Creating an Oasis of Enterprise Services with Apache Camel

Keith Babo

Topics

- Camel and JBoss
- Enterprise Services
- Use Cases
- Talk it Out

Assumptions

- Camel Knowledge
  - Basic .... advanced
- JBoss Knowledge
  - J-what? .... advanced
- You're ready to rock!

Howdy

- Core Developer at JBoss
- Project Lead for SwitchYard
- Last gig was at Sun
  - ESB/EAI/B2B product development
  - JBI

Background

- Camel and JBoss?
- Drools
- JBoss ESB
- SwitchYard

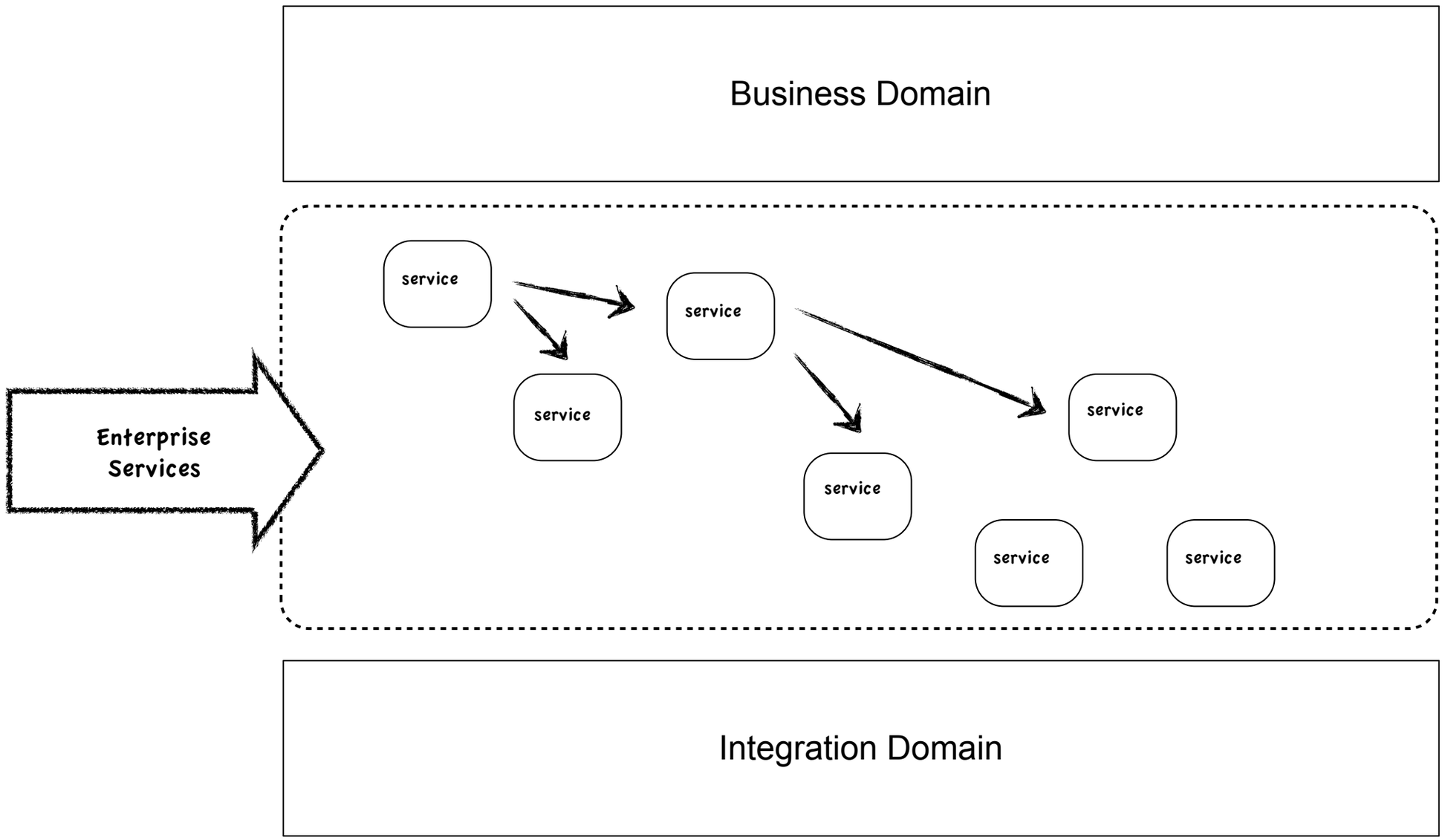
Setting The Stage

- Experience
- Opportunities
- Challenges
- Contribution / Collaboration

Enterprise Services

- Business-y
- Composed and Composable
- Loosely Coupled
- Scaleable, manageable, monitorable, versionable, interoperable, unbreakable, unstoppable ... oh, just shoot me now

This Is What I'm Talking About



Use Cases

- Business Rules
- Complex Event Processing
- Service-Oriented Applications
- Business Process Management

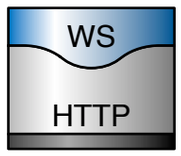
Business Rules

Rules are Everywhere

"When remainder < .01 move remainder to Swiss bank account"

Enteprise Application

"Orders > $1M get priority processing"



Web Service

Web App

"Users in role reviewer can see submissions"

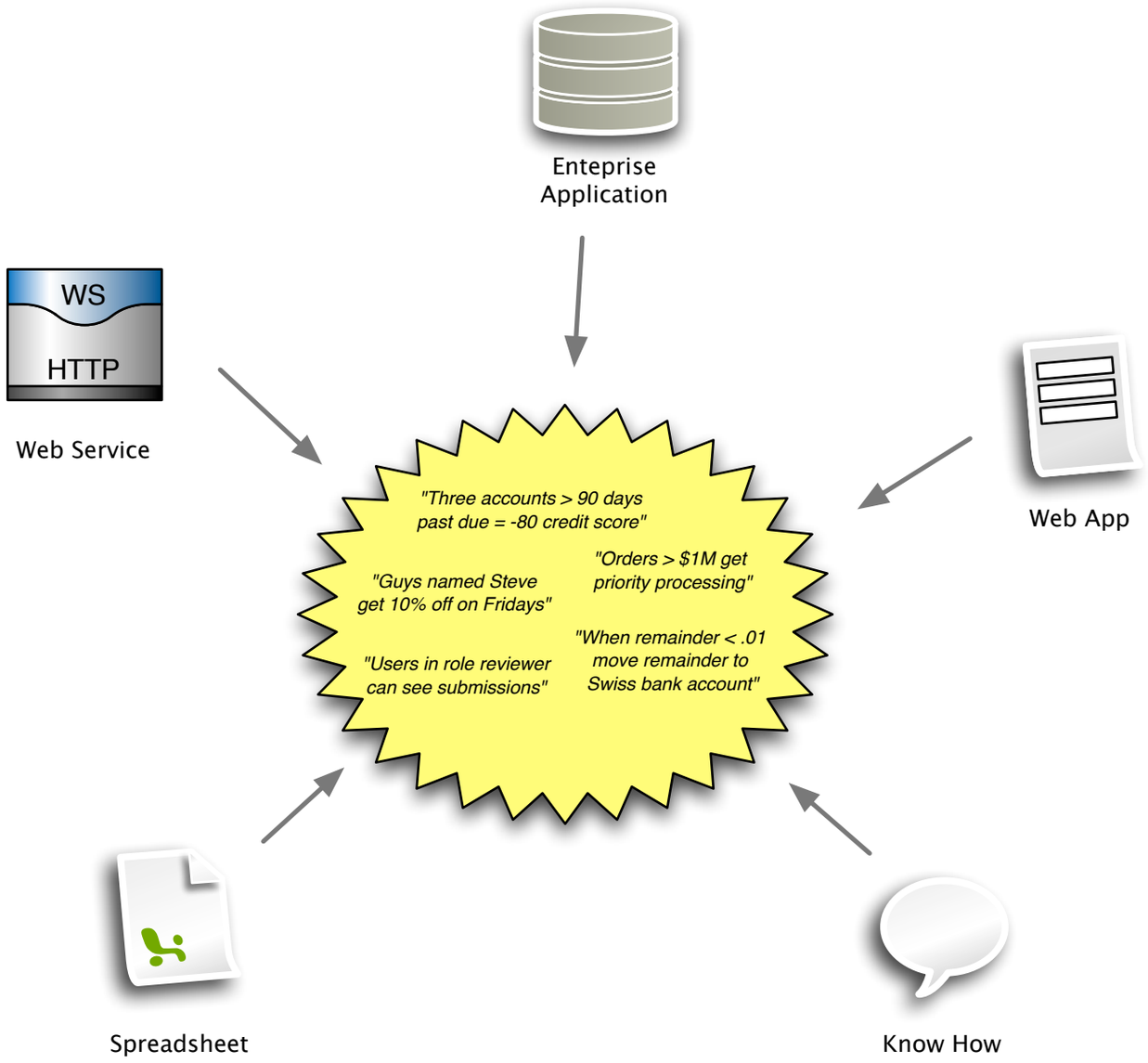
Spreadsheet

"Three accounts > 90 days past due = -80 credit score"

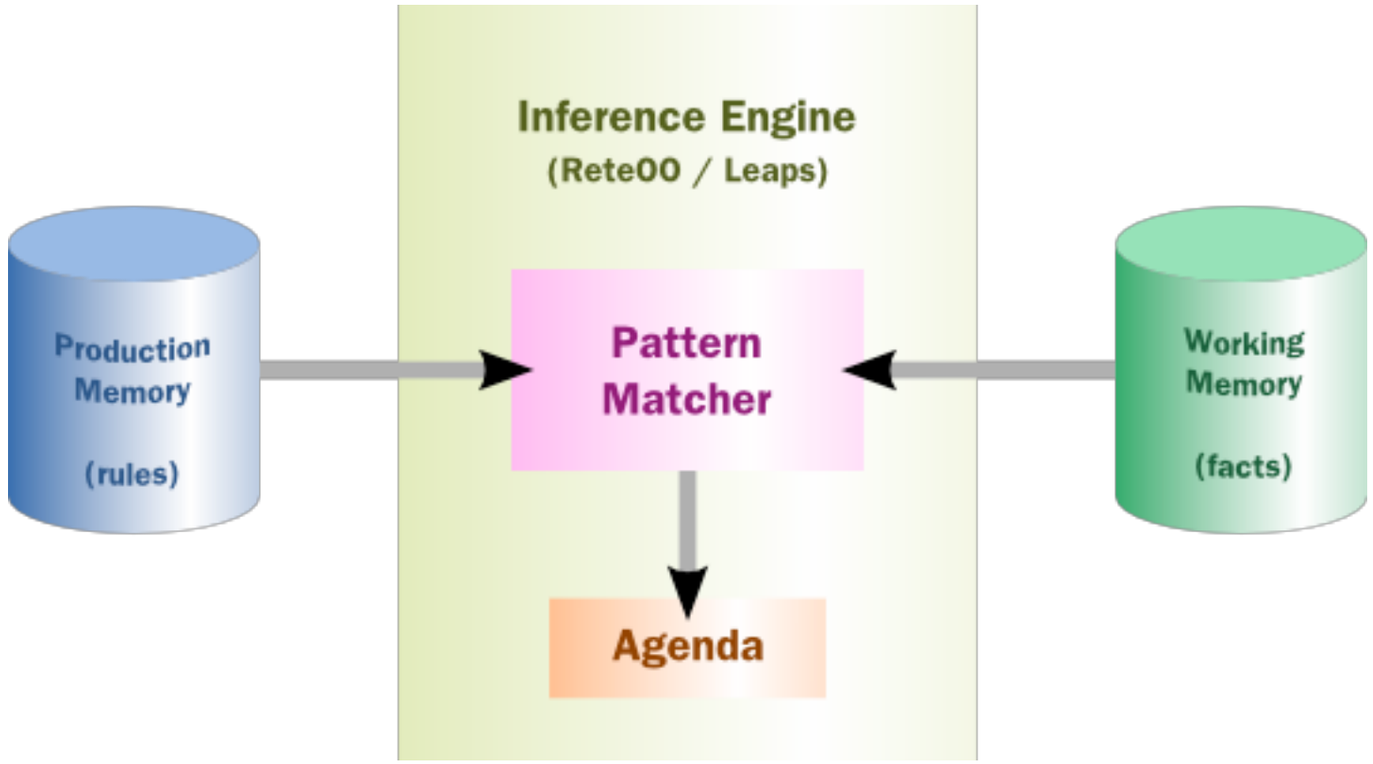
Know How

"Guys named Steve get 10% off on Fridays"

All Your RuleBase Are Belong In One Place



Drools Expert



Facts

```
public class Applicant {

    private String name;
    private int age;
    private boolean valid;

    // getter and setter methods here
}
```

or ...

```
declare Applicant
    name  : String
    age   : int
    valid : boolean
end
```

Rules

```
package com.company.license

rule "Is of valid age"
when
    $a : Applicant( age < 18 )
then
    $a.setValid( false );
end
```

Runtime

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();

kbuilder.add( ResourceFactory.newClassPathResource(
        "licenseApplication.drl", getClass() ), ResourceType.DRL );

if ( kbuilder.hasErrors() ) {
    System.err.println( builder.getErrors().toString() );
}

kbase.addKnowledgePackages( kbuilder.getKnowledgePackages() );
StatelessKnowledgeSession ksession = kbase.newStatelessKnowledgeSession();

Applicant applicant = new Applicant( "Mr John Smith", 16 );
assertTrue( applicant.isValid() );
ksession.execute( applicant );
assertFalse( applicant.isValid() );
```

Camel As A Runtime

```
<beans>
  <drools:kbase id="kbase1" node="node1">
      <drools:resources>
        <drools:resource type="DRL" source="classpath:org/drools/camel/component/licenseApplication.drl"/>
    </drools:resources>
  </drools:kbase>

  <drools:ksession id="ksession1" type="stateless" name="ksession1" kbase="kbase1" node="node1"/>

  <cxf:rsServer id="rsServer" address="http://localhost:9002/rest"
                serviceClass="org.drools.jax.rs.CommandExecutorImpl">
     <cxf:providers>
        <bean class="org.drools.jax.rs.CommandMessageBodyReader"/>
     </cxf:providers>
  </cxf:rsServer>

  <bean id="droolsPolicy" class="org.drools.camel.component.DroolsPolicy" />

  <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <route>
       <from uri="cxfrs://bean://rsServer"/>
       <policy ref="droolsPolicy">
         <unmarshal ref="xstream" />
         <to uri="drools:node1/ksession1" />
         <marshal ref="xstream" />
       </policy>
    </route>
  </camelContext>
</beans>
```

Initialize Rules

```
<drools:kbase id="kbase1" node="node1">
  <drools:resources>
     <drools:resource
        type="DRL"
        source="classpath:org/drools/camel/component/licenseApplication.drl"/>
  </drools:resources>
</drools:kbase>

<drools:ksession id="ksession1"
                 type="stateless"
                 name="ksession1"
                 kbase="kbase1"
                 node="node1"/>
```

Define Endpoint

```
<cxf:rsServer id="rsServer"
              address="http://localhost:9002/applications"
              serviceClass="org.drools.jax.rs.CommandExecutorImpl">
   <cxf:providers>
      <bean class="org.drools.jax.rs.CommandMessageBodyReader"/>
   </cxf:providers>
</cxf:rsServer>
```

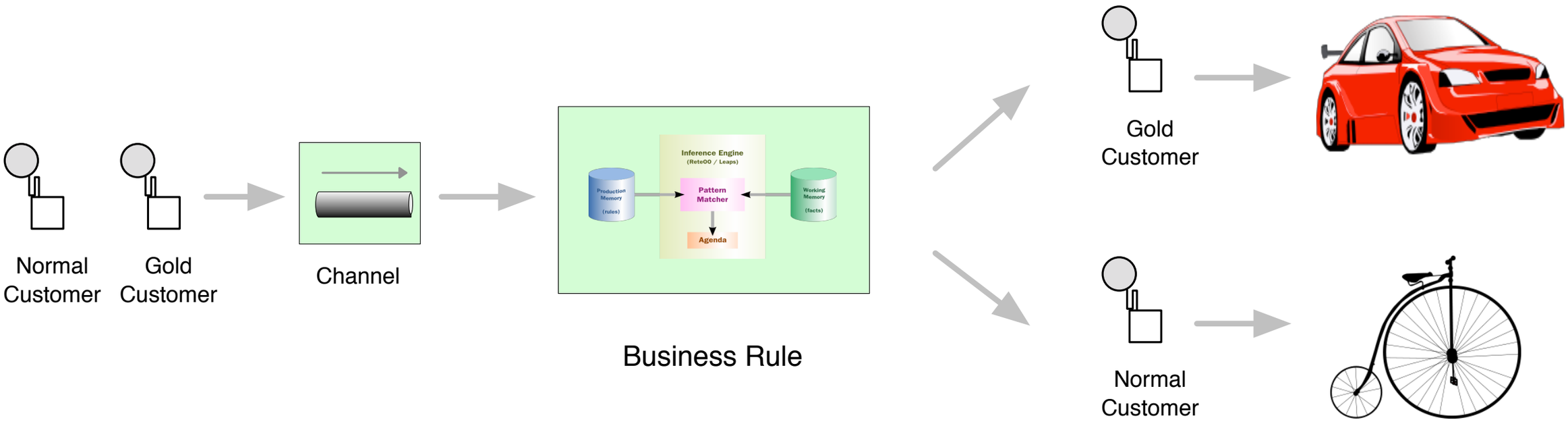
A Little Special Sauce

```
<bean id="droolsPolicy"
      class="org.drools.camel.component.DroolsPolicy" />
```

Route to the Rules

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <route>
       <from uri="cxfrs://bean://rsServer"/>
       <policy ref="droolsPolicy">
         <unmarshal ref="xstream" />
         <to uri="drools:node1/ksession1" />
         <marshal ref="xstream" />
       </policy>
  </route>
</camelContext>
```

Drools Component



Rules In Camel

```
import org.apache.camel.Message;
rule "LongLiveTheBurgeoisie"
  when
    $message : Message(headers["STATUS"] == "GOLD");
  then
    $message.setHeader("routingSlip", "bean:RsvpService?method=fancyCar");
end

rule "DownWithTheProletariat"
  when
    $message : Message(headers["STATUS"] == "NORMAL");
  then
    $message.setHeader("routingSlip", "bean:RsvpService?method=twoWheels");
end
```

```
<route>
 <from uri="activemq:queue:router"/>
 <to uri="drools:brokerNode/cbrKSession?action=insertMessage"/>
 <routingSlip uriDelimiter="#">
     <header>routingSlip</header>
 </routingSlip>
</route>
```

http://blog.athico.com/2011/01/drools-content-based-routing-with-camel.html

Complex Event Processing

#define CEP

"Complex Event Processing, or CEP, is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the event cloud."

Drools Fusion

- Event Detection
  - Select meaningful events from a cloud or stream
- Event Correlation
  - Reasoning based on temporal constraints
- Event Abstraction
  - Create events from a set of atomic events

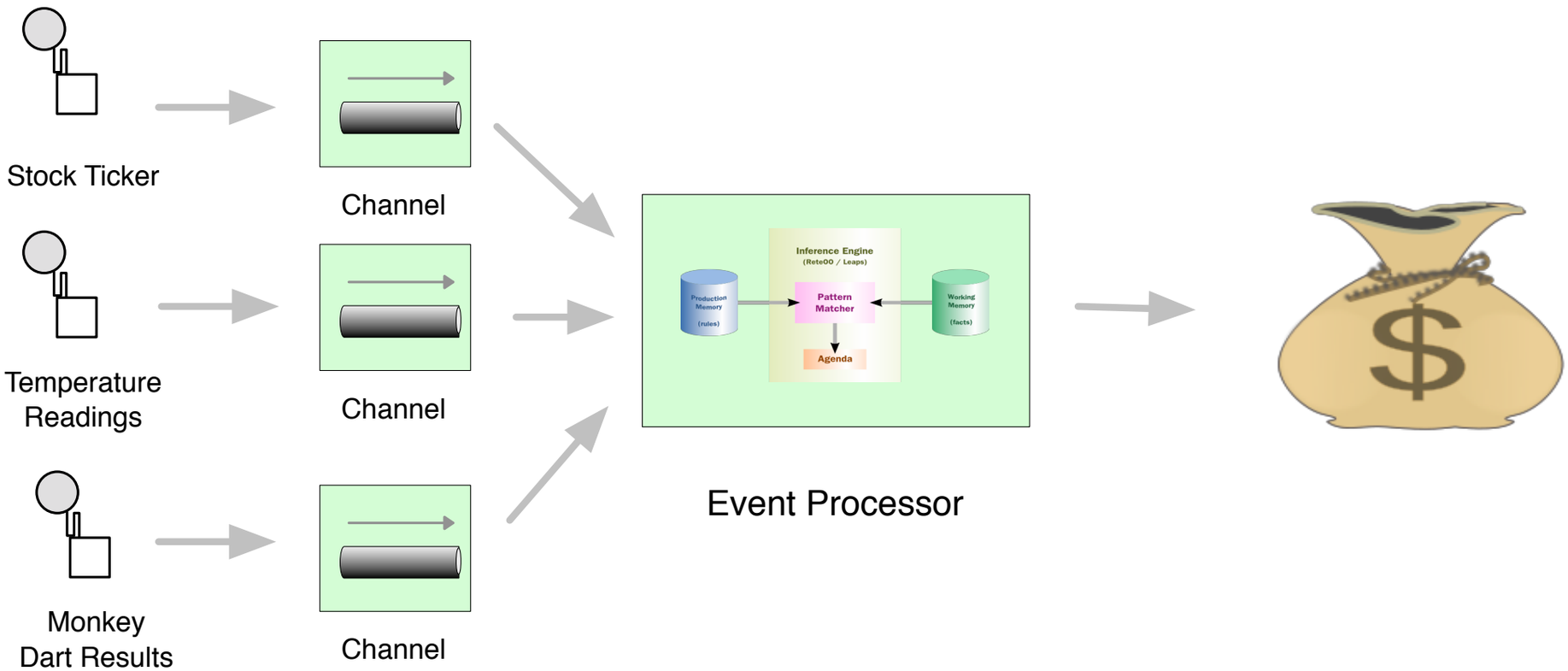
Events Rule!

```
declare StockTick
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    symbol   : String
    price    : double
end
```

```
declare MonkeyDart
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    dart : org.dartboard.Position
end
```

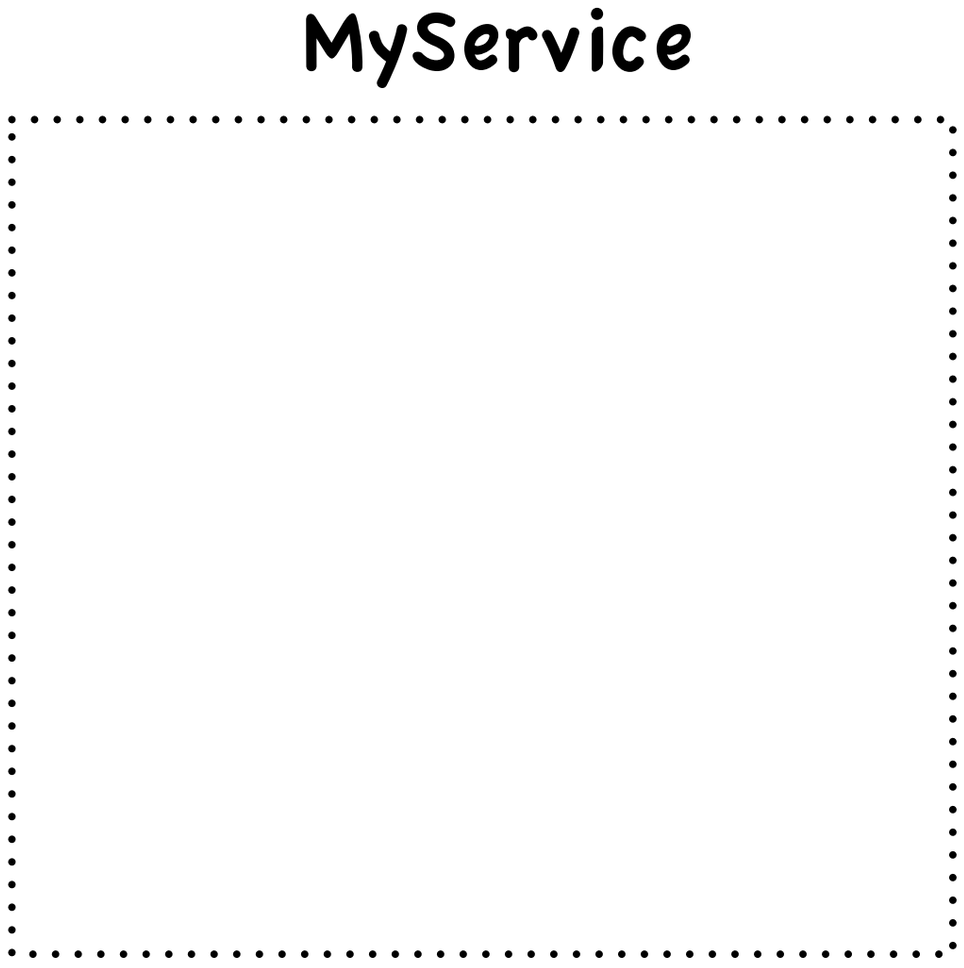
```
rule "Buy Low Sell High"
when
    YearLow( $low : low )
    $st : StockTick( price <= $low )
    $md : MonkeyDart( this after[0,5m] $st )
then
    // buy all the stock you can get!
end
```

CEP In Camel

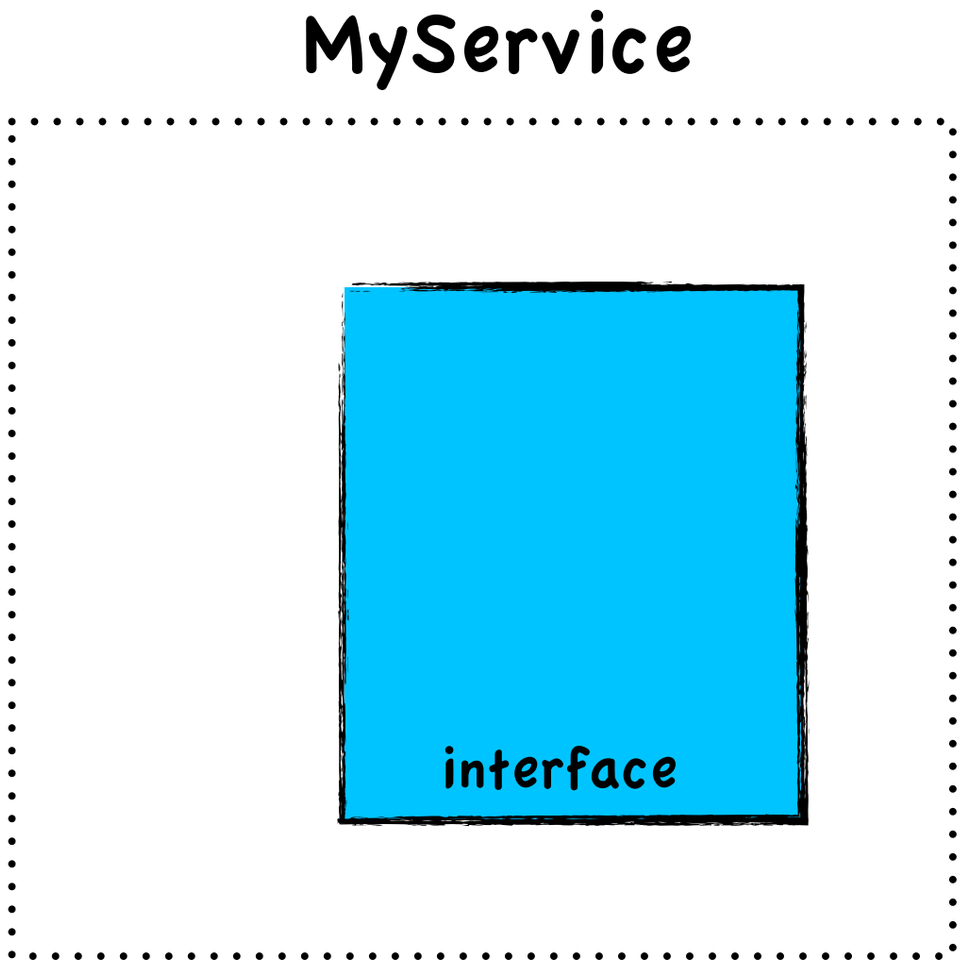

Stock Ticker
Channel
Temperature Readings
Channel
Monkey Dart Results
Channel
Inference Engine
(ReteOO / Leaps)
Production Memory
(rules)
Pattern Matcher
Working Memory
(facts)
Agenda
Event Processor

Service-Oriented Applications

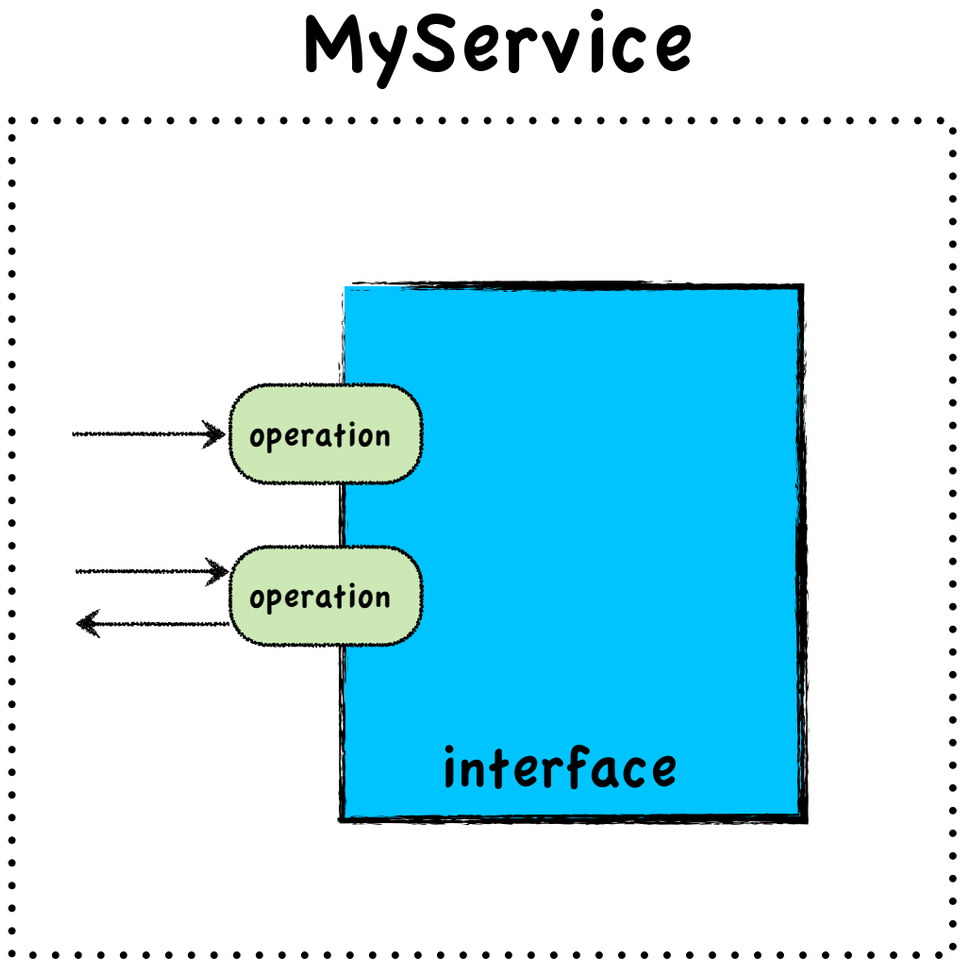
Service-Oriented App



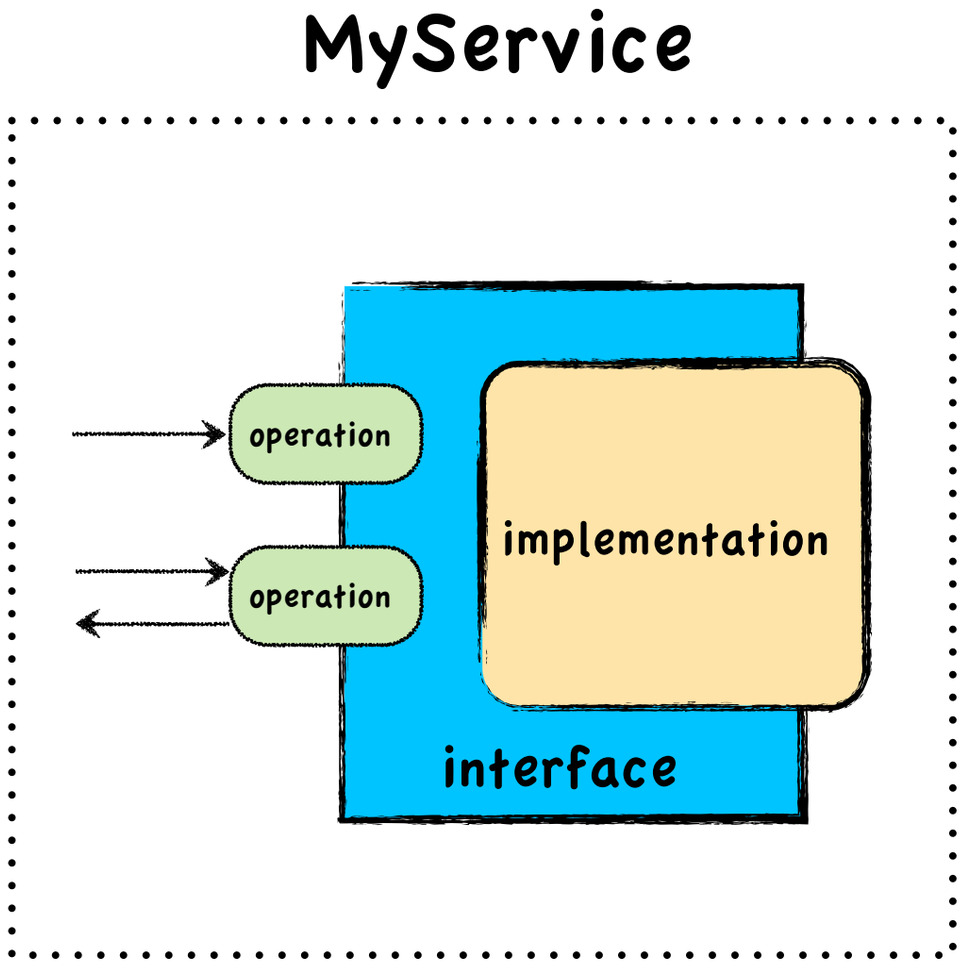
Service-Oriented App



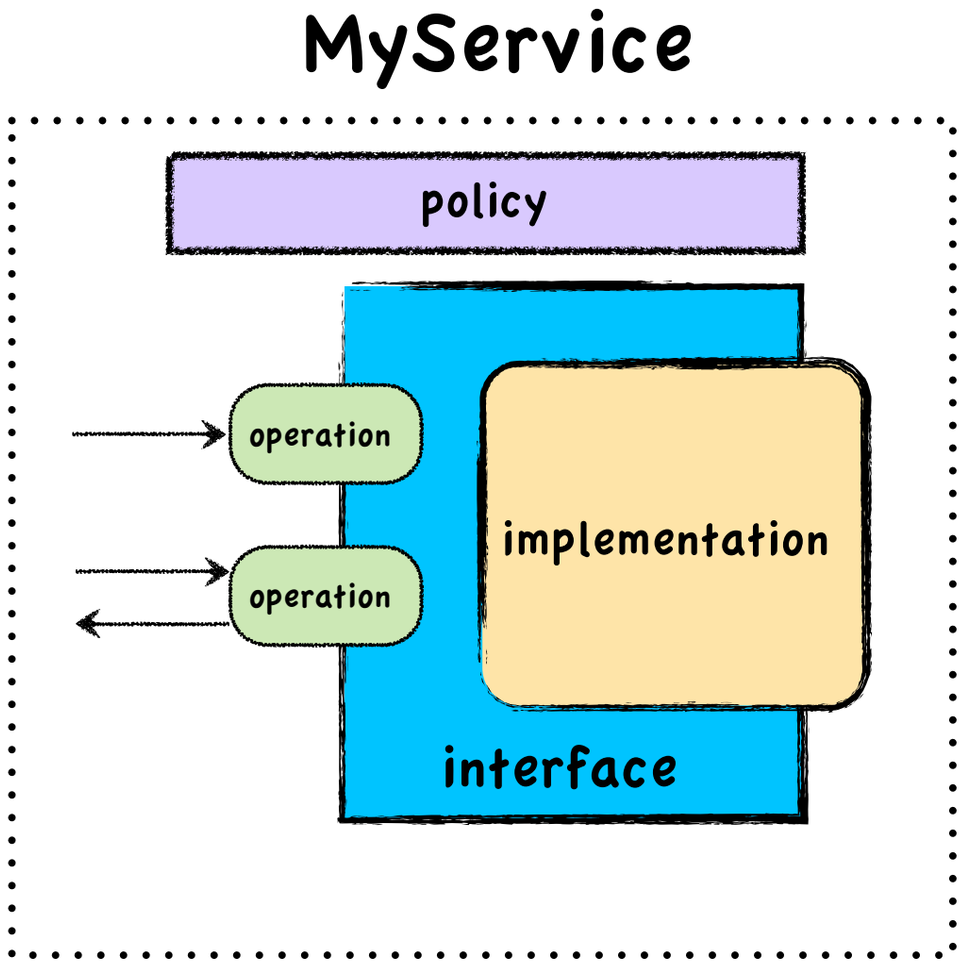
Service-Oriented App



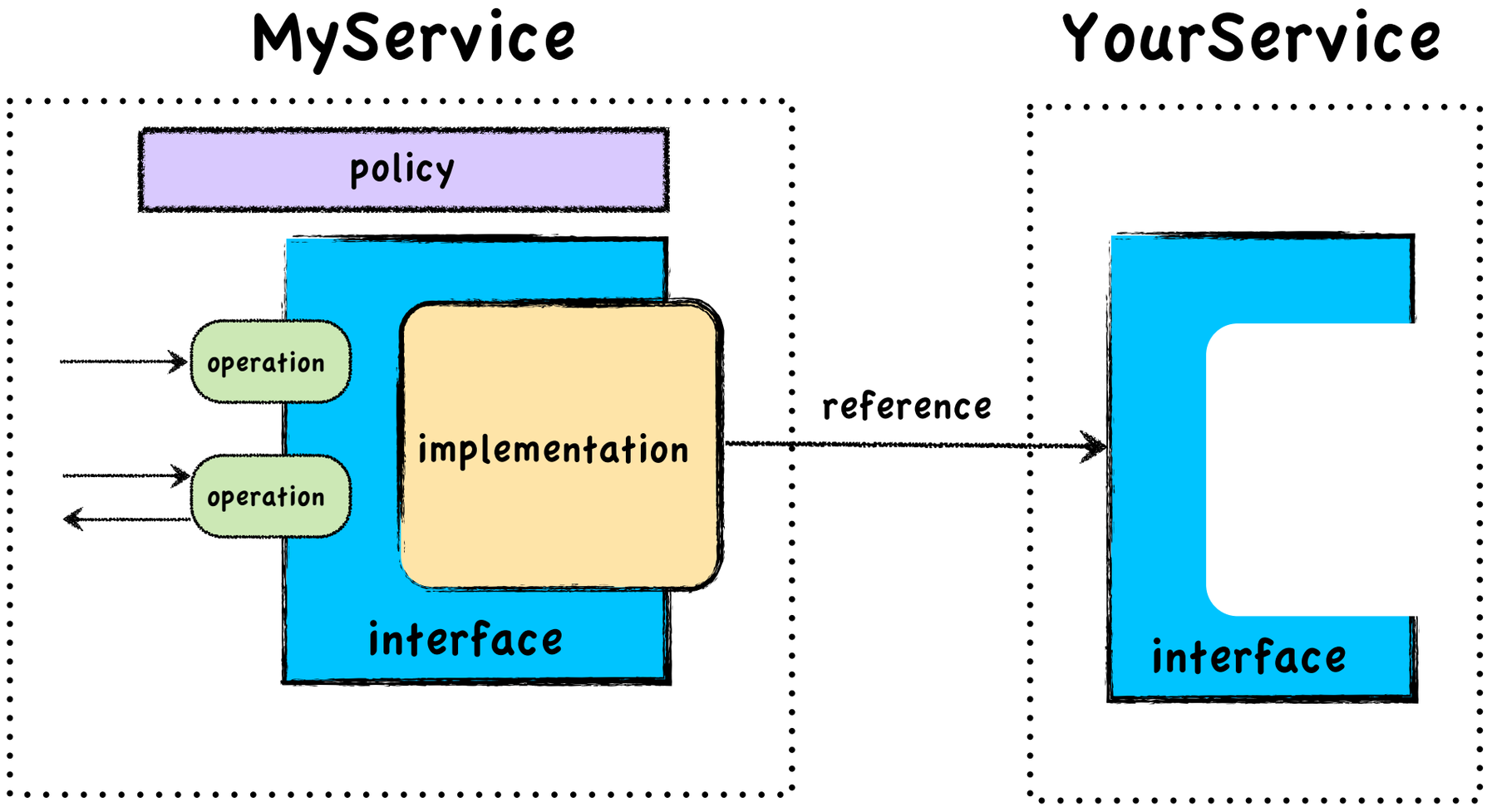
Service-Oriented App



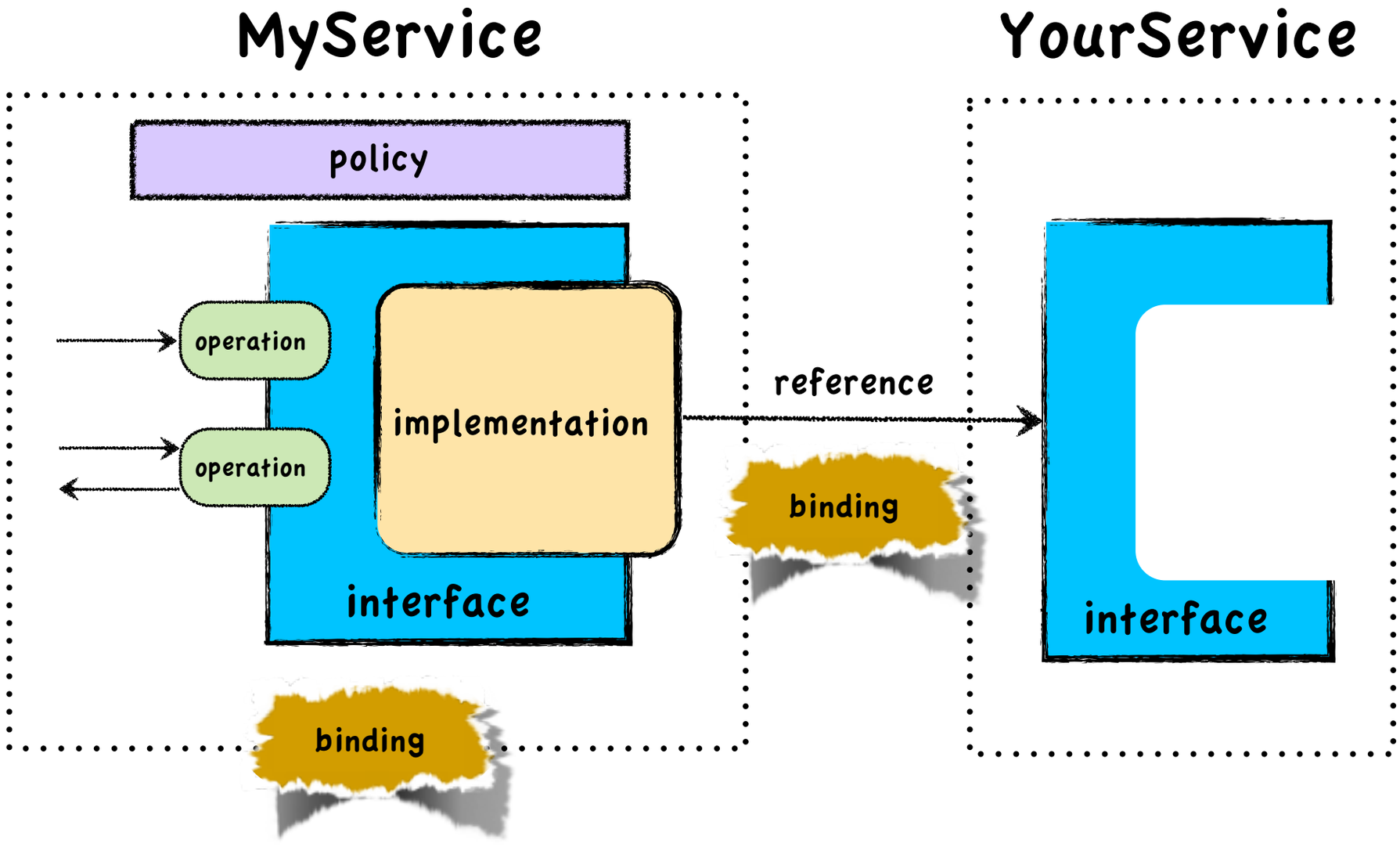
Service-Oriented App



Service-Oriented App



Service-Oriented App

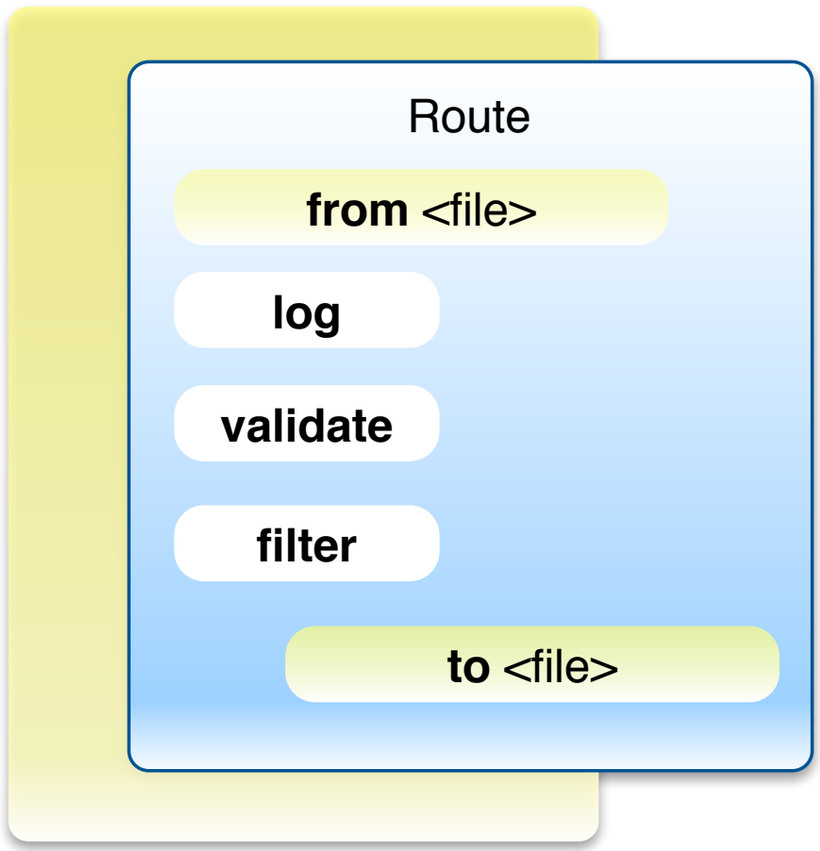


- New ESB project in JBoss Community
- Remembering the 'SO' in SOA
- Camel is an important ingredient
  - Routing
  - Gateways

Example Route

```
<route>
  <from uri="file://orders/in"/>
  <log message="Order Received : ${body}"/>
  <to uri="OrderValidator”/>
  <filter>
    <xpath>/order[@priority='high']</xpath>
    <to uri="file://shipping/in"/>
  </filter>
</route>
```

Example Route



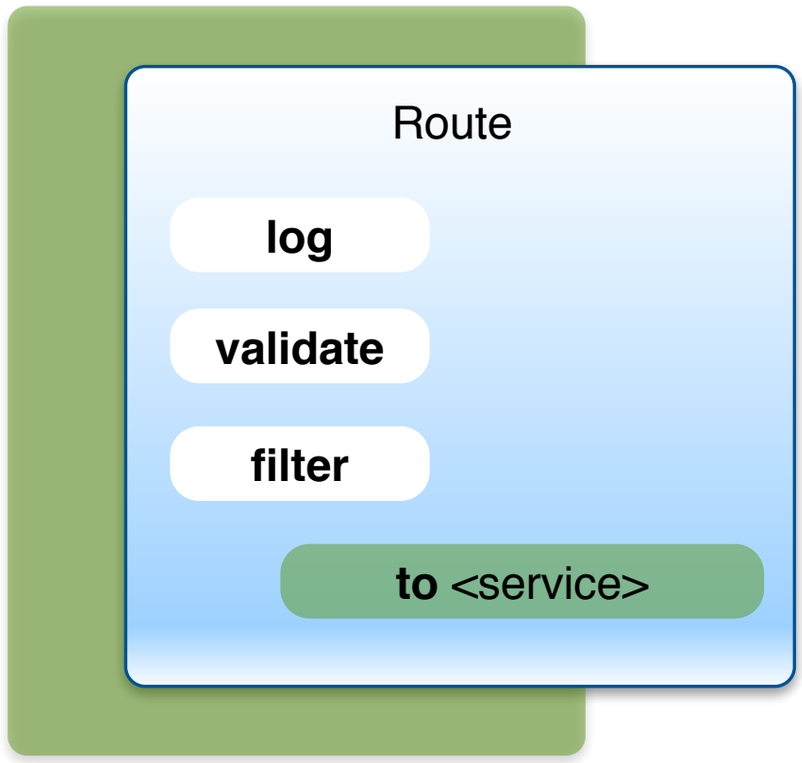
Route As A Service

```
<sca:component name="CamelComponent">

   <sca:service name="OrderService" >
      <sca:interface.java interface="org.example.OrderService"/>
   </sca:service>

   <sca:reference name="ShippingService">
      <sca:interface.java interface="org.example.ShippingService"/>
   </sca:reference>

   <implementation.camel>
      <route>
        <log message="Order Received : ${body}"/>
        <to uri="OrderValidator”/>
        <filter>
          <xpath>/order[@priority='high']</xpath>
          <to uri="switchyard://ShippingService"/>
        </filter>
      </route>
   </implementation.camel>

</sca:component>
```

Same Route ...

```
public class OrderServiceRoute {
    @Route(OrderService.class)
    public void define(ProcessorDefinition<RouteDefinition> route) {
        route.log("Order Received : ${body}")
            .filter().xpath("/order[@priority='high']”)
                .to("switchyard://ShippingService")
    }
}
```

Route As A Service



Transformation

- Ubiquitous challenge in application integration and SOA
- Three flavors
  - Change in data representation
  - Change in data format
  - Change in data itself

Conversion

- Change in representation
- Representation = Java type
- Transformation is simply a type conversion
- No semantic knowledge required

java.lang.String

```
<order>
   <item>XYZ123</item>
   <quantity>5</quantity>
</order>
```

message.getBody(String.class)

=

org.w3c.dom.Node

```
<order>
   <item>XYZ123</item>
   <quantity>5</quantity>
</order>
```

message.getBody(Node.class)

=

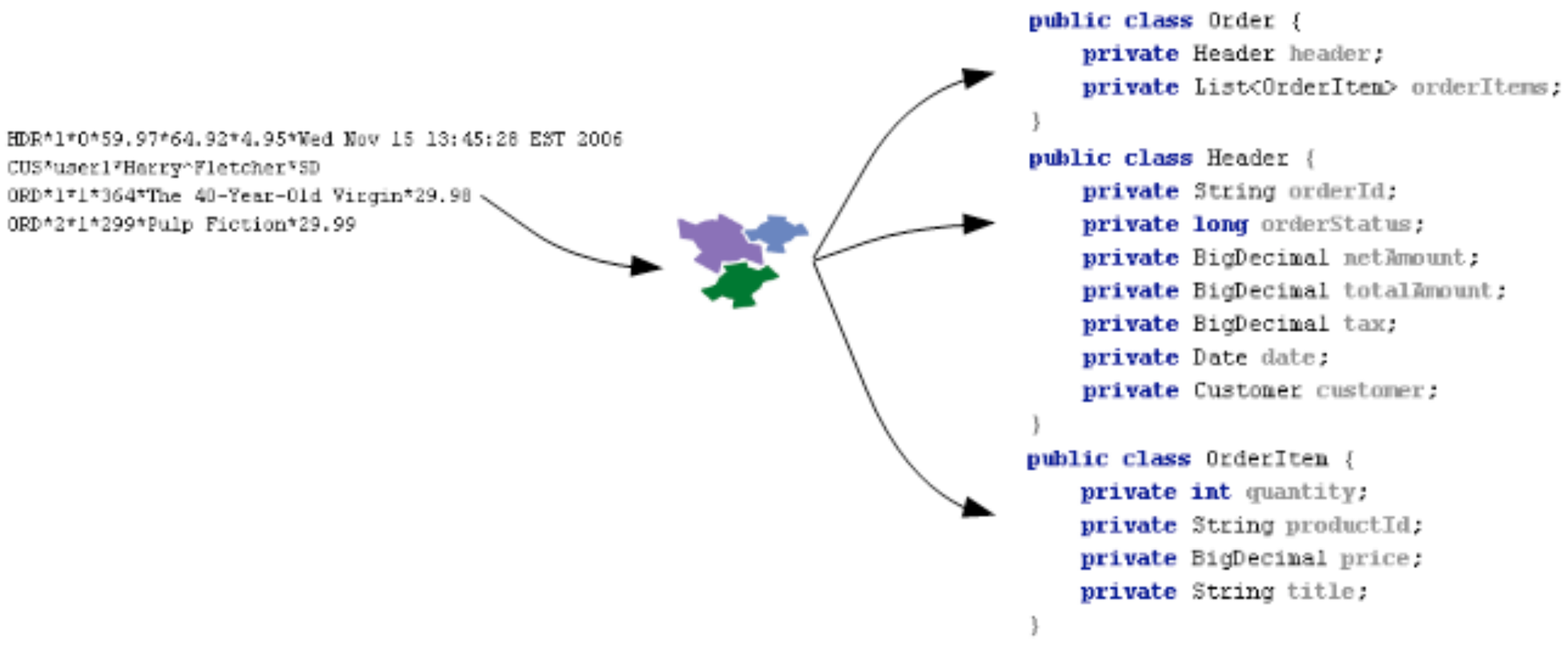
java.io.InputStream

```
<order>
   <item>XYZ123</item>
   <quantity>5</quantity>
</order>
```

message.getBody(InputStream.class)

Translation

- Requires semantic understanding of data types
- Machines cannot do this on their own ...



SwitchYard Transformers

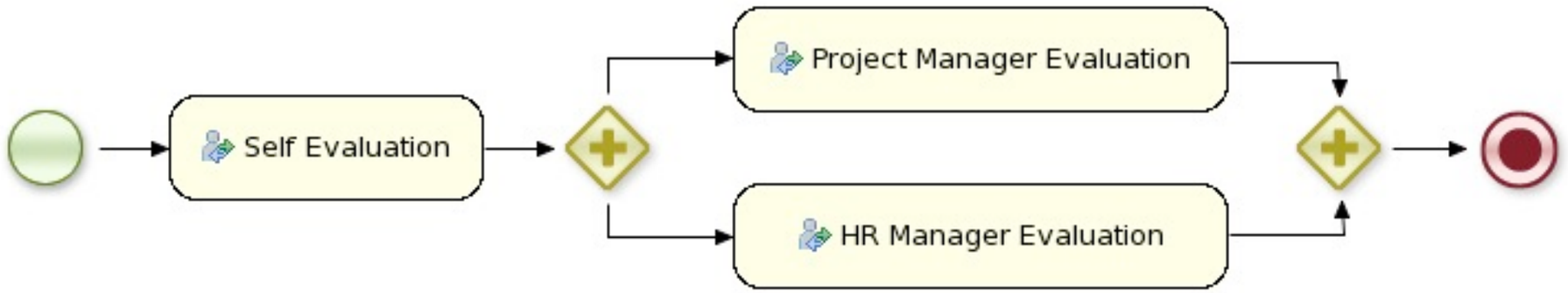
- Transformation is wired into bus
  - Types declared via service contract
  - Transformer resolved dynamically at runtime

```
public class OrderTransform {

    @Transform(to="{urn:examples:orders:1.0}purchaseOrder")
    public Element transform(Order from) {
        // ... doesn't matter
    }
}
```

Business Process Management

#define BPM

Business Process Management



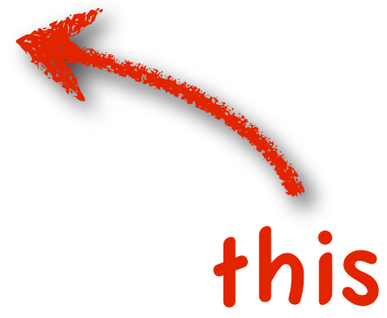
A ***business process*** is a process that describes the order in which a series of steps need to be executed, using a flow chart.

Compositional Models

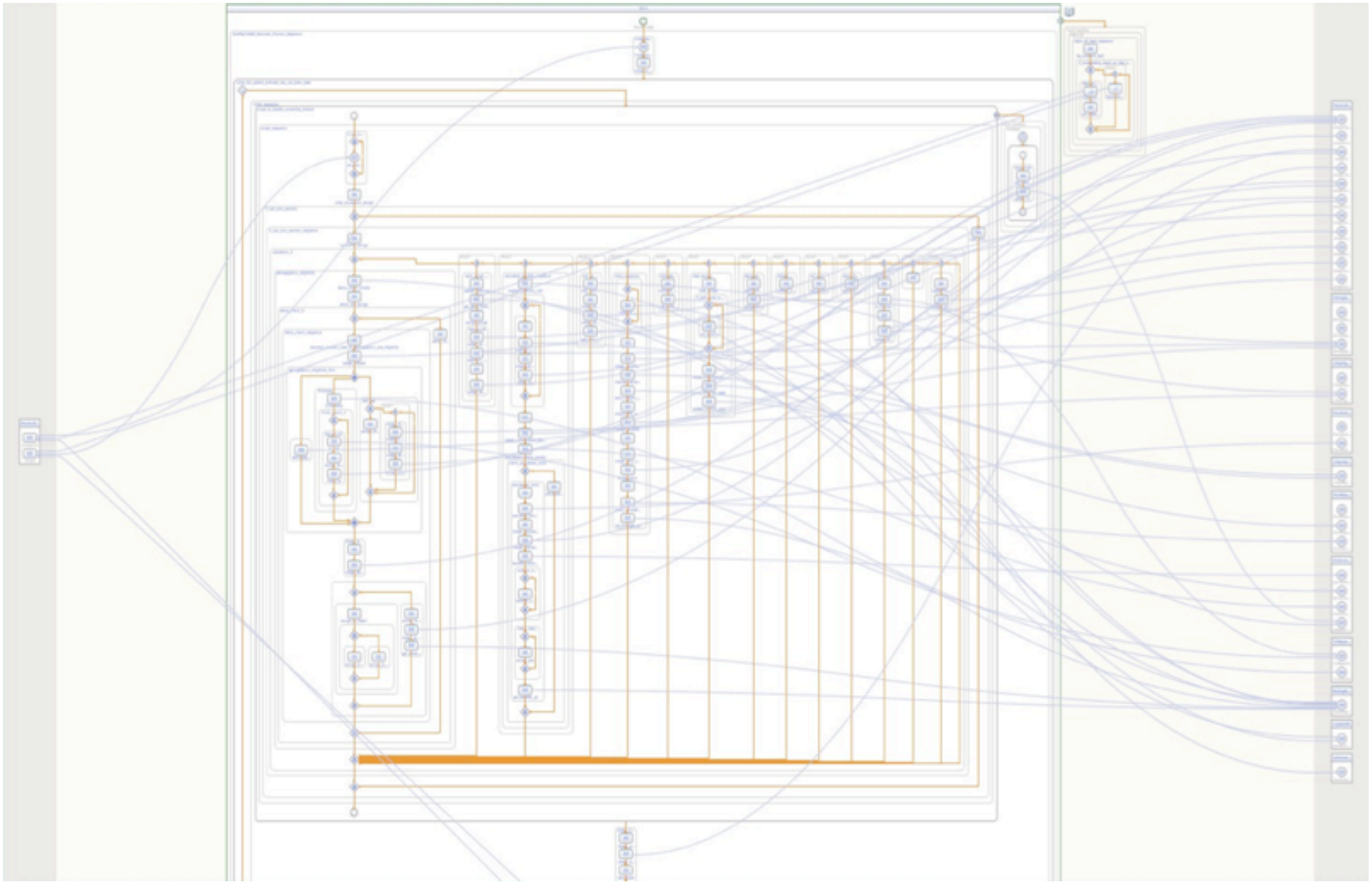
- Direct Reference
- Pipeline Execution
- Orchestration

Compositional Models

- Direct Reference
- Pipeline Execution
- Orchestration

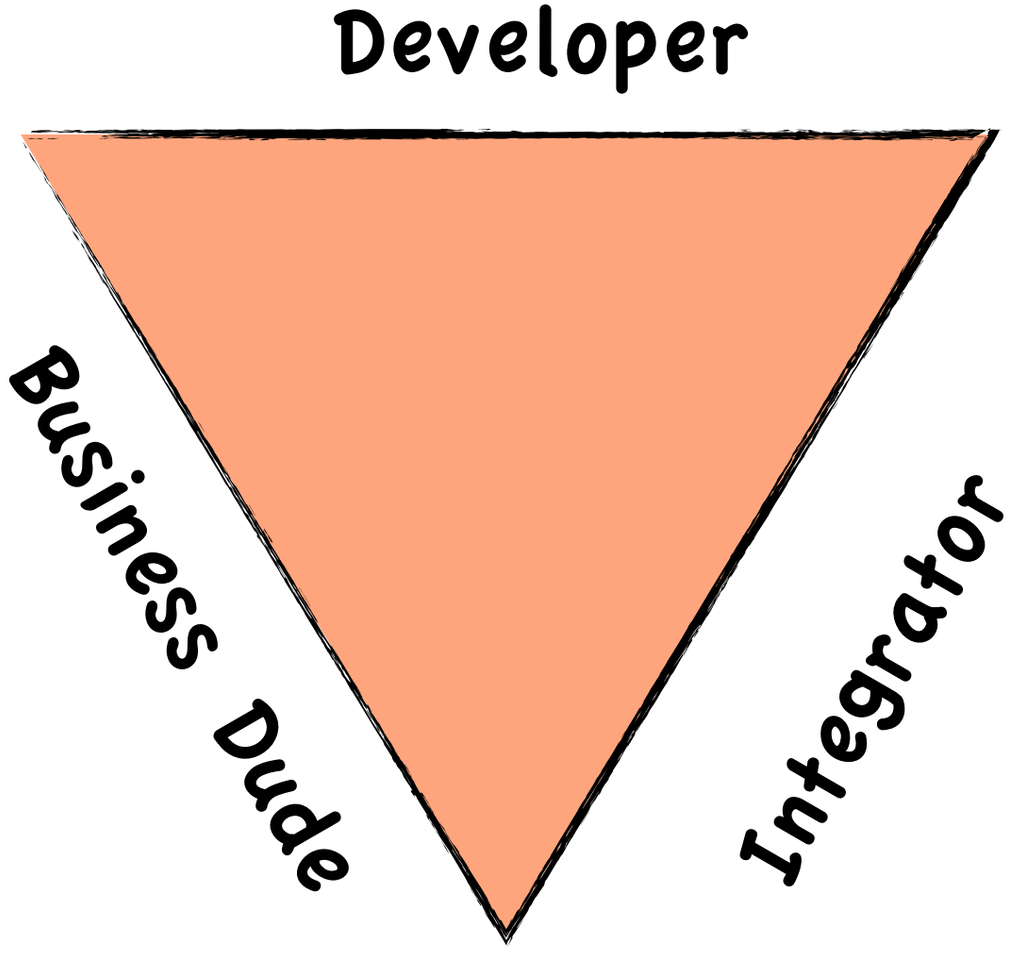


When Good Composition Goes Bad



What Composition and When?

- Who's asking?

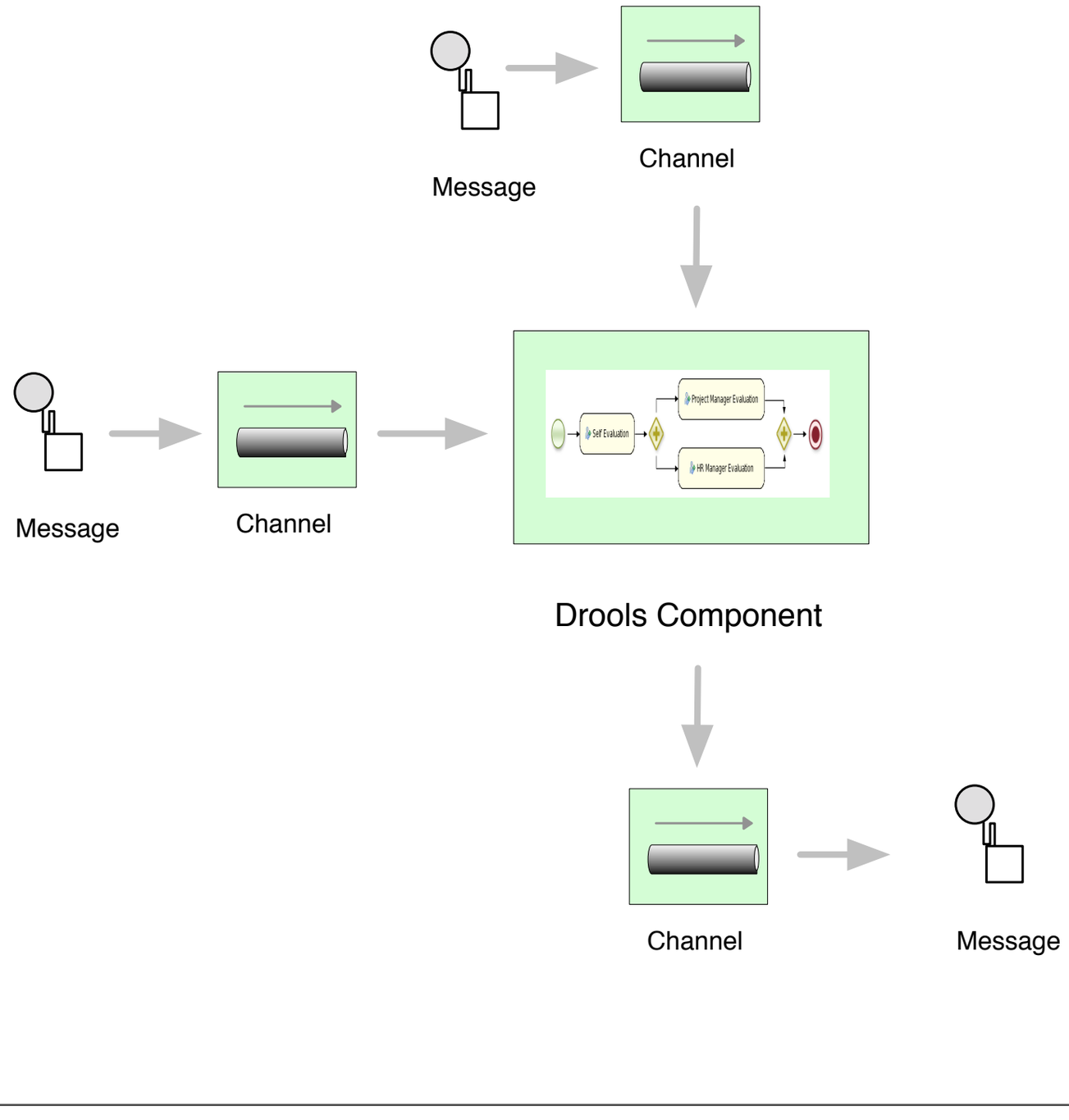


What Composition and When?

- What are you trying to do?
  - Make business analyst's head explode
  - Parallel activities
  - Long-lived transactions
  - Compensation
  - Human workflow
  - Activity monitoring

- Java-based workflow engine
- Native BPMN 2.0 execution
- First class integration with business rules and event processing
- Friendly to developers and business users

jBPM & Camel


Message
Channel
Message
Channel
Self Evaluation
Project Manager Evaluation
HR Manager Evaluation
Drools Component
Channel
Message

Where to Go From Here

- Learn
  - http://www.jboss.org/drools
  - http://www.jboss.org/switchyard
- Chat
  - irc.codehaus.org #camel (babo)
  - chat.freenode.net #switchyard
  - irc.codehaus.org #drools
- Fork
  - https://github.com/droolsjbpm
  - https://github.com/jboss-switchyard

Q & A