Architectures orientées services (SOA)
Architecture applicative et cartographie

ECOLE NATIONALE POLYTECHNIQUE D'ORAN – MAURICE AUDIN
Département Mathématiques et Informatique
Filière IMSI : 4ème année ingénieur

M. SABRI

Sommaire

① Applications d'entreprise
- Etat des lieux des SI
- Modèle de référence

② Patrons d'architecture pour les applications
- Architecture en couches
- Modèle n-tiers
- Composants
- Infrastructures logicielles

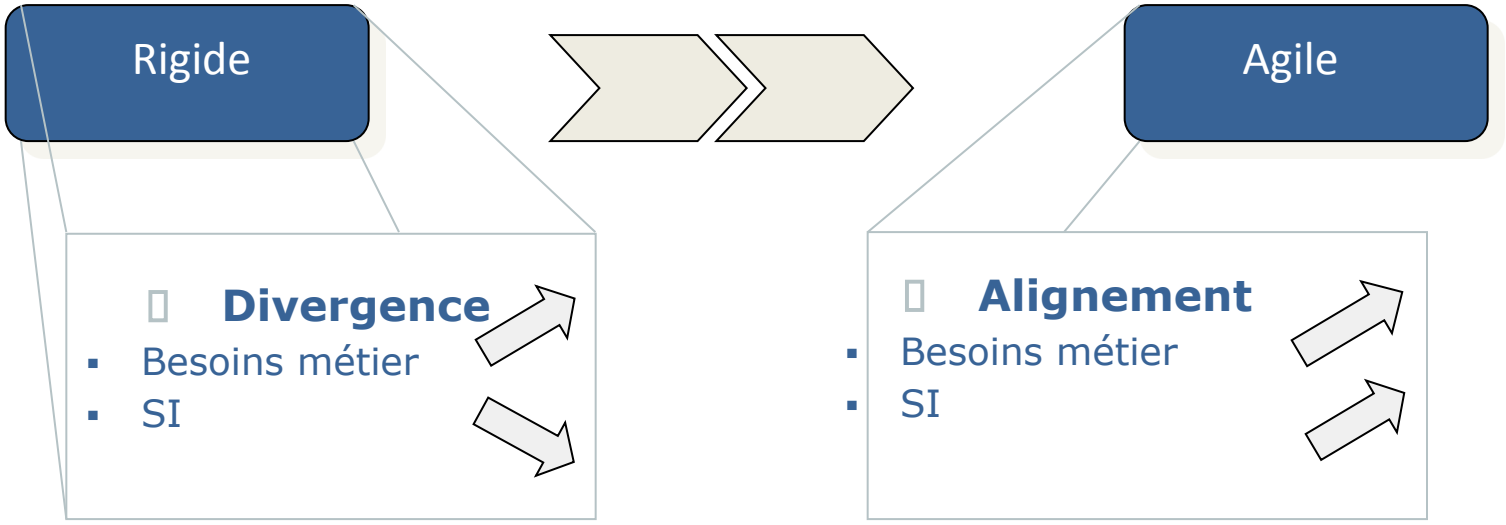
③ Flux
- Typologie
- Qualification des flux

④ Architectures d'échange
- Typologie des architectures
- Solutions logicielles

État des lieux des SI

- **État actuel**
  - Hétérogène
  - Redondant
  - Coût de maintenance

- **État cible**
  - Homogène
  - Rationnel



État des lieux des SI

- La généralisation de l'outil informatique dans les directions métiers fait que les utilisateurs attendent toujours plus du SI, soit au niveau d'agilité, soit au niveau de l'intégration des acteurs externes de leur SI.

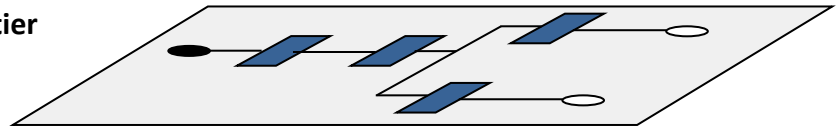
- **Il devient donc vital de trouver des moyens de mieux gérer et homogénéiser le système d'information.**

- **Le constat est que les DSI doivent trouver un moyen d'organiser et de maîtriser cette hétérogénéité.**

- **Et rationaliser la mise en commun d'information transverses à l'entreprise - un référentiel.**

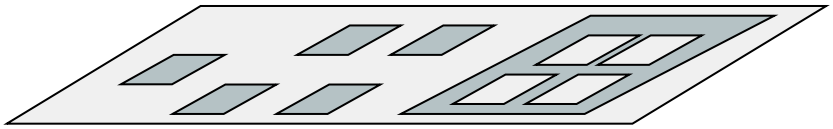
Modèle de référence

Processus métier
Quels métiers ?



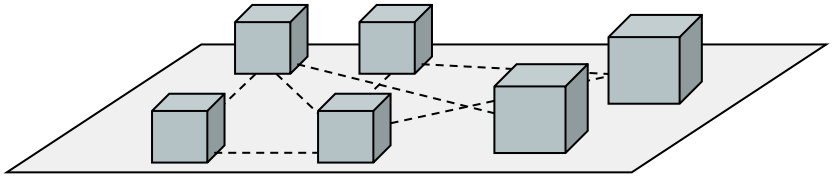
Vue métier
Les processus métier et leurs activités, l'organisation

Fonctionnel
Quoi?



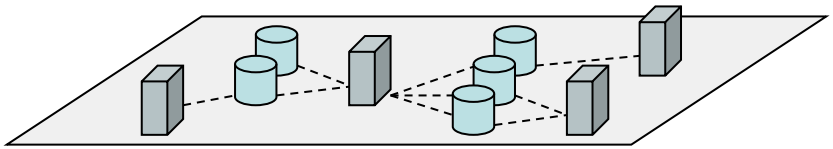
Vue fonctionnelle
Les fonctions du SI supportant les processus métier

Applicatif
Applications & logiciels
Comment?



Vue applicative
Les blocs applicatifs, les messages, les données

Physique
Infrastructure
Avec quoi?



Vue technique
Les matériels, les logiciels, les technologies

Modèle de référence

- La première réponse historique à la rationalisation du SI est la démarche dite l'urbanisation .

- L'urbanisation des SI travaille sur :
  - Processus métiers
  - Communications inter applicative
  - Sur la mise en place d'un référentiel transverses

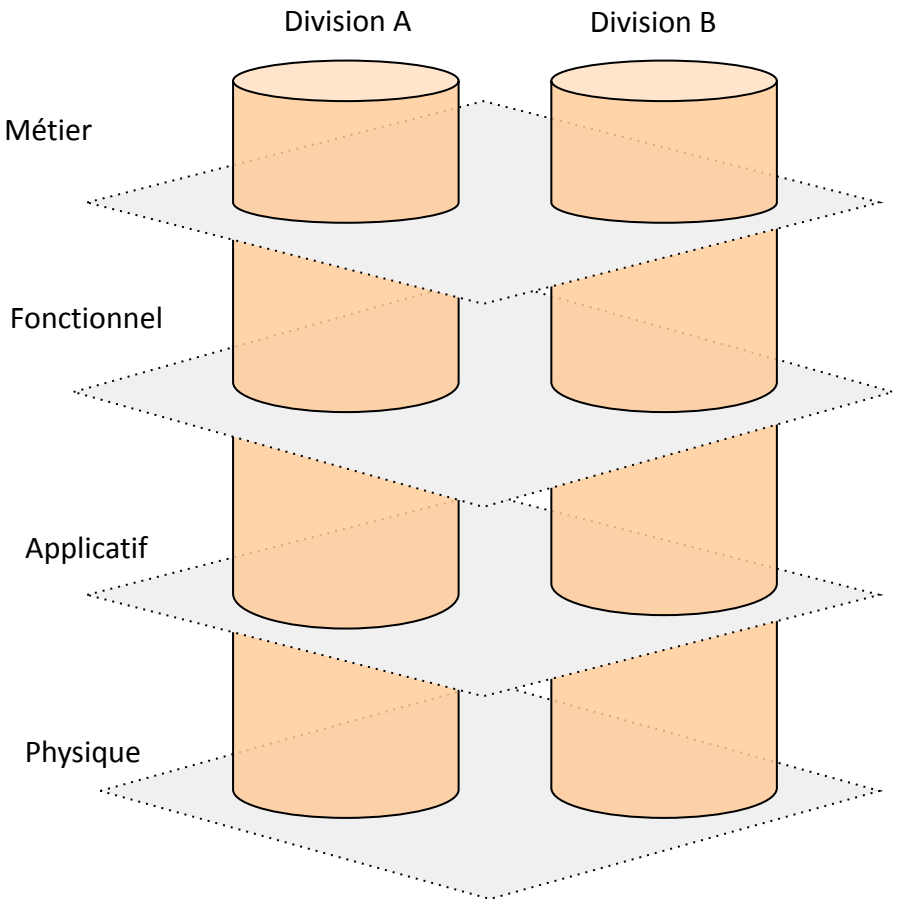
Objectif est d'avoir une vue d'ensemble du SI

Modèle de référence

- Le système d'information d'une entreprise est représenté selon quatre niveaux d'architecture:

  - **L'architecture métier** : processus métier de l'entreprise
  - **L'architecture fonctionnelle** : blocs fonctionnels et flux d'information supports à la réalisation des processus métier, indépendamment des technologies mises en œuvre,
  - **L'architecture applicative :** blocs applicatifs et échanges, supports à la réalisation des blocs fonctionnels et des flux.
  - **L'architecture technique :** infrastructure sur laquelle sont implémentées et exécutés les blocs fonctionnels .

Phénomène vertical



- **Processus rigides**
  - Processus complexes
  - Processus non transférables

+

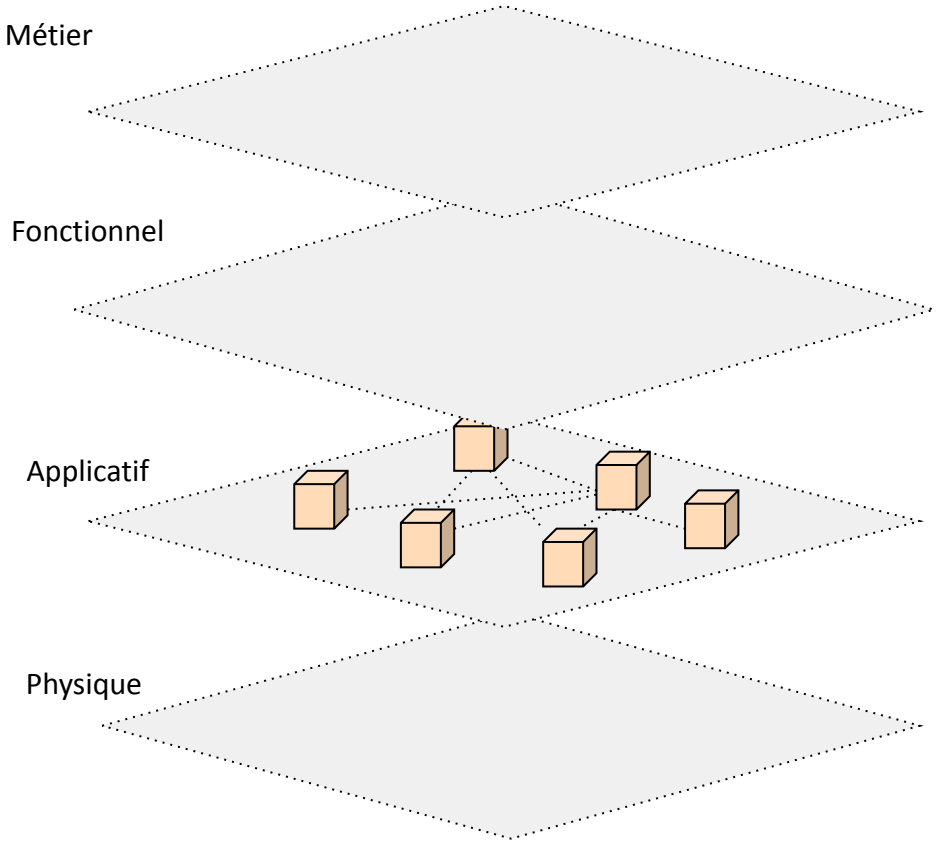
- **Composants peu réutilisables**
  - Hétérogénéité technologique

=

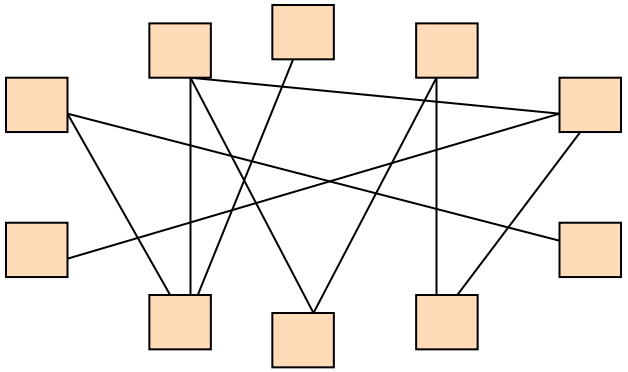
Problématiques des silos applicatifs

Le problème hétérogénéité technologique et plus que les applications sont mal équipés pour communiquer avec les autres blocs on parle de fonctionnement en silos

Phénomène vertical



- **Redondance**
  - Données
  - Traitements
- **Parc applicatif rigide**
  - Interdépendance élevée
  - Difficulté d'évolution



« **Syndrome du plat de spaghettis** ???»

Phénomène vertical

Chaque silo est généralement autonome et isolé en terme de processus, d'interface home/machine, de socle technique.

Ainsi la vision par couche des urbanistes se heurte à une réalité technique qui bloque leur démarche globale de rationalisation.

Et les communications inter applicatives sont souvent gérées au cas par cas en fonctions des besoins d'où l'idée « **Syndrome du plat de spaghettis** »

Sommaire

① Applications d'entreprise
- Etat des lieux des SI
- Modèle de référence

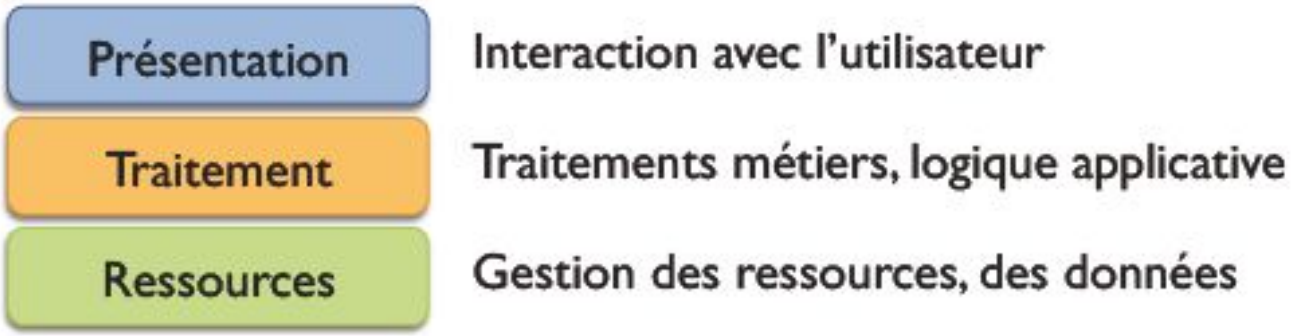
② Patrons d'architecture pour les applications
- Architecture en couches
- Modèle n-tiers
- Composants
- Infrastructures logicielles

③ Flux
- Typologie
- Qualification des flux

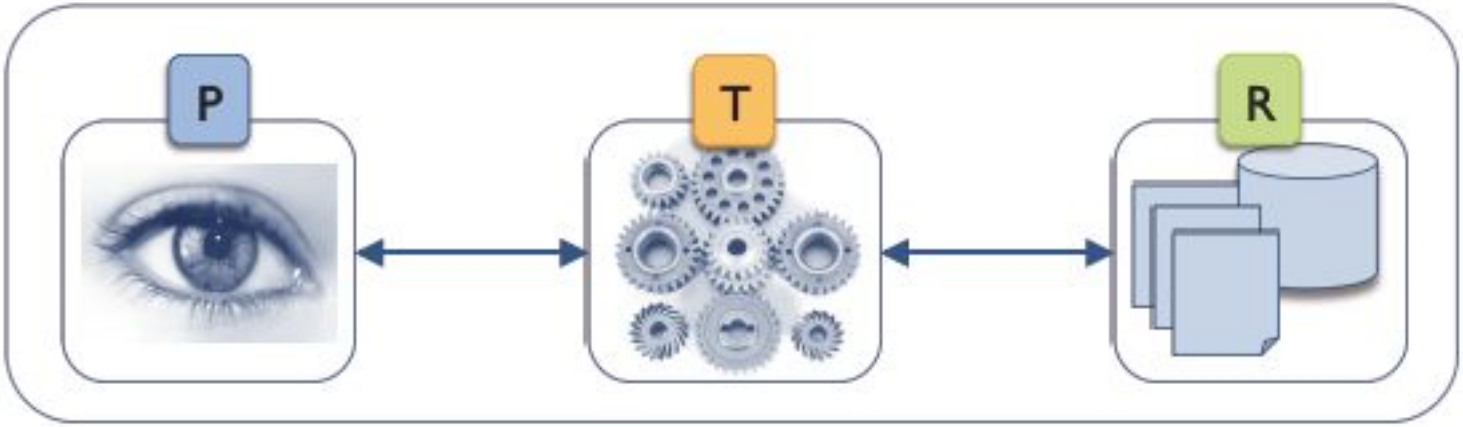
④ Architectures d'échange
- Typologie des architectures
- Solutions logicielles

Couches applicatives (rappel…)

- 3 types de responsabilités = 3 couches principales

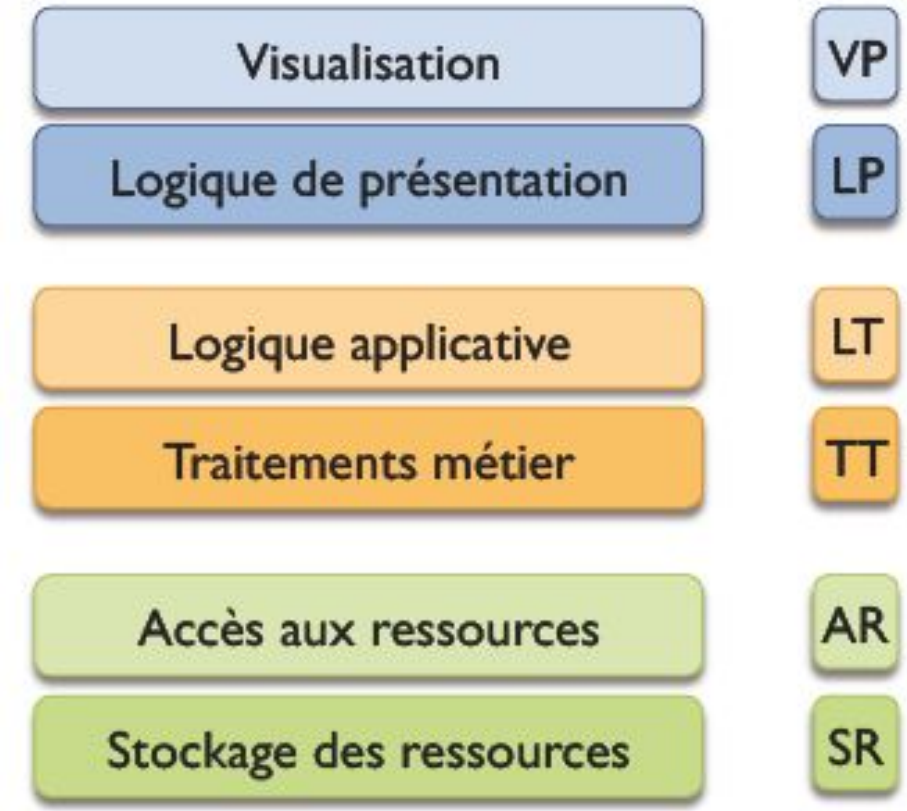


- Principe de conception = séparation des responsabilités



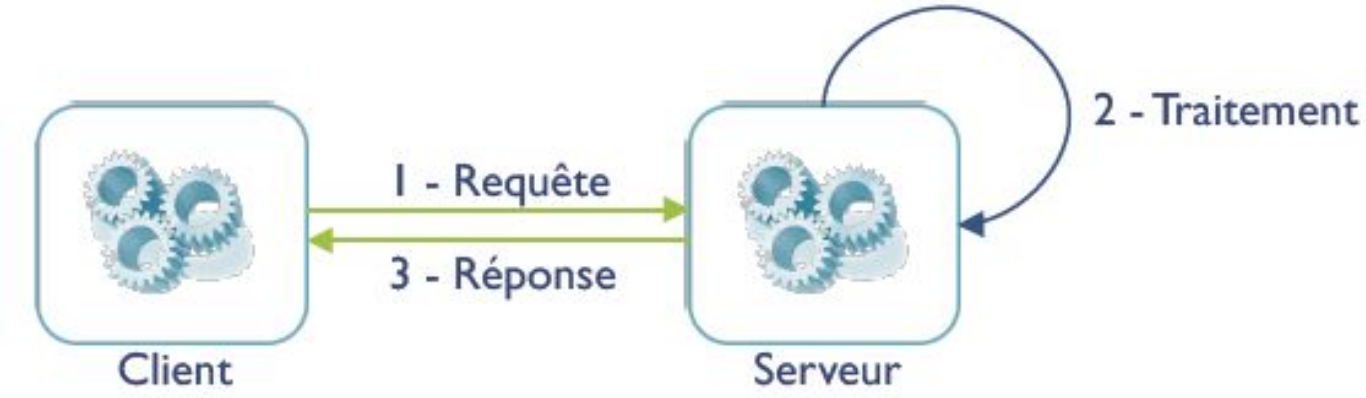
Couches applicatives détaillées

- Vue plus fine des responsabilités



Modèle Client-Serveur (rappel…)

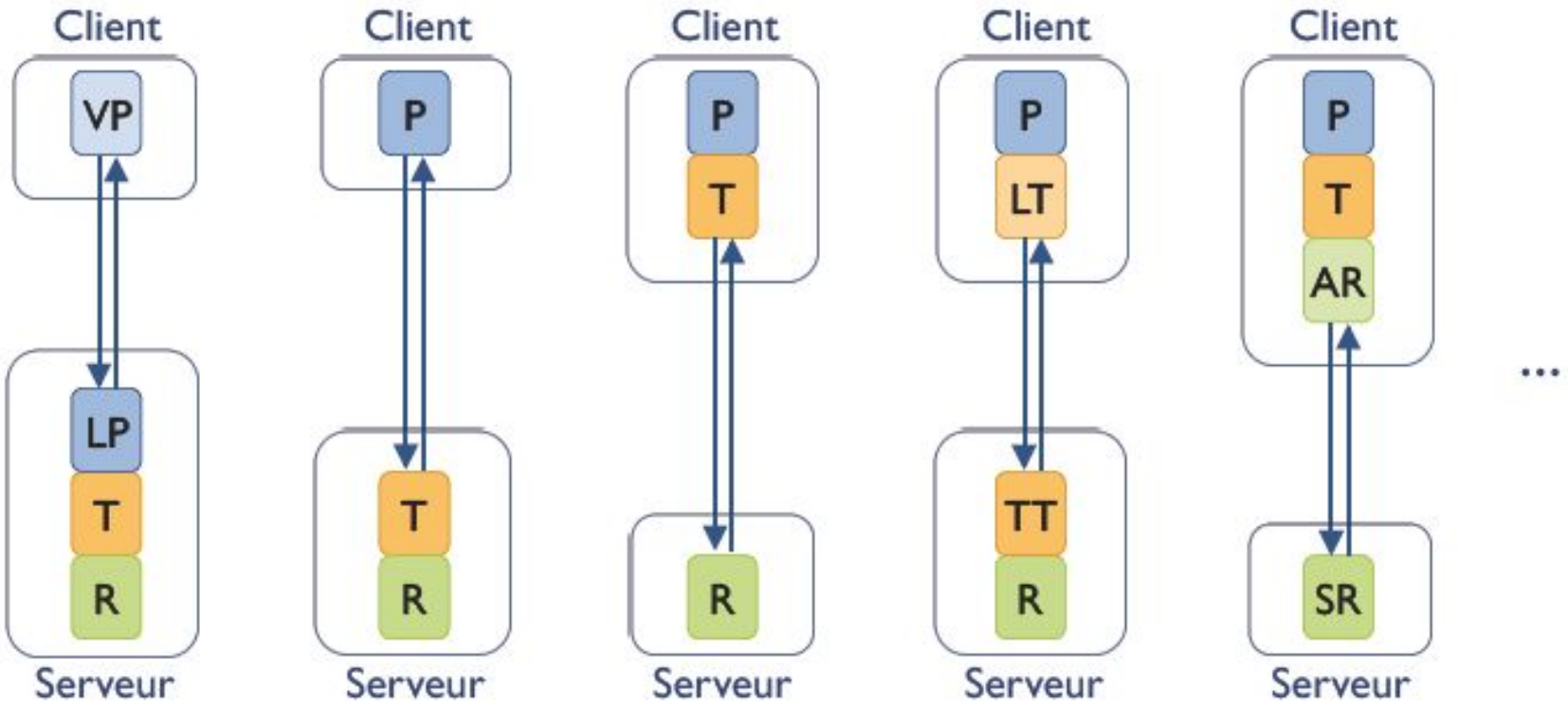
- Modèle Client-Serveur = 2 programmes +1 protocole
  - Programme « serveur » = offre un service a des clients
  - Programme « client » = utilise un service fourni par un serveur
  - Protocole = moyen de communication



- Indépendant de la notion de « machine »
  - Client et serveur sur la même machine
  - Client et serveur sur des machines différentes

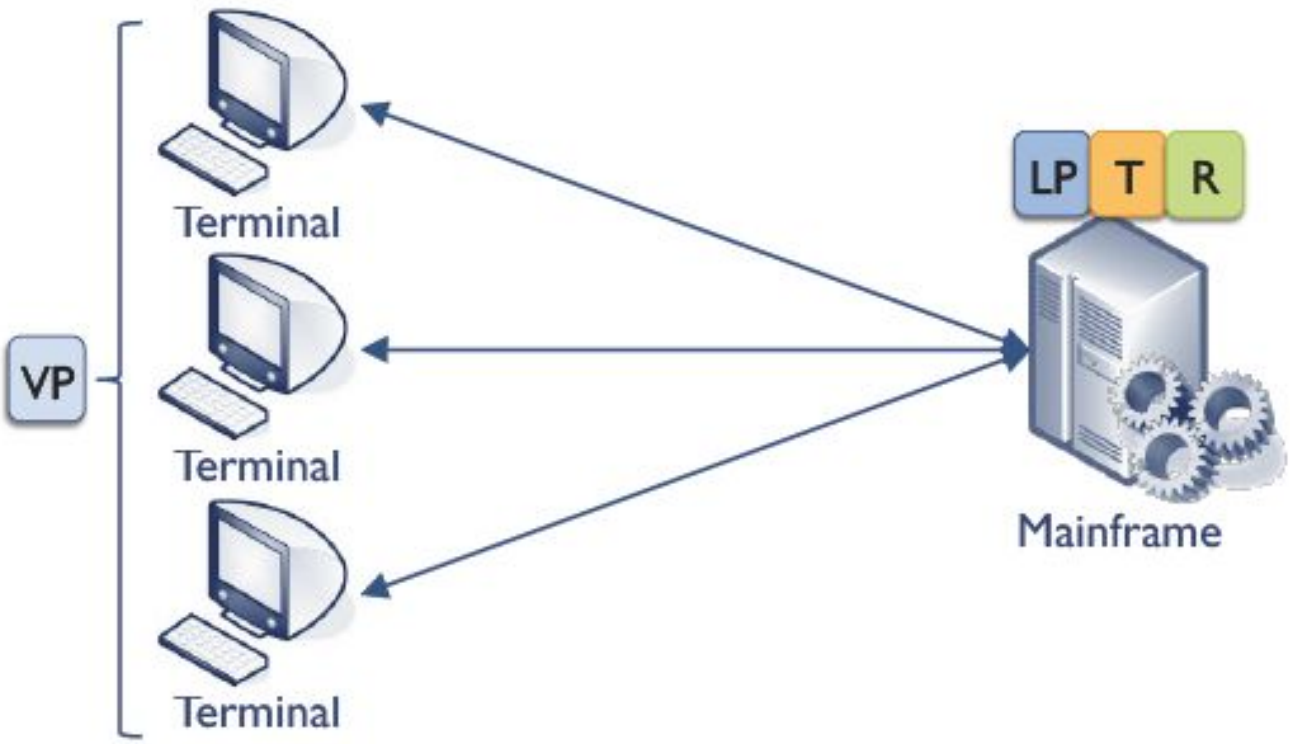
Modèle Client-Serveur
Où placer les couches applicatives ?

- Distribution des couches
- Possibilités multiples = typologies multiples (Gartner)



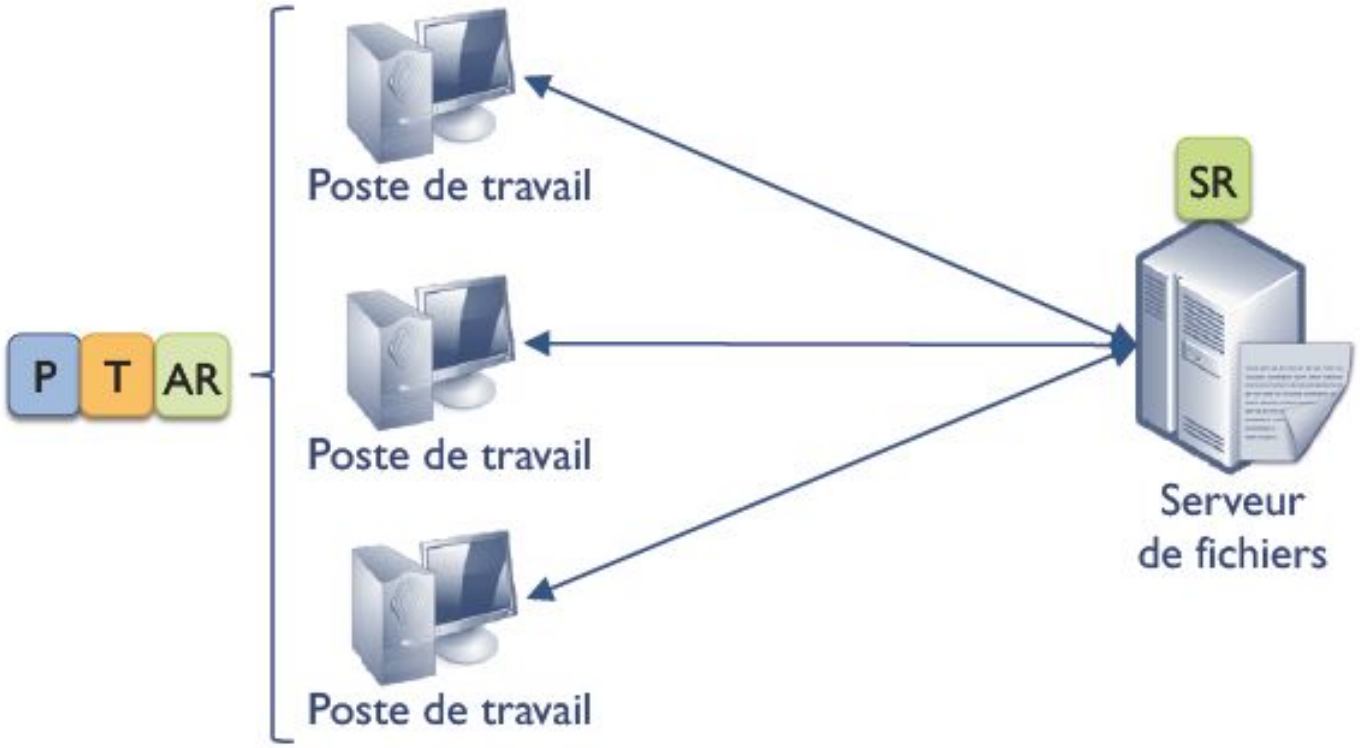
Architecture 1-tiers (mainframe)

- L'application est sur un serveur (éventuellement distant)
- Le client est une application « légère » de visualisation (client « passif »)
- Exemple :



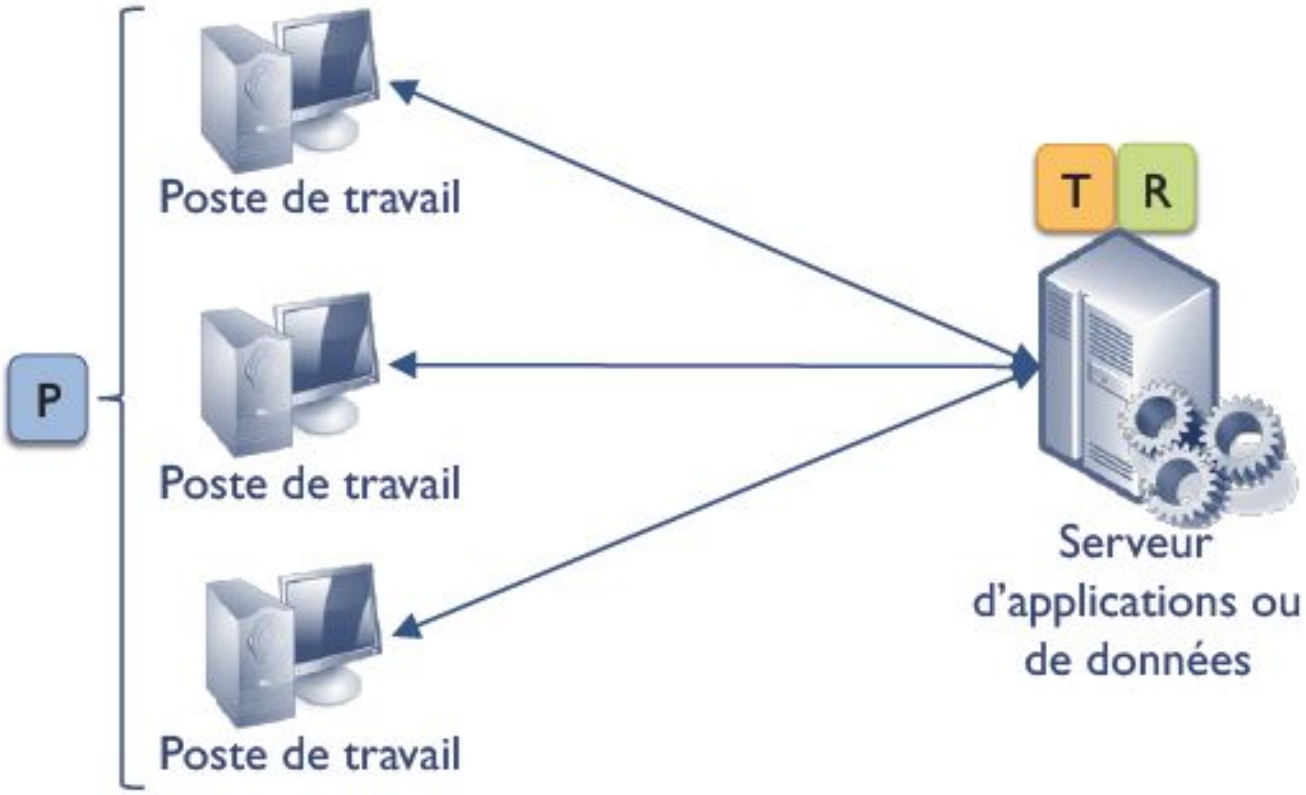
Architecture 1-tiers (client autonome)

- L'application est sur le client
- Les données sont sur un serveur (éventuellement distant)
- Exemple :



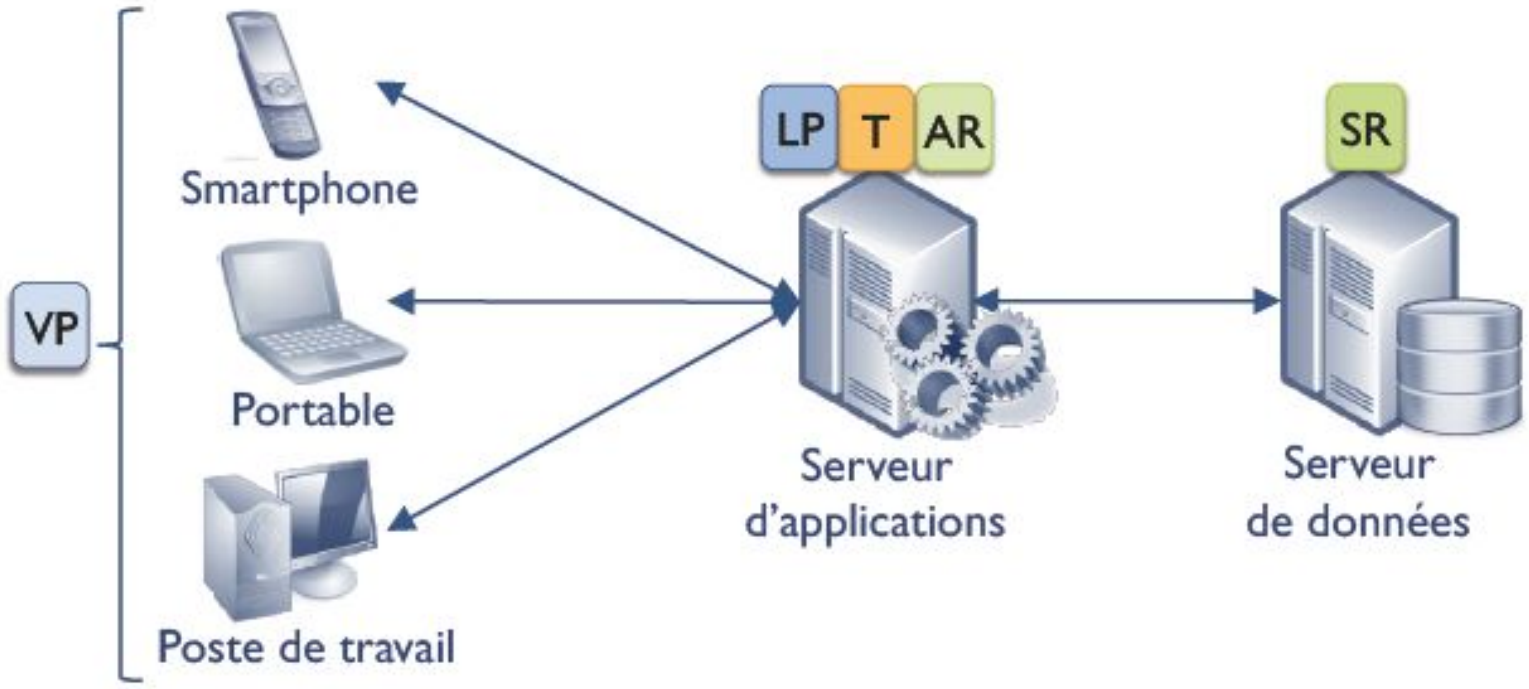
Architecture 2-tiers (client lourd)

- Le cœur de l'application est sur un serveur (éventuellement distant)
- La couche présentation (IHM) de l'application est sur le client
- E



Architecture 3-tiers (client léger)

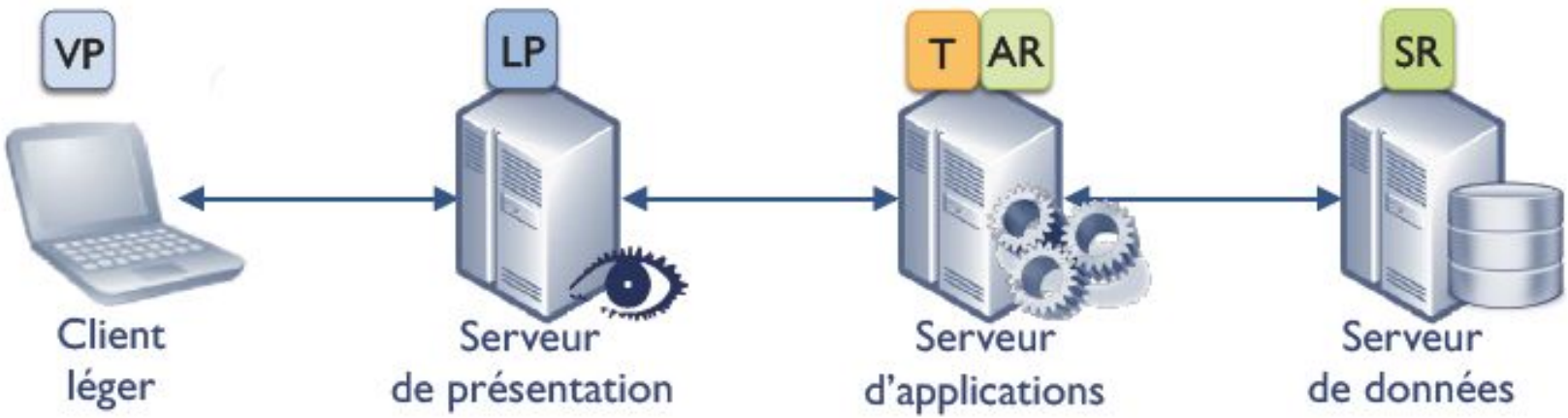
- Le cœur de l'application est sur un serveur
- Les données sont sur un autre serveur
- Le client est une application « légère » de visualisation (ex : navigateur web)
- Exemple :



Architecture n-tiers

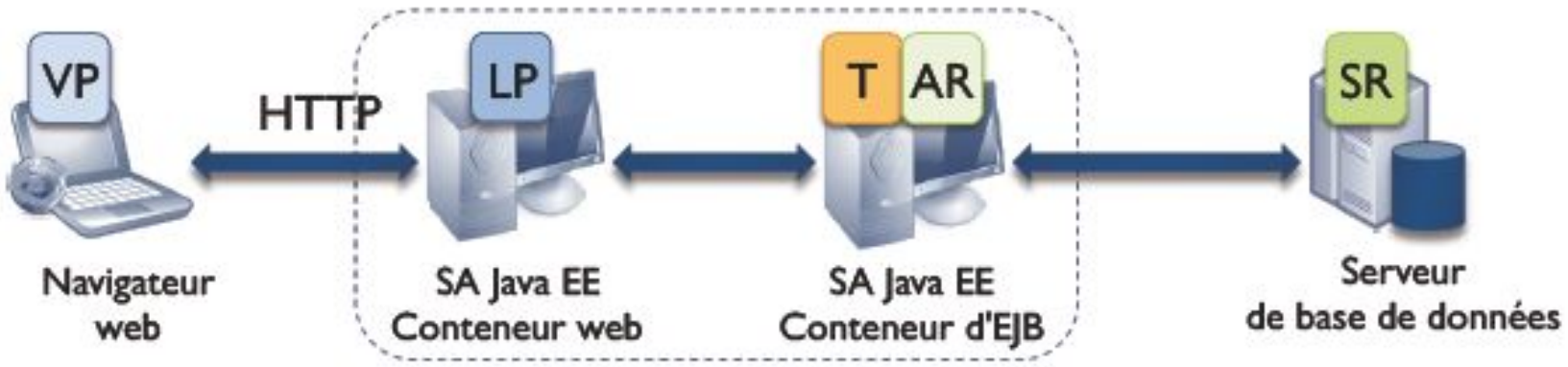
- Généralisation des modèles précédents (remarque : un serveur peut être un client pour un autre serveur !)
- Distribution des responsabilités en 4 ou + tiers

- Architecture type :



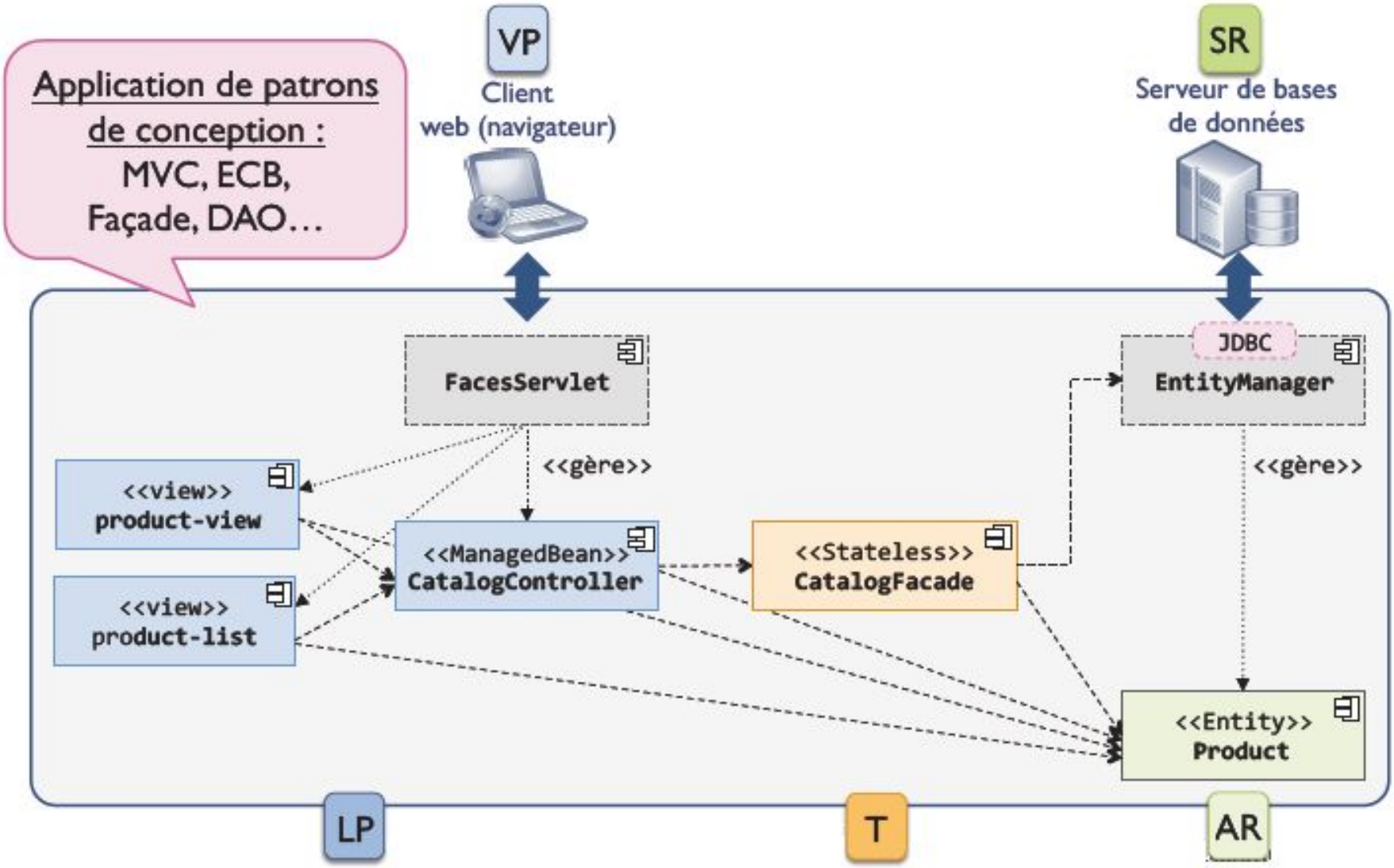
Exemple avec Java EE

- Application de gestion de catalogue de produits
- Architecture 3 ou 4 tiers :
  - Client léger
  - Serveur de présentation (pouvant être fusionné avec le serveur de traitement dans le cas de Java EE)
  - Serveur d'application
  - Serveur de données



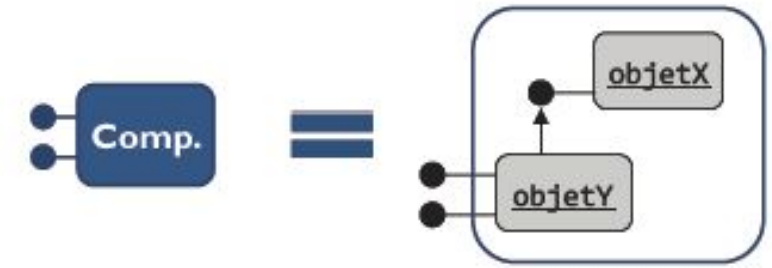
Comment « découper » une application en tiers ?

Gestion de catalogue avec Java EE



Composants

- Composant = unité logique de traitement
  - Assemblage d'objets interdépendants
  - Rend un service (fonction)
  - Vue boîte noire



- Propriétés
- Identification : nom unique, référencé dans un annuaire
- Indépendance : utilisable tout seul
- Réutilisation : utilisable dans différents contextes
- Intégration : combinable avec d'autres composants

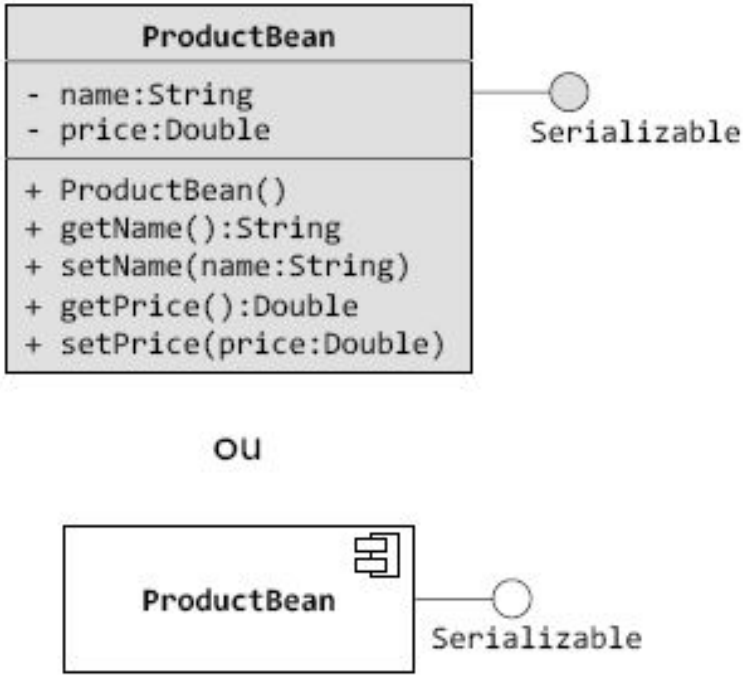
- Technologies d'implémentation multiples

Exemple avec Java : JavaBeans

- Composant implémenté par une classe Java
  - ≈ Plain Old Java Object (POJO)
- Conventions à respecter
  - Sérialisation
  - Constructeur par défaut
  - Propriétés privées avec accesseurs (encapsulation et introspection)
    - Public *<returntype>* get*Propertyname()*
    - Public void set*Propertyname(parameter)*
  - Méthodes d'interception d'événements
    - Utilisation d'écouteurs et génération d'événements
      - Ex : PropertyChangeListener

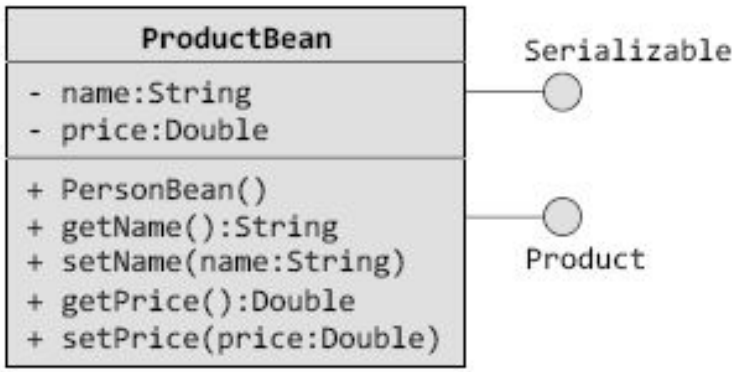
JavaBeans
Exemple

```
public class ProductBean implements Serializable {
    private static final long serialVersionUID = 1L;

    private String name;
    private Double price;

    public ProductBean() {
        this.name = "";
        this.price = 0.0;
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Double getPrice() {
        return this.employed;
    }
    public void setPrice(Double price) {
        this.price = price;
    }
}
```

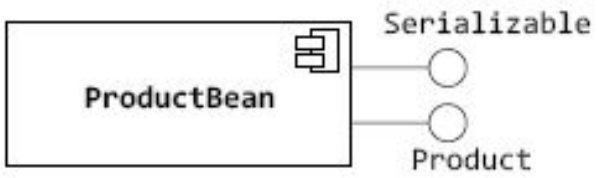


Interfaces d'un JavaBean

```java
public interface Product {
  public String getName();
  public void setName(String name);
  public Double getPrice();
  public void setPrice(Double price);
}
```



ou

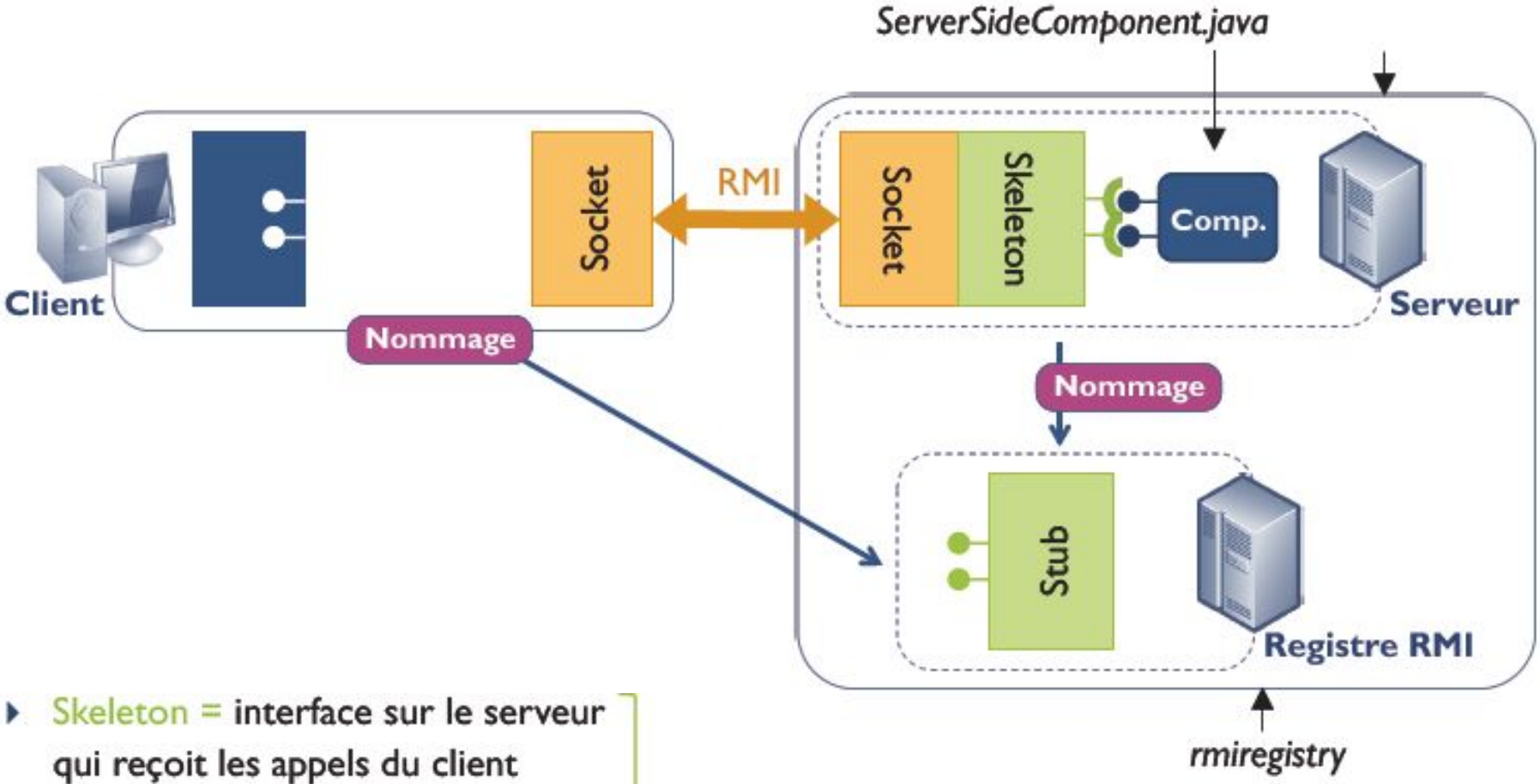



```java
public class ProductBean implements Product,
                                    Serializable {
    private String name;
    private Double price;

    public ProductBean() {
        this.name = "";
        this.price = 0.0;
    }

    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Double getPrice() {
        return this.employed;
    }
    public void setPrice(Double price) {
        this.price = price;
    }
}
```

Composants distribués

- Un Client veut utiliser un composant qui se trouve sur un Serveur (distant)

Solution Java :
RMI (Remote Method Invocation)

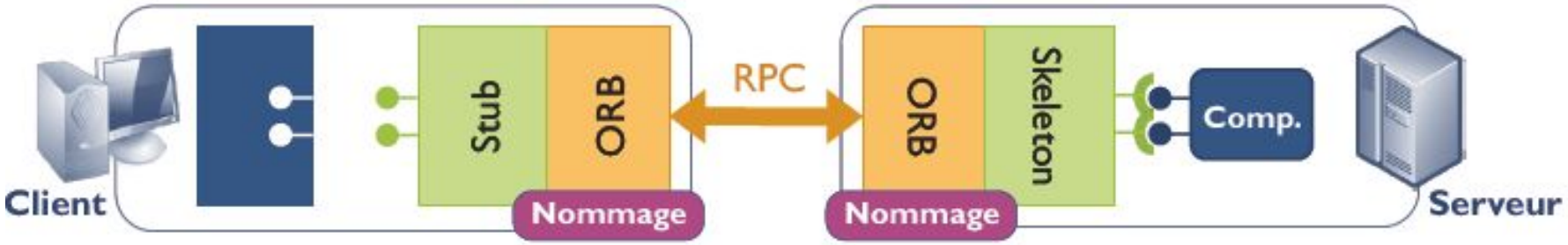


- Skeleton = interface sur le serveur qui reçoit les appels du client
- Stub = interface sur le client qui envoie les appels au serveur

dérivés du composant

Solution CORBA

- Un Client veut utiliser un composant qui se trouve sur un Serveur (distant)



- Object Request Broker (ORB) = « bus logiciel » qui permet au client de rechercher le composant sur le serveur et de communiquer avec lui
- Skeleton = interface sur le serveur qui reçoit les appels du client
- Stub = interface sur le client qui envoie les appels au serveur

dérivés du composant

- Remote Procedure Call (RPC) = appel de procédure distant

Problématiques

- Applications n-tiers à base de composants = composants distribués avec responsabilités distribuées

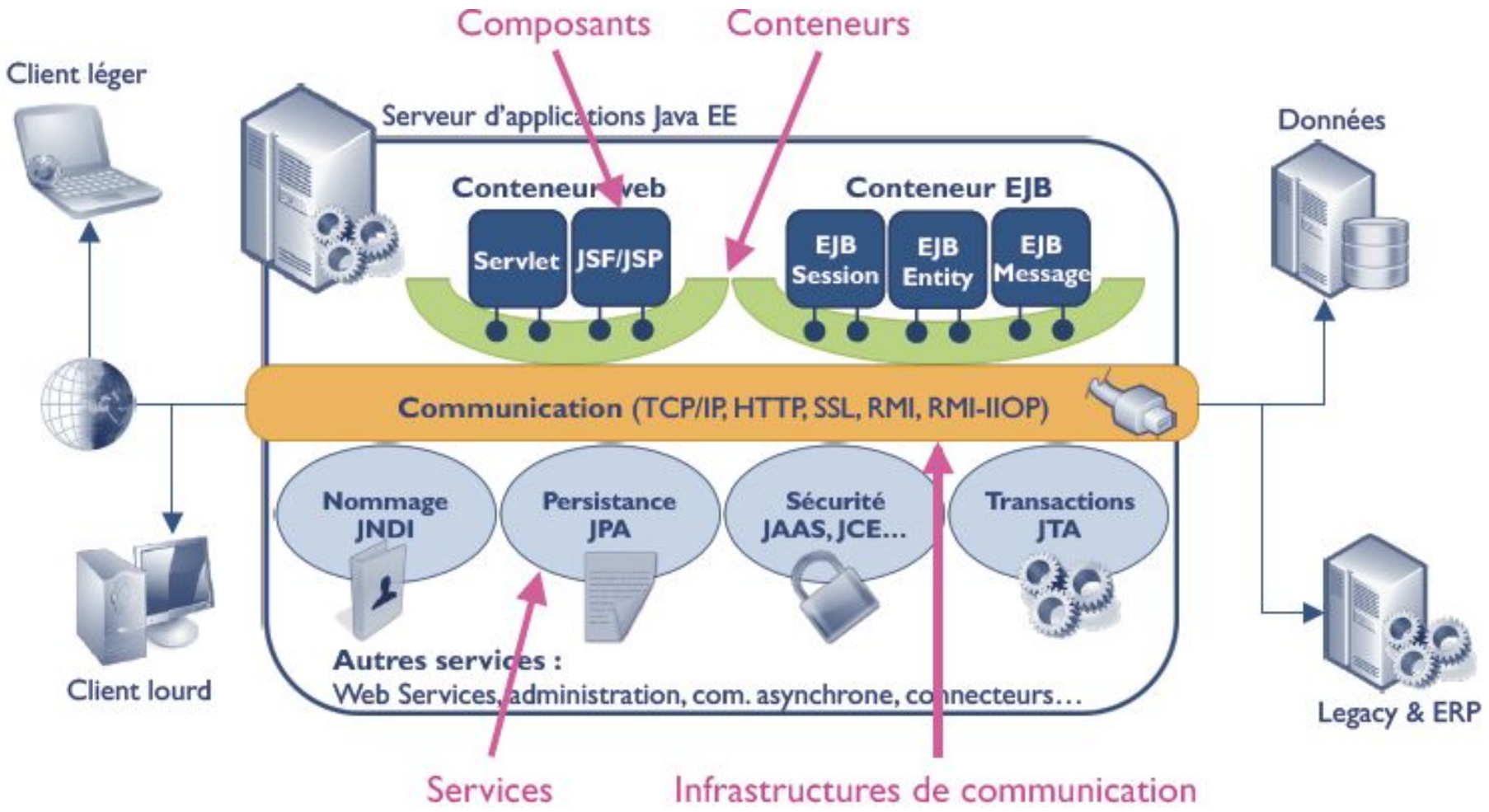
- Problématiques :
  - Complexité
    - Conception des composants et des applications
    - Développement des composants et des applications
    - Gestion des aspects transverses : sécurité, disponibilité, communication, persistance, transactions…
  - Interopérabilité des composants
  - Administration des composants et des applications
  - Sécurisation de bout en bout des applications
  - …

Serveurs d'application & frameworks de développement

- Serveur d'Applications (SA) = conteneur et fournisseur de services pour des composants et des applications
  - Gestion du cycle de vie des applications et des composants
  - Administration des applications et des composants
  - Allocation de ressources
    - Processeur, mémoire, réseau, composants logiciels externes…
  - Support pour les aspects transverses
    - Sécurité, gestion des transaction, accès réseau…
  - Support pour l'interopérabilité

- Frameworks de développement =
  - Cadres pour la conception et le développement de composants (déployés sur SA) et d'applications à base de composants

Exemple : SA + framework Java EE



Sommaire

① Applications d'entreprise
- Etat des lieux des SI
- Modèle de référence

② Patrons d'architecture pour les applications
- Architecture en couches
- Modèle n-tiers
- Composants
- Infrastructures logicielles

③ Flux
- Typologie
- Qualification des flux

④ Architectures d'échange
- Typologie des architectures
- Solutions logicielles

Echanges d'informations ?

- Flux = données qui passent d’un point A à un point B
  - D’une application à une autre,
  - D’un module applicatif à un autre
  - D'un utilisateur à un autre
  - D’une base de données à une autre
  - D’une entreprise à une autre
  - …
- Exemples de flux
  - Transfert de fichier
  - Partage de fichier
  - Appel de procédure distant
  - Requête sur une base de données
  - …

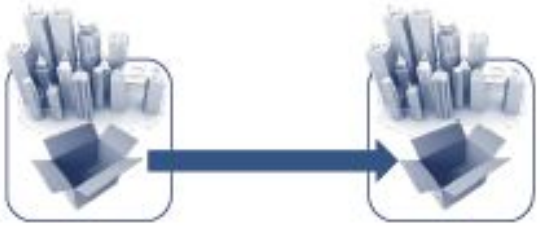
Périmètre

- Flux privé = intra-application (entre composants)

- Flux public = inter-applications
  - Bonne gestion des flux publics = flexibilité !

- Flux AtoA = flux inter-applications sur un périmètre intra-entreprise

- Flux BtoB = flux inter-applications sur un périmètre inter-entreprises
  - *Exemple : envoi d'une commande de pièce à un fournisseur*



Granularité / Fréquence

« Evénementiels »

- Flux unitaire = données transmises une à une

- Flux au fil de l'eau = données transmises dès qu'elles sont disponibles

Exemple : transmission des commandes à la plate-forme logistique au fur et à mesure de leur validation

« Batch »

- Flux de masse = données regroupées en lots

- Flux cadencés = données transmises à des moments prédéterminés

Exemple : transmission chaque soir des données concernant l'ensemble des ventes de la journée pour stockage dans l'entrepôt de données

Exemples de flux

- Souvent pour les flux unitaires au fil de l'eau (« événementiels ») :
  - Appels distants entre composants (CORBA, RMI…)
  - Transferts de fichiers
  - Partage de base de données
  - Electronic Data Interchange (EDI) = norme définissant le(s) protocole(s) + le format d'échange de données pour le B2B
  - Web Services

- Souvent pour les flux de masse cadencés (« batch ») :
  - Transferts de fichiers
  - Partage de base de données
  - Batch = script qui ordonne et cadence un déplacement de données en volume
    - Solution « historique » et sans doute encore l'une des plus utilisées
  - Extract-Transform-Load (ETL) = progiciel de batch « à grande échelle » permettant de connecter des entrepôts de données

Modalité

- Flux synchrone = **bloquant pour l'émetteur et le récepteur**
  - Suppose la disponibilité de l'émetteur et du récepteur au même moment
  - *Exemple : appel de méthode RMI*
- Flux asynchrone = **non bloquant** (émission / réception différées)
  - Suppose l'existence d'une zone de stockage intermédiaire
  - *Exemple : emails*
- Flux requête-réponse = l'émetteur et le récepteur se connaissent
  - Contact direct
  - *Exemple : récupération d'une donnée dans un référentiel*
- Flux publication-abonnement = les récepteurs s'abonnent aux flux sans connaître les émetteurs
  - Contact indirect
  - *Exemple : abonnement aux mises à jour d'un référentiel de données*

Sommaire

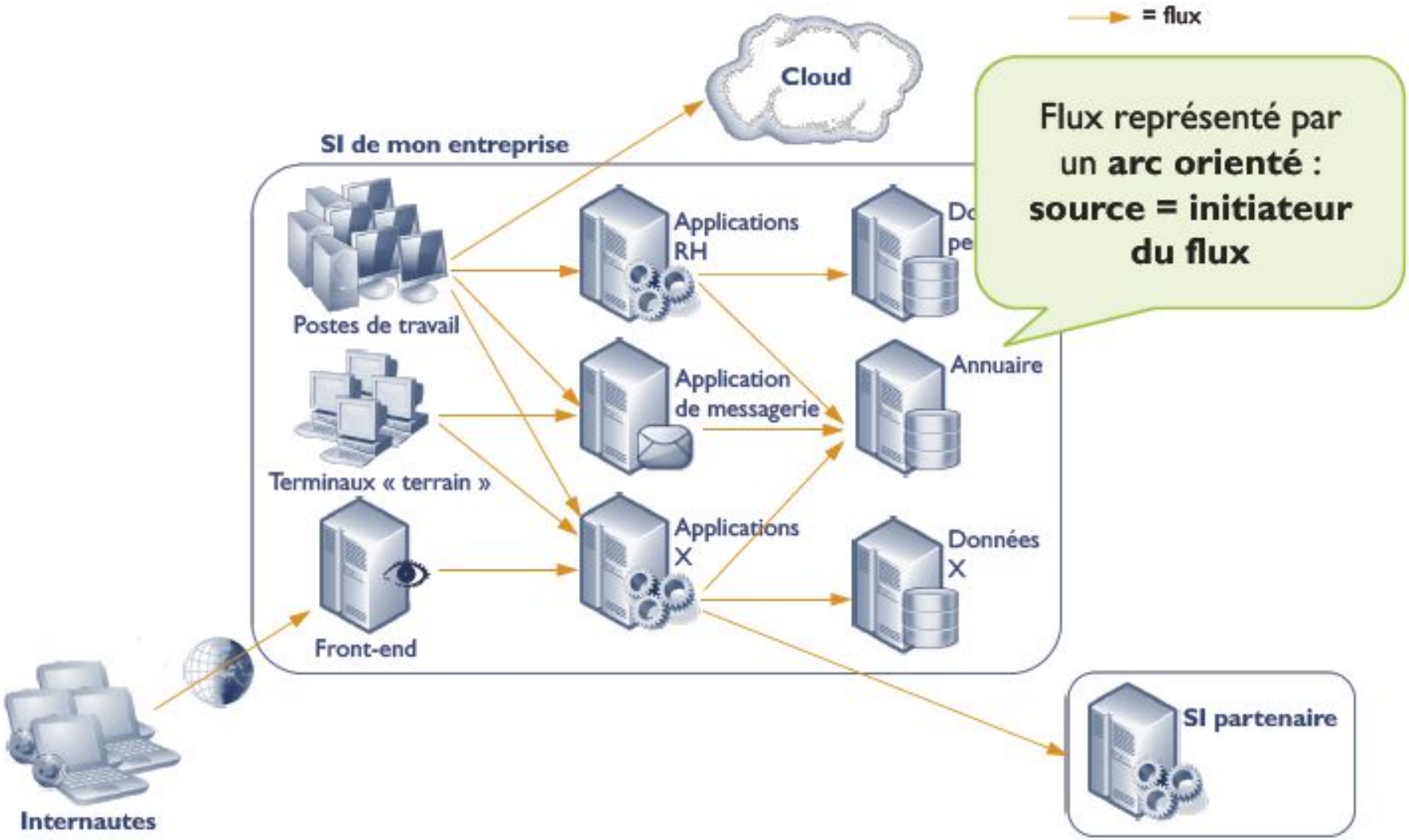
① Applications d'entreprise
- Etat des lieux des SI
- Modèle de référence

② Patrons d'architecture pour les applications
- Architecture en couches
- Modèle n-tiers
- Composants
- Infrastructures logicielles

③ Flux
- Typologie
- Qualification des flux

④ Architectures d'échange
- Typologie des architectures
- Solutions logicielles

Architecture point-à-point



Analyse des solutions point-à-point

- Architecture « intuitive »
- Simplicité de mise en œuvre dans le cas où le nombre d'applications à intégrer est faible

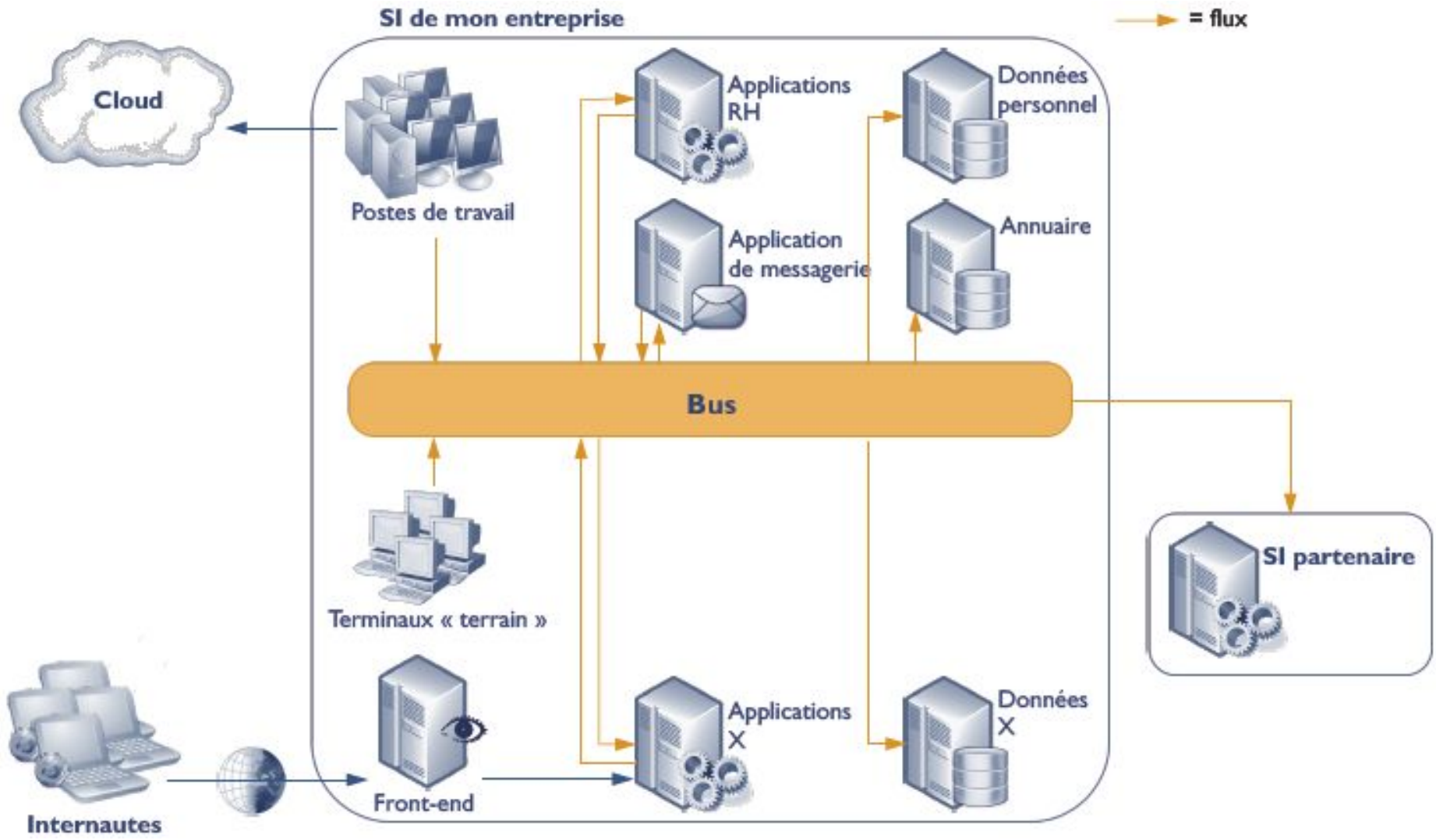
- Efficacité des échanges directs

- Problème de passage à l'échelle
  - Si N applications, N(N-1)/2 liens…
  - Effet « plat de spaghettis »

- Evolutivité très réduite
  - Intégration d'une nouvelle application = ajout de nombreux nouveaux liens
  - Couplage fort entre les applications
    - ➜ fort impact des évolutions (notamment interfaces)

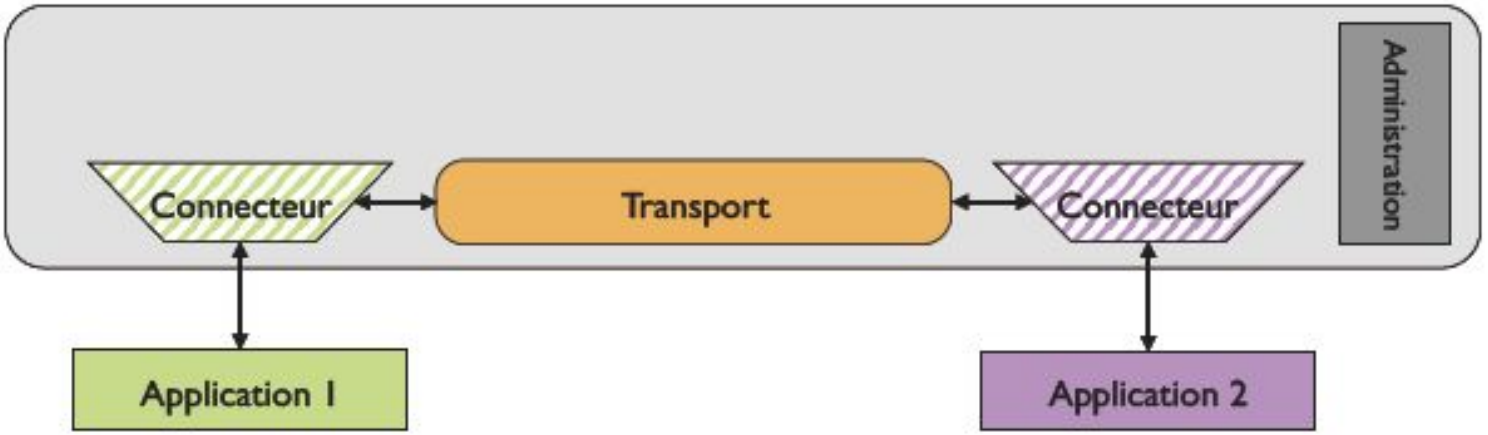
- Exploitation et administration complexes
  - Manque de visibilité sur les échanges

Architecture bus



Exemple de solution de type bus : le MOM

- Middleware Orienté Message (MOM) = bus logiciel de transport qui permet à des applications de recevoir des messages émis par d'autres
  - Connectivité : supporte différents protocoles de communication
  - Transport :
    - Garantit l'acheminement (intégrité, gestion des erreurs)
    - Gère différentes modalités : synchrone/asynchrone, publication/abonnement…
    - Gère les transactions
  - Existe aujourd'hui sur la plupart des serveurs d'applications



Exemple avec Java EE : JMS et EJB Message

- JMS (Java Message Service) = interface Java standard pour les MOM
  - Files d'attentes (queues) *pour le mode requête / réponse*
  - Sujets (topics) *pour le mode publication / abonnement*

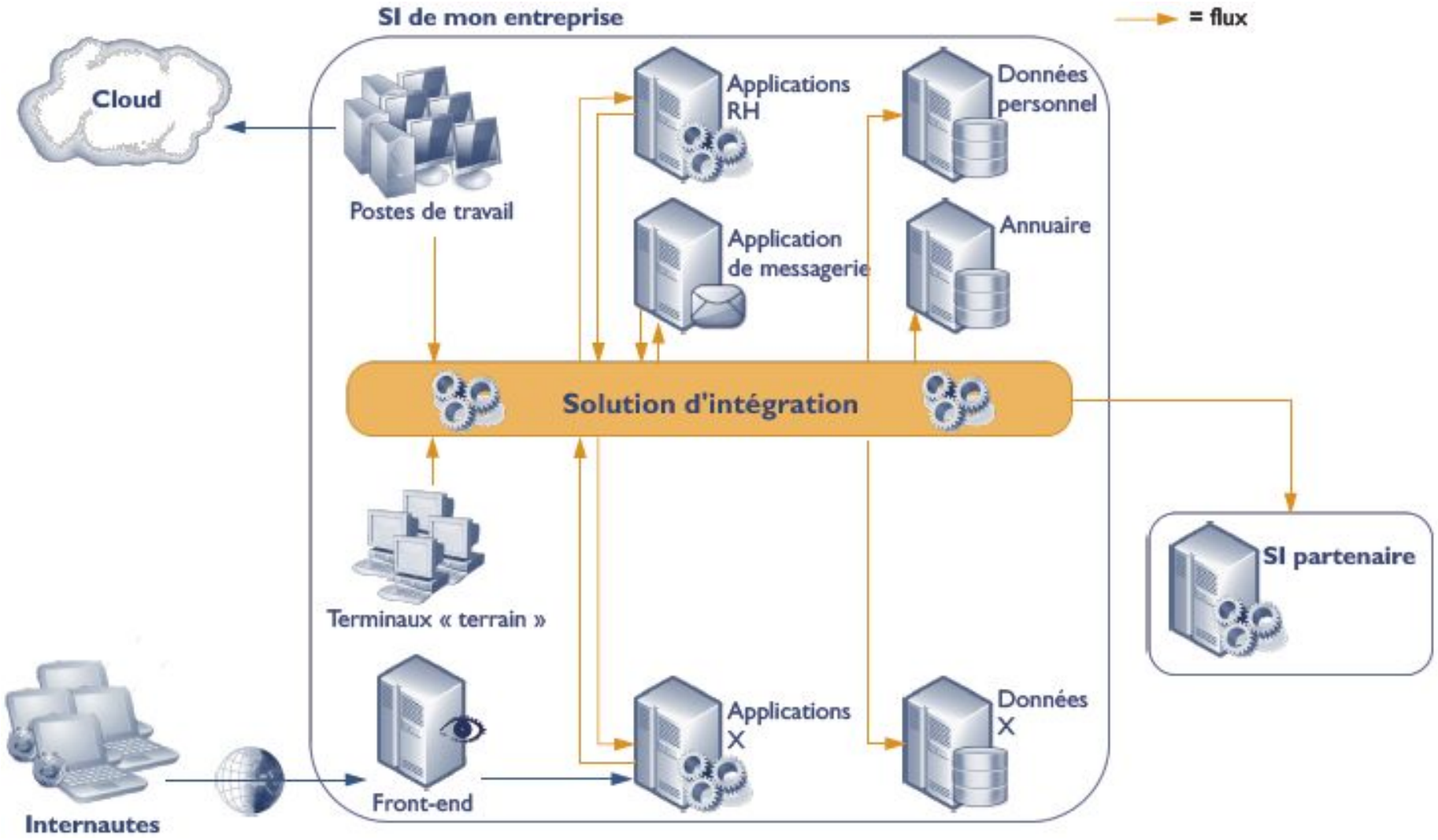
- EJB Message = composant invoqué par messages
  - Traite les messages postés dans une file / sujet
  - Poste des messages réponse dans la file / sujet

Analyse des solutions de type bus

- Passage à l'échelle facilité
  - Si N applications, au plus N liens bidirectionnels
- Meilleure évolutivité
  - Intégration d'une nouvelle application = un seul connecteur
- Couplage faible entre les Applications
- Services de transport
  - Acheminement garanti des données (reprise sur erreur, gestion des doublons)
  - Intégrité des données

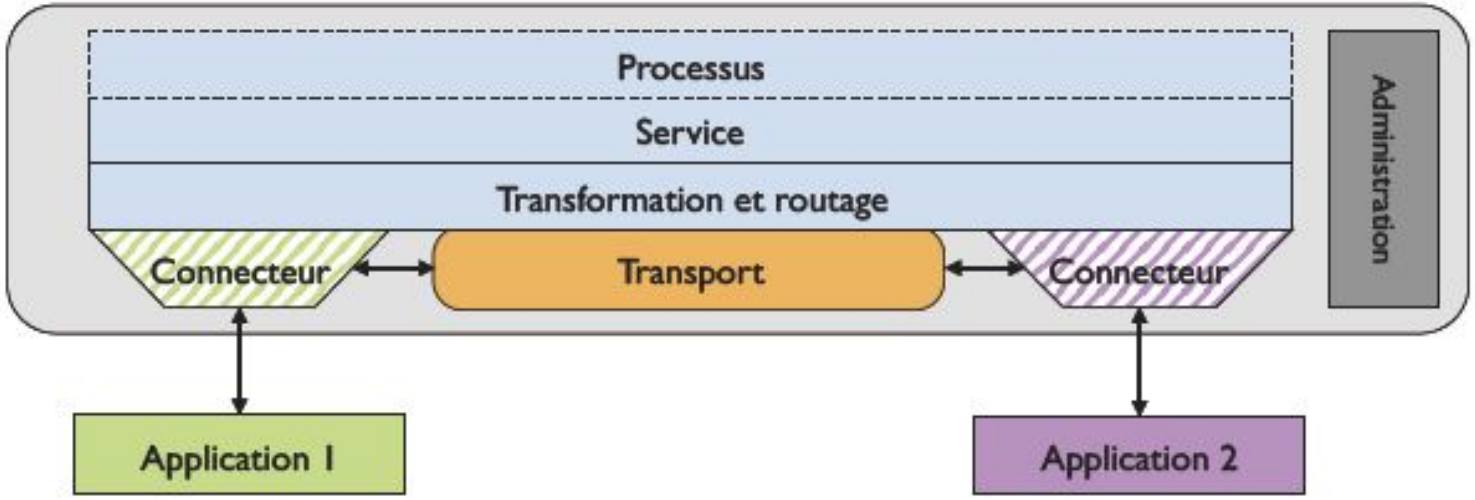
- Adhérence forte entre les applications et le bus
- Couplage fort entre les formats des données
  - ➜ fort impact des évolutions du format d'échange
- Plate-forme centralisée
  - ➜ hautement critique
  - ➜ goulot d'étranglement

Architecture intégrée



Exemple de solution intégrée : l'EAI

- Enterprise Application Integration (EAI) = progiciel d'intégration inter-applicative
- Connectivité & Transport
- Transformation : gère l'hétérogénéité des formats et des valeurs
- Routage : adresse intelligemment les données aux différents destinataires
- Service : encapsule la logique d'intégration
- + Eventuellement, processus : orchestre les services d'intégration



Analyse des solutions EAI

- Relations entre processus métiers et échanges inter-applicatifs plus lisibles
  - *Urbanisation fonctionnelle*
  - *+ Urbanisation technique*
- Services applicatifs riches
- Coûts d'administration moins importants
- Coûts de développement réduits
- Important travail d'urbanisation et /ou de réorganisation fonctionnelle indispensable
  - *Sinon « plat de spaghetti » dans l'outil…*
- Experts indispensables (et malheureusement très rares)
- Projets transverses par essence
  - Difficulté à établir les responsabilités
  - Problématiques organisationnelles (nombreux acteurs, besoin de processus)
- Technologies d'interconnexion propriétaires

Autre solution intégrée : ESB

- Enterprise Service Bus (ESB) ≈ EAI basé sur les standards
  - Connectivité & Transport
  - Transformation : gère l'hétérogénéité des formats et des valeurs
  - Routage : adresse intelligemment les données aux différents destinataires
  - Service : encapsule la logique d'intégration
  - Processus : orchestre les services d'intégration
  - Monitoring : supervise le déroulement des processus

- L'ESB est considéré comme le socle technique de l'approche SOA

Enterprise Service Bus (ESB)