Architectures Orientées Services

Chapitre 3
Standards autour des web services

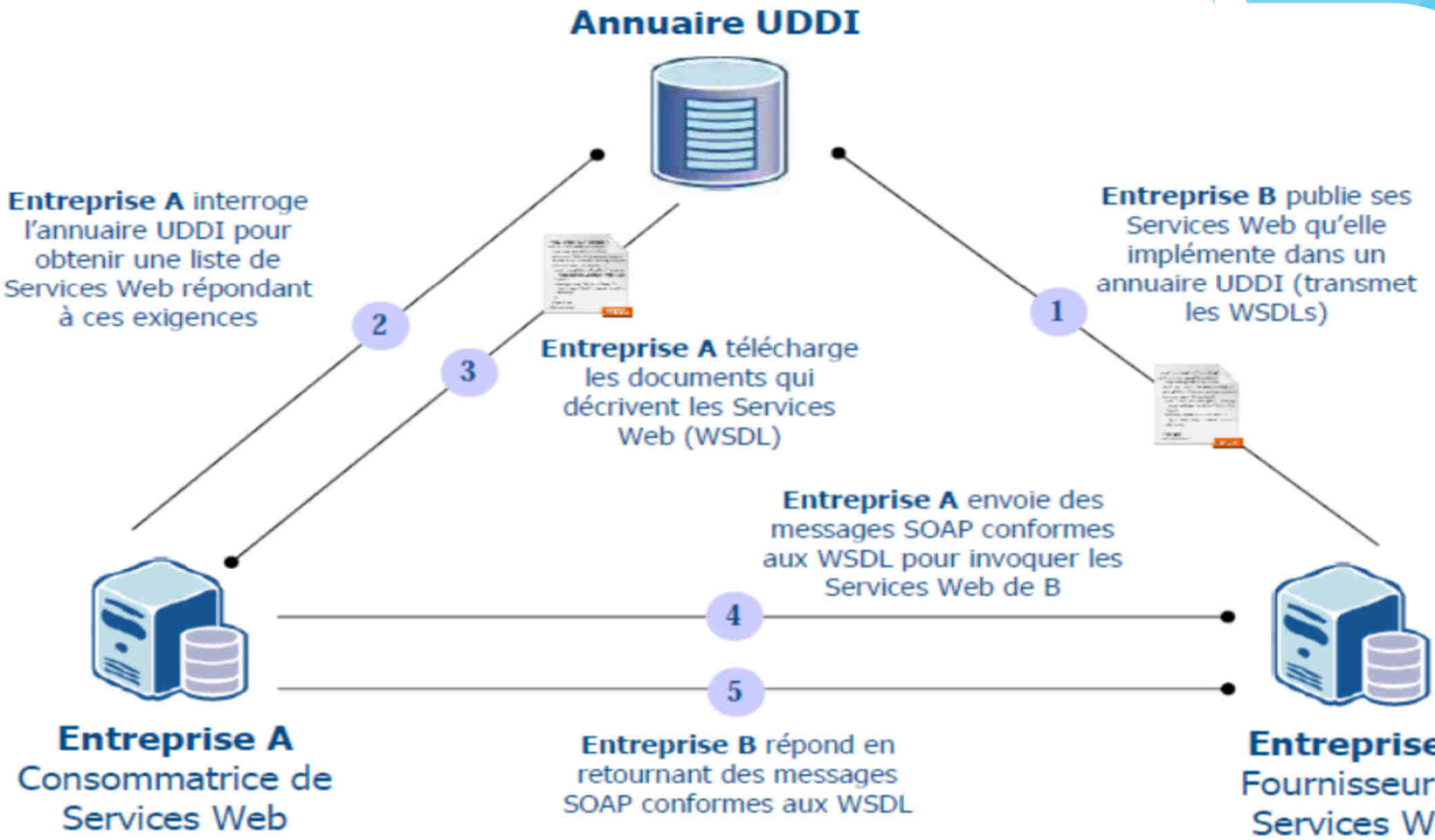
Hajer ALAYA

Hajer.adahmeni@gmail.com

Définition Services Web (W3C)

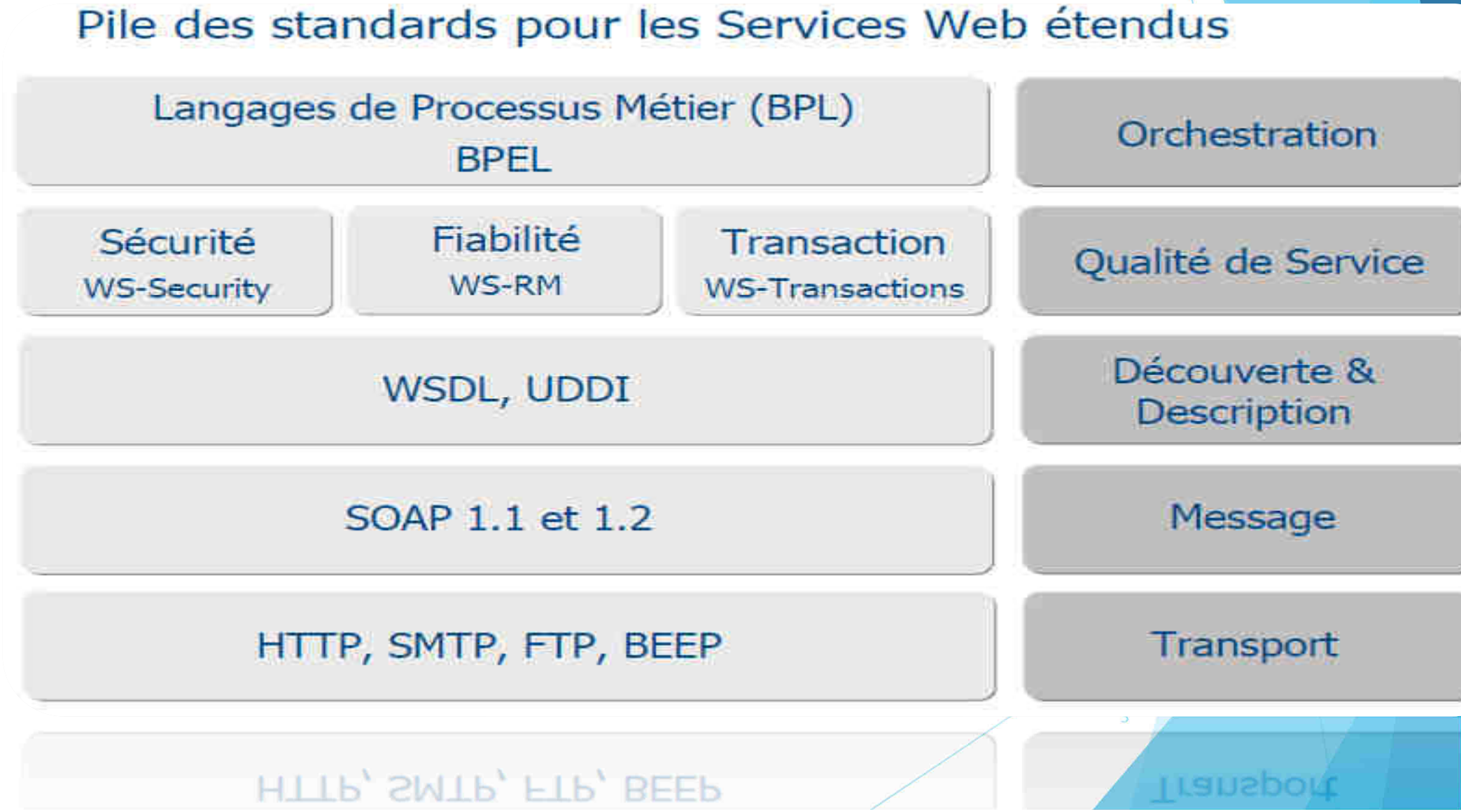
Un service web est un système logiciel **identifié par un URI** (***Unified Resource Identifier***), dont les **interfaces publiques et les liaisons** sont définis et décris **en utilisant XML**. Sa définition peut être **découverte par d'autres systèmes logiciels**. Ces systèmes peuvent interagir avec le service web d'une manière prédéfinie par sa définition, en utilisant des **messages basés sur XML**, acheminés par des **protocoles internet**.

Services Web



Source : M. Baron WSDL

Pile de service web



Apports des Services Web

- Peuvent correspondre à des applications nouvelles ou déjà existantes
- S'appuient sur des standards répandus
  - Meilleure interopérabilité
- Ne sont pas centrés sur un langage ou technologie
  - On peut faire communiquer du .Net avec du Java ou du Python

Standards pour les Web Services

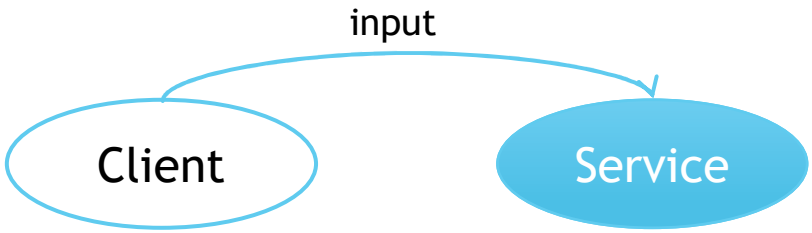
- Format de données
  - XML, XML Schéma
- Transport
  - HTTP, HTTPS, SMTP, FTP
  - SOAP, XML-RPC
- Description de service
  - WSDL, WSCL (*Web Services Conversation Language* : définit les interfaces abstraites du service web)
- Composition
  - MS XLANG, BPEL
- Découverte, Publication
  - UDDI (Universal Description Discovery and Integration)

⇨ **L'utilisation des standards rend l'accès aux services indépendant de la plateforme d'impl**

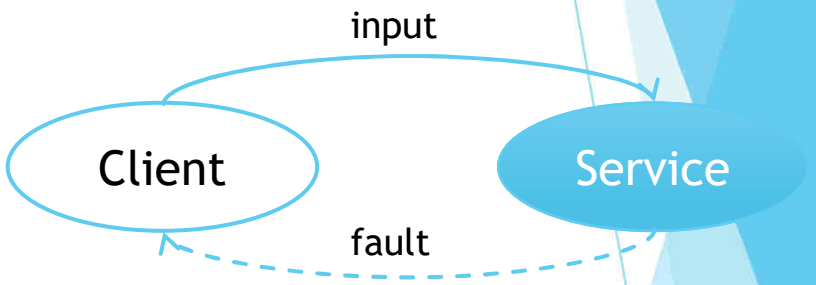
SOAP
Service Oriented Architecture Protocol

- Acronyme initialement pour *Simple Object Access Protocol*, changé dans sa version 1.2 vers *Service Oriented Architecture Protocol*
- Un message SOAP
  - Permet la transmission d'un message au format XML
  - Va d'un nœud émetteur vers un nœud receveur
  - Passe éventuellement par un certain nombre de « nœuds » intermédiaires
    - Maintenir des logs, faires des vérifications ou modifier ou rediriger les messages
- SOAP s'appuie sur :
  - XML et XML Schema pour la représentation des messages
  - Les protocoles internet classiques pour la transmission des messages
    - HTTP, SMTP, JMS, FTP…

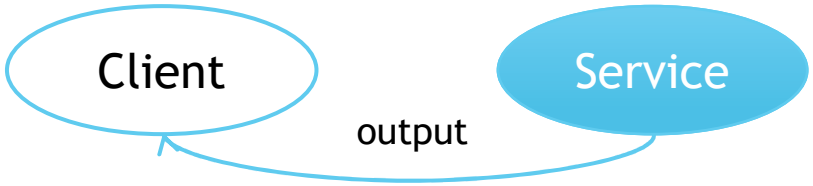
Modèles d'Échange des Messages (1/2)



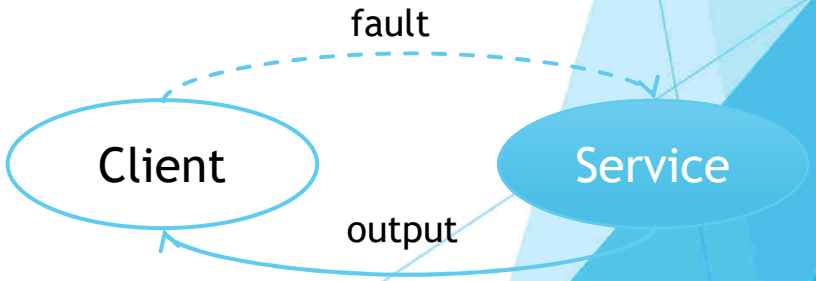
in-only
Le consommateur envoie un message au fournisseur qui change uniquement son statut



Robust In-only
Même chose que *in-only* si le statut du service change. Si une erreur est générée, une faute est envoyée

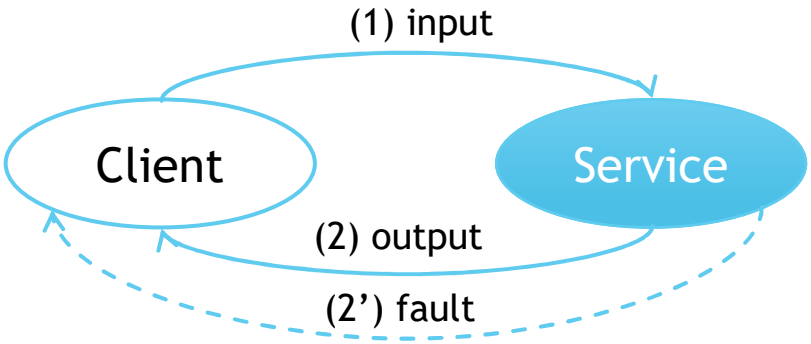


out-only
Le contraire de *in-only*, principalement une notification d'évènement

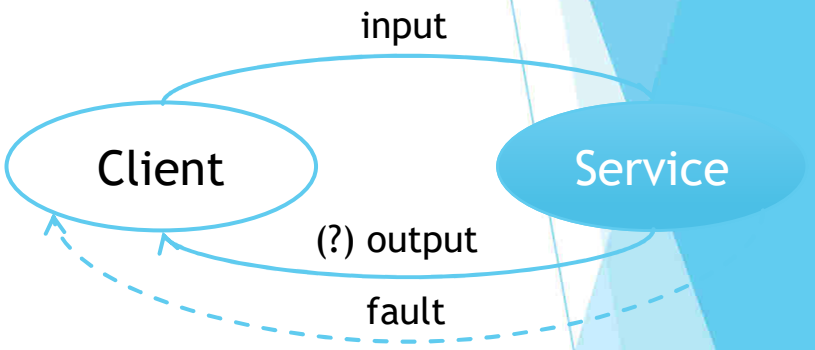


Robust out-only
Même chose que le *out-only*, sauf que ça peut provoquer un message d'erreur

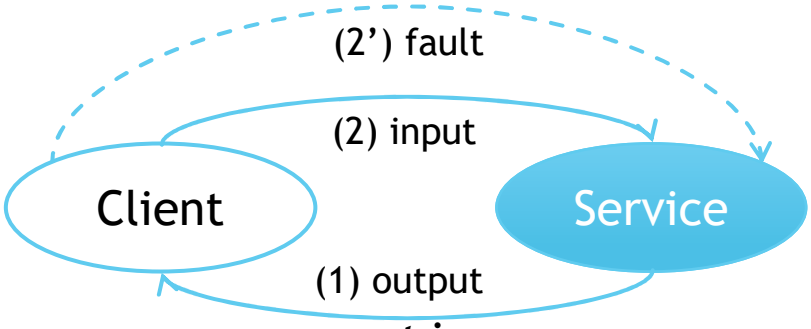
Modèles d'Échange des Messages (2/2)



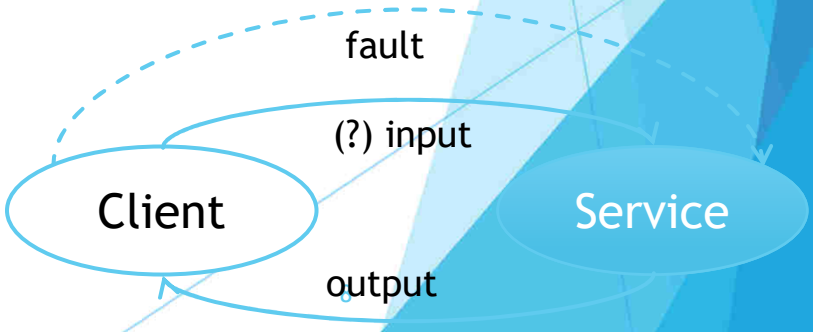
in-out
Équivalent à requête/réponse : le consommateur initie la requête, le fournisseur répond par un message ou une erreur.



in-optional-out
Un échange à deux directions standard où une réponse fournisseur est optionnelle.



out-in
Contraire de in-out : le fournisseur initie l'échange.



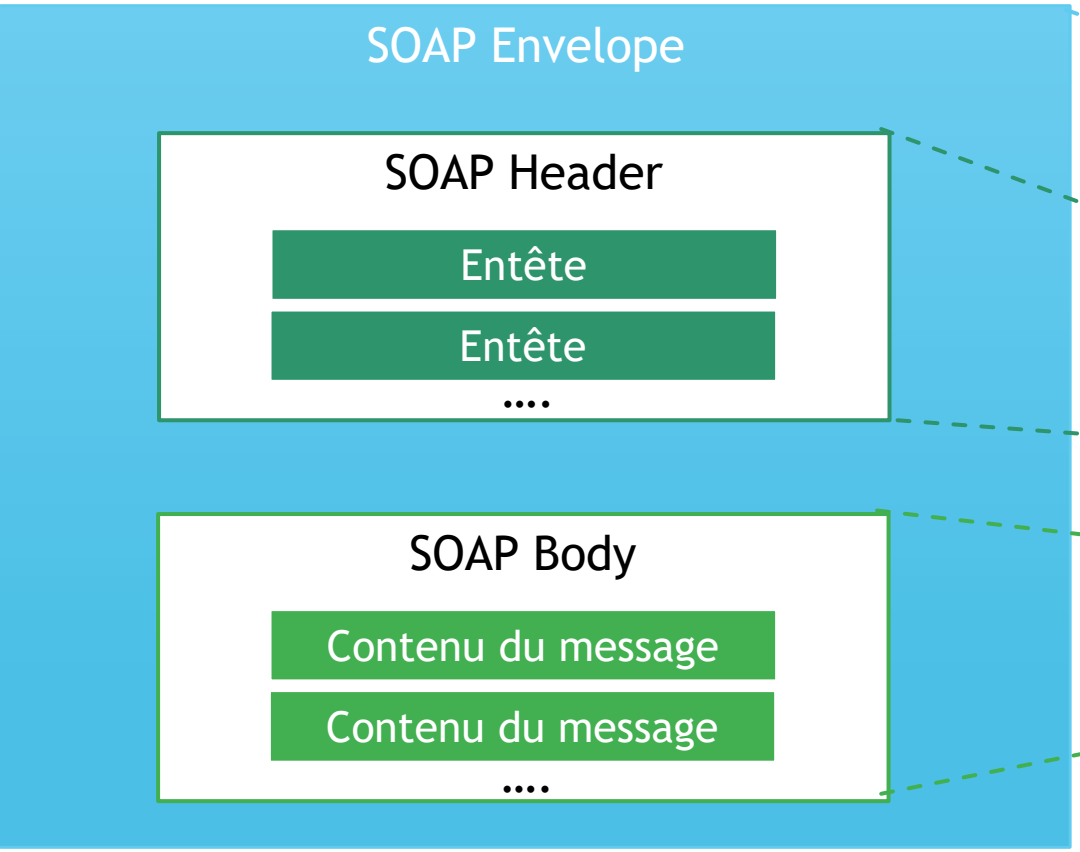
Out-optional-in
Contraire de in-optional-out.

Structure d’un Document SOAP

- Dans SOAP, les messages sont vus comme des **enveloppes** où l’application renferme les données à envoyer

- Une enveloppe SOAP contient une entête (*Header)* optionnelle, et un corps (*Body*) obligatoire
  - Leur contenu dépend de l’application, pas de la spécification SOAP

Structure d'un Document SOAP




```
<?xml version='1.0' ?>
<env:Envelope xmlns:env=

    "http://www.w3.org/20
        soap-envelope">

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

    <env:Body>
        .. <!-- Message
>
    </env:Body>

</env:Envelope>
```

Entête SOAP (*Header*)

- Paramètres annexes : Propriétés du message
  - Corrélation de messages
  - Informations d'authentification
  - Informations liées aux transactions
- Chaque application peut décider de la signification des headers
- Les nœuds intermédiaires peuvent ajouter/supprimer/traiter/transformer des headers
- Un nœud intermédiaire peut traiter un ou plusieurs éléments du header
  - Sauf spécification contraire (utilisation de *env:relay*), cet élément est retiré du message
  - *Env:role* permet de spécifier les nœuds destinés à traiter l'élément

Corps SOAP (*Body*)

- Endroit où les données XML spécifiques à l'application, et qui doivent être échangées, sont placées
- L'élément *Body* est obligatoire, mais peut être vide.
- Il peut contenir plusieurs fils (appelés *des entrées*), qui peuvent être:
  - **Données spécifiques à l'application**: information échangée dans le WS. Représentée par une méthode et ses arguments (requête) ou une ou plusieurs valeurs (réponses)
  - **Message de faute** : utilisé en cas d'erreur.
- Un message SOAP contient des données ou un message de faute, mais pas les deux à la fois.

Exemple de Message SOAP (*Request*)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header/>
    <env:Body>
          <commande xmlns="http://www.glaces.com">
                <parfum>Fraise</parfum>
                <quantite>16</quantite>
                <livraison>
                      <adresse>8 bvd N.Bohr Villeurbanne</adresse>
                      <destinataire>M2TI</destinataire>
                </livraison>
          </commande>
    </env:Body>
</env:Envelope>
```

Exemple de Message SOAP (*Response*)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header/>
    <env:Body>
          <confirmation xmlns="http://www.glaces.com">
                <heure>10h</heure>
            </confirmation>
       </env:Body>
</env:Envelope>
```

Fautes SOAP

- Utilisé lorsqu'une erreur survient
  - En général dans une réponse à un message reçu auparavant
- Se place dans la partie *Body*
- Le contenu est une explication de l'erreur:
  - Une valeur
  - Une explication textuelle
  - Un morceau de document spécifique à l'application
  - Le rôle du nœud à l'origine de l'erreur
    - Utile en présence de nœuds intermédiaires

Exemple de Faute SOAP

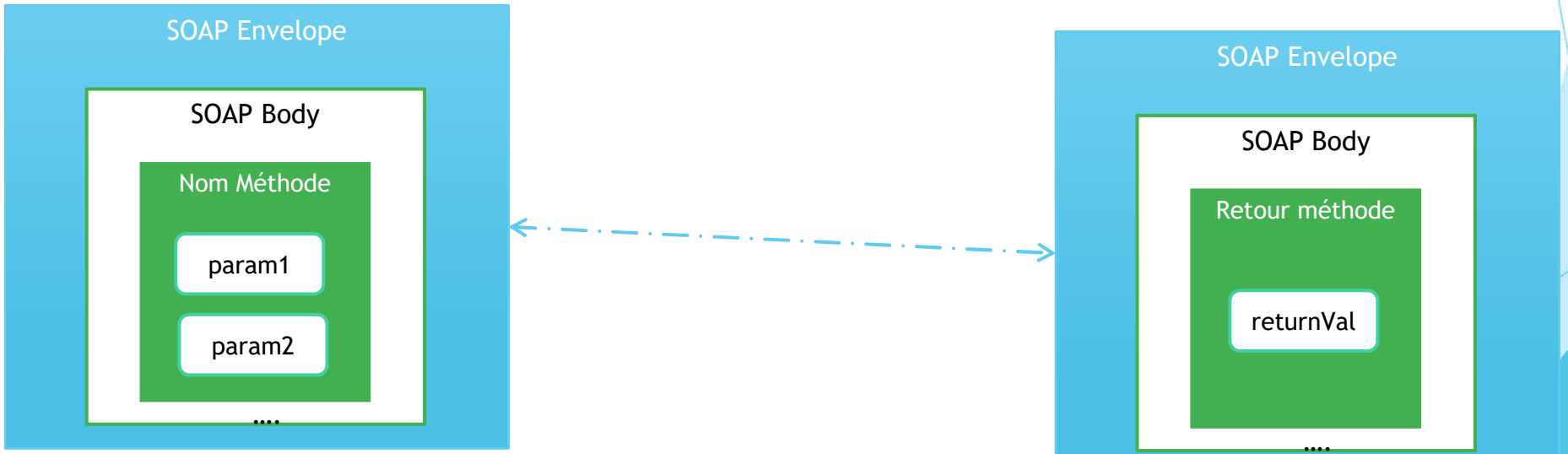
```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap- envelope">
       <env:Header/>
       <env:Body>
          <env:Code>
                    <env:Value>env:Receiver</env:Value>
             </env:Code>
          <env:Reason>
                    <env:Text xml:lang="fr-FR">Plus de glaces à la
fraise
                    </env:Text>
                    <env:Text xml:lang="en-GB">Nomore strawberry ice-
creams
                    </env:Text>
             </env:Reason>
       </env:Body>
</env:Envelope>
```

Modèles de Communication SOAP

- SOAP supporte deux styles de communication:
  - RPC (Remote Procedure Call)
  - Document (ou message)

Services SOAP *RPC-Style*

- Un service web RPC-style apparaît à l'application cliente comme un objet distant
- L'interaction entre un client et un service Web RPC est centrée autours d'une interface de ser
- Les clients expriment leurs requêtes sous forme d'appel de méthode avec un ensemble d'arguments, et le service retourne une réponse contenant une valeur de retour



Services SOAP *RPC-Style* : Exemple

Requête

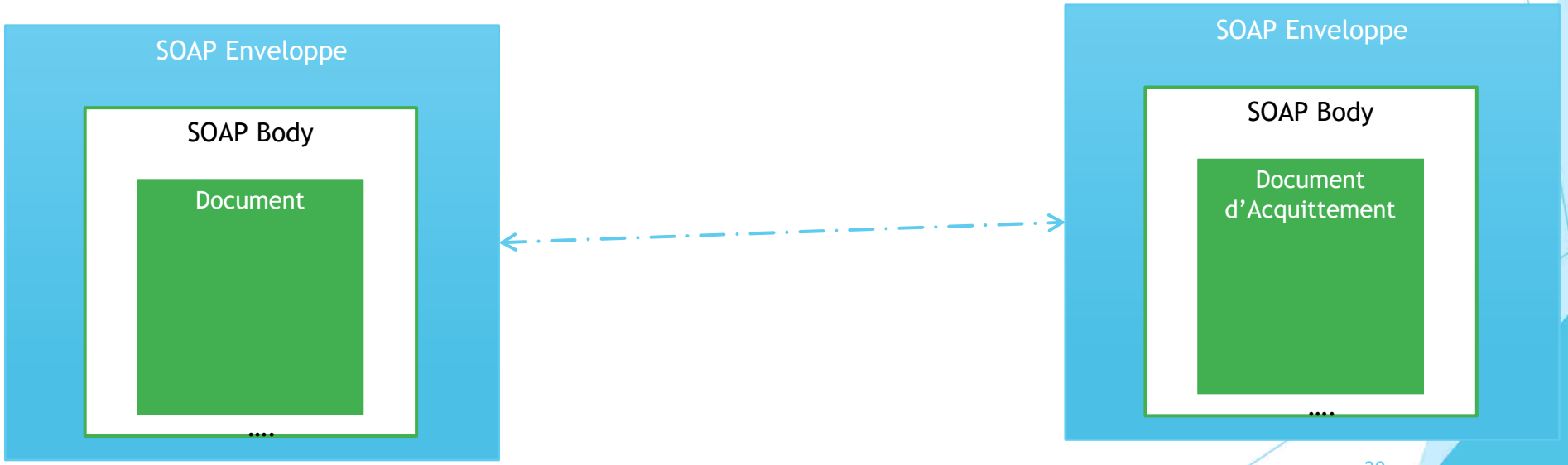
```
<env:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/soap-
envelope"
xmlns:m="http://www.plastics_supply.com/pro
duct-prices">

    <env:Header>
        <tx:Transaction-id
xmlns:t="http://www.transaction.com/transac
tions" env:mustUnderstand='1'>
            512
        </tx:Transaction-id>
    </env:Header>

    <env:Body>
        <m:GetProductPrice>
            <id> 450R6OP </id >
        </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

Réponse

```
<env:Envelope
xmlns:SOAP="http://www.w3.org/2003/05/so
ap-envelope"
xmlns:m="http://www.plastics_supply.com/
product-prices">

    <env:Header> </env:Header>


    <env:Body>

        <m:GetProductPriceResponse>
            <price> 134.32 </price>

        </m:GetProductPriceResponse>

    </env:Body>

</env:Envelope>
```

Services SOAP *Document-Style*

- Le corps SOAP contient un fragment de document
  - Ce n'est pas une structure XML
- L'environnement d'exécution SOAP accepte ce corps SOAP tel quel, et le transmet à l'applicat
réceptrice sans le changer.



Services SOAP *Document-Style* : Exemple

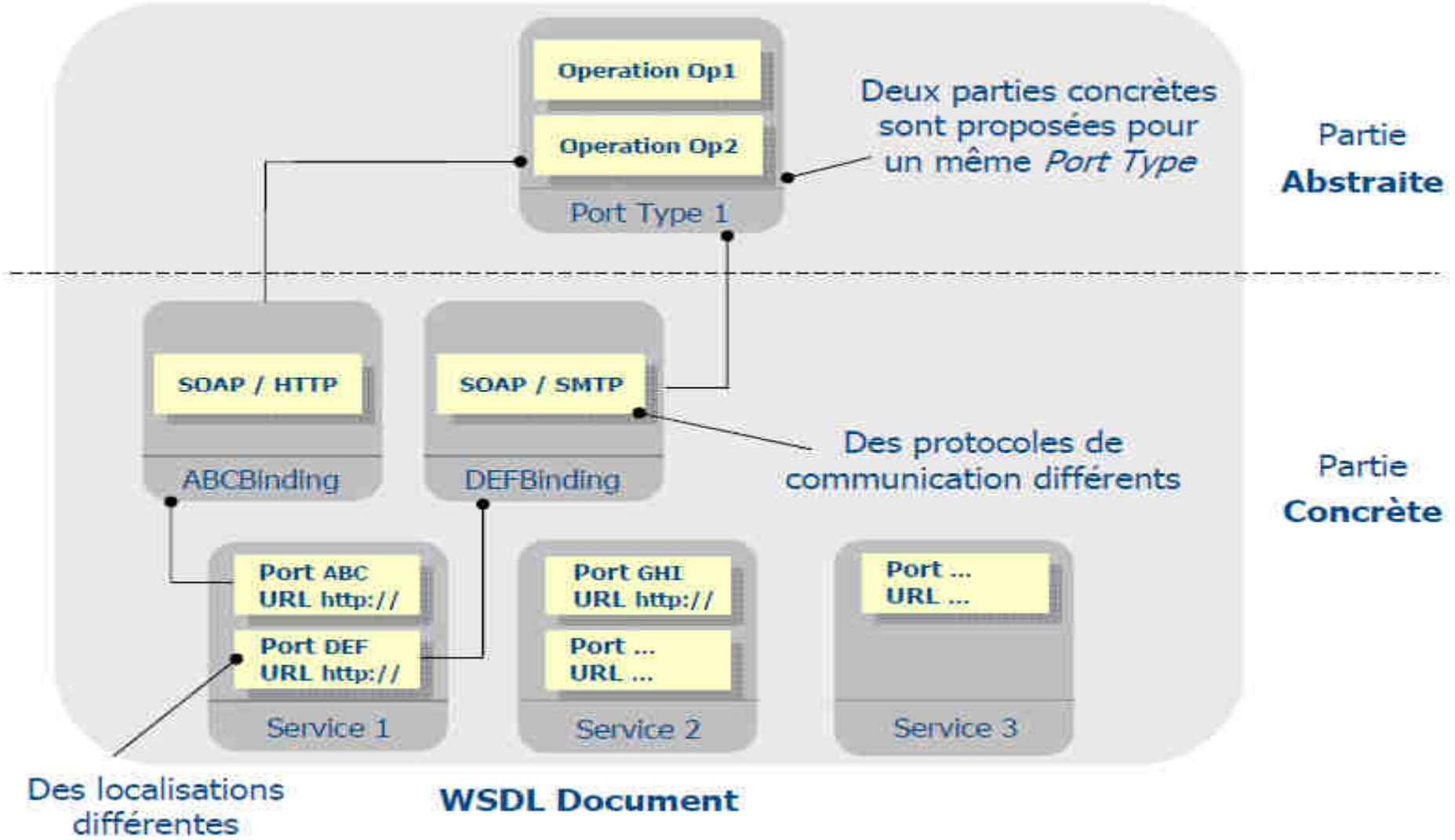
```
<env:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
     <env:Header>
          <tx:Transaction-id xmlns:t="http://www.transaction.com/transactions" env:mustUnderstand='1'>
               512
          </tx:Transaction-id>
     </env:Header>
     <env:Body>
          <po:PurchaseOrder oderDate="2004-12-02" xmlns:m="http://www.plastics_supply.com/POs">
               <po:from>
                    <po:accountName> RightPlastics </po:accountName>
                    <po:accountNumber> PSC-0343-02 </po:accountNumber>
               </po:from>
               <po:to>
                    <po:supplierName> Plastic Supplies Inc. </po:supplierName>
                    <po:supplierAddress> Yara Valley Melbourne </po:supplierAddress>
               </po:to>
               <po:product>
                    <po:product-name> injection molder </po:product-name>
                    <po:product-model> G-100T </po:product-model>
                    <po:quantity> 2 </po:quantity>
               </po:product>
          </ po:PurchaseOrder >
     </env:Body>
</env:Envelope>
```

WSDL
Web Services Description Language

- Permet d'automatiser les détails de la communication entre les partenaires
- Peut être découvert à travers les registres de services
- Il est possible de vérifier si la communication entre les services est conforme au WSDL
- Éléments WSDL
  - Types
  - Messages
  - Operation
  - Port Type
  - Binding
  - Port
  - Service

```
<?xml version="1.0"?>
<definitions name="StockQuote"
             targetNamespace="http://example.com/stockquote.wsdl" … >
    <!-- types -->
    <!-- messages-->
    <!-- port types -->
    <!-- bindings -->
    <!-- service -->
</definitions>
```

Organisation d'un document WSDL



WSDL : *Types*

- Définitions des types de données
- Utilisés pour décrire les messages échangés
- Utilisent les schémas XML comme type canonique

```
ema  target Namespace="http://example.com/stockquote.xsd"
     xmlns="http://www.w3.org/2000/10/XMLSchema">
<element  name="TradePriceRequest">
     <complexType>
          <all>
               <element  name="tickerSymbol"  type="string"/>
          </all>
     </complexType>
</element>
<element  name="TradePrice">
     <complexType>
          <all>
               <element  name="price"  type="float"/>
          </all>
     </complexType>
</element>
chema>
```

WSDL : *Messages, Operations & Port Type*

- Messages
  - Définitions abstraites et typées des données échangées
- Operations
  - Description abstraite d'une action
  - Fait référence à un message d'entrée et/ou de sortie
- Port Type
  - Collection d'opérations
  - Définition abstraite d'un service

```
<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"
</message>

<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
    </operation>
    <!-- More operations -->
</portType>
```

WSDL : *Binding, Port* & *Service*

- Binding
  - Protocole et format de données concrets pour un *port type* (ex: SOAP 1.1 over HTTP, SOAP encoding…)
- Port
  - Définit une extrémité de communication unique
  - Adresse de l'extrémité pour le binding
  - URL (pour HTTP), email (pour SMTP)
- Service : ensemble de ports connexes

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortTyp
    <soap:binding style="document"
          transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
        <soap:operation
              soapAction="http://example.com/GetLastTradePrice"/>
        <input> <soap:body use="literal" />
        </input>
        <output> <soap:body use="literal" />
        </output>
    </operation>
</binding>

<service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding
        <soap:address location="http://example.com/stockquote"/>
    </port>
</service>
```

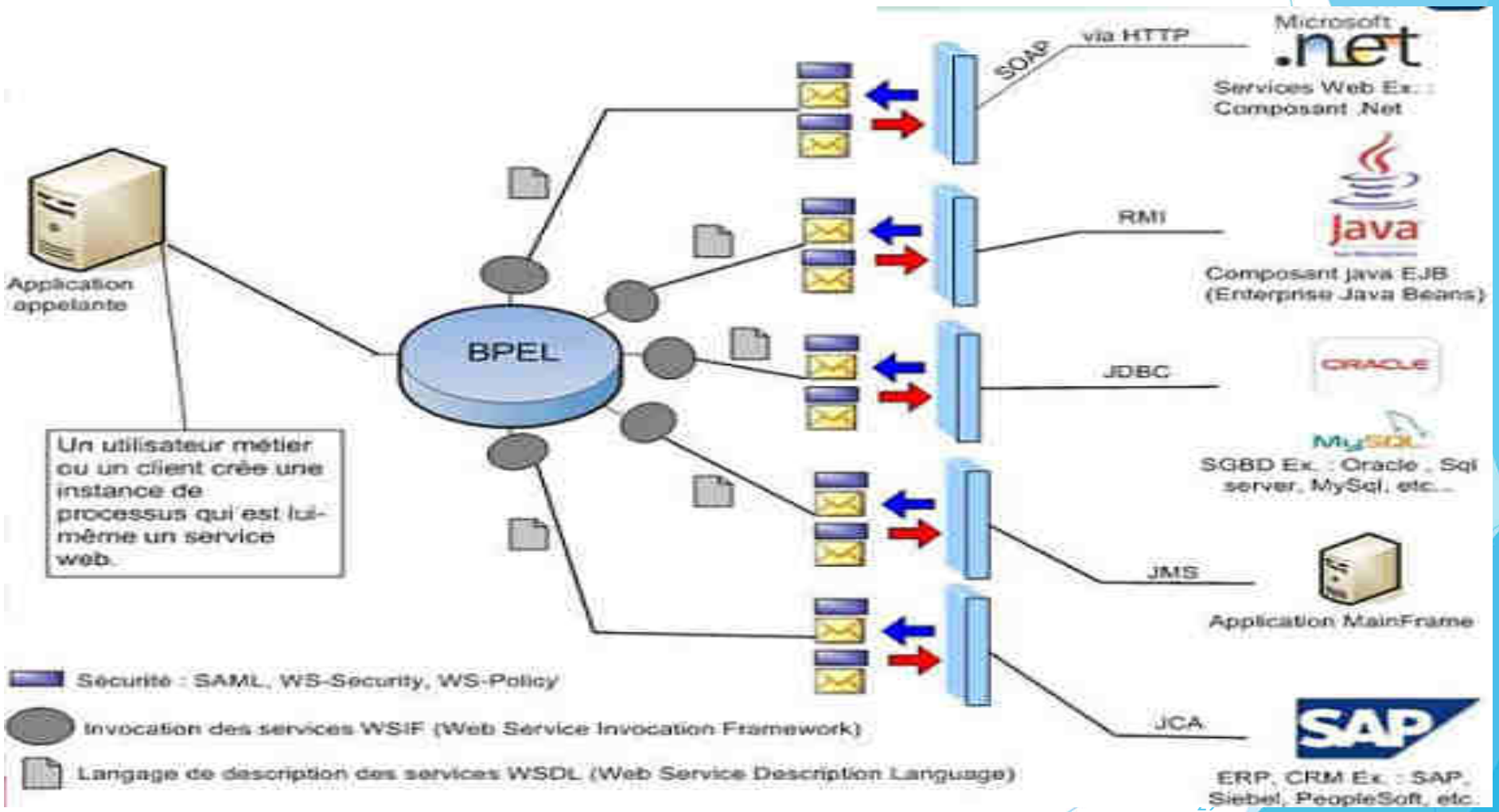
UDDI
Universal Description, Discovery and Integration

- Spécification pour la description et découverte des web services.
- Décomposé en 3 parties
  - White Pages (informations générales)
    - Noms, adresses, contacts, identifiants… des entreprises enregistrées.
    - Informations de catégorisation pour faciliter les recherches spécifiques selon le métier de l'entreprise
    - Entités : ***Business Entity***
  - Yellow Pages (catégories de services)
    - Détails sur le métier de l'entreprise, les services qu'elle propose
    - Entités: ***Business Service***
  - Green Pages (business rules)
    - Informations techniques sur les services proposés
    - Références vers les spécifications des services web, les détails nécessaires à leur utilisation
    - Entités : ***Binding Template*** *(coordonnées de services, URL et/ou tModel)*, et ***Technology Model*** *(tMode*
      description technique du service)

Le langage BPEL
Business Process Execution Language

- Standard de l'OASIS
- Norme permettant de décrire des processus en XML
- Propose les fonctions basiques d'un langage de programmation: sequence, flow, loop, switch…
- Identification des Instances de Process
- Gestion des transactions longue durée (scope,compensation)
- Gestion des erreurs

BPEL le chef d'orchestre



ESB : Définition

- Architecture de services distribuée, qui inclut un modèle de conteneur léger pour héberger des composants d'intégration comme services distants
- Permet de :
  - Délivrer des messages entre applications et services
  - Réaliser les transformations des données XML
  - Router les messages selon leur contenu
- Framework de sécurité flexible
- Infrastructure de gestion qui permet de configurer, déployer et gérer vos services distants

ESB:Synthèse

Connectivité	« Super-connecteur », supporte plusieurs protocoles de transport synchrones et asynchrone
Routage	Routage de messages basé sur des règles (de contenu, contexte..), et s'appuyant sur un annuaire de services et moteur de règles
Médiation	Adaptation du format des messages, protocole
Exposition de services	Transformation en service de tout composant ou traitement d'application
Agrégation simple de services	Agrégation simple de services de niveau N pour construire des services de nivea N+1. Pour les agrégations complexes, utilisation d'un moteur d'orchestration
Traitement d'évènements complexes	Création de règles de corrélation et de jointure d'évènements
Contrat de Service	SLA, QoS, gestion des priorités, sécurisation des messages, garantie de livraison.
Supervision et Audit	Audit, traçabilité, mesure, administration et exploitation.

Sources

- Cours
  - **M. Laaroussi**: *Web Services*, INSAT, Tunis
  - **E. Coquery** : *Services Web*, Université de Lyon
  - **M. Mecella** : *A very Short Introduction to Web Services*, Université de Rome