
Des architectures orientées services aux SOAs dynamiques

OSGI

Plan

- Introduction
- Motivations
- Éléments techniques d'OSGI
- Bundle et Service
- Manifest
- Bundles et services standards
- Démonos et revues de code

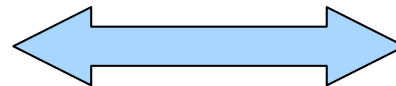
1. Vision du marché

- De plus en plus de périphérique sont intelligents et adaptables
 - Automobile
 - Téléphonie mobile
 - Home Cinéma
- Les réseaux de données sont omniprésents
- Tous ces composants nécessitent des couches logicielles de plus en plus complexes

1. Opportunité commerciale

- La technologie OSGi :
 - Rend les composants logiciels faciles à gérer
 - Permet la portabilité des services à valeur ajouté à travers les marchés et les équipements
 - Permet la mise en place de nouveaux business-models

MultiVendeurs, Ouverture

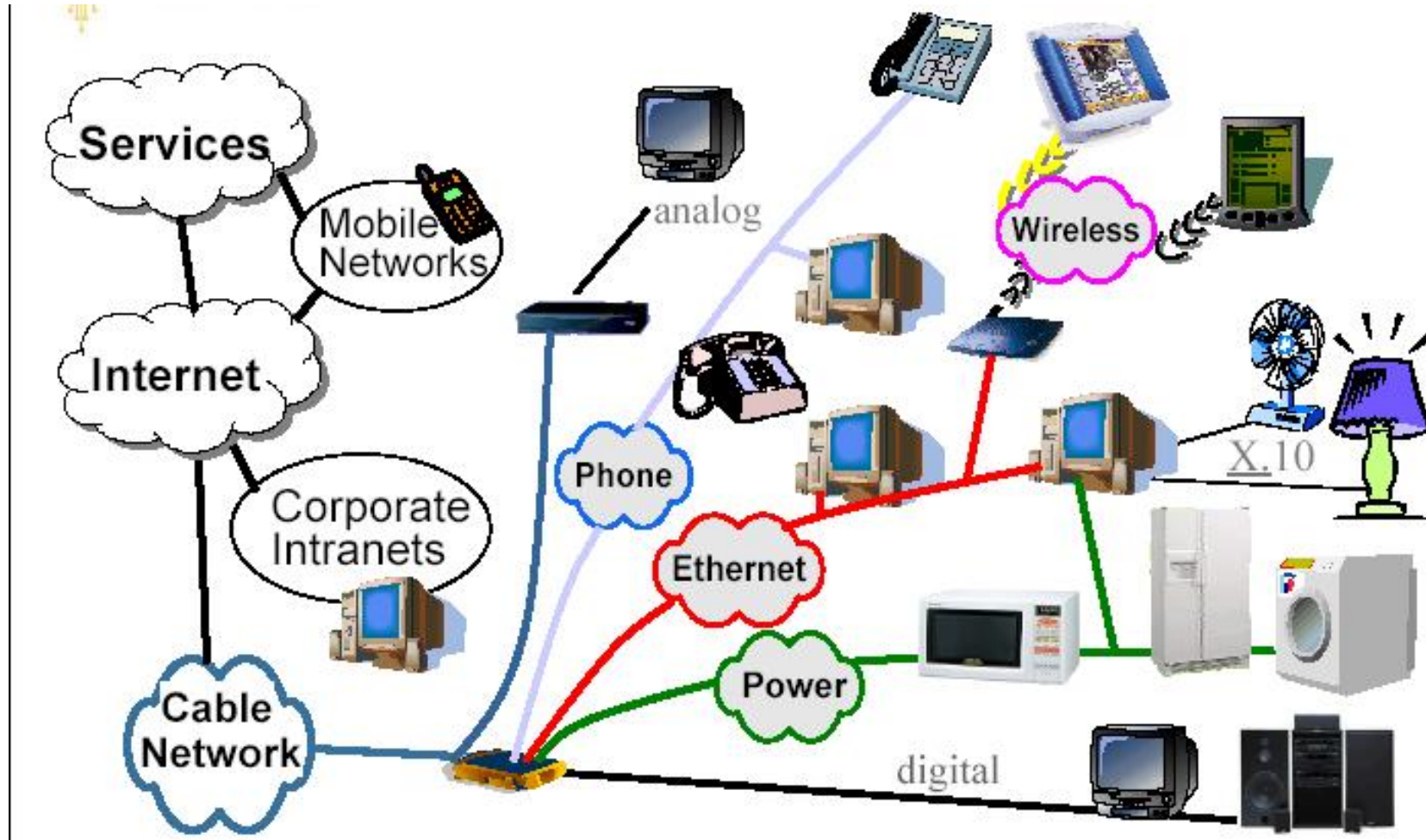


Solution propriétaire

1. Marchés ciblés

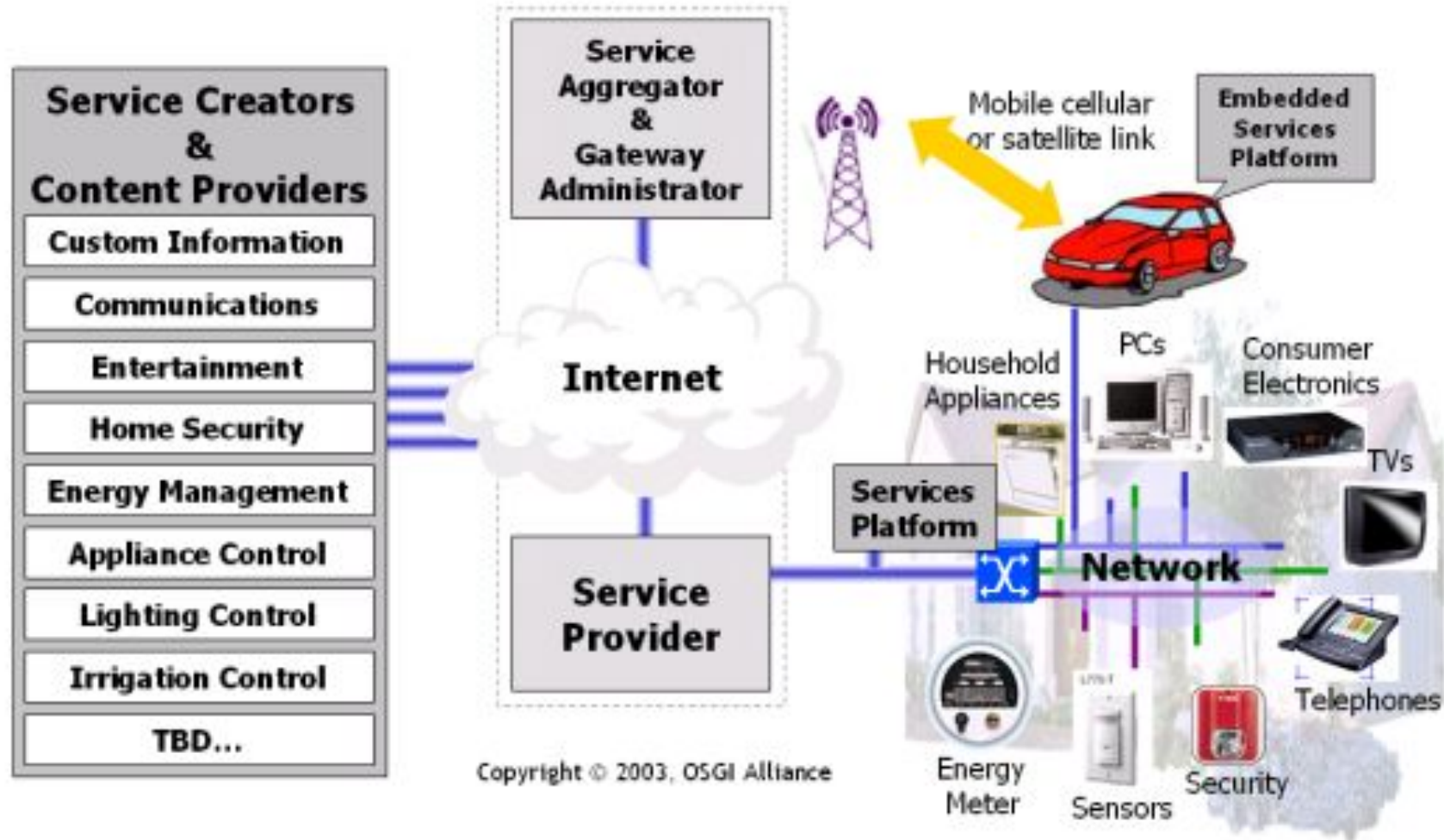
- Equipements Mobiles
 - 180Million d'éléments/an
 - Premier équipement OSGi 2005 (Motorola)
- Equipement automobile
 - 60Million voiture/an
 - OSGi est déjà présent dans certains véhicules (BMW)
- Réseaux résidentiels
 - 190Million d'équipements blancs, 40Million de modem et SetTop ADSL
 - OSGi choisi par siemens, Motorola, Philips...

1. Exemple



Cable Labs CableHome Project - Juillet 2001

1. Solutions OSGi de bout en bout



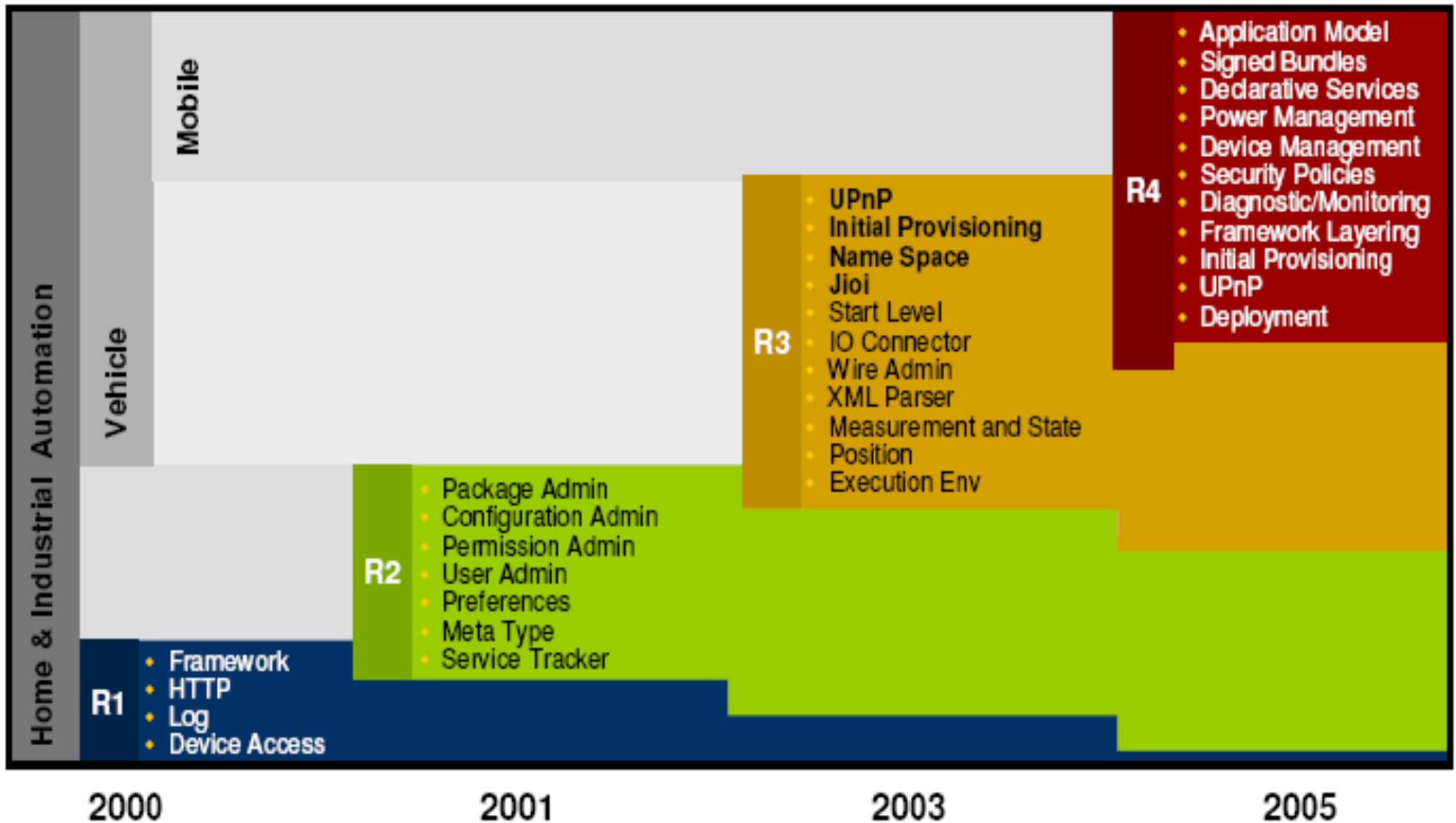
1. OSGi Alliance

- Organisation Internationale, à but non-lucratif formée pour développer et promouvoir des spécifications ouvertes pour la livraison par le réseau, de services administrés vers des périphériques dans la maison, la voiture ou d'autres environnements
- <http://www.osgi.org>

1. Qu'est ce que OSGi™ ?

- Corporation indépendante
 - fondé en Mars 1999
 - Soutenus par les acteurs majeurs des IT, home/building automation, telematics, (car automation), ...
 - de la téléphonie mobiles (Nokia et Motorola pilotent la R4)
 - et Eclipse pour les plugins de son IDE ! et maintenant Apache pour ses serveurs
- Spécification OSGi™
 - définit un framework qui permet à une diversité de services logiciels d'être chargés et exécutés dans un 'service gateway' (serveur embarqué) tel qu'une set top box, modem, PC ou une passerelle résidentielle dédiée.
- Releases
 - 22/11/1999: SUN transfère le JSR008 du JCP à OSGi
 - 1.0 : Mai 2000 (189 pages)
 - 2.0 : Octobre 2001 (288 pages)
 - 3.0 : Mars 2003 (602 pages)
 - 4.0: October 2005 (1000 pages)

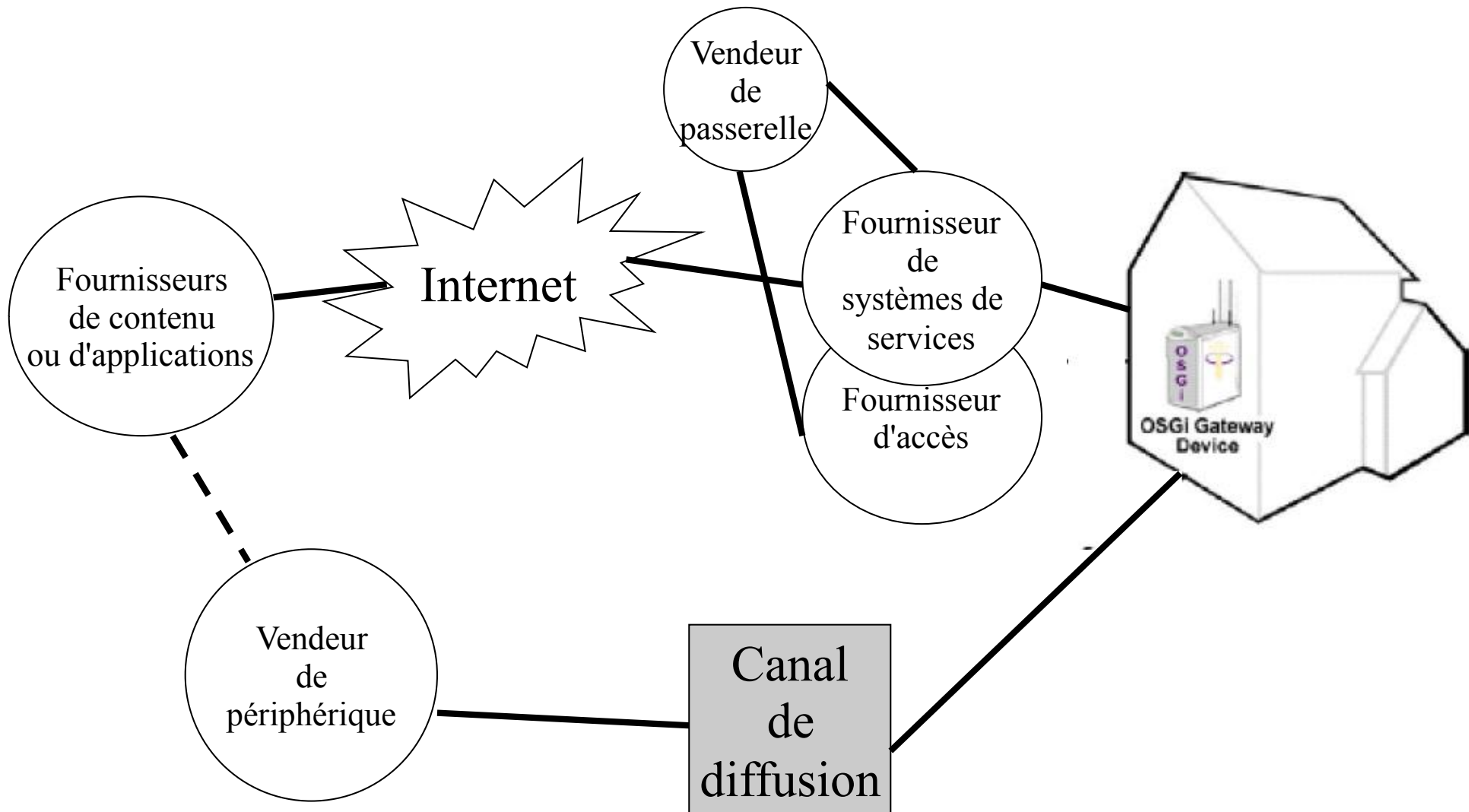
1. OSGi RoadMap



1. Points clés

- Mise à jour dynamique de logiciels
- Contrôle à distance
- Maintenance à distance
- Diagnostic distant
- Echange de données

1. Chaîne de valeurs OSGi



Motivations

Questions to ask

- Is there a need?
- What do we get?
- What does it cost?
 - How much do we need to change our code?
 - How much bloat does it add?
 - Any efficiency concerns?

What's wrong with standard java?

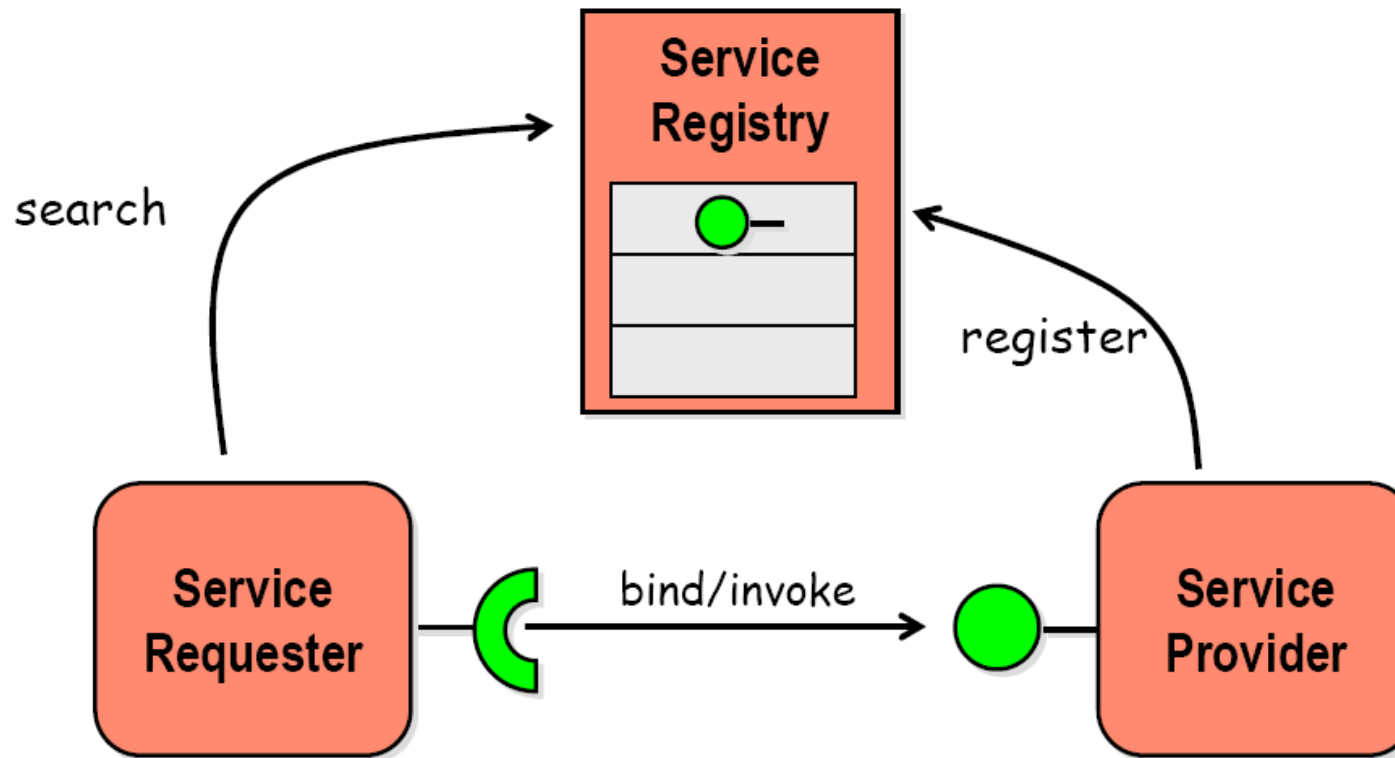
- Simplistic version handling
 - First found in class path
- Implicit dependences
 - Class path ordering
- Split packages by default
 - Could get mix of versions
- Limited support for dynamic code
 - Can use class loaders but quite low level

2. Motivations

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

2. Architecture orientée service (SOA)

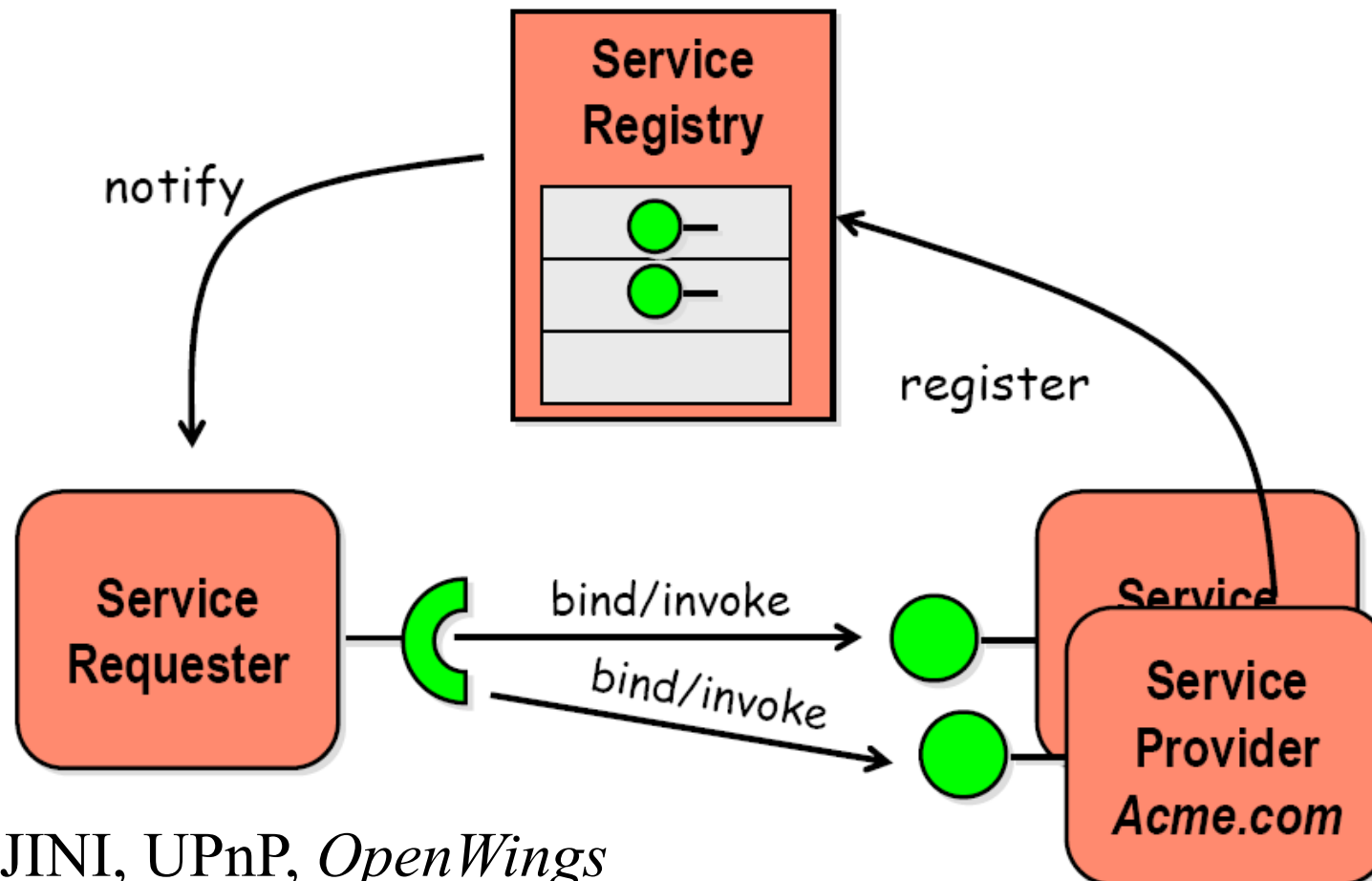
- Les services (contrats) ● sont « invariants »



- Web services

2. Rappel: Dynamic SOA

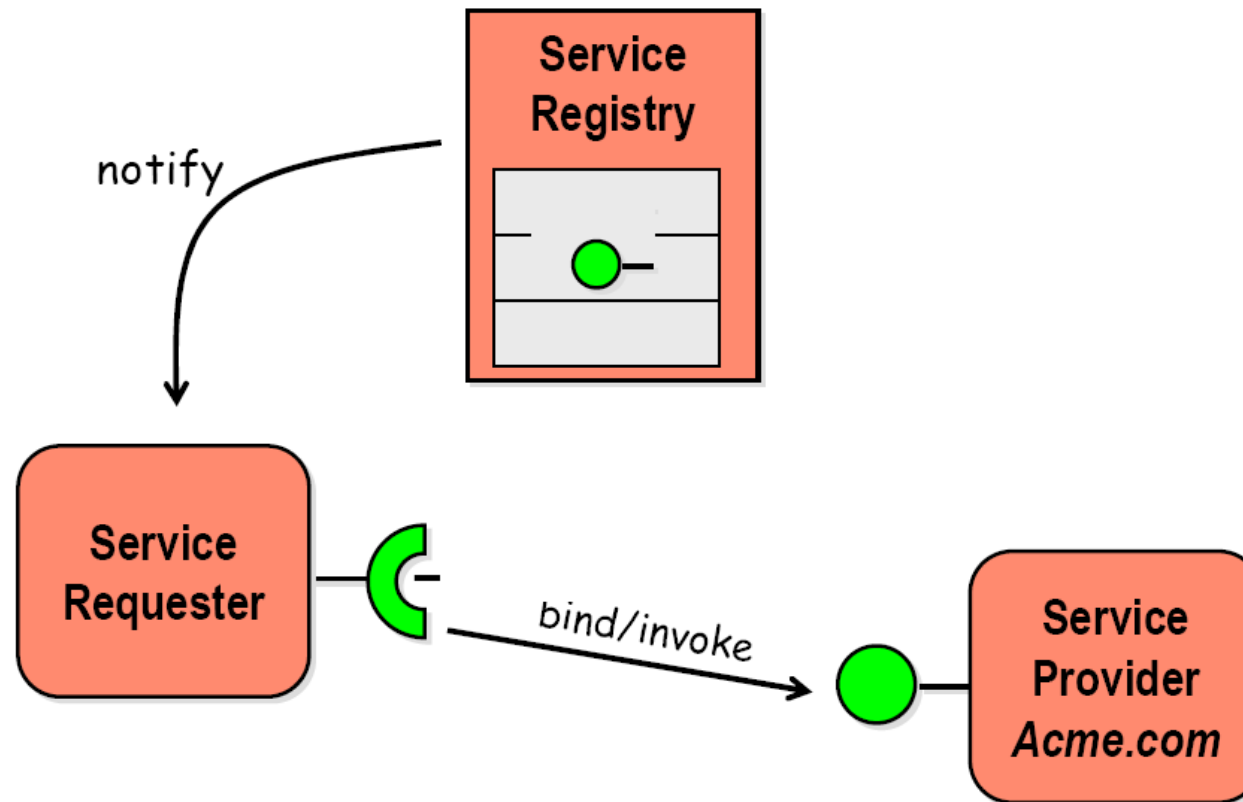
- Arrivée dynamique de nouveaux services



- JINI, UPnP, *OpenWings*
- OSGi, J2ME/CDC/PBP

2. Rappel: Dynamic SOA

- Retrait dynamique de nouveaux services



- JINI, UPnP, *OpenWings*
- OSGi, J2ME/CDC/PBP

2. Domaines d'application

- Systèmes embarqués
 - Véhicule de transport (*automotive*)
 - Passerelle résidentiel/domotique/immotique
 - Contrôle industriel
 - ...
- Cependant
 - Tout concepteur d'application est gagnant à distribuer son application comme un ensemble de bundles
 - Cela évite le casse tête du CLASSPATH, lib/ext du JRE ou J2SESDK, ...

 - Exemples : ANT, JEdit, JMF... OCF/javax.comm
 - Eclipse Equinox 3.0 utilise OGSi pour le chargement/déchargement des plugins (<http://www.eclipse.org/equinox/index.html>)

2. Résumé

- OSGi seule plate-forme de services pour des services administrés
- OSGi est le point d'intégration des offres réseau à la maison
- OSGi est la plate-forme de choix pour les leader de l'industrie
- OSGi a le soutien des fournisseurs de la chaîne de valeur
- OSGi est adopté par les fournisseurs de plate-forme et est déployé par les opérateurs
- OSGi est une proposition de services complémentaires pour les telco

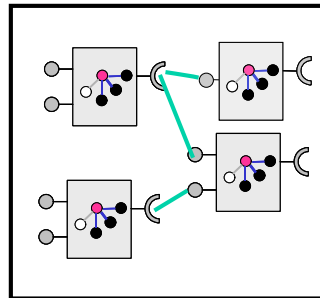
Eléments techniques d'OSGi

3. Eléments techniques d'OSGi

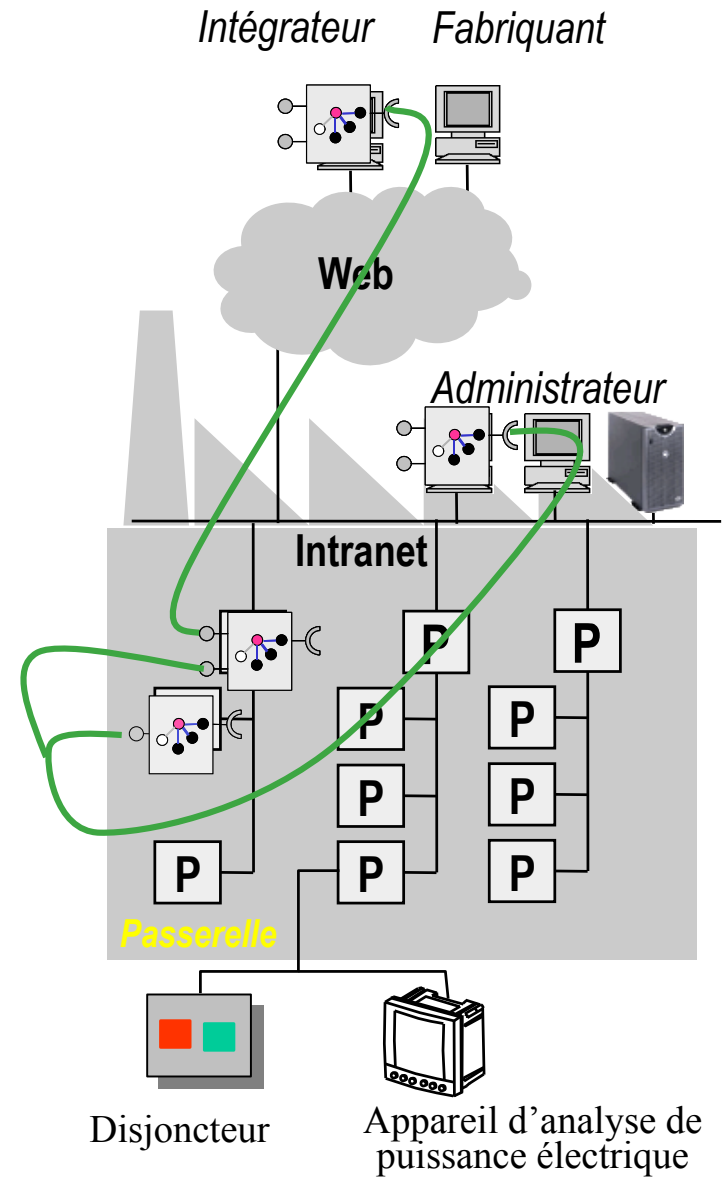
- Repose sur :
 - Java
 - La programmation orientée services
 - Trois couches : le bundle, le package, le service
- N'est pas :
 - Un système distribué (Pas d'invocation distante)
 - Seul le chargement de bundle peut se faire à distance

3. Une passerelle de services embarqués

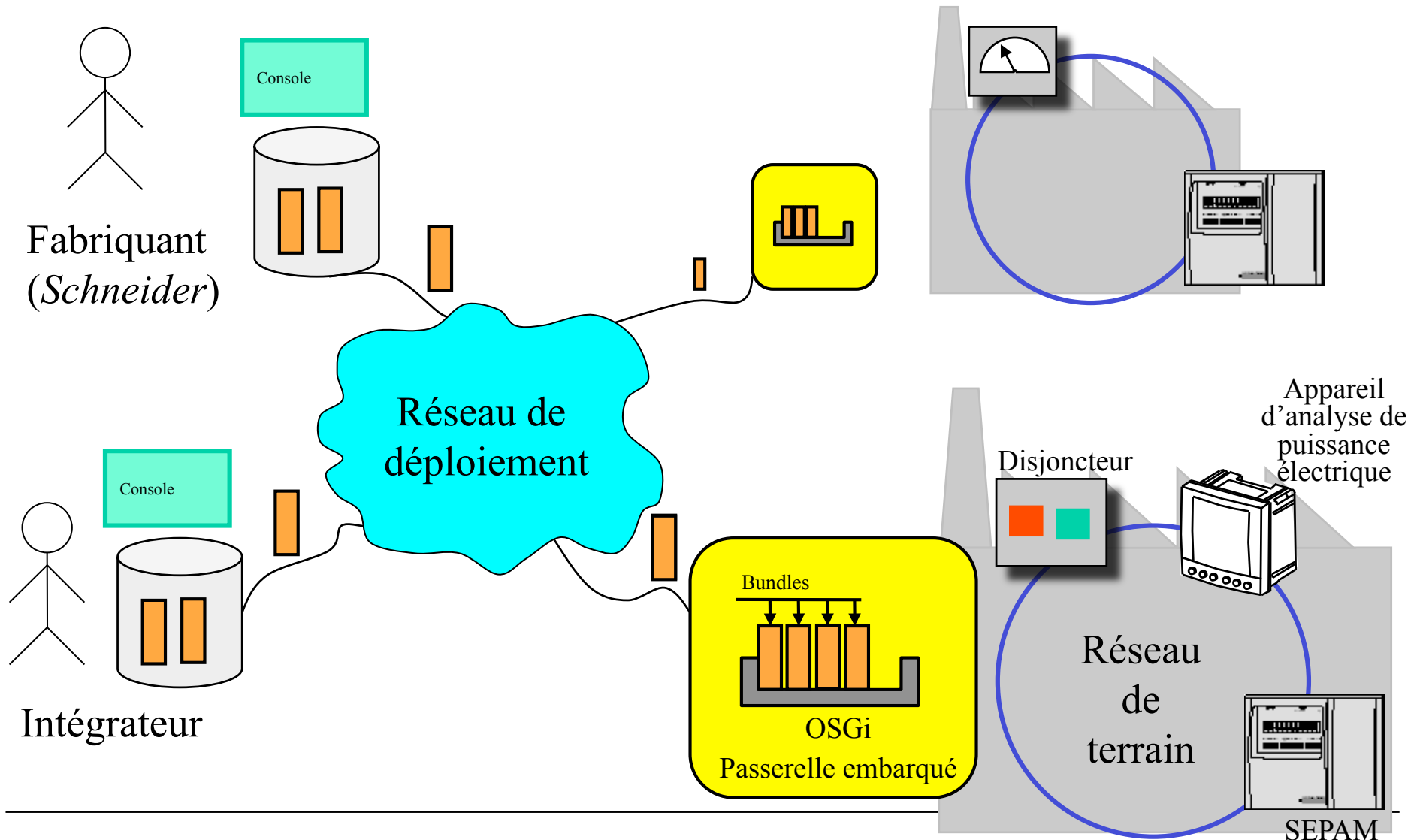
Un application



- ✓ Adaptable
- ✓ Maintenable
- ✓ Flexible
- ✓ Dynamique

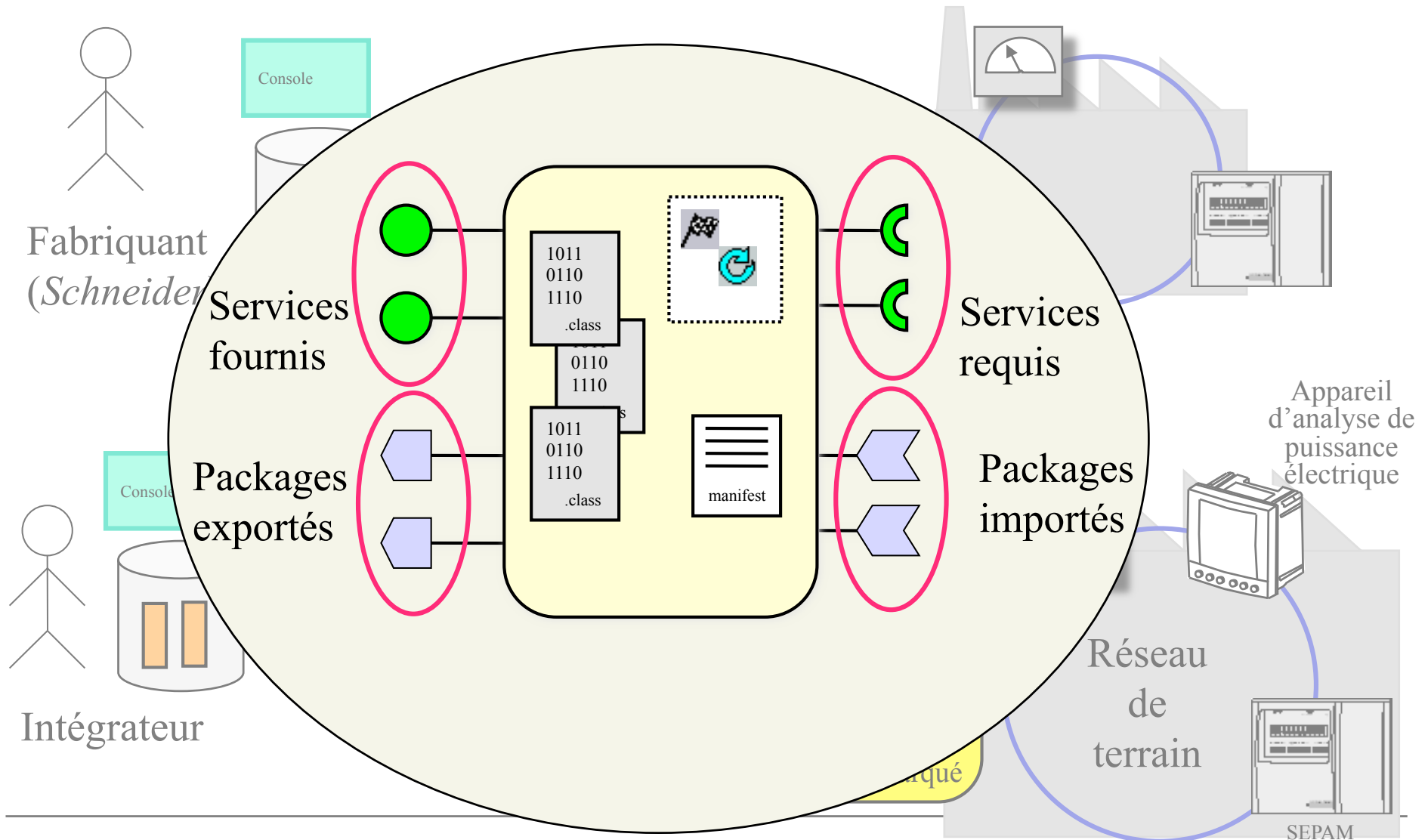


3. Une passerelle de services embarqués

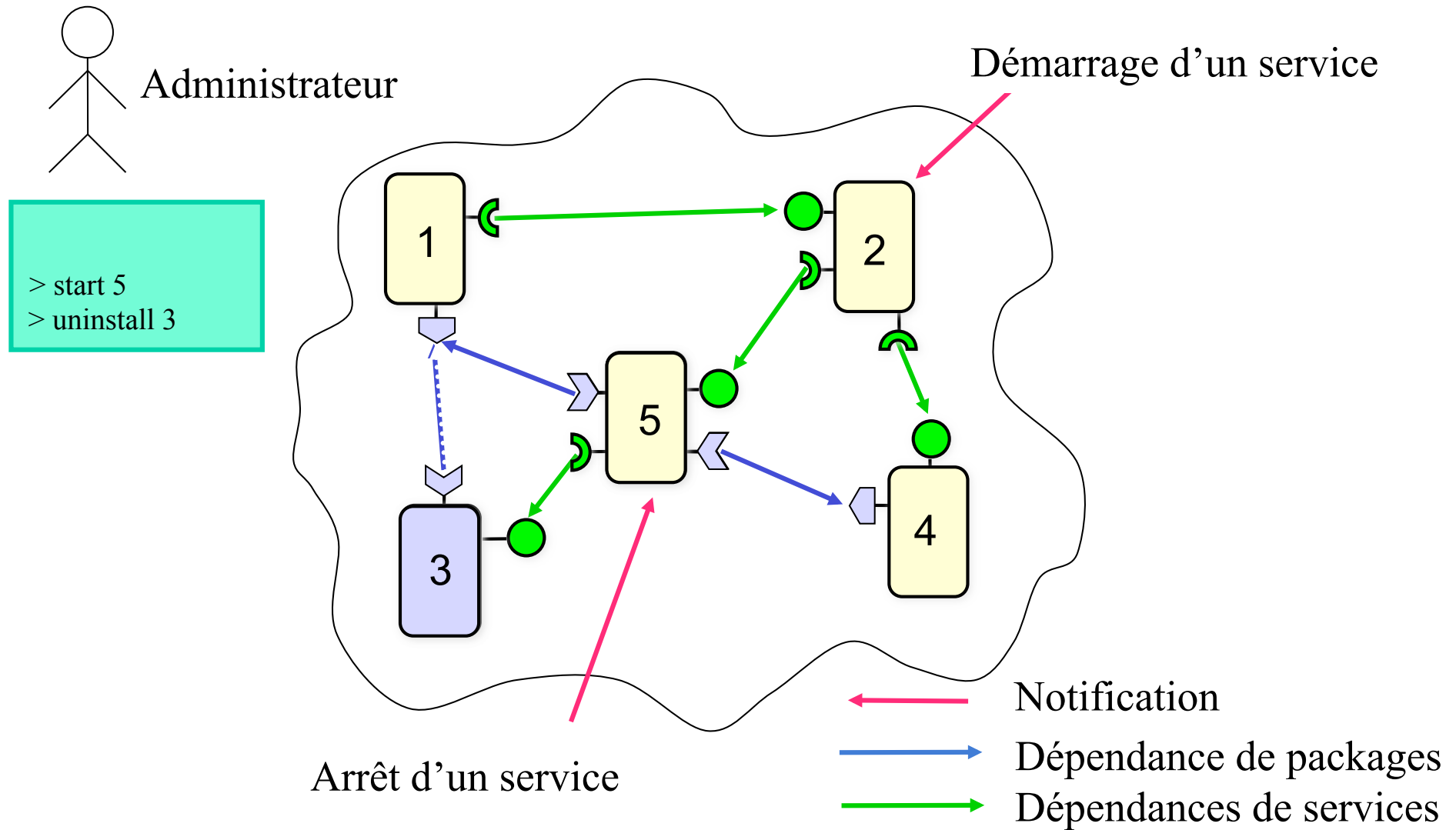


Bundle et Service

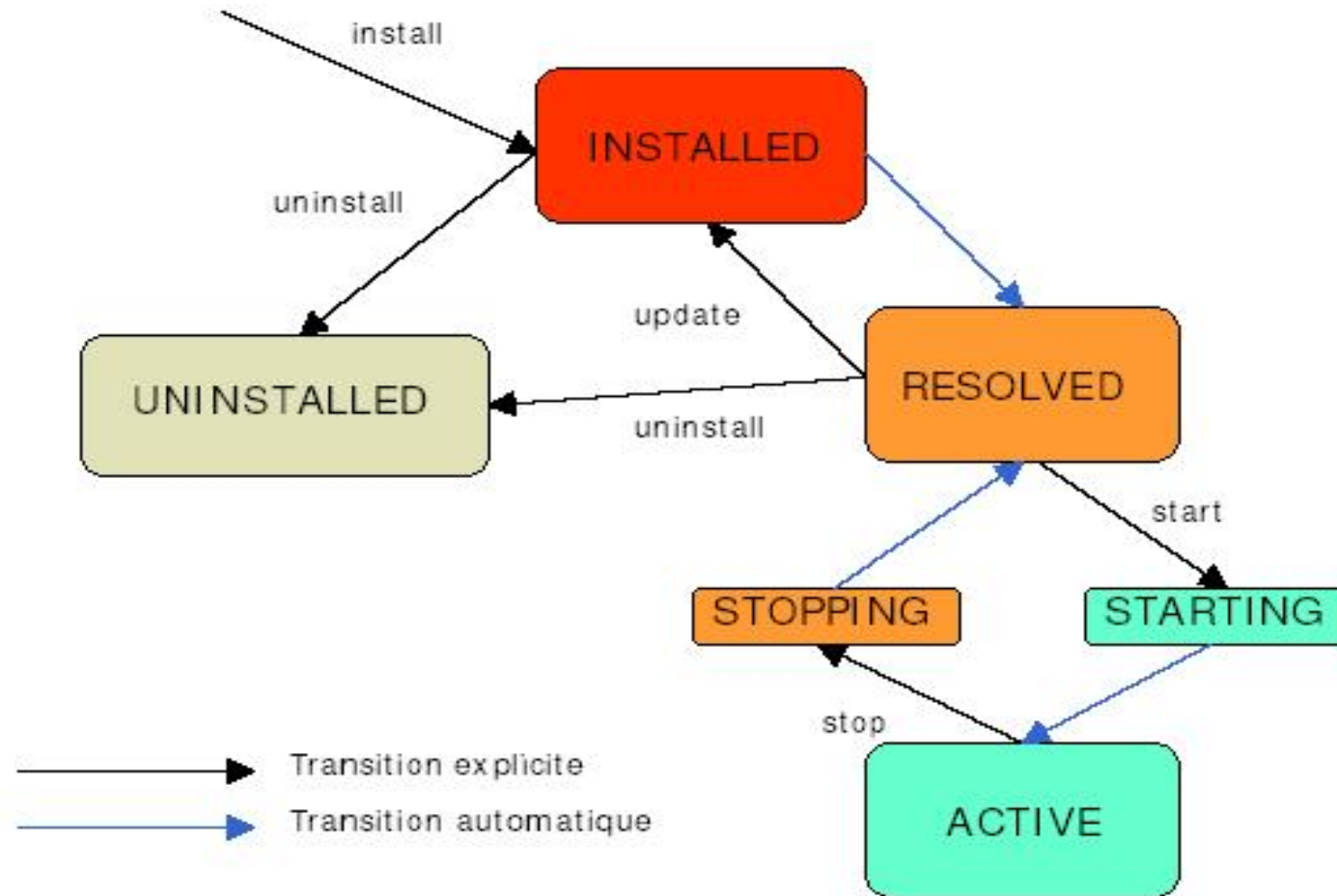
4.OSGi : structure d'un bundle



4. OSGi : Une application type

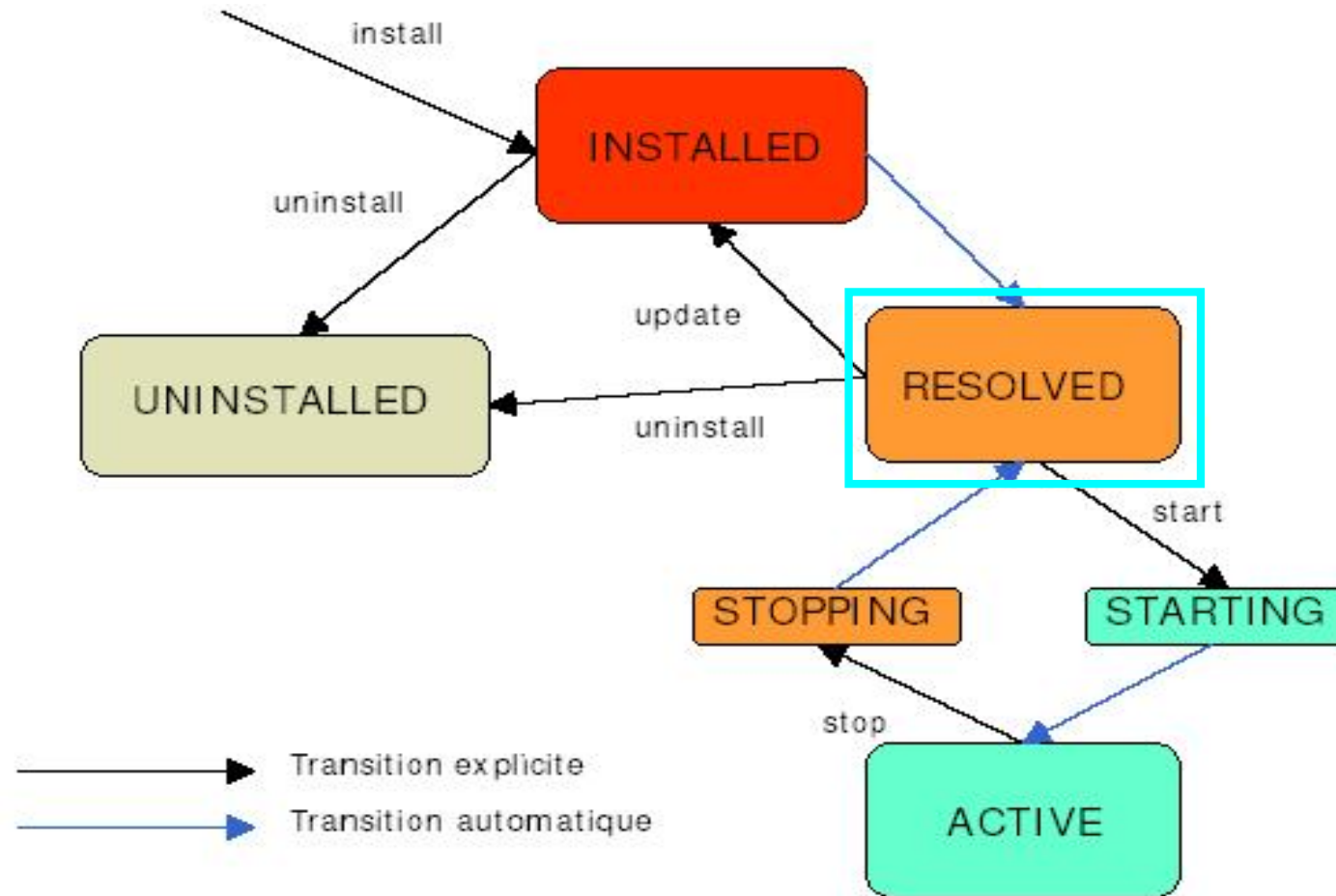


4. Cycle de vie d'un *bundle*



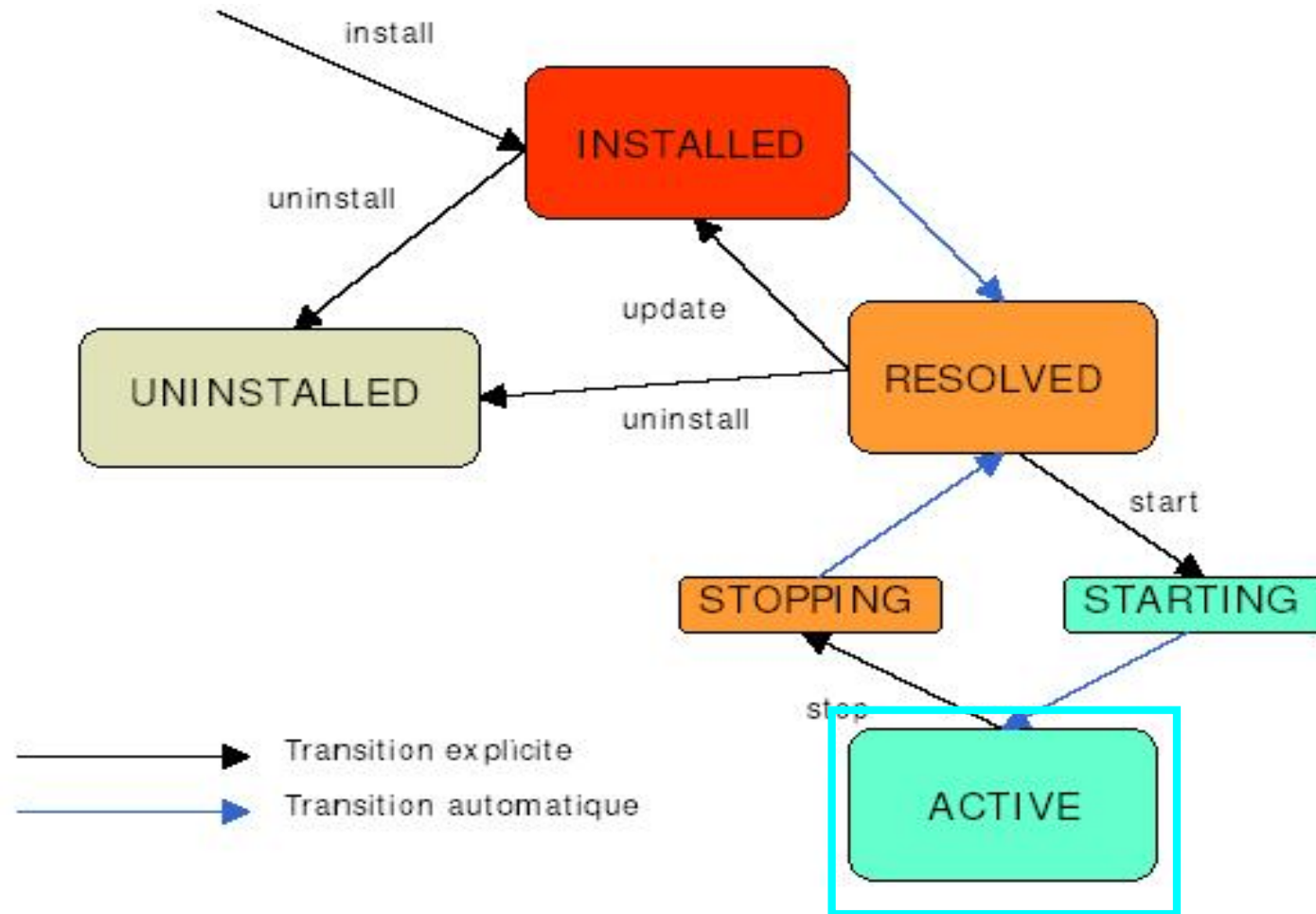
Cycle de vie d'un *bundle*

4. Cycle de vie d'un *bundle*



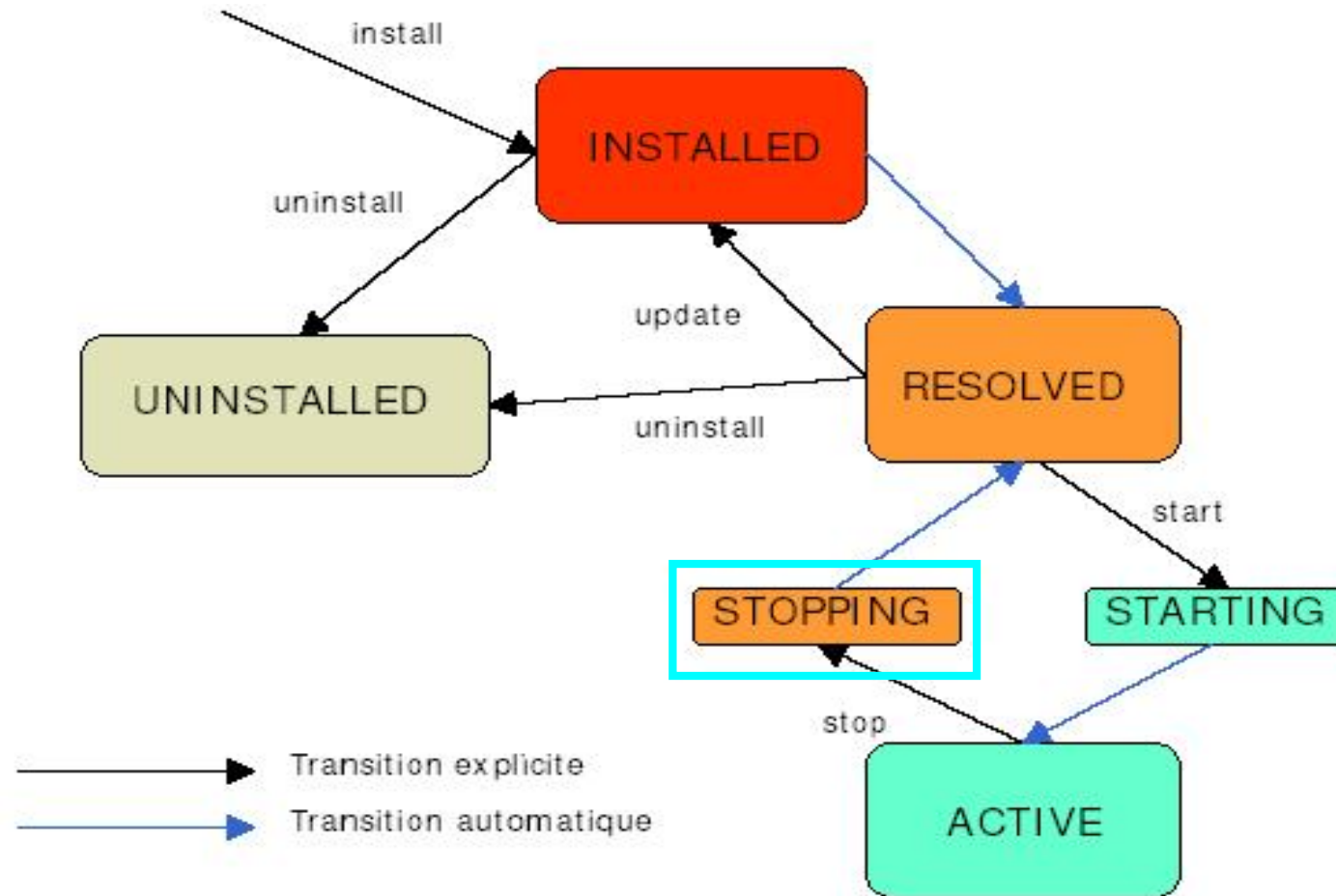
Cycle de vie d'un *bundle*

4. Cycle de vie d'un *bundle*



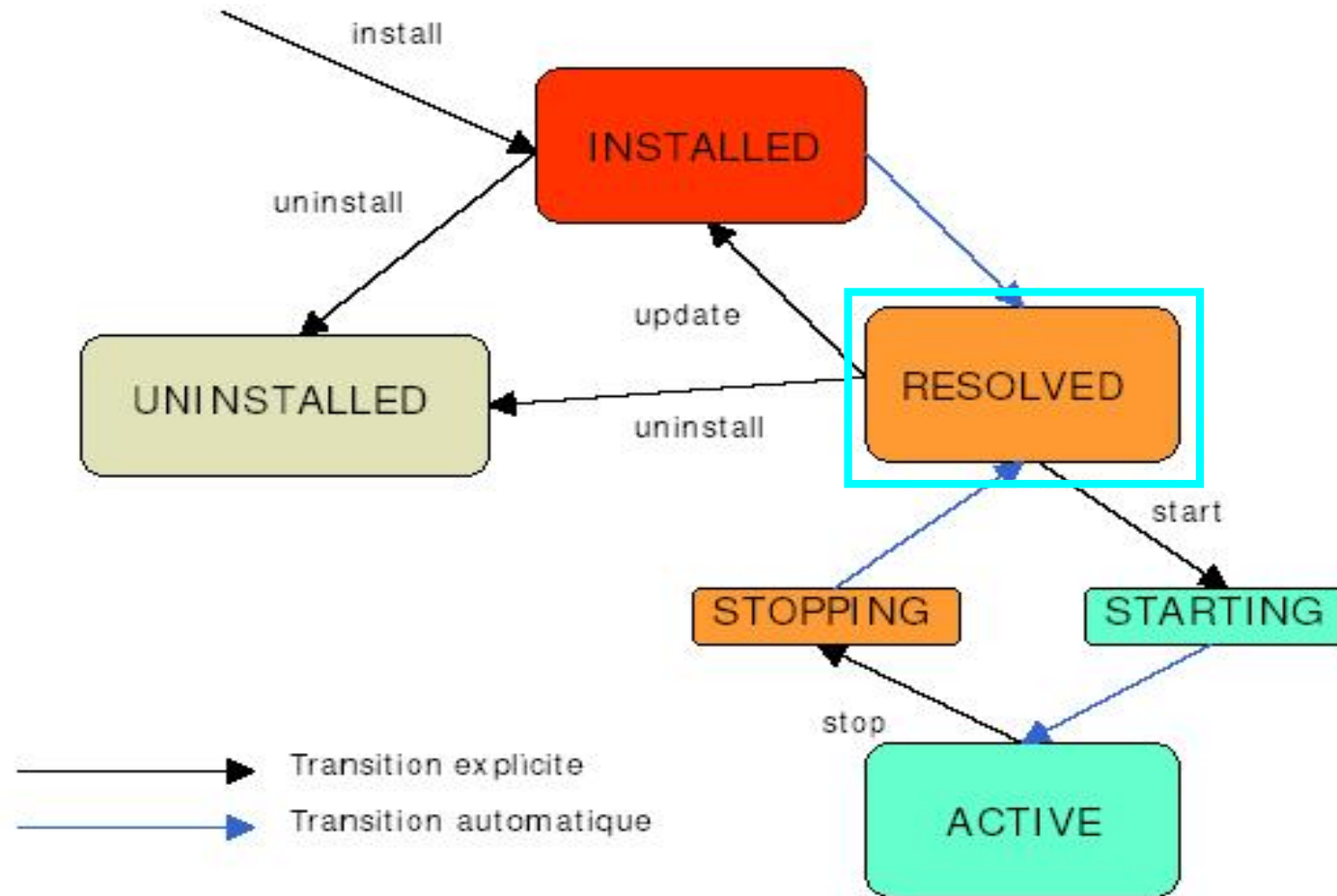
Cycle de vie d'un *bundle*

4. Cycle de vie d'un *bundle*



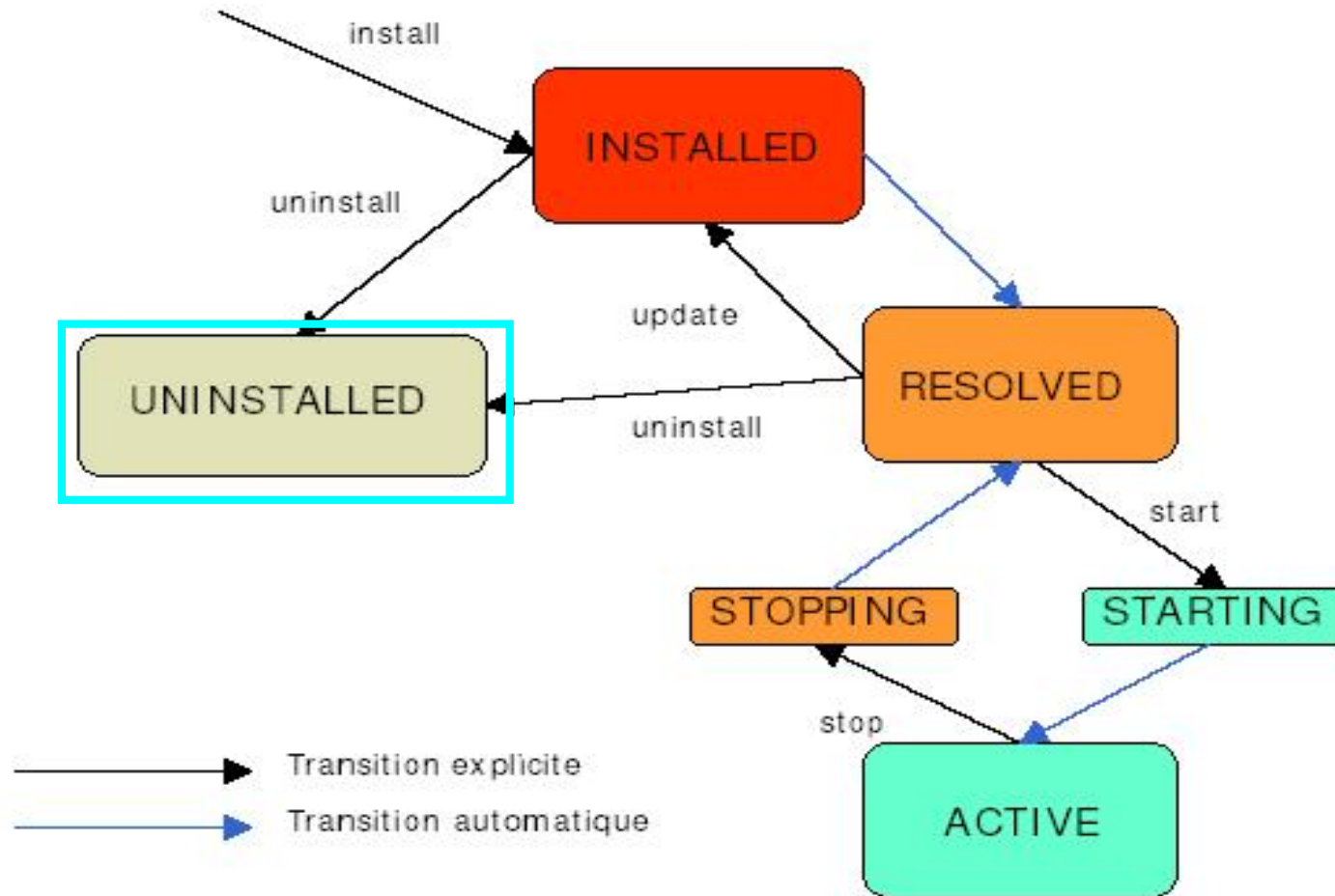
Cycle de vie d'un *bundle*

4. Cycle de vie d'un *bundle*



Cycle de vie d'un *bundle*

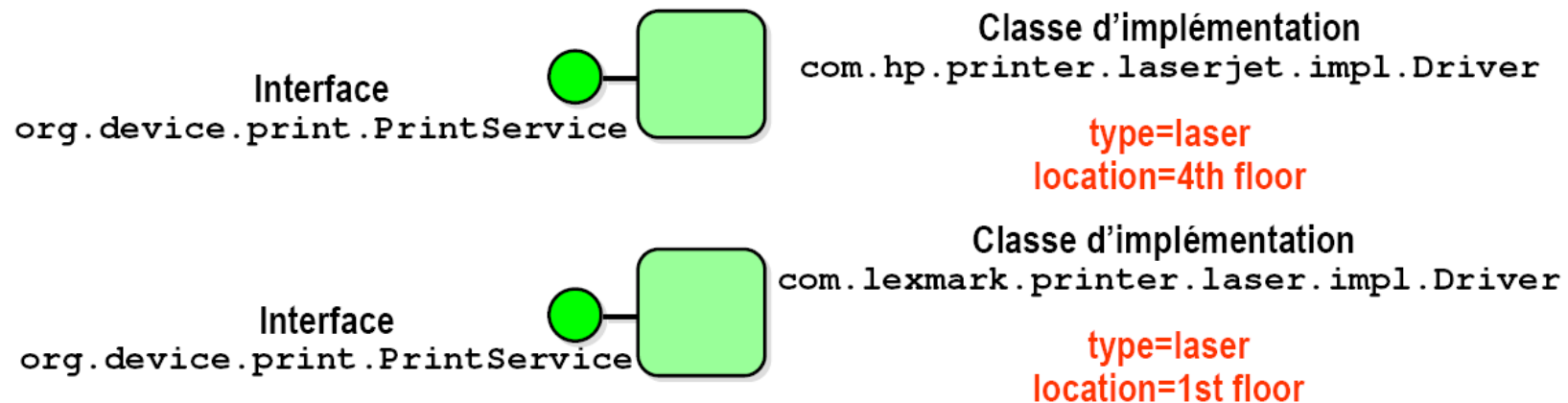
4. Cycle de vie d'un *bundle*



Cycle de vie d'un *bundle*

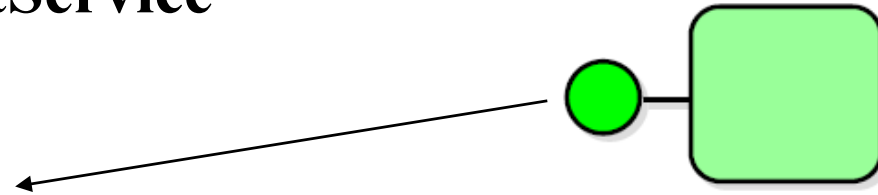
4. Services

- Une interface «published» et des implémentations
 - se trouvent dans des packages différents
 - implémentation normalement non publique.
 - multiples implémentations possibles
- « emballées » dans les bundles.
- Qualifié par des propriétés.



4. Exemple de service

Interface **org.device.print.PrintService**



```
package org.device.print;
```

```
import java.io.OutputStream;
```

```
import javax.print.PrintException;
```

```
public interface PrintService {
```

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

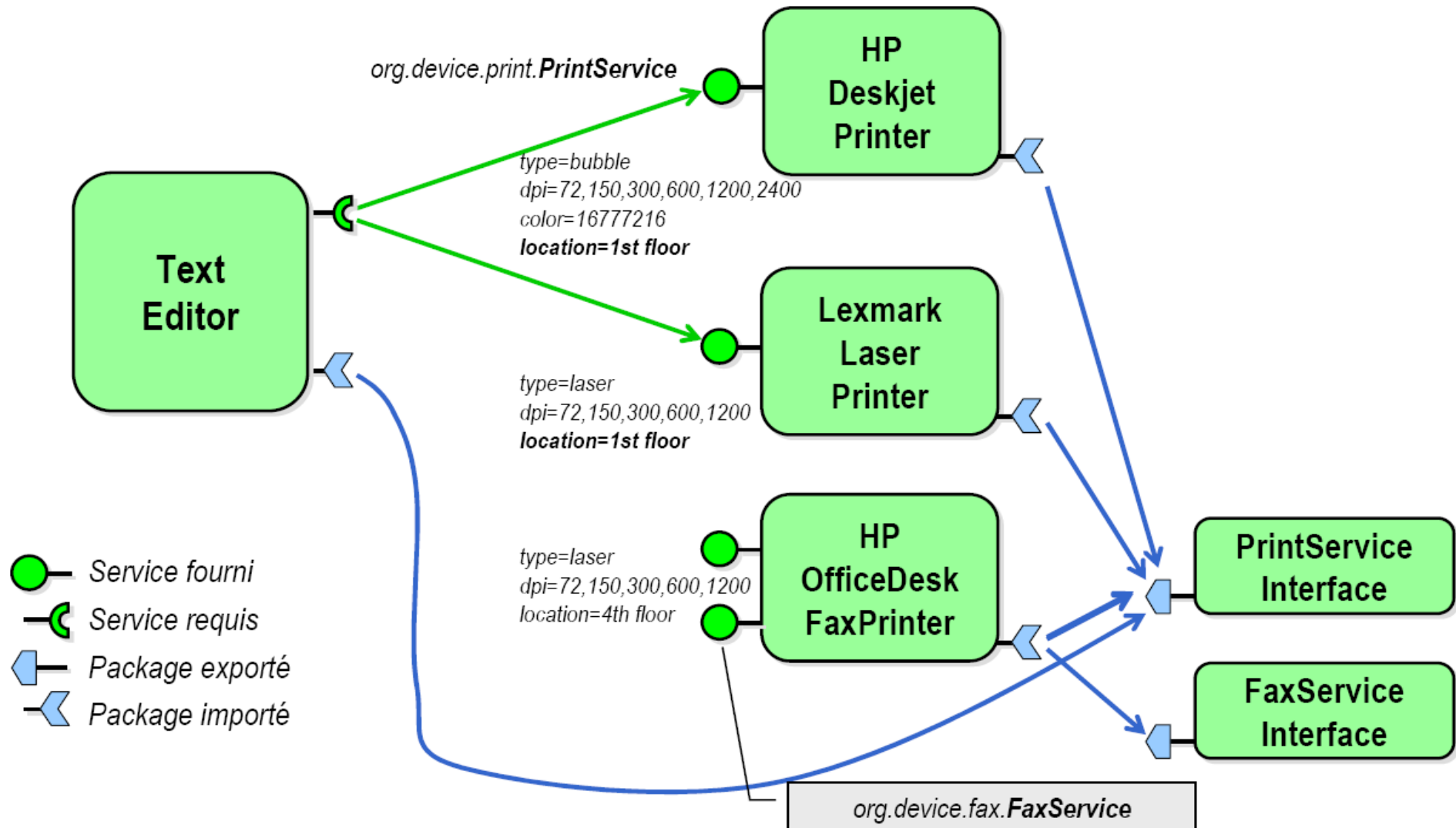
```
    public void kill(int jobnumber) throws PrintException;
```

```
    public Job[] list() throws PrintException;
```

```
    public Job status(int jobnumber) throws PrintException;
```

```
}
```

4. Exemple d'application



Manifest

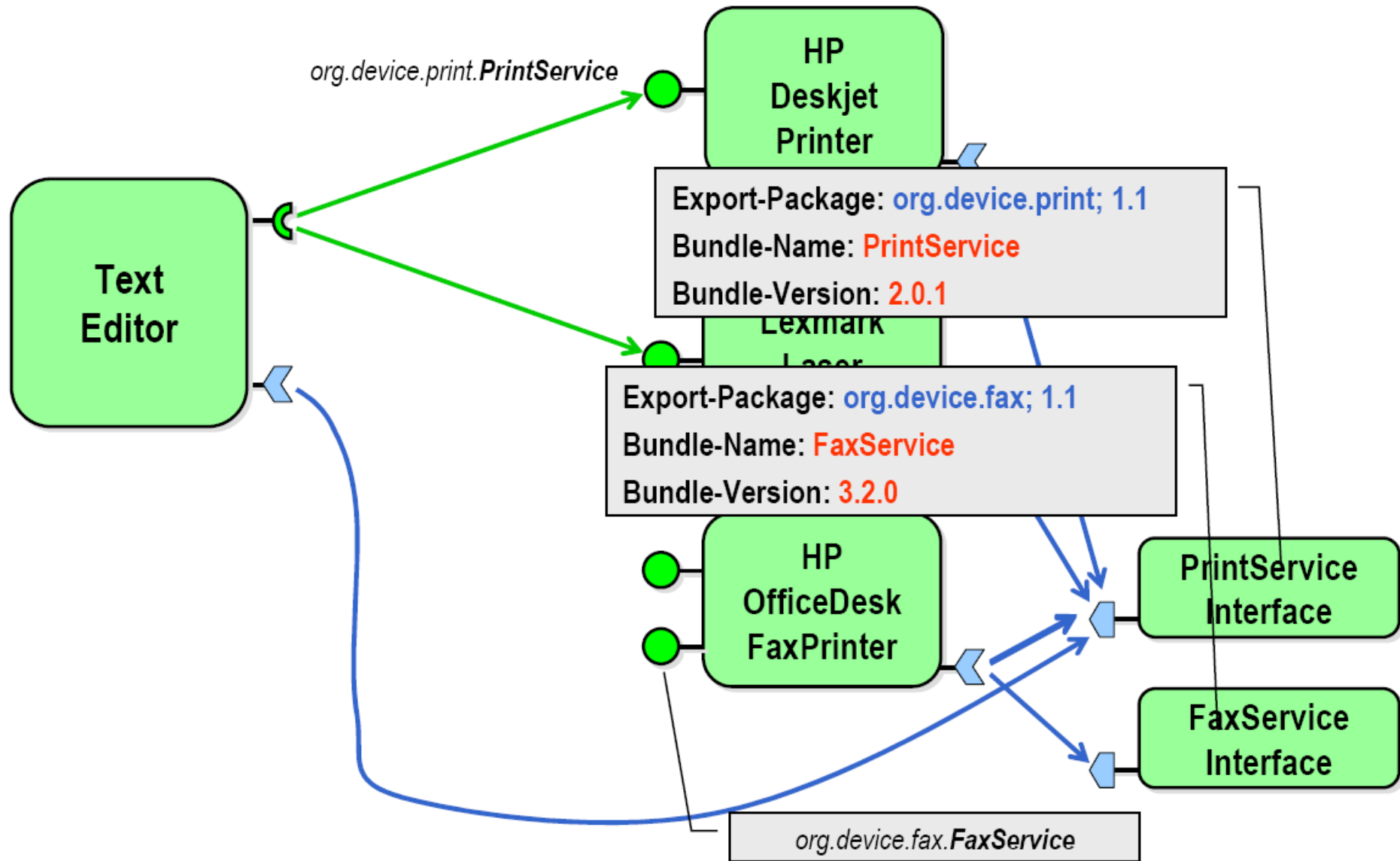
5. Fichier manifest (i)

Import-Package	Packages requis (avec/sans la version de spécification)
Export-Package	Packages fournis (avec/sans la version de spécification)
Import-Service	Services requis
Export-Service	Services fournis
<hr/>	
Bundle-Activator	Nom de la classe Activator
<hr/>	
Bundle-ClassPath	Emplacement des classes et ressources du bundle
Bundle-NativeCode	Bibliothèques natives à charger en fonction du processeur, du SE, ...
<hr/>	
Bundle-UpdateLocation	URL des mises à jour du bundle

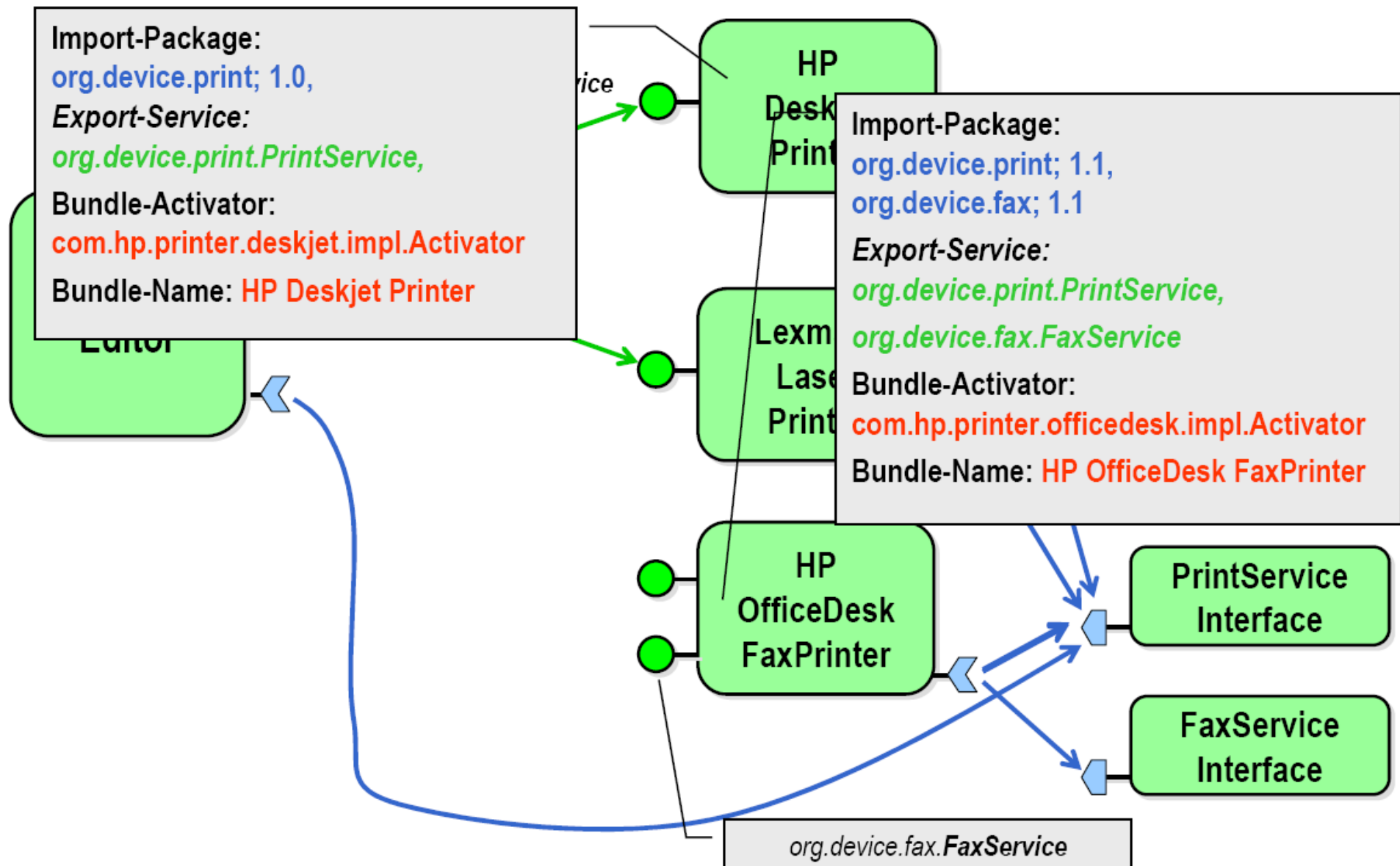
5. Fichier manifest (ii)

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-Copyright	Copyright du bundle
Bundle-Category	Catégorie du bundle
Bundle-RequiredExecution Environment	Liste d'environnement qui doivent être présents sur la plateforme (exemple : CDC-1.0/Foundation-1.0, OSGi/Minimum-1.0)
DynamicImport-Package	Liste de package qui pourront être importés en cours d'exécution (com.acme.plugin.*)

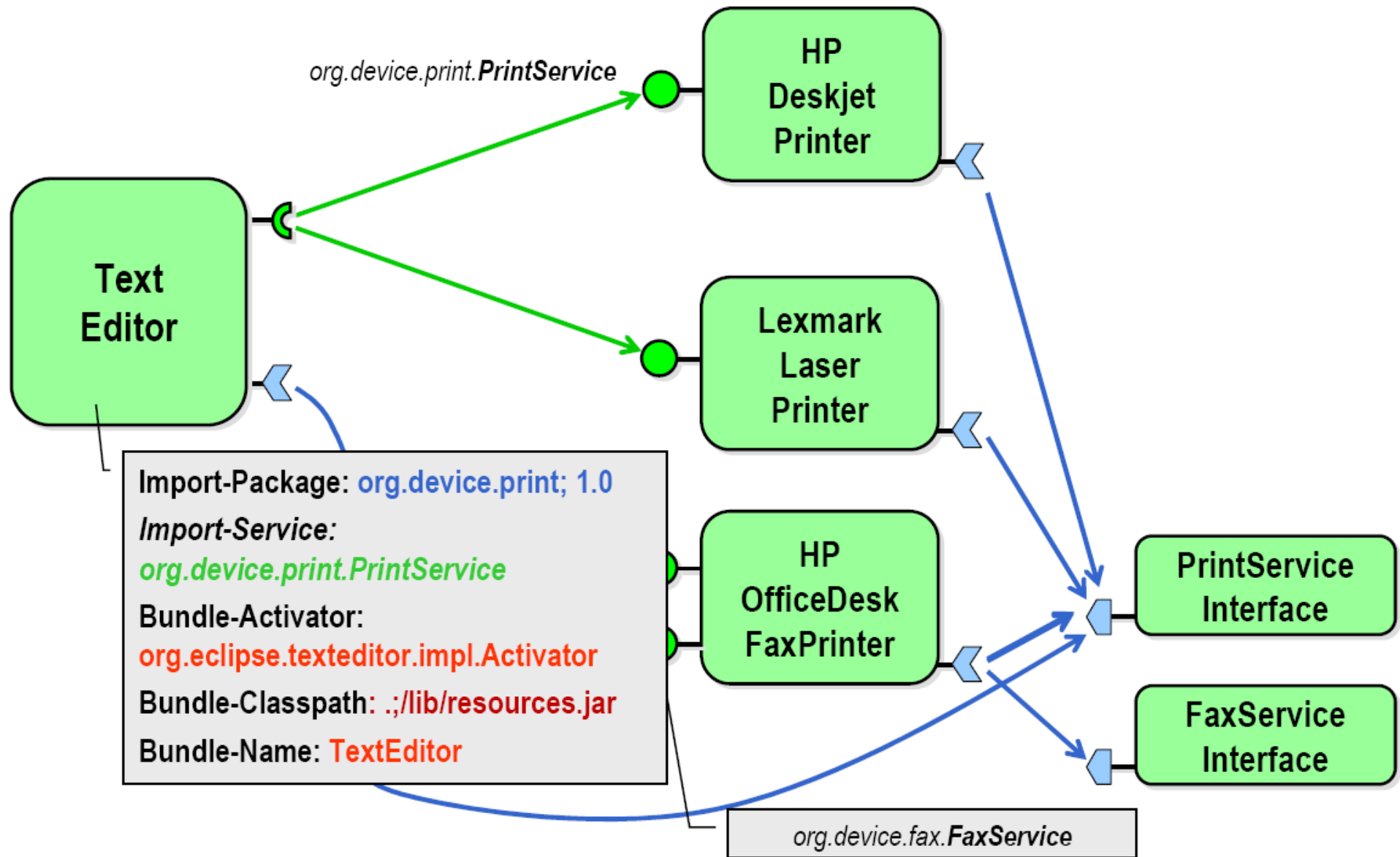
5. Exemple de manifest (i)



5. Exemple de manifest (ii)



5. Exemple de manifest (iii)



Activator et BundleContext

6. La classe *Activator* du bundle

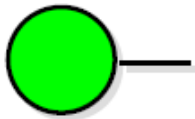
- Classe publique
 - Implémente les 2 méthodes `start()` et `stop()` de `BundleActivator`
 - qui reçoivent une référence sur un contexte.
- *start(BundleContext ctxt)*
 - recherche et obtient des services requis auprès du contexte et/ou positionne des listeners sur des événements
 - enregistre les services fournis auprès du contexte
- *stop(BundleContext ctxt)*
 - désenregistre les services fournis
 - relâche les services requis
 - Cependant le FW fait ces opérations si `stop()` est oublié !
- il peut ne pas y avoir d'Activator dans un bundle

6. *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 événements 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*

6. 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();
    }
}
```



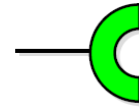
6. Recherche de services (*TextEditor*)

```

package org.eclipse.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]);
        }
        ...
    }
    ...
}

```

6. 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 `&`, `|`, `!`

6. Recherche (Courtage) 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)");
```

Bundles et services standards

7. Bundles et Services standards

- SystemBundle
- PackageAdmin
- PermissionAdmin
- UserAdmin
- ConfigurationAdmin
- Preference
- Metatype
- ServiceTracker
- LogService
- HttpService

7. *SystemBundle*

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

7. PackageAdmin

- The PackageAdmin class provides the following methods:
 - `getExportedPackage (String)`
 - Returns an ExportedPackage object that provides information about the requested package. This information can be used to make the decision to refresh the package.
 - `getExportedPackages (Bundle)`
 - Returns a list of ExportedPackage objects for each package that the given bundle exports.
 - `refreshPackages (Bundle [])`
 - The management bundle may call this method to refresh the exported packages of the specified bundles.

7. PermissionAdmin

- In the Framework, a bundle can have a single set of permissions associated with it. These permissions are used to verify that a bundle is authorized to execute privileged code. For example, a FilePermission defines what files can be used and in what way.
- The policy of providing the permissions to the bundle should be delegated to a management bundle. The Framework provides the Permission Admin service so that a management bundle can administrate bundle's permissions and provide defaults for all bundles.

7. org.service.http.HttpService

- 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)

7. LogService

- Permet de journaliser des traces ou de se mettre en l'écoute de ces traces

```

public class LogReaderActivator implements BundleActivator {
    LogReaderService logReaderService;
    LogListener loglistener;
    public void start(BundleContext cxt) {
        logReaderService= ...;
        loglistener=new LogReaderImpl();
        logReaderService.addLogListener(loglistener);
    }
    public void stop(BundleContext cxt) {
        logReaderService.removeLogListener(loglistener);
    }
}

class LogReaderImpl implements org.osgi.service.log.LogListener {
    public final void logged(LogEntry entry) {
        synchronized(this) {
            System.out.println("Level:"+entry.getLevel());
            if( entry.getBundle() != null) {
                System.out.println("bundle : "+
                    entry.getBundle().getBundleId()+" ");
            }
            System.out.println(entry.getMessage());
        }
    }
}

```

7. Device Manager

- Motivations
 - Faire apparaître les drivers des périphériques matériels comme des
- Services OSGi
 - 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

7. 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.
- Pour cela, il utilise les services :
 - DriverLocator
 - Driver

7. Event Admin Service

- 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
 - Événements 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 défectueux ou consommant trop de CPU.