

# OSGi for Eclipse Developers

Chris Aniszczyk (EclipseSource)

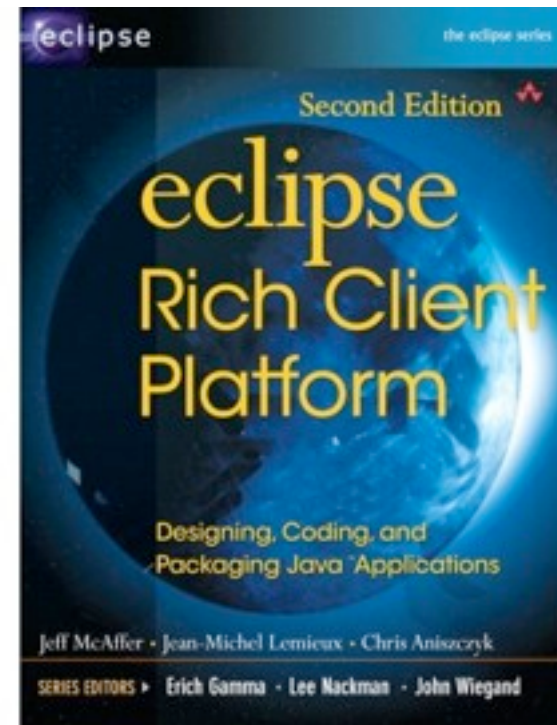
Senior Software Engineer

[zx@eclipsesource.com](mailto:zx@eclipsesource.com)

<http://twitter.com/caniszczyk>

## Who Am I?

- Senior Software Engineer at EclipseSource
- PDE Co-Lead
- Board of Directors @ Eclipse Foundation
- Eclipse {Architecture, Planning} Council
- Co-Author, Eclipse RCP Book 2nd Edition



## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - Versioning
  - Extensions and Services
  - Compendium Services
  - OSGi Tooling
- Conclusion

## OSGi...

- The dynamic module system for java
- What's a module?

*“**Modular programming** is a software design technique that increases the extent to which software is composed from separate parts, called modules. Conceptually, modules represent a separation of concerns, and improve maintainability by enforcing logical boundaries between components.”*

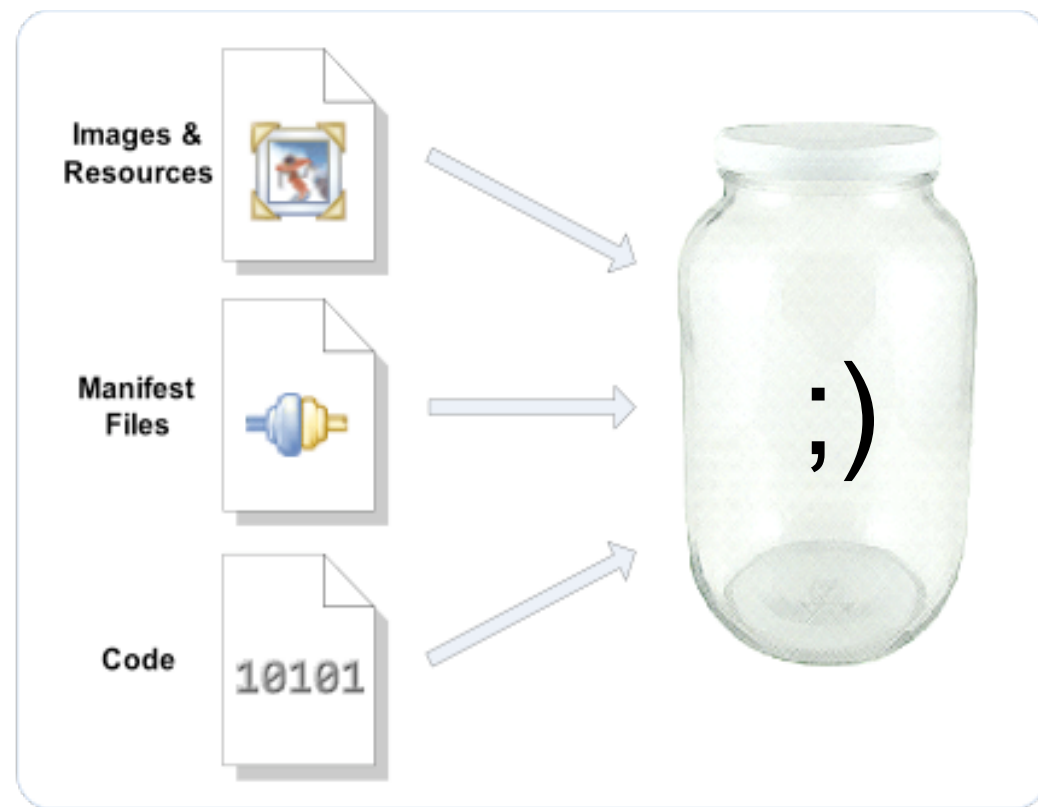
## JARs != Modules

- JARs are deployment artifacts
- JARs have dependencies
- They are not modules... missing crucial information
  - ◆ identifier (file name isn't good enough)
  - ◆ version
  - ◆ vendor
  - ◆ exports
  - ◆ dependencies



## How does OSGi help?

- OSGi Bundle == Module
- Just a JAR with module related metadata
  - ◆ identifier
  - ◆ version
  - ◆ vendor
  - ◆ exports
  - ◆ dependencies (imports)



# MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ClassPath: junit.jar
Bundle-Vendor: Eclipse.org
Bundle-Localization: plugin
Bundle-RequiredExecutionEnvironment: J2SE-1.3
Bundle-Name: JUnit3
Bundle-SymbolicName: org.junit
Export-Package: junit.awtui;version="3.8.2",junit.extensions;version="3.8.2",junit.framework;version="3.8.2",junit.runner;version="3.8.2",junit.swingui;version="3.8.2",junit.swingui.icons;version="3.8.2",junit.textui;version="3.8.2"
Bundle-Version: 3.8.2.v20090203-1005
Bundle-ManifestVersion: 2
```



## OSGi in Details

- OSGi Alliance
  - Worldwide consortium of technology innovators that advances OSGi technology
- OSGi Technology
  - Set of *specifications* that define a **module** system for Java
  - Enables modular programming for Java
- Originally designed for embedded systems...
  - Home automation... set-top boxes... vehicles...
- Now in widespread use in desktop and servers...

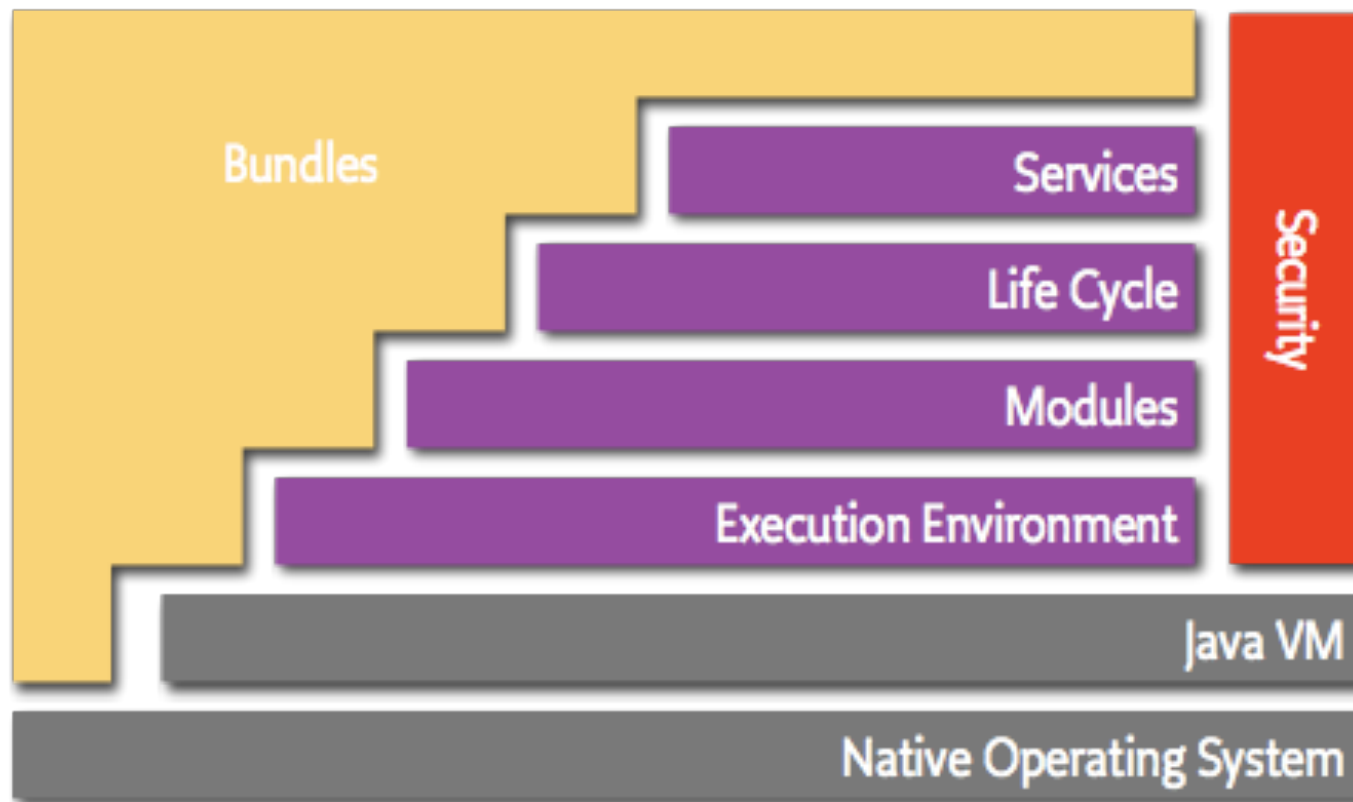


## OSGi Alliance

- An open standards organization founded in March 1999
  - ♦ Founded by Ericsson, IBM, Motorola and Sun Microsystems
- Five expert groups that produce specifications
  - ♦ Core Platform
  - ♦ Enterprise
  - ♦ Mobile
  - ♦ Vehicle
  - ♦ Residential
- Now ~100 members
  - ♦ IBM, Oracle, Tibco, Siemens, SAP, Motorola, Red Hat, Sun...

## OSGi Layers

- OSGi has a layered model...



# OSGi Technology

- Bundles
  - ♦ OSGi modules made by the developers
- Services
  - ♦ Connects bundles in a dynamic way by offering a publish-find-bind model
- Life Cycle
  - ♦ API to install, start, stop, update and uninstall bundles
- Modules
  - ♦ Defines how a bundle can import and export code
- Security
  - ♦ Handles the security aspects, think permissions
- Execution Environment
  - ♦ Defines what methods and classes are available in a specific platform

## OSGi Specifications

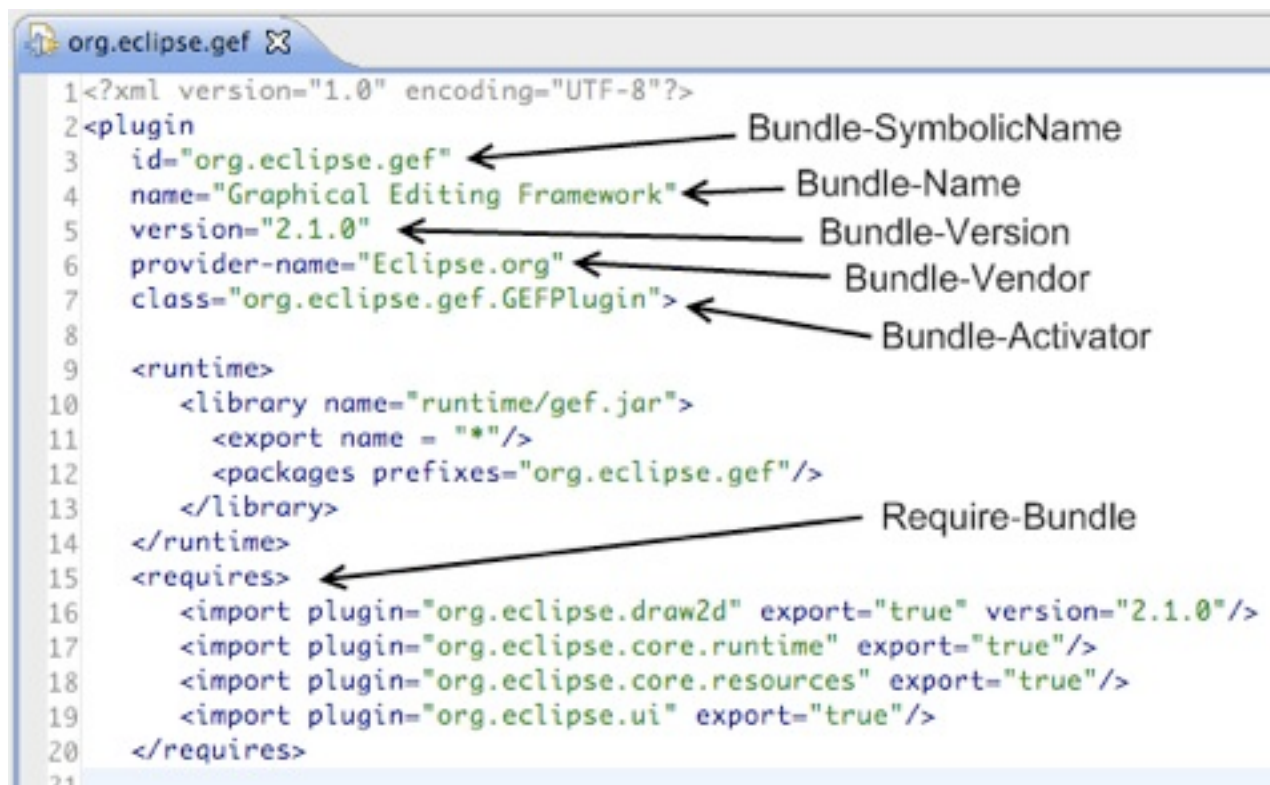
- Two major specifications
- Core Specifications
  - ◆ Covers the core layers of OSGi
- Service Compendium
  - ◆ Contains a variety of useful services

## OSGi Specification Releases

- OSGi Release 1 (R1): May 2000
- OSGi Release 2 (R2): October 2001
- OSGi Release 3 (R3): March 2003
- OSGi Release 4 (R4): October 2005
  - ♦ Core Specification (R4 Core): October 2005
  - ♦ Mobile Specification (R4 Mobile / JSR-232): September 2006
- OSGi Release 4.1 (R4.1): May 2007 (AKA JSR-291)
- OSGi Release 4.1 (R4.2): (Summer 2009)
- OSGi EEG spec coming in 4Q09

## How does this relate to Eclipse?

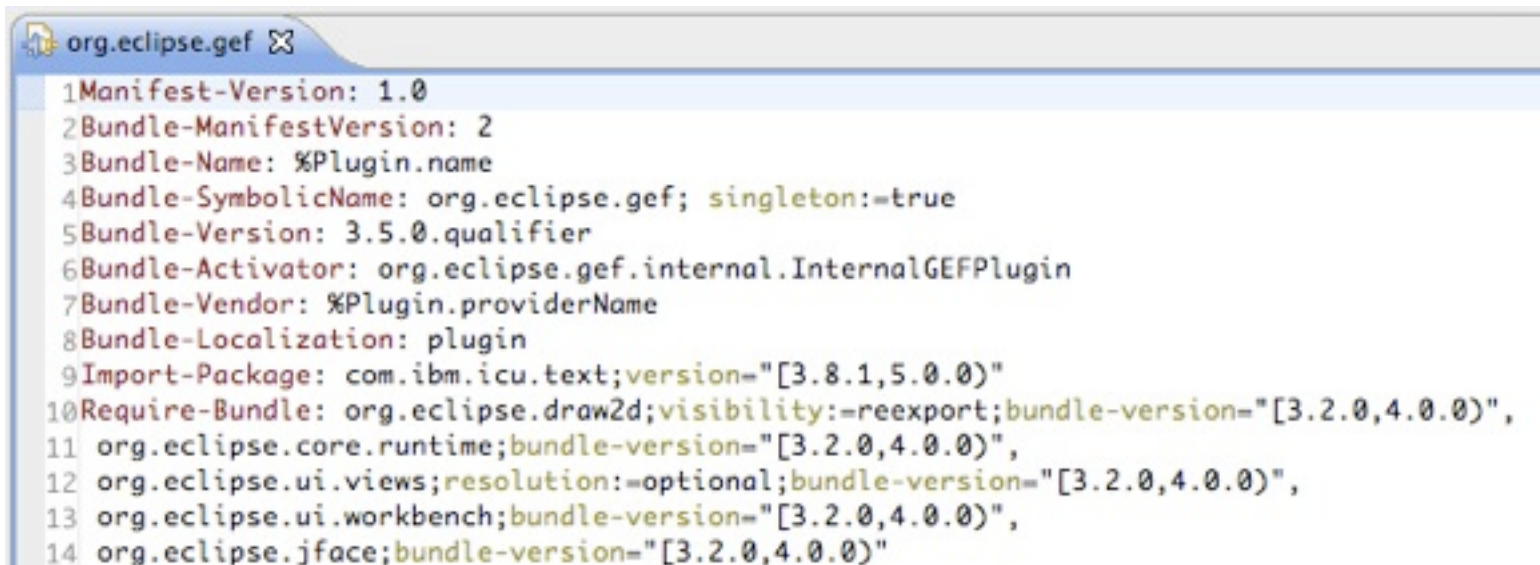
- Eclipse had its own non-standard plug-in model
- OSGi and old Eclipse plug-in model were similar



```
1<?xml version="1.0" encoding="UTF-8"?>
2<plugin
3  id="org.eclipse.gef" ← Bundle-SymbolicName
4  name="Graphical Editing Framework" ← Bundle-Name
5  version="2.1.0" ← Bundle-Version
6  provider-name="Eclipse.org" ← Bundle-Vendor
7  class="org.eclipse.gef.GEFPlugin" ← Bundle-Activator
8
9  <runtime>
10    <library name="runtime/gef.jar">
11      <export name = "*" />
12      <packages prefixes="org.eclipse.gef" />
13    </library>
14  </runtime>
15  <requires> ← Require-Bundle
16    <import plugin="org.eclipse.draw2d" export="true" version="2.1.0" />
17    <import plugin="org.eclipse.core.runtime" export="true" />
18    <import plugin="org.eclipse.core.resources" export="true" />
19    <import plugin="org.eclipse.ui" export="true" />
20  </requires>
21
```

## What happened?

- The world didn't need two modular systems
- Eclipse went to OSGi in 3.0 with Equinox\*
  - Eclipse needed something robust and standard
  - Put OSGi on the map!



```
1Manifest-Version: 1.0
2Bundle-ManifestVersion: 2
3Bundle-Name: %Plugin.name
4Bundle-SymbolicName: org.eclipse.gef; singleton:=true
5Bundle-Version: 3.5.0.qualifier
6Bundle-Activator: org.eclipse.gef.internal.InternalGEFPlugin
7Bundle-Vendor: %Plugin.providerName
8Bundle-Localization: plugin
9Import-Package: com.ibm.icu.text;version="[3.8.1,5.0.0)"
10Require-Bundle: org.eclipse.draw2d;visibility:=reexport;bundle-version="[3.2.0,4.0.0)",
11 org.eclipse.core.runtime;bundle-version="[3.2.0,4.0.0)",
12 org.eclipse.ui.views;resolution:=optional;bundle-version="[3.2.0,4.0.0)",
13 org.eclipse.ui.workbench;bundle-version="[3.2.0,4.0.0)",
14 org.eclipse.jface;bundle-version="[3.2.0,4.0.0)"
```

*\*<http://portal.acm.org/citation.cfm?id=1086616>*

OSGi Service Platform  
Core Specification  
The OSGi Alliance

Release 4  
August 2005







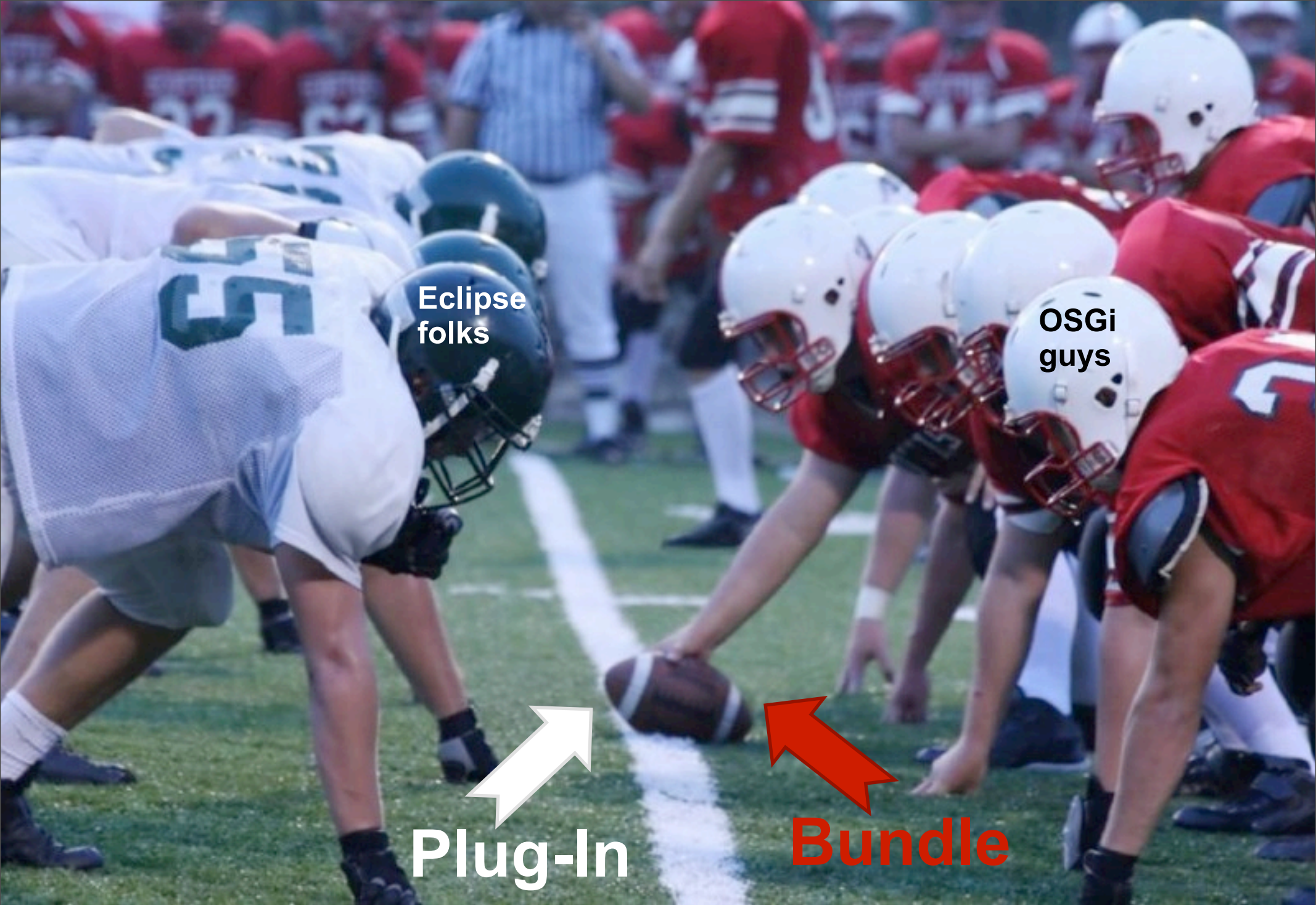




Eclipse  
folks

Plug-In





Eclipse  
folks

OSGi  
guys

Plug-In


Bundle

## Overview

- Introduction
- Topics
  - **Frameworks**
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - Versioning
  - Extensions and Services
  - Compendium Services
  - OSGi Tooling
- Conclusion

## Frameworks

- There's a world outside of Eclipse and Equinox

Gatespace  
TelematicsSAMSUNGSIEMENSespialIBMProSystHITACHIAPACHE  
felixeclipsemakewaveOSGi<sup>™</sup>  
Alliance



## Framework Implementations

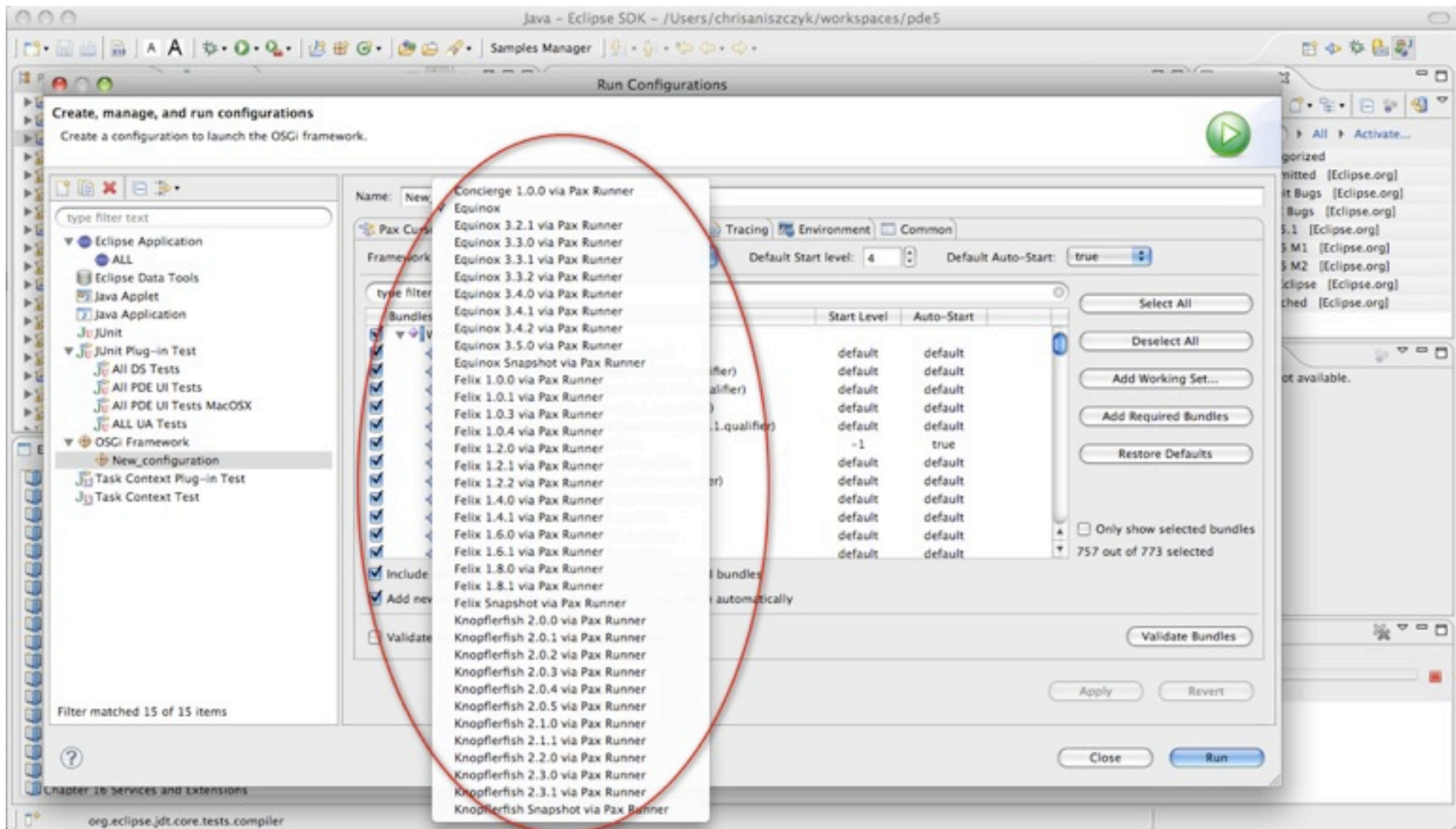
- Equinox (open source)
  - ♦ Reference implementation for the core framework and various services
  - ♦ Base runtime for all of Eclipse (rich client, server side and embedded)
- Felix (open source)
  - ♦ Implementation developed at Apache
  - ♦ Ships with GlassFish
- Knopflerfish (open source)
  - ♦ BSD license
- Concierge (open source)
  - ♦ Highly optimized and tiny R3 implementation
  - ♦ Runs in tiny devices
- Many other commercial and private implementations



## Be aware of special headers...

- Forget about:
  - Eclipse-BuddyPolicy
  - Eclipse-PatchFragment
  - Eclipse-SourceBundle
  - Eclipse-...

➔ Otherwise you are tied to Equinox
- Tip: PAX Runner to test against multiple frameworks
  - <http://wiki.ops4j.org/display/ops4j/Pax+Runner>



## Overview

- Introduction
- Topics
  - Frameworks
  - **Import-Package vs. Require-Bundle**
  - Dynamic Bundles
  - Versioning
  - Extensions and Services
  - Compendium Services
  - OSGi Tooling
- Conclusion

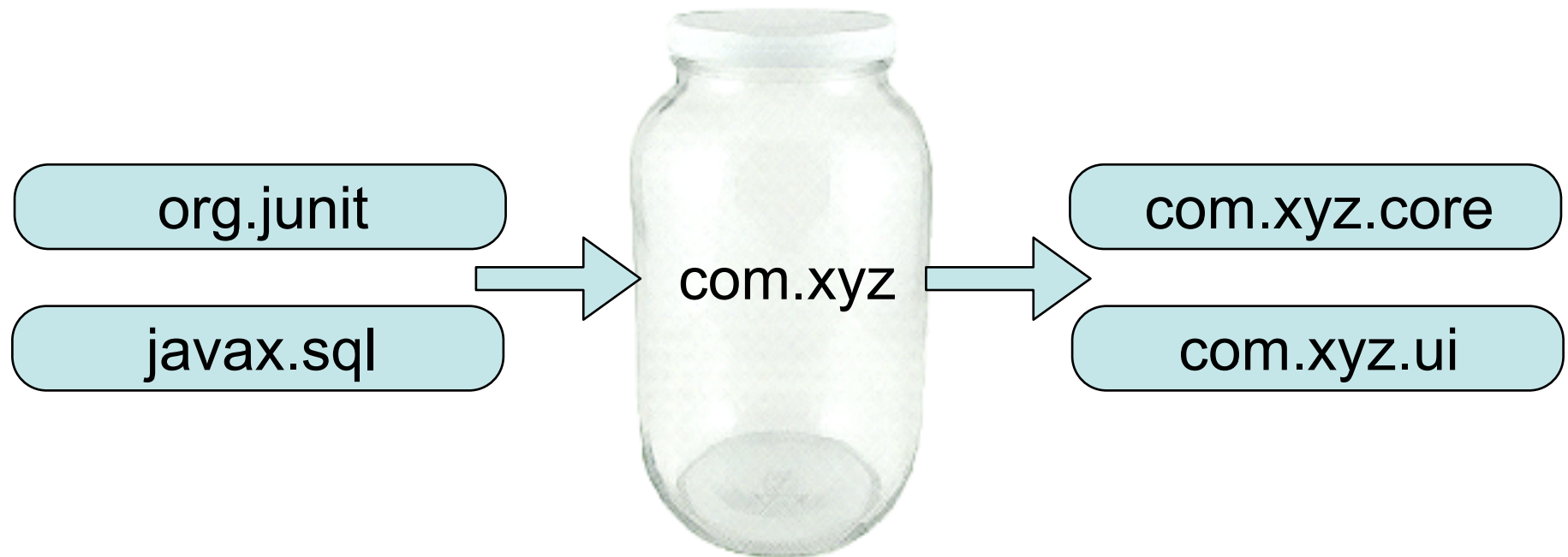


## Dependency Management

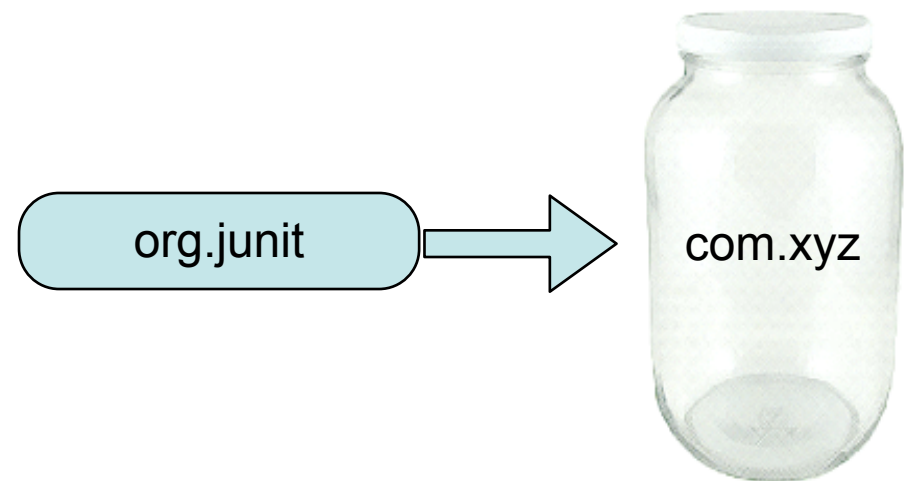
- Eclipse
  - Dependencies are *traditionally* using Require-Bundle
  - Never heard of Import-Package, sounds strange
- OSGi
  - Please don't use Require-Bundle **at all**
  - Instead, define dependencies using Import-Package



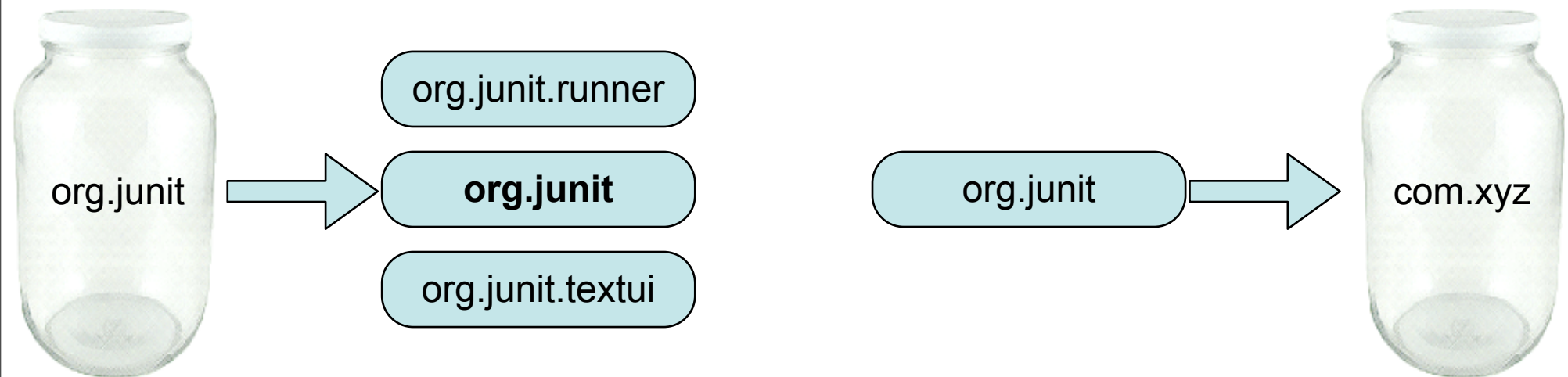
## Imports and Exports



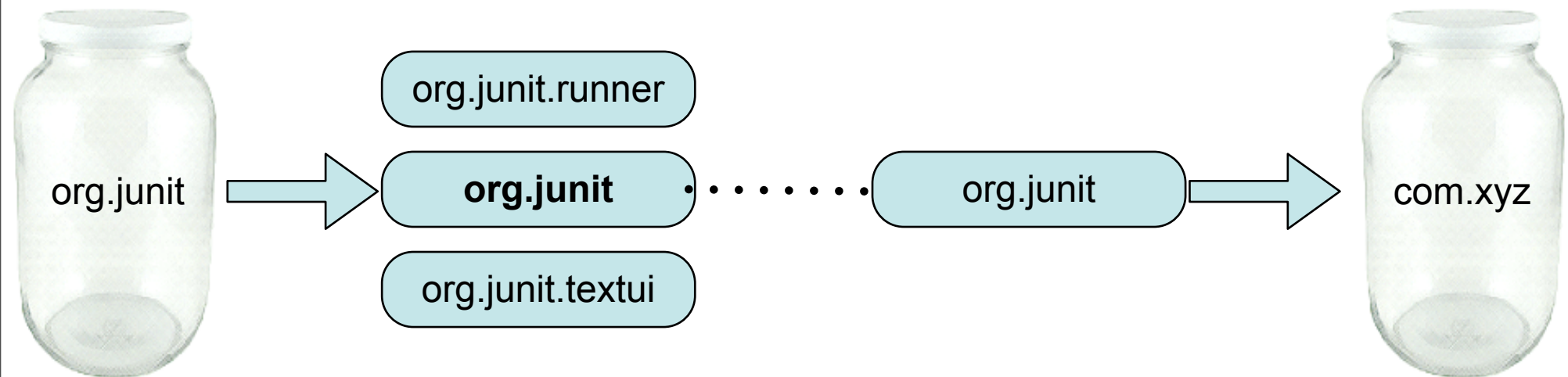
## Package Resolution



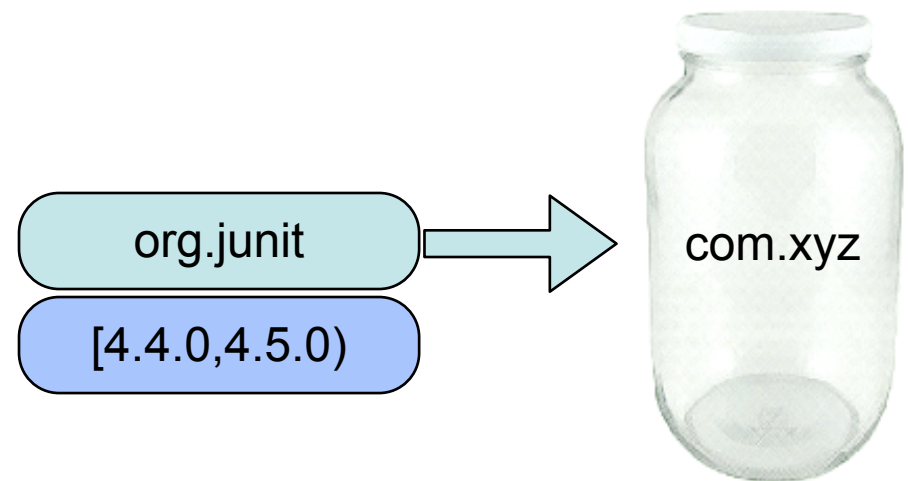
## Package Resolution



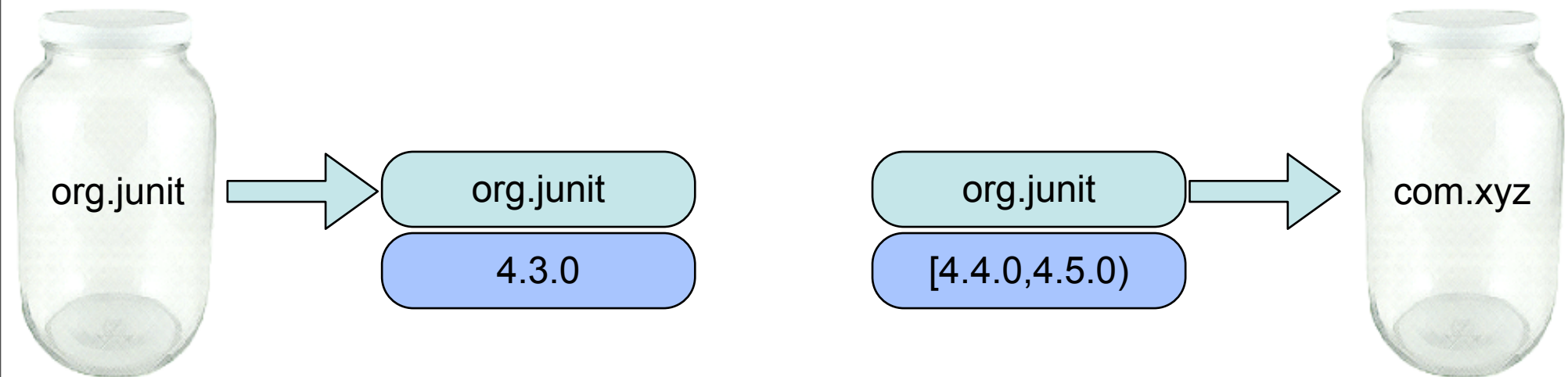
## Package Resolution



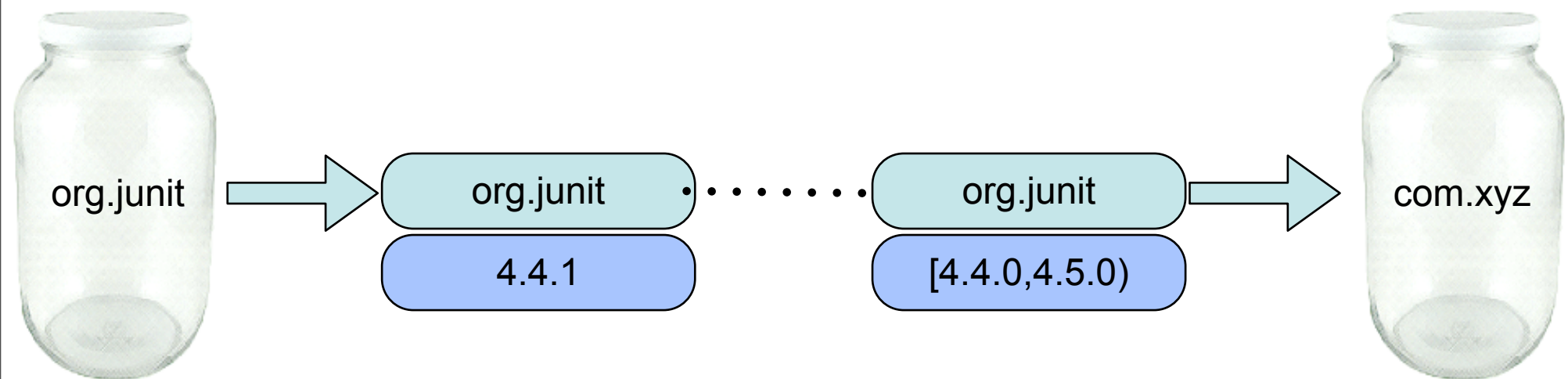
## Versioned Package Resolution



## Versioned Package Resolution

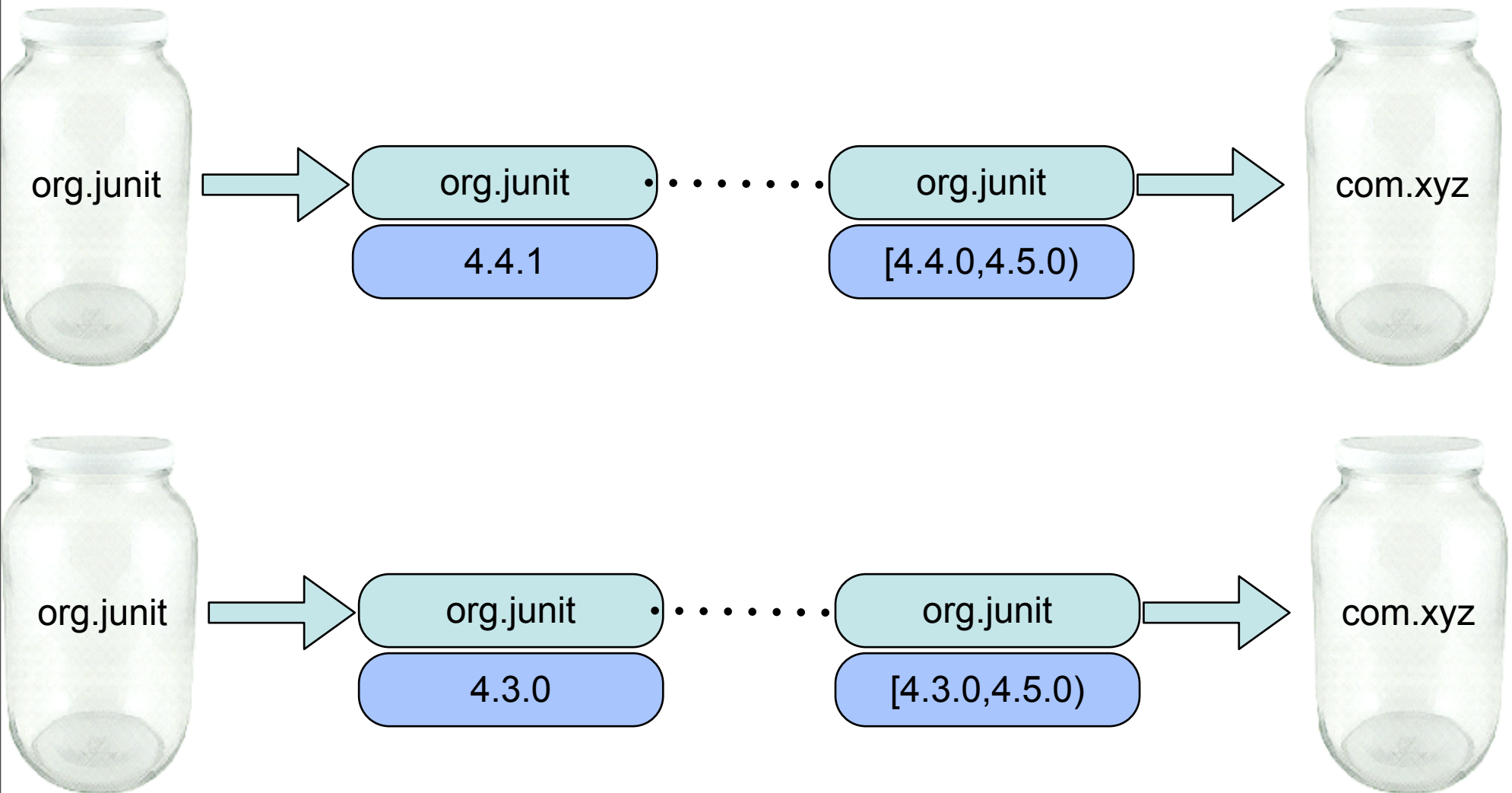


## Versioned Package Resolution



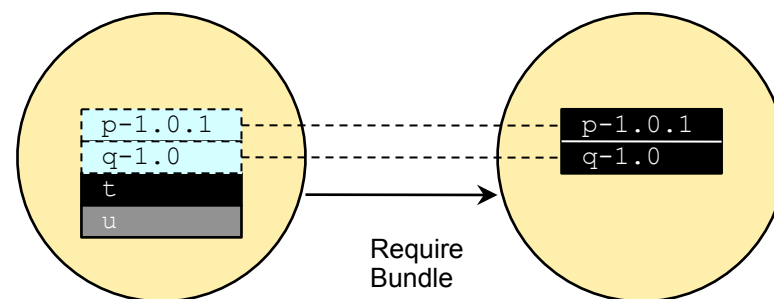


## Versioned Package Resolution

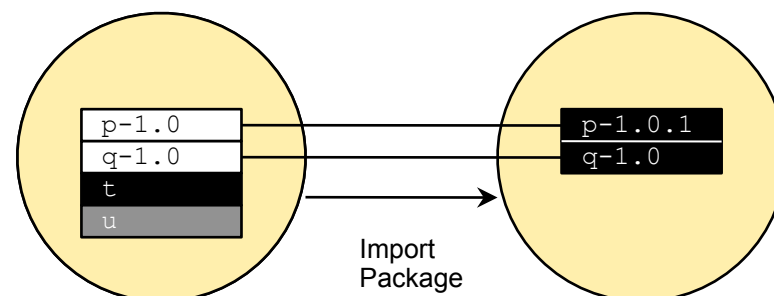


## What is the difference?

- Require-Bundle
  - Imports all exported packages of the bundle, including re-exported and split bundle packages

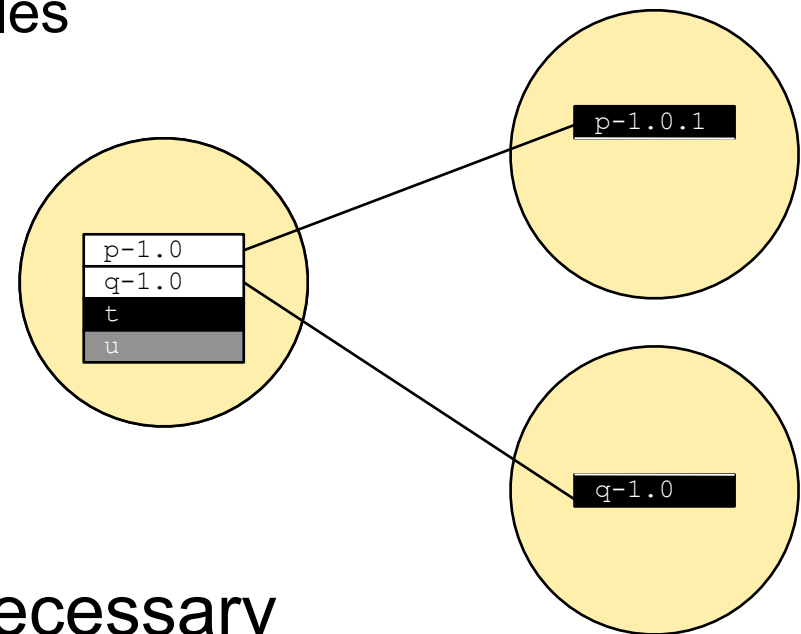


- Import-Package
  - Import just the package you need



## When to use what?

- Prefer using `Import-Package`
  - Lighter coupling between bundles
  - Less visibilities
  - Eases refactoring

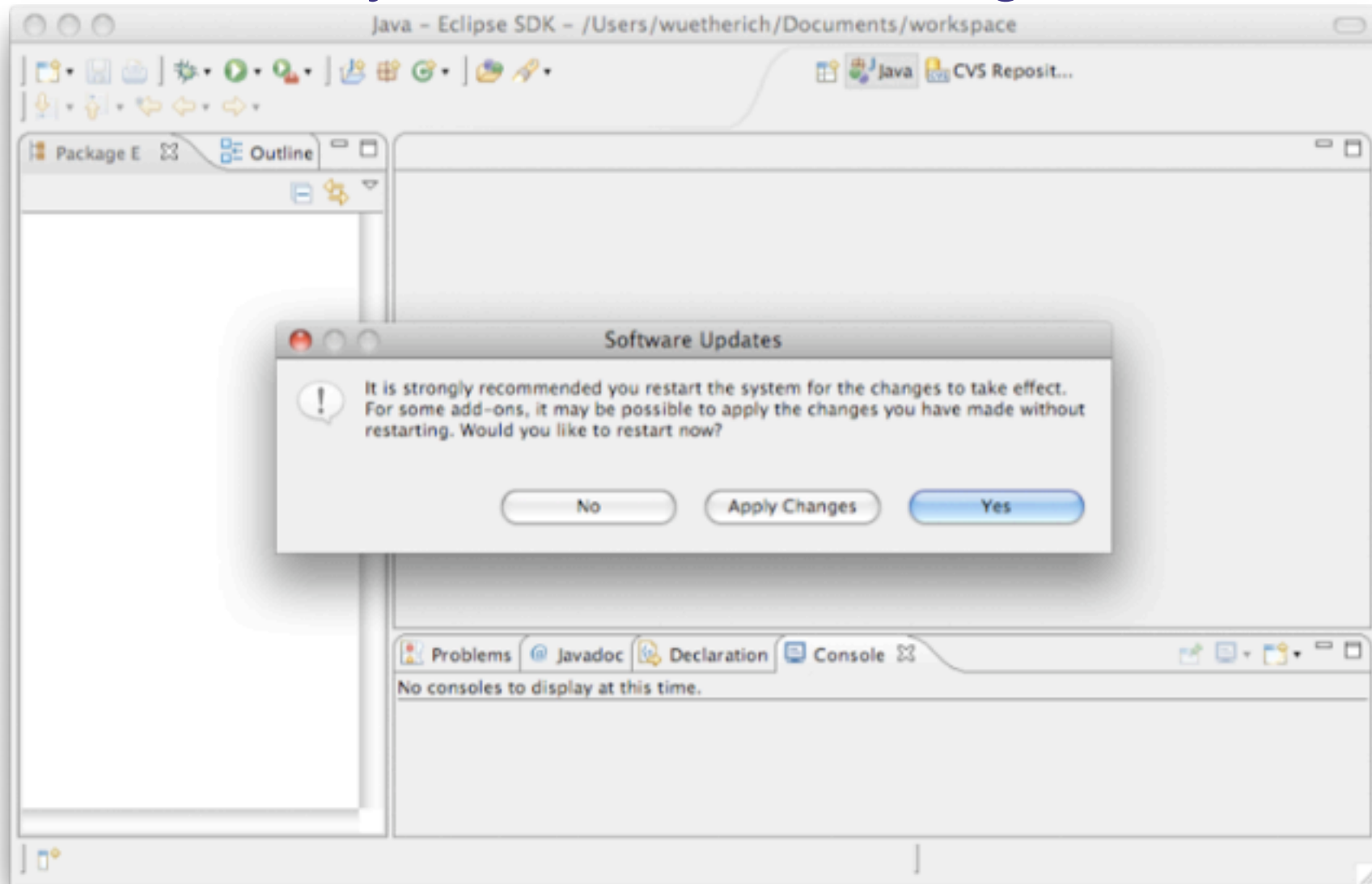


- Require-Bundle, **when necessary**
  - Don't mind higher coupling between bundles
  - split packages (same package in different bundles)

## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - **Dynamic Bundles**
  - Versioning
  - Extensions and Services
  - Compendium Services
  - OSGi Tooling
- Conclusion

## Bundles are dynamic? You're kidding...



## Dynamics with OSGi

- OSGi allows you to manage bundles at runtime
  - Install
  - Update
  - Uninstall
- But there is no magic behind the scenes
  - nothing is changed automatically
  - objects stay the same
  - references remain valid
- This means you need to cleanup after yourself!
  - ...so the GC can help you!

## Updating a bundle at runtime means...

- Dependent bundles (with wires to the updated bundle via `Require-Bundle` or `Import-Package`) are stopped and re-started
  - The consequence:
    - updating a bundle might cause the system to “restart”
    - this is not what I associate with “cool dynamics”
- ➔ When programming anticipate OSGi’s dynamics

## Think about dependencies

- **Less is more!**
  - Less dependencies
  - Separation of Core/UI
  - Dependency Inversion Principle (DIP)
- Think more about APIs
  - API in separate bundle
  - depend only on API bundle
  - implementation can change



## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - **Versioning**
  - Extensions and Services
  - Compendium Services
  - OSGi Tooling
- Conclusion

## Versioning Guidelines

- Bundle-Version: 3.6.0.qualifier
  - ♦ 3 - major version
  - ♦ 6 - minor version
  - ♦ 0 - micro version
- From the OSGi specification...

Version ranges encode the assumptions about compatibility. This specification does not define any compatibility policy; the policy decision is left to the importer that specifies a version range. A version range embeds such a policy.

However, the most common version compatibility policies are:

- major – An incompatible update
- minor – A backward compatible update
- micro – A change that does not affect the interface: for example, a bug fix

- Eclipse Versioning Guidelines
  - ♦ [http://wiki.eclipse.org/Version\\_Numbering](http://wiki.eclipse.org/Version_Numbering)

## Versioning Bundles

- On Bundle level
  - Each bundle has a version
  - You should set a version when using `Require-Bundle`
- On Package level
  - Packages should also have a version when exported
    - Remember: `Import-Package`
  - Package imports should have version ranges as well!
- Summary
  - Version everything!
  - A version isn't a marketing number!

## Versioning Tools

- PDE API Tools
  - ♦ <http://www.eclipse.org/pde/pde-api-tools/>
- Assists with the mechanics of API evolution
  - ♦ Binary compatibility (breaking) issues
  - ♦ API leaks
  - ♦ API freeze issues
  - ♦ API usage scans
  - ♦ Suggestions bundle versions
  - ♦ Runs headless and in your workspace

## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - Versioning
  - **Extensions and Services**
  - Compendium Services
  - OSGi Tooling
- Conclusion

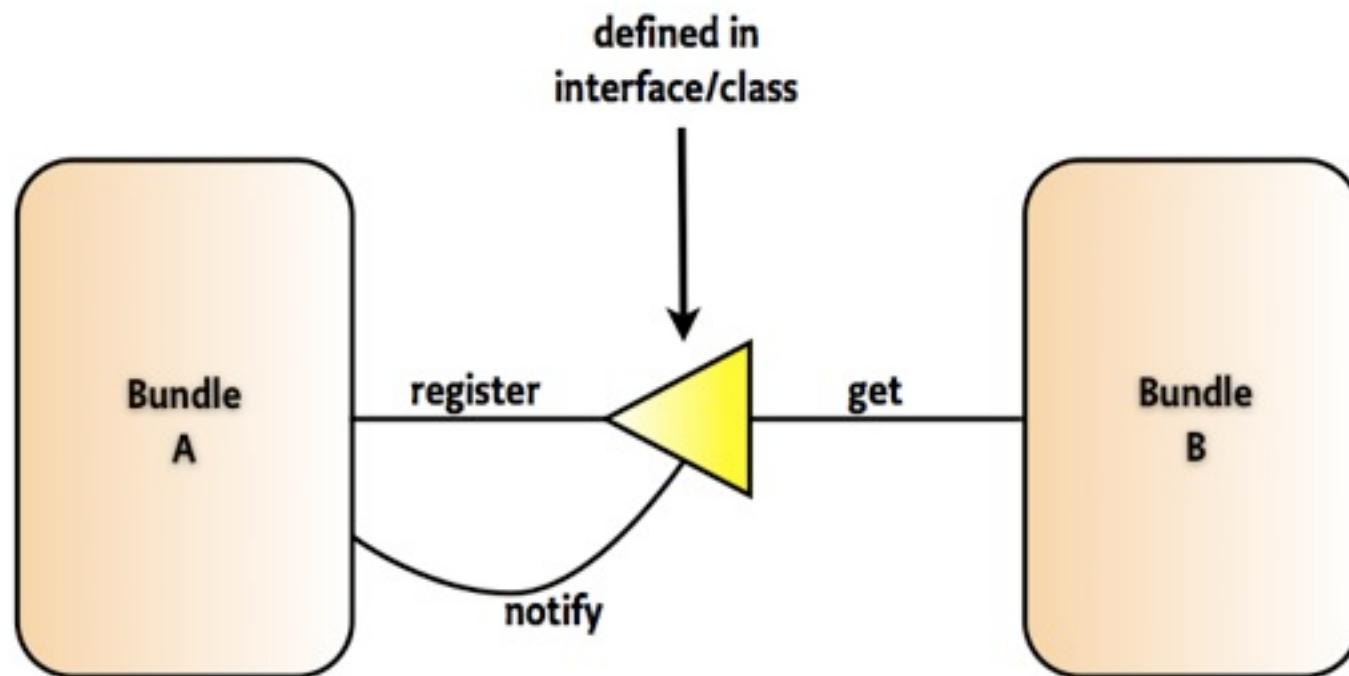
## OSGi Services

OSGi Service providers:

implement an interface and register  
an implementation

OSGi Service consumers:

lookup a service via the interface



## Versioned Contracts

- the service interface is the contract
    - many consumers possible
    - many producers possible
  - this contract is versioned
    - multiple versions of service might be available
    - you get only those that matches your dependencies
- ➔ You cannot get that with extension points
- singleton bundles
  - you always get the latest version

## “You come and go”

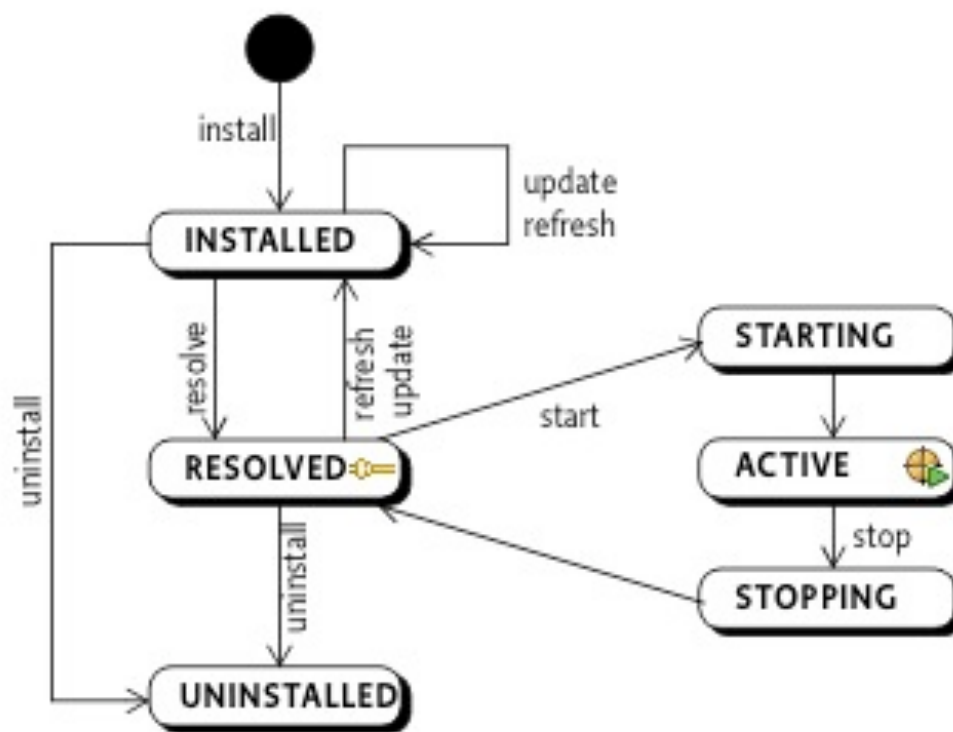
- A bundle is started:
  - services are registered
  - and available from that on
- A bundle is stopped:
  - services are unregistered
  - no longer available
- OSGi services are dynamic by definition!





## Life Cycle Differences

- Services are bound to the ACTIVE state
- Extensions are available in RESOLVED state



## Declarative and lazy

- OSGi services are **bound to the active state**
  - they need class loading to happen
  - they need objects to be created
- Lazy and declarative approaches for services
  - OSGi Declarative Services
  - OSGi Blueprint
  - iPOJO

## When to use what?

- OSGi Services:
  - Dependencies between bundles
  - Dynamics
  - Looser coupling
  - "I provide a service for anybody out there"
  - "I need a service and don't care who delivers it"
- Extension Registry:
  - UI contributions (too small for OSGi services)
  - Non-code contributions
  - "I open up myself for extensions that I don't know upfront"
  - If you have tons of thousand of extensions

## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - Versioning
  - Extensions and Services
  - **Compendium Services**
  - OSGi Tooling
- Conclusion

## Compendium Services

- OSGi has spec'd 20+ services
- LogService
- EventAdmin
- HttpService
- Declarative Services
- Configuration Admin



## LogService

- A general purpose message logger (20kb)
- LogService
  - Log message, level, exception, service ref, bundle
- LogReaderService
  - Retrieve current or previous log entries
- Note: ExtendedLogService (bug [260672](#))
  - named loggers
  - extended log entry (e.g., thread id)
  - filters for log listeners

## EventAdmin

- An inter-bundle pub-sub system (30kb)
- EventAdmin
  - publish events synchronously and asynchronously
    - `postEvent(new Event("com/acme/timer", time));`
    - `sendEvent(new Event("com/acme/timer", time));`
- EventHandler
  - handle events based on topics
    - `handleEvent(Event event)`
- Event
  - has topic and properties as attributes

## HttpService

- A way to register servlets and resources
- HttpService
  - Register servlets and resources
- HttpRegistry (org.eclipse.equinox.http.registry)

```
<extension point="org.eclipse.equinox.http.registry.servlets">  
  <servlet  
    alias="/test"  
    class="com.example.servlet.MyServlet"/>  
</extension>
```



## Declarative Services

- A declarative model for publishing, finding and binding to OSGi services (150kb)
- `ServiceTracker`
  - the programmatic way to get a service – sucks!

## ConfigAdmin

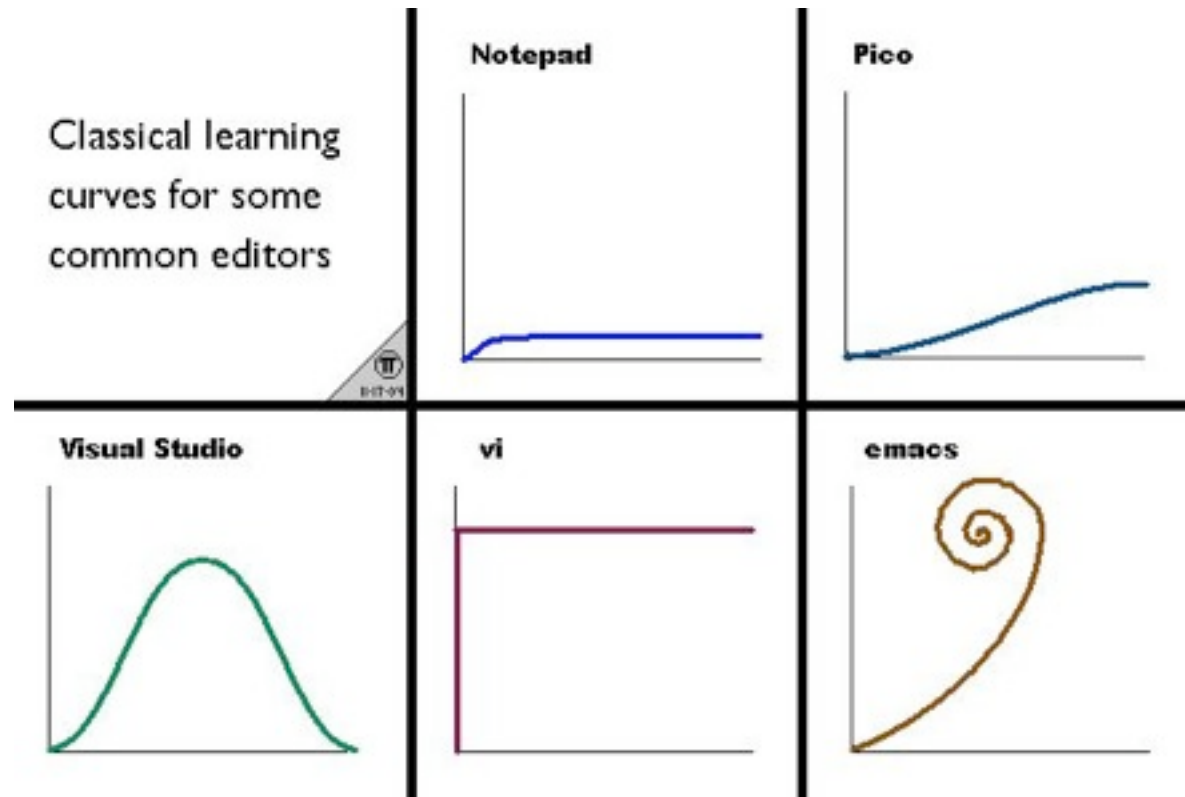
- A service to configure components (bundles)
  - A configuration is a list of key-value pairs
- The configuration admin service persists and distributes these configurations to interested parties
- Components to be configured register a `ManagedService`
- To apply several configurations of the same kind you could use a `ManagedServiceFactory`

## Overview

- Introduction
- Topics
  - Frameworks
  - Import-Package vs. Require-Bundle
  - Dynamic Bundles
  - Versioning
  - Extensions and Services
  - Compendium Services
  - **OSGi Tooling**
- Conclusion

## Tools Tools Tools

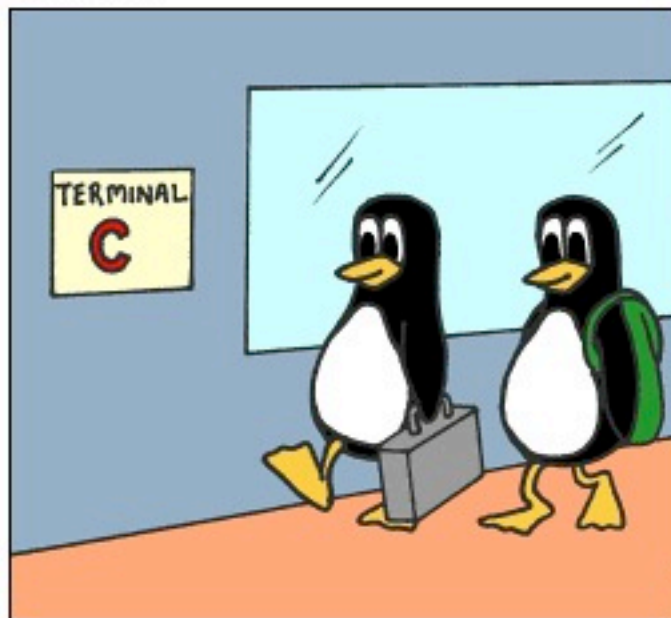
- In OSGi land, there are lots of options...
  - ♦ PDE, Maven, BND...



## Tools and Religion

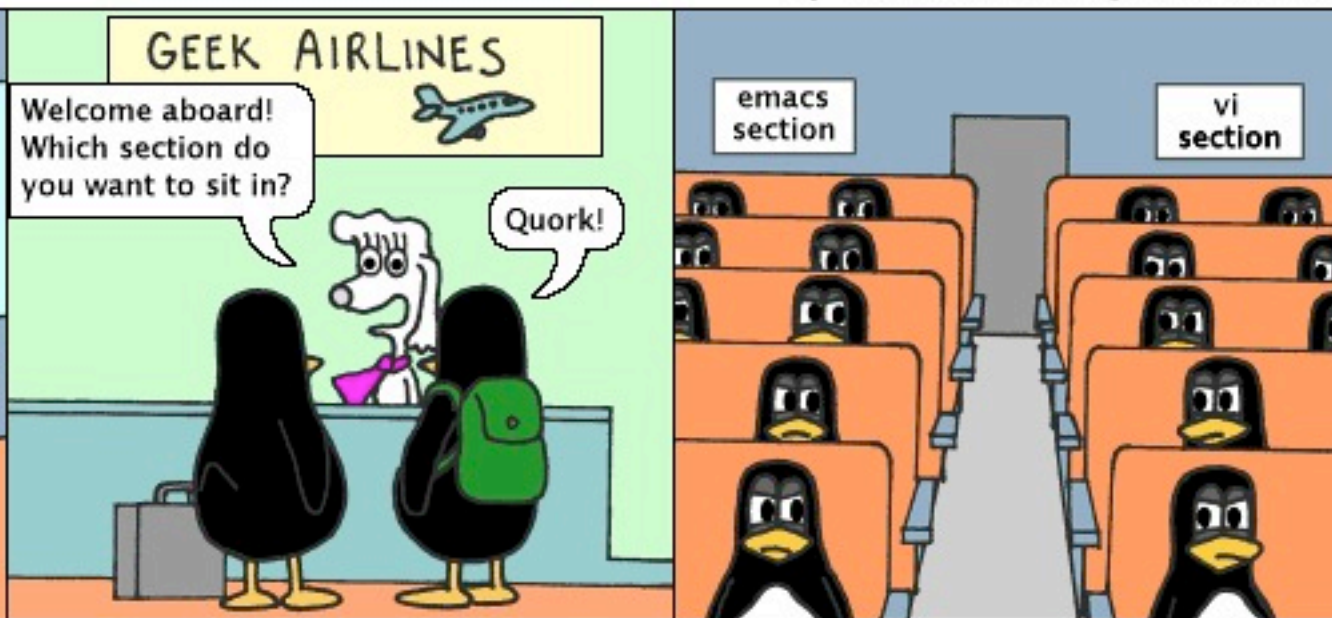
- Tools inspire religious debate sometimes...
- Choice of tooling will be like your choice of religion...
- Remember emacs vs. vi ;)?

Hackles



<http://hackles.org>

By Drake Emko & Jen Brodzik



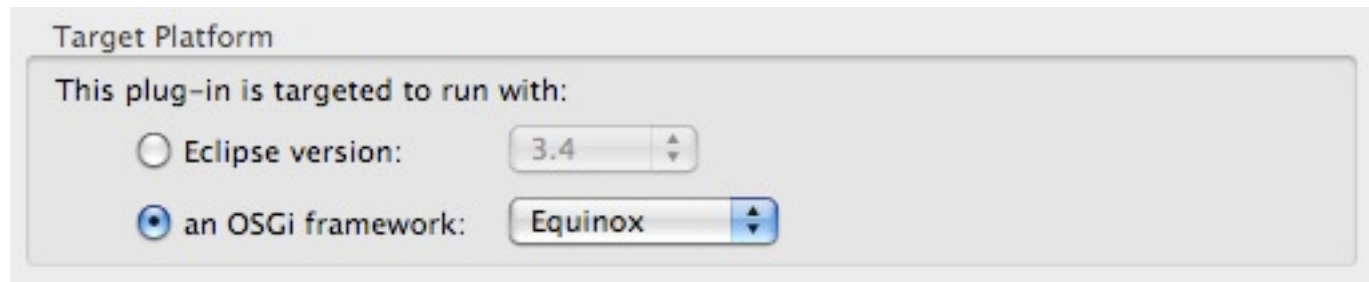
Copyright © 2003 Drake Emko & Jen Brodzik

## OSGi Tooling Types

- Two main types of OSGi tooling
- Manifest First (e.g., PDE)
  - ♦ Provides tooling to hand craft OSGi artifacts
  - ♦ Centers around the OSGi manifest
- Template Driven (e.g., BND)
  - ♦ Provides tooling to use templates to generate OSGi artifacts

## PDE

- Eclipse has been tooling OSGi forever with PDE
  - Plug-ins == Bundles! Blugins?
  - Tens of thousands of developers using PDE for over 5 years
- PDE provides world class tooling for OSGi:
  - Bundles
  - Fragments
  - Declarative Services
- New Plug-in Project wizard has OSGi love



## BND

- Bundle Tool (BND)
  - creates and diagnoses OSGi bundles
  - Maven, Eclipse and Ant integration
  - <http://www.aqute.biz/Code/Bnd>
- Relies on specification (.bnd file) + classpath

```
Export-Package: aQute.service.*  
Import-Package: javax.servlet.http;version="[2,3)", *
```

- Generates bundle artifacts like manifests
- Useful for converting third party libs to bundles



## Sigil

- Provides OSGi Tooling
  - <http://sigil.codecauldron.org/>
  - driven by sigil.properties file
  - BND used under the covers
  - being donated to Apache Felix project
- bundles fetched from repositories
  - based on your Import-Package statements





## Thank you for your attention!

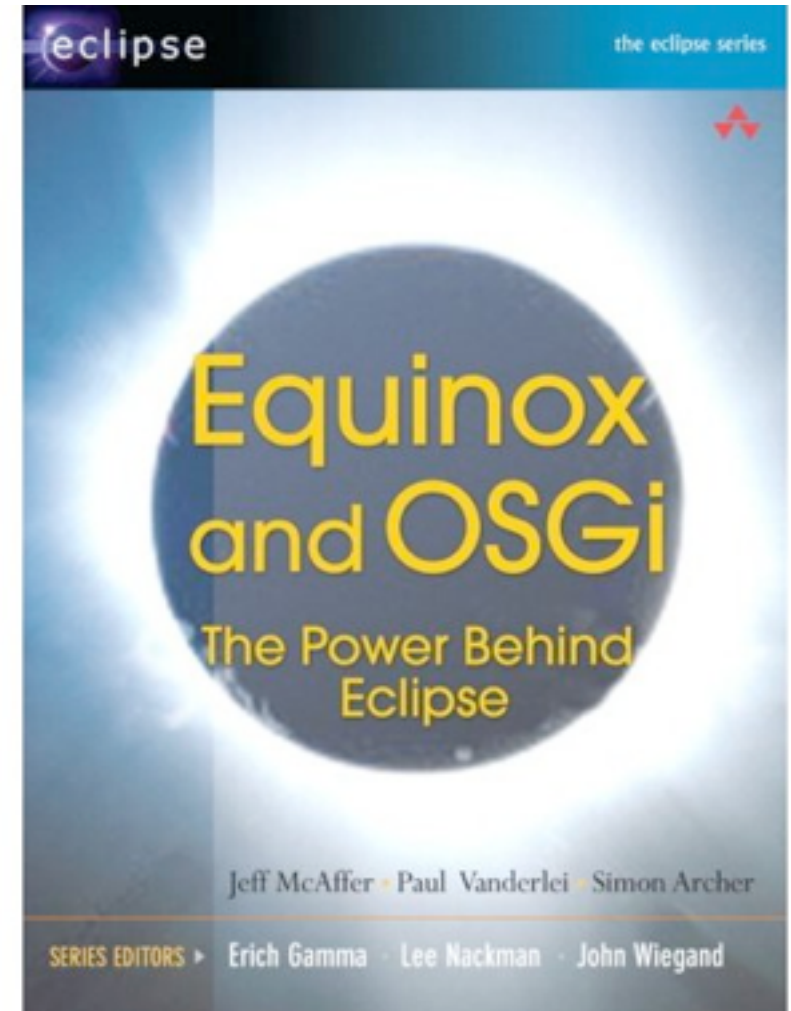
- In Summary...
  - OSGi is more than Eclipse
  - Eclipse is building on OSGi
- Questions and feedback welcome!

Chris Aniszczyk  
[zx@eclipsesource.com](mailto:zx@eclipsesource.com)  
<http://twitter.com/caniszczyk>

Thanks to...  
Neil Bartlett  
Bernd Kolb  
Martin Lippert

## Want to learn more?

- Equinox OSGi Book
  - ◆ Learn OSGi using Eclipse
  - ◆ TOAST Demo



<http://my.safaribooksonline.com/9780321561510>