ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

La plate-forme dynamique de service OSGi

Didier Donsez

Université Joseph Fourier (Grenoble 1)

IMA IMAG/LSR/ADELE + SARDES (2006-2007)

Didier.Donsez@imag.fr

Didier.Donsez@ieee.org

Sommaire

- Motivations et Rappel
- Conditionnement et Service
- Enregistrement et recherche de services
- Composants
- Services standards
- Acteurs, Concurrences et Perspectives

Qu'est ce que OSGi™ ?

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 pilotent la R4)
- et Eclipse pour les plugins de son IDE !
- et maintenant Apache pour ses serveurs

Qu'est ce que OSGi™ ?

Histoire

- Mars 1999 : Fondation de l'OSGi Alliance
- Novembre 1999: SUN transfère le JSR008 du JCP à OSGi
- 1.0 : Mai 2000 (189 pages)
- 2.0 : Octobre 2001 (288 pages)
- 3.0 : Mars 2003 (602 pages)
- 4.0: October 2005 (1000 pages)

Remarque

- *Open Services Gateway Initiative est obsolète*

Principales propriétés du canevas OSGi

- Chargement/Déchargement de code dynamique
  - Langage Java
- Déploiement dynamique d'applications sans interruption de la passerelle
  - Installation, Lancement, Mise à jour, Arrêt, Retrait
- Résolution des dépendances versionnées de code
- Programmation orientée service

- Vise des systèmes à mémoire restreinte
  - J2ME/CDC

Rappel sur la programmation OO

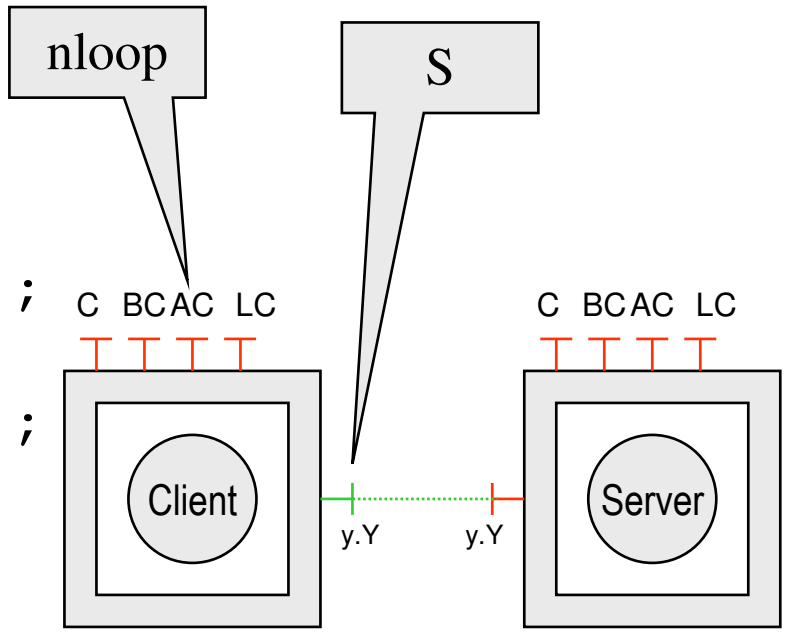
- Un client C invoque N fois la méthode execute() d'un serveur S

- `S s=new S()`
- `C c1=new C(s, N);`
- `C c2=new C(s, N);`

- Problème: Architecture ? Configuration ?

- **Un client C invoque N fois la méthode execute() d'un serveur S**

```
S s=SFactory.create()
C c1=SFactory.create();
C c2=SFactory.create();
c1.setProperty("nloop",N);
c1.bind("S",s);
c2.setProperty("nloop",N);
c2.bind("S",s);
s.start()
c1.start();
c2.start();
…
```



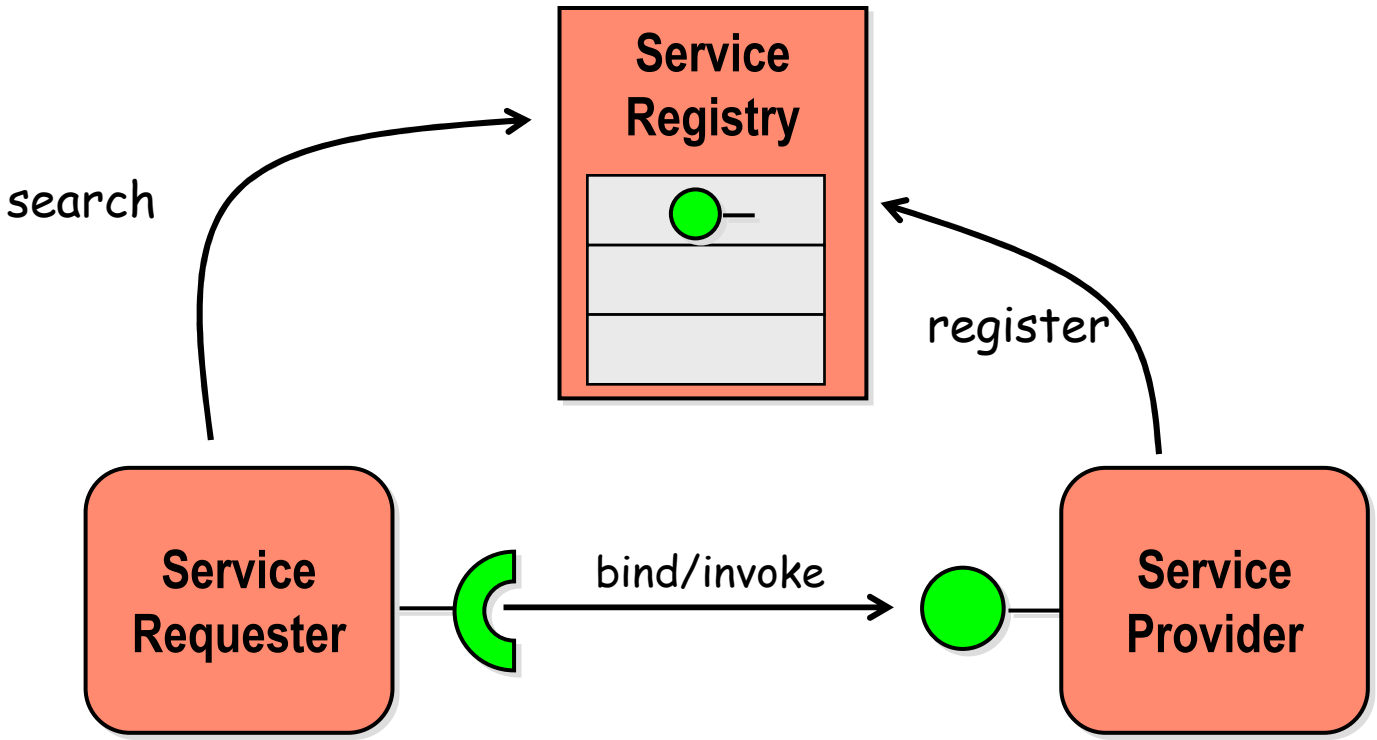
Rappel sur la programmation Composant

- …
- `S s2=SFactory.create()`
- `c2.stop();`
- `c2.bind("S",s2);`
- `s2.start()`
- `c2.start();`

- **Problème: Multi-domaines d'administration (WS)**

Rappel:
Architecture orienté service (SOA)

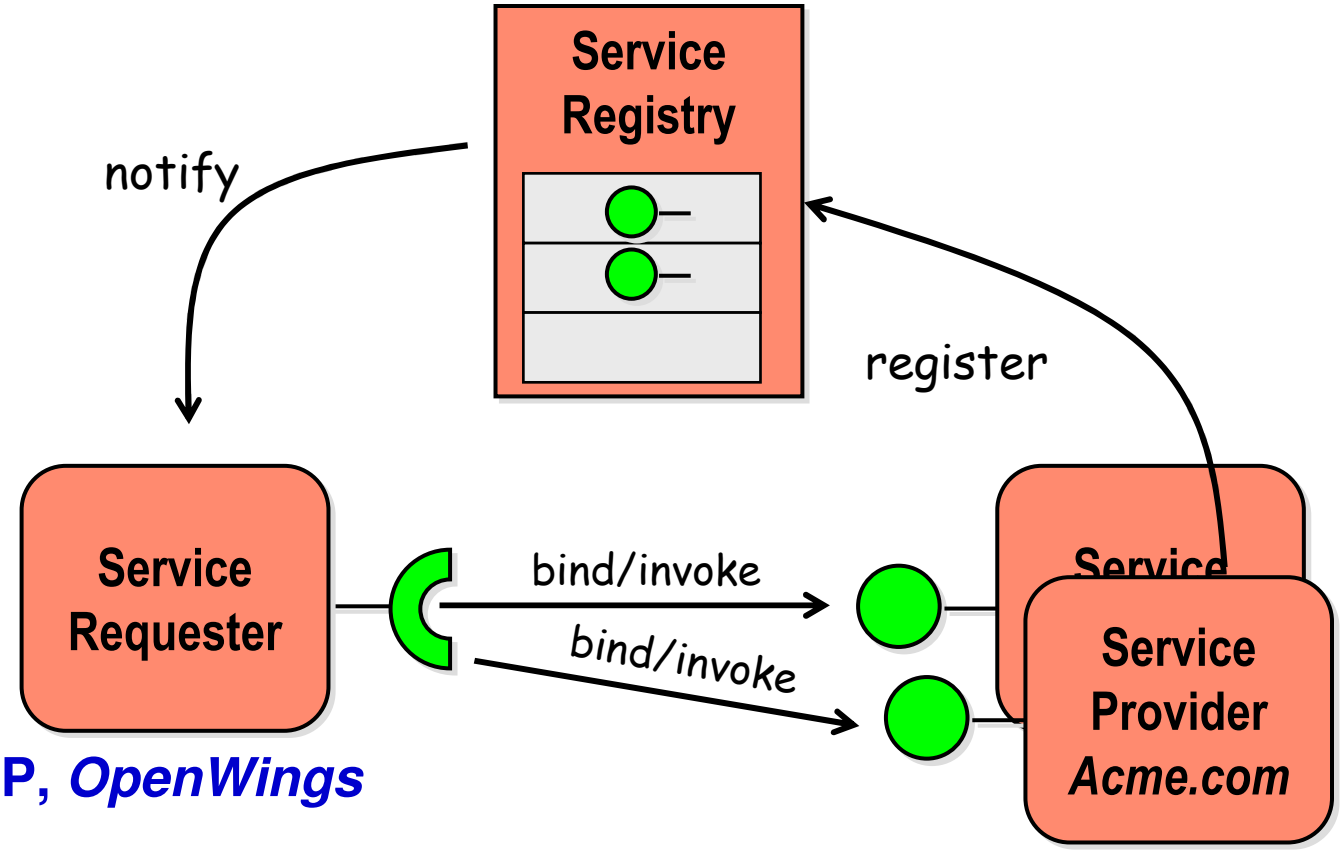
- Les services (contrats) sont « invariants »



- WebServices, TORBA, …

Rappel:
SOA Dynamique

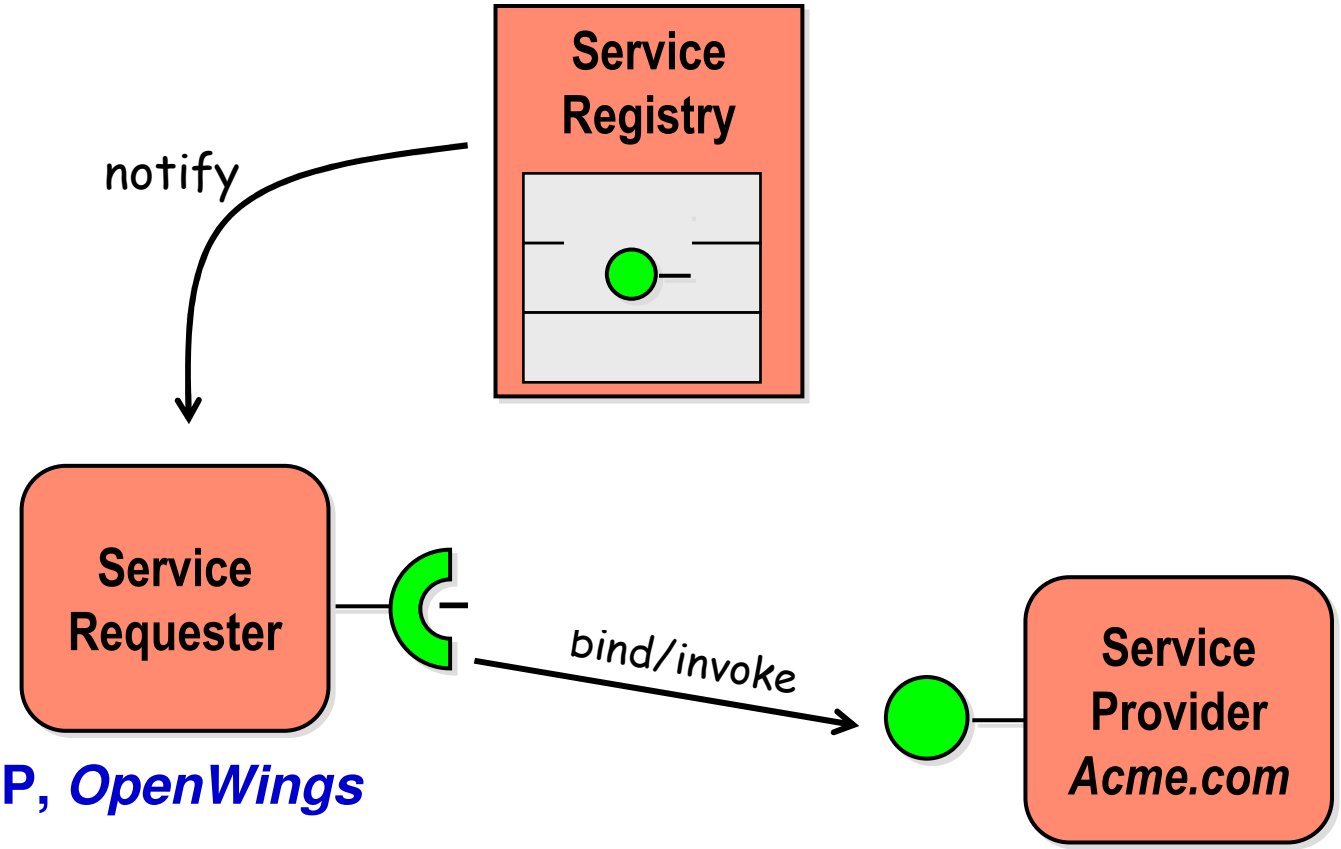
- Arrivée dynamique de nouveaux services



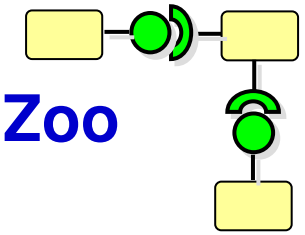
- JINI, UPnP, *OpenWings*
- OSGi

Rappel:
SOA Dynamique

- Retrait dynamique de services utilisés



- JINI, UPnP, *OpenWings*
- OSGi



Dynamic Service Platform Zoo

	Invocation	Removal	Registry Type	Programming Language
JINI	Remote (RMI)	Lease	Distributed (ad-hoc)	Java
OpenWings	Remote (RMI IIOP)	Connector	Distributed (?)	Java
CORBA CosTrading	Remote (IIOP)	No	Distributed (?)	all
UPnP V1	Remote (HTTP/SOAP1.0)	Message Bye	Distributed (ad-hoc)	all
Web Services DPWS	Remote (HTTP/SOAP1.2)	No Message Bye	Centralized (UDDI) WS-Discovering	all
SLP / DNSSD	/	Message Bye	Distributed	all
OSGi	Locale (Référence)	Java Event	Centralized	Java

ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

---

OSGi

Modèle d'administration et Domaines d'application

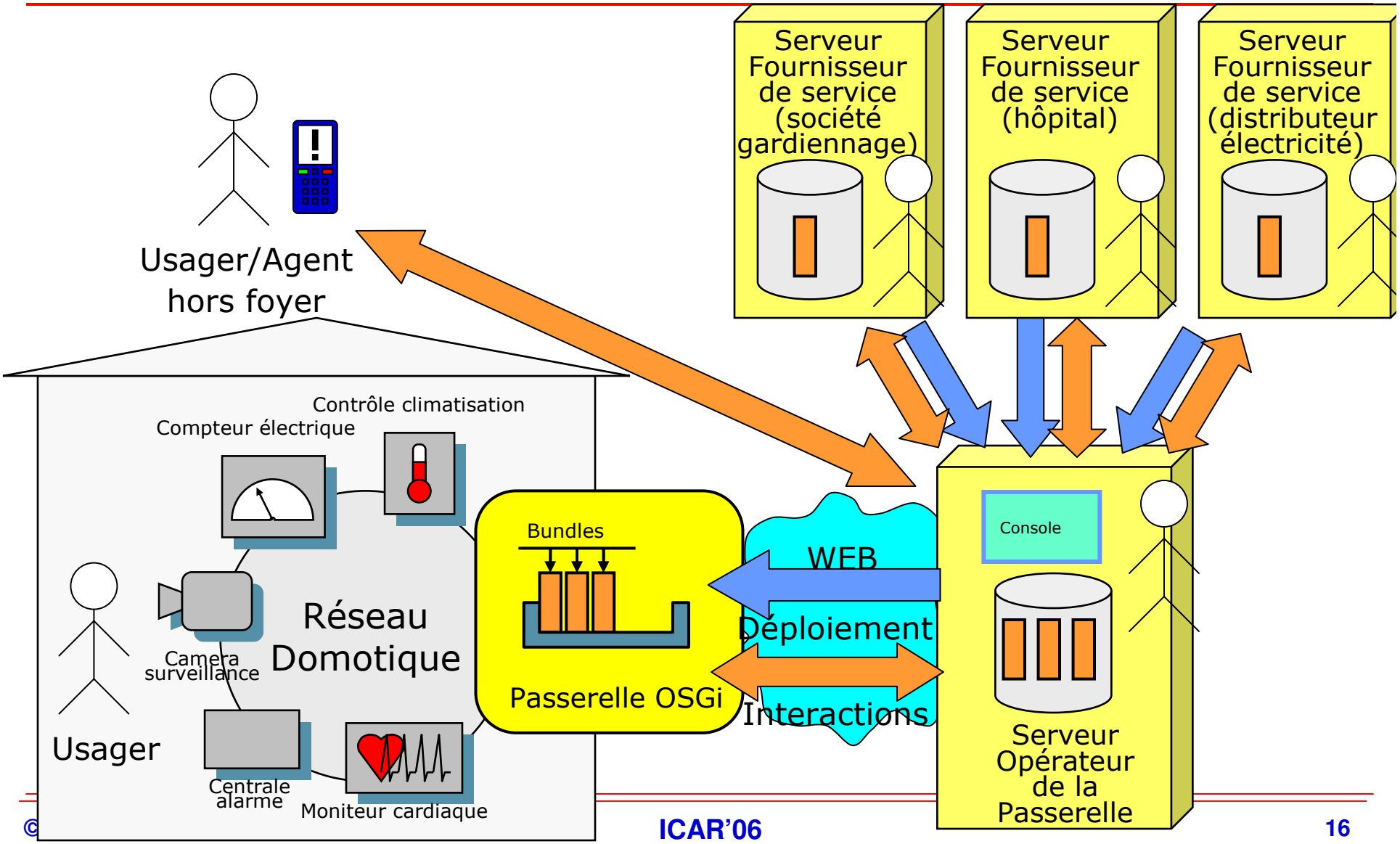
Domaines d'application

- **Systèmes embarqués**
  - Véhicule de transport (*automotive*)
  - Passerelle résidentiel/domotique/immotique
  - Contrôle industriel
  - Téléphonie mobile

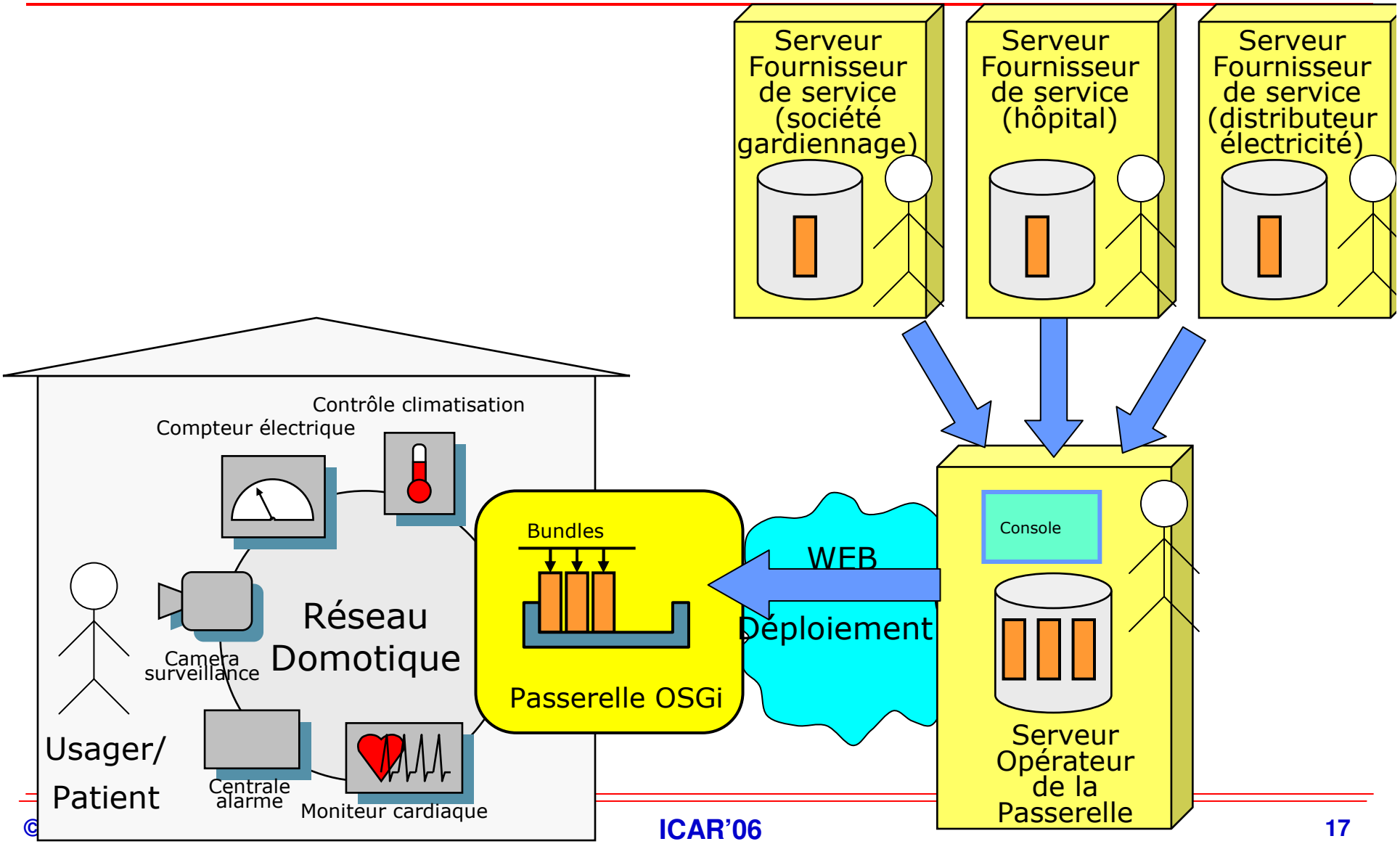
- **Cependant**
  - Tout concepteur d'application est gagnant à distribuer son application sous forme de plugins conditionnés dans des bundles OSGi

  - Cela évite l'enfer du CLASSPATH
    - CLASSPATH, lib/ext du JRE ou J2SESDK, …

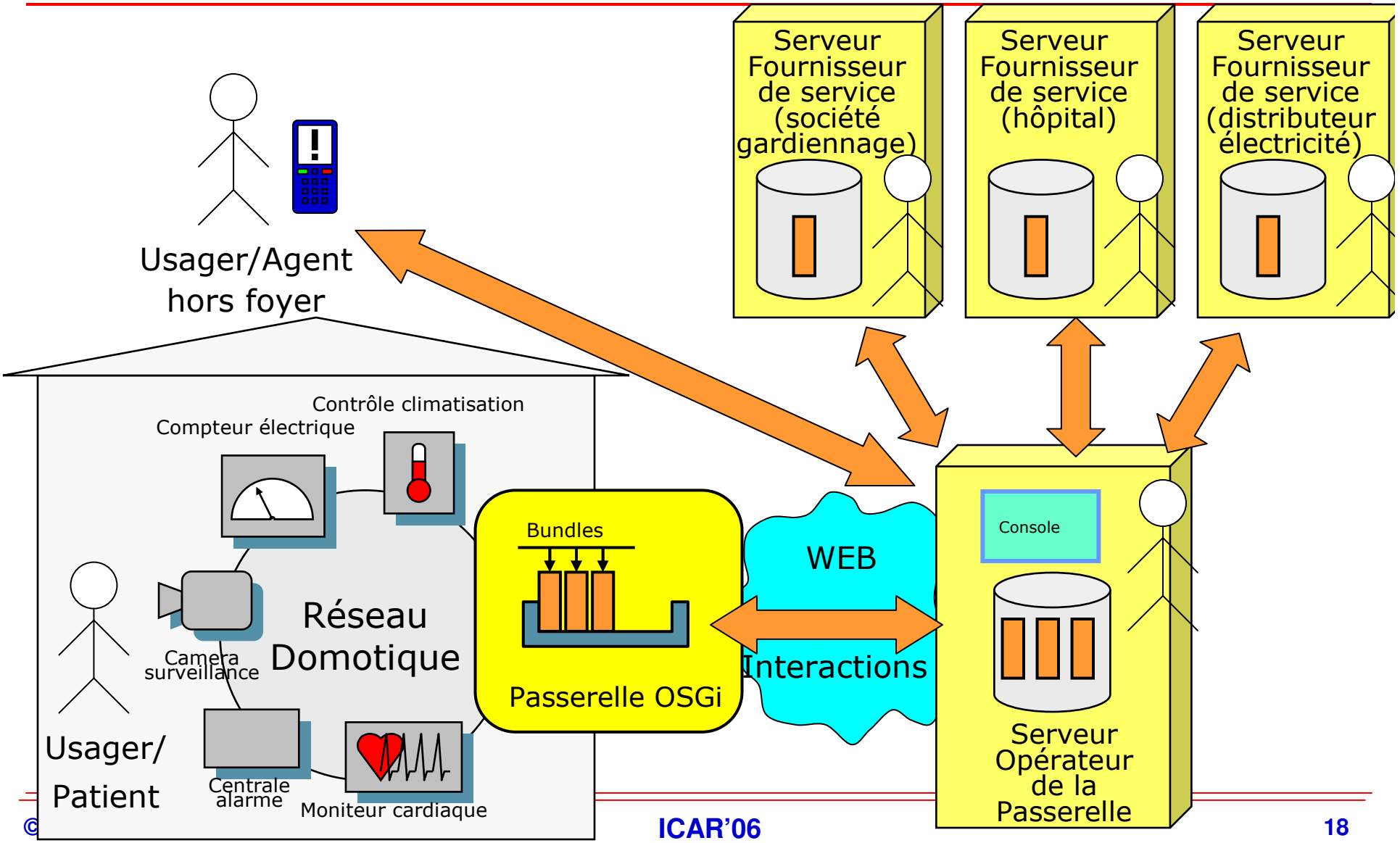
Architecture générale



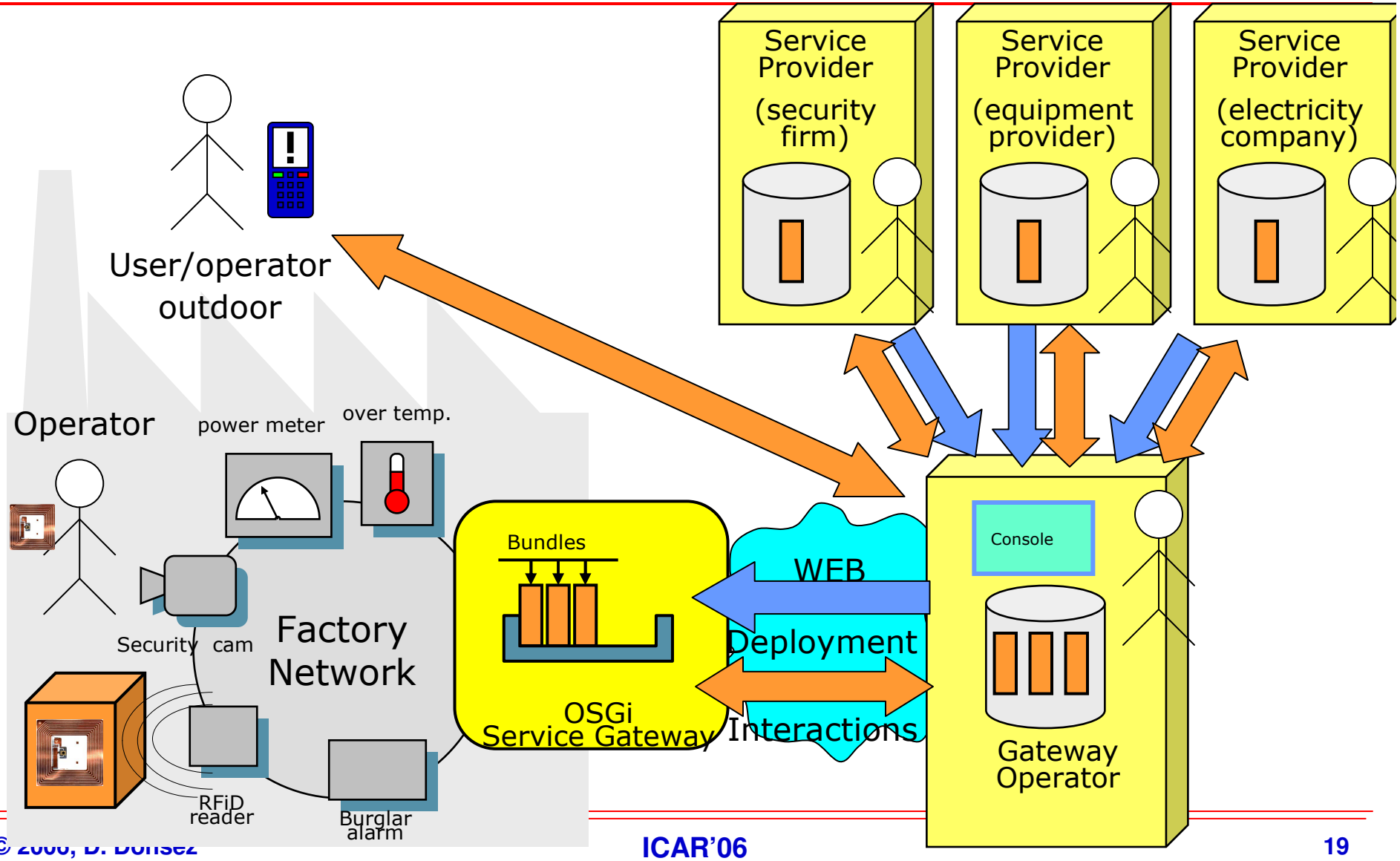
Architecture générale (i)
Déploiement



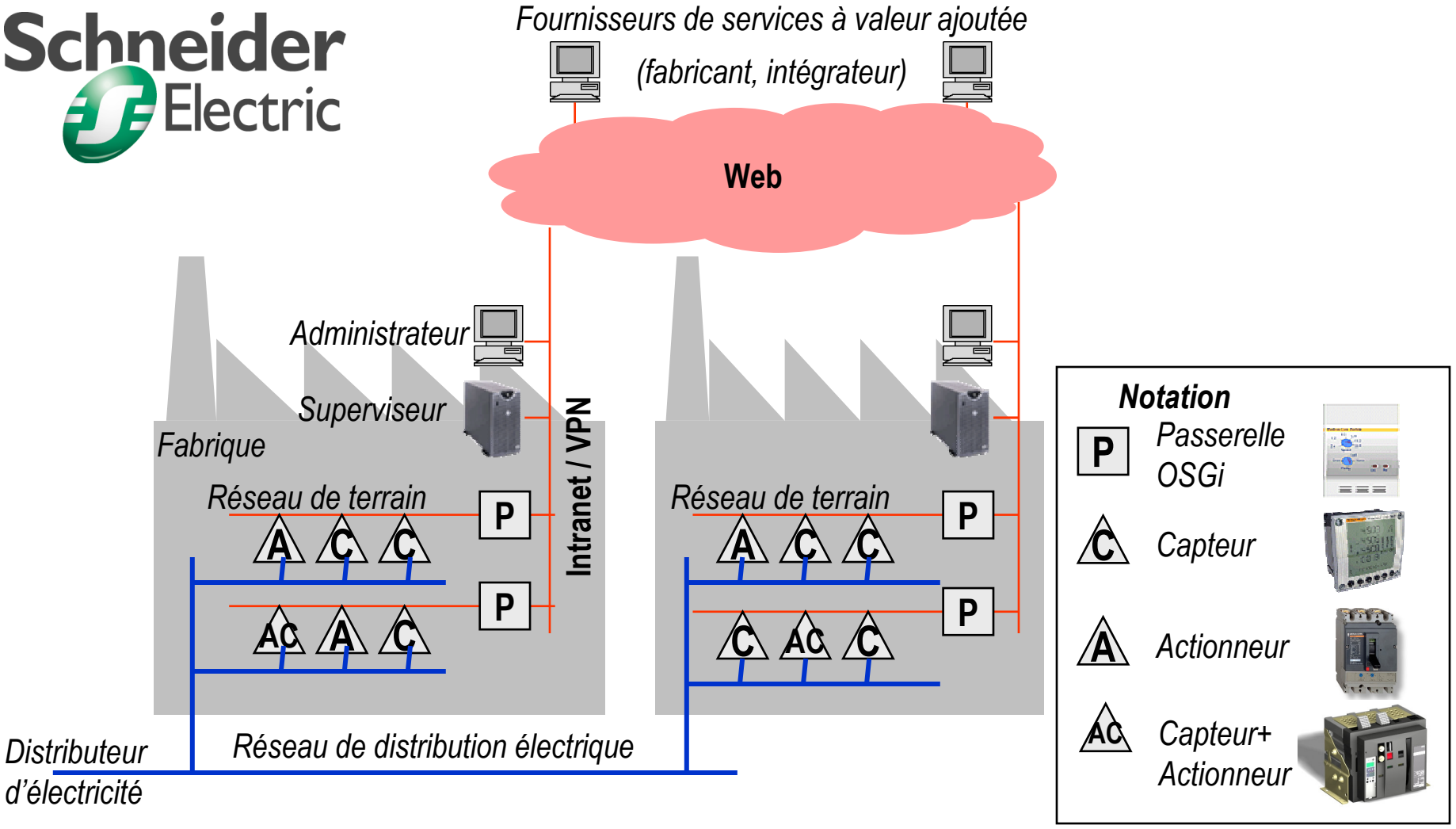
Architecture générale (ii)
Interactions



Même architecture générale (iii)
Contexte différent



Application à la Distribution Electrique chez Schneider Electric



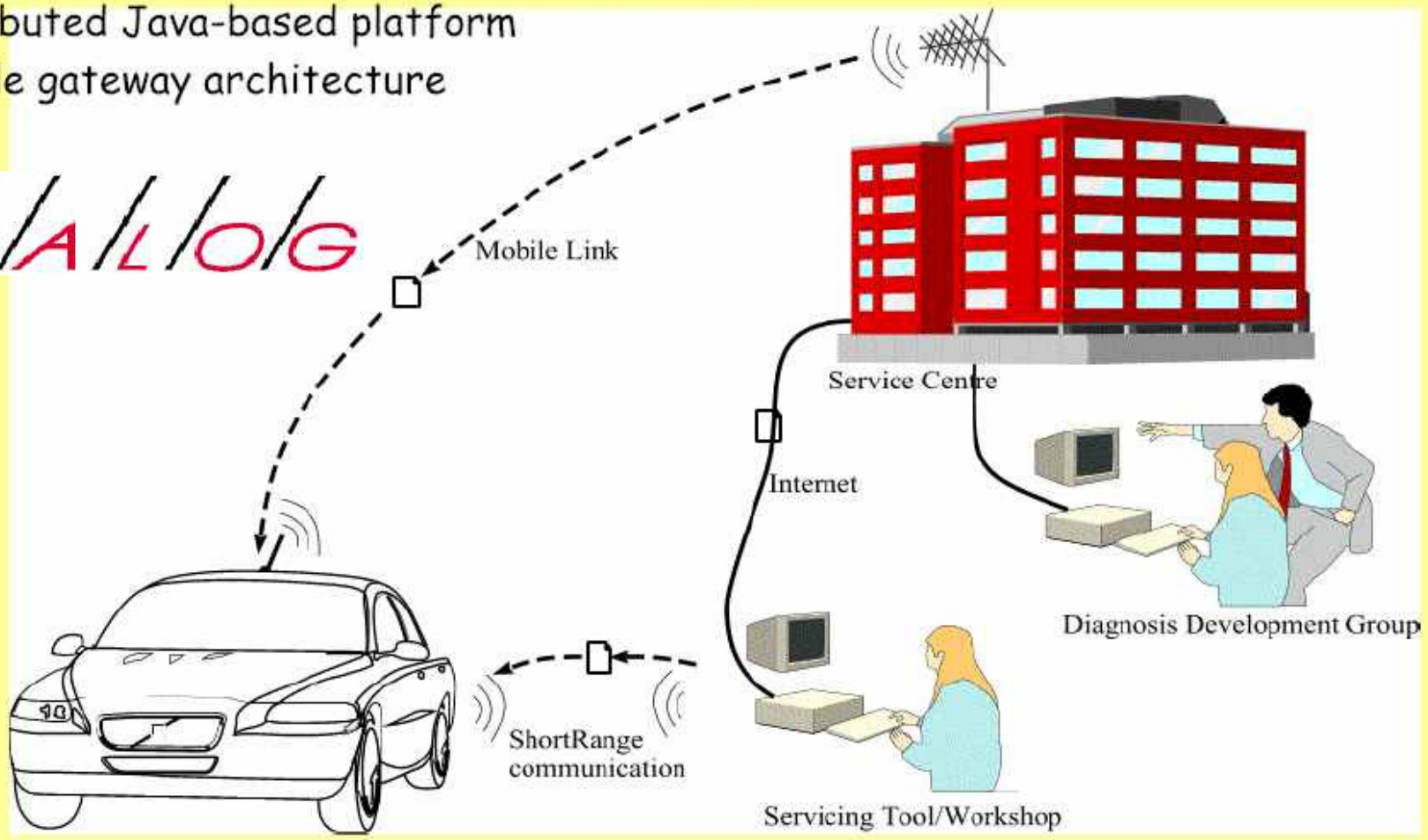
Diagnostic de véhicules à distance

Supporting both remote and local connections

Distributed Java-based platform

Vehicle gateway architecture

TRIALOG



Le M2M (Machine-to-Machine)

ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

---

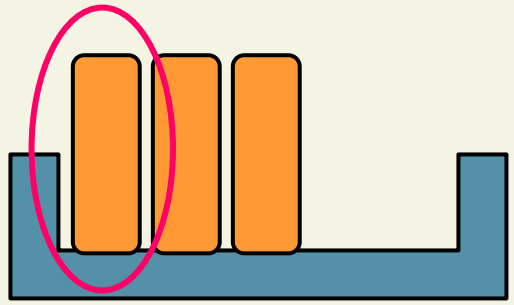
OSGi

Conditionnement, Déploiement et Service

Bundle

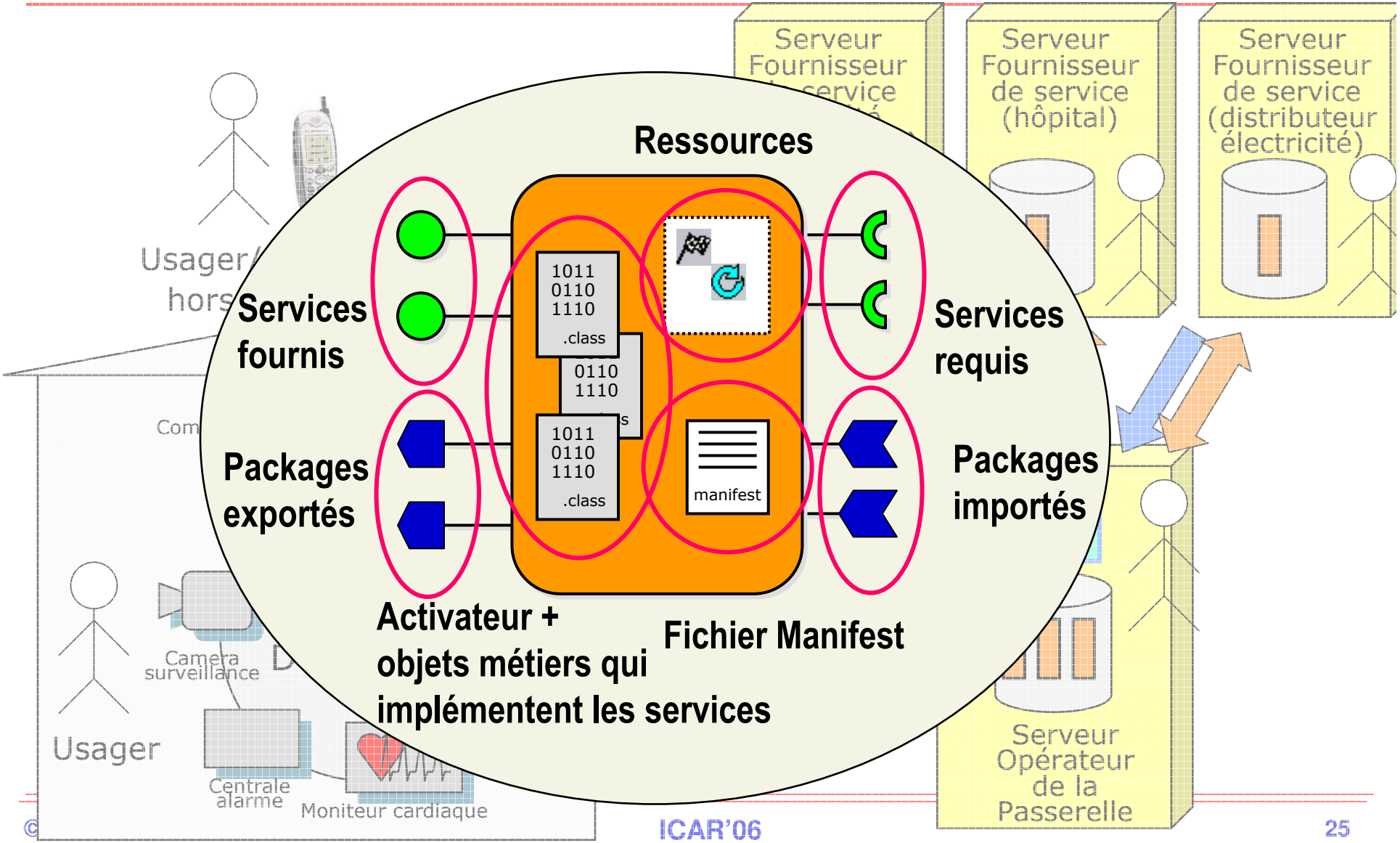
Bundle

- Unité de livraison et de déploiement sous forme d'un jar
- Unité fonctionnelle (offre des services)

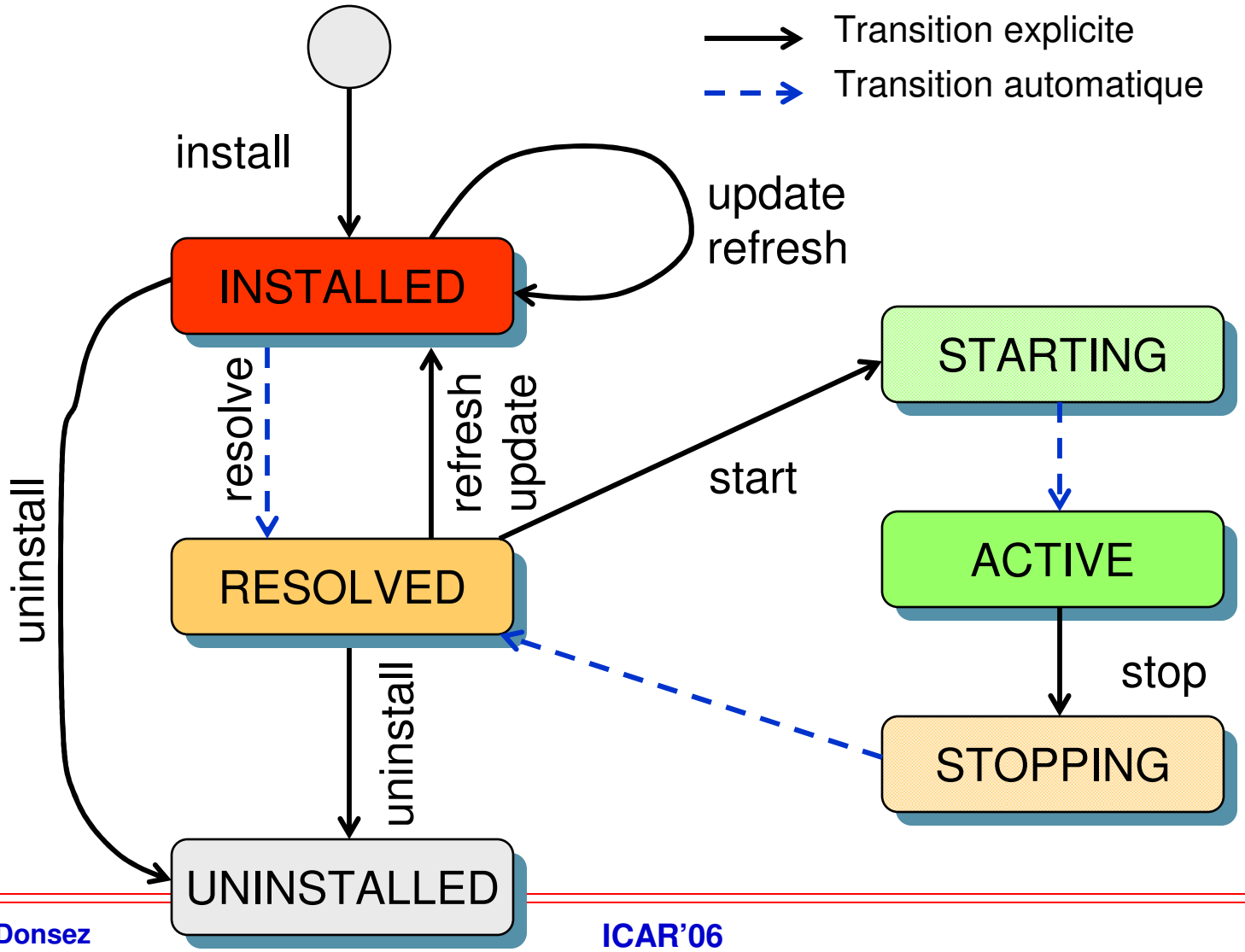


Un serveur embarqué

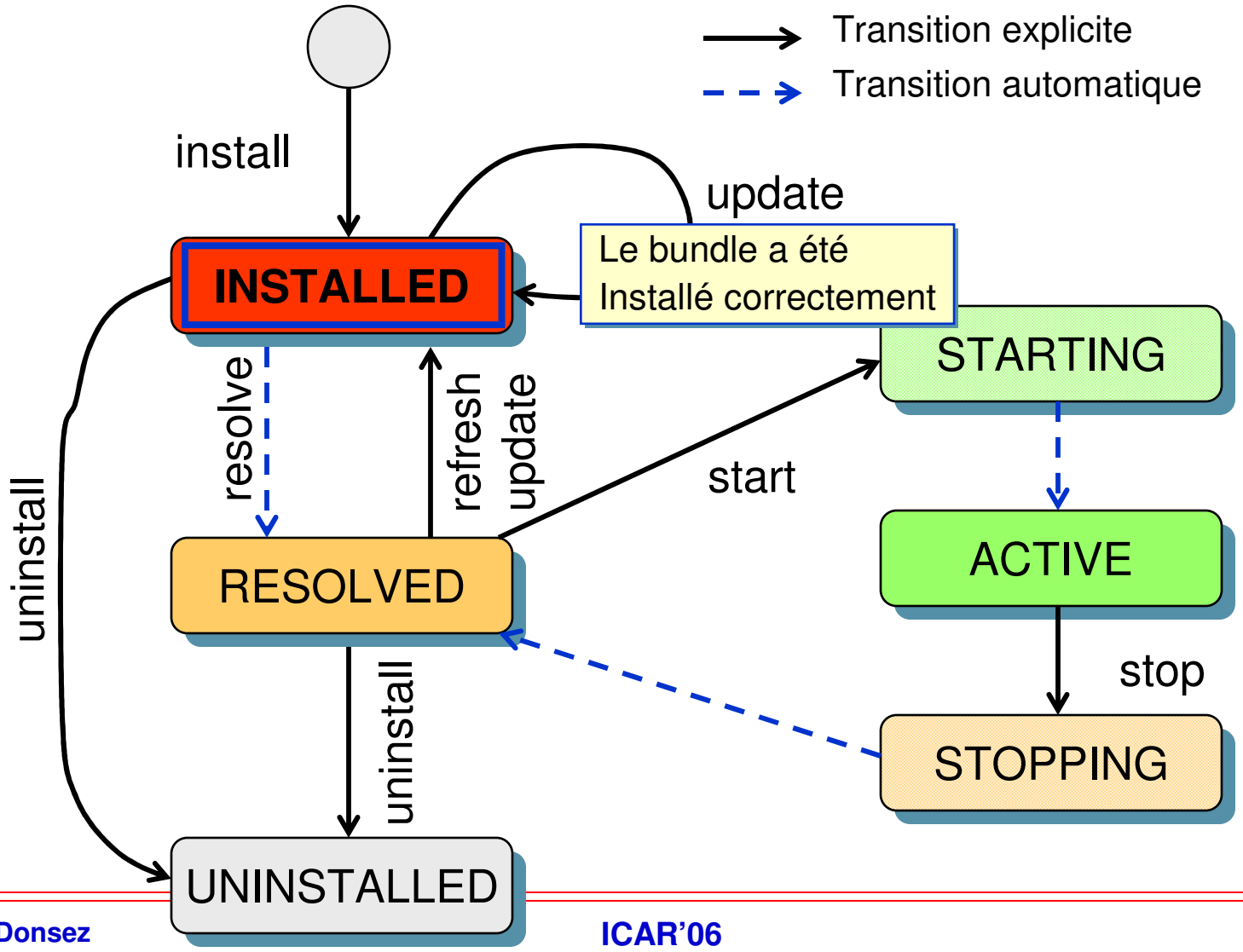
Structure d'un bundle



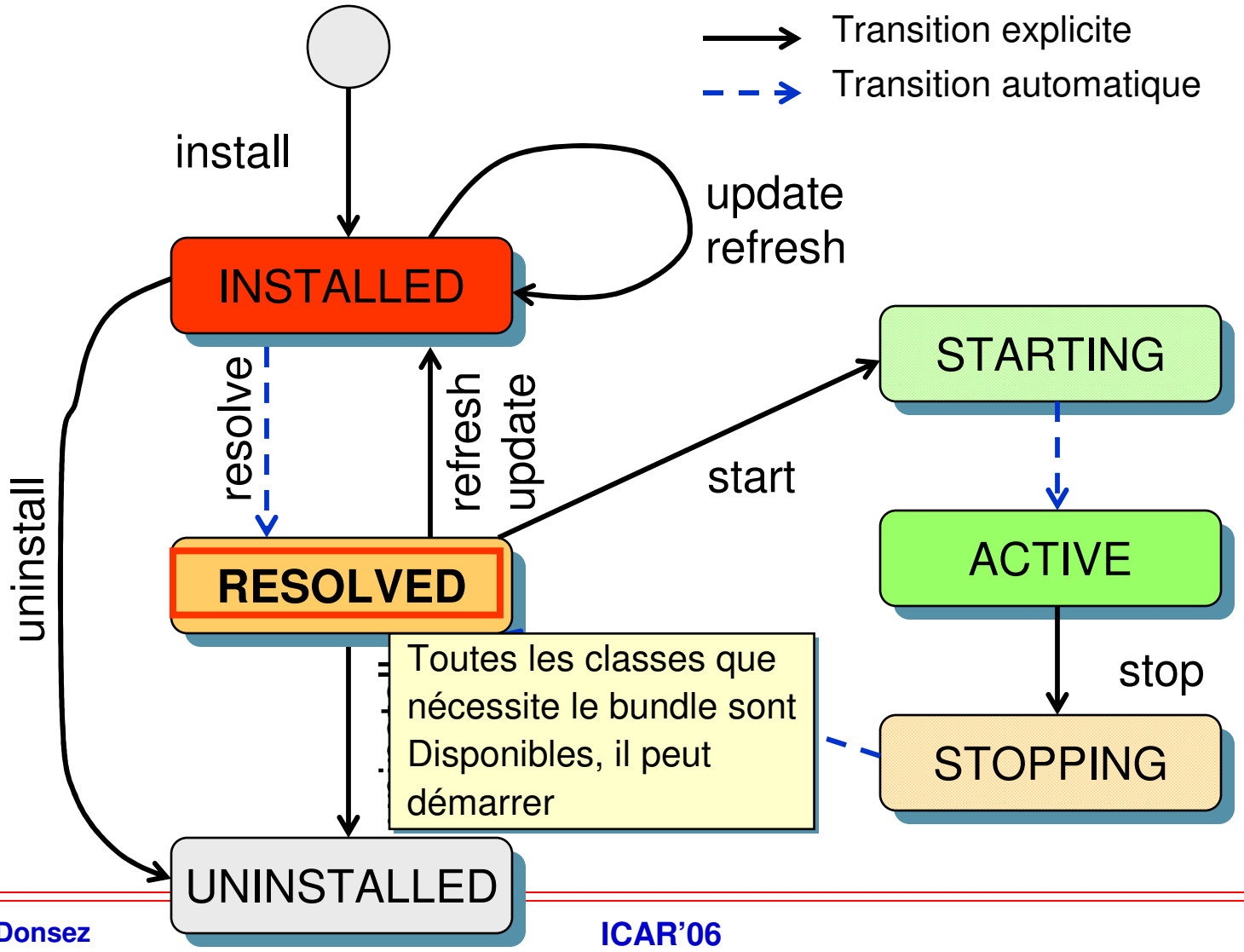
Cycle de vie d'un Bundle



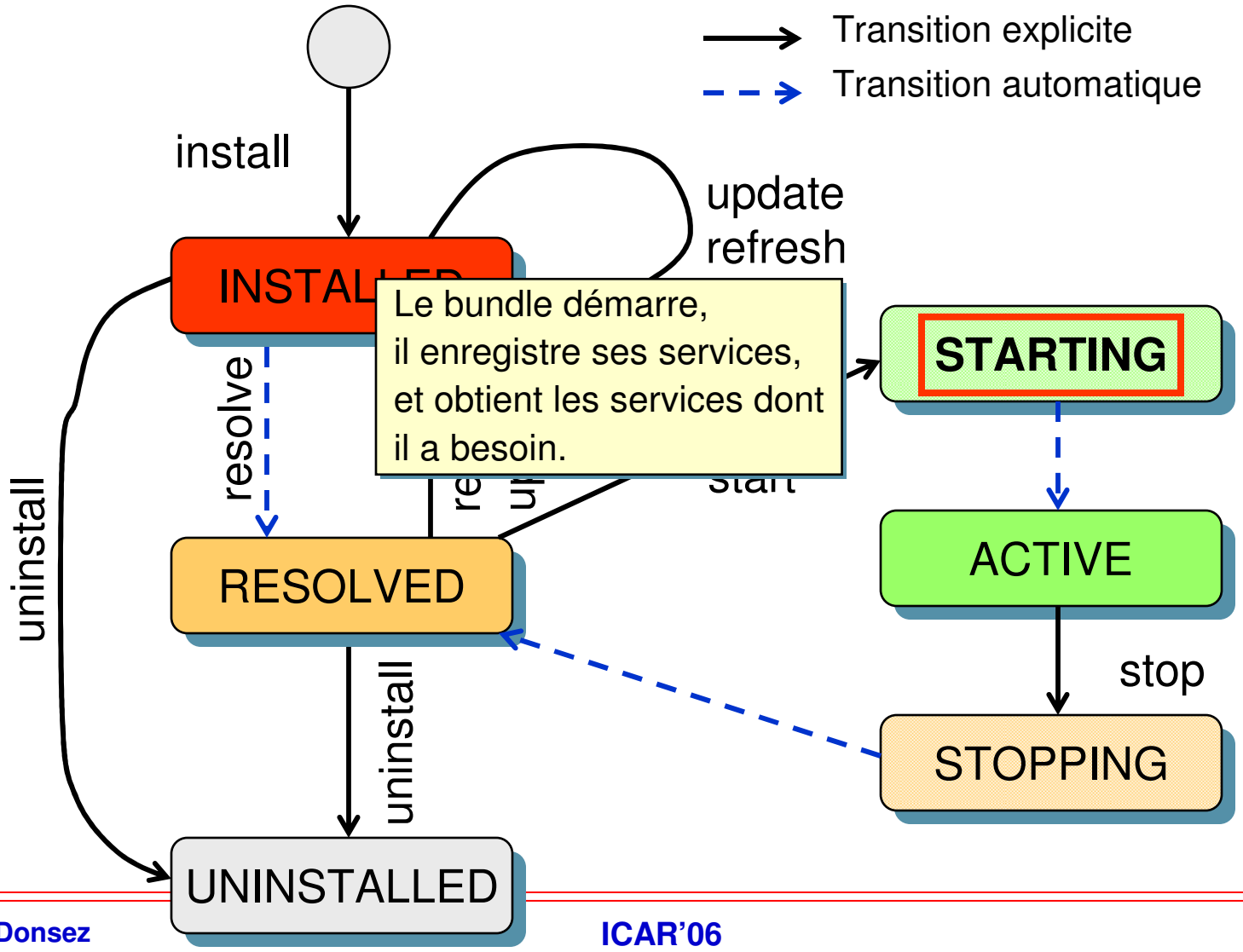
Cycle de vie d'un Bundle



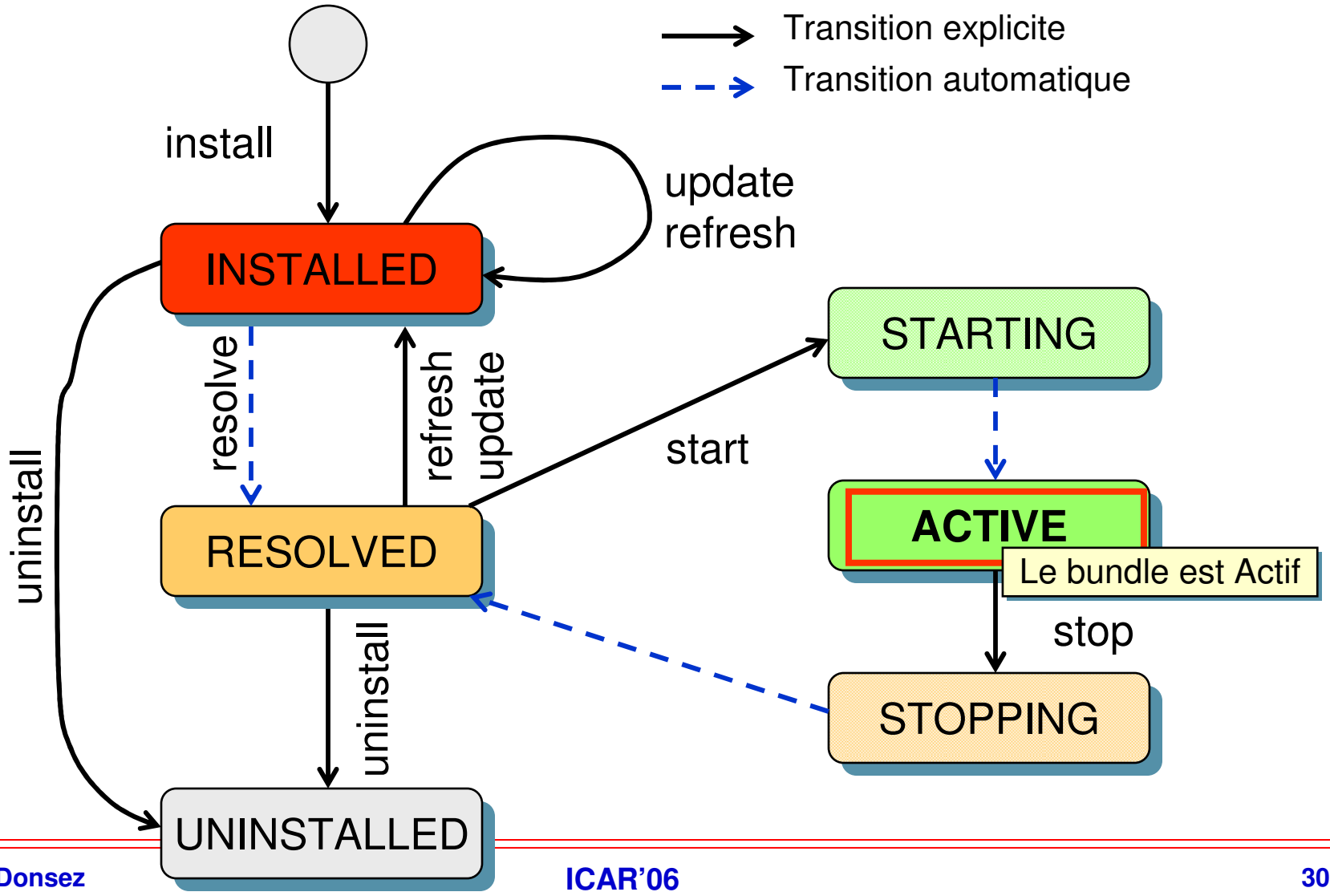
Cycle de vie d'un Bundle



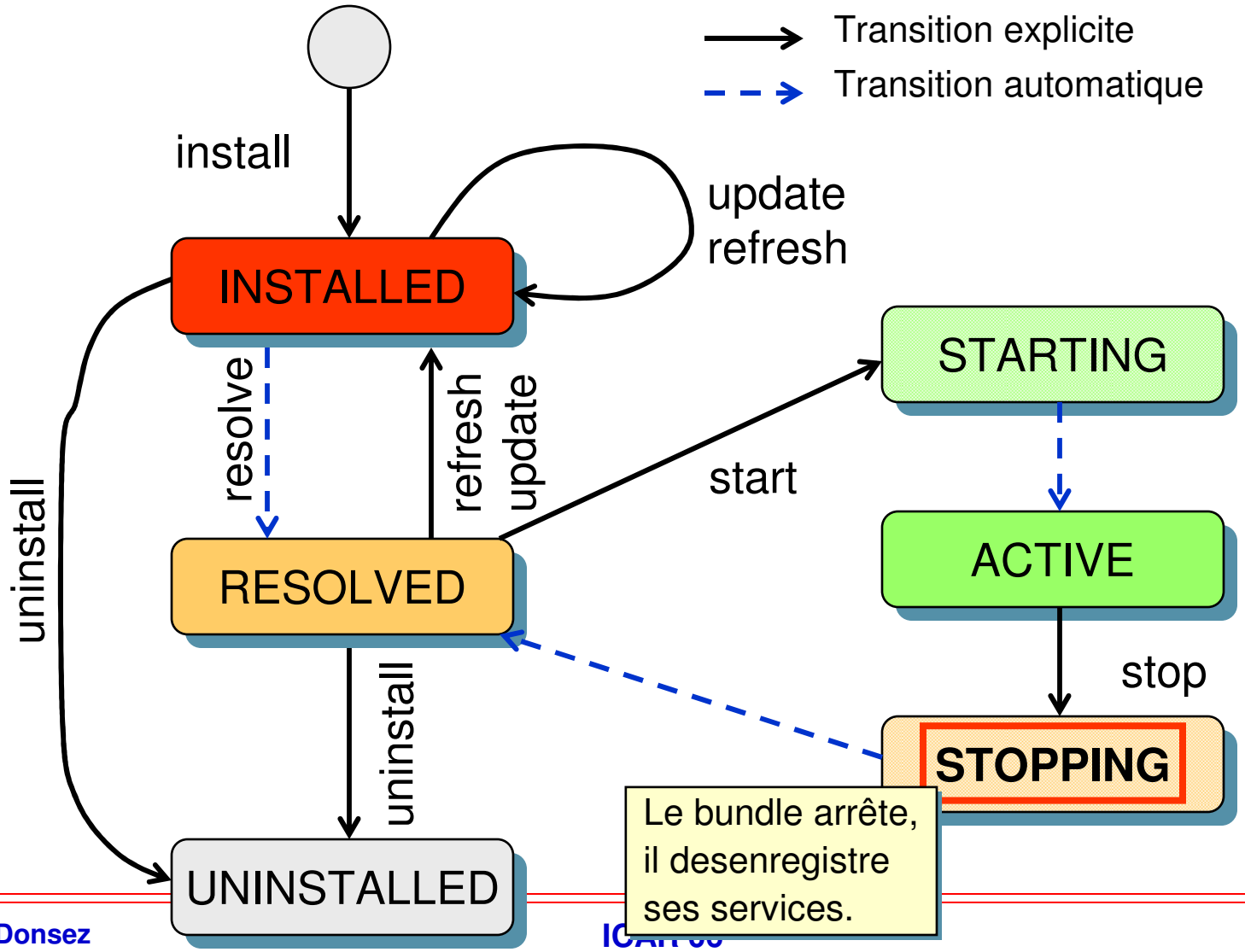
Cycle de vie d'un Bundle



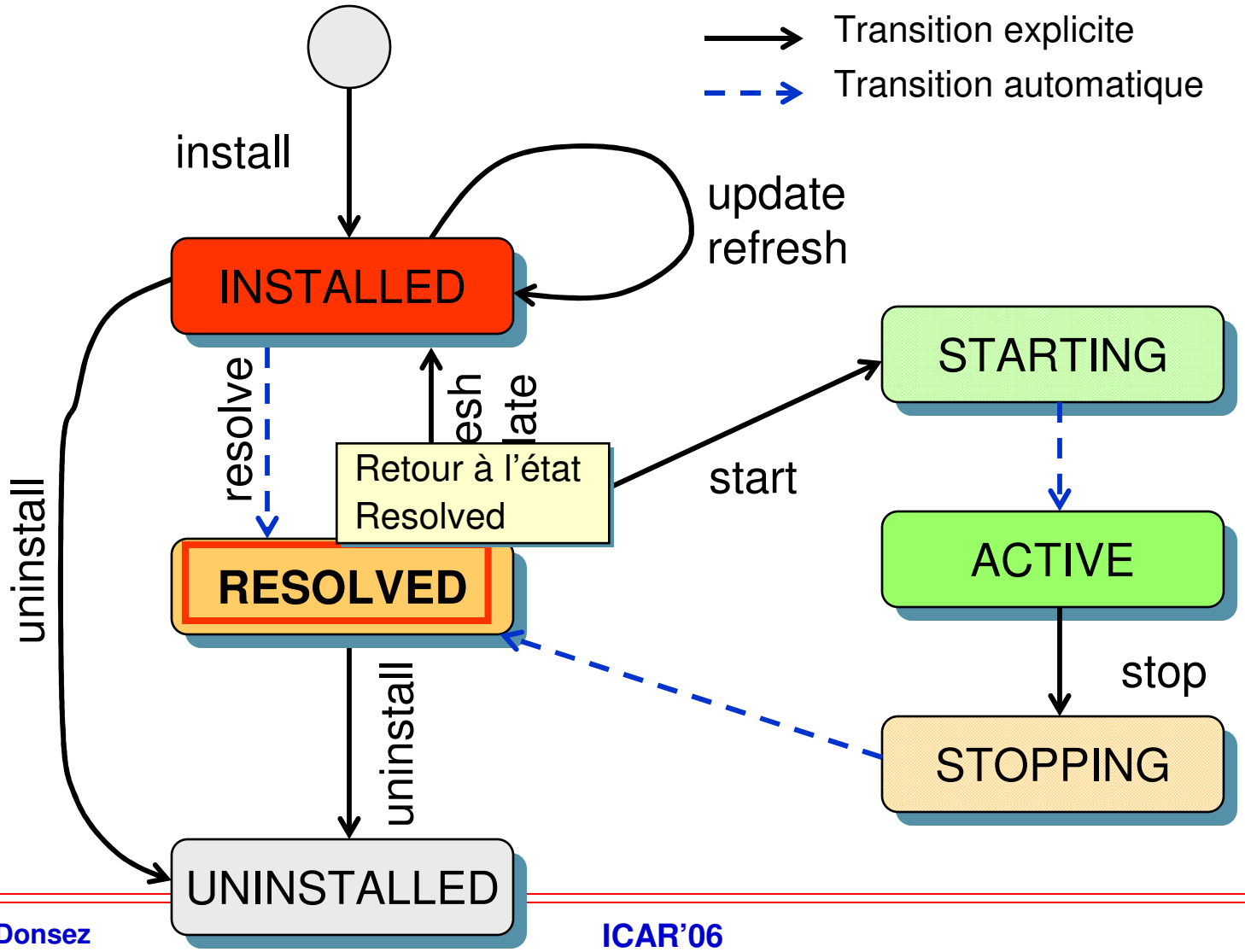
Cycle de vie d'un Bundle



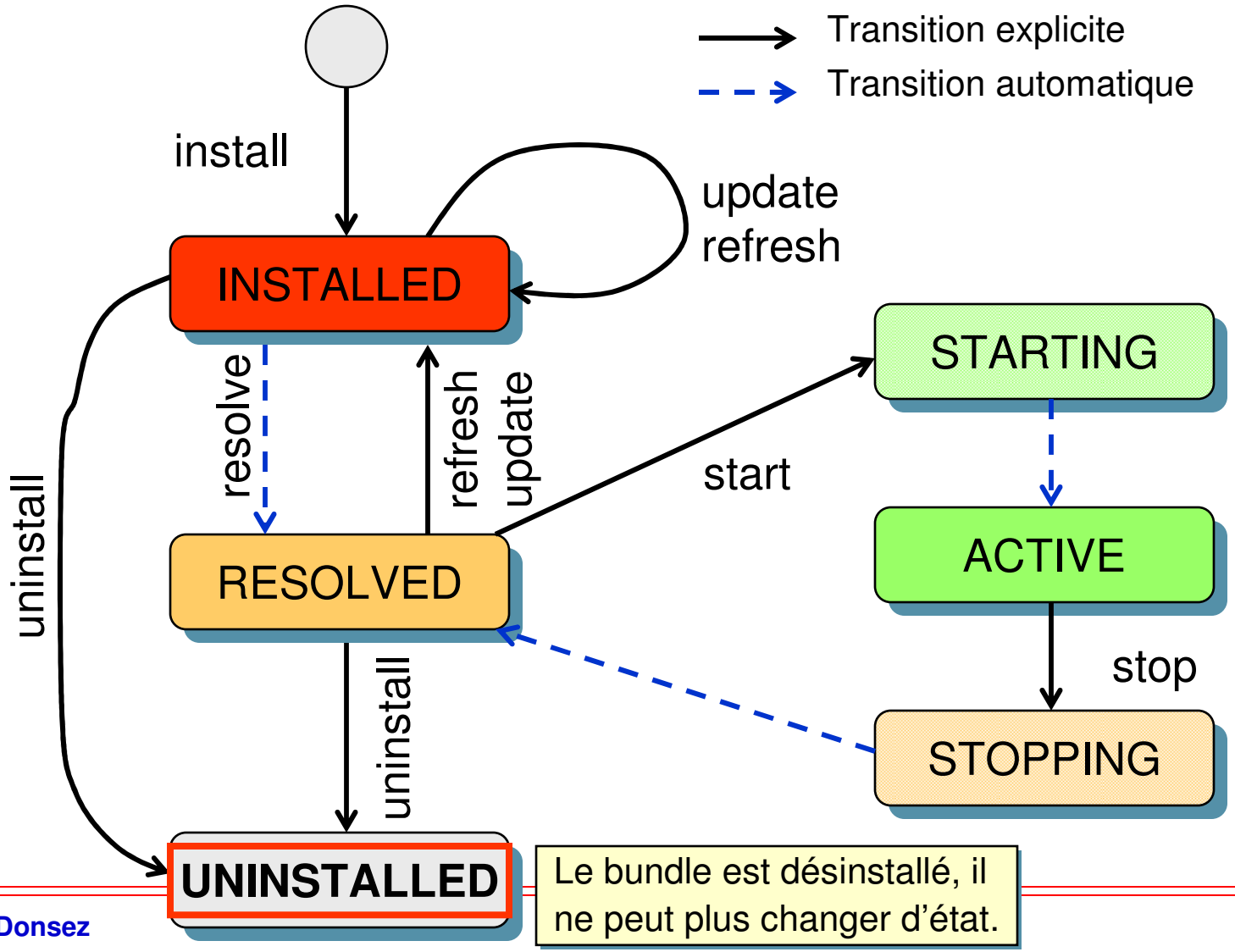
Cycle de vie d'un Bundle



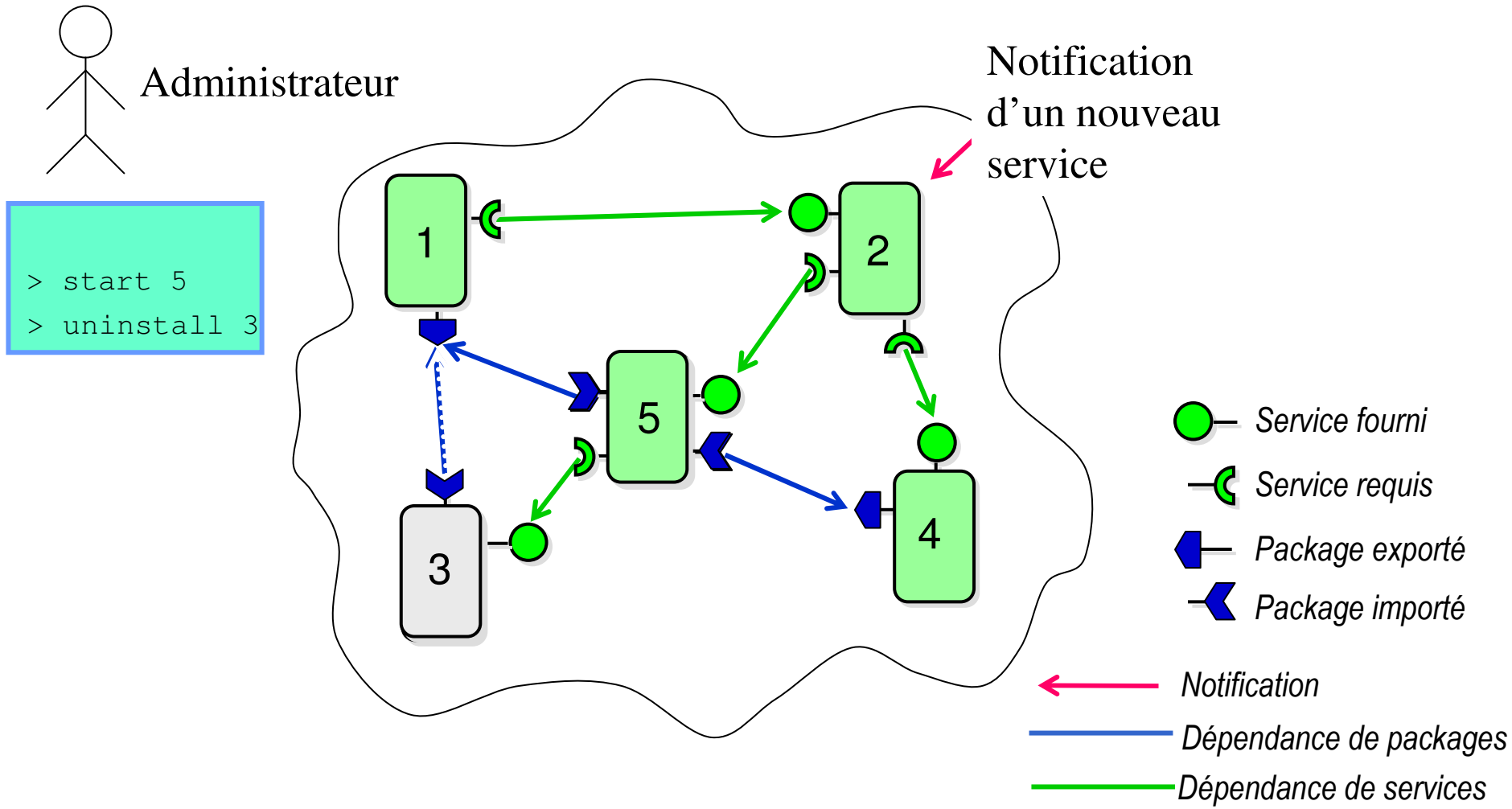
Cycle de vie d'un Bundle



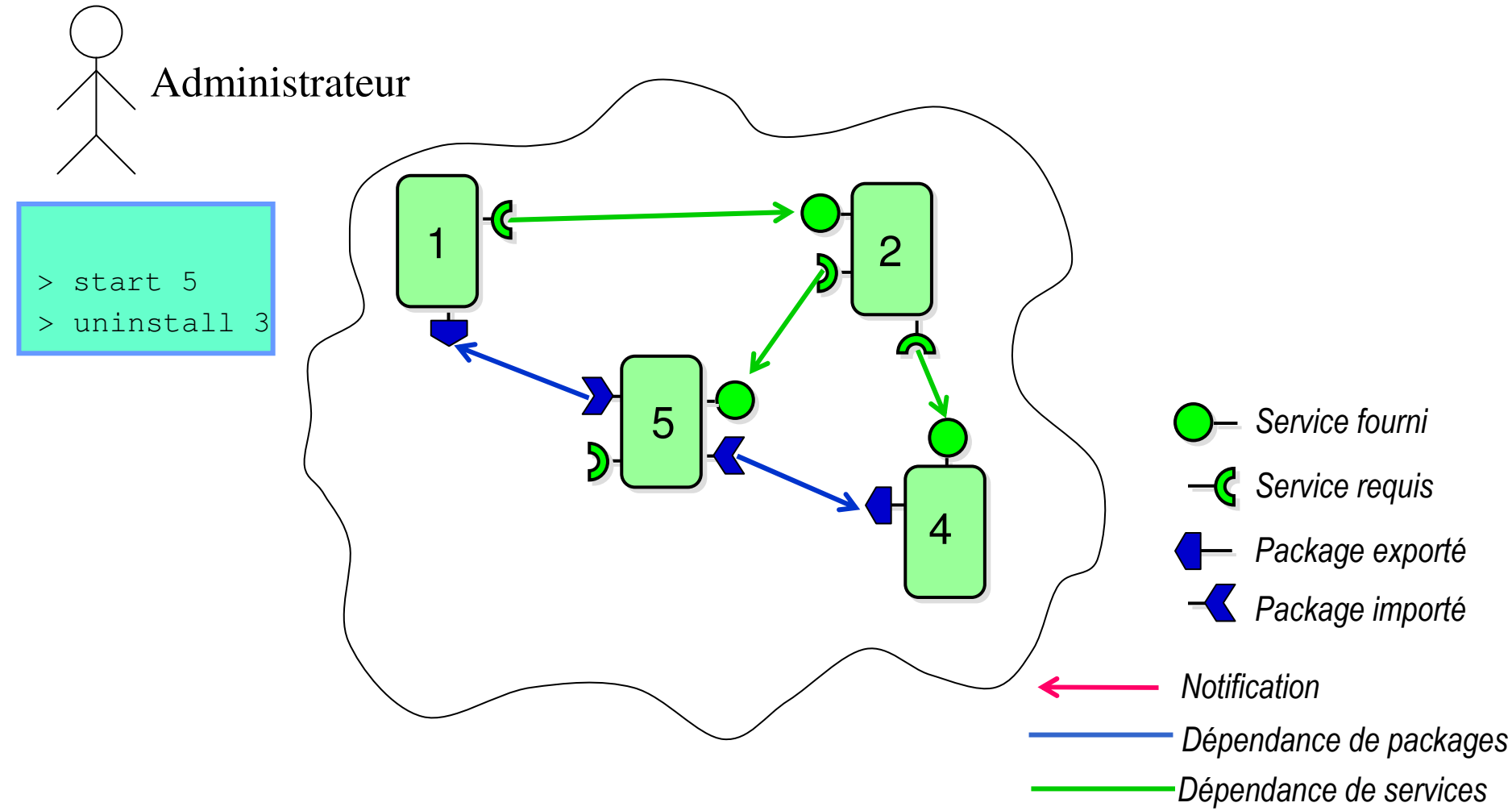
Cycle de vie d'un Bundle



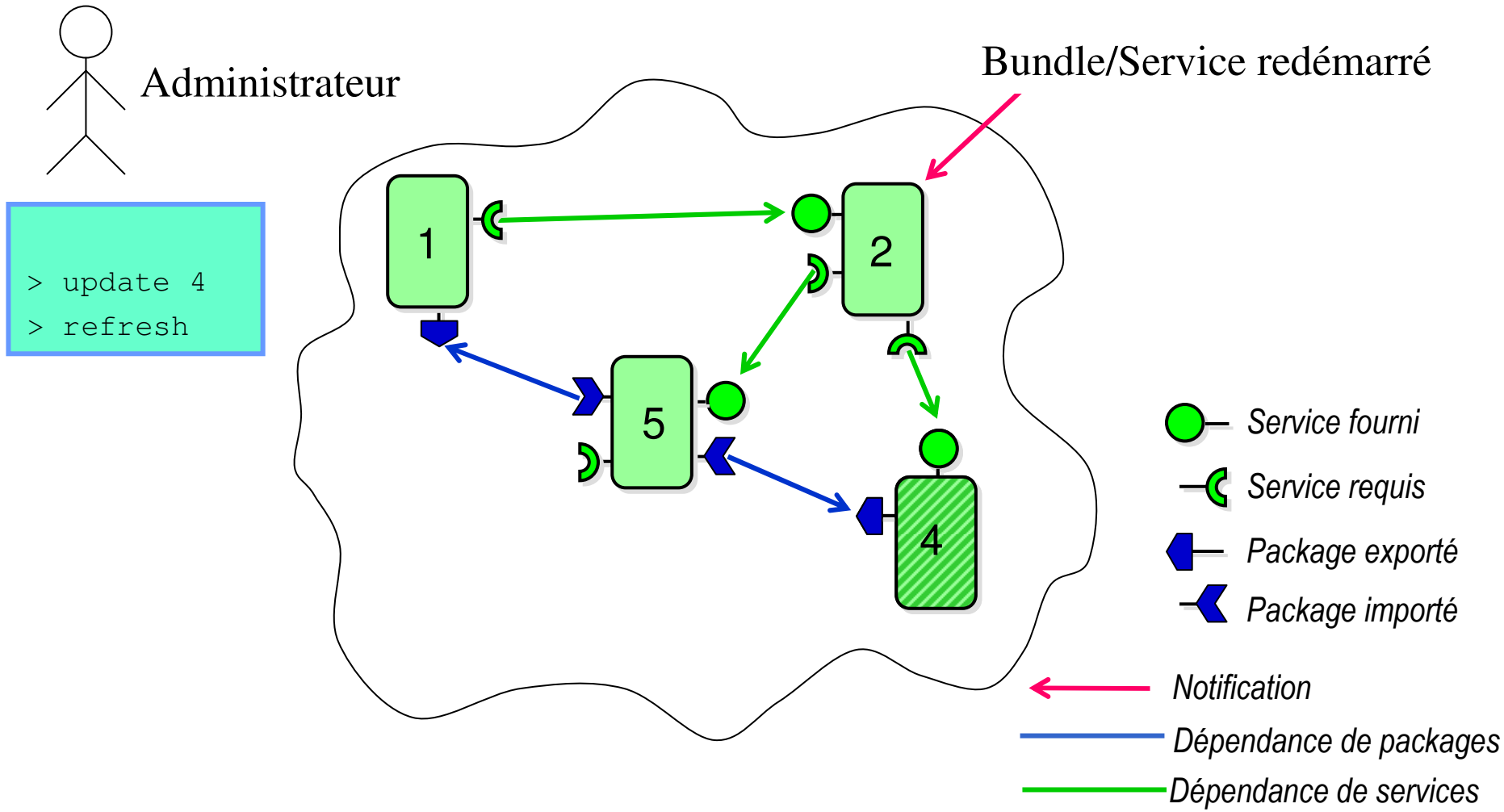
Dépendance & Dynamisme



Dépendance & Dynamisme



Dépendance & Dynamisme



Service

- Une interface (ou plusieurs)
- Des implémentations
  - multiples implémentations possibles conditionnées dans les bundles.
  - implémentation normalement non publique.
  - se trouvent dans des packages différents
- Qualifié par des propriétés.

Classe d'implémentation
`com.hp.printer.laserjet.impl.Driver`

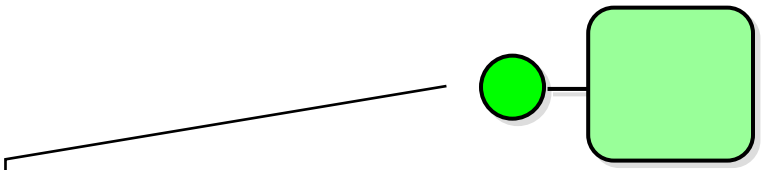
org.device.print.type=laser
location=4th floor

Classe d'implémentation
`com.lexmark.printer.laser.impl.Driver`

org.device.print.type=laser
location=1st floor

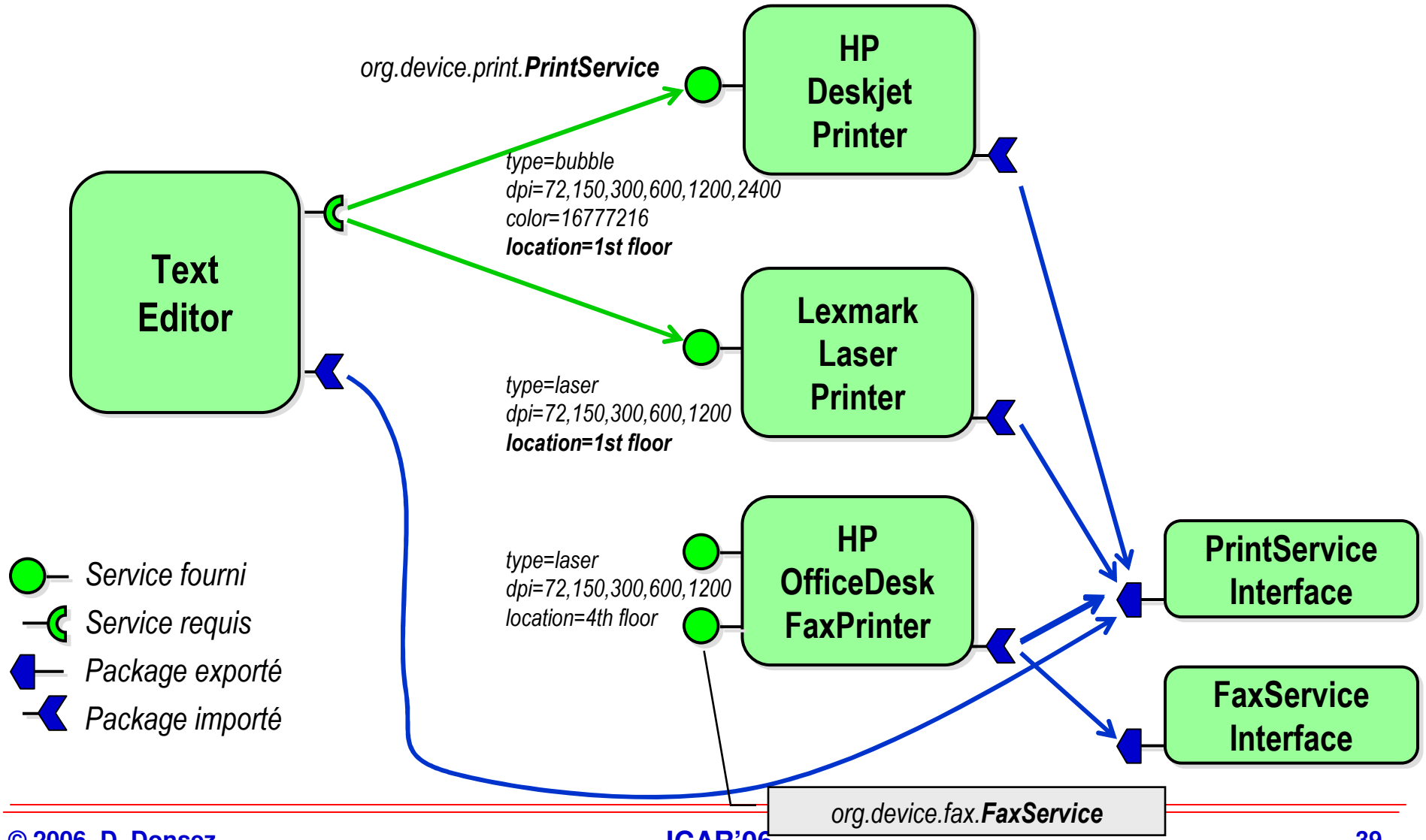
Exemple de service

Interface
org.device.print.PrintService



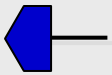
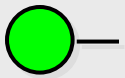
```
package org.device.print;

public interface PrintService {

  public int print(OutputStream out,
                   String[] printparams)
              throws PrintException;

  public Job[] list()
              throws PrintException;

}

public interface Job[] { … }


public class PrintException extends Exception { … }
```

Exemple d'application



Fichier manifest (i)

Informations nécessaires au framework

Import-Package 	Packages requis (avec/sans la version de spécification)
Export-Package 	Packages fournis (avec/sans la version de spécification)
Import-Service 	Services requis **(indicatif, n'est pas utilisé par le FW)** 
Export-Service 	Services fournis **(indicatif, n'est pas utilisé par le FW)** 

Bundle-Activator	Nom de la classe Activator

Bundle-ClassPath	Emplacement des classes et ressources du bundle
Bundle-NativeCode	Bibliothéques natives à charger en fonction du processeur, du SE, …

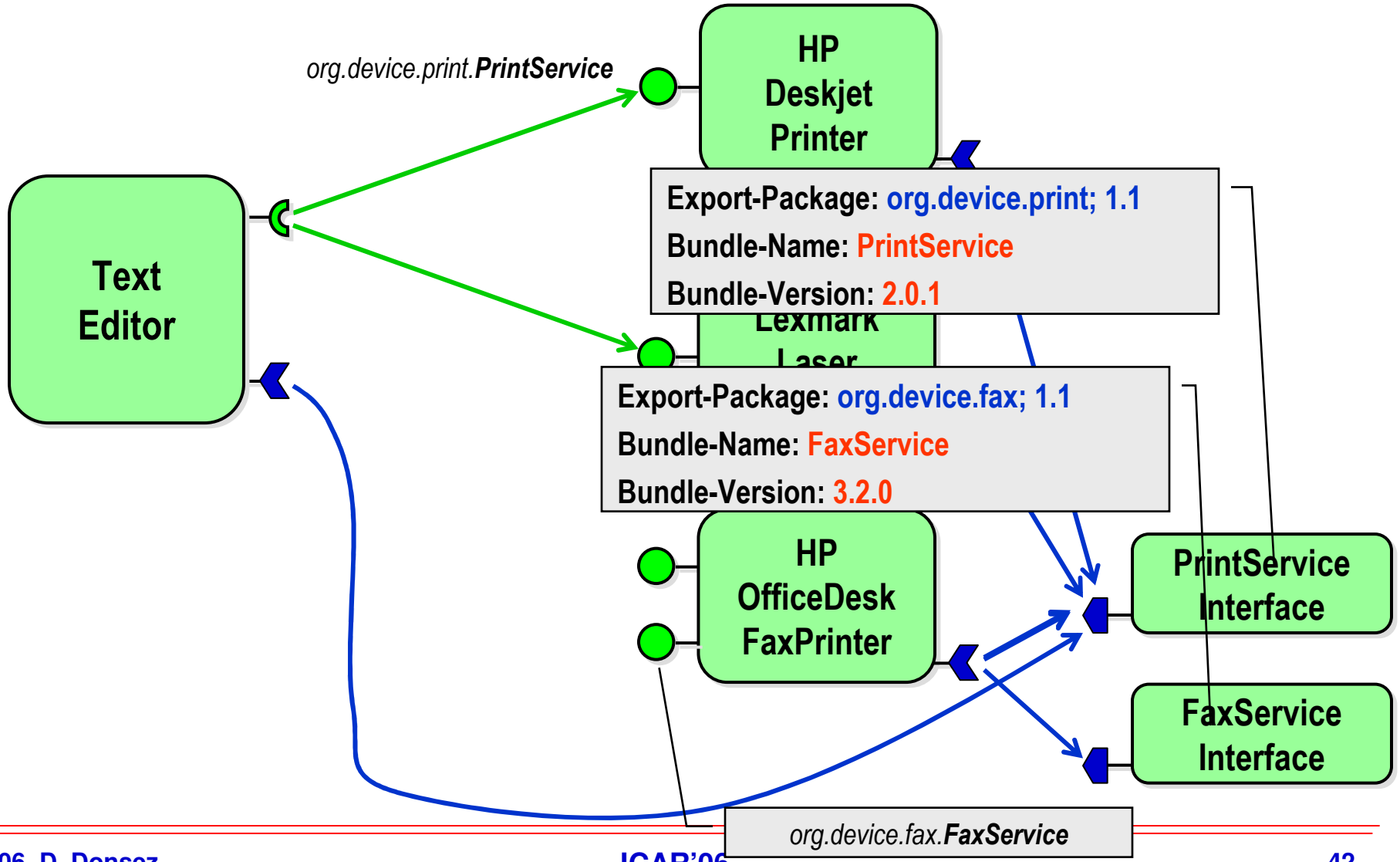
Bundle-UpdateLocation	URL des mises à jour du bundle

Fichier manifest (ii)

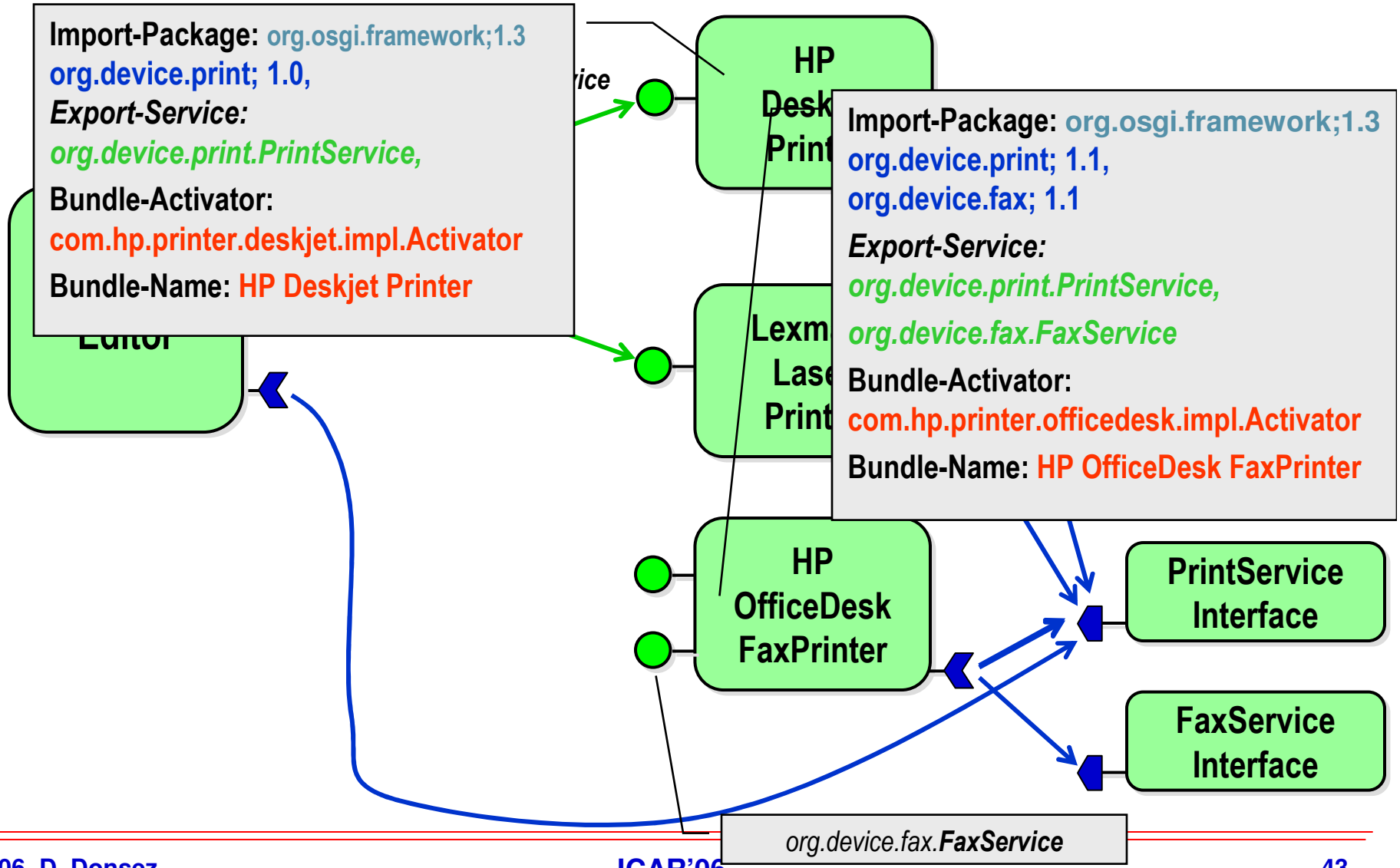
Informations nécessaires au framework

`Bundle-SymbolicName` r4	Nom symbolique du bundle (sert à l'identification)
`Bundle-Name`	Nom du bundle
`Bundle-Description`	Description du bundle
`Bundle-Version`	Version du bundle
`Bundle-DocURL`	URL de la documentation du bundle
`Bundle-ContactAddress`	Coordonnée du propriétaire du bundle
`Bundle-Category`	Catégorie du bundle
`Bundle-RequiredExecution Environment` r3	Liste d'environnement qui doivent être présents sur la plateforme (exemple : CDC-1.0/Foundation-1.0, OSGi/Minimum-1.0)
`DynamicImport-Package` r3	Liste de package qui pourront être importés en cours d'exécution (com.acme.plugin.*)

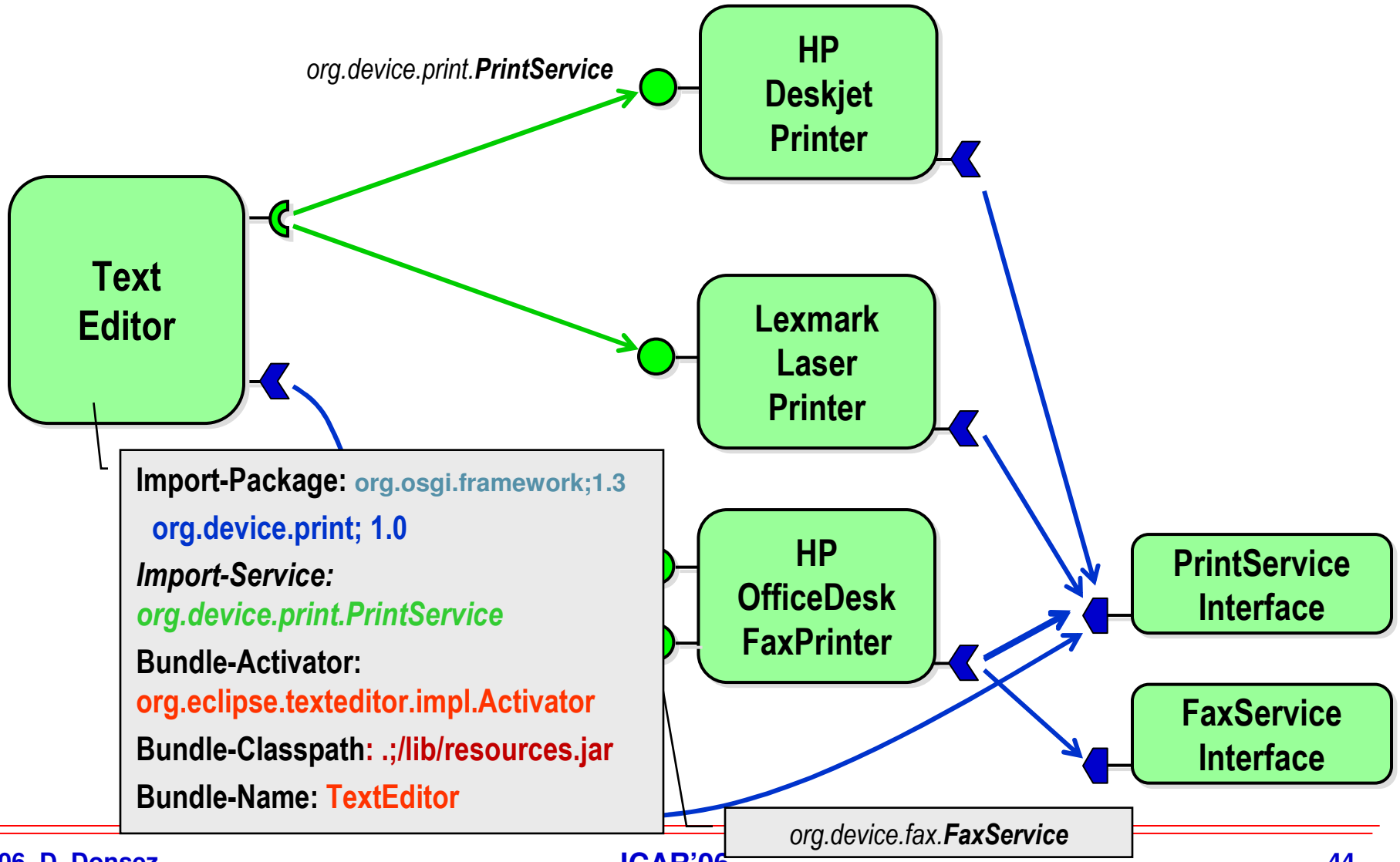
Exemple de manifest (i)



Exemple de manifest (ii)



Exemple de manifest (iii)

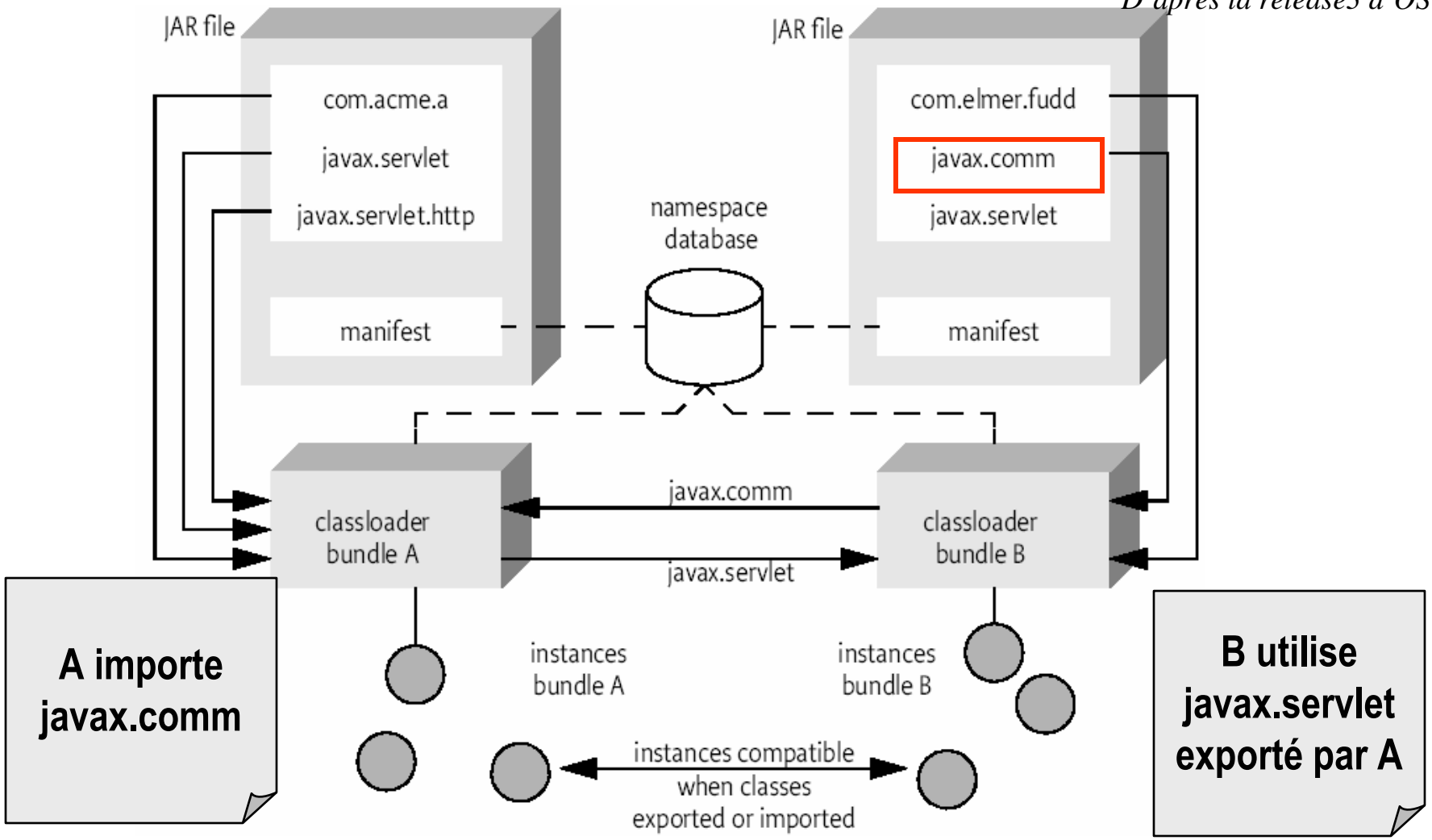


Chargement de classes (i)

- 1 ClassLoader par Bundle
  - Chargement, Mise à Jour, Déchargement
- Principe de la recherche des classes
  - La classe est dans le JRE
  - La classe est dans un package ni importé ni exporté
    - Utilisation de la classe chargée à partir du BUNDLE-CLASSPATH
  - La classe est dans un package importé
    - Utilisation de la classe chargée par le CL d'un autre bundle
  - La classe est dans un package exporté mais déjà exporté par un autre bundle
    - Utilisation de la classe chargée par le CL de l'autre bundle
  - La classe est dans un package exporté mais non exporté par un autre
    - Utilisation de la classe chargée à partir du BUNDLE-CLASSPATH

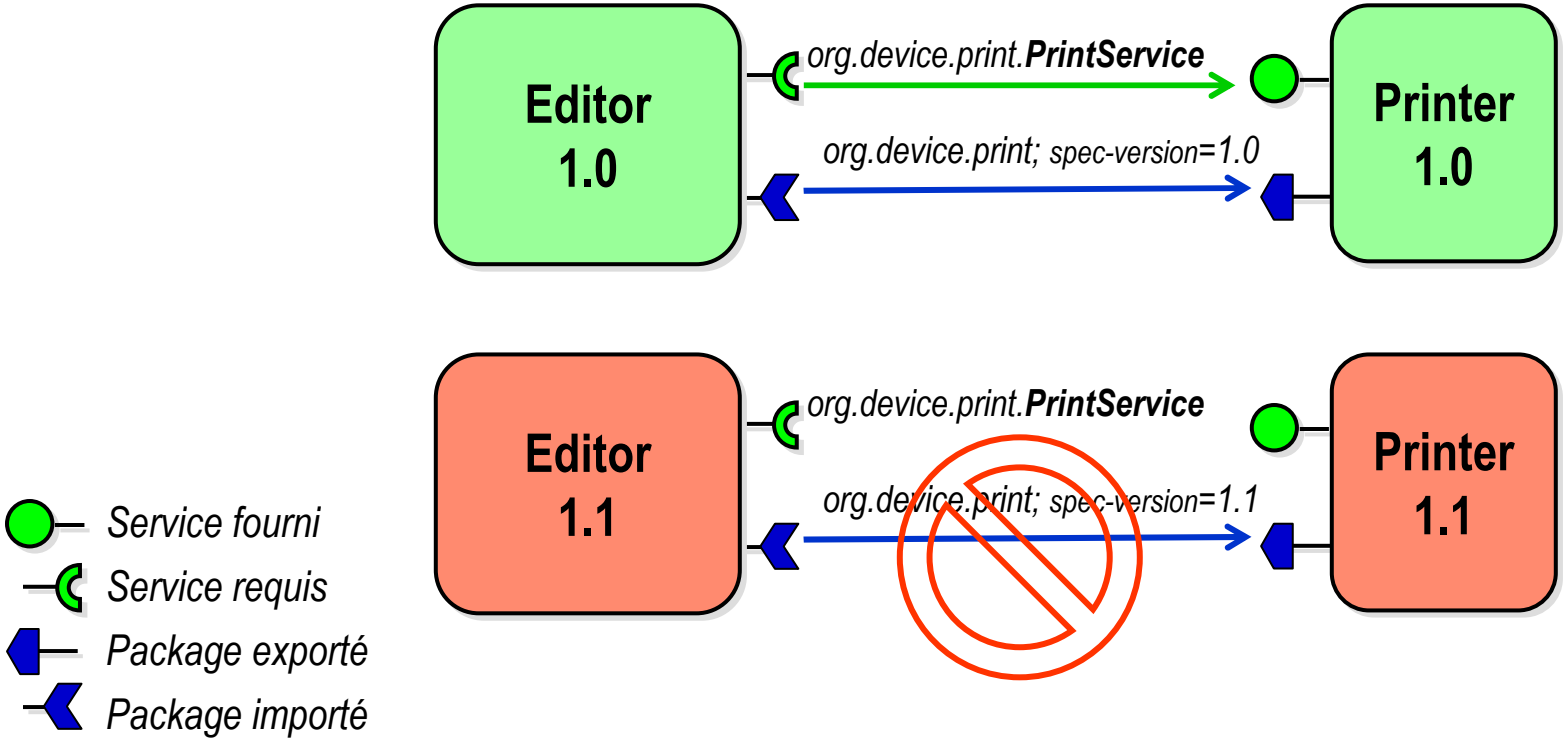
Chargement de classes (ii)

D'après la release3 d'OSGi



Les limites r3

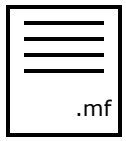
- Pas d'activation tant que tous les imports ne sont pas résolus
- Un service package actif à la fois
- Compatibilité ascendance à assurer *Ad vitam eternam*



Les avancés

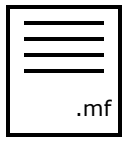
- R3
  - Importation dynamique
- R4
  - Bundle fragment
  - Bundle requis
  - Bundle extension
  - Intervalle de version, Politiques sur les versions
  - Importation et Exportation conditionnelles (attribut et filtre)
  - Activation simultanée de plusieurs version de packages
- La suite : le JSR 277, JSR 294 …

- Richard S. Hall, “Java modularity, OSGi, and JSRs 277, 291, and 294”, ApacheCon EU 2006
- http://docs.safehaus.org/download/attachments/2995/osgi-apachecon-20060628.pdf



Bundle-Classpath

- **Représente (dans le manifeste) les chemins (dans le JAR) de recherche des classes et des ressources**
- **3 cas**
  - `Bundle-Classpath: .` **ou Pas de** `Bundle-Classpath`
    - **Recherche dans le JAR**
  - `Bundle-Classpath: .;demo/nested.jar;test/nest.jar`
    - **Recherche dans le JAR puis dans le JAR inclus**
  - `Bundle-Classpath: demo/nested.jar`
    - **Recherche dans le JAR inclus**
    - **Aucune classe ou ressource n'est recherchée dans le JAR**
- **Intérêt des JAR inclus**
  - **Conservation des signatures, manifestes, …**
  - **Possibilité de *patcher* un sous ensemble des ressources/classes !**



Bibliothèques natives

- **Bibliothèques de fonctions natives (C) dépendantes du processeur et de l'OS**
  - **Exemple : Pilotes matériel (javax.comm), Patrimonial (codec), …**
- **Bundle-NativeCode dans le MANIFEST**
  - **Spécifie l'emplacement des bibliothèques dépendantes du système et du processeur, à charger dynamiquement (par le ClassLoader)**
  - **Exemple**
    - **Bundle-NativeCode:** com/mycomp/impl/nativesample/libnat.so;
      osname=Solaris; processor=sparc; osversion=5.5,
      com/mycomp/impl/nativesample/libnat.so;
      osname=SunOS; processor=sparc; osversion=2.5,
      com/mycomp/impl/nativesample/nat.dll;
      osname=Windows NT; processor=x86; osversion=4.0
- **Remarque : Propriétés du framework**
  - org.osgi.framework.processor, org.osgi.framework.language, org.osgi.framework.os.name, org.osgi.framework.os.version

La classe d'activation du bundle

- **Classe publique**
  - Implémente les 2 méthodes start() et stop() de BundleActivator
  - qui reçoivent une référence sur un contexte.
- ***start(BundleContext ctxt)***
  - recherche et obtient des services requis auprès du contexte et/ou positionne des listeners sur des événements
  - enregistre les services fournis auprès du contexte
- ***stop(BundleContext ctxt)***
  - désenregistre les services fournis
  - relâche les services requis
    - Cependant le FW fait ces opérations si stop() en oublie !

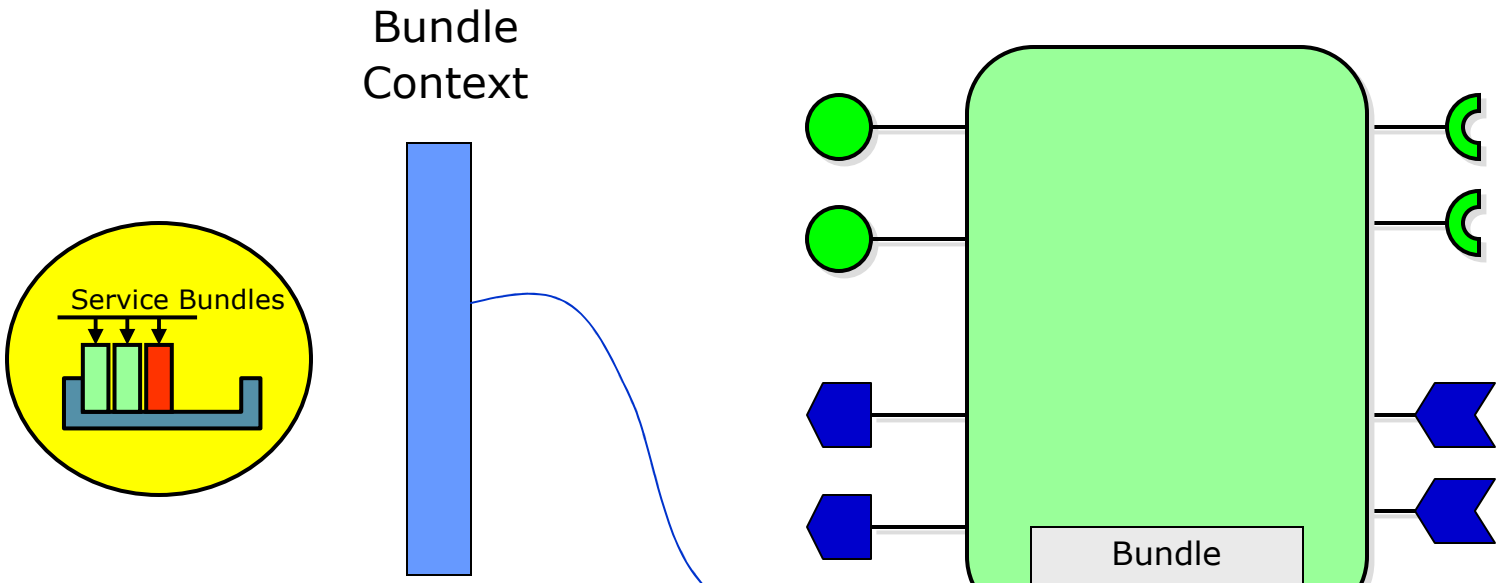
il peut ne pas y avoir de BundleActivator dans un bundle

- Livraison de classes et ressources

BundleContext

- **Interface vers le framework**
  - Passé lors des invocations de start() et stop() de l'Activator
- **Permet**
  - L'enregistrement de services
  - Le courtage de services
  - L'obtention et la libération des services
  - La souscription aux évènements du Framework.
  - L'accès aux ressources du bundle
  - *L'accès aux propriétés du framework*
  - *L'installation de nouveaux bundles*
  - *L'accès à la liste des bundles*

BundleContext et Activator



- registerService()
- getServiceReferences()
- getService()
- getDataFile()
- addServiceListener()
- addBundleListener()
- addFrameworkListener()

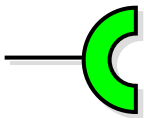
- start(BundleContext bc)
- stop(BundleContext bc)
- serviceChanged()

Enregistrement de services (Lexmark Laser Printer)

```
package com.lexmark.printer.laser.impl;
public class Activator implements BundleActivator {
        private ServiceRegistration reg=null;
        private PrintService theService=null;
        public void start(BundleContext ctxt) throws BundleException {
                theService=new PrintServiceImpl();
                Properties props=new Properties();
                props.put("type", "laser");
                props.put("dpi", "72,150,300,600,1200");
                props.put("location", "1st floor");
                reg=ctxt.registerService(
                        "org.device.print.PrintService", theService, props);
        }
        public void stop(BundleContext ctxt) throws BundleException {
                if(reg != null) reg.unregister();
        }
}
```

Recherche de services (TextEditor)

```
package org.eclispe.texteditor.impl
import org.device.print.PrintService;
class Activator implements BundleActivator {
   public void start(BundleContext ctxt) throws BundleException {
            private PrintService ser;
            // On va voir si quelqu'un offre un PrintService ...
            ServiceReference[] tempRefs
                        =ctxt.getServiceReferences
                        ("org.device.print.PrintService","(location=1st floor)");
            if(tempRefs!=null) {
               System.out.println("Found a PrintService! I will use it!!!");
               // On prend le premier offert!
               ser=(PrintService) ctxt.getService(tempRefs[0]);
            }
            …
   }
   …
}
```



Recherche (Courtage) de services

- **Filtrage par des expressions de condition LDAP (RFC1960) sur les propriétés enregistrées par les services**
- **Expressions de filtrage**
  - Expressions simples (attribut opérateur valeur)
  - Valeurs de type `String, Numerique, Character, Boolean, Vector, Array`
  - Attribut insensible aux majuscules/minuscules
  - L'attribut `objectClass` représente le nom du service
  - Opérateurs `>=, <=, =, ~=` (approximativement égal), `=*` (présent)
  - Connecteurs logiques `&, |, !`

Recherche de services

- **Tous les services d'impression**

```
refs=bundleContext.getServiceReferences("org.device.print.PrintService", null);
refs=bundleContext.getServiceReferences(null,
                "(objectClass=org.device.print.PrintService)");
```

- **Certains services d'impression**

```
refs=bundleContext.getServiceReferences("org.device.print.PrintService",
                "(&(!(type=laser))(capability=double-sided)(!(dpi<=300))(location=*)" );
```

- **Tous les services de org.device**

```
refs=bundleContext.getServiceReferences(null,"(objectClass=org.device.*)");
```

- **Le service d'impression et de fax au 3ième étage**

```
refs=bundleContext.getServiceReferences(null,
    "(&(objectClass=org.device.print.PrintService)(objectClass=org.device.fax.FaxService)"
    + "(location=4th floor))" );
```

Événements dans le Framework

- **FrameworkEvent**
  - Notifie le démarrage et les erreurs du Framework
  - interface `FrameworkListener` méthode `frameworkEvent`

    Traitement séquentiel et asynchrone des listeners (par event dispatcher)
- **BundleEvent**
  - Notifie les changements dans le cycle de vie des bundles
  - interface `BundleListener` méthode `bundleChanged`

    Traitement séquentiel et asynchrone des listeners (par event dispatcher)
  - interface `SynchronousBundleListener` méthode `bundleChanged`

    Traitement séquentiel et synchrone des listeners (avant le traitement du changement d'état)

    v2
- **ServiceEvent**
  - Notifie l'enregistrement ou le retrait de services
  - interface `ServiceListener` méthode `serviceChanged`

    Traitement séquentiel et synchrone des listeners

Prendre en compte l'enregistrement et le retrait de service (i)

- Les bundles « requesters » doivent impérativement prendre en compte l'enregistrement et le retrait de services « importés »
- Exemple

```
public class PrintListenerActivator implements BundleActivator {
    PrintServiceListener listener = null;
    public void start(BundleContext context) {
            PrintServiceListener listener = new PrintServiceListener(context);
            context.addServiceListener(listener);
    }
    public void stop(BundleContext context) {
            context.removeServiceListener(listener);
    }
}
```

Prendre en compte l'enregistrement et le retrait de service (ii)

- **Exemple simpliste et inutile**

```
class PrintServiceListener implements ServiceListener {
 public void serviceChanged(ServiceEvent e) {
  ServiceReference ref = e.getServiceReference();
  if(((String)ref.getProperty("objectClass").equals("org.device.print.PrintService"))){
   switch (e.getType()) {
   case ServiceEvent.REGISTERED:
      println(ref + " has been registered by "+ ref.getBundle().getLocation()); break;
   case ServiceEvent.UNREGISTERING:
      println(ref + " is being unregistered"); break;
   case ServiceEvent.MODIFIED:
      println("properties of "+ref+" have been modified:");
      String[] keys = ref.getPropertyKeys();
      for (int i=0; i<keys.length; i++) println(keys[i] + "=" + ref.getProperty(keys[i])); break;
   } }}
 void println(String msg) {System.out.println("events: "+msg); } }
```

Prendre en compte l'enregistrement et le retrait de service (iii)

Exemple 2 :

```
public class Activator implements BundleActivator {
 final static String filterStr
                ="(&(objectClass=org.device.print.PrintService)(location=4th floor))";
 Map/*<ServiceReference,PrintService>*/ printservices;
 BundleContext context;
  public void start(BundleContext context) throws BundleException {
    this.context=context;
    printservices=new HashMap();
    BindingController ctlr=new BindingController(context,filterStr,printservices);
    ctlr.open();
    context.addServiceListener(ctlr);
  }
```

Prendre en compte l'enregistrement et le retrait de service (iv)

```
public class BindingController implements ServiceListener {
   Map/*<ServiceReference,Object>*/ services;
   String filterStr;
   Filter filter;
   BundleContext context;

   public BindingController(BundleContext context, String filterStr, Map services){
      this.context=context;
      this.filterStr=filterStr;
      this.services=services;
      filter=context.createFilter(filterStr);
   }
```

Prendre en compte l'enregistrement et le retrait de service (v)

```
…
   public void open() {
      // fill the services map
      ServiceReference[] refs=context.getServiceReferences(null,filterStr);
      for(int i=0;i<refs.length;i++){
         Object svc = context.getService(refs[i]);
         if(svc!=null) services.put(refs[i],svc);
      }
   }
 public void close() {
   // release the references to service
      …
   }
…
```

Prendre en compte l'enregistrement et le retrait de service (vi)

```
…
public void serviceChanged(ServiceEvent e) {
    ServiceReference servref = e.getServiceReference();
    Object ref;
    switch (e.getType()) {
    case ServiceEvent.REGISTERED:
        if(filter.match(servref)){
            println(servref + " (from "+ servref.getBundle().getLocation() + ") is added");
            services.put(servref,context.getService(servref));
        };
        break;
…
```

Prendre en compte l'enregistrement et le retrait de service (vii)

```
....
case ServiceEvent.UNREGISTERING:
   ref=services.remove(servref);
   if(ref!=null) {
      println(servref + " is removed");
      context.ungetService(servref);
   } break;
case ServiceEvent.MODIFIED:
   ref=services.get(servref);
   if(ref!=null && !filter.match(servref)){
      println(servref + " is removed since properties has changed");
      services.remove(servref);
      context.ungetService(servref);
   } break; }}
```

Prendre en compte l'enregistrement et le retrait de service (viii)

Mini conclusion

- Vous avez suivi ?

Solutions

- La classe utilitaire ServiceTracker (OSGi R2)
  - Ne gère pas le cycle de vie
- Service Component Runtime (OSGi R4)
  - ~ ADL pour cycle de vie et liaison
  - Repris de ServiceBinder (IMAG/LSR/ADELE)

Modèle simple de composants orienté service

- Gère le cycle de vie du composant
  en fonction des dépendances obligatoires de services

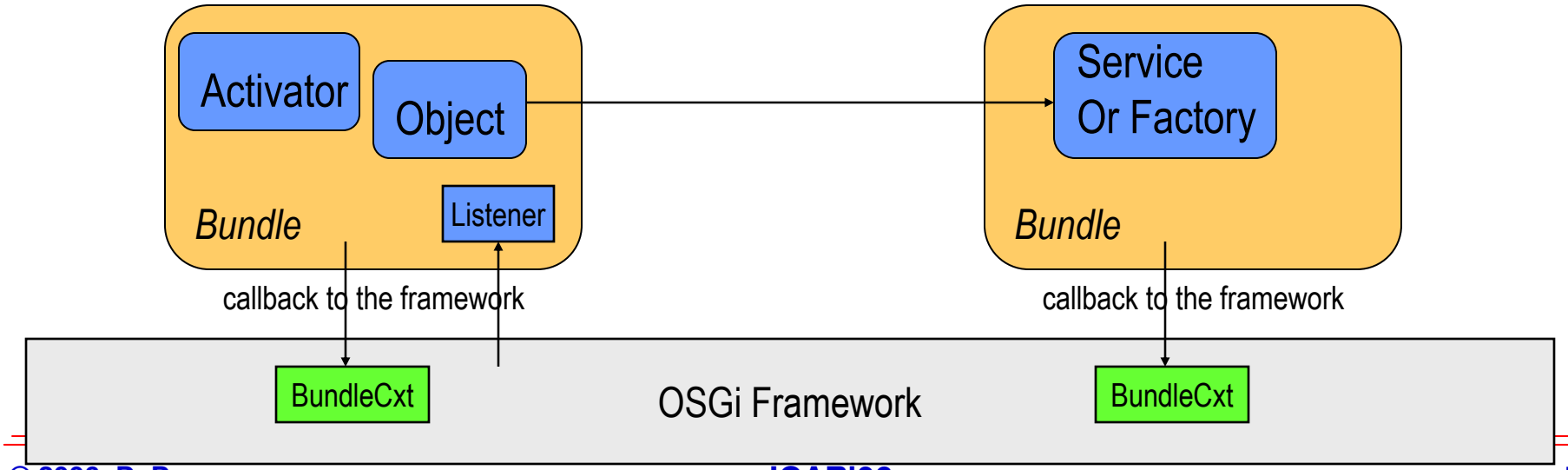
Caractéristiques

- Fabrique de Services
- Activation retardée (*lazy-activation*)
- Gestion des liaisons
  - stratégie événementielle
  - stratégie de recherche (via le contexte)
- Entrée du manifeste : `Service-Component`
- Descripteur XML
  - xmlns:scr="http://www.osgi.org/xmlns/scr/v1.0.0"
- …

Service Component Runtime

Programmation SOC Dynamique

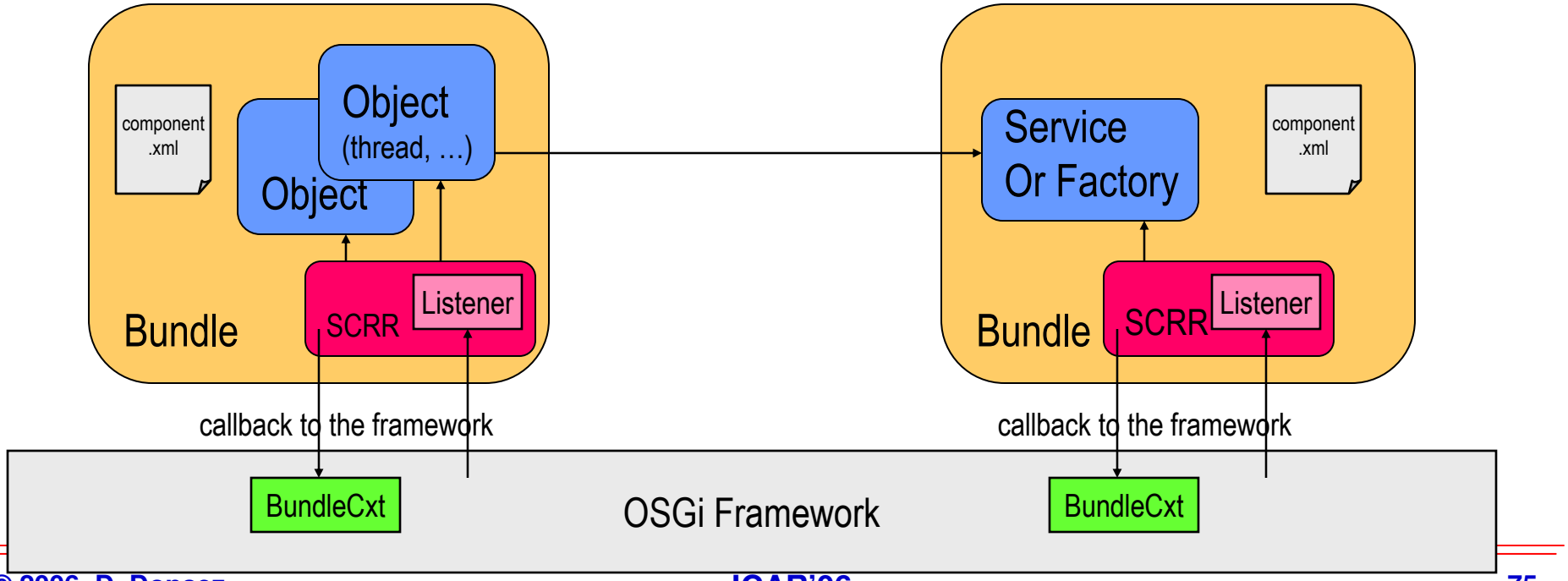
- Démarrage des instances de services
- Listeners pour gérer la liaison dynamique vers les services requis



Service Component Runtime

- **Gestion descriptive des instances de services et des liaisons**
  - **component.xml**

SCRR: SCR Runtime



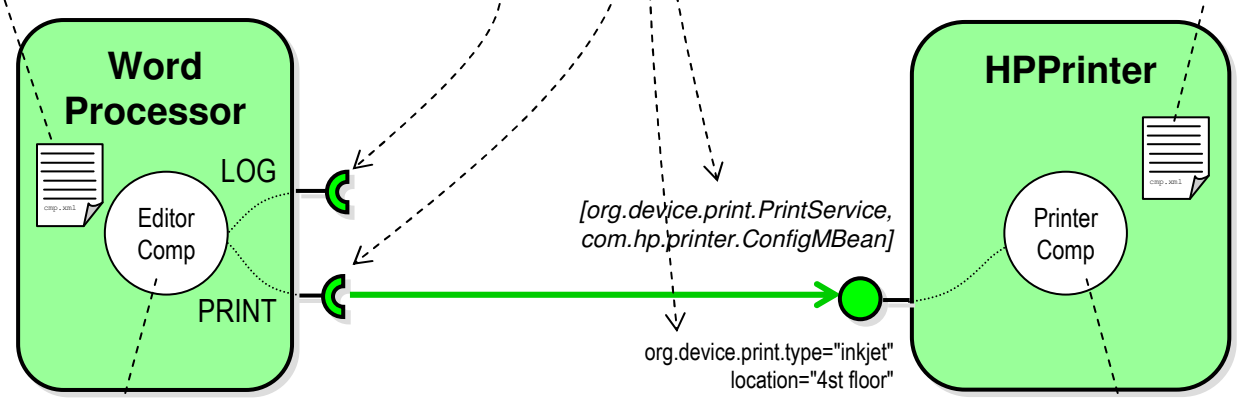
Service Component Runtime

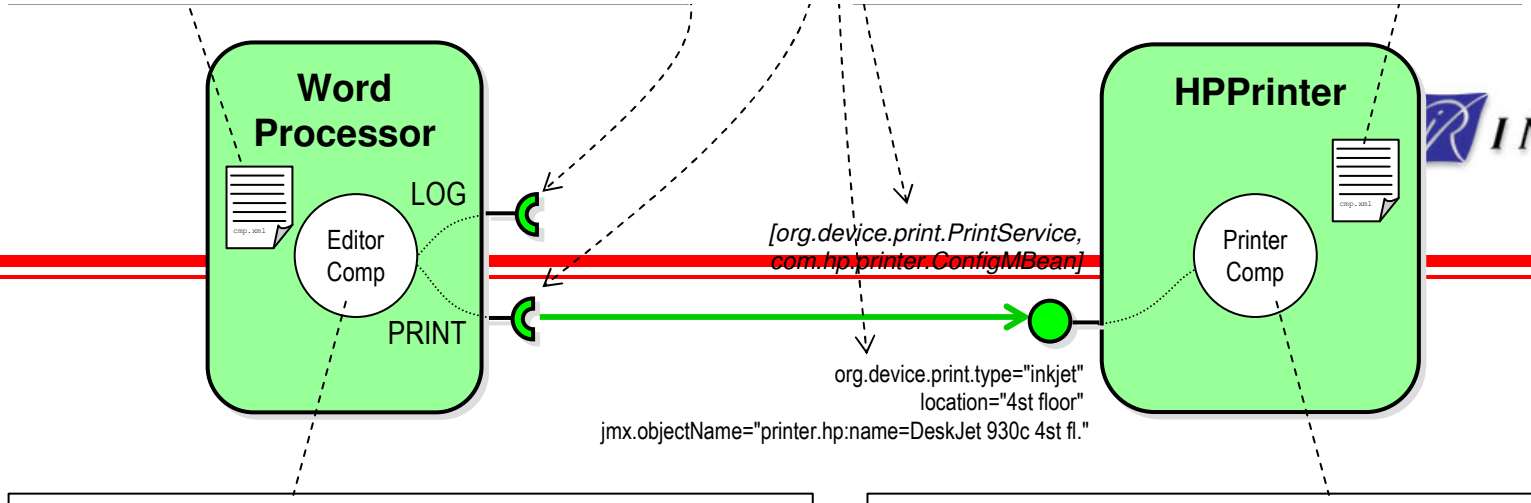
```
<component name="editor">
   <implementation
      class="org.eclispe.texteditor.impl.EditorComp"/>
   <reference name="PRINT"
         interface="org.device.print.PrintService"
         target="(&(location=*)(org.device.print.type=inkjet))"
         cardinality="1..n"
         policy="dynamic"
         bind="bindPrintService"
         unbind="unbindPrintService"
   />
   <reference name="LOG"
         interface="org.osgi.service.log.LogService"
         cardinality="0..1"
         policy="dynamic"
   />
</component>
```

/OSGI-INF/component.xml

```
<component name="printer">
   <implementation
      class="com.hp.printer.deskjet.impl.PrinterComp"/>
   <property name="org.device.print.type"
            value="inkjet" type="String"/>
   <property name="location"
            value="4st floor" type="String"/>
   <property name="jmx.objectName"
            value="printer.hp:name=DeskJet 930c 4st fl."
            type="String"/>
   <service>
      <provide interface="org.device.print.PrintService"/>
      <provide interface="com.hp.printer.ConfigMBean"/>
   </service>
</component>
```

/OSGI-INF/component.xml






```
public class EditorComp {
   private List printServices = new ArrayList()

   public void activate(ComponentContext ctxt) {
      LogService log = (LogService)ctxt.locateService("LOG");
     if(log!=null) log.log(LogService.LOG_INFO, "Editor starting"));
       log=null; // must release this reference
       // démarre la thread principale du traitement de texte
     …
   }
   public void deactivate(ComponentContext ctxt) {
      // arrête la thread principale
      …
     LogService log = (LogService)ctxt.locateService("LOG");
     if(log!=null) log.log(LogService.LOG_INFO, "Editor stopped"));
     log=null;
  }
  public void bindPrintService(PrintService ref){
      synchronized (printServices ){
          printServices.add(ref); }
   }
   public void unbindPrintService(PrintService ref){
      synchronized (printServices ){
         printServices.remove(ref); }
   }
   …
}
```

```
public class PrinterComp
  implements PrintService, ConfigMBean {

   // méthodes de PrintService
   public Job[] list(){
     ...
   }

   public Job print(InputStrean in, Dictionary printParams )
      throws PrintException {
     …
   }

   // méthodes de ConfigMBean
   public long getPrintedPageCounter(){
     …
   }

   public void setDefaultTrailer(int trailerId){
     …
   }
   …
}
```

Autres travaux

- Fractal : FROGi / KROGi, ProActive, …
- OSGi AspectJ
- Dependencies Manager
- iPOJO
- Eclipse ECR (Enterprise Component Runtime)
- Spoon OSGi
- Spring / OSGi
- EasyBeans/OSGi *(EJB3.0, JSR 181 (WS-Metadata))*
- *…*

iPOJO (Felix)

Motivations

- Exécuter des pur POJOs
- pour en faire des services ou utilisés des services
  sans utiliser l'API org.orgi.framework

Principes

- Injection de bytecode pour instrumenter les membres (compile time)
- Intercepter *xload/xstore* et *invokex* pour passer le contrôle à des *handlers*
- Les handlers travaillent en fonction des metadonnées décrites dans un descripteur (.mf,.xml, …)

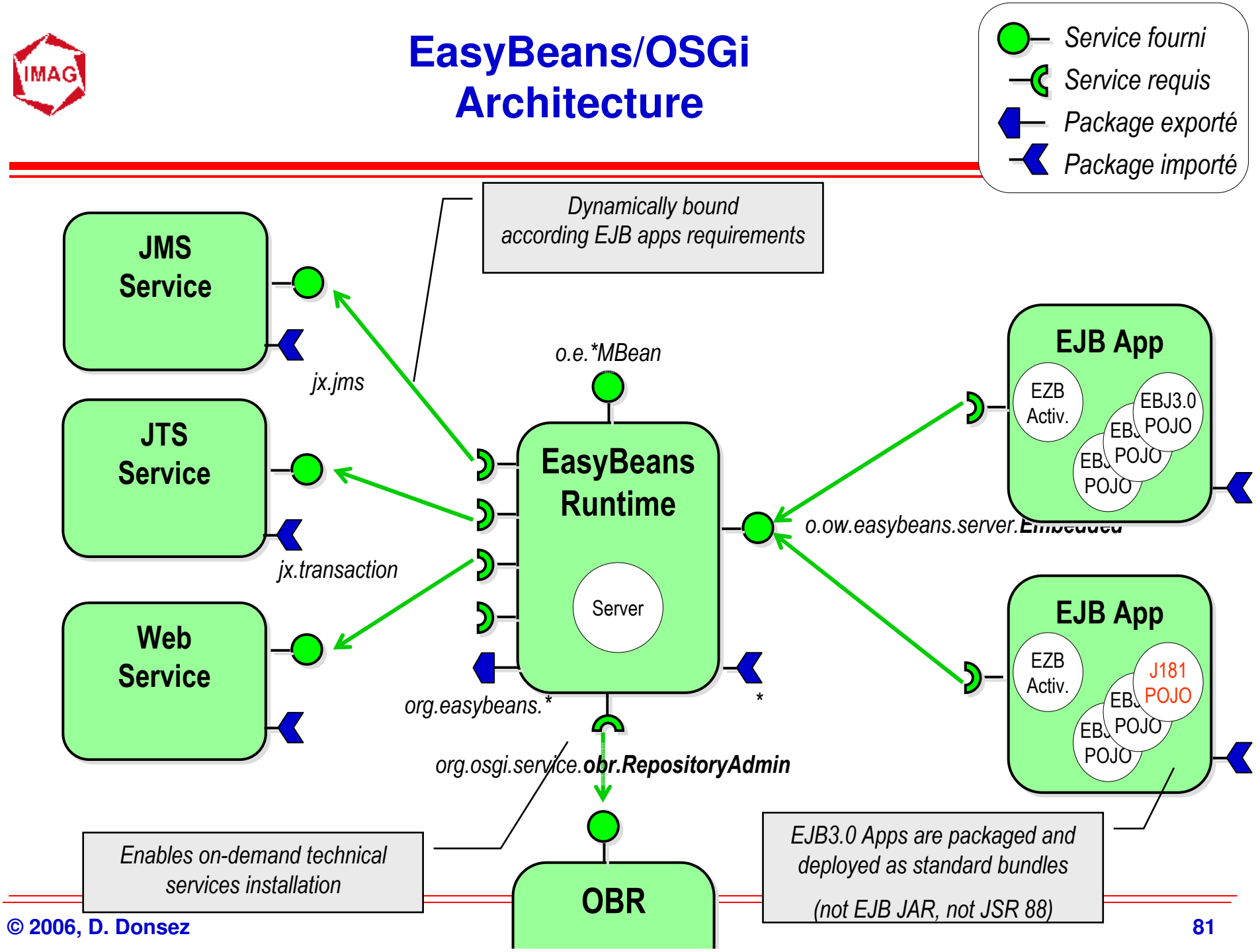
http://plop-plop.net/ipojo

EasyBeans/OSGi

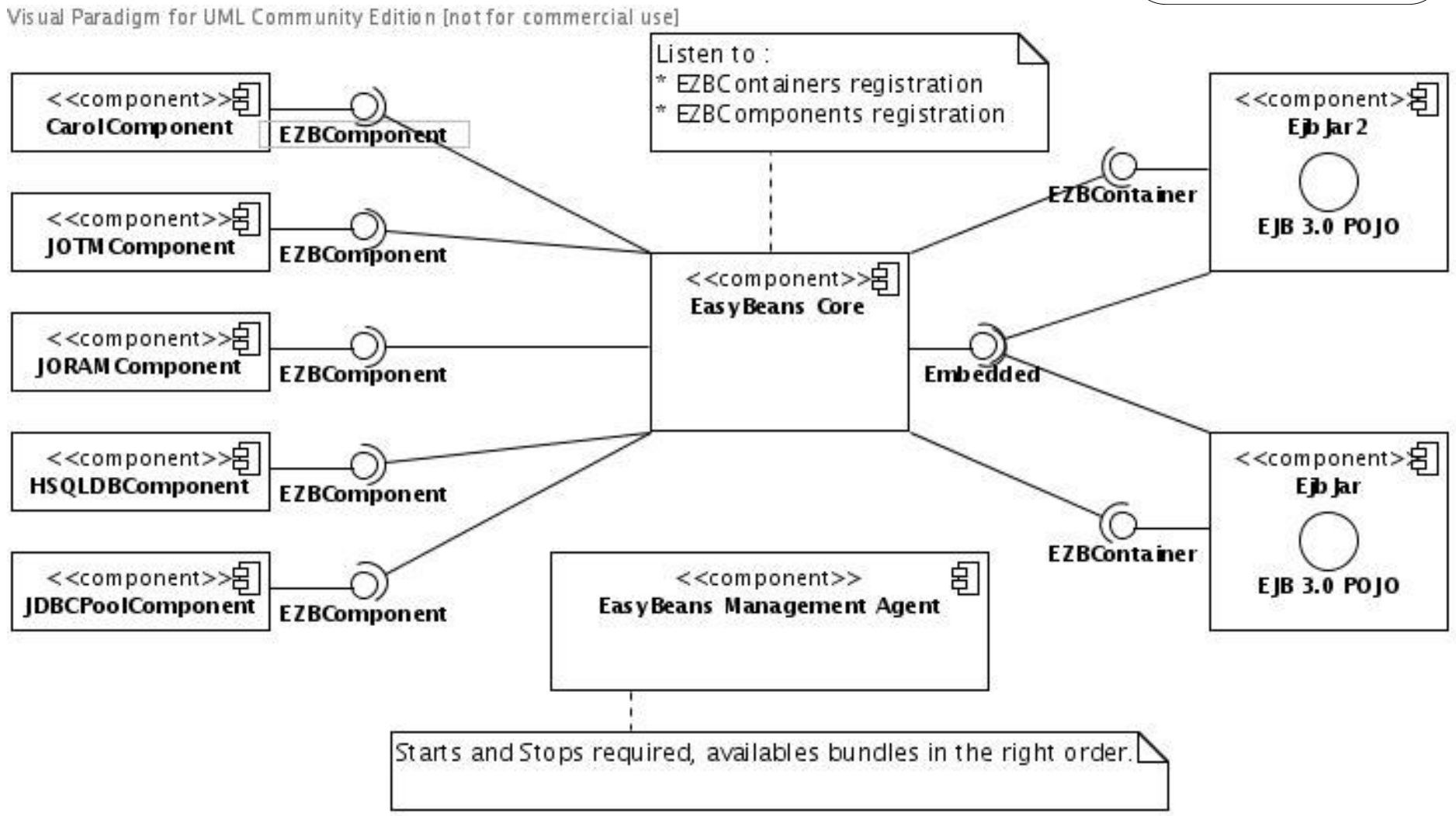
- **EasyBeans www.easybeans.org**
  - Containeur EJB3.0 – JSR 220 (annotations 5.0) *+ JSR 181*
    - *JSR 220 ~= « EJB for the dummies »*
    - *JSR 181 = Web Services Metadata for the Java™ Platform*
- **Motivations**
  - Exécuter des POJOs annotés JSR-220 sur OSGi
- **Principes**
  - Conditionner les classes des POJOs dans un bundle
    - l'ejbjar est emballé dans le bundle mais plus de JSR88 !
  - L'activateur crée un CL qui analyse et injecte les .class annotés et utilise le service du Runtime d'EasyBeans
  - Le Runtime d'EasyBeans enregistre les EB Homes et fait l'intermédiaire avec les services techniques requis

http://wiki.easybeans.org/xwiki/bin/view/Main/OSGi

EasyBeans/OSGi Architecture



EasyBeans/OSGi
Architecture (état 30/08/2006)



ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

---

OSGi

Guide de Bonnes pratiques

Modularité

- Séparer les classes « published » (ie contrat) des classes « propriétaires » dans des paquetages des différents
  - Seul les classes « published » doivent être exportées
- Conditionner les contrats et les implémentations dans des bundles séparés
  - Les contrats ne varient peu et sont partagés par plusieurs bundles
- Import-Package plutôt que Require-Bundle (R4)
  - substitutabilité avec d'autres fournisseurs de packages
- Limitez l'usage des fragments
- Evitez l'usage de DynamicImport-Package
  - Sauf cas particulier (livraison dynamique de plugin, …)

Conditionnement

- **JAR enfouis**
  - Ne déconditionnez pas les JAR
    - Utilisez le Bundle-ClassPath
  - Conservation des signatures, manifestes, …
  - Possibilité de les patcher !
    - Bundle-ClassPath: patch.jar,original.jar

Service (i)

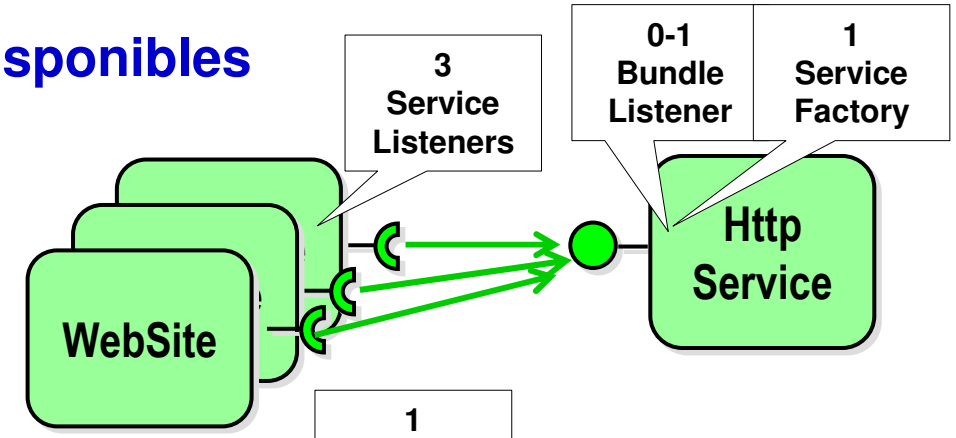
- **Rappel**

1. **Code de fourniture d'un service << Code d'usage d'un service**

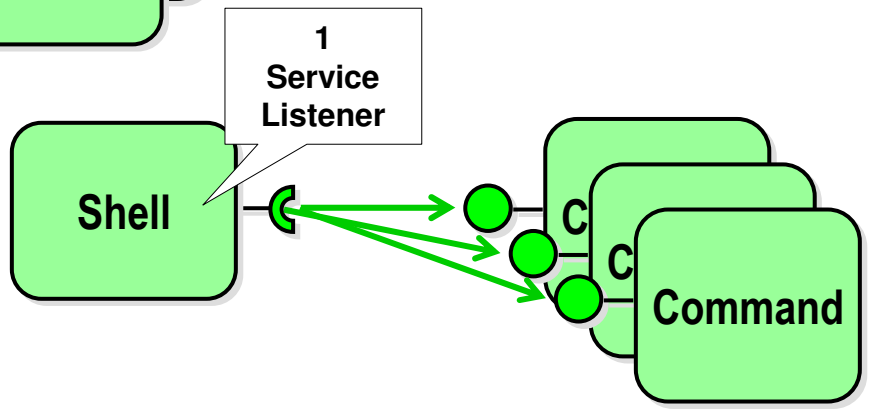
2. **1 service = 1 entrée dans le registre**
   - **Courtage sur plusieurs milliers/millions de services**

Service (ii)

- 2 patrons disponibles
- *Listener*



- *Whiteboard*



- Preferez (le plus possible) le patron *Whiteboard* (sauf si (2))
  - Listeners Considered Harmful: The “Whiteboard” Pattern
  - http://www.osgi.org/documents/osgi_technology/whiteboard.pdf

Service (iii)

- **Granularité**
- ***Objet << Composant << Service << Bundle***

Membres statiques

- **AIE !**
- **Evitez les dans les classes « published »**
  - **Difficile de tracker le cycle de vie du bundle qui les fournit**
- **Il est fréquent de voir**

```
class MyActivator implements BundleActivator {
  public static BundleContext bundleContext;
  …
}
```

Chargeurs de classe (ClassLoaders)

- **Usage**
  - Dynamic bytecode injection
  - Dynamic aspect-weaving
  - Dynamic annotation processing
  - …
- **Principe**
  - Le chargeur de classe doit avoir pour parent le chargeur du bundle
    - ie MyActivator.class.getClassLoader()
  - Le chargeur peut demander la récupération des ressources (.class) au bundle
    - Enumeration e = bundle.findEntries("/", "*.class", true);
- **Exemples**
  - ProActive, Julia (FROGi), EasyBeans, …

Cycle de vie

- **Vous développez le start()**
- **Pensez aussi au stop()**
  - **Proscrire System.exit()**
    - **Votre bundle n'est pas le seul sur la JVM**
    - **il y en a d'autres qui bossent**
  - **Libérez les ressources**
    - **Fermez les fichiers, *sockets*, connections, …**
  - **Arrêtez les *threads* démarrés**
    - **Mieux encore rendez les au service de *pool* de *threads***
  - ***Nullifiez* les références vers les servants**
    - **Garbage collection (objets et classes)**

ICAR'06

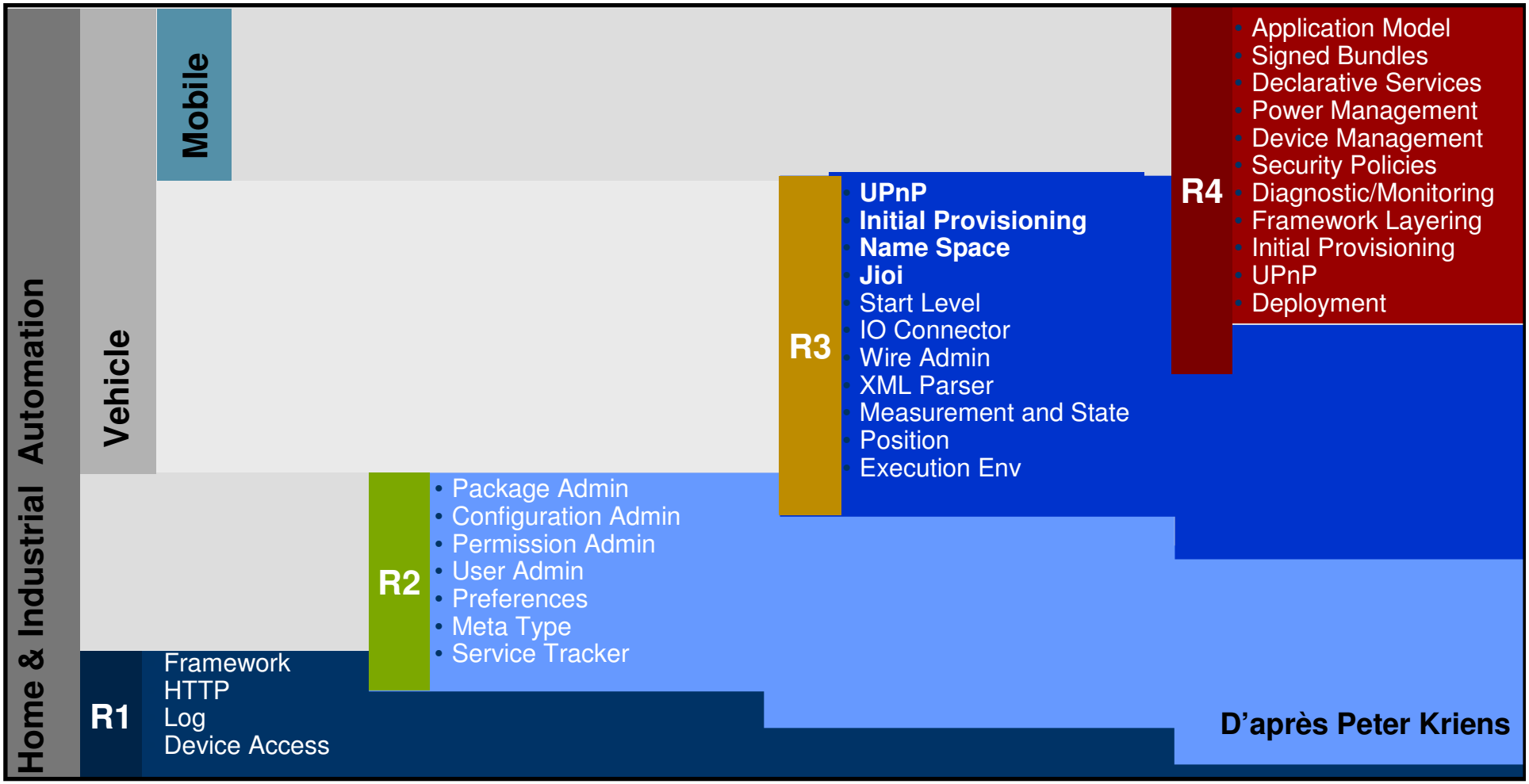
École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

---

OSGi

Les services standard

Les services OSGi standard



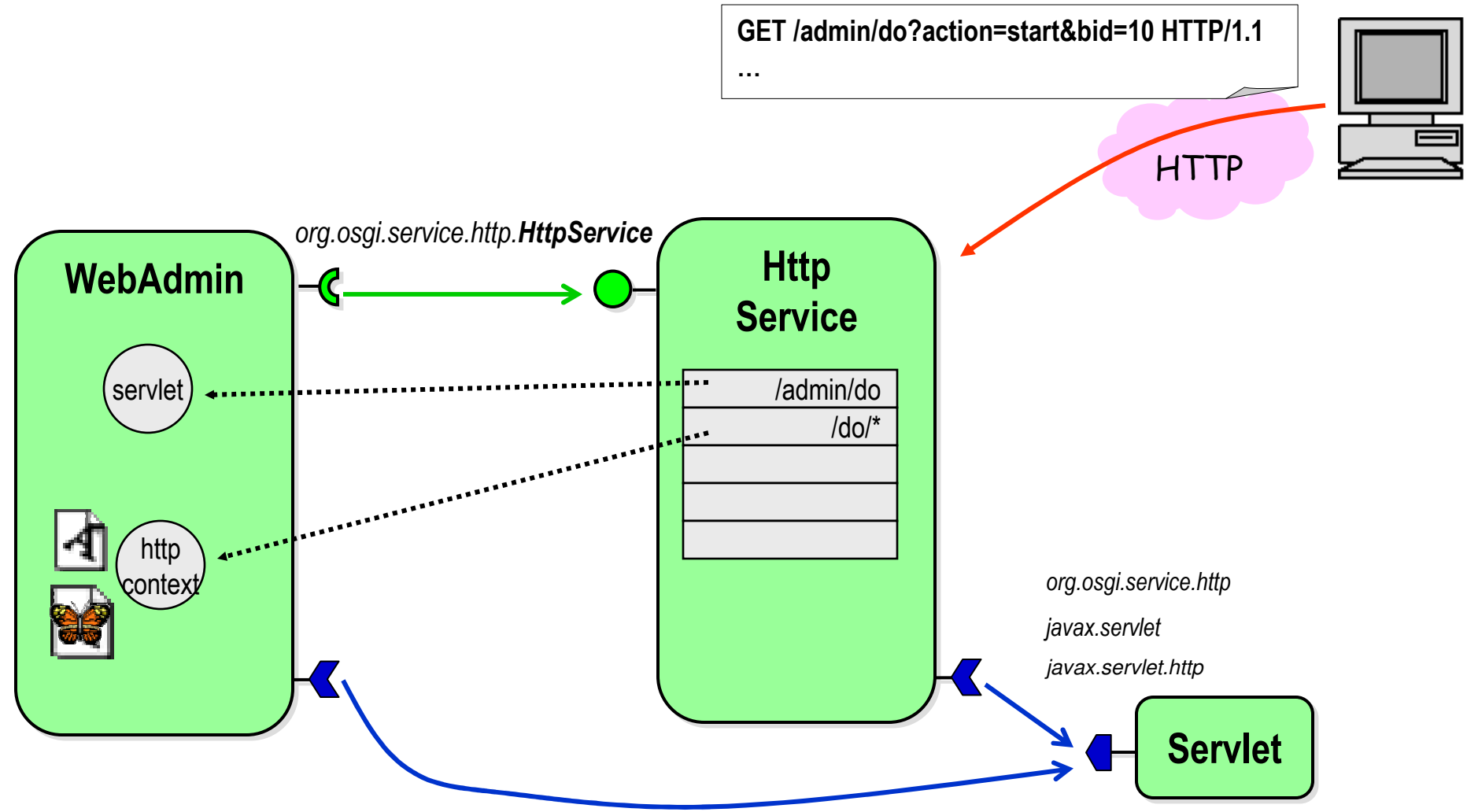
HttpService

R1

- Service permettant à d'autres bundles de publier des servlets et ressources par HTTP
  - Important : Web-based management

- Implémentations
  - embarquent un serveur HTTP compact (Jetty,…)
  - Authentification et Autorisation (BasicSchema, SSL)
  - Servlets Web-Services : XML-RPC, kSOAP, SOAP/HTTP
- Extra
  - <jspc> pour éviter d'embarquer un compilateur de JSP
  - Convertisseurs WAR to Bundles
  - Canevas Web (Cocoon, Wicket …)

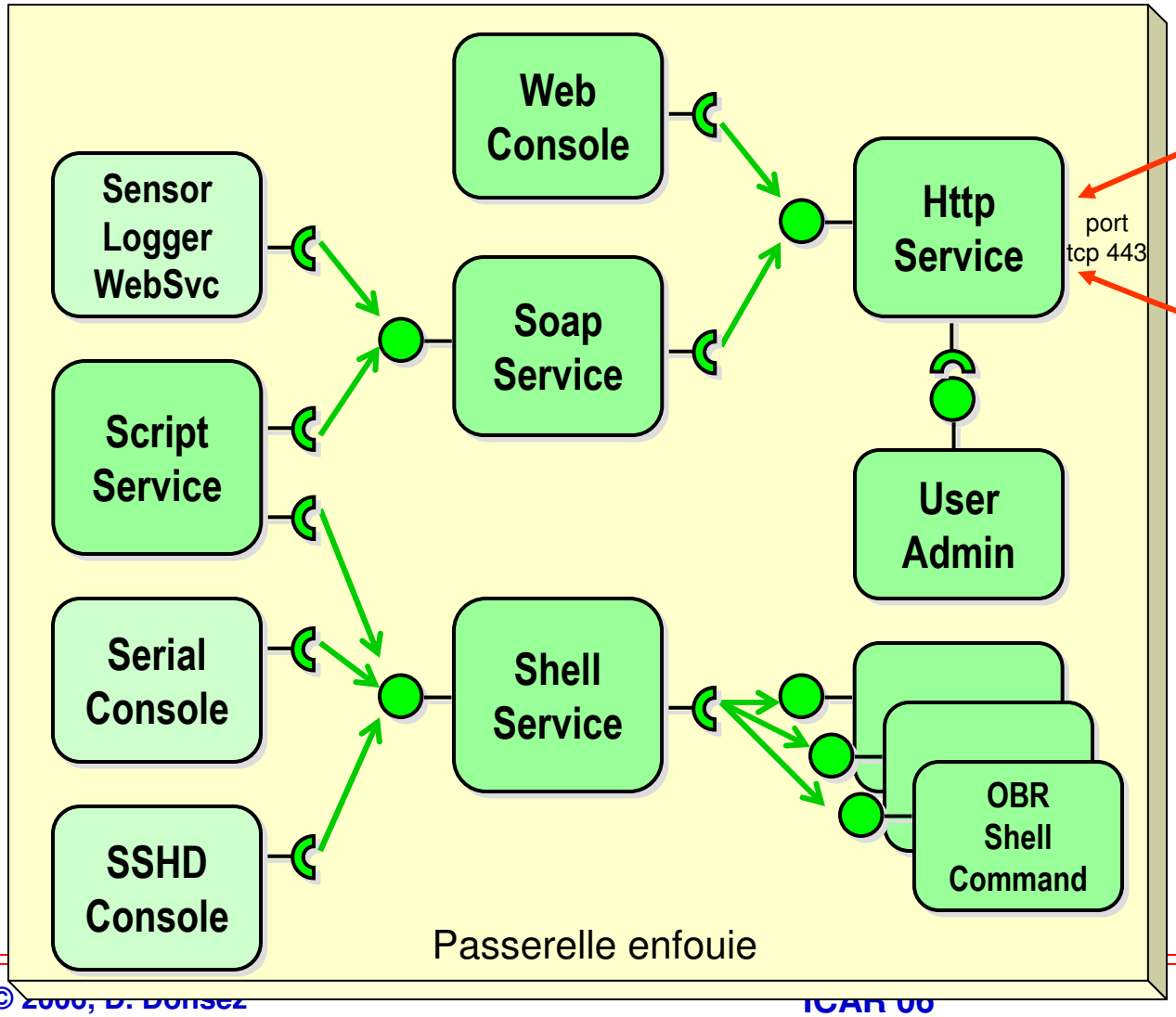
HttpService : Usage (i)

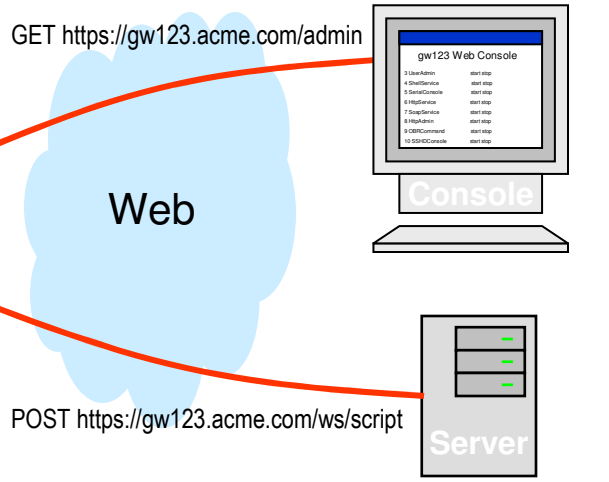


HttpService : Usage (ii)

```
HttpService http= bundleContext.getService(serviceReference);
String WEBROOT = "/webroot"; // embedded ressources in BUNDLE-CLASSPATH jarfiles
String WEBROOT_ALIAS = "/admin";
String SERVLET_ALIAS = WEBROOT_ALIAS +"/do";
Servlet servlet=new WebAdminServlet(param1,param2);
http.registerServlet(SERVLET_ALIAS, servlet, null, servlet);
HttpContext docsContext = new HttpContext() {
  public String getMimeType(String name) {
     return (name.endWith("htm"))?"text/html":null; }
  public boolean handleSecurity(HttpServletRequest req,HttpServletResponse resp) { return true; }
  public URL getResource(String name) {
     URL u = this.getClass().getResource(name);
     System.out.println(this.getClass().getName());
     return u;
  }
  http.registerResources(WEBROOT_ALIAS, WEBROOT, docsContext );
```

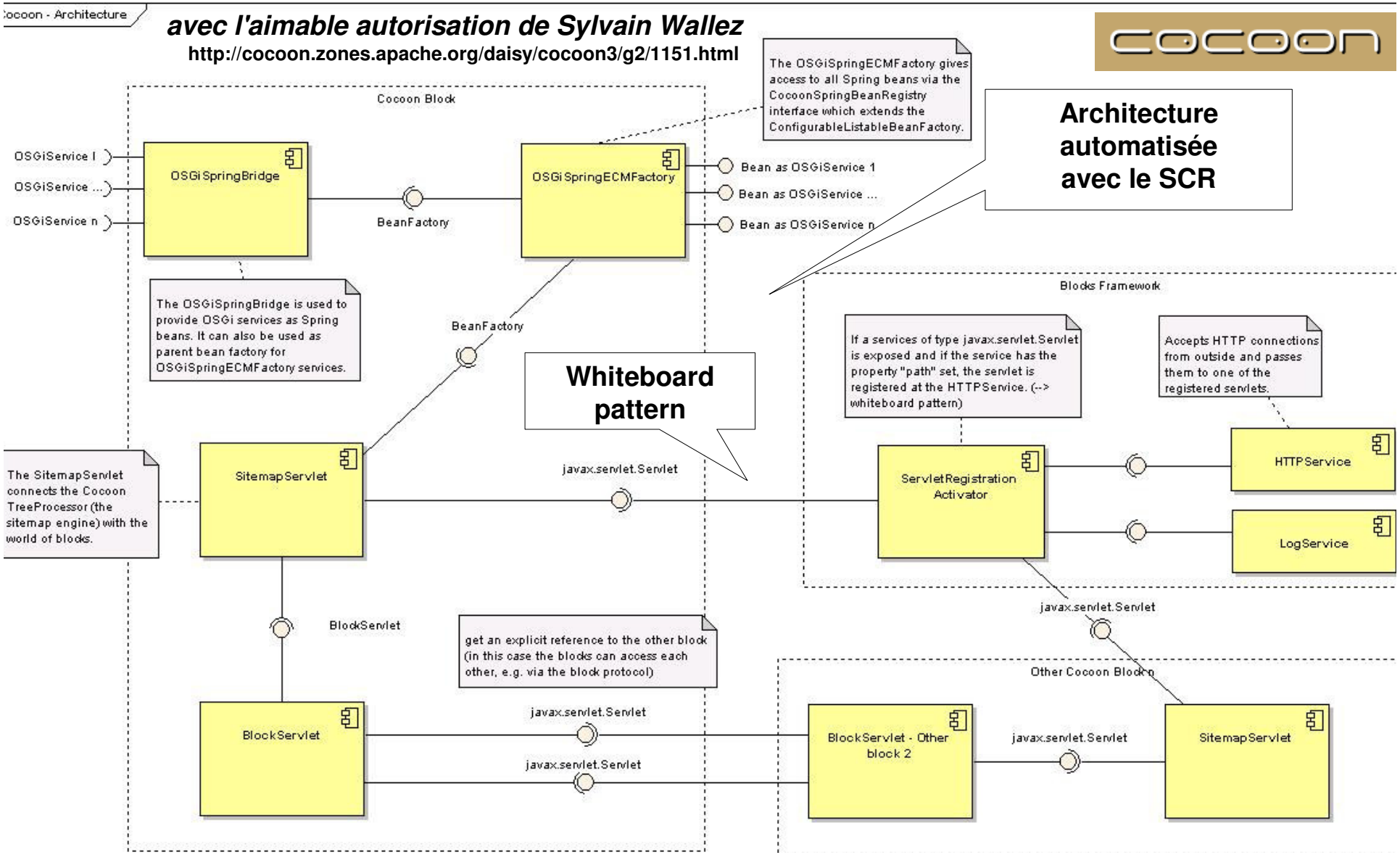
Exemple d'utilisation pour l'administration de passerelles






```
POST /ws/script  HTTP/1.1
...
<SOAP-ENV:Envelope xmlns:SOAP-ENV=...>
 <SOAP-ENV:Body>
  <s:ExecuteScript xmlns:s=...>
   <s:script SOAP-ENC:arrayType="s:cmd[3]">
    <cmd>obr install "SensorLoggerWebSvc"</cmd>
    <cmd>install http://bundleserver/dep.jar</cmd>
    <cmd>install http://bundleserver/main.jar</cmd>
   </s:script>
  </s:ExecuteScript>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un autre exemple: l'architecture de Cocoon 3.0



Un autre exemple: Wicket OSGi

Wicket

- *« Wicket is a **Java web application framework** that takes simplicity, **separation of concerns** and ease of development to a whole new level. Wicket pages can be mocked up, previewed and later revised using standard WYSIWYG HTML design tools. Dynamic content processing and form handling is all handled in Java code using a first-class component model backed by **POJO data beans** that can easily be **persisted** using your favourite technology » from JavaGeek.org*

Wicket sur OSGi

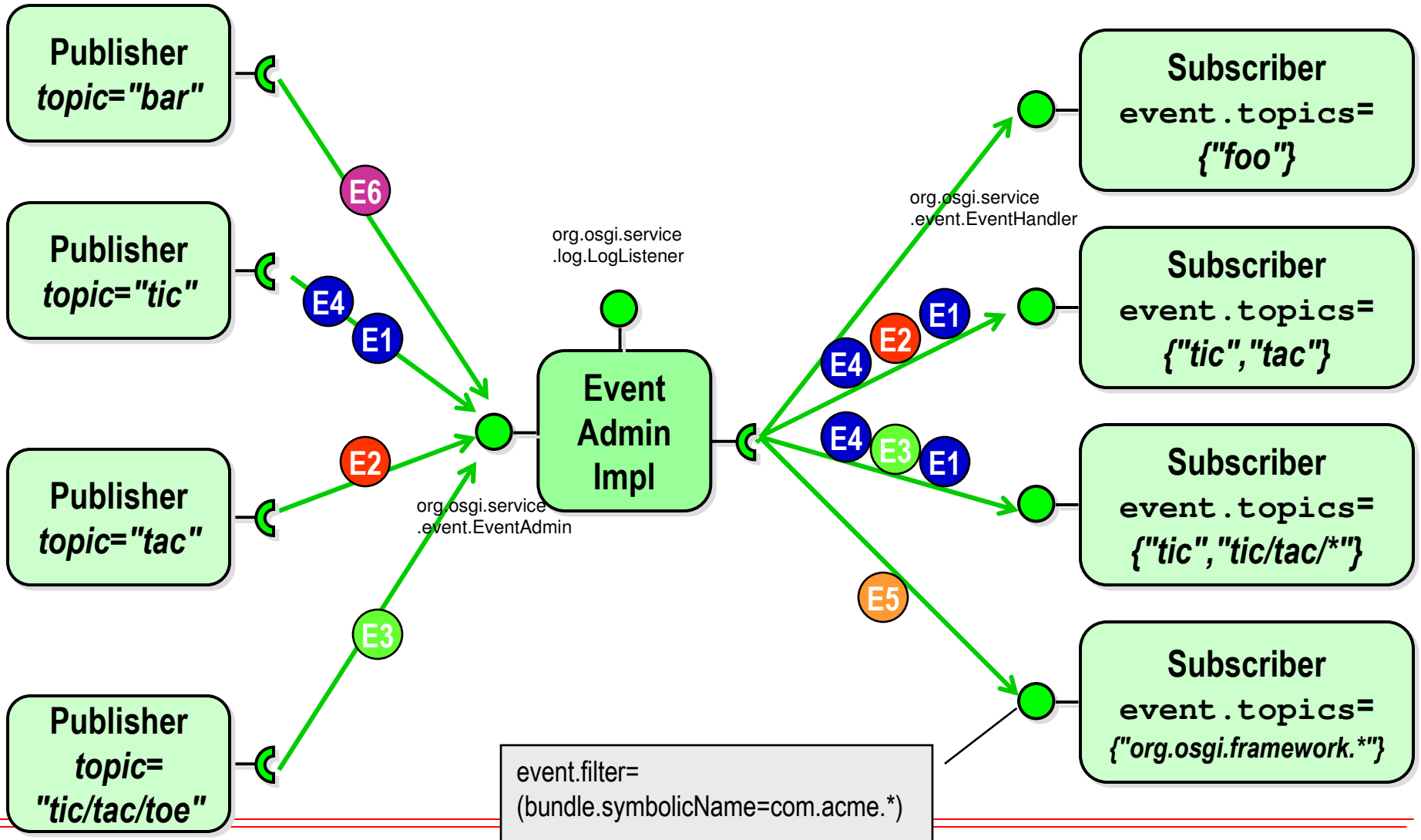
- **TODO**
- **Utilise le SCR**

- **http://www.wicket-wiki.org.uk/wiki/index.php/OSGi**

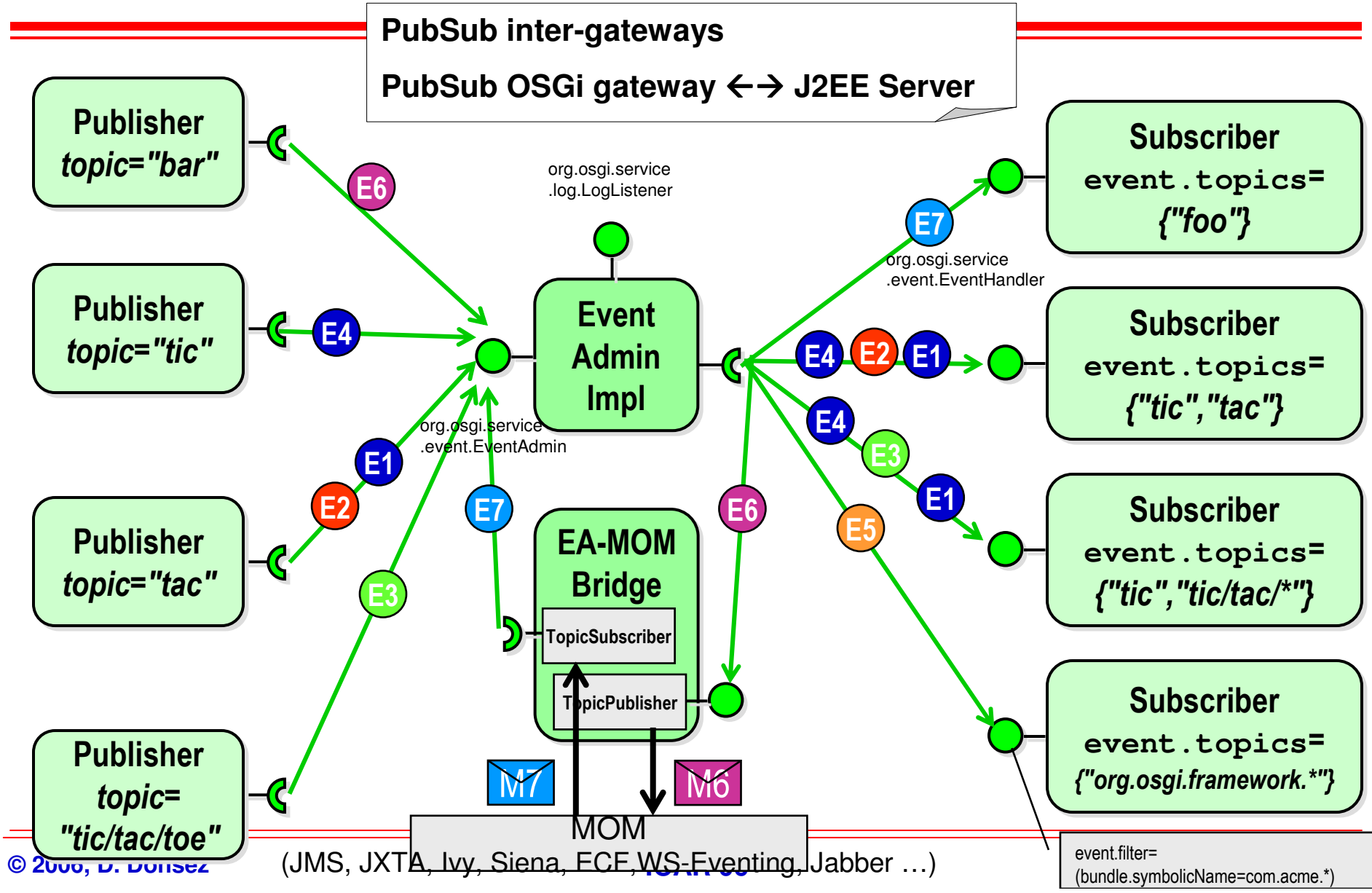
Event Admin Service (i) r4

- Offre un modèle de communication événementiel entre les bundles.
- Objet `Event` = *topic* + propriétés.
- Médiateur de Publication-souscription d'événement
  - L'éditeur poste un événement au service `EventAdmin`
  - L'`EventAdmin` le diffuse en parallèle à tous les souscripteurs du *topic*.
  - Chaque souscripteur enregistre un service `EventHandler`.
  - L'éditeur peut être synchronisé (ou non) sur la terminaison des exécutions // de tous les services `EventHandler` concernés.
- Remarque
  - Evénements spéciaux liées
    - au cycles de vie des Services, bundles et framework
    - au LogService, UPnP Base Driver, …
  - Le service `EventAdmin` gère une *liste noire* des `EventHandler` défectieux ou consommant trop de CPU.

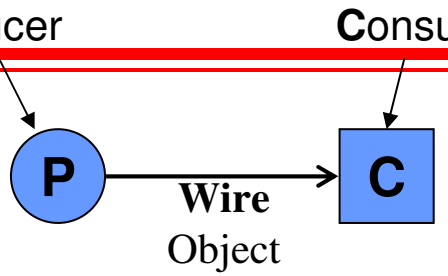
Event Admin Service (ii)



Bridging Event Admin Service and MOM

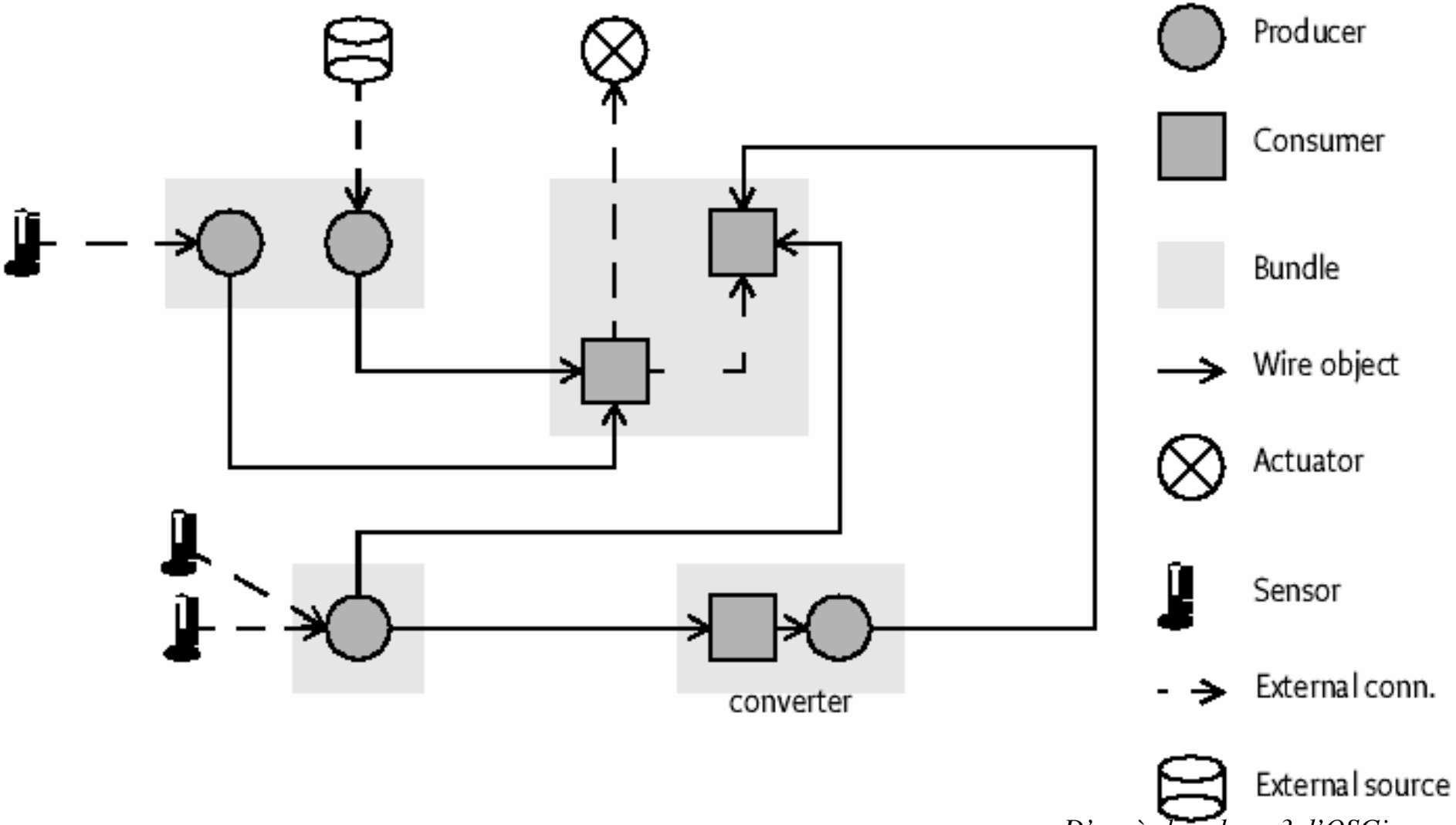


Wire Admin Service



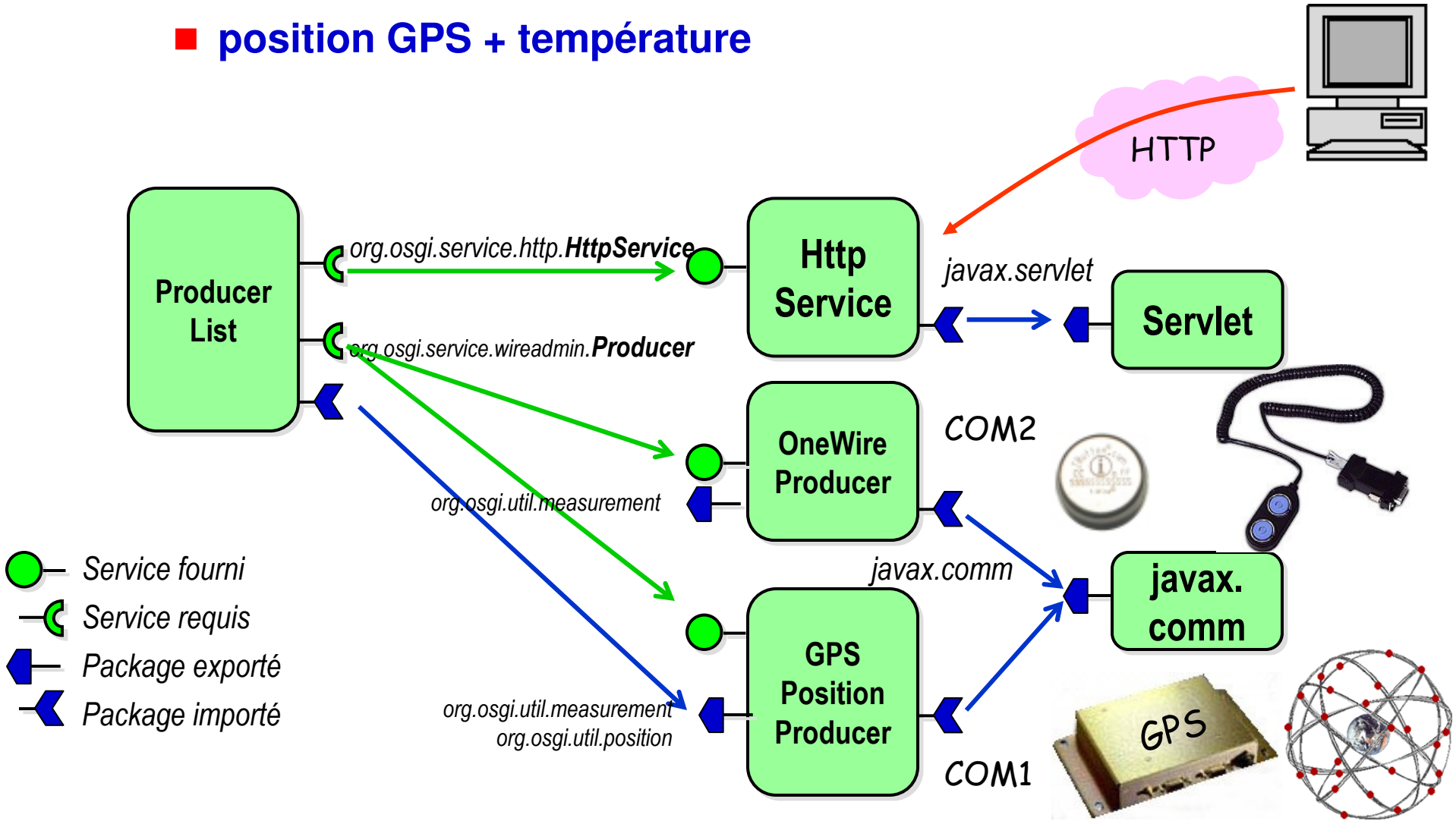
- **Motivation**
  - Patron (*design pattern*) de services producteurs-consommateurs de données
    - Données : mesure physique, position, état (discret), …
- **Domaine d'application**
  - Services basés Capteurs
  - Machine-to-Machine (M2M)
- **WireAdmin**
  - Médiateur reliant 0..N producteurs à 0..M consommateurs
    - Administrable globalement
      - WireAdmin, WireAdminListener
  - Contrôle de la « comptabilité » et adaptation de types de données échangées au travers du Wire
    - Flavors

Wire Admin Service
Patron *Producer-Wire-Consumer*



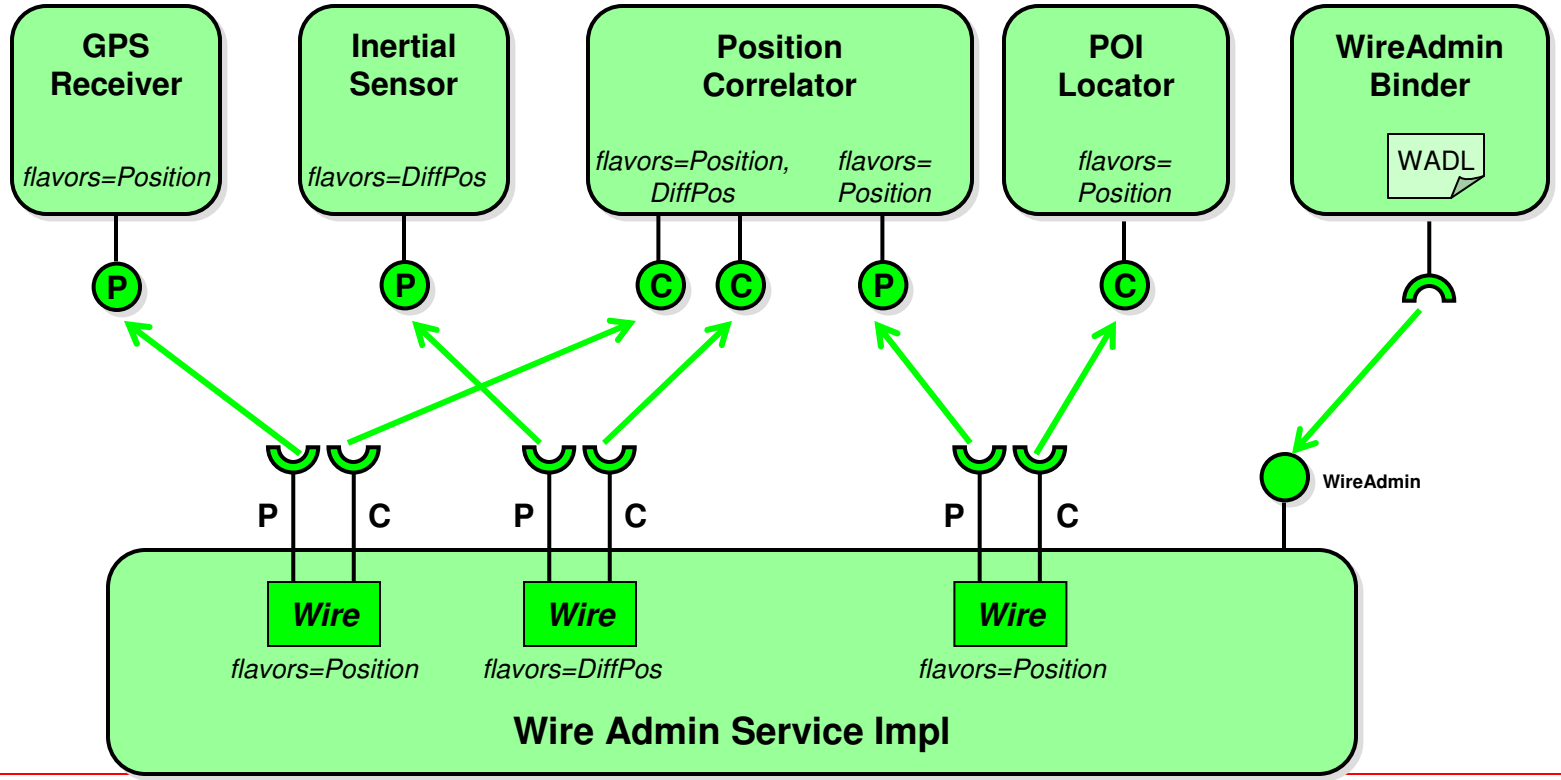
Wire Admin Service
Exemple d'application M2M (i)

- Consultation de mesures via le Web
  - position GPS + température



Wire Admin Service
Exemple d'application

Aide à la navigation

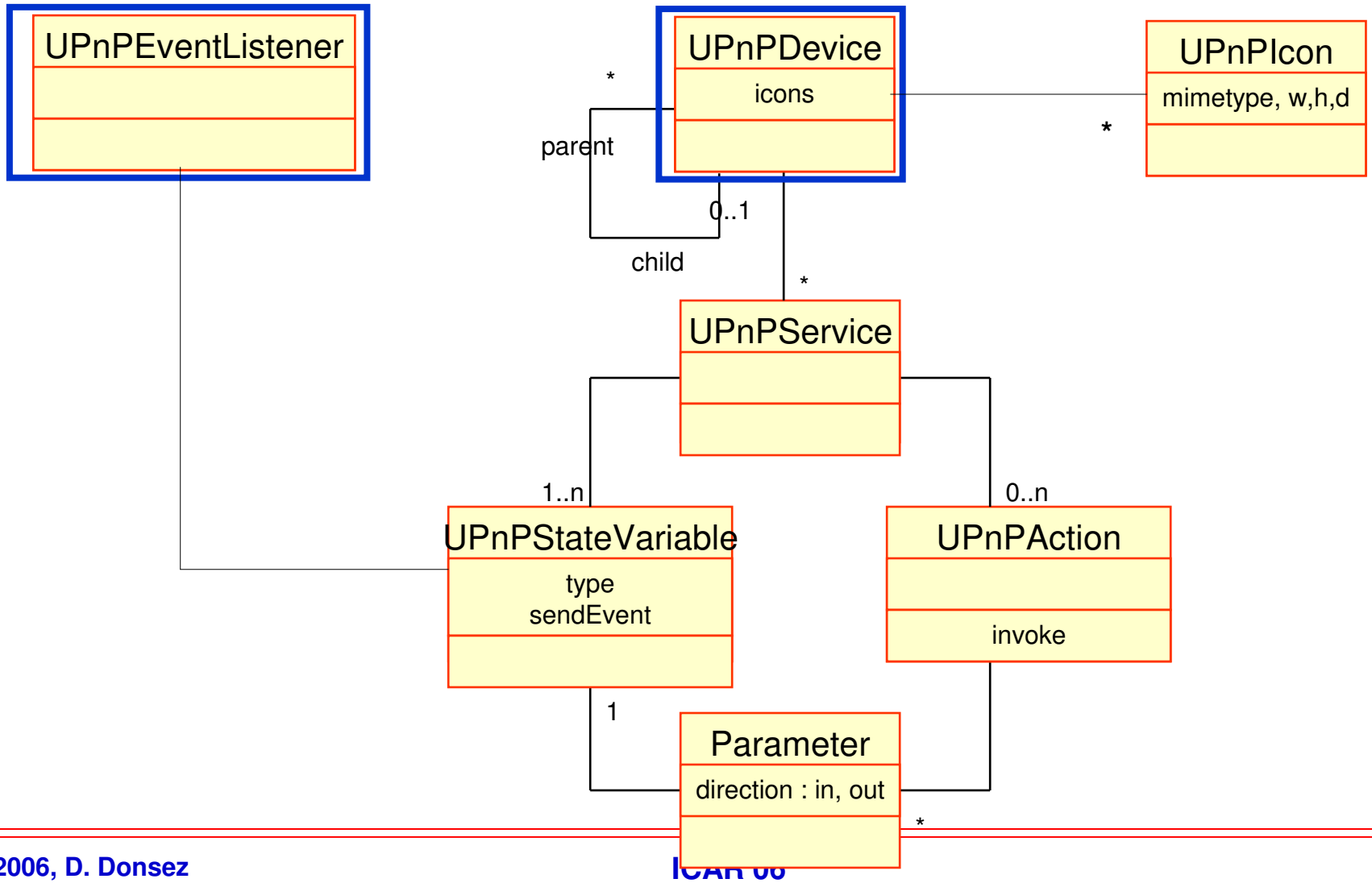


UPnP Driver Service R3

- **UPnP (Universal Plug and Play)**
  - Protocoles de découverte d'équipements (SOHO) et d'utilisation leur services
    - Basé sur SOAP/HTTP (TCP, UDP, UDP Multicast)
  - Alternative à JINI
  - Largement répandu et soutenu par les équipementiers SOHO

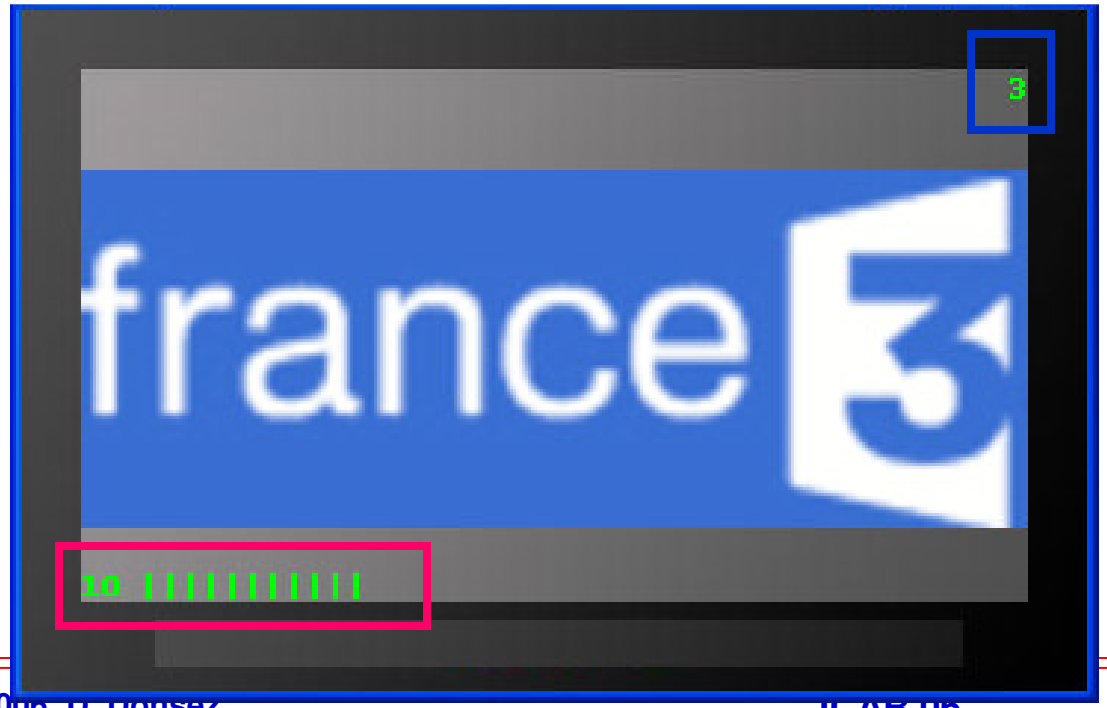
- **Motivations du service UPnP Driver Service**
  - Spécifie comment des bundles OSGi doivent être développés pour interoperer avec
  - des equipéments UPnP (devices)
  - des points de contrôle UPnP (control points)
  - en respectant la spécification UPnP Device Architecture

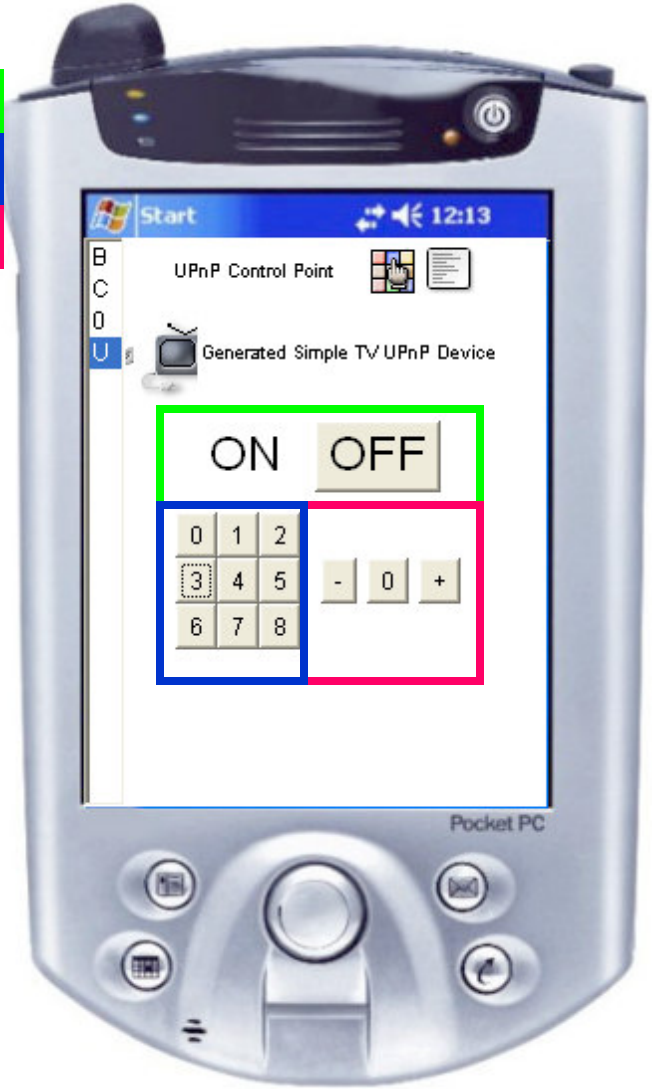
UPnP Driver Service
Les interfaces représentant les équipements UPnP



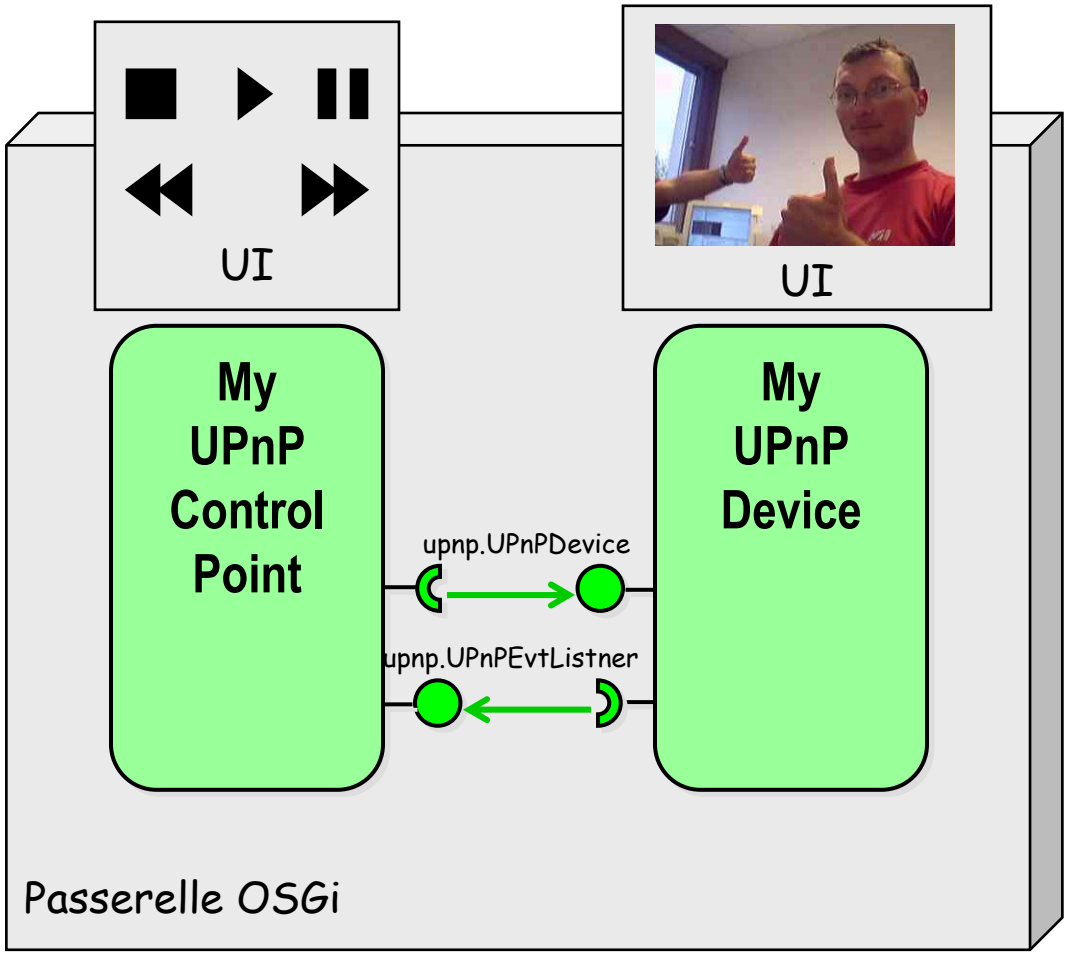
Exemple: un *device* Téléviseur et son point de contrôle

- 3 services
  - urn:schemas-upnp-org:service:SwitchPower:1
  - urn:*schemas-adele-imag-fr*:service:ChannelSelector:1
  - urn:*schemas-adele-imag-fr*:service:VolumeSelector:1

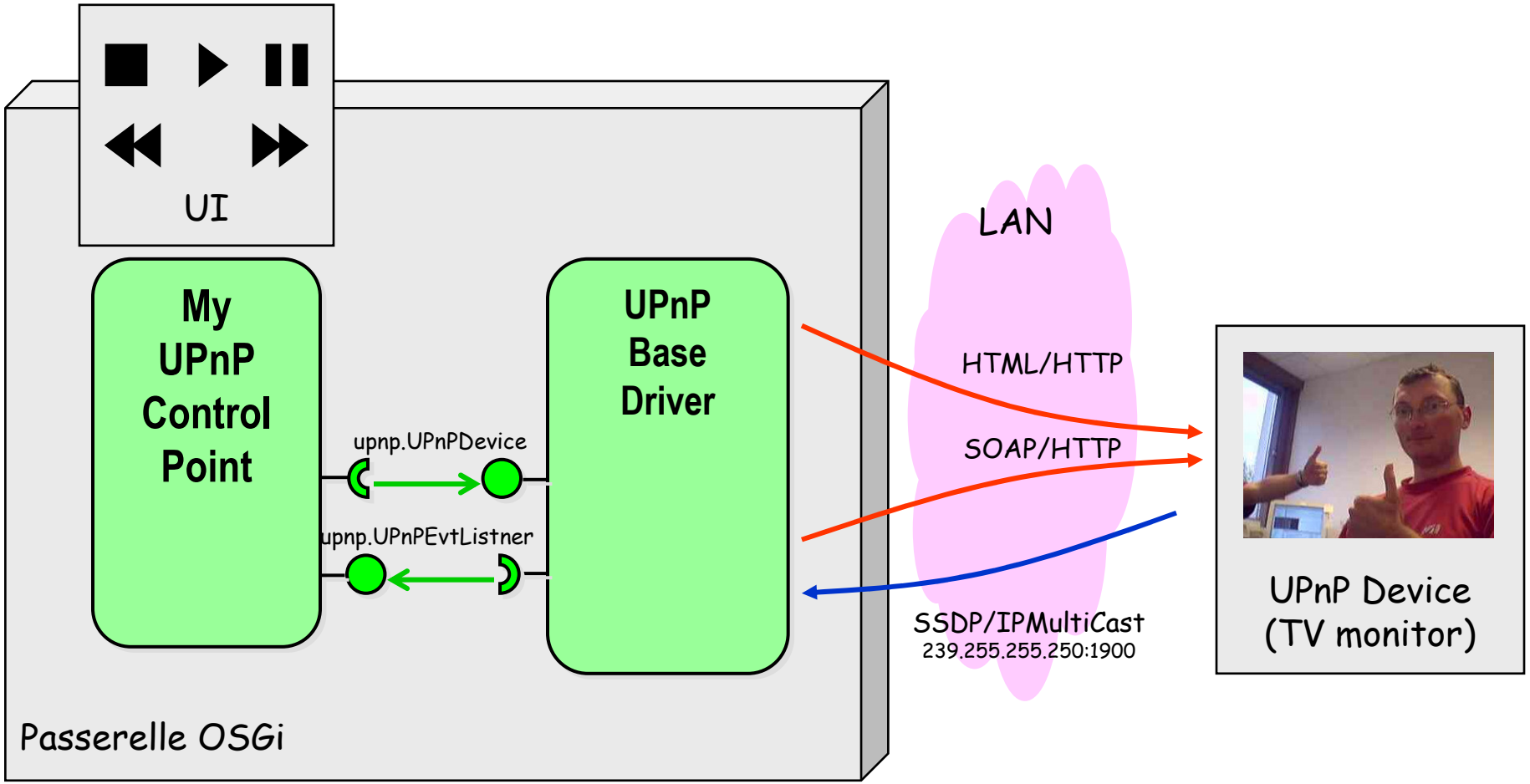




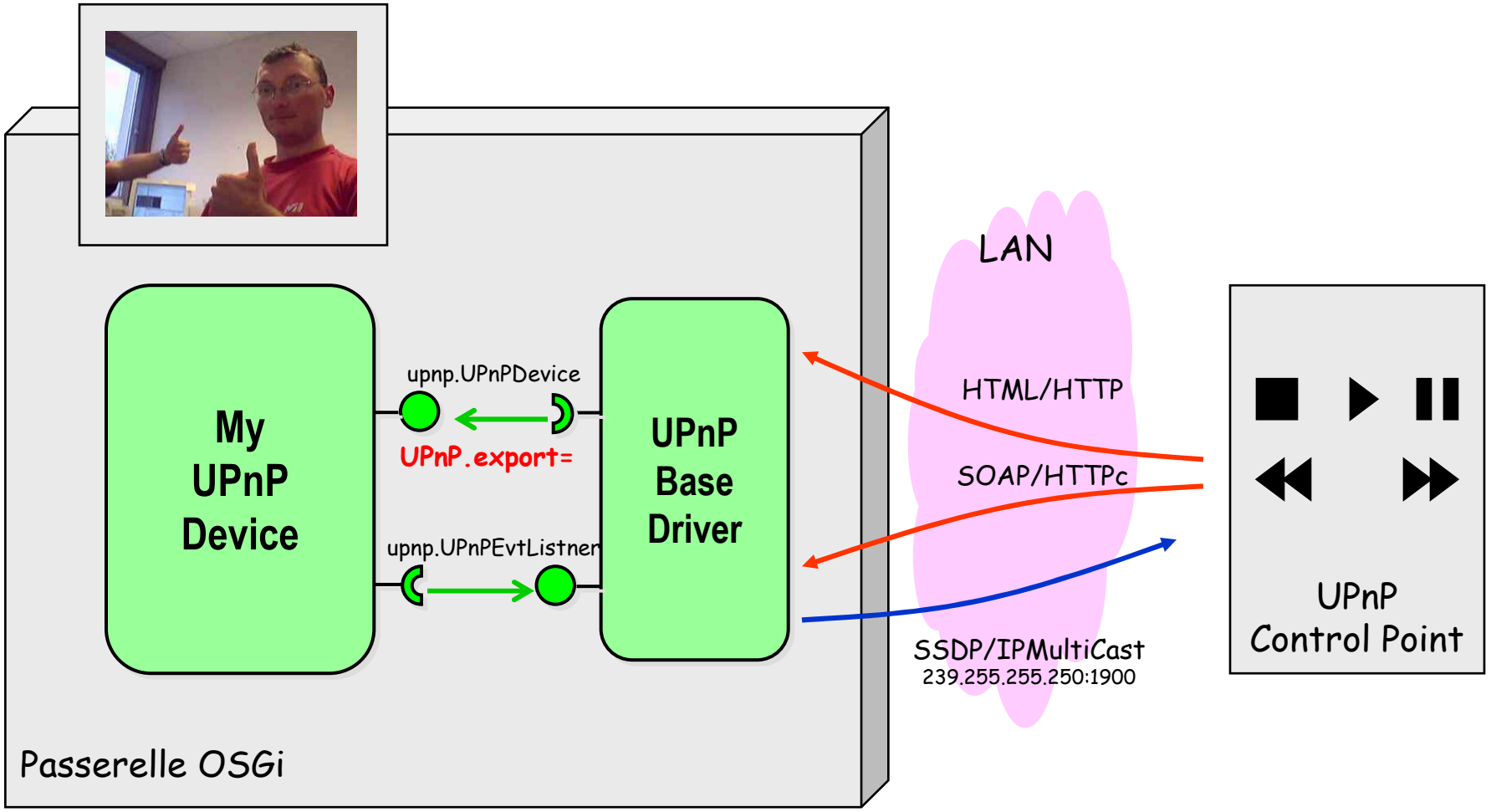
UPnP Device Driver : Mise en œuvre Collocalisé



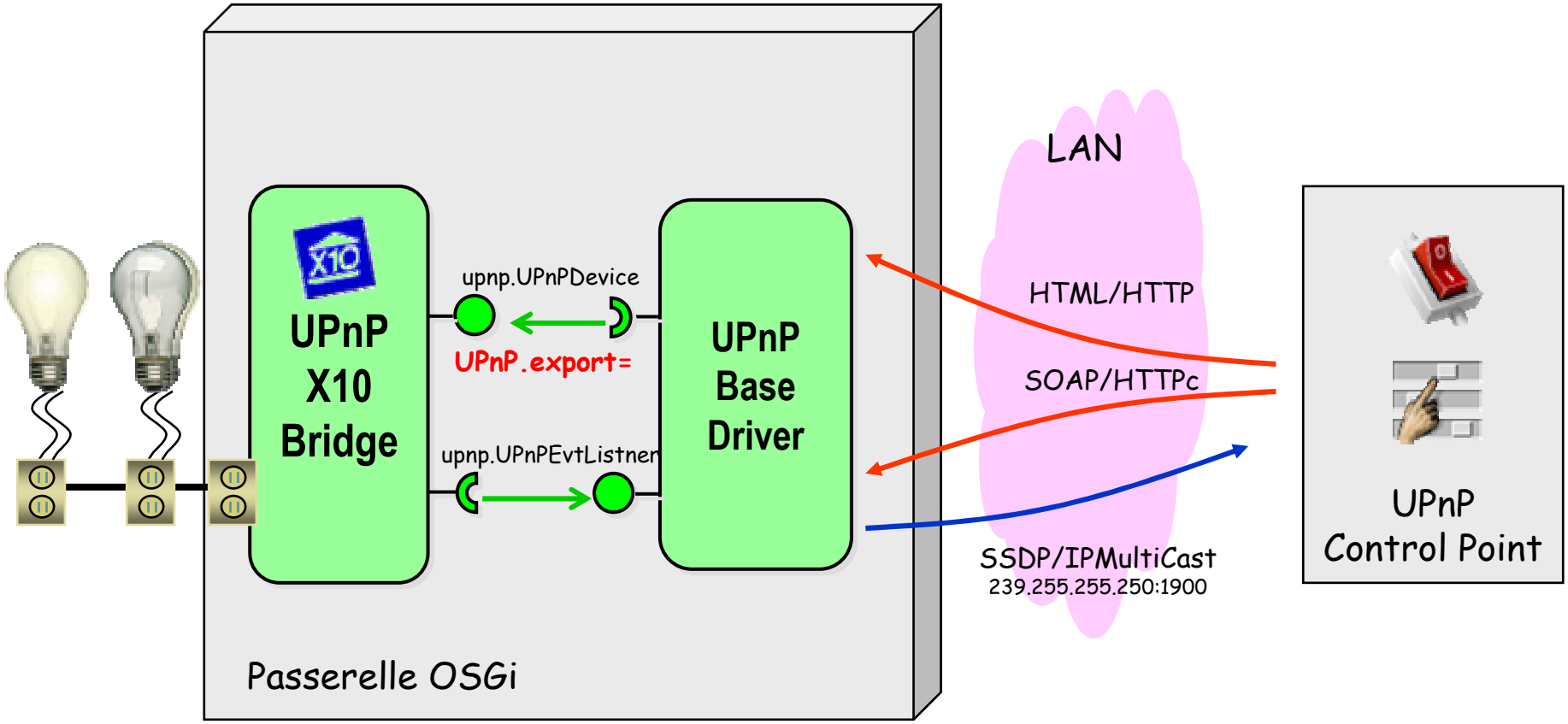
UPnP Device Driver : Mise en œuvre
Point de contrôle



UPnP Device Driver : Mise en œuvre Equipement



UPnP Device Driver : Mise en œuvre
Passerelle micro-monde



Conclusion intermédiaire

+ Gestion des dépendances de package
+ Déploiement dynamique de bundles
+ Environnement varié:
  embarqué, station de travail, serveur.
+ Fonctionnalité de base pour le déploiement de composants

- Programmation complexe des connexions entre services
  à la charge du développeur
- Centralisé mais pas mal de travaux sur la distribution

Q&R

ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

---

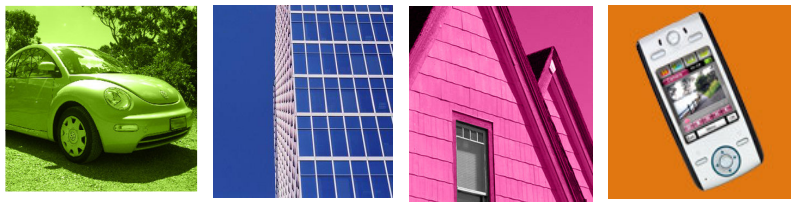
OSGi

Acteurs, concurrences, tendances et perspectives

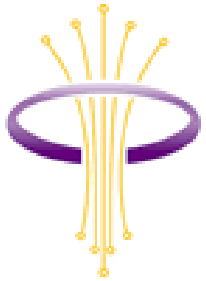
L' OSGi™ Alliance

- actuellement 44+ membres
- de plusieurs domaines industriels

- sur les 4 segments



L'OSGi™ Alliance



OSGi

OPEN SERVICES GATEWAY INITIATIVE

=

Alpine Electronics Europe Gmbh , Aplix Corporation , Belgacom , BMW Group , Cablevision Systems , Computer Associates , Deutsche Telekom AG , Echelon Corporation , Electricité de France (EDF) , Ericsson Mobile Platforms AB , Esmertec , Espial Group, Inc. , ETRI Electronics and Telecommunications Research Institute , ~~France Telecom~~ , Gatespace Telematics AB , Gemplus , Harman/Becker Automotive Systems GmbH , IBM Corporation , Industrial Technology Research Institute , Insignia Solutions , Intel Corporation , KDDI R&D Laboratories, Inc. , KT Corporation , Mitsubishi Electric Corporation , Motorola, Inc. , NEC Corporation , Nokia Corporation , NTT , Oracle Corporation , Panasonic Technologies, Inc. , Philips Consumer Electronics , ProSyst Software GmbH , Robert Bosch GmbH , Samsung Electronics Co., Ltd. , SavaJe Technologies, Inc. , Sharp Corporation , Siemens AG , Sun Microsystems, Inc. , Telcordia Technologies, Inc. , Telefonica I+D , TeliaSonera , Toshiba Corporation , Vodafone Group Services Limited

Les acteurs incontournables

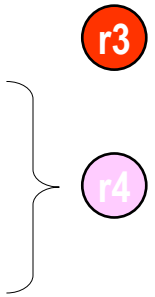
- **IBM**
  - OSGi est au cœur de la stratégie d'IBM
  - placé sur la partie « edge » du système IT
    - poste de travail (RCP)
    - serveur enfoui
  - Remarque:
    - Eclipse 3.0 (donc WebSphere Studio) est désormais développé au dessus d'OSGi (Equinox)
- **Nokia**
  - Pousse pour
    « Java (MIDLet) dans toutes les poches » (2002)
    « Java Server dans toutes les poches » (2005)

Produits

- ~~*SUN* Java Embedded Server (JES)~~ *l'étincelle*
- *Echelon LonWorks* Bundle Deployment Kit
- *Ericsson* - Residential e-services
- *Gatespace* AB
- *IBM* WebSphere Studio Device Developer
- *Insignia*
- *Nano Computer System*
- *Opensugar* Cube
- *ProSyst Software* mBedded Server
- *Wind River*
- *Siemens VDO TLA*
- …

La communauté open-source

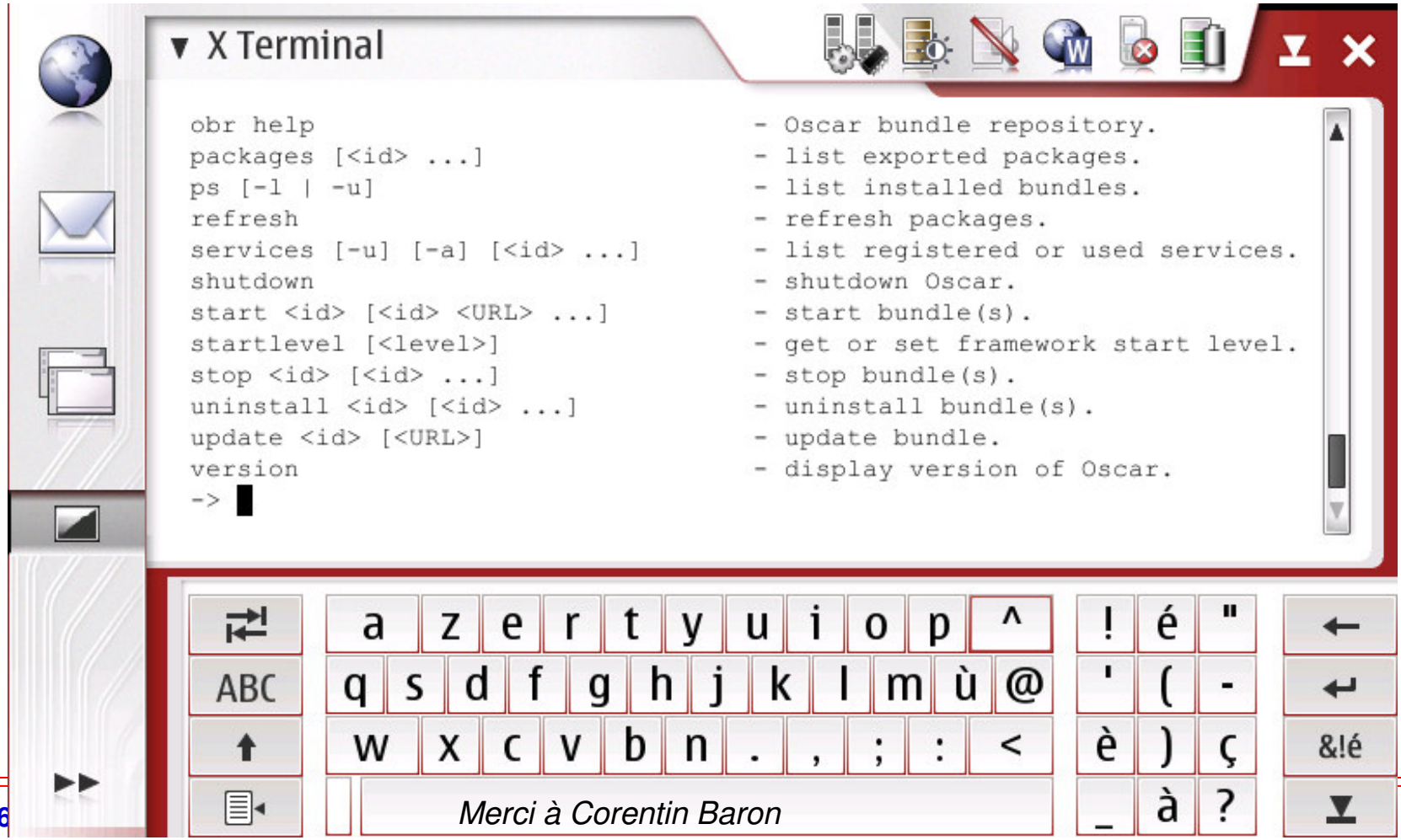
- **Plusieurs implémentations et communautés**
  - ObjectWeb Oscar
  - Knopflerfish
  - Eclipse Equinox (donation IBM SMF)
  - Apache Felix (suite d' ObjectWeb Oscar)



- **Abaissement des barrières de l'OSGi pour le développement open-source.**
  - Dépôts de bundles (org.osgi.service.obr)

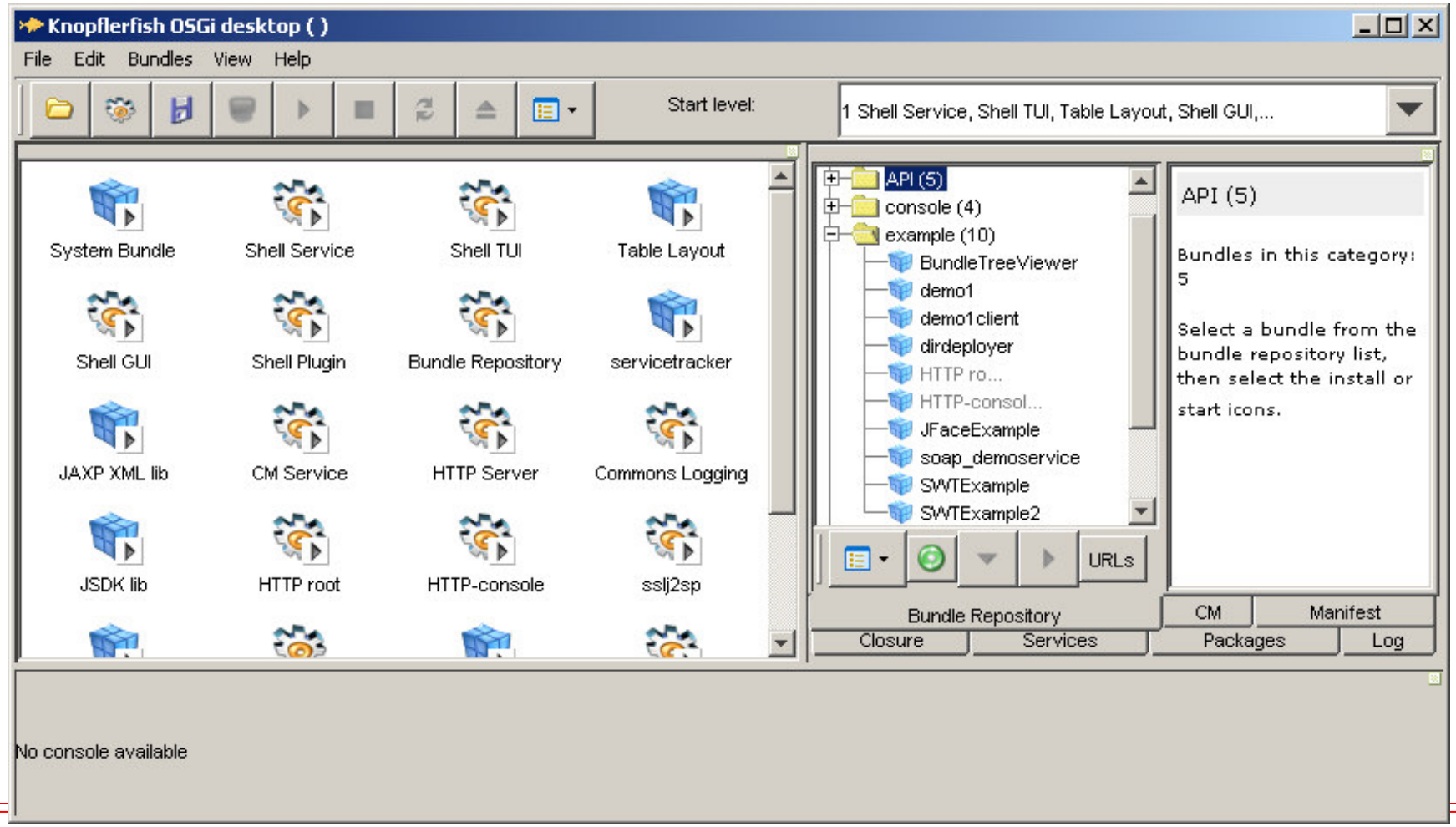
Oscar/Felix

- **Console texte sur Nokia 770 (JamVM)**



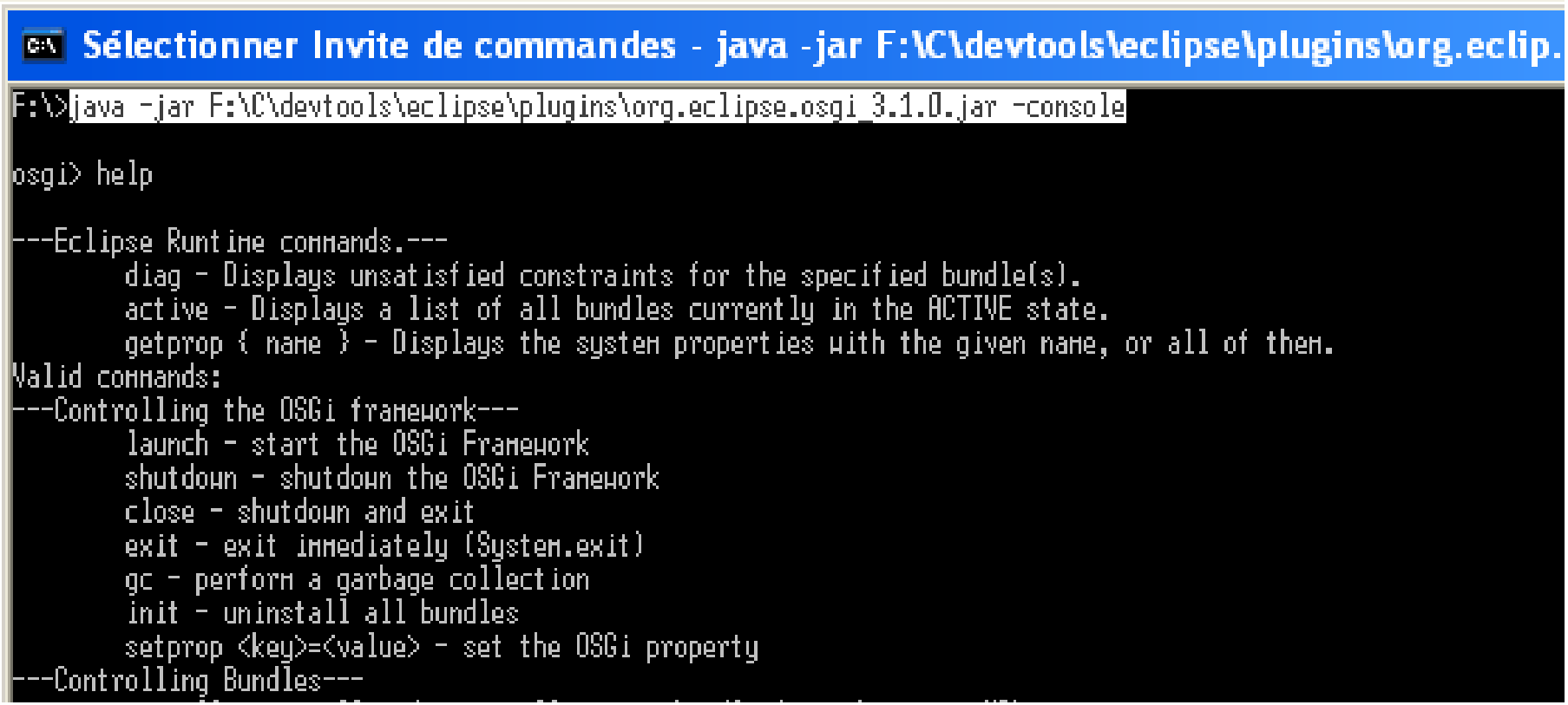
Merci à Corentin Baron

La console GUI



Eclipse/OSGi

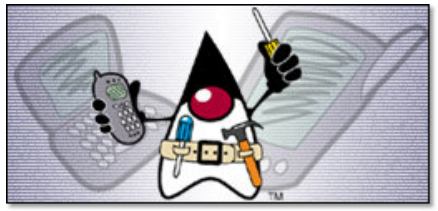
- **embarqué dans Eclipse/IDE et Eclipse/RCP**
  - **Pour le conditionnement et le déploiement des Plugins**
- **La console texte**
  - **java -jar %ECLIPSE_HOME%\plugins\org.eclipse.osgi_3.1.0.jar -console**



OSGi™ TCK

- **Suite de tests pour vérifier la compatibilité**
  - Du framework (Core)
  - Des services standards (Compendium)
  - Les tests concernent les fonctionnalités obligatoires et optionnelles (fragments, …)
- **Remarque**
  - Seulement accessible aux membres de l'OSGi Alliance

La concurrence : J2ME/CLDC/MIDP



- **Mobile Information Device Profile**
  - Modèle de développement et de déploiement d'applications (MIDLet) pour des Java phones
- **La lutte existe (même au sein des grands acteurs)**
- **Avantages de MIDP**
  - Simple
- **Avantages d'OSGi**
  - Applicable aussi à l'intergiciel du Java phone

La concurrence : *embedded*-Linux

- **Plate-forme**
  - Multi-application
  - Multi-fournisseurs (users)
  - Éventuellement Temps Réel
- **Solutions**
  - Chargement / Déchargement dynamique de .so
  - Processus (comme sandbox)
- **Avantages**
  - Très très répandu …
- **Inconvénient**
  - Coûts des échanges (IPC,socket) entre les « services »

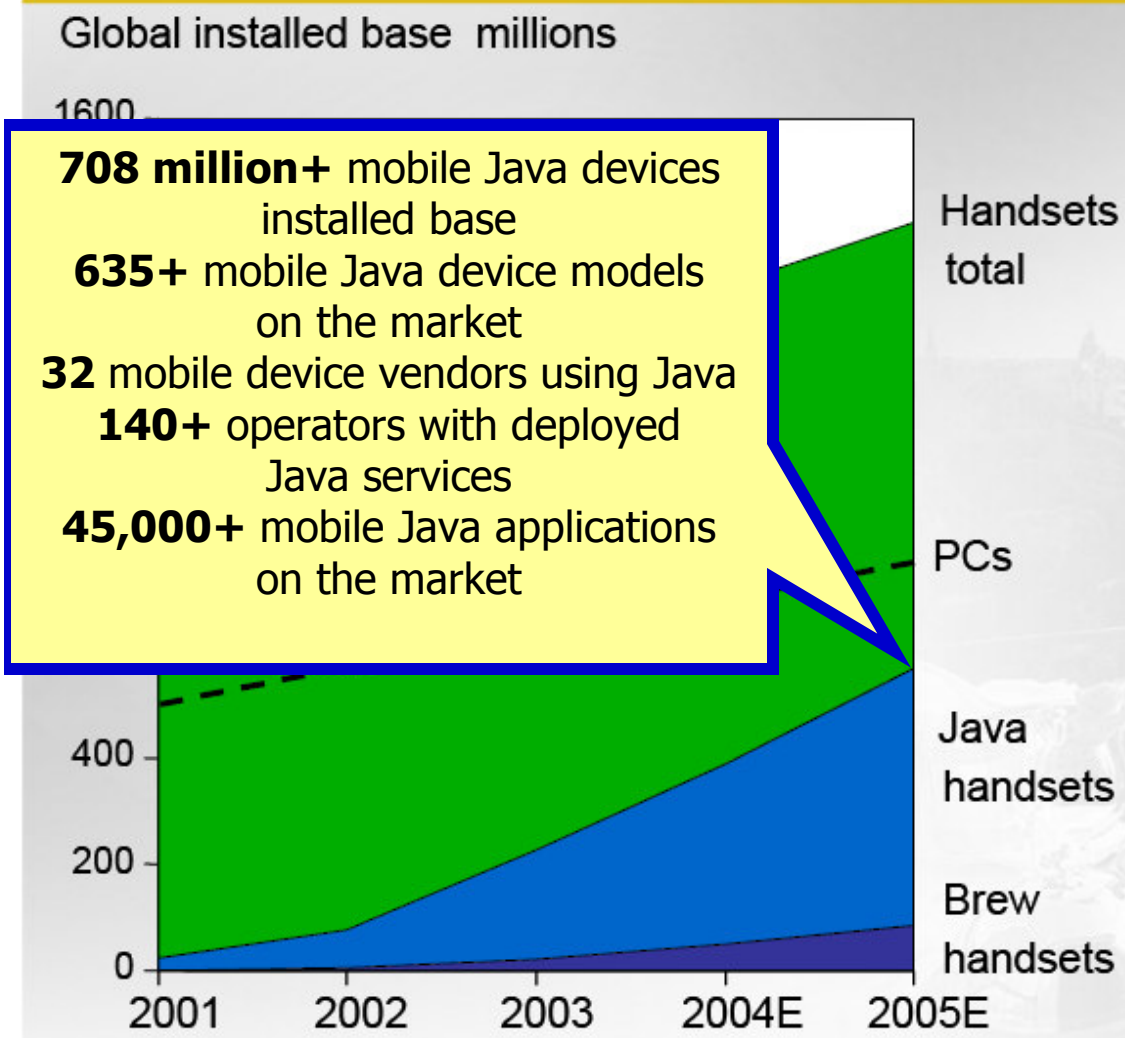
La concurrence : MicroSoft .NET

- .NET
  - Alternative à Java (et à machine virtuelle)
  - Mais des approches similaires
    - bytecode, JIT, chargeur de classes, …
  - et différentes
    - Multi-langage, cache de compilation, domaine d'application, …
  - Compact.NET alternative à J2ME/CDC
  - Très récente annonce de .NET Micro Framework
- Le problème de .NET (1 et 2)
  - Le déchargement d'une classe requière l'arrêt du domaine d'application.
  - Donc pas de mise à jour partielle d'une application
- Des compromis (non gratuits) restent possibles [Escoffier06] (peut être en attendant .NET 3.0)

La tendance

- **OSGi couvre désormais un spectre étendu de domaine**
  - Passerelle résidentiel
  - Passerelle véhiculaire
  - Passerelle industrielle
  - Téléphonie mobile
  - Application sur poste de travail (Eclipse RCP)
  - Enterprise Side
    - Serveur IT (J2EE, …), Web Framework, …
    - JOnAS, Geronimo, ApacheDS, JAMES, …
    - ECP
  - Enterprise Expert Group à l'OSGi Alliance
    - Workshop sur OSGi + J2EE (11/09/2006, San José)

Java is the leading mobile Application Development Environment



708 million+ mobile Java devices installed base
635+ mobile Java device models on the market
32 mobile device vendors using Java
140+ operators with deployed Java services
45,000+ mobile Java applications on the market

Key message in 2002 JavaOne:

"We will put Java in every pocket"

...done.

Key message in 2005 JavaOne:

"We will put Java server in every pocket"

...working on it...

OSGi Alliance

Les perspectives

- Open-source
  - Match entre Eclipse et Apache ?
- JSR 277 Java™ Module System
  - ~ couvert par OSGi R4 Module Layer
  - Prévu pour Java Platform 7.0
- JSR 291 Dynamic Component Support for Java™ SE
  - ~ couvert par OSGi R4 SCR
  - Prévu pour Java Platform 7.0
- Quel nouveau rôle pour OSGi™ ?
  - Définition de Services Standards
- Quand même « une crainte pour les « penseurs » de l'Alliance »
  - Faire venir les développeurs de logiciels embarqués à J2ME et à OSGi …

Pour terminer
Un peu de publicité

- **OSGi™ Users' Group France**
  - Association d'utilisateurs de la technologie OSGi™
    - Développeurs, consultants, enseignants, chercheurs, …
    - de FT R&D, EDF R&D, Schneider Electric, Trialog, Siemens VDO, Gemplus, Alcatel, Bull, Thomson, Scalagent …
    - et de l'INRIA, CNRS, …
  - 4 réunions depuis Décembre 2004
  - 5ème réunion : 5 Septembre 2006 au CNAM (Paris)

- **En savoir plus**
  - http://www-adele.imag.fr/osgi
  - *http://www.osgiusers-france.org*

Conclusion finale

+ Très fort potentiel

Ne passez pas à coté

Q&R

Bibliographie & Webographie

Spécification

- Open Services Gateway Initiative, « OSGi service gateway specification », http://www.osgi.org

Ouvrage (1 seul pour le moment)

- Kirk Chen, Li Gong, « Programming Open Service Gateways with Java Embedded Server Technology », Pub. Addison Wesley, August 2001 ISBN#: 0201711028. 480 pages

Articles

- Li Gong, « A Software Architecture for Open Service Gateways », IEEE Internet Computing, January/February 2001 (Vol. 5, No. 1), pp. 64-70
- Dave Marples, Peter Kriens, The Open Services Gateway Initiative, an Introductory Overview, IEEE Communications Magazine, December 2001

Bibliographie & Webographie

- **Framework open source**
  - Oscar, Felix, Equinox, Knopperfish
- **Index de bundles**
  - http://bundles.osgi.org/browse.php
- **Exhibitions**
  - http://www.osgiworldcongress.com/
- **Blog de Peter Kriens**
  - http://www.osgi.org/blog/
- **Complément de cours**
  - Donsez, Hall, Cervantes http://www-adele.imag.fr/users/Didier.Donsez/cours/osgi.pdf
  - Frénot http://citi.insa-lyon.fr/~sfrenot/cours/OSGi/
  - INTech http://rev.inrialpes.fr/intech/Registration?op=511&meeting=27

A cette après midi

- Le contenu de l'atelier pratique
  - Installation d'Apache Felix
  - Premières commandes via différentes consoles
  - Déploiement de bundles
  - Développement d'un servant
  - Développement d'un composant SCR
    - Utilisant Event Admin Service
    - Utilisant Wire Admin Service
    - Utilisant Http Service Service
  - Démonstration de l'UPnP Base Driver
  - Démonstration d'administration de passerelle avec JMX

http://www.nslu2-linux.org

ICAR'06

École d'été sur les Intergiciels et sur la Construction d'Applications Réparties

Bonus Track

JSR 294: Improved Modularity Support in the Java™Programming Language

- **D'aprés Rick Hall**
- **Rational**
  - Java programming language needs better support for hierarchical, modular organization
    - Primarily to support information hiding
    - Java packages inadequate for this purpose
    - Only two levels of visibility: internal to the package or public to everyone
- Module "Files"

```
super package com.foo.moduleA {
    // Exported classes/interfaces
    export com.foo.moduleA.api.*;
    export com.foo.moduleA.ifc.InterfaceC;
    // Imported modules
    import org.bar.moduleD;
    // Module membership
    com.foo.moduleA.api;
    com.foo.moduleA.ifc;
    org.apache.stuff;
}
```