Using Scala in an Eclipse/OSGi environment

Martin Gamwell Dawids
mgd@maconomy.com

Michael Werner-Gram
mgh@maconomy.com

maconomy
don't waste

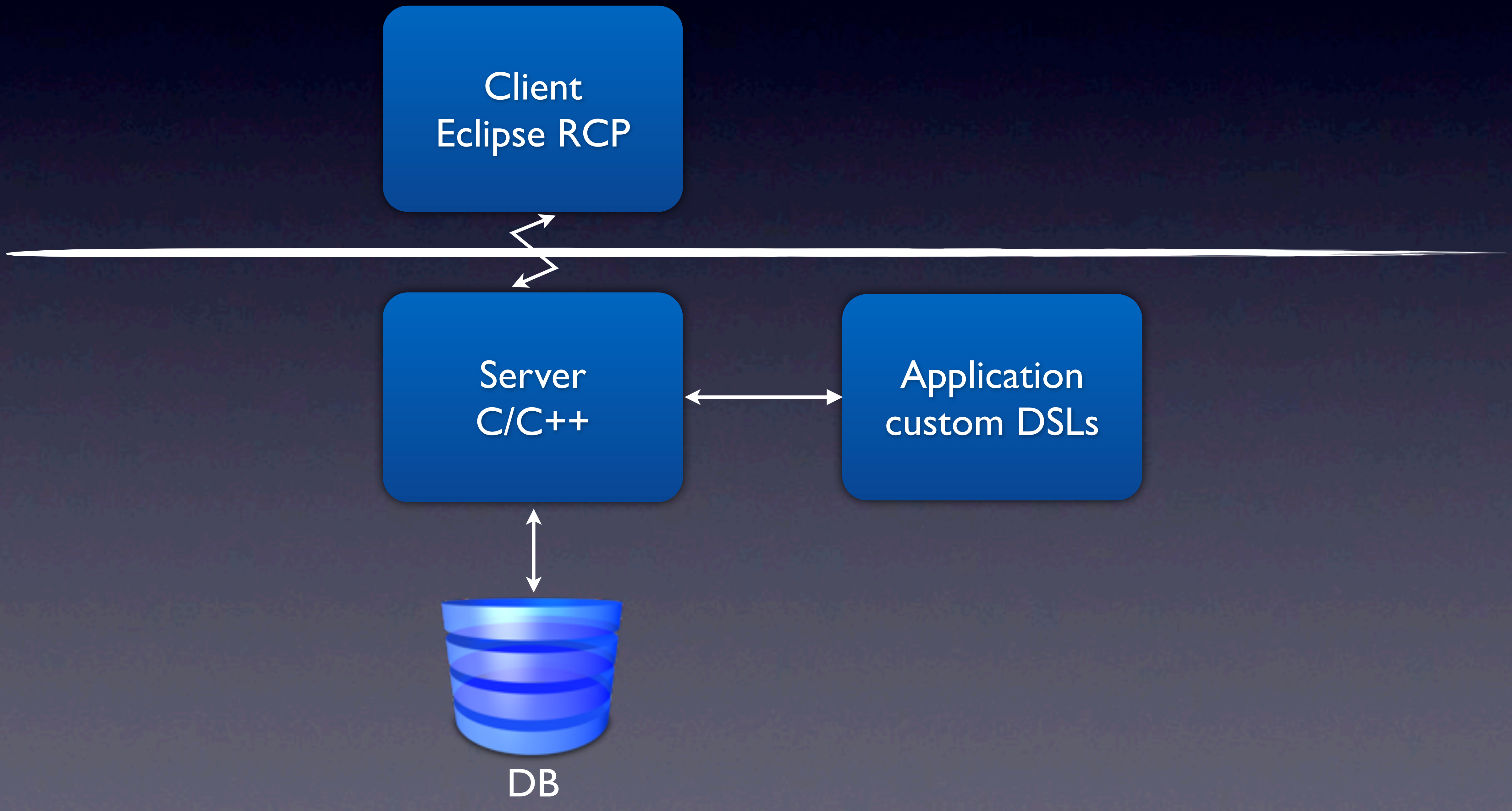
Outline

- OSGi development with Eclipse and Scala
    - Eclipse IDE developer experience
    - Scripting builds
- Making an OSGi-aware Scala compiler

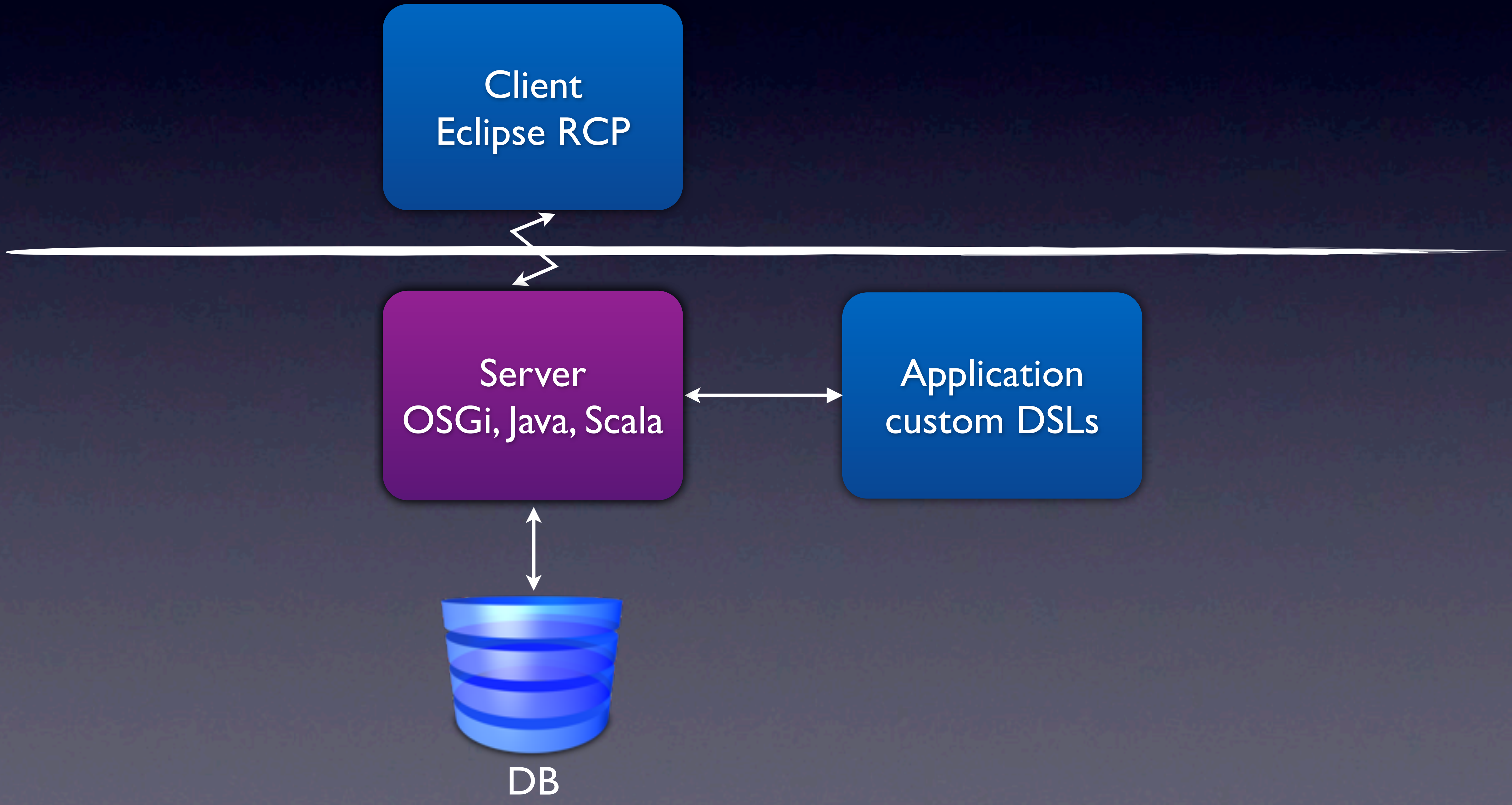
Maconomy

- Large business application for financial management, job costing, project management, …
- Custom C/C++ application server
  - 20 years old (originally in Pascal)
  - 2+ million lines of code
- Application written in custom domain specific languages (DSLs)
  - 15+ different DSLs
  - 3+ million lines of code

Maconomy architecture



Maconomy architecture

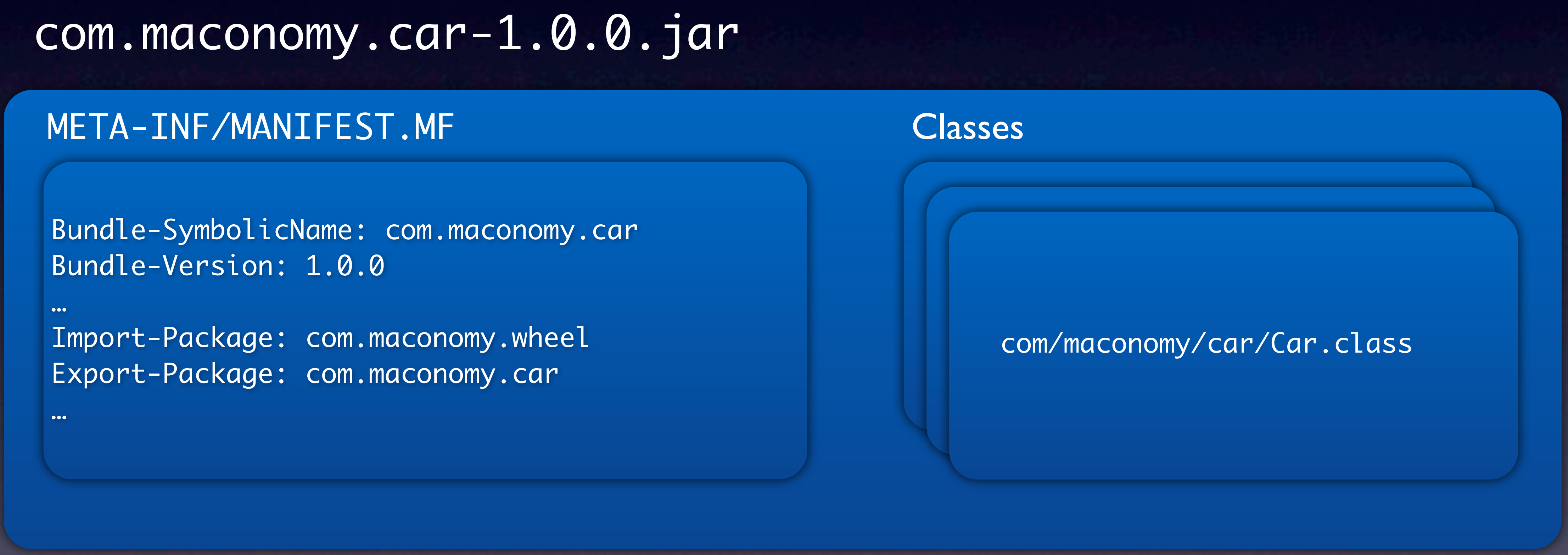


Project choices

- Scala as main programming language
  - High productivity, readability, maintainability
- OSGi platform
  - Large project needs modularisation mechanism
- Eclipse IDE
  - Currently best OSGi developer experience
  - Share project definitions with existing projects

OSGi

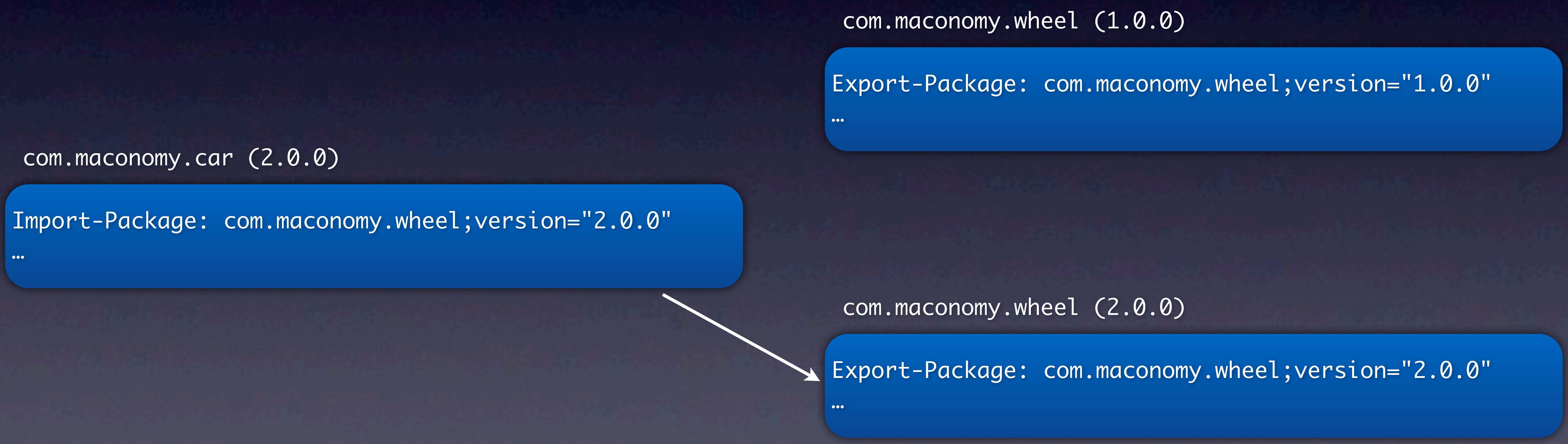
Code organised in bundles



```
Bundle-SymbolicName: com.maconomy.car
Bundle-Version: 1.0.0
…
Import-Package: com.maconomy.wheel
Export-Package: com.maconomy.car
…
```

OSGi

Dynamic dependency resolution between bundles



com.maconomy.wheel (1.0.0)

```
Export-Package: com.maconomy.wheel;version="1.0.0"
…
```

com.maconomy.car (2.0.0)

```
Import-Package: com.maconomy.wheel;version="2.0.0"
…
```

com.maconomy.wheel (2.0.0)

```
Export-Package: com.maconomy.wheel;version="2.0.0"
…
```

Eclipse & OSGi development

- PDE – Eclipse extension for OSGi development
- OSGi-aware Java compiler
  - Automatically resolves bundle dependencies when coding
  - Enforces OSGi import/export constraints at compile-time

Eclipse & OSGi development

- IDE assistance for Java
  - Importing packages, nice GUI editors, etc.
- Provides model for testing OSGi bundles

Eclipse, OSGi & Scala development

- Scala plugin for Eclipse
  - Works well in standard Eclipse
- Bringing in PDE complicates matters
  - No IDE assistance for OSGi import-package, …
  - Changes in MANIFEST.MF requires manual recompilation
- Scala compiler itself is not OSGi-aware
  - *All* packages are visible in bundle's dependencies
  - Runtime errors – no static guarantees

maconomy
don't waste

Eclipse PDE developer experience

Feature	Java	Scala
OSGi-aware compiler	✓	✗
Nested JARs in bundles	✓	✓
OSGi metadata GUI editors	✓	✓
OSGi import-package assistance	✓	✗
PDE testing model	✓	✓
Bundle dependency resolution	✓	✓
Automatic rebuild on MANIFEST change	✓	✗

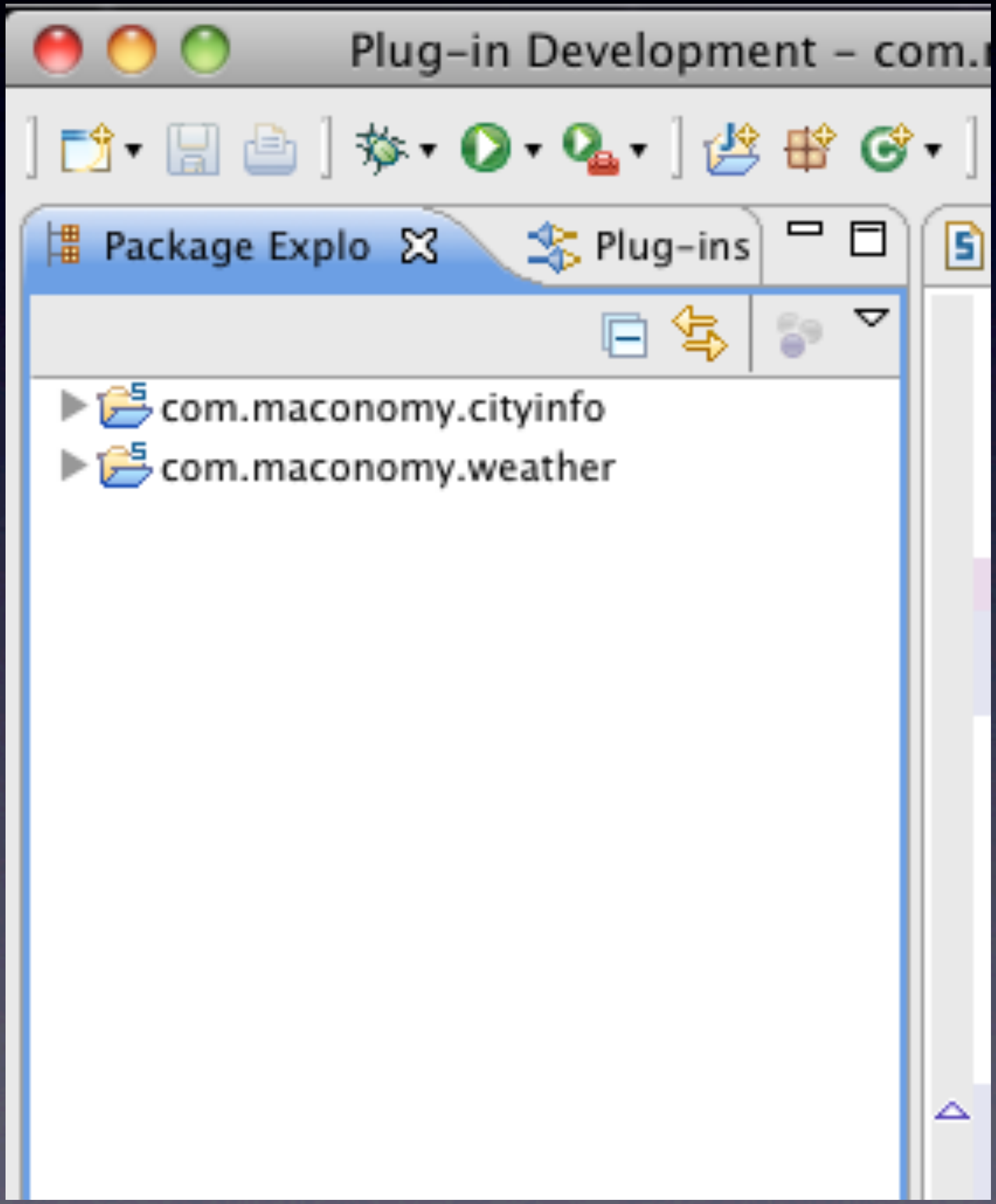
Scripted builds

- Our product shouldn’t just run in the IDE
- Continuous Integration (CI) for building and testing
- Release

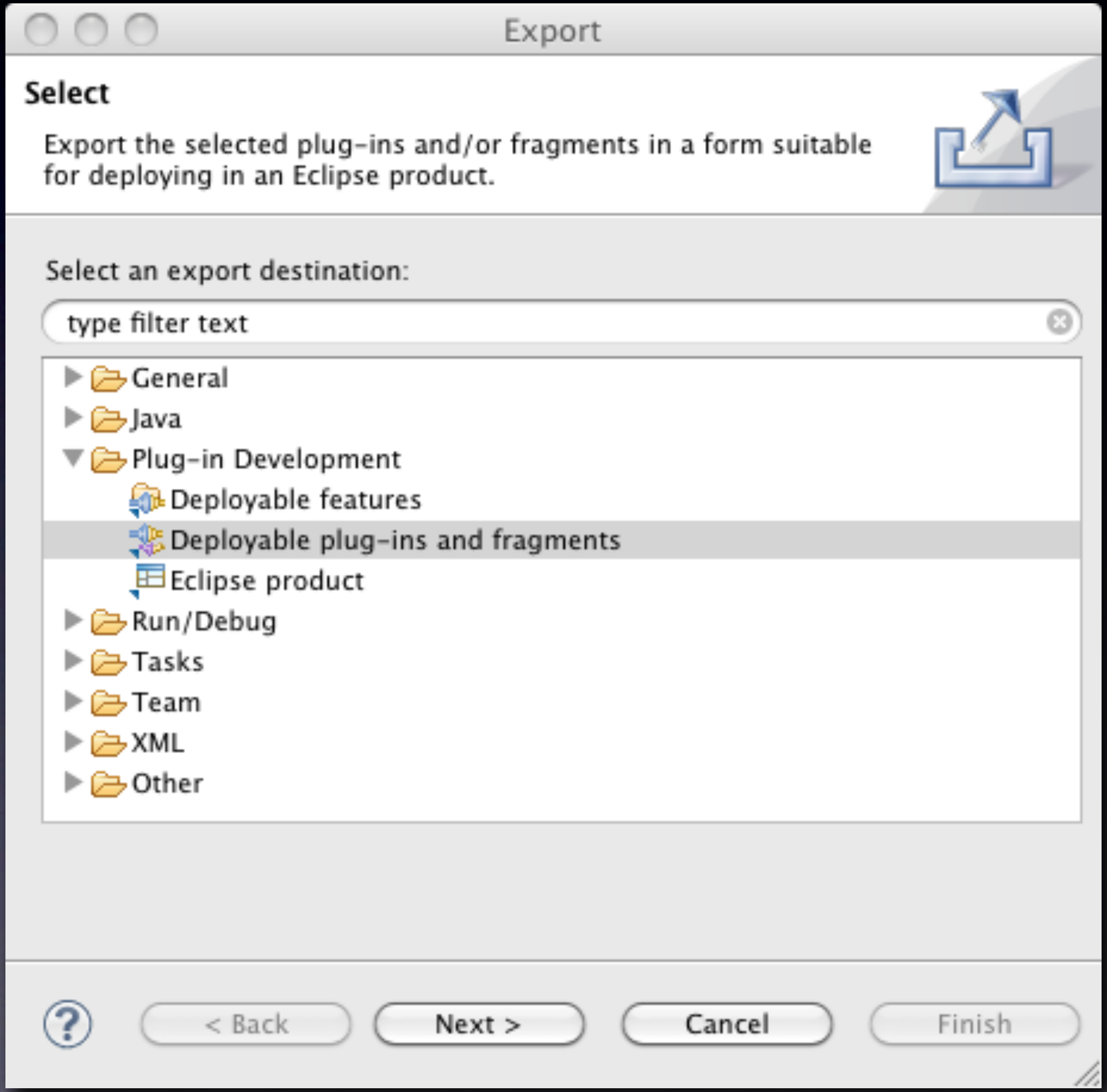
Requirements for builds

- Eclipse as single point of development
  - Shared project definitions between IDE and CI
  - No tweaks to external build scripts necessary
- Compilation must behave the same in CI
- Test execution in CI like in Eclipse

Building with Eclipse



Building with Eclipse



Export

Select

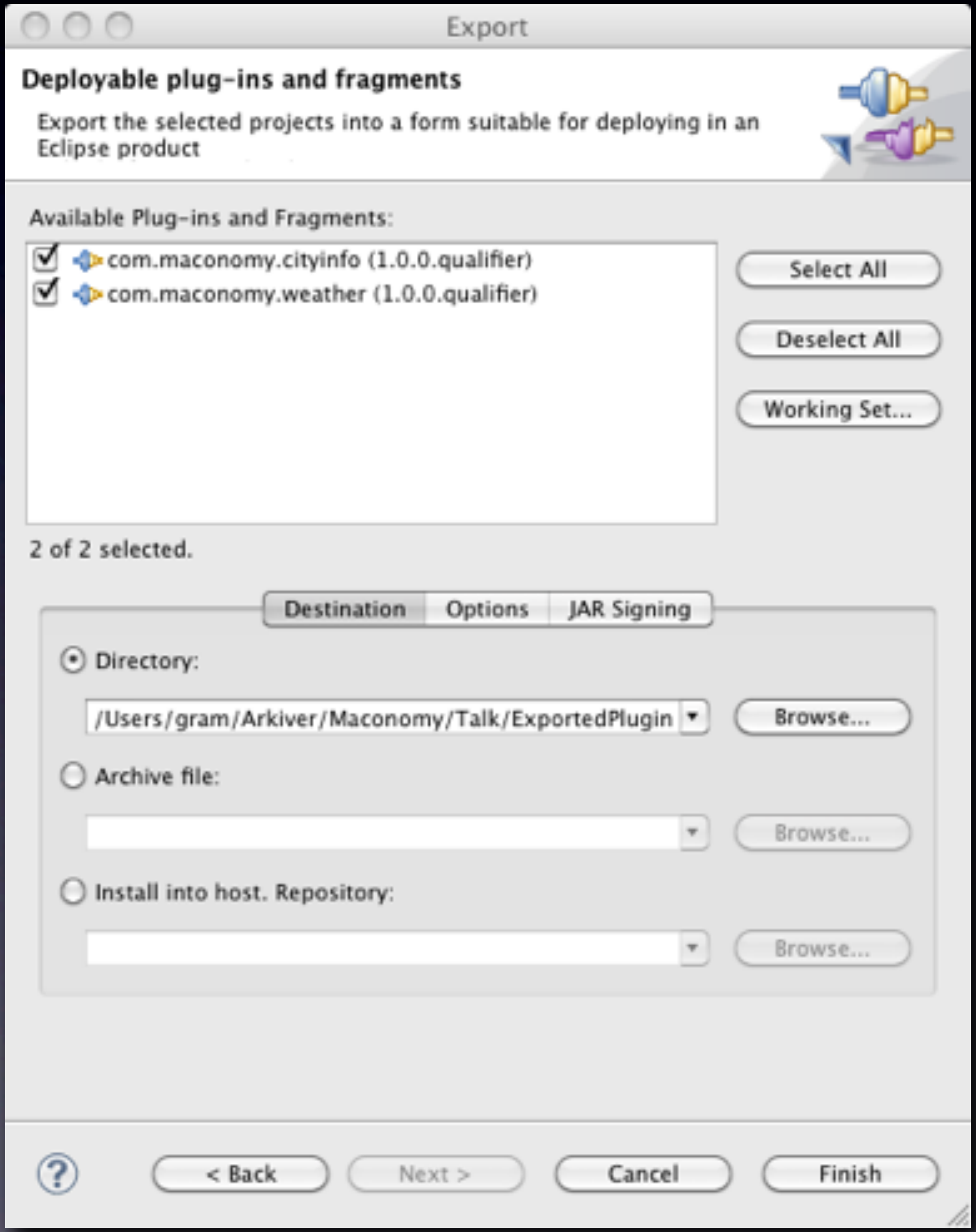
Export the selected plug-ins and/or fragments in a form suitable for deploying in an Eclipse product.

Select an export destination:

type filter text

maconomy
don't waste

Building with Eclipse



Export

Deployable plug-ins and fragments

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Plug-ins and Fragments:

- [✓] com.maconomy.cityinfo (1.0.0.qualifier)
- [✓] com.maconomy.weather (1.0.0.qualifier)

2 of 2 selected.

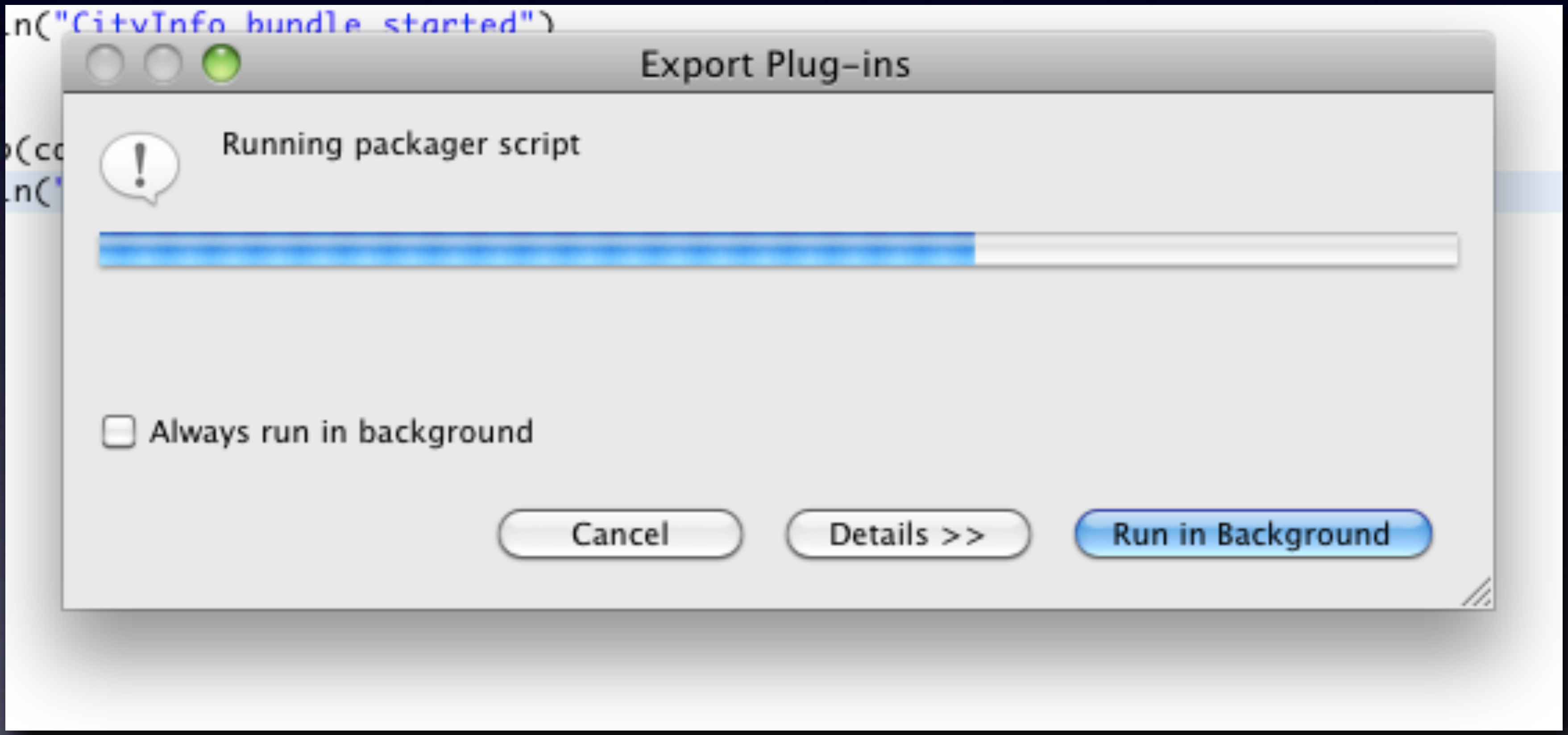
Destination | Options | JAR Signing

[✓ Directory:] /Users/gram/Arkiver/Maconomy/Talk/ExportedPlugin

Archive file:

Install into host. Repository:

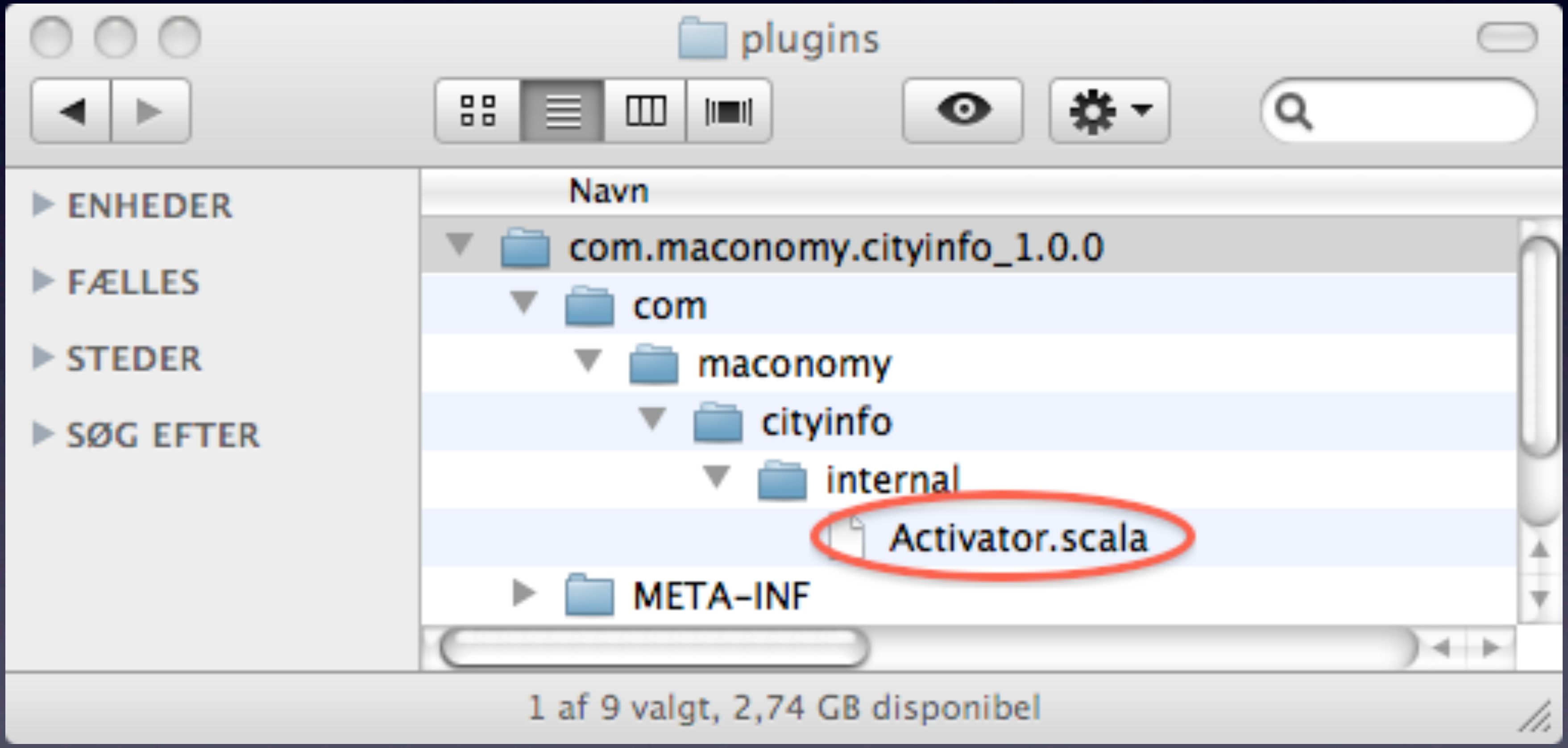
Building with Eclipse



Running packager script

Always run in background

Building with Eclipse



```
com.maconomy.cityinfo_1.0.0
  com
    maconomy
      cityinfo
        internal
          Activator.scala
  META-INF
```

1 af 9 valgt, 2,74 GB disponibel

maconomy
don't waste

PDE build issues

- Eclipse PDE only compiles Java – no matter what the Eclipse IDE can do…
- Workaround: Custom Callbacks
  - Hopeless error reporting, missing documentation
- Scripting PDE builds
  - Obscure Ant build.xml files – poorly documented
- Poor CI tools integration

maconomy
don't waste

Scripted builds

Feature	PDE Export
OSGi-aware Java compiler	✔
OSGi-aware Scala compiler	✘
Shared project definitions	✔
Test execution like Eclipse	✔
Eclipse dependency resolution	✔
Good CI tools integration	✘

Maven?

- Managing and building projects
- Integrates well with many CI tools
- Wealth of plugins enhances Maven
- Well-documented
- Useful for us?

Maven & maven-bundle-plugin

- Used by many projects for building bundles
- Serious issues in our context:
  - Generates MANIFEST.MF – we use Eclipse for editing manifests
  - Uses Sun's Java compiler – not OSGi-aware
  - Must maintain two sets of project configurations
  - Dependency resolution not like Eclipse, is manual

maconomy
don't waste

Scripted builds

Feature	PDE Export	Maven bundle plugin
OSGi-aware Java compiler	✔	✘
OSGi-aware Scala compiler	✘	✘
Shared project definitions	✔	✘
Test execution like Eclipse	✔	✘
Eclipse dependency resolution	✔	✘
Good CI tools integration	✘	✔

Maven & Tycho

- Tycho is a set of Maven plugins for building Eclipse OSGi projects
- Re-uses Eclipse project files
- No explicit specification of dependencies
  - Resolves dependencies like Eclipse and injects them into the Maven project model
- Uses Eclipse's Java compiler – OSGi-aware
- Run tests like inside Eclipse

maconomy
don't waste

Scripted builds

Feature	PDE Export	Maven bundle plugin	Tycho
OSGi-aware Java compiler	✔	✘	✔
OSGi-aware Scala compiler	✘	✘	✘*
Shared project definitions	✔	✘	✔
Test execution like Eclipse	✔	✘	✔
Eclipse dependency resolution	✔	✘	✔
Good CI tools integration	✘	✔	✔

* Uses maven-scala-compiler plugin

OSGi-aware Scala compiler

- Just one feature missing now:
  - Checking import/export constraints
- How does the Eclipse Java compiler do it?

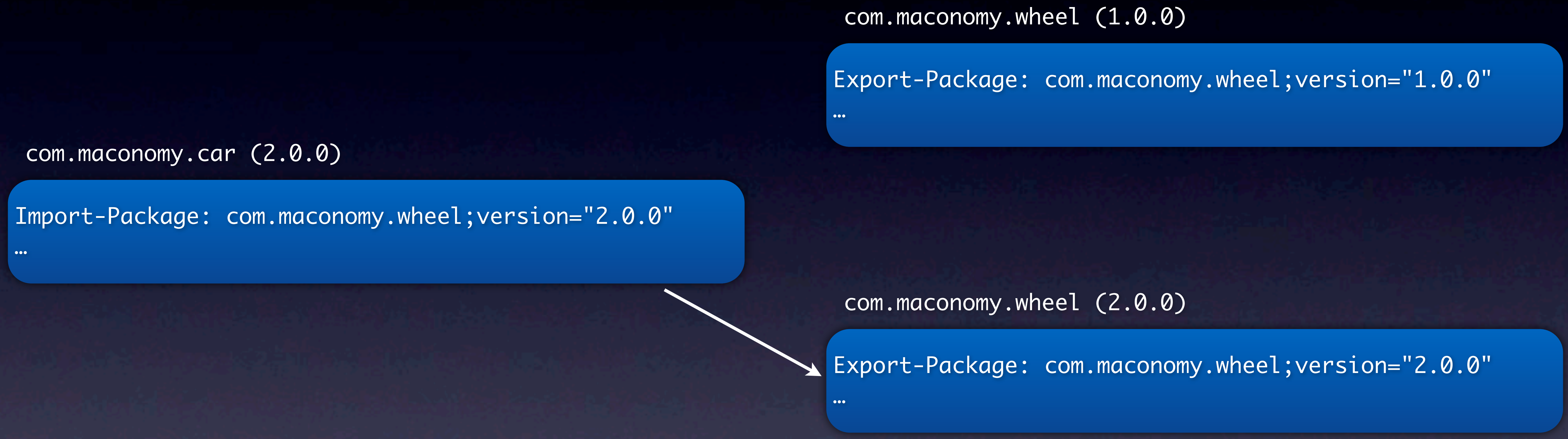
Classpaths

- Ordinary classpath for class resolution
  `/path/to/bundle1.jar:/path/to/bundle2.jar`
- Compiler just takes first class that matches in classpath
- When compiling a bundle, OSGi modularity must be honored
  - *Only* exported packages can be accessed
  - …and *only* when explicitly imported

Annotated classpaths

- Eclipse Java compiler uses annotations
  - Each classpath entry annotated with constraints
  - Tells the compiler in which packages it is allowed to look for classes
  - Differs for each bundle being compiled
  - Constraints calculated by Eclipse and Tycho from OSGi specifications and resolved bundle dependencies

Annotated classpaths



com.maconomy.wheel (1.0.0)

```
Export-Package: com.maconomy.wheel;version="1.0.0"
…
```

com.maconomy.car (2.0.0)

```
Import-Package: com.maconomy.wheel;version="2.0.0"
…
```

com.maconomy.wheel (2.0.0)

```
Export-Package: com.maconomy.wheel;version="2.0.0"
…
```

Classpath when compiling com.maconomy.car-2.0.0.jar

```
com.maconomy.wheel-2.0.0.jar[+com/maconomy/wheel/*:?**/*]
```

Scala OSGi compiler

- Custom Scala compiler plugin
  - Understands annotated classpaths
  - Checks that all classes used in a compilation unit does not violate the constraints
- Custom Maven Scala plugin
  - Gets annotated classpath from Tycho
  - Passes annotated classpath to Scala compiler plugin

maconomy
don't waste

Scripted builds

Feature	PDE Export	Maven bundle plugin	Tycho	Tycho + Scala plugins
OSGi-aware Java compiler	✓	✗	✓	✓
OSGi-aware Scala compiler	✗	✗	✗	✓
Shared project definitions	✓	✗	✓	✓
Test execution like Eclipse	✓	✗	✓	✓
Eclipse dependency resolution	✓	✗	✓	✓
Good CI tools integration	✗	✓	✓	✓

Conclusions

- Eclipse as single point of development
    - Scripted builds using
      Maven + Tycho + Scala + custom plugins
    - OSGi-aware Scala compiler in scripted builds
- Wish-list for better developer experience
    - Native OSGi-aware Scala compiler
    - Scala Eclipse Plugin and PDE integration