TRESOR – the modular cloud

Building a domain specific cloud platform with OSGi

OSGi Community Event
Ludwigsburg
Eclipsecon 2013

About myself

Alexander Grzesik

Head of Development
medisite Systemhaus

Working 15 years in software development

Java
Software Architecture
Medical Software

alexander.grzesik@medisite.de

Building a domain specific cloud platform with OSGi

Cloud – the future ?

By David Fletcher

Building a domain specific cloud platform with OSGi

TRESOR Partners

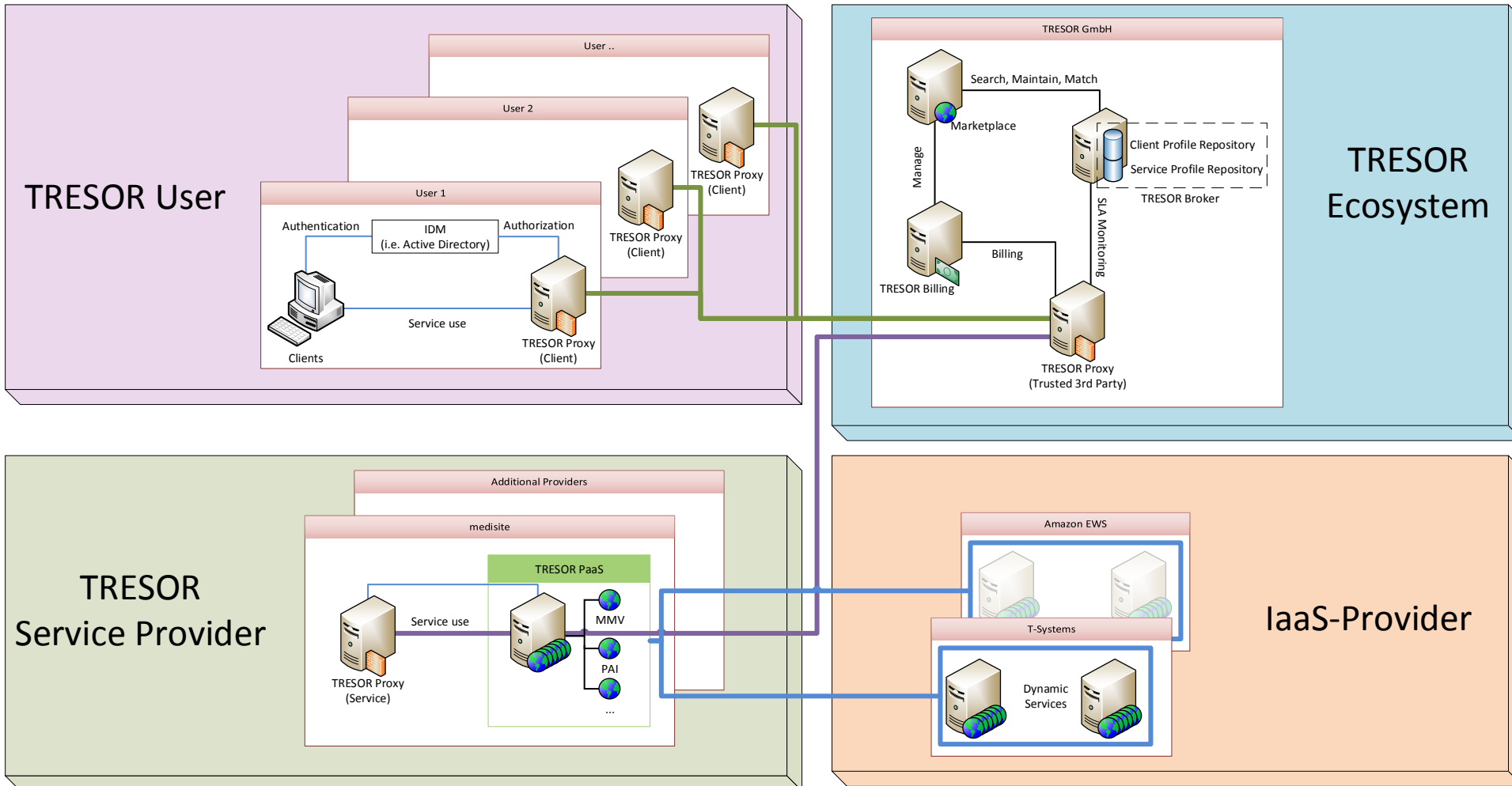Trusted Ecosystem for Standardized and Open cloud-based Resources

TRESOR is funded within the Trusted Cloud project by the Federal Ministry of Economy on basis of a resolution passed by the German Bundestag
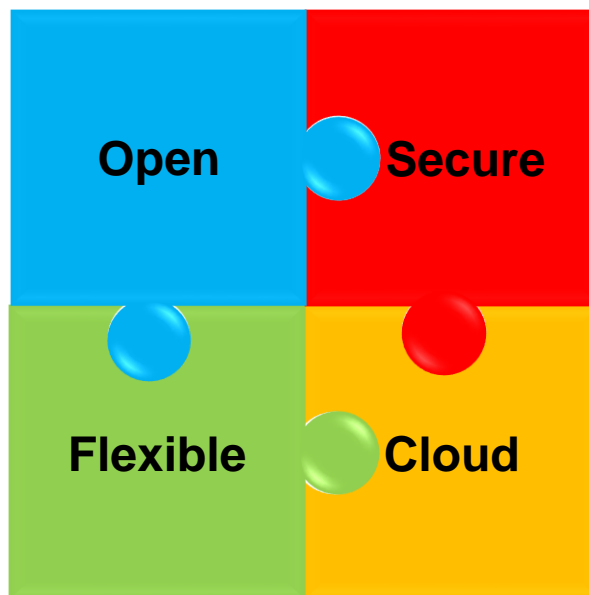
TRESOR Cloud Ecosystem

Eclipsecon 2013

Building a domain specific cloud platform with OSGi

TRESOR Goals

Extensible

OSGi based

Use of Standards

Development tools

Data Security

Encrypted Data

Secure Communication

Certified

Open **Secure**

Flexible **Cloud**

Fast Time-to-Market

No Vendor Lock-In

Flexible deployment

Scalable

Reliable

High Availability

Powered by OpenShift

Eclipsecon 2013

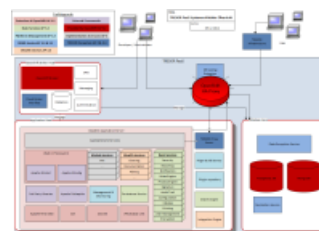Building a domain specific cloud platform with OSGi

TRESOR PaaS at a glance
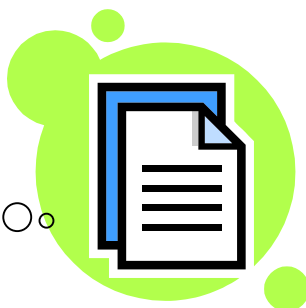
Open Platform

Polyglot Persistence

Strong Encryption

```
6dfg4854 fgf72548 151fd545
5454sff5 44485d1 151538fd
179hg45g 758d54d1 15414gfg
584551gh 1fghf15 154215jh
2152fgh5 4g25fg1 15325sgd
78dfd15d 7654fghd 897fg21d
98dfgh2d 874dfg6d 3544sdfg
```

Modular Architecture

Powered by OpenShift

Domain specific API

Building a domain specific cloud platform with OSGi

OPENSHIFT

Domain specific API

Application

Healthcare Applications and Services

Vaadin Web Framework	UI Components	UI Module Management

Management & Monitoring	Laboratory Diagnostic	Theraphy Planning	Radiology Diagnostic	Document Management
	Patient Adminstration	Patient Timeline	Clinical Documentation	Order Entry

Terminology	Reporting	Encryption	Business Rules	Object Mapping	User Management
Configuration	Search and Index	Security	Process Engine	Persistence	Notification

Apache Felix OSGi	Aries Blueprint	Enterprise OSGi	Java EE	JPA/Eclipse Link	3rd Party Bundles

- TRESOR Proxy Server
- Plugin Build Service
- Plugin repository
- Elastic Search
- Encryption Engine
- OpenAM
- Integration Engine

Cooking in the Cloud with OSGi

Building a domain specific cloud
platform with OSGi

The Challenges

Component Management

Provisioning

Dependencies

Architecture

Security

Java Enterprise Integration
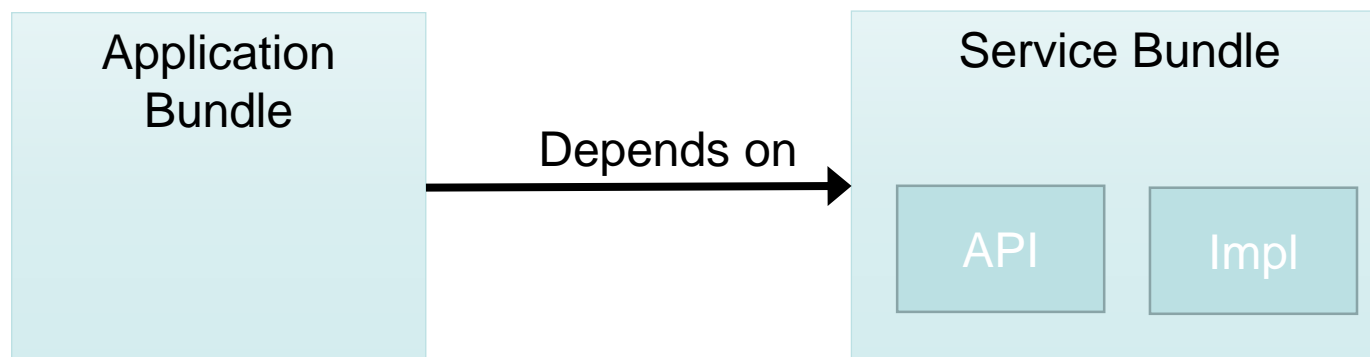
Configuration

Building a domain specific cloud platform with OSGi
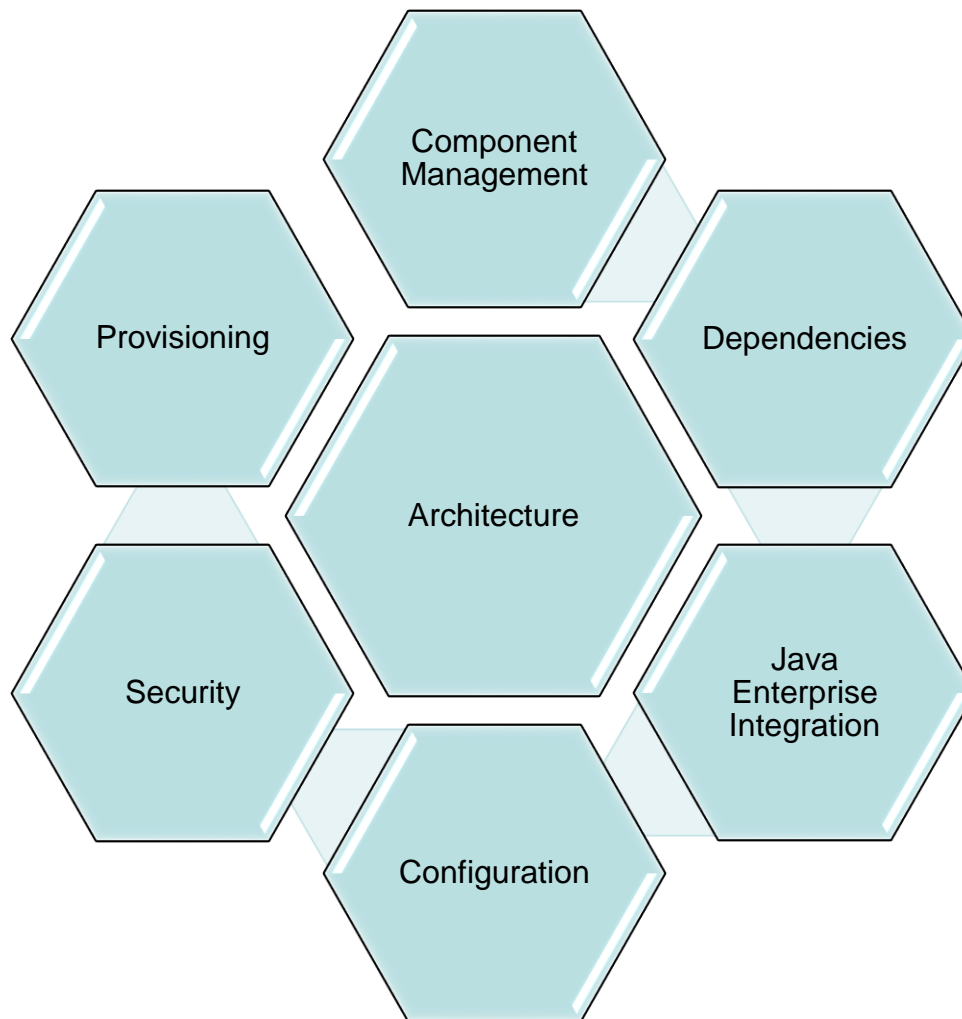
Bundle Structure

Straightforward

Tightly coupled API to Implementation

Replacing Implementation with high overhead

Building a domain specific cloud
platform with OSGi

Better: Bundle Separation

Application Bundle		Service API Bundle		Service Impl Bundle
	Depends on →	API	← Depends on	Impl
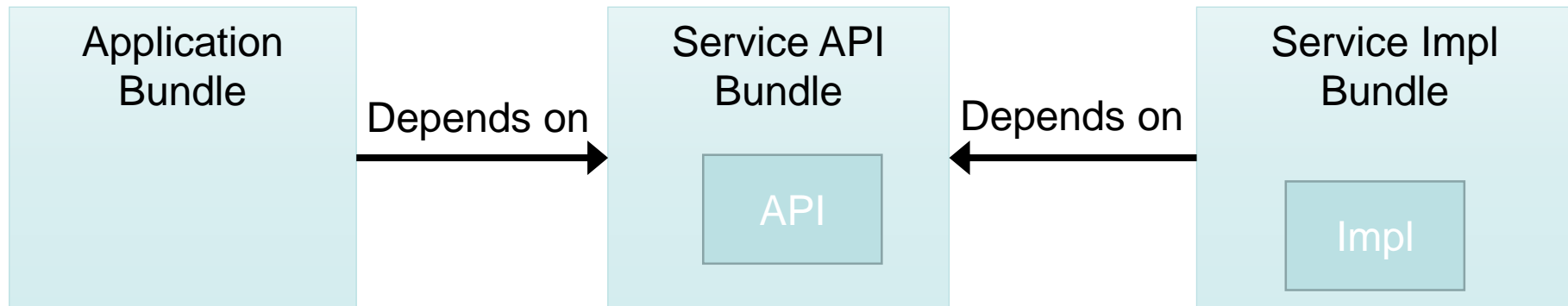
Separate API from Implementation

Application only depends on API

Implementation may be changed transparently

Managing with Blueprint

- ## Spring-style dependency injection

```
<bean id="bmiServiceBean"
class="medisite.eclipsecon.bmi.impl.BmiServiceImpl">
  <property name="calculatorBean" ref="bmiCalculatorBean"/>
</bean>
```

- ## Keeps code clean of OSGi dependencies

```
<service id="bmiService" ref="bmiServiceBean"
     interface="medisite.eclipsecon.bmi.BmiService" />
```

- ## Handles Service Lifecycle

```
<reference id="importedService"
interface=" medisite.eclipsecon.bmi.BmiService"
availability="mandatory"/>
```

- ## Enterprise Extensions

Managing Dependencies

Building a domain specific cloud platform with OSGi

Manage Dependencies

- Maven allows managing dependencies and versions

- Maven Bundle Plugin creates your bundles

```
<plugin>
  <groupId>org.apache.felix</groupId>
  <artifactId>maven-bundle-plugin</artifactId>
<configuration>
    <instructions>
     <Export-Package>medisite.eclipsecon.bmi.*</Export-Package>
     <Import-Package>
        org.slf4j;provider=paxlogging,
        *
     </Import-Package>
    </instructions></configuration></plugin>
```

Non OSGi dependencies

- Problem:
  A dependency is not an OSGi Bundle

- Option 1: Wrap bundle

```
<goals>
<goal>wrap</goal>
</goals>
<configuration>
  <wrapImportPackage>;</wrapImportPackage>
</configuration>
```

- Option 2: Embed dependency

```
<Embed-Dependency>
*;scope=compile|runtime;type=!pom;inline=false
</Embed-Dependency>
<Embed-Transitive>false</Embed-Transitive>
<Import-Package>*;resolution:=optional</Import-Package>
```

Java Enterprise Integration

Building a domain specific cloud platform with OSGi

Persistence

- JPA Persistence Units as OSGi Bundles
- Create Persistence.xml and include in bundle:

```
<Meta-Persistence>
    META-INF/persistence.xml
</Meta-Persistence>

<Include-Resource>
 META-INF/persistence.xml=src/main/resources/META-
INF/persistence.xml
</Include-Resource>

<JPA-PersistenceUnits>
 persistence-test
</JPA-PersistenceUnits>
```
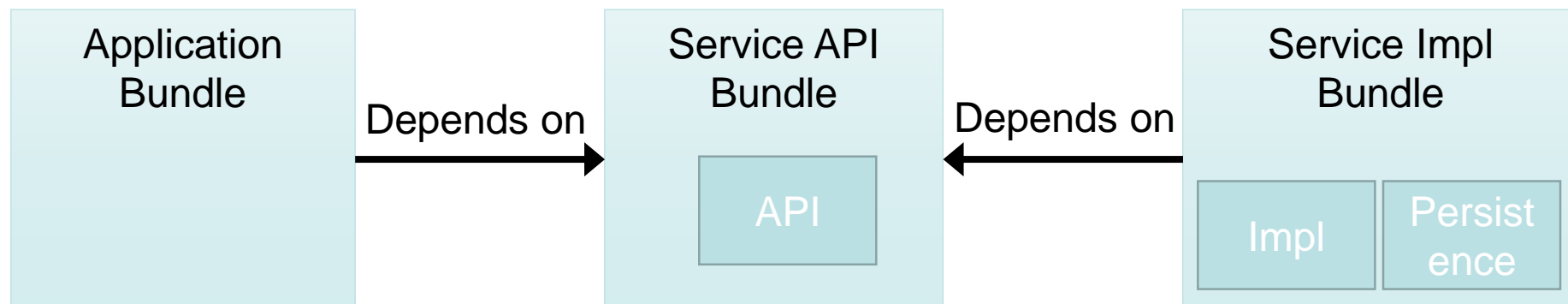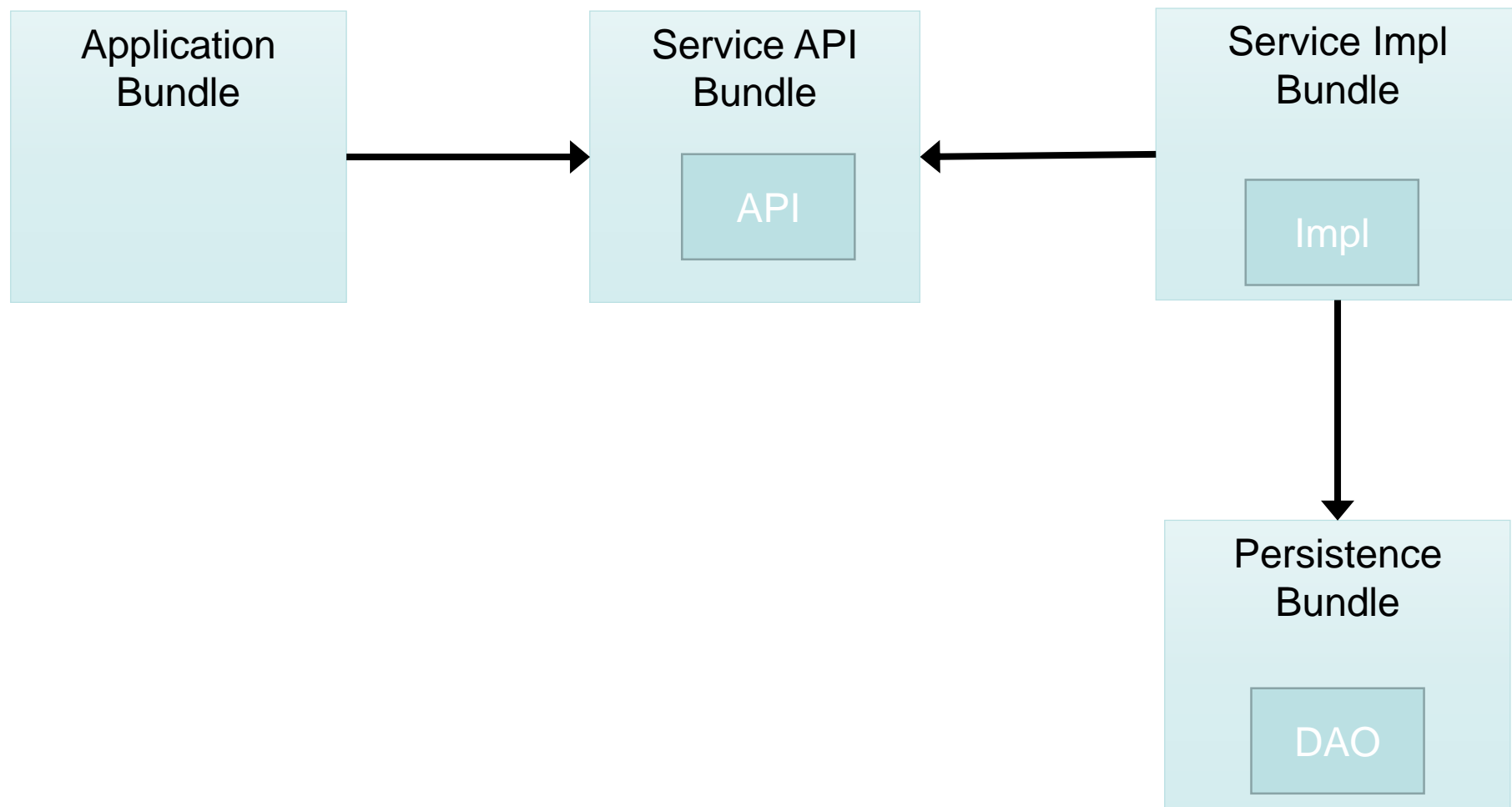
Embeded Persitence Unit

Application Bundle		Service API Bundle		Service Impl Bundle
	Depends on →	API	← Depends on	Impl / Persistence

Persistence Service

Application Bundle	→	Service API Bundle	←	Service Impl Bundle
		API		Impl

Persistence Bundle

DAO

Building a domain specific cloud platform with OSGi

Persistence Service via Blueprint

- Managed by Blueprint container (Aries)
- Declarative Transaction Management

```
<bean id="bmiDAO" class="medisite.eclipsecon.persistence.impl.BmiDAO"
init-method="init">
  <jpa:context property="entityManager" unitname="persistence-test" />
  <tx:transaction method="*" value="Requires" />
</bean>
```

Web Application Bundle

- ## Deploy a war as OSGi bundle (wab)

```
<instructions>
      <Web-ContextPath>/bmi</Web-ContextPath>
      <_wab>src/main/resources</_wab>
</instructions>
```

- ## Interact with OSGi Services from Servlet

```
BundleContext ctxt = (BundleContext)
servletContext.getAttribute("osgi-bundlecontext");
```

- ## JNDI Integration

```
InitialContext ic = new InitialContext();
IBmiService calculator = (IBmiService)
ic.lookup("osgi:service/" + IBmiService.class.getName());
```

Configuration

Building a domain specific cloud
                                              platform with OSGi

Configuration Administration Service

- ## Blueprint integration from Apache Aries

```
<cm:property-placeholder id="property-placeholder" persistent-
id="frameworkConfig" update-strategy="reload">
    <cm:default-properties>
        <cm:property name="selfTest" value="false"/>
    </cm:default-properties>
</cm:property-placeholder>
<bean id="testerBean" class="medisite.eclipsecon.bmi.impl.BundleTest>
  <property name="selfTest" value="${selfTest}"/>
</bean>
```

- ## Managed Properties

```
<bean id="bmiService" class="medisite.eclipsecon.impl.BmiServiceImpl">
<cm:managed-properties persistent-id="bmiServiceConfig"
  update-strategy="container-managed"/>
</bean>
```

Security

Building a domain specific cloud
platform with OSGi

ConditionalPermissionAdmin

- ## Control Permissions

```
ConditionalPermissionAdmin cpa = getConditionalPermissionAdmin(context);
ConditionalPermissionUpdate u = cpa.newConditionalPermissionUpdate();
List infos = u.getConditionalPermissionInfos();
infos.clear();
for (String encodedInfo : encodedInfos)
{
  infos.add(cpa.newConditionalPermissionInfo(encodedInfo));
}
if (!u.commit())
  throw new ConcurrentModificationException("Permissions changed during
update");
```

Policy File Reader

```
ACCEPT {
[org.osgi.service.condpermadmin.BundleSignerCondition
"CN=tresor,O=medisite Systemhaus GmbH,C=de"]
 ( java.security.AllPermission "*" "*")
}


DENY
{
(org.osgi.framework.PackagePermission "medisite.eclipsecon.*" "IMPORT")
}


ALLOW
{
(org.osgi.framework.PackagePermission "*" "IMPORT")
}
```

More thoughts on Security

- Make sure PolicyManager starts before custom bundles

- Restrict access to ConditionalPermissionAdmin

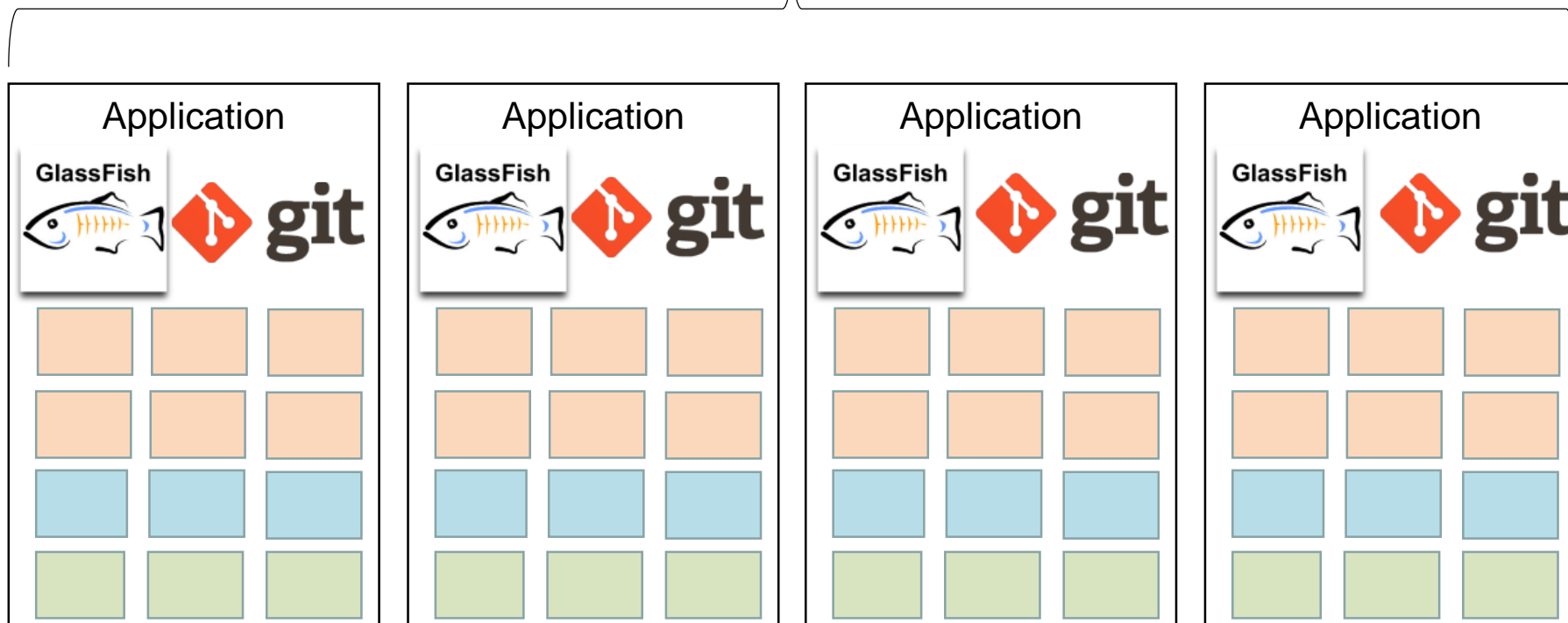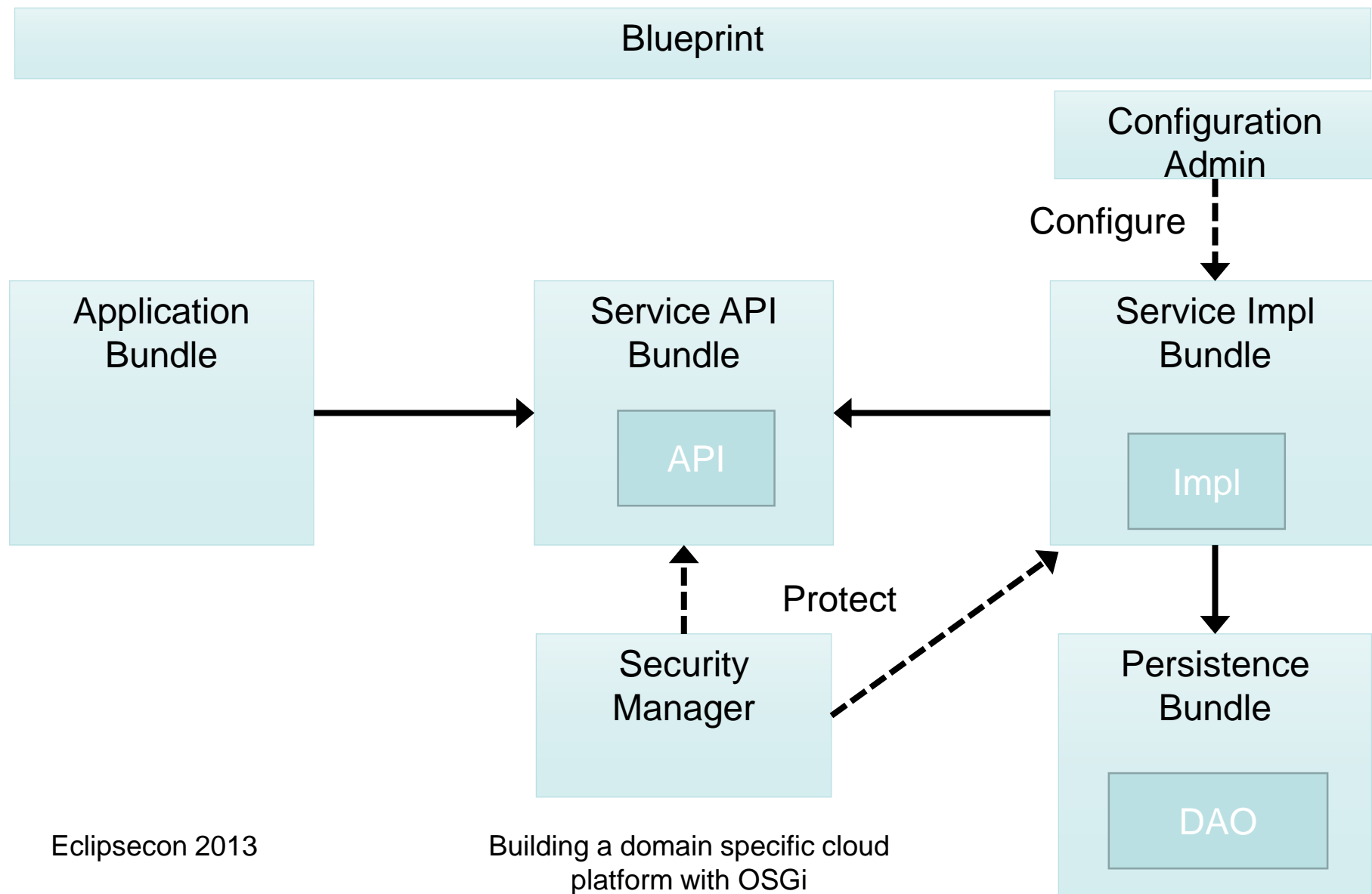- Application Permissions with blueprint interceptor (Aries)

Provisioning

Building a domain specific cloud
platform with OSGi

Cloud Provisioning

Putting it all together

Blueprint

Configuration Admin

Configure

Application Bundle

Service API Bundle

API

Service Impl Bundle

Impl

Protect

Security Manager

Persistence Bundle

DAO

Building a domain specific cloud platform with OSGi

Lessons learned

- Steep learning curve
- Detailed information is often missing
- From jar hell to bundle hell
  - Managing dependencies is challenging
  - Not all libraries support OSGi
- Difficult to migrate non-OSGi application to OSGi

Building a domain specific cloud
platform with OSGi

Benefits of OSGi for Architecture

- Separation of components

- Loose coupling

- Detect dependencies

- Encapsulation

- Versioning

- Integrating Java EE

Building a domain specific cloud
platform with OSGi

Think about your architecture

Building a domain specific cloud platform with OSGi
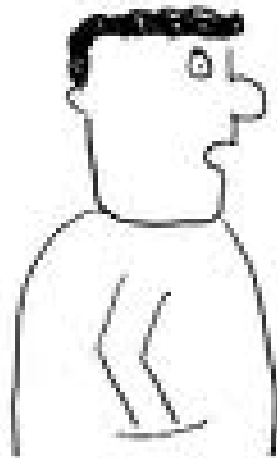
Useful Resources

- The OSGi Standard
- OSGi Books
  - OSGi in Action
  - OSGi in Depth
  - Enterprise OSGi in Action
- IBM Websphere Documentation
- Pax OSGi Projects

Questions ?