AOP im Einsatz mit OSGi und RCP

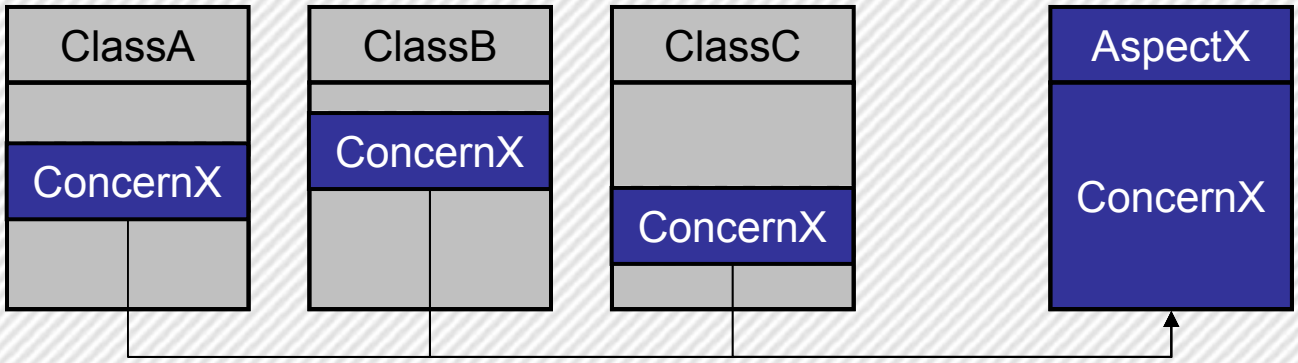
Martin Lippert, Peter Friese
und Heiko Seeberger

Agenda

- Einführung
- Aspect-Weaving im Überblick
- Aspect-Weaving für OSGi:
  - Equinox Aspects
- Anwendungsfall: Security für Eclipse-RCP
- Abschluss

Aspektorientierte Programmierung

- Modularisierung mit OO-Mitteln ist gut
  - Klassen, Interfaces, Vererbung, etc.
- AOP ergänzt die OO-Mittel
  - AOP modularisiert „Cross-Cutting Concerns“



AOP im Einsatz

- Mittlerweile etabliertes Werkzeug
  - AspectJ: Eine mächtige Spracherweiterung für Java
  - Spring AOP: Einfache Handhabung für Enterprise-Anwendungen
  - App-Server: Verwenden intern AOP-Mittel

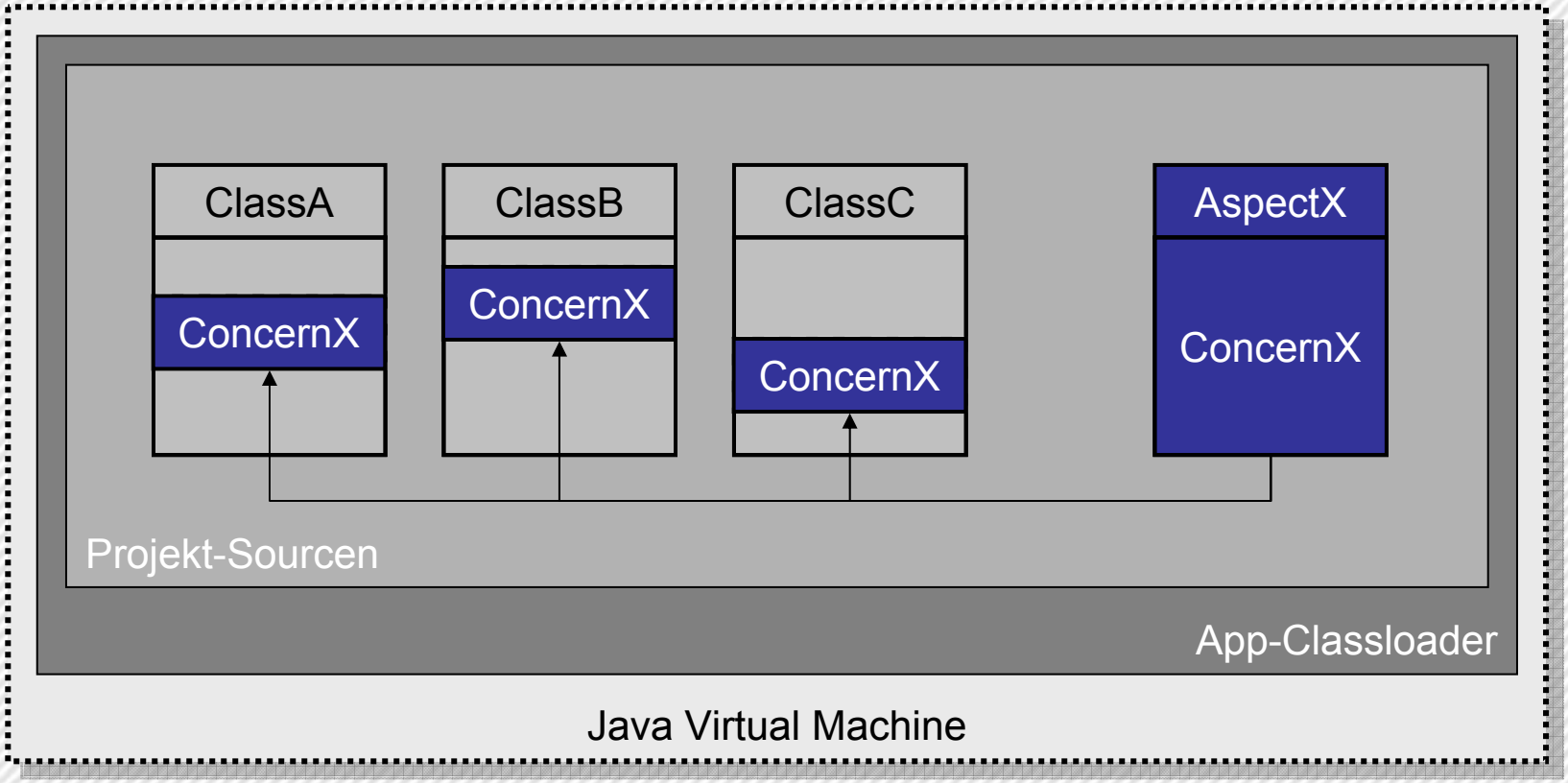
AspectJ = AOP für Java

- Mächtige AOP-Erweiterung für Java
- Eclipse-Projekt: www.eclipse.org/aspectj
- Gute Tool-Unterstützung:
  - AJDT für Eclipse
  - Spring-IDE für Eclipse
    - Für die Verbindung von Spring-AOP und AJDT

Wie funktioniert es?

- Der Standard-Fall:
  - AspectJ compiliert die Aspekte und verwebt diese (Compile-Time Weaving)
  - Sehr gute Unterstützung in der IDE
    - Inkrementelles Compilieren
    - Marker und Crosscutting View
  - Transparente Technologie für den Entwickler

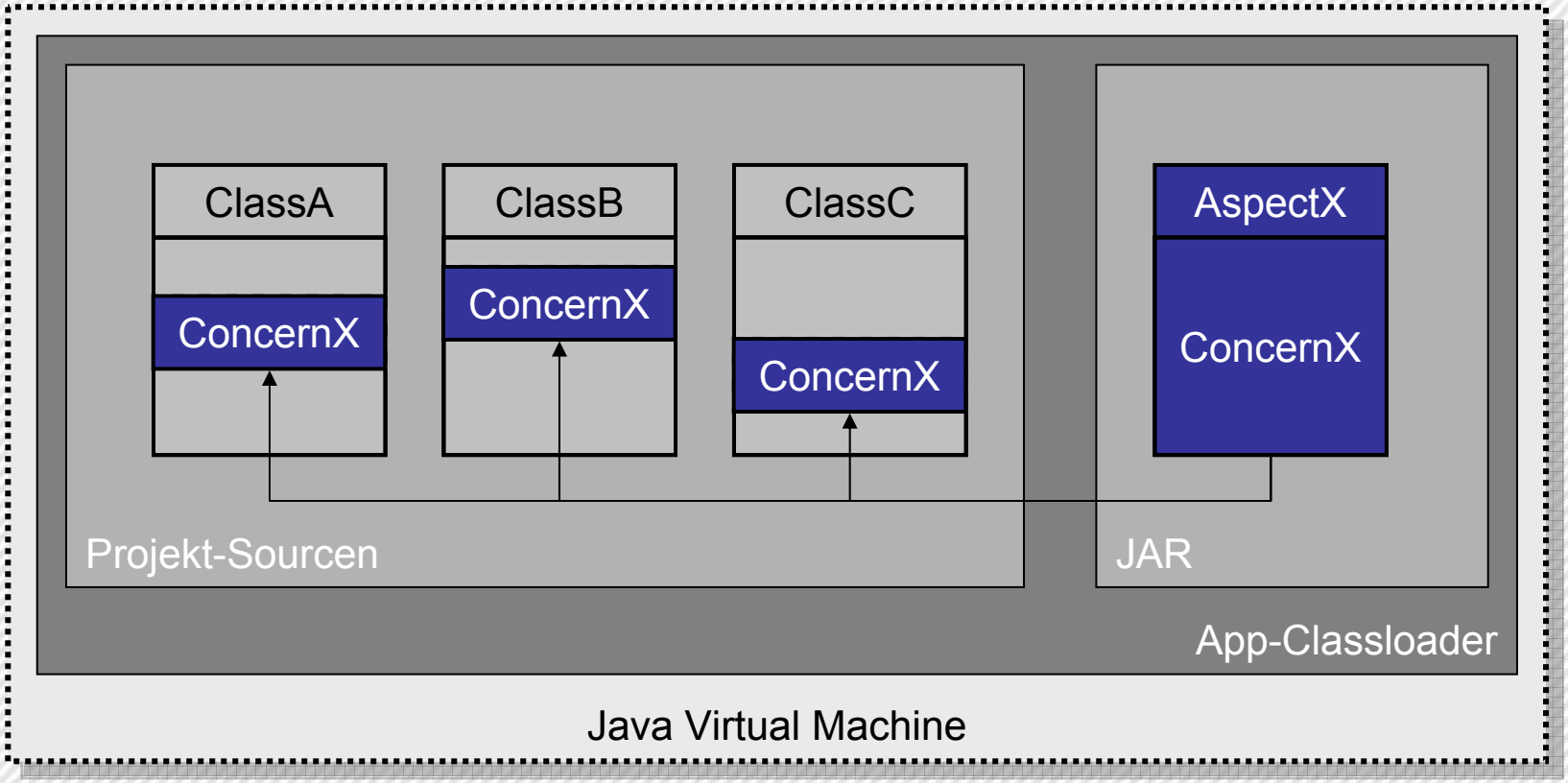
Der Standard-Fall



Aspect-Libraries

- Aspekte werden mit AspectJ compiliert und in ein JAR-File verpackt
  - Z.B. auch abstrakte Aspekte
- Verweben mit anderen Klassen und Aspekten beim Build
  - In der IDE oder beim „Headless Build“
  - Als ob die Aspekte direkt in der IDE vorhanden wären

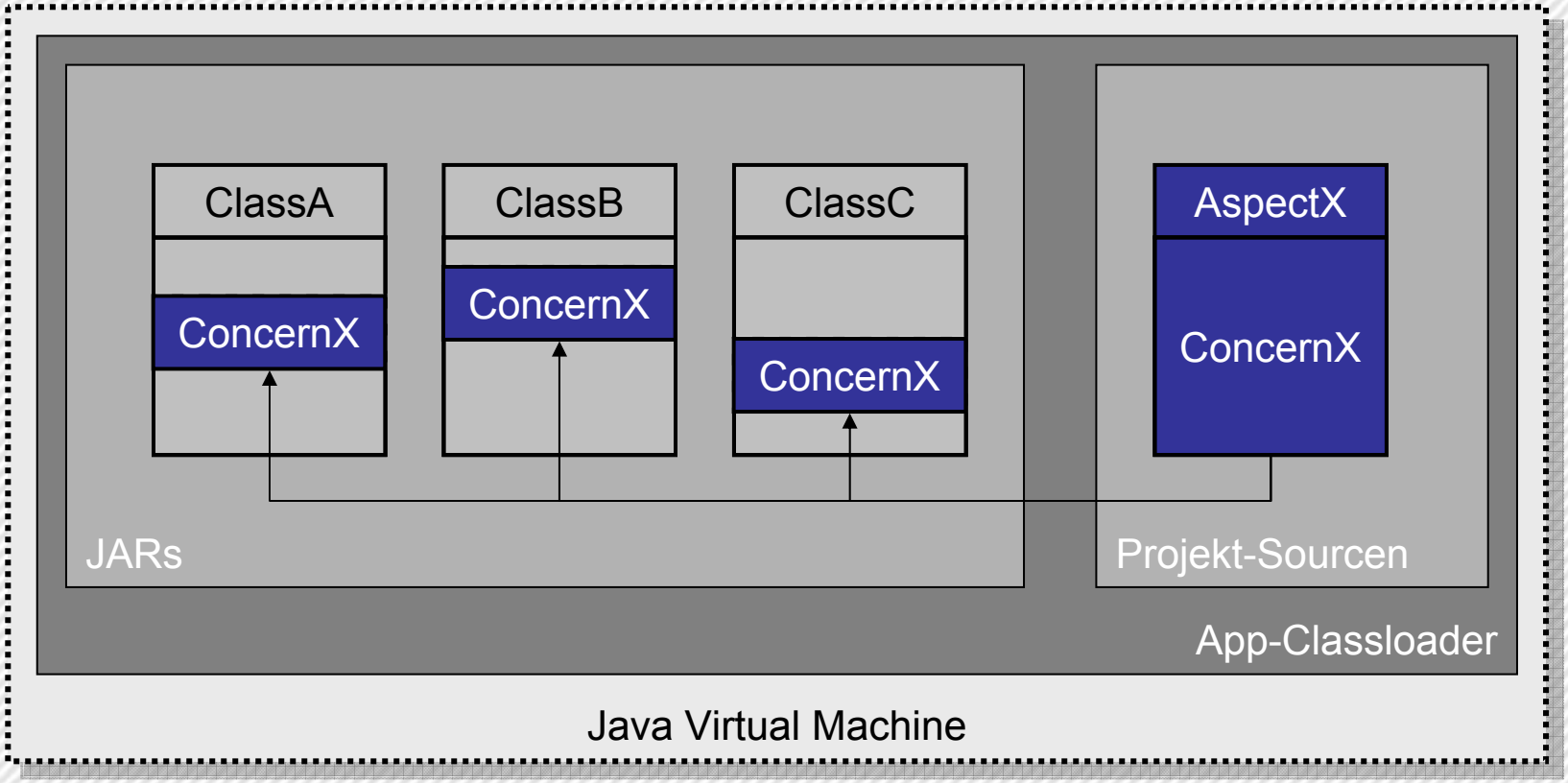
Aspect-Libraries



„Projektfremde“ Targets

- Was passiert, wenn sich Aspekte auf vorcompilierte Klassen auswirken sollen?
  - Bibliotheken
  - Frameworks
  - andere Projekte
  - etc.

„Projektfremde“ Targets



Variante 1: Compile-Time

- Der Compiler bekommt die JARs und webt die Aspekte direkt in die JARs ein
- Vor- und Nachteile:
  - \+ Direkt in der IDE
  - \+ Zur Auslieferung ist alles fertig
  - \- Alle Libs und Bibliotheken müssen bekannt und veränderbar sein
  - \- Wenn sich etwas verändert, muss neu gebaut werden

Variante 2: Load-Time

- Aspect-Weaving beim Laden der Klassen in die VM (Load-Time Weaving - LTW)
- Vor- und Nachteile:
  - \+ Kann sogar mit System-Libs umgehen
  - \+ Funktioniert, auch wenn sich etwas verändert
  - \+ Es müssen nicht alle Libs bekannt sein
  - \- Kostet einmalig beim Laden der Klasse Performance
  - \- Man sieht nicht alles in der IDE

Load-Time Weaving

- Modularisierung kann lange aufrecht erhalten werden
- Aber nur, solange Aspekte und Klassen durch den gleichen Classloader geladen werden
  - Normalerweise der App-Classloader

Aspekte und Bundles

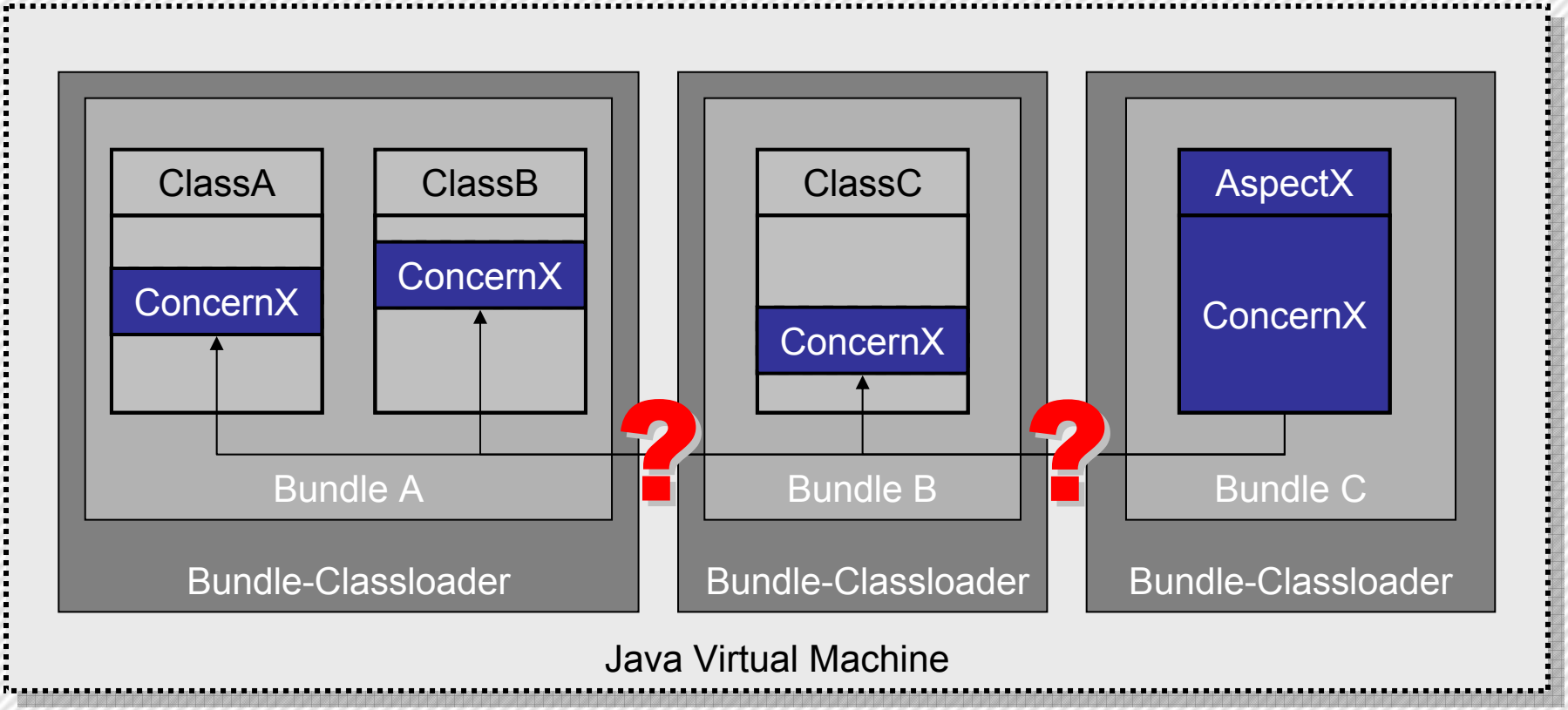
- Im Kontext von OSGi werden Klassen innerhalb von Bundles gekapselt
  - Echte Modularisierung oberhalb von Packages
- **Unser Ziel**: Aspekte in Bundles modularisieren
- Aber: Jedes Bundle hat einen eigenen Classloader

LTW und OSGi ?

- OSGi basiert auf einer ausgefeilten Class Loading Architecture.
- AspectJ Doc: *„All load-time weaving is done in the context of a class loader …“.*

Herausforderung: Wie können die beiden Ansätze zusammengebracht werden?

LTW und OSGi ?


ClassA
ConcernX
ClassB
ConcernX
Bundle A
Bundle-Classloader
?
ClassC
ConcernX
Bundle B
Bundle-Classloader
?
AspectX
ConcernX
Bundle C
Bundle-Classloader
Java Virtual Machine

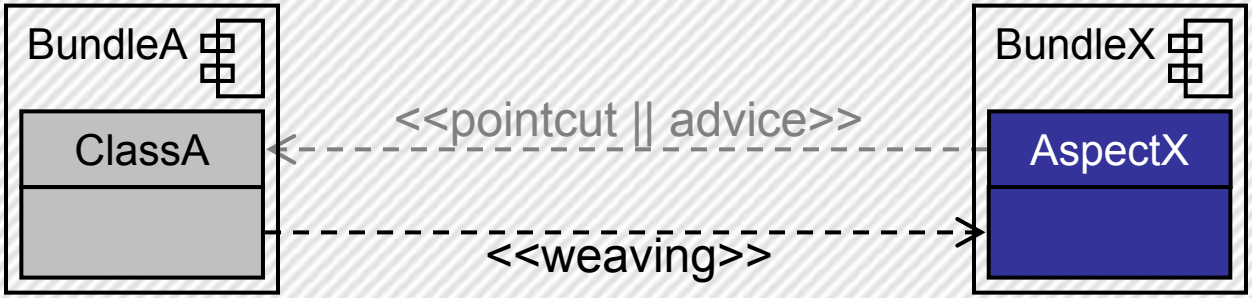
LTW in OSGi integrieren

Lösung: OSGi-Classloader erledigen das Aspect-Weaving beim Laden einer Klasse

- Aspekte werden in Bundles verpackt.
- OSGi-Runtime wird angepasst, sodass das Weaving beim Class Loading erfolgt.

Abhängigkeiten

- Weaving erzeugt neue Abhängigkeiten, die nicht im Bundle-Manifest definiert sind.



Herausforderung: Wie können die Aspekte von der Klasse aufgelöst werden?

Dynamische Abhängigkeiten

Lösung: Die Weaving-Runtime fügt die benötigten Abhängigkeiten dynamisch ein.

- Dazu muss die OSGi-Runtime entsprechend angepasst werden.

Equinox Aspects

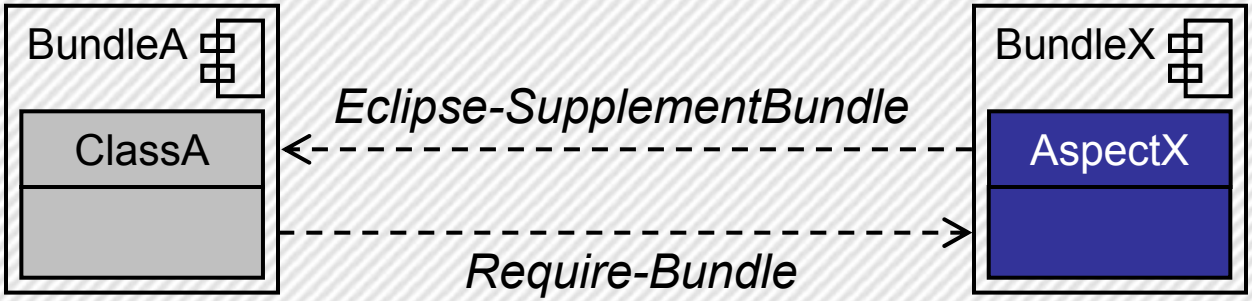
- Equinox-Incubator-Project
  - www.eclipse.org/equinox/incubator/aspects
- Framework Ext. *org.aspectj.osgi*.
- *ClassLoadingHook* bindet Weaving ein.
- *BundleFileWrapperFactoryHook* berücksichtigt die „neuen“ Abhängigkeiten.
- Weaving und Caching als OSGi-Services.

Equinox Aspects verwenden 1

- aop.xml-Dateien:
  - Nur die aufgeführten Aspekte werden beim Weaving berücksichtigt.
  - Müssen in exportierten Packages liegen.
  - Werden bekannt gegeben über: *org.aspectj.weaver.loadtime.configuration*, z.B. *…configuration=org/aspectj/aop.xml*.

Equinox Aspects verwenden 2

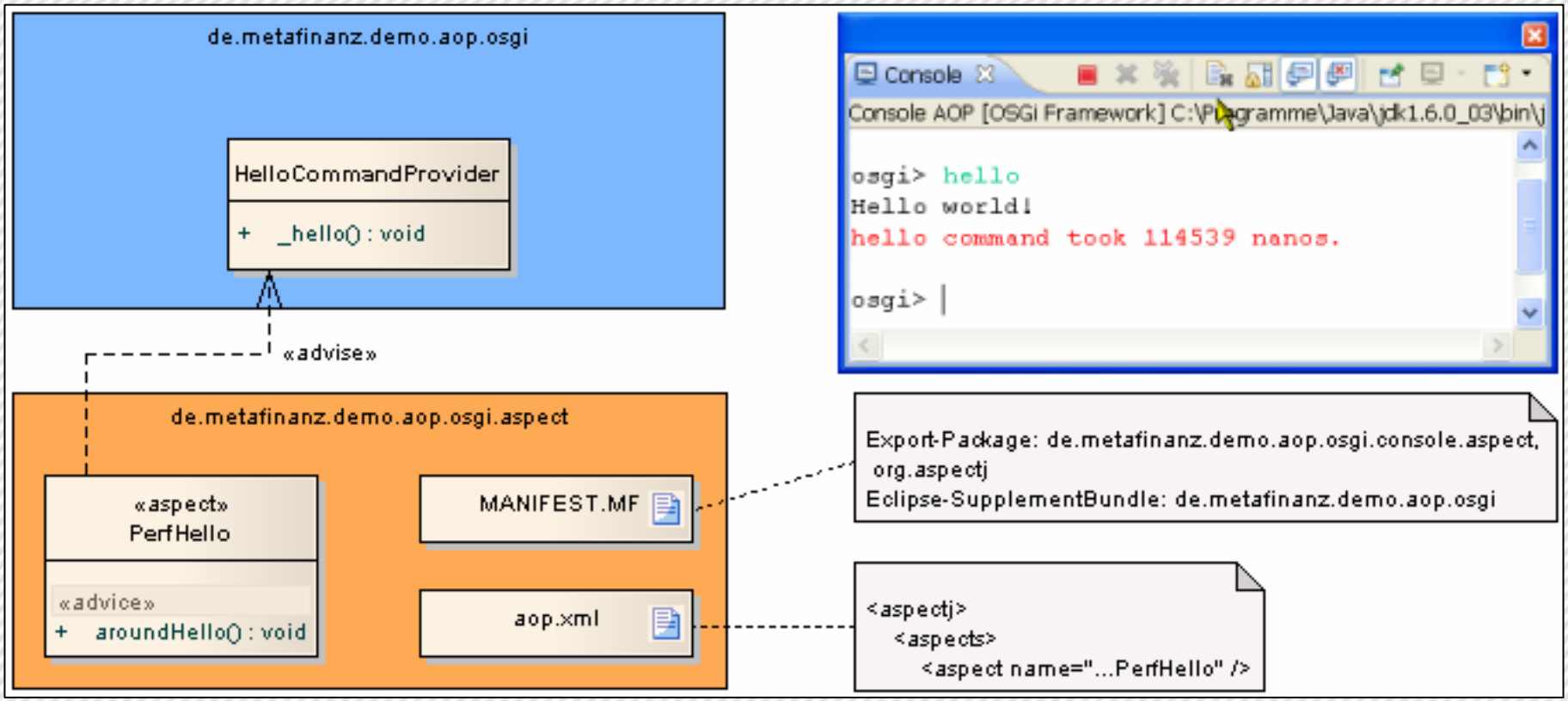
- Spezielle Manifest-Header für Aspect-Bundles, z.B. *Eclipse-SupplementBundle*.
- Aufgeführte Bundles erhalten zur Laufzeit Abhängigkeit auf das Aspect-Bundle.



Equinox Aspects verwenden 3

- Co-Location von *org.eclipse.osgi* und *org.aspectj.osgi*
- *osgi.framework.extensions=org.aspectj.osgi*

Demo: Equinox Aspects



AJEER

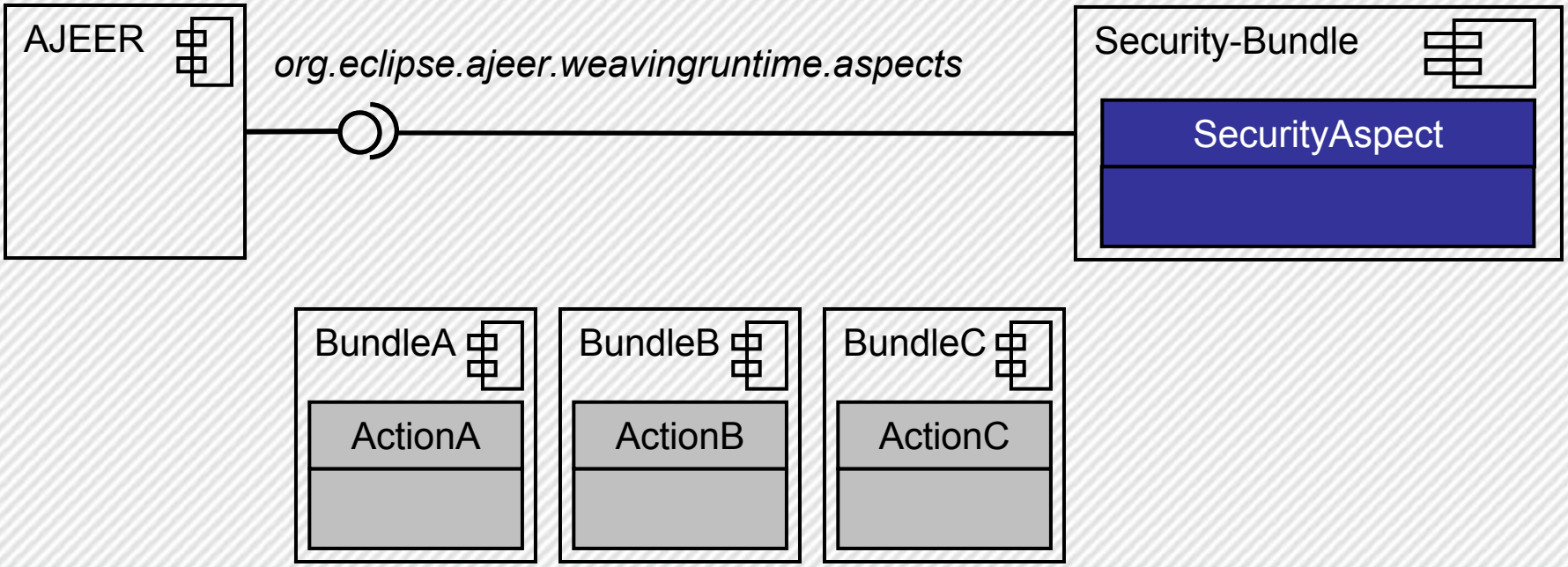
- „AspectJ-enabled Eclipse Runtime“:
  - Frühes Projekt zu AspectJ-Load-Time-Weaving für die Eclipse-Runtime
    - Gab es schon für Eclipse 2.1, jetzt bis 3.3
  - Aspekte werden per Extension-Point bekanntgemacht
- Mündet in Equinox Aspects

Anwendungsfall: Security

- Was soll erreicht werden?
  - Actions deaktivieren
  - Ausführung von Actions verhindern
- Umsetzungsmöglichkeiten
  - Eigene Basisklasse für Actions einführen
  - Capabilities
  - Aspekte nutzen

AJEER verwenden 1

- Security Aspekt in separatem Bundle
- Actions in anderen Bundles werden advised



AJEER verwenden 2

Aspekt definieren

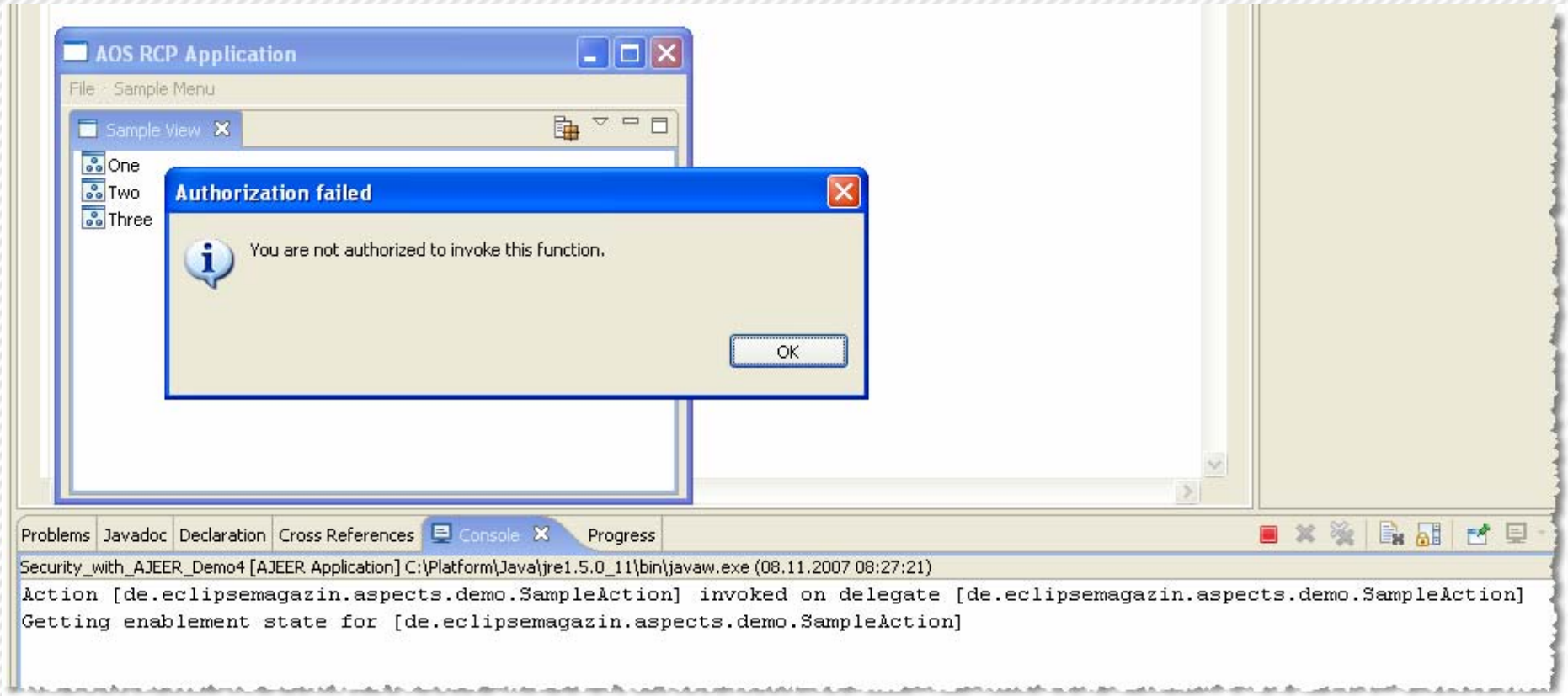
```
public aspect AuthorizationAspect {
   pointcut actionDelegateInvocation(
      IActionDelegate delegate, IAction action):
         target(delegate) && args(action) &&
         execution(void IActionDelegate.run(IAction));

   void around(IActionDelegate delegate, IAction action):
      actionDelegateInvocation(delegate, action) {
      boolean enabled = isAuthorized(function);
      if (enabled) {
         proceed();
      } else {
         MessageDialog.openInformation(…, "Authorization failed", …);
      }
   }
}
```

AJEER verwenden 3

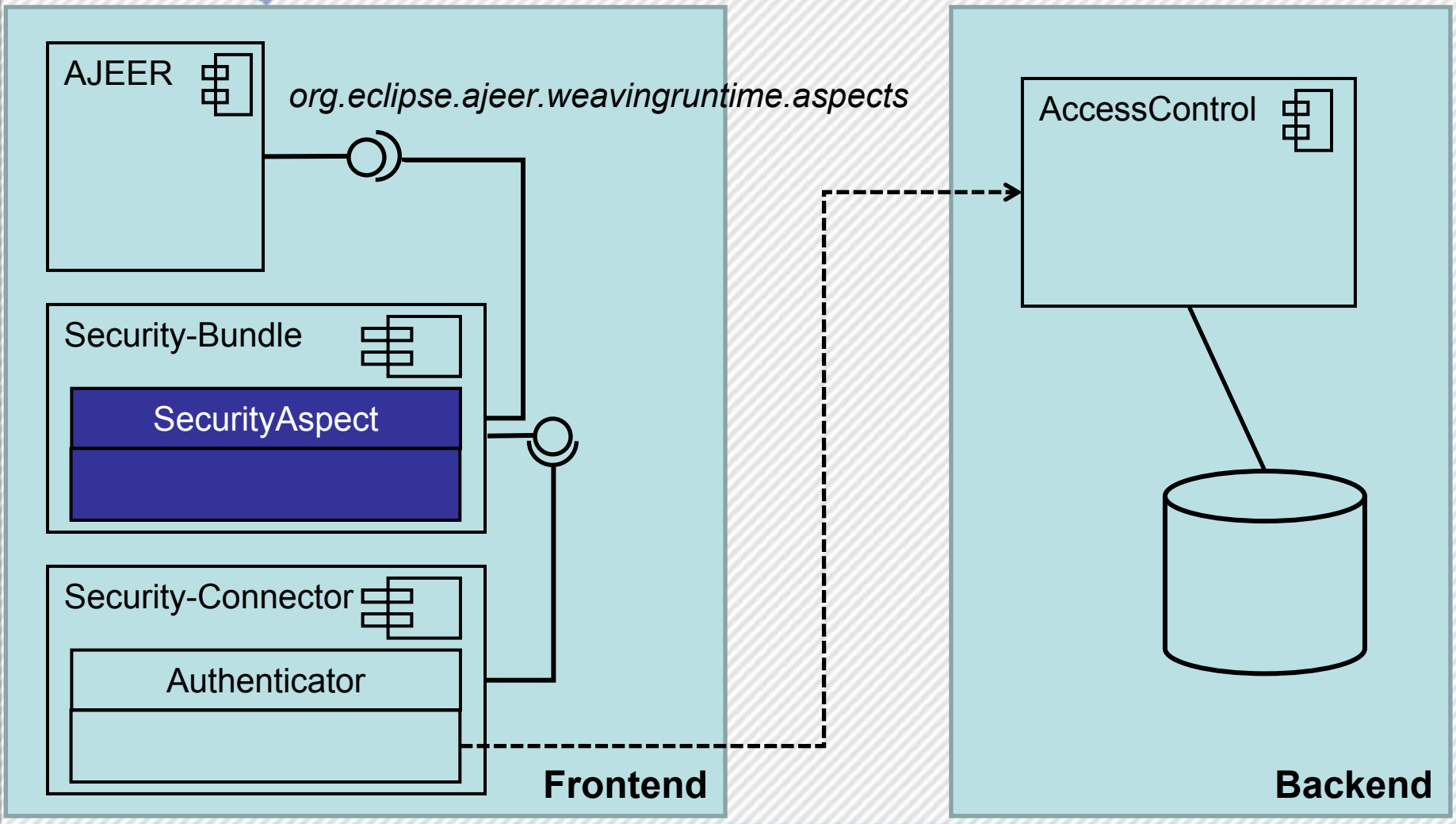
Aspekt über Extension Point anmelden

```
<extension
    point="org.eclipse.ajeer.weavingruntime.aspects">
    <aspect
        class="jax.security.AuthorizationAspect"/>
</extension>
```

Demo: AJEER Aspects



Security für Enterprise RCP



Vielen Dank...

- … für die Aufmerksamkeit!!!

Martin Lippert (martin.lippert@akquinet.de)
Peter Friese (peter.friese@gentleware.com)
Heiko Seeberger (heiko.seeberger@allianz.at)

Zusatzmaterial

- Wie genau funktionieren die Adaptor-Hooks für Equinox?

Equinox Hookable Adaptor

- Equinox ist sehr flexibel:
  - *FrameworkAdaptor* impl. oder erweitern.
  - Hookable Adaptor (seit 3.2) bietet Hooks für „die wichtigsten“ Funktionalitäten.
- Hooks verwenden:
  - Framework Extension Bundle.
  - *HookConfigurator* registriert Hooks.

Demo: Equinox Hookable Adaptor