

# Les Architectures Orientées Services

SOA

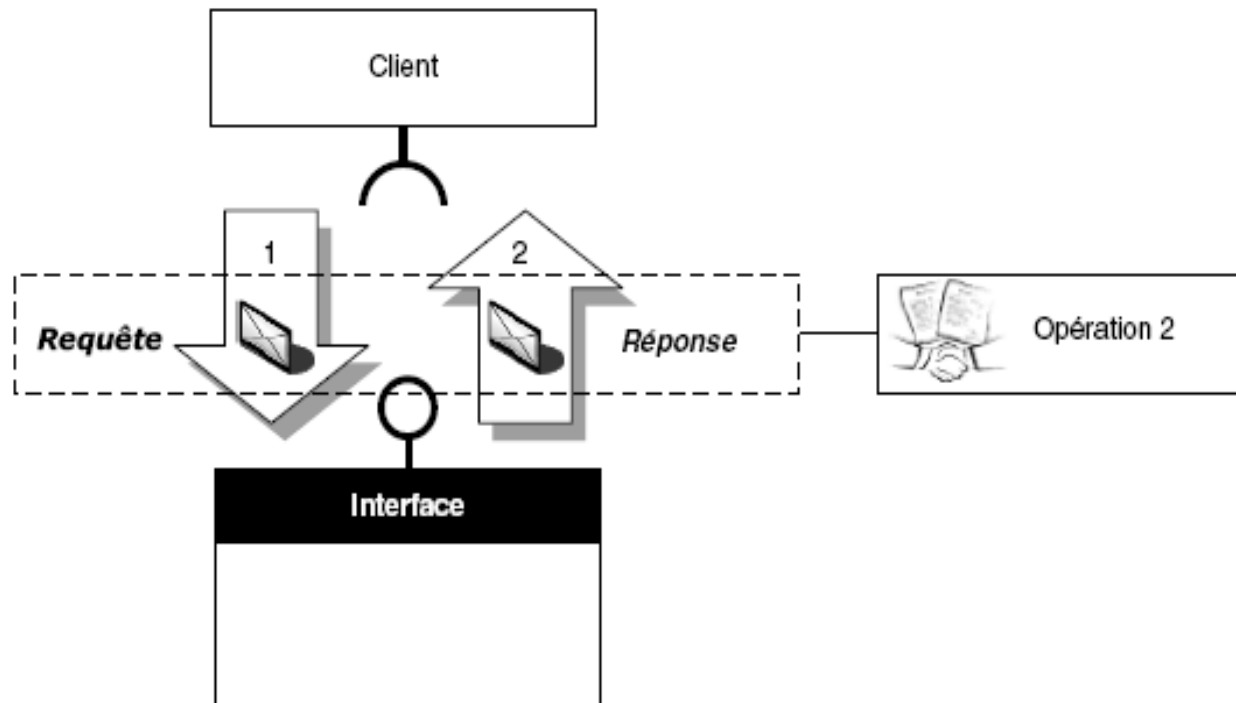
**Contrat et Message de Service - Suite**

# Plan

- ~~Contrat de service~~ - Décrire
- ~~Registre de services~~ - Découvrir
- Message - Communiquer
- Bus de messages - Transporter

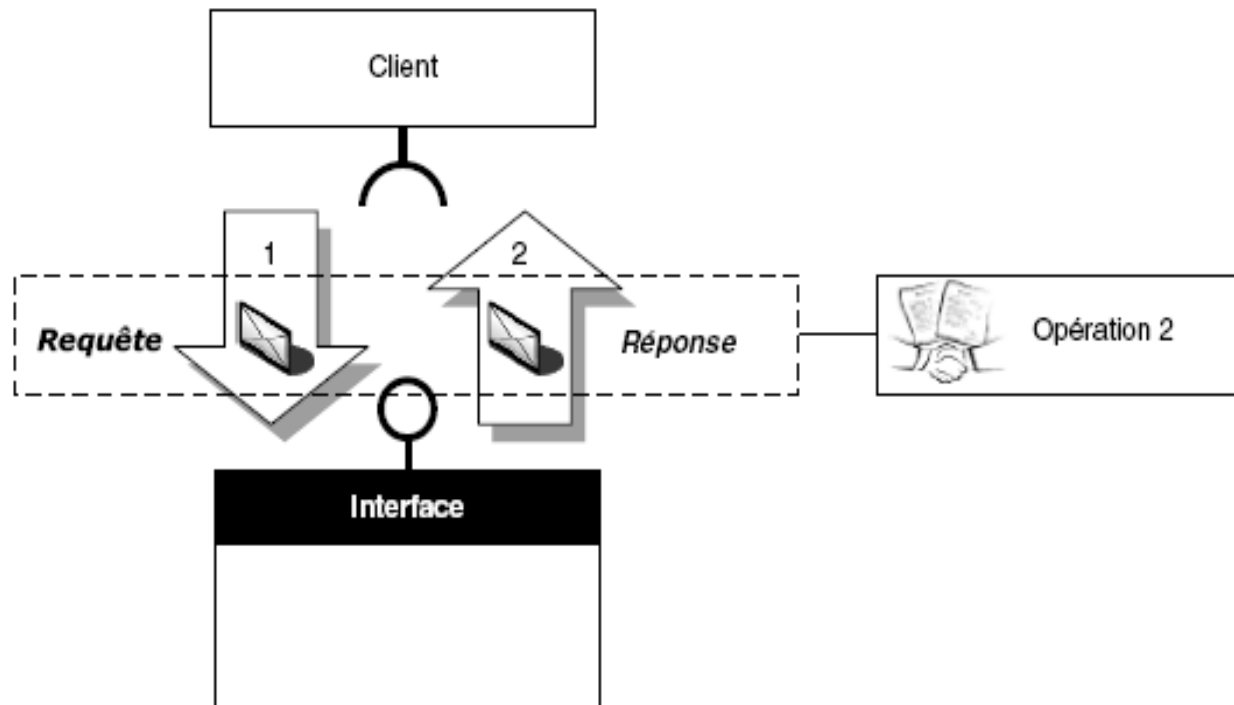
## Messages - Communiquer

- Lorsqu'un consommateur souhaite utiliser une opération d'un service, il lui transmet **un message** transportant sa requête.
- Le message est transporté sur le réseau via un **protocole de transport** (Binding).



## Messages - Communiquer

- Le consommateur doit connaître le protocole d'échange (transmission) du fournisseur.
- Le consommateur doit connaître le format d'échange du fournisseur.



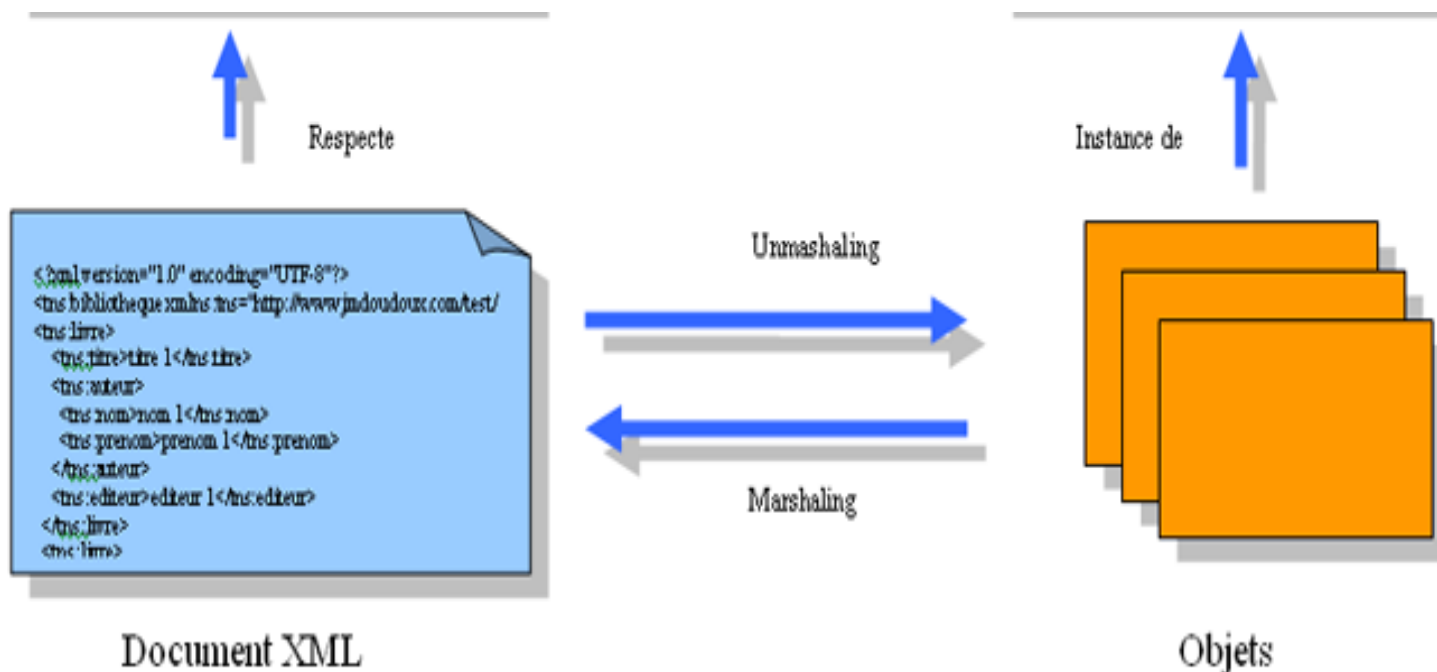
## Messages - Communiquer

### Exemple - **SOAP**

- SOAP pour *Simple Object Access Protocol*, c'est-à-dire protocole simplifié pour l'accès à des objets distants, une norme W3C.
- Est un protocole de transmission de messages en requête/réponse.
- Permet des appels de procédures à distance s'appuyant principalement sur le protocole HTTP et sur XML, donc indépendant de tout langage d'implémentation.
- Permet la **sérialisation**/désérialisation de tout objet métier, sous une forme XML utilisable quelle que soit la technologie d'implémentation.

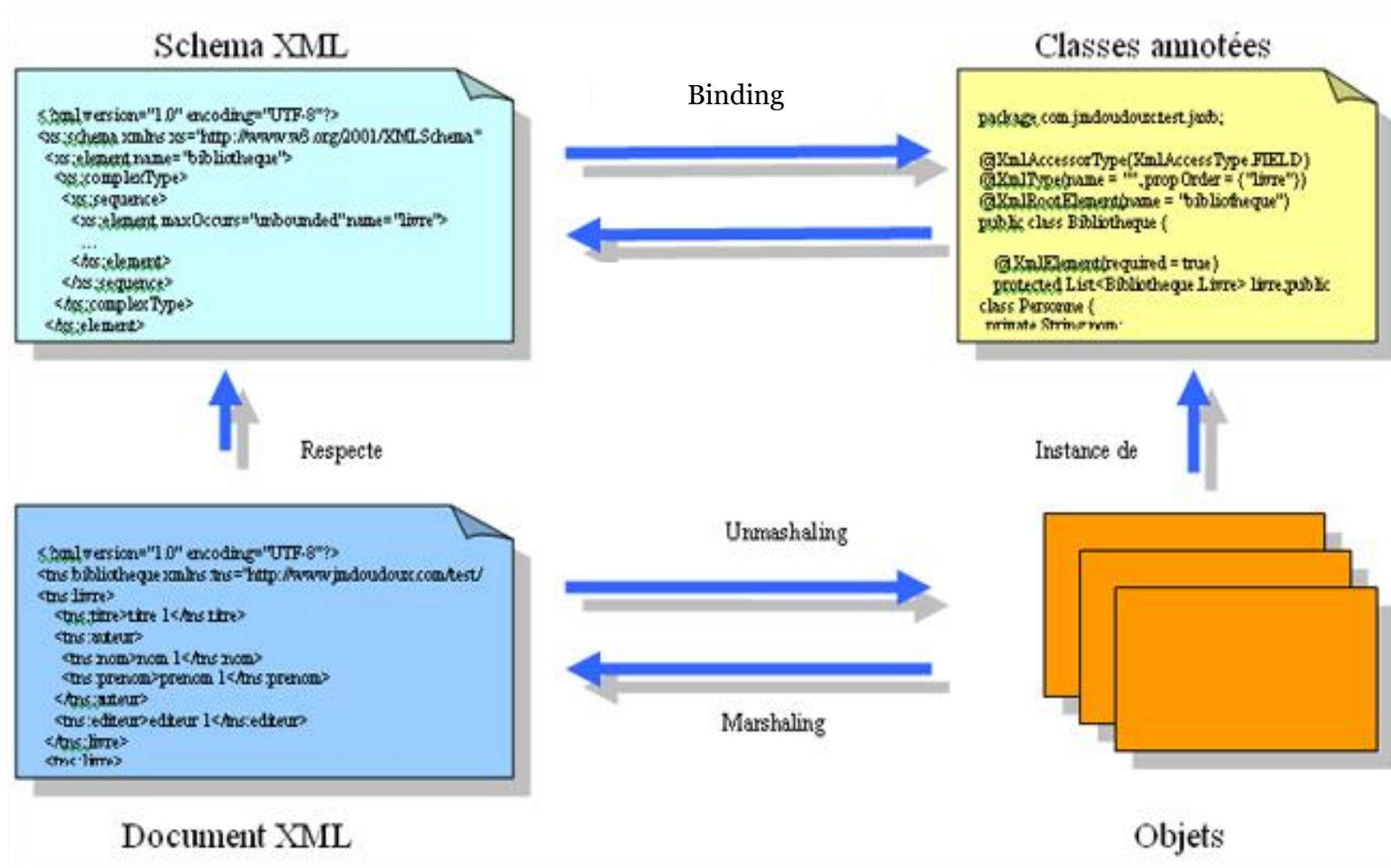
# Messages - Communiquer

## SOAP : Sérialisation/Désérialisation



# Messages - Communiquer

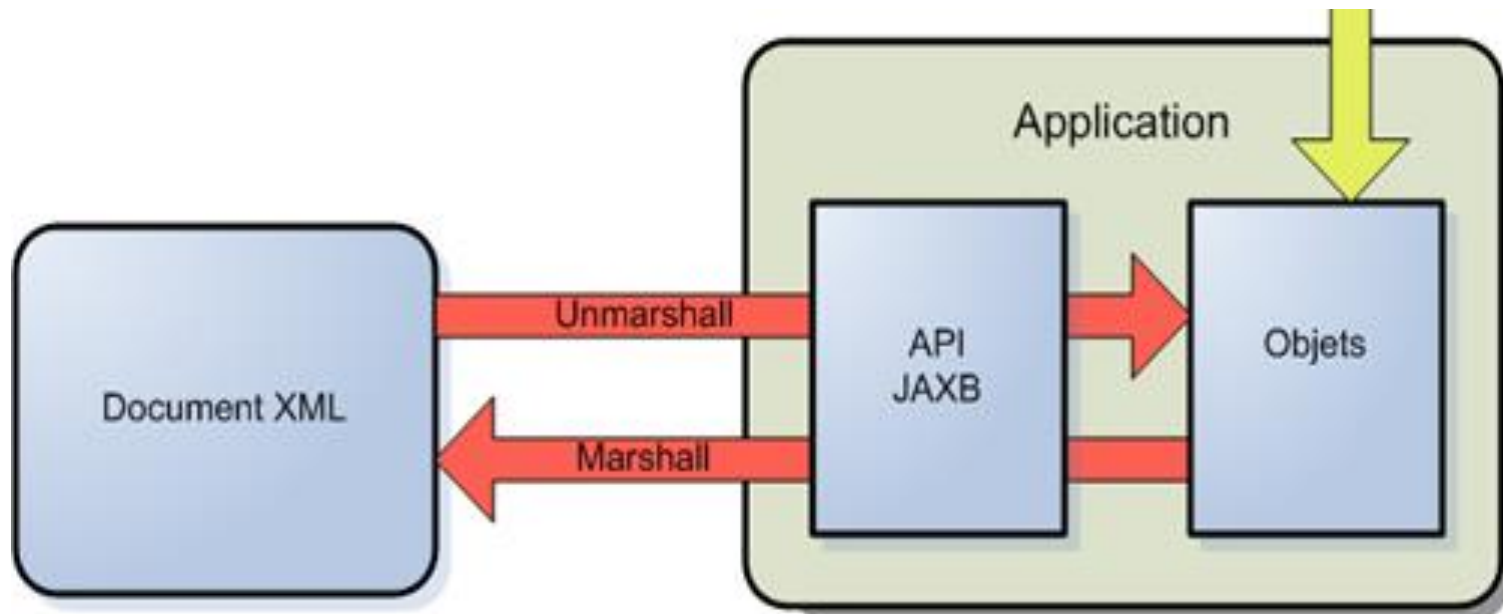
## SOAP : Sérialisation/Désérialisation



## Messages - Communiquer

### SOAP : Sérialisation/Désérialisation

- **Java Architecture for XML Binding - JAXB**: manipuler de l'XML en Java.
- Sérialiser/ désérialiser , binding, validation, etc.
- Se charge de générer les documents XML décrivant le service web tels que le fichier WSDL, les schémas XML et les messages Soap.





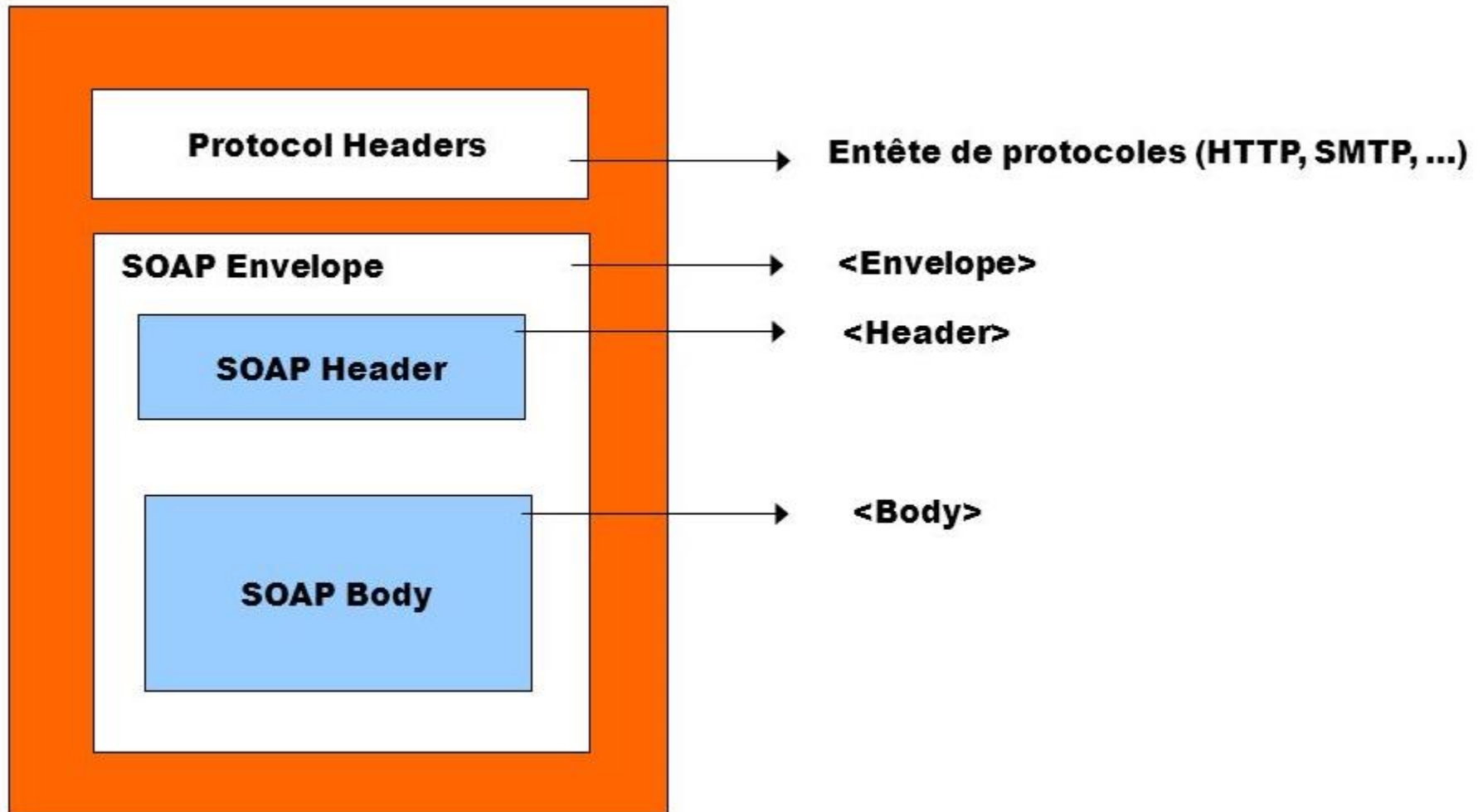
# Messages - Communiquer

## **SOAP**

- Fonctionnement Coté Client
  - Ouverture d'une connexion HTTP
  - Requête SOAP qui est un document XML décrivant :
    - ✓ Une méthode à invoquer sur une machine distante
    - ✓ Les paramètres de la méthode
  
- Fonctionnement coté Serveur
  - Récupère la requête + Conversion
  - Exécute la méthode avec les paramètres reçus
  - Renvoie une réponse SOAP (document XML) au client

## Messages - Communiquer

### SOAP : Structure



## Messages - Communiquer

### SOAP

- La requête et la réponse ont la même structure.
- Structure Requête ou Réponse SOAP:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">  
  <soapenv:Header>  
    ...  
  <soapenv:Header/>  
  <soapenv:Body>  
    <!-- Contenu de la Requête/Réponse -->  
  </soapenv:Body>  
</soapenv:Envelope>
```

## Messages - Communiquer

### **SOAP : L'enveloppe**

- L'enveloppe est la racine d'un message SOAP identifiée par la balise `<soapenv:Envelope>`
- La spécification impose que la balise et les sous balises soient explicitement associées à un namespace (espace de nommage).
- La spécification SOAP définit deux namespaces :
  - ✓ **SOAP-ENV** ou **soapenv** ou **S** : <http://schemas.xmlsoap.org/soap/envelope/>
  - ✓ **SOAP-ENC** : <http://schemas.xmlsoap.org/soap/encoding/>

# Messages - Communiquer

## **SOAP : L'en-tête**

- L'en-tête d'un message SOAP est utilisé pour transmettre des informations supplémentaires sur ce même message.
- L'en-tête est défini par la balise <soapenv:Header>
  - L'élément peut être facultatif
  - Doit être placé avant le corps
- Différents usages de l'en-tête ?
  - Informations authentifiant l'émetteur
  - Contexte d'une transaction
- Pour certains protocoles de transport, l'en-tête peut être utilisé pour identifier l'émetteur du message.
- Un message SOAP peut transiter par plusieurs intermédiaires avant le traitement par le récepteur final.

# La pile Services Web

## SOAP : L'en-tête

Injecter l'adresse IP à l'entête **HEADER** du message SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <ipAddress xmlns="http://tp.ws.soa.org/">
      127.1.1.0
    </ipAddress>
  </S:Header>
  <S:Body>
    <!-- Contenu de la Requête/Réponse -->
  </S:Body>
</S:Envelope>
```

### **SOAP : Le corps**

- Le corps d'un message SOAP est constitué d'un élément <soapenv:Body>
- L'élément <soapenv:Body> peut contenir soit :
  - Des informations adressées au destinataire du message SOAP
  - Une erreur en réponse à une requête (élément <soapenv:Fault>)
- Appel d'une opération représentée par une struct :
  - ✓ Le nom de la structure est celui de l'opération à appeler
  - ✓ Chaque paramètre de l'opération est défini comme un sous élément de la structure.

## Messages - Communiquer

### SOAP

#### ➤ Requête SOAP - Exemple :

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/  
                    xmlns:ns2="http://tp2.ws.soa.org/">  
  <soapenv:Header/>  
  <soapenv:Body>  
    <ns2:getUserByEmail >  
      <email>email@email</email>  
    </ns2:getUserByEmail>  
  </soapenv:Body>  
</soapenv:Envelope>
```



## Messages - Communiquer

### SOAP

#### ➤ Réponse SOAP - Exemple :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ns2:getUserByEmailResponse xmlns:ns2="http://tp2.ws.soa.org/">
      <userRetourne>
        <num>3</num>
        <email>email@email</email>
        <password>pass@pass</password>
      </userRetourne>
    </ns2:getUserByEmailResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

# Messages - Communiquer

## SOAP

- Structure **Réponse** SOAP – Fault Block en cas d'erreur:

```
<soapenv:Envelope ... >

  <soapenv:Body>
    <soapenv:Fault>
      <faultcode xsi:type="xsd:string">
        SOAP-ENV:Client
      </faultcode>

      <faultstring xsi:type="xsd:string">
        Description de l'erreur
      </faultstring>
    </soapenv:Fault>
  </soapenv:Body>


</soapenv:Envelope>
```

## Messages - Communiquer

### SOAP : Le corps

Exemple : corps d'un message SOAP pour appeler l'opération **addUser**

```
<soapenv:Body>  
  <ns2:addUser xmlns:ns2="http://tp2.ws.soa.org/">  
    <num>3</num>  
    <email>email@email</email>  
    <password>pass@pass</password>  
  </ns2:addUser>  
</soapenv:Body>
```



**struct**

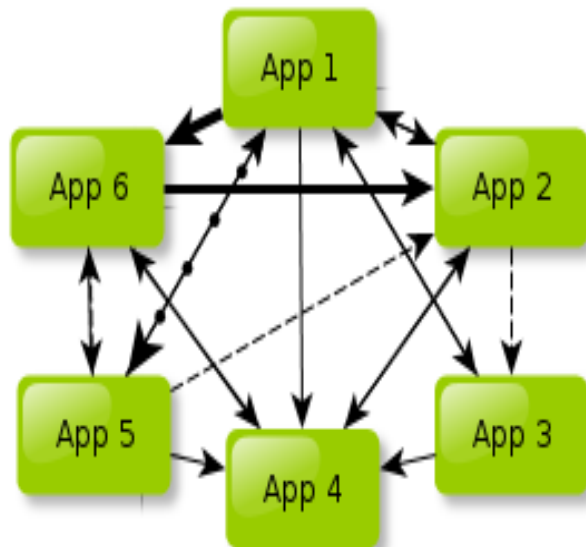
## Messages - Communiquer

### SOAP : Le corps

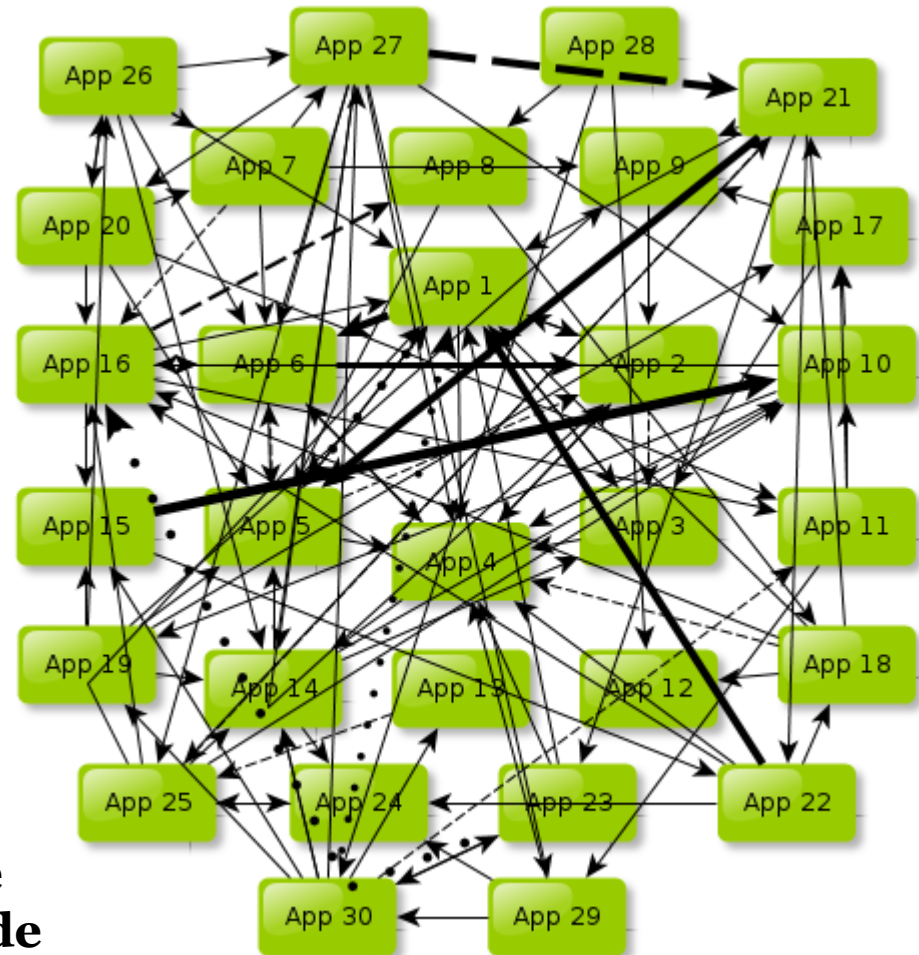
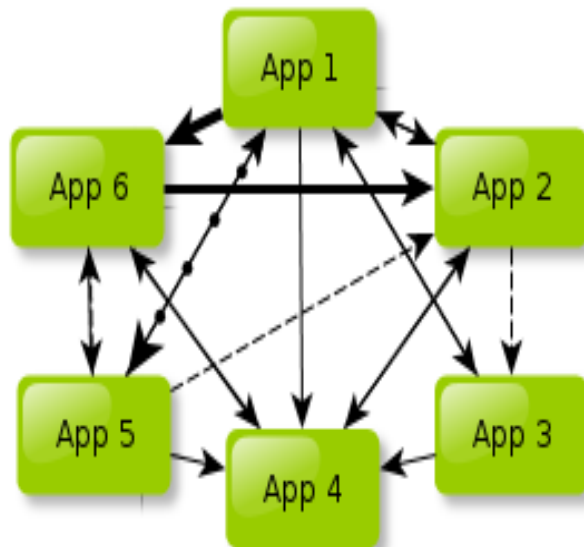
Exemple : corps d'un message SOAP pour le résultat d'une opération **addUser** qui n'est pas exposée par le service web - Fault

```
<soapenv:Body>
  <soapenv:Fault>
    <faultcode xsi:type="xsd:string">soapenv:Server</faultcode>
    <faultstring xsi:type="xsd:string">
      Failed to locate method (addUser) in class ...
    </faultstring>
  </soapenv:Fault>
</soapenv:Body>
```

## Messages - Communiquer



## Messages - Communiquer

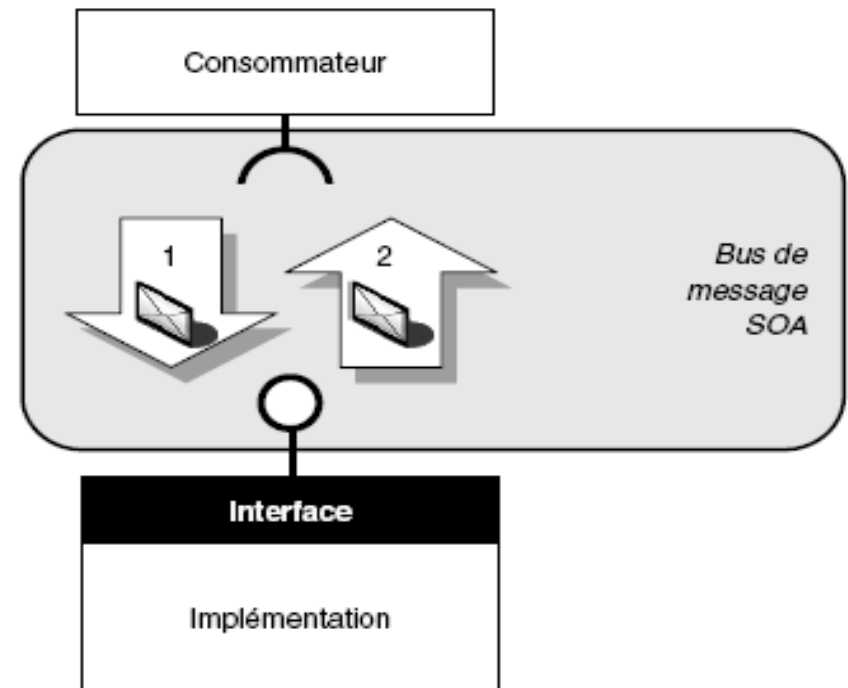


➔ Ceci implique la mise en place d'un middleware adapté : un **bus de messages**.

## Messages - Communiquer

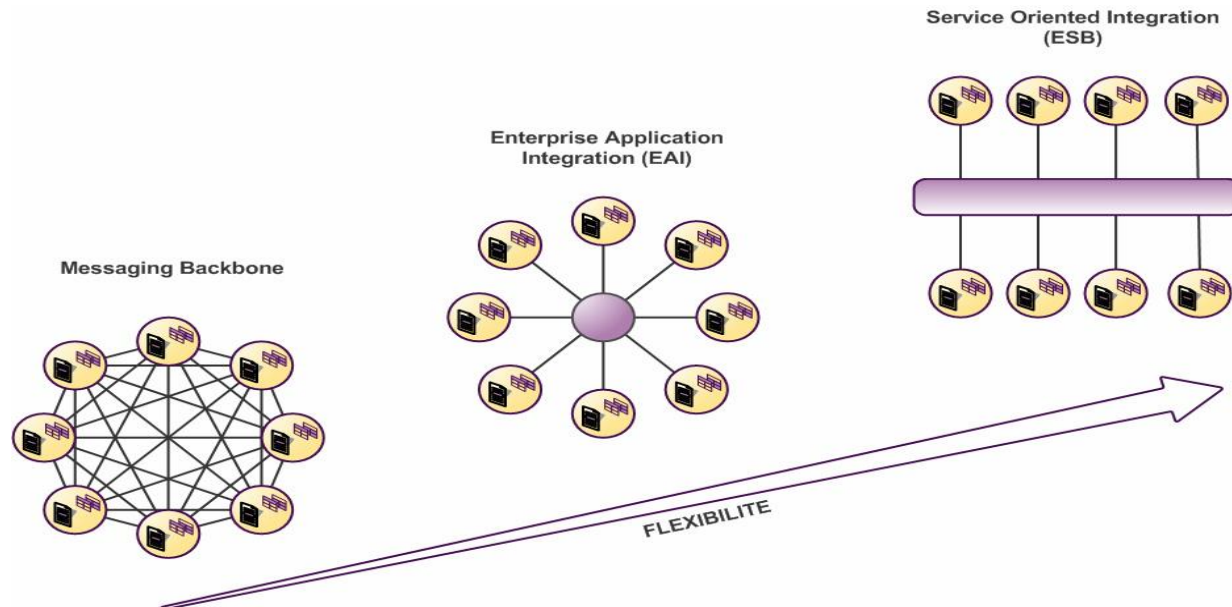
- Différents protocoles de transport peuvent être utilisés pour véhiculer un message HTTP, SMTP, ou JMS.
- Pour qu'un message soit transportable sur cette variété de protocoles, un langage commun et standard est privilégié (basé XML). ➔ Enveloppe normalisée.
- Un message échangé doit être conforme au contrat de service.

➔ Ceci implique la mise en place d'un middleware adapté : le **bus applicatif de messages**.



## Bus de messages - ESB

- Bus de messages SOA = ESB – Entreprise Service Bus.
- Les ESB sont les héritiers directs des EAI : Bea, Tibco, Oracle, IBM, etc.
- Ils sont aujourd'hui la technologie d'intégration interapplicative privilégiée pour la mise en œuvre d'une AOS.
- L'ESB est un composant logiciel chargé d'assurer les échanges d'informations entre services.





## Bus de messages - ESB

- ESB garanti **l'interopérabilité** entre des services hétérogènes.
- Il permet de rendre les consommateurs des services aussi indépendants que possible :
  - ✓ Du protocole de communication utilisé par le consommateur et/ou par le service : le consommateur doit pouvoir accéder au service via HTTP, mais aussi via FTP, JMS, SMTP, RMI, .NET Remoting etc.
  - ✓ De la technologie de déploiement des services : que ce service soit déployé comme Web Service, composant .NET, comme EJB, en Java ou PHP, le consommateur doit – dans l'idéal – y accéder de la même façon.
  - ✓ Des systèmes d'exploitation et des langages.
  - ✓ De la localisation des services.

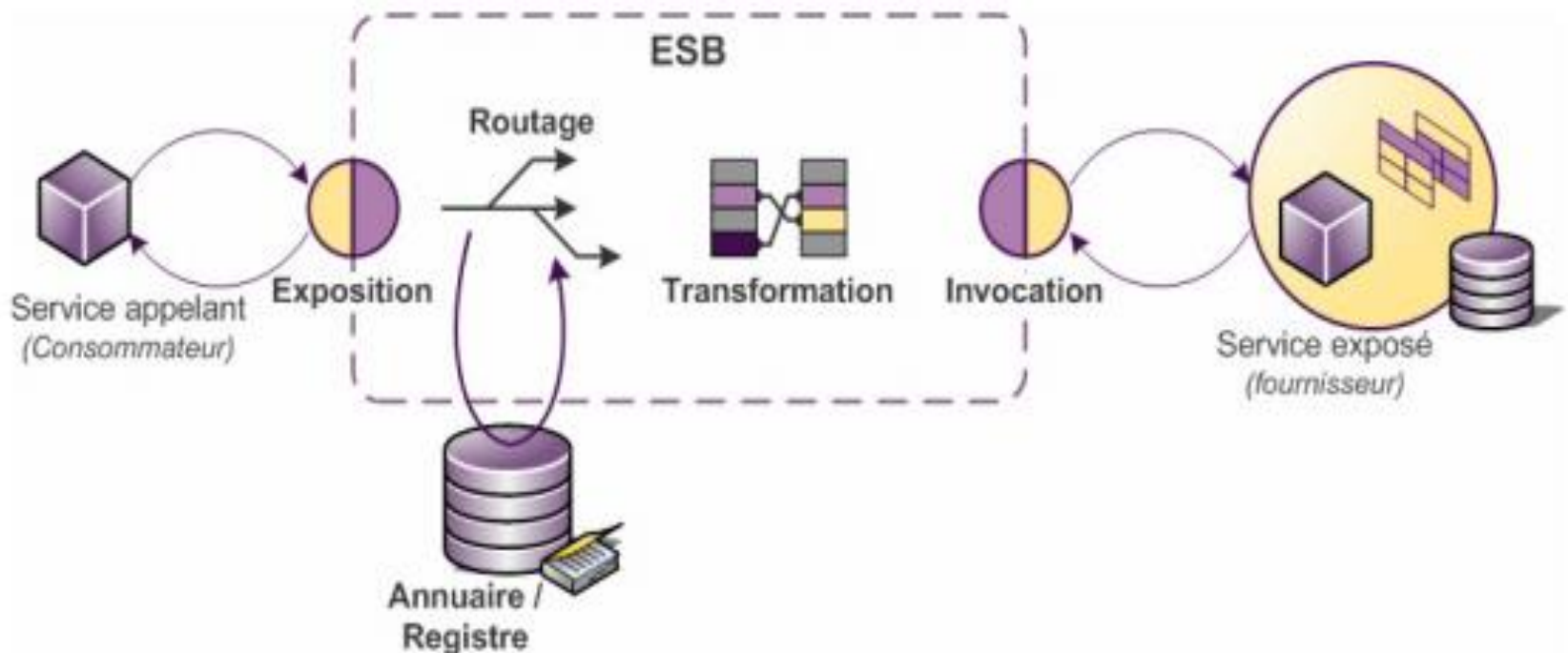
## Bus de messages - ESB

- L'**ESB** s'appuie sur les **responsabilités** suivantes qu'il se doit d'implémenter :
  - ✓ La **découverte** dynamique : les services ainsi que la description associée sont enregistrés dans un annuaire partagé.
  - ✓ **L'orchestration** des services : orchestrer automatiquement les services nécessaires à l'implémentation des processus métiers.
  - ✓ La **création**, **l'hébergement**, et la **distribution** forte des services: les services sont distribués sur le réseau de l'entreprise ou sur Internet.
  - ✓ La **communication** par messages : les services s'échangent des messages + adaptateurs de protocoles de communications.
  - ✓ La médiation et le **routage** intelligent qui découple l'expéditeur du message de son destinataire.
  - ✓ Les **transformations** des messages échangés entre fournisseurs et consommateurs .

## Quelques cas d'utilisation d'un ESB

### Couplage lâche

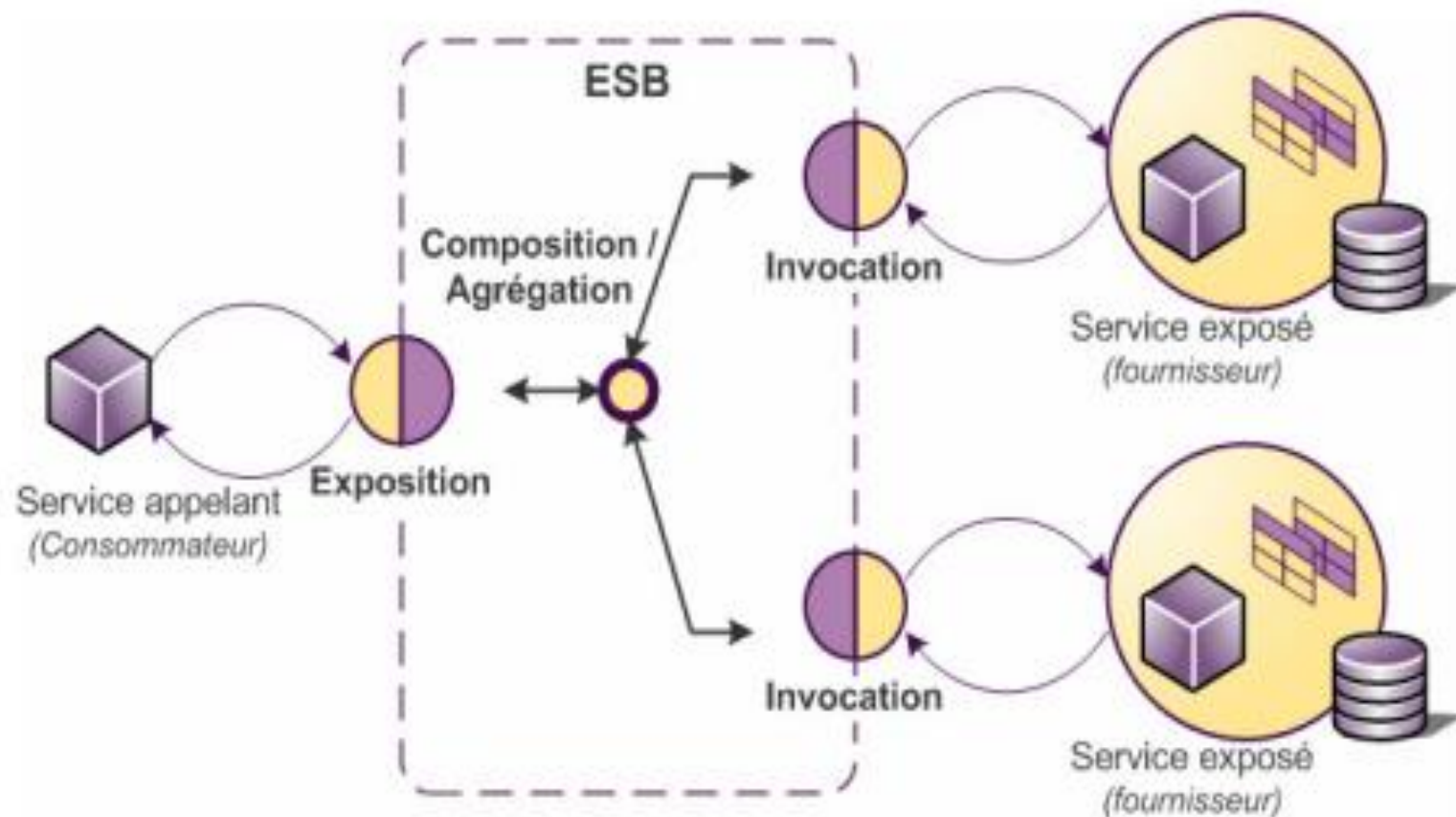
- Le service invoqué peut être exposé dans une autre technologie, utiliser une autre représentation des données, etc.
- L'ESB va prendre en charge ces adaptations pour libérer le consommateur appelant des adhérences avec le service qu'il appelle.



## Quelques cas d'utilisation d'un ESB

### Composition de services

- L'ESB expose un service virtuel qu'il construit par composition de plusieurs autres services.

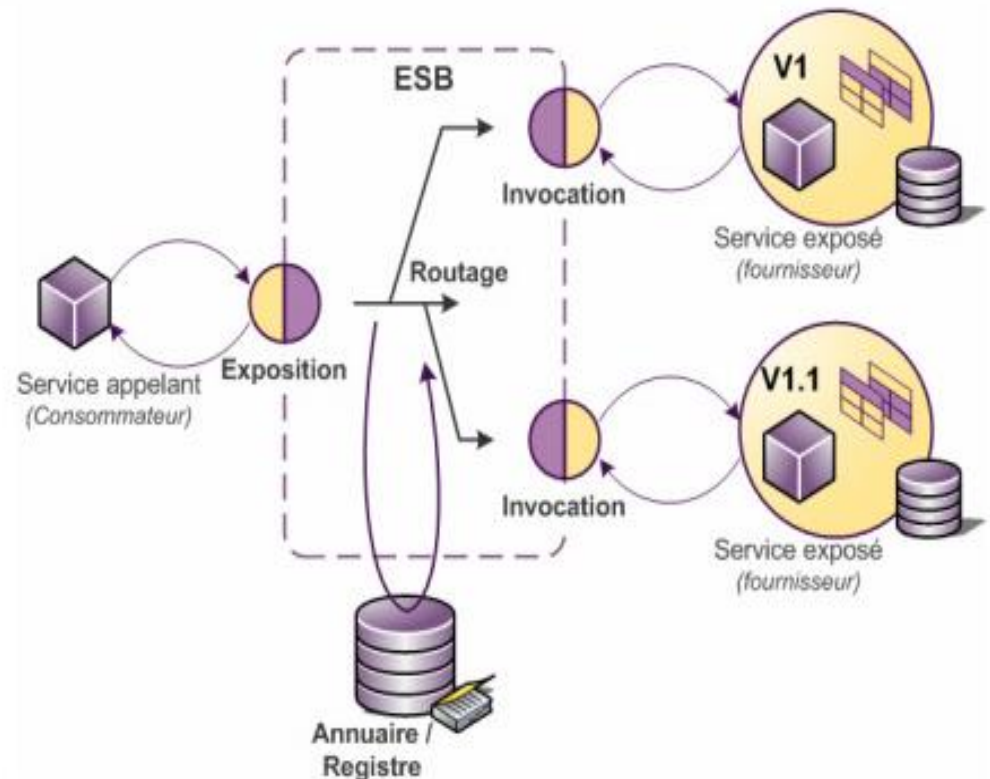


# Quelques cas d'utilisation d'un ESB

## Gestion de version

Plusieurs situations peuvent se présenter :

- Les versions sont incompatibles entre elles : le choix d'une version se fait par **routing**.
- La nouvelle version est une extension compatible avec la version précédente :
- ✓ Ici L'ESB effectue l'appel vers la nouvelle version en appliquant une **transformation** du format d'entrée, du format de sortie ou des deux formats.



## Bus de messages - ESB

### **Appeler un service - Modes d'échange de messages**

- Le bus met à disposition des consommateurs de services plusieurs modes d'appel (ou mode d'interaction) de ces services :
  - Mode synchrone/conversationnel (suite de requêtes et de réponses).
  - Mode asynchrone « *one way* ».
  - Mode asynchrone «abonnement/notification ».

## Bus de messages - ESB

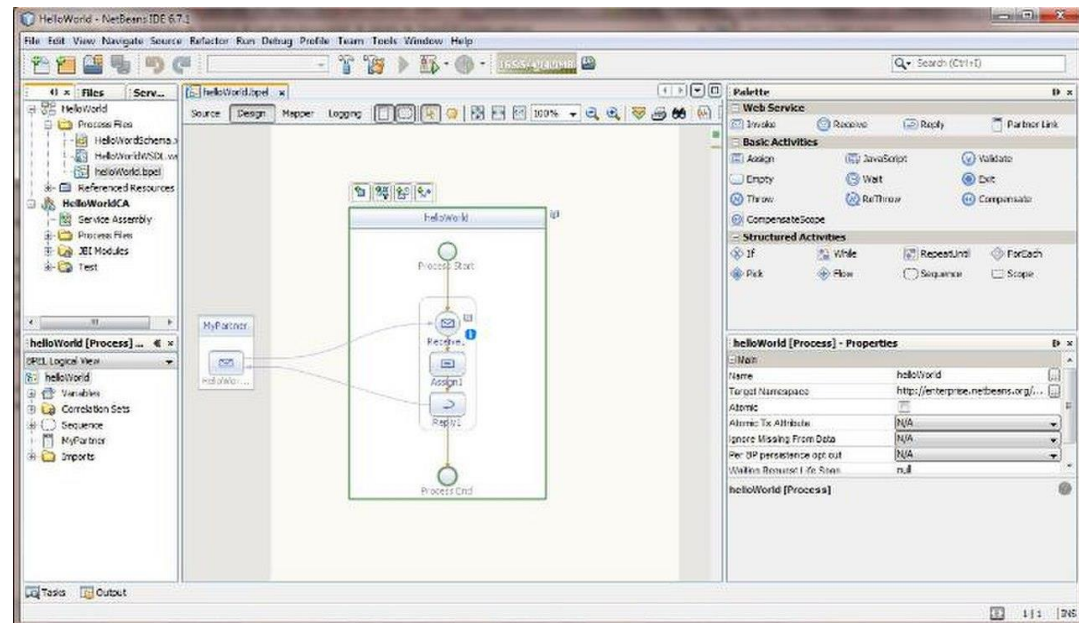
### ... Et la **sécurité**

- Le bus répond aux problématiques de sécurité :
  - ✓ Comment s'assurer de l'identité du fournisseur ou du consommateur ?
  - ✓ Comment définir et exposer les droits d'accès à un Service ?
  - ✓ Comment assurer la confidentialité des échanges ?
  - ✓ Comment assurer la conservation des messages lors d'un échange sensible mettant en jeu plusieurs partenaires ?
  - ✓ Etc.
- En mettant en place une sécurité par le réseau (Firewall, VPN, etc.).
- En utilisant une sécurité au niveau protocolaire (SSL, etc.).
- En déployant des systèmes d'authentification (clés, chiffrements etc.).
- Etc.

# Bus de messages - ESB

## Exemple – OpenESB

- Un ESB Gratuit et open source. <http://www.open-esb.net/>
- OpenESB Studio est un IDE qui propose de composer graphiquement les services : il s'agit de décrire par des objets graphiques le contenu du service.
- Netbeans, Glassfish, WSDL, XSD, BPEL, etc.
- Démo :





## Plateforme SOA

**Services**

**Services**

## Plateforme SOA

Services

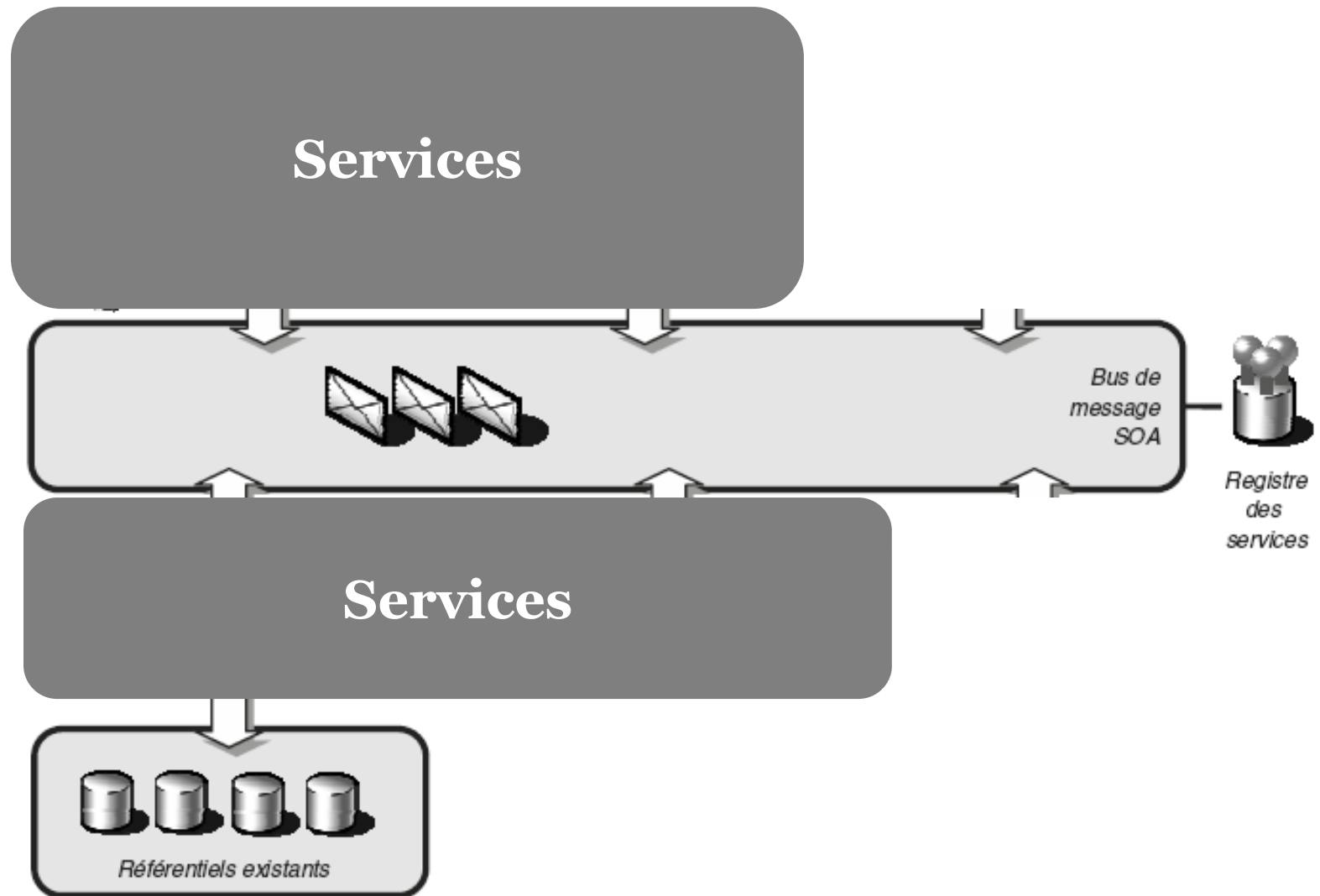


*Registre  
des  
services*

Services

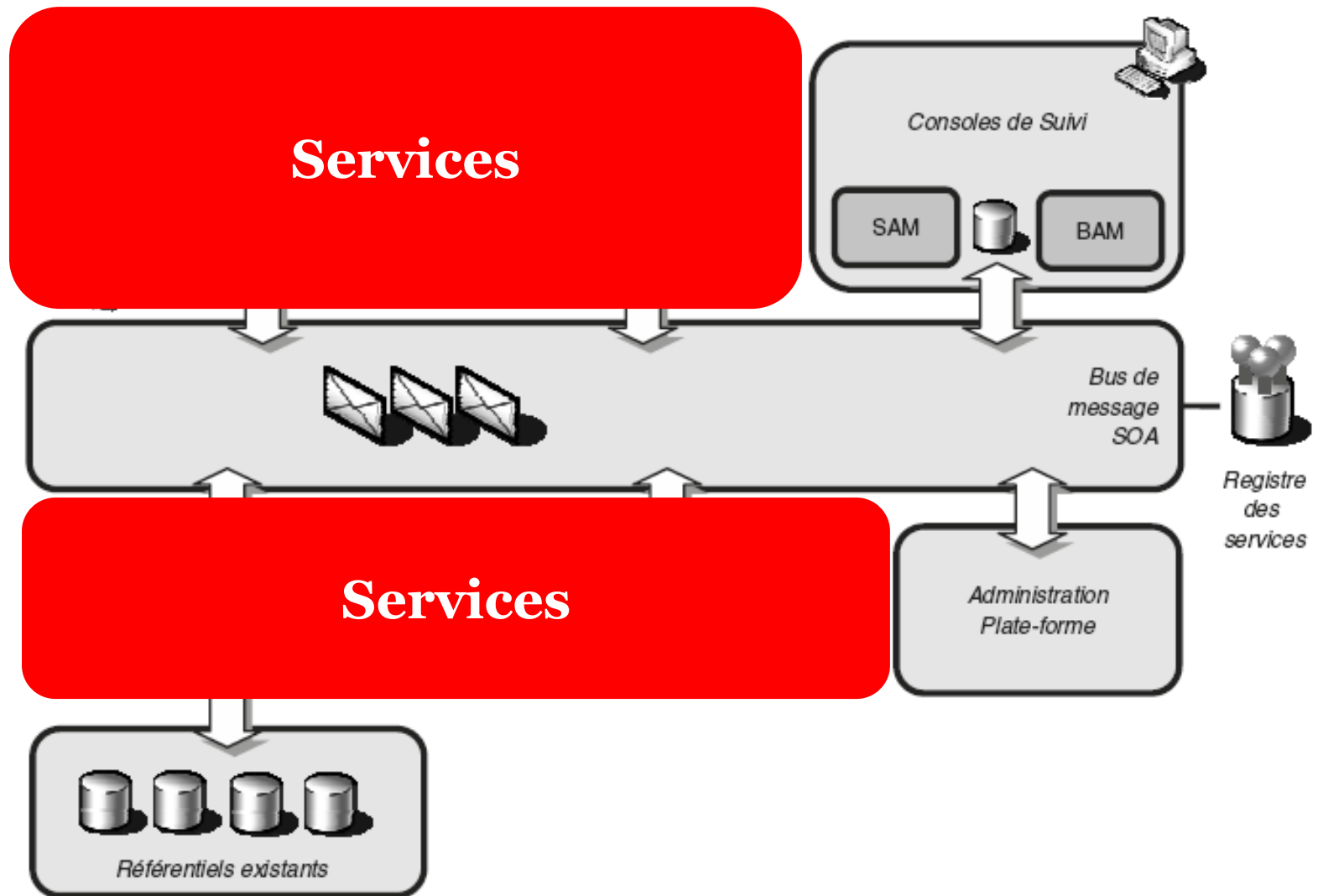


## Plateforme SOA

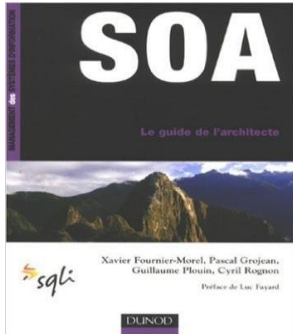


# Plateforme SOA

Business Activity Monitoring  
Service Activity Monitoring



# Ressources



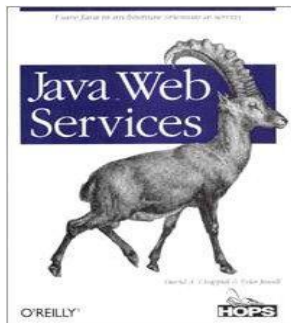
## **Le guide de l'architecte du SI**

- ✓ Auteur : Xavier Fournier-Morel, Pascal Grosjean, ...
- ✓ Éditeur : Dunod
- ✓ Edition : Octobre 2006 - 302 pages - ISBN : 2100499726



## **SOA Principles of Service Design**

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



## **Java Web Services**

- ✓ Auteur : David Chappell & Tyler Jewell
- ✓ Éditeur : O'Reilly
- ✓ Edition : Mars 2002 - 276 pages - ISBN : 0-596-00269-6

# Ressources

## **Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing**

- ✓ Auteur : Armando Fox and David Patterson
- ✓ Éditeur : Strawberry Canyon LLC
- ✓ Edition : Aout 2012 - 412 pages - ISBN : 0984881212

Livre blanc SOA : Architecture Logique : Principes, structures et bonnes pratiques Auteur: Gilbert Raymond.Version 2.

Cours – Mickael Baron – SOA et Microservices

- ✓ [http://mbaron.developpez.com/#page\\_soa](http://mbaron.developpez.com/#page_soa)

Livre Blanc - Comprendre et savoir utiliser un ESB dans une SOA

- ✓ <http://xebia.developpez.com/tutoriels/java/esb-soa/#LII-B-2>