OSGi im Embedded Bereich

Service Kompendium und Best Practices

Michael Grammling - M.Sc. Dipl.-Inform. (FH)

traffic mobility logistics.

PTV AG – Planung Transport Verkehr

- **Geschäftsfelder**
  - **Verkehrsplanung und –Management**
  - **Navigationslösungen und Telematik**
  - **Logistik- und Flottenplanung**
  - **Forschung**
- **Bekannte Produkte**
  - **Falk Navigator**
  - **PC-Routenplaner**
  - **http://www.reiseplanung.de**

Zu meiner Person

- **geschäftlich: michael.grammling@ptv.de**
- **privat: michael.grammling@sertos.de**

- **Softwareentwickler für den embedded und mobilen Bereich**
- **Erfahrungen mit OSGi seit 2003 (R2 – R4.1)**
- **Forschungsprojekte, Konsortien und kommerzielle Projekte**
  - **OSGi VEG (Vehicle Expert Group)**
  - **(EU funded) GST/EFCD Konsortium (EFCD Architektur und Implementierung)**
  - **(EU funded) ASK-IT Konsortium (OSGi-Consulting, Fußgängernavigation)**
  - **(EU funded) FeedMAP (Kartenfehlerdetektion)**
  - **(EU funded) CVIS Konsortium**
  - **Mautsysteme**

Die OSGi Alliance

- **OSGi steht für die Open Services Gateway Initiative**
- **wurde im März 1999 gegründet**
- **besteht derzeit aus mehr als 30 Firmen**
- **Mobile Operational Management (2003) – JSR232**
- **Webseite: http://www.osgi.org**

Zielsetzung:

Framework, das dynamische Installationen von Service-Applikationen über eine Remote-Schnittstelle zur Laufzeit ermöglicht.

► **Java als Technologiegrundlage**

Die OSGi Alliance

- **OSGi steht für die Open Services Gateway Initiative**
-
-
-
-
-

Korrektur – OSGi:

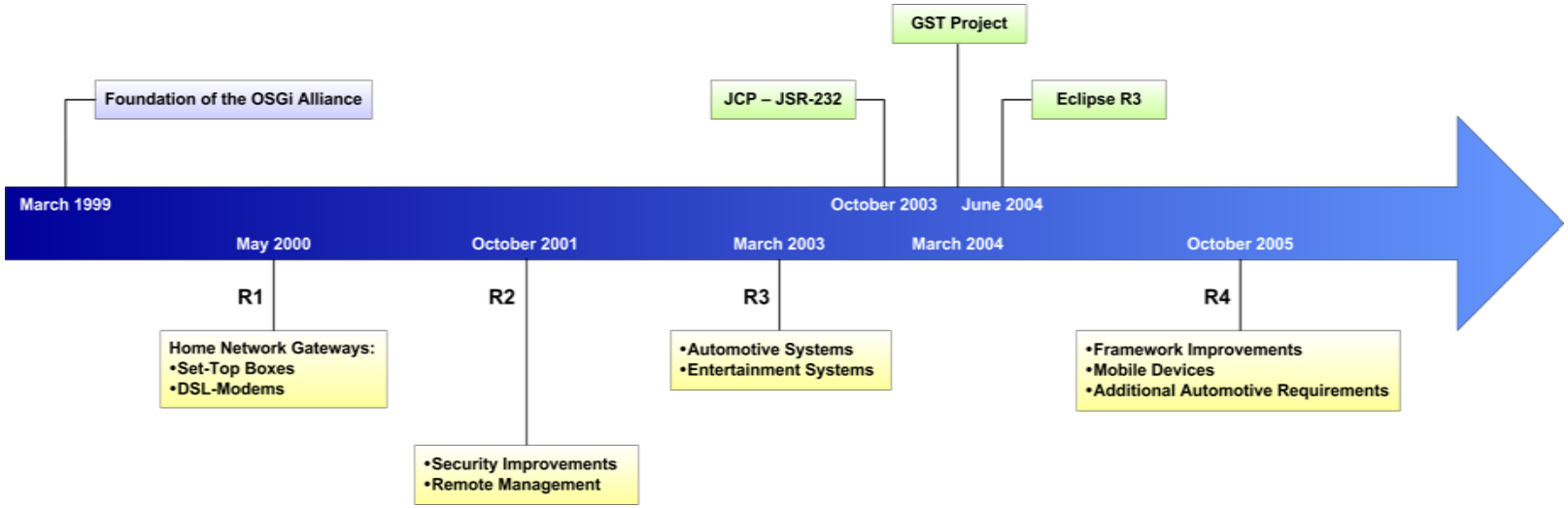
- **OSGi soll attributiv verwendet werden**
- **eine direkte Übersetzung gibt es nicht**
- **eine mögliche Übersetzung: Open Service Platform**

Zi

Framework, das dynamische Installationen von Service-Applikationen über eine Remote-Schnittstelle zur Laufzeit ermöglicht.

► **Java als Technologiegrundlage**

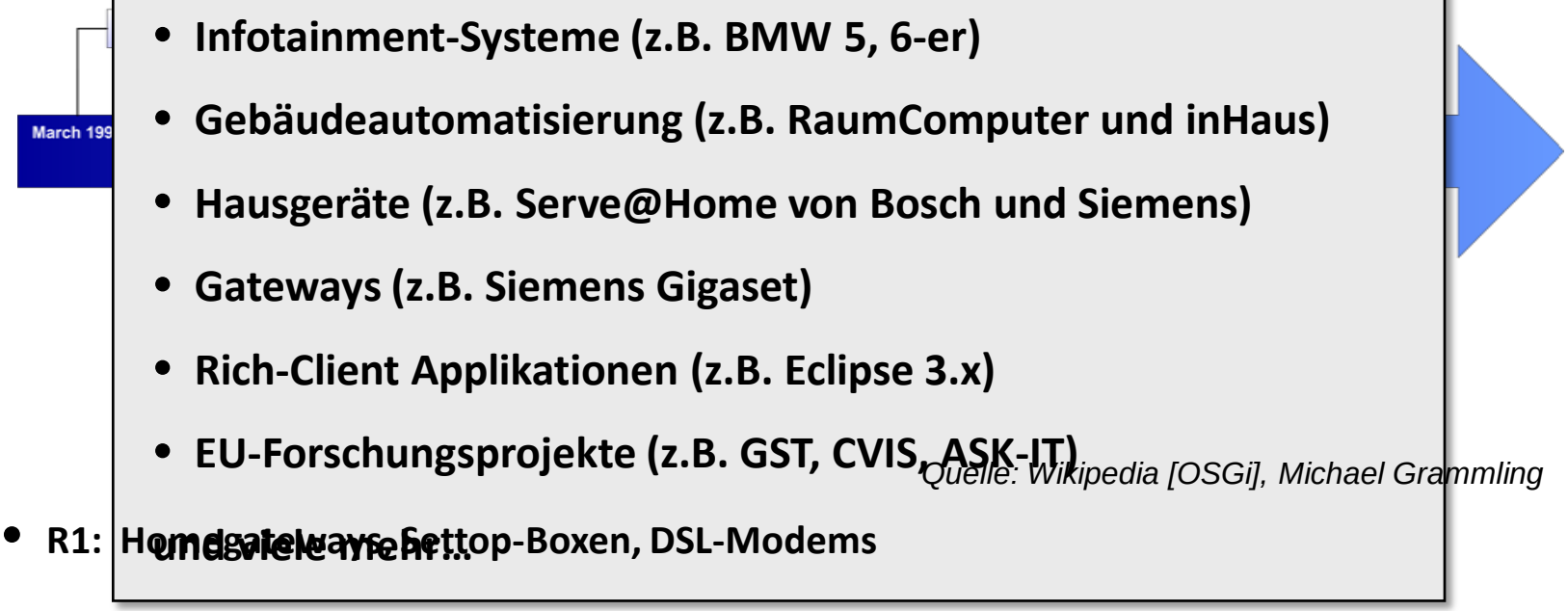
Die OSGi Releases



Quelle: Wikipedia [OSGi], Michael Grammling

- **R1: Homegateways, Settop-Boxen, DSL-Modems**
- **R2: Mobiltelefone, Automotive, Sicherheitssysteme**
- **R3: Gebäudesteuerung, Telemetrie, erweiterte Telefoniefunktionen**
- **R4: erweiterte Automotivefunktionen und Mobiltelefon-Erweiterungen**

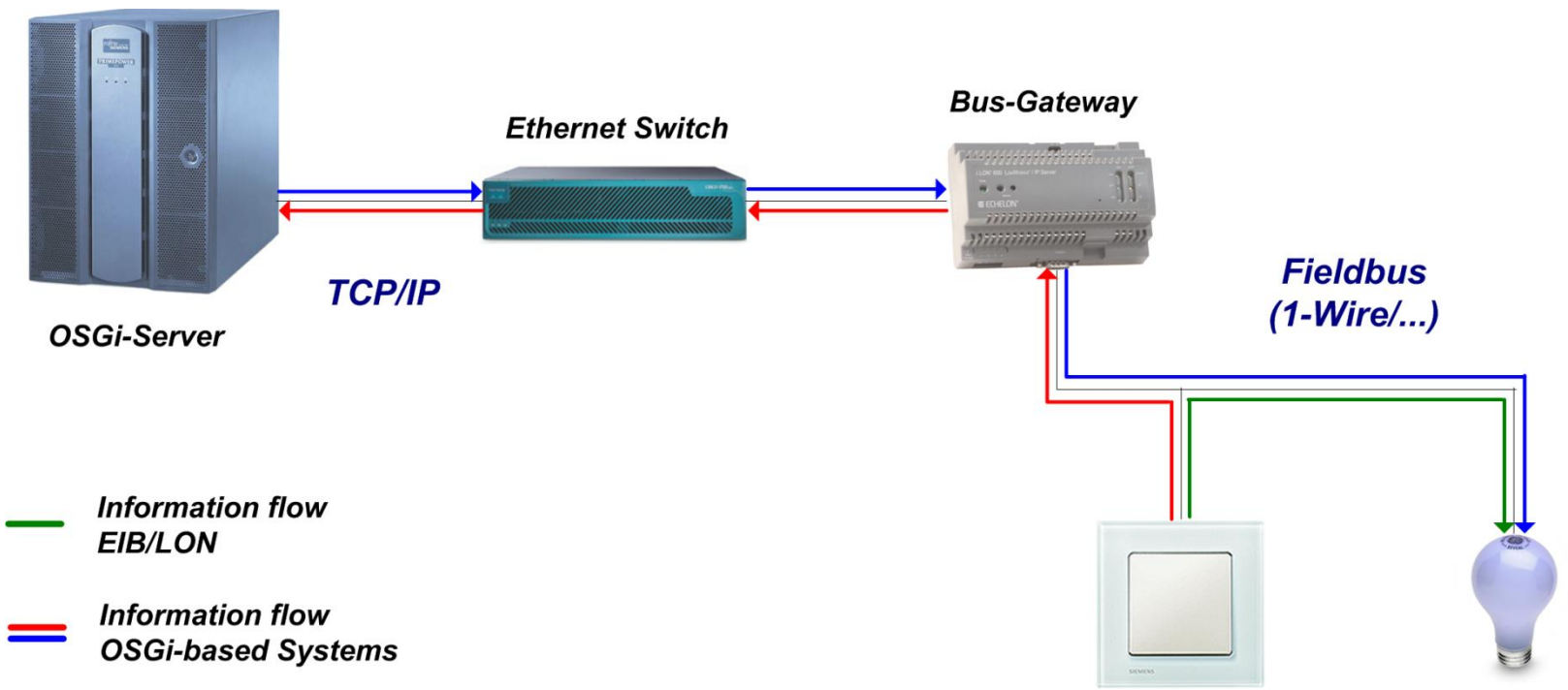
Die OSGi Releases



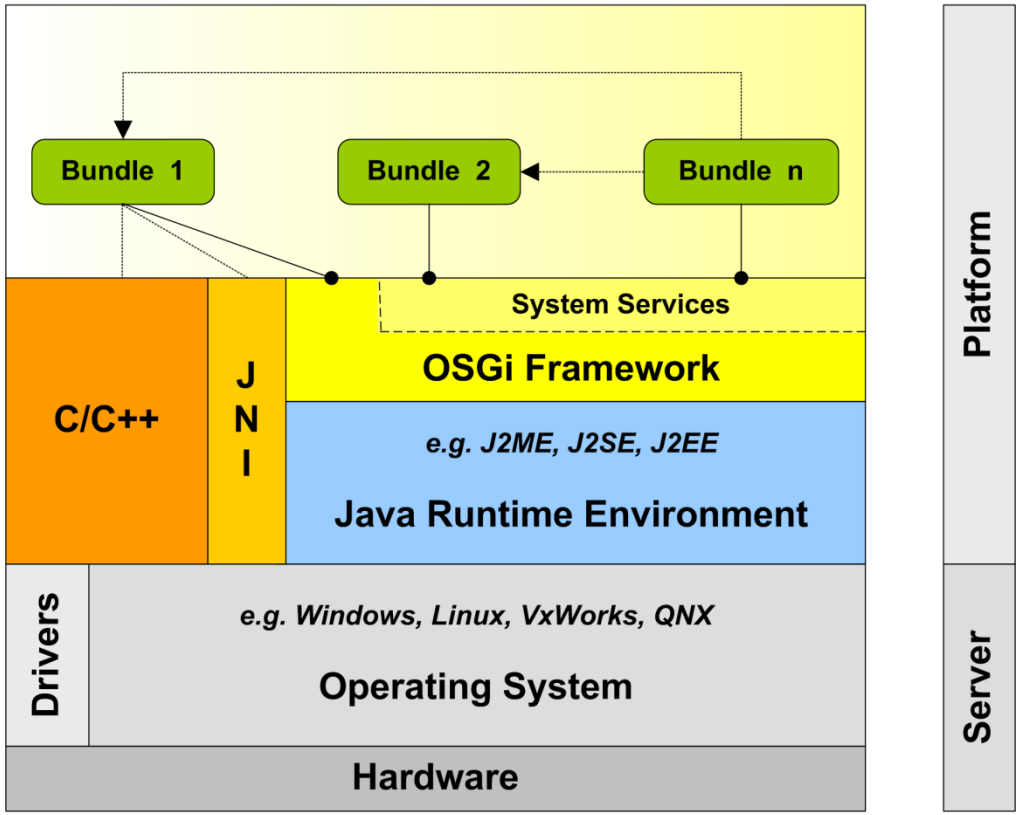
Quelle: Wikipedia [OSGi], Michael Grammling

- **R1: Homegateways, Settop-Boxen, DSL-Modems**
- **R2: Mobiltelefone, Automotive, Sicherheitssysteme**
- **R3: Gebäudesteuerung, Telemetrie, erweiterte Telefoniefunktionen**
- **R4: erweiterte Automotivefunktionen und Mobiltelefon-Erweiterungen**

Gebäudeautomatisierung

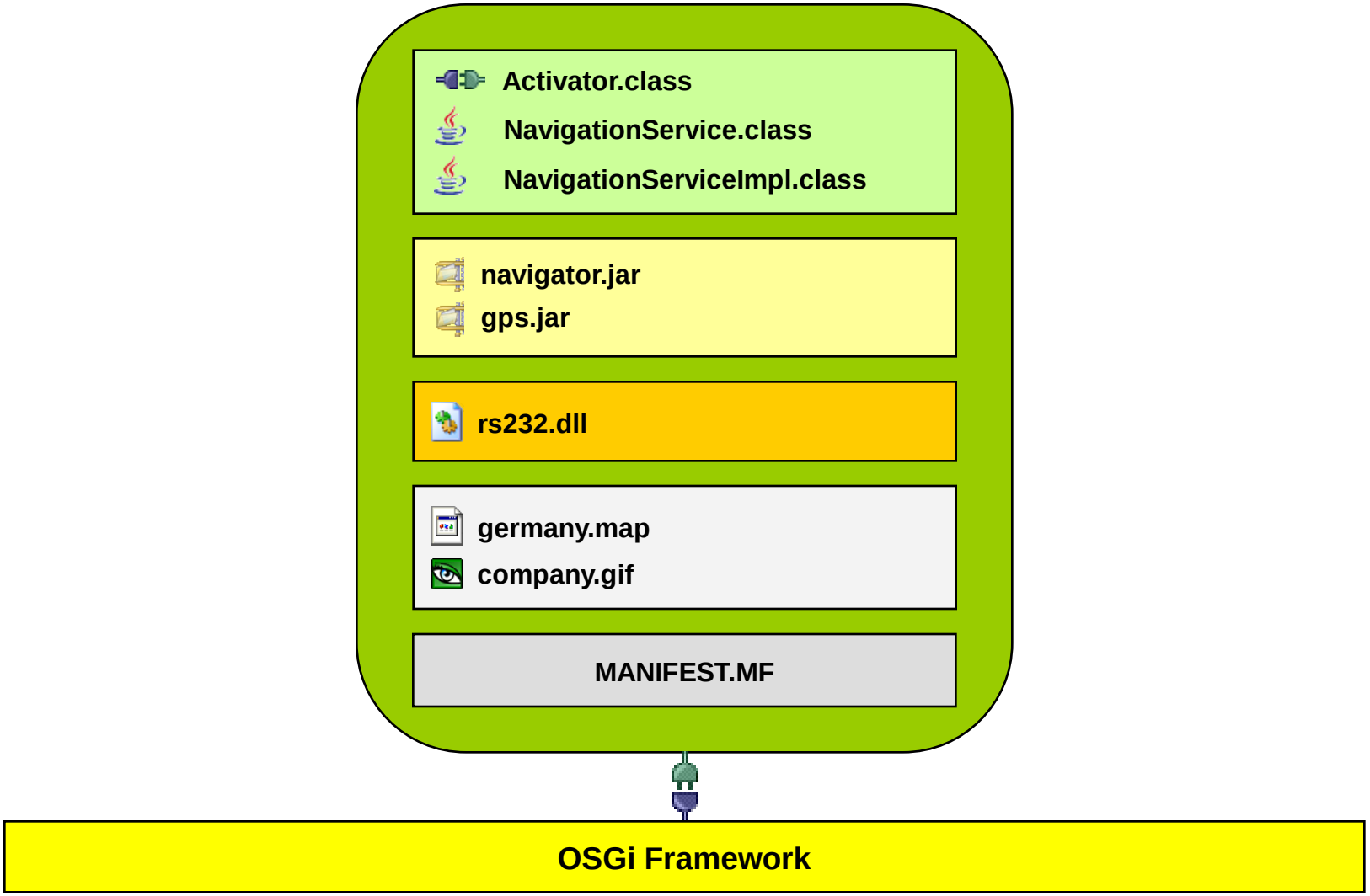


Systemschichtung und OSGi

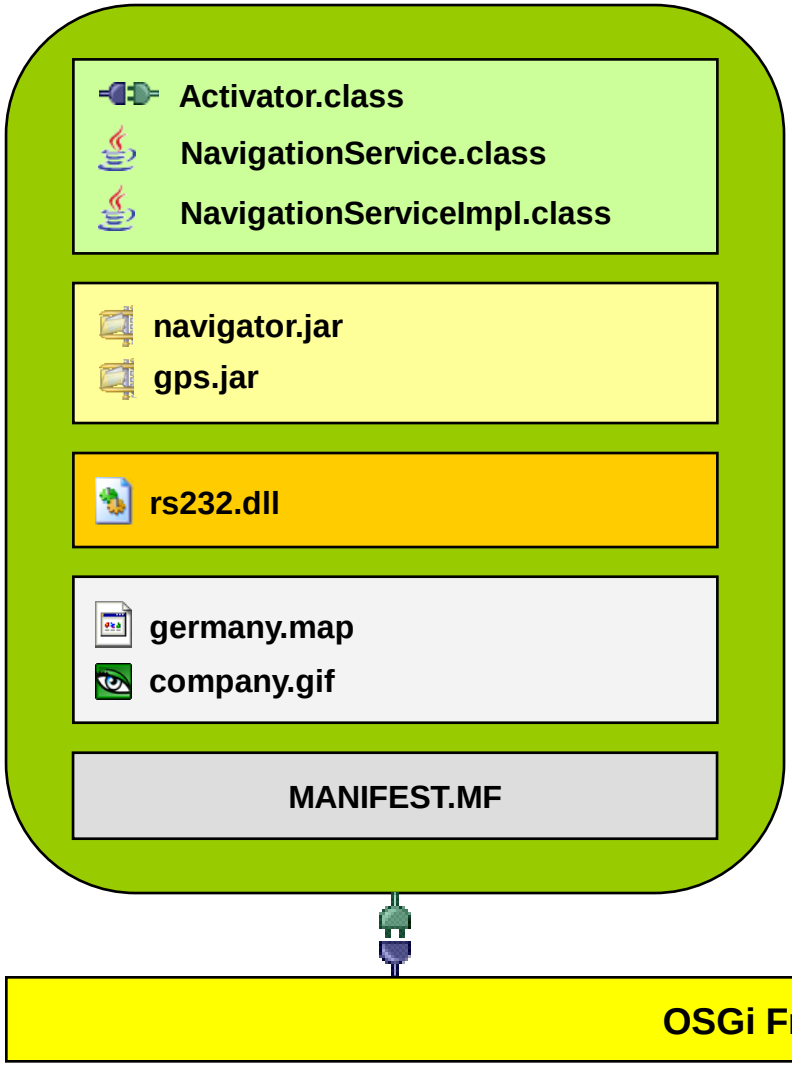


Quelle: Wikipedia [OSGi], Michael Grammling

Bundle Container (JAR-File)



Bundle Container (JAR-File)




```
class Activator
    implements BundleActivator {

  public void start(
    BundleContext context) {…}

  public void stop(
    BundleContext context) {…}
}
```

Eigenschaften:

- der BundleActivator steuert das Bundle
- eingeschlossene JARs werden unterstützt
- native Dateien werden unterstützt
- die MANIFEST.MF beschreibt das Bundle

Metafile – MANIFEST.MF (R3 Syntax)

```
Manifest-Version: 1.0
Bundle-Category: jame
Bundle-Name: Navigator Bundle
Bundle-DocURL: http://www.ptvag.com
Bundle-Description: This bundle provides a navigation service.
Bundle-Vendor: PTV AG
Bundle-ContactAddress: michael.grammling@ptv.de
Bundle-Copyright: Copyright (c) 2008, PTV AG
Bundle-Version: 1.0
Bundle-Activator: com.ptvag.jame.navigator.impl.Activator
Bundle-Classpath: ., gps.jar, navigator.jar
Import-Package:
        org.osgi.service.log,
        org.osgi.util.position,
        org.osgi.util.measurement,
Export-Package:
        com.ptvag.jame.navigator
Bundle-NativeCode:
/native/win32/rs232.dll;processor=x86;osname="Windows 2000";osname="Windows XP",
/native/wince/rs232.dll;processor=arm;osname="Windows CE"
```

Wesentliche Services in OSGi R3

- Device Access (dynamische Treiberverwaltung)
- HTTP Service (Webserver mit Servlet-Container)
- Log Service (Logging und LogReader Spezifikation)
- Preferences Service (persistenter Datenspeicherbereich für Bundles)
- Configuration Admin Service (Konfigurationsmanagement)
- Service Tracker (Überwachung des Lebenszyklus von Services)
- User Admin Service (Rollen-basierte Benutzerverwaltung)
- Wire Admin Service („verdrahtet“ Services)
- UPnP Device Service (Universal Plug and Play Unterstützung)
- Start Level Service (steuert den Startvorgang des Frameworks)
- IO Connector Service (IO-Verwaltung von J2ME)

Wesentliche Neuerungen in OSGi R4

- **Bundle-Signaturen**
- **Bundle-Versionierung mit entsprechender Laufzeitumgebung**
- **Event Admin Service (Abonnement-basierter Ereignisdienst)**
- **Declarative Services (Deklaratives Binden von Services)**
- **Deployment Admin (Remote Management Unterstützung)**
- **DMT Admin (OMA DM Unterstützung)**
- **Monitor Admin (Überwachung der Services)**

OSGi Service Tracking

- ServiceEvent
- ServiceTracker
- Declarative Services

- Whiteboard Pattern

Declarative Services – I

- bindet Services
- sorgt für geringeren Speicherverbrauch und erhöht die Startup-Geschwindigkeit
  - Services werden nur gebunden, wenn sie benötigt werden
  - reduziert den Energieverbrauch
- unterstützt den Lebenszyklus von Services
  - „Services können jeder Zeit kommen und gehen.“
- der BundleActivator ist nicht mehr notwendig
  - vereinfacht das Entwickeln von Services auf Basis von POJOs

- Benötigt wird dazu:
  - XML-Konfigurationsdatei
  - MANIFEST.MF, die auf die XML-Konfigurationsdatei verweist

Declarative Services – II

- **Beispiel für eine Konfiguration:**
  - **Komponente wird sofort gestartet**

/META-INF/MANIFEST.MF:

```
Service-Component: /OSGI-INF/activator.xml
```

/OSGI-INF/activator.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<component name="com.ptvag.jame.navigator.NavigationService">
  <implementation class="com.ptvag.jame.navigator.impl.NavigationServiceImpl"/>
</component>
```

NavigationServiceImpl.class:

```
public class NavigationServiceImpl {
  public NavigationServiceImpl() {...}
  protected void activate(ComponentContext ctxt) {...}
  protected void deactivate(ComponentContext ctxt) {...}
}
```


Declarative Services – III

- **Beispiel für eine Konfiguration:**
  - **Komponente erfordert den LogService**

/OSGI-INF/activator.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<component name="com.ptvag.jame.navigator.NavigationService">
  <implementation class="com.ptvag.jame.navigator.impl.NavigationServiceImpl"/>
  <reference name="LOG" interface="org.osgi.service.log.LogService"/>
</component>
```

NavigationServiceImpl.class:

```
public class NavigationServiceImpl {
  protected void activate(ComponentContext ctxt) {
    LogService logger = (LogService) ctxt.locateService("LOG");
    logger.log(LogService.LOG_INFO, "Hello Logger!"));
  }
}
```


Declarative Services – IV

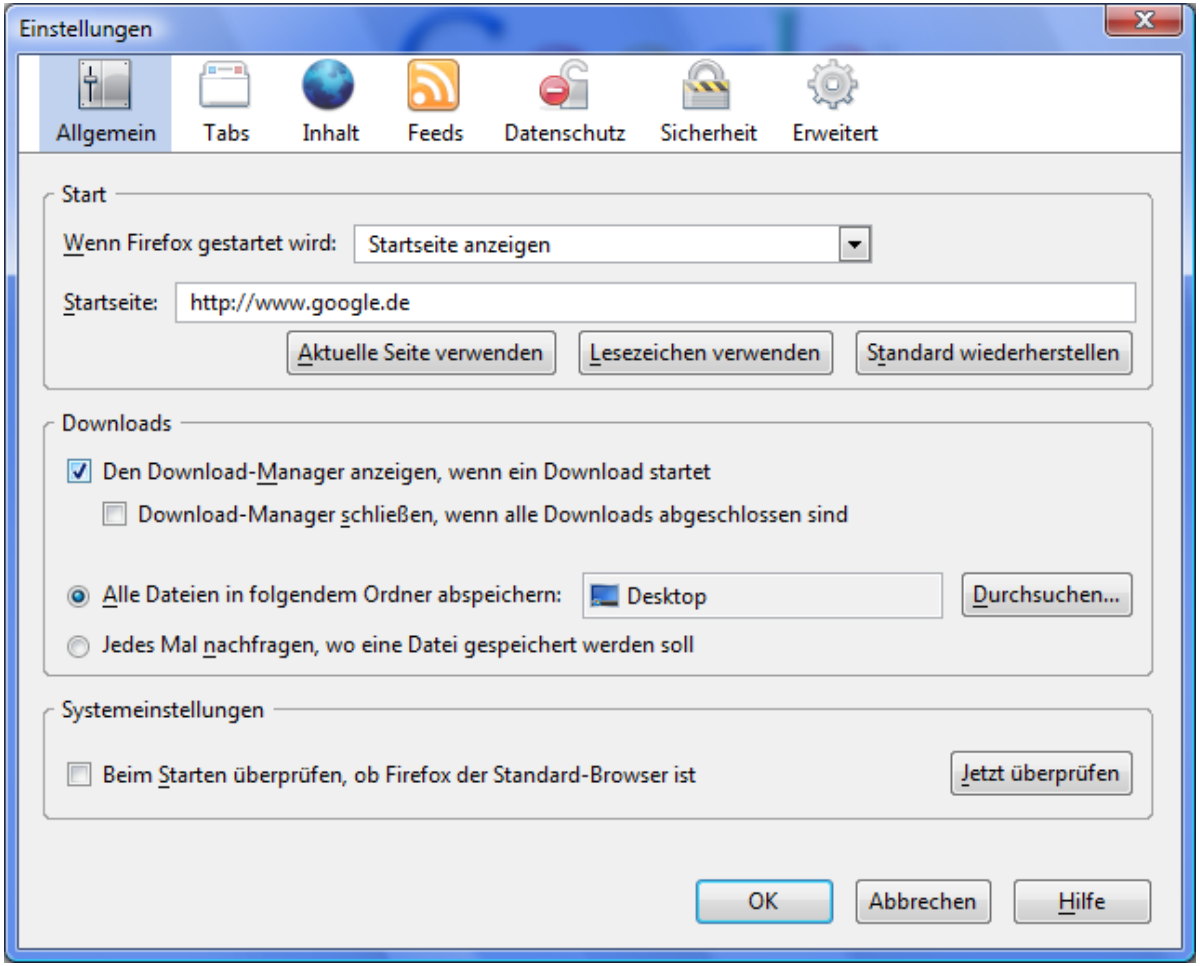
- **Beispiel für eine Konfiguration:**
  - **Komponente erfordert den LogService**

```
<?xml version="1.0" encoding="UTF-8"?>
<component name="com.ptvag.jame.navigator.NavigationService">
  <implementation class="com.ptvag.jame.navigator.impl.NavigationServiceImpl"/>
  <reference name="LOG"
    interface="org.osgi.service.log.LogService"
    bind="setLog"
    unbind="unsetLog"
  />
</component>

public class NavigationServiceImpl {
  LogService logger;
  protected void activate(ComponentContext ctxt) {
    logger.log(LogService.LOG_INFO, "Hello Logger!"));
  }
  protected void setLog(LogService logService) { logger = logService; }
  protected void unsetLog(LogService logService) { logger = null; }
}
```


Declarative Services – V

- der Zeitpunkt, wann Services verfügbar sind, kann festgelegt werden
- Filter zum Auffinden von Services werden unterstützt
- die activate / deactivate Methoden können weggelassen werden
- der ComponentContext ist ein erweiterter BundleContext
- Kardinalitäten werden unterstützt
- Factories werden unterstützt
- zyklische Abhängigkeiten zwischen Services werden erkannt, aber nicht automatisch aufgelöst!

Configuration Admin Service – I

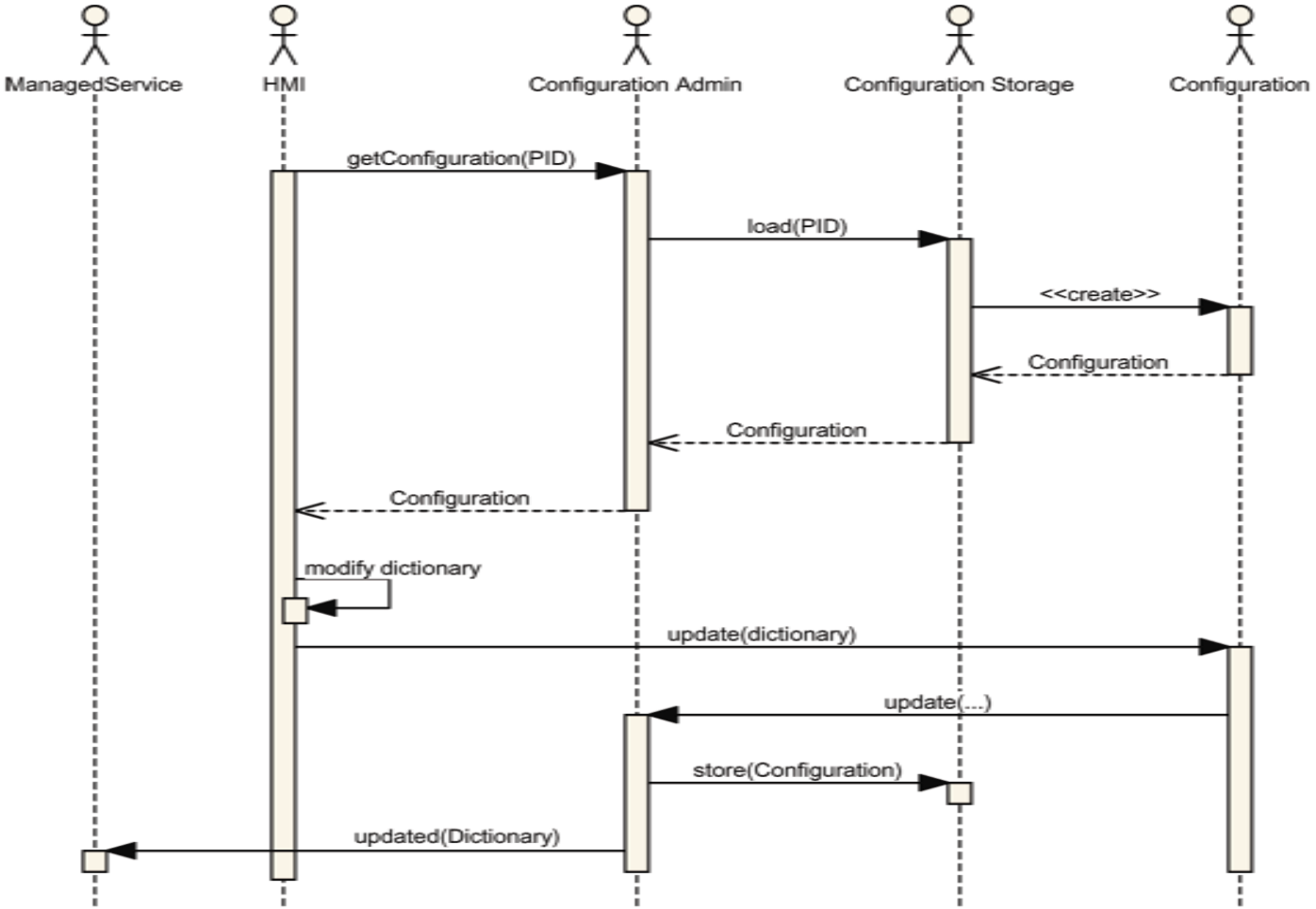


Configuration Admin Service – II

- ein konfigurierbarer Service muss:
  - die OSGi ManagedService Schnittstelle implementieren
  - sich als ManagedService in der OSGi Registry registrieren
  - eine PID (Persistent Identifier) in den Service Properties setzen
    - PID: z.B. „com.ptvag.jame.navigator.pid“ (meist Namen des Packages)

- Konfigurationen
  - sind beschränkt auf:
    - primitive Datentypen
    - eindimensionale Arrays und Vektoren einfachen Datentyps
  - werden asynchron versendet (Synchronisation ggf. notwendig!)

Configuration Admin Service – III



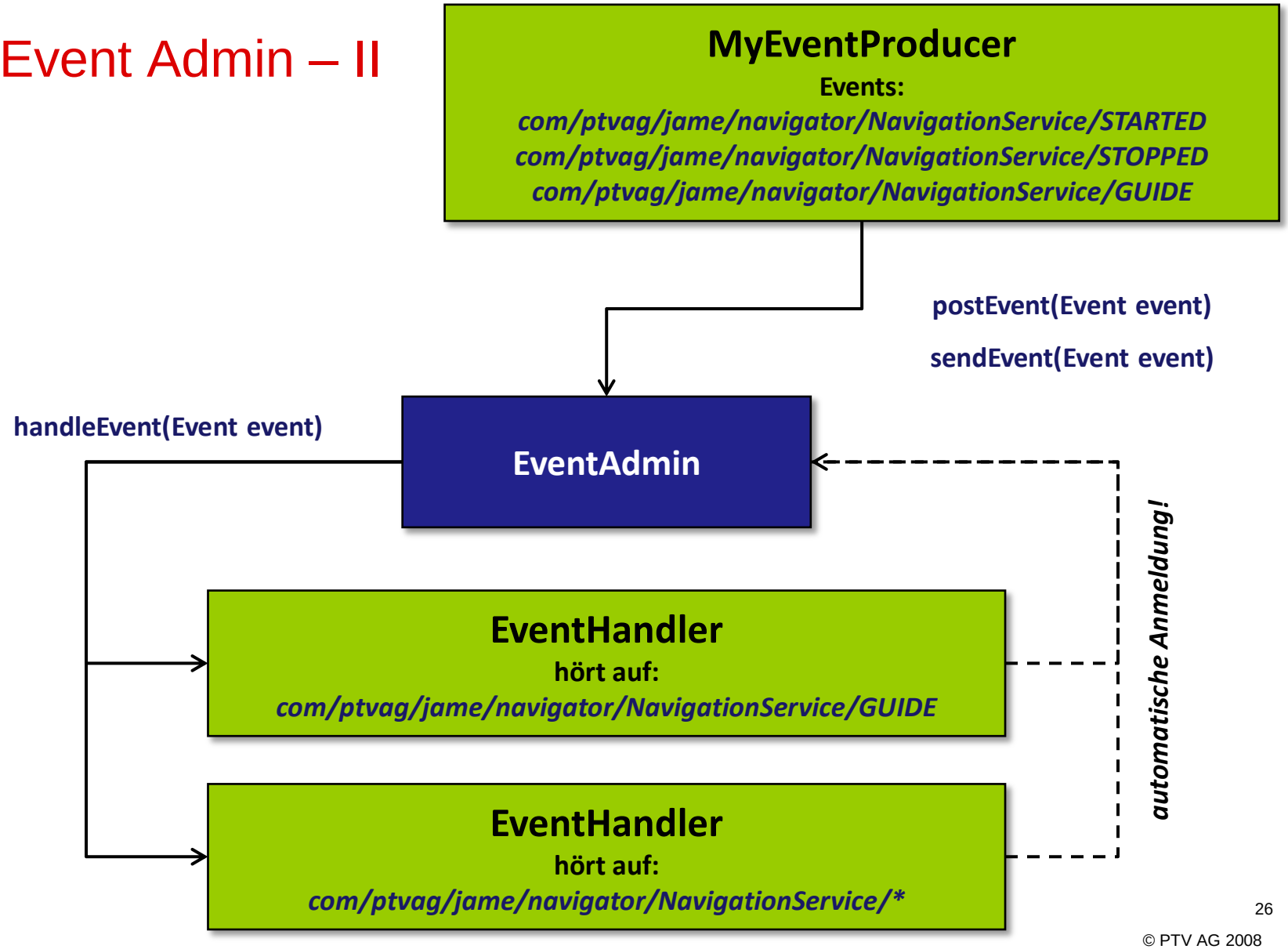
Configuration Admin Service – IV

```
public boolean setConfiguration(
    ConfigurationAdmin configAdmin, String pid, Dictionary properties) {

  try {
    Configuration config = configAdmin.getConfiguration(pid, null);
    Dictionary currentProperties = config.getProperties();

    if (currentProperties == null) {
      // no old configuration exists
      config.update(properties);
    } else {
      // synchronize properties
      Enumeration enumKeys = properties.keys();

      while (enumKeys.hasMoreElements()) {
        Object key = enumKeys.nextElement();
        Object value = properties.get(key);
        currentProperties.put(key, value);
      }
      config.update(currentProperties);
    }
  } catch (Exception ex) {
    return false;
  }
  return true;
}
```

Event Admin – I

- realisiert das Observer (publish-subscribe) Pattern
- verwirklicht die Idee des OSGi Whiteboard Patterns
  - keine eigene Liste mit Subscribern (Listenern) führen
  - keine Stale-References
  - filtert Events
- standardisierte Events verfügbar
  (z.B. Framework, Bundle und ConfigAdmin Events)

Event Admin – II



Event Admin – III

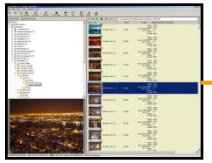
- **Events werden unter einem Topic (type) verbreitet:**
  - **das Topic ist case-sensitive und unique („unique identifier“)**
  - **dient als first-level Filter**
  - **reverse domain name soll verwendet werden:**
    z.B. com/ptvag/jame/navigator/NavigationService („.“ muss durch „/“ ersetzt werden!)
- **Events tragen Properties (Event-Daten)**
  - **Datencontainer: Dictionary (Hashtabelle)**
  - **Properties sind case-sensitive**
  - **die Werte sollen unveränderlich sein und Standarddatentypen entsprechen**
- **Events können synchron oder asynchron gesendet werden**
  - **asynchron: postEven(Event event)**
  - **synchron: sendEvent(Event event)**
  - **die Verbreitung wird nicht durch Exceptions unterbrochen (defensiv)**
- **Events können gefiltert werden**
  - **z.B. „com/ptvag/jame/navigator/NavigationService/GUIDE“**
  - **z.B. „com/ptvag/jame/navigator/NavigationService/*“ (Wildcard)**
  - **OSGi-typischer LDAP-Filter kann verwendet werden**

Event Admin – IV

```
Hashtable eventContent = new Hashtable();
eventContent.put("INSTRUCTION", new Integer(1));
eventAdmin.sendEvent(
  new Event("com/ptvag/jame/navigator/NavigationService/GUIDE", eventContent));
```

```
public MyEventReceiverService implements BundleActivator, EventHandler {
  private String topics[] = new String[] {
    "com/ptvag/jame/navigator/NavigationService/*",
    "org/osgi/service/log/LogEntry/LOG_ERROR"
  };

  public void start(BundleContext context) {
    Hashtable props = new Hashtable();
    props.put(EventConstants.EVENT_TOPICS, topics);
    props.put(EventConstants.EVENT_FILTER, "(bundle.symbolicName=com.acme.*)");
    context.registerService(EventHandler.class.getName(), this, props);
  }

  public void stop(BundleContext context) { . . . }

  public void handleEvent(Event event) { . . . }
}
```

Device Access Specification



Application Service

sucht

Digicam Device Service

Device Manager Service

DEVICE_CATEGORY
DEVICE_SERIAL
DEVICE_DESCRIPTION

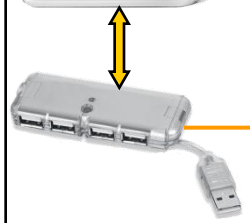
USB Device Service

Driver Locator Service (xy)

Driver Locator Service (Sony)

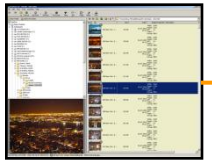
Driver Locator Service (Canon)

USB Base Driver Service



DB

Device Access Specification



Application Service

sucht

Digicam Device Service

Neues Gerät gefunden!

Device Manager Service

DRIVER_ID

Driver Locator Service (xy)

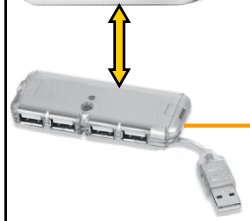
Driver Locator Service (Sony)

Driver Locator Service (Canon)

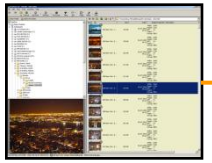
DB

USB Device Service

USB Base Driver Service



Device Access Specification



Application Service

sucht

Digicam Device Service

installiert

Universal Driver Service (xy)

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

Device Manager Service

Driver Locator Service (xy)

Driver Locator Service (Sony)

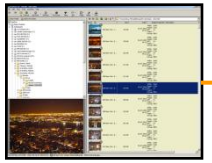
Driver Locator Service (Canon)

DB

USB Device Service

USB Base Driver Service

Device Access Specification



Application Service

sucht

Digicam Device Service

match

Universal Driver Service (xy)

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

Device Manager Service

Driver Locator Service (xy)

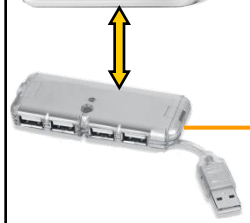
Driver Locator Service (Sony)

Driver Locator Service (Canon)

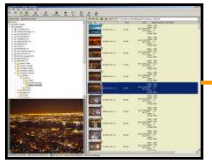
DB

USB Device Service

USB Base Driver Service



Device Access Specification



Application Service

sucht

Digicam Device Service

MATCH_VERSION

Universal Driver Service (xy)

Device Manager Service

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

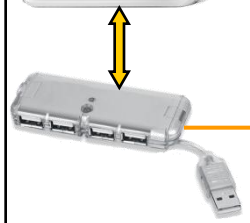
USB Device Service

Driver Locator Service (xy)

Driver Locator Service (Sony)

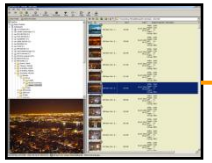
Driver Locator Service (Canon)

DB



USB Base Driver Service

Device Access Specification



Application Service

sucht

Digicam Device Service

Driver Selector Service

Universal Driver Service (xy)

Device Manager Service

attach

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

USB Device Service

Driver Locator Service (xy)

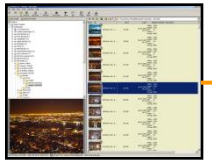
Driver Locator Service (Sony)

Driver Locator Service (Canon)

DB

USB Base Driver Service

Device Access Specification



Application Service

sucht

Digicam Device Service

Driver Selector Service

Device Manager Service

Universal Driver Service (xy)

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

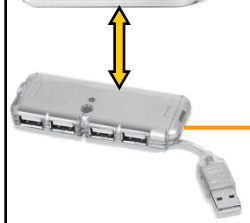
USB Device Service

Driver Locator Service (xy)

Driver Locator Service (Sony)

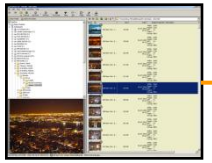
Driver Locator Service (Canon)

USB Base Driver Service



DB

Device Access Specification



Application Service

attached

Digicam Device Service

Driver Selector Service

Device Manager Service

Universal Driver Service (xy)

DSC-P200 Driver Service (Sony)

DSC-F828 Driver Service (Sony)

USB Device Service

Driver Locator Service (xy)

Driver Locator Service (Sony)

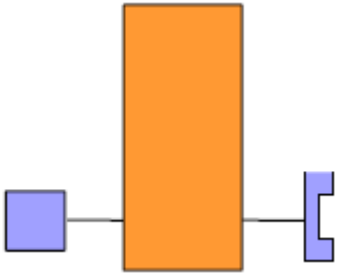
Driver Locator Service (Canon)

USB Base Driver Service

DB

Das Bundle Dilemma (nach Richard Hall) – I

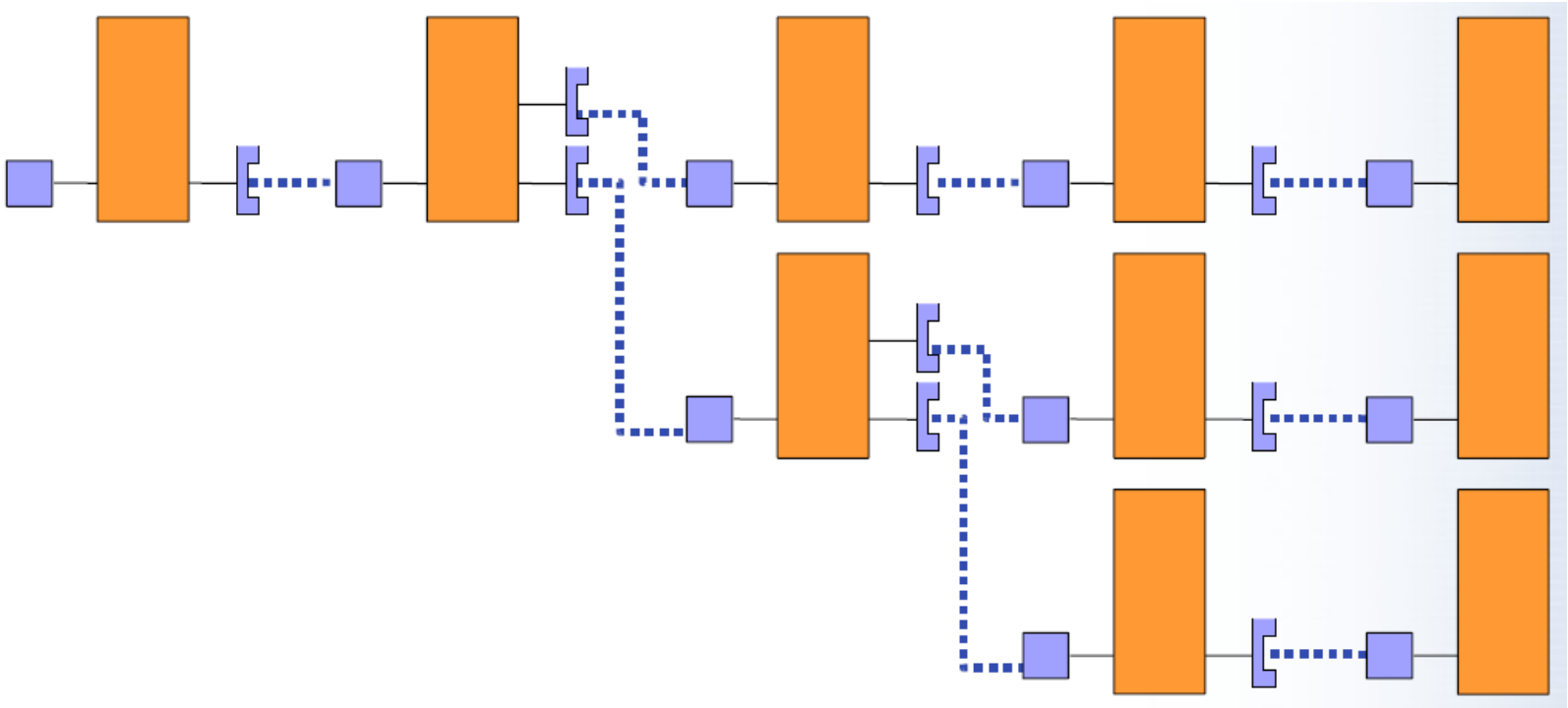
Sie finden ein Bundle und möchten es installieren, aber dann…



-> start 4
org.osgi.framework.BundleException:
Unresolved package in bundle 4: package;
(&(package=org.apache.felix.foo)
(version>=1.0.0))

Das Bundle Dilemma (nach Richard Hall) – II

… diese Bundles sind auch noch notwendig…



Das Bundle Dilemma (nach Richard Hall) – II

Idee von Richard Hall: **OBR (OSGi Bundle Repository)**

Oder?

Bundles entwickeln, die möglichst überhaupt keine Abhängigkeiten zu anderen besitzen.

Das kann zu Copy&Paste Aktionen führen.
Nicht übermäßig schlimm, wenn ein Buildsystem, wie. z.B. MAVEN2 eingesetzt wird.

Es bleibt die Versionsproblematik, die durch ein oder mehrere Library Bundles gelöst werden kann.

Vorgehensmodell – Von der API zum OSGi-Service

- **APIs definieren nach Anforderungen:**
  - **Schnittstelle als Service definieren**
  - **Implementation der Schnittstelle sollte Methoden zur Steuerung des Lebenszyklus enthalten (z.B. protected): (start/stop), shutdown**
  - **ausführliches Javadoc ohne Definitionslücken (Input / Output / Exceptions).**
  - **Methoden mit langer Ausführungszeit für hohe Interaktion unterbrechbar machen:**
    - **Timeout**
    - **„Cancel“ – Funktion**
  - **API soll ohne OSGi lauffähig sein und garantiert die Testbarkeit mit z.B. JUnit**

- **OSGi Service-Schnittstelle (ist i.d.R. identisch mit der API Service-Schnittstelle):**
  - **der Activator oder Declarative Services soll die OSGi-Interaktion kapseln**
  - **ein Service soll im Fehlerfall das Gesamtsystem nicht beeinträchtigen**
  - **Bundle erstellen (z.B. API einbetten).**

Etablierte Konventionen in der OSGi-Community

- **Service Schnittstelle:** **<package>MyClassService**
- **Service Implementation:** **<package>.impl.MyClassServiceImpl**
- **für Bundle-Activator:** **<package>.impl.Activator**
- **für Delclarative Services:** **OSGI-INF\activator.xml**

- **Designkonventionen**
  - **Verwendung des Observer Patterns vermeiden (z.B. Whiteboard Pattern nutzen)**

Links zu OSGi Frameworks

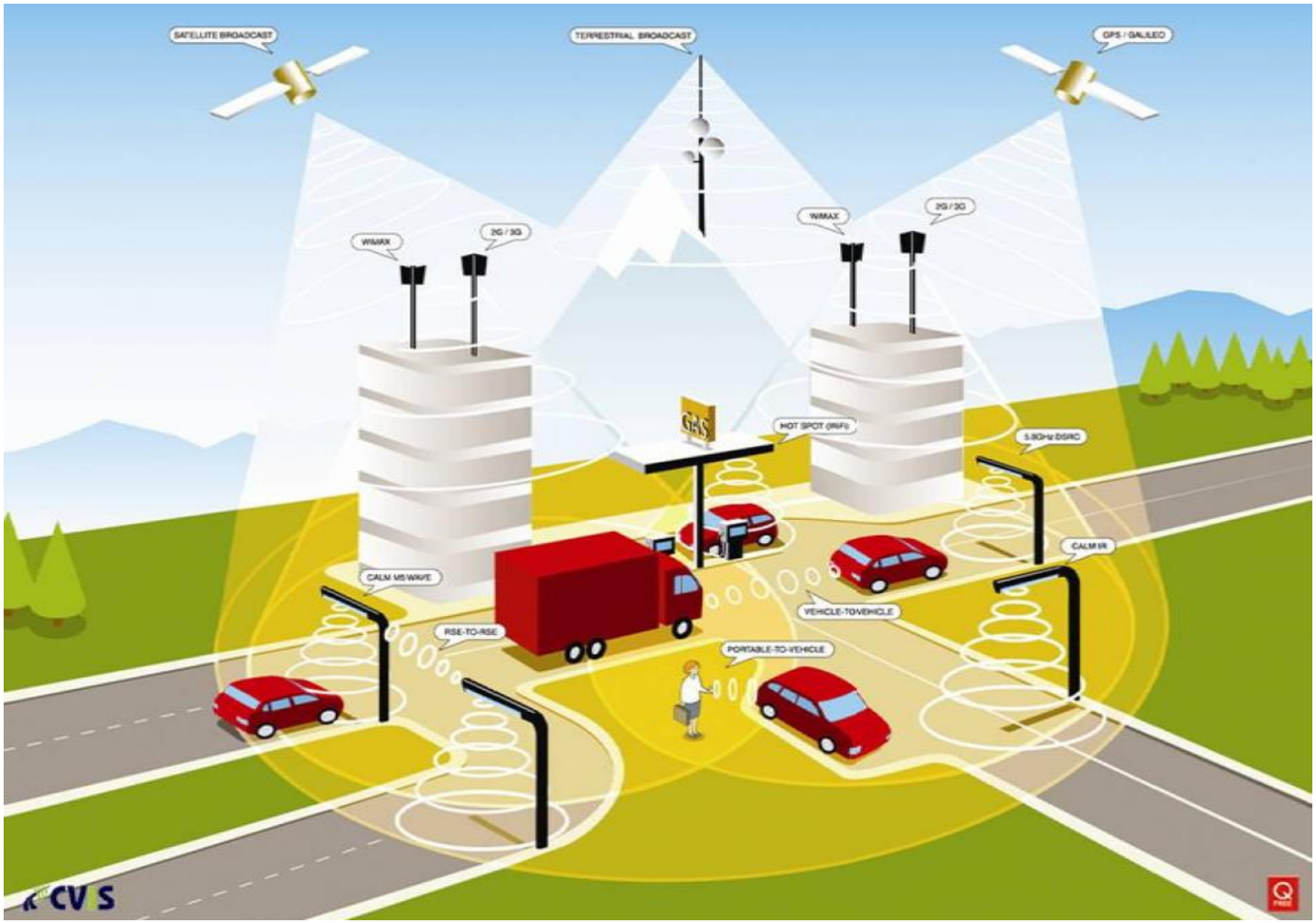
- **Prosyst mBeddedServer Equinox Edition (EPL)**
  - **http://dz.prosyst.com/oss/**
- **Prosyst mBeddedServer (kommerziell)**
  - **http://www.prosyst.com/**
- **Knopflerfish (BSD)**
  - **http://www.knopflerfish.org/**
- **Knopflerfish Pro (kommerziell)**
  - **http://www.makewave.com/**
- **Eclipse Equinox (EPL)**
  - **http://www.eclipse.org/equinox/**
- **Felix (Apache License v2.0)**
  - **http://felix.apache.org/**

PTV – first class transportation.

PTV Planung Transport Verkehr AG, 76131 Karlsruhe

traffic mobility logistics.

CVIS (Cooperative Vehicle-Infrastructure System)



Wire Admin

- die elektronische „Verteilerbox“
- bindet Services (nachdem ein Wire erstellt wurde)
- handelt Datenformat von Mitteilungen aus („Flavors“)
- sendet Mitteilungen (push und pull wird unterstützt)

- Nachteile:
  - umständliche Verwendung
  - Wires müssen verwaltet werden
  - die Präsenz eines Service kann nicht direkt ermittelt werden
  - keine Energieersparnis (da Service(s) bereits laufen muss/müssen)
  - ggf. komplexeres Exception-Handling

Was tut sich? – I

- **Dynamic Component Support für JSE – JSR291 (seit Feb. 2006)**
  - **OSGi für JSE**
  - **leider keine wesentlichen Erweiterungen (J2ME bleibt), keine Generics**
- **OSGi mal anders:**
  - **Java Module System für JSE 7 – JSR277 (seit Juni 2005)**
  - **Bug-Voting: http:bugs.sun.com/bugdatabase/view_bug.do?bug_id=6650394**
  - **JSR294 – Superpackages**
- **Google Android**

Was tut sich? – II

- **Gründung von weiteren Expert Groups:**
  - **Enterprise Expert Group**
  - **Vehicle Expert Group**
  - **Remote Management Expert Group**

Was tut sich? – III

- **ARM entwickelt IP-Cores mit Jazelle™ Technologie:**
  - **ARM 926EJ-S (200 – 500 MHz)**
  - **ARM7EJ-S (320 – 620 MHz)**
  - **ARM 1026EJ-S (266 – 540 MHz)**
  - **ARM 1136J(F)-S (320 – 620 MHz, optional: FPU)**
  - **ARM 1176JZ(F)-S (320 – 620 MHz, ARM Trustzone™, optional: FPU)**
- **Die Jazelle™ Technologie findet u.a. Einsatz in:**
  - **BenQ S80, LG U900, Microsoft Zune**
  - **Sony Ericsson: P990i, K500, K600, K700, K750i, S700, Z800, V600, V800**
- **Samsung entwickelt Prozessoren auf Basis der Jazelle™:**
  - **S3C2412 (ARM 926EJ, 200 MHz)**
  - **S3C2413 (ARM 926EJ, 266 MHz)**
  - **S3C2460 (ARM 926EJ, 266 MHz)**
  - **S3C24A0 (ARM 926EJ, 200 / 266 MHz)**
  - **Findet u.a. Einsatz in: HP iPAQ Pocket PC**
- **FreeScale entwickelt Prozessoren auf Basis der Jazelle™:**
  - **i.MX21(S), i.MX27, i.MX31**