# Architecture orientée service

## 1.1. Introduction

Les systèmes d'information ont besoin de supporter les changements dans la gestion de l'entreprise de façon rapide et efficace, et de s'adapter au développement rapide des technologies. La majorité des systèmes d'information d'entreprise sont hétérogènes, contiennent une gamme de différents systèmes, d'applications, de technologies, et d'architectures **[01]**.

Pour gérer les problèmes liés aux changements des besoins, au développement technologique, et à l'intégration, différentes solutions ont été proposées et utilisées a travers le temps mais ces solutions ont plus ou moins échoué **[01]**. L'architecture SOA (Service Oriented Architecture) est venue combler certains vides laissés par ces technologies.

On va d'abord définir ce qu'est une architecture avant de définir ce qu'est SOA :
Une architecture logicielle décrit les composants du système et la manière dont ils interagissent **[02]**.

## 1.2. Définition

SOA est un style architectural qui permet de construire des solutions d'entreprises basées sur les services **[03]**.

Le service est une action exécutée par un **fournisseur** à l'attention d'un **client**, cependant l'interaction entre client et fournisseur est faite par le biais d'un médiateur (qui peut être un bus) responsable de la mise en relation des composants **[04]**.

**L'aspect le plus important de l'architecture SOA est qu'elle permet de séparer l'implémentation du service de son interface.**

L'architecture orientée services est une nouvelle vision pour le système informatique. Ce dernier n'est plus décrit comme un ensemble d'applications mais comme un ensemble de services. Donc, plutôt que de privilégier une architecture applicative basée sur des contraintes techniques, l'architecture orientée service (SOA) propose de découper les fonctionnalités d'une application en services métier, réutilisables dans d'autres applications. En se concentrant sur les services, les applications sont agrégées pour fournir des processus opérationnels plus riches et plus significatifs.

Les services d'une architecture SOA répondant, notamment, aux critères suivants:

- **Faiblement couplés** : les applications traditionnelles incluent dans leur code les données métiers de l'entreprise. Elles sont complètement liées aux systèmes pour lesquels elles ont été conçues. Cette contrainte implique la difficulté de toute demande de modification, qu'elle concerne l'accès aux données, les règles de gestion ou celles de présentation. Un faible couplage permet une scission des aspects métiers du code qui permettra une simple reconfiguration des processus quand les fonctions métiers évoluent.
- **Distribués** : les services qui composent les applications peuvent être physiquement répartis sur des différents systèmes dans l'entreprise, mais aussi au-delà.
- **Invocables et publiables :** les services doivent être invocables et publiables quels que soit les systèmes utilisés.

### 1.3. Les composants de la SOA

Une architecture de services (SOA) est constituée de trois (ou 4) composants primaires. Le premier est le prestataire de services (le service réel). Vient ensuite le demandeur du service, autrement dit le composant qui accède au service. Enfin, l'agence de services fournit des services de découverte et d'enregistrement.

Le paradigme "découvrir, interagir et exécuter" comme montré dans la figure 1.1, ce paradigme permet au consommateur du service (client) d'interroger un **annuaire** pour le service qui répond à ses critères. Si l'annuaire possède un tel service, alors il renvoie au client le **contrat** du service voulu ainsi que son adresse. SOA consiste en quatre entités configurées ensemble pour supporter le paradigme découvrir, interagir et exécuter **[02]**.

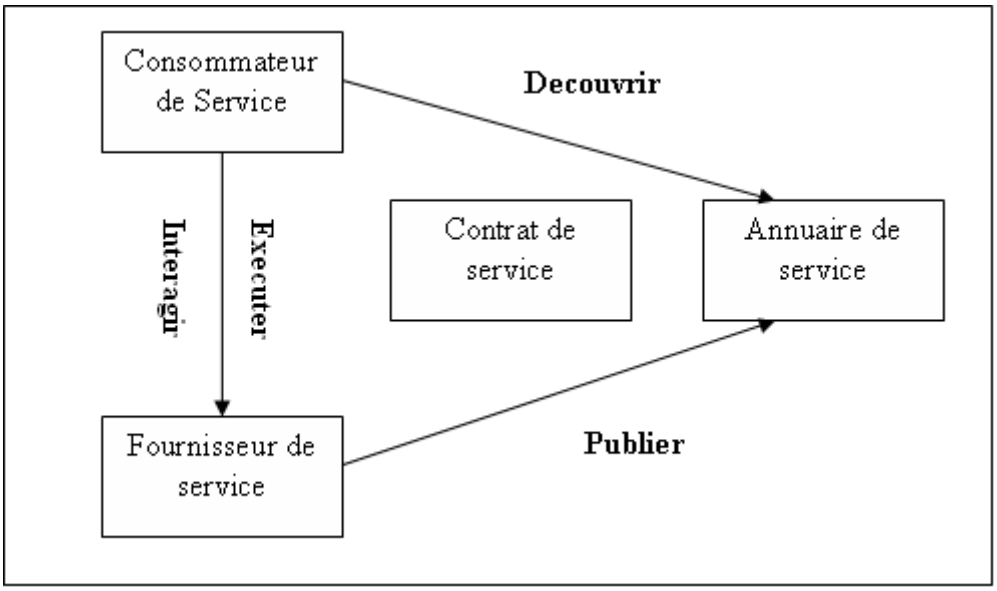



**Figure 1.1 Le paradigme "découvrir, interagir et exécuter" [02]**

#### 1.3.1. Le consommateur de service

Le consommateur de service est une application qui requière un service. C'est l'entité qui initie la localisation du service dans l'annuaire, interagit avec le service à travers un protocole et exécute la fonction exposée par le service **[02]**.

### 1.3.2. Le fournisseur de service

Le fournisseur de service est une entité adressable via un réseau, il accepte et exécute les requêtes venant d'un client **[02]**.

Le fournisseur de service publie le contrat de service dans l'annuaire pour qu'il puisse être accédé par les clients **[02]**.

### 1.3.3. L'annuaire de service

L'annuaire de service est un annuaire qui contient les services disponibles. C'est une entité qui accepte et sauvegarde les contrats du fournisseur de service et présente ces contrats aux éventuels clients **[02]**.

### 1.3.4. Le contrat de service

Le contrat spécifie la manière dont le client de service va interagir avec le fournisseur de service. Il spécifie le format de la requête et la réponse du service **[02]**.

## 2. Les web services

### 2.1. Introduction

D'après la définition, SOA est une approche architecturale qui ne fait aucune hypothèse sur la technologie de mise en œuvre. En particulier, l'amalgame souvent faite entre SOA et les web services est une erreur **[05]**.

Cependant, la conception des spécifications Web services a été menée dans l'objectif de répondre au mieux aux enjeux de l'architecture SOA **[05]**. Les web services fournissent les bases technologiques nécessaires pour réaliser l'interopérabilité entre les applications en utilisant différentes plateformes, différents systèmes d'exploitation et différents langages de programmation **[06]**.

### 2.2. Définition

Un web service est une application logicielle identifiée par une URI, qui possède une interface publique définie en utilisant XML. Sa définition peut être découverte par d'autres systèmes. Ces systèmes peuvent interagir avec le web service selon la manière prescrite par sa définition, en utilisant des messages basés sur XML et portés par des protocoles internet **[07]**.

### 2.3. Les standards des web services

L'objectif de cette section est le parcours des différentes spécifications des Web services. Ces spécifications pourront être mises en œuvre dans le cadre d'une architecture SOA basée sur les Web service **[07]**.

On va présenter les spécifications de base des Web services : SOAP, WSDL, UDDI.

#### a- SOAP

SOAP est un protocole basé sur XML, qui permet aux applications d'échanger des informations à travers HTTP **[07]** :

- SOAP est l'acronyme de Simple Object Access Protocol
- SOAP est un protocole de communication
- SOAP sert à la communication entre les applications (clients et services)
- SOAP est un format d'envoi de messages
- SOAP est conçu pour la communication à travers internet
- SOAP est indépendant de toute plateforme
- SOAP est indépendant de tout langage
- SOAP est simple et extensible
- SOAP permet d'éviter les difficultés causées par les pare-feux
- SOAP est un standard du W3C

#### Syntaxe de SOAP

Un message SOAP est un document XML ordinaire qui contient les éléments suivants :

- L'élément *Envelope* qui identifie le document XML comme étant un message SOAP
- L'élément *Header* qui est optionnel et qui contient des informations d'entête
- L'élément *Body* qui contient l'appel ainsi que la réponse retournée
- L'élément *Fault* qui est optionnel et qui fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message

Tous ces éléments cité ci-dessus sont déclares dans les *namespace* de l'enveloppe SOAP :

***https://www.w3.org/2003/05/soap-envelope/***

Et le *namespace* pour le SOAP encoding et les types de données :

***http://www.w3.org/2003/05/soap-encoding*** (***http://schemas.xmlsoap.org/soap/encoding***)

**Squelette d'un message SOAP**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  ...
  ...
</soap:Header>

<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

**L'élément Envelope**

Cet élément est la racine de tout message SOAP, il définit le document XML comme étant un message SOAP.

Noter l'utilisation du namespace xmlns:soap. Il doit toujours avoir la valeur :

***https://www.w3.org/2003/05/soap-envelope/***

Et il définit l'enveloppe comme étant une enveloppe SOAP.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

**Le namespace xmlns : soap**

Un message SOAP doit toujours avoir un élément Envelope associé au namespace :

http ://www. w3.org/2001/12/soap-envelop (***https://www.w3.org/2003/05/soap-envelope/***)

Si un autre namespace est utilisé, l'application doit générer une erreur.

**L'attribut encodingStyle**

Cet attribut est utilisé pour définir les types de données utilisés dans le document. Cet élément peut apparaître dans n'importe quel élément SOAP, et il sera appliqué au contenu de cet élément ainsi que tous ses éléments fils. Un message SOAP n'a pas d'encodage par défaut.

**Syntaxe**

```
soap:encodingStyle="URI"
```

**Exemple**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
...
Message information goes here
...
</soap:Envelope>
```

**L'élément Header**

C'est un élément optionnel et contient des informations spécifiques à l'application (par exemple des informations sur l'authentification) sur le message SOAP. Si cet élément est présent, il doit être le premier fils de l'élément Envelope.

**Note** Tous les éléments fils de l'élément Header doivent être qualifiés par un namespace.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">234</m:Trans>
</soap:Header>

...
...

</soap:Envelope>
```

Dans l'exemple ci-dessus l'élément *header* a l'élément *Trance* comme fils qui a la valeur 234 et l'attribut *mustUnderstand* de valeur 1.

SOAP définit trois attributs pour l'élément Header. Ces attributs sont *actor*, *mustUnderstand*, et *encodingStyle*. Les attributs définis dans l'élément Header indiquent comment le récepteur doit traiter le message SOAP.

**L'attribut actor**

Un message SOAP parcourt un chemin du l'émetteur vers le récepteur en passant par différents points (*endpoints*). L'attribut *actor* peut être utilisé pour adresser l'élément header à un endpoint particulier.

**Syntaxe**

```
soap:actor="URI"
```

**Exemple**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/appml/">
234
</m:Trans>
</soap:Header>

...
...

</soap:Envelope>
```

**L'attribut mustUnderstand**

Cet attribut indique si l'élément header doit être traité ou pas par le récepteur.

**Syntaxe**

```
soap:mustUnderstand="0|1"
```

**Exemple**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
<m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">
234
</m:Trans>
</soap:Header>

...
...

</soap:Envelope>
```

**L'élément SOAP Body**

Cet élément contient le message envoyé au récepteur. Le fils de l'élément Body peut être qualifié par un namespace.

**Exemple**

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
   <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
   </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

**L'élément SOAP Fault**

Un message d'erreur est porté par cet élément. Si un élément Fault est présent, il doit apparaître comme étant fils de l'élément Body. Un élément Fault ne peut apparaître qu'une seule fois dans un message SOAP.

L'élément Fault a les éléments fils suivants :

| élément fils | Description |
|---|---|
| <faultcode> | cet élément contient un code qui identifie l'erreur |
| <faultstring> | cet élément contient un texte décrivant brièvement l'erreur pour un humain |
| <faultactor> | cet élément contient le composant (i.e., son URL) qui a généré l'erreur |
| <details> | cet élément contient des informations spécifiques à l'application et concernant l'erreur |

**Les codes des erreurs**

Pour décrire une erreur l'élément *faultcode* utilise les valeurs suivantes dépendamment de l'erreur qui s'est produite :

| élément fils | Description |
|---|---|
| <VersionMismatch> | Namespace du bloc Envelope non valide |
| <MustUnderstand> | Un élément fils du bloc Header qui devait absolument être compris ne l'a pas été. |
| <Client > | Message mal formaté ou non valide |
| <Server> | Un problème lors du traitement du message |

**b- WSDL**

WSDL est un langage basé sur XML utilisé pour décrire les web services et comment les accéder **[07]** :

- WSDL est l'acronyme de Web Service Description Language
- WSDL est écrit en XML
- WSDL est un document XML
- WSDL est utilisé pour décrire les web services
- WSDL est un standard du W3C

**WSDL décrit les web services**

Le document décrit le web service. Il spécifie la localisation du web service et les opérations (méthodes) qu'expose ce web service.

**WSDL est une recommandation du W3C**

WSDL est devenu une recommandation du W3C en Juin 2007.

**La structure d'un document WSDL**

WSDL décrit un web service en utilisant ces principaux éléments :

- **Types:** précise les types de données complexes, pour lequel WSDL emploi XML Schema.

- **Message:** l'abstraction décrivant les données échangées entre services.
- **Operation:** l'abstraction décrivant une action implémentée par un service Web.
- **Port types:** Cet élément définit de manière abstraire une collection d'opérations ou d'actions, chaque opération est déclenchée par une requête, puis génère une réponse.
- **Binding (liaison):** Cet élément spécifie de manière concrète le protocole de communication (exemple : SOAP1.1, HTTP, MIME (*Multipurpose Internet Mail Extension*), …) et le format des donnés pour les opérations et messages définit par un type de port particulier.
- **Port:** Cet élément définit un point de communication unique avec l'adresse réseaux à laquelle elle est liée.
- **Service:** Cet élément définit une collection d'adresses (ports) permettant d'invoquer un service. Il sert à regrouper un ensemble de points de communication. En général ; il correspond à une URL invoquant un service SOAP.

Chaque document WSDL peut être documenté grâce à une balise **<documentation>**. Cet élément est facultatif.

Un document WSDL est divisé en deux parties : l'interface du service et son implémentation. L'interface du service est la partie réutilisable de la définition du service, elle peut être référencée par de multiples implémentations du service. Cette partie contient les éléments : WSDL : binding, WSDL : portType, WSDL : message et WSDL:type.

Dans l'élément WSDL:portType, les opérations d'un service Web sont définies. Ces opérations définissent comment un message XML peut apparaître dans les flux des données entrants et sortants. Une opération est comprise comme une signature d'une méthode dans un langage de programmation OO. L'élément WSDL:message spécifie comment les types de données XML constituent les différentes parties d'un message. L'élément WSDL:message est utilisé pour définir les paramètres entrants et sortants d'une opération. L'utilisation des types de données complexes dans le message est décrite dans l'élément WSDL : types. L'élément WSDL : binding décrit le protocole, le format de données, la sécurité et autres attributs pour une interface d'un service particulier (WSDL : portType) [KREG 01].

La définition d'implémentation d'un service est un document WSDL qui décrit comment une interface particulière d'un service est implémentée par un fournisseur donné. Un service Web est modélisé par un élément WSDL:service. Un élément service contient une collection (habituellement une seule) d'éléments WSDL:port. Un port associé un «endpoint»

(par exemple une adresse d'un endroit sur le réseau ou une URL) à un élément WSDL : binding d'une définition d'interface d'un service [KREG 01].

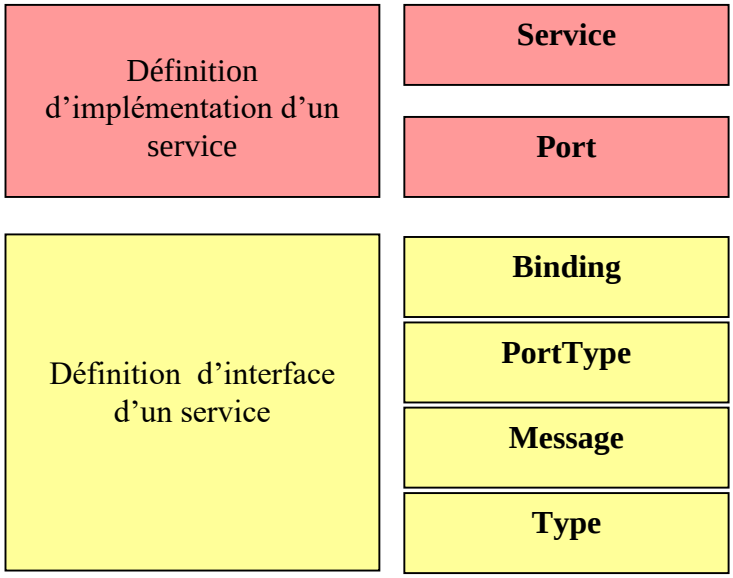



**Figure 7 : Description WSDL d'un service** [KREG 01].

La structure principale d'un document WSDL ressemble à :

```
<definitions>

<types>
   definition of types........
</types>

<message>
   definition of a message....
</message>

<portType>
   definition of a port.......
</portType>

<binding>
   definition of a binding....
</binding>

</definitions>
```

**Types d'opérations**

L'opération de type requête/réponse est la plus commune mais il ya d'autres types :

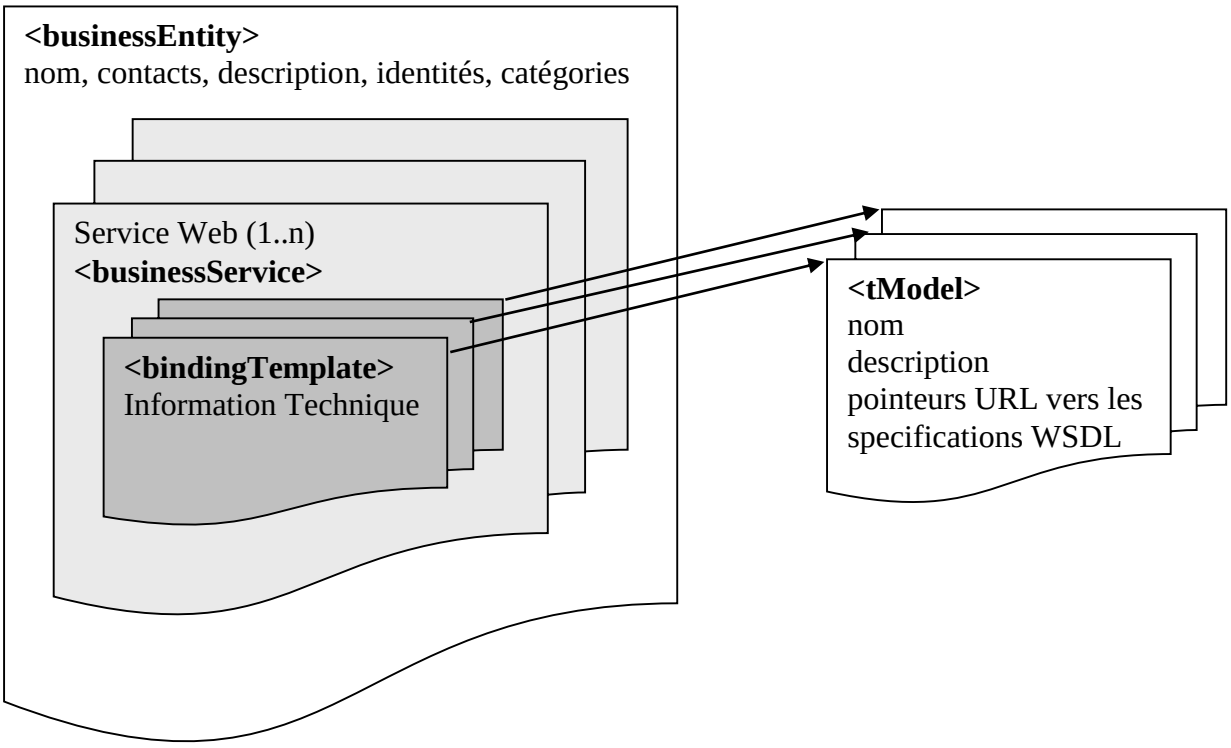
| Type | Definition |
|---|---|
| One-way | L'opération peut recevoir un message mais ne renvoie pas une réponse |
| Request_response | L'opération peut recevoir une requête et renvoie une réponse |
| Solicit-response | L'opération peut envoyer une requête et attendre la réponse |
| Notification | L'opération peut envoyer un message mais n'attend pas une réponse |

**c- UDDI**

Initialement définie par Ariba, IBM et Microsoft, UDDI est un protocole d'annuaire permettant aux entreprises de publier et de découvrir, d'une manière standard, des informations relatives aux fournisseurs et aux types de services qu'ils proposent. Ainsi, les clients peuvent savoir quels sont les services fournis par chaque fournisseur et les concepteurs de logiciels clients peuvent apprendre ce qu'ils ont besoin de connaître pour créer ces clients. UDDI est une technologie qui s'articule autour des protocoles HTTP et SOAP, ainsi que du langage XML. Les spécifications UDDI définissent les types d'annuaire de services Web distribues *: pages blanches* (nom de l'entreprises, adresse, contacts), *pages jaunes* (services classés par catégories industrielles) et *pages verts* (information d'implémentation des services Web proposes). Ainsi, UDDI se présente comme un ensemble de bases de données utilisées par les entreprises pour enregistrer leurs services Web ou pour localiser d'autres services Web. Grâce à UDDI, les entreprises peuvent enregistrer des données les concernant, des renseignements sur les services qu'elles offrent et des informations techniques sur le mode d'accès à ces services. Une fois l'enregistrement terminé, les informations sont automatiquement répliquées sur l'ensemble des annuaires. Ce fonctionnement permet aux services d'être découvert par un plus grand nombre d'entreprises **[09]**.

**Les types de données UDDI :**

L'XML schéma d'UDDI fournit 4 éléments obligatoires pour accéder et utiliser un Web service **[08]** (voir la figure ci-dessous)



**Structure des informations dans UDDI [08]**

- **L'entité commerciale : <BusinessEntity>**

Cet élément est la racine du document UDDI décrivant l'enregistrement du ou des web services d'un même fournisseur. Il contient l'identité de ce dernier, son adresse physique et électronique ainsi que des qualifications ou des mots-clés faisant référence aux taxonomies industrielles standards **[10]**.

- **La description des services : <BusinessServices >**

A l'intérieur de l'élément précédent, les services proprement dits sont délimites par la balise *businessService*. Chacun de se sous éléments contient pour l'essentiel le nom et la description du service, sa catégorie dans une taxonomie propre à UDDI, des clés de recherche et des pointeurs vers des classes de liaisons (bindingTemplates) **[10]**.

- **La liaison UDDI : <BindingTemplates >**

Comme dans WSDL, la liaison UDDI regroupe, pour un protocole de communication donné, les données techniques nécessaire à l'exploitation du web services par un programme : adresse IP, noms de domaines et le cas échéant, des informations sur les modalités d'usage du service (hébergement, paramétrage initial, et.). Un même service peut disposer de plusieurs points d'accès, par exemple, selon des protocoles différents (SOAP, SMTP…) **[10]**.

- **Les modèles données *:* <tmodels >**

Le model est une structure « creuse » qui est utilisée dans la description des entités commerciales comme référence à un autre document décrivant un modèle de données ou toute autre information nécessaire aux requêtes de recherche et aux interactions avec l'entité commerciale considère. Dans le cas courant de l'enregistrement d'un web service, par exemple, le *tmodel* pointera, en général, vers le document WSDL décrivant l'interface publique du service. Plus généralement, le tmodel peut renvoyer à d'autres documents spécifiant par exemple, les conventions employées dans les échanges ou bien encore les taxonomies industrielles les attributs peuvent faire référence, etc. **[10]**.

**API**

Un annuaire UDDI offre plusieurs point d'entrées (API), mais les deux principales bibliothèques d'appels sont : l'API de requête utilisé par les utilisateurs des services Web pour chercher et exploiter les services et l'API de publication pour publier les services Web par les fournisseurs (les entreprises).

**Publication d'un document WSDL dans un annuaire UDDI**

Pour enregistrer un service dans un annuaire UDDI, il faut publier ses deux documents WSDL : le document interface qui contient la définition du service (<Types>, <Message>, <Portype>, <binding>) et le document implémentation qui contient la description du service lui-même (<Service>, <Port>) où ce dernier importe le document interface.

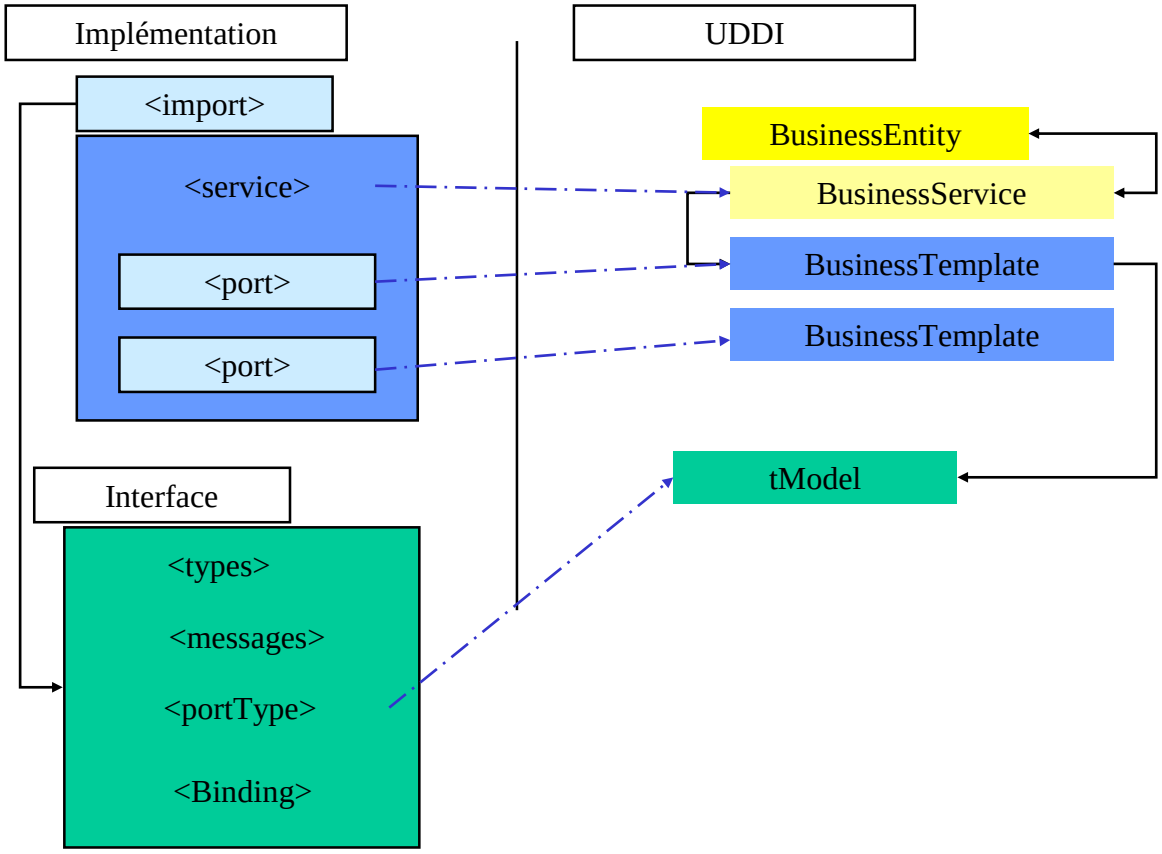



**Figure : Enregistrement du document WSDL dans un annuaire UDDI**