

**JB1 1.0**  
**Java Business Integration**  
**JSR 208**

**Christoph Hartmann**



---

**“There is no simple answer for  
enterprise integration”**

**Enterprise Integration Patterns**

# Agenda

---

## ➤ Introduction

### ■ Enterprise Service Bus

## ➤ JBI

### ■ Architecture

### ■ Assemblies & Components

### ■ Running Example

### ■ Component Life Cycle

### ■ Administration

### ■ Quality of Services

## ➤ Summary

## ➤ Bibliography

# Introduction

---

## ➤ JBI is

- An open and flexible way to integrate applications
- Specifies:
  - Plug-in components
  - Message Routing
  - Component life cycle
  - Component administration
  - Component packaging
- Is based upon WSDL 2.0

## ➤ Why is JBI important?

- Try to solve syntactically integration problems
- Try to avoid problems which are known from classic Enterprise Application Integration (EAI) frameworks

## ➤ How does it work?

- ...

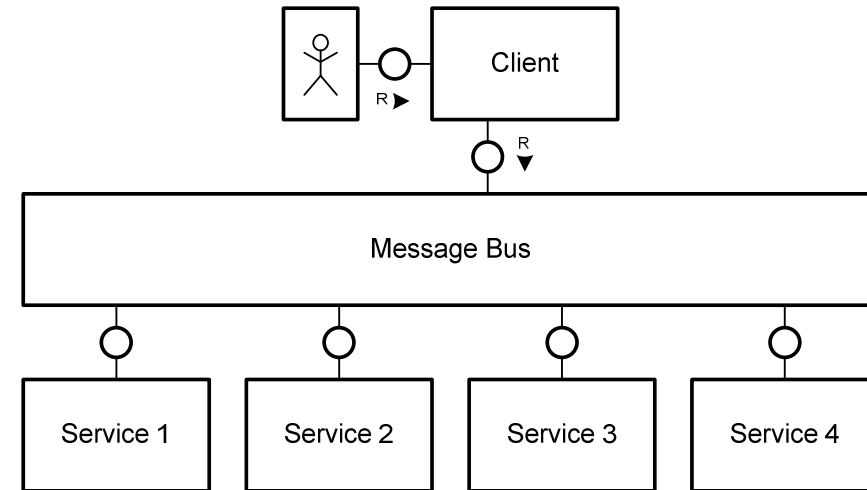
# Introduction: Enterprise Service Bus

---

➤ “An Enterprise Service Bus providing a Service-Oriented Architecture approach to building composite applications” *open esb*

➤ ESB combine

- Messaging
- Data transformation
- Reliable routing
- Web Services

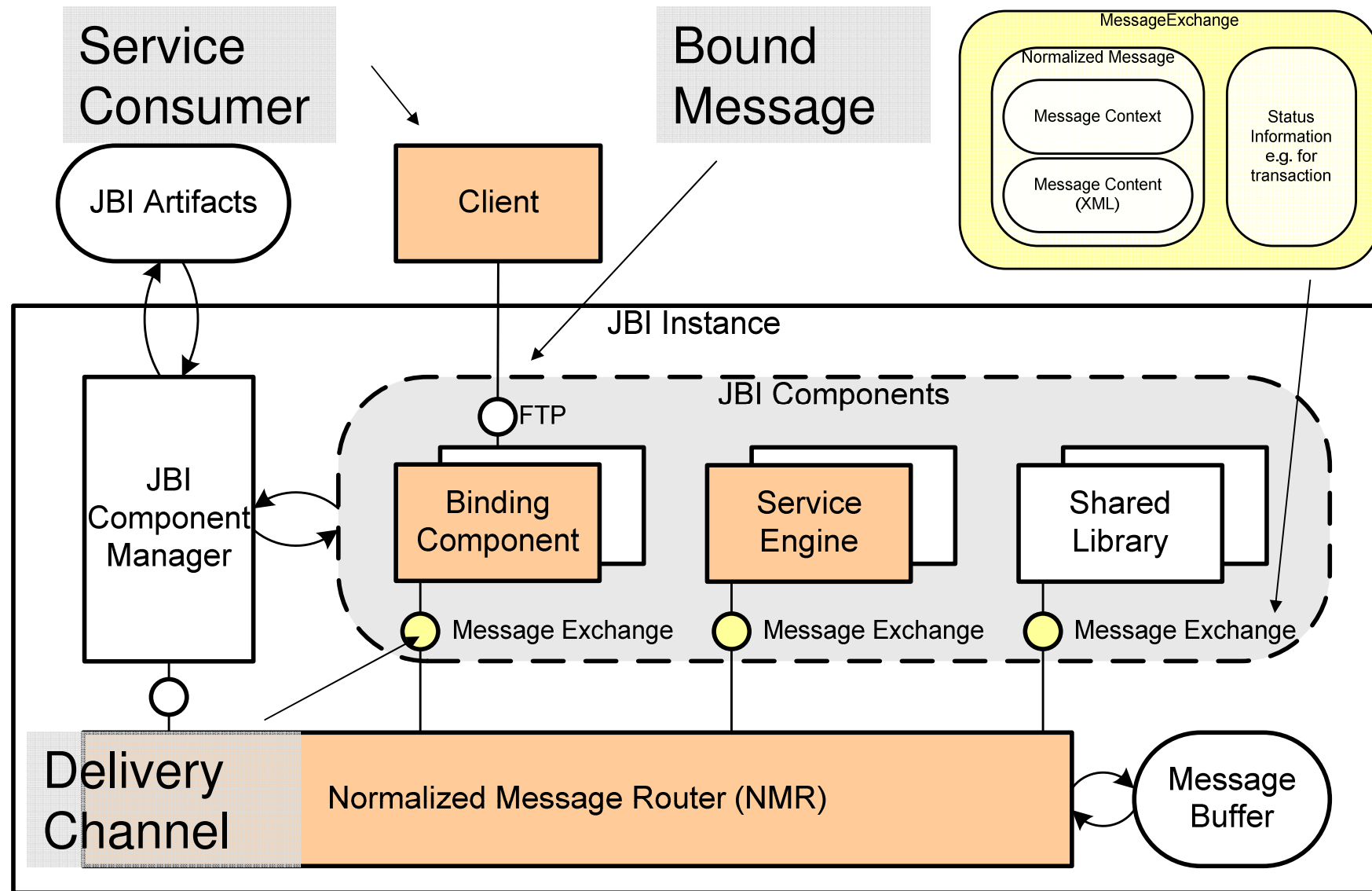


➤ ESB provides an implementation infrastructure for SOA based applications

➤ Standard based integration platform

- Often called “integration network”

# JBI: Architecture



# JBI: Assemblies & Components

## ➔ Binding Components (BC)

- Provide protocols and transport (e.g. JMS binding, SOAP binding)
- Proxy for service consumers

## ➔ Service Engines (SE)

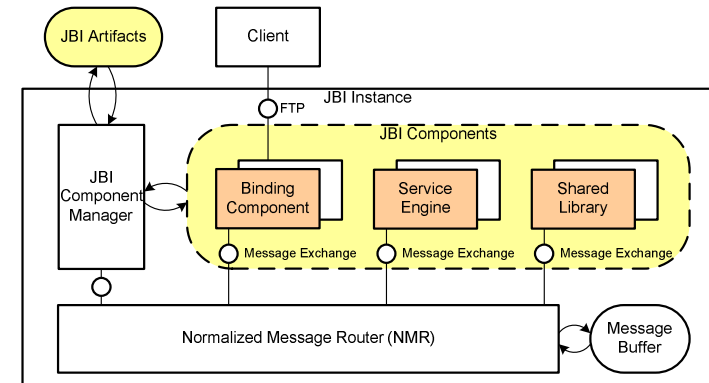
- Provide services (e.g. BPEL runtime)
- Includes business logic

## ➔ Shared Libraries

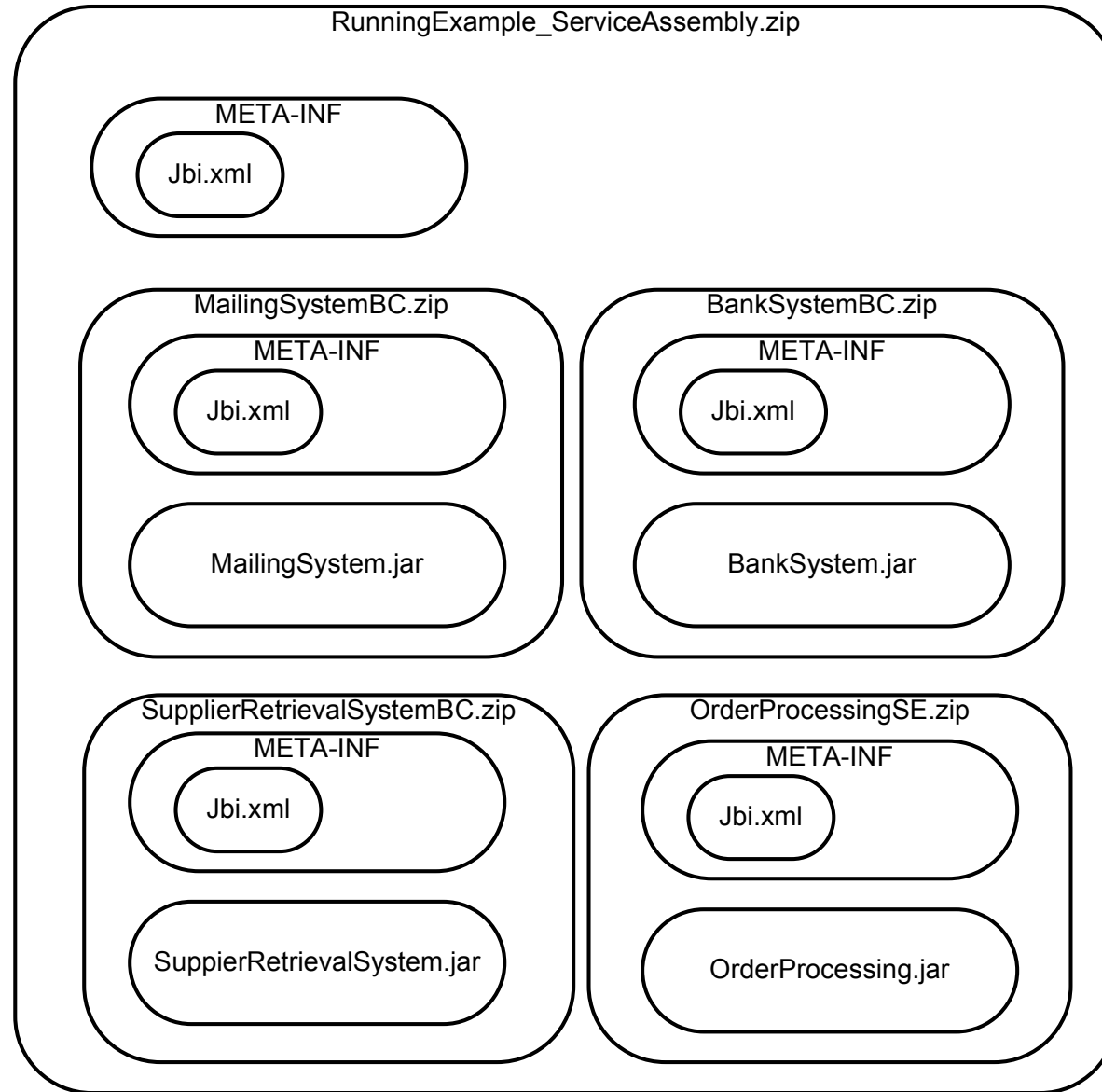
- Common logic which is used in different service engines or binding components

## ➔ Service Assembly (SA)

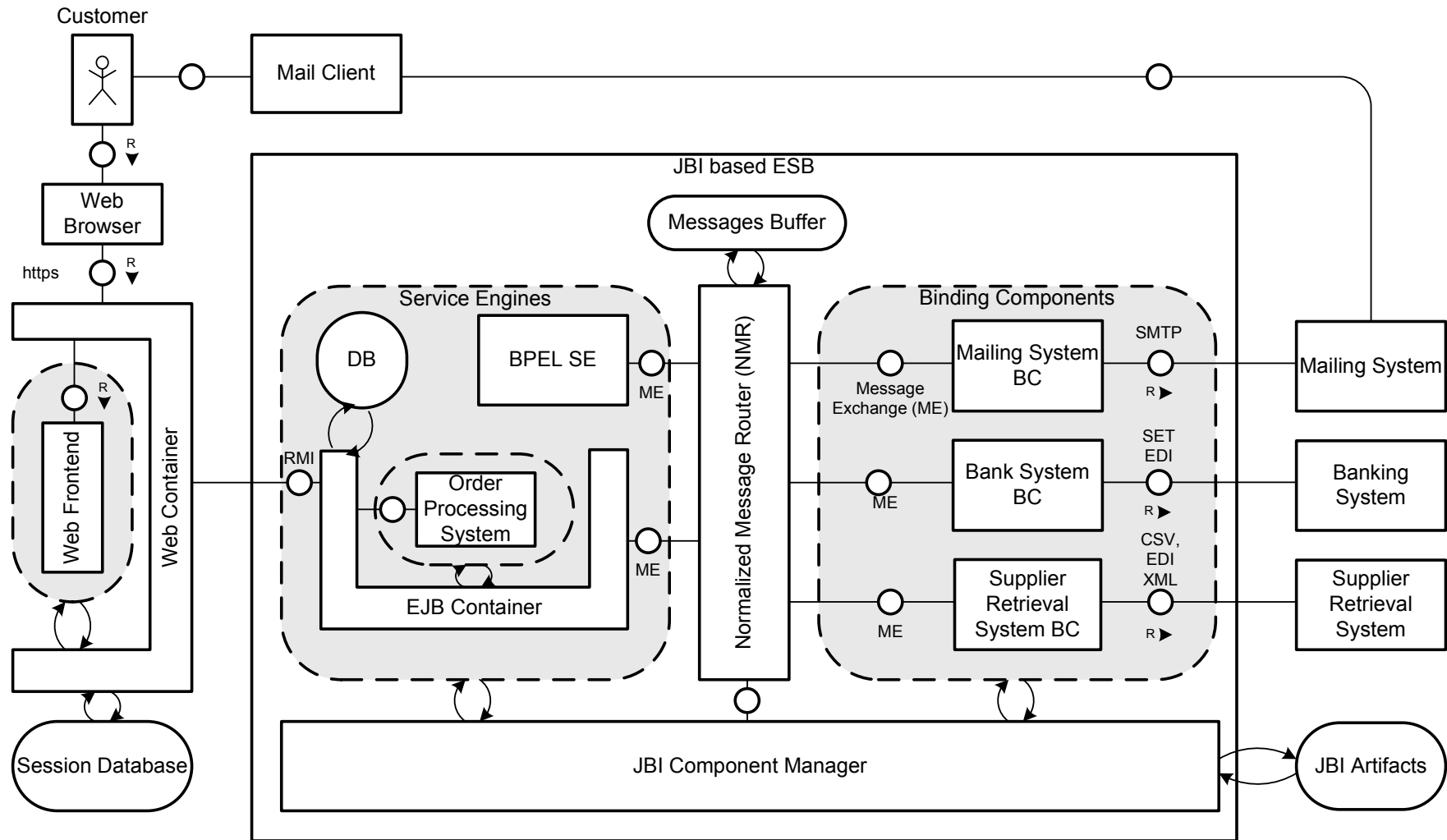
- Contains Binding Components, Service Engines and Shared Libraries of a SOA application



# JBI: Artifacts



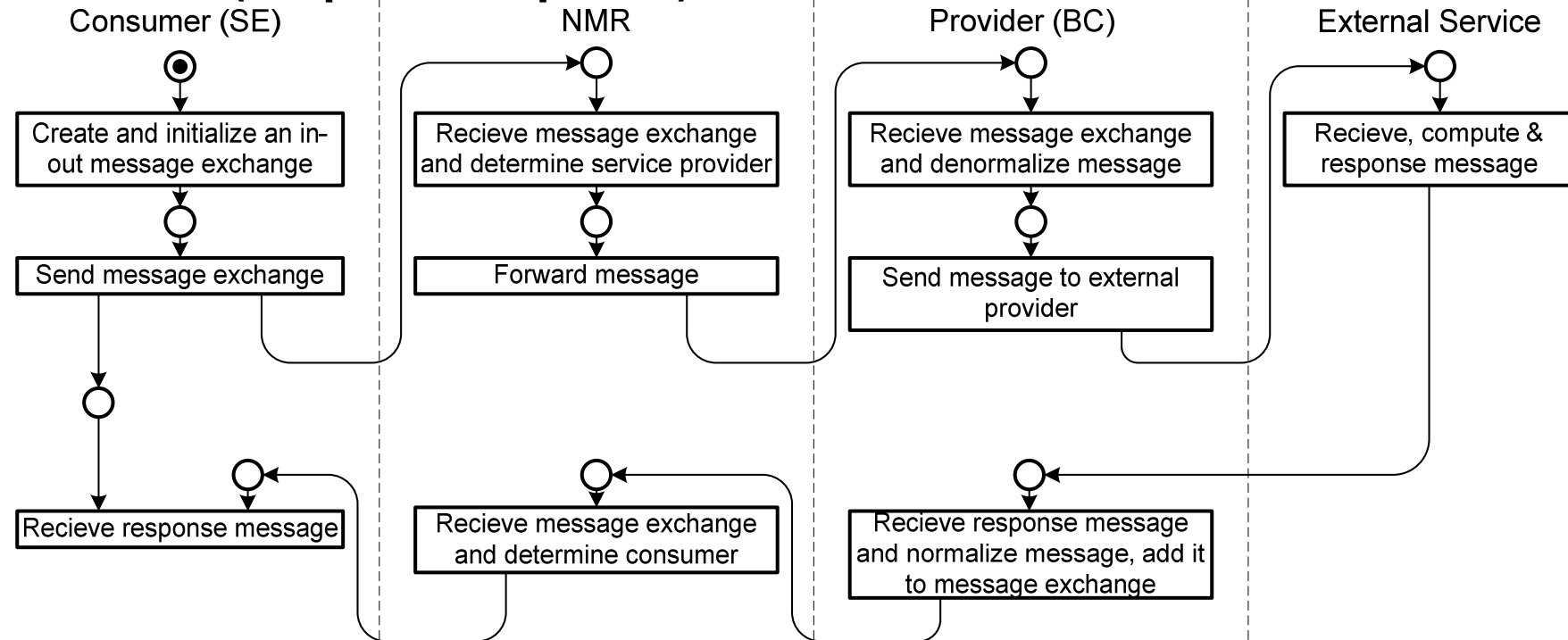
# JBI: Running Example



# JBI: Message Exchange Patterns

➤ Based upon WSDL 2.0 service invocations

➤ In-Out (Request-Response)



➤ In-Only

➤ Other message patterns available:

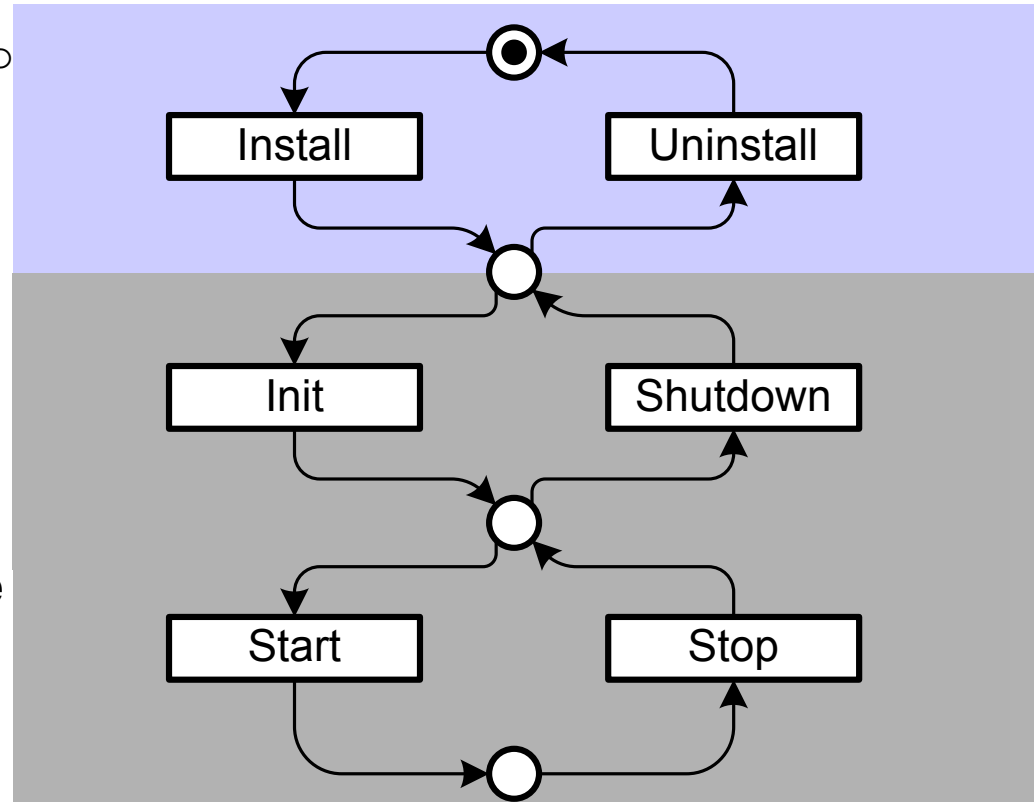
➤ Robust In-Only

➤ In-Optional Out

➤ can provide more patterns

# JBI: Component Life Cycle

- Implementation of `Bootstrap` contains methods for install and uninstall
- Methods for init, shutdown, start and stop contained in `ComponentLifeCycle`
- Deployment lifecycle is implemented via the interface `ServiceUnitManager`



# JBI: Administration

---

- use Java Management Extension (JMX)
  
- JBI specify administration of
  - Standard component installation
  - Manipulation of component lifecycle
  - specific component management functions
  
- JBI ManagementBeans are:
  - `InstallationServiceMBean`
  - `IntstallerMBean`
  - `DeploymentServiceMBean`
  - `LifeCycleMBean`
  - `ComponentLifeCycleMBean`

# JBI: Quality of Services

---

## ➤ Transactions

- JBI support local transactions within one JVM / JBI instance
- Distributed transactions are not supported yet

## ➤ Recoverability & Security

- Is not specified
  - E.g. access control lists, persistence of messages
- Have to be handled by the JBI components

# Summary

---

- **JB I is an open specification**
- **Reduce the number of interfaces between applications**
  - **ESB approach**
  
- **Current implementations and tools are available**
  - **Part of the new Java EE 5 Edition (OpenESB)**
  - **JB I support in Netbeans 5.5**
  - **Other JB I implementation (ServiceMix, Mule JB I, ...)**
  
- **Open points:**
  - **No distributed transactions**
  - **Does not cover „JB I distributed“ or Security**
  - **Solve no semantic problems**

# Bibliography

---

- **JB1 Specification**  
<http://www.jcp.org/en/jsr/detail?id=208>
- **Sun Service Orientated Business Integration**  
<http://java.sun.com/integration/>
- **Java Enterprise Edition 5**  
<http://java.sun.com/javaee/>
- **Hohpe, Gregor and Bobby Woolf, *Enterprise Integration Patterns*, Addison Wesley, 2004**
- **David A. Chappel, *Enterprise Service Bus. Theory in Practice.*, O'Reilly, 2004**

---

# Thank you!