

<http://www2.lifl.fr/~seinturi>
<http://membres-liglab.imag.fr/donsez/cours>

Ingénierie logicielle à base de composants

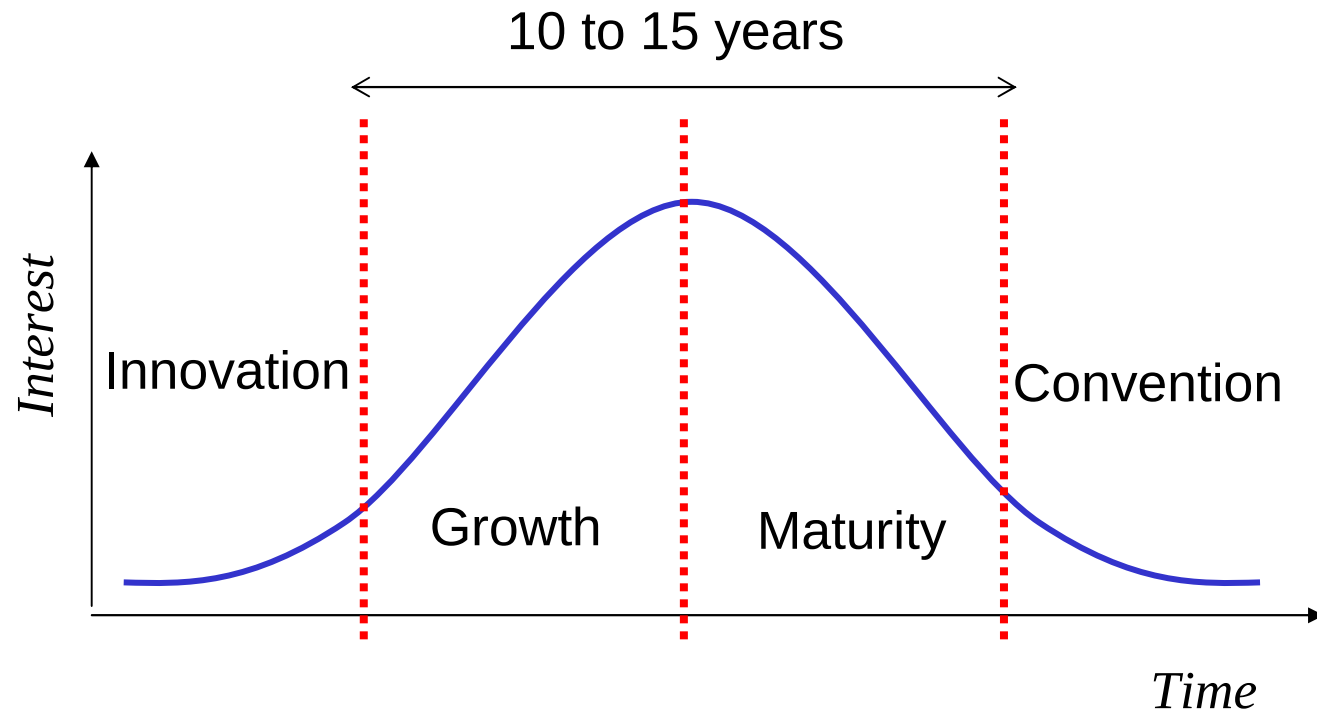
Lionel SEINTURIER (USTL/LIFL/Adam)
Didier DONSEZ (UJF/LIG/Adèle)

En anglais

- CBSE : Component Based Software Engineering

Un petit rappel

- Racoon [1997]



Un petit rappel

- Racoon [1997]

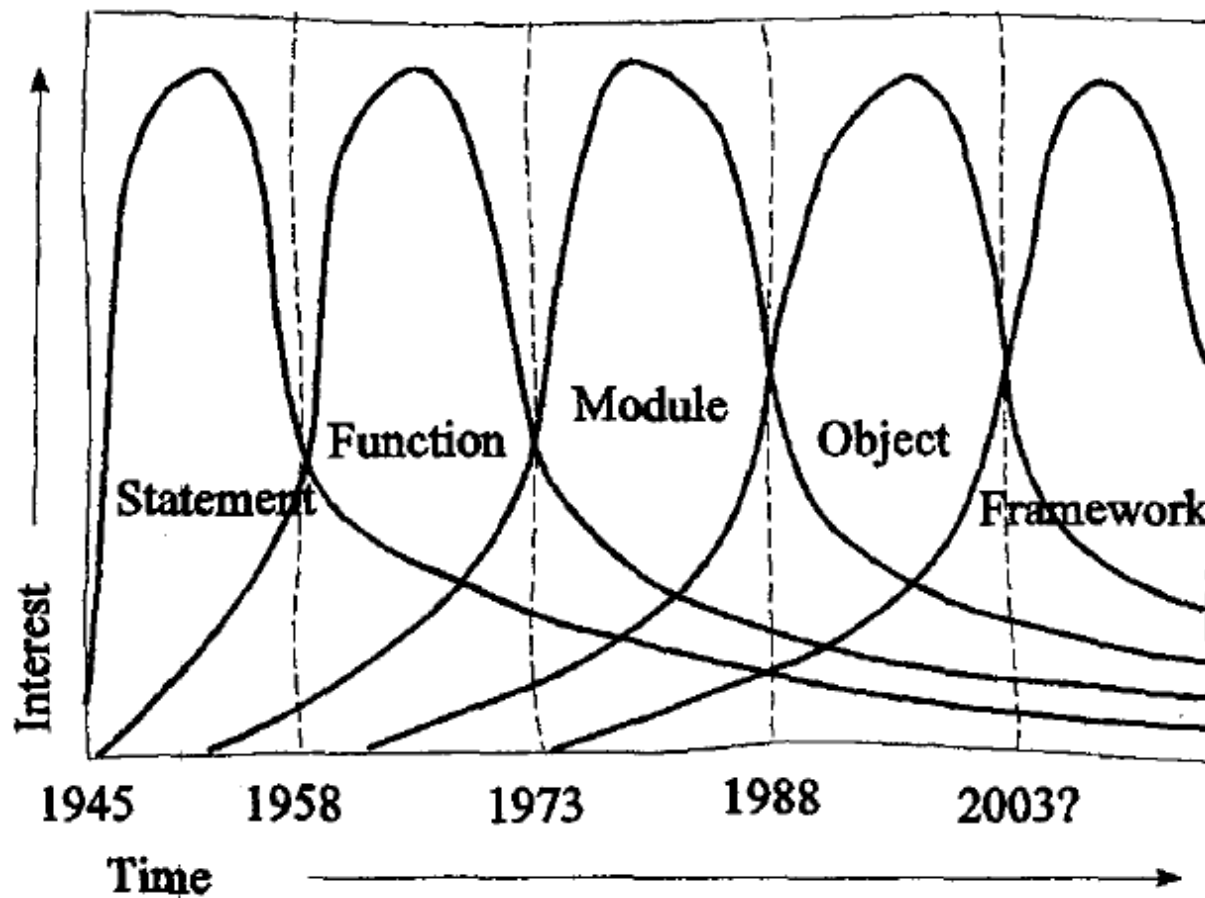
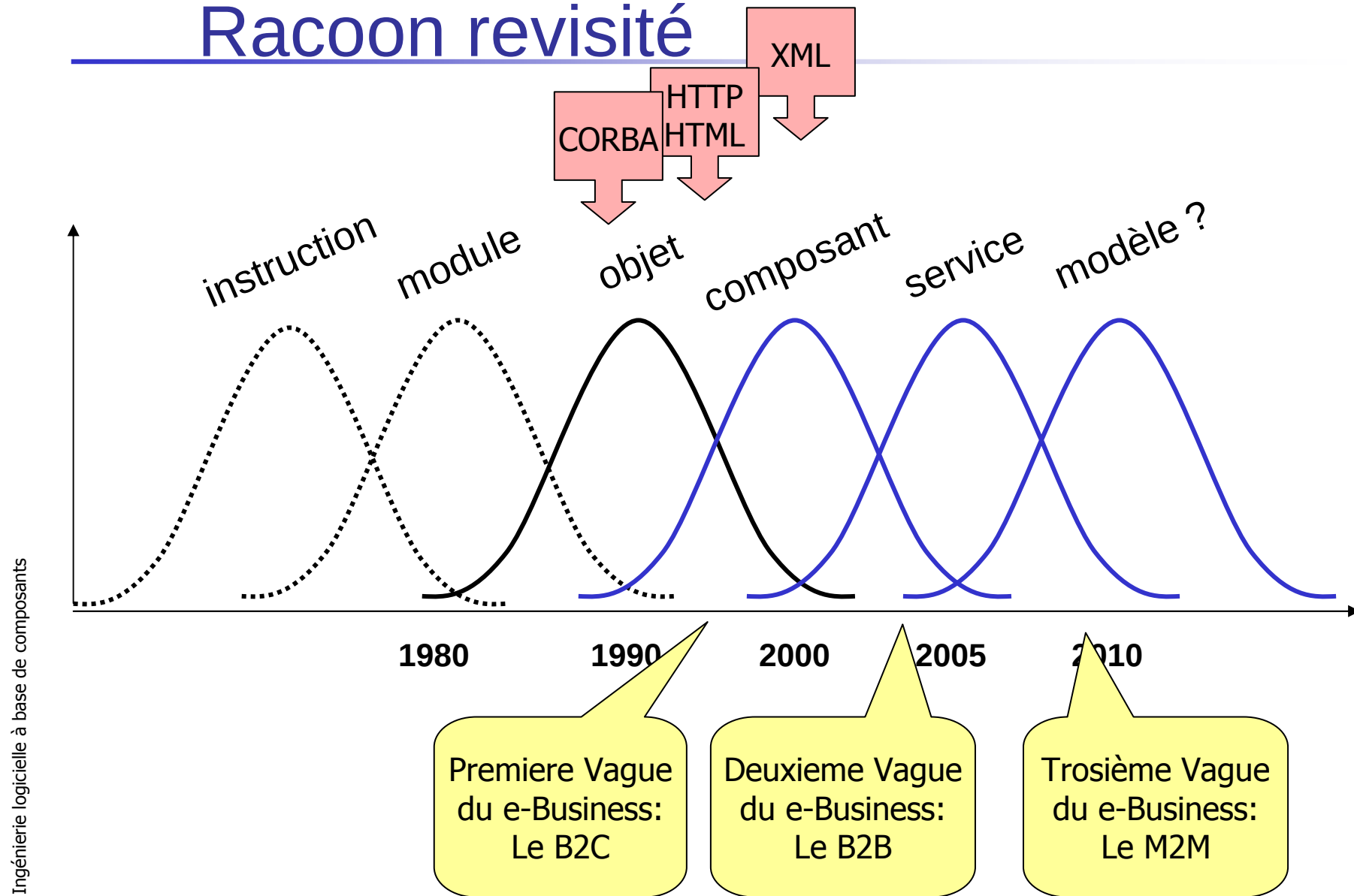


Figure 2: The Organization Stream.

Racoon revisité



Rappel sur la programmation OO

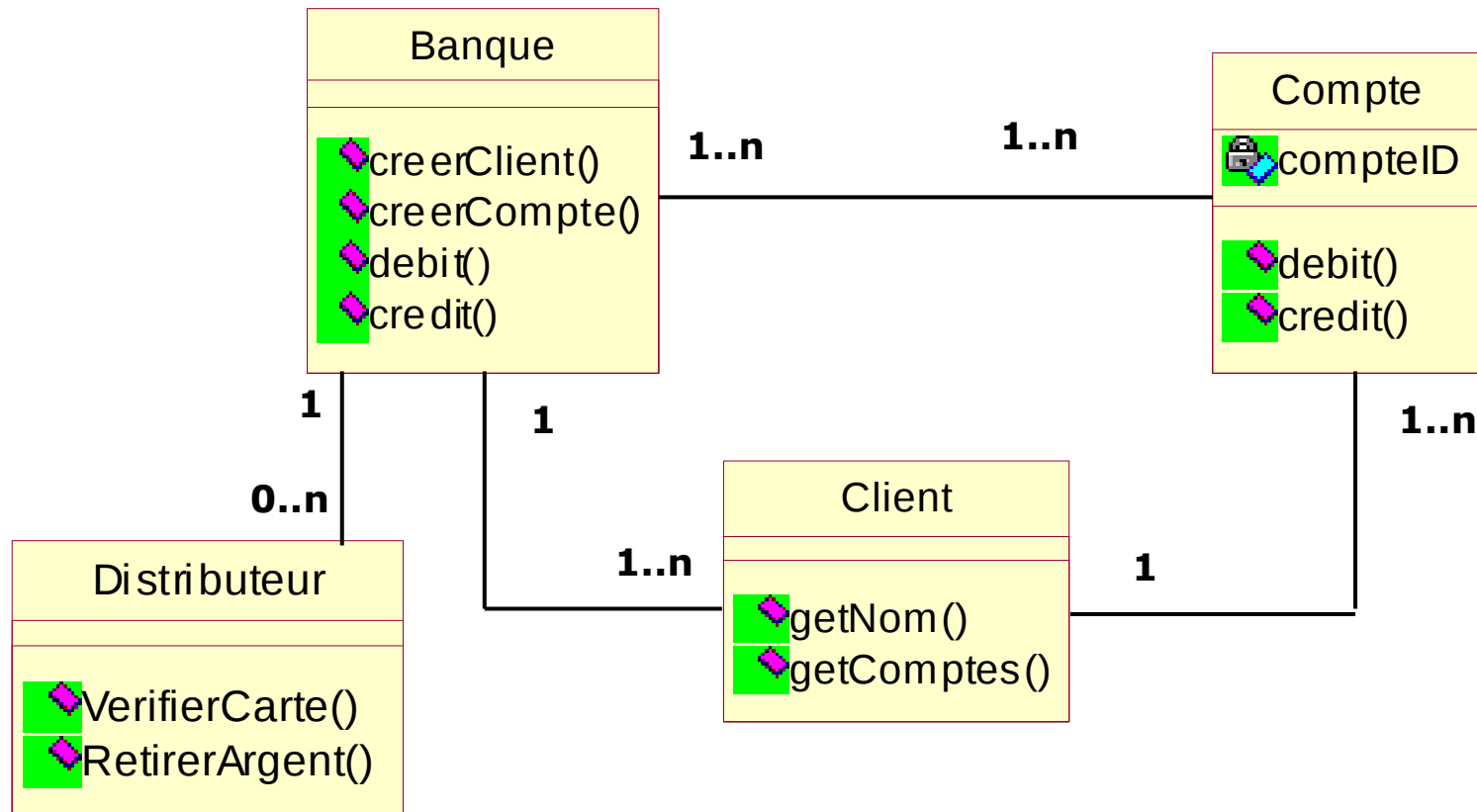
« *programming in the small* »

- Mouvance des années 1990
 - CORBA, DCOM, Java RMI ...
- Exemple de code
 - Un client C invoque N fois la méthode execute() d'un serveur S

```
S s=new SImpl()  
C c1=new CImpl1(s, N);  
C c2=new CImpl2bis(s, N);  
  
new Thread(c1).start();  
new Thread(c2).start();
```

Rappel sur la programmation OO

« *programming in the small* »



- => Composition de fonctionnalités parallèles, bon découpage dans le code des objets

Propriétés non-fonctionnelles

(extra-fonctionnelles, techniques)

- Propriétés requises pour la mise en œuvre d'un logiciel dans un environnement opérationnel
 - Exemple: sécurité, fiabilité, persistance, passage à l'échelle, distribution, facturation, contrôle de l'usage ...
- Mise en œuvre via les API de services non-fonctionnelles (*ou extra-fonctionnelles, ou techniques*)
 - SSO, moniteur transactionnel, bases de données relationnelles, annuaire LDAP ...
- Problème:
 - multiple expertises métier
 - rare et donc cher
 - Non indépendance entre les services
 - Transaction et Persistance, ...

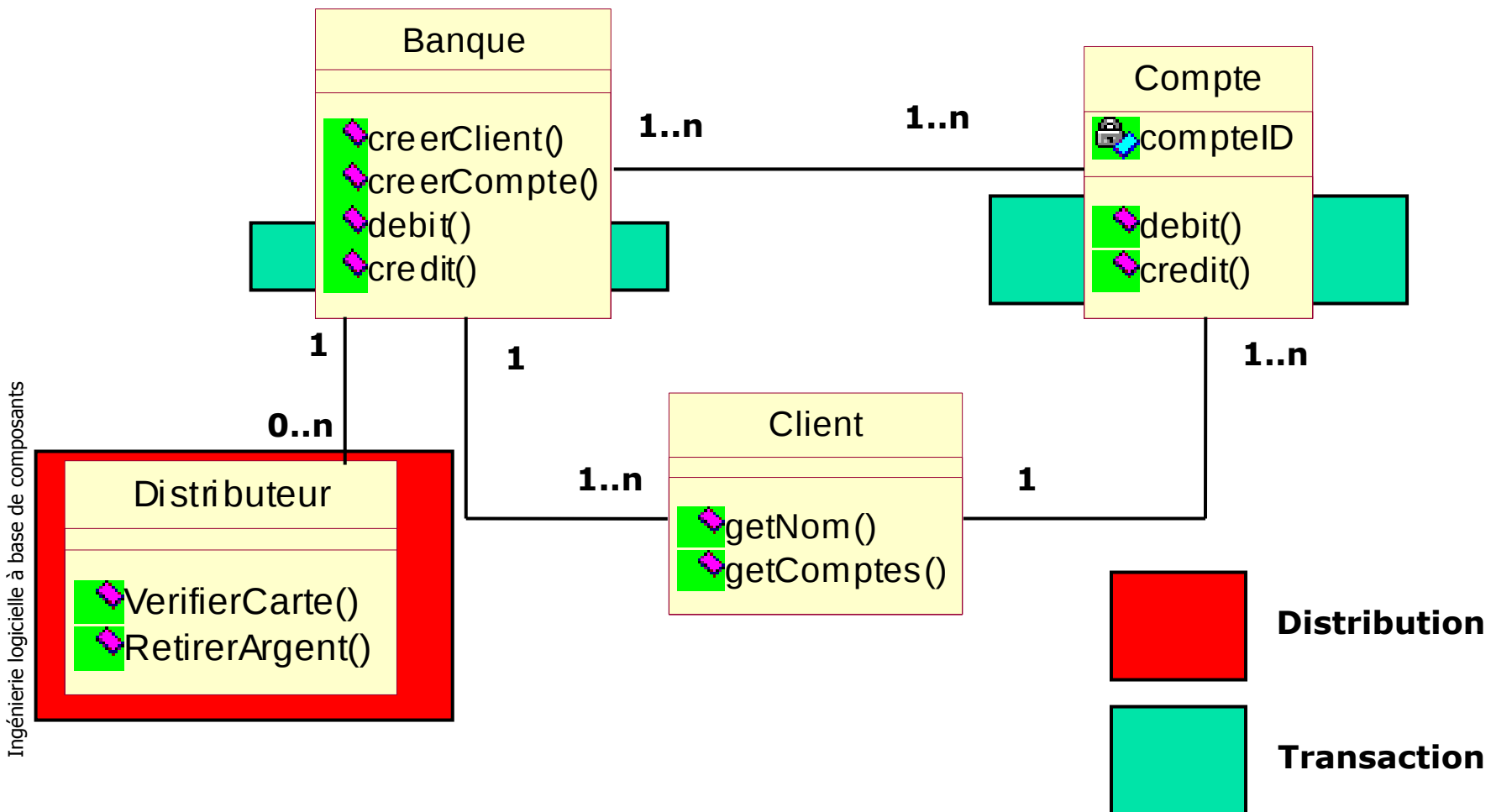
Rappel sur la programmation OO

« *programming in the small* »

- Probleme :
 - Il existe des fonctionnalités orthogonales qui concernent des ensembles d'objets
- Ex: Application Bancaire
 - Gestion de transactions (débit / crédit)
- Ex: Application Monitorée
 - Trace de méthodes, statistiques
- Ex: Applications Distribuée
 - Web services
 - RMI
 - CORBA IIOP
 - ORPC (DCOM)
- Ex: Applications Sécurisées
 - Agent Authentifié
 - Canal confidentiel
- Ex: Application Bancaire Monitorée et Distribuée et Sécurisé

Rappel sur la programmation OO

« *programming in the small* »



Rappel sur la programmation OO

« *programming in the small* »

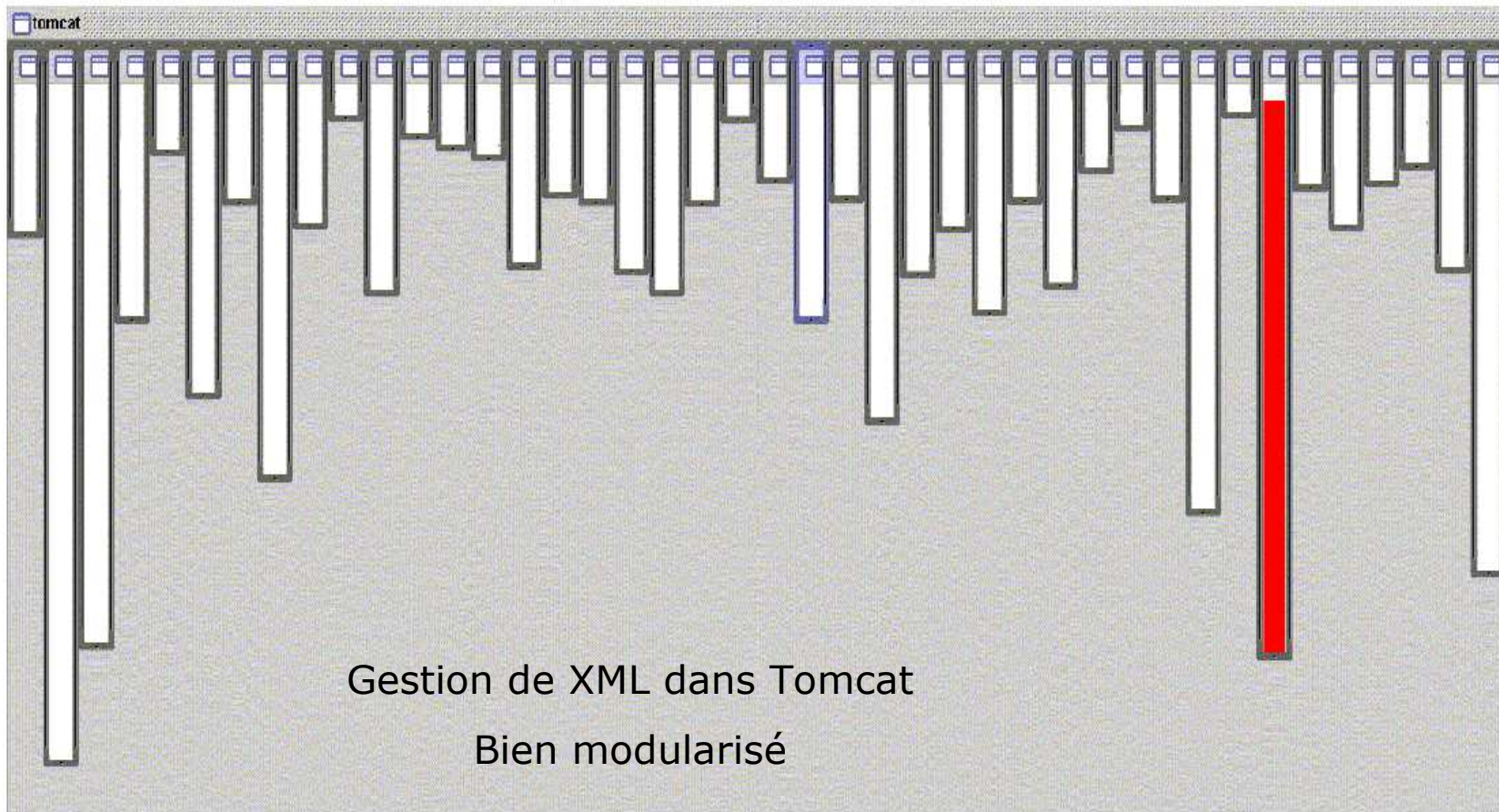
- Problème de cette solution
 - Bonne expertise de tous les programmeurs
 - Transaction, Distribution complexes
 - Banque: Transaction, Compte: Transaction
 - Maintenance et évolution du code
 - Bug copié / collé dans beaucoup d'objets
 - Modifier le même code éparpillé à différents endroits pour la maintenance et l'évolution
- Mauvaise Productivité,
 - Mauvaise évolution,
 - Mauvaise réutilisation

Rappel sur la programmation OO

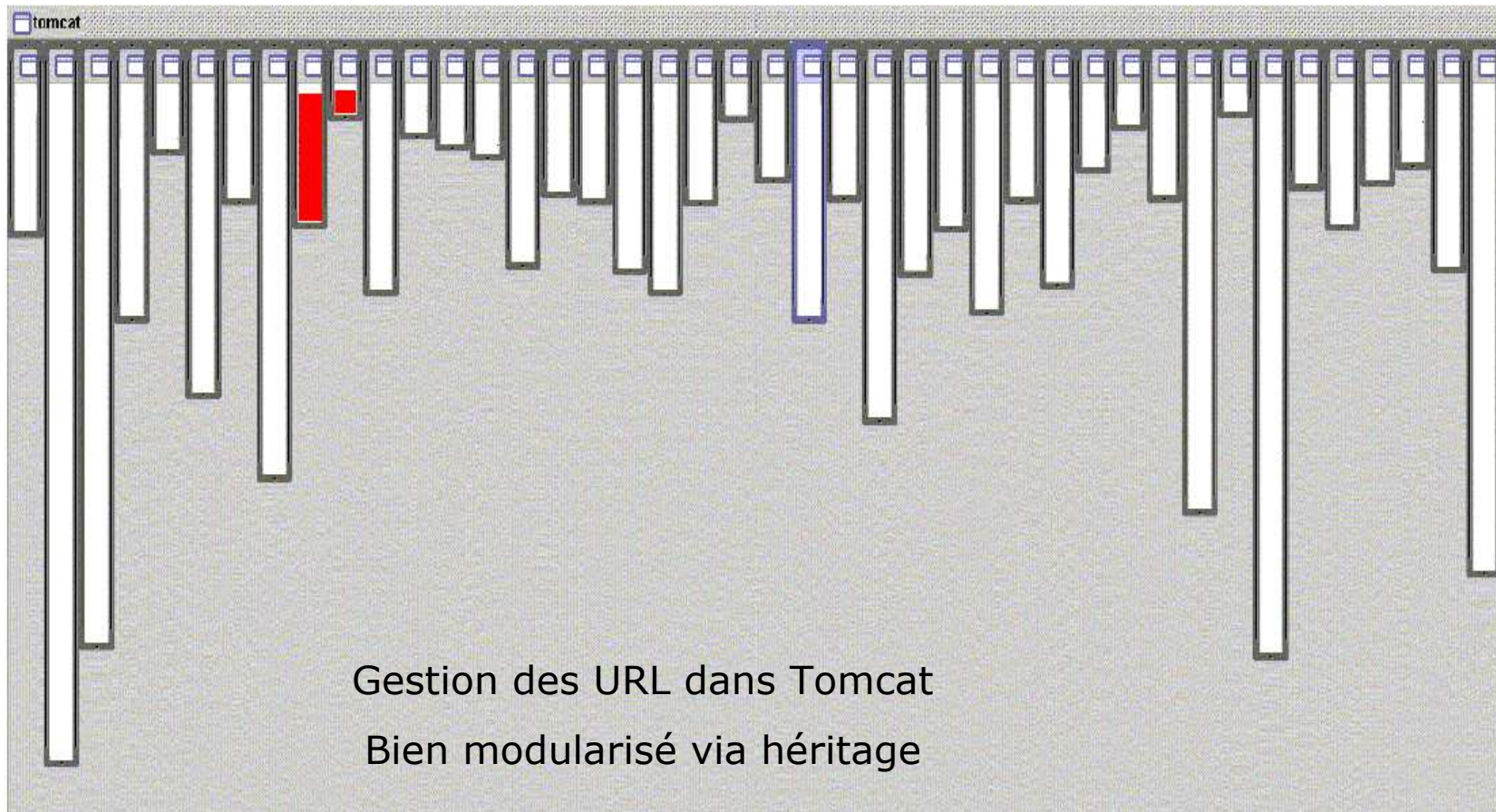
« *programming in the small* »

- Limitations des objets (comme unité de construction)
 - Pas d'expression explicite des ressources requises par un objet (autres objets, services fournis par l'environnement)
 - Pas de vue globale de l'architecture d'une application
 - Pas de possibilité d'expression de besoins "non fonctionnels" (non spécifiques à une application)
 - persistance, fiabilité, sécurité, performance ...
 - Peu d'outils pour le déploiement et l'administration

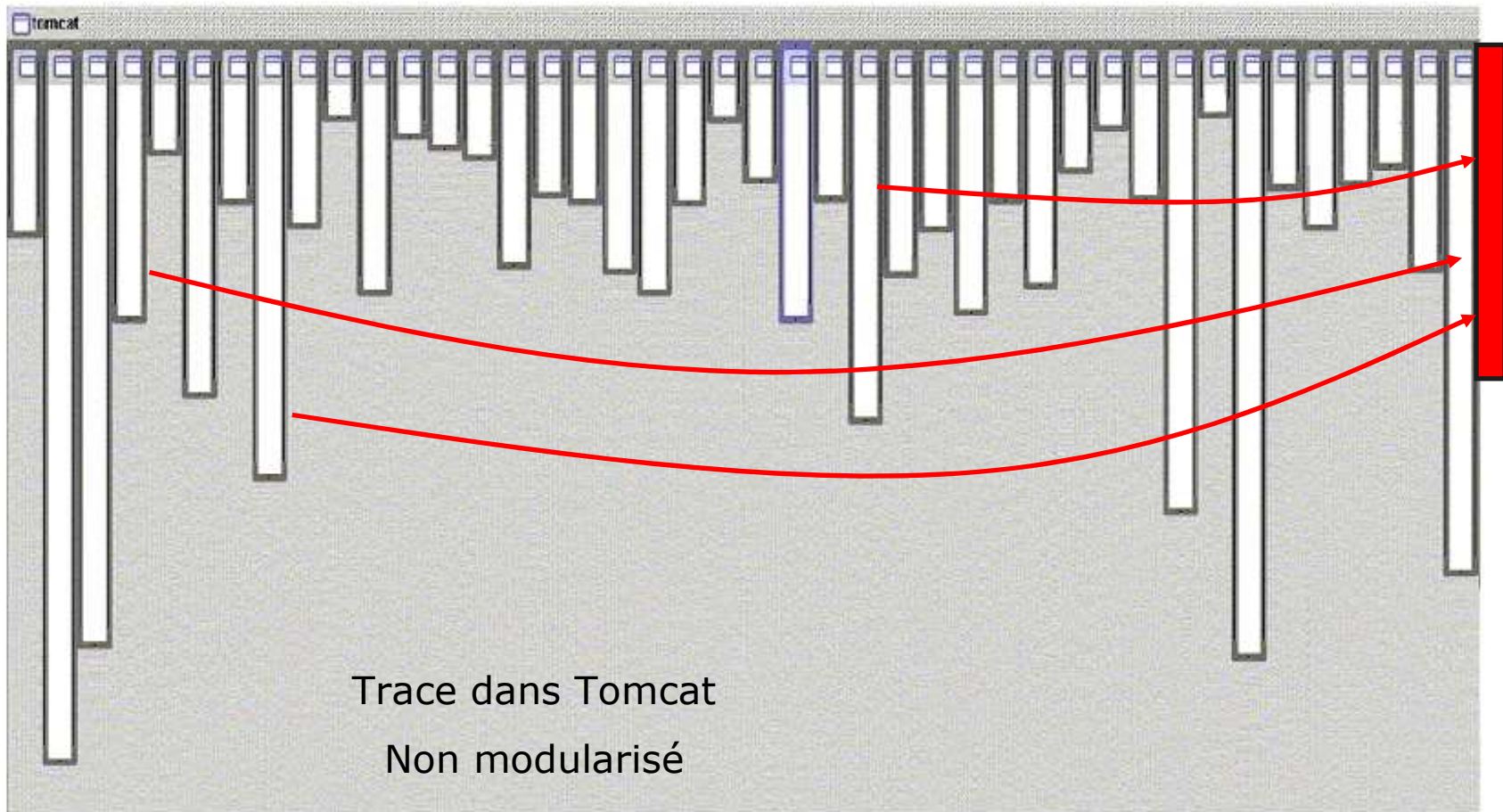
Séparation des Préoccupations



Séparation des Préoccupations



Séparation des Préoccupations



Avantages de la SdP

- Minimise le code
 - Code présent qu'une seule fois
 - => plus facile à maintenir et à faire évoluer
 - A chaque programmeur son expertise
 - Code Fonctionnel (ex application bancaire)
 - Les différents services transversaux
 - => transparences entre le travail des programmeurs
- Service Non Fonctionnel
 - => Frontière Fonctionnel / Non Fonctionnel floue
 - Transaction dans le cahier des charges ??
 - Service Primordial, Services secondaires

Les approches de la SdP

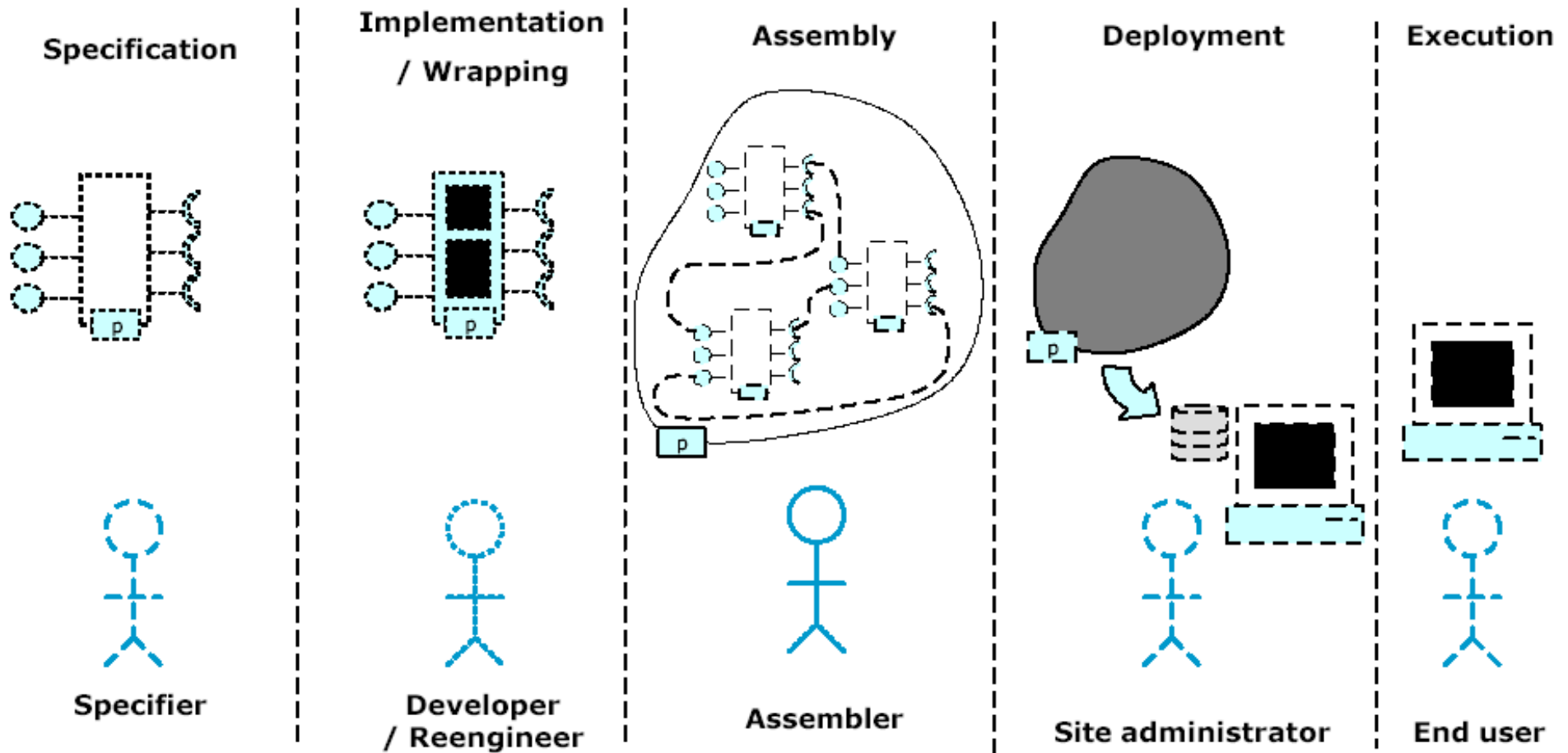
- Programmation Orientée Aspect
 - Voir cours 1 (Jacky Estublier)
- Programmation à base de Composants

Développement orienté composant

Ce que l'on espère avec les composants (vs objets)

- plus haut niveau abstraction
 - Moins d'entités à manipuler (gros grain)
- meilleure encapsulation, protection, autonomie
 - programmation + systématique + vérifiable
- communications plus explicites
 - port, interface, connecteur
- connectables
 - schéma de connexion (ADL) : « plan » applicatif
- séparation « métier » - technique
- meilleure couverture du cycle de vie
 - conception, implémentation, *packaging*, déploiement, exécution

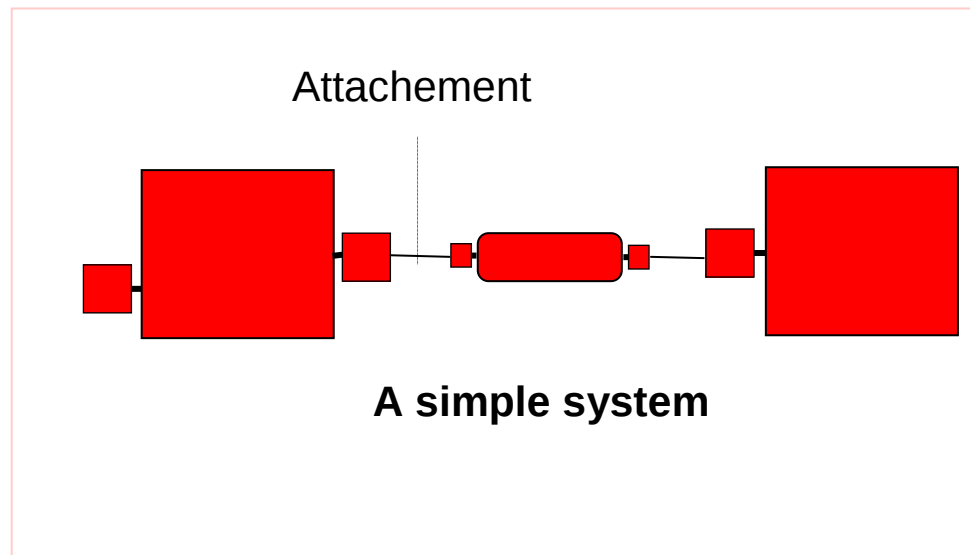
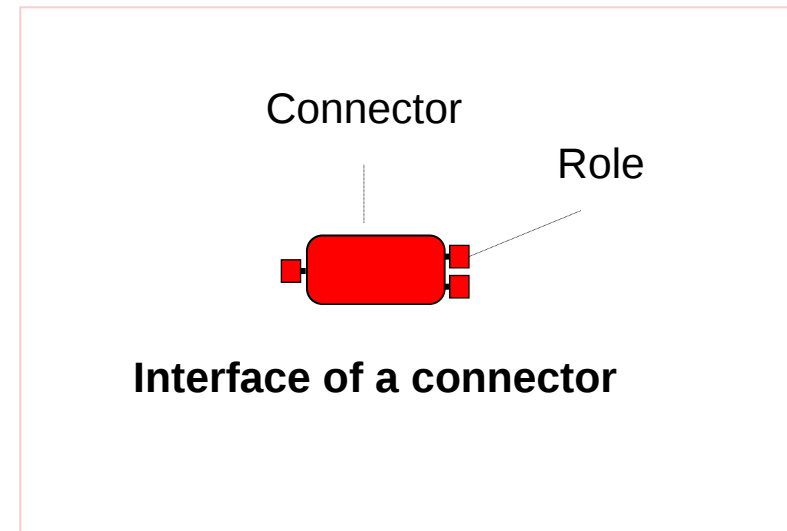
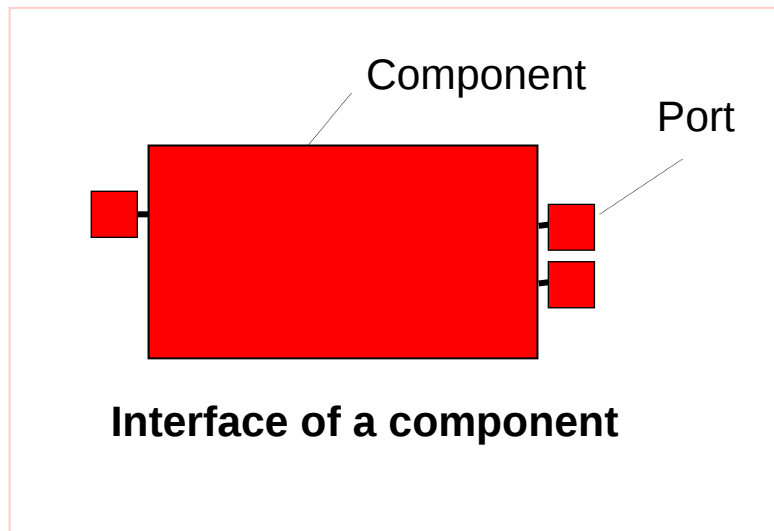
Cycle de développement



Définition composant

- 1ère apparition du terme [McIlroy 68]
 - *Mass Produced **Software Components***
 - <http://www.cs.dartmouth.edu/~doug/components.txt>
- 30 ans + tard : une réalité industrielle avec
Sun JavaBeans & EJB, OMG CCM, MS .NET/COM+, ...
- *A component is a unit of composition with **contractually specified interfaces** and **context dependencies** only. A software component can be **deployed** independently and is subject to **composition** by third parties. [Szyperski 97]*
- Recensement [Szyperski 02] : 11 définitions +/- $\equiv$

Notion de base



Notion de base

Component

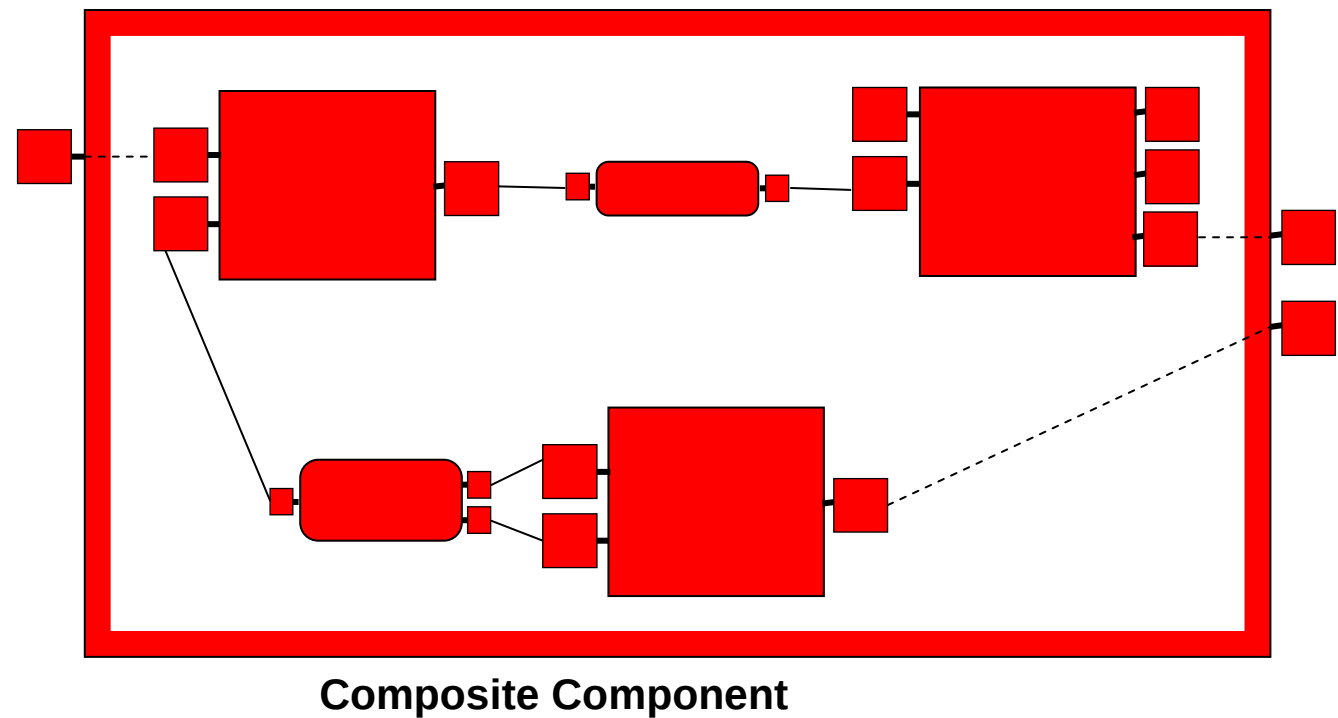
Connector

Port

Role

Attachement

Binding



Connecteurs

- Types
 - Appel de procédure (procedure call)
 - Client – Serveur
 - Requis – Fournis
 - Événement (event)
 - *Patron* Observable – Observateur
 - Source – Puit
 - Flot (data stream)
 - *Patron* Producteur – Consommateur
 - ...

Connecteurs

Exercice 1

- Proposer un connecteur dédié au transport réseau de flots vidéo (MPEG2) sur UDP et UDP/Multicast

Connecteurs

Exercice 2

- Proposer un connecteur dédié au calcul distribué sur une grille de calcul
 - Propriété: équilibrage de charge, fail-over ...

Conteneurs et structures d'accueil

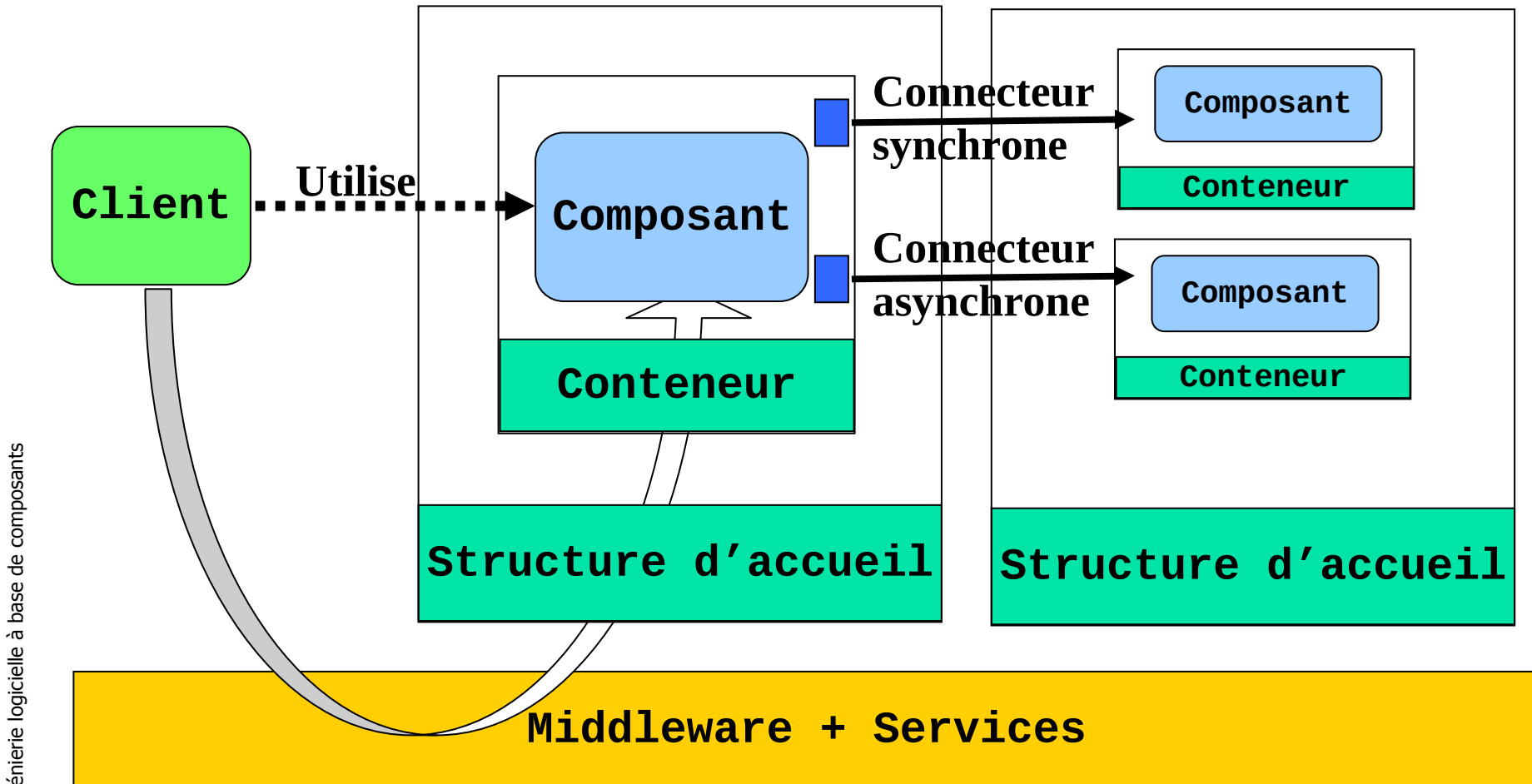
■ Conteneur

- encapsulation d'un composant (et ses composantes)
- prise en charge (masque) les services systèmes
 - nommage, sécurité, transaction, persistance ...
- prise en charge partielle des connecteurs
 - invocations et événements
- techniquement par interposition (ou délégation)

■ Structures d'accueil

- Espace de déploiement du code du composant
- Espace d'exécution des conteneurs et des composants
- Médiateur entre les conteneurs et les services systèmes

Conteneurs et structures d'accueil



Nombreux modèles de composant

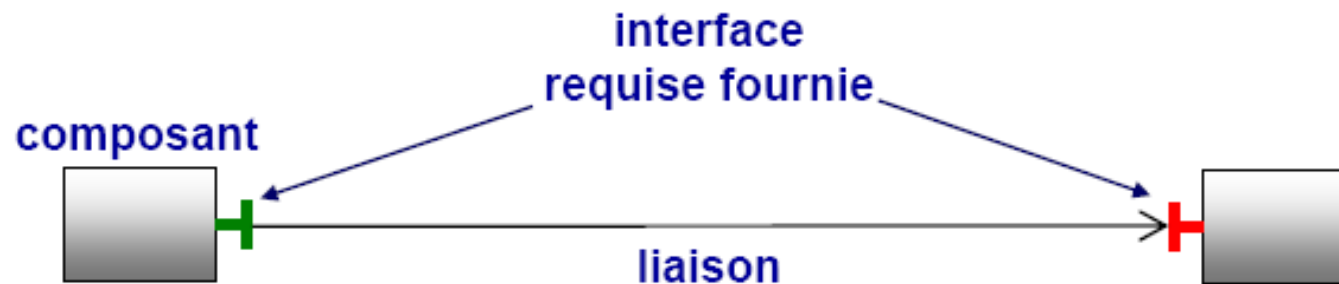
- (20+)
- construits au-dessus Java, C, C++, C#
- Java Beans, EJB, CCM, COM+, JMX, OSGi, SCA, CCA, SF
- Avalon, Fractal, K-Component, Comet, Kilim, OpenCOM, FuseJ, Jiazzi, SOFA, ArticBeans, PECOS, Robocop, Draco, Wcomp, Rubus, Koala, PACC-Pin, SB/DS, iPOJO, OMISID, Osagaia, ...
- Bonobo, Carbon, Plexus, Spring
- au niveau analyse/conception : UML2, eCore, ...

Conséquence de la multiplicité des modèles

- multiplicité du vocabulaire
 - composant, *bean*, *bundle*
 - interface/liaison, port/connecteur, port/conduit, facette, puits/source
 - requis/fourni, client/serveur, export/import, service/référence
 - conteneur, membrane, services techniques, contrôleur
 - *framework*, serveur d'applications
- grande variabilité dans les propriétés attachées aux notions
- exemples
 - Fractal : composant, interface, liaison, client/serveur
 - CCM : composant, facette, port, puits, source
 - UML 2 : composant, fragment, port, interface
 - OSGi : *bundle*, *package* importé/exporté, service/référence
- un même terme peut avoir des acceptations $\neq$ selon les modèles
- qualifier les notions (« connecteur au sens ... »)
- pas toujours facile de définir les équivalences

2 grande catégorie de modèle de composants

- 1^{er} triptyque : composant, interface, liaison
 - un composant fourni et/ou requiert une ou plusieurs interfaces
 - une liaison est un chemin de communication entre une interface requise et une interface fournie



2 grande catégorie de modèle de composants

- 2^{ème} triptyque : composant, port, connecteur
 - un composant fourni et/ou requiert une ou plusieurs ports
 - un connecteur implémente un schéma de communication entre des composants (client/serveur, diffusion, etc.)
 - un composant est relié à un connecteur via un ou plusieurs ports



- connecteur $\approx$ liaison avec comportement
 - on peut considérer connecteur = composant (de communication)
 - composant, interface, liaison

Quelques « poncifs » à propos des composants

- COTS Commercial Off The Shelf
 - vieux discours (voir procédures, fonctions, objet, ...)
 - taille applis ↗ donc besoin : toujours plus de réutilisation
 - mais *quid* de la contractualisation ?
- « *Programming in the large* »
 - vs « *programming in the small* » (objet)
 - vrai d'un certain point de vue mais nouveaux points à traiter (liés au non fonctionnel par ex.)
- Composants patrimoniaux (*legacy*)

Architecture logicielle

■ Definition (une des nombreuses)

“Une architecture logicielle est une **spécification abstraite** du système qui contient des **composants fonctionnels** décrits par l'intermédiaire de leur **comportement**, de leurs **interfaces** et de leur **assemblage**.” (Hayes-Roth, 1994)

“L'architecture logicielle d'un programme ou d'un système d'information est la structure ou les structures d'un système; elle comprend les **composants logiciels**, leur **propriétés externes visibles** et leurs **relations**.” (Clements et al., 1997).

■ Langage de Description d'Architecture (ADL)

- Spécialisé, souvent en XML

■ *Survey* : [Medvidovic 00]

- Wright, Darwin, ...

Architecture logicielle : bénéfices

- Compréhension
 - abstraction de haut niveau de la structure du système
- Cadre pour la construction de l'application
 - identification des opportunités de réutilisation de modules logiciels
 - base pour la génération de code
- Vérification
- Support de l'évolution du logiciel
 - préservation des invariants
 - impact d'une modification

Complémentarité

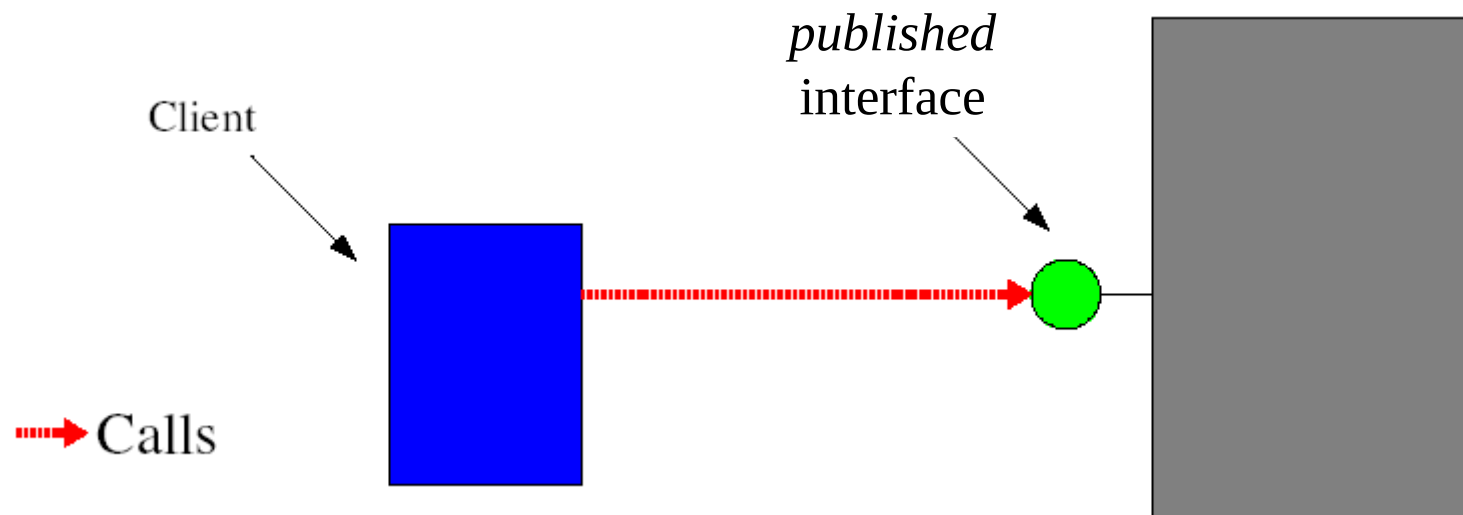
- Architecture
 - construite à partir de composants
- Composants
 - assemblés pour construire une architecture
- 2 visions complémentaires
 - architecture : *top-down*
 - composants : *bottom-up*

Design patterns pour les composants

- Separation of interface and implementation
- Factory
- Inversion of control
- Interceptor
- Service registry

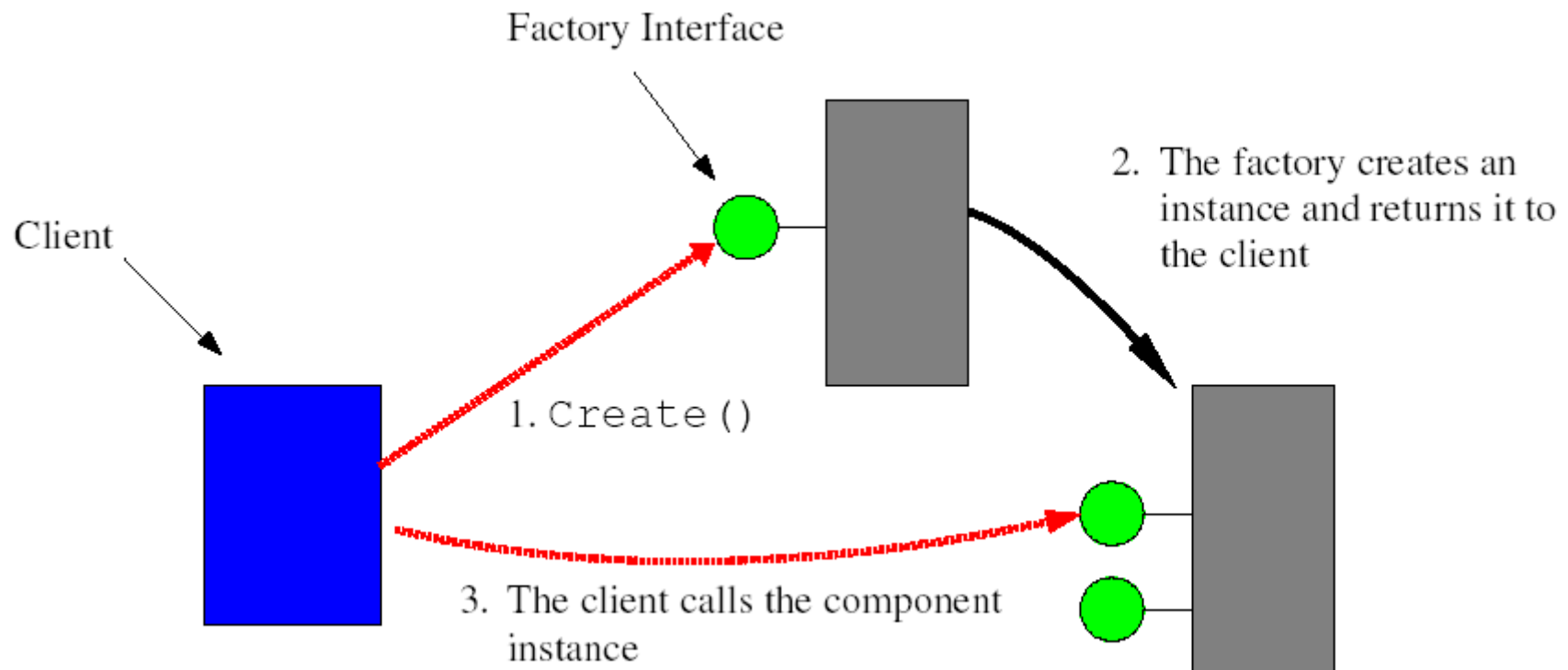
Separation of interface and implementation

- Clients communicate with component instances only through well defined interfaces (contract)
- Different implementations can implement the same interface



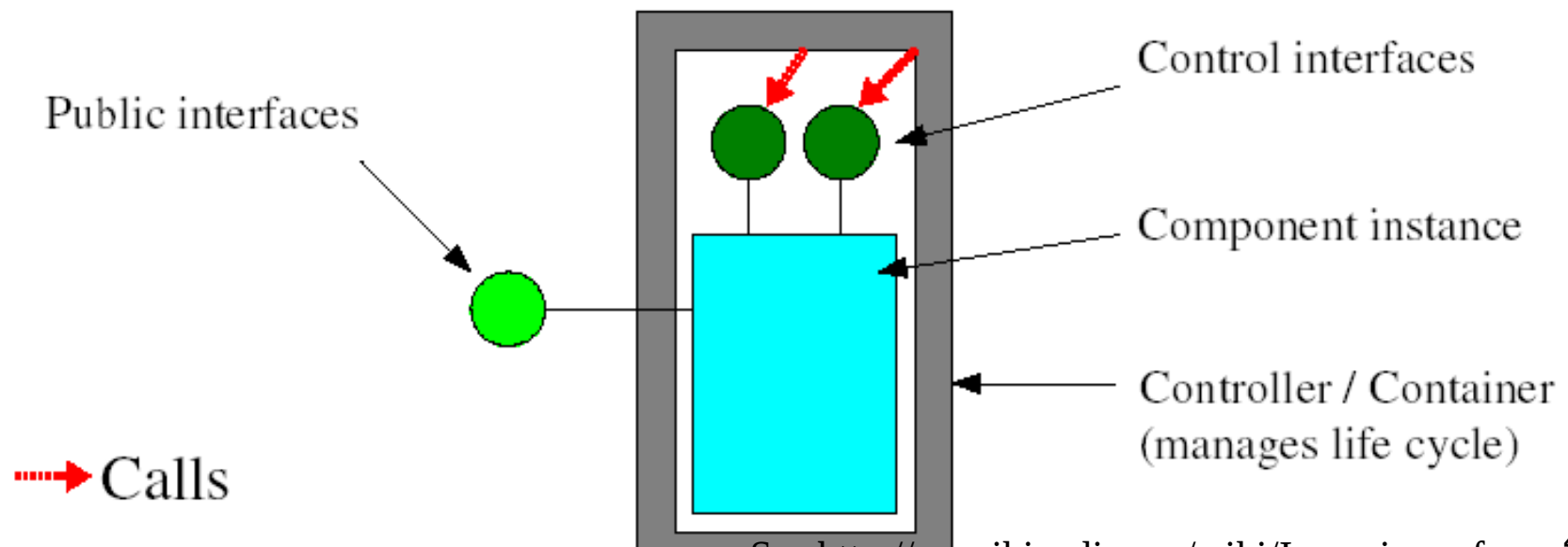
Factory

- Provide and indirect instantiation mechanism



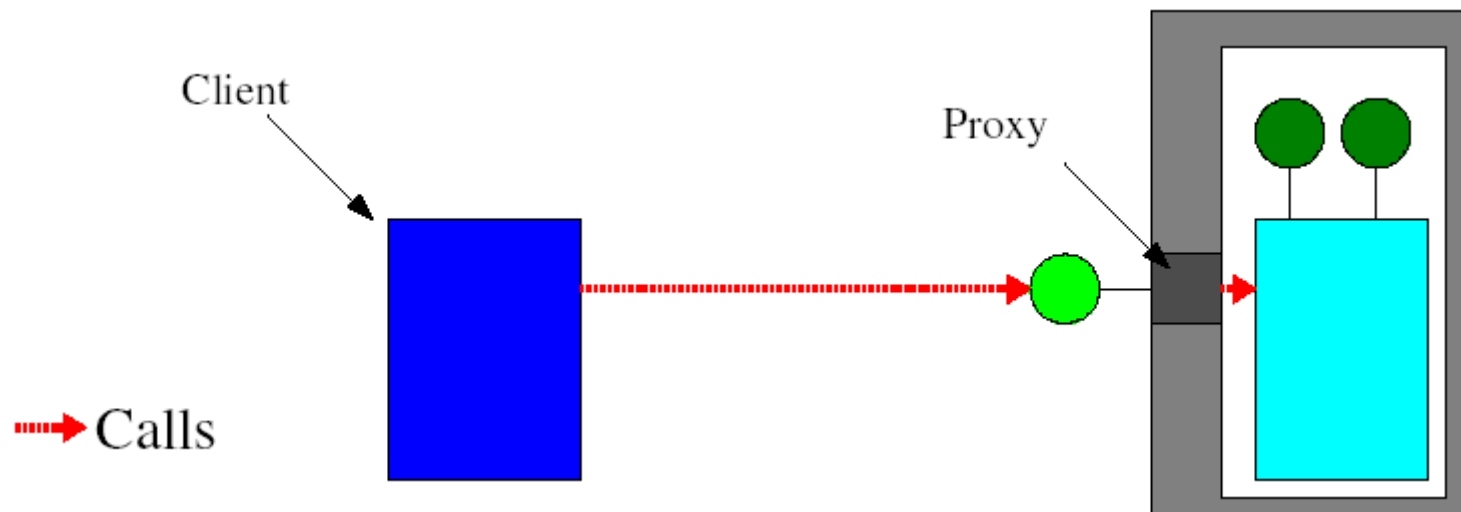
Inversion of control

- A component instance is always externally managed through a set of control interfaces that provide life cycle methods
- Control interfaces (Controller) can be discovered at run time



Interceptor

- Clients do not communicate directly with the component instance, calls are received by a proxy
- The interceptor may insert, block or replace the component's functionalities



Classification des modèles de composants

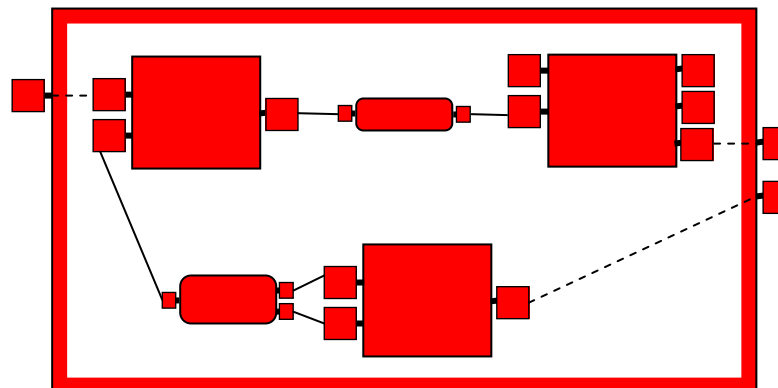
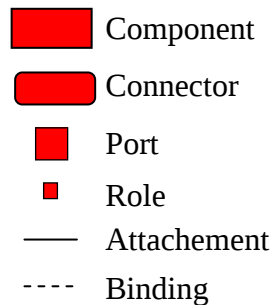
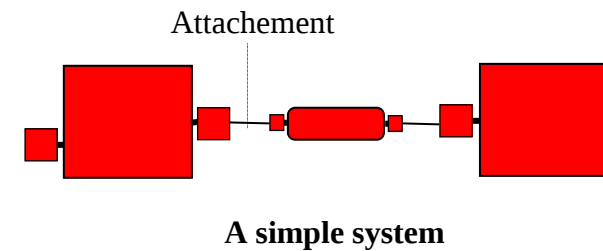
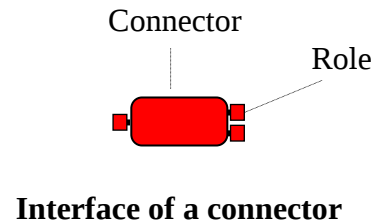
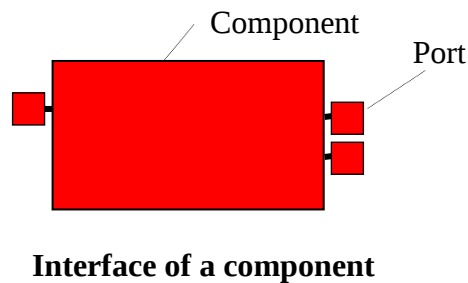
- Application
 - Généralement : Spécialisation par domaine
 - Entreprise: EJB, CCM, .NET/COM+, SCA, Spring
 - Supercomputing: CCA
 - Embarqué (contraint, temps réel): KOALA, PECOS, BIP, Fractal /Cecilia, CCM-Light
 - Multimédia : SOFA, OMISID, Osagaia
- *Middleware*
 - Fractal, Avalon, Plexus, JMX, OpenCOM, OSGi, ...
- Cependant la frontière n'est pas nette
 - JMX, Spring/OSGi, SCA/OSGi, Fractal/Cecilia...
 - Souvent lié aux services techniques utilisés

Modèles de composant

- ACME
- JavaBeans
- COM
- Fractal
- ArchJava
- KOALA
- PECOS
- SOFA
- EJB
- CCM
- CCA
- Spring

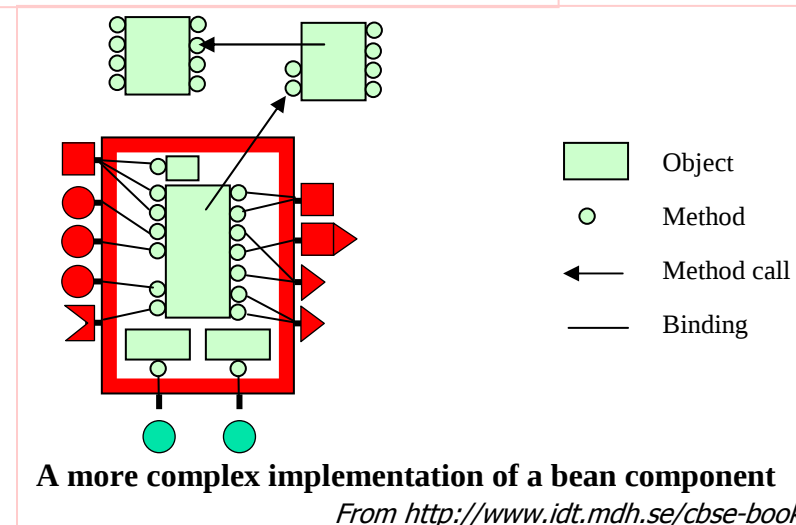
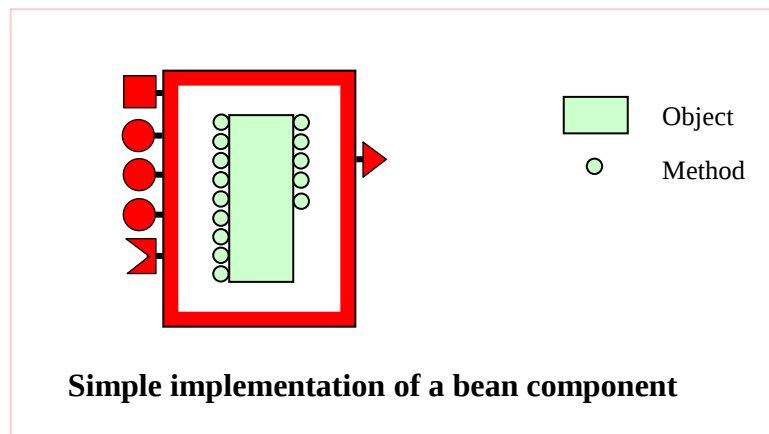
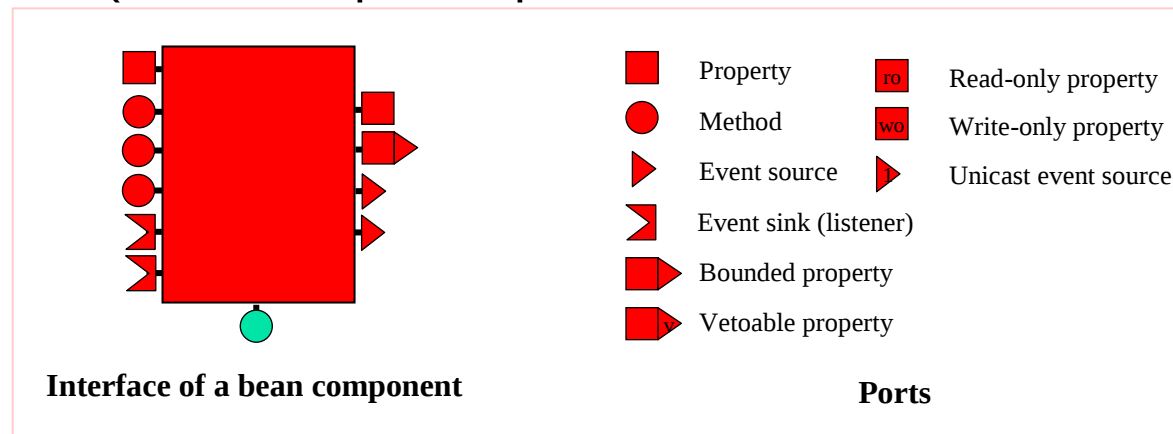
ACME

- Modèle académique
 - Pas de réalité industrielle
- Concepts de base des modèles de composants

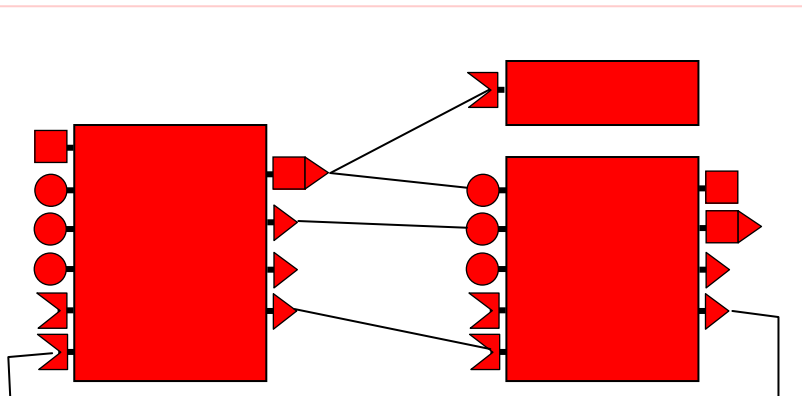


JavaBeans (Sun)

- **Domaine**
 - Application Java standalone (IHM AWT et Swing)
- Pas de SNF (forme simple de persistance via la s rialisation Java)



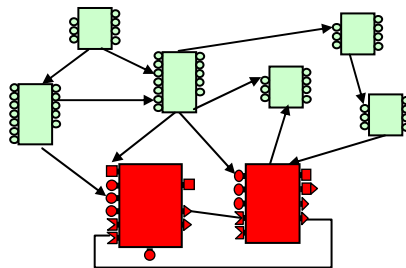
JavaBean (Sun)



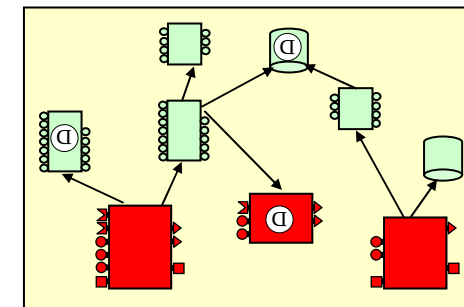
Component-based assembly



Example of connectors (adaptors)

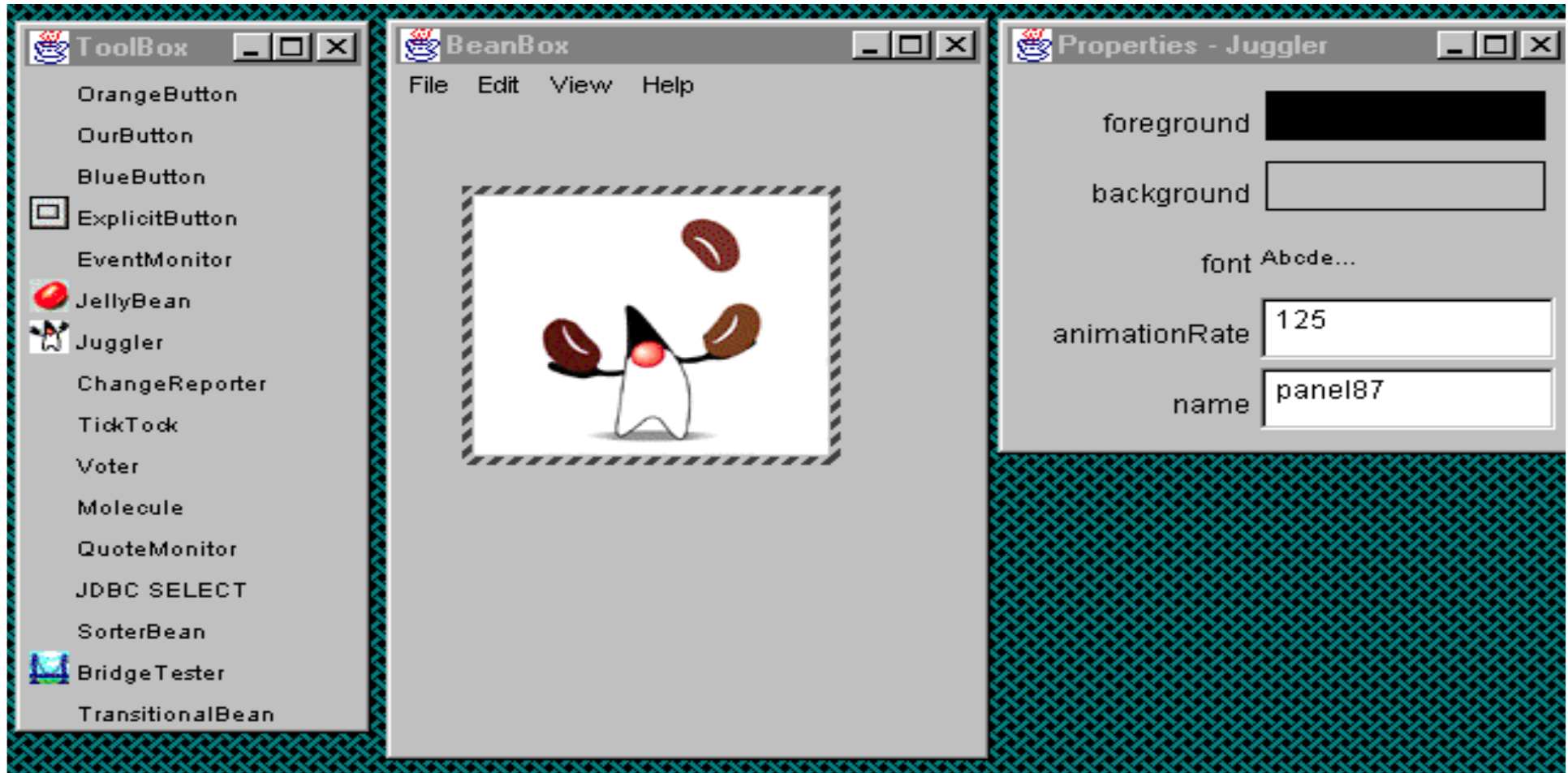


Heterogeneous programming/assembly



Packaging bean components

JavaBeans (demo BeanBox)



Ingénierie

Réserve de composants

Application en construction

Propriété du composant sélectionné

Fractal

- Modèle abstrait de composants
 - Hiérarchique
 - une composition est un composant
 - Extensible
 - nouvelles membranes, nouveau connecteurs, nouveaux contrôleurs ...
- Implémentations
 - Julia : implémentation de référence en Java
 - Think, Cecilia : C pour des ordinateurs nu (sans OS)
 - AOKell, Julius, Julias, Proactive/Fractal, Fractalk, FractNet, ...
- FractalADL
 - ADL statique syntaxe XML
- Nombreux travaux et nombreux canevas « fractalisés »
 - SAFRAN, Oz/K, ...
 - Perseus, JOTM, Jade, Dream, JOnAS à la carte ...
 - Modélisation en Alloy <http://hal.inria.fr/inria-00338987/>

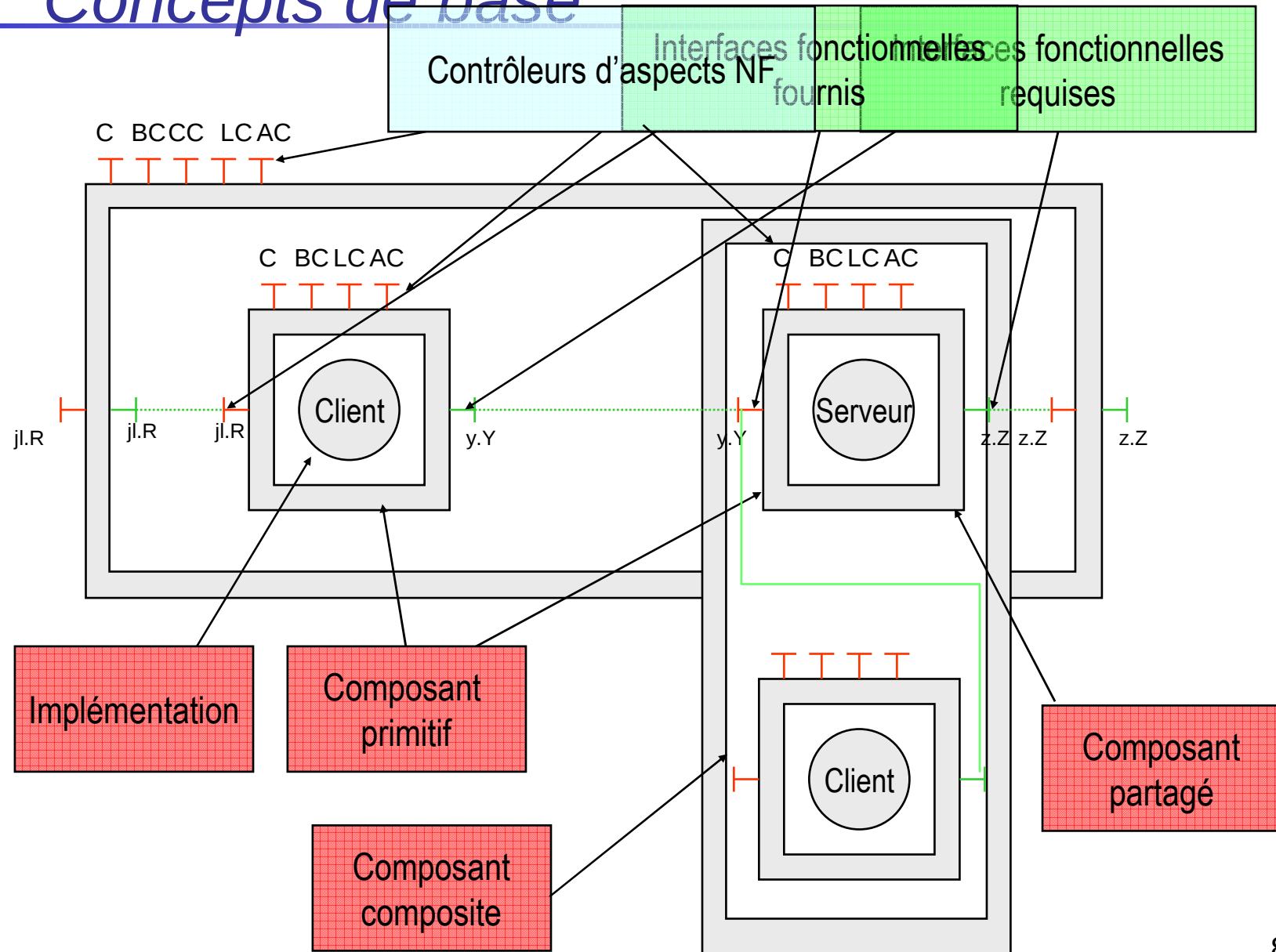
Fractal

Concepts de base

- Typage
- Controleur
- Membrane
- Factory
- Template
- Bootstrap
- Connecteurs

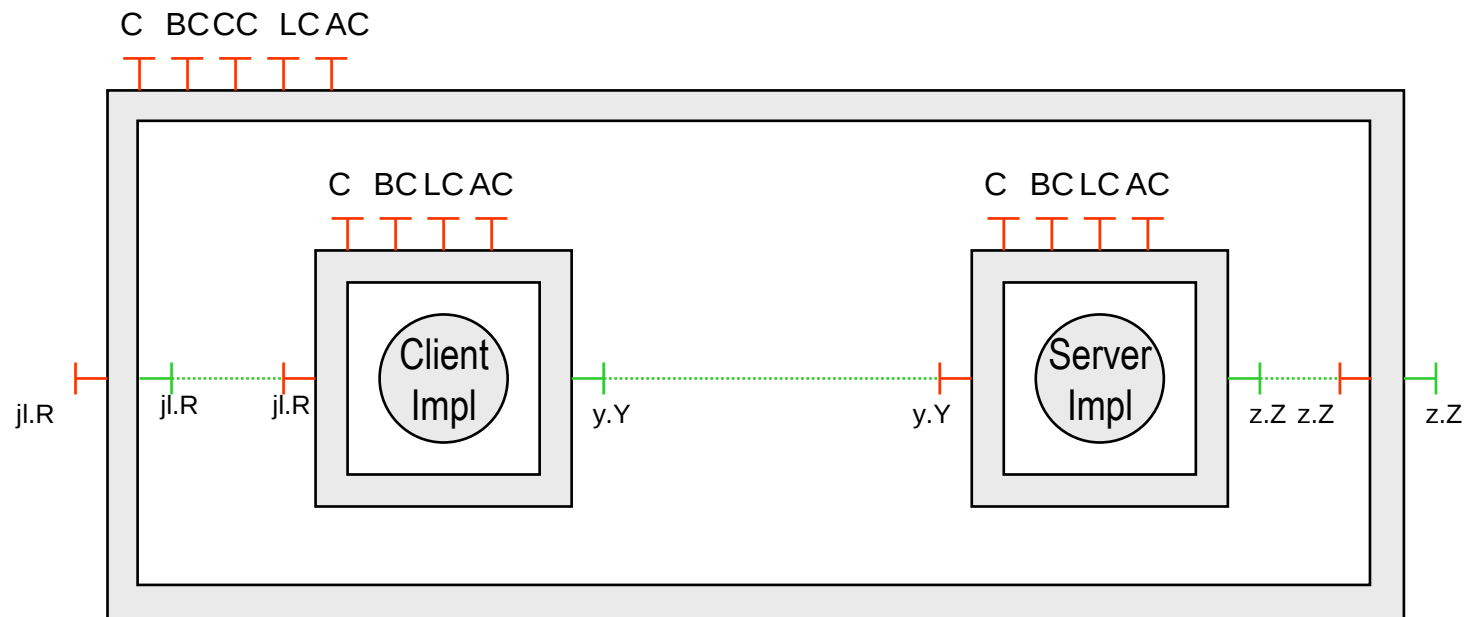
Fractal

Concepts de base



Exemple d'application Fractal

Notation graphique



Fractal ADL

```
<definition name="HelloWorld">
```

```
  <interface name="main" role="server"
    signature="java.lang Runnable"/>
```

```
  <component name="client">
```

```
    <interface name="x1" role="server" signature="java.lang Runnable"/>
```

```
    <interface name="cy2" role="client" signature="y.Y"/>
```

```
    <content class="ClientImpl"/>
```

```
  </component>
```

```
  <component name="server">
```

```
    <interface name="y2" role="server" signature="y.Y"/>
```

```
    <interface name="cz3" role="client" signature="z.Z"
      cardinality="collection" contingency="optional"/>
```

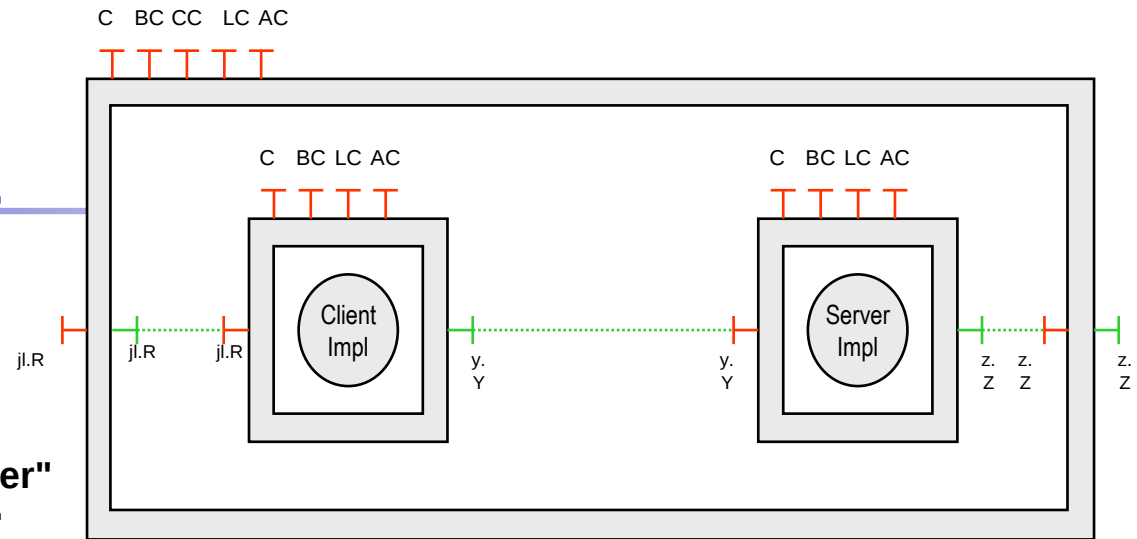
```
    <content class="ServerImpl"/>
```

```
  </component>
```

```
  <binding client="this.x1" server="client.x1"/>
```

```
  <binding client="client.cy2" server="server.y2"/>
```

```
  <binding client="server.z3" server="this.cz3"/>
```



Fractal

Niveaux de conformance

	Introspection		(Re)Configuration	Instantiation		Dynamic (Basic) Typing
	C	I	BC, CC, SC, LC, AC	F	T	
0						
0.1			X			
1	X		X			
1.1	X					
2	X	X				
2.1	X	X	X			
3	X	X				X
3.1	X	X	X			X
3.2	X	X	X	X		X
3.3	X	X	X	X	X	X

Legend :

C : Component
 I : Interface
 BC : BindingController
 CC : ContentController
 SC : SuperController
 LC : LifeCycleController
 AC : AttributeController
 F : Factory
 T : Template

Think

Julia, AOKell

Fractal

■ A lire

- Eric Bruneton, Thierry Coupaye, Matthieu Leclercq, Vivien Quéma, Jean-Bernard Stefani: The FRACTAL component model and its support in Java. Softw., Pract. Exper. 36(11-12): 1257-1284 (2006)
 - <http://dx.doi.org/10.1002/spe.767>
- Philippe Merle, Jean-Bernard Stefani, A formal specification of the Fractal component model in Alloy
 - <http://hal.inria.fr/inria-00338987/fr/>

■ A tester

- Le tutoriel de Julia
 - <http://fractal.ow2.org/java.html>

ArchJava

■ *Principles*

- *Specifies* architecture within Java code
- *Verifies* control flow architecture
 - Statically checked (except for casts, as in Java)
 - Code and architecture evolve together
- Supports dynamically changing architectures

■ Concepts

- Component class vs Ordinary class (ie standard Java)
 - Hierarchical component
- Named Ports
 - Required and provided methods
- Connections between ports
- Ordinary objects
 - Are passed thru ports
 - Can be shared between components

■ No industrial reality

ArchJava

Component

- Component

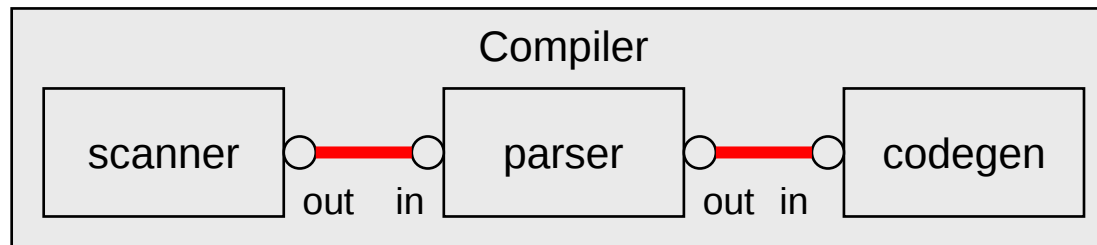


```
public component class Parser {  
    public port in {  
        requires Token nextToken();  
    }  
    public port out {  
        provides AST parse();  
    }  
    AST parse() {  
        Token tok=in.nextToken();  
        return parseExpr(tok);  
    }  
    AST parseExpr(Token tok) { ... }  
    ...  
}
```

ArchJava

Hierarchical component

- Hierarchical component
 - Components instances (new) connected (connect) thru ports



```
public component class Compiler {  
    private final Scanner scanner = new Scanner();  
    private final Parser parser = new Parser();  
    private final CodeGen codegen = new CodeGen();  
    connect scanner.out, parser.in;  
    connect parser.out, codegen.in;  
}
```

Koala Component Model

- Motivation
 - build (software) product families of consumer electronics (TV set, DVD player, VCR).
- Key features
 - ADL derived from Darwin
 - Notion of *interface* as inspired by MS COM.
 - Explicit *provides* and *requires* interfaces and *third party binding*.
 - *Modules* for adding composition specific glue code (C, Koala).
 - *Diversity interfaces* and *switches* for handling product variation.
 - Various *reflection mechanisms* for self adapting components.
 - *Partial evaluation* for handling resource limitations.
 - Multi-threading, Event-passing, ...
 - Components repositories (for configuration management)
- Site : <http://www.extra.research.philips.com/SAE/koala/>

Koala

Interface

```
interface ITuner
{
    void SetFrequency(int f);
    int  GetFrequency(void);
}
```

Composant

```
component CTunerDriver
{
    provides ITuner ptun;
    IInit pini;
    requires II2c ri2c;
}
```

Interface can be
mandatory or optional

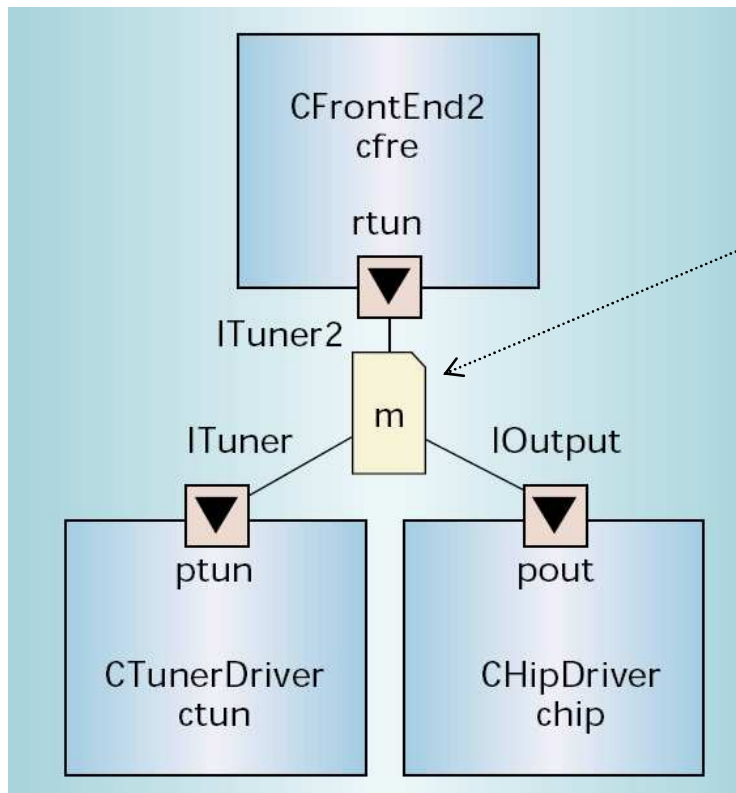
Composant composite

```
component CTvPlatform
{
    provides IProgram pprg;
    requires II2c slow, fast;
    contains
        component CFrontEnd cfre;
        component CTunerDriver ctun;
    connects
        pprg          = cfre.pprg;
        cfre.rtun      = ctun.ptun;
        ctun.ri2c      = fast;
}
```

Connections

Koala Module

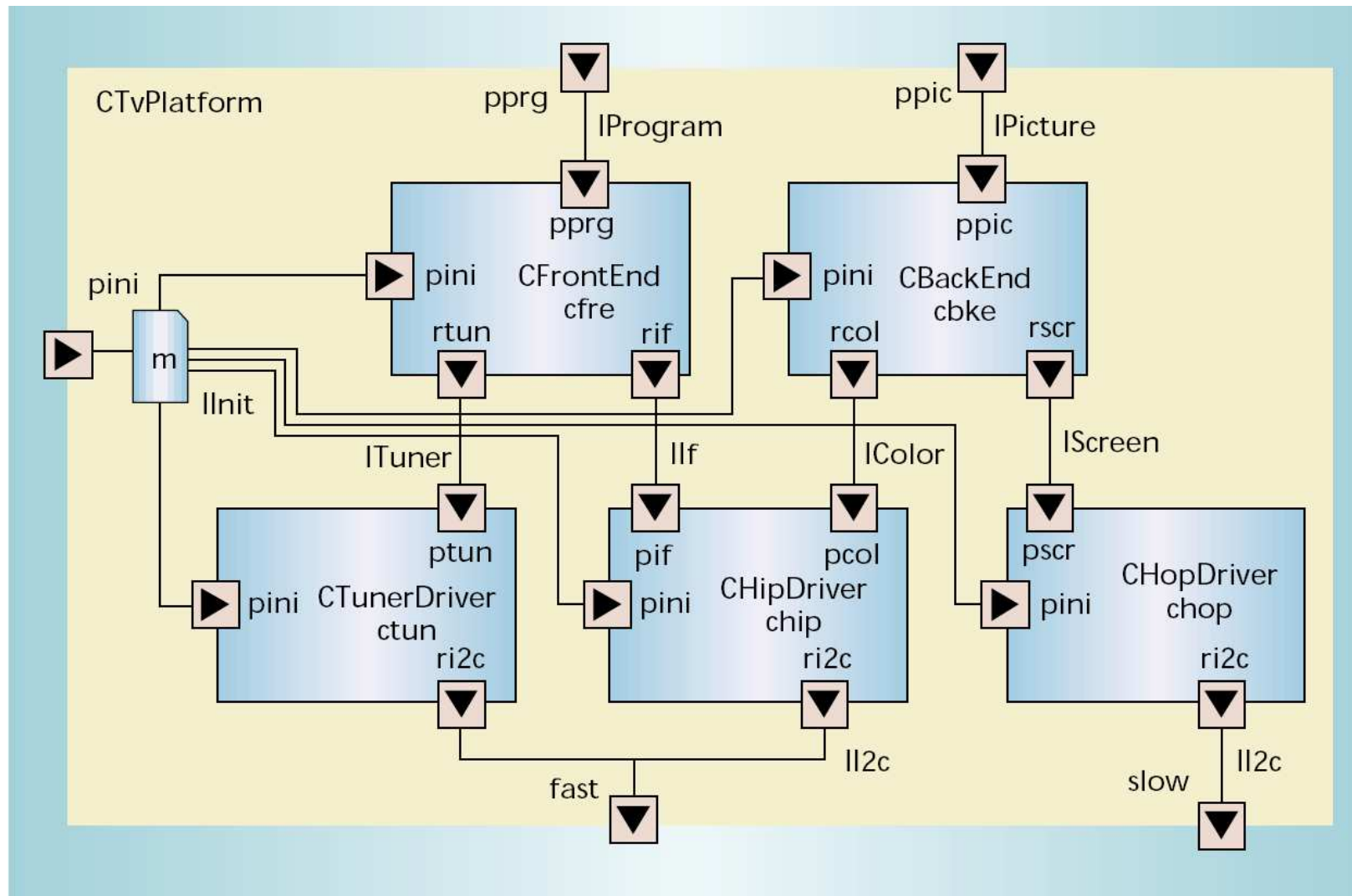
- for adding composition specific glue code
 - C or Koala language



```
within m {  
    cfre.rtun.SetFrequency(x) =  
        ctun.ptun.SetFrequency(x);  
    cfre.rtun.GetFrequency() =  
        ctun.ptun.GetFrequency();  
    cfre.rtun.EnableOutput(x) =  
        chip.pout.EnableOutput(x);  
}
```

Koala

Exemple d'application



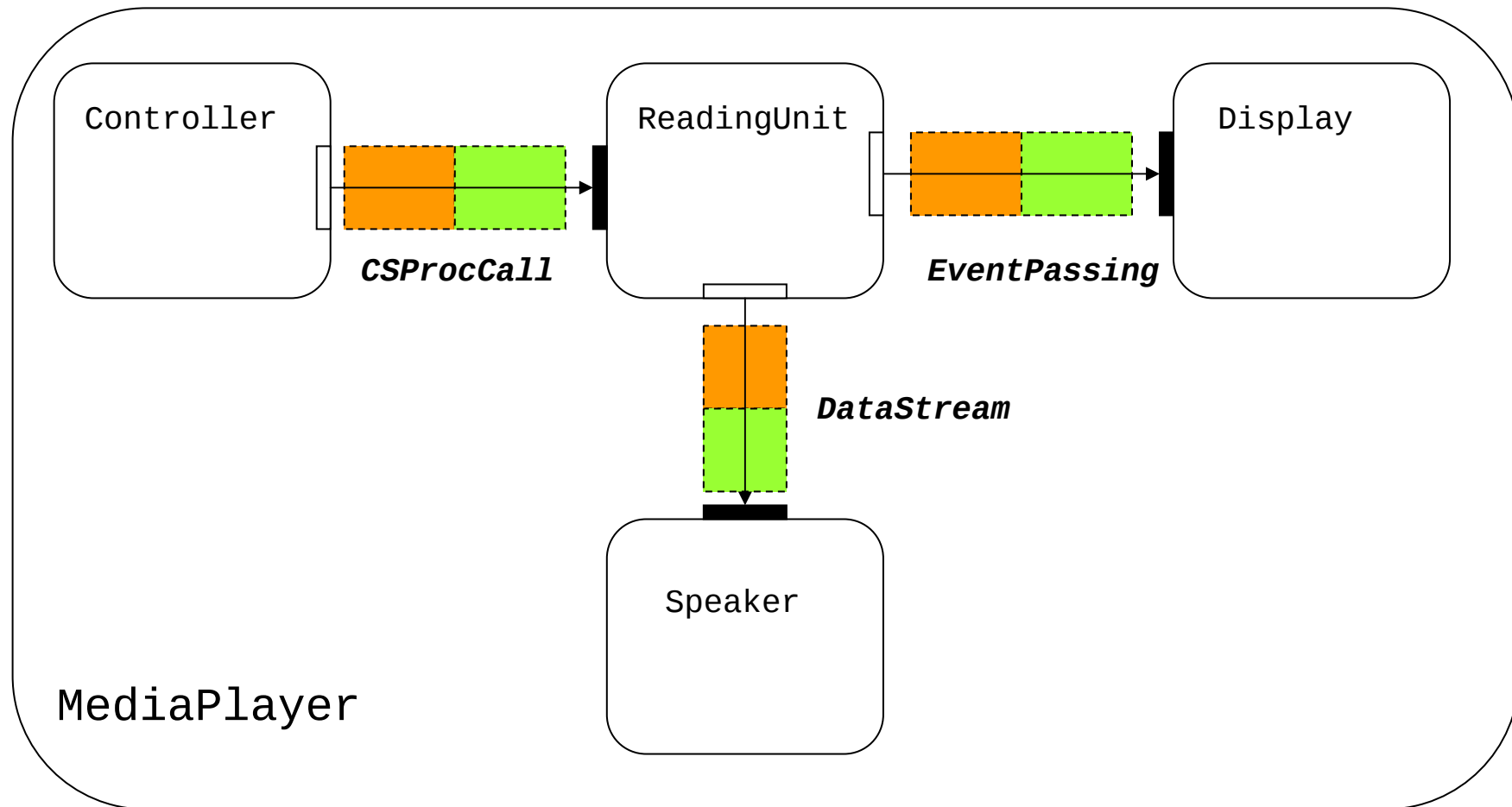
PECOS (PErvasive COmponent Systems)

- Motivation
 - Application du domaine des équipements industriels (ABB)
 - Construction des systèmes embarqués (réactifs)
- Modèle de composants ultra-legers
 - Un langage de description/composition (textuel et graphique)
 - Un environnement de composition
 - Un environnement ultra léger d'exécution
 - Implémentations en C et C++
- Trois types de composants
 - Composant actif (thread)
 - Composant passif
 - Composant à évènement

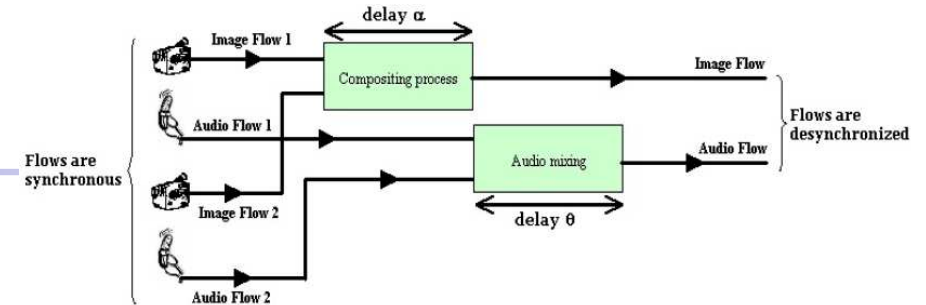
SOFA (Univ. Charles, Pragues)

- SOFA
 - Software Appliances
- Component model
 - DCUP
 - Dynamic Component Updating
- Component distribution/delivery
 - SOFA node(s)
 - SOFA Connectors
 - Cl/Sv, Event Passing, Binary Data Stream
- Multiplatform
 - Java and C++

SOFA Connectors



Osagaia



■ Domaine

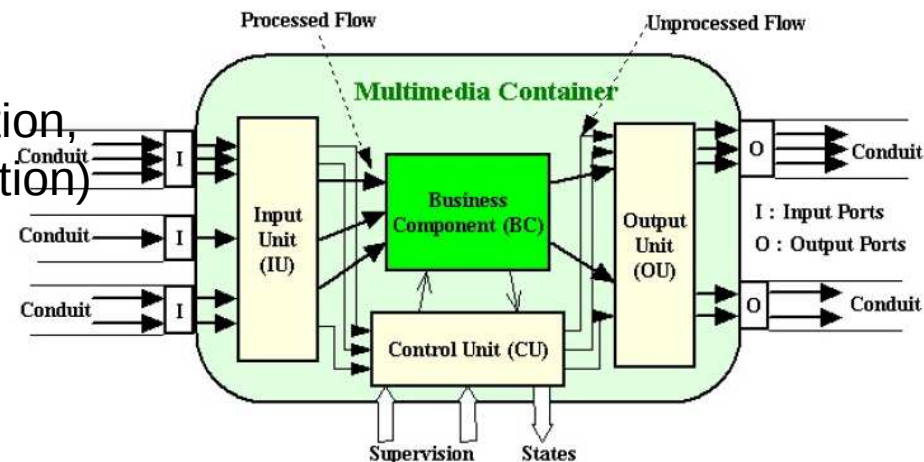
- Multimedia distributed processing
 - Continuous media : Video and audio stream
 - Discret media : subtitle, background picture, ...

■ Principles

- Compose processing stages
preserving the synchronization relationship between media flows

■ Concepts

- Components
 - Business Components
 - Operators (merge, separation, disjunction, conjunction)
- connected by Conduits

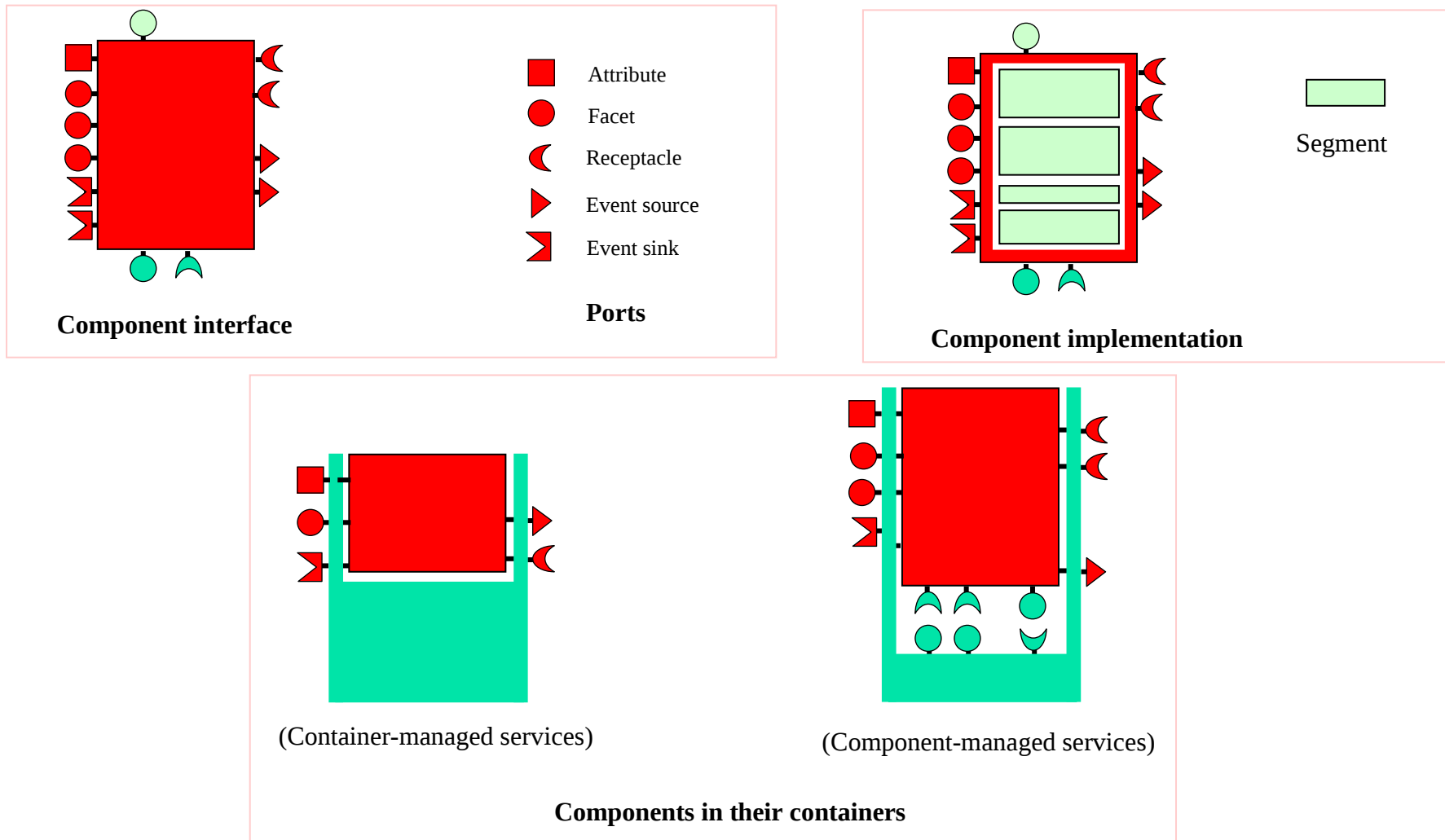


CORBA Component Model (CCM)

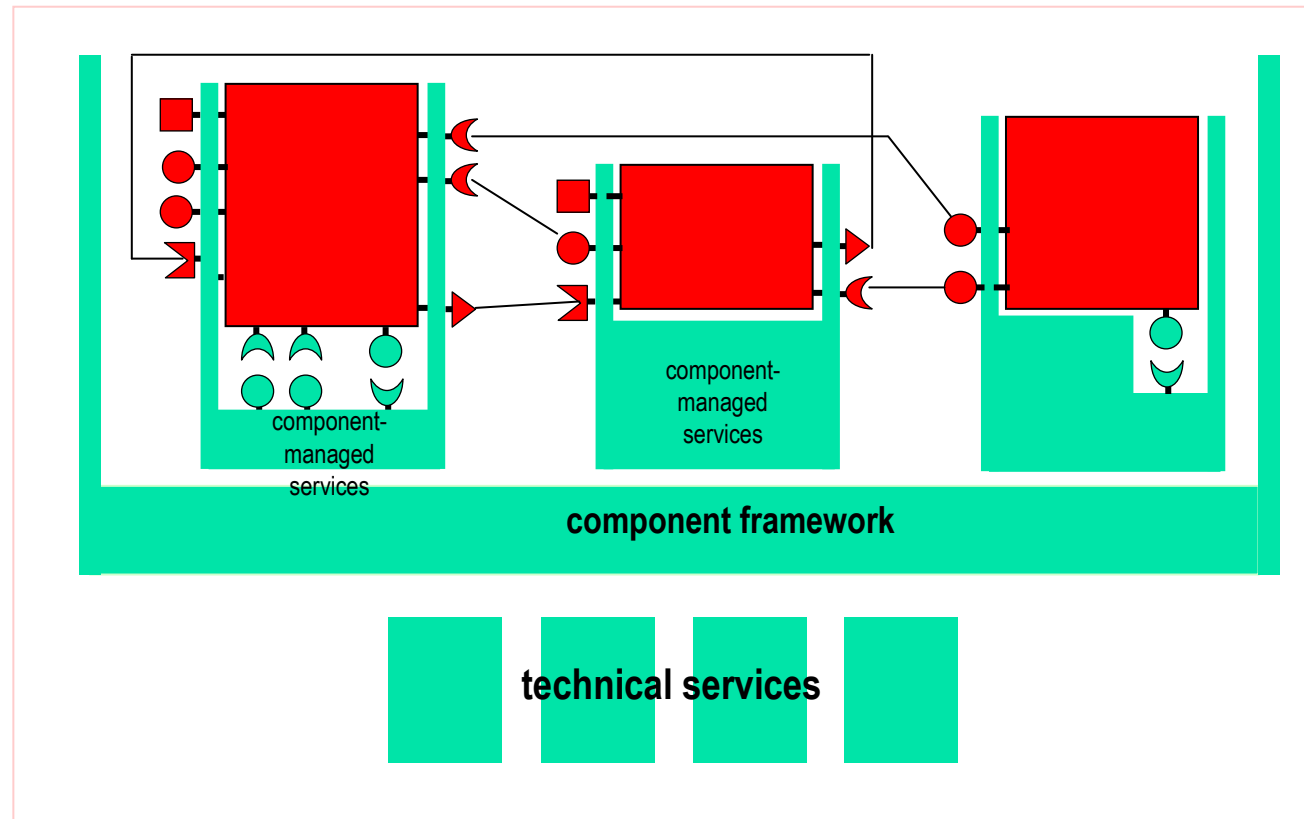
- **Domaine**
 - Enterprise (initialement) → niche des intergiciels distribués temps réel
- **Ports**
 - Client Server : receptacle → facet
 - Event passing : source → sink
- **Type**

Le modèle d'utilisation	L'API du container	La catégorie du component	La référence d'objet	Esclave/OID
Sans état	Session	Service	Transient	1:N
Conversa-tional	Session	Session	Transient	1:1
Durable	Entité	Processus	Persistant	1:1
Durable	Entité	Entité	Persistant	1:1

CCM Corba Component Model



CCM Corba Component Model



EJB Enterprise JavaBeans

- Central dans la spécification Java Enterprise Edition
 - Très largement connu, enseigné, et répandu
 - Malgré ces défauts (conventions de programmation)
- Domaine
 - Composants logiciels écrits en Java implémentant la logique métier des applications d'entreprise coté serveur
- Services non fonctionnels
 - *Bean-managed* versus *Container-managed*
 - Liste fixe :
 - Persistance, Transaction, Contrôle de concurrence, Sécurité, Gestion mémoire, Montée en charge (distribution), Tolérance aux pannes (réplication)
- Plusieurs générations
 - EJB1
 - EJB2 (XML, Relation, MDB ...) + Enterprise MediaBeans
 - EJB3 (approche à la POJO, Annotations+Generic Java 5, Persistance JPA, JDO, ...)

Exemple de EJB 3.0

```
@Entity @NamedQuery(name = "tousLesAuteurs", query = "select o FROM Auteur o")
public class Auteur implements Serializable {
    /** Clef primaire (generee automatiquement).*/
    private long id;

    /** Nom de l'auteur.*/
    private String nom = null;

    /** Liste de livres ecrits par l'auteur.*/
    private Collection<Livre> livres;

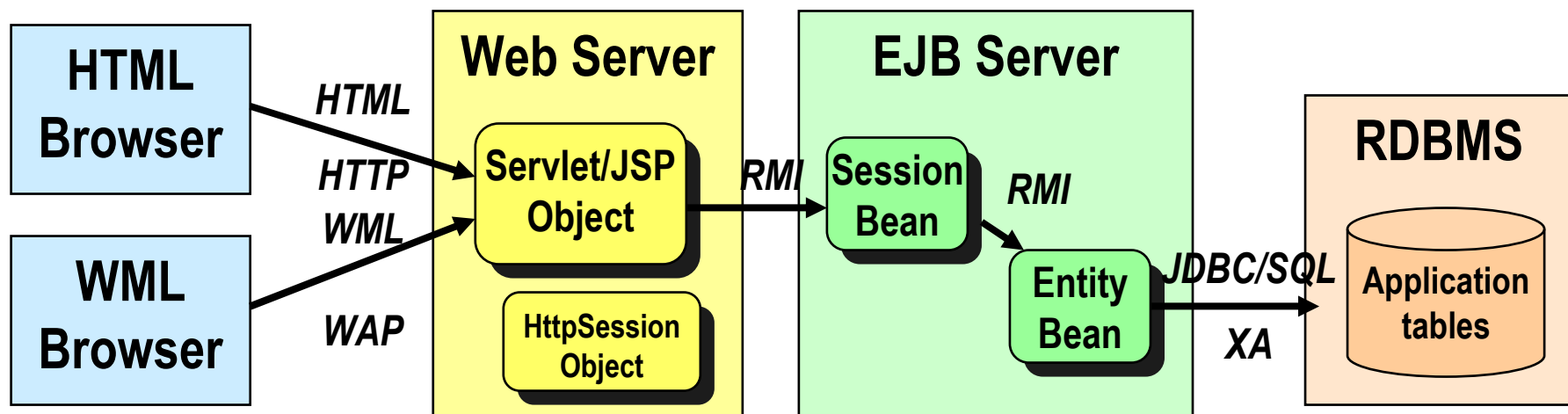
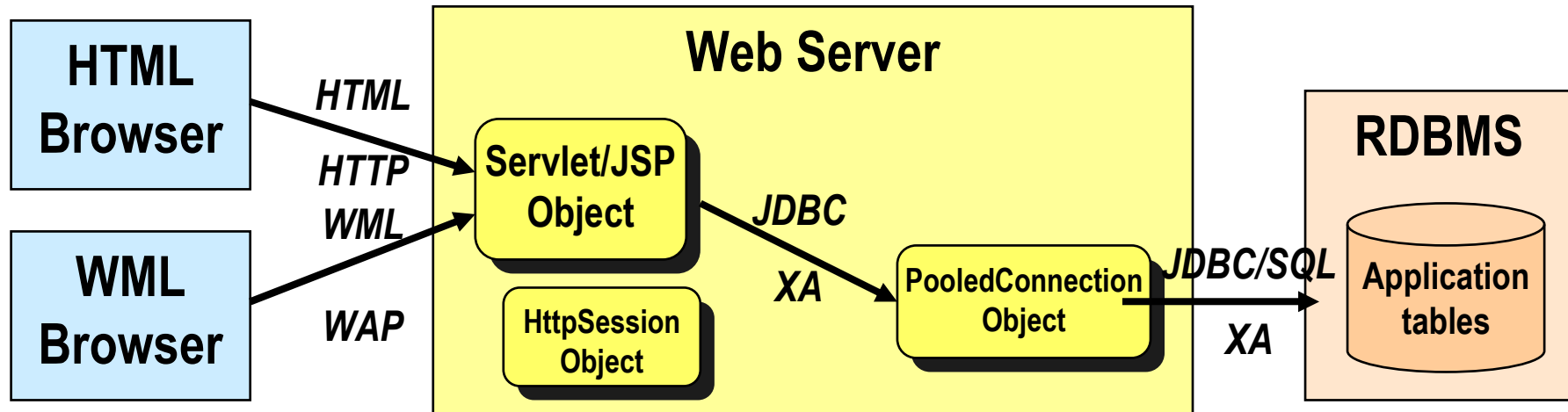
    /** Constructeur par default.*/
    public Auteur() {
        livres = new ArrayList<Livre>();
    }

    /** Constructeur utilise pour initialiser cet entity bean.*/
    public Auteur(final String nom) {
        this(); setNom(nom);
    }

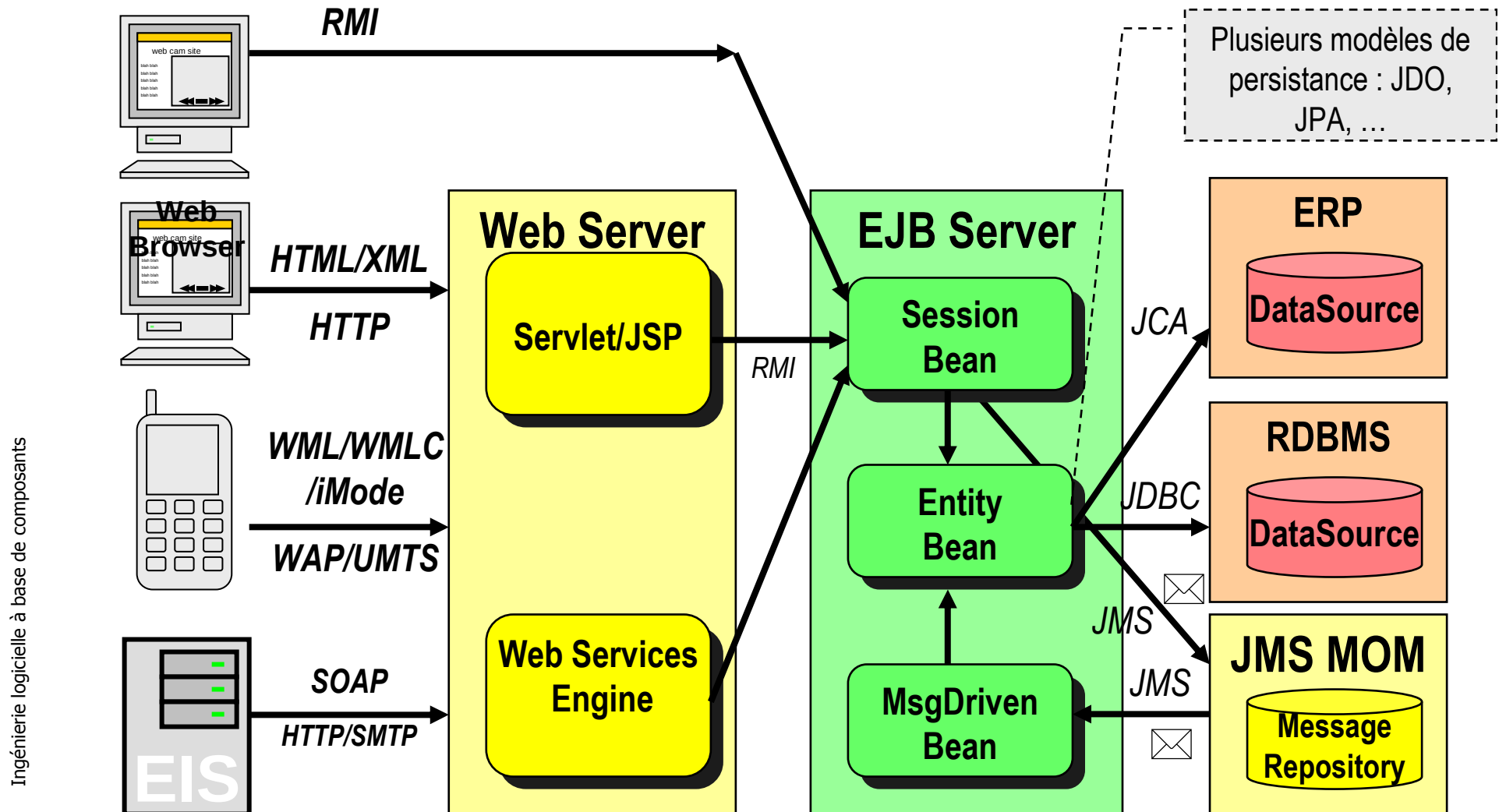
    @OneToMany(mappedBy = "auteur", fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    public Collection<Livre> getLivres() {
        return livres;
    }

    /** Ajout d'un livre avec son titre.*/
    public void ajoutLivre(final String titre) {
        Livre livre = new Livre();
        livre.setTitre(titre);
        livre.setAuteur(this);
        livres.add(livre);
    }
    ...}
```

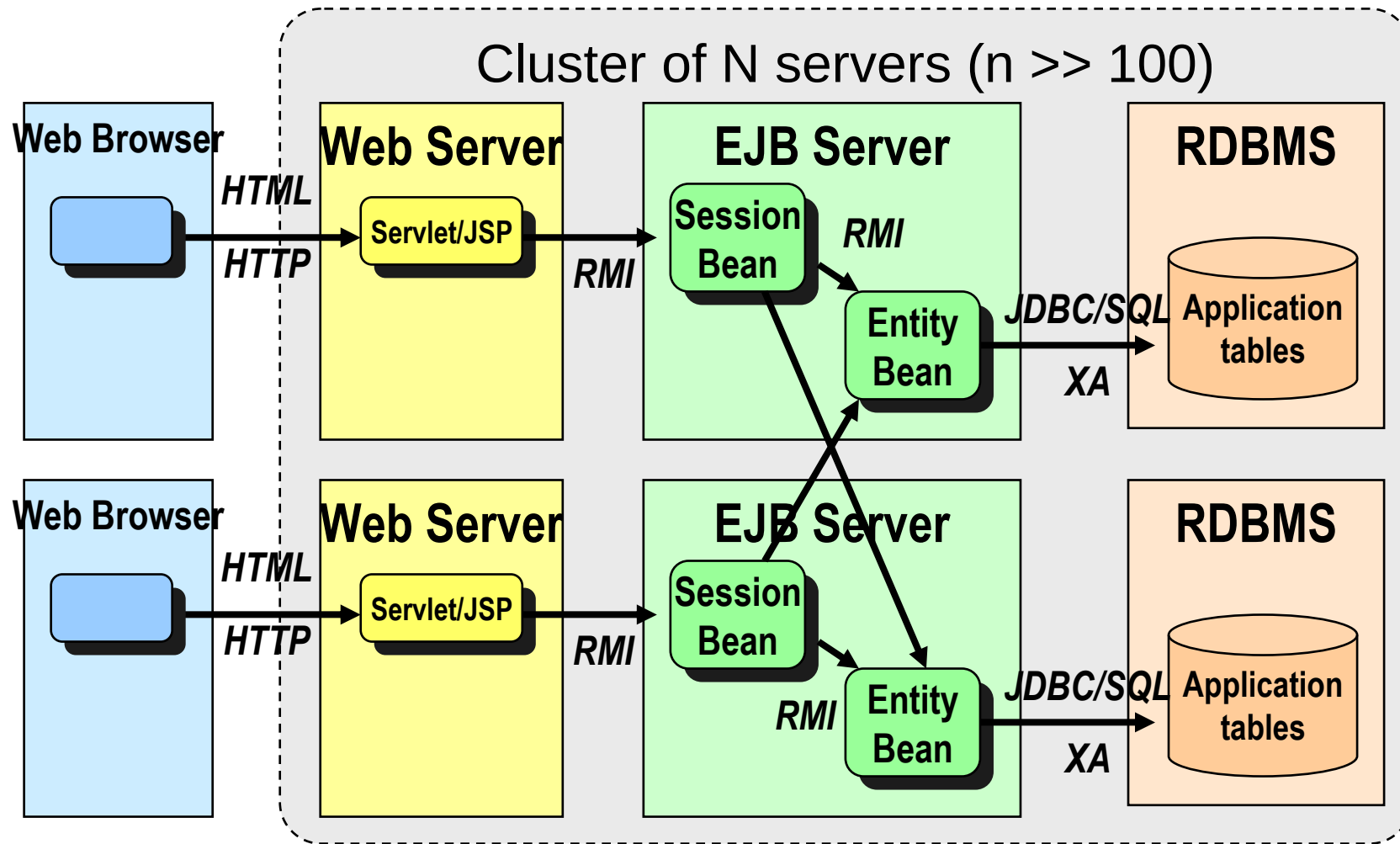
Architecture d'une application d'entreprise avec et sans EJB



Les EJB dans une architecture Java Enterprise Edition

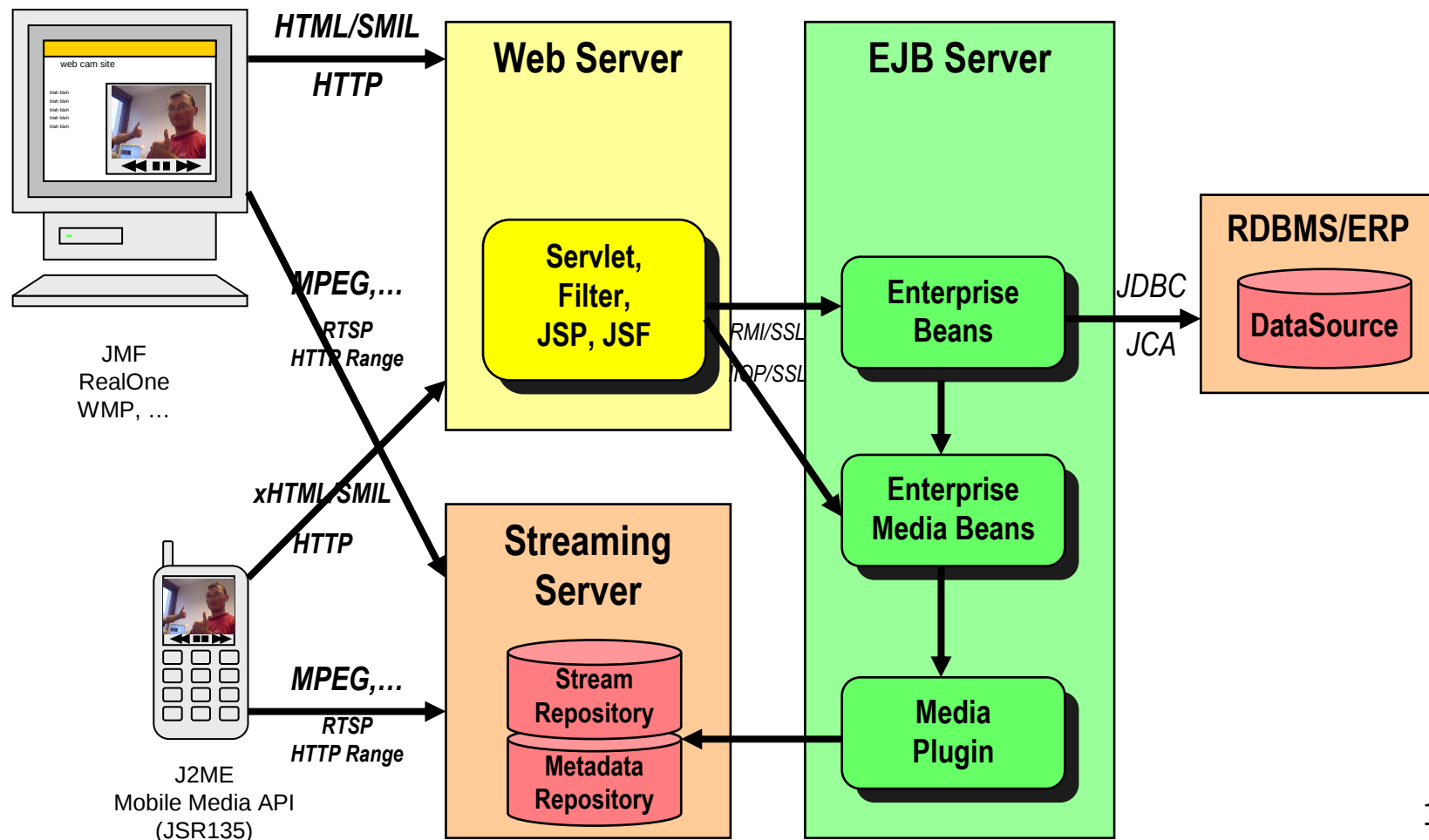


Composants EB distribués sur une grappe de serveurs



Composants Enterprise Media Beans (EMB)

- Intégration de contenu multimédia dans les applications JavaEE (adaptation du contenu, ...)



Web Beans (JSR 299)

■ Goal

- unify the JSF managed bean component model with the EJB component model, resulting in a significantly simplified programming model for web-based applications (JavaEE).

■ Characteristics

- Descriptions using annotations or XML descriptors
- IoD

■ Example

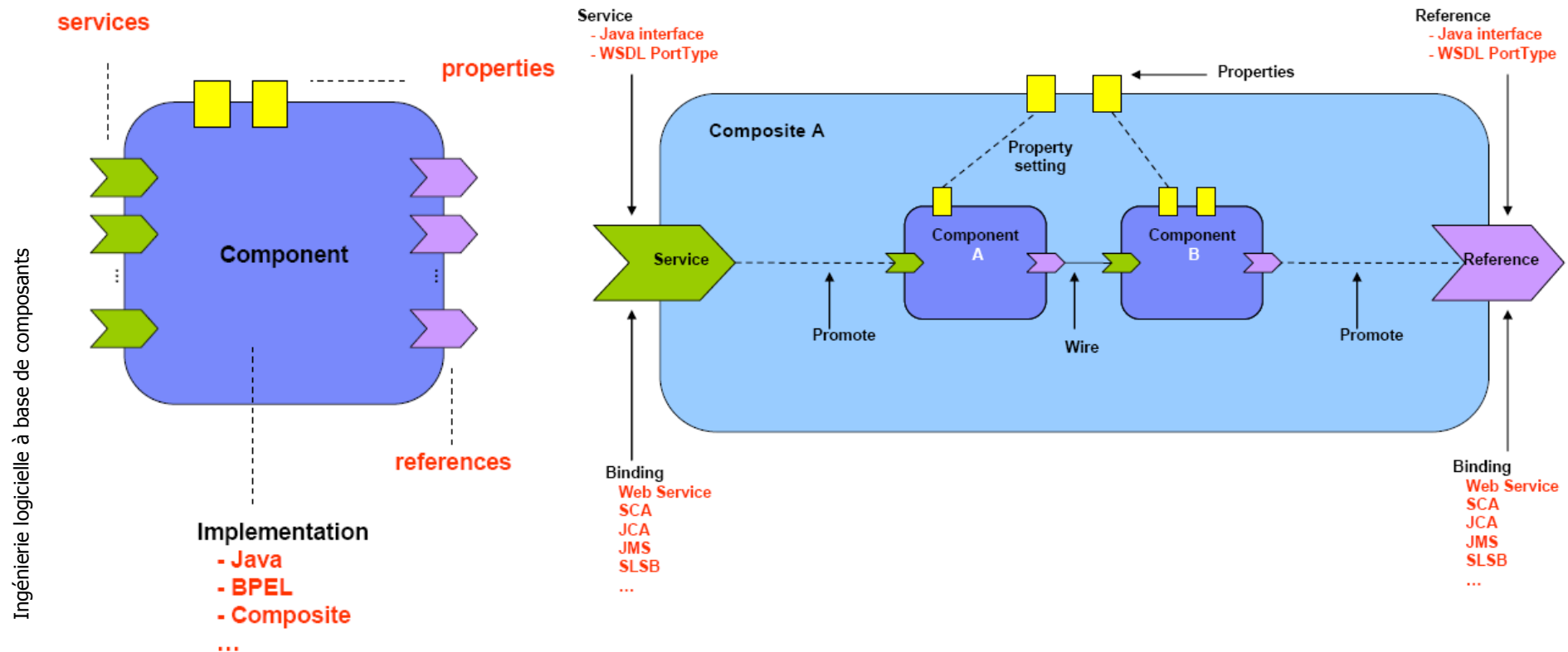
```
@ConversationScoped @Component public class Order {  
    @In Product product;  
    @Current User customer;  
    ...  
}
```

Service Component Architecture (SCA)

- Modèle de composants pour structurer les applications orientées service
 - Initiative : IBM, Oracle, BEA, SAP, Sun, TIBCO, ...
 - Spécifications : 12/2005, v1 03/2007
 - Implémentations : Apache Tuscany, Newton, fabric3, ...
- Assembly model
 - how to define structure of composite applications
- Component implementations specifications
 - how to write business services in particular languages
 - Java, C++, Python, PHP, Spring, BPEL, EJB SLSB, ...
- Binding specifications
 - how to access services
 - Web services, JMS, RMI-IIOP, REST, ...
- Policy framework
 - how to add infrastructure services
 - security, transactions, reliable messaging, ...

SCA

Composant et Assemblage



Assemblage SCA

- service/référence
 - type
 - langage de définition d'interface (IDL) : Java ou WSDL
 - *binding* : association avec une technologie de communication
 - Web Service, RMI-IIOP, JMS, appel local intra-JVM
- liens
 - référence vers service : *wire*
 - service/service ou référence/référence : lien de promotion (*promote*)

SCA

Modèle de programmation

- Assembly model
 - ADL basé XML
- Component implementations
 - 1 ensemble de règles de programmation par langage
 - Java : utilise intensément annotations Java 5 (comme EJB3, ...)

- Exemple avec Java

```
@Service(AccountService.class)
public class AccountServiceImpl implements AccountService {
    @Reference public AccountDataService accountDataService;
    @Reference public StockQuoteService stockQuoteService;
    @Property public String currency;
    public AccountReport getAccountReport(String s) { ... }
}
```

CCA Common Component Architecture

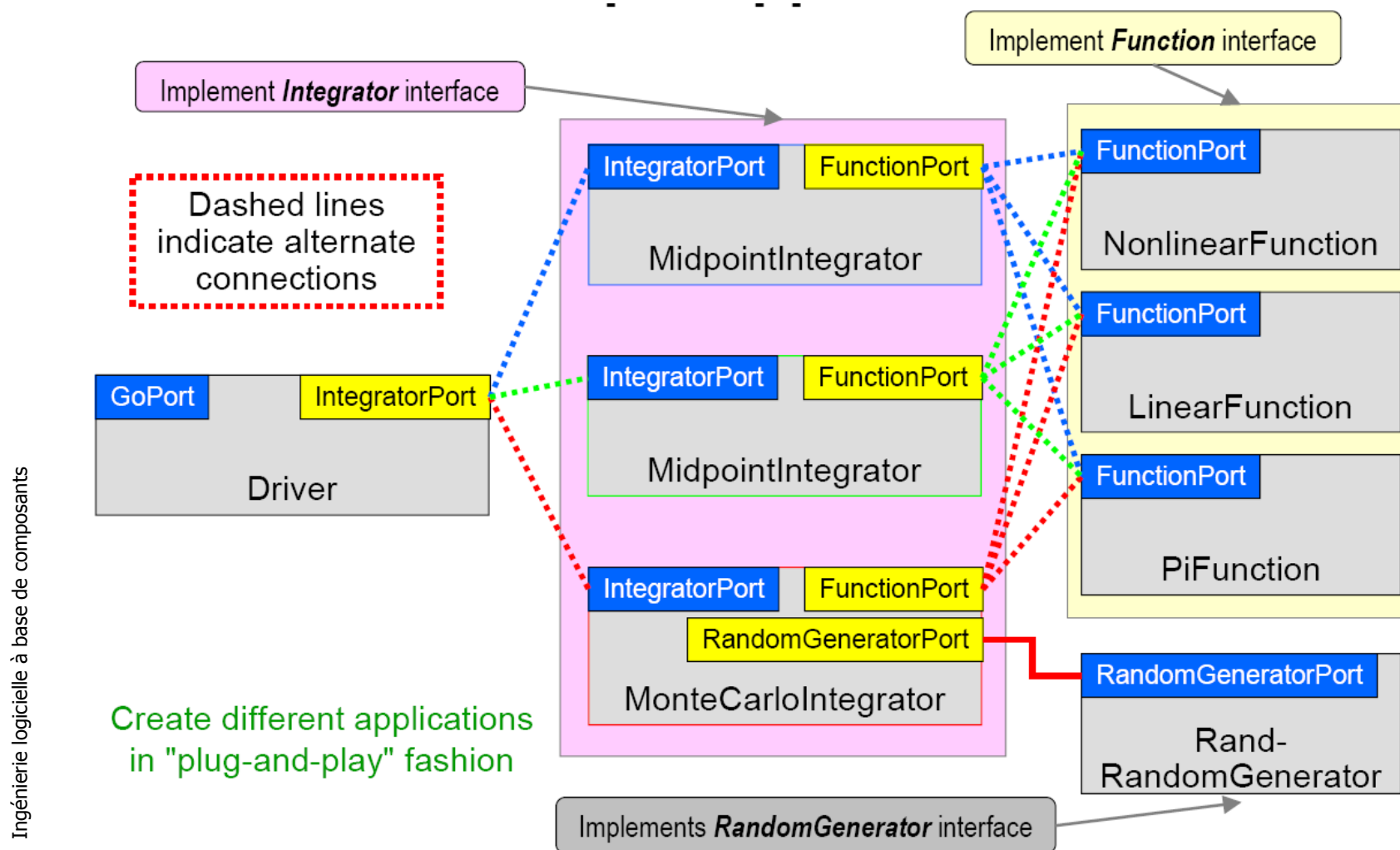
■ Domaine

- Initialement projet du US DoE
- high-performance computing
 - Scientific codes (matrix, ...)
 - sur supercalculateur SMP au grilles de machines

■ Modele

- Philosophie minimaliste
- Langage d'interface : S(cientific)IDL
- Multi-langages : C++, Fortran 90, ...
- Ports fournis et requis, Connecteurs multiples (collecteur, répartiteur, ...), Composants parallèle, Pipeline multi-étages, ...
- Mais pas d'ADL, pas de composants hierarchiques

CCA Exemple d'assemblage



Grid Component Model (GCM)

- Domain
 - grid distributed computing
 - SPMD (Single Process, Multiple Data)
- Model
 - Extended the Fractal CM
 - Hierarchical, ADL, ...
 - Notions
 - Virtual Node (VN)
 - Component can be spread on several VN
 - Distributed computing oriented interfaces (client/server)
 - multicast, gathercast, MxN ...
 - Misc
 - MPI process wrappers, « hybrid » CCA-GCM component, ...
 - Dynamic controllers
 - componentized containers (membranes) using DC interfaces
 - Pluggable NF server interfaces (ie self-* manager, monitoring, ...)
- Implementations
 - Supported by the CoreGrid NoE
 - GCM/ProActive
 - GCM' Mapping component $\leftrightarrow$ ProActive' Active Objects (Java)
 - Proposed at ETSI standardization
 - ...

GCM

Distributed computing oriented interfaces

- Binding with Future features
 - async method calls with full-fledge Wait-by-Necessity
- Multicast interfaces
 - Parallelism & Asynchronism
 - Dispatch invocations wrt binding
 - Data distribution (invocation parameters)
 - Broadcast or scattering (using a distribution/split functions)
- Gathercast
 - ~join invocation
 - Timeout / drop policy
 - Aggregation of parameters into lists +function
- MxN interfaces
 - Optimize Multicast/Gathercast components
 - Direct communications bypassing composites

Spring Framework

- TODO

Avalon

- TODO

Plexus

- TODO

Qi4j

<http://www.qi4j.org/>



- Qi4J is an Java implementation of Composite Oriented Programming.
 - The short answer is that Qi4j is a framework for domain centric application development, including evolved concepts from AOP, DI and DDD.
 - Qi4j is an implementation of Composite Oriented Programming, using the standard Java 5 platform, without the use of any pre-processors or new language elements.
- Composite Oriented Programming builds on some principles that are not addressed by Object Oriented Programming at all.
 - Behavior depends on Context
 - Decoupling is a virtue
 - Business Rules matters more.
 - Classes are dead, long live ***interfaces***.

Bibliographie

- Szyperski, C., Component Software: Beyond Object-Oriented Programming, Publ. Addison-Welsey, Dec. 1997, ISBN 0-201-17888-5
- Szyperski, C. (2002). Component Software : Beyond Object-Oriented Programming. Addison-Wesley. 2nd ed. 2002, 589 pp.
- Ivica Crnkovic and Magnus Larsson (Editors), Building Reliable Component-Based Software Systems, Artech House Publishers, July 2002, ISBN 1-58053-327-2
 - <http://www.idt.mdh.se/cbse-book/>
- Shaw, M. and Garlan, D. Software Architecture: Perspectives on an Emerging Discipline, Pub. Prentice-Hall, 1996.
- Kung-Kiu Lau, Zheng Wang, "Software Component Models," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 709-724, Oct., 2007,
 - <http://doi.ieeecomputersociety.org/10.1109/TSE.2007.70726>

Conferences

- Principalement
 - CompArch Component-Based Software Engineering (CBSE)
 - <http://www.informatik.uni-trier.de/~ley/db/conf/cbse/index.html>
 - Euromicro Conference on Software Engineering and Advanced Applications (SEAA) track CBSE
 - <http://www.informatik.uni-trier.de/~ley/db/conf/euromicro/index.html>
 - Software Composition (SC)
 - <http://www.informatik.uni-trier.de/~ley/db/conf/soco/index.html>
 - En France: Journées Composants (JC)
- Egalement
 - International Conference on Software Engineering (ICSE)
 - <http://www.informatik.uni-trier.de/~ley/db/conf/icse/index.html>
 - ETAPS
 - OOPLSA
 - GPCE
 - AOSD
 - En France : LMO
- Journaux
 - *IEEE Software*
 - *IEEE Transactions on Software Engineering*
 - *Revue L'Objet* editée par Hermes (en français)

Travaux pratiques

Construction d'un mashup au moyen de composants Microsoft PopFly

Microsoft® Popfly (<http://www.popfly.com/>) is the fun, easy way to build and share mashups, gadgets, games, Web pages, and applications.

TODO

Alternatives

Yahoo Pipe, Google ???

PopFly – MashUp Creator

The screenshot shows the Microsoft Popfly MashUp Creator interface. At the top, there's a navigation bar with the Microsoft Popfly logo, a status indicator 'Unsaved Mashup', and buttons for 'Save', 'Help', and 'Sign Out'. Below this are tabs for 'Create Stuff', 'My Stuff', 'Help Stuff', and 'Overview Stuff'. The main workspace features a toolbar with 'Edit', 'Html', and 'Run' buttons. On the left, a 'Blocks' sidebar lists various services categorized by 'New & Updated', 'Display', 'Fun & Games', 'Images & Video', and 'Local Information'. The 'Upcoming' block is currently selected in the workspace. On the right, a 'Help' sidebar provides instructions on how to use blocks and create mashups.

Ingénierie

« Drop » d'un composant de recherche d'évènements (upcoming)
 – Paramètre/input : music -output : id,name, description, **géo localisation**,...

Merci à Jean-Sébastien Sottet pour ces transparents

PopFly – MashUp Creator

Ingénierie

« Drop » d'un composant « Virtual Earth » : carte du monde – input donnée géo localisation (longitude/latitude), title, url, etc. - output ?

Merci à Jean-Sébastien Sottet pour ces transparents

PopFly – MashUp Creator

Microsoft® Popfly™

Unsaved Mashup

Save Help Sign Out JayEsse

Create Stuff My Stuff Help Stuff Overview Stuff

Edit <> Html Run

Blocks

newsdisplay addressoutp. X

3 Search Results

★★★★★

Indeed Jobs

News Reader

News Updater

Upcoming

Virtual Earth

Missing Key

Help

More Help

How do I...? Videos Samples

When a block is on the surface you can zoom in on it to set what the block does. Do this by double clicking on it or clicking the wrench icon.

But to make a mashup, what you really need is more than 1 block!

Try searching for more blocks that go with the block you have; or if the block you have supports it, you can click on the light bulb for suggestions.

Mashups work by taking the output of one block and using it as input for the next block. For example you can use the Facebook block to get a list of your friends and then connect it to the Virtual Earth block to place them on a map.

You connect blocks by clicking on them and then clicking on the block that you want to send the output to.

© Microsoft 2009/10 Dariusz Lars Rahua Dennis Ahura

Ingénierie

Relation entre l'output du composant upcoming et l'input de Virtual Earth

Merci à Jean-Sébastien Sottet pour ces transparents

PopFly – MashUp Creator

Microsoft® Popfly™

Unsaved Mashup

Save Help JayEsse Sign Out

Create Stuff My Stuff Help Stuff Overview Stuff

Edit Html Run

Blocks

newsdisplay addressoutp X

3 Search Results

★★★★★

Indeed Jobs

News Reader

News Updater

Upcoming

Missing Key

Virtual Earth

Operation: addPushpin

latitude: latitude from Upcoming

longitude: longitude from Upcoming

url:

title: name from Upcoming

description: description from Upcoming

centerMapOnPushpin: true

Help

More Help

How do I...? Videos Samples

When a block is on the surface you can zoom in on it to set what the block does. Do this by double clicking on it or clicking the wrench icon.

But to make a mashup, what you really need is more than 1 block!

Try searching for more blocks that go with the block you have; or if the block you have supports it, you can click on the light bulb for suggestions.

Mashups work by taking the output of one block and using it as input for the next block. For example you can use the Facebook block to get a list of your friends and then connect it to the Virtual Earth block to place them on a map.

You connect blocks by clicking on them and then clicking on the block that you want to send the output to.

Ingénierie

Virtual Earth :

input latitude -> composant Upcoming
 input longitude -> composant Upcoming
 etc.

Merci à Jean-Sébastien Sottet pour ces transparents

PopFly – MashUp Creator

The screenshot shows the Microsoft Popfly web application interface. At the top, there's a navigation bar with 'Microsoft Popfly' logo, 'Unsaved Mashup' status, and icons for 'Save', 'Help', 'JayEsse', and 'Sign Out'. Below this is a secondary bar with 'Create Stuff', 'My Stuff', 'Help Stuff', and 'Overview Stuff' tabs. The main workspace features a toolbar with 'Edit', 'Html', 'Run', 'Debug', and 'Console' buttons. The central area displays a world map with numerous red location pins. A pop-up window is visible over Europe, titled 'Radiohead', containing the text: 'Second of Radiohead's London dates for the summer tour. Tickets £42.50, on sale to the public on Friday December 7th 2007'. On the right side, there's a 'Help' section with a 'More Help' link and a 'How do I...?' section containing links to 'Videos' and 'Samples'. At the bottom right of the map, there's a scale bar indicating '2500 miles' and copyright information: '© 2007 Microsoft Corporation © 2007 NAVTEQ Image courtesy of NASA'.

Ingénierie

Résultats : combinaison des données sur la carte du monde : données de géolocalisation + titre + description (exemple radiohead)

Merci à Jean-Sébastien Sottet pour ces transparents