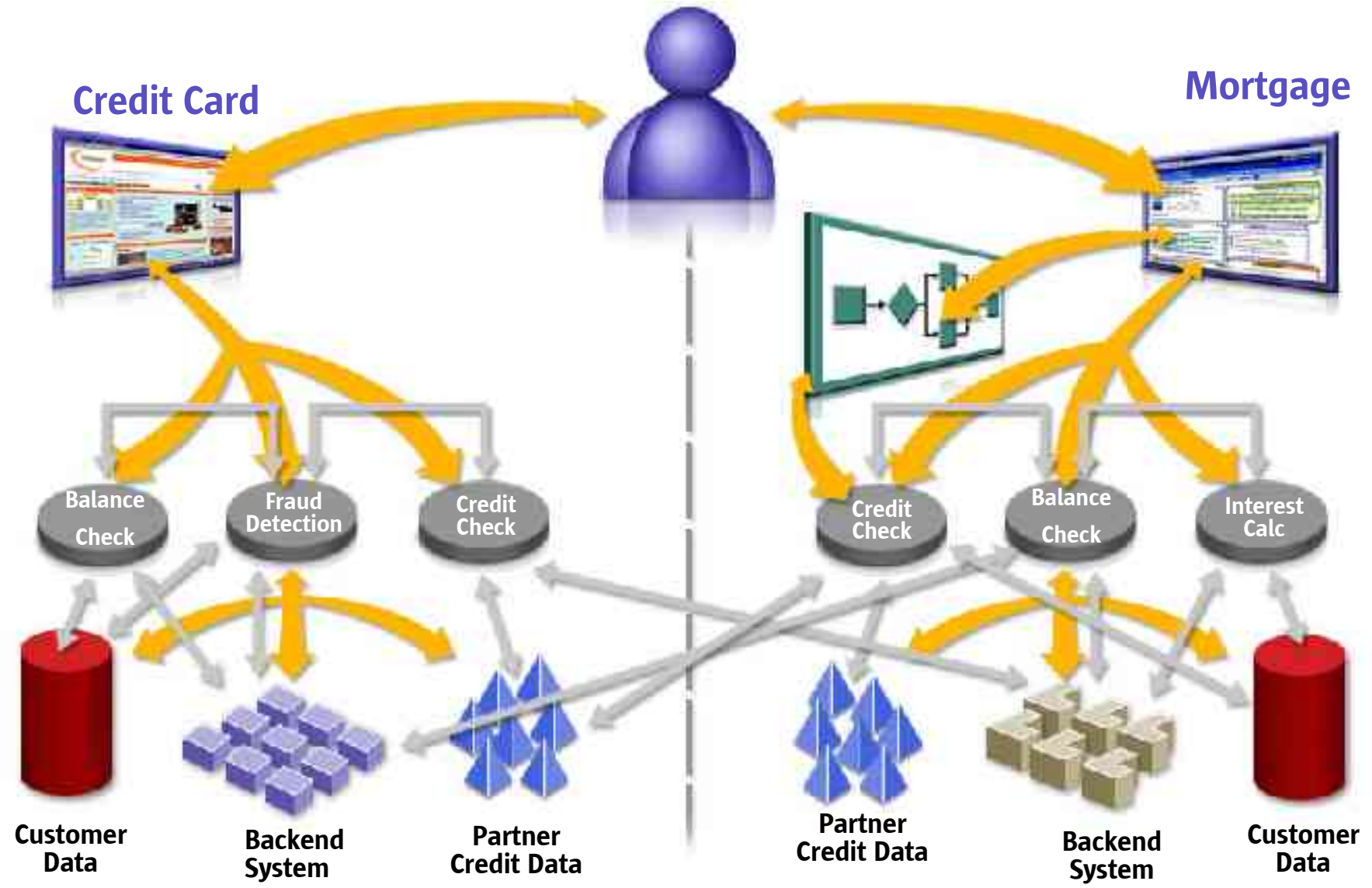
JSR-208: Java Business Integration

Armin Wallrab
Software Architect
Enterprise Web Services Practice
Sun Microsystems GmbH

Agenda

- Service Oriented Architecture (SOA)
- Java Business Integration (JBI)
  - Komponenten
  - Messaging Modell
  - Packaging
  - Management
- Entwicklung einer Service Engine
- Zusammenfassung
- Q&A

IT Architektur heute



Forderungen an eine moderne IT

- Kosten
  - > gekürzte Budgets
  - > weniger Mitarbeiter
  - > Controlling
- Präsenz
  - > Globalisierung
  - > Multi Channel
  - > Security
- Komplexität
  - > Übernahmen & Merger
  - > Offshoring & Outsourcing
  - > Heterogenität
- Agilität
  - > Wettbewerb
  - > Standardisierung & Regulierung
  - > Vermeidung des “Vendor Lock-in”

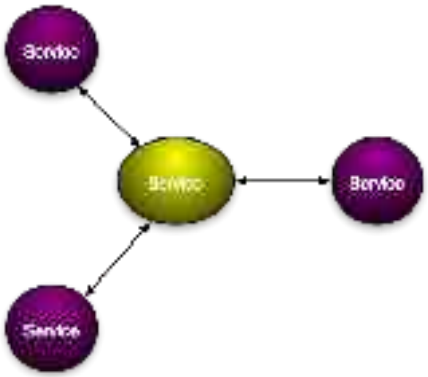
SOA – Philosophie

“Business drives IT”

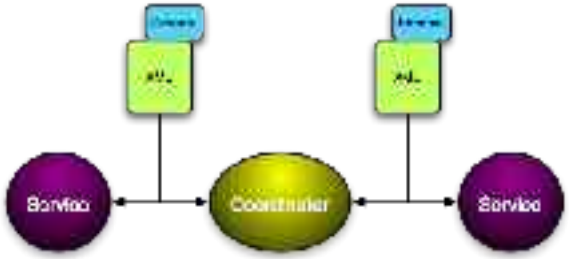
- Evolution statt Revolution
  - Think big – start small
  - Kapselung und Wiederverwendung
  - Heterogenität nicht eliminieren, sondern managebar machen
- Bessere Kommunikation zwischen Business und IT
  - Data Dictionary
  - Zentrales Architekturmanagement
- Integration von IT Systemen **und** Prozessen
  - Aktives Management
- Standardisierung von Systemen, Software und Interfaces

Technische Merkmale

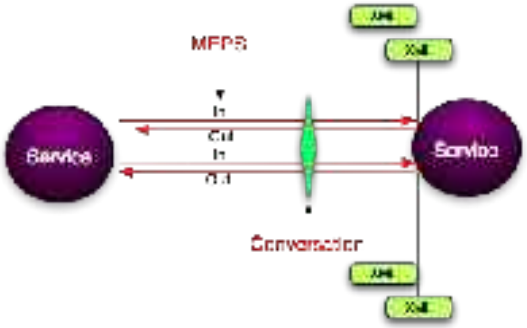
coarse grained



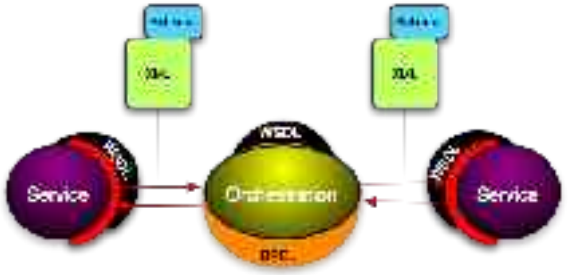
XML document based
WSDL described
registered & discoverable



asynchronous & conversational

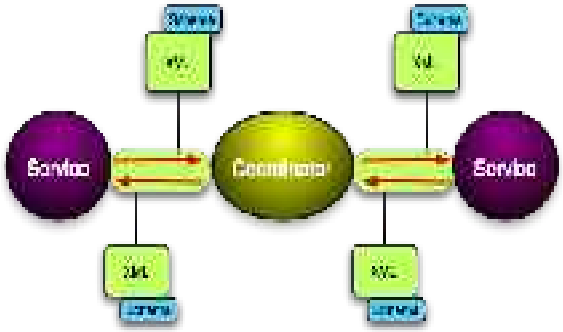


orchestrated

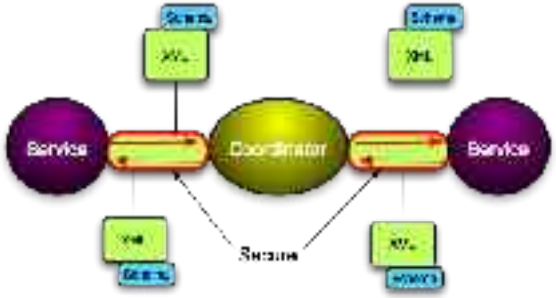


Weitere Anforderungen

reliable

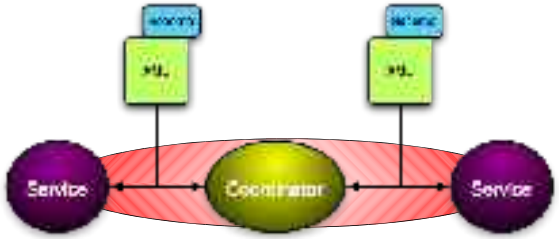


secure

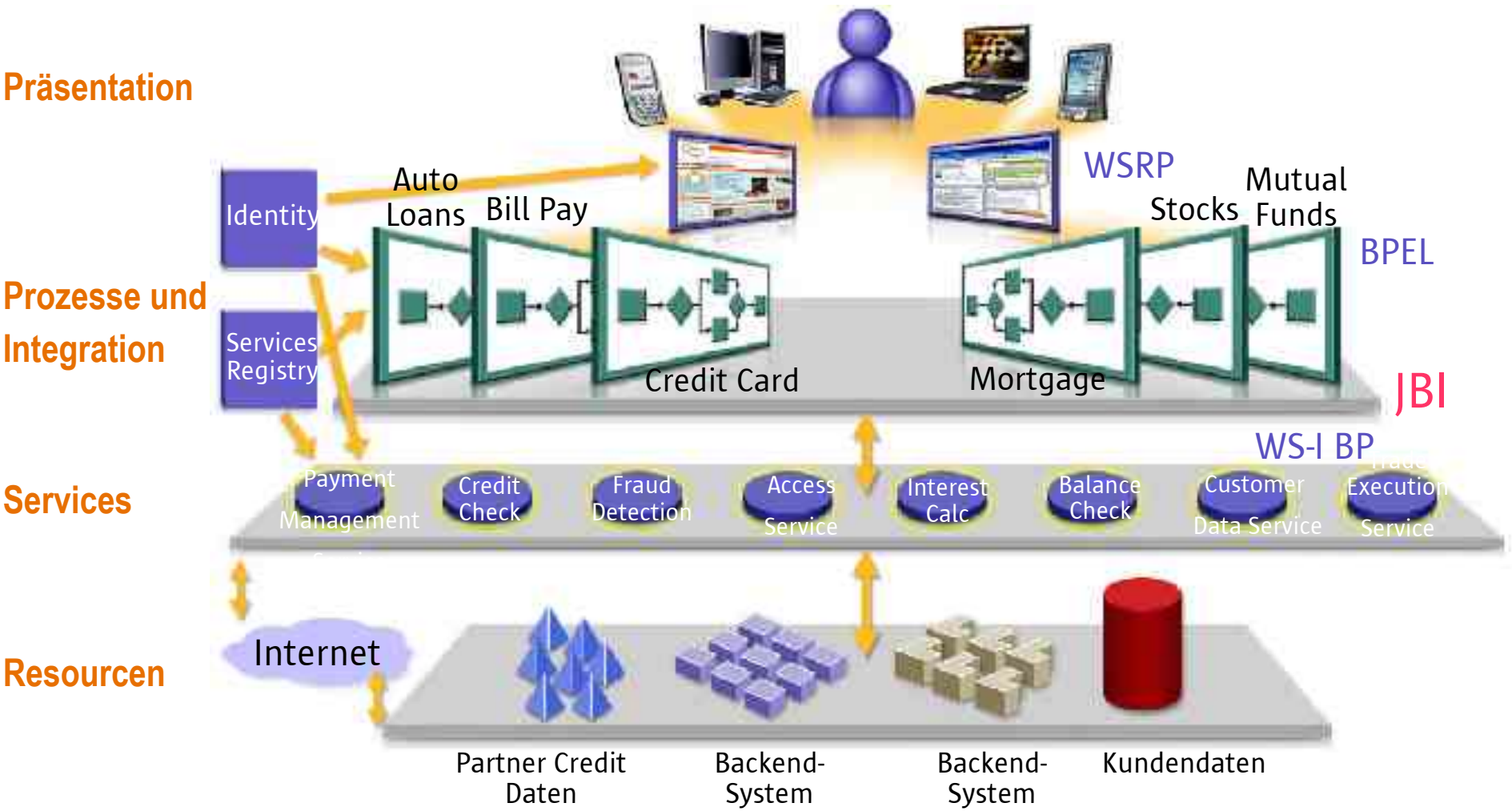


policy driven

transactional

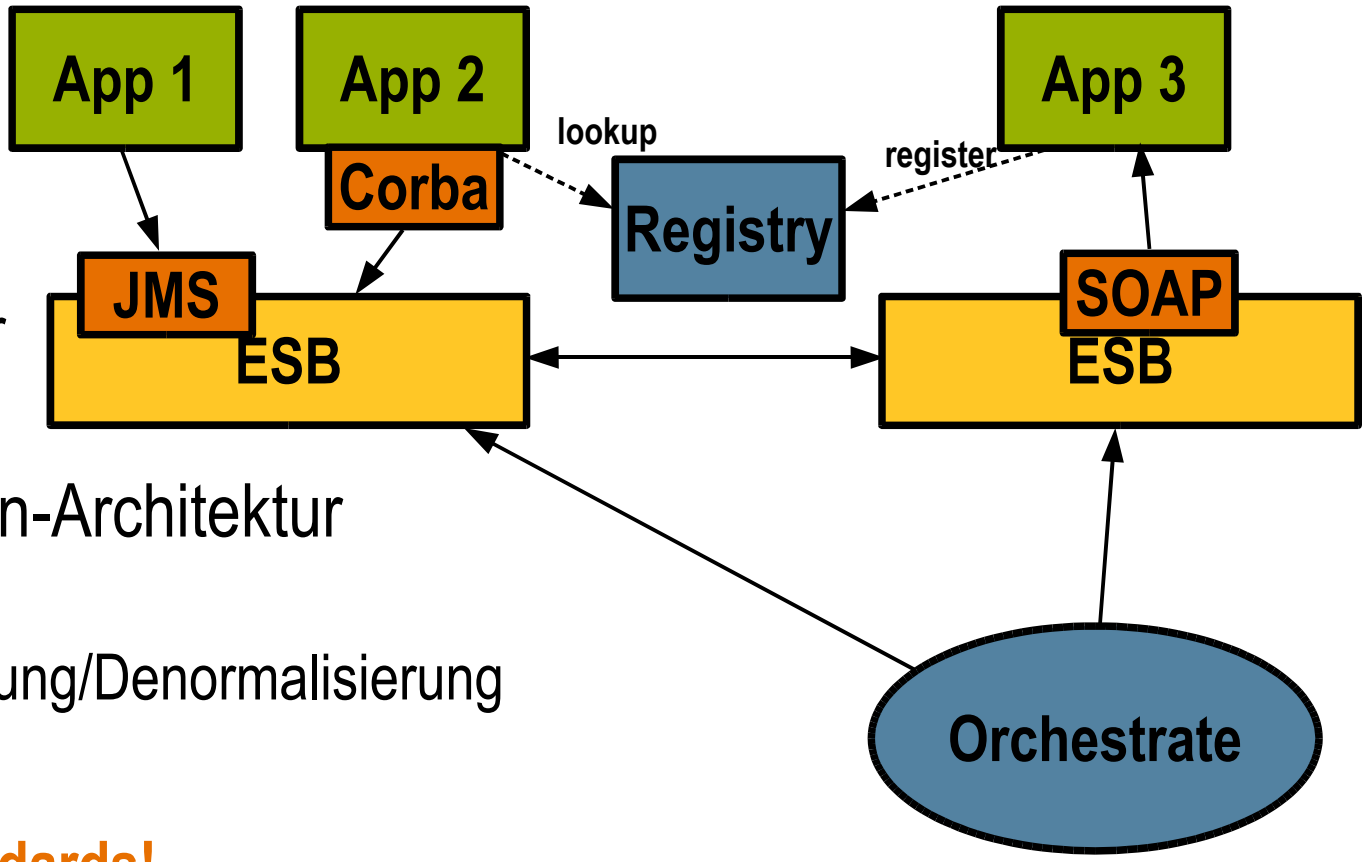


SOA – Schichten



Beispiel: Enterprise Service Bus (ESB)

- Verteilung
  - > Infrastruktur
  - > Registry
- Komponenten-Architektur
  - > Plugins
  - > Normalisierung/Denormalisierung
- Aber:
  - > **Keine Standards!**



Warum JBI?

“SOA does not need Java – Java needs SOA.”

Mark Hapner, Sun Microsystems

- Integrationsfähigkeiten in den Java-Kern bringen
- Entwicklung eines Industriestandards
- Entwicklung einer Basistechnologie für SOAs
- Herstellerunabhängigkeit – höhere Agilität
- Basierend auf “Best-Practices” und Standards

Prinzip: “Collaboration and Competition”

JSR-208 Expert Group

- Apache
- Borland
- Cap Gemini
- Collaxa
- Deutsche Post
- Fujitsu
- Intalio
- IONA
- IOPSIS
- JBoss
- Nokia
- Novell
- Oak Grove Systems
- Oracle
- Research in Motion
- SAP
- SeeBeyond
- Sun Microsystems
- Sybase
- TIBCO
- TMax Soft
- Vignette
- WebMethods
- ...

Missing:
IBM, BEA

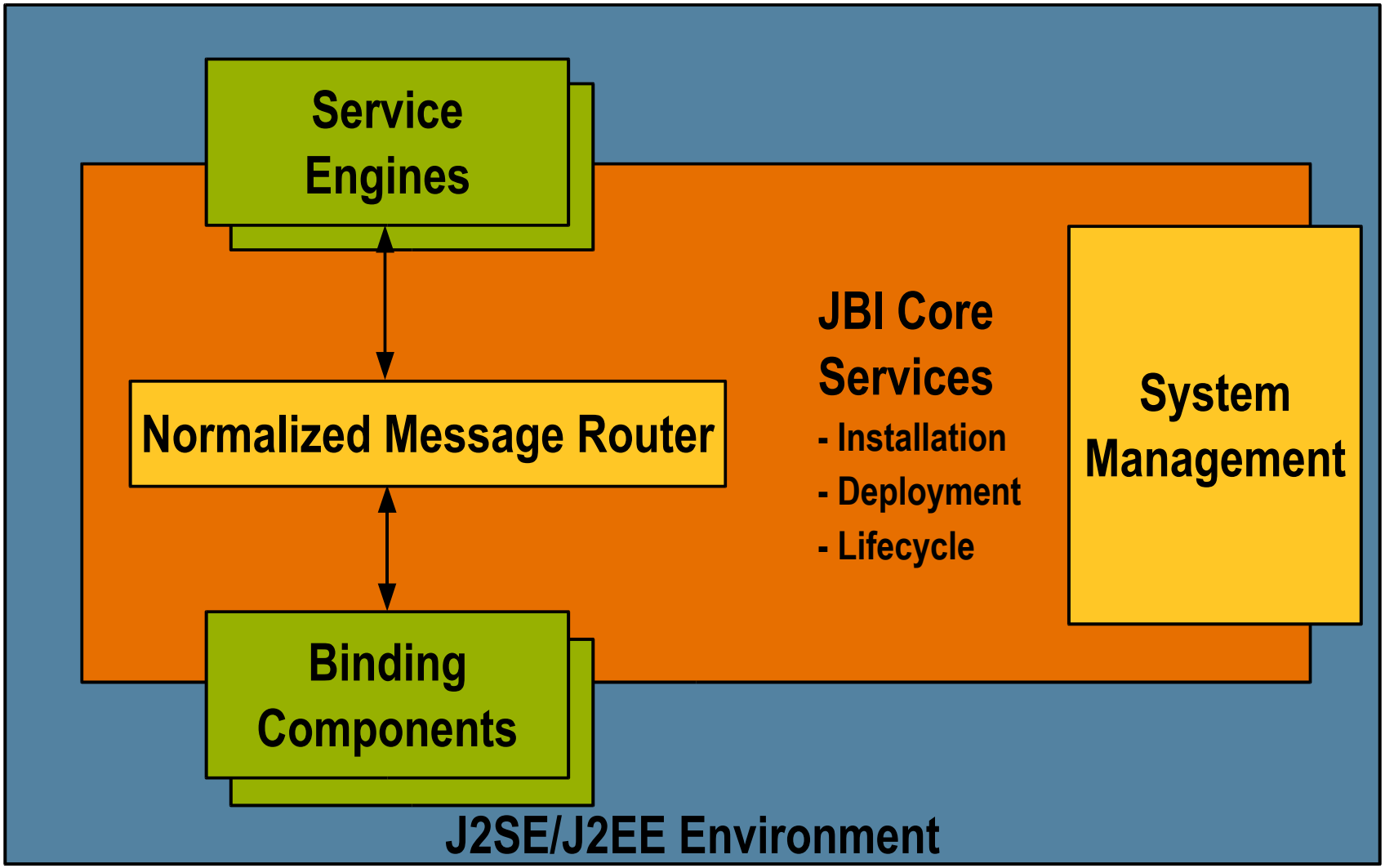
Java Community Process (JCP)

- JSR-208: http://jcp.org/en/jsr/detail?id=208
- Expert Group
  - Spezifikation
- Sun
  - Reference Implementation (RI)
  - Test Compatibility Kit (TCK)
- JBI 1.0 SDK: http://java.sun.com/integration/download
  - Basiert auf Sun Java Enterprise System Application Server PE 8.1
  - Enthält
    - Referenz Implementierung (mit jeweils zwei Service Engines & Binding Componenten)
    - Sourcefiles einer Service Engine und einer Binding Component
    - Deployment-Artefakte für ein umfassendes Beispiel-Szenario

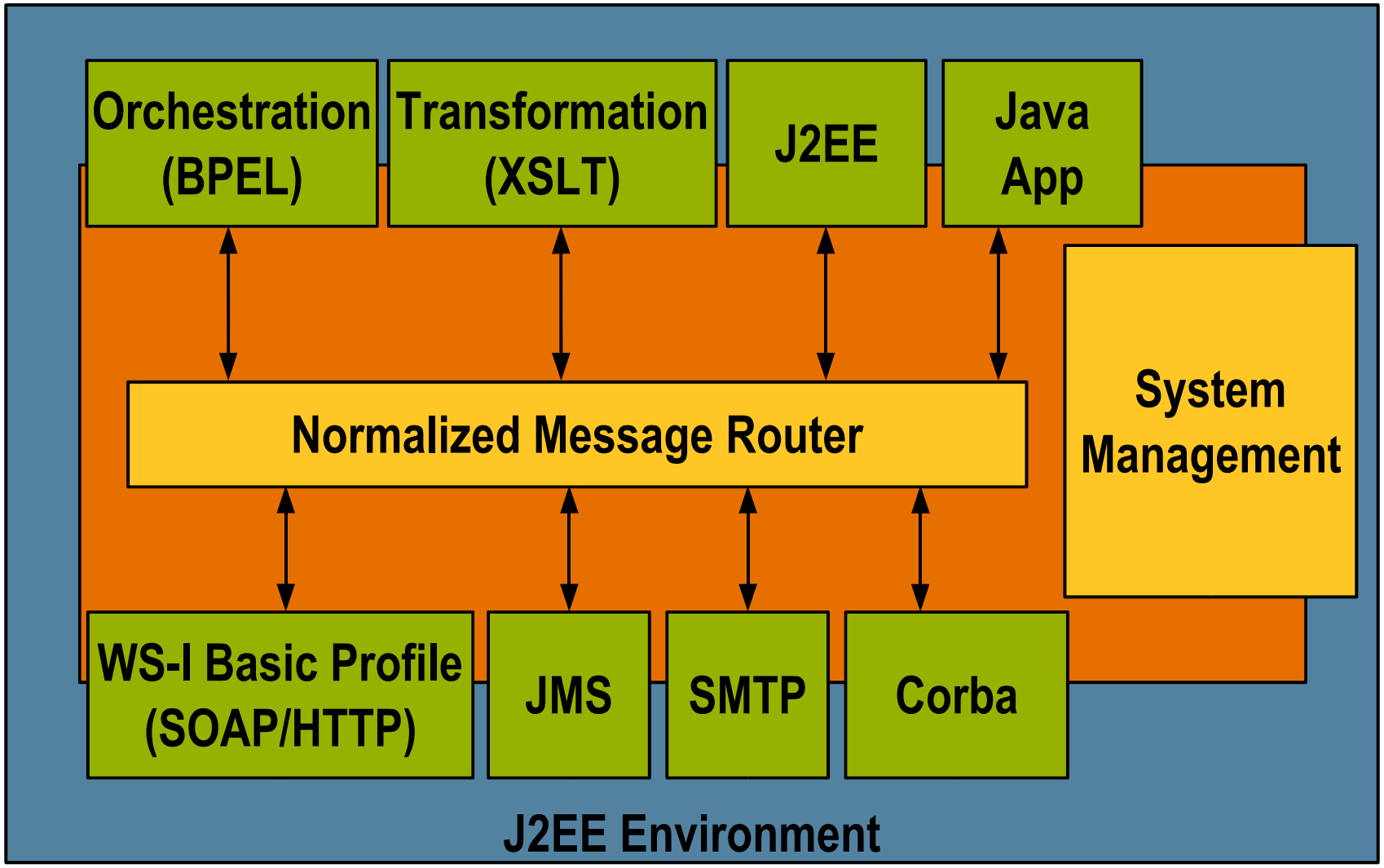
JBI – Überblick

- Service-orientierter Meta-Container
- Komponenten-Architektur
  - Plugins
  - Standardisiert: Schnittstellen und Packaging
- Komponenten-Arten
  - Service Engines
    - Business Logik
    - Services
  - Binding Components
    - Proxy für Service Consumers
    - Remote Service Providers
- System Management mit JMX

JBI – Überblick



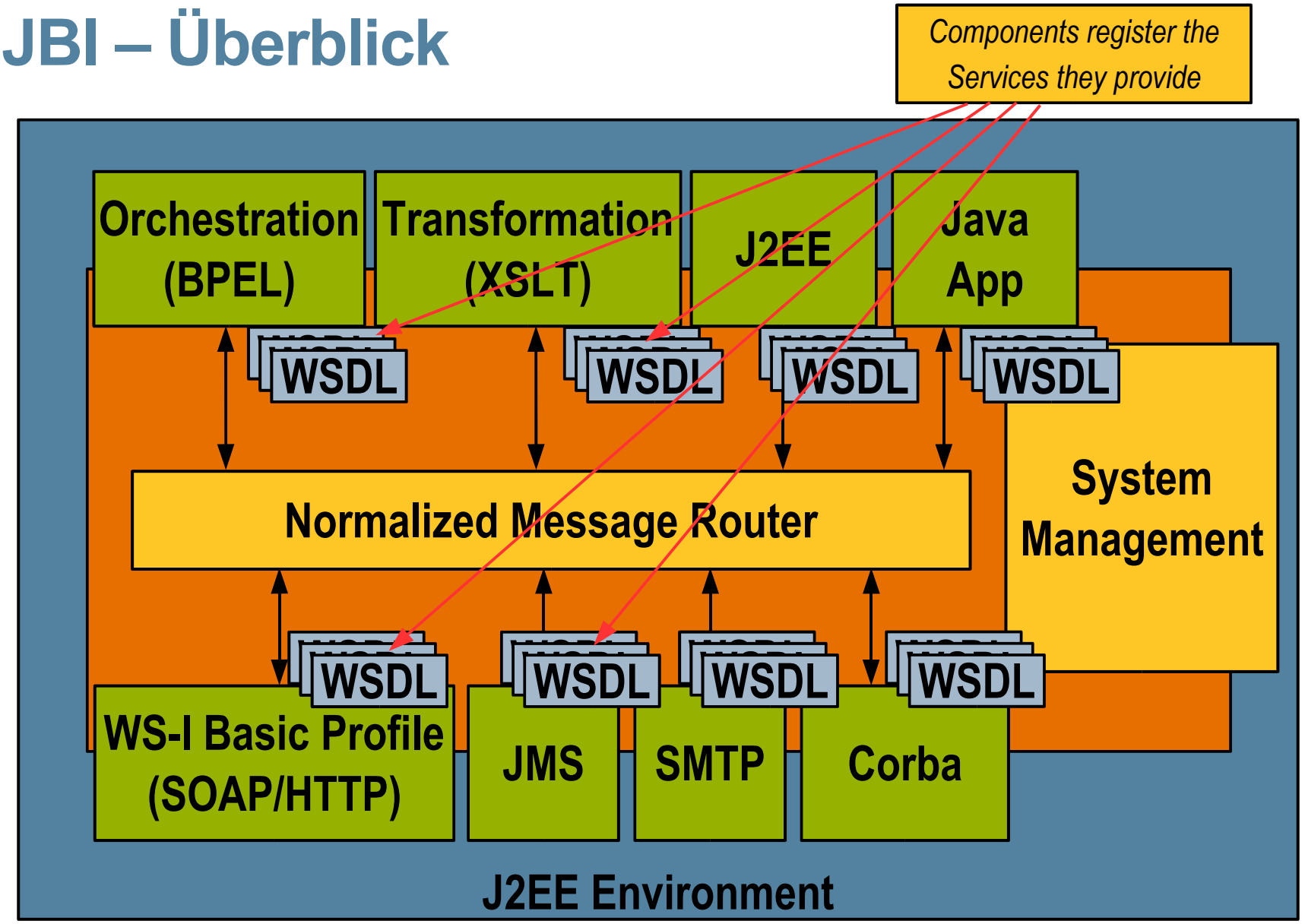
Beispiel-Umgebung



Service-orientierte Aspekte

- Plugin-Komponenten implementieren Rollen
  - Service Consumer
  - Service Provider
  - ... oder beides zugleich
- Provider sind “self-describing”
  - Message-Typen
  - Operationen
  - Services
  - Endpoints
  - Message Exchange Patterns
  - ... basierend auf WSDL 1.1 und 2.0
- Document-centric, XML

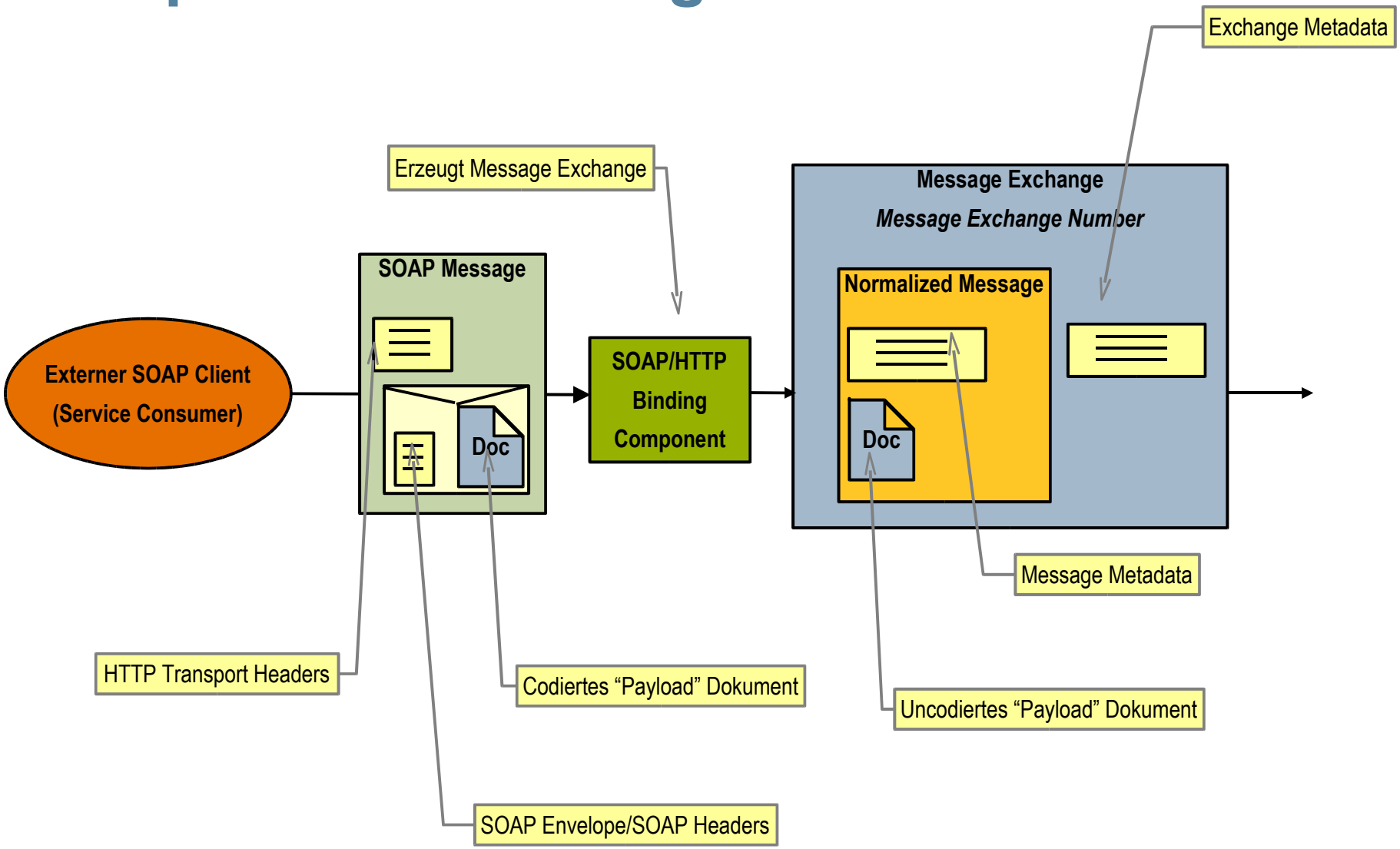
JBI – Überblick



Normalized Messages (NM)

- Komponenten erzeugen NMs aus “Bound Messages”
- Payload: WSDL abstract message part
  - WSDL 2.0 Interface Operation Message Definition
- Metadata: generische Properties
  - Protokoll-spezifische Kontext-Informationen
  - Security Tokens
  - Transaktions-Informationen
  - ... alle anderen Informationen, die eine weitere Komponente verstehen könnte
- Kein *kanonisches* Message Format!

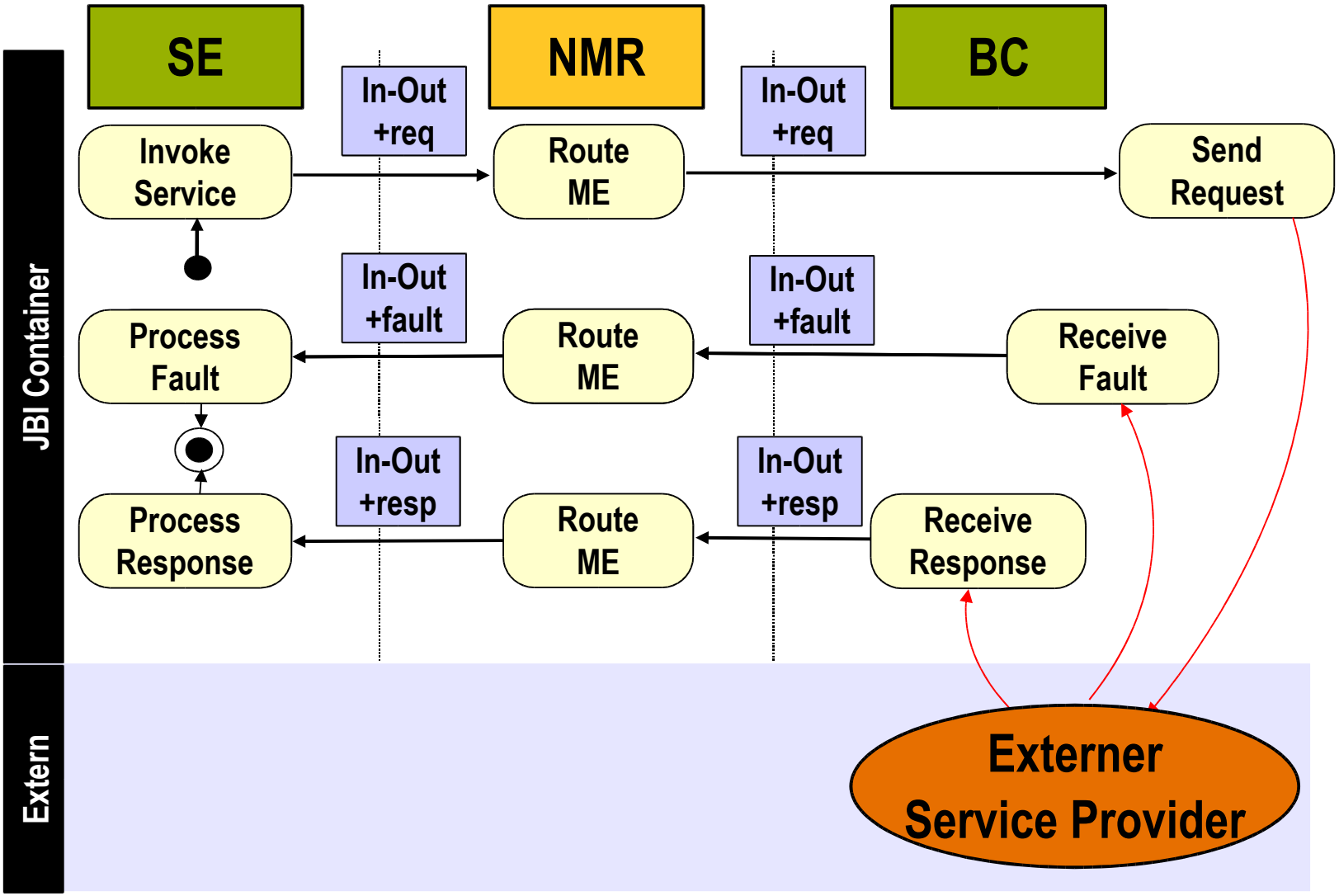
Beispiel: Umwandlung SOAP → NM



Message Exchange Patterns (MEP)

- Basieren auf den WSDL 2.0 MEPs
- MEPs werden aus der Provider-Sicht definiert
  - z.B. In-Out, In-Only, Robust In-Only
- Beschreiben **Reihenfolge** und **Kardinalität** für den Messageaustausch zwischen zwei Knoten (“Nodes”)
- JBI spezifiziert zwei Arten von Nodes
  - Service Provider
  - Service Consumer
- MEPs werden in JBI direkt durch Java-Klassen/Objekte abgebildet

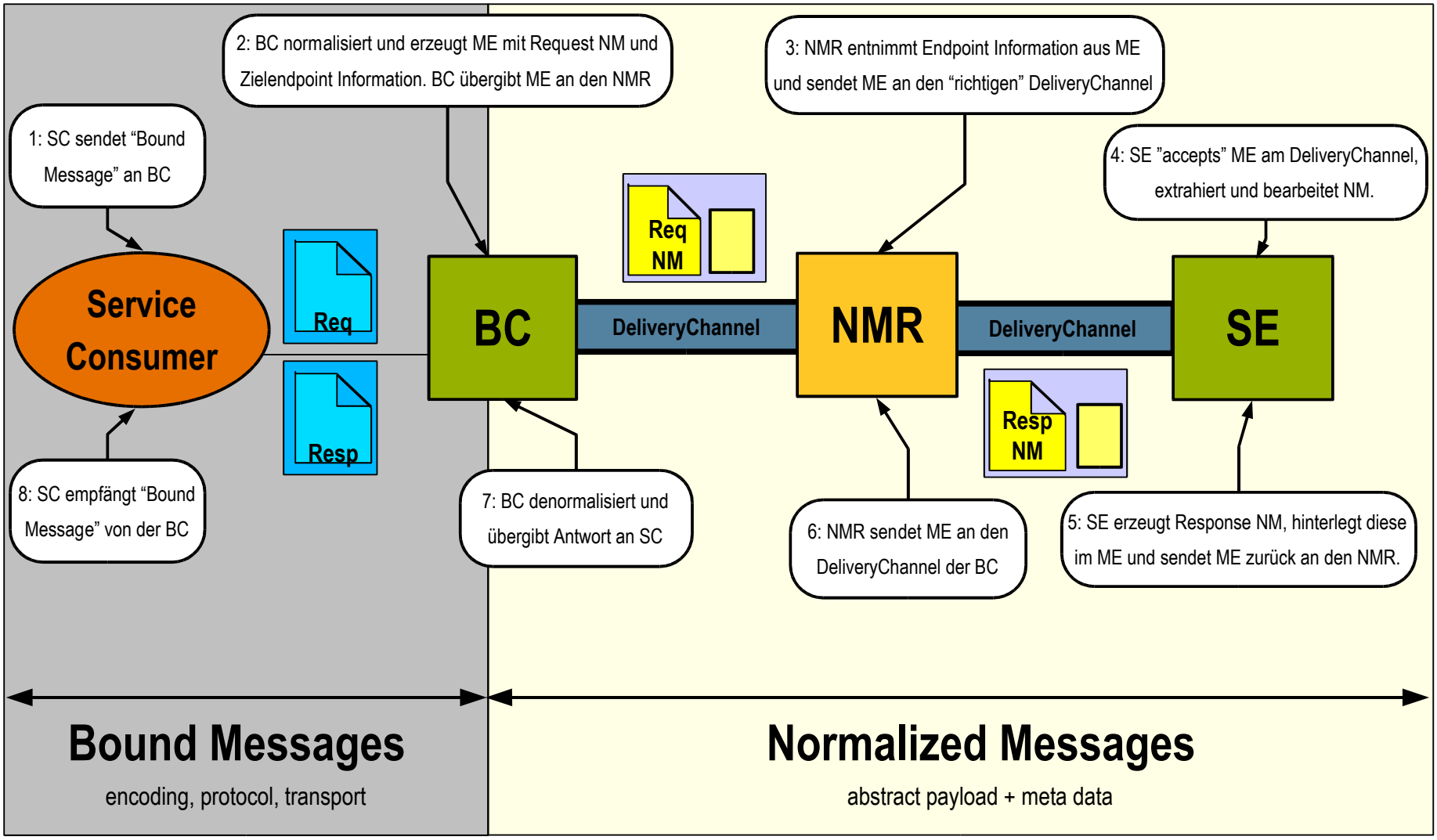
Beispiel: In-Out Exchange



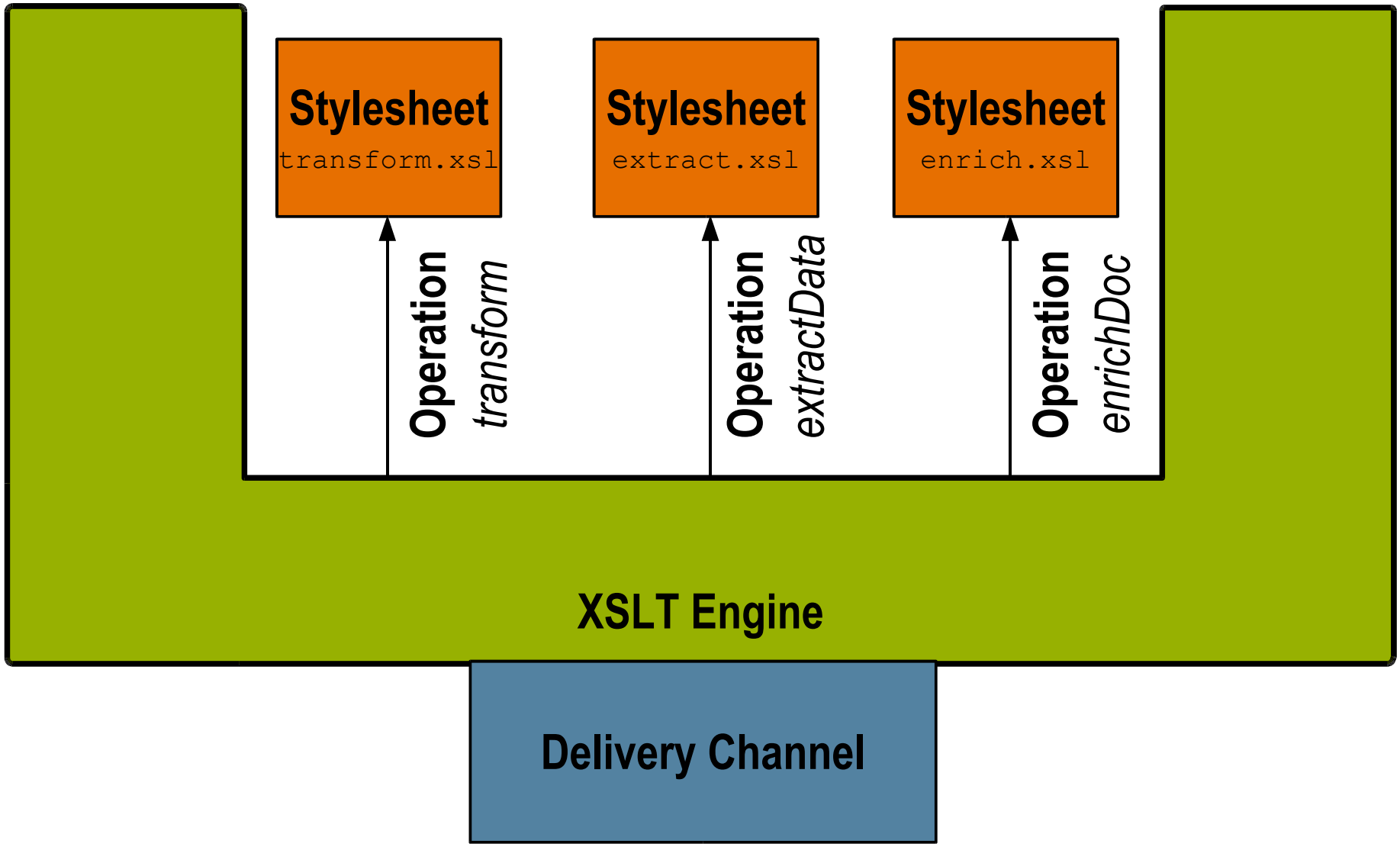
Normalized Message Router

- Sorgt für Interoperabilität zwischen Komponenten
  - Routing zu den “richtigen” Endpoints
  - Mediation der Message Exchanges
  - Austausch Normalisierter Messages
    - Payload
    - Metadaten
  - Unterstützt WSDL 2.0 Message Exchange Patterns
- Stellt horizontale Dienste für die Komponenten bereit
  - APIs
  - SPIs
  - Extension Point

Beispiel: Normalized Message Router



Komponenten als Container



Komponenten Management&Adminstration

- Portable Installation
  - > Standard Descriptor
  - > Standard Packaging
  - > JMX-basierte Installation
- Erweiterbarer LifeCycle
  - > JMX-basiertes Starten/Stoppen
  - > Komponentenspezifische Erweiterungen via JMX extensionMBean

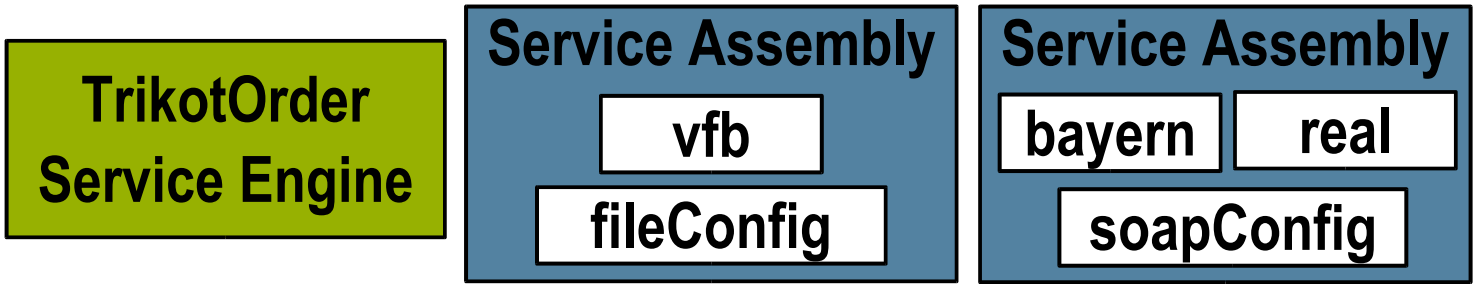
Ein standardisiertes Installationspackage
Ein standardisiertes Administrationsinterface

Demo: FanStore

- Die Firma “FanStore” verkauft Fussballtrikots
  - Order-Processing durch Austausch von XML-Dateien über das Filesystem
    - send “playerNumber”, “teamName” und “location”
    - receive “playerName”, “trikotColor”
  - Nur ein Team unterstützt (“VfB Stuttgart”)
- FanStore will expandieren
  - Mehr Mannschaften
  - Anbindung anderer Systeme über SOAP/HTTP Interface
- Vorgehensweise
  - Verwendung von JBI
  - “Imitation” des derzeitigen Systems
  - Erweiterung der Funktionalität

Was wird benötigt?

- Service Engine
  - Muss den benötigten Service bereitstellen
  - Muss “dynamisch” sein, um mehrere Teams zu unterstützen
  - Pro Team ein Deployment-Artefakt
- Bindings:
  - FileBinding & SOAPBinding
    - Beide in der Referenzimplementierung vorhanden
  - Für jedes Binding geeignete Deployment-Artefakte



TrikotOrder Service Engine

JBI Component Package (`trikotorder.jar`)

Component Descriptor (`jbi.xml`)

```
<jbi version="1.0" xmlns="http://java.sun.com/xml/ns/jbi" xmlns:xsi="http://www.w3.org/2001/XML...
    <component type="service-engine">
        <identification>
            <name>TrikotOrderServiceEngine</name>
            <description>This is a sample engine for the JavaForum in Stuttgart.</description>
        </identification>
        <component-class-name>com.fanstore.trikotorder.engine.JOComponent</component-class-name>
        <component-class-path>
            <path-element>trikotorder_impl.jar</path-element>
        </component-class-path>
        <bootstrap-class-name>com.fanstore.trikotorder.engine.JOBootstrap</bootstrap-class-name>
        <bootstrap-class-path>
            <path-element>trikotorder_impl.jar</path-element>
        </bootstrap-class-path>
    </component>
</jbi>
```

Implementation (`trikotorder_impl.jar`)

```
com.fanstore.trikotorder.engine.JOComponent
com.fanstore.trikotorder.engine.JOBootstrap

+ all the component's other implementation classes
```

Service Units (SU)

- Service Units sind “opaque” für den JBI Container
  - Keinerlei Annahmen oder Einschränkungen
  - Portabel
- Für “dynamische” Komponenten
  - Service Engines, z.B.
    - Stylesheets (XSLT Engine)
    - BPEL Process Definitionen (BPEL Engine)
    - Java-Klassen (Java/J2EE Engine)
  - Binding Components, z.B.
    - Host/Port/URL (SocketBinding, SOAPBinding)
    - Queue/Topic (JMSBinding)
    - Verzeichnisinformationen (FileBinding)
- JBI Umgebung sorgt für Persistenz von Service Units

VfB Service Unit

Service Unit Package (`vfb_su.zip`)

Opaque Service Description (`servicelist.xml`)

```
<servicelist xmlns="http://fanstore.com" xmlns:xsi="http://www.w3.org/2001/XMLSchem...
    <attributes>
        <service>
            <namespace-uri>http://fanstore.com</namespace-uri>
            <local-part>TrikotOrderService</local-part>
        </service>
        <endpoint-name>VfBEndpoint</endpoint-name>
            <description>VfB Stuttgart Team and Players</description>
            <operation>
                <namespace-uri>http://fanstore.com</namespace-uri>
                <local-part>createTrikotOrder</local-part>
            </operation>
            <mep>http://www.w3.org/2004/08/wsdl/in-out<mep>
     </attributes>
</servicelist>
```

Other Opaque Artefacts (`jars, xslts, etc.`)

```
team.properties
players.properties

+ all the other stuff your component's implementation needs, e.g.:
     - xslt
     - jar files
     - config files
```

Service Assemblies (SA)

- Enthält
  - Descriptor
  - Eine oder mehrere Service Units
- Komposition mehrerer SUs
  - über verschiedene Service Engines
  - über verschiedene Binding Components
  - über verschiedene Service Provider und Consumer
- Deployment-Vorgang
  - JMX-basiert
  - Portabel

Service Assembly

Service Assembly Package (`trikotorder_sa.jar`)

Service Assembly Descriptor (`jbi.xml`)

```
<jbi version="1.0" xmlns="http://java.sun.com/xml/ns/jbi" xmlns:xsi="http://www.w3.org...
    <service-assembly>
        <identification>
            <alias>sa1</alias>
            <name>sa1</name>
            <description>Service Assembly for the TrikotOrder Service</description>
        </identification>
        <service-unit>
            <identification>
                <alias>VfB Stuttgart Team</alias>
                <name>vfb</name>
                <description>Service Unit with VfB Team and Players</description>
            </identification>
            <target>
                <artifacts-zip>vfb_su.zip</artifacts-zip>
                <component-name>TrikotOrderServiceEngine</component-name>
            </target>
        </service-unit>
        <service-unit>
            ...
    </service-assembly>
</jbi>
```

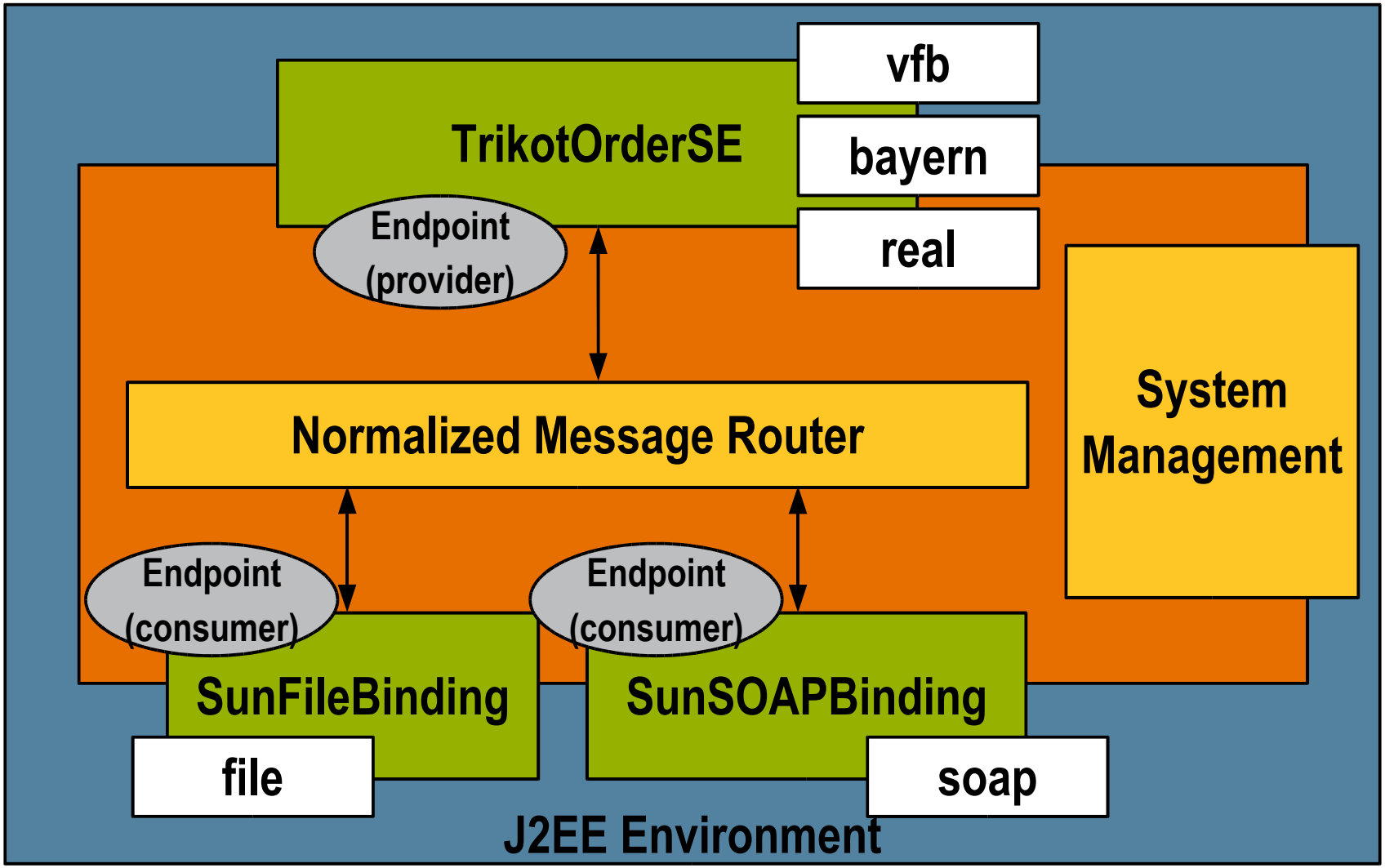
Service Unit (`vfb_su.zip`)

More Service Units (`file_su.zip`)

Demo

- Service Engine
  - Klassen erstellen und in Jar-File packen
    - `trikotorder_impl.jar`
  - Package erzeugen
    - `jbi.xml + trikotorder_impl.jar → trikotorder.jar`
  - Installieren
  - Starten
- Service Units & Service Assembly
  - Packages erzeugen
    - `vfb_su.zip, file_su.zip`
    - `jbi.xml + vfb_su.zip + file_su.zip → vfb_sa.jar`
  - Service Assembly deployen
- Testen, Administration und Management mit JMX
- Zweite SA mit SOAP und anderen Teams

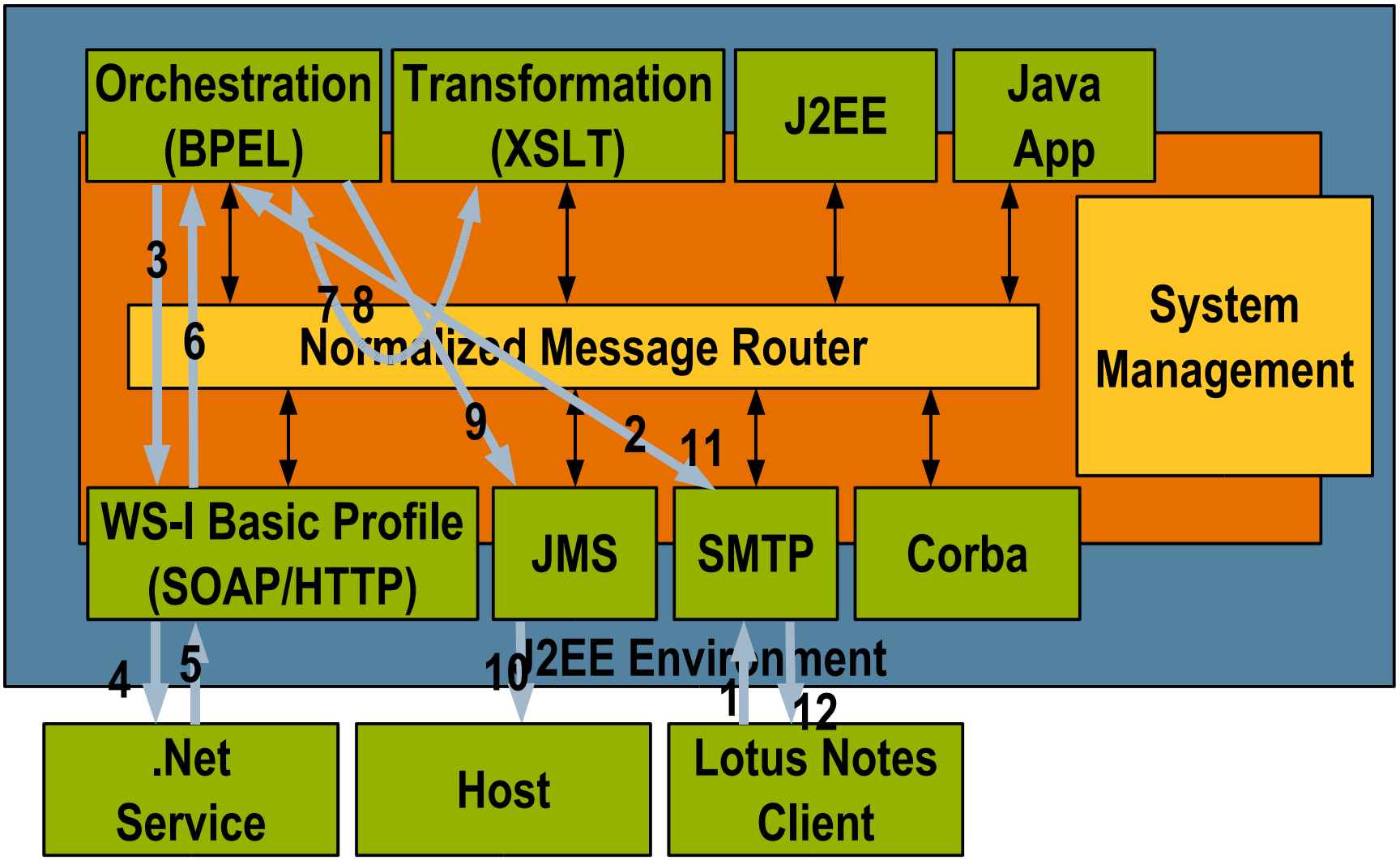
Demo


vfb
TrikotOrderSE
bayern
real
Endpoint
(provider)
System
Management
Normalized Message Router
Endpoint
(consumer)
Endpoint
(consumer)
SunFileBinding
SunSOAPBinding
file
soap
J2EE Environment

System Management

- 100% JMX
  - System LifeCyle
  - Component LifeCycle
    - erweiterbar
  - Installation von Komponenten
  - Deployment von Artefakten
  - Monitoring
- Viele Admin-Oberflächen möglich
  - Apache Ant, Kommandozeile, Web GUI
  - Integrierbar
    - z.B. HP OpenView, IBM Tivoli, AppServer Admin Console

Beispiel-Szenario



Was ist nicht im JSR-208 spezifiziert?

- Zugriff auf J2EE Resourcen
- Semantische Kopplung unterschiedlicer JBI Container
- Verteilung eine JBI Umgebung
- Externe Registry
- Kein “restriktives” Programmiermodell
- J2ME

Zusammenfassung

- Umsetzung bestehender Industrie-Standards
- Infrastruktur für SOAs
- Vendor-Ökosystem statt Vendor-Lockin
  - Spezialisierung der Hersteller möglich
- Technische Highlights
  - Komponentenarchitektur mit standardisierten Mechanismen
  - J2SE und J2EE unterstützt
  - WSDL 1.1 & 2.0 basierend
  - WS-I Basic Profile Unterstützung
  - JMX

Weitere Informationen

- Literatur
  - "*Enterprise SOA: Service-Oriented Architecture Best Practices*", Dirk Krafzig, Karl Banke, Dirk Slama
  - "*Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*", Gregor Hohpe, Bobby Woolfe
  - "*Patterns of Enterprise Application Architecture*", Martin Fowler
  - "*Enterprise Service Bus*", David Chappell
- Links
  - JSR-208: http://jcp.org/en/jsr/detail?id=208
  - JBI 1.0 SDK: http://java.sun.com/integration/download
  - "Developing a Service Engine": http://java.sun.com/integration/reference/techart/jbi/
  - WSDL 2.0: http://www.w3c.org/TR/wsdl20/

Vielen Dank!

Q&A

Armin Wallrab
armin.wallrab@sun.com