



Chapitre 02

**Architecture Orienté Service
(SOA) & Web Services**

1) Introduction :

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la démocratisation de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre applications. Notamment, l'architecture orientée service (Service Oriented Architecture, ou SOA) a été mise en avant afin de permettre des interactions entre applications distantes. Cette architecture est construite autour de la notion de service, qui est matérialisé par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service est toujours accompagné d'une description fournissant aux applications les informations nécessaires à son utilisation.

Les web services sont des technologies émergentes et prometteuses pour le développement, le déploiement et l'intégration d'applications sur l'Internet. Ils constituent la technologie de base pour le développement d'architectures orientées services. Ces architectures sont de plus en plus répandues sur le Web. Un des avantages majeurs des web services par rapport à ses prédécesseurs (CORBA, DCOM, COM, ..) est l'apport de l'interopérabilité sur l'Internet. Le principe essentiel de l'approche web service est de transformer le Web en un dispositif distribué d'échange et de calcul, où les web services peuvent interagir d'une manière intelligente. Actuellement, de nombreuses infrastructures pour supporter des web services, sont déployées par différents organismes. La diversité de ces infrastructures et des organismes qui les déploient entraîne des hétérogénéités.

2) Qu'est-ce que Architecture Orientée Service?

L'architecture orientée services (SOA) est une approche architecturale permettant la création des systèmes basés sur une collection de services développés dans différents langages de programmation, hébergés sur différentes plates-formes avec divers modèles de sécurité et processus métier. Chaque service représente une unité autonome de traitement et de gestion de données, communiquant avec son environnement à l'aide de messages. Les échanges de messages sont organisés sous forme de contrats d'échange. L'idée maîtresse de l'architecture orientée service est que tout élément du système d'information doit devenir un service identifiable, documenté, fiable, indépendant des autres services, accessible, et réalisant un ensemble de tâches parfaitement définies. [R.Ben Halima,2009]

SOA est une architecture, pas une technologie, une méthode de conceptualisation, la conception, L'objectif de la SOA est de permettre aux entreprises d'étendre la fonctionnalité et la durée de vie de leurs actifs informatiques existants, de réduire la complexité architecturale, de diminuer le dédoublement des services et des données, et accroître la flexibilité et l'agilité des entreprises pour répondre aux changements du marché. SOA est un moyen de moulage technologies de l'information autour des besoins de l'entreprise, au lieu de moulage de l'entreprise autour d'elle.

Les principaux avantages d'une architecture orientée services sont :

- ✓ Une modularité permettant de remplacer facilement un composant (service) par un autre.
- ✓ Une réutilisabilité possible des composants (par opposition à un système tout-en-un fait sur mesure pour une organisation).
- ✓ De meilleures possibilités d'évolution (il suffit de faire évoluer un service ou d'ajouter un nouveau service).
- ✓ Une plus grande tolérance aux pannes.
- ✓ Une maintenance facilitée.

L'architecture orientée services est une réponse très efficace aux problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes qui implémentent leurs systèmes d'information. Les architectures SOA ou AOS ont été popularisées avec l'apparition de standards comme les Web Services dans l'e-commerce (commerce électronique) (B2B, inter-entreprises, ou B2C, d'entreprise à consommateur), basés sur des plateformes comme J2EE ou .NET. Elles mettent en application une partie des principes d'urbanisation. Au sein de l'architecture orientée services, on distingue les notions d'annuaire, de bus, de contrat et de service, ce dernier étant le noyau et le point central d'une architecture orientée services. [J.Bonnel,2008]

3) Historique de l'Architecture Orienté Service.

Au cours de la décennie 1980-1990, la problématique de l'interopérabilité des systèmes d'information, particulièrement complexe lors de la fusion ou de l'acquisition d'entreprises, a donné naissance au domaine de recherche de l'interopérabilité des données. Ces recherches aboutissent aux systèmes multi bases, aux développements des bases de données réparties et à l'architecture fédérée. En raison de nombreux problèmes insolubles, l'architecture fédérée fut pratiquement abandonnée.

Ce terme est apparu au cours de la période 2000-2001 et concernait à l'origine essentiellement les problématiques d'interopérabilité syntaxique en relation avec les technologies d'informatique utilisées en entreprise. Cette conception a évolué pour désigner maintenant le sous-ensemble particulier d'architecture de médiation en fonction de la technologie disponible.

Dans la vie de tous les jours, un fournisseur offre un service à un client le consommant dans une relation de confiance établie entre les deux parties. En général, le client s'intéresse uniquement au résultat produit du service sans avoir le besoin ni le souci de savoir comment ce dernier est obtenu. L'architecture SOA suit ce même principe. Le service est une action exécutée par un « fournisseur » (ou « producteur ») à l'attention d'un « client » (ou « consommateur »), cependant l'interaction entre consommateur et producteur est faite par le biais d'un médiateur

responsable de la mise en relation des composants. Le service étant à grandes mailles, il englobe et propose les fonctionnalités des composants du système. Ces systèmes peuvent aussi être définis comme des couches applicatives.

L'architecture orientée services est une réponse très efficace aux problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes qui implémentent leurs systèmes d'information. Les architectures SOA ou AOS ont été popularisées avec l'apparition de standards comme les Web Services dans l'e-commerce (commerce électronique) (B2B, inter-entreprises, ou B2C, d'entreprise à consommateur), basés sur des plates-formes comme J2EE ou .NET. Elles mettent en application une partie des principes d'urbanisation. Au sein de l'architecture orientée services, on distingue les notions d'annuaire, de bus, de contrat et de service, ce dernier étant le noyau et le point central d'une architecture orientée services. La déclinaison ou plus précisément la mise en œuvre de la SOA qui repose entièrement sur Internet est appelée la WOA (Web Oriented Architecture).

4) Caractéristiques de SOA :

L'approche SOA confère aux entreprises davantage de flexibilité dans leur activité en améliorant les applications et l'infrastructure informatique. Elle contribue à réduire les dépenses informatiques. Elle fournit un accès rapide à des informations plus précises et permet à l'entreprise d'identifier et de résoudre plus efficacement les problèmes de flux. Parmi ces caractéristiques, on peut citer les plus primordiales suivantes :

- Les services SOA ont auto-description des interfaces dans la plate-forme indépendante des documents XML. Web Services Description Language (WSDL) est la norme utilisée pour décrire les services.
- Les SOA services sont communiquent avec des messages formellement définie par un schéma XML. La communication entre les consommateurs et les fournisseurs de services ou se produit généralement dans des environnements hétérogènes, avec peu ou aucune connaissance sur le fournisseur.

Les messages entre les services peuvent être considérés comme des documents clés de l'entreprise traitées dans une entreprise.

- Les services SOA sont maintenus dans l'entreprise par un registre qui agit comme un répertoire. Les applications peuvent consulter les services dans le registre et appeler le service. Universal Description, Définition, et de l'intégration (UDDI) est la norme utilisée pour le registre de service.
- Chaque service SOA possède une qualité de service (QoS) qui lui est associée. Certains des éléments clés de qualité de service sont les exigences de sécurité, telles que l'authentification et l'autorisation, la messagerie fiable, et les politiques concernant les personnes qui peuvent invoquer des services.

5) Qu'est-ce qu'un Service ?

Un service est une fonction logicielle autonome qui accepte des requêtes et qui renvoie des réponses à travers une interface standard bien définie. Un service est donc une unité de traitement qui fournit un résultat à un consommateur. Fournisseurs et consommateurs sont habituellement des agents logiciels qui agissent par délégation de leurs propriétaires. Les services ne doivent pas dépendre de l'état d'autres fonctions ou d'autres traitements externes. Les technologies employées pour réaliser un service comme le langage de programmation ne font pas partie de la définition d'un service. Dans une architecture orientée services, tous les messages ou toutes les requêtes pour un service spécifique sont envoyés à l'adresse unique du service.[J-Paul Figer,2005]

Les principales caractéristiques d'un service sont décrites comme suit :

- La communication avec un service est basée sur un ensemble de standards et protocoles ouverts.
- Il est localisé à travers une adresse.
- Il possède une définition précise pour répondre aux besoins des clients.
- Il peut s'exécuter d'une manière synchrone ou asynchrone.
- Il possède un propriétaire (fournisseur de service).

- Il met en œuvre une grande flexibilité en fournissant une granularité large des opérations.

6) Implémentation de SOA :

La mise en œuvre d'une architecture SOA nécessite des services d'intégration permettant d'organiser les échanges entre applications, et plus particulièrement en termes de conversion de données. Les langages de description de données comme XML sont utilisés pour représenter les données à échanger.

L'industrie informatique propose des solutions technologiques pour simplifier et rendre efficace la mise en œuvre des systèmes d'information à base de service. Les premiers Web services sont apparus au début des années 2000. Les Web services peuvent être utilisés pour résoudre les problèmes en termes d'interopérabilité entre les systèmes d'informations des entreprises. Les solutions basées sur les web services ont adopté les solutions d'intégration basées sur l'approche SOA. L'adoption des Web services s'est généralisée et est devenue un standard reconnu dans le contexte de E-business. De nombreux éditeurs ont déjà adopté les principes des architectures orientées services (SOA) avec les standards de Web services. Ces nouvelles technologies ont facilité l'usage de l'Internet comme une infrastructure de collaboration intra et interentreprises. Donc les Web Services présenter comme des composants de base pour la SOA. [M.JABER,2009]

7) Web Services :

7.1 Qu'est-ce qu'un Web Services ?

La technologie des Web services est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les Web services sont basés sur le modèle SOA. D'autres technologies telles que RMI, DCOM et CORBA ont précédemment adopté ce style architectural mais ont généralement échoué en raison de la diversité des plates-formes utilisées dans les organisations et aussi parce que leur usage n'était pas adapté à Internet. Les applications réparties fondées sur ces technologies offrent des solutions caractérisées par un couplage fort entre les objets. Les solutions proposées par les Web services, permettent néanmoins un couplage moins fort. De plus, l'utilisation des technologies standards du Web telles HTTP et XML par les Web services facilite le développement d'applications réparties sur Internet, et permet d'avoir des applications très faiblement couplées. L'intégration est sans doute le facteur essentiel qui favorise l'utilisation des Web services.

On retrouve plusieurs définitions des web services :

Citation : W3C

Un web service est un composant logiciel identifié par une URL, dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les web services peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet.

Citation : Dico du Net

Une technologie permettant à des applications de dialoguer à distance via Internet indépendamment des plates-formes et des langages sur lesquels elles reposent.

7.2 Historique des Web Services :

Les Web services sont nés de l'effort de plusieurs organisations qui ont partagé un intérêt commun en développant et en maintenant "un marché électronique". Celles-ci souhaitaient pouvoir communiquer plus simplement et sans avoir à se concerter sur chacune de leur transaction pour pouvoir interpréter leurs différentes données. Elles souhaitaient supprimer l'isolement de leur système informatique avec les autres.

Vers la fin des années 80, l'évidence fut que l'âge des systèmes informatiques isolés touchait à son terme tandis que différents ordinateurs, de tailles, de capacités et de formes variées, apparaissaient au sein d'une même organisation. Les départements informatiques voulurent bien évidemment exploiter au mieux et au plus bénéfique la précieuse puissance d'analyse qu'ils avaient à disposition. Il fallut donc rendre les applications informatiques capables de déplacer leurs travaux, c'est-à-dire de procéder à un véritable traitement distributif.

Pour répondre à cette nouvelle situation, De nombreuses technologies ont alors émergé depuis avec toujours cet objectif de connecter des logiques métiers au travers d'un réseau, C'est la naissance de CORBA, DCOM, Unix RPC, Java RMI, mais aucune de ces technologies n'a réellement réussi à s'imposer car souvent rattaché à un système d'exploitation, à un éditeur ou à un langage particulier.

Aujourd'hui, les web services provoquent un intérêt certain auprès des architectes et des décideurs. Dès à présent, les Web Services sont sortis du champ des échanges interentreprises pour s'adapter à celui du référencement et de la mise à disposition des ressources de l'entreprise, empiétant en ce sens sur les technologies de type EAI. Cette utilisation à elle seule prouve la qualité du modèle et sa pérennité, notamment au niveau des couches les plus basses. Par contre, la normalisation complète d'une architecture distribuée construite sur les Web Services n'est pour l'instant pas encore tout à fait établie.

7.3 Architecture des Web Services :

L'architecture classique de référence des Web Services se fonde sur le modèle SOA (Service Oriented Architecture) [Nickul et al., 2005] qui fait intervenir trois acteurs : un client, un fournisseur ainsi qu'un intermédiaire jouant le rôle d'annuaire (Figure .II.01).

- A. **Le fournisseur de services** : correspond au propriétaire d'un service. Il correspond à l'entité qui réalisera effectivement le service demandé. D'un point de vue technique, il est constitué par la plateforme d'accueil du service.
- B. **Le client** : correspond au demandeur d'un service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application client peut être elle-même un Web Service qui sert alors l'intermédiaire aux utilisateurs de services.
- C. **L'annuaire des services** : correspond à un registre de description de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de service à l'intention des clients. Il offre aux fournisseurs la capacité de publier leurs services et aux clients le moyen de localiser les services répondant à leurs besoins.

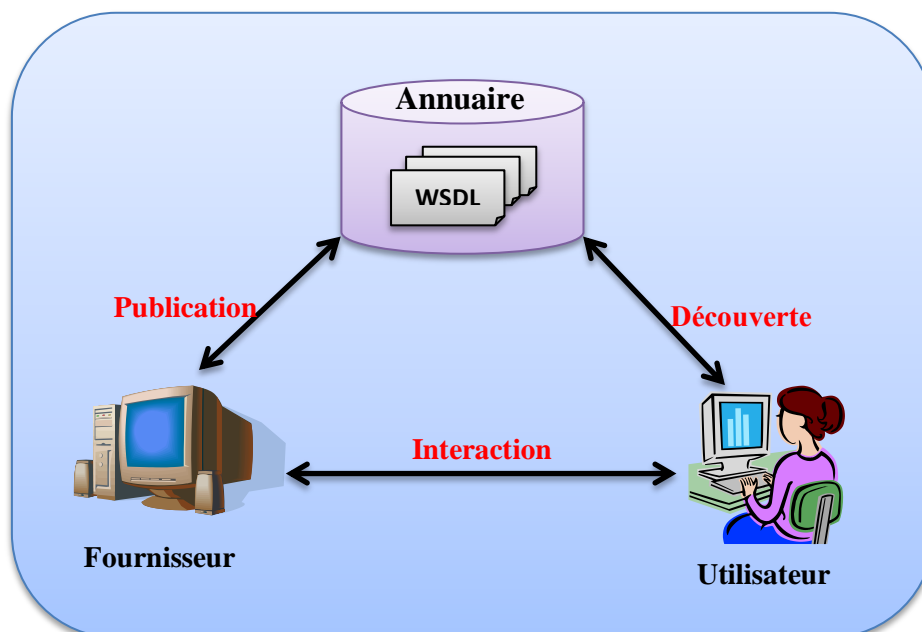


Figure .II.01 : Architecture classique des Web Services.

7.4 Scénario générale de fonctionnement des Web Services :

Le scénario classique du processus d'invocation d'un web service est décrit comme suit :

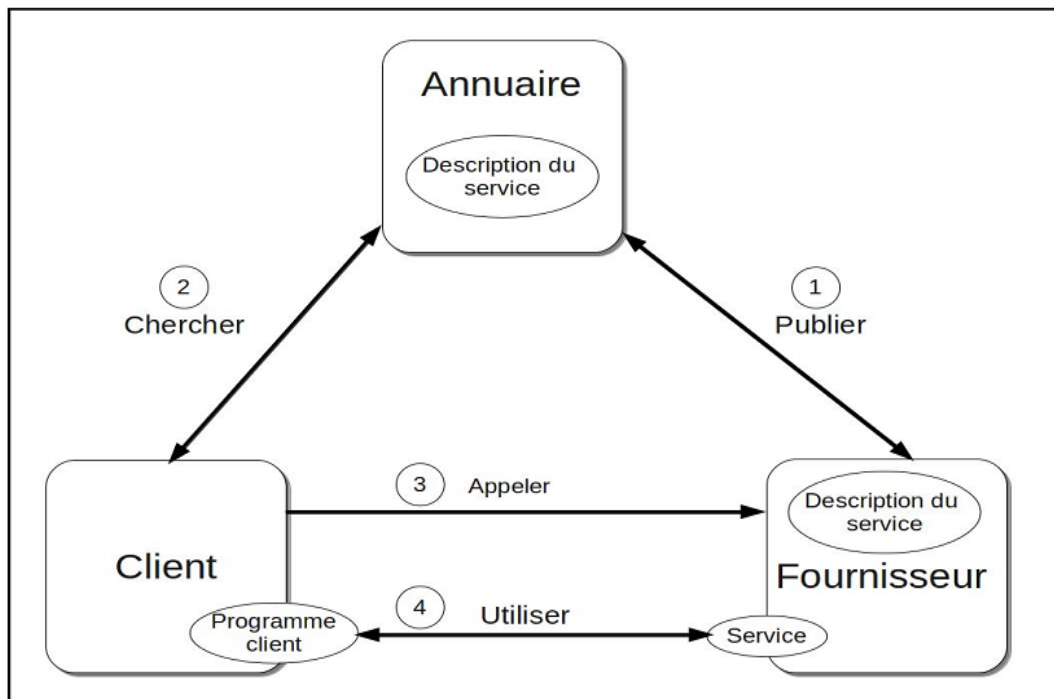


Figure .II.02 : Processus d'invocation d'un web service.

1. Le fournisseur de web service déploie son service et le rend accessible à travers internet. Ensuite, il le publie dans un annuaire en décrivant ses caractéristiques dans un fichier WSDL.
2. Ensuite, le client du web service envoie une requête SOAP à l'annuaire de service en spécifiant ses propres exigences de recherches. L'annuaire cherche les services demandés selon les critères spécifiés et retourne au client tous les URL des services disponibles.
3. Une fois le client sélectionne le service voulu, il établit un contrat avec le fournisseur pour invoquer le web service.
4. Le client interagit à partir de son application avec le web service à travers des requêtes SOAP en spécifiant les méthodes à appeler. [W.SELLAMI, 2011]

7.5 L'intérêt d'un Web Service .

Les Web services fournissent un lien entre les applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde.

Les Web services sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des Web services puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les Web services n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les Web services ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme CORBA qui définit des ponts spécifiques entre le langage de définition IDL et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.

Les Web services représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

7.6 Les technologies standards des Web Services :

Le terme Web Services regroupe un ensemble de technologies basées sur XML, permettant de créer des composants logiciels distribués, de décrire leurs interfaces et de les utiliser indépendamment du langage d'implémentation choisi et de la plate-forme d'hébergement. SOAP, UDDI, WSDL ou dernièrement Rest sont les technologies qui rendent possibles la construction et la publication de tels services.

7.6.1 XML (eXtensible Markup Language)

XML est un langage permettant la représentation de données ainsi que de documents structurés sans l'utilisation de balises prédéfinies. Ainsi, ce langage peut être étendu de façon à y ajouter des balises spécialisées afin de décrire au mieux chaque type de données. Cette flexibilité confère à XML la popularité dont il jouit.

Historiquement, XML a été développé par le W3C en 1996, a profité des meilleurs aspects de SGML 26 et est, depuis 1998, une recommandation du W3C. [W3C, 2006]

XML constitue la technologie de base des architectures Web services c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet. En effet, il apporte à l'architecture des Web services l'extensibilité et la neutralité vis à vis des plates-formes et des langages de développement.

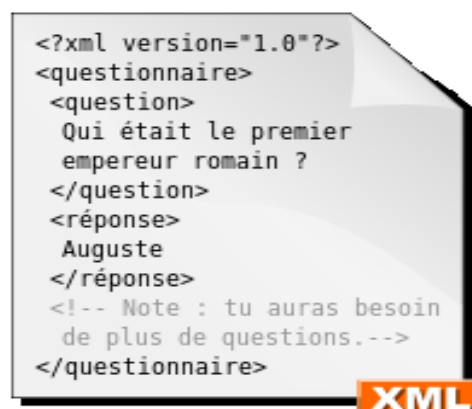


Figure .II.03 : Documents XML.

La technologie des Web services a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des Web services est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type). [T.MEL,2004]

7.6.2 SOAP (Simple Object Access Protocol):

SOAP est appelé aussi Service Oriented Access Protocol, est un protocole de la famille XML servant à l'échange d'informations dans un environnement distribué et décentralisé. Il est considéré comme la technologie la plus importante des Web Services. Le standard SOAP a été proposé au W3C par Microsoft, IBM, Lotus. [G.Babin et M.Lebanc, 2003]

SOAP assure l'interopérabilité entre composants tout en restant indépendant des systèmes d'exploitation et des langages de programmation, donc, théoriquement, les clients et serveurs de ces dialogues peuvent fonctionner sur n'importe quelle plate-forme et être écrits dans n'importe quel langage à partir du moment où ils peuvent formuler et comprendre des messages SOAP. Il représente donc un composant de base pour développer des applications distribuées, qui exploitent des fonctionnalités publiées comme services par des intranets ou Internet. [S.Bouchra, 2010]

Contrairement aux autres protocoles, IIOP pour CORBA, ORPC (Object RPC) pour DCOM, ou JRMP (Java Remote Method Protocol) pour RMI qui sont des protocoles binaires, SOAP se base sur XML pour encoder les données. Les messages échangés via ce protocole jouissent donc des avantages que lui procure le langage XML pour structurer les données. [R.Ben Halima,2009]

Structure d'un message SOAP :

La grammaire de SOAP est assez simple à comprendre. Elle procure un moyen d'accès aux objets par appel de 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.

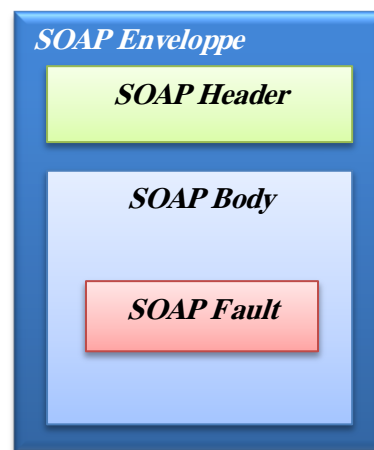


Figure .II.04 : Structure d'un message SOAP.

- 1) **L'enveloppe SOAP** : sert de conteneur aux autres éléments du message SOAP, elle est définie au début par la balise <soap:Envelope> et se termine par la balise </soap:Envelope>. Les messages SOAP ne peuvent pas être envoyés en lots, autrement dit l'enveloppe contient un seul message constitué d'un entête facultatif (SOAP header) et d'un corps obligatoire (SOAP body).
- 2) **L'entête SOAP (Header)**, est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.)

- 3) **Le corps SOAP (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 l'enveloppe. Ce bloc contient les données transportées par le message SOAP qui doit voir tous ces sous-éléments correctement qualifiés par des espaces de nom. Il doit contenir, en envoi, le nom de la méthode appelée, ainsi que les paramètres appliqués à cette méthode. En réponse, il contient soit un appel de méthode, soit une réponse à sens unique, ou finalement un message d'erreur détaillée.
- 4) **SOAP fault (erreur)** : est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs.

La figure .II.05 montre un exemple de message SOAP requête d'un web service qui additionne deux entiers.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <soapenv:Header/>
- <soapenv:Body>
- <Addition xmlns="http://doc">
  <a>4</a>
  <b>7</b>
</Addition>
</soapenv:Body>
</soapenv:Envelope>
```

Figure .II.05 : Le formatage visuel d'un message SOAP.

7.6.3 WSDL (Web Services Description Language).

A ce stade, dans la pile des couches des web services, une problématique naît d'un point de vue de la description des services puisque le protocole SOAP ne s'occupe que de l'échange de messages, comment l'utilisateur du service connaît-il les paramètres, les noms de méthodes de services distants et le type de réponse ?

La solution à cette problématique a été apportée par les sociétés Ariba, IBM et Microsoft, qui ont proposé la spécification WSDL permettant la description de web services et en particulier de leur interface au moyen de documents WSDL.

WSDL C'est un format de schéma XML permettant de décrire un web service en précisant les méthodes disponibles, les formats des messages d'entrée et de sortie et comment y accéder. Il définit un espace de travail extensible pour décrire des interfaces de Web services.

Comme les autres technologies XML, WSDL est tellement extensible et possède un si grand nombre d'options qu'assurer une compatibilité et une interopérabilité entre les différentes implémentations serait difficile. Si l'expéditeur et le destinataire d'un message peuvent partager et comprendre le même fichier WSDL de la même façon, alors l'interopérabilité sera assurée.

Structure d'un document WSDL

Un document WSDL est un document XML qui est constitué des éléments suivants :

- **Définition:** elle contient le nom du service qui est décrit dans le document ainsi que les namespaces utilisés. Il s'agit de l'élément Root du document.
- **Types:** il s'agit de la définition des types de données qui seront utilisés dans les messages. On y trouvera donc par exemple des schémas XML.
- **Messages:** il s'agit d'une description de la structure des messages qui seront échangés. On y retrouve deux types de messages : les messages entrants et les messages sortants. Chaque message peut être composé de plusieurs parties décrivant les différents paramètres de ces derniers.

- **PortTypes**: il s'agit d'un ensemble d'opérations, chacune se référant à un message entrant et sortant.
- **Bindings**: ils spécifient quels protocoles seront utilisés.
- **Service**: il s'agit des informations permettant la localisation du service.

7.6.4 UDDI (Universal Description Discovery and Integration)

L'UDDI est considéré comme un des annuaires universels les plus connus, créé à l'initiative d'Ariba, IBM et Microsoft et hébergé par OASIS. UDDI est conçu pour héberger des informations sur les entreprises et leurs Web services de façon structurée. Au travers de l'UDDI, il est possible de publier et de découvrir des informations sur une entreprise et ses Web services.

L'UDDI est un standard très récent (2000) qui propose une architecture d'appel dynamique aux Web services. Cette architecture est basée sur un annuaire situé quelque part sur Internet.

Structures de données UDDI :

Un registre UDDI se compose de quatre types de structures de données, le *businessEntity*, le *businessService*, le *bindingTemplate* et la *tModel*. Cette répartition par type fournit des partitions simples pour faciliter la localisation rapide et la compréhension des différentes informations qui constituent un enregistrement.

- **BusinessEntity (entité d'affaires)** : Les « businessEntities » sont en quelque sorte les pages blanches d'un annuaire UDDI. Elles décrivent les organisations ayant publié des services dans le répertoire. On y trouve notamment le nom de l'organisation, ses adresses (physiques et Web), des éléments de classification, une liste de contacts ainsi que d'autres informations.
- **BusinessService (service d'affaires)** : Les « businessServices » sont en quelque sorte les pages jaunes d'un annuaire UDDI. Elles décrivent de manière non technique les services proposés par les différentes organisations. On y trouve essentiellement le nom et la description textuelle des services ainsi qu'une référence à l'organisation proposant le service et un ou plusieurs « bindingTemplate ».

- **BindingTemplate (modèle de rattachement)** : UDDI permet de décrire des web services utilisant HTTP, mais également des services invoqués par d'autres moyens (SMTP, FTP...). Les « bindingTemplates » donnent les coordonnées des services. Ce sont les pages vertes de l'annuaire UDDI. Ils contiennent notamment une description, la définition du point d'accès (une URL) et les éventuels « tModels » associés.
- **tModel (index)** : Les « tModels » sont les descriptions techniques des services. UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML, par exemple). C'est à ce niveau que WSDL intervient comme le vocabulaire de choix pour publier des descriptions techniques de services.

7.6.5 REST (REpresentational State Transfer) :

Qu'est-ce que REST ?

REST est une architecture de web services, à la manière de SOAP et de XML-RPC. C'est l'acronyme de *REpresentational State Transfer*. Elaboré en l'an 2000 par Roy Fielding, l'un des créateurs du protocole HTTP, du serveur Apache http et d'autres travaux fondamentaux, REST est à l'origine une tentative de décrire les principes de l'architecture du Web.

Cette architecture part du principe selon lequel Internet est composé de ressources accessibles à partir d'une URL. Par exemple, pour avoir le temps à Alger, un utilisateur pourrait utiliser une adresse de la forme <http://www.meteo.dz/alger/>. Alger serait alors une ressource telle que définie par Météo Algérie.

A la requête de cet URL serait renvoyée une représentation de la ressource demandée (alger.php, par exemple). Cette représentation place l'application cliente dans un état (state) donné.

Si l'application cliente lance un appel sur un des liens de la représentation en cours, une autre ressource est appelée, dont une représentation est envoyée. Ainsi, l'application cliente change d'état (state Transfer) pour chaque représentation de ressource. Il faut bien noter que REST n'est pas en soi un standard, il n'existe pas de

spécification du W3C pour la décrire. Il s'agit plutôt d'un style d'architecture, d'un "mode de compréhension du Web" sur lequel le développeur construit ses services (Web).

REST fait en revanche usage des standards Web : protocole HTTP, URLs, formats de fichiers pour la représentation des ressources (XML, HTML, JPEG...), types MIME pour la description de ces représentations... Le Web lui-même est d'ailleurs un système REST à part entière.

7.7 Les avantages & les inconvénients des Web Services :

Le succès des web services dans l'industrie n'est pas un hasard. En effet, les web services présentent bon nombre d'avantages.

- **L'interopérabilité** : les web services peuvent être utilisés par d'autres applications indépendamment du système d'exploitation ou des langages de programmation dans lesquels ces derniers sont implémentés.
- **Un déploiement facile et rapide** : une entreprise utilisant des web services peut mettre à disposition des nouveaux services tout en limitant les coûts ainsi que le temps nécessaire à leur déploiement. Ainsi, une entreprise peut aisément créer un nouveau service en combinant d'autres services déjà existants.
- **Protocoles et standards ouverts** : les web services utilisent des protocoles et des standards qui sont ouverts. Les données ainsi que les protocoles étant au format texte, ils sont plus facilement compréhensibles et lisibles par les développeurs.
- **L'utilisation du protocole HTTP** : les web services utilisant le protocole http bénéficient d'un avantage non négligeable, à savoir leur fonctionnement à travers de nombreux firewalls sans avoir à modifier les règles de filtrage.

Bien que la simplicité des web services est un avantage à certains égards, il peut aussi être un obstacle. Les web services utilisent des protocoles en texte clair qui utilisent une méthode assez verbeuse pour identifier les données. Cela signifie que les demandes de web services sont plus grandes que les demandes encodées avec un

protocole binaire. La taille extra est vraiment seulement un problème plus les connexions bas débit, ou via des connexions extrêmement occupés.

Les protocoles Web de base sont simples (HTTP et HTTPS), ils ne sont pas vraiment fait pour à long terme des séances. Typiquement, un navigateur établit une connexion HTTP, demande une page Web et peut-être quelques images, et puis se déconnecte. Dans un typique CORBA ou de l'environnement RMI, un client se connecte au serveur et pourrait rester connecté pour une période de temps prolongée. Le serveur peut envoyer périodiquement des données vers le client. Ce type d'interaction est difficile avec les web services.

Le problème avec le protocole HTTP et HTTPS quand il s'agit de web services, c'est que ces protocoles sont «apatride»-l'interaction entre le serveur et le client est généralement brève et quand il n'y a pas de données échangées, le serveur et le client n'ont aucune connaissance de l'autre.

Plus précisément, si un client fait une demande au serveur, reçoit des informations, puis se bloque immédiatement en raison d'une panne de courant, le serveur ne sait jamais ce que le client n'est plus actif. Le serveur a besoin d'un moyen de garder une trace de ce qu'est un client fait et aussi pour déterminer le moment où un client n'est plus actif. Typiquement, un serveur envoie une sorte d'identification de session au client lorsque le client accède pour la première du serveur. Le client utilise ensuite cette identification quand il fait de nouvelles demandes sur le serveur. Cela permet au serveur de rappeler tous les renseignements qu'il a sur le client. Un serveur doit normalement s'appuyer sur un mécanisme de temporisation afin de déterminer si un client n'est plus actif. Si un serveur ne reçoit pas une demande d'un client après une période de temps prédéterminée, il suppose que le client est inactif et supprime toutes les informations client, il a été tenu. Cette charge supplémentaire signifie plus de travail pour les développeurs de web services.

7.8 Web Services & Web Sémantique :

Les web services sémantiques sont à la convergence de la technologie de web services et du Web sémantique, ce sont des web services à descriptions dotées de sémantique. Actuellement les web services sont décrits par le langage WSDL qui permet de définir les opérations et les paramètres autorisés par le web service. Cela est fait en attribuant des noms aux opérations et aux paramètres, puis associer ces paramètres à des types abstraits de données (string, char,...).

Cependant, le problème avec ce type de description est que lorsqu'on veut automatiser les divers aspects liés aux web services (découverte, composition,...), l'agent manipulant les paramètres du web service ne peut pas avoir la signification de ces données. Pour l'agent logiciel c'est juste des variables dénotées contenant de l'information. A ce stade, les web services ne sont décrits qu'au niveau syntaxique. Un développeur voulant programmer une application cliente interagissant avec un web service doit tout d'abord avoir connaissance de la syntaxe de sa description, l'interpréter, puis écrire le code client conforme aux paramètres de la description du web service. Cependant, un agent logiciel ne peut lire la description d'un web service comme un humain, il peut avoir connaissance de la structure syntaxique de la description mais pas sa sémantique. Les web services sémantiques sont des web services décrits de telle sorte qu'un agent logiciel puisse interpréter les fonctionnalités offertes par ces web services.

Un agent logiciel doit être capable de lire la description d'un web service pour déterminer si le web service fournit les fonctionnalités désirées, et s'il est lui-même capable d'utiliser ce service. Pour permettre cela, la description du web service doit être complétée en information sémantique interprétable par machine. Les paramètres du web service doivent être décrits de façon qu'un agent logiciel puisse avoir connaissance de leur signification. Cela est fait par la définition de vocabulaires organisés en ontologies.

Ainsi pour qu'un agent logiciel puisse avoir connaissance de la sémantique d'une description d'un web service, il lui suffit juste d'accéder à une ontologie du domaine.

Un agent essayant d'interpréter la description du web service aura juste à utiliser l'ontologie où les annotations sémantiques sont définies. En utilisant un moteur d'inférence, l'agent peut inférer les similitudes entre la sémantique employée pour décrire le service et la sémantique connue par l'agent. La combinaison des technologies du Web sémantique à celles des Web services permettra de : Automatiser la découverte de web services, c'est à dire localisation automatique des web services qui fournissent une fonctionnalité particulière et qui répondent aux propriétés demandées par l'utilisateur. Pour pouvoir effectuer une découverte automatique, le procédé de découverte devrait être basé sur la similitude sémantique entre la description déclarative, faite par l'utilisateur, du service demandé et celle du service offert. [GeeK, 2011]

7.8.1 Moyens descriptifs proposés pour les web services sémantiques :

A. WSDL-S (web service description language-semantic).

WSDL-S est un langage de description sémantique des web services. Une description WSDL-S de web service est une description WSDL augmentée de sémantique, cette sémantique est ajoutée en deux étapes : La première étape consiste à faire référence, dans la partie définition de WSDL, à une ontologie dédiée au service à publier. La seconde étape consiste à annoter les opérations de la définition WSDL de sémantique en ajoutant deux nouvelles balises : La balise Action et la balise Contrainte.

B. OWL-S (Ontology Web Language for Services) :

OWL-S est un langage de description et une ontologie OWL (Web Ontology Language) de web services. La structuration de l'ontologie supérieure de OWL-S est motivée par la nécessité de fournir trois types d'information essentiels pour un service, à savoir : Que fait le service ? Comment fonctionne le service ? Comment accède-t-on au service ?

C. DAML-S (DARPA Agent Markup Language for Services) :

DAML-S est un langage de description sémantique des web services, il est constitué d'une ontologie lui permettant de décrire les web services. Il permet l'automatisation de nombreuses tâches à savoir : la découverte et la sélection d'un service qui consistent à mettre en correspondance un demandeur de service avec un fournisseur qui détient ce service, l'invocation de ce service qui permet de réaliser l'appel effectif du service découvert, l'inter opération et la composition de services qui autorisent la composition des services simples afin d'obtenir un service complexe qui répond aux exigences d'un client et la surveillance de l'évolution du processus qui sert à surveiller l'état d'accomplissement et de déroulement du service. DAML-S définit une ontologie particulière qui permet la description des propriétés des web services et de les rendre disponibles au monde.

7.9 Sécurité des web services :

Les web services XML vont rouvrir 70% des chemins d'attaques fermés par les pare-feu lors de la dernière décennie. Ils peuvent transporter virtuellement toutes les données utiles sur le port 80 et le pare-feu ne peut les arrêter.

Les web services apportent des bénéfices significatifs pour des applications basées sur l'Architecture Orientée Services, mais exposent des risques significatifs en termes de sécurité. Créer et gérer un environnement sécurisé pour les web services nécessitent la manipulation et la maîtrise de spécifications et standards divers et variés ainsi que des technologies et logicielles et matérielles conséquentes.

Il convient d'étudier la mise en œuvre de la sécurité pour les Architectures Orientées Services (SOA) via les quatre volets suivants :

- *La sécurité niveau Transport* : pare-feu (firewall), VPN (Virtual Private Networks), authentification basique, non-répudiation et cryptage
- *La sécurité niveau Message* : Utilisation des jetons de sécurité afin de valider l'identité du consommateur du service ou du processus, utilisation des assertions d'autorisation pour valider l'accès aux services.

- *Sécurité niveau application* : Sécuriser les composants appelés par les Web Services, EJBs, Servlets appelés via les web services.
- *Sécurité niveau Données* : Cryptage et signature des messages afin de protéger les données stockées ou transmises.
- *Sécurité niveau Environnement* : Monitoring, logging et audit afin d'identifier les problèmes qui doivent être fixés et résolus et établir des communications sûres et fiables.

L'émergence des web services pose plusieurs problématiques dont celle de la sécurité des échanges de messages entre partenaires. Dans une architecture Orientée Services, les web services peuvent exposer des processus métier sensibles qui nécessitent un traitement particulier en termes de sécurité aux deux bouts du canal de communication. En plus, les web services sont des technologies récentes, ceci implique de nouvelles vulnérabilités et attaques ou menaces.

Les web services sont utilisés dans les cas suivants :

- Intégration de systèmes point-à-point.
- Intégration d'applications entreprise
- Collaboration et partenariat Business
- E-Business
- Composition des processus métier
- Protection et ouverture des systèmes d'information
- Réduction des coûts du cycle de vie du développement et la maintenance des systèmes d'information

La sécurité est au cœur des préoccupations des entreprises pour garantir la cohérence et la pérennité des systèmes. Pour ce faire, la sécurité doit être prise en compte dès la conception des web services et ceci à tous les niveaux.

Au-delà du tout sécuritaire, une approche pragmatique de la sécurité pilotée par les risques est préférable. Rappelons les fonctions de sécurité disponibles qui pourront s'appliquer à des services:

- *Authentication* : le mécanisme qui permet de vérifier l'identité d'une personne/d'un système
- *Habilitation* : le droit d'accéder ou non à une fonctionnalité, à une donnée.
- *Intégrité (ou signature)* : la non modification d'une donnée échangée
- *Imputabilité* : traçabilité des actions d'un individu sur un système
- *Confidentialité (ou chiffrement)* : la non lisibilité de la donnée par un tiers ne partageant pas un secret.

Les stacks de sécurité des web services (WS-Security & co) sont à priori complexes et chers à mettre en œuvre. Toutefois, on peut trouver des exemples concrets où ces solutions ont un réel intérêt (signature des messages par exemple). [X.Vaccari,2008]

7.9.1 WS-Security

WS-Security est un standard pour sécuriser des web services SOAP uniquement. Ce dernier aborde les aspects classiques comme l'authentification, l'habilitation mais également des besoins plus rares comme le chiffrement, la signature. Cette norme s'applique uniquement au niveau des messages échangés et n'a aucun impact sur le protocole et le transport choisis. Il repose sur l'ajout d'éléments dans les headers/en-têtes SOAP. Ce standard appelé aussi WSS (Web Services Security) tente de normaliser les échanges sécurisés de messages et de garantir l'interopérabilité entre systèmes.

Des Profils sont disponibles pour couvrir l'ensemble des problématiques liées à la sécurité (authentification, chiffrement et signature). Le consortium OASIS vise la standardisation des spécifications WS pour permettre une meilleure interopérabilité entre les technologies.[X.Vaccari,2008]

7.10 Outils technologiques pour le développement des web services :

La disponibilité des produits sur le marché informatique et la facilité de mise en œuvre de cette technologie avec des coûts très bas encouragent de plus en plus les entreprises et les développeurs pour s'orienter vers cette nouvelle solution. Le développement, l'exploitation et le déploiement des Web services peut être réalisé en employant un simple logiciel de serveur d'application telle que :

- JAX-WS (Jax, 2007) qui constitue l'implémentation de référence de Java EE est Open Source et intégré dans GlassFish et utilisable dans d'autres environnements. Son extension WSIT (aussi appelée "Project Tango") propose une implémentation de WS-ReliableM essaging, WS-SecureConversation, WS-Trust, ...
- Tomcat (deux projets open source d'Apache Software Foundation) (Apache, 2007).
- XFire de CodeHaus offre un framework Java avec une approche différente de Axis.
- Serveurs HTTP IIS de Microsoft (avec le framework .NET).
- WebSphere Application Server d'IBM (basé sur le serveur d'Apache et la plateforme de J2EE).
- Oracle Application Serveur d' Oracle Corporation
- Bibliothèque pour les développeurs de web services en PHP NuSOAP.
- gSOAP: bibliothèque pour les développeurs de web services en C++.
- JBoss Application Server de la société JBoss. Composant du JEMS (JBoss Enterprise Middleware System) dont fait également partie le framework de persistance relationnelle Hibernate.

7.11 Conclusion

Dans ce chapitre, nous avons présenté une étude détaillée de l'émergence de l'architecture orientée service, Nous nous sommes concentrés sur la technologie de service web comme instance de ces services qui semblent ouvrir de nouvelles perspectives pour les systèmes distribués.

Dans le chapitre suivant nous présenterons la conception de systèmes par langage UML.