

IBM Software Group

# Architectures Orientées Services : architecture applicative et impact infrastructure

Dominique Delhumeau  
Executive I/T Architect, IBM Software Group



# Agenda



## **1. Architecture Orientée Services**

**Constatations**

**Principes**

**Architecture applicative**

## **2. Architecture logique**

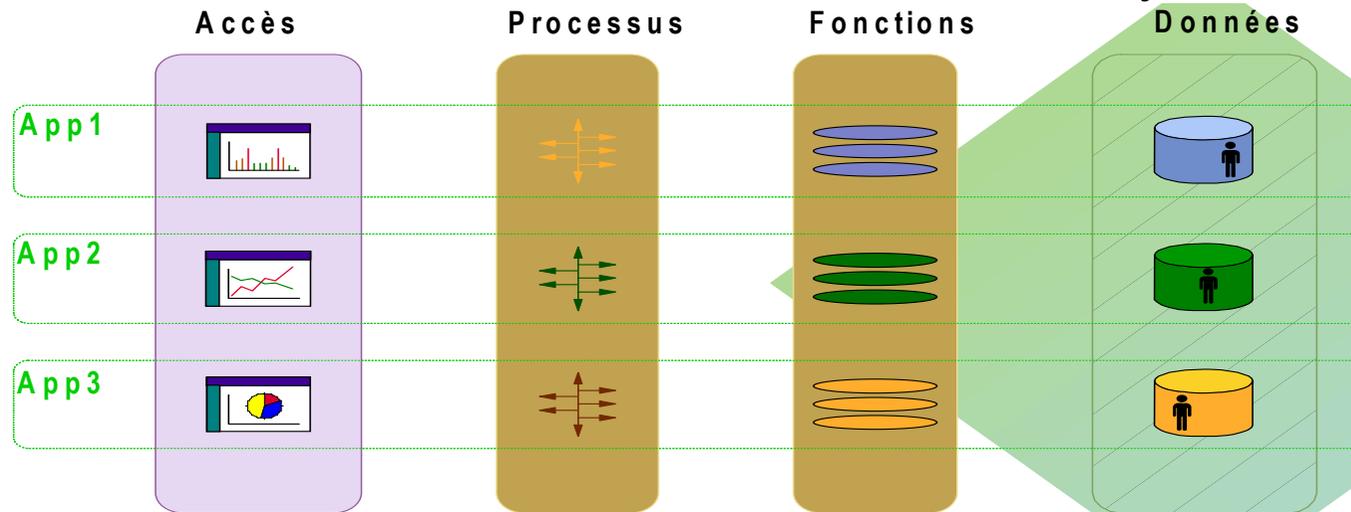
Protocoles et Standards

Enterprise Service Bus

Couches d'Architecture logique



1ère constatation: Les systèmes d'information sont encore largement organisés par silos produits. Les modèles Métier et Technologique nécessitent une mutualisation des services et des moyens



### Evolution depuis . . .

Des processus, fonctions, infrastructures spécifiques à chaque silo d'application ou canal de distribution

### Vers ....

**Une mutualisation des ressources** applicatives et techniques

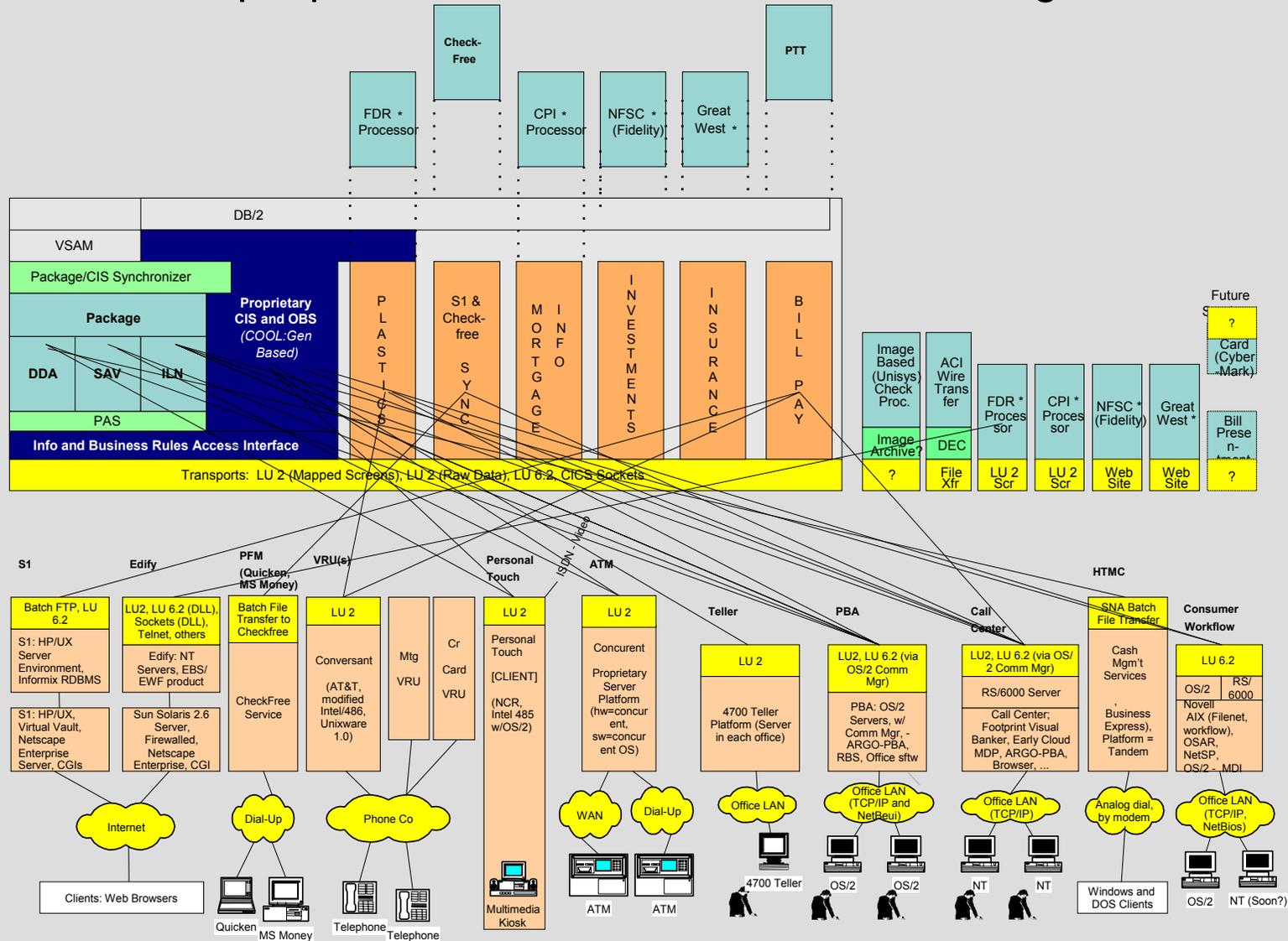
**Une ouverture des canaux de distribution** vers des partenaires

**Une extension des systèmes de production** à intégrer

**Une focalisation** sur les fonctions « Core Business »



# 2ème constatation: Les systèmes d'information nécessitent une urbanisation technique pour résoudre le cauchemar de l'intégration



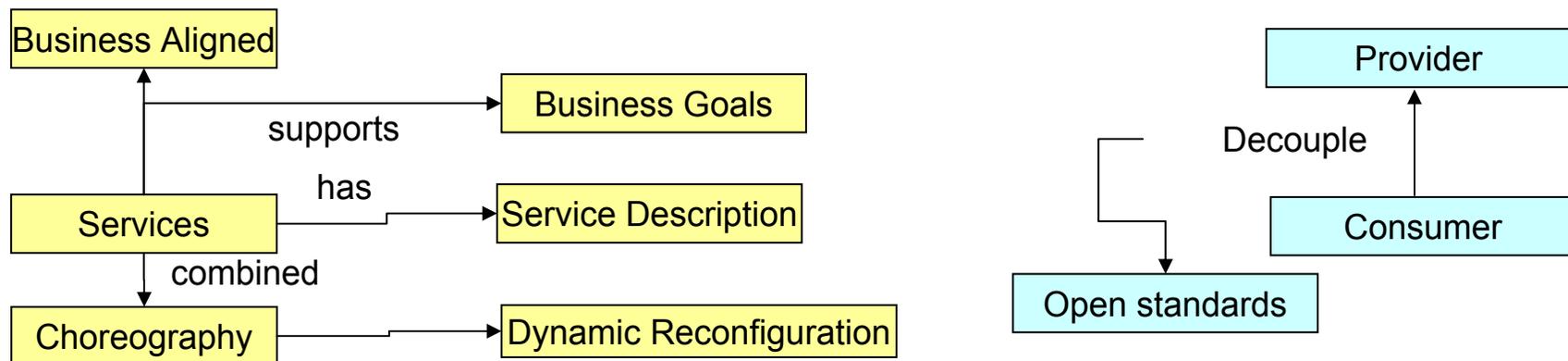
## Les enjeux des systèmes d'information d'aujourd'hui

- **Être flexible et réactif aux changements de l'organisation et du métier**
  - ▶ Évolution de la configuration fonctions - applications du système d'information
  - ▶ Évolution des partenariats
  - ▶ Évolution des processus métier
  
- **Réduire les coûts marginaux d'évolution logicielle et matérielle liés à l'évolution du métier**
  - ▶ Les évolutions technologiques ne doivent pas rendre l'existant obsolète
  - ▶ La valeur métier de l'existant doit être réutilisable dans les nouveaux projets
  - ▶ Les nouveaux composants applicatifs doivent
    - S'intégrer sans impact sur les composants des autres domaines fonctionnels
    - Réutiliser les fonctions existantes
    - Apporter des fonctions pérennes qui pourront être réutilisées à leur tour



SOA permet la synergie entre Métier et IT, en découplant les conditions et stratégies d'utilisation de l'implémentation concrète des fonctions

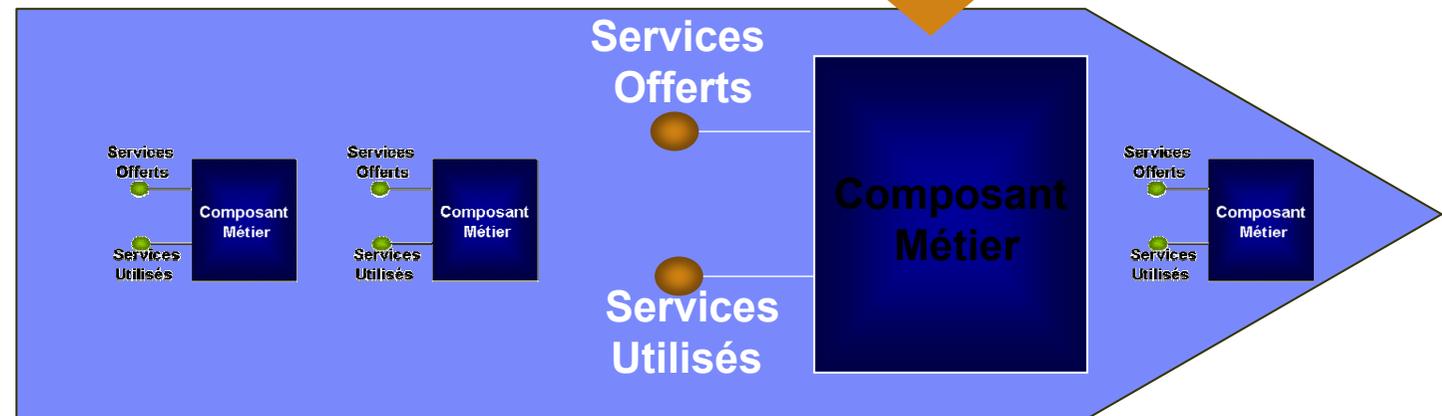
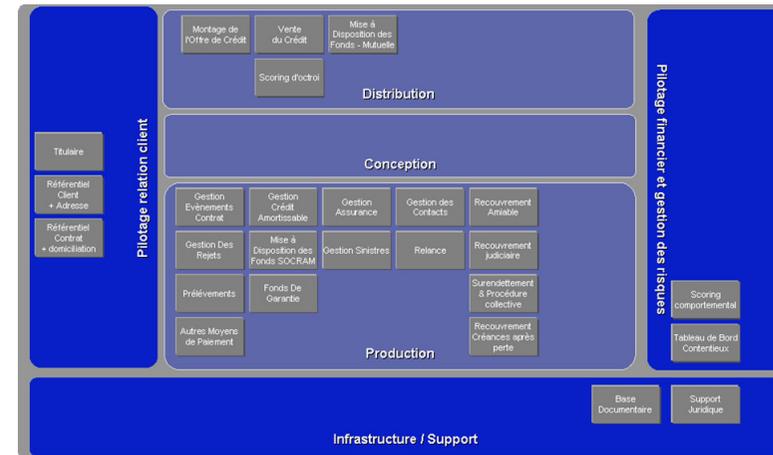
- Un **ensemble de principes d'architecture et de modèles (patterns)** tels que *modularité, encapsulation, couplage lâche, composition, chorégraphie*
- Un **style d'architecture** dans lequel un fournisseur de service offre une description de son service, et le consommateur trouve (statique ou dynamique) et met en œuvre (binding) l'implémentation à travers un référentiel
- Un **modèle de programmation** s'appuyant sur des standards, des méthodes, des outils et des technologies (ex.web services)
- Un **lien entre Services Métier et Alignement IT** qui supporte une capacité de réactivité



## 3 éléments « Métier » distincts : Composant, Service, Processus

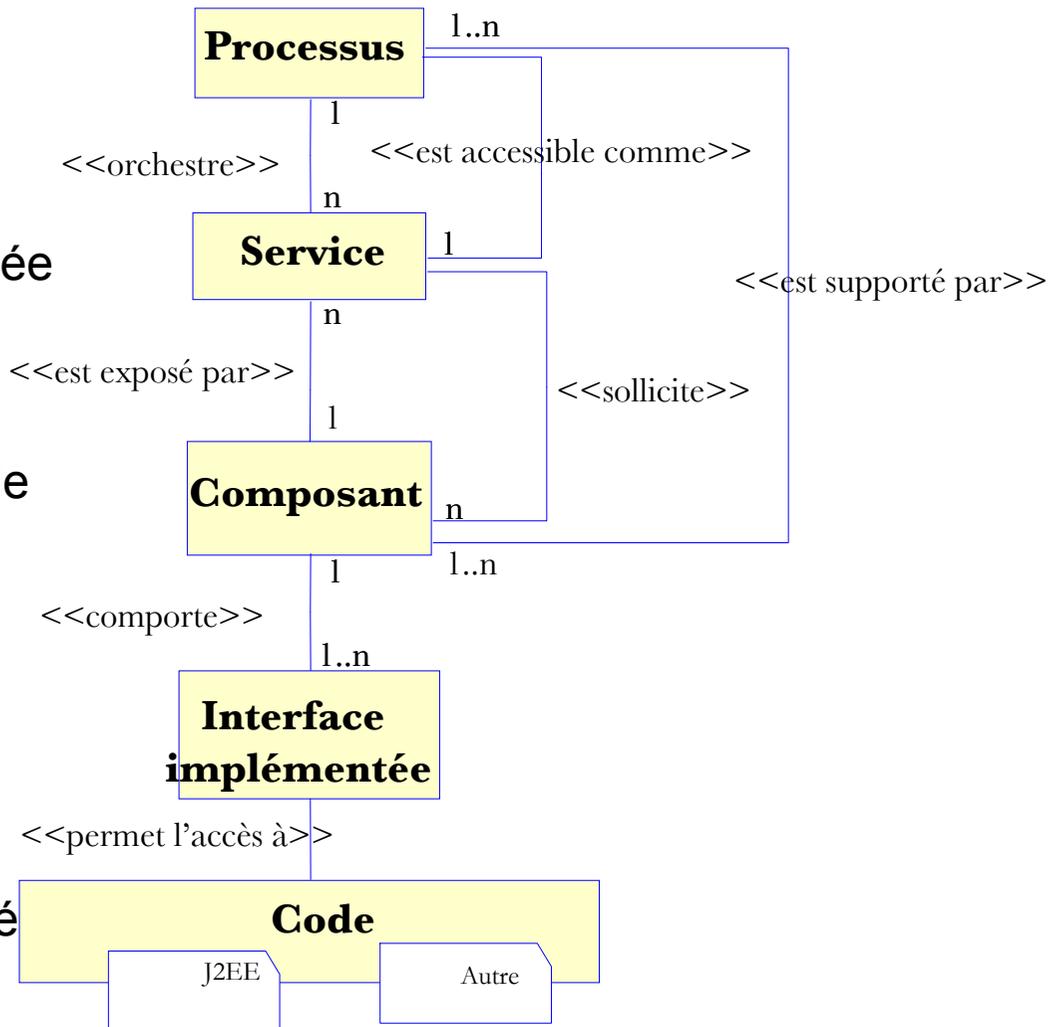
Du métier sont identifiés les services, et les services s'appuient sur la technologie pour réaliser la fonction

- Les services métier représentent des fonctions qu'un **composant métier** offre ou consomme à partir d'autres composants métier (internes ou externes)
- Un **processus métier** peut être défini comme une chaîne de composants métiers reliés par les services métier

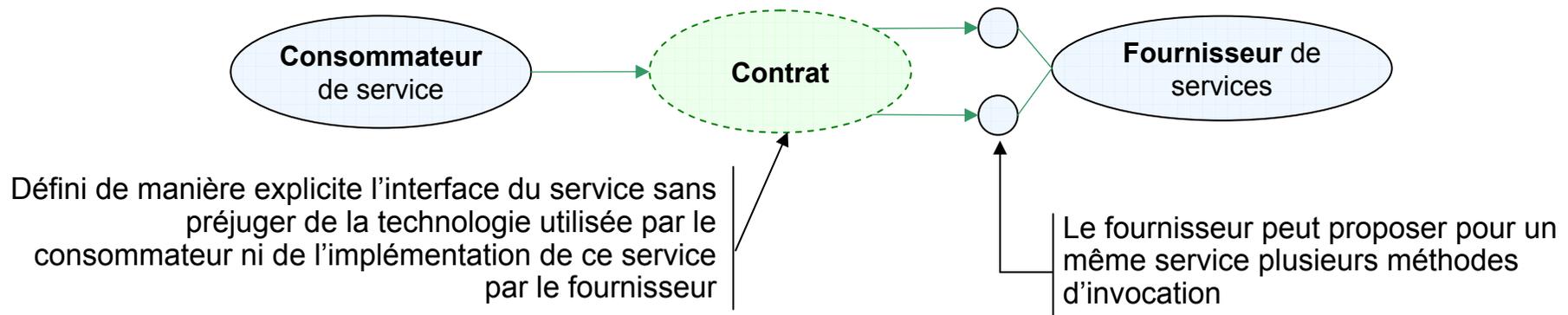


# Les concepts fondamentaux manipulés dans les Architectures Orientées Services

- Valeur Métier produite
- Activité métier automatisée
- Regroupement logique de fonctions homogènes
- Comment on accède physiquement au traitement
- Le traitement implémenté



## Une architecture orientée service (SOA) repose sur la définition et la publication de services



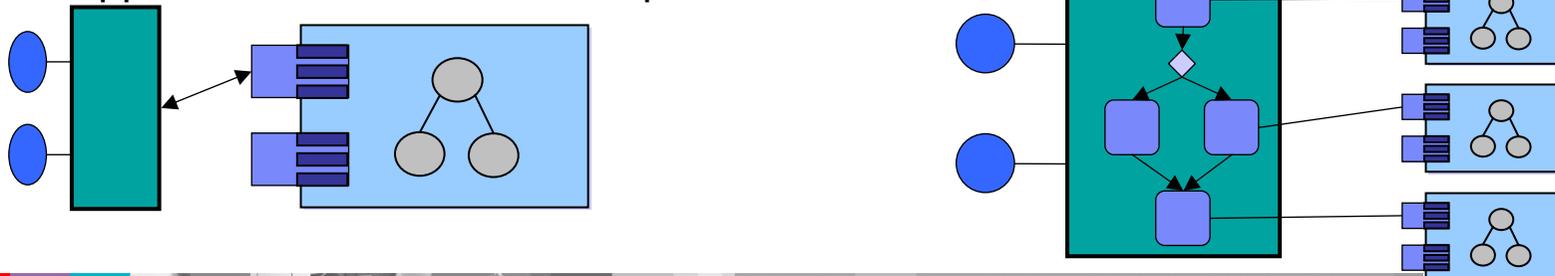
### ▪ Les services publiés doivent répondre aux principes de base suivants :

- **Neutralité** vis-à-vis du protocole utilisé par le consommateur
- **Couplage faible** entre le consommateur du service et le fournisseur
- **Granularité importante** : le service doit pouvoir être utilisé par le consommateur sans que ce dernier n'ait à connaître le modèle de donnée du fournisseur ou les détails de l'implémentation du service

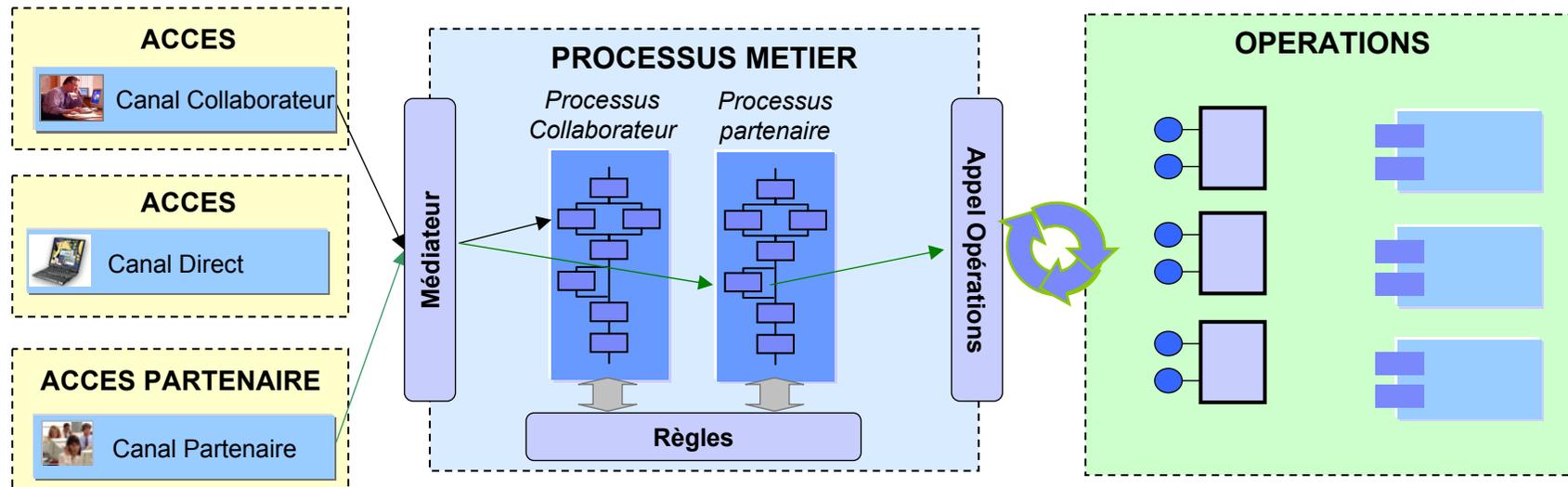


## Granularité des Services

- **Les Services doivent être si possible « à gros grain » afin de représenter des fonctions autonomes complètes, échangeables, réutilisables**
  - ▶ Le fait d'exposer en Web service une fonction technique, n'est pas en général optimum.
  - ▶ Les fonctions unitaires peuvent être composer en services atomiques à maille plus large.
  - ▶ Les services doivent avoir un couplage lâche. Le traitement dans un service ne doit pas être dépendant de l'état d'un autre service.
- **L'identification des services résulte souvent d'une approche « Top Down » en partant de l'analyse des processus**
  - ▶ Commencer avec les besoins métier et les définitions de cas d'utilisation
  - ▶ Représenter les services identifiés avec des classes abstraites
  - ▶ Mapper les classes avec des implémentations concrètes



L'architecture applicative doit pouvoir s'adapter aux différentes stratégies de distribution/ production: Intégrateur, Agrégateur, Fournisseur



### Accès

- Responsable des interactions utilisateurs humains et informatiques
- Isole les processus métiers de la complexité liée aux différentes interfaces

⇒ Ouverture aux canaux de distribution

### Processus métier

- Responsable de l'exécution des processus métiers et de la liaison entre le niveau accès et le niveau opérations
- Externalise les différentes règles de gestion liées aux processus des applications

⇒ Ouverture aux offres des partenaires

### Opérations

- Présente aux processus les différents services disponibles susceptible de répondre à leurs besoins sans préjuger de la localisation (physique et organisationnelle) de ce service

⇒ Réutilisation



# Agenda

## 1. Architecture Orientée Services

Constatations

Principes

Architecture applicative



## 2. Architecture logique

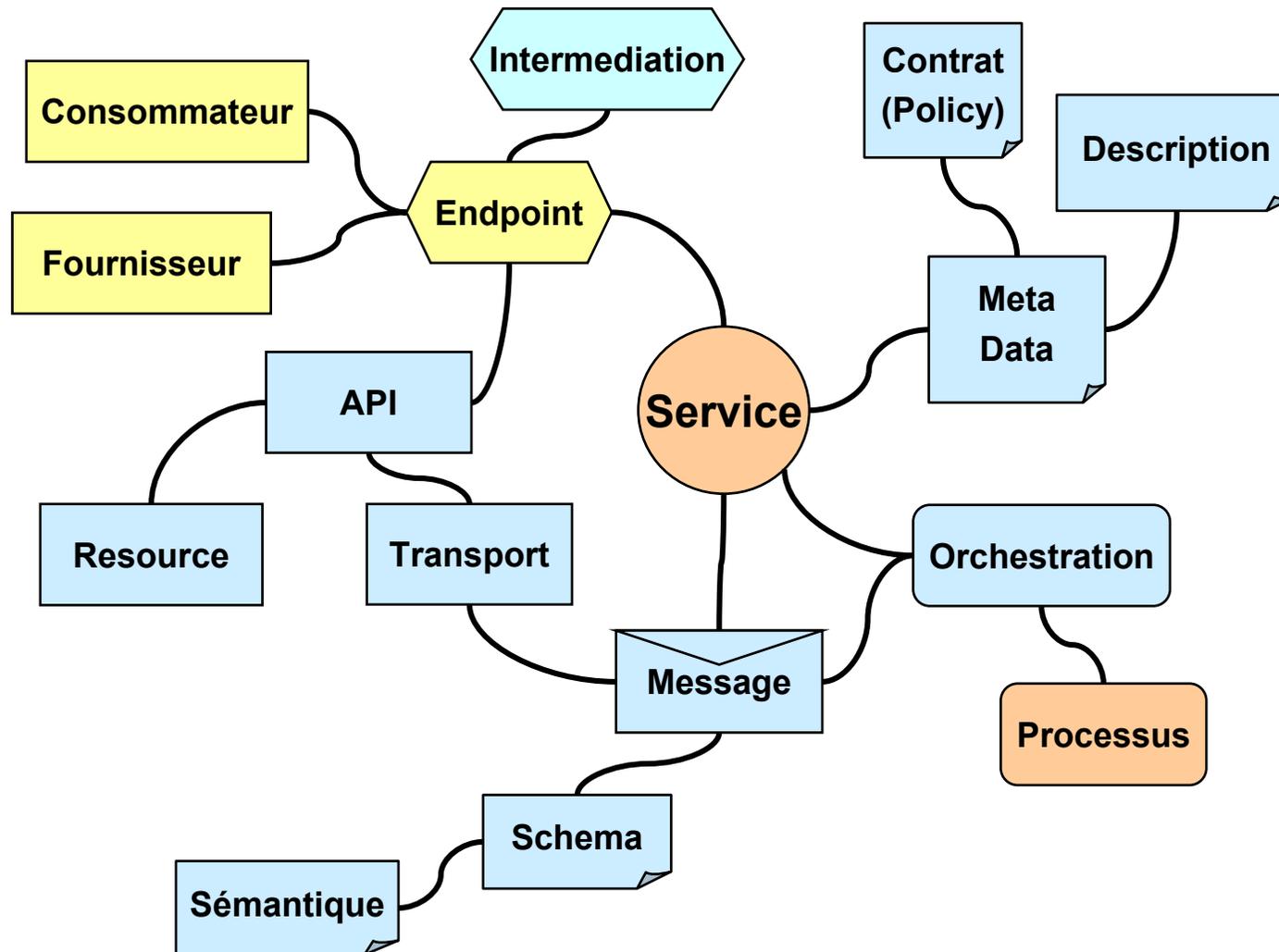
**Protocoles et Standards**

**Enterprise Service Bus**

**Couches d'Architecture logique**

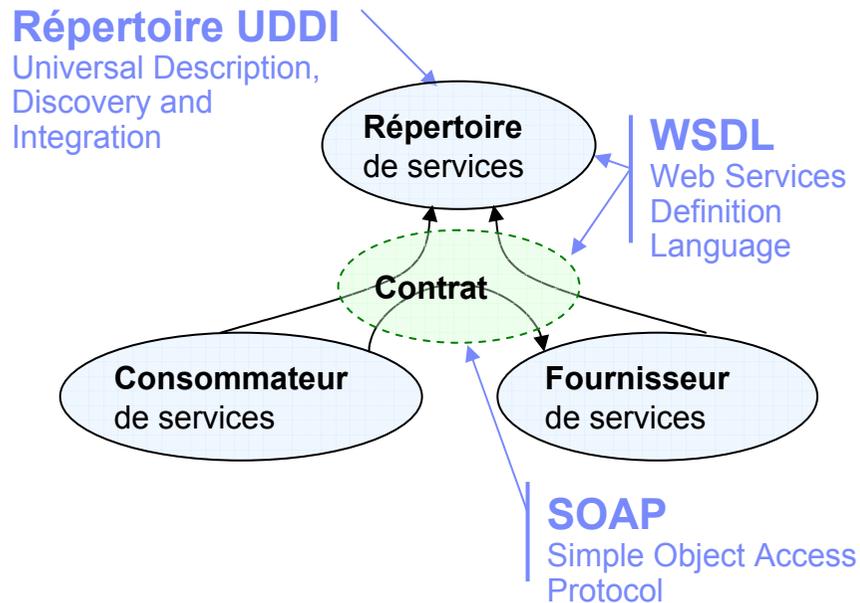


# Les différents éléments techniques de SOA



## Normes et standards : Web Services

### *Les principales technologies utilisées*



### SOAP

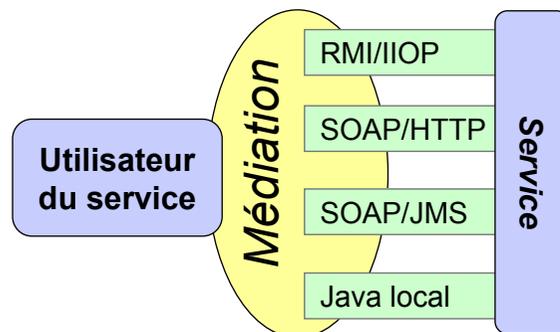
- Format de communication utilisant XML
- Défini l'entête technique et le corps du message
- Utilise un protocole de transport (souvent HTTP)

### WSDL

- Description de l'interface du service, de sa localisation, des différents protocoles de communication proposés

### UDDI

- Répertoire contenant les WSDL des services publiés.
- Il existe deux catégories de UDDI :
  - Les répertoires « publiques » accessibles par tous les consommateurs
  - Les répertoires « privés » internes à l'entreprise
- Évolution « métier » (Ontologie)



- Le fournisseur d'un service peut proposer plusieurs méthodes d'invocation
- en fonction de qualité de service
  - en fonction des besoins d'interopérabilité
  - en fonction des protocoles utilisables

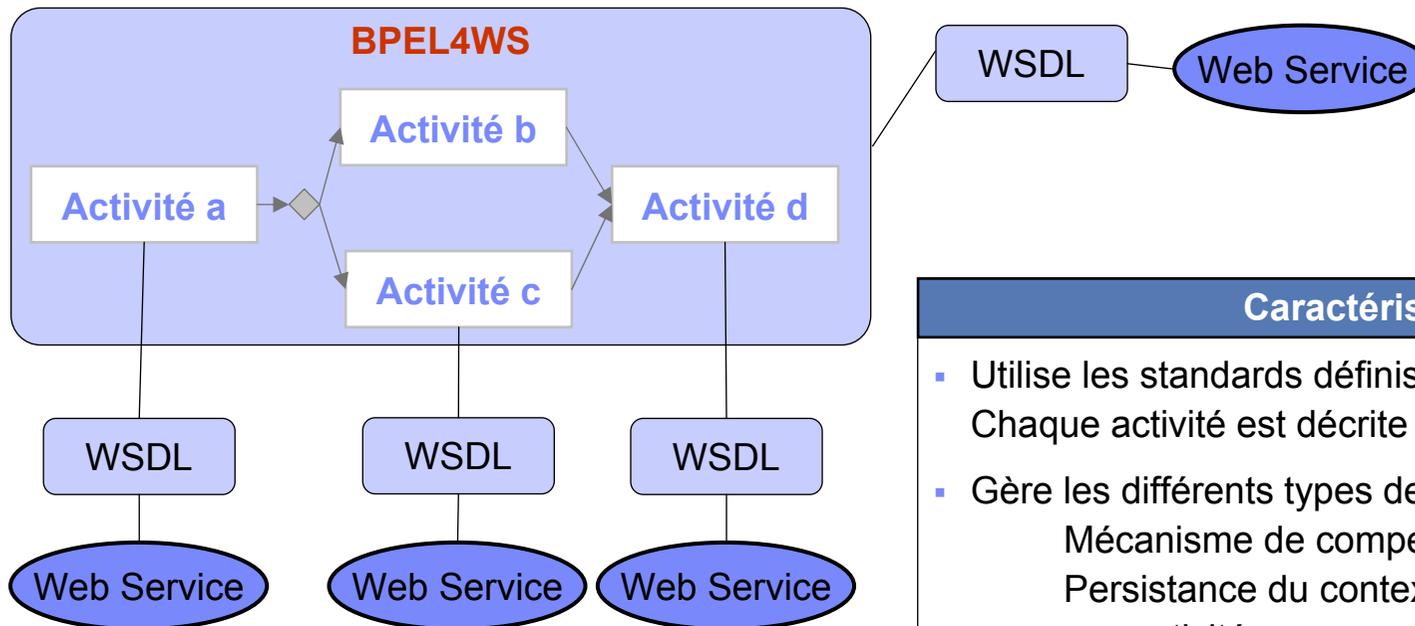
Les Web Services s'appuient sur un ensemble de "Stacks" normalisés  
... ou en cours de normalisation



## Normes et standards : les Web Services

### *BPEL4WS : Business Process Execution Language for Web Services*

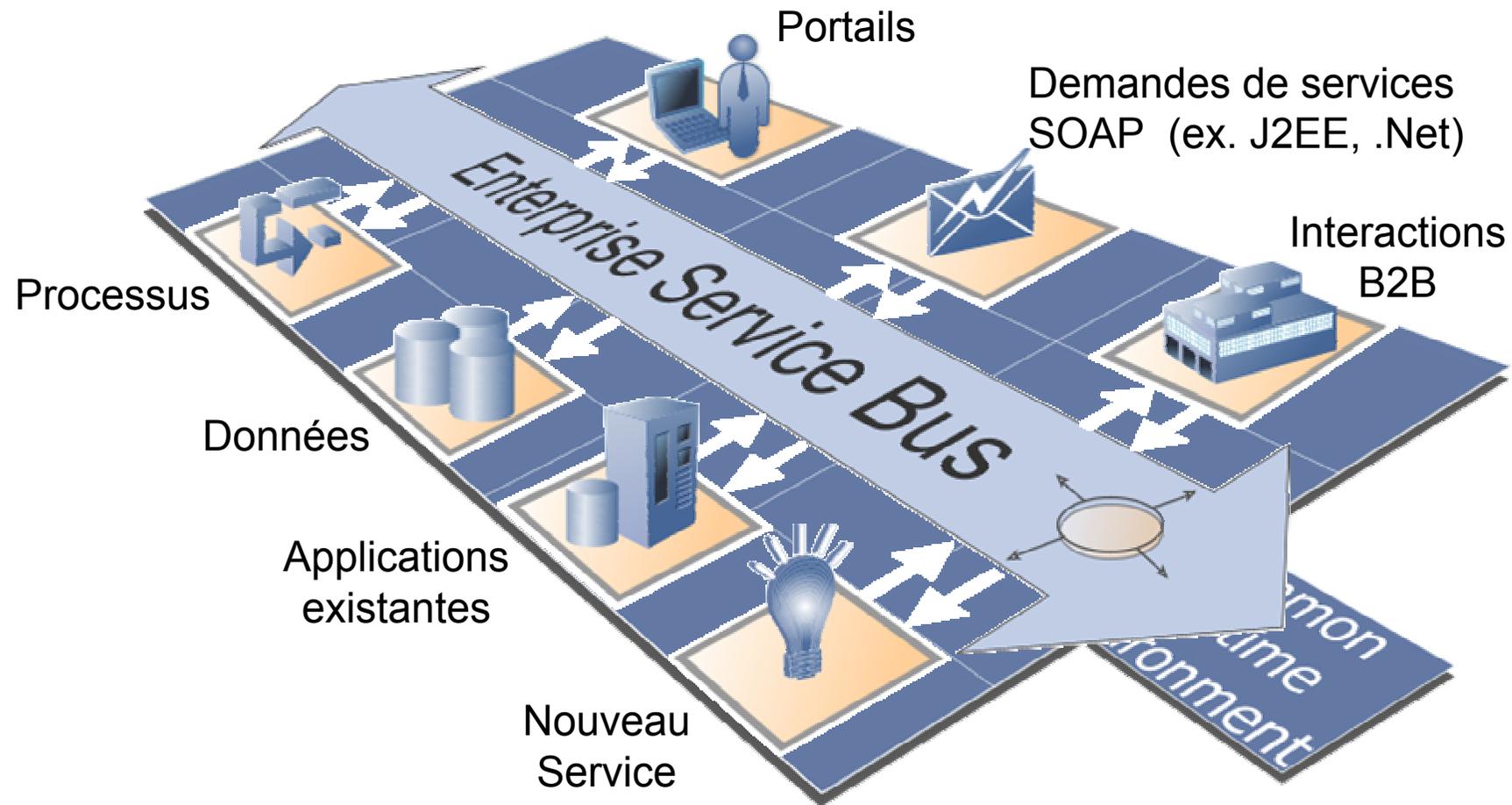
- **BPEL4WS** est un standard partagé par les principaux acteurs du marché (IBM, BEA, Microsoft) permettant de définir et d'exécuter un processus enchaînant des activités
- **Le processus est lui-même accessible en tant que Web Service**



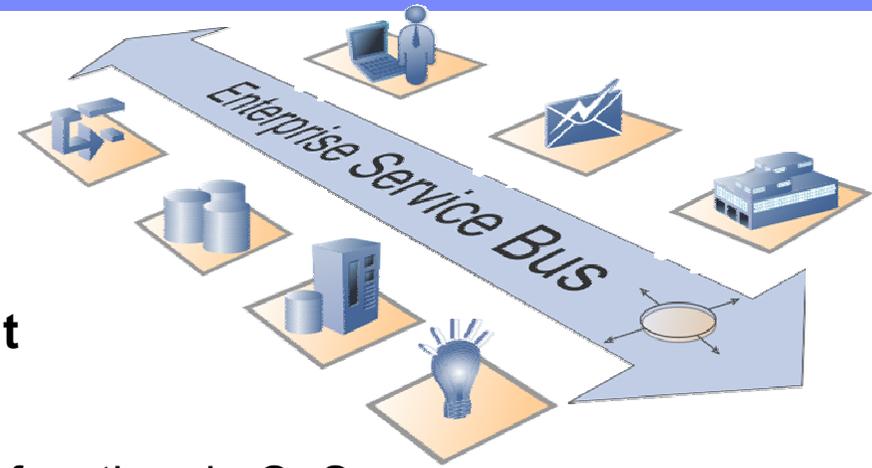
#### Caractéristiques

- Utilise les standards définis pour les Web Services :  
Chaque activité est décrite par un WSDL
- Gère les différents types de sphères d'activités :  
Mécanisme de compensation,  
Persistance du contexte du processus entre ses activités  
Contexte transactionnel

L'ESB est une couche d'infrastructure qui optimise les échanges entre les consommateurs de service et les fournisseurs de services



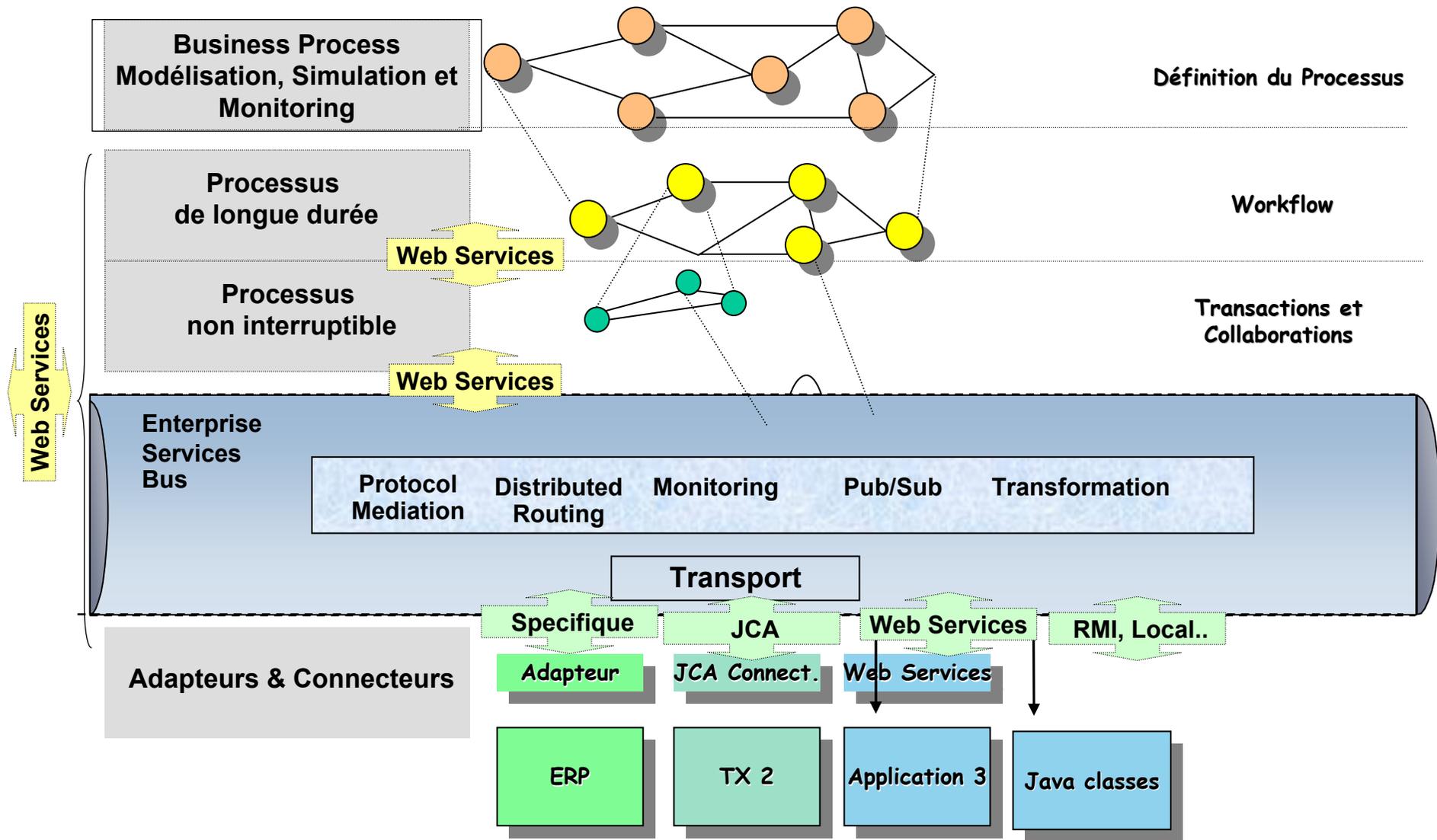
## Qu'est-ce qu'un ESB ?



- **Un modèle d'architecture – Pas un produit**
  - ▶ Décrit un “Pattern” d'interopérabilité
  - ▶ Les services interagissent entre eux en fonction de QoS
- **Connecte et Intègre les services de l'entreprise**
  - ▶ Plates-formes et Environnements hétérogènes
    - Différents sites, différentes méthodes de transport, différentes plates-formes
  - ▶ Fournit un « Bus »
    - pour des requêtes SOA, du Messaging et de la Notification (Événements)
- **Assure les fonctions de médiation entre Consommateurs et Fournisseurs**
  - ▶ Services de routage, de transformation, etc.
  - ▶ Transparence vis-à-vis du type de connectivité



# L'ESB sépare l'aspect "Métier" de l'implémentation et du middleware



Les fonctions d'un ESB dépendent de l'entreprise et résultent des besoins Métier et I/T.

<b>Communications, ex.</b> <ul style="list-style-type: none"> <li>■ Routage, Adressage, Protocoles</li> <li>■ pub/sub, asynchronisme, notification</li> </ul>	<b>Appel de Service, ex.</b> <ul style="list-style-type: none"> <li>■ Définition de l'Interface, Référentiels</li> <li>■ Substitution de service</li> </ul>
<b>Intégration, ex.</b> <ul style="list-style-type: none"> <li>■ DB, Transactionnels existants, ERPs</li> <li>■ Agrégation de services, Transformation de protocoles.</li> </ul>	<b>Qualité de Service, ex.</b> <ul style="list-style-type: none"> <li>■ Transactionnel</li> <li>■ Persistence</li> </ul>
<b>Sécurité, ex.</b> <ul style="list-style-type: none"> <li>■ Authentification, Autorisation, Non-répudiation, confidentialité</li> <li>■ Support des standards (WS-Security, Kerberos)</li> </ul>	<b>Niveau de Service, ex.</b> <ul style="list-style-type: none"> <li>■ Performance, Débit,</li> <li>■ Disponibilité, Scalabilité</li> </ul>
<b>Traitement des Messages, ex.</b> <ul style="list-style-type: none"> <li>■ Transformation des données, Transformation des messages</li> <li>■ Analyse de contenu, Intermédiations, etc.</li> </ul>	<b>Gestion, ex.</b> <ul style="list-style-type: none"> <li>■ Audit, Log, Métrique, Monitoring, versioning.</li> <li>■ Service provisioning, Facturation</li> </ul>
<b>Modelisation, ex.</b> <ul style="list-style-type: none"> <li>■ GBO, formats de l'industrie, processus de l'industrie, etc.</li> </ul>	<b>Intelligence Métier, ex.</b> <ul style="list-style-type: none"> <li>■ Règles métier</li> <li>■ Comportement par "policies"</li> </ul>



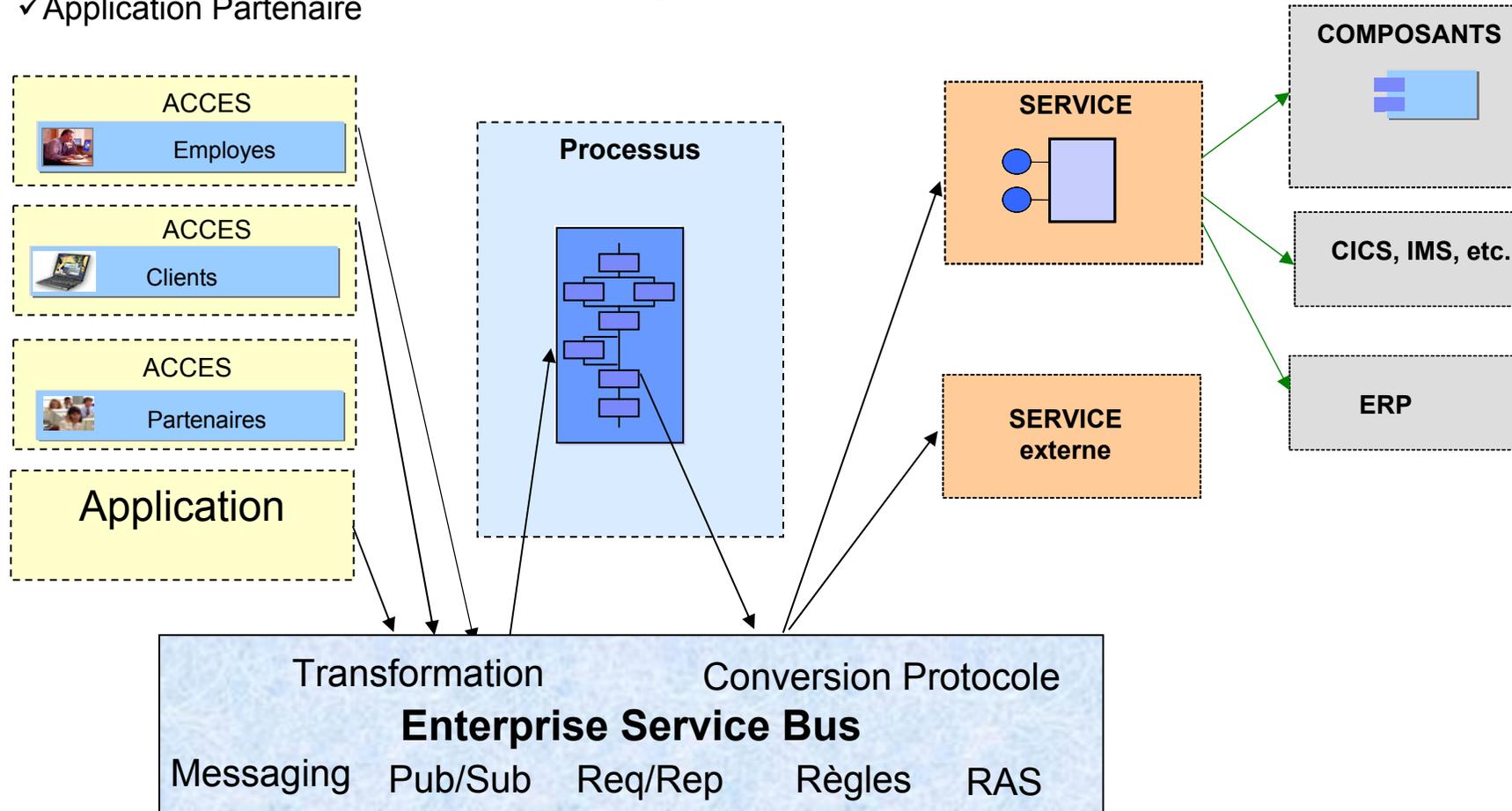
# En résumé, SOA s'implémente dans différentes couches de l'architecture et ces couches interagissent à travers l'ESB

- ✓ Client Léger ou Riche
- ✓ Portail
- ✓ Application Partenaire

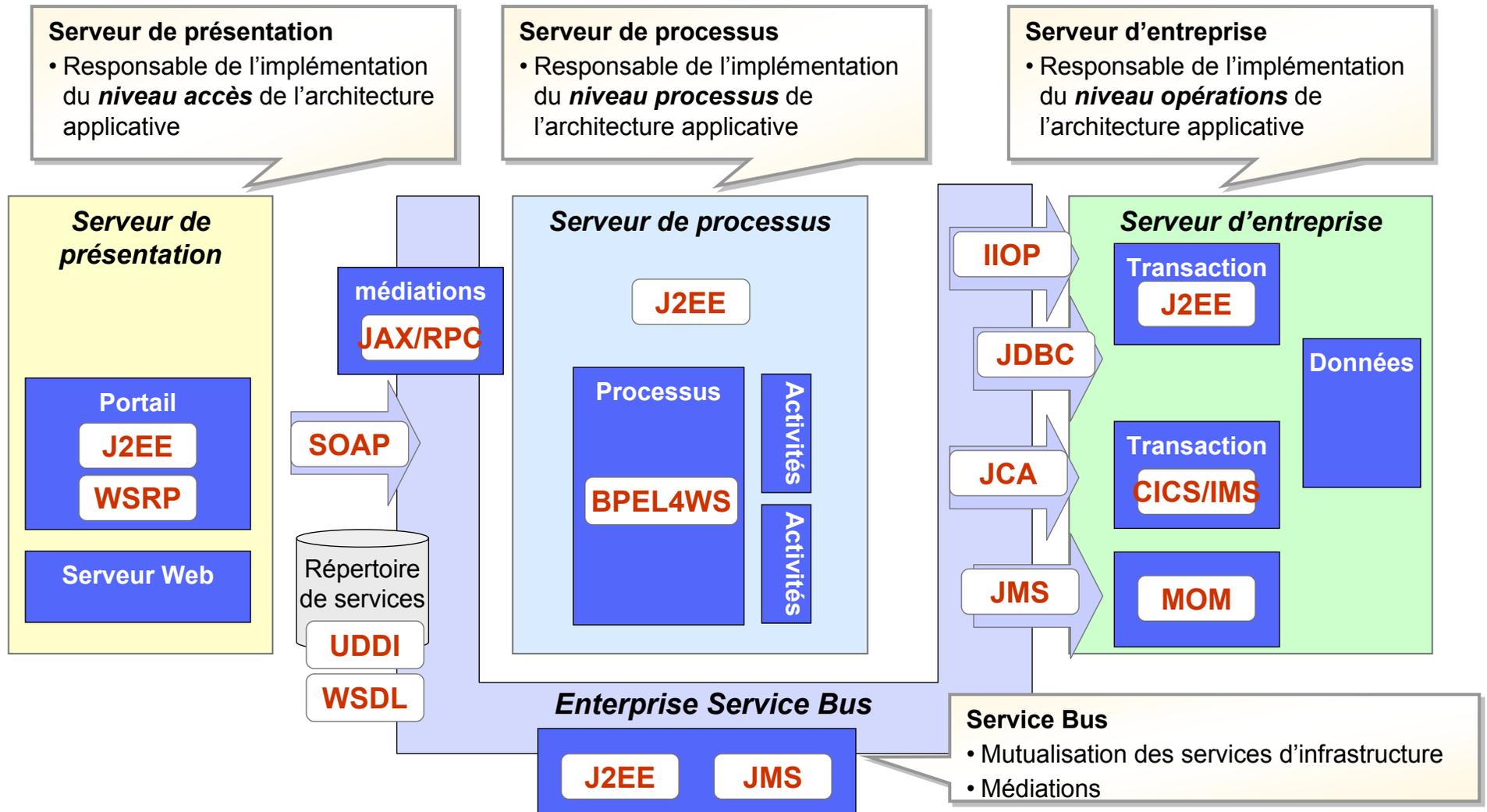
- ✓ Moteur de Workflow
- ✓ Moteur de règles

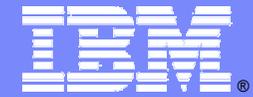
- ✓ Serveur d'intégration
- ✓ Adapteurs

- ✓ Applications



# Cette architecture logique s'appuie sur des standards





IBM Software Group

Merci pour votre attention.  
Des questions ?



© 2005 IBM Corporation