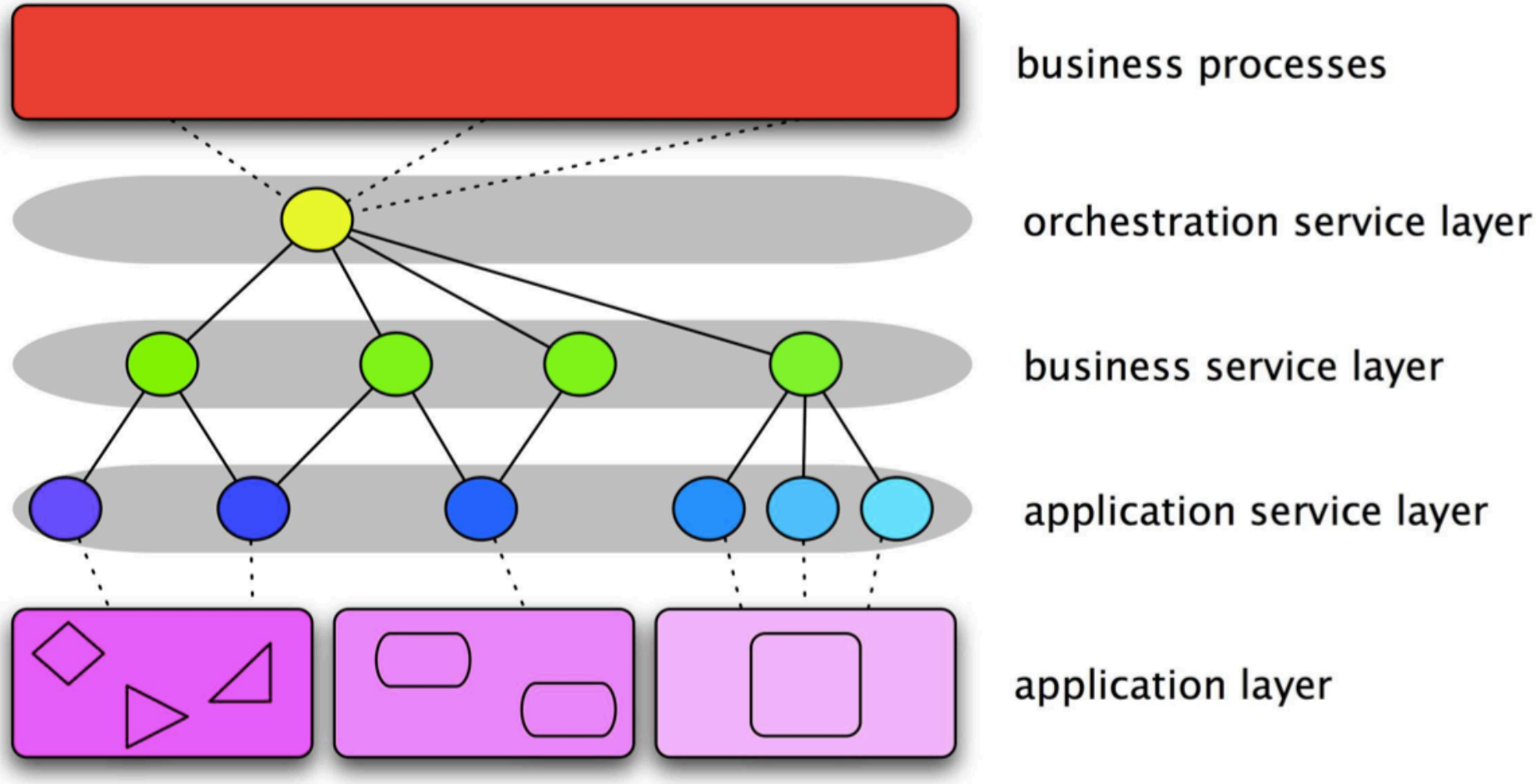
SOA integration and mediation

Oxford University
Software Engineering
Programme
June 2016

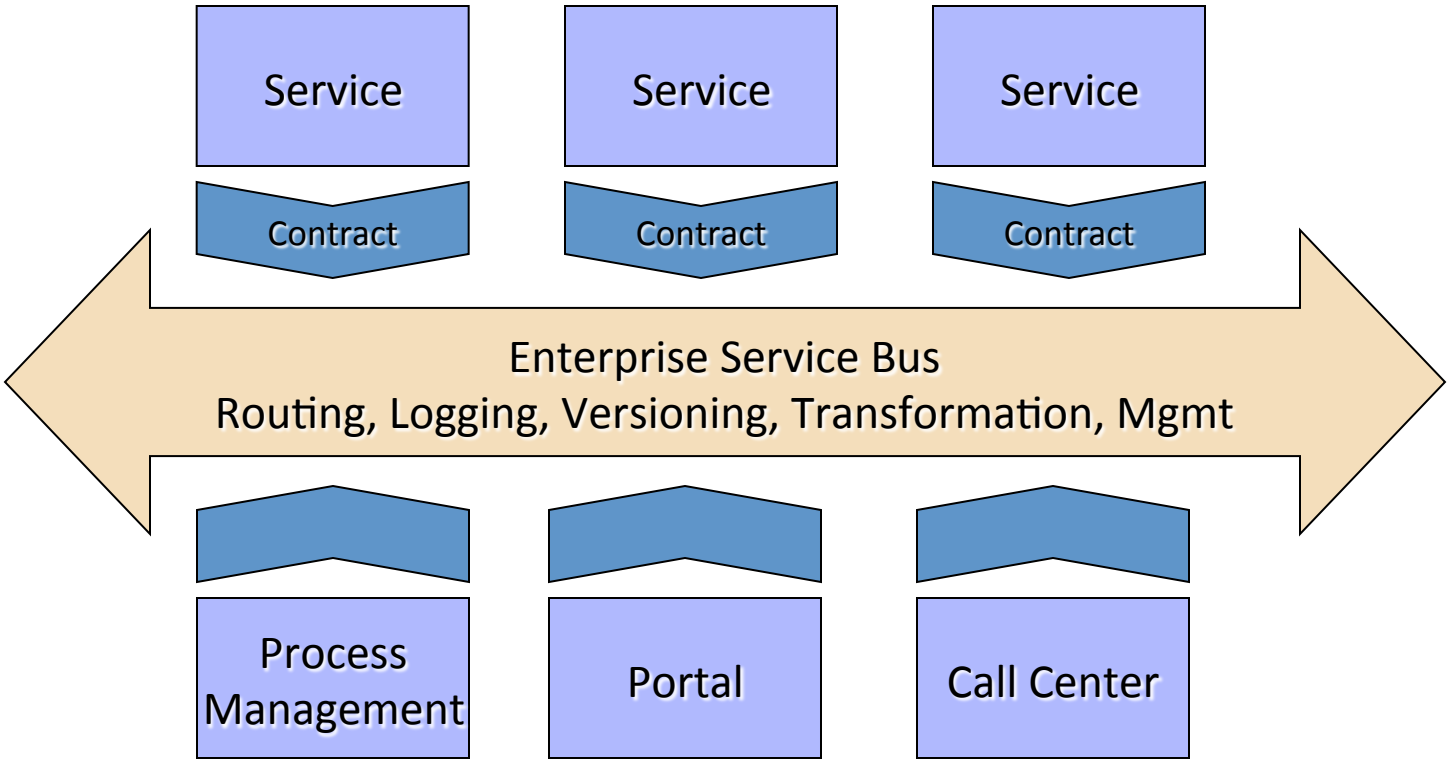
Recap on SOA model



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

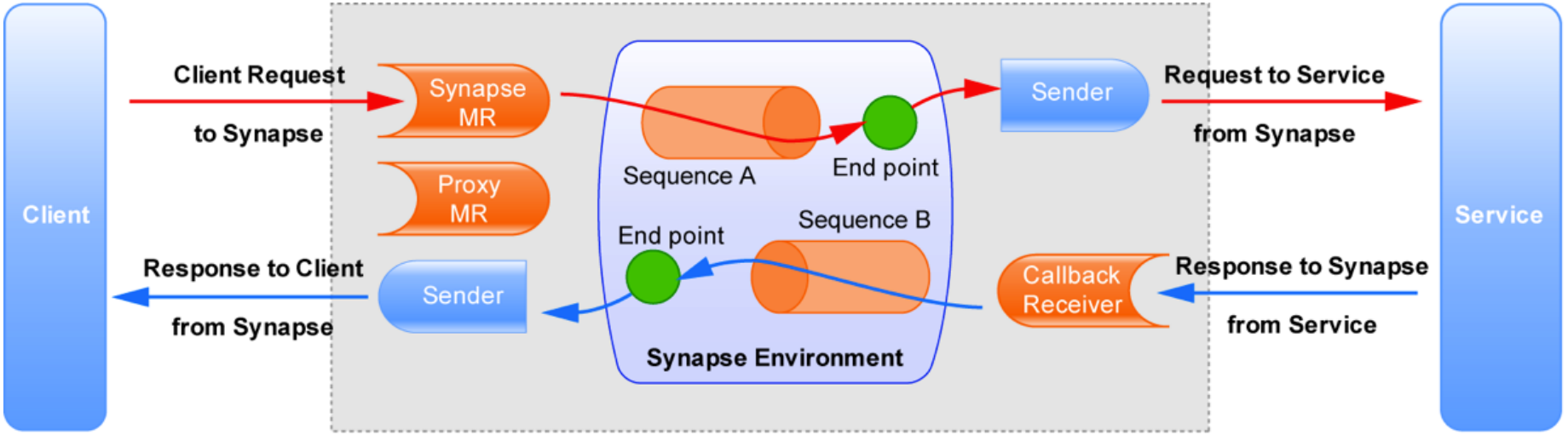
ESB models

- Almost all ESBs work on the same principle

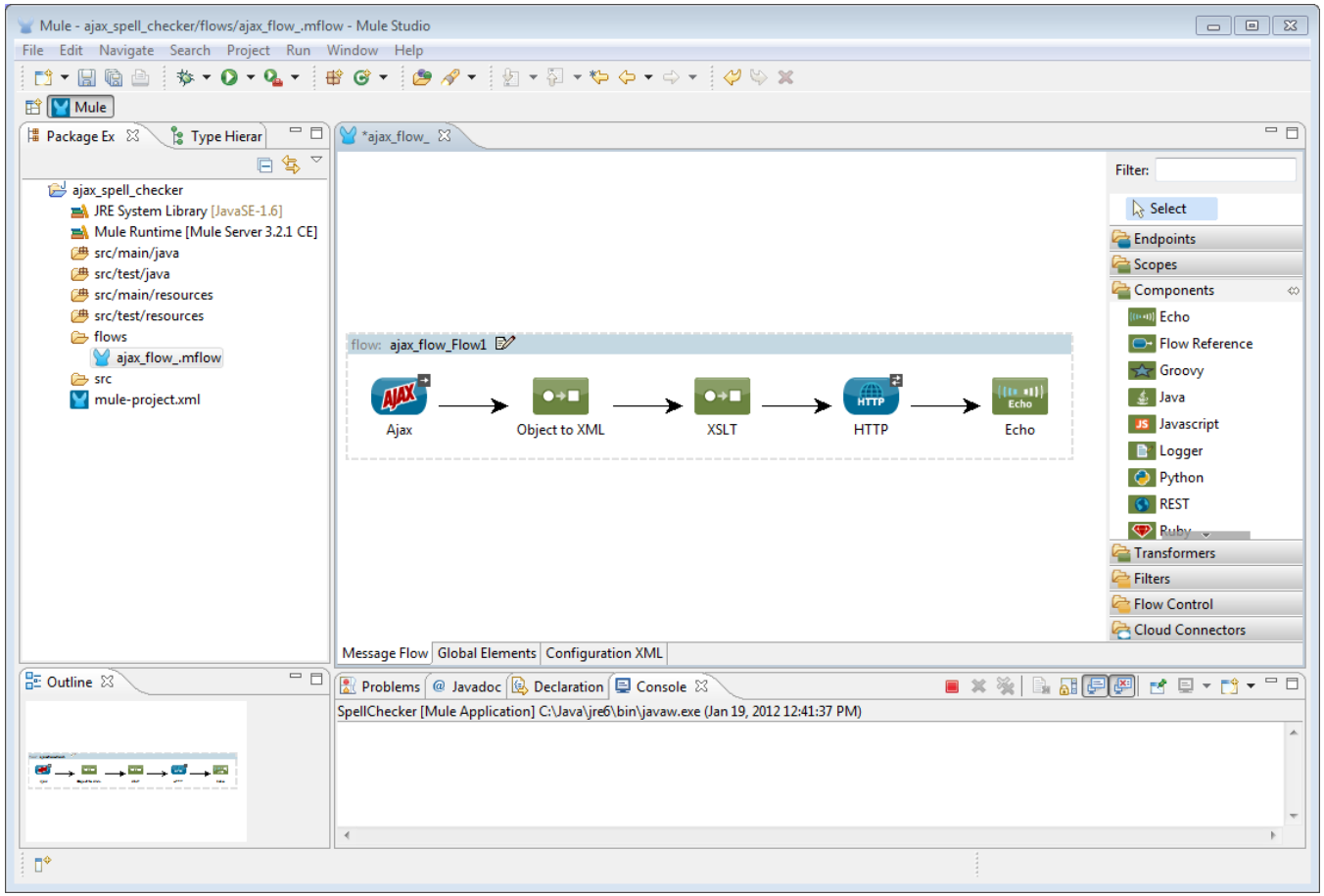
- Message arrives
- Sequence of actions (Pipeline)
- Message is sent on

Graphically

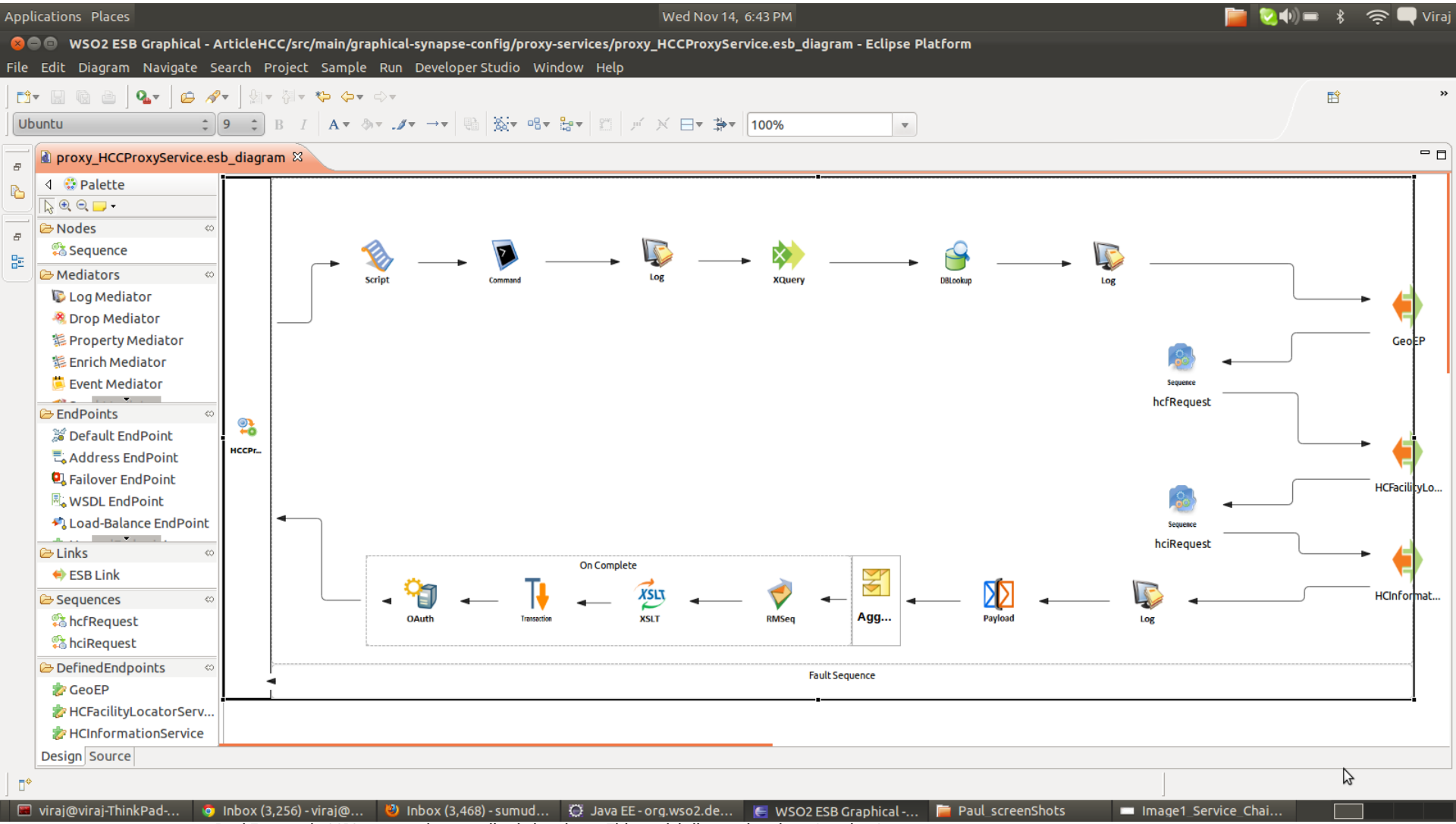
Apache Synapse terminology used

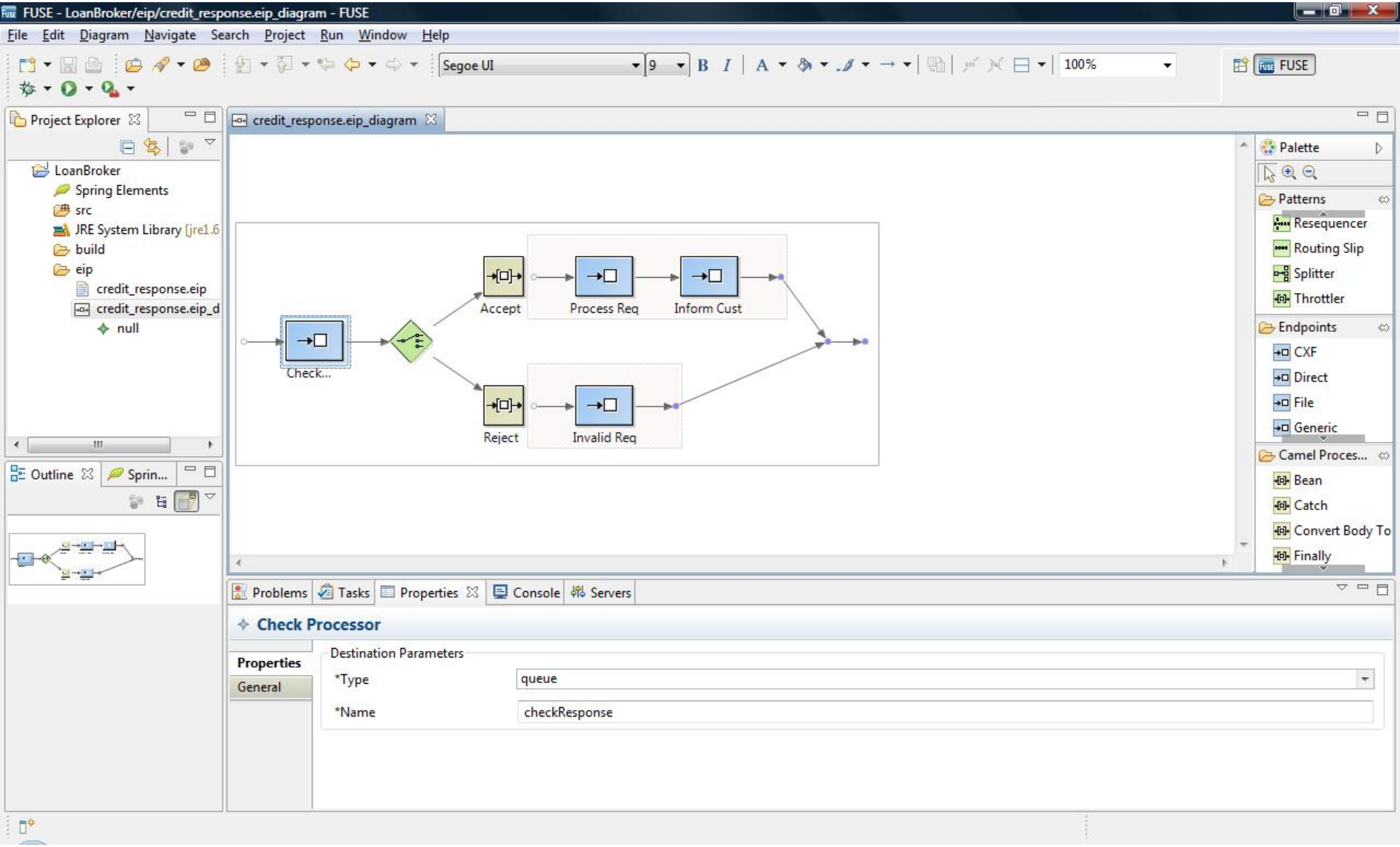


From some tooling

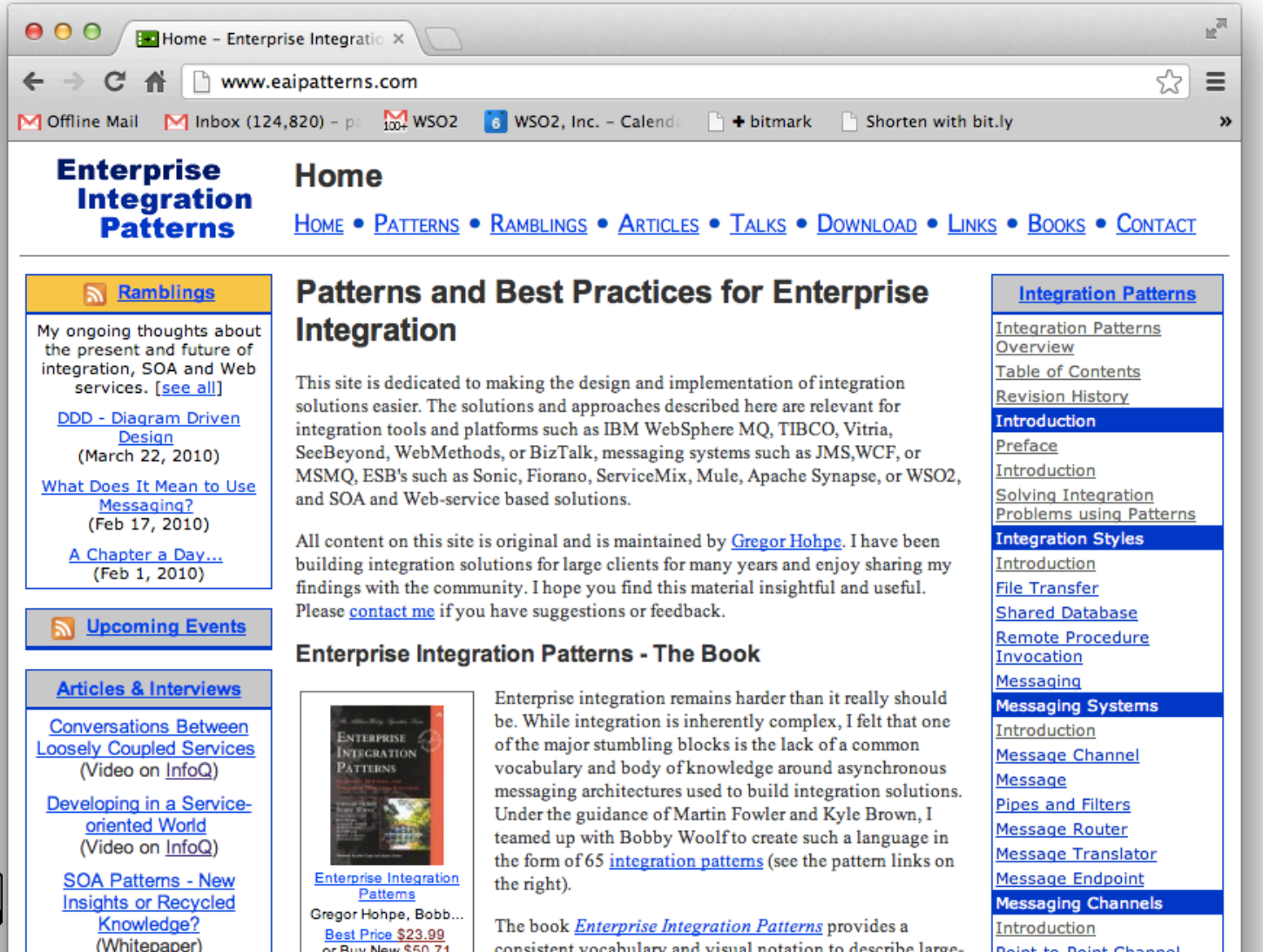


More Tooling



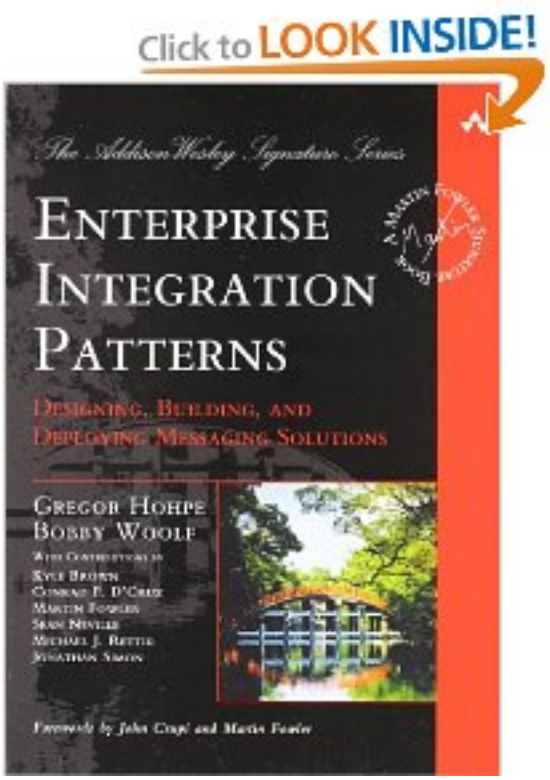


Enterprise Integration Patterns



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

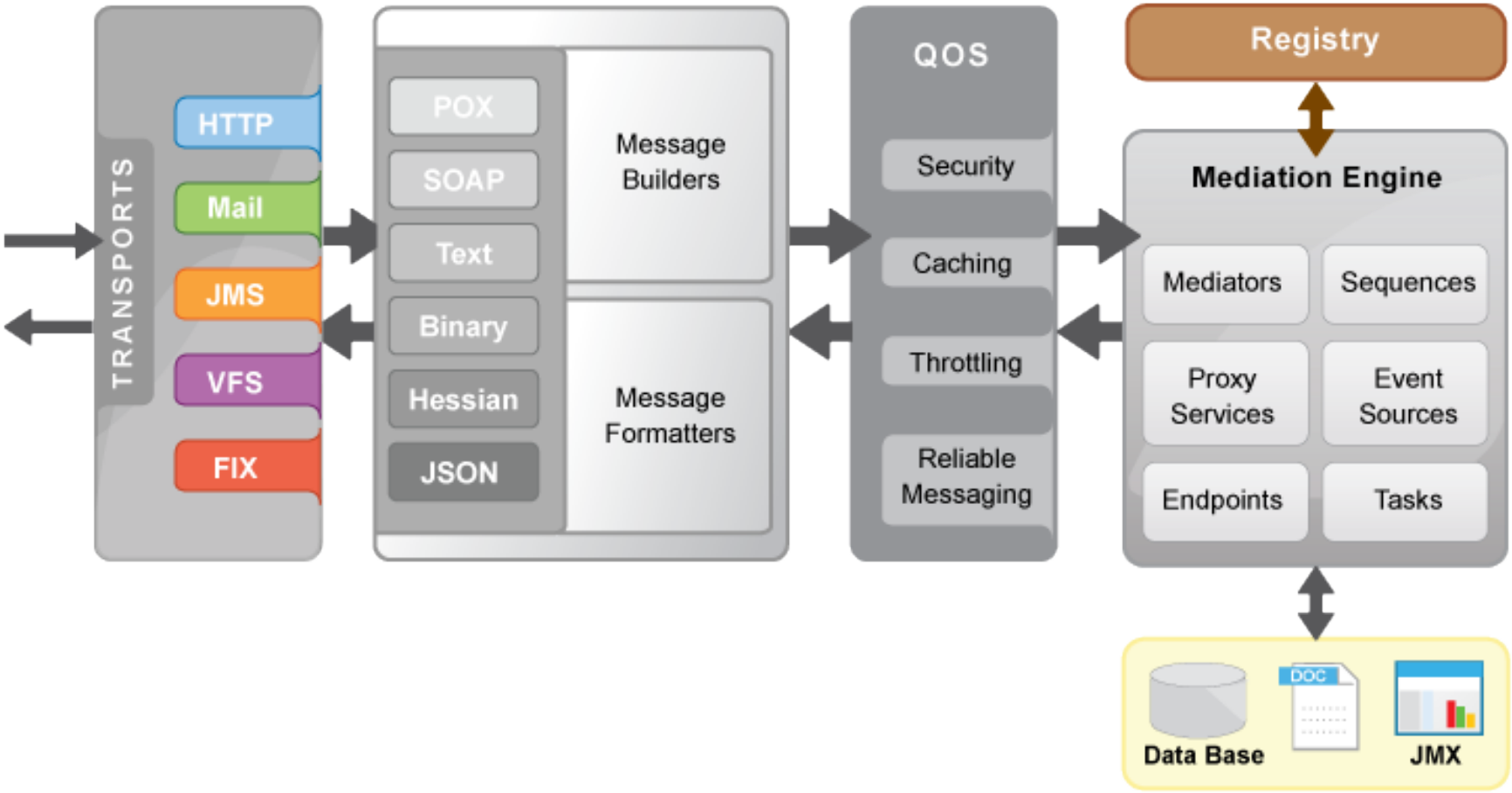
Apache Synapse

- 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

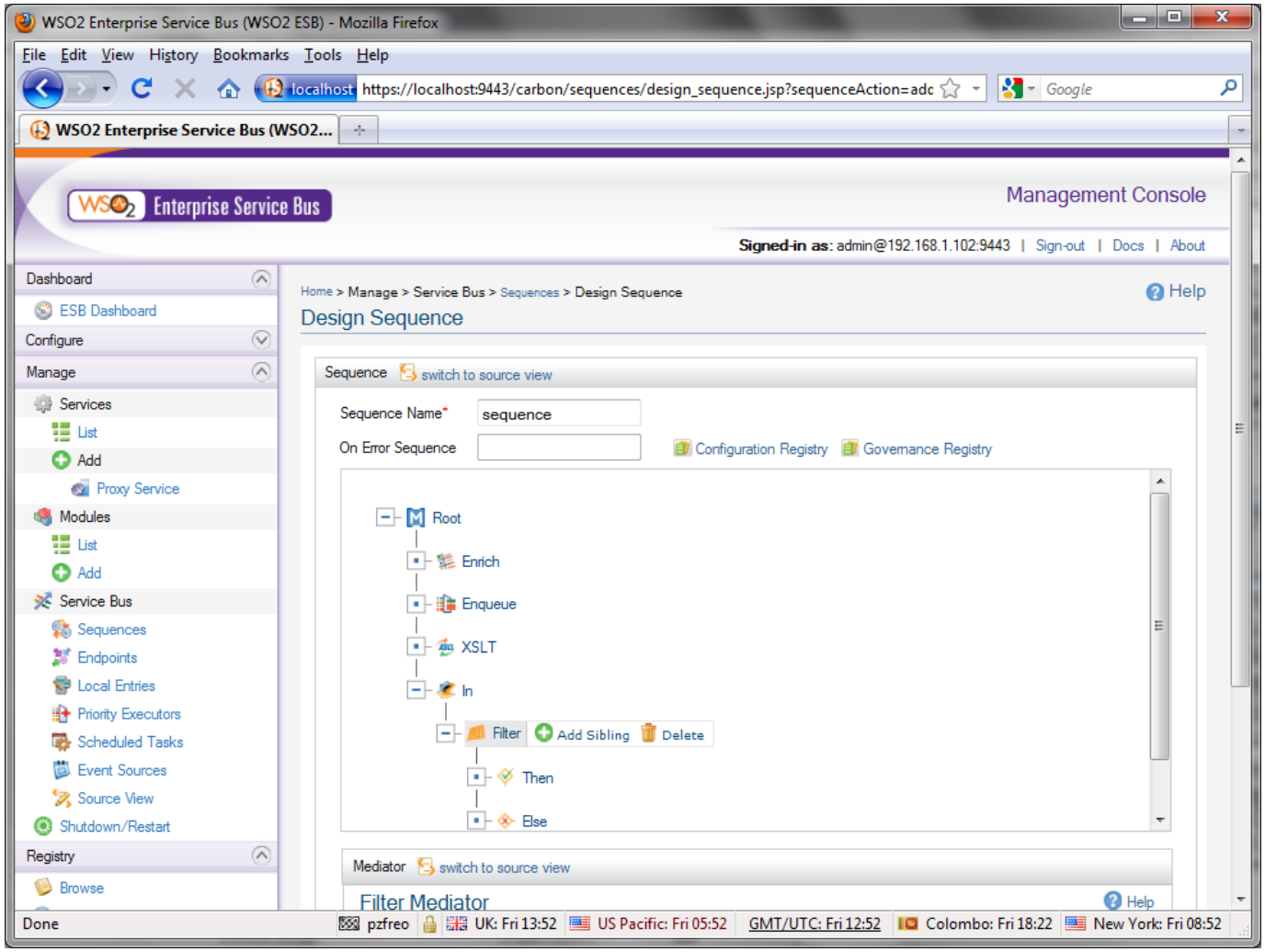
Apache Synapse



WSO2 ESB

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

ESB UI



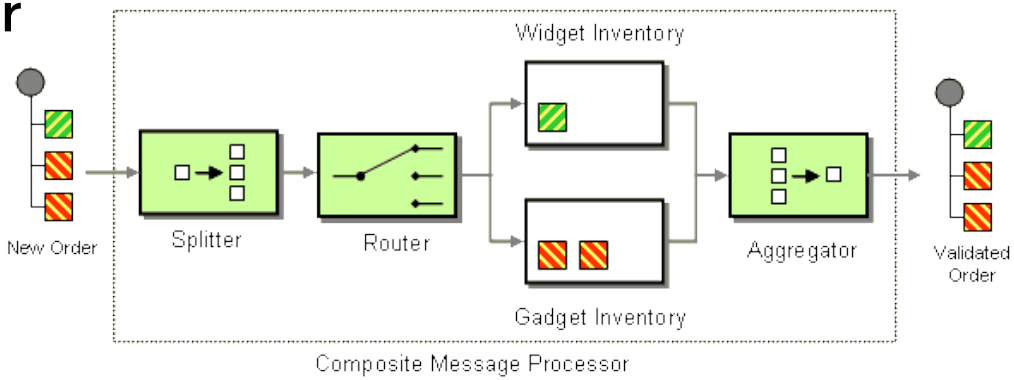
Transformation

- 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

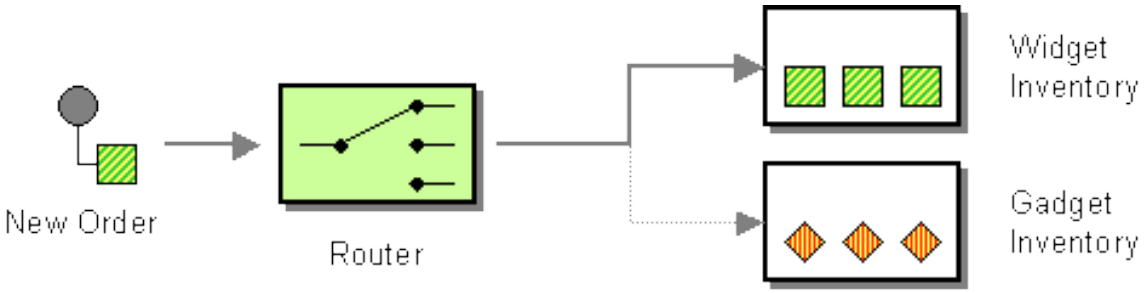
- Native Support for Common EIP
  - Content-based Router
  - Command Message
  - Message Filter
  - Message Splitter
  - Message Aggregator



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.

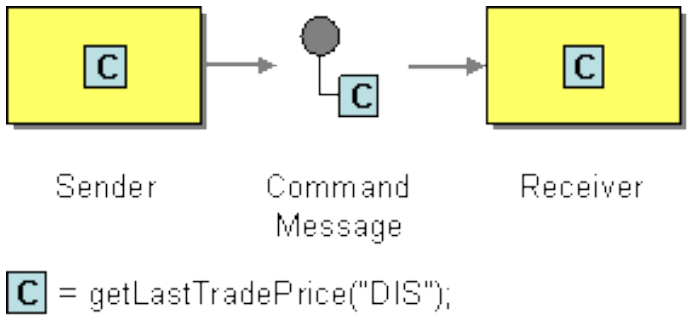
Content-Based Router

- <router> mediator



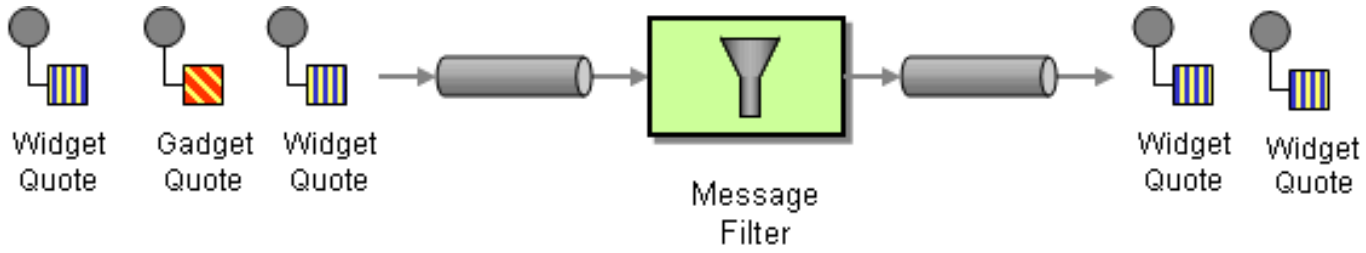
Command Message

- <call> mediator



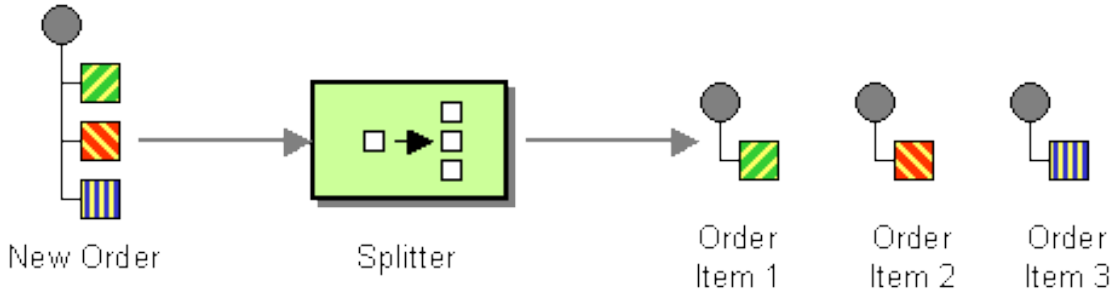
Message Filter

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



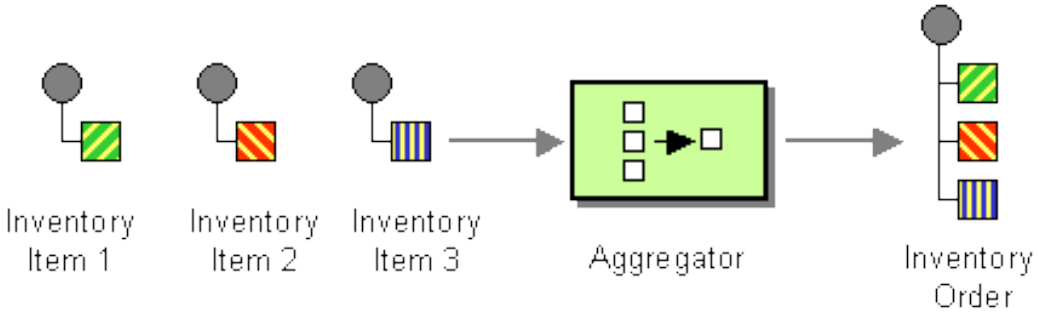
Splitter

- Iterate Mediator



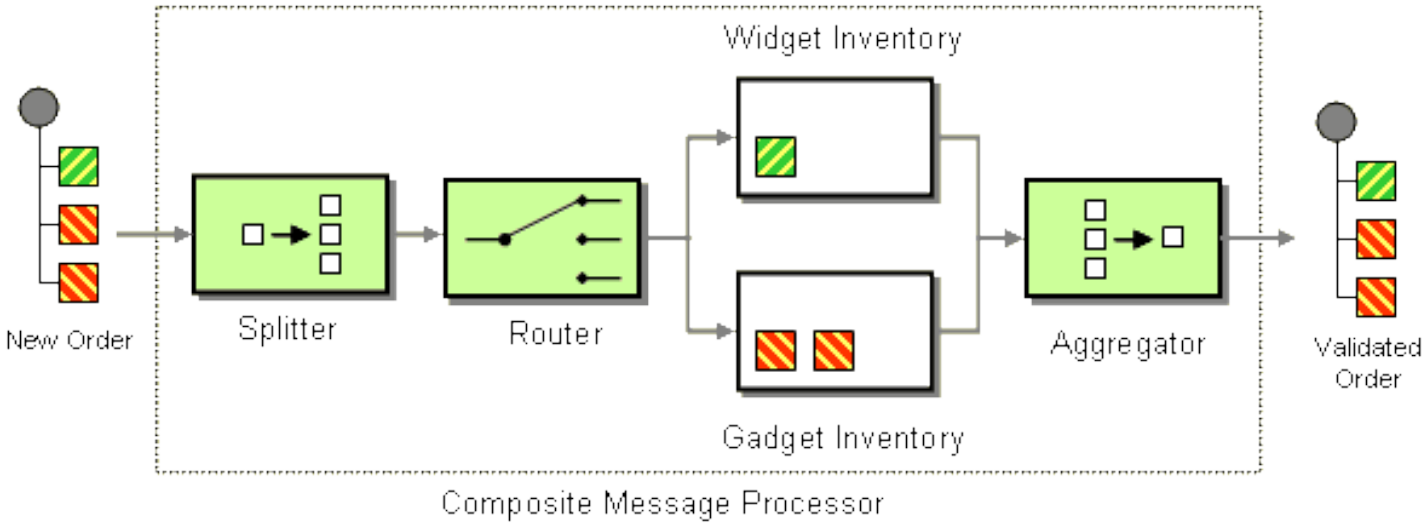
Aggregator

- Aggregate mediator



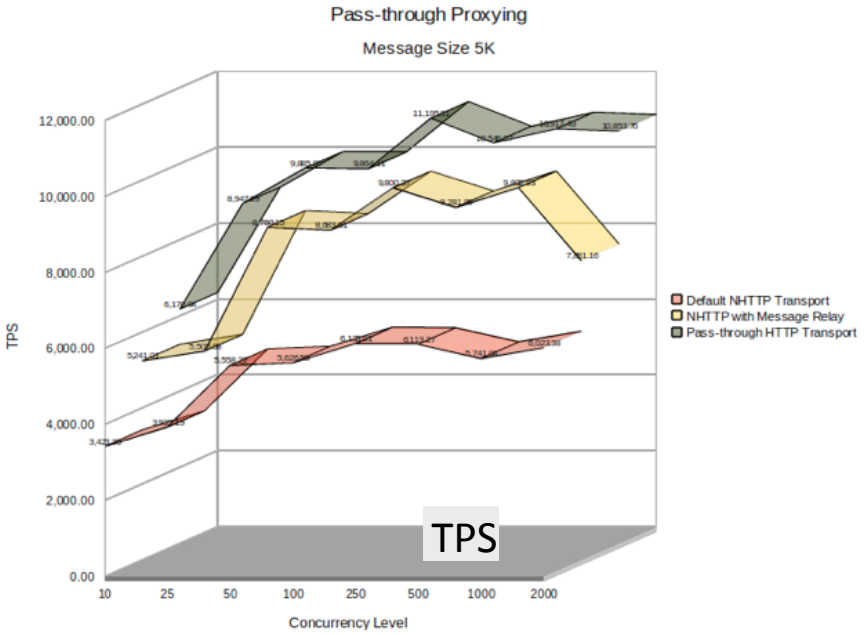
Composed Message Processor

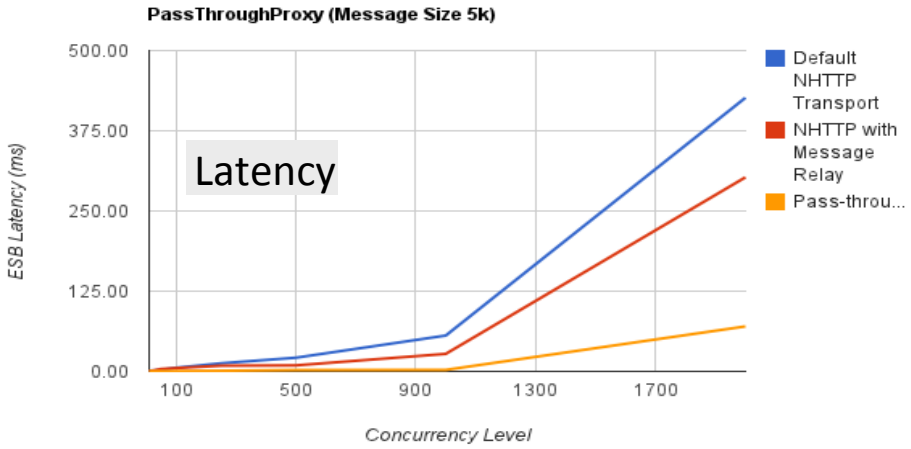
- <sequence>



High Performance and Stability

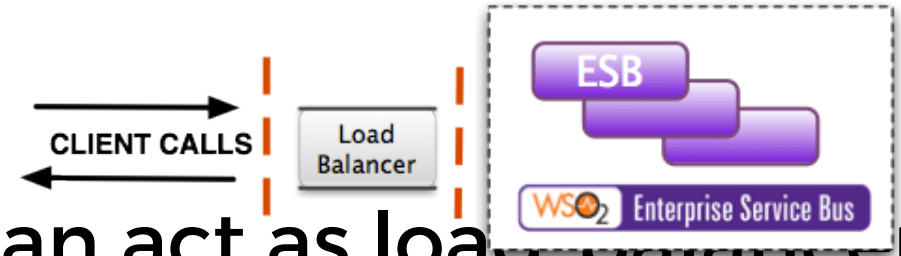
- Supports 1000s of concurrent non-blocking HTTP transaction per server
- Pure streaming and Optimization using Message relay (on-demand processing of messages)
- Very Low latency (0.5 ms for Non-Blocking IO transport)
- Long Term Execution Stability with Low Resources Utilization
- Response Caching





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?

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