Die OSGi Alliance – die dynamische Softwareplattform auf Java-Basis

W3L AG
info@W3L.de

2008

Inhalt

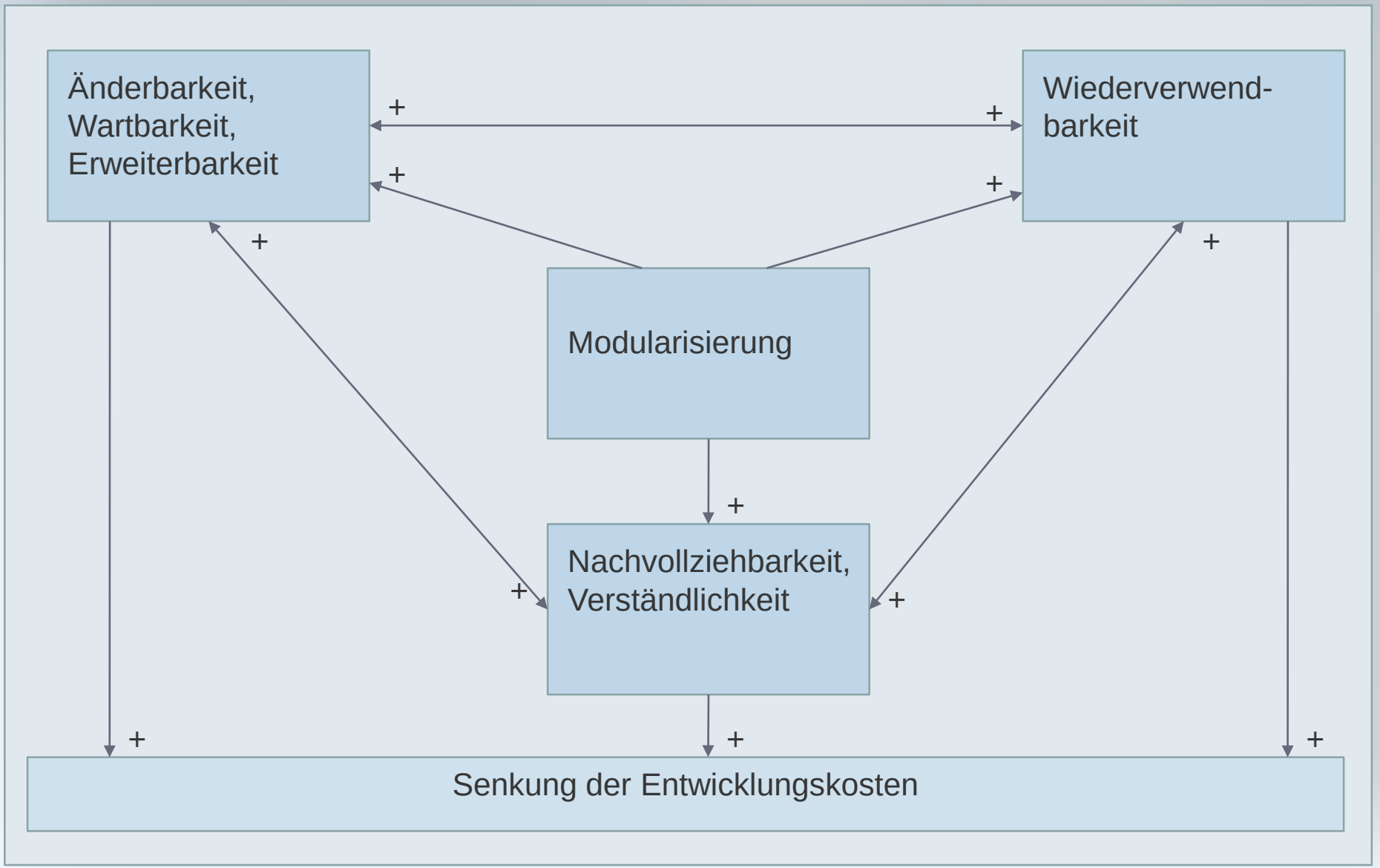
- Motivation
- OSGi
  - Überblick
  - Historische Entwicklung
  - Programmiermodell
  - Framework
- Fazit

zurück

Motivation

- **Zwickmühle der Software-Technik**
  - Software-Technik erarbeitet immer bessere Abstraktionsmechanismen...
  - Komplexität von Software nimmt immer weiter zu.
    - Kurze Produktzyklen
    - Höhere Anforderungen
    - Customizing
- **Modularisierung**
  - Altes »Hausmittel« der Software-Technik
  - Prinzip der Trennung der Belange und ihrer Lokalisierung *(principal of separation of concerns)*
  - Pioniere dieses Konzepts
    - Edsger Dijkstra
    - David Parnas
    - Niklaus Wirth

Motivation


Änderbarkeit,
Wartbarkeit,
Erweiterbarkeit
Wiederverwend-
barkeit
Modularisierung
Nachvollziehbarkeit,
Verständlichkeit
Senkung der Entwicklungskosten
+
+
+
+
+
+
+
+
+
+
+
+

Motivation

- **Objektorientierung**
  - Besitzt Modularisierungs-Mechanismen: Attribute, Operationen, Klassen, Schnittstellen, Pakete, Beziehungen, Komponenten, etc.
  - Gilt heute als Defacto-Standard.
  - Problem
    - Objektorientierung im Allgemeinen und Java im Speziellen besitzen kein Modulkonzept oberhalb von Packages.
    - Bestehende Komponentenmodelle (in Java) adressieren nur eine bestimmte Domäne und sind statisch.
  - Lösung
    - Ein neues Komponentenmodell namens **OSGi** *(Open Services Gateway Initiative)*.
    - Heute steht der Name OSGi für sich und ist keine Abkürzung mehr.

OSGi – Überblick

- **24/7-Kurzzusammenfassung**
  - In sieben Worten
    - OSGi definiert ein dynamisches Komponentenmodell für Java.
  - In 24 Sekunden
    - OSGi ist ein Framework für Java, bei dem Applikationen durch sogenannte Bundles dynamisch zusammengestellt werden können.
    - Jedes Bundle kann Dienste exportieren oder bestimmte Aktionen ausführen.
    - Da ein Bundle seine Abhängigkeiten explizit beschreibt, muss ein Container bei der Installation eines Bundles Versionskonflikte oder nicht auflösbare Abhängigkeiten erkennen.
    - Dieses Zusammenspiel zwischen Bundle und Container wird durch eine Spezifikation geregelt, so dass theoretisch irgendein OSGi Bundle innerhalb beliebiger OSGi konformer Container ausgeführt werden kann.

OSGi – Überblick

- **Eigenschaften**
  - Besitzt ein Abhängigkeitsmanagement inklusive Versionsmanagement.
  - Berücksichtigt nicht nur die Entwicklungsphase, sondern insbesondere die Laufzeitphase
    - Wie können Komponenten im laufenden Betrieb unter Einhaltung ihrer Abhängigkeiten installiert, aktualisiert oder entfernt werden?
  - Einführung eines serviceorientierten Programmiermodells
    - Komponenten können bereitgestellte Services anderer Komponenten nutzen.
    - Bereitstellung und Auffindung wird über die Infrastruktur zur Verfügung gestellt.
    - Lose Kopplung zwischen den Komponenten.

OSGi – Überblick

- **Vorteile**
  - Das Komponentenmodell ist ausgereift und in vielen Umgebungen im Einsatz:
    - Automatisierungsindustrie,
    - Mobile Endgeräte wie beispielsweise Telefone,
    - Anwendungs-Frameworks wie Spring oder Google Guice (sogenannte *Dependency Injection Frameworks*),
    - Applikationsserver wie GlassFish, IBM Websphere, Oracle/BEA Weblogic, JBoss.
  - **Global Player Commitment**
    - Ein Industriekonsortium aus über 100 großen Konzernen treibt die Standardisierung und Entwicklung voran und setzt die Technik ein.
    - Oracle, IBM, Samsung, Nokia, Motorola, Siemens, Hitachi, Deutsche Telekom, Ericsson, ...
  - Simpel und klein
    - Die OSGi-Kern-API besteht aus ca. 30 Klassen und Schnittstellen, die zusammen in einer ca. 300Kb großen Datei ausgeliefert werden.

OSGi – Überblick

- **Vorteile**
  - Unterstützt die Versionierung
    - Ähnlich der DLL-Hölle unter Windows gibt es eine vergleichbare JAR-Hölle.
    - **Beispiel:** Eine Bibliothek A benötigt eine Bibliothek B in der Version 1. Eine Bibliothek C benötigt ebenfalls die Bibliothek B, aber in der Version 2. Beide Bibliotheken A und B sollen nun gleichzeitig in einer Anwendung verwendet werden.
    - Java bietet zu diesem Problem keine Lösung!
    - In einer OSGi-Umgebung können diese Arten der Abhängigkeiten deklariert werden.
    - Ein ausgeklügeltes System von ClassLoadern unterstützt das Laden von verschiedenen Versionen einer Java-Klasse und ihrer Bindung!
  - Dynamische Updates
    - Das Komponentenmodell ist dynamisch!
  - ...

OSGi – Historische Entwicklung

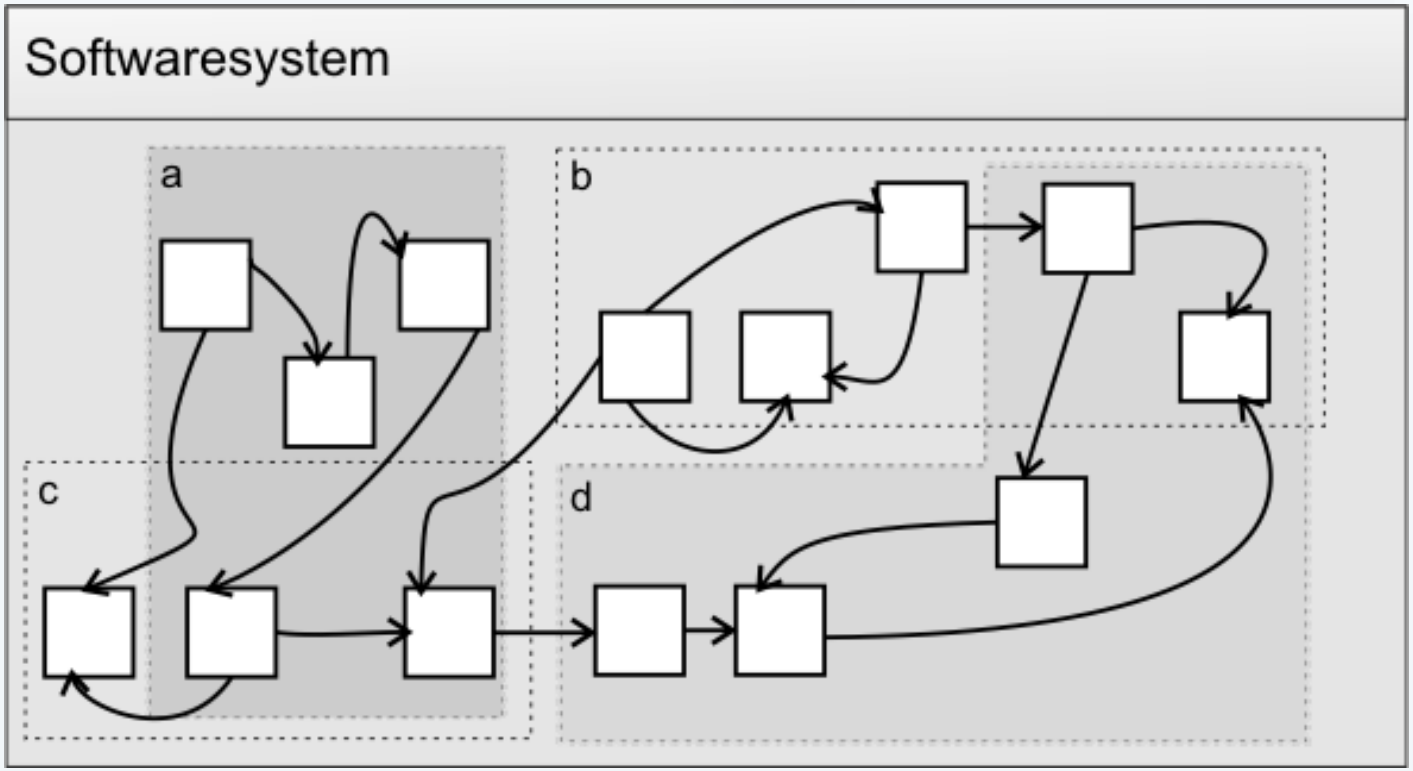
- **Hintergrund**
  - Die Firmen IBM, Oracle, Sun Microsystems und Ericsson haben im Dezember 1998 erste Idee für eine komponentenbasierte Software-Architektur für mobile Endgeräte.
  - Im März 1999 Gründung der non-profit **OSGi Alliance**, einer Organisation mit dem Ziel, eine Spezifikation für mobile Endgeräte und eingebettete Systeme zu erarbeiten, bei denen Dienste zur Laufzeit ausgerollt, ausgetauscht und verwaltet werden können.
  - Versionen
    - OSGi Release 1 (R1): Mai 2000
    - OSGi Release 2 (R2): Oktober 2001
    - OSGi Release 3 (R3): März 2003
    - OSGi Release 4 (R4): Oktober 2005 – September 2006
    - OSGi Release 4.1 (R4.1): May 2007 (JSR-291)

OSGi – Historische Entwicklung

- **OSGi und JCP**
  - Inzwischen als »JSR 291 *(Java Specification Request)*: *Dynamic Component Support for Java SE*« im Rahmen des JCP *(Java Community Process)* als offizielles dynamisches Komponentenmodell für Java angenommen (`http://www.jcp.org/en/jsr/detail?id=291`).
  - Weitere Relevante JSR's sind
    - JSR 277: Java Module System
    - JSR 232: Mobile Operational Management
    - JSR 246: Device Management API

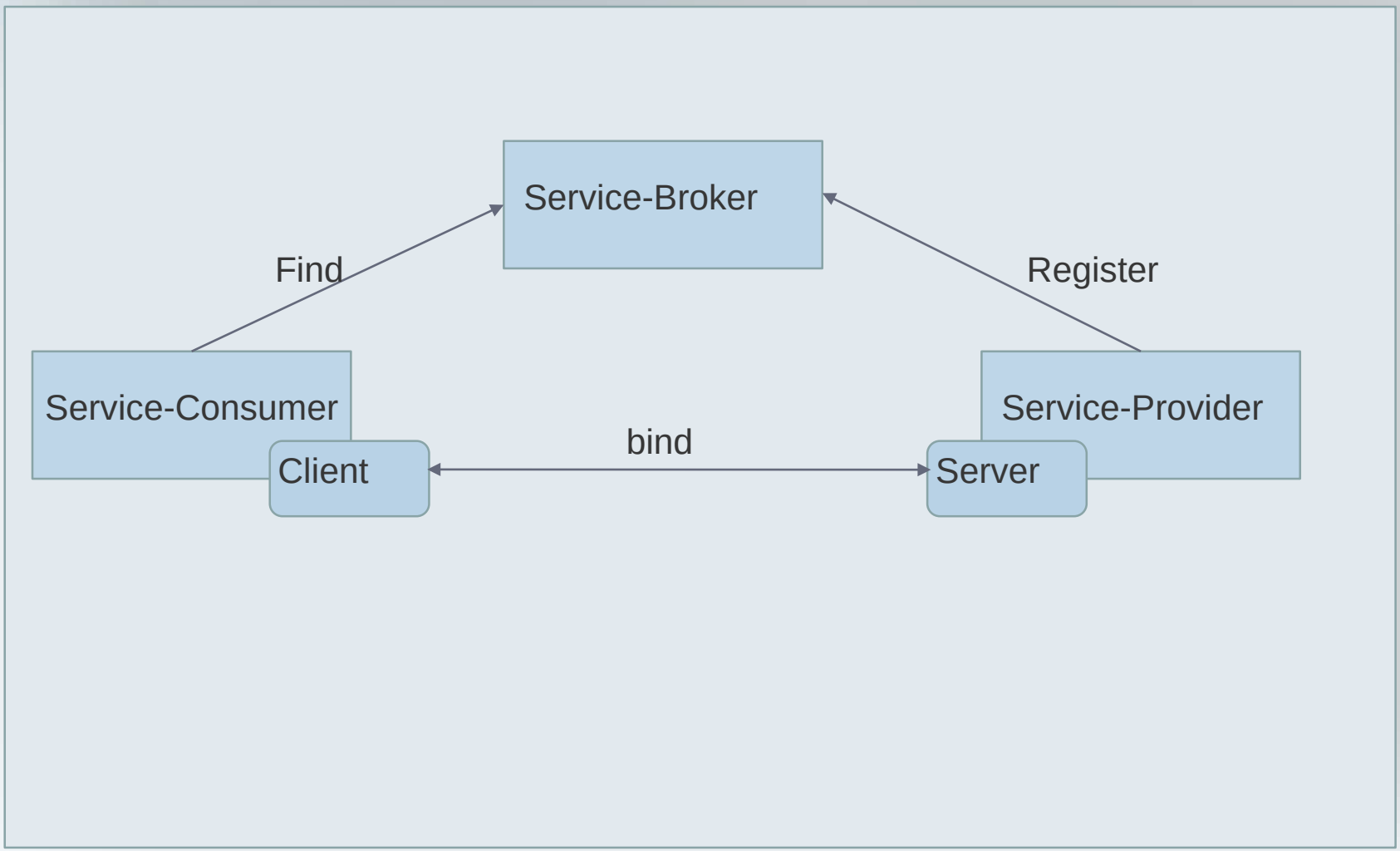
OSGi – Programmiermodell

- **Konventionelles Modell eines Softwaresystems**
  - Java-Klassen, die durch unterschiedliche JAR-Dateien angeboten werden.
  - Implizite Abhängigkeiten untereinander mit
  - einer starren und engen Kopplung.



OSGi – Programmiermodell

- **Whiteboard Pattern / Entkopplung durch Serviceorientierung**



OSGi – Framework

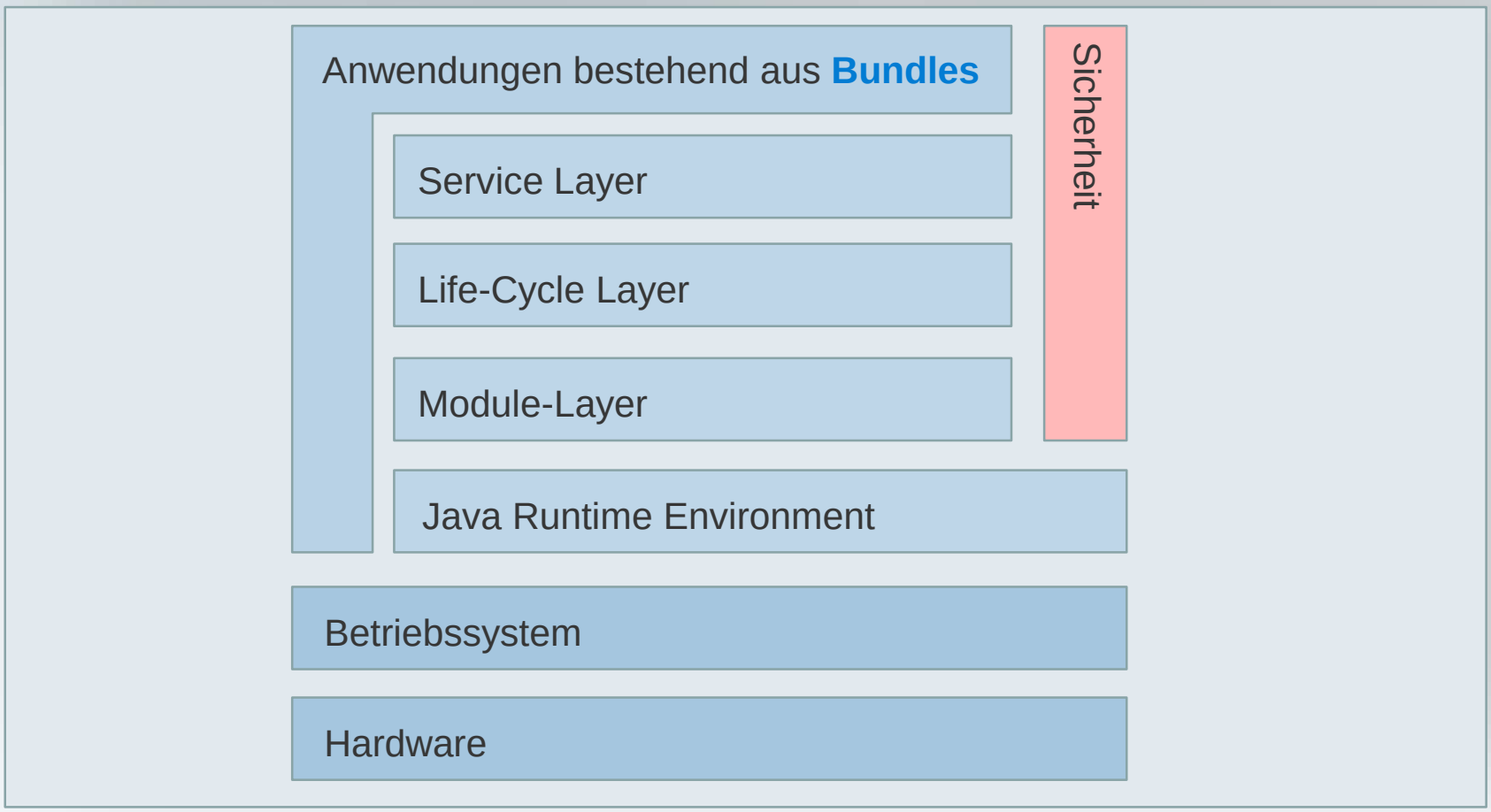
- **Die OSGi Alliance stellt**
  - die Spezifikation,
  - eine Referenzimplementierung,
  - Test-Werkzeuge für die Standardkonformitätsprüfung zur Verfügung
  - und ist für die Zertifizierung der Mitglieder verantwortlich.
- **Die Referenzimplementierung ist nicht für den Produktiveinsatz bestimmt!**
- **Solche OSGi-Frameworks sind von verschiedenen Anbietern erhältlich**
  - Kommerzielle OSGi-Frameworks
    - In der Regel zertifiziert und anwendungsbezogen. D.h. sie beinhalten neben dem Framework auch spezifische Komponenten.
  - Open Source OSGi-Frameworks
    - Nicht zertifiziert.
    - Zum Teil anwendungsbezogen, zum Teil aber auch nicht.
    - Eclipse Equinox
    - Apcahe Felix

OSGi – Framework

- **Zentrale Dekompositionseinheit**
  - Ein Framework, welches den OSGi Standard implementiert, stellt eine Umgebung zur Modularisierung von Anwendungen in sogenannte **Bundles** zur Verfügung.
  - Ein Bundle ist somit der zentrale Modularisierungsmechanismus der OSGi!
  - Jedes Bundle
    - ist in sich abgeschlossen,
    - beherbergt eine Menge dynamisch nachladbarer Java-Klassen und sonstiger Ressourcen (beispielsweise Grafiken),
    - definiert explizit seine externen Abhängigkeiten,
    - deklariert auf Paketebene, welche Klassen und Schnittstellen in welcher Version bereitgestellt bzw. exportiert werden,
    - wird in einem JAR-Archiv ausgeliefert.

OSGi – Framework

- **Schichtenarchitektur**



OSGi – Framework

- **Schichtenarchitektur**
  - Service Layer
    - Ermöglicht die lose, dynamische Kopplung von POJO.
    - Enthält Dienste zum Auffinden, Veröffentlichen und Binden von Diensten.
  - Life-Cycle Layer
    - Definiert den Lebenszyklus von Bundles und stellt ein API zur Verfügung, um Bundles zu installieren, zu deinstallieren, zu starten und zu stoppen, sowie zu aktualisieren.
  - Module Layer
    - Beschreibt die statische Sicht auf die Bundles.
    - Lädt ein Bundle und überprüft die Abhängigkeiten.
  - Sicherheit
    - Unterstützt Funktionen wie das Signieren von Bundles.
    - OSGi spezifische Autorisation.

OSGi – Framework

- **API-Klassen**
  - `BundleContext`
    - Der gesamte Zugriff auf die OSGi Infrastruktur läuft über diese Klasse.
    - Bietet Operationen, um Bundles zu registrieren oder zu finden.
  - `BundleActivator`
    - Besitzt die Operationen `start()` und `stop()`, welche beim Starten und Stoppen von Bundles durch das OSGi Framework aufgerufen werden.
    - In beiden Operationen wird ein `BundleContext`-Objekt übergeben.

OSGi – Framework

- **Beispiel Bundle**
  - Activator

```
package testequinox;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

public class Activator implements BundleActivator {

  public void start(BundleContext context) throws Exception {
    System.out.println("Mein Bundle startet.");
  }

  public void stop(BundleContext context) throws Exception {
    System.out.println("Mein Bundle wird beendet");
  }
}
```

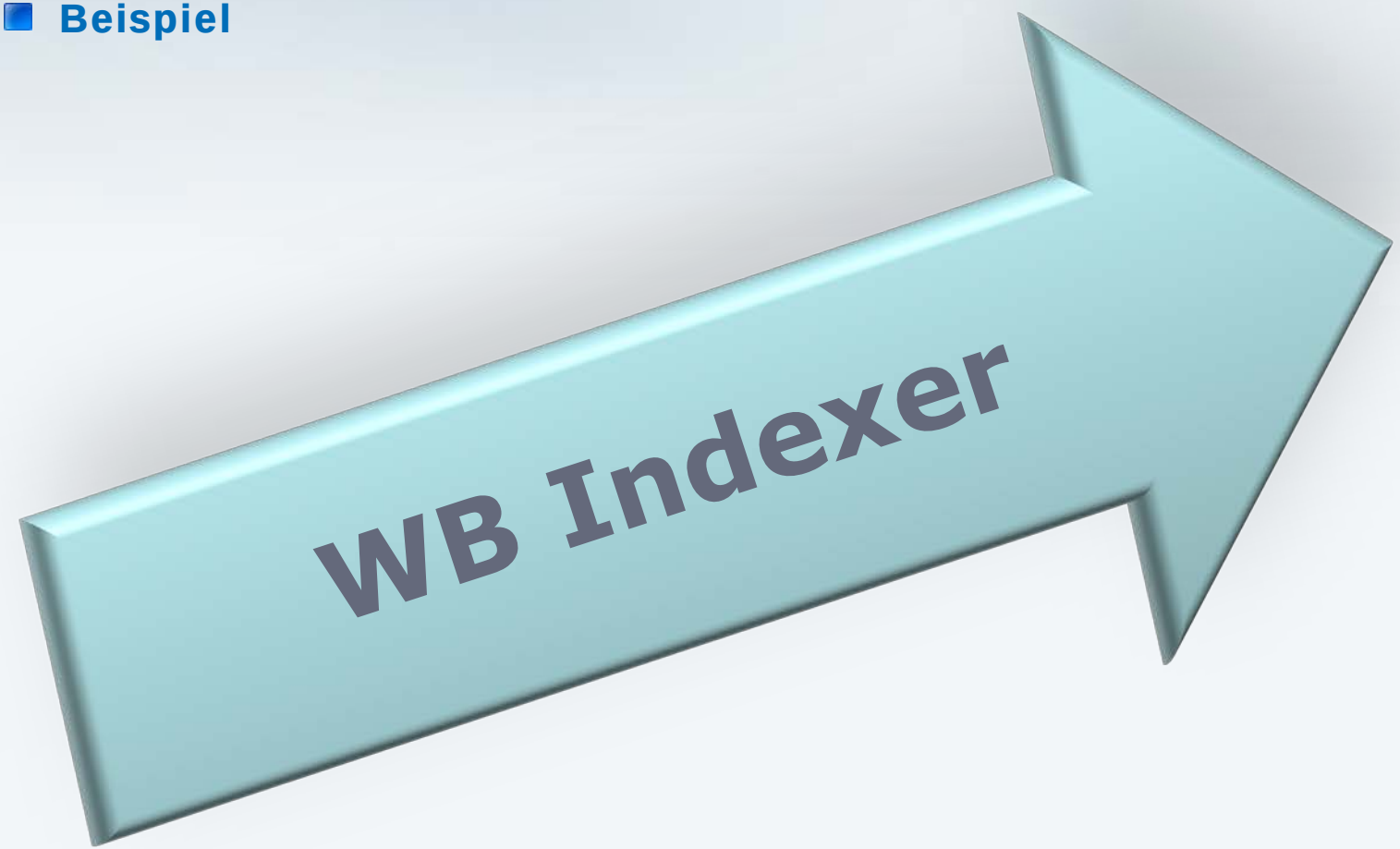
OSGi – Framework

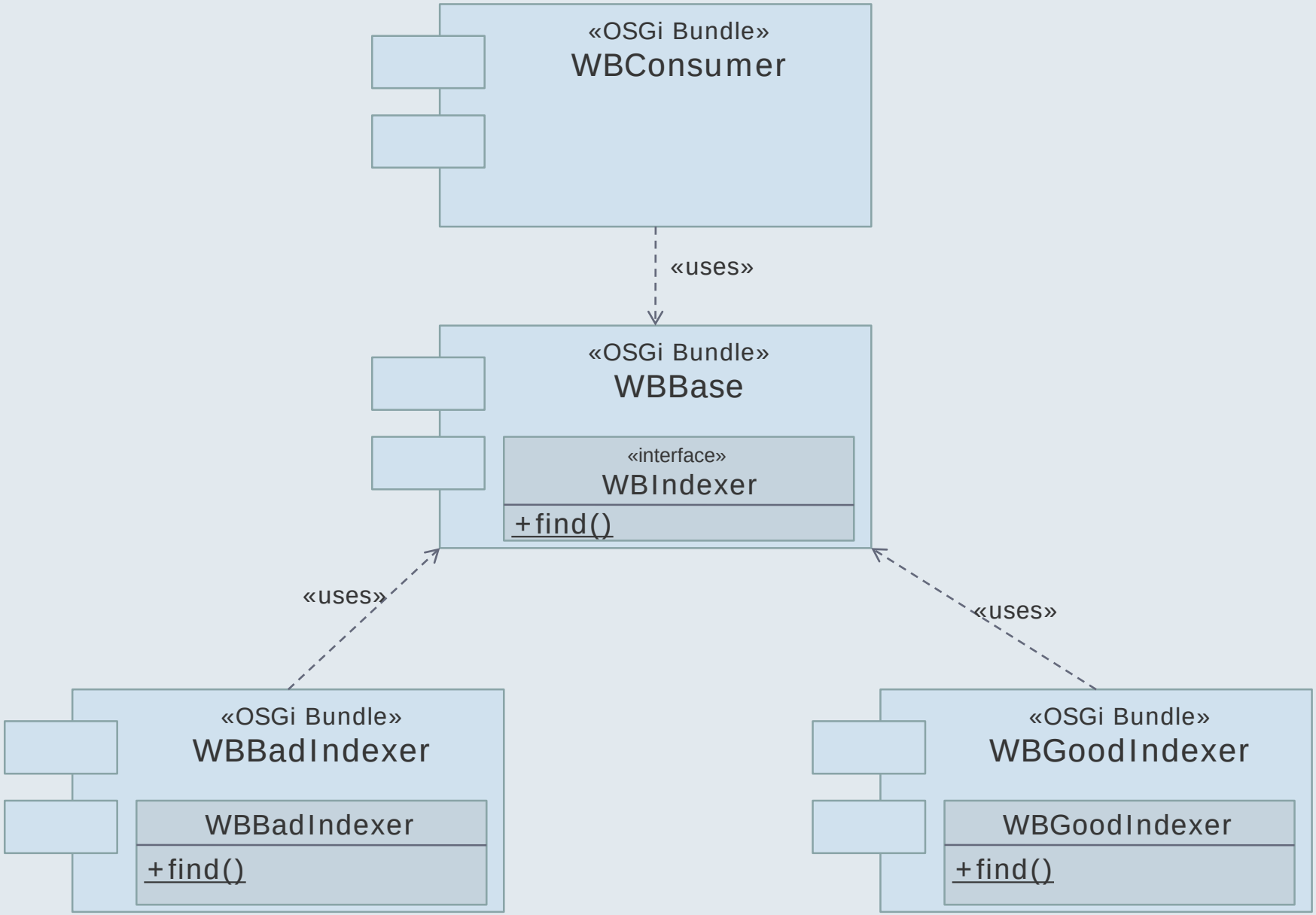
- **Beispiel Bundle**
  - Manifest

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: TestEquinox Plug-in
Bundle-SymbolicName: TestEquinox
Bundle-Version: 1.0.0
Bundle-Activator: testequinox.Activator
Import-Package: org.osgi.framework;version="1.3.0"
```

OSGi – Framework

- Beispiel




«OSGi Bundle»
WBConsumer
«uses»
«OSGi Bundle»
WBBase
«interface»
WBIndexer
+find()
«uses»
«uses»
«OSGi Bundle»
WBBadIndexer
WBBadIndexer
+find()
«OSGi Bundle»
WBGoodIndexer
WBGoodIndexer
+find()

Fazit

- **OSGi Alliance**
  - Garantiert einen offenen und weltweit akzeptierten Standard.
  - Das Global Player Commitment garantiert Investitionssicherheit.
- **OSGi Standard**
  - Offene, serviceorientierte, dynamische Software-Plattform.
  - Bereits in vielen Projekten im Einsatz: Eclipse, JBoss, GlassFish, etc.
  - Empfehlenswertes dynamisches Komponentenmodell!

Vielen Dank!

Inhouse-Schulungen

Wir bieten Inhouse-Schulungen und Beratung durch unsere IT-Experten und –Berater.

Schulungsthemen

- Softwarearchitektur (OOD)
- Requirements Engineering (OOA)
- Nebenläufige & verteilte Programmierung

Gerne konzipieren wir auch eine individuelle Schulung zu Ihren Fragestellungen.

Sprechen Sie uns an!
Tel. 0231/61 804-0, info@W3L.de

W3L-Akademie

In Zusammenarbeit mit der Fachhochschule Dortmund bieten wir

zwei Online-Studiengänge

- B.Sc. Web- und Medieninformatik
- B.Sc. Wirtschaftsinformatik

und 7 Weiterbildungen im IT-Bereich an.

Besuchen Sie unsere Akademie!
http://Akademie.W3L.de