الجمهوريّة الجزائريّة الدّيمقراطيّة الشّعبيّة
RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
وزارة التّعليم العالي و البحث العلمي
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعة فرحات عبّاس – سطيف –
UNIVERSITÉ FERHAT ABBES -SÉTIF-

Vice rectorat chargé de la post graduation نيابة الجامعة المكلّفة بدراسات ما بعد التّدرّج
مدرسة الدكتوراه في الإعلام الآلي تخصّص هندسة الأنظمة المعلوماتيّة
École doctorale d'informatique
Option
Ingénierie des Systèmes Informatiques

Mémoire

Présenté pour l'obtention du

Diplôme de Magister

par **Fateh Seghir**

Thème:

Composition automatique de services web sémantiques par planification et techniques d'intelligence artificielle dans des systèmes ouverts et dynamiques

Membres du Jury :

Président : Dr TOUAHRIA.M	Maître de conférences (U.F.A SETIF)
Rapporteur : Dr KHABABA.A	Maître de conférences (U.F.A SETIF)
Examinateur : Dr MOUSSAOUI.A	Maître de conférences (U.F.A SETIF)

Juillet 2009

REMERCIEMENTS

Comme est la tradition, au début toujours il y a un remerciement, alors comment ne pas remercier le créateur, qui m'a donner la santé, la volonté, le courage de faire ce modeste travail.

Je tien tout d'abord, à remercier mon encadreur Monsieur Khababa, pour l'aide qui m'a procuré, pour sa disponibilité à mon égard, ainsi que d'avoir accepter de diriger ce travail et de m'avoir bien orientés.

Je tiens également à remercier messieurs les membres du jury, qui ont accepté d'examiner ce travail.

Je remercie tous mes enseignants du département d'informatique.

Je ne peut manquer d'exprimer ma gratitude et mes remerciements à mes collèges de la promotion PG 2006

A tous ceux qui m'ont aidé de prêt ou de loin

Merci

SOMMAIRE

CHAPITRE IV **MISE EN PLACE D'UN ALGORITHME DE COMPOSITION DE SERVICES WEB**

Liste des tableaux et des figures

Introduction

La notion de service web désigne essentiellement une application (un programme) mise à disposition sur Internet ou sur un réseau interne d'une entreprise par un fournisseur de service, et accessible par les clients à travers des protocoles standards. Des exemples de services actuellement disponibles concernent les prévisions météorologiques, la réservation de voyage en ligne, les services bancaires, moteur de recherche (exemple : *google*) et de la vente en ligne (*amazon* par exemple).

L'évolution de l'Internet comme un support de communication entre les applications et les organisations a révolutionné les méthodes de coopération classiques. Les technologies réseaux actuels, notamment les services web, permettent le développement de nouveaux paradigmes de coopération. Les services web représentent la dernière évolution des technologies web ; même si les services web reprennent la plupart des idées et des principes du web, ils différent du World Wide Web dans leur portée **[44].**

HTTP et HTML ont été définis pour permettre une exploration, essentiellement en mode lecture. A l'opposé, les services web permettent des interactions fortement dynamique entre programmes. Les infrastructures actuelles s'avèrent suffisantes pour une utilisation des services web pour des opérations simples (telles que : la recherche simple, mise à jour distante, etc.) ; mais des carences furent mise en évidence pour des opérations plus complexes (particulièrement dans les intégrations B2B) aussi bien que dans la sécurité, les transactions, **composition**, fiabilité et la qualité de services. La génération des services web à partir de la composition de services déjà existent offre un gain considérable dans de nombreux cas. On pourrait citer, comme exemple, l'intégration d'applications d'entreprise hétérogènes où la mise en place d'une collaboration B2B.

L'objectif ultime de l'approche service web est de transformer le web en un dispositif distribué de calcul où les programmes (services) peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se **composer** en services plus complexes. En d'autres termes, l'idée poursuivie avec les web services, est de mieux exploiter les technologies de l'Internet en substituant, autant que possible, les humains qui réalisent actuellement un certain nombre de services, par des machines. L'un des points forts de cette nouvelle technologie est son indépendance totale vise à vis des plateformes où sont déployés ces services.

Notre travail dans ce mémoire concerne la thématique des systèmes d'information orientés web, et plus particulièrement la problématique de la *composition de services web dans un*

contexte de systèmes ouverts et dynamiques. L'objectif est de trouver une solution intelligente pour la composition de services web. Pour cela, nous allons fait une étude assez détaillée sur les approches basées sur des techniques d'intelligence artificielle et planification qui ont montrées leurs efficacités et robustesses pour résoudre ce type de problème. A travers cette étude nous avons sorti avec la proposition d'une approche de composition automatique basée sur deux algorithmes, le premier (GRAPHPLAN) est un algorithme de planification et le deuxième (Bellman Ford) permettant de trouver le plus court chemin dans un graphe orienté.

Ce mémoire s'articule autour de quatre chapitres :

- ***Chapitre I : « Introduction générale au services web et services web sémantiques »*** Le premier chapitre présente les principes essentiels des services web, et services web sémantiques. Nous avons commencé de donner en premier lieu l'architecture et cycle de vie d'un service web, ainsi que les standards utilisés pour assurer une interopérabilité entre services web via des plateformes et systèmes hétérogènes. Ensuite nous avons introduire le concept de service web sémantique et la présentation des différents langages utilisés pour l'annotation sémantique des services web.
- ***Chapitre II : « Planification classique et hiérarchique en intelligence artificielle »*** En particulier ce chapitre, fournit une description générale des techniques de planification du domaine d'intelligence artificielle, dont les principales techniques sont la planification classique et la planification hiérarchique.
- ***Chapitre III : « Etat de l'art sur la composition automatique de services web »*** Le troisième chapitre fournit une étude assez importante sur les principales approches développées dans le domaine de composition automatique de service web. Cette étude nous a permis de déterminer l'influence des techniques de planification du domaine d'intelligence artificielle vis-à-vis du problème de composition automatique de services web.
- ***Chapitre IV : «Mise en place d'un algorithme de composition automatique de services web »*** Dans ce dernier chapitre nous nous somme consacrés à la définition et modélisation du problème de composition automatique de services web, Nous avons commencé d'abord avec la présentation du problème de découverte et de composition de services web, ensuite proposer une approche basée sur une utilisation combinée de deux algorithmes, le premier qui est GRAPH EXPANSION de l'algorithme de planification

GRAPHPLAN qui se charge de construire un graphe de planification qui contient l'ensemble de services web impliqué dans le processus de composition, le deuxième algorithme est l'un des algorithmes fédérateurs dans la théorie du graphe permettant de trouver une séquence ordonnées de services web qui résoudre le problème de composition à travers la découverte de plus court chemin dans un graphe de services web. Cet algorithme à deux étapes, qui est simple dans le sens de sa structure et facilité d'implémentation.

Pour finir, une conclusion et des perspectives sont données afin de présenter les différents axes d'études que nous ouvrons afin d'élargir ce travail.

Chapitre I

Introduction générale au services web et services web sémantiques

Table des matières :

Introduction

Le web est un excellent outil permettant la connectivité de sources d'informations qui étaient inaccessibles. Au commencement du web le partage d'information est visé vers des chercheurs, mais avec l'évolution du web ce partage d'information ne reste pas seulement entre chercheurs mais avec les différentes organisations, entreprises et individus afin de mettre leurs données et applications accessibles via le web.

Sachant que les applications et les données des internautes sont développées avec des langages de programmations différents et sous plateformes hétérogènes. Pour assurer un échange d'information entre ces applications plusieurs paradigmes d'interaction entre applications sont développés, notamment les architectures orientées services (Service-Oriented Architecture, ou SOA) à été mise en avant afin de permettre des interactions entre applications distantes. Ces architectures sont construites autour de la notion de *service* ou *service web*.

Les services web ont pour objectif de rendre la communication, l'accès et l'échange de données entre applications transparent pour tout type de plateformes, et cela à travers l'utilisation de protocoles et standards d'interopérabilités.

Nos présentons dans un premier lieu, les architectures orientées computing (Service Oriented Computing SOC) ainsi que les concepts étudiés dans ce manuscrit *service web* et *composition de services*, ensuite nous introduisons le concept du web sémantique et son combinaison avec le service web, qui donne le *service web sémantique*. On terminera ce chapitre avec une présentation assez détaillée de principales approches et langages utilisés pour la description sémantique de service web.

1.1 Service Oriented Computing (SOC).

SOC est le paradigme de computing qui utilise des services comme étant les éléments fondamentaux pour la réalisation des applications, cette collection de services communique entre eux via des protocoles et technologies web standardisés (XML[1], HTTP[2], FTP[3], SMTP[4]….). L'utilisation de ces technologies et protocoles standards à pour but d'atteindre un objectif majeur qui est la réalisation d'applications distribuées en réutilisant des services ou blocks existants dans des architectures et plateformes qui sont souvent hétérogènes **[01] .** Un des plus importants secteurs qui bénéficie des applications SOC est E-beseniss, Ou la nécessité de communication aisé et facile entre entreprises et utilisateurs via des services web est primordiale.

Le paradigme de Service Oriented Computing **[02]** utilise des services pour supporter le développement rapide, avec un moindre coût et forte interopérabilité à travers des applications distribuées dans des grands réseaux.

Ces services sont des entités logicielles autonomes, indépendantes aux architectures et langages de programmations utilisés qui peuvent être représenté *(described)* , publié *(published)* , découvert *(discovered)* , invoqué (invoked) et facilement couplé pour répondre aux requêtes simples ou complexes des utilisateurs.

Donc on peut dire que le paradigme **SOC** envisage de transformer et présenter les applications existantes sous forme de services web.

Comme Les services web sont des applications (entités logicielles) autonomes donc ils peuvent être utilisés par des autres applications ou services Web, et pas seulement avec des utilisateurs. Il apparaît clairement qu'il y a une distinction entre un service web et application web (web portal), dont le dernier concept (web portal) sont des applications orientées humain ou utilisateur, s'est à dire ces services sont invoqués directement par les utilisateurs comme par exemple : les page web, par contre un service web peut être invoqué soit directement par un utilisateur, une autre application logicielle ou un autre service web. C'est que ramène la procédure d'invocation des services web automatique ou bien machine

[1] Extensible Markup Language. http://www.w3.org/XML/1999/XML-in-10-points.fr.html.
[2] HyperText Transfer Protocol. http://www.w3.org/Protocols/
[3] File Transfer Protocol.
[4] Simple Mail Transfer Protocol.

vers machine, car une application logicielle ou un service peut utiliser un autre service sans intervention de l'être humain.

1.2 Service web et composition de services web.

Dans cette section nous allons vous présenter et donner des définitions pour les concepts étudiés dans ce manuscrit à savoir **service web** et le problème de la **composition de services**

1.2.1 Service web.

Les services web sont développés en suivant un modèle d'architecture orienté service (SOA[1]), ce modèle utilise des protocoles INTERNET standard pour gérer une communication aisée et fiable entre les différents applications éparpillées sur le web.

Afin de comprendre c'est quoi un service web nous avons prendre les deux définitions suivantes du W3C[2] :

01- « A web service is *a software application identified by a URL, whose interfaces and bindings are capable of being defined, described and discovered as XML artefacts. A web service supports direct interactions with other software agents using XML- based messages exchanged via Internet-based protocols.* » **[03]**

02- « *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL[3]). Other systems interact with the Web service in a manner prescribed by its description using SOAP[4] messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.* » **[04]**

A partir de ces définitions on peut remarquer q'un service web vérifie les propriétés suivants :

- Il est identifié par une URL (Uniform Resource Locator).
- Ces interfaces et ces liens peuvent être décrits en XML.
- Sa définition peut être découverte par d'autres services.

---

1 Service Oriented Architecture. http://www.w3.org/TR/ws-arch/
2 World Wide Web Consortium. Organisme de normalisation des standards du web, http://www.w3.org/
3 Web Services Description Language. http://www.w3.org/TR/Wsdl
4 Simple Object Access Protocol. http://www.w3.org/TR/Soap

- Il peut interagir directement avec d'autres services web à travers le langage XML et en utilisant des protocoles Internet.

Ces propriétés montrent q'un service web est identifié par une URL pour le localiser a travers le web ainsi qu'il est toujours accompagné par un fichier de description fournissant aux applications les informations nécessaires à son utilisation. Le service web est implémenté par le fournisseur de service, ce dernier publie le fichier de description de service dans un annuaire pour permettre aux applications clients d'envoyer leurs requêtes à l'annuaire afin de sélectionner et découvrir les services recherchés. Après avoir découvrert le service et à partir de téléchargement de fichier de description, l'application cliente invoque directement le service concerné. Ce mécanisme est illustré par la ***figure.1.1.***

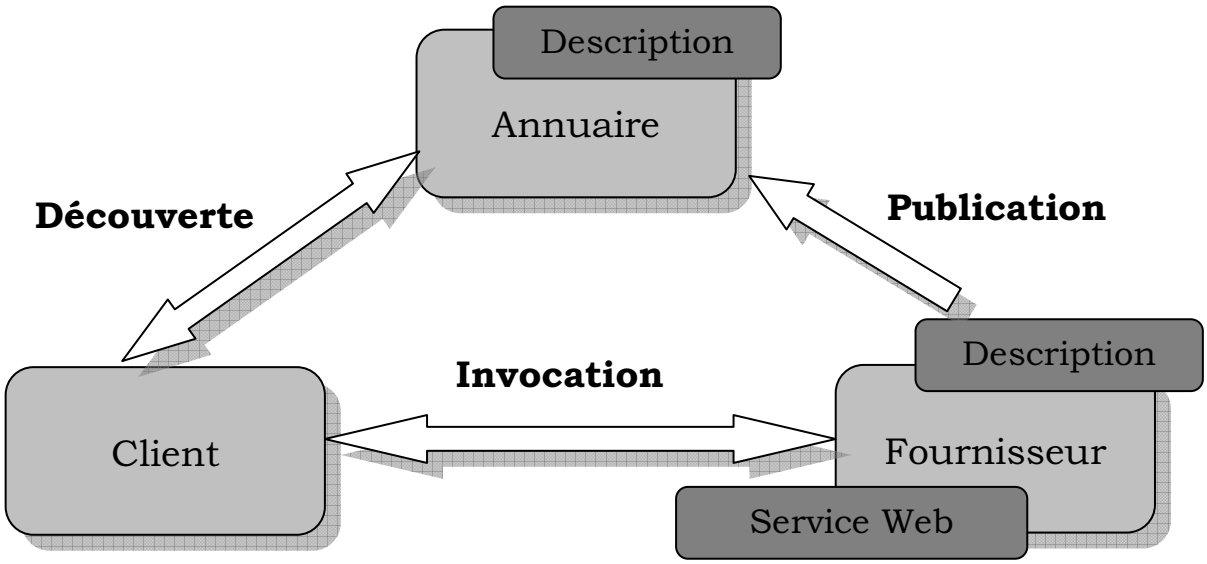


Fig.1.1 découverte, publication et invocation de services web.

1.2.1.1 Infrastructure et Architecture de service WEB.

En effet il en existe plusieurs architectures de services web proposées par différentes organisations. Cependant toutes ces architectures respectent des protocoles et technologies standard. Nous avons pris l'architecture proposée par K.Gottschalk et All **[12]** dans laquelle plusieurs couches (layers) sont superposées, chacune d'elles joue un rôle bien défini.

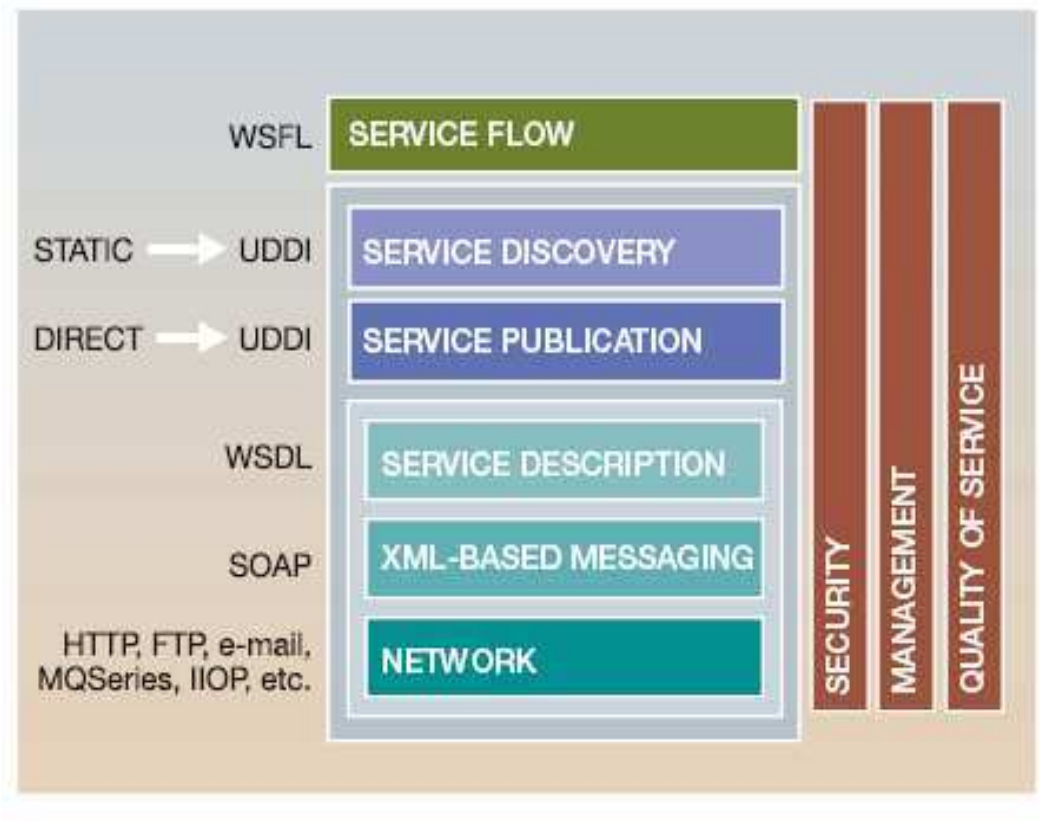


Fig.1.2 Architecture des services Web

Dans ce que suit, nous allons vous donner les fonctionnalités de chaque niveau Comme montre la ***figure.1.2*** ,K.Gottschalk et All **[12].**

- **La couche du transport :** C'est le niveau le plus bas ***Figure.1.2*** permettant un échange d'information entre les différentes services web, en utilisant des protocoles standards comme HTTP, FTP afin d'assurer une communication universelle entres les différents systèmes qui sont souvent hétérogènes. Dans cette couche, on trouve d'autres protocoles qui sont utilisés comme IIOP[1] et MQSeries[2], etc.
- **La couche de messagerie :** C'est une couche basée sur XML qui se trouve juste au dessus de la couche Network, facilitant la communication entre le service web et ses clients. En général, La couche de messagerie utilise le protocole SOAP.
- **La couche de description :** permettant la description du service web au client, qui utilise la langage WSDL comme langage de spécification et description de service web.

[1] **I**nternet **I**nter-**O**RB **P**rotocol.
[2] MQSeries: Service de messagerie développé par IB, Utilise des files d'attente et permet la communication entre les différents applications.

- **La couche de publication :** la publication de service est une action faite par le fournisseur de service pour rendre le ficher (WSDL) ou la description de fichier accessible aux autres services et aux clients. L'envoi du WSDL (ou un pointeur URL du WSDL) dans un annuaire de service est appelé la publications de services web afin que les utilisateurs peuvent découvrir et rechercher le service associé au fichier de description publié (pour plus de détail voir la section suivante 1.2.1.2).
- **La couche de découverte :** Après avoir publié le fichier de description WSDL dans l'annuaire de service. La couche Service Discovery ***figure.1.2*** gère l'accès des clients et les autres services aux fichiers de description qui contiennent la localisation de service web sélectionné (URL de service).
- **La couche de composition :** c'est une couche appuyant sur plusieurs langages et protocoles, permettant la composition des services web dans le cas ou un seul service ne suffit pas pour renvoyer le résultat d'une requête utilisateur mais l'agrégation de plusieurs service va répondre à cette requête.

  Plusieurs langages standardisés sont développés pour répandre au problème de composition de service, comme par exemple WSFL[1] crée par la firme IBM. Mais il existe d'autres langages de composition.

1.2.1.2 Publication, découverte et invocation de services web.

Les architectures orientées service cible a développent des services autonomes et indépendants de toute plateforme utilisée.

Il est clairement apparaître que l'interaction entre les différents services et application orientés services se basent sur les trois composants suivants ***Figure.1.1*** :

1- **le fournisseur de service (provider):** qui ce charge de concevoir et implémenter le service. ensuite ***publier*** *la description de* ce dernier dans un annuaire de service.

2- **L'annuaire de service (Service Registry) :** contient un fichier de description correspond au service implémenté afin que les utilisateurs ou applications web peuvent le découvrir à travers ce fichier de description

3- **Le client (Consumer):** qui est une application logicielle ou un utilisateur de service web. Mais avant que le client ***invoque*** directement le service il doit ***découvrir*** sa

[1] Web Services Flow Language

description dans l'annuaire de service. Cette description contient toutes les informations nécessaires pour l'invocation de service a savoir (URL de service,…..) ;

Pour publier, découvrir et invoquer un service dans un réseau qui est souvent de grande taille avec des systèmes et plateformes hétérogènes. L'utilisation des standards et technologies de communication, publication et invocation est devenue indispensable qui fait appel en générale à ces trois technologies: SOAP, WSDL, UDDI[1]. **[05]**

1.2.1.2.1 Protocole de communication SOAP.

Le protocole SOAP (Simple Object Access Protocol), initialement crée par Microsoft et ensuite développé en collaboration avec IBM, lotus et Userland, basé sur le langage XML car sa syntaxe unique résoudre les conflits syntaxiques lors de l'encodage et transmission de données.

SOAP est un protocole permettant aux services web d'échanger des informations dans des systèmes décentralisées peer to peer et distribuées, indépendamment de touts plateforme ou langage de programmation utilisée **[05][06]**.

SOAP définit deux types de massages, Requête (Request) et Réponse (Response), permettant aux clients ou bien demandeurs de demander un service via une procédure d'accès à distance du service sollicité et aux fournisseurs de répondre à de telles requêtes **[19]**.

Le message SOAP se compose de deux parties : Entête (Header) et XML payLoad. L'entête diffère des couches de transport mais XML payload reste la même.

La partie XML de la requête SOAP se compose de trois parties principales :

Enveloppe (Envelope) : définit les divers namespaces qui sont utilisés par le reste du message SOAP.

L'entête (Header) : est un élément optionnel pour la diffusion des informations auxiliaires d'authentification, transaction et payement. Si l'entête est présente elle doit être le premier enfant de l'enveloppe.

Le corps (Body) : est l'élément principale du message, quand SOAP est utilisé pour exécuter un RPC[2] , le corps contient un élément simple contenant le nom de la méthode et ces arguments ainsi de l'adresse cible du service.

[1] Universal Description Discovery and Integration. http://www.uddi.org/ par OASIS Open
[2] Remote Procedure Call

Pour la *réponse SOAP* est retournée sous forme d'un document XML dans une réponse standard http. Le document XML retourné à la même structure que la *requête SOAP* sauf que le corps contient le résultat encodé de la méthode.

La ***figure.1.3.a*** montre que la requête SOAP est envoyée avec le protocole http au service « stock quote » indique dans le corps (Body) que la requête est transmit à l'adresse URL http://www.stockquoteserver.com/StockQuote. La requête indique également que l'opération GetLastTradePrice utilise le symbole MS (Stock ticker symbol) afin de renvoyer le dernier prix de demi-gros des actions de Microsoft.

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="http://example.com/stockquote.xsd">
           <symbol>MS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig.1.3.a Exemple Requête SOAP

La réponse SOAP contient le prix retourné 143 voir la ***figure.1.3.b***

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      <SOAP-ENV:Body>
         <m:GetLastTradePriceResponse
mlns:m="http://example.com/stockquote.xsd">
              <Price>143</Price>
         </m:GetLastTradePriceResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig.1.3.b Exemple Réponse SOAP

1.2.1.2.2 Annuaire UDDI.

Les fichiers de description WSDL ne jouent aucun rôle si un annuaire de service n'existe pas, donc un annuaire va stocker et localiser les description des services pour permettre aux autres services et applications clientes de trouver et sélectionner le service correspond a leurs besoins.

UDDI (Universal Description Discovery and Integration) offre aux utilisateurs une manière systématique unifiée pour la découverte des services web qui sont stockés dans un annuaire de service **[05]**. Cet annuaire est semblable à un annuaire téléphonique dans lequel on cherche des numéros pour accéder aux personnes ou services. Donc UDDI à une fonctionnalité majeur qui est la recherche des services fournis par les fournisseurs dans un référentielle de services.

UDDI se caractérise par *XML Schema* pour les messages SOAP et la description de spécification *UDDI APIs* **[19]**

1- *UDDI XML Schema* définit quatre structures de données principales : ***Business entities*** , ***Business Services***, ***binding templates*** et ***tModels***.

- **Entités de business (Business entities)** décrivent des informations sur l'entreprise, y compris leur nom, description, services offertes et des informations de contact.
- **Business Services *:*** offre plus de détaille pour chaque service offert par le fournisseur de service ou l'entreprise.
- **binding templates :** chaque service peut avoir multiple binding templates, chacun décrivant un point d'entré technique pour le service (mailto, http, ftp, etc.).
- **tMode*ls :*** décrit quelles sont les spécifications ou les standards particuliers utilisées par un service.

2- *UDDI APIs* contient les messages pour l'interaction avec l'annuaire UDDI, les APIs sont fournies pour localiser les fournisseurs de services, les services, bindings et tModels. La publication des APIs est incluse pour la création et suppression des données UDDI dans l'annuaire de services. Des APIs des UDDIs sont basées sur le SOAP.

Par exemple la ***figure.1.4.a*** montre la découverte d'un service auprès de fournisseur de services "GetQuote Company " dans l'annuaire de services via l'utilisation d'un message SOAP.

```
< Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
< Body>
  <find_business xmlns="urn:uddi-org:api" generic="1.0">
    <name>GetQuote Company</name>
  </find business>
</Body>
</Envelope>
```

Fig.1.4.a Message SOAP pour la découverte d'un service

La réponse de cette requête sont les services "*stock quote*" et "Portfolio Management" ***figure.1.4.b***.

```
<businessInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3">
      <name>GetQuote Company</name>
       <description xml:lang="en">Features portoflio managers, stock quote and other
            stock market related services</description>
       <serviceInfos>
            <serviceInfo businessKey="0076B468-42E5-AC09-9955CFF462A3"
                 serviceKey="1FFE1F71-B788-09AF7FF151A4">
       <name>Stock Quote</name>
       </serviceInfo>
        <serviceInfo businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
          serviceKey="8BF2F51F-8ED4-43FE-B665-38D8205D1333">
       <name>Portfolio Management</name>
       </serviceInfo>
       </serviceInfos>
</businessInfo>
```

Fig.1.4.b Réponse SOAP de la requête de découverte

1.2.1.2.3 Protocole de description WSDL.

Proposé par IBM et Microsoft, et comme son nom l'indique WSDL (Web Services Description Language) est un langage de description de services web, et plus précisément la description des interfaces auquel sont associées a ces services.

Cette description est faite dans un fichier de description (WSDL) qui se compose de deux grandes parties : ***interface*** et ***implémentation*** de service web **[07]**.

Il est utilisé pour définir la fonctionnalité à un niveau élevé d'un service web en termes de son interface externe et pour décrire comment ce service web peut être consulté au-dessus de Web. C'est, en effet, un modèle abstrait qui a pour format ecrit sur XML pour décrire des services web. **[18]**

Pour qu'une application puisse utiliser un service web, l'interface de ce dernier doit être décrite et spécifié avec précision dans un document WSDL, ce document définit un service web comme une collection de point finaux dans un réseau (network endpoints) ou *ports*. La définition abstraite du endpoints et message sont séparées de leur format de données dans le réseau **[19]**. Ceci permet la réutilisation des définitions abstraites :

- *Messages :* qui sont des descriptions abstraites de données échangées.
- *Port type :* qui sont des collections abstraites d'opérations.

Les caractéristiques concrètes et spécification du format et protocole pour un port particulier constituent une attache réutilisable (reusable binding).Un port est défini en associant l'adresse réseau du service avec le reusable binding, et une collection de ports définit un service, par conséquent, un document de description WSDL utilise les sept éléments ***figure.1.5*** suivants dans la définition de service web : ***Type*** : un récipient pour la définition de types de données.

Message : Une définition abstraite de données échangées

Operation : Une description abstraite de l'action supportée par le service.

Port Type : un ensemble abstrait d'opérations supportées par un ou plusieurs endpoints.

Binding : une spécification concrète de format de données et protocole pour un *port type* particulier.

Port : un endpoint définit comme une combinaison de l'adresse réseau de service et binding.

Service : collection de endpoints

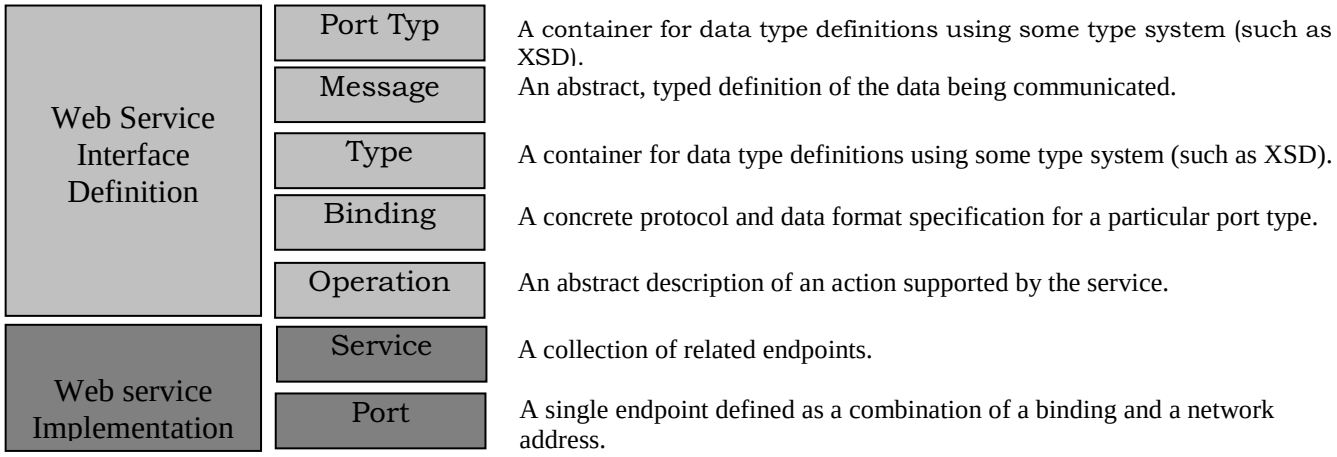


Fig. 1.5 Architecture WSDL

La ***figure.1.6*** illustre un exemple d'une description WSDL comme elle trouvé dans les spécifications de W3C pour la version 1.2 de WSDL. Cet exemple est pour un service réservation de vol, cette description est utilisée pour répondre aux questions (requêtes) demandant les vols disponibles pour une date particulière et l'heure entre une ville spécifique d'origine et une destination. Afin de faire une réservation pour un vol spécifique identifié en terme de son numéro de vol, noms de villes d'origine et destination, et donner le temps de départ.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="TicketAgent"
   targetNamespace="http://airline.wsdl/ticketagent/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:tns="http://airline.wsdl/ticketagent/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsd1="http://airline/">
   <import location="TicketAgent.xsd" namespace="http://airline/"/>
   <message name="listFlightsRequest">
      <part name="depart" type="xsd:dateTime"/>
      <part name="origin" type="xsd:string"/>
      <part name="destination" type="xsd:string"/>
   </message>
   <message name="listFlightsResponse">
      <part name="result" type="xsd1:ArrayOfString"/>
   </message>
   <message name="reserveFlightRequest">
      <part name="depart" type="xsd:dateTime"/>
      <part name="origin" type="xsd:string"/>
      <part name="destination" type="xsd:string"/>
      <part name="flight" type="xsd:string"/>
   </message>
   <message name="reserveFlightResponse">
      <part name="result" type="xsd:string"/>
   </message>
   <portType name="TicketAgent">
      <operation name="listFlights" parameterOrder="depart origin destination">
         <input message="tns:listFlightsRequest" name="listFlightsRequest"/>
         <output message="tns:listFlightsResponse" name="listFlightsResponse"/>
      </operation>
      <operation name="reserveFlight" parameterOrder="depart origin destination
flight">
         <input message="tns:reserveFlightRequest" name="reserveFlightRequest"/>
         <output message="tns:reserveFlightResponse" name="reserveFlightResponse"/>
      </operation>
   </portType>
</definitions>
```

Fig. 1.6 Exemple d'une description WSDL

1.2.2 Composition de services web.

Les services web sont des applications accessible sur Internet réalisant chacune une tache bien spécifiée, par exemple pour réserver un hôtel sur Internet un service web disponible sur Internet peut faire cette requête, mais généralement pour remédier à des requêtes complexe comme par exemple solliciter un service de départ au vacances ou réservation de voyage cela nécessite l'assemblage de plusieurs services web (réservation hôtel, avion, voiture..). Dans ce cas la en parle de la ***composition de service***.

Donc en peut dire que la composition de service est l'opération de regrouper ou agréger plusieurs services pour constituer un seul service à l'utilisateur demandeur de ce service.

La composition de service web peut se faire de trois manières **[09]**:

1- **Composition manuelle :** l'utilisateur lui-même qui se charge d'assembler les services fournissant leur besoin on sélectionnant les services web désirée. Cette démarche très complexe qui est au-delà du processus manuel de composition de services.

La complexité vient généralement des sources suivantes : **[08]**

- Le nombre de services web disponible est très important, et augmente dramatiquement ces dernières années si que ramène une énorme opération de découverte de service dans l'annuaire avec un temps de recherche très important.
- Les services web sont crées est actualisés (mis à jour) à la volé, donc le système de composition doit connaître le changement des services en temps réel pour prendre une meilleure décision de composition et invocation de services.
- Les services web sont développés par plusieurs organisations et entreprises qui ne présentent pas le mêmes langages et modèles de conceptions de services, cependant il n'existe pas un langage et modèle unique de définition et évaluation de services web

2- **Composition automatique :** comme la composition manuelle de services est une tâche très complexe, il est fortement recommandé d'automatiser le processus de composition de service. Cette approche de composition consiste de concevoir un compositeur qui est en sorte un agent logiciel chargé de sélectionner, composer et interopérer les services impliqués dans le processus de composition.

3- **Composition semi automatique :** dans ce type de composition l'utilisateur maintiendra un certain contrôle dans le processus de composition. C'est une composition qui présente un pas en avant en comparaison avec la composition manuelle, dans le sens q'elle fait des suggestions sémantiques pour aider à la sélection des services web dans le processus de composition.

1.3 Services web sémantique.

Dans cette partie nous allons voir l'évolution du service web sémantique à travers l'introduction du web sémantique, ensuite une description assez détaillée des différents approches et langages développées pour apporter une sémantique aux services web

1.3.1 Web sémantique.

Le web est essentiellement composé de documents écrits en HTML[1], un langage utilisé pour la présentation visuelle d'informations textuelles, images etc. un très grand nombre d'informations présents sur le Web sont conçus pour la consommation, l'utilisateur lui-même qui se charge de consulter et de comprendre le sens des informations**.**

Les informations dans le web peuvent être définies pour que les machines puissent les utiliser et les interpréter non seulement pour les afficher, mais assurer une interopération et intégration entre systèmes et application. Donc pour qu'un échange d'information entre deux machines puisse se faire il faut que ces informations doivent dotées avec un sens pour que la machine comprenne ces informations, qui est l'objectif du web sémantique. **[13]**

Le web sémantique est un extension du web antérieur (web avant année 2000) "*The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation*." Citation de monsieur (Berners-Lee, Hendler et al. 2001) inspirée de **[13]** dans lesquelles les informations ont un *sens* permettant aux utilisateurs de décharger d'une grande partie de recherche de construction et combinaison de résultats. Et cela grâce à la capacité accrue des machines à accéder aux *contenus* des ressources et à effectuer des *raisonnements* sur ceux-ci. Et comme les approches services web et celles du web sémantiques partagent le but commun de rendre l'information sur le web plus accessible aux machines. La nécessité de doter les services web avec une certaine sémantique pour permet aux machines un accès sur en al dit les services web sémantiques **[10].**

[1] HyperText Markup Language : Langage permettant de créer des pages web, il utilise une structure formé avec des balises permettant la mise en forme du texte. Nécessite un navigateur web pour la visualisation.

1.3.2 Service web sémantique.

L'idée de base pour un service web sémantique est de donner un sens et plus précisément une sémantique à un service web pour rendre ce dernier plus compréhensible et facile à interroger par des applications logicielles ou machines sans l'intervention direct de l'être humain.

Comme nous avons le vu dans les sections précédentes le web sémantique est une extension du web, en rajoutant une sémantique au web, et le web service est une application logicielle autonome syntaxiquement compréhensible par n'importe quel système ou application car son interface est conçu par des langages standard comme (XML, Etc.) pour rendre un accès dynamique au web .L'intersection de ces deux (Web sémantique et service web) engendre le service web sémantique **[13]**. Donc le service web sémantique est le résultat du l'évolution syntaxique du web service avec le web sémantique. Ce principe est illustré par la ***figure.1.7*** Pour apporter une sémantique à un service web plusieurs approches et langages de représentation de connaissances pour les services web sont développées, parmi lesquels en peut citer **DALM-S [11]. OWL-S [14], SWDL-S [16], WSMO [17],** que nous allons les voir en détaille dans les sections suivantes

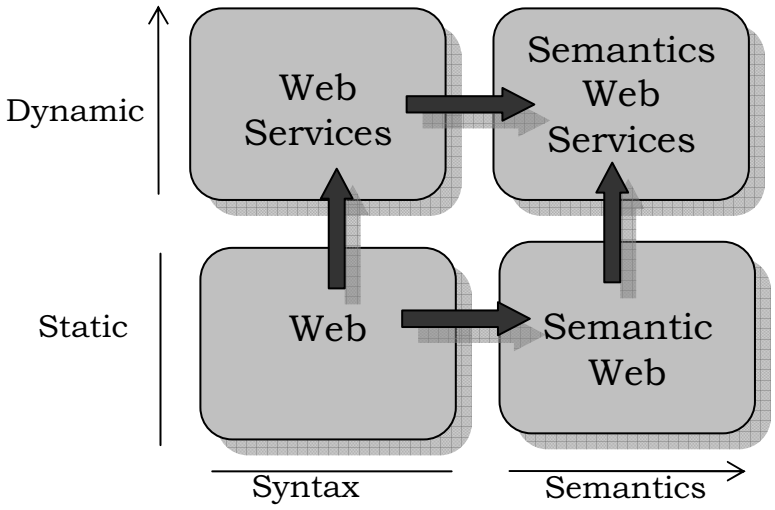


Fig. 1.7 la nature du service web sémantique

1.3.3 Le langage OWL-S.

OWL-S anciennement appelé DAML-S, est un langage de description de service Web sémantique basé sur le langage OWL[1] et (DAML+OIL)[2] , a pour but d'ajouter une description sémantique au services web en plus de la description syntaxique WSDL.

OWL-S est un langage offre une description sémantique du service web permettent de donner une plus grande flexibilité et d'une manière automatique pour invoquer, composer et exécuter les services web par des applications logicielle. Car l'aspect sémantique ajouté au service web rendre ce dernier accessible et facile à manipuler par les autres services pour l'invocation, exécution et la composition avec une possibilité automatisée sans l'intervention de l'être humain. Dans ce que suit, on va donner les aspects motivants l'intégration du OWL-S aux autres standard utilisées par le service web **[14]** :

- **Découverte automatique de services web :** La découverte automatique de services web et un processus automatique permettant de localiser automatiquement le service web dans l'annuaire de services. Actuellement cette tâche est faite manuellement par l'être humain qui se charge d'utiliser un moteur de recherche ou consulter la page web contenant le service, ensuite exécuter le service et voir si les résultats fournis par le service correspond a son besoin. Avec OWL-S, l'information nécessaire pour la découverte automatique de service web est spécifiée sémantiquement dans les pages web contenant les services web, permettant une interprétation automatique par machine des propriétés et capacités des services.
- **Invocation automatique de services web :** l'invocation automatique de services web est le processus d'invocation par un programme informatique ou agent logiciel. OWL-S décrit le domaine des ontologies spécifié par OWL fournis des propriétés déclaratives sous forme des API contenants une sémantique des arguments d'entrées et sorties (Input/output) pour permettre au machines de comprendre quelles sont les informations nécessaires pour

[1] Ontology Web Language : un langage de représentation d'ontologies pour le Web Sémantique.
[2] Darpa Agent Markup Language , Ontology Inference Layer : prédécesseur du OWL qui est aussi un langage de représentation sémantique pour les ressources web (pour plus de détaille voir http://www.w3.org/TR/daml+oil-reference)

exécuter le service web ainsi que les informations retournées par le service exécuté.

- **Composition et interopérabilité de services web :** cette tâche implique la sélection, composition et interopérabilité automatique de services web pour résoudre certaines tâches complexe nécessitant l'utilisation de plusieurs services web. Pour atteindre cet objectif automatiquement sans l'intervention de l'être humain. OWL-S fourni les informations nécessaires pour la sélection et composition de services web encodées dans des sites web, ces informations sont interprétables par des machines.

OWL-S présente la description sémantique du service web sous formes de trois sous-ontologies reliées entre eux qui sont : le profil (***Profile***), le modèle de processus (***ProcessModel***) et les liaisons avec le service (***Grounding***) **[14][15]**. De manière générale, **ServiceProfile** fournit les informations nécessaires pour que l'agent logiciel puisse découvrir le service tandis que **ServiceModel** et **ServiceGrounding** fournissent assez d'informations permettant l'utilisation du service une fois trouvé par l'agent logiciel

La figure.1.8 montre ces trois concepts. Chacune d'elles répondre à un besoin bien défini, ce que nous allons le voir en détail :

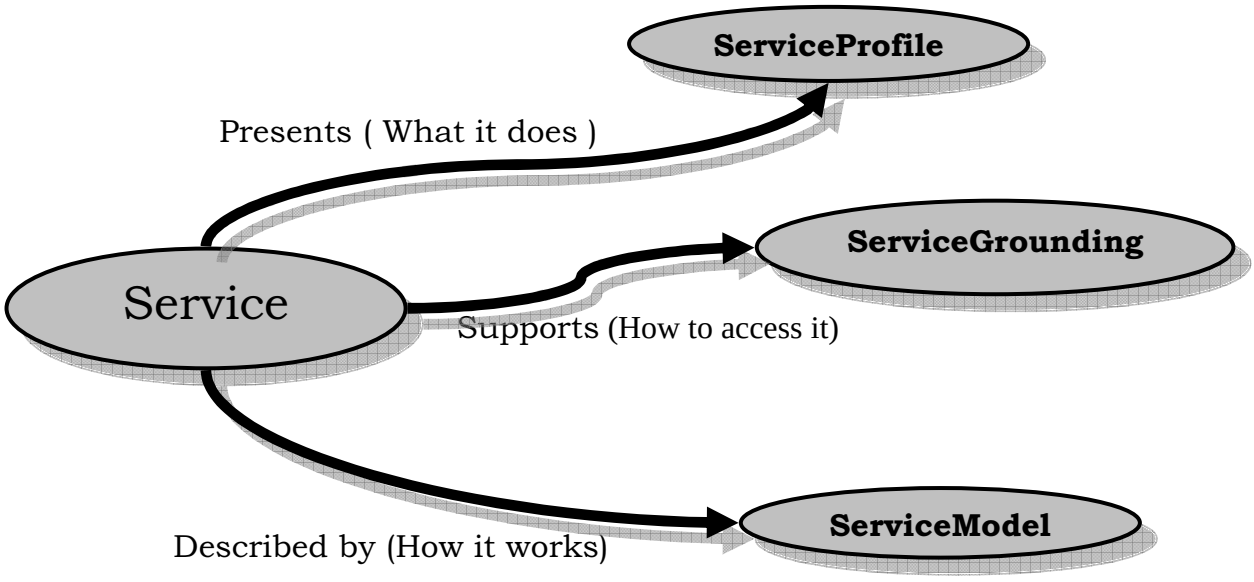


Fig. 1.8 Ontologie de services

Service Profile : il donne une réponse au questions suivantes : Quelles sont les informations concernant l'agent utilisateur (humain ou logiciel) nécessaires à l'exécution du service ? Quelles sont les informations renvoyées ? « What it Does ? ». Service profile donne une présentation sémantique inclus la description de quoi on attend du service (fonctionnalités de service), sa limitation applicable, qualité de service et les conditions requises par l'agent demandeur de service pour qu'il puisse exécuter le service avec succès.

Service ProcessModel : Donne les conditions nécessaires à l'exécution d'un service ainsi que ces conséquences. Il inclut des informations concernant les entrées, sorties, préconditions et effets d'un service. Dans le cas d'un service complexe (composé de plusieurs étapes dans le temps), le modèle de processus détaille la décomposition du service en composants plus simples en en explicitant les liaisons et structures de contrôle permettant l'enchaînement de ces composants.

Service Grounding : il spécifié la manière, pour un agent, d'accéder au service concerné. Typiquement, elle spécifie les protocoles de communication à utiliser ainsi que les éléments spécifiques au service comme, par exemple, les ports à utiliser : elle fait le lien entre la description OWL-S et la spécification WSDL (***figure.1.9* image inspiré du [14])**. De plus le *Service Grounding* doit définir, pour chaque type abstrait spécifié dans le *Service Model*, une façon non ambiguë d'échanger des données de ce type avec le service.

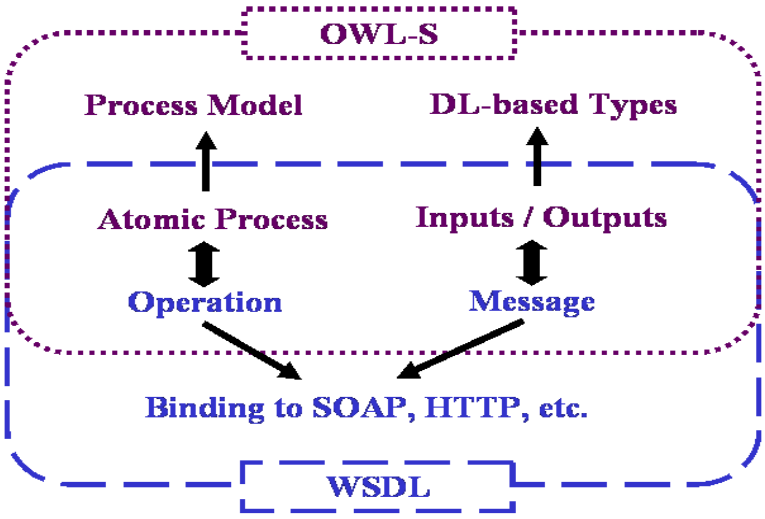


Fig. 1.9 Relation entre OWL-S et WSDL

1.3.4 Le langage SWDL-S.

L'une des solutions pour créer des services web sémantique est le (tracement) ***mapping*** des concepts offert par le WSDL c'est-à-dire la description de service web aux concepts d'ontologie. En utilisant cette approche, les utilisateurs peuvent définir explicitement la sémantique du service web pour un domaine donné. **[13]**

Avec l'utilisation des ontologies, la sémantique ou le sens des données de service ainsi que son fonctionnalité sont ajoutées au ficher de description de web service pour donner une description sémantique du service web. Le résultat de cette approche d'intégration et composition automatique de service est abouti avec un degré élevé de sucées.

WSDL (Web service description language) présente la description syntaxique du service qui présente une absence totale du sémantique pour le web service web. (***R.Akkiraju & all)*** dans **[16]** propose une nouvelle approche qui se base sur l'ajout des annotations sémantique aux documents de description WSDL. Cette approche est guidée par les principes suivants :

- ***Développement d'une approche basée sur les standards utilisés dans les services web*** : les standards utilisés dans les services web (http, FTP, SOAP, WSDL, UDDI..) sont rapidement devenus les technologies préférées pour l'intégration et interopérabilité d'application, car ils présentent une forte interopérabilité entre application et systèmes hétérogènes. Et comme la majorité des entreprises et organisations mettent leurs investissements sur les projets d'intégrations basées sur les services web. ***R.Akkiraju et all*** croient que Le développement d'une approche basée sur l'ajout sémantique au service web revient fortement à utiliser ou faire une extension aux standards existants, dans leurs cas ou ils ont utilisés WSDL.
- ***Le mécanisme d'annotation sémantique de services web doit être obligatoirement indépendant aux langages de représentations sémantiques :*** il y a plusieurs langages de représentation sémantique parmi lesquelles en peut citer OWL, WSMO, UML[1]. Chaque langage offre différents niveaux d'expression sémantique et supports de développement. Donc il est fortement recommandé de développer une approche qui n'est pas attaché à un langage

[1] Unified Modeling Language. Il s'agit d'un langage de modélisation de 3ème génération. Voir http://www.uml.org/

de représentation sémantique particulier, cette approche qui utilise un mécanisme de représentation sémantique séparée au langage sémantique utilisé offre une grande flexibilité permettant aux développeurs de sélectionner le langage de représentation sémantique favorisé pour eux.

- ***Le mécanisme d'annotation sémantique de services web permet une association de multiples annotations écrites par différents langages de représentation sémantique :*** nous avons vu précédemment qu'il y on a plusieurs langages de représentation sémantique. Les fournisseurs de services web ont le choix de langage sémantique utilisé pour l'annotation sémantique de leurs services web. Donc le mécanisme d'annotation sémantique associé aux services web doit présenter une possibilité d'annotation sémantique de différents langages sémantiques.
- ***Supporter l'annotation sémantique des services web dont leurs types de données décrits avec l'XML schema*** : une pratique courante dans les services web basée sur intégration est la réutilisation des interfaces décrites en XML, la définition des documents d'affaires (business documents) utilisant XML shema est une pratique répandue et réussie. XML schema est une définition importante de formats de données dans le futur, les auteurs croyons que l'annotation sémantique des entrées et sorties de service web doit supporter l'annotation XML schemas.
- ***Fournir l'appui pour des riches traçant des mécanismes entre types schema de services web et ontologies* :** il est important de faire l'annotation XML schemas dans les descriptions de services web, l'attention devrait être donnée au problème de tracer (mapping) les types complexes de XML schemas au concepts d'ontologie.

WSDL-S est un langage de description sémantique développé par IBM et l'université de Georgia, ce langage est une extension du WSDL utilise un mécanisme d'annotation sémantique aux services web. Dans WSDL-S l'expressivité et la description WSDL de service web ont augmenté sémantiquement en utilisant les mêmes concepts semblables au OWL-S. Les avantages de cette évolution en ajoutant l'aspect sémantique au WSDL sont **[16]** :

- les utilisateurs et d'une manière compatible vers le haut décrivent les niveaux sémantiques et opérationnels dans WSDL qui est un langage reconnu par les développeurs
- une description sémantique indépendamment au langage de représentation sémantique utilisé, permettant aux développeurs de choisir le langage sémantique adéquat est préféré.

Dans la section suivante, nous allons vous présenter brièvement l'annotation sémantique de trois éléments essentiels du document WSDL.

1.3.4.1 Annotation sémantique du document WSDL.

Elle est basé sur l'analyse de la description WSDL, trois types d'éléments peuvent faire augmenter leur sémantique à travers l'annotation sémantique avec les concepts d'ontologies qui sont : *Opération, Messages, Préconditions et Effets*. Tous ces éléments existent dans le fichier WSDL. **[13]**

Opérations : Chaque description WSDL peut avoir un certain nombre d'opérations avec différentes fonctionnalités. Par exemple une description WSDL peut avoir des opérations de réservation et annulation de billets d'avion. Et afin de décrire leurs fonctionnalités l'ajout et l'annotation sémantique (mapping) de ces opérations avec des concepts d'ontologies vont donner au plus de la description syntaxique une description sémantique des fonctionnalités de ces opérations.

Messages : les messages qui sont les paramètres d'entrée et sortie (Input/output), des opérations sont définies en WSDL en utilisant XLM schema. Les ontologies qui sont plus expressive que XML schema peut être utilisées pour l'annotation sémantique des messages WSDL. L'utilisation des ontologies apporte non seulement la publication de service dans un espace conceptuel commun mais aider l'utilisation et l'application des mécanismes de raisonnement.

Préconditions et effets : chaque opération décrite dans un document WSDL peut avoir un nombre de préconditions et effets. Les préconditions sont des conditions logiques qui doivent être vérifiées afin d'exécuter l'opération concernée, et les effets sont les changement ou les résultats qui se produisent après l'exécution de l'opération vérifiant les conditions logiques. Après l'annotation sémantique des opérations et messages (input/output) les préconditions et effets peuvent également annotées. L'annotation sémantique des préconditions et effets est importante pour les services web, puisqu'il est

possible pour un q'un nombre d'opérations ont les mêmes fonctionnalités et mêmes paramètres d'entrée sortie mais qui produisent des différents effets.

```
< ? XML version = "1.0 " encoding =  "UTF-8 " ?>
<definitions name =  "StudentManagement "
targetNamespace=
 "http :…/StudentMnagement.wsdl20 "
              xmlns=  "http://www.w3.org/2004/03/wsdl "
              xmlns:tns= "http…/StudentManagement.wsdl20 "
              xmlns:sm = " http :…/StudentMng.ow# "
              xmlns:mep=http:…./TR/wsdl20-patterns>
<interface name =  "StudentManagementUMA ">
       <operation name =  "RegisterStudent "
              pattern =  "mep :in-out ">
       <action element =

 "sm :RegisterStudent "/>
       <input messageLabel =  "student "
                     element = "sm :StudentInfo "/>
       <output messageLabel =  "ID "
                     element  =  "sm :StudentID "/>
       </operation>
       <operation name =  "StudentInformation "
              pattern =  "mep :in-out ">

              < action element =

        "sm :StudentInformation "/>
              <input messageLabel =  "ID "
                           element =  "sm :StudentID "/>
              <output messageLabel =  "student "
                           element =  "sm :StudentInfo "/>
       </operation>
       <operation name =  "checkStatus "
         pattern =  "mep :in-out ">
……….
       </opertaion>
 </interface>
</definitions>
```

Fig. 1.10 Exemple WSDL-S

Dans l'exemple WSDL-S de la ***figure.1.10*** on trouve deux opérations : 'RegisterStudent' et 'StudentInformation'. La première opération à une entrée appelée 'student', décrit sémantiquement par le concept ontologique 'sm :Student' et une sortie appelé 'ID' , décrit sémantiquement par le concept ontologique 'sm :StudentID'.

L'opération 'RegisterStudent' est annotée sémantiquement avec le concept ontologique 'sm :RegisterStudent'. La seconde opération 'StudentInformation' utilise des concepts d'ontologie semblable à celle de l'opération 'RegisterStudent' pour annoter sémantiquement l'entré, sortie et action de l'opération 'StudentInformation'. Les concepts d'ontologie utilisés sont exprimés dans http://dme.uma.pt/jcardoso/StudentMng.owl#, qui est spécifie avec le langage sémantique OWL.

1.3.5 Le langage WSMO.

WSMO est un nouveau framework proposée par Cristina Feier et all **[17]** qui est une extension et prolongement du langage WSMF[1] pour la description sémantique de service web. WSFM définit un modèle conceptuel riche pour le description et développement de services web basé sur deux conditions principales : *Découplage maximal* et une *forte médiation.* Dans **[17]** les auteurs ont donnés les concepts principaux du WSMO à savoir :

Conformité du Web : WSOM hérite le concept URI (Universal Ressource Identifier) pour une identification unique de ressources comme un principe essentiel pour la conception du Web.

Utilisation des Ontologies : les ontologies sont utilisées comme un modèle de données dans WSMO, signifiant que toutes les descriptions de ressources à savoir toutes les données durant l'utilisation de service est basée sur des ontologies. L'utilisation des ontologies augmente le traitement automatique de l'information ainsi qu'une meilleure interopérabilité entre services et applications.

Découplage strict : chaque ressource WSMO est spécifiée indépendamment par rapport aux autres ressources. Qui est motivé par la nature ouverte et distribuée du web.

Centralité de la médiation : la médiation adresse la manipulation des environnements ouverts hétérogènes, et afin de complémenter le principe de *découplage strict*, WSMO donne une importance de la médiation pour obtenir un déploiement réussi de service web par la mise on œuvre de la médiation en premier ordre de l'architecture du framework WSMO.

Séparation de rôle d'ontologies (Ontological Role Separation) : les demandes des utilisateurs sont formulées indépendamment et dans un contexte différent pour les services

[1] Web Service Modeling Framework

web disponibles. L'épistémologie fondamentale du WSMO met une différence entre ce désirent les utilisateurs avec les services web disponibles.

1.3.5.1 les concepts basics du WSMO.

WSMO définit quatre éléments nécessaires pour la description sémantique de service web, en se basant sur les composants déjà existants avec WSMF qui sont : *Ontologies (Ontologies), Services web (Web Services), buts (Goals) et médiateurs (Mediators).* WSMO hérite ces quatre éléments de WSMF un plus de raffinage et extension de celles-ci.

1. Ontologies : les ontologies représentent l'élément principal dans WSMO puisqu'ils fournissent un domaine spécifique de terminologies pour la description des autres éléments. En effet les ontologies atteignent un objectif double : la définition formelle de la sémantique de l'information et la liaison entre la machine et les terminologies humaines.

2. Services web : les services web connectent des ordinateurs et dispositifs en utilisant des protocoles standards du Web, pour échanger et combiner des données d'une nouvelle manière. La distribution de données offre l'avantage d'indépendance de plateforme. Chaque service web représente une entité atomique de fonctionnalité qui peut être réutilisée pour construire des nouveaux services plus complexes. Les services web sont décrits dans WSMO selon trois perspectives différentes : *Non fonctionnel* (non-fonctional), *propriétés* (properties) et finalement *fonctionnalité et comportement* (functionality and behavior).

3. Buts (Goals) : Indiquent les objectifs qu'un client pourrait avoir en invoquant un service web

4. Médiateurs (Mediators): les médiateurs visent à surmonter les disparités qui apparaissent entre les différents composants qui accumulent une description WSMO. L'existence de médiateurs permet de lier probablement les ressources hétérogènes et résoudre des problèmes d'incompatibilité et cela selon différents niveaux :

- ***niveau de données :*** médiation entre différents terminologies utilisées, plus spécifiquement la résolution du problème d'intégration d'ontologies.
- ***niveau de processus :*** médiation entre les modèles hétérogènes de communication. Ce genre d'hétérogénéité apparaît pendant la communication entre les différents services web.

Conclusion

Les technologies de services web jouent un rôle très important dans le nouveau paradigme des architectures orientées service, permettant un échange de données et dialogue entres les différentes applications éparpillées dans le web et cela indépendamment de plateformes et langages sur lesquelles elles reposent.

Pour assurer cette portabilité de communication entre ces différentes applications plusieurs protocoles et standards sont utilisés (SOAP, WSDL, UDDI).

Plusieurs organisation et entreprises utilisent cet nouveau concept de service web pour rendre la possibilité de réutilisation d'autres services existants possible et cela d'une manière automatique permettant aux machines d'utiliser les services web sans l'intervention de l'être humain. Et comme les services web ne présentent pas un aspect sémantique qui offre la possibilité aux machines de comprendre les fonctionnalisées de services web, la combinaison du service web avec le web sémantique engendre ce que n'appel le *service web sémantique* la ou au plus de la description syntaxique la description sémantique (fonctionnalités et capacités) est ajoutée aux services web via les différents langages et approches sémantiques de services web développées comme par exemple (OWL-S, WSMO, WSDL-S…).

Nous avons vu que généralement pour résoudre certains problèmes ou services complexes sollicités par les utilisateurs dans le web revient à combiner et regrouper plusieurs services simples ou atomiques pour résoudre ces problèmes, ce mécanisme de regroupement de services web est appelé la *composition de service web*. La composition de services web faite selon l'une de trois approches suivantes :

- *manuellement* par l'intervention directe de l'utilisateur qui se charge de sélectionner et assembler les services impliquées dans le processus de composition.
- *semi automatique* ou l'utilisateur maintiendra un certain contrôle dans le processus de composition.
- purement *automatique* via l'utilisation d'un agent logiciel ou compositeur qui se charge de sélectionner, invoquer et composer les services web impliquées.

La dernière approche présente plusieurs avantages parmi lesquels l'abstraction dans le processus de composition dont lequel l'utilisateur ou un autre service web ne voie pas comment sa demande de service complexe (requête complexe) est sollicitée. La question qui se pose comment et quelles sont les approches utilisées permettant la composition automatique de services web. Ce que nous allons le voir dans le chapitre trois.

Chapitre II

Planification classique et hiérarchique en intelligence artificielle

Table des matières :

Introduction.

Les recherches, qui ont menées à la planification actuelle, ont commencé dans les années 60 sous la forme de programmes dont le but était de simuler la capacité de raisonnement de l'être humain. Un des premiers programmes fut le General Program Solver (GPS) par Newell et Simon **[36]**. GPS fonctionne par recherche dans un espace d'états. Cette recherche est guidée par le calcul des différences entre l'état courant et l'état final (i.e., le but) du problème. À la fin des années 60, Green propose l'utilisation de prouveurs de théorèmes **[37]**. Cependant, du fait de l'immaturité des techniques de preuves de théorèmes à cette époque, l'approche fut abandonnée au profit d'algorithmes de planification plus spécialisés. Ces algorithmes, basés sur une approche déductive de la logique ont pris leur essor à la fin des années 90 conséquemment au développement de programmes plus sophistiqués pour la résolution du problème de satisfaisabilité de la logique propositionnelle classique **[38].**

Dans ce chapitre, nous introduisons les concepts principaux de la planification ainsi que les algorithmes fédérateurs.

2.1 La planification classique et hiérarchique

2.1.1 Vue d'ensemble du domaine de la planification

Le domaine d'application par excellence de la planification est le domaine de la robotique. En effet, un robot « intelligent » est un mécanisme informatique qui a des capteurs

lui permettant d'observer son environnement puis d'en construire une représentation. Pour être utile, il doit être capable d'agir sur son environnement. Un robot a des effecteurs, qui sont des périphériques lui permettant d'interagir avec cet environnement, c'est-à-dire se déplacer, prendre des objets …etc. À un niveau abstrait, un robot est un mécanisme qui met en correspondance les observations, obtenues par ses capteurs, avec les actions qu'il peut effectuer en utilisant ses effecteurs. La planification est la prise de décisions faite en produisant une séquence d'actions à partir d'une séquence d'observations ***figure2.1.***

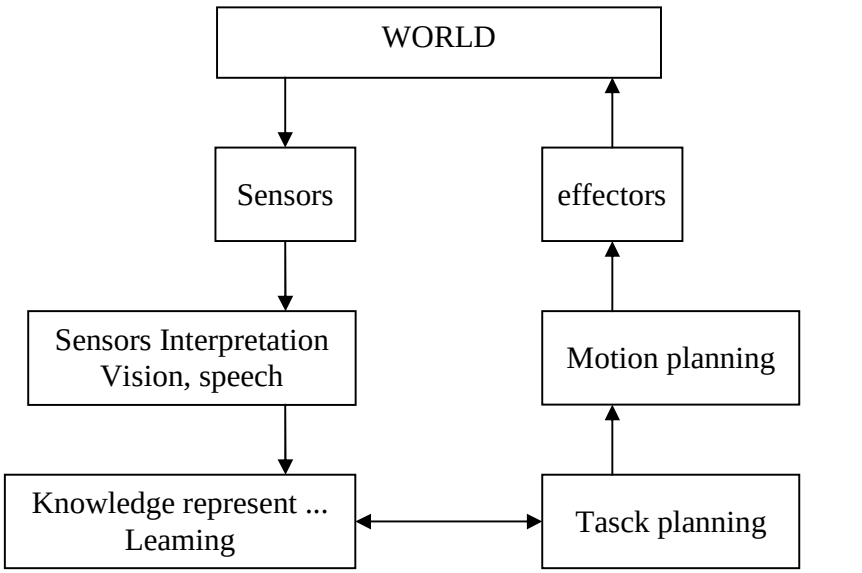


Fig.2.1 Architecture logiciel d'un robot

Le mot « planification » est très général et peut prendre de nombreux sens, même dans le domaine de l'intelligence artificielle et de la robotique. En effet, comme il existe de nombreux types d'actions, il existe de nombreux types de planifications parmi lesquels, dans le cadre du robot, la planification de chemins, de mouvements, de perceptions et de navigations. La planification de chemins (path planning) est nécessaire pour permettre au robot de se déplacer d'un endroit à un autre. La planification de mouvements (motion planning) lui permet d'interagir avec l'environnement : déplacer ses « pieds », saisir des objets. Le robot a également besoin de planifier sa perception (perception planning). Cette planification intervient dans les tâches telles que modéliser l'environnement, identifier les objets, localiser un objectif, ou plus généralement identifier l'état courant de l'environnement. La planification de

navigation (navigation planning) combine la planification de mouvements et de perceptions pour atteindre un but ou pour explorer une zone de l'environnement.

Afin de réaliser ces différentes planifications, deux approches sont envisageables : l'approche dite « spécifique au domaine » et l'approche dite « indépendante du domaine ».

L'approche spécifique au domaine est une approche qui consiste à adresser chaque problème avec une représentation spécifique et avec des techniques adaptées au problème. Cependant, on peut reprocher à cette approche les points suivants :

- les points communs entre les différents domaines de planification ne sont pas mis en facteur ;
- il devient vite coûteux de développer un planificateur pour chaque domaine abordé ;
- l'approche domaine-specifique n'est pas satisfaisante pour élaborer une machine autonome et intelligente. Ses capacités délibératives seraient limitées aux domaines adressés par le planificateur utilisé.

Le but de l'approche indépendante du domaine est de fournir un cadre général à la résolution des problèmes. Pour résoudre un problème particulier, un planificateur domaine-indépendant prend en entrée les spécifications du problème, mais également les connaissances concernant le domaine. Cette approche se fonde sur des modèles abstraits et généraux des actions dont les principaux sont :

- La planification de projets dans laquelle les modèles des actions sont réduits principalement à des contraintes temporelles et des contraintes de priorité ;
- L'ordonnancement et l'allocation de ressources dans lesquels on ajoute des contraintes sur les ressources qui sont utilisées dans les actions ;
- la synthèse de plans dans laquelle le modèle d'actions est enrichi par l'ajout de conditions nécessaires à l'exécution des actions ainsi que les effets produits sur le monde lors de ces exécutions.

Dans la suite de ce document, par planification, nous faisons référence à la synthèse de plans

2.1.2 Modèle conceptuel de la planification

Le modèle conceptuel de la planification peut être représenté comme un système de transition d'états, c'est-à-dire un quadruplets $\Sigma = (S, A, E, \gamma)$ ou :

$S= \{S_1,S_2,S_3,S_4,\ldots\ldots S_n\}$ est un ensemble fini et récursivement énumérable d'états ;

$A= \{a_1,a_2,a_3,a_4,\ldots\ldots a_n\}$ est un ensemble fini et récursivement énumérable d'actions ;

E= {e1,e2,e3,e4,…….en} est un ensemble fini et récursivement énumérable d'événements ;

$\gamma : S \times A \times E \longrightarrow 2^S$ est une fonction de transition entre états.

Ce système de transition d'états peut être représenté par un graphe direct dont les noeuds sont les états de S. Les arcs sont appelés transitions d'états.

Le modèle de la planification peut être décrit par l'interaction entre trois composants ***Figure.2.2***

- le système de transition d'états qui évolue par sa fonction de transition entre états en fonction des événements et des actions qu'il reçoit ;
- un contrôleur qui prend en entrée l'état s du système et qui fournit une action a en fonction du plan à exécuter ;
- un planificateur qui prend en entrée une description du système Σ , une situation initiale et des objectifs à atteindre. Il synthétise un plan en fonction des objectifs puis le transmet au contrôleur.

La ***figure.2.3*** montre un système de transition d'états impliquant un container, une grue qui est capable de charger et de décharger le container, ainsi qu'un robot qui peut transporter ce container d'un endroit à un autre. Dans cet exemple :

- L'ensemble d'états est S = {s0, s1, s2, s3, s4, s5} ;
- l'ensemble d'actions est A = {take, put, load, unload, move1, move2};
- l'ensemble d'événements E est vide ;
- la fonction de transition y est définie par : si a est une action et y(s, a) est non vide alors a est applicable à l'état s.

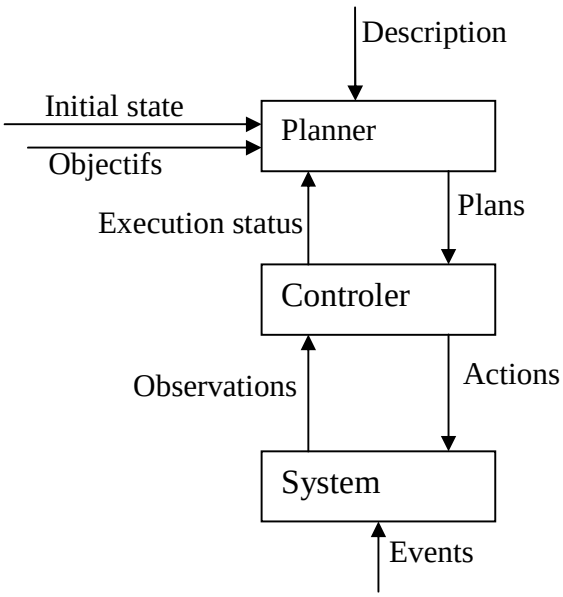


Fig.2.2 Modèle conceptuel de la planification

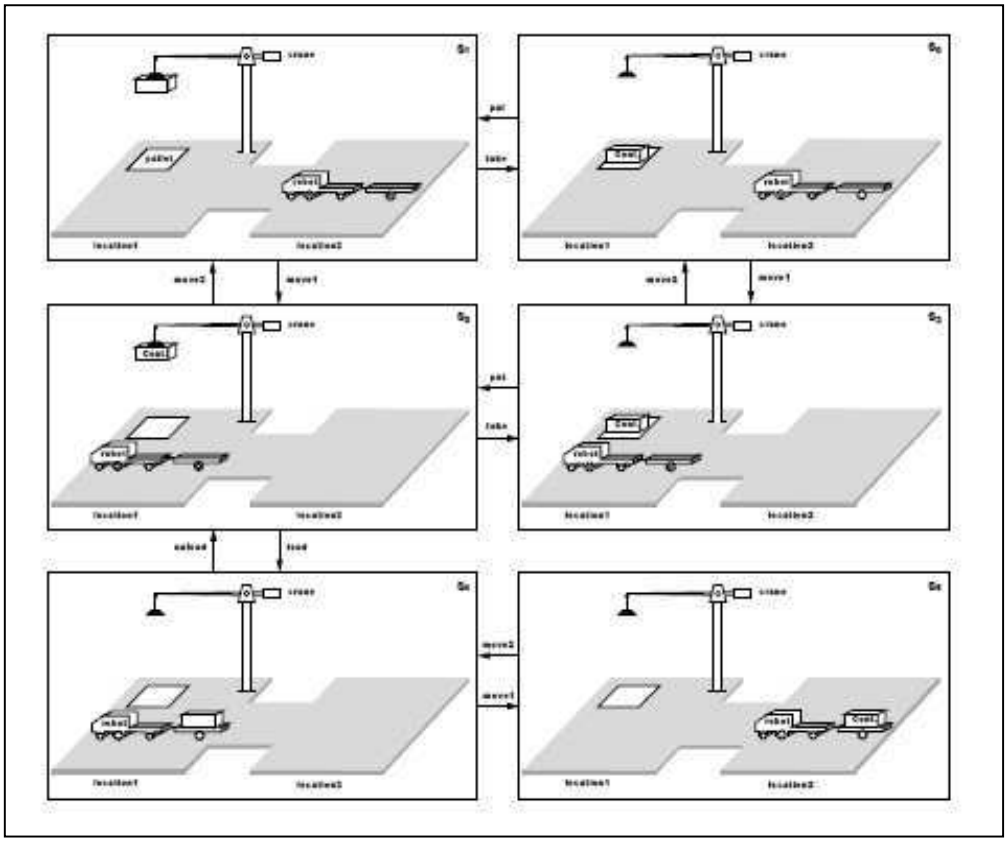


Fig.2.3 Exemple de système de transition d'états

Le modèle conceptuel ainsi défini n'est pas directement opérationnel. Il est nécessaire d'émettre un certain nombre d'hypothèses restrictives, dont les principales sont :

- Le système Σ a un ensemble fini d'états ;
- Le système Σ est totalement observable, i.e., la connaissance concernant les états de Σ est complète ;
- Le système Σ est déterministe, i.e., si une action est applicable à un état, le résultat de cette application est un état unique ;
- Le système Σ est statique, i.e., l'ensemble d'événements E est vide. Σ n'a pas de dynamique interne ;
- Le planificateur ne manipule que des buts restreints qui sont spécifiés sous la forme d'états à atteindre ;
- Un plan solution d'un problème de planification est une séquence d'actions linéaire, ordonnée et finie ;
- Les actions n'ont pas de durée. Ce sont des transitions instantanées entre états ;
- La planification en cours n'est pas affectée par les changements pouvant intervenir dans le système de transition d'états.

2.1.3 Représentation classique de la planification

Une donnée nécessaire à tout algorithme de planification est la représentation du problème à résoudre. En pratique, il est impossible d'énumérer tous les états et les transitions entre états possibles : la description d'un problème peut être excessivement large, et la générer entièrement nécessite plus de travail que la résolution du problème. Pour résoudre cette difficulté, il faut un langage de représentation du problème permettant de calculer ces états et ces transitions entre états à la volée.

Dans la littérature, on recense trois façons de représenter les problèmes de planification classique **[39].** Ces trois représentations ont le même pouvoir d'expression, un domaine de planification exprimé dans l'une de ces représentations peut être exprimé similairement en utilisant une autre représentation.

- **la représentation dans la théorie des ensembles (*Set-theoretic representation*):** chaque état du monde est un ensemble de propositions et chaque action est une expression spécifiant les propositions qui doivent appartenir à l'état courant pour que l'action puisse être exécutée, ainsi que celles qui seront ajoutées et enlevées de l'état suite à l'exécution de l'action;
- **la représentation classique (*Classical representation*) :** contrairement à la représentation précédente, des prédicats du premier ordre et des connecteurs logiques sont utilisés à la place des propositions ;
- **la représentation par variables d'états (*State variable representation*) :** chaque état est représenté par un tuple de ***n*** variables d'états évaluées $\{x_1, x_2, ..., x_n\}$ et chaque action est représentée par une fonction partielle qui permet de passer d'un tuple à un autre tuple de variables d'états instanciées.

Dans la suite, nous introduisons les principaux concepts du domaine de la planification en utilisant le formalisme de la représentation classique qui généralise la représentation basée sur la théorie des ensembles et qui utilise des notations dérivées de la logique du premier ordre. Les états sont représentés sous la forme d'atomes logiques qui sont vrais ou faux dans une certaine interprétation. Les actions sont représentées par des opérateurs de planification qui changent la valeur de ces atomes.

Les états : Soit ***L*** un langage du premier ordre qui ne contient pas de symbole de fonctions. Un état est un ensemble d'atomes instantiés de ***L***. Comme ***L*** ne contient pas de fonctions, l'ensemble ***S*** de tous les états est donc fini. On peut alors représenter l'ensemble des formules de ***L*** sous la forme d'un ensemble de propositions et utiliser les algorithmes classiques du calcul propositionnel. Un prédicat ***p*** est vérifié dans un état ***s*** si et seulement si ***p*** peut être unifié avec un prédicat de ***s*** tel que $\sigma(p) \in s$ ou σ est la substitution résultant de l'unification de ***p*** avec le prédicat de ***s***. Dans le cas contraire, la propriété du monde représentée par ***p*** est considérée comme fausse du fait de l'hypothèse du monde clos : un prédicat, qui n'est pas explicitement spécifié dans un état, est considéré comme faux dans cet état.

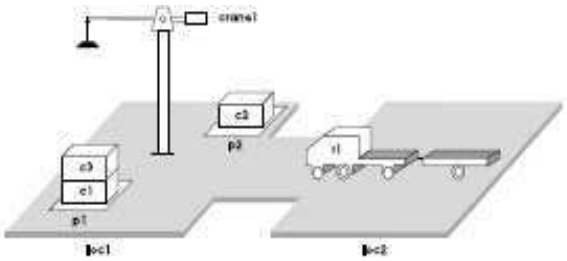


Fig.2.4 Exemple de représentation d'un état

$$S_0 = \left\{ \begin{array}{l} \textbf{attached(}p1,loc1\textbf{), attached(}p2,loc1\textbf{),in(}c1,p1\textbf{),in(}c3,p1\textbf{),} \\ \textbf{top(}c3,p1\textbf{),in(}c2,p2\textbf{),top(}c2,p2\textbf{),empty(}cranel\textbf{),at(}r1,loc2\textbf{),} \\ \textbf{adjacent (}loc1,loc2\textbf{),adjacent(}loc2,loc1\textbf{),occupied(}loc2\textbf{),unloaded(}r1\textbf{)} \end{array} \right\}$$

Les opérateurs et actions. Une action est définie par l'application d'un opérateur de transformation. Le principe de la modélisation des actions par des opérateurs de transformation consiste à spécifier les propriétés du monde nécessaires à son application, i.e., les préconditions de l'action, ainsi que les propriétés du monde engendrées par son exécution, i.e., les effets de l'action. Les préconditions d'un opérateur représentent le domaine de définition d'une action, i.e., l'ensemble des états dans lesquels l'action peut être appliquée. Ce calcul peut être réalisé en unifiant les préconditions positives ***(Precond$^+$)*** et négatives ***(Precond$^-$)*** de l'opérateur avec l'état courant. Les préconditions positives expriment les propriétés qui doivent être vérifiées, par exemple ***unloaded(r1)***. Les préconditions négatives expriment celles qui doivent être absentes de l'état pour que l'action soit appliquée, par exemple ***not(occupied(loc2)).*** Les effets d'un opérateur spécifient les propriétés du monde modifié par l'exécution d'une action. D'un point de vue formel, si une action est définie par un opérateur qui transforme un état s_i en un état s_{i+1}, les effets d'une action sont représentés par les propriétés ajoutées ***(effets$^+$)*** et les propriétés enlevées ***(effets$^-$)*** à s_i pour obtenir s_{i+1}.

Définition. 2.1 (Opérateur)

Un opérateur peut être défini par le quadruplet

o = (name(o), precond(o), add(o), del(o))

- name(o), le nom de l'opérateur, est défini par une expression de la forme n(x_1, . . . , x_k) ou n est un symbole d'opérateur et x_1, . . . , x_k représentent les paramètres de l'opérateur.
- precond(o) représentent les préconditions de l'opérateur o, i.e., les propriétés du monde nécessaires à son exécution.
- add(o) et del(o) définissent deux ensembles de propriétés décrivant respectivement les faits à ajouter et les faits à supprimer de l'état du monde après l'exécution de o.

Exemple d'opérateur. L'opérateur take(k,l,c,d,p) peut être défini de la façon suivante :

;; crane k at location l takes container c off of container d in pile p take(k,l,c,d,p)

precond: belong(k,l), attached(p,l), empty(k), top(k), on(c,d)

add: holding(k,c), top(d,p)

del: empty(k), in(c,p), top(c,d), on(c,d)

Définition. 2.2 (action)

Une action est une instance d'un opérateur. Si ***a*** est une action et s_i un état tel que $\boldsymbol{precond^+(a)} \in \mathbf{s_i}$ et $\boldsymbol{precond^-(a)} \cap \boldsymbol{s_i} = \phi$, alors a est applicable à $\boldsymbol{s_i}$, et le résultat de cette application est l'état :

$s_{i+1} = \gamma(s_i, a) = (s_i - \text{effets}^-(a)) \cup (\text{effets}^+(a)$

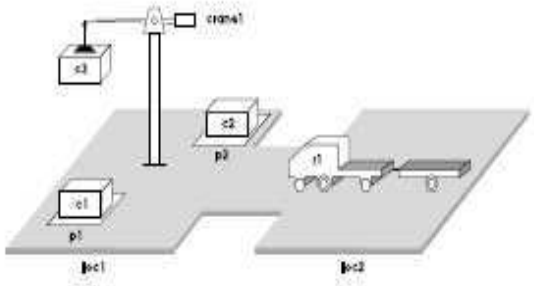


Fig.2.5 Application de l'action take(crane1,loc1,c3,c1,p1)

L'état résultant de l'application de l'action take sur l'état s_0 est alors :

S_1= { **attached(***p1,loc1***), attached(***p2,loc1***),in(***c1,p1***),top(***c1,p1***),**
in(*c2,p2***),top(***c2,p2***),at(***r1,loc2***),adjacent(***loc1,loc2***),**
adjacent (*loc2,loc1***), occupied(***loc2***),unloaded(***r1***),holding (***cranel,c3***),** }

Les domaines et problèmes. En planification, un domaine définit l'ensemble des opérateurs qui peuvent s'appliquer sur le monde. Un problème doit spécifier l'état initial ainsi que le but à atteindre.

Définition. 1.3 (Domaine de planification)

Un domaine D de L est un système de transition d'états restreint $\sum$ = (S, A, γ) tel que :

– S = $2^{\{\text{les atomes instanciés de L}\}}$

– A = {l'ensemble des opérateurs instanciés de O} ou O est l'ensemble des opérateurs

– γ(s, a) = (s − effets^{-} (a)) $\cup$ (effets+ (a) si a $\in$ 2 A et a est applicable à s$\in$ S

Exemple de domaine. Représentation du domaine de planification dans le langage PDDL **[31].**

```
(define (domain (robot-transportation-domain)
    ( :requirement :strips :typing)
    ( :types location pile robot crane container)
    ( :predicates
        (adjacent ?l1 ?l2 - location)(attached ?p - pile ?l - location)
        (belong ?k -crane ?l - location)(at ?r - robot ?c - container)
        (occupied ?l - location) etc.)
    ( :action move
        ( :parameters ( ?r - robot ?from ?to - location))
        ( :precondition (and (adjacent ?from ?to)(at ?r ?from)
            (not(occupied ?to))))
    ( :effect (and (at ?r ?to)(not (occupied ?from))(occupied ?to)
        (not (at ?r ?from)))))
    ( :action load
        etc.))
```

Définition.2.4 (Problème de planification)

Un problème P pour une domaine D est un triplet P = (O,s_0,g) ou :

– s_0, l'état initial, est un état quelconque de S ;

– g, le but, définit un ensemble cohérent de prédicats instanciés, i.e., les propriétés du monde devant être atteints ;

– O est l'ensemble des opérateurs applicables.

Exemple de but :

g_1 = { loaded(r1, c3), at(r1, loc1)}

Les plans. Un plan solution se définit comme un chemin dans un espace d'états. Le passage d'un état à l'autre s'effectue par l'application d'une action, i.e., un opérateur complètement instantié. Par conséquent, un plan solution pour un problème de planification

P = (Q,s_0,g) est une séquence d'actions décrivant un chemin d'un état initial s_0 `a un état final s_n tel que le but g soit inclus dans s_n. Autrement dit, un plan $\prod$ est une solution pour le problème P si $\gamma(s_0, \prod)$ satisfait g.

Exemple de plan. Considérons le plan suivant :

$$\prod = \begin{pmatrix} \textbf{take(}\textit{cranel,loc1,c3,c1,p1}\textbf{),} \\ \textbf{move(}\textit{r1,loc2,loc1}\textbf{),} \\ \textbf{load (}\textit{cranel,loc1,c3,r1}\textbf{),} \end{pmatrix}$$

Ce plan est applicable à l'état s_0 de la ***figure 2.4*** et produit l'état s_3 ***figure 2.6.***

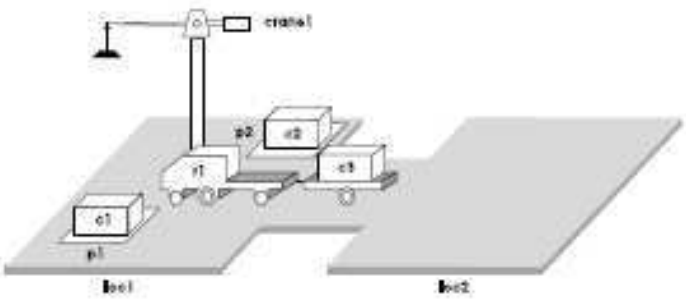


Fig.2.6 Représentation de l'état s_3 contenant le but g_1

$$S_3 = \left\{ \begin{array}{l} \textbf{attached(}p1,loc1\textbf{), attached(}p2,loc1\textbf{),in(}c1,p1\textbf{),top(}c1,p1\textbf{),} \\ \textbf{in(}c2,p2\textbf{),top(}c2,p2\textbf{),at(}r1,loc1\textbf{),adjacent(}loc1,loc2\textbf{),} \\ \textbf{adjacent (}loc2,loc1\textbf{), occupied(}loc1\textbf{),loaded(}r1,c3\textbf{),empty (}cranel\textbf{),} \end{array} \right\}$$

Extensions. Le formalisme de planification classique étant très restreint, des extensions sont nécessaires afin de décrire des domaines plus complexes. Ces principales extensions sont le typage des variables, les opérateurs de planification conditionnelle, les expressions de quantification, les préconditions disjonctives ou encore les axiomes d'inférences. Le langage de planification **PDDL** (Planning Domain Description Language) **[40]** permet d'exprimer ces différentes extensions (cf. Exemple de domaine).

2.1.4 Espace d'états et espace de plans

Les algorithmes de planification classique peuvent être regroupés en deux grandes familles. D'un côté, les algorithmes de planification dans un espace d'états (*state space planning*) qui sont les algorithmes les plus simples ; de l'autre côté, les algorithmes de planification dans un espace de plans (*plan space planning*).

2.1.4.1 Planification dans un espace d'états

Les algorithmes de planification dans un espace d'états sont des algorithmes de recherche dans lesquels l'espace de recherche est un sous-ensemble de l'espace d'états : chaque noeud correspond à un état du monde, chaque arc correspond à une transition entre deux états, et le plan courant correspond au chemin courant dans l'espace de recherche. Les principaux algorithmes de planification dans un espace d'états sont la recherche par chaînage avant, la recherche par chaînage arrière et l'algorithme STRIPS **[41]** :

- **la recherche par chaînage avant :** partant de l'état initial, l'objectif est de trouver un état qui satisfait le but. L'algorithme prend en entrée un problème P. Si P est résolvable, alors l'algorithme retourne un plan solution. Le plan retourné à chaque invocation récursive est appelé plan partiel car c'est une partie du plan solution qui sera retourné par l'algorithme ;

- **la recherche par chaînage arrière :** l'idée est de partir du but et d'appliquer l'inverse des opérateurs de planification pour produire des sous-buts jusqu'`a arriver à l'état initial. Si l'algorithme permet de remonter à cet état initial, alors il retourne le plan solution trouvé ;
- **l'algorithme STRIPS :** le principal inconvénient des algorithmes de planification dans un espace d'états est la taille de l'espace de recherche. L'algorithme STRIPS fonctionne de la même façon que l'algorithme de recherche par recherche arrière, mais son avantage réside dans sa capacité à réduire l'espace de recherche. Cette optimisation est caractérisée par les deux points suivants :
  - à chaque appel récursif de l'algorithme, les sous-buts qui doivent être satisfaits sont les préconditions du dernier opérateur ajouté dans le plan, ce qui a comme conséquence de réduire substantiellement le facteur de branchement de l'algorithme;
  - si l'état courant satisfait toutes les préconditions d'un opérateur, alors STRIPS « marque » l'opérateur. Ainsi, en cas d'échec, le retour arrière se fera à partir de cet opérateur.

2.1.4.2 Planification dans un espace de plans.

Les algorithmes de planification dans un espace de plans travaillent sur un espace de recherche plus sophistiqué : les nœuds représentent des plans partiellement spécifiés et les arcs sont des opérations de raffinement de plans, i.e., qui permettent de réaliser un but ouvert ou d'enlever une inconsistance potentielle (par exemple lorsque deux actions sont en conflit). Une autre différence fondamentale concerne la définition du plan solution : la planification dans un espace de plans utilise une structure de plans plus générale que la séquence. Un plan est défini comme étant un ensemble d'opérateurs reliés par des contraintes d'ordre et des contraintes d'instanciation. Une contrainte d'ordre est, par exemple, « l'action A doit s'exécuter avant l'action B ». Un exemple de contraintes d'instanciation pourrait être « le robot r qui intervient dans l'action A est le même que celui qui intervient dans l'action B ». Mais les contraintes d'ordre ne sont pas suffisantes. En effet, rien n'indique que les effets produits par l'action A

soient toujours valables lors de l'exécution de l'action B. Par exemple, une action C s'exécutant parallèlement peut modifier l'état courant, ainsi les préconditions nécessaires `a l'exécution de l'action B peuvent ne plus être disponibles. Pour garantir le plan, il est nécessaire d'ajouter des liens causaux, c'est-à-dire des liens qui définissent les relations entre deux actions en terme de producteur/consommateur. Ainsi les prédicats de l'état courant qui ont été produits par l'action A ne pourront être consommés que par l'action B.

Afin de limiter la complexité, la recherche dans un espace de plans peut être guidée par des heuristiques. Une des heuristiques les plus efficaces est de décomposer les tâches à effectuer en sous-tâches : c'est la planification hiérarchique.

2.1.5 Planification hiérarchique

La planification hiérarchique (Hierarchical Task Network, HTN) **[42]** utilise un langage similaire au langage de la planification classique. En effet, chaque état du monde est représenté par un ensemble de prédicats instanciés et chaque action correspond à une transition déterministe entre deux états. Cependant la planification hiérarchique diffère de la planification classique sur la fa¸con de planifier.

Dans un planificateur HTN, l'objectif n'est pas d'atteindre un ensemble de buts (exemple : porte ouverte) mais de résoudre un ensemble de tâches (exemple : ouvrir porte). Les entrées du planificateur sont, comme en planification classique, un ensemble d'opérateurs, mais également, un ensemble de méthodes. Une méthode décrit la manière de décomposer une tâche en un ensemble de sous-tâches. Le principe de la planification HTN est de décomposer les tâches non primitives en sous-tâches de plus en plus petites, jusqu'`a obtenir des tâches primitives qui peuvent être résolues en utilisant les opérateurs de planification.

2.1.6. Graphes de planification.

Un graphe de planification contient une séquence de niveaux correspondant aux instants discrets dans le plan, où le niveau 0 indique l'état initial. Chaque niveau contient un ensemble de littéraux qui peuvent être vrais à un instant, en fonction des actions exécutées aux instants précédents. Grosso modo, ce sont également toutes les actions qui pourraient présenter des préconditions satisfaites à l'instant considéré, en fonction des littéraux présents**[45]**.Nous employons le terme « grosso modo », car le graphe n'enregistre qu'un sous-ensemble restreint

d'interactions négatives possibles parmi les actions. Par conséquent, il peut être optimiste à propos du nombre minimum d'instant requis pour qu'un littérale devienne vraie. Cependant, ce nombre d'étapes du graphe de planification fournit une bonne estimation de difficulté à accomplir un littérale donnée à partir de l'état initial. Plus important encore, le graphe de planification est défini de façon à être construit très efficacement.

Les graphes de planifications ne fonctionnent que pour les problèmes de planification propositionnels- sans variables. Les représentations STRIPS et ADL peuvent être traduites sous forme propositionnelle. Pour les problèmes mettant en jeu un grand nombre d'objets, cela peut prendre exponentiel le nombre de shémas. Malgré tout, les graphes de planification se sont avérés très efficaces pour résoudre des problèmes de planification complexes.

Nous allons illustrer la notion de graphe de planification par un exemple simple(des exemples plus complexes donnerait des graphes trop volumineux pour tenir sur une page.) la ***figure 2.7*** représente un problème et la ***figure 2.8*** son graphe de planification. On commence au niveau d'état S_0, qui représente l'état initial du problème. On continue avec le niveau d'action $A_{0,}$ dans lequel on place toutes les actions dont les préconditions sont satisfaites au niveau précèdent. Chaque action est reliée à ses préconditions dans S_0 et ses effets dans S_1, ici en introduisant dans S_1 de nouveaux littéraux qui ne sont pas dans S_0.

Init(Avoir(Gâteau))
But (Avoir(Gâteau) $\wedge$ Mangé(Gâteau))
Action(Mangé(Gâteau))
PRECOND : Avoir(Gâteau)
EFFECT : $\neg$ Avoir(Gâteau) $\wedge$ Mangé(Gâteau)
Action(FaireCuire(Gâteau))
PECOND : $\neg$ Avoir(Gâteau)
EFFECT : Avoir(Gâteau)

Fig.2.7 le problème « avoir et manger un gâteau »

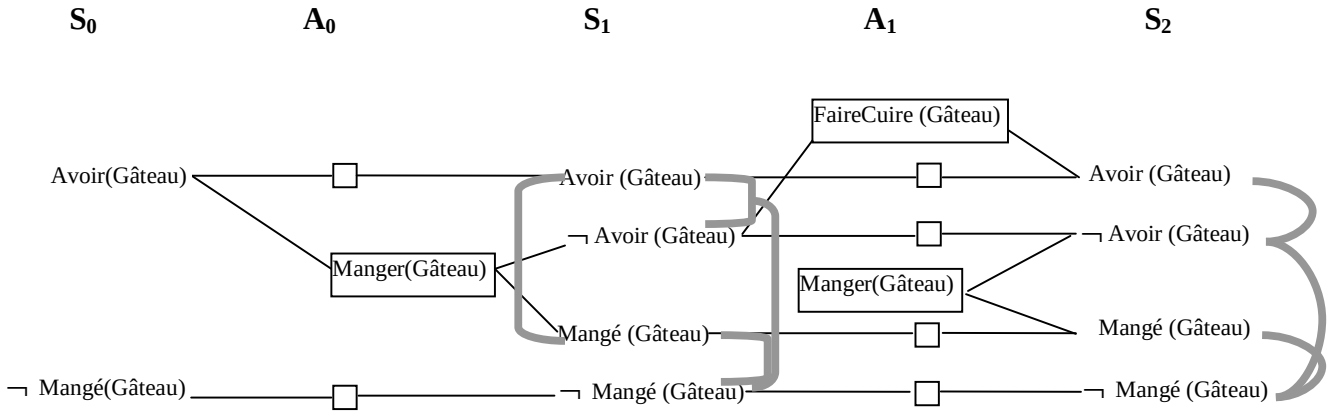


Fig.2.8 Graphe de planification pour le problème « avoir et manger un gâteau »

Dan la figure 2.8, et jusqu'au niveau S_2. les rectangles représentent les actions (les petits carrés indiquent les actions persistantes) et les lignes droites représentent les préconditions et les effets. Les lignes grises incurvées indiquent les liens d'exclusion appelés mutex.

2.1.6.1 L'algorithme GRAPHPLAN.

Cette section explique comment extraire un plan directement du graphe de planification, au lieu d'utiliser simplement le graphe pour obtenir une heuristique. L'algorithme GRAPHPLAN (voir ***figure2.9***) présente deux étapes principales, qui se succèdent alternativement à l'intérieur d'une boucle. Tout d'abord, il vérifie si tous les littéraux buts sont présent dans le niveau courant, sans lien mutex entre chaque pair de littéraux possible. Si c'est le cas, alors une solution pourrait exister à l'intérieur su graphe courant .l'algorithme essaie donc d'extraire cette solution. Sinon, il étend le graphe en ajoutant les actions pour le niveau courant et les littéraux d'état pour le niveau suivant. Le processus se poursuit jusqu'à ce qu'une solution soit trouvée, ou bien que l'on découvre qu'aucune solution n'existe.

```
Fonction GRAPHPLAN(problem) retourne solution ou échec
        Graphe← INITIAL-PLANNING-GRAPH(problème)
        Buts←GOALS[problème]
        répéter
                si les buts sont tous non-mutex au dernier niveau du graphe alors
                 solution←EXTRACT-SOLUTION(graphe,buts,LENGTH(graphe))
                   si solution ≠ échec alors retourner solution
                     sinon si NO-SOLUTION-POSSIBLE(graph) alors retourner échec
        graphe← EXPAND-GRAPH(graphe,problème)
```

Fig.2.9 L'algorithme GRAPHPLAN.

GRAPHPLAN passe successivement d'une étape d'extraction de solution à une étape d'expansion du graphe. ECTRACT-SOLUTION regarde si un plan peut âtre trouvé, en partant de la fin et en effectuant une exploration arrière. EXPAND-GRAPH ajoute les actions pour le niveau courant et les littéraux d'état pour le niveau suivant.

Conclusion

Dans ce chapitre, nous avons abordé les principes de la planification classique et hiérarchique, le langage utilisé ainsi que les principaux algorithmes, que ce soit dans un espace d'états ou un espace de plans. En intelligence artificielle, la planification suscite actuellement beaucoup d'intérêt. Un planificateur peut être vu soit comme un programme qui recherche une solution, soit comme un programme qui démontre (de façon constructive) l'existence d'une solution. GRAPHPLAN est l'un des meilleurs algorithmes de planification permettant de construire un graphe de planification de manière incrémentale, en partant de l'état initial. Chaque couche contient un sur-ensemble de tous les littéraux ou actions pouvant se produire à un instant donné. Chaque couche encode aussi les relations d'exclusion mutuelle, ou **mutex,** entre les littéraux et les actions qui ne peuvent pas se produire simultanément. Les graphes de planification produisent des heuristiques très utiles aux planificateurs en espace d'état ou ordre partiel. Ils peuvent également être utilisées directement dans l'algorithme GRAPHPLAN. L'algorithme GRAPHPLAN parcourt le graphe de planification, en utilisant une exploration arrière pour extraire un plan. Il autorise certains ordonnancements partiels des actions.

Chapitre III

Etat de l'art sur la composition automatique de services web

Table des matières :

Introduction :

L'objectif principal de l'approche service web est de transformer le web en un dispositif distribué de calcul ou les programmes (services) peuvent interagir d'une manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes.

Ce chapitre débutera tout d'abord par la classification du problème de composition de services web. Ensuite, on présentera un modèle abstrait pour la composition automatique de services web ainsi que la présentation de quelques travaux portant sur la composition automatique et on terminera sur les perspectives de ces approches.

3.1 Classification du problème de composition de services web

(SEOG-CHAN OH & all) dans **[20]** donnent une classification du problème de composition de services web (WSC[1] problem) selon trois facettes :

3.1.1 Composition manuelle vs. Automatique (Manual vs. Automatic Workflow Composition)

La composition de services web peut se faire soit :(1) ***manuellement*** par le biais des experts de domaine, ou (2) ***automatiquement*** via l'utilisation des outils de composition logiciels. Dans l'approche manuelle les utilisateurs se charge de sélectionner les services web ensuite de mettre en ouvre le processus de composition, sachant que ces utilisateurs doivent connaître le domaine (puits de domaines (par exemple : domaine d'ontologie)). En générale les utilisateurs sont soutenus avec des interfaces graphiques **(GUI[2])** pour faciliter la composition, mais malgré l'utilisation de ces interfaces, la composition manuelle demande beaucoup d'efforts pour assurer une composition efficace qui est inappropriée dans le cas ou le problème de composition de services web est un problème avec un échelle large (Large-scale WSC problem). En revanche, l'approche de composition automatique utilise des agents logiciels qui sont chargés de mettre la connexion entre les différents services web impliqués dans le problème de composition, sachant que ces agents ont la capacité de savoir si deux services peuvent se connecter ou non via l'utilisation des correspondances

[1] Web Service Composition
[2] Graphique User Interface

syntaxiques ou sémantiques entre les paramètres des services web (Syntactic matching or semantic matching).

3.1.2 Simples Vs. Complexes Opérateurs (Simple vs. Complex Operator)

Un problème de composition de services web utilise seulement des opérateurs séquentiels de composition (AND composition), (par exemple rechercher des données du service web a_1, ET ensuite du service web b_5, ET du service c7…etc.). Mais un problème de composition de services web plus complexe, utilise autres opérateurs que le AND (par exemple OR, XOR, NOT) ou bien des contraintes (par exemple : une requête d'un utilisateur ***r*** préfère les services web dans l'Afrique que celles de l'Europe) et cela selon l'un ou les deux modes de composition parallèle et séquentiel.

3.1.3 Petit vs. Large échelle (Small vs. Large Scale)

En générale un problème de composition de services web peut se réduit en un problème de satisfiabilité (appelé problème *NP-Complet*). Et comme il est difficile ou presque impossible de trouver un algorithme polynomiale pour résoudre un problème de composition de service web, un algorithme de recherche exhaustive est utilisé que pour les problèmes de petite échelle. En revanche pour les problèmes en large échelle, des algorithmes approximatifs qui trouvent des solutions plus ou moins optimales sont désirées

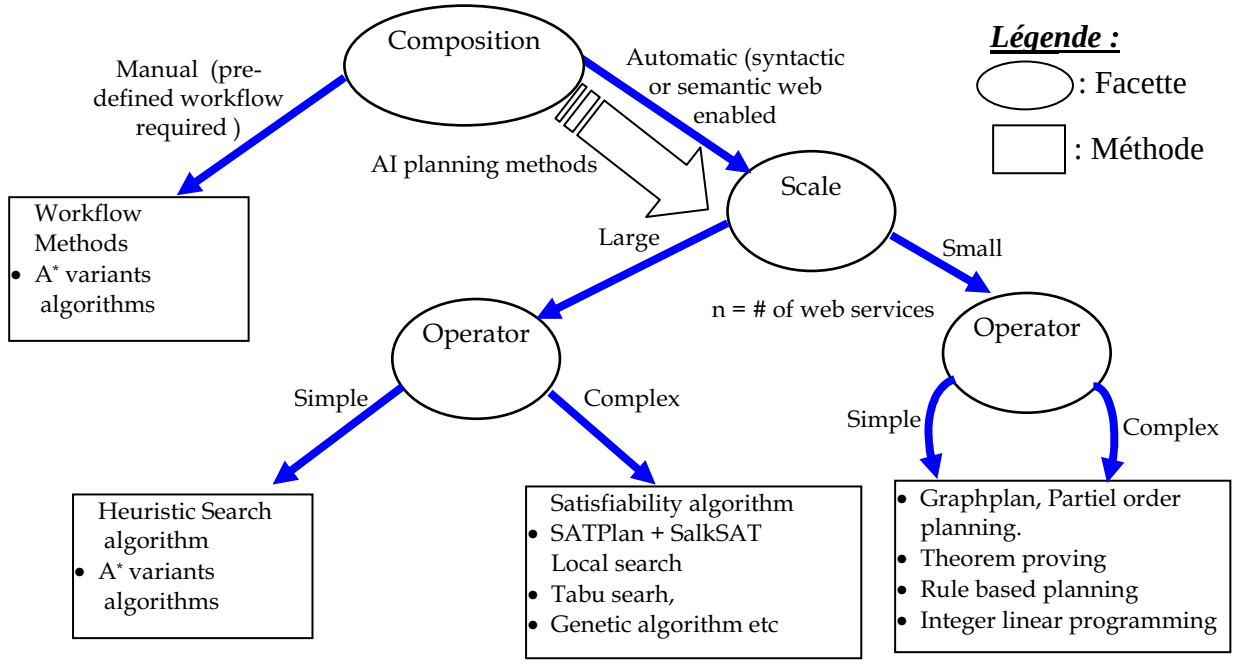


Fig.3.1 : Un arbre de décision des solutions de AI pour le problème de WSC

La ***figure.3.1*** (inspiré de **[20])** illustre un arbre de décision pour nous aider à mieux sélectionner et utiliser des méthodes selon les trois facettes mentionnées ci-dessus. L'approche manuelle de composition peut se fonder sur des experts en matière de programmes et de domaine de logiciel pour lier des déroulements des opérations générées manuellement aux ressources concrètes correspondantes. Par conséquent, cette approche n'est pas appropriée pour des scénarios où on doit composer le déroulement des opérations réel des milliers de services web disponibles. METEOR-S, Proteus, et Kepler sont des exemples de cette approche.

Quand la complexité du problème de WSC augmente, la composition automatique devient plus souhaitable .L'approche automatique de composition peut être complémentaire à l'approche manuelle tels que quelques déroulements des opérations faisables produits de l'approche automatique alternativement sont présentés aux experts en matière de domaine qui peuvent choisir l'un d'entre eux, et a raffiner plus loin manuellement .En particulier, quand des opérateurs complexes tels que la négation n'est pas priés dans la composition, des algorithmes sub-optimaux heuristiques tels que A* sont utilisées.

Sur les autres mains, quand les opérateurs sont complexes et quelques règles spécifiques de contrainte doivent être vérifiées, les systèmes experts basés sur les règles peuvent fonctionner bien. Cependant, vu la croissance rapide des services web, établir une pleine base de connaissance en convertissant tous les services web en axiomes, sera cher, SWORD est un exemple de cette approche.

Cependant, pour un problème plus général de WSC, souvent, les techniques de planification de l'intelligence artificielle sont des solutions performantes telles que STRIPS ou Graphplan, ou les méthodes de programmation linéaires de nombre entier fonctionnent mieux. Par exemple, STRIPS est le premier système principal de planification de AI à décrire les actions en termes de leurs conditions préalables et effets. Le Graphplan est un planificateur d'usage universel et plus générale que STRIPS utilisant des algorithmes de la théorie de graphe. Graphplan emploie une recherche en arrière pour extraire un plan et tient compte de commande partielle parmi des actions .Comme approche de satisfiabilité pour les problèmes de planification, l'algorithme de SATPlan est une méthode locale de recherche qui traduit un problème de planification en axiomes propositionnels et trouve un modèle qui correspond à un plan valide

3.2 Un modèle abstrait pour la composition automatique de services Web :

Dans ce que suit nous allons vous présenter un modèle générale (general framework) pour la composition automatique de services web. Cet framework est sous une forme abstraite à haute niveau qui ne prendre pas en considération des langages particuliers, plateformes ou méthodes utilisées dans le processus de composition.

Plusieurs modèles sont présentés parmi lesquelles en prendre le modèle de (**J.Rao et X.Su [21],[22])**, dans ce modèle générale **(*figure.3.2*)**, le système de composition automatique de services web à deux participants qui sont le fournisseur de service (***Provider***) et l'utilisateur ou le consommateur de service (***Service requester***).le fournisseur de service propose les services web pour utilisation et le consommateur de service utilise le service proposé par le fournisseur de service. Le système contient aussi les composants suivants : Translateur (***Translator***), générateur de processus (***Process Generator***), évaluateur (***Evaluator***) et un moteur d'exécution et découverte de service (***Execution engine and service repository***).

Le *translateur* est un médiateur qui se charge de faire la translation entre les langages utilisés par les participants (Fournisseurs et utilisateurs de services) et le langage interne utilisé par le générateur de processus. Pour chaque requête fourni par l'utilisateur le *générateur* de processus maître ou génère des plans pour composer les services disponibles dans l'annuaire de services afin de répondre à la requête d'utilisateur. Si le générateur de processus trouve plusieurs plans permettant la résolution de la requête de l'utilisateur, l'*évaluateur* se charge de choisir le meilleur plan pour l'exécution et finalement le *moteur d'exécution et de découverte* exécute le plan choisi et retourne le résultat d'exécution à l'utilisateur.

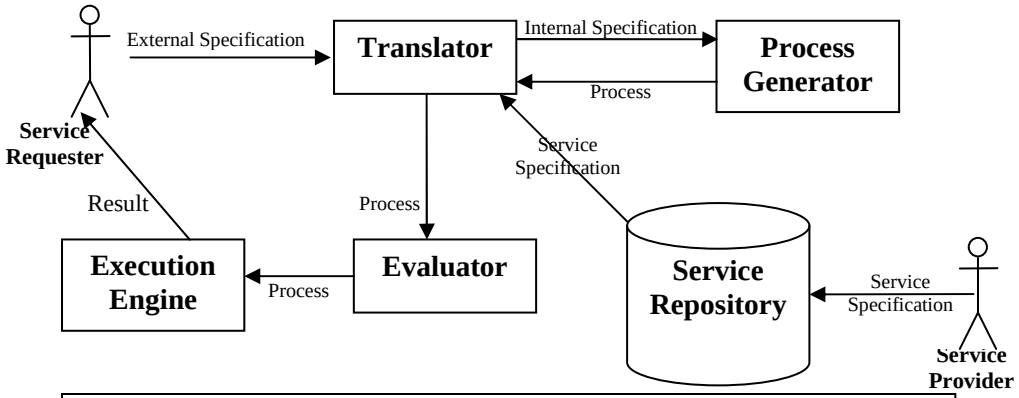


Fig.3.2 : Modèle abstrait pour la composition automatique de services Web

Plus précisément, le processus de composition automatique de services Web contient les phases suivantes :

3.2.1 Présentation du service

Premièrement les fournisseurs de services publient leurs services atomiques dans un emplacement global centralisé. Pour publier un service Web (***Advertise***) Il y o na plusieurs langages universels permettant la publication des services web , **UDDI**, **DAML-S [23]** sont des exemples de ces langages **.** Pour bien présenter et publier un service web, le service doit contenir les attributs essentiels suivants ***Signature***, ***états*** et les ***valeurs non fonctionnelles***. La signature est représenté par les entrées, sorties et exceptions des services Web (Inputs, Outputs and exceptions), ils fournirent des informations sur les informations transmis durant le processus d'exécution d'un service web. Les états sont spécifiés par les pré-conditions et post-conditions qui modélisent la transformation du service web d'un état à un autre état. Les valeurs non fonctionnelles sont les attributs qui sont utilisées pour l'évaluation d'un service à savoir la description du coût, qualité et temps de réponse pour un service web.

3.2.2 Translation des langages

Plusieurs méthodes de composition de service différentient entre les langages de spécification interne et externe du service web. Les langages externes sont utilisés pour faciliter l'accès aux utilisateurs dans le sens ou ces utilisateurs peuvent exprimés comment ou quels sont les services utilisés d'une manière relativement simple. Les langages externes de spécification de services Web sont souvent différents des langages internes utilisés par le processus générateur de composition de services, car le processus générateur de composition à besoin des langages formels et précis comme par exemple les langages de programmation logiques (Logical Programming Languages). En revanche de ceux des langages externes utilisées par les utilisateurs qui sont généralement des langages standards (par exemple : WSDL, DAML-S..). Donc les composants de la translation entre les langages externes de spécification de service web et les langages internes utilisés dans le processus de composition automatique de services web sont devenus obligatoires à développés.

3.2.3 Génération du modèle de processus de composition

L'utilisateur ou le consommateur (Service Repuester) de services peut exprimer son besoin à l'aide d'un langage de spécification de services. Le processus générateur se charge de résoudre la requête fournie par l'utilisateur par la découverte et composition de services atomiques fournis par les fournisseurs. Le processus générateur prendre les fonctionnalités de services comme entrées et génère le modèle du processus (process model) qui décrit le service composite. Le modèle du processus contient un ensemble de services atomiques sélectionnés d'une façon permettant un échange de flots de contrôles et flots de données entre ces services.

3.2.4 Evaluation du service composite

Parfois en trouve plusieurs services qui ont des fonctionnalités similaires. Donc il est possible que le processus générateur peut générer plusieurs plans de service composite solvant les besoins d'un utilisateur (requête utilisateur). Dans ce cas le module évaluateur des services composites engendrés se charge de sélectionner le meilleur plan du service composite en utilisant généralement les informations et attributs non fonctionnels des services web.

3.2.5 Exécution du service composite

Après la sélection unique du meilleur plan du processus composite de services, le service composite est prés pour être exécuté. L'exécution du service composite suit une séquence de message définit par le modèle du processus de service composite. Le flux de données du service composite est définit par l'échange des actions ou de données entre un service et un autre service impliqué dans le processus de composition, cet échange est traduit par le transfert des sorties (outputs) d'un service au entrées (inputs) d'un autre service web.

Dans ce que suit, nous donnerons un aperçu sur les méthodes employées pour le générateur de processus de composition pour produire du processus service composite. Les méthodes peuvent être entièrement automatisées ou semi-automatiques.

3.3 Composition de services web via l'utilisation des techniques de WorkFlow.

Dans les méthodes et les techniques de WorkFlow de composition, nous devrions distinguer la génération statique et dynamique de déroulement des opérations dans le processus de composition. L'approche statique signifie que le demandeur ou l'utilisateur devrait établir

un modèle de processus abstrait avant que la planification de composition commence. Le modèle de processus abstrait inclut un ensemble de tâches et de leur dépendance de données. Chaque tâche contient une requête qui est employée pour rechercher le vrai service web atomique pour accomplir la tâche. Dans ce cas, seulement le choix et l'attachement des services web atomiques est fait automatiquement par programme. La méthode statique le plus généralement utilisée est d'indiquer le modèle de processus sous un graphe. D'autre part, la composition dynamique crée le modèle de processus et choisit des services atomiques automatiquement. Ceci exige du demandeur d'indiquer plusieurs contraintes, y compris la dépendance entre services, la préférence de l'utilisateur et ainsi de suite.

3.4 Composition de services web via AI planning :

Plusieurs recherches ont étés proposées pour résoudre le problème de composition automatique de services web via l'utilisation des techniques d'intelligence artificielle et plus précisément les méthodes de planification du l'intelligence artificielle (***AI Planning***). En générale un problème de planification est décrit comme suit : problème de cinque-uplets **(S,S_0,G,A,T)** , tel que **S** est l'ensemble de tous les états possibles du système, $\mathbf{S_0} \subset \mathbf{S}$ est *l'état initiale* du système , $\mathbf{G} \subset \mathbf{S}$ est *l'état finale* (***Goal***) pour lequel cette état est recherché par le système de planification. **A** est l'ensemble *d'actions* effectuées par le planificateur (***Planner***) permettant le changement entre les différents états du système, et la relation (***fonction***) de translation $\mathbf{T} \subseteq \mathbf{S}\times\mathbf{A}\times\mathbf{S}$ qui définit les *préconditions* et *effets* pour l'exécution de chaque action.

Dans le terme de services web, $\mathbf{S_0}$ et **G** sont l'état initial et l'état final (***Goal***) spécifiés dans les requêtes fait par l'utilisateur (Wbe service requesters). **A** est l'ensemble de services web disponibles. **T** dénote la fonction de changement d'état pour chaque service.

DAML-S (appelé OWL-S dans ces versions récentes) est le seul langage de service web qui mettre une connexion direct avec les méthodes de planification (***AI planning***). Le changement d'état est produit par l'exécution de service web qui est spécifié par les propriétés pré-conditions et effets du ServiceProfile dans DAML-S. les pré-conditions présentent les conditions logiques qui doivent être vérifiées à priori avant que le service concerné est exécuté. Les effets sont les résultats de l'exécution de service web concerné.

DAML-S est un langage construit via l'utilisation du **DAML+OIL,** ces derniers langages utilisent la description logique pour la représentation des expressions sémantiques des données. La majorité des méthodes de planification (*AI Planning*) utilisent DAML-S pour la composition automatique de services web.

Dans ce que suit nous allons vous présenter une liste des méthodes de composition automatique de service web basées sur les méthodes de planification de l'intelligence artificielle (*AI Planning*). Ces méthodes et surtout celles de classes 3 et 5 dans le tableau ci-dessous sont fréquemment utilisées et qui peuvent être classifiées selon plusieurs classes et sous classes **[24]**, le tableau suivant montre les différentes catégories des techniques de planification:

Classe	Sous Classe
1. Classical Planning	• State-Space Planning • Plan-Space Planning
2. NeoClassical Planning	• Planning-Graph Techniques • Propositional Satisfiability Techniques • Constraint Satisfaction Techniques
3. Heuristics and Control Strategies	• Domain-Independent Heuristics • Control Ruels in Planning • Hierarchical Task Network Planning • Situation Calculus • Dynamic Logic
4. Planning with Time and ressours (Extension to (neo-)classical planning)	• Temporal planning • Planning whith ressources
5. Planning under Uncertaintly	• Planning Based on Markov Decision Process • Planning Based on Model Cheking • Uncertainty with Neoclassical Techniques
Others Approaches to planning	• Case-based Planning

Tableau.3.1: Classification des techniques de planifications (AI Planning).

3.4.1 Situation Calculas :

(McLLraith et. Al) **[25, 26,27]** adaptent et font une extension au langage Golog pour la construction automatique de services web. Golog est un langage logique de haut niveau, développé par l'université de Toronto pour la spécification et l'exécution d'actions complexes dans un domaine dynamique. Ce langage est à base de Calcul Situationnel ; il été développé à l'origine pour opérer sans prise en compte des contraintes sur les actions. Toutefois, la prise en charge de ses contraintes est indispensable pour les applications web. Pour cela, (McLLraith &. Al), proposent la spécification de Golog. En somme, l'idée est que l'agent (le service web dans notre cas) doit être capable de raisonner. Pour cela, la requête de l'utilisateur et les contraintes sont représentées par la logique de premier ordre des Calculs Situationnels. Les auteurs distinguent deux catégories d'action pour les services web :

- **Les actions primitives (Primitive Action):** représentent les actions capables de modifier le domaine ou la base de connaissance.
- **Les actions complexes (Complex Action) :** sont définies par un ensemble d'actions primitives. La base de connaissance fournit les préconditions logiques et leurs effets par des Calculs situationnels.

Un langage procédural est utilisé pour construire le service composite et un mécanisme de déduction pour les contraintes. Plusieurs interpréteurs ont été mis au point à base du langage Golog. On citera des exemples tel que ECLIPSE, IndiGolog (SWIProlog) qui est un interprète Golog écrit en prolog.

3.4.2 Planning Domain Definition Language (PDDL)

PDDL fait partie des langages reconnus et normalisés dans le domaine de planification. La communauté scientifique travaillant sur les planificateurs a très vite relevé la ressemblance existant entre DAML-S[3] et PDDL dans leurs représentations.

PDDL est destiné à exprimer la physique d'un domaine, c'est-à-dire quels sont les prédicats existants, quelles sont les actions possibles, quelle est la structure des compositions d'actions et quels sont les effets des actions. La résolution d'u problème de composition automatique peut se faire par un mapping de la description DAML-S en PDDL, et utiliser un planificateur déjà existant.

[3] DARPA Agent Markup Language-Service Ontology

Afin que PDDL couvre toutes les particularités apportées par la composition automatique de services web, McDamott introduit un nouveau type de connaissance, appelé *value of an action* **[28]**. Dans la perspective de construire un service composite, il est important de distinguer entre les informations transformées et les états. L'information présentée comme paramètre en entrée et sortie est réutilisable. Ainsi, cette donnée peut être utilisée dans l'exécution de plusieurs services. Par contre, l'état du monde évolue par l'exécution d'un service ; ce changement d'état se traduit par le passage d'un état à un autre.

3.4.3 Planification basée sur les règles (Rule-based Planning)

Medjahed et al, **[29]** présentent une technique basée sur la description déclarative de haut niveau et des ontologies. Cette technique de composition se subdivise en quatre étapes :

1. **phase de spécification** pour cela, on autorise la description de haut niveau de ce que l'on désire composer, en utilisant un langage appelé Composite Service Specification Language (CSSL).
2. **phase d'arrangement (matchmaking)** ou les règles de composabilité permettent de générer un plan conforme aux spécifications de la requête de départ.
3. **phase de sélection** c'est dans cette phase qu'ont sélectionnent les services web allant constitué notre service composite. Aussi, dans le cas ou plusieurs plans sont disponibles, un choix sera effectué selon de critères de qualité (QOC).
4. **phase de génération** la description de notre service composite sera générée selon le plan.

Dans sa forme actuelle, un document WSDL contient les informations sur les types de données qui seront utilisés, le format des messages qui seront échangés, la spécification des opérations qui propose le service,etc. Cependant, une telle description ne contient pas d'informations sur la sémantique des opérations, i.e. quels sont les objectifs d'une opération, à quel domaine elle appartient, etc. les auteurs proposent d'ajouter de telles informations sémantiques à cette description afin de permettre la composition automatique de services web.

L'auteur dans **[30]** dans leur formalisation, un web service se définit par une description (une définition informelle de ce que fait le service) et d'un ensemble d'opérations. Une opération se définit comme dans WSDL par une description, un message en entrée, un

message en sortie, un mode[4]. De plus, trois informations sémantiques sont associées à chaque opération : un *objectif,* une *catégorie* et une *qualité.* Un objectif se définit par un nom de fonction (ce que fait l'opération), un ensemble de synonymes de la fonction et un ensemble de caractéristiques de la fonction. Une catégorie se définit par un nom de domaine (associé à l'opération), un ensemble de synonymes du domaine et un ensemble de caractéristiques du domaine. La qualité est définit par un coût de l'exécution de l'opération, un booléen sécurité permettant de déterminer si l'opération se déroule dans un mode sécurisé, un booléen confidentiel permettant de déterminer si des informations sensibles sont transmises par l'opération. Les auteurs proposent une extension du langage WSDL, le CSSL (Composite Service Specification Language), permettant de décrire les services web dans ce formalisme.

A partir des informations ***syntaxiques*** et ***sémantiques,*** une notion de composabilité des opérations peut être définie. Celle-ci peut être décomposée en la *composabilité syntaxique* et la *composabilité sémantique*. Deux opérations sont syntaxiquement composables si elles sont de mode opposé : si l'une est en mode « sens unique » l'autre doit être en mode « notification » et inversement et si l'une est en mode « requête-réponse » l'autre doit être en mode « demande réponse ». Par ailleurs le message en entrée (resp. en sortie) d'une opération doit être de même type que le message en sortie (resp. en entrée) de l'autre opération. Deux opérations sont sémantiquement composables lorsqu'elles ont des objectifs, catégories et qualités similaires.

3.4.4 Hierarchical Task Network (HTN)

Deux classes principales de planificateurs sont généralement identifiés : ceux (largement majoritaires) qui planifient dans l'espace d'état, et ceux qui planifient dans l'espace de plans partiels. Dans la première catégorie, on retrouve TLplan, TALplan ; ils exploitent des règles de contrôle dans la recherche de plan, en mettant en œuvre des logiques temporelles particulières. On y trouve aussi les planificateurs à base de réseau de taches hiérarchiques (HTN), comme SHOP2, ou les taches sont hiérarchiques explicitement, selon un shèma de décomposition explicitement fourni dans le domaine de planification.

SHOP2 est un système de planification indépendant du domaine. Les planificateurs HTN exploitent une définition préalable de moyens de décomposer des taches de haut niveau

[4] Sens unique, notification, demande de réponse, requête-réponse

(dites méthodes) en sous taches plus élémentaires, jusqu'à parvenir aux taches les plus élémentaires appelées opérateurs. Un plan est alors donné par un ensemble d'opérateurs partiellement ordonnés. Les planificateurs HTN s'opposent aux planificateurs dans lesquels la structure du domaine et les dépendances entre taches ne sont pas données à priori au système : le planificateur exploite lors des opérateurs disponibles pour évoluer dans l'élaboration d'un plan, en effectuant la recherche soit dans l'espace des plans partiels, soit dans l'espace des états.

Sirin et Al, **[31]** proposent une technique de composition semi automatique. Le principe est que, chaque fois que l'utilisateur sélectionne un service ; tous les services compatibles avec ce dernier soient mise en avant et présenté à l'utilisateur. Cette capacité de détection des compatibilités est réalisée à travers une technique de raffinement de la sélection de manière graduelle. A ce sujet, la sélection ce fait aussi bien a base d'attributs fonctionnel que non fonctionnels ; vue que les fonctionnalités (paramètres) se voit décrit par des classes OWL[5]**.** Un raisonnement est fait grâce à OWL par un moteur d'inférence OWL afin de vérifier la compatibilité des services. Par exemple, la distance sépare deux services n'appartenant à la même feuille dans l'arbre des ontologies. Dans le cas ou deux services sont compatibles, le système filtre ces services à partir des propriétés non fonctionnels spécifié par l'utilisateur ; c'est seulement après que ce service sera présenté au prestataire. Cette technique s'avère particulièrement intéressante. En effet, la difficulté de la composition automatique vient de la difficulté à comprendre le comportement que l'on veut obtenir du service composite.

3.4.5 Démonstration automatique de théorème (Theorem proving)

Waldinger et al **[32]** élabore une idée pour la synthèse de service en utilisant la technique de preuve de théorème. Initialement les services disponibles et la requête du client sont décrits dans un langage de premier ordre et les preuves constructives sont générées avec la preuve du *théorème Snarck,* les descriptions de la composition de service sont extraites depuis des preuves particulières.

Lammermann et al**,[33]** proposent l'utilisation de **SSP**[6], pour la composition automatique. SSP est une approche déductive pour l'assemblage de services par des spécifications. La spécification du service inclus uniquement les propriétés structurelles pour distinguer les

[5] Web Ontology Language
[6] Structural Synthesis Program

paramètres d'entrée/sortie et la logique propositionnelle intuitive afin de résoudre les éventuels problèmes de composition.

Ruo et Al **[34,35]** ont introduit une méthode pour la composition automatique de services web sémantiques en utilisant la *Linear logic theorem proving.* La méthode utilise le langage de service web sémantique (DAML-S) pour la représentation externe de services web. Intérieurement, les services sont représentés par les axiomes de l'extralogical et les preuves dans la logique linéaire.

Conclusion :

Dans ce chapitre nous avons présenté les différentes techniques et méthodes utilisées dans la composition automatique de services web. Premièrement, nous avons commencé avec une classification du problème de composition de services web ensuite de donner les cinq étapes essentielles utilisées dans le processus de composition automatique. Le processus de composition automatique est un ensemble de processus qui sont : processus de **Présentation et description,** processus de **translation,** processus de **génération,** processus d'**évaluation** et le processus d'**exécution.** Chaque étape utilise différents langages, plateformes et méthodes.

Dans ces cinq étapes, nous nous concentrons sur les méthodes de génération de processus de composition de services web. Les méthodes sont soit de type déroulement des opérations (Workflow) ou techniques de planification du domaine d'intelligence artificielle (AI). Les méthodes de déroulement des opérations sont la plupart du temps employées dans la situation où la demande a déjà défini le modèle de processus, mais le programme automatique est exigé pour trouver les services atomiques pour remplir la condition. Les méthodes de planification de AI est employées quand le demandeur n'a aucun modèle de processus mais a un ensemble de contraintes et de préférences. Par conséquent le modèle de processus peut être produit automatiquement par le programme de composition de services.

Chapitre IV

Mise en place d'un algorithme de composition automatique de services web

Table des matières :

4.1 Introduction

La composition de services web représente une étape fondatrice de l'évolution de ce paradigme. La compréhension des propriétés fondamentales de la composition des services web est nécessaire afin de tirer profit de ce paradigme. L'identification du niveau d'abstraction adéquat pour l'expression de cette composition permettre en particulier de séparer clairement les niveaux technologiques (XML, WSDL, UDDI, etc.) des niveaux conceptuels (langages de haut niveau). La composition automatique de services web a pour but la réduction de la complexité dans la création et la mise en place de services web. L'automatisation est donc un concept clé qui doit être présent à chaque étape du processus de conception et de déploiement de services web. Elle est essentielle car elle permet d'intégrer les facteurs aussi importants que le passage à l'échelle, la réduction des coûts et prendre en compte de critères de qualité, etc. A cette fin, plusieurs groupes de recherches tentent de définir les fondements de base nécessaire à une composition automatique.

Parmi les approches émergentes, on citera (Seog-chan oh & al) **[20].** Ils présentent une technique de composition basée sur l'utilisation des techniques de planification du domaine d'intelligence artificiel. Ainsi, leurs travaux ont donnée lieu à la réalisation d'un prototype (WSPR[1]). Ce dernier permet de déterminer s'il est possible de générer à partir d'un groupe de services sous un échelle très important (c'est-à-dire un grand nombre de services web) un nouveau service (composite) répondant à des besoins prédéfinis à l'avance. Dans le cas ou une composition est possible une liste de services web sollicitant les besoins du demandeur est renvoyée.

Le problème sur laquelle porte ce travail est d'appliquer une combinaison de techniques de planification et plus précisément l'algorithme GraphPlan avec des algorithmes fédérateurs permettant la recherche de plus court chemin dans un graphe et plus précisément l'algorithme de Bellman Ford, sur le problème de composition de services. Cet algorithme va nous améliorer le processus de découverte de l'ensemble de services web impliqué dans la composition. Donc l'algorithme va générer un plan d'exécution sous forme d'une liste

[1] Web Service PlanneR Algorithme

ordonnée de services web qui répond aux besoins du demandeur de service avec une plus haute qualité de service (temps d'exécution très court).

Dans ce que suit, on essaye de donner un exemple motivant la compréhension du problème ainsi que les différents définitions du problème de composition de services web qui sont généralement inspirées du travaux de (Seog-chan oh & al)**[20]**

4.2 Exemple

Pour utiliser un service web dans un réseau qui est soit de grande taille comme par exemple Internet ou un réseau de petite taille d'une entreprise. Un programme d'un client localise d'abord le serveur de services web qui peut satisfaire certaines demandes à partir d'une page jaune (UDDI), et obtient des spécifications détaillées (WSDL) du service web sollicité. Puis, on utilisant ces spécifications, le client envoie une demande au service web considéré par l'intermédiaire d'un protocole standard de message et communication (SOAP), et dans le retour, il reçoit une réponse de service.

En particulier, un problème d'intérêt pratique concerne les deux issues suivantes. Supposant une requête ***r***, parmi des milliers de services web trouvés dans UDDI :

1- comment trouver les services web qui satisfont la requête ***r***.
2- comment composer des services multiples à satisfaire ***r*** quand il n'existe pas un service web tout seul satisfaire la demande de l'utilisateur.

Pour mieux comprendre le problème, considérant les quatre services web du **tableau.4.1**, comme illustré avec la notation WSDL :

- Etant donnée *l'hôtel*, la *ville* et *l'information de l'état*, le service FindHotel retourne l'adresse et le code postal de l'hôtel.
- Etant donnée le *code postal* et la *nourriture préférée*, le service FindRestaurant retourne le nom, le numéro de téléphone et l'adresse du restaurant, ces derniers satisfont la nourriture préférée et l'hôtel le plus proche au code postal donné.
- Etant donnée la *référence courante d'endroit* et de *nourriture*, le service GuideRetaurant retourne l'adresse du restaurant le plus étroit et de son estimation qui satisfaire l'endroit et la nourriture donnée.

- Etant donnée les adresses *source* et *destination*, le service FindDirection retourne en détail avec une image de carte du direction de l'adresse source jusqu'à l'adresse de destination.

```
<message name='findHotel_Request'>
      <part name='hotel' type='xs:string'>
      <part name='city' type='xs:string'>
      <part name='state' type='xs:string'>
</message>
<message name='findHotel_Response'>
       <part name='addr' type='xs:string'>
       <part name='zip' type='xs:string'>
</message>
```

(a) FinfHotel

```
<message name='findRestaurant_Request'>
         <part name='zip' type='xs:string'>
         <part name='foodPref' type='xs:string'>
</message> <message name='findRestaurant_Response'>
         <part name='name' type='xs:string'>
         <part name='phone' type='xs:string'>
         <part name='addr' type='xs:string'>
</message>
```

(b) findRestaurant

```
<message name='guideRestaurant_Request'>
            <part name='foodPref' type='xs:string'>
            <part name='currAddr' type='xs:string'>
</message> <message name='guideRestaurant_Response'>
            <part name='rating' type='xs:string'>
            <part name='destAddr' type='xs:string'>
</message>
```

(c) guideRestaurant

```
<message name='findDirection_Request'>
           <part name='fromAddr' type='xs:string'>
           <part name='toAddr' type='xs:string'>
</message>
<message name='findDirection_Response'>
           <part name='map' type='xs:string'>
           <part name='direction' type='xs:string'>
</message>
```

(d) findDirection

Tableau 4.1 : Exemple de services web

Maintenant, on considère les deux requêtes suivantes de « State College, PA , USA » :

r_1: trouver l'adresse de l'hôtel ''Atherton'', et

r_2 : trouver un restaurant de Thaï prés de l'hôtel ''Atherton'' avec une direction.

Pour accomplir r_1, il suffit d'appeler le service FindHotel. C'est-à-dire appelant FindHotel avec les paramètres de r_1 : FindHotel(''Atherton'',''State College'', ''PA''). Le service retourne l'adresse de l'hôtel comme : ''100 Atherton Street'' avec le code postal de ''16801''. Cependant, aucun des quatre services web ne peut seul satisfaire r_2.les deux services web, FindRestaurent et GuideRestaurant peuvent trouvés un restaurant de Thaï prés de l'hôtel désiré. Mais ne peuvent pas fournissent un direction. D'autre part, le service web FindDirection peut donner une direction d'un endroit à un autre, mais ne peut pas localiser aucun restaurant. Par conséquent, on doit employer un ensemble de services web pour satisfaire entièrement la requête r_2. Il y a deux méthodes possibles pour effectuer cette tâche : Après l'obtention de l'adresse de l'hôtel en utilisant FindHotel, on peut suivre l'un ou l'autre de ces deux assertions :

1. appeler GuideRetaurant (''Thai'', ''100 Atherton, 16801, PA'') pour obtenir l'adresse du restaurant le plus étroit, par exemple ''410 S.Allen St, 16802, PA'' puis, appeler le service web FindDirection(''100 Atherton Street, 16801, PA'',''410 S.Allen St, 16802, PA'') pour obtenir un direction.
2. Appeler FindRestaurant (''16801'',''Thaï'') pour obtenir l'adresse du restaurant le plus étroit, par exemple ''410 S.Allen St, 16802, PA''. Puis appeler le service web FindDirection (''100 Atherton Street, 16801, PA'',''410 S.Allen St, 16802, PA'') pour obtenir une direction.

4.3 Définition du problème

Un service web w a en général deux ensembles de paramètres :$\mathbf{w^i}$ ={I_1,I_2,I_3,…..} Pour les requêtes SOAP (comme entrées) et $\mathbf{w^o}$={O_1,O_2,O_3,…..} pour les réponses SOAP (comme sorties).Quand w est appelé avec tous les paramètres d'entrée $\mathbf{w^i}$, il renvoie les paramètres de sortie $\mathbf{w^o}$ .Nous supposons que pour appeler w , tous les paramètres d'entrée $\mathbf{w^i}$ doivent être fournis.

Définition.5.1 (Problème de découverte de service web): On suppose qu'une requête ***r*** a des paramètres initiaux d'entrée r_i et des paramètres de sortie r_o .Le problème de la découverte de service web (WSD[2]) à pour solution de trouver un ensemble de services web, tel que :

(1): $r_i \supseteq w^i$ et

(2): $r_o \subseteq w^o$.

Il est apparaît clairement que la résolution du problème de découverte de services web est peut être facilement résolu s'il existe au moins un et un seul service web vérifiant les paramètres d'entré et de sortie de la requête fournie par l'utilisateur tel que les conditions mentionnées dans la définition ci-dessus sont vérifiées. Par conséquent, dans le reste de ce document, nous nous concentrons sur le cas où aucun service web satisfont entièrement la requête ***r*** de l'utilisateur, et donc on doit composer de multiples services web pour répondre à la requête du demandeur. Ce type de problème désigné sous le nom du problème de composition de services web (WSC[3]).

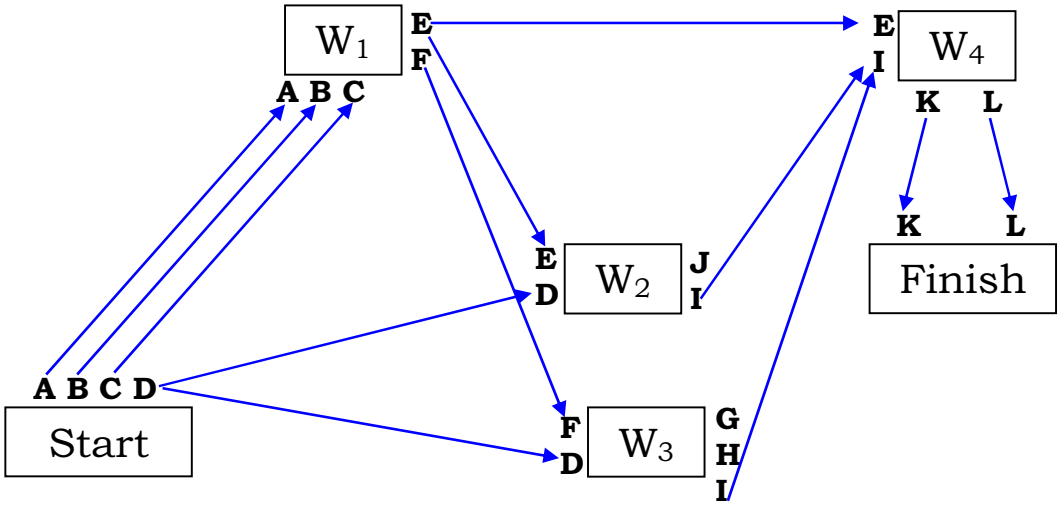
Comme nous avons le vu dans le chapitre précédent, il en existe plusieurs méthodes et techniques pour résoudre ce genre de problème, parmi ces techniques est les plus répondus sont les techniques de planification.

Par exemple le problème de composition de services web peut être formellement défini en utilisant un problème de planification dans un modèle de STRIPS[4] du 4- tuple : $\prod=(P,W,r^i,r^o)$, ou :

- (1)- P est un ensemble de paramètres, dans l'exemple donnée ci-dessus, p= {''Hotel-name'',''Hotel-City'',''Hotel-Adress'',…….}.
- (2)- W est un ensemble de services web, dans l'exemple donnée précédemment, W= { ''FindHotel'',''FindRestaurant'',…..}
- (3)- $r^i \subseteq$ P, est les paramètres d'entré initiales.
- (4)- $r^o \subseteq$ P , est les paramètres de sotie désirés.

[2] Web-Sevice Discovery Problem
[3] Web-Service Composition Problem.
[4] STRIPS est le premier système principal de planification de l'intelligence artificielle qui représente des actions en terme de leurs préconditions et effets.

La ***figure.4.1***, illustre l'exemple donné précédemment sous un modèle de planification STRIPS :



W : ensemble de services Web	
FindHotel	W_1
FindRetaurent	W_2
GuideRetaurant	W_3
FindDirection	W_4

P : ensemble des paramètres			
HotelName	A	RetaurantName	G
HotelCity	B	RestaurantPhoneNumber	H
HotelState	C	RetaurantAdress	I
FoodPreference	D	RestaurantRate	J
HotelAdress	E	MapHotelAdress	K
HotelZipCode	F	DirectionHotelRestaurant	L

Figure.4.2 : Modèle STRIPS de l'exemple

Définition.5.2 (problème de composition de services web) : supposant une requête ***r*** a des paramètres d'entrée initiales r^i, et des paramètres de sortie désirés r^o. La résolution du problème de composition de services web revient à trouver une séquence finie de services web,

$w_1, w_2, w_3, \ldots\ldots, w_n$, tel que :

(1)- w_i est invoqué séquentiellement de 1 à n.

(2)- $(r^i \cup w_1^o \cup w_2^o \cup \ldots\ldots \cup w_n^o) \supseteq r^o$.

(3)- le coût total $\sum_{i=1}^{n} c(w_i)$, est réduit au minimum.

4.4 Conditions nécessaires pour l'établissement d'un outil idéal de composition automatique de services web.

Pour que la composition automatique de services web soit efficace les propriétés suivantes doivent être vérifiées **[43]**:

4.4.1 Exactitude (Correctness)

Un des plus importantes propriétés nécessaire pour assurer que l'outil chargeant de mettre la composition automatique de service web est que ce dernier (outil de composition) doit vérifier des résultats corrects. C'est-à-dire la composition de services satisfaire tous les besoins de l'utilisateur définis par la requête. Donc le processus de composition est capable de trouver tous les services vérifiant les besoins de la requête.

4.4.2 Un petit temps d'exécution de la requête (Small Query Execution Time)

Le temps de réponse de l'exécution de la requête qui interroge l'annuaire de services doit être raisonnable et acceptable (exemple : un temps de réponse en quelques millisecondes).

4.4.3 Changements incrémental (Incremental Updates)

L'ajout ou la mise à jours des services web dans les annuaires de services prend un temps de mise a jours raisonnable. Un bon outil de composition automatique de services web doit prendre en considération ces changement en mettre a jours les données nécessaires des ces nouvelles changements dans son base de données de ces services mais pas tous les données de cette base de données. C'est-à-dire l'ajout des données que pour ces nouveaux services.

4.4.4 Fonction des coûts (Cost Function)

S'il y on a des coûts associées pour chaque service dans l'annuaire de services. Un bon outil de composition automatique de services web doit rendre des résultats basés sur les paramètres des coûts pour ces services. Ces paramètres peut être vus comme un vecteur d'attributs de coûts pour chaque service, et l'outil de composition chargé de rendre les résultats des requêtes fournis par l'utilisateur, applique des fonctions de minimisation et maximisation sur ces vecteurs d'attributs.

4.5 Description sémantique de services web

Un service web est une application ou programme conçu pour assurer une interopérabilité entre machines ou systèmes .Il a une interface qui est décrite dans une format de sorte que d'autres services web ou systèmes puissent agir avec se service web sans l'intervention de l'utilisateur.

Donc les tâches automatiques de découverte, composition, …etc. d'un services web, peut avoir lieu efficacement seulement si les descriptions sémantiques formelles des services web sont disponibles. Actuellement, il y a un certain nombre d'approches pour décrire la sémantique de Services web tels que **DALM-S [11]. OWL-S [14], SWDL-S [16], WSMO [17].**

Et comme Le problème de composition de services web doit intégrer l'information des sources hétérogènes. Puisque différents services web sont créés en isolation, leurs vocabulaires ont souvent des problèmes avec des abréviations, des formats différents, ou des erreurs typographiques .En outre, deux termes avec des appellations différentes peuvent avoir la même signification sémantique, (exemple ''price'' et ''fee'')

Réciproquement, deux termes avec la même appellation peuvent avoir des significations différentes (par exemple, le titre peut signifier le titre de livre ou le titre du travail).

En réponse à ces défis, les chercheurs ont développés des divers schémas de correspondances entre données (matching schemes) .on considère x et y sont des objets de données (par exemple, paramètres de services web ; champ record d'individu) avec un vecteur des attributs : $x = (x_1 ; x_2 ,…., x_k)$, et $y = (y_1 , y_2 ,….., y_k)$, où k est le nombre d'attributs. Nous pouvons mesurer la similarité ''Similarity'' entre x et y par une fonction de distance, d(x, y) avec des propriétés suivantes **[20]** :

- (1)- d(x,y) >= 0 ; où l'égalité tient si et seulement si x =y ,
- (2)- d(x,y) = d(y,x) , la propriété symétrique
- (3)- d(x,y) <= d(x,z) + d(z,y) , l'inégalité de triangle

Donc en fonction de l'utilisation de fonction de distance, on obtient différentes similarités entre données. En générale trois approches de similarité entre données peuvent être utilisées pour définir la similitude entre différentes données :

1. **Approche-1 :** similarité exacte, en utilisant l'équivalence syntaxique (c'est la cas dans notre approche de composition automatique de services web utilisée -***paragraphe 4.6.***).
2. **Approche-2 :** similarité approximative, en utilisant des fonctions de distance.
3. **Approche-3 :** similarité sémantique en utilisant les ontologies (par exemple RDF, OWL,....etc.)

Dans l'Approach-1, deux objets x et y sont considérés similaires si et seulement si x= y. Cependant, avec cette approche, deux objets avec des représentations légèrement différentes (Par exemple, ''William Jefferson Clinton''et ''Bill Clinton'') ne peut pas être similaires. Pour cette raison, dans l'Approach-2, si deux objets sont assez semblables selon certains fonction de distance (c.-à-d., d (x,y) est au-dessus d'un certain seuil), ils sont considérés pour être similaires. Bien que l'approach-2 soit beaucoup flexible que l'approach-1, elle n'est pas assez suffisant d'identifier que le ''price "et le''fee'' sont interchangeables. Dans la réponse, les chercheurs ont créés la vision de web sémantique, où les données ont la structure et les ontologies décrit la sémantique des données. On se basant sur le web sémantique, l'Approache3 peut adresser le problème de correspondance d'ontologies (ontology-matching) pour trouver le mapping sémantique entre deux ontologies, indiqués par des langages comme **DAML+OIL, OWL et RDF**.

4.6 Nouvelle approche de composition automatique de services web

Notre approche est basée sur l'utilisation mixte de deux algorithmes, le premier algorithme est la première phase de GraphPlan (Forward Search) qui se charge de déterminer l'ensemble de services web impliqués dans le processus de composition. Ces services sont représentés sous forme de graphe. Le deuxième algorithme et l'algorithme de Bellman Ford permettant de donner une composition optimale via la génération de plus court chemin de services web dans un graphe orienté de services web généré dans la première étape de l'algorithme. Donc Le but de cet algorithme est de trouver une composition qui produit les sorties désirées (outputs) dont

le nombre de services web impliqué dans la composition est le minimum possible et le temps d'exécution de l'ensemble de services web du service composite est le plus court possible.

La ***figure.4.3*** suivante présente la stratégie de notre approche.

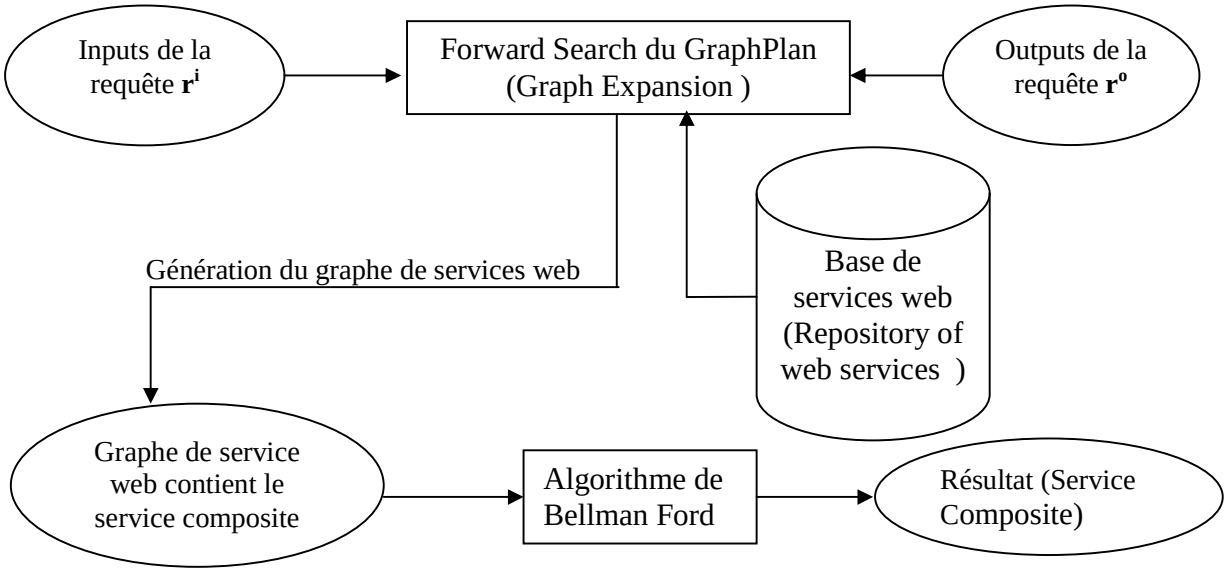


Figure.4.3. : composition automatique de services web basée sur GraphPlan et Bellman Ford

Dans la figure ci-dessous il apparaît que nous avons modélisé le problème de composition automatique de services web comme étant un problème de recherche dans une base de services web, car le problème de composition de services web revient à trouver un ***service composite*** que se compose de plusieurs services, s'il n'existe pas un seul service qui va répondre à la requête de l'utilisateur. Notant que le problème de découverte d'un service web est un cas particulier du problème de composition de services web, car l'ensemble généré par le module (algorithme) de composition dans ce cas est un seul service web. Pour cela et dans notre cas nous avons utilisé une recherche en avant (forward search) de l'algorithme GraphPlan qui va générer un graphe orienté de services web, ensuite lancer un algorithme d'optimisation (Bellman Ford) permettant de trouver une séquence de services web la plus optimale.

4.6.1 Représentation formel d'un service web, requête et annuaire de service

Avant de commencer de donner la structure de cet algorithme une représentation formelle des concepts suivants doit être modélisés : Service web, Annuaire de services web et Requête de l'utilisateur.

Définitions :

1. ***Service Web :*** Un service web est un 2-uplet, S= (*I, O*), tel que :
   I : les paramètres d'entrée de service web (Inputs), et
   O : les paramètres de sortie de service web (Outputs).
2. ***Annuaire de services*** : ensemble de services web (base de données de services web)
3. ***Requête*** : une requête est définie comme suit : Q = (*QI* , *QO*) ; tel que
   QI : la liste des paramètres d'entrée de la requête (inputs).
   QO : la liste des paramètres de sortie de la requête (outputs).

4.6.2 Algorithme de composition

Pour obtenir une composition efficace d'un ensemble de services web, l'algorithme doit suivre les deux étapes suivantes :

Etape1 : Graph expansion du GraphPlan : trouver l'ensemble des services web qui satisfont aux paramètres d'entrée et de sortie de la requête :

Démarche :

Prendre les paramètres d'entrée de la requête et produit une liste de services web qui répondent au paramètres de sortie de la requête. L'algorithme compare les paramètres d'entrée et de sortie de la requête avec celles des paramètres d'entrée et de sortie des services web, deux cas possibles :

(1) s'il existe un service web qui vérifié la propriété suivante

$\{\text{Requête}_{\text{inputs}} \supseteq \text{WebService}^{\text{inputs}}$ et $\text{Requête}_{\text{outputs}} \subseteq \text{WebService}^{\text{outputs}}\}$, alors le résultat de composition dans se cas est un seul service web.

(2) s'il n'existe pas un service web vérifiant la propriété donnée ci-dessus, l'algorithme va suivre la démarche suivante :

Démarche :

Phase(1). L'algorithme utilise les paramètres d'entrée (inputs), et de sortie (outputs) de la requête. Ensuite il parcourir la liste de services web disponibles afin d'obtenir tous les services qui acceptent ou moins un paramètre d'entrée qui se trouve dans la liste des paramètres d'entrées de la requête (c'est-à-dire, les paramètres d'entrée par un service web doivent être un sous ensemble de l'ensemble des paramètres d'entrée de la requête)

phase (2). On répète la phase précédente *Phase(1)* pour la nouvelle liste de paramètres d'entrées qui est dans ce cas la liste initiale des paramètres d'entrée de la requête union tous les paramètres de sortie des services web trouvées dans la *Phase(1)* . La liste des services trouvés dans cette phase et celles qui répondent à cette nouvelle liste de paramètres en respectant les mêmes conditions données dans la *Phase (1).*

La phase (2), répété autant de fois jusqu'au les paramètres de sortie de la requête sont trouvés, et la liste des services est celles de la première phase union la nouvelle liste des services web dans la deuxième phase. La ***figure 4.3*** montre le principe de ce processus.

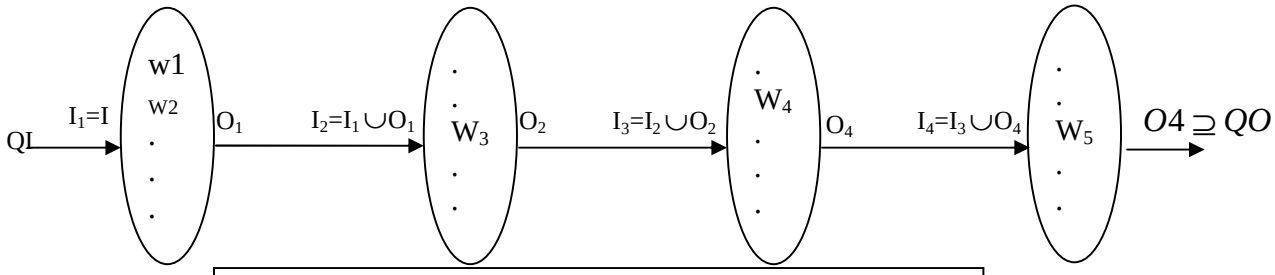


Fig.4.4 technique de génération du service composite

Etape 2 : Construction du graphe de services web et génération du service composite.

1. Après la génération de l'ensemble de services web dans l'étape précédente, il est lieu de construire le graphe de services web. Ce dernier est construit en mettre touts les liaisons entre les différents services de l'ensemble de services générés dans *l'étape1* en respectant la règle suivante :

On met un lien orienté du service web W^1 vers le service web W^2, si $W_2^{Inputs} \subseteq W_1^{Outputs}$, par exemple la ***figure.4.5*** présente le graphe construit à partir de la liste de services web de l'exemple donnée au début de ce chapitre.

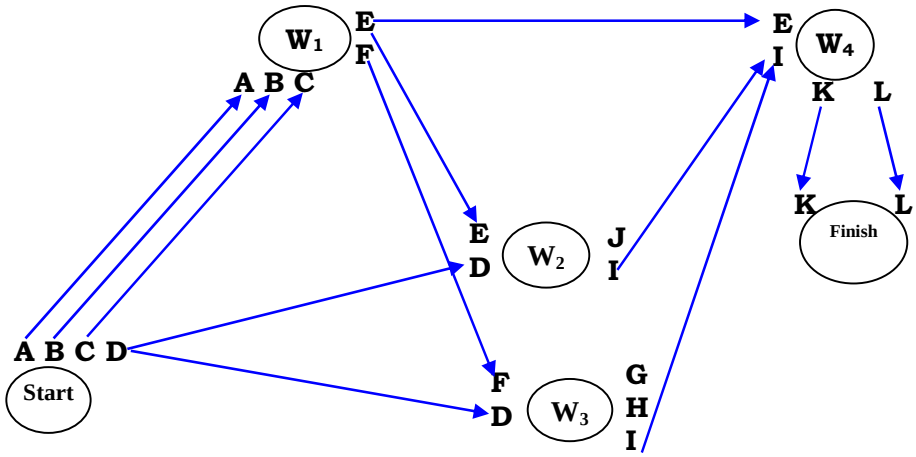


Fig.4.5 Graphe de services web

2. À partir du graphe construit en génère la liste des services web du service composite, qui correspond au plus court chemin dans un graphe orienté. Par exemple dans la ***figure.4.5***, le service composite est soit $\{W_1 \rightarrow W_2 \rightarrow W_4\}$ ou $\{W_1 \rightarrow W_3 \rightarrow W_4\}$. Dans notre cas nous avons utilisé l'algorithme de Bellman Ford.

4.6.3. Présentation algorithmique de l'algorithme.

En effet l'algorithme est composé de plusieurs étapes, l'étape de découverte de l'ensemble de service web du service composite via l'utilisation de Forward Search du GraphPlan, ensuite la construction du graphe de composition et finalement le lancement du Bellman Ford pour trouver la chaîne ordonnée de services web la plus optimale.

Algorithme Composition ;

Variables : Q (QI, QO), // Requête Q , avec QI , QO : respectivement Inputs et Outputs de la requête.

P// ensemble des paramètres d'entrés et de sorties, initialement p égale à QI ;

S(I,O),//Service Web avec I,O : inputs et Outputs de service web

Ws,// annuaire de services web : Ws={S_1,S_2,S_3,……S_n}.

S_{comp}// Service Composite, ensemble de services impliqués dans le processus de compsotion .

Début

// lancement du Forward Seach du GraphPlan.

$S_{comp} \leftarrow \emptyset$;

Pour tous $S_i \subseteq$ Ws **faire** //i={1,2,…..n} ;

Si $S_{inputs} \subseteq$ P alors

$P \leftarrow P \subseteq S_{outputs}$;

$S_{comp} \leftarrow S_{comp} \cup S_i$;

Fin si

Fin pour

La phase ***graph expansion,*** du GraphPlan pour la découverte de l'ensemble de services web.

Si $S_{comp} = \emptyset$ Alors

Ecrire ('Il n'existe pas un ensemble de services web qui répond à la requête de l'utilisateur') ;

Sinon

Si nombre(S_{comp}) = 1 **Alors**

//nombre (S_{comp}) ; indique le nombre de services dans l'ensemble S_{comp} ;

Ecrire ('Il existe un seul service web qui répond à la requête de l'utilisateur ') ;

Sinon // nombre(S_{comp}) >= 2 ;

Construction du graphe de composition de services et lancement de Bellman Ford

Lancer_Bellman_Ford(Construirte_Graphe (S_{comp})) ;

{Commentaire :**(1)-** Construire_Graphe(S_{comp}) est une fonction retourne un Graphe orienté, cette fonction construit un graphe orienté à partir de S_{comp} sachant que deux nœuds sont ajoutés à l'ensemble S_{comp} , le nœud Start et le nœud Finish. Tel que Start à pour inputs l'ensemble vide et pour outputs l'ensemble QI, et Finish à pour inputs l'ensemble QO et pour outputs l'ensemble vide. La construction du graphe orienté suit la règle suivante : On met un lien orienté du service web S_1 vers le service web S_2, si $S_2^{Inputs} \subseteq S_1^{Outputs}$. Tel que S_1, S_2 sont deux services web appartiennent à l'ensemble S_{comp}. **(2)-** Lancer_Bellman_Ford (Grpahe) est une fonction qui recherche le plus court chemin dans un graphe orienté, afin de retourner l'ensemble ordonné de services web impliqué dans le processus de composition}.

Fin Si

Fin Si

Fin

4.7 Mise en ouvre et testes.

4.7.1 Les données d'entrées et de tests.

Dans ce qui suit nous allons vous présenter le fonctionnement générale de notre programme (application) :

Au départ nous avons besoin de deux entrées :

1. une collection ou ensemble de fichiers WSDL à lire (répertoire ou repository de services web).
2. Un fichier XML (composition.xml), ou sont spécifiées les données désirées, c'est-à-dire :
   - les entrées (inputs de départ) ou paramètres d'entrés de la requête utilisateur.
   - Les sorties (outputs) données d'on désir obtenir ou paramètres de sorties de la requête.

Dans notre interface graphique ***(figure.4.6.)*** ; les champs permettant de recevoir les deux entrées citées précédemment sont :

1- Premier champ de texte a coté du premier bouton Rechercher : Qui désigne le chemin de fichier XML contenant les données d'entrées et de sorties de la requête utilisateur.

2- Deuxième champ de texte a coté du deuxième bouton Rechercher : Qui désigne le chemin de la collection de fichiers WSDL ou se trouvent l'entrepôt de fichiers WSDL.

L'application ou le programme se charge de lire les données d'entrées qui sont se forme de fichiers (XML et WSDL) qui nécessitant l'utilisation de la Bibliothèque WSDL4J, cette dernière nous permet de retirer les différents informations et structures (opérations, ….) du fichiers WSDL.

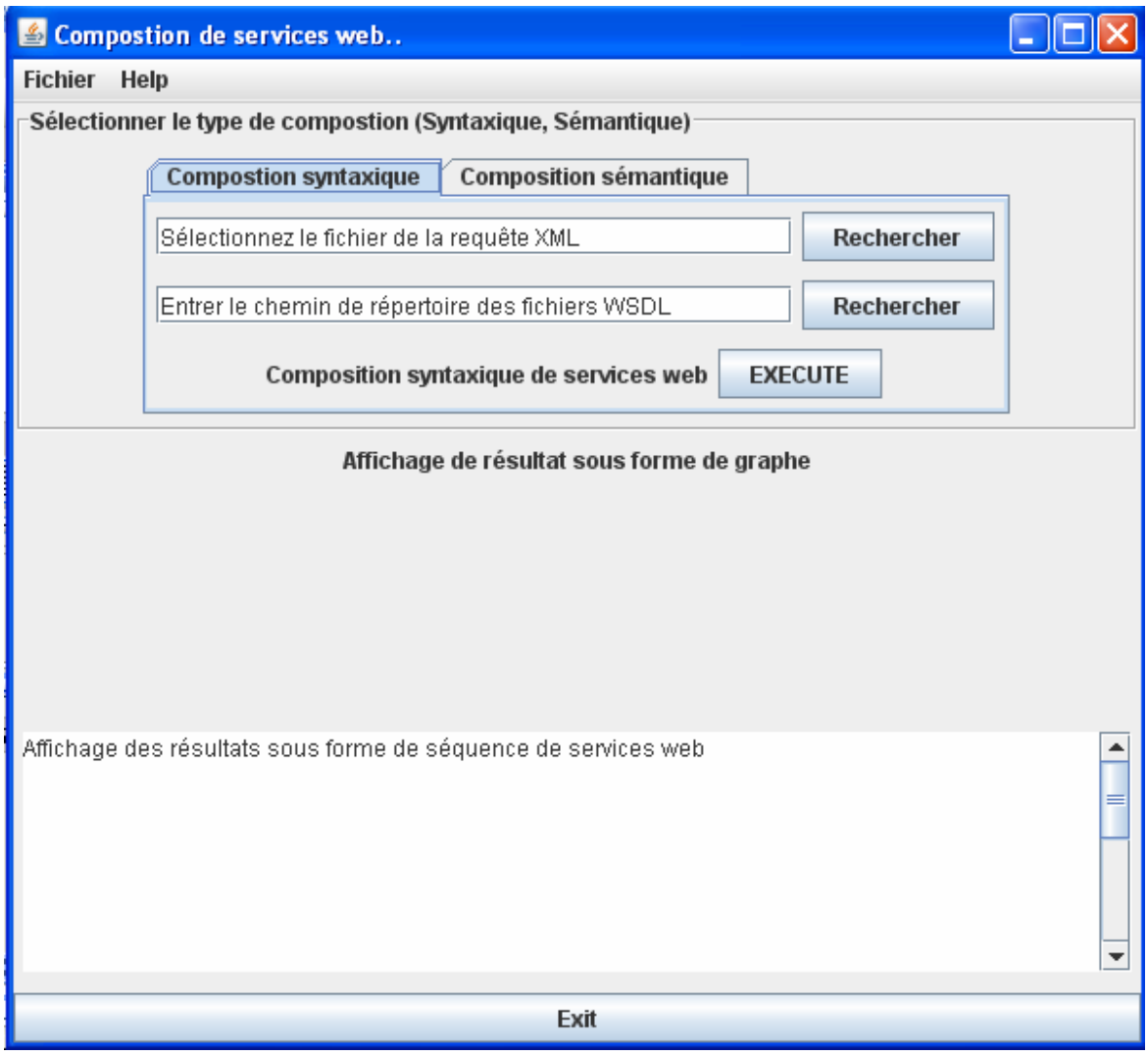


Fig.4.6 Interface Graphique de réception des entrées et d'affichage de résultat

La figure suivante nous montre ce déroulement ***(figure.4.7.)*** :

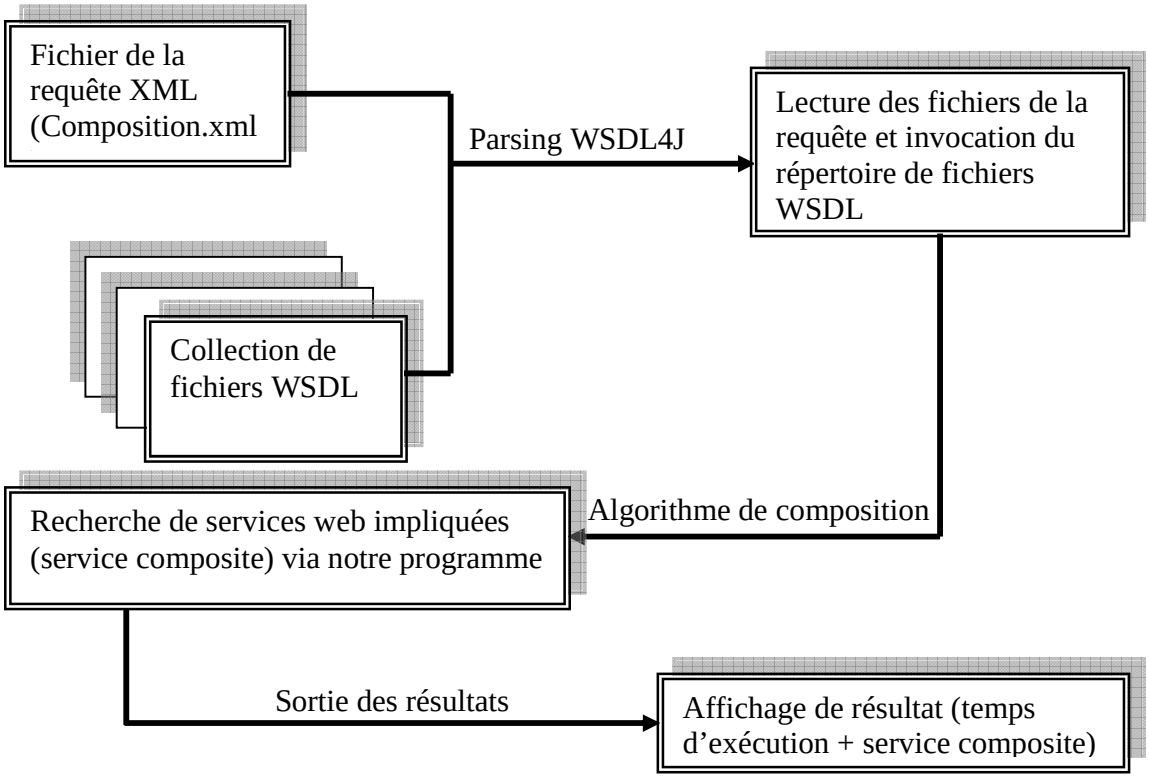


Fig.4.7. flux de traitement de notre application

4.7.2. Mise en ouvre de l'algorithme.

Avant d'aborder l'implémentation de notre approche, il est convient d'abord de présenter les bibliothèques utilisées pour la mise en ouvre du programme.

4.7.2.1 WSDL4J

Introduction

Nous avons utilisé WSDL4J pour beaucoup de raisons. D'abord, c'est une bibliothèque ouverte (open source) développée par IBM en langage Java. Ensuite, elle permet l'accès aux éléments d'un document WSDL par catégorie ou par nom, ce qui est assez convenable et pratique pour notre application.

Par ailleurs, WSDL4J permet la création, la représentation et la manipulation des documents WSDL qui décrivent des services web (WSDL4J). Cet outil, au lieu de parser manuellement les documents et accéder à leurs informations en traversant l'arbre créé par le parser, accède

directement aux méthodes et aux paramètres et retourne toutes les valeurs et les informations contenues dans un document WSDL.

Installation

D'abord, il faut télécharger le package se trouvant à l'URL suivante: http://www124. ibm. com/developerworks/projects/wsdI4j/ et installer les archives disponibles sur l'ordinateur. Ces archives contiennent dans le répertoire lib deux jars: wsld4j.jar et qname.jar, qu'il faut rajouter au classpath (WSDL4J).

Exemple d'utilisation de WSDL4J

Dans l'exemple suivant, la bibliothèque WSDL4J est utilisée pour obtenir une liste de toutes les méthodes ou opérations contenues dans le document WSDL du service web.

```
public void parseFile( String f )
        {
                try
                {
                        WSDLReaderImpl wsdlreader = new WSDLReaderImpl();
                        wsdlreader.setFeature("javax.wsdl.verbose",false);
                        Definition def = wsdlreader.readWSDL(null, new InputSource(f));
                        Map portTypes = def.getPortTypes();
                        Iterator portTypeIterator = portTypes.values().iterator();
                        while (portTypeIterator.hasNext())
                        {
                                PortType portType = (PortType)portTypeIterator.next();
                                setServiceName(portType.getQName().getLocalPart());
                                if (!portType.isUndefined())
                                {
                                        List operations = portType.getOperations();
                                        for (int i = 0; i < operations.size(); i++)
                                        {
                                                Operation o = (Operation)operations.get(i);
                                                addInputMessage(o.getInput().getMessage());
                                                addOutputMessage(o.getOutput().getMessage());
                                        }
                                }
                        }
                        Map messages = def.getMessages();
                        Iterator msgIterator = messages.values().iterator();
```

```
                while (msgIterator.hasNext())
                {
                        Message msg = (Message)msgIterator.next();
                        if (!msg.isUndefined())
                        {
                                Map parts = msg.getParts();
                                Iterator prtIt = parts.values().iterator();
                                while (prtIt.hasNext())
                                {
                                        Part prt = (Part)prtIt.next();
                                        String prtN = prt.getName();
                                        if (outputMessages.contains(msg))
                                                addOutput(prtN);
                                        else
                                                addInput(prtN);

                                }
                        }
                }
        } catch(Exception e)
        {
                e.printStackTrace();
        }
}
```

Fig.4.7.a. Exemple d'analyse d'un fichier WSDL via l'utilisation de WSDL4J

4.7.2.2 Fichiers testés.

Pour faire nos tests sur des données réels (fichiers WSDL) nous avons utilisées ceux du *Web Service challenge*. Sur ce site plusieurs entrepôts (repository) de fichiers sont donnés. Dans notre cas nous avons choisi l'entrepôt (EEE05 repository). Cet entrepôt contient 100 services web. Et pour les requêtes de tests, un fichier de requêtes ainsi que leurs solutions sont données sous forme d'un fichier pdf ***(figure 4.7.b et figure 4.7.c)*** afin de comparer nos résultats sur l'entrepôt à travers l'utilisation de notre algorithme.

Pour le modèle de fichier de la ***figure 4.7.c*** les inputs de la requête sont les paramètres encadrés par les balises *(provided)* et les outputs de la requêtes ou les sorties escomptés sont les paramètres encadrés par les balises *(resultant).*

Composition

1. purchaseALT.wsdl->reserveRental.wsdl-> reserveRoom.wsdl->createItinerary.wsdl
2. findMostRelevantStocks->getStockPriceMany->performStockResearch->purchaseOptimalStock
3. findMostRelevantStock->getStockPrice->purchaseStock
4. getStockPrice->purchaseStock
5. purchaseFlowers
6. findCloseAttorney->scheduleAttorney
7. findCloseLibrary->findISBN->checkOutLibraryBook
8. findCloseVideoStore->rentVideo
9. findThemePark->purchaseThemeParkTicket
10. getForecast
11. findCloseFlorist->purchaseFlowers
12. findDate->scheduleDate->findDateRestaurant->reserveDateRestaurant
13. findCloseMountain->purchaseLiftTicket
14. findCloseVeterinarian->scheduleVeterinarian
15. scheduleShipment

Fig.4.7.b Résultats des requêtes de test pour l'entrepôt EEE05 Repository

```
<CompositionRoutine>
        <Provided> partname1, partname2, partname3
    </Provided>
        <Resultant> partname1, partname2, partname3
    </Resultant>
</CompositionRoutine>
```

Fig.4.7.c format d'un fichier XML pour une requête de test

4.7.2.3 Analyse de résultats :

Pour évaluer la performance de l'approche (programme) utilisé dans la composition automatique de service web, nous avons utilisé les fichiers de tests de site Web Service challenge. Et plus précisément l'entrepôt EEE05.

Pour montrer l'efficacité du programme nous avons utilisée deux critères ou mesures d'évaluation :

- **Le temps d'exécution :** mesuré en milliseconde, afin de nous donner le temps nécessaire pour trouver la solution de la composition (service composite).
- **Le nombre de service web de la solution :** c'est le nombre de services web nécessaire pour trouver la solution cherchée.

Donc la solution la plus performante celle qui nous donne un temps d'exécution très petit avec un nombre de service web plus restreint.

Nous avons réalisé 15 tests qui nous donnent les résultats dans le tableau suivant :

Numéro de la requête utilisée du (EEE05 Test number)	Nombre de services web	Temps d'exécution en milliseconde
1	5	1453
2	5	1215
3	3	1125
4	2	422
5	1	820
6	5	1220
7	3	1125
8	2	1001
9	4	943
10	1	63
11	3	863
12	1	700
13	2	875
14	2	732
15	1	47

Tableau 4.8 : Résultats des test effectués sur l'entrepôt EEE05

4.8 Conclusion

Ce chapitre présente une modélisation du problème de composition de services web, ainsi que le problème de découverte de services web. Après l'utilisation de la formulation de ces problèmes donnés par (Seog-chan oh & al) **[20],** nous avons essayer de donner une approche de résolution du problème de composition automatique de services web basée l'utilisation mixte de deux algorithmes. Le premier algorithme qui est une partie ou bien la première étape de GraphPlan (Forward Search ou GraphExpansion), ce dernier sert à trouver un seul service web ou un ensemble de services web du service composite, et la deuxième étape de l'algorithme est à pour but de construire le graphe de dépendance entre les services web trouvés dans l'étape 1, ainsi que la génération d'une liste ordonnée de services web du service composite, cette étape s'articule sur l'algorithme Bellman Ford afin de donner la solution la plus optimale à travers la génération de plus court chemin de services web du service composite dans un graphe orienté.

Conclusion

Ce mémoire nous a permis de nous confronter au problème de la composition automatique de services web à travers :

- Etude des architectures orientées service : dans lesquelles nous avons étudié le principe de fonctionnement d'un service web et les standards utilisées dans le développement du service web.
- Etude des différentes approches développées dans le domaine de composition automatique de services web. Et surtout celles qui utilisent les techniques de planification en intelligence artificielle.
- Proposition d'une approche basée sur l'utilisation mixte de deux algorithmes, le premier algorithme est la première étape de l'algorithme graphPlan, qui est ***graph expansion*** qui se charge de trouver l'ensemble de services web impliqué dans le processus de composition ainsi que la construction du graphe de composition. Le deuxième algorithme est l'algorithme ***Bellman Ford*** permettant de trouver une séquence ordonnée de services web la plus optimale via la recherche de plus court chemin du service composite dans un graphe orienté de services web.

On termine ce mémoire en évoquant, à titre de perspective un certain nombre d'idées :

- L'implémentation de cet algorithme dans un environnement capable de générer des interfaces utilisateur à la volé afin d'interagir avec des services
- Evaluation de cet algorithme de composition automatique dans un environnement réel.
- Inclure à l'algorithme de composition des propriétés non fonctionnels (Coût, fiabilité, disponibilité, sécurité….etc.), ainsi que l'utilisation d'un langage de description sémantique pour la description sémantique des services web.

Bibliographie

[01] Daniela Berardi, ***Automatic Service Composition.Models, Techniques and Tools***. Thèse de doctorat, University degli Studi di Roma "La Sapienza", italia, 2005.
[02] Papazoglou , Challenges MP, P Traverso, S Dustdar, F Leymann ,***Service oriented computing state of the art and Research*** - COMPUTER, 2007 -doi.ieeecomputersociety.org
[03] http://www.w3.org/2002/ws/arch/2/06/wd-wsa-arch-20020605.html, Rapport, consultée le 12/05/2008.
[04] http://www.w3.org/TR/ws-arch/ , Rapport consulté le 12/05/2008.
[05] Francisco Curbera, Matthew Duftler, Rania Khalaf,William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana ***Unraveling the web services web, an introduction to SOAP, WSDL and UDDI ,*** IEEE Internet computing, 2002 - cat.inist.fr.
[06] http://www.w3.org/TR/soap/ , Rapport consulté le 05/07/2008
[07] http://www.w3.org/2002/ws/desc/, Rapport consulté le 05/07/2008
[08] Jinghai Rao,Xiaomeng Su, ***A Survey of Automated Web Service Composition Methods*** , 2004, Springer
[09] Nerea Arenaza, ***Composition Semi-Automatique de services Web ,*** Projet de Master Février 2006**.**
[10] Philippe Laublet1, Chantal Reynaud2, Jean Charlet3***, Sur Quelques aspect du Web sémantique.*** Actes des deuxièmes assises nationales du GdRI3, 2002 - emse.fr.
[11] DAML-S Coalition: Anupriya Ankolekar2, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katia Sycara ***, DAML-S: Web Service Description for the Semantic Web,*** Proc. First International Semantic Web Conference ISWC, 2002 – Springer.
[12] K. Gottschalk S. Graham H. Kreger J. Snell, ***Introduction to Web Services Architecture.*** IBM Systems Journal, 2002 - research.ibm.com
[13] Richard S. Patterson, John A. Miller, Jorge Cardoso, Mike Davis, ***bringing semantic security to semantic web services***, The Semantic Web Real-world Applications from Industry, 2008, Springer.
[14] David Martin ,Mark Burstein ,Drew McDermott ,Sheila McIlraith ,Srini Narayanan ,Massimo Paolucci ,Bijan Parsia ,Terry Payne Evren ,Sirin Naveen ,Srinivasan Katia ,Sycara Ora Lassila**,** ***OWL-S: Semantic Marckup for Web Services***, diponible sur: http://www.w3.org/Submission/OWL-S/ , Consulté le 10/06/2008.
[15] David Martin, Massimo Paolucci, Sheila McIlraith ,Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou ,Monika Solanki ,Naveen Srinivasan, Katia Sycara, ***Bringing Semantics to Web Services: The OWL-S Approach,*** International Workshop on Semantic Web Services and Web Process Composition (SWSWPC) 2004- Springer.
[16] R.Akkiraju, J.Farrell, J.A.Miller, M.Nagarajan, A.Sheth , K.Verma , ***Web Service Semantics – WSDL-S***, IBM Reshearche Report 2006, disponible sur www.iswc2006.semanticweb.org
[17] Cristina Feier, Dumitru Roman, Axel Polleres, John Domingue, Michael Stollberg, and Dieter Fensel,***Towards Intelligent Web Services: The Web Service Modeling Ontology (WSMO)****, International Conference on Intelligent Computing (ICIC), 2005.
[18] Anura Guruge, ***Web Services: Theory and Practice*** , Digital Press, 2004.
[19] Aphrodite Tsagatidou, Thomi Pilioura, ***An overview of standards and related technologies in web services,*** Distributed and parallel Databasees, 2002- Springer.
[20] Seog-chan oh, Dongwon lee, and Soundar r. t. kumara **,** ***A Comparative Illustration of AI Planning-based Web Services Composition,*** *2006.* Springer

[21] J. Rao and X. Su. ***Toward the composition of Semantic Web services.*** In *Proceedings of the Second International Workshop on Grid and Cooperative Computing, GCC'2003*, volume 3033 of *LNCS*, Shanghai, China, December 2003. Springer-Verlag.
[22] J. Rao and X. Su. ***A survey of automated Web service composition methods***. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC'2004*, LNCS, San Diego, USA, July 2004. Springer-Verlag.
[23] D. Martin et al., DAML-***S(and OWL-S) 0.9 draft release***.
http://www.daml.org/services/daml-s/0.9/, Rapport consulté le 07/08/2008.
[24] Malik Ghallab, Dana Nau, and Paolo Traverso, ***Automated Planning: Theory and Practice***, Amsterdam Morgan Kaufmann Publisher, 2004.
[25] S. McIlraith, T. C. Son, and H. Zeng. ***Semantic Web services***. IEEE Intelligent Systems, 2001.
[26] S. Narayanan and S. McIlraith. ***Simulation, verification and automated composition of Web service***. In Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA, May 2002.
[27] S. McIlraith and T. C. Son. ***Adapting Golog for composition of Semantic Web services***. In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning(KR2002), Toulouse, France, 2002.
[28] Jinghai Rao, ***Semantic web service composition via logic-based program synthesis***. Master's thesis, Norwegian University of Science and Technology, 2004.
[29] Brahime Medjahed, Athman bouguettaya, and Ahmed K.Elmagarmid**. *Composing web services on the semantic web* VLDB,** 2003
[30] Benjamin Habegger. ***Extraction d'informations à partir du web***, 2004.
[31]. E. Sirin, J. Hendler, and B. Parsia**. *Semi-automatic composition of Web services using semantic descriptions,*** In Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003, 2002.
[32] R.Waldinger**. *Web agents cooperating deductively,*** Springer-Verlag, 2001.
[33] S. L¨ammermann**. *Runtime Service Composition via Logic-Based Program Syn-thesis.*** PhD thesis, Department of Microelectronics and Information Technology, Royal Institute of Technology, June 2002.
[34] S. R. Ponnekanti and A. Fox**. *SWORD: A developer toolkit for Web service composition,* In Proceedings of the 11th World Wide Web Conference, Honolulu, HI, USA, 2002.**
[35] J. Rao, P. K¨ungas, and M. Matskin**. *Application of Linear Logic to Web service composition,*** In Proceedings of the 1st International Conference on Web Services, Las Vegas, USA, June 2003.
[36] G.Ernst , A.Newell, , and H.Simon., GPS : ***A case study in generality and problem solving.*** Academic Press, 1969.
[37] C.Green, ***Application of theorem-proving to problem solving***. In Proceedings of the 1st International Joint Conference on Artificial Intelligence ,2001
[38] H.Kautz, , and B.Selman, ***Pushing the envelope : planning, propositional logic, and stochastic search***. In Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference, 2003.
[40] D.McDermott, M.Ghallab, , A. Howe, C.Knoblock, A.Ram, M.Veloso, D.Weld, and D.Wilkins, ***PDDL - the planning domain definition language***. Technical Report CVC TR-98-003/DCS
[41] R.Finke and N.Nilsson, STRIPS : ***A new approach to the application of theorem proving to problem solving***. Artificial Intelligence 2, 1971.
[42] D.Nau, T.Au, O.Ilghami, U.Kuter, W.Murdock, , D.Wu, and Y. Yaman, ***Shop2 : An HTN planning system***. Journal of Artificial Intelligence Research 20, 2003
[39] M.Ghallab, D.Nau, and P. Traverso**, *Automated Planning Theory and Practice.*** May 2004.
[43] Srividya Kona, Ajay Bansal, and Gopal Gupta **, *Semantics-based Efficient Web Service Discovery and Composition* ,** Springer-Verlag , 2006.
[44] Patrick Kellert, Farouk Toumani**, *Les web services sémantiques,****Web sémantique,raport final décembre 2003***

[45] Stuart Russell et Peter Norvig **,** ***Intelligence artificielle, avec prés de 400 exercices***, Pearson Education France, **2006**.

Résumé

Les services web représentent la dernière évolution majeure dans les domaines des technologies web et d'intégration d'applications. La composition de services web représente une étape fondatrice de l'évolution de ce paradigme. La compréhension des propriétés fondamentales de la composition des services web est nécessaire afin de tirer profit de ce paradigme. Parmi les approches émergentes nous citerons les méthodes de composition basées sur l'algorithme **GRAPHPLAN**, qui est un algorithme de planification très puissant dans le domaine d'intelligence artificielle. Notre travail propose l'utilisation mixte de l'algorithme **GRAPHPLAN** avec l'algorithme **Bellman Ford** pour procéder à une composition optimale de services web.

Mots clés : service web, web sémantique, techniques de planification en intelligence artificielle, composition de services web.

Abstract

Web services represent the last major development in the areas of web technologies and application integration. The composition of Web services represents a phase of the evolution of this paradigm. Understanding the fundamental properties of the composition of web services is necessary to take advantage of this paradigm. Among the emerging approaches we mention the methods of composition based on the algorithm **GRAPHPLAN**, a planning algorithm very powerful in the field of artificial intelligence. Our work suggests the mixed use of the algorithm **GRAPHPLAN** with **Bellman Ford** algorithm for an optimal composition of web services.

Keywords : web services, semantic web, techniques in artificial intelligence planning, composition of web services.

ملخـــــص

خدمات الشبكة تمثل آخر التطورات في مجالات تقنيات الإنترنت وتطبيق التكامل. تركيب خدمات الويب يمثل مرحلة من مراحل تطور هذا النموذج. فهم الخصائص الأساسية لتكوين شبكة الإنترنت وخدمات ضرورية للاستفادة من هذا النموذج. بين النهج المستجدة نذكر طرق تكوينها على أساس خوارزمية **GRAPHPLAN** ، تخطيط خوارزمية قوية جدا في مجال الذكاء الاصطناعي. في عملنا نقترح استخدام مدمج لخوارزمية **GRAPHPLAN** مع خوارزمية **Bellman Ford** ودلك من اجل تركيب امثل لخدمات الشبكة.

الكلمات المفتـــــاحية : خدمات الإنترنت ، تقنيات التخطيط في مجال الذكاء الاصطناعي، تركيب خدمـــات الشبكـــــــة.