient indisponibles ou dévient de leur exécution normale.

Dans [Casati et al., 2001], les auteurs redéfinissent la plate-forme de composition de services EFlow [Casati et al., 2000] et proposent un prototype de langage de définition de services composés (Composite Service Definition Language : CSDL). Une intéressante fonctionnalité de CSDL est la distinction entre les services et les opérations des services. Les auteurs définissent donc les nœuds de services et des nœuds d'opérations, les premiers faisant référence aux services, les seconds s'adressant directement à une méthode particulière d'un service. Selon les auteurs, CSDL fournit les fonctionnalités adaptatives et dynamiques qui correspondent à l'évolution rapide des entreprises et des environnements informatiques et technologiques dans lesquels les services Web sont utilisés.

Polymorphic Process Model (PPM) [Schuster et al., 2000] utilise une méthode qui combine le modèle de plan statique avec le modèle de plan dynamique. Les auteurs décrivent la composition de services Web à l'aide d'activités. PPM comporte deux types d'activités, les activités génériques, proposées et exécutées par le moteur de PPM et les activités spécifiques à l'application qui doivent être définies par l'utilisateur. PPM choisit de séparer les implémentations de ces activités, réalisées par les services Web, et les interfaces de ces activités, qui sont une représentation des services Web. Les interfaces des activités sont modélisées comme des machines à états, qui incluent des états et des opérations permettant la transition d'états, et sont également décrites par leurs entrées/ sorties. Les activités spécifiques à l'application sont des abstractions du comportement des services Web. La réalisation des activités par PPM est effectuée en liant les opérations des activités avec des opérations de services Web spécifiques. Cette tâche est similaire à la liaison des services Web dans EFlow, pour lequel les implantations des interfaces sont liées lors de l'exécution.

4.4.2 La planification

La planification est un des domaines de l'Intelligence Artificielle (IA) qui permet de choisir et d'organiser des actions, en fonction d'un but donné. Le mot planification est également utiliser dans plusieurs autres domaines, tels que l'économie, la politique, l'architecture et encore dans la vie de tous les jours.

La planification en IA peut aussi être conceptualisée comme un choix et une organisation d'actions pour changer l'état d'un système. Par extension, elle produit un plan qui est présenté sous la forme d'une collection de descriptions d'opérateurs. Le planificateur ou générateur de plans est le système qui produit un plan. L'exécution est la réalisation des actions présentes dans le plan. L'exécution d'un plan modifie les propriétés du monde en le faisant évoluer de l'état initial jusqu'au but désiré [Ghallab et al.,2004] [Regnier,2004].La Figure 4.6 représente un problème de planification[Albers, 1997].

La plupart des approches de planification utilise un modèle conceptuel de transition d'état. Plus formellement, le problème de la planification correspond à un quadruplet [Ghallab et al., 2004] :

$\sum = (\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{E}, \boldsymbol{\gamma})$

Où

S = {s1,s2, ...} est un ensemble d'états fini.

A = {a1,a2, ...} est un ensemble d'actions fini.

E = {e1,e2, ...} est un ensemble d'événements fini.

et $\boldsymbol{\gamma : S \times A \times E \rightarrow 2^S}$ est la fonction de transition d'état.

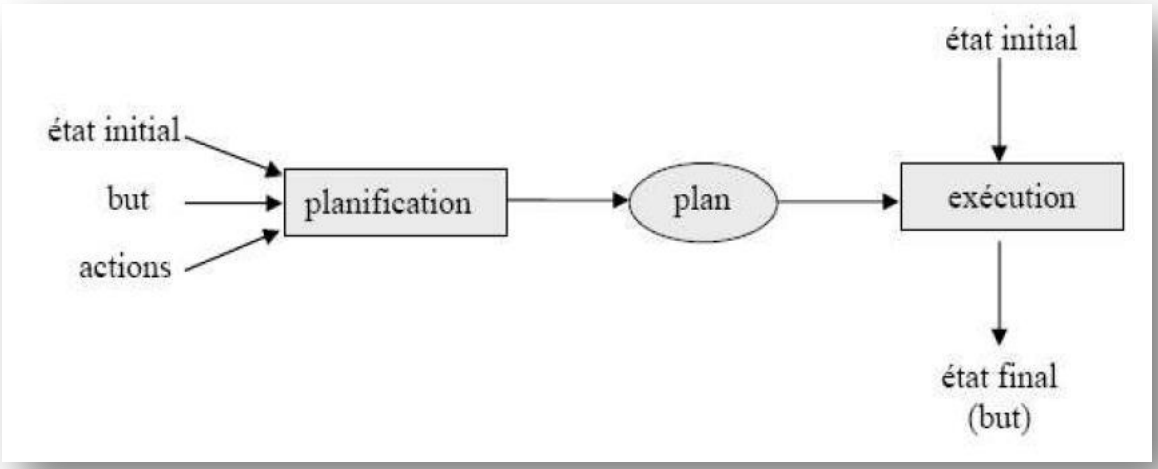

Figure 4.6 : Problème de planification

Le monde correspond à un ensemble de faits qui évoluent selon des événements ou des actions. C'est cet ensemble de faits qui représente le monde dans le système de planification. Tout ensemble de faits doit donc correspondre le mieux possible à un état du monde réel.

Les événements sont des transitions qui ne sont pas contrôlées par le système de planification. Les actions sont quant à elles des transitions contrôlées par le système de planification.

Si une action **a** est applicable dans un état **s**, le monde évoluera vers un autre état **γ(s, a).** Donc, en considérant un système de transition d'état **P**, la planification a pour but de trouver une suite d'actions pour atteindre un objectif à partir d'une situation donnée.

Maintenant que nous avons défini le problème de planification, nous allons établir la relation avec la composition des services web. L'état initial et l'état final sont directement associés à une requête d'un client. Les actions peuvent être effectivement comme des

services web [Rao et Su, 2004] [Sirin et Parsia, 2004a][Pistori et al., 2005]. Cette relation entre la planification et les services web a permis l'évolution des langages tels que l'OWL-S, qui actuellement est le plus approprié, voire le seul langage permettant de décrire les services web, ainsi que ses pré-conditions et ses effets.

Dans la section suivante, nous allons détailler quelque techniques de planification à savoir le calcul de situation, le PDDL, la planification basé sur les règles, les preuves par théorèmes et la planification hiérarchique.

4.4.2.1 Le calcul de situation (situation calculus)

Le Calcul de Situation [Regnier, 2004] [Ghallab et al., 2004] est un langage basé sur la logique du premier ordre permettant de représenter et raisonner sur un domaine dynamique. Ce formalisme a été introduit par John McCarthy et Patrick J. Hayes en 1969 [McIlraith et al., 2002] .Les principaux éléments sont les actions, les fluents et les situations. Le monde est conçu comme un arbre de situations, débutant par la situation initiale **s0**, et évoluant jusqu'à la nouvelle situation par l'application d'une ou plusieurs actions. Une situation **s** donnée correspond toujours à un historique de l'ensemble d'actions réalisées sur **s0** (Voir la Figure 4.7).

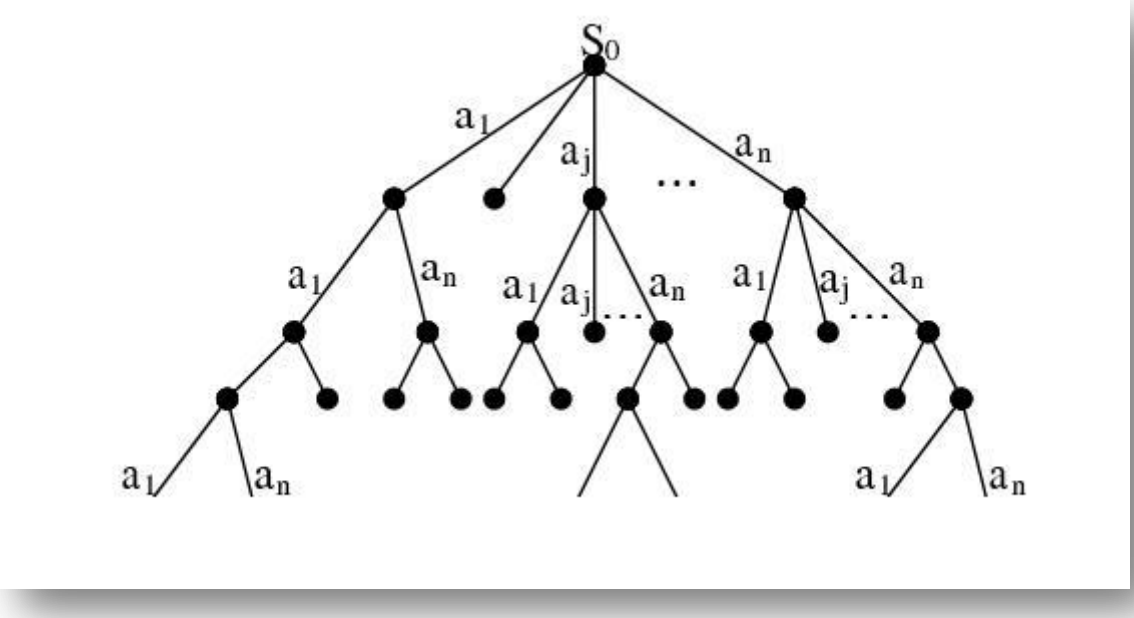

Figure 4.7 : Le calcul de situation

Dans cette approche, la dynamicité du monde provient d'une succession de situations, résultant de l'application d'actions diverses. Comme, l'on peut voir dans la Figure 4.7, une situation représente l'historique des occurrences d'actions. Une situation où aucune action n'a été exécutée auparavant, laisse inchangé le monde et correspond à la situation **s0**. Une nouvelle situation est calculée par la fonction **faire**. Donc, si **a** est une action et **s** une situation, le résultat de l'exécution de **a** dans **s** est calculé par la fonction **faire (a,s)**.

Le problème de la composition de services Web est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en termes de prédicats du premier ordre, dans le langage de calcul situationnel. Les services Web sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et sont instanciées à l'exécution à partir des préférences utilisateur.

[McIlraith et al., 2002] proposent une extension de Golog. Ce dernier est un langage de programmation logique basé sur la situation calculus. La technique présentée repose sur un ensemble de procédures génériques de haut niveau et sur la personnalisation des contraintes associées. Les procédures génériques représentent la requête utilisateur. Les contraintes à respecter sont écrites avec un langage de premier ordre. Chaque service est modélisé comme une action. Deux types d'actions sont possibles : des actions primitives et des actions complexes telles que ces dernières sont des compositions d'actions individuelles. Les auteurs proposent une approche qui permet à l'utilisateur de traduire manuellement les descriptions OWL-S des services impliqués en Prolog. Après cette traduction, le système utilise des règles d'inférences logiques pour automatiser le choix d'un service pour chaque action en respectant les contraintes associées.

4.4.2.2 PDDL (Planning Domain Definition Language)

Le Planning Domain Definition Language (PDDL) est un langage de planification largement adopté par les communautés de l'Intelligence Artificielle.[McDermott, 2002] décrit les similarités entre l'approche de description et de composition OWL-S et le PDDL. Il présente une approche pour traduire les descriptions OWL-S en PDDL afin de modéliser la composition comme un problème de satisfaction d'un "planning". Il propose la notion de

« value of an action » qui modélise l'exécution d'un service. Elle exprime des changements qui peuvent avoir lieu dans l'environnement.

4.4.2.3 Planification basé sur les règles (rule based planning)

[Medjahed et al., 2003] présentent une technique pour générer une composition de services à partir d'une description de haut niveau. La méthode utilise des règles de composabilité pour déterminer si deux services sont composables. Elle se décompose en quatre phases. La première, la phase de spécification , permet un haut niveau de description de la composition décrite en utilisant un langage de spécification de services composés appelé CSSL (Composite Service Specification Language). La seconde, la phase d'appariement, utilise des règles de composabilité pour générer des plans de composition qui sont conformes aux spécifications de la requête de l'utilisateur. La troisième est la phase de sélection. Si plus d'un plan est généré durant la seconde phase, l'utilisateur du service choisit un plan en fonction de différents paramètres tels qu'un paramètre de qualité de composition, son rang ou son coût. La phase finale est la phase de génération. Une description détaillée de la composition des services Web est générée automatiquement et est présentée à l'utilisateur du service.

Nous allons présenter plus en détails les règles de composition afin de mieux exposer comment est généré le plan. Les règles de composabilité considèrent des propriétés syntaxiques et sémantiques des services Web. Les règles syntaxiques incluent les règles pour les modes de fonctionnement, par exemple l'envoi de message uniquement (one-way) ou l'envoi de requête puis l'attente d'une réponse (request-response), et les règles de liaison des protocoles d'interaction des services Web. Les règles sémantiques incluent les sous-ensembles suivants :

- **composabilité des messages définis :** deux services Web sont composables si et seulement si le message de sortie d'un service est compatible aux sens des auteurs avec le message d'entrée de l'autre service,
- **composabilité sémantique de l'opération :** elle définit la compatibilité entre les domaines, les catégories et les propositions (fonctionnalités) de deux services,

- **qualité de composition :** elle définit les préférences de l'utilisateur vis-à-vis de la qualité des opérations pour la composition de services,

- **pertinence de la composition :** elle évalue si une composition de services est bien fondée.

4.4.2.4 Planification Hiérarchique

Dans [Wu et al., 2003], le planificateur de SHOP2 [Kuter et al., 2005] est utilisé pour la composition automatique de services Web qui sont décrits grâce au langage DAML-S.SHOP2 est un planificateur de type réseau de tâches hiérarchiques (HTN, hierarchical task network). Les auteurs pensent que le concept de décomposition de tâches dans le planificateur HTN est très similaire au concept de décomposition de processus composés, décrits dans l'ontologie de processus proposée par DAML-S. Les auteurs avancent aussi que le planificateur HTN est plus efficace que les autres langages de planification , comme par exemple Golog.

[Sirin et al., 2002] présentent une méthode semi-automatique pour la composition. La composition semi-automatique consiste à assister l'utilisateur lors de la création d'une composition. En fait, la composition est assistée par ordinateur. [Sirin et al., 2002] proposent une méthode qui utilise DAML-S. La méthode proposée utilise les attributs fonctionnels et des attributs non-fonctionnels définis dans le ServiceProfile pour proposer à l'utilisateur les services qui semblent les plus appropriés pour répondre à sa requête. Pour ce faire, les attributs fonctionnels comme paramètres d'entrées et de sorties du service sont représentés par des classes OWL et sont filtrés par un moteur d'inférence basé sur OWL et Prolog. Le moteur d'inférence peut ordonner les services Web filtrés en fonction de l'ordre d'éloignement des concepts. L'éloignement est un paramètre qui dénote l'importance des différences entre les classes OWL. Pour affiner le résultat, si plus d'une correspondance est trouvée, le système filtre le résultat en fonction des contraintes fournies par l'utilisateur sur les attributs non-fonctionnels. Les attributs non-fonctionnels sont les attributs utiles qui ne sont pas fournis par les attributs fonctionnels, par exemple, la localisation de la mesure lorsque le service ne la fournit pas lui-même. Seuls les services Web qui passent le filtre de contraintes sont présentés à l'utilisateur. La composition semi-automatique est attrayante car elle permet de surmonter les difficultés liées, tout d'abord, à la capture des besoins de l'utilisateur, puis à la composition automatique qui, le plus souvent, ne permet pas de garantir l'exactitude de la composition. L'utilisateur étant proactif, il peut s'assurer que la composition est bien celle

qu'il souhaitait. De plus, la méthode proposée est simple et montre que la génération de services Web composés peut être effectuée en combinant les fonctionnalités de la machine et les compétences de l'homme.

4.4.2.5 Preuves par théorèmes (theorem proving)

[Waldinger et al., 2001] propose de générer les services Web composés à partir de preuves de théorèmes. L'approche est basée sur la déduction automatique de preuves. La requête de l'utilisateur est décrite comme un théorème que l'on souhaite prouver. Initialement, les services Web disponibles et les pré-requis de l'utilisateur sont décrits dans la logique du premier ordre. L'utilisateur définit des axiomes de la logique classique, par exemple, des implications ou des équivalences, qui vont être utilisées pour générer une ou plusieurs preuves de la requête. Ensuite, une preuve est générée par le générateur de preuve de théorème SNARK.Enfin,la description de la composition de services est extraite d'une preuve particulière en fonction de la disponibilité des services Web.

[Lämmermann, 2002] applique une synthèse structurelle de programme (SSP, Structural Synthesis of Programme) pour une composition automatique. SSP est une approche déductive pour la synthèse de programme à partir de spécifications. Il définit un langage logique dans lequel les spécifications sont définies par des interfaces. Les interfaces définissent les services Web ou la requête. Ces interfaces sont composées de variables typées, de constantes, de liaisons entre les variables et constantes (binding) d'un axiome. L'axiome inclut seulement les propriétés structurales, i.e. les informations des entrées/sorties, en liant les entrées avec les sorties. Il étend son langage en y ajoutant des "variables de contrôle" qui sont des variables sans valeur mais qui peuvent servir à la définition des pré-/post-conditions. Les interfaces des services Web sont vues comme des axiomes et l'interface de la requête est vue comme un théorème à prouver. SSP déduit des interfaces prédéfinies une "preuve" de l'interface qui représente la requête, et déduit de la preuve la composition des services Web. Lämmermann assimile les services Web composés à des programmes et utilise le fait que les programmes sont des preuves.

4.4.3 Automates et Réseaux de Pétri

Les automates ou systèmes de transition d'états annotés offrent une sémantique précise et adaptée à la représentation de la composition. Les techniques de raisonnement applicables aux automates permettent de vérifier la structure de la composition et sa cohérence. Un automate est constitué d'un ensemble d'états, d'un ensemble de transitions annotées entre

états et d'un ensemble d'actions. Les annotations des transitions contiennent des actions représentant le passage d'un état à un autre.

L'ontologie WSMO associe à chaque élément webService des interfaces pour représenter son comportement. L'interface est décrite par une chorégraphie ou par une orchestration.[Scicluna et al., 2006] proposent un modèle de représentation de la chorégraphie et de l'orchestration en WSMO basé sur les machines abstraites d'états (ASM) . Les auteurs ne développent que l'aspect modélisation qui insiste sur la présence d'états et de transitions. Cette étude illustre le rôle des médiateurs WSMO dans une composition.

[Foster etal., 2004]suggèrent de représenter la composition de services par l'implémentation d'une couche multi-stakeholder distributed system (MSDS). MSDS est un environnement distribué ou des nœuds définis par des fournisseurs différents interagissent de façon concurrente sans connaissance totale de l'environnement. Les auteurs introduisent la notion de compatibilité entre services à trois niveaux : interface, comportement et paramètres d'entrée et de sortie. Ils présentent une approche de vérification qui consiste à traduire une composition exprimée en BPEL en un modèle d'automates finis et à compiler ce dernier en un système de transition d'états. Ils utilisent un outil de vérification (LTSA pour le model checking) qui permet de s'assurer de la consistance du graphe et d'affirmer qu'il ne contient pas des "impasses" (deadlock free).

[Bordeaux et al., 2004] suggèrent de représenter le comportement du service par un système de transition d'états. Cela revient à modéliser le service par un ensemble d'états reliés par des actions. Les auteurs précisent que ce modèle est déterministe dans la mesure ou "une action qui a lieu à un état donné conduit à un état déterminé".

Un réseau de Pétri est un ensemble de places et de transitions reliées par des arcs. L'aspect dynamique du réseau est décrit par un ensemble de règles. Chaque règle définit (1) les conditions à valider pour qu'une transaction ait lieu et (2) les effets de cette transaction. Les réseaux de Pétri (Petri Nets) fournissent une représentation précise et sémantiquement riche de l'ordre d'exécution des activités. Ils permettent un raisonnement plus efficace sur la composition. Pratiquement, les réseaux de Pétri sont utilisés pour modéliser des systèmes concurrentiels.

[Narayanan et McIlraith, 2002] proposent une traduction du process model de OWL-S en un réseau de Pétri afin de raisonner sur la vérification automatique de la composition. Ils proposent un outil qui génère automatiquement un réseau de Pétri pour une composition exprimée par un process model de OWL-S. Cet outil vérifie les propriétés de la composition générée. Il juge la consistance du graphe modélisant le flux d'interaction. Il détermine, par exemple, si tous les états (nœuds) peuvent être atteints, si les contraintes décrites sont respectées ou si des impasses existent, etc.

[Ouyang et al., 2005] proposent une traduction des contrôles de flux BPEL en réseaux de Pétri annotés. Cette traduction permet de raisonner automatiquement sur la structure du graphe d'interaction et sur ses propriétés.

4.5 Synthèse des travaux de recherche

Dans [Weise et al., 2007] les auteurs proposent trois algorithmes pour la composition automatique des services web fondés sur leurs descriptions sémantiques : L'algorithme IDDFS (Iterative Deepening Depth First Search), un algorithme heuristique et un algorithme génétique. Les paramètres d'E/S des services web sont annotés par des concepts sémantiques. L'utilisateur exprime ses besoins via une requête qui est elle aussi caractérisée par des concepts d'entrée et des concepts de sortie. Afin de satisfaire une requête donnée, les concepts d'entrée du service offert (composite) doivent être plus générales que les concepts d'entrée de la requête. En revanche les concepts de sortie du service offert doivent être plus spécifiques que les concepts de sortie de la requête. Le processus de composition vise à trouver une composition $S = \{S1, S2, \ldots . Sn\}$qui vérifie l'équation 1 :

$$\text{isGoal}(S) \Leftrightarrow \forall\, A \in S1.\,\text{in}\ \exists\, B \in R.\,\text{in}: \text{subsume}\,(A, B) \wedge$$

$$\forall\, A \in Si.\,\text{in}, i \in \{2..n\}\ \exists B \in R.\,\text{in} \cup S\,i-1.\,\text{out} \cup \ldots \cup S1.\,\text{out}: \text{subsume}\,(A, B) \wedge \quad \textbf{(1)}$$

$$\forall\, A \in R.\,\text{out}\ \exists B \in S1.\,\text{out} \cup \ldots \cup Sn.\,\text{out} \cup R.\,\text{in} : \text{subsume}\,(A, B)$$

Où subsume(A,B): (M × M) → {true, false} et A,B deux concepts appartiennent à la taxonomie M :

- ✓ subsumes(A, B) est vrai si A est une généralisation de B
- ✓ subsumes(B, A) est vrai si A est une spécialisation de B
- ✓ si subsumes(A, B) ou subsumes(B, A) n'est pas vrai ,A et B ne sont pas reliés
- ✓ subsumes(A, B) and subsumes(B, A) est vrai si et seulement si A=B

L'algorithme IDDFS est spécialement rapide à trouver des solutions si le répertoire des services web est de taille petite ou moyenne. L'algorithme heuristique est plus efficace et convient pour un répertoire de services web plus grand; et l'algorithme génétique approprié aux répertoires de services web de taille particulièrement grande.

[Chouchani, 2010] propose un outil basé sur les techniques des algorithmes génétiques pour la découverte et la composition de services web, le consommateur exprime le service web désiré par un fichier de description WSDL, le système interroge un espace de recherche sous la forme de collection de services Web et donne comme résultat une liste de services possibles (population). Le critère de sélection d'un service est sa valeur de similarité syntaxique avec le service cible. Les éléments de la population résultat sont des services qui existent dans l'espace de recherche (découverte d'un service approprié) ou qui ont été crées par le biais de la composition de services existants. Pour trouver un service web particulier, il faut trouver les différentes opérations qui le constituent, l'espace de recherche est formé donc par un ensemble d'opérations et l'algorithme génétique tente à trouver une opération similaire à l'opération cible.

[Alkamari, 2008] propose un mécanisme de composition dynamique de services Web spécifiés sous forme de fichiers WSDL par appariement de types (signatures). Cette approche ne demande aucun rajout aux fichiers WSDL, et aucune manipulation sur le registre UDDI. A partir d'un ensemble de départ composé d'un ou plusieurs types, elle va atteindre un ensemble d'arrivée composé d'autres types. Pour y arriver, nous composerons un flux de services Web découverts au fur et à mesure grâce aux types disponibles à chaque étape. Sur le plan pratique, cette approche met en œuvre l'algorithme « EnumerateRealizations » qui est présenté dans « Intelligent Component Retrieval for Software Reuse », et le teste sur différents fichiers WSDL.

[Casati et al., 2000] et [Casati et al., 2001] proposent une plate-forme nomée EFlow pour la spécification, l'établissement et le management des services Web composés. EFlow utilise un générateur de modèles de plan statique. Un service Web composé est modélisé par un graphe qui définit l'ordre d'exécution des tâches, celles-ci sont représentées par des nœuds dans le graphe et peuvent être réalisées par des services Web. Le graphe modélisant un service Web composé et la base de données des services Web sont créés manuellement. Le graphe peut contenir des nœuds de services, de décisions et d'événements. Les nœuds de services

représentent l'appel à un service Web élémentaire ou composé, les nœuds de décisions spécifient les alternatives et les règles contrôlant l'exécution et le flux de données, par exemple une disjonction ou une jonction. Enfin, les nœuds d'événements permettent aux services Web de recevoir et d'envoyer un certain nombre de signaux durant l'exécution. Les arcs du graphe montrent les dépendances entre les nœuds,i.e. nœuds de services, de décisions ou d'évènements. Lors de l'appel d'un service Web composé, EFlow sélectionne automatiquement les services Web correspondant aux noeuds de services. Lors de la définition d'un service Web composé, l'utilisateur peut rechercher et définir les services Web qui seront utilisés pour chaque nœud de services.

[Shankar et al., 2002] présentent SWORD, un outil de composition basé sur le rule-based planning. SWORD raisonne sur des services décrits selon le modèle Entité-Relation. Il associe aux services des règles logiques de type Horn. La création d'un service web composite est ramenée à la spécification d'un état de départ et d'un état d'arrivée. Les états intermédiaires seront générés par le moteur de planification à partir des descriptions des services disponibles.

Dans [Boukhadra, 2011] l'auteur propose une plateforme pour la composition dynamique des services web sémantiques en se basant sur les techniques d'alignements des ontologies OWL-S et les mesures de similarités. Cette plateforme traite la prise en compte des hétérogénéités des données échangées entre les services Web sémantiques engagés dans une architecture SOA distribuée, et ouvert ,elle s'articule sur quatre phases :

- la première phase : la requête de l'utilisateur est exprimée à travers une interface interactive et conviviale du système.
- la deuxième phase : comporte le traitement et l'analyse de la requête exprimée par l'utilisateur;
- La troisième phase comprend la découverte des services Web sémantiques qui peuvent satisfaire la requête d'utilisateur;
- La quatrième phase consiste à proposer des plans pour la composition dynamique des services web sémantiques candidats d'une manière automatique en se basant sur la technique de planification.

[Charif, 2007] propose une approche dynamique pour la composition de services. L'approche étudiée s'appuie sur la collaboration décentralisée entre une collection de services, dont le but est d'atteindre un objectif donné (chorégraphie des services). En faite, modéliser des capacités de collaboration au niveau des services nécessite qu'ils soient dotés de propriétés d'autonomie, d'adaptabilité et d'interactivité. Ces propriétés, qui font défaut aux services web, ont par ailleurs été largement étudiées dans les communautés agent et multi-agent. L'approche proposée se base sur un modèle de coordination multi-agent, pour la chorégraphie de services Web, dans lequel les capacités de collaboration des services sont modélisées à l'aide d'agents introspectifs. Ces agents sont capables de raisonner sur leurs propres actions (ou sur les services qu'ils offrent), de décomposer dynamiquement une tâche en fonction de leurs compétences, et de se coordonner avec d'autres agents pour pallier leurs limites et couvrir les besoins à satisfaire. L'approche se déroule en quatre étapes :

- un utilisateur humain formule ses besoins en services à travers une interface graphique.
- un module de découverte de services extrait ensuite des mots clé de la requête de l'utilisateur pour découvrir des services candidats à la composition. Ceux-ci sont recherchés à partir d'un registre en ligne de services web.
- les services candidats tentent alors de satisfaire cette requête en s'envoyant des messages. C'est la phase de chorégraphie de services. Leurs interactions sont régies par un protocole de coordination.
- enfin, parmi les services ayant pu répondre à la requête de l'utilisateur, un sous-ensemble répondant aux contraintes globales énoncées par celui-ci est sélectionné.

[Châtel, 2010] son premier objectif est de favoriser la réactivité des systèmes répartis complexes, pour cela, il propose la mise en place d'un mécanisme de liaison tardive de services lors de l'exécution d'un processus métier, sur la base des valeurs courantes de QoS des services. Son deuxième objectif vise à améliorer la performance des systèmes répartis complexes en mettant en place un modèle qualitatif de préférences utilisateur employé pour maximiser l'utilité des liaisons entre producteurs et consommateurs de services. Ainsi sa contribution vise à mettre en œuvre une composition de services active, utile et agile en prenant en compte de multiples contraintes non-fonctionnelles. Une composition active sert à compléter efficacement les approches dynamiques classiques, pour en améliorer les capacités d'adaptation aux changements non-fonctionnels. Elle repose sur une transposition du principe

de liaison tardive au contexte des SOA, alors disposé pour intégrer la QoS courante des services à l'exécution. La composition utile est liée au formalisme LCP-net pour l'expression de préférences utilisateur. L'élicitation de préférences non-fonctionnelles, établies entre les propriétés de QoS des services et leurs valeurs, permet d'obtenir un ordre total ou quasi total sur chaque ensemble de services candidats lors de leur sélection. Enfin, la composition agile correspond à la « somme » des deux précédentes : il s'agit d'une utilisation habile des préférences utilisateurs LCP-net lors de la liaison tardive des services.

[Chalbabi, 2006] propose une approche de découverte de services web sémantiques à base d'agents prenant en compte le contexte de l'utilisateur dans une architecture SOA. Pour cela il a développé une ontologie pour modéliser le contexte de ce dernier, ce contexte est formé par un ensemble de paramètres (identité de l'utilisateur, ses préférences en terme de QOS du service web, le type de dispositif utilisé (PDA, ordinateur portable)). De plus il a proposé deux extensions aux niveaux de l'ontologie des services Web OWL-S en termes de paramètres qualités de service et paramètres de description des dispositifs supportant les données transmises par le service Web. Ces paramètres font référence à deux ontologies ; une ontologie de Qualité de Service (QoS) et une ontologie de dispositif de connexion. L'ontologie qualité de service permet de définir le contexte du service Web en termes de paramètres de QoS. L'ontologie de dispositif permet de définir les types et caractéristiques des dispositifs que prend en charge le service Web.

[Benatallah et al., 2006] présentent une approche pour automatiser la découverte des services web. Ils proposent de formaliser la découverte comme un problème de réécriture de concepts en logique descriptive. Ils décrivent un algorithme pour découvrir la meilleure couverture sémantique d'une requête par les concepts d'une ontologie donnée. Ils démontrent que le problème de découverte de la meilleure couverture sémantique présente des similitudes avec le problème de calcul à cout minimal des transversales minimes d'un hypergraphe attribué (computing the minimal transversals with minimum cost of a weighted hypergraph).

[Paolucci et al., 2002] présentent une approche de matching sémantique entre des services définis avec OWL-S. Ils décrivent un algorithme qui assure la correspondance entre les profiles des services décrits par des « service profile » et le besoin de l'utilisateur définit aussi par une structure « service profile ». Une fonction de classement permet de différencier les profils grâce à plusieurs degrés de correspondance. L'algorithme consiste d'abord à comparer

les paramètres de sortie des profils publiés avec ceux du profil requis ensuite il compare les paramètres d'entrées des profils publiés avec ceux du profil requis. Le service publié est considéré équivalent à la description recherchée si ses paramètres de sortie et ses paramètres d'entrée coïncident respectivement avec ceux du profil recherché. Les auteurs dénotent trois niveaux de match. Le premier est le match exact, il a lieu si les paramètres de sortie du profil recherché sont équivalents ou constituent une sous-classe des paramètres de sortie du profil publié. Le second niveau de match est le plugIn, il a lieu si les paramètres de sortie du profil recherché sont englobés (subsumed) par les paramètres de sortie du profil publié. Le troisième degré de match est le subsume, il a lieu quand les paramètres de sortie du profil requis englobent (subsume) les paramètres de sortie du profil publié, dans ce cas le service publié répond partiellement au besoin recherché. Finalement les auteurs notent que le processus de matching décrit n'est pas symétrique. En effet, Match(A, B) <> Match(B, A) parce que l'ordre dans lequel les paramètres d'entrée et de sortie sont comparés est important.

[Claro, 2006] propose un canevas appelé SPOC (Semantic based Planning Optimized Compositions) pour la composition automatique de services web dédiés à la réalisation de devis, il a été élaboré en quatre étapes :

- la découverte de services: cette phase permet d'identifier les services à partir d'une ontologie de services UDDI. Dans cette phase, SPOC utilise le web sémantique comme langage de description des services Web ou l'auteur propose une ontologie du domaine qui organise les services web selon une similarité sémantique. Une fois une tâche définie, le processus de découverte cherche dans cette ontologie les services Web qui la réalisent.
- la planification: cette phase concerne la composition automatique, elle permet d'ordonner les tâches à réaliser et de choisir les services web qui peuvent réaliser chaque tâche. Dans SPOC, Claro utilise JESS (Java Expert System Shell), qui est un moteur de règles de planification, pour déterminer les services Web qui participeront à la composition ainsi que leur ordonnancement.
- l'exécution: cette phase envoie une requête à chaque service pour obtenir des informations comme le coût, la durée,..etc.

- l'optimisation: cette phase a pour but de proposer à l'utilisateur un certain nombre de compositions optimisées selon des critères de qualité prédéfinis (e.g., le coût, le prix, le chiffre d'affaire, la réputation) en utilisant l'algorithme génétique NSGA-II (Non-Dominated Sorting Genetic Algorithm).

Dans [Talantikite et al., 2009]les auteurs proposent une approche automatique pour la découverte et la composition des services web sémantique ,dans laquelle un réseau d'interaction est crée ou les nœuds représentent les services web sémantiques et les liens d'interaction, appelés « liens de dépendance sémantique », sont calculés en utilisant les opérateurs de mise en correspondance sémantiques introduits par [Paolucci et al 2002].Sachant qu'ils cherchent avant tout à satisfaire les buts de la requête, ils proposent l'utilisation d'un algorithme de chaînage arrière à une passe pour la recherche automatique de compositions. Une solution est sélectionnée parmi l'ensemble des plans de compositions retournés à partir d'un critère de qualité. Pour la partie expérimentale, les performances de l'algorithme sont testées sur un réseau construit à partir d'une collection de 60 descriptions artificielles. Dans ce travail, les auteurs se concentrent sur des descriptions sémantiques. Les données considérées sont artificielles et la collection de descriptions est de petite taille.

[Gharzouli, 2011] l'objectif principal de son contribution est de fournir un scénario purement décentralisé qui supporte la composition automatique des Web services sémantiques distribués sur un réseau P2P. A partir d'un réseau purement non structuré, il cherche à composer un Web service qui répond à une requête particulière. Ce service peut être composé de plusieurs services appartenant à différents pairs afin de créer un espace collaboratif entre les différents pairs participants à la réalisation d'un but commun. Il a développé un algorithme épidémique qui permet de rechercher des services distribués sur tous les pairs du réseau afin de composer de nouveaux services personnalisés. Aussi, dans cette solution, il a utilisé une table –dite table de composition- afin de préserver la trace du chemin de composition pour une éventuelle future réutilisation. Cette table est considérée comme une mémoire cache utilisée pour accélérer la recherche des Web services déjà composés.

[Chabeb, 2011] sa première contribution consiste à proposer un langage de description sémantique des services web nommé YAZAWSDL : Yet Another Semantic Annotation for WSDL (en plus court YASA), qui est une extension de SAWSDL en définissant deux types d'ontologies. Une première ontologie dite "ontologie technique" contenant des concepts définissant des sémantiques -fonctionnelles et/ou non fonctionnelles- de services et une deuxième ontologie dite "ontologie de domaine" contenant des concepts définissant les sémantiques métiers des services. Sa deuxième contribution consiste à proposer une approche de découverte sémantique basée sur un algorithme d'appariement et d'agrégation sémantiques. L'algorithme d'appariement offre des techniques d'appariement qui parcourt la plupart des éléments des deux services, requis et offert, avec un mécanisme d'appariement qui identifie précisément et fidèlement la similarité entre les éléments des deux services, requis et offert.

[El Falou, 2010] Le cœur de son travail consiste à fournir une architecture multi-agents de composition de services fondée sur les techniques de planification distribuée. Dans son architecture, chaque agent raisonne sur ses services afin de déterminer le meilleur plan local pouvant satisfaire en partie l'utilisateur. Ensuite, les agents se coordonnent et fusionnent leurs plans locaux afin d'atteindre un but commun prédéfini par l'utilisateur, créant ainsi un plan global représentant une composition possible de leurs services. Une autre contribution de son travail consiste à intégrer au processus de planification, un processus de diagnostic actif et de réparation. Le but est de détecter les fautes pendant l'exécution du service composite, de calculer l'ensemble des états fautifs possibles et de réutiliser le processus de planification pour raffiner les états et réparer les fautes.

[Lécué et al., 2006] proposent un modèle formel pour la composition de services Web sémantiques. Les correspondances sémantiques entre services Web sont assimilées à des liens causaux entre opérations (I/O) où ces derniers sont pondérés par les degrés de correspondance sémantique ("exact", "plug-in", "Subsume", "disjunction") telles celles introduites par [Paolucci et al., 2002]. Leur modèle de composition faisant appel à la planification en IA suit une méthode de chaînage arrière des services Web. Une originalité de la méthode de composition tient dans l'optimisation apportée par un pré-calcul des correspondances sémantiques des entrées et des sorties de tous les services web découverts et compatibles avec le but global à atteindre, dans une matrice dite des « liens causaux » qui ne conserve que les liens entrée-sortie sémantiquement compatibles.

Partant du but à atteindre et de la matrice de "liens causaux", l'algorithme de chaînage arrière proposé agit récursivement sur une liste de services web liés par les liens découverts i.e., pour chaque service de la liste, une nouvelle liste est découverte de telle sorte que les paramètres de sortie correspondent sémantiquement aux paramètres d'entrées. Une fois les différentes chaînes (ou plan) de composition possible trouvées, ils proposent de calculer la composition optimale en termes de correspondance sémantique entre les services web, qu'ils peuvent étendre aisément à la prise en compte de propriétés non fonctionnelles (QoS, Coût, …). Cette méthode ne tient pas compte les effets et les pré-conditions.

[Canfora et al., 2005] propose une approche pour la composition des services web basé sur un algorithme génétique. La composition y est réalisée sur la base de caractéristiques à la frontière du fonctionnel et du non-fonctionnel (Qos). Elle se décline, d'un point de vue architectural, en un évaluateur de QoS ("QoS evaluator") et un composant de liaison tardive ("late binding").Etant donné une composition S (spécifié par un langage de description de workflow (par ex BPEL4WS))de n services abstraits S=(S1,S2,……Sn), chaque service abstrait Si peut être lié à un service parmi les m services concrets qui sont fonctionnellement équivalents. Un service concret est alors sélectionné pour l'affectation globale si sa QoS propre, ajoutée à celle des précédents choix, ne viole pas les contraintes globales imposées par le processus. Pour cela il fait appel à des fonctions d'agrégation de QoS qui permettent d'obtenir la valeur globale pour un processus d'une propriété de QoS à partir des multiples QoS locales des services.

4.6 Etude comparative des approches de composition/découverte

Dans cette partie nous allons présenter une étude comparative de quelques approches de composition. Nous commençons d'abord par décrire quelques critères qui nous ont permis d'effectuer cette comparaison.

- **Automatisation**

D'après notre étude des différentes approches de composition, on a bien constaté que la génération du schéma de composition doit être au moins partiellement (s'il n'est pas entièrement) automatique afin de minimiser l'intervention de l'utilisateur et d'accélérer le processus de production d'un service composite qui satisfait les besoins des utilisateurs prédéfinis.

- **Dynamicité**

Une caractéristique essentielle d'une approche de composition de service est de savoir si elle produit un schéma de composition statique ou dynamique.

- **Sémantisation**

Cette caractéristique permet de spécifier si une approche de composition prend ou non en compte la dimension sémantique dans la description des services web.

- **Qualité de service (QoS)**

« QoS-aware approches » prennent en compte non seulement les propriétés fonctionnelles des services, mais aussi non fonctionnelles, traitant des aspects de qualité tel que le temps de réponse, le prix, la disponibilité, etc. Considérant les aspects de qualité de service au moment de décider quels services à inclure dans un schéma de composition des services est important lorsque les exigences fonctionnelles sont satisfaites par plus d'un service.

- **Stratégie de la découverte/ composition**

Cette caractéristique permet de spécifier la technique adoptée par l'approche pour réaliser la découverte ou la composition des services (Planification, SMA,Workflows, etc.).

- **Le contexte**

On trouve dans la littérature beaucoup de définitions de contexte, [Dey et al., 1999] définissent le contexte comme étant : " toutes les informations pouvant être utilisées pour caractériser la situation d'une entité, où une entité est une personne, un lieu, ou un objet qui peut être pertinent pour l'interaction entre l'utilisateur et l'application, y compris l'utilisateur et l'application eux-mêmes". Par cette définition chacune des deux entités, utilisateur et service Web, a son propre contexte. Le contexte d'un service Web peut grouper la localisation du service (restriction d'usage géographique du service), le coût d'utilisation, la catégorie de service, etc. Le contexte de l'utilisateur peut être formé de la localisation de l'utilisateur, son profil,..etc.

Le tableau 4.1 montre la comparaison des différentes approches étudiées selon les critères de qualité qu'on a défini.

Travaux étudiés	Automatisation	Dynamicité	Sémantisation	QoS	Stratégie de la découverte/composition	Contexte
[Chouchani, 2010]	Semi-automatique	Dynamique	Des descriptions WSDL (pas de sémantique)	-	Composition basée sur un algorithme génétique	-
[Weise et al 2007]	Automatique	Dynamique	Des descriptions WSDL annotés par des concepts ontologiques	-	Composition basée sur un I/O matching dans les trois algorithmes proposés	-
[Casati et al 2001]	Manuelle	Statique	Pas de descriptions sémantiques des services web	-	Composition basée sur les workflows	-
[Shankar et al 2002]	Automatique	Statique	Pas de descriptions sémantiques des services web	-	Composition adoptant une technique de planification basée sur les règles	-
[Alkamari 2008]	Semi-automatique	Dynamique	Des services WSDL (Pas de descriptions sémantiques des services web).	-	Composition basée sur l'appariement des signatures	-
[Boukhadra 2011]	Automatique	Dynamique	Des services OWL-S	-	Composition basée sur la planification	-
[Charif 2007]	Semi-automatique	Dynamique	Pas de descriptions sémantiques des services web	+	Composition basée sur un modèle de coordination multi-agents	-
[Châtel 2010]	Automatique	Dynamique	Pas de descriptions sémantiques des services web	+	Composition basée sur un mécanisme de liaison tardive des services web	-
[Chalbabi 2006]	Automatique	Dynamique	Des services OWL-S	+	Découverte basée sur une ontologie du contexte de l'utilisateur	de l'utilisateur (sa localisation, son profil,..)
[Benatallah et al 2005]	Automatique	Dynamique	Non spécifié	-	Découverte basée sur un mécanisme de réécriture de concepts en logique descriptive.	-
[Paolucci et al 2002]	Automatique	Dynamique	Des descriptions DAML	-	Découverte basée sur un I/O matching	-
[Claro 2006]	Automatique	Dynamique	Non spécifié	+	Composition basée sur la planification	-
[Talantikite et al 2009]	Automatique	Statique	Des descriptions OWL-S	+	Composition basée sur un I/O matching	-
[Gharzouli 2011]	Automatique	Statique	Des descriptions OWL-S	-	Composition basée sur un algorithme épidémique dans un réseau P2P	-
[Chabeb 2011]	Automatique	Dynamique	Des descriptions YAZAWSDL	-	Découverte basée sur un I/O matching	-
[El Falou 2010]	Automatique	Dynamique	Des descriptions OWL-S	-	Composition basée sur une approche multi-agents	-
[Lécué et al 2006]	Automatique	Dynamique	Des descriptions OWL-S	-	Composition basée sur un I/O matching et une technique de planification	-
[Canfora et al 2005]	Semi-automatique	Statique	Pas de descriptions sémantiques des services web	+	Composition basée sur un algorithme génétique	-

Tableau 4.1 : Tableau comparative des travaux de découverte/ composition de services Web

4.7 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur la composition et la découverte des services web où on a examiné les différents travaux présentés. Cette étude comparative a démontré le besoin d'une approche de composition plus précise basée sur une expressivité sémantique plus explicite. Dans le chapitre suivant nous allons décrire notre approche qui utilise les technologies du web sémantique et celles des algorithmes génétiques pour offrir une solution permettant la composition des services web sémantiques.

Chapitre5 : Composition Automatique de Services Web Sémantiques à base des AG's

5.1 Introduction

L'objectif de la composition des services web est de créer de nouvelles fonctionnalités en combinant des fonctionnalités offertes par d'autres services existants, composés ou non en vue d'apporter une valeur ajoutée en fonction d'une requête de l'utilisateur. Contrairement aux processus métier dans un environnement statique, les services web composés s'exécutent dans un environnement versatile où le nombre de services disponibles évolue très rapidement. Afin d'automatiser ce processus, il est primordial d'expliciter la sémantique dans les descriptions des services web soit par l'ajout d'annotations sémantiques ou en utilisant des ontologies de haut niveau tels qu'OWL-S ou WSMO, évitant ainsi les problèmes d'hétérogénéité sémantique qui peuvent survenir.

Notre contribution consiste à proposer une approche pour la composition dynamique des services web sémantiques basée sur un algorithme génétique, ainsi nous modélisons ce problème comme un processus d'optimisation, le but est de trouver une solution (une combinaison de services web) qui répond au mieux à la requête de l'utilisateur.

Pour évaluer le degré de similarité entre une requête donnée et un service composite nous proposons un mécanisme de matching (appariement) sémantique basé sur les concepts ontologiques annotant les paramètres d'entrée/sortie des services web participant à la composition ainsi que les concepts ontologiques annotant les paramètres d'E/S de la requête.

Il est possible d'avoir des services web composites équivalents d'un point de vue fonctionnel (c.-à-d. même similarité sémantique) mais qui sont différents d'un point de vue non fonctionnel (QoS), pour prendre en compte ces considérations nous proposons d'intégrer les propriétés non fonctionnelles dans le processus de composition. Notre approche prend aussi en considération les exigences de l'utilisateur exprimés par des contraintes globales non fonctionnelles, par exemple le cout global du service composite doit être supérieur à 20$ et inférieur à 100$.

Ce chapitre est organisé comme suit : nous commençons d'abord par présenter notre approche de composition basée sur la technique des AG's, ensuite nous présentons notre modèle sémantique qui décrit le mécanisme de matching sémantique. Par la suite nous présentons le modèle QoS qui définit la valeur globale QoS de service composite. Dans la dernière section nous présentons notre AG en détail (codage, opérateurs génétiques, fonction fitness,…..).

5.2 Architecture de composition proposée

Notre approche de composition se base sur un algorithme génétique mono-objectif qui prend en compte deux aspects [Bekkouche et al., 2012a]:

- **L'aspect fonctionnel :** où les descriptions sémantiques des services web sont exploitées pour calculer le degré de correspondance sémantique entre la requête de l'utilisateur et une solution candidate (service composite).
- **L'aspect non fonctionnel :** où les valeurs QoS des services composants sont utilisées pour calculer la valeur globale QoS de la solution candidate.

Notre intérêt pour les algorithmes génétiques est du à deux raisons :

- Les algorithmes génétiques ont prouvés leur efficacité dans divers domaines pour résoudre des problèmes d'optimisation NP hard (comme le problème de voyageur de commerce).
- Ils n'imposent aucune hypothèse sur la résolution d'un problème donné, ainsi leur principe consiste à définir une fonction objective (à optimiser) pour trouver une solution de bonne qualité.

L'AG reçoit en entrée les besoins de l'utilisateur exprimés via une interface graphique, ces besoins incluent une requête exprimée en termes de concepts d'E/S ainsi que des contraintes non fonctionnelles et il retourne comme résultat une solution quasi-optimale qui représente une composition des services web (voir Figure 5.1).

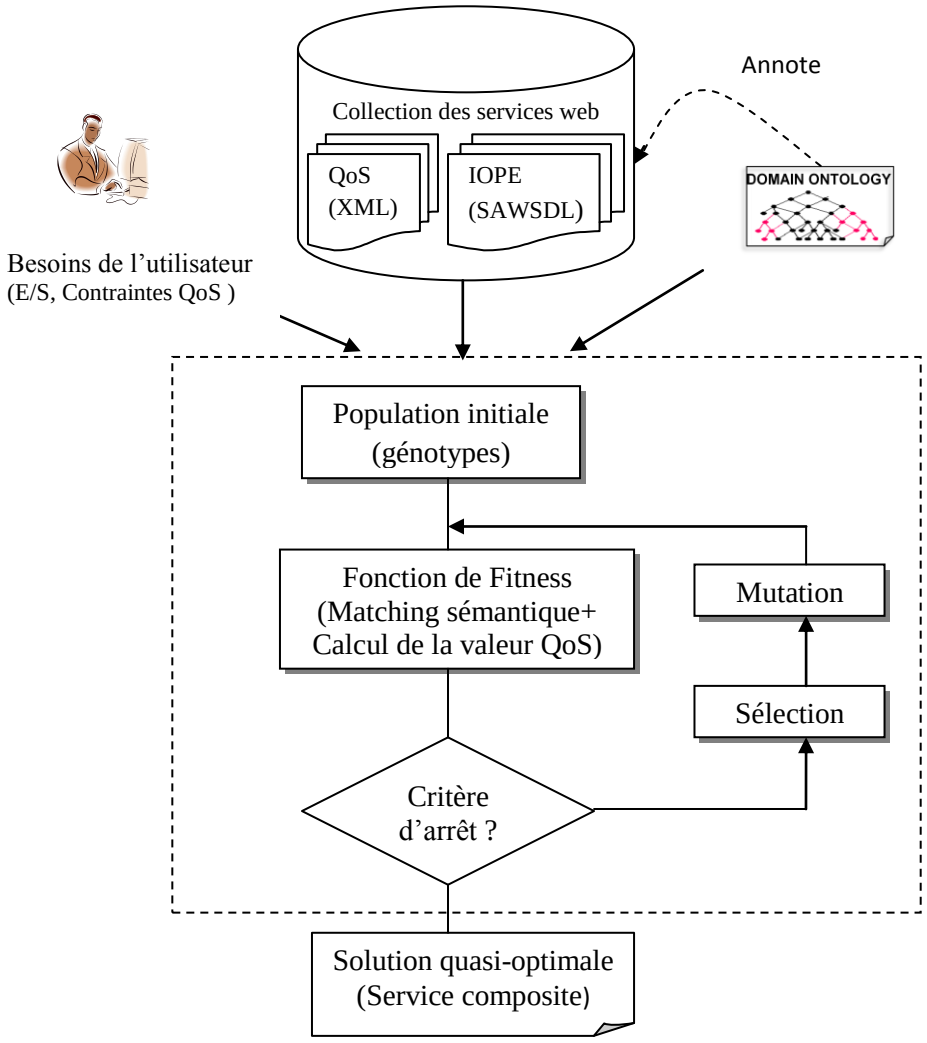

Figure 5.1 : Architecture de composition proposée

Les services web dans la collection sont décrits par le langage SAWSDL[1].Ce langage présente un mécanisme permettant d'annoter sémantiquement les services décrits avec WSDL et leurs schémas XML associés. Ces annotations sémantiques établissent la correspondance entre chaque élément d'une description WSDL (entrées, sorties, pré-conditions, effets) et les concepts d'une ontologie de référence. Une ontologie de référence (ex. OWL) permet de représenter un domaine par des structures interprétables par une machine. Elle contient la taxonomie des concepts qui sont décrits sous forme hiérarchique. Cette arborescence permet de déduire les relations de généralisation (subsomption) entre les concepts et qui sera utile lors de calcul de degré de similarité sémantique entre une requête donnée et un service composite.

[1] www.w3.org/TR/SAWSDL/

De plus un fichier XML qui contient les valeurs des attributs de QoS (temps de réponse, coût, disponibilité,…) est associé à chaque description de service web.

5.3 Le modèle sémantique (Modèle fonctionnel)

5.3.1 Représentation conceptuelle

Avant de parler de la similarité sémantique d'un service composite il est nécessaire de donner quelques définitions formelles d'un service web, un service composite ainsi que la requête de l'utilisateur.

Définition 1 (service web)

Un service web est caractérisé par des propriétés fonctionnelles et des propriétés non fonctionnelles .Formellement :$S = \{ID, CE, CS, q\}$

Où *ID* est l'identifier du service S, *CE* et *CS* sont les concepts d'entrée et de sortie respectivement, q est la liste des valeurs des attributs de QoS (coût, temps de réponse, disponibilité, etc.).

Définition 2 (service composite)

Un service web composite (SC) est définit par un ensemble de services web. Formellement $SC = \{S_1,\ S_2, \dots, S_n\}$ ou n est le nombre des services web.

Définition 3 (requête)

Les besoins de l'utilisateur sont définis comme suit:

$$\mathrm{R} = \{\ \mathrm{CE}, \mathrm{CS}, \mathrm{G}\}$$

Où *CE et CS* représentent respectivement les concepts d'entrée et de sortie de la requête R. *G* est la liste des contraintes globales non fonctionnelles, c'est un ensemble de relations de la forme « variable, opérateur de comparaison, constante ».Par exemple coût (SC)> 20$ et coût (SC) < 100$.

5.3.2 Calcul de degré de similarité sémantique d'un service composite

5.3.2.1 Introduction

Le degré de similarité sémantique entre une requête donnée et un service composite est déterminé en fonction de la sémantique des éléments des descriptions à apparier. Selon la littérature on a principalement trois approches de matching [Chabeb, 2011]:

- ***IO-matching*** *:* dit aussi "IO-matching de profil de service". Ce type d'appariement est déterminé à partir des données sémantiques des paramètres de service : les entrées : inputs (I) et les sorties : outputs (O). Ce type d'appariement est adopté dans [Paolucci et al., 2002], [Fan et al., 2005] et [Paolucci et al., 2004].
- ***PE-matching :*** Ce type d'appariement est déterminé à partir d'appariements sur des pré-conditions (P) et des effets (E) des services et des requêtes. Ce type d'appariement est adopté dans PCEM [Schumacher et al., 2008] avec des pré-conditions et des effets spécifiées en Prolog.
- ***IOPE-matching :***Ce type d'appariement est déterminé à partir d'appariements sur les données sémantiques des inputs (I), des outputs (O), des pré-conditions (P) et des effets (E) des services et des requêtes. Cet appariement est adopté dans [Jaeger et al., 2005], [keller et al., 2005] [Küster et al., 2008]et [Stollberg et al., 2007] .

Dans notre approche on a adopté un *IO-matching* qui est basé sur le raisonnement par *subsomption*. Ce mécanisme estime le degré de *similarité* entre la requête de l'utilisateur et le service composite.

Afin de réaliser le processus de l'appariement, nous utilisons les quatres types de matching *Exact, Plugin, Subsume et Fail* introduits dans [Paolucci et al., 2002], sachant que dans la découverte des services web, l'appariement est effectué sur les mêmes catégories de paramètres, c'est à dire sur les paires d'entrées, ou les paires de sorties. A l'opposé, pour la composition de web services, le problème est adressé par l'étude de l'appariement sur des paires distinctes de paramètres d'entrée et de sortie des différents services web.

- ***Exact*** **:** deux paramètres sont dits similaires si leurs concepts C1 et C2 sont équivalents : $\boldsymbol{C1 \equiv C2}$
- ***Plugin*** **:** deux paramètres sont dits similaires si le concept C1 du premier paramètre est subsumé par le concept C2 du deuxième paramètre(C1 est plus spécifique que C2): $\boldsymbol{C1 \subseteq C2}$.Considérons le fragment d'ontologie Figure 5.2 et deux services respectivement nommés **NiveauScolaire_ManuelBiologie** et **ManuelSecondaire_Prix**. Le premier service prend en entrée un niveau **secondaire** et fournit une liste de manuels de **biologie**. Le deuxième service prend en entrée tous les types de **manuels** scolaires et en fournit le **prix**.Le premier

service peut entrer en interaction avec le second dans une relation plugin car le concept manuel est plus général que le concept biologie.

- ***Subsume* :** le deux paramètres sont dits similaires si le concept C1 du premier paramètre est plus général que le concept C2 du deuxième paramètre (C1 subsume C2) :$\boldsymbol{C1 \supseteq C2}$. Considérons le même fragment d'ontologie Figure 5.2et deux services définis comme suit. Le premier service nommé **NiveauSecondaire_ManuelTechnologie** prend en entrée le niveau scolaire **secondaire** et fournit une liste de manuels de **technologie** (biologie, informatique, etc.).Le deuxième service nommé **ManuelInformatique_Prix** prend en entrée des manuels d'**informatique** et en fournit le **prix.** Le premier service peut entrer en interaction avec le second dans une relation subsume car le concept informatique est plus spécifique que le concept technologie.
- ***Fail* :** signifie qu'il n'y a aucune relation de subsomption entre les concepts. Soient les deux services suivants décrits sur la base de l'ontologie Figure 5.2. Le premier service **NiveauSecondaire_ManuelSecondaire** fournit tout type de **manuels** secondaires à partir du niveau fourni. Le deuxième service **ManuelUniversitaire_Tarif** fournit les **tarifs** de manuels **universitaires**. Dans cette situation, aucune interaction entre ces deux services n'est possible.

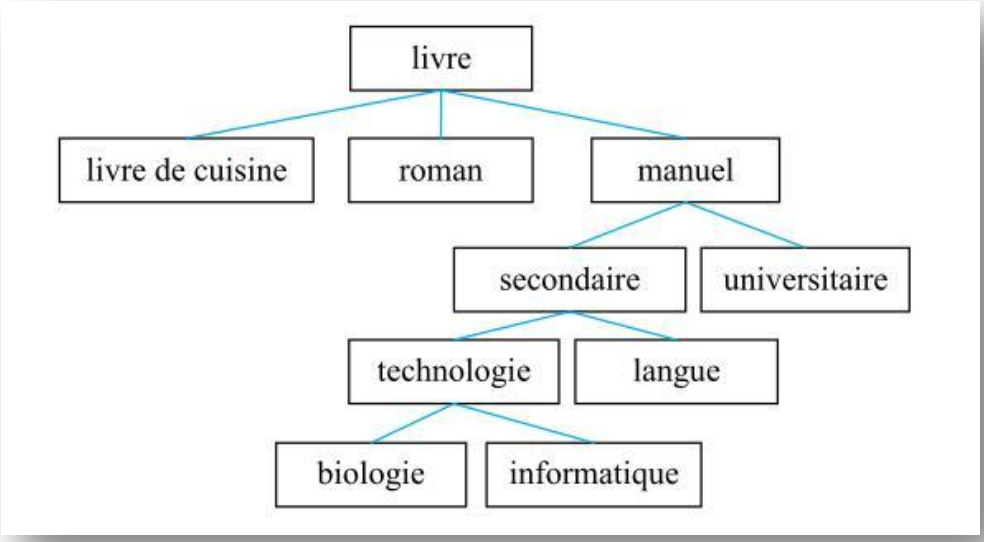

Figure 5.2 : Fragment d'ontologie relative aux livres

On définit 4 étapes pour obtenir la similarité sémantique entre une requête et une composition (SC) :

Etape 1 : calcul de la similarité entre le paramètre d'entrée (concept d'entrée) de la requête et le paramètre d'entrée (concept de sortie)de SC selon la fonction ME_{RIN} (définit dans la section suivante).

Etape 2 : calcul de la similarité entre le paramètre de sortie(concept de sortie) de la requête et le paramètre de sortie (concept de sortie)de SC selon la fonction ME_{ROUT} (définit dans la section suivante).

Etape 3 : calcul de la similarité entre les concepts d'E/S des services qui composent la solution selon la fonction ME_S (définit dans la section suivante).

Etape 4 : le matching sémantique global est calculé à partir des scores obtenus lors des étapes 1,2 et 3 (il est définit dans la section suivante).

Les fonctions précédentes utilisent une fonction nommé ***SIM*** qui permet de calculer la similarité entre deux concepts C1 et C2, son pseudo code est donné par la suite :

$$\boldsymbol{SIM(C1, C2)} = \begin{cases} \boldsymbol{Exact} & \boldsymbol{if\ \ C1 \equiv C2} \\ \boldsymbol{Plugin} & \boldsymbol{if\ \ C1 \subseteq C2} \\ \boldsymbol{Subsume} & \boldsymbol{if\ \ C1 \supseteq C2} \\ \boldsymbol{Fail} & \boldsymbol{Autrement} \end{cases} \quad \textbf{(1)}$$

5.3.2.2 Procédure de calcul de la similarité sémantique

Supposant qu'on a un service composite $SC = \{S_1,\ S_2, \dots, S_n\}$ et unerequêteR $= \{\mathrm{CE}, \mathrm{CS}, \mathrm{G}\}$,le calcul de la similarité sémantique passe par les étapes suivantes :

Etape 1 :

On calcule la similarité entre les concepts d'entrée de la requête et les concepts d'entrée de SC par la fonction ME_{RIN} /$ME_{RIN} = SIM(R.CE, S_1.CE)$.

Etape 2 :

On calcule la similarité entre les concepts de sortie de la requête et les concepts de sortie de SC par la fonction$\mathrm{ME_{ROUT}}$/ $\mathrm{ME_{ROUT}} = SIM(\mathrm{R.CS}, \mathrm{S_N.CS})$.

Etape 3 :

On calcule la similarité entre les concepts d'E/S des services composants par la fonction ME_S/ $ME_S = SIM(S_i.CS, S_{i+1}.CE)$.

Les fonctions $EM_S, EM_{RIN} et EM_{OUT}$ retournent des valeurs dans [0,1] en fonction de type de matching (exact, plugin, subsume, fail) entre les concepts. Dans le Tableau 5.1 nous récapitulons les fonctions de matching sémantique avec leurs scores.

Type de matching	**Exact**	**Plugin**	**Subsume**	**Fail**
$ME_S(S_i.CS, S_{i+1}.CE)$	1	2/3	1/3	0
Description Logique	*S_i.CS et S_{i+1}.CE sont équivalent*	*S_i.CS est subsumé par S_{i+1}.CE*	*S_i.CS subsume S_{i+1}.CE*	Autrement
$ME_{RIN}(R.CE, S_1.CE)$	1	2/3	1/3	0
Description Logique	*R.CE et S_1.CE sont équivalent*	*R.CE est subsumé par S_1.CE*	*R.CE subsume S_1.CE*	Autrement
$ME_{ROUT}(R.CS, S_N.CS)$	1	2/3	1/3	0
Description Logique	*R.CS et S_N.CS sont équivalent*	*R.CS est subsumé par S_N.CS*	*R.CS subsume S_N.CS*	Autrement

Tableau 5.1 : Les fonctions de matching sémantique

Etape 4 :

Le matching global entre la requête et la composition SC se fait en calculant la moyenne des scores des 3 premières fonctions (Equation (2)).

$$\boldsymbol{MG(SC) = ((ME_{RIN}(R.CE, S_1.CE) + \sum_{i=1}^{N} ME_S(S_i.CS, S_{i+1}.CE) + (ME_{ROUT}(R.CS, S_N.CS))/((N-1)+2)} \quad \textbf{(2)}$$

Où N représente le nombre de services composants.

Remarque : Dans le cadre de notre travail, nous nous limitons volontairement à une vision restrictive de matching sémantique qui suppose que tous les services et la requête

utilisent la même ontologie. Lorsque deux concepts appartiennent à la même ontologie, leur comparaison est facilement réalisée en exploitant la hiérarchie ontologique.

5.4 Modèle Qualité de Service (Modèle non fonctionnel)

Dans cette section, nous présentons d'abord les critères de qualité dans le contexte des services web élémentaires, avant de tourner notre attention vers des services web composites.

5.4.1 Critères de qualité des services web élémentaires

Les propriétés non fonctionnelles d'un service web sont exprimées par des attributs de QoS, nous distinguons les attributs génériques et les attributs non génériques [Zeng et al., 2003].

Attributs génériques :

- **Le temps de réponse :** le temps requis pour compléter une requête par un service web.
- **Le coût :** c'est le prix qu'un client du service doit payer pour bénéficier du service web.
- **La disponibilité :** la probabilité pour que le service est disponible à une certaine période de temps.
- **La réputation :** c'est une mesure de crédibilité d'un service web.
- **La fiabilité :** la capacité d'un service de répondre correctement à une requête.

Attributs non génériques :

C'est des attributs qui sont spécifiques à un domaine donné comme par exemple la bande passante pour des services web multimédia.

L'ensemble des attributs de QoS peuvent être devisé en deux catégories: attributs négatifs et attributs positifs. Les attributs positifs sont à maximiser (e.x.,la disponibilité,la fiabilité,..) tandis que les attributs négatifs sont à minimiser(e.x.,le temps de réponse, le coût,..).

On utilise un vecteur $q = \{q_1(S), q_2(S), \dots\dots\dots, q_l(S)\}$ pour représenter les attributs de QoS d'un service S, où $q_i(S)$ détermine la valeur de la i^{th} attribut de S.

5.4.2 Critères de qualité d'un service composite

Definition 4 (la QoS du service Composite)

Les critères de qualité d'un service composite peut être définit comme suit :

$$\boldsymbol{Q(SC) = \{Q_1(SC), \dots\dots\dots, Q_l(SC)\}} \qquad (\mathbf{3})$$

Où $Q_i(SC)$ représente la valeur de la i^{th} attribut de SC.

5.4.3 Calcul de la valeur QoS agrégée d'un service composite

La valeur QoS agrégée d'un service composite est calculé à partir des valeurs des attributs QoS de ses services composants et en fonction du modèle de la composition (séquentiel, parallèle, ..).Dans notre travail, on considère que le modèle séquentiel(dans une séquence, un service web est disponible une fois que les services web précédents ont terminés leur exécution).Donc la valeur QoS globale de SC est calculée par l'agrégation des valeurs QoS de ses composants[Cardoso, 2002].Le Tableau 5.2 montre trois exemples de fonctions d'agrégation des valeurs QoS.

Type d'agrégation	**Attributs QoS**	**Fonction d'agrégation**
Sommation	Temps de réponse	$Q(CS) = \sum_{i=1}^{N} S_i . q$
	Coût	
	Réputation	$Q(CS) = {}^{1}/_{N} \sum_{i=1}^{N} S_i . q$
Multiplication	Disponibilité	$Q(CS) = \prod_{i=1}^{N} S_i . q$
	Fiabilité	

Tableau 5.2: Les fonctions d'agrégation des attributs de QoS

5.4.4 Les contraintes globales non fonctionnelles

Les contraintes globales non fonctionnelles représentent les exigences de l'utilisateur en terme de QoS, ils peuvent être exprimés par les bornes inférieures (valeurs minimales) et les bornes supérieures (valeurs maximales) des valeurs QoS globales du service composite. Par exemple : Temps de réponse (SC)<100 ms et Temps de réponse (SC)>20 ms.

5.5 Formalisation du problème

Notre objectif est de trouver une composition $SC = \{S_1, S_2, \dots, S_n\}$ tel que :

- $MG(SC)$est maximisée.
- Les valeurs des attributs de QoS de SC sont maximisées.
- Les contraintes globales de l'utilisateur sont satisfaites.

5.6 L'Algorithme Génétique

5.6.1 Les paramètres de l'AG

La solution quasi-optimale (représenté par un chromosome) est déterminé par l'évolution d'une population initiale à travers un nombre déterminé de générations en appliquant des opérateurs génétiques jusqu'à ce qu'un critère d'arrêt soit satisfait .Dans ce qui suit nous détaillons tous ces paramètres.

5.6.1.1 Le codage

Le chromosome (service composite) est définit par une liste de tableaux .Le nombre de tableaux est égal au nombre de services web composants et chaque tableau (gène) représente un service web qui est décrit par son identificateur, son concept d'entrée, son concept de sortie ainsi que ses attributs de QoS. La Figure 5.3 montre le codage d'un SC avec 3 services et deux attributs de QoS (le coût et le temps de réponse).

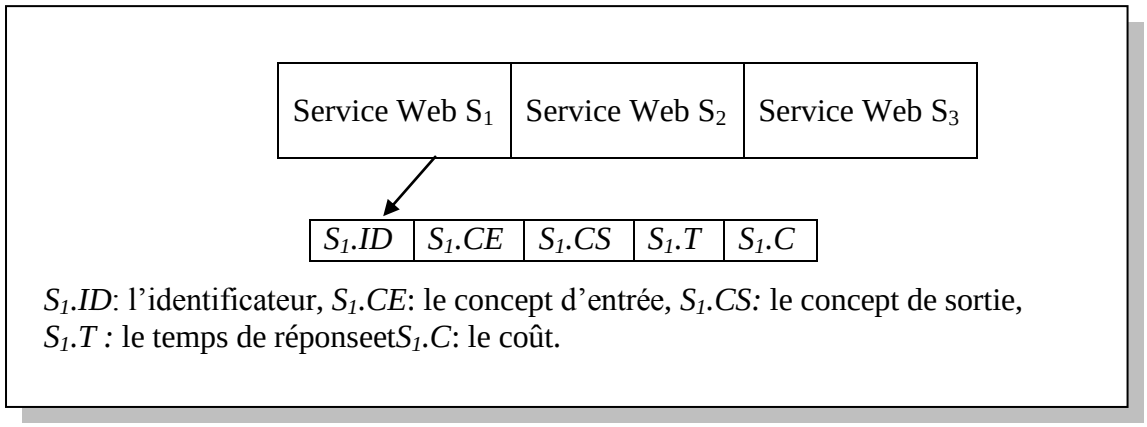

Figure 5.3: Codage d'un service composite

5.6.1.2 Population initiale

Elle est constituée d'un ensemble de compositions (caractérisées par ses chromosomes) ou les services web composants sont sélectionnés aléatoirement.

5.6.1.3 Fonction Fitness (fonction objective)

La fonction fitness joue un rôle prépondérant dans le déroulement de l'AG, elle sert à maximiser la similarité sémantique et la valeur QoS globale du SC (les attributs négatifs de QoS sont convertis en des attributs positifs en multipliant leur valeurs par (-1) afin d'obtenir une seule fonction objective à maximiser).Elle est définit par l'équation(4).

$$\boldsymbol{F(SC) = W_s MG(SC) + \sum_{i=1}^{l} W_i Q'_i(SC)} \qquad (\mathbf{4})$$

Où l est le nombre des attributs de QoS.

$W_k \in [0, 1]$ est le poids assigné à la k^{th} facteur de fitness et $\sum W_k = 1$.

Q'_iest la valeur normalisée[2] dans [0,1] de la i^{th} attribut de QoS de SC (Q_i), elle est définit comme suit :

$$\boldsymbol{Q'_i = \frac{Q_i - \sum_{j=1}^{N} min\ S_j . q_i}{\sum_{j=1}^{N} max\ S_j . q_i - \sum_{j=1}^{N} min\ S_j . q_i}} \qquad (\mathbf{5})$$

Où $min\ S_j . q_i$et$max\ S_j . q_i$ sont respectivement la valeur minimale et la valeur maximale de la i^{th} attribut QoS du service Sj ; N : nombre de services composants.

De cette façon, l'utilisateur peut ajuster les poids des facteurs de la fonction fitness selon ses préférences.

De plus la fonction fitness$\boldsymbol{F(SC)}$doit favoriser l'évolution vers la satisfaction des contraintes globales, ainsi les compositions qui ne satisfont pas ces contraintes sont pénalisées. Beaucoup de fonctions de pénalité sont proposées dans la littérature (statique, dynamique, adaptative,..).Nous avons choisi la fonction adoptée par [Lécué, 2009].

[2] Normalisation : les attributs de QoS utilisent des unités de mesure différentes , par exemple le temps de réponse en ms, le coût en dollars ,il est nécessaire de normaliser ces valeurs pour calculer une valeur globale unique de QoS du SC.

$$\boldsymbol{F'(SC) = F(SC) - W_p \sum_{i=1}^{l} \left(\frac{\Delta Q'_i}{Q'_i{}^{max}(SC) - Q'_i{}^{min}(SC)}\right)^2} \qquad (6)$$

Où $Q'_i{}^{max}, Q'_i{}^{min}$ sont respectivement la valeur minimale (borne inférieure) et la valeur maximale (borne supérieure) de la i^{th} contrainte globale, l est le nombre des attributs QoS, W_p est le poids assigné à la fonction pénalité, et $\Delta Q'_i$ est définit par :

$$\Delta \boldsymbol{Q'_i} = \begin{cases} \boldsymbol{Q'_i - Q'_i{}^{max}} & \boldsymbol{if\ Q'_i > Q'_i{}^{max}} \\ \boldsymbol{0} & \boldsymbol{if\ Q'_i{}^{min} \leq Q'_i \leq Q'_i{}^{max}} \\ \boldsymbol{Q'_i{}^{min} - Q'_i} & \boldsymbol{if\ Q'_i < Q'_i{}^{min}} \end{cases} \qquad (7)$$

5.6.1.4 Les opérateurs génétiques

La mutation

Nous avons spécifié un opérateur de mutation spécialisée inspirée de celle décrite par [Weise et al., 2007].Cet opérateur soit il supprime un service web d'un service composite SC (mutate1) ou il remplace un service web dans SC par un autre service web tel que $MG(SC)$ est non nul en assurant ainsi une similarité minimale avec la requête de l'utilisateur (mutate2).Contrairement à [Weise et al., 2007], les services web supprimés et remplacés sont choisis aléatoirement. Nous utilisons une variable d'ajustement pour appliquer mutate1 ou mutate2 suivant un seuil σ.

$$\mathbf{mutate_1(SC)} \equiv \begin{cases} \mathbf{SC'} = \{\mathbf{s_1, s_2, \dots, s_{|SC|}}\} - \mathbf{s_{1 \leq i \leq |SC|}} & \mathbf{if |SC| > 1} \\ \mathbf{SC} & \mathbf{Autrement} \end{cases} \qquad (8)$$

$$\mathbf{mutate_2(SC)} \equiv \mathbf{SC'} + \mathbf{s_{1 \leq i \leq |SC|}} \mid \mathbf{MG(SC'} + \mathbf{s_{1 \leq i \leq |SC|}) > 0} \qquad (9)$$

$$\boldsymbol{mutate\ (CS)} \equiv \begin{cases} \boldsymbol{mutate_1(CS)} & \boldsymbol{if\ random() > \sigma} \\ \boldsymbol{mutate_2(CS)} & \boldsymbol{Autrement} \end{cases} \qquad (10)$$

La sélection

Cet opérateur permet d'identifier statistiquement les meilleurs individus (solutions candidates) de la population courante qui seront autorisés à se reproduire, en fonction de leur valeur de fitness. Dans notre AG on a appliqué une sélection par tournoi binaire.

5.6.1.5 Critère d'arrêt

L'AG s'exécute tant que le nombre de génération (nbr_gen) est inférieur à nbr_max ou jusqu'à la stagnation de la population, formellement :

On arrête si les scores de la fonction fitness F' sont stables c.-à-d. :

$$\left|F'\left(SC_{popt}\right) - F'\left(SC_{popt+1}\right)\right| \leq \varepsilon$$

Où $F'\left(SC_{popt}\right)$: fitness du meilleur chromosome de la population t,

$F'(SC_{popt+1})$:fitness du meilleur chromosome de la population t+1 et εest une constante positive.

5.6.2 Pseudo code de l'AG

L'algorithme de la Figure 5.4 décrit notre algorithme génétique.

Algorithme génétique

Début

Créer la population initiale

Evaluer chaque individu par la fonction fitness définit par l'équation (6)

Tantque (nbr_gen<nbr_max) ou ($\left|\boldsymbol{F'}\left(\boldsymbol{SC_{popt}}\right) - \boldsymbol{F'}(\boldsymbol{SC_{popt+1}})\right| > \varepsilon$**)** **faire**

Sélectionner des individus pour la reproduction.

Appliquer la mutation (mutate 1 et muate 2) selon le seuil σ.

Evaluer de nouveaux les individus selon l'équation (6).

Fin Tant que

Fin

Figure 5.4 : Notre algorithme génétique

5.7 Conclusion

Dans ce chapitre, nous avons proposé une approche de composition dynamique des services web sémantiques en prenant en compte à la fois les propriétés fonctionnelles et les propriétés non fonctionnelles. Afin de valider notre approche, nous avons implémenté un prototype qui fera l'objet du chapitre suivant.

Chapitre6 : Implémentation et Expérimentations

6.1 Introduction

Nous venons de détailler dans le chapitre précédent l'architecture de composition proposée. Dans le présent chapitre nous décrivons un prototype implémentant cette architecture ainsi que les résultats obtenus.

Nous allons commencer par présenter les différents outils techniques liés à l'implémentation, ensuite nous présentons notre prototype baptisé « **GeneticComposer** » [Bekkouche et al., 2012b] ainsi que les expérimentations menées pour analyser notre approche.

6.2 Présentation des outils technologiques utilisés

Le prototype a été développé sur un Intel Core2duo, avec une vitesse de 1.6 GHZ, doté d'une capacité mémoire de 1GB de RAM sous Windows XP en utilisant des outils Open source graphiques et développés en JAVA. Nous détaillons, dans ce qui suit, chacun des outils et langages utilisés pour la manipulation des données ainsi que l'implémentation de l'interface utilisateur.

- **Le langage JAVA**

Pour le langage de programmation notre choix s'est porté sur le langage JAVA, et cela parce que JAVA est un langage orienté objet simple ce qui réduit les risques d'incohérence; il est portable, il peut être utilisé sous Windows, sous Linux, sous Macintosh et sur d'autres plateformes sans aucune modification , enfin il possède une riche bibliothèque de classes comprenant des fonctions diverses telles que les fonctions standards, le système de gestion de fichiers , les fonctions multimédia et beaucoup d'autres fonctionnalités;

- **Eclipse Indigo**

Pour le choix de l'environnement de développement, on a opté pour Eclipse car il possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plugins;

- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS;
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT;

- **L'API JDOM**

JDOM est une API open source Java, vue comme un modèle de documents objets dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. JDOM propose aussi une intégration de SAX, DOM, XSLT et XPath.

Un document XML est encapsulé dans un objet de type Document. Les éléments d'un document sont encapsulés dans des classes dédiées : Element, Attribute, Text, Processing Instruction, Namespace, Comment, DocType, EntityRef, CDATA. JDOM permet aussi de vérifier que les données contenues dans les éléments respectent la norme XML.

JDOM propose des réponses à certaines faiblesses de SAX et DOM. La simplicité d'utilisation de JDOM lui permet d'être une API dont l'utilisation est assez répandue.

- **L'API Pellet**

Cette API permet le raisonnement sur les ontologies formalisées avec OWL, elle offre des mécanismes d'inférences basés sur les logiques de description.

6.3 Présentation de la base des services web et l'ontologie OWL

La base utilisée dans nos expérimentations est SAWSDL-TC 3 (SAWSDL Service Retrieval Test Collection) que nous avons importé à partir du site web http://www.semwebcentral.org. Cette base comporte 1080 services web sémantiques décrits en SAWSDL ainsi que l'ontologie de référence OWL qui contient la taxonomie des concepts.

Par la suite nous avons effectué des prétraitements sur les services web et l'ontologie de cette base pour en extraire des descriptions XML selon les modèles montrés dans la Figure 6.1 et la Figure 6.2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<base>
<service num="serv0">
<input>con2</input>
<output>con3</output>
<cost>150</cost>
<time>100</time>
</service>
<service  num="serv1">
<input>con4</input>
<output>con3</output>
<cost>200</cost>
<time>50</time>
</service>
<service num="serv2">
<input>con0</input>
<output>con4</output>
<cost>50</cost>
<time>80</time>
</service>
<service  num="serv3">
<input>con4</input>
<output>con5</output>
<cost>300</cost>
<time>20</time>
</service>
```

Figure 6.1 : Le modèle XML des services web

- Les balises <*input*> …… </*input*>et <*output*> …… </*output*> contiennent respectivement les concepts d'entrée et de sortie du service identifié par l'identificateur contenu dans l'attribut *num.*
- Les balises<*cost*> …… </*cost*> et <*time*> …… </*time*> contiennent respectivement la valeur de l'attribut 'coût' et la valeur de l'attribut 'temps de réponse', ces valeurs sont générés aléatoirement [Yu et al., 2010] de telle façon que :
  -Les valeurs du coût (en $) sont prises dans l'intervalle [0,300].
  -Les valeurs du temps de réponse (en ms) sont prises dans l'intervalle [0,100].

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http:
  <owl:Ontology rdf:about="" />
<owl:Class rdf:ID="con0"/>

<owl:Class rdf:ID="con1">
    <rdfs:subClassOf rdf:resource="#con0" />
  </owl:Class>
<owl:Class rdf:ID="con2">
    <rdfs:subClassOf rdf:resource="#con0" />
  </owl:Class>
<owl:Class rdf:ID="con5">
    <rdfs:subClassOf rdf:resource="#con1" />
  </owl:Class>
<owl:Class rdf:ID="con3">
    <rdfs:subClassOf rdf:resource="#con1" />
  </owl:Class>
<owl:Class rdf:ID="con3">
    <rdfs:subClassOf rdf:resource="#con2" />
  </owl:Class>
<owl:Class rdf:ID="con4">
    <rdfs:subClassOf rdf:resource="#con2" />
  </owl:Class>
<owl:Class rdf:ID="con5">
    <rdfs:subClassOf rdf:resource="#con1" />
  </owl:Class>
<owl:Class rdf:ID="con5">
    <rdfs:subClassOf rdf:resource="#con4" />
  </owl:Class>
 <owl:Thing rdf:ID="inst5">
    <rdf:type rdf:resource="#con0" />
  </owl:Thing>
```

Figure 6.2 : Le modèle XML de l'ontologie OWL

6.4 Description du prototype

Afin d'illustrer les fonctionnalités de notre prototype « Genetic Composer », nous avons effectué quelques prises d'écrans montrant les différentes étapes nécessaires à la réalisation d'une composition automatique des services web, qui sont :

Chargement de la base des services web et l'ontologie de domaine

La première étape consiste à charger la base des services web ainsi que l'ontologie de domaine (Figure 6.3).

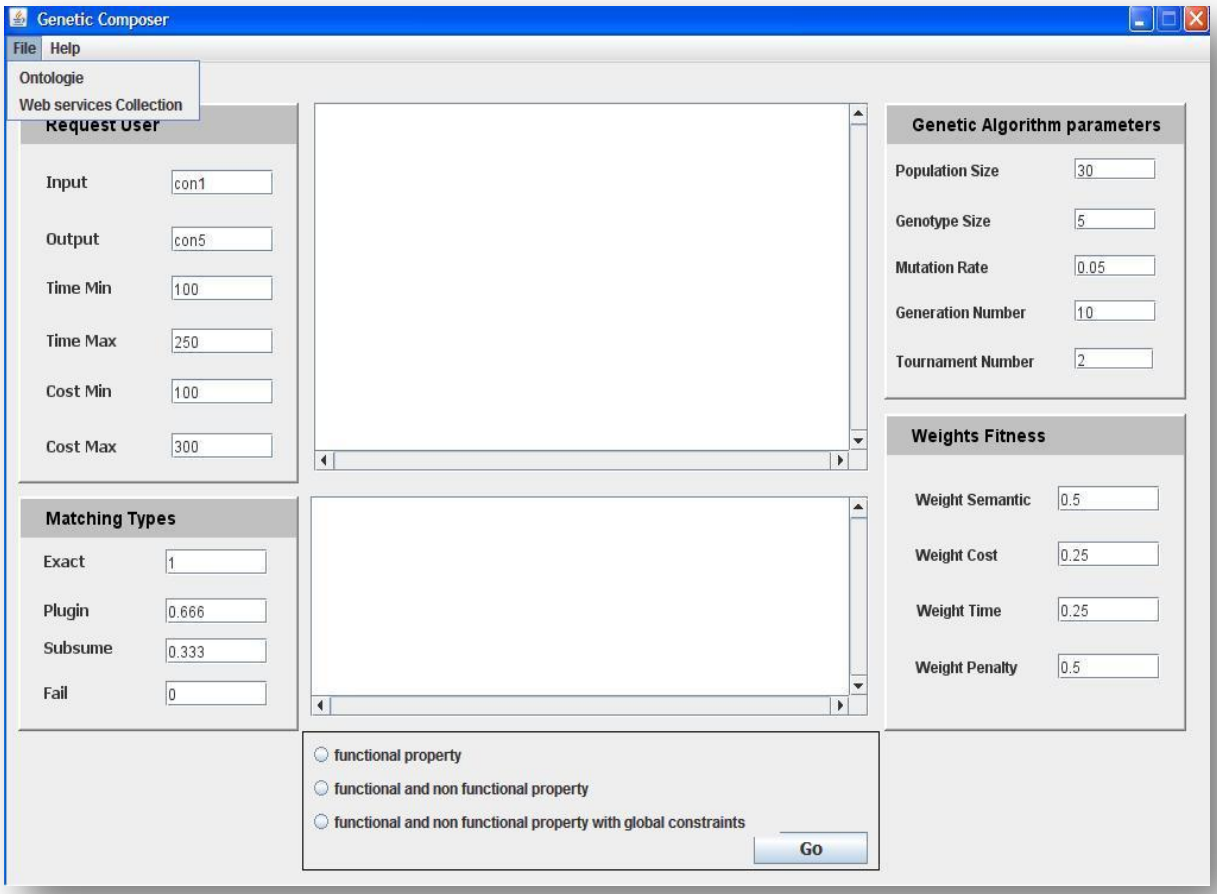

Figure 6.3 : Interface de prototype

Construction de la requête

Une fois que la base des services et l'ontologie OWL sont spécifiées et chargées, l'utilisateur pourra formuler sa requête. Il introduit les concepts d'E/S et les contraintes globales non fonctionnelles spécifiées par la borne inférieure(valeur minimale)et la borne supérieure(valeur maximale) voulus de l'attribut QoS de la composition. Dans le cadre de ce travail nous avons considéré deux attributs : le temps de réponse et le coût.

Paramétrage de l'application

A ce niveau, on peut ajuster les paramètres de l'AG (taille de la population, nombre de générations, taille de la composition désirée, le seuil de la mutation et le nombre de tournois)comme on peut aussi ajuster (selon nos préférences) les poids associés au différents facteurs de la fonction fitness (weight semantic,weight cost,weight time,weight penalty).

A la fin de cette étape, l'AG est exécuté et une solution quasi-optimale (combinaison de services web) est retournée. La Figure 6.4 montre une exécution de l'AG.

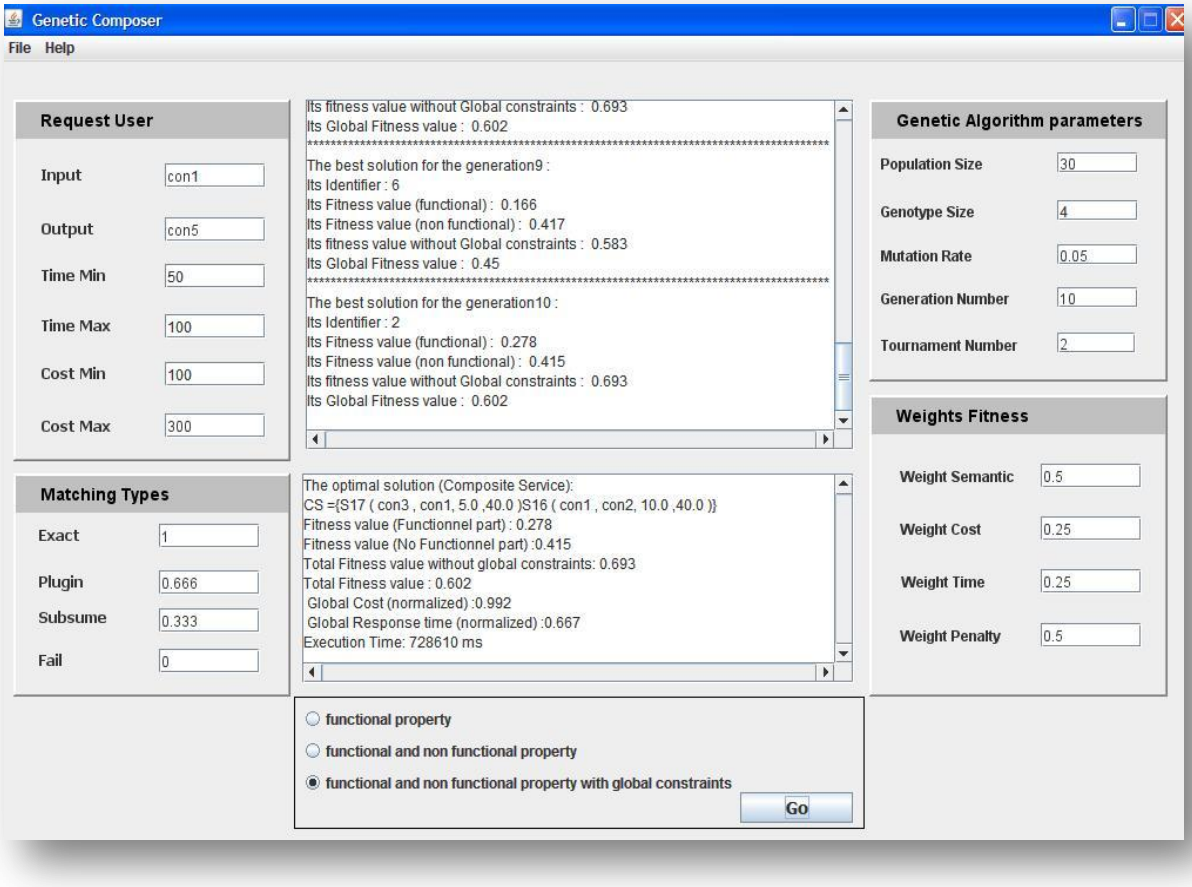

Figure 6.4 : Affichage de résultat d'une composition

6.5 Expérimentations

Dans cette section nous décrivons les expériences menées pour analyser les performances de notre approche.

Expérimentation 1 :

Dans cette 1ere expérimentation nous voulons enregistrer les meilleurs scores de la fonction fitness en fonction des itérations .Elle est conduite avec les paramètres suivants :

- La taille de la population : 50
- Le seuil de la mutation : 0.05
- La taille de chromosome (taille de la composition) : 5

- Le nombre maximal de générations : 50
- Les poids sont fixés comme suit : Ws=0.5, W_T=0.25, W_C=0.25, W_P=0.5.

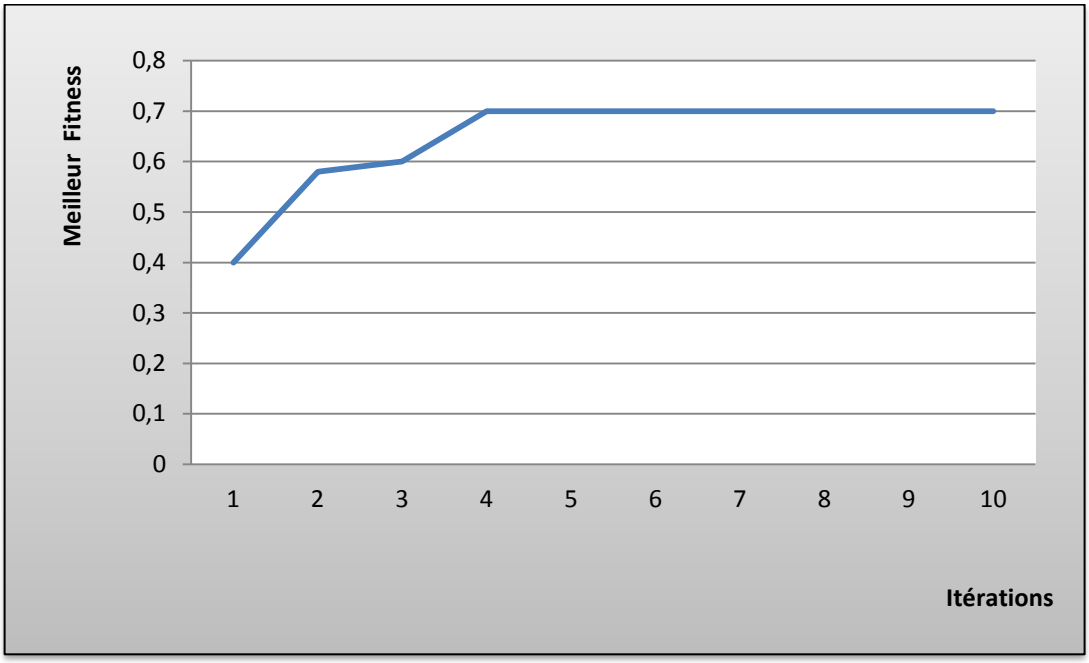

Figure 6.5 :L'évolution de la fonction fitness

La Figure 6.5 illustre les résultats de cette expérimentation. Nous remarquons une grande amélioration des performances entre l'itération 1et l'itération 4. Ceci est du à l'opération mutate2. Après l'itération 5, nous constatons une stagnation des résultats, ceci est expliqué par le fait que l'amélioration de l'aspect fonctionnel (sémantique) des composants du chromosome n'est plus possible, de ce fait les meilleurs chromosomes des anciennes itérations persistent.

Expérimentation 2:

Dans cette deuxième expérimentation nous voulons étudier l'impact de la taille de chromosome sur le temps d'exécution de l'AG. Les autres paramètres de l'AG sont les mêmes que l'expérience 1.

La Figure 6.6 montre les résultats de cette expérimentation. On remarque que si on augmente la taille de la composition alors le temps d'exécution sera trop élevé, cela peut être expliqué par la faible scalabilité de l'AG.

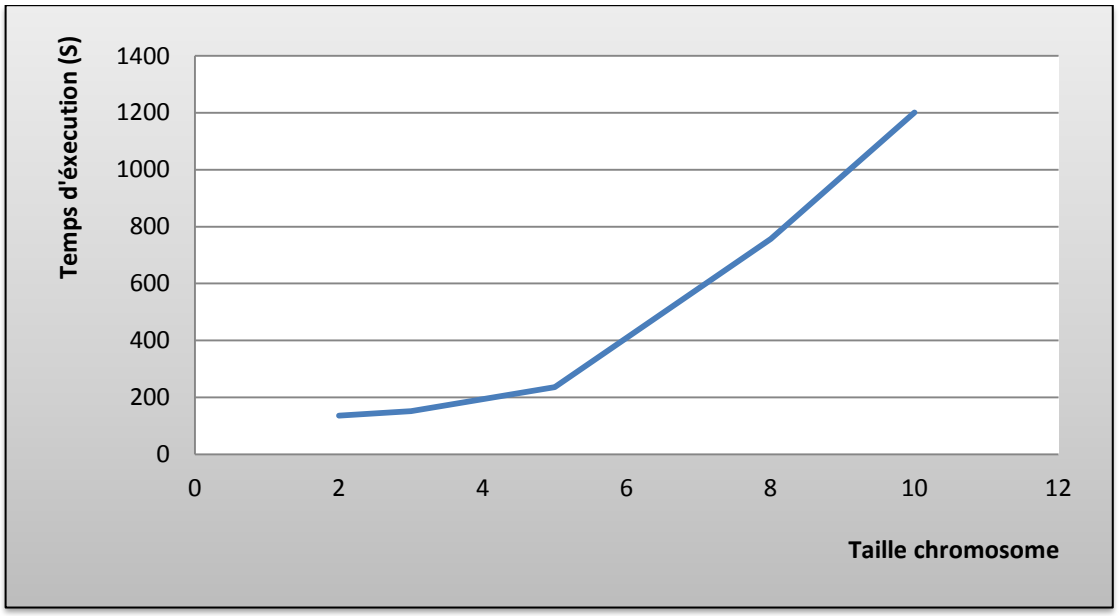

Figure 6.6 :L'impact de la taille de chromosome sur le temps d'éxécution

Expérimentation 3:

Dans cette expérimentation nous avons évaluer notre fonction objective en tenant en compte au début que les propriétés fonctionnelles des services web, ensuite en rajoutant les propriétés non fonctionnelles, et enfin en intégrant les contraintes globales non fonctionnelles.

La Figure 6.7montre la complexité du problème en fonction du nombre de sous fonctions objectives retenues. Si on a juste l'aspect fonctionnel des services web alors les scores peuvent atteindre 0.9. Si on ajoute l'aspect non fonctionnel alors les performances diminuent et ne peuvent dépasser 0.75. Enfin, en ajoutant les contraintes globales alors le problème va devenir NP hard et les scores ne peuvent dépasser 0.7.

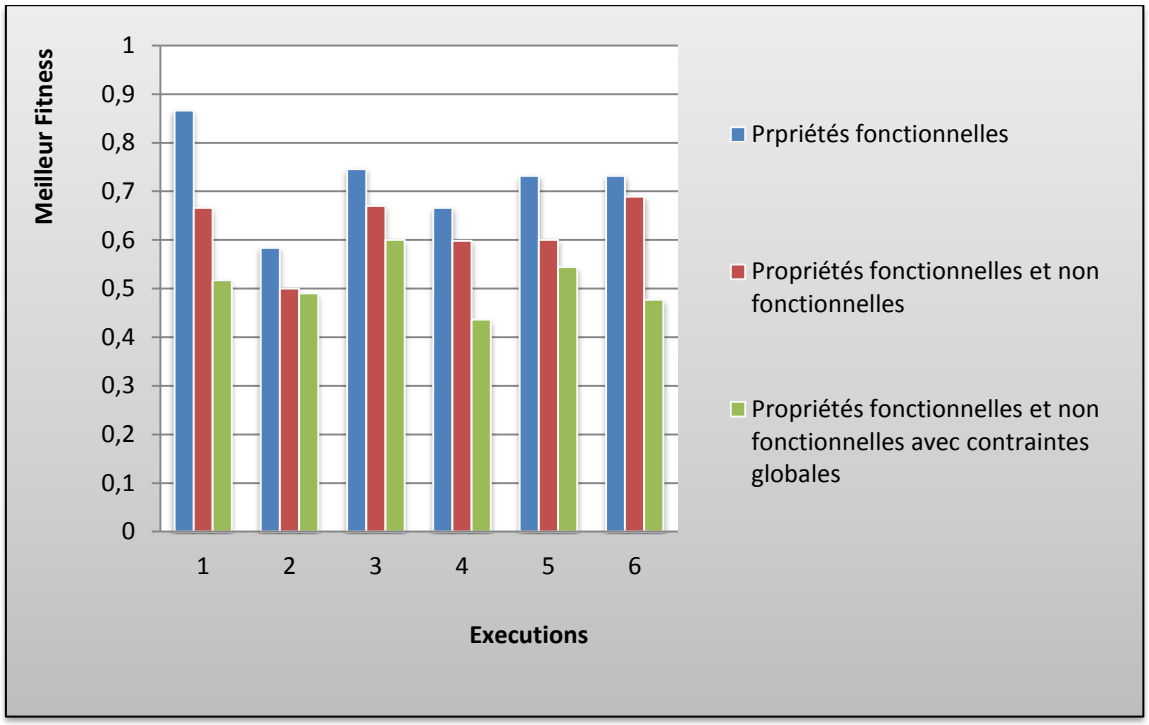

Figure 6.7: Evaluation de la fonction fitness avec les differentes propriétés des web services

6.6 Conclusion

Au cours de cette dernière étape de notre travail, nous avons réalisé un prototype de système de composition de services Web et nous avons présenté les outils utilisés pour son implémentation. L'AG proposé peut atteindre des solutions proches de l'optimum (une fitness au voisinage de 0.7 pour 3 sous fonctions objectives).Il possède aussi un temps d'exécution acceptable pour les compositions ayant une petite taille (≤ 6) mais pour les compositions de grande taille,l'AG présente un temps de réponse très élevé.

Conclusion Générale et Perspectives

Avec le développement rapide des technologies de l'information, la recherche de l'interopérabilité est de nos jours une problématique centrale des systèmes distribués. Ce domaine de recherche est favorisé par l'adoption de l'architecture orientée service comme modèle de développement, et particulièrement, par les services Web qui combinent les avantages de ce modèle aux langages et technologies développés pour Internet. Les services Web ont permis une avancée significative dans l'automatisation des interactions entre systèmes distribués. La composition de services Web, notamment, est considérée comme un point fort, qui permet de répondre à des requêtes complexes.

La majorité des travaux traitant la composition des services web sont orienté vers les approches sémantiques, certains d'entre eux utilisent les workflows, d'autres adoptent les techniques de planification et le reste se base sur les méta-heuristiques.

Dans ce travail de magister, nous avons proposé une approche de composition basée sur un algorithme génétique (les méthodes basés sur les méta-heuristiques)en utilisant les descriptions sémantiques (concepts d'E/S) des services web ainsi que leurs caractéristiques non fonctionnelles exprimées par les paramètres de QoS répondant à une requête de l'utilisateur. Ce dernier peut aussi exprimer ses exigences via des contraintes non fonctionnelles. En effet, nous avons proposé un mécanisme de matching sémantique basé sur les I/O. Pour explorer l'espace de recherche, l'AG proposé adopte deux fonctions de mutation, la première« mutate 1 » permet de retirer aléatoirement un service web, la deuxième fonction « mutate 2 » permet de remplacer un service web par un autre qui possède une compatibilité sémantique non nulle avec les voisinages.

Les résultats obtenus sont très encourageants et montrent l'efficacité(en terme d'optimalité) de notre approche dans la mesure où elle prend en considération les deux propriétés fonctionnelles et non fonctionnelles des services web pour faire la composition dynamique, cependant elle n'est pas scalable pour les compositions ayant une grande taille.

La réalisation de notre approche de composition nous a permis de dégager plusieurs perspectives de travail :

- Considérer que les services web ainsi que la requête sont caractérisés par un ensemble de concepts d'E/S.
- Enrichir les descriptions sémantiques par d'autres aspects fonctionnels telles que les pré-conditions et post-conditions et rendre ainsi plus intelligent le schéma de matching.
- Prendre en compte la pluralité des ontologies de domaine durant le processus de matching, ainsi nous pourrons effectuer un matching entre deux concepts appartenant à des ontologies différentes.
- Prendre en compte d'autres modèles de composition (parallèle, choix, etc.).
- Combiner l'algorithme génétique avec d'autres techniques d'optimisation comme les algorithmes de colonies de fourmis, les algorithmes d'optimisation par essaims particulaires dans le but d'accroitre son efficacité (temps et optimalité).
- Tester notre approche sur une large base de services web décrits par OWL-S et la comparer avec d'autres approches de composition.

Références Bibliographiques

[Abhijit et al., 2004] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework. In WWW, pages 553–562, 2004.

[Akkiraju et al., 2005] Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. Web service semantics - wsdls, November 2005.

[Albers, 1997]P. Albers. IxTeT : extension de la représentation pour la prise en des effets dépendant du contexte et des axiomes du domaine. PhD thesis, Université de Paul Sebatier de Toulouse, 1997.

[Alkamari, 2008]Aniss Alkamari, Mémoire présenté comme exigence partielle de la maîtrise en informatique, Composition de services Web par appariement de signatures, Université du Québec à Montréal, Janvier 2008.

[Amit et al., 2005]Sheth Amit, Miller John, Arpinar I. Budak, and Verma Kunal. Meteor-s :Semantic web services and processes, 2005.

[Andrews et al., 2003]Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. Business Process Execution Language for Web Services, Version 1.1. IBM Specification [en ligne], 2003. Disponible sur : <http://www-128.ibm.com/developerworks/library/specification/ws-bpel>.

[Ankolekar et al., 2002] Daml-s Coalition, Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, OraLassila, Drew Mcdermott, David Martin, Sheila A. Mcilraith, Massimo Pao-lucci, Terry Payne, and Katia Sycara. Daml-s : Web service description for the semantic web. pages 348–363, 2002.

[Arenaza, 2006] Nerea Arenaza, Composition semi-automatique des Web services, Projet de Master, Ecole Polytechnique Fédérale de Lausane, Février 2006.

[Arkin et al., 2002]Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacsi, P., Trickovic, I., Zimek, S. Web Service Choreography Interface (WSCI) 1.0. W3C Note [en ligne], 2002. Disponible sur : <http://www.w3.org/TR/wsci>.

[Barros et al., 2005b] Barros A., Dumas, M., Oaks, P. Standards for Web Service Choreography and Orchestration: Status and Perspectives. In: Procs of the 3 rd International Conference the Business Process Management (BPM 2005), 1 st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Sept. 2005, pp.1-15.

[Bekkouche et al., 2012a] Bekkouche, A., Benslimane S.M., Hadjila, F., QoS-Aware Semantic Web service Composition based on Genetic Algorithm. Accepted in the 1st International Conference on Distributed Systems and Decision, Oran, 21-22 November 2012.

[Bekkouche et al, 2012b] Bekkouche, A., Benslimane S.M., Hadjila, F., Automatic Composition of Semantic Web Services based on Genetic Algorithm. Submitted to the International Conference on Advanced Communication and Information Systems, Batna, 12-13 December 2012.

[Ben et Joseph, 2007] Ben Margolisand, et Joseph Sharpe, SOA for the Business Developer : Concepts,BPEL, and SCA, édition MC Press, October 2007.

[Benatallah et al., 2002]Benatallah, B., Dumas, M., Fauvet, M.-C., Rabhi, F.A., Sheng, Q.Z. Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges, 2002, vol.3, n°3, pp.9-16.

[Benatallah et al., 2005] Benatallah B., Dijkman R., Dumas M., Maamar Z. Service composition: Concepts, Techniques, Tools and Trends. In: Stojanovic Z., Dahanayake A., Eds. Service-Oriented Software Engineering: Challenges and Practices. Idea Group Inc (IGI), 2005, pp.48-66.

[Benatallah et al., 2006] Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, and Farouk Toumani.On automating web services discovery. The VLDB Journal,14(1) :84–96, 2006.

[Bordeaux et al., 2004] Lucas Bordeaux, Gwen Sala¨un, Daniela Berardi, and Massimo Mecella.When are two web services compatible ? In Ming-Chien Shan, Umeshwar Dayal,and Meichun Hsu, editors, TES, volume 3324 of Lecture Notes in Computer Science, pages 15–28. Springer, 2004.

[Boukhadra, 2011]AdelBoukhadra,Mémoire de Magister,La composition dynamique des services Web sémantiques a base d'alignement des ontologies owl-s,2011 .

[Bruijn et al., 2005] Jos de Bruijn, Christoph Bussler, John Domingue, Dieter Fensel, Martin Hepp,Uwe Keller, Michael Kifer, Birgitta Kenig-Ries, Jacek Kopecky, Ruben Lara,Holger Lausen, Eyal Oren, Axel Polleres, Dumitru Roman, James Scicluna,and Michael Stollberg. Web service modeling ontology (wsmo) - w3c member submission, June 2005.

[Bruijn et al., 2007] Jos de Bruijn, John Domingue, Dieter Fensel, Holger Lausen, Axel Polleres, Dumitru Roman, et Michael Stollberg, Enabling Semantic Web Services : The Web Service Modeling Ontology,édition Springer, 2007.

[Canfora et al., 2005]Canfora, G., Penta, M. D., Esposito, R., et Villani, M. L. An approach for qos-aware service composition based on genetic algorithms. In Proceedings of GECCO,pages 1069–1075,2005.

[Cardoso et Sheth, 2002]J. Cardoso and A. Sheth. Semantic e-workflow composition. Technical Report 02-004,LSDIS Lab., Computer Science Department, Univesity of Georgia, Athens, USA, 2002.

[Cardoso, 2002] J.Cardoso.Quality of Service and Semantic Composition of Workflows.PhD thesis,Univ of Georgia, 2002.

[Cardoso, 2007] Jorge Cardoso, Semantic Web Services : Theory, Tools, and Applications, University of Madeira, Portugal, Information Science reference, édition Dunod, 2007.

[Casati et al., 2000] Fabio Casati, Ski Ilnicki, LiJie Jin, Vasudev Krishnamoorthy, and Ming-Chien Shan.Adaptive and dynamic service composition in eflow. In Proceedings of 12th International Conference on Advanced Information Systems Engineering,Stockholm, Sweden, March 2000. Springer-Verlag.

[Casati et al., 2001] Fabio Casati and Ming-Chien Shan.Dynamic and adaptive composition of e-services.Information Systems, 26 :143 – 163, 2001.

[Cerami, 2002] Ethan CERAMI, Web Services Essentials, édition O'Reilly, février 2002.

[Chabeb, 2011]Yassin CHABEB, Contributions à la Description et la Découverte de Services Web Sémantiques, Thèse de doctorat. Université d'Evry-Val d'Essonne,novembre 2011.

[Chalbabi, 2006] M.Chalbabi,Découverte de Services Web Sémantiques : une Approche basée sur le Contexte,novembre 2006.

[Charif, 2007] Y. Charif, et N. Sabouret, Coordination in Introspective MultiAgent Systems, Proceeding of the International Conference on Intelligent Agent Technology (IAT'07), pages 412-415, Silicon Valley, California, USA. 2007.

[Châtel, 2010]Pierre Châtel. Une approche qualitative pour la prise de décision sous contraintes non-fonctionnelles dans le cadre d'une composition agile de services, Thèse de Doctorat de l'université Pierre et Marie Curie Paris 6, 2010.

[Chouchani, 2010] Imad Chouchani,Using a Genetic Algorithm for the composition of web services. Memory of the master's degree in computer science .Quebec university at Montreal,2010.

[Claro, 2006] Daniela BARREIRO CLARO, SPOC - Un canevas pour la composition automatique de services web dédiés à la réalisation de devis, Thèse de Doctorat de l'université d'Angers,Octobre 2006.

[Coello et Carlos, 2000] Coello Coello, Carlos A. (2000). "An updated survey of GA-based multiobjective optimization techniques". ACM Computing Surveys, ACM Press, Vol. 32,N o 2, pp.109-143.

[Collet, 2006]Etat de l'art sur la contractualisation et la composition. RNTL FAROS - Livrable F-1.1 - http://www2.lifl.fr/faros/pub/uploads/Main/RNTL-FAROS-F1-1.pdf,2006.

[Daconta et al., 2003]Michael Daconta , Leo Obrst, et Kevin Smith, Developing the Semantic Web : a Guide to the Future of XML, Web Services, and Knowledge Management, edition Wiley, 2003.

[Deb, 2000] Deb K. (2000). Introduction to selection.Dans : "Evolutionary computation 1: advanced algorithms and operators". Édité par : BäckT., Fogel D.B., et Michalewicz Z.; Institute of Physics Publishing, Bristol andPhiladelphia,p 331.

[Dey et al., 1999] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith,and Pete Steggles. Towards a better understanding of context and context-awareness. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, 1999.

[Dustdar et Schreiner, 2005] Dustdar, S., Schreiner, W. 2005. « A Survey on Web Services Composition », InJ. Web and Grid Services, Vol.l, No.l.

[El Falou, 2010] Mohamad El Falou, Thése de Doctorat, Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multi-agents, l'université de Caen-Basse-Normandie,2 juin 2010.

[Falquet et Mottaz, 2001] G. Falquet, et C.L. Mottaz-Jiang, Navigation hypertexte dans une ontologie multi-points de vue, Nîmes TIC, 2001.

[Fan et al., 2005] J.Fan,B.Ren and L.R.Xiong .An Approach to web service discovery based on the semantics.In FSKD(2),pages 1103-1106,2005.

[Farrell et Lausen, 2007] Joel Farrell and Holger Lausen. Semantic annotations for wsdl and xml schema, August 2007.

[Fensel et al., 2000] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erd-mann, and Michel C. A. Klein. OIL in a nutshell. In Knowledge Acquisition,Modeling and Management, pages 1–16, 2000.

[Fensel et Bussler, 2002]Dieter Fensel and Christoph Bussler.The web service modeling framework wsmf. Electronic Commerce Research and Applications, 1(2) :113–137, 2002.

[Foster et al., 2004]Howard Foster, Sebastian Uchitel, Jeo Magee, and Jeo Kramer.Compatibility verification for web service choreography.In ICWS '04 : Proceedings of the IEEE International Conference on Web Services, page 738, Washington, DC,USA, 2004. IEEE Computer Society.

[Garcia et Henriet, 2004] J.Garcia, E.Henriet, « Les services Web »,2004.

[Gardien , 2002] Georges GARDIEN, XML des bases de données aux Services Web, édition Dunod, 2002.

[Gesnu, 2003] Xavier Gesnu, Développer des Services Web XML et des composants serveurs, édition Du-nod, 2003.

[Ghallab et al., 2004] M. Ghallab, D. Nau, and P. Traverso. Automated Planning : Theory and Practice.Elsevier (Morgan Kaufmann Publishers), 2004.

[Gharzouli, 2011] Mohamed GHARZOULI, Composition des Web Services Sémantiques dans les systèmes Peer-to-Peer, Thèse de Doctorat de l'université Mentouri Constantine,Septembre 2011.

[Goldberg, 1989] Goldberg D.E., (1989). Genetic Algorithms in Search,Optimization and Machine Learning.Addison Wesley Longman, 412 p.

[Goldberg, 1994] Goldberg D.E., (1994). Algorithmes génétiques : exploration, optimisation et apprentissage automatique. Traduction de l'anglais (américain) par Vincent Corruble. Éditions Addison-Wesley France, 417 p.

[Gudgin et al., 2002a] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen.SOAP Version 1.2 Part 1Messaging Framework,19 Décembre 2002. http://www.w3.org/TR/2002/CR-soap12-part1-20021219.

[Gudgin et al., 2002b] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen.SOAP Version 1.2 Part 2 – Adjuncts, 19 Décembre 2002. http://www.w3.org/TR/2002/CR-soap12-part2-20021219.

[Gustavo et al., 2004] A. Gustavo, F. Casati, H. Kuno, and V. Machiraju. Web Services.concepts,Architectures and Applications. Springer-Verlag Berlin Heidelberg, 2004.

[Hack et al., 2007] S. Hack, et M. Lindemann, Enterprise SOA, Einführen, Bonn, Allemagne, Galileo Press,2007.

[Holland, 1975] J.H.Holland, « Adaptation in Natural and Artificial System »,Cambridge, MA: the M.I.T. Press, 1975.

[Huan et al., 2010] L.Huan,Farong Zhong;Bang Ouyang;Jiajie Wu;"An Approach for QoS-Aware Web Service Composition Based on Improved Genetic Algorithm" 2010 International Conference on Web Information Systems and Mining (WISM),vol.1,no.,pp.123-128,23-24 Oct.2010.

[Jacobson, 1991] Jacobson, I. (1991). Object-oriented software engineering.ACM.

[Jaeger et al., 2005]M.C.Jaeger,G.Rojec-Goldmann,G.Muhl,C.Liebetruth,and K.Geihs.Ranked Matching for Service Descriptions using OWL-S.pages 91-102,2005.In Paul Muller, Reinhard Gotzhein,and Jens B,editors,Kommunikation in verteilten Systemen (KiVS 2005),Kaiserslautern,Gemany,February 2005.Springer.

[Jennings, 2007]Frank Jennings, Ramesh Loganathan, et Poornachandra Sarang, Approach to Integration ; XML, Web services, ESB, and BPEL in real-world SOA projects, édition Packt Publishing Ltd, November 2007.

[Josuttis, 2007] Nicolai Josuttis, SOA in Practice, édition O'Reilly, Septembre 2007.

[Kadima et Monfort, 2003] Hubert Kadima, et Valérie Monfort, Les Web services techniques, démarches et outils,édition Dunod, 2003.

[Kavantzas et al., 2005] Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., Barreto, C. Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation [en ligne], 2005. Disponible sur : <http://www.w3.org/TR/ws-cdl-10>.

[Keller et al., 2005]U.Keller,R.Lara,H.Lausen,A.Pollers,and D.Fensel.Automatic location of services.In Proc.of the 2nd European Semantic Web conference (ESWC),pages 1-16,Heraklion,Crete,2005.LNCS 3532,Springer.

[Klusch et Kapahnke, 2008]Matthias Klusch and Patrick Kapahnke.Semantic web service selection with SAWSDL-MX. In Rub´en Lara Hernandez, Tommaso Di Noia, and Ioan Toma,editors, SMRR, volume 416 of CEUR Workshop Proceedings. CEUR-WS.org,2008.

[Kreger, 2001] Heather Kreger, Web Service Conceptual Architecture, édition IBM Software Group, Mai 2001.

[Küster et al., 2008] U.Küster and B.König-Ries.Evaluating semantic web service matchmaking effectiveness based on graded relevance. 7th International Semantic Web Conference (ISWC08),Karlsruhe,Germany,October 2008.

[Kuter et al., 2005] Ugur Kuter, Evren Sirin, Bijan Parsia, Dana Nau, and James Hendler.Informa-tion gathering during planning for web service composition. Web Semantics :Science, Services and Agents on the World Wide Web, 3 :183 – 205, 2005.

[Lämmermann, 2002] Sven Lämmermann. Runtime service composition via logic-based program synthesis. PhD thesis, Royal Institute of Technology, June 2002. Department of Microelectronics and information Technology, Royal Institute of Technology,Stockholm, Sweden.

[Lécué et al., 2006] Lécué F. and Léger. A., "A formal model for semantic web service composition". In ISWC ,the 5th International Semantic Web Conference, November 2006.

[Lécué, 2009] F.Lecue,Optimizing QoS-Aware Semantic Web Service Composition.2009.

[Leymann, 2001] Leymann F. Web Service Flow Language 1.0.IBM Report [en ligne],2001/ <http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>.

[Lopes, 2005] Denivaldo cicero pavão Lopes, Etude et applications de l'approche MDA pour des plates-formes de Services Web, Thèse de Doctorat, UFR Sciences et Techniques, Université de Nantes, le 11 Juillet 2005.

[Martin et al., 2004] David Martin, Massimo Paolucci, Sheila Mcilraith, Mark Burstein, Drew Mc-dermott, Deborah Mcguinness, Bijan Parsia, Terry Payne, Marta Sabou, Mo-nika Solanki, Naveen Srinivasan, and Katia Sycara. Bringing semantics to web services : The owl-s approach. pages 26–42. Springer, 2004.

[Matskin et al., 2005]M. Matskin, P. Küngas, J. Rao, J. Sampson, and S. Abbas Petersen. Enabling Web Services Composition with Softward Agents. In IMSA, pages 93–98,2005.

[McDermott, 2002]Drew V. McDermott. Estimated-regression planning for interactions with web services.In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors,AIPS, pages 204–211. AAAI, 2002.

[McIlraith et al., 2002]Sheila A. McIlraith and Tran Cao Son.Adapting golog for composition of semantic web services. In Dieter Fensel, Fausto Giunchiglia, Deborah McGuin-ness, and Mary-Anne Williams, editors, Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), pages 482–496, San Francisco, CA, April 22–25 2002. Morgan Kaufmann Publishers.

[Medjahed et al., 2003] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. The VLDB Journal, 12 :333–351, 2003.

[Melliti, 2004] Tarek MELLITI, "Interopérabilité des services Web complexes, Application aux systèmes multi-agents", thèse de Doctorat de l'Université Paris IX Dauphine, décembre 2004.

[Miller et al., 2004]J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashan-mugam. Wsdl-s : Adding semantics to wsdl, 2004.

[Mitra, 2002] Nilo Mitra. SOAP Version 1.2 Part 0 – Primer, 19 Décembre 2002.http://www.w3.org/TR/2002/CR-soap12-part0-20021219.

[Narayanan et McIlraith, 2002] Srini Narayanan and Sheila A. McIlraith.Simulation, verification and automated composition of web services. In WWW '02 : Proceedings

of the 11th international conference on World Wide Web, pages 77–88, New York, NY,USA, 2002. ACM Press.

[Newcomere, 2004] Eric Newcomere, Understanding Web Services Xml WSDL SOAP and UDDI, edition O'Reilly, 2004.

[Osman et al., 2005] T. Osman, D. Thakker, and D. Al-Dabass. Bridging the Gap between Work-flow and Semantic-based Web services Composition.In Proc. of the Web Service Composition Workshop WSCOMPS05, 2005.

[Ouyang et al., 2005] C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hofstede,and H. M. W. Verbeek. Formal semantics and analysis of control flow in WS-BPEL (revised version). BPM Center Report BPM-05-15, BPMcenter.org,2005.

[OWL-S, 2005] OWL-S Ontology Web Language - Semantics, 2005.http://www.daml.org/services/owl-s/1.0/owl-s.html.

[Paolucci et al., 2002]Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara.Semantic matching of web services capabilities.2002.

[Paolucci et al., 2004] N.Srinivasan,M.Paolucci,and K.Svcara.Adding OWL-S to UDDI,implementation and throughput.In First International Workshop on semantic web services and webprocess composition (SWSWPC 2004),pages 6-9,2004.

[Papazoglou et al., 2003] M.P. Papazoglou and D Georgakopoulos.Service Oriented-Computing. Communications of the ACM, 46(10) :25–28, 2003.

[Paul et Sandeep, 2004] Paul Palathingal and Sandeep Chandra. Agent approach for service discovery and utilization. In HICSS, 2004.

[Peltz, 2003] Peltz, C. Web Services Orchestration and Choreography. IEEE Computer, 2003, vol.36, n°10, pp.46-52.

[Pistori et al., 2005]M. Pistori, P. Traverso, and .Bertoli. Automated composition of web services by planning in asynchronous domains. In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), pages 2–11, Monterey, 2005.

[Ponge, 2004] Julien Ponge ; Comptabilité et substitution dynamique des web Services ; Mémoire de fin d'étude, Université Blaise Pascal Clermont II ; Juillet 2004.

[Rajasekaran et al., 2004] Preeda Rajasekaran, John A. Miller, Kunal Verma, and Amit P. Sheth.En-hancing web services description and discovery to facilitate composition. In SWSWPC, pages 55–68, 2004.

[Rampacek, 2006] Sylvain Rampacek ; Sémantique, interactions et langages de description des services web complexes ; Thèse de Doctorat ; 10 Novembre 2006.

[Rao et Su, 2004] J. Rao and X. Su. A survey of automated web service composition methods. In Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, volume 3387 of Lecture Notes in Computer Science, pages 43–54, San Diego, USA,2004. Springer.

[Regnier, 2004] P. Regnier. Algorithmique de la Planification en IA. Cepadues-Editions, 2004.

[Ricardo, 2004]Ricardo DE LA ROSA-ROSERO ; Découverte et Sélection de Service Web pour une application Mélusine ; l'Institut d'Informatique et de Mathématiques Appliquées de Grenoble, le 15 septembre 2004.

[Roman et al., 2005] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rub´en Lara, Mi-chael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web service modeling ontology. Appl. Ontol., 1(1) :77–106, 2005.

[Schulte et Natis, 1996]RW.Schulte and YV.Natis. Service oriented architectures, part 1 &2.http ://www.gartner.com, 1996.

[Schumacher et al., 2008] M.Schumacher,H.Helin,and H.Schuldt.Semantic Web Service Coordination.Chapter 4,CASCOM: Intelligent service Coordination in the Semantic Web,Birkhauser Basel,2008.

[Schuster et al., 2000] Hans Schuster, Dimitrios Georgakopoulos, Andrzej Cichocki, and Donald Baker.Modeling and composing service-based and reference process-based multi-enterprise processes. In Proceedings of 12th International Conference on Advanced Information System Engineering, Stockholm, Sweden, June 2000.Springer-Verlag.

[Scicluna et al., 2006] Dumitru Roman, James Scicluna, Cristina Feier, Michael Stollberg, and Die-ter Fensel. Ontology-based choreography and orchestration of wsmo services,March 2006.

[Scott, 2002] Scott Short, Construire des Services Web XML, édition Dunod, 2002.

[Shankar et al., 2002] Shankar R. Ponnekanti and Armando Fox.SWORD : A developer toolkit for web service composition. January 01 2002.

[Singh et al., 2005] M.P. Singh and M.N. Huhns.Service-Oriented Computing,

semantics, processes,agents. John Wiley and Sons, 2005.

[Sirin et al., 2002]Evren Sirin, James Hendler, and Bijan Parsia.Semi-automatic composition of web services using semantic descriptions. In Proceedings of Web Services : Modeling, Architecture and Infrastructure Workshop in conjonction with ICEI2003, 2002.

[Sirin et Parsia, 2004a] E. Sirin and B. Parsia. Planning for semantic web services. In Proceedings of Semantic Web Services : Preparing to meet the World of Business Applications Workshop of the third International Semantic Web Conference, november 2004.

[Sivashanmugam et al., 2003] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, and John Miller. Ad-ding semantics to web services standards. In Liang-Jie Zhang, editor, ICWS,pages 395–401. CSREA Press, 2003.

[Solanki et al., 2004] Monika Solanki, Antonio Cau, and Hussein Zedan. Augmenting semantic web service descriptions with compositional specification. In Proceedings of the 13th international conference on World Wide Web (WWW'04), pages 544–552, New York, NY, USA, 2004.ACM.

[Stollberg et al., 2007]M.Stollberg,U.Keller,H.Lausen,and S.Heymans.Two-Phase Web Service Discovery Based on Rich Functional Descriptions.In ESWC'07:Proc.of the 4th European conference on the Semantic Web,pages 99-113,Berlin,Heidelberg,2007.Springer-Verlag.

[Sun et al., 2003] Sun, H., Wang, X., Zhou, B. and Zou, P. 2003. «Research and Implementation of Dynamic Web Services Composition », APPT 2003, LNCS 2834, Springer-Verlag Berlin Hbidelberg, pp.457--466.

[Talantikite et al., 2009] H. Nacer Talantikite, D. Aissani, and N. Boudjlida, "Semantic annotations for web services discovery and composition," Computer Standards Interfaces, vol. 31, no. 6, pp. 1108-1117, 2009.

[Thatte, 2001] Thatte, S., XLANG: Web Services for Business Process Design. Microsoft Specification, 2001.

[Waldinger et al., 2001] Richard J. Waldinger.Web agents cooperating deductively. In FAABS '00 : Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers, pages 250–262, London, UK, 2001.Springer-Verlag.

[Weise et al., 2007] Thomas Weise, Steffen Bleul, and Kurt Geihs.Web Service Composition Systems for the web Service Challenge – A Detailed Review. University of Kassel, Wilhelmsh¨oher Allee 73, 34121 Kassel, Germany, November 19, 2007.

[Weise et al., 2009] Thomas Weise, Steffen Bleul, and Kurt Geihs."The web service challenge-A Review on Semantic Web Service Composition" in service oriented computing.SOC 2009 March 5, 2009.

[Wohed et al., 2003] Wohed, P., van der Aalst, W., Dumas, M., ter Hofstede, A. Analysis of Web Services Composition Languages: The case of BPEL4WS. In: Procs of the 22 nd International Conference on Conceptual Modeling, Oct. 2003, Chicago, IL, USA. LNCS Springer 2003, pp.200-215.

[Wu et al., 2003] Dan Wu, Evren Sirin, James Hendler, Dana Nau, and Bijan Parsia. Automatic web services composition using SHOP2. In Workshop on Planning for Web Services, Trento, Italy, June 2003.

[Wynen, 2003] Wynen F. Conception et développement d'une plate-forme de coopération inter-organisationnelle : le point de synchronisation. Mémoire d'ingénieur. CNAM– Conservatoire Nationale des Arts et Métiers, Nancy, 2003, 102p.

[Yu et al., 2010] Q Yu, A Bouguettaya. Foundations for Efficient Web service selection Springer Science Business Media,2010.

[Zeng et al., 2003] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam,and Q. Z. Sheng. Quality driven web services composition. In Proceedings of the International World Wide Web Conference, pages 411–421, 2003.