

**RED HAT
SUMMIT**

**LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.**

**June 11-14, 2013
Boston, MA**





JBoss SOA Platform 6 Quick Start

Kevin Conner

SOA Platform Architect, Red Hat

Keith Babo

SwitchYard Project Lead, Red Hat

June 13th, 2013



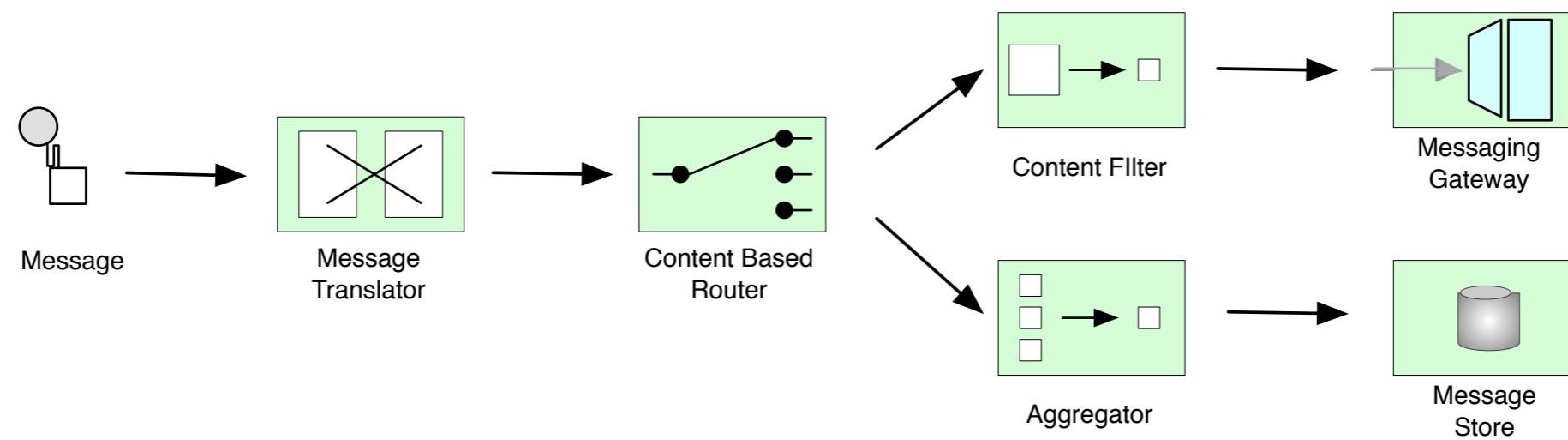
There is Still Time To Leave

- Technical cruise through SOA Platform 6
- Important high-level details
- Technical meat
- Live demonstration

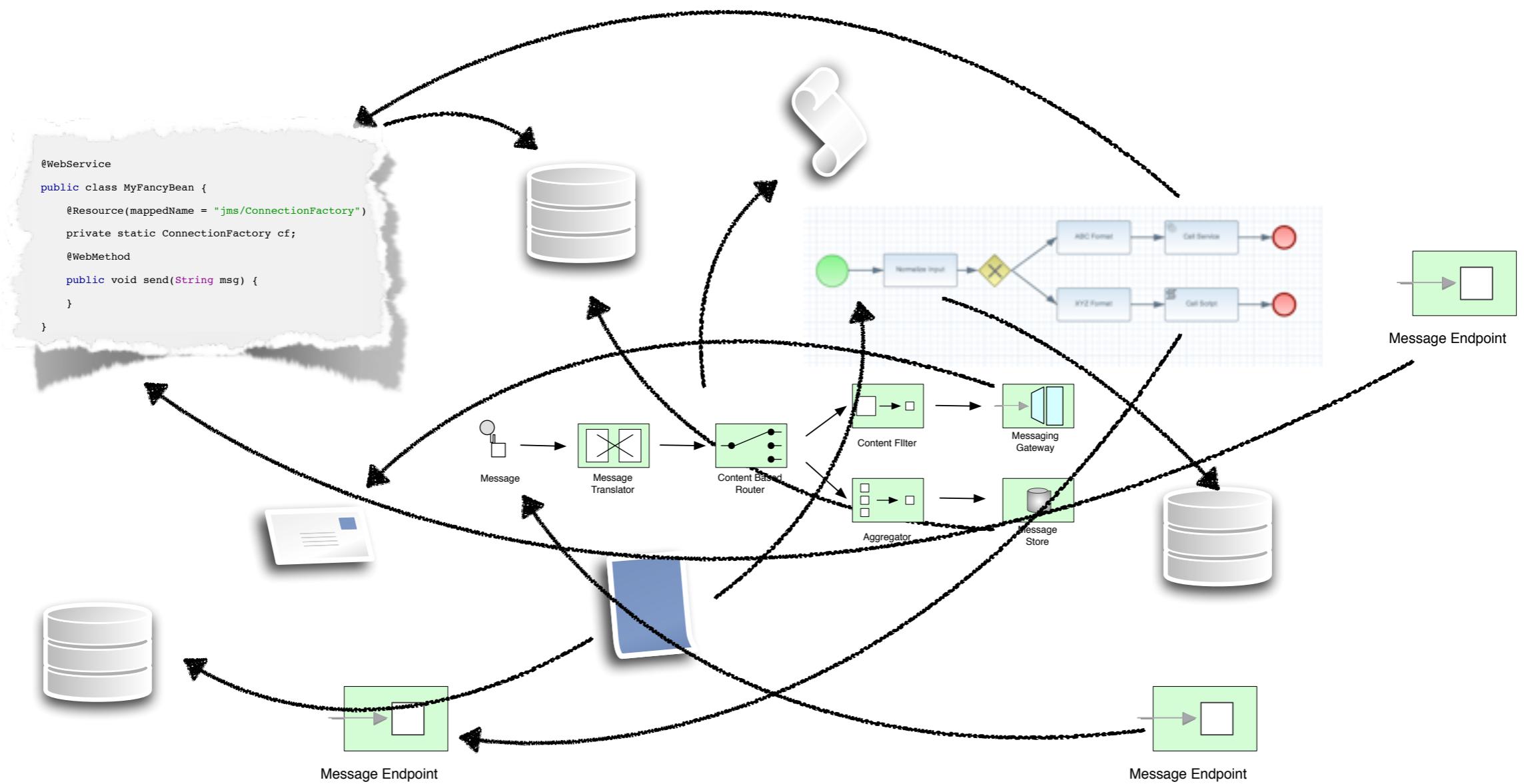
SOA 6 Platform Themes

- Structured, service-oriented development model
- Ease of use
- Technology refresh
- Governance
- Functional equivalence with SOA 5

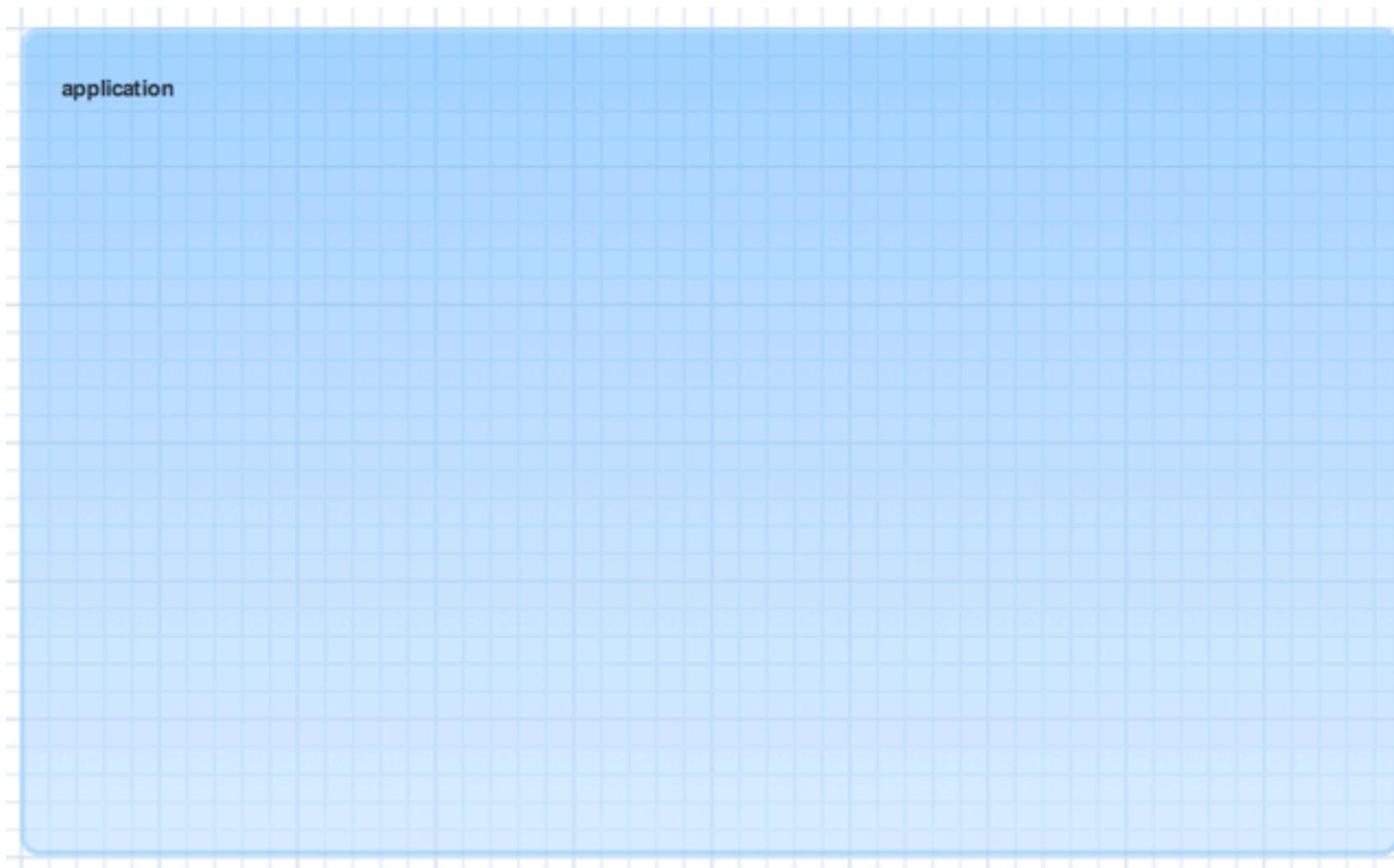
With Great Power



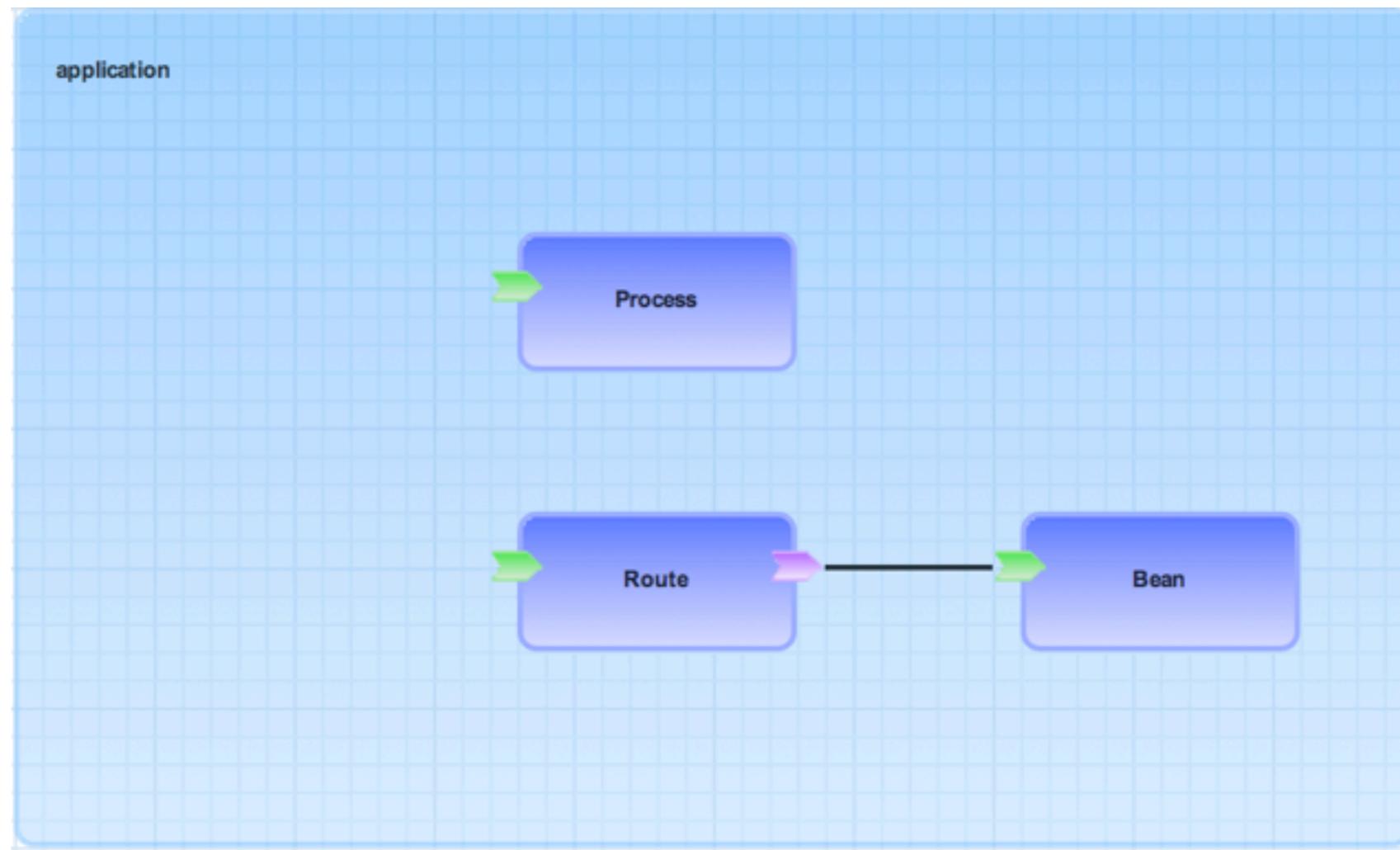
Comes Great Responsibility



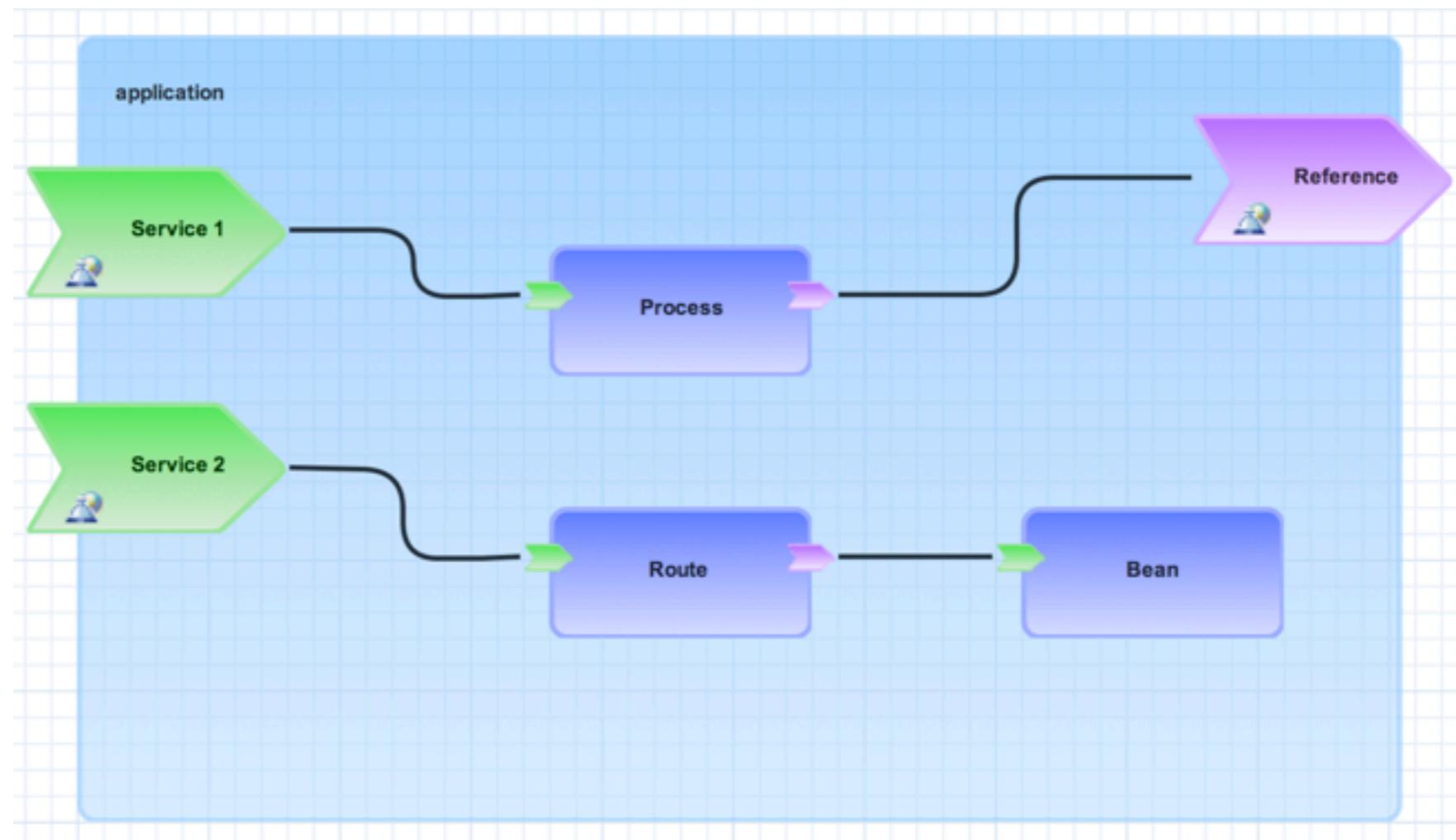
Structured Development



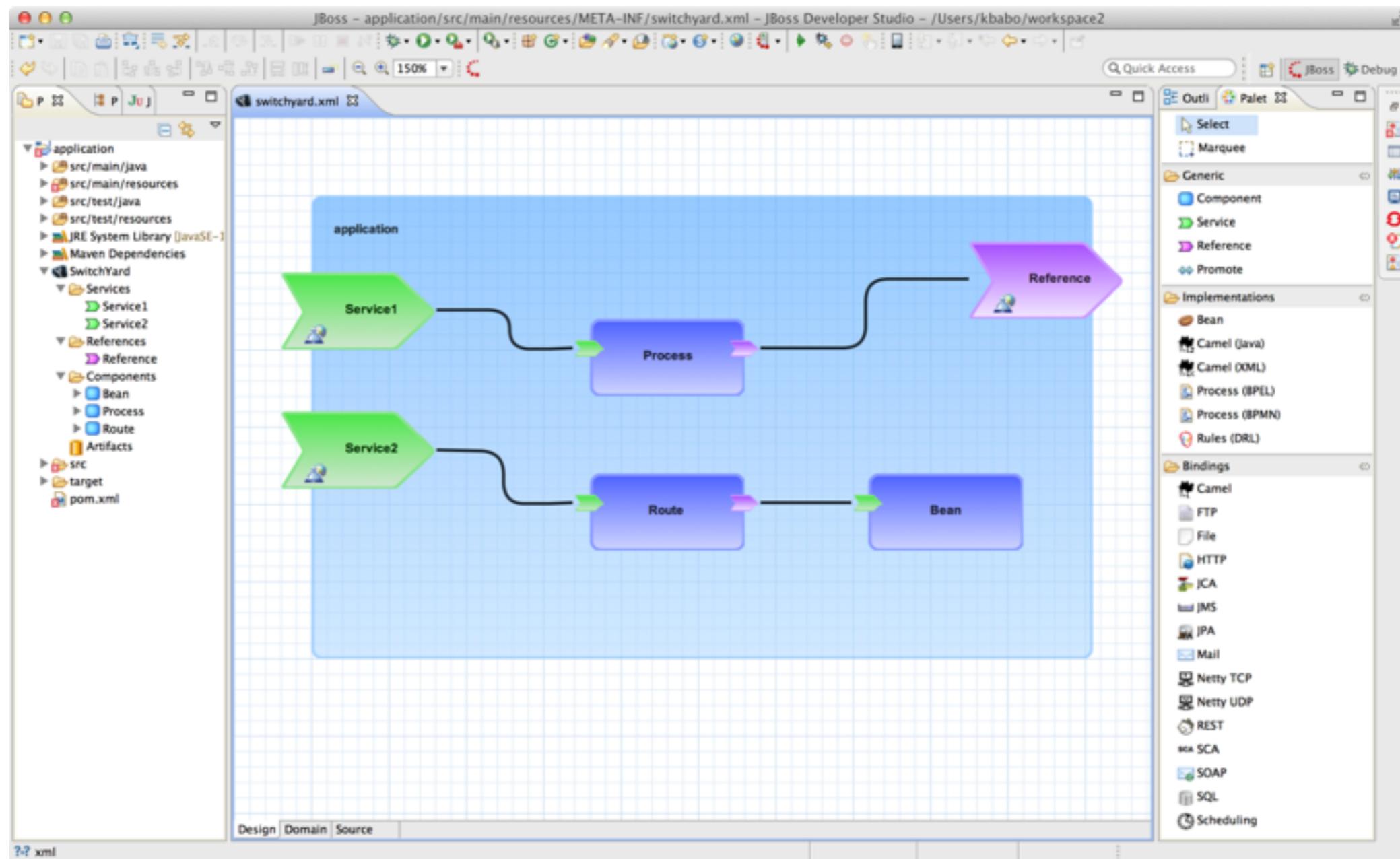
Structured Development



Structured Development



Ease of Use



#redhat #rhsummit



Technology Refresh

- EAP 6
- Java EE 6 / CDI
- Camel
- Maven
- OASIS (SCA, S-RAMP)

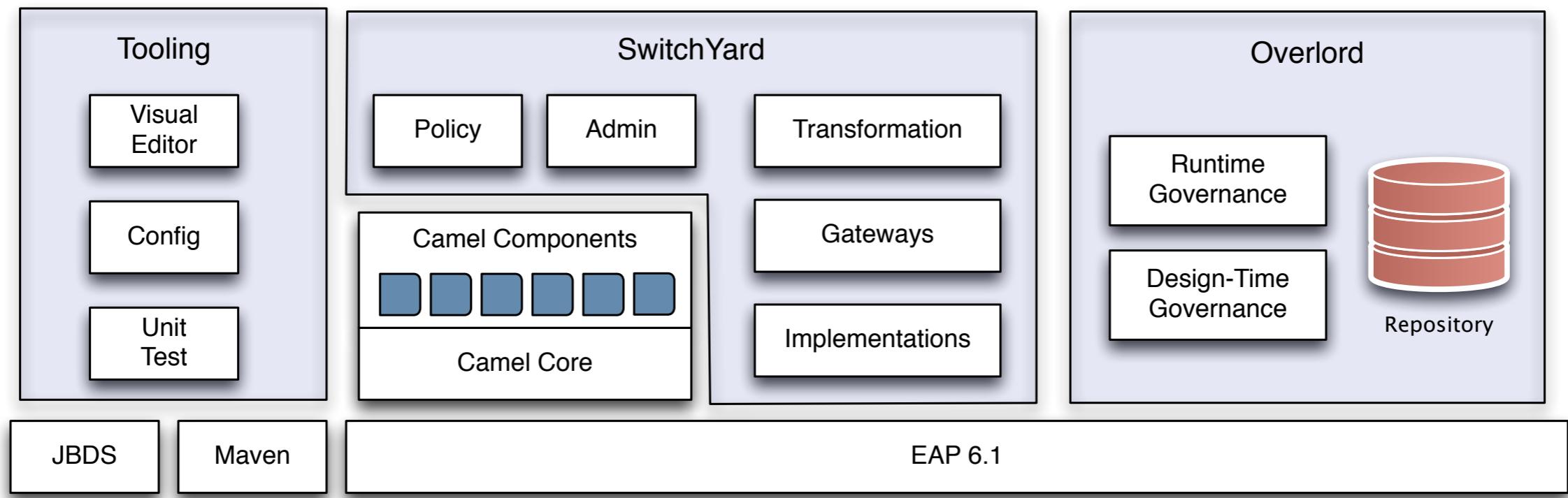
Governance

- SOA Repository
- Design Time Governance
- Runtime Governance

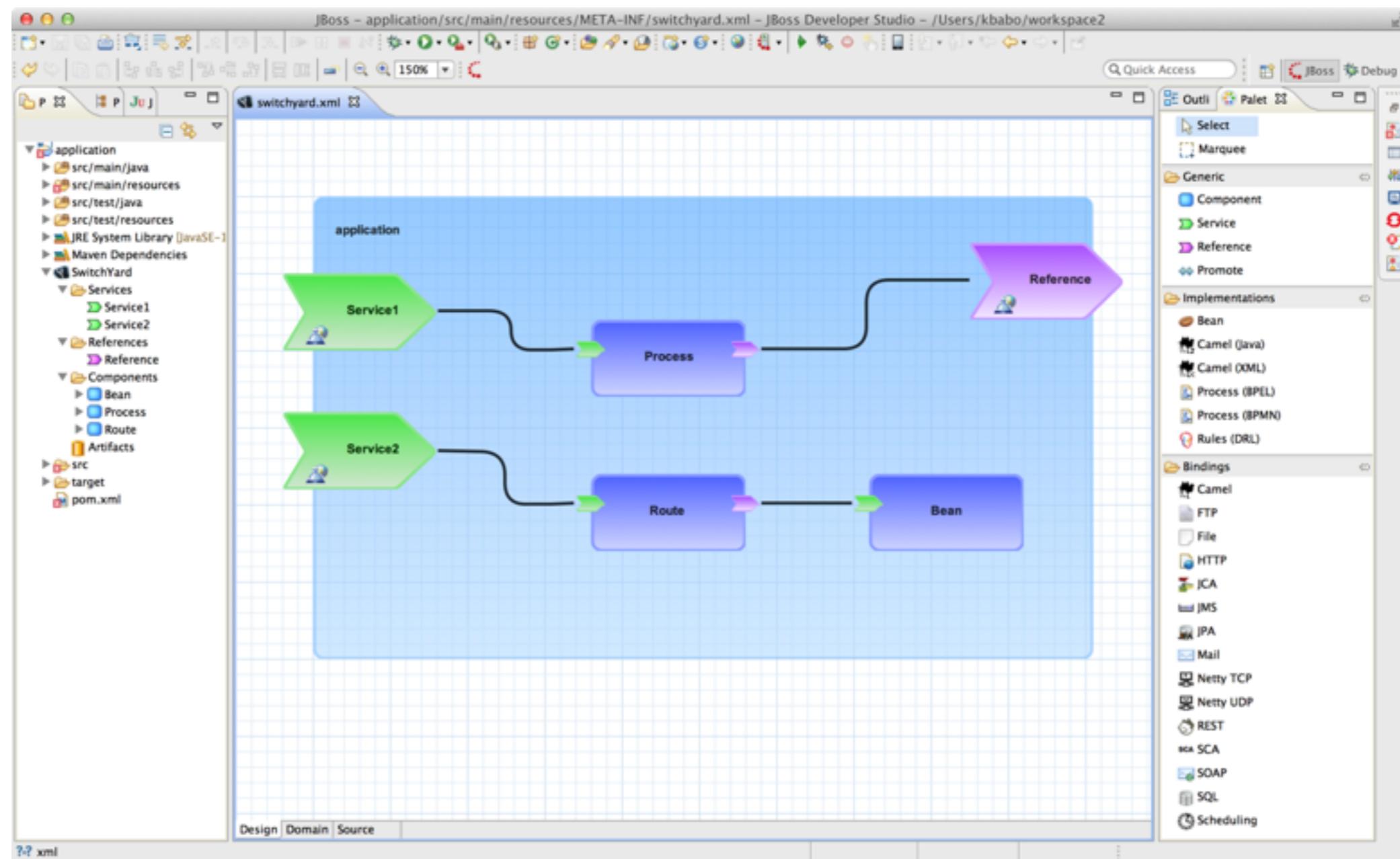
Relationship to SOA 5

- Functional equivalence to capabilities provided in SOA 5
- Conceptually similar in many ways
- Interoperability for side-by-side deployment
- Migration assistance

I Hate This Diagram



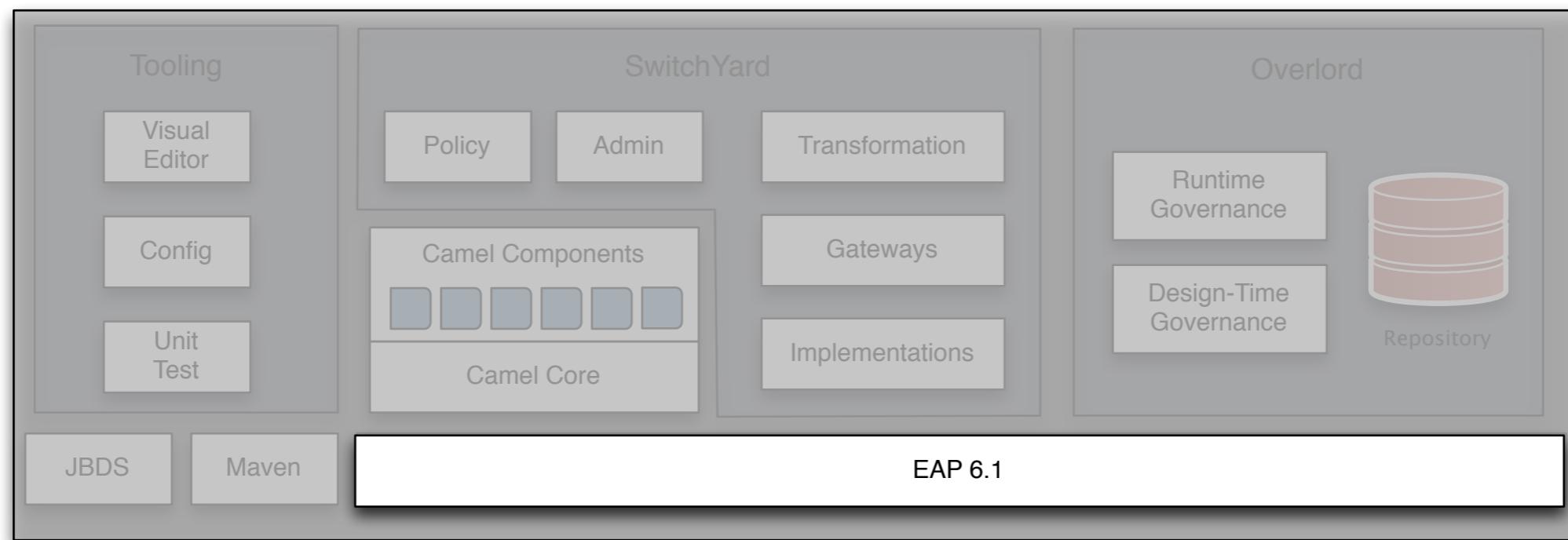
More My Speed



#redhat #rhsummit



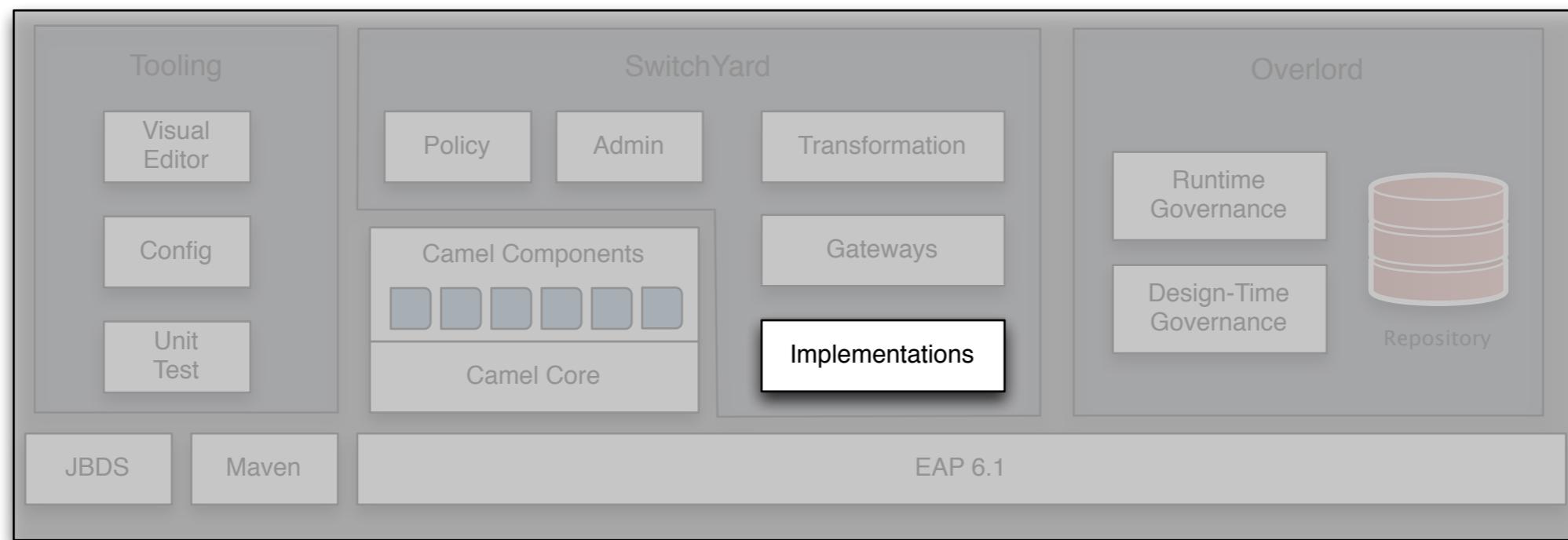
Runtime



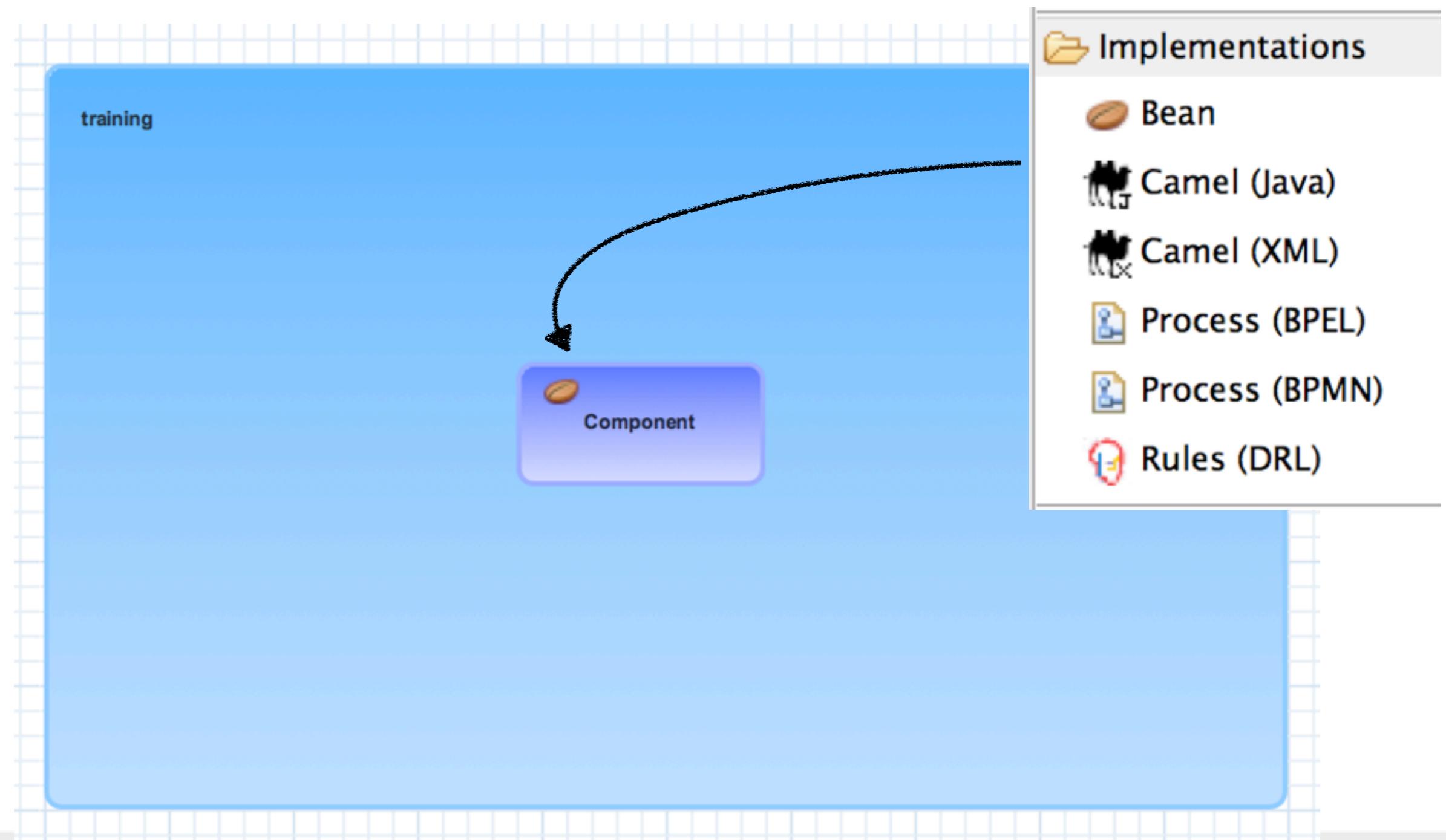
Runtime

- Layered Platform on EAP 6.1
- Standalone and Domain mode
- Subsystem and modules
- Deep integration
 - Administration
 - Security
 - Container services

Implementations



Implementations



Bean Implementations

- POJO = Service ... ‘nuff said
- Easy to use
 - Annotation-driven
 - No wrapper/routing logic
 - Service auto-registered
- Standard programming model (Java EE / JSR 299)
- Straightforward integration into the web tier

Bean Service

```
public class OrderServiceBean implements OrderService {  
  
    private InventoryService inventory;  
  
    public OrderAck submitOrder(Order order) {  
        // Check the inventory  
        Item orderItem = inventory.lookupItem(  
            order.getItemId());  
    }  
}
```

Bean Service

```
@Service(OrderService.class)
public class OrderServiceBean implements OrderService {

    private InventoryService inventory;

    public OrderAck submitOrder(Order order) {
        // Check the inventory
        Item orderItem = inventory.lookupItem(
            order.getItemId());
    }
}
```

Bean Service

```
@Service(OrderService.class)
public class OrderServiceBean implements OrderService {

    @Inject @Reference
    private InventoryService inventory;

    public OrderAck submitOrder(Order order) {
        // Check the inventory
        Item orderItem = inventory.lookupItem(
            order.getItemId());
    }
}
```

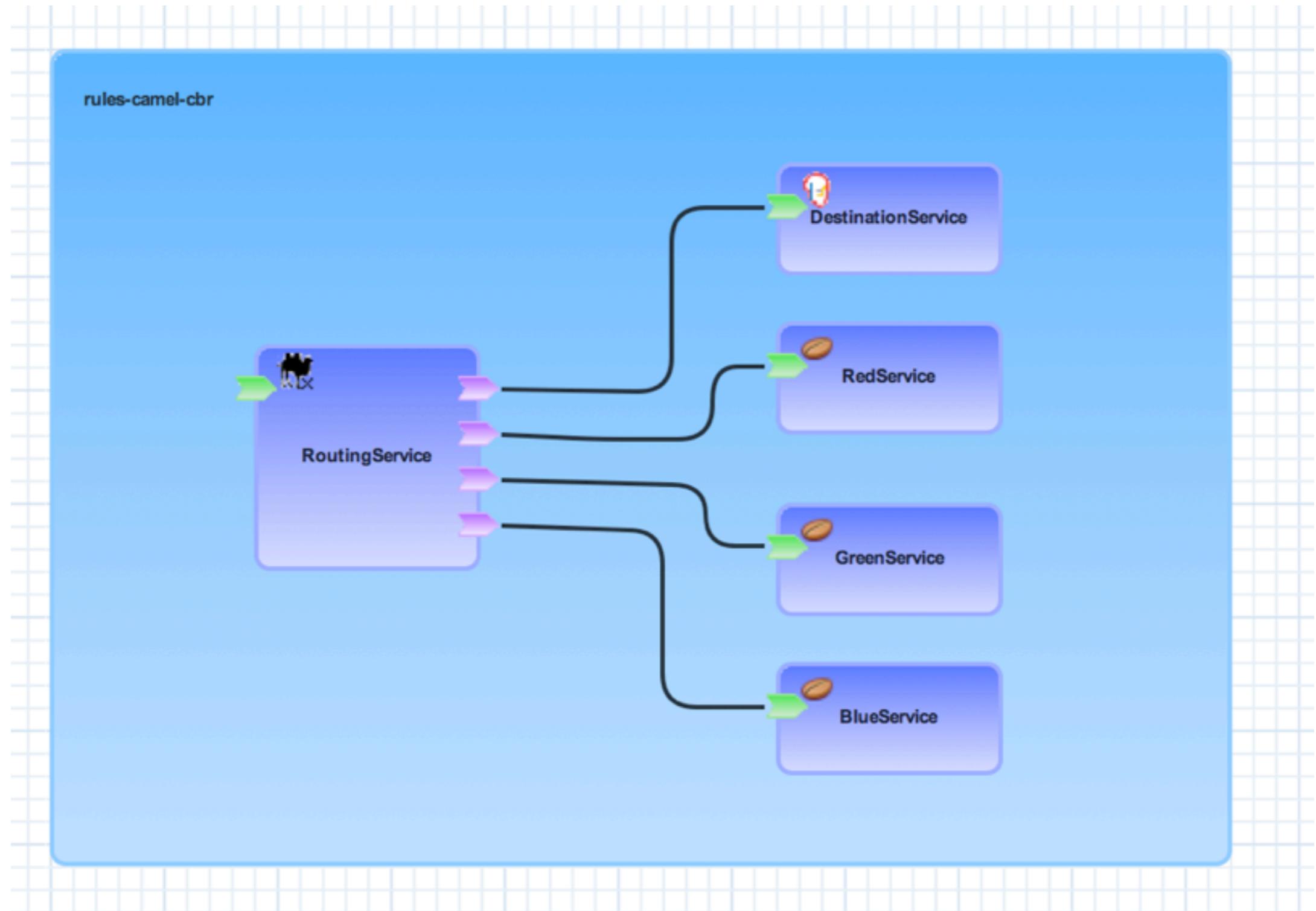
Camel Implementations

- Camel route as a service implementation
- Replaces action processing pipeline in JBoss ESB
- Use it for
 - Routing
 - Enterprise Integration Patterns
 - Scripting Languages
 - Data Formats

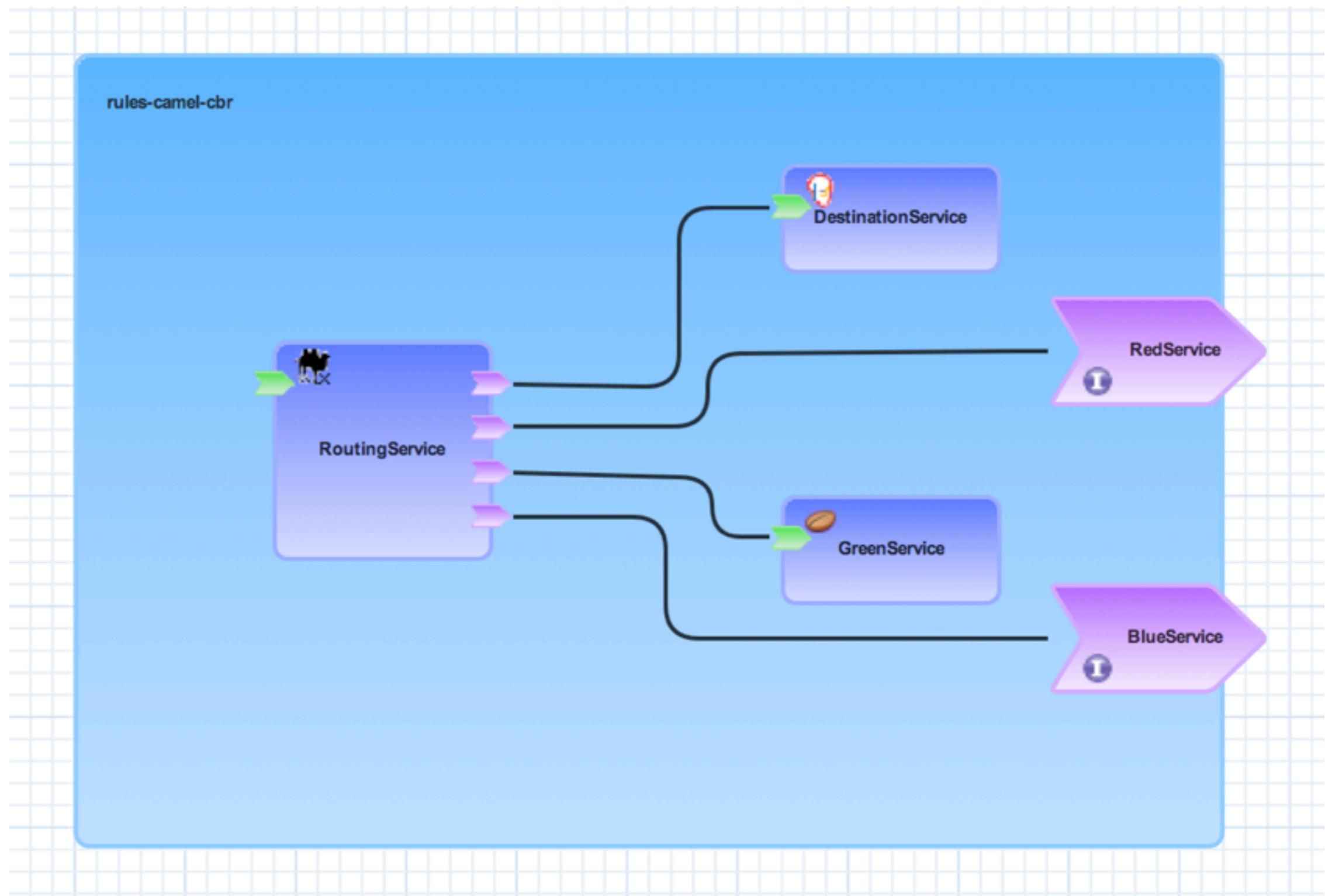
Camel Implementation

```
<route xmlns="http://camel.apache.org/schema/spring">
    <from uri="switchyard://RoutingService"/>
    <to uri="switchyard://DestinationService"/>
    <choice>
        <when>
            <simple>${body.destination} == 'Red'</simple>
            <to uri="switchyard://RedService"/>
        </when>
        <when>
            <simple>${body.destination} == 'Green'</simple>
            <to uri="switchyard://GreenService"/>
        </when>
        <when>
            <simple>${body.destination} == 'Blue'</simple>
            <to uri="switchyard://BlueService"/>
        </when>
        <otherwise>
            <log message="Unknown Destination!"/>
        </otherwise>
    </choice>
</route>
```

Local Reference



Remote Reference



Beans In Camel

- Allows Java objects to be called inside a route
- Very useful for fine-grained integration tasks
 - EIP configuration - route, split, etc.
 - Metadata access
 - Bolt-on logic
- Bean registry is pluggable

CDI Bean

```
import javax.enterprise.context.ApplicationScoped;
import javax.inject.Named;

@Named("MyBean")
@ApplicationScoped
public class SomeBean {

    public String doSomething(String content) {
        return "I did something with " + content;
    }
}
```

CDI Bean in Camel

```
import org.apache.camel.builder.RouteBuilder;

public class CamelServiceRoute extends RouteBuilder {

    public void configure() {
        from("switchyard://InventoryService")
            .split(body(String.class). tokenize("\n"))
            .filter(body(String.class). startsWith("item:"))
            .bean("bean:MyBean");
    }
}
```

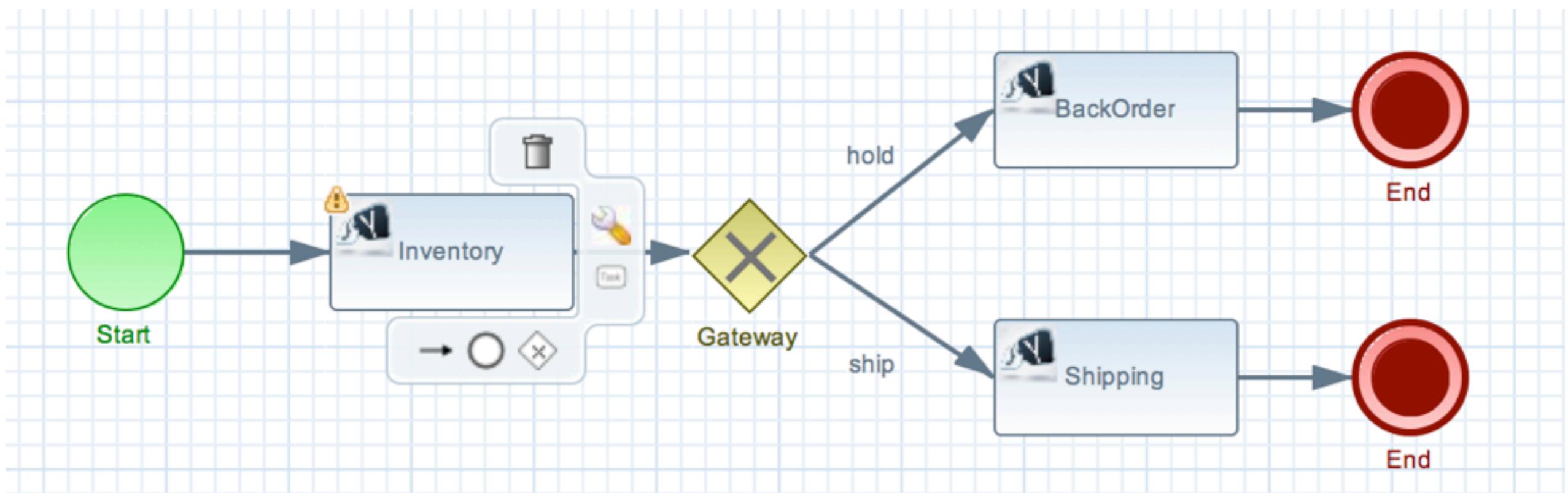
Rules Implementations

- Based on Drools 6
- Stateful and stateless sessions
- Complex Event Processing (with BRMS)
- Local and remote knowledge definitions
- Provide and consume services

BPMN Implementations

- Based on jBPM 6, integrates with BPM Platform
- Start and interact with process instances
- Provide and consume services
 - Standard BPMN 2 Service Task
 - Dynamic Service Task
- Fault handling
- Human task integration

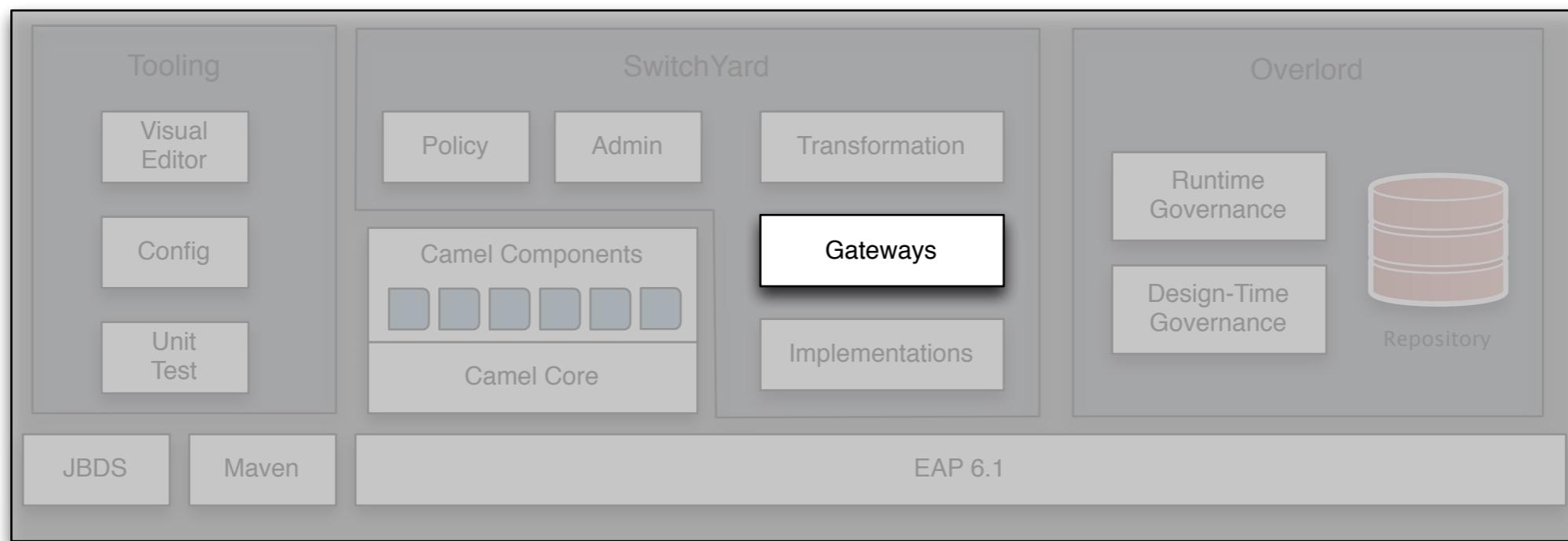
Service Orchestration with BPMN 2



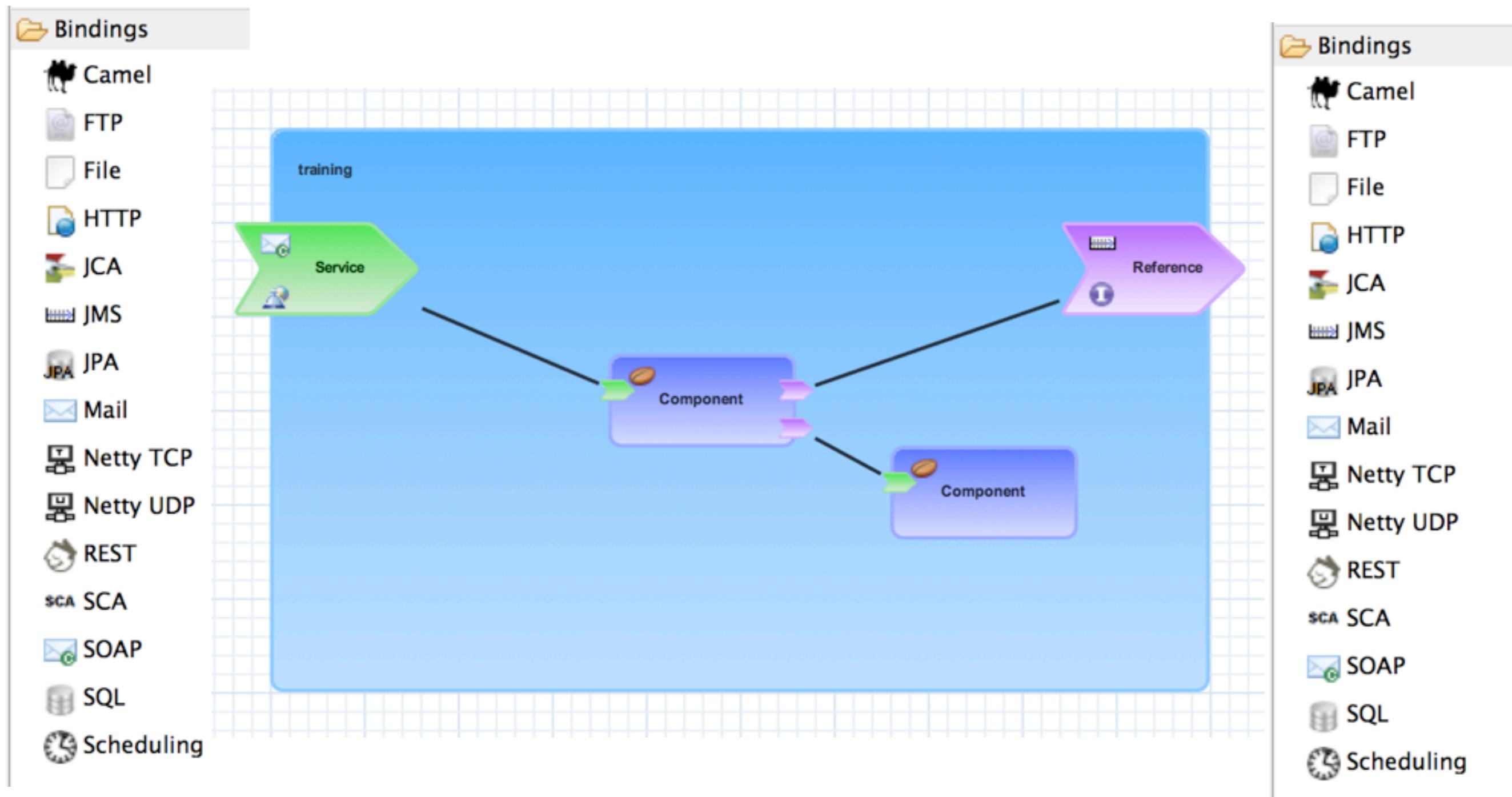
BPEL Implementations

- WS-BPEL 2.0 courtesy of Riftsaw
- Provide and consume from BPEL
- Combine with any gateway binding
- Native support for WSDL contracts

Gateways



Gateways



#redhat #rhsummit



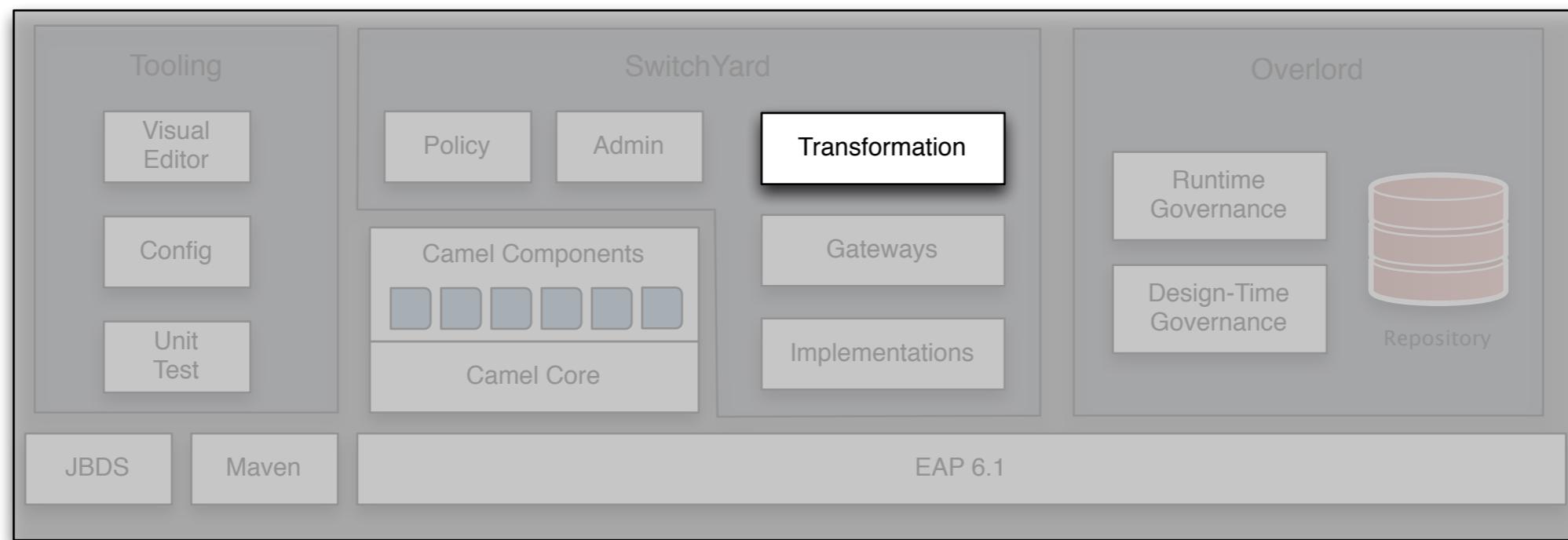
Gateways

- Enable communication outside the application through bindings
- Decoupled from service implementations
- Configuration defined by XML Schema
- Extensions supported through Camel

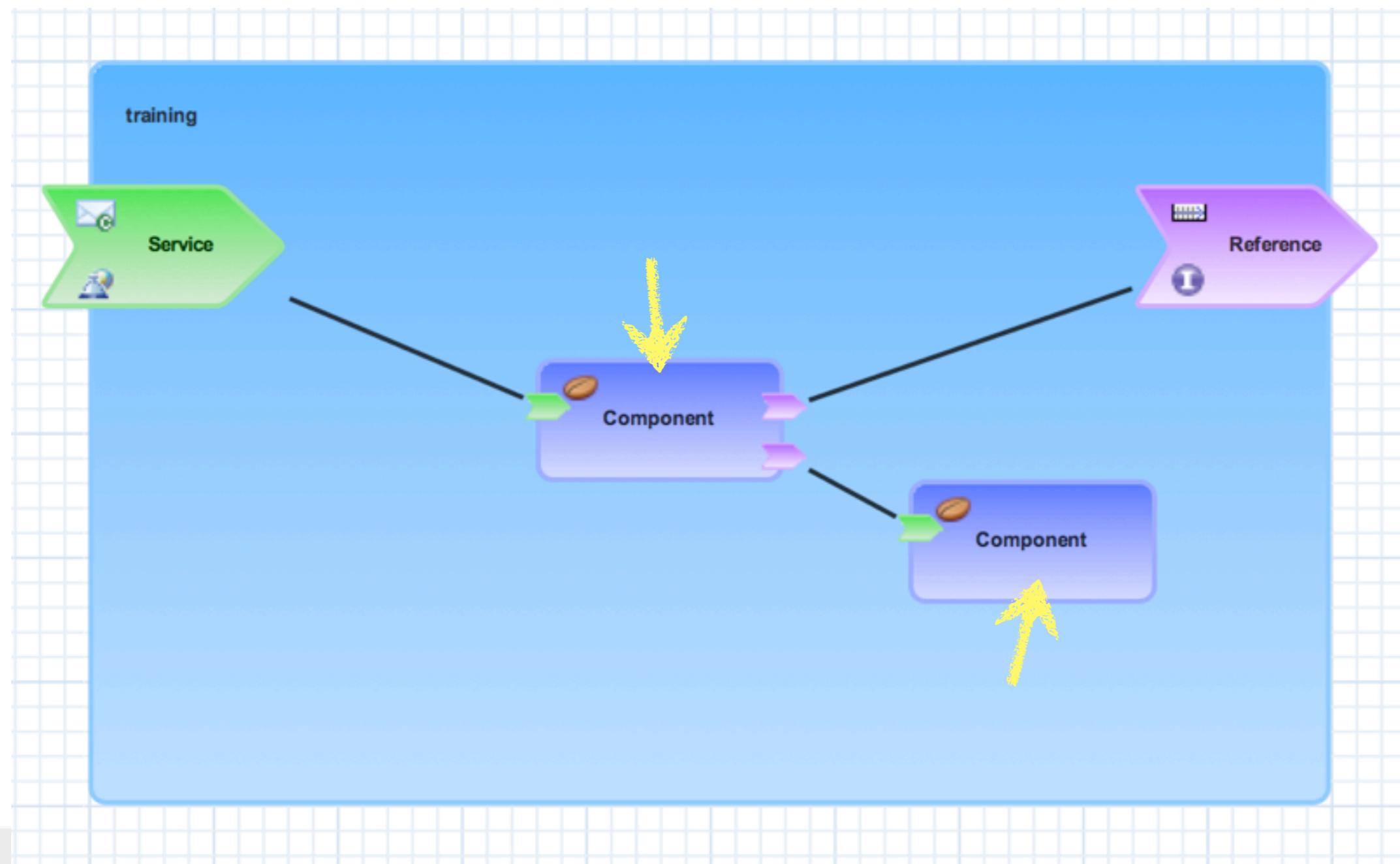
Extensions

- Camel components not included directly in SwitchYard
 - Fuse
 - Custom
- Using extensions
 - Create module(s)
 - Add to runtime
 - Configure via Camel URI gateway

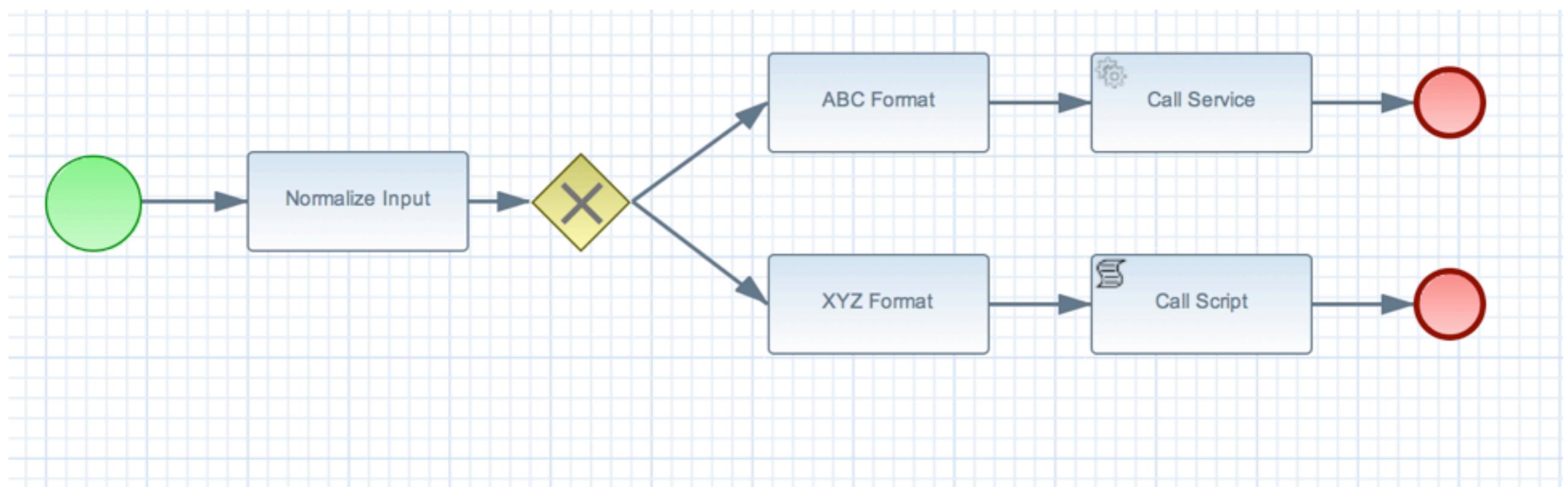
Transformation



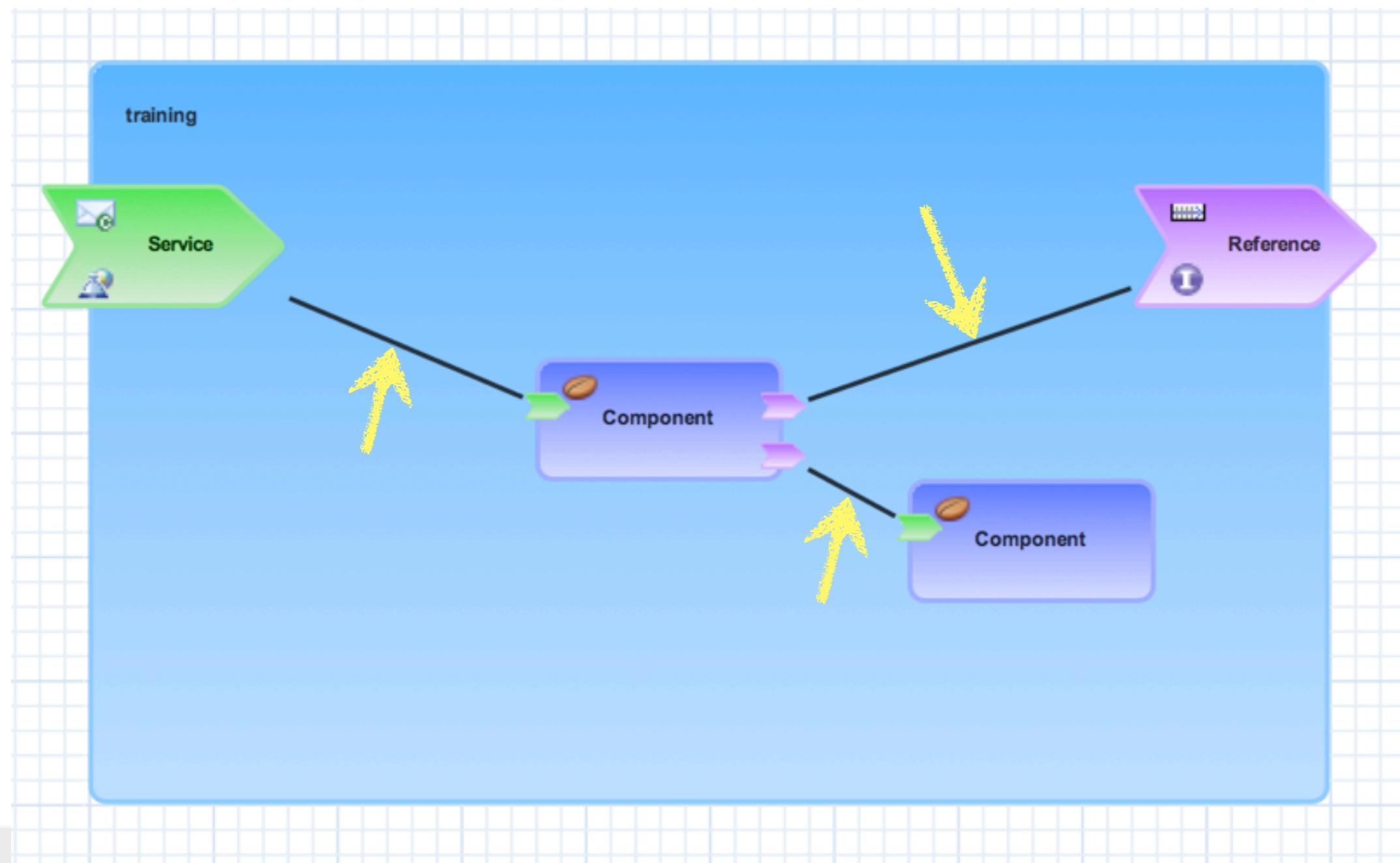
Procedural



Procedural in BPM

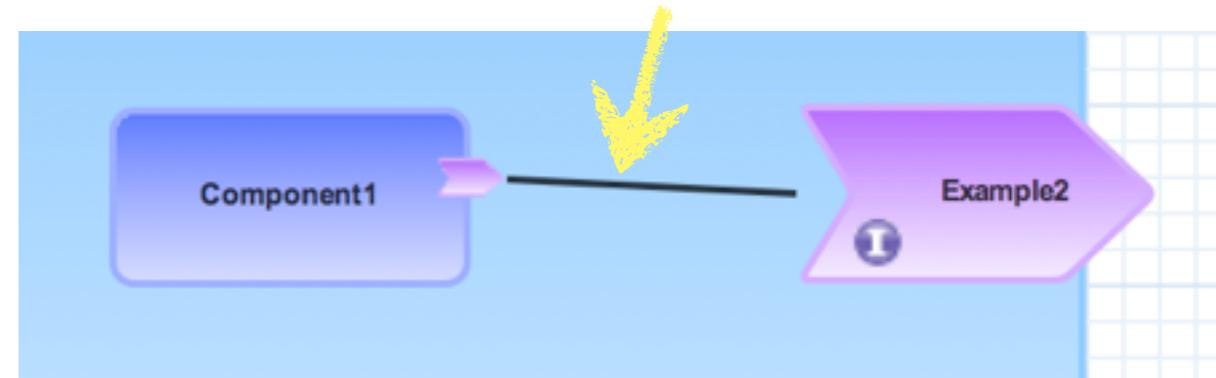
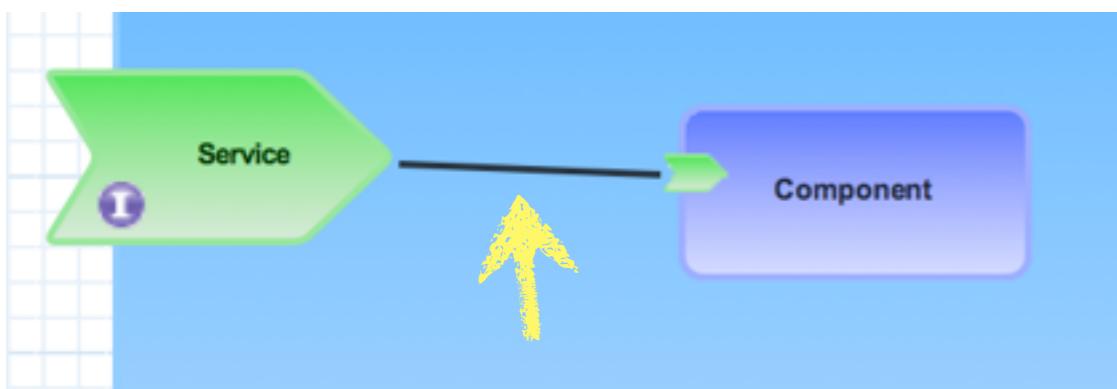


Declarative



Transformers

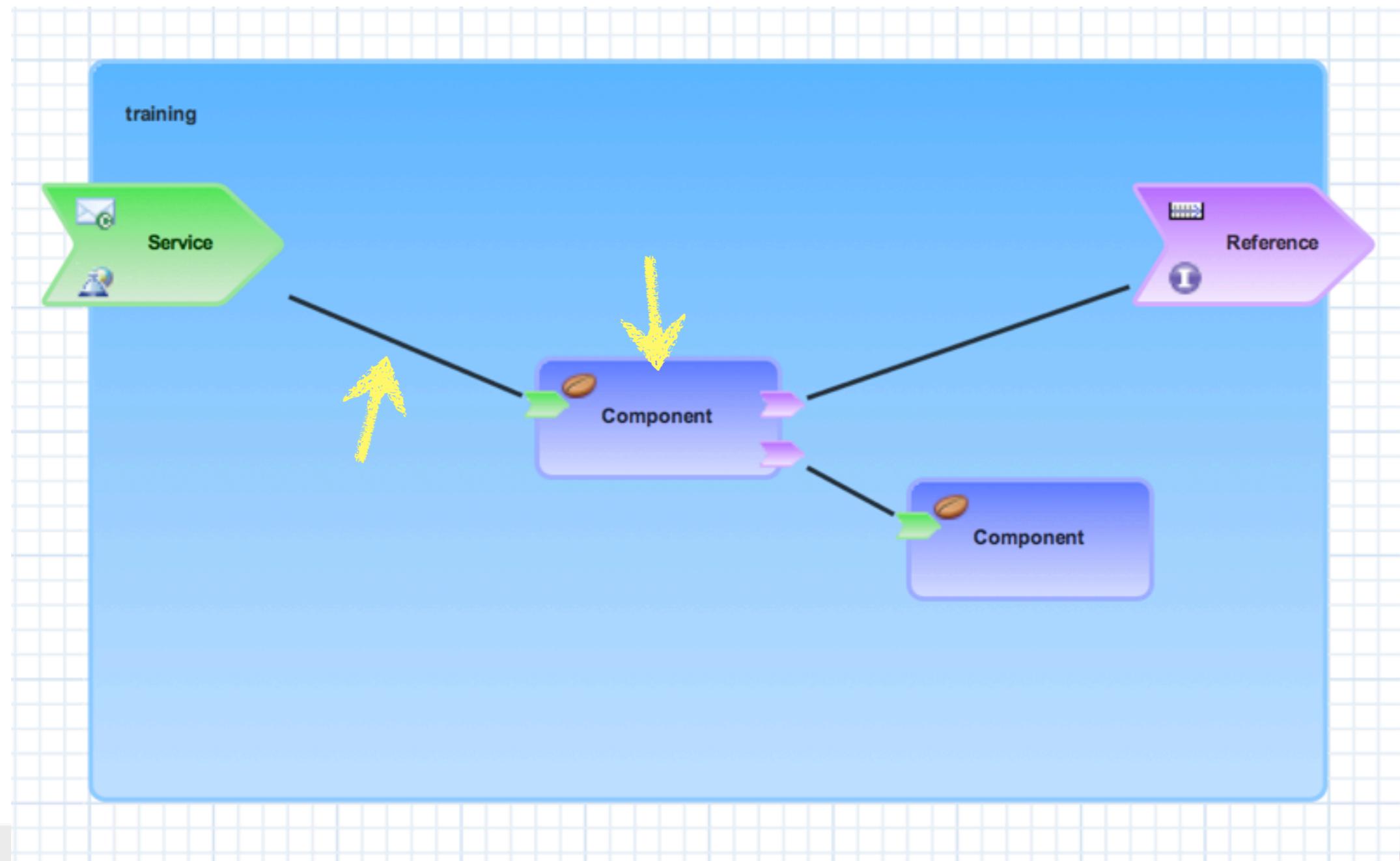
- Transformation is wired into SwitchYard core
 - Types declared via service contract
 - Transformer resolved dynamically at runtime
- Declarative, not procedural



Java Transformer

```
@Transformer(from = "{urn:switchyard-example:orders:1.0}submitOrder")
public Order transform(Element from) {
    return new Order()
        .setOrderId(getElementValue(from, "orderId"))
        .setItemId(getElementValue(from, "itemId"))
        .setQuantity(Integer.valueOf(getElementValue(from, "quantity")));
}
```

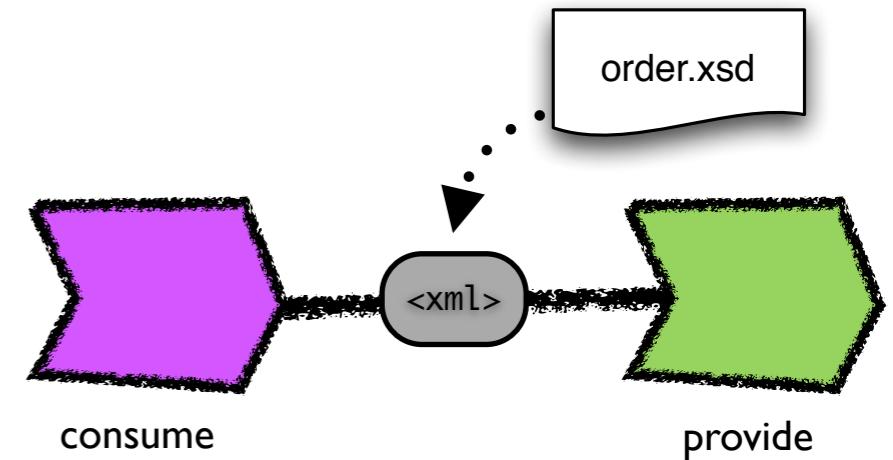
Hybrid



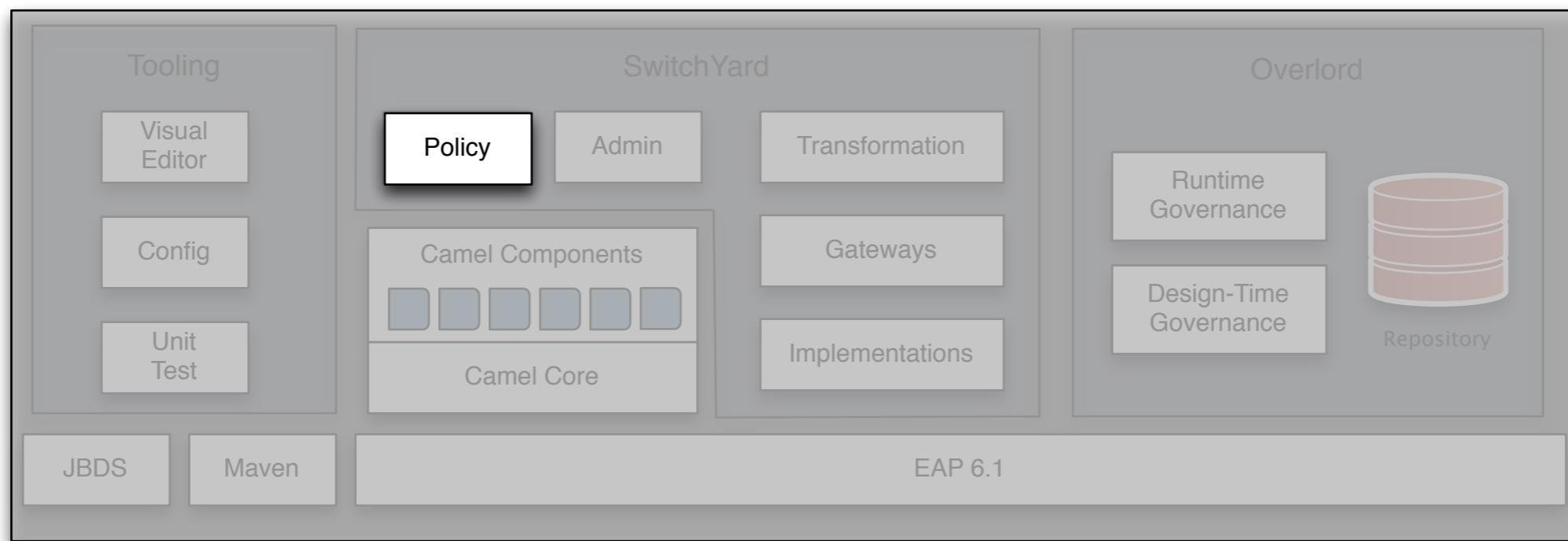
Validators

- Declarative validation
- Supports XML Schema and Java validation
- Executes pre and post transformation

```
<validate.xml  
    schemaType="XMLSCHEMA"  
    name="{urn:example:purchasing}order"  
    schemaFile="xsd/order.xsd"/>
```



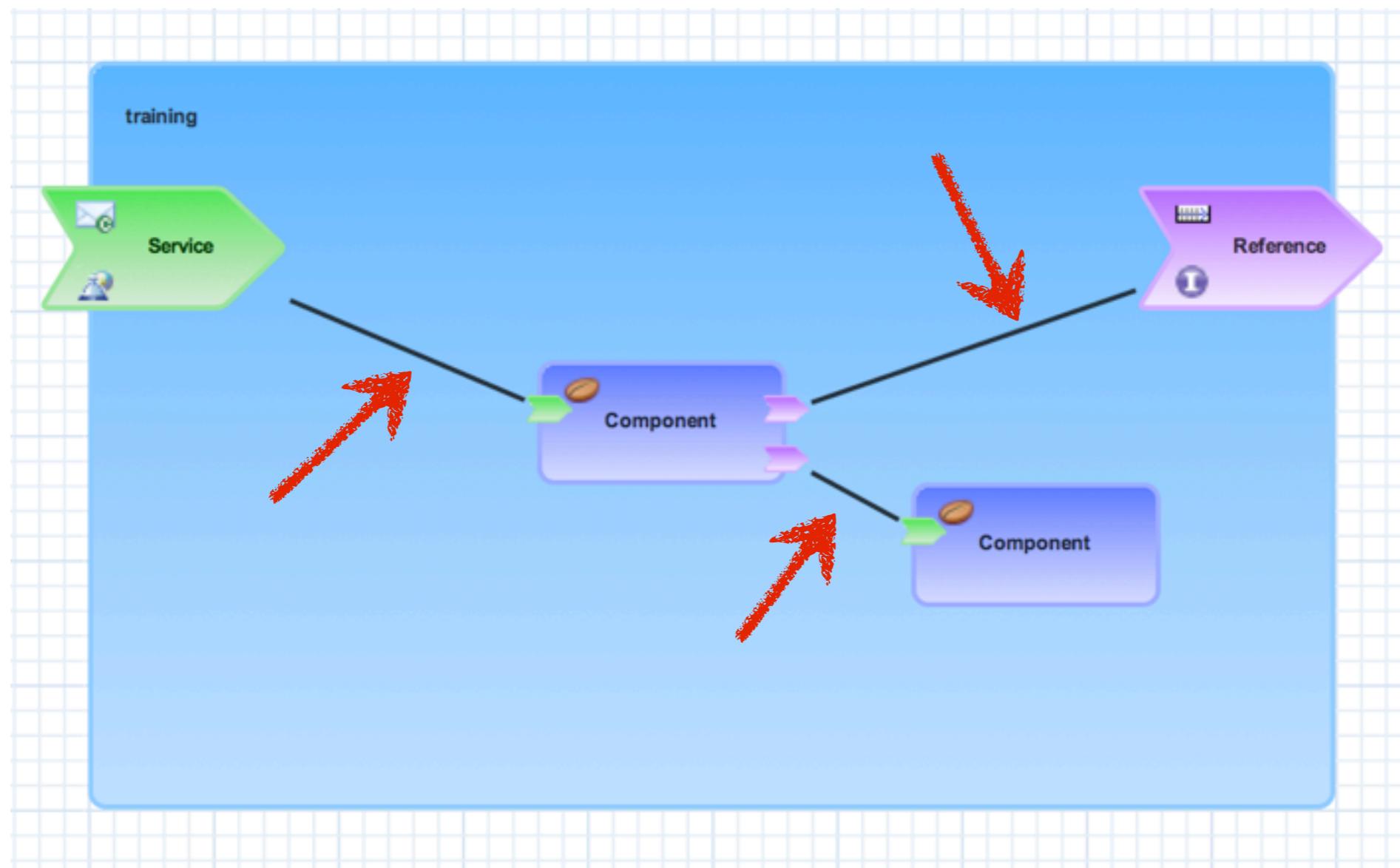
Policy



Declarative Policy

- Declare the ‘what’, defer the ‘how’
- Requirements attached to
 - Component service
 - Component reference
 - Implementation
- Runtime injects policy enforcement point

Policy Enforcement Points



Transaction Policy

- Interaction Policy
 - Suspend transaction
 - Require transaction
- Implementation Policy
 - Global transaction
 - Local transaction
 - No transaction

Interaction Policy

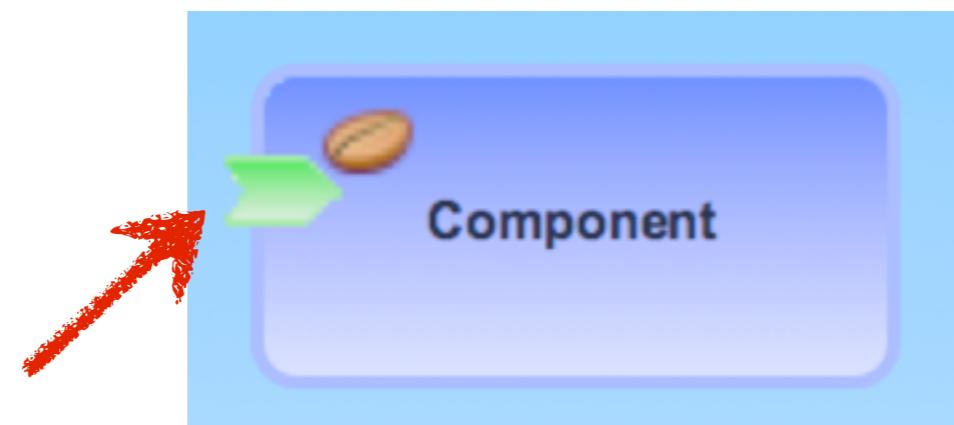


A screenshot of an IDE interface, likely Eclipse, showing the 'Properties' view. The 'Main' tab is selected in the left sidebar. The 'Policy Details' section is expanded, showing the 'Transaction Policy:' dropdown menu. The 'None' option is currently selected. Other options listed in the dropdown are 'propagatesTransaction' and 'suspendsTransaction'.

Security Policy

- Interaction policy only
 - Authorization
 - Client Authentication
 - Confidentiality
- Integrated with container
 - JAAS
 - PicketLink
 - WS-Security

Service Security

A screenshot of the Eclipse IDE interface. The top bar shows tabs: Problems, Properties (selected), Servers, Console, and OpenShift Explorer. On the left, there's a sidebar with categories: Main, Interface, Transaction Policy, and Security Policy (selected). The main content area is titled "Security Policy" and contains two checkboxes: "Client Authentication" and "Confidentiality".

Properties

Main

Interface

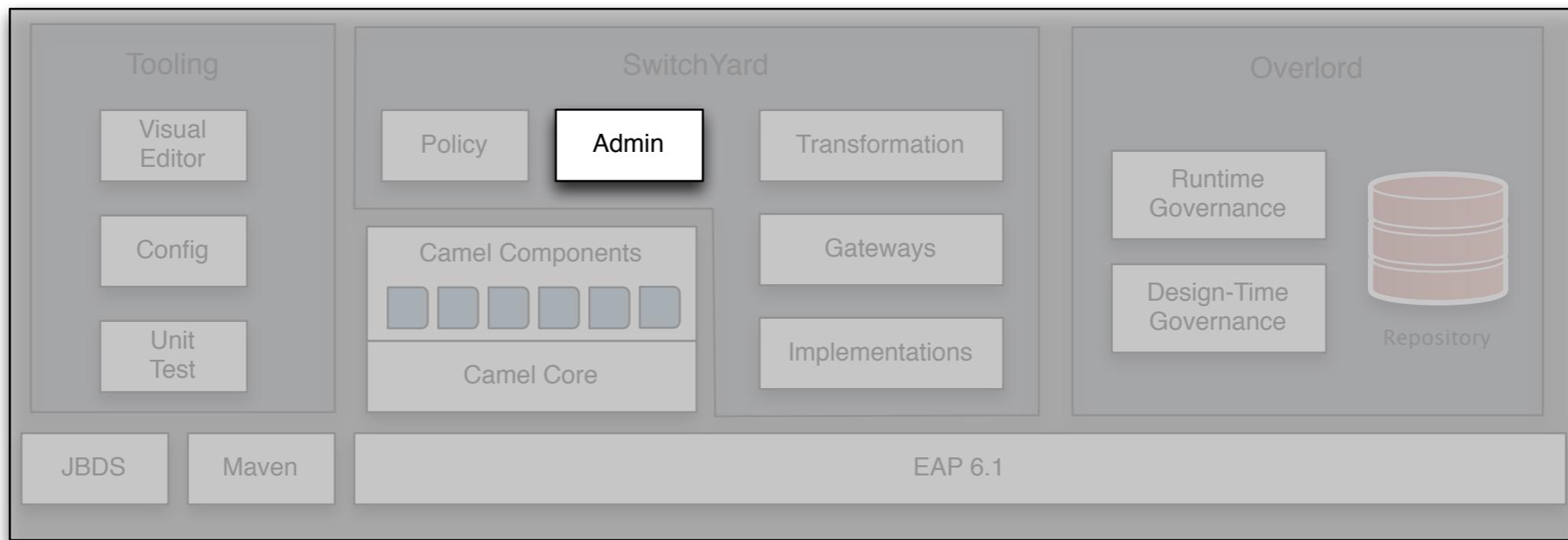
Transaction Policy

Security Policy

Client Authentication

Confidentiality

Administration



Administration Console

```
switchyard-quickstart-demo-multi-order-consumer
├── src/main/java
├── src/main/resources
├── src/test/java
├── src/test/resources
├── JRE System Library [JavaSE-1.6]
├── Maven Dependencies
└── SwitchYard
    ├── Services
    │   └── OrderInput
    ├── References
    │   └── OrderWebService
    ├── Components
    │   └── OrderInputComponent
    └── Artifacts
        └── OrderService
├── src
└── target
    └── order-consumer.jpg
    ├── pom.xml
    └── README.md
```

The screenshot shows the JBoss Application Server 7.1 Administration Console interface. On the left, there is a sidebar with various configuration sections like Profile, Core, Connector, Messaging, Container, Security, Web, OSGi, Infinispan, and SwitchYard. The 'SwitchYard' section is expanded, and 'Applications' is selected. The main content area displays a table of deployed applications:

Name	Target Namespace
consumer-service	urn:switchyard-quickstart-demo:multiapp:0.1.0
orders	urn:switchyard-quickstart-demo:multiapp:0.1.0
web	urn:switchyard-quickstart-demo:multiapp:0.1.0

Below the table, there is an 'Application Details' section with fields for 'Application Name' (set to 'orders') and 'Application Namespace' (set to 'urn:switchyard-quickstart-demo:multiapp:0.1.0'). There are tabs for 'Services', 'Artifact References', and 'Transformers'. Under 'Artifact References', there is a table:

Name	URL
OrderService	http://localhost:8080/guvnorsoa/rest/packages/OrderService

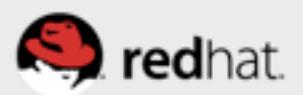
Administration Console

The screenshot shows the JBoss Management interface for JBoss Application Server 7.1. The URL in the browser is <http://localhost:9990/console/App.html#sy-artifacts>. The left sidebar has a tree view under 'Profile' with nodes like Core, Connector, JCA, Datasources, Resource Adapters, Mail, Messaging, Container, Security, Web, OSGi, Infinispan, and SwitchYard. Under SwitchYard, 'Artifact References' is selected. The main content area is titled 'SwitchYard Artifact References' and contains two tables. The first table, 'Artifact References', shows a single entry for 'OrderService' with the URL <http://localhost:8080/guvnorsoa/rest/packages/OrderService>. The second table, 'Applications Using Artifact', shows three entries: 'consumer-service' with target namespace `urn:switchyard-quickstart-demo:multiapp:0.1.0`, 'orders' with target namespace `urn:switchyard-quickstart-demo:multiapp:0.1.0`, and 'web' with target namespace `urn:switchyard-quickstart-demo:multiapp:0.1.0`.

Name	URL
OrderService	http://localhost:8080/guvnorsoa/rest/packages/OrderService

Name	Target Namespace
consumer-service	<code>urn:switchyard-quickstart-demo:multiapp:0.1.0</code>
orders	<code>urn:switchyard-quickstart-demo:multiapp:0.1.0</code>
web	<code>urn:switchyard-quickstart-demo:multiapp:0.1.0</code>

#redhat #rhsummit



Service Metrics

The screenshot shows the JBoss Management interface for JBoss Application Server 7.1. The left sidebar contains navigation links for Server Status, Configuration, JVM, Subsystem Metrics (with SwitchYard selected), Datasources, JPA, JMS Destinations, Transactions, Web, Runtime Operations, OSGi, and Deployments. The main content area is titled "JBoss Application Server 7.1" and "SwitchYard Message Metrics". It displays message metrics for individual services. A table lists services with their names and target namespaces:

Name	Target Namespace
loanService	urn:bpel:test:1.0
OrderService	urn:switchyard-quickstart:bean-service:0.1.0
ProcessOrder	urn:switchyard-quickstart:bpm-service:1.0
RedService	urn:switchyard-quickstart:rules-camel-cbr:0.1.0
RoutingService	urn:switchyard-quickstart:rules-camel-cbr:0.1.0

Below the table, there are sections for "Message Counts" and "Processing Times", each with a table and corresponding bar charts.

Message Counts:

Metric	Actual
Total Count:	381 33%
Success Count:	381 100%
Fault Count:	0 0%

Processing Times:

Metric	Actual
Total Processing Time:	292794 50%
Average Processing Time:	768.488188976378
Min. Processing Time:	5
Max. Processing Time:	1992

Reference Metrics

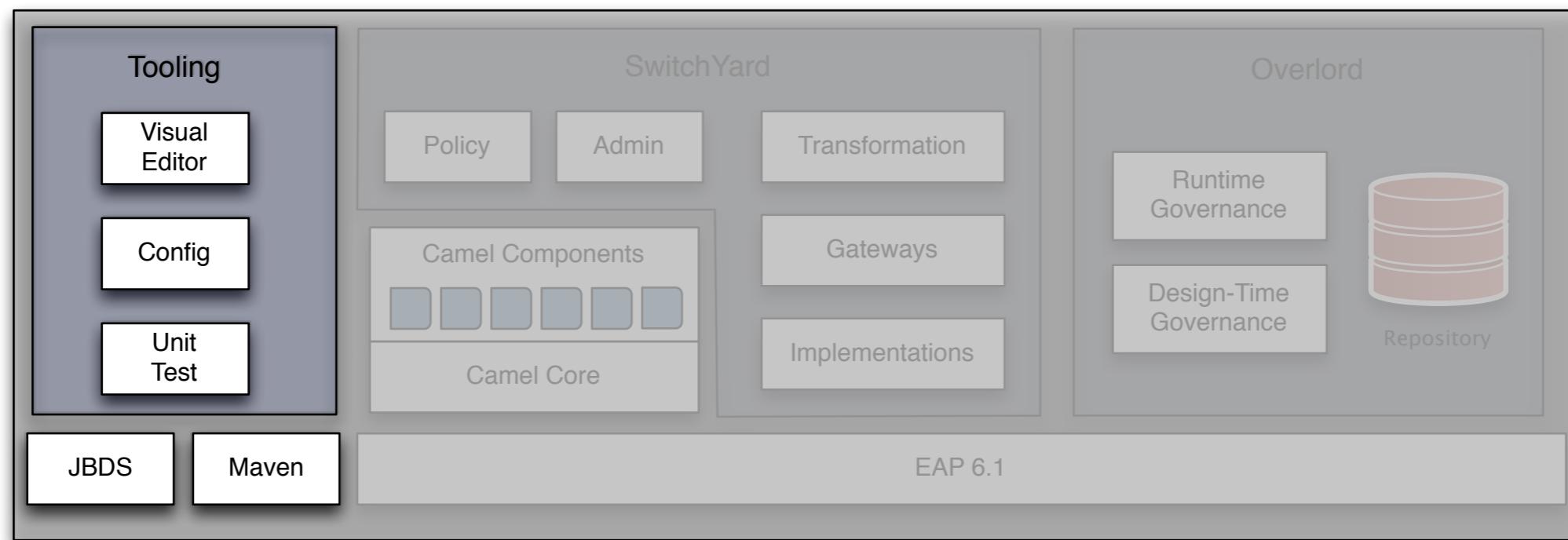
Service Metrics Reference Metrics

Referenced Service Metrics

Name	Message Count	Average Time	Time %	Fault %
BlueService	0	0	0%	0%
DestinationService	382	0.1518324607329843	0%	0%
GreenService	191	49.74869109947644	0%	0%
RedService	190	1483.9315789473685	96%	0%

1-4 of 4

Tooling



**RED HAT
SUMMIT**

Demo



SOA Governance

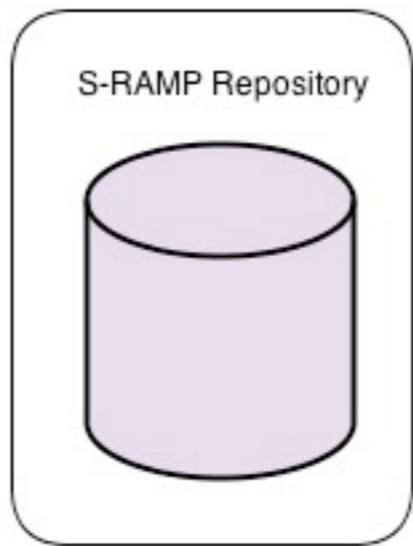
- Processes to manage delivery of Service Oriented Architecture (SOA)
 - Compliance (e.g. Sarbanes-Oxley)
 - Artifact Lifecycle
 - Inception through to Retirement
 - Policy enforcement
 - Monitoring
 - Auditing

Overlord

- Umbrella project for SOA Governance projects
 - <http://www.jboss.org/overlord>
 - <https://github.com/governance>
- Primary projects are
 - S-RAMP (SOA Repository)
 - DTGov (Design Time Governance)
 - RTGov (Runtime Governance)



S-RAMP



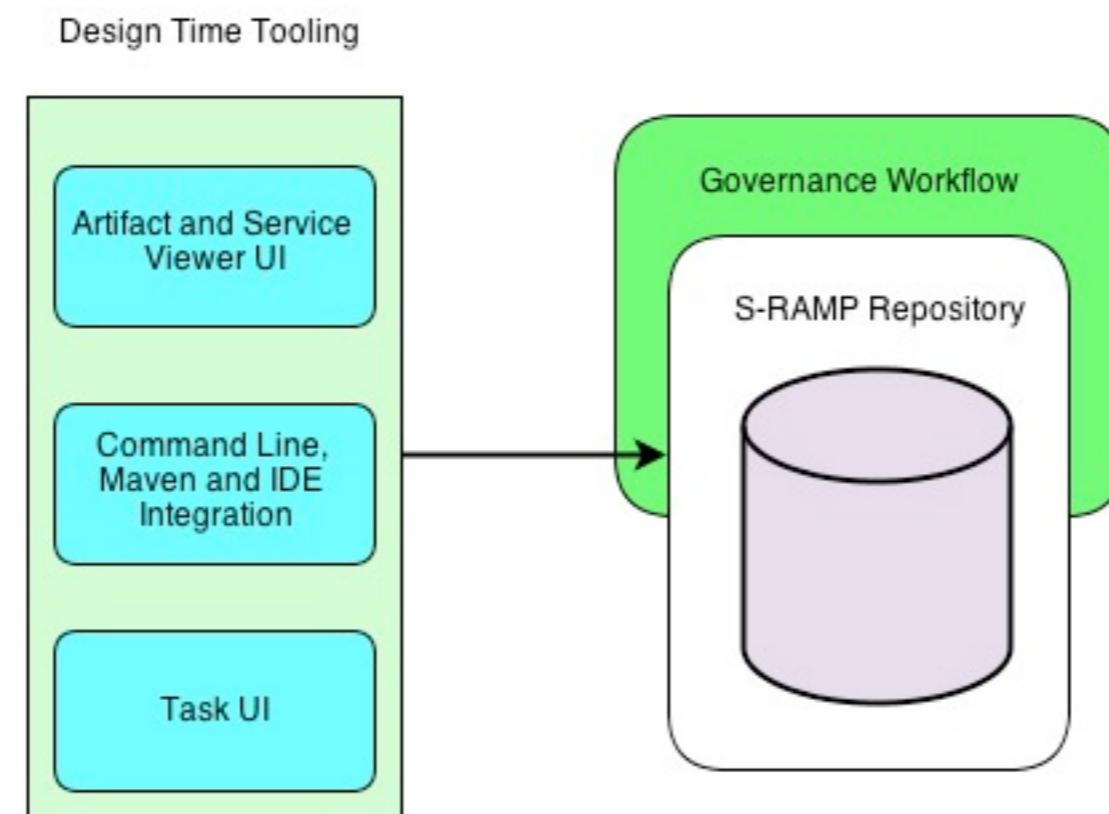
S-RAMP

- SOA Repository Artifact Model and Protocol
 - OASIS Technical Committee
- Common Data Model for SOA
- Interaction protocol
 - ATOM REST
- Enables
 - Common Tooling
 - Data Sharing

S-RAMP

- Central Artifact Repository
- Extract Metadata
 - Classifiers
 - Relationships
- Event Notification
- ModeShape (JCR) + Infinispan

Design Time Governance



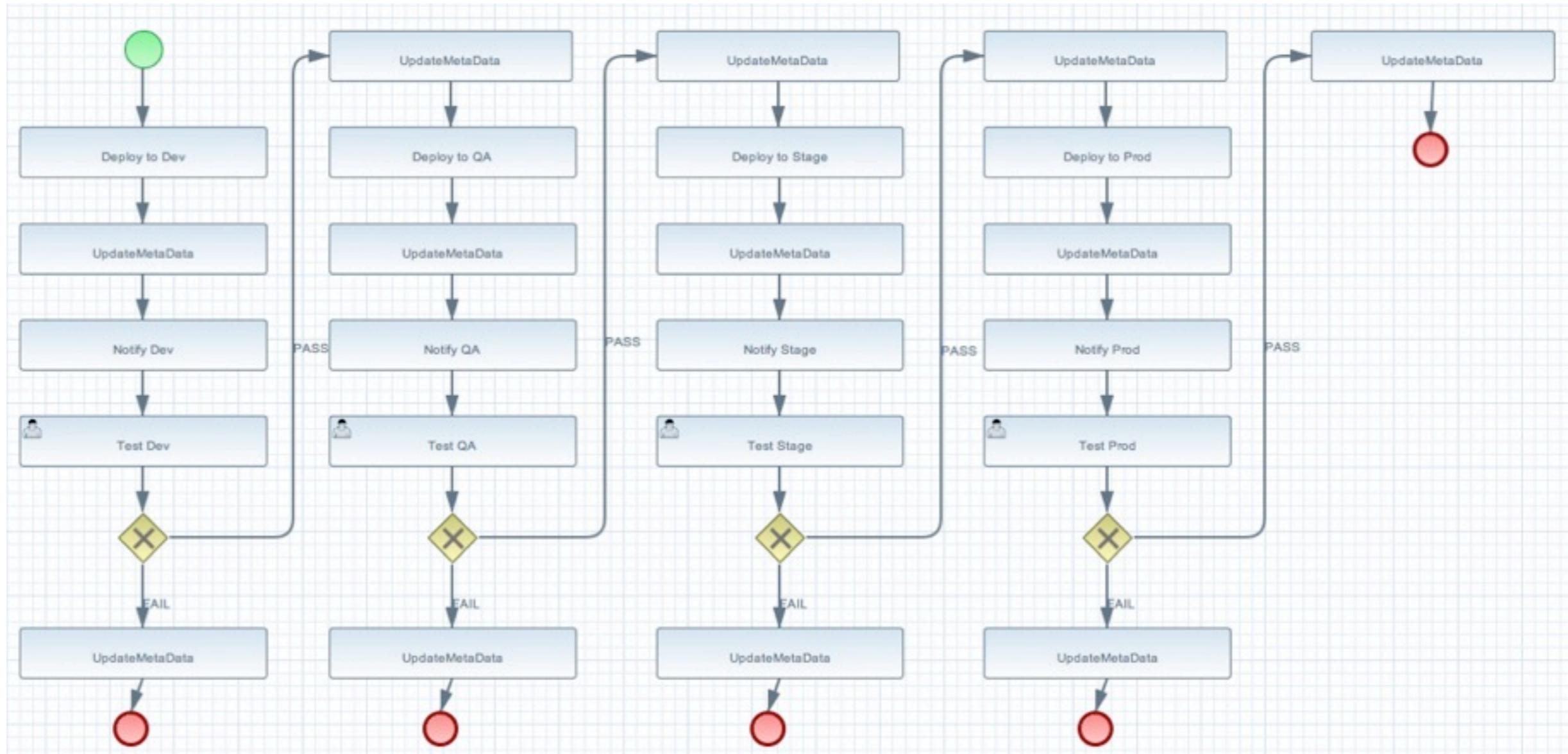
Design Time Governance

- Artifact Discovery
- Artifact Lifecycle
 - Services
 - Policies
- Relationship Visualisation
- Change Notification
- Impact Analysis

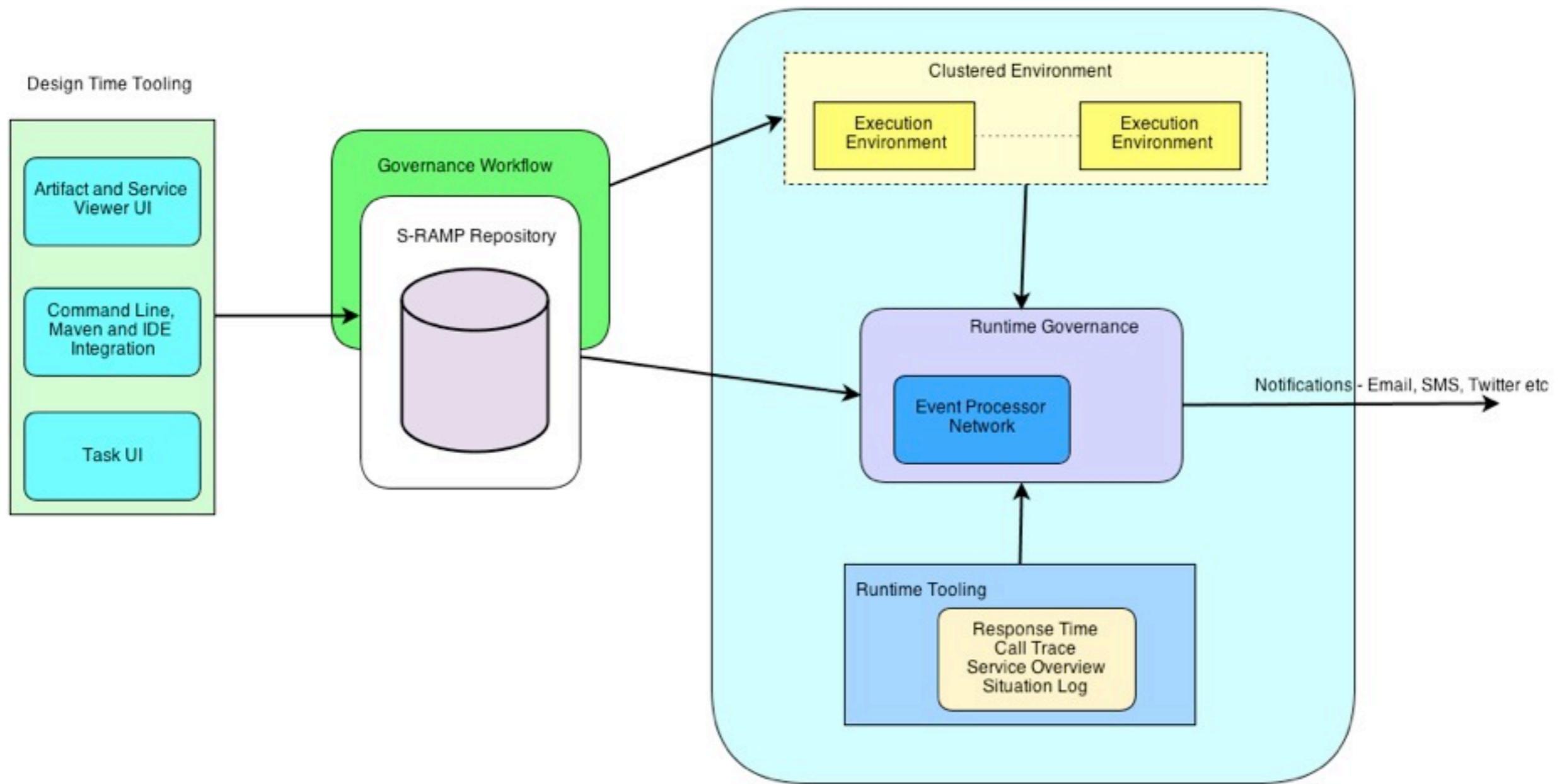
Design Time Governance

- Workflow Integration (BPMN2)
 - Triggered by S-RAMP
 - Manage Artifact Lifecycle
 - Deploy to Execution Environment (direct, JON)
- Task UI

Design Time Governance



Runtime Governance



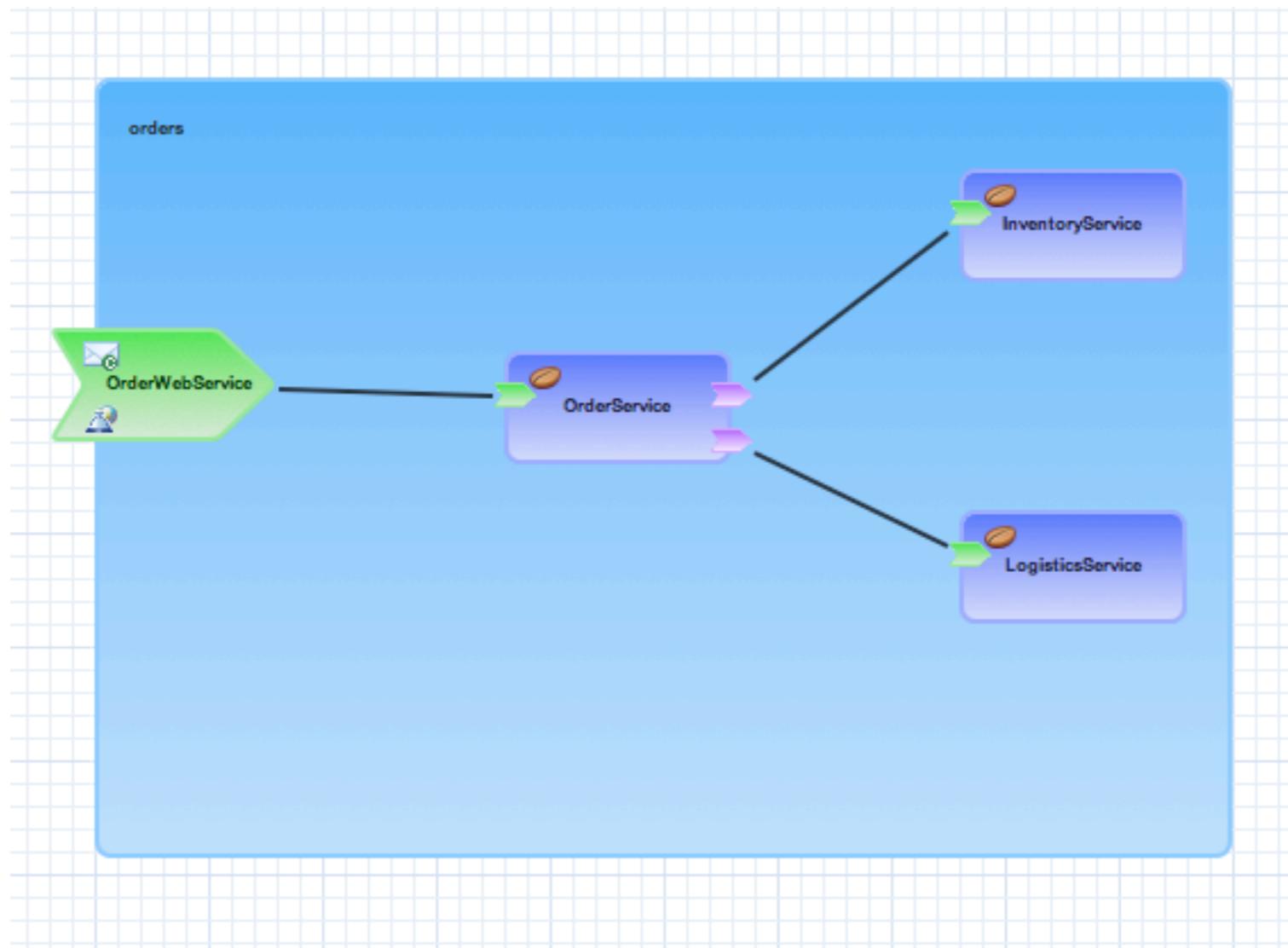
Runtime Governance

- Real Time Analysis of Business Transactions
 - Customisable Analysis and Presentation
- Policy Enforcement
- Historical Analysis
 - Business Reporting

Event Processors

- Synchronous or Asynchronous
- Monitor SLAs
- Trace Execution
- Enforce Business Policies
 - Block Transaction
 - Extract Correlation Information
 - Implement using Java, Drools and MVEL

Demo Application



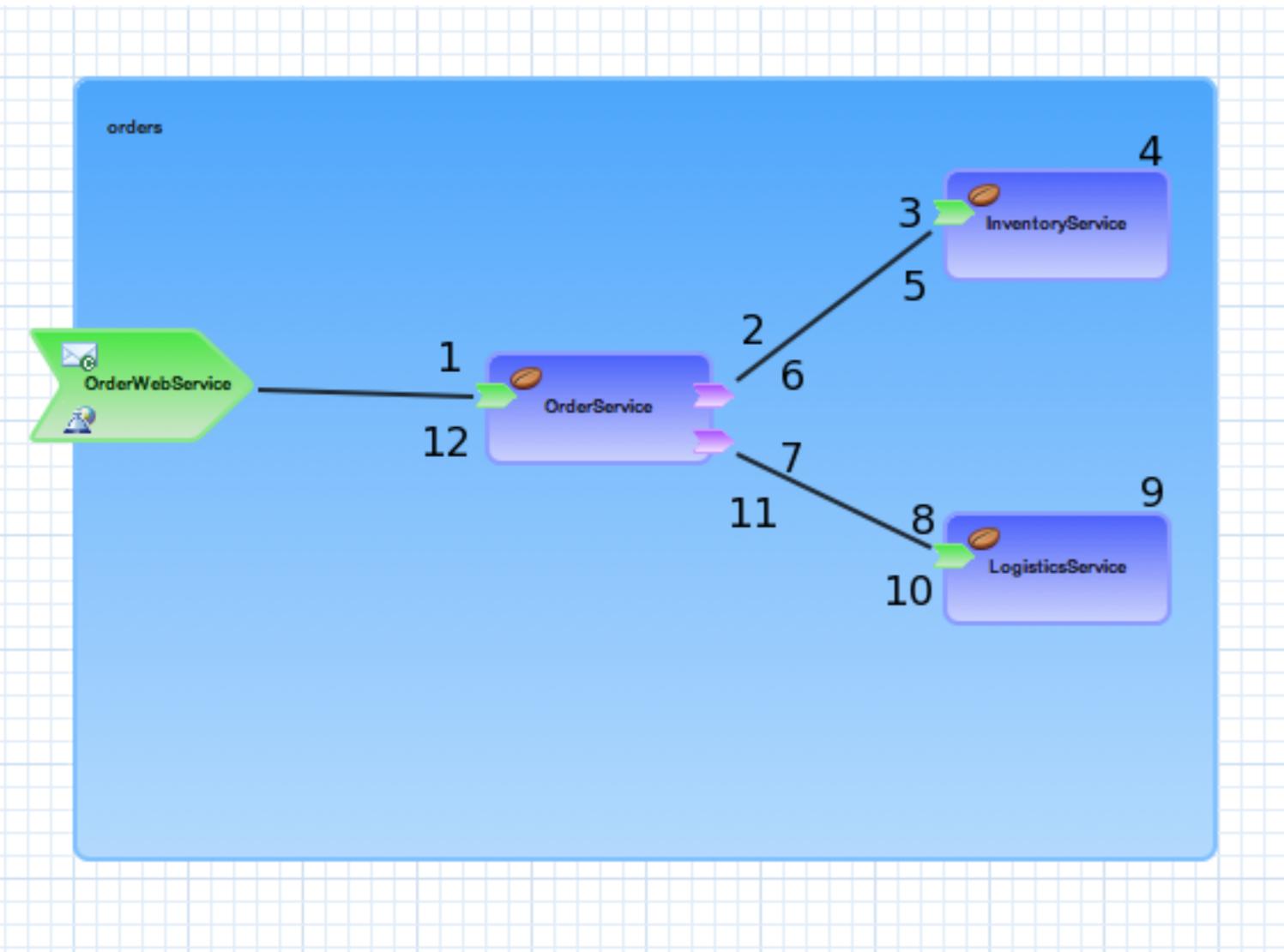
Services

- OrderService
 - submitOrder
 - makePayment
- InventoryService
 - lookupItem
- LogisticsService
 - deliver

Policies

- Credit Assessment
 - Suspend account if exposure > \$150
- Monitor SLA
 - Critical - 400 ms,
 - High - 320 ms
 - Medium - 260 ms
 - Low - 200ms
- Limit Invocations
 - One every two seconds

Events



- RequestReceived
 - 1, 3, 8
- RequestSent
 - 2, 7
- ResponseSent
 - 5, 10, 12
- ResponseReceived
 - 6, 11
- LogMessage
 - 4, 9

**RED HAT
SUMMIT**

Demo



More SOA 6 @ Summit

- Sessions
 - Loosely Coupled, Reusable & Interoperable : Integration Overview & Roadmap*
- Labs
 - Getting Started with Red Hat JBoss SOA Platform 6*
 - Migrating Applications from Red Hat JBoss SOA Platform 5 to 6
 - Choose Your Own Adventure : Getting Started with Red Hat JBoss Middleware

* catch the recordings or rent a time machine

**RED HAT
SUMMIT**

Questions?

