14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Mit dem Kamel durchs Nadelöhr

SOA-Integration mit Apache Camel

Frank Pientka

MATERNA GmbH, Dortmund

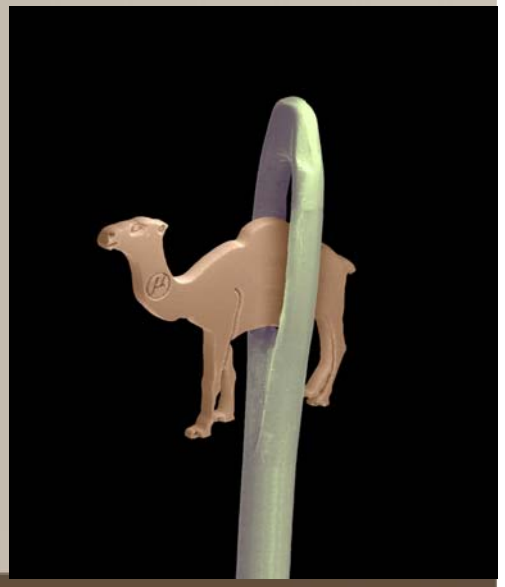
14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Mit dem Kamel durchs Nadelöhr

SOA-Integration mit Apache Camel



Frank Pientka

MATERNA GmbH, Dortmund

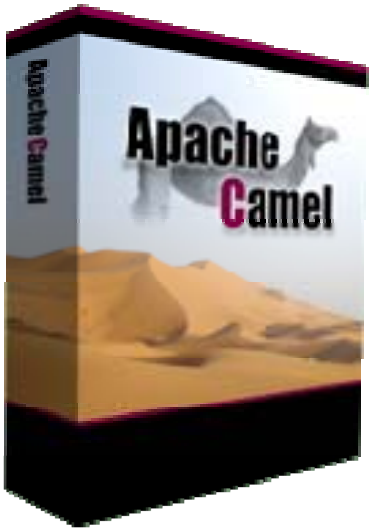
Vorstellung des Referenten: Frank Pientka

Dipl.-Informatiker (TH Karlsruhe), Senior Architekt in Dortmund
Über 10 Jahre Erfahrung mit Java und Middleware
Veröffentlichungen und Vorträge zu:

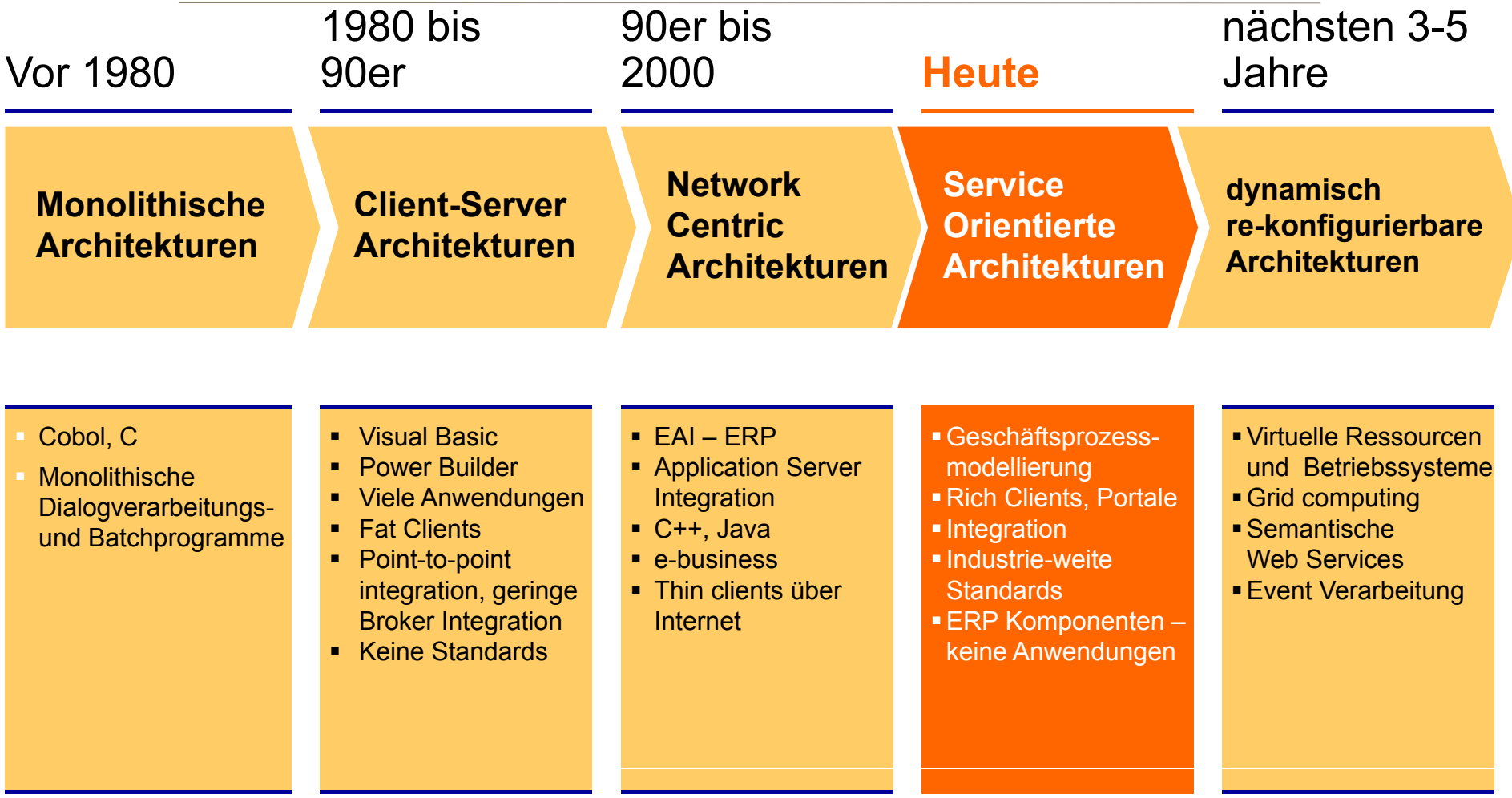
Datenbanken, Applikations- und Portalservern

Mit dem Kamel durchs Nadelöhr

- Was ist SOA? – Herausforderung Integration
- SOA Muster – Lösung des Integrationschaos
- OpenSource SOA – flexibel, schlank
- Apache Camel – Mediator, Router
- Der Ritt durchs Nadelöhr…
- Apache SOA – Produkte, Auswahlkriterien
- Ausblick



Die Evolution von Software Architekturen



SOA: Spaghetti-Orientierte Architektur

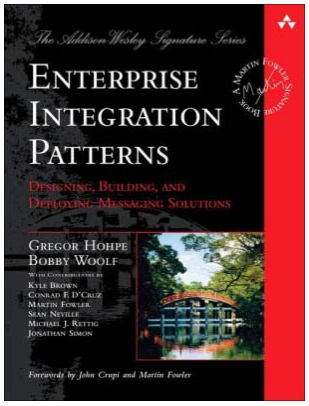
Risikopotential: versteckte Komplexität bei SOA?

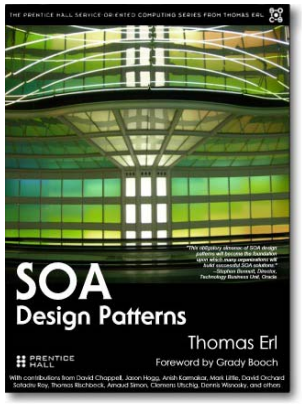
- Schnittstelle
- Protokoll
- Implementierung
- Infrastruktur
- Betrieb
- Überwachung
- Änderung
- Hohe Kosten, Risiken

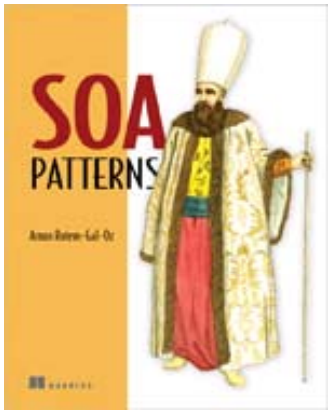
“The first law of distributed objects: Don’t distribute your objects”
Martin Fowler

SOA-Muster

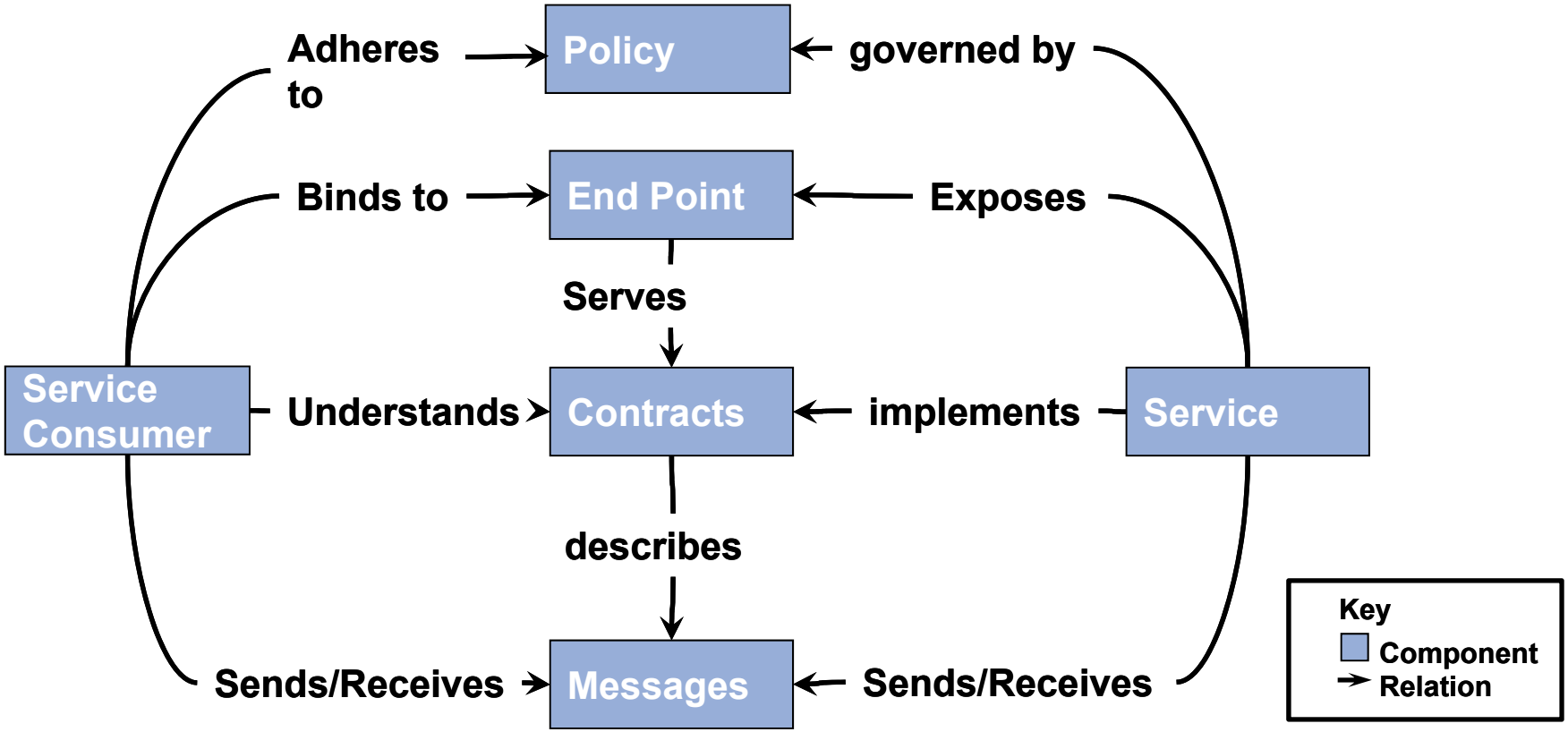
- Schmidt, Stal, Rohnert, Buschmann (19 Muster)
- Hoope, Wolf (65 Muster)
- Arnon Rotem-Gal-Oz (30 Muster + 20 Anti-Muster)
- Erl (85 Muster)





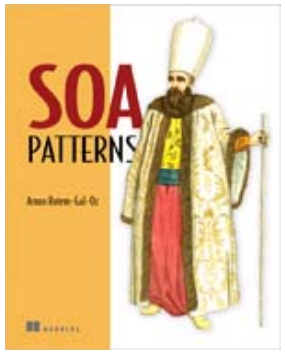


SOA-Komponenten technisch (Rotem-Gal-Oz, 2009)



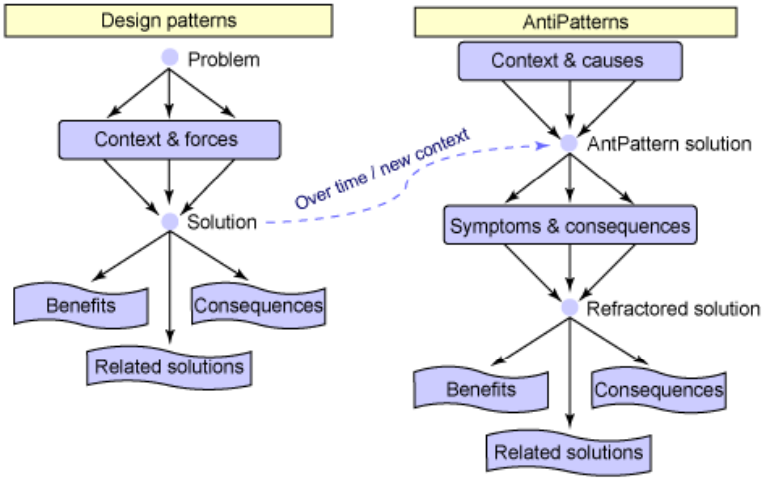
SOA Patterns

- Basic Structural Patterns
- Performance, Scalability & Availability Patterns
- Security & Management Patterns
- Message Exchange Patterns
- Service Interaction Patterns
- Composition Patterns
- UI Interaction Patterns

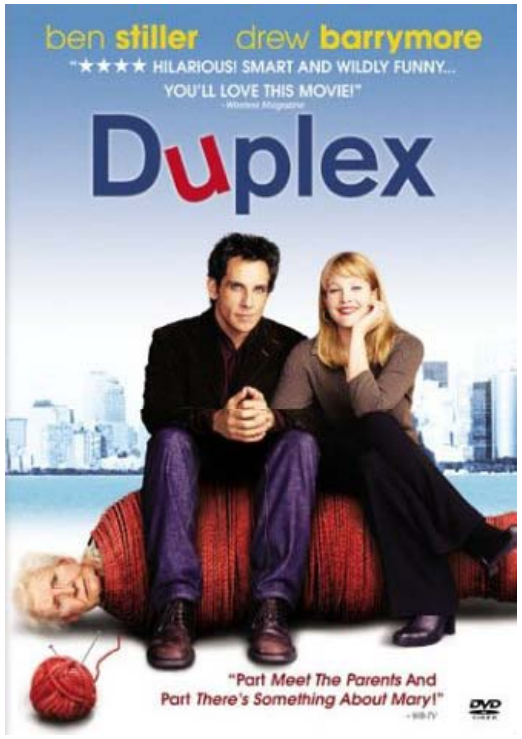


SOA Anti-Patterns

- SOA Adoption Anti-Patterns:
  - Technology Bandwagon, Shiny Nickel, Technology Altar
  - So, What's New?
  - The Big Bang
- Service identification & design Anti-Patterns:
  - Web service = SOA
  - The Silo Approach
  - Misbehaving Registries
  - Architectural Stovepipe
- Service realization Anti-Patterns:
  - Chatty Services
  - Point-to-point Services
  - Component-less Services
  - Performance



Message Exchange Patterns



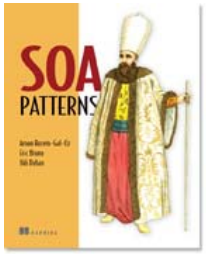
- Incoming
- Outgoing
  - Broadcast
  - Publish/Subscribe

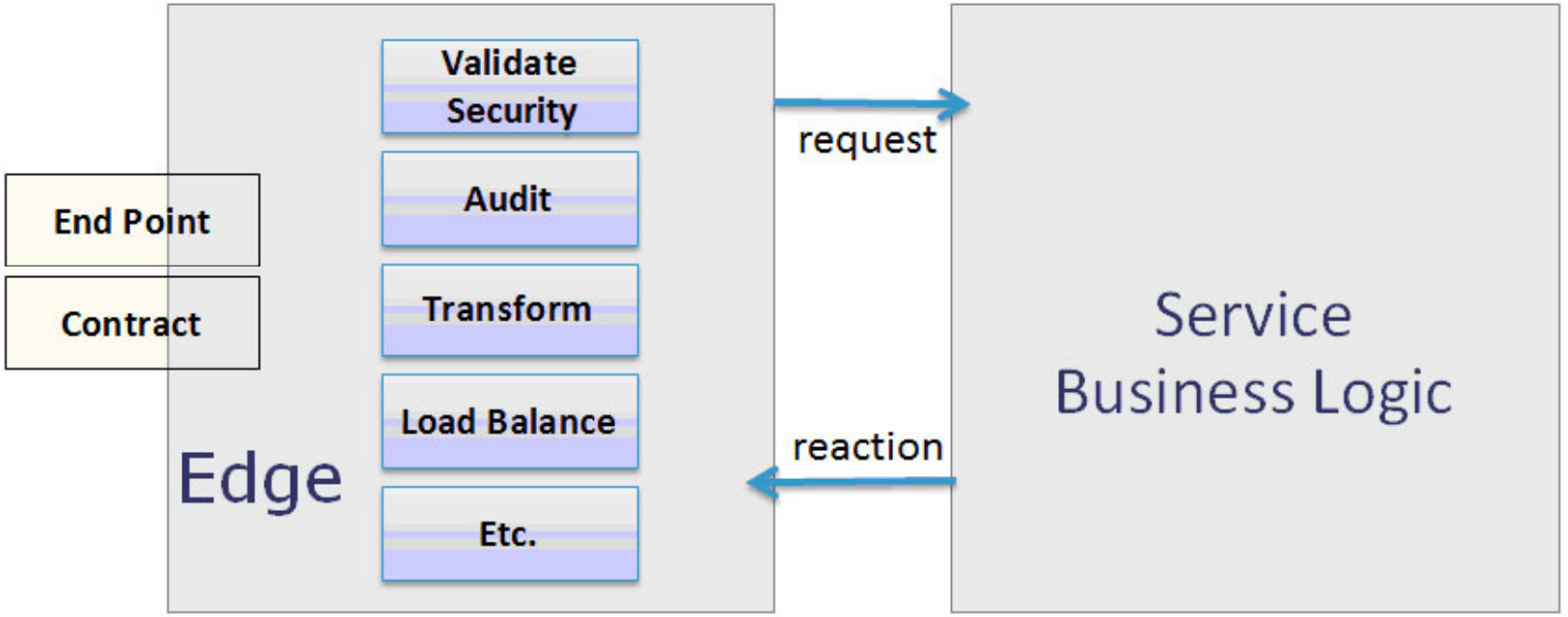
- Two one-ways
- Request/Reply
- Request/Reaction

Was ist ein Service-Endpunkt?

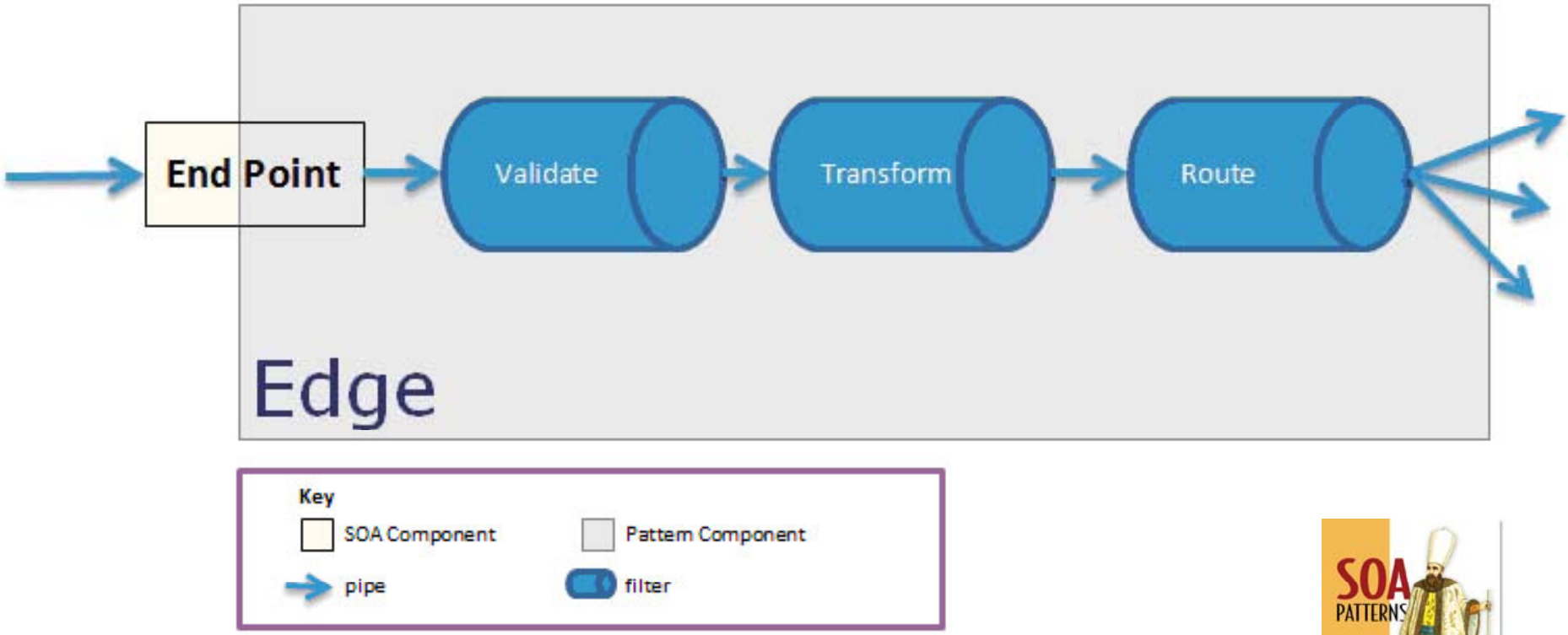
- Wer? Anbieter, Komponente
- Wie? Binding, Protokoll
- Was? Vertrag, Schnittstelle
- Wo? Adresse, Port: URI

SOA Muster: Endpunkt





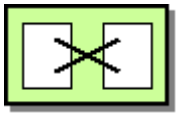
SOA Muster: Endpunkt mit Pipe & Filter



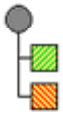
Integrationsmuster

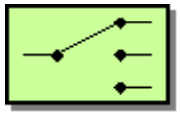
Enterprise Integration Patterns

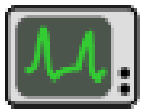
- Channel Patterns
  - Message Channel
  - Point-to-Point Channel
  - Publish Subscribe
- Channel
- Message Patterns
  - Return Address
  - Correlation Identifier
- Routing Patterns
  - Message Router
  - Splitter
  - Aggregator
  - Resequencer
  - Auction



- Transformation Patterns
  - Data Enricher
  - Content Filter
  - Check Baggage
- Endpoint Patterns
  - Polling Consumer
  - Event-Driven Consumer
  - Messaging Mapper
- Management Patterns
  - Message Store
  - Test Message

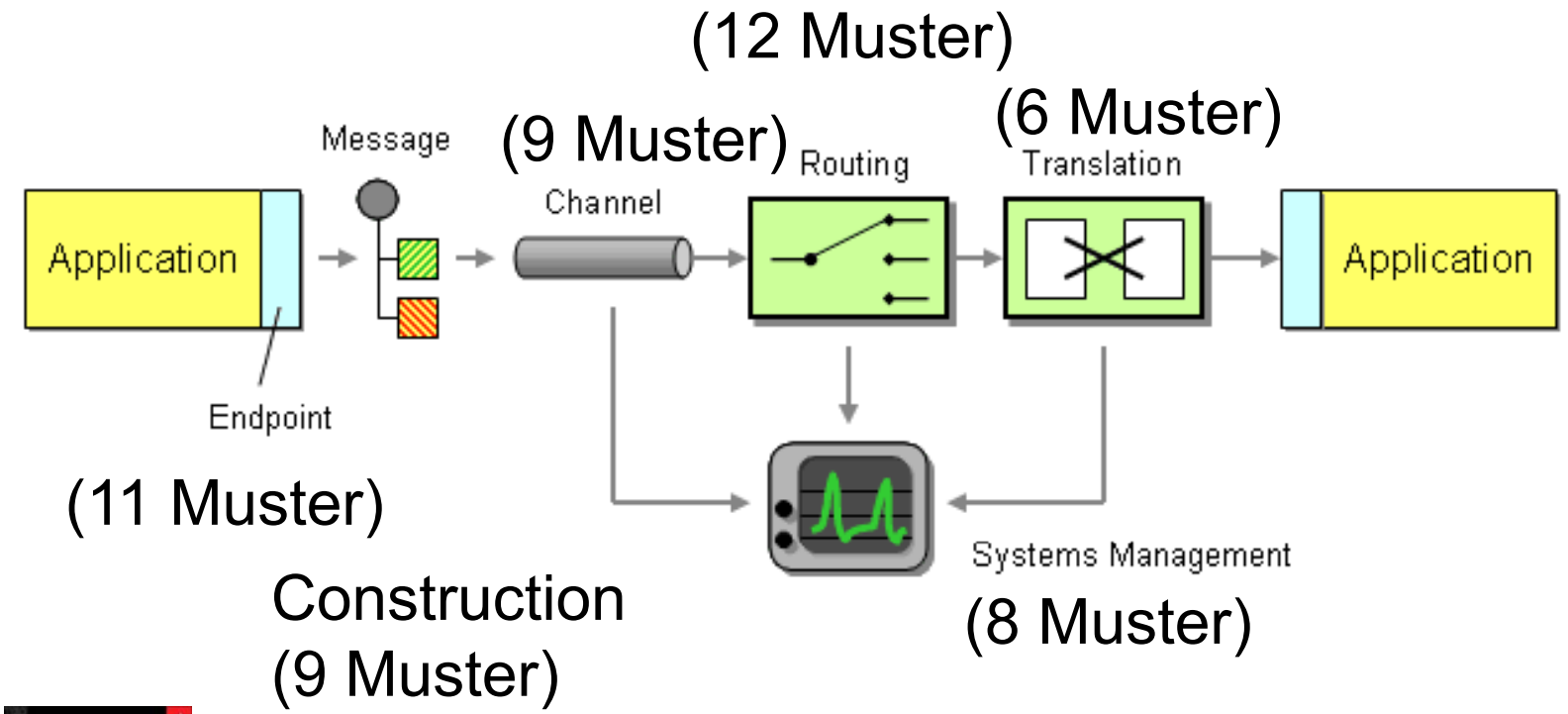


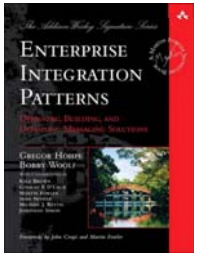




Einfache Integrationslösung mit Mustern

Enterprise Integration Patterns





Enterprise Integration Patterns: Gregor Hohpe, Bobby Woolf, 2003
Katalog mit 65 Mustern

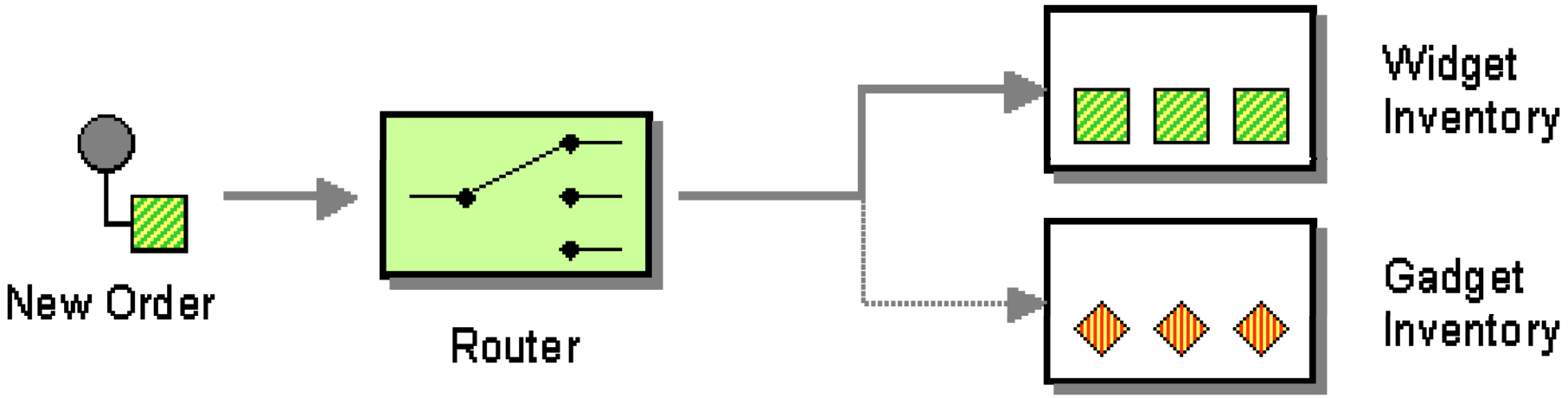
12 EIP-Hauptmuster

Enterprise Integration Patterns

1. Pipes and Filters
2. Message Router
3. Content-Based Router
4. Message Translator
5. Message Filter
6. Dynamic Router
7. Recipient List
8. Splitter
9. Aggregator
10. Resequencer
11. Dead Letter Channel
12. Wire Tap

Content-Router

Enterprise Integration Patterns



Problem:

- Wie kann Empfänger Nachricht abhängig vom Inhalt erhalten?

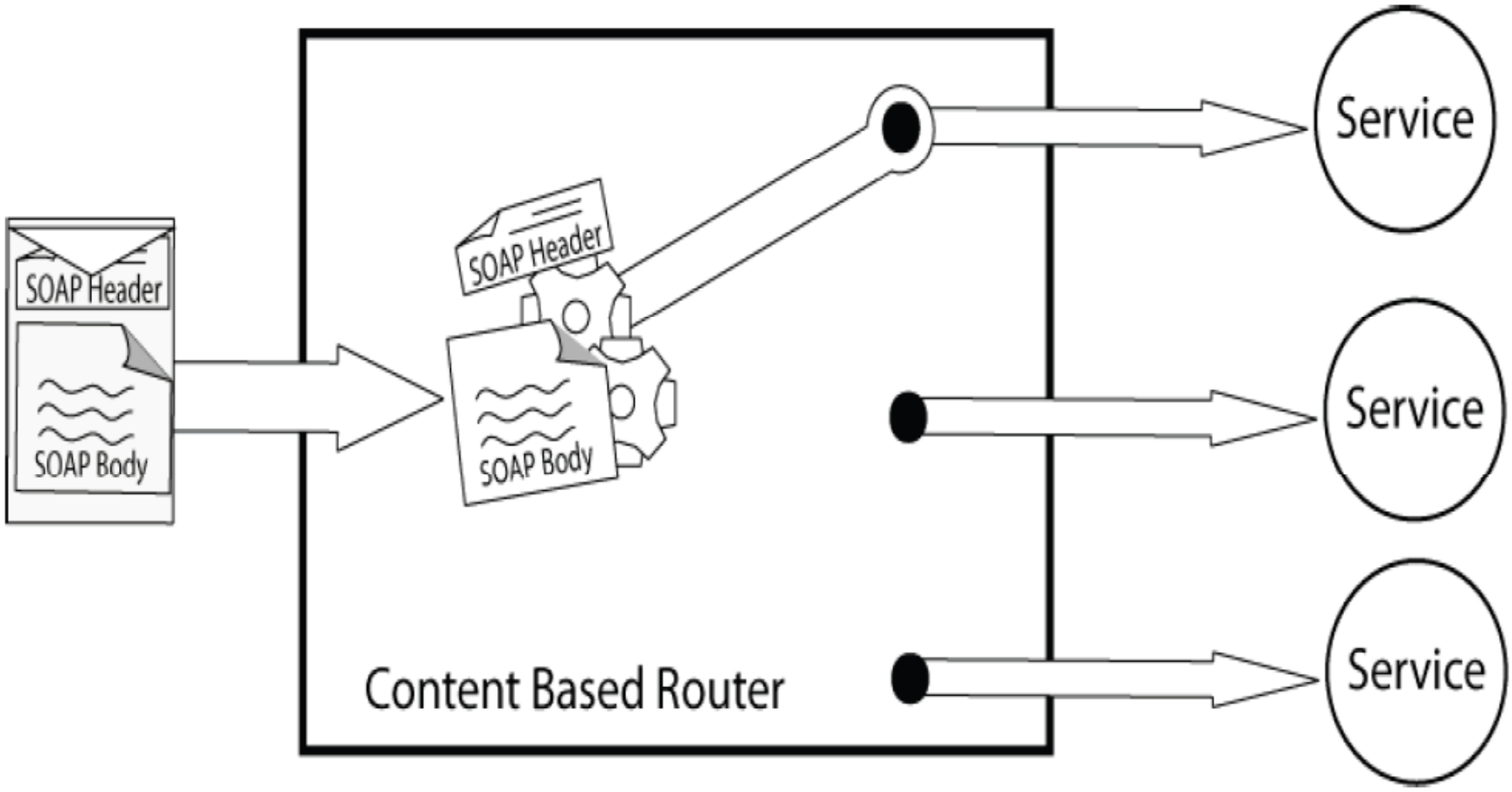
Lösung:

- Ein Content-Router leitet die Nachrichten nach Regeln weiter.

Vorteil:

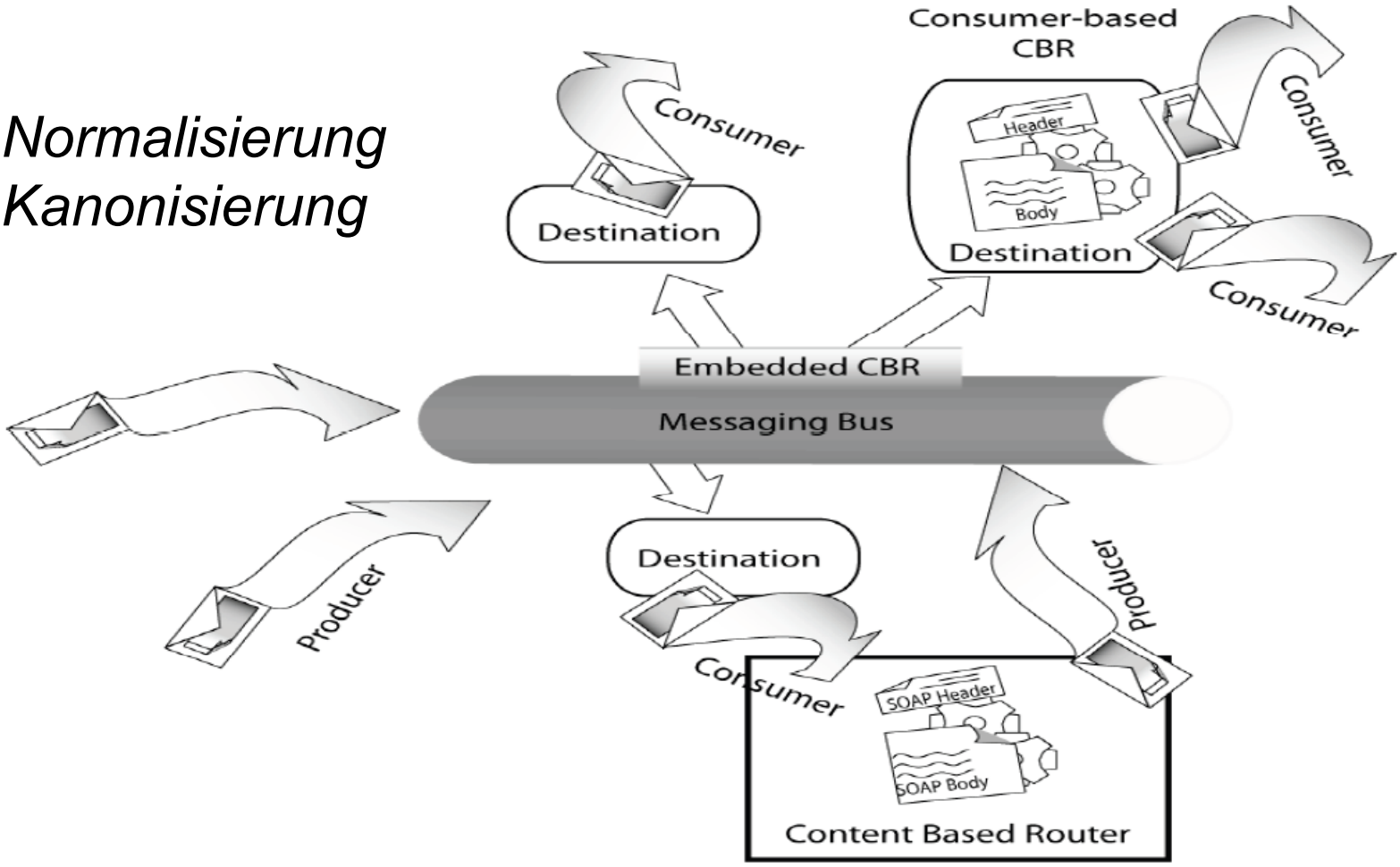
- Transparenz, Anpassbarkeit

Contentrouting mit WebService im Endpunkt

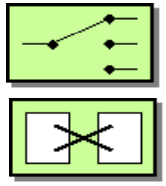


Contentrouting im ESB

Normalisierung
Kanonisierung



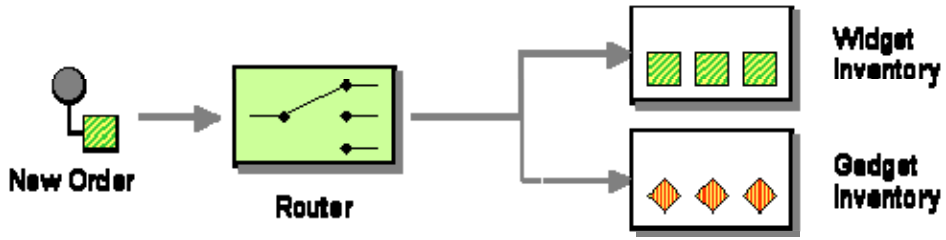
Apache Camel 2.0



- Regelbasierte Routing- und Mediationsengine
- Konfiguration über Java-API, spring-context.xml
- Einsatz mit Apache ServiceMix, ActiveMQ und CXF
- Support für FUSE Mediation Router über Progress
- Unterstützung von 35-41 EAI-Mustern
- Handbuch (498 Seiten)
- camel.apache.org (seit 2007)

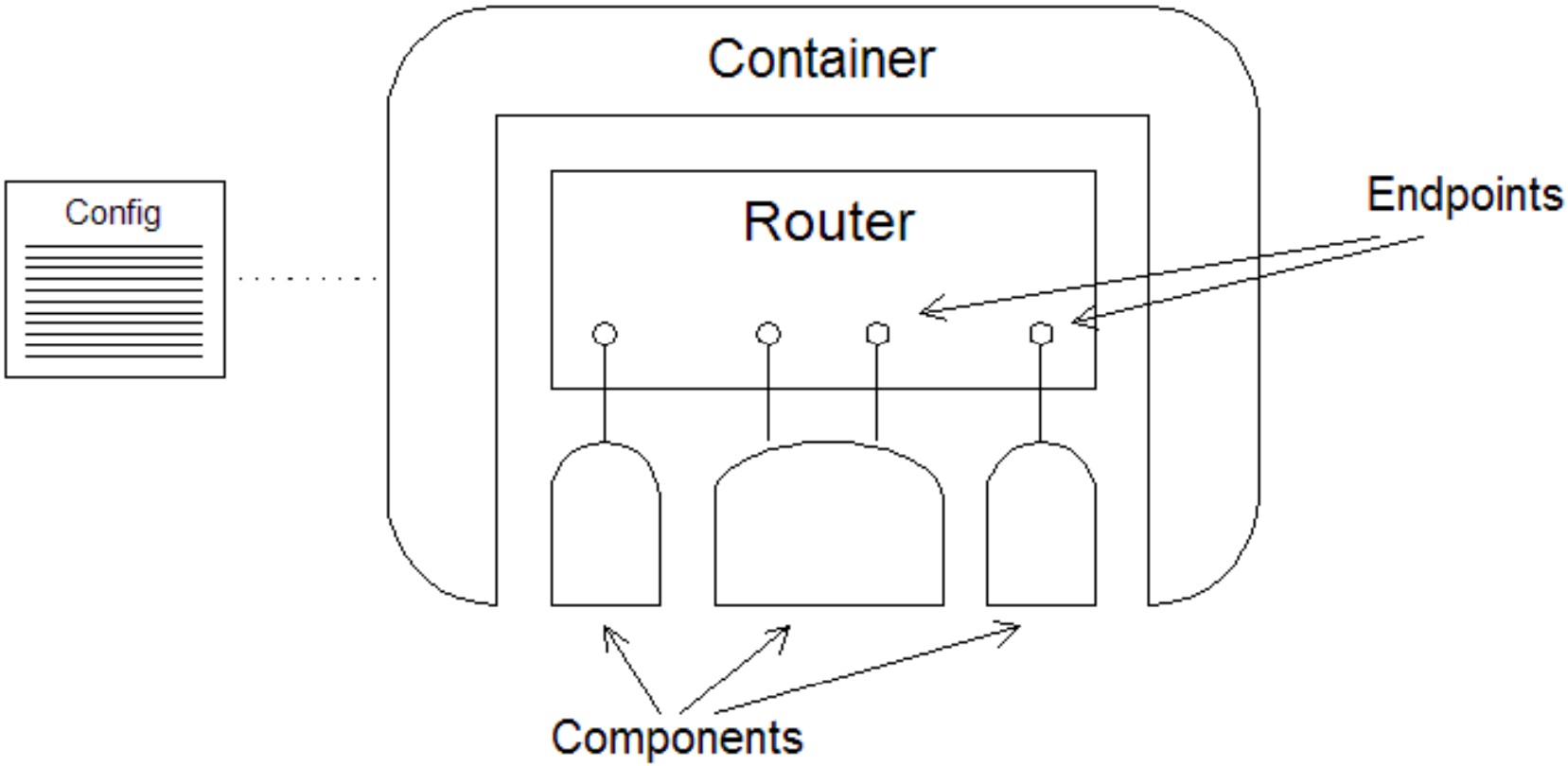
Content-Router: mit Java-DSL und Spring


```
from("jms:queue:order")
.choice()
.when(header("type").in("widget","wiggy"))
.to("jms:queue:order:widget")
.when(header("type").isEqualTo("gadget"))
.to("jms:queue:order:gadget")
.otherwise().to("jms:queue:order:misc")
.end();
```

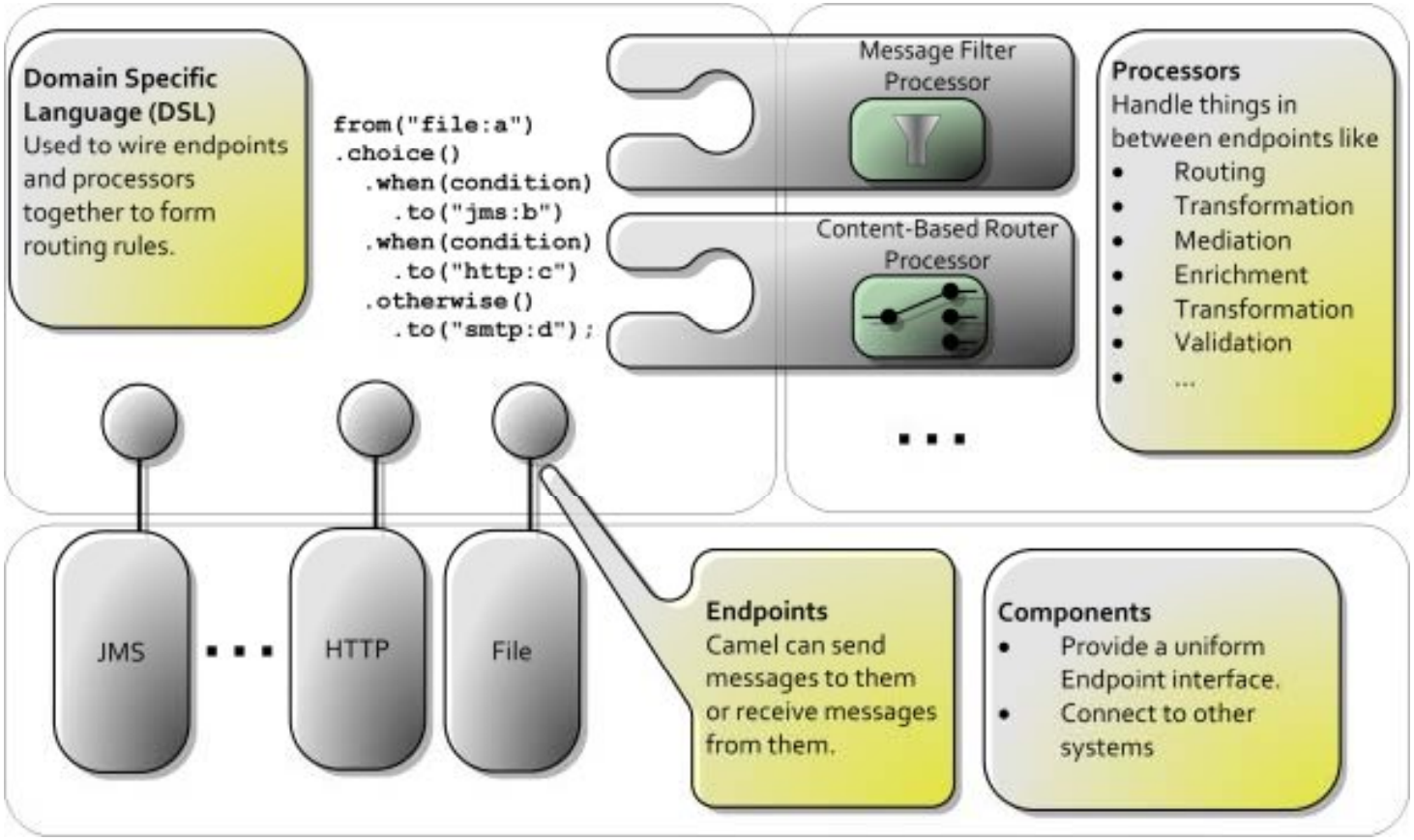



```
<route>
<from uri="jms:queue:order"/>
<choice>
<when>
<simple>${header.type} in
  'widget,wiggy'</simple>
<to
  uri="jms:queue:order:widget"/>
</when>
<when>
<simple>${header.type} ==
  'gadget'</simple>
<to
  uri="jms:queue:order:gadget"/>
</when>
<otherwise>
<to
  uri="jms:queue:order:misc"/>
</otherwise>
</choice>
</route>
```

Camel Architektur



Camel Architektur



Camel 4 Hauptelemente

1. **Komponenten**: Verbindungseinstellungen, Protokolle
2. **Endpunkte**: zum Senden und Empfangen von Nachrichten
3. **Prozessoren**: zum Weiterleiten und Umwandlung der Nachrichten
4. Domain-Specific-Language (**DSL**), **SPRING-XML**: Zusammenstellen der Elemente

Camel Begriffe

- **Endpunkt** ist die Quelle/Ziel einer Nachricht und wird per URI identifiziert
- Der **CamelContext** konfiguriert Routen und enthält die Regeln für den Nachrichtenaustausch zwischen Endpunkten
- Ein **Router** ist ein **CamelContext**-Objekt
- Das CamelContext-Objekt enthält die **Weiterleitungsregeln**, die durch **RouteBuilder**-Objekte und konfigurierte Komponente definiert sind

Camel Komponenten (>70)

activemq	bean	cxf	direct
file	ftp	http	jdbc
jbi	jms	jpa	log
mail	mock	pojo	quartz
rmi	seda	smtp	validation
vm	xquery	xslt	webdav

Wichtigsten Komponenten

- **Bean**: Aufruf von POJO (Plain Old Java Objects) als Java bean
- **Direct**: synchroner Aufruf des Konsumenten-Endpunkt in derselben JVM , wenn Nachricht erzeugt wird
- **File** : Dateioperationen (Lesen, Schreiben, Löschen, Verschieben)
- **VM**: asynchroner Aufruf in derselben JVM

Endpoint URI


```
<component-name>:<component-spec>[?params]

from("file:/tmp?consumer.delay=1000")

from("bean:myBean?methodName=getOrders")

from("test-server:server.cfg?log=#loggingFactoryBean")
```

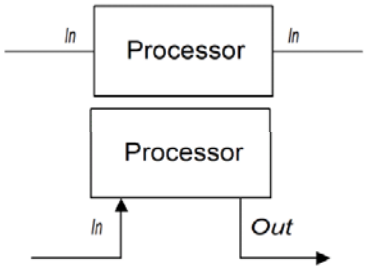
Camel Komponenten URIs (Auswahl)

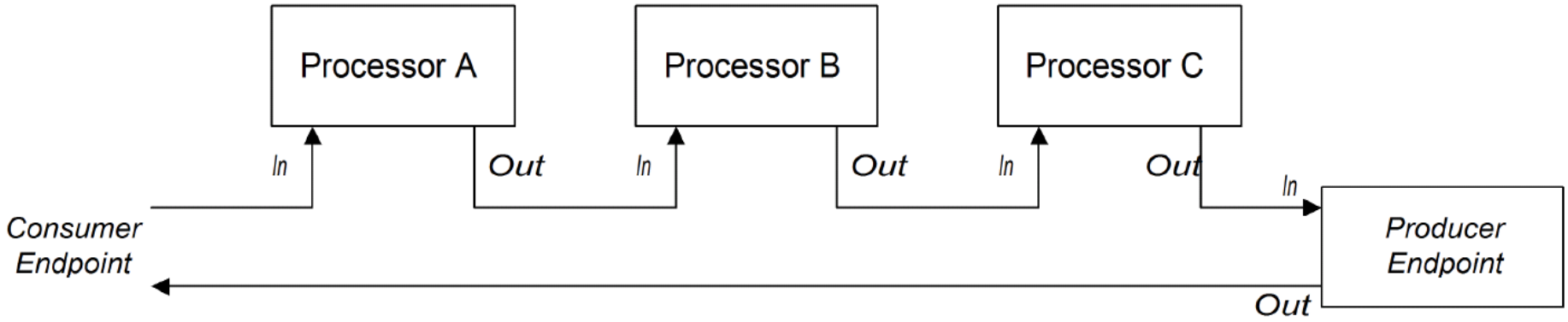
- **ActiveMQ** / activemq-camel/ activemq:[topic:]destinationName JMS Messaging mit Apache ActiveMQ
- **Bean** / camel-core/ bean:beanName[?methodName=someMethod] POJO (Plain Old Java Objects)
- CXF / camel-cxf/ cxf:address[?serviceClass=...] Apache CXF web services
- FTP / camel-ftp/ ftp://host[:port]/fileName
- **File** / camel-core/ file://nameOfFileOrDirectory
- HTTP / camel-http/ http://hostname[:port]
- Hibernate / camel-hibernate in camel-extra/ hibernate://entityName
- JBI / servicemix-camel/ jbi:serviceName
- JDBC / camel-jdbc/ jdbc:dataSourceName?options
- JMS / camel-jms/ jms:[topic:]destinationName
- JPA / camel-jpa/ jpa://entityName
- LDAP / camel-ldap/ ldap:host[:port]?base=...[&scope=<scope>]
- **Log** / camel-core/ log:loggingCategory[?level=ERROR]

Unterstützte Message Exchange Muster

`org.apache.camel.ExchangePattern`

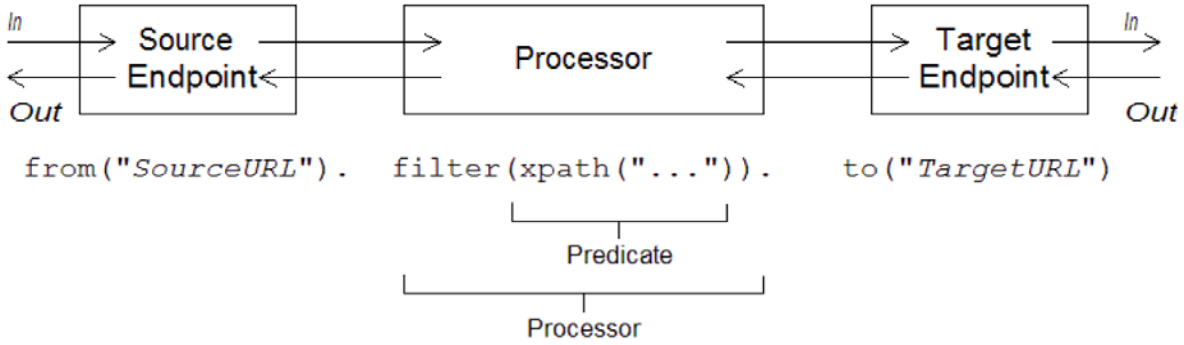
1. InOnly
2. RobustInOnly
3. InOut
4. InOptionalOut
5. OutOnly
6. RobustOutOnly
7. OutIn





Unterstützte Prozessoren

1. Filter
2. Auswahl
3. Pipeline
4. Empfängerlis
5. Splitter
6. Aggregator
7. Resequencer
8. Throttler
9. Delayer
10. Eigenimplementierung



Unterstützte Prozessoren

- Filter:

```
from("SourceURL").filter(header("foo").isEqualTo("bar")).to("TargetURL");
```

- Auswahl

```
from("SourceURL").choice().when(Predicate1).to("Target1")
.when(Predicate2).to("Target2")
.otherwise().to("Target3");
```

- Pipeline

```
from("SourceURL").pipeline("Target1","Target2","Target3");
```

- Empfängerliste

```
from("SourceURL").to("Target1","Target2","Target3");
```

- Splitter

```
from("SourceURL").splitter(bodyAs(String.class).tokenize("\n")).to("TargetURL");
```

Unterstützte Prozessoren

- Aggregator

```
from("SourceURL").aggregator(header("stockSymbol")).to("TargetURL");
```

- Resequencer

```
from("SourceURL").resequencer(header("timeOfDay").batch(new
BatchResequencerConfig(300, 4000L)).to("TargetURL");
```

- Throttler

```
from("SourceURL").throttler(100).to("TargetURL");
```

- Delayer

```
from("SourceURL").delayer(2000).to("TargetURL");
```

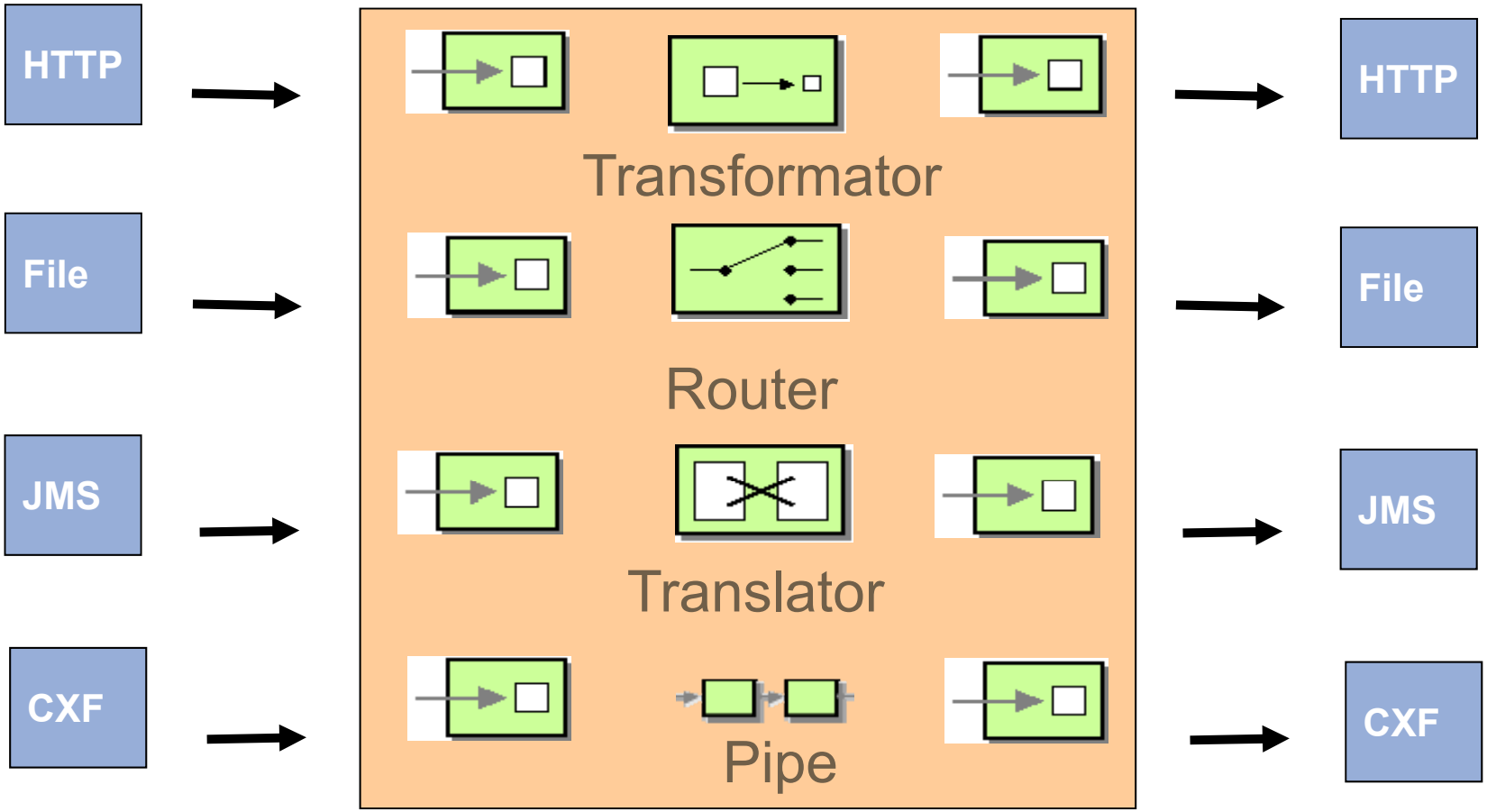
- Eigenimplementierung

```
public class MyProcessor implements org.apache.camel.Processor {

public void process(org.apache.camel.Exchange exchange) {

  inMessage = exchange.getIn(); if (inMessage != null) {inMessage.removeHeader("foo");}}};
```

Apache Camel-Architektur: Muster, Komponenten, Endpunkte



Der erste Ausritt mit Java DSL, CamelContexts und RouteBuilders

1. CamelContext erstellen
2. Endpunkte mit Komponenten hinzufügen
3. Route festlegen mit RouteBuilder oder Xml
4. Context starten

```
CamelContext context = new
 DefaultCamelContext();
// Add components to the
 CamelContext.
// ... (not shown)
// Add routes to the
 CamelContext.
// ... (not shown)
// Start the context.
context.start();
// End of main thread.
```

Einfaches Routing


```
from(“a”).to(“b”)

from("file:///tmp/myFile.txt").

to("bean:MyBean?method=handleMessage").

to("jms:TEST.Q");
```

Einfaches Routen

```
class MyRouteBuilder extends RouteBuilder {
void configure() throws Exception {
 from("direct:a").to("mock:a");
 from("direct:b").to("mock:b");
 }
}
class MyRouteBuilder extends RouteBuilder {//SCALA
 "direct:a" to "mock:a"
 "direct:b" --> "mock:b"
}
```

Werkzeuge

- Maven
- ANT: DOT-Graph
- Web Console zum Monitoren der Endpunkte
- Eclipse:
  - Plugin für Apache Camel
  - Eclipse Templates für Apache Camel
  - FUSE Integration Designer
- Visio, OpenOffice Schablonen

Arbeiten mit der Webkonsole: http://localhost/8080/routes

Apache Camel

Apache Camel

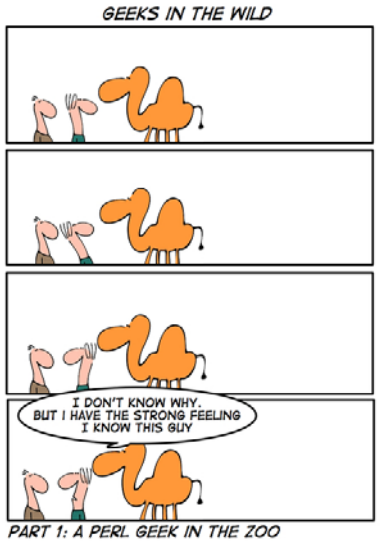
Home | Endpoints | Routes

Route route1

This is an example route which you can start, stop and modify

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<route id="route1" xmlns:ns2="http://camel.apache.org/schema/web" xmlns="http://camel.apache.org/schema/spring">
    <description>This is an example route which you can start, stop and modify</description>
    <from uri="seda:foo"/>
    <to uri="mock:results" id="to1"/>
</route>
```

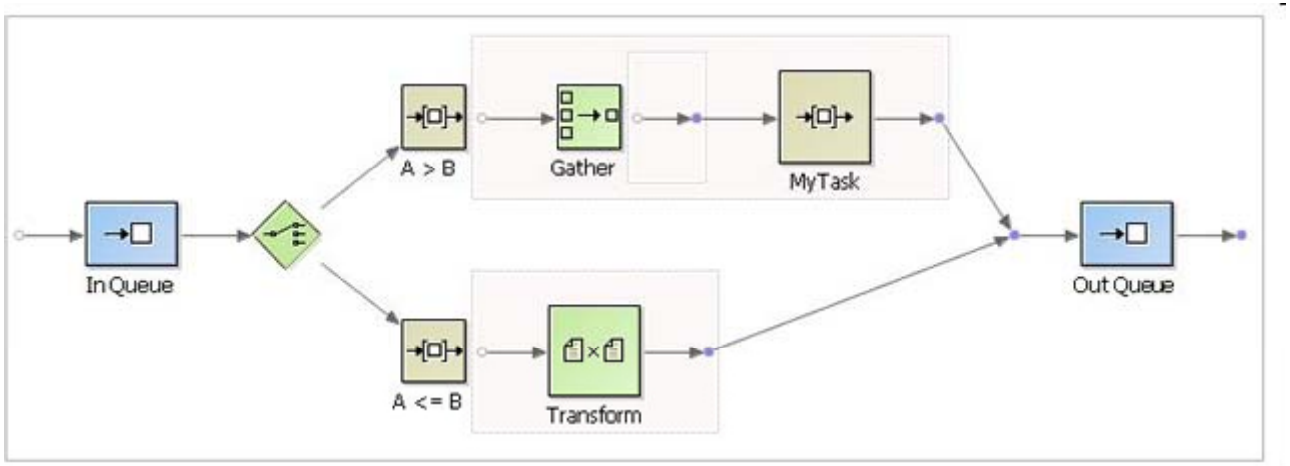
- Edit Route in Xml
- Edit Route in Groovy
- Edit Route in Ruby
- Edit Route in Scala



Visualisierung der Routen

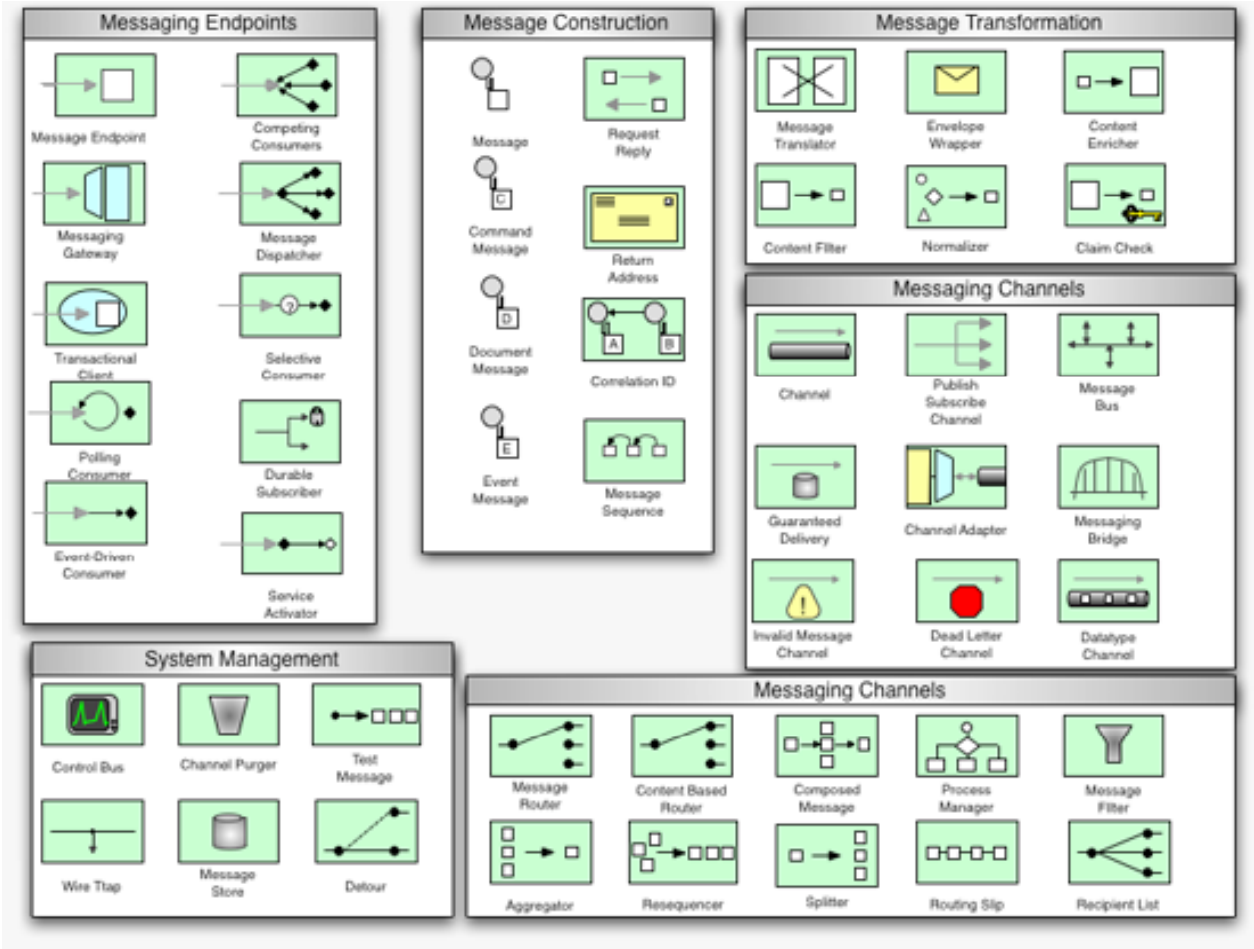
Progress FUSE

- FUSE Integration Designer 1.2



Visio Schablonen

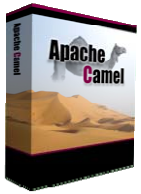
Enterprise Integration Patterns



Vorteile Apache Camel

- Regelbasiertes Weiterleiten
- Transportprotokoll-Übersetzung
- Muster basiert
- Code orientiert (DSL, XML)
- Einfach integrierbar, testbar
- Keine kanonische Normalisierung
- Wahlfreiheit beim Deployment von Camel-Routen
  - JVM
  - Spring basierte Container (ActiveMQ, ServiceMix, JavaEE)
  - OSGi-Container

Änderungen in Camel 2.0



Neuerungen

- 7 Komponenten
- 4 Beispiele
- Skala als Sprache
- Sort EAI-Muster
- 2 Annotationen
- 6 Datenformate

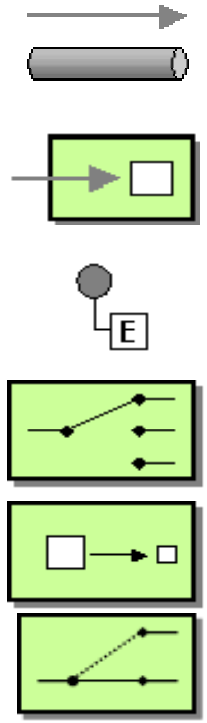
Verbesserungen

- Komponenten: File2, FTP2, JDBC, JMS, ASYNC
- Fehlerbehandlung
- AOP, Interzeptoren
- Testen, Mock, CamelTestSupport-Klasse
- Geschwindigkeit: asynchron, batch, threads, pools
- API-, Schema-Änderungen, Komponenten-Umbenennungen

OpenSource SOA-Stack Apache

- Axis/ CXF (Webservice)
- Tomcat/Geronimo (Applikationsserver)
- Pluto/ Jetspeed (Portal)
- ActiveMQ/ ServiceMix (ESB/JBI)
- Synapse-Axis (WS-ESB)/ Camel-CXF
- ODE (Orchestration Director Engine)
- Tuscany (SCA/SDO/DAS)
- jUDDI (Service Registry)

Musterverwendung in Apache Produkten



Mustername	ActiveMQ	Camel	ServiceMix
Kanal (9)	9	9	9
Endpunkt (11)	8	10	9
Erstellung (10)	9	10	10
Routing (12)	--	8	10
Transformation (7)	2	7	7
Überwachung (8)	3	6	6

Quelle: *Implementing Enterprise Integration Patterns Using Open Source Frameworks Robert Thullner, 2008*

Support für Apache SOA-Produkte über FUSE

CXF

Jax-ws, ws-*, REST

ActiveMQ

JMS, Messaging

Camel

Integration Patterns

ServiceMix

ESB, JBI, OSGi

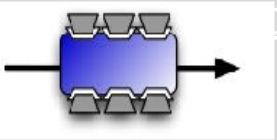
Integrate, test, package, support

Monitor:
FUSE HQ
ActiveMQ Monitor
Camel, ServiceMix Console

FUSE Services Framework

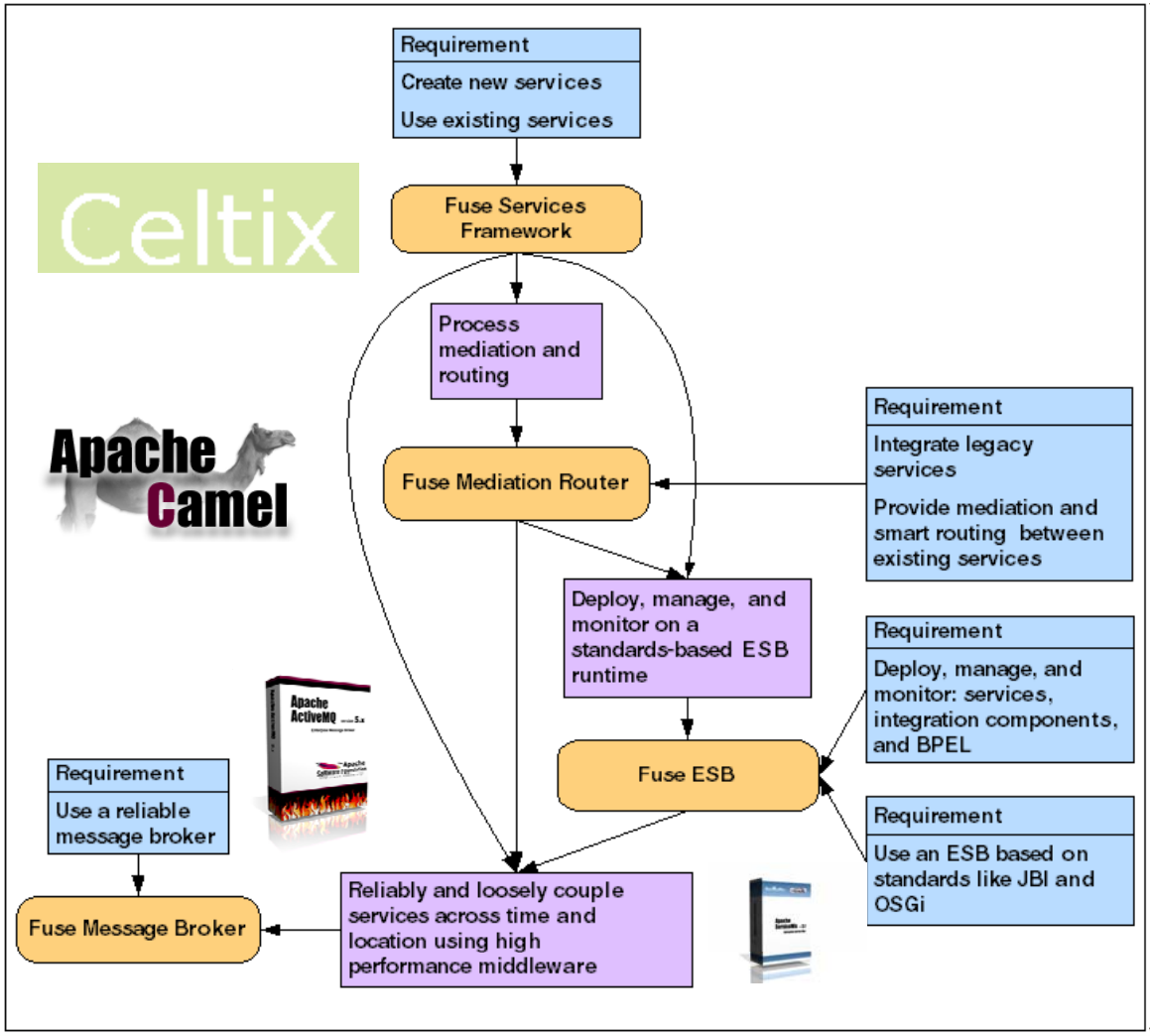
FUSE Message Broker

FUSE Mediation Router
Integration Designer

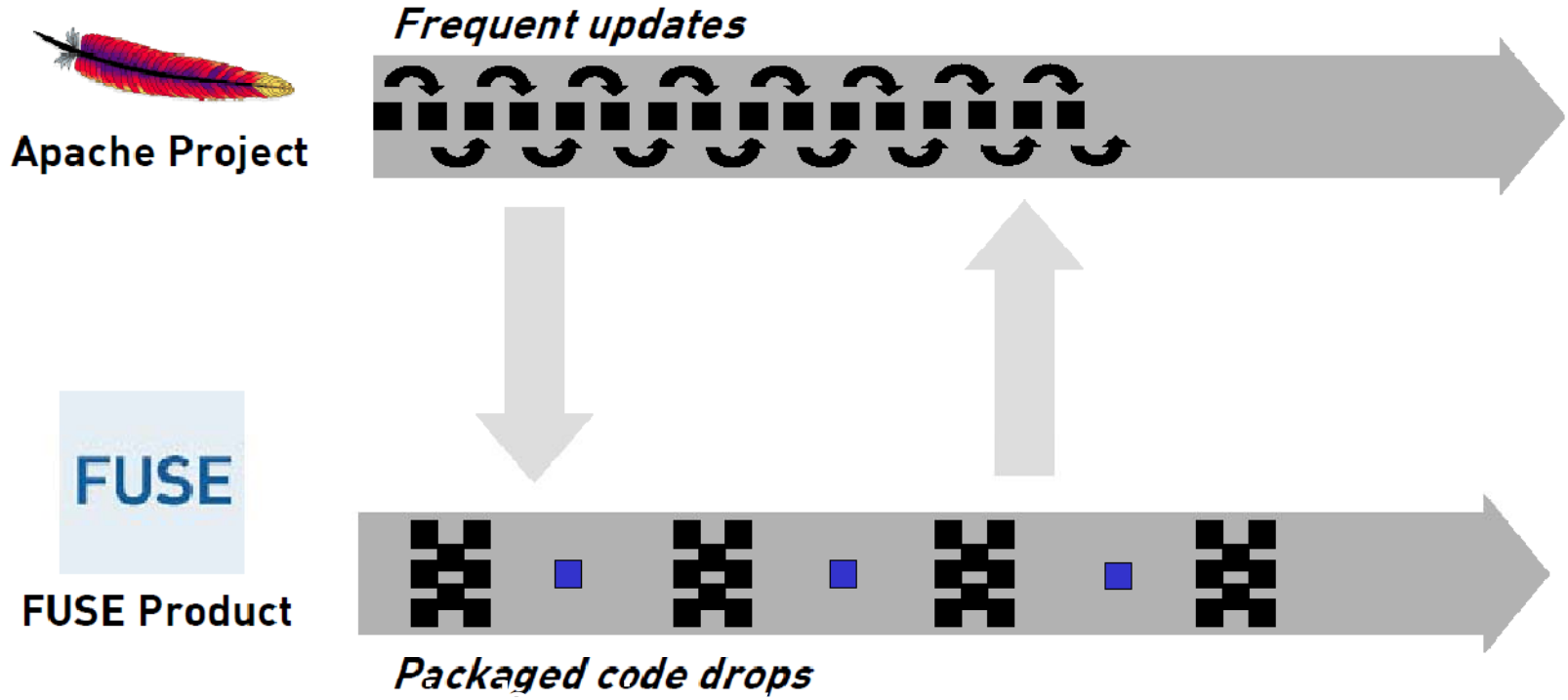


FUSE ESB

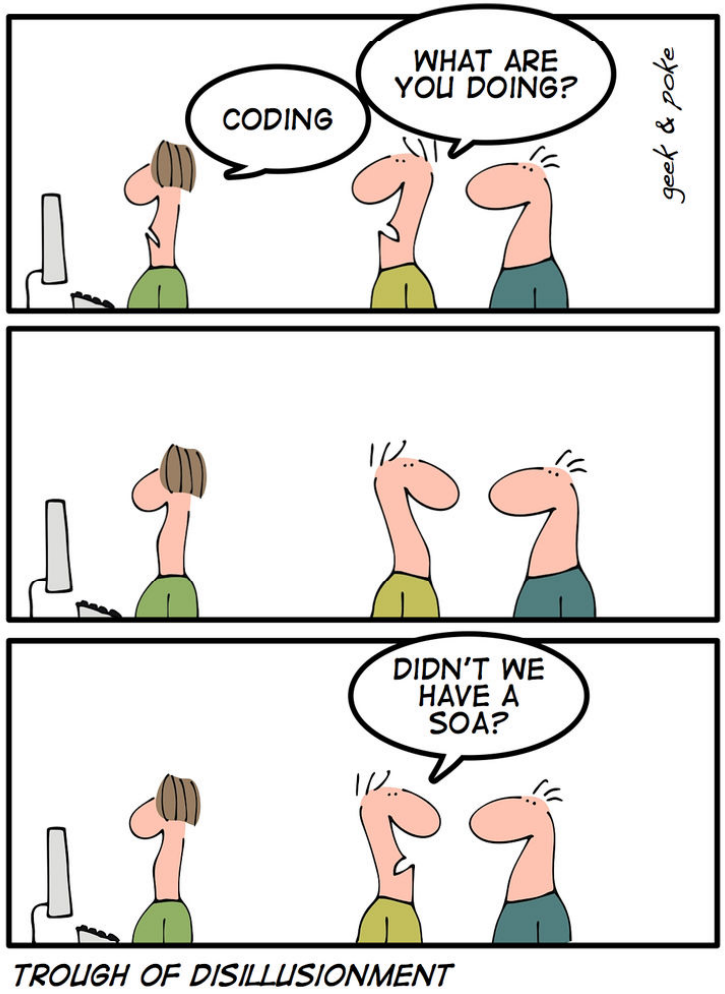
Welches Produkt für welchen Zweck? Entscheidungsbaum (CXF, ActiveMQ, Camel, ServiceMix)



Releasezyklen Apache/Fuse Produkte



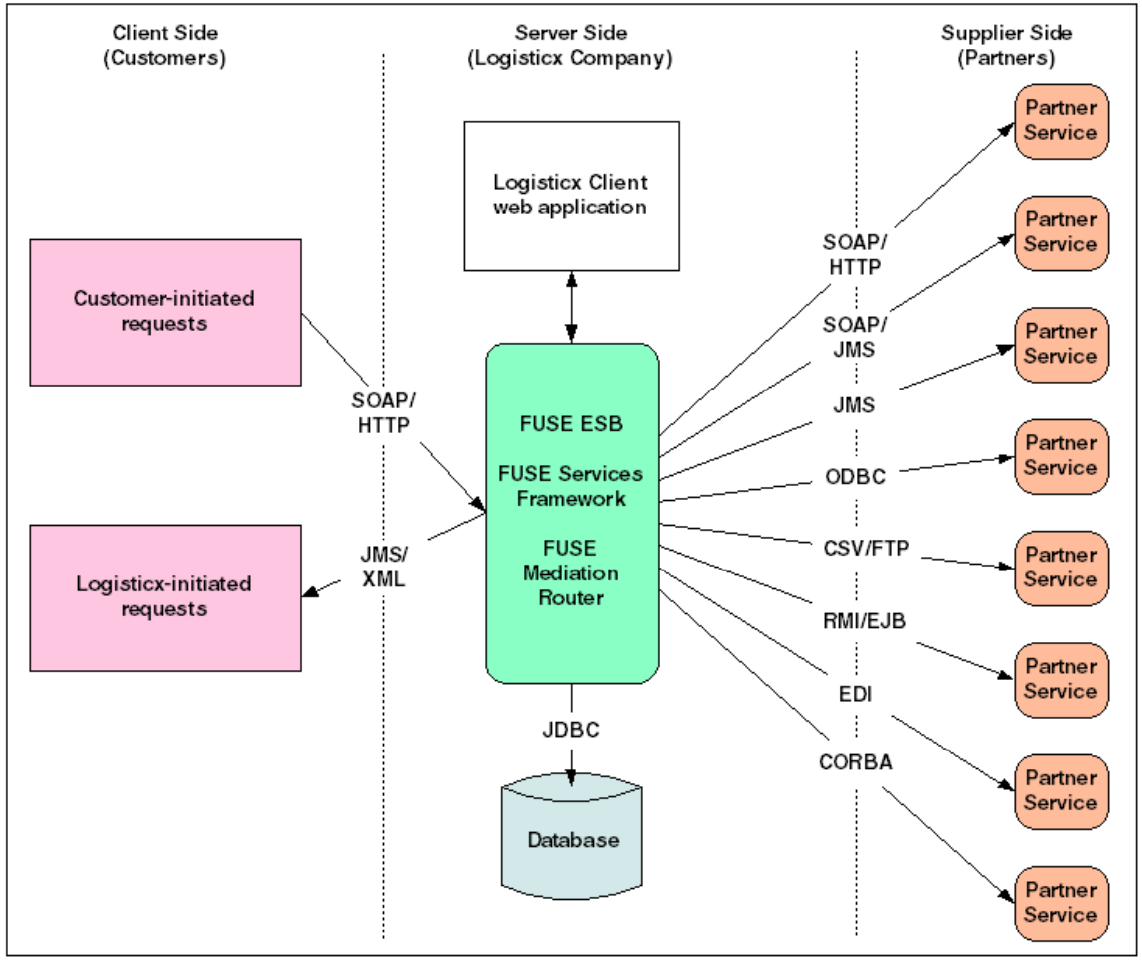
Wie wird SOA umgesetzt?



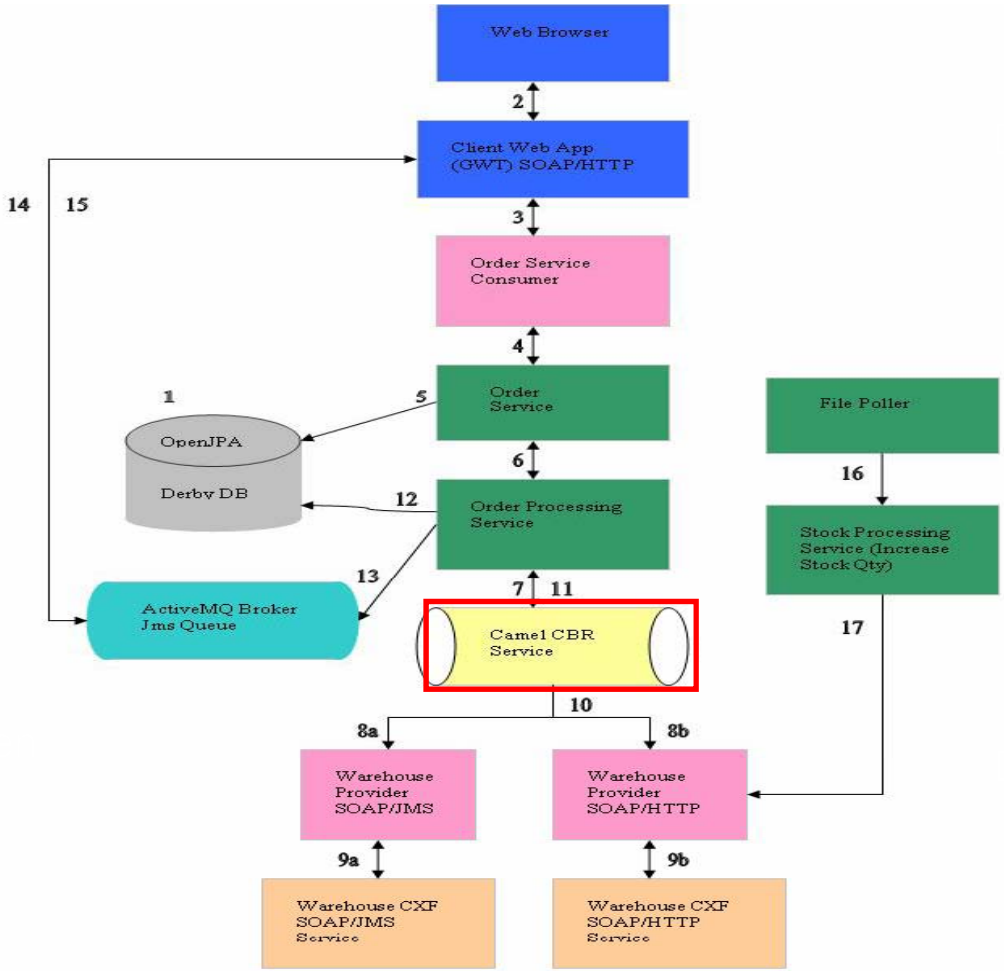
Logisticx-Demoanwendung

- Kunden können über die Logisticx-Firma Bestellungen absetzen.
- Die Logisticx-Firma leitet die Bestellungen an Partnerlieferanten weiter.
- Die Logisticx-Firma kümmert sich um die auslieferung der Bestellungen und um die Verwaltung des Lagerbestandes der Partner
- Es müssen mehre Schnittstellen und Protokolle (HTTP, SOAP, JMS, JDBC) unterstützt werden

Logisticx Demo Überblick Technik



Logisticx-Demoanwendung mit Apache SOA-Produkten



Fazit Apache Camel

- Deklarieren vs Programmieren
  - SPRING vs DSL
- Viele fertige Komponenten
- Viele EIP-Muster
- Gute Werkzeuguntersützung
- Wenig Abhängigkeit an Ablaufumgebung (JVM !)
- Gut dokumentiert, supportet
- Aktive Community
- Bietet leichte, flexible Architektur
  - Gut integrier-, test-, anpassbar
  - Gut erweiterbar: Fuse, ServiceMix, CXF, ActiveMQ, Spring

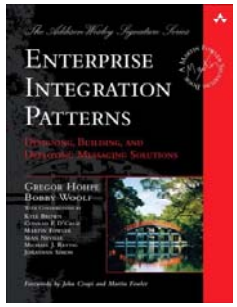
Fazit Apache Camel

- Warum Bus fahren, wenn man auch mit einem Kamel reisen kann?
- Kamele sind:
  - Höhere Säugetiere (Eutheria)
  - Ordnung: Paarhufer
  - Last- und Nutztier
  - Haben einen mehrkammerigen Magen
  - Gut an extreme Umgebungen angepaßt
  - langen, dünnen Hals, einen kleinen Kopf, relativ langgestreckte, schlanke Beine
  - Hybride sind fortpflanzungsfähig
- Mit einem Kamel kommt man auch durch unübersichtliche SOA-Landschaften

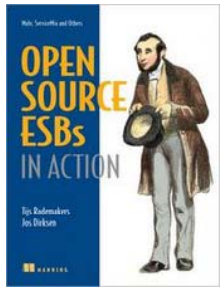
Links

- Apache Camel camel.apache.org
- FUSE Mediation Router fusesource.com/docs/router
- Enterprise Integration Patterns ww.eaipatterns.com
- Artikel:
  - wiki.apache.org/ws/StackComparison
  - architects.dzone.com/articles/apache-camel-integration
  - architects.dzone.com/articles/pattern-based-development-with-0
- Blogs:
  - macstrac.blogspot.com/search/label/camel
  - davsclaus.blogspot.com
  - janstey.blogspot.com
  - fusesource.com/resources/combined-blogs

Bücher



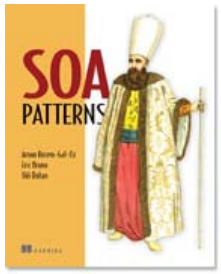
- Enterprise Integration Patterns, Gregor Hohpe, Bobby Woolf, Addison-Wesley, 2003



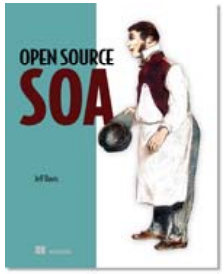
- Open-Source ESBs in Action, Tijs Rademakers, Jos Dirksen, Manning, 2008

- Apache Geronimo, Frank Pientka, dpunkt, 2009

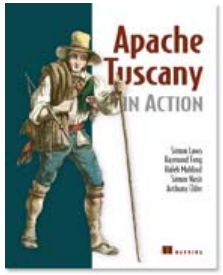
Bücher



- SOA Patterns, Arnon Rotem-Gal-Oz, Eric Bruno, Udi Dahan, Manning, 2010



- Open Source SOA, Jeff Davis, Manning, 2009



- Apache Tuscany in Action, Simon Laws, Manning, 2009

14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank & Erfolg beim Kamelreiten!

Frank Pientka

MATERNA GmbH, Dortmund