La plate-forme dynamique de service OSGi™

Didier Donsez

Université Joseph Fourier (Grenoble 1)

PolyTech'Grenoble LIG/ADELE

Firstname.Lastname@imag.fr

Firstname.Lastname@ieee.org

http://membres-liglab.imag.fr/donsez/cours/osgi.pdf

Sommaire

- Motivations et Rappels
- Conditionnement et Service
- Enregistrement et recherche de services
- Composants
- Services standards (survol)
- 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)

- 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: Octobre 2005 (1000 pages)
- 4.1: Juin 2007 (optimisation du

OPEN SERVICES GATEWAY INITIATIVE (OSGi) TO DRIVE DEVELOPMENT OF GATEWAY STANDARD FOR HOMES SOHO AND REMOTE LOCATIONS
Sun's Java™ Technology Accelerates Development of Specification

PALO ALTO, Calif., - November 22, 1999 - Open Services Gateway Initiative (OSGi) and Sun Microsystems, Inc. announc today that Sun has transferred the effort to define an open gateway specification from the Java™ Community Process to th Open Services Gateway Initiative. …

Remarque

- *Open Services Gateway Initiative est un terme obsolète*

L'ancêtre :
JSR-8 : Open Services Gateway (OSG)

- Java Embedded Server
  - JavaOne e-Fridge
- Domain : SOHO / ROBO Gateway
- EG
  - Spec leader : Robert Mines (Sun)
  - Sun Microsystems, IBM, Nortel, Alcatel, Cable and Wireless, EDF, Enron, Ericsson, Lucent, Motorola, NCI, Phillips, Sybase, Toshiba
- Package names
  - javax.osg.servicespace
  - javax.osg.remote
  - javax.osg.service
- Transfered to the OSGi Alliance

Principales propriétés du canevas OSGi

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

Rappel sur les chargeurs de classes

- **java.lang.ClassLoader**
  - Objet (java) chargé de charger en mémoire la définition des classes (`.class`)
- **Principe de la délégation**
  - Tout chargeur a un chargeur parent
    - sauf le chargeur primordial
  - Tout chargeur vérifie si la classe à charger n'a pas déjà été chargée par un chargeur parent
- **Arbre de délégation basique**
  - ClassLoader bootstrap ou primordial
  - `sun.misc.Launcher$ExtClassLoader` *(extension)*
  - `sun.misc.Launcher$AppClassLoader` *(application ou system)*

  → Possibilité de personnaliser les chargeurs

Rappel sur les chargeurs de classes
Pourquoi utiliser les chargeurs de classes

- Classes non présentes dans le CLASSPATH ou le $JAVA_HOME/lib/ext
  - URLClassLoader, AppletClassLoader, RMIClassLoader…
    - ex: WEB-INF/classes et WEB-INF/lib d'une WebApp
    - ex: CODEBASE d'une applet, …
- Emballage particulier
  - JavaEE EAR, OSGi bundle (fichiers JAR imbriqués), java.util.jar.Pack200, Google Android DEX format …
- Modification du ByteCode à la volée au chargement
  - Instrumentation
    - AOP (Aspect Oriented Programming)
    - BCEL, ASM, …
  - Protection
- Chargement de ressources associées à la classe
  - properties, images, …
- Déchargement et Mise à jour du bytecode lors de l'exécution de la VM (runtime)
  - Chargeurs de OSGi

Rappel sur les chargeurs de classes
Pourquoi NE PAS utiliser les chargeurs de classes

- Beaucoup trop complexe pour le commun des mortels (et également pour les autres)
- Indispensable de comprendre le fonctionnement !
- Car malheureusement beaucoup bricolent avec !

Rappel sur la programmation OO
« programming in the small »

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

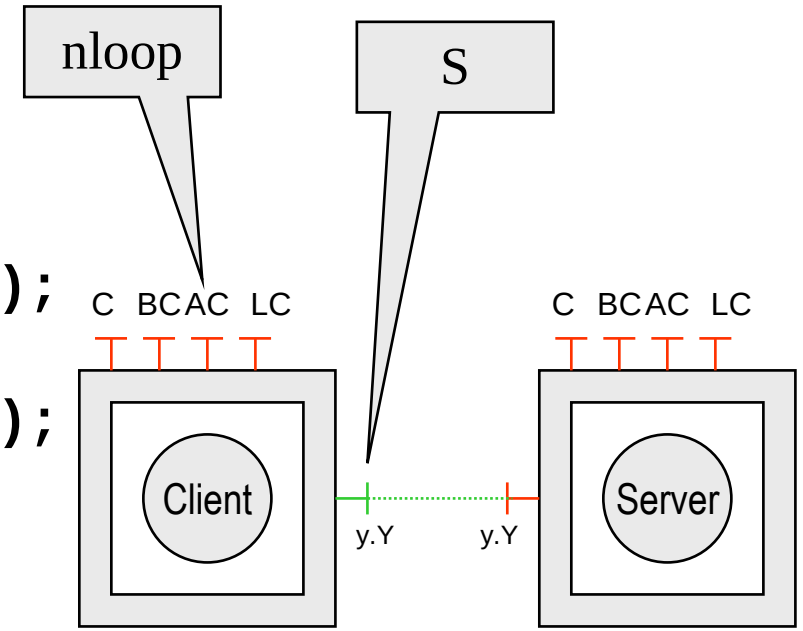
- `S s=new S()`
- `C c1=new C(s, N);`
- `C c2=new C(s, N);`

- **Problème: Architecture ? Configuration ?**

Rappel sur la programmation Composant
« programming in the large »

- **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();
…
```



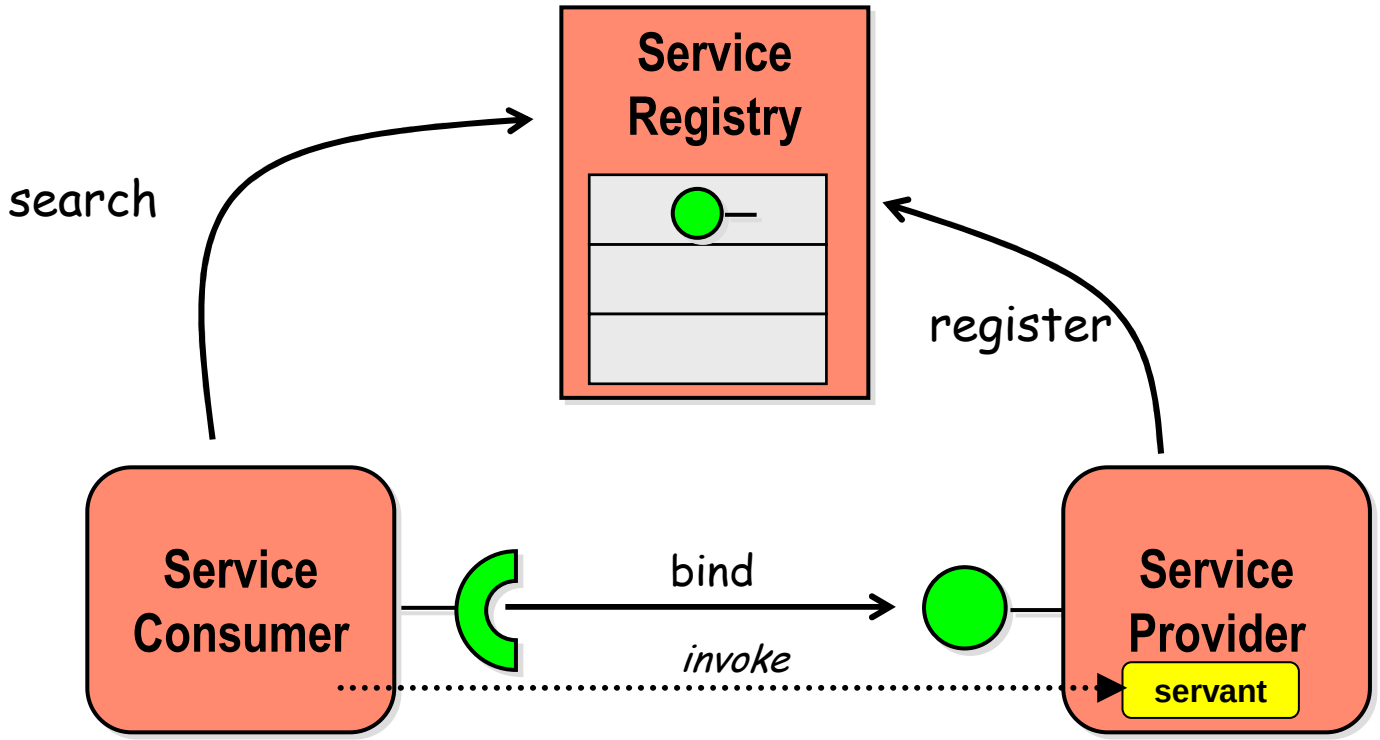
Rappel sur la programmation Composant

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

- **Problème: Multi-domaines d'administration**
  - **carte GSM SIM, WS, iTV STB, passerelle domotique, …**

Rappel:
Architecture orienté service (SOA)
« programming in the VERY large »

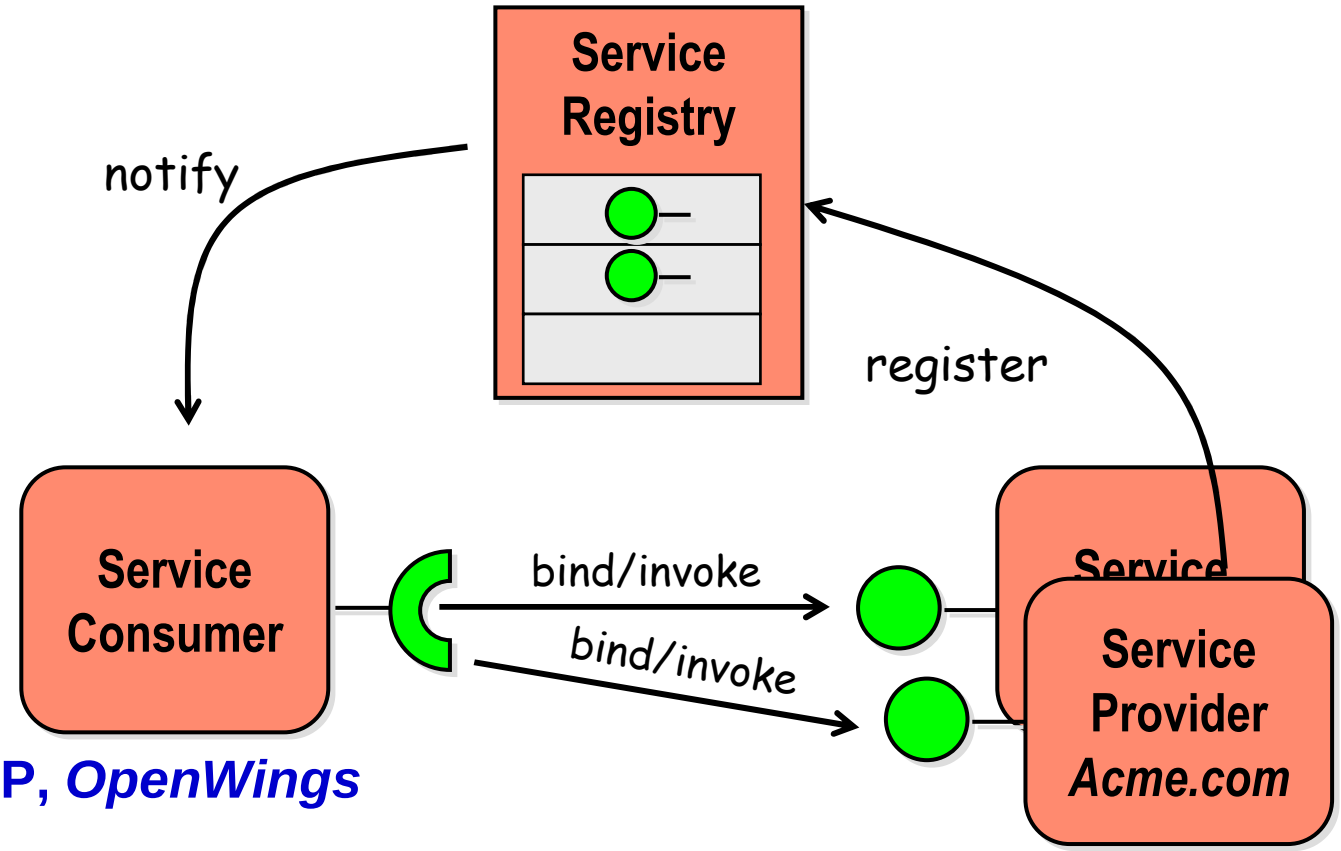
- Les services (contrats) sont « invariants »



- WebServices, TORBA, …

Rappel:
SOA Dynamique

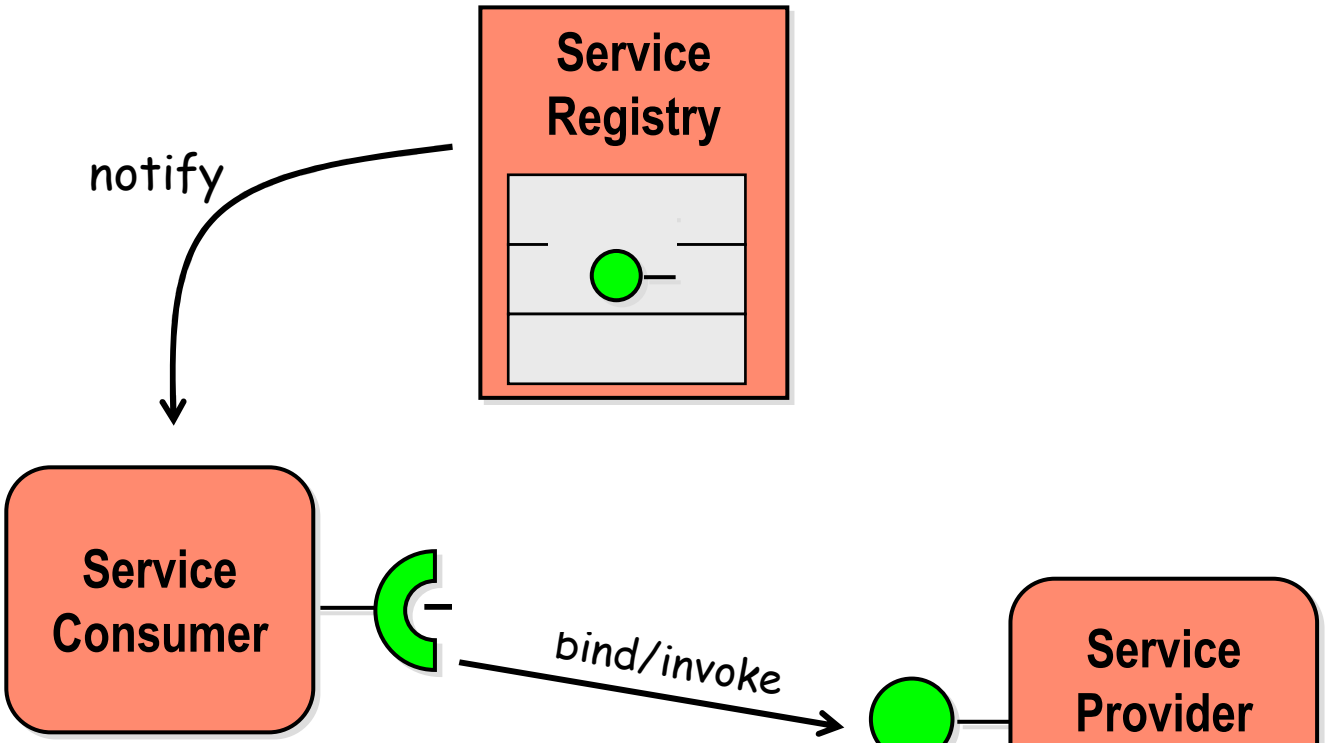
- Arrivée dynamique de nouveaux services



- JINI, UPnP, *OpenWings*
- OSGi

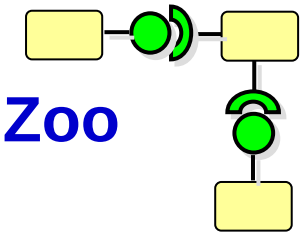
Rappel:
SOA Dynamique

- 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

OSGi

Modèle d’administration
et Domaines d’application

Domaines d'application

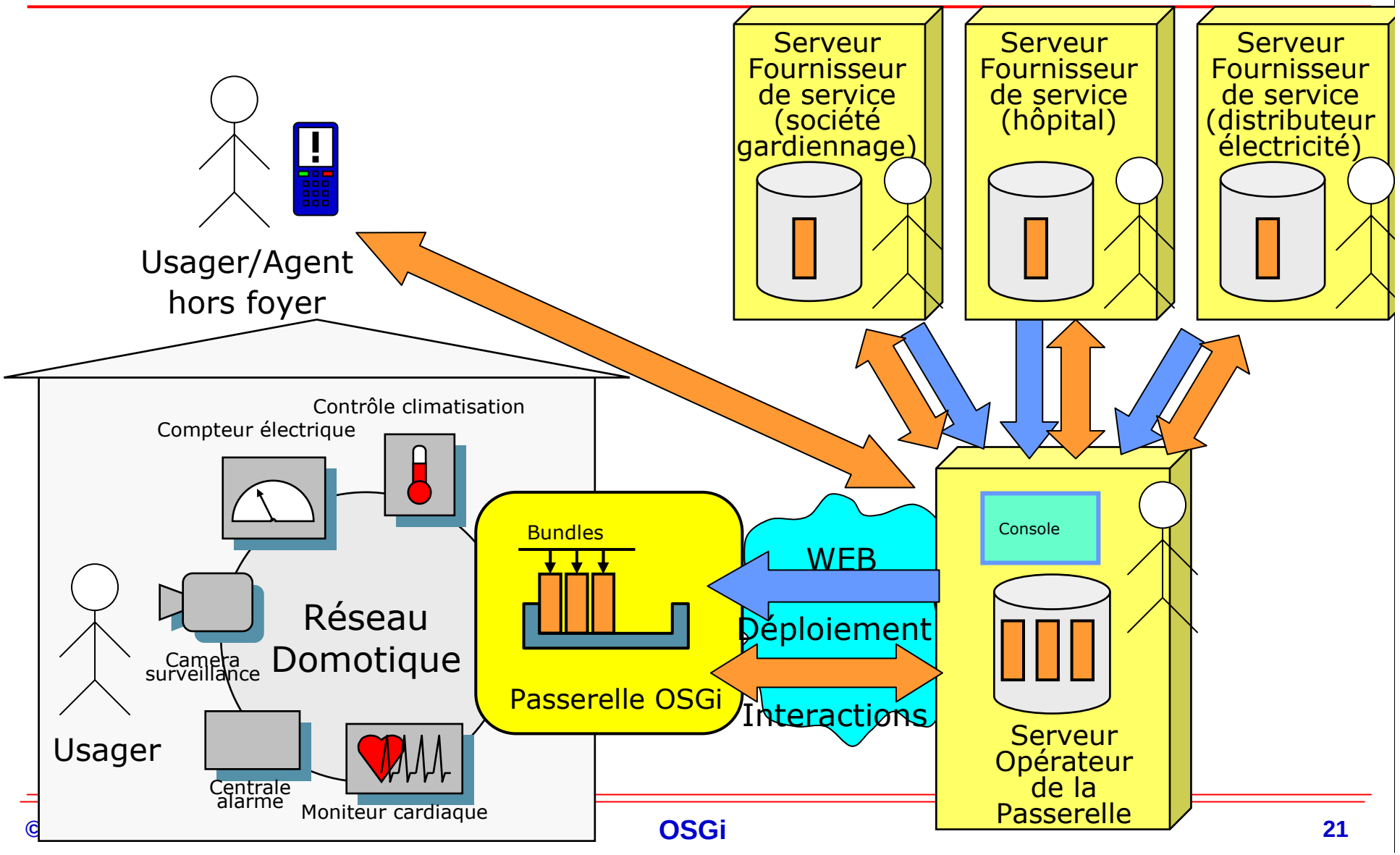
- **Initialement, Systèmes embarqués**
  - Véhicule de transport (*automotive*)
  - Passerelle résidentiel/domotique/immotique
  - Contrôle industriel
  - Téléphonie mobile

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

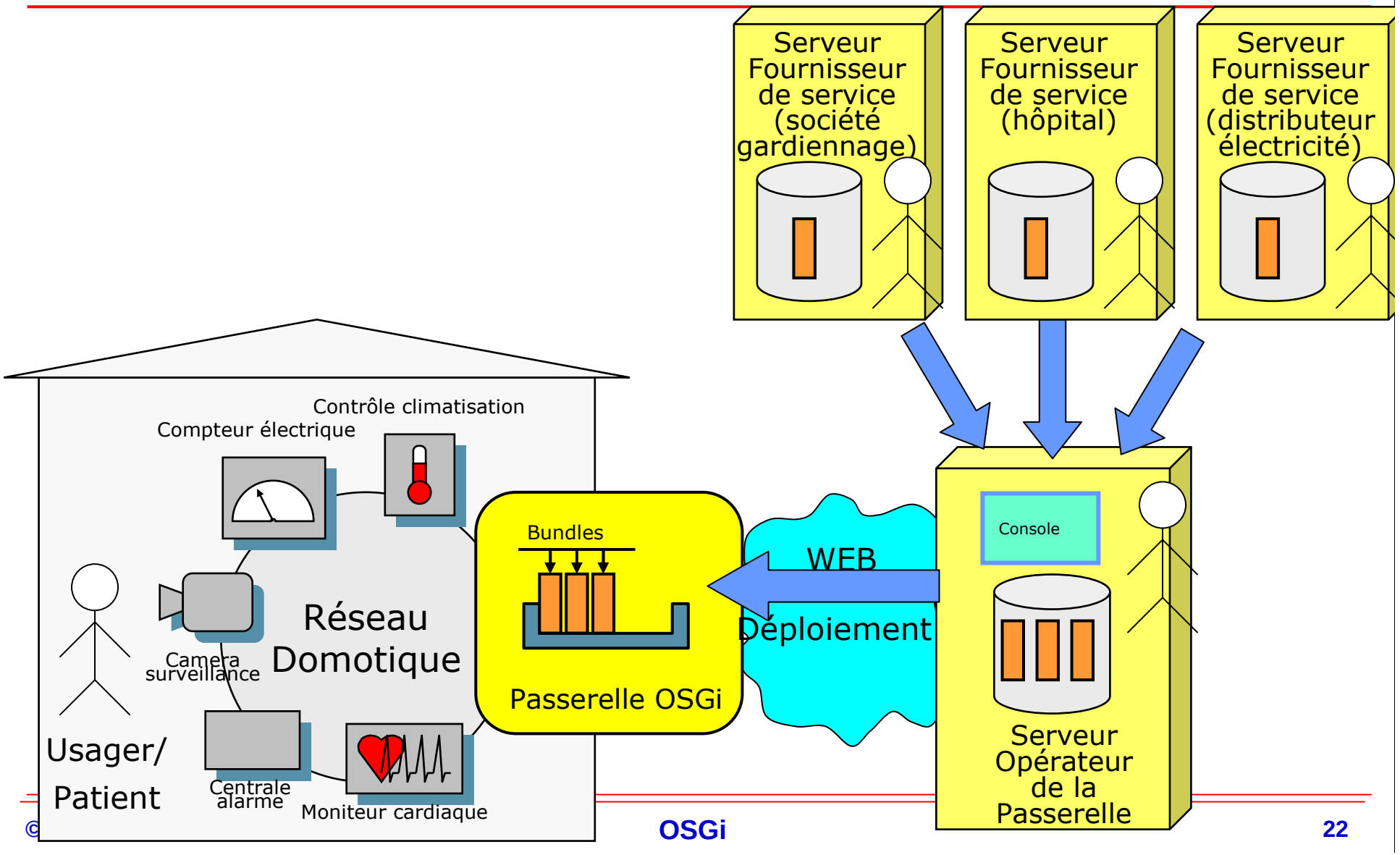
  - Cela évite l'enfer du CLASSPATH
    - CLASSPATH, lib/ext du JRE ou JavaEE, …

- **Maintenant**
  - Eclipse RCP, JavaEE, *Harmony JRE pieces, …*

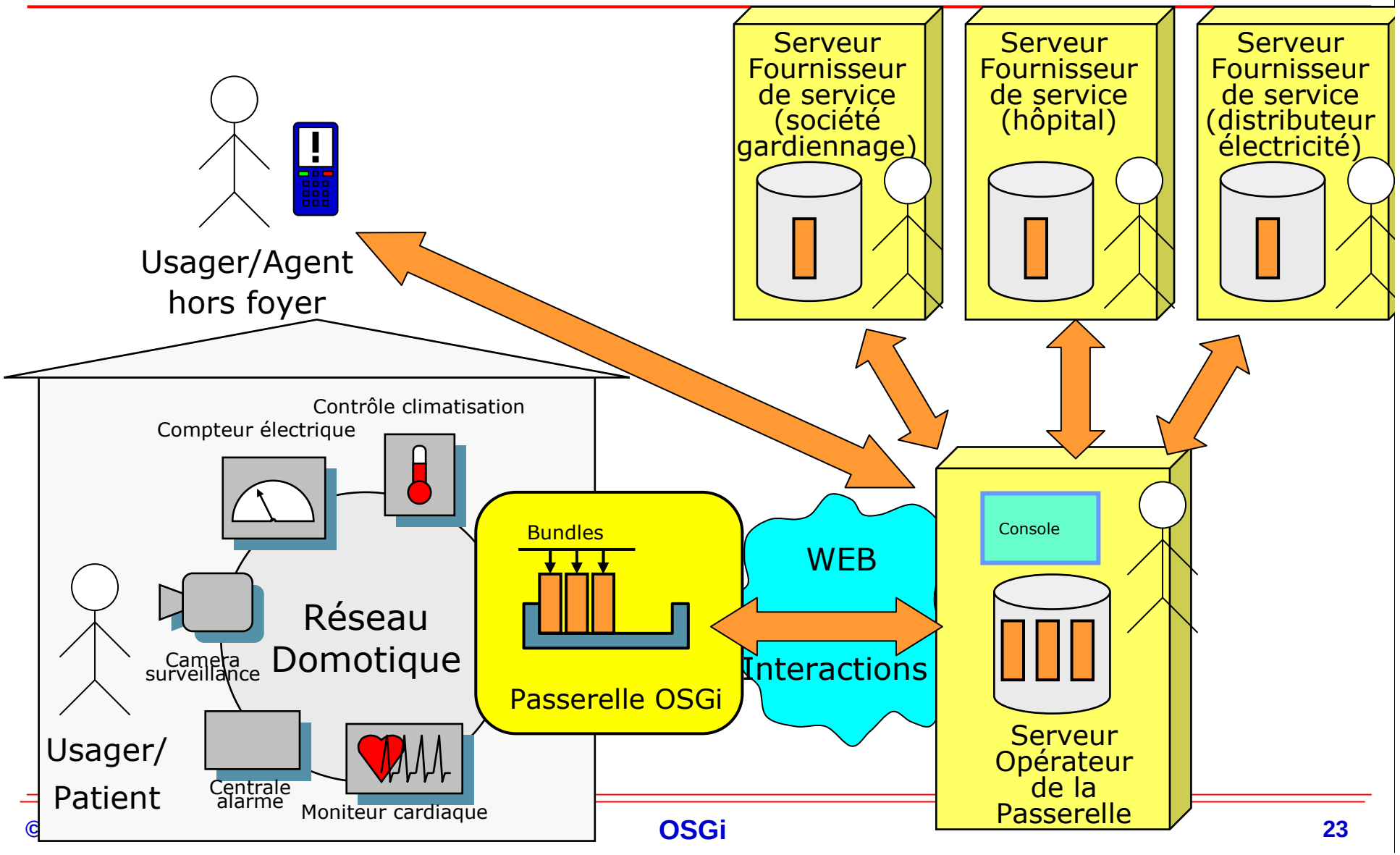
Architecture générale



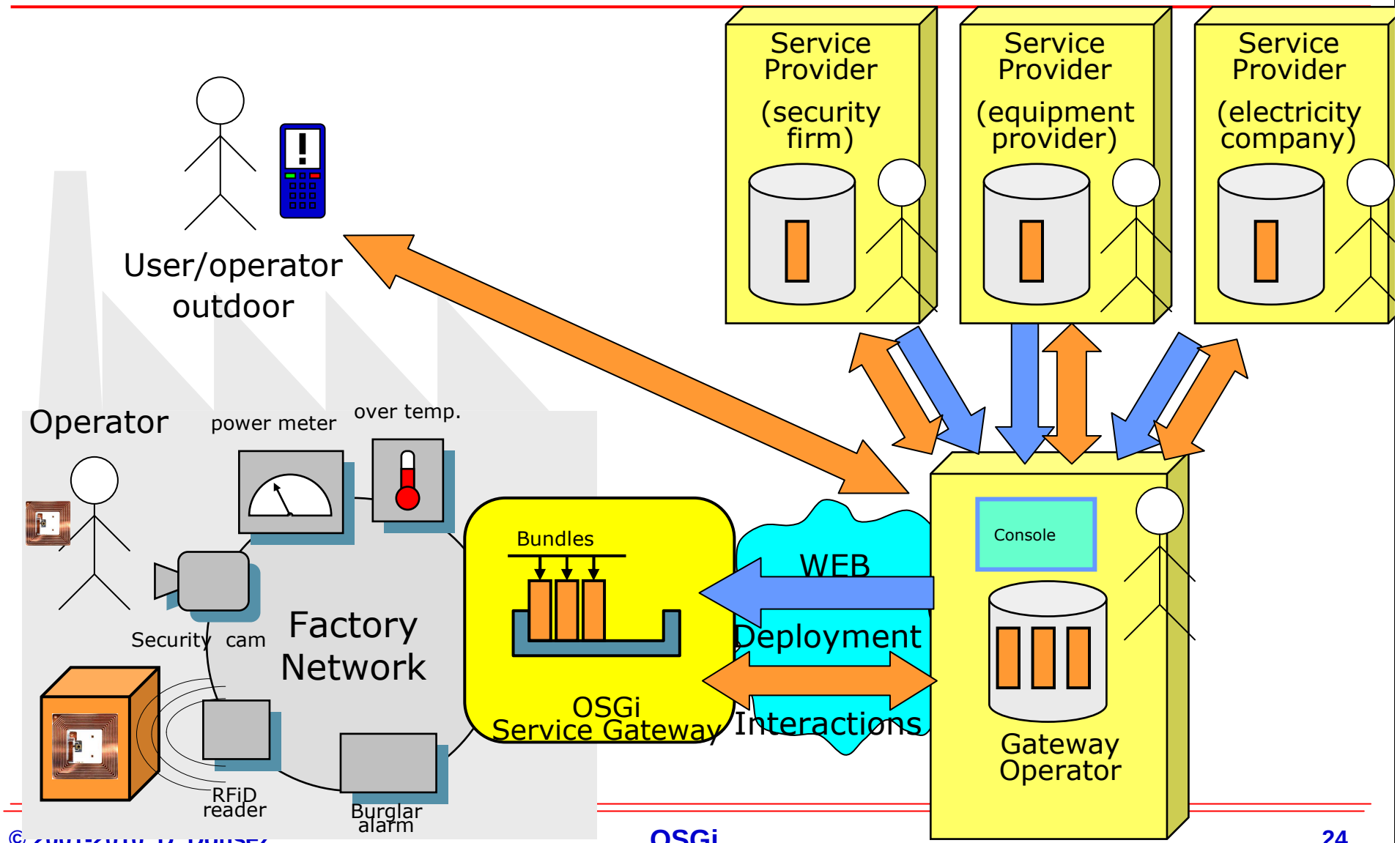
Architecture générale (i)
Déploiement

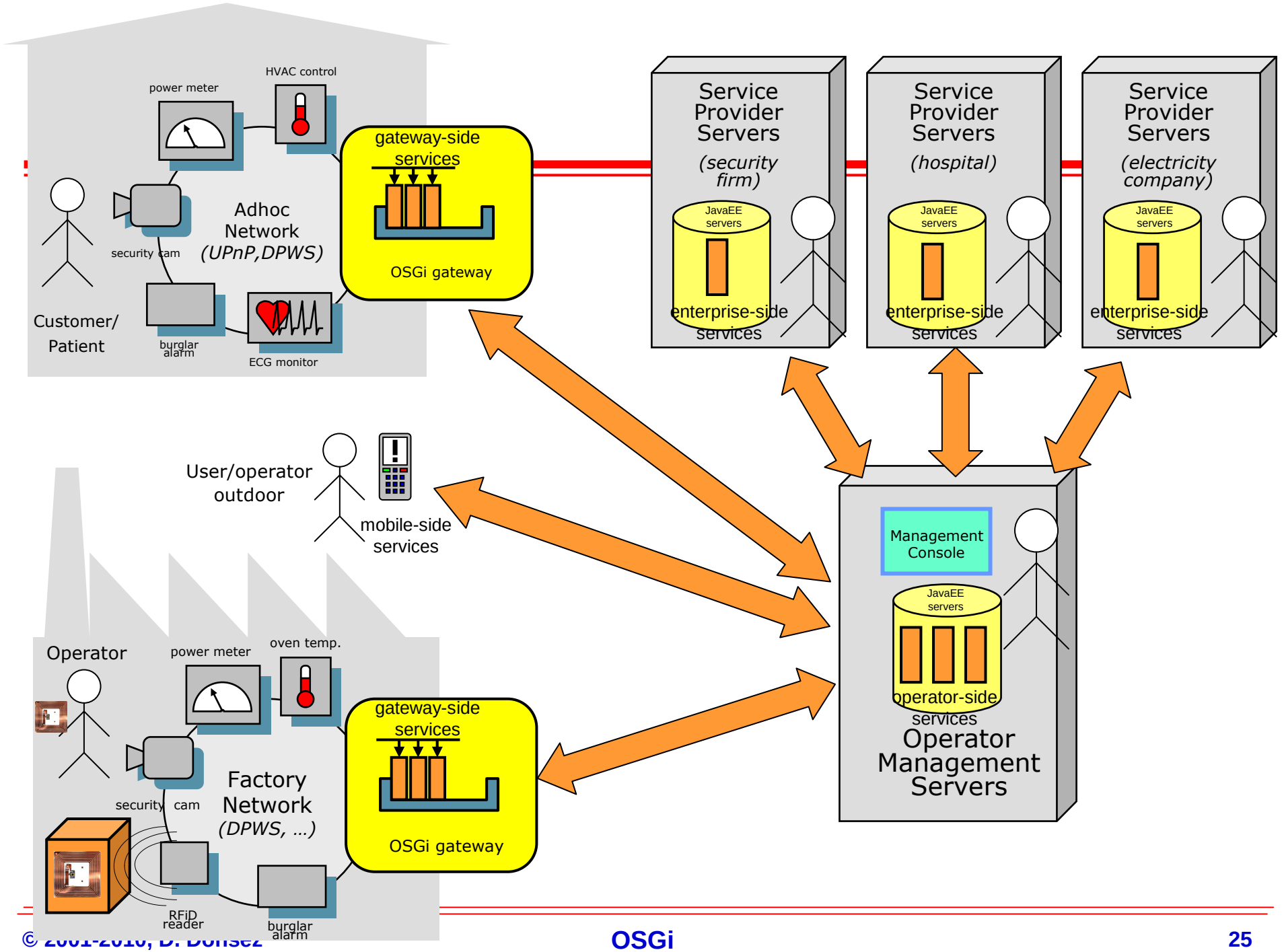


Architecture générale (ii)
Interactions

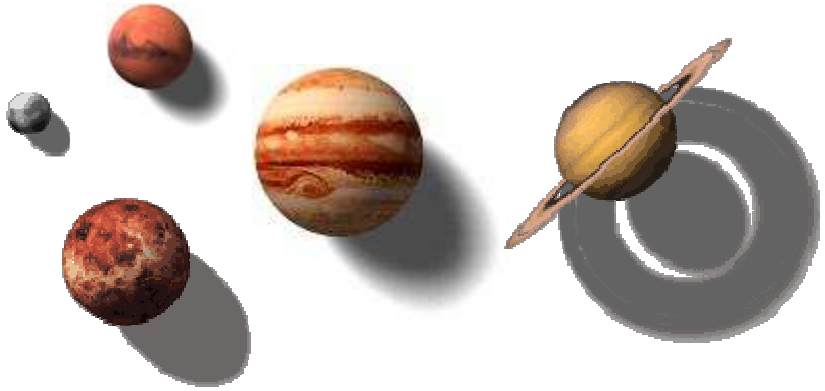


Même architecture générale (iii)
Contexte différent

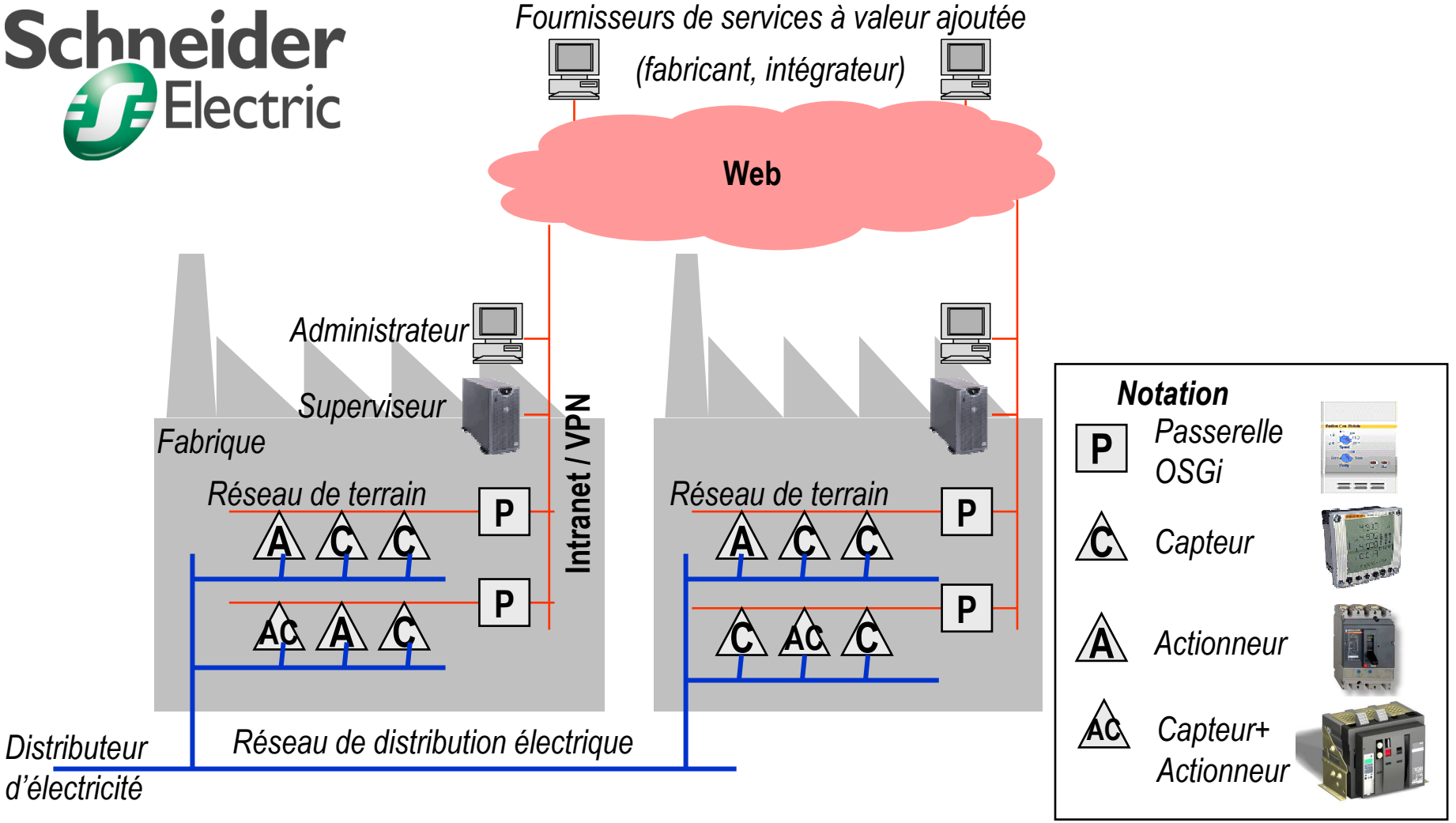



power meter
HVAC control
gateway-side services
Adhoc Network (UPnP,DPWS)
security cam
OSGi gateway
Customer/ Patient
burglar alarm
ECG monitor
Service Provider Servers (security firm)
Service Provider Servers (hospital)
Service Provider Servers (electricity company)
JavaEE servers
enterprise-side services
User/operator outdoor
mobile-side services
Management Console
operator-side services
Operator Management Servers
Operator
power meter
oven temp.
gateway-side services
Factory Network (DPWS, ...)
security cam
OSGi gateway
RFiD reader
burglar alarm

Même architecture générale (iv)
Contexte différent

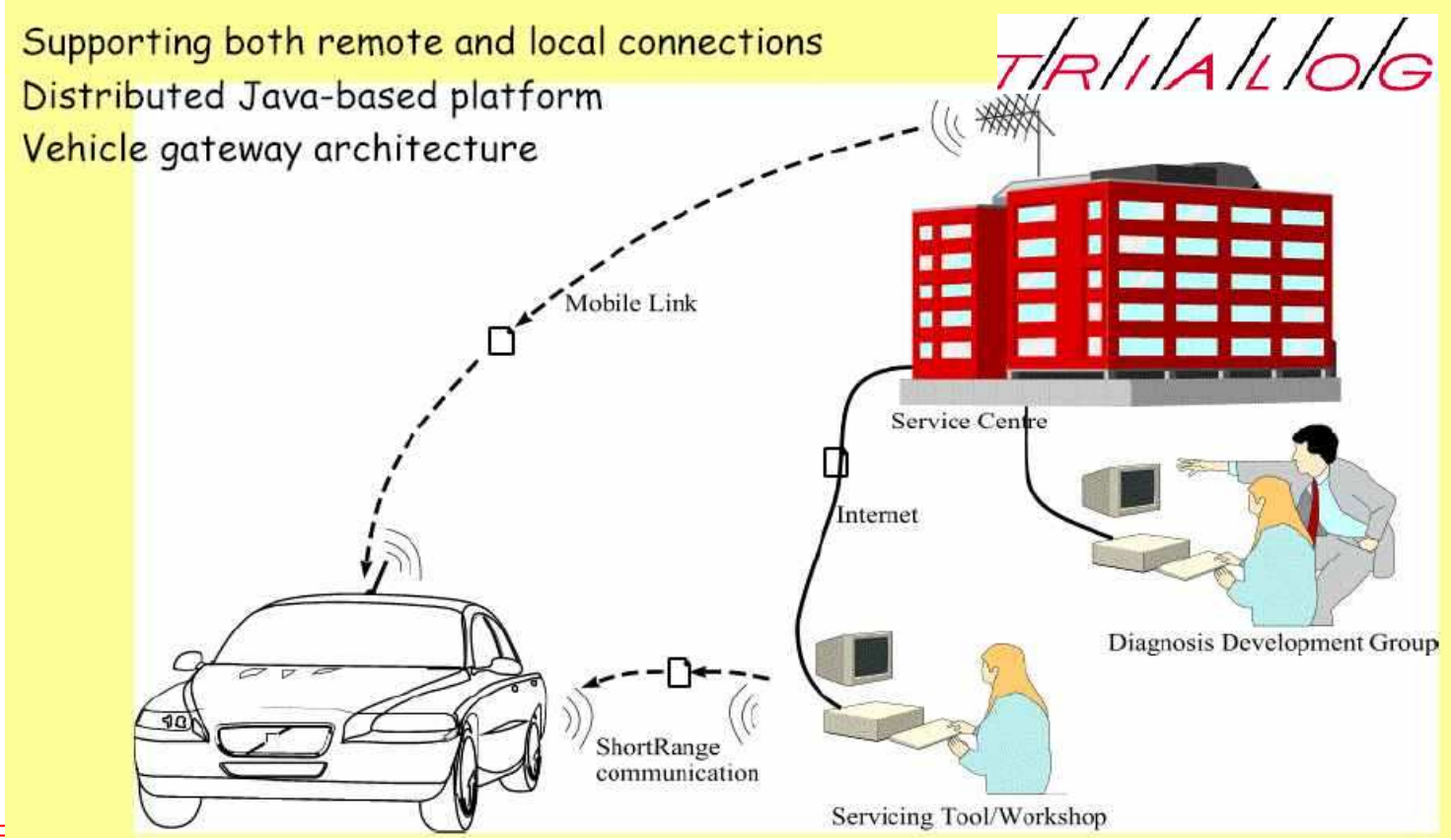


Application à la Distribution Electrique chez Schneider Electric



Diagnostic de véhicules à distance

- **Aujourd'hui la part de l'électronique dans la conception d'un véhicule est de 35%.**
  - **90% des innovations se font dans le domaine électronique**
- **Cependant 50% des pannes sont provoquées par des défaillances dans ce domaine.**



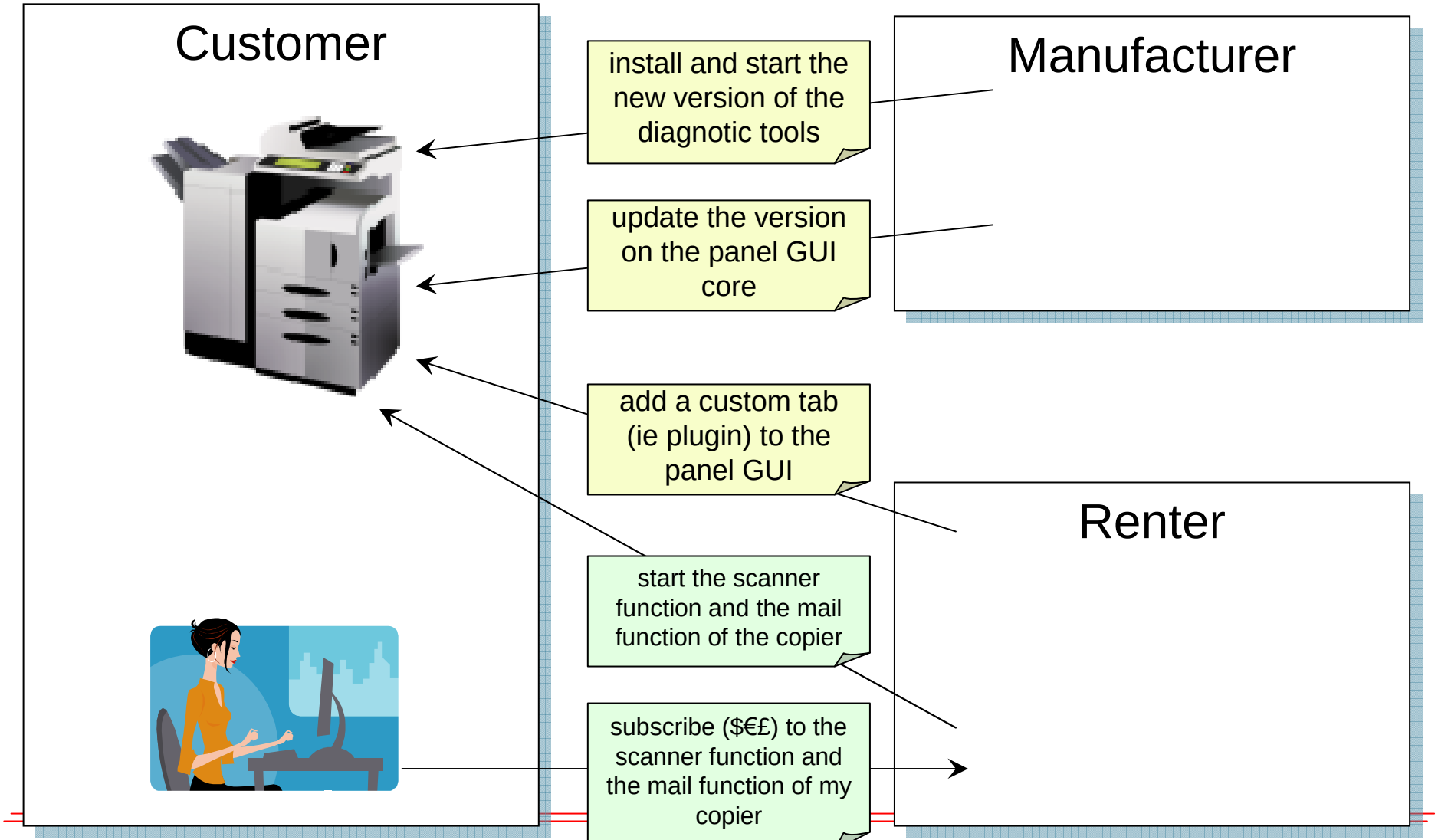
Personal gateway

- Human
  - Cardiac patient, Baby, Prisoner, Soldier on battle field
- Gateway (cell phone, watch, bracelet …)
- Between
  - wearable Sensors
    - ECG, Sphygmomanometer, GPS/GSM, …
  - and wearable actuators
    - PaceMaker, Heart Defibrillator, Tourniquet (garotte),
- and providers
  - Emergency team, Parents, Prison service, Army , …

Exemple de Scénario
Le photocopieur en location

- Le photocopieur est installé en location chez le client
- Le loueur (est une organisme financier) facture mensuellement à la consommation
  - Fixe mensuel + tarif par feuille
- Le loueur sous-traite la maintenance simple à une société spécialisée
- La société de maintenance réalise un diagnostic à distance avant d'envoyer un agent
- L'agent de maintenance interroge sur place le logiciel de diagnostic
- Le fabricant peut mettre à jour le logiciel embarqué
  - RICOH (26% copier market share) inclut une passerelle OSGi dans ses photocopieurs (en 2006).
    - http://www2.osgi.org/wiki/uploads/Conference/OSGiCommunityBushnaq.pdf

Exemple de Scénario
Le photocopieur en location



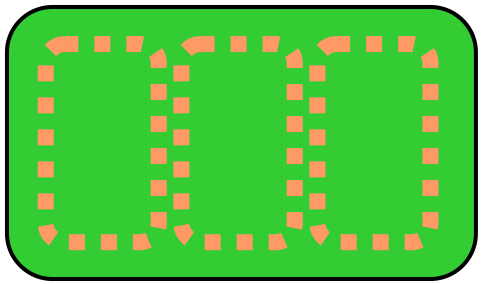
Le M2M (Machine-to-Machine)

OSGi

Conditionnement, Déploiement et Service

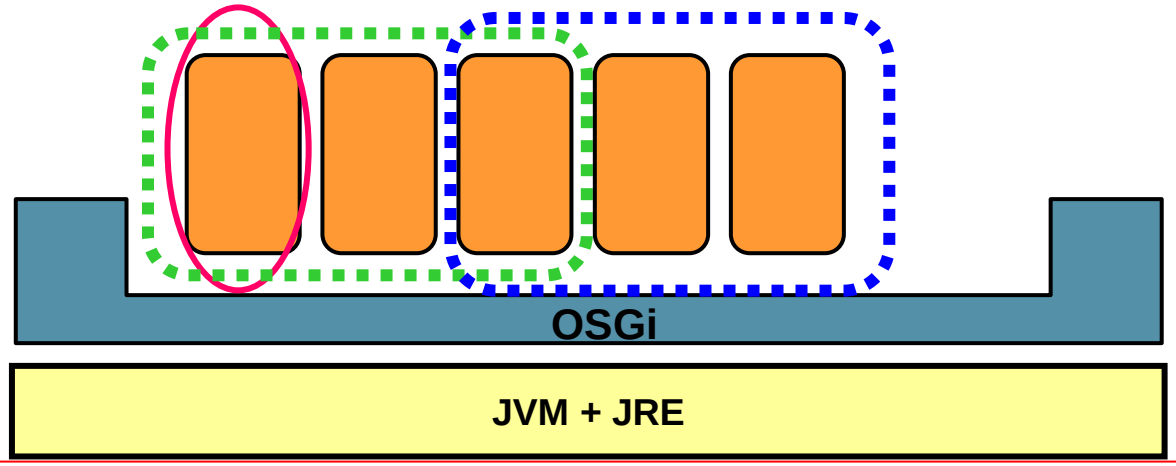
Rappel :
Une application Java non modulaire

- **Ensemble de jarfiles**
  - **placés statiquement dans le CLASSPATH ou $JRE_HOME/lib/ext**

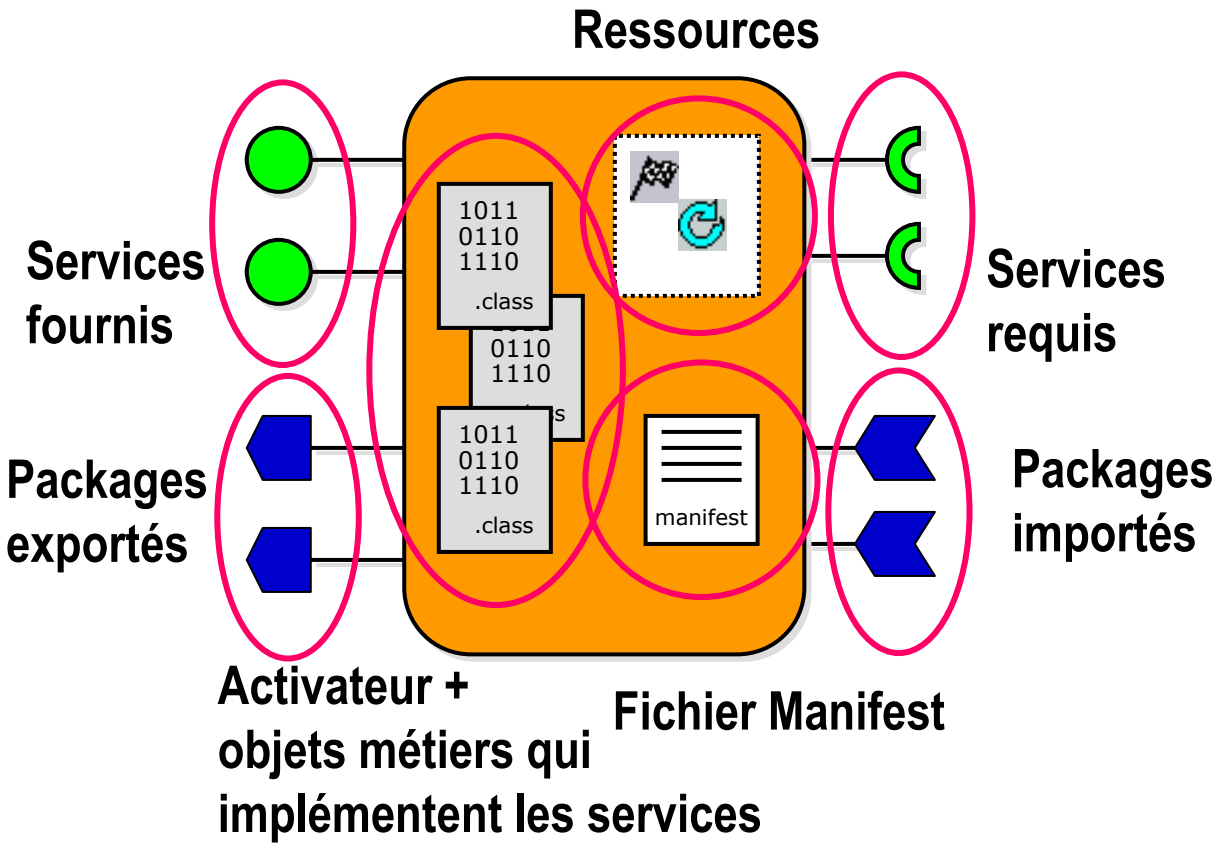


Bundle

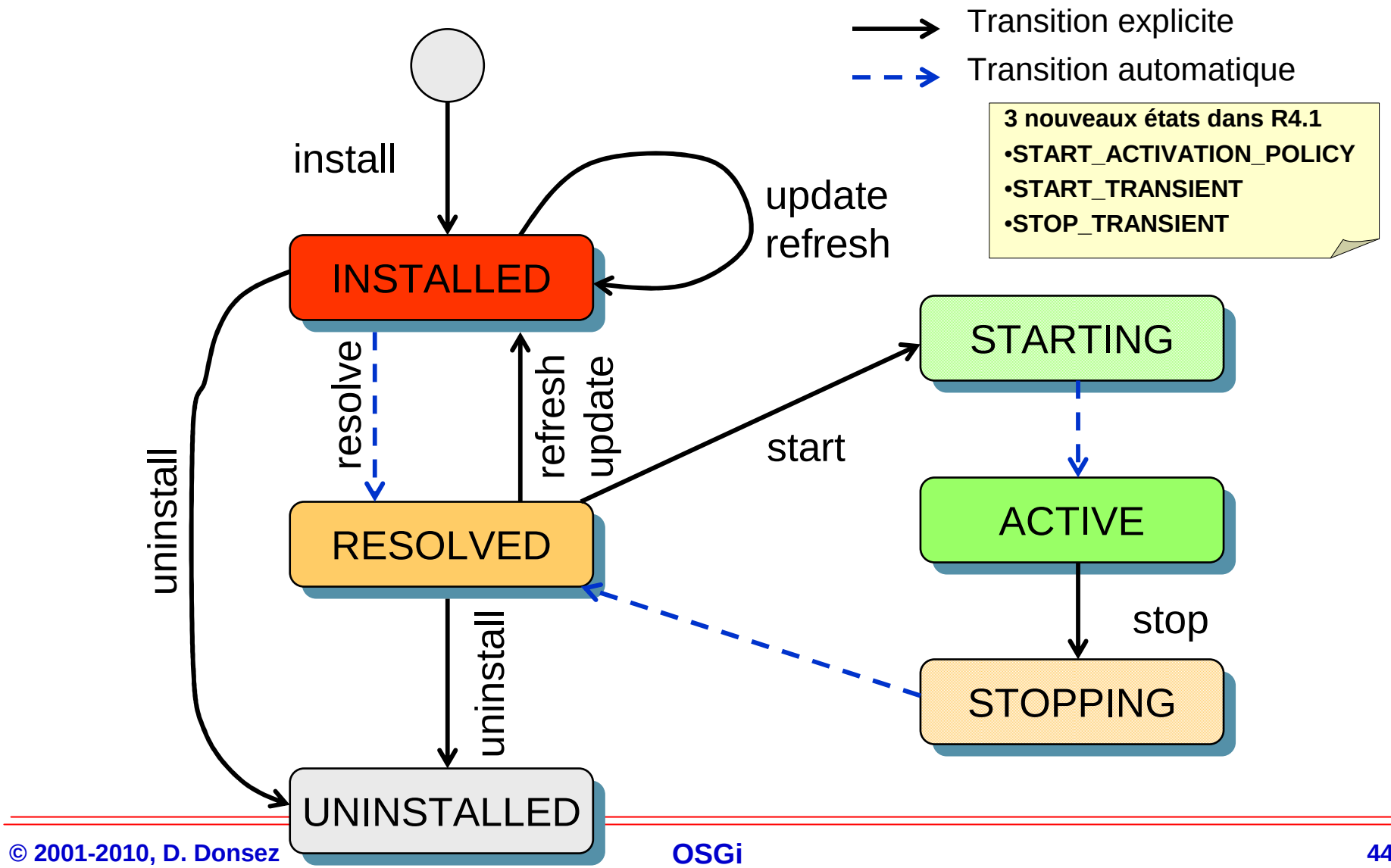
- **Bundle**
  - Unité de livraison et de déploiement sous forme de jarfile
  - Unité fonctionnelle (offre des services)
- **Application**
  - « Ensemble » de bundles
    - livrés dynamiquement
    - et éventuellement partagés avec d'autres applications



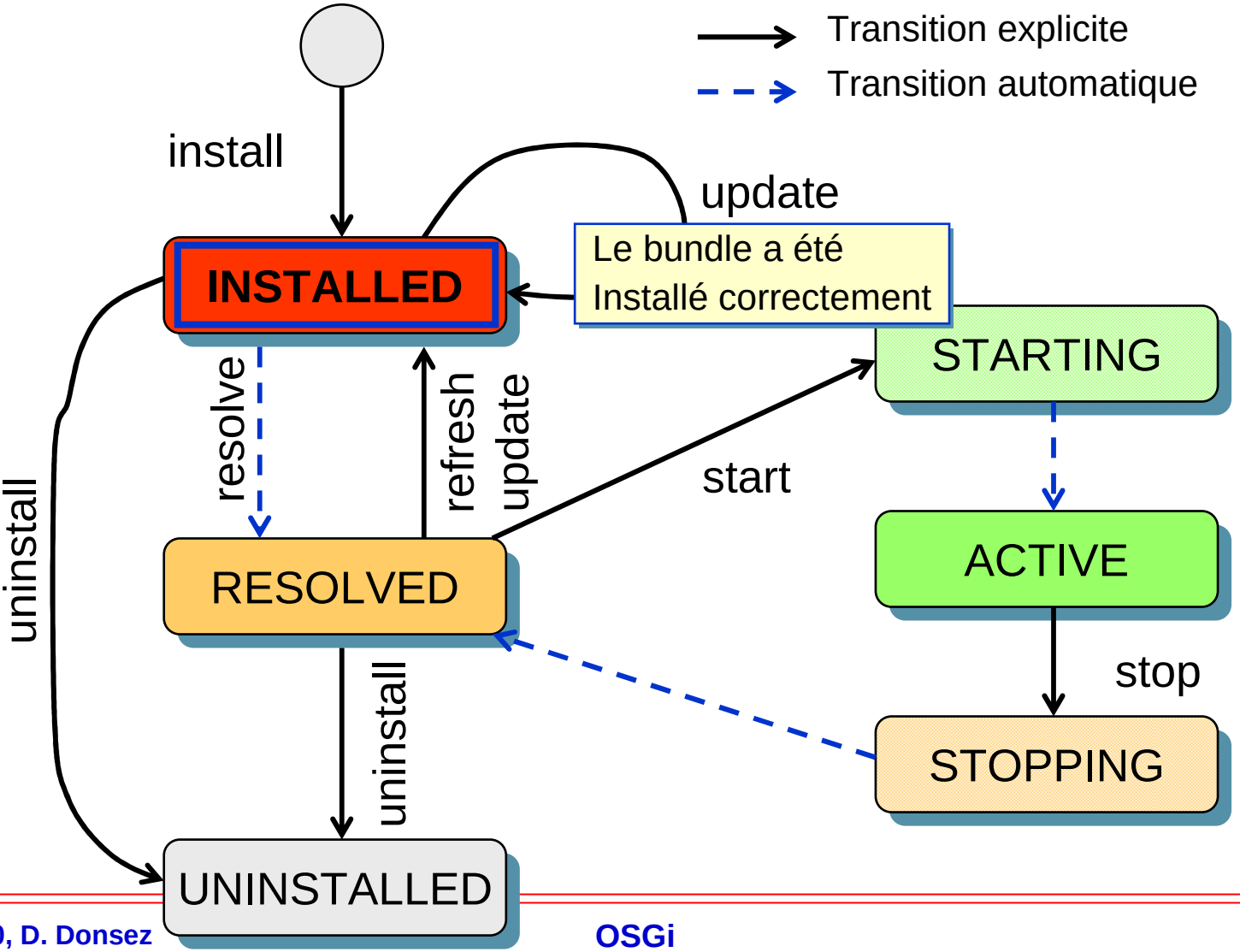
Structure d'un bundle



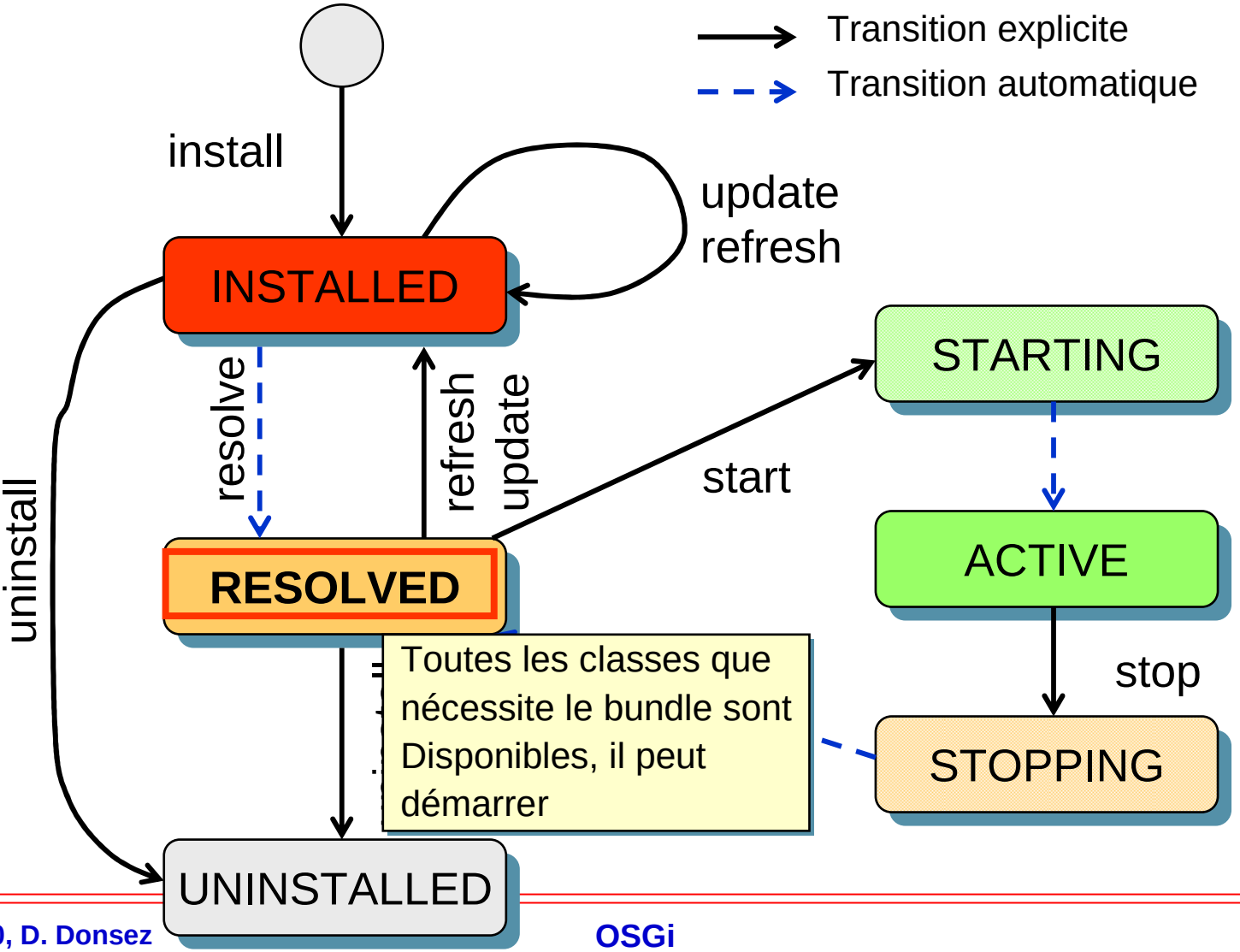
Cycle de vie d'un Bundle (R4)



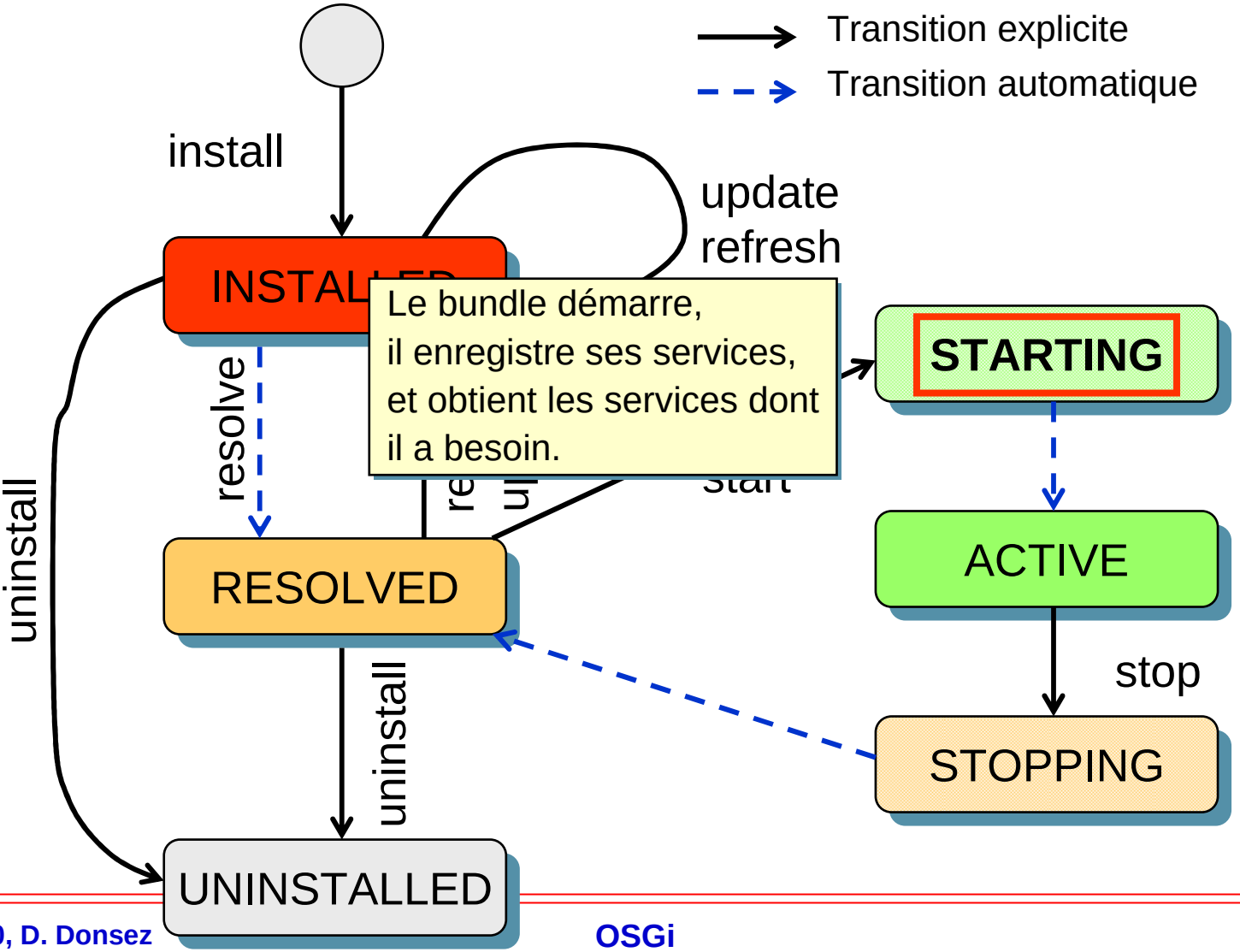
Cycle de vie d'un Bundle



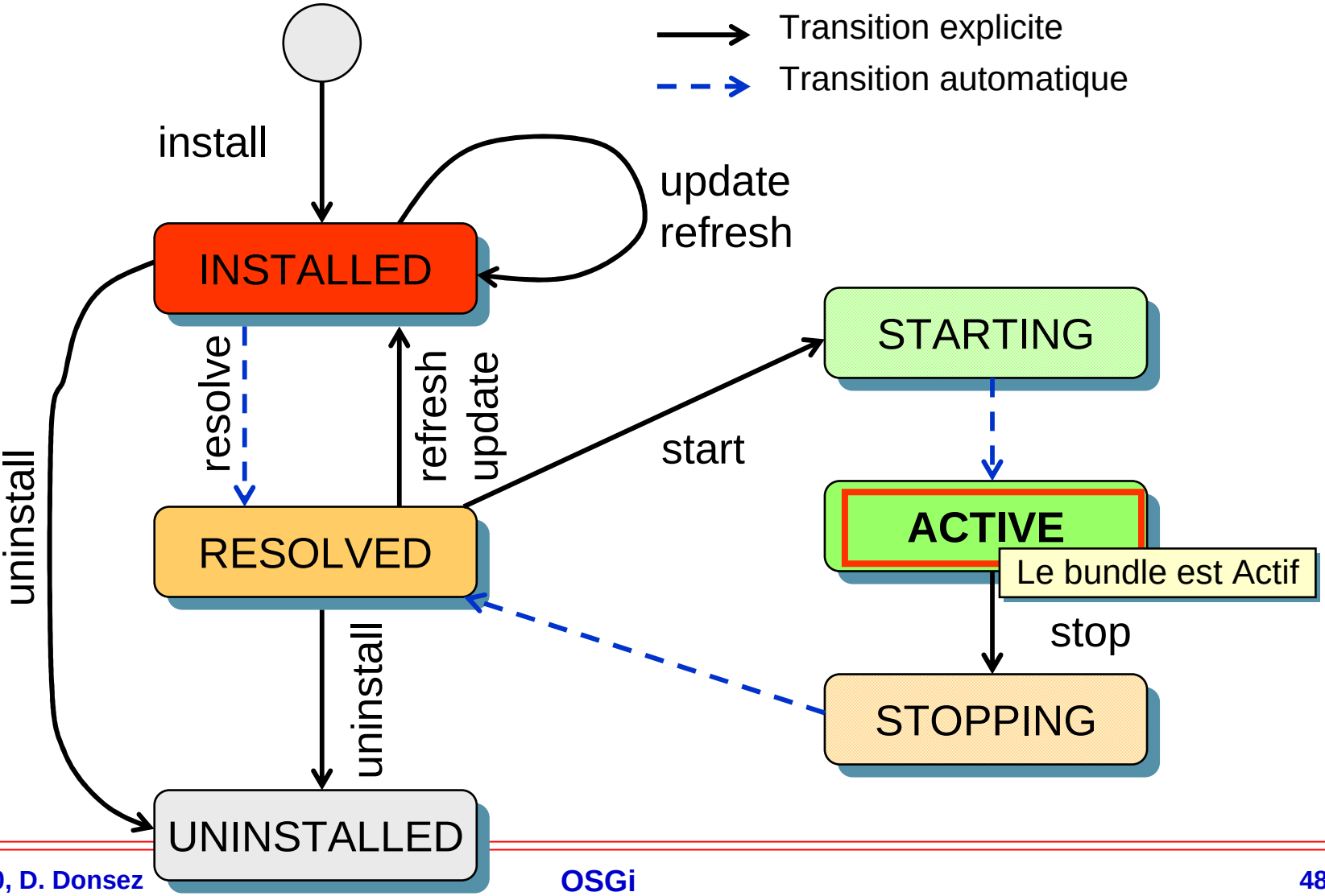
Cycle de vie d'un Bundle



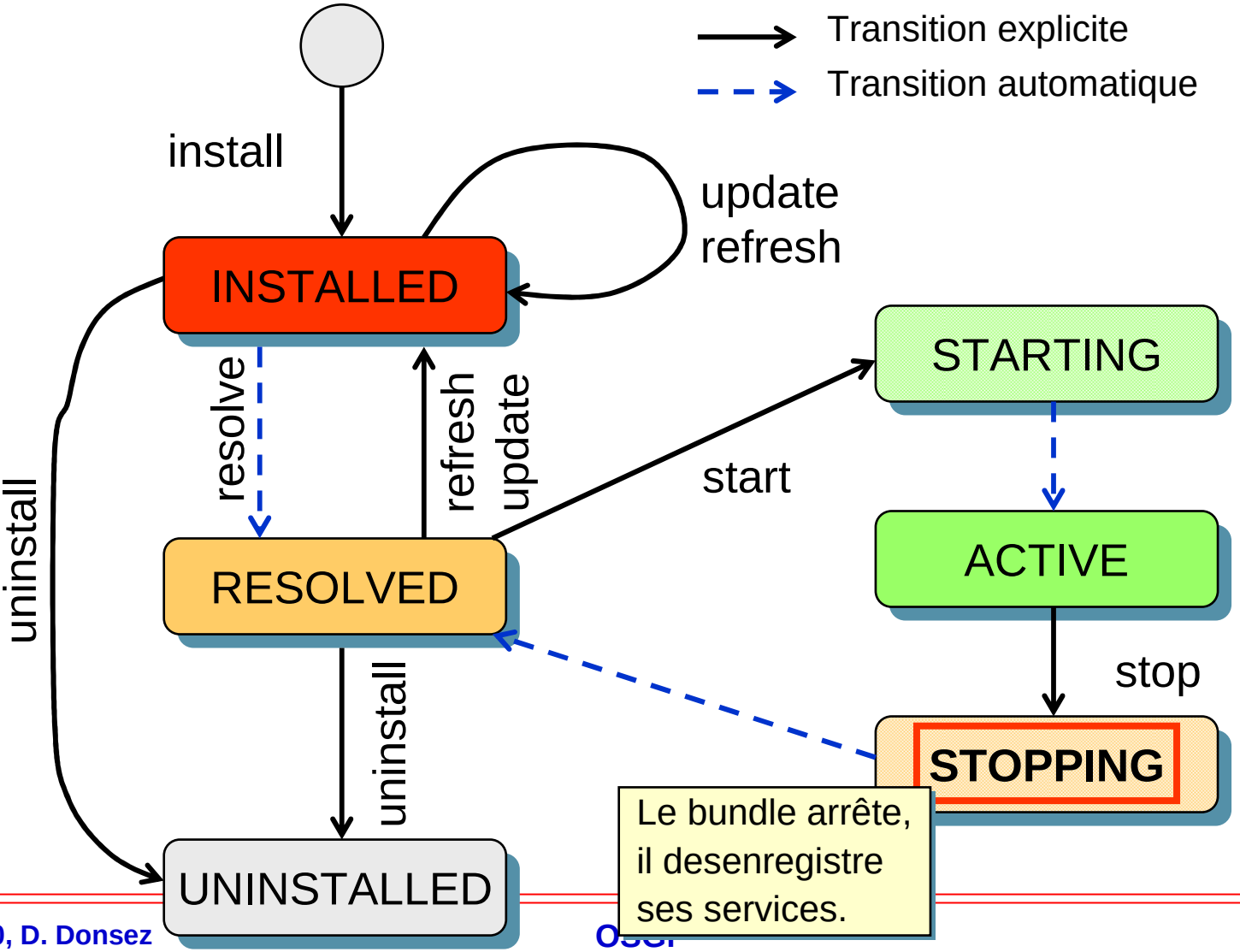
Cycle de vie d'un Bundle



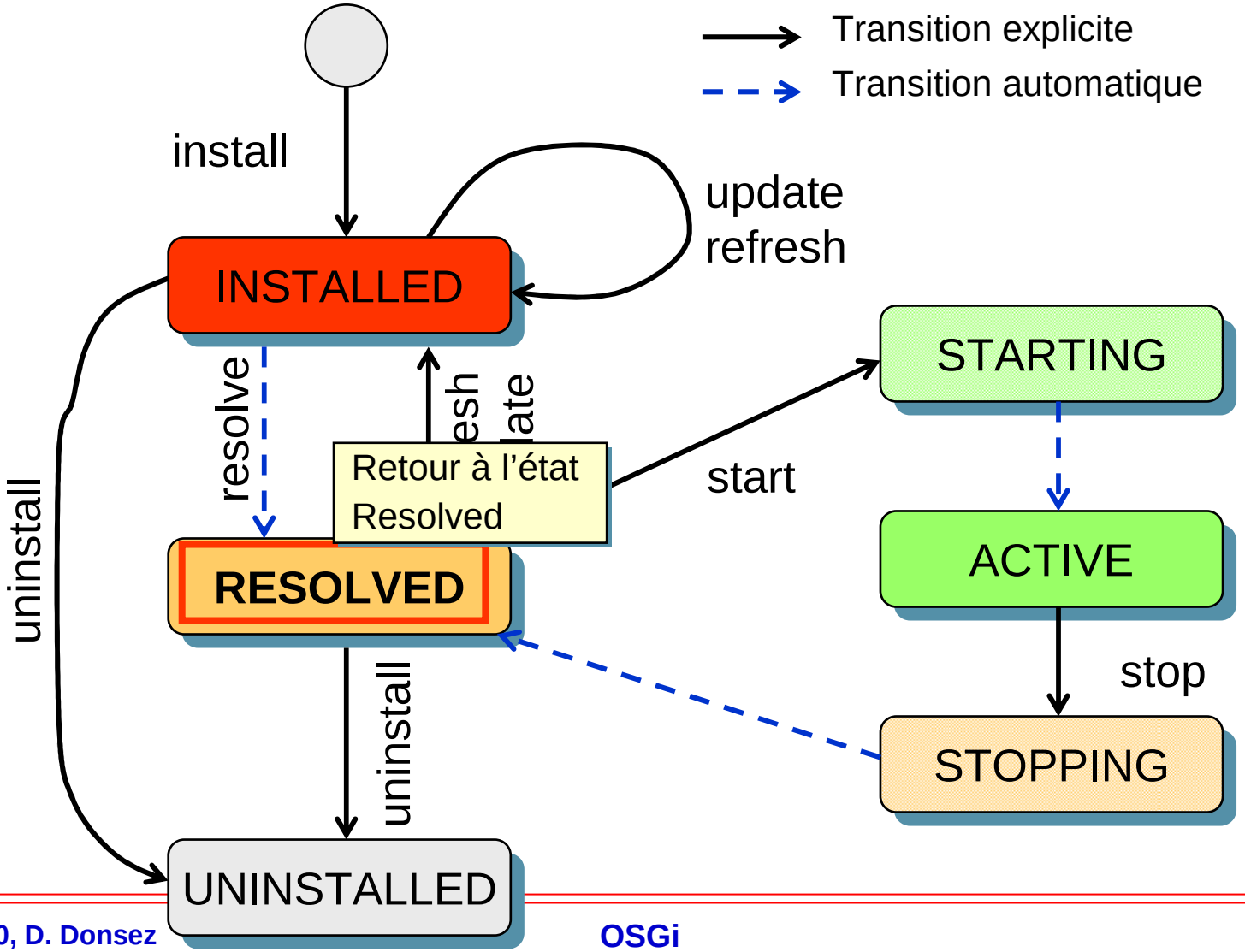
Cycle de vie d'un Bundle



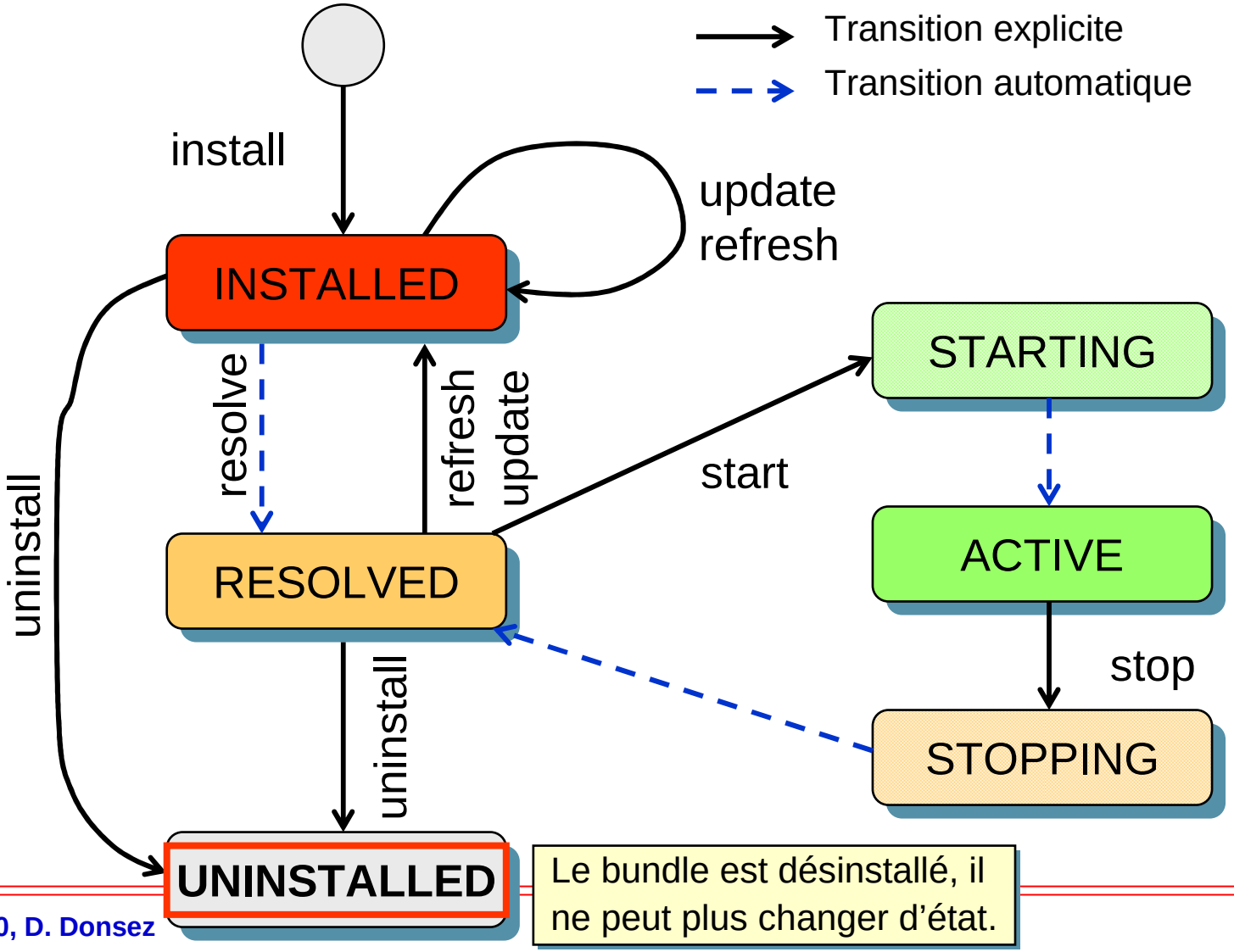
Cycle de vie d'un Bundle



Cycle de vie d'un Bundle



Cycle de vie d'un Bundle

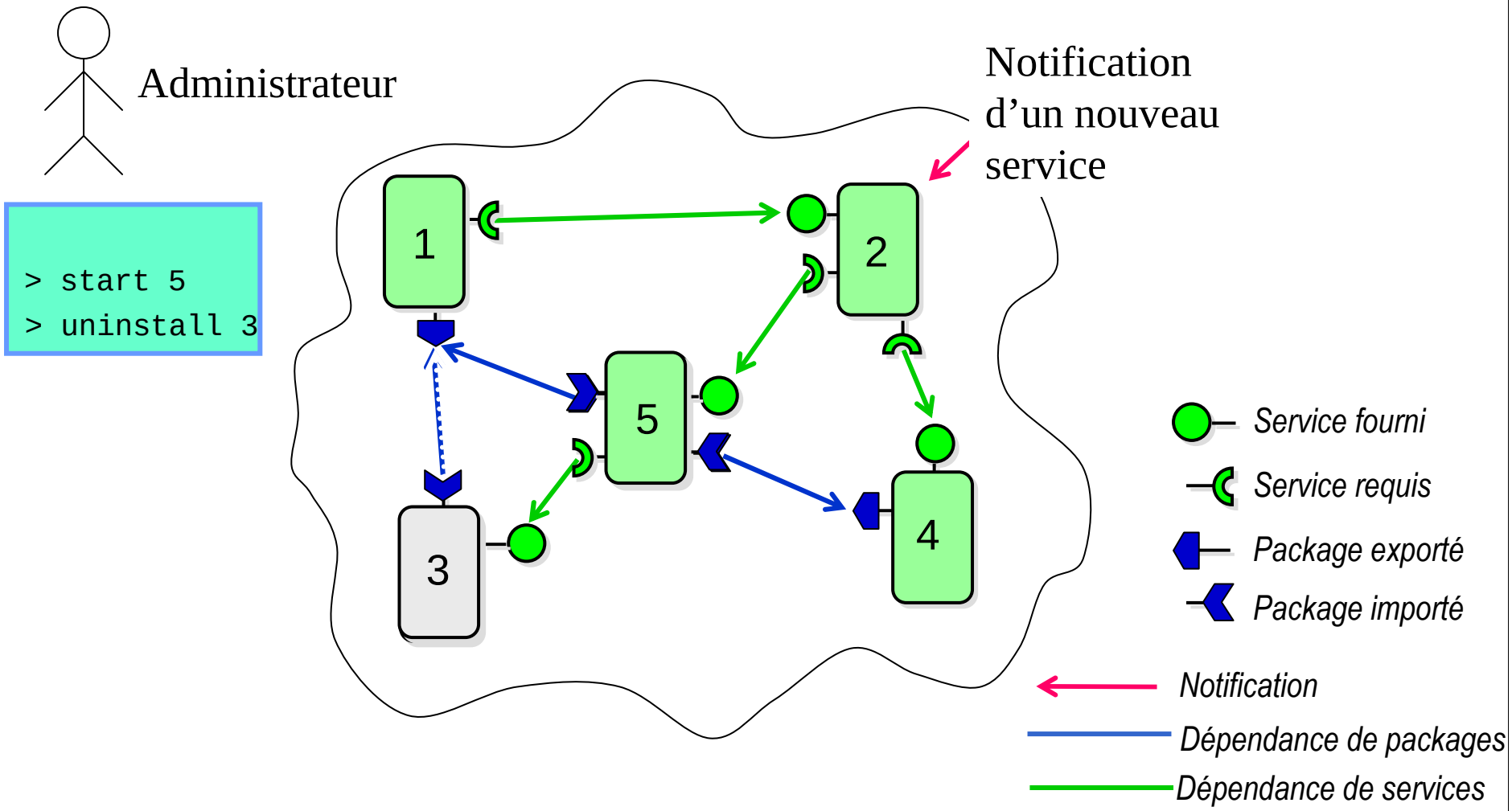


News in R4.1

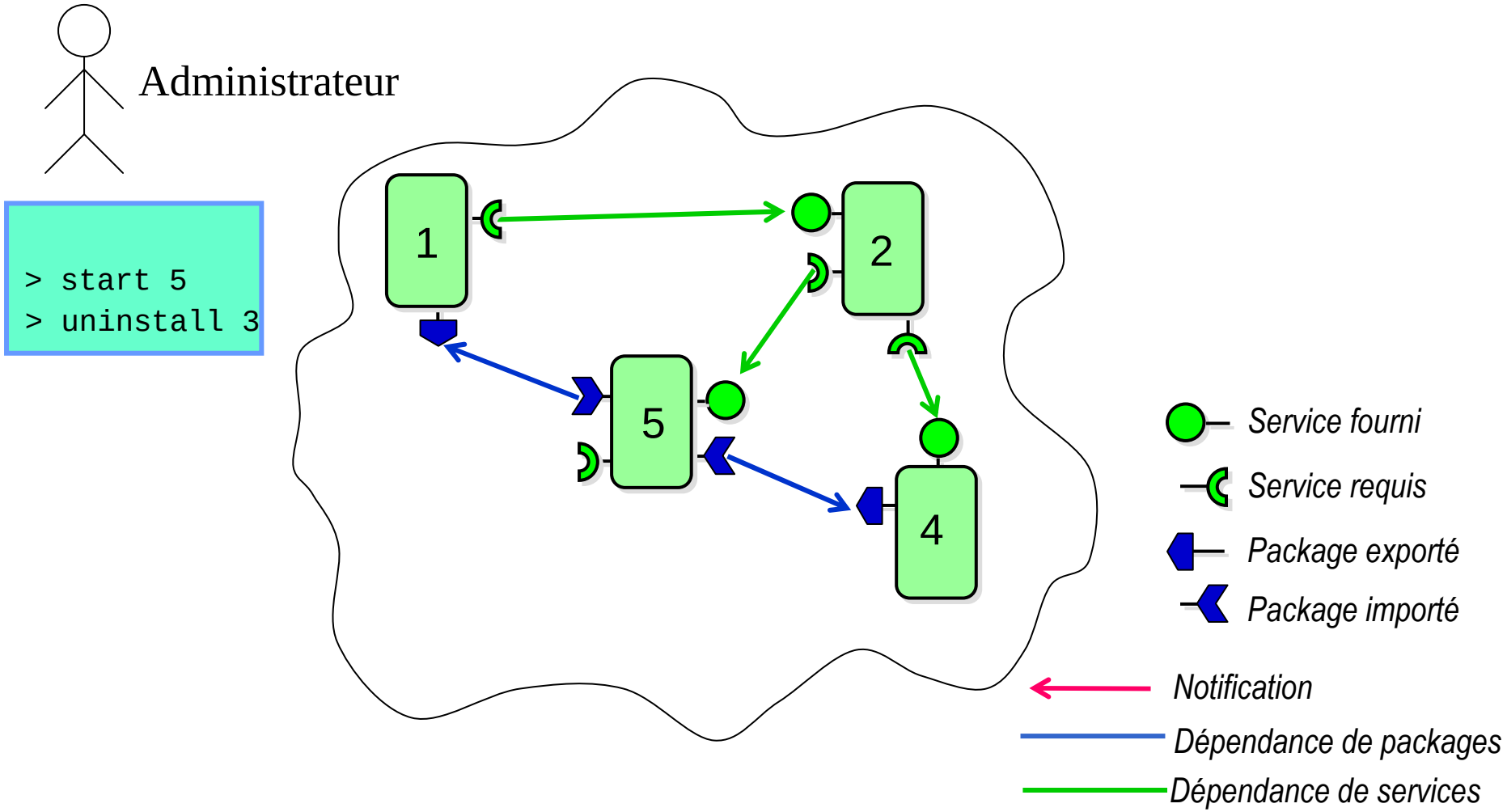
- By default, persistent start and stop
- Transient start and stop
  - Bundle.start(int transientFlag) and Bundle.stop(int transientFlag)
    - Flag sample: do not restart after FW restart (ie do no alter the autostart settin of the bundle)
- By default, « Eager » activation
  - The BundleActivator is instanciated when Bundle.start()
- Lazy Activation
  - The bundle is activate when a exported class is required by an other bundles

    → new bundle event (lazy activated)
    : useful for extender model
- A lire
  - http://www2.osgi.org/wiki/uploads/Conference/OSGi4.1Overview.pdf

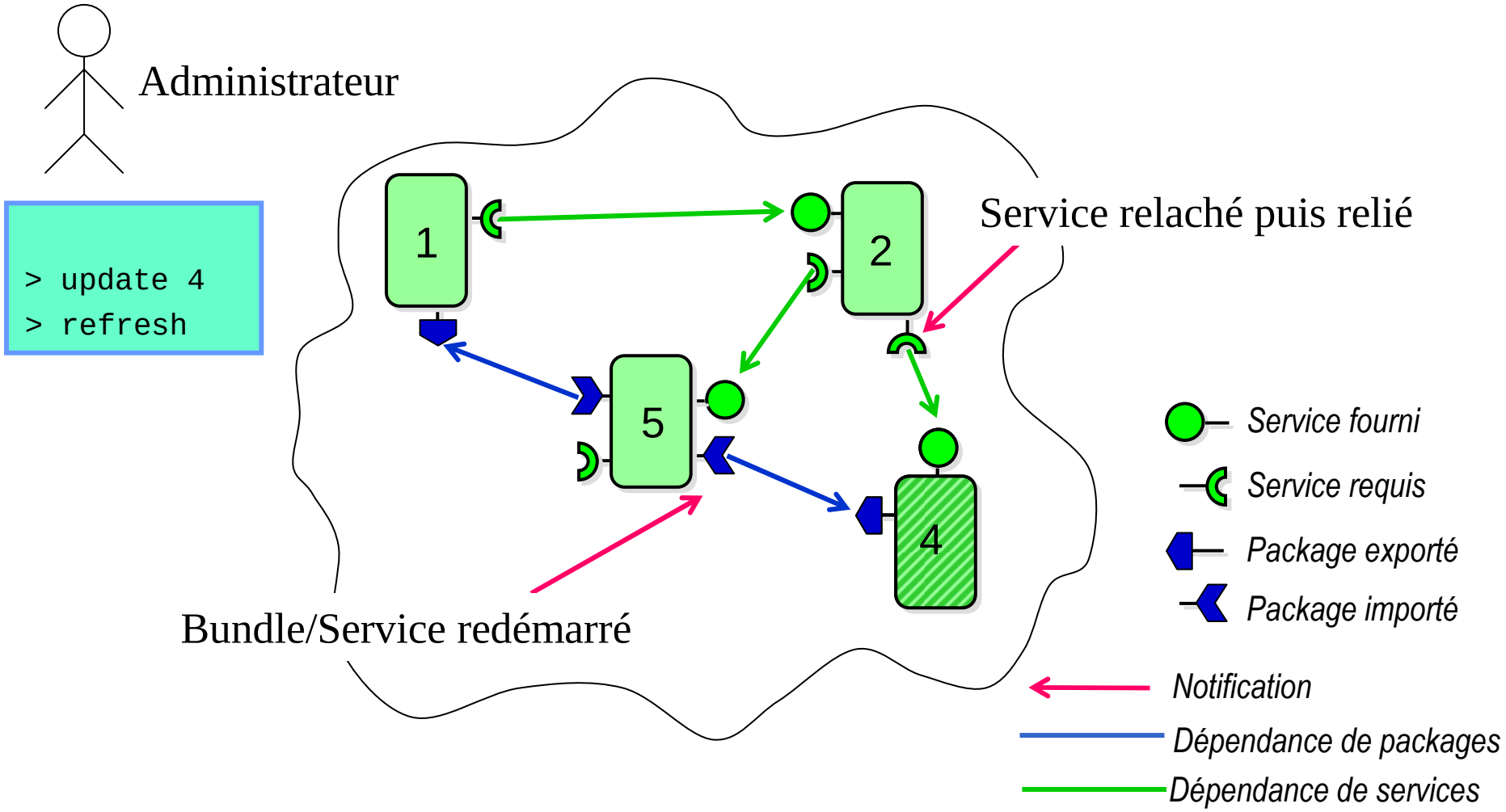
Dépendance & Dynamisme



Dépendance & Dynamisme

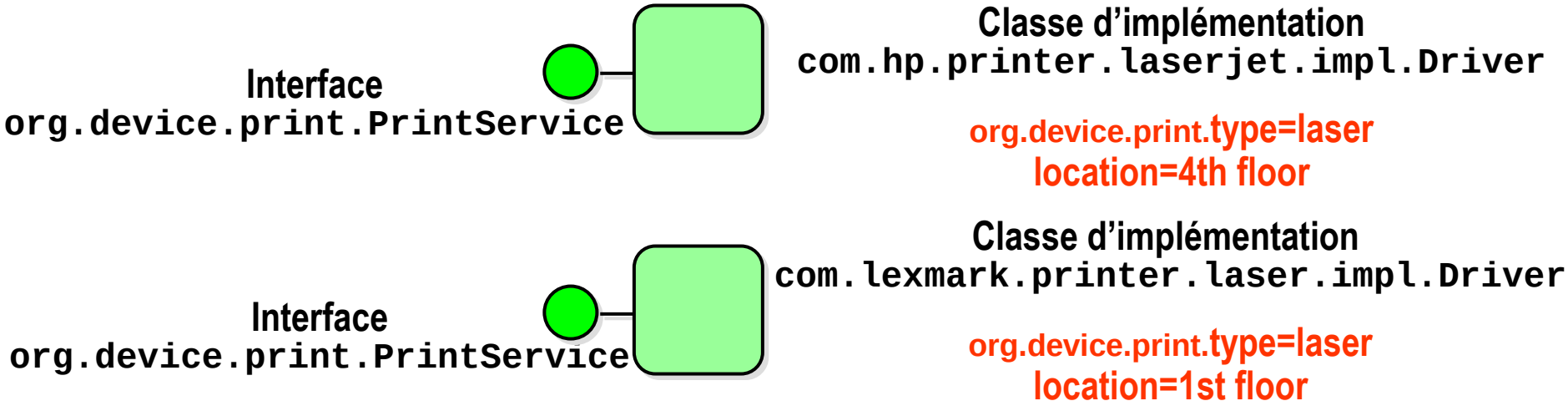


Dépendance & Dynamisme



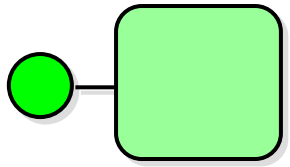
Service

- 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.



Exemple de service

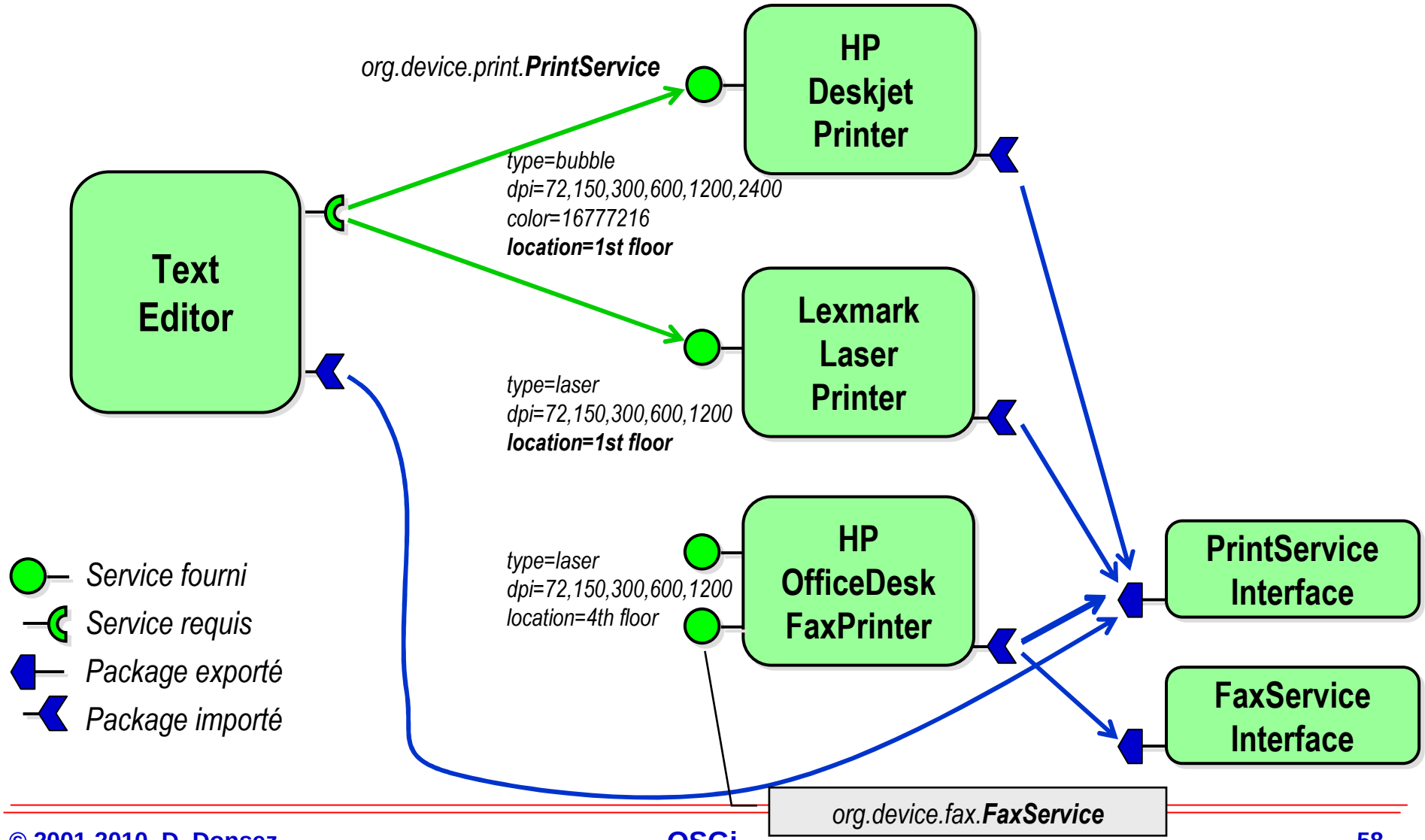
Interface
org.device.print.PrintService

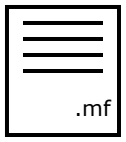


```
package org.device.print;
public interface PrintService {
  public static final String TYPE="org.device.print.type";
  public int print(OutputStream out,
                   String[] printparams)
            throws PrintException;
  public Job[] list()
            throws PrintException;
}
public interface Job[] { … }

public class PrintException extends Exception { … }
```

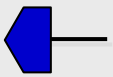
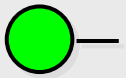
Exemple d'application





Fichier manifest (i)

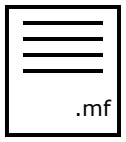
Informations nécessaires au framework

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

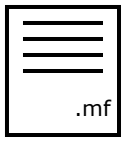


Fichier manifest (ii)

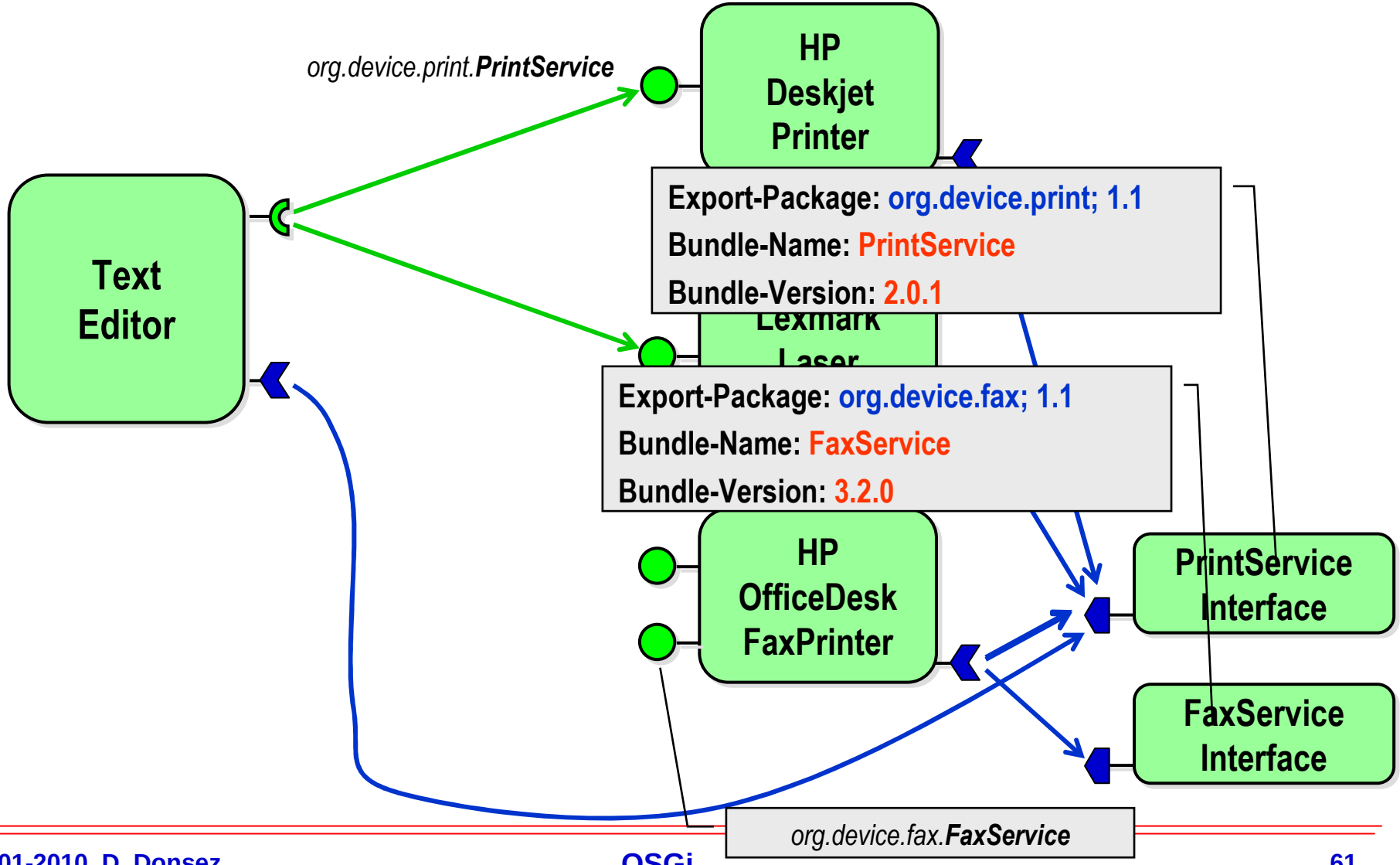
Build by Maven 2

Informations nécessaires au framework

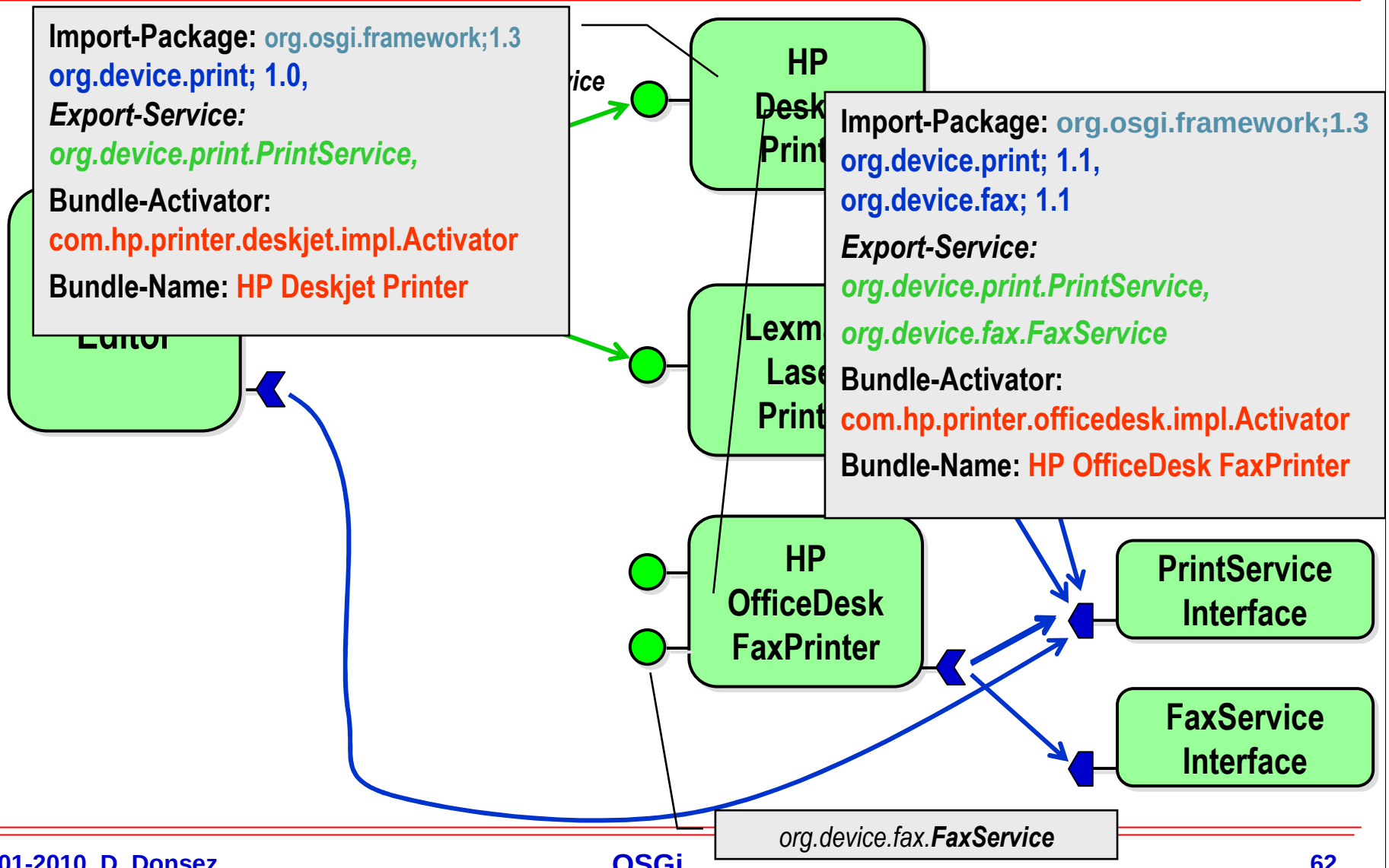
Bundle-SymbolicName r4	Nom symbolique du bundle (sert à l'identification)
Bundle-Name	Nom du bundle
Bundle-Description	Description du bundle
Bundle-Version	Version du bundle
Bundle-DocURL	URL de la documentation du bundle
Bundle-ContactAddress	Coordonnée du propriétaire du bundle
Bundle-Category	Catégorie 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.*)



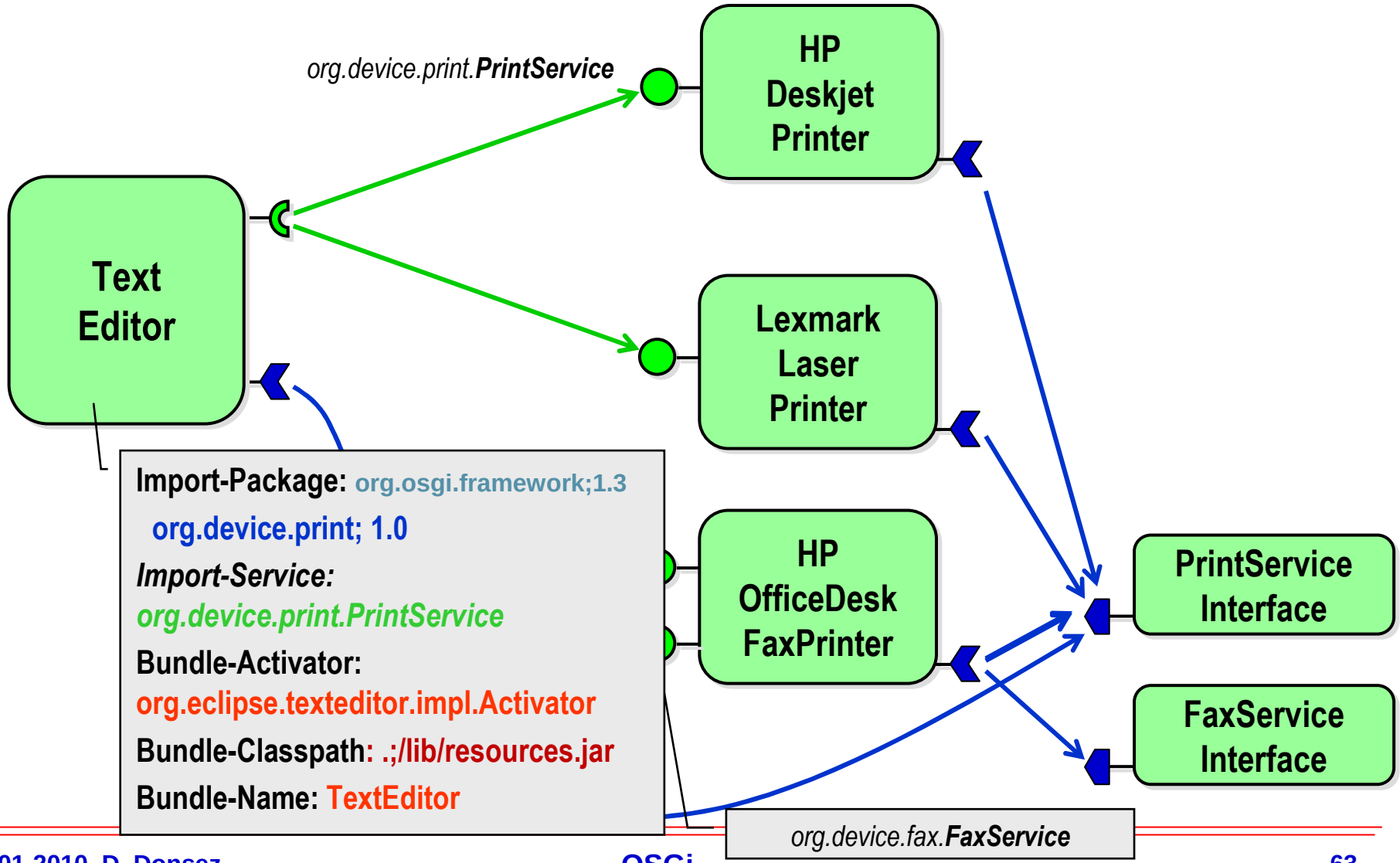
Exemple de manifest (i)



Exemple de manifest (ii)



Exemple de manifest (iii)

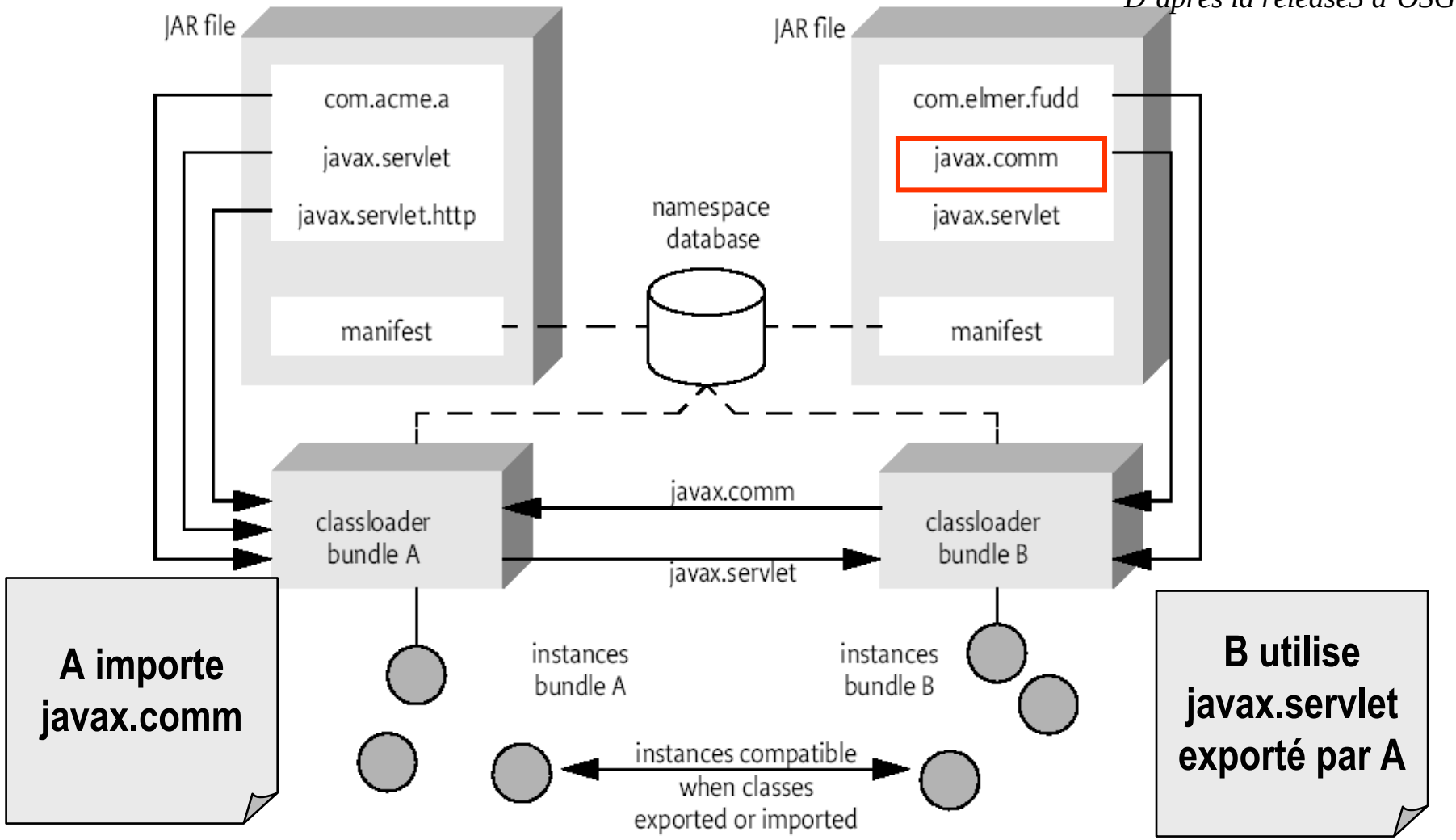


Chargement de classes (i)

- 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

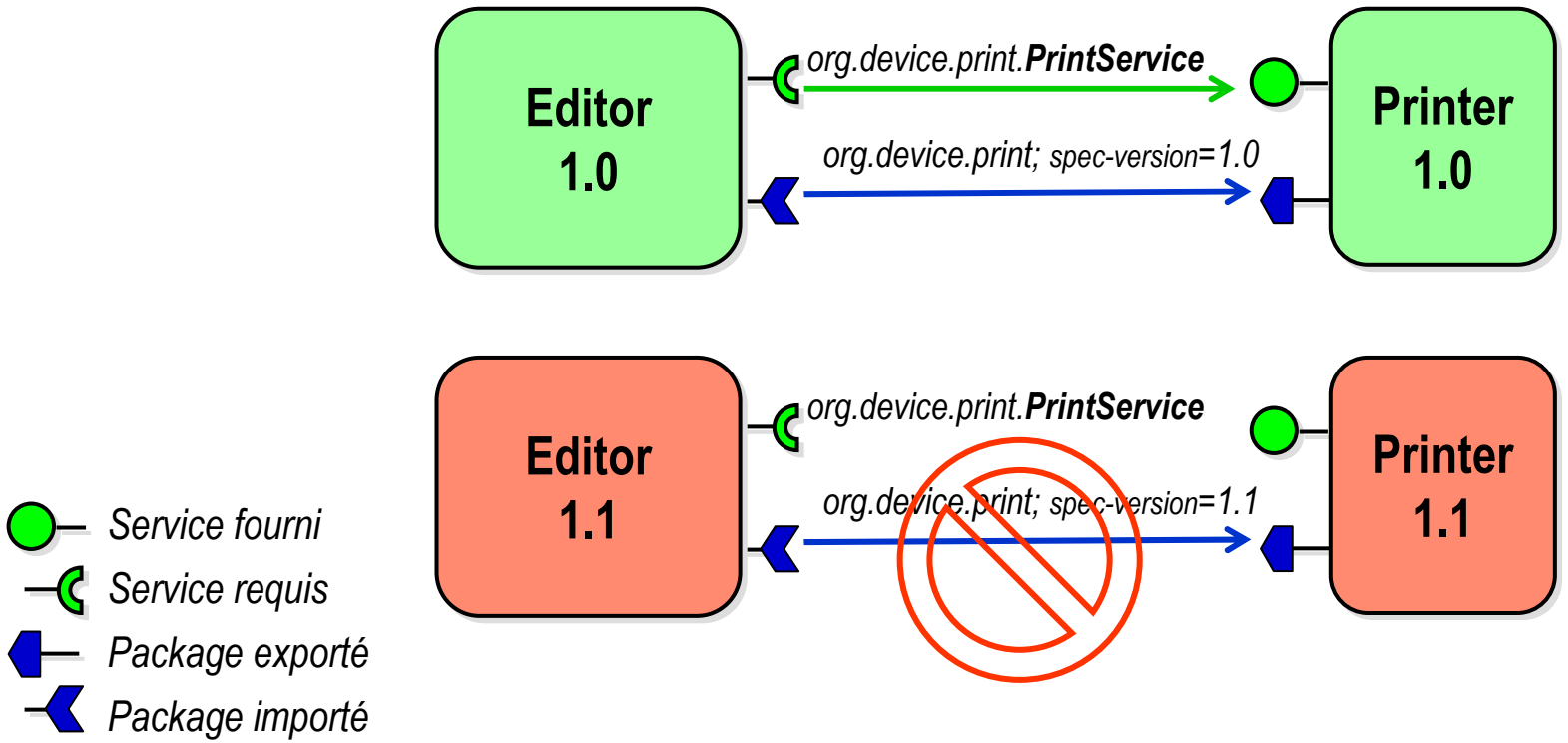
Chargement de classes (ii)

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 … Jigsaw

- 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

- http://www.osgi.org/blog/2008/12/project-jigsaw.html

DynamicImport-Package

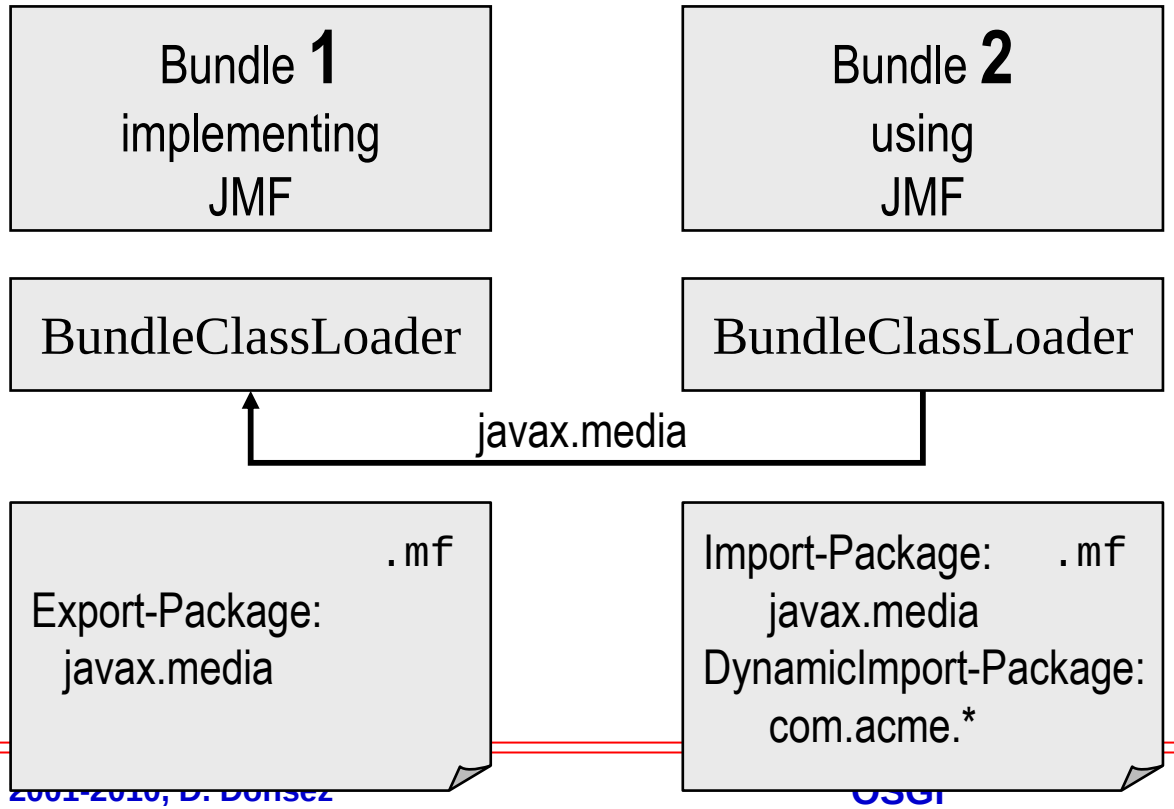
- Permet en cours d'exécution d'importer des packages non présents à la résolution
  - activator.getClass().getClassLoader().loadClass(clazzname)
  - *Surtout pas Class.forName(clazzname)*
- L'entrée `DynamicImport-Package` du mainfeste liste les packages qui pourront être importés en cours d'exécution
- Usage : framework à plugin ou service provider
  - Exemple : JMF, JCE, JDBC, …

DynamicImport-Package
Exemple avec JMF (i)

Installation de bundle 2

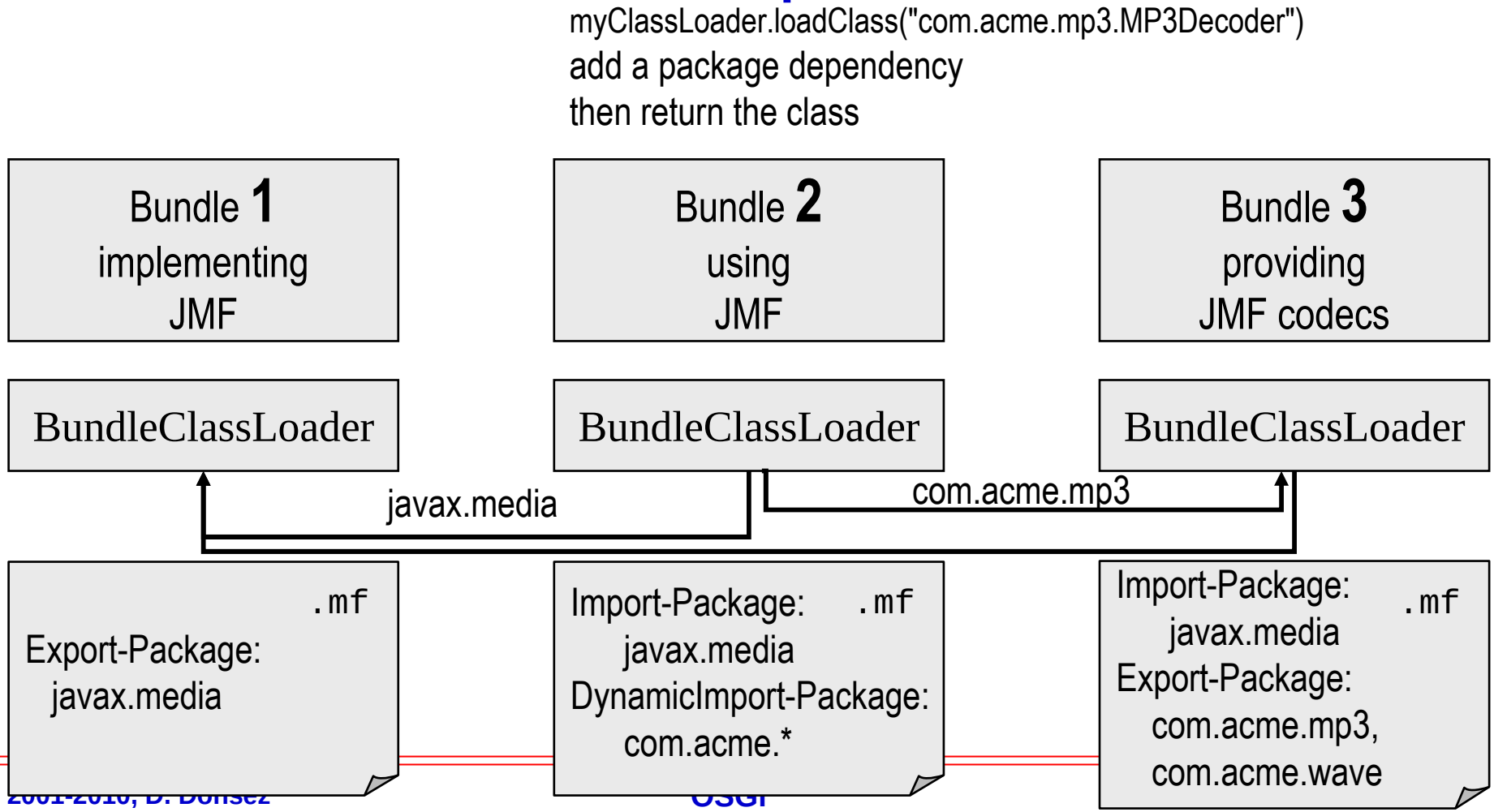
- **il passe à l'état ACTIVE**

myClassLoader.loadClass("com.acme.mp3.MP3Decoder")
throws java.lang.NoClassDefFoundError



DynamicImport-Package
Exemple avec JMF (ii)

Installation de bundle 3 : il passe à l'état ACTIVE



Importation et Exportation conditionnelles r4

- Motivation
  - La compatibilité descendante (*backward compatibility*) est lourde à maintenir (surtout en embarqué).
  - Les @deprecated disparaissent parfois lors des chargements des majeures de versions
- Idées
  - Intervalle de version
  - Import-Package:
    javax.servlet; version="[2.0.0,2.4.0)"; resolution="optional"
- Autres
  - Export-Package:
    org.foo.service; version=1.1; vendor="org.foo",
    org.foo.service.bar; version=1.1; uses="org.foo.service",
    org.foo.service.fizz; include:="*Impl"; exclude:="Test*"

  - Richard S. Hall, Java Modularity Support in OSGi R4, ApacheCon (San Diego), December 14th, 2005
    - http://docs.safehaus.org/download/attachments/2995/osgi-apachecon-20051214.pdf

Accès aux ressources et aux fichiers

Ressources

- this.getClass().getRessourceAsStream(String path)

  path="/" correspond à la racine du JAR (BUNDLE-CLASSPATH)

Support de persistance

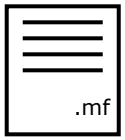
- BundleContext.getDataFile(String path)

  path="" correspond à la racine du cache du bundle

- FileService

  accès au système de fichier local (s'il existe !)

  et contrôle les permissions du Bundle au passage



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 !

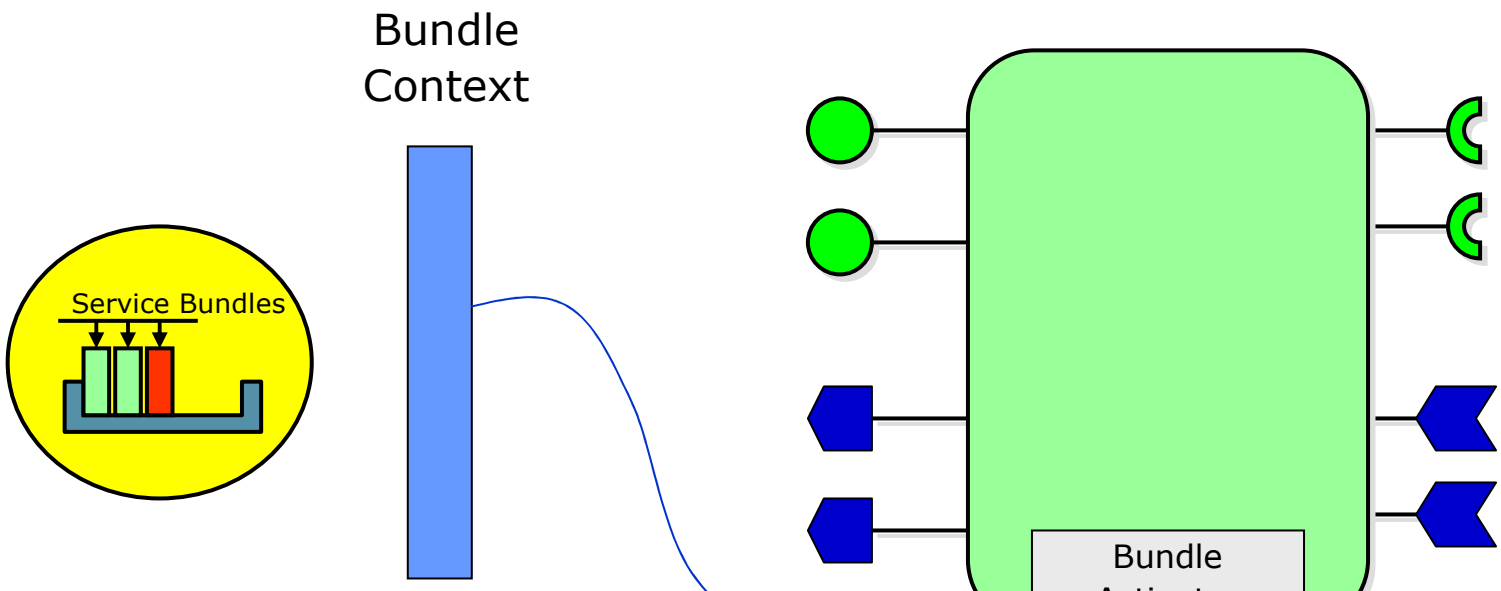
il peut ne pas y avoir de BundleActivator dans un bundle

- Livraison de classes et ressources
- Eclipse extension points
- Extender model

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

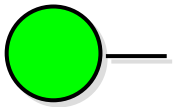


- registerService()
- getServiceReferences()
- getService()
- getDataFile()
- addServiceListener()
- addBundleListener()
- addFrameworkListener()

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

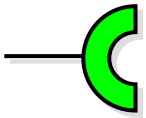
Enregistrement de services (Lexmark Laser Printer)

```
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))" );
```

Comparaison avec le courtage de JINI

- JINI
  - Typage fort sur le nom de l'interface et la signature de ces méthodes
  - Sous-typage des interfaces
  - Propriétés de courtage
  - Distribué
  - Notion de groupe
  - Bail (lease) d'un enregistrement
- OSGi
  - Typage sur le nom de l'interface
  - Non
  - Oui
  - Centralisé (même JVM)
  - Non
  - Non

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)

    R2
- **ServiceEvent**
  - Notifie l'enregistrement ou le retrait de services
  - interface `ServiceListener` méthode `serviceChanged`

    Traitement séquentiel et synchrone des listeners

Service Registry Hooks (RFC 126 R4.2)
org.osgi.framework.hooks

- Hooks on the service registry operations

- PublishHook
  - Bundles registering this service will be called during framework service publish (register, modify, and unregister service) operations. This method is called prior to service event delivery when a publishing bundle registers, modifies or unregisters a service and can filter the bundles which receive the event.
- FindHook
  - Bundles registering this service will be called during framework service find (get service references) operations. This method is called during the service find operation by the finding bundle and can filter the result of the find operation.
- ListenerHook
  - Bundles registering this service will be called during service listener addition and removal. The hook is notified of the collection of service listeners and what they may be listening for and well as future changes to that collection.

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); } }
```

Ajout de ServiceEvent.MODIFIED_ENDMATCH en R4.2

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)

- **Le BindingController est incomplet**
  - Pas de synchronisation
  - Les services apparus et disparus entre le getServiceReferences et le addServiceListener

- **Mini conclusion**
  - Vous avez suivi ?
  - Et maintenant avec 6 services dont 3 obligatoires

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 (Cervantes & Hall)

ServiceTracker R2

- **Motivation**
  - Simplifier l'usage des ServiceListeners
- **ServiceTracker**
  - Classe utilitaire pour suivre un type de service (interface + filtre LDAP)
- **ServiceTrackerCustomizer**
  - Interface de rappel
- **Remarque**
  - peut être fourni par le framework afin d'en optimiser l'implémentation

ServiceTracker : Exemple

Dans le start(BundleContext)

```
serviceTracker=new ServiceTracker(
                bundleContext,
                bundleContext.createFilter(
                "(&(objectClass=org.device.print.PrintService)(type=laser))"),
                (ServiceTrackerCustomizer)null);
serviceTracker.open();
```

Dans les méthodes du service

```
PrintService ps=(PrintService) serviceTracker.getService();
ps.print(…); …
Job[] jobs=ps.list(); …
ps=null;
```

Dans le stop(BundleContext)

```
serviceTracker.close();
```

ServiceTrackerCustomizer : Exemple

```
class MyServiceTrackerCustomizer implements ServiceTrackerCustomizer  {

    public Object addingService( ServiceReference reference) {
      Object obj = context.getService(reference);
      HttpService servant = (HttpService)obj;
      // Register the Servlet using servant
      ...
      return servant;
    }

    public void removedService( ServiceReference reference, Object object ){
      HttpService servant = (HttpService)obj;
      // Unregister the Servlet using servant
      ...
      context.ungetService(reference);
    }

}
```

Service Component Runtime

Modèle simple de composants orienté service

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

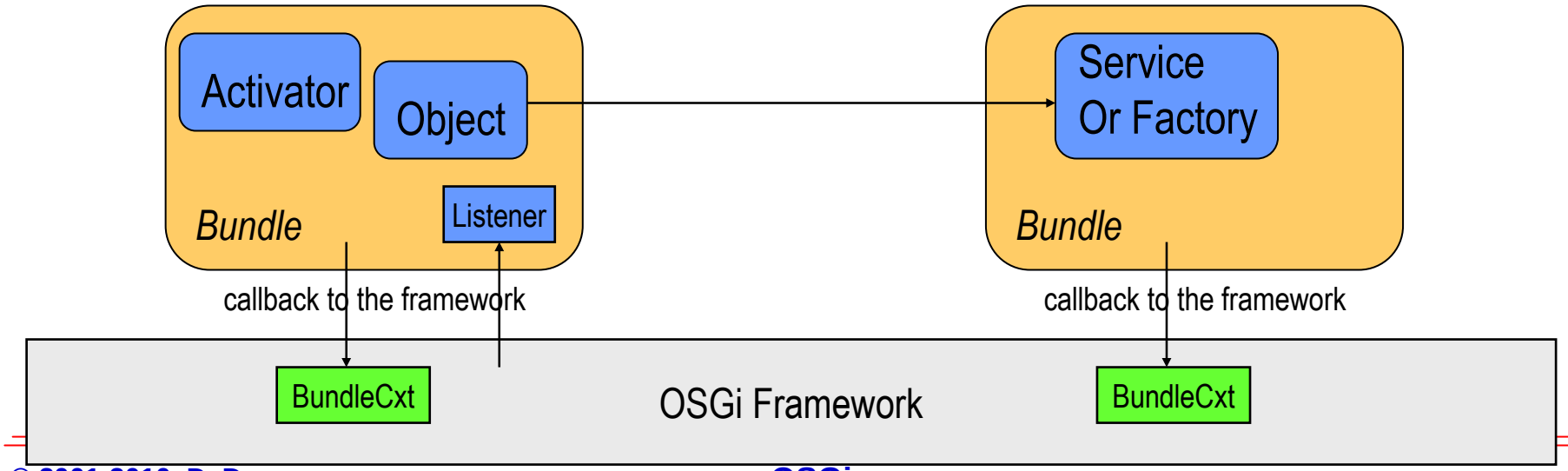
Caractéristiques

- 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

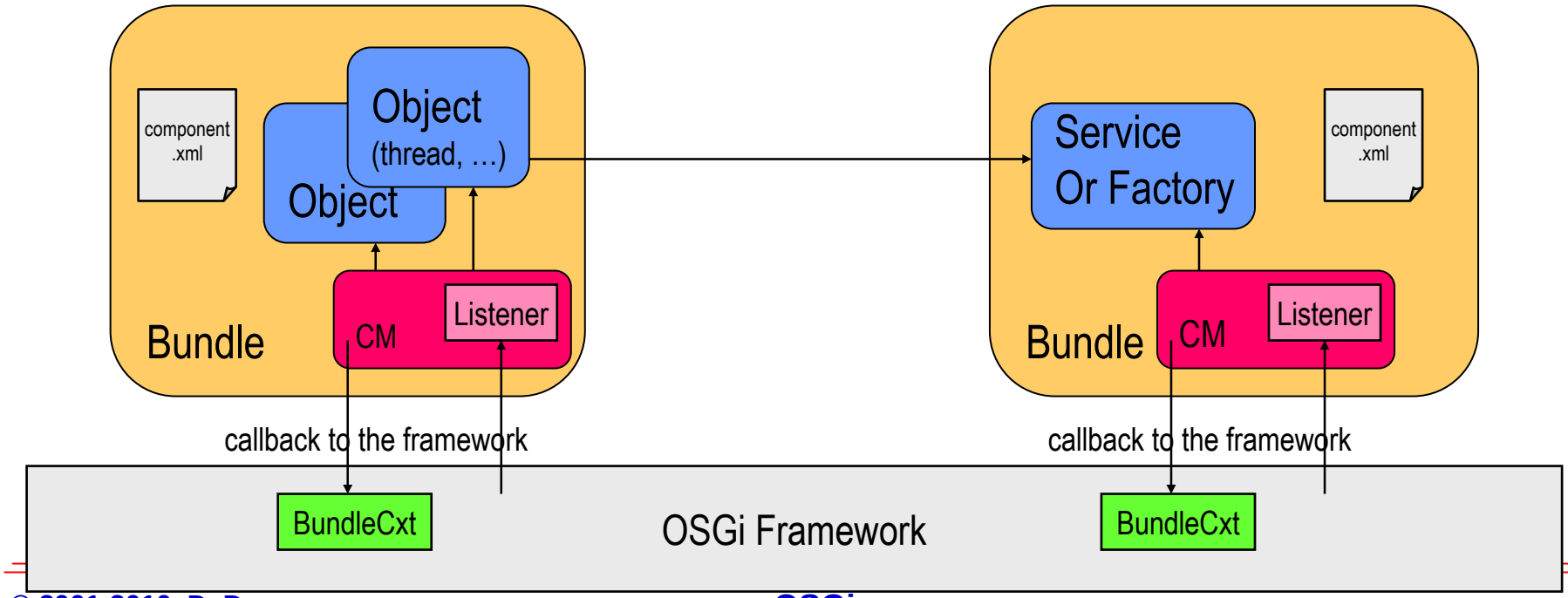
- Démarrage des instances de services
- Listeners pour gérer la liaison dynamique vers les services requis



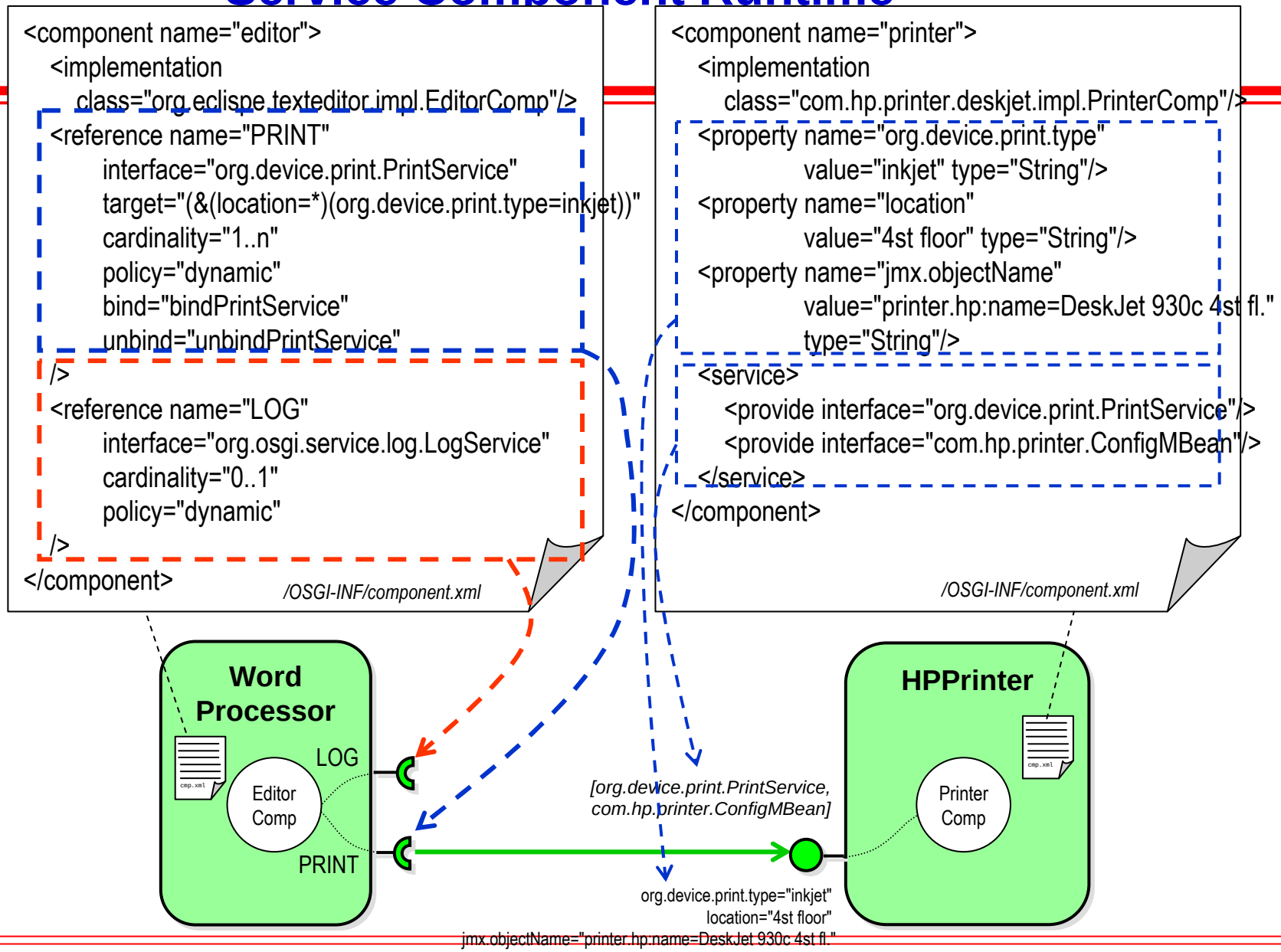
Service Component Runtime

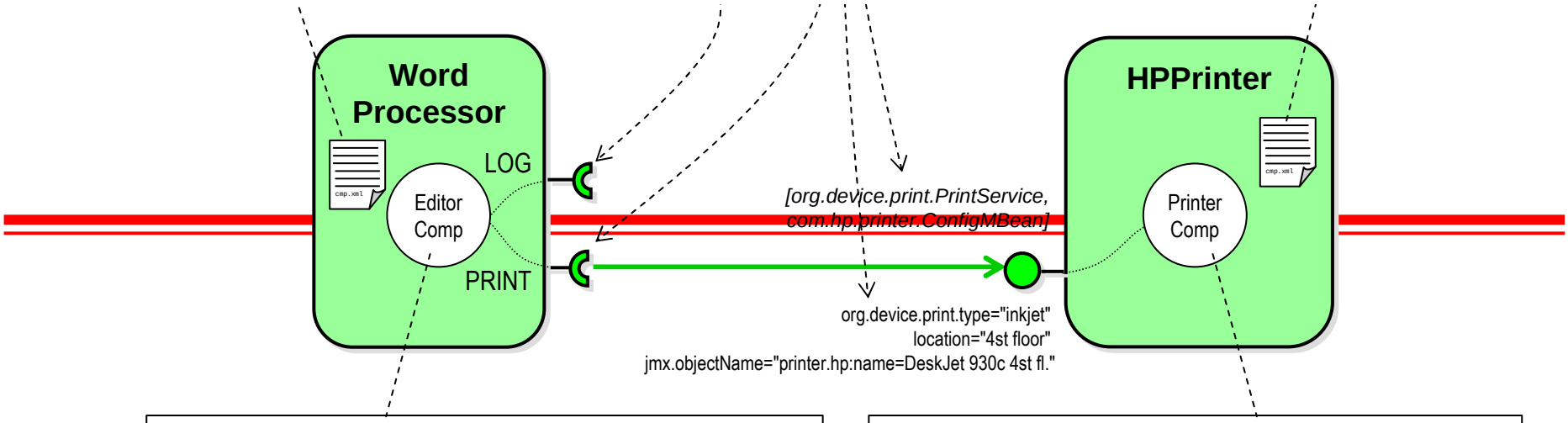
- **Gestion descriptive des instances de services et des liaisons**
  - **OSGI-INF/component.xml**

CM: SCR Component Manager



Service Component Runtime






```
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); }
   }
   …
}
```

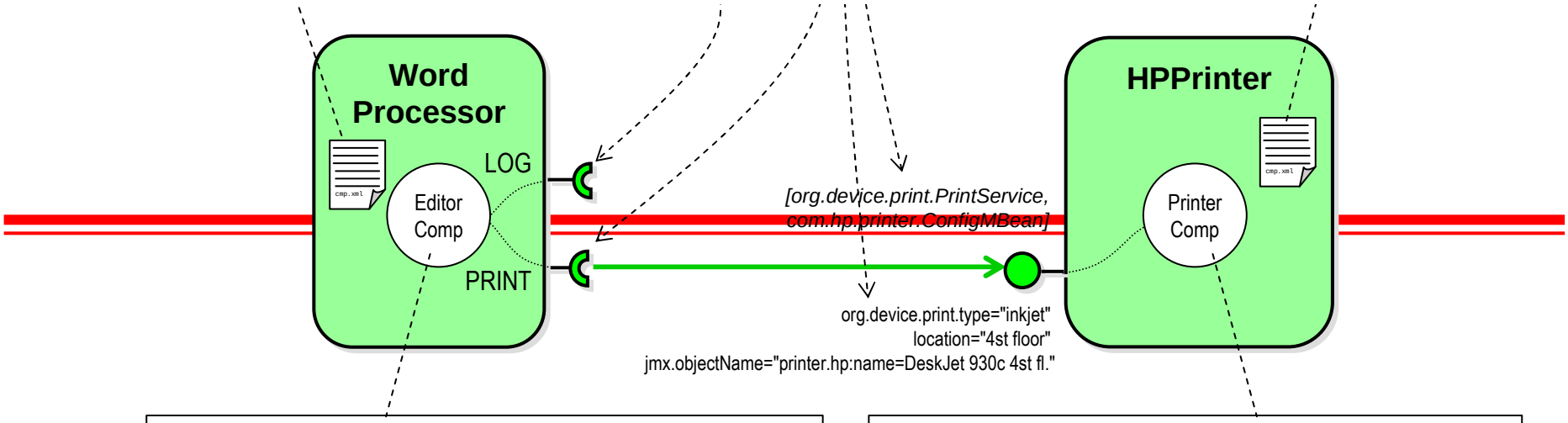
```
public class PrinterComp
  implements PrintService, ConfigMBean {

   // 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){
     …
   }
   …
}
```




```
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){
     …
   }
   …
}
```

iPOJO (Apache Felix)

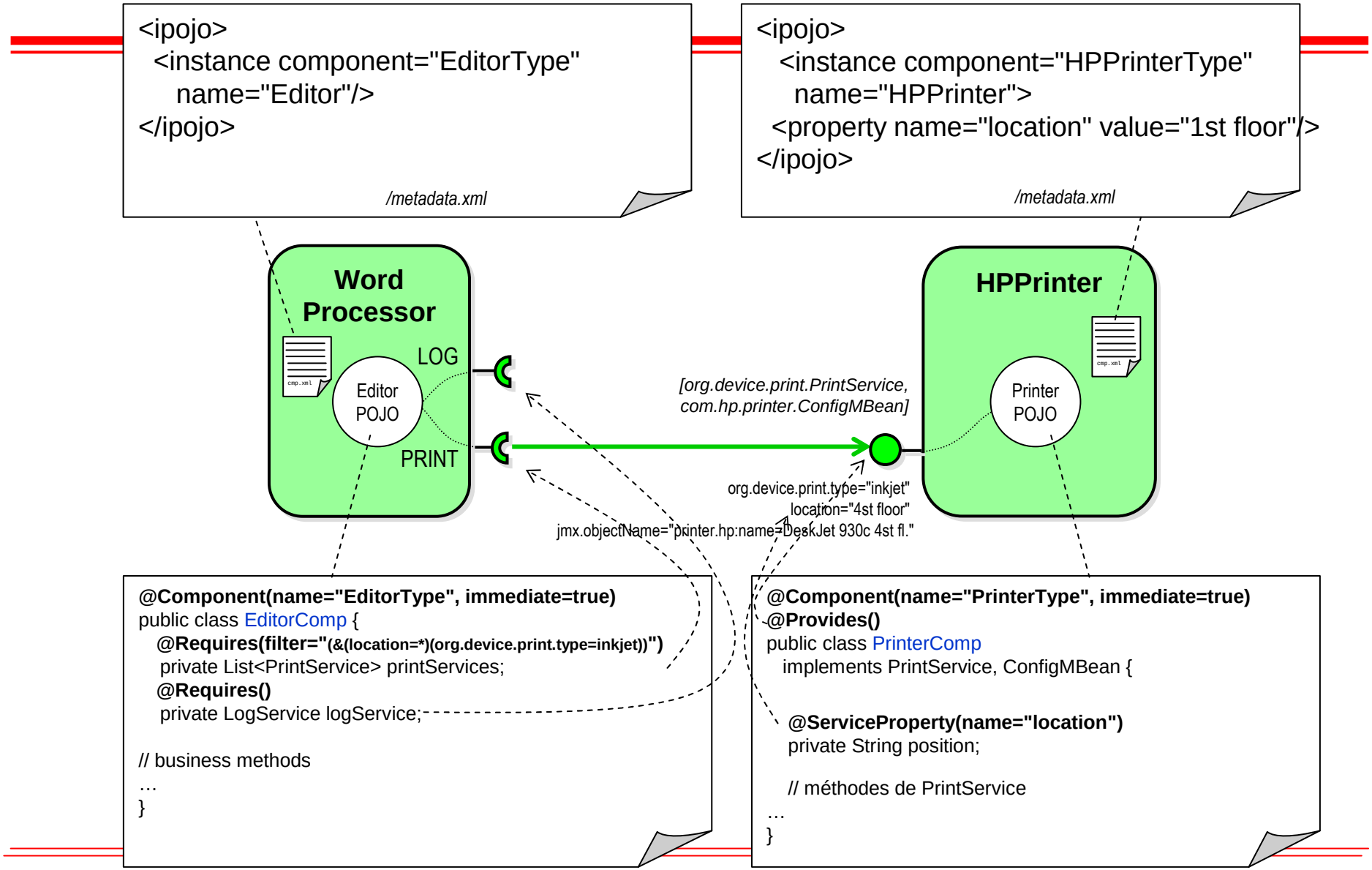
Motivations

- Exécuter de pur POJOs (Plain Old Java Object)
- 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 (eux meme des iPOJO)*
- Les handlers travaillent en fonction des metadonnées décrites dans un descripteur (.mf,.xml, @nnotations 1.5, *doclet* …)

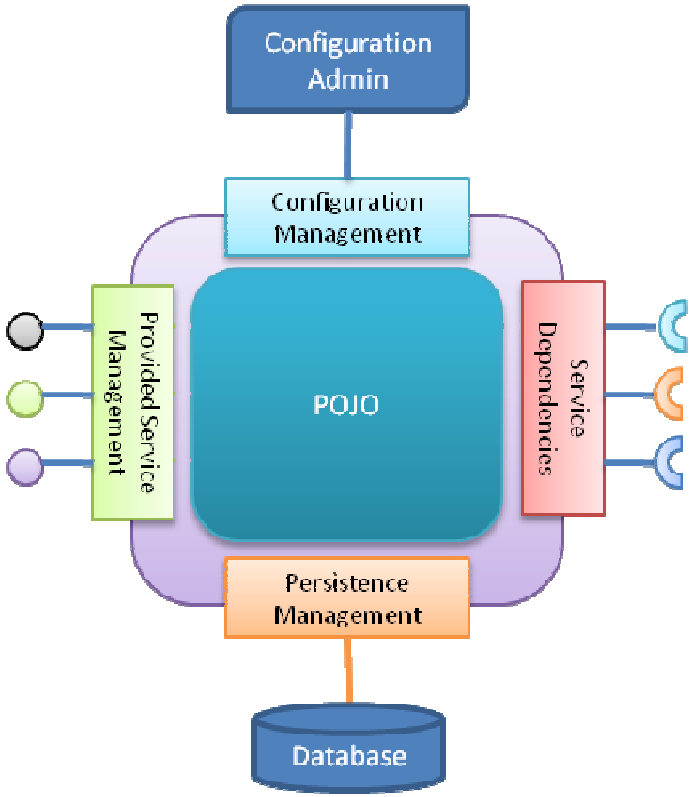
iPOJO example



iPOJO Handlers

- Provide
- Service Requirement
  - Method injection and Field injection
  - Nullable object, Default implementation
- Temporal Service Requirement
- Lifecycle callback
- Config
- JMX
- Event
- Extender Pattern
- Whiteboard Pattern

- Custom ones …



iPOJO Architecture handler

Disable architecture

- <component classname="fr.imag.adele.escoffier.hello.impl.HelloServiceImpl" architecture="false">

Command

- arch => displays instances name & state (equivalent to arch \-instances)
- arch -instance $instance_name => displays complete information about the instance $instance_name
- arch -factories => display the list of available factories
- arch -factory $factory_name => display complete information about the factory $factory_name
- arch -handlers => list available handlers

iPOJO Configuration Handler

- **Add a CM ManagedService**

```
<iPOJO>
<Component className="system.impl.DiskServiceImpl">
  <Provides>
    <Property name="quota" field="m_quota"/>
  </Provides>
  <Properties propagation="true"
              updated="afterReconfiguration"
    />
    <Property name="quota" field="m_quota"/>
    <Property name="threshold" method="updateThresholdArray"/>
    <Property name="disk.name" type="java.lang.String"/>
  </Properties>
</Component>
<instance component="sysrtem.impl.DiskServiceImpl" name="DiskService">
  <property name="quota" value="100"/>
  <property name="threshold" value="\{10, 20, 30}"/>
  <property name="disk.name" value="D"/>
  <property name="managed.service.pid" value="com.acme.system.disk.D"/>
</instance>
</iPOJO>
```

Updates are propagated to the Service Registration

Invoke the method on update

Static property

Instance level

service.pid

```
class DiskServiceImpl implement DiskService {
  int m_quota;
  public void updateThresholdArray(int[] a) { … }
  public void afterReconfiguration(Dictionary config) { ... }
}
```

iPOJO JMX Handler

- XML

```
<ipojo xmlns:jmx="org.apache.felix.ipojo.handlers.jmx">
      ...
<jmx:config>
   <jmx:property name="quota" field="m_quota"
                        rights="r" notification="true"/>
   <jmx:method name="setQuota"/>
 </jmx:config>
</ipojo>
```

- @nnotations (not JSR 255 JMX annotations)

```
@Component
@Config(domain="my-domain", usesMOSGi=false)
public class Disk {
   @Property(name="quota", notification=true, rights="r", description="Disk quota")
  private String m_quota;
   @Method(description="set the quota") // Method published in the MBean
   public void setQuota(int q) {
      if(q>0) m_quota = q;
   }
}
```

iPOJO Event Handler Example

Publication

```
@org.apache.felix.ipojo.handlers.event.Publisher(name="p3", synchronous=true,
    topics="bar")
org.apache.felix.ipojo.handlers.event.publisher.Publisher publisher3;

public void doSomething() {
      Dictionary d = new Properties();
      // Fill out the even
          publisher3.send(d); // Send event
  }
```

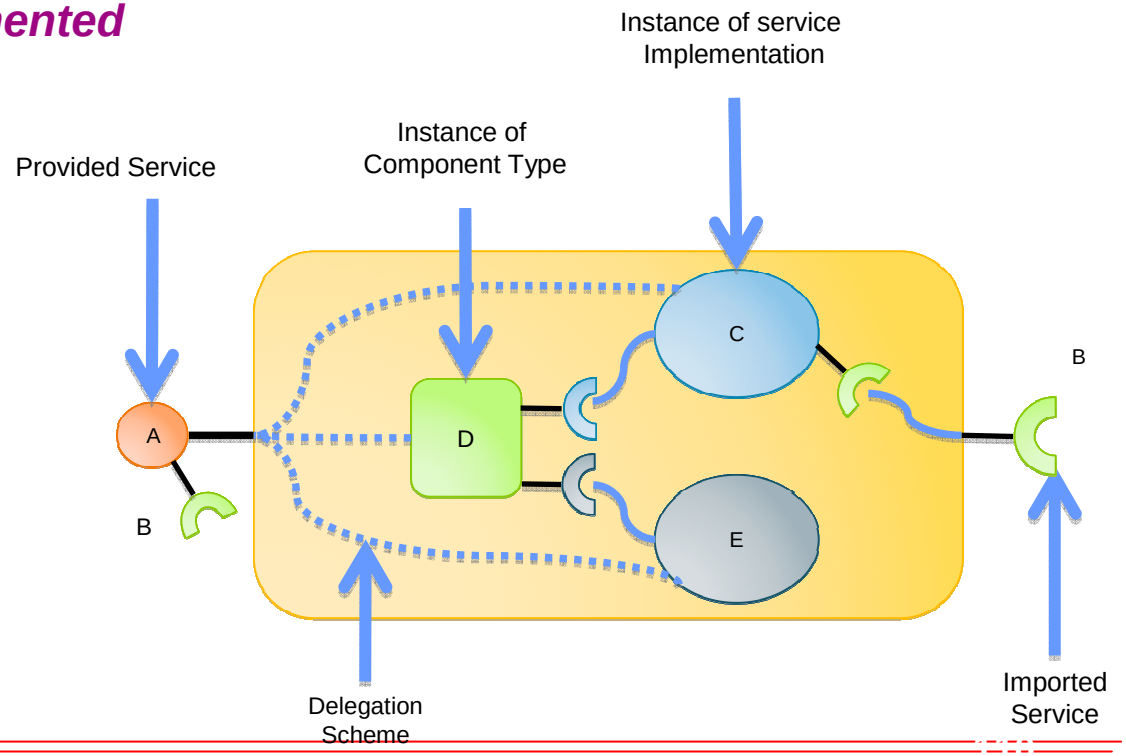
Subscription

```
@Subscriber(name="s1", data_key="data")
public void receive1(Object o) { // Nothing }
@Subscriber(name="s2", topics="foo,bar", filter="(data=DIDIER*)")
public void receive2(Event e) { // Nothing }
@Subscriber(name="s3", topics="foo", data_key="data", data_type="java.lang.String")
public void receive3(String s) { // Nothing }
```

iPOJO Composite

Architecture Description Language for

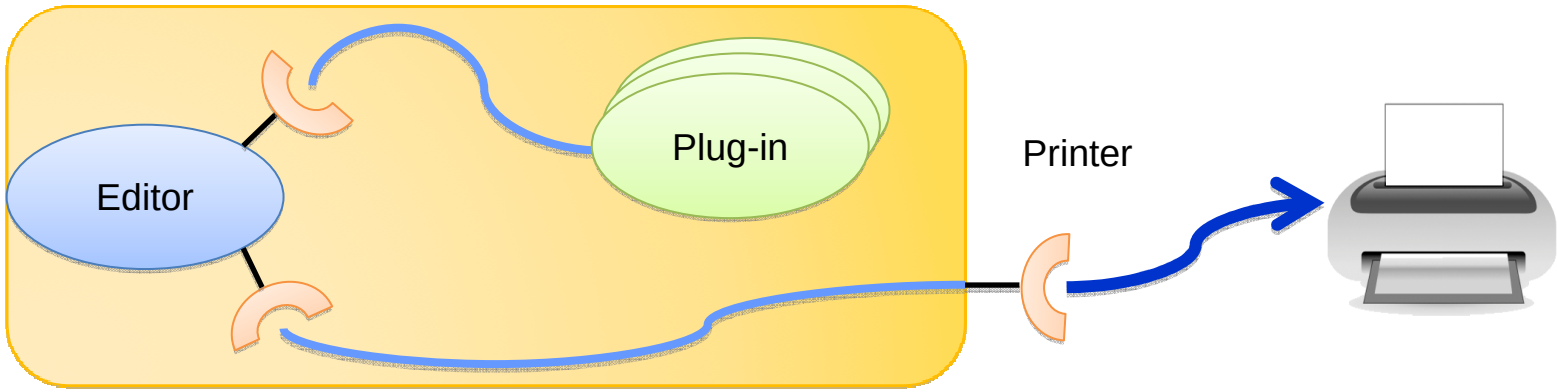
- Required Service Specifications
  - Instantiated and Imported
- Provided Service Specifications
  - Exported and *Implemented*
- Component Types



from Clément Escoffier' thesis defense

iPOJO Composite Example

```
<composite name=“Editor1">
<subservice action="instantiate“
      specification=“…Editor“/>
<subservice action="instantiate“
       specification=“… Plugin“       aggregate="true" />
<subservice action="import“
       specification=“…Printer“   optional="true”/>
</composite>
```

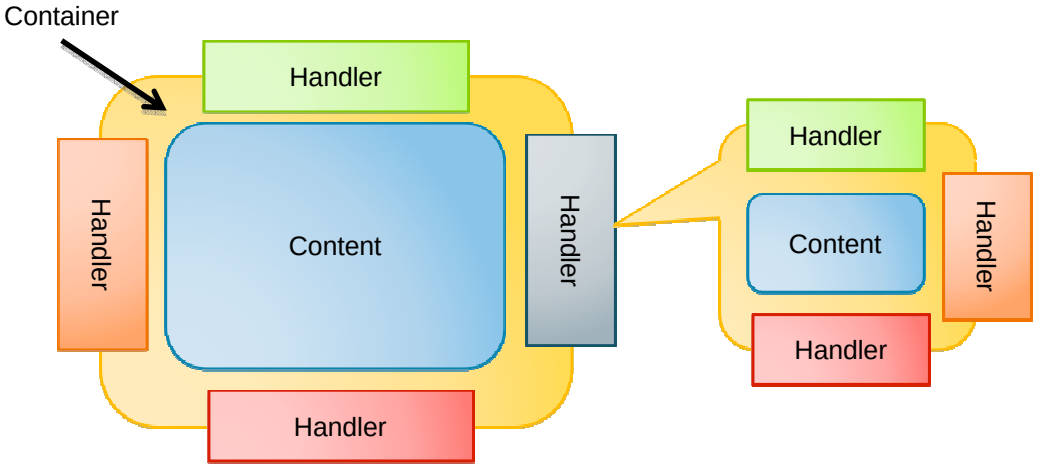


from Clément Escoffier' thesis defense

iPOJO implementation

Main features

- *Bytecode* manipulation (ASM)
- Extensible through *Handlers*
  - Handlers are iPOJO instances
  - Natively support dynamism
- Heavy use of threads and synchronization constructions
- on top of OSGi R4.0 (Felix, Equinox, KF) and various JVM 1.4, 1.5+



from Clément Escoffier' thesis defense

Spring Dynamic Modules (ex Spring OSGi)

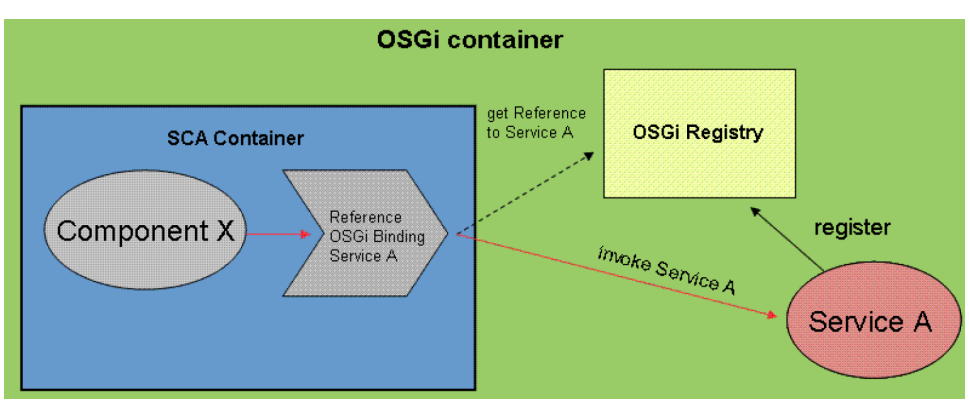
- **Rappel : Framework POJO (bean) pour les applications *server-side***
  - ***Pour les déçus des EJB2***
    - ***xml ou @nnotation 1.5***

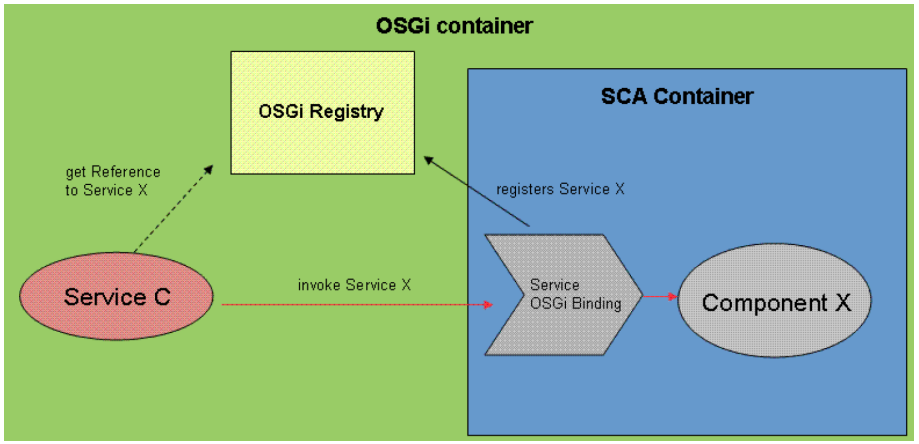
- ***Spring Framework + Beans***
  - ***Conditionnés en bundles***
  - ***Livrés sur OSGi***
  - ***+ binding à la Declarative Services (~metadata SCR)***

- ***Remark***
  - ***Focus JavaEE …***

SCA OSGi

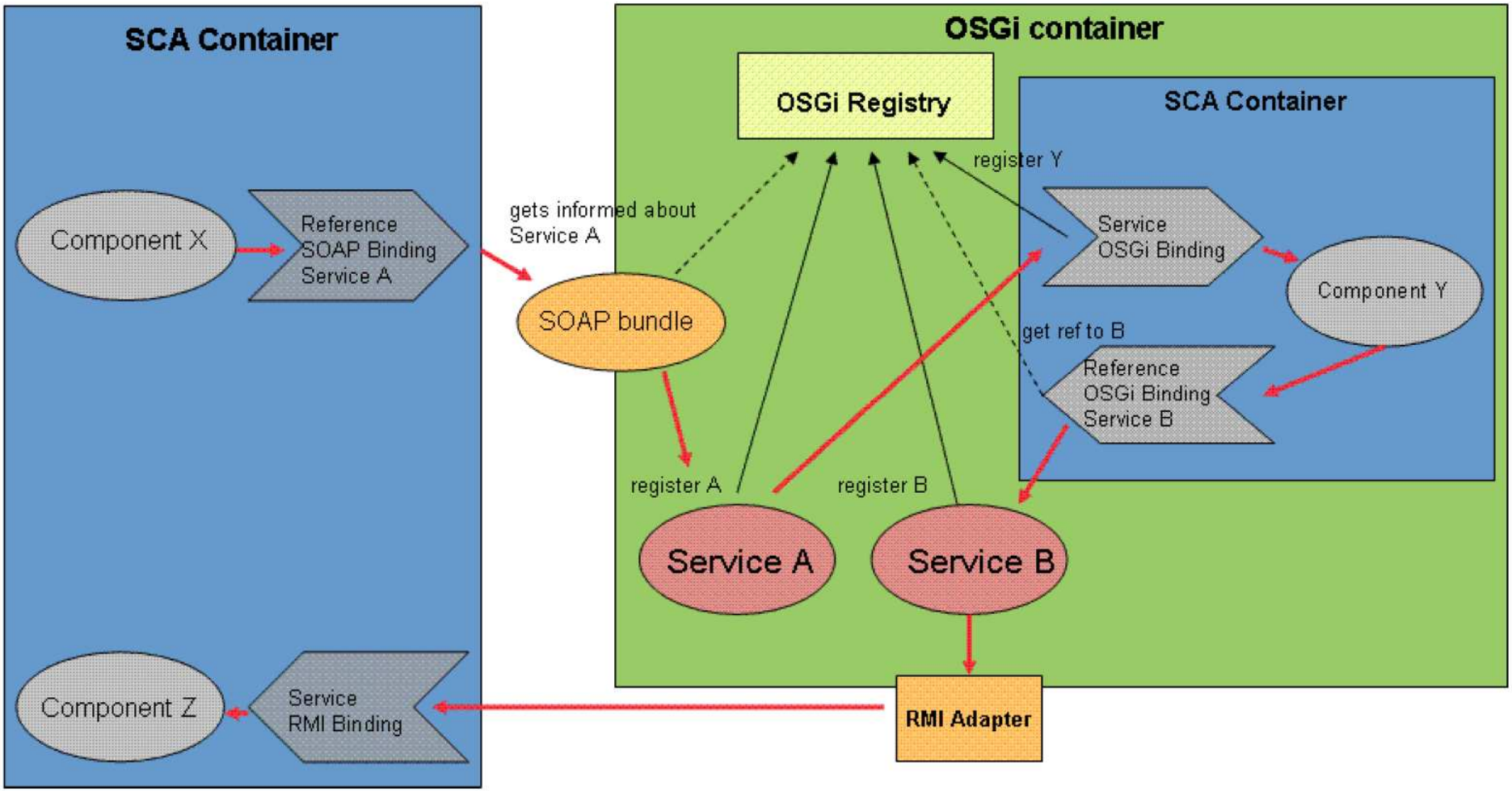
- Sujet chaud pour l'EEG
- Service Component Architecture (SCA)
  - Hierarchical component model for SO applications
    - Java, C#, BPEL, JavaScript, Python …
- SCA/OSGi
  - SCA Container packaged as OSGi bundles then deployed in a OSGi framework





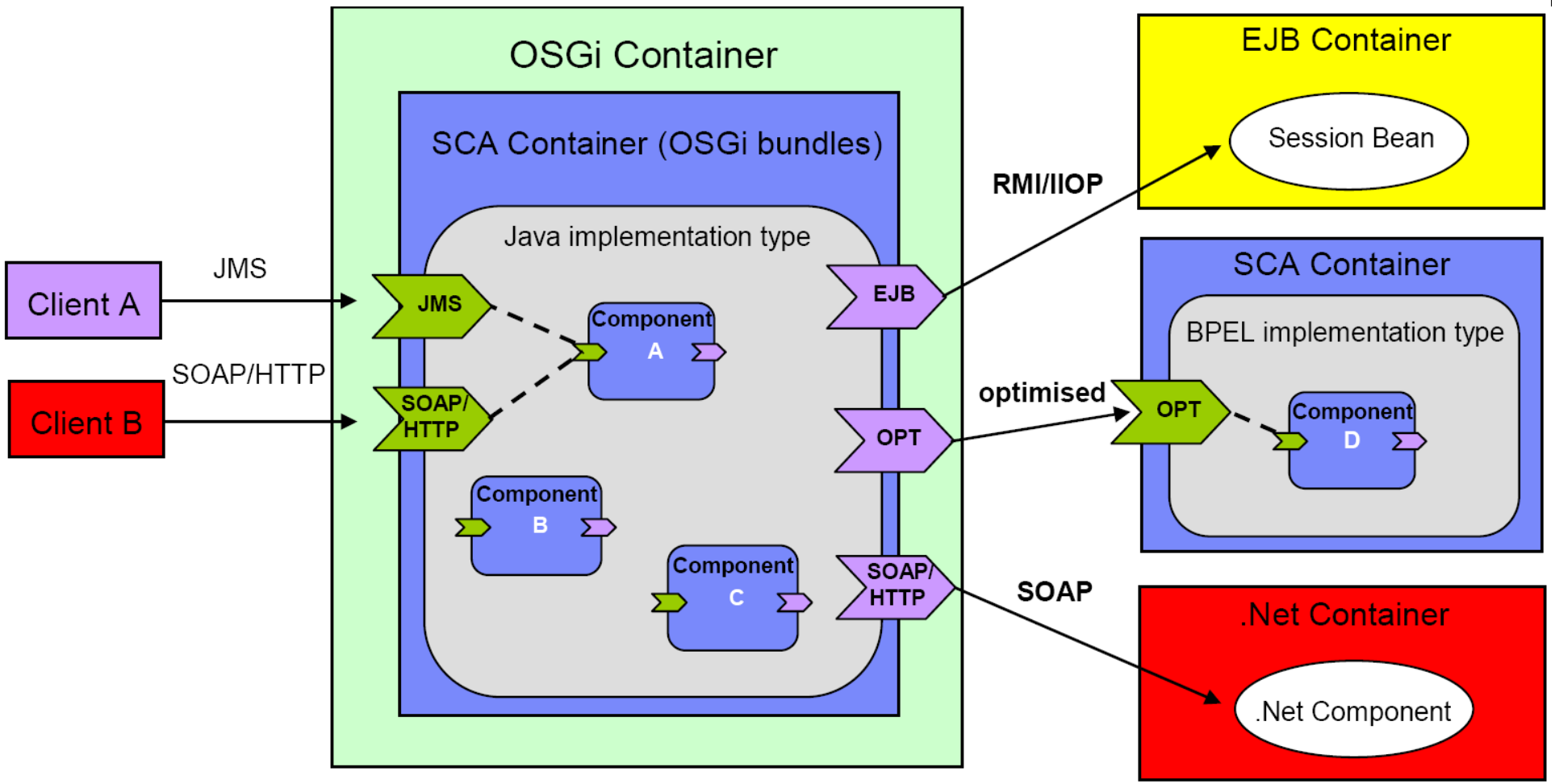
http://www.osoa.org/download/attachments/250/Power_Combination_SCA_Spring_OSGi.pdf?version=3

SCA/OSGi
Remote Invocation



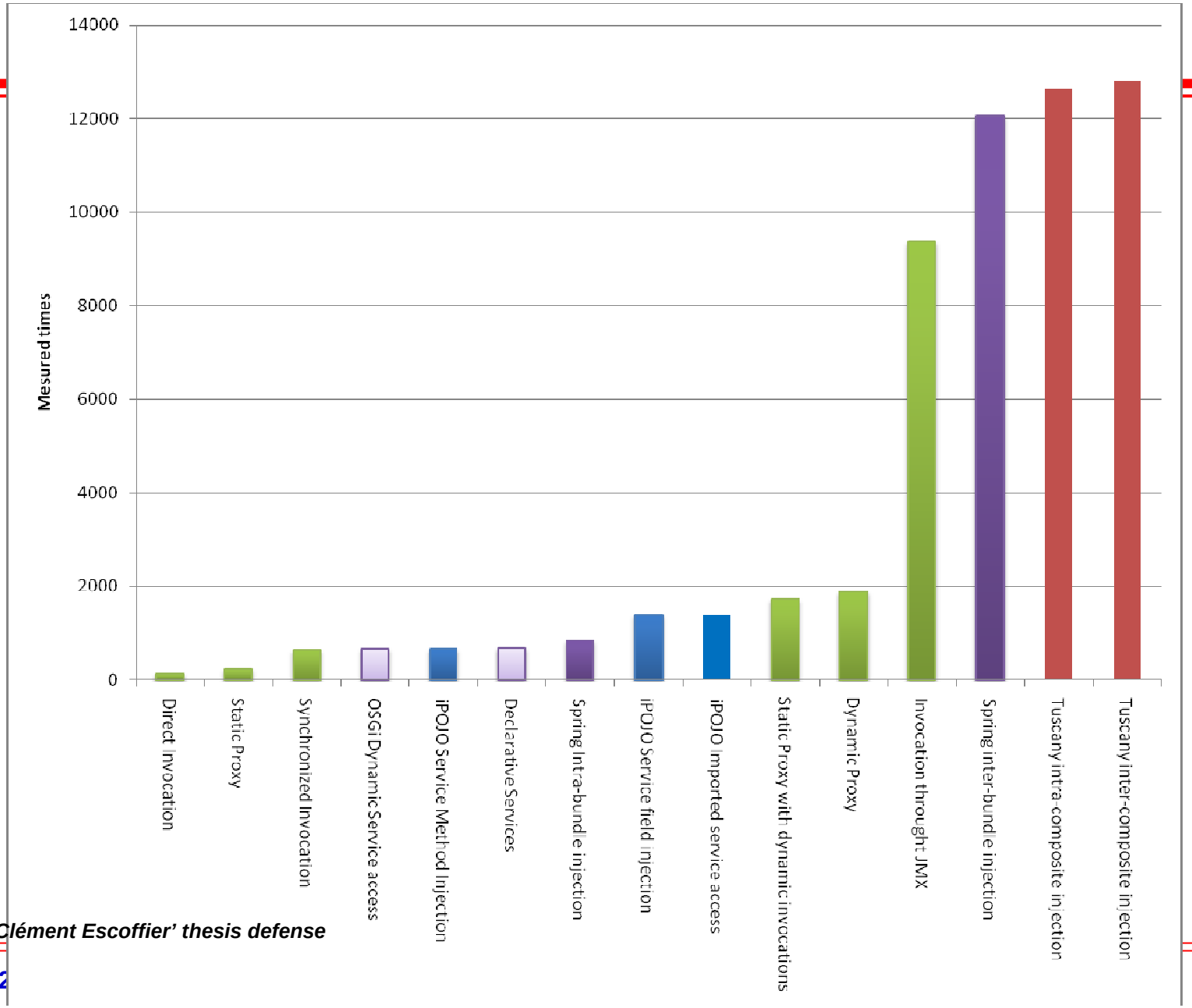
http://www.osoa.org/download/attachments/250/Power_Combination_SCA_Spring_OSGi.pdf?version=3

SCA OSGi



http://www.osoa.org/download/attachments/250/Power_Combination_SCA_Spring_OSGi.pdf?version=3

SOCA Benchmark :



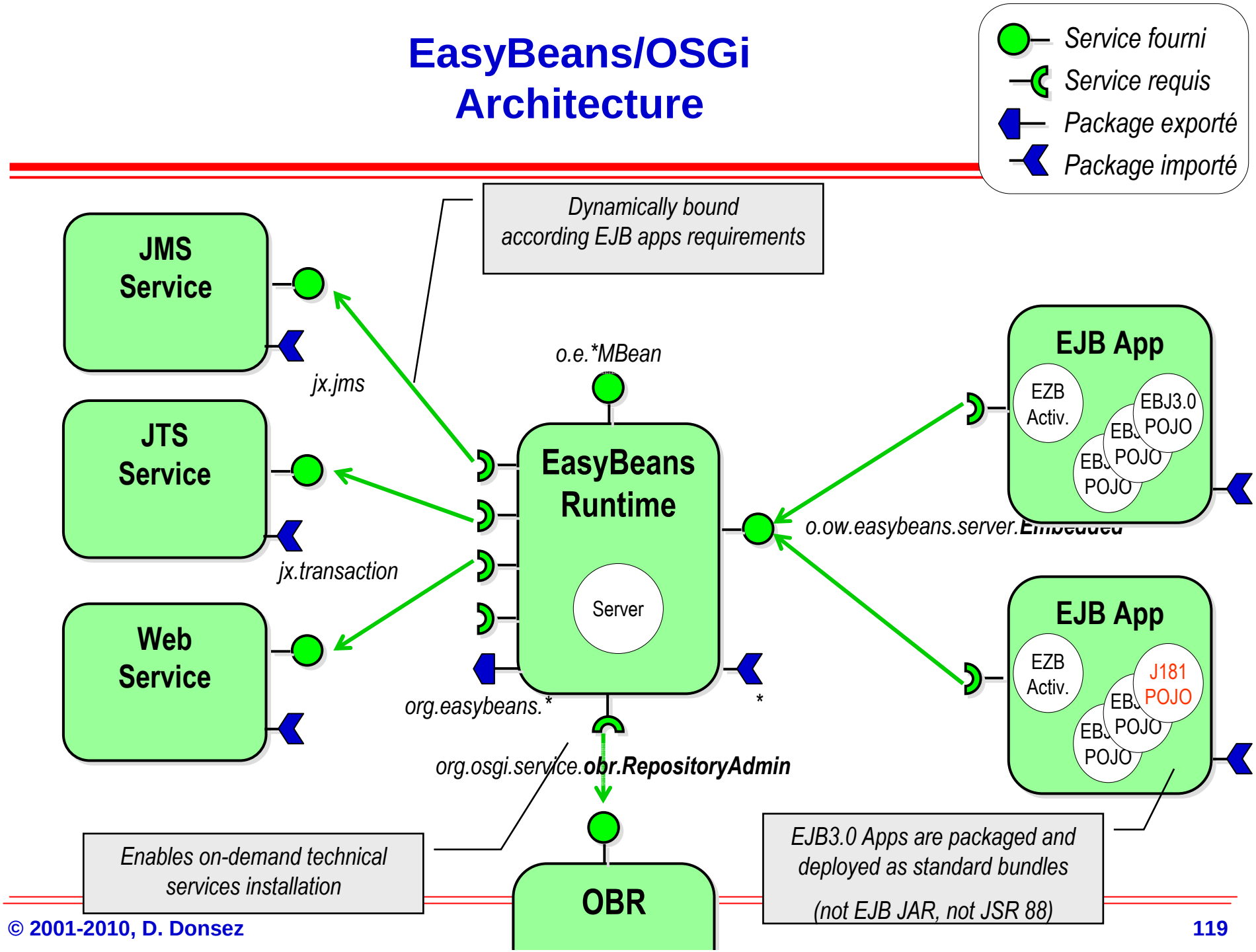
from Clément Escoffier' thesis defense

EasyBeans/OSGi

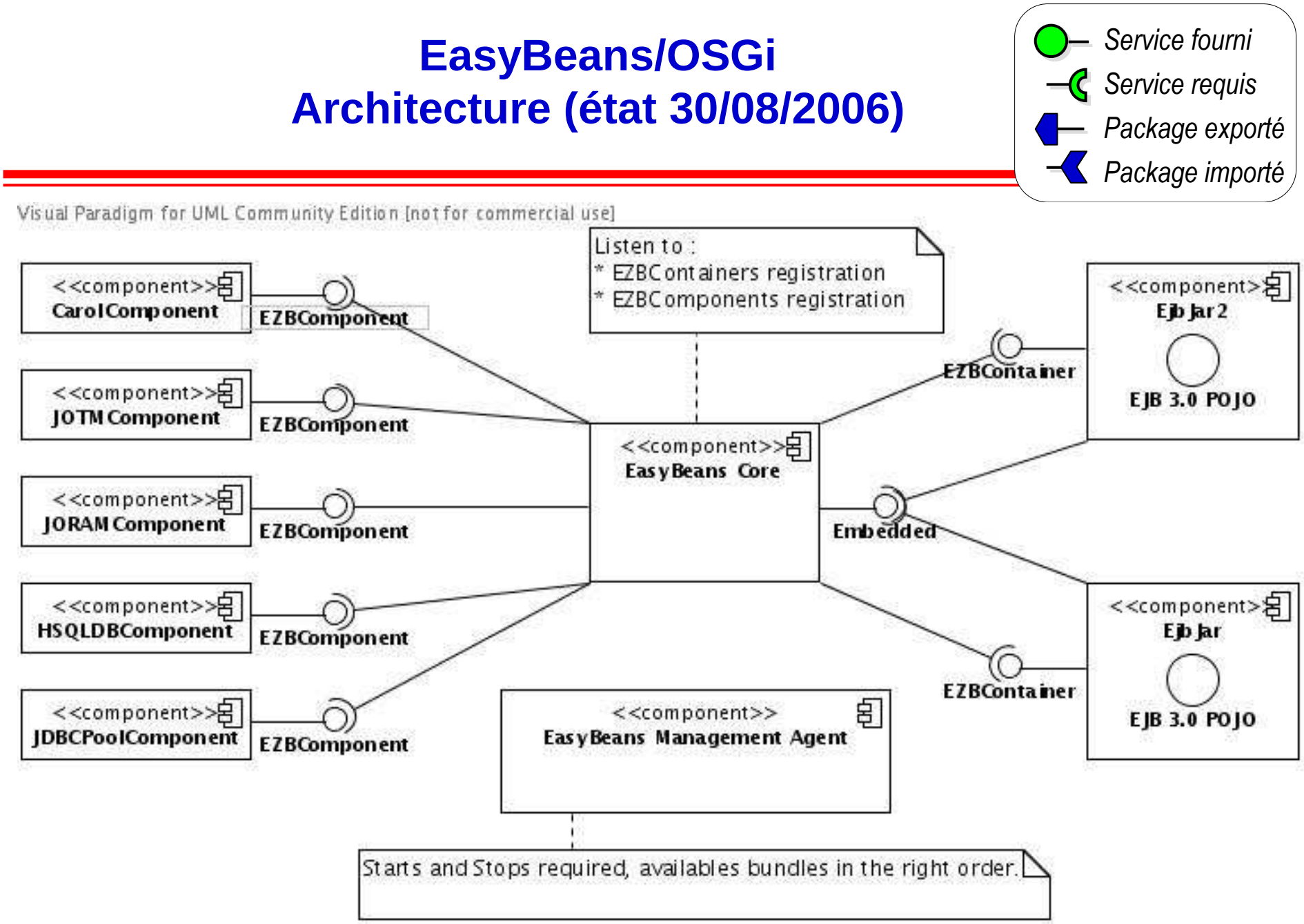
- **EasyBeans www.easybeans.org**
  - Containeur EJB3.0 – JSR 220 (annotations 5.0) + *JSR 181*
    - *JSR 220 ~= « EJB for the dummies »*
    - *JSR 181 = Web Services Metadata for the Java™ Platform*
- **Motivations**
  - Exécuter des POJOs annotés JSR-220 sur OSGi
  - Injecter le BundleContext dans les Enterprise Beans (@OSGiRessource)
    - Enregistrement de services / Utilisation de services
- **Fonctionnement**
  - 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)



Guice-OSGi

http://wiki.ops4j.org/confluence/display/ops4j/Guice-OSGi

- Google Guice : IoD injector based on Java 5 annotations
- Guice-OSGi injects required and provided services in a Java class
- Example
  - @Inject @OSGiService MyService unaryProxy;
  - @Inject @OSGiService Iterable<MyService> multipleProxy;
  - @Inject @OSGiService("(code=simple)") /* custom LDAP filter */ MyService filtered;
  - @Inject @OSGiService(interfaces = {MyService.class}) /* custom interface list */ MyOSGiListener listener;
  - @Inject @OSGiServiceRegistration MyService registeredService;
  - @Inject @OSGiServiceRegistration("lang=en,location=uk") /* custom service properties */ MyService configuredService;
  - @Inject @OSGiServiceRegistration(interfaces = {MyBaseService.class}) /* custom interface list */ MyService customizedService;
    - Registered OSGi services can be controlled using the static methods in the GuiceOSGi utility class (ie. enable/disable/modify).
  - @Inject BundleContext bc;
    - Inject the bundle context

ServiceFactory

Motivation

- Retourne une instance par bundle client
  - différentes instances pour un même service
  - Attention : ~~~ patron de conception de la fabrique (Gamma)
- Nécessaire à un singleton manipulant son ServiceRegistration

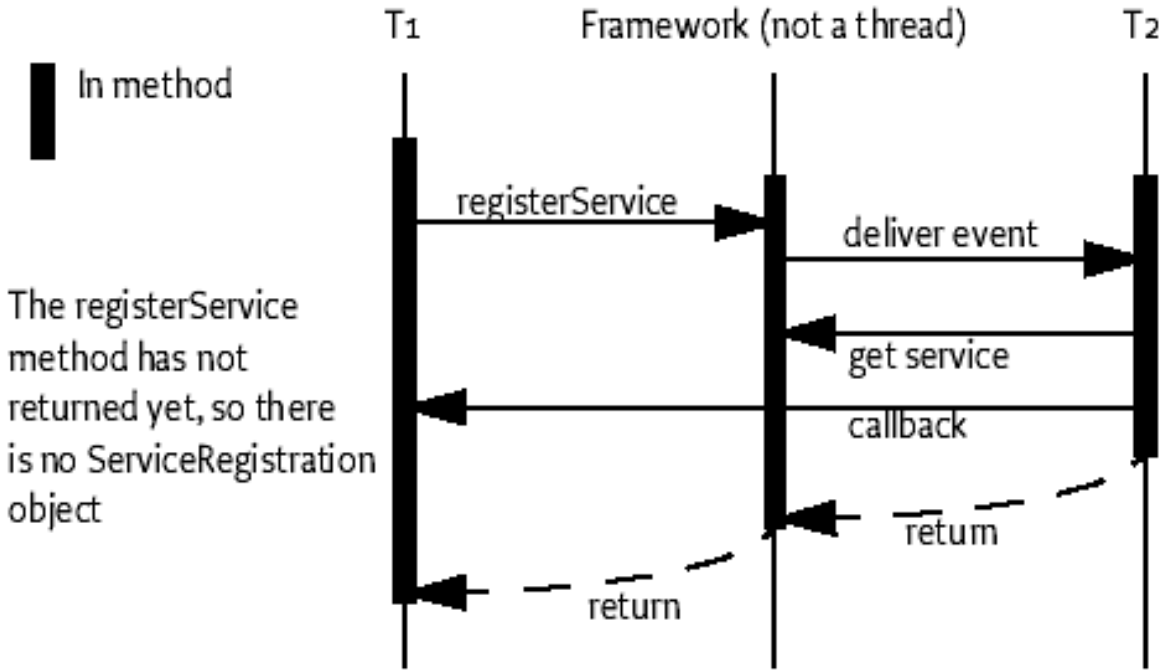
Utilisation

- Gestion de sessions multiples
  - Multi-fenêtrages
  - plusieurs shells
- Contrôle d'accès (le bundle réclamant le service est connu)
  - Contrôle des permissions
- Suivi des références « cachées »
  - Patron « Whiteboard » (chapitre suivante)

ServiceFactory

Interface à implémenter

```
interface ServiceFactory {
  Object getService(Bundle bundle, ServiceRegistration registration)
  public void ungetService(Bundle bundle, ServiceRegistration registration, Object service)
}
```



Sécurité

- Basé sur les permissions du JDK1.2
  - Le SecurityManager vérifie les permissions de chaque bundle
    - Exemple : FilePermission, DialPermission, …
  - Accès aux fichiers
    - Ceux du cache et aux ressources du bundle
- 3 permissions propres à OSGi
  - AdminPermission
    - Autorise l'accès aux fonctions d'administration du framework.
  - ServicePermission
    - Contrôle l'enregistrement et la récupération de services
  - PackagePermission
    - Contrôle l'import et l'export de packages
- org.osgi.service.PermissionAdmin
  - Service de gestion des permissions des bundles
- Conditional Permission Admin (R4)

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, …)
- Définissez le ExecutionEnvironnement
  - Et utilisez le pour la compilation !!!

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

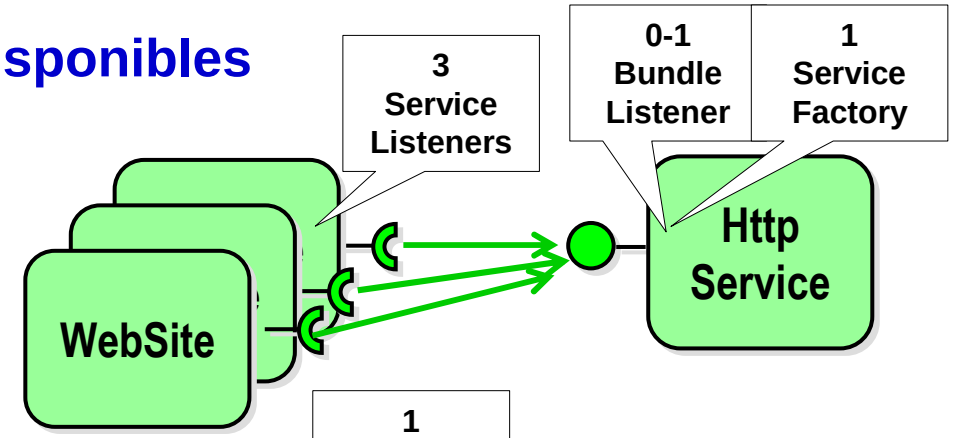
Service (i)

- 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
3. Traitement séquentiel (et synchrone) des listeners
   - Quid pour plusieurs milliers de listeners par framework

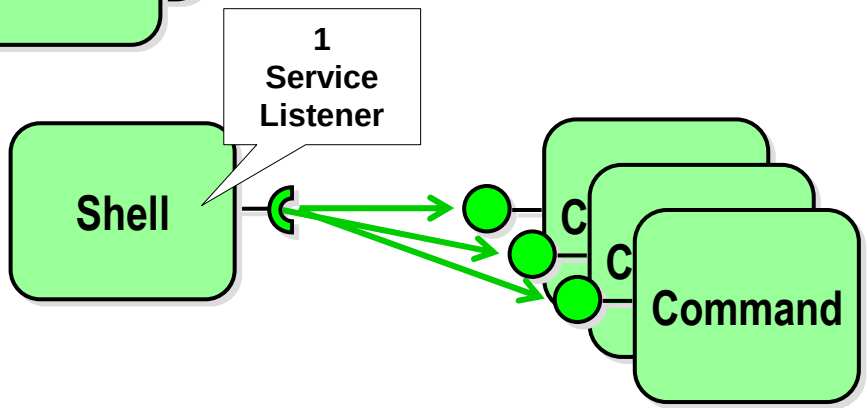
Service (ii)

- 2 patrons disponibles
- *Listener*



Exemple:
LogService,
HttpService,
…

- *Whiteboard*



Exemple:
EventAdmin,
IOConnector,
Felix' Shell,
…

- 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

Le patron par extension (Extender)

- Limite l'usage des services (listeners)
- Principe
  - 1 BundleListener scrute les bundles contenant des metadatas particulières (WEB-INF/web.xml, OSGI-INF/component.xml, plugin.xml …)
  - et instancie des objets depuis le ClassLoader du bundle scruté lors du démarrage, dépose les objets crées
- Exemples
  - SCR, iPOJO, Spring DM, Eclipse Extension Point Registry…

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);

r4

Exemples

- ProActive, Julia (FROGi), EasyBeans, …

Cycle de vie

Activation

- 1 sec par bundle
  - → 1 minute pour 60 bundles → 5 minutes pour 300 bundles → …
- Evitez des activateurs qui durent longtemps

Solutions

- Rendre la main au canevas le plus vite
  - Utilisez l'eventing
  - Ou une thread en arrière plan
- Préférez l'activation paresseuse (R4.1 *lazy activation*)
  - *lazyness is goodness*

Cycle de vie

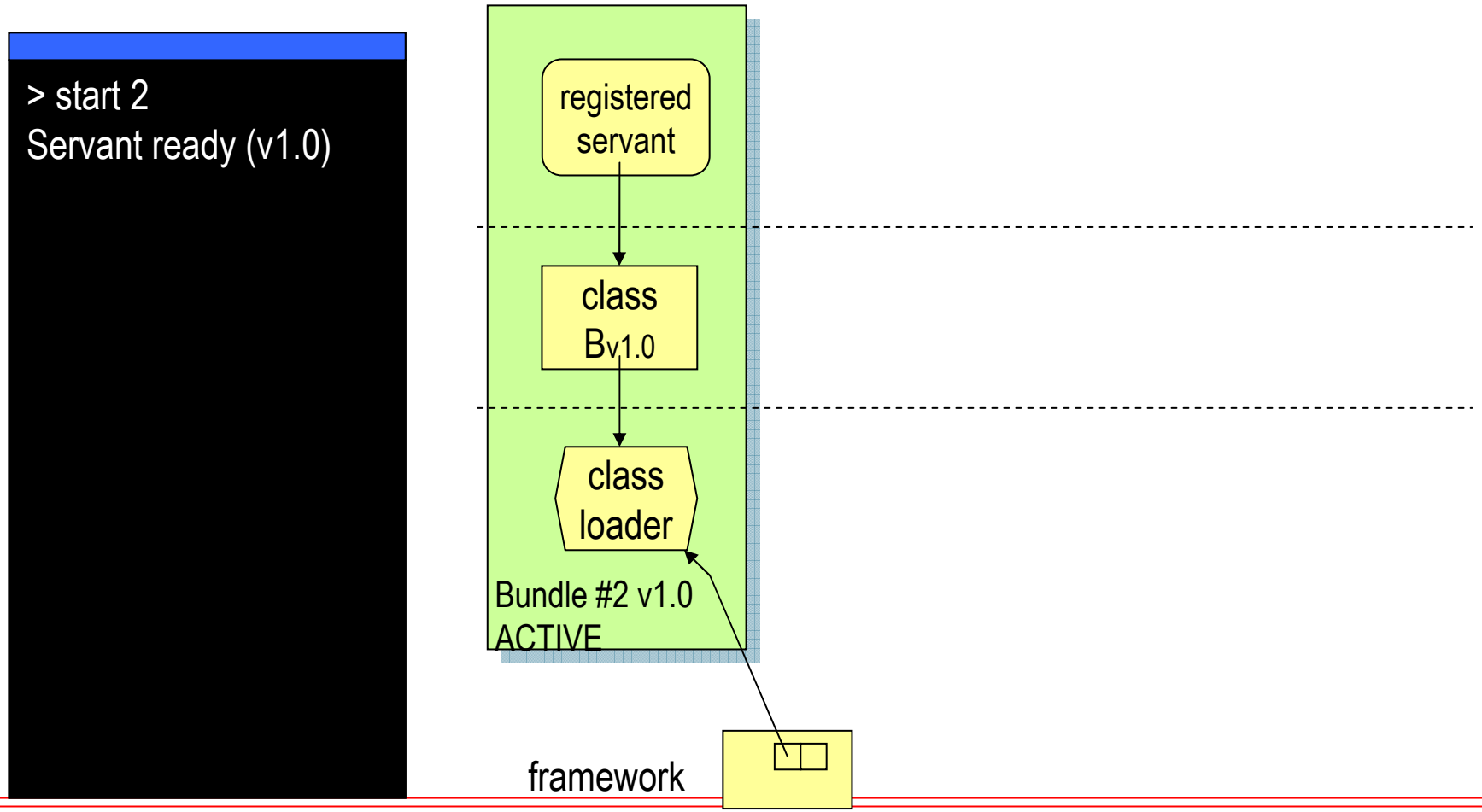
- Vous développez le start()
- Pensez aussi au stop()
  - Proscrire System.exit()
    - Votre bundle n'est pas le seul sur la JVM
    - il y en a d'autres qui bossent
  - Libérez les ressources
    - Fermez les fichiers, *sockets*, connections, …
  - Arrêtez les *threads* démarrés
    - Mieux encore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

What are Stale References?

"a reference to a Java object that belongs to the class loader of a bundle that is stopped or is associated with a service object that is unregistered"

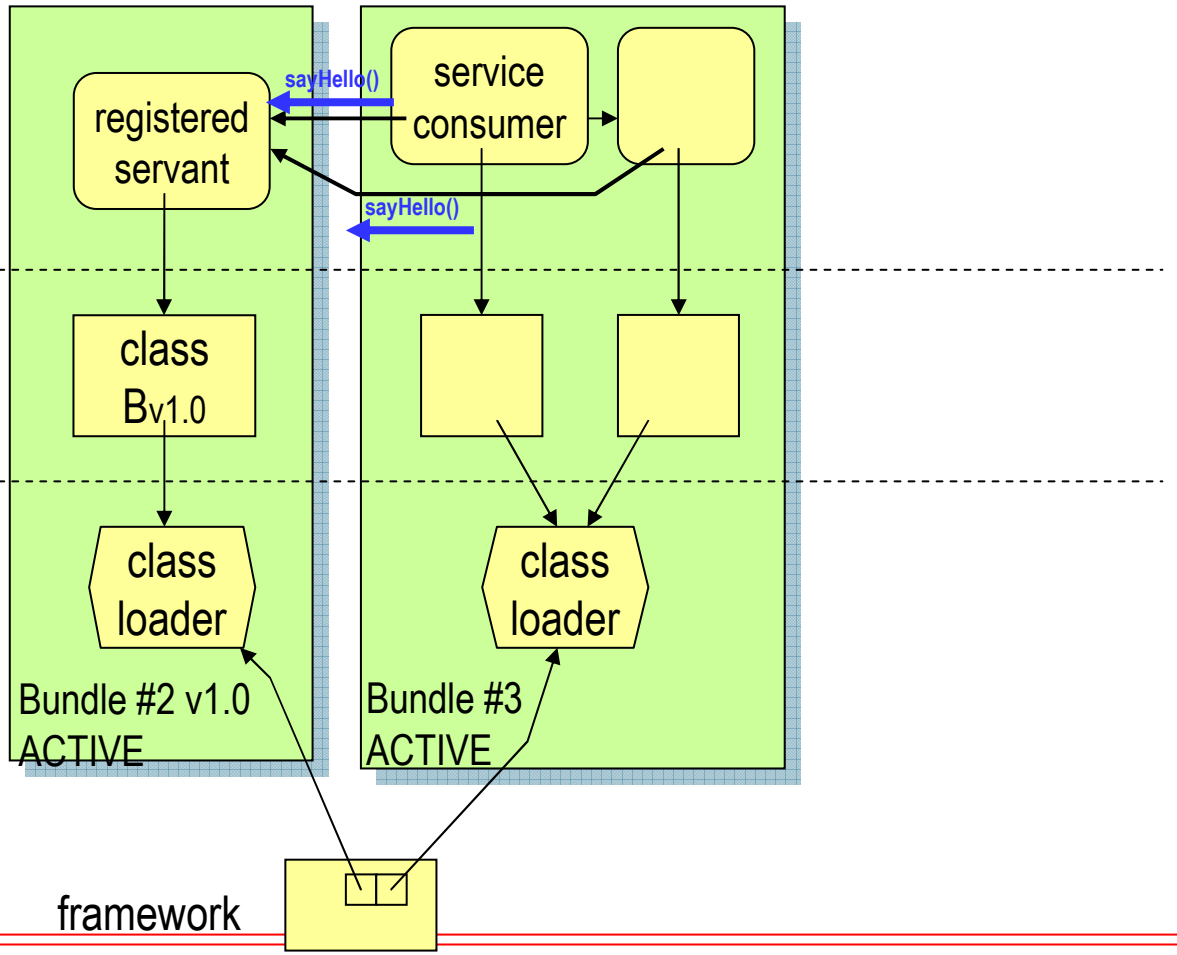
OSGi R4 Section 5.4

An example of Stale Reference Pathology?
(i) initial

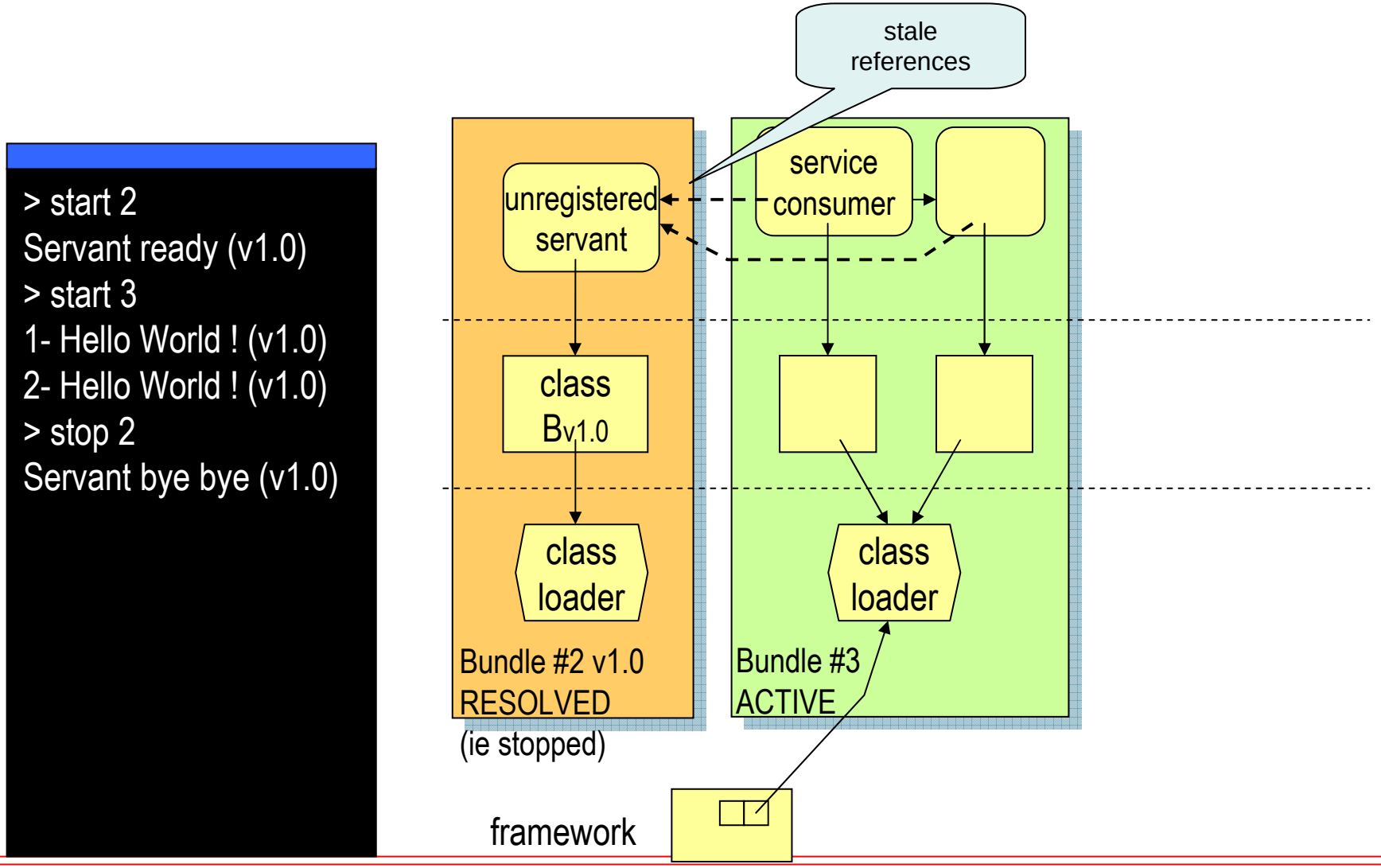


An example of Stale Reference Pathology?
(i) initial

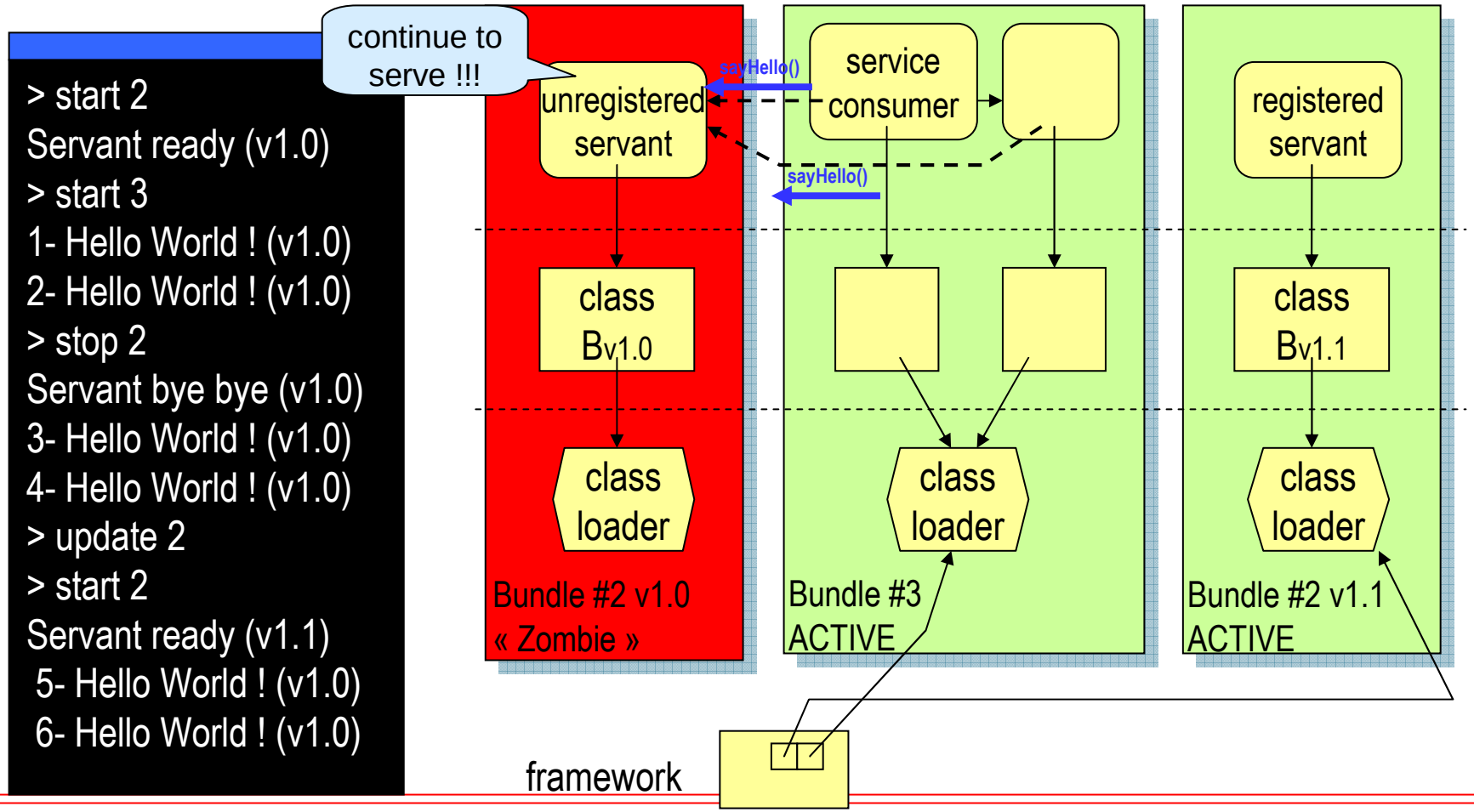
```
> start 2
Servant ready (v1.0)
> start 3
1- Hello World ! (v1.0)
2- Hello World ! (v1.0)
```



An example of Stale Reference Pathology?
(ii) After stop 2



An example of Stale Reference Pathology?
(iii) After update 2 & start 2



Bad Consequences

- **Memory leaks**
  - Retention of the classloader of a stopped or uninstalled bundle
  - Retention of all java.lang.Class loaded by that bundle
- **Utilization of invalid services → Inconsistencies!**
  - Service is unregistered but still used (wrong!)
  - Its context is most likely inconsistent
    - e.g. closed connections, old date
  - Possible exceptions upon service calls
    - good because we can see the problem
  - Silent propagation of incorrect results (worst case!)
    - E.g. Returning old cached-data

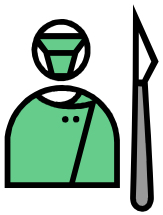
Other « stale » pathologies

- “Forwarded references”
  - From one bundle to another
- “Stale” threads
  - bundle has stopped but created threads has not
- Unregistered MBeans, RemoteObjects, …
- Unreleased resources
  - sockets, file descriptors, locks, …
- Stale ExtensionPoints
  - Eclipse sugests to restart after updating ! ;-(

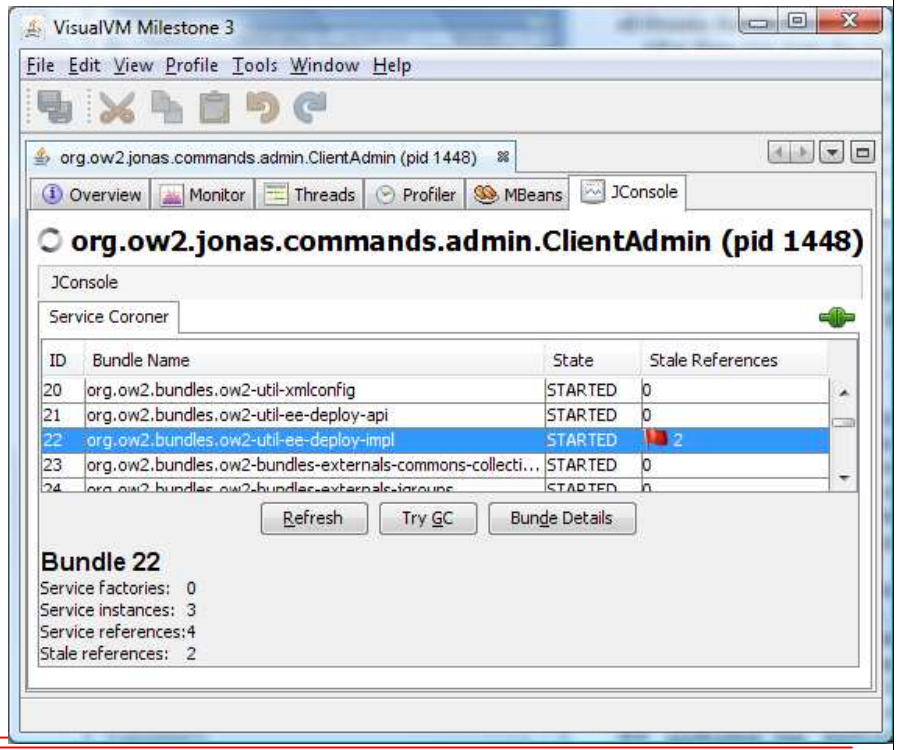
How to ensure « stale reference free » applications?

- 2 cases of OSGi™ applications
- From-scratch OSGi™ development
- Bundlization of Legacy codes
  - Really frequent
  - Module with or without Services/Extension Points
- Good OSGi™ programming practices
  - Who trusts their developers ?
- Component Models
  - Necessary but not enough
- Stale references may be there but we can't see them…
- → We need Diagnosis
  victim bundles x guilty bundles

The ServiceCoroner tool



- A diagnostics tool for detecting stale references in OSGi™ applications
- “Inspector” of services death
- Runtime diagnosis
- Points out victim bundles/services and possible suspects



*The coroner is a legal examiner that investigates the causes of unnatural deaths in English speaking countries. Not all coroners have forensic pathology knowledge, but for illustration purposes we have named our tool as ServiceCoroner.

Stale References are not a myth !

Experiment results (May 2008)

I	**OSGi-based software**	JOnAS (JavaEE server)	SIP Comm. (multiprotocol VoIP and Chat UA)	Newton (SCA container)	Sling (Content Repository)
II	**Version**	**5.0.1**	**Alpha 3**	**1.2.3**	**2.0 incubator snapshot**
III	**OSGi Impl.**	**Felix 1.0**	**Felix 1.0**	**Equinox 3.3.0**	**Felix 1.0**
IV	**Bundles using Component Models**	**20 iPOJO**	**6 Service Binder**	**0**	**18 Declarative Services**
V	**Lines of Code**	**Over 1 500 000**	**Aprox. 120 000**	**Aprox. 85 000**	**Over 125 000**
VI	**Total Bundles**	**86**	**53**	**90**	**41**
VII	**Initial No. of Service Refs.**	**82**	**30**	**142**	**105**
VIII	**No. of Bundles w/ Stale Svcs.**	**4**	**17**	**25**	**2**
IX	No. of Stale Services Found	7	19	58	3
X	No. of Stale Threads	2	4	0	0
XI	Stale Services Ratio (IX/VII)	8.5 %	63 %	40.8%	2.8%

[1] Actually the whole Newton implementation is an SCA constructed on top of OSGi, but its bundles did not use an OSGi component model like the other analyzed applications did.

Plus

A lire

- BJ Hargrave & Peter Kriens, « OSGi Best Practices! », Session TS 1419, 2007 JavaOne Conference et OSGi Users Community Event 2007
  - http://www2.osgi.org/wiki/uploads/Conference/OSGiBestPractices.pdf
- « Extensions vs Services: Digging Deeper », OSGi Users Community Event 2007
  - http://www2.osgi.org/wiki/uploads/Conference/NeilBartlett.pdf

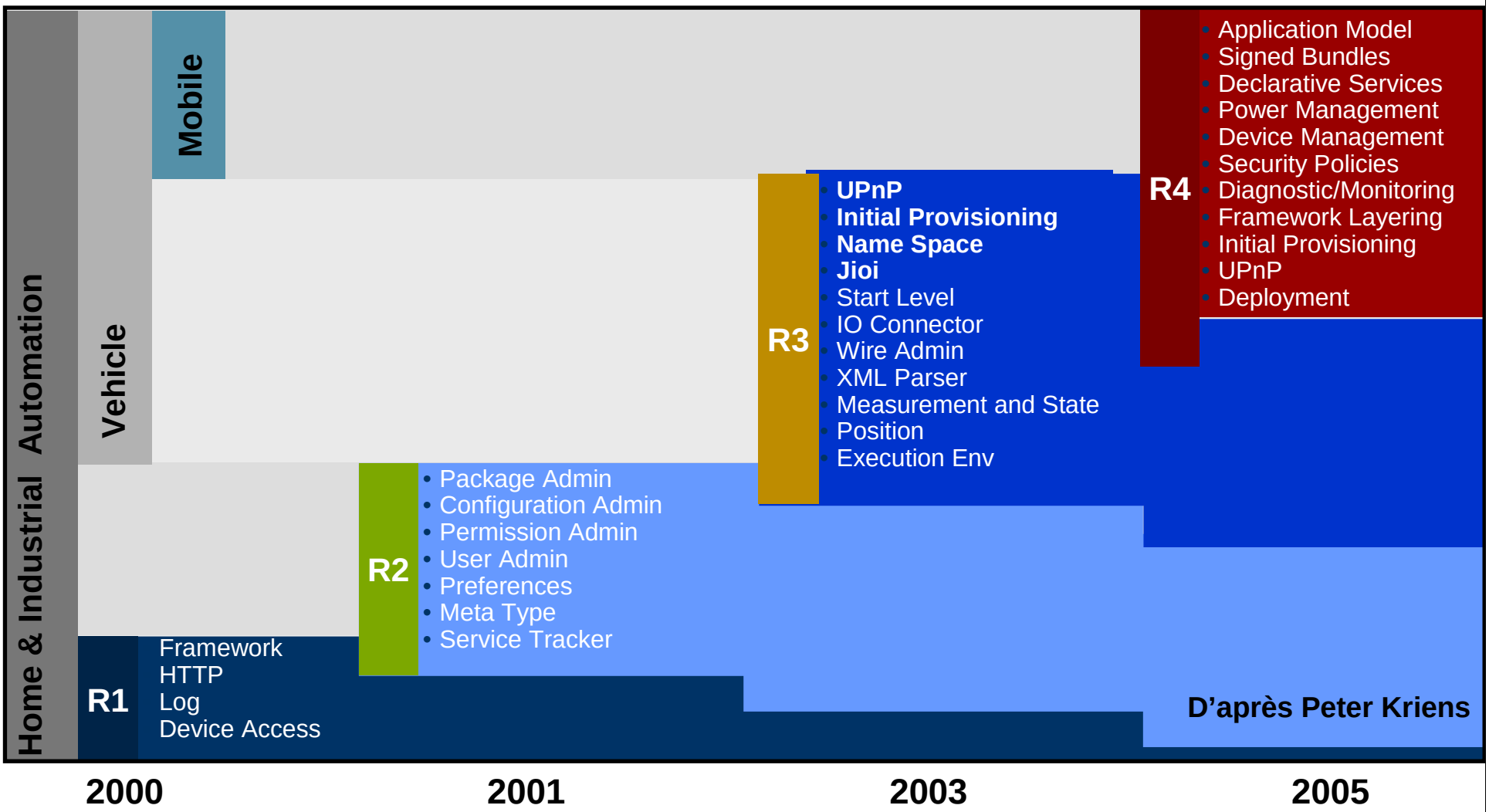
Autres

- **Les outils arrivent …**
  - PDE, Management (Console)
- **Les recettes arrivent …**
  - Migration vers OSGi en 2 étapes
    - Modularisation
      - Ecueil : ClassLoader, Thread context switch, Ressources, JNDI, static, default packages (includes for rsc.), …
    - (D)SOA ou Extension Points
  - Exemple personnel :
    - Tomcat (dans JOnAS), JacORB, …

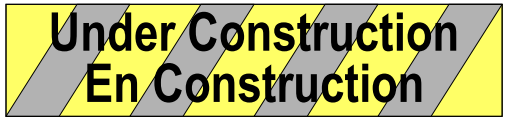
OSGi

Les services standard

Les services OSGi standard



SystemBundle



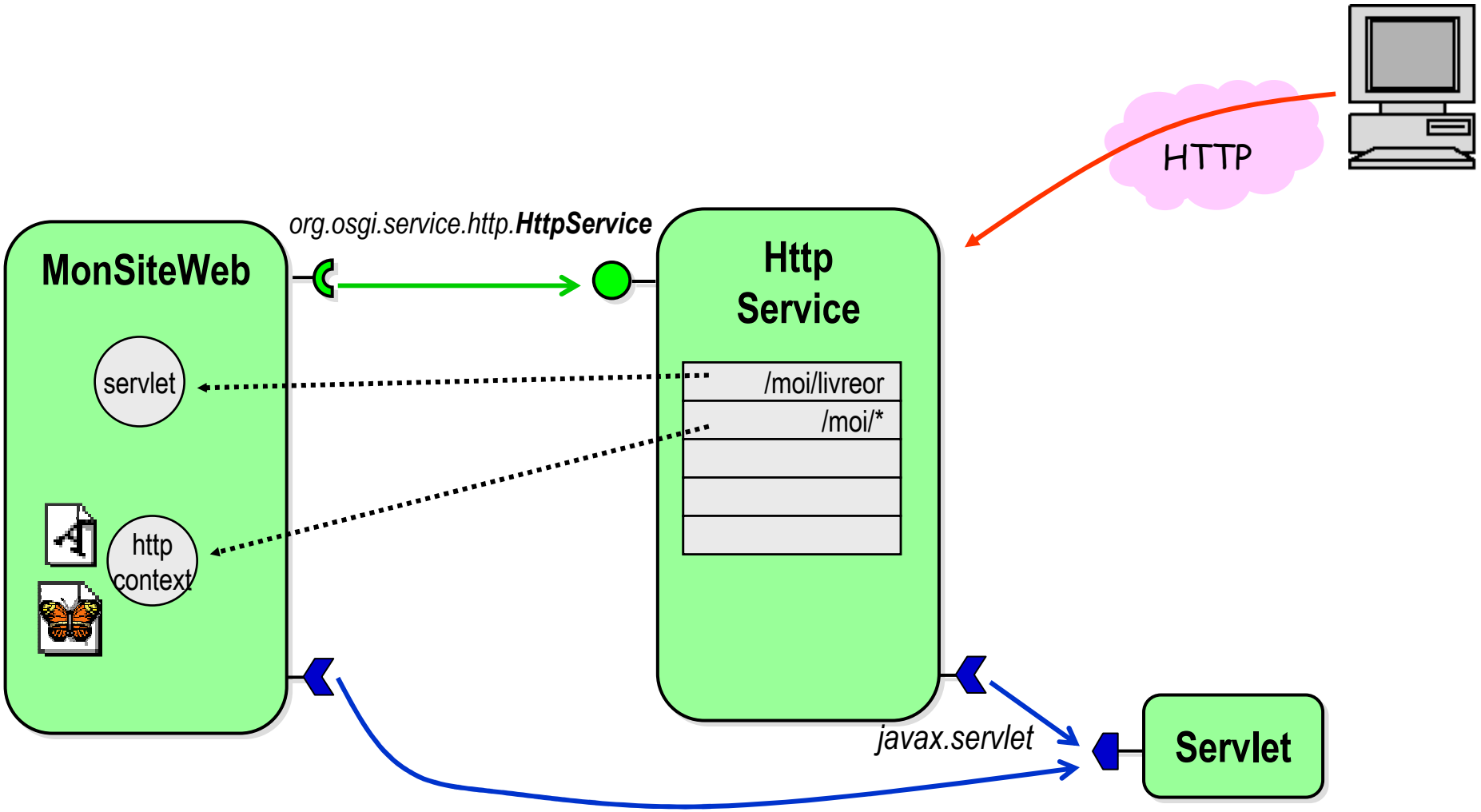
- Représente le framework. Son identifiant est toujours 0.
- Son cycle de vie correspond au démarrage et à l'arrêt du Framework

HttpService
org.service.http.HttpService (i)

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, RESTFul, …
- Extra
  - <jspc> pour éviter d'embarquer un compilateur de JSP
  - Convertisseurs WAR to Bundles
  - Canevas Web (Cocoon, Wicket, DysoWeb …)

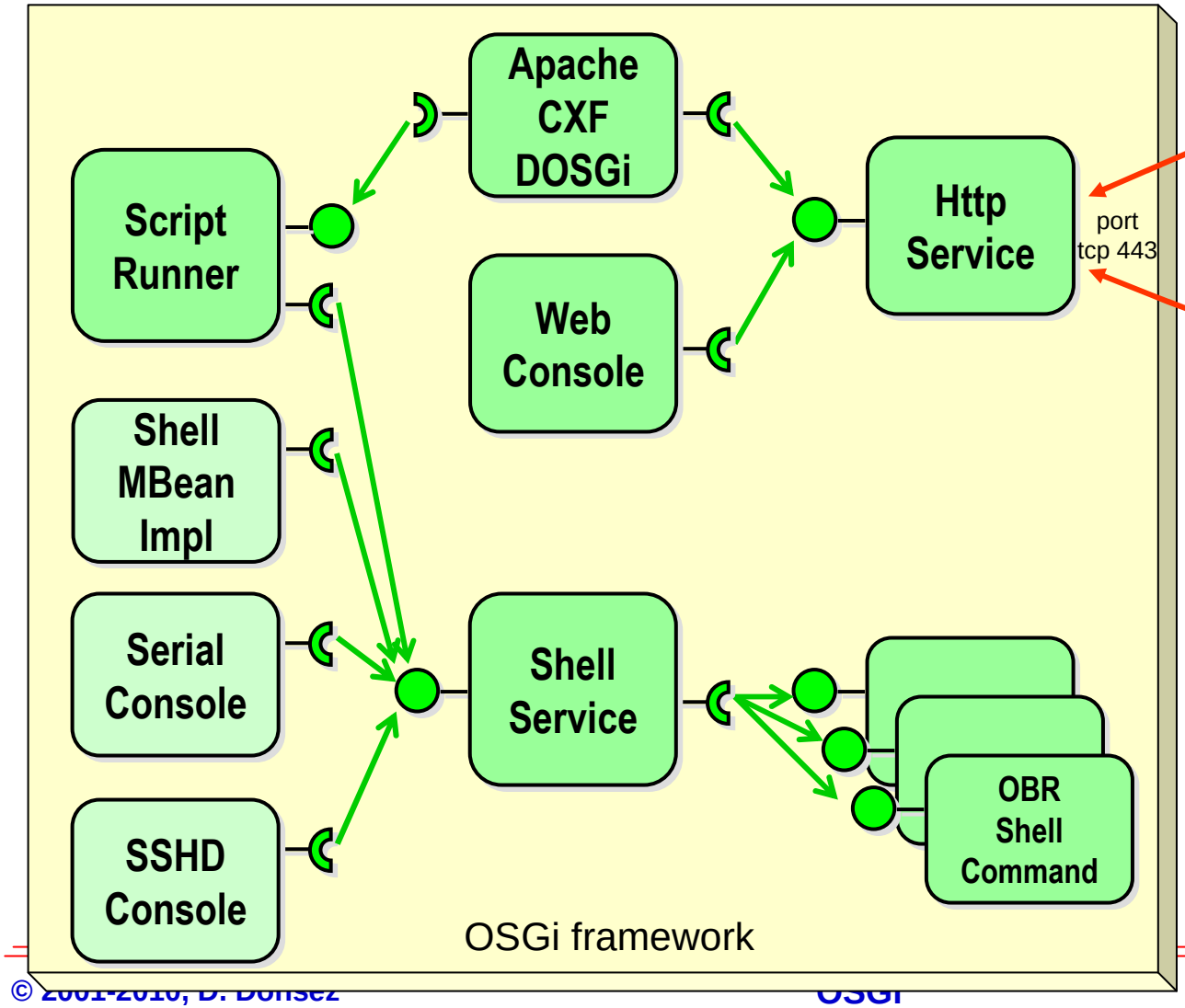
org.service.http.HttpService (ii)
Usage

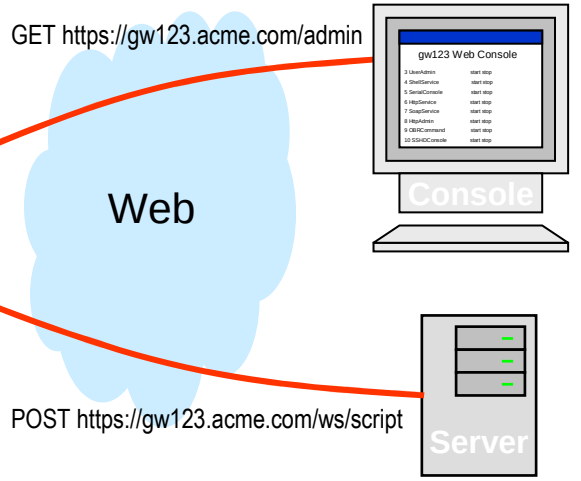


org.service.http.HttpService (iii)
Exemple d'activateur d'un site Web

```
HttpService https= ...;
String WEBROOT = "/webroot"; // embedded ressources in BUNDLE-CLASSPATH jarfiles
String WEBROOT_ALIAS = "/moi";
String SERVLET_ALIAS = WEBROOT_ALIAS +"/livreor";
Servlet servlet=new LivreOrServlet(param1,param2);
https.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;
  }
https.registerResources(WEBROOT_ALIAS, WEBROOT, docsContext );
```

Exemple d'architecture pour un shell



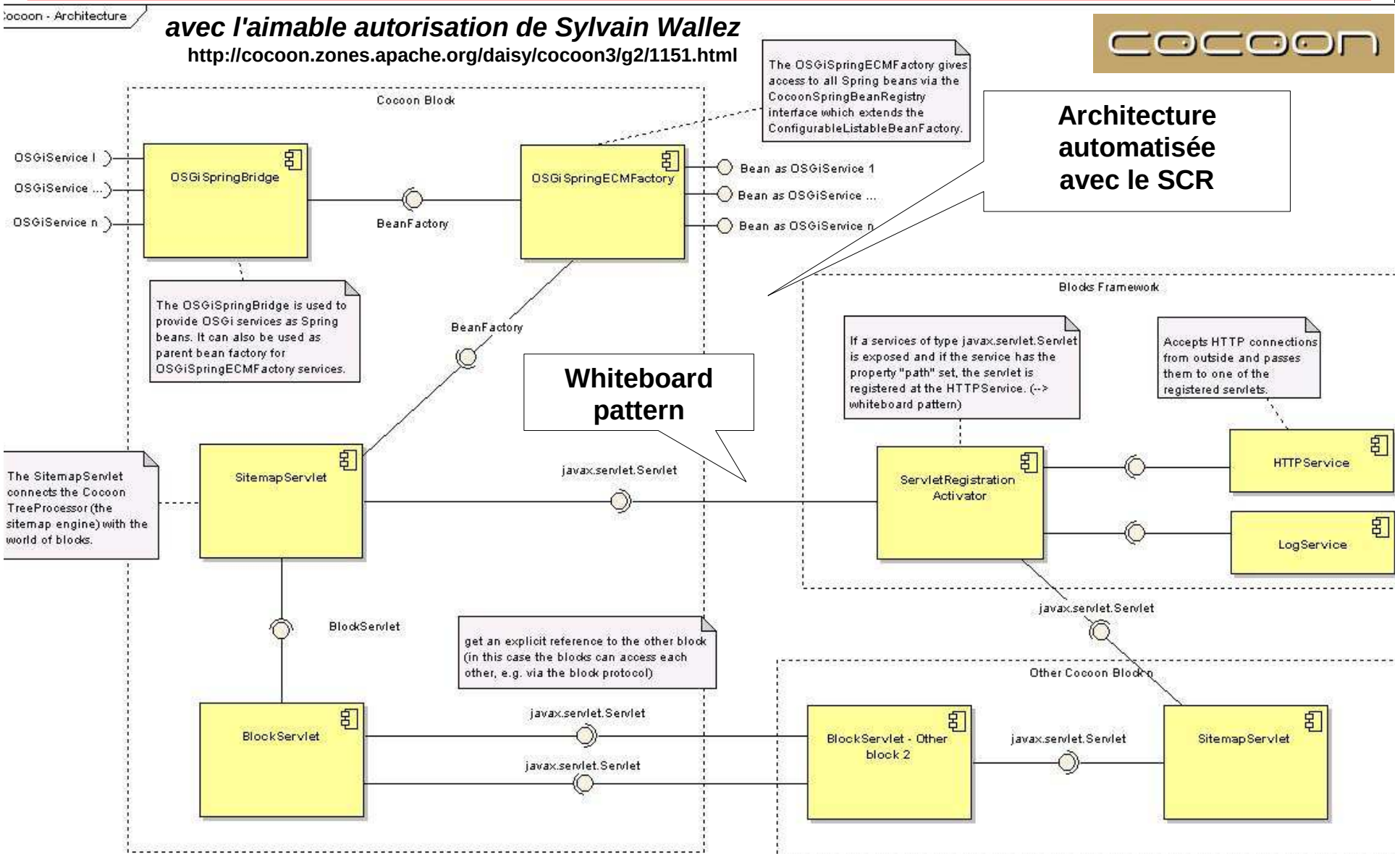



```
POST /ws/script  HTTP/1.1
...
<SOAP-ENV:Envelope xmlns:SOAP-ENV=...>
 <SOAP-ENV:Body>
  <s:ExecuteScript xmlns:s=...>
   <s:script SOAP-ENC:arrayType="s:cmd[3]">
    <cmd>obr install "SensorLoggerWebSvc"</cmd>
    <cmd>install http://bundleserver/dep.jar</cmd>
    <cmd>install http://bundleserver/main.jar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

org.osgi.service.http Limitations

- Motivations
  - org.osgi.service.http is limited to javax.servlet v2.1, portlet JSR 168
    - No filter chain, no listener, basic JSP, no TagLib, …
  - Hidden org.osgi.service.http stuff to JEE developers
  - Keep compatibility with standard webapps (.war)
- Propositions
  - Cocoon
  - Wicket
  - DysoWeb
    - Dynamic sub-webapps
    - Embed Felix in a War : subwebapps are deployed by Felix FW
    - Tested with Tomcat
  - HttpRegistry
    - Server-side equinox http://www.eclipse.org/equinox/server/
    - Embed Equinox in a War
    - Use extender model
    - Convertor GWT (Google Web Toolkit) module → 2 OSGi bundles
    - Tested with Tomcat and Jetty

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**

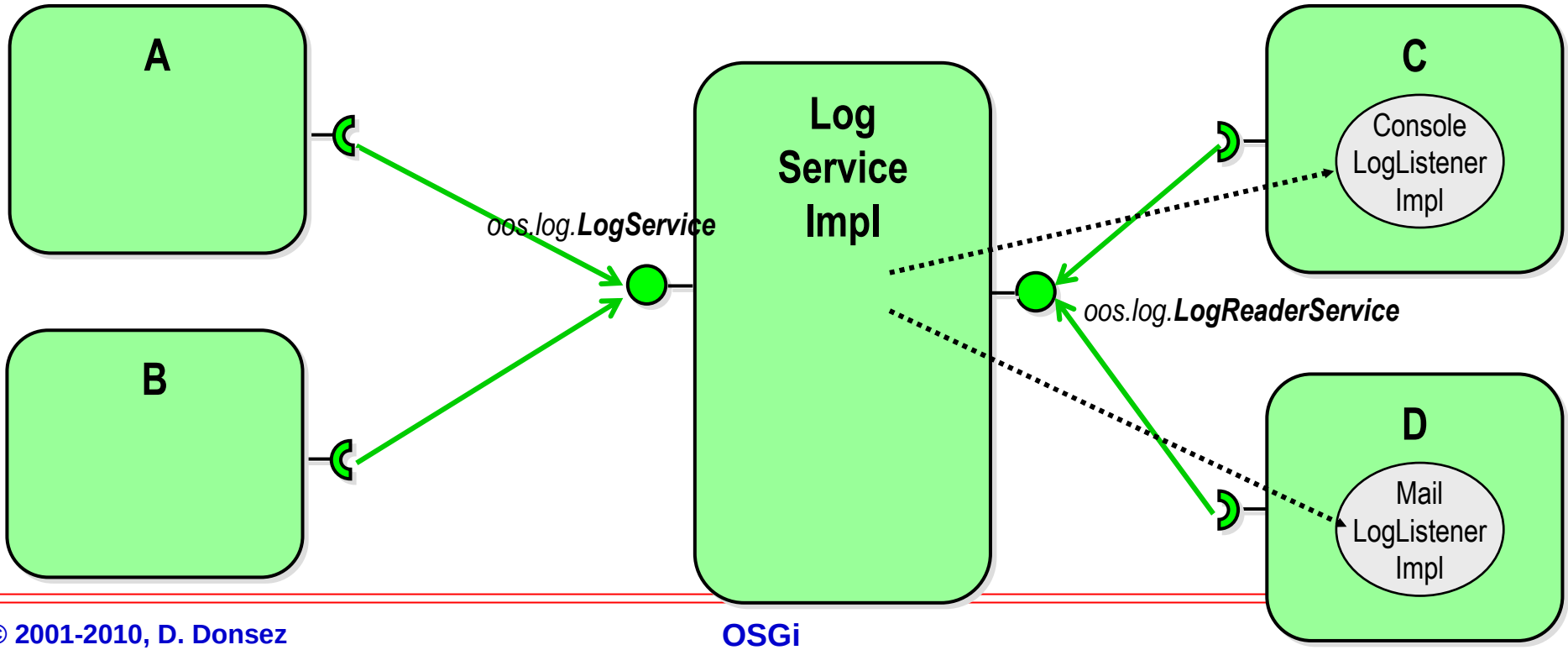
- **http://www.wicket-wiki.org.uk/wiki/index.php/OSGi**

LogService

Motivation

- journaliser des traces/événements
- se mettre en l'écoute de ces traces/événements
- Rappel: java.util.logging.Logger est statique

Architecture



Exemple de LogListener

```
public class ConsoleLogListenerActivator implements BundleActivator {
    LogReaderService logReaderService;
    LogListener  loglistener;
    public void start(BundleContext cxt) {
     logReaderService= ...;
     loglistener=new ConsoleLogListenerImpl();
     logReaderService.addLogListener(loglistener);
    }
    public void stop(BundleContext cxt) {
     logReaderService.removeLogListener(loglistener);
    }}
class ConsoleLogListenerImpl implements org.osgi.service.log.LogListener {
    public final void logged(LogEntry entry) {
                    System.out.println("Level:"+entry.getLevel());
                    if( entry.getBundle() != null) {
                                System.out.println("bundle : "+ entry.getBundle().getBundleId()+" ");
                    }
                    System.out.println(entry.getMessage());
     }}
```

Device Manager

- **Motivations**
  - Faire apparaître les drivers des périphériques matériels comme des Services OGSi
  - Charger les drivers grâce aux bundles
  - Mise à jour des drivers
  - Un driver fournit plusieurs services plus ou moins raffinés
  - Plug-and-Play
    - Le branchement d'un périphérique provoque l'enregistrement d'un service.
    - Le retrait du périphérique provoque le désenregistrement du service
- **Plusieurs éléments**
  - DeviceService
  - Driver
  - DriverLocator
  - DeviceManager

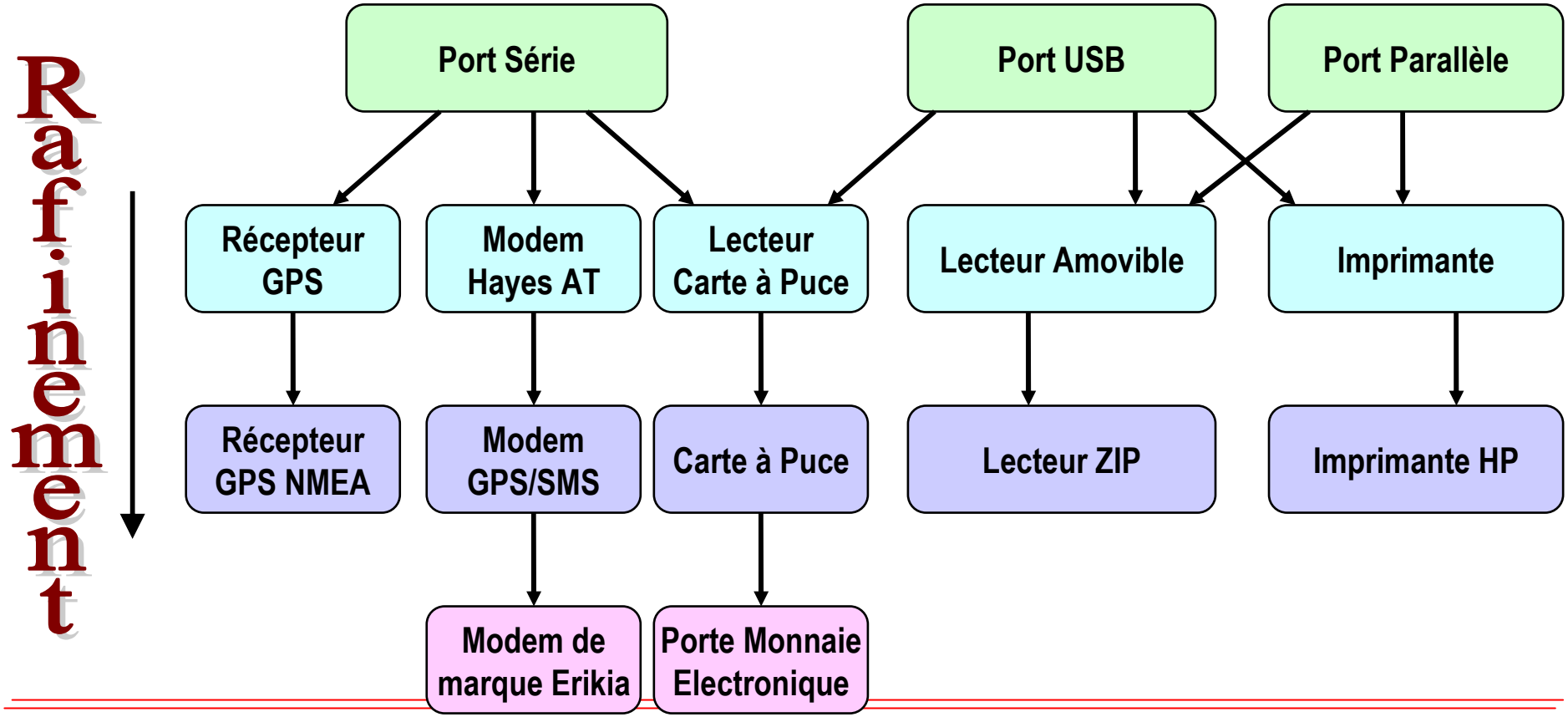
Device Manager

- **Moteur de raffinement des devices**
  - **A l'arrivée d'un nouveau service Device, le Device Manager cherche à enregistrer d'autres Device de plus haut niveau.**
- **Services utilisés**
  - **DriverLocator**
  - **Driver**

Device Manager

Device

- Gestion d'un périphérique



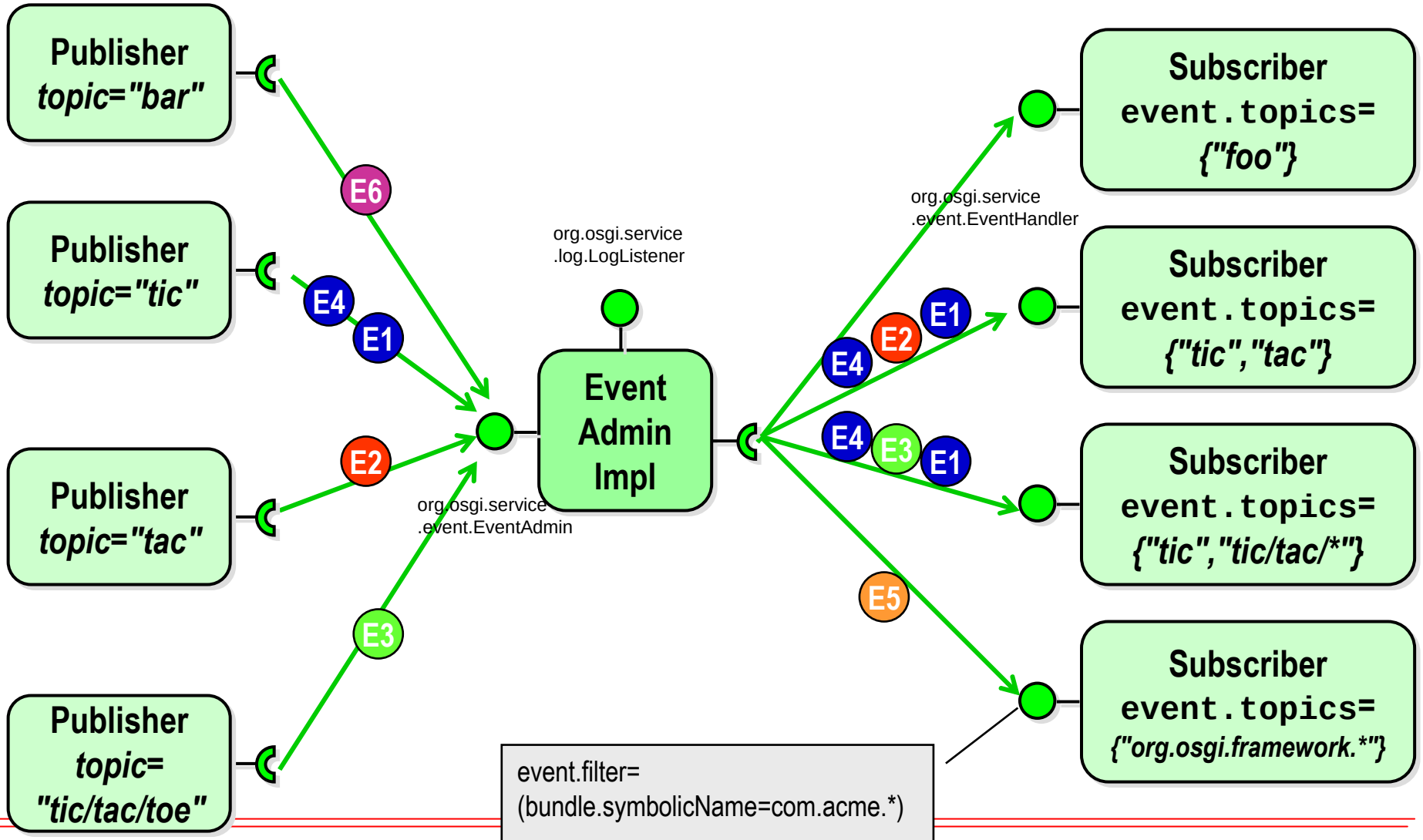
Device Manager

- **DriverLocator**
  - Permet d'associer un Device à Driver
    - Device → Id
    - Id → URL
  - Utiliser par le DeviceManager
  - Exemple d'implementation
    - Filtre LDAP → Id → URL
- **Driver**
  - Vérifie si l'on peut installer le Device associé à ce Driver
  - Retourne une valeur d'évaluation de l'association
    - Non possible si =0
  - Implémente une méthode d'attachement.

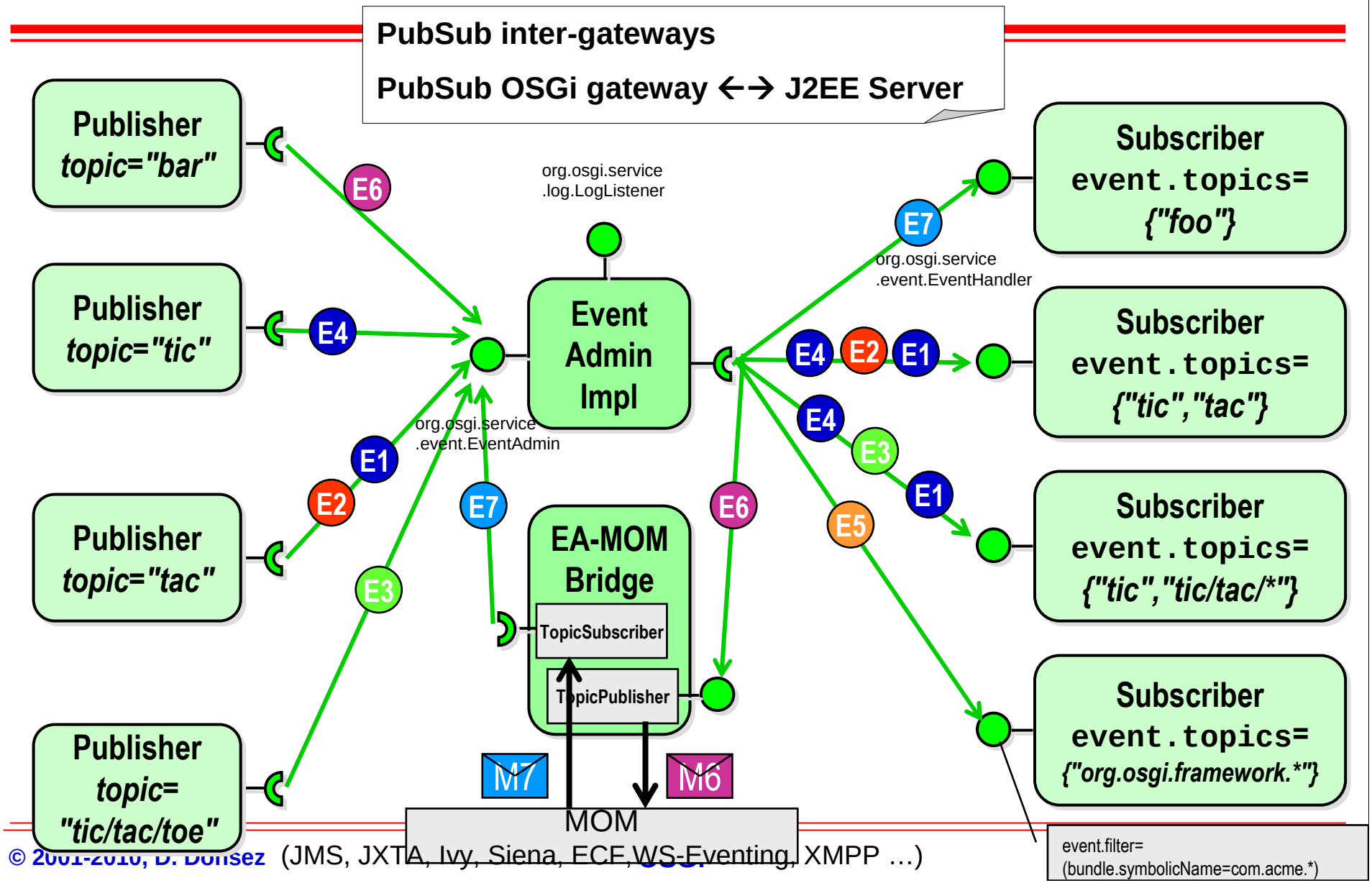
Event Admin Service (i) r4

- 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.

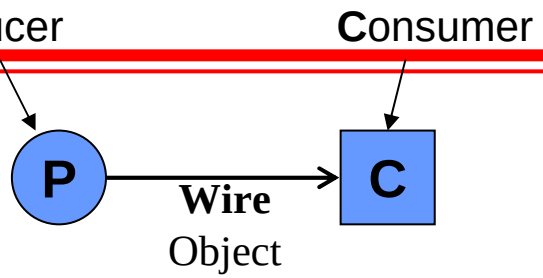
Event Admin Service (ii)



Bridging Event Admin Service and MOM



Wire Admin Service R3



Motivation

- Patron (*design pattern*)
  de services producteurs-consommateurs de données
  - Données : mesure physique, position, état (discret), …

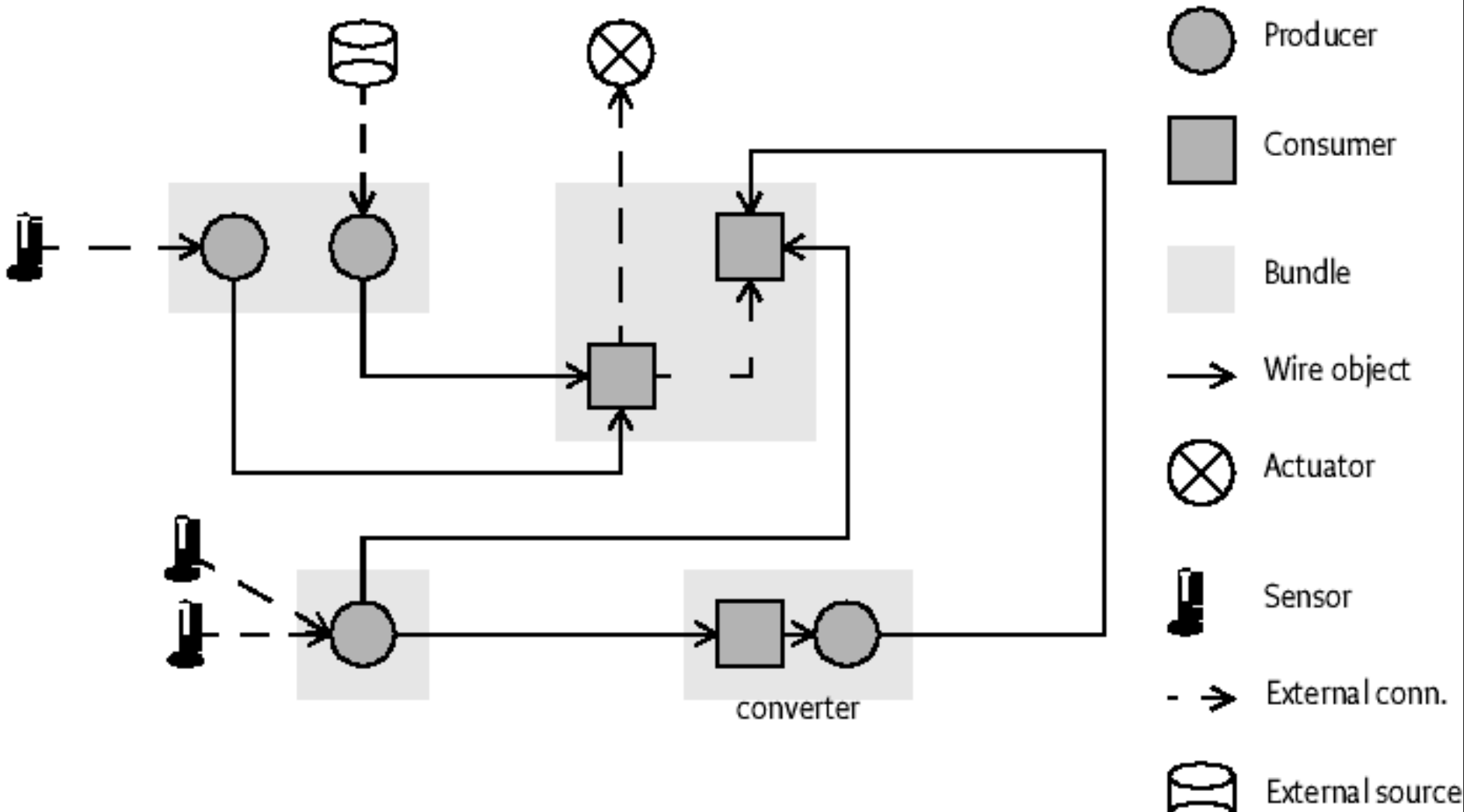
Domaine d'application

- Services basés Capteurs
- Machine-to-Machine (M2M)

WireAdmin

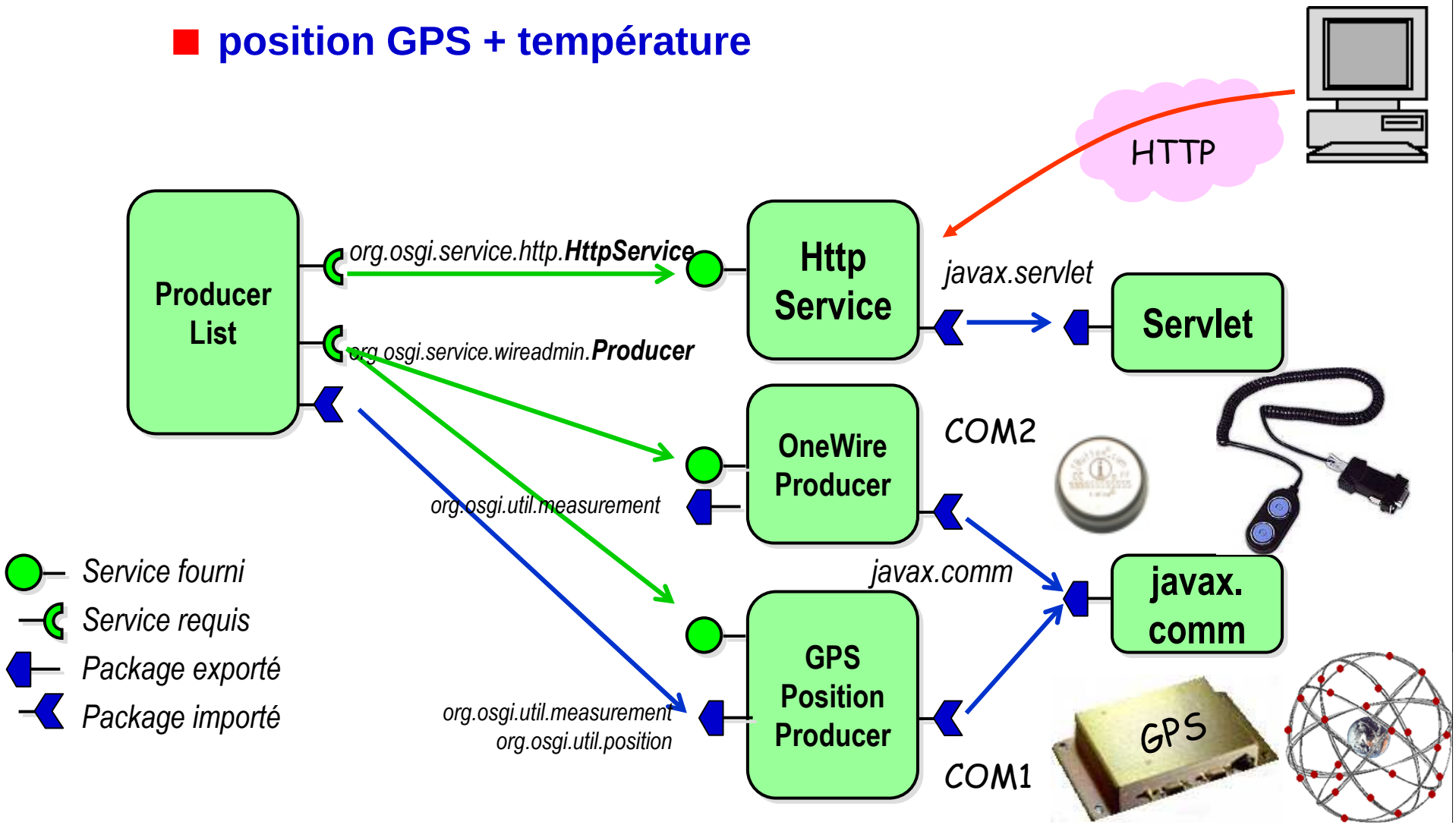
- Médiateur reliant 0..N producteurs à 0..M consommateurs
  - Administrable globalement
    - WireAdmin, WireAdminListener
- Contrôle de la « comptabilité » et adaptation
  de types de données échangées au travers du Wire
  - Flavors

Wire Admin Service
Patron *Producer-Wire-Consumer*



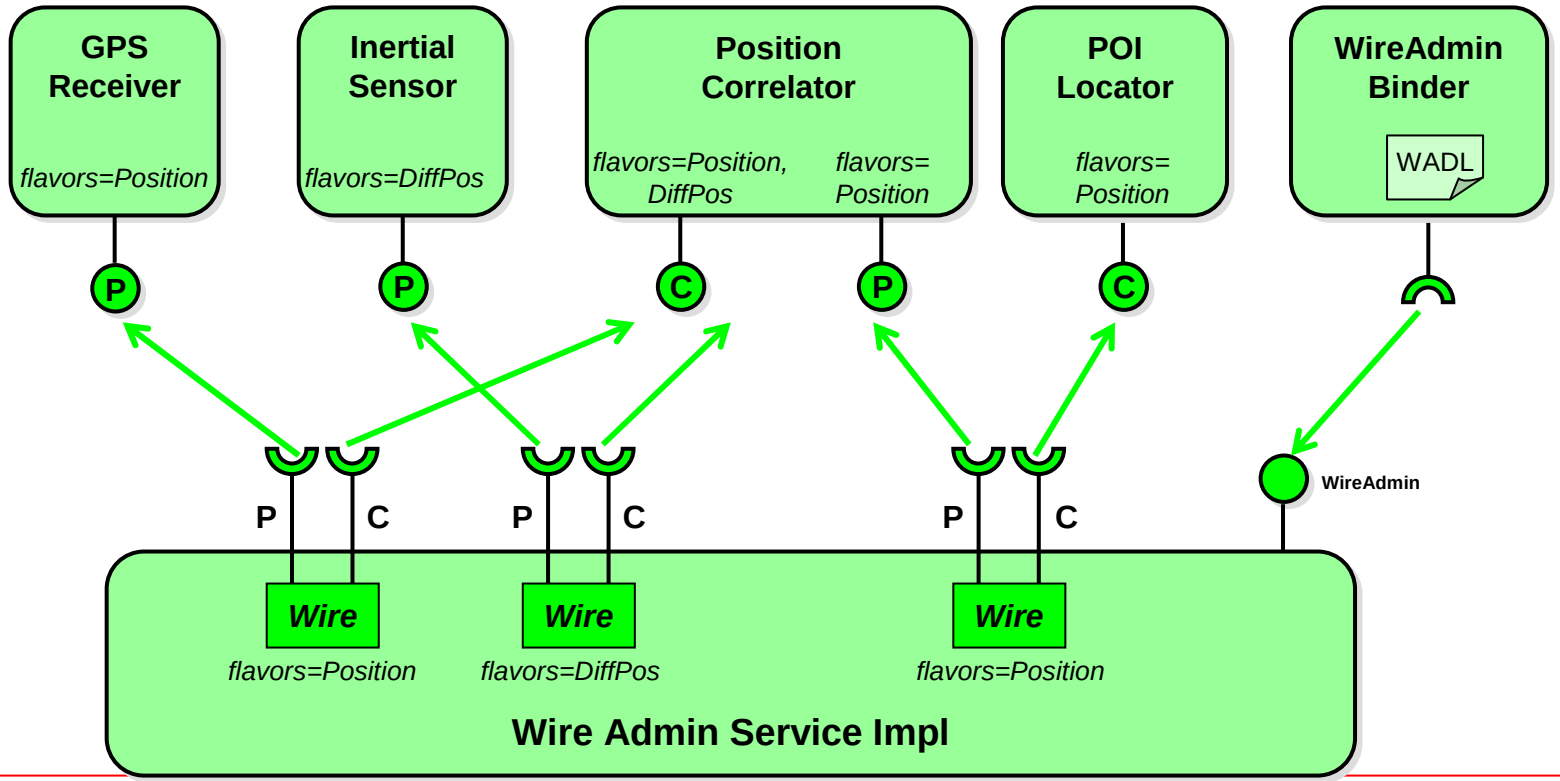
Wire Admin Service
Exemple d'application M2M (i)

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



Wire Admin Service
Exemple d'application

Aide à la navigation

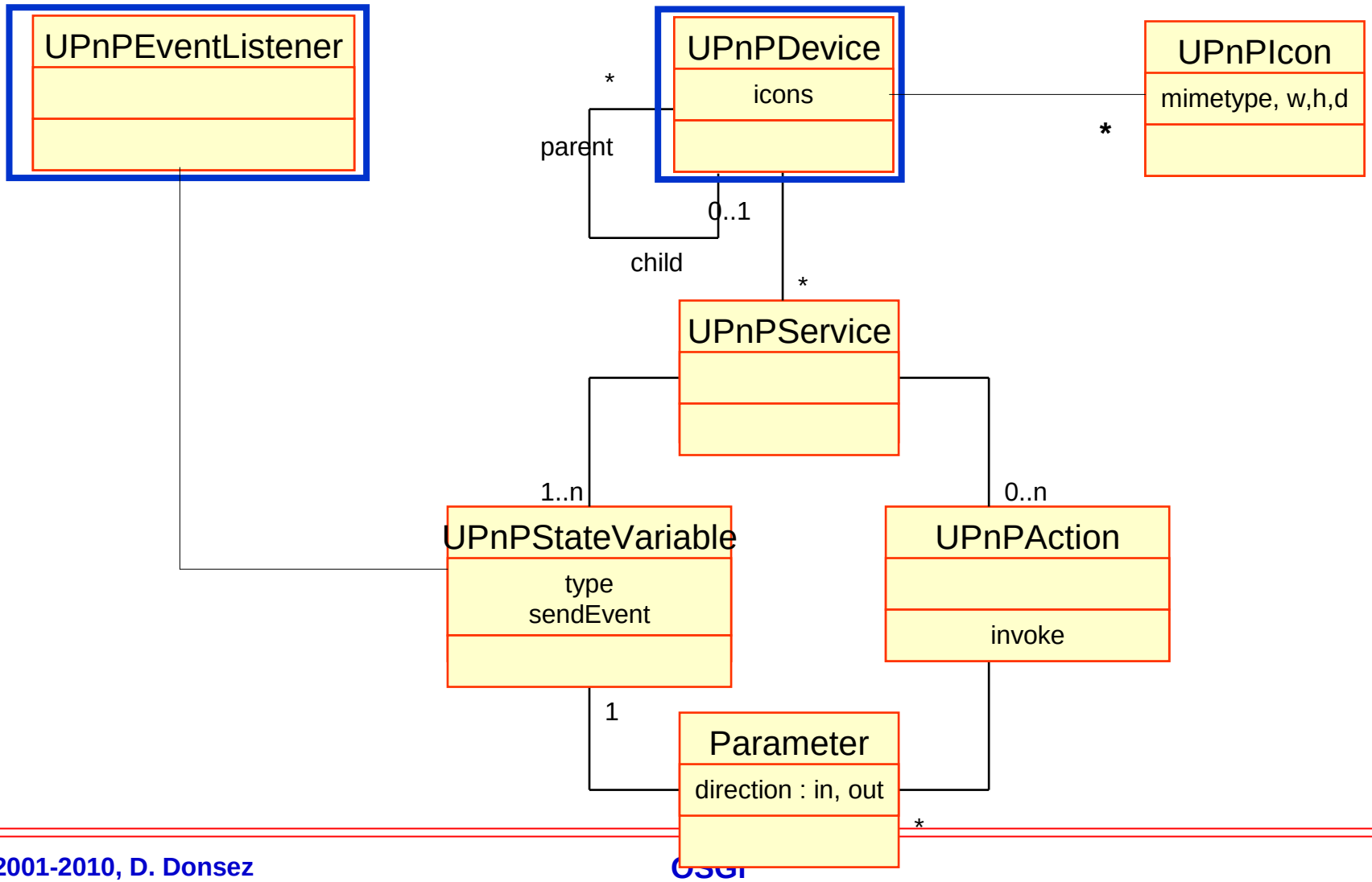


UPnP Driver Service R3

- **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

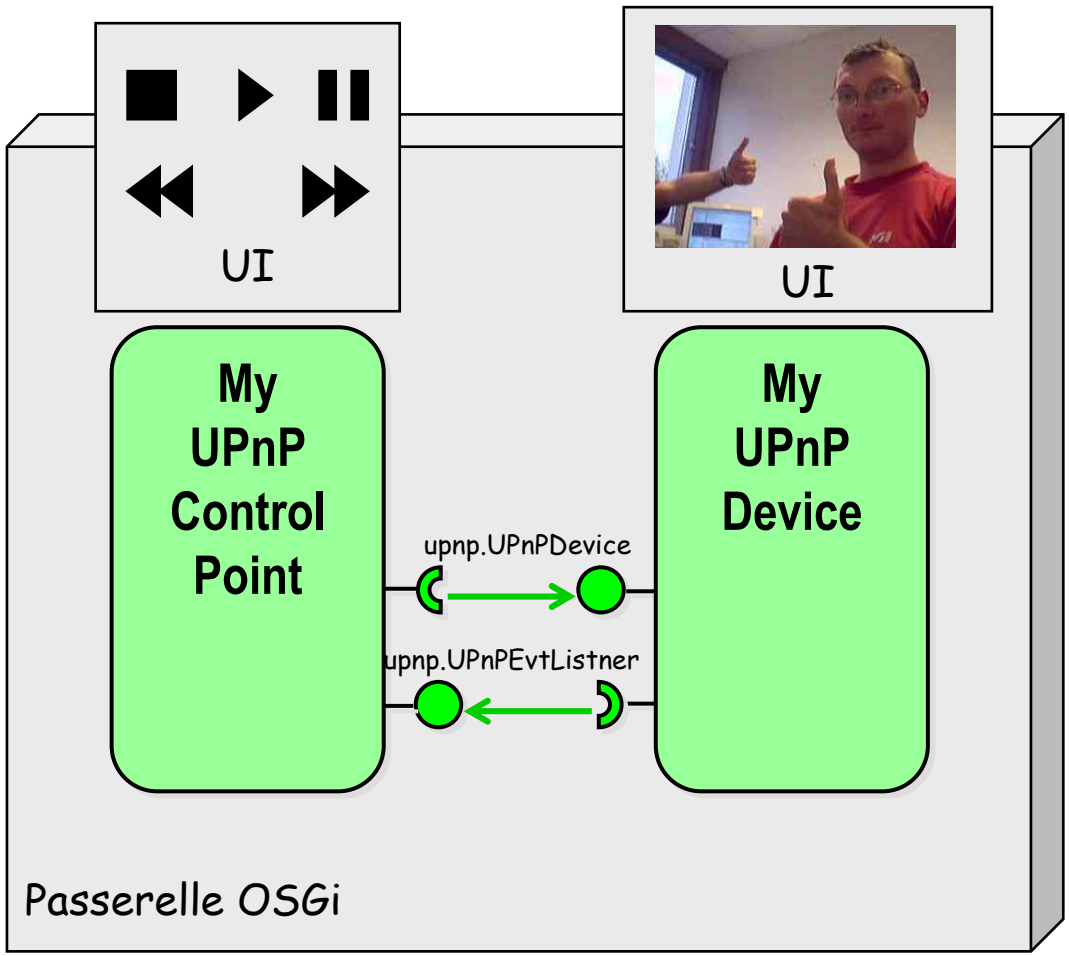
UPnP Driver Service
Les interfaces représentant les équipements UPnP



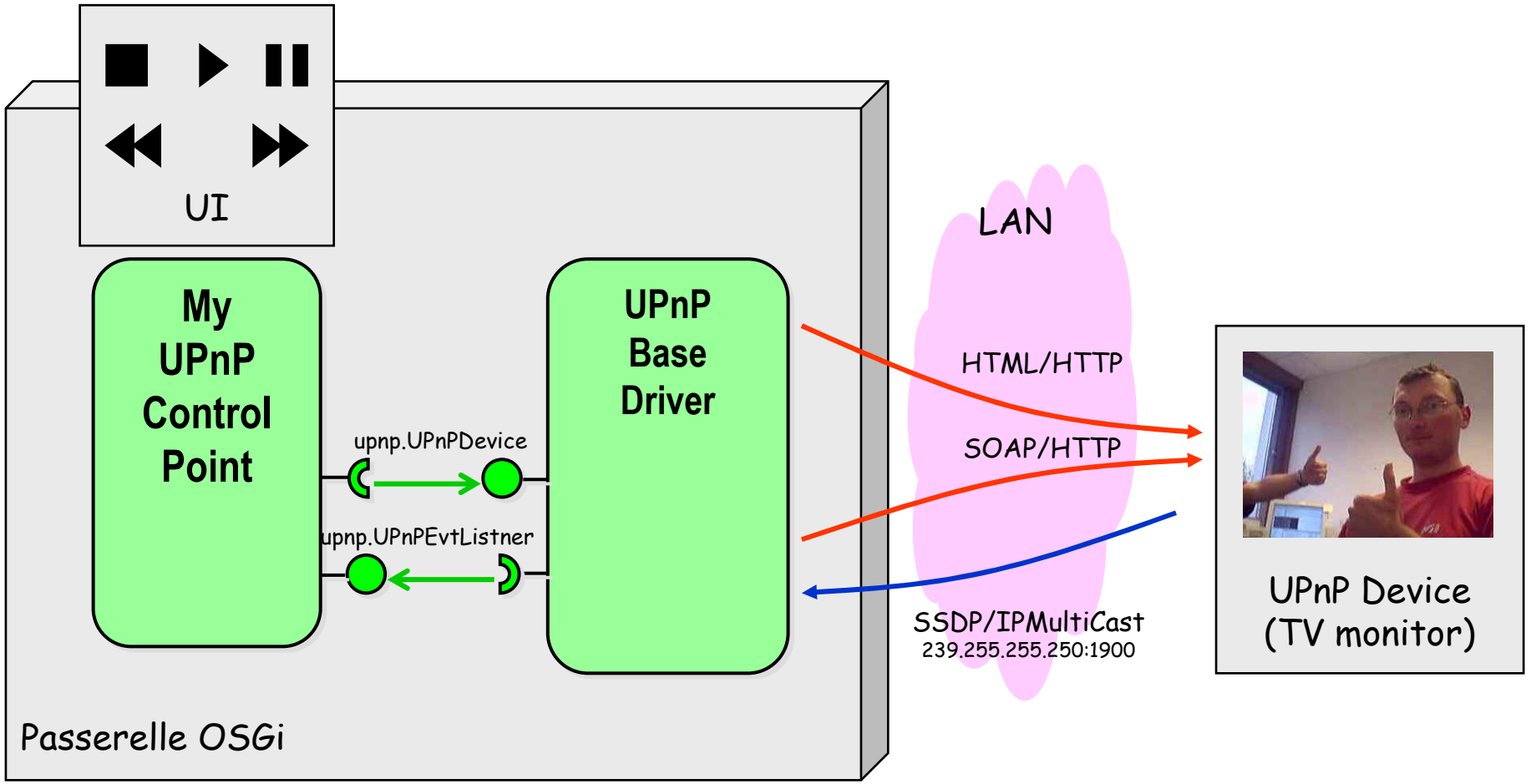
Exemple: un *device* Téléviseur et son point de contrôle

- 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

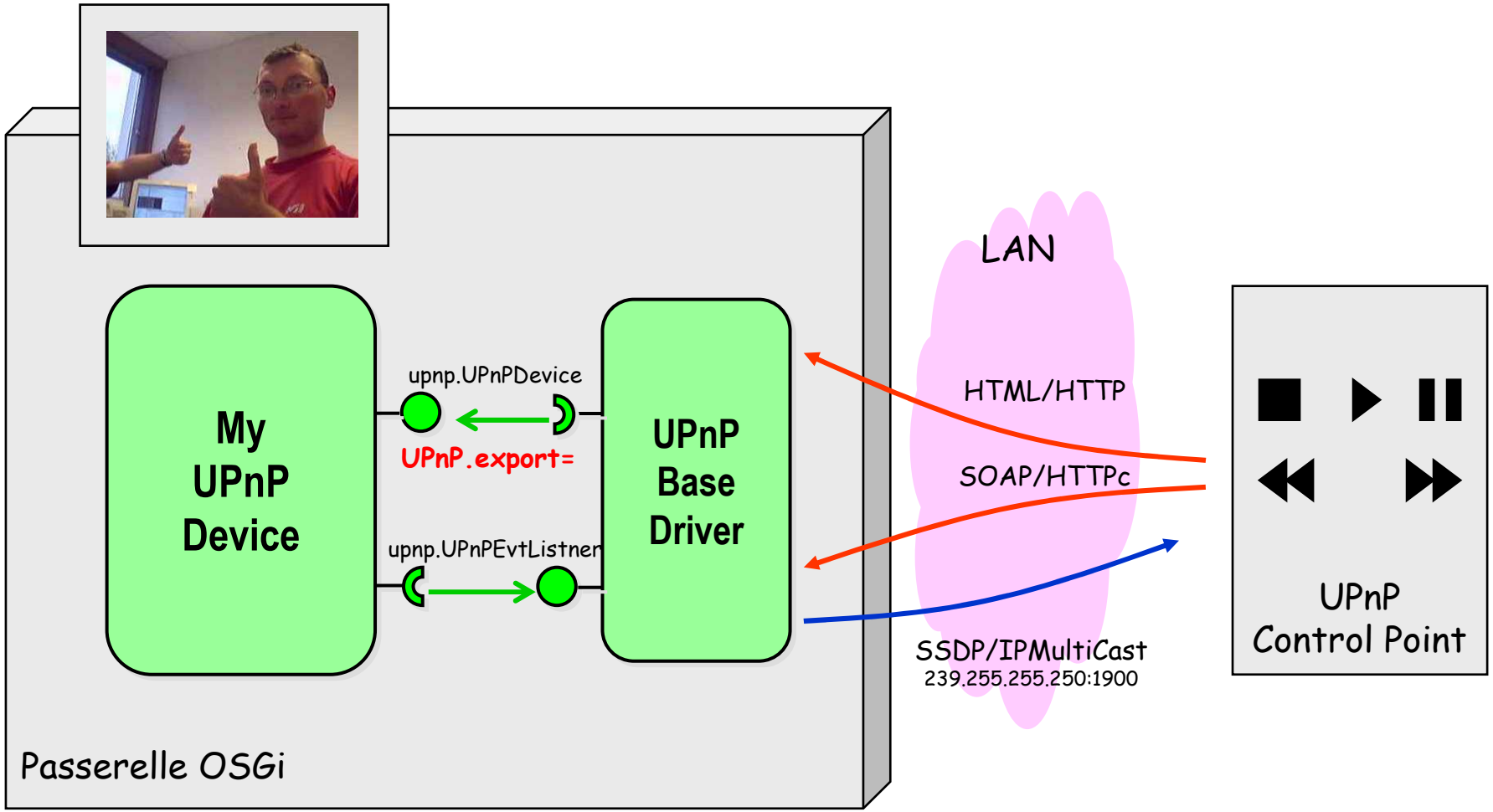
UPnP Device Driver : Mise en œuvre Collocalisé



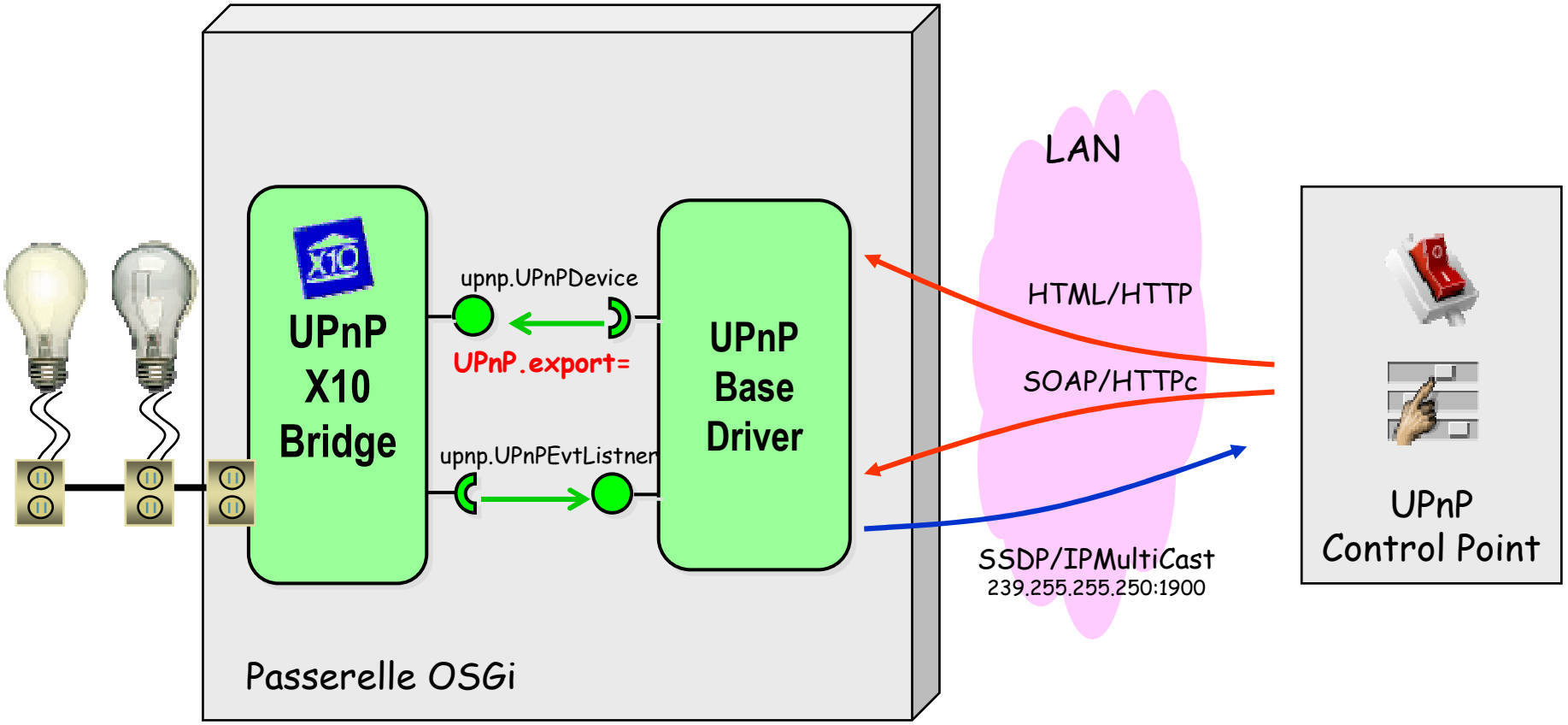
UPnP Device Driver : Mise en œuvre
Point de contrôle



UPnP Device Driver : Mise en œuvre
Equipement



UPnP Device Driver : Mise en œuvre
Passerelle micro-monde



Under Construction
En Construction

R3

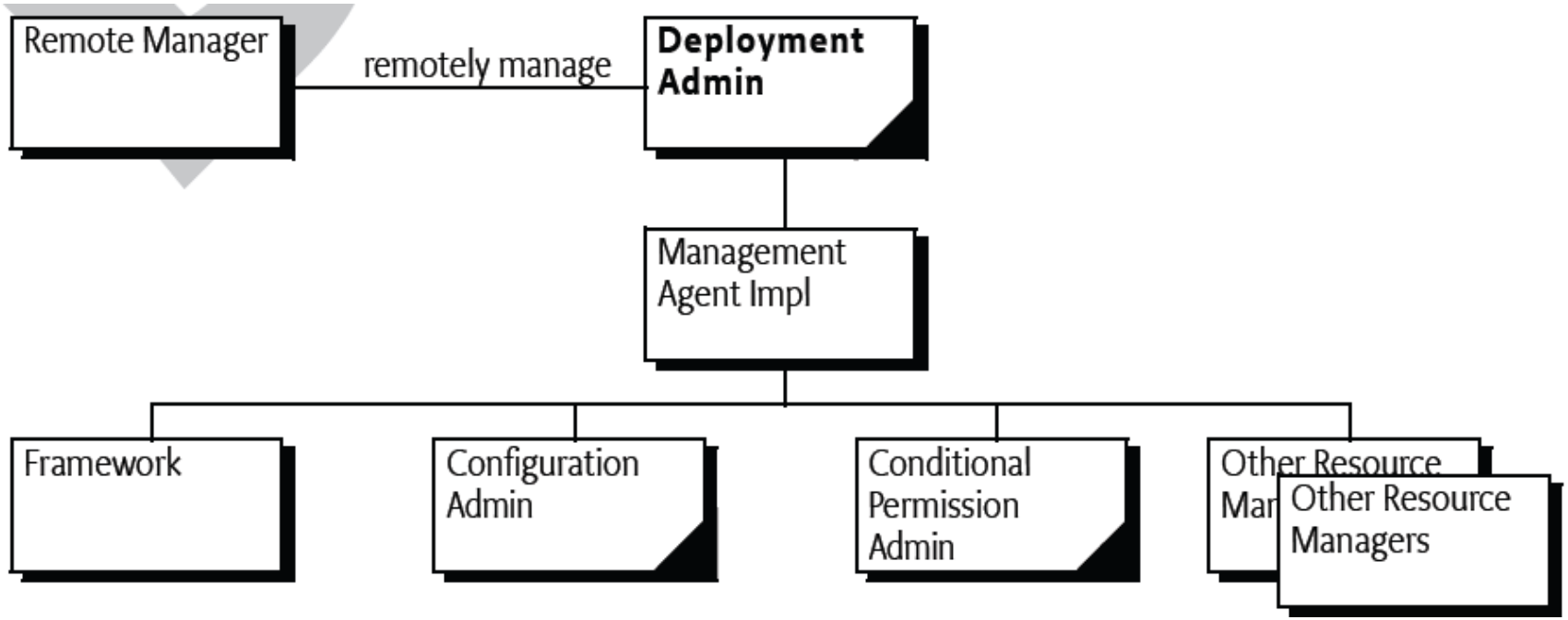
Execution Environment Specification

- **Définit l'environnement d'exécution (java) requis pour que le bundle puisse s'exécuter**
  - **Il doit être inclut dans la liste des environnements du framework d'accueil pour passer dans l'état RESOLVED**
  - **Propriété `org.osgi.framework.executionenvironment`**

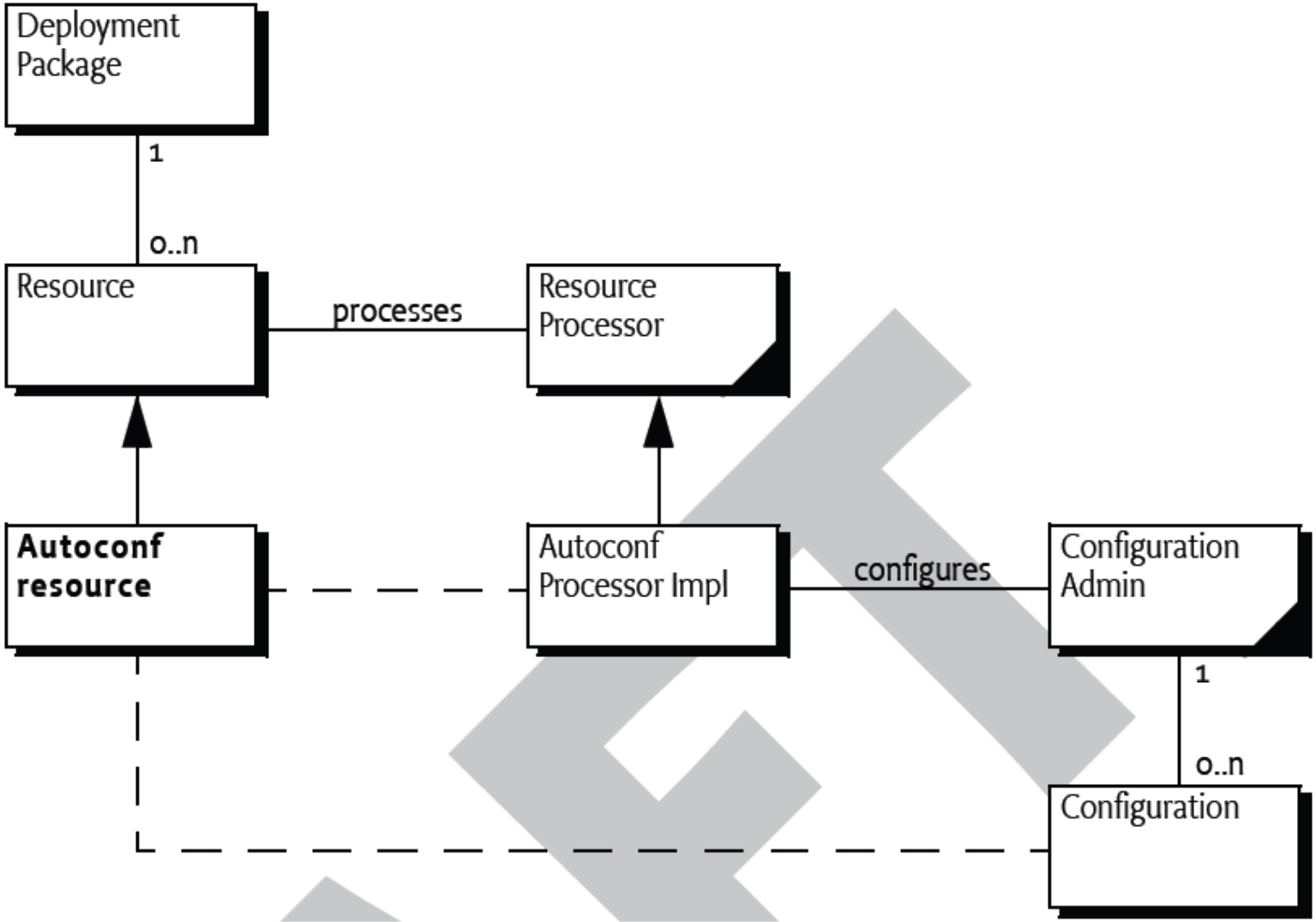
- **Entête de Manifest**

```
Bundle-RequiredExecutionEnvironment: \
   CDC-1.0/Foundation-1.0, OSGi/Minimum-1.0
```

114 Deployment Admin Specification

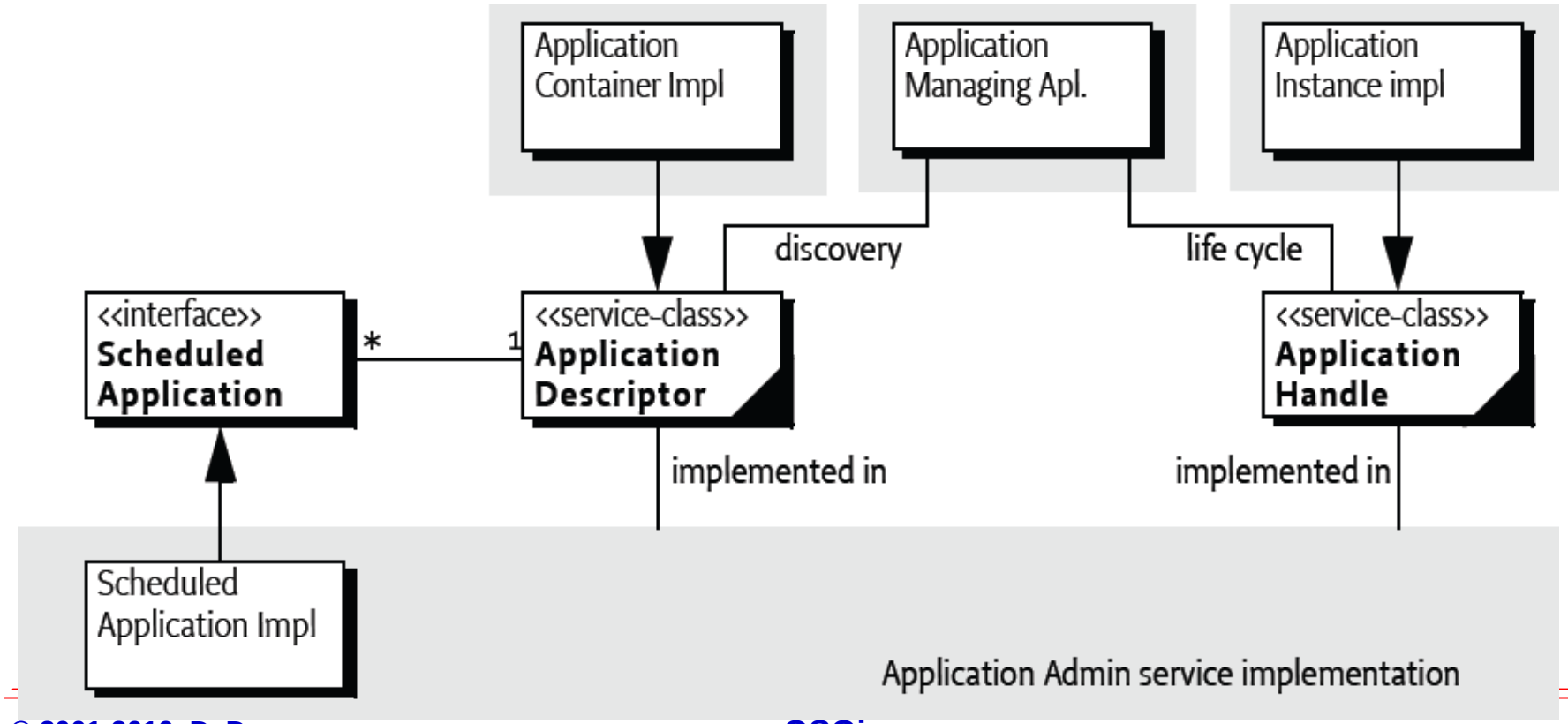


115 Auto Configuration Specification



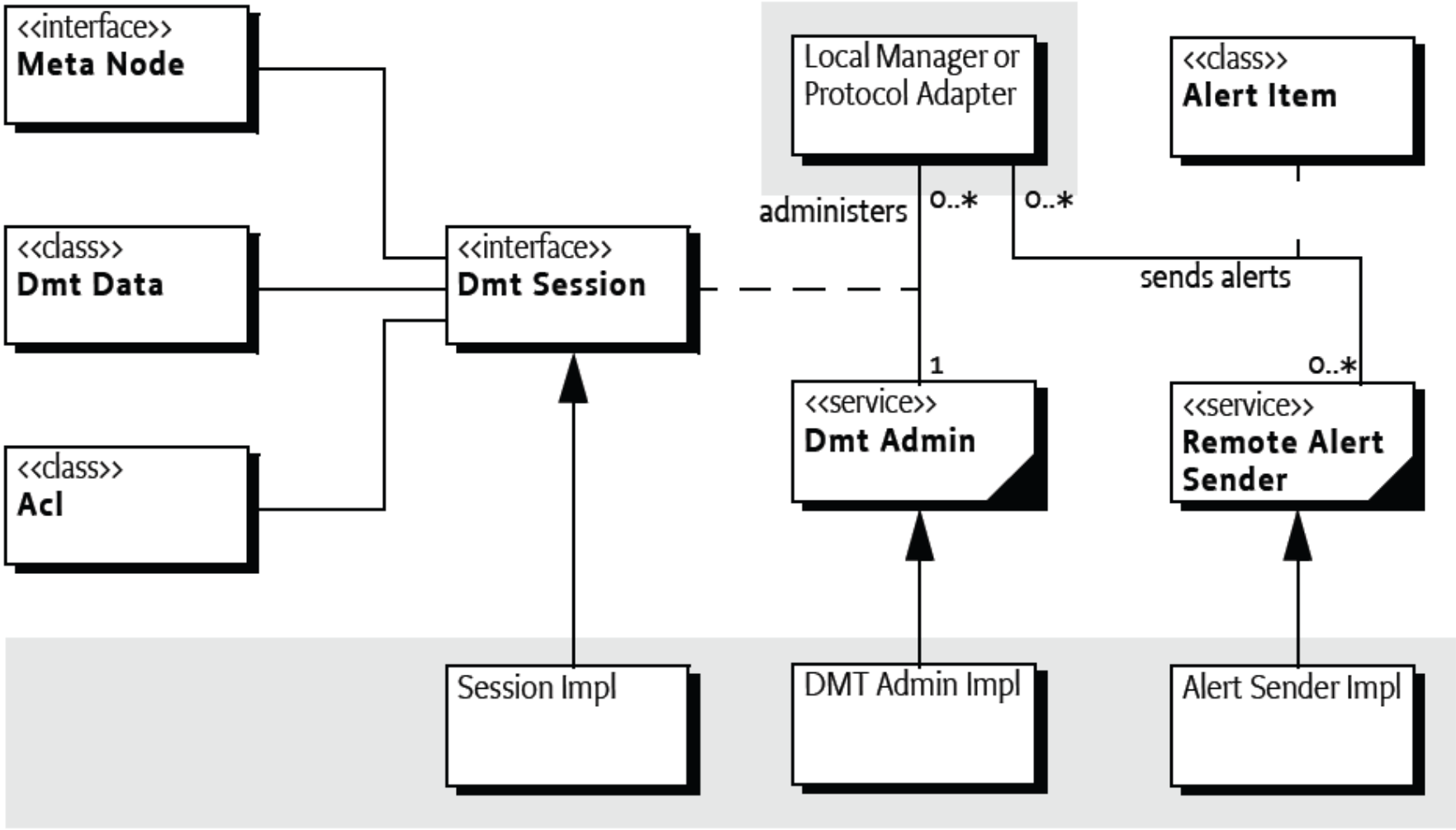
116 Application Admin Service Specification

- ***org.osgi.service.application***



117 DMT Admin Service Specification

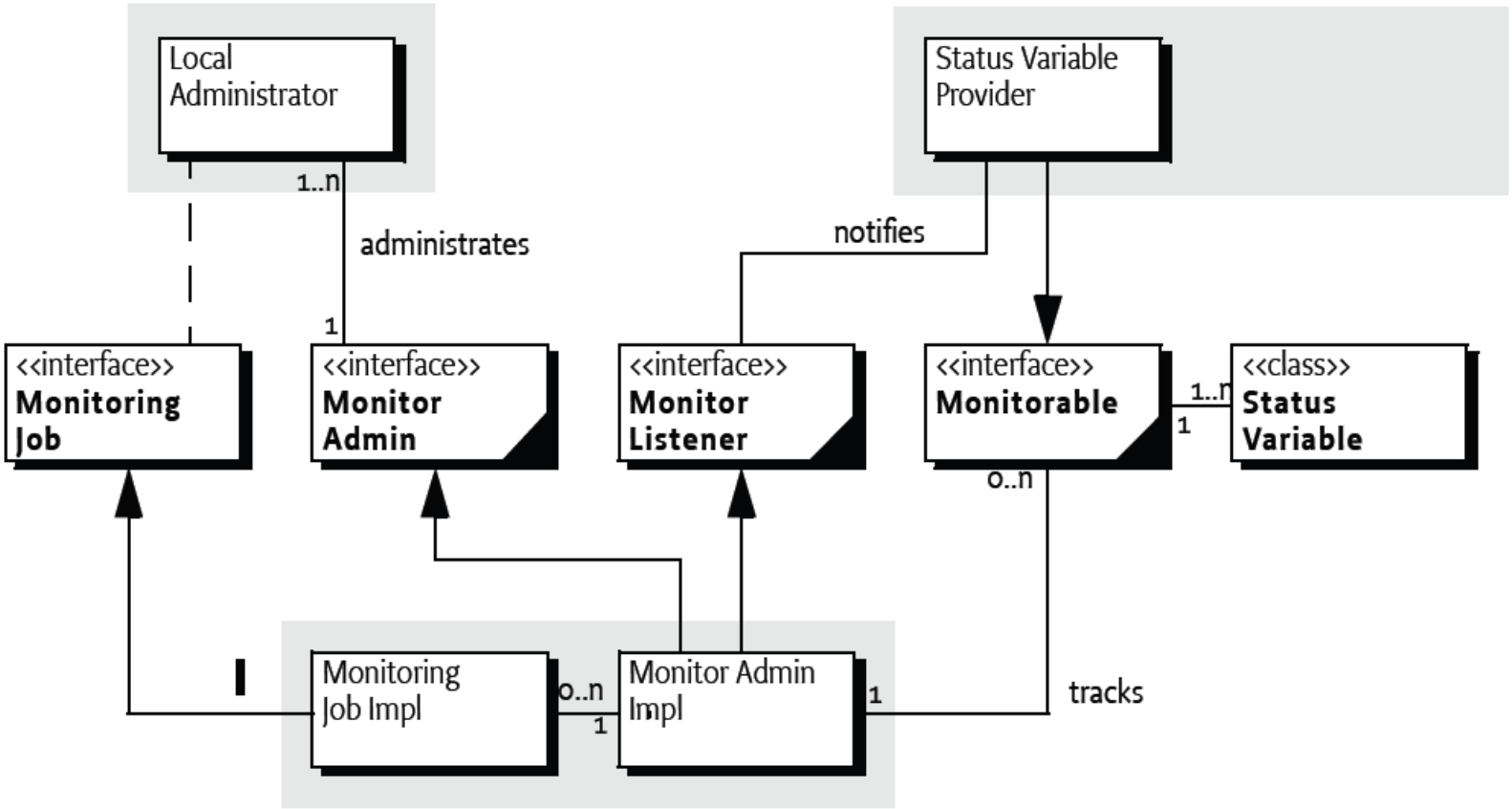
- *org.osgi.service.dmt*



119 Monitor Admin Service Specification
org.osgi.service.monitor

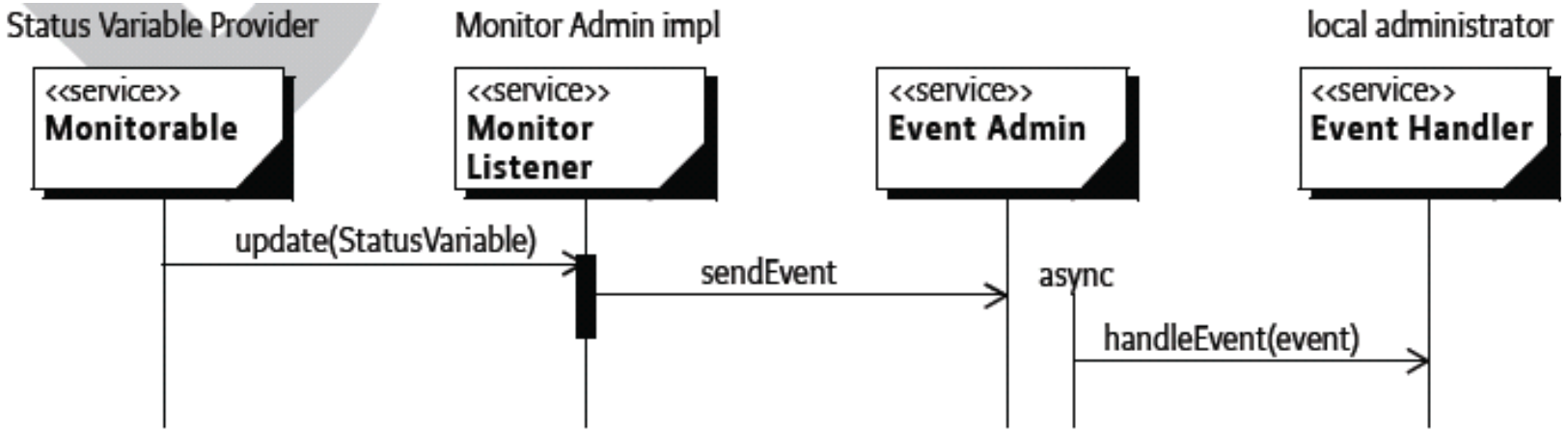
Under Construction
En Construction

- **Motivation**
  - **Découvrir des variables d'état et publier/écouter leurs modifications**
    - **mémoire disponible, niveau d'énergie de la batterie, nombre de SMS envoyés …**
  - **S'appuie sur l'Event Admin**



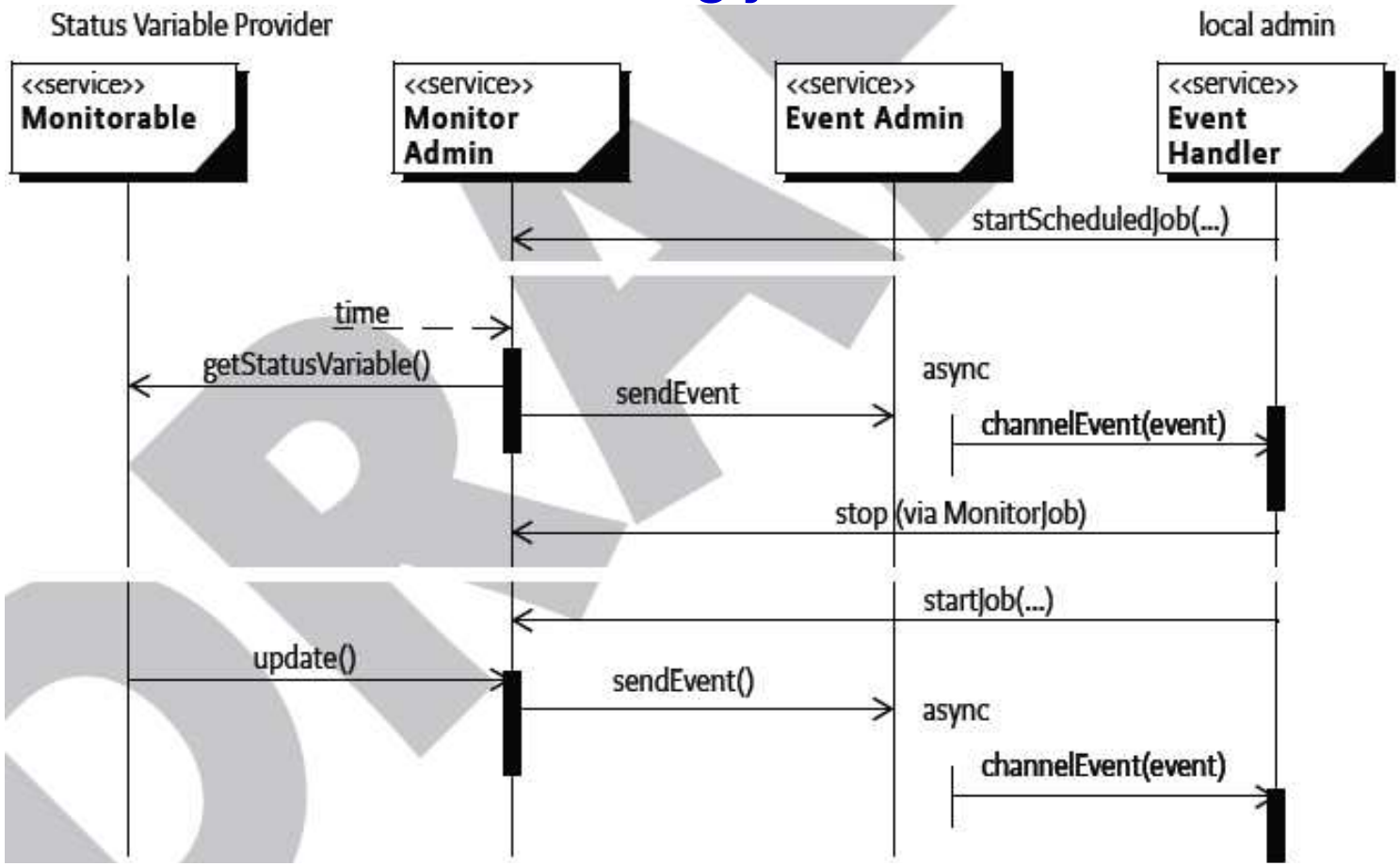
119 Monitor Admin Service Specification

Notification sur changement



119 Monitor Admin Service Specification

Time and Event monitoring job



End-to-End Management

- Motivations
  - Gestion d'un (très grand) parc de plateformes OSGi
  - Et des équipements qui y sont attachés
- Besoins
  - Déploiement « transactionnel »
  - Politiques de déploiement
  - …
- E2E Remote Management EG
  - Expert group a l'Alliance
- Produits
  - Prosyst Remote Manager
  - IBM Expeditor Framework
  - …

Enterprise-side OSGi

- Eric Newcomer « OSGi could not be a YAJEEC »
- Deal with legacy technologies
  - CORBA, MQ, CICS, Tuxedo, JEE, .NET
- Combining with the next technologies
  - Spring-OSGi, SCA-OSGi

Distributed OSGi

- Motivations
  - Expose/Provides local services to remote consumers
  - Consumes remote services
- Proposition
  - Extended Service Binder
  - R-OSGi
  - SCA/OSGi
  - RFC 119
    - http://www.osgi.org/download/osgi-4.2-early-draft3.pdf).
- RFC 119
  - http://cxf.apache.org/distributed-osgi.html
    - provides the Reference Implementation of the Distribution Software (DSW) component of the Distributed OSGi Specification
    - using Web Services, leveraging SOAP over HTTP and exposing the Web Service over a WSDL contract.

OSGi and JMX

- MOSGi
- RFC 139

Conclusion intermédiaire

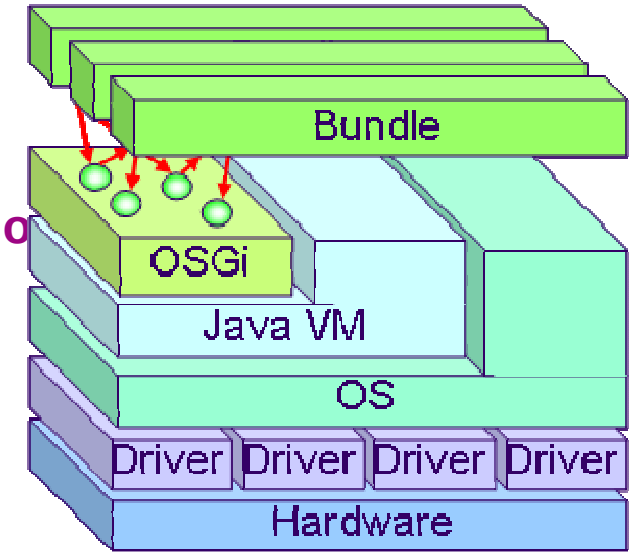
+ Gestion des dépendances de package
+ Déploiement dynamique de bundles
+ Environnement varié:
  embarqué, station de travail, serveur.
+ Fonctionnalité de base pour le déploiement de composants

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


Q&R

What is OSGi Alliance ?

- Consortium founded in March 1999
- Objective
  - Create open specifications for delivering administrated services through a network
- Define
  - A common platform (framework)
    - Services deployment
    - Services execution and administratio
  - A set of based services:
    - Log Service, Http Service…
  - A device access specification
  - A deployment unit, a *bundle*



OSGi Main Concepts

Framework:

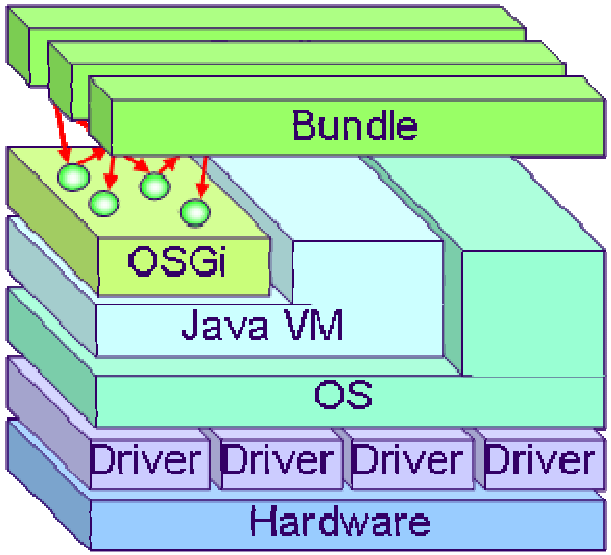
- Bundles execution environment
  - Oscar (Objectweb) / Felix (Apache), Knopperfish, Equinox, SMF, ProSyst, Siemens VDO, …
- Event notification

Bundles:

- Services diffusion and deployment unit

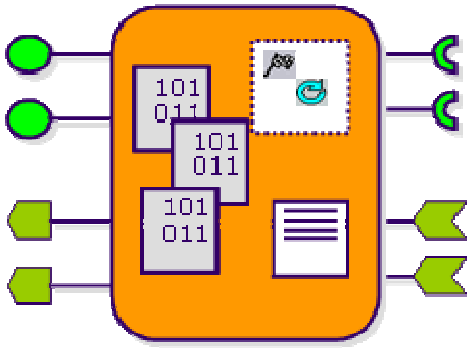
Services:

- Java Object implementing a well define contract



Middleware and Application Packaging

- **Modularize the middleware/application**
  - Distribute the different middleware services
  - Better component visibility
  - Need of a deployment container
  - Partial update without restart all

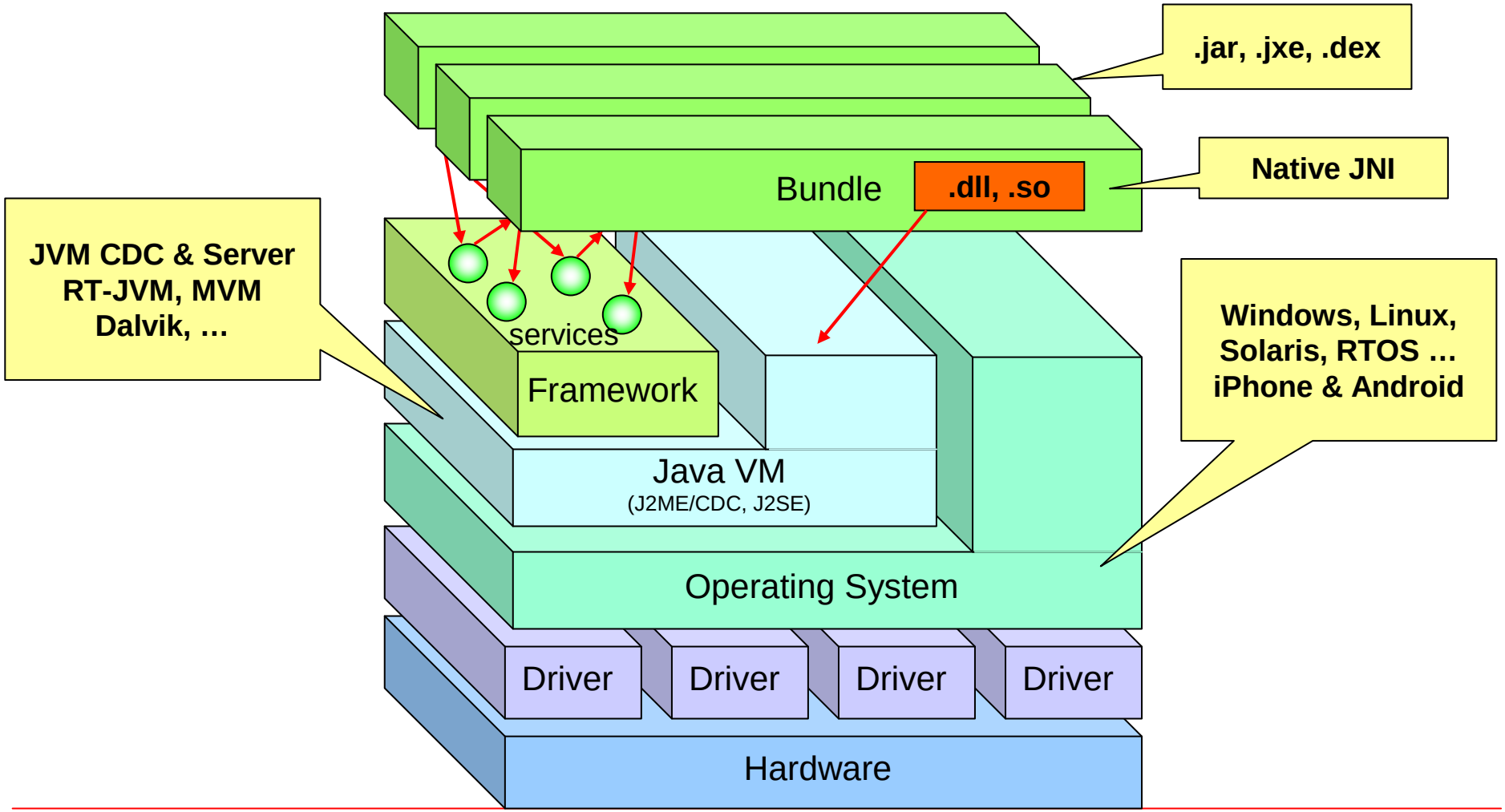


- **Implementation**
  - Based on Jarfile and Manifest entries
  - Explicit Package dependencies and Versioning (range)

- **Ready for probably next generation standard**
  - JSR 277, Java Module System
  - Overtake JNLP(JSR-56), J2EE EAR, OSGi R3 bundle
  - Java Platform 7.0 (2010), Jigsaw

The OSGi « stack »

avec l'aimable autorisation de Peter Kriens



OSGi

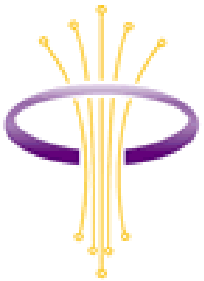
Acteurs, concurrences, tendances et perspectives

L' OSGi™ Alliance

- actuellement 44+ membres
- de plusieurs domaines industriels

- sur les 4 segments

L'OSGi™ Alliance



=

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

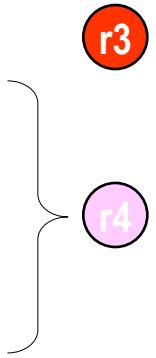
- **IBM**
  - OSGi est au cœur de la stratégie d'IBM
  - placé sur la partie « edge » du système IT
    - poste de travail (RCP)
    - serveur enfoui
  - Remarque:
    - Eclipse 3.0 (donc WebSphere Studio) est désormais développé au dessus d'OSGi (Equinox)
- **Nokia**
  - Pousse pour
    « Java (MIDLet) 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
- MakeWave (ex *Gatespace* AB)
- *IBM* SMF
- *Insignia*
- *Nano Computer System*
- *ProSyst Software* mBedded Server
- *Wind River*
- *Siemens VDO TLA (R3)*
- …

La communauté open-source

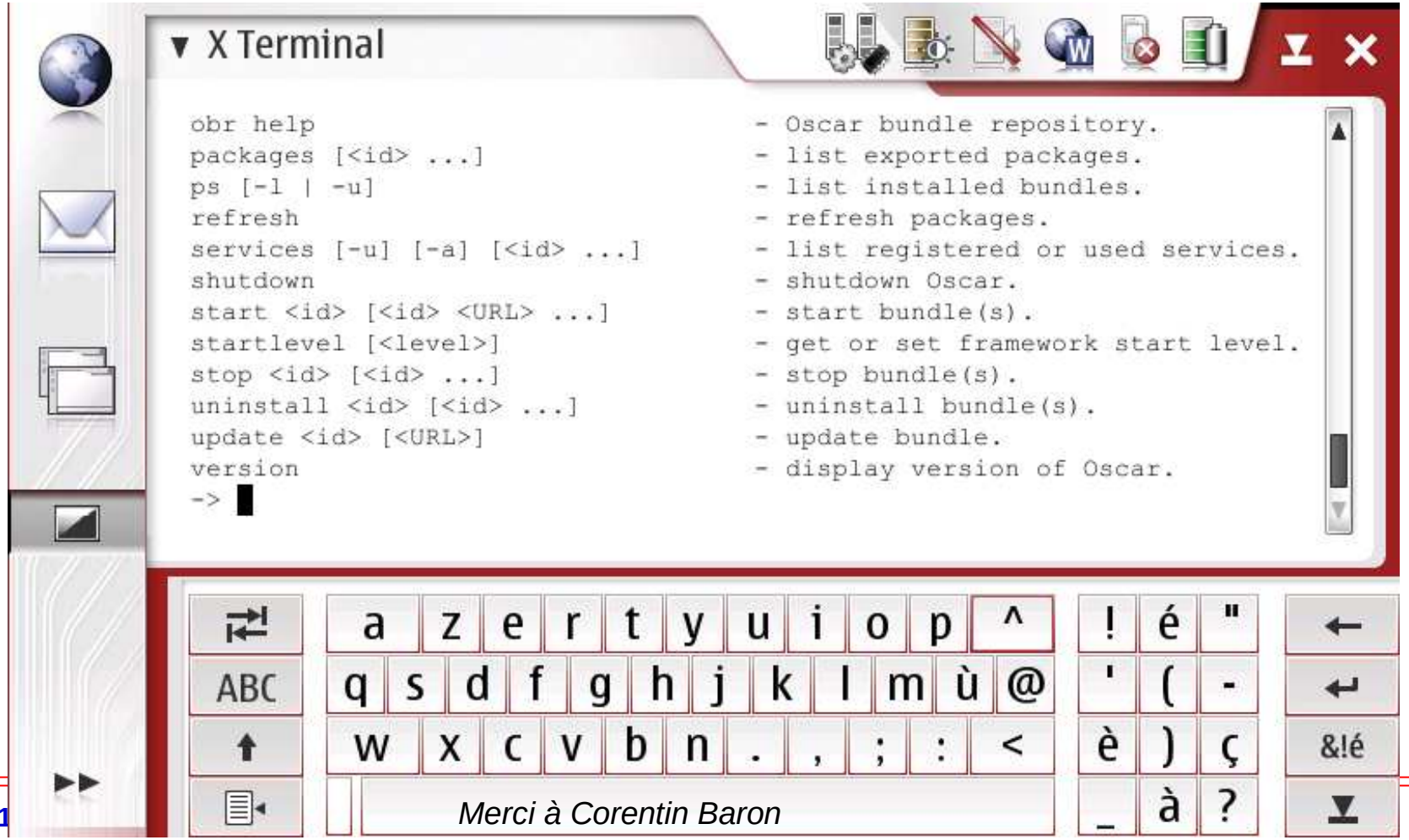
- **Plusieurs implémentations et communautés**
  - ObjectWeb Oscar
  - Knopflerfish
  - Eclipse Equinox (donation IBM SMF)
  - Apache Felix (suite d' ObjectWeb Oscar)
  - JBoss microContainer (http://labs.jboss.com/jbossmc)



- **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)**



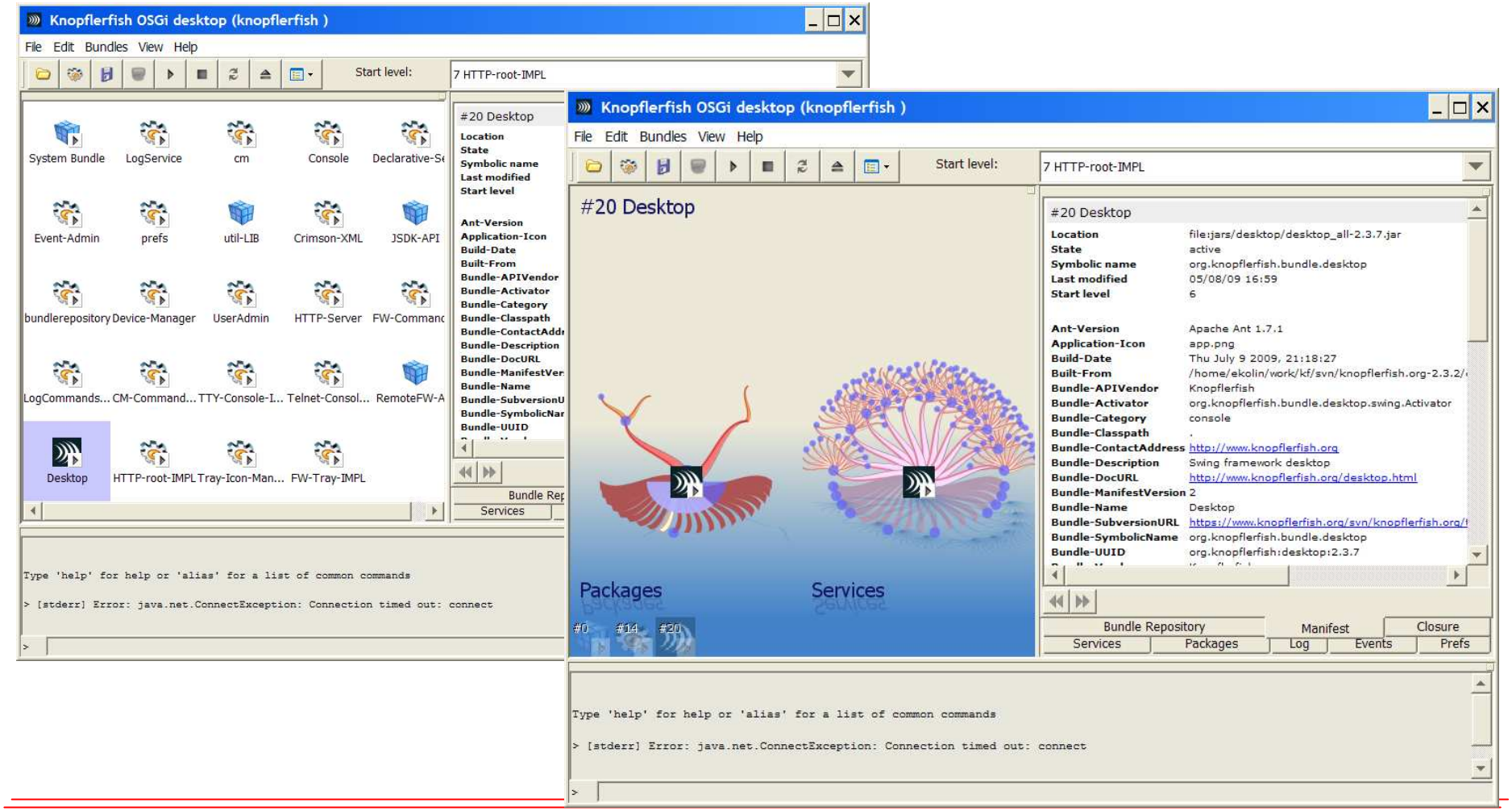
Apache Felix on Google Android

- Google Android / Dalvik VM
  - JVM and JRE for the Google phone
    - Support Java 5 features (annotations, …)
    - Uses a proprietary class format (.dex instead of .class/.jar)

- Felix running on Android
  - See
    - http://felix.apache.org/site/apache-felix-and-google-android.html

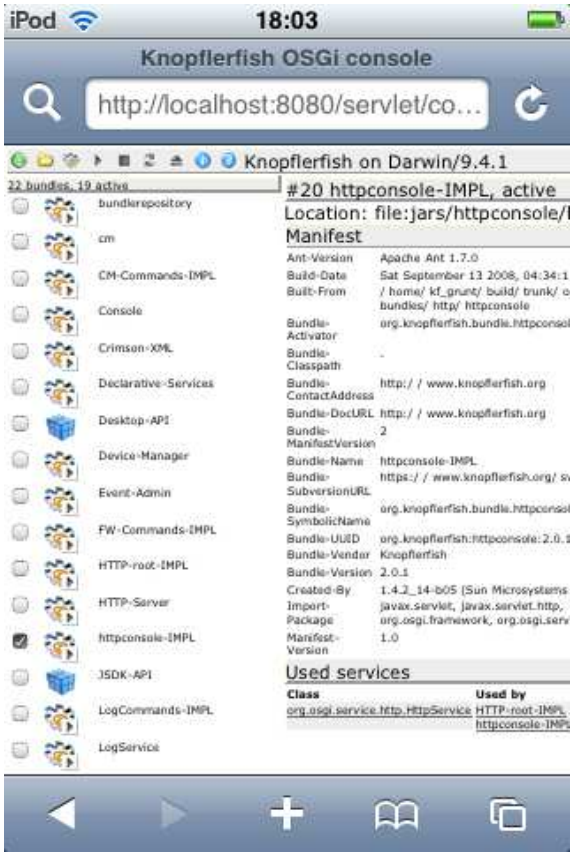
Knopflerfish OSGi

La console GUI



Knopflerfish OSGi

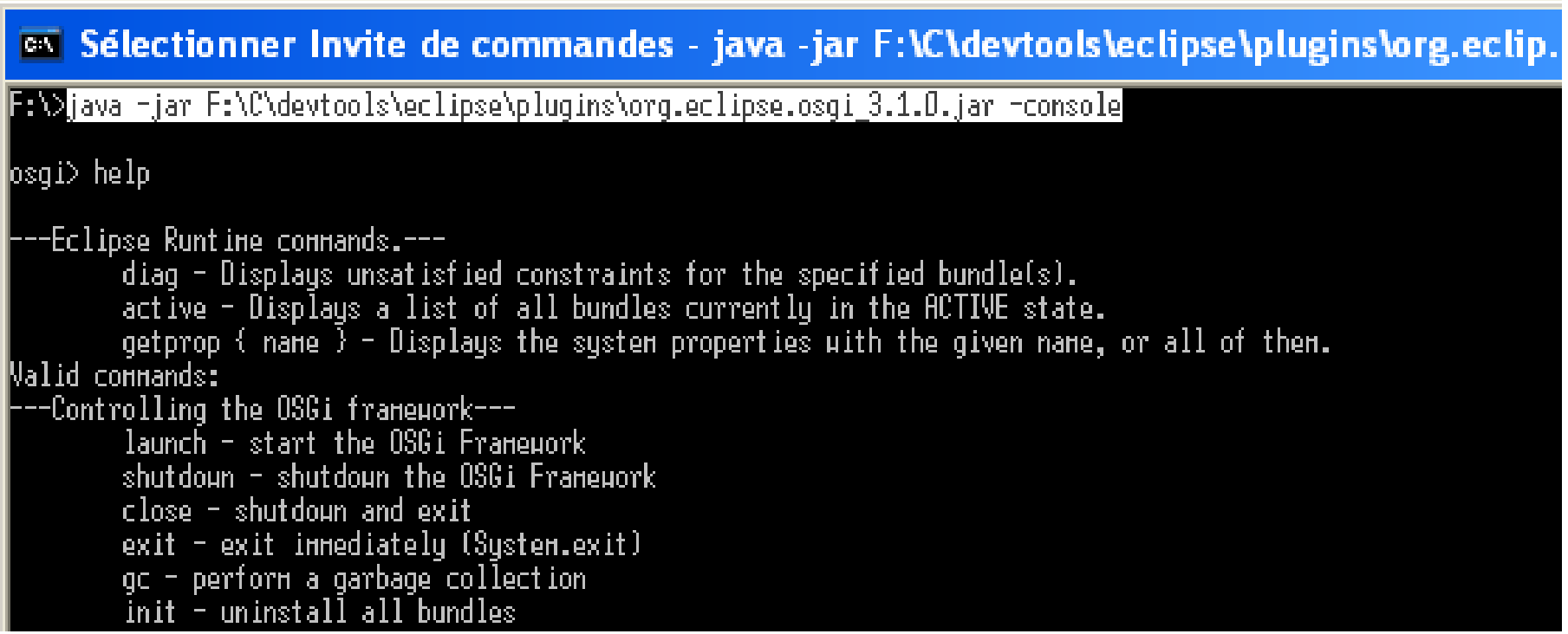
- La console GUI sur le iPod Touch/iPhone 2.1 (JamVM)



D'après http://knopflerfish.blogspot.com/2008/09/knopflerfish-osgi-on-ipod-touchiphone.html

Eclipse/OSGi

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



Concierge

http://concierge.sourceforge.net/

- **OSGi R3 implementation optimized for constrainted mobile devices (iMote2, BUG (http://buglabs.net/products), NSLU, WRT54, …)**

http://www.spectrum.ieee.org/video?id=223

- *Jan S. Rellermeyer, Gustavo Alonso: Concierge: A Service Platform for Resource-Constrained Devices. In: Proceedings of the 2007 ACM EuroSys Conference, Lisbon, Portugal, 2007.*

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
  - En principe accessible à quelques plateformes open sources (Felix, Equinox, …)

JSR 277 : Java™ Module System

- *defines a distribution format and a repository for collections of Java code and related resources.*
- *also defines the discovery, loading, and integrity mechanisms at runtime.*

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

Notion of super-package

- **indroduce in Java**
- **in order to define class visibility outside the deployment unit (JSR277)**

Example

```
superpackage A {
  member package P;
  member superpackage B;
}
superpackage B member A {
  member superpackage C;
  export superpackage C;
}
superpackage C member B {
  member package Q;
  export Q.X;
}
```

A type P.Y can access type Q.X because Q.X is exported from superpackage C, and C is exported from B, so Q.X is available in A

Jigsaw

- **JDK7 is big**
  - **http://blogs.sun.com/mr/entry/jigsaw**

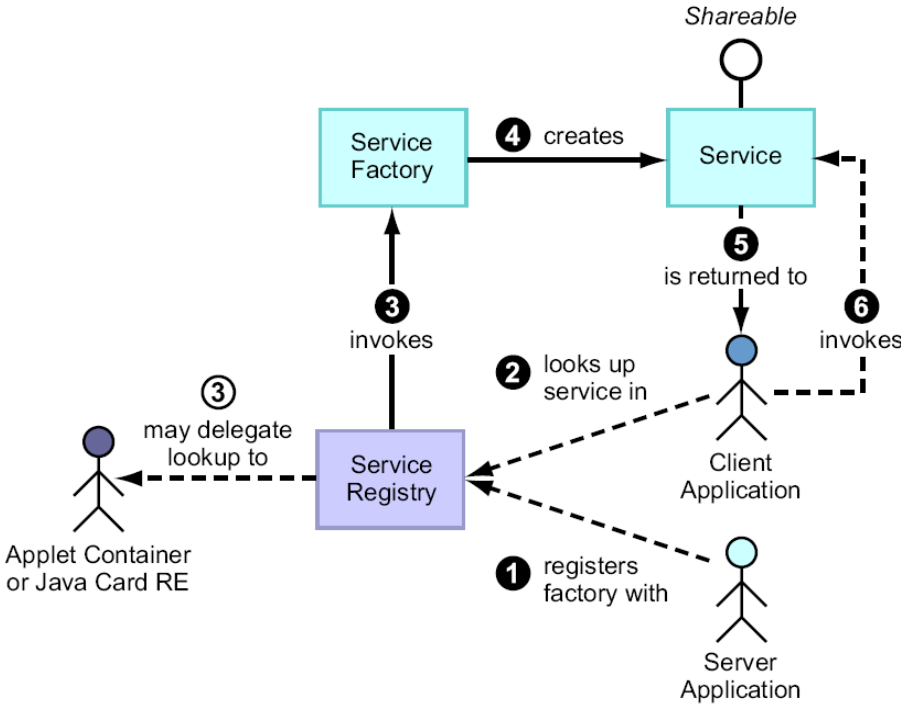
HK2 « Hundred Kilobytes Kernel »

https://hk2.dev.java.net

- **Small kernel (than OSGi™) to build modular softwares**
  - **consist of two technologies :**
- **Modules subsystem**
  - **offer a better level of isolation between parts of the application.**
  - **path to the implementation of modules (JSR 277 & 294) in Java SE 7.**
- **Component Model**
  - **define components which can also be seen as Services.**
    - **These components can be automatically and dynamically discovered by the runtime and can use innovative technologies such as Inversion of Control or injection of dependencies as well as automatic resolution of dependencies.**
    - **Annotations (org.jvnet.hk2.annotations)**
      - **@Contract, @Service, @Inject, @Extract, @Factory, @Scoped, @ContractProvided, …**
- **Remark: foundation for the GlassFish V3 application server**
  - **April 2008 : GlassFish v3 on Apache Felix !!!**

JavaCard 3.0 Connected Edition
Shareable Interface Object-based Services (SIO)

- **Service-oriented communications between on-card applications**



- **Limitations**
  - **Service classes must inherit of the interface javacard.framework.Shareable**
  - **Service lookup is only based on service URI (e.g. sio:///transit/pos/ticketbook)**
  - **javacardx.facilities.ServiceRegistry lookup methods return Shareable objects**
  - **No event (javacardx.facilities.StandardEvent) for SIO registration/unregistration**

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 : *JUNGO OpenRG*

http://www.jungo.com/openrg/index.html

- **TODO**
- ***"OpenRG™ is a complete and integrated gateway software platform for developing network devices in the digital home and small office including triple play residential gateways, home/SOHO routers, home gateways, wireless access points, cable/DSL routers and voice gateways."***

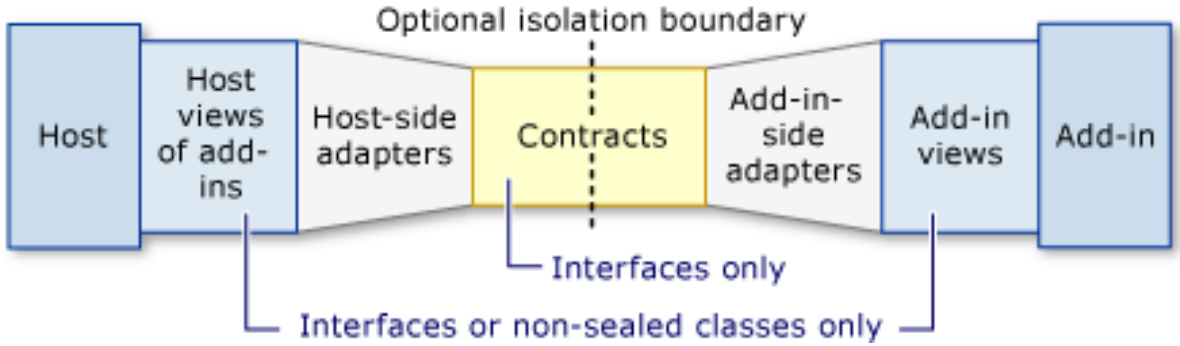
La concurrence : MicroSoft .NET

- **Passerelles industrielles (SCADA) OPC** http://en.wikipedia.org/wiki/OPC_Foundation
  - COM/DCOM : drivers field bus, domaine SCADA

- **.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 (CLR sur *bare metal*)
- **Le problème de .NET (1 et 2)**
  - Le déchargement d'une classe requière l'arrêt du domaine d'application (AppDomain).
  - Donc pas de mise à jour partielle d'une application
- **Des compromis (non gratuits) restent possibles**
  - [Escoffier06]
  - SOF - An OSGI-like modularization framework for C++
    - http://sof.tiddlyspot.com/ http://www.codeproject.com/KB/library/SOF_.aspx

La concurrence : MicroSoft .NET

- **Managed AddIn Framework (MAF, aka System.AddIn)**
  - **Based on Application Domains**



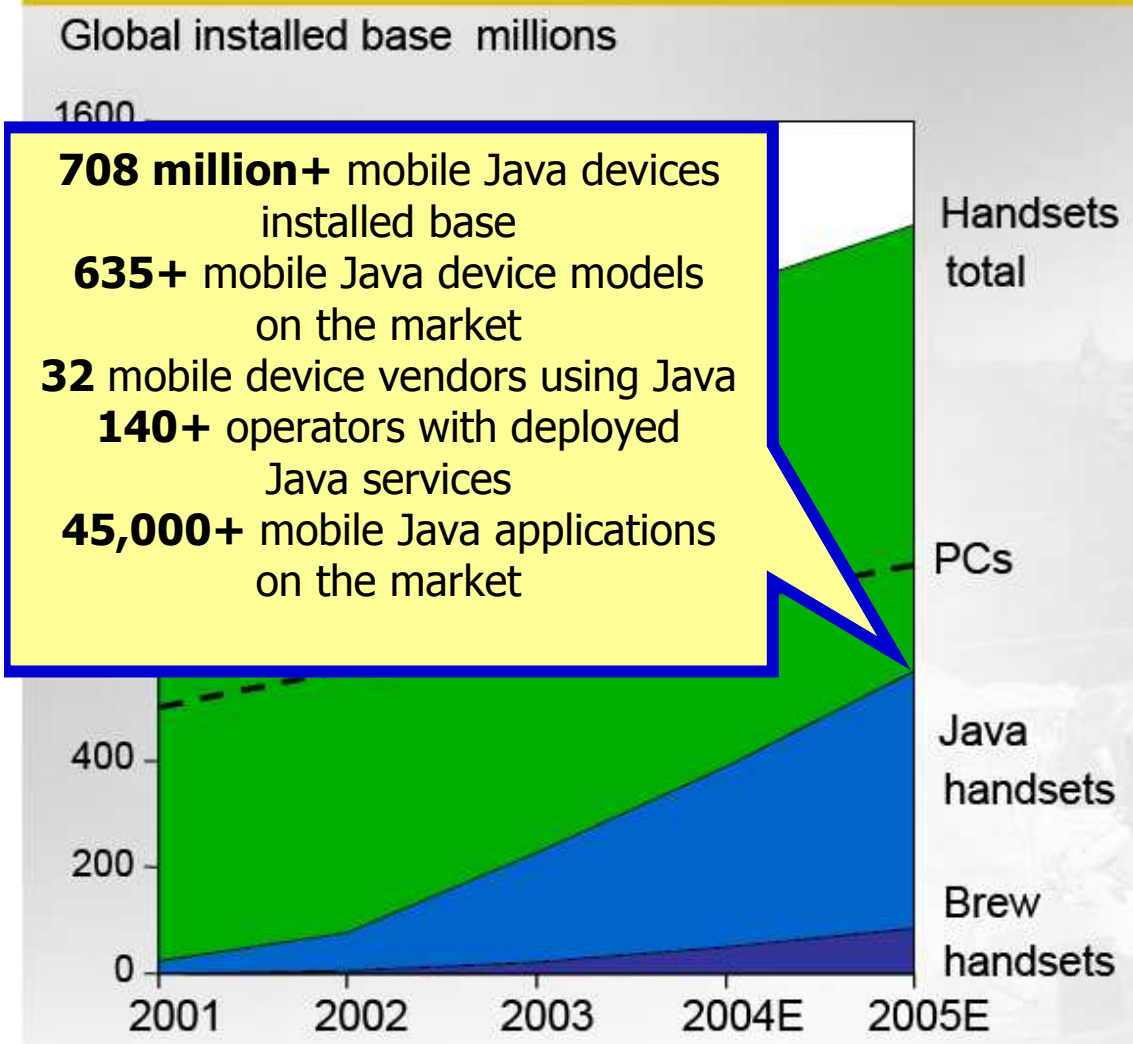
  - Discovery **– Find addins at runtime**
  - Activation **– Load an addin at runtime**
  - Versioning **- Backwards/forward compatibility**
  - Isolation **– Load addins into separate AppDomains/Processes so they cannot crash they whole app, among other reasons**
  - Lifetime Management **– MAF will handle addin lifetimes and memory/AppDomain management**
  - Sandboxing **– Load addins with specific permission sets like "Internet"**
  - Unloading **– Unload an addin without worrying about tedious AppDomain management**

- **Other: Managed Extensibility Framework (MEF) in .NET 4**

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 OSGi
    - Serveur IT (J2EE, …), Web Framework, …
    - JOnAS 5, Geronimo, ApacheDS, JAMES, WebSphere, JBoss 5 …
    - ECP
  - Enterprise Expert Group à l'OSGi Alliance
    - Workshop sur OSGi + J2EE (11/09/2006, San José)
  - End-to-End Remote Management Expert Group à l'OSGi Alliance

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
- JSR 313 Java Platform, Enterprise Edition 6
  - Extensibilité du serveur JEE
- 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, …
- 7 réunions depuis Décembre 2004
- Ecole d'été : 6-7 Septembre 2007
  Ile de Bender (près de Vannes)
- Prochaine réunion le 29/1/2008 à Paris

En savoir plus

- *http://france.osgiusers.org*

Conclusion finale

+ **Très fort potentiel**

Ne passez pas à coté


Q&R

Bibliographie & Webographie

- **Spécification**
  - OSGi Alliance, « OSGi service gateway specification », http://www.osgi.org
- **Ouvrage**
  - Kirk Chen, Li Gong, « Programming Open Service Gateways with Java Embedded Server Technology », Pub. Addison Wesley, August 2001 ISBN#: 0201711028. 480 pages (un peu ancien)
  - Niel Barlett, « OSGi in Practice », http://neilbartlett.name/blog/osgibook/
  - Richard S. Hall, Karl Pauls, and Stuart McCulloch, « OSGi in Action », May 2009, 375 pages, ISBN: 1933988916, http://code.google.com/p/osgi-in-action & http://manning.com/hall
- **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
  - puis plein d'autres
- **Blogs**
  - Peter Kriens, *l'évangeliste OSGi,* http://www.osgi.org/blog/
  - Java modularity, JSR 277, JSR 291, JSR 294, OSGi, open source, and software design, http://underlap.blogspot.com/

Bibliographie & Webographie

- Framework open source
  - Oscar, Felix, Equinox, Knopperfish
- Index de bundles
  - http://bundles.osgi.org/browse.php
- Exhibitions
  - http://www.osgiworldcongress.com/
- 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
- Plus
  - http://france.osgiusers.org/Main/Documentation

Travaux pratiques

- **Sujet**
  - http://www-adele.imag.fr/users/Didier.Donsez/cours/exemplesosgi/tutorialosgi.htm
- **Programme**
  - 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 sur NSLU2
  - Démonstration d'administration de passerelle avec JMX
  - Mini-projet (1 bundle utilisant d'autres)

http://www.nslu2-linux.org

Mini-Projet : http.script

- Servlet exécutant une séquence de commandes auprès du service org.apache.felix.shell.ShellService
- Réutiliser le code de
  - http.webadmin
  - shell.scriptcmd
- Utiliser le SCR pour la gestion des liaisons et du cycle de vie

OSGi

Les Outils

Les outils

- Eclipse PDE
- Maven Bundle Plugin
- Bindex
- BND
- Console OSGi
- Unit testing

Eclipse PDE

- Since Eclipse 3, Plugins are OSGi bundles
- TODO

BND *(http://www.aqute.biz/Code/Bnd)*

- Tool to create and diagnose OSGi R4 bundles
  - **Available as Command line tool, Eclipse plugin, Ant Task, Maven Plugin**
- Key functions
  - Compute the classpath, import, export of a bundle
  - Show the manifest and JAR contents of a bundle
  - Wrap a JAR so that it becomes a bundle
  - Create a Bundle from a specification and a class path
  - Verify the validity of the manifest entries
- Examples
  - wrap an existing jar
    - *java -jar bnd.jar* **wrap** -output myOSGi.jar myPlain.jar
  - More TODO

BND – Directives

- Export-Package LIST of PATTERN
  - lists the packages that the bundle should export, and thus contain.
- Include-Resource LIST of iclause
  - makes it possible to include arbitrary resources; it contains a list of resource paths. See Include Resource.
- Private-Package LIST of PATTERN
  - lists the packages that the bundle should contain but not export. See Private Package.
- Import-Package LIST of PATTERN
  - lists the packages that are required by the contained packages. See Import Package.
- Conditional-Package LIST of PATTERN experimental
  - Works as private package but will only include the packages when they are imported. When this header is used, bnd will recursively add packages that match the patterns until there are no more additions.
- Bundle-
- SymbolicName
  - The default is the name of the main bnd file, or if the main bnd file is called bnd.bnd, it will be the name of the directory of the bnd file. An interesting variable is ${project} that will be set to this default name.
- Bundle-Name
  - If the Bundle-Name is not set, it will default to the Bundle-SymbolicName.
- Bundle-
- ManifestVersion 2
  - The Bundle-ManifestVersion is always set to 2, there is no way to override this.
- Bundle-Version VERSION
  - The version of the bundle. If no such header is provided, a version of 0 will be set.
- Service-Component LIST of component

BND - Directives

Operator @ d'expansion de jarfiles

```
<Include-Resource>
  @hexdump.jar,
  @libdbus-java.jar,
  @unix-*.jar,
  lib/libunix-java.so
</Include-Resource>
<Bundle-NativeCode>
  libunix-java.so;osname=Linux;processor=x86
</Bundle-NativeCode>
```

BND - Directives

Macro findpath

- **${findpath;regexp[;replacement]}**

- **traverse the files in the built jar file and check each path with the regular expression. If the regexp matches it puts it on a comma separated list. This allows us to construct the Bundle-Classpath from the lib directory in the built jar.**

```
<Include-Resource>lib=lib</Include-Resource>
<Bundle-Classpath>${findpath;lib/.*.jar}</Bundle-Classpath>
<Bundle-NativeCode>
  libunix-java.so;osname=Linux;processor=x86
</Bundle-NativeCode>
```

Maven

- Tool for building and managing Java projects
  - Standard organization of the project directory
  - Repositories (local and remote, public and private) of artifacts
  - Project description in pom.xml (Project Object Model)
  - Set of predefined commands
    - mvn clean
    - mvn install
    - mvn eclipse:eclipse
- Extensibles by plugins
  - MOJO (in Java, Groovy, …)
  - For OSGi, Maven Bundle Plugin (Apache Felix)

Maven Bundle Plugin (Apache Felix)

http://felix.apache.org/site/maven-bundle-plugin-bnd.html

- **Maven plugin to package a Java (Maven) project as a OSGi R4 bundle**
  - **Use BND tools (and directives) to calculate import/export packages**
- **Example of pom.xml**

```
…
<build>
   <plugins>
    <plugin>
     <groupId>org.apache.felix</groupId>
     <artifactId>maven-bundle-plugin</artifactId>
     <extensions>true</extensions>
     <configuration>
      <instructions>
       <Import-Package>*</Import-Package>
       <Private-Package>${pom.artifactId}.*</Private-Package>
       <Export-Package>${pom.artifactId};version=1.0.0</Export-Package>
       <Bundle-Activator>${pom.artifactId}.impl.Activator</Bundle-Activator>
       <Export-Service>org.osgi.service.event.EventHandler</Export-Service>
       <Import-Service>org.osgi.service.log.LogService</Import-Service>
…
```

Maven Archetypes for OSGi projects

- **Maven archetype**
  - Parameterized template to start quickly a Maven-enabled project
- **Archetypes for OSGi bundle projects**
  - General purpose
    - http://gforge.inria.fr/scm/?group_id=526
  - SpringOSGi
    - spring-osgi-bundle-archetype
  - …

Pax Construct

http://wiki.ops4j.org/confluence/display/ops4j/Pax+Construct

- **Templates for kick-start bundle development**
  - Project / bundles / common techniques
- **Maven pax plugin**
  - Manages project builds file
    - mvn clean install pax:provision
  - + interactive scripts (Windows only)
- **Example**

```
pax-create-project
    -p com.example.client
    -n simple-test-client
    -a kxml2
    -v 0.1.0-SNAPSHOT
    -Dunpack=true
```

Simple Pax use case
a web based OSGi application

- pax-create-project -g simple.project -a osgi-web-app
- cd osgi-web-app
- pax-wrap-jar -g javax.servlet -a servlet-api -v 2.5
- pax-import-bundle -g org.ops4j.pax.logging -a api -v 0.9.4
- pax-import-bundle -g org.ops4j.pax.logging -a jcl -v 0.9.4
- pax-import-bundle -g org.ops4j.pax.logging -a slf4j -v 0.9.4
- pax-import-bundle -g org.ungoverned.osgi.bundle -a http -v 1.1.2
- pax-create-bundle -p my.osgi.code.myBundle -n myBundle
- mvn install pax:provision

Pax Runner

http://wiki.ops4j.org/confluence/display/ops4j/Pax+Runner

- Provides URL handlers (mvn,wrap,classpath) to deploy bundles or simple jarfiles from a Maven 2 repository
- Provides scripts to setup initial provisioning config files for Felix, Equinox and KF

BIndex *(http://www2.osgi.org/Repository/BIndex)*

- Tool to create RFC-0112 Bundle Repository indexes from bundles manifests.
  - **Available as Command line tool and Ant Task**
  - **Maven plugin : org.apache.felix:maven-obr-plugin**
    - **Incremental index update**
- Options

```
java -jar bindex.jar
        [-help]
        [-r repository.xml (.zip ext is ok)]
        [-l file:license.html ]
        [-t http://w.com/servl?s=%s&v=%v %s=symbolic-name %v=version %p=relative path %f=name]
        [-q (quiet) ]
        <jar file>*
```

- the RFC 112
  - http://www2.osgi.org/div/rfc-0112_BundleRepository.pdf

- **bushel** ***http://code.google.com/p/bushel/***
  - ***Bushel is a simple utility to convert ad-hoc OSGi bundles into a local Ivy repository.***

Unit testing

- Motivation
  - Unit testing of individual bundles (w w/o services)
  - Before packaging and deployment
  - Running of a OSGi platform (without rebooting)
  - Integration to builders (Maven Surefire) and continuous integration
- Architectures
  - Whiteboard pattern
  - Extended pattern
- Example
  - JUnit4OSGi
    - http://felix.apache.org/site/apache-felix-ipojo-junit4osgi.html
  - Pax Exam
    - http://wiki.ops4j.org/display/paxexam/Pax+Exam

OSGi and AOT compilation

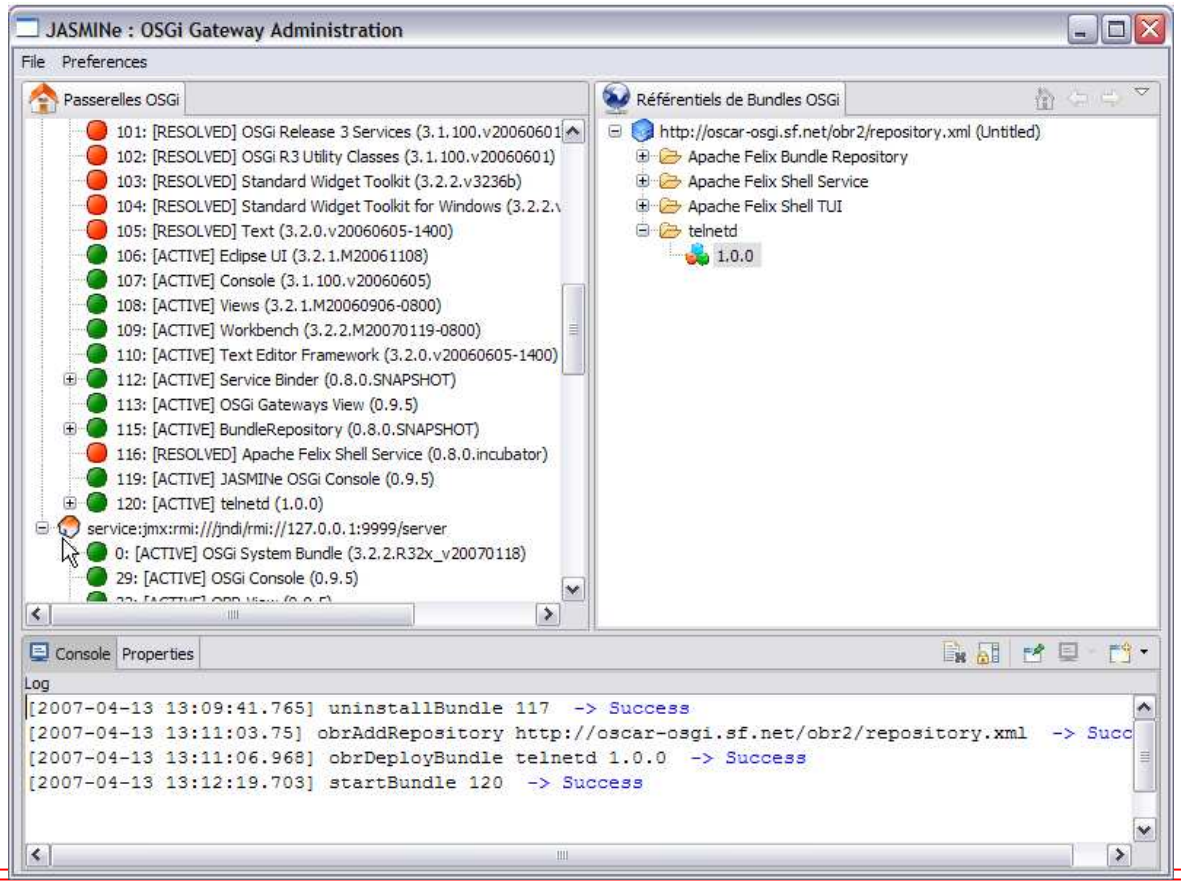
- **Motivations**
  - **“Accelerate”and protect OSGi applications**
    - **On-the-fly code injection is harder**
    - **Retrocompilation is harder**
  - **Reduce the Runtime size**
- **Tools**
  - **Excelsior JET**
    - **Testing with RCP bundles**

OSGi graphical consoles

- Felix mOSGi console
- KF
- Jasmine OSGi console
- Prosyst mConsole
- …

Jasmine OSGi Console
(http://jasmine.objectweb.org/doc/console/jasmine.htm)

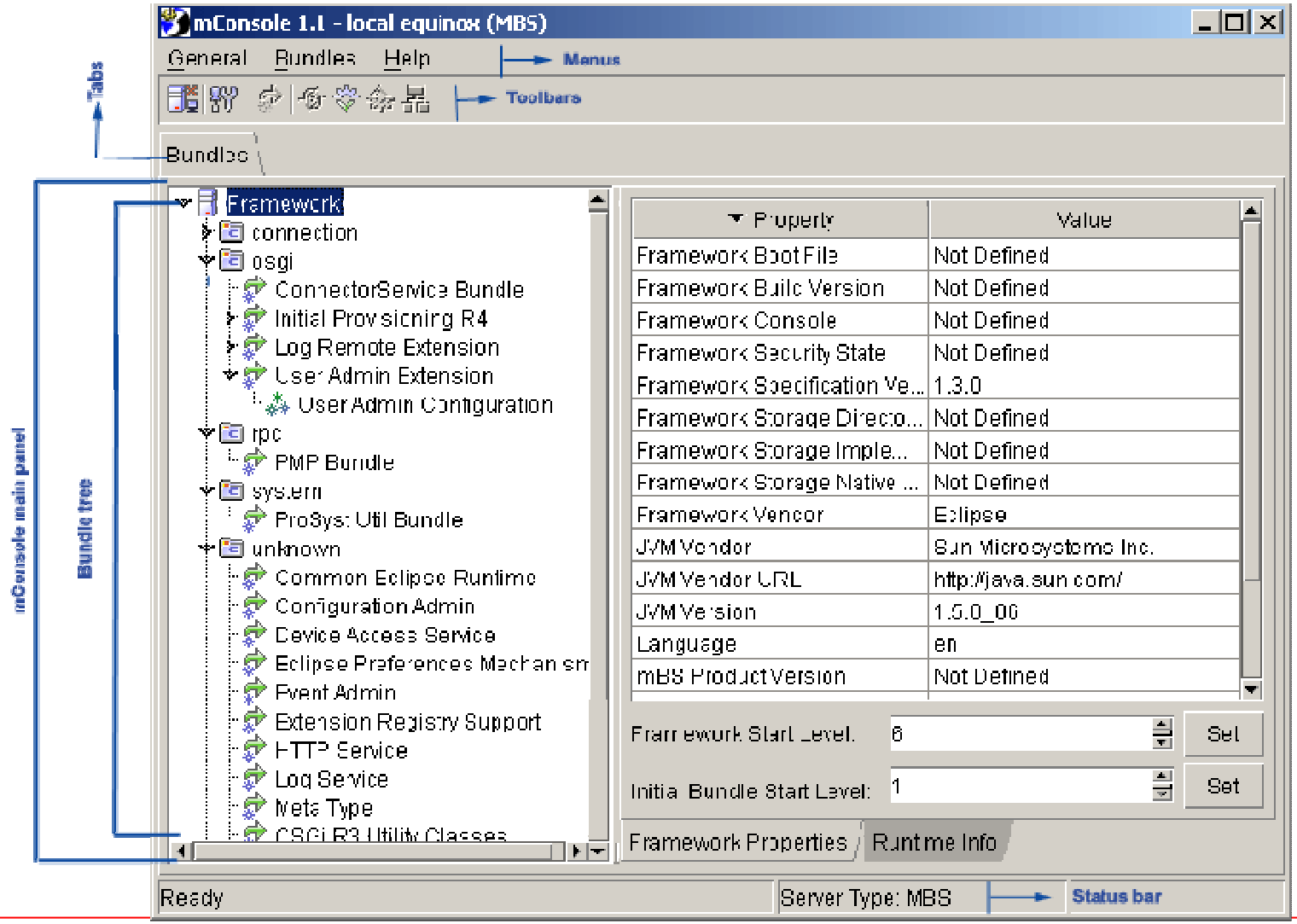
- **Eclipse RCP & Plugin to manage OSGi platforms**
  - **connection to the platform with JMX, WS, UPnP, …**
  - **OBR plugin**



Apache Felix mOSGi console

- TODO

Prosyst mConsole



Bonus Track

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

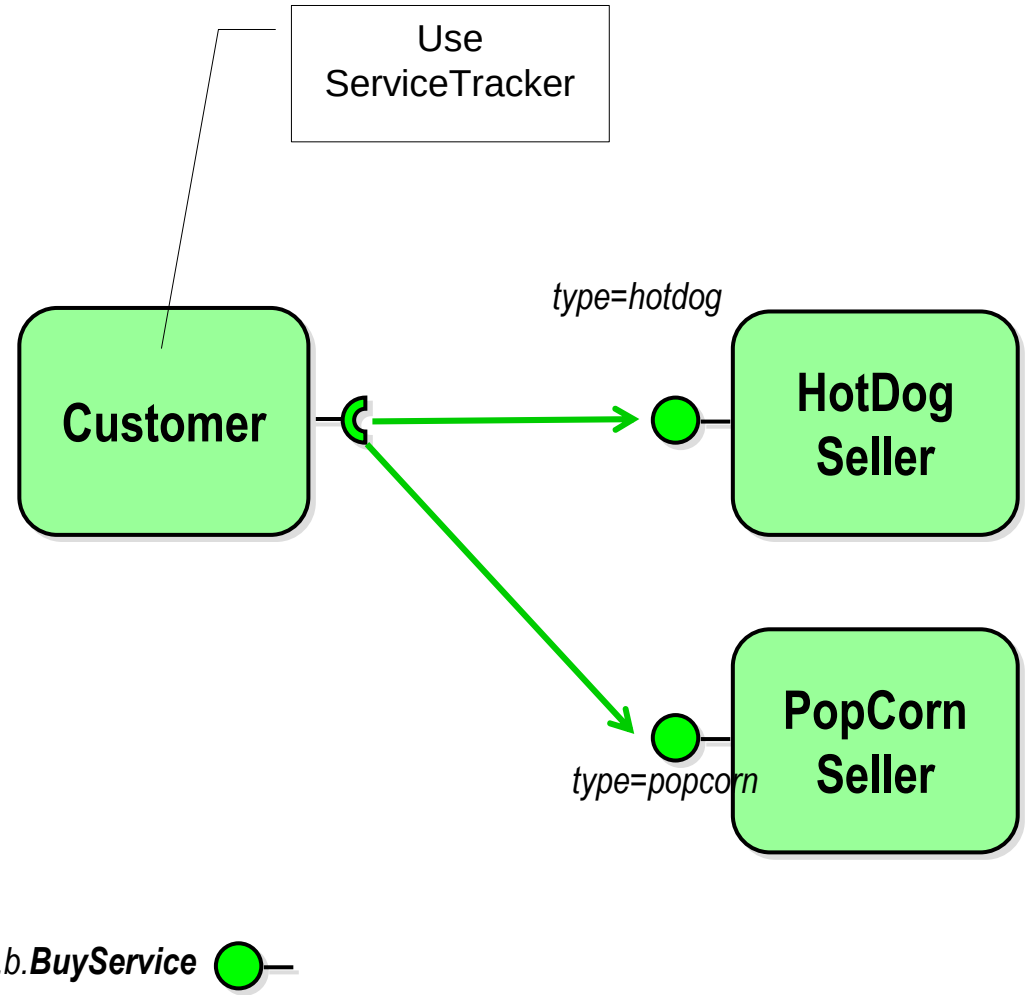
- **D'aprés Rick Hall**
- **Rational**
  - Java programming language needs better support for hierarchical, modular organization
    - Primarily to support information hiding
    - Java packages inadequate for this purpose
    - Only two levels of visibility: internal to the package or public to everyone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

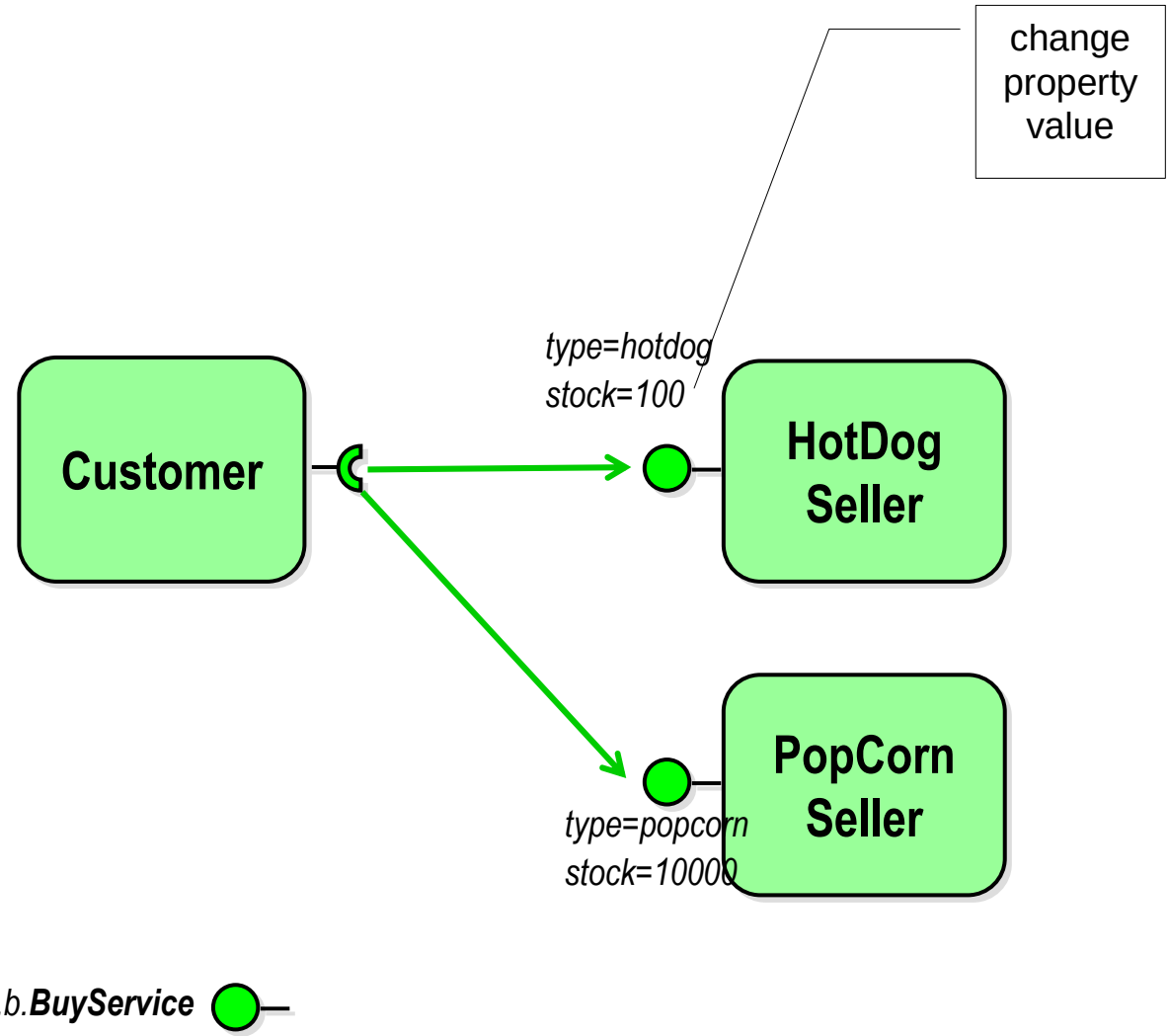
Application à développer

Ecole d’été OSGi 2007

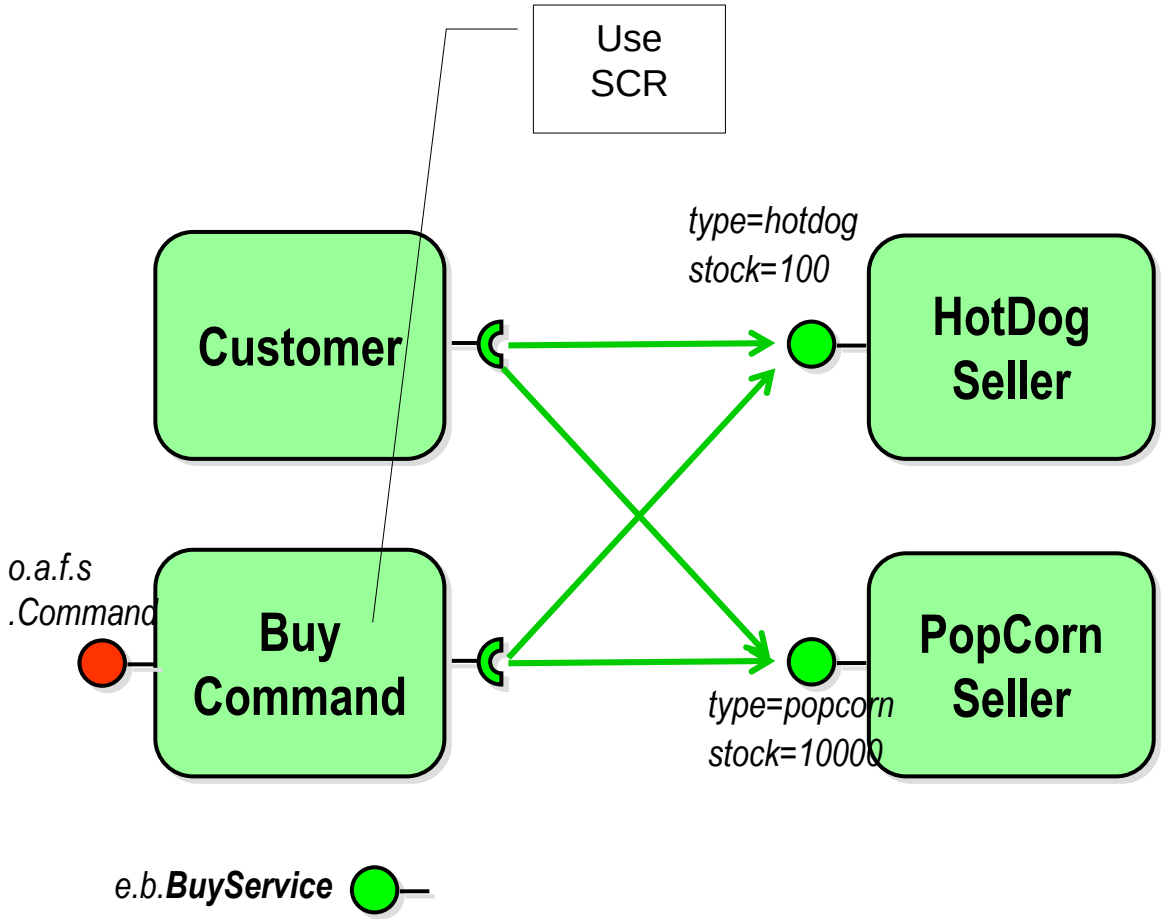
Application 1



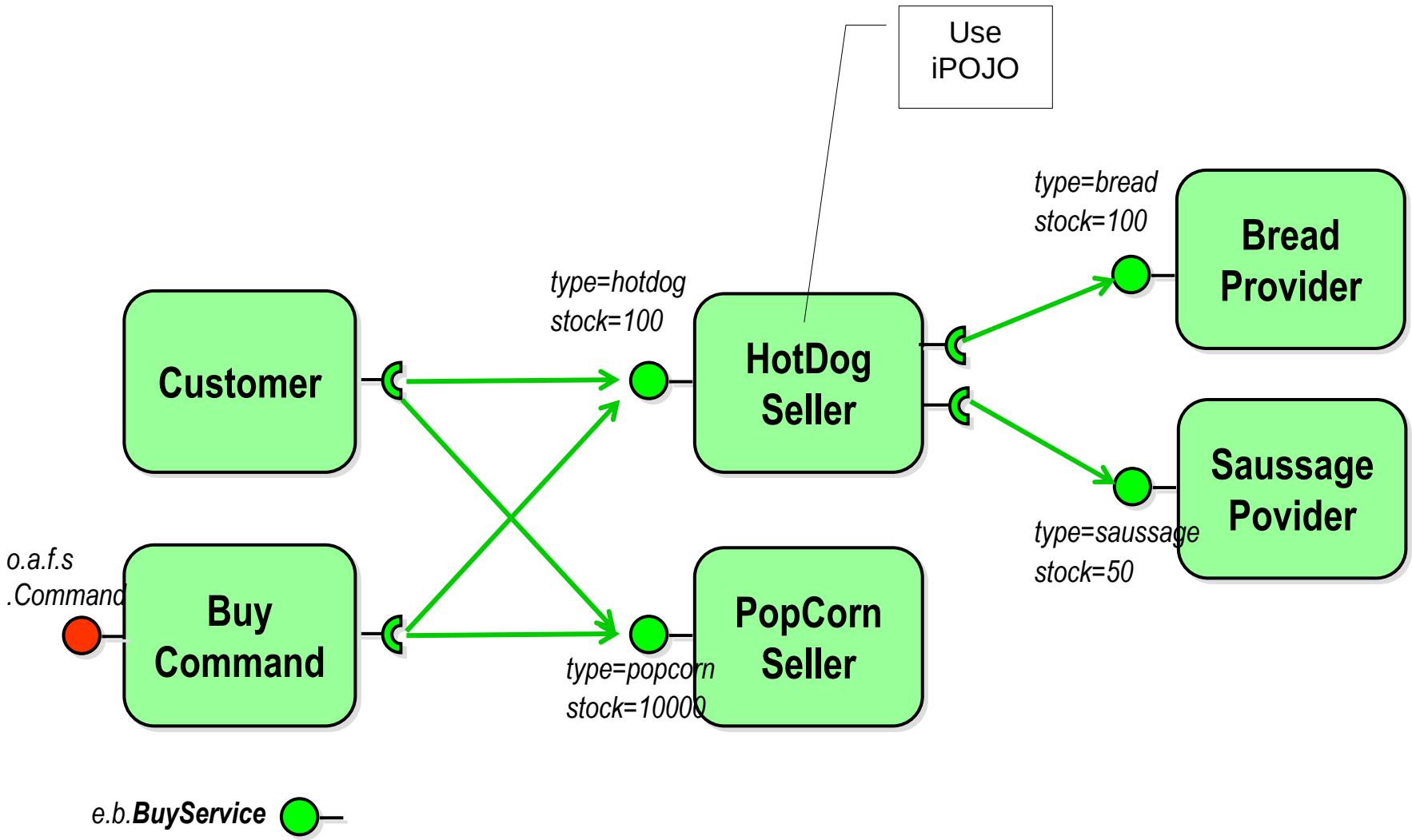
Application 2



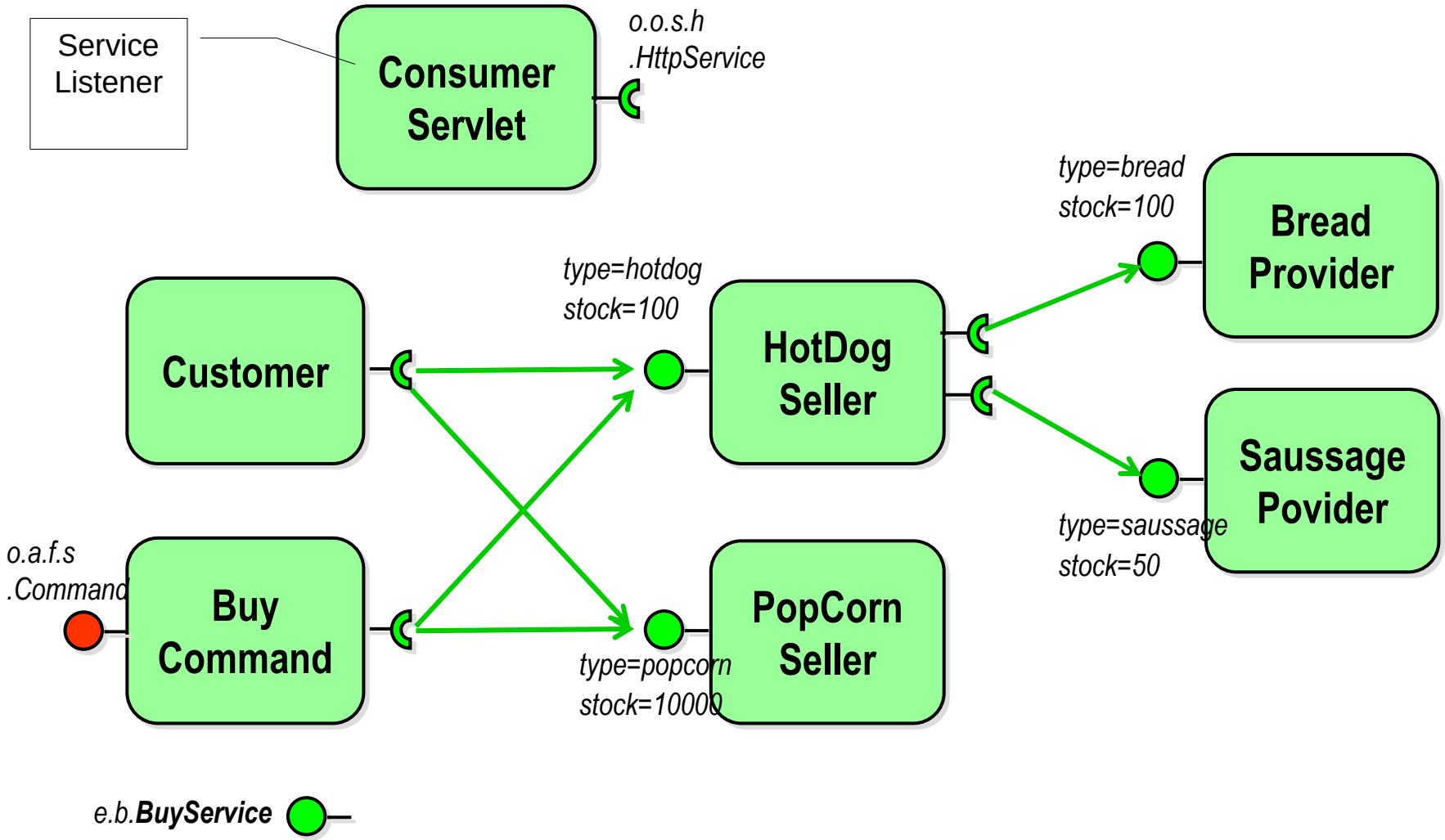
Application 3



Application 4



Application 5

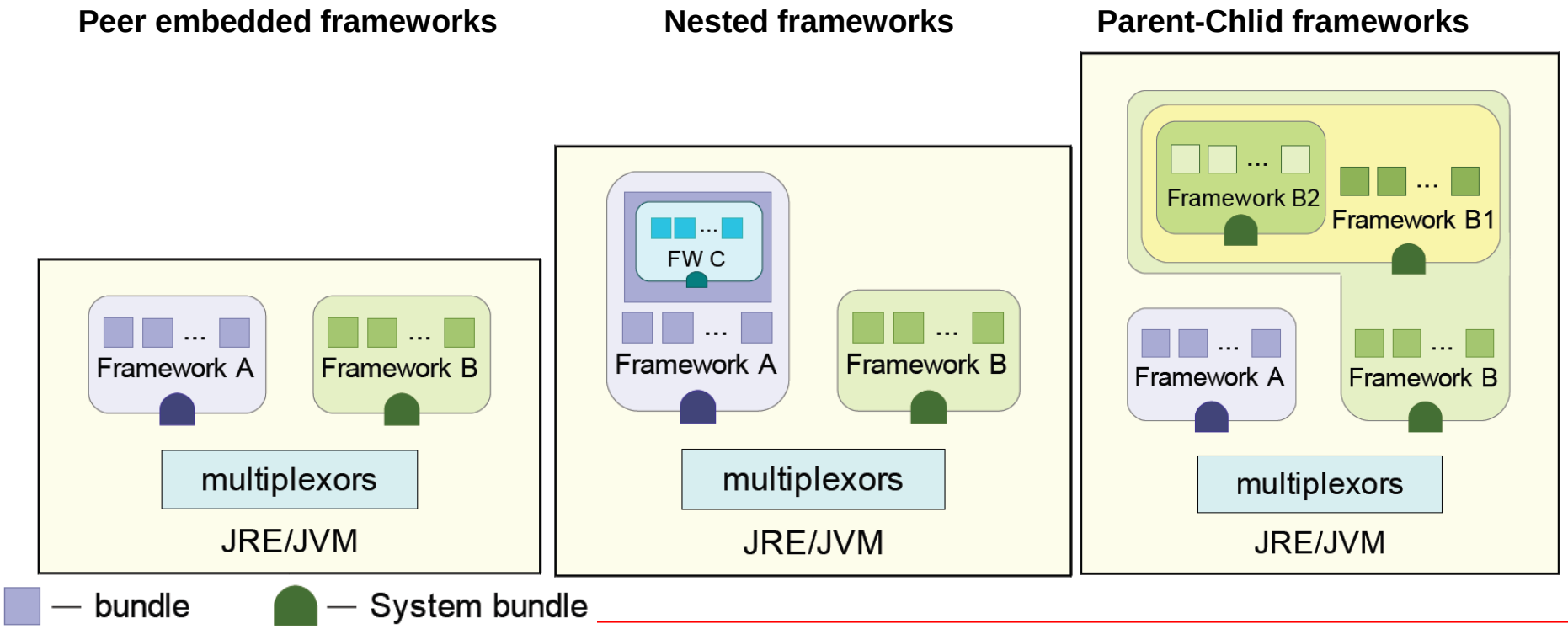


RFC 138 Multiple Frameworks In One JVM

- **Motivations**
- **Limitations**
  - Multiplex singleton factories (URLHandler, XML, …)
  - Multiplex System properties
  - Security Manager for Conditional Permission Admin
- **Solutions**
  - Several FW (several vendors) hosted in the same JVM

RFC 138 Multiple Frameworks In One JVM

Configurations



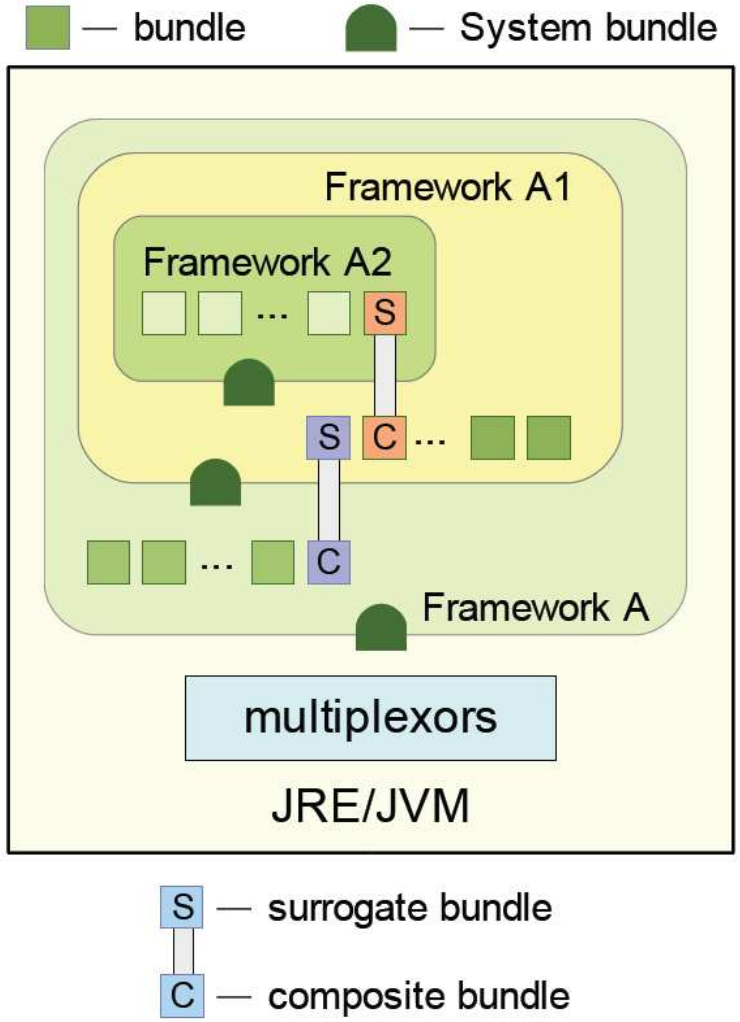
RFC 138

Surrogate Bundle

- Represents the parent in the child FW

Composite Bundle

- Represents a child in a parent FW



Lifecycle

Import / Export

- Package import
- Package export

Import / Export

- **Parent Service import**
  - **Filtering** CompositeServiceFilter-Import
- **Child Service export (to the parent)**
  - **Filtering** CompositeServiceFilter-Export
- **service.id (globally unique)**

RFC 139 Distributed OSGi