L’architecture orientée services

I. Architecture orientée services

Les technologies des services web (SOAP, WSDL, UDDI, BPEL,…etc.) sont présentées comme le moyen d'implémentation des architectures orientées services. Cette architecture introduit la relation de service et les rôles de clients et de prestataires joués par les applications participantes.

L'architecture orientée services (AOS) est le terme utilisé pour désigner un modèle d'architecture pour l'exécution d'applications logicielles réparties. Dans ce modèle d'architecture, le concept service joue le rôle primaire, contrairement aux autres environnements reparties (CORBA, DCOM,…etc.) qui sont généralement des architectures par composants logiciels repartis plutôt que des architectures orientées services [**4**].

Rappelons de deux objectifs des environnements repartis, tel que CORBA, DCOM, qui sont :

- La standardisation du mécanisme d'invocation de traitements distant.
- La transparence de la localisation des composants dans un système réparti.

Le terme « service » est généralement absent de leur terminologie (sauf, par exemple, dans CORBA où l'on parle de services CORBA à propos de fonctions offertes par la plate-forme middleware aux composants applicatifs).

Construire une architecture orientée services signifie d'abord concevoir une architecture en réseau de relations de services, entre applications réparties. La description d'une relation de services est formalisée par un contrat de services. Ce contrat décrit les engagements réciproques du prestataire et du client du service.

A. Qu'est-ce qu'un service

La notion de service est le produit d'une démarche d'abstraction par rapport au logiciel qui l'implémente, au processus qui l'exécute et au port qui le localise. La réalisation d'un service passe par des messages échangés, des transitions d'état et des actions que l'application cliente suppose assurés de la part de l'application prestataire du service [**7**].

Pour faire partie du club des services il y a quatre critères à satisfaire :

Frontières explicites

Un service définit clairement ses points d'entrée. Un Service ne peut pas partager l'état avec un autre service . Ainsi, un service ne permet pas qu'on s'intègre à lui en partageant sa base de données, en appelant directement ses implémentations interne (sous forme de composants

métier ou autres), ou de quelque autre manière. Il définit clairement ses points d'entrées, et bloque toute tentative de pénétration par un quelconque autre accès.

Échanges par contrat

Les échanges se font par contrat : Un service ne donne aucune indication sur la manière dont il est implémenté, de la technologie avec laquelle il est créé, ou de la plateforme sur la quelle il est déployé. Il expose simplement un contrat qui stipule se que sont les formats de données échangés, les points d'entrée, quelles informations sont acceptée en entrée pour chaque point d'entrée, et quelles informations sont données en réponse.

Autonomie

Un service doit être autonome. Si les frontières explicites forcent déjà qu'un service ne communique pas de façon transversale avec un autre service, et qu'il ne partage aucun état avec un autre service, l'autonomie va encore plus loin. Un service doit "posséder" ses propres données. Cette possession est exclusive.

Données propres

La compatibilité est négociée en fonction de stratégies : Un service se doit de définir toutes les méthodes et protocoles d'échange qu'il permet en termes de stratégies. Si le service permet des séquences d'appels successifs, ils doivent être définis dans ces stratégies. Ainsi, un autre service, en utilisant simplement les contrats et les stratégies, peut définir s'il peut appeler ce service, et quelles méthodes et protocoles d'échange utiliser.

B. Les éléments du service

Les applications orientées services Peuvent jouer au moins l'un des deux rôles impliqués dans une relation de service [**4**] :

- **Prestataire de service** : c'est l'application qui exerce une activité dont les résultats sont directement ou indirectement exploitables par d'autres applications, éventuellement réparties sur un réseau.
- **Client** : les applications qui bénéficient de la prestation de service.

Les activités d'un prestataire de service peuvent être classées en trois groupes :

- **Informations** : c'est les résultats de l'application prestataire, qui transmis à l'application cliente sous forme de messages qui peuvent être par exemple des résultats d'un calcul complexe ;

- **Etats** : l'application prestataire gère les états et les changements d'états des ensembles de données. Les états peuvent être volatiles, persistants (les données sont stocker sur un mémoire secondaire) et durables (la possibilité de contrer les défaillances). Le changement d'état devrait être toujours durable.
- **Effets de bord** : c'est les interactions de l'application prestataire avec son environnement. par exemple des dispositifs d'entrées-sorties. Les effets de bords sont, à l'inverse des changements d'état, irréversibles par définition.

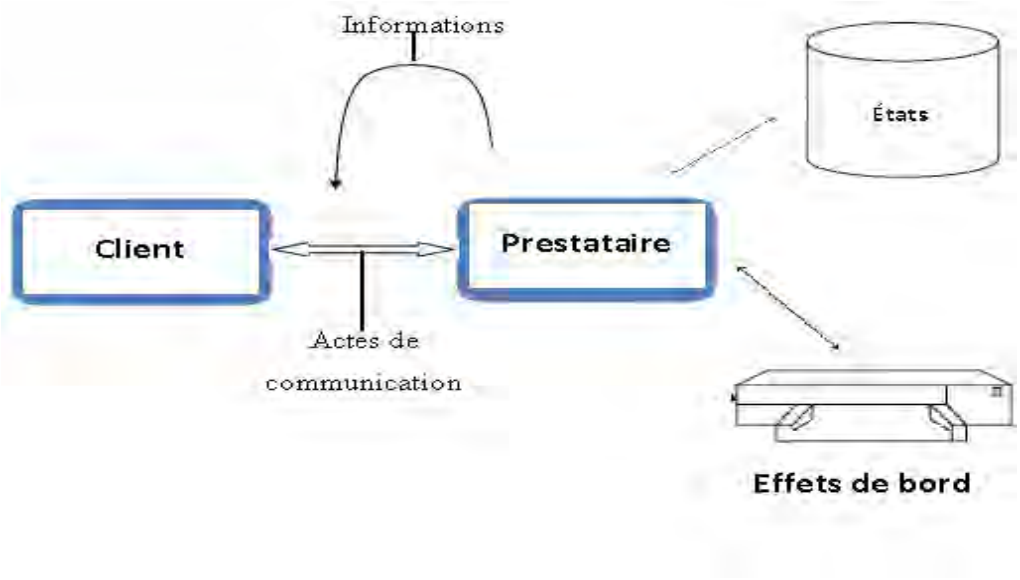

Figure 1 : Eléments d'une prestation de service.

C. Contrat de service

La description d'une relation de service est formalisée par un contrat de service. Ce contrat décrit les engagements réciproques du prestataire et du client du service. Toute application pouvant satisfaire les engagements du prestataire peut interpréter le rôle de prestataire (Idem pou le Client). Ce contrat doit être facilement extensible, aussi doit être lisible par des agents humains et exploitable par des agents logiciels (généré, interprété, agrégé, indexé, mémorisé,…, etc.). Donc le choix d'un format universel et extensible de structuration de documents comme XML s'impose.

Les éléments d'un contrat de service sont organisés en six parties [**4**] :

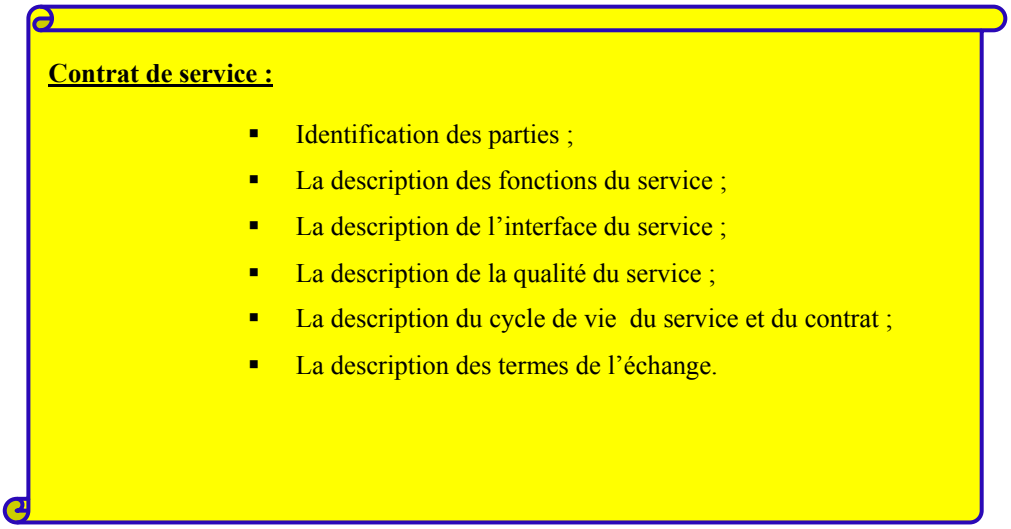

Figure 2 : Contrat de service.

- **Identification des parties** : L'identification de toutes les parties n 'est p as obligatoire. Par exemple identification du prestataire est en générale nécessaire, alors que les clients peuvent rester anonymes.
- **Fonctions du service** : Un système en générale peut être décrit en termes d'objectifs (à quoi il sert), de moyens d'interactions (comment on s'en sert), d'informations et de règles de fonctionnement du service (connaissance permettant au prestataire de service d'accomplir sa tache).

  La définition des fonctions du service est la définition de l'objet du contrat. C'est-à-dire ce que l'application prestataire fait en termes de restitution d'information, de gestion d'états, de gestion des effets de bord, d'échange d'actes de communication) et ce que les données qu'elle échange signifient [**4**].

  Dans le modèle fonctionnel, le système utilise les informations et les règles en sa possession pour sélectionner les actions qui lui permettent d'accomplir ses objectifs. Ce modèle est différent de celui de l'implémentation qui est une description technique de l'application logiciel qui met en œuvre les fonctions du service. Il faut noter qu'il est incorrect d'inclure les modèles d'implémentation des applications prestataires dans les contrats de service. L'implémentation de l'application prestataire est donc une boîte noire par rapport au contrat de service, sauf pour ce qui touche l'implémentation de la communication [**4**].

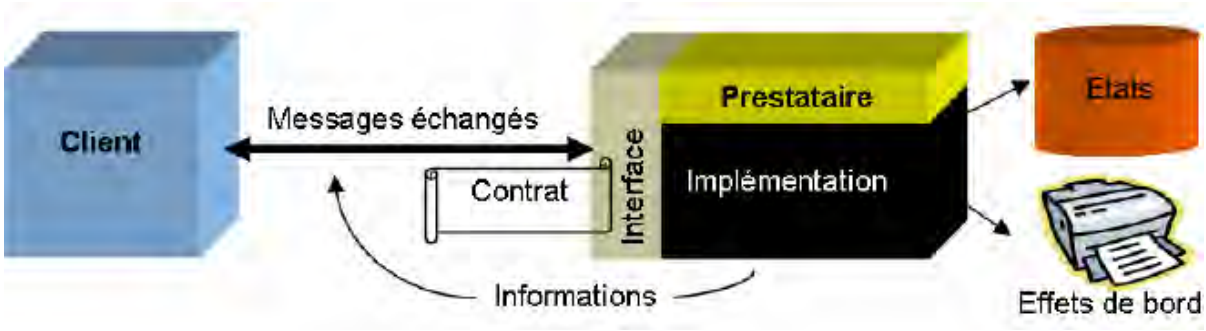

Figure 3 : Architecture boîte noire avec une interface transparente.

- **Interface du service** : Une architecture orienté services se caractérise par le fait que les interfaces de service constituent les seuls moyens de communication entre agents participants. La description de l'interface du service s'articule en deux parties :
  - **Description de l'interface abstraite** : La description de l'interface du service est donc d'abord la description des actes de communication entre le client et le prestataire dans le cadre de la réalisation de la prestation de services. L'interface abstraite qui représente l'ensemble des actes de communication entre le client et le prestataire, indépendamment des moyens utilisés.
  - Description de **l'implémentation de l'interface** : cette description comprend quatre parties :
    - **L'interface concrète** : une interface abstraite peut être implémenté par plusieurs interfaces concrètes. Ces derniers permettent de définir les différents styles d'échange et les différents formats de messages.
    - **Liaison** : une interface concrète est mise en œuvre sur une infrastructure de communication (liaison). une infrastructure de communication est constituée d'une convention de codage du contenu du message et d'un protocole de transport.
    - **Désignation des ports de réception** : Désignation du ou de s adresses des prestataires via Leurs URI et leurs n uméros de ports. A la place de la désignation des ces adresses, des mécanismes d'indirection qui permettent de découvrir ces adresses au moment de la réalisation de la prestation (par exemple un annuaire). Le langage WSDL 1.1 pe rmet de spécifier les adresses de communication via les éléments port.

Chaque port établit une association entre une liaison (implicitement, une interface) et une seule adresse de communication toujours sous format URI.

- **Chaines d'acheminement (routing) :** décrit la chaine d'intermédiaires qui joue des rôles de prestataires de services secondaires (par exemple de sécurité). Les informations nécessaires à l 'acheminement des messages de la part des intermédiaires sont notamment c ontenues dans l'en-tête, alors que l e corps du message étant destiné au récepteur final. Le chemin du m essage peut être construit de façon dynamique pendant la transmission du message.

❖ **Qualité de service** : La qualité de service est un ensemble de propriétés opérationnelles du service que l'on doit constater dans la réalisation de la prestation. Ces propriétés représentent un ensemble d'exigences concernant la mise en œuvre du service. Ces propriétés peuvent être réparties en six groupes :

- **Le périmètre de la prestation** : la partie du contrat sur le périmètre de la prestation précisent :
  - Limites de la prestation : il s'agit de préciser le caractère optionnel de certaines fonctions, de préciser explicitement les exclusions;
  - Droits et obligations du client ;
  - Conformité aux normes et aux standards : gestion des versions et la configuration de ces normes et standards qui sont par définition évolutifs.
- **La qualité de fonctionnement** : c'est un ensemble de propriétés opérationnelles qui caractérisent la réalisation des fonctions :
  - Dimensionnement : il s'agit de deux mesures de type nombre d'objets et de taille d'objets manipulés ;
  - Exactitude et précision : l'exactitude est une mesure de la dérivation de la valeur véhiculée par le prestataire par rapport à l a valeur théorique exacte (erreur standard). Alors que la précision est la mesure de la finesse de la valeur ;
  - Performance : il s'agit des mesures des critères de performances tel que le délai et le débit ;

- Accessibilité : est la probabilité d'obtenir avec succès la prestation à un instant T ;

- **La sécurité** : le contrat de service formalise les exigences de sécurité de la prestation de services (authentification, contrôle d'accès, confidentialité, intégrité, non-répudiation) ;
- **La robustesse** : c'est la résistance aux défaillances. La robustesse est l'ensemble de propriétés opérationnelles du service qui définissent par exemple:
  - Les taux d'exposition aux défaillances (fiabilité, disponibilité) du prestataire. Pour la fiabilité on peut distinguer : la fiabilité des échanges, la fiabilité fonctionnelle du service et la fiabilité des serveurs. La fiabilité de la transmission est la probabilité de transmission d'un message dans son intégralité, éventuellement dans la séquence d'émission, éventuellement sans répétition. La fiabilité fonctionnelle est une mesure de la conformité entre l'implémentation des fonctions du service de la part du prestataire et leur définition dans le contrat. La fiabilité des serveurs est une mesure de durée de service ininterrompu. la disponibilité se mesure comme la probabilité d'un prestataire d'être en service. Il faut noter que pour augmenter la disponibilité, il faut augmenter la fiabilité et diminuer le temps de rétablissement du service.

❖ **Cycle de vie du service et du contrat** : Le déroulement de la prestation de service peut être géré de manière explicite par:

- Service de gestion du cycle de vie du service primaire (activation, suspension, redémarrage, arrêt) ;
- Service de pilotage du service primaire via la modification dynamique des paramètres de la prestation ;
- Service d'interrogation de l'état du service primaire ;
- Service de journalisation des activités du service primaire ;

❖ **Description des termes de l'échange** : Décrit si le service est gratuit, Payant ou troqué. Dans un service troqué, la prestation est exécutée en échange de prestations de services compensatoires et cet échange est formalisé dans le contrat.

D. Le base de l'AOS

La base d'une AOS repose sur des services répondant, notamment, aux critères suivants :

- **Faiblement couplée** : Un service est en couplage faible quand il n'est pas autorisé à appeler directement un autre service. Il délègue cette responsabilité à un traitement spécialisé que l'on nomme fonction d'orchestration. Grâce au couplage faible, on peut réutiliser un service sans devoir reprendre des services qui lui sont liés. A partir de cette première définition, on ajoute d'autres principes techniques qui contribuent à découpler les services entre eux :
  - Un service est activable indépendamment de sa t echnologie. Pour ce faire, l'activation se réalise par l'envoi (et la réception) d'un message XML.
  - Un service peut être activé suivant un mode asynchrone. Dans ce cas, le service s'abonne à un événement auprès de la fonction d'orchestration.
- **Distribution** : Là où l'entreprise devait maintenir certaines fonctions et données comme les tarifs de fournisseurs, elle pourra interroger directement, via son applicatif, le service du fournisseur sans se préoccuper de sa mise à jour.
- **Invocation et publication** : Les services doivent être invocables et publiables quels que soient les systèmes utilisés.
- **orientation métiers** : Les services permettent de gérer les applications avec une approche fonctionnelle, par l'intermédiaire de processus métiers intégrés à l'architecture, permettant de piloter les applications sans développement conséquent.

E. L'agrégation et la dissémination de service

Une architecture orientée services peut être conçue par une approche incrémentale, résultat de la combinaison de deux démarches de base, présentées ci-dessous, qui sont [**10**] :

- **L'agrégation de services** : Plusieurs applications se r épartissent les tâches et les publient sous forme de services. Chaque service est issu de la composition (ou de l'orchestration) de services atomiques.

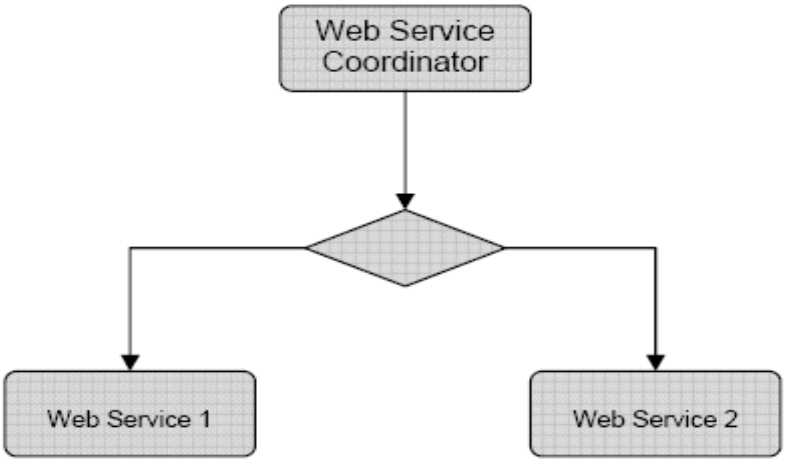

Figure 4 : L'agrégation de services.

- **La dissémination de services** : La dissémination de services permet d'effectuer la décentralisation des données. Donc on pe ut avoir plusieurs applications qui sont prestataires du même service sur des données différentes.

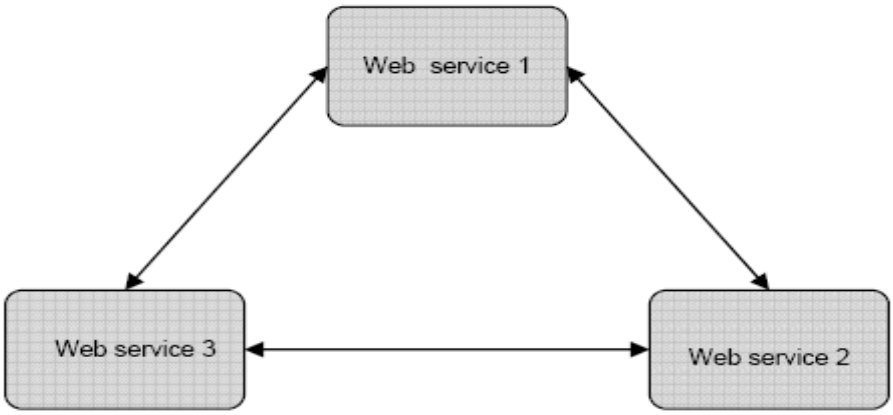

Figure 5 : La dissémination de services.

F. L'architecture orientée services && les services Web

La cible des technologies de service Web est bien les architectures orientées services. Un service Web est naturellement une application orientée services. Mais l'inverse ce n'est pas vrai. Selon la définition du W 3C, une application orientée services peut être qualifiée de service Web si elle exhibe le profil technologique suivant :

- Identification : le service Web est identifié par un URI (Uniform ressource Identifier). Un URI c'est un mécanisme permettant aux utilisateurs de localiser leurs ressources.
- Description : les interfaces et les liaisons d'un service Web sont décrites (et donc peuvent être définies et découvertes) au moyen d'un langage XML ;
- Message : un service Web communique avec les autres agents logiciels au moyen de messages au format XML ;
- Transport : les messages sont transmis via des protocoles Internet.

Identification	URI
Description	Language XML (WSDL)
Message	Format XML (SOAP, XML)
Transport	protocoles Internet (HTTP, SMTP, …etc.)

Figure 6 : profil technologique d'un service Web

Dans la mise en place des architectures de service Web, la fonction du contrat est essentiellement portée par le document SWDL. Un document WSDL permet de définir l'implémentation de l'interface, à savoir :

- Les styles d'échange ;
- Les formats de messages ;
- Les conventions de codage ;
- Les protocoles de transport ;
- Les ports de réception.

Pour approfondir la notion d'architecture des services Web, voyons la façon dont cette architecture est mise à contribution. La figure suivante illustre une utilisation typique des composants de cette architecture :

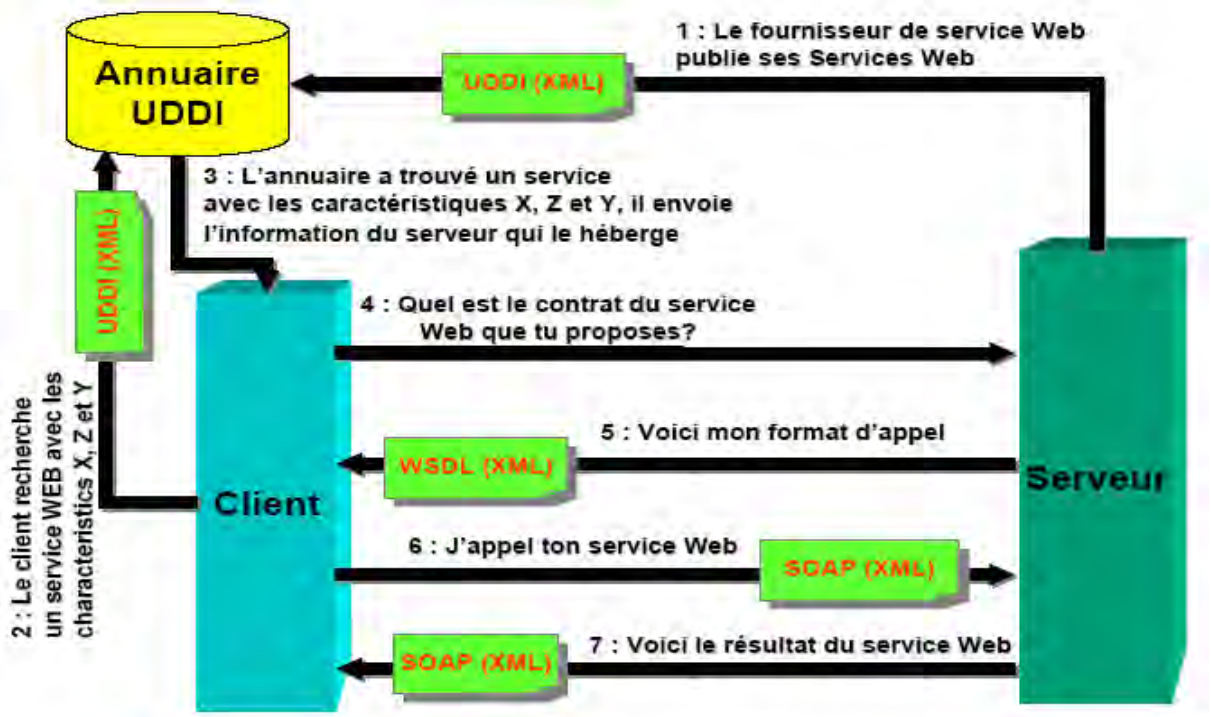

Figure 7 : Architecture générale de Service Web.

1. Tout d'abord, le fournisseur crée le service Web. Ce service doit s'inscrire auprès d'un référentiel UDDI en indiquant plusieurs de ses caractéristiques (l'interface ainsi que les informations d'accès au service), y compris sa description WSDL ;
2. L'application cliente va consulter l'annuaire de service (un référentiel UDDI) afin de sélectionner le service Web qu'elle souhaite utiliser. L'annuaire de service rend disponible l'interface du service ainsi que ses informations d'accès pour n'importe qu'elle cliente potentielle ;
3. Une fois que le service Web est sélectionné, on peut consulter sa description WSDL pour savoir comment interagir avec celui-ci. Cette description permet de connaitre quel message SOAP est à envoyé et quelle message SOAP sera reçu en retour du traitement souhaité ;
4. Il faut à présent contacter le service Web, afin de demander le contrat du service Web que le serveur (fournisseur) propose, et donc établir la communication avec ce dernier. Cette connexion à lieu lors de l'envoi du premier message SOAP ;
5. Le message reçu du coté de service Web indique le format d'appel ;