On-demand Service Installation and Activation with OSGi

Vincent Lestideau, Didier Donsez

Université Joseph Fourier (Grenoble 1)

IMAG/LSR/ADELE

Vincent.Lestideau@imag.fr

Didier.Donsez@imag.fr

www.objectweb.org

ObjectWeb
Open Source Middleware

Agenda

- **Service-On-Demand Use-Cases**
- **GDF OSGi Overview**
  - Deployment with OSGi
  - Deployment Dependencies
  - GDF OSGi
- **Service-On-Demand Principles**
  - Installation and activation
- **Conclusion**
- **Project Roadmap**

Service-On-Demand Use-Cases

- Use-Case 1 : iTV STB
- Use-Case 2 : ressource-constrained gateway
- Use-Case 3: unused plugins

Use-Case 1 : OSGiTV

OSGi-based (DVB MHP) STB

- **Hundreds of iTV applications are available on the iTV operator repositories from EPG (Electronic Program Guide)**
- **But only a few can be installed/started on the SetTopBox (iTV terminal)**

Use-Case 1 : OSGiTV

OSGi-based (DVB MHP) STB

Mosaïc EPG XLet

Weather XLet

TCasino XLet

User Profile C.

Xlet Manager

Carousel Receiver

Local Cache

iTV Terminal

Jar

Extractor

Bundle

AIT

BIT

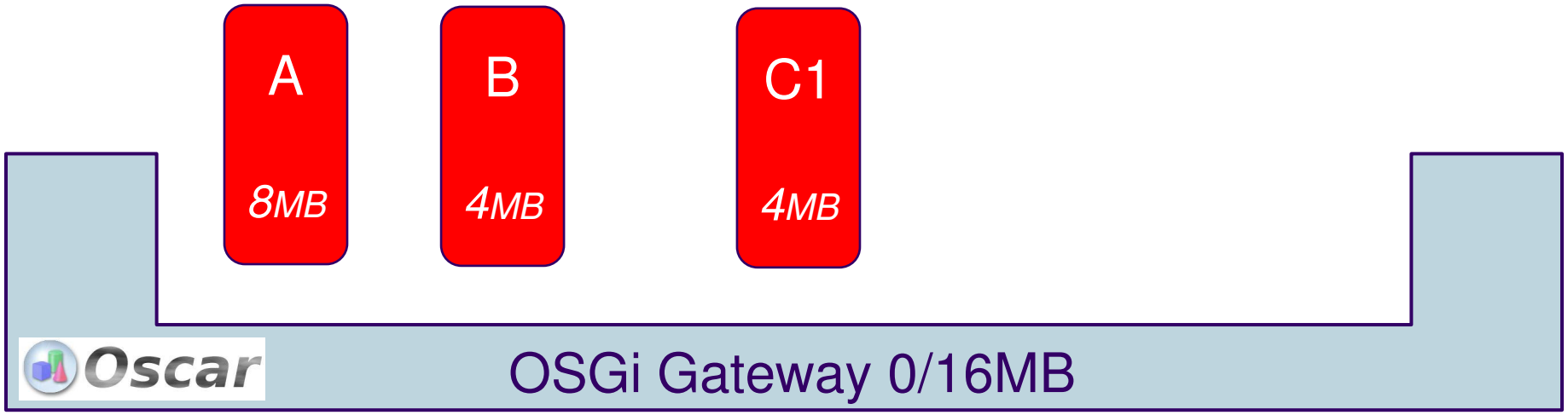
Carousel BroadCaster

AIT

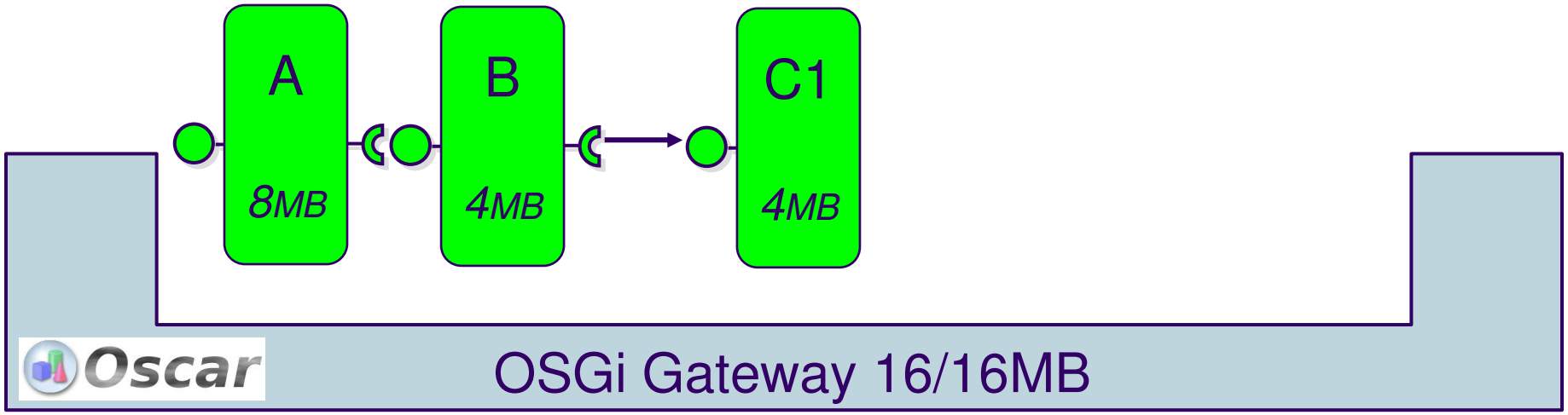
Use-Case 2 : Ressource-constrained gateway

- **Can install bundles (in the limit of the Secondary Memory)**
- **Can not start all at the same time (constrained by Primary Memory)**
  - Some can be « passivate » to limit memory comsuption

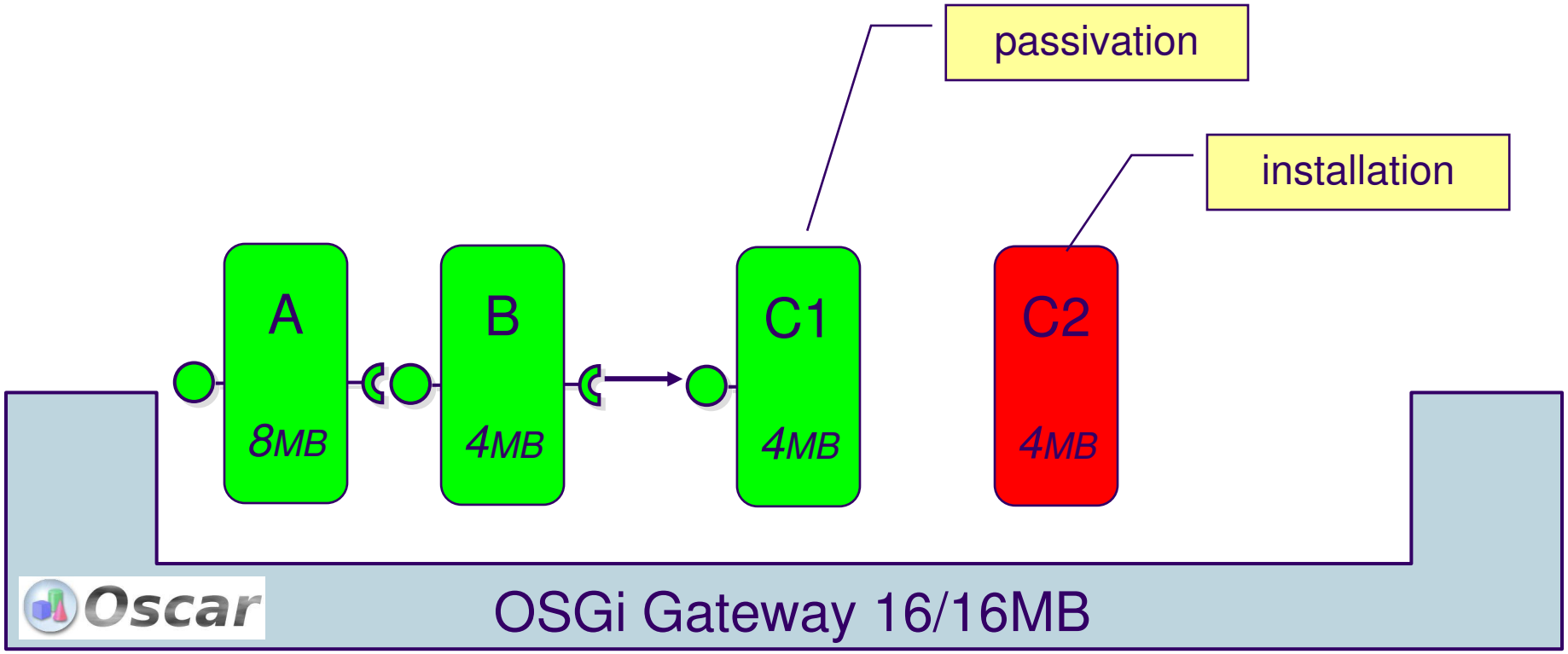
Use-Case 2 : Ressource-constrained gateway



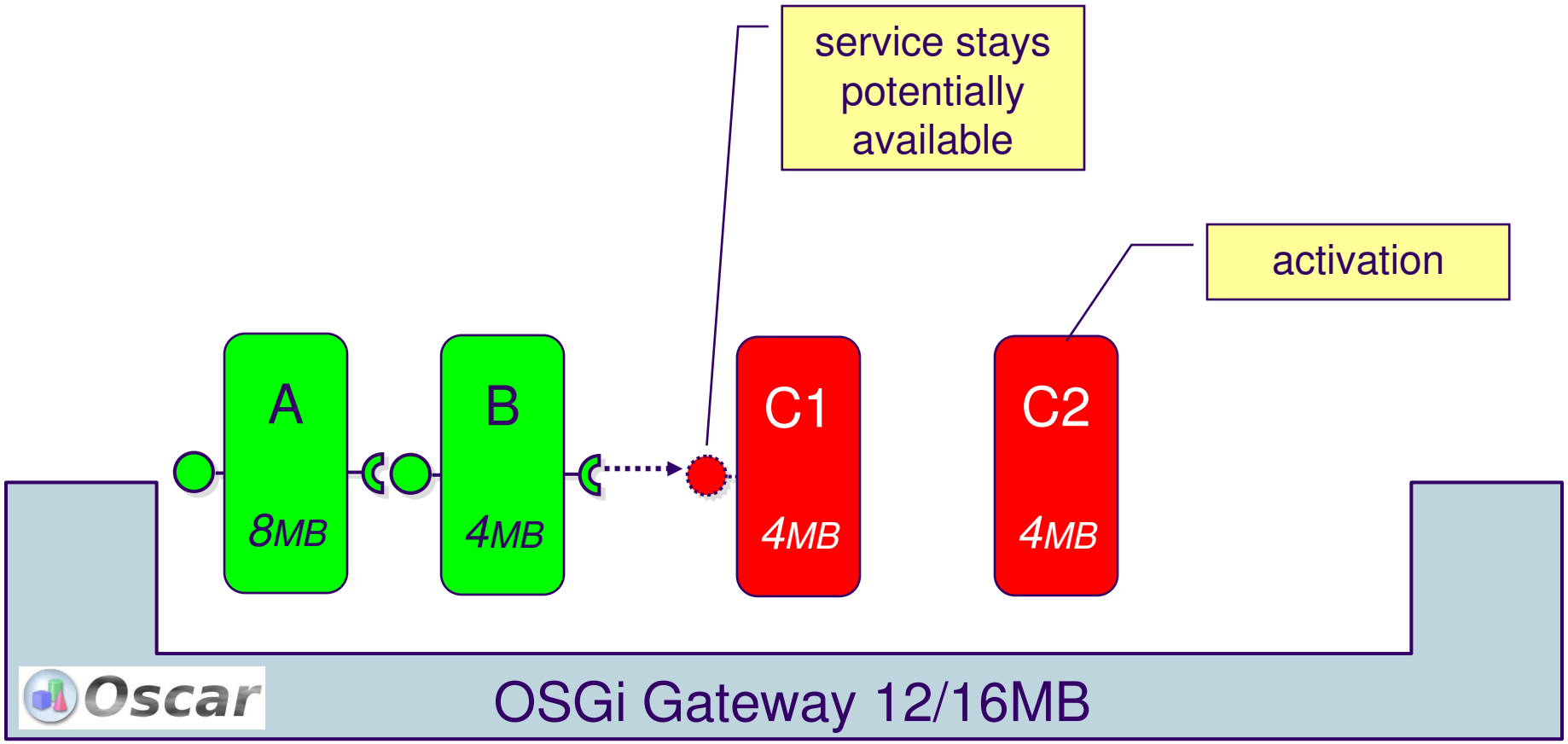
Use-Case 2 : Ressource-constrained gateway



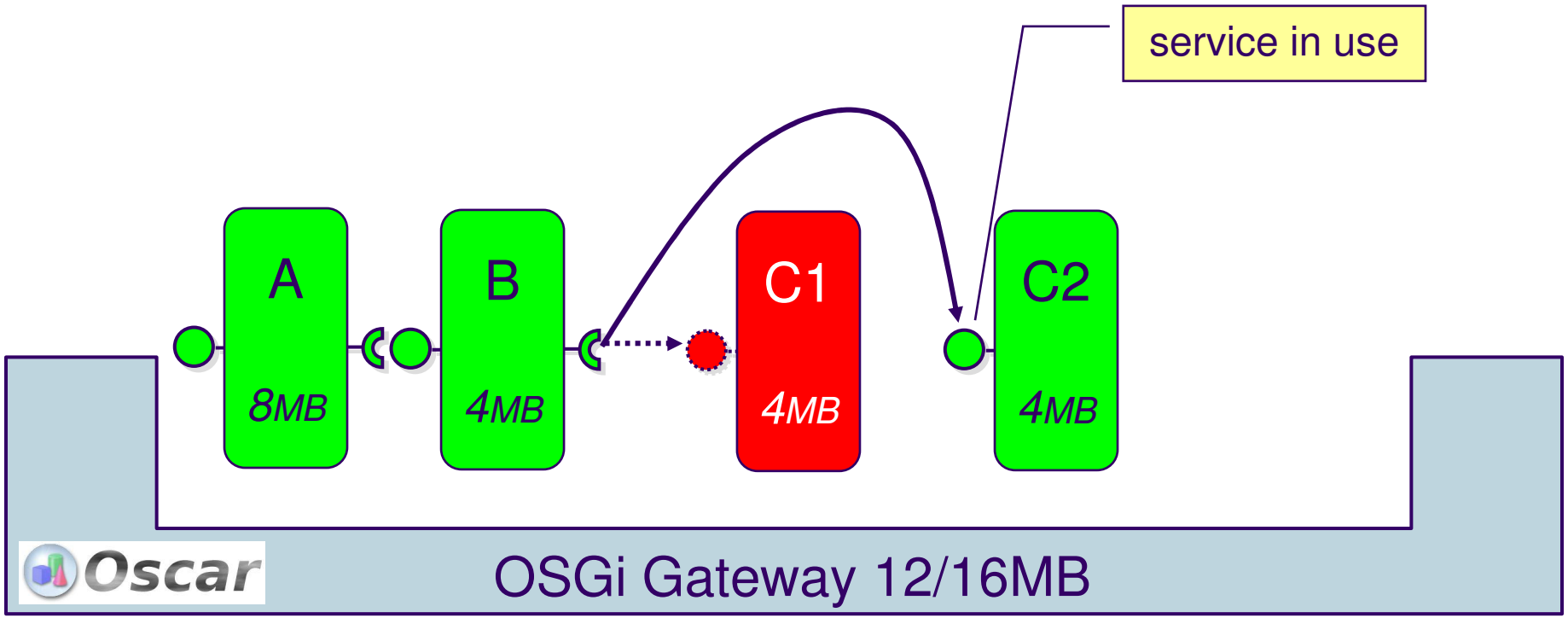
Use-Case 2 : Ressource-constrained gateway



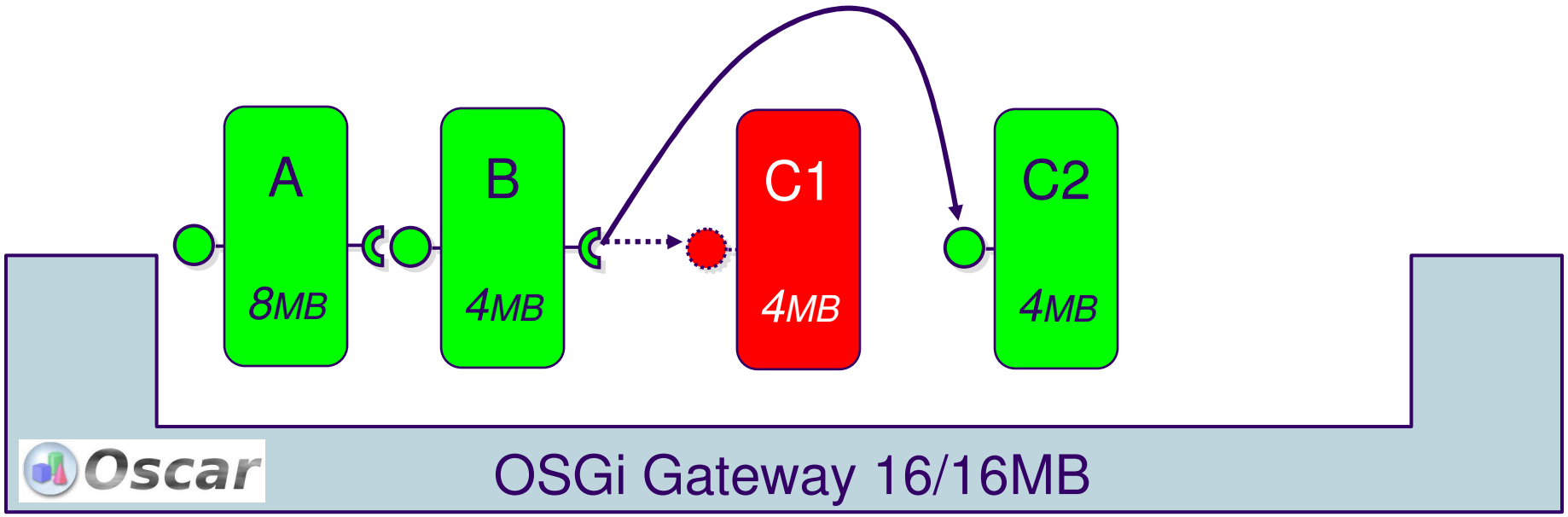
Use-Case 2 : Ressource-constrained gateway



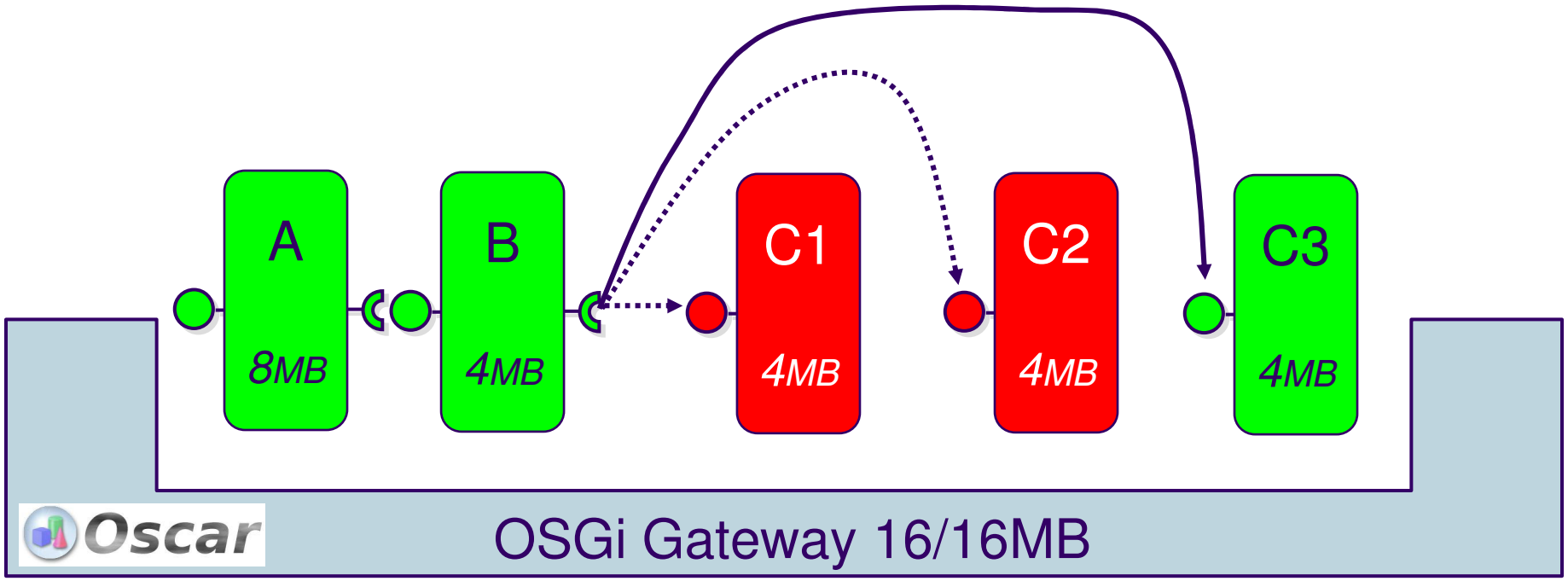
Use-Case 2 : Ressource-constrained gateway



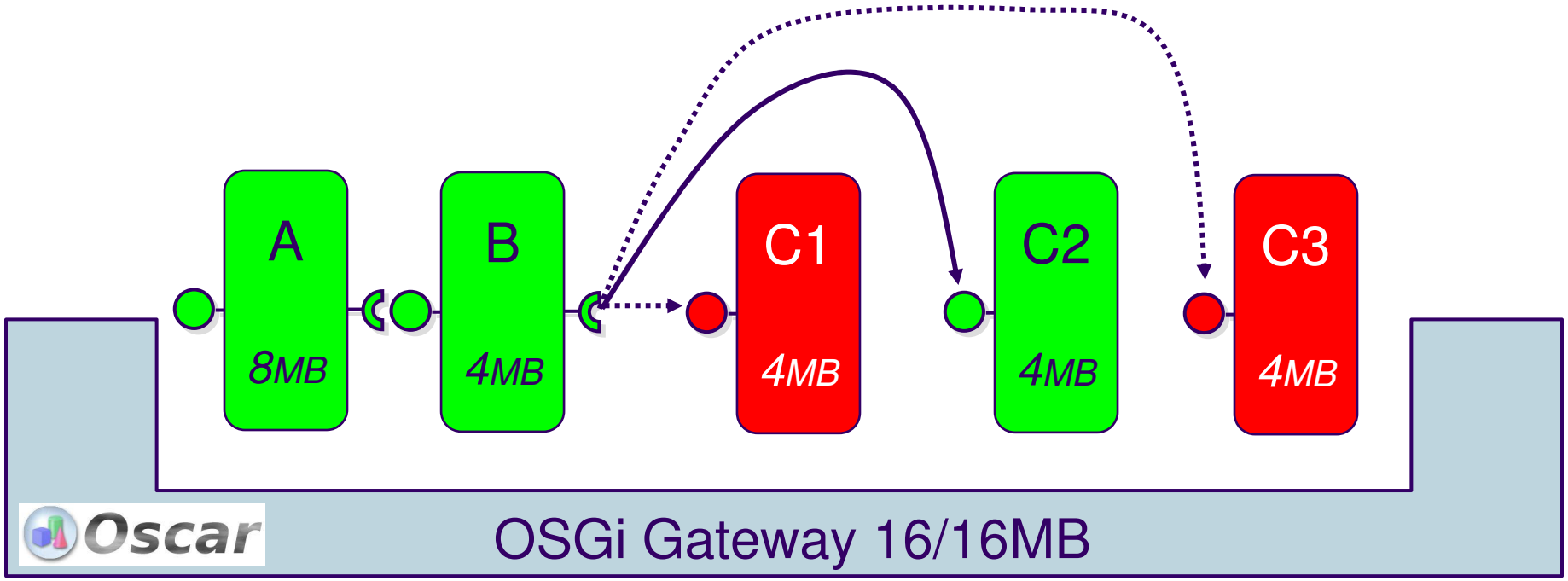
Use-Case 2 : Ressource-constrained gateway



Use-Case 2 : Ressource-constrained gateway



Use-Case 2 : Ressource-constrained gateway



Use-Case 3 : incremental plugIn activation

- **Plugin are installed and are started only if they are really used !**
  - Eclipse, JMF, ANT …

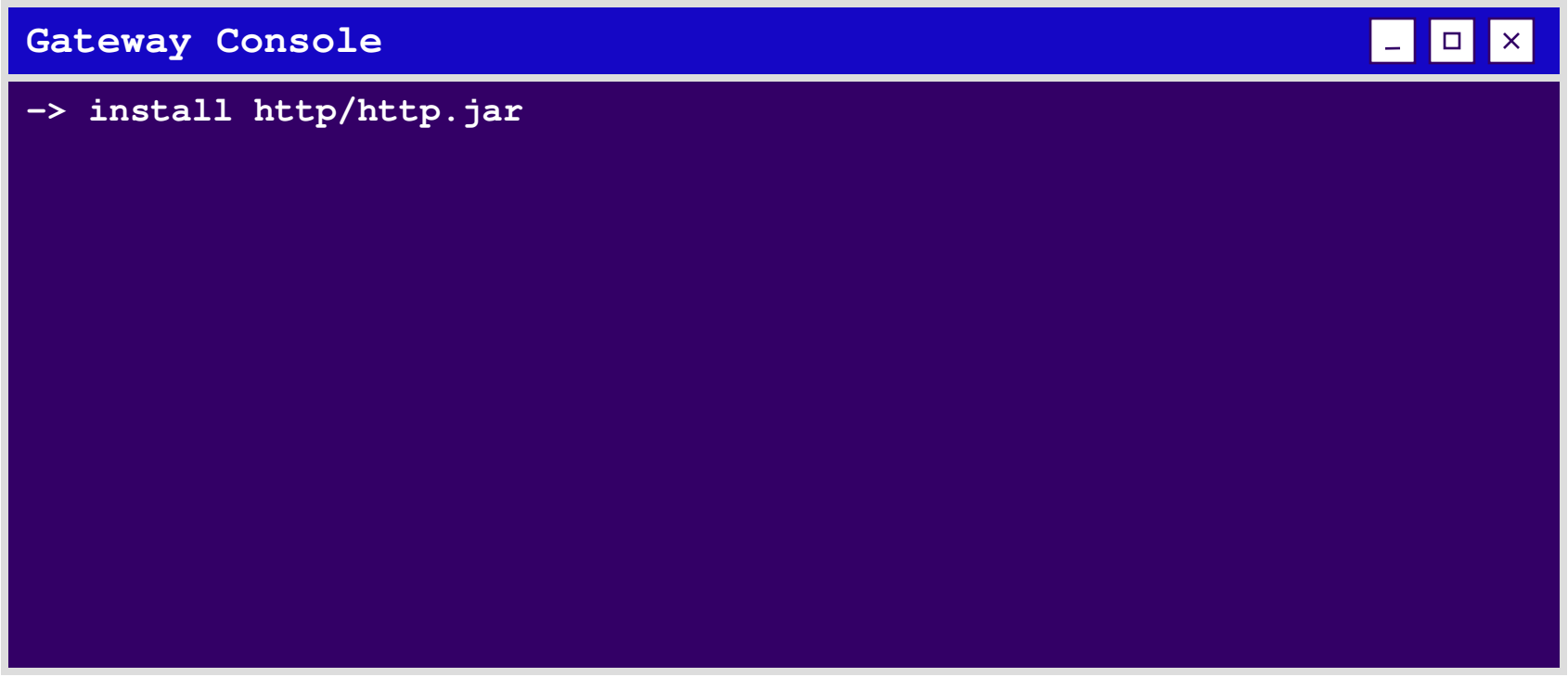
Service-On-Demand Use-Cases

- **Use-Case 1 : iTV STB**
- **Use-Case 2 : ressource-constraint gateway**
- **Use-Case 3: unused plugins**

- **The solution**
  - On-Demand Service Installation
  - On-Demand Service Activation

Deployment of an application on the OSGi gateway

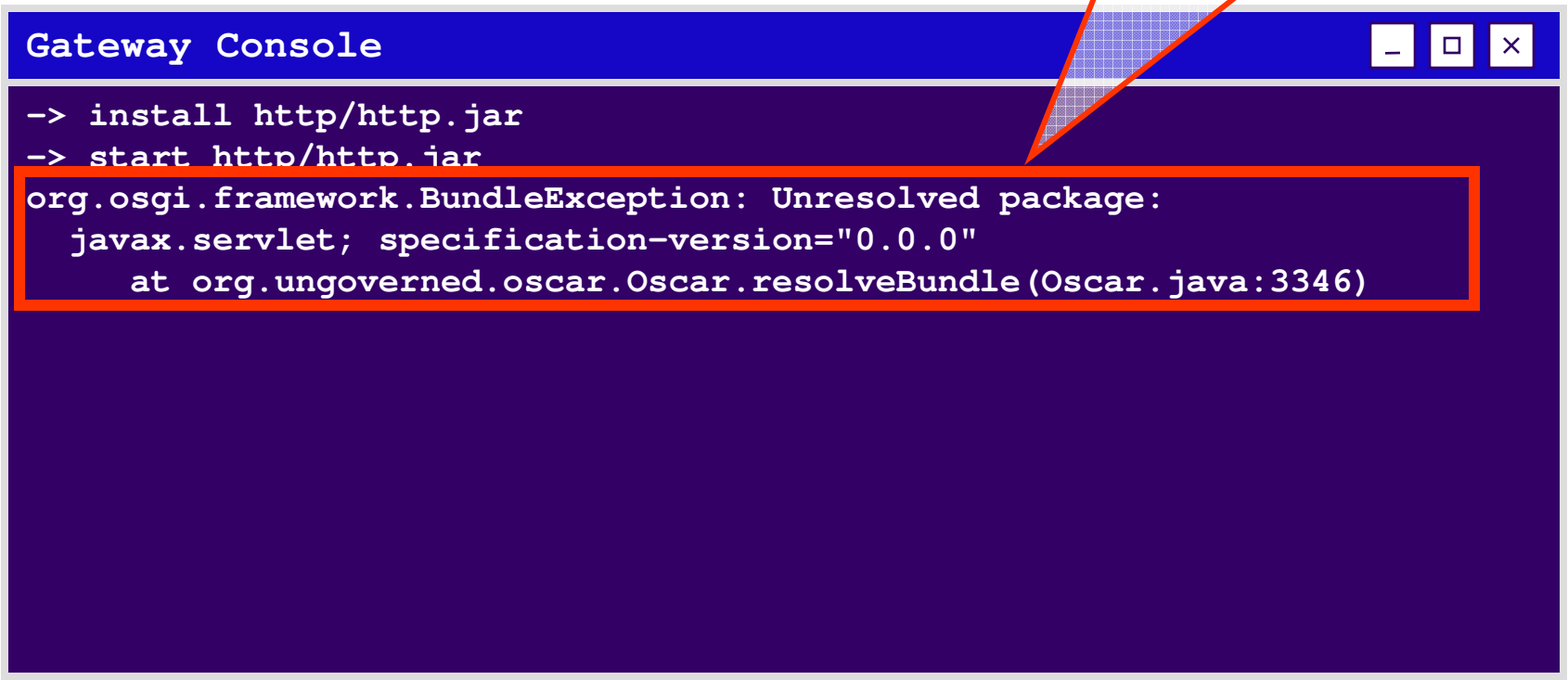
- Deploy an application
  - → Deploy a set of bundles
- From a console (System.in, telnet, Http Admin)



Deployment of an application on the OSGi gateway

- Deploy an application
  - → Deploy a set of bundles
- From a console (System.in, telnet, Http Admin)

Problem :
Start requires packages before !
Which bundle provides them ?



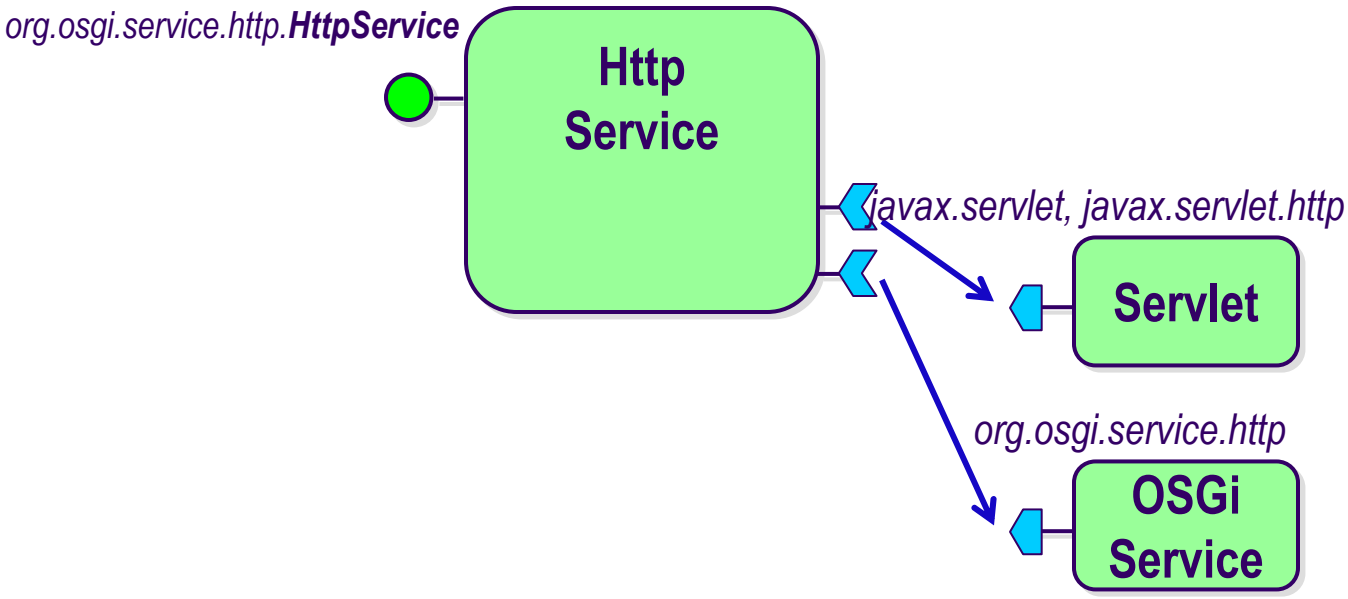
Deployment of an application on the OSGi gateway

- Deploy an application
  - → Deploy a set of bundles
- From a console (System.in, telnet, Http Admin)

Gateway Console

```
-> install http/http.jar
-> start http/http.jar
org.osgi.framework.BundleException: Unresolved package:
  javax.servlet; specification-version="0.0.0"
     at org.ungoverned.oscar.Oscar.resolveBundle(Oscar.java:3346)
-> install servlet/servlet.jar
-> start servlet/servlet.jar
-> start http/http.jar
11:03:56.208 EVENT  Starting Jetty/4.2.x
11:03:56.359 EVENT  Started SocketListener on 0.0.0.0:8080
11:03:56.369 EVENT  Started org.mortbay.http.HttpServer@b6548
->
```

Deploying Http Service



OBR *OSCAR Bundle Repository*

First, an index of bundles metadata

```
<bundle>
 <bundle-name>Servlet</bundle-name>
 <bundle-updatelocation> http://bundles.mycomp.com/servlet/servlet.jar</bundle-updatelocation>
 <export-package package="javax.servlet" specification-version="2.3"/>
 <export-package package="javax.servlet.http" specification-version="2.3"/>
 <export-package package="javax.servlet.jsp" specification-version="1.2"/>
</bundle>

<bundle>
 <bundle-name>HTTP Service</bundle-name>
 <bundle-updatelocation>http://bundles.mycomp.com/http/http.jar</bundle-updatelocation>
 <import-package package="org.osgi.framework"/>
 <import-package package="javax.servlet" specification-version="2.3"/>
 <import-package package="javax.servlet.http" specification-version="2.3"/>
 <import-package package="org.osgi.service.http" specification-version="1.1.0"/>
</bundle>
```

define an Import-Export Package Dependency

OBR *OSCAR Bundle Repository (ii)*

Second

- a Planner
  - compute the list of bundles to install/start (or update) according to Import-Export Package dependencies graph
- and a PlanManager
  - Run the plan computed by the planner

OSCAR Shell command

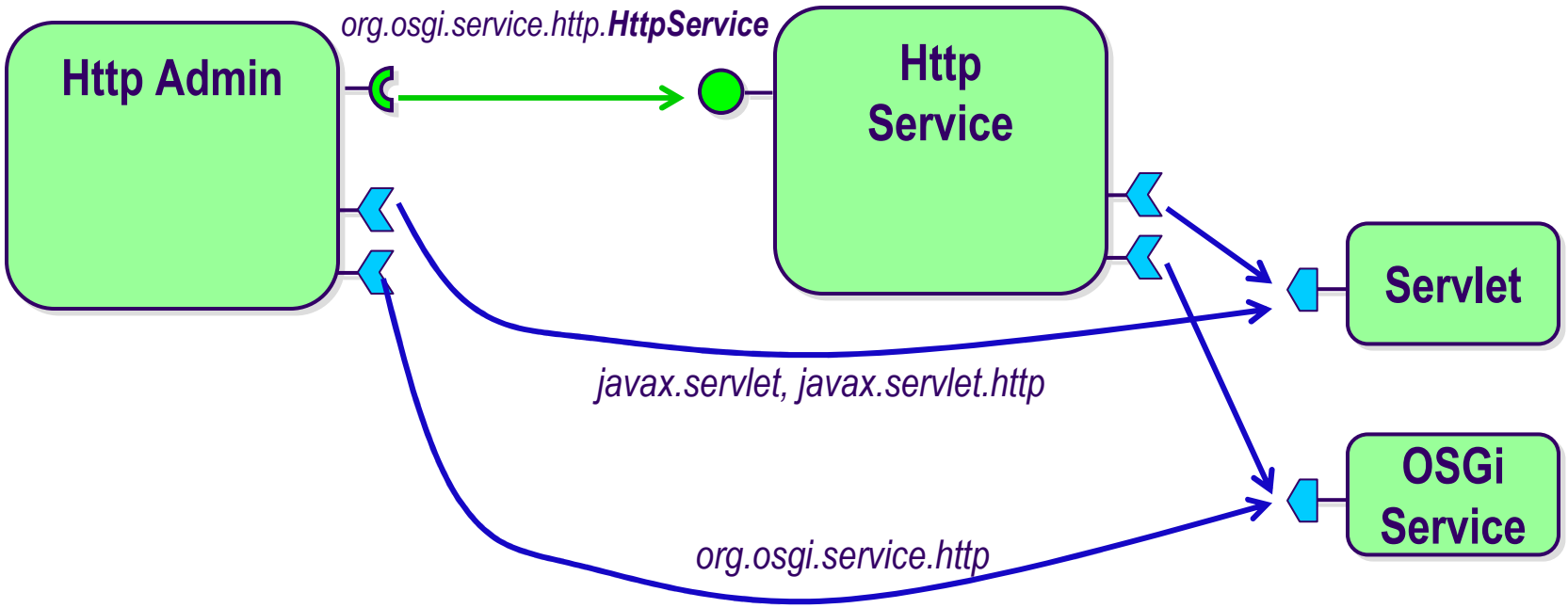
- to use the BundleRepository Service
- List, info, deploy, install, start, update…

OBR *(iii)* *Example*

Gateway Console

```
-> obr start "Http Service"
Installing dependency: Servlet
Installing dependency: OSGi Service
Installing: HTTP Service
11:03:56.208 EVENT  Starting Jetty/4.2.x
11:03:56.359 EVENT  Started SocketListener on 0.0.0.0:8080
11:03:56.369 EVENT  Started org.mortbay.http.HttpServer@b6548
->
```

Deploying Http Admin



OBR *(iv)* *Example*

```
Gateway Console
-> obr start "Http Admin"
Installing dependency: Servlet
Installing dependency: OSGi Service
Installing: HTTP Admin
-> ps
START LEVEL 1
   ID   State         Level  Name
...
[  4] [Active     ] [    1] Servlet (1.0.0)
[  5] [Resolved   ] [    1] OSGi Service (1.3.0)
[  6] [Active     ] [    1] HTTP Admin (1.0.0)
->
```

➔ What is wrong ?

- Everything seems OK !
- But « Http Service » is not deployed and it's required by « Http Admin »
- **Service Dependencies are not in the « graph »**

What is missing ?
Import-Export Service Dependency

```
<bundle>
  <bundle-name>HTTP Service</bundle-name>
  <bundle-updatelocation>http://bundles.mycomp.com/http/http.jar</bundle-updatelocation>
  <import-package package="org.osgi.framework"/>
  <import-package package="javax.servlet" specification-version="2.3"/>
  <import-package package="javax.servlet.http" specification-version="2.3"/>
  <import-package package="org.osgi.service.http" specification-version="1.1.0"/>
  <export-service name="org.osgi.service.http.HttpService"/>
</bundle>
<bundle>
  <bundle-name>HTTP Admin</bundle-name>
  <bundle-updatelocation>http://bundles.mycomp.com/httpadmin/httpadmin.jar</bundle-updatelocation>
  <import-package package="org.osgi.framework"/>
  <import-package package="javax.servlet" specification-version="2.3"/>
  <import-package package="javax.servlet.http" specification-version="2.3"/>
  <import-package package="org.osgi.service.http" specification-version="1.1.0"/>
  <import-service name="org.osgi.service.http.HttpService"/>
</bundle>
```

define an Import-Export Service Dependency

Other kinds of dependencies

- **Bundle dependencies**
  - An another bundle
  - A service
  - A package
- **Environnement dependencies**
  - Software (Linux RPM, Windows External DLL or Applications…)
  - Hardware
- **Pattern dependencies**
  - Producer-Consumer dependency (WireAdmin)
  - Export-Import Service dependency
    - Deploy a *Command* requires a *Shell*

GDF *Generic Deployment FrameWork*

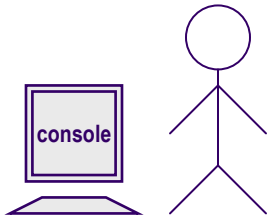
Multi-typed dependencies planner

- Input
  - Set of dependencies to satisfy
    - Cardinality : min and stopthreshold
    - Contingence: mandatory or optional
  - Repository index (OBR metadata (standard + custom))
  - Installed Bundles on the gateway
- Output
  - computed Deployment plan

Two Forms

- On-gateway planner and manager
- Server-side planner
  - Deployment plan is transformed in an ANT build file (to run)

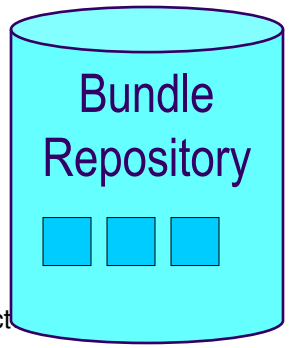
GDF
server-side planner

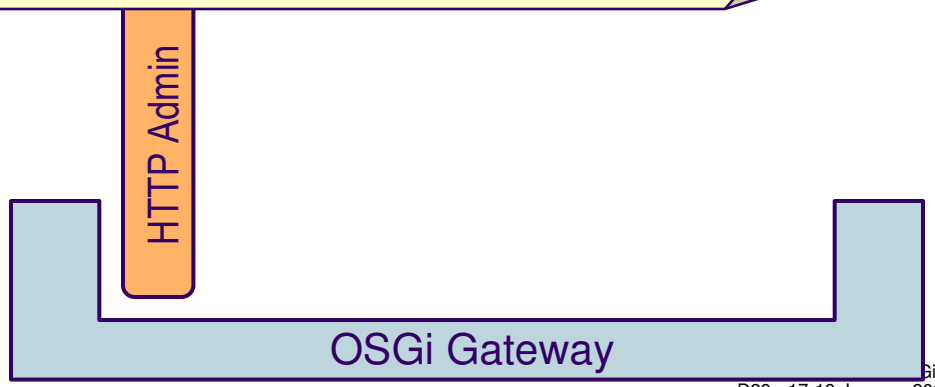


Deployer

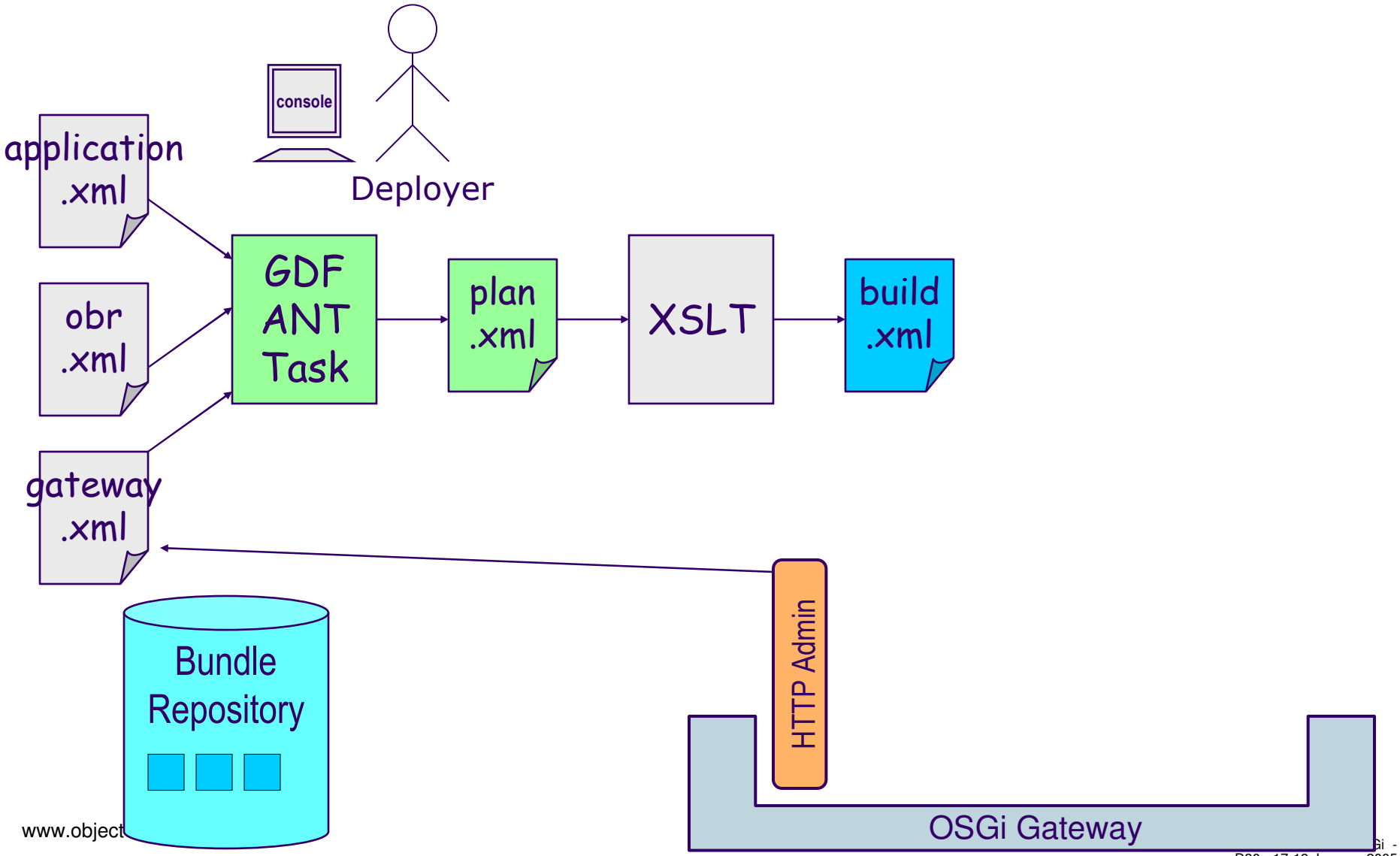
application.xml

```
<!-- eTrack application -->
<dependencies>
  <unitdependency id="areadetection.jar"/>
  <importservicedependency
          service="org.osgi.service.wireadmin.Producer"
          filter="(wireadmin.consumer.flavors=*org.osgi.util.position.Position)"
          minimalcardinality="2" stopat="4"/>
  <importservicedependency
          service="org.osgi.service.io.ConnectionService"
          filter="(|(io.scheme=*sms)(io.scheme=*mms))"
          minimalcardinality="1" stopat="1"/>
<dependencies>
```

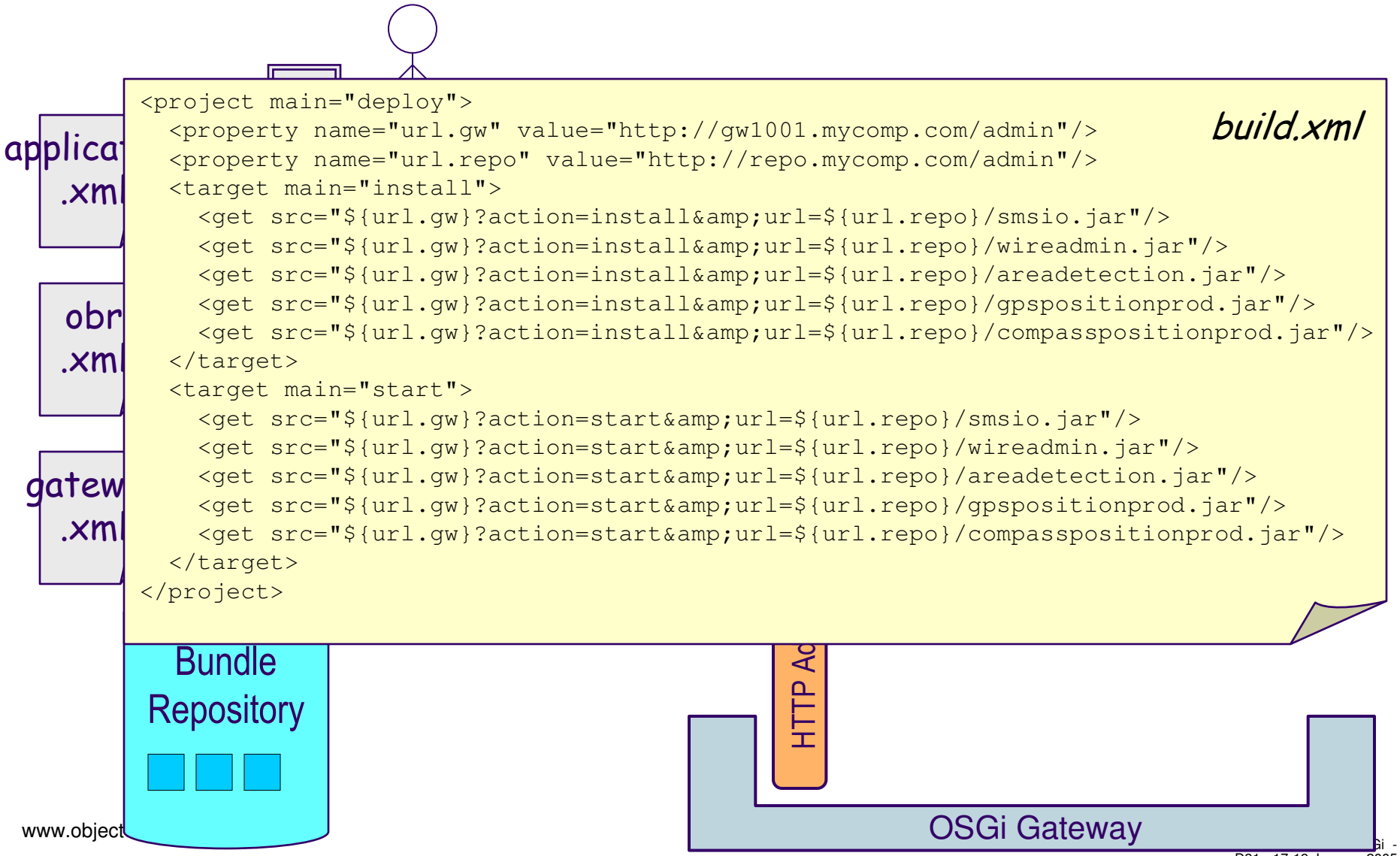




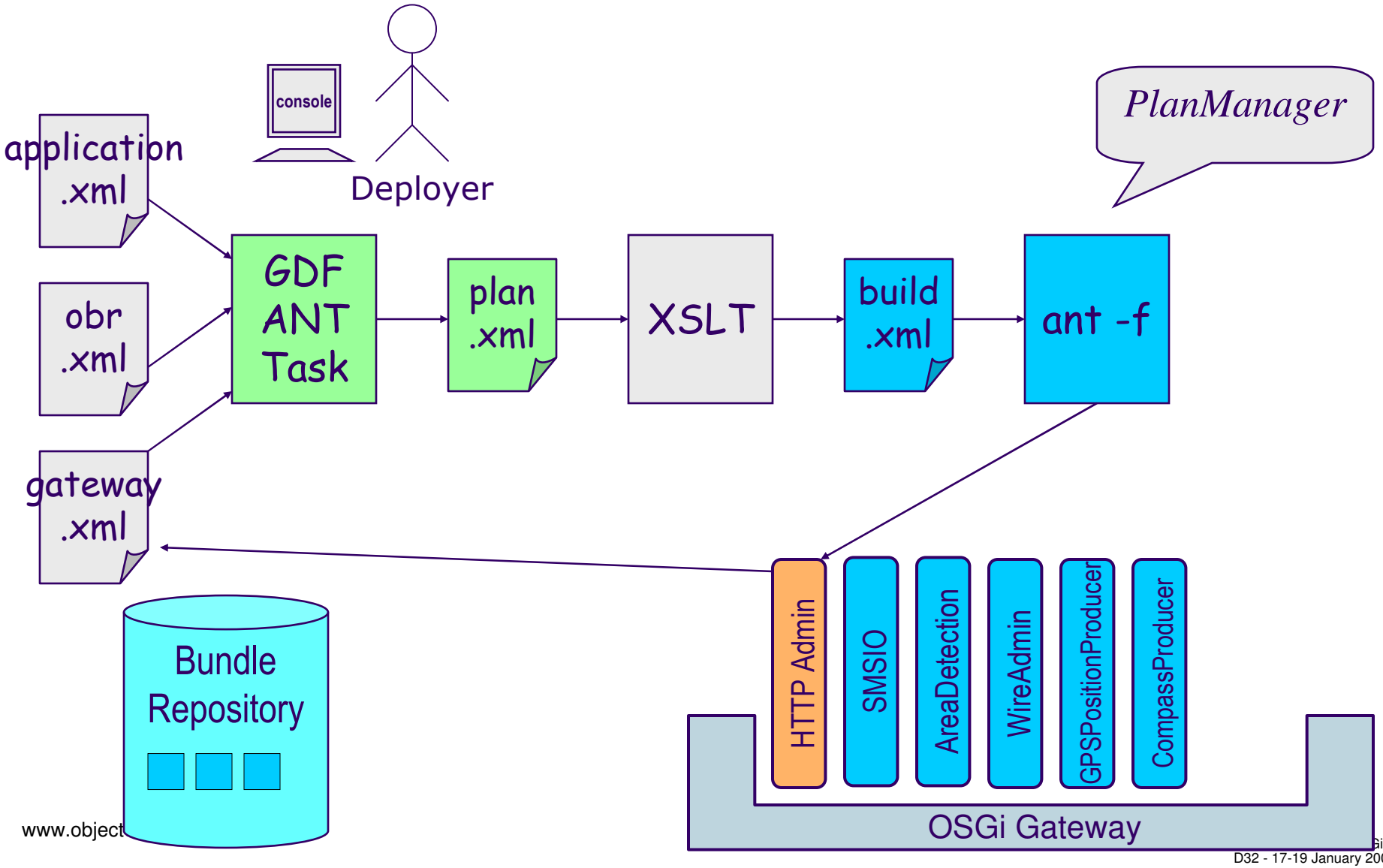
GDF
server-side planner



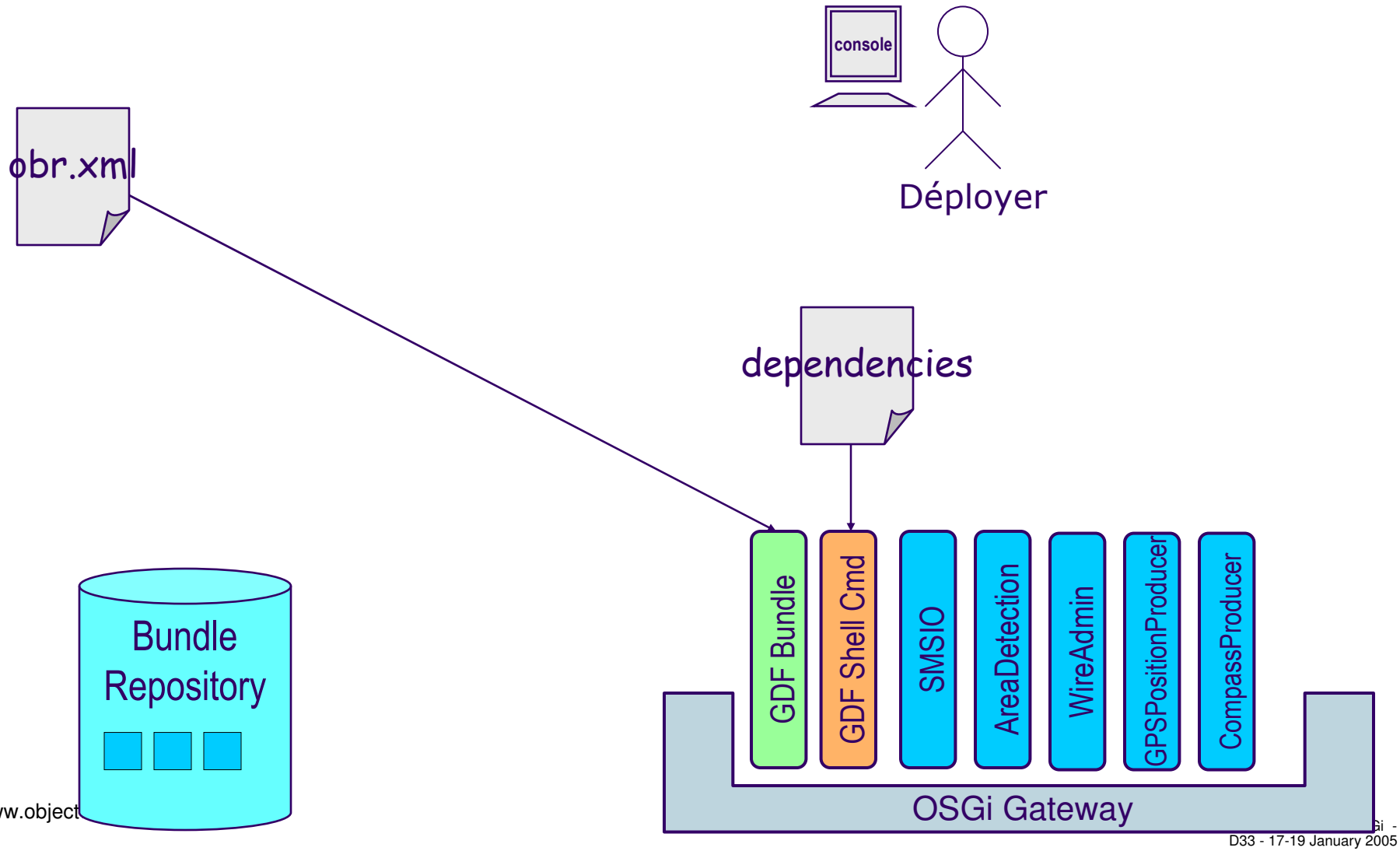
GDF
server-side planner



GDF
server-side planner



GDF
on-gateway planner (as OBR)



GDF
Concrete Dependencies

Common

- **UnitDependency**
- **LdapFilterDependency**

Constructor

- **ConjunctionDependency**
- **DisjunctionDependency**
- **NotDependency**

OSGi

- **VersionedUnitDependency**
- **ImportEportPackageDep (OBR)**
  - Eg: the unit requires the package javax.servlet; version=2.3
- **ImportExportServiceDep**
  - Eg: A telnet deamon requires a Shell
- **QualifiedImportExportServiceDep**
  - Eg Service Binder
- **ExportImportServiceDep**
  - Eg: A command requires a Shell
- **ConsumerProducerWireAdminDep**
  - Eg: Requires consumers consuming temperatures
- **ProducerConsumerWireAdminDep**
  - Eg: Requires producers producing GPS positions
- **NativeDependency**
- **…**

Is everything OK ?

No 

1. **Developers must describe ServiceDependencies**
   → what to do with legacy systems ?

2. **Bundles (provided services) are installed-started**
   even if they are not immediatly used

On-Demand Service

On-Demand Service Installation

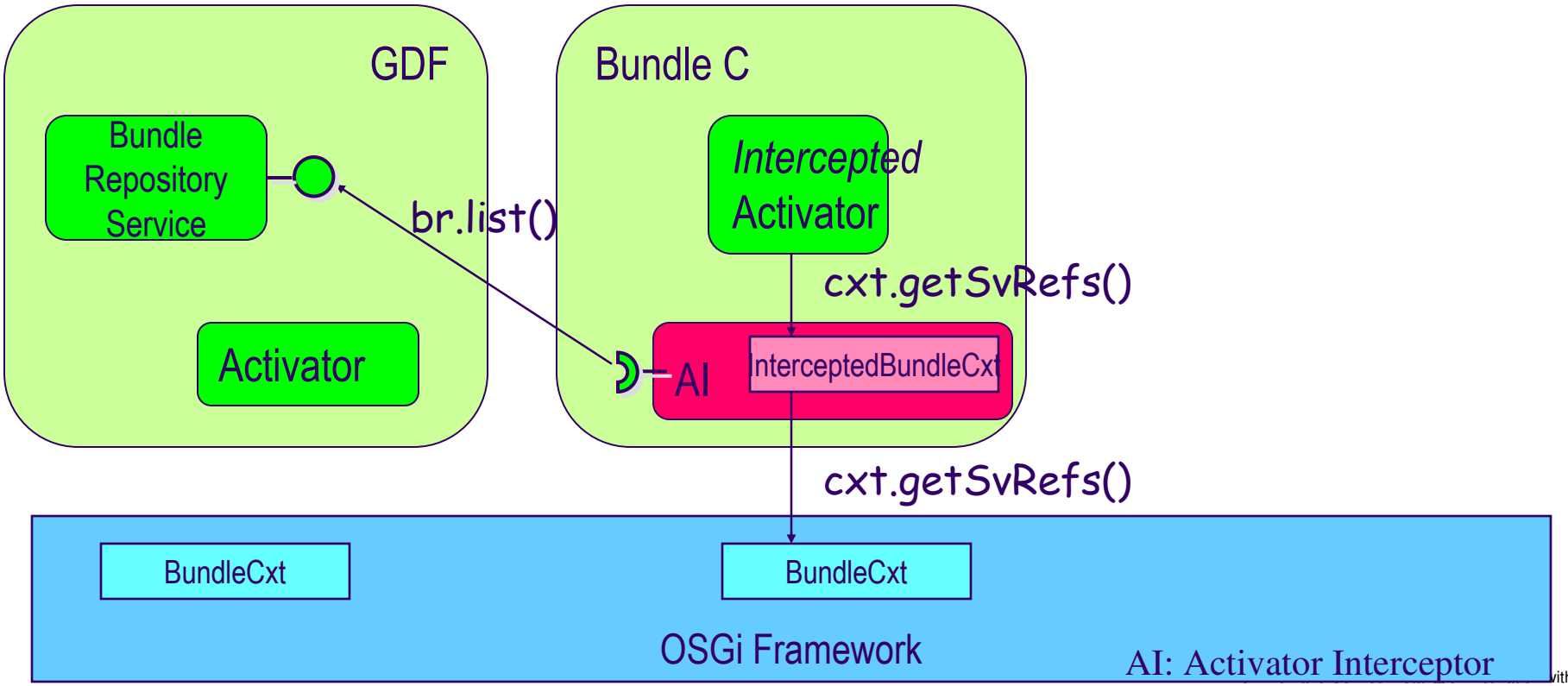
- When service dependencies are unknown until runtime

On-Demand Service Activation

- When services are passivated
  - Memory consumption
- After installation

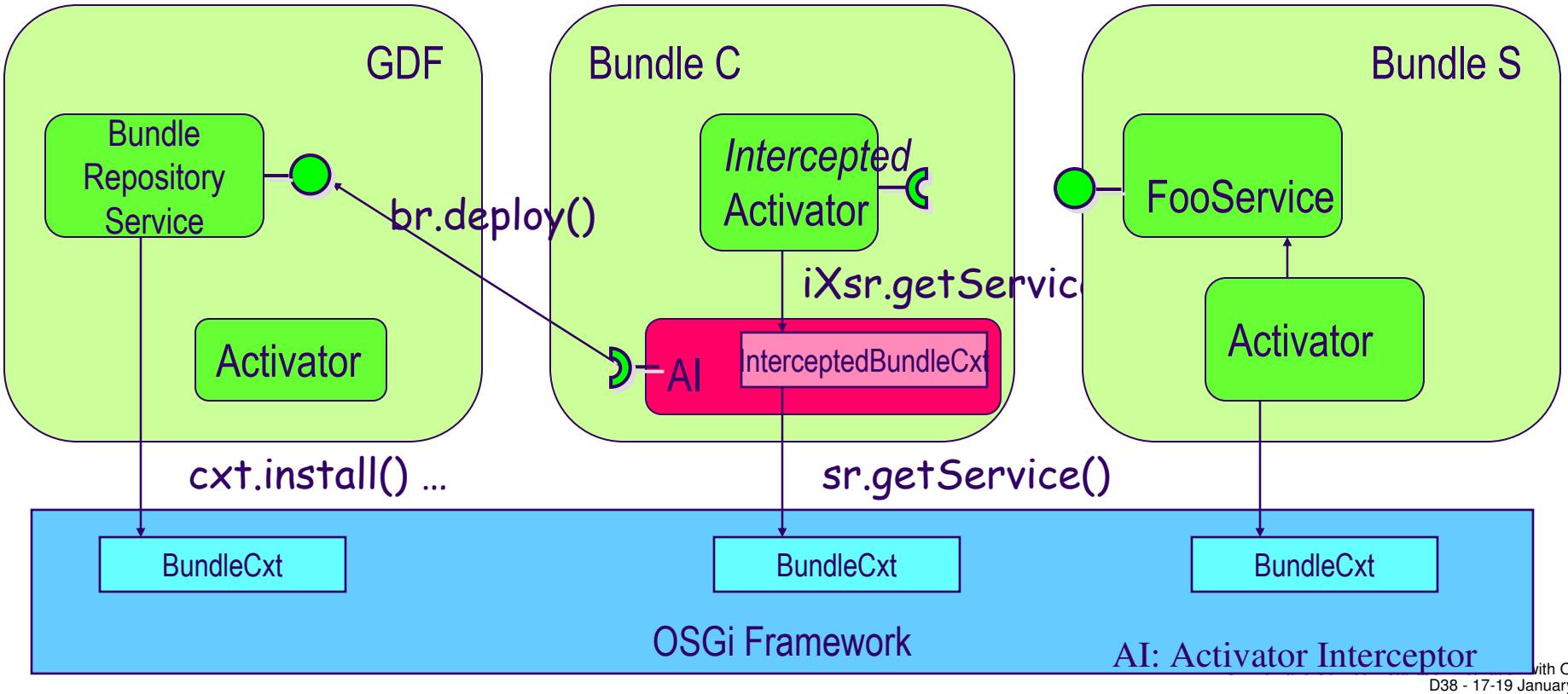
On-Demand Service Installation/Activation

- **Calls to BundleContext are intercepted**
  - ServiceReferences are simulated from OBR/GDF metadata



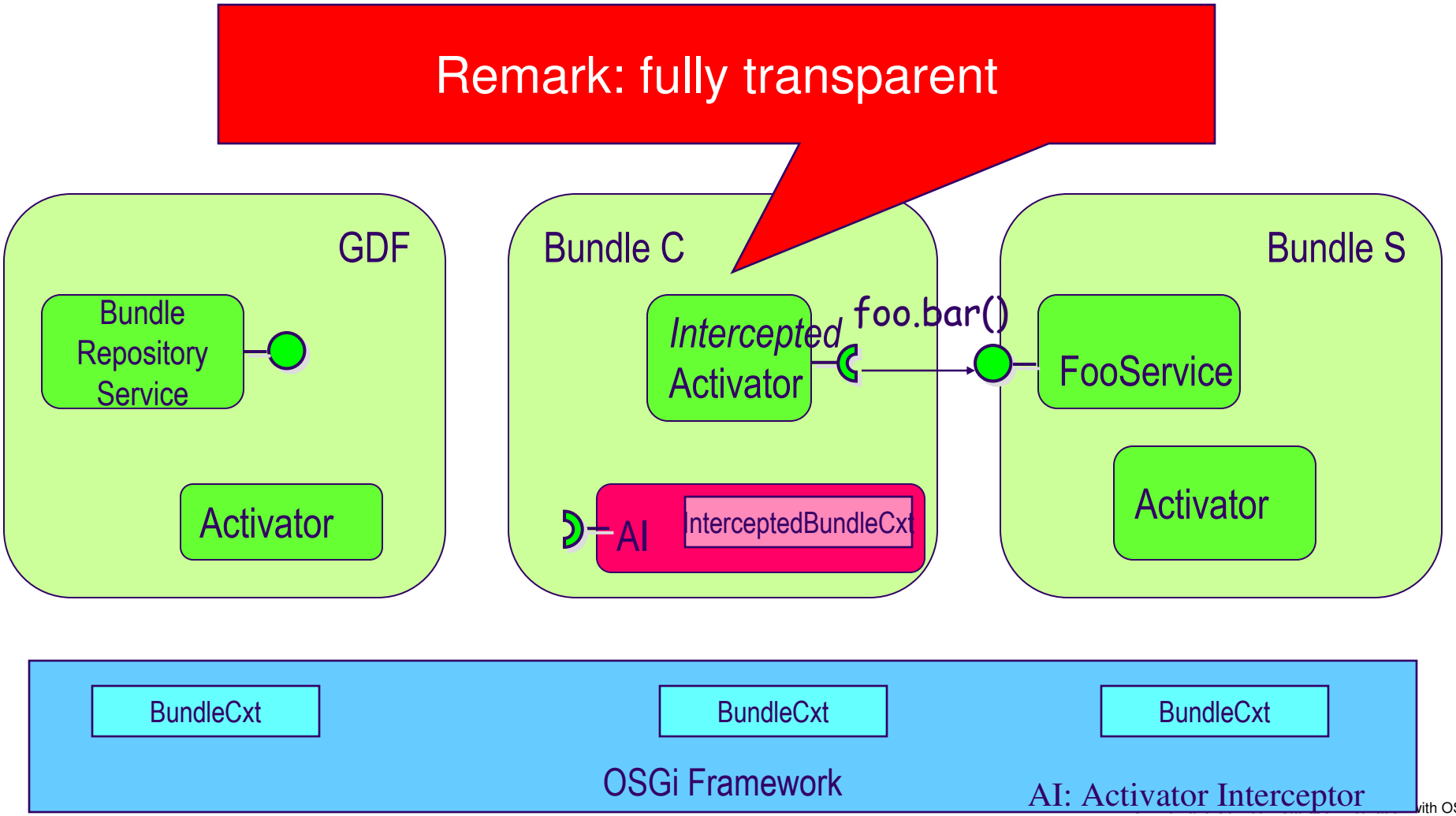
On-Demand Service Installation/Activation

- **Calls to BundleContext are intercepted**
  - ServiceReferences are simulated from OBR/GDF metadata
  - ServiceReference.getService() trigs Installation/Activation



On-Demand Service Installation/Activation

Service invocation

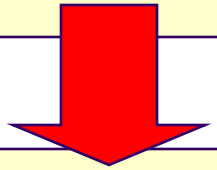


Transform your bundle to be « On-Demand »

➔ **Just change 3 lines in the Manifest and add an « empty » class inheriting AI**

```
Bundle-Name: Bundle C
Bundle-Activator: org.my.client.Activator
Import-Package: org.my.foo
```

.mf



```
Bundle-Name: Bundle C
Bundle-Activator: serviceondemand.util.ActivatorInterceptor
X-Bundle-Activator:org.my.client.Activator
Import-Package: org.my.foo,
    org.ungoverned.osgi.service.bundlerepository,
    serviceondemand.util
```

.mf

Conclusion

- **Multi-typed Dependencies planner to deploy OSGi « applications »**
- **On-Demand Service Installation**
  - Legacy systems
- **On-Demand Service Activation**
  - Differed activation when memory is limited
- **Available on**
  - GDF on ObjectWeb SourceForge
  - http://www-adele.imag.fr/~donsez/dev/osgi/serviceondemand/

Project Roadmap

GDF

- Service Passivation
- Bundle configuration step
- Bundle Update/Uninstall (i.e. Bundle Garbage Collector)
- OMG D&C Compliance

ServiceOnDemand

- Used in OSGiTV *(COMPiTV project, French Gov. Funding, 2003)*
- Test with other OSGi FW (KF, Eclipse, SMF, …)

Part of the ITEA OSMOSE project