Ingénierie des Intergiciels : de l'Objet aux Services

Eric Gressier-Soudan

Introduction

Pour conclure le cours SMB111 :

Elaborer des applications distribuées, c'est aussi se poser des problèmes de génie logiciel en relation avec les plateformes d'exécution, on parle aussi d'ingénierie de systèmes.

C'est une discipline frontière entre systèmes distribués et génie logiciel.

Les différentes technologies vues en cours permettent de faire interagir des objets ou des services de façon distribuée.

Mais l'approche objets/services distribués ne suffit pas en elle-même. On voit bien des besoins supplémentaires par rapport à la gestion de codes, par exemple :

- Assemblage, gestion des dépendances
- Empaquettage
- Configuration,
- Déploiement
- Dynamicité

Les composants : un niveau d'abstraction de plus haut niveau

- Une abstraction de plus haut niveau, car l'objet (au sens CORBA 2.x) ne suffit pas.
- Meilleure encapsulation
- Formulation des interactions plus explicite
- Différencier la partie métier de la partie environnement/plateforme
- Couvrir le cycle de vie du code

Les réponses :

- Les Composants
- La programmation par Aspects
- L'approche Services
- L'approche MDE

Technologies à composants

Les langages de programmation supportés :
C++, C#, Java, éventuellement C mais moins naturellement

Liste non exhaustive :

- EJB, Java Beans, JXTA/JMX, OSGi
- CCM
- COM, DCOM, .NET
- Fractal, Kilim

Pour la conception : UML 2

D'où une grande diversité des modèles/du vocabulaire/ des concepts

Par exemple : composant CCM vs bundle OSGi

Important : modèles orientés application (ex : EJB) vs orienté middleware (ex : OSGi)

Consensus sur les définitions ?

La proposition OSI-RETINA : RM-ODP Qui a inspiré Fractal

Problématique à résoudre

Spécifier des comportements et des contraintes de QoS au niveau applicatif pour en déduire les contraintes sur l'architecture d'interaction sous-jacente !

Besoins :

Modèle de Description d'Application

Modèle de Spécification de contraintes

Modèle d'Architecture

Opérations de Gestion de contraintes de QoS

- **Spécification :** création d'un contrat entre l'application, et l'environnement d'exécution **(SLS)**

  Négociation: en vue d'obtenir un accord entre utilisateur et fournisseur **(SLA)**

- **Contrôle d'Admission :** tests qui déterminent si le système est capable de supporter le contrat requis

  Réservation de Ressources: pour garantir le contrat accepté

- **Surveillance :** surveillance par l'utilisateur du respect des contraintes de QoS qui ont été garanties par le fournisseur,

  Vérification : surveillance par le fournisseur du respect du contrat de QoS par l'utilisateur

  Maintenance : actions prises par le fournisseur en cas de défaut constaté sur la QoS qui a été garantie

- **Renégociation :** si la maintenance ne parvient pas à rétablir le niveau de service demandé, l'utilisateur doit pouvoir renégocier son contrat

Modèle de Spécification pour les applications : RM-ODP

RM-ODP: Reference Model of Open Distributed Processing

Il définit

une Architecture des Systèmes Répartis Ouverte et Hétérogène

Modèle Générique

fondé sur la notion de **Points de vue** (aspects d'analyse et spécification) :

- Entreprise (besoins des applications, contraintes opérationnelles et organisationnelles)
- Information (modèle des données)
- Traitements (modèle des traitements sous forme d'Objets ou de composants qui interagissent)
- **Ingénierie** (objets, OS, protocoles, liens réseau, contraintes sur l'infrastructure logicielle et matérielle ...)
- Technologie (implantation et conformité à la spécification...)

Approche Orientée Objet

Une plate-forme de type CORBA satisfait le Point de Vue de l'Ingénierie de RM-ODP.

Rappels sur l'Orienté Objet

Objet :

- Etat + Données Internes (inaccessibles de l'extérieur)
- Opération = point d'accès à une fonction exécutable par un objet
- Interface regroupe des opérations, sert à la désignation des Opérations

Signature d'une opération : nom d'opération + paramètres + type du résultat

Un objet n'évolue que par l'utilisation des opérations de son interface.

Propriétés des objets :

. Encapsulation
. Héritage
. Polymorphisme
. Agrégation

Avantages :

. Séparation entre spécification et implantation
. Modularité et Extensibilité
. Réutilisabilité des composants
. Adéquation des abstractions : Analyse, Spécification, Conception, Programmation

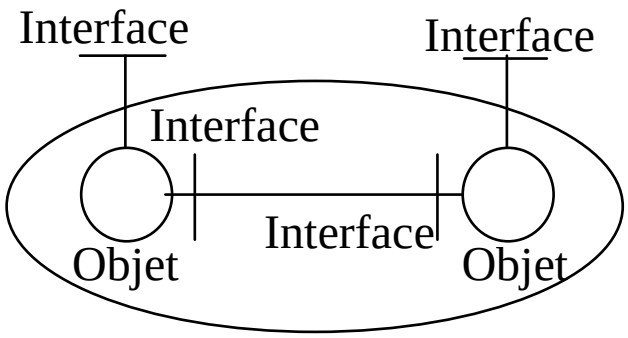
Modèle Objet d'ODP

Modèle objet des langages de programmation
+ plusieurs interfaces par objet
+ notion d'objet composite (réparti)

une interface = une vue abstraite de l'objet (concept de rôle)

exemple : une interface d'accès aux opérations, une interface pour les opérations de gestion

Schématisation d'un objet RM-ODP :



Objet Composite

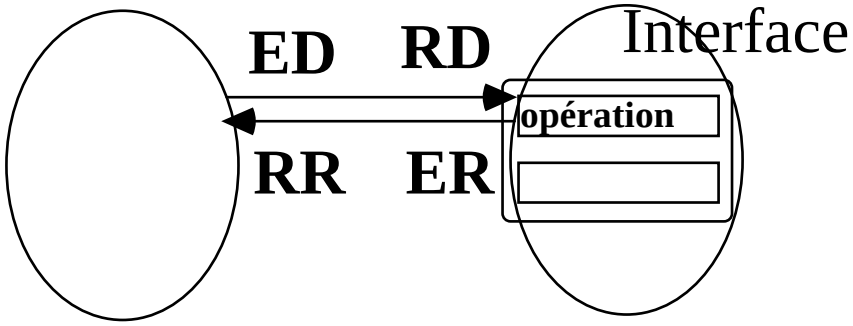
Modèle des Traitements de ODP

Repose sur la notion d'Objets en environnements répartis.

Application = ensemble d'objets

Interaction = Invocation d'Opération, ou Réaction d'Invocation décrite à l'aide d'événements

types d'événements



ED = Emission de Demande; RD = Réception de Demande
ER = Emission de Réponse; RD= Réception de Demande

Chaque événement a :

- un Type,
- un Nom,
- une Valeur

Spécification de Contraintes de QoS

Modèle des Traitements :
=> Pouvoir Spécifier des Comportements et des Propriétés
=> Vérifier les Spécifications

Règles de Conformités exprimées sur les événements, donc sur les interfaces et sur les communications entre objets

Exemples :
- règle comportementale : ED et RR sur un objet Client
- règle temporelle : le délai entre RD et ER est Δ

Qualité de Service orientée performance:
- taux d'erreur message,
- taux d'erreur bit,
- débit,
- garantie de délai
...

On a bien une notion de **Contrat** associé à une interface.

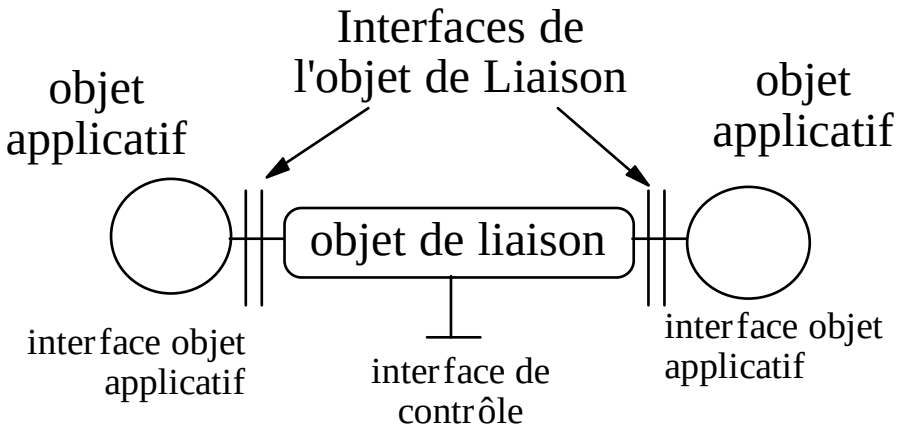
La QoS garantie par un objet implique la possibilité de supporter des propriétés de QoS par l'environnement d'exécution.

Liaisons entre Objets

Une Liaison correspond à l'abstraction du mécanisme qui supporte une interaction entre Objets.

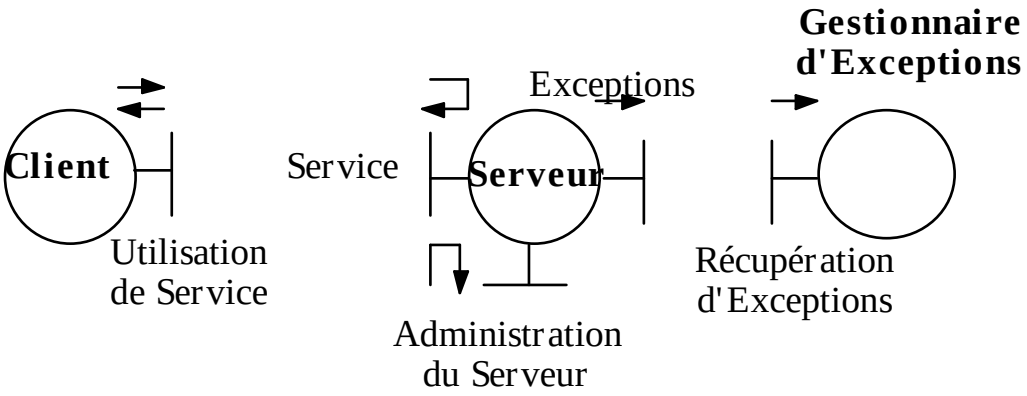
Les contraintes de QoS sur les interactions entre Objets imposent des contraintes de QoS sur les Liaisons.

Les Liaisons deviennent des Objets avec une Interface de Contrôle qui permet de les commander, de les superviser, de les configurer.



On voit ici l'extension de la notion de connexion ou d'association qu'on trouvait dans les couches du modèle ISO. C'est aussi le binding/la liaison de CORBA.

Modèle d'objets de base



correspond aux objets dans CORBA

Il existe des objets particuliers qui ont la charge d'ajouter des nouveaux objets à l'environnement. Ces objets particuliers s'appellent des **fabriques à objets** (object factory).

Les objets sont fabriqués à partir de modèles (template) complètement spécifiés y compris le contrat de QoS, c'est un processus d'instanciation.

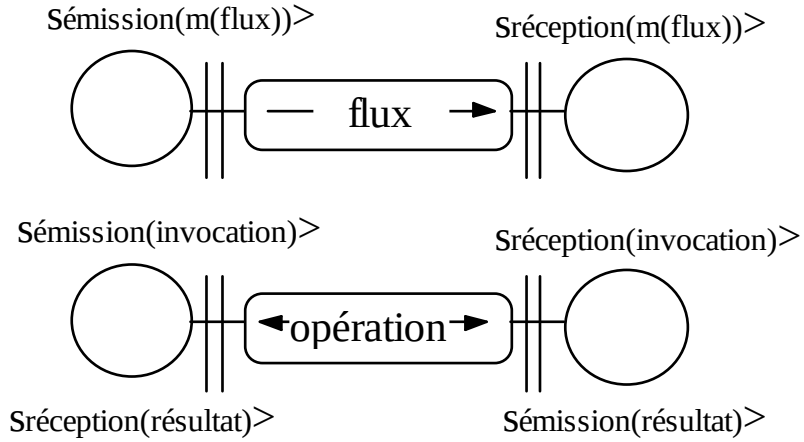
Extensions Multimédia du Modèle Objet

Interface production et consommation de flux

Une interface de type flux supporte des interactions multimédia de type continu, elle est définie en terme d'un ou plusieurs flots de données. Chaque flot est caractérisé par un nom, un type de media géré, et la direction du flot.

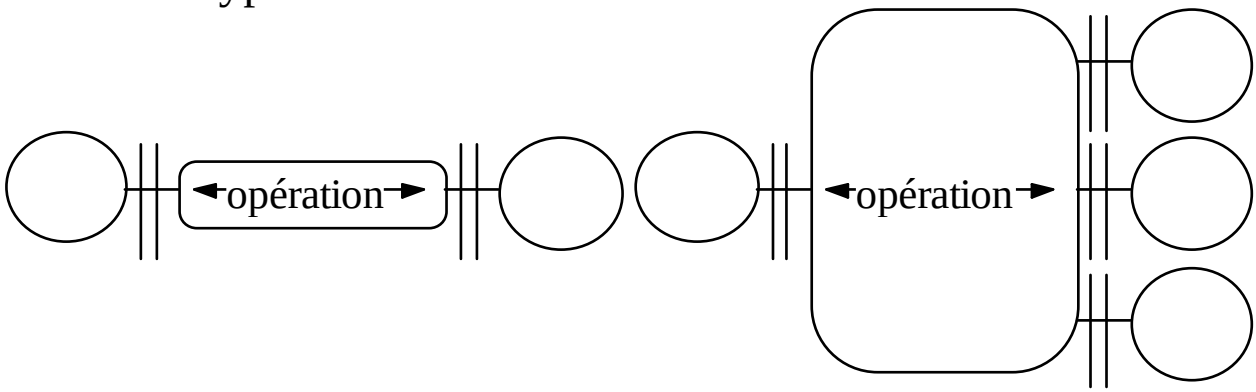
Interface signaux ou événements temps réel

Les interfaces de type signal sont utilisées pour la gestion de la QoS et la synchronisation temps réel. Chaque signal dans un interface est représenté par un nom, le type, sa valeur, et la causalité induite (générateur, consommateur).
Les signaux peuvent être utilisés aussi bien pour des flux que pour des services, dans chaque cas ils représentent l'émission ou la réception d'un message.

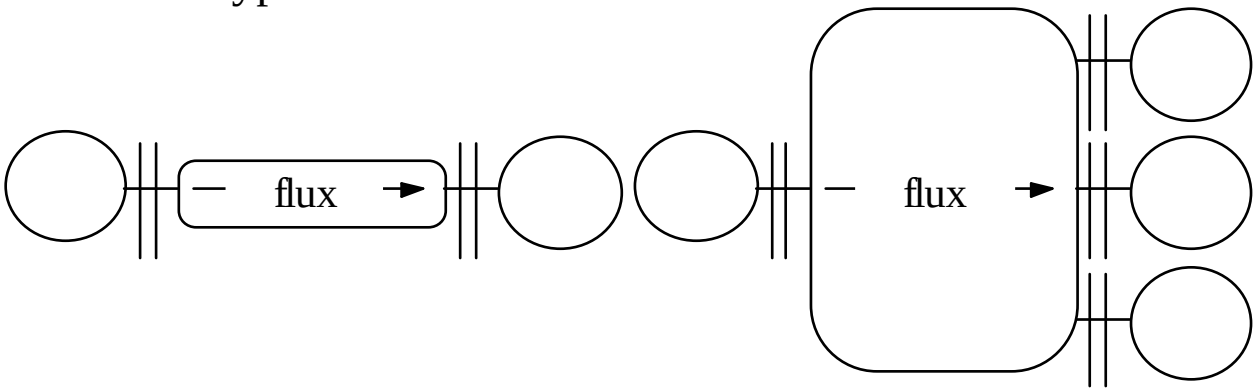


Modèles de Liaisons entre Objets

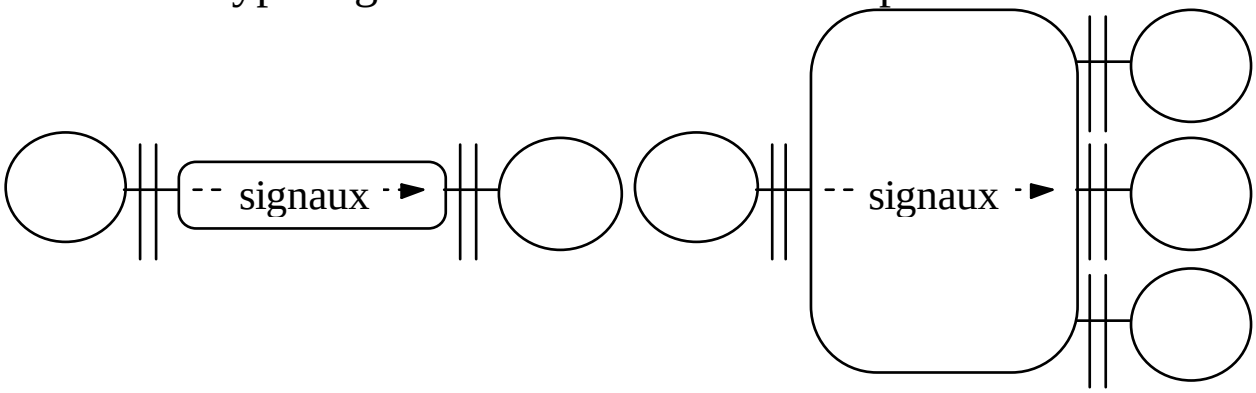
Liaison de type Service



Liaison de type Flux



Liaison de type Signaux ou Evénements Temps Réel



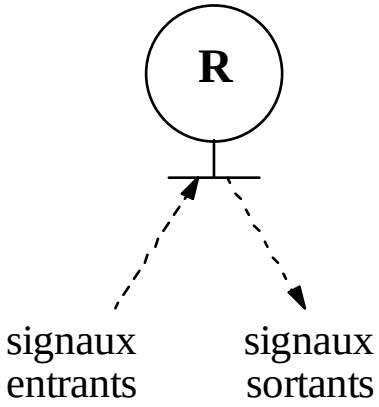
Objets réactifs

Les objets présentés sont des objets asynchrones. Objets ou objets de liaison, ils prennent un certain temps pour exécuter leur traitement.

La spécification des contraintes de QoS temporelle sert à contraindre le comportement des objets, et à édicter des contraintes environnementales sur l'architecture.

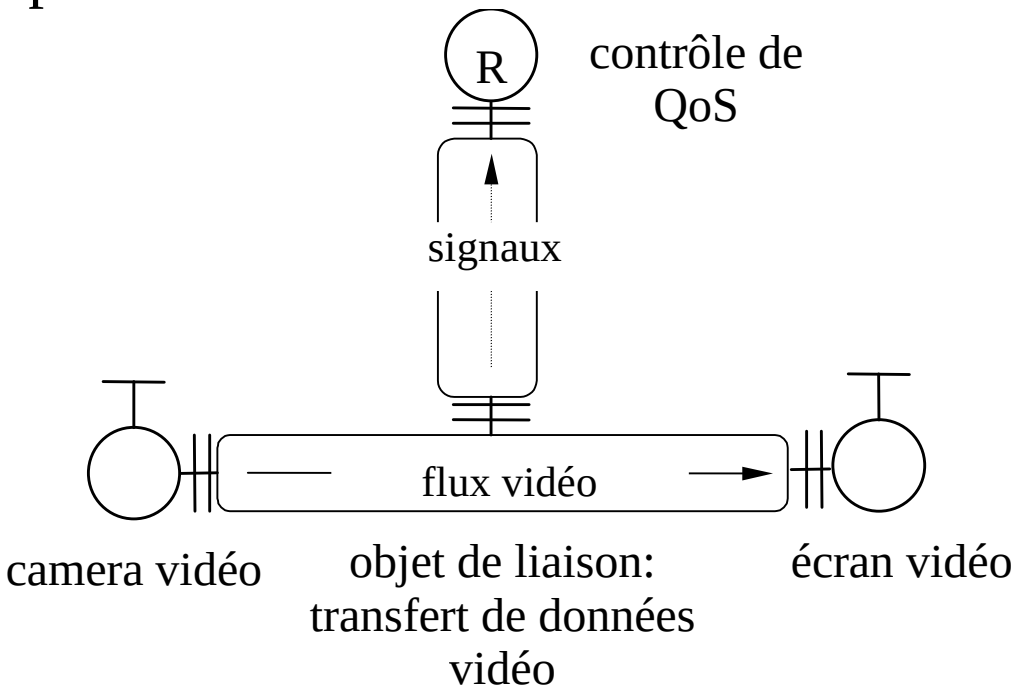
Les Objets Réactifs ont un rôle particulier, ils servent à contrôler le comportement d'objets, en particulier à observer l'évolution de la QoS ou à gérer la synchronisation temps réel. En ce sens, ils reçoivent des signaux, les traitent et envoient des signaux.

Les objets réactifs ont un comportement synchrone. Ils s'exécutent en un temps nul (hypothèse de modèle, en réalité c'est un temps borné inférieur à une période d'activation). Ils sont programmés en langage synchrone (LUSTRE, SIGNAL, ESTEREL).



Exemple de schématisation d'une application Multimédia

Exemple de modélisation des traitements :



Echange Vidéo avec Moniteur de QoS

Temps Réel Multimédia = QoS + Contrôle

- Objets applicatifs ou système (asynchrones contraints par une spec de QoS) = > Traitements
- Objets réactifs (à exécution immédiate) =>Contrôle

Avantages :

. Les contraintes temporelles sont formalisées explicitement par des équations de QoS

. Claire séparation entre le contrôle et l'application

. Portabilité, seule les équations de QoS doivent être recalculées avec un nouvel environnement

Equations de QoS

La spécification d'équations de QoS (ou plutôt d'inégalités) est fondée sur un modèle événementiel (suppose donc un temps discret). Exemples avec expression déterministe :

Latence bornée d'un transfert d'images vidéo :

pour tout n, (n représentant l'occurence d'un événement) :

|date(réception-image-video,n) - date(émission-image-video,n)|
<= latence

Gigue d'un transfert d'images vidéo :

pour tout n :

min<=|date(réception-image-video,n)-date(émission-image-video,n)|<=max

gigue = max - min

Débit max d'un transfert d'images vidéo :

pour tout n :

|date(réception-image-video,n+k)- date(réception-image-video,n)|<= durée

débit = k/durée, en images par seconde

Taux de perte d'un transfert vidéo :

pour tout n :

(|date(émission-image-video,n+k)-date(émission-image-video,n)| <=d)
et
(|date(réception-image-video,n+k')-date(réception-image-video,n)| <= d)
et
(k' <= k)

taux de perte 1-(k'/k)

Exemple 1/2

Dans la réalité un événement est repéré par :
nom-d'une-interface.nom-d'un-signal.causalité avec causalité pouvant prendre les valeurs (ED/ES[1],RD/RS,[ER,RR] conformément au modèle précédent)

Exemple du transfert vidéo entre une caméra et un écran

Objets :
Caméra avec l'interface vidéoOut, et écran avec l'interface vidéoIn

L'expression de la contrainte "latence bornée" devient :
pour tout n :
|date(écran.vidéoIn.RS,n)
- date(caméra.vidéoOut.ES,n)| <= 10ms

La spécification de paramètres de QoS est construite en indiquant, des clauses requises (Req) et des clauses fournies (Prov) , avec la relation Req(A) -> Prov(A).

[1] D pour Donnée et S pour Signal, dans le cas des signaux, il n'y a pas de réponse (R)

Exemple 2/2

Objet Liaison transfertVideo :

```
// interface avec la caméra
interface flux transfertVideoIn {
        fluxEntrant videoIn (video) ;
}
//interface avec l'écran de restitution
interface flux transfertVideoOut {
        fluxSortant videoOut(video);
}
//interface contrôle de QoS
interface signal qosControle {
        signalOut videoEmis (date);
        signalIn videoDélivré (date) ;
}
```

Clause Fournie
//tranfert soutient 25 images/s avec une latence entre 40 et 60 ms

pour tout n, date(transfertVideoOut.videoOut.ES, n+25)

<= date (transfertVideoOut.videoOut.ES, n) + 1000 ms

et pour tout n,

40ms<=

|date(transfertVideoIn.videoIn.RS,n)-date(transfertVideoOut.videoOut.ES,n)|

<=60ms

et pour tout n, date(qosControle.videoEmis.ES,n)

= date(transfertVideoIn.videoIn.RS, n)

et pour tout n, date(qosControle.videoDélivré.RS,n)

= date(transfertVideoOut.videoOut.ES, n)

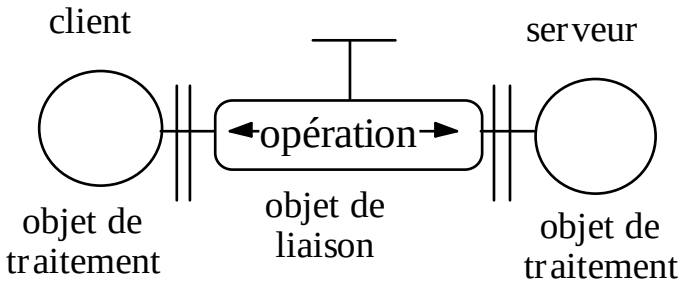
Clause Requise
// si la caméra envoie 25 images/s

pour tout n, date(transfertVideoIn.videoIn.RS, n+25)

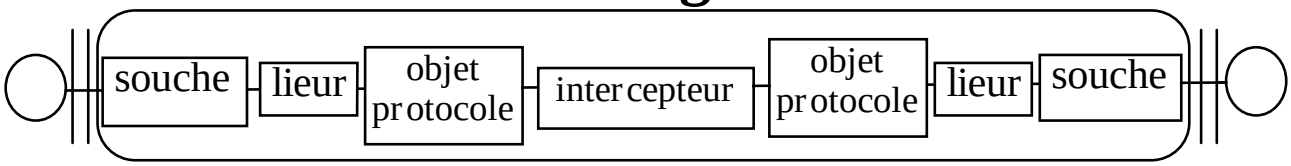
<= date (transfertVideoIn.videoIn.RS, n) + 1000 ms

Point de Vue de l'Ingénierie

Modèle des traitements :

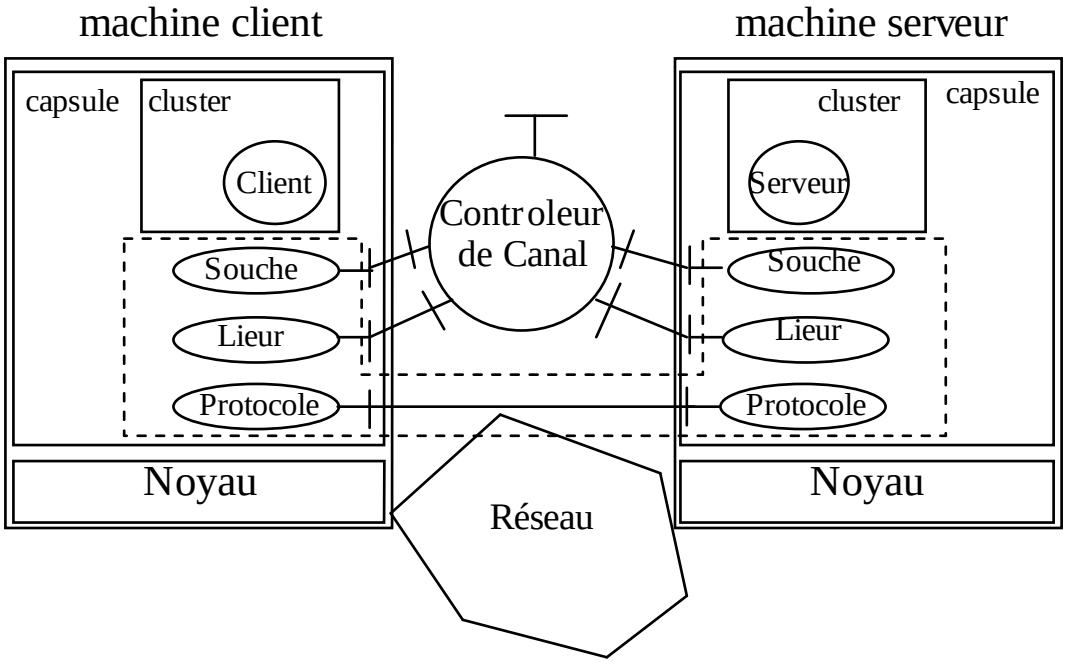


Modèle d'ingénierie :



Canal

intercepteur = fonction équivalente à pont CORBA/passerelle (conversion de protocole)



Les équations de QoS peuvent se projeter sur chacun des composants d'ingénierie, pas seulement le réseau.

Points Délicats d'une architecture à QoS

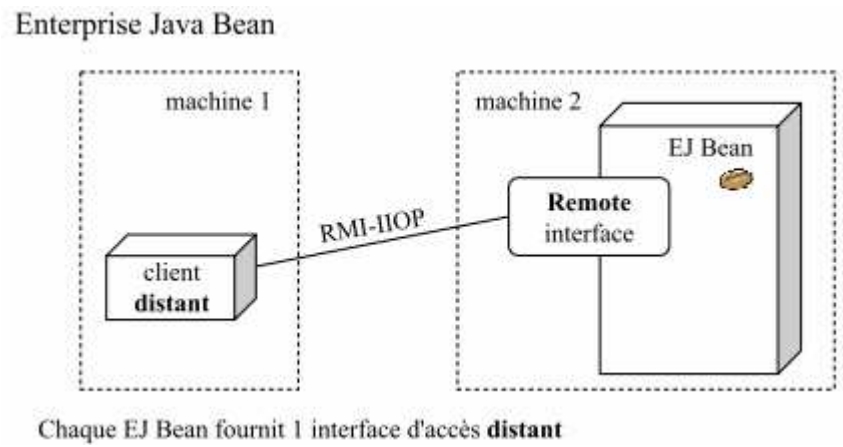
La spécification de contrat de QoS et le SLA qui en découle ne sont qu'un début.

Ce qui devient clef, c'est le suivi minute après minute de la QoS offerte par le réseau :

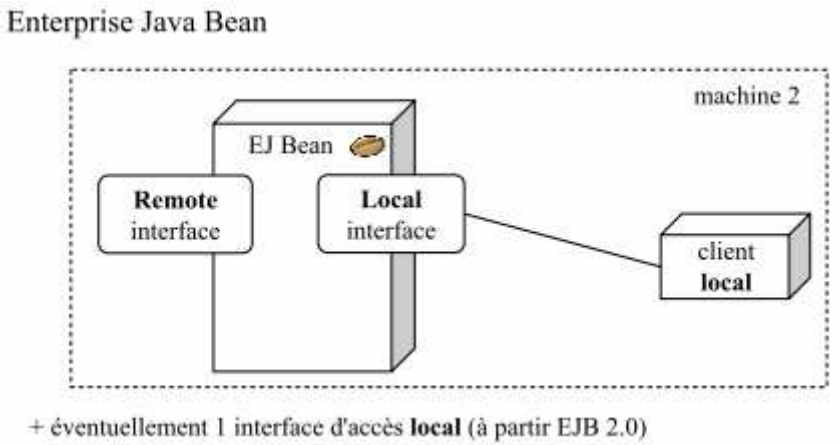
- Outils de mesure,
- Aide à l'analyse,
- Supervision et maintenance de la QoS

Il existe un besoin d'outils déterminants.

Une autre sorte d'influence : EJB (d'après le support EJB de Lionel Seinturier LIFL, *http://www2.lifl.fr/~seinturi/middleware/ejb3.pdf*)



Chaque EJ Bean fournit 1 interface d'accès **distant**

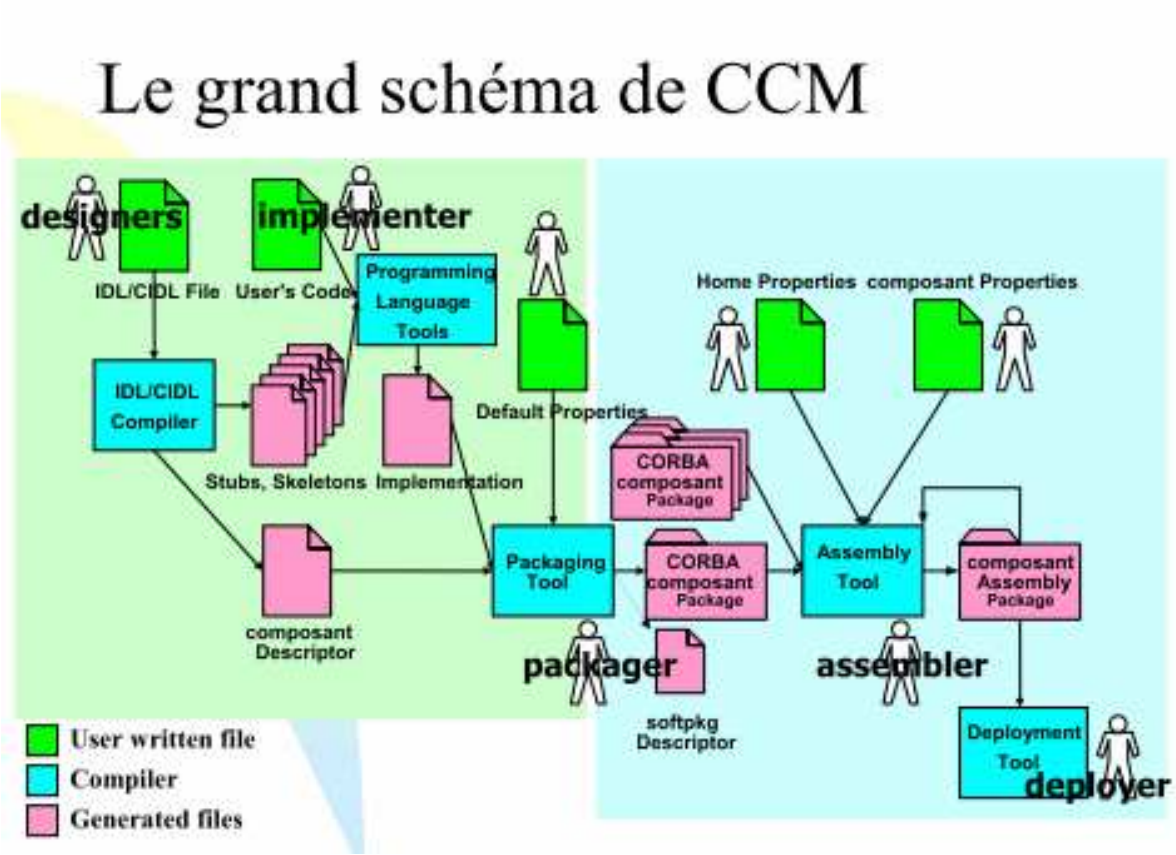


+ éventuellement 1 interface d'accès **local** (à partir EJB 2.0)

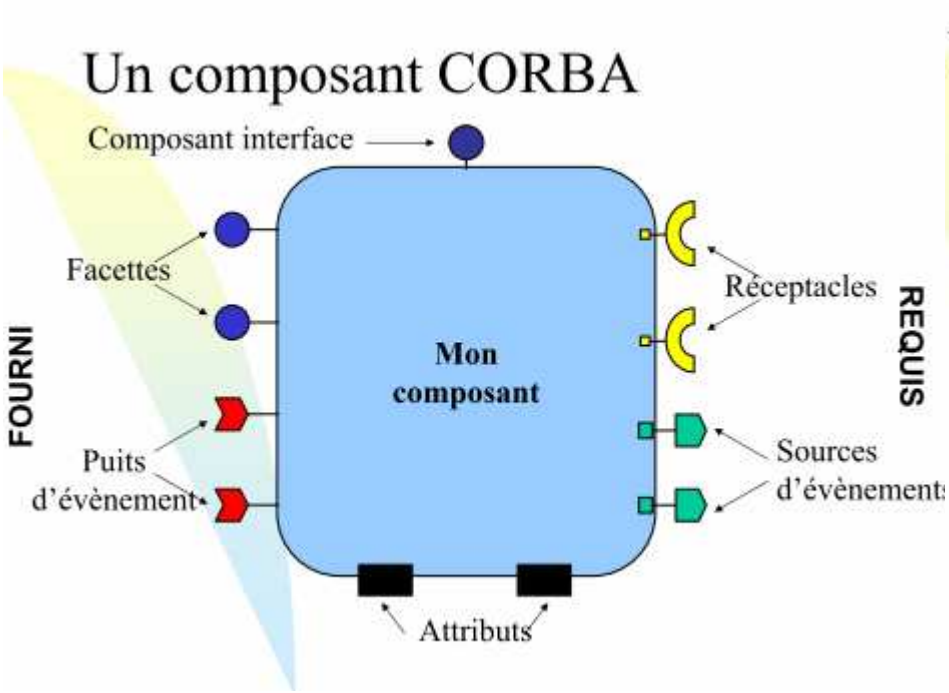
L'interface locale offre les mêmes services aux programmes locaux que distants.

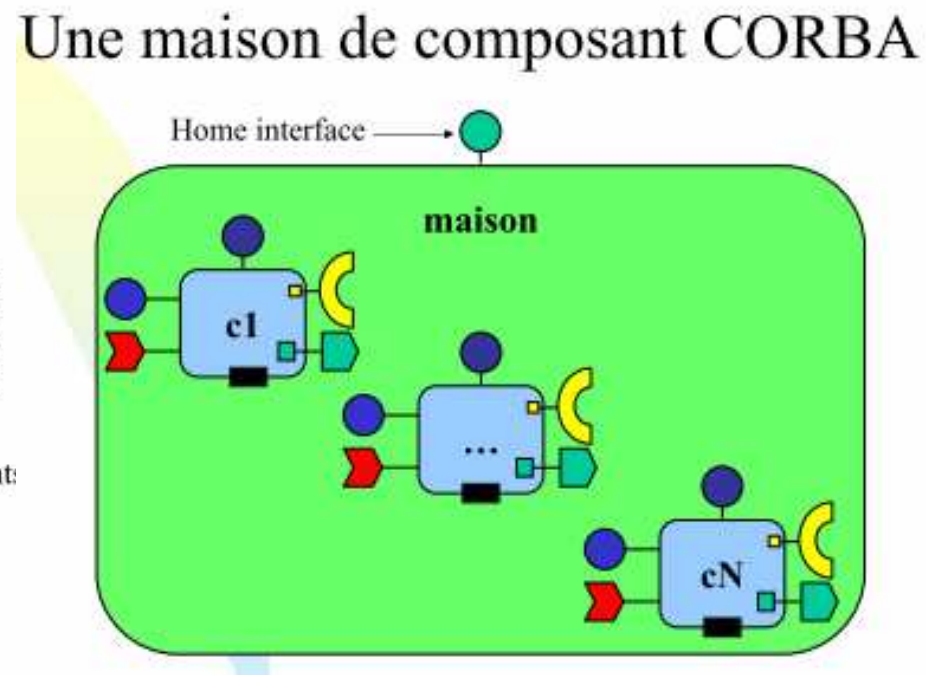
Une autre sorte d'influence : OMG CCM (d'après le support CCM de Lionel Seinturier LIFL, *www2.lifl.fr/~seinturi/middleware/ccm.pdf*)

CORBA Component Model : OMG OBJECT MANAGEMENT GROUP

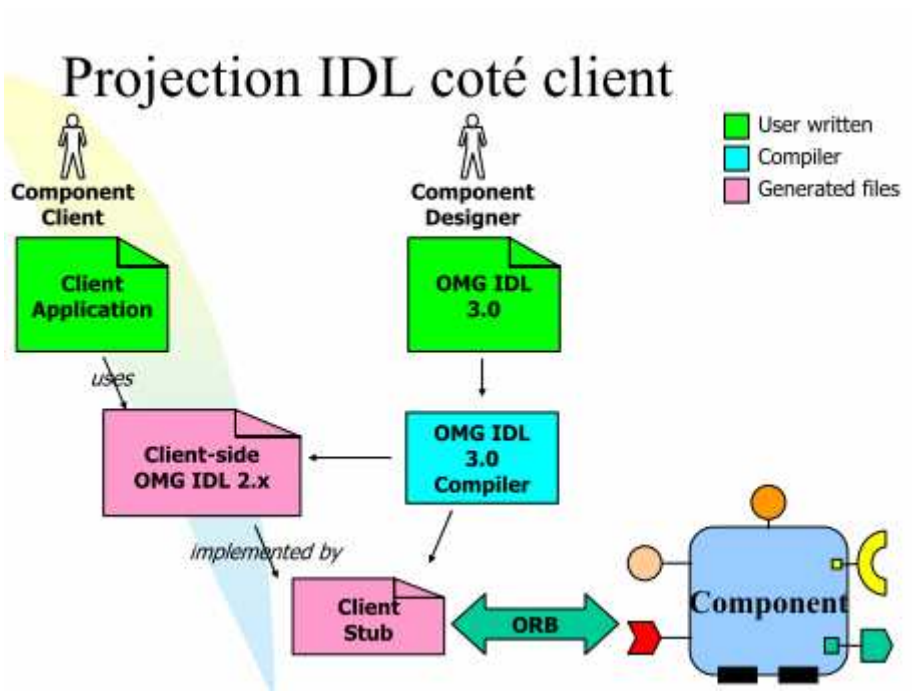


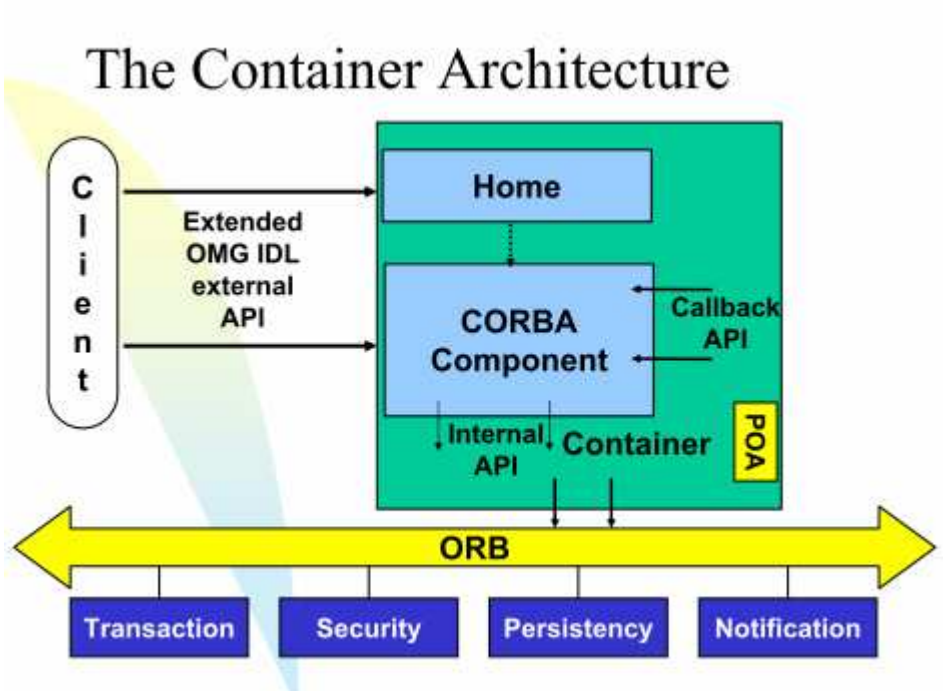
Modélisation du Composant CORBA





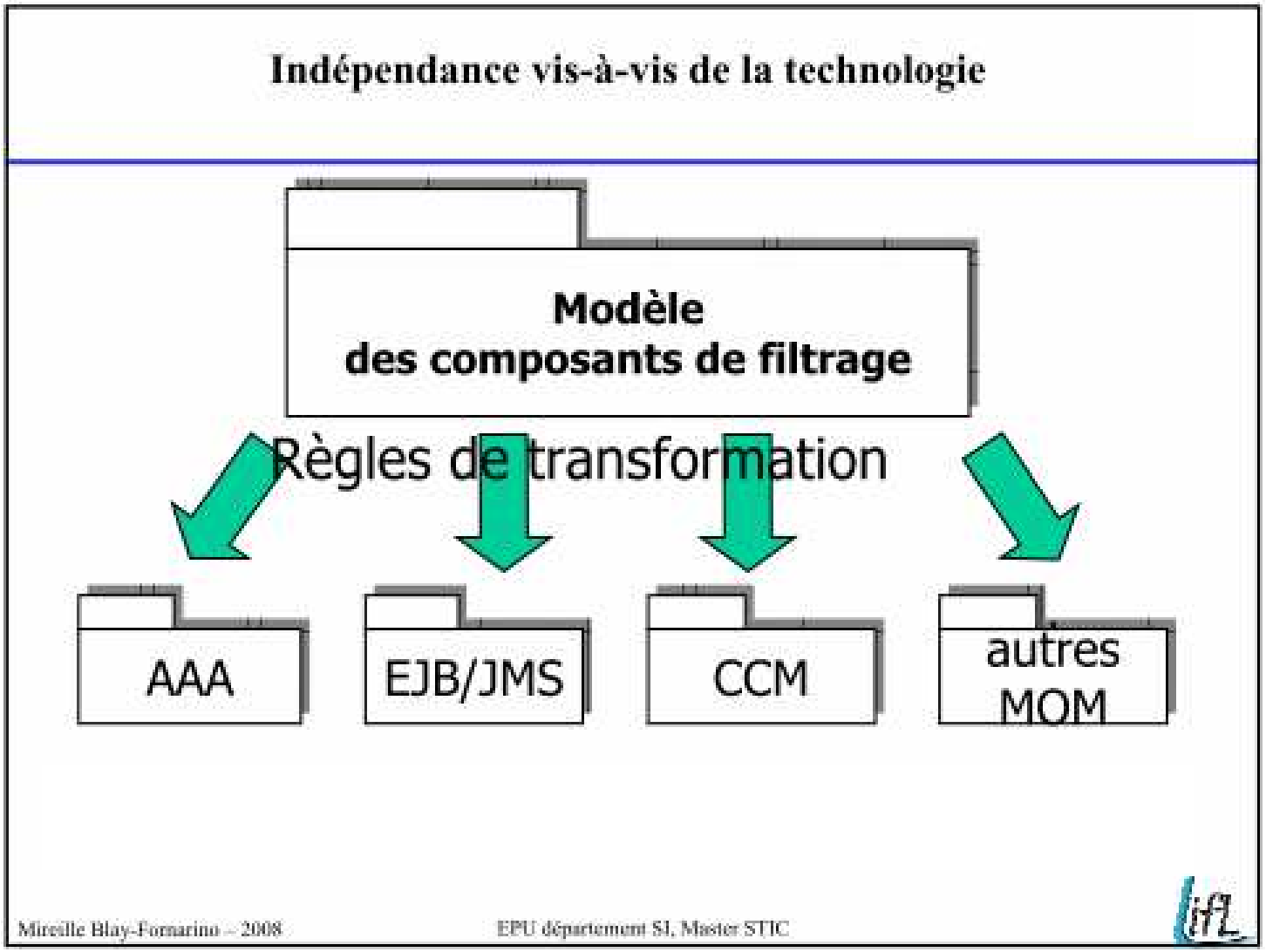
Le tout s'exécute sur l'ORB CORBA 2.x





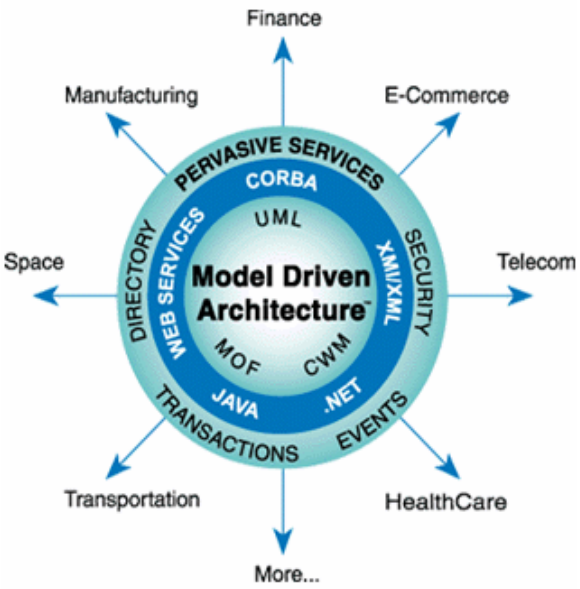
Les EJB ont aussi eu une influence sur le génie logiciel des middlewares.

Principe de Réalité



Une sorte d'influence : l'approche MDA

MDA : Model Driven Architecture



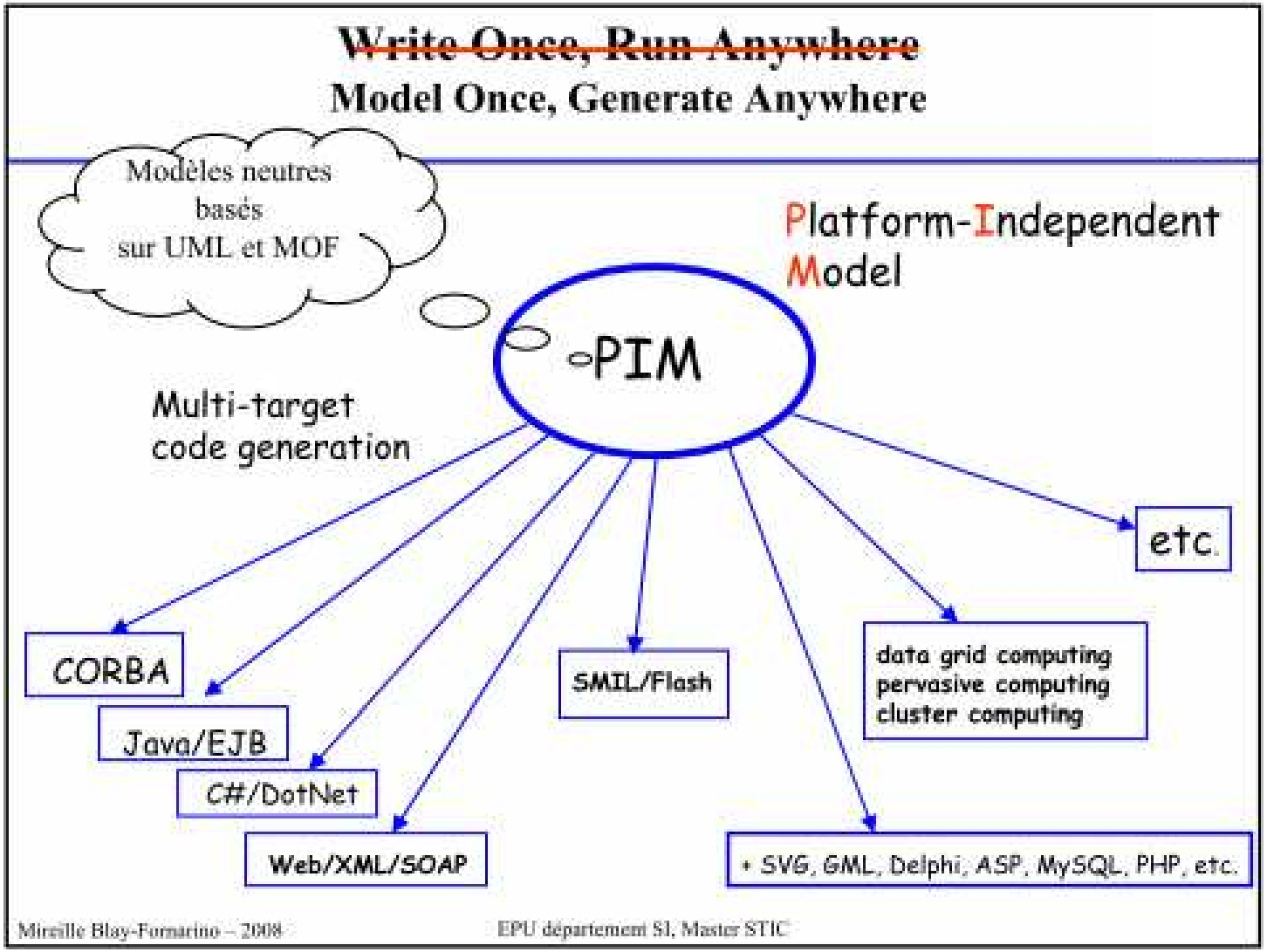
Objectifs : Modéliser (Méta-modélisation)

- se concentrer sur la partie métier, qui crée de la valeur ajoutée :

  PIM (Platform Independant Model)

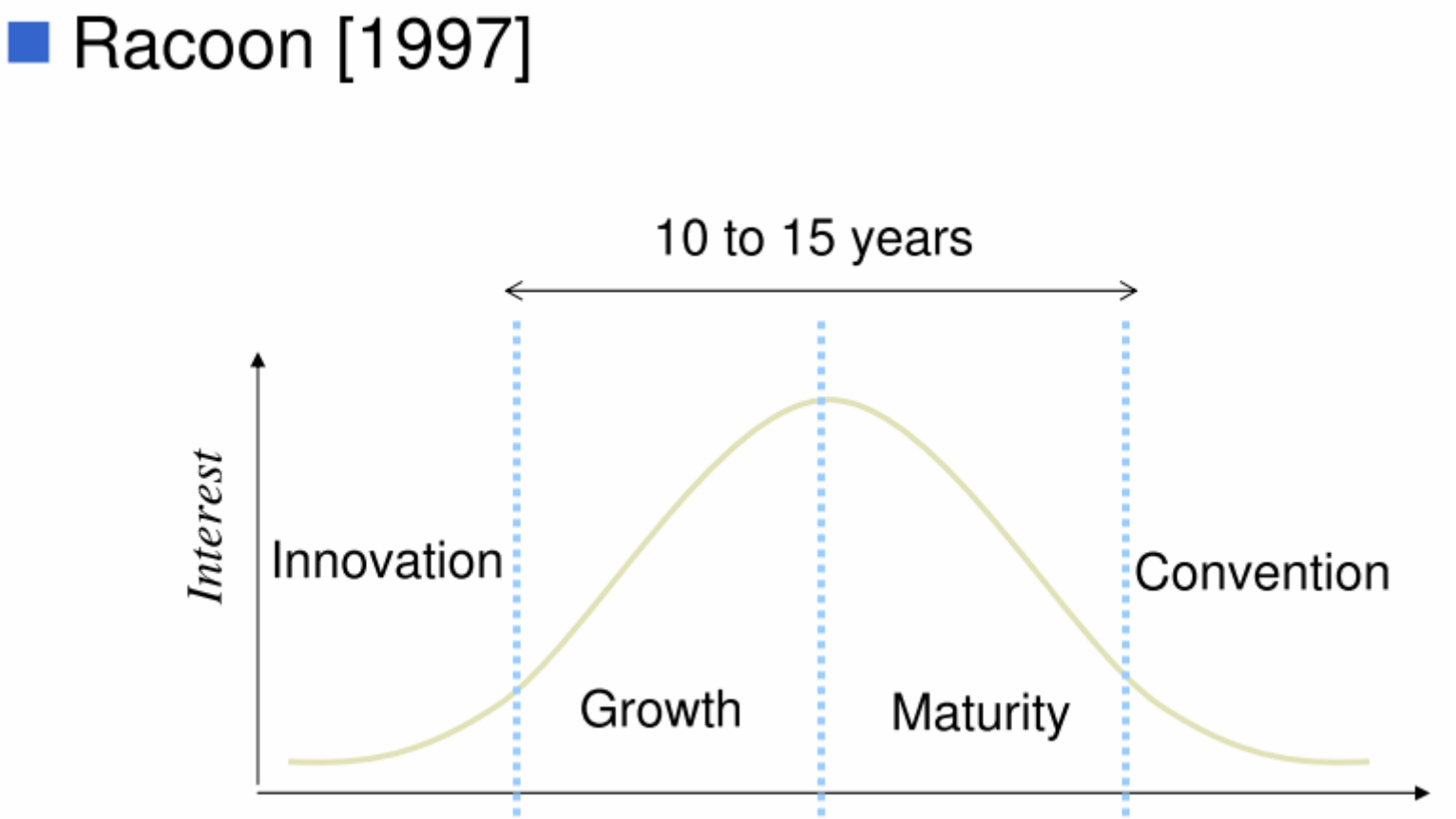
- générer automatiquement le code pour la plateforme d'exécution à partir de modèles abstraits de plateformes

  PSM (Platform Specific Model)


Write Once, Run Anywhere
Model Once, Generate Anywhere
Modèles neutres
basés
sur UML et MOF
Platform-Independent
Model
PIM
Multi-target
code generation
CORBA
Java/EJB
C#/DotNet
Web/XML/SOAP
SMIL/Flash
data grid computing
pervasive computing
cluster computing
etc.
+ SVG, GML, Delphi, ASP, MySQL, PHP, etc.
Mireille Blay-Fornarino – 2008
EPU département SI, Master STIC

Approche SOA (d'après le cours SOA de D. Donsez)

- Racoon [1997]



Le modèle des vagues appliqué aux paradigmes de programmation 1/2

- Racoon [1997]

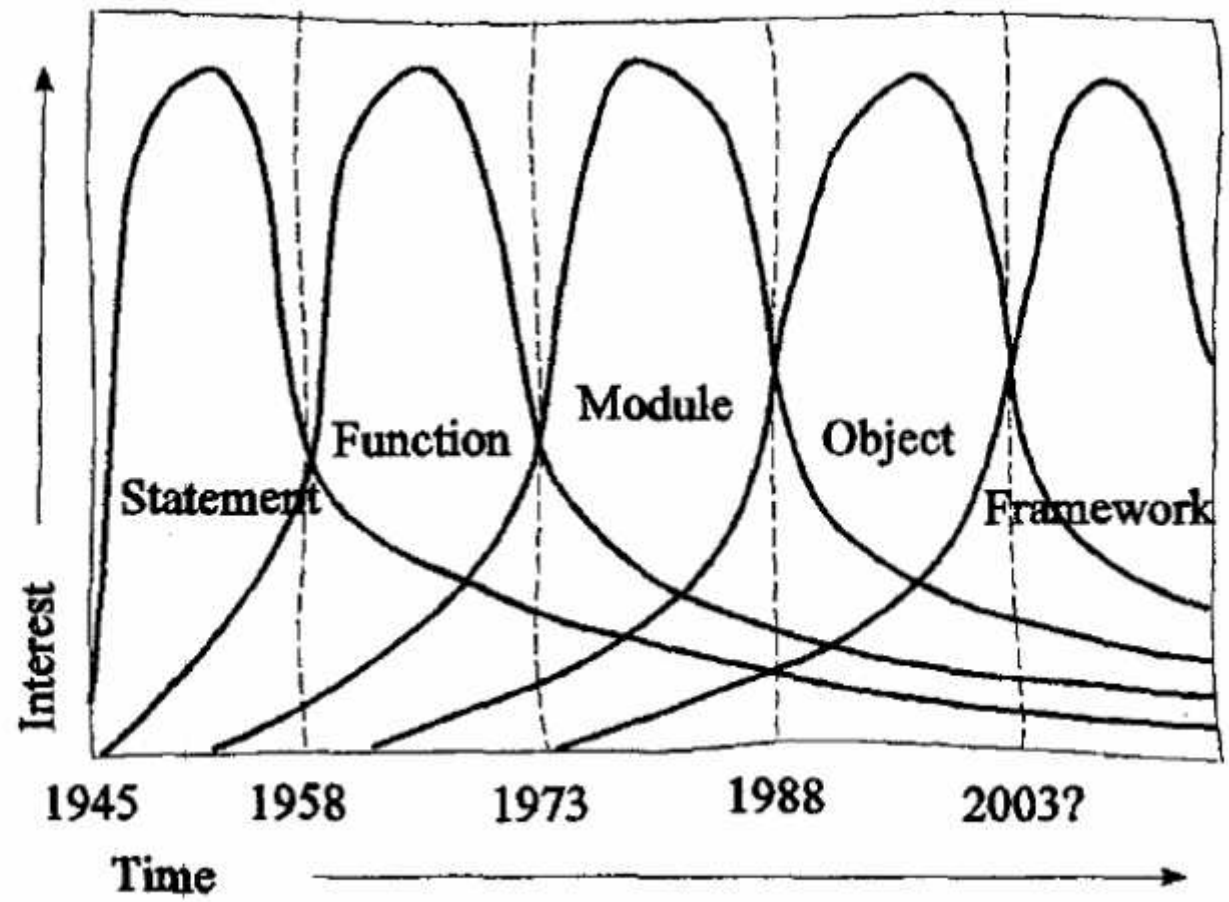
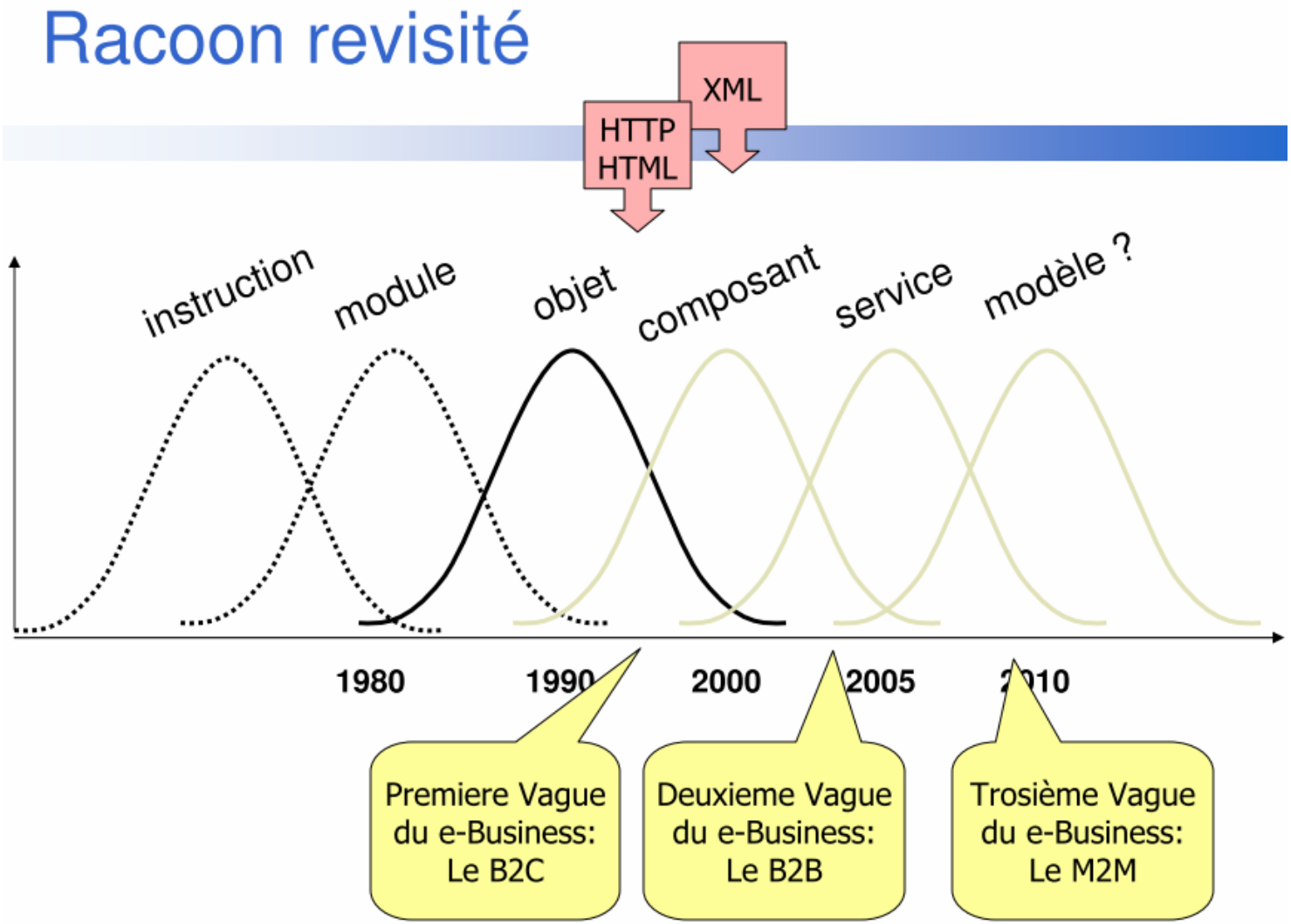
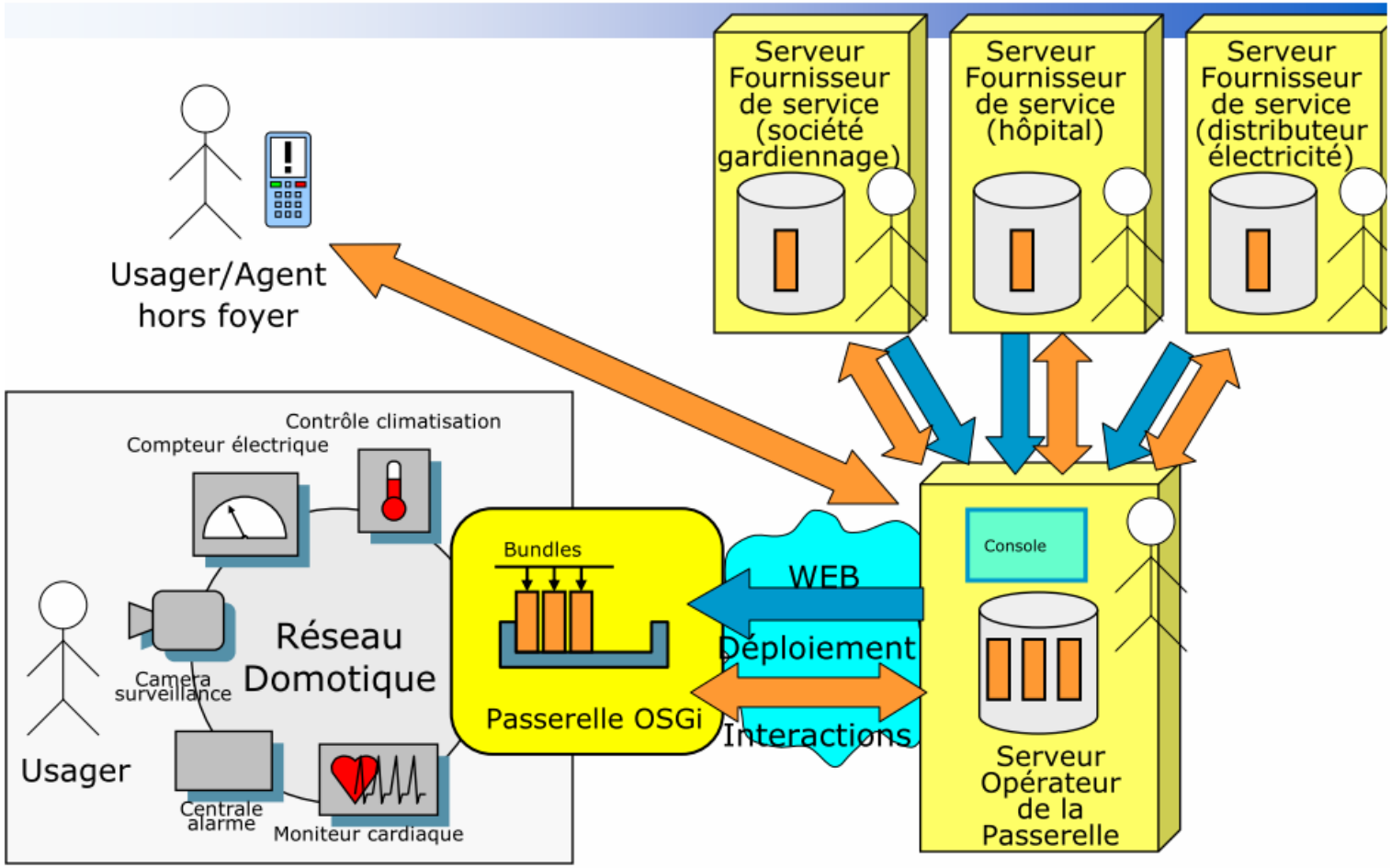


Figure 2: The Organization Stream.

Le modèle des vagues appliqué aux paradigmes de programmation 2/2

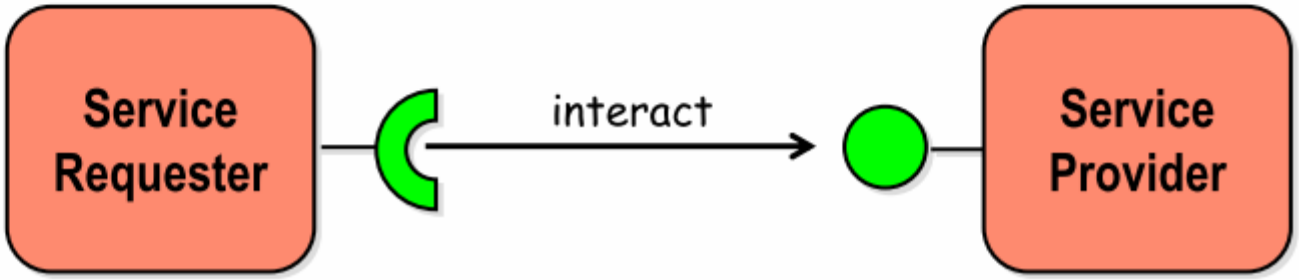


Exemple de Scénario de M2M

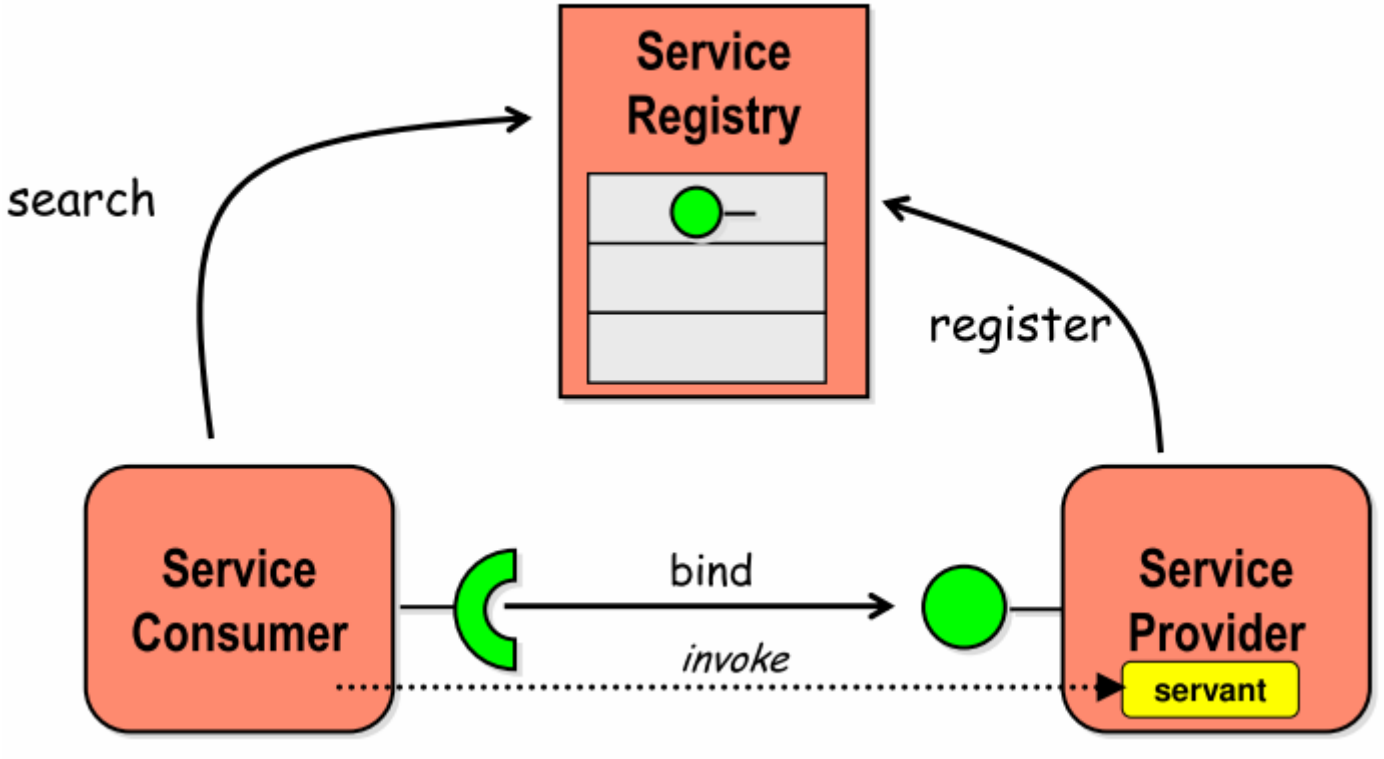


Service
Définition

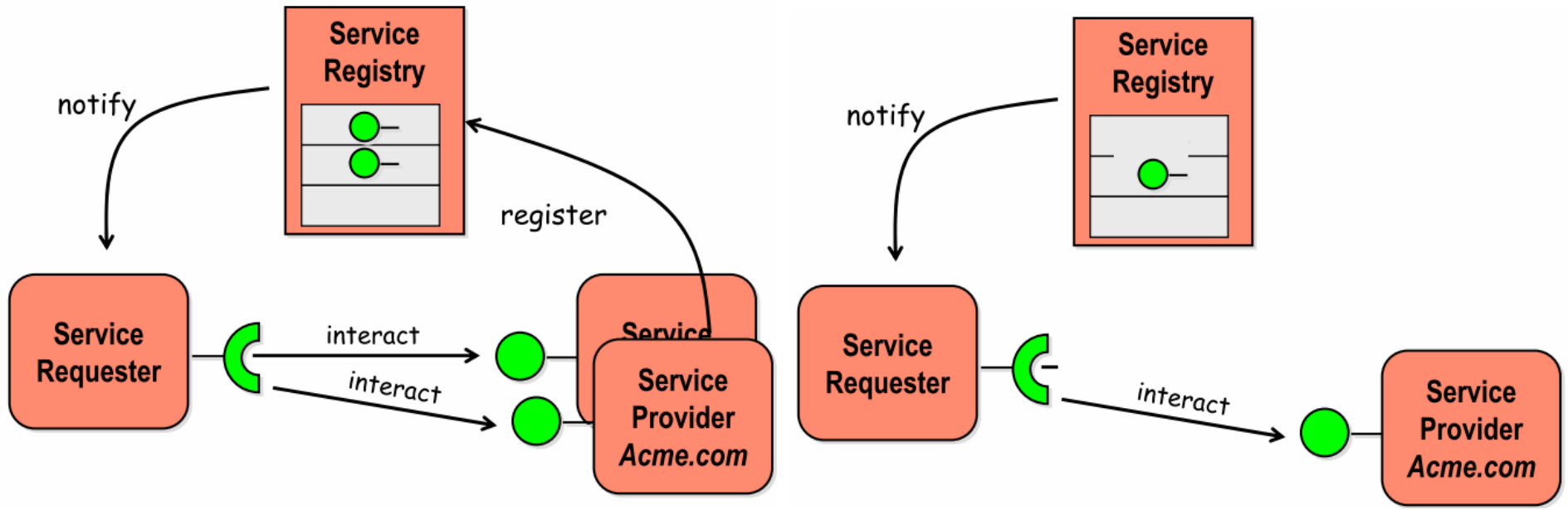
- « un **service** est un **comportement** défini par **contrat**, qui peut être réalisé et **fourni** par tout *composant* pour être **utilisé** par tout *composant*, sur la base unique du **contrat** »
[Bieber and Carpenter 2002].



On retrouve le schéma bien connu



avec édition de liens dynamiques



OSGi : univers Java à composant - centralisé

- **Spécification OSGi**
  - définit un canevas de déploiement et d'exécution de services Java
  - multi-fournisseur, télé-administré
  - Cible initiale : set top box, modem cable, ou une passerelle résidentielle dédiée.

- **OSGi Alliance**
  - Corporation indépendante
  - Soutenus par les acteurs majeurs des IT, home/building automation, telematics (car automation), …
  - de la téléphonie mobiles (Nokia et Motorola)
  - et Eclipse pour les plugins de son IDE !
  - et maintenant Apache pour ses serveurs

27 membres (Full Members), http://www.osgi.org/About/Members et beaucoup de participants !

Propriétés principales d'OSGi

- **Modularisation des applications**
  - Chargement/Déchargement de code dynamique
    - Langage Java
  - Déploiement dynamique d'applications sans interruption de la plateforme
    - Installation, Lancement, Mise à jour, Arrêt, Retrait
    - *« No reboot »*
  - Résolution des dépendances versionnées de code
- **Architecture orientée service**
  - Couplage faible, late-binding
  - Reconfiguration dynamique des applications (plugins, services techniques)
- **Vise des systèmes à mémoire restreinte**
  - s'accroche à J2ME/CDC
  - même si de plus en plus Java Platform 1.5, 6, 7, …

Les concepts principaux d'OSGi

- Framework:
  - Environnement d'exécution pour bundles
    - Oscar (Objectweb) / Felix (Apache), Knopperfish, Equinox, SMF, ProSyst, Siemens VDO, …
  - Notification d'événements
- Bundles:
  - Unité d'exécution et de déploiement
- Services:
  - Objets Java implantant un contrat