

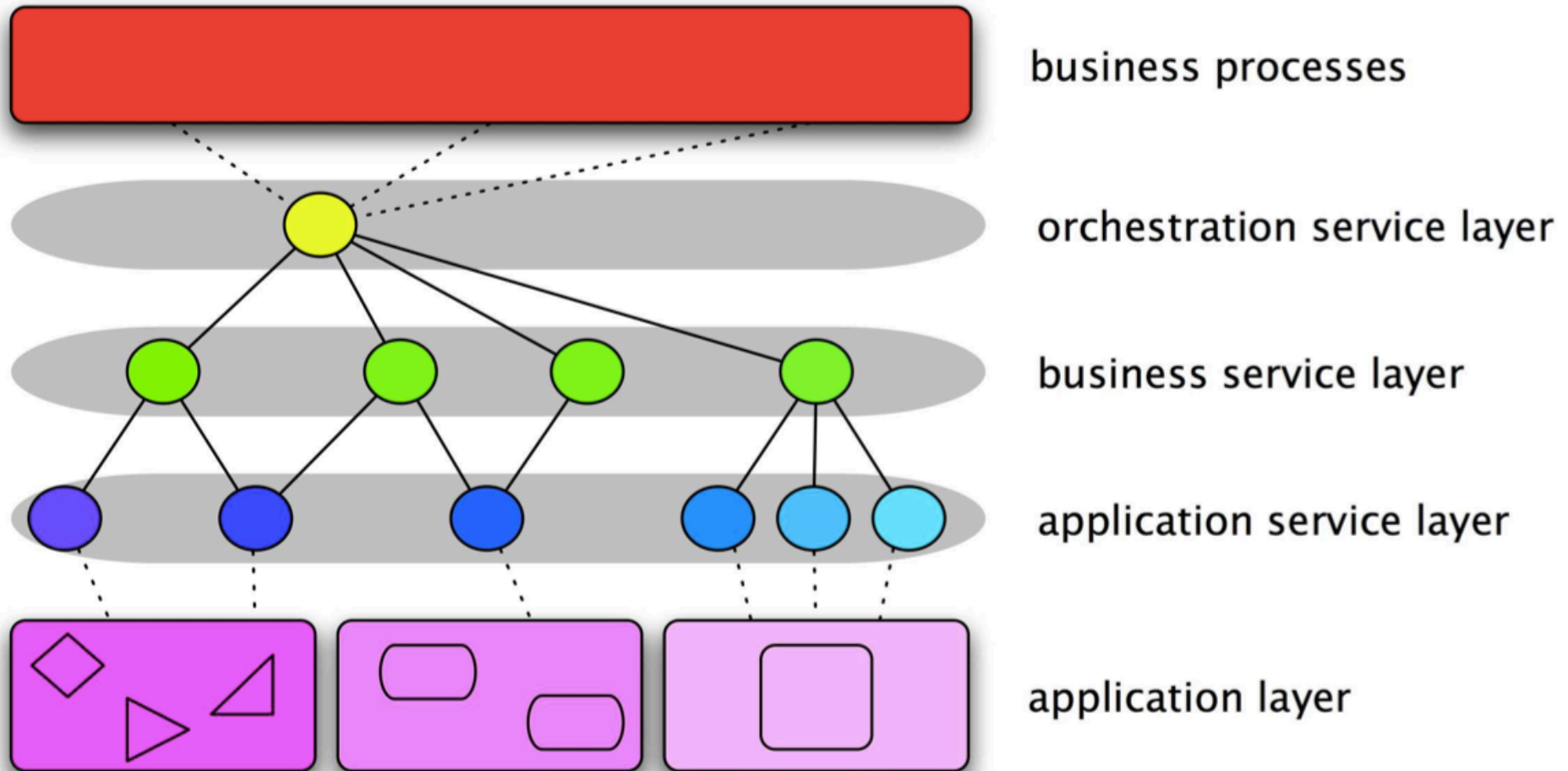
SOA integration and mediation

Oxford University
Software Engineering
Programme
May 2017



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Recap on SOA model

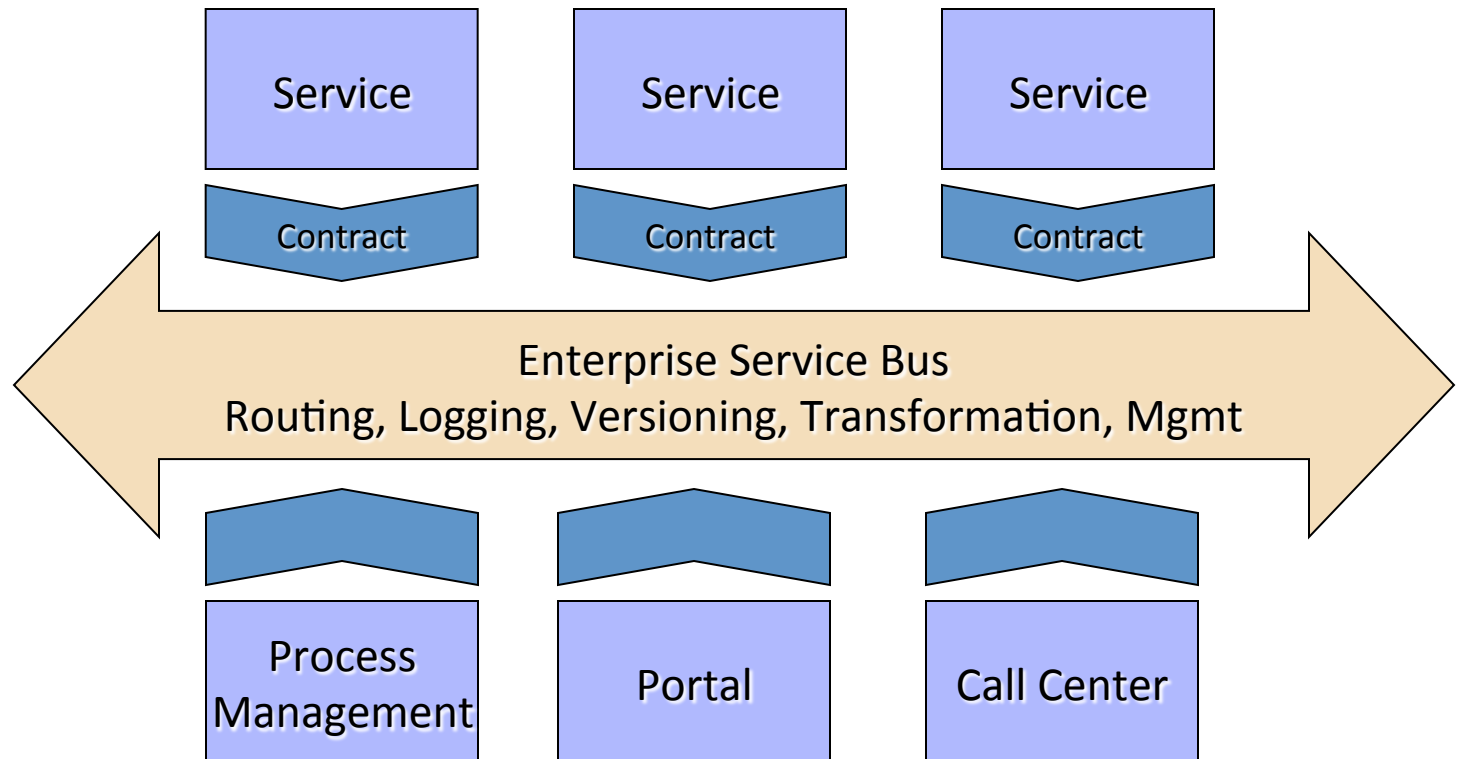


Enterprise Service Bus (ESB)

- A software architecture
 - A logical intermediary through which every message flows
 - Offers a policy based approach to decide what to do to each message or interaction
- The benefits of the gateway model
 - Without a physical hub and spoke
- Many vendors offer ESB products
 - Often a layer over an existing messaging framework



ESB as the implementation of SOA



Different approaches

- Point to Point
- Traditional EAI
- ESB
- Event Driven Architecture



Pros and Cons of an ESB

Pros

- Faster and cheaper accommodation of existing systems
- Increased flexibility: easier to change as requirements change
- Standards-based
- Scales to enterprise wide deployment
- Configuration rather than coding
- No central broker

Cons

- May end up with a proprietary solution
 - no common standards for the overall config and policies yet
- Requires more hardware to run
- New skills to learn to configure ESB
- Hard to get ROI on a small number of projects



ESB options

- **Proprietary**
 - IBM, Oracle, Tibco, SAP
- **Open Source**
 - Mule, Fuse, WSO2
 - Apache ServiceMix, Apache Synapse, Apache Camel



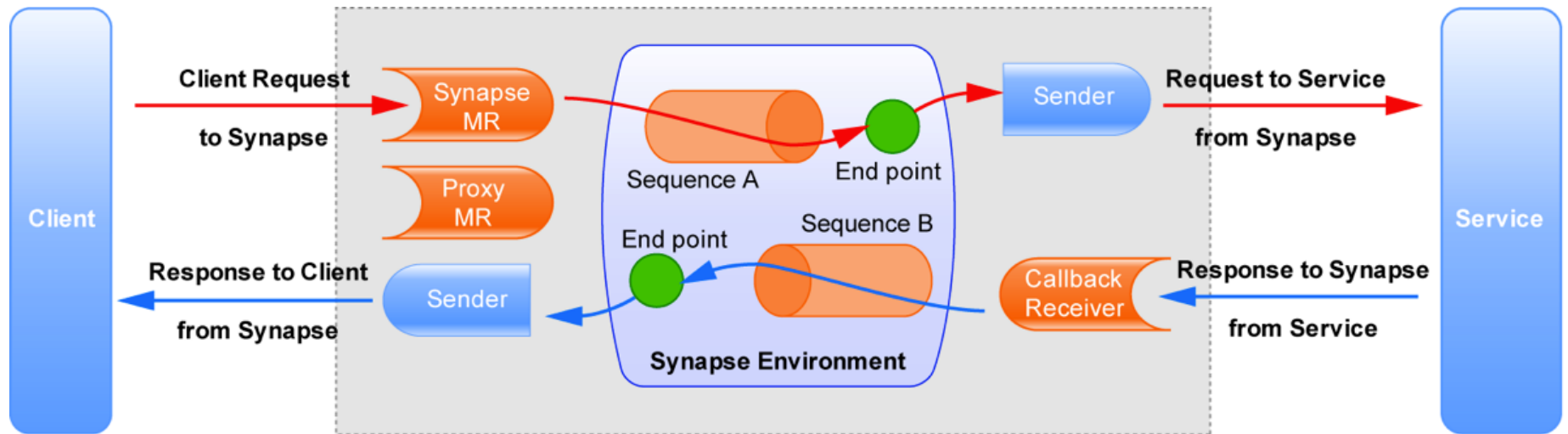
ESB models

- Almost all ESBs work on the same principle
- Message arrives
- Sequence of actions (Pipeline)
- Message is sent on

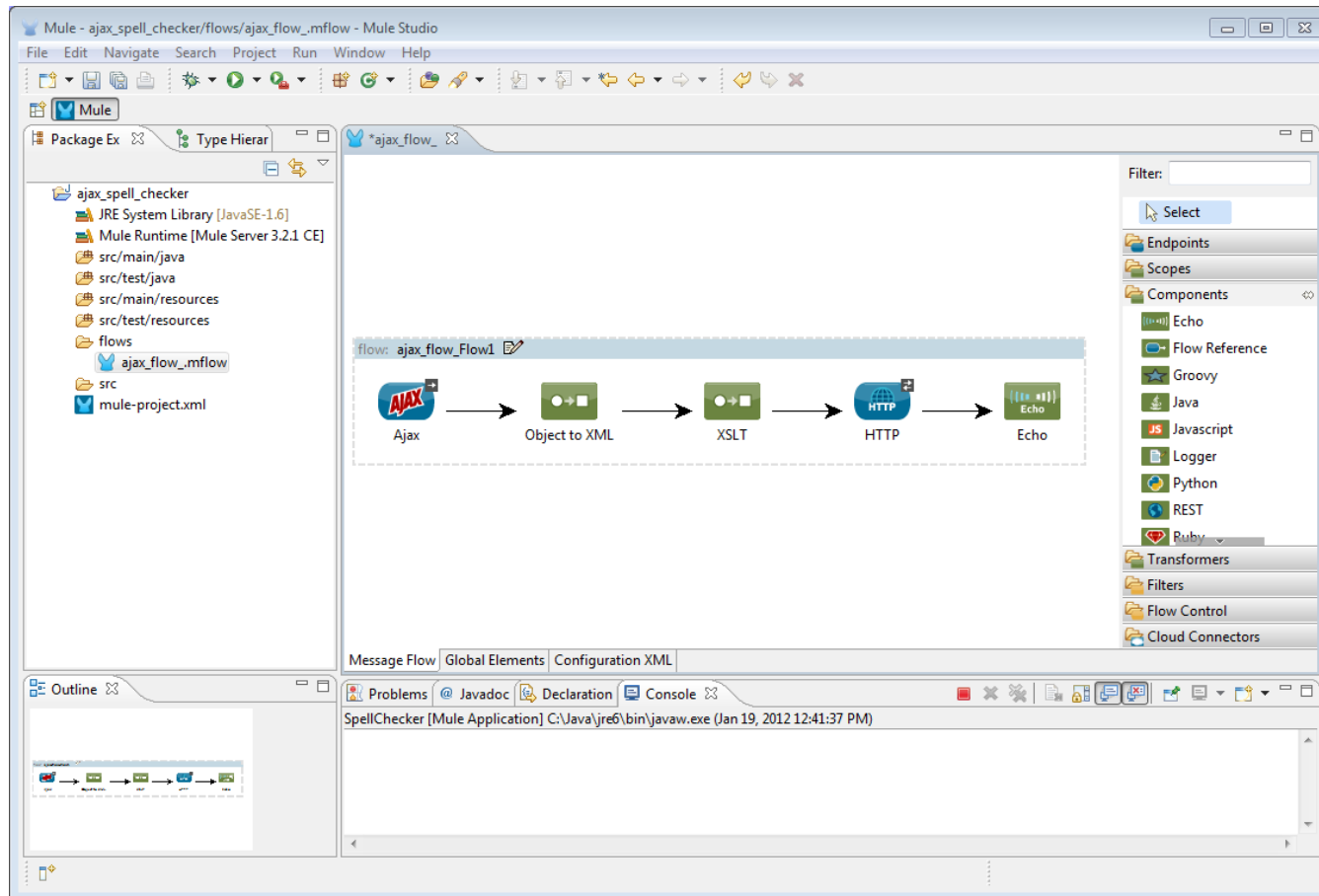


Graphically

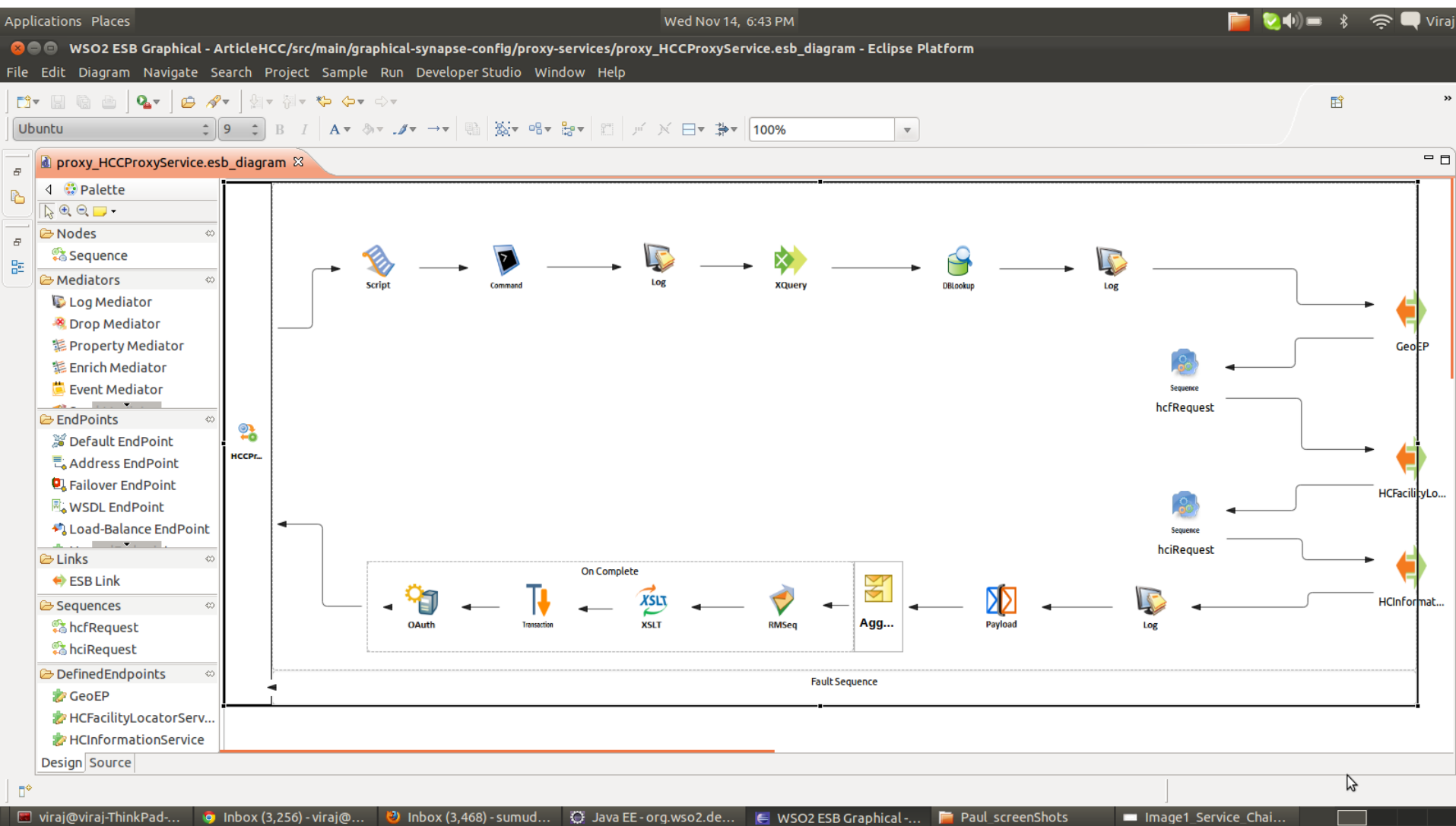
Apache Synapse terminology used



From some tooling



More Tooling



FUSE - LoanBroker/eip/credit_response.eip_diagram - FUSE

File Edit Diagram Navigate Search Project Run Window Help

Segoe UI 9 B I A 100%

Project Explorer

- LoanBroker
 - Spring Elements
 - src
 - JRE System Library [jre1.6]
 - build
 - eip
 - credit_response.eip
 - credit_response.eip.d
 - null

credit_response.eip_diagram

```
graph LR; Start(( )) --> Check[Check...]; Check --> Router{ }; Router --> Accept[Accept]; Router --> Reject[Reject]; Accept --> ProcessReq[Process Req]; ProcessReq --> InformCust[Inform Cust]; InformCust --> End(( )); Reject --> InvalidReq[Invalid Req]; InvalidReq --> End;
```

Palette

- Patterns
 - Resequencer
 - Routing Slip
 - Splitter
 - Throttler
- Endpoints
 - CXF
 - Direct
 - File
 - Generic
- Camel Proces...
 - Bean
 - Catch
 - Convert Body To
 - Finally

Check Processor

Properties

General

Destination Parameters

*Type queue

*Name checkResponse



Enterprise Integration Patterns

Home - Enterprise Integratio x


www.eaipatterns.com

Offline Mail Inbox (124,820) - p 100+ WSO2 WSO2, Inc. - Calend + bitmark Shorten with bit.ly

Enterprise Integration Patterns

Home

HOME • PATTERNS • RAMBLINGS • ARTICLES • TALKS • DOWNLOAD • LINKS • BOOKS • CONTACT


 **Ramblings**

My ongoing thoughts about the present and future of integration, SOA and Web services. [\[see all\]](#)

[DDD - Diagram Driven Design](#)
(March 22, 2010)

[What Does It Mean to Use Messaging?](#)
(Feb 17, 2010)

[A Chapter a Day...](#)
(Feb 1, 2010)

 **Upcoming Events**

Articles & Interviews

[Conversations Between Loosely Coupled Services](#)
(Video on [InfoQ](#))

[Developing in a Service-oriented World](#)
(Video on [InfoQ](#))


[SOA Patterns - New Insights or Recycled Knowledge?](#)
(Whitepaper)

Patterns and Best Practices for Enterprise Integration

This site is dedicated to making the design and implementation of integration solutions easier. The solutions and approaches described here are relevant for integration tools and platforms such as IBM WebSphere MQ, TIBCO, Vitria, SeeBeyond, WebMethods, or BizTalk, messaging systems such as JMS, WCF, or MSMQ, ESB's such as Sonic, Fiorano, ServiceMix, Mule, Apache Synapse, or WSO2, and SOA and Web-service based solutions.

All content on this site is original and is maintained by [Gregor Hohpe](#). I have been building integration solutions for large clients for many years and enjoy sharing my findings with the community. I hope you find this material insightful and useful. Please [contact me](#) if you have suggestions or feedback.

Enterprise Integration Patterns - The Book



[Enterprise Integration Patterns](#)
Gregor Hohpe, Bobby Woolf
Best Price \$23.99
or Buy New \$50.74

Enterprise integration remains harder than it really should be. While integration is inherently complex, I felt that one of the major stumbling blocks is the lack of a common vocabulary and body of knowledge around asynchronous messaging architectures used to build integration solutions. Under the guidance of Martin Fowler and Kyle Brown, I teamed up with Bobby Woolf to create such a language in the form of 65 [integration patterns](#) (see the pattern links on the right).

The book [Enterprise Integration Patterns](#) provides a consistent vocabulary and visual notation to describe large-

Integration Patterns

[Integration Patterns Overview](#)

[Table of Contents](#)

[Revision History](#)

Introduction

[Preface](#)

[Introduction](#)

[Solving Integration Problems using Patterns](#)

Integration Styles

[Introduction](#)

[File Transfer](#)

[Shared Database](#)

[Remote Procedure Invocation](#)

[Messaging](#)

Messaging Systems

[Introduction](#)

[Message Channel](#)

[Message](#)

[Pipes and Filters](#)

[Message Router](#)


[Message Translator](#)

[Message Endpoint](#)

Messaging Channels

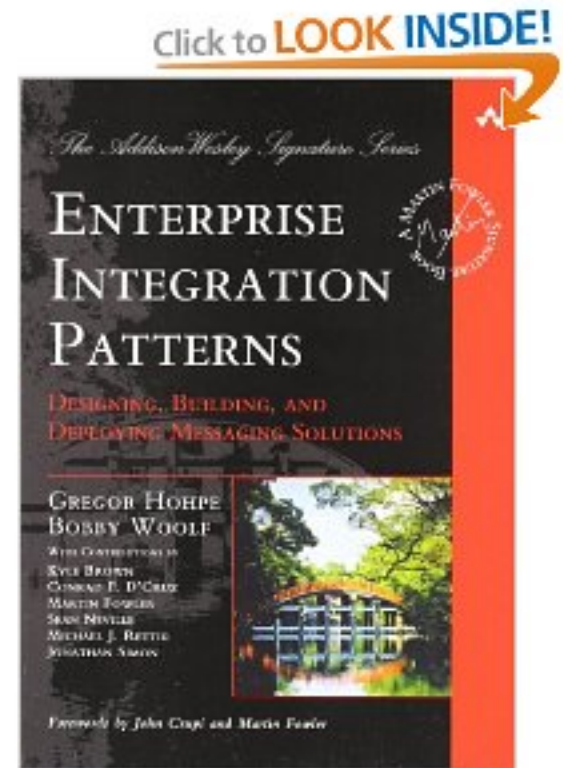
[Introduction](#)

[Point-to-Point Channel](#)



Enterprise Integration Patterns

- <http://www.eaipatterns.com/>
- The book
 - Enterprise Integration Patterns
 - Gregor Hohpe, Bobby Woolf



What actions

- The aim is to re-use existing adapters, transports and mediators/transformers
- Why?
 - Minimize custom coding
 - Utilize optimal components
 - e.g. streaming high-performance
 - Shorten test cycles
 - Be more agile



Common mediators

- Logging
- Routing
- Transformation
 - XSLT
 - Xquery
 - Template-ing
- Split/Aggregate
- Filter
- Clone/Tee
- Callout
- Enrich
- Drop
- Fault
- etc



Apache Synapse

- Designed to be simple to use and manage
 - XML configuration
 - No complex deployment
 - Hot deploy and update if needed
 - Separation of configs for different teams
 - Highly performant and scalable
 - Asynchronous core / non-blocking model

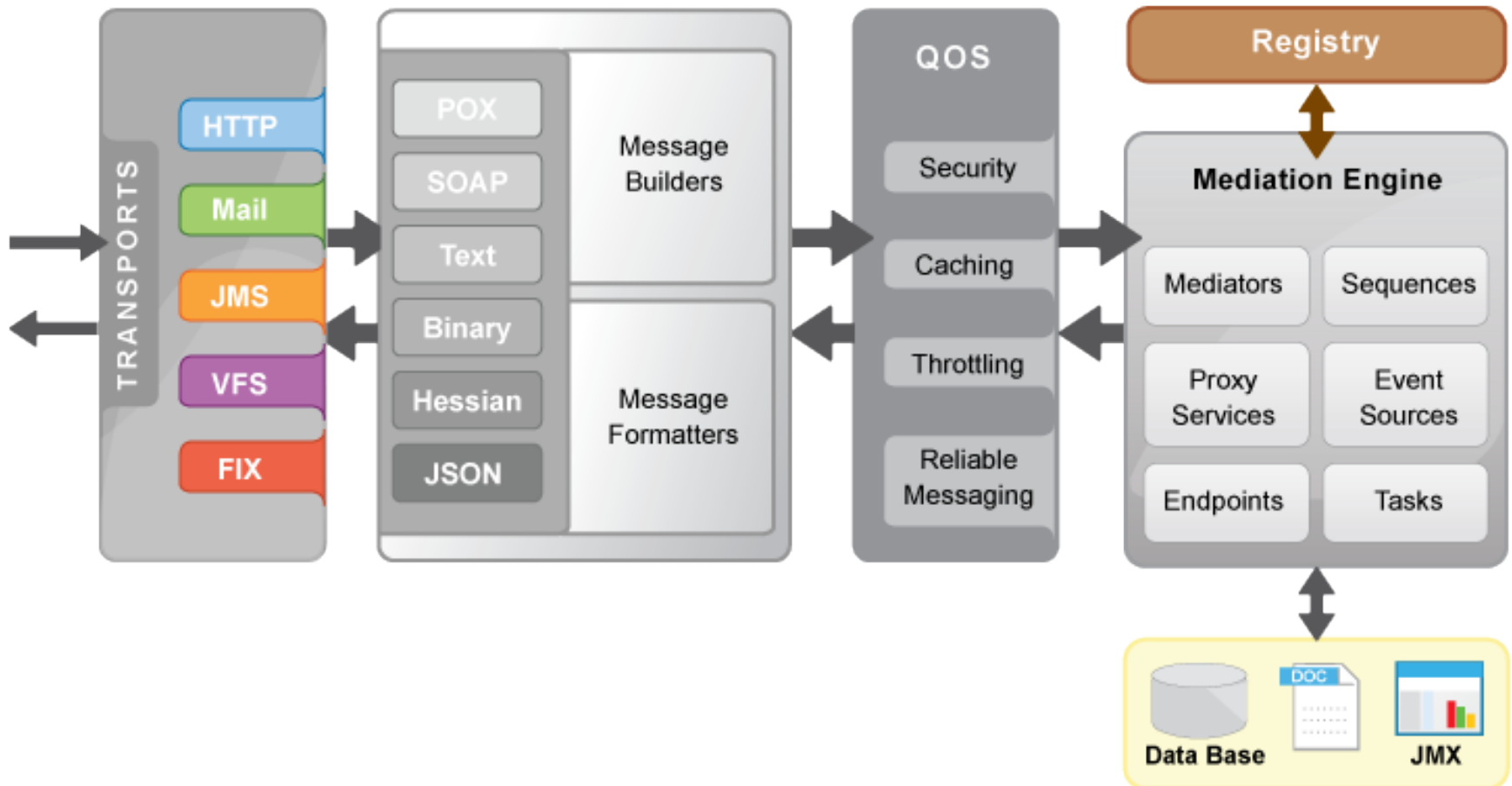


C10k Problem

- How to handle 10k concurrent requests
- Without 10k concurrent threads 😊
- Need to disassociate the socket from the thread
- Async handling
- Reactor pattern



Apache Synapse

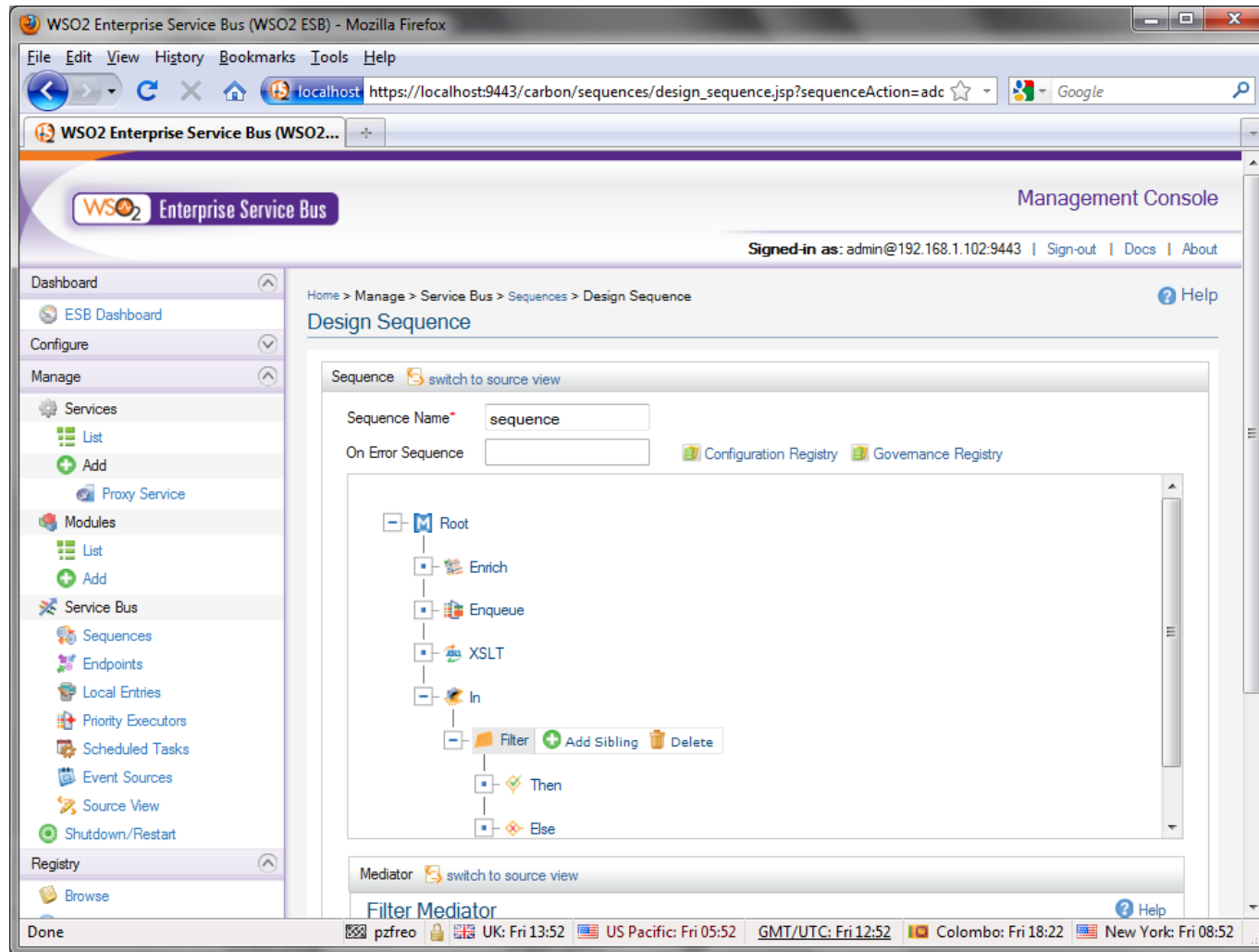


WSO2 ESB

- Also Apache License Open Source
- Adds a Graphical Web Interface
- Registry/Repository
- Deployment management/
synchronization
- Other pluggable components










ESB UI



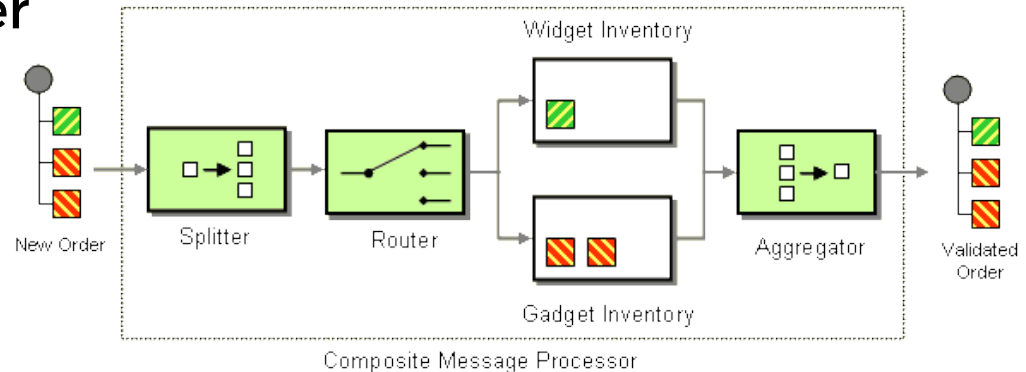
Transformation







- Transform via XSLT, XQuery, or Smooks
- Enrich via XPATH
- URL/Headers Management

Name		Description
XSLT Mediator		Invokes XSLT transformation on current message (v1.0 and v2.0 are supported)
XQuery Mediator		Invokes XQuery transformation on current message
Smooks Mediator		Invokes embedded Smooks Engine (v1.5) - Supports binary transformations (EDI, CSV, etc.)
Enrich Mediator		Enrich message contents using XPATH (replace, append, remove)
URL Rewrite Mediator		Rewrite protocol / URL contents
Header Mediator		Set / Remove Headers
Payload Factory		Override Message Contents

Enterprise Integration Patterns

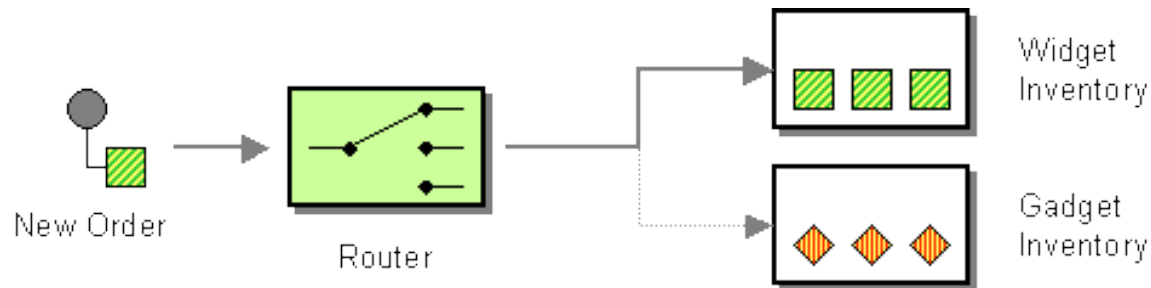
- Native Support for Common EIP
 - Content-based Router
 - Command Message
 - Message Filter
 - Message Splitter
 - Message Aggregator



Name		Description
Route Mediator		Routes message to given endpoint
POJOCommand		Creates instance of specific command class.
Iterate Mediator		Iterates over message and splits it into number of different messages derived from the parent message using XPATH.
Clone Mediator		Clones the entire message N times, each message is then treated in parallel
Aggregate		Aggregates multiple responses or messages, using XPATH.
Filter Mediator		Executes action based on evaluation of message contents against regular expression.

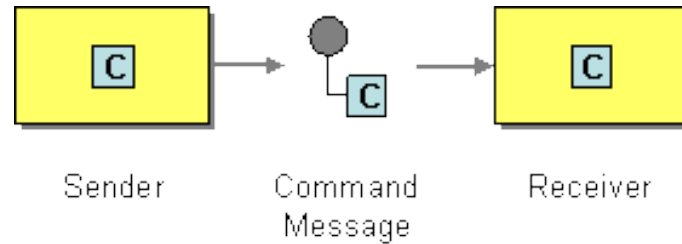
Content-Based Router

- `<router>` mediator



Command Message

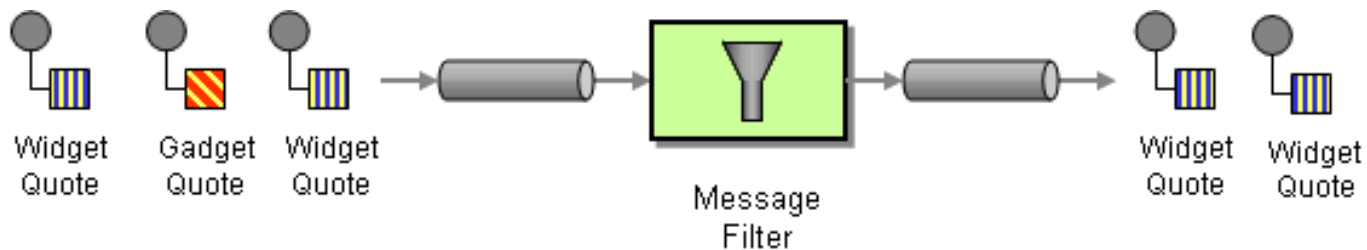
- <call> mediator



`C = getLastTradePrice("DIS");`

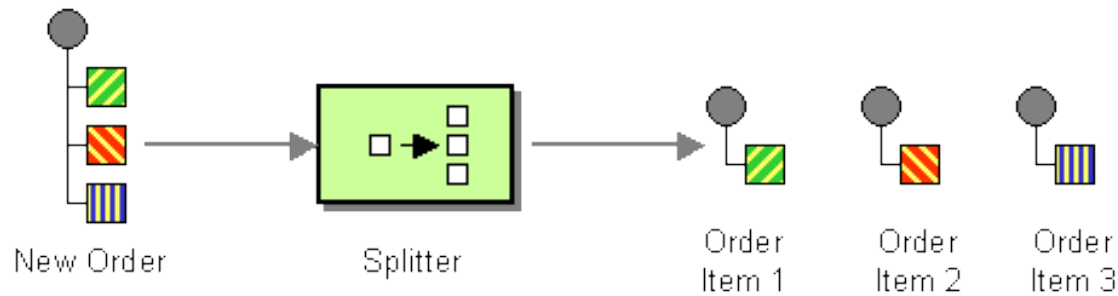
Message Filter

- <filter> mediator (with <drop> mediator)



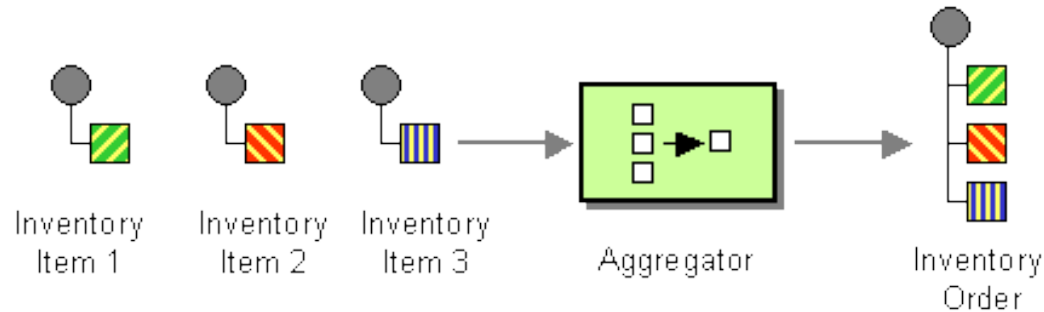
Splitter

- Iterate Mediator



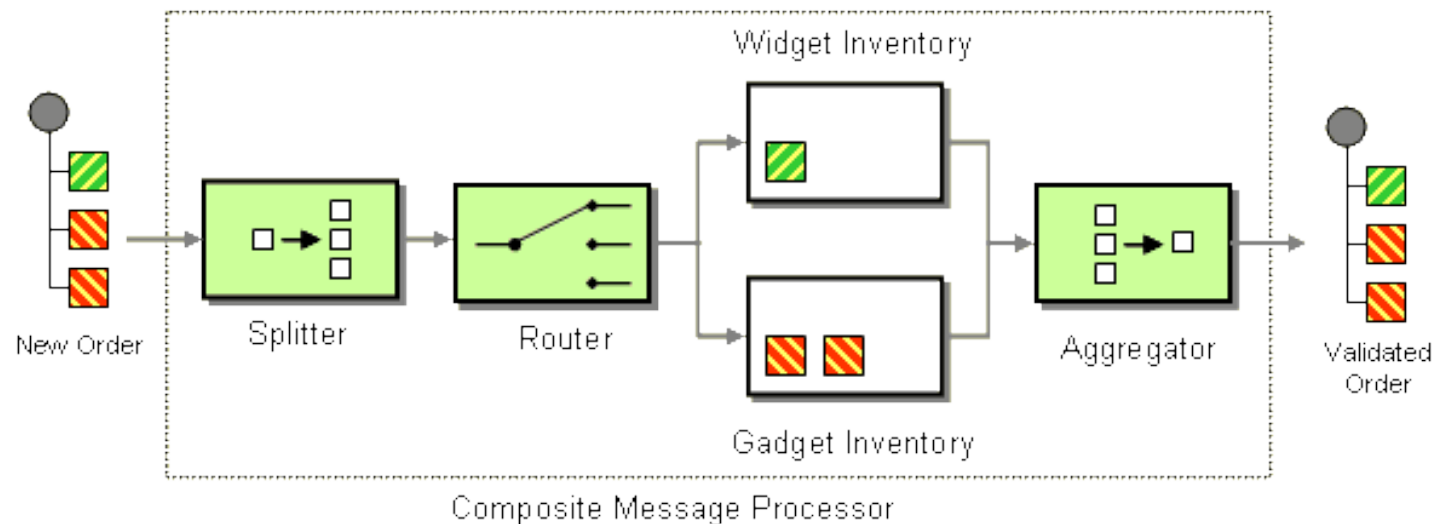
Aggregator

- Aggregate mediator



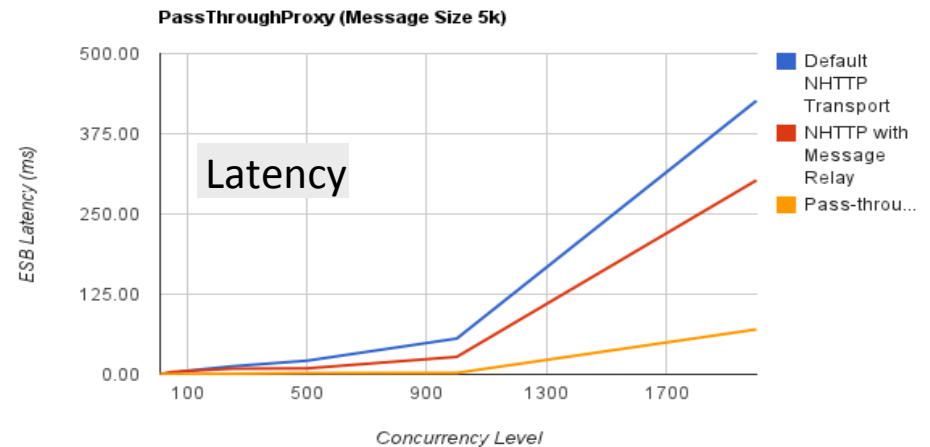
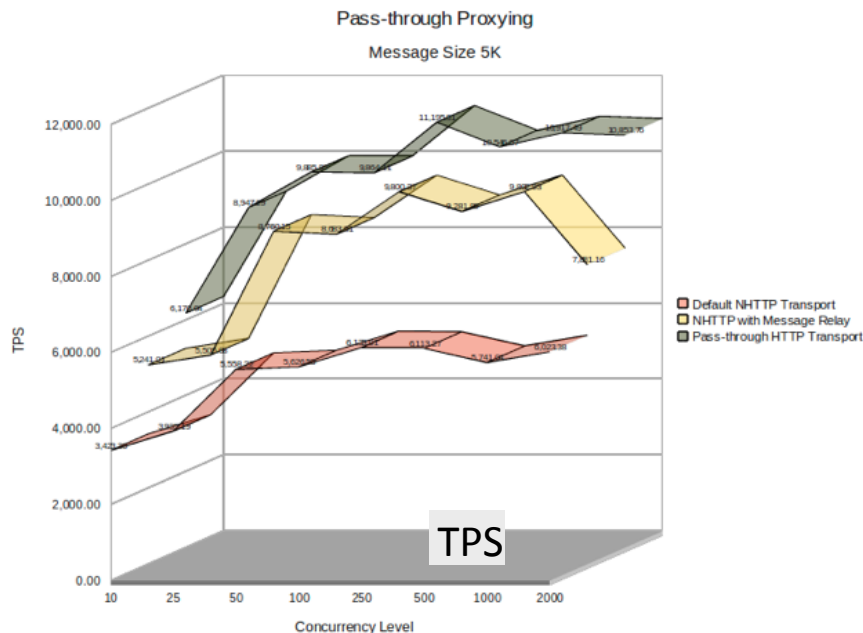
Composed Message Processor

- <sequence>



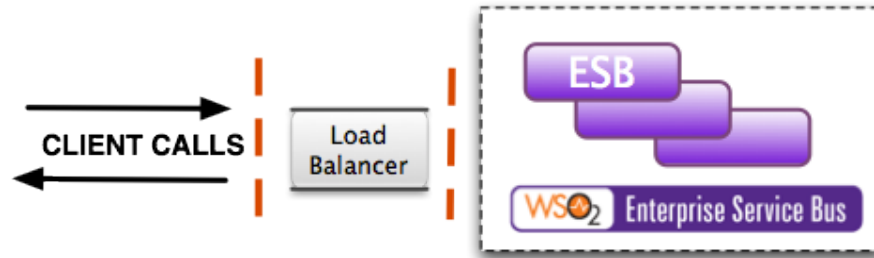
High Performance and Stability

- Supports 1000s of concurrent non-blocking HTTP transaction per server
- Pure streaming and Optimization using Message relay (on-demand processing of messages)
- Very Low latency (0.5 ms for Non-Blocking IO transport)
- Long Term Execution Stability with Low Resources Utilization
- Response Caching



High Availability and Scalability

- Supports Active/Active, Active/Passive Scenarios



- ESB itself can act as load-balancer.
- Auto-scaling using Load Balancer component
- Deployment Synchronizer can be used to maintain configuration across clusters.

How does mediation / integration fit into Microservices / Containers?

- One view:
 - Smart endpoints and dumb pipes
- Another approach
 - Micro-integrations
 - Each container just handles a single flow
 - Apache Camel with Java DSL is good for this
 - In some scenarios
- Where is the canonical model in this world?
- Do we still need declarative languages with better DevOps?



Ballerina

- A new integration language and framework for Microservices, RESTful SOA
- Based on Swagger and Sequence Diagrams
- Textual and graphical are 100% interchangeable
- <http://ballerinalang.org>



Ballerina diagram and language

echoService.bal

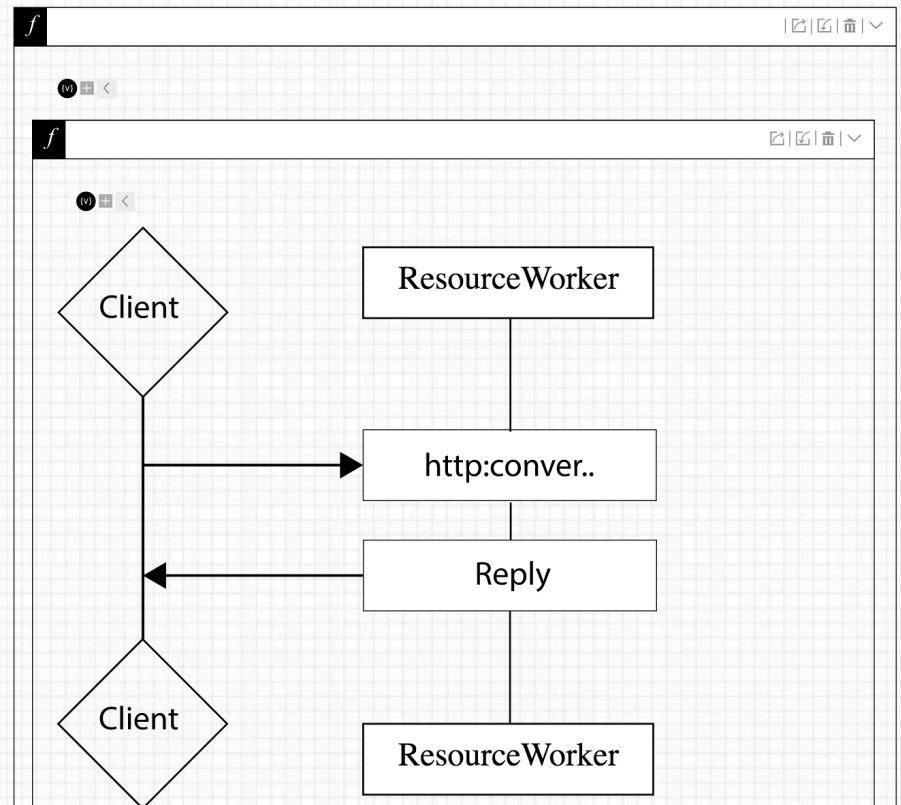
```
//This sample will echo the request message

import ballerina.net.http;

@http:BasePath ("/echo")
service echo {

    @http:POST
    resource echo (message m) {
        http:convertToResponse(m);
        reply m;
    }
}
```

Type your msg here



Resources

- Wikipedia!
 - http://en.wikipedia.org/wiki/Enterprise_service_bus
- Books
 - David Chappell: ESB
 - Open Source ESBs in Action
- Open Source
 - synapse.apache.org
 - wso2.com/products/enterprise-service-bus
 - servicemix.apache.org

