

Umstieg auf OSGi - aber wie?

Martin Lippert & Matthias Lübken
akquinet it-agile

OSGi

“The dynamic module system for Java”

Imports Services Versionen
Bundles Dependencies Lifecycle
Exports Dynamic Reuse
Declarative Services

Ausgangssituation



Zwei Aspekte

Modularität & Dynamik

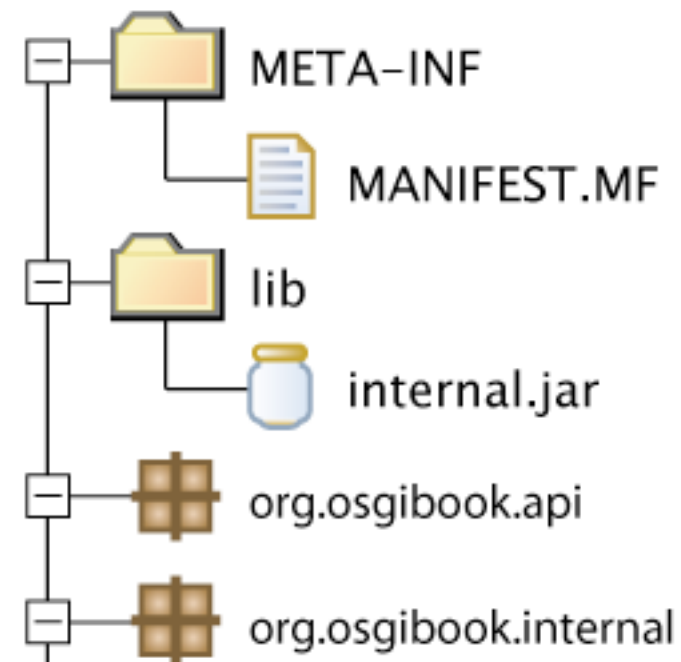
Siehe Vortrag:
"Patterns und Best Practices
für dynamische OSGi-
Applikationen"
Tödter, Wütherich
15.15 - 16.15

Agenda

➔ Bundles erstellen

- Patterns zur Modularisierung

Was ist ein Bundle



Manifest erstellen

- Per Hand
- Per PDE
- Per bnd

PDE

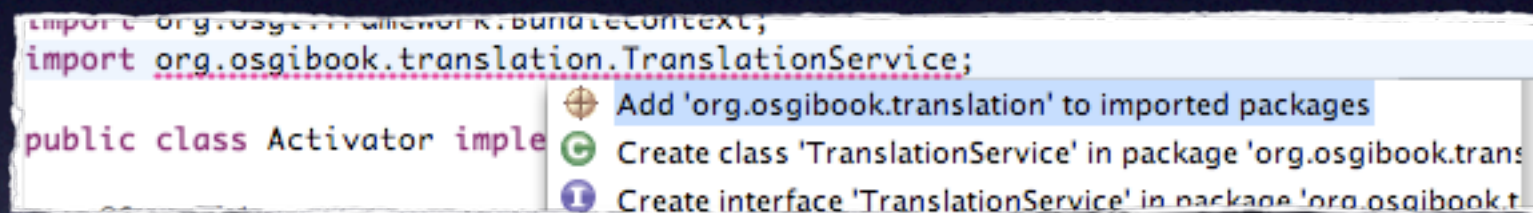
- Manifest Editor (Text & UI)
- Wizards
- Siehe auch:
Help > Plug-in Development Environment Guide

PDE Beispiel I

Quick Fixes

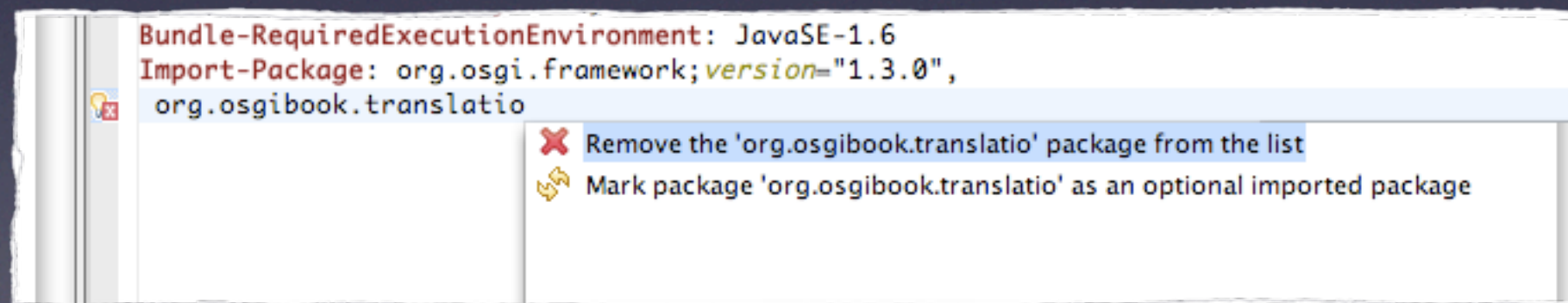
Java Source:

```
import org.osgi.framework.BundleContext;  
import org.osgibook.translation.TranslationService;  
  
public class Activator implements
```



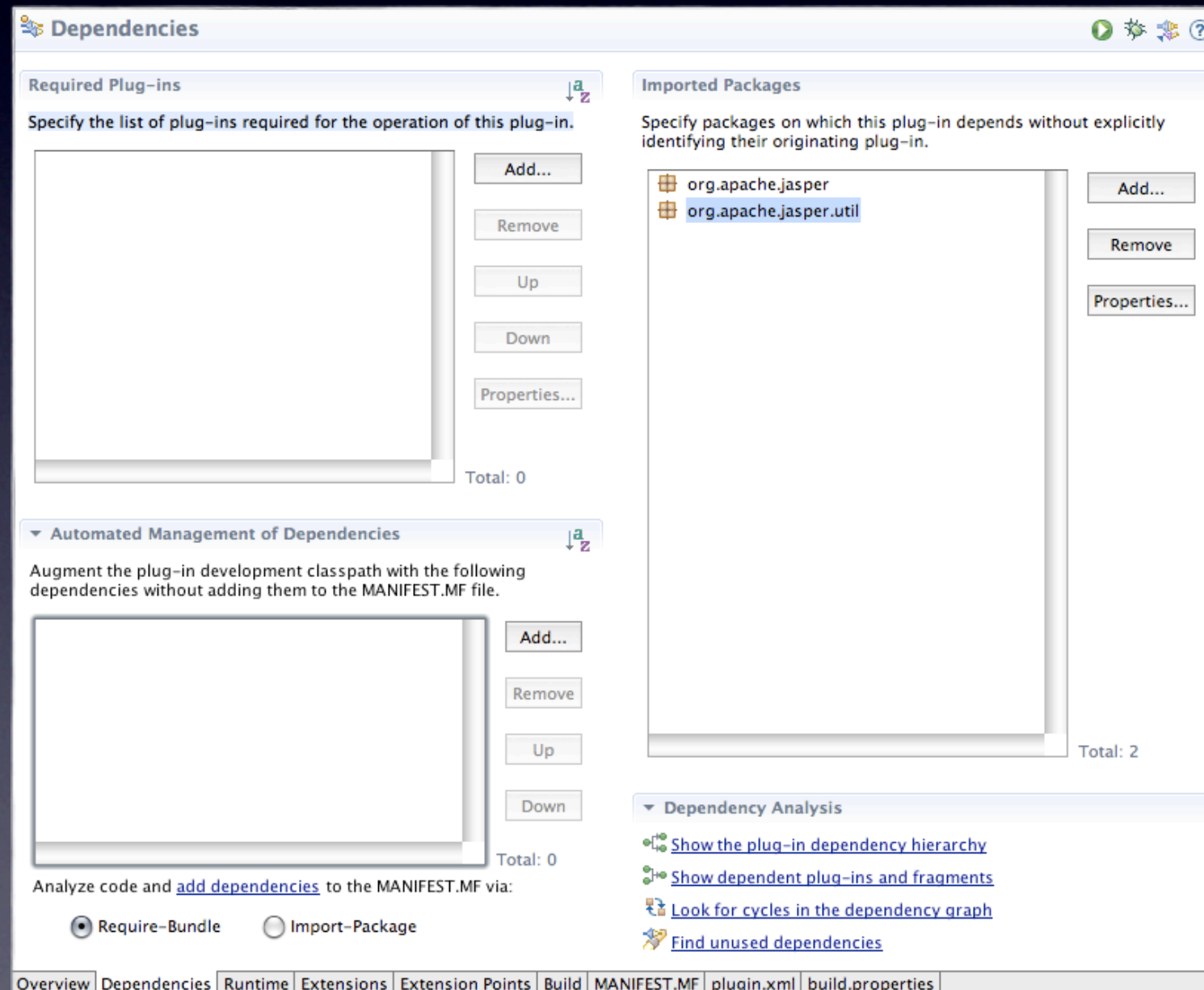
Manifest:

```
Bundle-RequiredExecutionEnvironment: JavaSE-1.6  
Import-Package: org.osgi.framework;version="1.3.0",  
org.osgibook.translatio
```



PDE Beispiel 2

Manifest Editor



PDE Beispiel 3

Wizard: Plugin from existing JAR archives

The screenshot shows the 'New Plug-in from Existing JAR Archives' wizard. The title bar reads 'New Plug-in from Existing JAR Archives'. The main heading is 'Plug-in Project Properties' with the instruction 'Enter the data required to generate the plug-in.' and a small icon of a jar with a green arrow.

Project name: org.apache.commons

☒ Use default location

Location: /Users/mdl/workspaces/wjax/org.apache.commons Browse...

Plug-in Properties

Plug-in ID: org.apache.commons

Plug-in Version: 1.0.0

Plug-in Name: Commons Plug-in

Plug-in Provider:

☒ Analyze library contents and add dependencies

Target Platform

This plug-in is targeted to run with:

☒ Eclipse version: 3.4

☐ an OSGi framework: Equinox

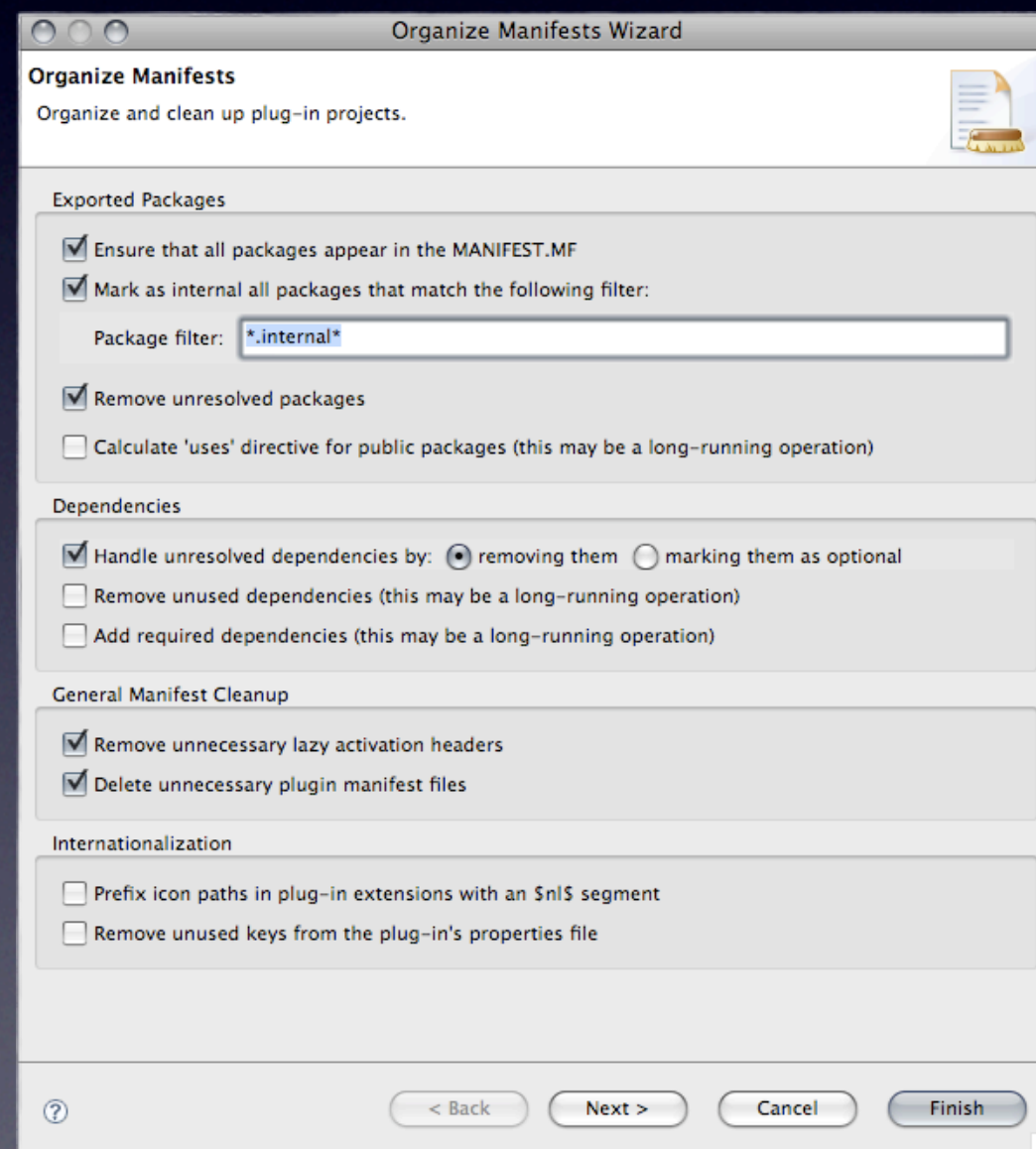
☒ Unzip the JAR archives into the project

☐ Update references to the JAR files

At the bottom, there are buttons: ? < Back Next > Cancel Finish.

PDE Beispiel 4

Wizard: Organizing Manifests



bnd

- BuNDle tool
- <http://www.aqute.biz/Code/Bnd>
- Command line, Eclipse, Maven
- Analysiert Java Source und erzeugt Manifest Einträge

bnd-Datei

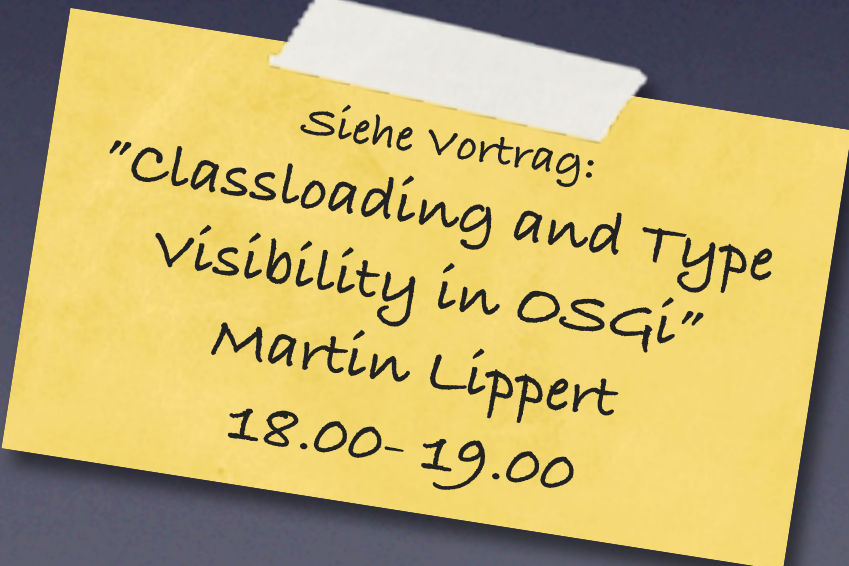
- .bnd-Datei enthält Vorgaben
- z.B.: Version, Export nur von API-Klassen
- Features wie
 - Variablen: `${version}`
 - Wildcards: `Export-Package: org.impl.*;`
 - Negativ: `Export-Package: !org.internal.*;`

Repositories

- OSGi Bundle Repository (ORB)
<http://www.osgi.org/Repository/HomePage>
- Eclipse Orbit
<http://www.eclipse.org/orbit/>
- Apache OSGified projects
<http://wiki.apache.org/commons/CommonsOsgi>
- Apache Felix Commons
<http://felix.apache.org/site/apache-felix-commons.html>
- SpringSource Bundle Repository
<http://www.springsource.com/repository/>

Classloading

- Context-Classloader
- Buddy-Classloader
- Dynamic-Imports



Siehe Vortrag:
"Classloading and Type
visibility in OSGi"
Martin Lippert
18.00-19.00

Agenda

- Bundles erstellen
- ➔ Patterns zur Modularisierung

Startstrategien

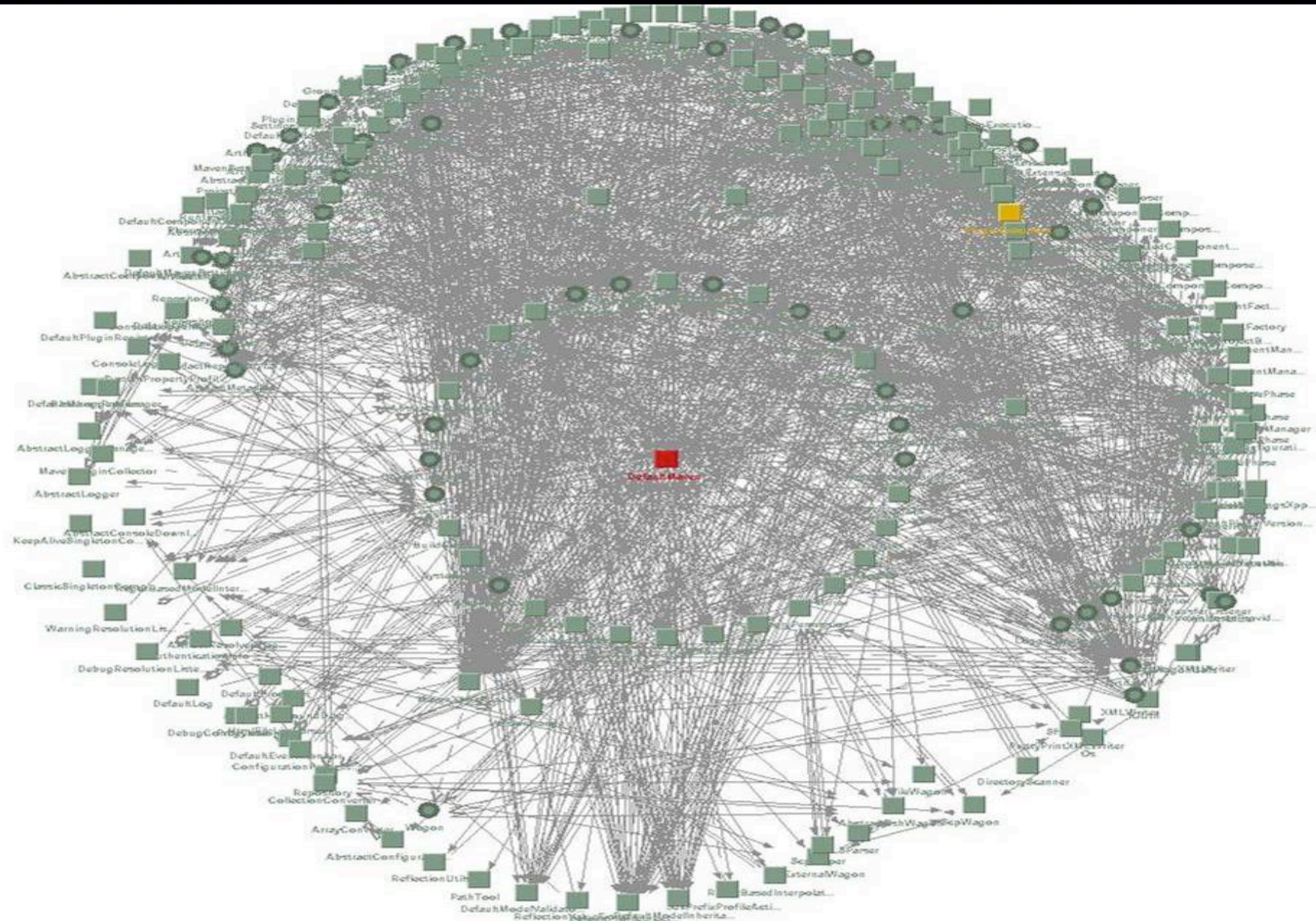
- Anwendung in ein Bundle stecken und Teile einzeln rausziehen
- Kern der Anwendung identifizieren und Module einzelnen hinzufügen

Aber mal ehrlich...

- Sieht Ihre Anwendung wirklich so aus?



Oder eher so?

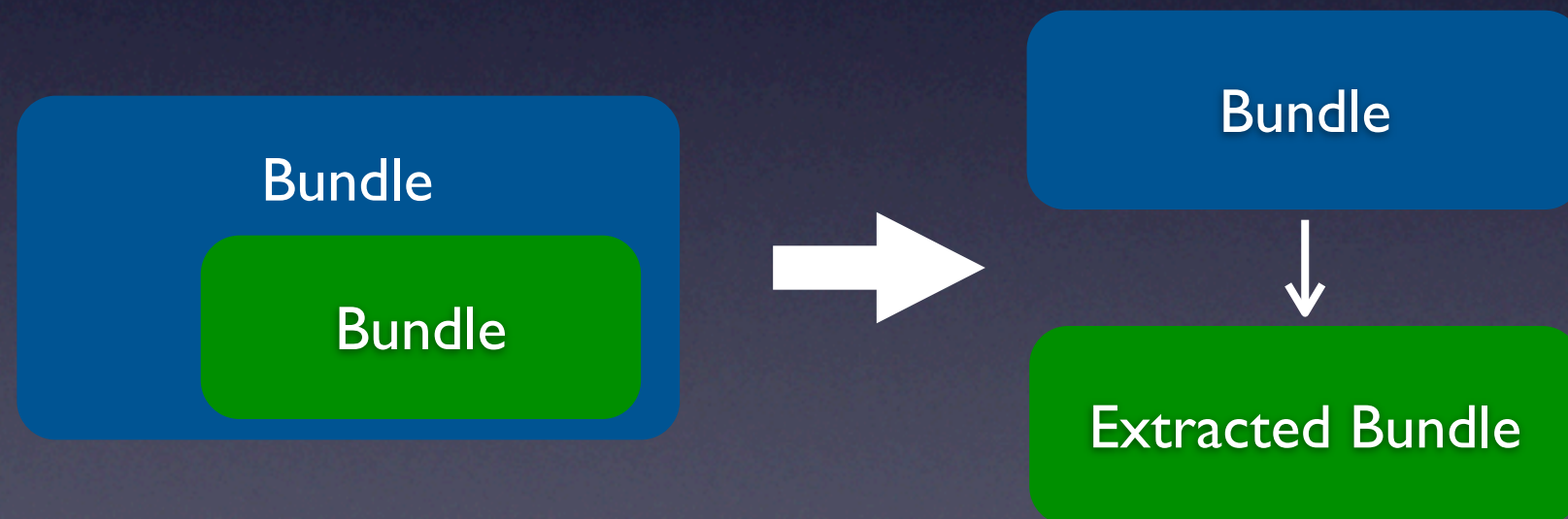


Libraries

- Libraries in eigene Bundles extrahieren
 - Versionierung
 - Klare Abhängigkeiten
 - Keine duplizierten Libs in unterschiedlichen Bundles

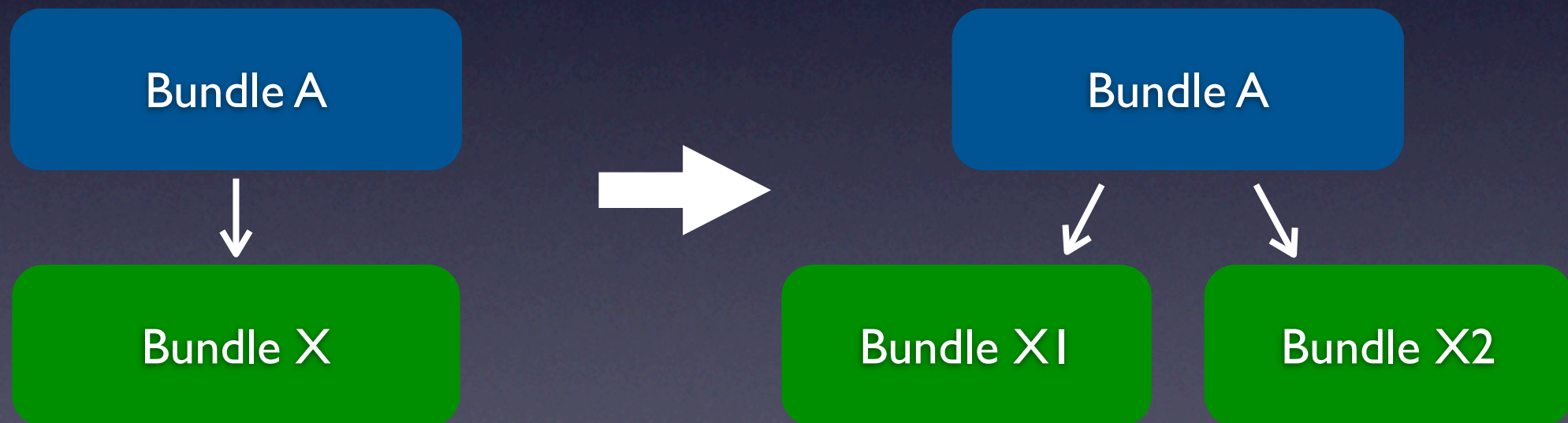
Extract Bundle

- Funktionalität aus einem existierenden Bundle herauslösen



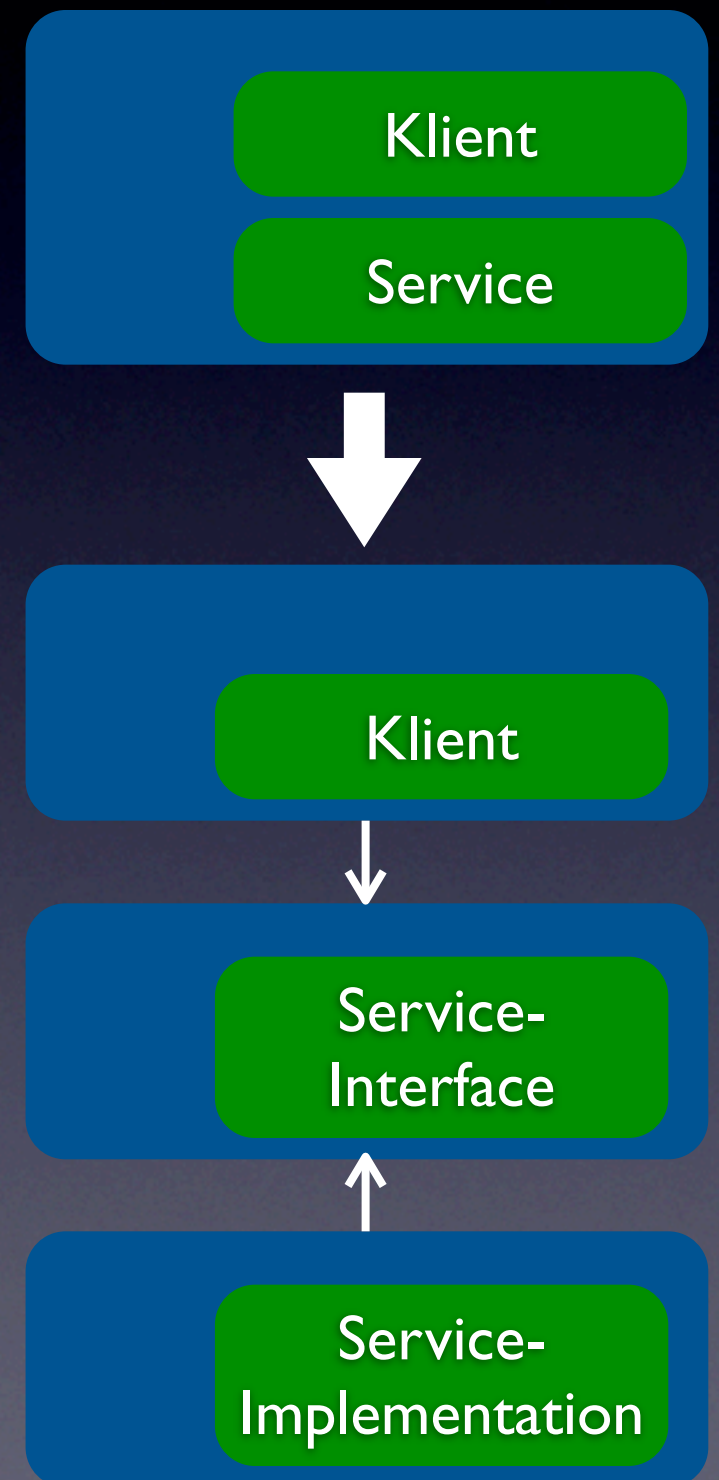
Split Bundle

- Funktionalität in zwei Bundles aufteilen



Interfaces & Services

- Einen service-artigen Teil aus einem Bundle herauslösen
 - Ein Interface einführen
 - Implementation als Service registrieren
 - Für Klienten des Services Lookup verwenden (Service-Tracker)
 - Interface und Service in separate Bundles herausziehen



Services wofür?

- Wo setze ich OSGi-Services ein?
 - Entkopplung zwischen Interface und Funktionalität (klassisch)
 - Dynamik und Austauschbarkeit
 - Re-think existing Patterns... ;-)

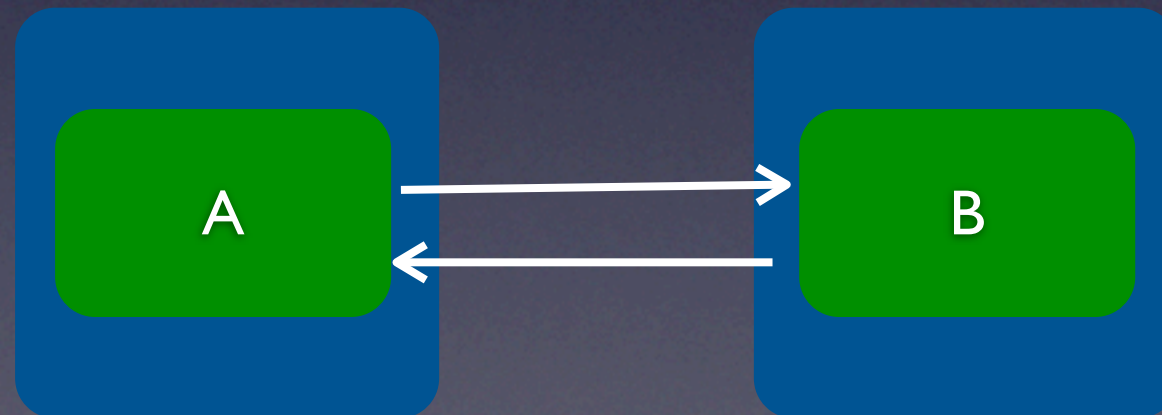
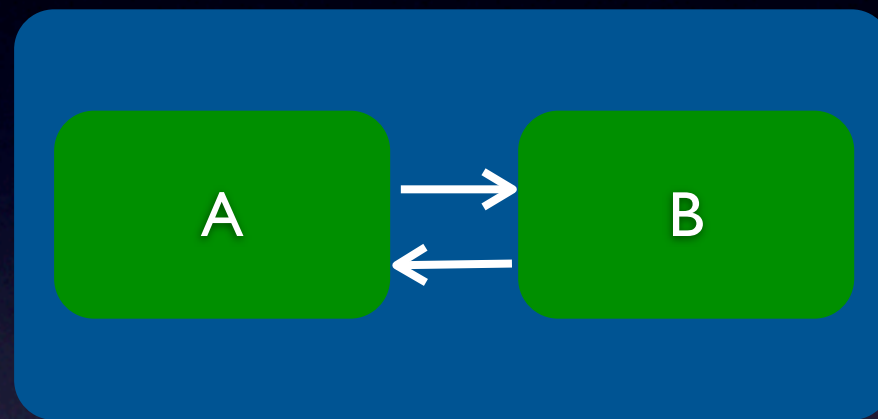
Beobachter-Muster

- Typisches Listener-Registrierungs-Modell überdenken
- Whiteboard-Pattern in Betracht ziehen
 - Event-Source ohne Register-Methoden
 - Listener als OSGi-Services
 - Event-Source benachrichtigt alle entsprechend verfügbaren OSGi-Services im Falle eines Events

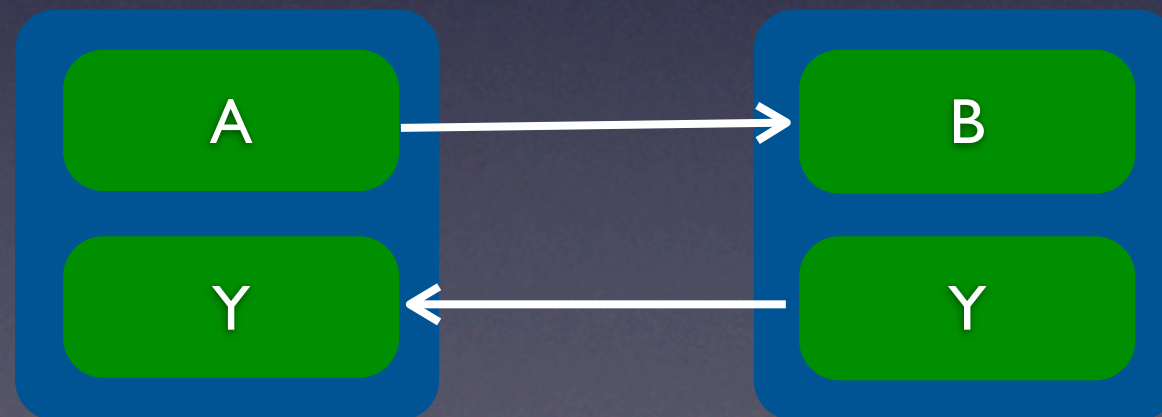
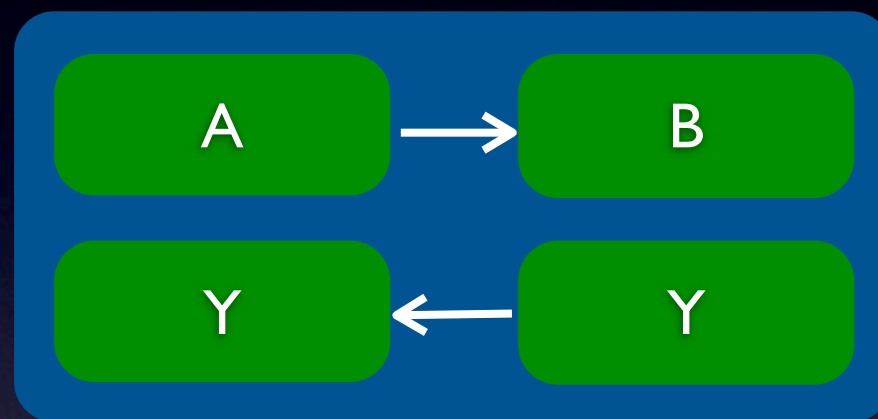
Zyklen auflösen

- Aufspalten kann Zyklen führen oder existierende Zyklen deutlich machen
- Zyklen sind evil
 - Sie waren es schon immer
- **Zyklen müssen aufgelöst werden!!!**

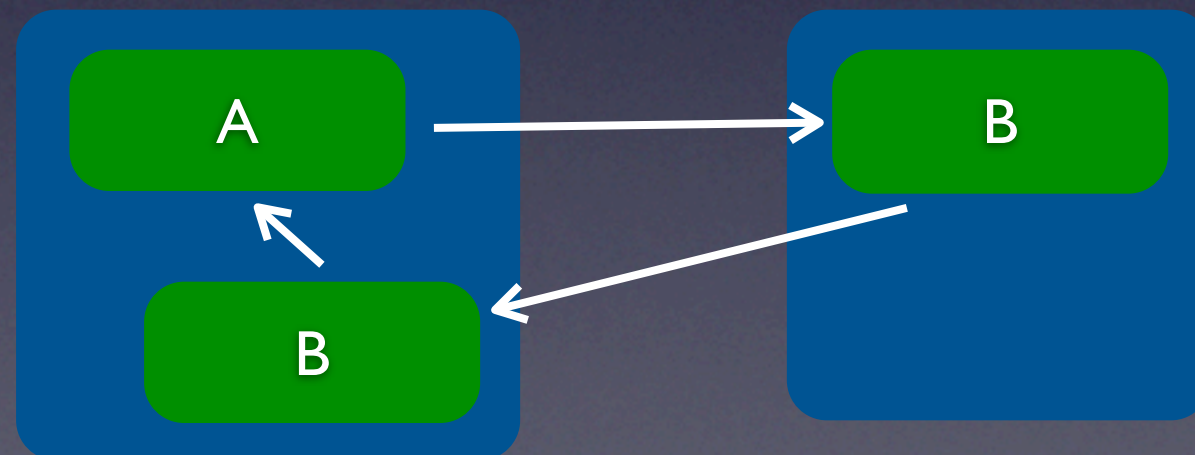
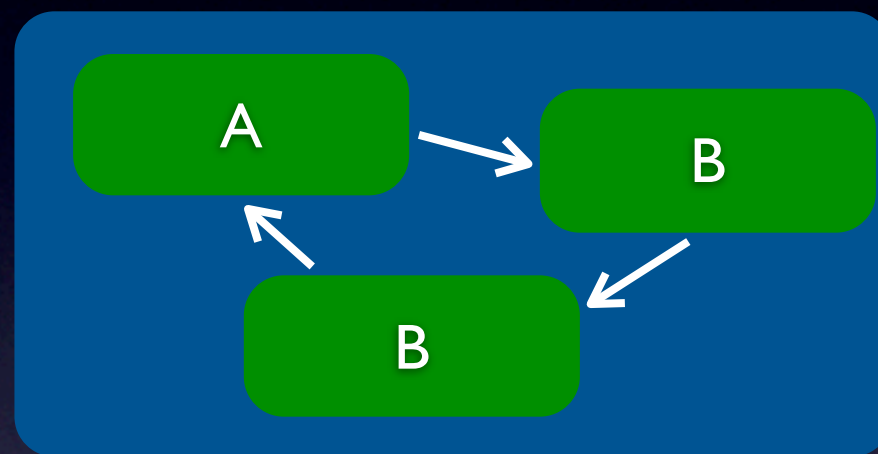
Beispiel I



Beispiel 2

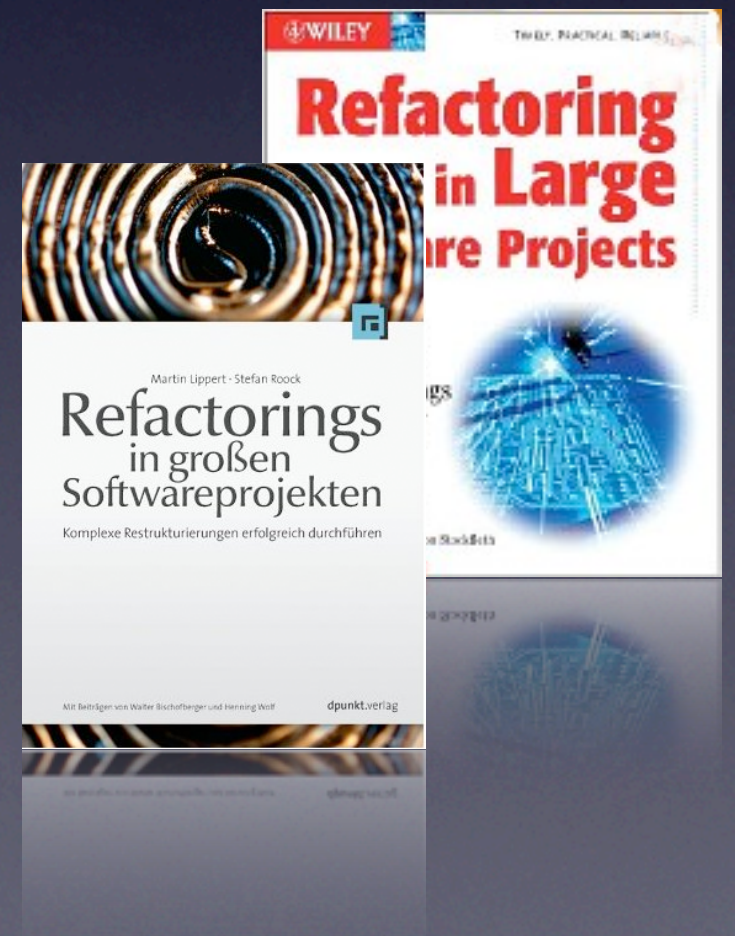


Beispiel 3



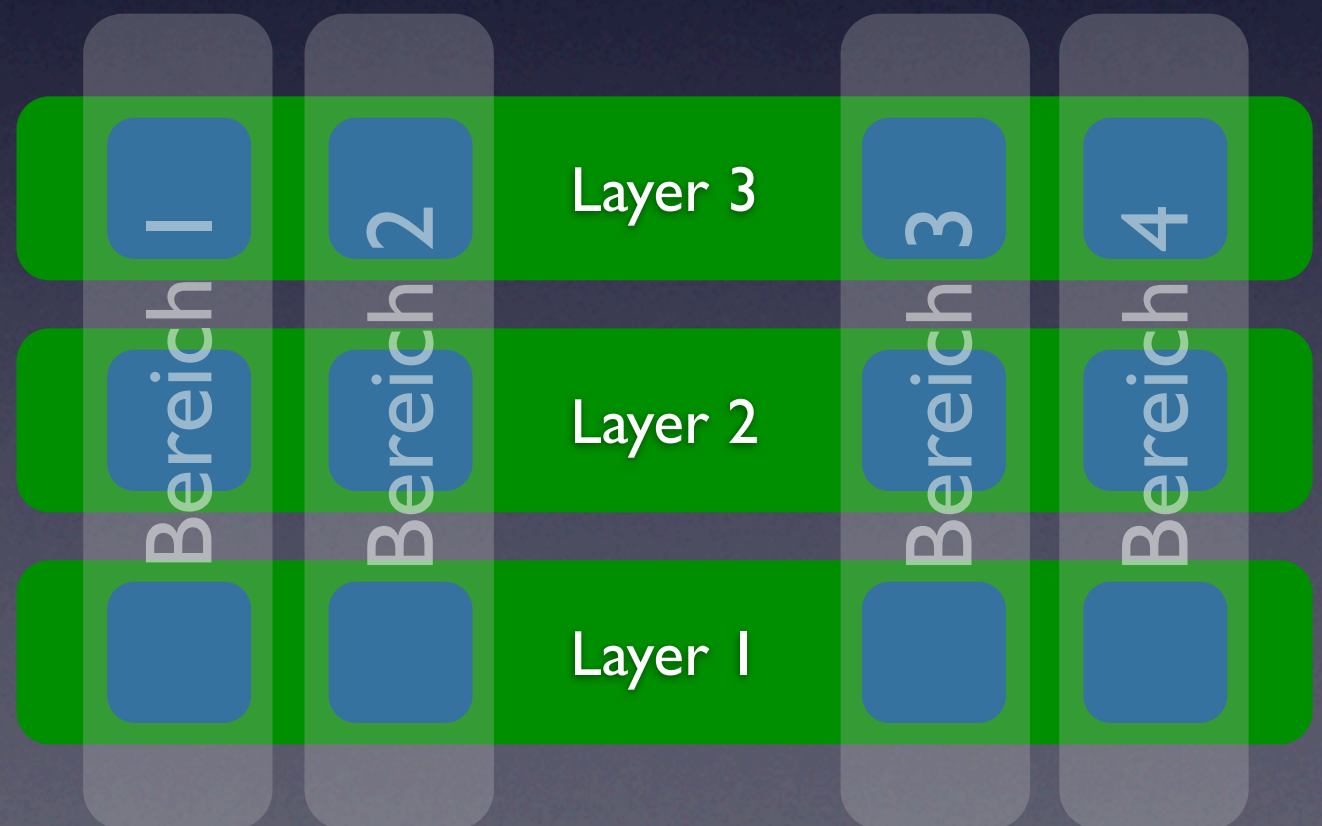
Zyklen auflösen

- Es gibt etablierte Patterns, um Zyklen aufzulösen
- z. B. dritte Komponente extrahieren



Wie teilt man Bundles auf?

- Typische horizontale Layer
- Vertikale Bereiche
- Beispiel:
 - Vorgang, VorgangUI
 - Brief, BriefUI



Typen von Anwendungen

- ✓ Client-seitige Anwendungen
- ✓ Server on OSGi
- ? OSGi inside the Server

Zusammenfassung

- Die ersten Schritte können schmerzhaft sein
- Anschließend auf OSGi aufbauen zu können ist unbezahlbar

Vielen Dank!

<http://www.it-agile.de/osgi>

