# Enterprise Application Integration – The Cloud Perspective

**CONFERENCE PAPER** · JULY 2013

DOI: 10.1007/978-3-642-39200-9_50



**2 AUTHORS:**

Joerg Laessig

Hochschule Zittau/Görlitz

**78** PUBLICATIONS **159** CITATIONS

SEE PROFILE

Markus Ullrich

Hochschule Zittau/Görlitz

**5** PUBLICATIONS **3** CITATIONS

SEE PROFILE

# Structure

1. **Introduction: Group** / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Application Integration

## The Cloud Perspective

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

13th International Conference on Web Engineering
8-12 July 2013, Aalborg, North Denmark

# University of Applied Sciences Zittau/Görlitz

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**


# University of Applied Sciences Zittau/Görlitz

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

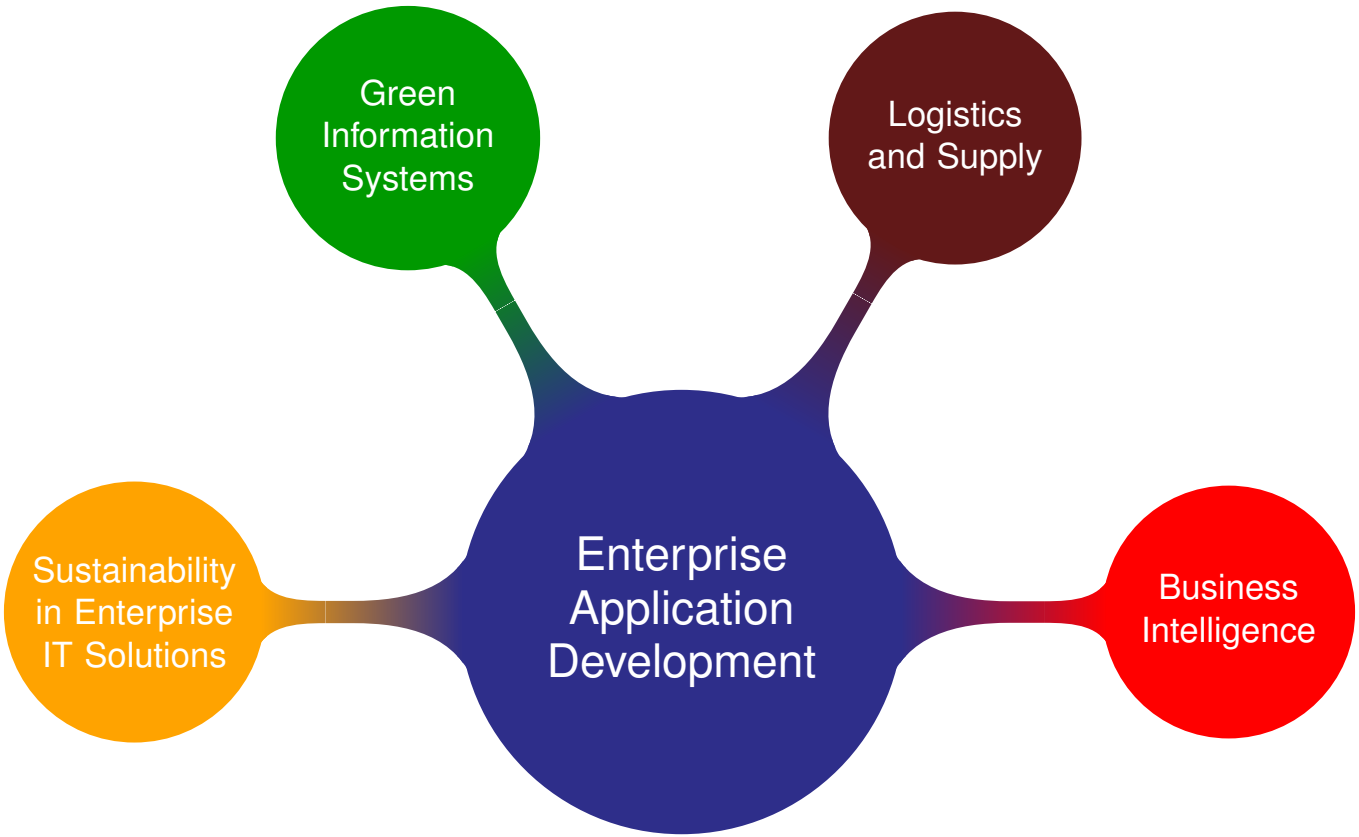
**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# EAD Group

# Enterprise Applications Development Group (EAD)

## Directions and Competences

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**




www.Enterprise-Application-Development.org/

# Springer Edited Volume Computational Sustainability

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz


- Book: Computational Sustainability, Springer Studies in Computational Intelligence
- Call: `http:\www.sita-research.org/call_ComputationalSustainability`
- Deadline Abstract: July 31, 2013
- Deadline Chapter: September 30, 2013
- Contact: joerg.laessig@sita-research.org

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Contact

# Contact

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

**Jörg Lässig, Markus Ullrich**
University of Applied Sciences Zittau/Görlitz
Department of Electrical Engineering and Computer Science

Obermarkt 17
D-02826 Görlitz
Germany

Tel.: +49 3581 7 92 53 54
Email: {jlaessig,mullrich}@hszg.de
Web: www.enterprise-application-development.org/

# Structure

1. Introduction: Group / **SOA+ESBs** / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Introduction

## Service Oriented Architectures (SOA)
## Enterprise Service Buses (ESBs)

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

# Topics

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# SW Quality connected with SW Architecture

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Development time qualities:

- maintainability
- extensibility
- evolvability
- reusability

## Run-time qualities:

- usability, configurability and supportability
- correctness, reliability, availability
- performance (throughput, response time, transit delay, latency, etc.)
- safety properties such as security and fault tolerance
- operational scalability

# SOA revisited

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## SOA seeks to eclipse previous efforts such as:

- modular programming
- code reuse
- object-oriented software development techniques

## SOA = set of:

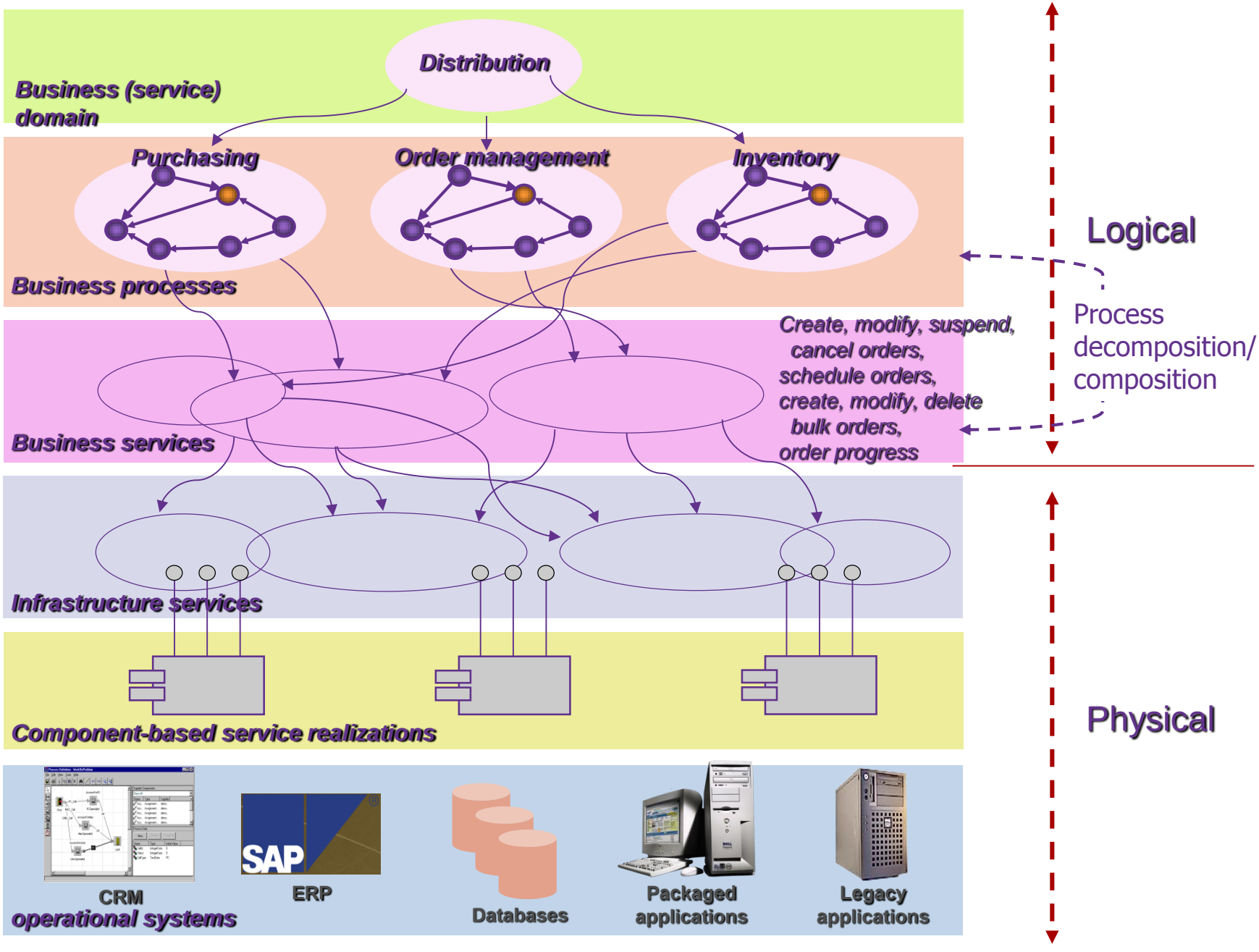
- guidelines, principles, techniques

## SOA is designed to overcome many complex implementation challenges:

- distributed software
- application integration
- multiple platforms and protocols
- numerous access devices

# SOA collaborating services

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz


Business (service) domain
Distribution
Purchasing
Order management
Inventory
Business processes
Create, modify, suspend, cancel orders, schedule orders, create, modify, delete bulk orders, order progress
Business services
Infrastructure services
Component-based service realizations
CRM
ERP
Databases
Packaged applications
Legacy applications
operational systems
Logical
Process decomposition/ composition
Physical

# What is an Enterprise Service Bus?

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- ESB is a standards-based backbone for connecting heterogeneous systems
- A new way of building and deploying enterprise SOAs
- Combines features from different types of middleware into one package
- Facilitates the deployment of Web services and solves integration problems

# Enterprise Service Bus - Key Features

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Key features:

- Web Services: SOAP, WSDL
- Messaging: asynchronous store-and-forward delivery
- Transformation
- Routing: content-based, publish/subscribe

## Platform neutral:

- Connects to anything in the enterprise: Java, .NET, legacy, etc.

# What are ESB products?

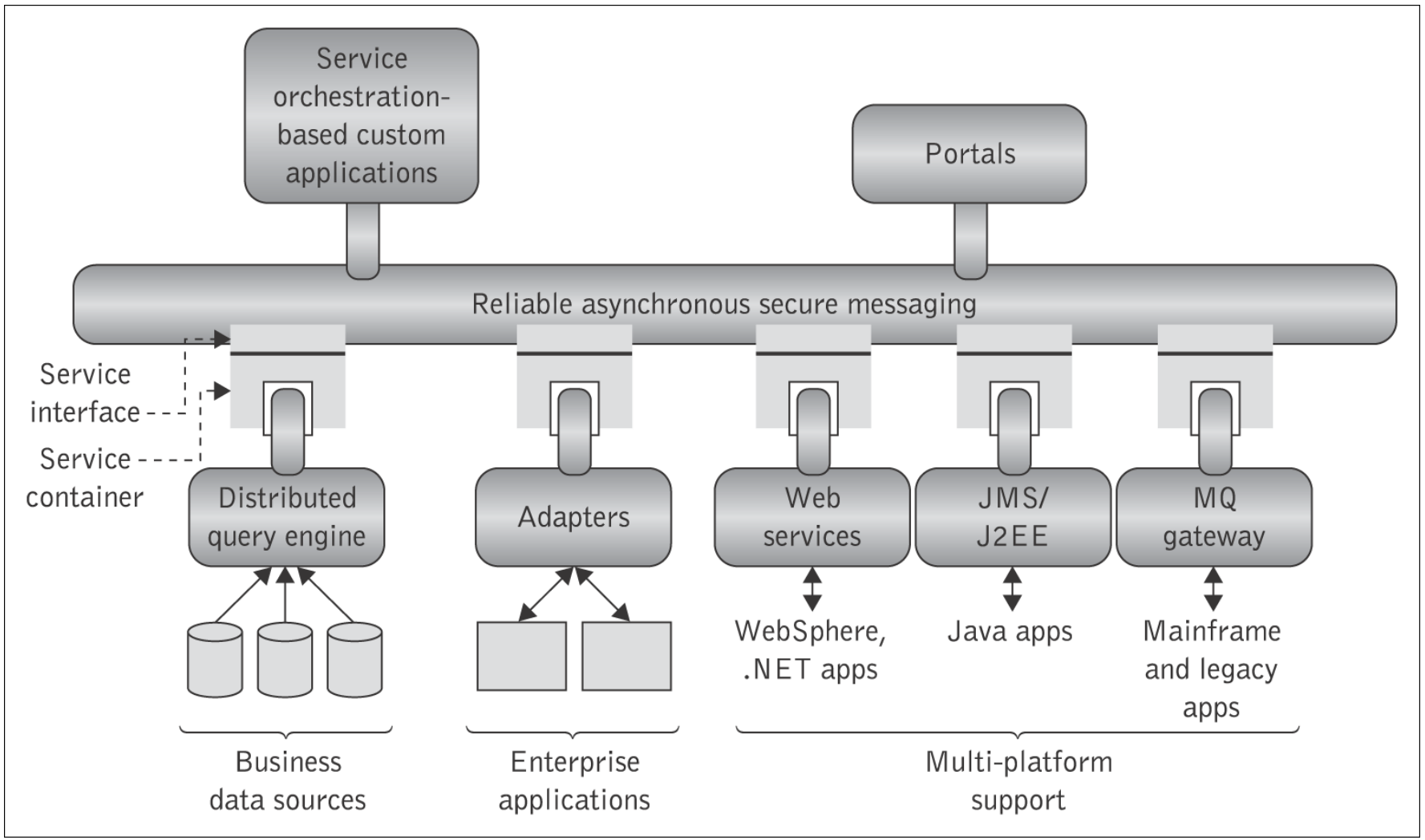
**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Sun Microsystems: Sun Java Composite Application Platform Suite/ Open ESB
- Open Source: FUSE ESB/ Talent ESB
- IBM: WebSphere ESB
- Microsoft: Biztalk Server
- Oracle: Oracle ESB
- Apache Software Foundation: Apache Service Mix
- SAP: NetWaver
- JBoss: JBoss ESB
- Software AG: WebMethods ESB-Platform

# Enterprise service bus

## connecting diverse applications and technologies:

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz



# Event-driven nature of SOA

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## SOA implementations provide actionable information to human users to interact with a business process:

- The ESB must recognize meaningful events and respond to them appropriately.
- The response could be either by automatically initiating new services and business processes or
- by notifying users of business events of interest and
- suggesting the best courses of action.

## In an ESB-enabled event-driven SOA, applications and services are treated as abstract service endpoints to readily respond to asynchronous events.

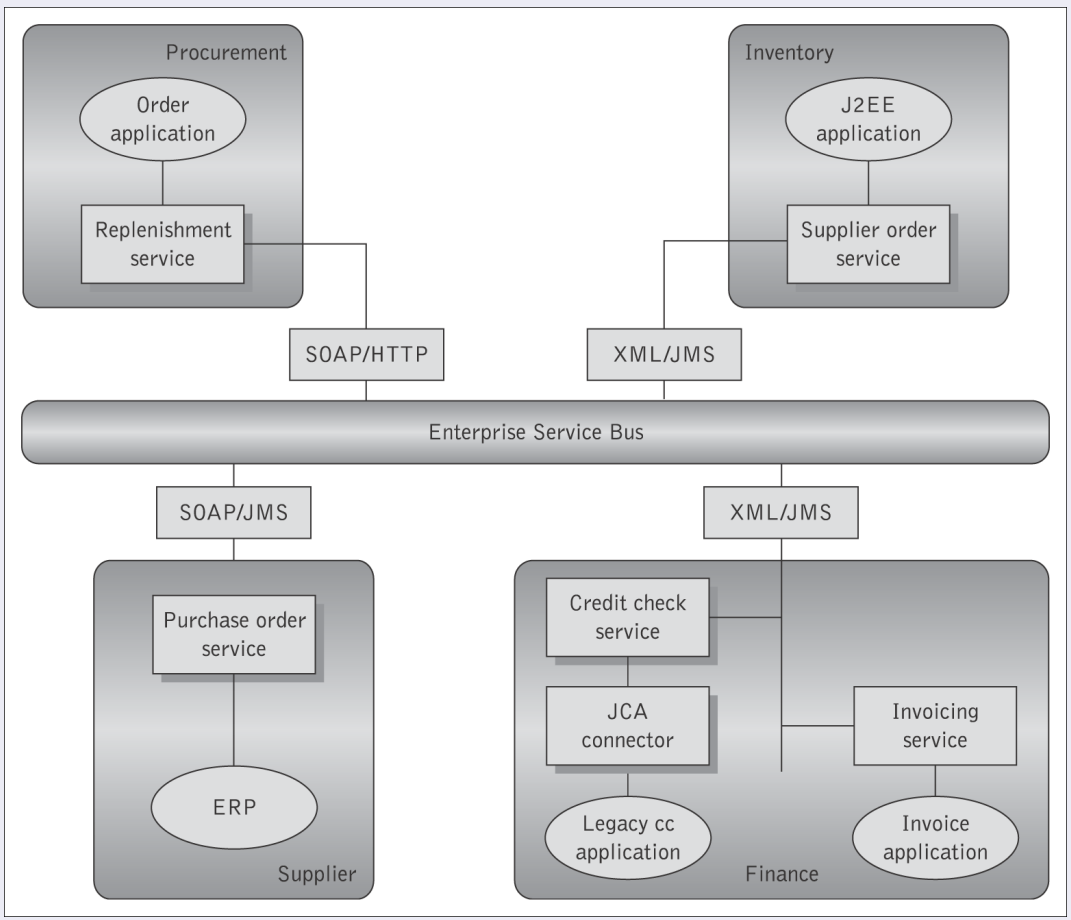
- An event-driven SOA requires that two participants in an event (server and client) are decoupled.

# Enterprise service bus
## connecting remote services:

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Data vs. Application Integration

The ESB infrastructure includes a distributed processing framework and XML-based Web services to orchestrate the behavior of services in a distributed process.



# ESB integration styles

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

ESBs employ a service-oriented integration solution that leverages amongst other things open standards, loose coupling, the dynamic description and discovery capabilities of Web services.

## A service-oriented integration solution deals with:

- Integration at the presentation-tier
- Application connectivity
- Application integration
- Business process integration
- Business data integration

# ESB elements:
## Business Process Management

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

BPM provides end-to-end visibility and control over all parts of a long-lived, multi-step information request or transaction process that spans multiple applications and human actors in one or more enterprises.

Monitors both the state of any single process instance & all process instances in a service aggregation, using real-time metrics that translate actual process activity into key performance indicators.

**BPM software solutions provide workflow-related business processes, process analysis, and visualization techniques in an ESB setting.**

- they allow the separation of business processes from the underlying integration code.
- they allow a process orchestration engine (BPEL) to be layered onto the ESB.

# Literature

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**


- Michael P. Papazoglou, Web Services: Principles and Technology, Pearson Prentice Hall, 2008
- Martin Kalin, Java Web Services - Up and Running, OŔeilly, 2009
- Will Iverson, Real World Web Services, OŔeilly, 2005
- Leonard Richardson, Sam Ruby, RESTful Web Services, OŔeilly, 2007
- Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, Web Services - Concepts, Architectures and Applications, Springer, 2010
- Oliver Heuser, Andreas Holubek, Java Web Services in der Praxis, dpunkt.Verlag, 2010
- Eben Hewitt, Java SOA Cookbook, OŔeilly, 2009
- Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, John Domingue, Enabeling Semantic Web Services, Springer, 2007
- Steve Graham, Doug Davis, Simeon Simeonov, Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle, Dieter König, Claudia Zentner, Building Web Services with Java, Sams Publishing, 2005

# Structure

1. Introduction: Group / SOA+ESBs / **Enterprise Application Integration**
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Application Integration

## Integration Methods

**Jörg Lässig, Markus Ullrich**

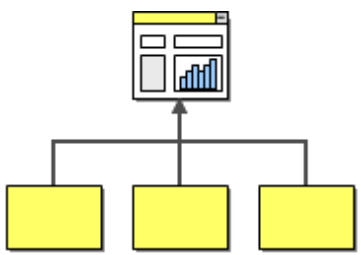
Department of Computer Science

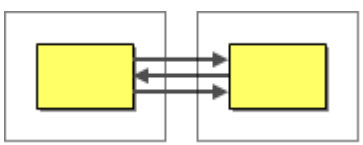
July 8, 2013

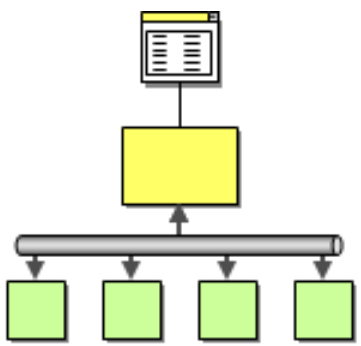
**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Why Integration?

- enterprises comprised of many different applications
  - custom build
  - acquired from a third party
  - part of a legacy system
- even ERP systems perform only a fraction of typical business functions
- each business function can be performed by most appropriate software







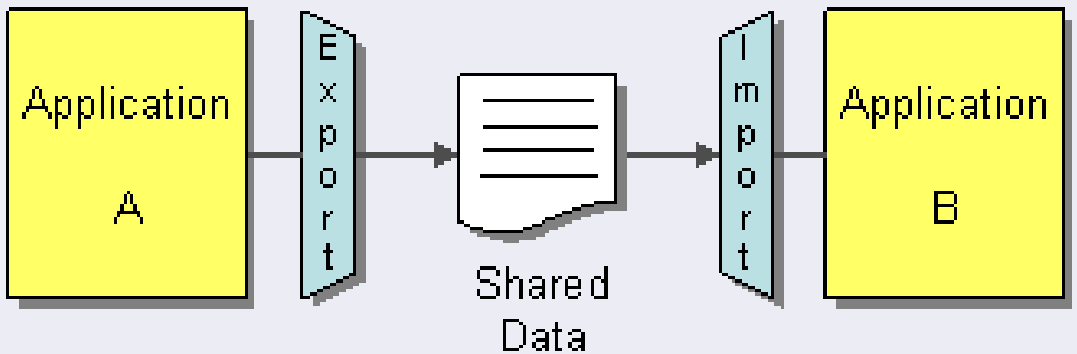
Images: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Integration Challenges

- communication between multiple computer systems, business units and IT departments
- incorporate most critical business function ⇒ proper functioning becomes vital to business
- limited amount of control over participating applications
- still not many established standards for integration
- differentiation between common presentation (e.g., XML) and common semantics
- operating and maintaining an integration solution
- unreliable and slow networks

# File Transfer




*"Have each application produce files that contain the information the other application must consume. Integrators take the responsibility of transforming files into different formats. Produce the files at regular intervals according to the nature of the business."* - Martin Fowler

Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## File Transfer: Advantages

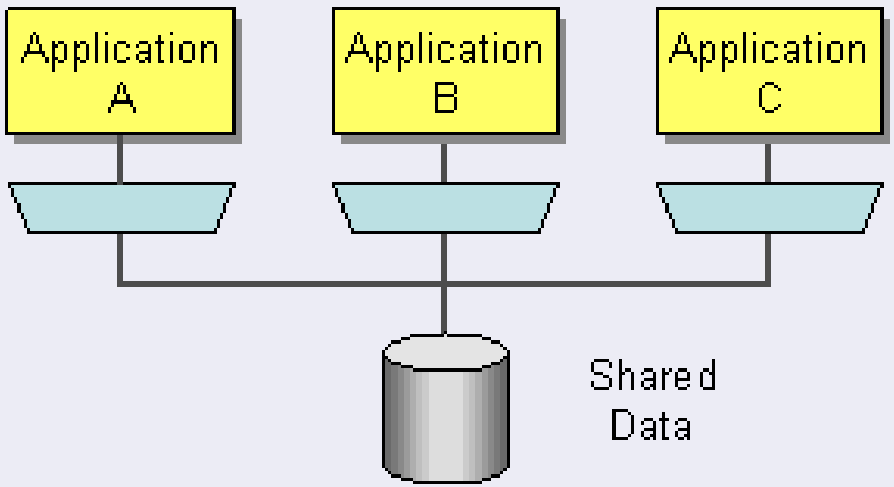
- decoupling applications is much easier
- format and contents are negotiated with the integrators
- applications need no knowledge about each other

## File Transfer: Disadvantages:

- effort to create and read files
- inconsistencies if updates are not frequent enough
- applications have to agree on naming conventions and directories

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Shared Database




*"Integrate applications by having them store their data in a single Shared Database and define the schema of the database to handle all the needs of the different applications."* - Martin Fowler

Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Shared Database: Advantages

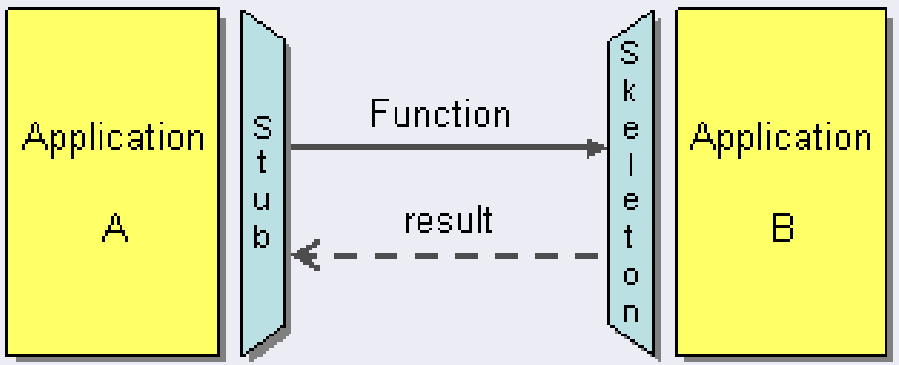
- no necessity of managing multiple file formats
- no inconsistencies as with file transfer
- applications need no knowledge about each other

## Shared Database: Disadvantages:

- creating a unified schema
- packaged applications may not work with a different schema
- possible performance bottleneck

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Remote Procedure Invocation




*"Develop each application as a large-scale object or component with encapsulated data. Provide an interface to allow other applications to interact with the running application."* - Martin Fowler

Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Remote Procedure Invocation: Advantages

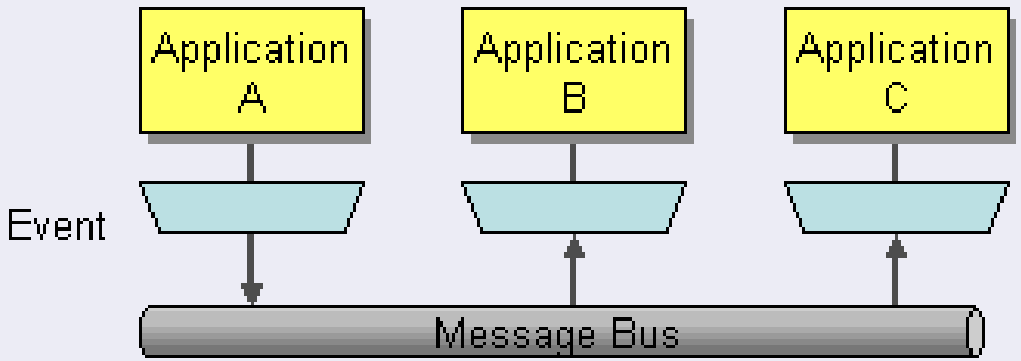
- integration of functionality rather than data
- each application can manage its data and formats on its own
- easy support for multiple points of view

## Remote Procedure Invocation: Disadvantages:

- huge differences in performance and reliability compared to desktop software
- most integrations designed as single application
- $\Rightarrow$ tight instead of loose coupling

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Messaging




*"Use Messaging to transfer packets of data frequently, immediately, reliably and asynchronously, using customizable formats."* - Gregor Hohpe and Bobby Woolf

Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Messaging: Advantages

- sender has not to wait on the receiver
- easy to separate integration decisions from application development
- less inconsistency problems compared to file transfer

## Messaging: Disadvantages:

- asynchronous thinking
- new issues to consider
  - data transfer
  - data direction
  - data formation
  - connect applications

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Messaging Vocabulary

- Message: packets of data, consist of header and body
- Channel: logical pathways - connect programs and convey messages
- Sender or Producer: message sending program, writes message to a channel
- Receiver or Consumer: message receiving programm, reads message from a channel

## What is a Messaging System?

- coordinates and manages the sending and receiving of messages
- paths of communication are defined by channels
- makes sure each data record is safely persisted

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Message Transmission Steps

1. Create: The sender creates the message and populates it with data.
2. Send: The sender adds the message to a channel.
3. Deliver: The messaging system moves the message from the sender's computer to the receiver's computer.
4. Receive: The receiver reads the message from the channel.
5. Process: The receiver extracts the data from the message.

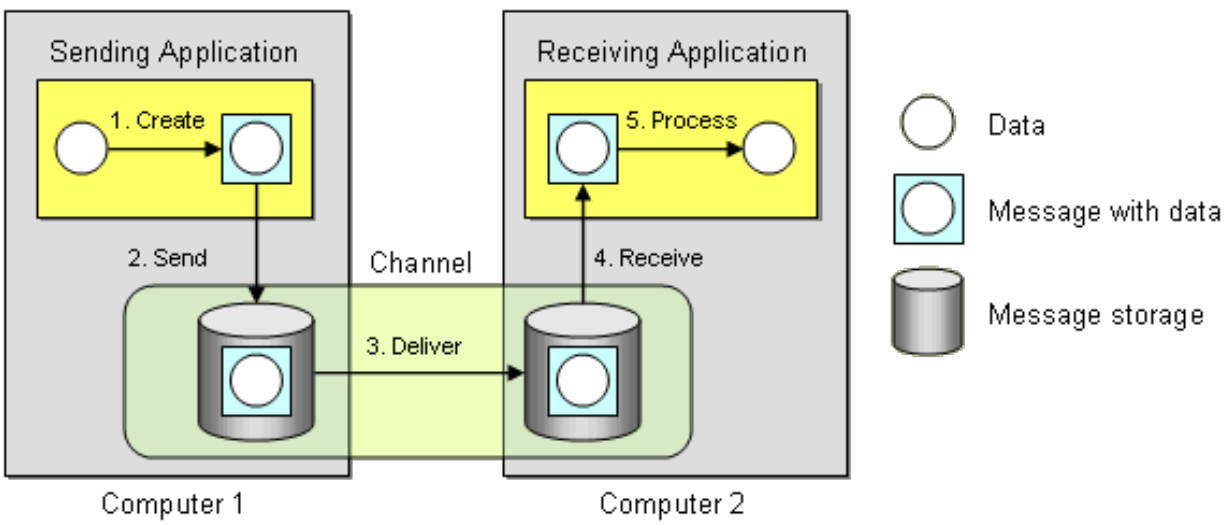



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Channels

- virtual pipe connecting sender and receiver
- different channels for different types of information
- important to decide all necessary channels in advance
- may be cheap when it comes to resource allocation but not for free

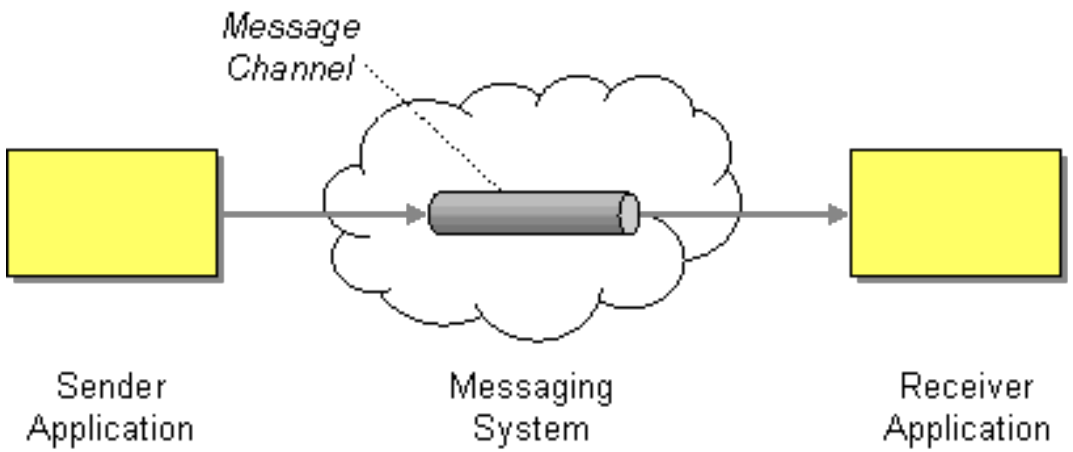



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Messages

- atomic packet of data
- transmitted over a channel repeatedly until success
- consists of two basic parts:
  - Header: information describing the transmitted data like origin and destination
  - Body: the transmitted data itself - generally ignored by the messaging system

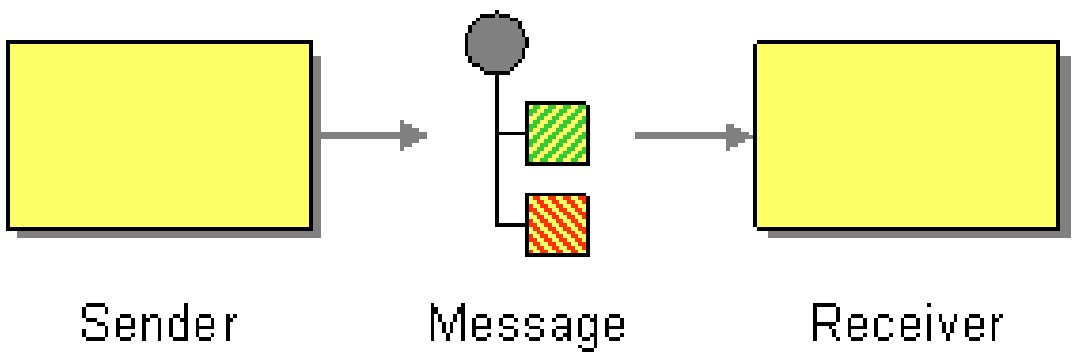



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Pipes and Filters

- architectural style describing how multiple processing steps are chained together
- complex tasks are divided into a sequence of smaller tasks
- independence and flexibility are maintained
- complex message processing still possible
- enables Pipeline Processing and improved Parallel Processing

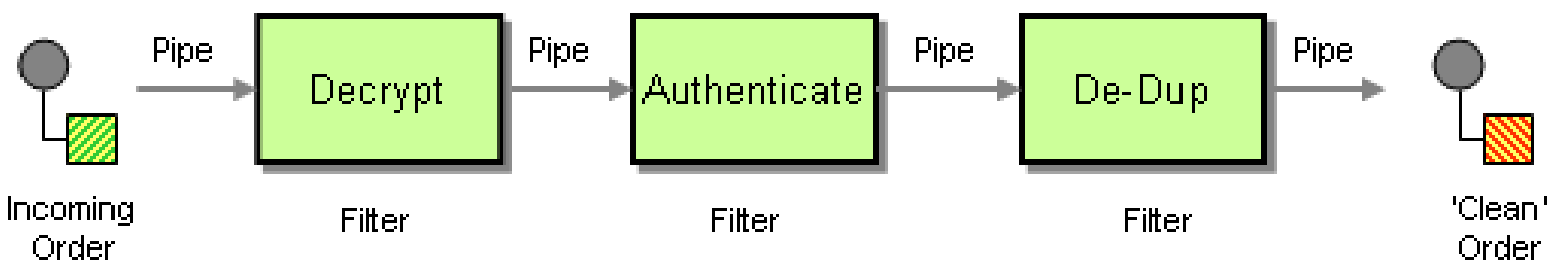



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Routing

- determines how to navigate the channel topology
- directs message to final receiver or next router
- surrounding components are unaware of its existence
- concerns itself only with the message direction (content remains unchanged)

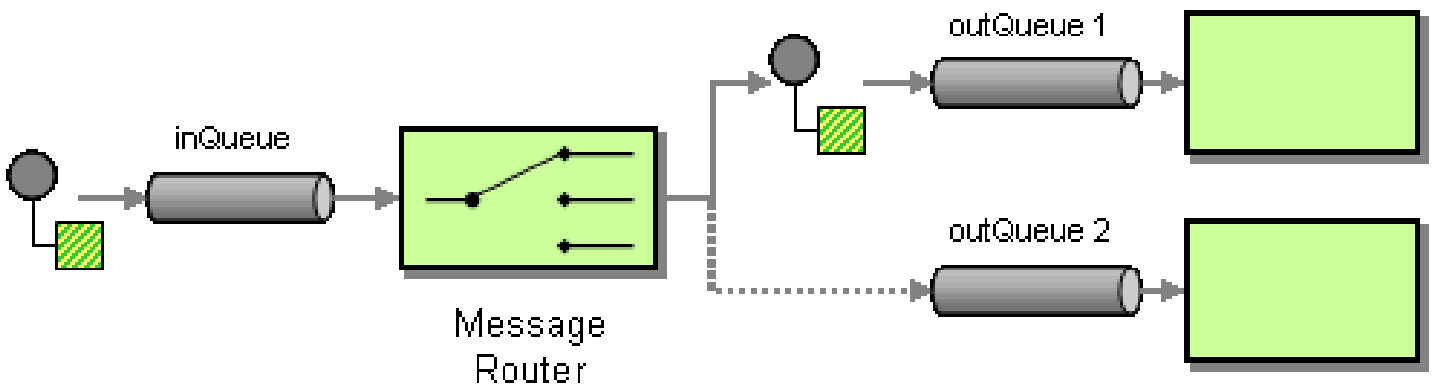



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Transformation

- translates one data format into another
- used on different levels:
  - Data Structures (Application Layer): f.e. condense many-to-many relationship into aggregation
  - Data Types: f.e. convert ZIP code from numeric to string
  - Data Representation: parse data representation and render in different format, de-/encrypt as necessary
  - Transport: move data across protocols without affecting message content

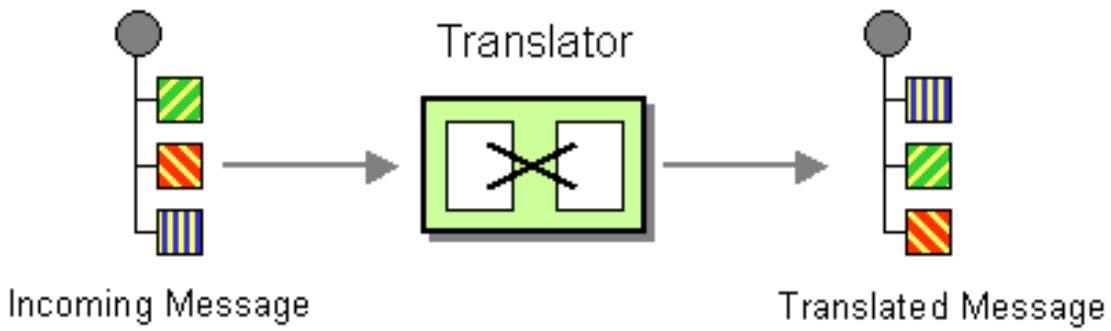



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Endpoints

- enable applications to send and receive messages
- capability to interface with messaging system not necessary
- layer of code that knows how application and the messaging system works
- encapsulates messaging system from application

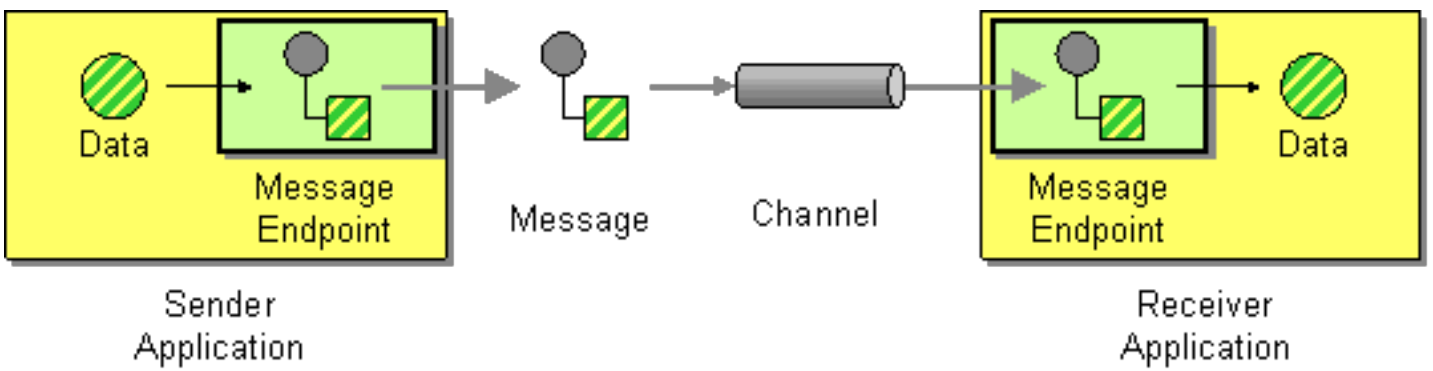



Image: Hohpe and Woolf

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. **Messaging: Message Channels**
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Message Channels

## Enterprise Integration Patterns

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 08, 2013

# Message Channels

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## How does one application communicate with another using messaging?

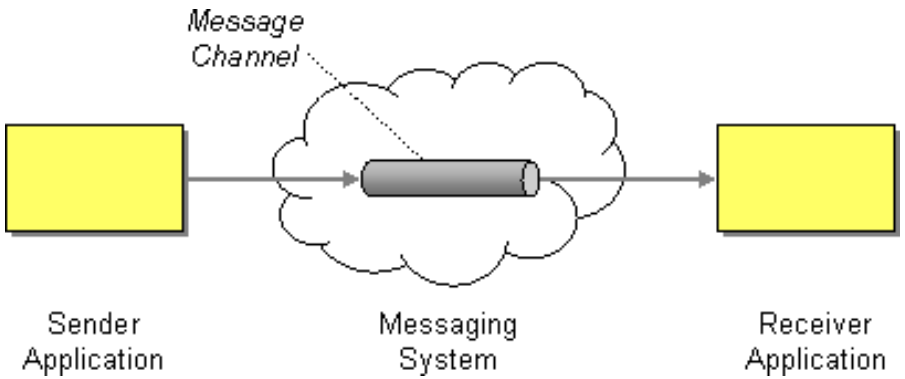



Image: Hohpe and Woolf

- Connect the applications using a Message Channel
- One application writes information to the channel
- Other one reads that information from the channel

# Message Channels

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Message Channel is unidirectional or bidirectional?

- Technically, it's neither; a channel is more like a bucket
- Some applications add data to and other applications take data from
- Because the data travel from one application to another, that gives the channel direction, making it unidirectional
- Question: What if a channel were bidirectional?
  - If a channel were bidirectional, that would mean that an application would both send messages to and receive messages from the same channel, which (while technically possible) makes little sense because the application would tend to keep consuming its own messages.

# Message Channels

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Message Channel is unidirectional or bidirectional?

- Technically, it's neither; a channel is more like a bucket
- Some applications add data to and other applications take data from
- Because the data travel from one application to another, that gives the channel direction, making it unidirectional
- Question: What if a channel were bidirectional?
  - If a channel were bidirectional, that would mean that an application would both send messages to and receive messages from the same channel, which (while technically possible) makes little sense because the application would tend to keep consuming its own messages.

# Message Channel Themes

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

- Fixed set of channels:
  - A developer has to know where to put what types of data
  - Likewise where to look for what types of data coming
  - Channels need to be agreed upon at design time
- Mostly, channels must be statically defined, but there are exceptions
  - For example Request-Reply in which the requestor can create or obtain a new channel the replier knows nothing about, specify it as the Return Address of a request message, and then the replier can make use of it.
- How to determine set of channels?
  - Depends on different factors, e.g., application's specific needs and type of communication.

# Point-to-Point Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the caller be sure that exactly one receiver will receive the document or perform the call?**

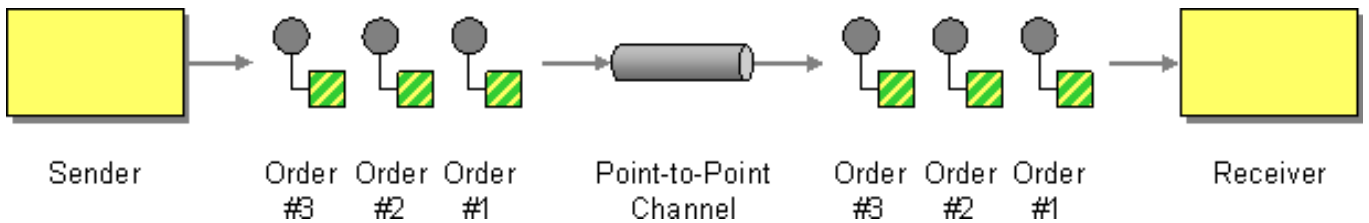



Image: Hohpe and Woolf

- Send the message on a Point-to-Point Channel, which ensures that only one receiver will receive a particular message.
- Example: MessageQueue in .Net, QueueSender in Java

# Point-to-Point Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the caller be sure that exactly one receiver will receive the document or perform the call?**

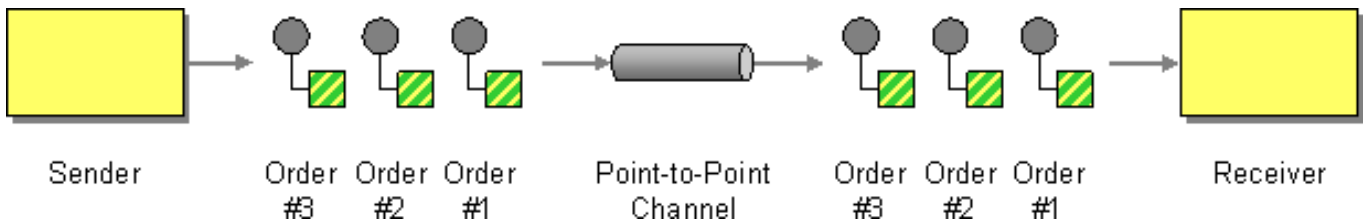



Image: Hohpe and Woolf

- Send the message on a Point-to-Point Channel, which ensures that only one receiver will receive a particular message.
- Example: MessageQueue in .Net, QueueSender in Java

# Publish-Subscribe Channel

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## How can the sender broadcast an event to all interested receivers?

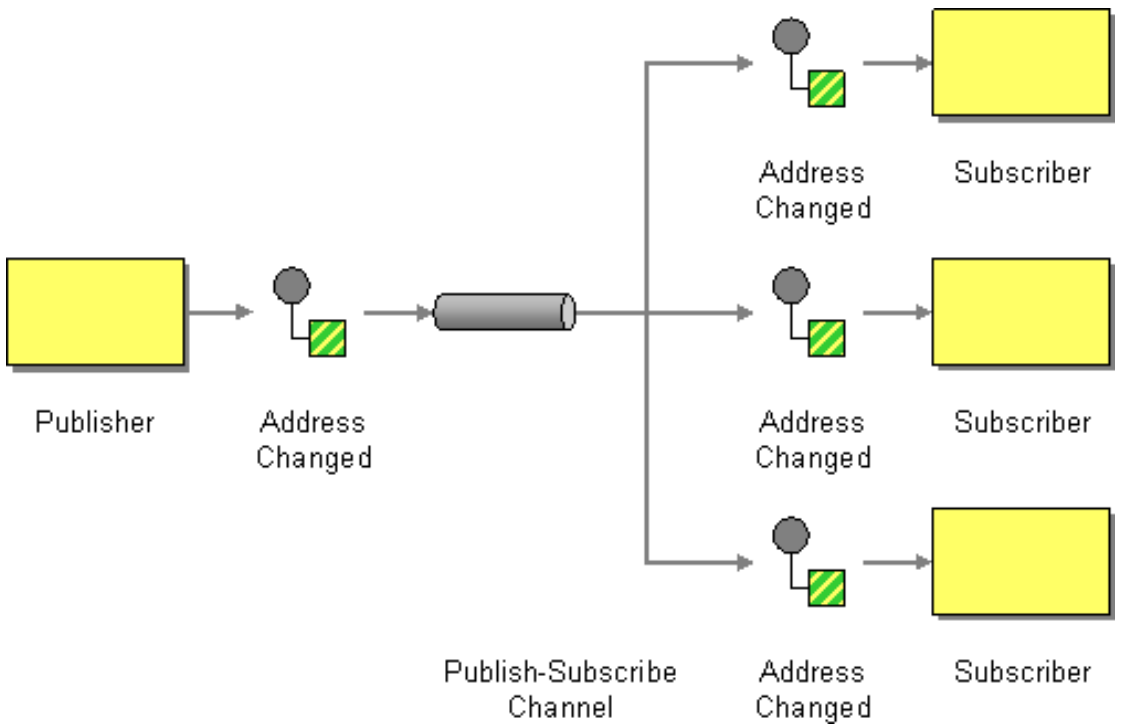



Image: Hohpe and Woolf

- Send the event on a Publish-Subscribe Channel, which delivers a copy of a particular event to each receiver.

# Publish-Subscribe Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## How can the sender broadcast an event to all interested receivers?

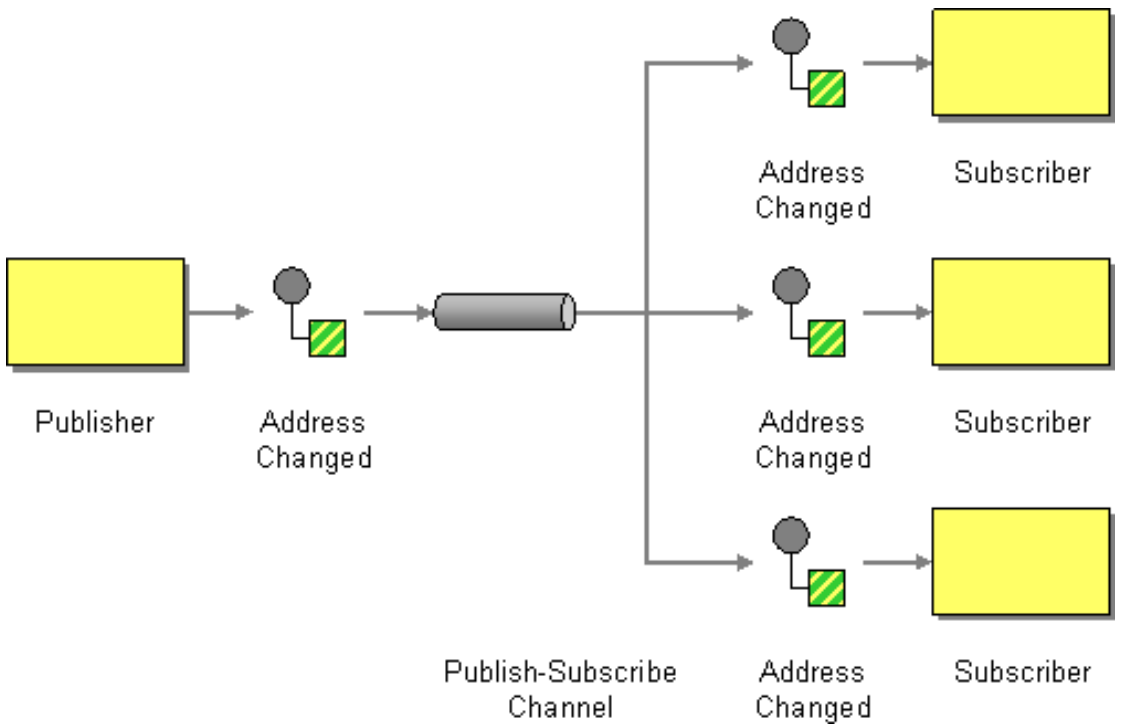



Image: Hohpe and Woolf

- Send the event on a Publish-Subscribe Channel, which delivers a copy of a particular event to each receiver.

# DataType Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the application send a data item such that the receiver will know how to process it?**

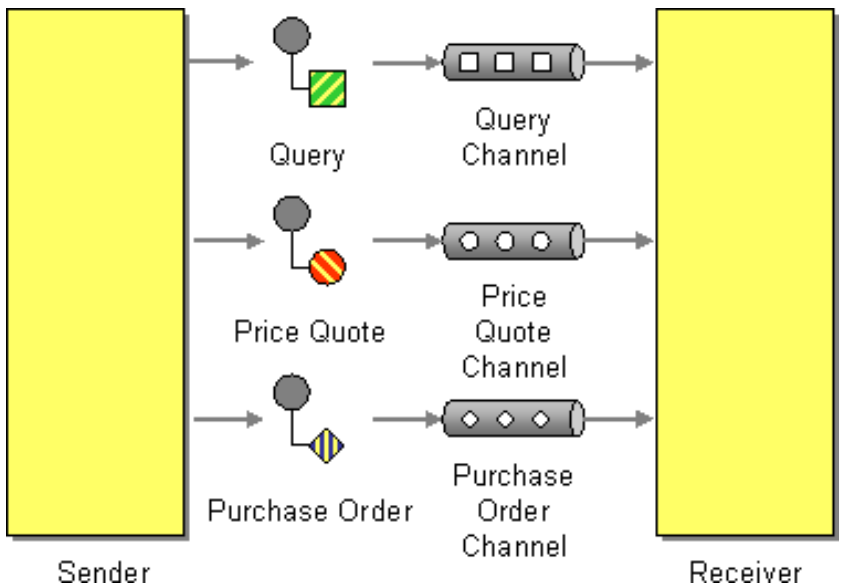



Image: Hohpe and Woolf

- Use a separate Datatype Channel for each data type, so that all data on a particular channel is of the same type.

# DataType Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the application send a data item such that the receiver will know how to process it?**

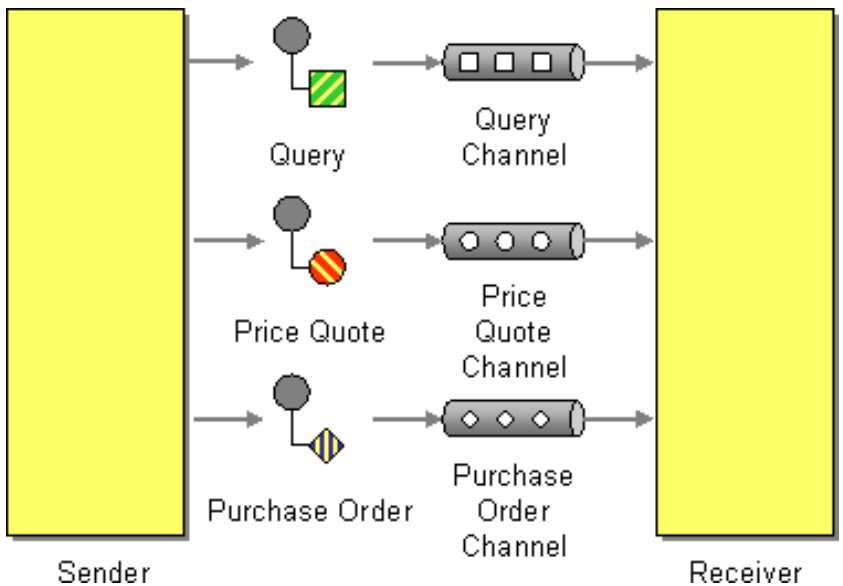



Image: Hohpe and Woolf

- Use a separate Datatype Channel for each data type, so that all data on a particular channel is of the same type.

# Invalid Message Channel

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

**How can a messaging receiver gracefully handle receiving a message that makes no sense?**

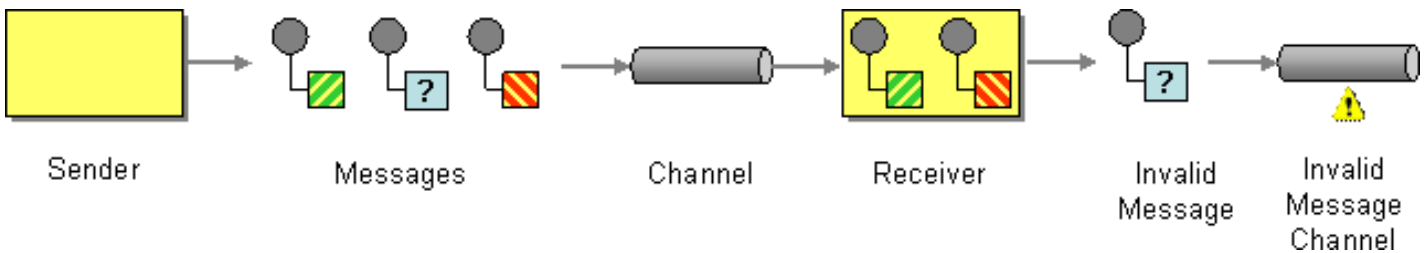



Image: Hohpe and Woolf

- The receiver should move the improper message to an Invalid Message Channel.

# Invalid Message Channel

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

**How can a messaging receiver gracefully handle receiving a message that makes no sense?**

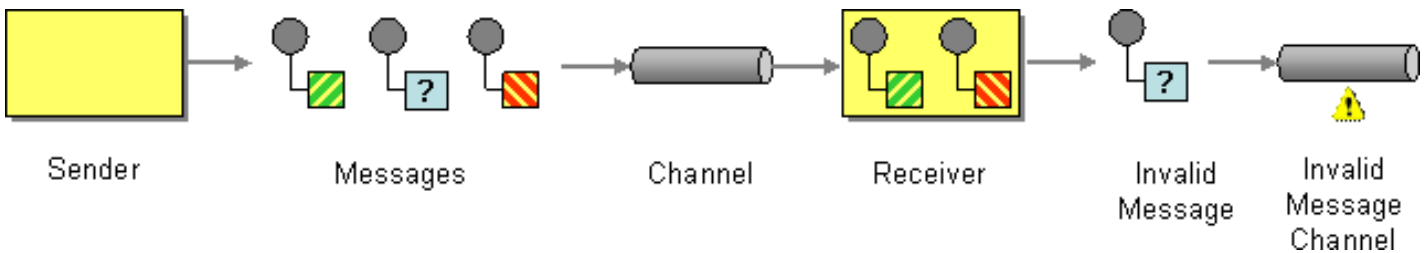



Image: Hohpe and Woolf

- The receiver should move the improper message to an Invalid Message Channel.

# Dead Letter Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## What will the messaging system do with a message it cannot deliver?

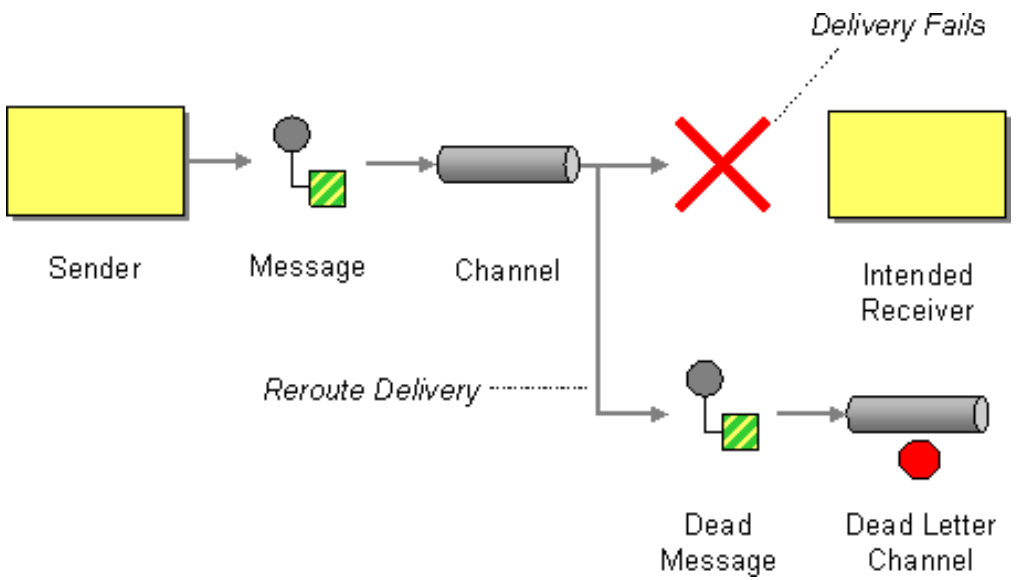



Image: Hohpe and Woolf

- When a messaging system determines that it cannot or should not deliver a message, it may elect to move the message to a Dead Letter Channel.

# Dead Letter Channel

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## What will the messaging system do with a message it cannot deliver?

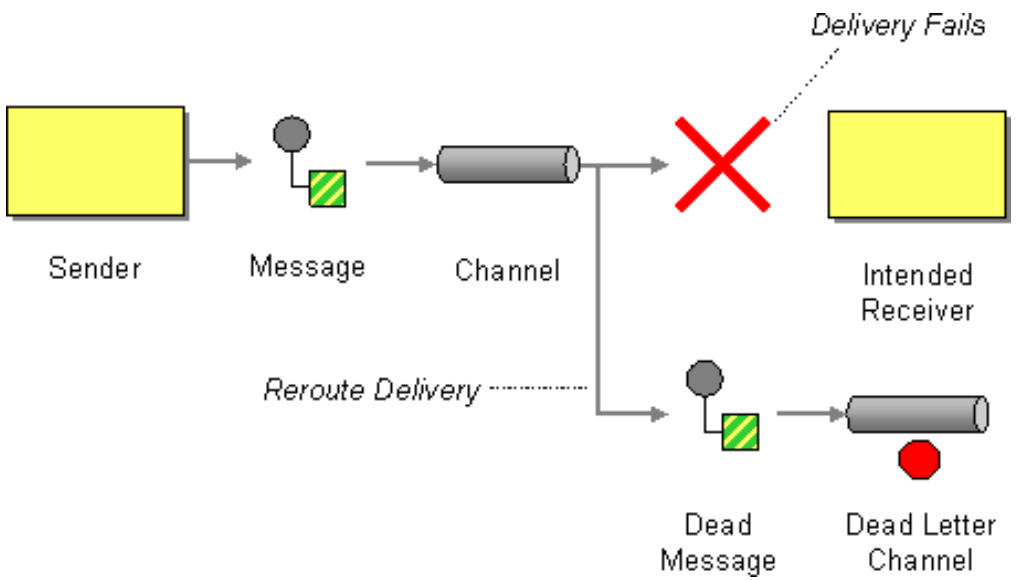



Image: Hohpe and Woolf

- When a messaging system determines that it cannot or should not deliver a message, it may elect to move the message to a Dead Letter Channel.

# Guaranteed Delivery

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the sender make sure that a message will be delivered, even if the messaging system fails?**

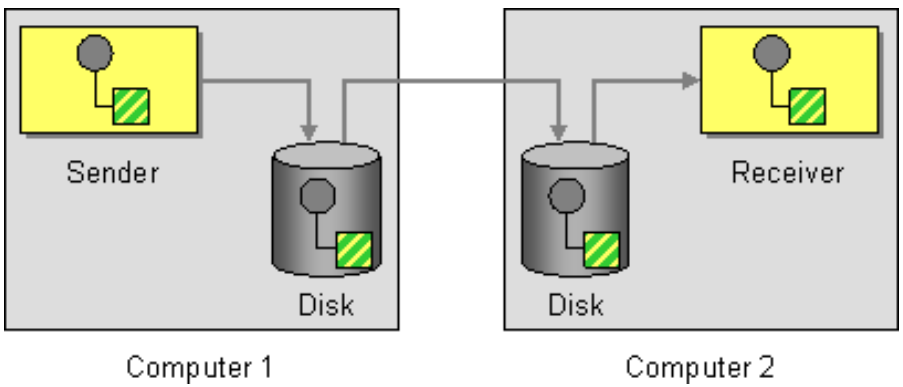



Image: Hohpe and Woolf

- Use Guaranteed Delivery to make messages persistent so that they are not lost even if the messaging system crashes.

# Guaranteed Delivery

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can the sender make sure that a message will be delivered, even if the messaging system fails?**

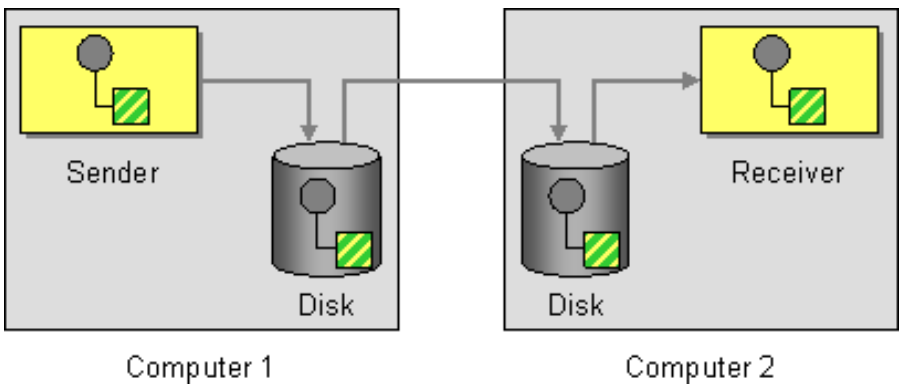



Image: Hohpe and Woolf

- Use Guaranteed Delivery to make messages persistent so that they are not lost even if the messaging system crashes.

# Channel Adapter

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

**How can you connect an application to the messaging system so that it can send and receive messages?**

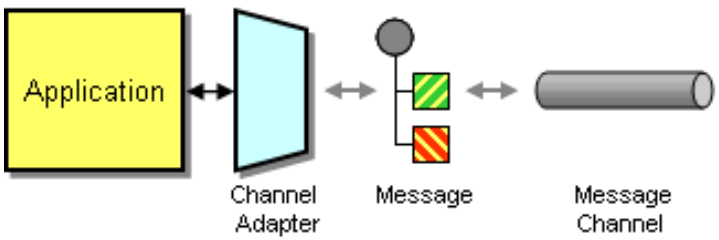



Image: Hohpe and Woolf

- Use a Channel Adapter that can access the application's API or data and publish messages on a channel based on this data, and that likewise can receive messages and invoke functionality inside the application.

# Channel Adapter

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

**How can you connect an application to the messaging system so that it can send and receive messages?**

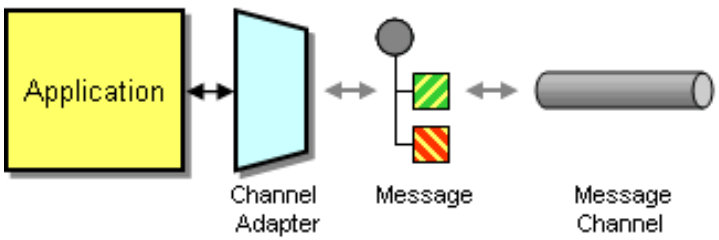



Image: Hohpe and Woolf

- Use a Channel Adapter that can access the application's API or data and publish messages on a channel based on this data, and that likewise can receive messages and invoke functionality inside the application.

# Messaging Bridge

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can multiple messaging systems be connected so that messages available on one are also available on the others?**

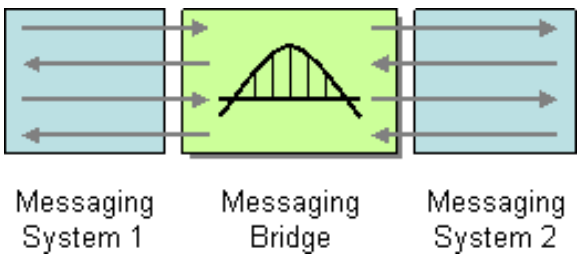



Image: Hohpe and Woolf

- Use a Messaging Bridge, a connection between messaging systems, to replicate messages between systems.

# Messaging Bridge

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can multiple messaging systems be connected so that messages available on one are also available on the others?**

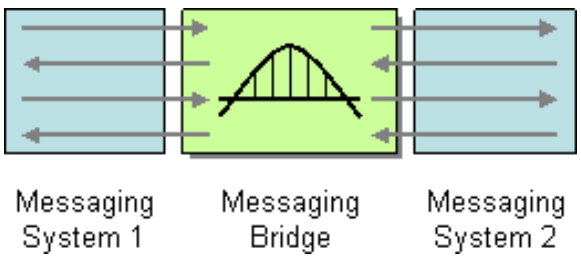



Image: Hohpe and Woolf

- Use a Messaging Bridge, a connection between messaging systems, to replicate messages between systems.

# Message Bus

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**What is an architecture that enables separate applications to work together, but in a decoupled fashion such that applications can be easily added or removed without affecting the others?**

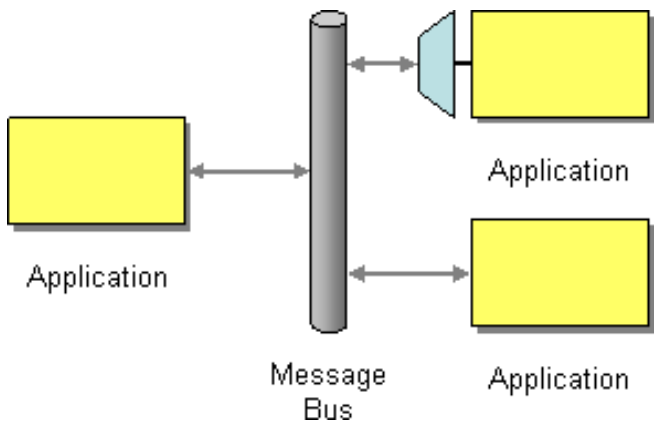



Image: Hohpe and Woolf

- Structure the connecting middleware between these applications as a Message Bus
- It enables them to work together using messaging

# Message Bus

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**What is an architecture that enables separate applications to work together, but in a decoupled fashion such that applications can be easily added or removed without affecting the others?**

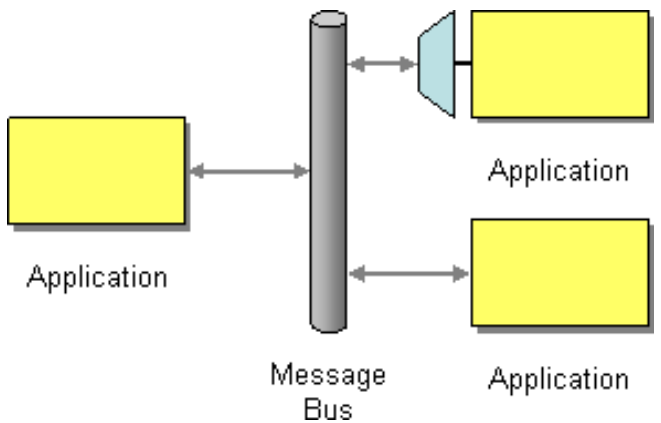



Image: Hohpe and Woolf

- Structure the connecting middleware between these applications as a Message Bus
- It enables them to work together using messaging

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. **Demo**
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. **Messaging: Message Construction** / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Integration Patterns

## Message Construction

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

# Content

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

- Focus on patterns
- Different message types
- Examples given (Java, XML, SOAP)

- Command Messages: Invoke a procedure in another application
- Document Messages: Transfer data between applications
- Event Messages: Transmit events from application to another

- Request-Reply: Get response from the receiver
- Special Characteristics: Return Address, Correlation Identifier, Message Sequence, Message Expiration, Format Indicator

# Message Types

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Command Messages

- Used to reliably invoke a procedure in another application

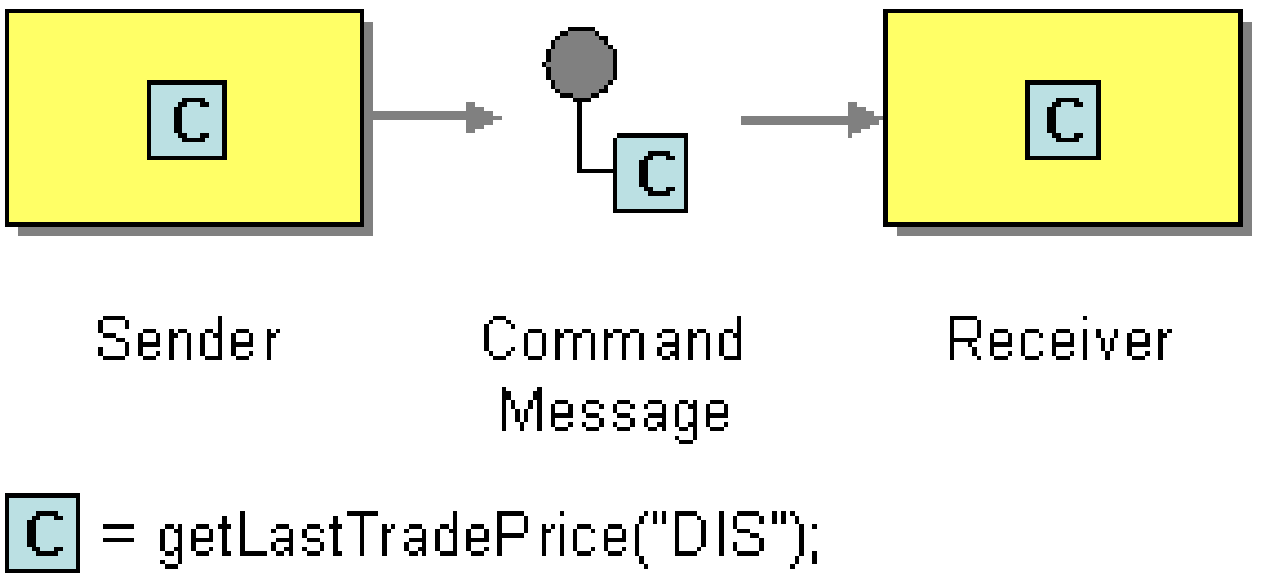



Image: Hohpe and Woolf

# Message Types

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Document Messages

- Used to transfer data between applications

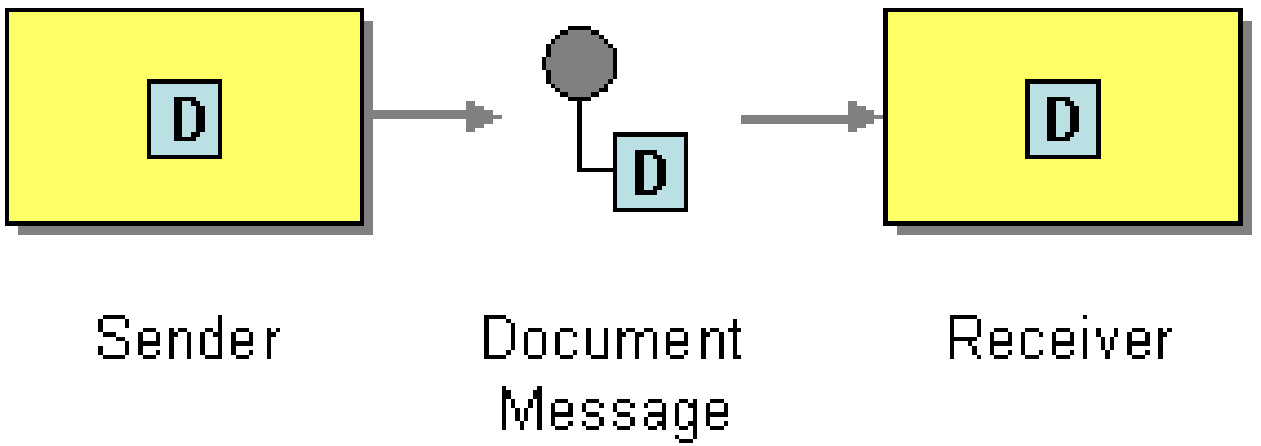



Image: Hohpe and Woolf

# Document Messages

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Message Sequence

- Used to transfer a large amount of data between applications

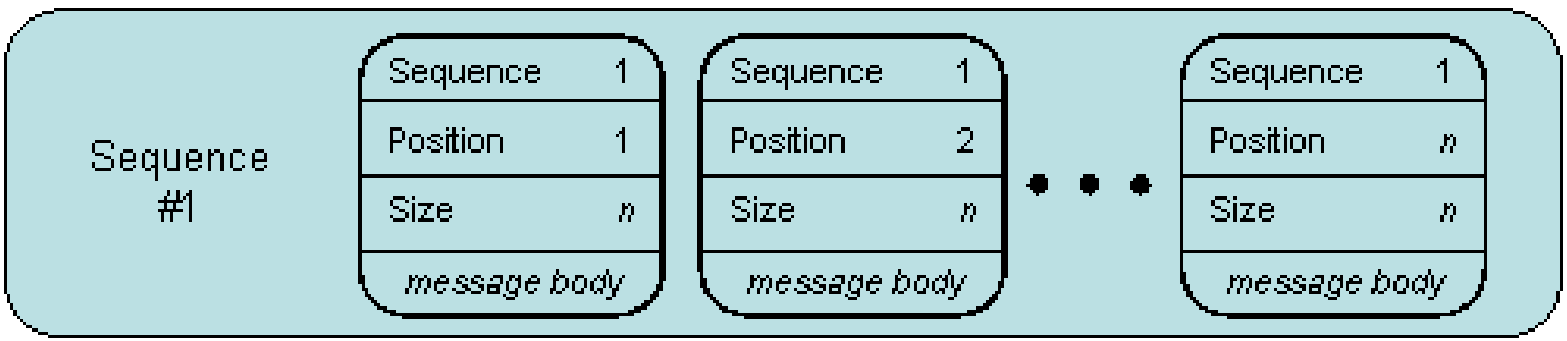



Image: Hohpe and Woolf

# Message Types

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Event Messages

- Used to transmit events between applications

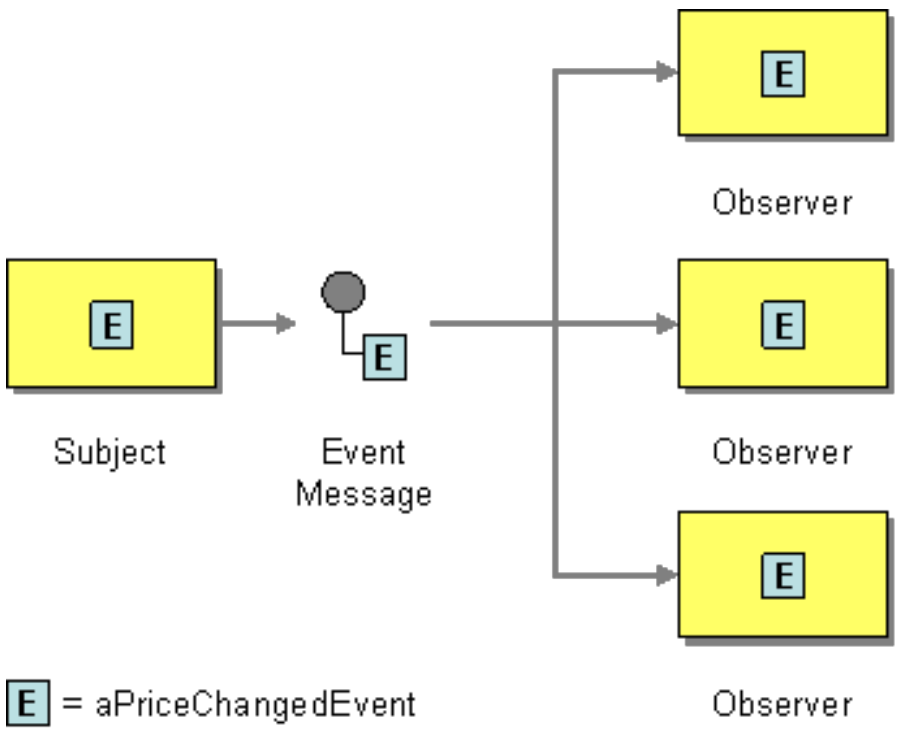



Image: Hohpe and Woolf

# Message Types

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

## Request-Reply

- Enables two-way conversation via Messaging

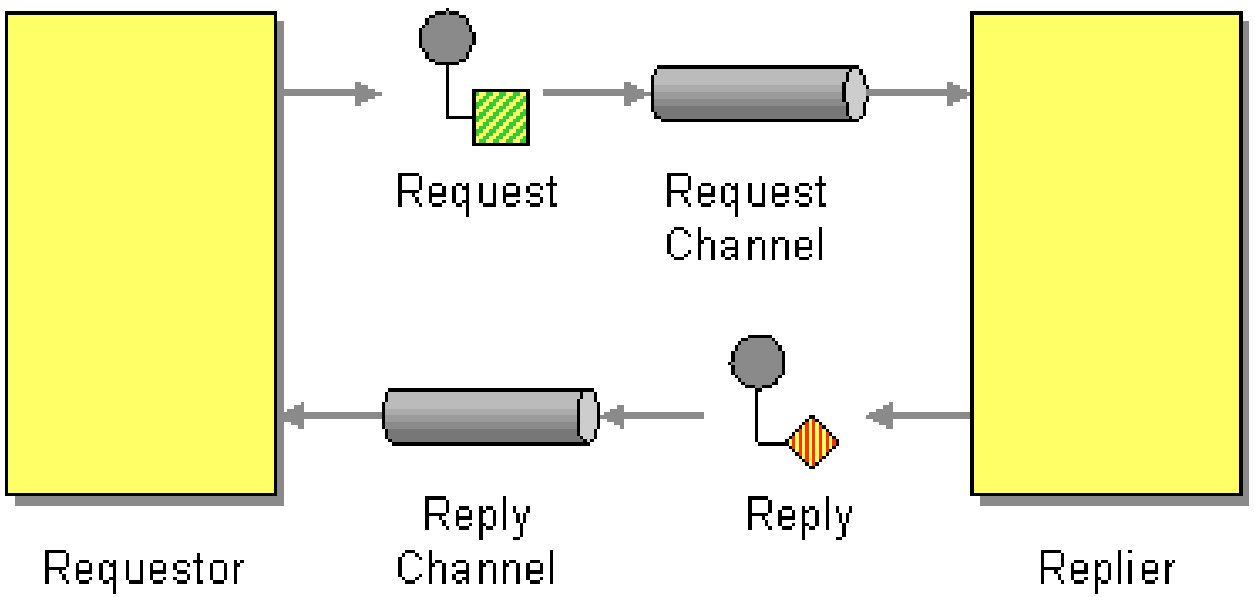



Image: Hohpe and Woolf

# Request-Reply

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Return Address

- Used to determine where to send the reply
- Add a Return Address to indicate the destination

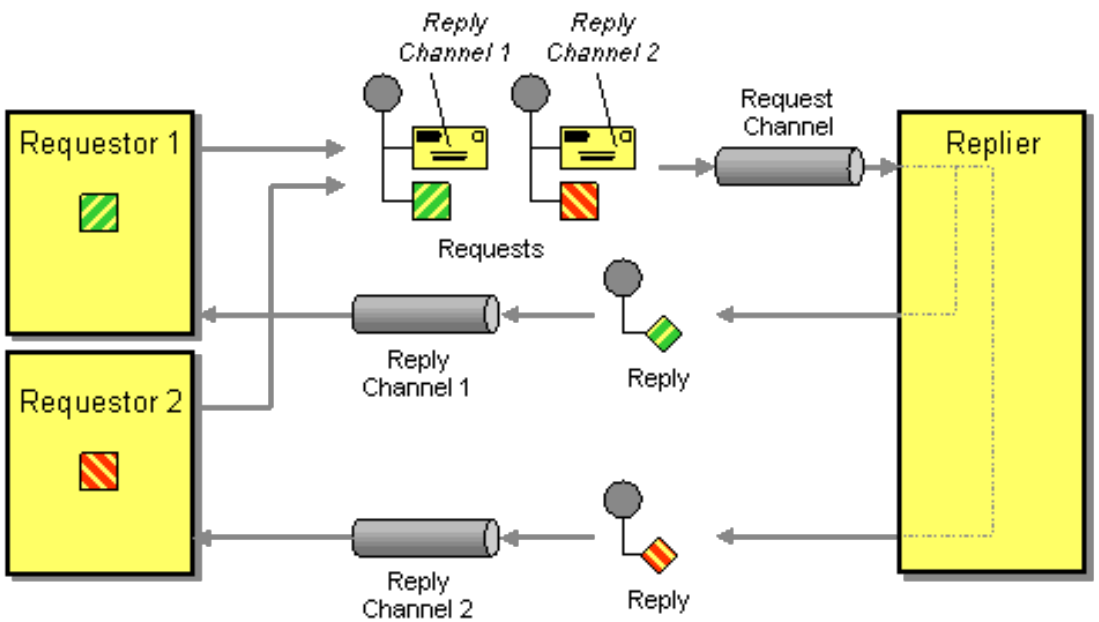



Image: Hohpe and Woolf

# Request-Reply

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Correlation Identifier

- Define which request belongs to which reply
- Add a unique Correlation Identifier for Identification

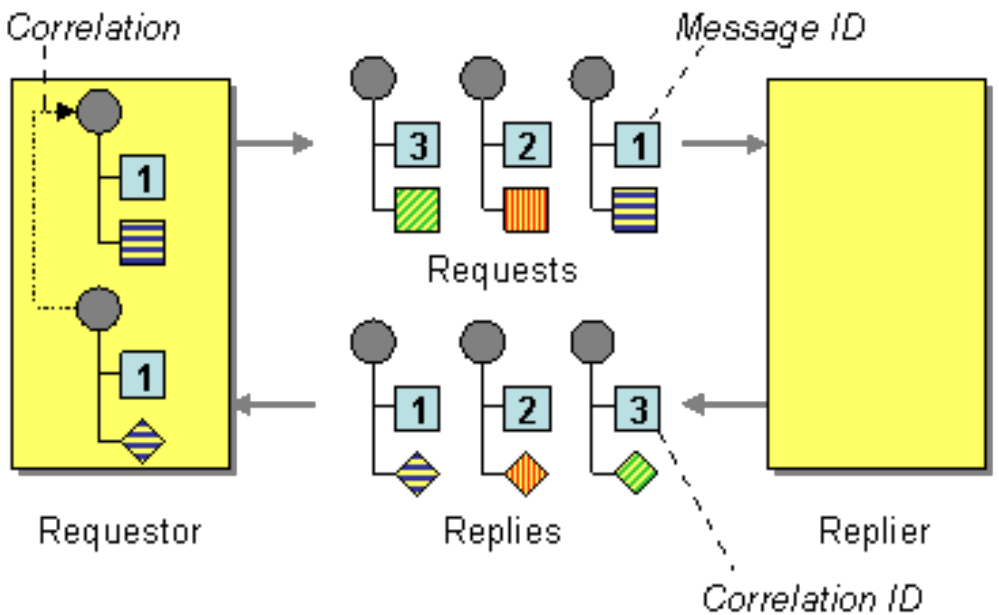



Image: Hohpe and Woolf

# Special Pattern

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Message Expiration

- Used to decide if a message is useful anymore

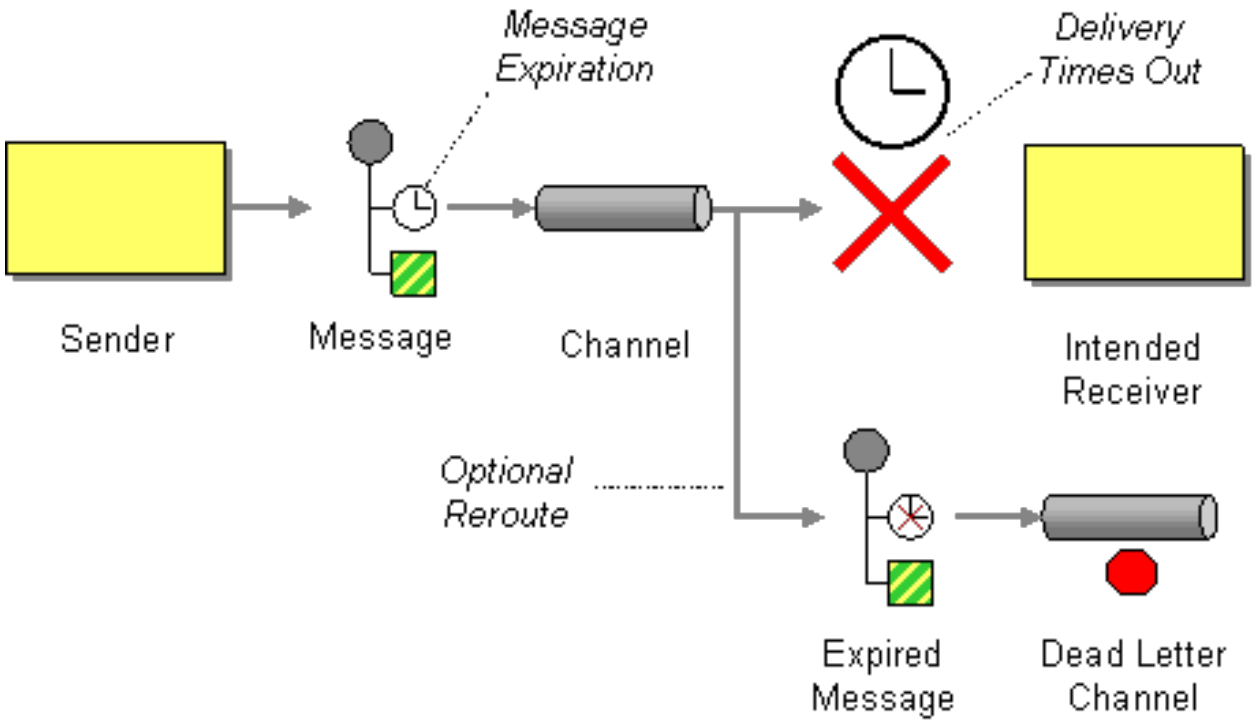



Image: Hohpe and Woolf

# Special Pattern

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Format Indicator

- Used for possible future changes
- Add a Format Indicator to the messages to specify which format is used
  - Version Number
  - Foreign Key
  - Format Document

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / **Message Routing**
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Integration Patterns

## Message Routers

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Message Router

- determines how to navigate the channel topology
- directs message to final receiver or next router
- surrounding components are unaware of its existence
- concerns itself only with the message direction (content remains unchanged)

## Key Benefit

"...the decision criteria for the destination of a message are maintained in a single location."[a]

[a] Gregor Hohpe and Bobby Woolf, *Enterprise Integration Patterns*

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Problems with Message Routers

- maintenance bottleneck if list of possible destinations changes frequently
- possible performance bottleneck
  - message decoding necessary before placing it on a channel
  - multiple routers in parallel minimize effect (message order problem)
- difficult to understand the message flow with multiple routers
  - loose coupling $\Rightarrow$ sender does not know receiver of messages
  - alleviate problem using the Message History pattern

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Overview of Simple Routers

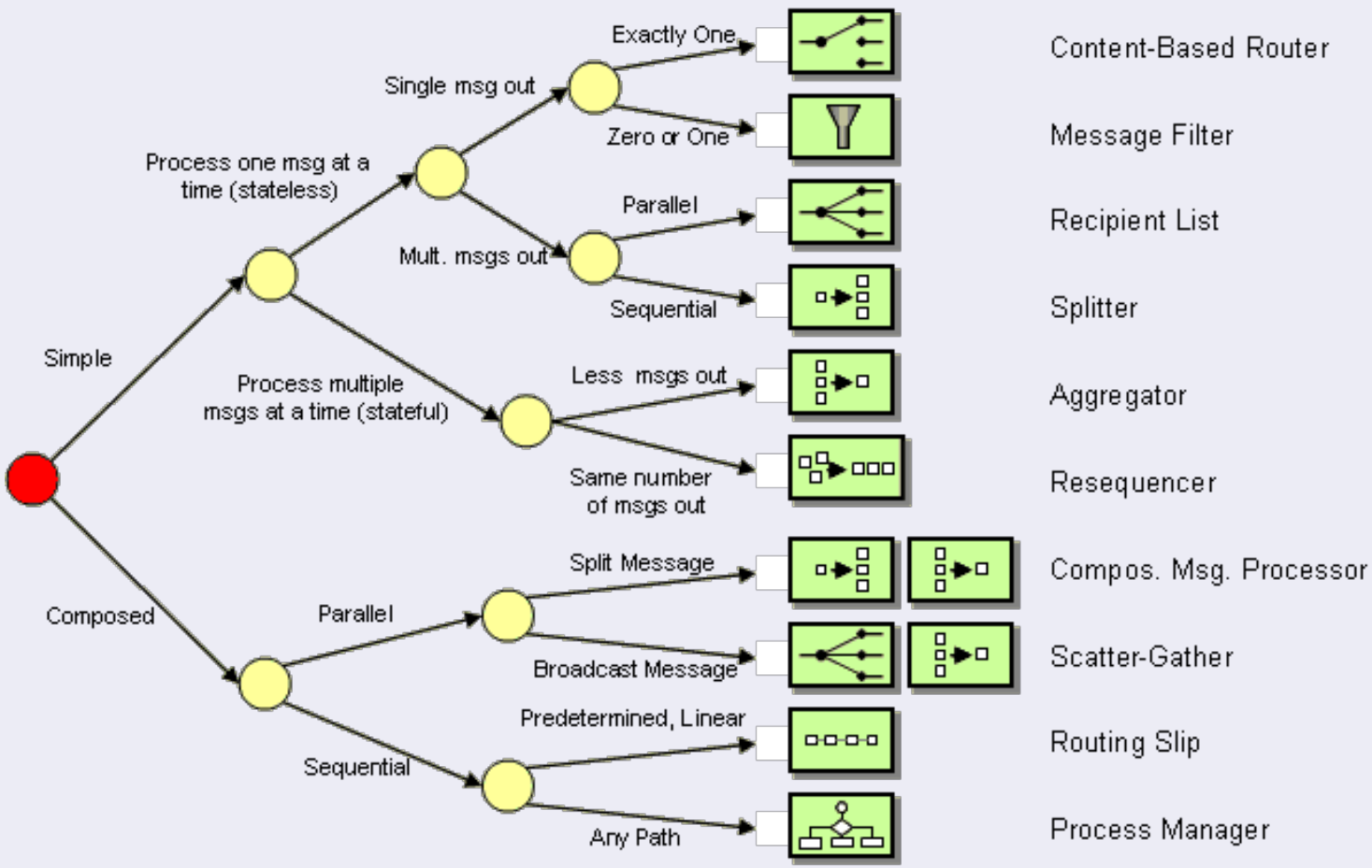



Image: Hohpe and Woolf

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Content-Based Router

- stateless
- consumes one message in one step
- publishes one message in one step
- routing based on message content, field values, etc. (content is not changed)
- not suited for frequent changes of recipients
- possible alternatives are Message Filter and Routing Slip patterns

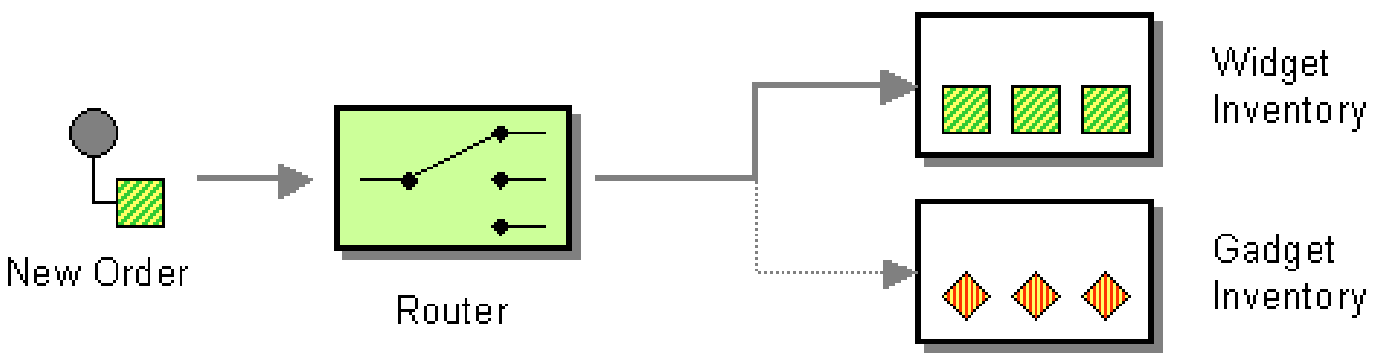



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Message Filter

- mostly stateless
- consumes one message in one step
- publishes zero or one message in one step
- eliminates undesired messages from a channel
- stateful if message limit exists or to prevent duplicates
- routing functionality in combination with a Publish-Subscribe Channel

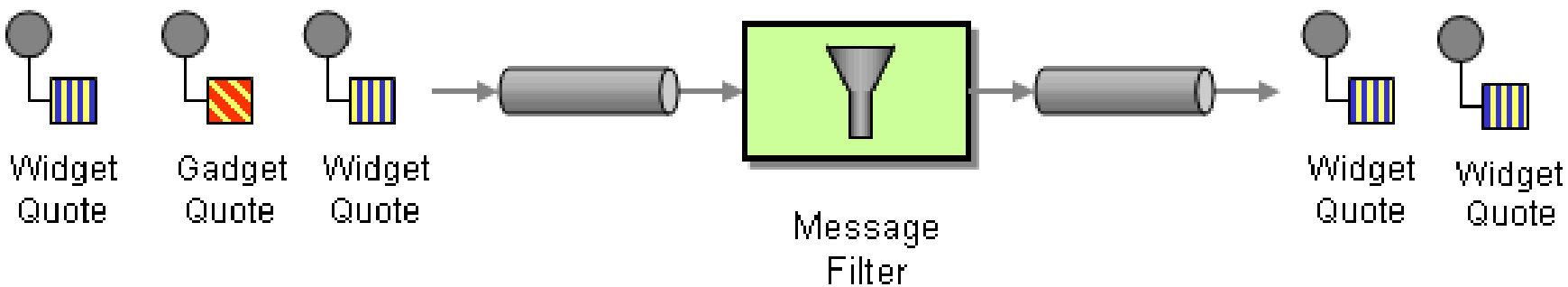



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Pattern Comparison

| **Content-Based Router** | **Publish-Subscribe Channel with Message Filters** |
|---|---|
| Exactly one consumer per message. | More than one consumer possible. |
| Central control and maintenance - predictive routing. | Distributed control and maintenance - reactive filtering. |
| Router needs to know about participants. | No knowledge of participants required. |
| Often used for business transactions, e.g., orders. | Often used for event notifications or information messages. |
| Generally more efficient with queue-based channels. | Generally more efficient with publish-subscribe channels. |

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Dynamic Router

- dynamic variant of the Content-Based Router
- additional control channel for special configuration messages used for self-configuration
- configuration rule conflicts have to be solved:
  1. ignore control messages that conflict with existing rules
  2. send message to first recipient whose criteria match
  3. send message to all recipients whose criteria match (rule violation)

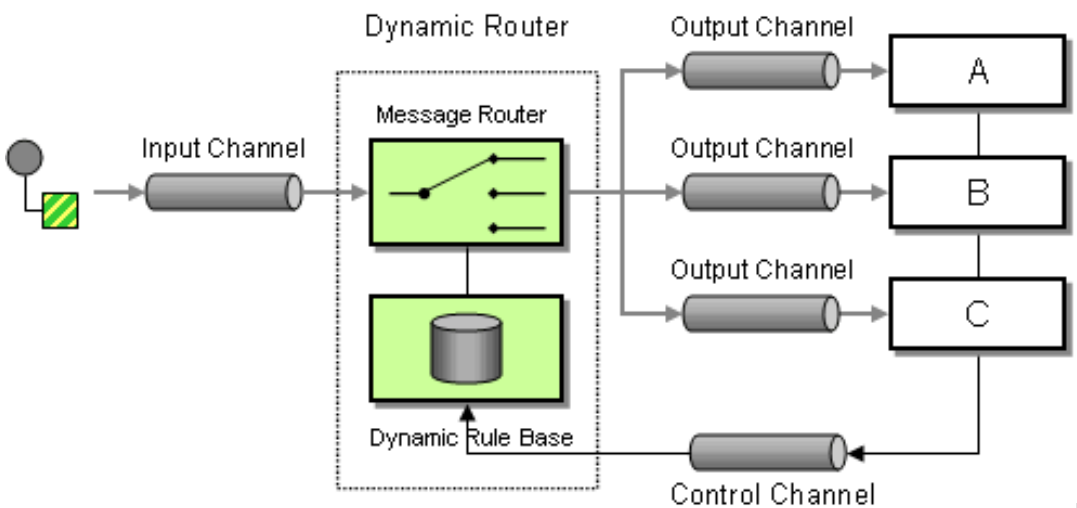



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Recipient List

- stateless
- consumes one message in one step
- publishes multiple messages at a time (incl. 0)
- complete operation must be atomic ⇒ multiple solutions:
  1. Single transaction: use transactional channels and single commit
  2. Persistent recipient list: already sent messages stored on disk or DB
  3. Idempotent receivers: resend all messages on restart possible

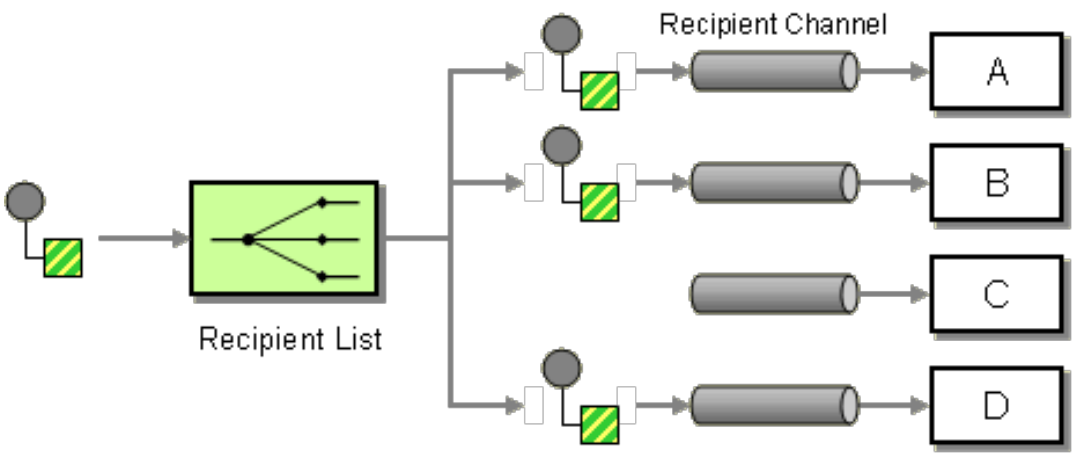



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Splitter

- stateless
- consumes one message in one step
- publishes multiple messages in one step
- breaks one message into a series of individual messages
- some elements may be included in each child message
- useful to add sequence number and reference to original message
  $\Rightarrow$ improves traceability and simlifies the task of an Aggregator

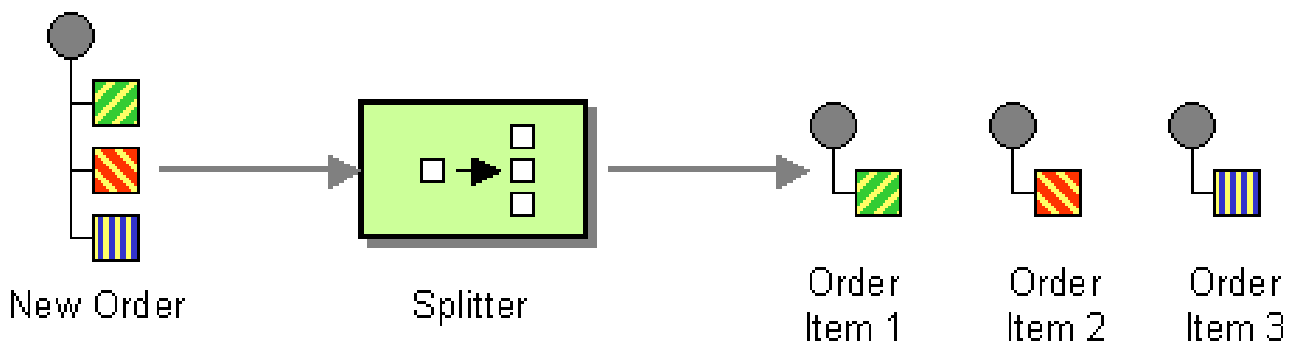



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Aggregator

- stateful
- consumes multiple messages in one step
- publishes one message in one step
- collects and stores individual messages
- publishes single message for a complete set
- important to specify Correlation, Completeness Condition and Aggregation Algorithm
- *Correlation:* typically by message type or correlation identifier

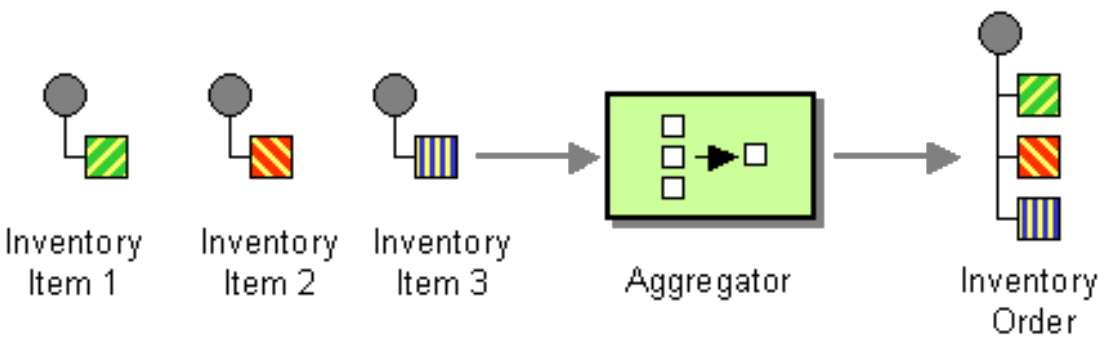



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Aggregator: Completeness Condition Strategies

1. Wait for All: common in an order scenario
2. Timeout: common in bidding scenarios
3. First Best: only practical where response time is critical
4. Timeout with Override: early abortion with very favorable response
5. External Event: via control channel or specially formatted message

## Aggregator: Aggregation Strategies

1. Select the "best" answer: f.e. in a bidding scenario
2. Condense data: reduces network traffic from a high traffic source
3. Collect data for later evaluation: if no decision can be made

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Resequencer

- stateful
- consumes multiple messages in one step
- publishes same number of messages it consumes
- collects and reorders messages
- sequence numbers have to be in order and consecutive $\Rightarrow$ identify missing messages
- out-of-order messages have to be kept until missing messages arrive (or compute stand-in message)

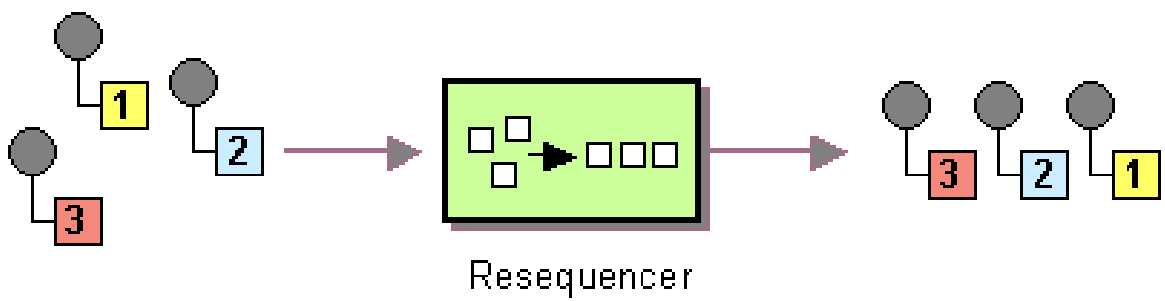



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Composed Message Processor

1. split a composite message
2. route submessages to appropriate destinations
3. reaggregate responses into single message

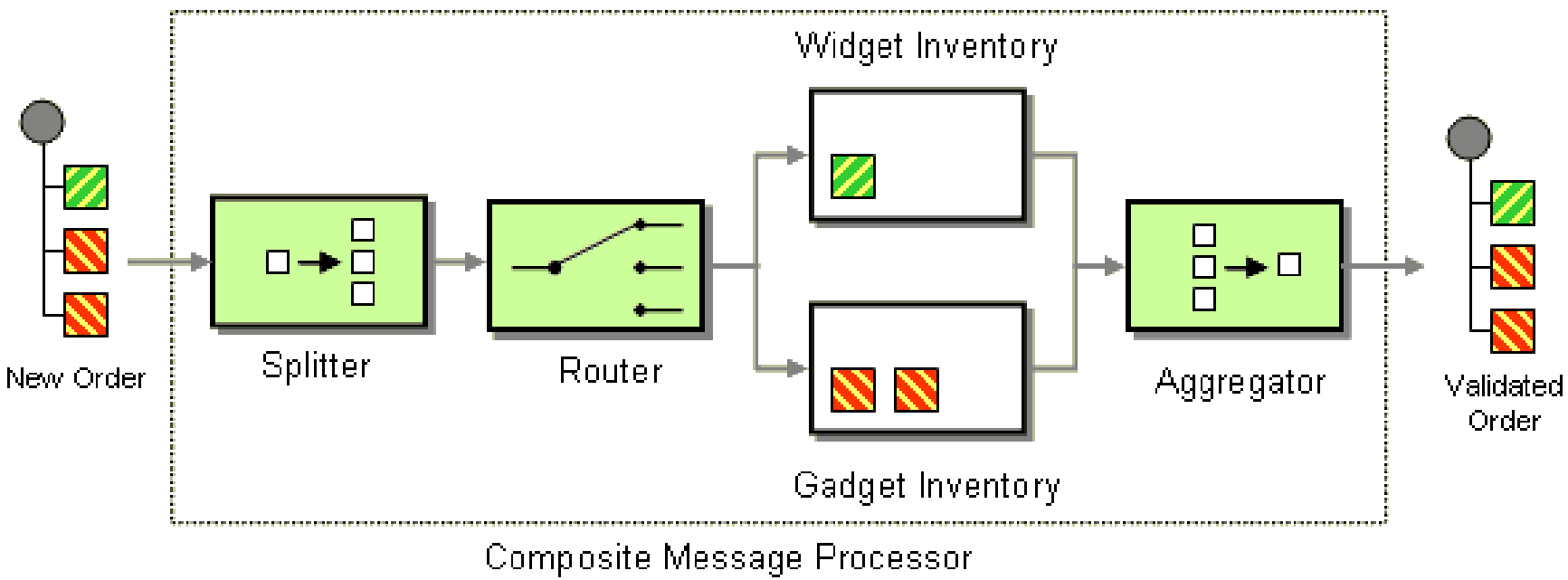



Image: Hohpe and Woolf

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Scatter-Gather

1. broadcast message to multiple destinations
2. reaggregate responses into single message

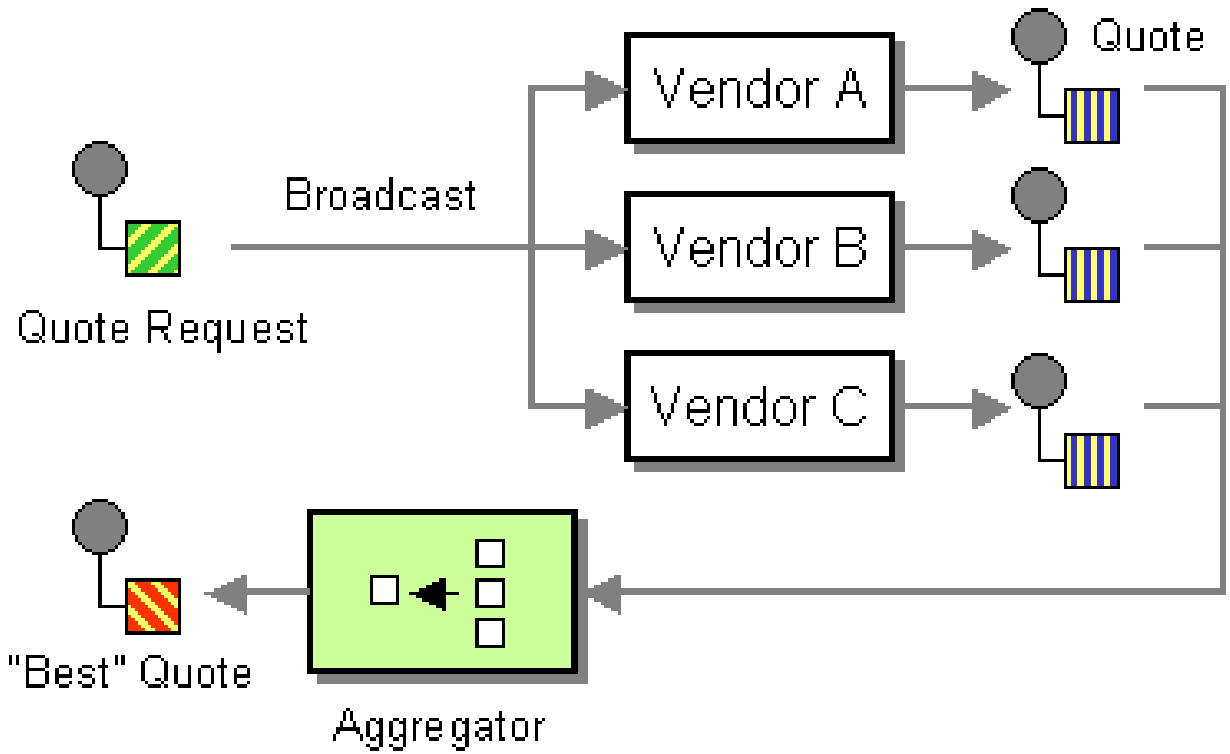



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Routing Slip

1. attach routing slip to message, specifying the sequence of processing steps
2. wrap each component with a special message router
3. read the routing slip and route the message accordingly

- used to gather messages from multiple locations, each requiring a different transformation

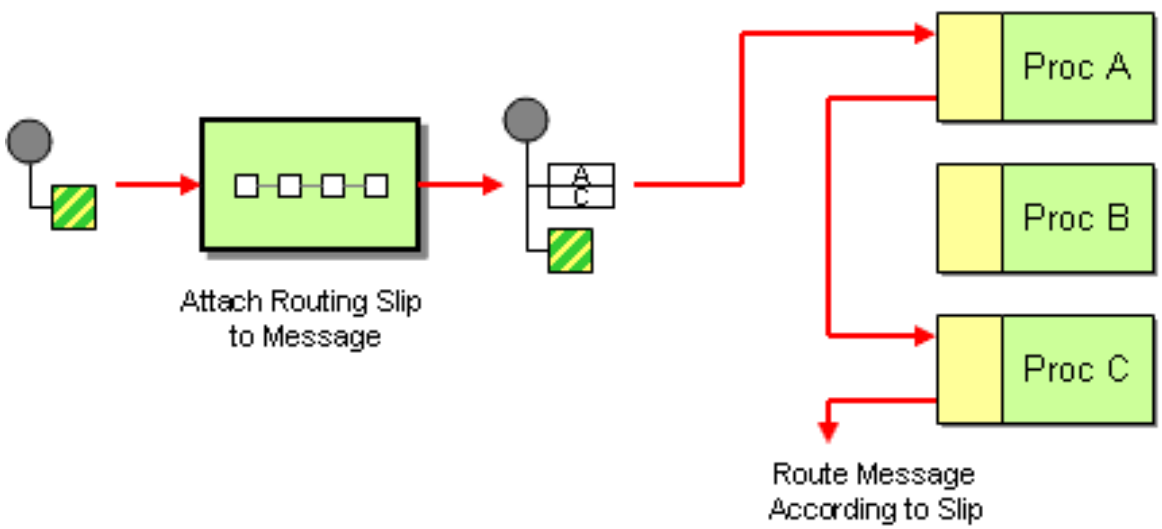



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Process Manager

- used if processing steps are not known during design time
- determines next processing step based on intermediate results
- solves almost every routing problem
- can cause significant performance overhead
- may distract from the core design issue

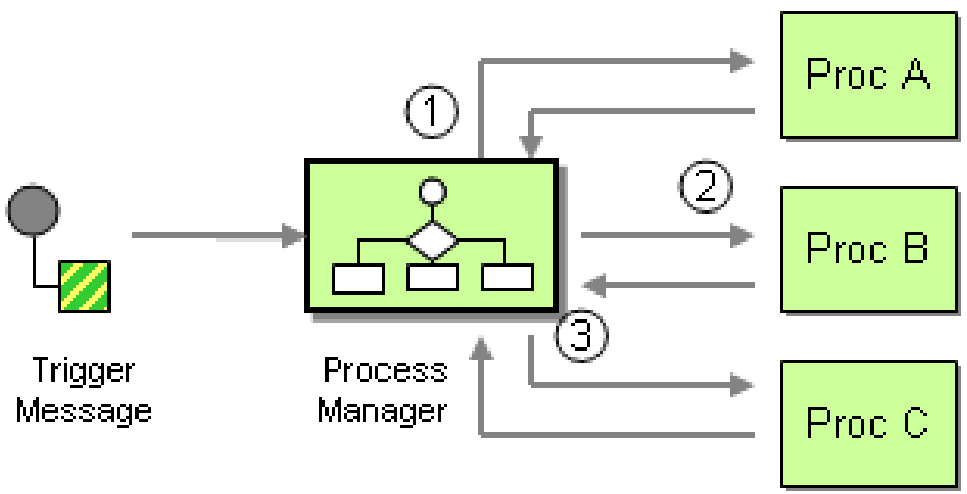



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Process Manager: Points to Consider

- Managing State: store actual processing step and intermediate results for every returned message
- Process Instances: create process instance for each trigger message to manage parallel executions
- Correlation: use a correlation identifier to associate incoming messages with a process instance
- Keeping State in Messages: allows better reporting and simpler debugging
- Creating the Process Definition: mostly XML documents, describing the process model using the BPEL4WS language, are used - can be generated with visual tools

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Message Broker

- receives messages from multiple destinations
- determines the correct destination for each message
- routes messages to the correct channel
- internals are implemented using other message routers
- multiple message brokers may communicate with a central message broker

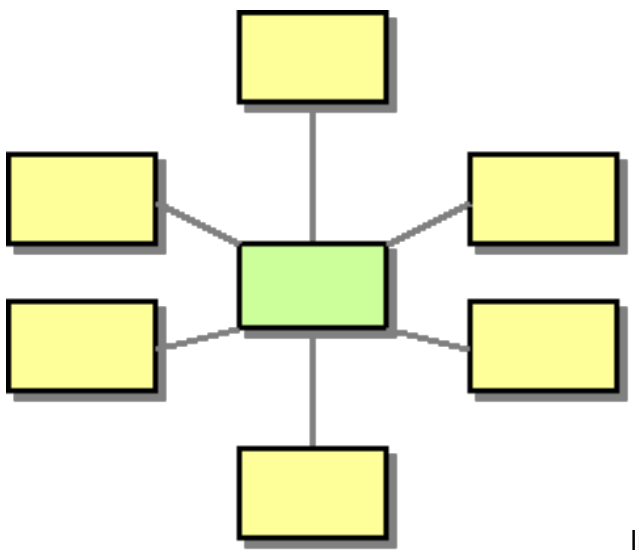


Image: Hohpe and Woolf

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. **Demo**
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. **Messaging: Message Transformation** / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Integration Patterns

## Message Transformation

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Message Translator

- translates one data format into another
- used on different levels:
  - Data Structures (Application Layer): f.e. condense many-to-many relationship into aggregation
  - Data Types: f.e. convert ZIP code from numeric to string
  - Data Representation: parse data representation and render in different format, de-/encrypt as necessary
  - Transport: move data across protocols without affecting message content

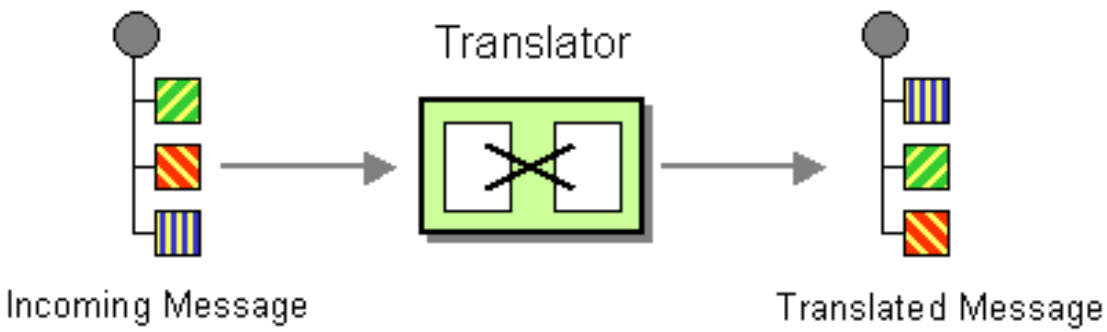



Image: Hohpe and Woolf

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Metadata Management

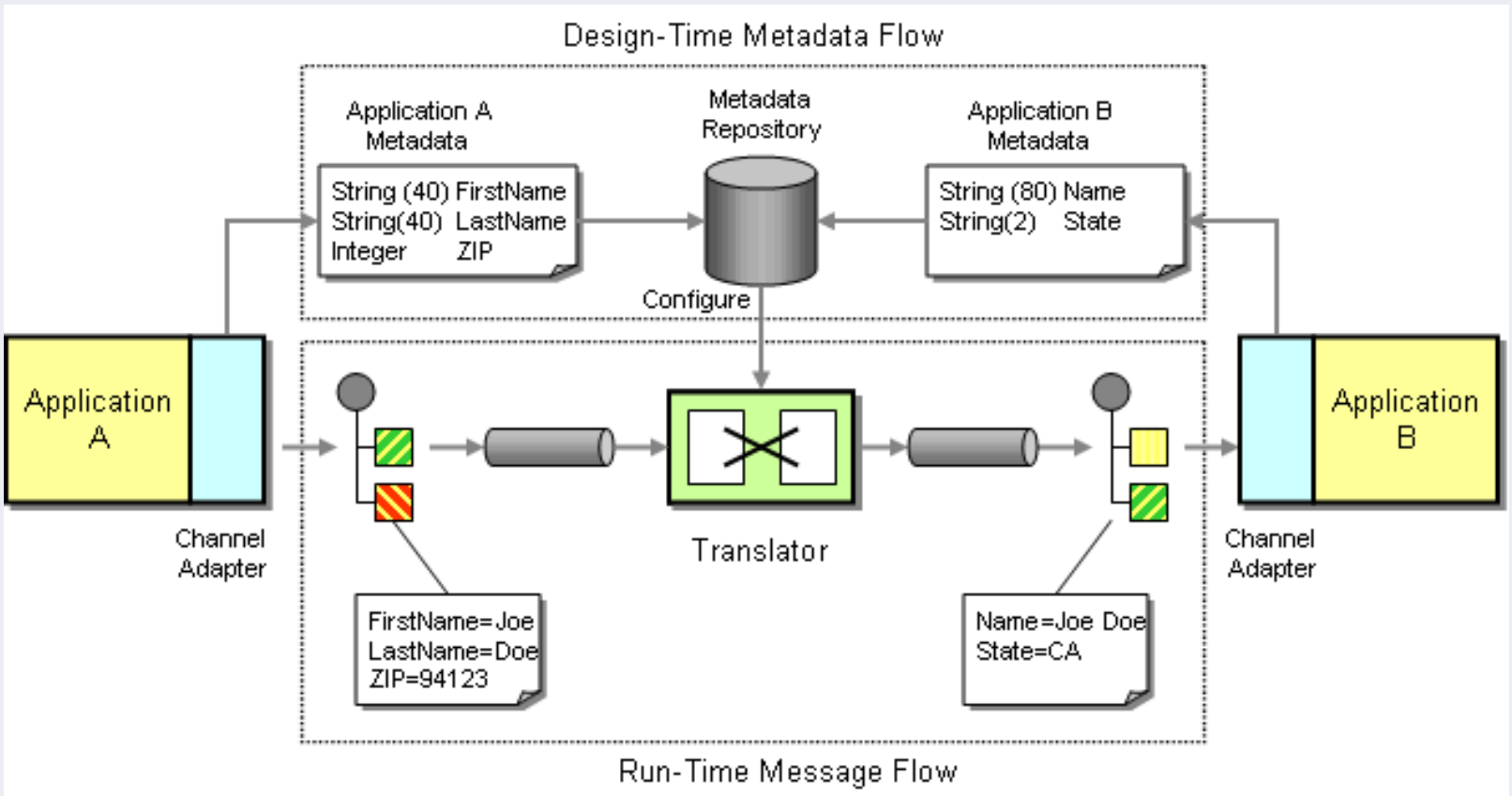



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Envelope Wrapper

- wraps application data inside an envelope
- compliant with the message infrastructure
- unwraps the message at its final destination
- ⇒ systems can participate in messaging exchange with specific formation requirements
  - message header fields
  - message encryption

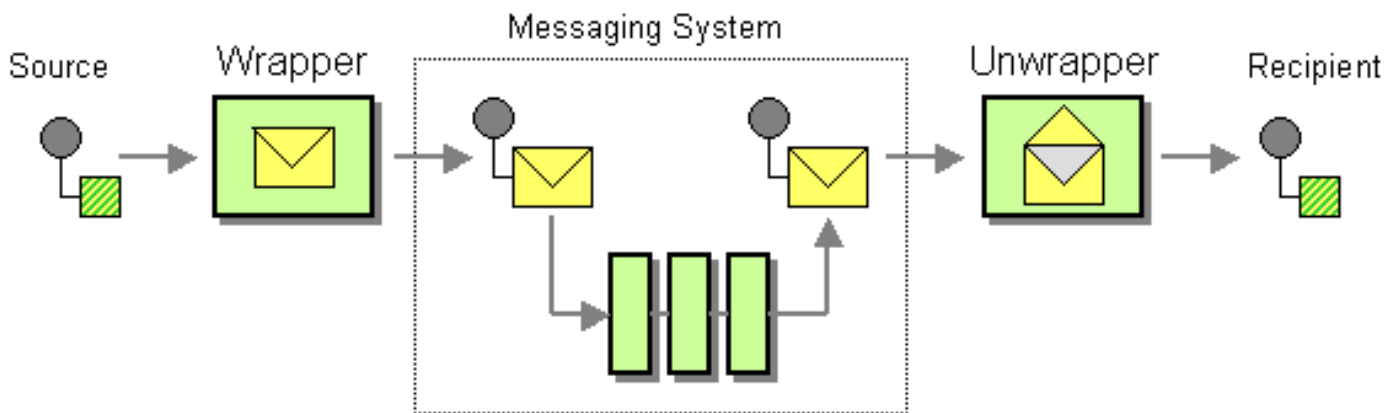



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Content Enricher

- augments a message with missing information
- uses external data source, the most common are:
  - Computation
  - Environment (e.g. timestamps)
  - Another System (e.g. database, LDAP directory, etc.)
- needed if message originator does not have all required data items available

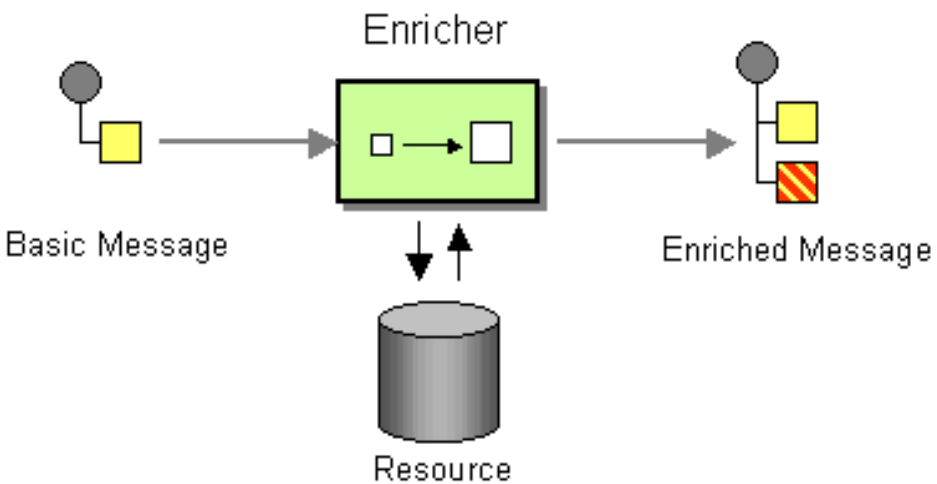



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Content Filter

- removes unimportant items from a message
- only important items are left

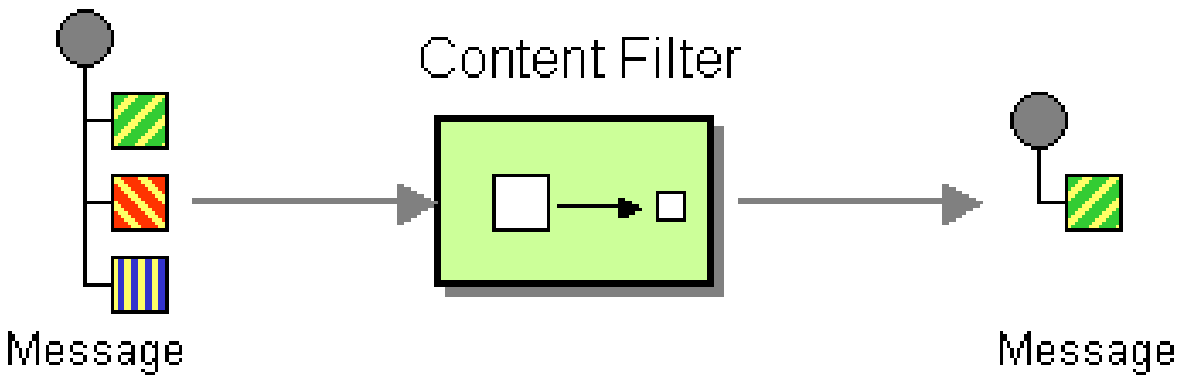



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Claim Check

- stores message data in persistent store
- claim check is passed to subsequent components
- stored information can be restored using the claim check
- used to reduce data volume of a message without sacrificing information content

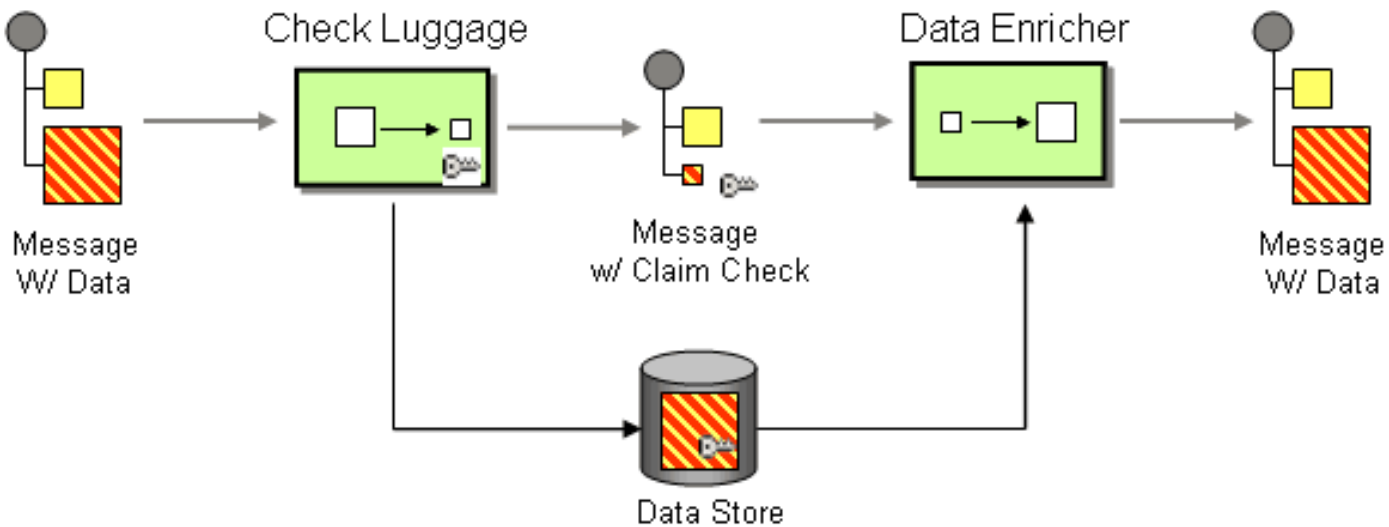



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Normalizer

- routes each message type through a custom message translator
- resulting messages match a common format
- enables processing of semantically equivalent messages that arrive in a different format

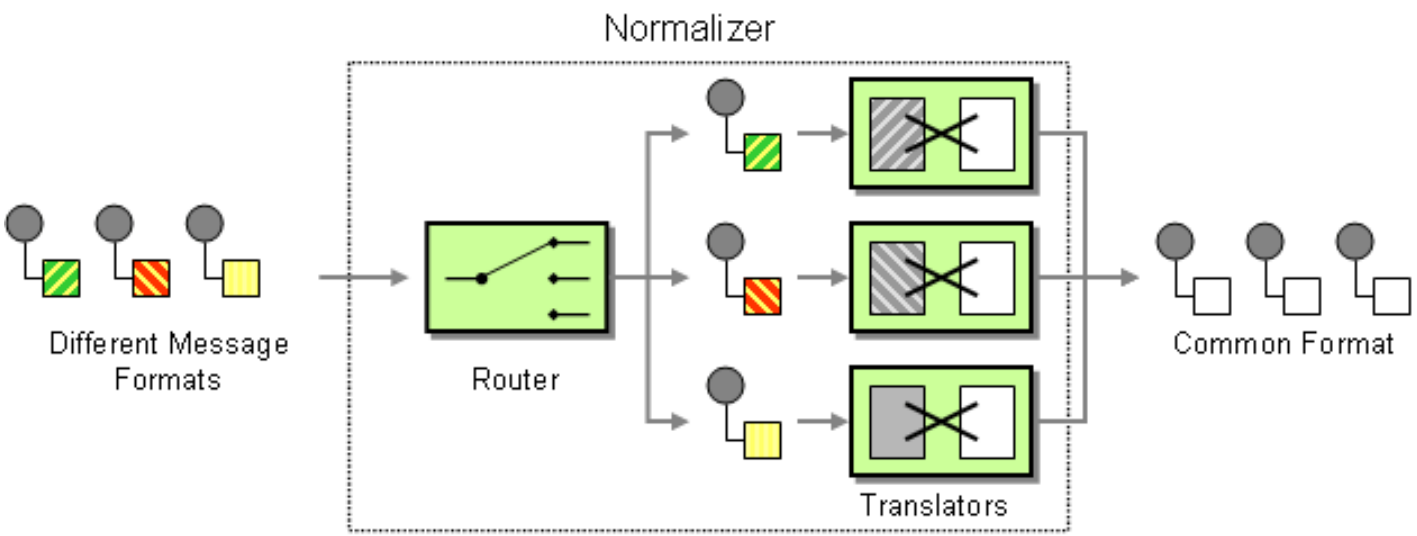



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Canonical Data Model

- independent from any specific application
- has to be able to store information for all participating applications
- each application is required to produce and consume messages in this common format
- minimizes dependencies when integrating applications with different data formats

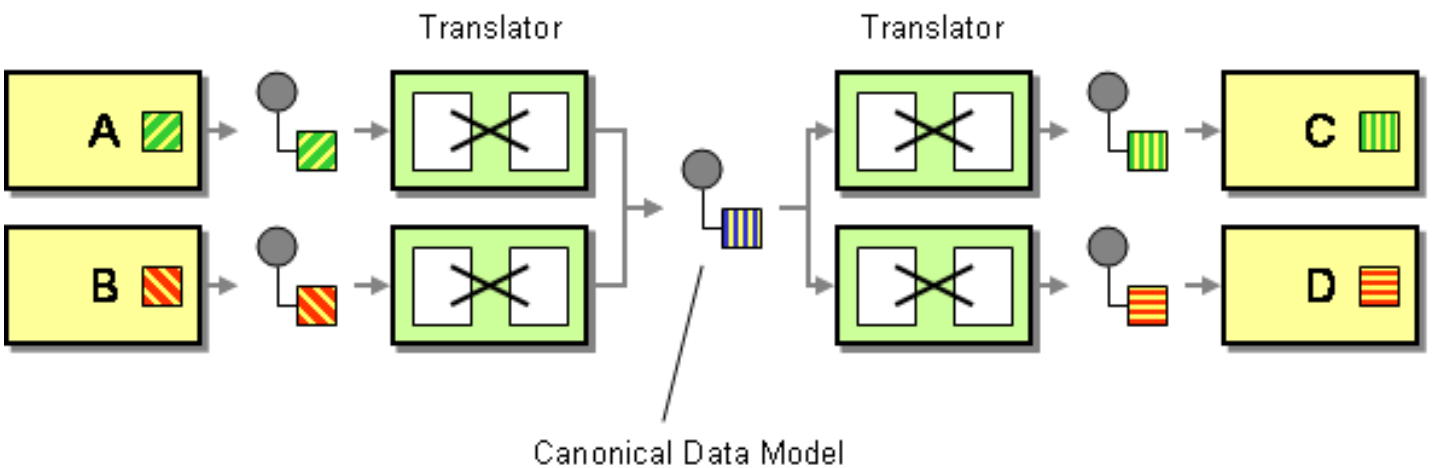



Image: Hohpe and Woolf

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / **Message Endpoints**
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Enterprise Integration Patterns

## Messaging Endpoints

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Messaging Endpoints

- Application connects to a messaging system so that it can send and receive messages
- Transfer packets of data frequently, immediately, reliably, and asynchronously, using customizable formats
- Several Patterns to perform this job for both sending and receiving message
- Sending Messages is easier than receiving

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Messaging Endpoints

Receiving messages is tricky and involves lots of issues

- Throttling: Application does not have to process the messages as rapidly as they are delivered
- Synchronous or Asynchronous consumer: Polling Consumer or Event Driven Consumer
- Message Assignment : Competing Consumers, Message Dispatcher
- Accept all messages or filter: Selective Consumer
- Subscribe while disconnected: Durable Subscribe
- Idempotency: Idempotent Receiver

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Patterns for Senders and Receivers

- Send and Receive Patterns: How applications relate to the messaging systems
- Messaging Gateway: Encapsulates the messaging code
- Messaging Mapper: Data translation
- Transactional Client: Control or regulate multiple messages transactions

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Messaging Gateway

- Data transmission takes place via Messaging Channel
- When designing an application, a developer has to know where to put what types of data, to share that data with other applications, and likewise where to look for what types of data coming from other applications
- When application integrates with others with messaging, there is a thin layer of code is which attaches the application to the messaging system called Messaging Gateway

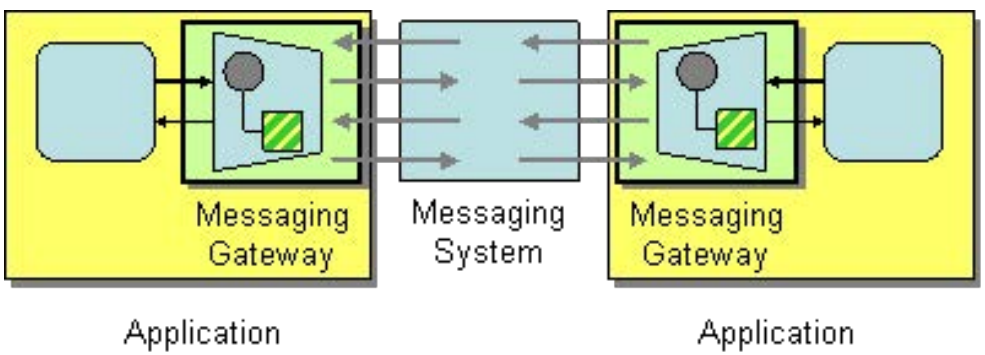



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Messaging Mapper

- Moving data between domain objects and the messaging infrastructure
- In many cases, message format is defined by a common messaging standard and may not correspond to the domain object
- One way is to use Message Translator in the messaging layer or the domain can itself create message in the required format using Message Mapper
- Shortcoming: The domain must be aware of the message format. Hence the domain maintenance becomes difficult if the format changes

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Transactional Client

- A messaging system uses transactional behaviour internally
- Useful for external clients to regulate scope of the transactions as per their requirement or capacity
- Both a sender and a receiver can be transactional.
- A sender that uses explicit transactions can be used with a receiver that uses implicit transactions, and vice versa.

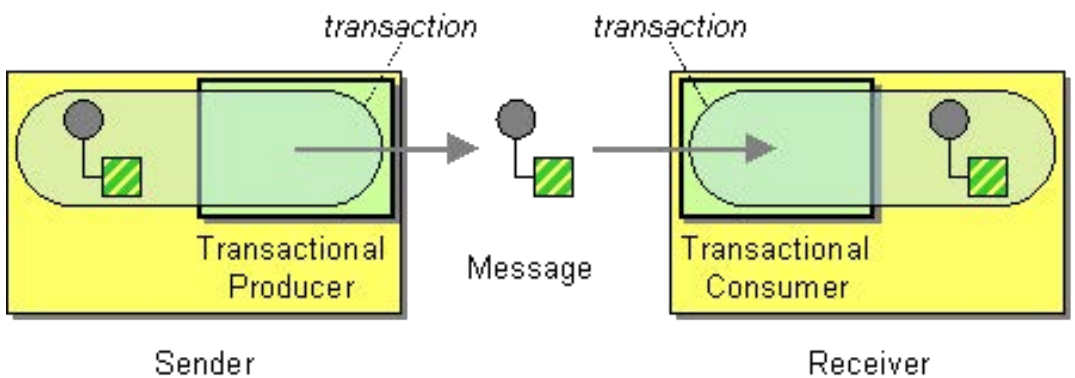



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Polling Consumer

- Necessary for an application
- Explicitly makes a call when it wants to receive a message
- Receiver thread blocks until a message is received
- Receiver polls for a message, processes it, then polls for another

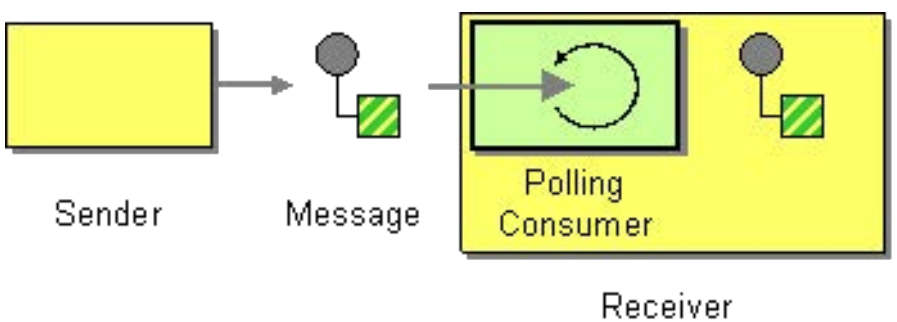



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Event-Driven Consumer

- Enables application to consume Messages as soon as they are delivered
- Application is automatically handed the messages as they are delivered on the channel
- Also known as an asynchronous receiver, because the receiver does not have a running thread until a callback thread delivers a message
- Receiver acts like the message delivery is an event that triggers the receiver into action

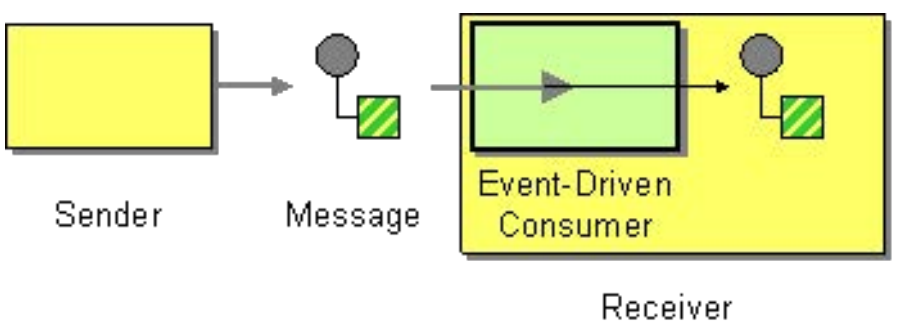



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Competing Consumers

- An application cannot process messages as fast as they are being added to the channel
- Create multiple Competing Consumers on a single channel so that the consumers can process multiple messages concurrently
- Competing Consumers are multiple consumers that are all created to receive messages from a single Point-to-Point Channel
- The messaging system's implementation determines which consumer actually receives the message, but in effect the consumers compete with each other to be the receiver
- Each consumer processes a different message concurrently, so the bottleneck becomes how quickly the channel can feed messages instead of how long a consumer takes to process a message

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Competing Consumers

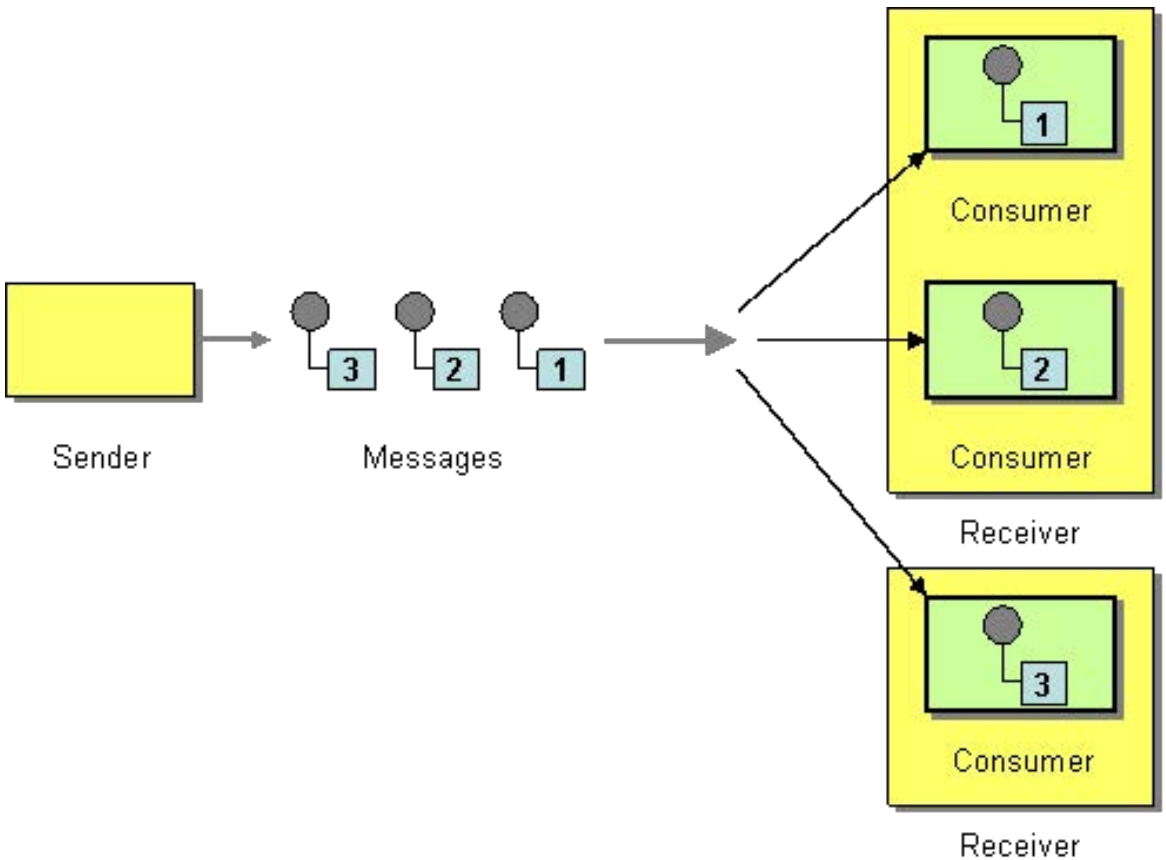



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Message Dispatcher

- The application which is using messaging need multiple consumers on a single channel to work in a coordinated fashion
- Message Dispatcher consumes messages from a channel and distribute them to performers
- A Message Dispatcher consists of two parts:
  - Dispatcher: Consumes message from a channel and forwards it to an available performer
  - Performer: Consumes message from the dispatcher to process it. A performer can delegate to the rest of its application to process its message

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Message Dispatcher

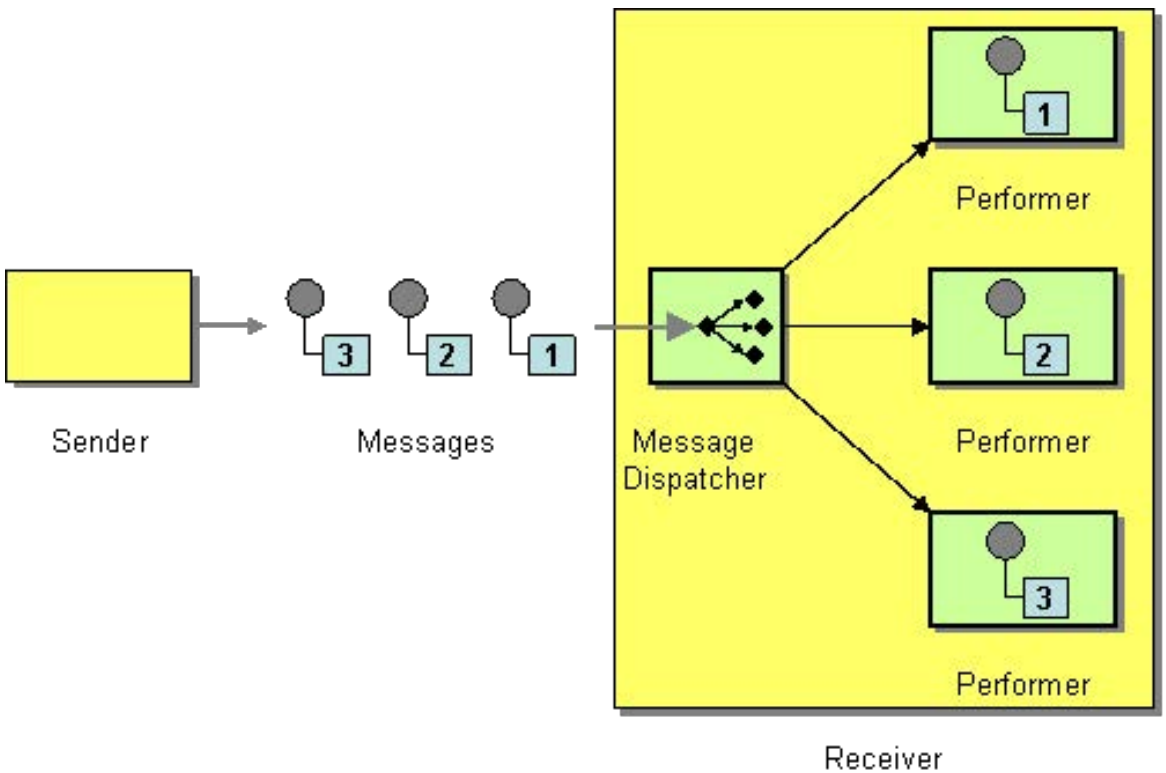



Image: Hohpe and Woolf

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

# Selective Consumer

- Selective Consumer filters the messages delivered by its channel so that it only receives the ones that match its criteria
- There are three parts to this filtering process:
  - Specifying Producer: Specifies the message's selection value before sending it
  - Selection Value: Allows the consumer to decide whether to select the message or not
  - Selective Consumer: Only receives messages that meet its selection criteria

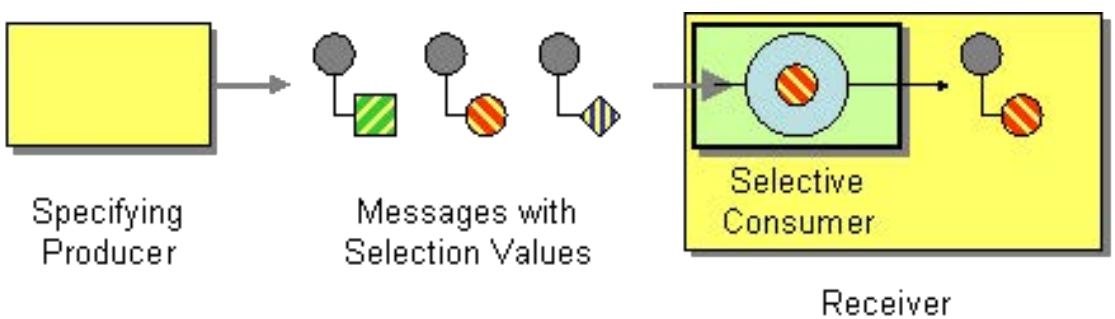



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Durable Subscriber

- Durable Subscriber makes the messaging system save messages published while the subscriber is disconnected
- A durable subscription saves messages for an inactive subscriber and delivers these saved messages when the subscriber reconnects
- Hence a subscriber will not loose any messages even though it disconnected

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Durable Subscriber

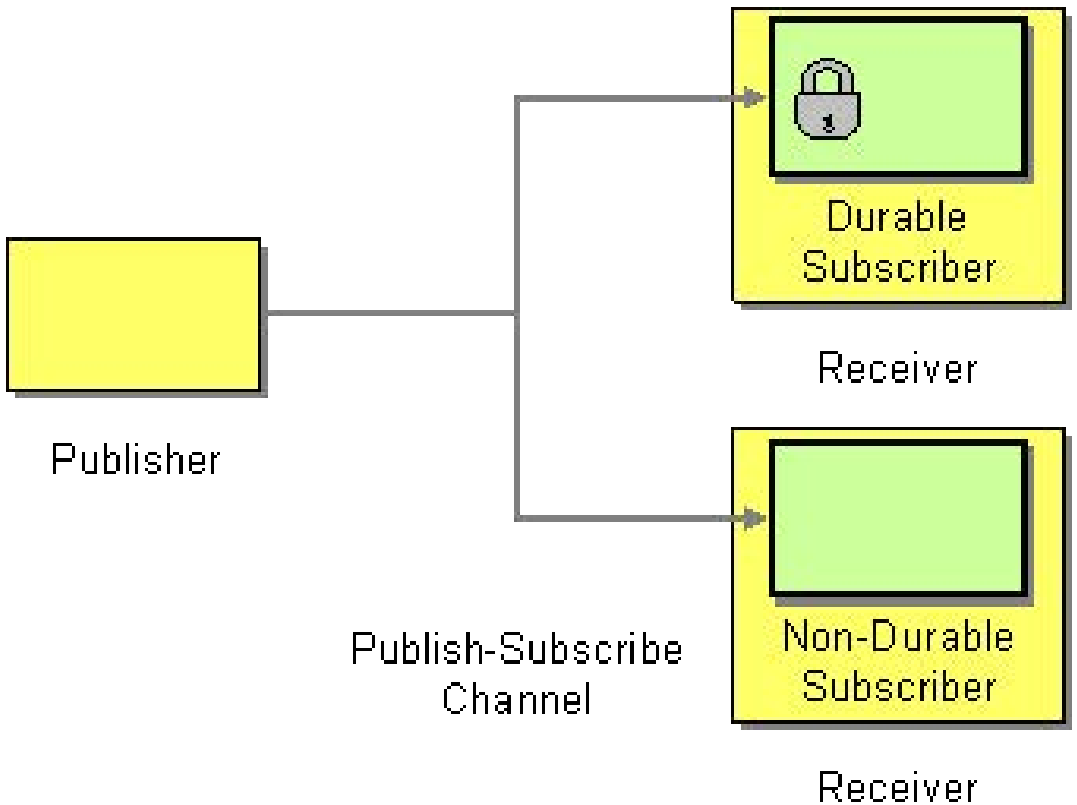



Image: Hohpe and Woolf

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Idempotent Receiver

- Idempotent function: $f(x) = f(f(x))$
- Even when a sender application only sends a message once, the receiver application may receive the message more than once
- Idempotent Receiver can safely receive the same message multiple times with the same effect
- A message can safely be resent without causing any problems even if the receiver receives duplicates of the same message

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

## Service Activator

- An application has a service that it would like to make available to other applications. How can an application design a service to be invoked both via various messaging technologies and via non-messaging techniques?
- Design a Service Activator that connects the messages on the channel to the service being accessed.
- A Service Activator can be one-way (request only) or two-way (Request-Reply).
- The activator can be hard-coded to always invoke the same service, or can use reflection to invoke the service indicated by the message.
- The activator handles all of the messaging details and invokes the service like any other client, such that the service doesn't even know it's being invoked through messaging.

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Service Activator

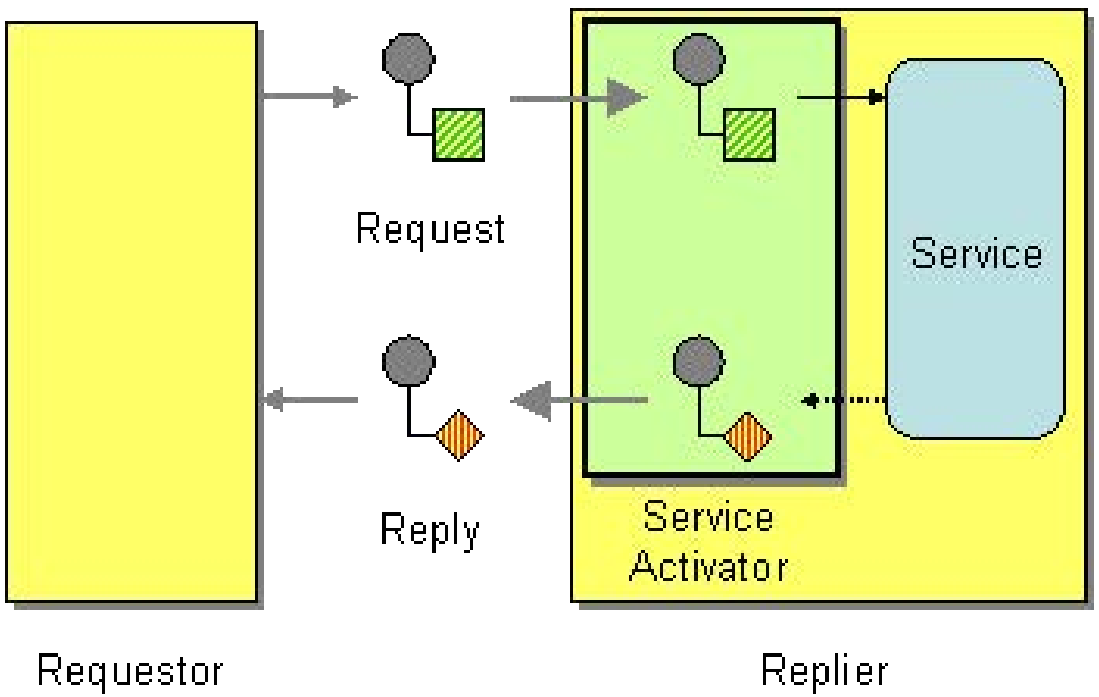



Image: Hohpe and Woolf

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. **Demo**
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. **System Management**
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# System Management

## Enterprise Integration Patterns

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 08, 2013

# System Management

For message-based integration solutions

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

System management is necessary due to following reasons:

- Thousands or even millions of messages
- Exceptions
- Performance bottlenecks
- Changes in the participating systems
- Many platforms and machines that can reside at multiple locations
- Testing and debugging harder due to loose coupling
- Messaging infrastructure typically guarantees the delivery of the message but not the delivery time

# System Management

## For message-based integration solutions

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- System management monitors:
  - How many messages are being sent?
  - How long it took a message to be processed?
- Observing and analyzing message traffic
- Message history for testing and debugging
- Analyzing all possible paths a message can take
- Analyzing the asynchronous flow of messages

# Control Bus

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we effectively administer a messaging system that is distributed across multiple platforms and a wide geographic area?**

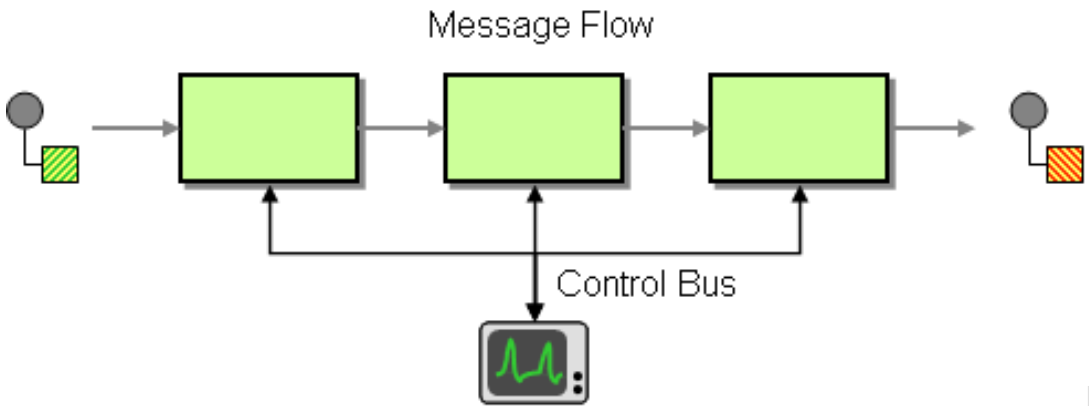



Image: Hohpe and Woolf

- Use a Control Bus to manage an enterprise integration system
- Control Bus uses the same messaging mechanism used by the application data
- Control Bus uses separate channels to transmit data

# Control Bus

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we effectively administer a messaging system that is distributed across multiple platforms and a wide geographic area?**

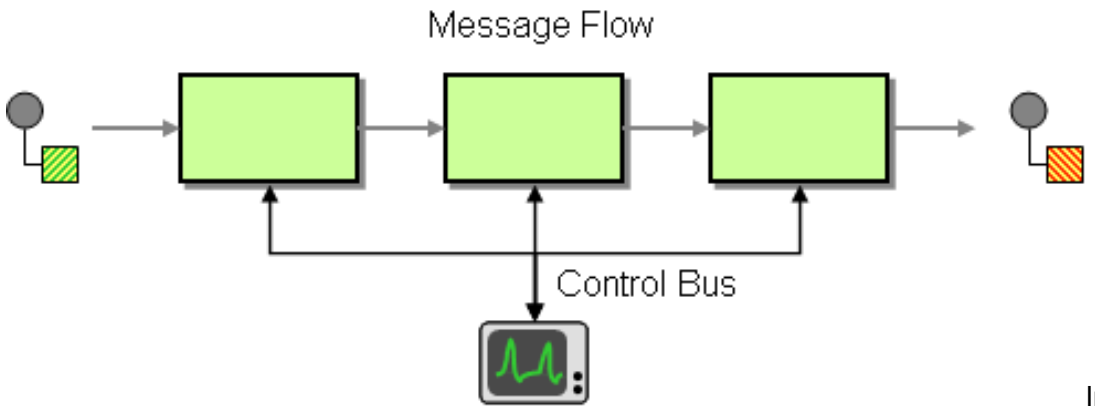



Image: Hohpe and Woolf

- Use a Control Bus to manage an enterprise integration system
- Control Bus uses the same messaging mechanism used by the application data
- Control Bus uses separate channels to transmit data

# Wire Tap

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How do you inspect messages that travel on a point-to-point channel?**

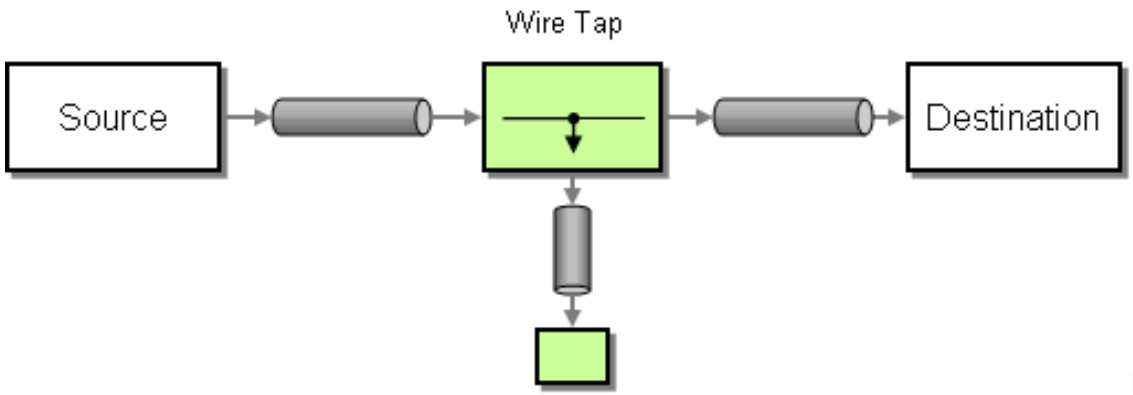



Image: Hohpe and Woolf

- Insert a simple Recipient List into the channel that publishes each incoming message to the main channel and a secondary channel.

# Wire Tap

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How do you inspect messages that travel on a point-to-point channel?**

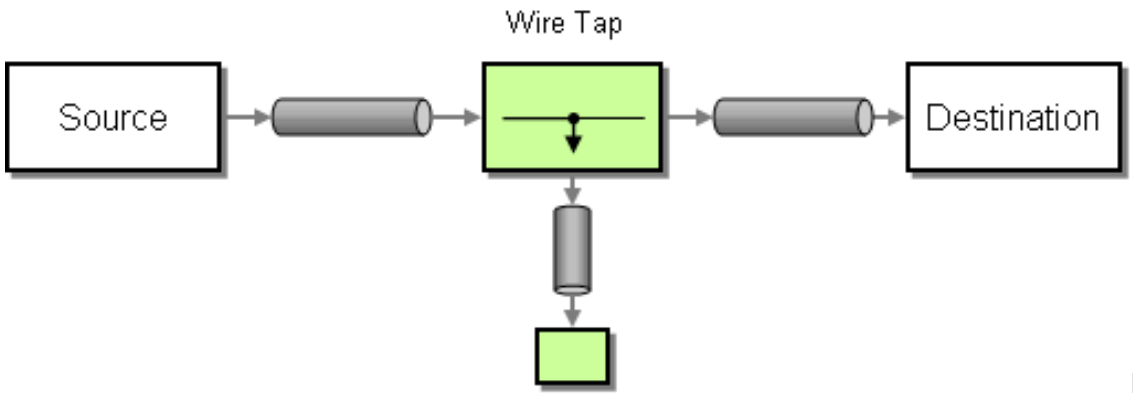



Image: Hohpe and Woolf

- Insert a simple Recipient List into the channel that publishes each incoming message to the main channel and a secondary channel.

# Message History



Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we effectively analyze and debug the flow of messages in a loosely coupled system? Can Control Bus be useful here?**

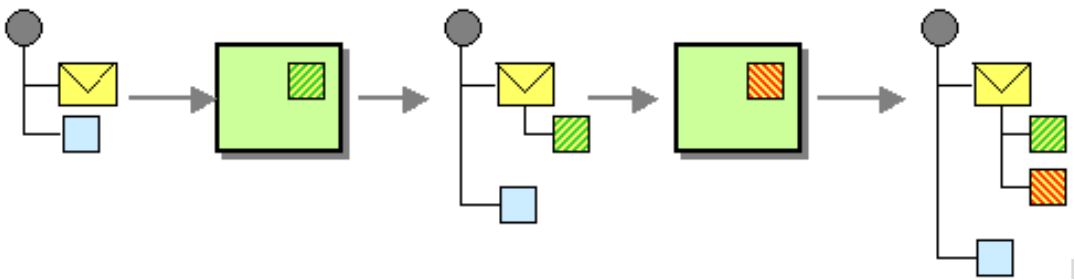


Image: Hohpe and Woolf

- Attach a Message History to the message
- The Message History is a list of all applications that the message passed through since its origination
- Useful if a series of messages flows through a number of filters

# Message History

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we effectively analyze and debug the flow of messages in a loosely coupled system? Can Control Bus be useful here?**

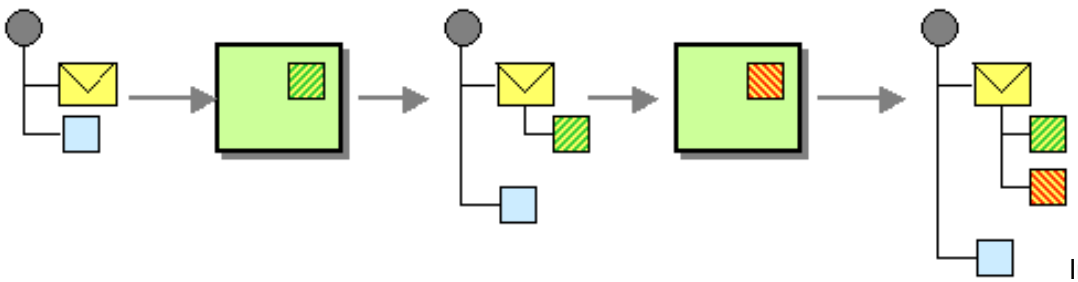


Image: Hohpe and Woolf

- Attach a Message History to the message
- The Message History is a list of all applications that the message passed through since its origination
- Useful if a series of messages flows through a number of filters

# Message Store

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we report against message information without disturbing the loosely coupled and transient nature of a messaging system? Why we can't use Message History?**

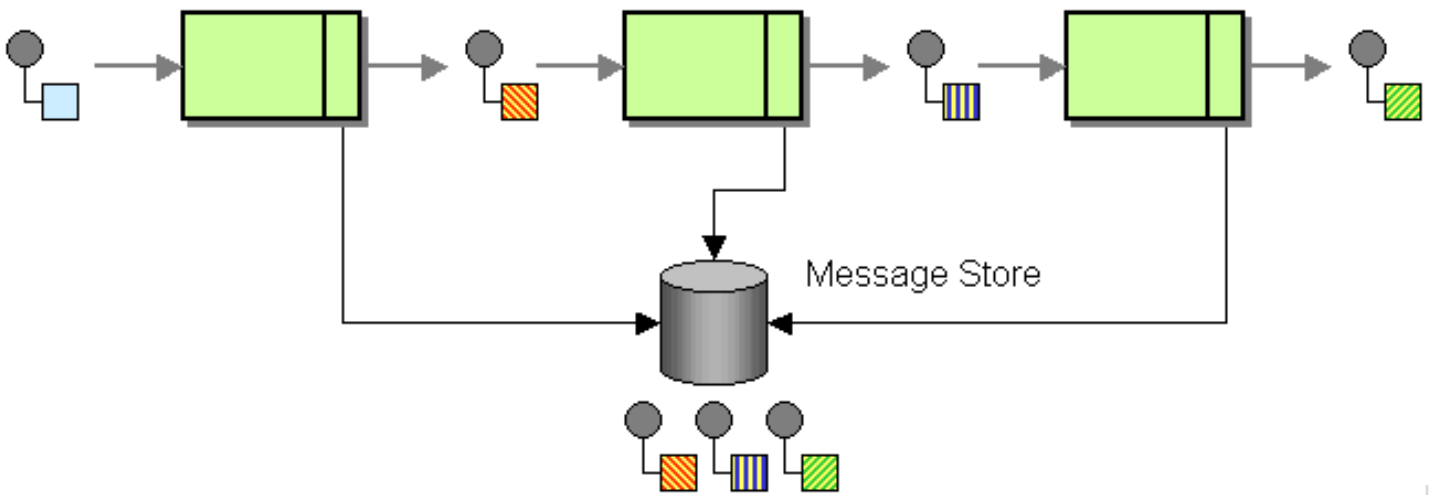



Image: Hohpe and Woolf

- Use a Message Store to capture information about each message in a central location.

# Message Store

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can we report against message information without disturbing the loosely coupled and transient nature of a messaging system? Why we can't use Message History?**

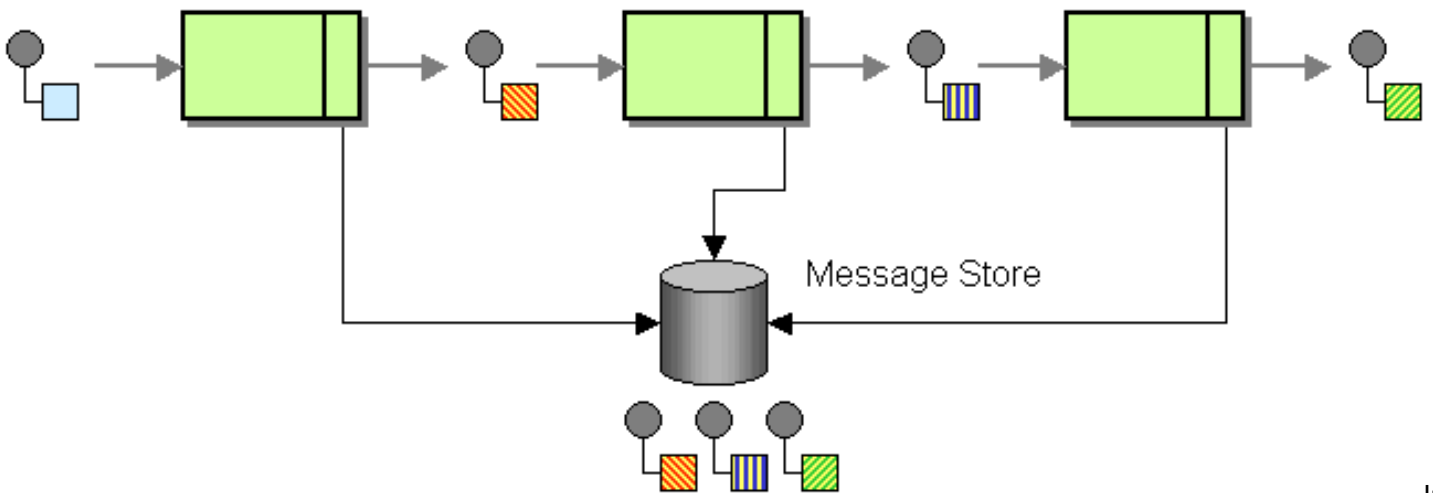



Image: Hohpe and Woolf

- Use a Message Store to capture information about each message in a central location.

# Smart Proxy

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can you track messages on a service that publishes reply messages to the Return Address specified by the requestor?**

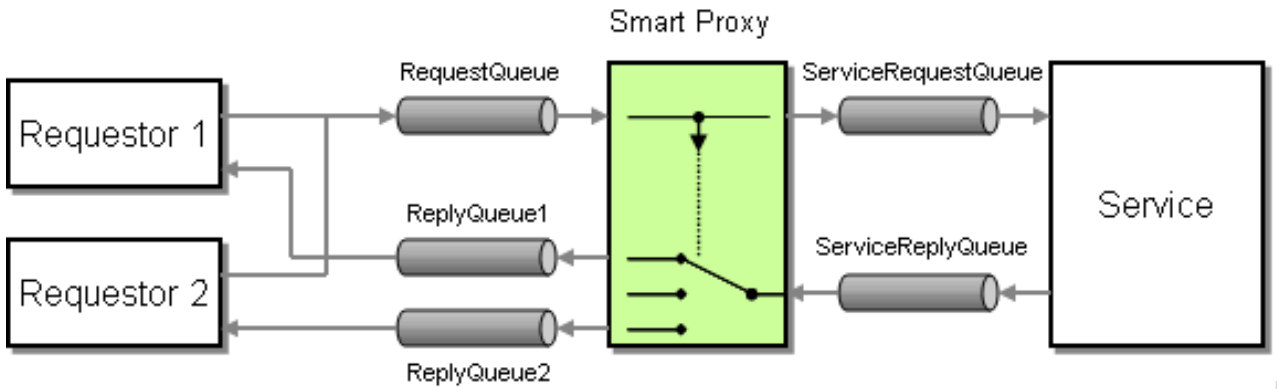



Image: Hohpe and Woolf

- Use a Smart Proxy to store the Return Address supplied by the original requestor
- Replace it with the address of the Smart Proxy
- When the service sends the reply message route it to the original Return Address

# Smart Proxy

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can you track messages on a service that publishes reply messages to the Return Address specified by the requestor?**

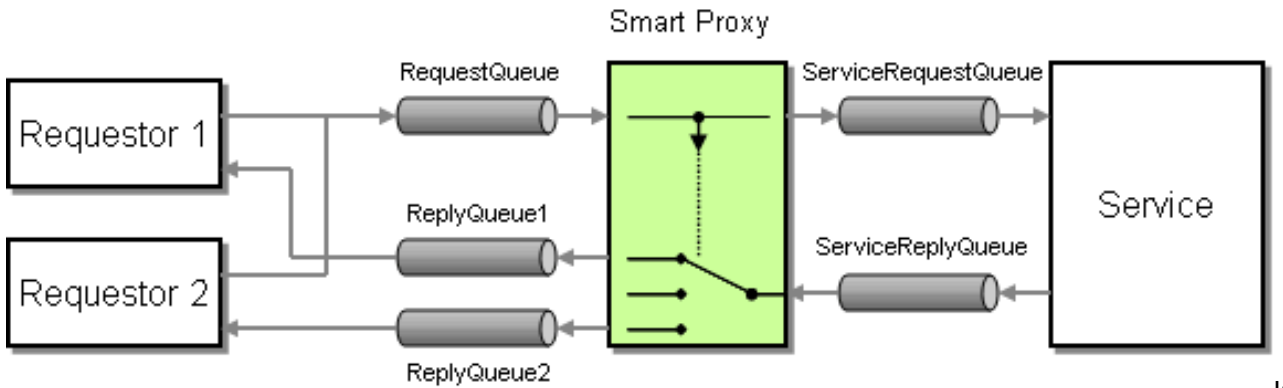



Image: Hohpe and Woolf

- Use a Smart Proxy to store the Return Address supplied by the original requestor
- Replace it with the address of the Smart Proxy
- When the service sends the reply message route it to the original Return Address

# Test Message

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**What happens if a component is actively processing messages, but garbles outgoing messages due to an internal fault?**

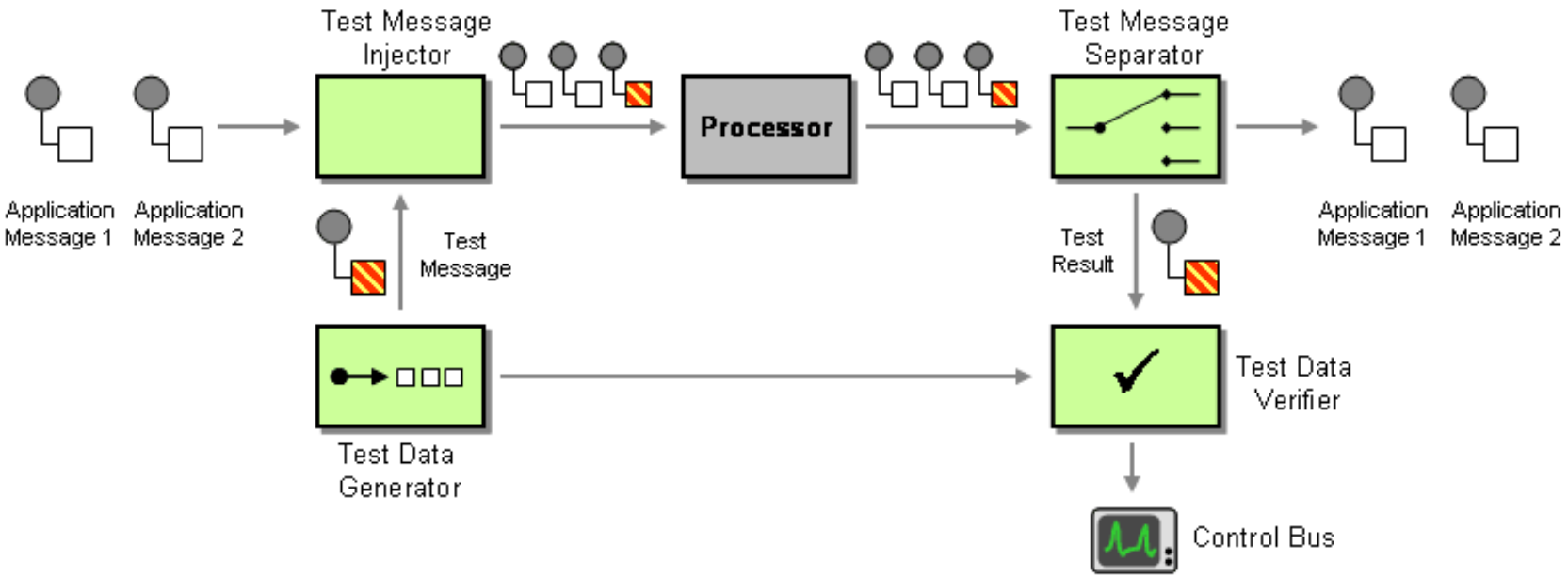



Image: Hohpe and Woolf

- Use Test Message to assure the health of message processing components

# Test Message

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**What happens if a component is actively processing messages, but garbles outgoing messages due to an internal fault?**

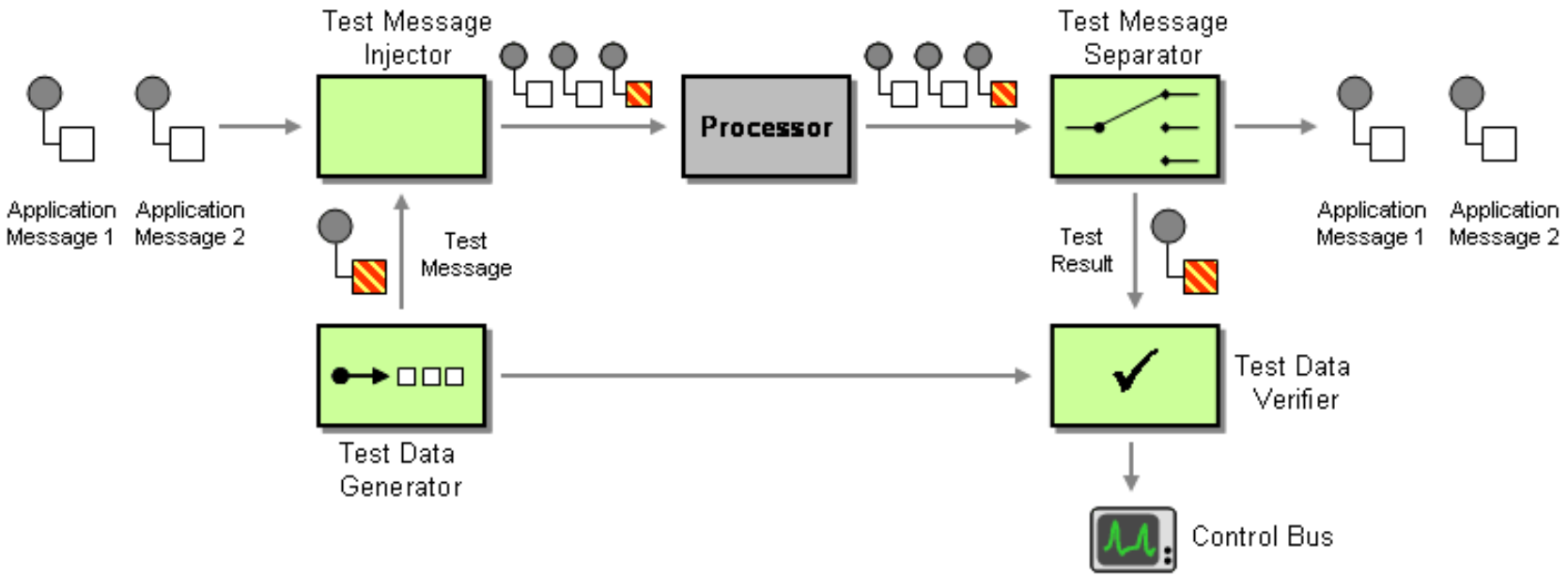


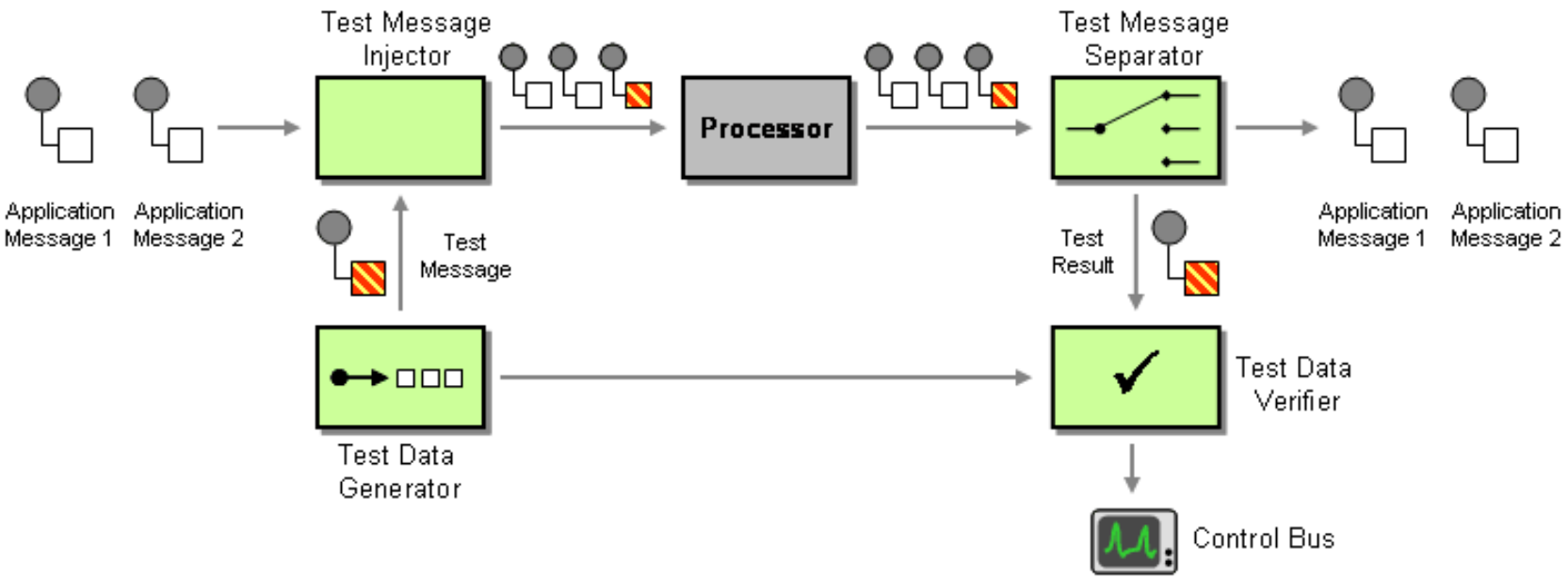

Image: Hohpe and Woolf

- Use Test Message to assure the health of message processing components

# Test Message (continued)

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

- Test Data Generator creates messages to be sent to the component for testing
- Test Message Injector inserts test data into the regular stream of data messages
- Test Message Separator extracts the results of test messages from the output stream
- Test Data Verifier compares actual results with expected results



# Channel Purger

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can you keep left-over messages on a channel from disturbing tests or running systems?**

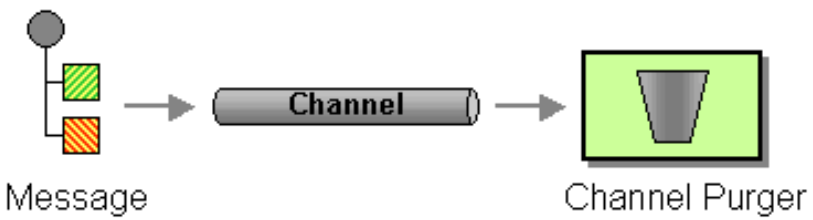



Image: Hohpe and Woolf

- Use a Channel Purger to remove unwanted messages from a channel.

# Channel Purger

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

**How can you keep left-over messages on a channel from disturbing tests or running systems?**

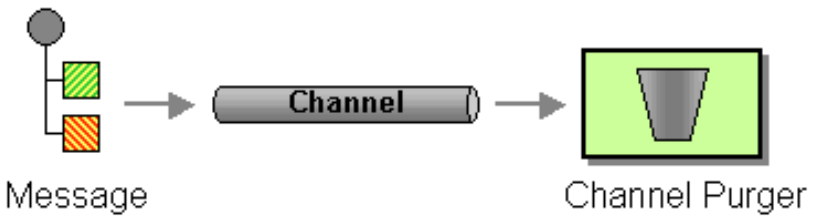



Image: Hohpe and Woolf

- Use a Channel Purger to remove unwanted messages from a channel.

# Selected References I

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

C. Emmersberger and F. Springer. Tutorial: open source enterprise application integration - introducing the event processing capabilities of apache camel. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, DEBS '13, pages 259–268, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1758-0.

G. Hohpe and B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004. ISBN 0-321-20068-3. URL `http://www.enterpriseintegrationpatterns.com/`.

D. S. Linthicum. *Enterprise Application Integration*. Addison-Wesley Professional, 2000. ISBN 0-201-61583-5.

X. Pan, W. Pan, and X. Cong. Soa-based enterprise application integration. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 7, pages 564–568, 2010.

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. **Demo**
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. **Cloud Integration**
11. Brief Introduction: Business Process Execution Language (BPEL)
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Cloud Integration

## EAI in the Cloud

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 08, 2013

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Why Cloud Integration?

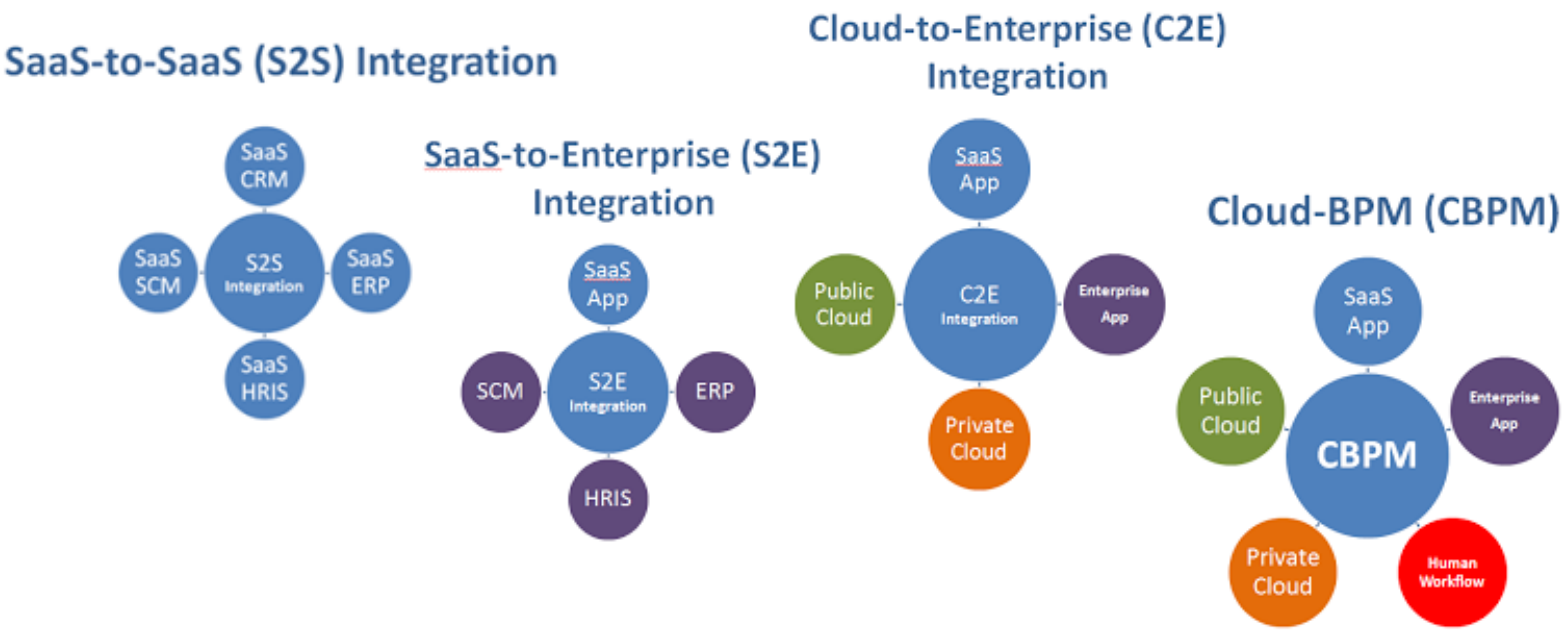
- increasing percentage of business software applications hosted in the cloud
  - SaaS applications like NetSuite or Salesforce
  - Cloud-based analytics platfoms
  - data moves to the Cloud as well
- cloud computing makes it much easier to consume services
- evolution of virtualization efforts to private clouds
- software packages built for the cloud are new
- access data in real time, anywhere, from any device
- scalability for future expansions

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## "Cloud integration is..."

*"...the provision of interfaces and data exchange between a hosted computing service and any other endpoint."* –Glenn Johnson [1]

- not just EAI in the cloud or sharing data between SaaS applications
- Multiple levels of cloud integration exist:



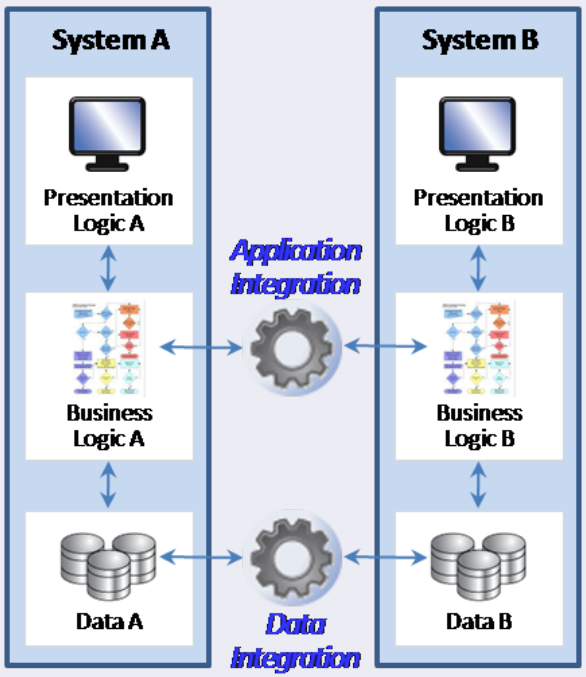

[1] [Johnson, 2010]

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Data vs. Application Integration

## Data Integration

- synchronizing data between different data sources and targets
- can involve a lot of data manipulation, reconciliation, de-duplication, cleansing, standardization and other data-intensive operations
- doesn't run hundreds or thousands of times an hour
- may involve a single record of data or Terabytes of data



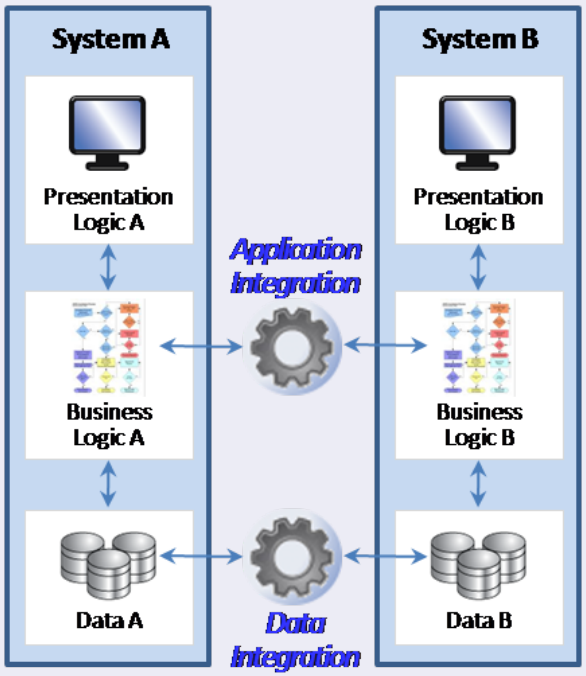

[Tibbetts, 2011]

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz

# Data vs. Application Integration

## Application Integration

- reliable and timely exchange of messages between applications
- typically operates at near real-time speeds
- typically less data involved in single integration task
- may involve many thousand integration tasks in an hour




[Tibbetts, 2011]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Cloudstreams: The next step of cloud integration?

*"...flexible, well defined integrations of services at the API level using policies to orchestrate the data, messages and invocations associated with those services."* –Daryl C. Plummer, Gartner Research [1]

- packaged cloud integration template
- description of everything necessary to govern, secure and manage interactions

[1] [Plummer, 2010]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Challenges of Cloud Integration - The beginners perspective

- secure and encrypted data transfer
- data replication (not necessary but may be required)
- frequently changing APIs that differ from each other
- consumers expect fast deployment of cloud integration solutions on many different levels
  - integrating services and information in clouds
  - security models across clouds and traditional environments
  - monitoring and management
- $\Rightarrow$ Hybrid Cloud Integration (*"No Cloud is an island."* –Marc-Thomas Schmidt, IBM)[1]

[1] [Schmidt, 2011]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Cloud Integration - The Cloud Adopters View[1]

**Top priorities:**

- security and manageability (80+ percent)
- cloud-to-cloud integration and better mobile access (75+ percent)
- enhancing existing cloud applications (65 percent)

**Biggest drivers:**

- business agility (25 percent)
- total cost of ownership (TCO) reduction (18 percent)
- exploiting the latest technology
- supporting mobile workers

[1] [Narasimhan and Nichols, 2011]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Already Existing Solutions

**Middleware Systems for Cloud Integration:** [1]

- IBM Altocumulus
  - deploy applications to a variety of clouds
  - supports EC2, Eucalyptus, Google AppEngine (GAE), IBM HiPODS
  - no hybrid application deployment across different clouds supported
- AppScale
  - implements GAE open APIs
  - interface for running GAE applications across "non-Google clouds"
  - offers hybrid cloud platform environment

[1] [Marpaung et al., 2013]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# Already Existing Solutions

**Middleware Systems for Cloud Integration (cont.):** [1]

- GigaSpaces Cloudify
  - designed to support any application regardless of the application stack
  - no code modifications necessary for migration
  - details required to run the application provided via recipes
  - supports Azure, EC2, CloudStack, OpenStack, and Rackspace - GoGrid and XenServer in development
- mOSAIC
  - open-source API for multi-Cloud oriented applications
  - follows RPC model - functionalities include marshalling, request/response correlation and error detection
  - uses semantic-oriented ontology for describing Cloud resources

[1] [Marpaung et al., 2013]

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Already Existing Solutions

**Inter-Cloud Architecture for Interoperability and Integration:** [1]

- based on existing standards in Cloud Computing
- multi-layer Cloud Services Model that combines commonly adopted cloud service models (IaaS, PaaS, SaaS)
- corresponding inter-layer interfaces
- still an on-going research project

[1] [Demchenko et al., 2012]



*"Cloud is very, very shiny today. Don't get distracted by it. ...Moving your business processes to the cloud is not the goal. Using the cloud to optimize business outcomes is the goal."* –Michele Cantara, Gartner Research [1]

[1] [Stuart, 2013]

# References I

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

Y. Demchenko, C. Ngo, M. Makkes, R. Stgrijkers, and C. de Laat. Defining inter-cloud architecture for interoperability and integration. In *The Third International Conference on Cloud Computing, GRIDs, and Virtualization (INTERCLOUD 2012)*, pages 174–180, Jul 2012.

G. Johnson. Cloud integration defined. can cloud bpm be far behind? Web, 08 2010. `http://it.toolbox.com/blogs/integrate-my-jde/cloud-integration-defined-can-cloud-bpm-be-far-behind-40537`.

J. Marpaung, M. Sain, and H.-J. Lee. Survey on middleware systems in cloud computing integration. In *15th International Conference on Advanced Communication Technology (ICACT), 2013*, pages 709–712, Jan 2013.

B. Narasimhan and R. Nichols. State of clous applications and platforms: The cloud adoptors' view. *Computer*, 44(3):24–28, Mar 2011. doi: 10.1109/MC.2011.66.

# References II

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

D. C. Plummer. Cloudstreams: The next cloud integration challenge. Web, 11 2010. http://blogs.gartner.com/daryl_plummer/2010/11/08/cloudstreams-the-next-cloud-integration-challenge/.

M.-T. Schmidt. Hybrid cloud integration or no cloud is an island. Web, 10 2011. https://www.ibm.com/developerworks/community/blogs/mts_on_soa/entry/hybrid_cloud_integration.

A. Stuart. Insights from gartner's 2013 bpm summit. Web, 04 2013. http://www.ebizq.net/blogs/bpminaction/2013/04/insights_from_gartners_2013_bp.php.

H. Tibbetts. Application integration in the cloud - finally here. Web, 12 2011. http://www.ebizq.net/blogs/integrationedge/2011/12/application-integration-in-the-cloud---finally-here.php.

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. **Brief Introduction: Business Process Execution Language (BPEL)**
12. Demo

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

# BPEL: Business Process Execution Language

**Jörg Lässig, Markus Ullrich**

Department of Computer Science

July 8, 2013

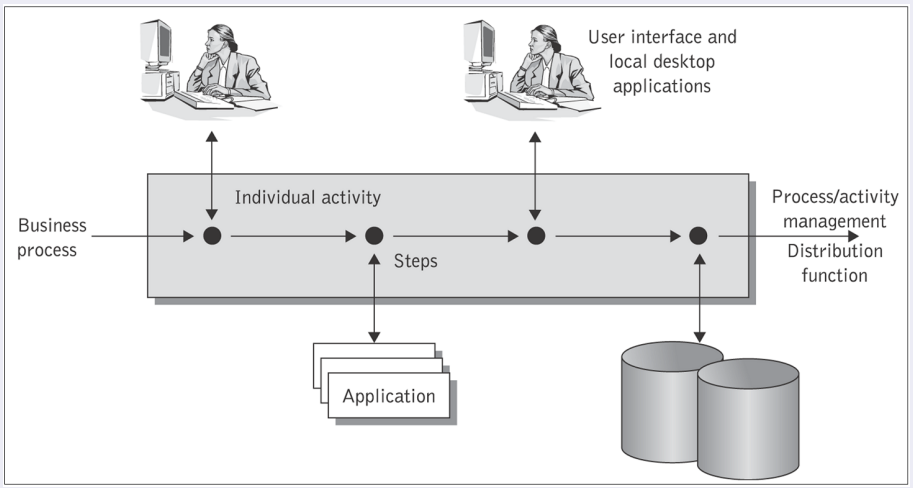
# Topics

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Business processes
- Workflows
- Web services orchestration and choreography
- The Business Process Execution Language

# Business processes

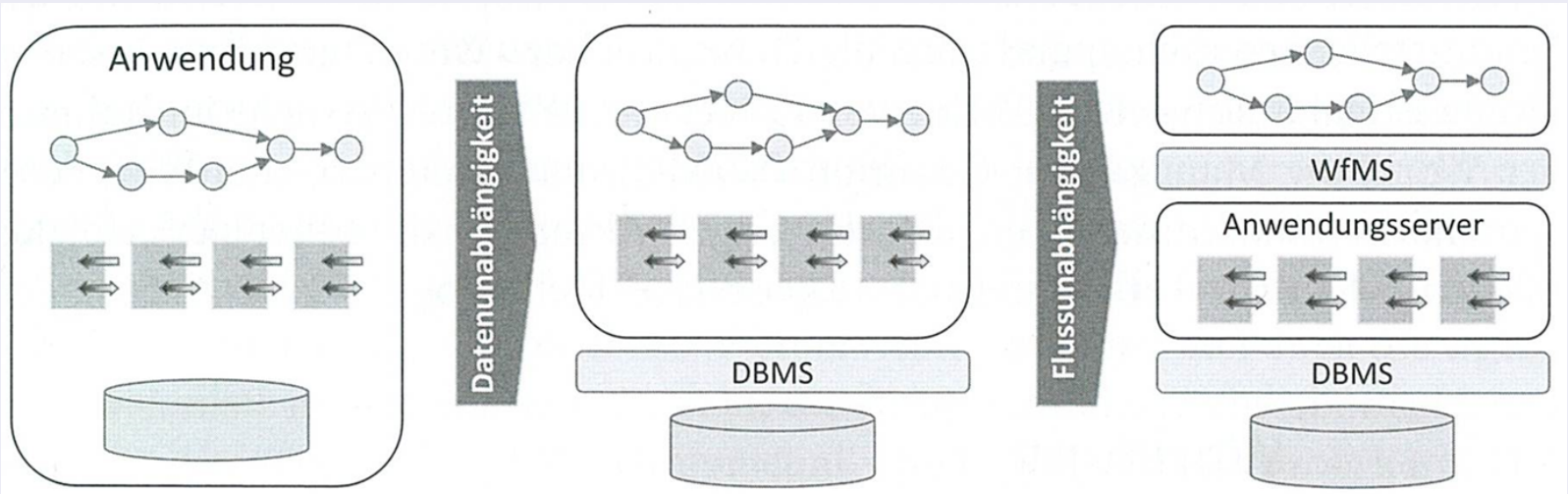
**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**




- A process is a sequence of steps that:
  - is initiated by an event,
  - transforms its inputs (in terms information, materials, or business commitments),
  - produces a specified output.
- A business process is a set of logically related tasks performed to achieve a well-defined business outcome.
  - It implies a horizontal view on an organization that looks at processes as sets of interdependent activities designed & structured to produce a specific output for a customer or a market.
  - It defines the results to be achieved, the context of the activities, the relationships between the activities & the interactions with other processes and resources.

# Business processes

Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz




- In the 1960s DBMS have been introduced
- Workflow Technology has been developed in the 1990s
- Distinction between programming in the large (business logic) and programming in the small (implementation of business functions)

# Topics

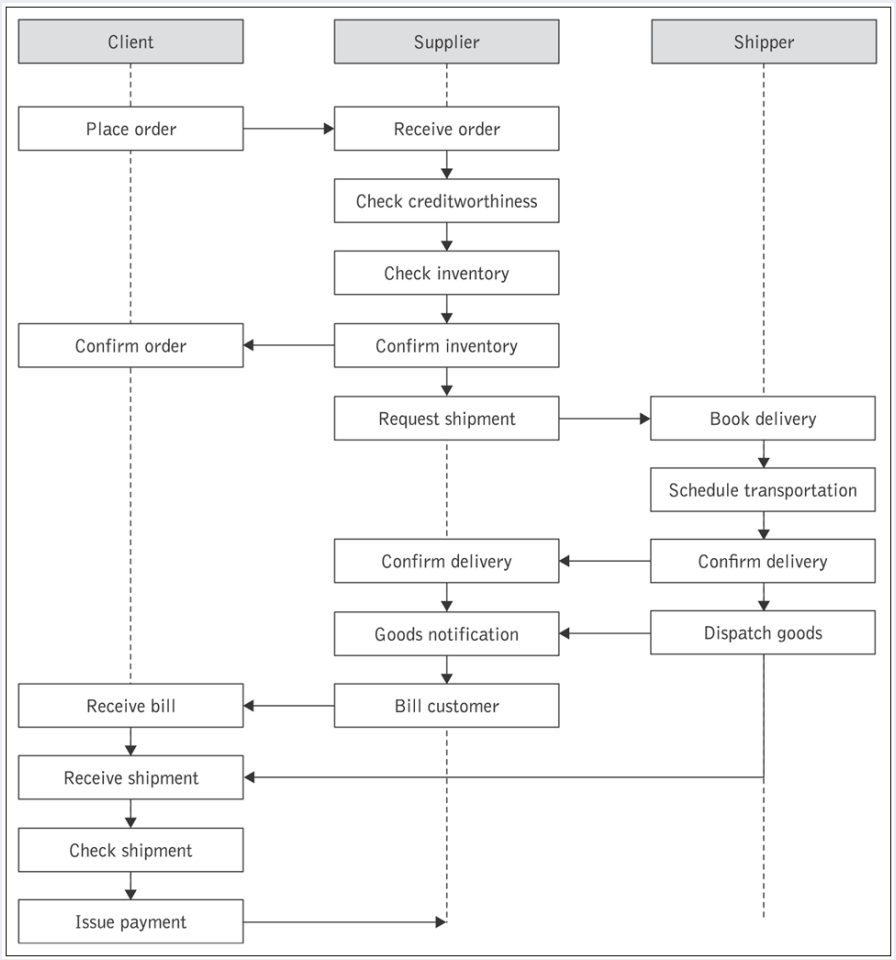
**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Business processes
- Workflows
- Web services orchestration and choreography
- The Business Process Execution Language

# Process-oriented workflows

**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz

- Process-oriented workflows are used to automate processes whose structure are well defined and stable over time.
- A process-oriented workflow can depict various aspects of a business process, e.g., automated and manual activities, decision points and business rules, parallel and sequential work routes, and how to manage exceptions to the normal business process.
  - An order management process or a loan request is an example of a well-defined process.
  - Certain process-oriented workflows may have transactional properties.



# Topics

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Business processes
- Workflows
- Web services orchestration and choreography
- The Business Process Execution Language

# Orchestration vs. Choreography

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Orchestration

- An executable business process describing a flow from the perspective and under control of a single endpoint (commonly: Workflow)
- BPEL handles Orchestration

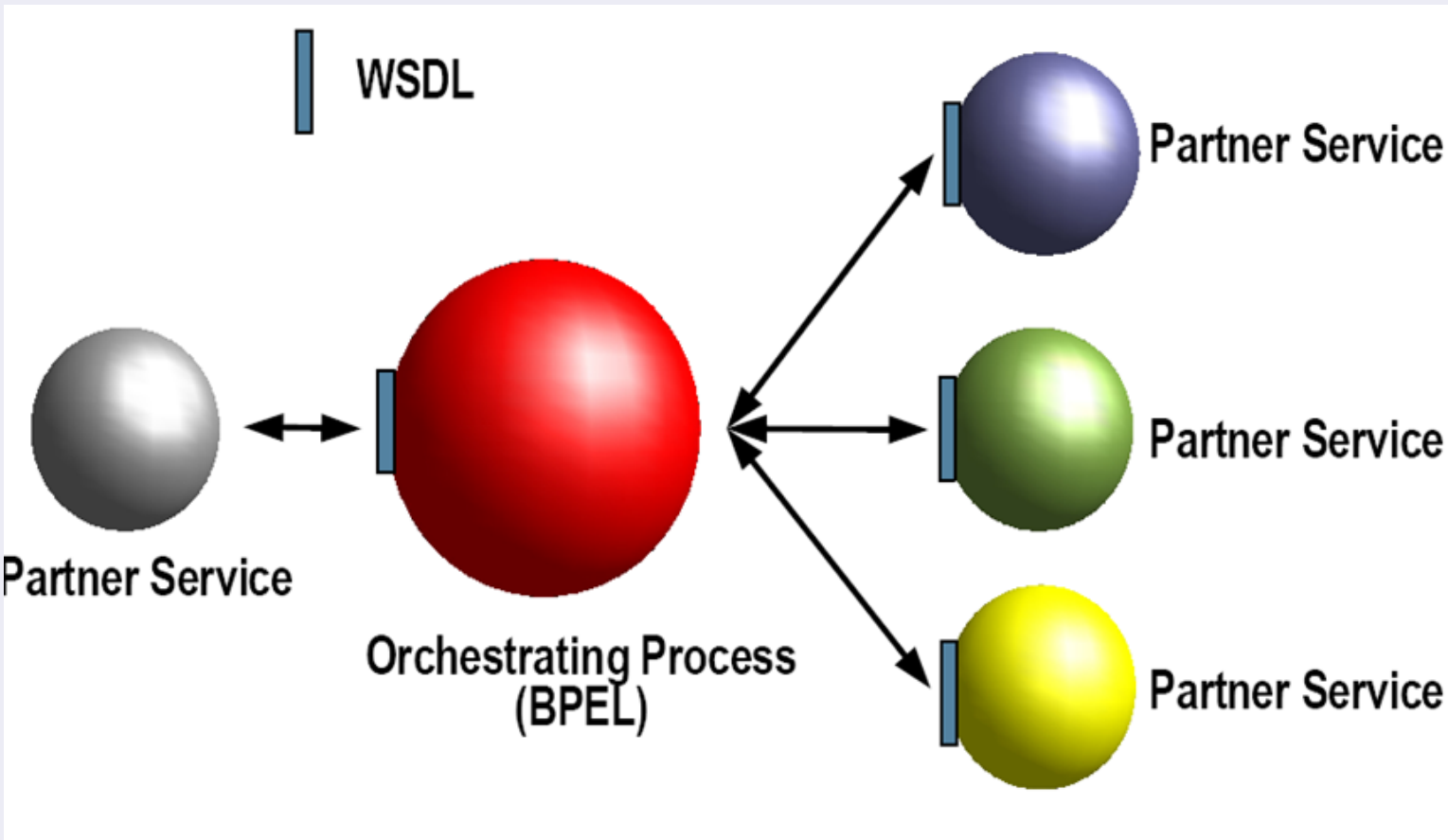
## Choreography (WSDL)

- The observable public exchange of messages, rules of interaction and agreements between two or more business process endpoints
- WSDL handles Choreography

# BPEL: Relationship to Partners



Enterprise Application Development Group
University of Applied Sciences Zittau/Görlitz



# Topics

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Business processes
- Workflows
- Web services orchestration and choreography
- The Business Process Execution Language

# BPEL is a Web Service Sequencing Language

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Process defines “conversation” flow chart
  - Conversation consists of only WSDL-described message exchanges
  - BPEL provides and consumes WSDL defined services
- Process instance is a particular conversation following the chart
  - Execution systems can support multiple concurrent conversations

# Business Process Execution Language

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- BPEL as a service composition (orchestration) language facilitates the modeling and execution of business processes based on Web services. It:
  - models business process collaboration (<partnerLink>s);
  - models the execution control of business processes;
  - supports fault handling & compensation;
  - supports service composability (structured activities can be nested and combined arbitrarily);
  - supports context (<scope> mechanism);
  - spawns off & synchronizes processes (<pick> & <receive> activities);
  - supports event-handling.

# BPEL Activities

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

## Basic Activities

- <invoke>
- <receive>
- <reply>
- <assign>
- <throw>
- <wait>
- <empty>

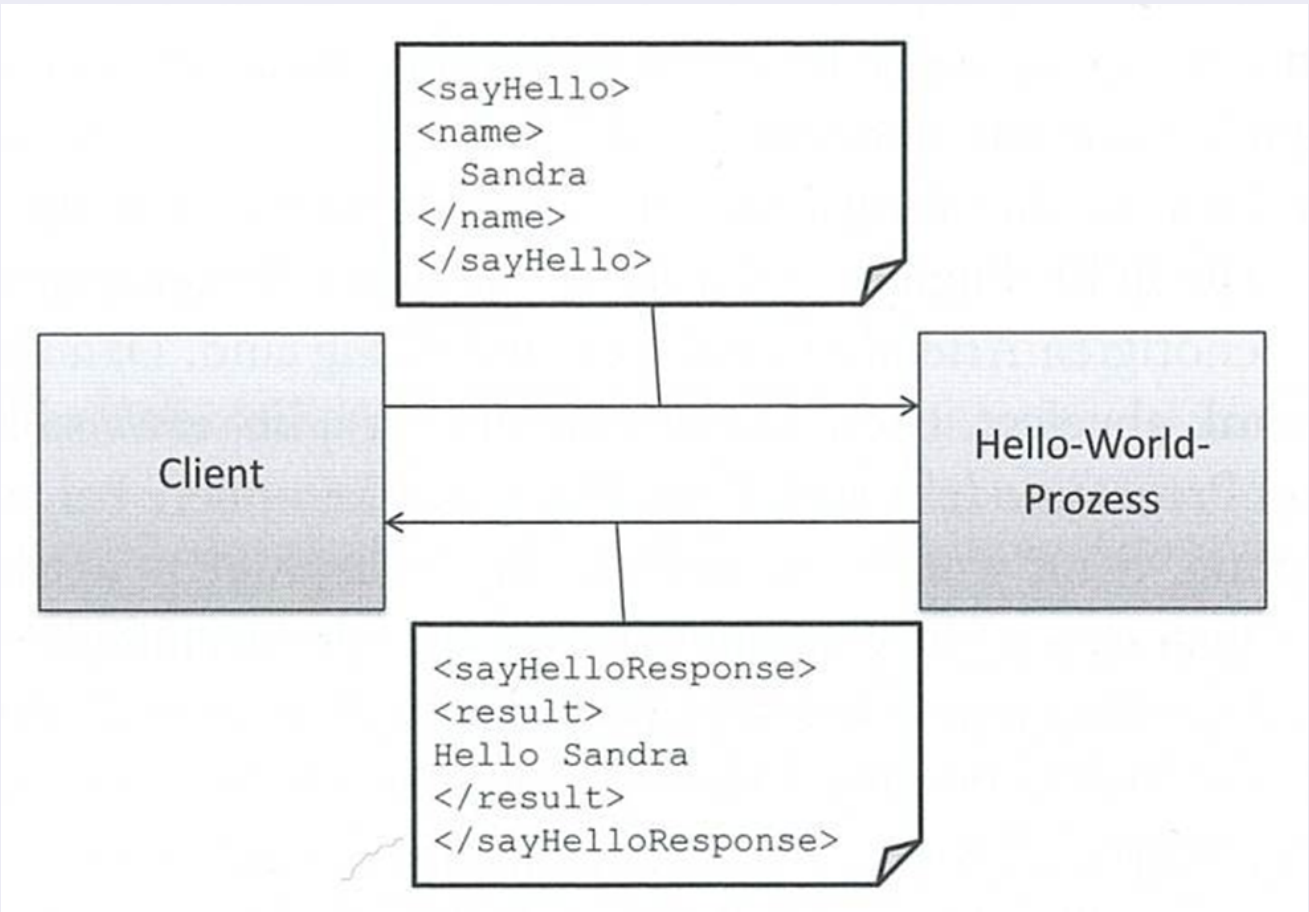
## Structured Activities

- <sequence>
- <while>
- <flow>
- <scope>
- <compensate>
- <switch>
- <link>

# Hello World Example



**Enterprise Application Development Group**
University of Applied Sciences Zittau/Görlitz



```
<wsdl:definitions name="HelloWorld"
  targetNamespace="http://www.bpelbuch.de/helloworld"
  xmlns:tns="http://www.bpelbuch.de/helloworld"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <wsdl:types>
    <xs:schema attributeFormDefault="unqualified"
       elementFormDefault="qualified"
       targetNamespace="http://www.bpelbuch.de/helloworld"
       xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="sayHello">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="sayHelloResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="result" type="string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="sayHelloMessage">
    <wsdl:part name="parameters" element="tns:sayHello" />
  </wsdl:message>
  <wsdl:message name="sayHelloResponseMessage">
    <wsdl:part name="parameters"
        element="tns:sayHelloResponse" />
  </wsdl:message>

  <wsdl:portType name="HelloWorld">
    <wsdl:operation name="sayHello">
      <wsdl:input message="tns:sayHelloMessage" />
      <wsdl:output message="tns:sayHelloResponseMessage" />
```

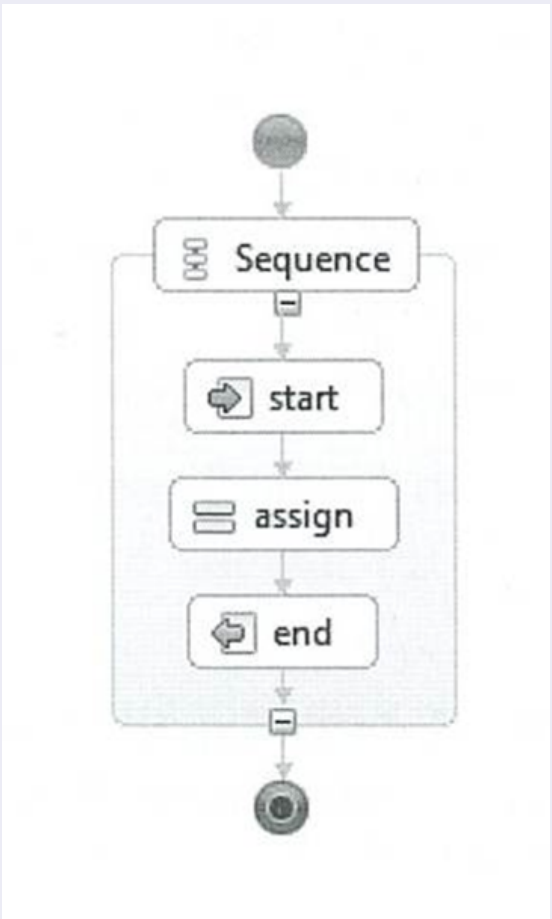
- WSDL with partnerLinkType
- Defines the relationsship between the process and its partner services
- partnerlinkType has role as portType

```
    </wsdl:operation>
  </wsdl:portType>

  <plnk:partnerLinkType name="HelloWorld">
    <plnk:role name="HelloWorldProvider"
          portType="tns:HelloWorld" />
  </plnk:partnerLinkType>
</wsdl:definitions>
```

# Process Model in Eclipse BPEL Designer

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**




- Process model with three sequential steps
- Visualization of an XML document (see next slide)

```
<bpel:process
  name="HelloWorld"
  targetNamespace="http://www.bpelbuch.de/helloworld"
  suppressJoinFailure="yes"
  xmlns:tns="http://www.bpelbuch.de/helloworld"
  xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
      executable">

  <bpel:import location="HelloWorld.wsdl"
      namespace="http://www.bpelbuch.de/helloworld"
      importType="http://schemas.xmlsoap.org/wsdl/" />

  <bpel:partnerLinks>
    <bpel:partnerLink name="client"
        partnerLinkType="tns:HelloWorld"
        myRole="HelloWorldProvider" />
  </bpel:partnerLinks>

  <bpel:variables>
    <bpel:variable name="input"
        messageType="tns:sayHelloMessage" />
    <bpel:variable name="output"
        messageType="tns:sayHelloResponseMessage" />
  </bpel:variables>

  <bpel:sequence name="main">
    <bpel:receive name="Start"
        partnerLink="client" portType="tns:HelloWorld"
        operation="sayHello" variable="input"
        createInstance="yes" />

    <bpel:assign validate="no" name="Assign">
      <!-- Variable initialisieren -->
      <bpel:copy>
        <bpel:from>
          <bpel:literal>
            <tns:sayHelloResponse
                xmlns:tns="http://www.bpelbuch.de/helloworld">
              <tns:result />
            </tns:sayHelloResponse>
          </bpel:literal>
        </bpel:from>
        <bpel:to part="parameters" variable="output" />
      </bpel:copy>
      <!-- Strings zusammenführen -->
      <bpel:copy>
        <bpel:from>
          concat('Hello ', $input.parameters/tns:name)
        </bpel:from>
        <bpel:to>$output.parameters/tns:result</to>
      </bpel:copy>
    </bpel:assign>

    <bpel:reply name="Ende" partnerLink="client"
        portType="tns:HelloWorld"
        operation="sayHello" variable="output" />
  </bpel:sequence>
</bpel:process>
```

- Process element is root
- name and targetNamespace as identifier
- Import for external artifacts
- partnerLinks (here just one interface offered)

# Test Setup

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

- Apache ODE
- Apache Tomcat
- Eclipse BPEL Designer
- BPELUnit

# Literature

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**


- Michael P. Papazoglou, Web Services: Principles and Technology, Pearson Prentice Hall, 2008
- Martin Kalin, Java Web Services - Up and Running, O'Reilly, 2009
- Will Iverson, Real World Web Services, O'Reilly, 2005
- Leonard Richardson, Sam Ruby, RESTful Web Services, O'Reilly, 2007
- Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, Web Services - Concepts, Architectures and Applications, Springer, 2010
- Oliver Heuser, Andreas Holubek, Java Web Services in der Praxis, dpunkt.Verlag, 2010
- Eben Hewitt, Java SOA Cookbook, O'Reilly, 2009
- Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, John Domingue, Enabeling Semantic Web Services, Springer, 2007
- Steve Graham, Doug Davis, Simeon Simeonov, Glen Daniels, Peter Brittenham, Yuichi Nakamura, Paul Fremantle, Dieter König, Claudia Zentner, Building Web Services with Java, Sams Publishing, 2005
- Tammo van Lessen, Daniel Lübke, Jörg Nitzsche, Geschäftsprozesse automatisieren mit BPEL, dpunkt.verlag, 2011.

# Questions and Discussion

**Enterprise Application Development Group**
**University of Applied Sciences Zittau/Görlitz**

Thank you very much for your attention!

Any questions?

# Structure

1. Introduction: Group / SOA+ESBs / Enterprise Application Integration
2. Messaging: Message Channels
3. Demo
4. Messaging: Message Construction / Message Routing
5. Demo
6. Messaging: Message Transformation / Message Endpoints
7. Demo
8. System Management
9. Demo
10. Cloud Integration
11. Brief Introduction: Business Process Execution Language (BPEL)
12. **Demo**

**Special thanks to Abhishek Awasthi, Nico Dittmann and Waheed Aslam Ghumman who helped preparing those slides.**