



Introduction to OSGi

Matthew Perrins

Executive IT Specialist

Martin Gale

Certified IT Specialist

IBM Software Group

Open Services Gateway Initiative

Agenda

- Why is OSGi technology important?
- What is OSGi technology?
- OSGi™ Component Programming
- What changes are coming in the future?
- Who is the OSGi Alliance?

Granularity

- ✦ While Object Oriented languages like Java help
 - ✦ They only help at the object and class level
 - ✦ There are no higher levels of **modularity**



What is Modularity?

- ✦ “The property of a system that has been decomposed into a set of cohesive and loosely coupled modules.”

engronline.ee.memphis.edu/networkprogram/Lectures/Lecture3/object_terms.htm

- ✦ “(Desirable) property of a system, such that individual components can be examined, modified and maintained independently of the remainder of the system. Objective is that changes in one part of a system should not lead to unexpected behavior in other parts.”

www.maths.bath.ac.uk/~jap/MATH0015/glossary.html

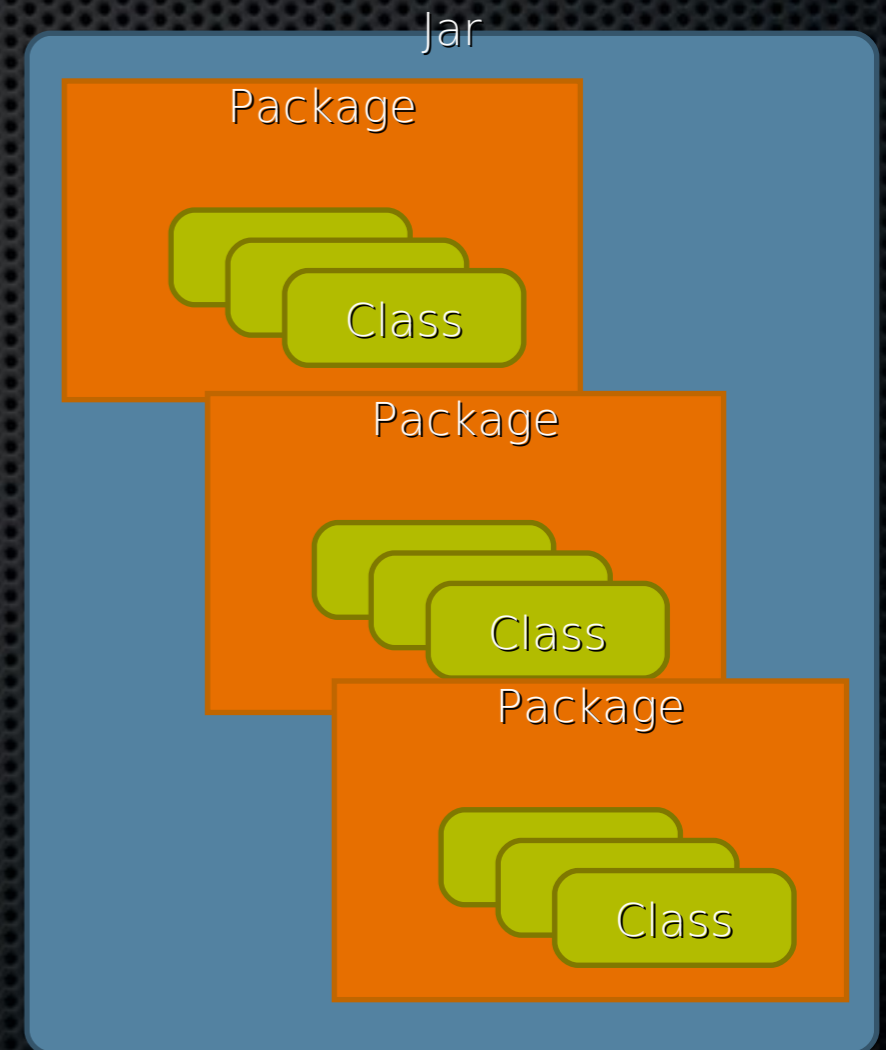
Open Source Proliferation

- ✦ Other open source projects consume each other
- ✦ Commercial software consumes them
- ✦ Integrating many OS projects is very difficult
 - ✦ Different dependencies of different projects
 - ✦ Conflicts and unexpected failures

Modularization in Java Apps

Visibility

- ▶ Java Platform Modularity
 - Classes encapsulate data
 - Packages contain classes
 - Jars contain packages
- ▶ Visibility Access
 - private, package private, protected, public
- ▶ Packages look hierarchical, but are not
- ▶ Jars have no modularization characteristics



Modularization in Java Apps

Classpath



JAR Hell!



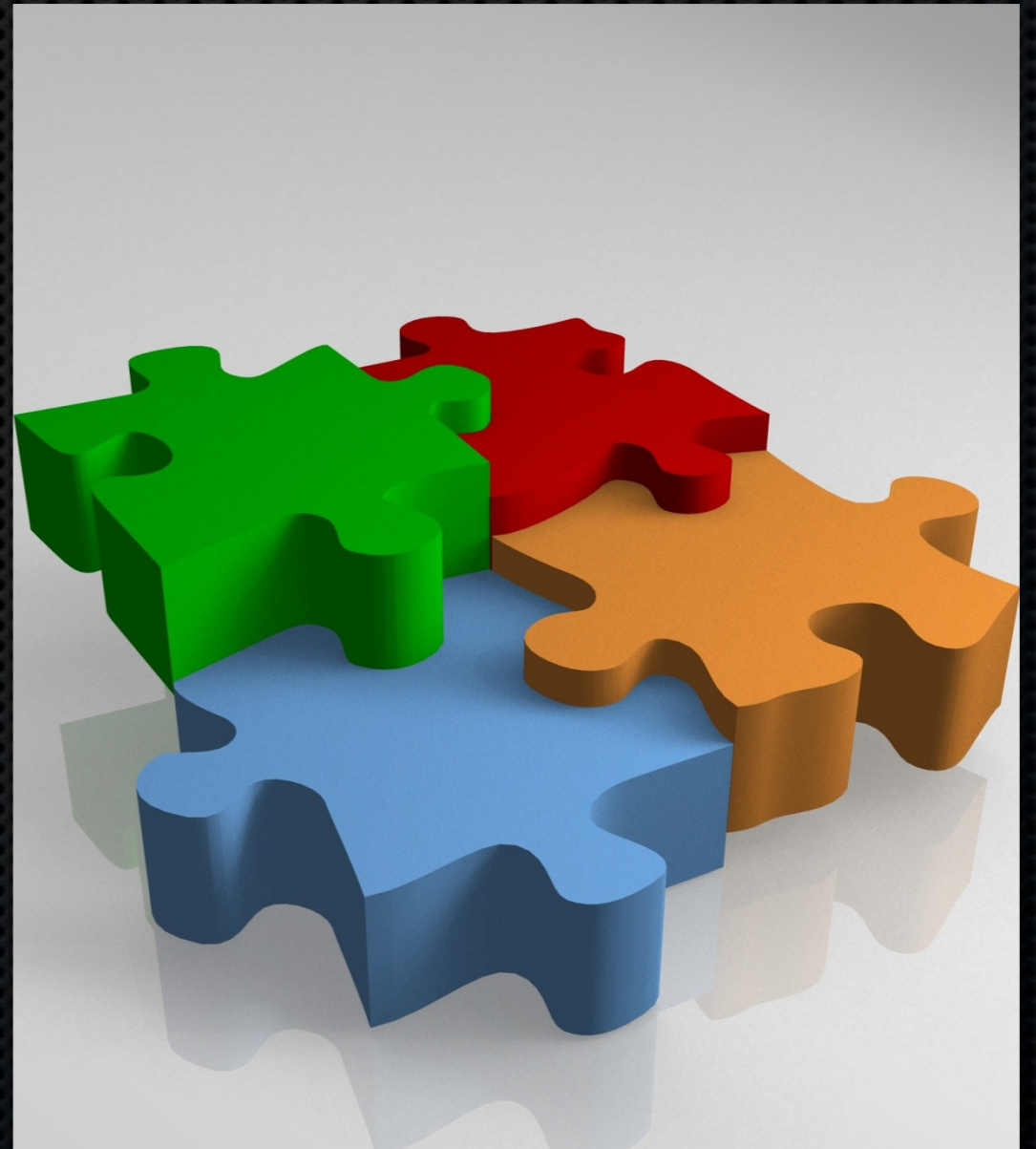


OSGi is the Solution

A mature, functional and ready-to-use technology that can address these issues

OSGi Modularity

- ✦ Higher level encapsulation
- ✦ Well specified coupling
- ✦ Explicit, versioned dependencies
- ✦ Reduced complexity
- ✦ Component reuse
- ✦ Solves JAR Hell!



Using OSGi

- Simple format: JAR file with manifest entries
- Easy to deploy
- Runs on most VMs including older versions
- Dynamic
 - Like the real world



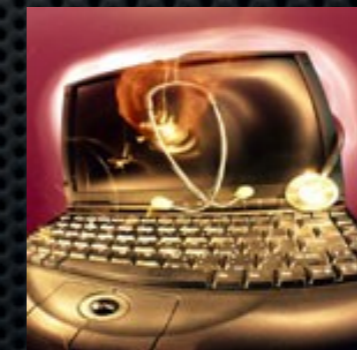
OSGi is Mature

- 10 years of experience and improvements
- 6 releases
- The *de facto* choice for Java components and modularity
- Used by Eclipse, IBM, Oracle, SpringSource, Sun, Redhat, ...
- Multiple open source implementations

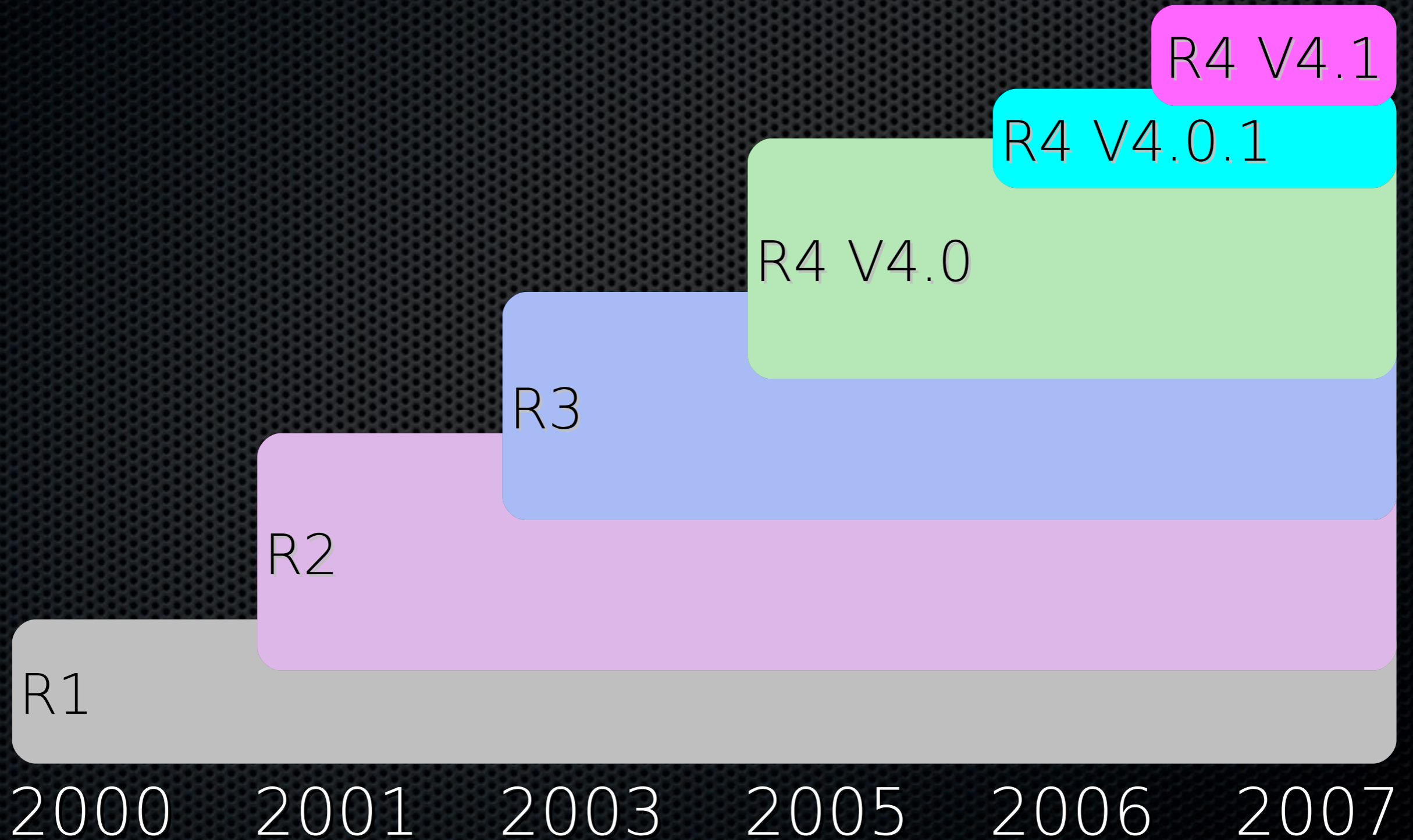
OSGi Adoption

Increasing traction in many verticals:

- Enterprise
- Vehicle
- Mobile
- "Smart" home
- More...
- Industrial automation
- Remote/Home Healthcare
- Mission Critical Embedded Solutions

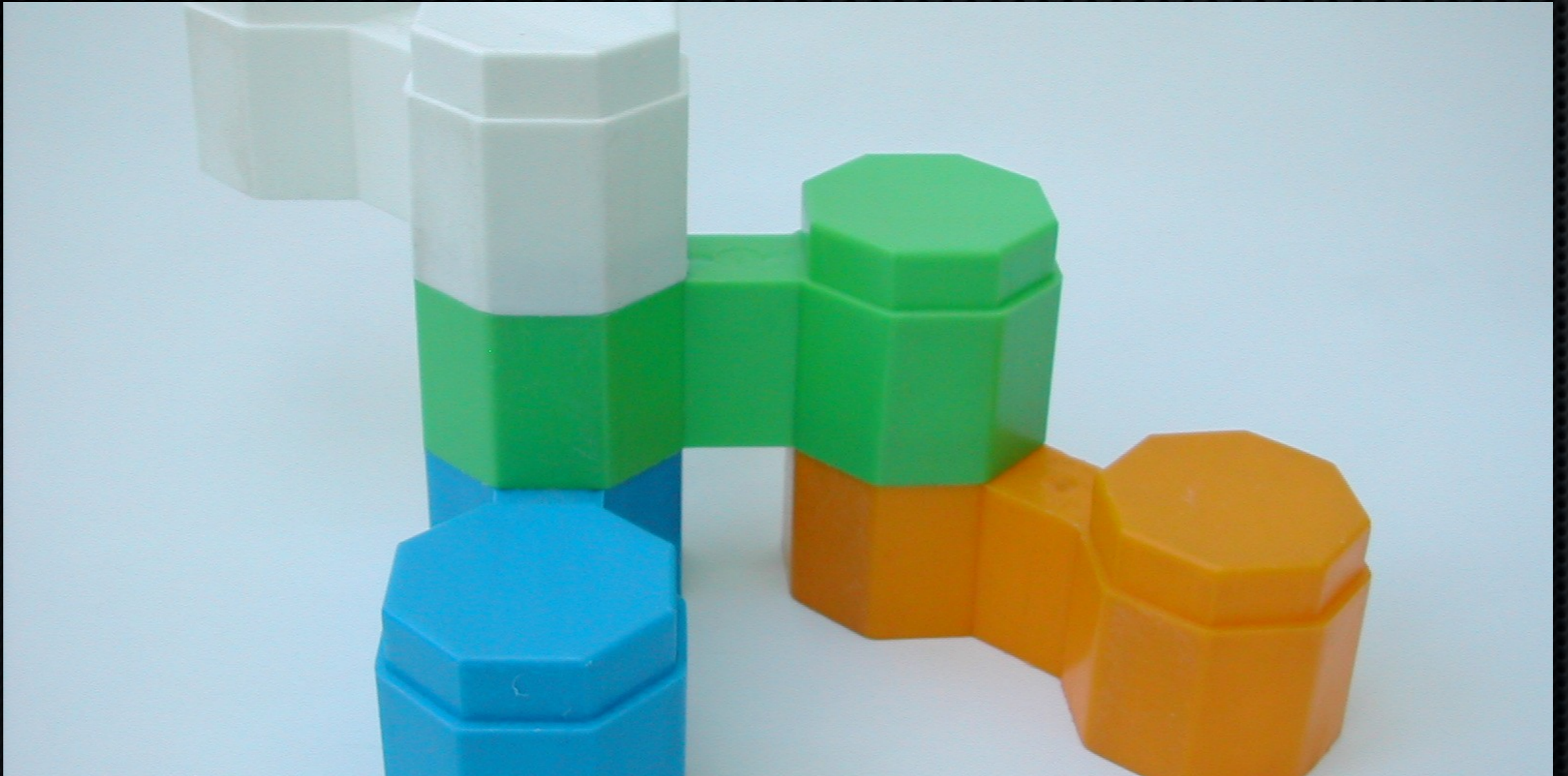


History



Agenda

- ✦ Why is OSGi technology important?
- ✦ What is OSGi technology?
- ✦ OSGi TM Component Programming
- ✦ What changes are coming in the future?
- ✦ Who is the OSGi Alliance?

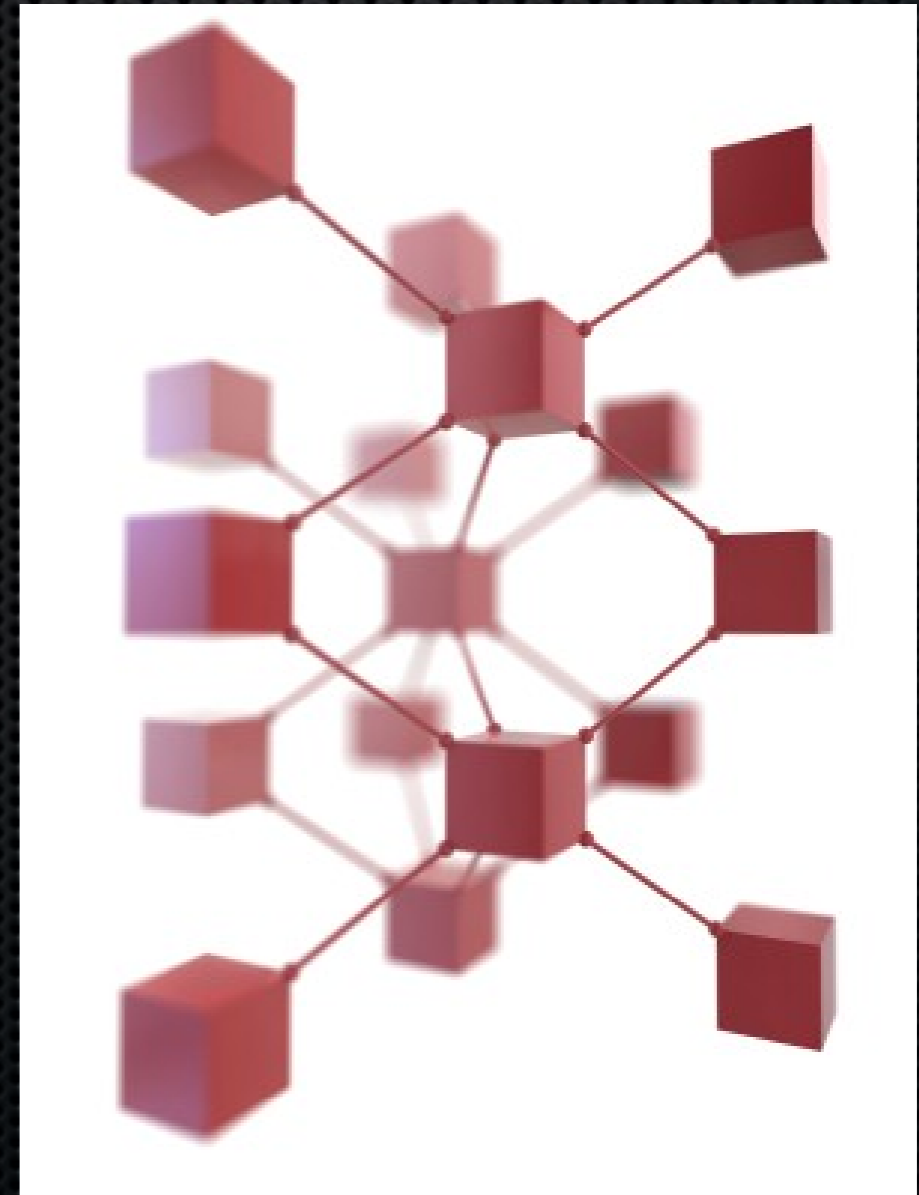


OSGi technology

The Dynamic Module System for Java™

OSGi is

- ✦ A module system
 - ✦ Bundles
- ✦ Visibility
- ✦ Dependencies
- ✦ Versioning



OSGi is

- ✦ Dynamic
 - ✦ Installing, starting, stopping, updating, uninstalling bundles, all dynamically at runtime



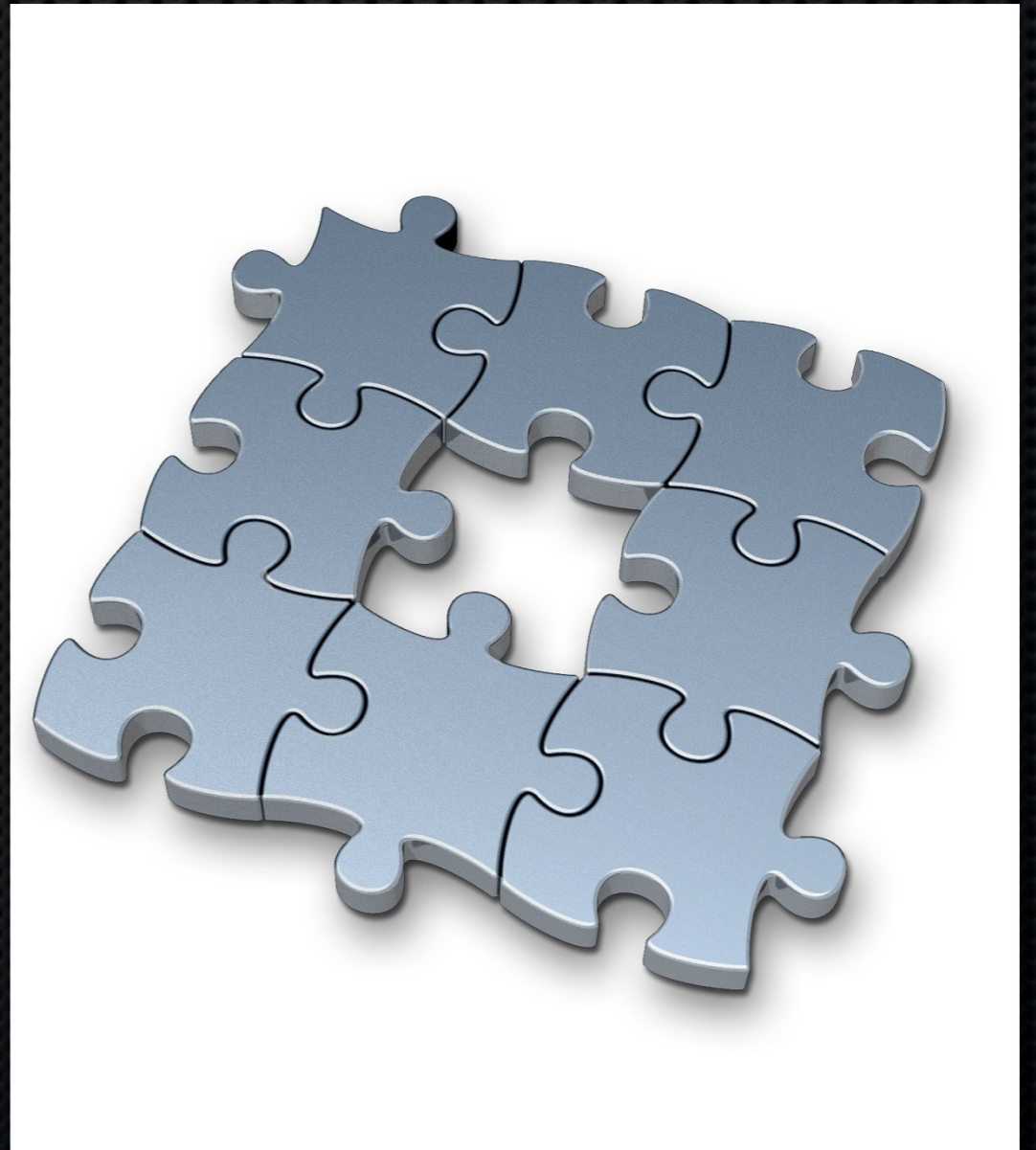
OSGi is

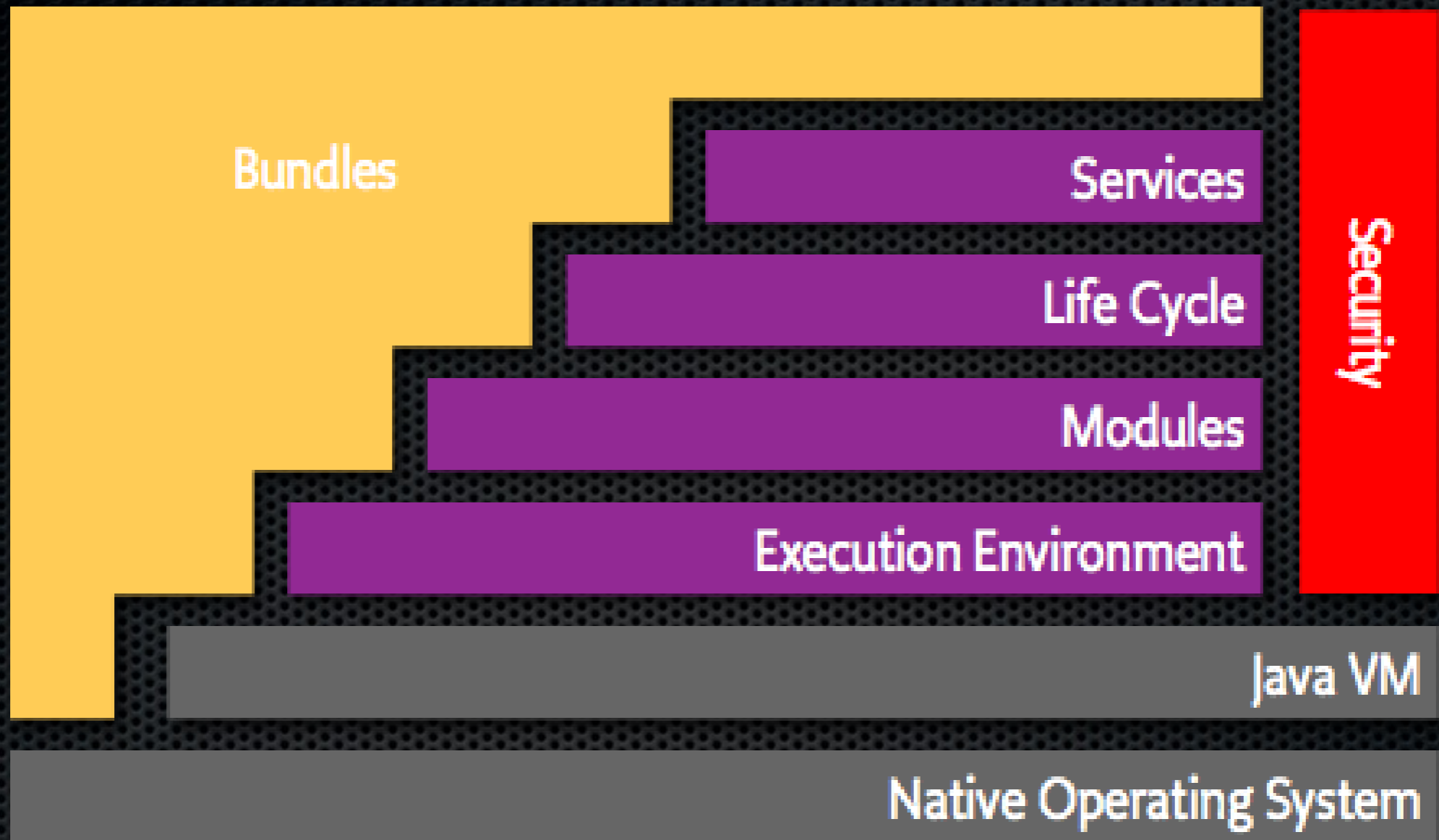
- Service oriented
 - Services can be registered and consumed inside a VM
 - All dynamically at runtime



OSGi Service Platform

- A set of specifications
 - Core specifications
 - Compendium specifications
- Compendium specifications are all optional

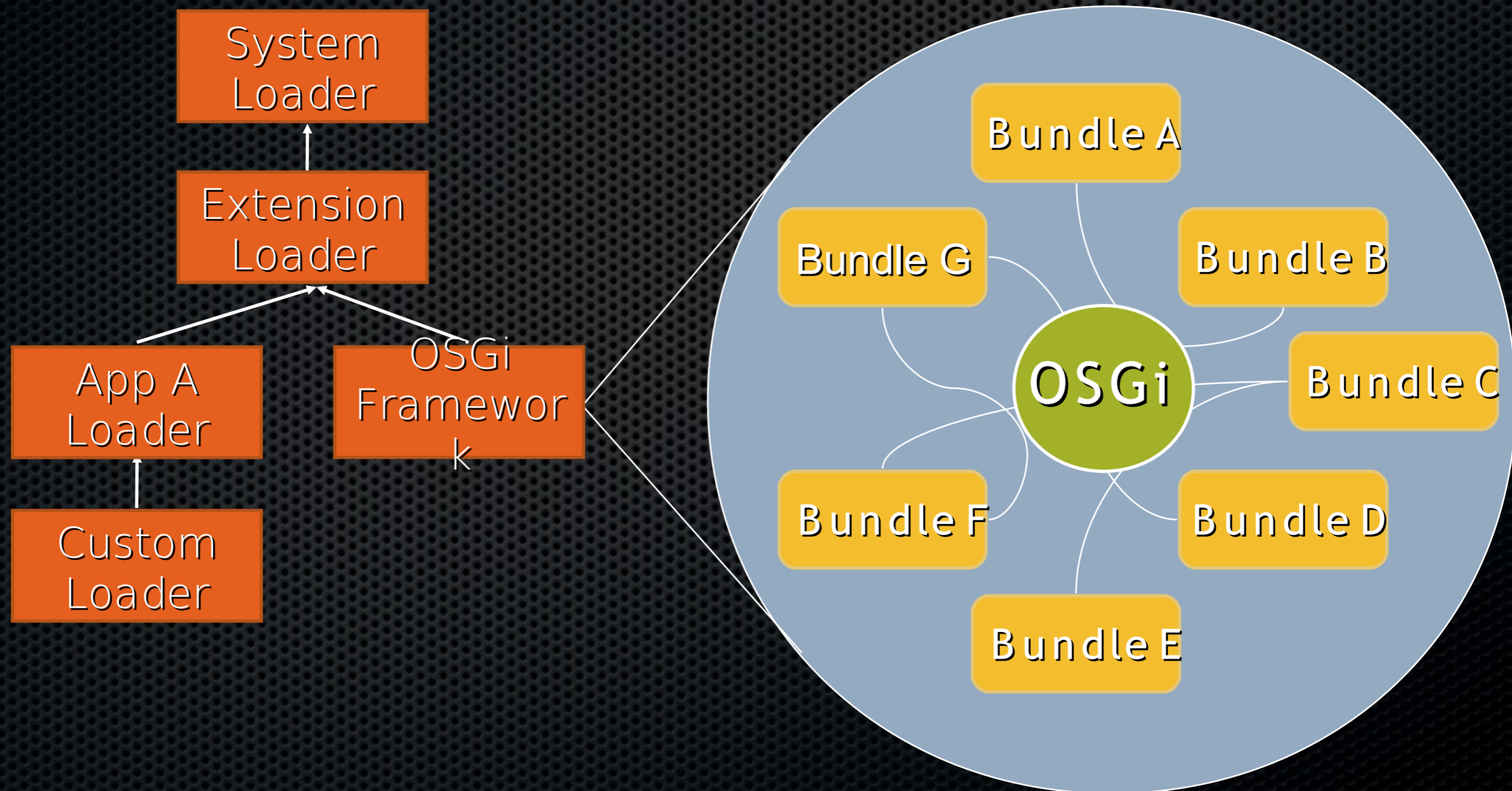




Core Specifications

The OSGi Framework

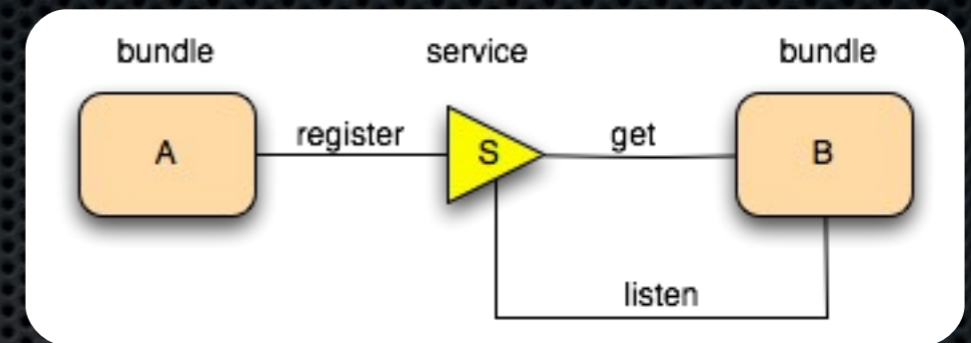
Networked Class Loaders on “Steroids”



Lifecycle Layer

- Provides APIs to control the life cycle of bundles
 - Install/Update/Uninstall/Start/Stop
 - Lifecycle event notifications
- API provides information on state of module layer
- Supports dynamic management of bundles in running VM

Service Layer



- ✦ Defines a publish/find/bind service model
 - ✦ Fully dynamic
 - ✦ Intra VM
 - ✦ Lifecycle event notifications
 - ✦ Non-durable service registry
- ✦ A service is a POJO published under one or more Java interface (or class) names with additional key/value pair metadata

Agenda

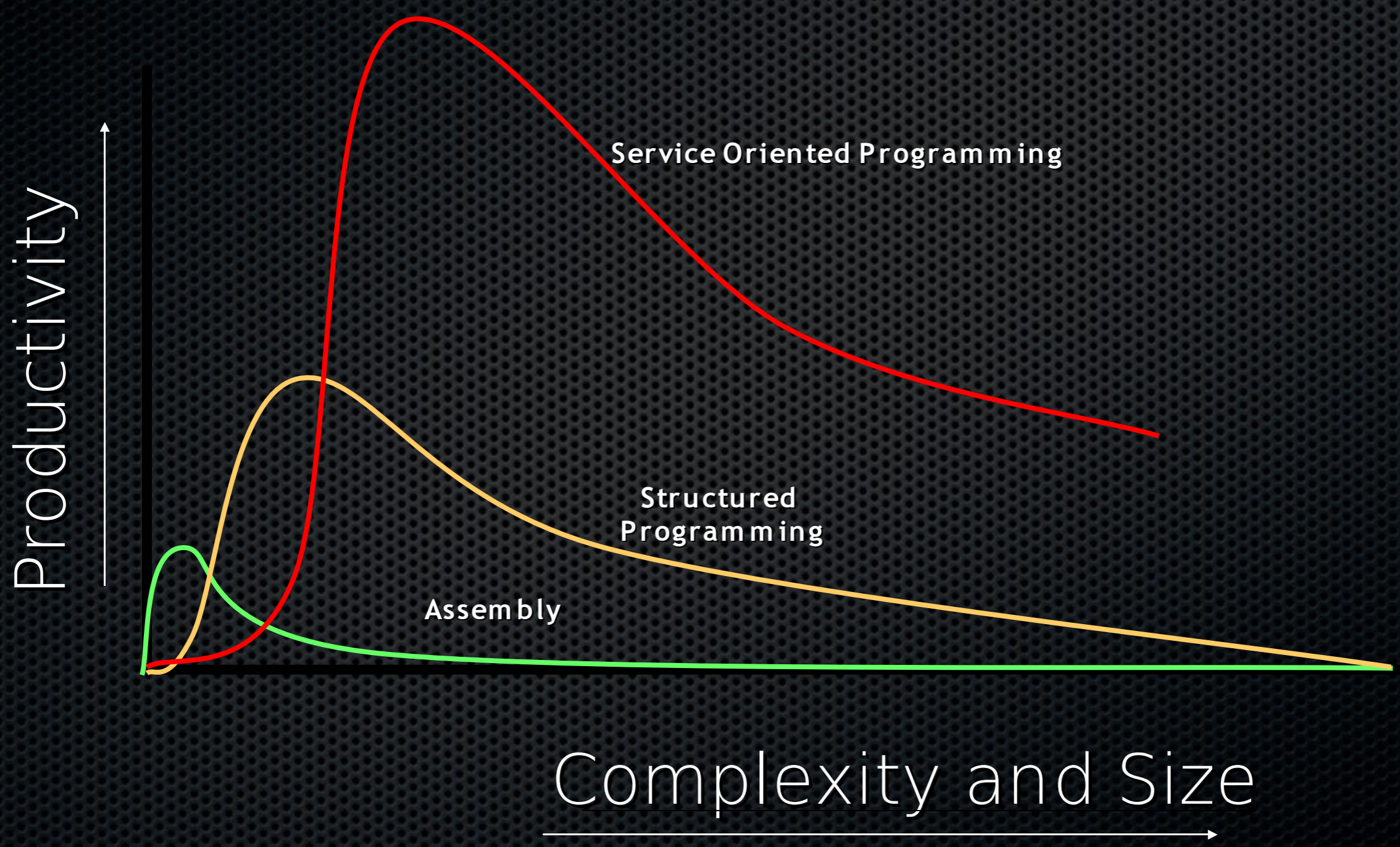
- ✦ Why is OSGi technology important?
- ✦ What is OSGi technology?
- ✦ OSGi™ Component Programming
- ✦ What changes are coming in the future ?
- ✦ Who is the OSGi Alliance?

OSGi™ Component Programming

Complexity of Software

- A DVD player can contain 1 Million lines of code
 - Comparison: Space Shuttle ~ 0.5 Million
- A BMW car can contain up to 50 networked computerized devices
- Eclipse contains 2.5 million lines of code
- An average programmer writes an average of 10 lines a day ...
- Houston ... we have a problem

Complexity of Software



Limits Object Oriented Technology

- Objects are great, but oh, the tangled webs we weave

...

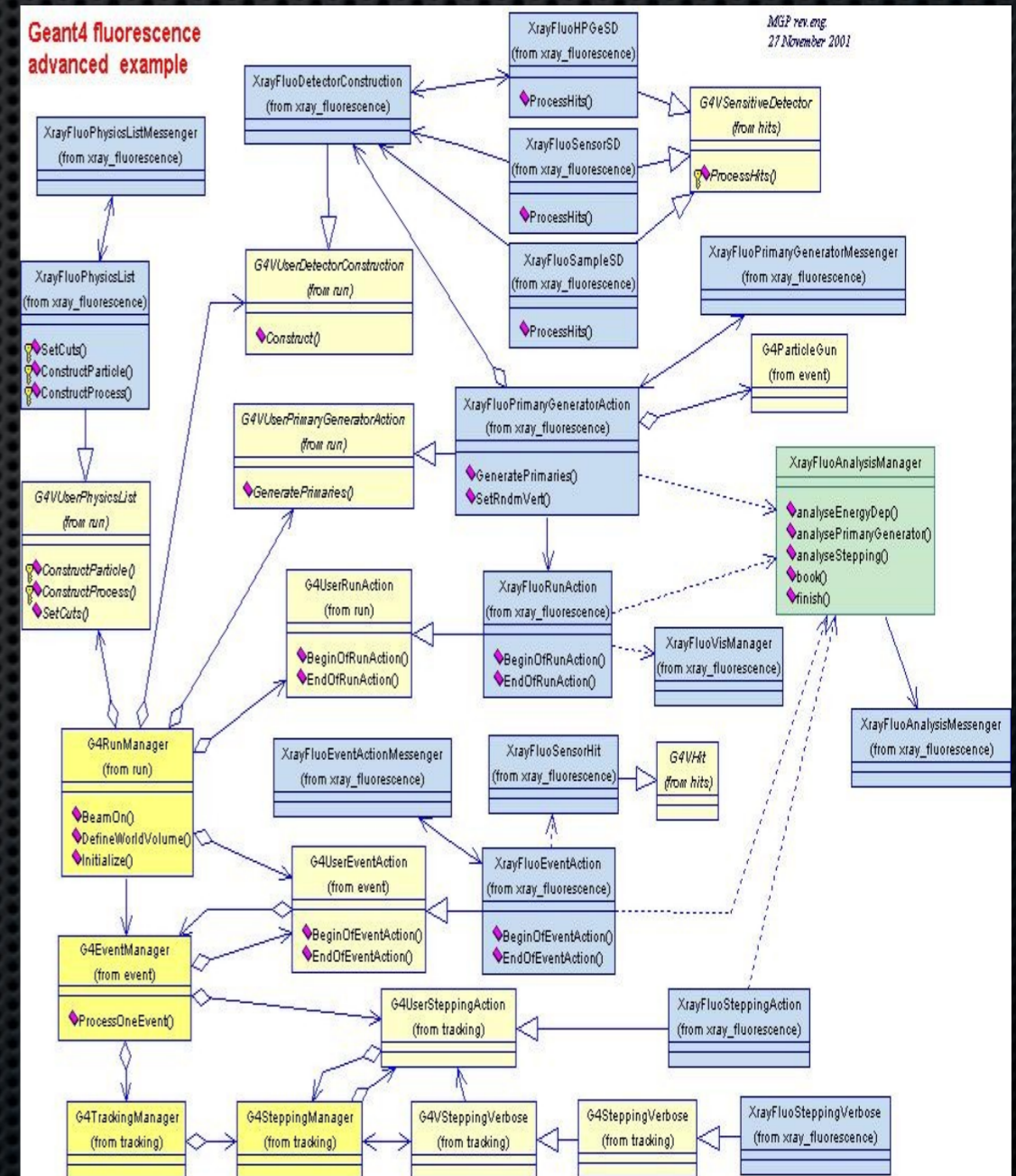
- Coupling severely limits reusability

- Using a generic object, can drag in a large number of other objects

- Creates overly large systems after a certain complexity is reached

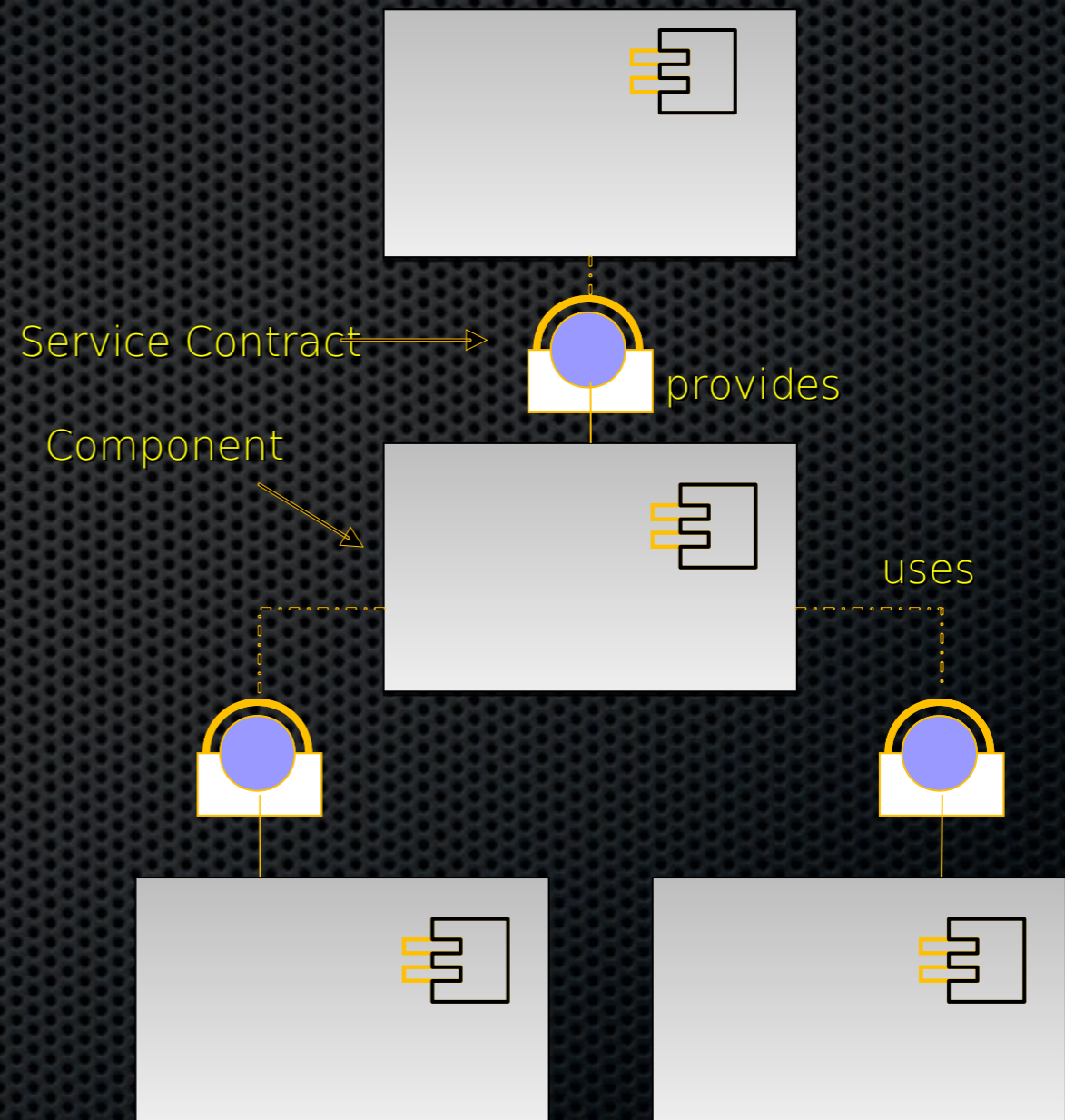
- Flexibility must be built in by the programmer

- *Plug-in* architectures

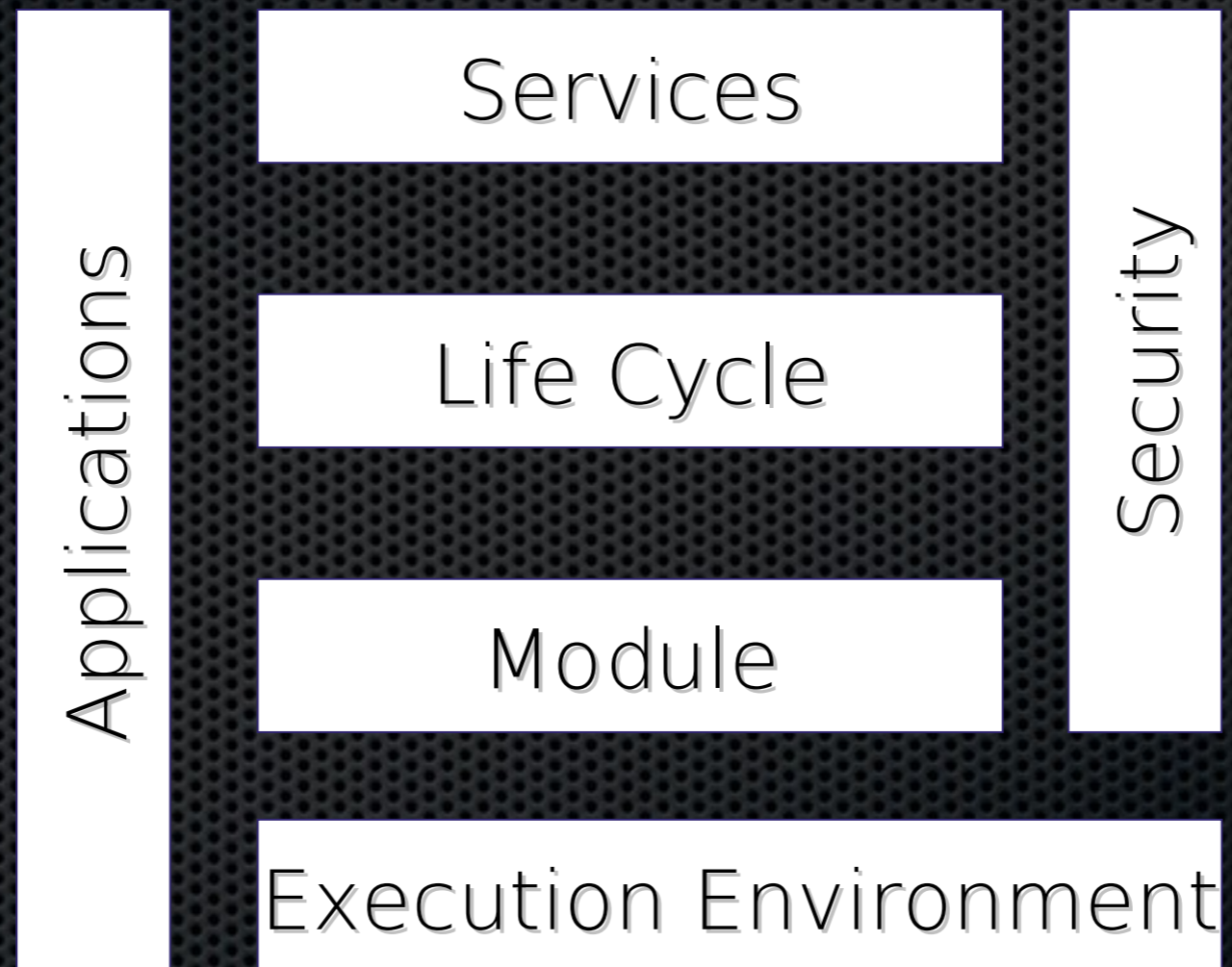


Service Oriented Architectures

- Separate the contract from the implementation
- Allows alternate implementations
- Dynamically discover and bind available implementations
- Based on contract (interface)
- Components are reusable
- Not coupled to implementation details

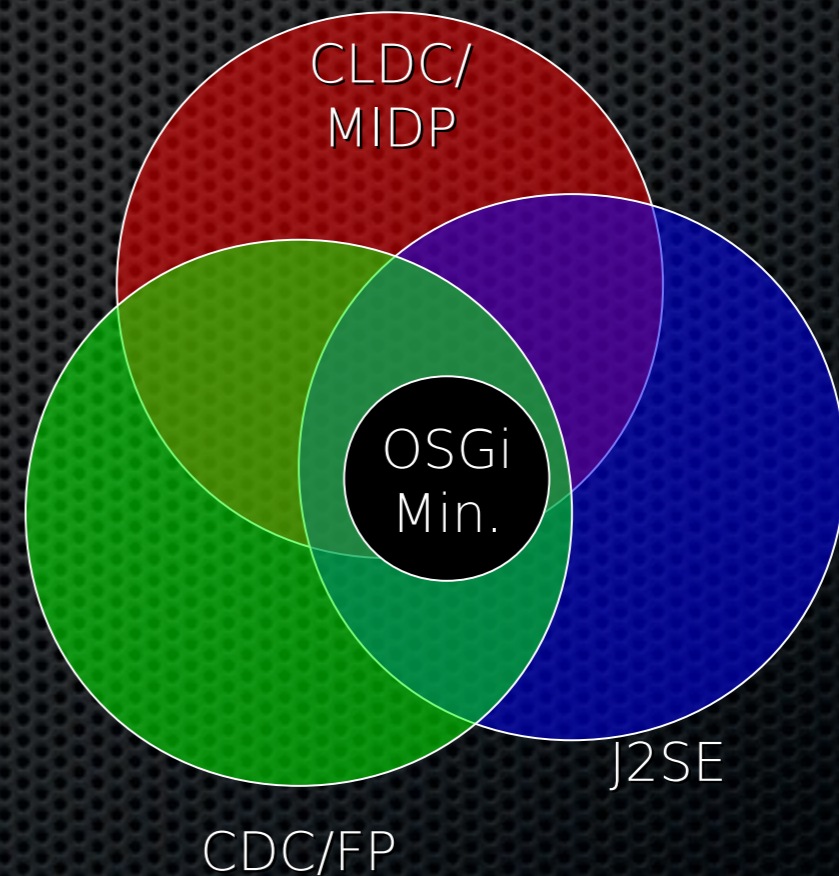


Layering



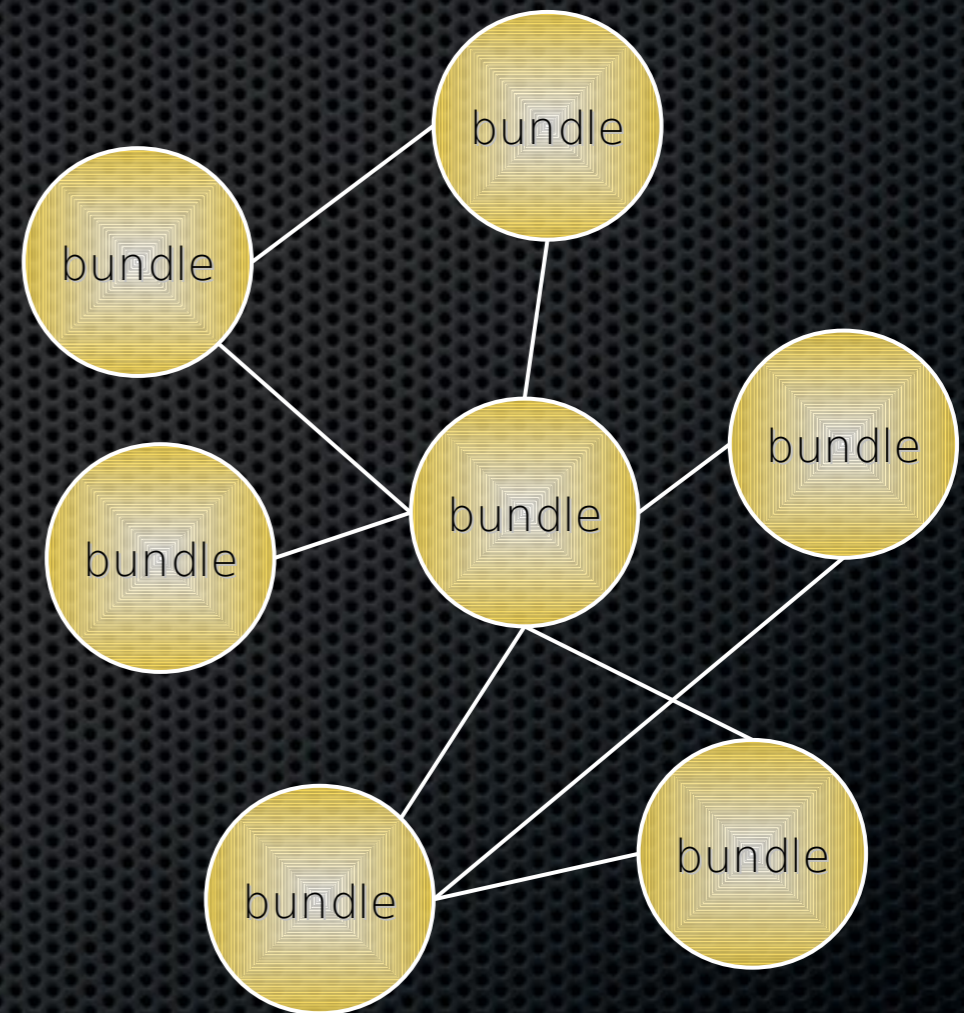
Execution Environment

- OSGi APIs only use a subset of J2SE and J2ME CDC
 - OSGi Minimum EE
- Matches most profiles
- Implementations can use more than the OSGi Minimum EE
- Security is not mandatory
- CLDC is possible if class loaders are added in a device specific way



Module Layer

- Packaging of applications and libraries in *Bundles*
 - Raw Java has significant deployment issues
- Class Loading modularization
 - Raw Java provides the Class Path as an ordered search list, which makes it hard to control multiple applications
- Protection
 - Raw Java can not protect certain packages and classes
- Versioning
 - Raw Java can not handle multiple versions of the same package

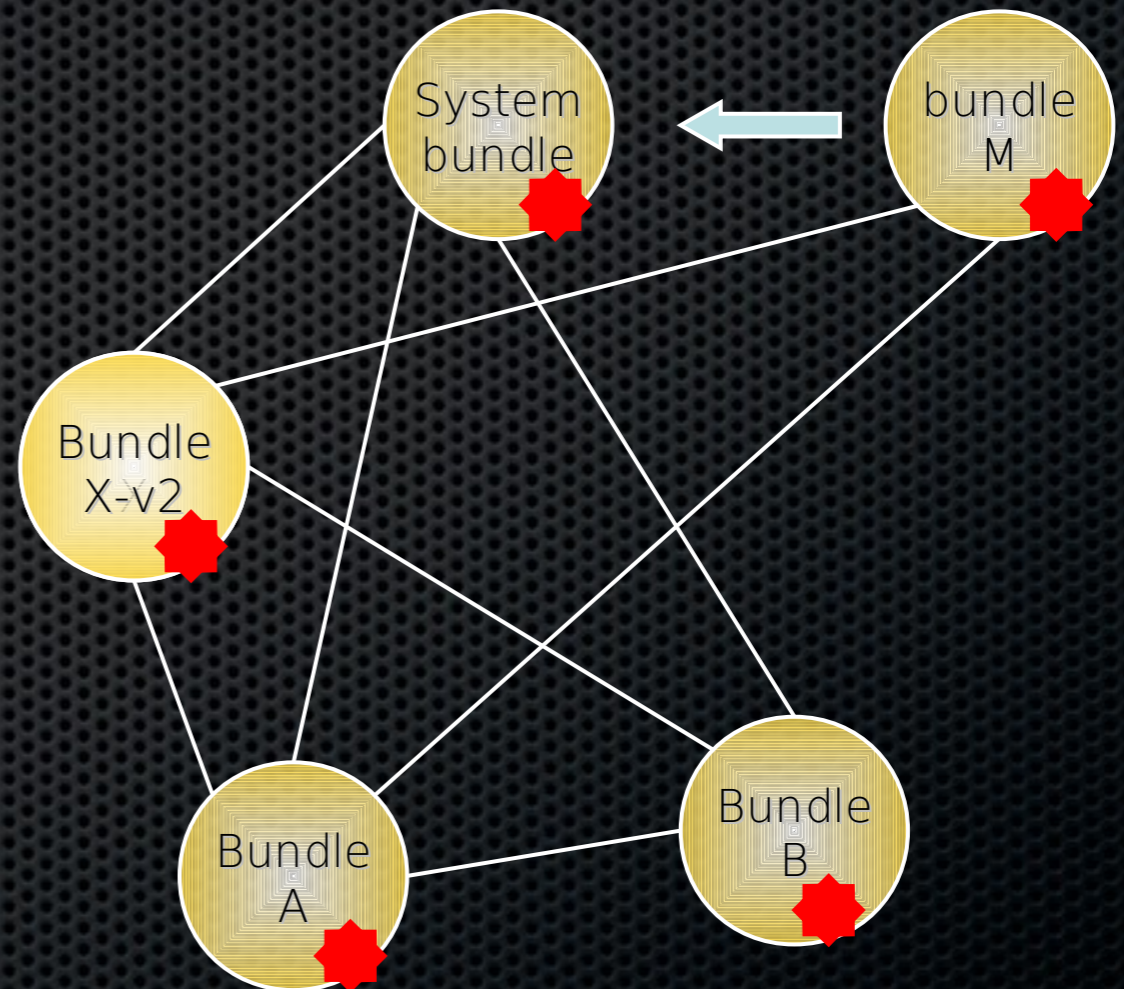


Life Cycle Layer

- System Bundle represents the OSGi Framework
- Provides an API for managing bundles

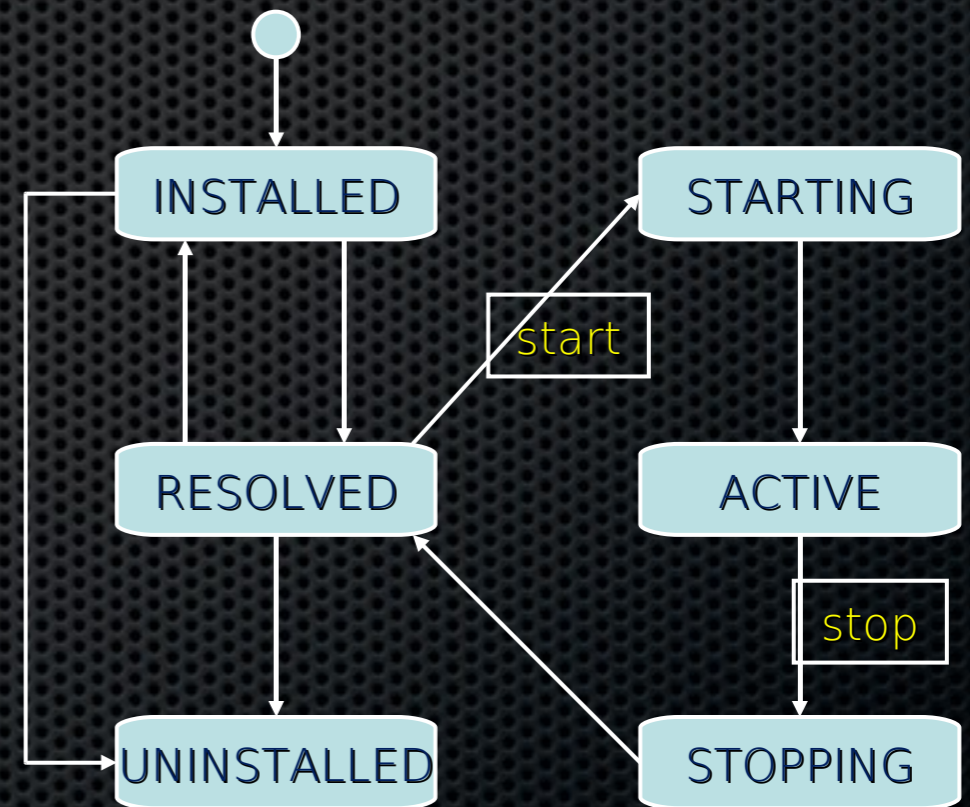
- Install
- Resolve
- Start
- Stop
- Refresh
- Update
- Uninstall

- Based on the module layer



Life Cycle Layer

- Bundle is started by the Bundle Activator class
- Header in Manifest refers to this class
- Interface has 2 methods
 - Start: Initialize and return immediate
 - Stop: Cleanup
- The Activator gets a Bundle Context that provides access to the Framework functions
- Framework provides Start Level service to control the start/stop of groups of applications



Service Layer

- Provides an in-VM service model
 - Discover (and get notified about) services based on their interface or properties
 - Bind to one or more services by
 - program control,
 - default rules, or
 - deployment configuration
- SOA Confusion
 - Web services bind and discover over the net
 - The OSGi Service Platform binds and discovers inside a Java VM
- The OSGi Alliance provides many standardized services

Benefits of Using the OSGi Service Platform

- Components are smaller
 - Easier to make
- Components are not coupled to other components
 - Gives reusability
- Excellent model for the myriad of customizations and variation that are required of today's devices
- Collaborative model
 - Allows reuse of other components for most problems

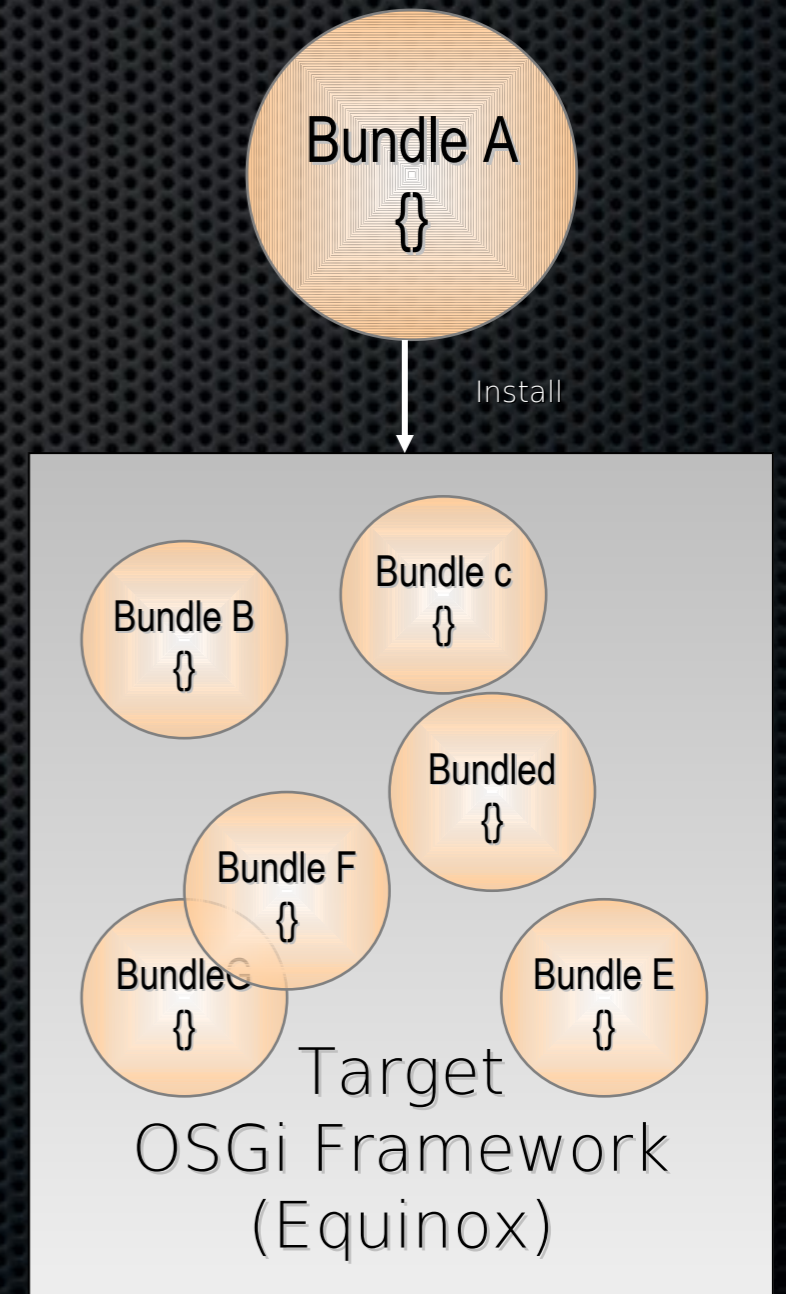
Eclipse/Equinox

What is Equinox ?

- An open source community focused on OSGi Technology
 - Develop OSGi specification implementations
 - Prototype ideas related to OSGi
- An OSGi Framework implementation
 - Core of the Eclipse runtime
 - Provides the base for Eclipse plug-in collaboration
 - Fully compatible with the OSGi R4 specification
- New for Eclipse 3.2 – Other specification implementations
 - Device Manager, Declarative Services, Event Admin, HTTP Service, Log Service, Metatype Service, Preferences Service, User Admin, Wire Admin – More on the way!!

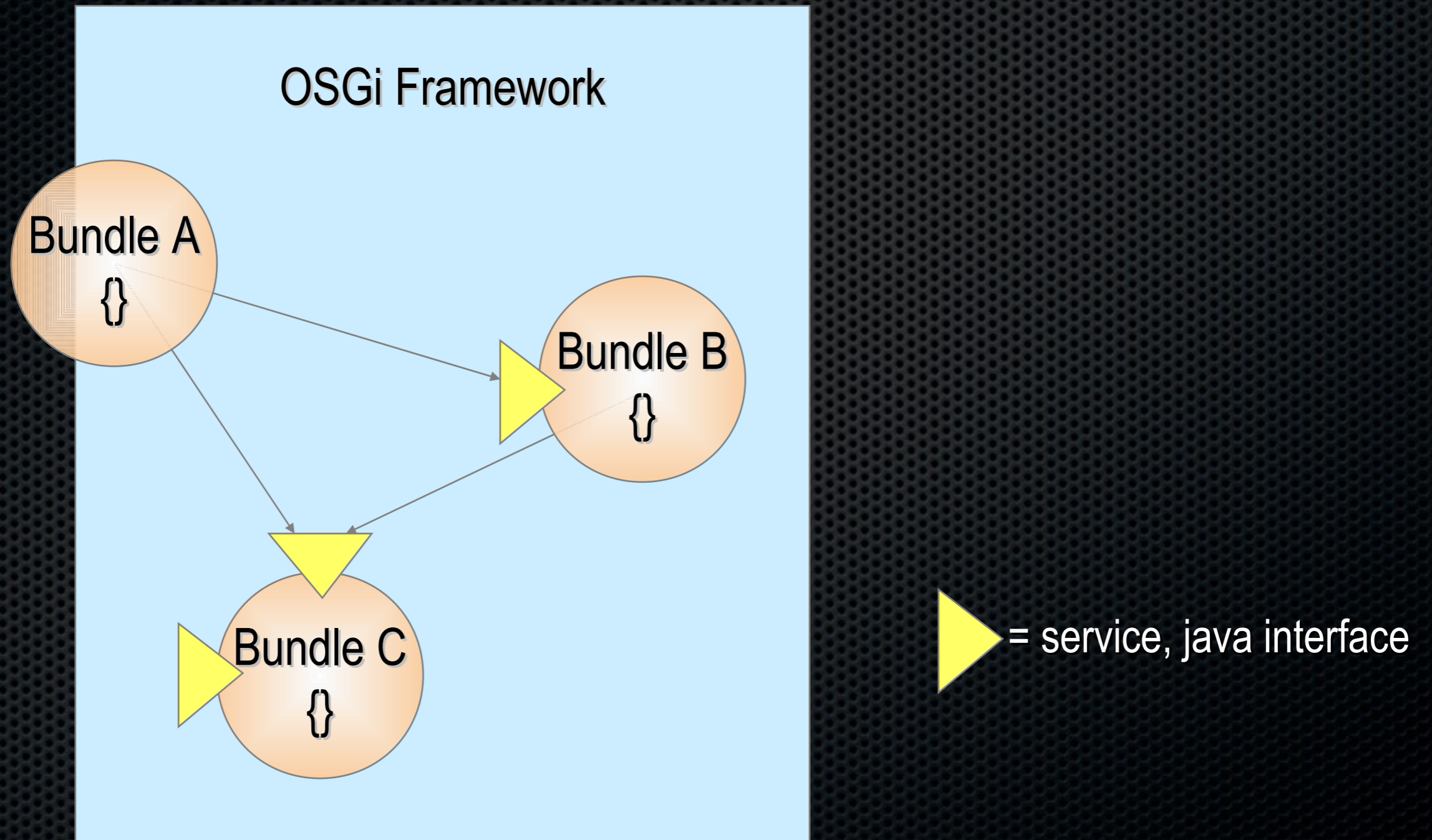
The Equinox Target Environment

- Eclipse makes it *easy* to develop for OSGi Service Platforms
- *A target platform*
 - Contains a set of bundles
 - Defines runtime parameters
- To Define the Target Platform, goto:
 - Preferences -> Plug-in Development -> Target Platform
 - Select the target project in your workspace as location



Fundamental OSGi concepts

Framework Entities

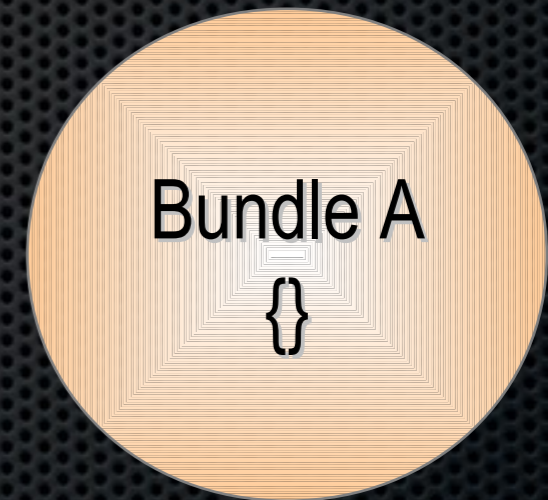


Bundles

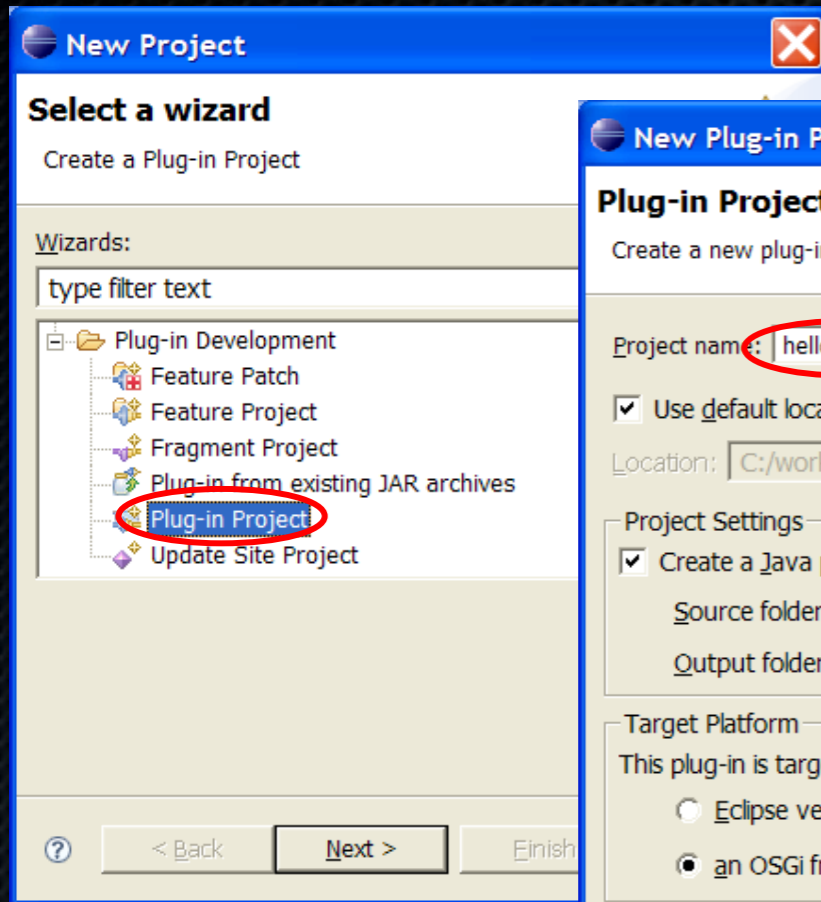
- A *bundle* is the deliverable application
 - Like a Windows EXE file
 - Content is a JAR file
- A bundle registers zero or more services
 - A service is specified in a Java interface and may be implemented by multiple bundles
 - Services are bound to the bundle life-cycle
- Searches can be used to find services registered by other bundles
 - Query language

What is in a Bundle?

- A Bundle contains (normally in a JAR file):
 - Manifest
 - Code
 - Resources
- The Framework:
 - Reads the bundle's manifest
 - Installs the code and resources
 - Resolves dependencies
- During Runtime:
 - Calls the Bundle Activator to start the bundle
 - Manages java classpath
 - Handles the service dependencies
 - Calls the Bundle Activator to stop the bundle

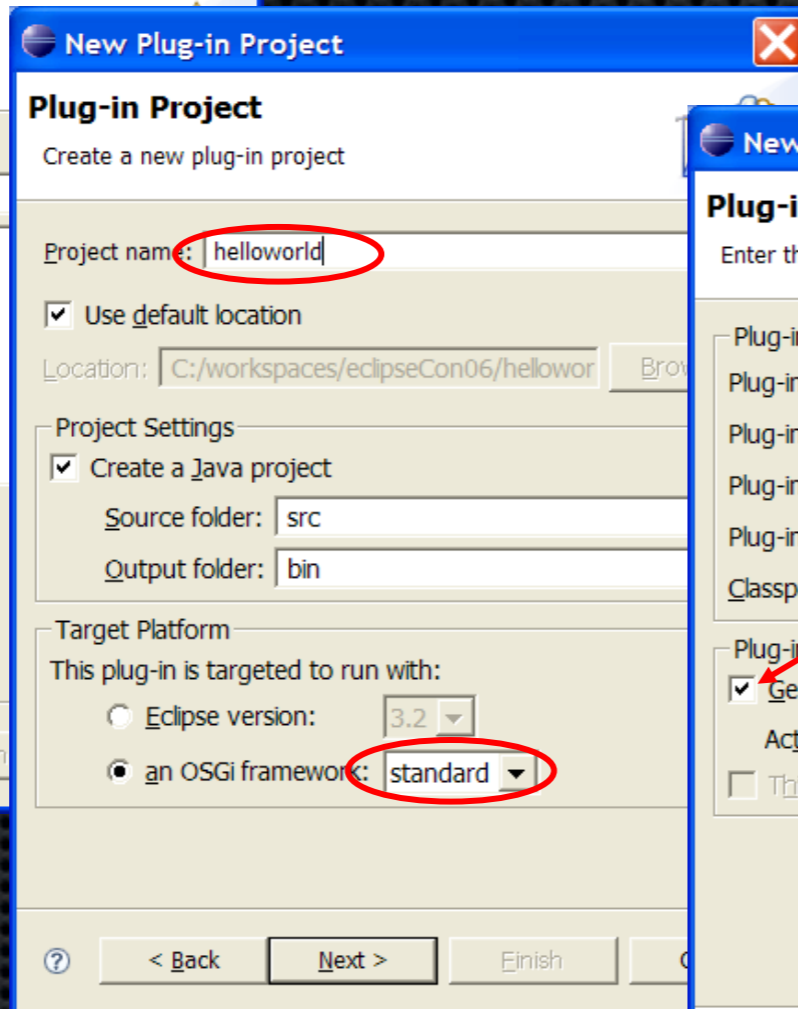


Create the Hello World bundle



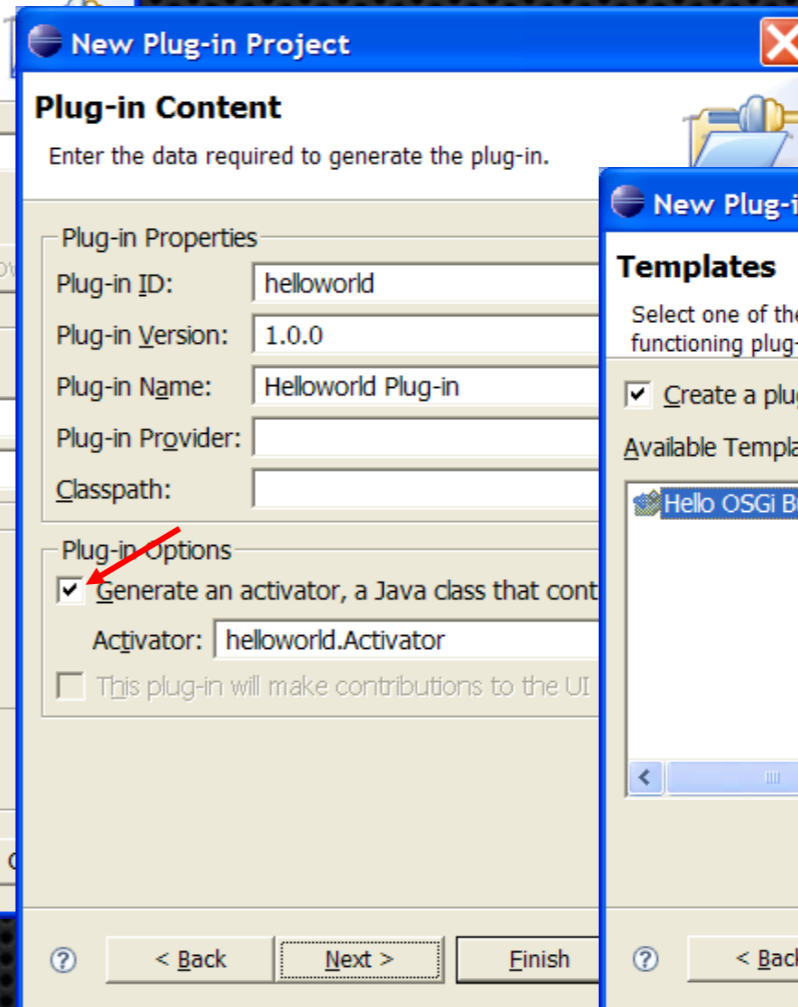
Step 1.

Create new plug-in project



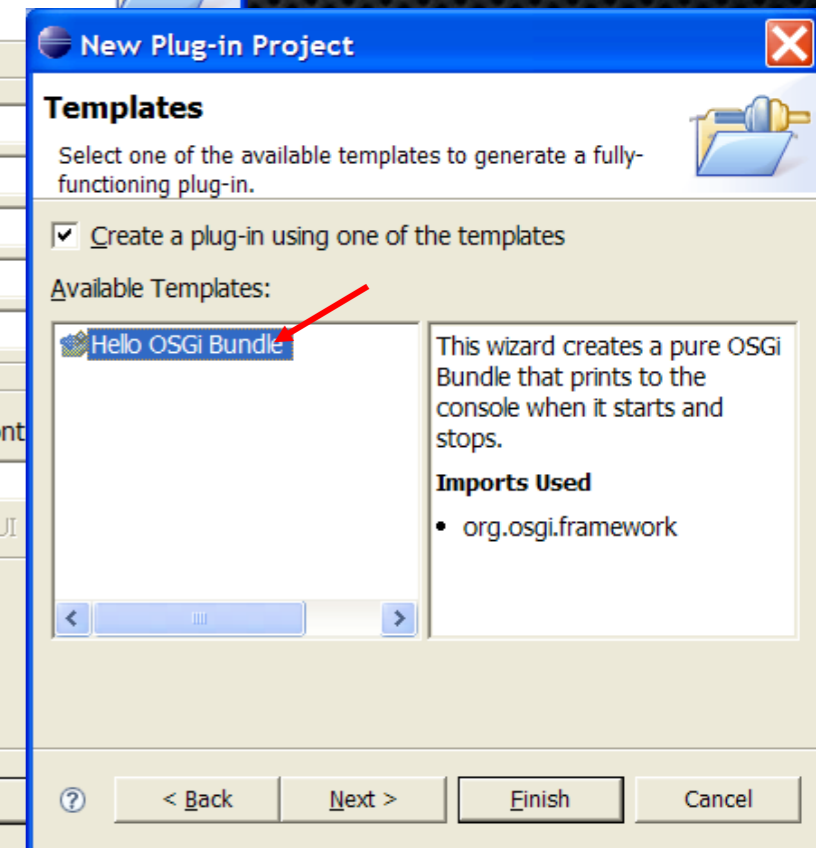
Step 2

Project name: helloworld
an OSGi framework: standard



Step 3

Generate an activator



Step 4

Use the Hello OSGi Bundle
template

Real code! Hello World (and Goodbye)

- The wizard has generated the code on the left
- This class implements the BundleActivator so that the Framework can start/stop the class
- The activator is referenced in the manifest

HelloWorld.java

```
package helloworld

public class HelloWorld

    implements BundleActivator {

    public void start(

        BundleContext context)

        throws Exception{

        System.out.println(

            "Hello world!!");

    }

    public void stop(

        BundleContext context)

        throws Exception {

        System.out.println(

            "Goodbye world!!");

    }

}
```

Real code! Hello World (and Goodbye)

- The Manifest (in META-INF/MANIFEST.MF) is also generated by the wizard
- Eclipse provides convenient editors for the manifest
 - For the source: click on MANIFEST.MF
- Notice:
 - Bundle-Activator (used to notify the bundle of lifecycle changes)
 - Import-Package (dependencies)

META-INF/MANIFEST.MF

Manifest-Version: 1.0

Bundle-ManifestVersion: 2

Bundle-Name: Helloworld Plug-in

Bundle-SymbolicName: helloworld

Bundle-Version: 1.0.0

Bundle-Localization: plugin

Bundle-Activator: helloworld.Activator

Import-Package:

org.osgi.framework;version="1.3.0"

Eclipse Launch Configuration

- The Launch Configuration is prepared for you
 - Run -> Run ... -> EclipseTutorial
- Deselect "Target Platform" checkbox
 - This removes all possible bundles from the launch configuration
- Select "Add Required Plug-ins"
 - This calculates from the dependency information, which bundles are required to run our *helloworld* example
- Press Run
- The Framework is a console application

Equinox Launch Configuration

The screenshot shows the Eclipse IDE's 'Run' dialog box for configuring the launch of the Equinox OSGi framework. The dialog is titled 'Run' and has a subtitle 'Create, manage, and run configurations'. Below the subtitle, it says 'Create a configuration to launch the Equinox OSGi framework.' The main area is divided into a left sidebar and a main configuration area. The sidebar shows a tree view of configurations: Eclipse Application, Equinox OSGi Framework (expanded), EquinoxTutorial (selected), Java Applet, Java Application, JUnit, JUnit Plug-in Test, and SWT Application. The main configuration area has a 'Name' field containing 'EquinoxTutorial'. Below the name, there are tabs for 'Plug-ins', 'Arguments', 'Settings', 'Tracing', 'Environment', and 'Common'. The 'Plug-ins' tab is active. It shows a table of plug-ins with columns for 'Plug-ins', 'Start Level', and 'Start'. The 'Default start level' is set to '1' and 'Start plug-ins automatically (Default)' is set to 'true'. The table lists 'Workspace Plug-ins' and 'Target Platform'. Under 'Workspace Plug-ins', 'helloworld (1.0.0)' is selected. Under 'Target Platform', several Equinox plug-ins are listed, including 'org.eclipse.equinox.common (1.0.0)', 'org.eclipse.equinox.device (1.0.0.v2', 'org.eclipse.equinox.ds (1.0.0.v2006', 'org.eclipse.equinox.event (1.0.0.v20', and 'org.eclipse.equinox.http (1.0.0.v200'. To the right of the table are buttons for 'Select All', 'Deselect All', 'Add Working Set...', 'Add Required Plug-ins', and 'Restore Defaults'. Below the table, there are three checkboxes: 'Include optional dependencies when computing required plug-ins' (checked), 'Add new workspace plug-ins to this launch configuration automatically' (checked), and 'Validate plug-in dependencies automatically prior to launching' (unchecked). At the bottom right, there are buttons for 'Apply', 'Revert', 'Run', and 'Close'. A status bar at the bottom left says 'Filter matched 8 of 8 items'.

Run Create, manage, and run configurations
Create a configuration to launch the Equinox OSGi framework.

Name: EquinoxTutorial

Plug-ins Arguments Settings Tracing Environment Common

Default start level: 1 Start plug-ins automatically (Default): true

Plug-ins	Start Level	Start
<input checked="" type="checkbox"/> Workspace Plug-ins		
<input checked="" type="checkbox"/> helloworld (1.0.0)	default	default
<input checked="" type="checkbox"/> Target Platform		
<input type="checkbox"/> org.eclipse.equinox.common (1.0.0.		
<input type="checkbox"/> org.eclipse.equinox.device (1.0.0.v2		
<input type="checkbox"/> org.eclipse.equinox.ds (1.0.0.v2006		
<input type="checkbox"/> org.eclipse.equinox.event (1.0.0.v20		
<input type="checkbox"/> org.eclipse.equinox.http (1.0.0.v200		

Include optional dependencies when computing required plug-ins
 Add new workspace plug-ins to this launch configuration automatically
 Validate plug-in dependencies automatically prior to launching

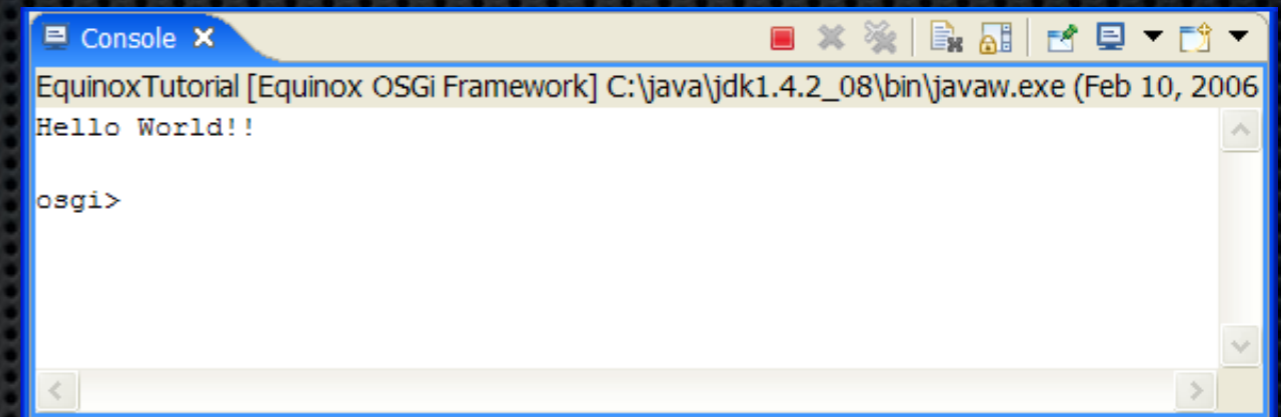
Buttons: Select All, Deselect All, Add Working Set..., Add Required Plug-ins, Restore Defaults, Apply, Revert, Run, Close

2 out of 27 selected

Filter matched 8 of 8 items

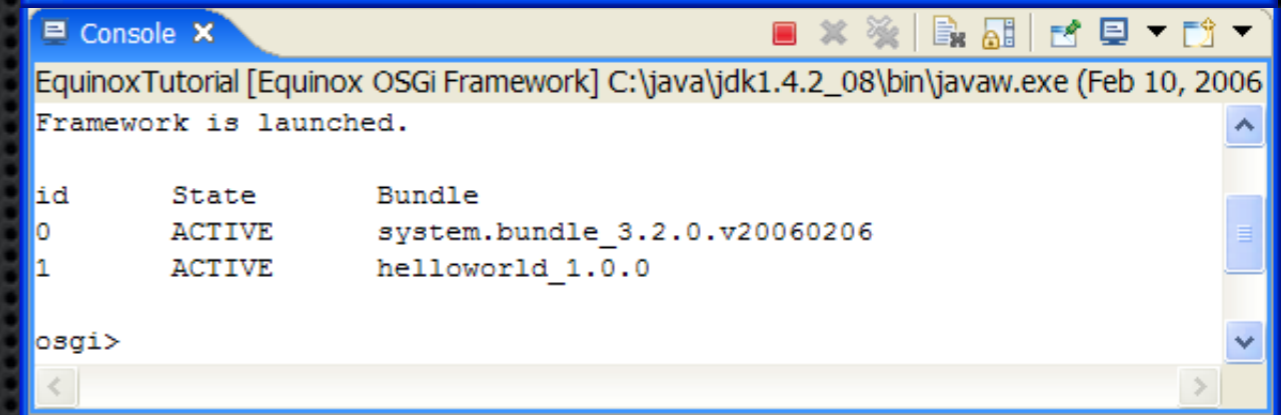
Run the Hello World bundle

- The Framework now runs the *helloworld* example
 - See the printed text
- It also runs a Framework console
 - Equinox specific
- Type "ss" (show status)
 - Look at the active bundles
 - Notice the number for the helloworld bundle. This is the bundle id.
- Type "stop <symbolic-name>" || bundle number



```
EquinoxTutorial [Equinox OSGi Framework] C:\java\jdk1.4.2_08\bin\javaw.exe (Feb 10, 2006)
Hello World!!

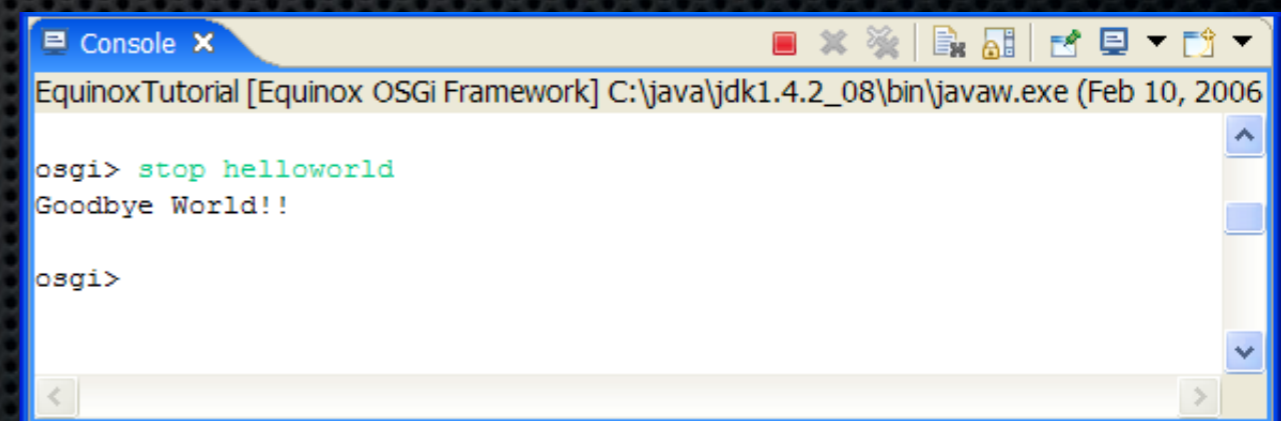
osgi>
```



```
EquinoxTutorial [Equinox OSGi Framework] C:\java\jdk1.4.2_08\bin\javaw.exe (Feb 10, 2006)
Framework is launched.

id      State      Bundle
0       ACTIVE    system.bundle_3.2.0.v20060206
1       ACTIVE    helloworld_1.0.0

osgi>
```



```
EquinoxTutorial [Equinox OSGi Framework] C:\java\jdk1.4.2_08\bin\javaw.exe (Feb 10, 2006)

osgi> stop helloworld
Goodbye World!!

osgi>
```

Creating deployable bundles – how it works

- The build.properties file specifies the content of the bundle jar

- Specifies the source and output folders of the different libraries

- source.. – The source directory of the project. Used for compilation and resources.

- output.. – The output directory where class files and resources are copied to

- bin.includes – What is included in the JAR from the project directory

- Export the content of a project into a bundle jar

- Bundle jars can be installed across multiple OSGi Framework implementations

- The *Deployable plug-ins and fragments* wizard can be used to generate a bundle jar from a project.

- File -> Export -> Deployable plug-ins and fragments

build.properties

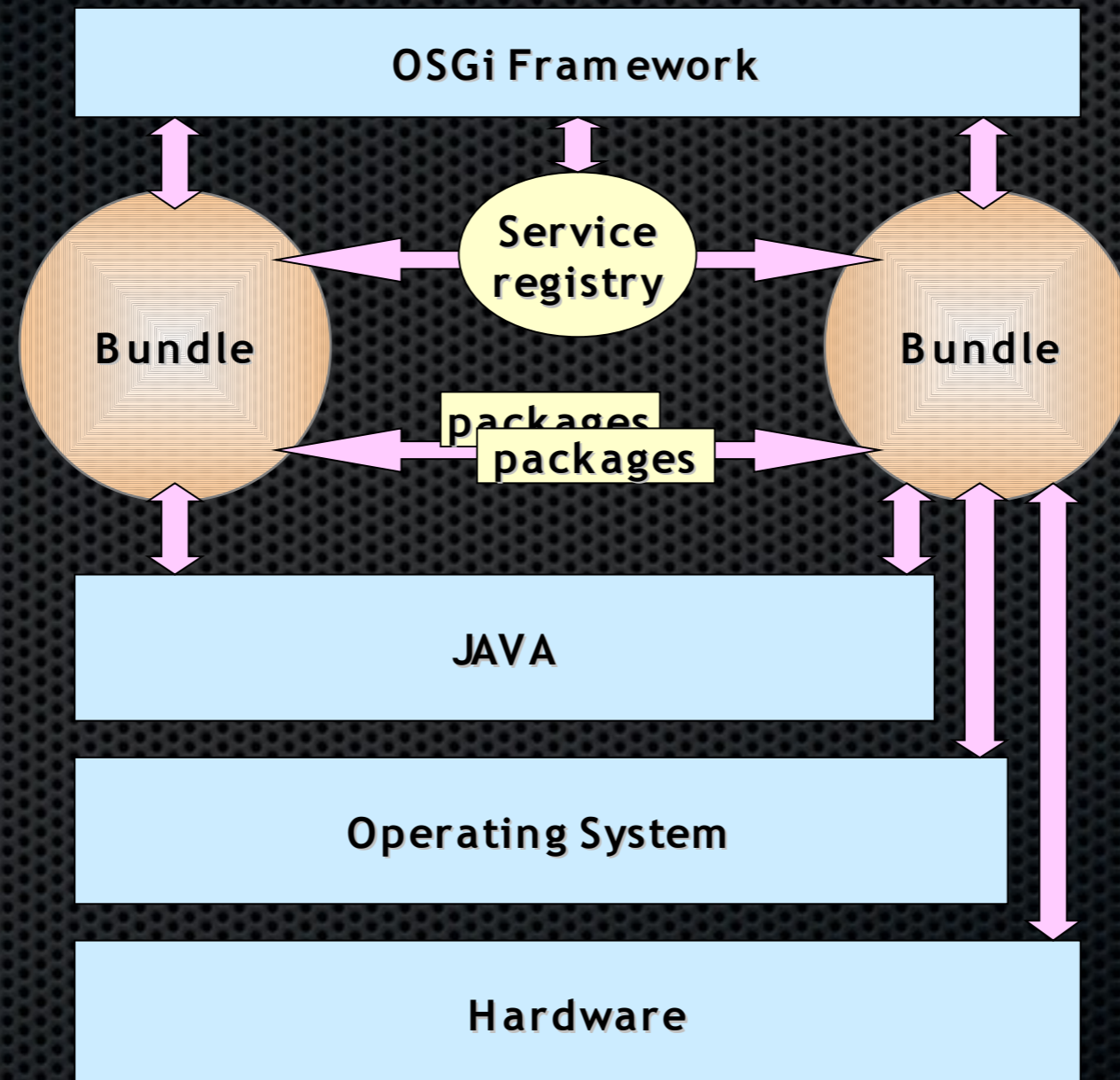
```
source.. = src/  
output.. = bin/  
bin.includes = META-INF/, \
```

Component interaction and collaboration

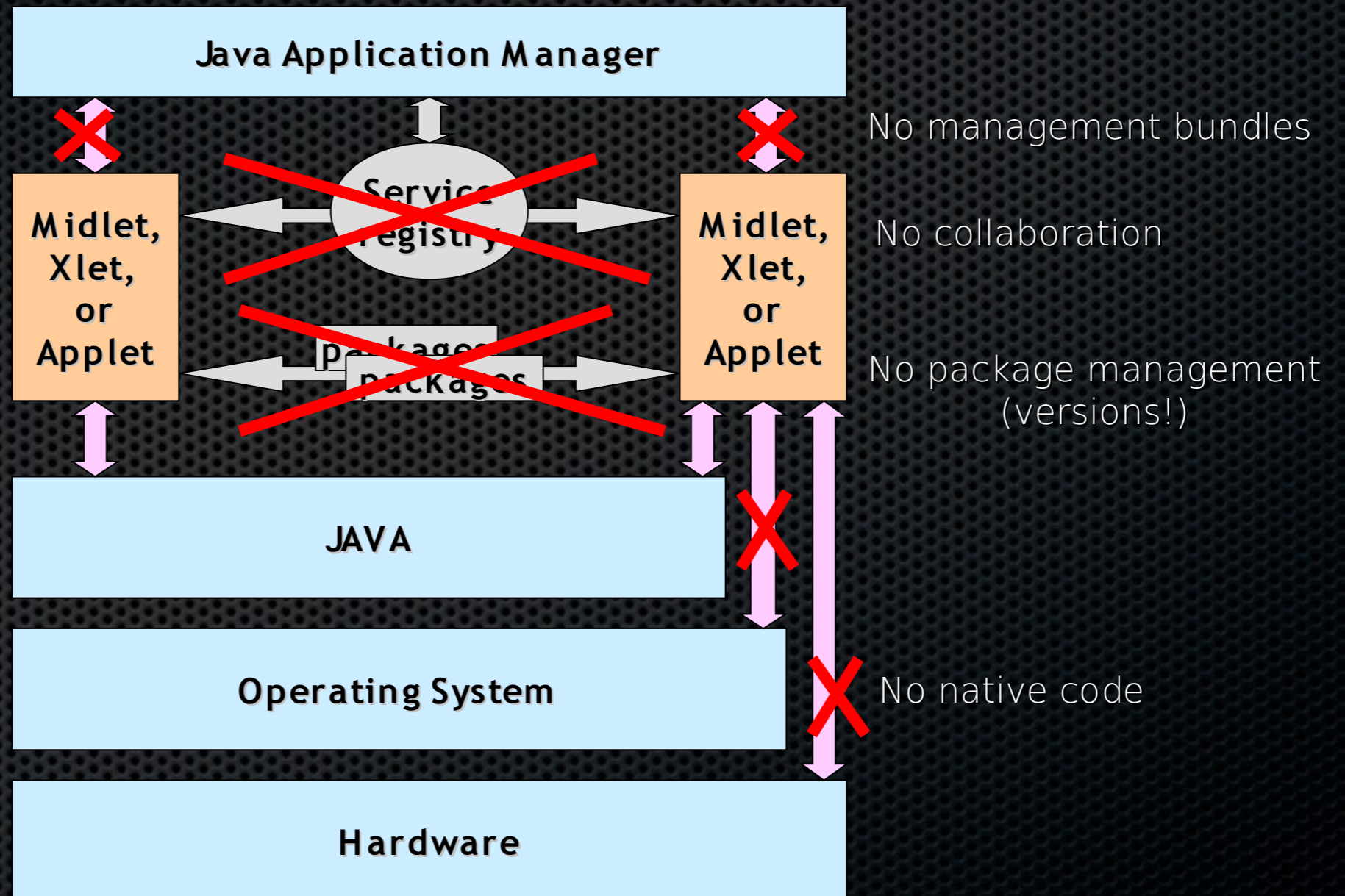
Collaborative model

- *OSGi is more* than an Applet, MIDlet, Xlet runner
- Bundles can collaborate through:
 - *service objects*
 - *package sharing*
- A dynamic registry allows a bundle to find and track service objects
- Framework fully manages this collaboration
 - Dependencies, security

Collaborative model

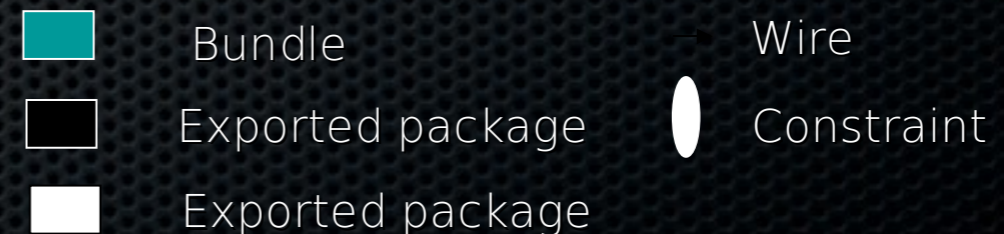
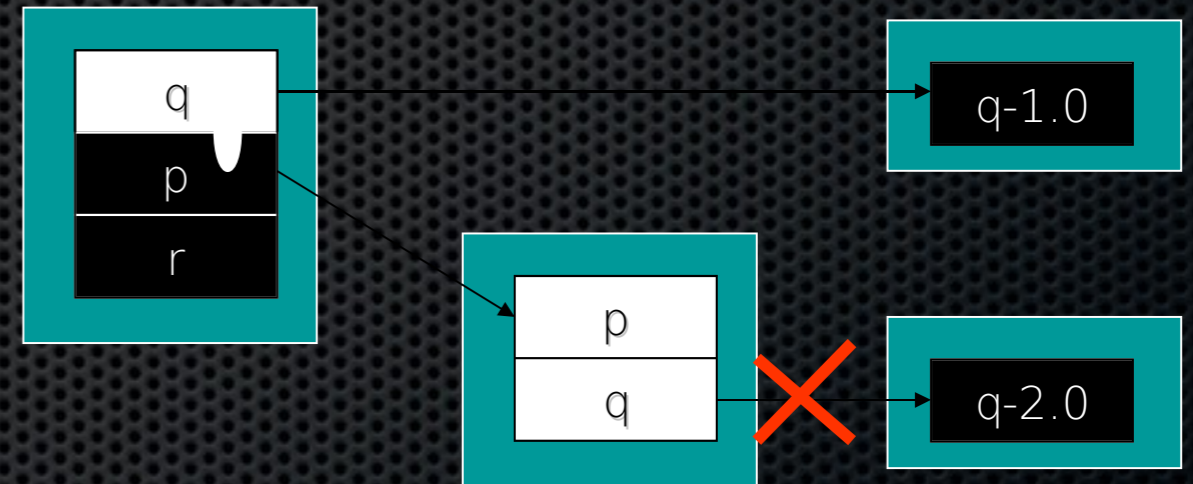


Collaborative model

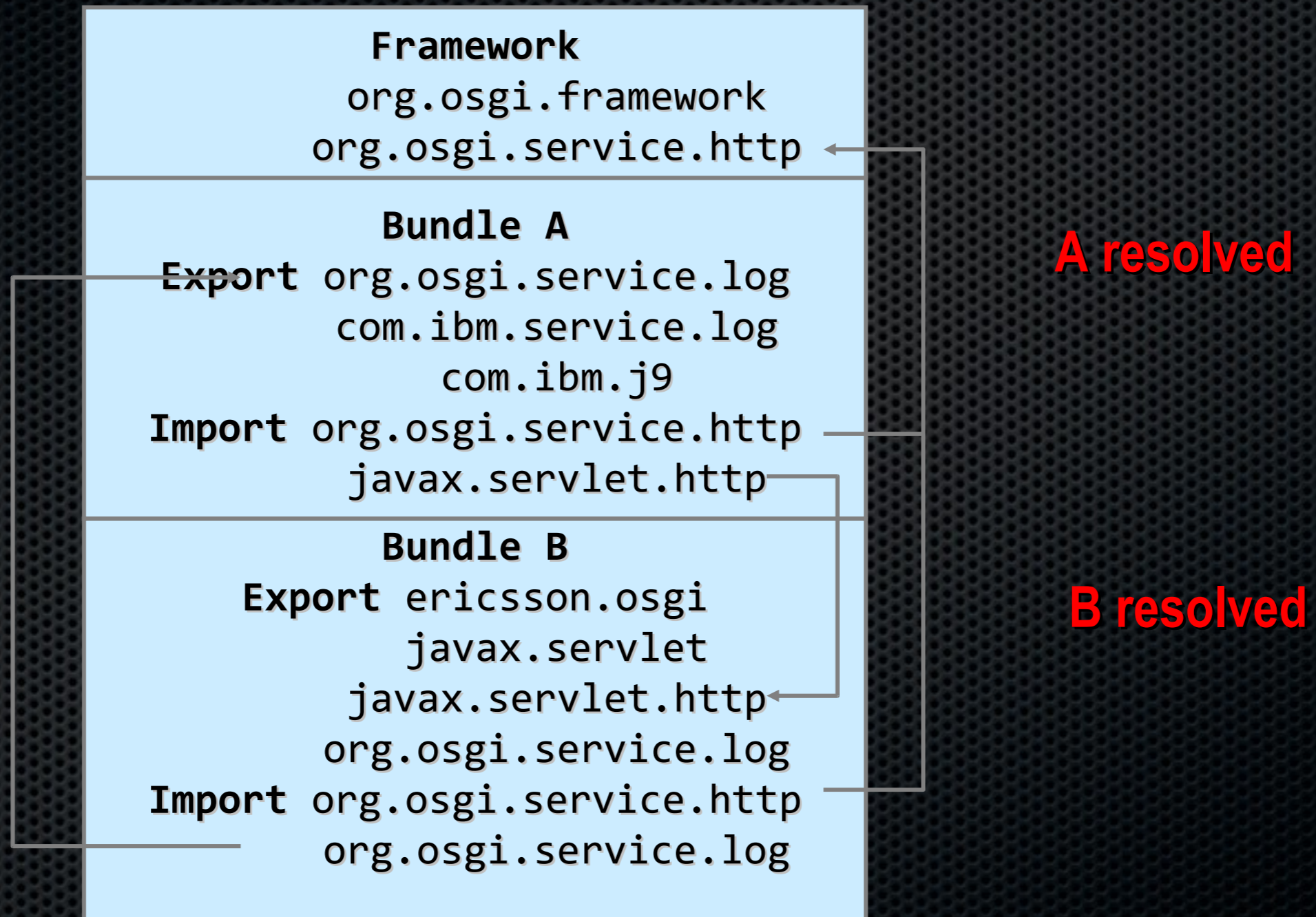


Classpath issues

- Java applications consists of *classes* placed in *packages*
- Java searches for a package or class in different jar files and directories
 - These are usually specified in the CLASSPATH environment variable
- An OSGi Framework is a network of class loaders.
 - Parameterized by the Manifest headers
- Any dependencies between bundles are resolved by the Framework
- It is possible to fetch bundles on demand
- Complicated – But an OSGi Framework makes it painless to use

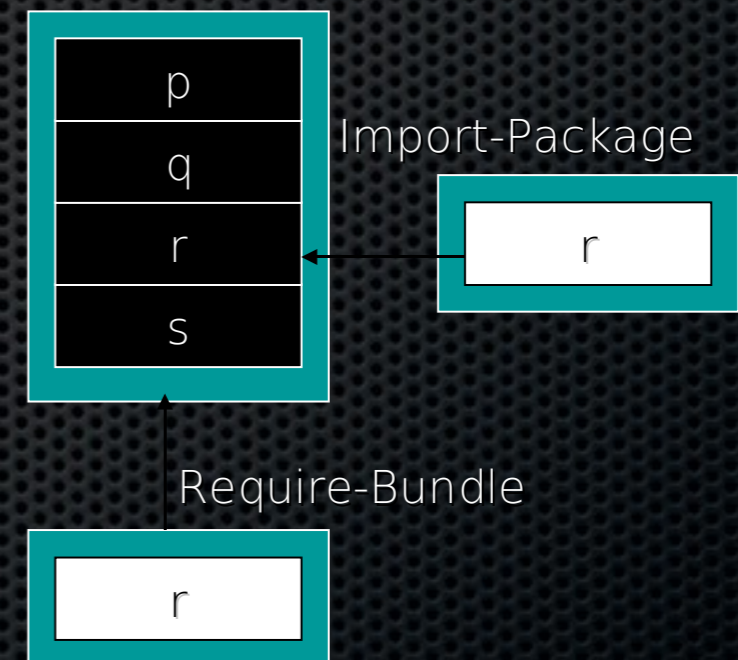


OSGi dependency resolution



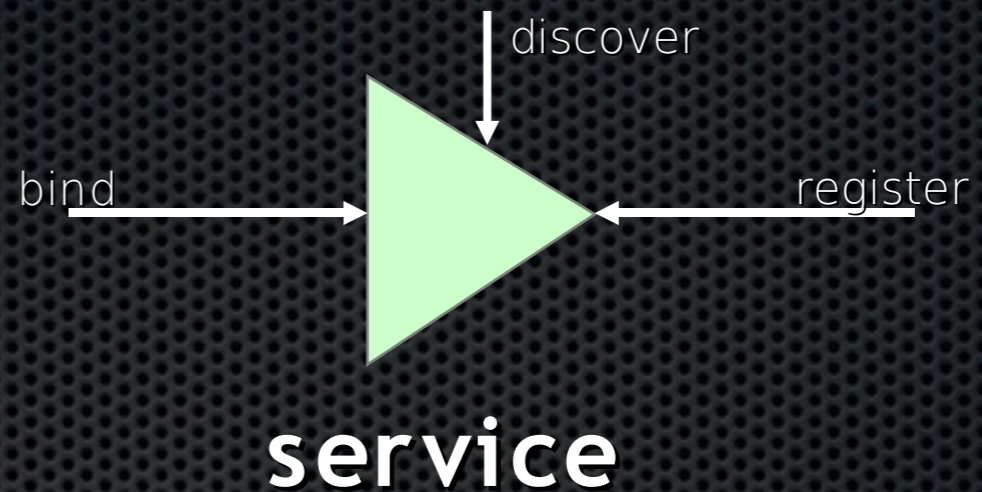
Package or Bundle Dependencies?

- The OSGi Specifications supports both Require-Bundle and Import-Package
- Require-Bundle creates a dependency on a complete bundle
 - Simple to use
 - Imports packages that are not used
- Import-Package creates a dependency on just a package
 - Creates less brittle bundles because of substitutability
 - More cumbersome to use (Tools!)
- In almost all cases, Import-Package is recommended because it eases deployment and version migration
- The specifications detail a number of additional problems with Require-Bundle



Service Specifics

- A *service* is an object registered with the Framework by a bundle to be used by other bundles
- The semantics and syntax of a service are specified in a Java interface
- A bundle can register a service.
- A bundle can use a service (bind to)
 - 1..1
 - 0..1
 - 0..n
- A service can be discovered dynamically
- Services can go away at any time!



```
package org.osgi.service.log;
import org.osgi.framework.ServiceReference;
public interface LogService {
public static final int LOG_ERROR= 1;
public static final int LOG_WARNING= 2;
public static final int LOG_INFO= 3;
public static final int LOG_DEBUG= 4;
public void log(int level,
String message);
public void log(int level,
String message, Throwable exception);
public void log(ServiceReference sr,
int level, String message);
public void log(ServiceReference sr,
int level, String message,
Throwable exception);
}
```

Manipulating Services

- The BundleContext provides the methods to manipulate the service registry
- Services registrations are handled by ServiceRegistration objects

- They can be used to unregister a service or modify its properties

- Service References give access to the service as well as to the service's properties
- Access to service objects is through the getService method. These services should be returned with the ungetService method

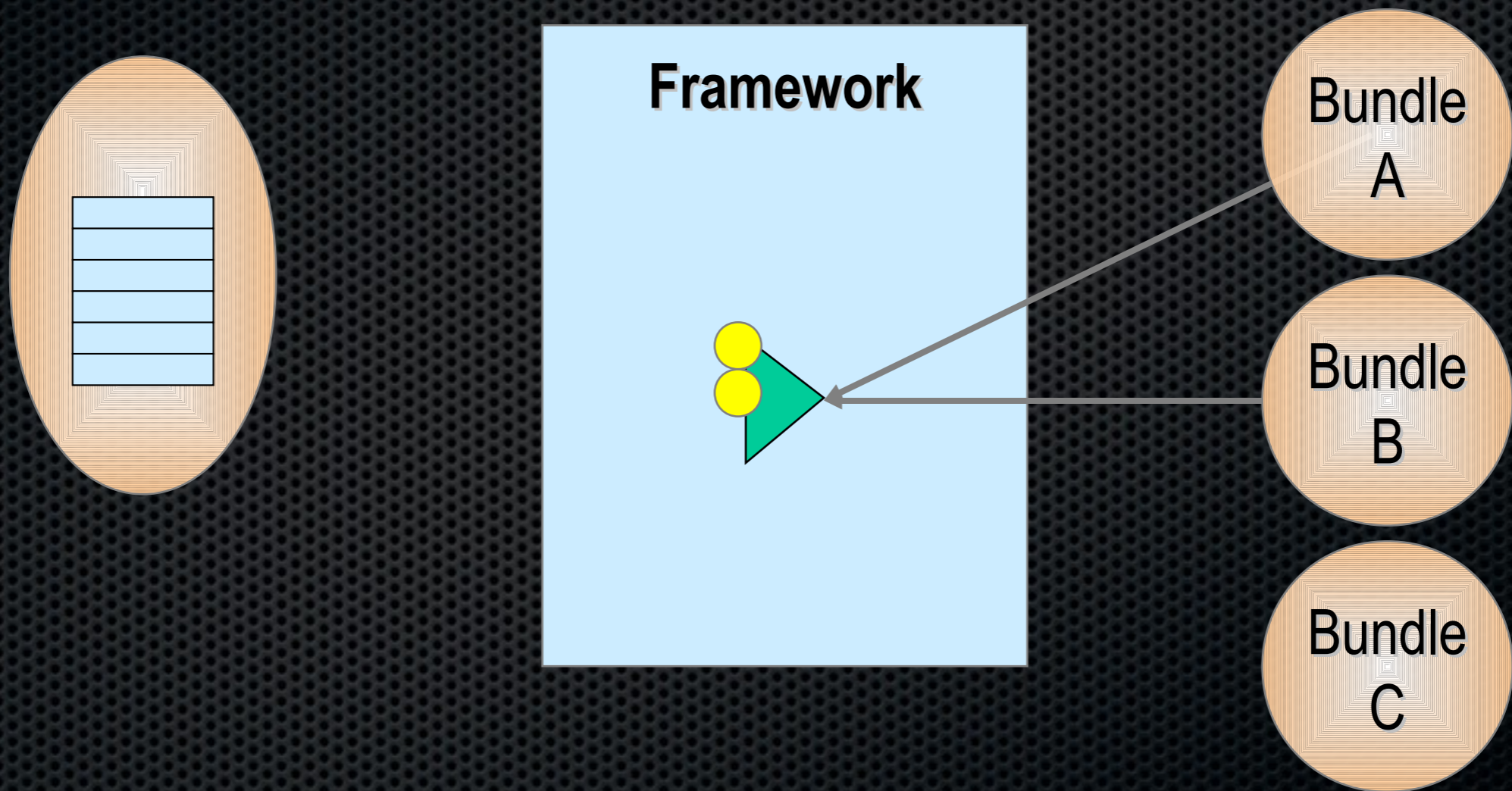
```
ServiceRegistration registerService(  
    String cls,  
    Object srvc,  
    Dictionary prprts)  
  
ServiceReference[]  
    getServiceReferences(  
        String cls,  
        String fltr)  
  
Object getService(  
    ServiceReference reference)  
  
boolean ungetService(  
    ServiceReference rfrnc);
```

Service Tracking

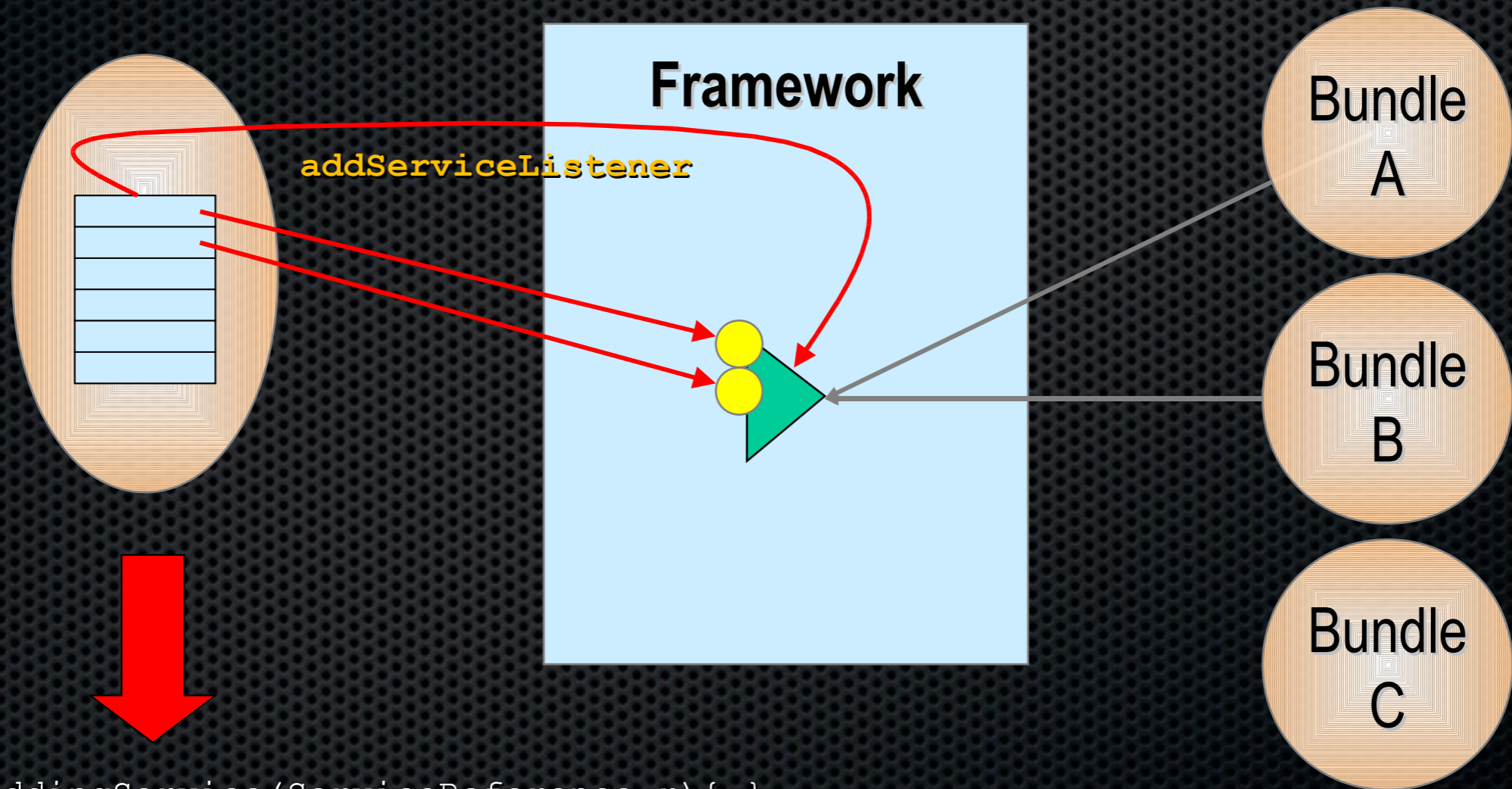
The Case for the ServiceTracker

- Finding services for each message is kind of expensive.
- The ServiceTracker in org.osgi.util.tracker package is intended to simplify this task
- A service tracker maintains a list of services based on:
 - A filter
 - A specific class
- It reports any existing or new services as well as any services that become unregistered
 - **Object addingService**
 - **void modifiedService**
 - **void removedService**
- The service tracker is used to track channels and store them in a Map

ServiceTracker: create

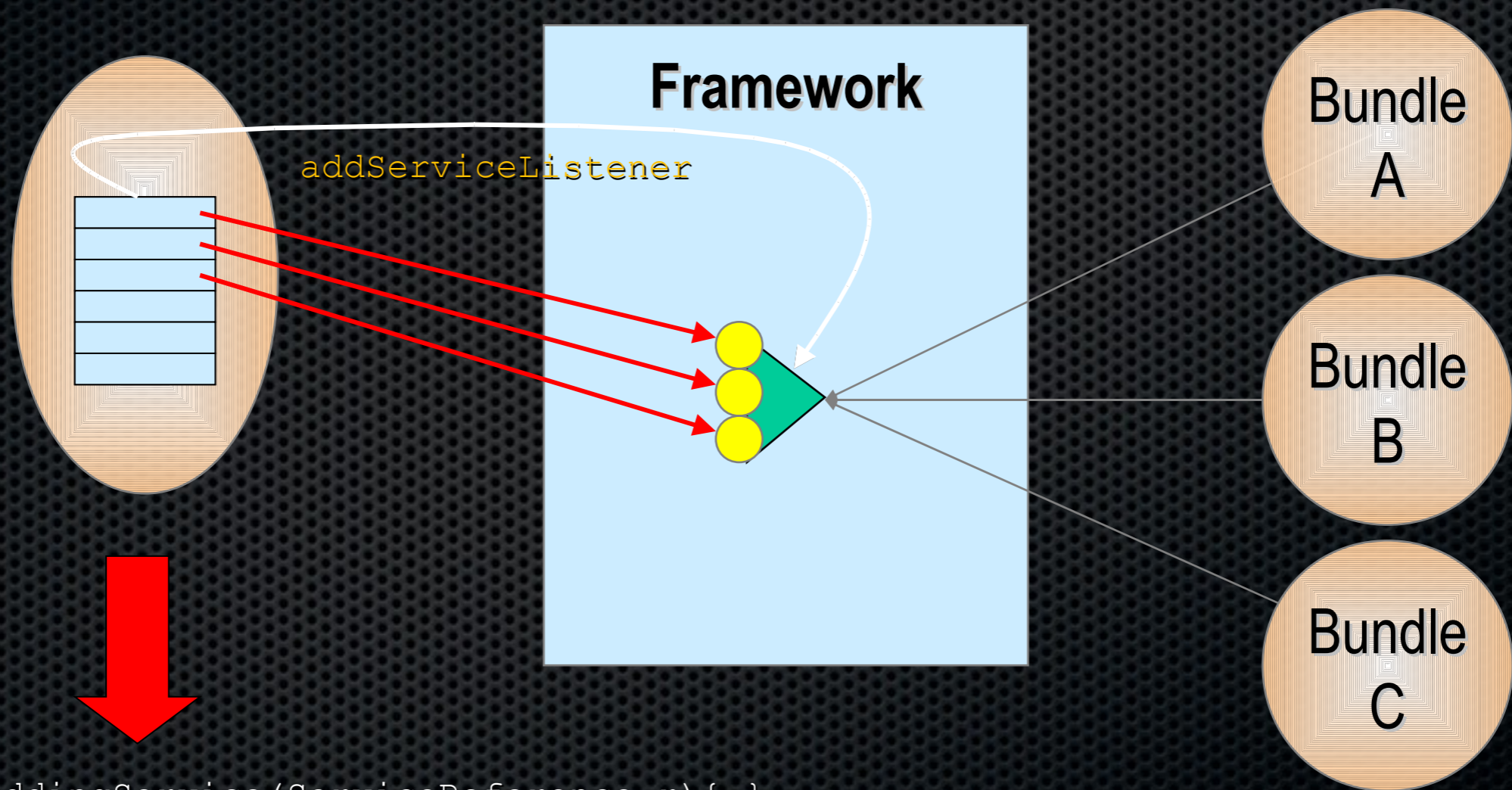


ServiceTracker: open



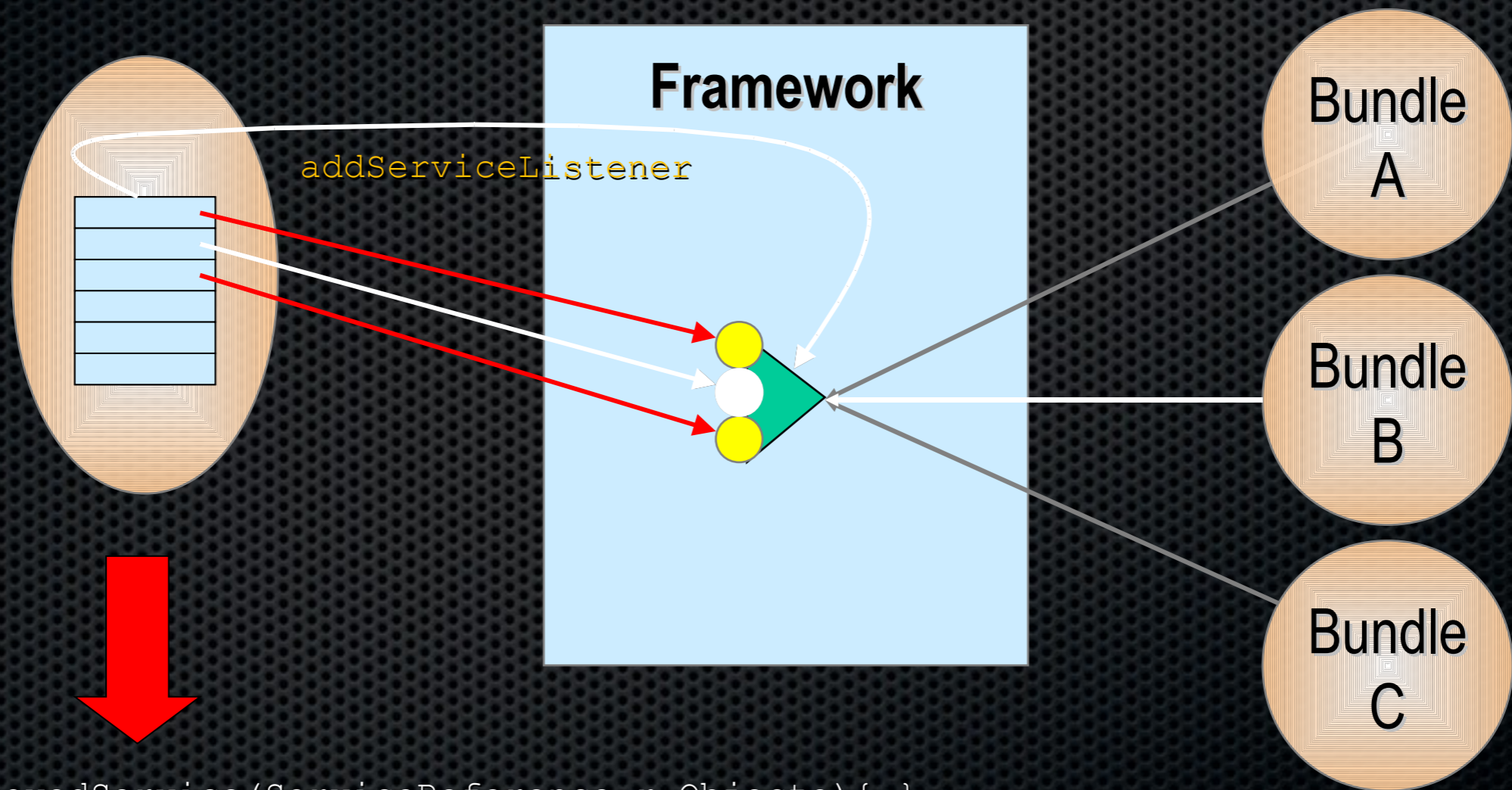
Object addingService(ServiceReference r) {...}

ServiceTracker: adding



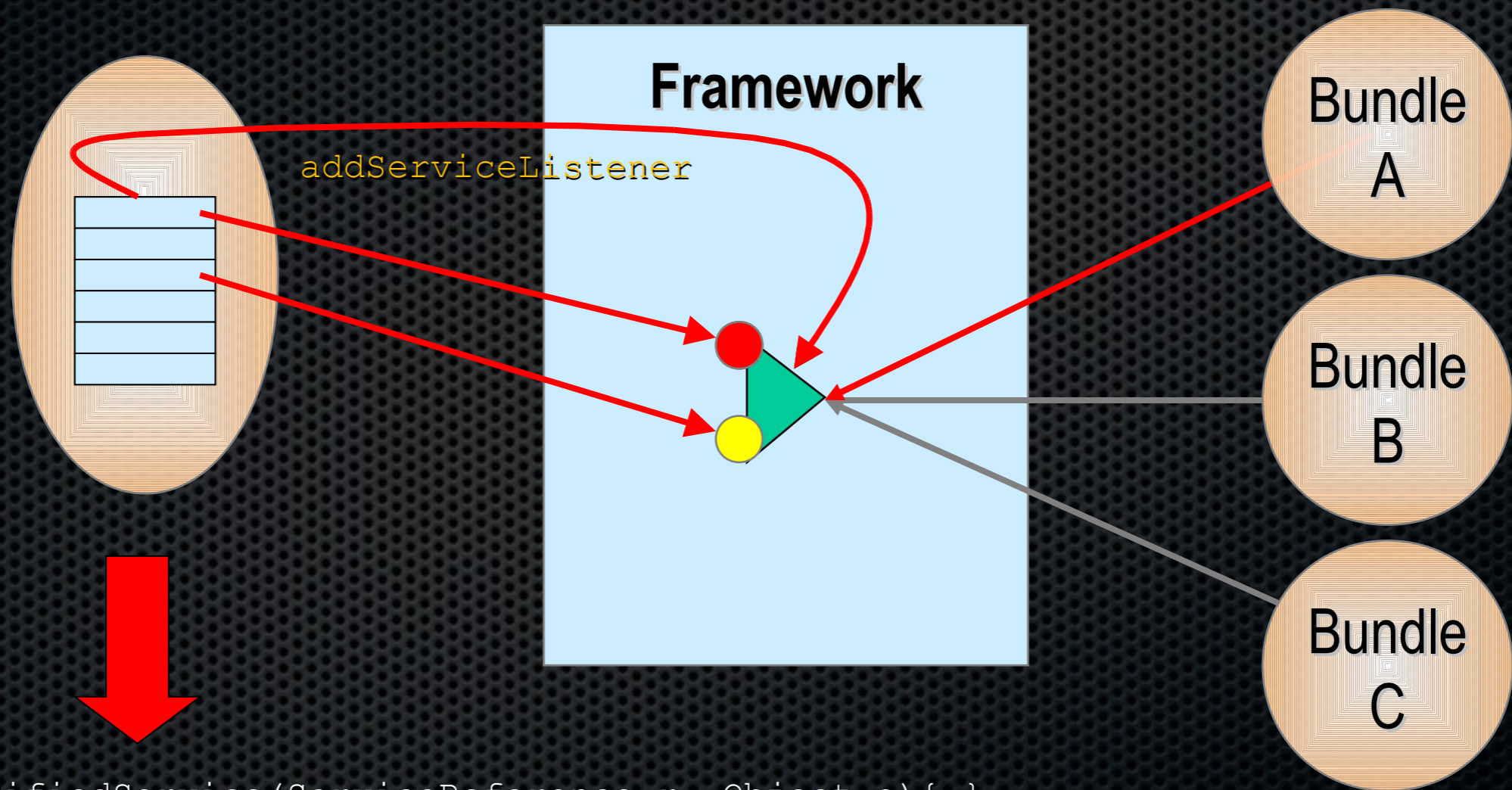
```
Object addingService(ServiceReference r) {...}
```

ServiceTracker: removing



```
void removedService(ServiceReference r, Object o) {...}
```

ServiceTracker: modified



```
void modifiedService(ServiceReference r, Object o){...}
```

Conclusion

- The OSGi R4 Specifications consists of considerable more details than elucidated in this tutorial
- There are many independent OSGi implementations on the market, both commercial and open source
 - Apache Felix, Atinav, Eclipse Equinox, Espial, IBM SMF, Knopflerfish/Ubiserv of Gamespace, ProSyst, ...
- The OSGi specification are today running on mobile phones, PDAs, embedded computers, desktops, mainframes and inside most Enterprise Servers
- Both in managed and unmanaged configurations
- The OSGi specifications solve real world problems
- The OSGi Alliance is working on making the OSGi specifications *the* standard for portable applications. Join us!

Agenda

- ✦ Why is OSGi technology important?
- ✦ What is OSGi technology?
- ✦ OSGi TM Component Programming
- ✦ What changes are coming in the future ?
- ✦ Who is the OSGi Alliance?



Release 4 Version 4.2

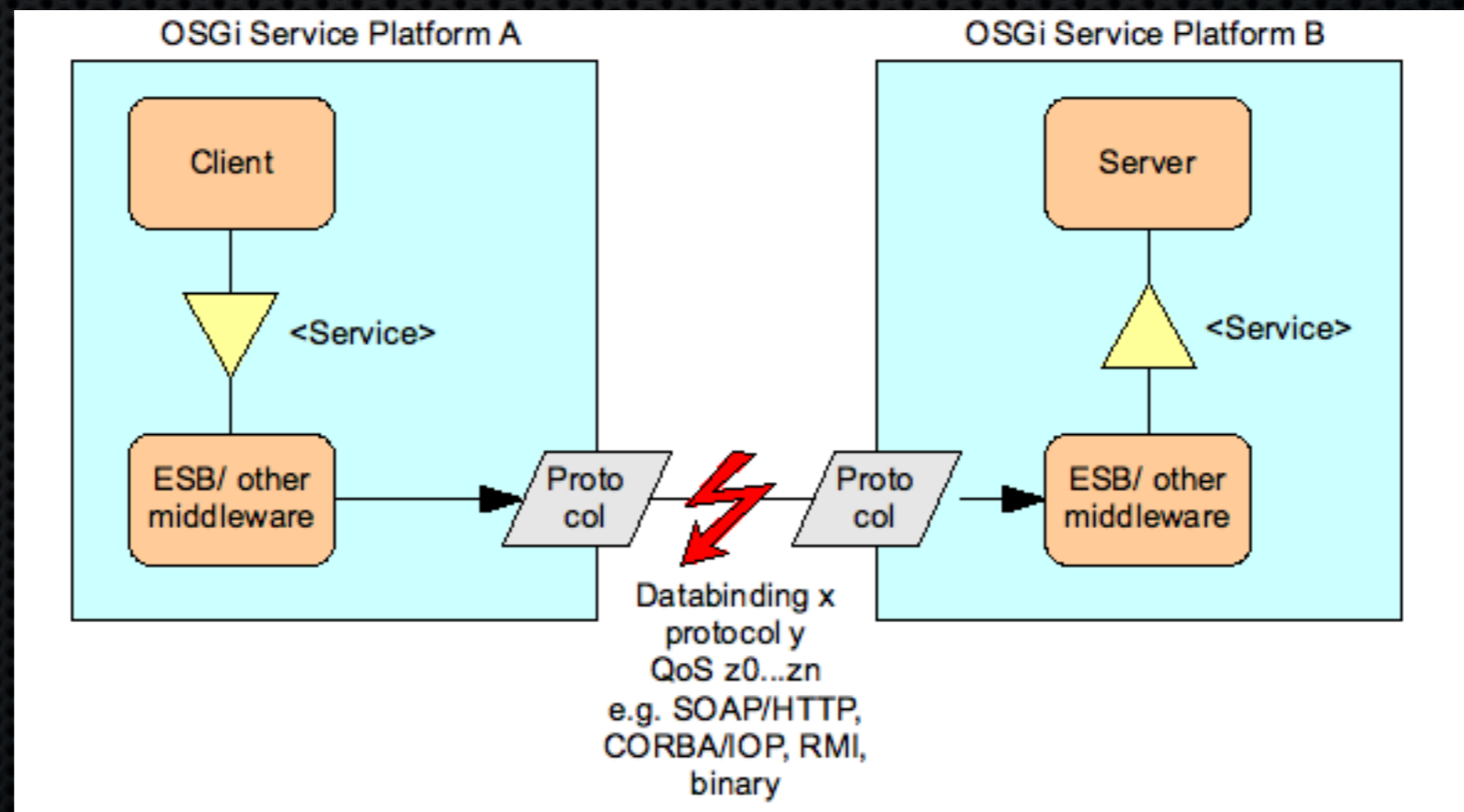
Enterprise Focus

Highlights

- ✦ RFC 119 Distributed OSGi
- ✦ RFC 120 Permission Management Enhancement
- ✦ RFC 121 Bundle Tracker
- ✦ RFC 124 Blueprint (Spring Dynamic Modules)
- ✦ RFC 126 Service Hooks
- ✦ RFC 132 Framework launching
- ✦ RFC 138 Multiple Frameworks in a VM

RFC 119 Distributed OSGi

- Discovering and using services in remote OSGi frameworks
- Expose services for remote use



RFC 120 Permission Management Enhancement

- Allow permission management to be simplified by declaring denied permissions
- ALLOW/DENY model
 - like Apache `httpd`



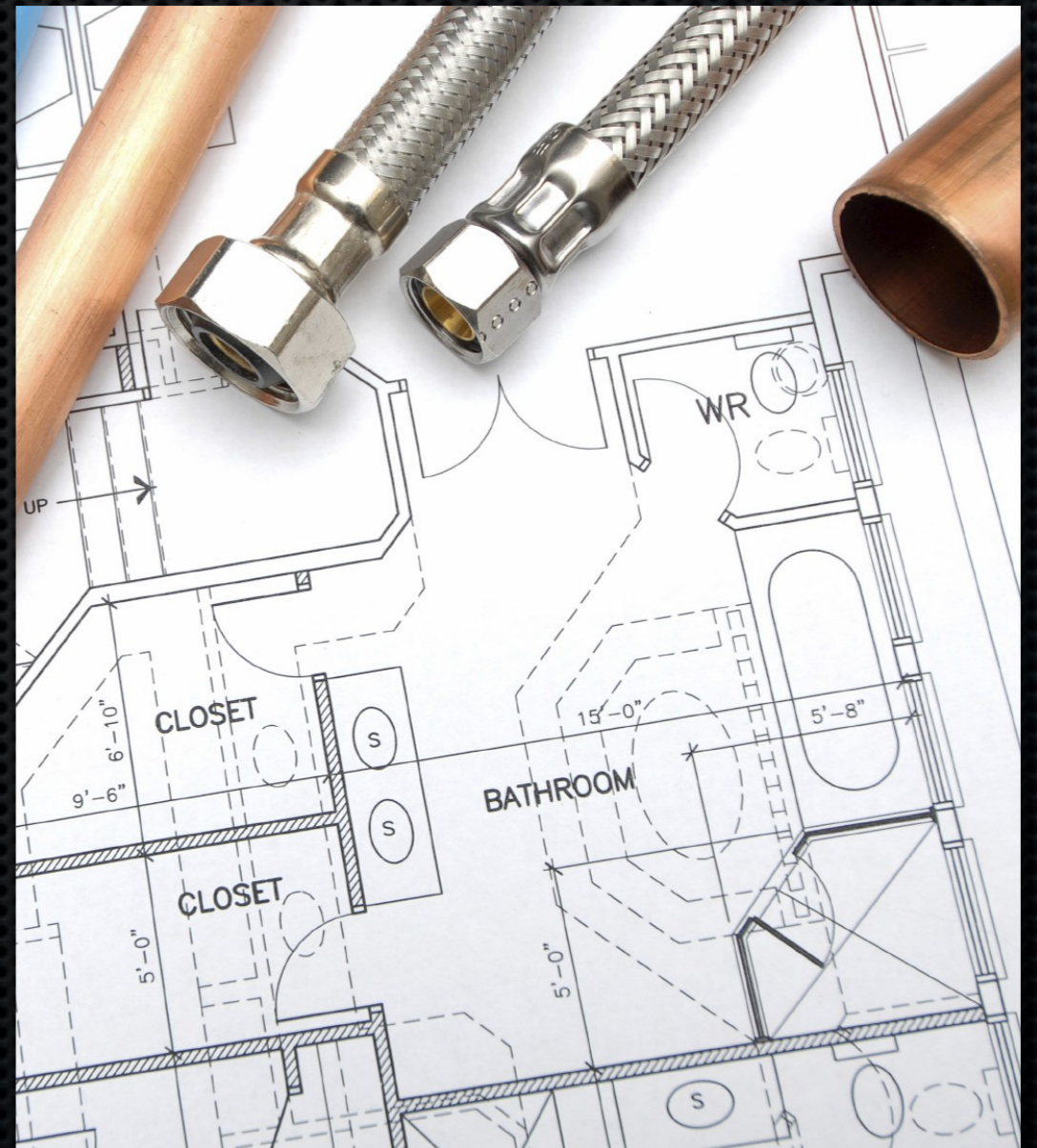
RFC 121 Bundle Tracker

- ✦ Track bundles based upon bundle state
 - ✦ Like ServiceTracker but for bundles
- ✦ For Extender Pattern



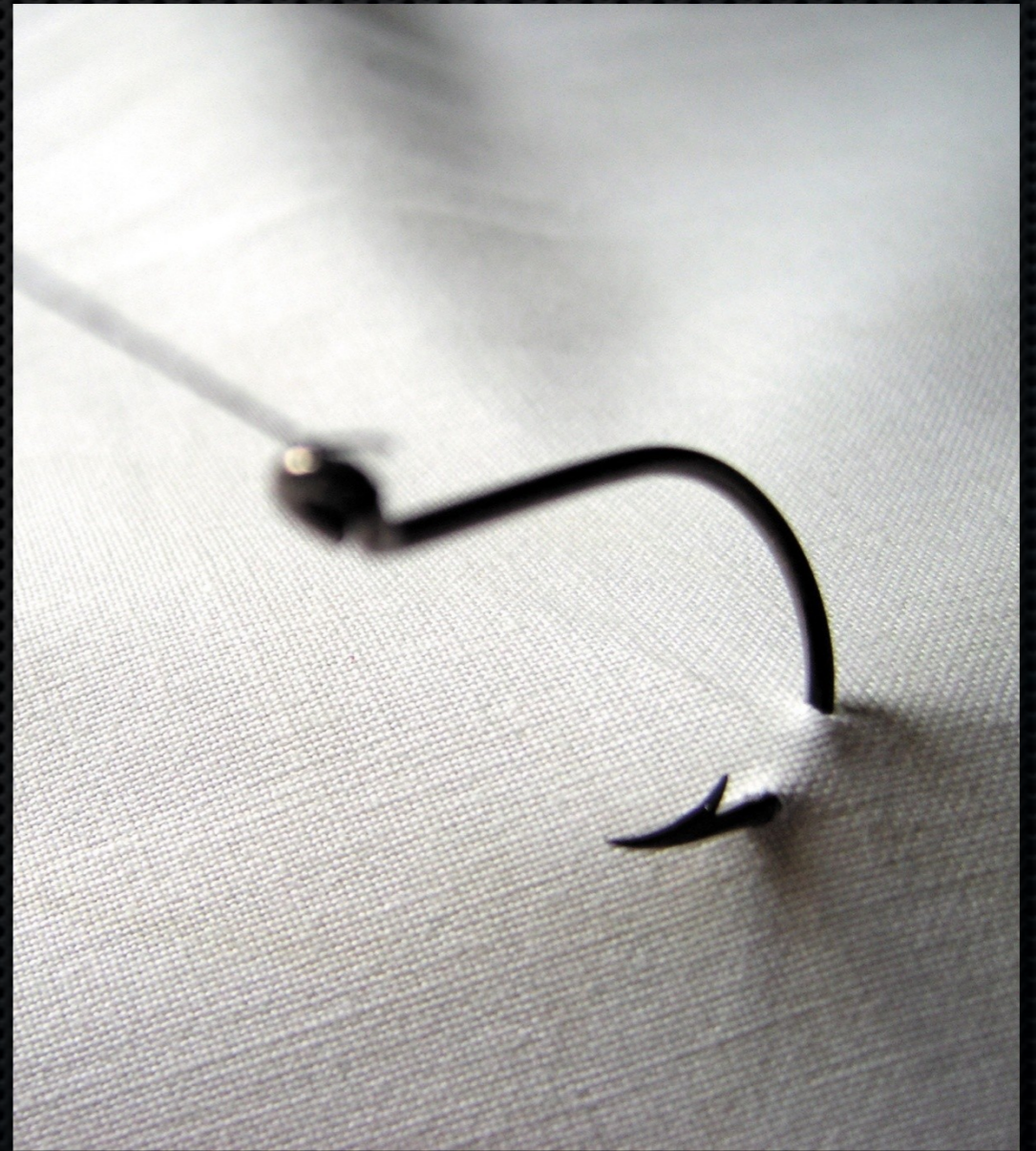
RFC 124 Blueprint

- A richer Dependency Injection(DI) runtime for OSGi
 - from Spring Dynamic Modules
- Complements Declarative Services



RFC 126 Service Hooks

- A new mechanism to observe and react to service operations
 - Publish, Find, Listen
- Can be used to "hide" services from bundles
- Used by RFC 119



RFC 132 Framework Launching

- Create, configure, launch, manage and shutdown an OSGi framework instance



RFC 138 Multiple Frameworks in a VM

- Allow multiple frameworks to function in a single VM
- Peers(heterogeneous), parent/child (homogeneous)
- Scoping for stack products and applications



Beyond 4.2

- ✦ Framework 2.0
 - ✦ Modernize API
 - ✦ JSR 294 support
 - ✦ Support 1.x bundles

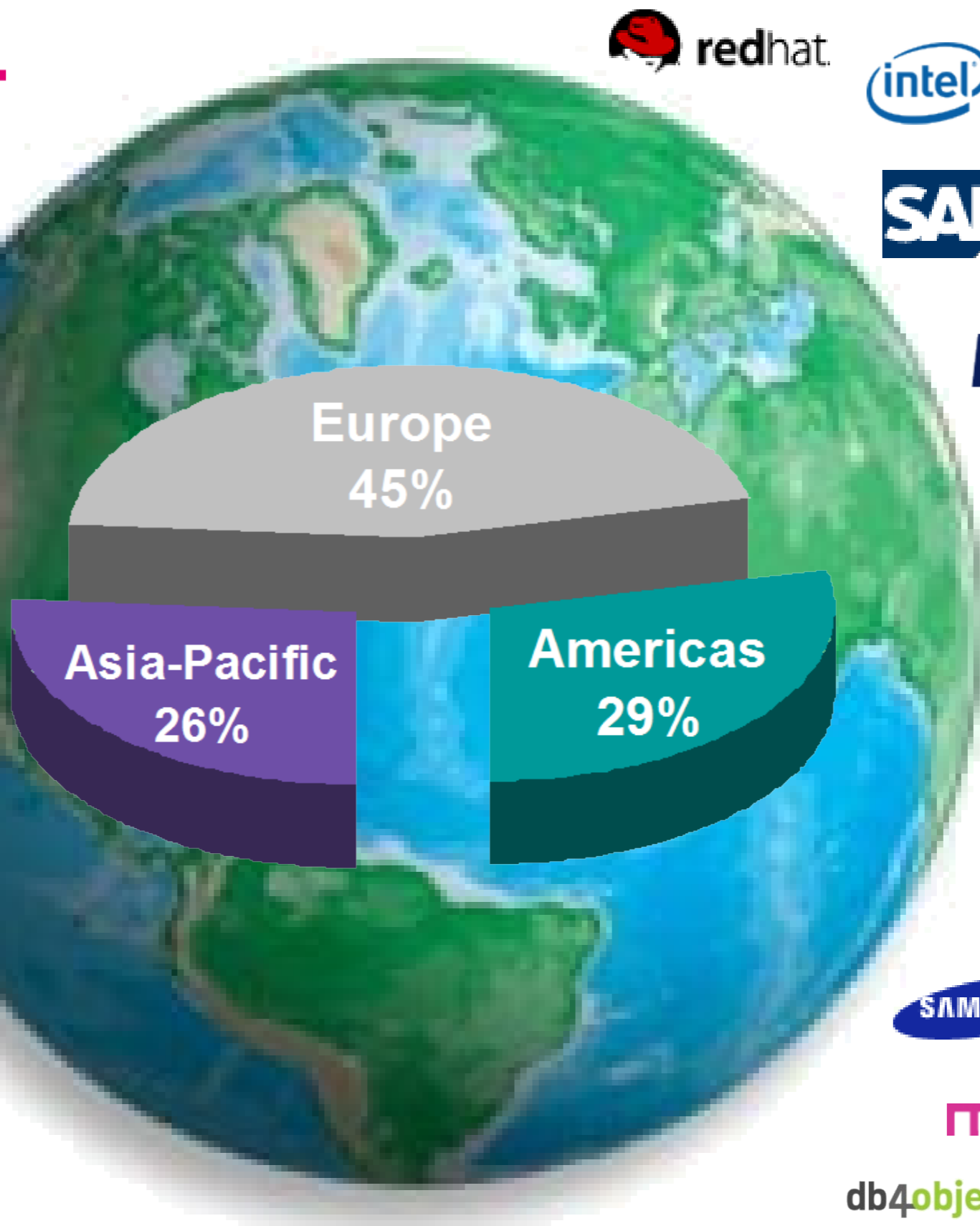


Agenda

- ✦ Why is OSGi technology important?
- ✦ What is OSGi technology?
- ✦ Technical Deep Dive
- ✦ What changes are coming in the future?
- ✦ Who is the OSGi Alliance?

OSGi Alliance Members

Telefonica



Thank you!



