

Les Architectures Orientées Services

SOA

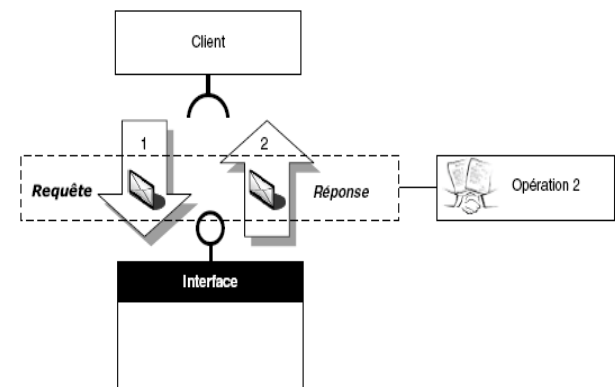
Contrat, Registre, Bus, et Messages de Services - Suite

Plan

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

Messages - Communiquer

- **Communiquer et échanger** des données dans le contexte de SOA indépendamment des plateformes et langages. Utilisation de **standards**.
- Les services communiquent uniquement par **messages** - Appels via le réseau vu que les services sont distribués en SOA.
- 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**.
- Le **Binding** indique le modèle à utiliser pour communiquer avec le service : Protocole de transport et format des messages.



Messages - Communiquer - Exemple

SOAP - Protocole de communication

- SOAP pour *Simple Object Access Protocol*, c'est-à-dire protocole simplifié pour l'accès à des objets distants, une norme W3C.
- SOAP est un protocole d'invocation de méthodes sur des services **distant**s.
- Est un protocole de **transmission** de messages en requête/réponse.
- Basé sur **XML**, SOAP a pour principal objectif d'assurer la **communication entre machines**. Plus populaire et utilisé que XML-RPC.
- Le protocole permet d'appeler une méthode RPC et d'envoyer des messages aux machines distantes via **HTTP**.
- Permet la **sérialisation**/désérialisation de tout objet métier, sous une forme XML utilisable quelle que soit la technologie d'implémentation.

Messages - Communiquer - Exemple

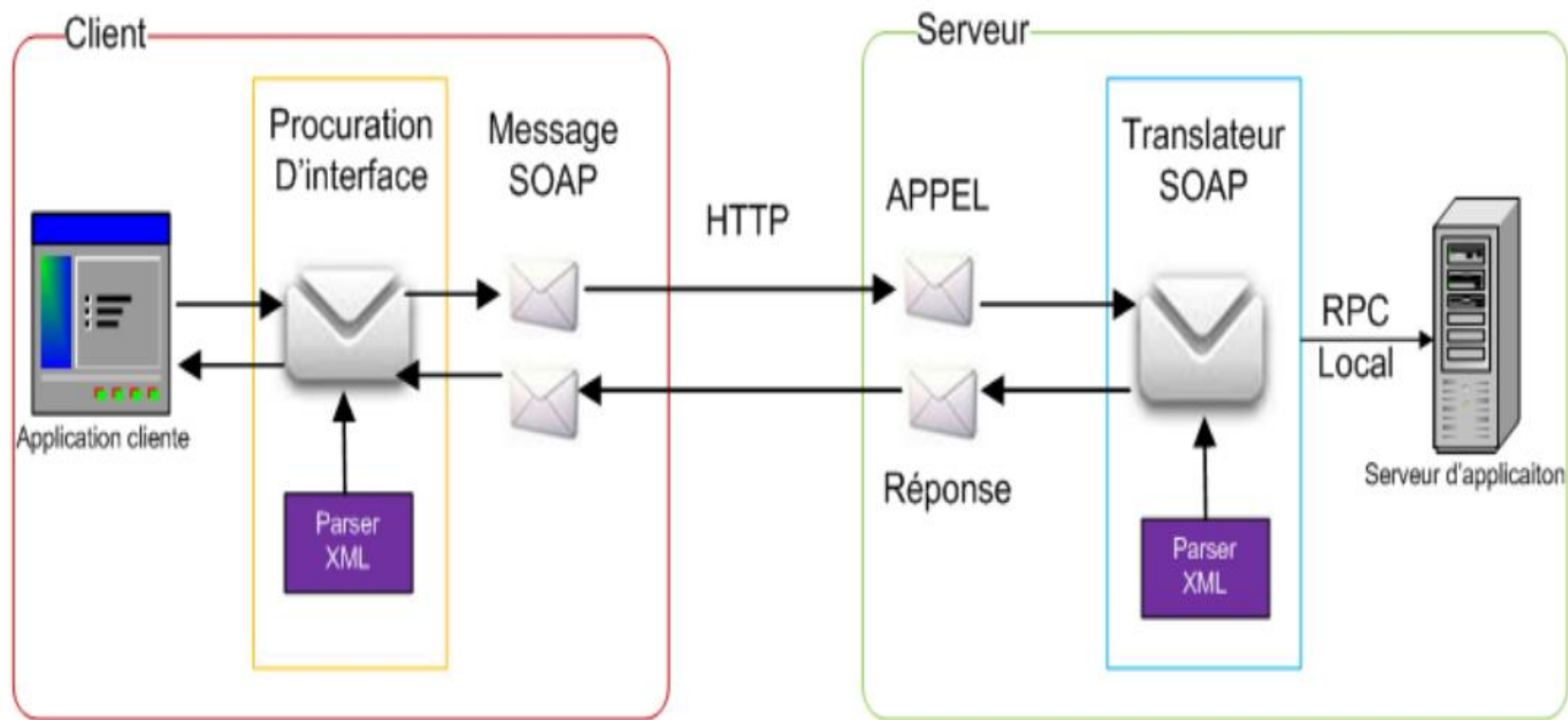
SOAP - Protocole de communication

- Fonctionnement Coté Client
 - Ouverture d'une connexion HTTP
 - **Requête SOAP** qui est un document XML **sérialisé** 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 sérialisé) au client

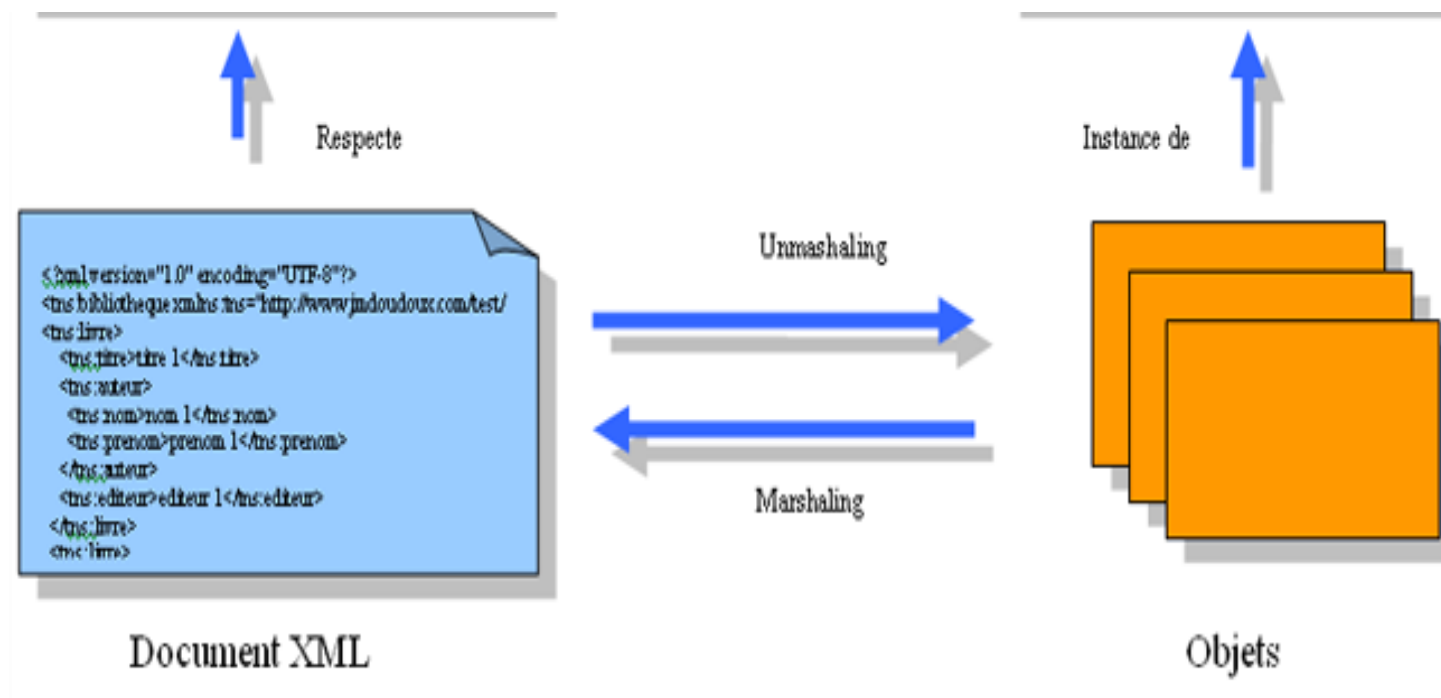
Messages - Communiquer - Exemple

SOAP - Protocole de communication



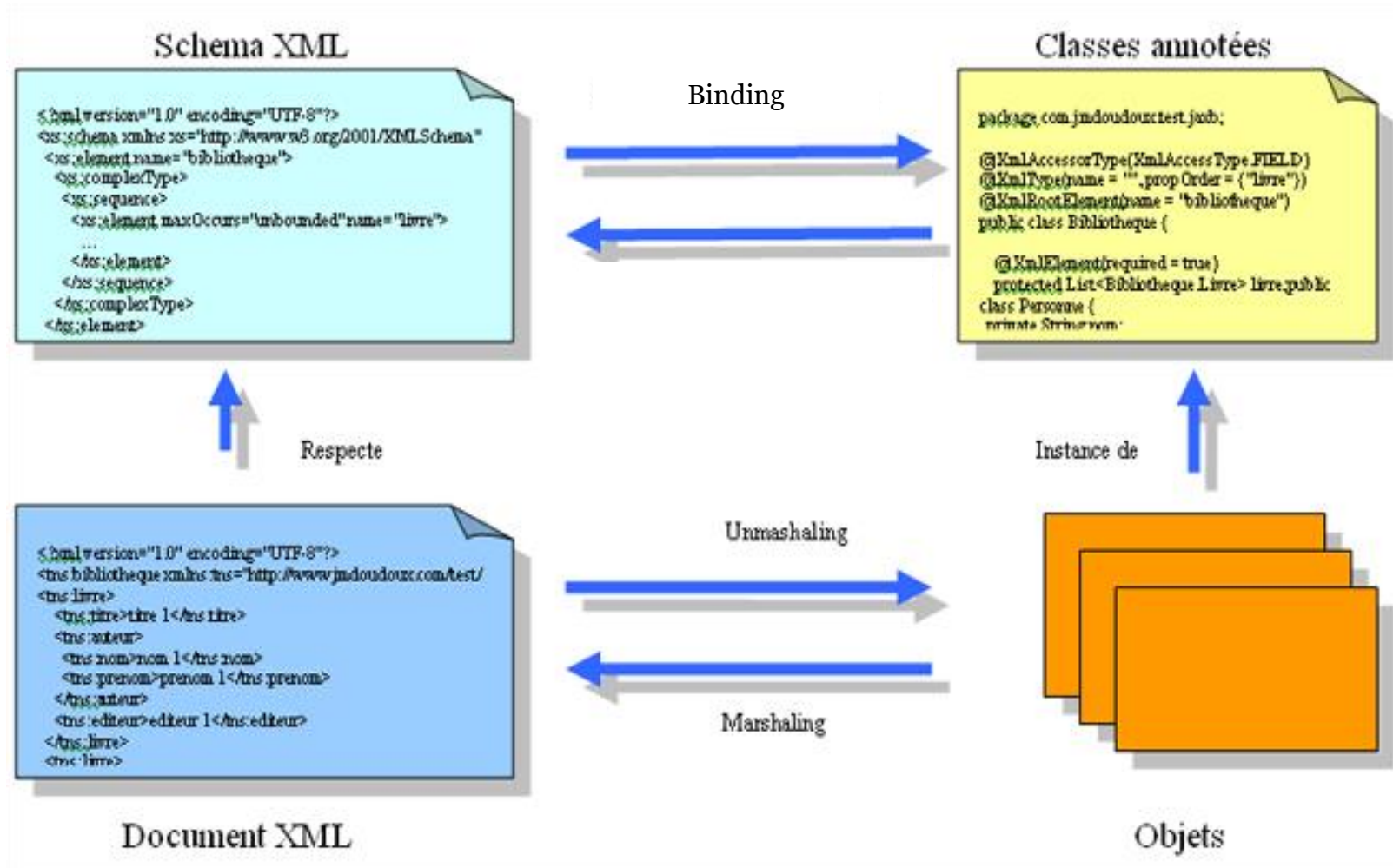
Messages - Communiquer

SOAP : Sérialisation/Désérialisation



Messages - Communiquer

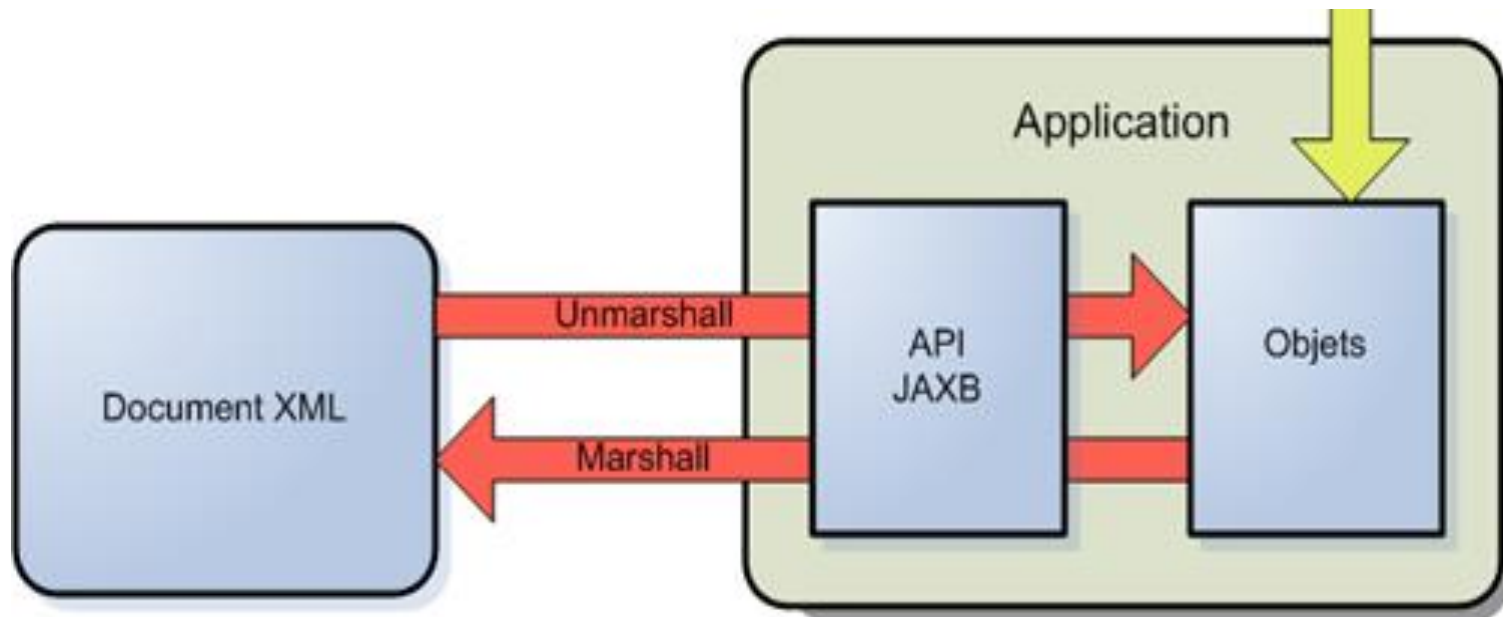
SOAP : Sérialisation/Désérialisation



Messages - Communiquer

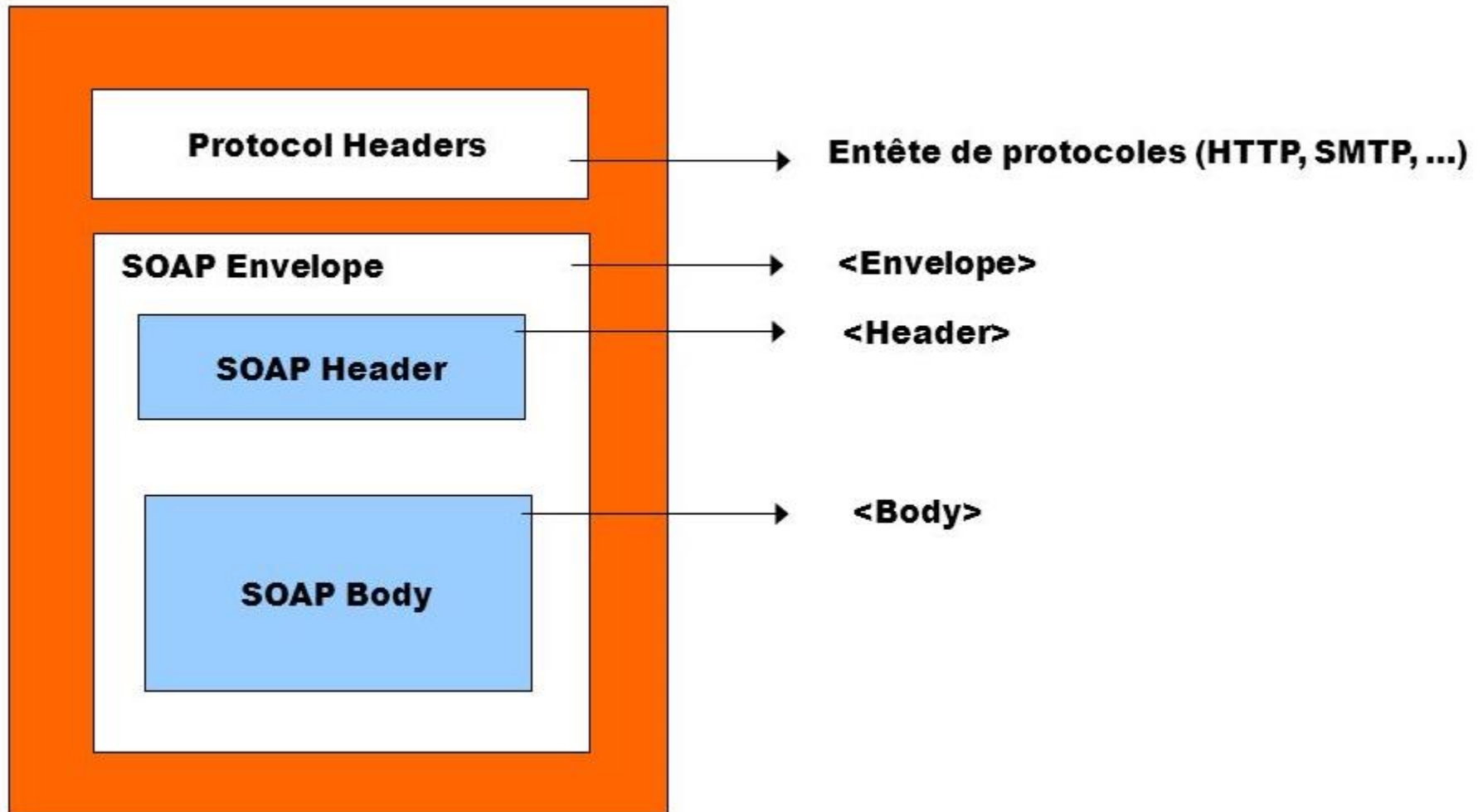
SOAP : Sérialisation/Désérialisation

- **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.



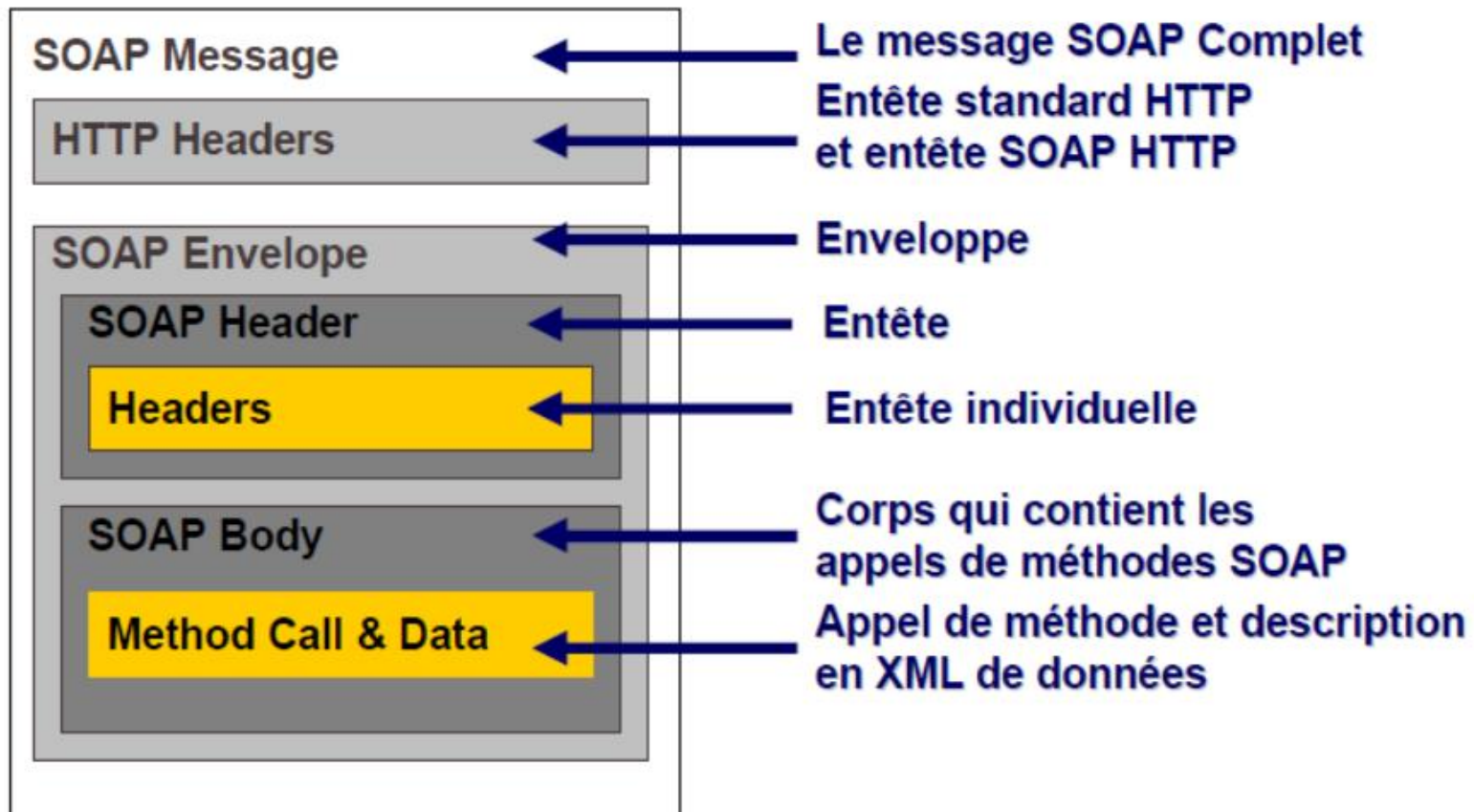
Messages - Communiquer

SOAP – Structure d'un message



Messages - Communiquer

SOAP – Structure d'un message transporté par HTTP



Messages - Communiquer

SOAP – Structure d'un message en XML

- 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 SOAP sert de conteneur aux autres éléments du message SOAP.
- L'enveloppe est la racine d'un message SOAP identifiée par la balise `<soapenv:Envelope> ... </soapenv:Envelope>`.
- La spécification impose que tous les attributs contenus dans l'enveloppe SOAP soient explicitement associées à un namespace (espace de nommage), de manière à supprimer toute ambiguïté/conflits.
- La spécification SOAP définit un namespace (xmlns):
 - ✓ **SOAP-ENV** ou **soapenv** ou **S** : <http://schemas.xmlsoap.org/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).

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> ... </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.

Messages - Communiquer

SOAP : Le corps

- Le corps SOAP est un élément **obligatoire** dans le message SOAP.
- 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>**)
- Le **corps** doit fournir le **nom de la méthode** invoquée par une requête ainsi que les **paramètres** associés à celle-ci.
- Appel d'une opération représentée par une structure *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 : Le corps

➤ 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 : Le corps

➤ 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>
```

SOAP : Le corps

- **Fault Block** en cas d'erreur: la balise `<soap:fault>` qui est contenue dans le corps SOAP.
- Cette balise est utilisée pour communiquer un **problème** qui a eu lieu dans la tentative de réalisation de la demande adressée au service Web.
- L'élément d'erreur est facultatif et figure uniquement dans les messages de **réponse**, il ne peut y apparaître qu'une seule fois.
- La balise `<soap:fault>` peut contenir quatre autres balises facultatives :
 - **faultcode** : cet élément est requis. Il contient un code indiquant la nature du problème.
 - **faultstring** : est la version lisible par l'homme de la balise faultcode. C'est la traduction en langage naturel du code d'erreur.

SOAP : Le corps

- **Fault Block** en cas d'erreur: la balise `<soap:fault>` qui est contenue dans le corps SOAP.
- Cette balise est utilisée pour communiquer un **problème** qui a eu lieu dans la tentative de réalisation de la demande adressée au service Web.
- L'élément d'erreur est facultatif et figure uniquement dans les messages de **réponse**, il ne peut y apparaître qu'une seule fois.
- La balise `<soap:fault>` peut contenir quatre autres balises facultatives :
 - **faultactor** : indique le service qui a généré l'erreur. Cela est important lorsqu'une chaîne de services a été utilisée pour traiter la demande.
 - **detail** : cet élément doit contenir autant d'informations que possible sur l'état du serveur à l'instant de l'apparition de l'erreur. Il contient souvent des valeurs de variables au moment de l'échec.

Messages - Communiquer

SOAP : Le corps

- 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

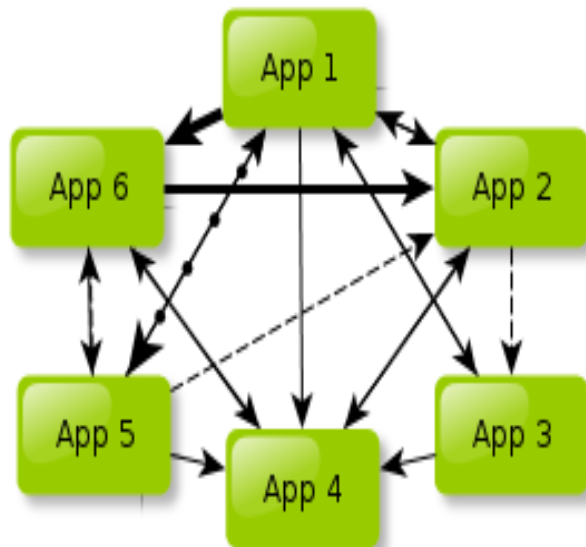
Messages - Communiquer

SOAP : Le corps

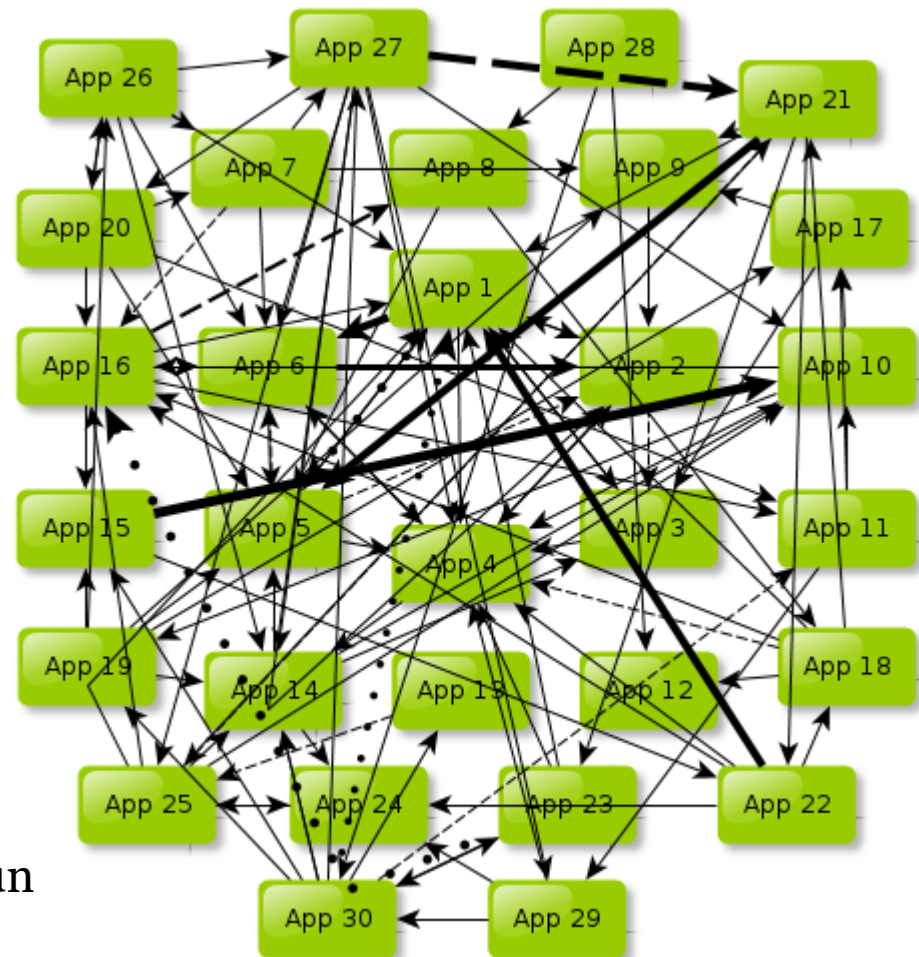
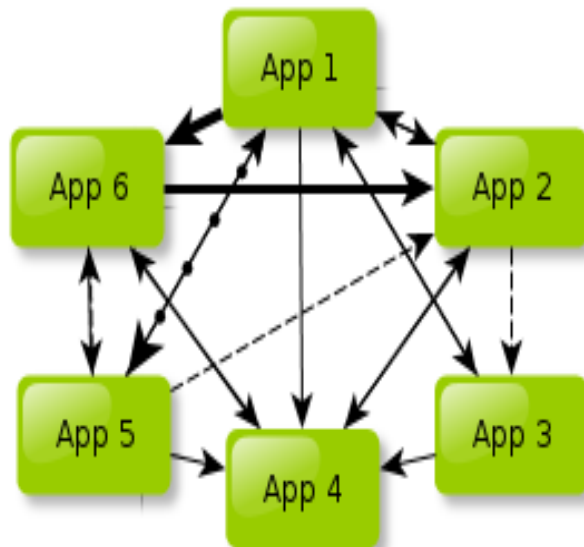
Exemple : corps d'un message SOAP pour le résultat d'une opération **addUser** qui n'est pas exposée par le service web - Fault

```
<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



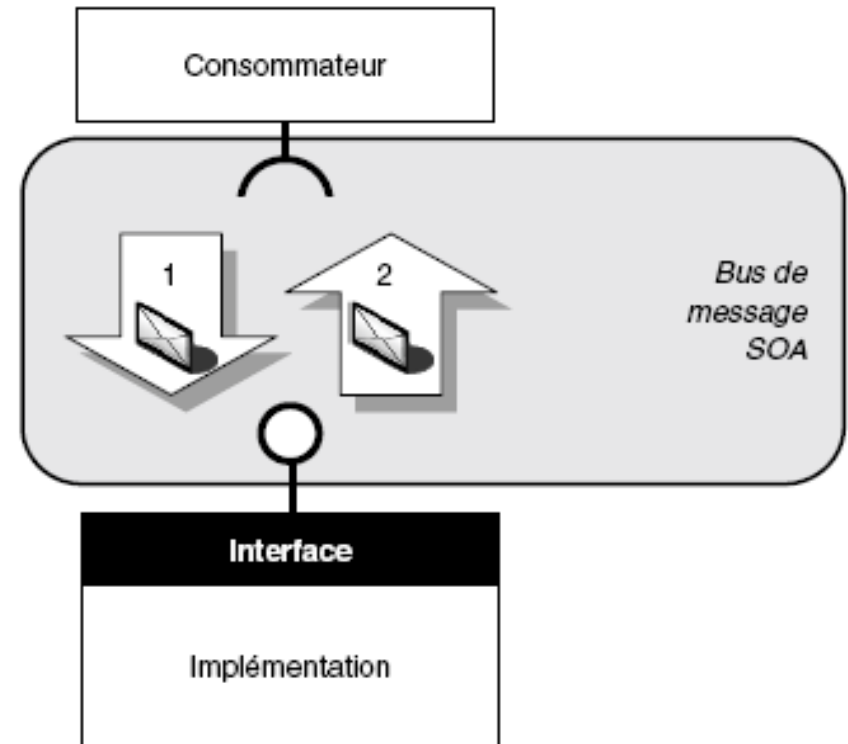
Messages - Communiquer



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

Bus de Message - ESB

- 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 de messages**.



Bus de messages - ESB

- Bus de messages **SOA** = ESB – **Enterprise Service Bus**.
- L'ESB est un **composant logiciel** chargé d'assurer les échanges d'informations entre services.
- L'ESB est une solution technologique, aussi nommé outil informatique middleware, qui facilite **l'interaction entre les applications et la circulation de l'information** dans un écosystème: dans le système d'information, mais également avec celui des fournisseurs, partenaires, client, etc.
- Ce bus applicatif agit comme une **colonne d'intégration** à travers laquelle les applications interagissent **sans contraintes de format ou de temporalité**.
- Il a un rôle de médiation et de communication entre les services, applications, bases de données, etc.
- **Solution d'intégration** privilégiée pour la mise en œuvre d'une architecture SOA, qui reprend les principes de l'EAI en se basant sur des standards : XML, SOAP, WSDL, WS-*, UDDI, BPEL, etc.

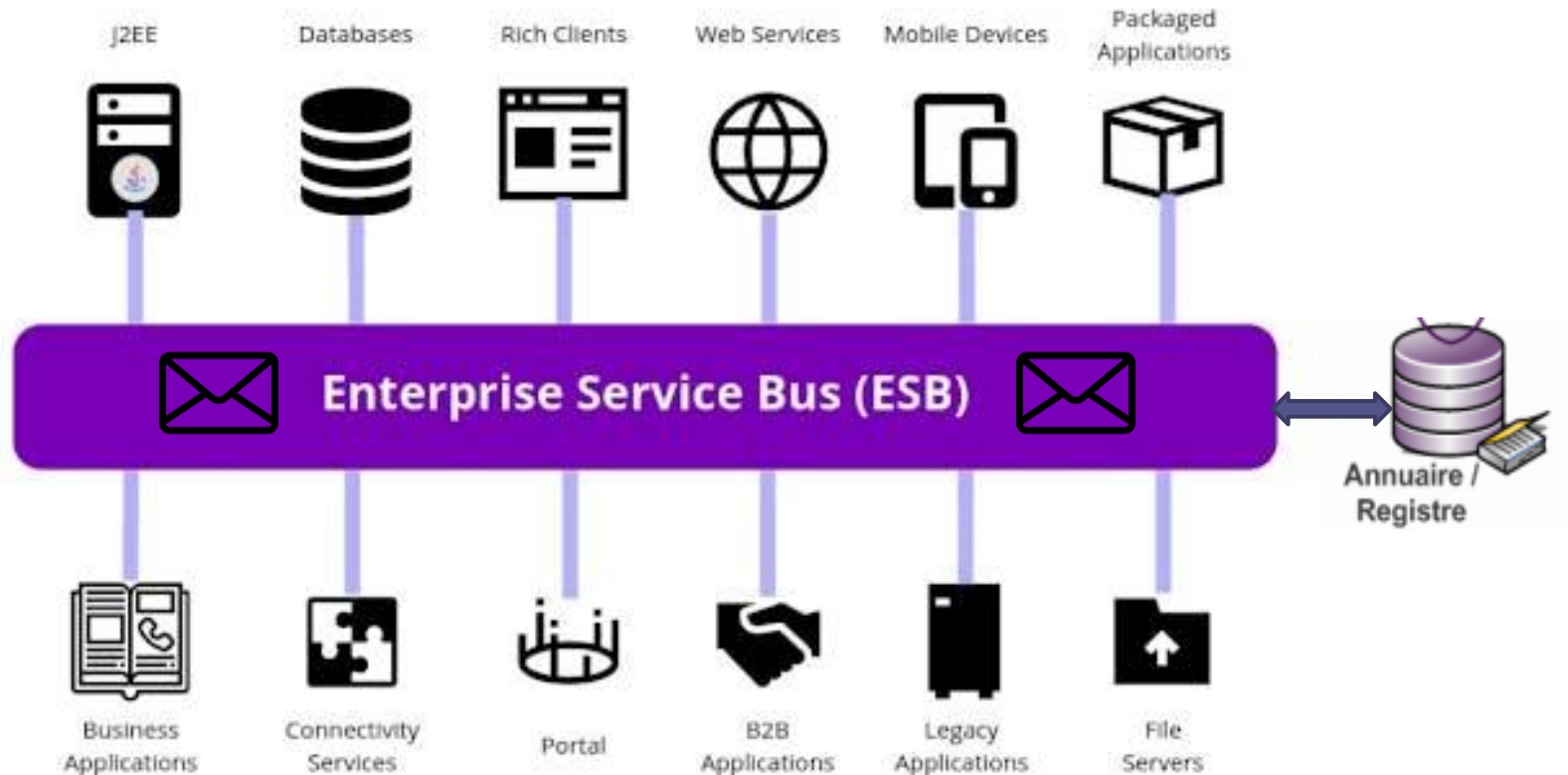
Bus de messages - ESB

- L'ESB permet à des applications/services qui ne parlent pas la même langue de se comprendre et communiquer via un **canal central**.
- Les ESB sont les héritiers directs des EAI : Bea, Tibco, Oracle, IBM, etc.




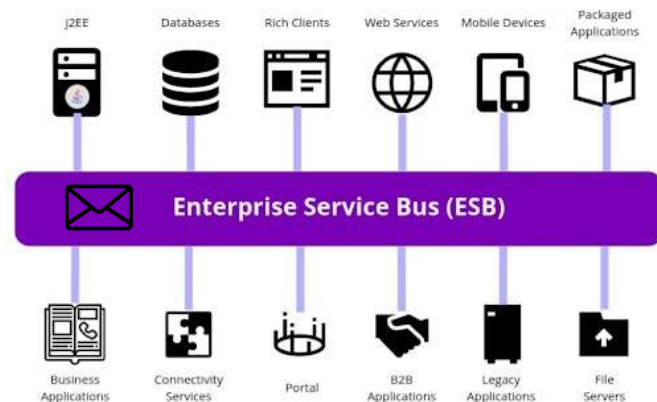
Bus de messages - ESB

- Les applications du SI vont proposer leurs **services** à l'ESB, qui va les publier dans son registre afin de les mettre à disposition de toutes les applications, qui peuvent s'y abonner.
- Les applications sont à la fois fournisseurs et consommatrices de services.



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

- ESB garanti **l'interopérabilité** entre des services hétérogènes.
- Et **l'agilité** du SI en limitant les contraintes techniques.
- Il permet de rendre les consommateurs des services aussi indépendants que possible (couplage faible):
 - 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 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 .

Bus de messages - ESB

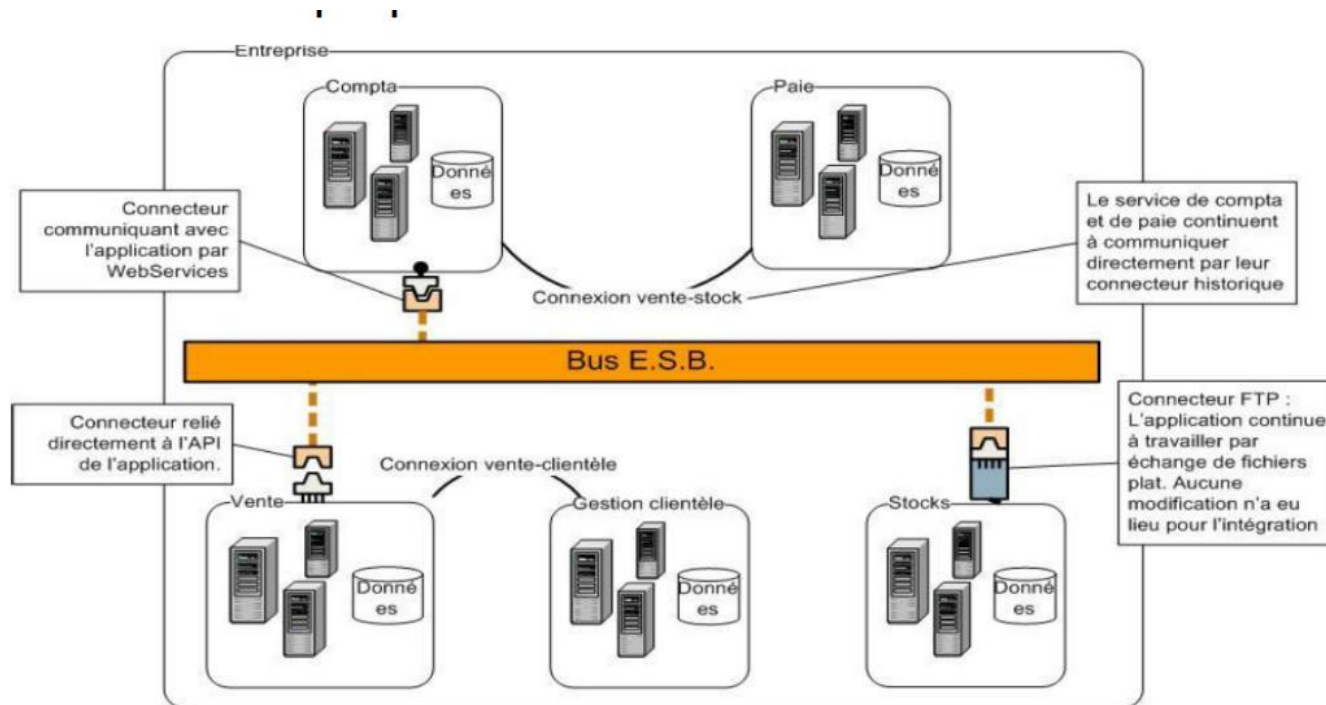
... Et la **sécurité**

- 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

➤ **Connectivité** de l'ESB:

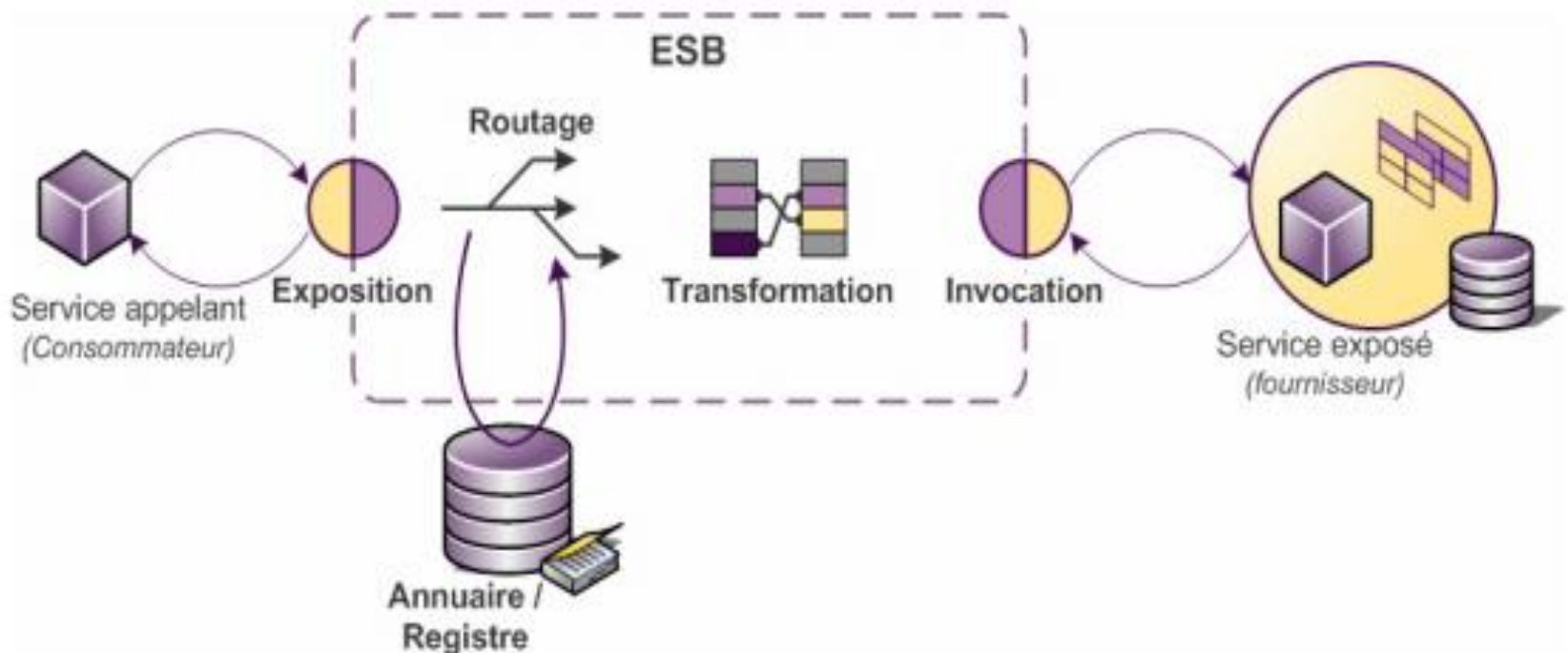
- L'ESB est connecté aux applications par des connecteurs d'adaptation de formats et spécificités.
 - Connecteurs **techniques** vers des formats d'échange : fichiers CSV, formats XML, RMI, RPC, etc.
 - Connecteurs **métiers** pour intégrer des progiciels.
 - Connecteurs **propriétaires**



Quelques cas d'utilisation d'un ESB

Couplage lâche

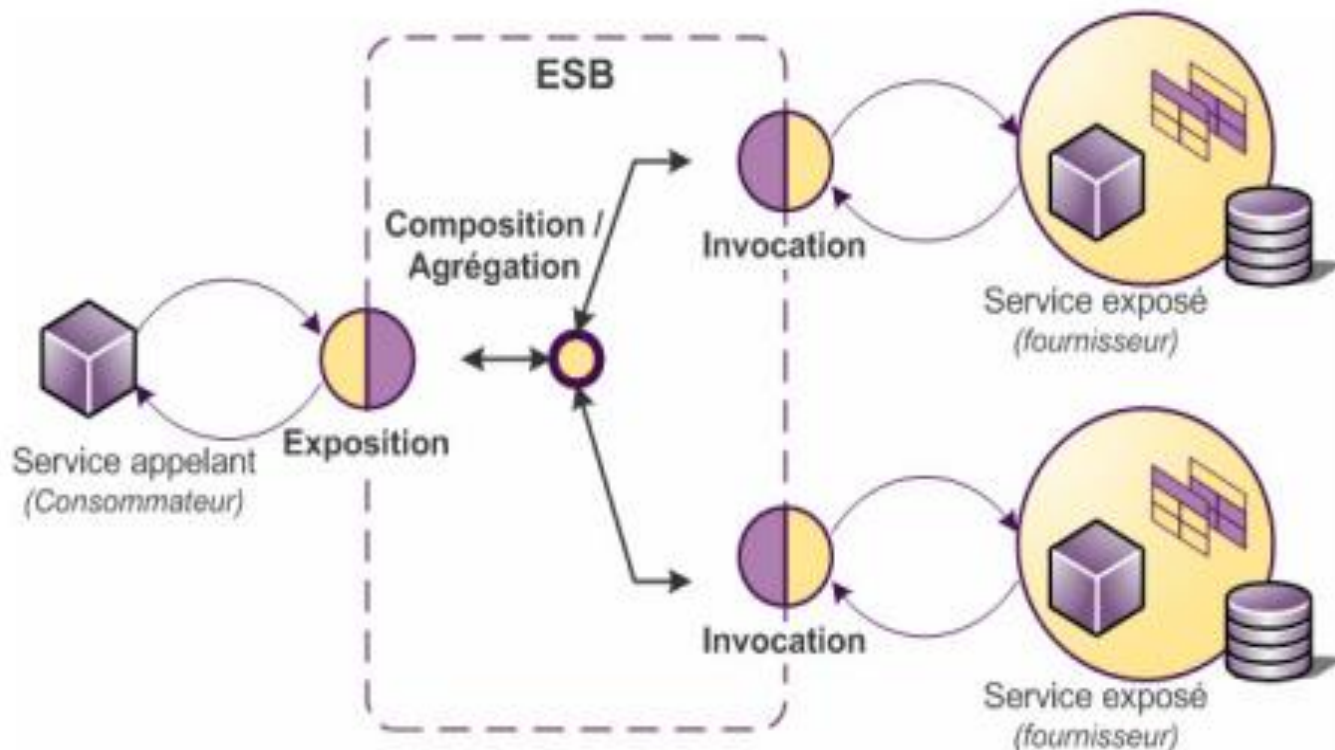
- Le service invoqué peut être exposé dans une autre technologie, utiliser une autre représentation des données, etc.
- L'ESB va prendre en charge ces **transformations** pour libérer le consommateur appelant des adhérences avec le service qu'il appelle.



Quelques cas d'utilisation d'un ESB

Composition et Orchestration de services

- L'ESB expose un service virtuel qu'il construit par **composition** de plusieurs autres services.
- **Uniformise** les traitements, les organise et centralise leur supervision.

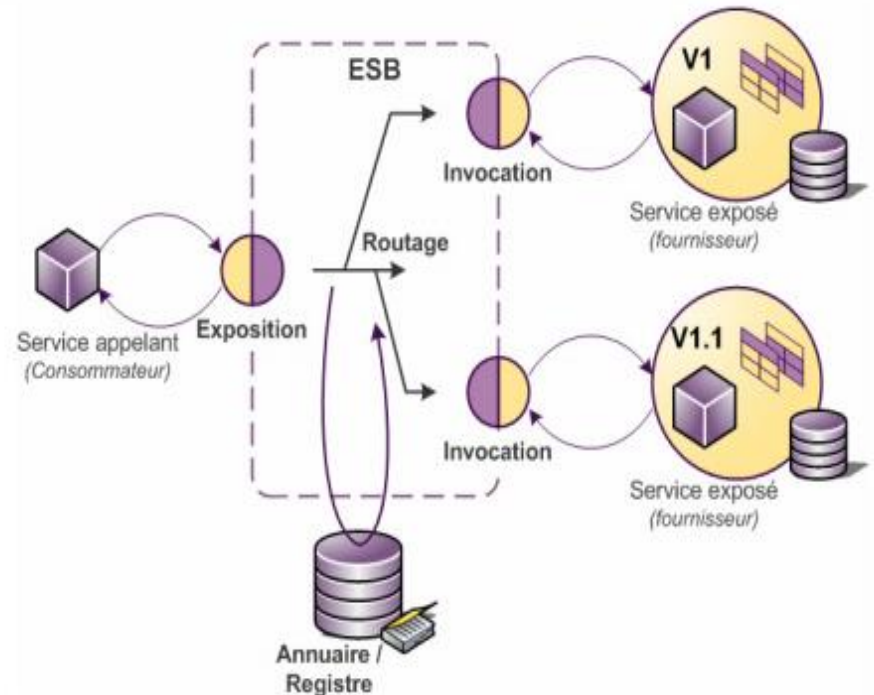


Quelques cas d'utilisation d'un ESB

Gestion de versions

Plusieurs situations peuvent se présenter :

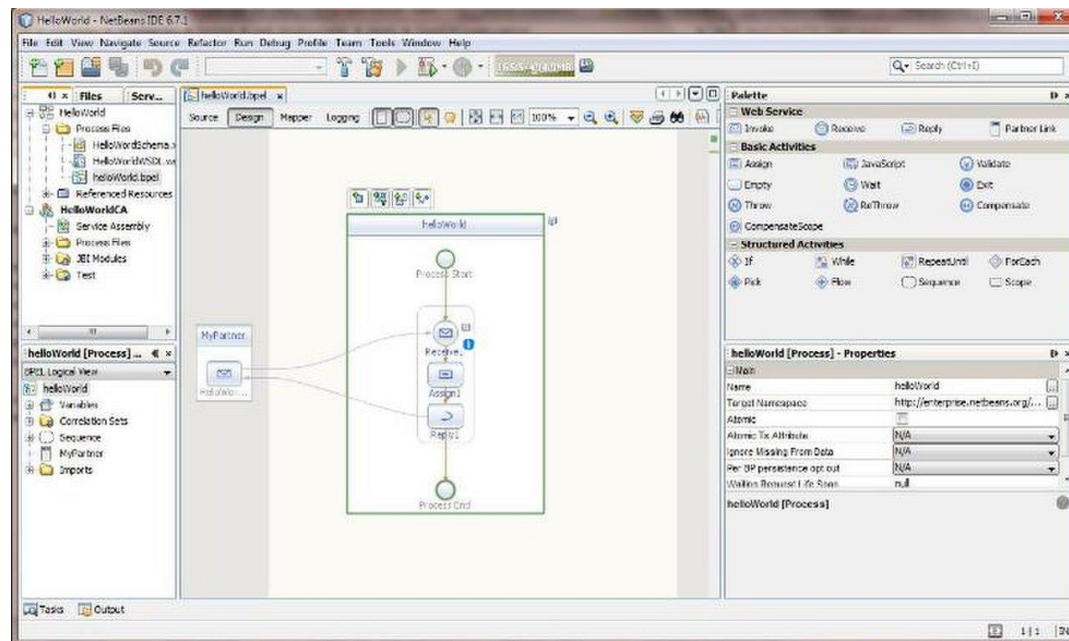
- Les versions sont incompatibles entre elles : le choix d'une version se fait par **routing**.
- La nouvelle version est une extension compatible avec la version précédente :
 - ✓ Ici L'ESB effectue l'appel vers la nouvelle version en appliquant une **transformation** du format d'entrée, du format de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



Bus de messages - ESB

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.



Plateforme SOA

Services

Services

Plateforme SOA

Services

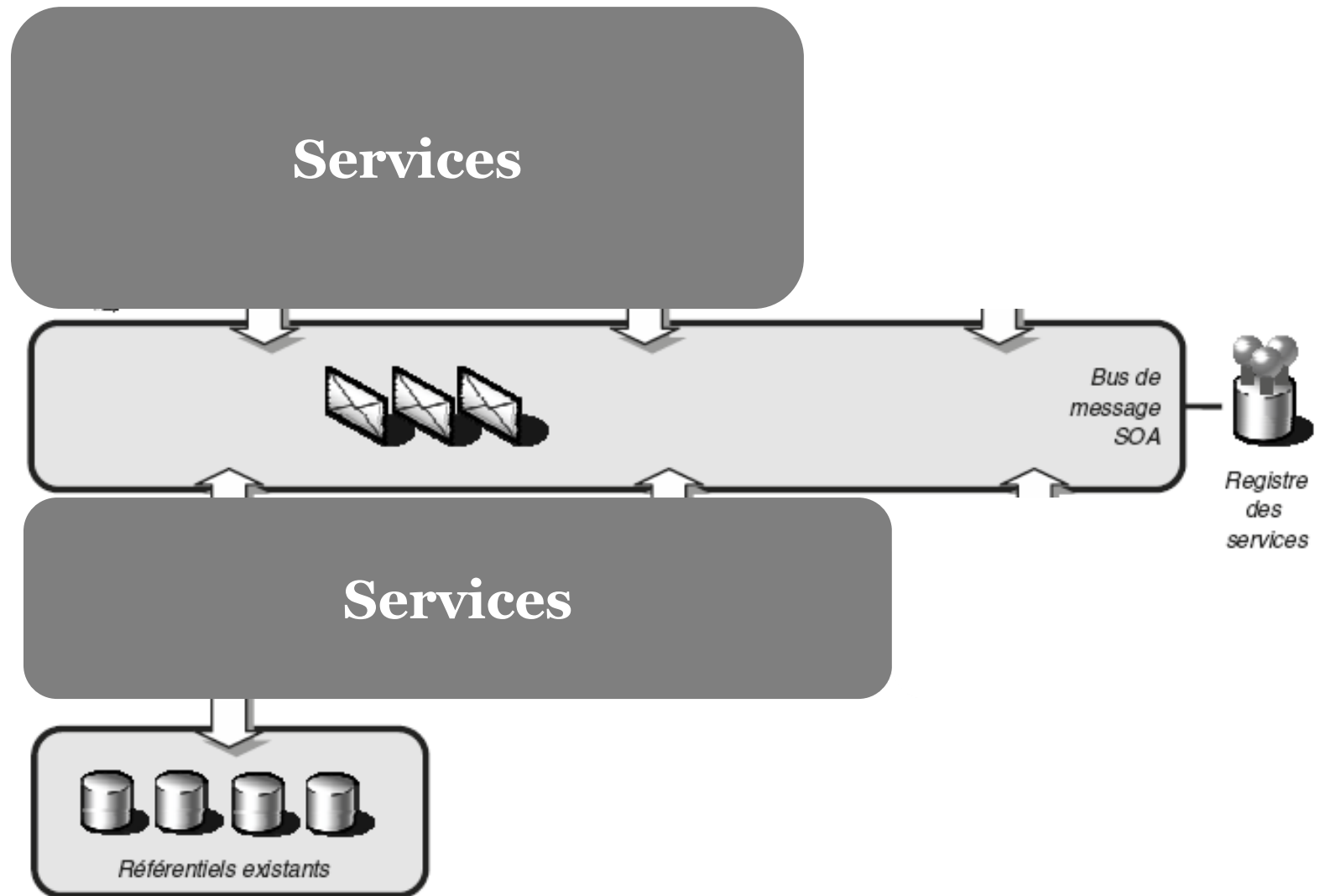


*Registre
des
services*

Services

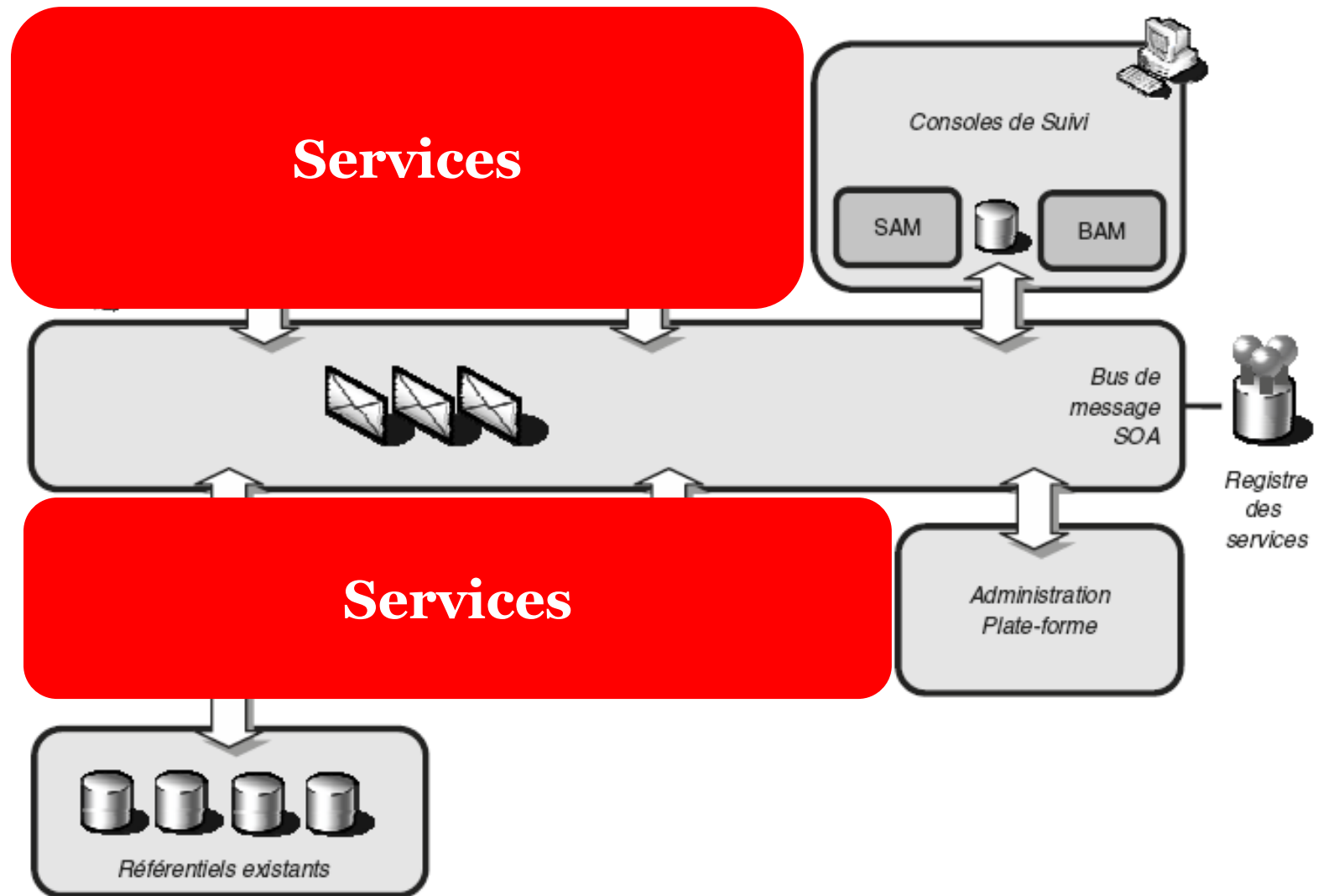


Plateforme SOA

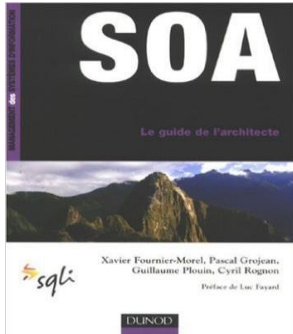


Plateforme SOA

Business Activity Monitoring
Service Activity Monitoring



Ressources



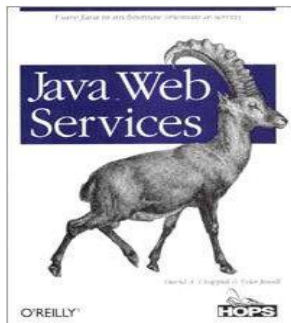
Le guide de l'architecte du SI

- ✓ 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

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

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

Blog – BlueWay – L'Entreprise Service Bus

- ✓ <https://www.blueway.fr/experience/enterprise-service-bus-esb>