CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL MIDI-PYRENEES

Architecture des Systèmes d'Information

Définition d'un Système d'Information

- Un système d'information (noté SI) représente l'ensemble des éléments participant à la gestion, au stockage, au traitement, au transport, à la diffusion et à la représentation de l'information au sein d'une organisation
- Aujourd'hui, la généralisation des applications web rend nécessaire une très forte interopérabilité des systèmes d'information
- Le système réparti est un moyen pour bâtir l'architecture de ces nouveaux systèmes d'information

Les architectures des SI

On va présenter et introduire :

- Les architectures client/serveur
- Les architecture 3-tiers et multi-tiers (tier = étage)
- Les architectures distribuées
- Les architectures orientées services (SOA)
- Les architectures orientées WEB (WOA)

Les architectures client/serveur

- Des logiciels clients envoient des requêtes à un ou plusieurs serveur (de préférence un)
- Serveurs spécialisés (serveur de fichier, de messagerie, de données, …)
- Le serveur est un esclave, le client est un maître
- Le client et le serveur doivent utiliser le même protocole de communication
- Le rôle d'un serveur est de centraliser l'information
- Le client et le serveur sont le plus souvent liés par la même couche spécialisée de communication

- Une architecture client/serveur est une architecture 2-tiers (1 pour le client, 1 pour le serveur)
  - Tier = couche (logiciel ou applicatif)
- Pour communiquer un client et un serveur utilise un "middleware"

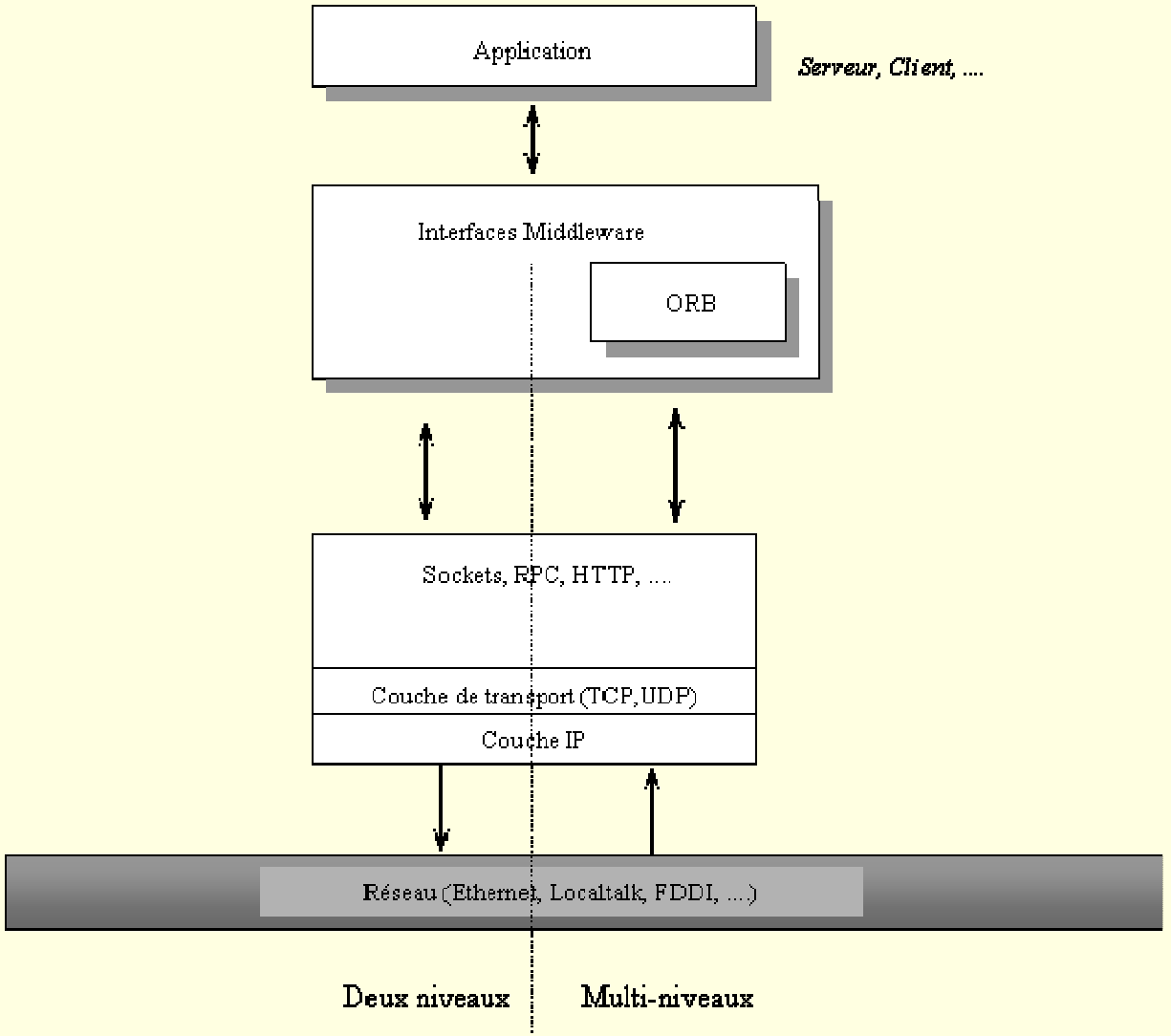
Le middleware (1/3)

- Un "Middleware" (littéralement: "intermédiaire d'articles fabriqués" ou "interlogiciel") est un ensemble de composants logiciels assurant les interfaces de communication des données et l'appel éventuel aux traitements entre le serveur et les clients
- Un middleware s'appuie sur un empilement de couches logiciels (ou tiers) plus ou moins sophistiquées assurant la communication physique des informations.
- Le niveau d'encapsulation et d'abstraction de cette pile de couches logiciels est déterminant dans la réalisation de l'ensemble afin de maîtriser :
  - le coût de développement
  - la robustesse et l'évolution
  - la facilité de mise en œuvre

Le middleware (2/3)

- Exemples de middleware : EAI, ETL, CORBA, HLA, file d'attente de message, pare-feu, ODBC, NEXUS, CFT (SopraGroup).
- Le middleware se situe "au-dessous" de l'applicatif, "au-dessus" du système d'exploitation et "entre" deux logiciels ayant besoin de communiquer entre eux.
- Par exemple, le couple [SQL*Net + ODBC] forme un middleware.
- Les middleware les plus en vogue dans les architectures dites trois tiers sont :
  - les middleware "orientés objets ou composants distribués" : ce sont les ORB ou Object Request Broker
  - les middleware "transactionnels" : ce sont les moniteurs transactionnels (comme CICS d'IBM, Tuxedo de BEA, MTS de Microsoft, JTS de Sun, TopEnd de NCR ou encore Jaguar de Sybase, ...)
  - les middleware "orientés messages" : ce sont les MOM (comme MQ Series d'IBM, JMS de Sun, MSMQ de Microsoft).

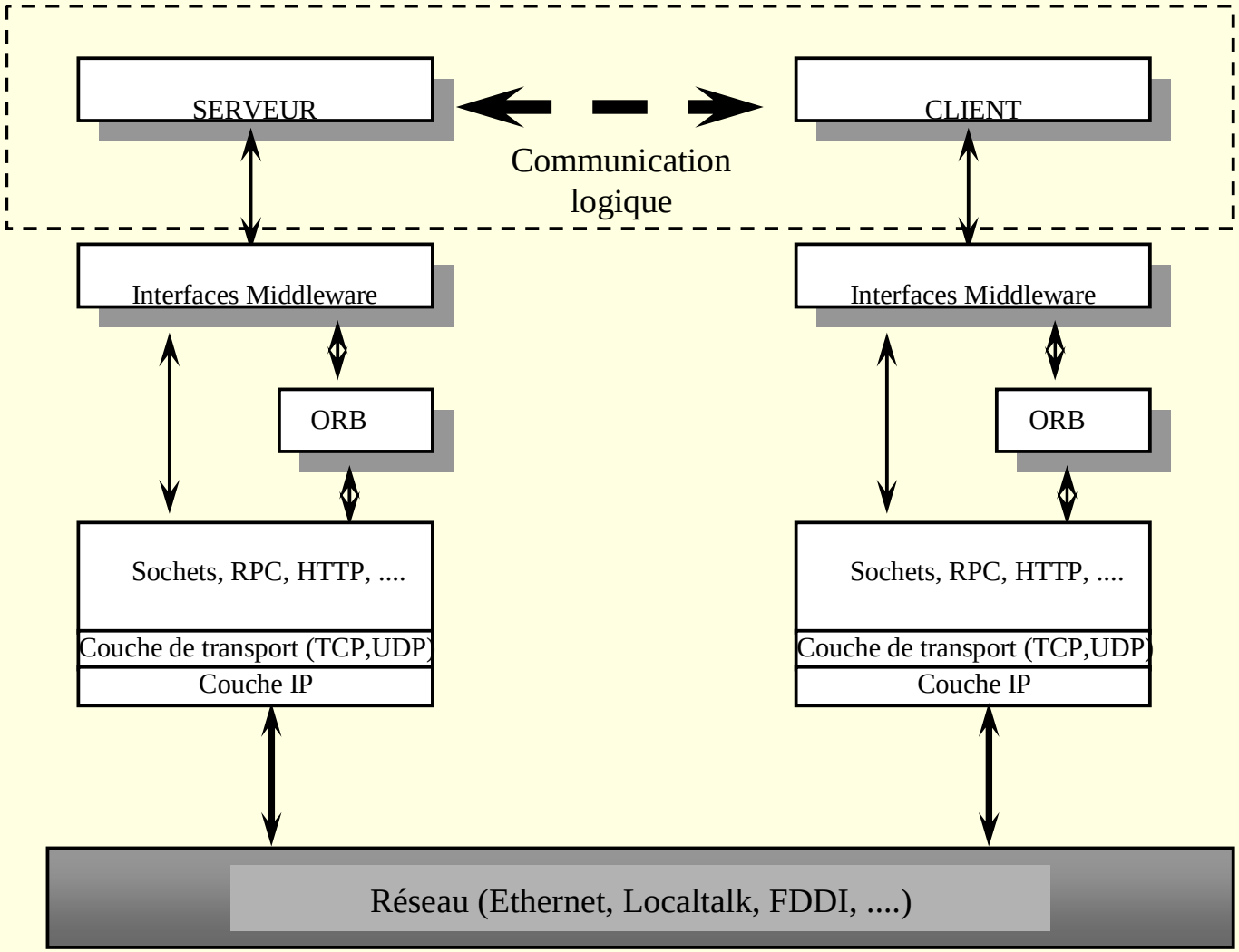
Le middleware (3/3)



La communication logique (1/2)

- L'objectif du choix d'une bonne architecture est d'abstraire au maximum les mécanismes d'échange des informations. On parle de "communication logique" entre le serveur et le client.
- Les couches bases sont encapsulées, cachées, rendues transparentes pour le programmeur.

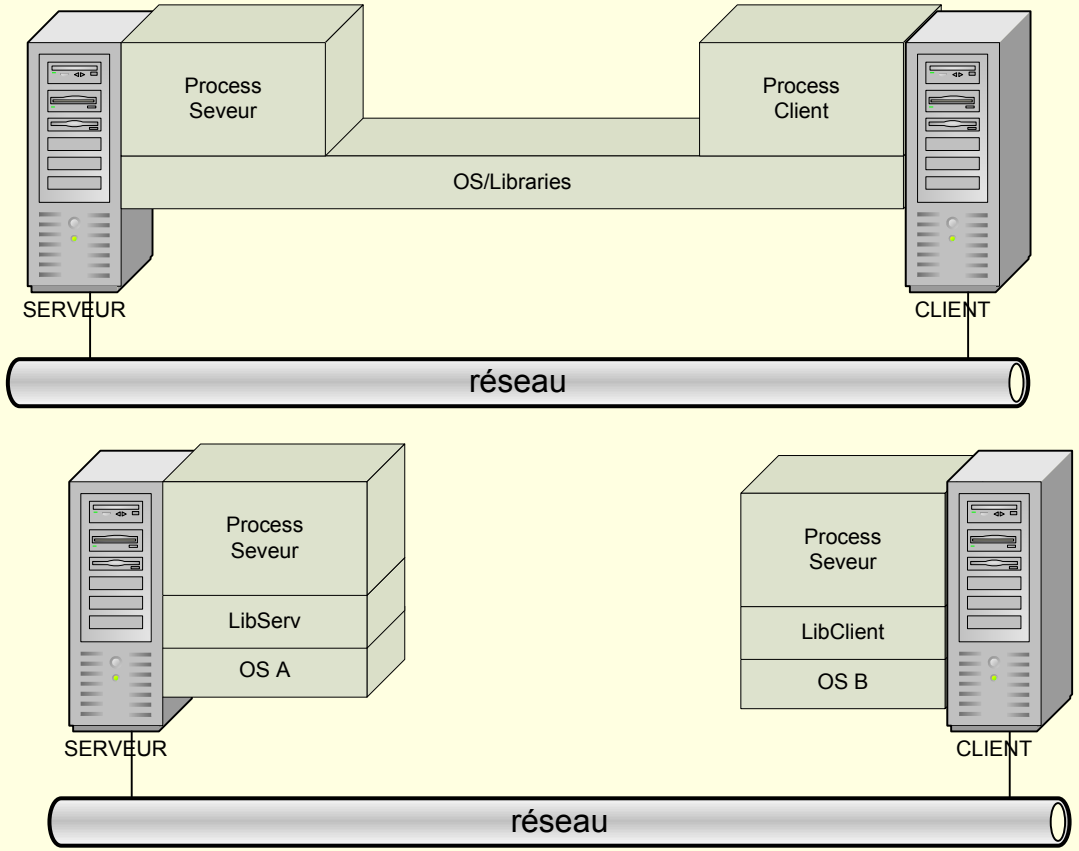
La communication logique (2/2)



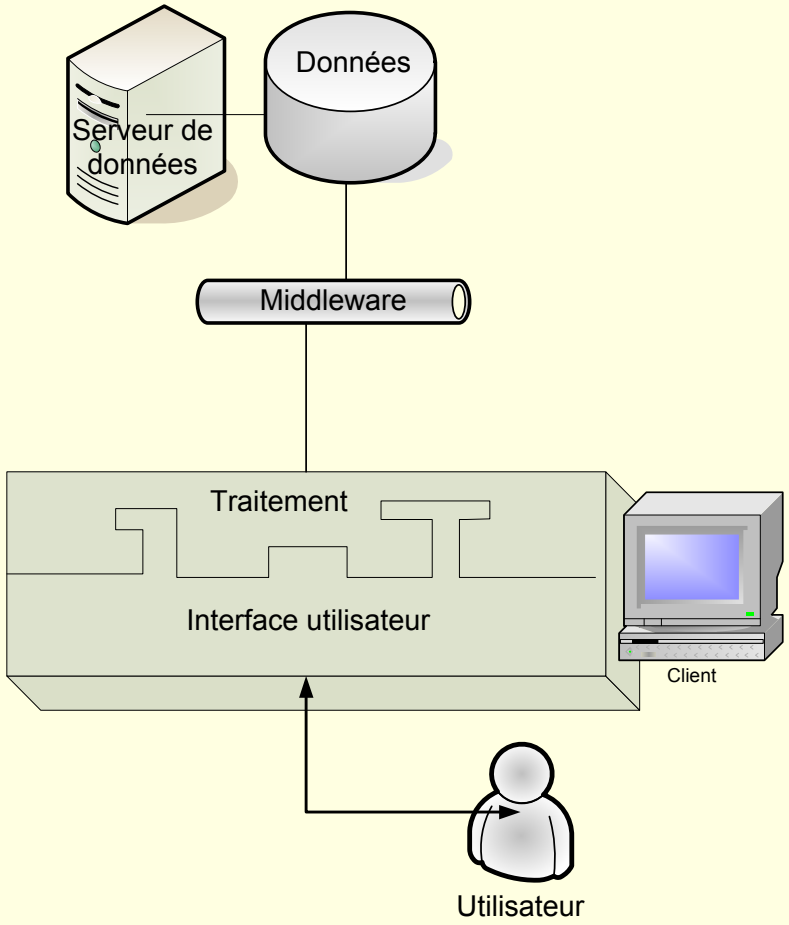
Les architectures client/serveur 2 niveaux (1/4)

- L'architecture 2 niveaux est l'architecture la plus couramment utilisée pour assurer la communication entre un serveur et un client.
- Dans ce cas, le client "discute" directement avec le serveur. Les moyens informatiques mis en œuvre pour réaliser le serveur et le client peuvent être les mêmes.

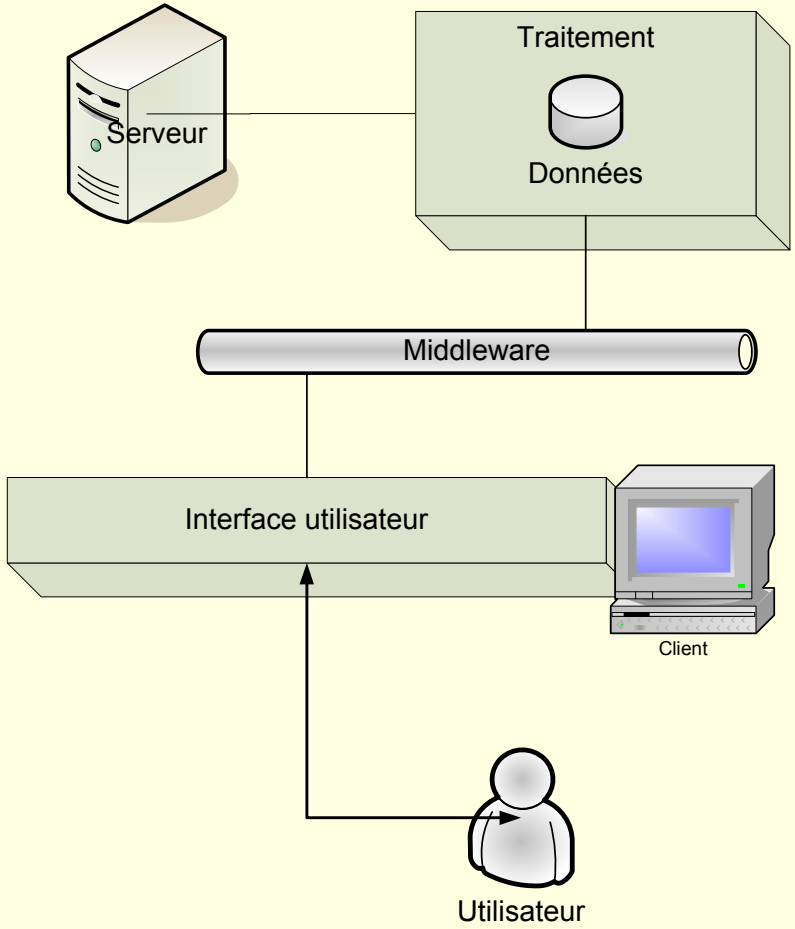
Les architectures client/serveur 2 niveaux (2/4)



Les architectures client/serveur 2 niveaux (3/4)



Architecture de premier type



Architecture de second type

Les architectures client/serveur 2 niveaux (4/4)

- Architecture du 1er type :
  - Cette architecture ne différencie pas l'interface utilisateur des traitements.
  - Les données sont gérées par un serveur de données, comme par exemple un serveur de bases de données ORACLE.
  - Les traitements de gestion des données sont liés à l'architecture de l'interface utilisateur. Les requêtes d'accès aux données sont lancées au serveur de données et le résultat est récupéré.
- Architecture de 2nd type :
  - Dans ce cas, on ne différencie pas les traitements des données. Les traitements sont totalement réalisés par le serveur. Le middleware assure la transmission des messages entre le serveur et le client.
  - Il y a bien séparation de l'architecture de l'interface utilisateur et des traitements mais ces deux parties restent liées par le middle-ware utilisé. Le code d'accès au serveur de données est dépendant du type de serveur utilisé (exemple de l'utilisation de requêtes ORACLE stockées dans le serveur mais demandées par le client sous une forme dépendant du serveur).
  - Dans ce cas, il est :
    - difficile de changer le middleware utilisé sans impacter fortement toutes les couches de logiciel
    - la réutilisation de ce type de logiciel reste une entreprise difficile.

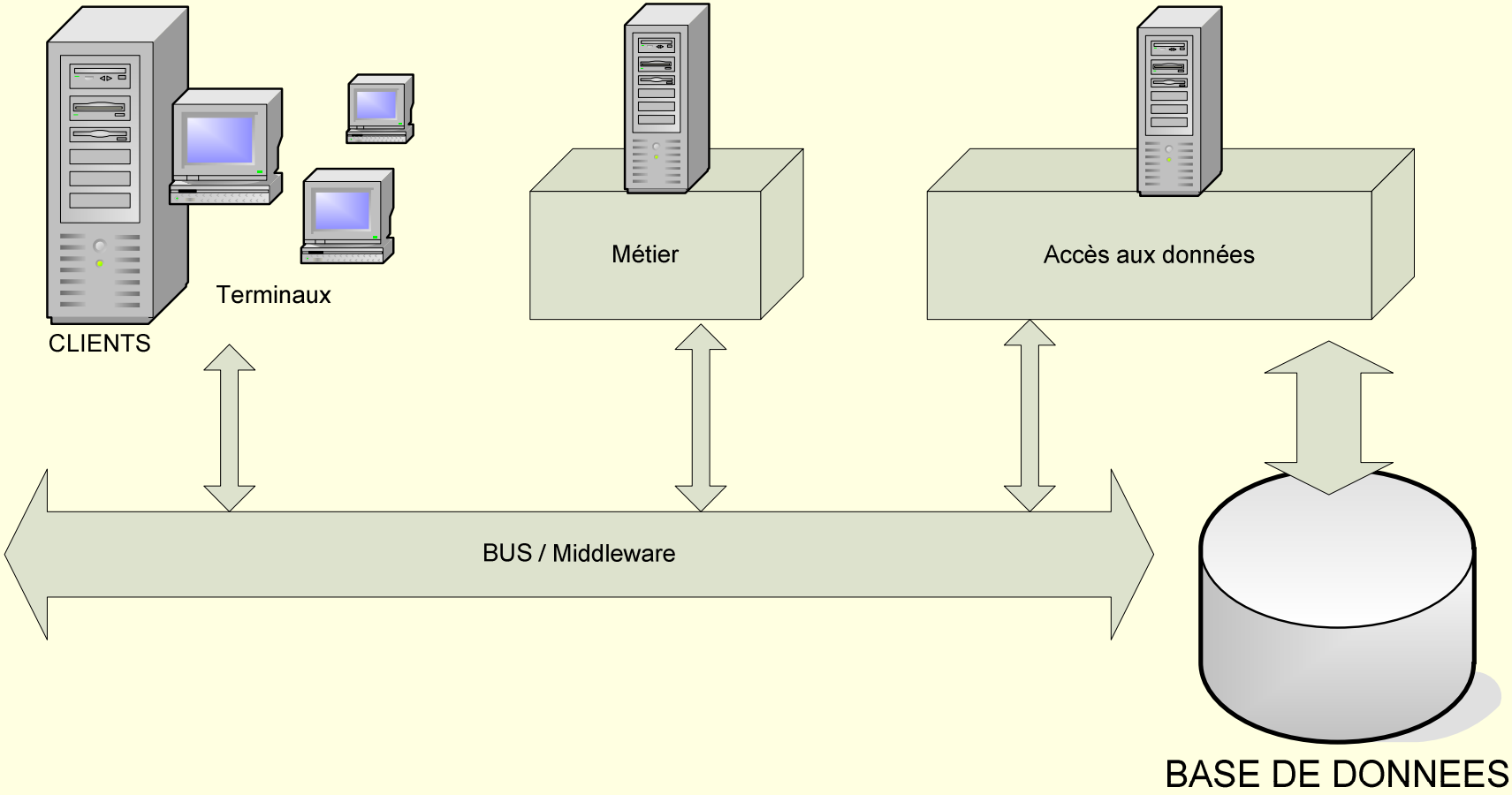
Les architectures 3-Tiers (1/2)

- ou architecture 3 niveaux (extension du modèle client/serveur)
- Ce type d'architecture est le plus courant des architectures multi-tiers
- L'architecture logique du système est divisée en trois niveaux :
  - couche présentation
    - IHM, client lourd ou léger, Web
  - couche métier
    - fonctionnel de l'application
    - en fonction des requêtes des utilisateurs, cette couche implémente la logique et décrit les opérations que l'application opère sur les données
  - couche accès aux données
    - abstraction de l'accès aux données pour la couche métier
    - persistance des données
    - données gérées de manière externe pour le système considéré
- Si la couche métier est découpée en plusieurs couches, on parle d'architecture N-tiers

Les architectures 3-Tiers (2/2)

- Cette architecture sépare le codage et la mis en œuvre entre l'interface utilisateur, les traitements et les données.
- Elle possède les avantages suivants :
  - interchangeabilité des différentes couches par le respect d'interfaces précises
  - autonomie des différentes couches face à une évolution toujours croissante
  - indépendance par rapport à la localisation physique
  - réutilisation facile de ses composants
  - bien adaptée aux applications **Intranet (et non internet)**
- La mise en œuvre de ce type d'architecture se fait notamment en utilisant une couche logicielle prédéfinie appelée le ORB (Object Request Broker).

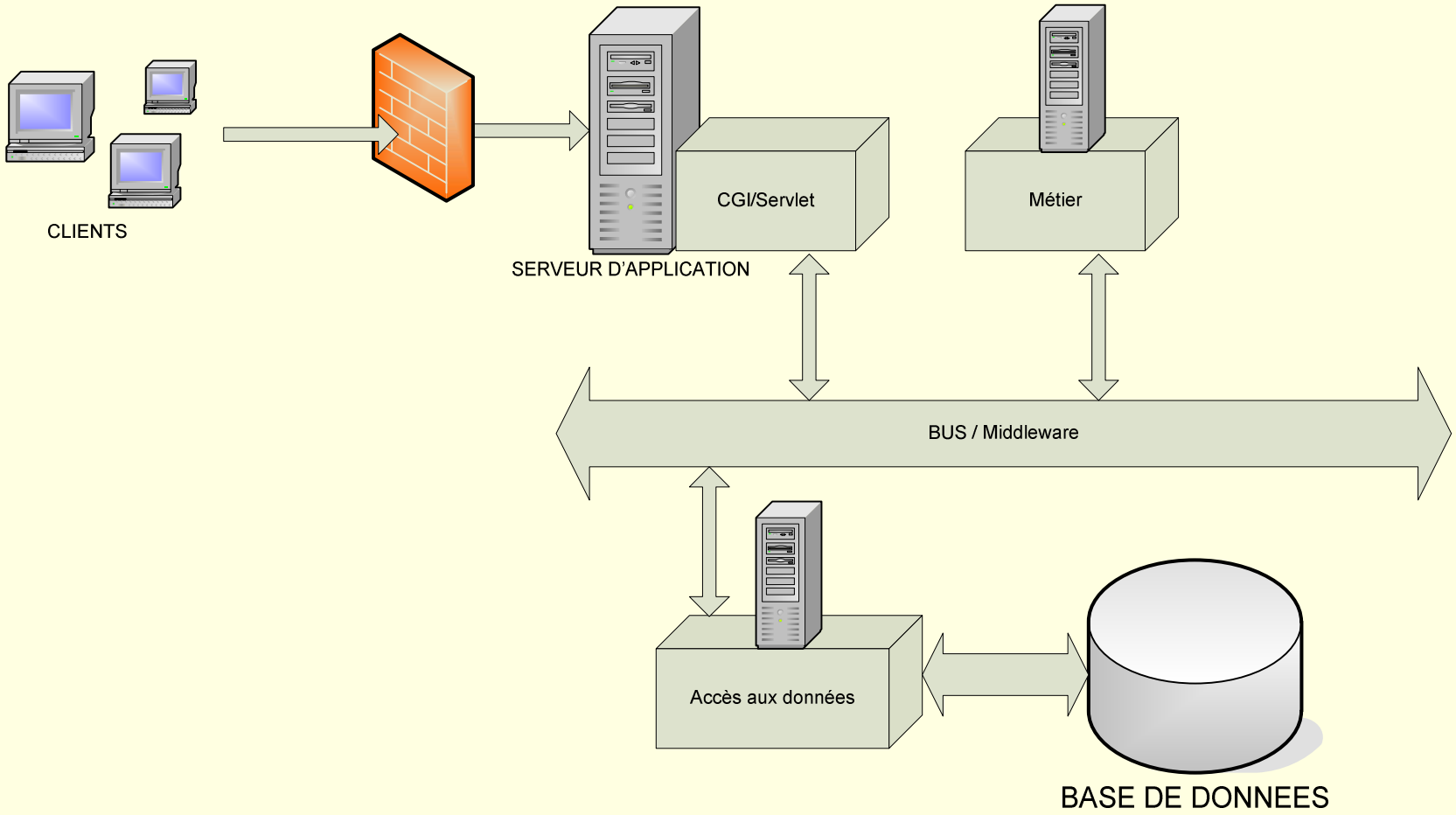
Les architectures 3-Tiers (schéma)



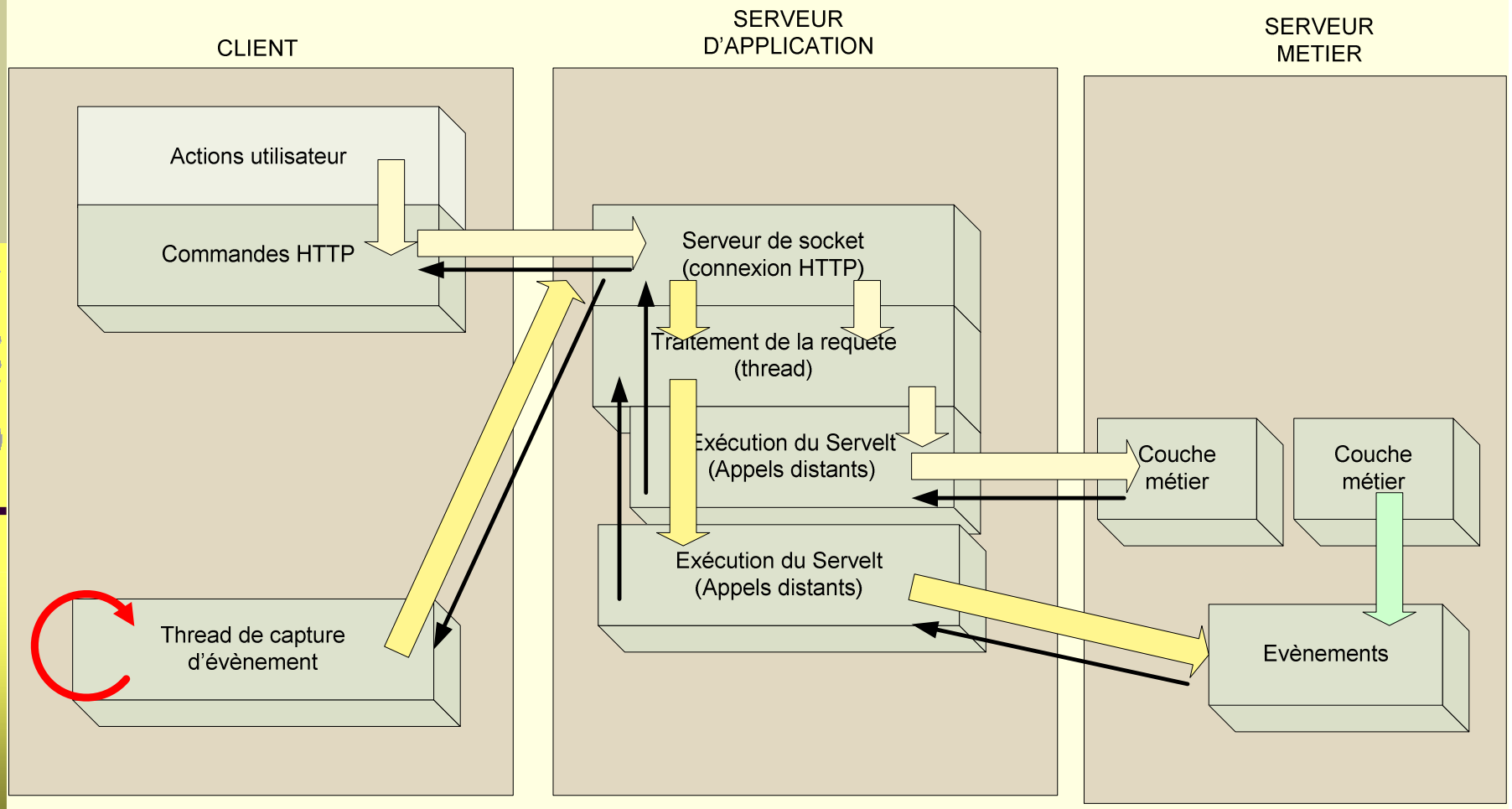
Une architecture 4-tiers très connue : le Web-service

- la couche de présentation constituée de client léger qui s'exécute dans un navigateur
- le serveur d'application (couche centrale des architectures N-tiers), services de base (authentification, sécurité, persistance, …)
  - JBoss, JOnAS, Tomcat, BEA Weblogic,
  - CORBA 3
- la couche métier
  - peut s'appuyer sur des services (EJB) du serveur d'application
- la couche accès aux données
  - la couche de persistance peut être un service du serveur d'application
- le serveur d'application intègre un middle-ware

L'architecture Web-service (schéma)



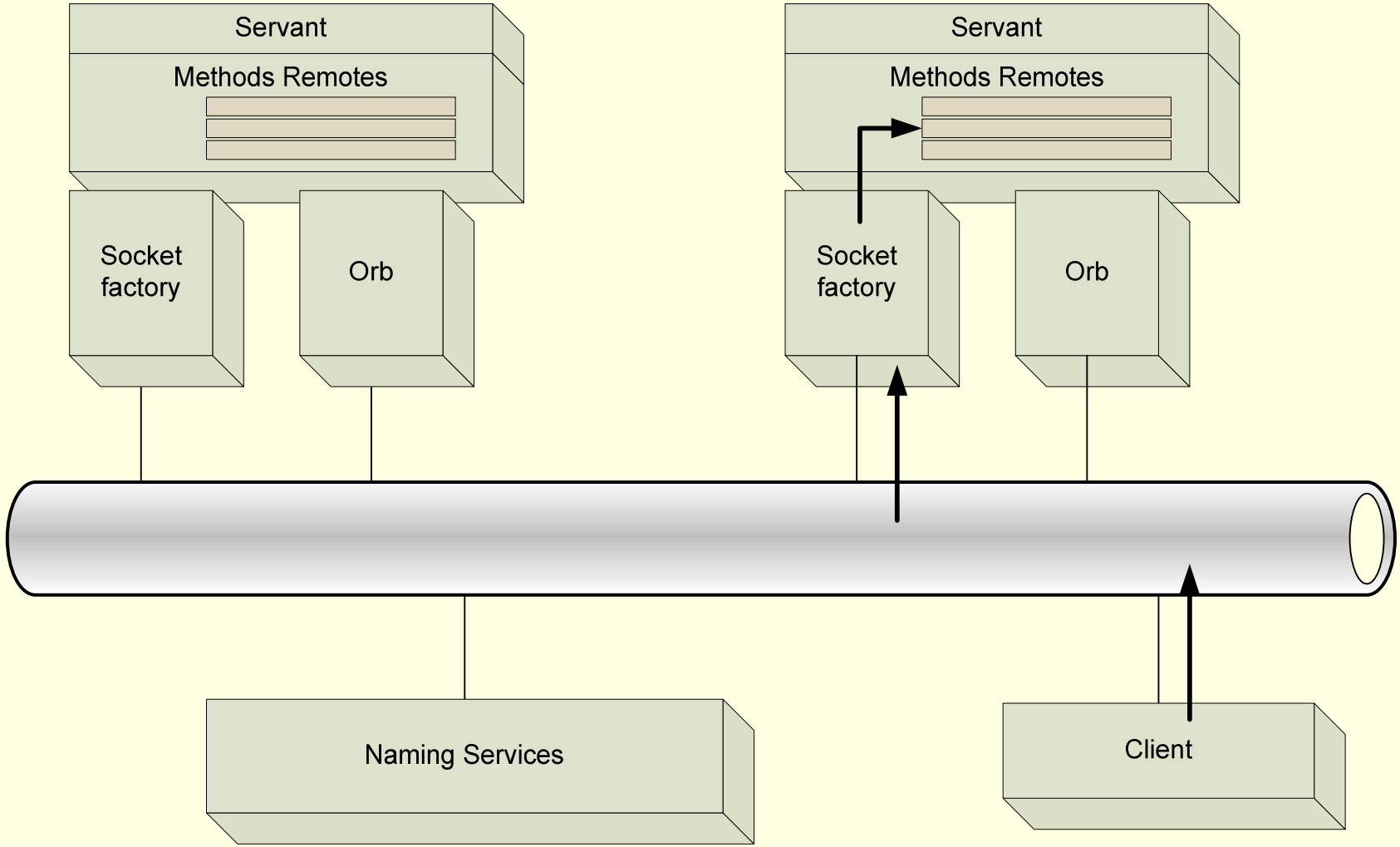
La dynamique de l'architecture Web-service



Les architectures distribuées

- architectures où les ressources du SI sont réparties sur le réseau
- répartition des données et des services
- répartition des calculs
- CORBA, RMI, service web XML, .NET Remoting, Windows Communication Foundation

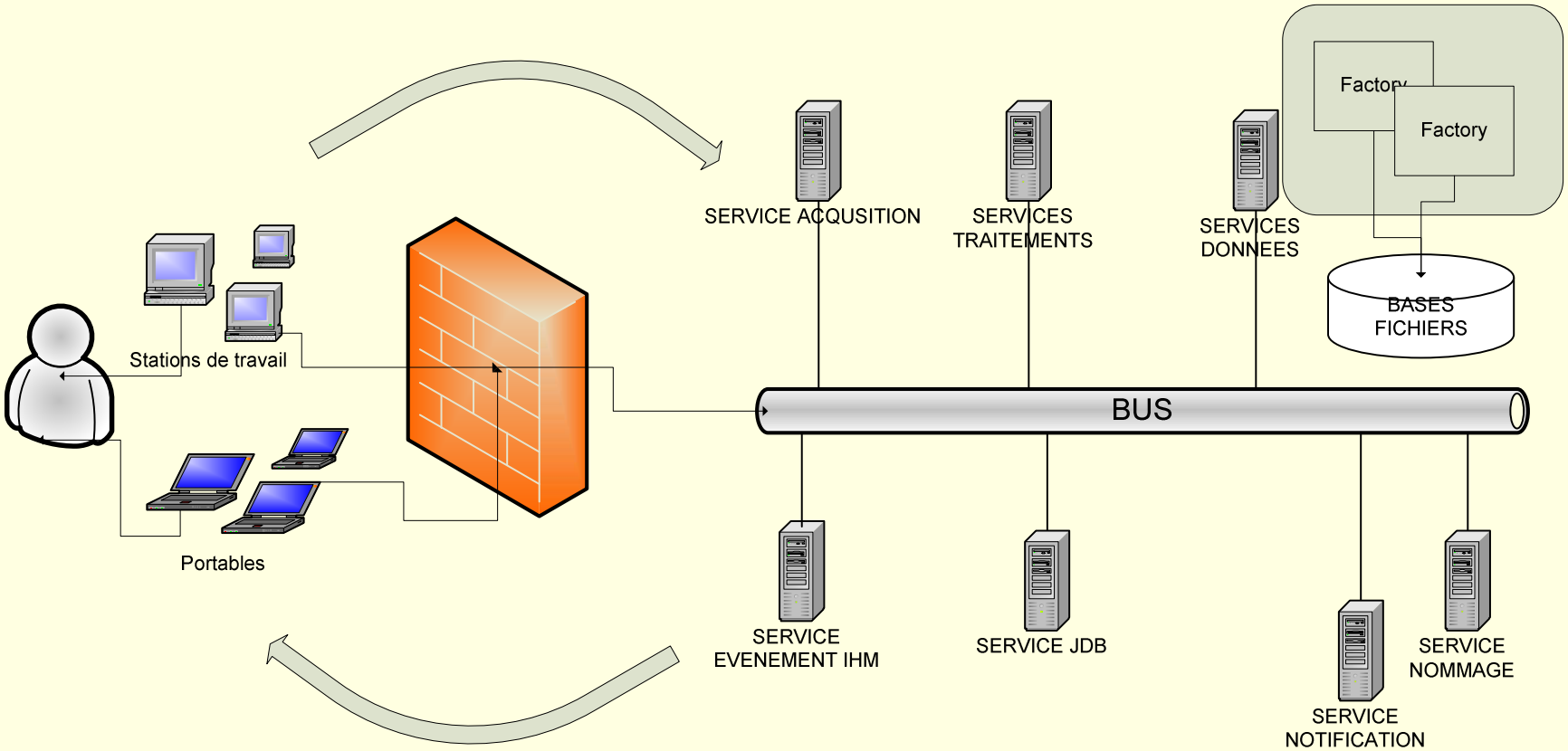
Les architectures distribuées (schéma)



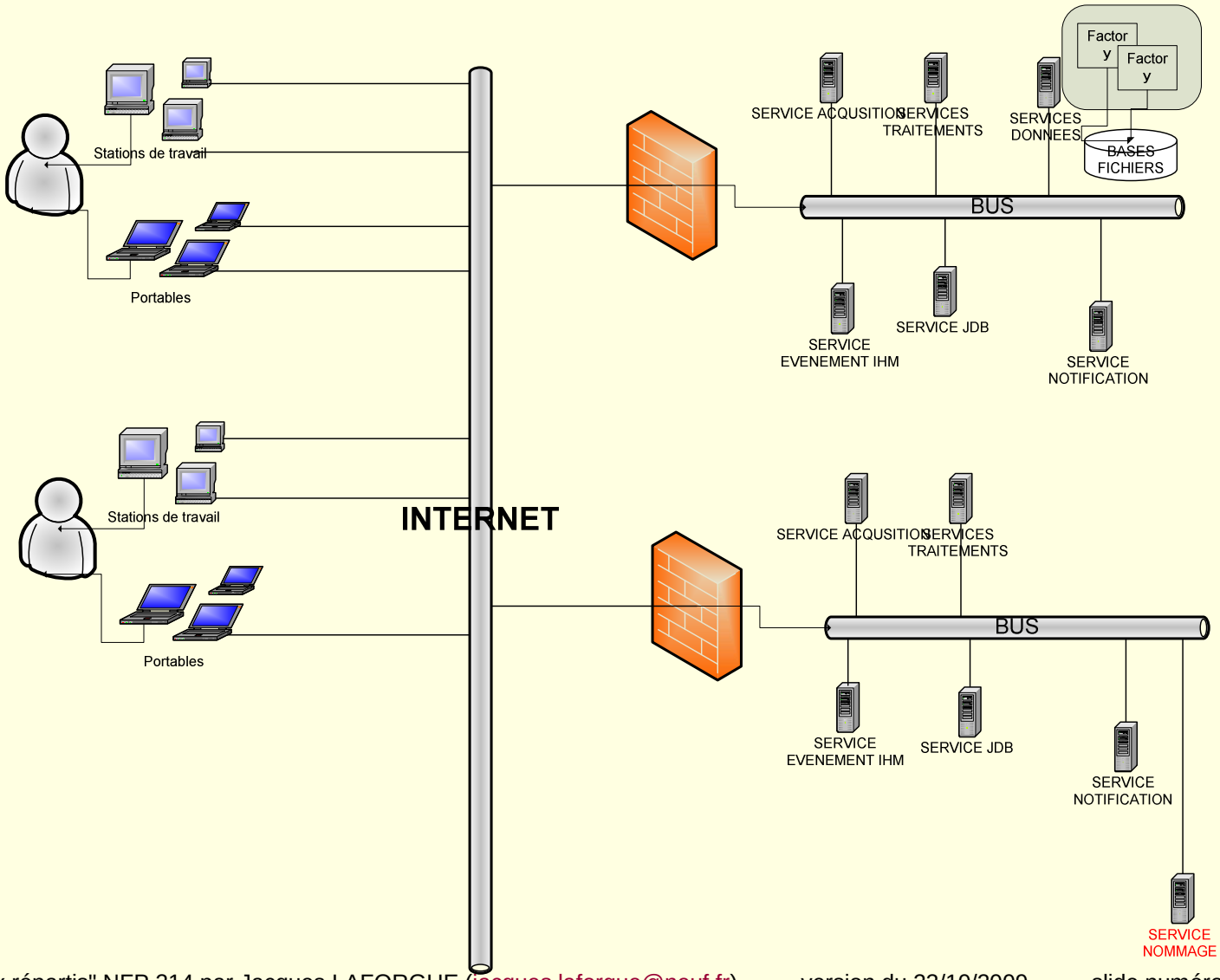
Les architectures orientées services (SOA) (1/2)

- intégration des services
- les architectures 4-tiers permettent l'accès à des services répartis sur un réseau local (derrière le frontal)
- le service est un nœud du modèle de répartition (ex: CORBA)
- interopérabilité des services entre eux
- conçues sur des plates-formes comme J2EE ou .NET
- le service peut être codé dans n'importe quelle langage et s'exécuter sur n'importe quelle plate-forme
- le service doit offrir un ensemble d'opérations dont les interfaces sont publiées sur le réseau
- utilisation d'un annuaire des services
- le bus de service
- quand le Web est le bus de service on parle de "Architecture Orientée Web" (SOW)

Les architectures orientées services (SOA) (2/2)



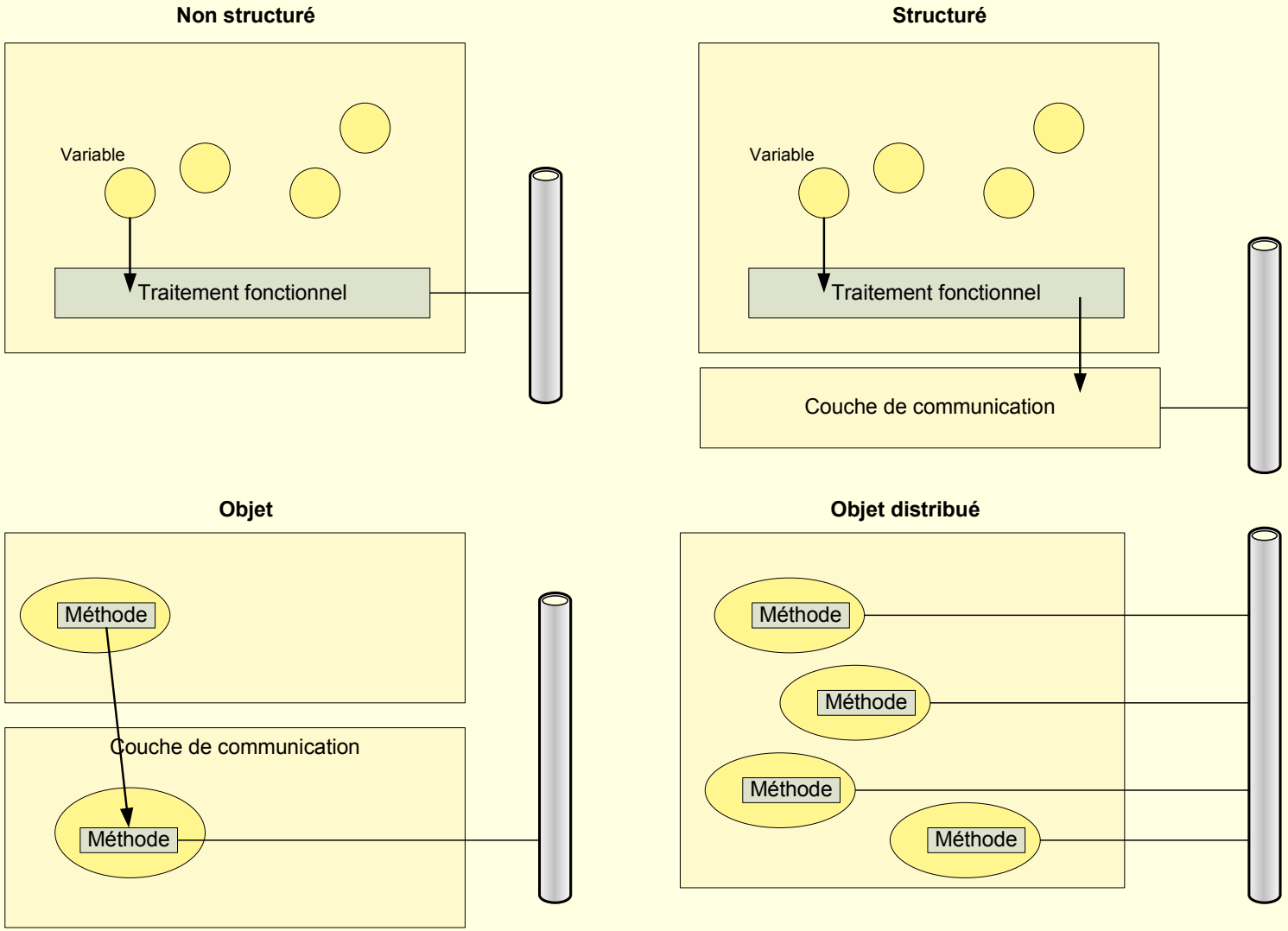
Les architectures orientées services (SOW)



Les modèles de programmation

- Les architectures des SI ont évoluées en fonction des modèles de programmation
- Le modèle non structuré
- Le modèle structuré
- Le modèle objet
- Le modèle service (ou objet distribué)

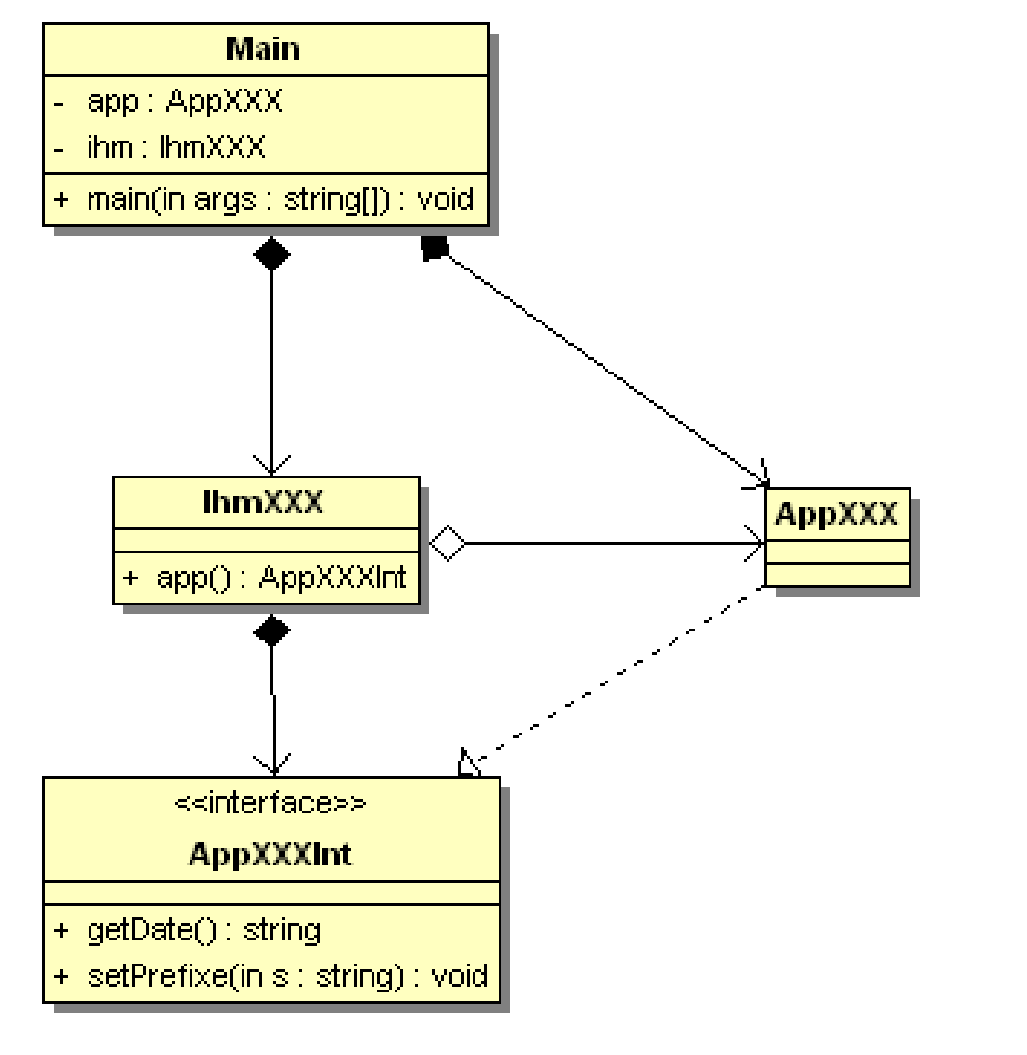
Les modèles de programmation (schéma)



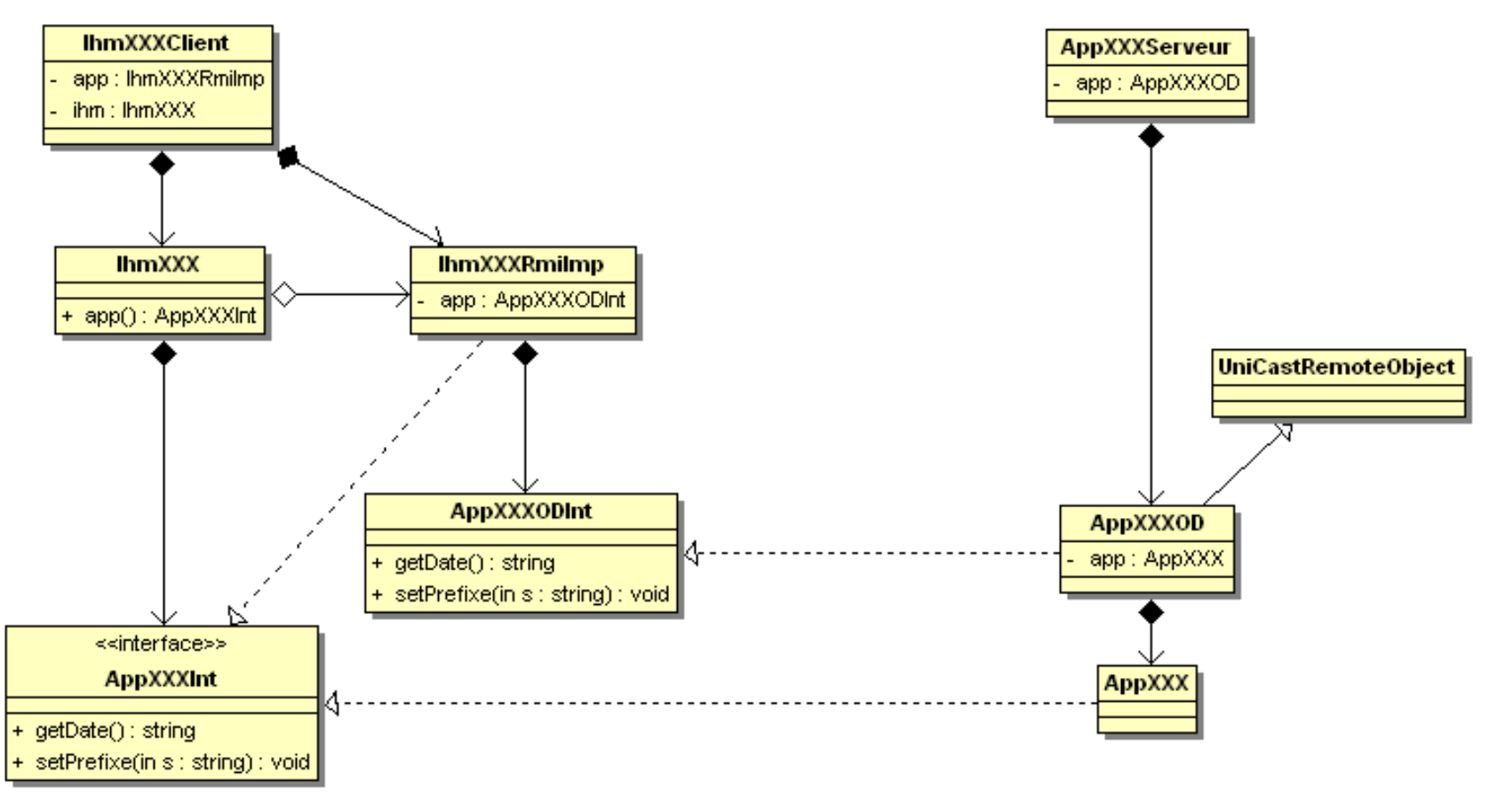
Le rôle des interfaces (Atelier 16)

- L'interface joue un rôle important dans l'architecture des SI
- L'interface permet d'utiliser un composant informatique indépendamment de son implémentation
- Elle permet l'interopérabilité des composants
- Elle permet d'abstraire les moyens de communication à mettre en place au sein du SI
- Ci-dessous des schémas UML de description de l'architecture d'une application IHM / Applicatif
  - cas 1 : programme unique
  - cas 2 : architecture client / serveur en RMI
    - l'applicatif se transforme en "serveur"
    - l'IHM utilise les services de l'applicatif par méthodes distantes
  - cas 3 : architecture web-services
    - l'IHM utilise un serveur HTTP qui à son tour utilise les services de l'applicatif par méthodes distantes
  - Dans les 3 cas, à aucun moment le code de l'IHM n'est modifié car utilise une interface

Architecture et interface : cas 1



Architecture et interface : cas 2



Architecture et interface : cas 3