

Table des matières

Dédicaces

Remerciements

Liste des figures

Liste des tableaux

Notations et Acronymes

Introduction générale.....12

Chapitre I: L'Architecture Orientée Service (SOA) et les services web

I. Introduction.....15

I.2.Définitions15

I.2.1.Définitions métier.....15

I.2.2Définitions technique15

I.3.Principe fondamentale de l'architecture SOA16

I.4.Les avantages et les inconvénients de SOA..... 16

I.4.1.Les avantages16

I.4.2.Les inconvénients.....17

II. Service Web 17

II.1.Historique du Service Web 17

II.2.Définition18

II.3.Cycle de vie des services web..... 18

II.4.Standard des Services Web20

II.4.1. SOAP	20
II.4.2. WSDL	22
II.4.3. UDDI	25
II.5.Caractéristiques des services Web	27
II.6.Les avantages et les inconvénients des services web.....	27
II.6.1.Les avantages	27
II.6.2.Les inconvénients.....	28
III.1. Sélection des Services Web	29
III.2.Qualité des services web.....	30
III.3. Exemple de motivation.....	32
III.4. Spécification du problème de sélection avec QoS déterministe.....	33
IV. Conclusion	34

Chapitre II: Conception et implémentation du prototype

I. Introduction	35
II. Présentation du Corpus.....	35
II.1.Description de base.....	35
II .2.Description de requête.....	35
III. Conception.....	35
III. 1.1. Etape1:Sélection local des services web.....	35
III. 1.2.Les fonctions utilisées.....	38
III. 1. 3. Exemple d'application.....	38

III. 2. 1. Etape2:compositions des services.....	41
III. 2. 2. Les fonctions utilisées.....	42
III. 2. 3. Exemple d'application.....	42
IV. Présentation du prototype.....	45
IV.1.Outils et environnement de développement.....	44
IV.2.Présentation de l'application.....	45
V. Expérimentation.....	48
VI. Conclusion.....	49
Conclusion générale	51
Références bibliographique	
Résumé	

Liste des figures

Figure 1 : Cycle de vie des services web	19
Figure 2 : Structure d'un message SOAP	21
Figure 3 : Exemple de la structure d'un message SOAP	22
Figure 4 : Structure d'un document WSDL.....	23
Figure5 : Exemple de Code WSDL.....	24
Figure 6 : Publication et recherche dans un registre UDDI	25
Figure 7 : Structure de l'information dans un registre UDDI.....	27
Figure 8 : Exemple de composition des services Web « Planification de voyage »	31
Figure 9 : Graphe du classement selon la probabilité dominance.....	40
Figure 10 : Fenêtre d'accueil.....	47
Figure 11 : Lancement des compositions de service.....	48
Figure 12:Les meilleures compositions.....	48

Liste des tableaux

Tableau 1: Les fonctions d'agrégation des QoS	32
Tableau 2 : Le score de probabilité dominance des services web	39
Tableau 3 : Le degré de chaque service web.....	40
Tableau 4 : La variation de QoS des services web	42
Tableau 5: La Médiane pour chaque service web.	43
Tableau 6: Résultats obtenus	44

Introduction

Générale

Rapport-Gratuit.com

I .Introduction générale

Contexte

La Technologie a toujours été un secteur d'innovation continue, notamment ces dernières années. En particulier, la communication et le partage de données et de ressources qui sont récemment un sujet qui fait l'objet de plusieurs travaux dans le domaine de la technologie. L'une de ces innovations technologiques qu'on l'appelle le service web. Ce dernier est basé sur le modèle SOA¹, qui a été créé comme une réponse aux inconvénients des technologies orientées composantes. Récemment, plusieurs entreprises se servent des services web, notamment les applications avec des liaisons fixes aux services web.

Problématique

L'un des problèmes majeurs de nos jours est la sélection des services web. En vue de l'augmentation des exigences des clients. Le problème principal posé est d'avoir une sélection basée sur l'une des QoS² dynamiques (qui varient dans le temps). On s'intéresse dans ce mémoire aux services web qui sont caractérisés par les critères positifs.

Notre objectif est de trouver les compositions des services qui optimisent les QoS médianes de ces critères, en plus, elles doivent satisfaire les contraintes globales. Dans ce mémoire, nous supposons que tous les critères sont positifs (c.-à-d. ils nécessitent une maximisation).

Contributions

Dans ce travail, nous proposons une adaptation de service web pour une sélection plus qualitative en utilisant un algorithme stochastique.

L'algorithme stochastique consiste à chercher une meilleure sélection de service web composite à l'aide de la fonction de dominance stochastique. Pour ce faire nous sommes inspiré des algorithmes stochastiques afin d'aider les clients à sélectionner les meilleurs services web par rapport à des critères positifs proposés.

Organisation du mémoire :

C¹e travail est organisé en deux (2) chapitres, il est structuré comme suit :

Chapitre 1: Ce chapitre se divisera en deux parties:

La première partie présente l'architecture orientée service (SOA), dont on cite ses principes et ses avantages et ses inconvénients. La deuxième partie met l'accent sur la technologie des services web, le cycle de vie ainsi que les principaux standards qu'ils supportent et ses avantages et les inconvénients, nous montrerons aussi un survol sur la sélection des services web composites à base de QoS et la spécification du problème de sélection avec Qos déterministe.

Chapitre 2: Ce chapitre est consacré à la conception et l'implémentation du prototype d'application. Au début, nous présenterons le corpus utilisé, puis nous expliquerons les algorithmes utilisés et leurs fonctions avec des exemples détaillés. Enfin, on présentera les outils utilisés et l'application.

1-SOA: Service Oriented Architecture ou Architecture oriented services.
2- QoS : quality of service.

Chapitre : I

L'architecture SOA

et

les services web

I. Introduction :

Le sigle SOA est devenu à la mode au début de 2005 grâce aux succès du déploiement d'Internet dans le public et dans les entreprises.

L'architecture orientée services SOA vise à créer une architecture qui s'appuie sur l'utilisation des services tels que les services Web RESTful³. Ces services remplissent des fonctions, comme la production des données, la validation d'un client ou la mise à disposition d'analyses simples.

Dans ce chapitre, nous présenterons l'architecture orientée service et sa principale réalisation, ainsi que ses avantages et ses inconvénients. Ensuite, nous donnerons les définitions des services web et décrirons brièvement leurs principaux standards. Enfin, nous aborderons le concept de service web en expliquant la sélection des services web.

I. L'Architecture Orientée Service (SOA) :

I.1.Définitions :

I.1.1Définition Métier :

« L'architecture orientée service est un ensemble de méthodes techniques, métiers, procéduraux, organisationnels et gouvernementaux pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information pendant la création d'un environnement métier, agile pour un² intérêt concurrentiel.»[1]

I.1.2 Définitions Techniques :

« L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoirs-faires distribués pouvant être des domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs-faires pour produire le résultat désiré avec des pré-conditions et des buts mesurables.»[2]

3-RESTful: representational state transfer.

« L'architecture SOA permet l'intégration d'applications et de ressources de manière flexible, en représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant à un service d'échanger des informations structurées (messages, documents, objets métiers), coordonnées et en organisant les services, afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement.»[3]

« Une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant des composants de services sécurisés, standardisés qui peuvent être combinés pour s'adresser aux priorités de changements métiers.»[4]

I.3.Principes fondamentaux de l'architecture SOA :

Une architecture orientée services repose sur ces principes :

- La localisation des services via un annuaire.
- Les services peuvent être synchrones ou asynchrones.
- La publication et la découverte des services.
- La propriété de couplage faible.
- Utilisation des standards.
- Pas de remise à cause de l'existant lors d'évolutions technologiques.
- Découplage entre fournisseur et consommateur de services.
- La réutilisation et la composition.
- L'indépendance par rapport aux aspects technologiques.
- La mise à l'échelle est rendue possible grâce à la découverte et à l'invocation des nouveaux services lors de l'exécution [5].

I.4.Les avantages et les inconvénients de SOA :

I.4.1.Les avantages :

- Une modularité permet de remplacer facilement un service par un autre.
- De meilleures possibilités d'évolution.
- Une plus grande tolérance aux pannes.
- Une maintenance facilitée.
- Localisation et interfaçage transparents et la possibilité de mise en place.
- Possibilité de partage des modules entre les autres applications et cette application.
- Réduction du coût de développement des grands projets.
- Possibilité de choisir le protocole de transport.
- Augmentation de la flexibilité [6] [7].

I.4.2. Les inconvénients :

- Les Coûts de conception et de développement initiaux plus conséquents.
- Nécessité d'appréhender de nouvelles technologies.
- Existant non SOA dans les entreprises.
- Performances réduites pour des traitements simples.

II. Service Web :

II.1. Historique :

Le World Wide Web s'accroît de manière exponentielle depuis les années suivantes :

- **En 1990** : ce phénomène est dû en partie aux entreprises qui ont vues dans l'Internet l'outil par excellence pour se faire connaître. Mais également aux organismes ayant vocation à fournir un service public d'information, de proximité ou non.

- **En 1993** : les entreprises et les états ont exploité Internet et en particulier les serveurs web pour faire le commerce électronique ou bien les gouvernements électroniques.
- **En 1998** : les services web prennent leurs origines dans l'informatique distribuée et dans l'événement de Web [8].

II.2. Définition :

On trouve plusieurs définitions des services web :

Définition d'IBM

« Les services web sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer » [9].

Définition de W3C

« Un service web est un système logiciel identifié par un URI⁴ dont les interfaces publiques et Les incarnations sont définies et décrites en XML⁵. Sa définition peut être découverte Dynamiquement par d'autres systèmes logiciels. Ces derniers peuvent ensuite interagir avec le service web d'une façon³ décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet » [10].

II.3 Cycle de vie des services web :

Les trois éléments les plus importants des services Web sont :

Les fournisseurs de service, les annuaires de service et les consommateurs de service.

4-URI: Uniform Resource Identifier.

5-XML: extensible Markup Language.

- Le fournisseur (ou serveur) crée le service Web et publie toutes ces caractéristiques dans l'annuaire de service.
- L'annuaire rend disponible les interfaces d'accès aux services et donne le contrat et l'architecture employée pour permettre les interactions.
- Le consommateur (ou client) quant à lui, accède à l'annuaire pour rechercher les services web dont il a besoin et avec lui les normalisations à obtenir. Il peut ainsi envoyer ses requêtes au service désiré et obtenir les réponses qu'il pourra analyser.

Cette architecture fonctionne de la manière suivante :

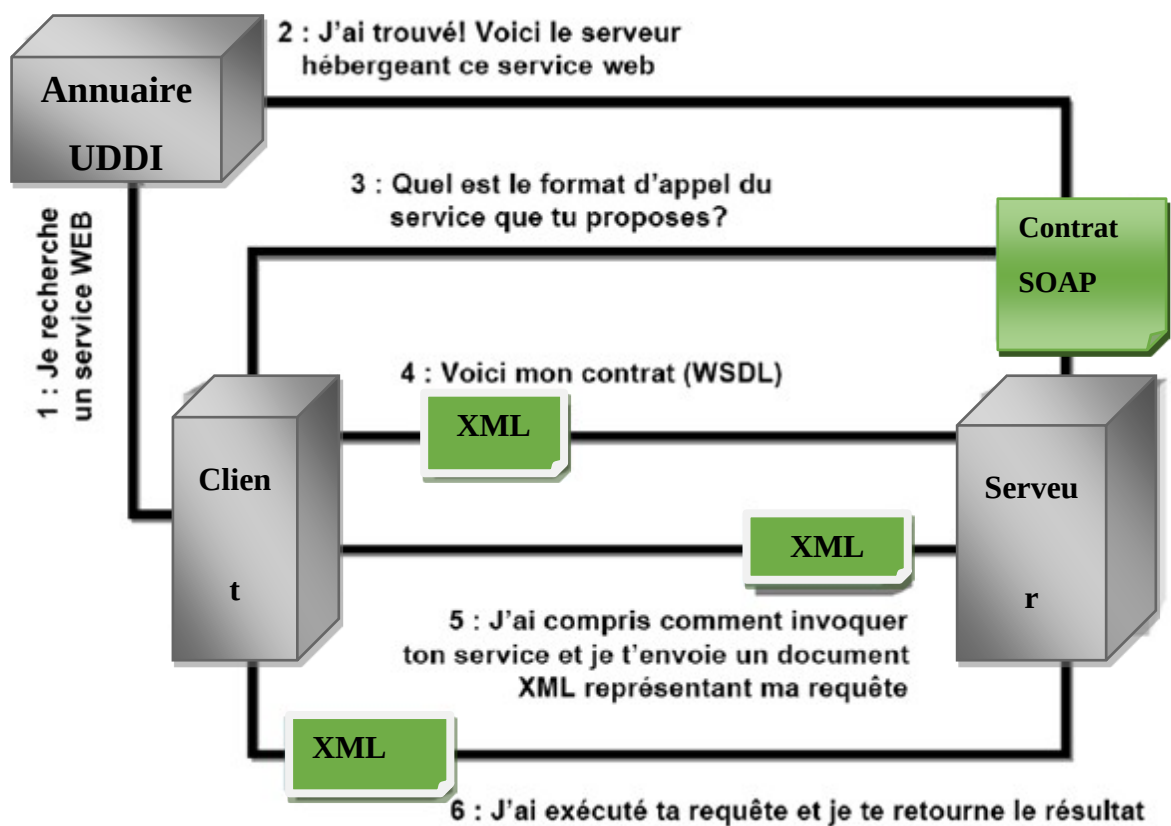


Figure 1: Cycle de vie des services web [11].

- Le client envoie une requête à l'annuaire de Service pour trouver le service web dont il a besoin.
- L'annuaire cherche pour le client, trouve le service web approprié et renvoie une réponse au client en lui indiquant quel serveur détient ce qu'il recherche.

- Le client envoie une deuxième requête au serveur pour obtenir le contrat de normalisation de ses données.
- Le serveur envoie sa réponse sous la forme établie par WSDL⁶ en langage XML.
- Le client peut maintenant rédiger sa requête pour traiter les données dont il a besoin.
- Le serveur fait les calculs nécessaires suite à la requête du client, et renvoie sa réponse sous la même forme normalisée [12].

II.4. Standard des Services Web :

Un fournisseur de services héberge un service web et le rend accessible à l'aide de protocoles de type SOAP⁷/http⁸. Il est aussi décrit par un document WSDL stocké sur le serveur du fournisseur ou dans un référentiel particulier, le document WSDL peut être référencé par le registre d'entités UDDI⁹.

II.4. 1. SOAP :

SOAP est un protocole de RPC permettant d'invoquer des méthodes d'objets distants. Il s'appuie sur des standards très connus. Il utilise XML pour définir les fonctions et les définitions disponibles. Il prend en charge divers protocoles de transport, tels que HTTP et SMTP, ainsi que différents formats comme MIME¹⁰. Ces derniers⁴ sont très répandus sur de multiples plates-formes, ce qui donne à SOAP une grande portabilité et interopérabilité.

6-WSDL: Web Service Description Language.

7-SOAP: simple Object Access Protocol.

8-HTTP: Hyper Text Transport Protocol.

9-UDDI: Universal Description Discovery and Integration.

10-MIME: Multipurpose Internet Mail Extensions.

SOAP est une spécification non-propriétaire. Il n'est pas lié à un protocole particulier. Il n'est pas non plus lié à un système d'exploitation ni au langage de programmation.

✚ Structure d'un message SOAP :

La grammaire de SOAP procure un moyen d'accès aux objets par appel aux méthodes à distance.

Les deux plus fortes fonctionnalités de SOAP sont sa simplicité et le fait que tout le monde a accepté de l'utiliser.

Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP, et une partie optionnelle : l'en-tête SOAP.

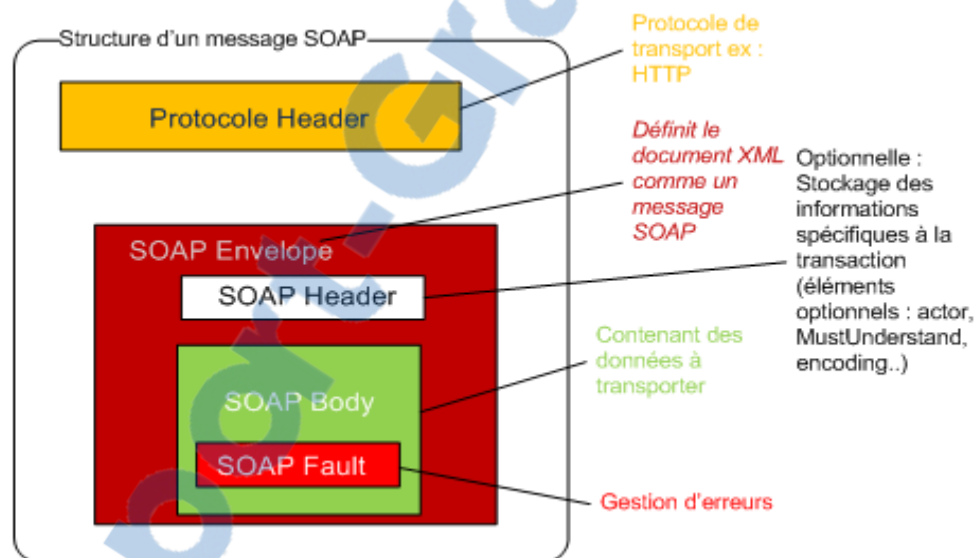


Figure 2. Structure d'un message SOAP [13].

- **Enveloppe:** qui contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Il peut contenir un attribut `encodingStyle` dont la valeur est un URL¹¹ vers un fichier de typage XML qui décrira les types applicables au message SOAP.

11-URL: Uniform Resource Locator, littéralement « localisateur uniforme de ressource ».

- **Header:** c'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc Body à l'intérieur du bloc Enveloppe.
- **Body:** c'est le bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc Enveloppe. SOAP ne définit pas comment est structuré le contenu de ce bloc. Cependant, il définit le bloc Fault qui peut s'y trouver.
- **Fault:** ce bloc est la seule structure définie par SOAP dans le bloc Body. Il sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire [14].

Exemple de fichier soap:

Voici un exemple typique qui décrit la structure d'un message SOAP :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <soap:Header>
    <!--optionnel -->
    ...
  </soap:Header>

  <soap:Body>
    <!--requis -->
    ...

    <soap:Fault>
      ...
    </soap:Fault>
```

```
</soap:Body>

</soap:Envelope>
```

Figure 3.Exemple de la structure d'un message SOAP.

II.4. 2. WSDL :

WSDL est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

✚ Les éléments de fichier WSDL:

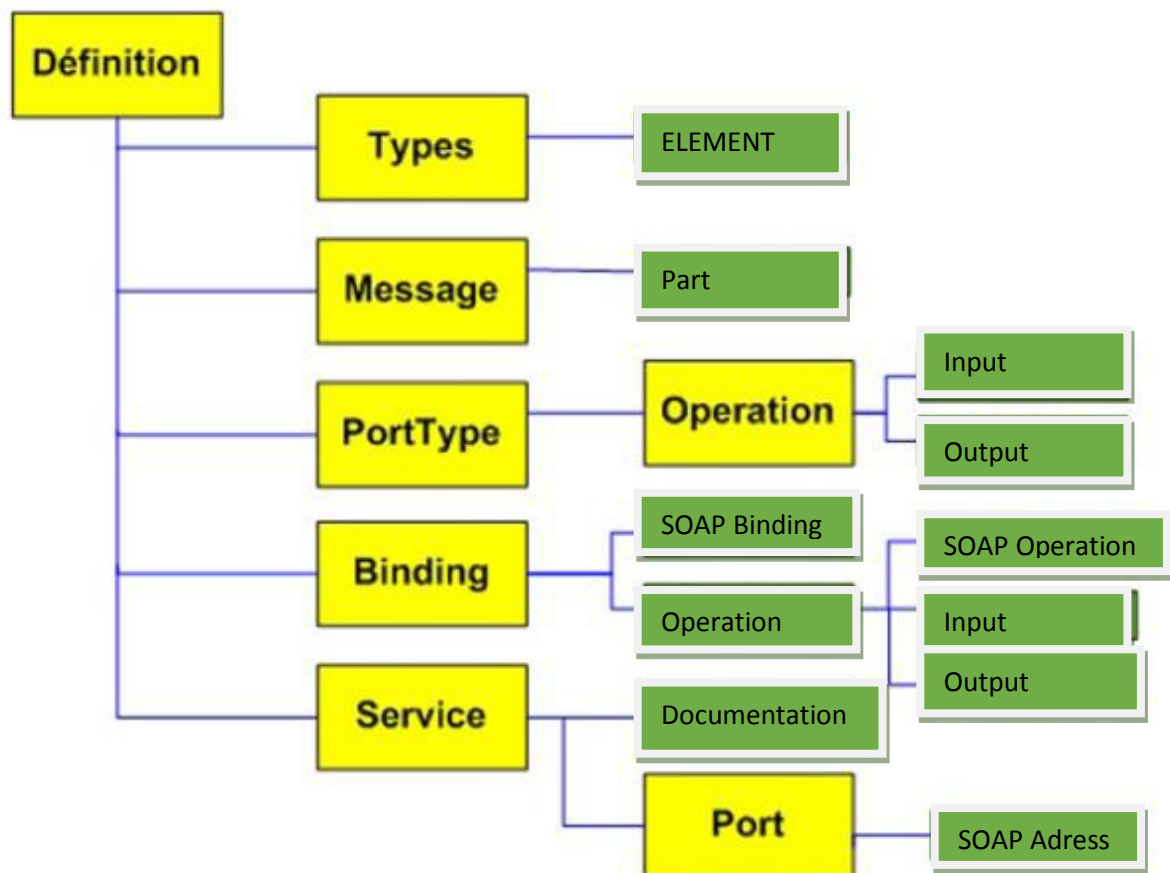


Figure 4: Structure d'un document WSDL [15].

Un fichier WSDL contient sept éléments :

- **Types:** fournit la définition de types de données utilisées pour décrire les messages échangés.
- **Messages:** représentent une définition abstraite (noms et types) des données en cours de transmission.
- **PortTypes:** décrit un ensemble d'opérations. Chaque opération à zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreur.
- **Binding:** spécifie une liaison entre un `<portType>` et un protocole concret (SOAP,HTTP...).
- **Service:** indique les adresses de port de chaque liaison.
- **Port:** représente un point d'accès de services défini par une adresse réseau et une liaison.
- **Opération:** c'est la description d'une action exposée dans le port [15].

 **Exemple de fichier WSDL:**

```

1  <!-- Définitions de types -->
2  <types>
3    <xsd:schema targetNamespace="urn:xml-soap-address-demo"
4      xmlns:xsd="http://www.w3.org/1999/XMLSchema">
5      <xsd:complexType name="telephone">
6        <xsd:element name="areaCode" type="xsd:int"/>
7        <xsd:element name="exchange" type="xsd:string"/>
8        <xsd:element name="numero" type="xsd:string"/>
9      </xsd:complexType>
10     <xsd:complexType name="adresse">
11       <xsd:element name="numeroRue" type="xsd:int"/>
12       <xsd:element name="nomRue" type="xsd:string"/>
13       <xsd:element name="ville" type="xsd:string"/>
14       <xsd:element name="codePos" type="xsd:int"/>
15       <xsd:element name="pays" type="xsd:string"/>
16       <xsd:element name="numTel" type="typens:telephone"/>
17     </xsd:complexType>
18   </xsd:schema>
19 </types>
20
21 <!-- Déclarations de messages -->
22 <message name="AjouterEntreeQuestion">
23   <part name="nom" type="xsd:string"/>
24   <part name="adresse" type="typens:adresse"/>
25 </message>
26 <message name="GetAdresseFromNomQuestion">
27   <part name="nom" type="xsd:string"/>
28 </message>
29 <message name="GetAdresseFromNomReponse">
30   <part name="adresse" type="typens:adresse"/>
31 </message>
32
33 <!-- Déclarations de type de port -->
34 <portType name="CarnetDAdresse">
35   <operation name="ajouterEntree">
36     <input message="AjouterEntreeQuestion"/>
37   </operation>
38   <operation name="getAdresseFromNom">
39     <input message="GetAdresseFromNomQuestion"/>
40     <output message="GetAdresseFromNomReponse"/>
41   </operation>
42 </portType>
43

```

Figure 5. Exemple de Code WSDL [16].

II.4. 3.UDDI :

UDDI Proposé par un groupe d'entreprises qui œuvre dans le domaine informatique.

UDDI (OASIS-UDDI, 2002) est un annuaire qui permet aux entreprises fournissant des services Web de s'enregistrer et de publier les services web qu'elles offrent au grand public.

L'idée principale d'UDDI est de standardiser le format des entrées d'entreprise et de services dans un annuaire.

Ceci permettant, éventuellement, d'automatiser le processus de découverte de services web afin de faciliter les échanges d'affaires entre entreprises (B2B)¹².

Un registre UDDI est défini comme un méta service qui utilise les protocoles SOAP et HTTP et qui suit un modèle d'échange de données basé sur le langage XML (Figure6).

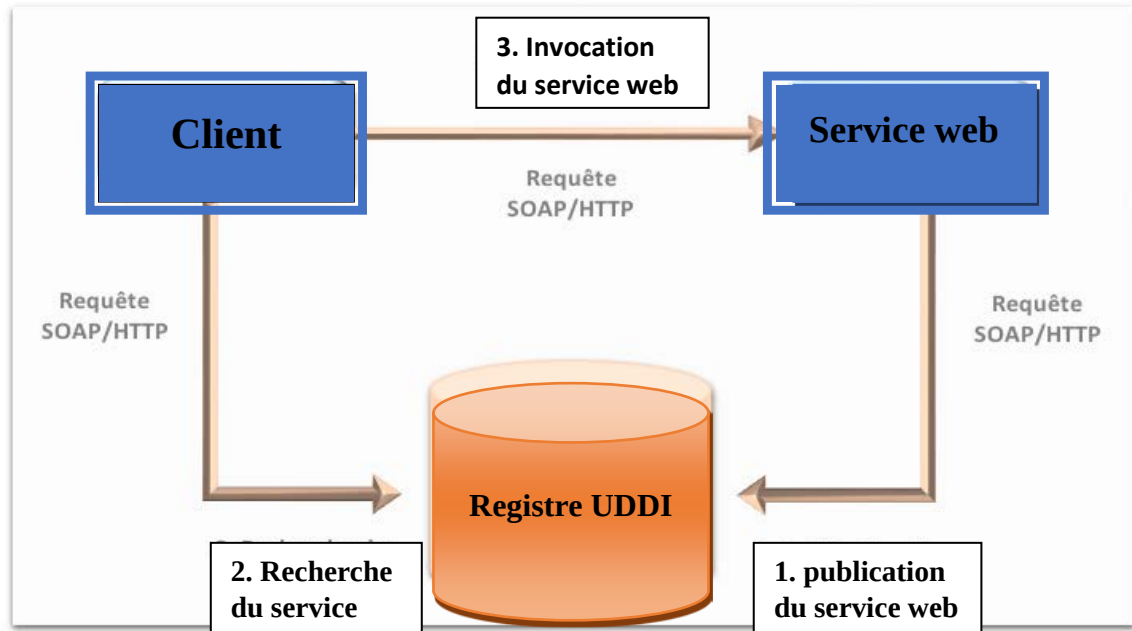


Figure 6. Publication et recherche dans un registre UDDI.

Actuellement, il existe sur le marché plusieurs implémentations de registres UDDI. Microsoft et IBM ont été les premiers à mettre sur le marché des registres UDDI accessibles à travers un ⁵navigateur Web. Ils fournissent aussi des registres UDDI d'essai pour encourager les entreprises à utiliser cette norme.

Structure de l'information dans un registre UDDI :

Un enregistrement UDDI est un document XML qui décrit une entreprise et les services Web qu'elle offre. Il contient trois types d'informations:

- **Page blanche:** contient les détails d'une entreprise. Ces détails sont décrits dans une entité de type BusinessEntity (Figure7).

12-B2B: business to business.

- **Page jaune** : contient les détails d'un service Web proposé par une entreprise. Ces détails sont décrits dans une entité de type BusinessService (Figure7).
- **Page verte** : contient les détails techniques d'un service Web et une référence vers une description détaillée de l'utilisation de ce service web. Ces détails sont décrits dans des entités de type Binding Template et tModel (Figure7).

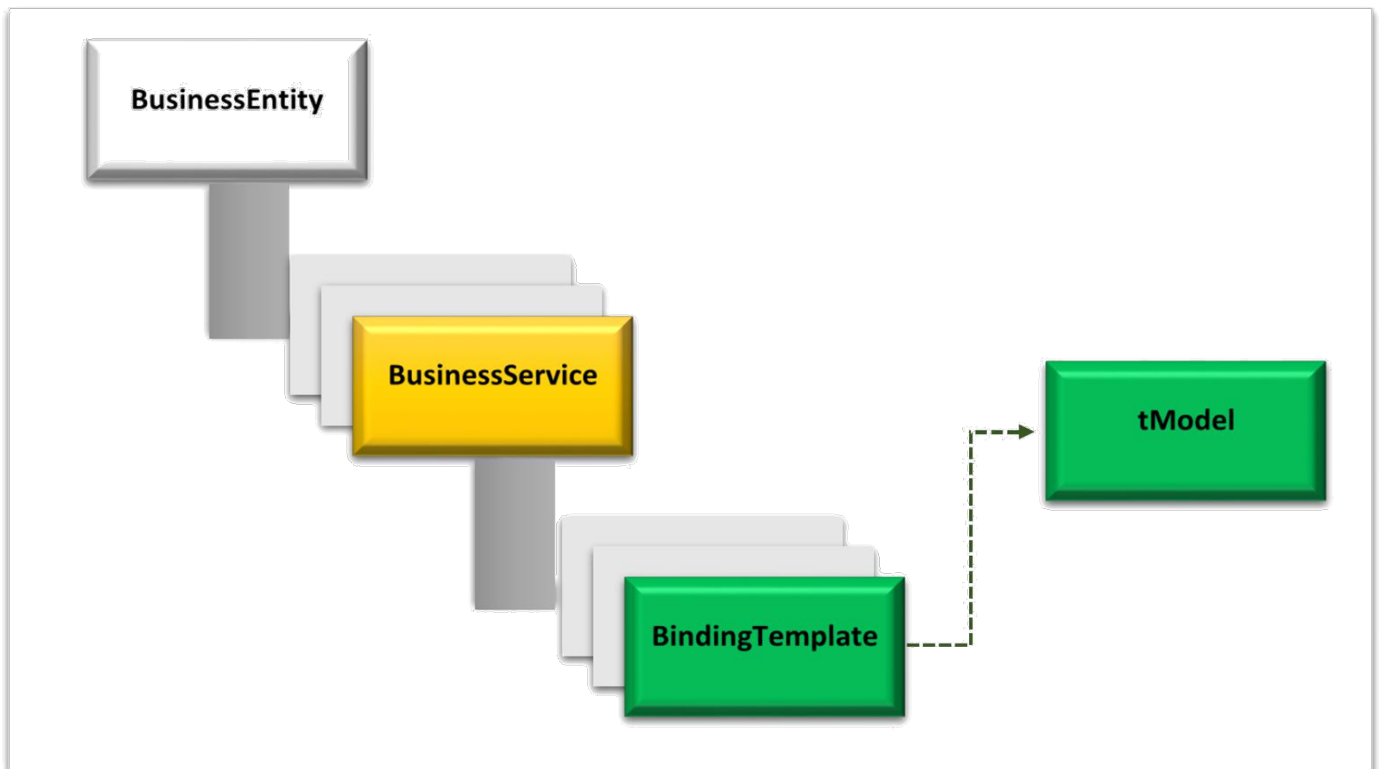


Figure 7. Structure de l'information dans un registre UDDI [17].

II.5. Les caractéristiques d'un service Web :

Un service web possède les caractéristiques suivantes :

- Les services web peuvent être définis, décrits et découverts à travers le Web.
- Il dispose d'une interface publique décrite en XML.
- Les descriptions sont stockées dans un annuaire.

- Les services Web peuvent communiquer par des différentes applications.
- En utilisant des messages XML, ces messages sont transportés par des protocoles Internet.
- Fonctionné quel que soit la plateforme.
- Indépendant des langages de programmation et des systèmes d'exploitation.
- Les services Web sont basés sur les protocoles et les langages du Web, en particulier HTTP et XML.
- Une application logicielle qui est reconnue par un URI.
- Il est accessible via le réseau.
- Le modèle Service Web est modulaire [18].

II.6. Les avantages et les inconvénients des services web :

II.6.1. Les Avantages :

- Les services web réduisent le temps de mise en marché des services offerts par les diverses entreprises.
- Les services web avec ses protocoles et ses standards avancent toujours vers plus de normalisation.
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.
- Réduisent la complexité des branchements entre les participants.
- Permettent d'ignorer l'hétérogénéité entre les différentes applications.
- Les services web ont la particularité d'être auto-descriptifs.
- Il est très flexible.

- Permettent de garantir l'interopérabilité entre les applications.
- Le client doit simplement exprimer ces besoins et les entreprises doivent simplement publier leur service web.
- Offrent une interface simple.
- Ils autorisent un couplage faible entre le client et le serveur [19] [20].

II.6.2 Les Inconvénients :

- Les relations de confiance entre différentes composantes d'un service web sont difficiles à bâtir, puisque parfois, ces mêmes composantes ne se connaissent même pas.
- Il est difficile de s'assurer de la fiabilité.
- On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (sémantique).
- Le transfert de données en XML est moins efficace qu'en binaire.
- Ils ne sont pas sécurisés à 100 %.
- Les normes de services Web dans certains domaines sont actuellement récentes.
- Les services web possèdent de faible performance par rapport à d'autres approches de l'informatique répartie telles que le RMI¹⁵, CORBA¹⁶, ou DCOM¹⁷.
- Manque de polyvalence [21].

III.1.Sélection des services web :

La sélection des services web consiste à choisir parmi les services déjà découverts ceux qui répondent au mieux aux requêtes sur des besoins fonctionnels et/ou non-fonctionnels des utilisateurs.

Les critères de QoS aident généralement les besoins non-fonctionnels des services web à s'exprimer. Ces derniers sont composés d'attributs positifs et négatifs. Les valeurs des attributs négatifs doivent être minimisées (temps de réponse, Coût ...), ⁶tandis que celle des attributs positifs doivent être maximisée (disponibilité, fiabilité,...). Afin de simplifier les calculs, nous considérons seulement les attributs positifs, pour les négatifs, il suffit de multiplier leurs valeurs par (-1).

Et pour sélectionner les compositions des services Web, on doit alors assigner un service Web à chaque classe. L'ensemble des plans d'exécution possibles est combinatoire : pour N classes avec un choix parmi M services pour chacune, il y a M^N de compositions possibles [22].

III.2. Qualité des services web :

La QoS a été définie [23] comme "l'ensemble des caractéristiques quantitatives et qualitatives d'un système multimédia, nécessaires pour atteindre la fonctionnalité requise par l'application".

Les principales composantes de la qualité de service sont :

- Le temps d'exécution d'un service.
- Le coût d'un service.
- La réputation d'un service.
- La disponibilité d'un service.
- La fiabilité d'un service.
- **Temps d'exécution:**
Également appelé **Le Temps de latence** ou de **performance** ou de **réponses**.

13-RMI: revenue minimum d'insertion.

14-CORBA: Common Object Request Broker, Object Management Group.

15-DCOM: Distributed Component Object Model.

- **Défini par** : Le temps nécessaire pour traiter une requête par un service web jusqu'au moment de la réception de la réponse.
- **Formule Temps d'exécution** = Le temps de réception de la réponse par le client - Le temps d'envoi de la requête par le client.
- **Quantification** : seconde [24] [25].

➤ **La Disponibilité :**

- **Définie par:** (la probabilité) Le pourcentage de requêtes réussites par le fournisseur à une certaine période de temps. Les réponses échouées correspondent aux exceptions reçues du côté client.
- **Formule Disponibilité** = Nombre de requêtes réussites / Nombre total de requêtes.
- **Quantification** : Pourcentage [24] [25].

➤ **Fiabilité** : est également appelée sûreté.

- **Définie par:** la capacité d'un service de répondre correctement à une requête dans le délai prévu au maximum.
- **Formule fiabilité** = le nombre de fois qu'opération a été exécutée avec succès / le nombre total d'appels.
- **Quantification:** Pourcentage [24] [25].

➤ **La réputation :**

- **Définie par:** une mesure de sa fidélité. Elle dépend principalement des expériences de l'utilisateur d'employer ce service [24] [25].

➤ **Coût** : également appelé **financière** ou le cours.

- **Défini par:** les frais liés à un service. Le prix d'exécution est la somme d'argent qu'un demandeur de service doit payer pour exécuter une opération d'un service [24] [25].

III .3. Exemple de motivation :

Pour présenter les problématiques de la sélection des compositions de services « QoS-aware service composition », on considère la situation suivante :

On suppose qu'il y a un utilisateur qui veut planifier un voyage, pour cela, il a besoin de consommer 3 types de services au minimum, une réservation de billet d'avion, une réservation d'hôtel et une location d'un Taxi comme le montre la (figure 8). On note aussi qu'on doit sélectionner un seul service (ou entreprise, fournisseur,...) de chaque catégorie (ou classe) en utilisant les critères de QoS (réputation, fiabilité, coût, temps d'exécution...). En plus, l'utilisateur exige des contraintes globales sur chaque critère de QoS. Une contrainte globale qui s'applique sur les 3 services sélectionnés, par exemple le coût total des 3 services ne doit pas excéder une certaine limite [26].

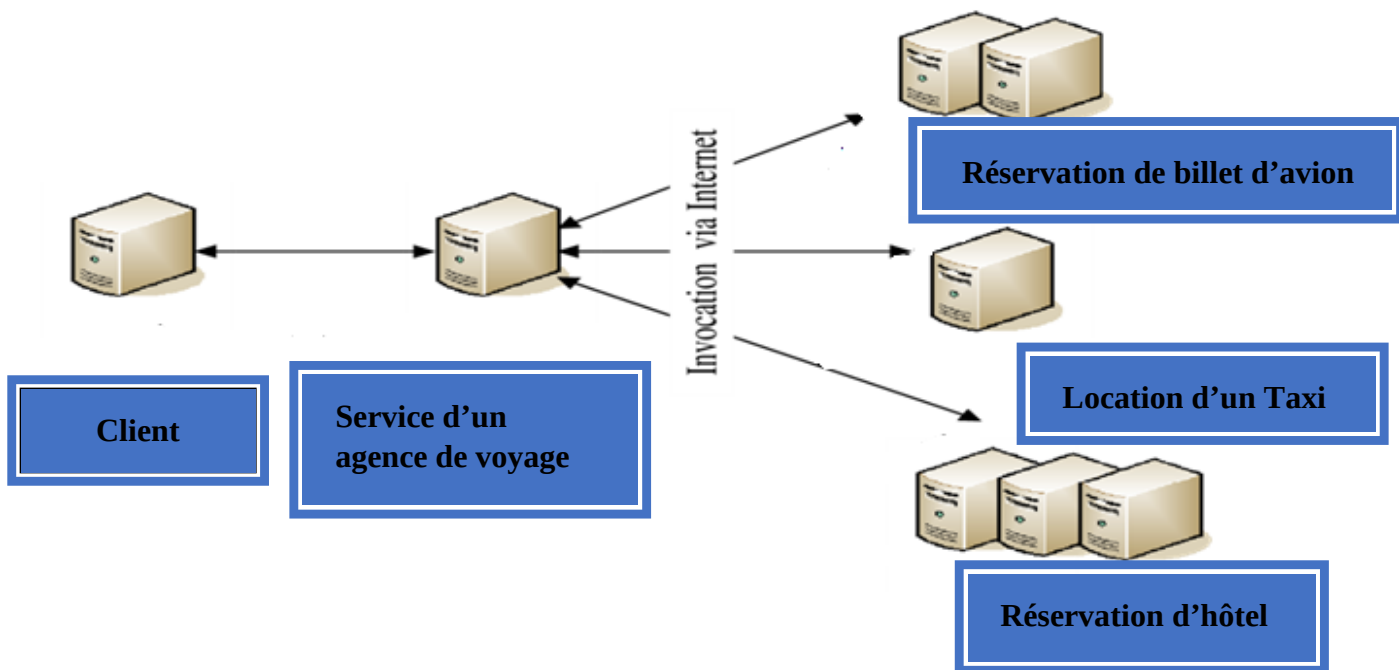


Figure 8. Exemple de composition des services Web « Planification de voyage ».

III.4. Spécification du problème de sélection avec QoS déterministe :

Nous définissons la fonction objective pour calculer une valeur hors du vecteur de QoS. Cela peut faciliter la comparaison de qualité. Comme les utilisateurs peuvent avoir des préférences sur la façon dont leurs requêtes sont traitées, elles peuvent indiquer l'importance relative des paramètres QoS. Nous attribuons des poids allant de 0 à 1 à chaque paramètre QoS pour refléter le niveau des importances.

Paramètres QoS	Fonction d'agrégation
Latence	$\sum_{i=1}^n \text{latence}(\text{op } i)$
Fiabilité	$\prod_{i=1}^n \text{fiabilité}(\text{op } i)$
Disponibilité	$\prod_{i=1}^n \text{disponibilité}(\text{op } i)$
Coût	$\sum_{i=1}^n \text{coût}(\text{op } i)$
Coût	$\frac{1}{n} \sum_{i=1}^n \text{réputation}(\text{op } i)$

Tableau 1: les fonctions d'agrégation des QoS.

Nous utilisons la fonction objective F suivante pour évaluer la qualité des plans d'exécution du service, l'optimisation QoS est de trouver le plan d'exécution avec le score maximum.

Q_i est la $i^{\text{ème}}$ qualité de service, Q_i est calculé par la fonction d'agrégation,

$$f(\text{comp}) = \sum_{Q_i \in (Q_1, \dots, Q_r)} \left(W_i \frac{QoS.AGC(\text{comp}, i) - Q_{i_{\min}}}{Q_{i_{\max}} - Q_{i_{\min}}} \right)$$

W_i représente le poids de la $i^{\text{ème}}$ indice de composant.



r décliné le nombre de service.

$Q_i \text{ max}$ est la valeur maximale pour le $i^{\text{ème}}$ paramètre QoS pour tous les plans possibles d'exécution du service et $Q_i \text{ min}$ est le minimum [27].

IV. Conclusion :

Dans ce chapitre on a présenté quelques définitions sur les services web avec une architecture ainsi que les standards de communication entre les services web. Après, on a présenté le problème de sélection en citant un exemple de motivation et on a terminé par la Spécification du problème de sélection. Et par la suite on va considérer dans le chapitre suivant l'implémentation de notre application et de donner une expérimentation.



Chapitre: II

Conception et Implémentation du prototype

I. Introduction :

L'objectif de ce dernier chapitre est présenté notre approche pour résoudre le problème de sélection des meilleurs compositions des services web avec QoS stochastique. On commence par une présentation de corpus, et ensuite nous présentons un algorithme dominance stochastique et les fonctions utilisées et un exemple détaillé pour illustrer notre travail et ainsi que, nous citons les différents outils et environnements de développement ainsi que la présentation de l'application.

II. Présentation du Corpus :

II .1.Description de base :

Notre base est un ensemble de fichiers XML «**datasetsrv1.xml, datasetsrv2.xml**» (c-a-d $N=2$) regroupant la liste des services Web, tel que chaque service possède, 7 instances se caractérisent par des qualités de service. Les valeurs de ces qualités sont générées en se basant sur la loi l'uniforme $[0,100]$.

II .2.Description de requête:

La requête de l'utilisateur consiste à vérifier la qualité de service de la composition sélectionnée par le système, en considérant les valeurs minimales des critères de QoS (puisque tous les attributs sont positifs). La borne minimale de tout critère de QoS est: $60*N$ (dans notre travail, chaque contrainte globale vaut 120, puisque $N=2$).

III. Conception :

Dans cette section, nous allons présenter les différents algorithmes que nous avons développés dans notre application.

III. 1.1.Etape 1 : Sélection local des services web.

 **Comparaison de deux services à base de dominance stochastique :**

Entrée

i, j : numéraux de service.

dataset : base de service.

ni : nombre des instances.

N.Qos : nombre de qualité de service.

Sortie: top k service.

Real domin_proba(int i, int j)

i ≠ j

Debut

Vecteur v1= Get_instances(i, dataset)

Vecteur v2 = Get_instances(j, dataset)

Score=0 ;

pour l de 1 a ni faire

pour m de 1 a ni faire

Inst1=Get_instance (l, v1)

Inst2=Get_instance (m, v2)

boolean b=true ;

pour k de 1 a N.Qos faire

SI (Inst1[k] <Inst2[k]) alors

b=false;

fin si;

fin pour;

SI(b) alors

Score=score*(1/ (ni) * (ni))

Fin si;

Fin pour ;

fin pour ;

Retourne score.

Classement local des services à base de dominance stochastique :

pour i ← 0 à |dataset|

pour j ← 0 à |dataset|

S1=Domin_Proba(Si, Sj)

S2=Domin_Proba(Sj, Si)

SI (S1 > S2) alors

Degre (Si) = +1

Sinon

Degre (Sj) = +1

Fin si;

Fin pour;

Fin pour;

Trier _order _décroissant (Degre);

Return (top_k (Trier _order _décroissant)).

L'approche probabilité dominance :

L'algorithme « **domin-proba** » que nous avons utilisé est basé sur la notion de probabilité dominance, composée de deux mots la Dominance et la probabilité.

La relation de dominance est une relation binaire, si on considère A et B deux éléments, on dit que A domin B si est seulement si A est plus grand est plus fort et plus pertinent par rapport à B.

Le calcul de la probabilité désigne l'utilisation de raisonnements probabilistes dans la résolution de problèmes purement déterministes.

Nous donnons La fonction de probabilité dominance entre les deux services représentée sous la forme suivante :

$$\text{domin-proba}(S_i, S_j) = \frac{1}{n_i^2} \sum_{L=1}^{n_i} \sum_{L'=1}^{n_i} |S_{i,L} \geq S_{j,L'}|$$

Tel que :

n_i : le nombre des instances.

$|S_i \geq S_j|$: indique que Si domin Sj.

III. 1.2. Les fonctions utilisées :

La fonction « domin_proba »:

Comparaison de deux services à base de dominance stochastique, elle prend comme paramètre le service **Si** et le service **Sj**, elle retourne une valeur qui représente la probabilité dominance entre eux.

La fonction «Trier_order_décroissant»:

Elle a pour rôle de trier la liste des services par ordre décroissant selon le degré.

La fonction «Get_instances»:

Permet de récupérer tout les instances de service.

La fonction «Get_instance»:

Permet de récupérer l'instance pour chaque service.

III. 1. 3. Exemple d'application :

Ayant les cinq services suivants: {S1, S2, S3, S4, S5}

$||\text{Dataset}|| = 5$.

S1 {(0.15, 0.18), (0.23, 0.32), (0.22, 0.25)}

S2 {(0.50, 0.1), (0.22, 0.1), (0.3, 0.2)}

S3 {(0.84, 0.9), (0.1, 0.1), (0.78, 0.64)}

S4 {(0.74, 0.6), (0.2, 0.3), (0.5, 0.52)}

S5 {(0.1, 0.2), (0.63, 0.48), (0.4, 0.3)}

En appliquant la formule Domin-Proba :

Domin-Proba(S1,S2)	$1/3(1/3*(0)+1/3*(1)+1/3*(1))=2/9=0.22$
Domin-Proba(S2,S1)	$1/3(1/3*(0)+1/3*(0)+1/3*(1))=1/9=0.11$
Domin-Proba(S1,S3)	$1/3(1/3*(1)+1/3*(1)+1/3*(1))=3/9=0.33$
Domin-Proba(S3,S1)	$1/3(1/3*(3)+1/3*(0)+1/3*(3))=6/9=0.66$
Domin-Proba(S1,S4)	$1/3(1/3*(0)+1/3*(1)+1/3*(0))=1/9=0.11$
Domin-Proba(S4,S1)	$1/3(1/3*(3)+1/3*(1)+1/3*(3))=7/9=0.77$
Domin-Proba(S1,S5)	$1/3(1/3*(0)+1/3*(1)+1/3*(1))=2/9=0.22$
Domin-Proba(S5,S1)	$1/3(1/3*(0)+1/3*(3)+1/3*(2))=5/9=0.55$
Domin-Proba(S2,S3)	$1/3(1/3*(1)+1/3*(1)+1/3*(1))=3/9=0.33$
Domin-Proba(S3,S2)	$1/3(1/3*(3)+1/3*(0)+1/3*(3))=6/9=0.66$
Domin-Proba(S2,S4)	$1/3(1/3*(0)+1/3*(0)+1/3*(0))=0/9=0$
Domin-Proba(S4,S2)	$1/3(1/3*(3)+1/3*(0)+1/3*(3))=6/9=0.66$
Domin-Proba(S2,S5)	$1/3(1/3*(0)+1/3*(0)+1/3*(1))=1/9=0.11$
Domin-Proba(S5,S2)	$1/3(1/3*(0)+1/3*(3)+1/3*(2))=5/9=0.55$
Domin-Proba(S3,S4)	$1/3(1/3*(3)+1/3*(0)+1/3*(3))=6/9=0.66$
Domin-Proba(S4,S3)	$1/3(1/3*(1)+1/3*(1)+1/3*(1))=3/9=0.33$
Domin-Proba(S3,S5)	$1/3(1/3*(3)+1/3*(0)+1/3*(3))=6/9=0.66$
Domin-Proba(S5,S3)	$1/3(1/3*(1)+1/3*(1)+1/3*(1))=3/9=0.33$
Domin-Proba(S4,S5)	$1/3(1/3*(3)+1/3*(1)+1/3*(2))=6/9=0.66$
Domin-Proba(S5,S4)	$1/3(1/3*(0)+1/3*(0)+1/3*(1))=2/9=0.22$

Tableau 2 : Le score de probabilité dominance des services.

✚ Le degré de chaque service :

Les services :	Degré de chaque service :
S1	1
S2	0
S3	4
S4	3
S5	2

Tableau 3 : Le degré de chaque service.

✚ Classement local des services selon le degré :

Le classement final de ces services est donné dans le graphe suivant :

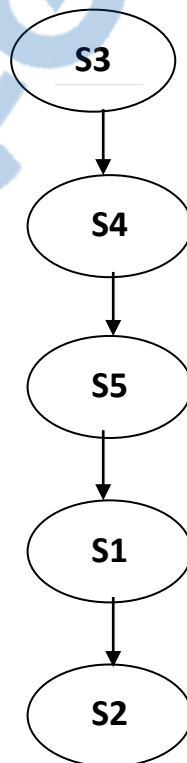


Figure 9. Graphe du classement selon la probabilité dominance.

III. 2. 1. Etape2: compositions des services.

Optmisation_globale :

Entrée : top k service.

k : nombre de compositions de sortie.

b1, b2, ..., b10 : contrainst global.

/Max de nbr_tache=5*/

Sortie : top k compositions.

Top_compo=<>; // liste vide.

pour i1 de 1 a k

S1=Get_current_service (i1, 1)

Update (compo, S1)

pour i2 de 1 a k

S2=get_current_service (i2, 2)

Update (compo, S2)

pour i3 de 1 a k

S3= Get_ current_service (i3, 3)

Update (compo, S3)

pour i4 de 1 a k

S4=Get_current_service (i4, 4)

Update (compo, S4)

pour i5 de 1 a k

S5=Get_current_service (i5, 5)

Update (compo, S5)

SI c.g.s (compo, b1, b2, b3, b4, b5, b6, b7, b8, b9, b10) alors // (au max) on vérifier la satisfaction des 10 contraintes globales.

```
{
```

```
SI size (topK_compo)< k // remplis la composition jusqu'à k
```

```
Add (compo,TopK_compo) // vérifiait les compositions selon la fonction objectif
```

```
}
```

III. 2. 2. Les fonctions utilisées :

«**c.g.s**» : Elle calcule la médiane pour chaque composition et compare avec les contraintes globales.

III. 2. 3. Exemple d'application :

Étant donné qu'on a deux classes et que chaque classe a deux services et que ces derniers ont deux qualités de services chacun, on va prendre une valeur pour chaque quatre (4) instances.

Les classes :	1				2			
Les services :	X		Y		S		T	
Les instances :	Réputation	Fiabilité	Réputation	Fiabilité	Réputation	Fiabilité	Réputation	Fiabilité
t1	0,3	0,3	0,2	0,1	0,5	0,2	0,3	0,5
t2	0,2	0,5	0,3	0,4	0,5	0,3	0,2	0,5
t3	0,6	0,3	0,3	0,6	0,7	0,3	0,4	0,7
t4	0,5	0,6	0,6	0,6	0,8	0,6	0,6	0,4

Tableau 4 : la variation de QoS des services web.

Calcul de la médiane

1				2			
X		Y		S		T	
Réputation	Fiabilité	Réputation	Fiabilité	Réputation	Fiabilité	Réputation	Fiabilité
0,4	0,4	0,3	0,5	0,6	0,3	0,35	0,5

Tableau 5. La Médiane pour chaque service.

Nous avons déjà calculé les médianes des qualités de service. Nous allons ensuite faire la somme des différentes qualités de service par paire puis nous les comparons avec les contraintes globales.

- Si dessous nous, avons la première composition de (Y, S) :
 - Médian réputation(Y) + Médian réputation(S)=0,3+0,6=0,9 > 0,8.
 - Médian fiabilité (Y) + Médian fiabilité(S)=0,5+0,3=0,8 < 0,9.
- o On peut voir que la première composition (Y, S) n'est pas réalisable vu qu'une contrainte globale a été violée.

- La deuxième composition consiste de (Y, T) :
 - Médian réputation(Y) + Médian réputation(T)=0,3+0,35=0,65<0,8.
 - Médian fiabilité (Y) + Médian fiabilité(T)=0,5+0,5=1 >0,9.
- o Comme on peut remarquer que la composition de Y et T n'est pas réalisable vu qu'une contrainte globale a été violée.

- La troisième composition consiste de X et S:
 - Médian réputation(X) + Médian réputation(S)=0,4+0,6=1>0,8.
 - Médian fiabilité (X) + Médian fiabilité(S)=0,4+0,3=0,7 < 0,9.
- o Comme on peut le remarquer la composition de X et S n'est pas réalisable vu qu'une contrainte globale a été violée.

- La composition X et T :
 - Médian réputation(X) + Médian réputation(T)=0,4+0,35=0,75<0,8.
 - Médian fiabilité (X) + Médian fiabilité(T)=0,4+0,5=0,9 = 0,9.
- o On peut voir que la composition (X, T) n'est réalisable car les contraintes globales sont violées.

✚ Donc on peut conclure que tout les compositions ne sont pas réalisables.

IV. Présentation du prototype :

IV.1.Outils et environnement de développement :

-les outils utilisés pour l'implémentation de notre application sont :

➤ Langage JAVA

Java est un langage de programmation et une plate-forme informatique qui a été créé par Sun Microsystems en 1995.

- Notre choix du langage java a été guidé par les principaux qu'offre la programmation orientée objet
 - o génère des applications portables,
 - o un langage simple et puissant,
 - o un langage qui introduit directement la notion de thread,
 - o une API très riche,
 - o une gestion automatique de la mémoire.
 - o diminue les risques de panne du programme.
 - o Java utilise des processus qui augmentent les performances des entrées/sorties.
 - o facilite l'internationalisation.

- o ne laisse pas la possibilité de mal utiliser la mémoire [28].

➤ **Netbeans**

Notre logiciel est écrit en Netbeans version 8.2 sous Windows. Le choix de la programmation sous Windows a été pris à cause de l'interface graphique qu'offre cet environnement; le choix de Netbeans a été fondamental puisqu'il est un logiciel permettant principalement le développement en java. Il fournit un environnement standard de développement pour créer des interfaces très puissantes.

➤ **JDOM**

Java Document Object Model, est une bibliothèque open source pour manipulation des fichiers XML en Java. Elle intègre DOM et SAX, et supporte XPath et XSLT.

Elle utilise des analyses syntaxiques externes pour construire les documents, permettant principalement de vérifier les données contenues dans les éléments respectent la norme XML [29].

IV.2.Présentation de l'application :

Nous allons présenter les différentes étapes de l'application, bien précisément les Interfaces. Elle est conçue de manière simple, compréhensible et facile à utiliser. Par la suite, nous donnons quelques captures d'écran qui montrent les différentes parties, composantes et fenêtres de notre application :

Commençant par la fenêtre principale de l'interface, cette première fenêtre nous donne une présentation générale sur le système.



Figure 10. Fenêtre d'accueil

La deuxième fenêtre concerne notre base des services web, elle permet d'effectuer plusieurs traitements :

Il faut charger les deux bases en utilisant le bouton « open XML files», il permet de configurer les deux fichiers XML «datasetsrv1.xml, datasetsrv2.xml » la première et la deuxième sont constitués de 500 services qui ont 7 instances chacun et pour chaque instance a 10 qualités de service.

Ensuite, on choisit premièrement le nombre de services utilisé (N) {100 ou 500}

Puis le nombre de qualités service voulue (D) soit : {4, 6, 8, 10}

Après le nombre des instances dont on a besoin (NI) soit {5 ou 7}

Enfin, on sélectionne le nombre de topk de composition a affiché {5 ou 10}

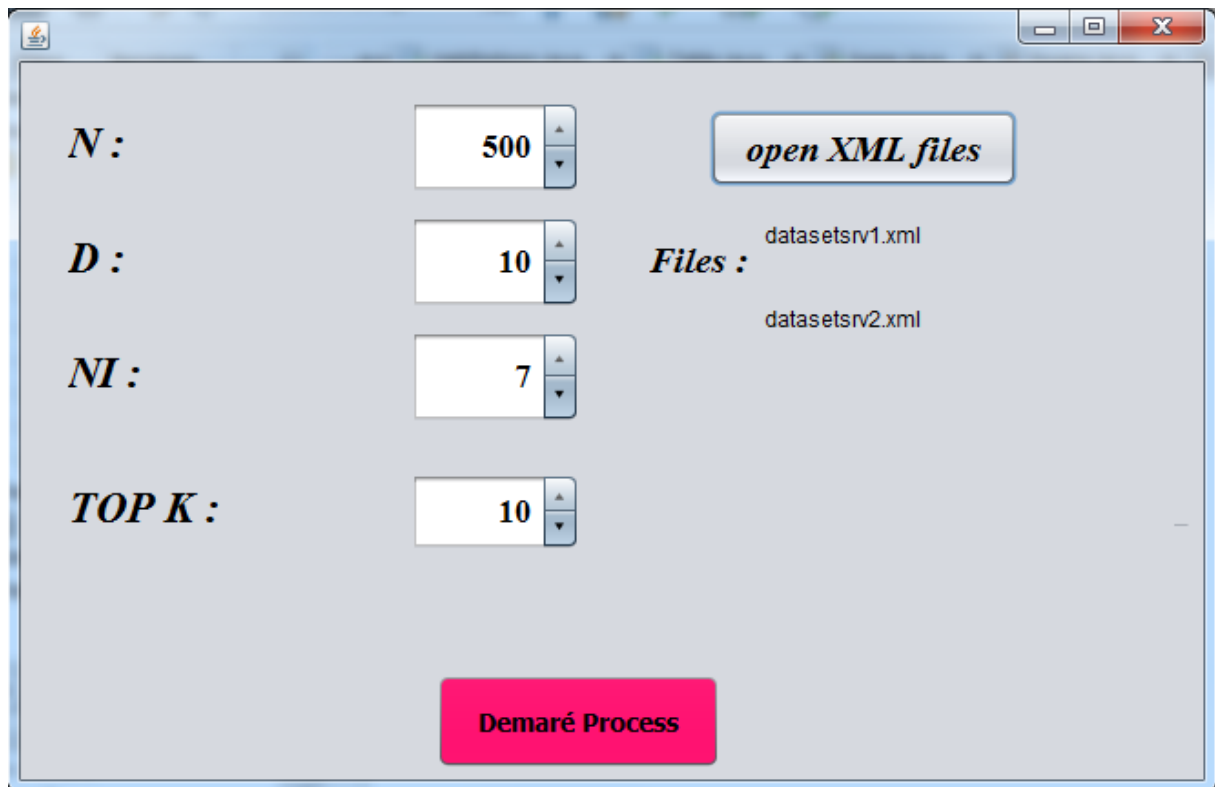


Figure 11. Lancement des compositions de service.

Une fois qu'on termine les choix (la configuration est terminée) on fait l'exécution, utilisant le bouton «démarré process» pour afficher la meilleure composition de service.

K	class1	class2	performance	Respect des contraintes
0	116	76	0.8651377178631778	1.0
1	116	71	0.8629067410725902	1.0
2	31	76	0.8523652586710391	1.0
3	417	76	0.8542123416525608	1.0
4	31	71	0.8501342818804515	1.0
5	322	76	0.8495984031837926	1.0
6	322	71	0.8473674263932051	1.0
7	417	71	0.8519813648619732	1.0
8	178	71	0.8406944502016801	1.0
9	322	29	0.8406353822606684	1.0

Figure 12: Les meilleures compositions.

V. Expérimentation:

Nous avons mené une expérience pour évaluer la performance de notre application. Le but de cette application est de démontrer pour choix de qualités de service et Topk.

Nous avons développé notre prototype sous NetBeans IDE de Sun Microsystems, la machine d'expérimentation possède les caractéristiques suivantes :

- La machine DELL INSPIRON N3521.
- Le système d'exploitation est Windows 7 (32 Bits).
- Processeur Intel® Core™i3-3217U CPU @1.80 GHz 1.80 GHz.
- 4 Go de RAM.

D	Topk	Temps d'exécution (m.s)
4	5	4 m.s
10	5	8 m.s
4	10	11 m.s
10	10	15 m.s

Tableau 6. Résultats obtenus.

VI. Conclusion :

Dans ce chapitre, nous avons présenté les algorithmes pour la sélection des services web avec des exemples bien détaillés. Ensuite nous avons proposé l'application pour illustrer notre travail. Nous avons aussi présenté les résultats de la validation de notre approche, les résultats obtenus sont testés par la base composée de service web.

Conclusion

générale

Conclusion générale :

Dans ce mémoire, nous avons présenté les technologies liées aux services Web, nous avons aussi proposé un algorithme de sélection des compositions des services web basés sur la dominance stochastique. Notre prototype sélectionne les compositions de services les plus satisfaisantes, en se basant sur les critères de QoS ($D \in \{4, \dots, 10\}$). La composition concrète recherchée doit maximiser l'ensemble des critères positifs. Concernant les perspectives de ce travail, nous envisageons la réalisation de plusieurs alternatives de classement local, telles que : le jugement majoritaire, et d'autres versions de la relation de dominance (floue, approximative...).

Références bibliographique

Références bibliographique

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

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

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

[4] : Frank Jennings, Ramesh Loganathan, et Poornachandra Sarang, Approach to Integration ; XML, Web services, ESB, and BPEL in real-world SOA projects, edition Pack tPublishing Ltd, November 2007.

[5] : R. S. Kaabi, Une approche méthodologique pour la modélisation intentionnelle des services et leur opérationnalisation, Thèse de Doctorat de l'Université Paris 1, Sorbonne, 2008.

[6] : Ulrich Duvent, Guillaume Anse, Master 2 Informatique ISIDIS 2009 -2010, Université du Littoral Côte d'Opale.

[7] : TORCHANE Zineb, MEDJADJI bouhadjar , La découverte des services web à base de Dominance Probabiliste et vote majoritaire, Master en Informatique ,Université Tlemcen, 2017.

[8] : DEHANE Aicha Djihad, KEBIR Zohra ,Evaluation des techniques de codage d'ontologies sur les performances de la composition de services Web, Master en Informatique, Université Tlemcen, 2012.

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

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

[11]: Cyrielle Lablanche, Florens Seine, Sébastien Gastaud, Les Web Services, Université de Nice-Sophia Antipolis, 2004–2005.

- [12]** : ISMAIL Adel, OUBIRA AMEZIANE, Conception d'une architecture pour la composition des services web byzantins, 2011.
- [13]** : Amrane bakhta, université d'oran,diplôme doctorat en sciences,Méthode de recherche de services web basée sur l'analyse formelle de concepts,2015.
- [14]** : Service Web (SOAP), Cours :NFE107 Urbanisation & Architecture des Systèmes d'Information, 2008-2009.
- [15]** : GHAFfour Ayoub ,TAHIR Fouzi, SELECTION DES SERVICES WEB ,Une approche à base de Skylines et Clustering Hiérarchique Ascendant, Master en Informatique, Université Tlemcen,2012.
- [16]** : Cyrielle Lablanche, Florens Seine, Sébastien Gastaud, Les Web Services, Licence d'Informatique, Université de Nice-Sophia Antipolis, 2004–2005.
- [17]**: Eric Newcomere, Understanding Web Services Xml WSDL SOAP and UDDI, edition O'Reilly, 2004.
- [18]** : Ethan CERAMI, Web Services Essentials, édition O'Reilly, février 2002.
- [19]** : ALILI Sofiane, ZIANI CHERIF Bassim, La sélection des services Web À base d'algorithme de la recherche Harmonique, Master en Informatique, Université Tlemcen, 2015.
- [20]**David Chappell et Tyler Jewell; Java Web Services; O'Reilly; March 2002.
- [21]** : ALILI Sofiane , ZIANI CHERIF Bassim, La sélection des services Web À base d'algorithme de la recherche Harmonique, Master en Informatique, Université Tlemcen, 2015.
- [22]** : Zeyneb KAMECHE, Sélection des Web Services à Base Des Essaims Particulaires.pdf, Master en Informatique, Université Tlemcen, 2011.
- [23]** : Andreas Vogel, Brigitte Kerhervé, Gregor von Ochmann, and Jan Gecsei.Distributed multimedia and qos : A survey. IEEE MultiMedia, 2(2) :10–19, 1995.
- [24]** : Daniel A. Menascé. Qos issues in web services. IEEE Internet Computing, 6(6) :72–75, 2002.

[25] : Abdulmotaleb El Saddik. Performance measurements of web Services based applications. IEEE Transactions on Instrumentation and Measurement, October 2006.

[26] : BOUROUIS Meriem , La sélection de services Web À base de l'algorithme d'abeilles , Master en Informatique, Université Tlemcen,2012

[27] : DALI YAHIA Mohammed, sélection de services web à base deQoS.pdf, Master en Informatique, Université Tlemcen,2010.

[28]: Xavier.Blanc@lip6.fr , Jerome.Daniel@der.edf.fr Le langage Java.

[29]: KARAOUZENE Meryem, Système de recommandation des services web sémantiques, Master en Informatique, Université Tlemcen,2014.

Résumé

La sélection des services web est l'une des principales étapes de l'architecture (SOA), nous constatons que les solutions existantes pour cette problématique considèrent toujours que la QoS est déterministe, de ce fait on ne peut pas gérer la nature incertaine des attributs de QoS.

Le travail proposé consiste à utiliser l'algorithme de dominance stochastique pour trouver les meilleures compositions de services web qui satisfont les contraintes globales.

Les résultats obtenus sont très satisfaisants et efficaces.

Mots-clés: sélection des services web, L'Architecture Orienté Service (SOA), QoS, l'algorithme de dominance stochastique, compositions de services web.

Abstract:

The selection of web services is one of the main steps of the architecture (SOA), we find that existing solutions for this problematic still consider that QoS is deterministic, and therefore we cannot manage the uncertain nature of the attributes of QoS.

The proposed work consists in using the stochastic dominance algorithm to find the best web service compositions that satisfy the global constraints.

The results obtained are very satisfactory and effective.

Keywords : selection of web services, Service Oriented Architecture (SOA), QoS, the stochastic dominance algorithm, web service compositions.

ملخص

يعتبر اختيار خدمات الويب إحدى الخطوات الرئيسية للهندسة المعمارية (SOA) ، ونجد أن الحلول الحالية لهذه المشكلة لا تزال تعتبر أن جودة الخدمة هي حتمية ، وبالتالي لا يمكننا إدارة الطبيعة غير المؤكدة للسمات من جودة الخدمة.

يتألف العمل المقترح من استخدام خوارزمية الهيمنة العشوائية للعثور على أفضل تراكيب خدمة ويب تلبي القيود العالمية.

النتائج التي تم الحصول عليها هي مرضية للغاية وفعالة.

الكلمات المفتاحية : اختيار خدمات الويب، الهندسة الموجهة للخدمة ، جودة الخدمة،للخدمة خوارزمية الهيمنة العشوائية، تراكيب خدمة الويب.

