



Gerd Kachel, Stefan Kachel, Ksenija Nitsche-Brodnjan | kachel GmbH, www.kachel.biz

Migration from Java EE Application Server to Server- side OSGi for Process Management and Event Handling

Content

- Migration problem areas
- Requirements
- Migration solutions and patterns
- Lessons learned
- Wish list

Motivation for migration

- Migrate from GINGER to Roots
 - Paradigm reduction: Process Management, Messaging, and Event Processing → **Event Processing**
 - Older components became hard to maintain, needed some re-engineering
 - CTO would like to have something new 😊

Why OSGi?

- Best support for:
 - Modularisation
 - Service-orientation
 - Component dependencies
- A lot of useful predefined services
- Open source implementations available

Existing Product: GINGER

Standards:
WfMC, JEE,
BPMN

Software Engineer

Process Designer

GINGER Server based on JBoss 4

Process
Engine

JBI
Transfer

Monitoring

Convert
SDOM

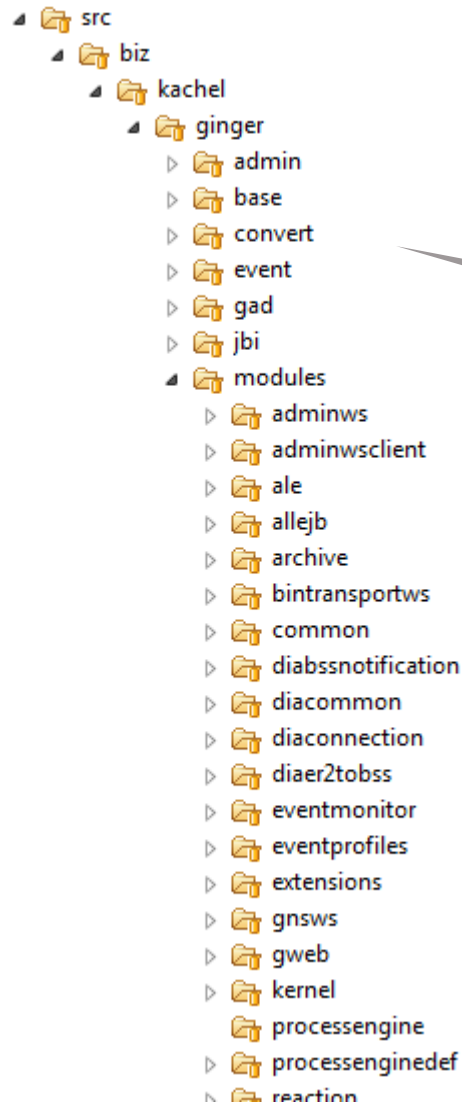
GSB

Repository

Console / Monitor

Administrator / User

Existing Software Structure



How to map to OSGi?

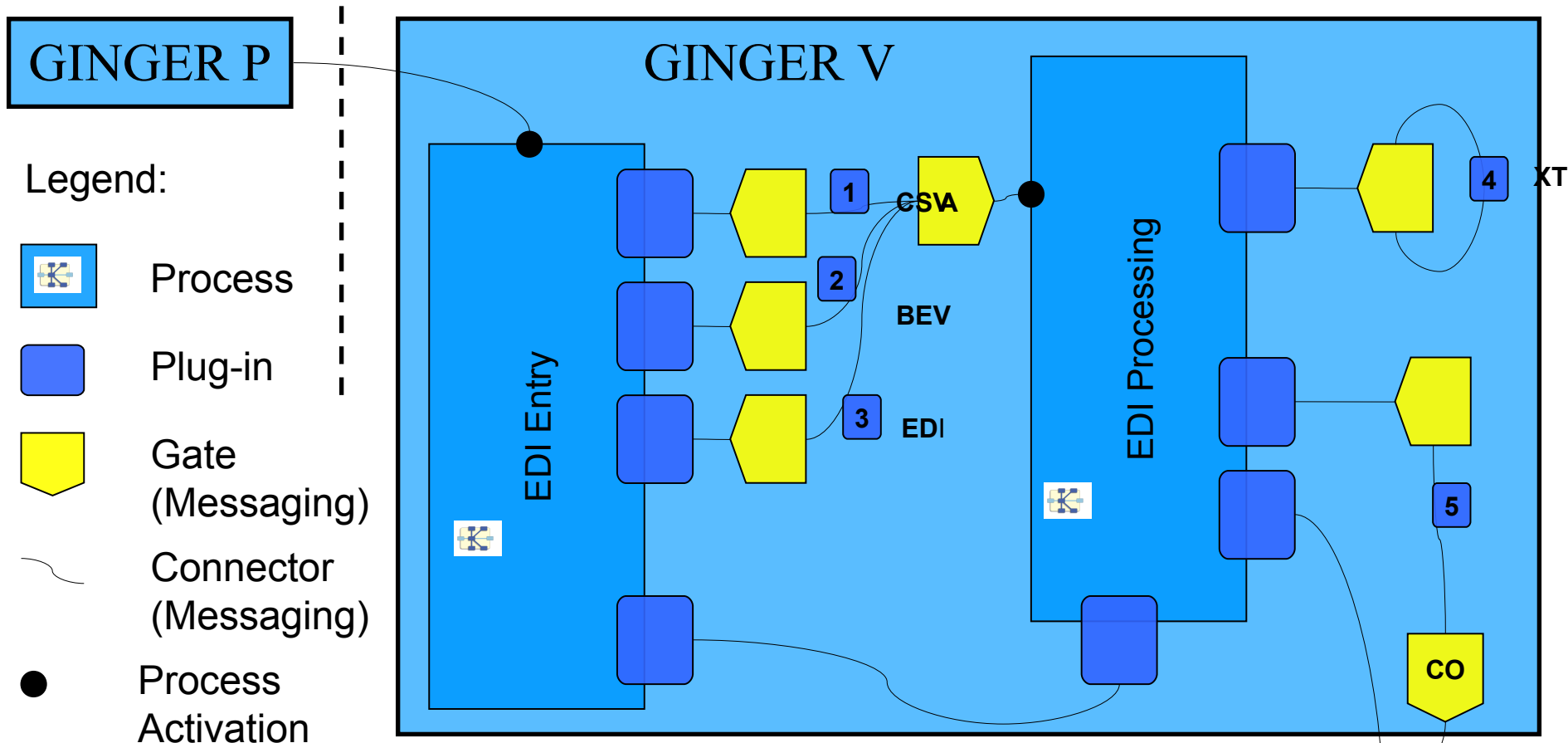
Components
(jars)

Modules
(JBoss MBean
services)

Ginger/JBoss

- Ginger/JBoss are composed of functional components and modules (FCM)
- Problems to solve
 - Find functional substitutes in OSGi
 - Identify FCM to embed in OSGi
 - Identify FCM to port to OSGi
 - Identify new FCM to be implemented

Existing Applications



**Plug-ins are POJOs:
How to map to OSGi?**

Cobol System

Summary: Problem Areas

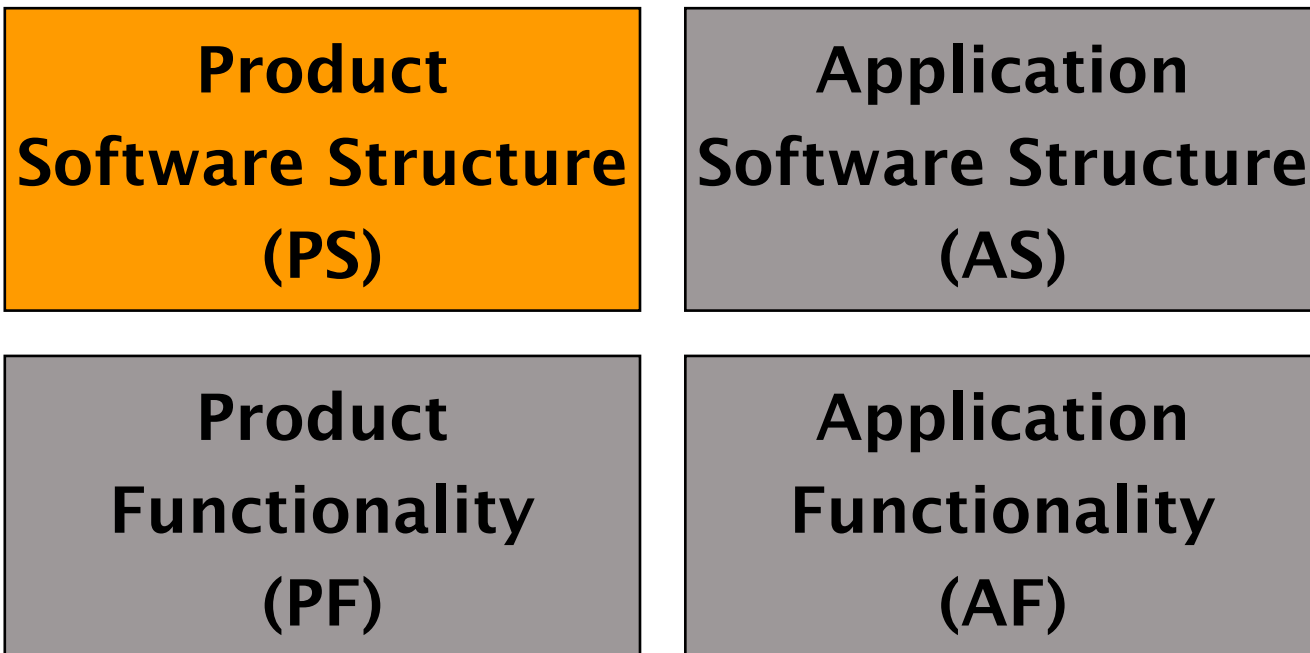
- Two kinds of software
 - Product
 - Applications of the product
- Two major problem areas
 - Mapping of software structure
 - Mapping of functional building blocks

Requirements

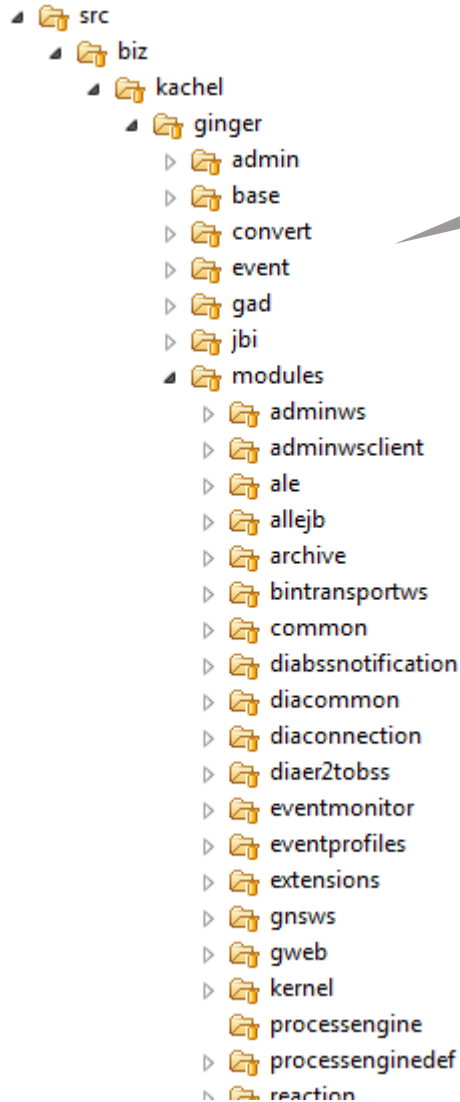
- Overall requirements
 - Minimum cost
 - Re-use as much as possible
 - Replace old components by OSGi technology
- Overall OSGi requirements
 - Use of Equinox (as starting point); today part of Eclipse RT
 - Use of declarative services

Migration Solutions / Patterns

Derived from problem areas:



Map Components to Bundles



Components
(jars)

Mappings:

- Intuitive:
Component → OSGi Bundle
- One to one: for good software designs

PS	AS
PF	AF

Bad Software Designs

lib

- activation.jar 5 27.09.08 16:18 gerd
- antlr-2.7.6.jar 5 27.09.08 16:18 gerd
- avalon-framework.jar 5 27.09.08 16:18 gerd
- bcel.jar 5 27.09.08 16:18 gerd
- bindingservice-plugin.jar 5 27.09.08 16:18 gerd
- bsh-1.3.0.jar 5 27.09.08 16:18 gerd
- bsh-deployer.jar 5 27.09.08 16:18 gerd
- cglib.jar 5 27.09.08 16:18 gerd
- commons-beanutils.jar 5 27.09.08 16:18 gerd
- commons-collections.jar 5 27.09.08 16:18 gerd
- commons-httpclient.jar 5 27.09.08 16:18 gerd
- commons-lang.jar 5 27.09.08 16:18 gerd
- commons-logging.jar 5 27.09.08 16:18 gerd
- ehcache-1.2.3.jar 5 27.09.08 16:18 gerd
- ejb3-persistence.jar 5 27.09.08 16:18 gerd
- gingeradminws.jar 86 03.04.10 15:11 gerd
- ainaealleib.iar 86 03.04.10 15:11 aerd

- JBoss flat class loading allows mud, e.g. dependency cycles

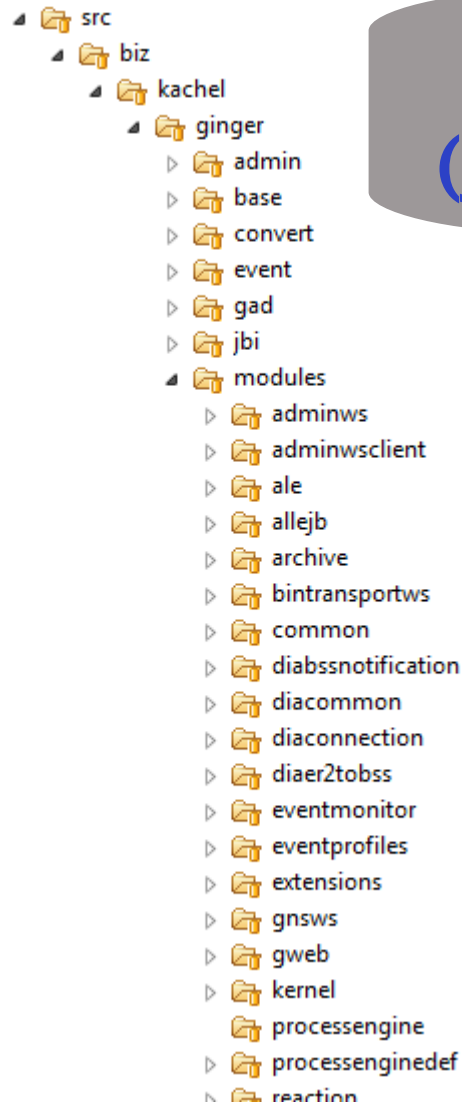
- OSGi forces good hierarchical software designs

→ re-engineering of components to hierarchical design

Bundle Design

- Old components were collections of packages
- Now, benefit from bundles:
 - Activator
 - Bundle respectively package dependencies
 - Hide internals, export packages to be used by other bundles:
 - Some adaptations required on component packages
 - Optional: add used libraries locally to bundle

Map Modules to Bundles



Modules (JBoss MBean services)

Mappings:

- Module → Bundle
- Modules: good software designs area
- Bundle design same as for components
- Add-ons required for services

PS	AS
PF	AF

Map MBean Services to Services

- <server>
- <mbean code="biz.kachel.ginger.modules.diaconnection.gmodule.
DiamantConnection"
name="biz.kachel.ginger:service=DiamantConnection">
- <!-- configuration attributes (and methods) -->
- <attribute name="MasterDatabase">sasystem</attribute>
- ...
- <depends>biz.kachel.ginger:service=system</depends>
- </mbean>
- </server>

Map one to one to Declarative Services in OSGi

Details on Service Mapping

- MBean service description
 - component service description
- MBean attributes
 - service properties
- MBean methods
 - to be provided as service interface
- MBean dependencies
 - referenced services

Special Features on Modules

- Modules with MBeans
 - Bundles with Component Services
- GModule = Ginger Module:
 - Wrapper for dependency injection
 - JBoss MBean Services
 - Spring Beans
 - Direct GModule implementation
 - Allows set of modules to be deployed as one bundle

Module Groups

Common for porting using a dependency injection framework (DIF):

- Keep set or sub-set of modules as one bundle
- Import DIF or include DIF libraries into bundle
- Start DIF on your bundle within bundle activator
- Export packages as externally required
- Provide services as externally required

That is great for porting to keep efforts low!

Third Party Software

- ▾ jboss
 - bin
 - client
 - docs
 - lib
 - scripts
 - ▾ server
 - default
 - ▾ ginger
 - conf
 - data
 - deploy
 - lib

JEE services: find
OSGi-like
equivalents, see
below

jars: wrap into
bundles

Summary: Product Structure

- Mapping of components, modules, and jars to bundles
- Mapping of MBeans to declarative services
- Bundle benefits support a good software design

Application Software Structure

- Migration mappings are the same as for the product structure
- In addition, mapping is desired for application code
 - Provided by POJOs

PS	AS
PF	AF

POJOs from Messaging

```
<InputGate Name=„BDProcessTrigger" Type="FILE">
  <FileInfo Format="Text">
    <Path>..\List\BaseDataImport</Path>
    <File>BaseDataStart.txt</File>
  </FileInfo>
  <PlugIns>
    <Class Name="biz.kachel.fair.BaseDataTransportPlugIns">
      <Method Name="readVTLs"/>
    </Class>
    <Class Name="biz.kachel.fair.ApplicationLog">
      <Method Name="work" Configuration="xml"/>
    </Class>
  </PlugIns>
</InputGate>
```

POJOs from Process Engine

```
<Application Id="clearSubCatalog_APP" Name="clearSubCatalog">
  <FormalParameters>
    <FormalParameter Id="CatalogSubName_FP" Mode="IN">
      <DataType>
        <BasicType Type="STRING"/>
      </DataType>
    </FormalParameter>
    ...
  </FormalParameters>
  <ExtendedAttributes>
    <ExtendedAttribute Name="Toolname"
Value="clearSubCatalog"/>
    <ExtendedAttribute Name="Tooltype"
Value="biz.kachel.fair.backbone.PlugInFurninetExport"/>
    <ExtendedAttribute Name="ExecutionFrame" Value="server"/>
  </ExtendedAttributes>
</Application>
```

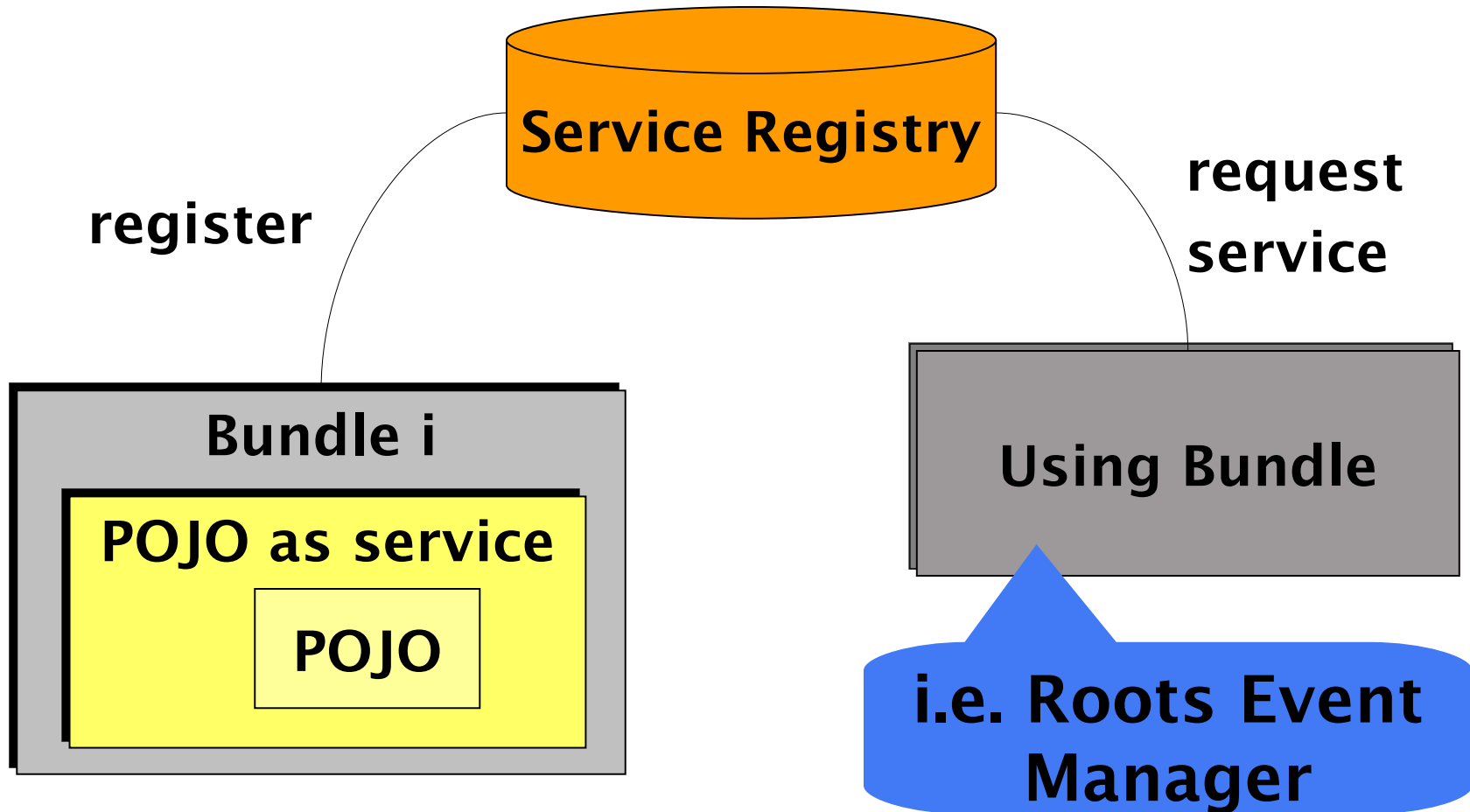
PS	AS
PF	AF

Common POJO Mapping

- Ginger/JBoss with messaging (MSG) and process engine (PE):
 - One class loader
 - Object instances are created via JAVA reflection API
- Roots/OSGi:
 - Bundle class loading
 - Such modules as MSG and PE are provided as bundle
 - POJO is part of application bundle
 - Problem:
 - MSG or PE are not aware of application code bundles

PS	AS
PF	AF

Common POJO Solution



PS	AS
PF	AF

Summary: Application Structure

- Mapping of POJOs
 - Provide POJO as service, service factory, or service providing POJO factory
 - Alternative:
 - fragments (limited)
 - not part of specification, e.g. Extensions

PS	AS
PF	AF

Product Functionality

- JBoss:
 - Look for substitute
- Ginger to Roots
 - Migrated according to software structure mappings
 - Look for substitutes

JBoss Substitutes

JBoss	Roots (OSGi, Eclipse RT)
Tomcat	HTTP Service, Jetty
WebService	Remote Services, Riena
Hibernate	Hibernate
EJB	–
log4j	Roots Logging, Events (DB-Track)

GINGER Substitutes

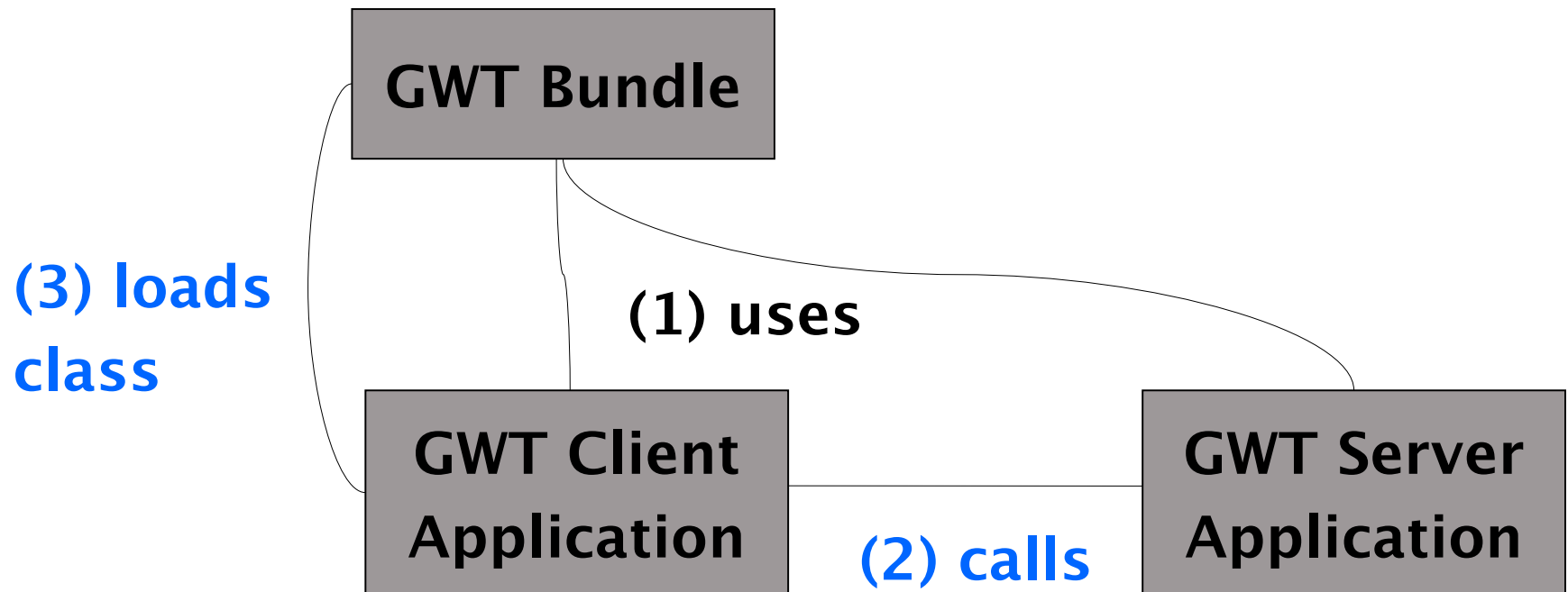
GINGER	Roots (OSGi, Eclipse RT)
Core	OSGi Core, Compendium partially
Process Engine	Event Manager
Messaging	Event Manager
Event Engine, Monitoring	Event Manager, Event Admin
UPnP	UPnP Device Service
Convert	Convert
GSB	Remote Services, Riena

Port to Event Processing

- Ginger Messaging and Process Engine are substituted by Roots Event Processing (Event Manager)
- In addition: migrate messaging gates to pro-active or re-active event processors

PS	AS
PF	AF

Google Web Toolkit Migration



Solve class loading problem!

→ **Equinox Buddy Class loading**

Summary: Product Functionality

- Used main JBoss and main Ginger components are directly substituted by OSGi and Eclipse RT components
- Direct porting according to migration rules
- Concept shot to event processing

PS	AS
PF	AF

Application Functionality

- Map messaging models and processes to event processes
- Map POJOs to services
- Add some glue if necessary

PS	AS
PF	AF

Lessions Learned

- Drawbacks
- Benefits
- Patterns
- Costs

Drawbacks

- Need invest in learning curve: about 4 weeks per person
- Non modularised designs are costly to migrate
- Bundles and services require implementation overhead: about 15 to 30 min per part

Benefits and Patterns

- Benefits
 - OSGi specification and implementation provides rich functionality
 - Add-ons by platform, e.g. Eclipse RT
 - Reduced software size
 - OSGi forces stronger design rules (good designs)
 - Fine grained modularisation
- Migration patterns
 - As shown above: components, modules, POJOs

Migrations Costs

- Learning curve: about 1 month per person
- GINGER core to one bundle: 1 person day (pd)
- GINGER module to bundle: about 1 pd
- Substitutes, new bundles, adaptations, test: 65 pd
- Total effort: about 110 pd (Ginger total development effort is about 2000 pd)

Wish List

- Declarative services via annotations
- Open implementation by specifications, e.g. call back or persistence layer for User Admin Service or Preferences Service
- Extend Event Admin features:
 - Open event property behaviour, e.g. event persistence
 - Open event processing modells