

Les Web Services

- Généralité
- Les technologies utilisées
 - XML
 - SOAP
 - WSDL
 - UDDI
- La composition de service : BPEL

Définitions (1)

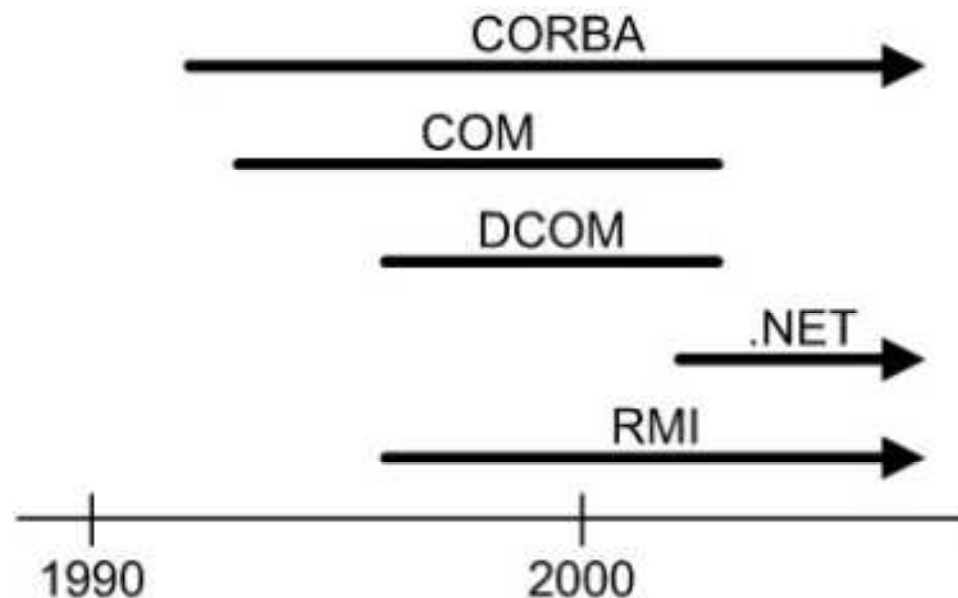
- Un service Web est une application logicielle identifiée par un URI dont les interfaces et les liaisons sont définies, décrites et découvertes en XML et qui supporte une interaction directe avec les autres applications logicielles en utilisant des messages XML via un protocole Internet. (W3C)
- Un service web est un composant applicatif mis à la disposition sur un réseau et disposant de méthodes que l'on peut invoquer à distance via l'emploi de protocole standard. Les services Web présentent l'avantage d'être faiblement couplés, indépendants des plateformes et réutilisables.

Définitions (2)

- **Un service Web est donc :**
 - Auto-descriptif, publiable et accessible en utilisant le langage XML
 - Indépendant du système d'exploitation et du langage de programmation
- **Intérêt:**
 - *Modularité*
 - Réutilisation et composition des services
 - *Interopérabilité*
 - Dialogue entre environnement hétérogène
 - Permet la communication synchrone/asynchrone
 - *Intégration du système d'information*

Evolution

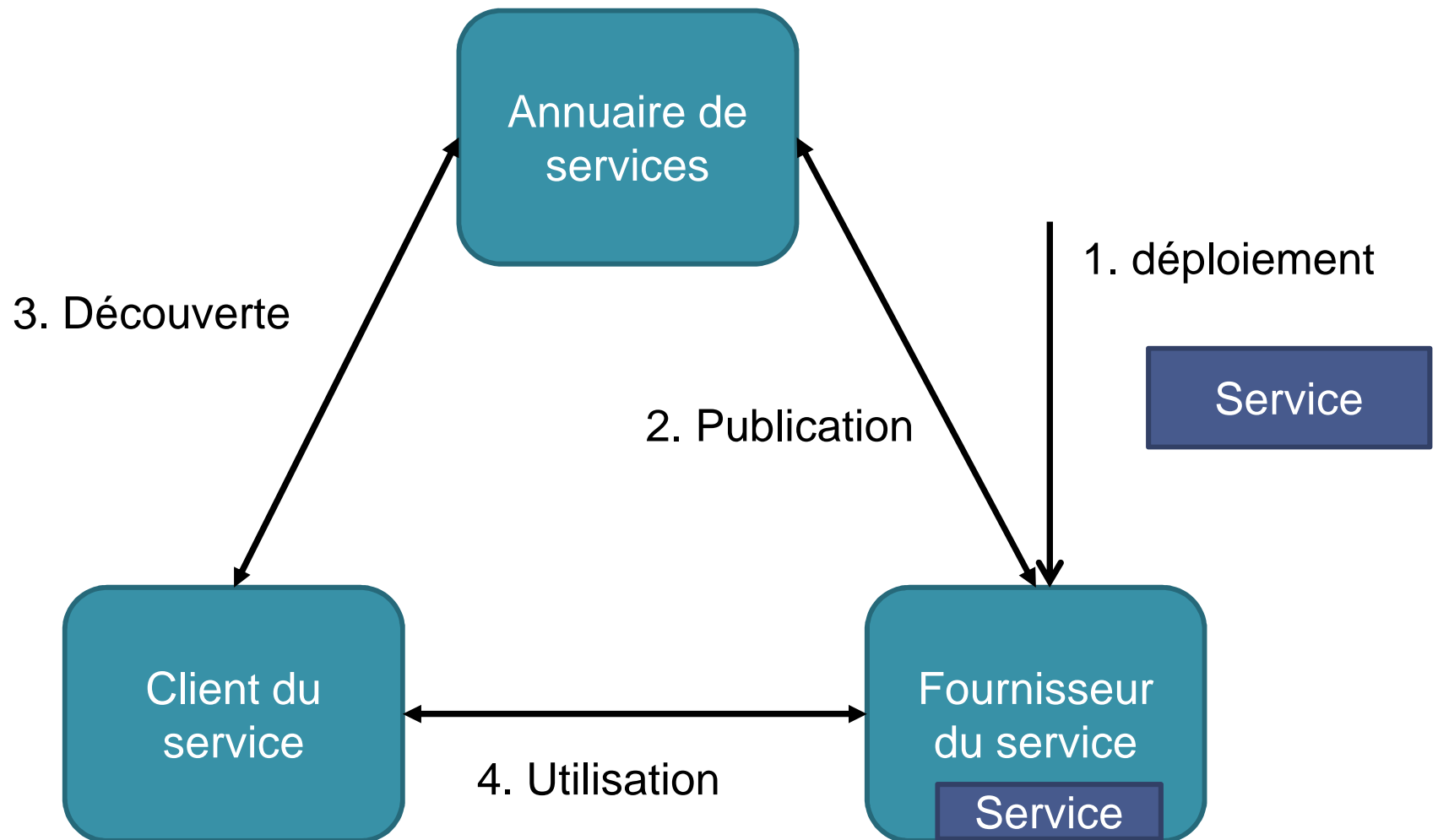
- **Plusieurs middleware existent**
 - RPC , puis évolution XML-RPC
 - MOM : Message Oriented Middleware
 - Java Message Service, Advanced Message Queuing Protocol , ...
 - Objets distribués
 - Corba, COM, DCOM, RMI, EJB, ...
 - Base de données Orientées Middleware



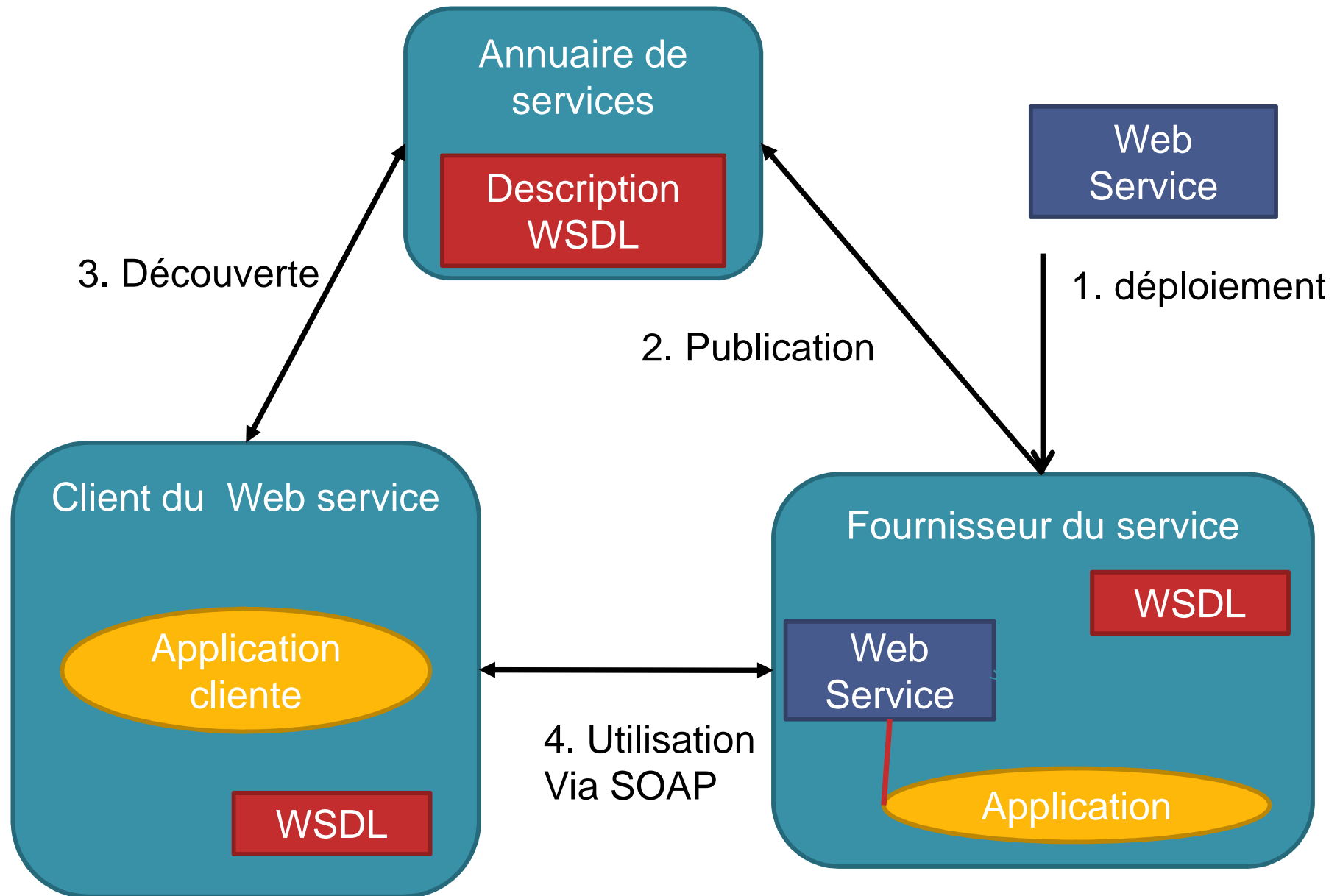
Web Services – Pourquoi ?

- Avoir des composants utilisables dans une architecture orientée service (SOA)
- Facilité l'interopérabilité entre des applications développées pour des plates-formes et dans des langages différents
- Permettre « d'ouvrir » des applications existantes sur le monde Internet
- Permettre d'utiliser des fonctions développées sur un système distant.

Architecture Orientée Services



SOA pour les Web Services



Web Services : technologies

- XML (eXtensible Markup Language)
 - Échange de messages XML entre client et serveur
 - Lisible, structuré
- HTTP, SMTP...
- SOAP (Simple Object Access Protocol)
 - Protocole définissant les échanges XML entre entités
- WSDL (Web Services Description Language)
 - Langage de description technique des services web proposés
- UDDI (Universal Description Discovery and Integration)
 - Annuaire des services web disponibles et des fournisseurs de ces services

XML (1)

- Norme du W3C depuis 1998, XML appartient au SGML (Standard Generalized Markup Language)
- Utilisation d'un balisage hiérarchique, mais chaque balise peut dépendre de l'application
- XML est utilisé pour l'échange des données et des messages, et peut donc être utilisé par tous les ordinateurs (langage fait pour eux)
- Langage stricte vérifiant la syntaxe (utilisation d'un chevron d'ouverture et de fermeture, respect de la casse)

XML (2)

- DTD (Document Type Definition) et les Schémas XML permettent de définir la syntaxe et le contenu utilisé pour l'application.
 - Peut-être incluse dans le fichier XML, mais généralement elle est séparée du document qu'elle représente.
- Un document XML commence toujours par une balise de déclarations et n'a qu'un seul élément racine.
 - Exemple :

```
<?xml version="1.0" ?>
<definitions .....>
<.....>
</definitions>
```
- La séparation du contenu d'un document, de sa structure et de sa représentation facilite l'échange d'informations entre les différents partenaires.

XML –RPC (1)

- **Définition**

- C'est une spécification et un jeu d'implémentations qui permet à des programmes tournant sur des systèmes hétérogènes d'effectuer des appels de procédure au travers d'Internet. C'est similaire au RPC, mais en utilisant http comme moyen de transport et XML pour l'encodage. Il est conçu pour être aussi simple que possible, tout en permettant la transmission, le traitement et le renvoi de structure de données complexes. (Dave Winer)

- En bref, **c'est une requête http de type POST, dont le corps du message est codé en XML.**
- Utilisation de XML pour simplifier le message, et de pouvoir évoluer plus simplement.

XML –RPC (2)

- **Syntaxe pour une requête**

- Une seule balise <methodCall>
- Qui contient une seule balise <methodName>
- Qui contient une seule balise <params>, pouvant contenir plusieurs balises <param> et <value>
- Le type <value> peut avoir :
 - <i4> ou <int> , <boolean>, <string>, <double>, <array>,<struct>....
- *Exemple:*

```
<methodCall>
  <methodName> exemple.sum </methodName>
  <params>
    <param> <value> <int>5</int> </value> </param>
    <param> <value> <int>8</int> </value> </param>
  </params>
</methodCall>
```

XML –RPC (3)

- **Syntaxe pour la réponse**


- Une seule balise <methodResponse>
- Qui contient une seule balise <params>, pouvant contenir plusieurs balises <param> et <value>

- *Exemple:*

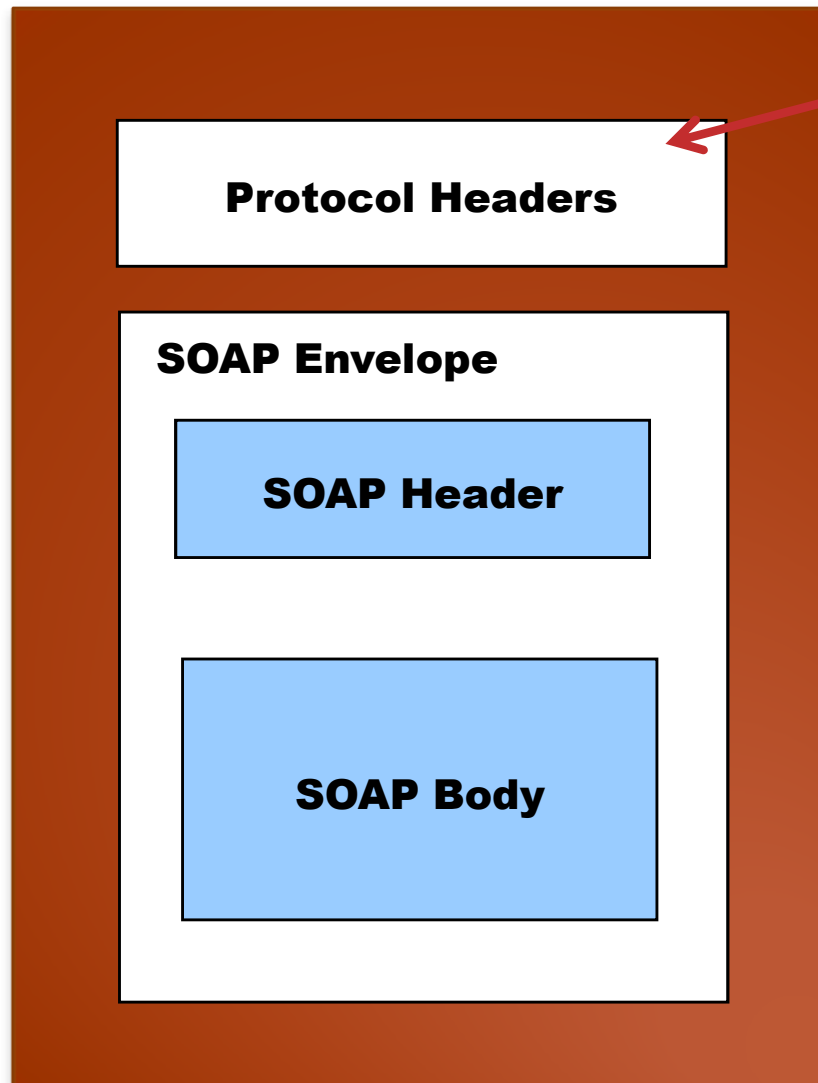
```
<methodResponse>  
  <params>  
    <param> <value> <int>13</int> </value> </param>  
  </params>  
</methodResponse>
```

- La balise <methodResponse> peut aussi contenir une balise <fault>, qui contient un élément <value>, contenant une structure avec deux champs String : FaultCode et FaultString.

SOAP (1)

- **SOAP** (Simple Object Access Protocol)
- Permet une normalisation des échanges de données. Les données sont encodées en XML et échangées par des appels de procédure à distance.
- Bien que le protocole de transport le plus commun soit http(s), il est possible d'utiliser SMTP
  aucune perturbation par les pare-feux
- Standard W3C, extensible et simple, indépendant de la plate-forme

SOAP (2)



Entête http ou smtp

- Éléments d'un message SOAP :
 - Enveloppe
 - Élément pouvant contenir des déclarations d'espaces de noms ou des sous-éléments
 - Header (optionnel)
 - Permet des extensions telles que authentification, session...
 - Body (obligatoire)
 - Définit la méthode appelée, contient les paramètres, peut contenir un élément Fault en cas d'erreur

SOAP (3)

- **L'enveloppe**
- Élément obligatoire dans un message SOAP.
- Permet de spécifier la version de SOAP utilisé, en utilisant un espace de nom
- Permet se spécifier les règles d'encodage (sérialisation et désérialisation) mises en œuvre dans le message.

```
POST /Orders HTTP/1.1
Host: xxx.xxx.xxx.xxx
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://www.someorders.com/GetOrders"
```

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


SOAP (4)

- **L'entête SOAP**
 - La balise Header est optionnelle
 - Si la balise Header est présente, elle doit être le premier fils de la balise Envelope
 - La balise Header contient des *éléments spécifiques à l'application*
- **3 sortes d'éléments possibles :**
 - L'attribut : *Actor* (désigne le destinataire)
 - L'attribut : *MustUnderstand*
 - L'attribut : *encodingStyle*

SOAP (5)

- **SOAP body**

- La balise Body est obligatoire
- La balise Body doit être le premier fils de la balise Envelope (ou le deuxième si il existe une balise Header)
- La balise Body contient des *entrées* qui sont des données applicatives et qui seront exécutées par le destinataire.
- Une *entrée* est n'importe quelle balise incluse optionnellement dans un namespace
- Une *entrée* peut être une Fault.

SOAP (6)

- **Exemple**

- Requête HTTP invoquant une méthode *GetOrders* :

- POST /Orders HTTP/1.1
Host: xxx.xxx.xxx.xxx
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "http://www.someorders.com/GetOrders"
- <?xml version="1.0"?>
- <SOAP-ENV:Envelope
 - xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 - SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 - <SOAP-ENV:Body>
 - <orders:GetOrders
 - xmlns:orders="http://www.someorders.com/orders">
 - <CustomerID>TOTO</CustomerID>
 - <SalesRepID>85</SalesRepID>
 - </orders:GetOrders>
 - </SOAP-ENV:Body>
 - </SOAP-ENV:Envelope>

SOAP (7)

- **SOAP Fault**

- Balise permettant de signaler des cas d'erreur
- D'après SOAP1.1, balise Fault contient :
 - Faultcode (obligatoire) : code permettant d'identifier le type d'erreur
 - Client, Server,
 - MustUnderstand, VersionMismatch
 - Faultstring (obligatoire) : explication en langage naturel
 - Faultactor : information identifiant l'initiateur de l'erreur
 - Detail : définition précise de l'erreur

SOAP (8)

- **Exemple:**

- `<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<s:Fault>
<faultcode xmlns="">s:Client</faultcode>
<faultstring xml:lang="fr-FR" xmlns="">
Une opération invalide s'est produite.</faultstring>
</s:Fault>
</s:Body>
</s:Envelope>`

Schéma

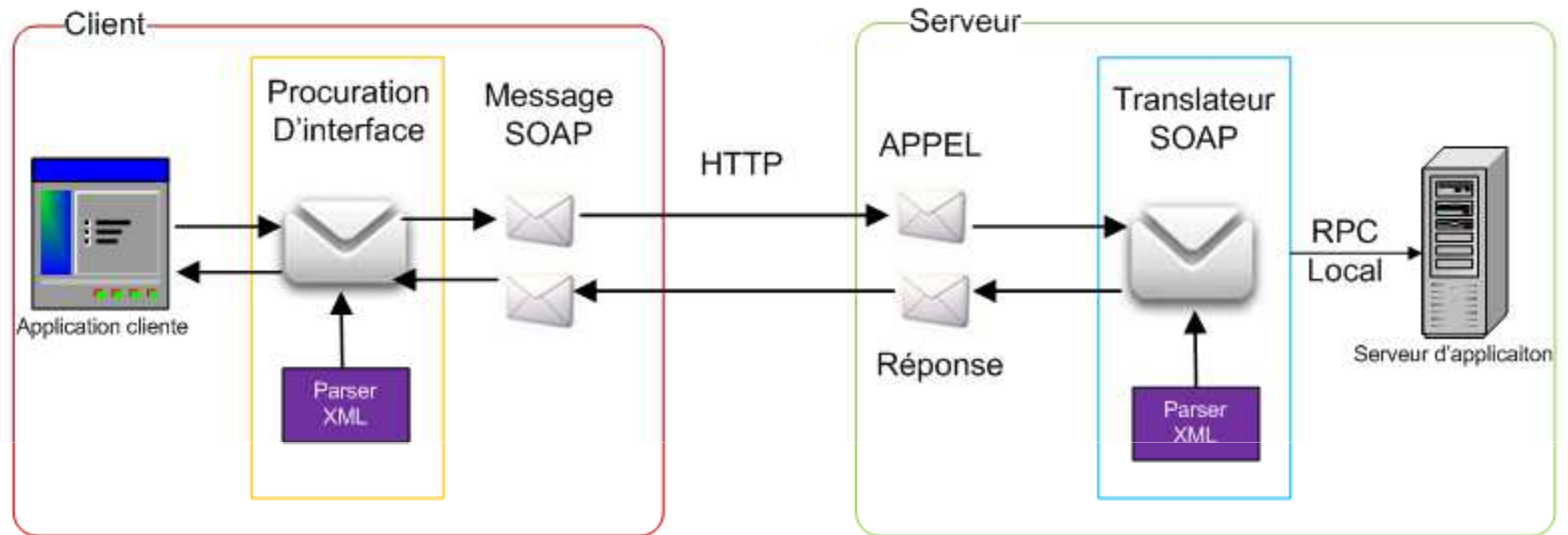


Image venant du site du zero

WSDL (1)

- **WSDL** (Web Service Description Language)
- Langage XML dédié à la description de tous les éléments nécessaires pour interagir avec un service réseau (ou service web)
- Description abstraite de l'ensemble des fonctionnalités
 - Informations : protocoles, ports, format des messages, paramètres en entrées et en sortie, exceptions possibles
- Un document WSDL est donc associé à un Web Service afin d'en décrire l'ensemble des éléments qu'il expose vers le monde extérieur :
 - types de données (Simple ou avec XSD)
 - types de messages
 - liaisons avec le protocole de transport et le format des messages (SOAP 1.X, HTTP Get et HTTP Post, ou encore MIME)

WSDL –Structure (1)

- Définitions

- Contient le nom du service et sin nécessaire, les namespaces faisant référence aux types utilisés dans le document.
- `<definitions name="CarnetAdresse" targetNamespace="http://exemple.com/carnetAdresse.wsdl" xmlns:per="http://exemple.com/carnetAdresse.wsdl" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">`

- Types

- Définition des types utilisés dans l'application
- Utilise en général la syntaxe XSD
- Peut renvoyer directement sur une structure xsd via un lien.

WSDL –Structure (2)

```
<types>
<xsd:schema targetNamespace="http://exemple.com/personne.xsd"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<xsd:complexType name="Personne">
  <xsd:element name="Nom" type="xsd:string">
  <xsd:element name="Prenom" type="xsd:string">
  <xsd:element name="Age" type="xsd:float">
</complexType>
</types>
```

- ```
<types>
<xsd:schema>
<xsd:import namespace="http://Webservice1/"
schemaLocation="http://172.16.65.168:8080/Newwebservice1/NewWeb
ServiceWSService?xsd=1"/>
</xsd:schema>
</types>
```

# WSDL –Structure (2 bis)

```
<xs:schema version="1.0" targetNamespace="http://Webservice1/">
 <xs:element name="hello" type="tns:hello"/>
 <xs:element name="helloResponse" type="tns:helloResponse"/>
 <xs:element name="test" type="tns:test"/>
 <xs:element name="testResponse" type="tns:testResponse"/>
 <xs:complexType name="hello">
 <xs:sequence>
 <xs:element name="name" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 </xs:complexType>
 <xs:complexType name="helloResponse">
 <xs:sequence>
 <xs:element name="return" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 </xs:complexType>
 <xs:complexType name="test">
 <xs:sequence/>
 </xs:complexType>
 <xs:complexType name="testResponse">
 <xs:sequence>
 <xs:element name="return" type="xs:string" minOccurs="0"/>
 </xs:sequence>
 </xs:complexType>
 </xs:schema>
```

# WSDL –Structure (4)

- Message(name, part())

- Définition abstraite des messages échangés entre deux nœuds, et la liste des paramètres qui peuvent être passés.

```
<message name="GetPersonne">
 <part name="body" type="Personne"/>
</message>
<message name="AddPersonne">
 <part name="Nom" type="xsd:string"/>
 <part name="Prenom" type="xsd:string"/>
 <part name="Age" type="xsd:float"/>
</message>
```

- portType(name, operation(name(input msg, output msg))

- Toutes les opérations associées aux méthodes, ainsi que les paramètres d'entrée/sortie.

# WSDL –Structure (5)

```
<portType name="CarnetAdresse">
 <operation name="AddPersonne">
 <input message="AddPersonne"/>
 </operation>
 <operation name="GetPersonneParNom">
 <input message="GetPersonne"/>
 <output message=" GetPersonne"/>
 </operation>
</portType>
```

- Binding(name, ...
  - Spécifie une liaison entre un <portType> et un protocole concret.
  - Spécifie le protocole de communication utilisé

## Différente information :

- Binding : association du service et d'un porttype. On indique aussi le type du format des messages
- Operation : indique le nom de l'opération
- Body : détails sur les entrées/sortie à utiliser.

# WSDL –Structure (5)

```
<binding name = "CarnetAdresseBinding" type="CarnetAdresse">
 <soap:binding transport=http://schemas.xmlsoap.org/soap/http style="rpc" />
 <operation name="GetPersonneParNom">
 <soap:operation soapAction="http://exemple.com/ GetPersonneParNom " />
 <input>
 <soap:body use="encoded" encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
 </input>
 <output>
 <soap:body use="encoded" encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
 </output>
 </operation>
</binding>
```

- Port , encapsulé dans service:
  - Spécifie l'adresse et le port du service à utiliser

```
<service name="CarnetAdresseService">
 <port name=" CarnetAdressePort " binding="CarnetAdresseBinding">
 <soap:address location="http://example.com/GetPersonneParNom"/>
 </port>
</service>
</definitions>
```

# UDDI

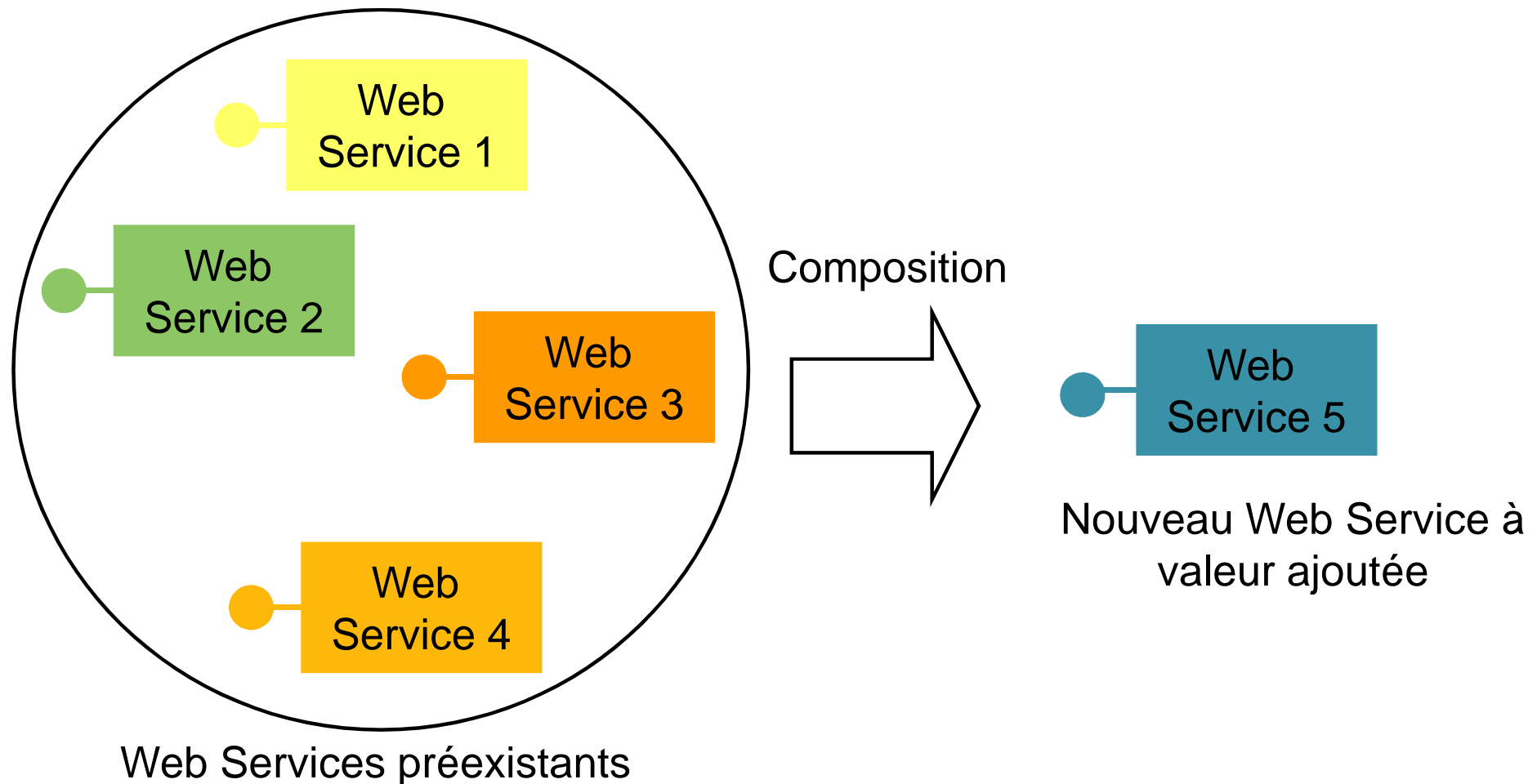
- **UDDI** : Universal Description, Discovery and Integration
- UDDI permet aux fournisseurs de services de s'inscrire et de répertorier les services qu'ils proposent
- L'annuaire est composé de :
  - *Pages blanches* : contiennent des informations sur les entreprises comme leur nom, adresse,...
  - *Pages jaunes* : recensent les Web service de chacune des entreprises, au format WSDL
  - *Pages vertes* : informations techniques sur les services web proposés

# Web Services REST

- REST est l'acronyme de Representational State Transfer.
- Avec REST, l'information de base est appelée **ressource**. Toute information qui peut être nommée est une ressource : un article d'un journal, une photo, un service ou n'importe quel concept
- L'URI décrit la ressource, on utilise les mêmes commandes qu'avec HTTP (PUT, GET, HEAD,...)
- Mais, chaque opération est auto-suffisante : **il n'y a pas d'état**

# Web Services – Composition (1)

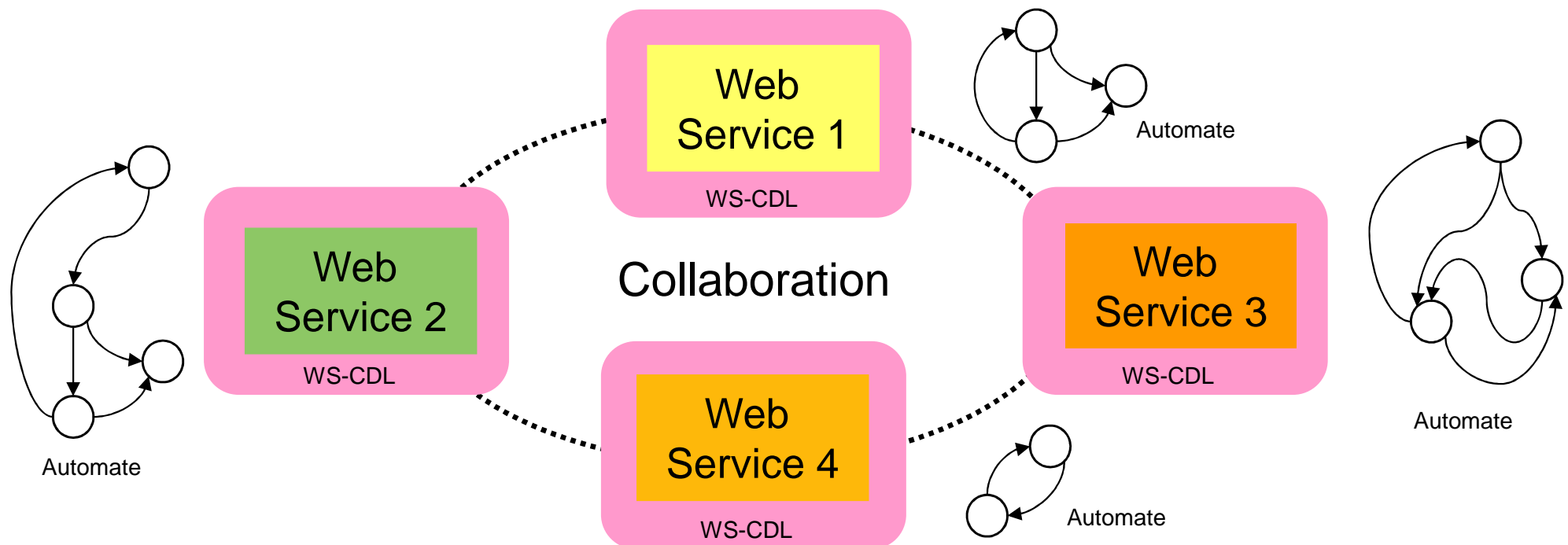
- Agréger des Web Services élémentaires pour aboutir à un nouveau Web Service à valeur ajoutée





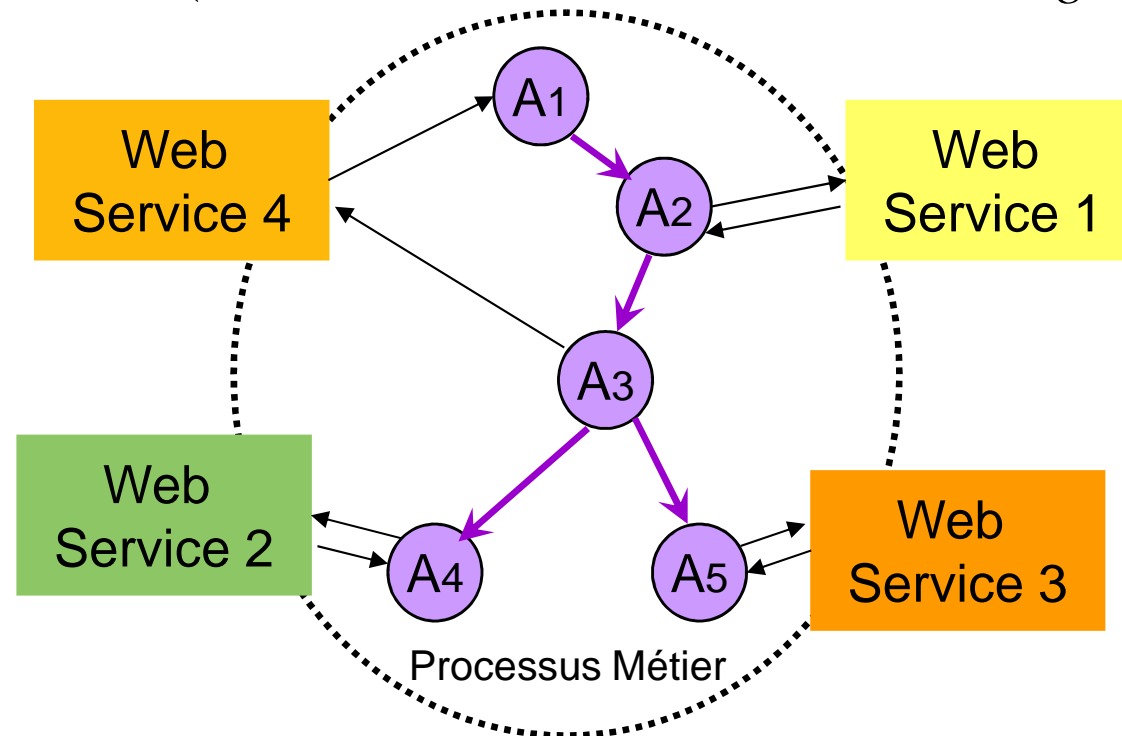
# Web Services – Composition (2)

- Chorégraphie
  - Vision distribuée
  - Collaboration publique entre participant
  - WS-CDL (*WS-Choreography Description Language*)



# Web Services – Composition (3)

- Orchestration
  - Vision centralisée
  - Implantation privée des participants
  - Exécutable
  - WS-BPEL (*WS-Business Process Execution Language*)

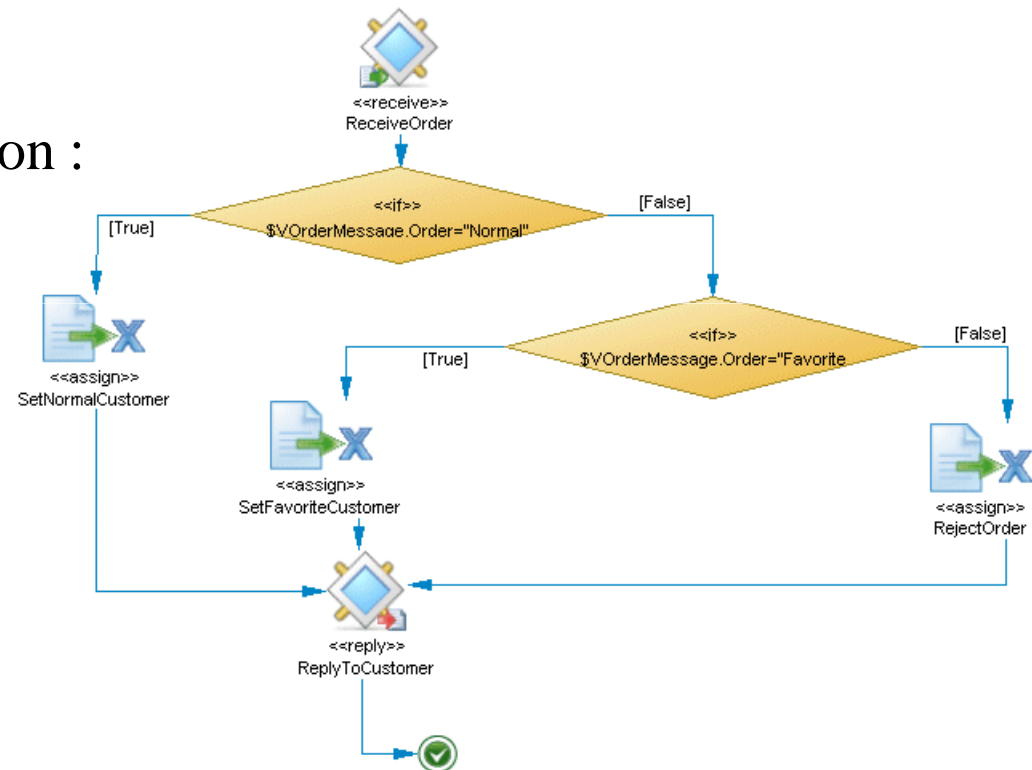


# Web Services – BPEL (1)

- BPEL : Business Process Execution Language
  - Standard créé par OASIS
  - Langage de programmation basé sur XML permettant de définir les interactions **intra et inter entreprises**.

- 2 modes d'utilisation :

- BPEL designer
- BPEL moteur



# Web Services – BPEL (2)

- Echanges de messages avec les partenaires métiers
  - Le format des messages échangés est décrit par des interfaces WSDL qui sont déclarés comme des "Partner Links"
  - Permettent de décrire à la fois les clients et les partenaires du processus BPEL.
- Processus Stateful
  - Gestion des états
  - Corrélation des messages avec les états
- Ordre des messages spécifiés explicitement
- Implémentation de la logique métier en interne du processus

# Web Services – BPEL (3)

- WS-BPEL:

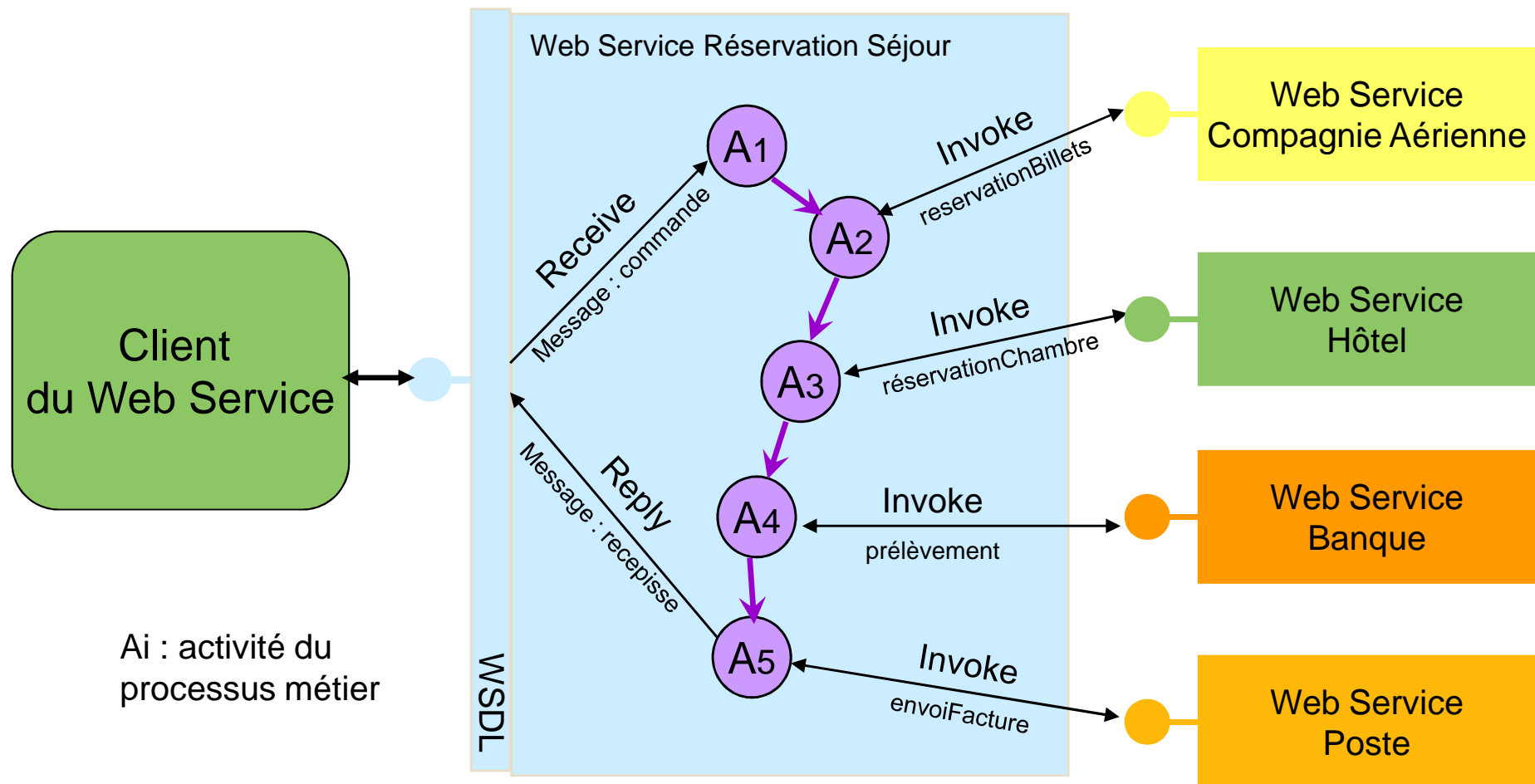
- Activité de base

- Invoquer un service Web <invoke>
    - Recevoir une invocation <receive>
    - Répondre à une invocation <reply>

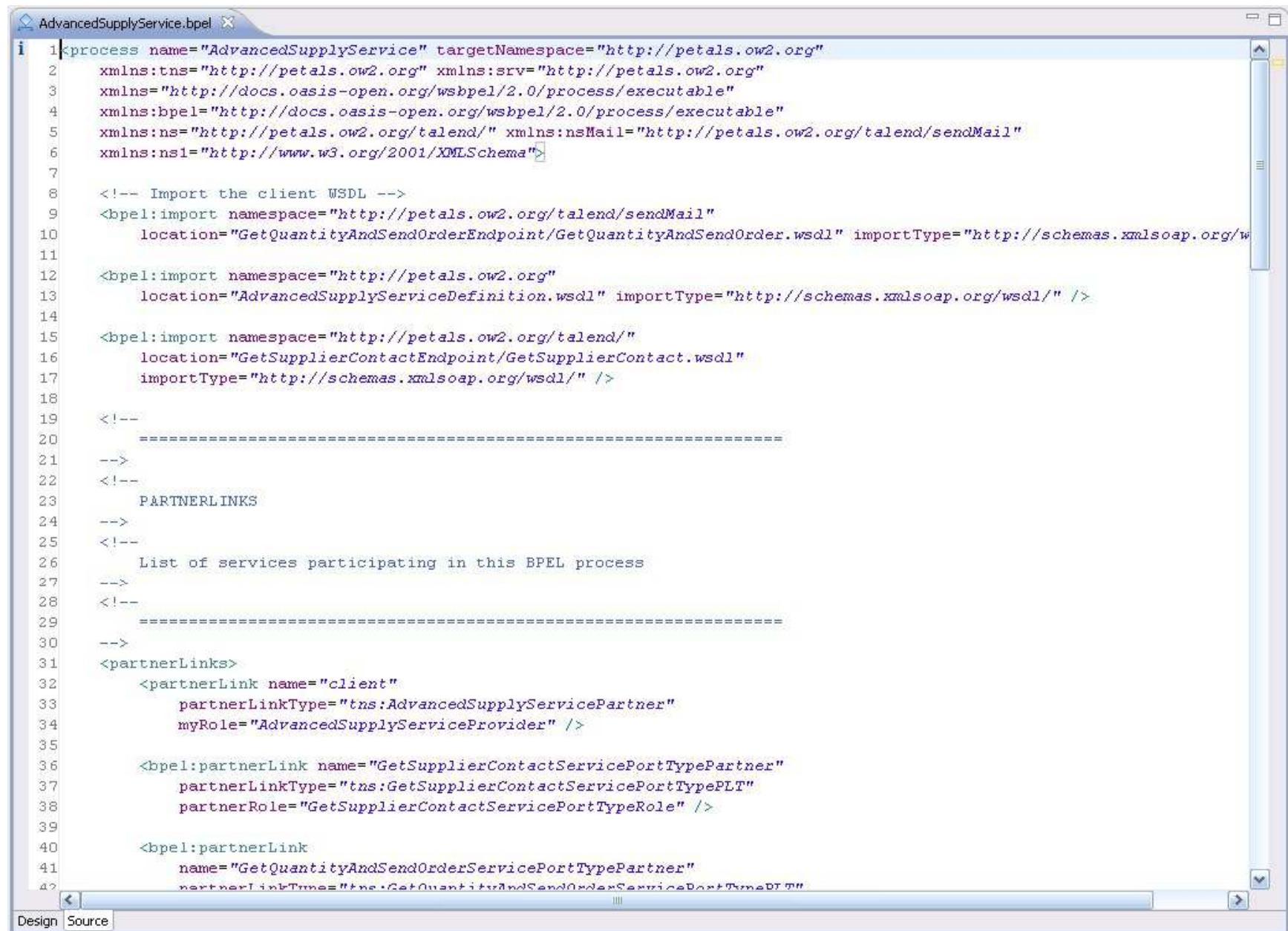
- Activité structurée

- Séquence ordonnée <sequence>
    - Branchement conditionnel <switch>,<if>
    - Boucle <while>
    - Exécution en parallèle <flow>
    - Affectation variable <assign>

# Web Services – BPEL (4)

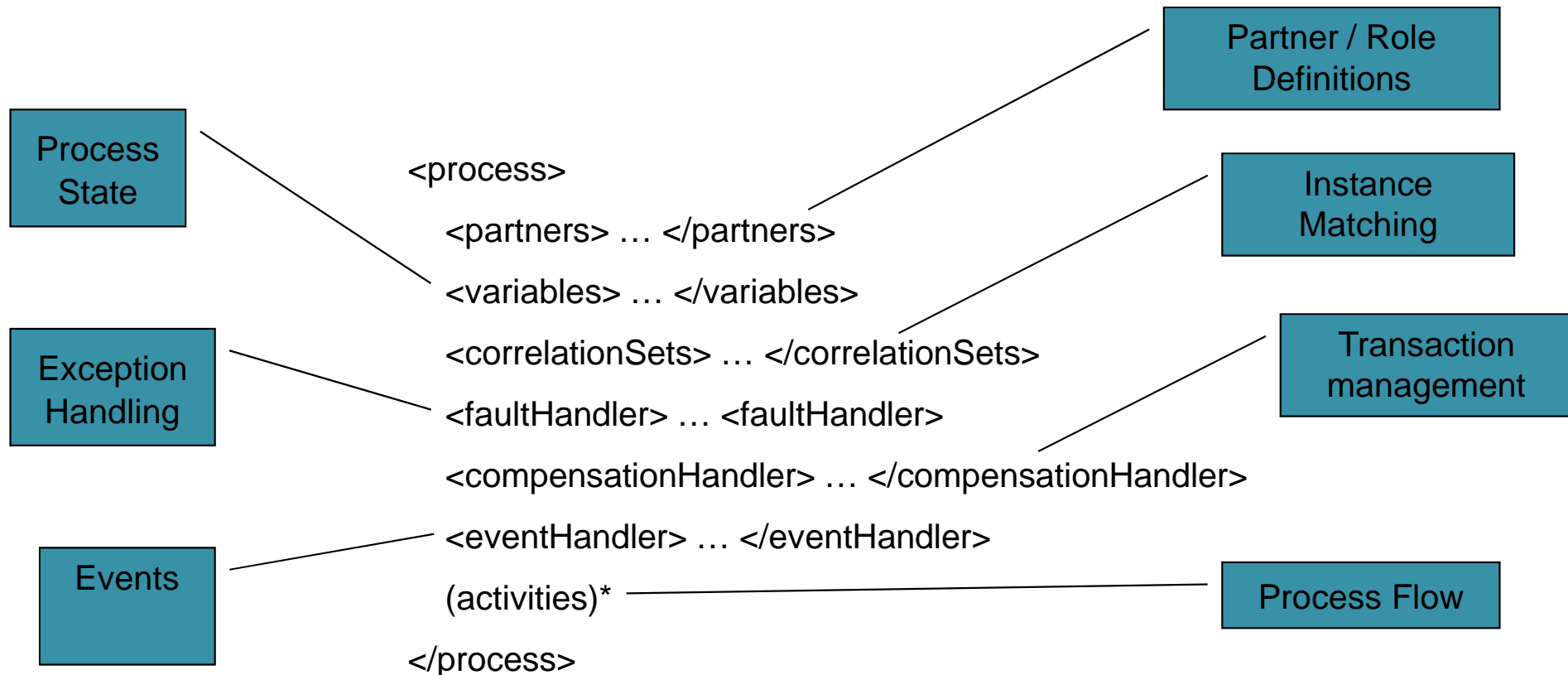


# Web Services – BPEL (4)



```
1 <process name="AdvancedSupplyService" targetNamespace="http://petals.ow2.org"
2 xmlns:tns="http://petals.ow2.org" xmlns:srv="http://petals.ow2.org"
3 xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
4 xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
5 xmlns:ns="http://petals.ow2.org/talend/" xmlns:nsMail="http://petals.ow2.org/talend/sendMail"
6 xmlns:ns1="http://www.w3.org/2001/XMLSchema">
7
8 <!-- Import the client WSDL -->
9 <bpel:import namespace="http://petals.ow2.org/talend/sendMail"
10 location="GetQuantityAndSendOrderEndpoint/GetQuantityAndSendOrder.wsdl" importType="http://schemas.xmlsoap.org/w
11
12 <bpel:import namespace="http://petals.ow2.org"
13 location="AdvancedSupplyServiceDefinition.wsdl" importType="http://schemas.xmlsoap.org/wsdl/" />
14
15 <bpel:import namespace="http://petals.ow2.org/talend/"
16 location="GetSupplierContactEndpoint/GetSupplierContact.wsdl"
17 importType="http://schemas.xmlsoap.org/wsdl/" />
18
19 <!--
20 =====
21 -->
22 <!--
23 PARTNERLINKS
24 -->
25 <!--
26 List of services participating in this BPEL process
27 -->
28 <!--
29 =====
30 -->
31 <partnerLinks>
32 <partnerLink name="client"
33 partnerLinkType="tns:AdvancedSupplyServicePartner"
34 myRole="AdvancedSupplyServiceProvider" />
35
36 <bpel:partnerLink name="GetSupplierContactServicePortTypePartner"
37 partnerLinkType="tns:GetSupplierContactServicePortTypePLT"
38 partnerRole="GetSupplierContactServicePortTypeRole" />
39
40 <bpel:partnerLink
41 name="GetQuantityAndSendOrderServicePortTypePartner"
42 partnerLinkType="tns:GetQuantityAndSendOrderServicePortTypePLT" />
43 </partnerLinks>
44 </process>
```

# BPEL Structure Overview



activities = <receive>, <reply>, <invoke>, <assign>, <throw>,  
<terminate>, <wait>, <empty>, <sequence>, <switch>,  
<while>, <pick>, <flow>, <scope>, <compensation>



# Web Services - Sécurité

- La problématique de **sécurité** des Web Services :
  - Authentification
  - Autorisation
  - Cryptage
- Les normes associées aux Web Services :
  - WS-Security;
  - SOAP Security Extensions;
  - XML Key Management;
  - ebXML Collaboration-Protocol Profile and Agreement Specification.