Architecture Orientée Service (AOS)

« It is not the strongest of the species that survive, nor the most intelligent, but the ones most responsive to change …»

Charles Darwin

Service Oriented Architecture?

Chaque rôle s'approprie SOA différemment :

Dirigeants

Un ensemble de services que l'entreprise souhaite exposer à leurs clients et partenaires, ou d'autres parties de l'organisation

Analystes métier

Un style architectural basé sur un fournisseur, un demandeur et une description de service, et supporte les propriétés de modularité, encapsulation, découplage, réutilisation et composabilité

Architectes

Développeurs

Un modèle de programmation avec ses standards, paradigmes, outils et technologies associées

Un intergiciel offrant des fonctionnalités en terme d'assemblage, d'orchestration, de surveillance et de gestion des services

Intégrateurs

A quels besoins répond le SOA ?

Pourquoi les solutions actuelles sont insuffisantes ?

Problématique de l'intégration en entreprise

- Les entreprises doivent s'adapter en permanence et être de + en + réactives aux variations des marchés
  - fusions
  - acquisitions
  - scissions
  - diversification des offres commerciales
  - changement technologiques
  - …
- Ces opérations ont un impact sur le système d'information (SI) des entreprises
- L'intégration difficile des SI est un frein à ces changements

➢ ***C'est l'activité qui doit piloter la technologie et non l'inverse***

Problématique de l'intégration en entreprise

- La création d'applications dans l'entreprise est très souvent pilotée par des besoins à très court terme
  - ✓ Développement d'une application sous tel délai avec telles fonctionnalités

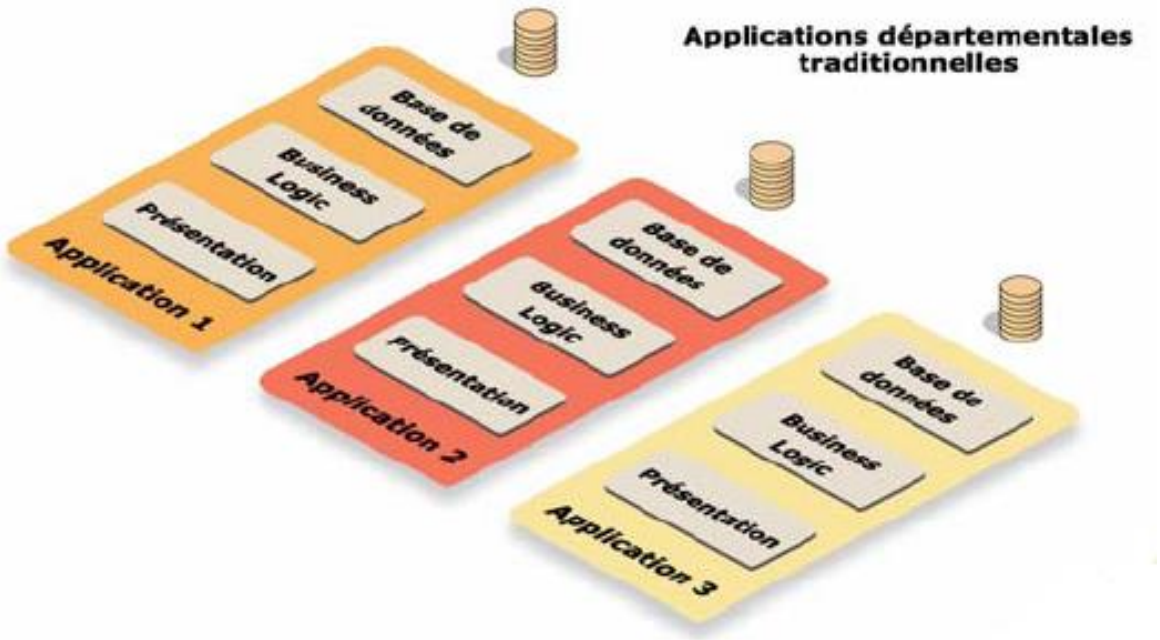
- Modélisation et développement dirigé par les choix/contraintes techniques
  - ✓ Pas de discussion entre maitrise d'ouvrage (MOA) et maitrise d'oeuvre (MOE)

- ***Décalage entre besoins métier et leur réalisation (constituants informatiques)***
- ***Pas de place pour la prise en compte de l'évolution des besoins fonctionnels au niveau de l'application***

Problématique de l'intégration en entreprise

- Le découpage présentation/traitement/base de données de l'architecture 3-tiers facilite le travail de la MOE mais favorise le cloisonnement en silos applicatifs indépendants

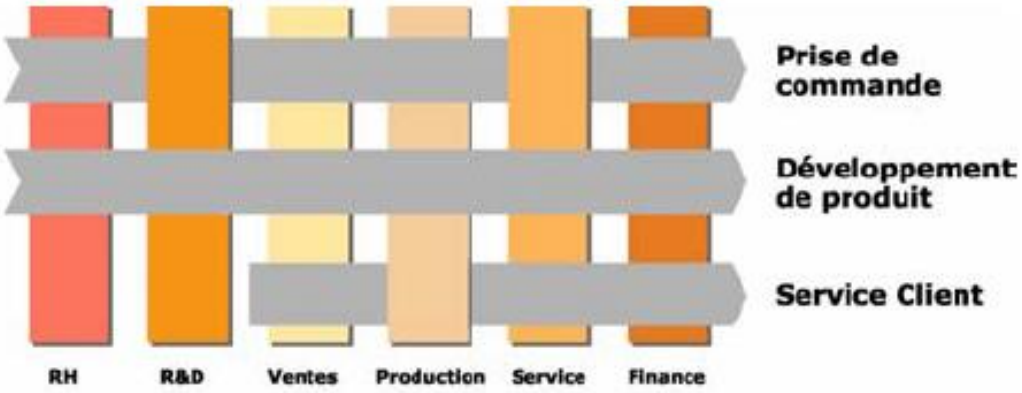
- Certaines fonctions sont redondantes : une version pour chaque application



➢ ***Pas de mutualisation des développements entre projets et peu de réutilisation possible***

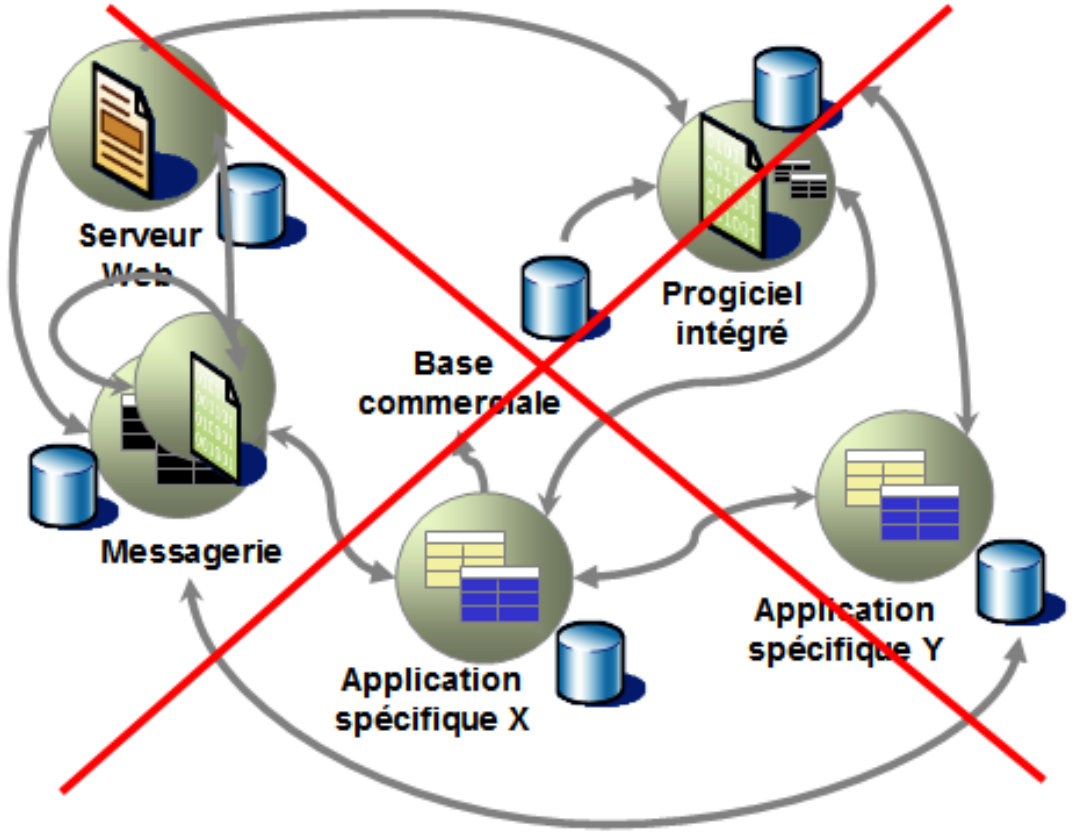
Problématique de l'intégration en entreprise

- Entreprises découpées en départements fonctionnels y compris le SI
- Processus métiers de + en + inter-départementaux
- Les processus franchissent les fontières de l'entreprise qui doit pouvoir prendre en compte les activités et processus des partenaires pour être reactive



➢ ***Coûts considérables dans la gestion des flux entre départements et dans l'intégration de leurs SI***

Hier : plat de spaghettis



- Développements coûteux
- Interconnexions redondantes (point à point)
- Grande complexité
- Maintenance difficile

Vers toujours plus d'abstraction

- Procédures
- Modules
- Modèles orientés objets
  - Packages
  - Encapsulation
- ...

Approche Objet

- L'évolution des langages de programmation a amené de nouveaux outils aidant à la conceptualisation des problèmes en informatique.

- La venue de l'orienté objet a facilité l'abstraction du problème à résoudre en fonction des données du problème lui même (par l'utilisation de classes et d'objets).

- La venue de la programmation objet a donné lieu à de nouvelles technologies de distribution des applications telle que :
  - RMI (Remote Method Invocation) : est une interface de programmation (API) pour le langage Java qui permet d'appeler des méthodes distantes

Approche Objet

- L'objectif principal des modèles objet était d'améliorer la modélisation d'une application, et d'optimiser la réutilisation du code produit.

- Cependant, l'intégration d'entités logicielles existantes peut s'avérer difficile si leur modèle d'exécution est incompatible avec le modèle imposé par le langage objet choisi pour le développement de nouvelles entités.

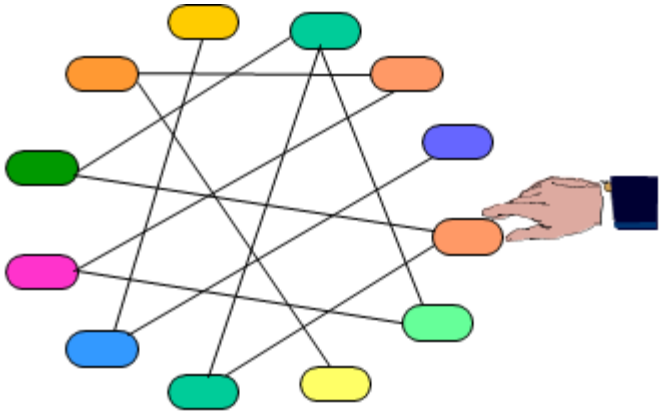
- Par ailleurs, les modèles et les langages objets ne sont pas, en général, adaptés à la description des schémas de coordination et de communication complexes.

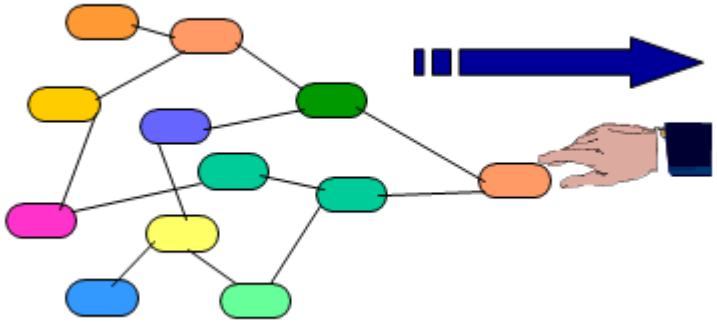
Limites de la programmation orientée Objet

- Structure et architecture de l'application peu visibles
- Interactions entre objets enfouies dans le code
- Évolution / modification difficile
- Recherche des bouts de code impliqués source d'erreur
- Gestion de la consistance d'un changement délicate

Objets et encapsulation

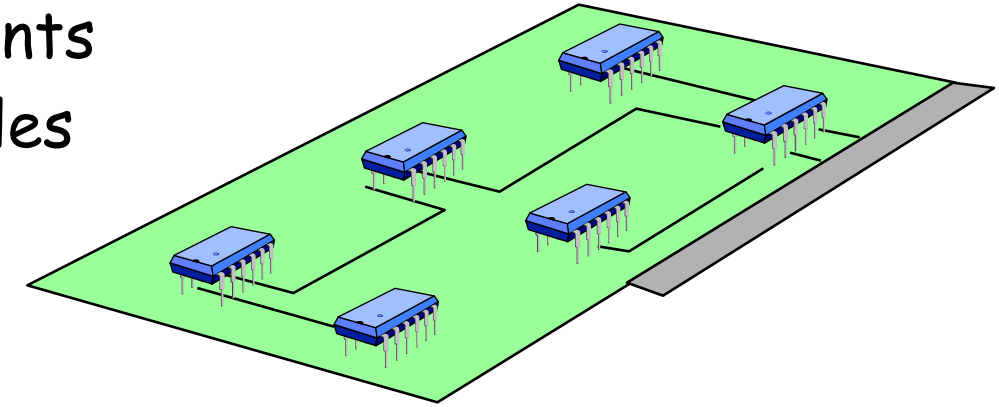
- **Granularité encore trop fine**
  - Mal adaptée à la programmation à grande échelle
- **Couplage fort**
  - Rend difficile la réutilisation
  - Accroît la complexité des Systèmes OO

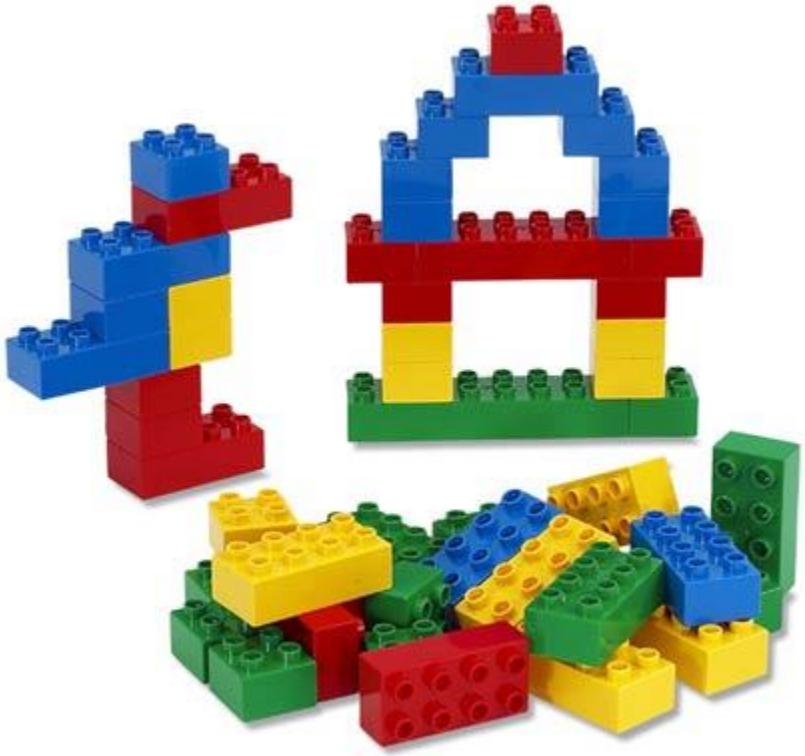


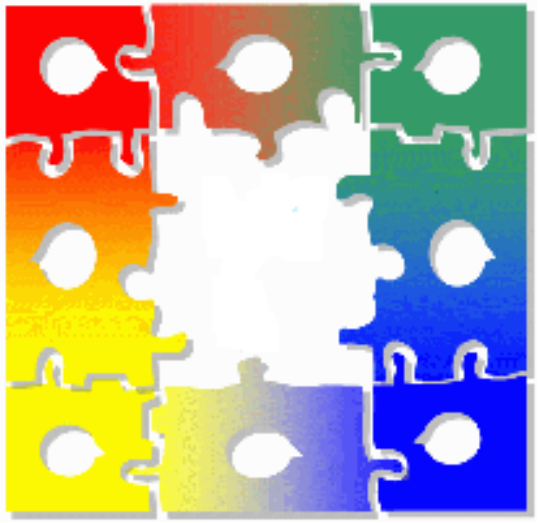


Encore plus de structuration avec les composants logiciels

Analogie avec les composants électroniques, légos, puzzles







Approche Composant

- Pour pallier les défauts de l'approche objet, l'approche composant est apparue.
- Cette approche est fondée sur des techniques et des langages de construction des applications qui intègrent d'une manière homogène des entités logicielles provenant de diverses sources.
- Un composant est une boite noire, communicant avec l'extérieur à travers une interface dédiée, permettant la gestion du déploiement, de la persistance, etc.

Un Composant : Qu'est-ce que c'est ?

Définition usuelle

- Une unité regroupant les fonctionnalités concernant une même idée
- Un module logiciel autonome pouvant être installé sur différentes plates-formes
  - qui exporte des attributs et des méthodes
  - qui peut être configuré (déploiement semi automatique)
  - capable de s'auto-décrire

Intérêt

Être des briques de base configurables pour permettre la construction d'une application par composition