Introduction à l'Architecture Orientée Service

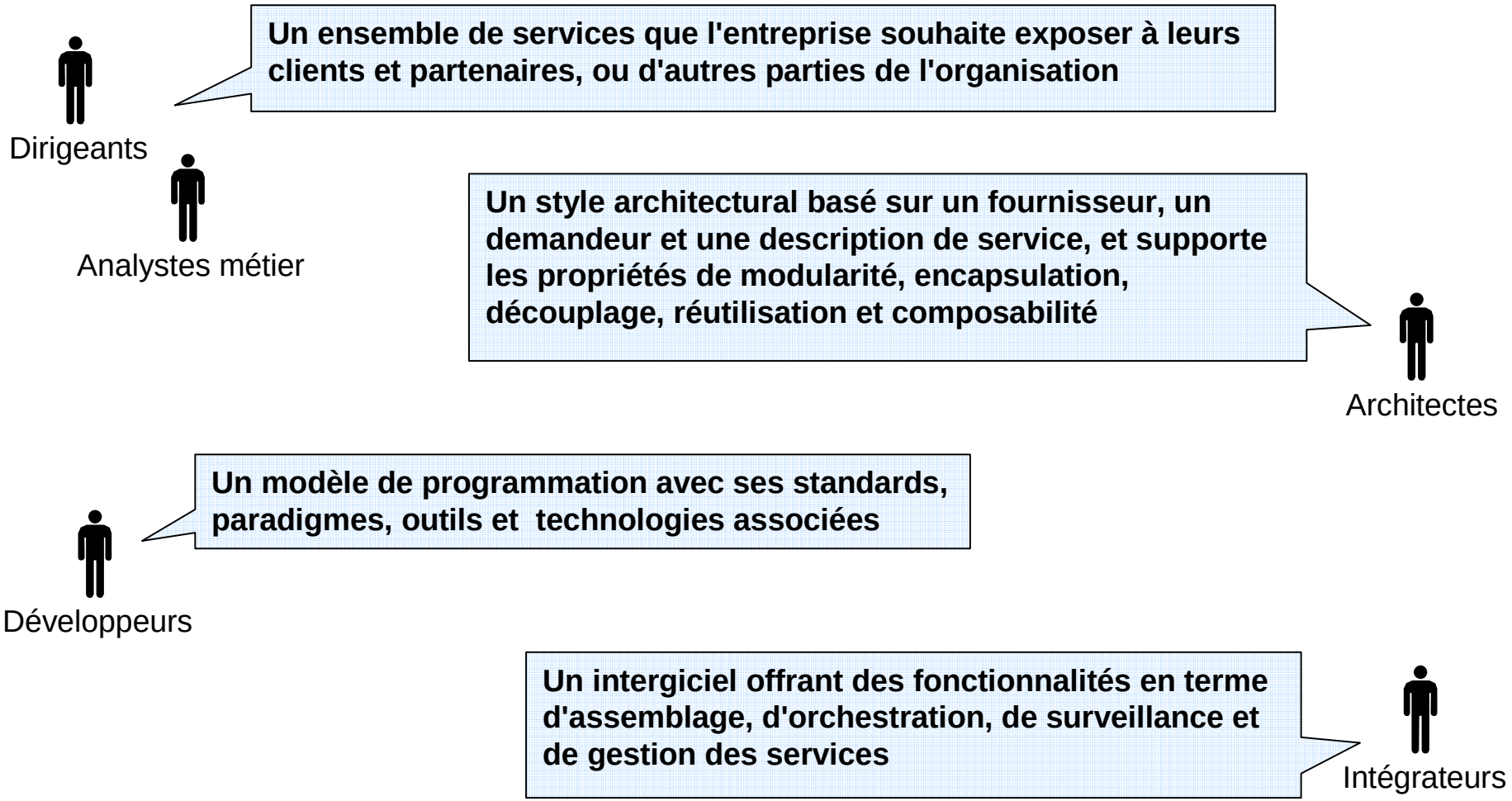
Modules SAR O2/SAR O3 – SI3

Revu par F. Baude, M2 MIAGE NTDP, 2008

(essentiellement simplification, raccourcissements, + quelques details)

Vous avez dit SOA?
Service Oriented Architecture

Chaque rôle s'approprie SOA différemment :



Plan du cours

- A quels besoins répond le SOA ?
- Pourquoi les solutions actuelles sont insuffisantes ?

- Quels sont les principes de base du SOA ?

- Quels sont les éléments clé d'une architecture orientée services ?

- Quel est le cycle de vie d'un service ?

- Quelles méthodes et outils permettent de mettre en oeuvre une architecture orientée services ?

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

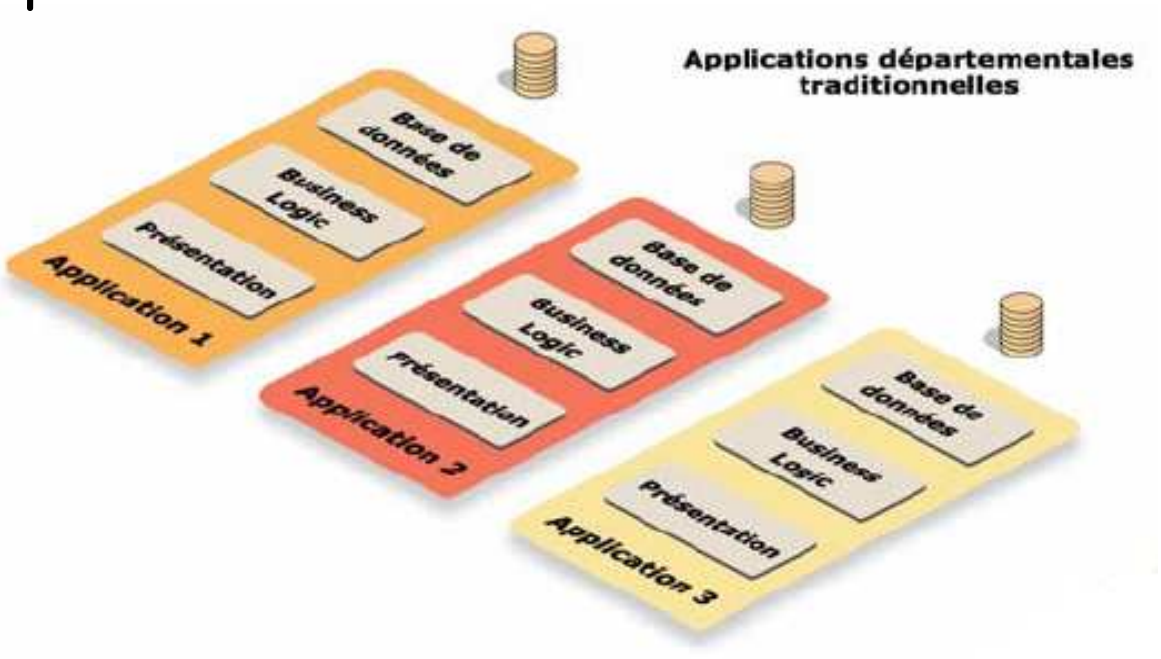
- La création d'applications dans l'entreprise est très souvent pilotée par des besoins à très court terme
  - ✓ Développement d'une application sous tel délai avec telles fonctionnalités

- 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

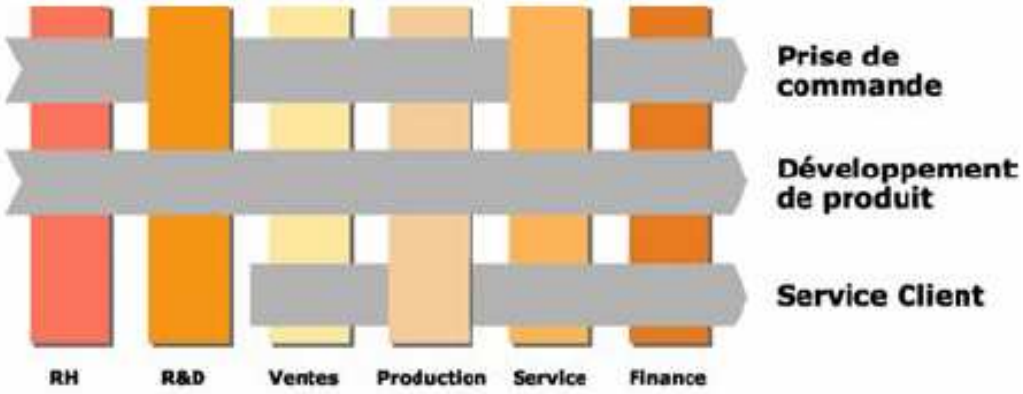
- Le découpage présentation/traitement/base de données de l'architecture 3-tiers facilite le travail de la MOE mais favorise le cloisonnement en silos applicatifs indépendants (blocs monolithiques)
- Certaines fonctions sont redondantes : une version pour chaque application



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

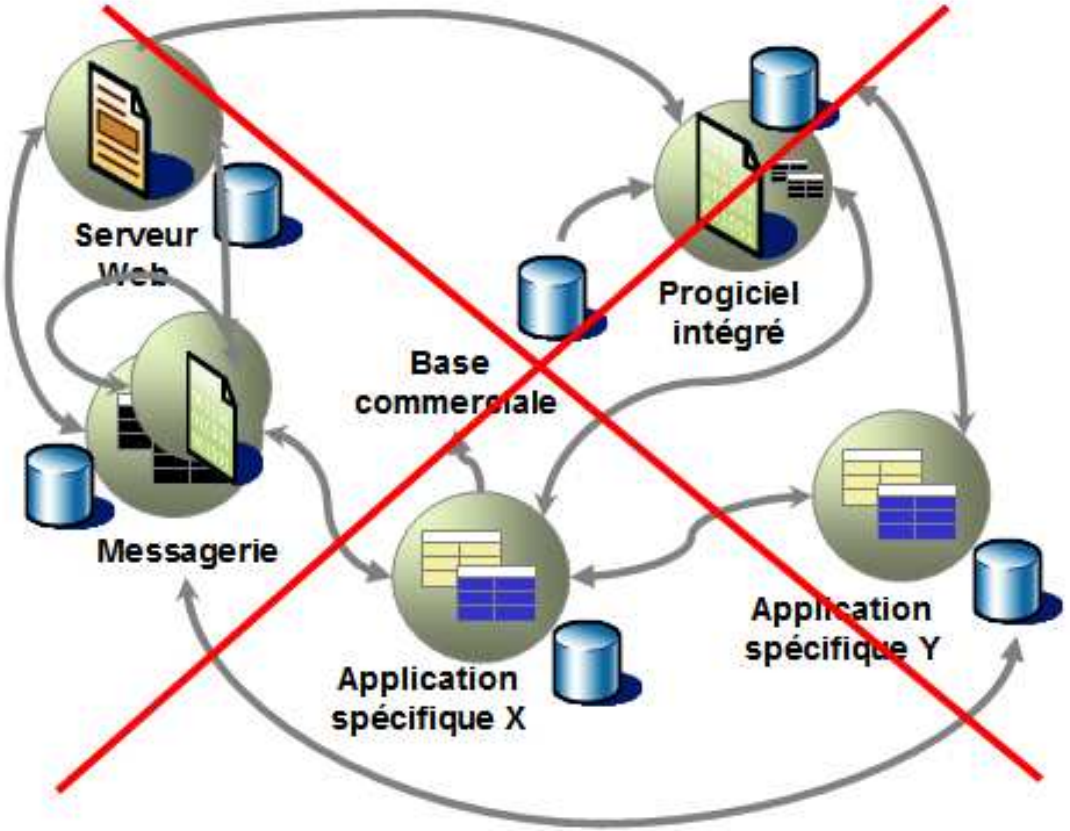
Problématique de l'intégration en entreprise

- 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
- Design pattern
- ...

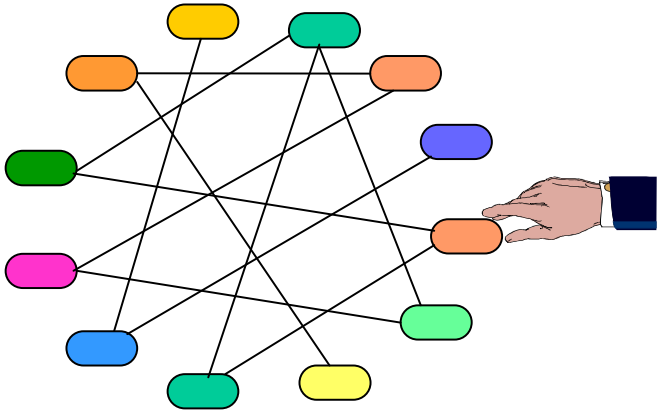
Limites de la programmation orientée Objet

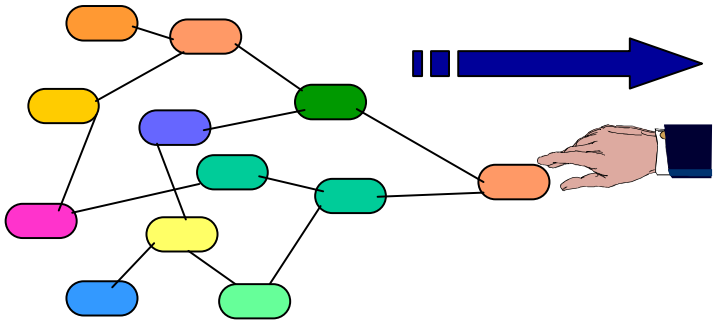
- Structure et architecture de l'application peu visibles
- Interactions entre objets enfouies dans le code
- Évolution / modification difficile
- Recherche des bouts de code impliqués source d'erreur
- Gestion de la consistance d'un changement délicate

Objets et encapsulation

- **Granularité encore trop fine**
  - Mal adaptée à la programmation à grande échelle

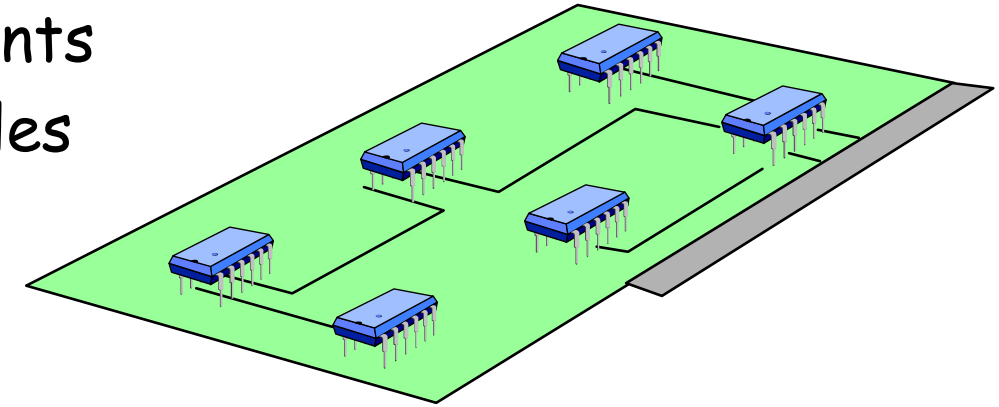
- **Couplage fort**
  - Rend difficile la réutilisation
  - Accroît la complexité des Systèmes OO

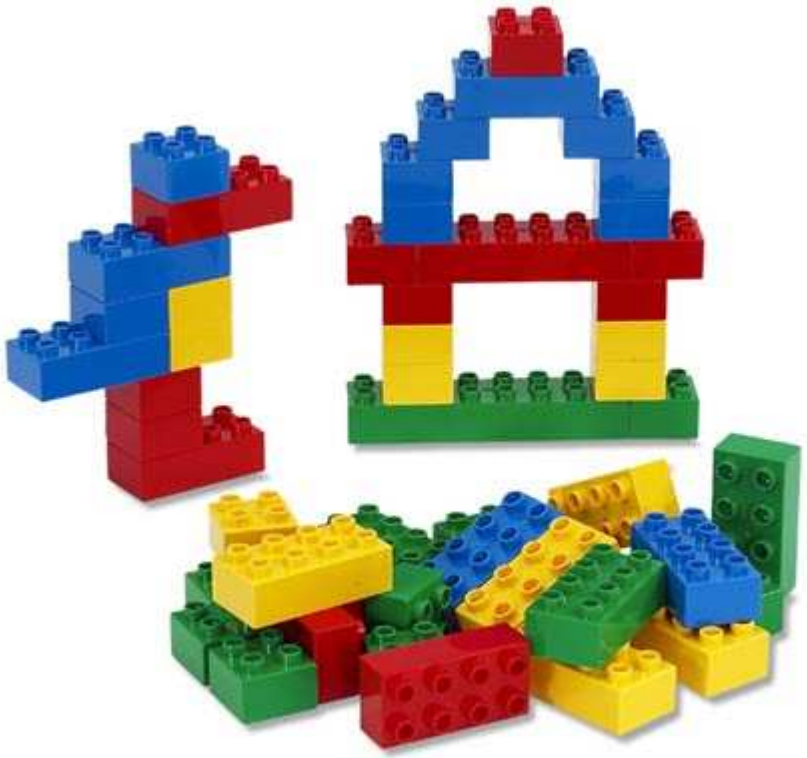


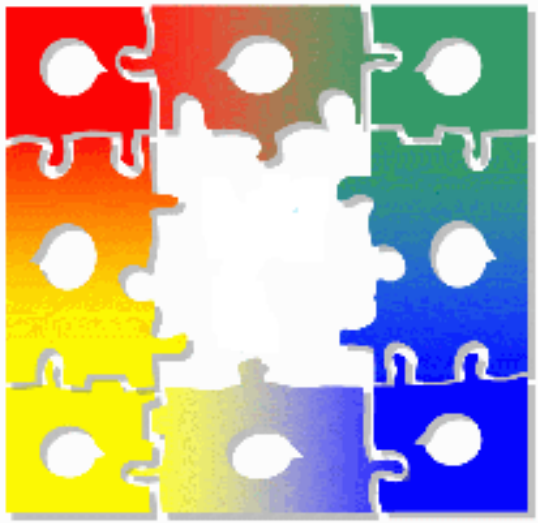


Encore plus de structuration avec les composants logiciels

Analogie avec les composants électroniques, legos, puzzles







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

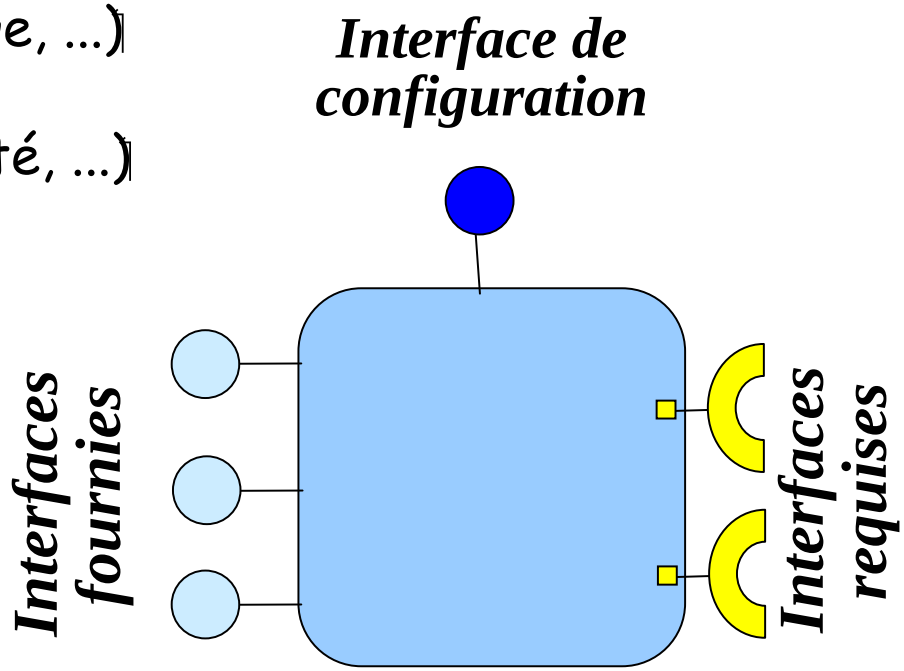
Structure d'un composant

Interactions avec un composant

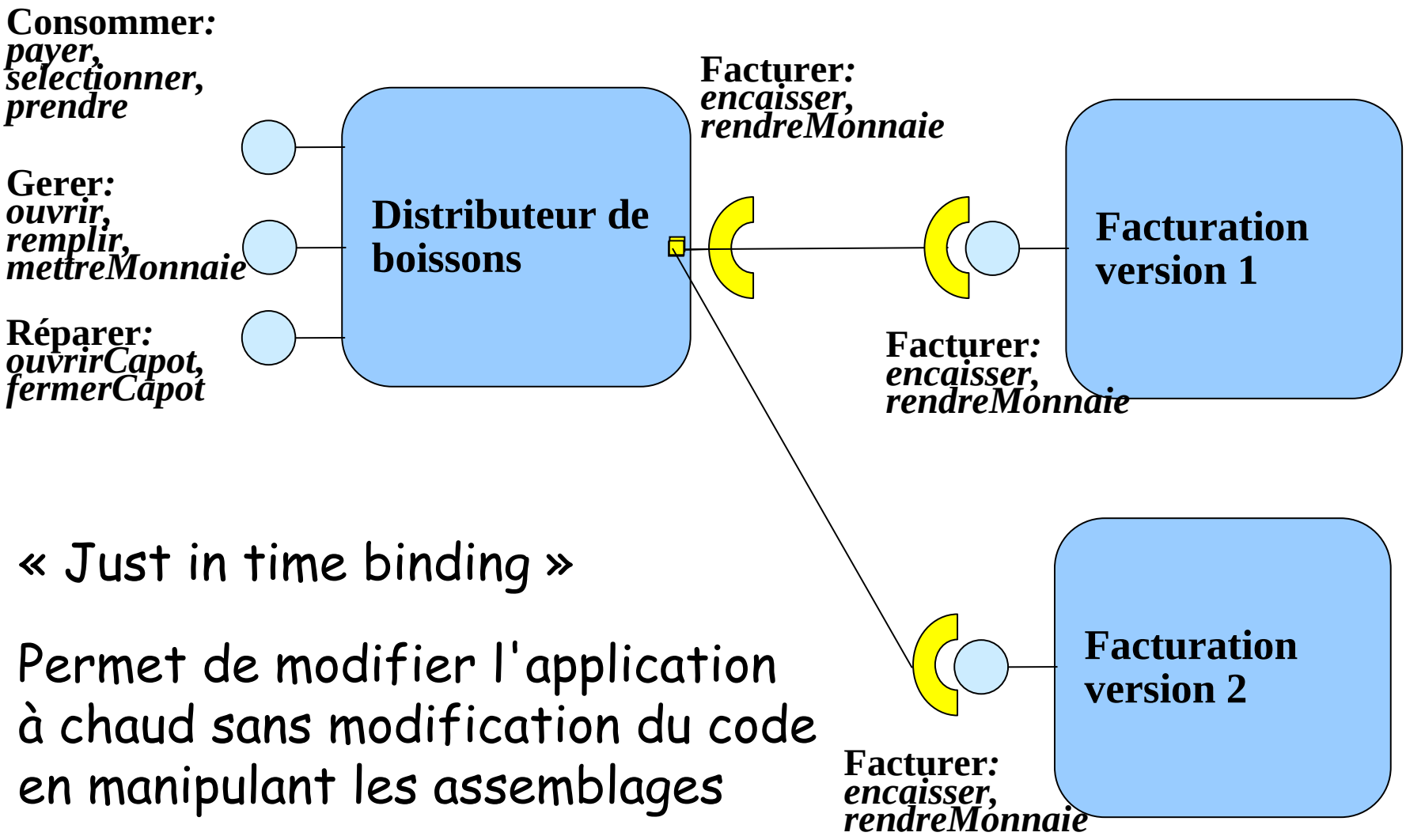
- ce qui est fourni par le composant
- ce qui est utilisé par le composant
- modes de communication

Configuration du composant

- propriétés (attributs publics)
- connexions
- cycle de vie (arret, redemarrage, ...)
- contraintes techniques (transaction, persistance, sécurité, ...)
- ...



Re-configuration dynamique



« Just in time binding »

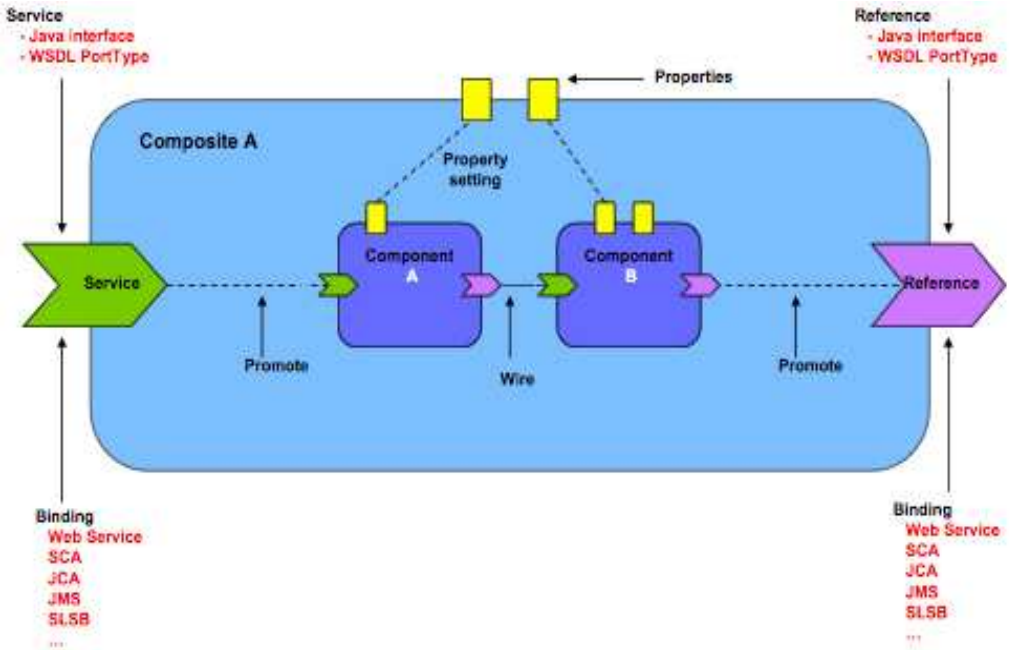
Permet de modifier l'application à chaud sans modification du code en manipulant les assemblages

Les composants dans la nature

- La modélisation des composants logiciels est intégrée à UML 2.0
- Spécification :
  - Composants CORBA (CCM)
  - Spring (JEE beans for Web apps)
  - Fractal (Etendu pour le réparti, voir GridComponentModel – Equipe I3S/INRIA OASIS)
  - **SCA (Service component Architecture) => utilisé pour SOA** (voir OSOA Tuscany, HydraSCA, IBMWebSphere pack for SOA, etc)
- Plates-formes d'éxecution
  - OpenCCM (GridCCM, Equipe PARIS IRISA Rennes)
  - Julia (Fractal), ProActive (GCM)
  - Sofa (Fractal)

Convergence Composants / Services

- Exposer les interfaces offertes par les composants selon une technologie au choix; Par exemple
  - Services web, avec binding SOAP
  - Interface Java avec binding RMI ou JMS
- Principe suivi par la norme SCA : Service Component Architecture
  - Notion de Composite Service



Convergence Composants / Services : Exemple

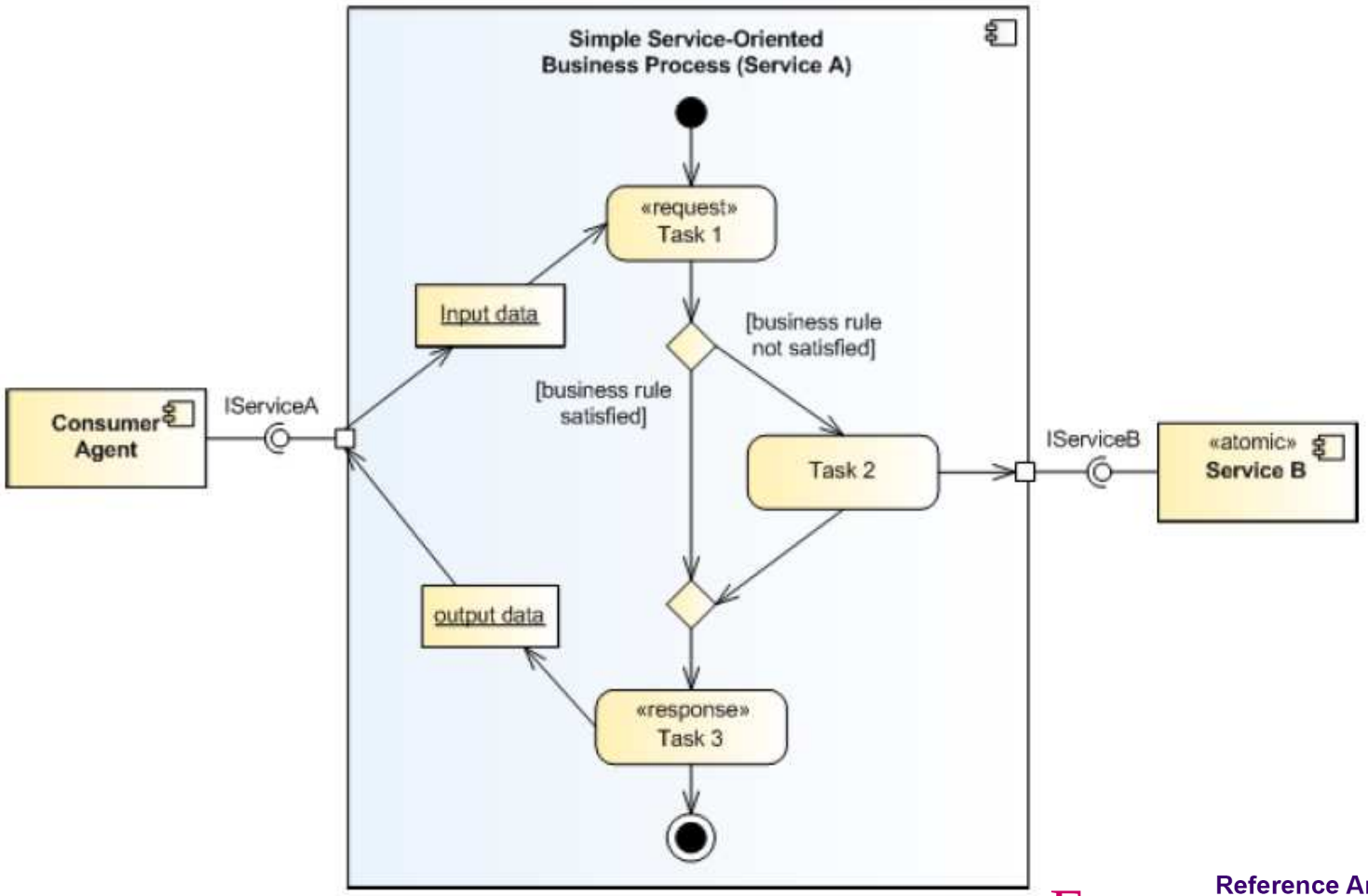


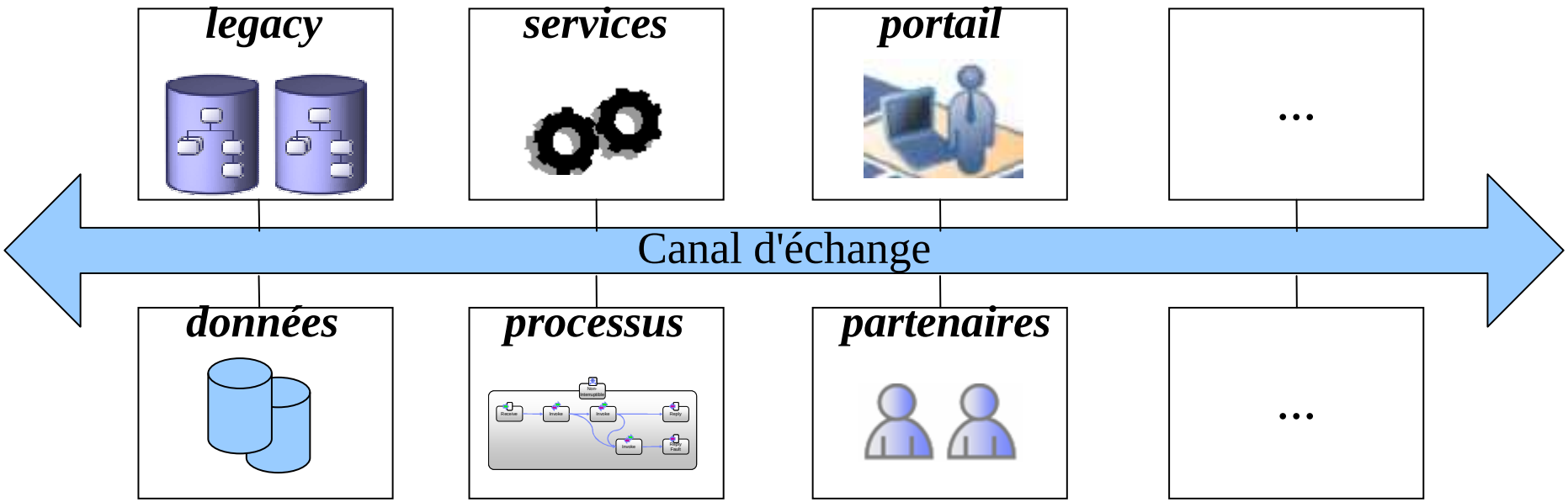
Figure 43 Abstract example of orchestration of service-oriented business process.

From : **Reference Architecture for Service Oriented Architecture Version 1.0**
Public Review Draft 1
23 April 2008

Demain : Architecture urbanisée

- L'**urbanisation** informatique définit l'organisation d'un SI à l'image d'une ville
  - découper le SI en modules autonomes (zone, quartier, îlot, bloc)
  - localiser les zones d'échange d'informations (routes, ponts, tunels) qui permettent de découpler les différents modules

- Objectif : faire évoluer le SI au même rythme que la stratégie et l'organisation des métiers de l'entreprise



Quels sont les principes de base du SOA ?

Principes fondamentaux de l'architecture SOA

Il n'existe pas une recette pour garantir le succès de la mise en place d'une SOA mais des principes à respecter :

- Discussion entre métier et IT
- Utilisation des use case métier

- Utilisation de standards
- Pas de remise en cause de l'existant lors d'évolutions technologiques

- Découplage entre fournisseur et consommateur de services
- Indépendance des ressources vis à vis de ceux qui les utilisent

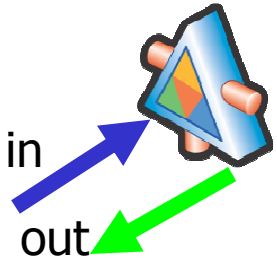
Qu'est ce qu'un Service (au sens SOA) ?

- Partage les caractéristiques suivantes d'un objet
  - Modulaire (ensemble de fonctionnalités qui font sens)
- Partage les caractéristiques suivantes d'un composant
  - Boite noire (séparation interface/implémentation)
  - Indépendant de la localisation
  - Neutralité vis-à-vis des protocoles de transport
- Correspond à un périmètre fonctionnel que l'on souhaite exposer à des consommateurs
- Est faiblement couplé (indépendant des autres services)
- Expose un petit nombre d'opérations offrant un traitement de bout en bout
- Sans état

4 propriétés du service à retenir

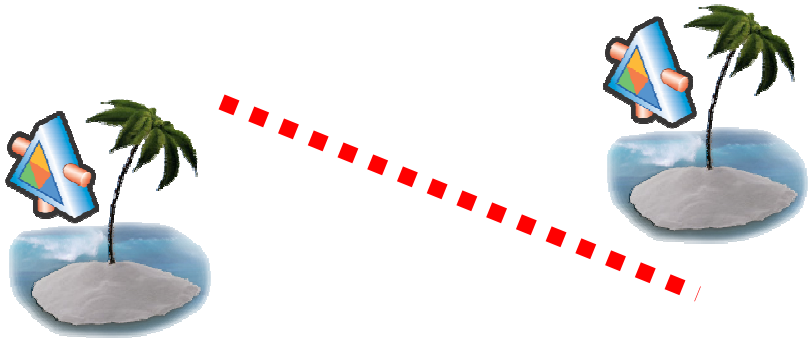
- Un Service est **Autonome** et sans **état** (en général, c.ex WSRF)

- Un Service expose un **Contrat**

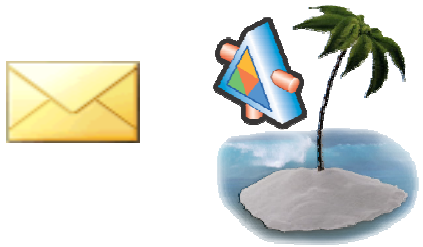


Conditions Générales de Vente
Règlement Intérieur
Vos droits/Vos devoirs

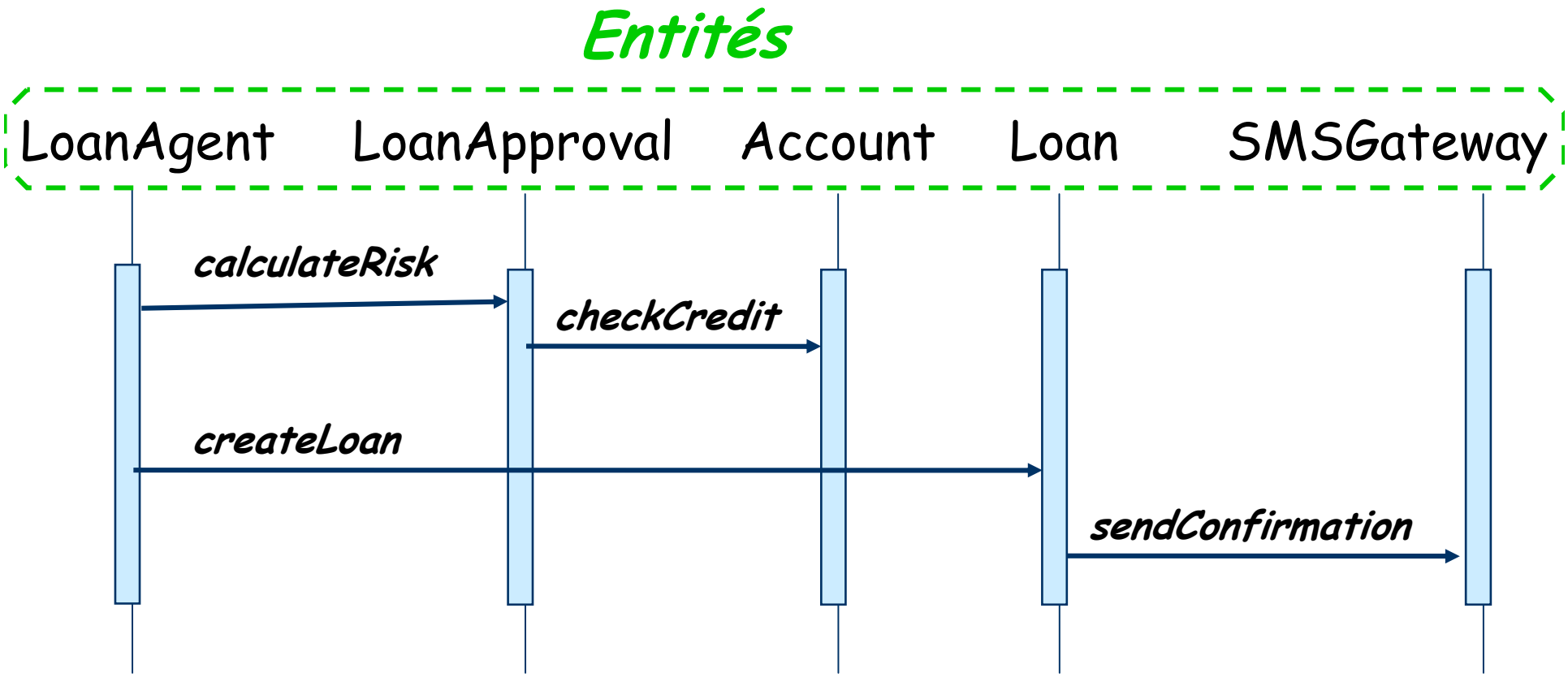
- Les Frontières entre services sont **Explicites**



- Les services communiquent par messages

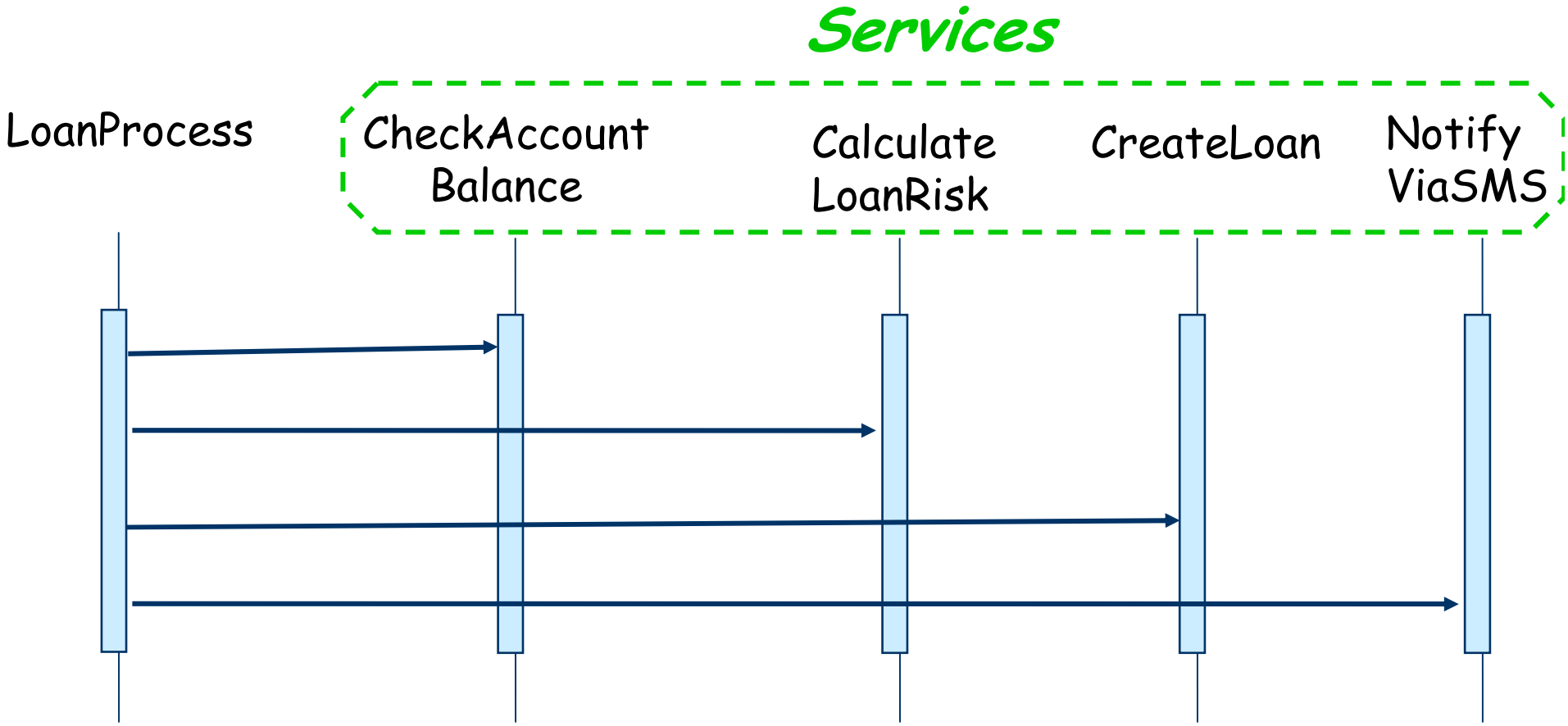


Exemple de couplage fort : Gestion de prêts



- LoanAgent est lié à LoanApproval et Loan
- LoanApproval est lié à Account
- Loan est lié à SMSGateway

Gestion de prêts en couplage faible



- Qu'est ce que LoanProcess ?
- Un **processus métier** !
  Il permet d'**orchestrer** les services => couplage lâche

Business Process Management (BPM)

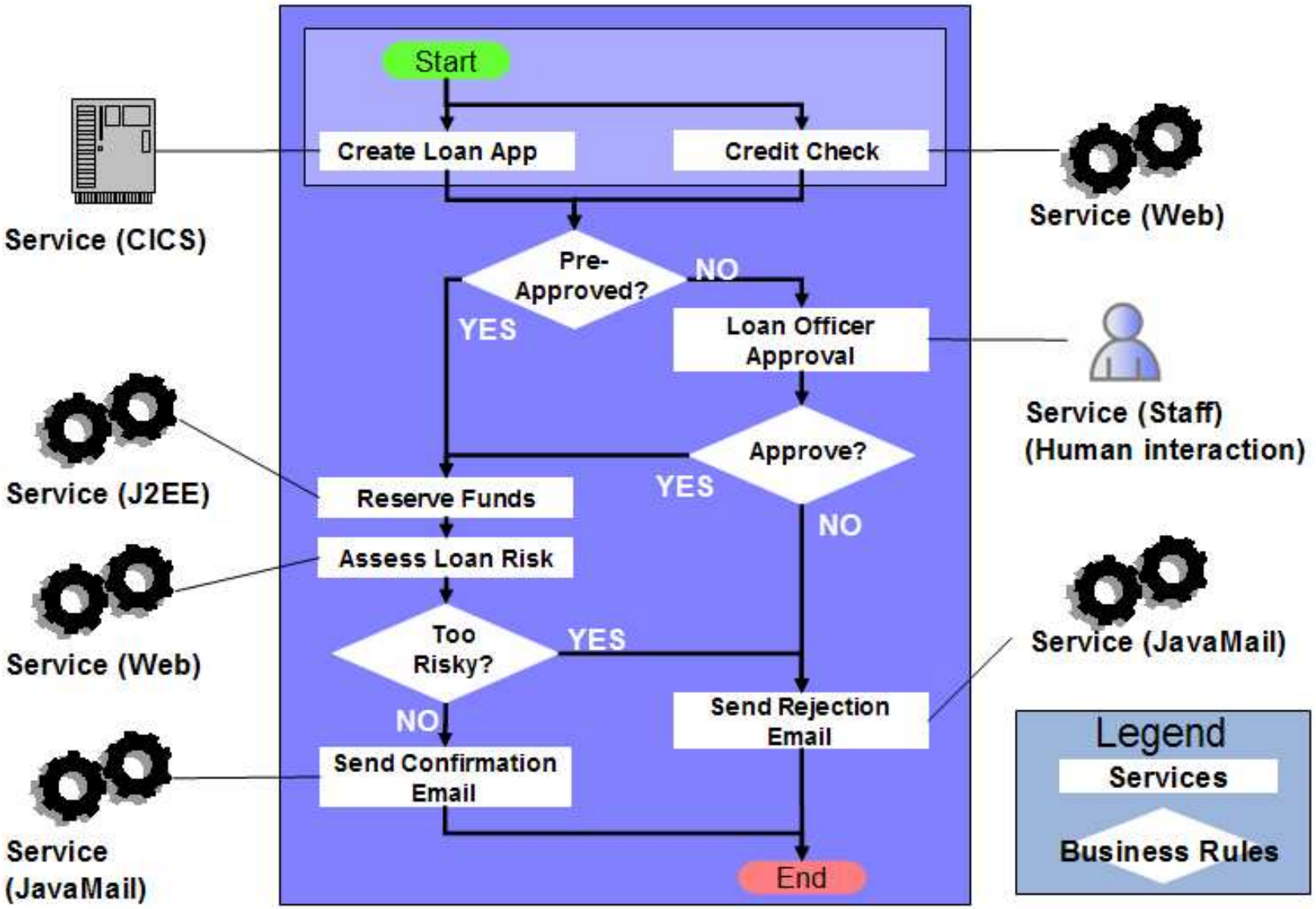
- But : Donner à l'Entreprise les moyens de gérer ses processus métiers de manière informatisée (modélisation, simulation, exécution et audit)
  - Optimisation, adaptation aux besoins en temps réel

- Un **processus** est composé de **sous processus**, de **décisions** (Business rules) et d'**activités**
- Un sous processus a son propre but, entrées et sorties
- Les activités
  - correspondent aux parties du processus métier qui n'incluent pas de décision et sont associées à des rôles
  - Sont réalisées par des systèmes ou des humains

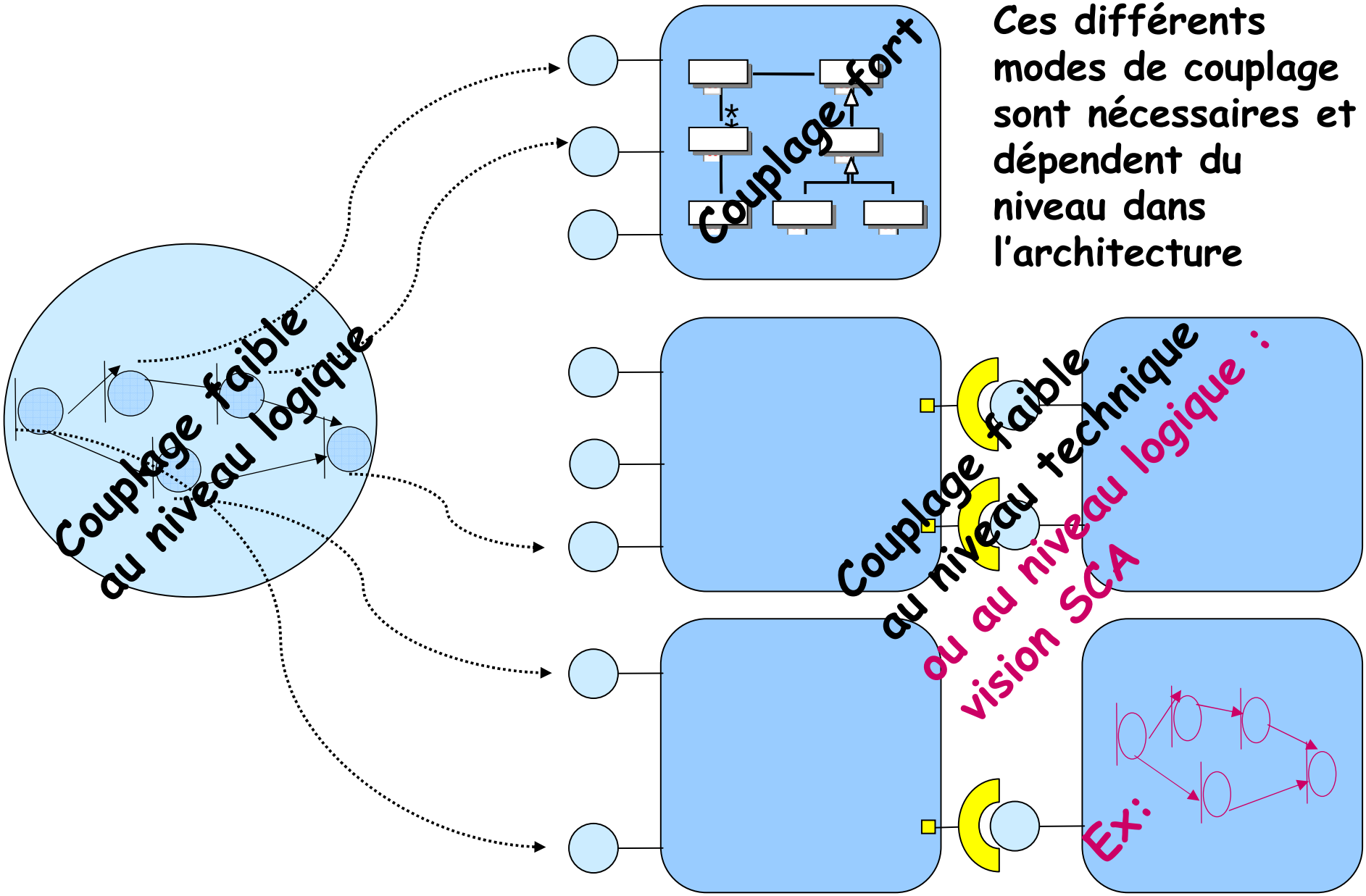
- Des **mesures** (KPI pour Key Performance Indicators) permettent de capturer les performances du processus

- Un processus est le **résultat d'une orchestration** de service
- Le processus est lui-même accessible en tant que service

BPM par l'exemple

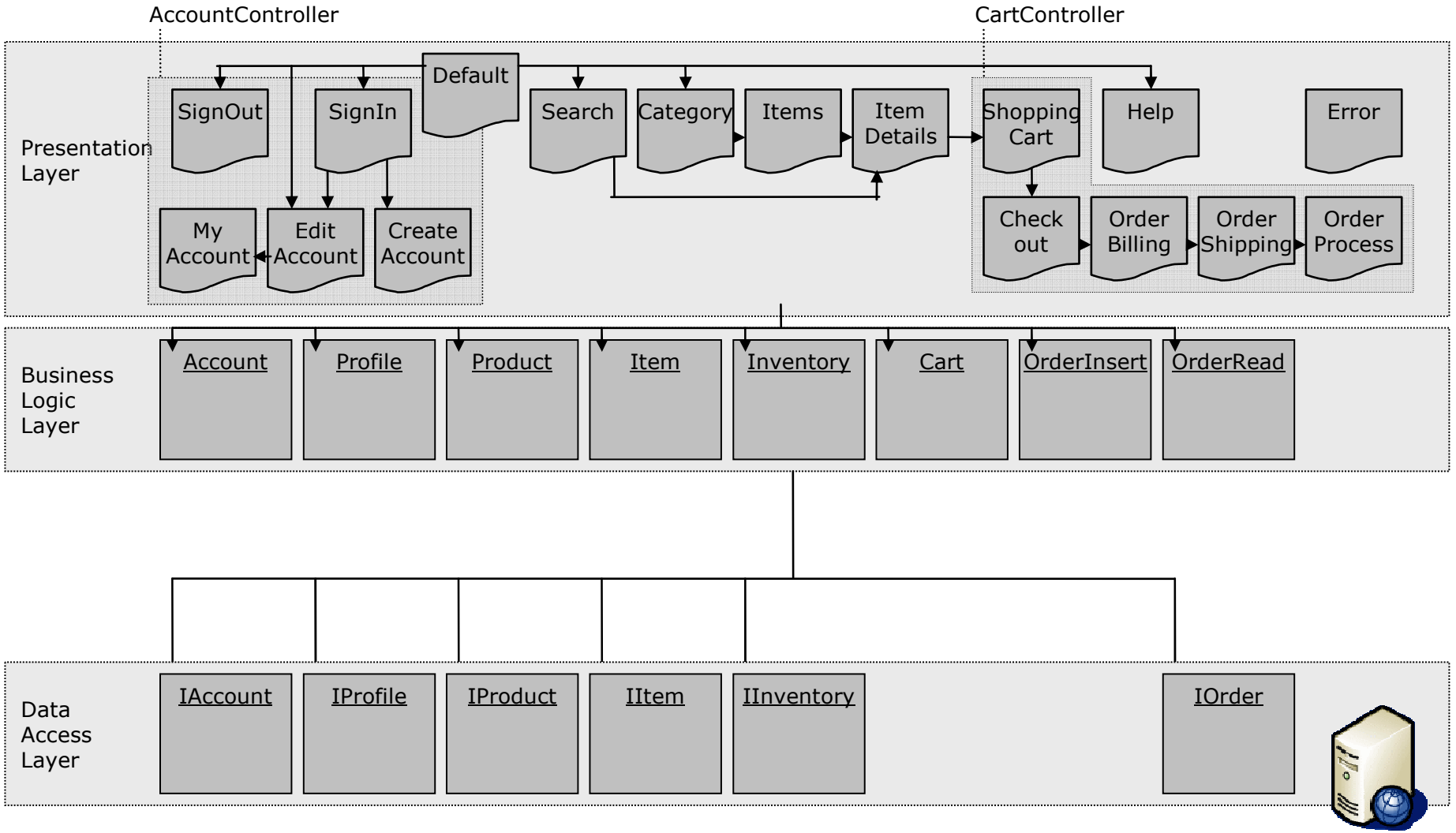


Les couches SOA

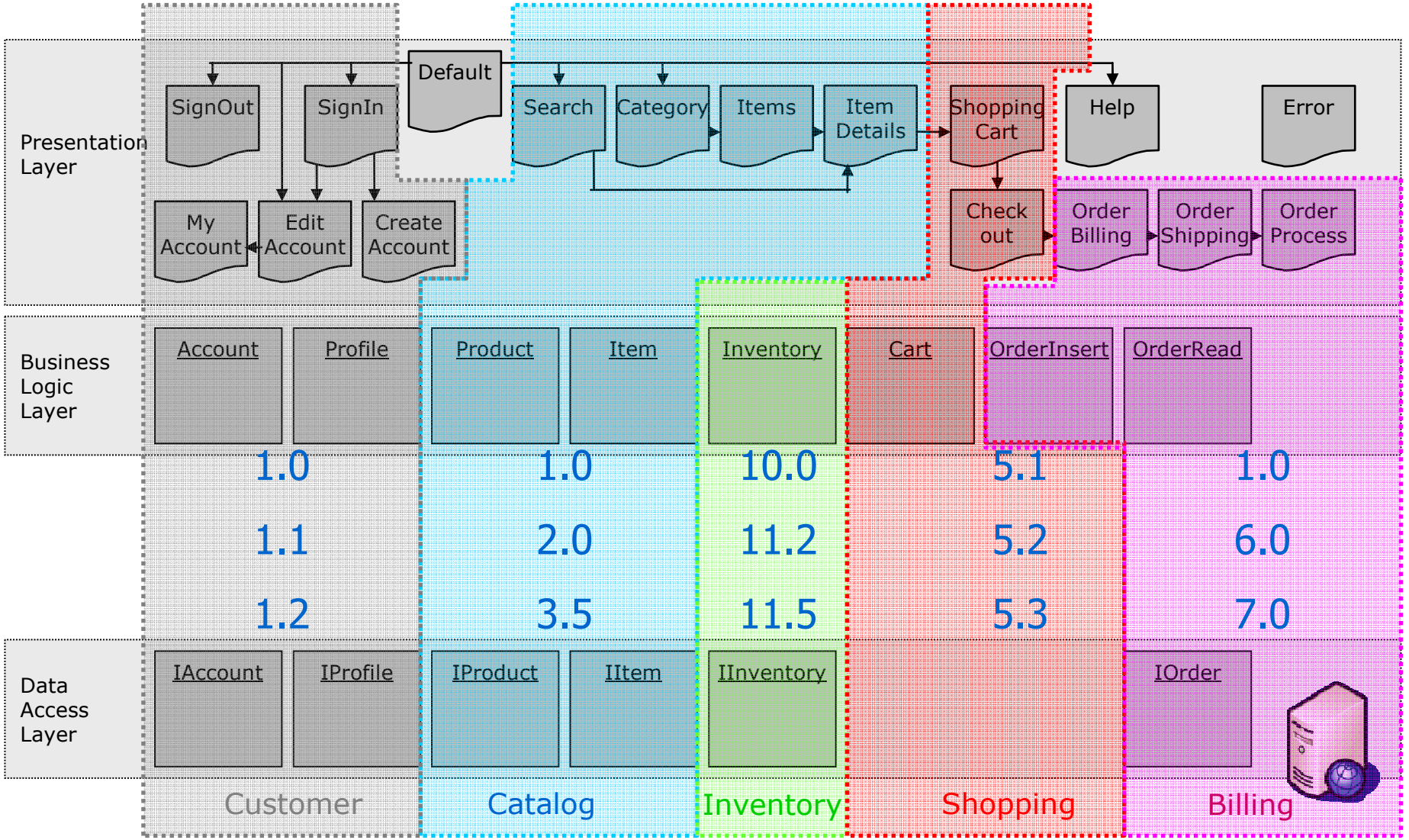


Ces différents modes de couplage sont nécessaires et dépendent du niveau dans l'architecture

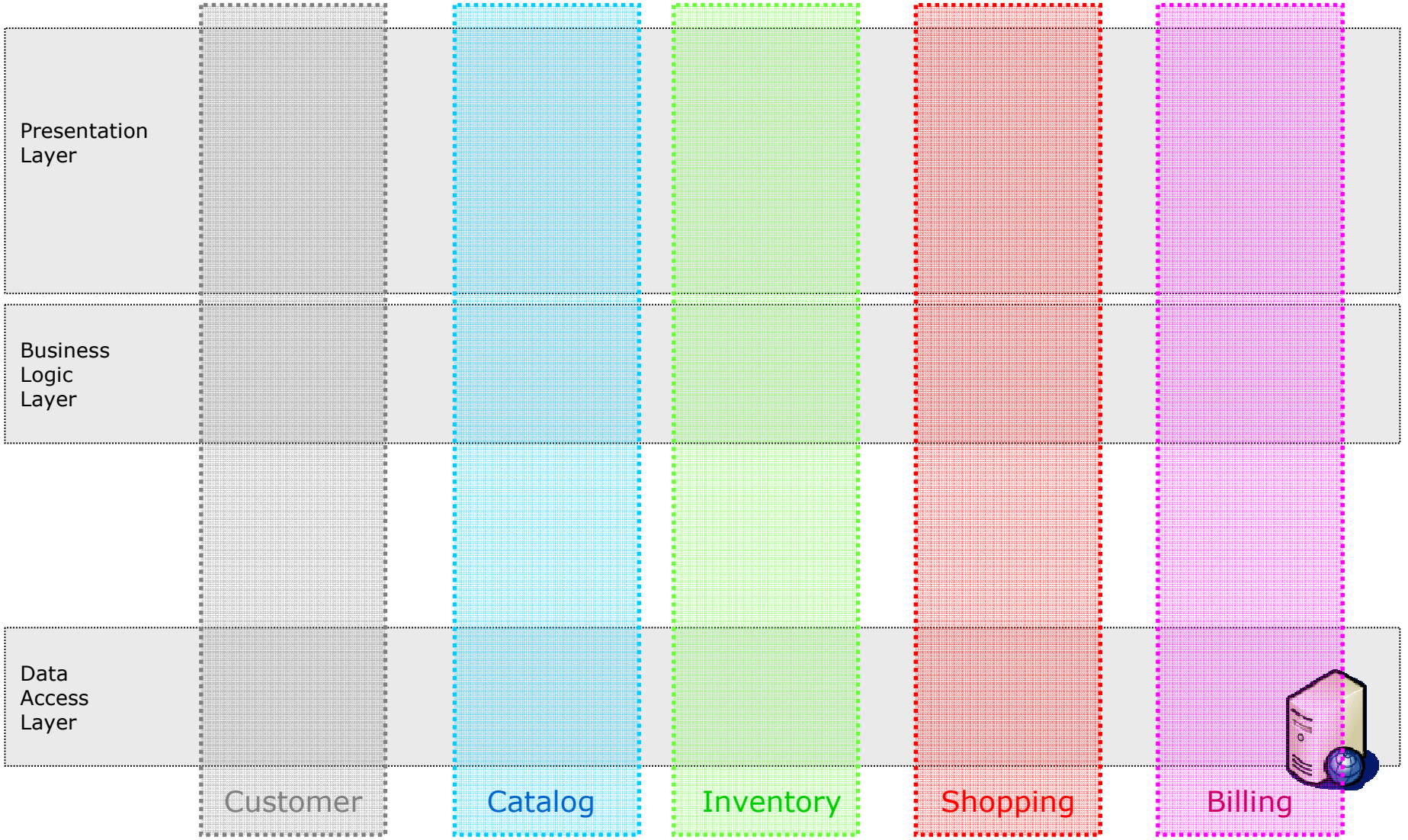
e-store : Couches



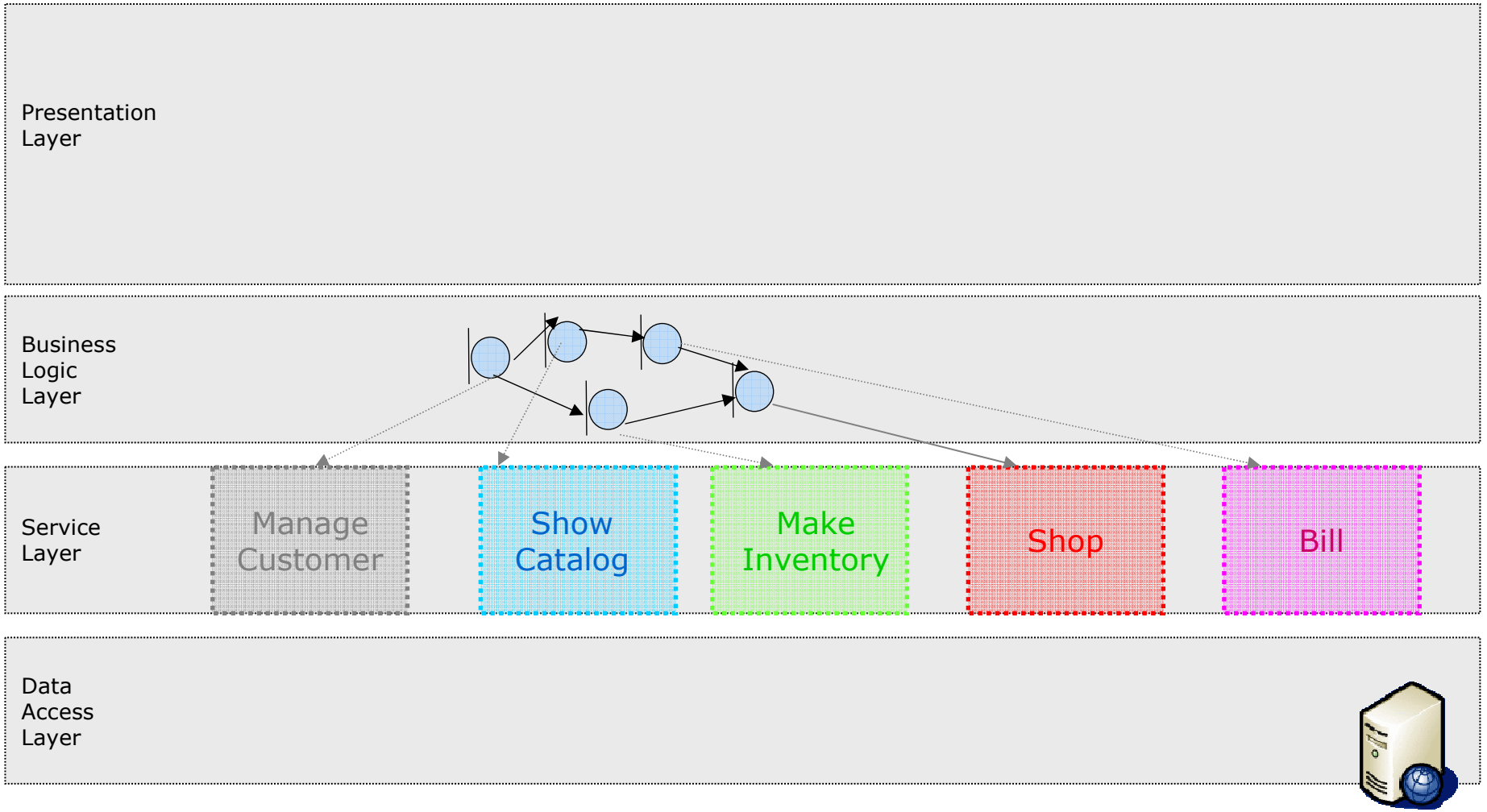
e-store : Domaines



e-store : Domaines



e-store : Services



Bénéfices métier

- Améliorer l'agilité et la flexibilité du métier
- Faciliter la gestion des processus métier

- Offrir la capacité à casser les barrières organisationnelles (silos)
- Réduire en temps le cycle de développement des produits

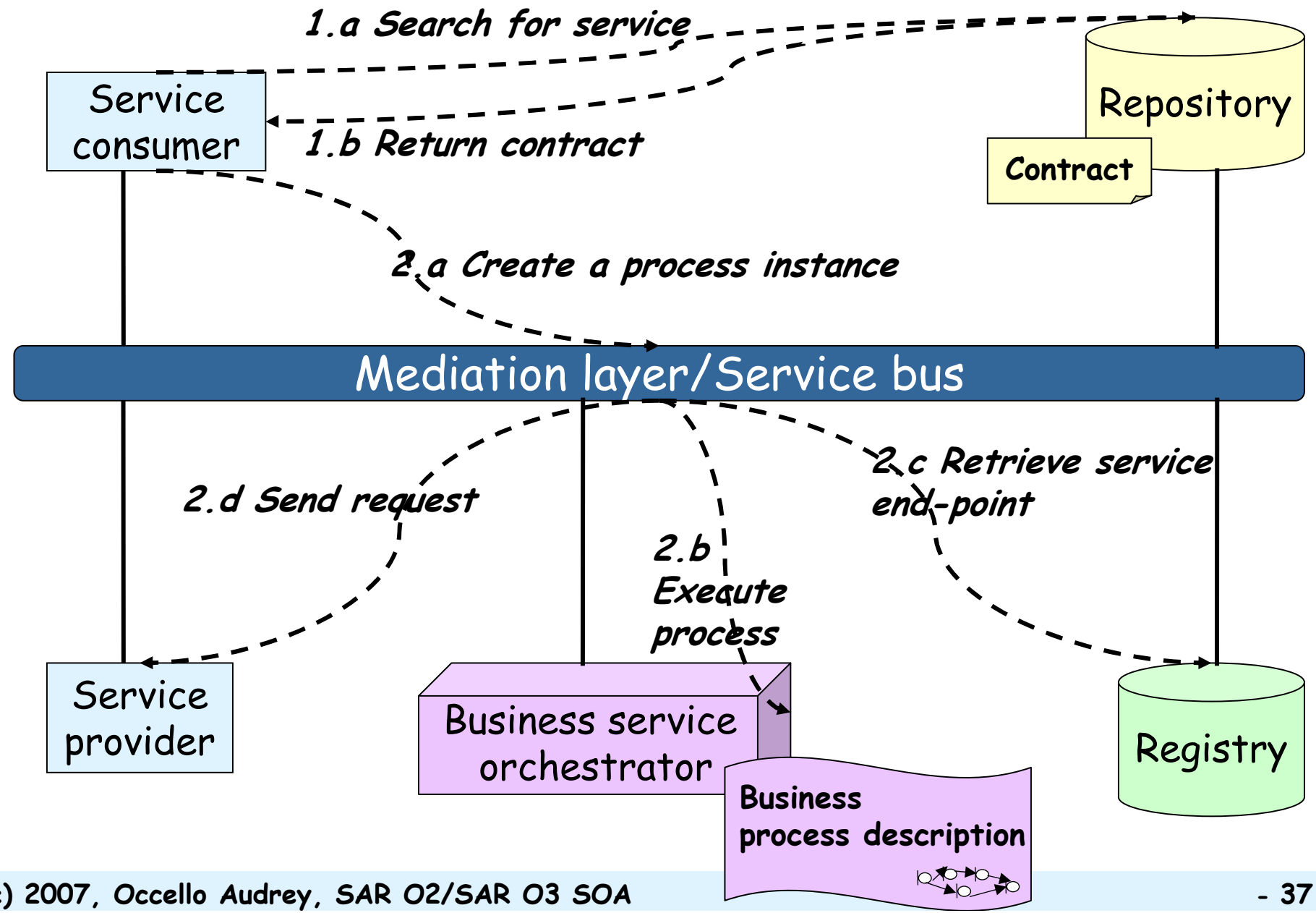
- Améliorer le retour sur investissement
- Accroître les opportunités de revenu

Bénéfices techniques

- Réduire la complexité de la solution
- Construire les services une seule fois et les utiliser fréquemment
- Garantir une intégration standardisée et le support de clients hétérogènes
- Faciliter la maintenabilité

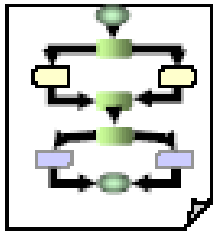
Quels sont les éléments clé d'une architecture orientée services ?

Points clés de l'architecture



Standards de l'architecture

Les standards sont un élément clé d'une SOA, ils assurent l'interopérabilité



SOAP
W3C
Simple Object Access Protocol

Transporte

WSDL
W3C
Web Services Description Language

Décrit le contrat

UDDI
Microsoft, IBM, HP
Universal Description Discovery and Integration

Spec pour Repository/Registry

BPEL
Oasis
Business Process Execution Language

Décrit les processus métier

Les trois piliers des Services Web

SOA et web services

- Attention à ne pas confondre les 2 !
  - SOA est un ensemble de concepts :
    Une SOA peut se mettre en œuvre sans Web Services
  - Les WS sont de l'ordre de la technologie :
    On peut utiliser les Web Services sans faire de SOA
- Les WS constituent la meilleure solution standardisée disponible
  - Un service métier = un webservice

Le langage BPEL

- Standard de l'OASIS
- Norme permettant de décrire des processus en XML

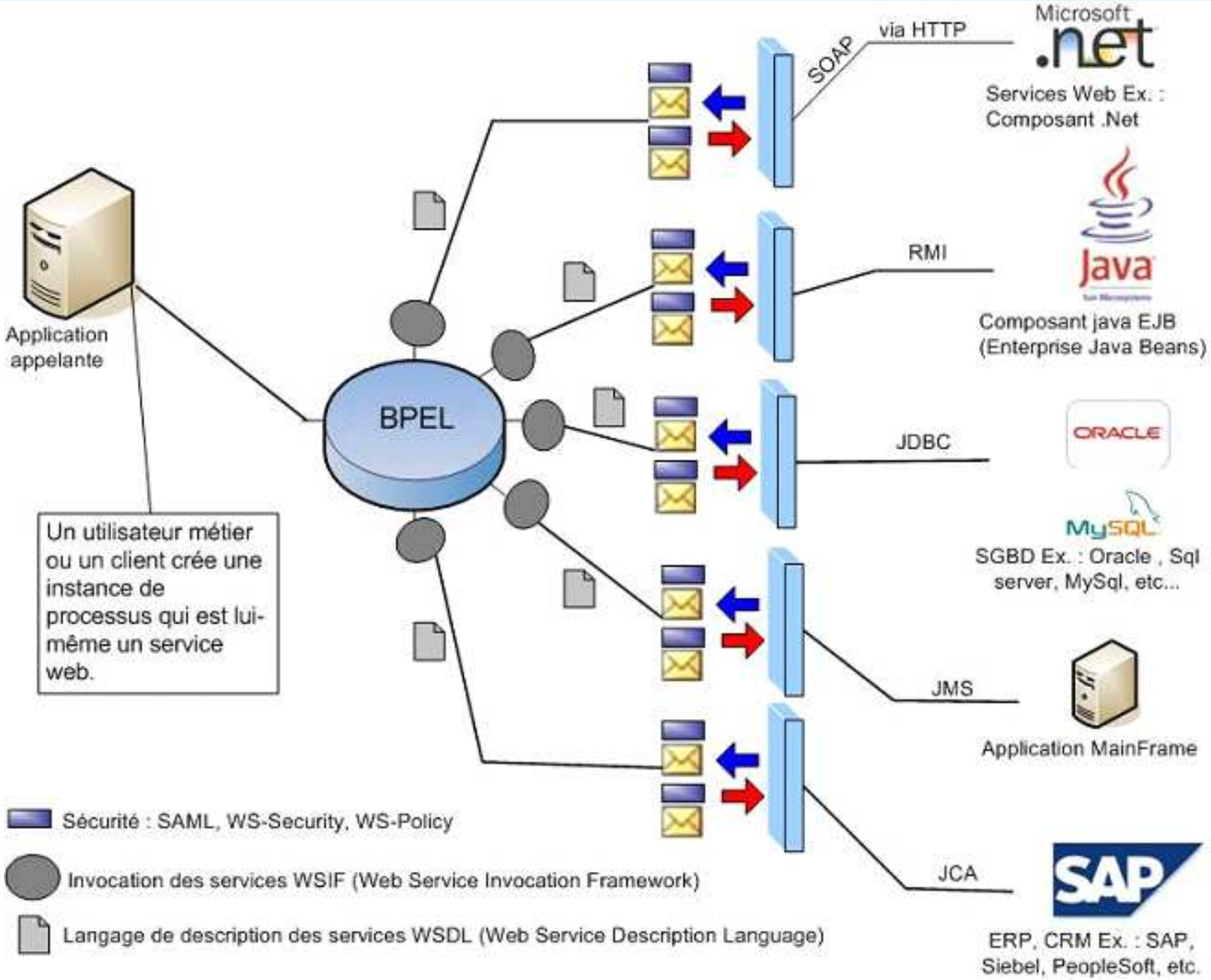
- Propose les fonctions basiques d'un langage de programmation:
  - sequence, flow, loop, switch…

- Identification des Instances de Process

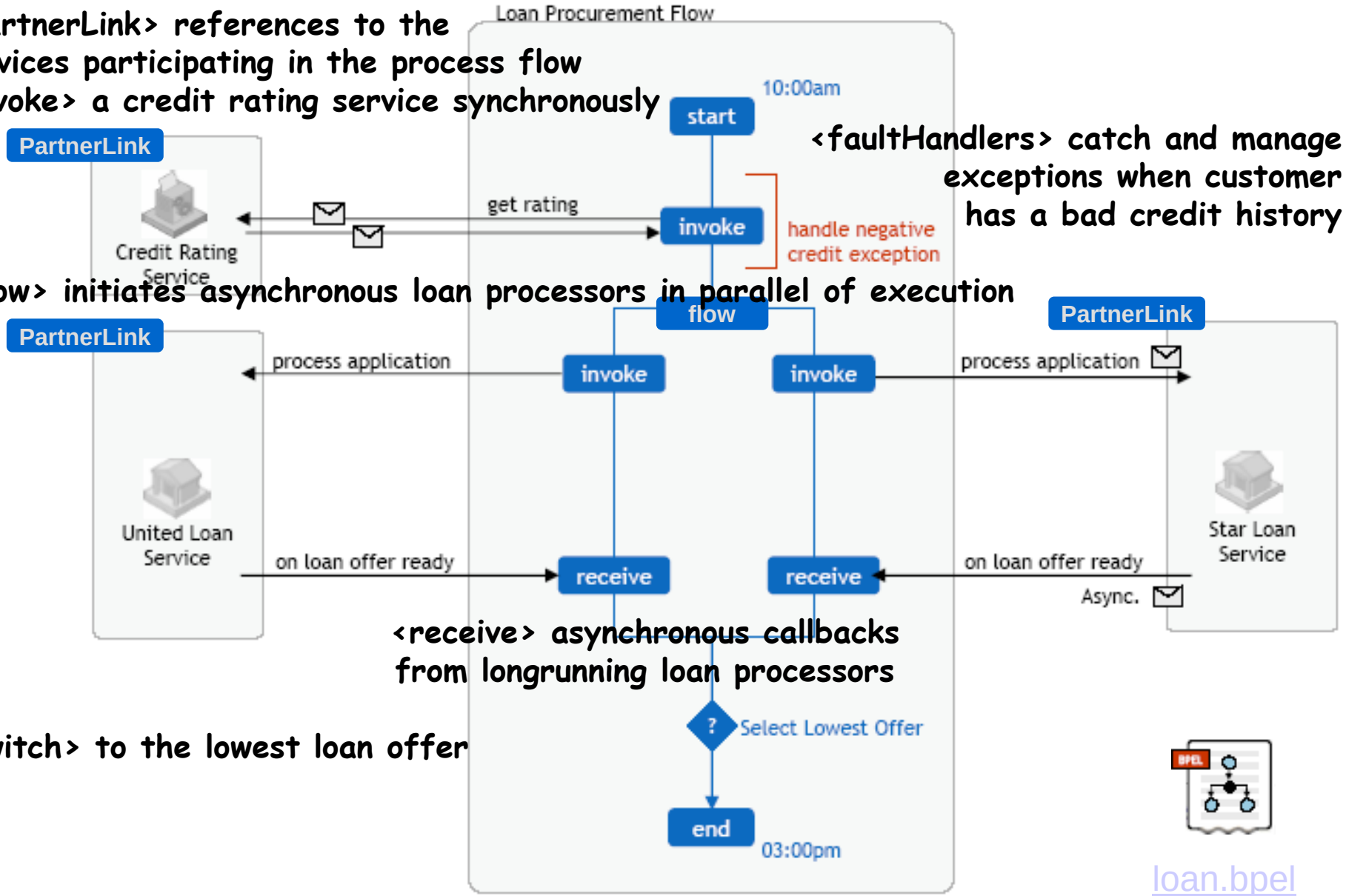
- Gestion des transactions longue durée (scope, compensation)

- Gestion des fautes

BPEL le chef d'orchestre



BPEL par l'exemple



Quelques détails sur le langage BPEL

- Transparents 52 -> 67 de

  http://arcad.essi.fr/riveill.old/enseignement/2007-08/SAR02/SAR%2002%20bpel.pdf

ESB : couche de médiation

- C'est le point d'entrée vers un service => invocation indirecte du service au travers du bus
- Ce point d'entrée doit être normalisé mais on ne sait pas qui fournit le service et comment il le fournit (implémentation).
- Infrastructure qui optimise les échanges entre consommateurs et fournisseurs de services. Il peut prendre en charge :
  - Routage
  - transformation des données
  - transactions,
  - sécurité,
  - qualité de service,
  - …

  Ex: voir http://petals.ow2.org/what-is-petals-esb.html
- Le but d'un ESB est de permettre de communiquer de manière simple et standardisée entre des applications hétérogènes

Quelques manières d'implémenter un ESB

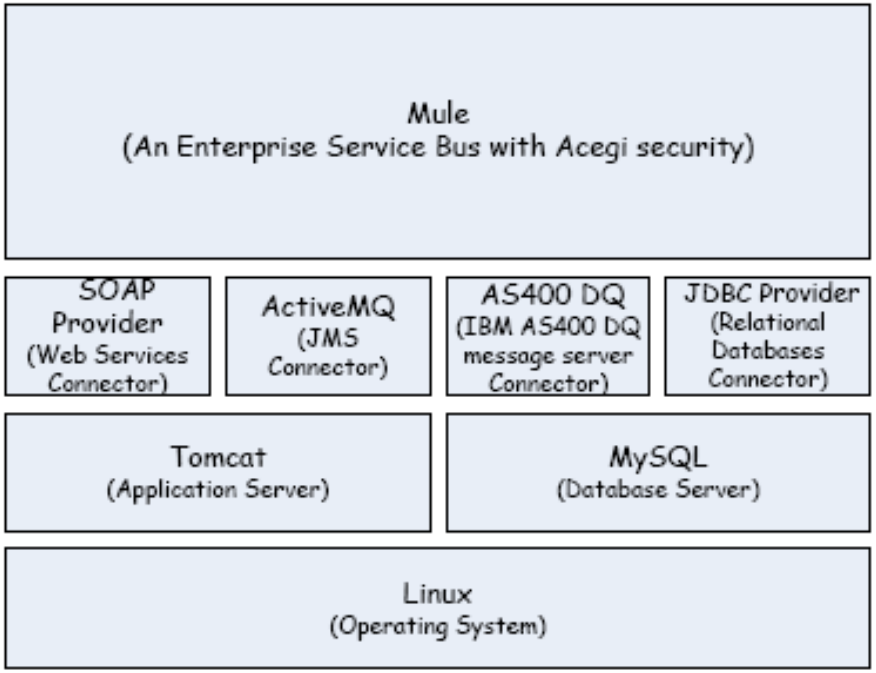
- Intergiciels de type MOM (Message Oriented Middleware)
- Intergiciels de type Bus (CORBA par exemple)
- Intergiciels de type EAI (Message Broker avec connecteurs propriétaires liés au moteur d'intégration)
- Routeurs Web services tel que WebSphere Web Services Gateway

➢ ***Selon le type d'implémentation retenu, l'ESB assurera plus ou moins de "services" : le choix dépend des besoins***

➢ ***L'ESB n'est pas obligatoire : mais il est fortement recommandé pour éviter le couplage* entre fournisseur et consommateur**

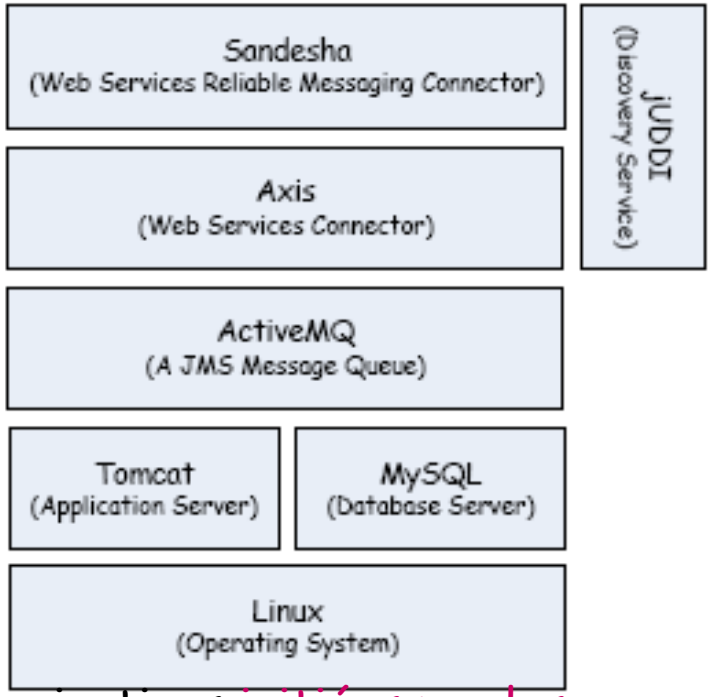
Exemples d'architecture techniques se basant ou pas sur un ESB

Avec ESB



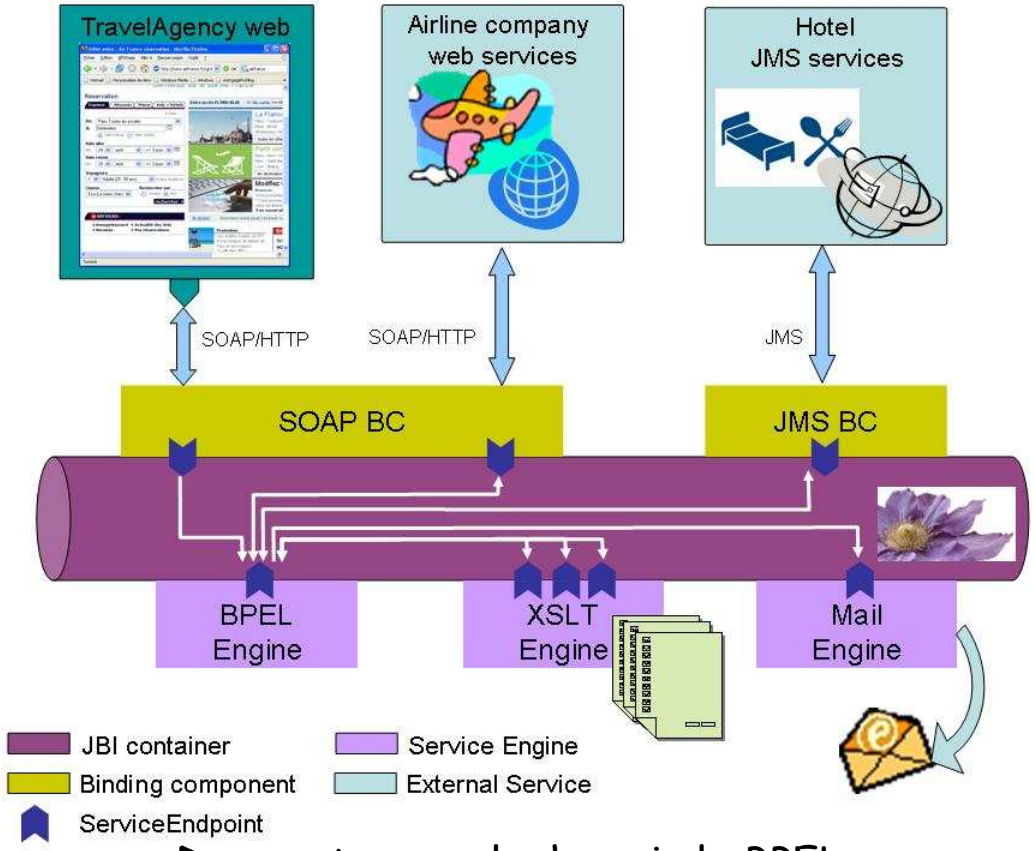
- Plusieurs connecteurs
- Orchestration importante
- Transactions conséquentes

Sans ESB



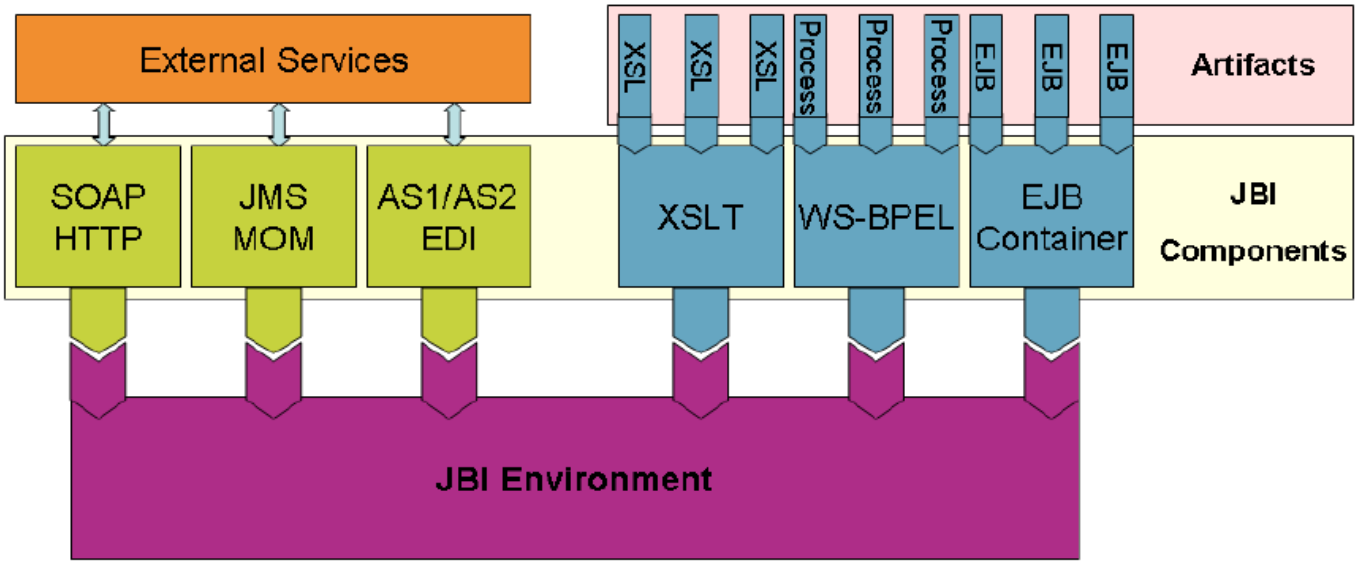
- Communications initiées par les applications seront donc homogènes
- Pas d'orchestration, parce que pas d'intermédiaire: invocations de services directement pilotées par le code
- Peu de transactions, ou alors les gérer "à la main"

Intégration applicative via un bus JBI



- Dans cet exemple, hormis le BPEL process, tous les autres éléments applicatifs sont des services externes au bus.
- Mais, par ex. un élément pourrait être un autre BPEL process ou un composant EJB, ou autre, déployé DANS le bus, et vu comme un service interne.

Specification JBI pour ESB (ouvert)



- BC et SE peuvent se rajouter (et s'enregistrer) sur le bus dynamiquement

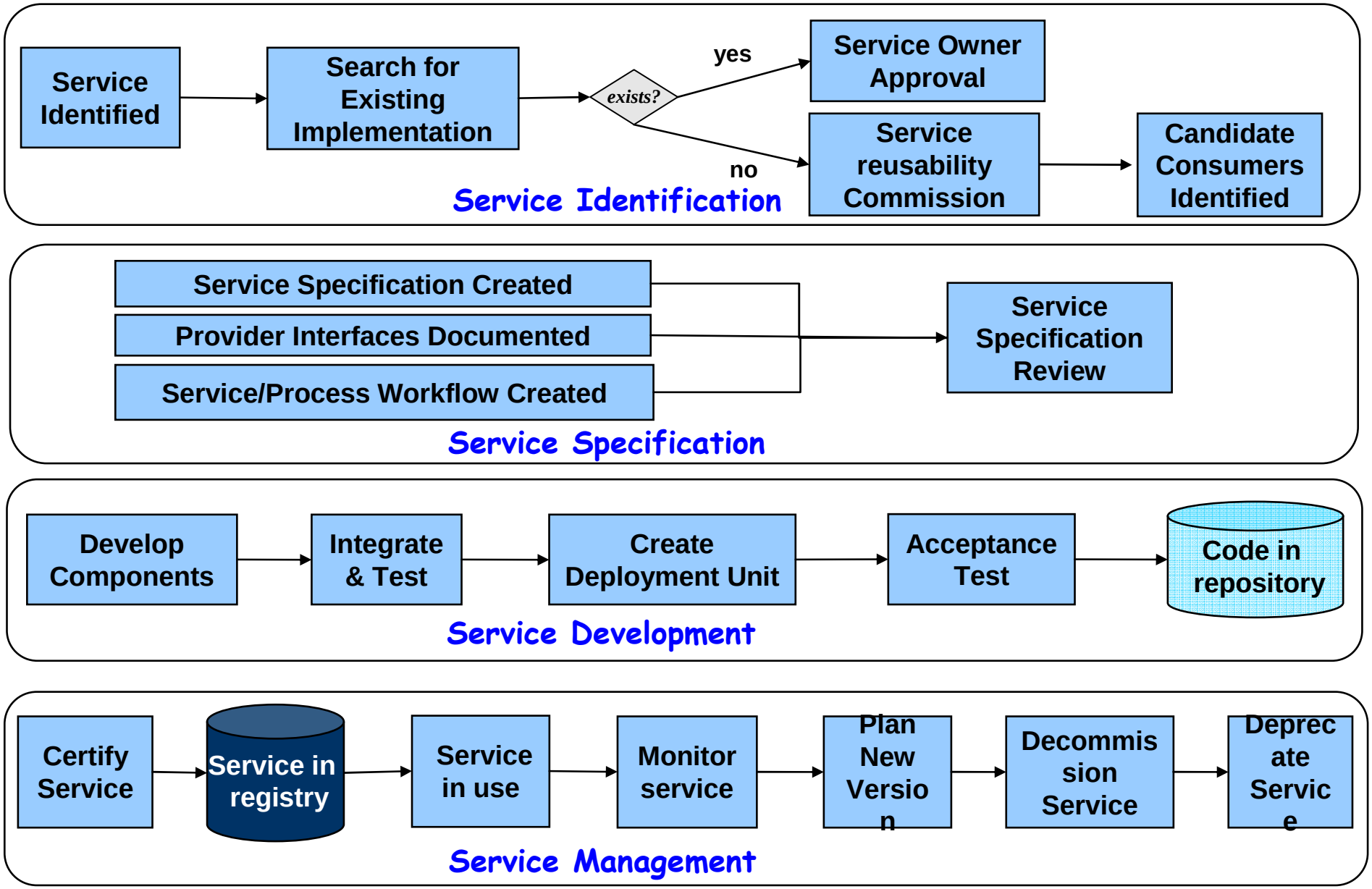
Quel est le cycle de vie d'un service ?

Découpage du cycle de vie d'un service

- 4 grandes phases :
  - Identification
  - Spécification
  - Développement
  - Gestion

- 1 aspect tranversal : la gouvernance
  - Les architectures orientées service impliquent une vision globale
  - La gouvernance permet de casser les silos de l'entreprise

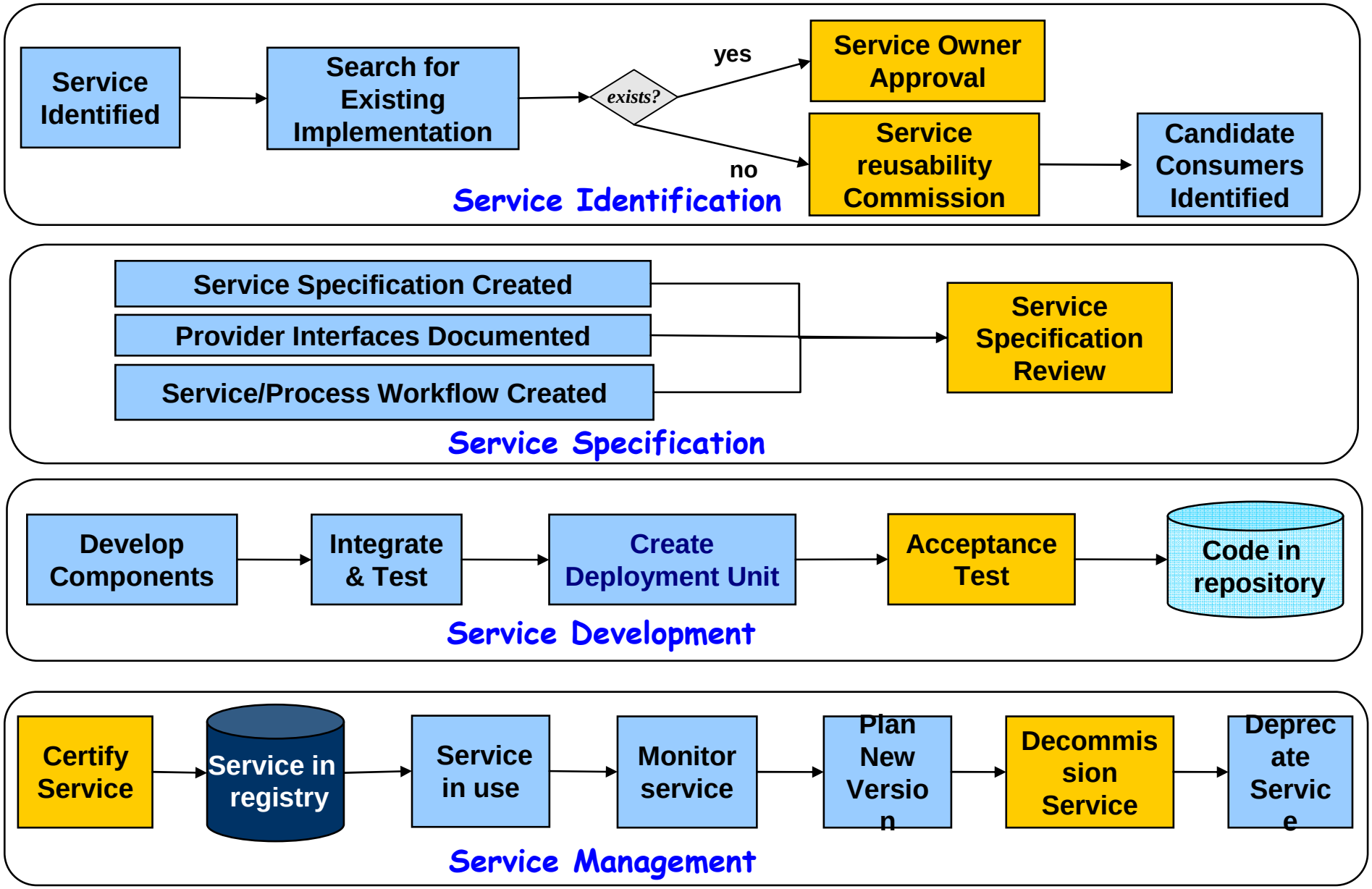
Cycle de vie des services



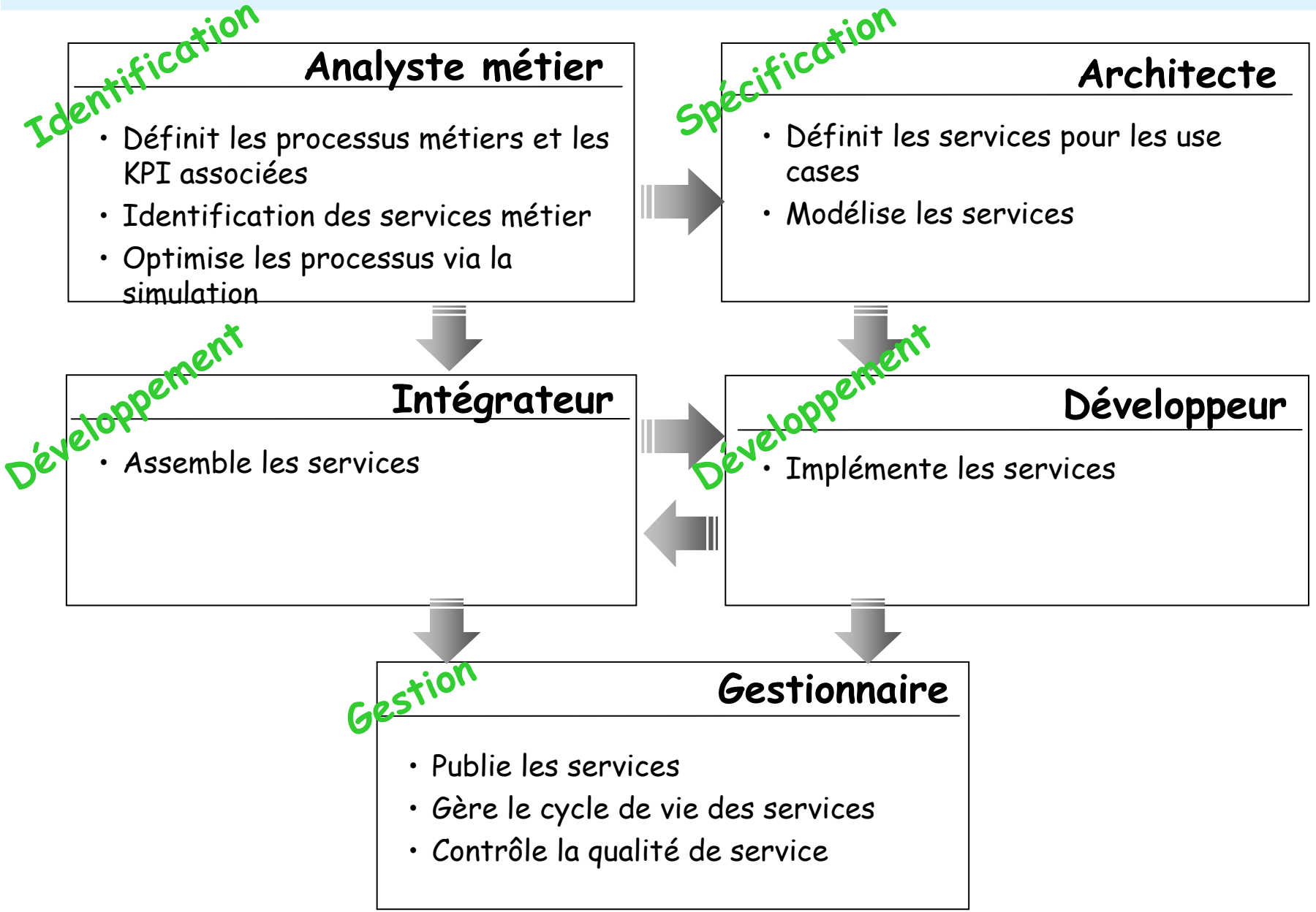
La gouvernance en quelques questions

- Qui définit et modifie les services ?
- Qui peut y accéder ?
- Quelle est la qualité que les services doivent offrir ?
- Qui paie pour ces services ?
- Qui est responsable de l'infrastructure ?
- Qui gère les interdépendances entre les services ?
- Comment exposer les services aux entreprises partenaires ?

Cycle de vie des services (activités de gouvernance)



Rôles associés au cycle de vie des services



Zoom sur la phase d'identification

- Un des problèmes centraux pour mettre en œuvre une SOA
- La granularité des services est fondamentale
  - détermine en grande partie la réutilisabilité des services
- Or succès SOA = % de réutilisation des services

- Éviter une granularité trop fine qui entraîne :
  - beaucoup d'interactions
  - des problèmes de performance

- On recommande des services à "gros grain"
  - attention à une granularité trop "épaisse"
  - un service qui fait trop de chose, risque de ne pas être réutilisable

➢ ***Trouver le juste milieu***

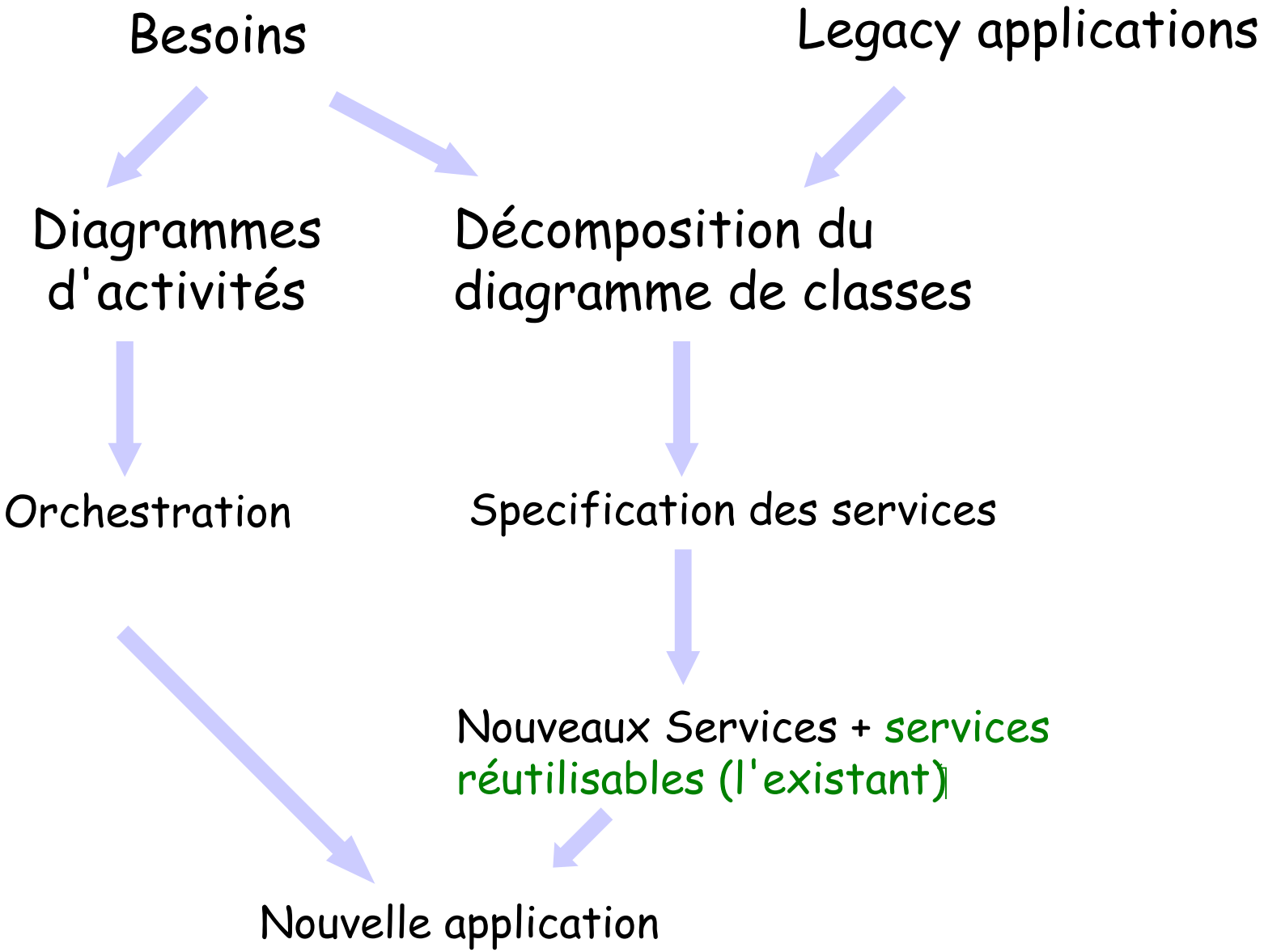
2 méthodes d'identification des services

- Une première phase d'indentification doit être effectuée sur l'ensemble du SI dans le cadre de son urbanisation en s'appuyant sur la cartographie des domaines métiers de l'entreprise et sur le code existant
- Approche incrémentale : une phase d'identification est nécessaire au démarrage de chaque nouveau projet SOA en s'appuyant sur les processus et services répertoriés précédemment

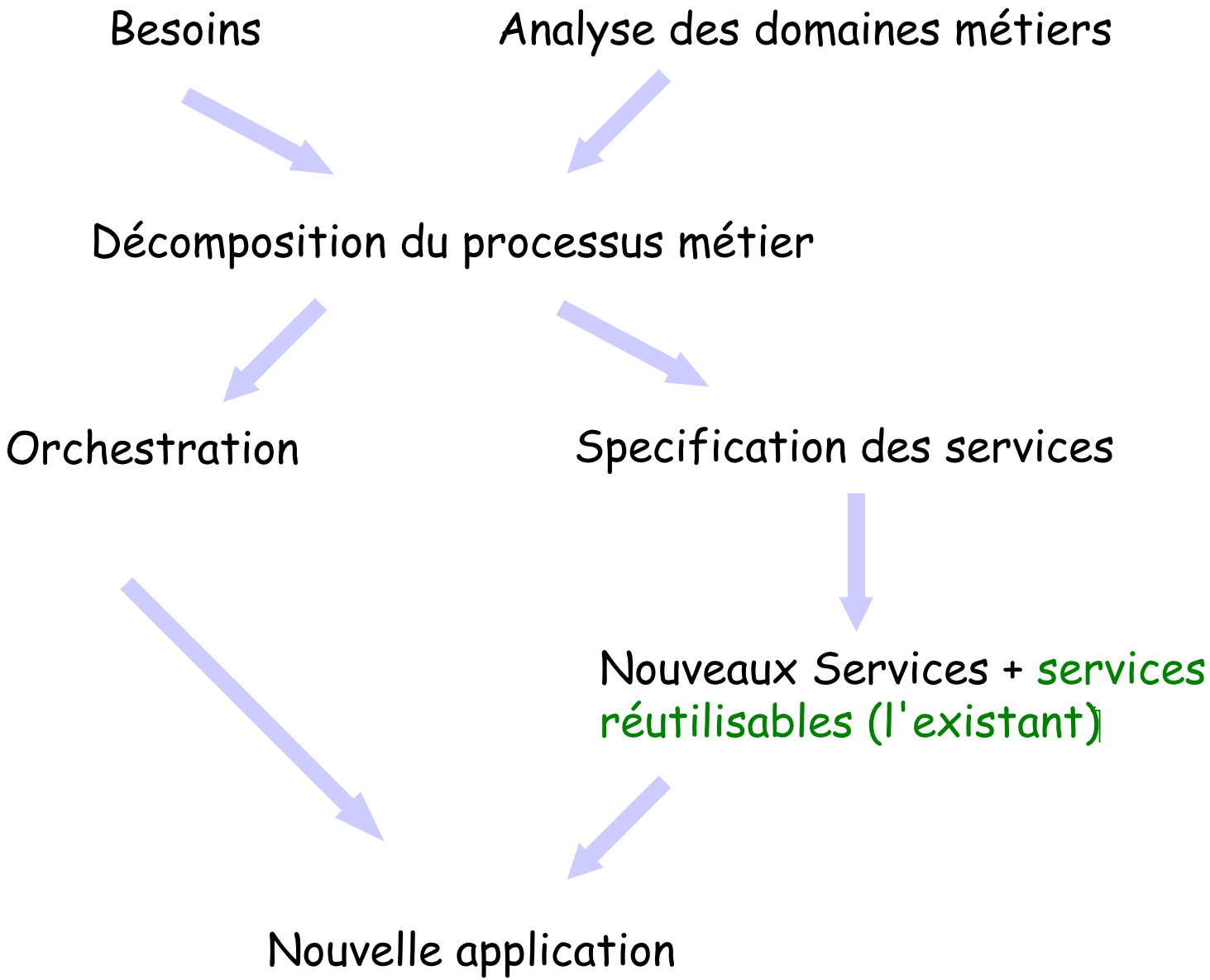
- Approche Bottom-up :
  - On part des briques informatiques, on rassemble les bouts (abstraction)
  - Réalisée généralement par la MOE
  - Plus adéquat pour réutiliser l'existant non "SOA-isé"

- Approche Top-down :
  - On part des interactions métier pour aboutir aux interactions techniques
  - Réalisée généralement par la MOA
  - Plus adéquat pour démarrer un nouveau projet

Approche "Bottom Up"

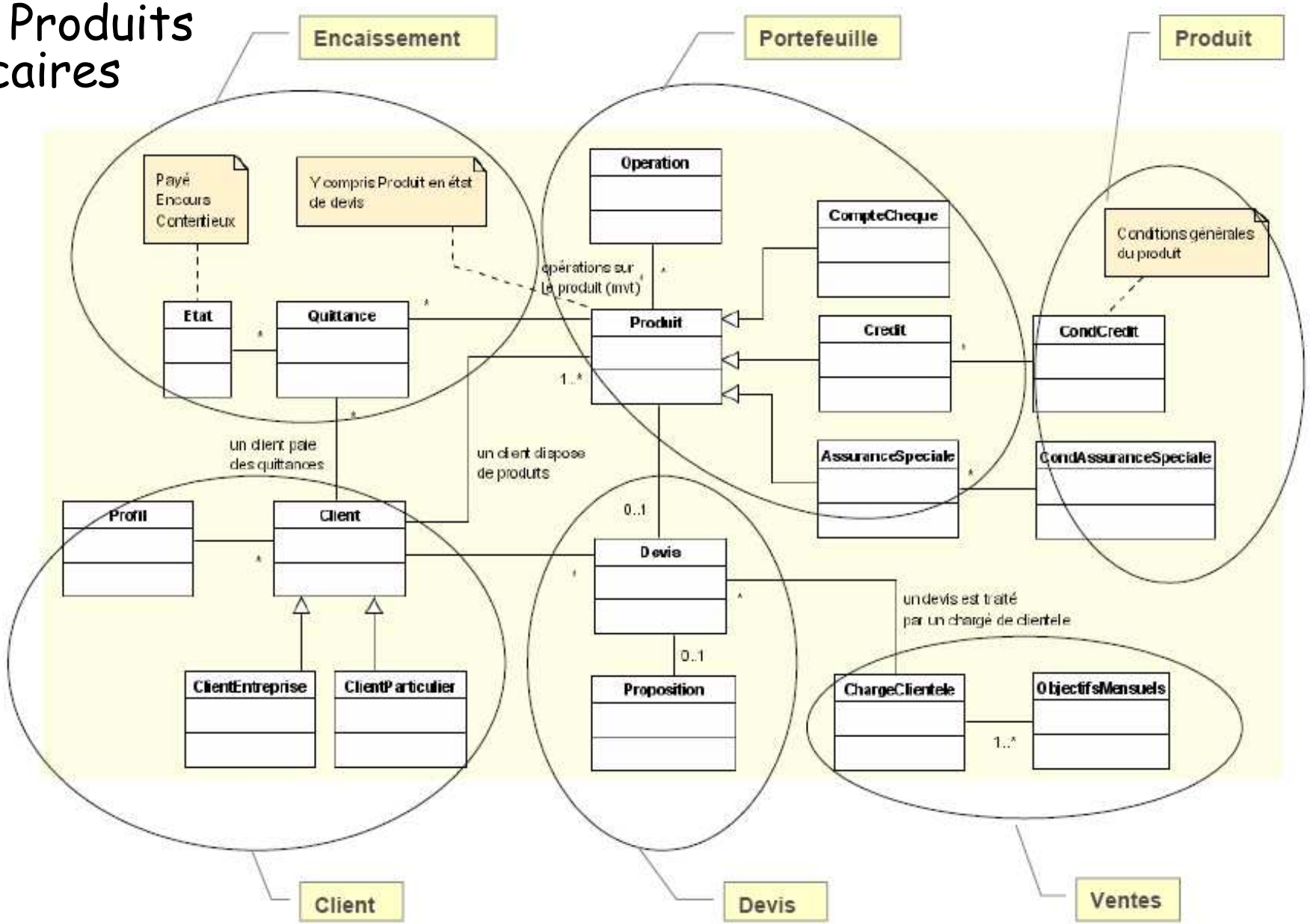


Approche "Top Down"



Méthode Orchestra - Cartographie

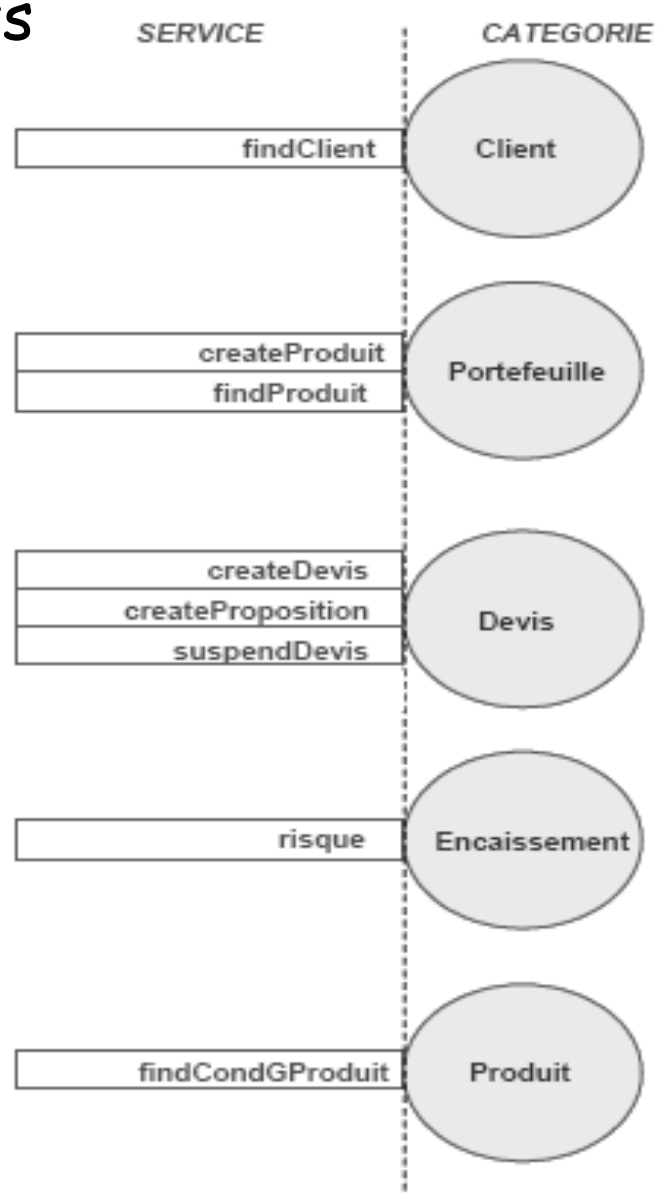
Ex : Produits bancaires

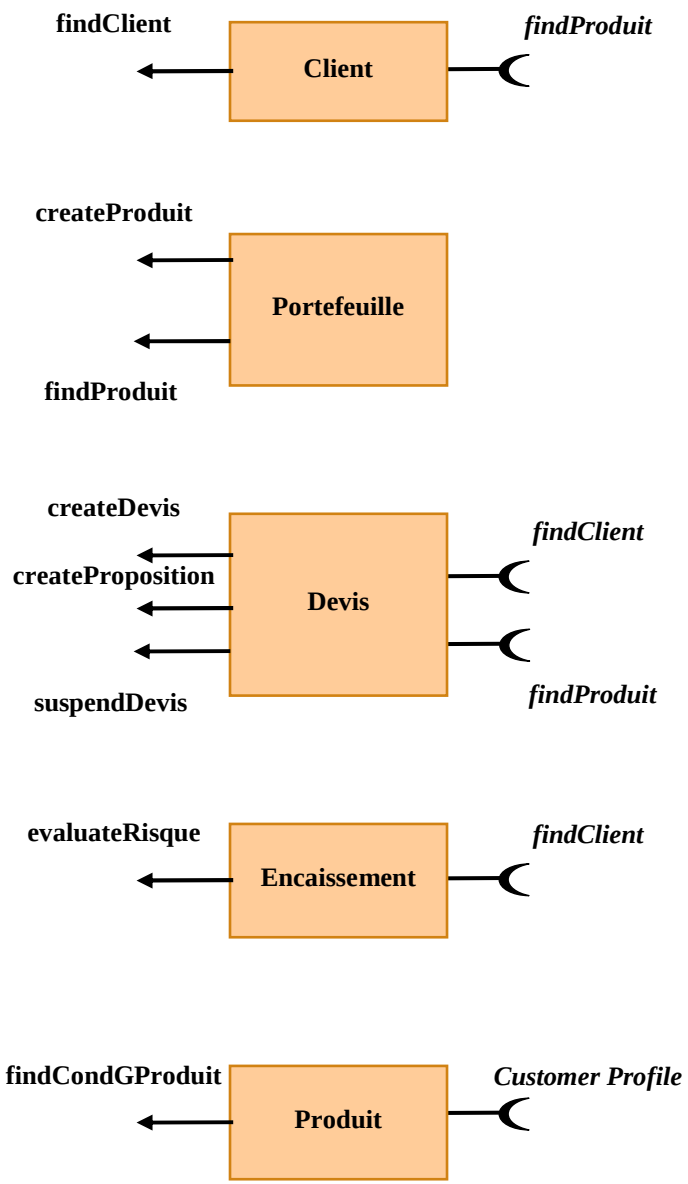


Pas plus de 12 classes par catégorie

Méthode Orchestra - services

Ex : Produits bancaires





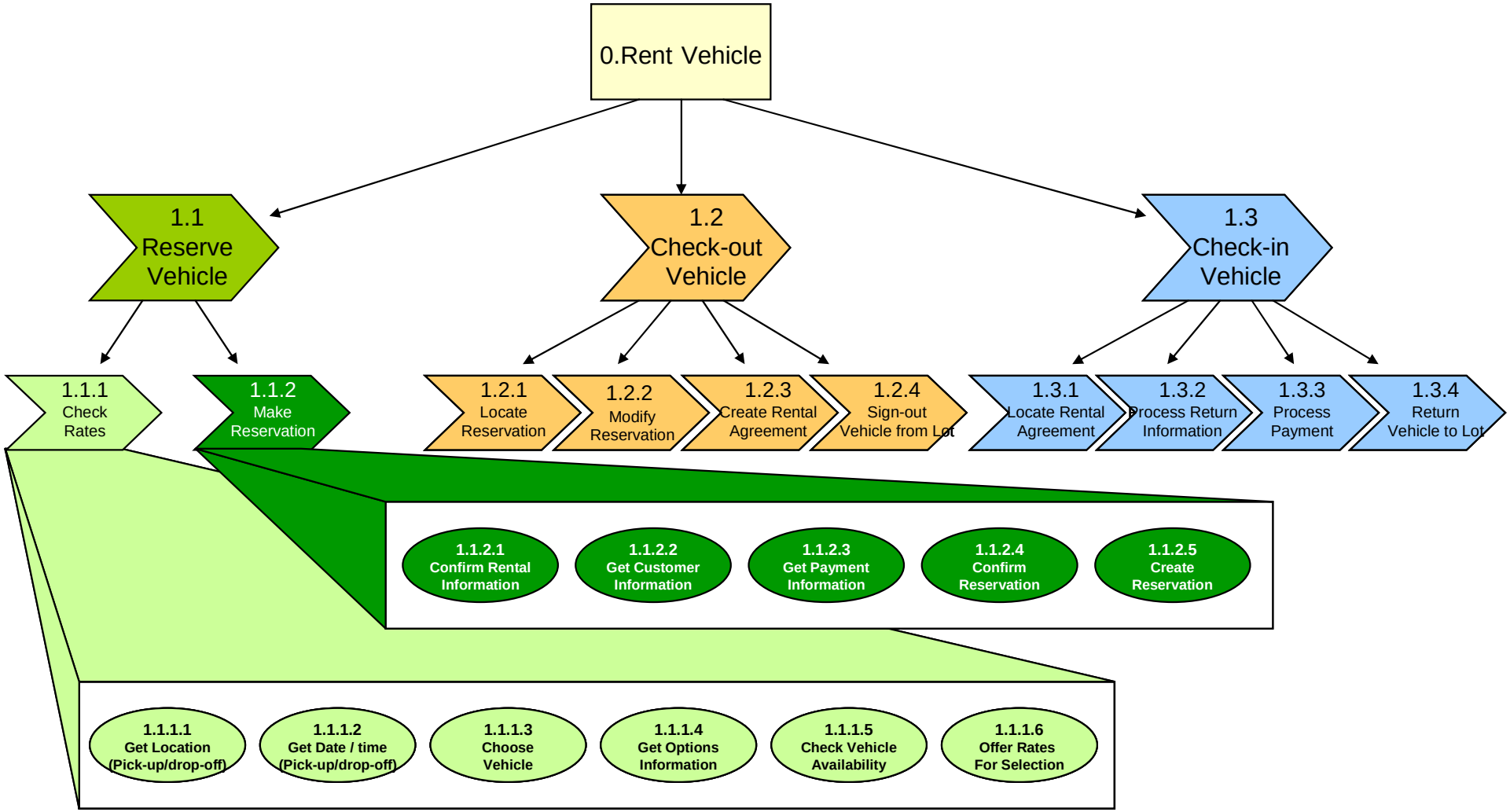
Méthode IBM SOMA : cartographie des domaines métiers

Component Business Model (CBM) – ex : Location de véhicules

	Marketing & Customer Mgt.	Products	Rentals management	Rental Fleet Logistics	Business Administration
Direct	Customer Segmentation Customer Relationship Strategy Marketing Strategy & Planning	Rental Product Strategy Product Development / Design	Location & Channel Strategy Location Design & Layout Channel Design & Layout	Fleet Strategy Fleet Planning OEM Relationship Planning	Corporate / LOB Strategy Financial Management & Planning Real Estate Planning
Control	Customer Behavior Modeling Market & Competitor Research Segmentation Management Call Center Campaign Management	Promotions Management Pricing Management	Channel & Location Profitability Location Operations Management Reservations Management Workforce Management	OEM Performance Management In-bound Logistics	Alliance Management Business Performance Reporting Legal & Regulatory Compliance Real Estate & Construction Management Risk Management Stock Ledger HR Management (Career Dev., Training, Recruiting)
Execute	Customer Service Preferred Member Mgmt Customer Communications Mass Marketing & Advertising Target Marketing	Purchasing / Sourcing Demand Forecasting	Rentals & Reservations Time & Attendance	Location Operations Fleet Servicing Fleet Management	HR Administration / Payroll Corporate Audit Corporate Accounting (GL, AP, A/R, Treasury, etc.) Indirect Procurement PR & Investor Relations IT Systems & Operations

Méthode IBM SOMA : décomposition des processus métiers

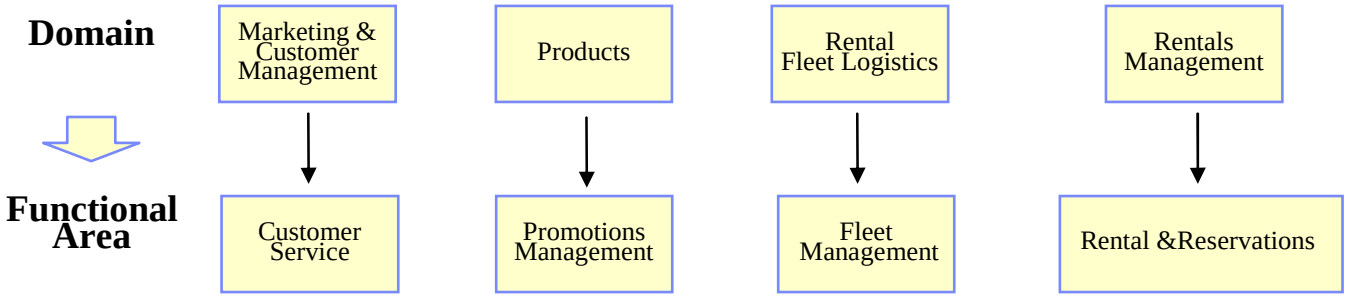
Ex : Location de véhicules

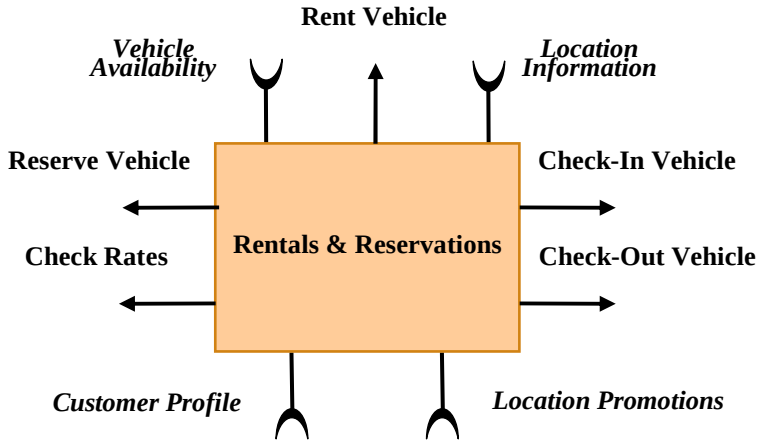


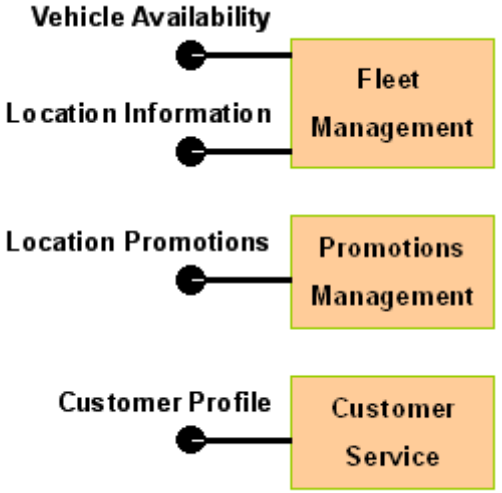
On s'arrête au troisième niveau

Méthode IBM SOMA : identification des services

Ex : Location de véhicules







Approche "Outside in"

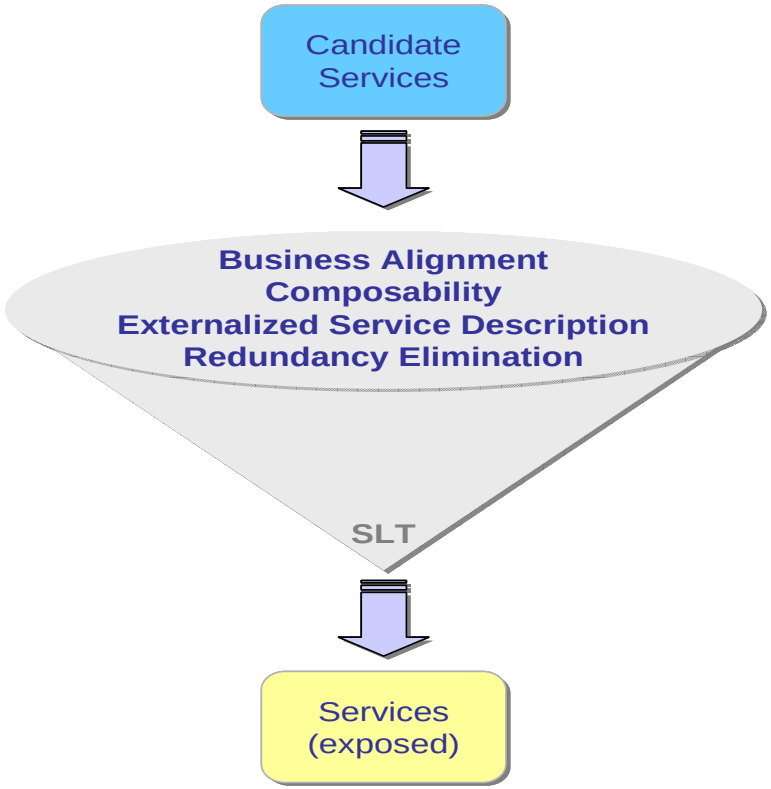
- Dans la pratique on utilise rarement une seule approche

- Pour obtenir une granularité pertinente des services, il est nécessaire de concilier les 2
  - Faire l'analyse Top-down sans se préoccuper de l'existant
  - Faire l'analyse Buttom-up en ne considérant que l'existant
  - Comparer les services "remontés" avec ceux déduits des processus
  - Faire les compromis nécessaires pour réutiliser le maximum de code

Zoom sur la phase de spécification

- Les services identifiés ne doivent pas être tous publiés :
  - Chaque service a un coût et un risque
  - Il faut éviter la prolifération des services

- Le "**Service Litmus Test**" d'IBM aide à trouver les "bons" services à exposer

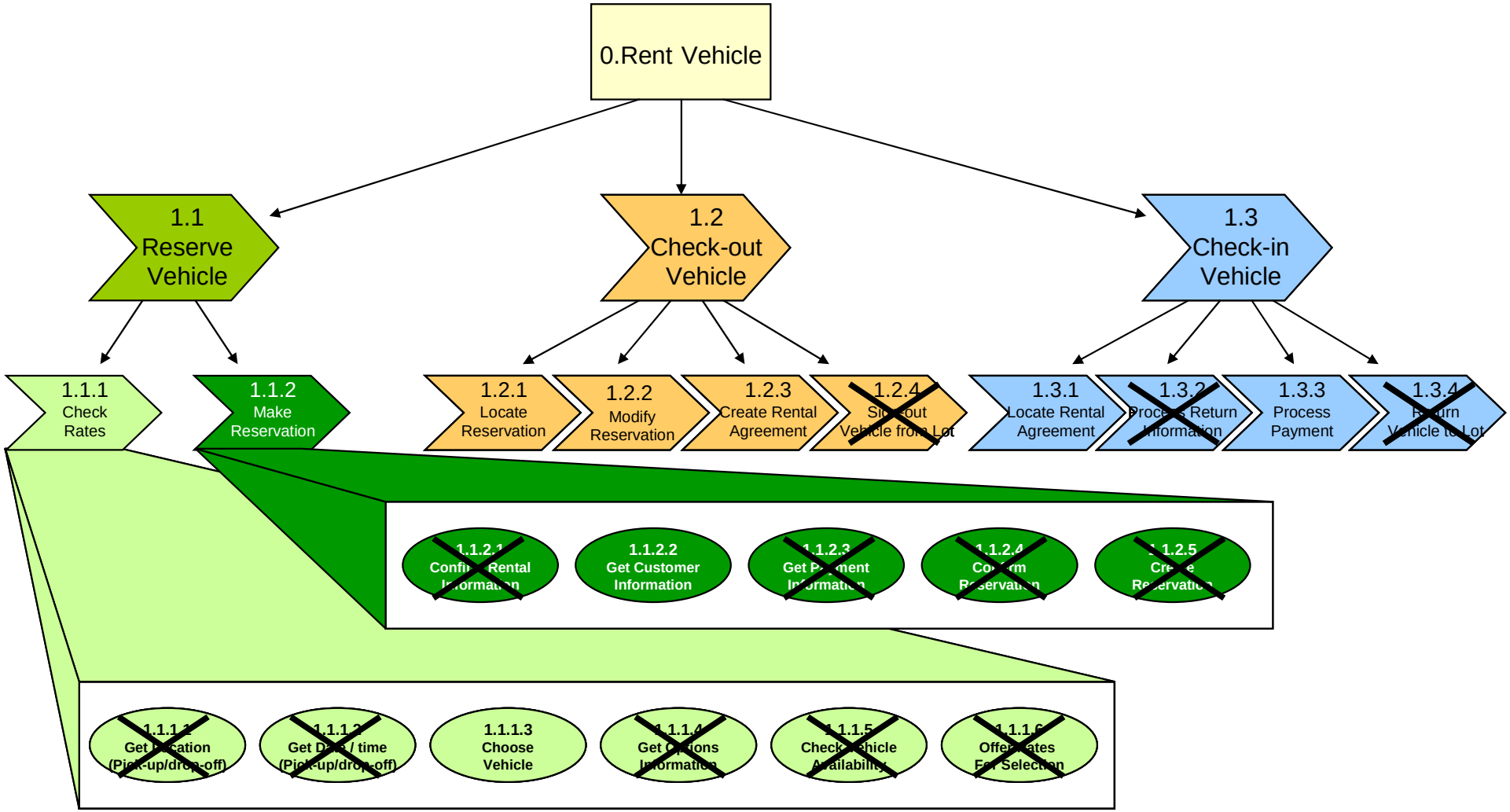


Quelques critères d' “exposabilité”

- Le potentiel d'un service est d'autant plus important qu'il :
  - permet d'automatiser un processus métier critique
  - est réutilisable par plusieurs domaines métiers
  - remplace une application désuette
  - supporte des besoins non fonctionnels (sécurité, logging, monitoring, ...)

- Les services non exposés

Location de véhicules : services exposés



Exemple : quels sont les services exposables ?

- A basic calculator for performing simple arithmetic operations (+, -, *, /)
- A printing application, shared by multiple applications, running in multiple environments
- A credit card authorization application
- A Database lookup that returns application-specific data
- A composite database lookup for customer information, searching across multiple databases

Quelles méthodes et outils permettent de mettre en oeuvre une architecture orientée services ?

Méthodes de conception des services

- SOMA (IBM)
- SODA (De Gamma)
- Praxeme (Unilog Management et Orchestra Networks)
- + toutes les formations proposées par les éditeurs tels que Softeam (SEA), DreamSoft, etc sur leur "savoir-faire"

➢ ***Autant d'offres que de méthodes différentes : de quoi s'y perdre !***

Modeleurs de processus

Outils de modélisation des processus métier

- **IBM WebSphere Business Modeler**
- **Bull Bonita**
- **De Gamma BPM**
- MEGA
- Aris
- Corporate Modeler
- WinDesign
- Power AMC
- Popkin System Architecture

Moteurs d'exécution de processus

- Plate-forme d'intégration
  - **IBM Websphere Process Server**
  - **BEA Weblogic Integrator/Acqualogic**
  - **Microsoft Biztalk**
  - **De Gamma Workflow**
  - Oracle BPEL PM
  - Bull Orchestra
  - SAP "Netweaver"
  - Apache ODE

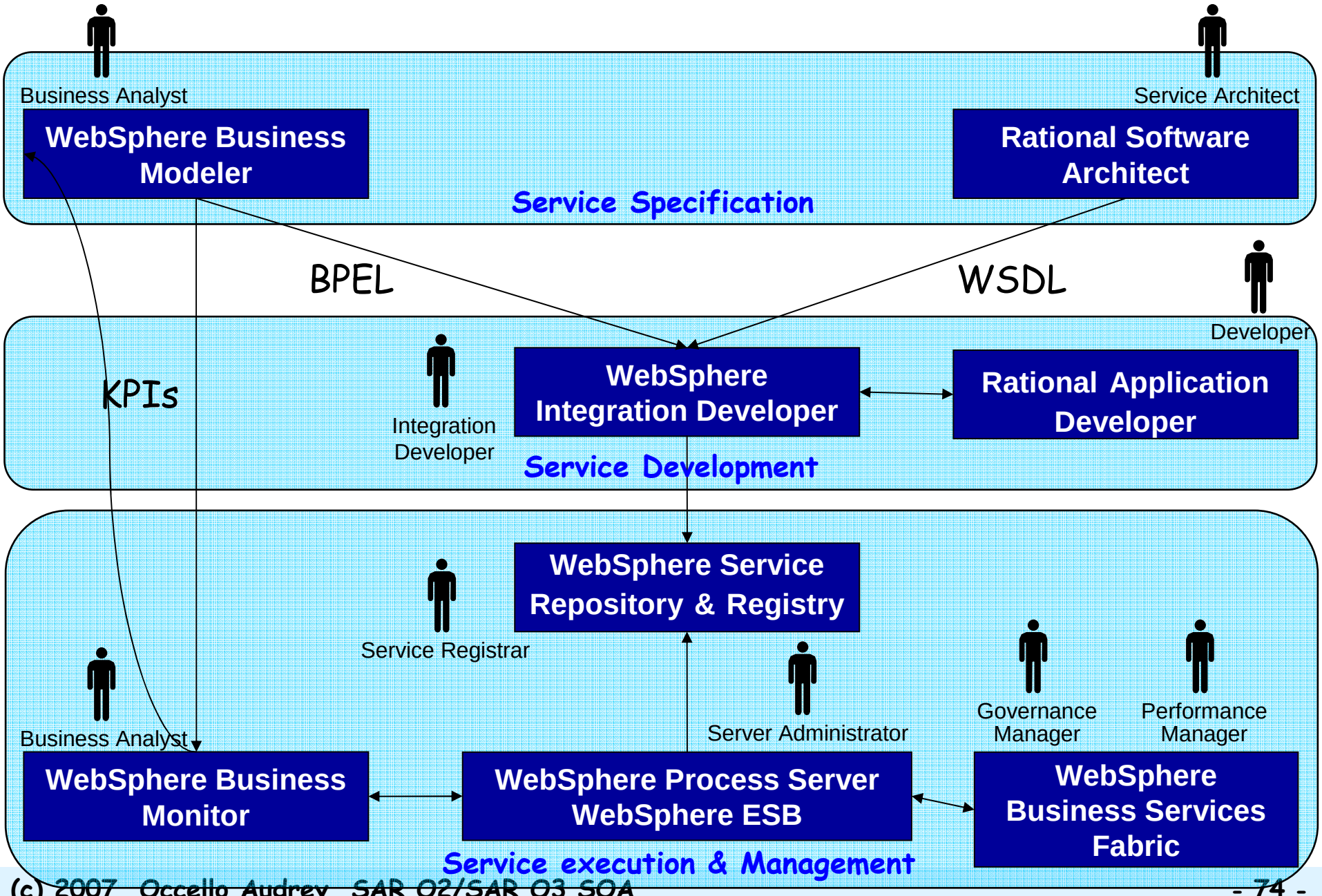
- ESB
  - **IBM Websphere ESB**
  - **Celtix hosted on ObjectWeb/IONA Technologies**
  - OpenESB (java.net)
  - Mule (codehaus.org)
  - Sonic ESB
  - EBM Web Sourcing Distributed Petals Bus (on OW2)

Contrôleurs/moniteurs

- BAM (Business Activity Monitoring)
  - **IBM WebSphere Business Monitor**
  - **Oracle BAM**
  - Systar Business Bridge
  - BMC Service Impact Manager

- Composants de sécurité
  - Oracle Web Service Manager
  - Oblix

Exemple: Gamme d'outils IBM couvrant le cycle de vie complet



Conclusions

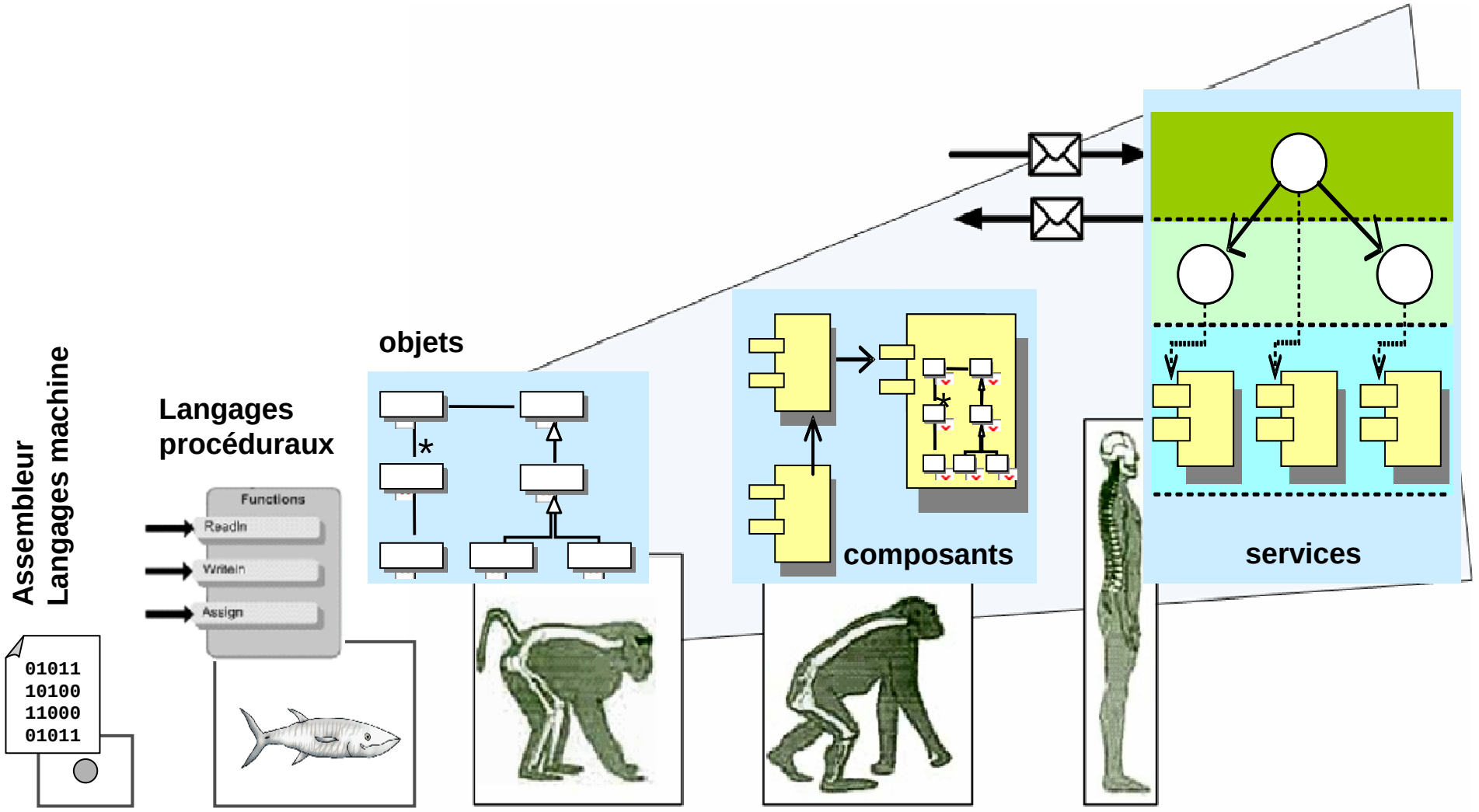
Du déjà vu ?

SOA est une évolution des plate-forme passées, tout en préservant les caractéristiques réussies des architectures traditionnelles

- Contractualisation des services
  - Design by Contract (Meyer)
- Découplage Interface/Implémentation, interopérabilité, transparence des communications, …
  - Middlewares à la CORBA
- Découplage fournisseur/comsommateur
  - Message Oriented Middleware (MOM)
- Orchestration des services
  - Travaux autour des workflows, langages de coordination

➢ ***SOA est une évolution plutôt qu'une révolution***

Chronique d'une évolution



➢ ***Niveaux d'abstraction grandissant***

Synthèse

HEVs
haute école valaisanne
hochschule wallis

Depuis...	...Vers...
• Orienté fonctionnalités • Conçu pour durer • Cycle de développement long	• Orienté processus • Conçu pour changer • Développement et déploiement interactif
• Silos applicatifs • Couplage fort • Orienté Objet	• Orchestration de Services • Couplage faible • Orienté message

Avantages et inconvénients

- Architecture adaptative
- Réutilisation du code
- Utilisation de standards
- Productivité accrue

- Manque de maturité des standards
- Lenteur d'exécution
- Difficile à effectivement implémenter
- Encore peu de chose sur la contractualisation

Paradoxe des principes fondamentaux

- Utilisation de standards
  - ***MAIS un standard reste un standard tant que tout le monde l'utilise*** (cf CORBA)
  - ***La course à la spécification fait rage***
    le W3C et l'OASIS se font la guerre
    - Spec des processus
    - Spec sur la sécurité
    - …
- Pas de remise en cause de l'existant lors d'évolutions technologiques
  - ***MAIS les vendeurs nous asservissent toujours avec leurs suites d'outils***

Paradoxe des principes fondamentaux

- Découplage entre fournisseurs et consommateurs de services
  - ***MAIS certains composants de services s'appellent directement au niveau du code: Couplage fort entre fournisseurs et consommateurs réintroduit par la couche IT***

- Indépendance des ressources vis à vis de ceux qui les utilisent
  - ***MAIS la gestion des données est encore peu prise en compte***

Quelques références ...

- "Urbanisation et BPM" - Yves Caseau, DSI Bouygues Télécom, Edition Dunod

- SOA à la sauce IBM
  http://www-306.ibm.com/software/fr/soa/

- SOA à la sauce Orchestra
  http://www.orchestranetworks.com/fr/soa/index.cfm

- CBM appliqué au scénario Rent-a-car
  http://www.research.ibm.com/journal/sj/444/cherbakov.html

Quelques références ...

- ***Composants***
  - CCM spec
    http://www.omg.org/cgi-bin/doc?ptc/2002-08-03
  - Fractal spec (GCM spec: proactive.inria.fr)
    http://fractal.objectweb.org/
  - Service Component Architecture (SCA)
    http://www-128.ibm.com/developerworks/library/specification/ws-sca/
  - OpenCCM
    http://openccm.objectweb.org/
  - Sofa
    http://dsrg.mff.cuni.cz/projects/sofa/tools/doc/compmodel.html