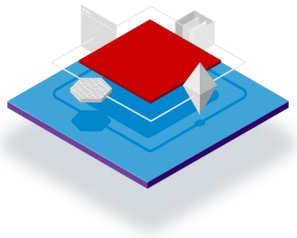


Red Hat Fuse – 7.x

A Distributed, Cloud-native Integration Platform

– **Avadhut**

Course Structure

- **Part – 2 : Fundamentals**
  - Integration options
  - History of Fuse
  - What is Enterprise Integration patterns?
  - What is OSGi?

Course Structure

- **Part – 3 : Core Concepts**
  - Core Concepts (To be covered in subsequent slides)
  - Fuse flavors/Offerings
  - Fuse Eco-system
  - Fuse Sub-systems
  - Fuse Architecture
  - Role of Spring Boot
  - Role of Apache Camel
  - Fuse Management and Command Line Interface (Fuse CLI)
  - Fuse Operations

Course Structure

- Part – 4 : Fuse Installation and CLI

- Fuse Installation
- Management HawtIO
- Understanding CLI/Karaf container in picture
- Implementation of OSGi in production

** Commands to interact with Karaf

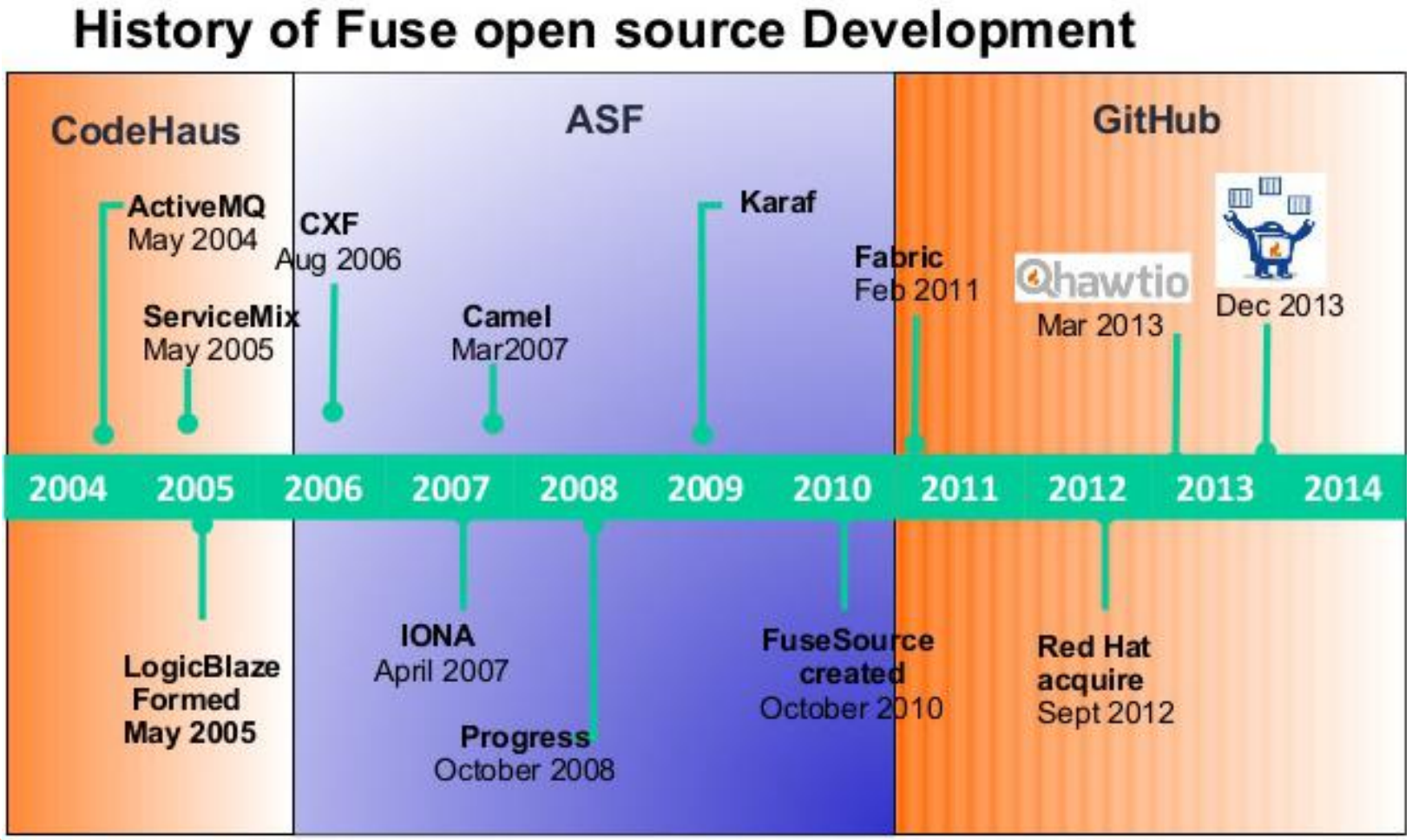
Red Hat Fuse Theory

Fundamentals

Integration Options

- Oracle Service Bus
- WSO2 ESB
- Mule ESB
- Talend ESB
- Azure Service Bus
- Webmethods Integration Server
- IBM Integration Bus/IBM ESB
- Tibco BW
- Red Hat (JBoss) Fuse

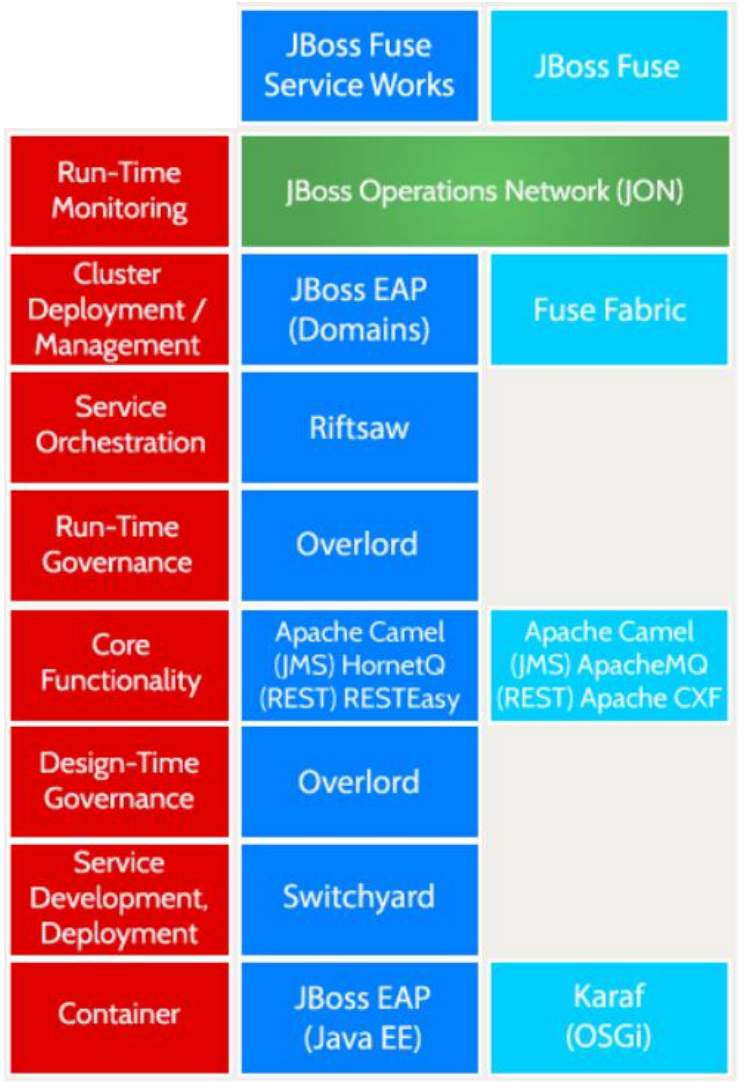
History Of Fuse



History Of Fuse

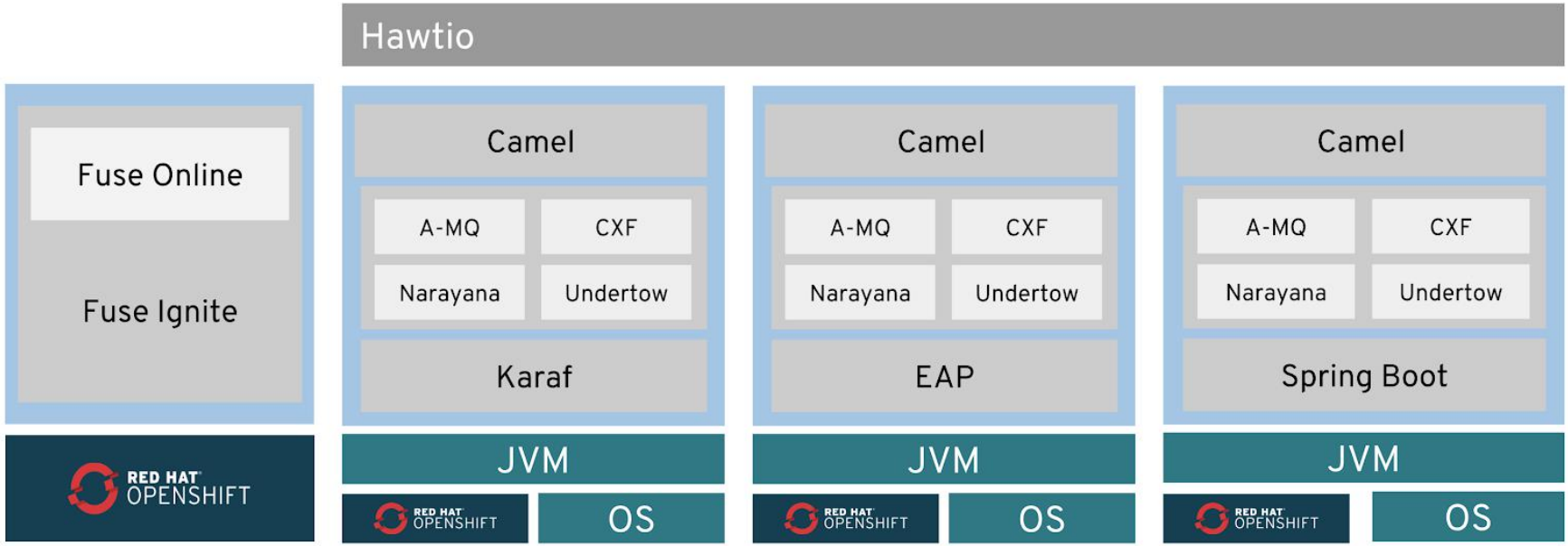
- In May, 2005, James Strachen and Hiram Chirino co-founded a software company named LogicBlaze Inc. and offered Fuse SOA
- In April, 2007, IONA Technologies, an Irish company acquired the LogicBlaze Inc. and got the ownership of SOA based Fuse.
- In September, 2008, Progress Software acquired IONA Technologies
- And made it subsidiary as FuseSource in October 2010
- and they renamed product as ‘Fuse ESB’
- Re Hat announced its acquisition of FuseSource from Progress Software in Sept, 2012

Evolution Of Fuse



Evolution Of JBoss Fuse Service Works to JBoss Fuse

Evolution to Fuse-7



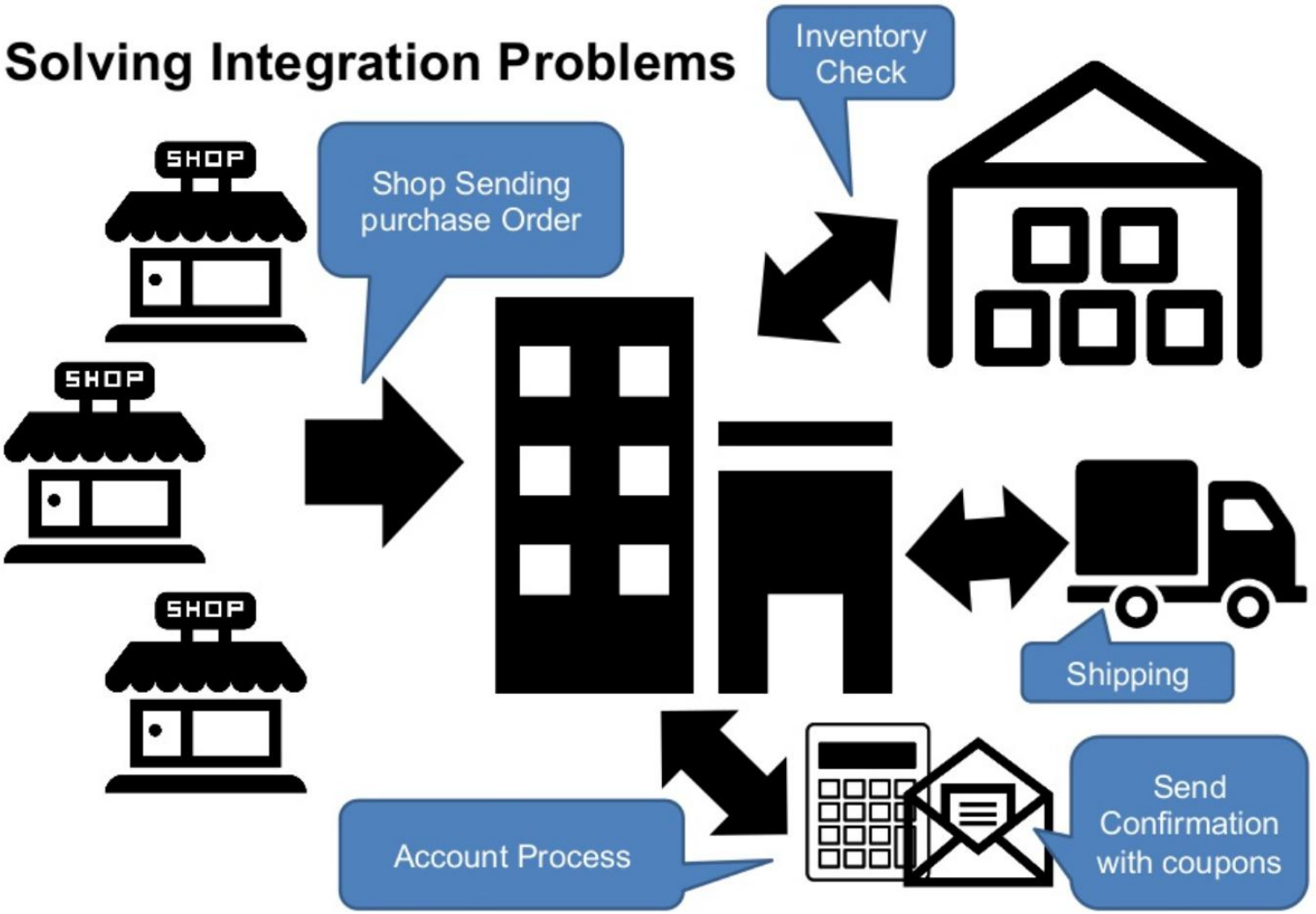
Enterprise Integration Patterns

- A design pattern is a general solution to a design problem that recurs repeatedly in many projects.
- A pattern describes the problem and its proposed solution and discuss any other important factors.
- EIP focuses on messaging patterns for enterprise application integration (EAI)

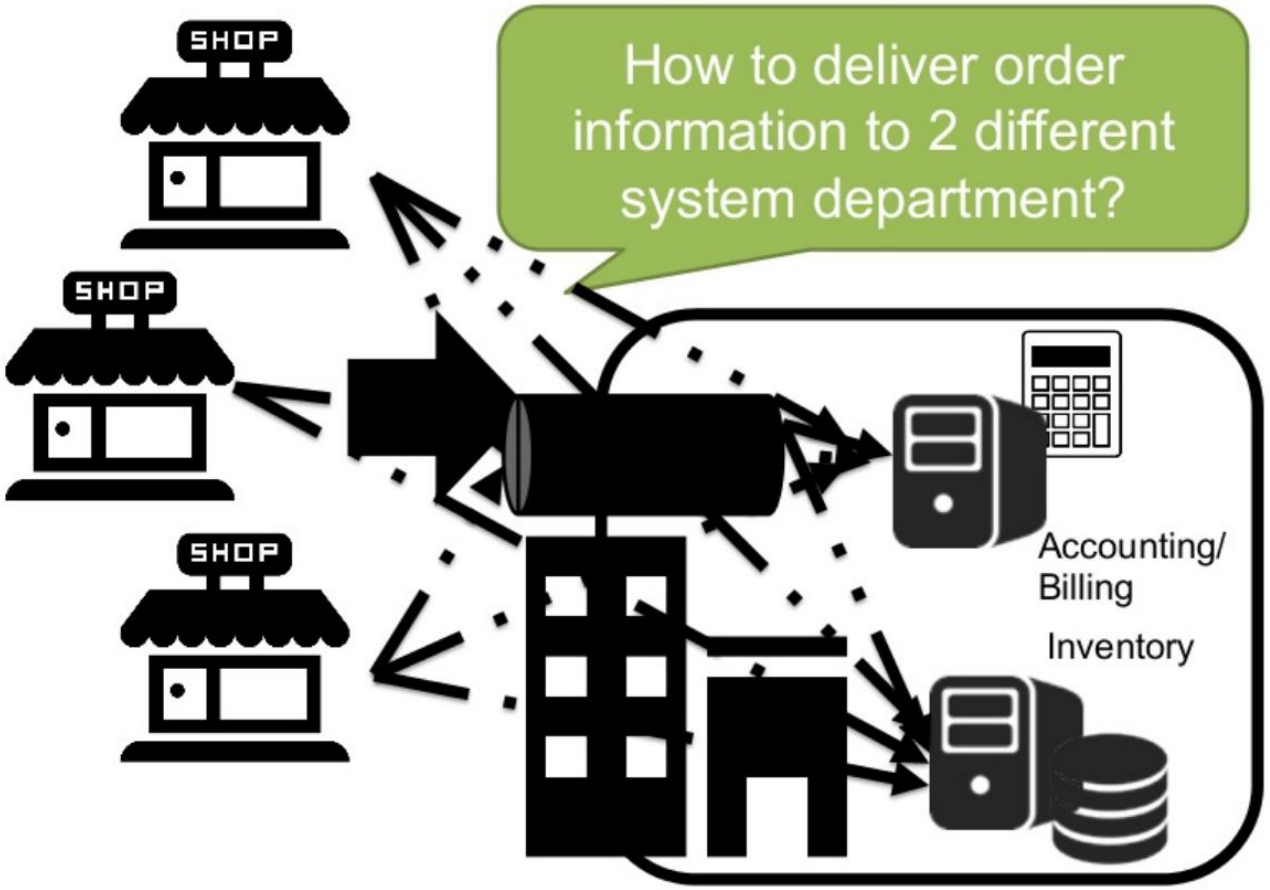
Enterprise Integration Patterns

Examples and explanations

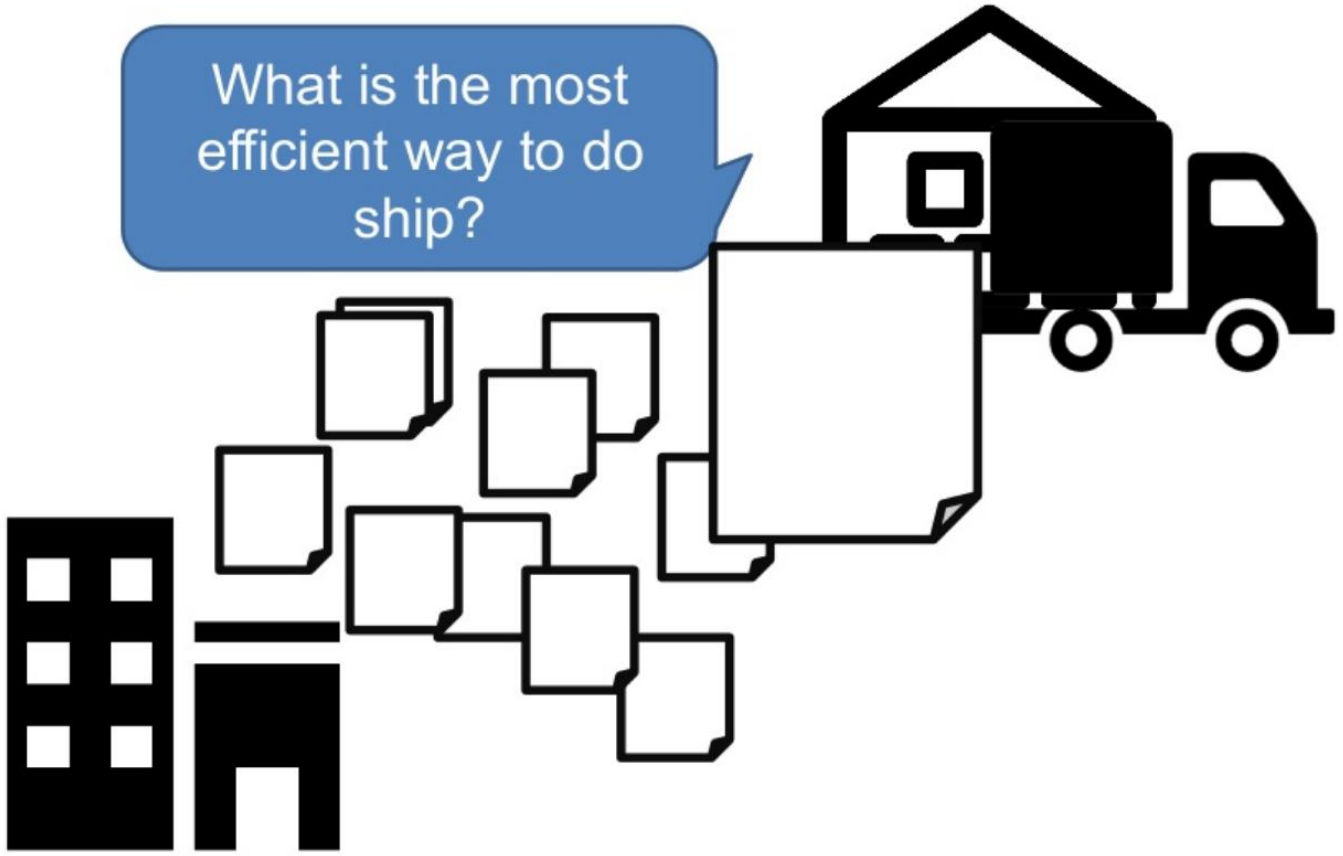
Enterprise Integration Patterns



Solving Integration Problems

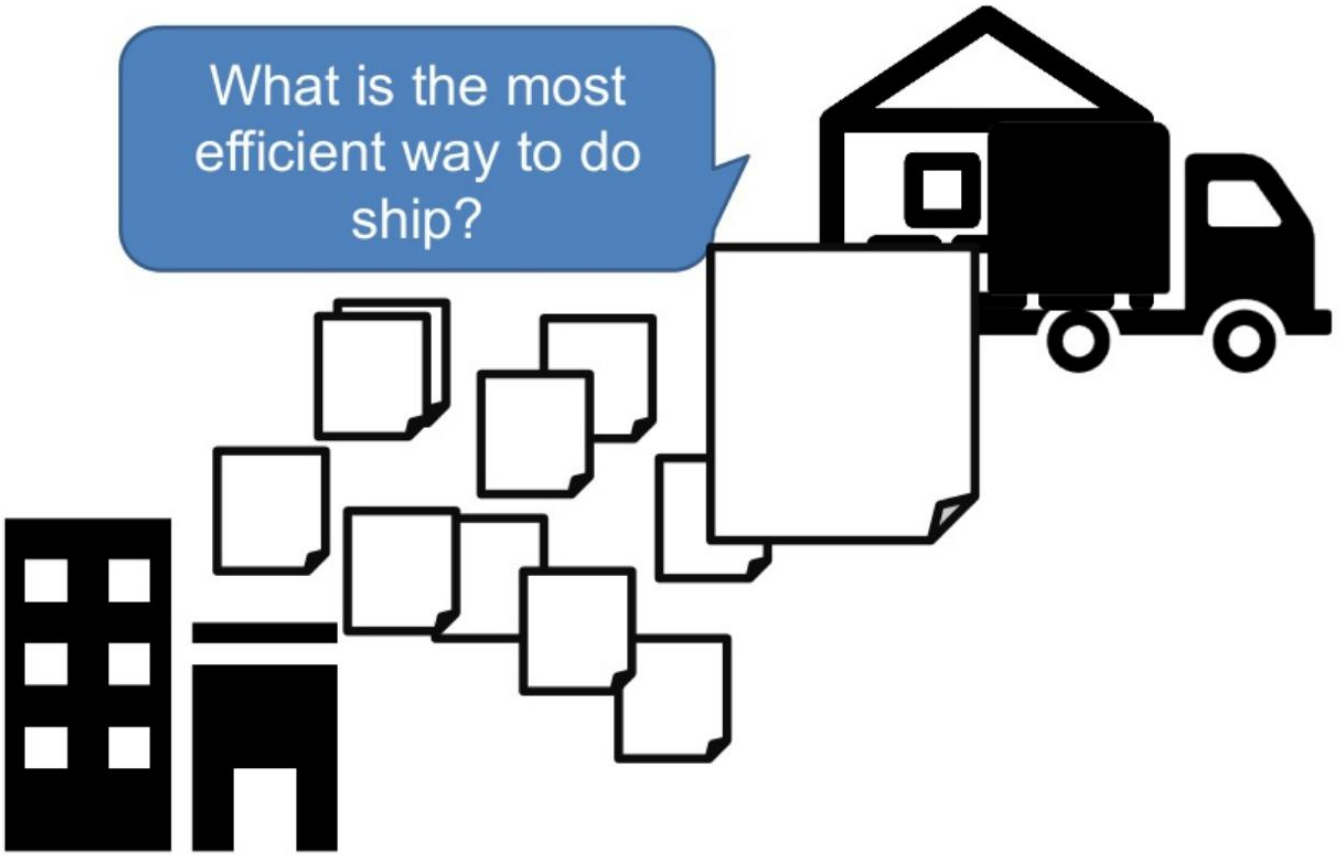


Enterprise Integration Patterns

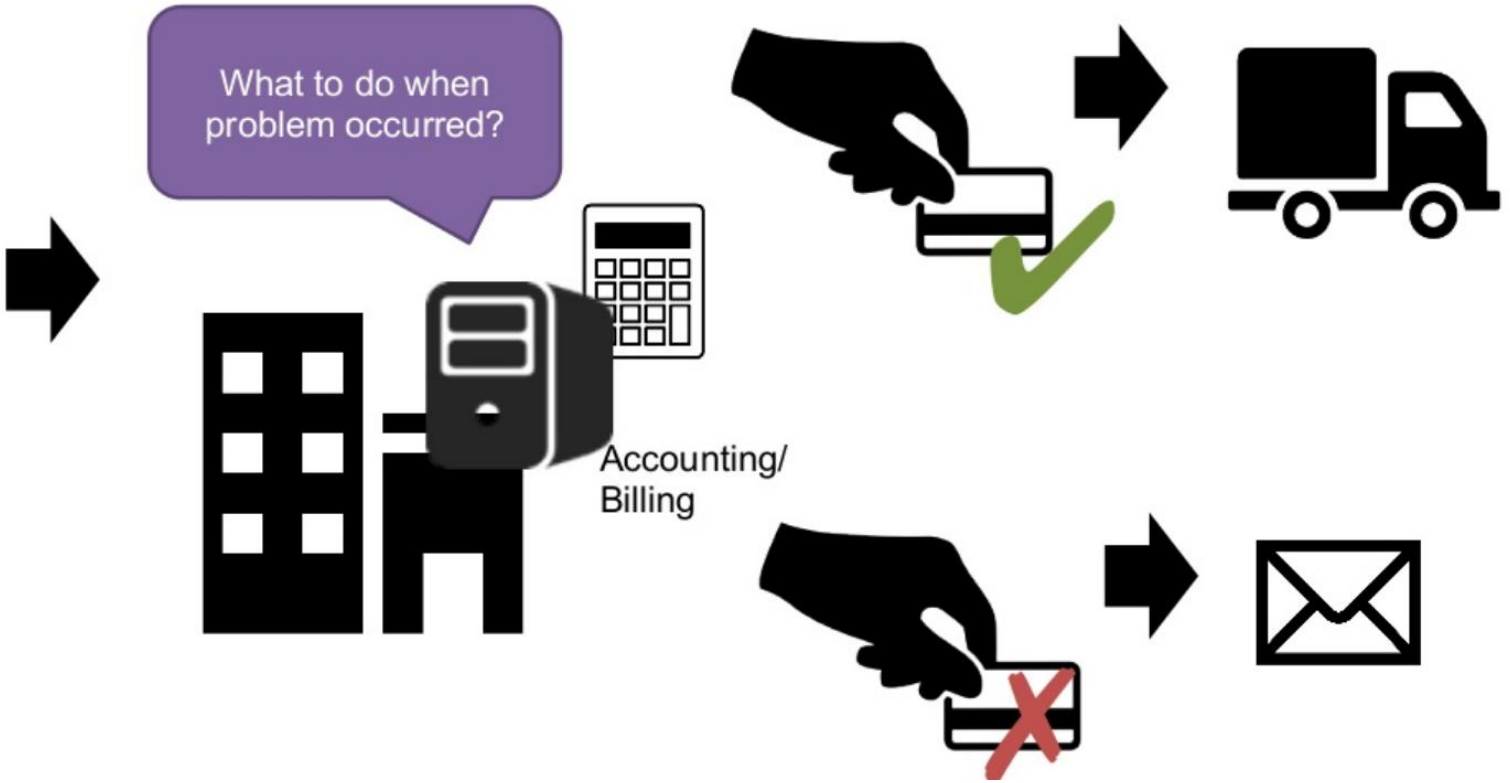


Enterprise Integration Patterns

Solving Integration Problems

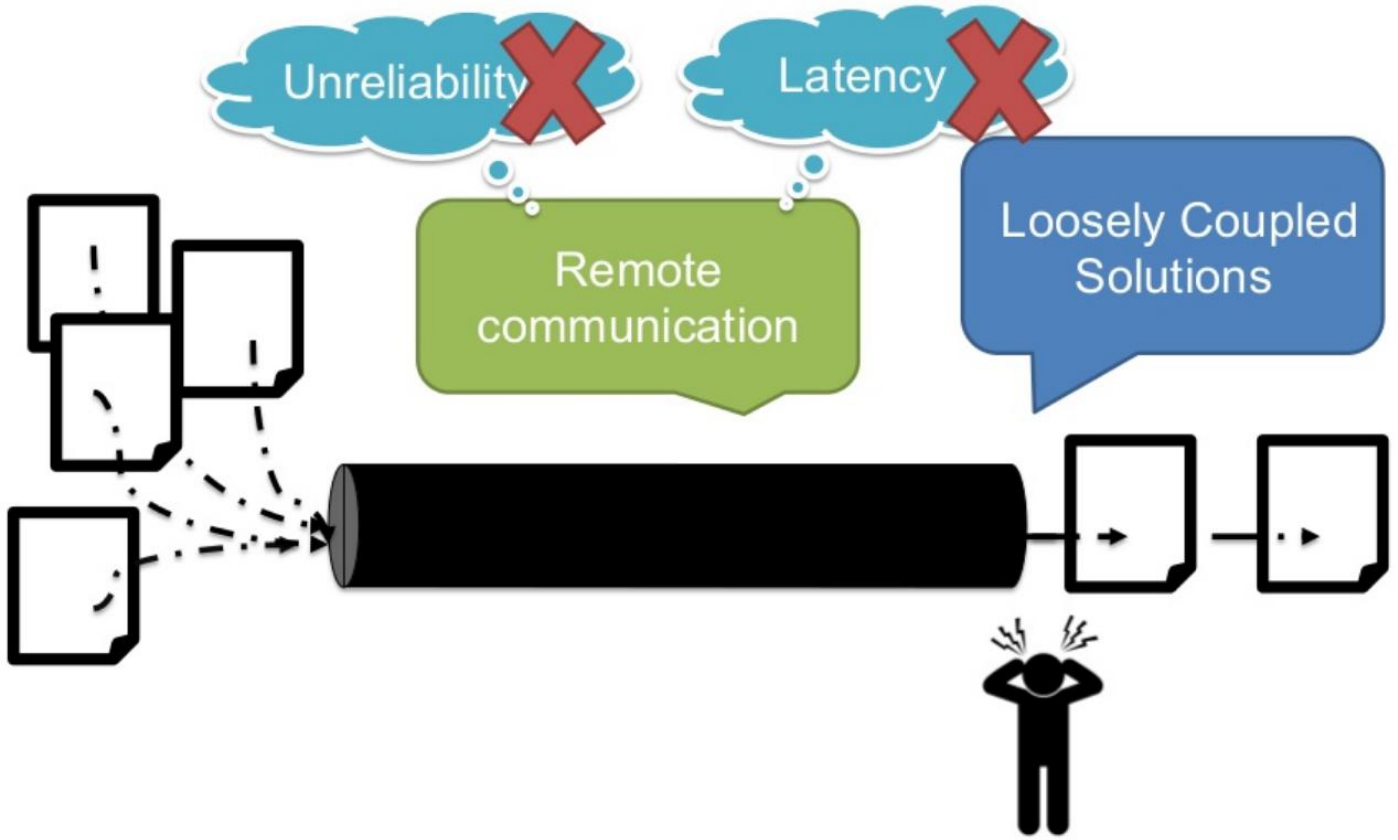


Enterprise Integration Patterns

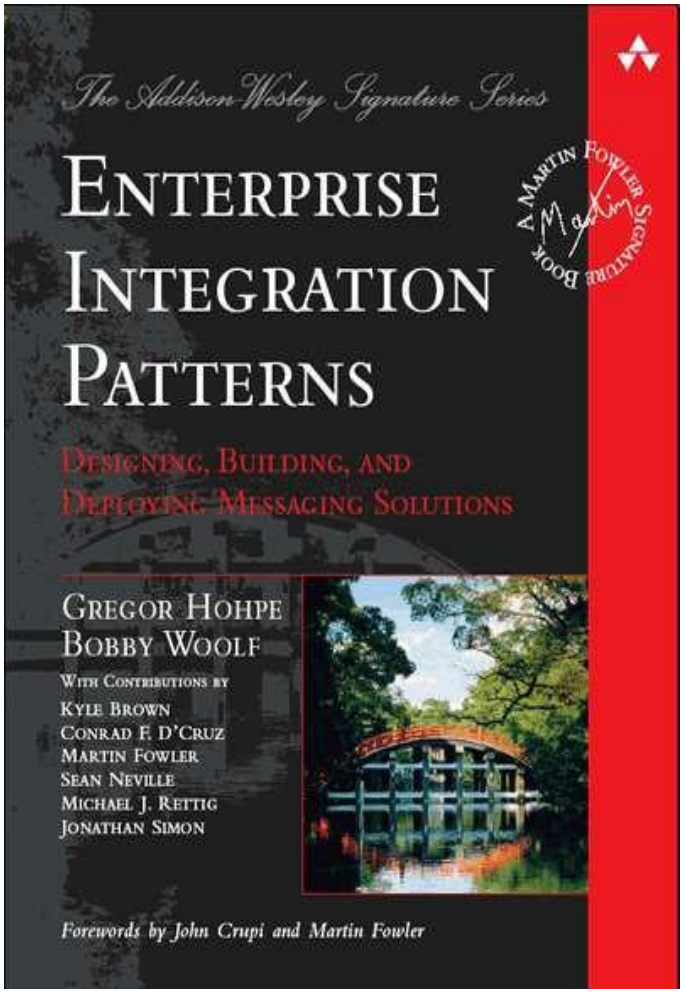


Enterprise Integration Patterns

Asynchronous Messaging Architectures



The Bible of Enterprise Integration Patterns

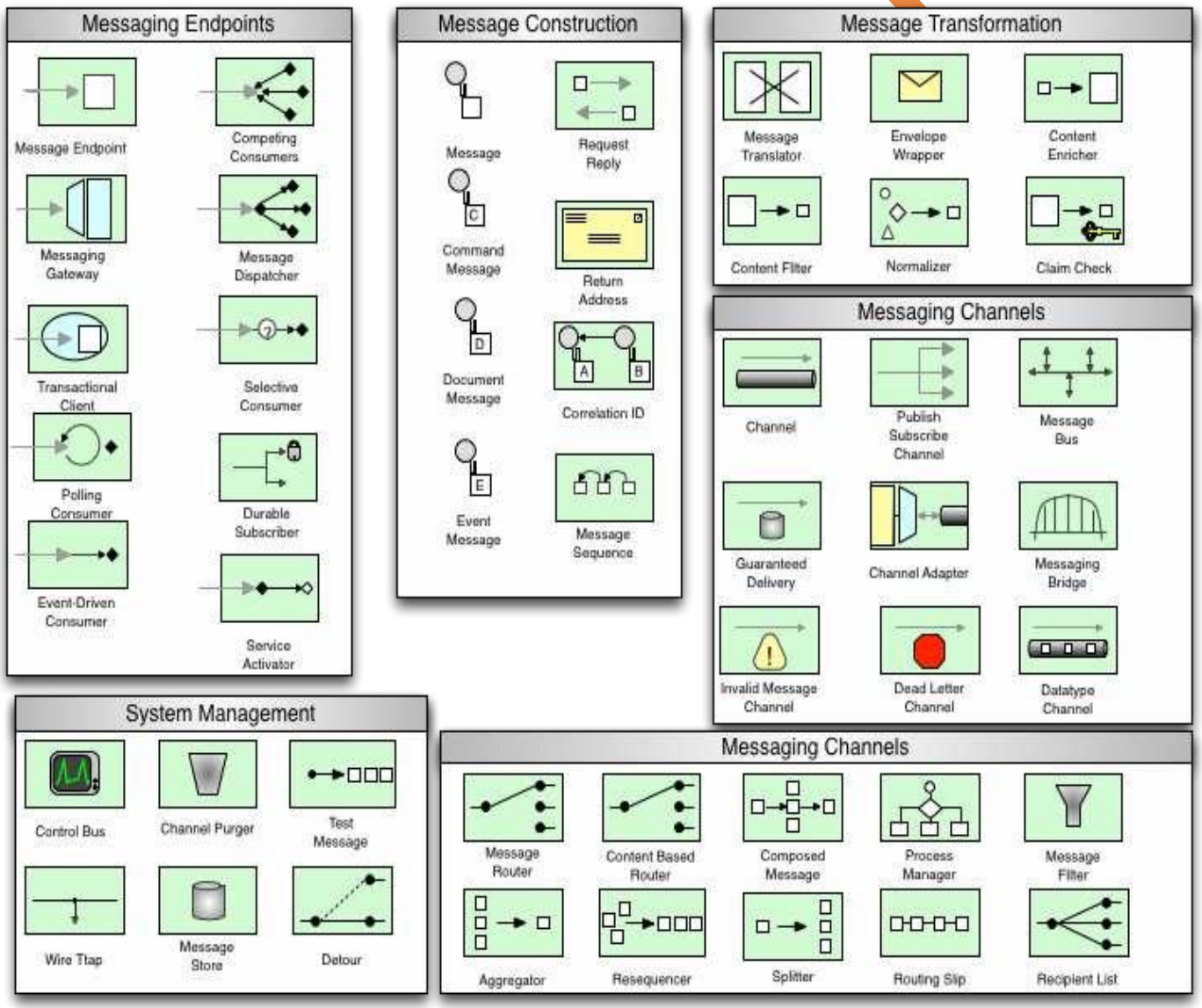


http://www.eaipatterns.com/toc.html

What is EIP?

- A design pattern is a general solution to a design problem that recurs repeatedly in many projects.
- A pattern describes the problem and its proposed solution and discuss any other important factors.
- EIP focuses on messaging patterns for enterprise application integration (EAI).
- Messaging makes it easier for programs to communicate across different programming environments (languages, compilers, and operating systems) because the only thing that each environment needs to understand is the common messaging format and protocol.
- Messaging patterns define the means by which different elements in a message-passing system connect and communicate to enable interaction among objects within programs and among various types of software -- which may be written in different languages and exist on different platforms in multiple locations

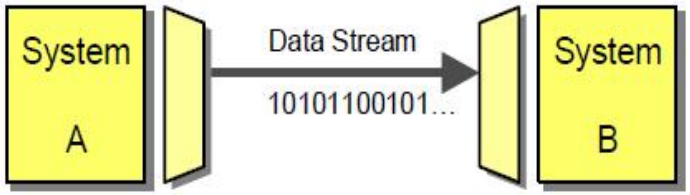
Visual Pattern Language

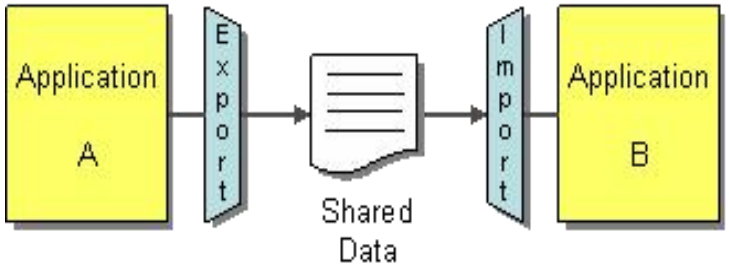


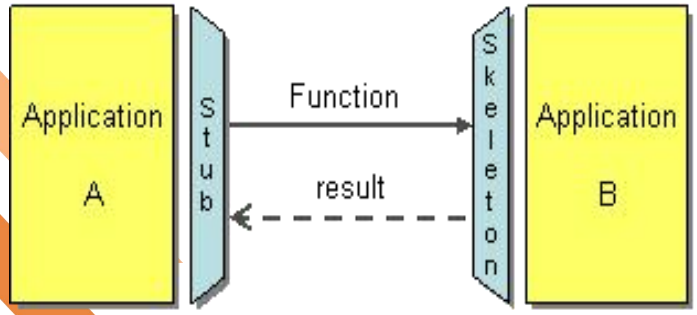
https://kodtodya.github.io/talks/

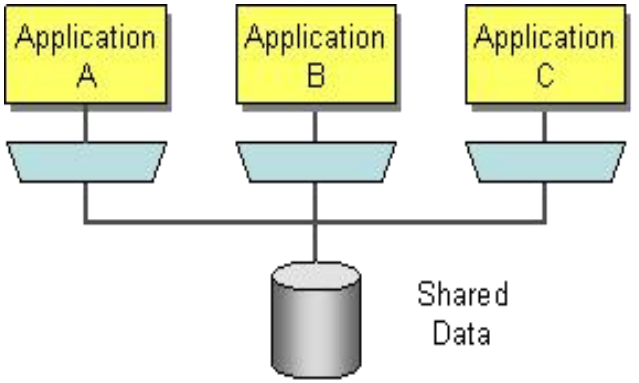
Basic Definitions

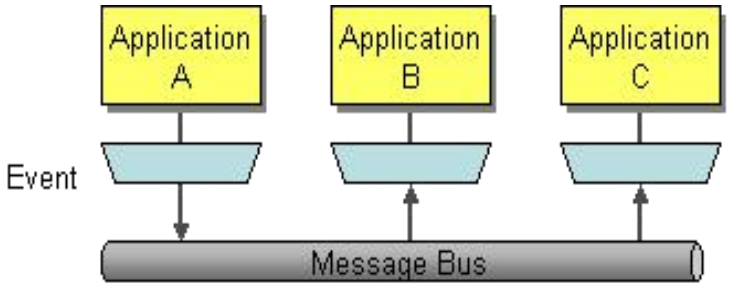
Integration styles





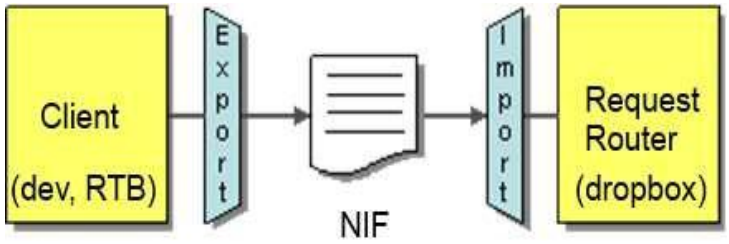


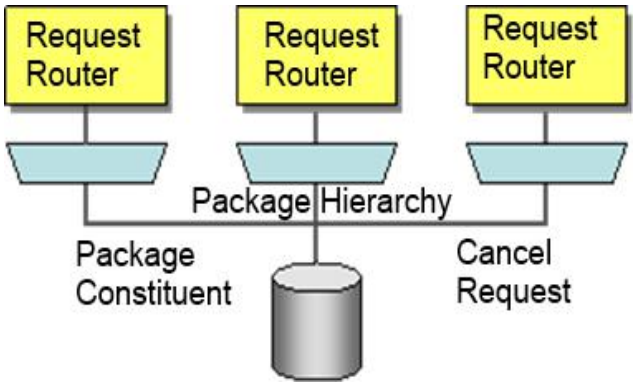


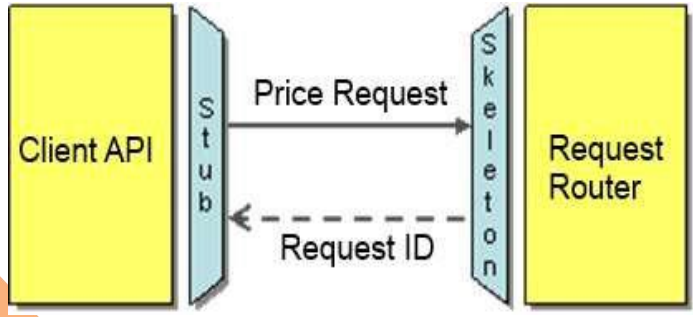


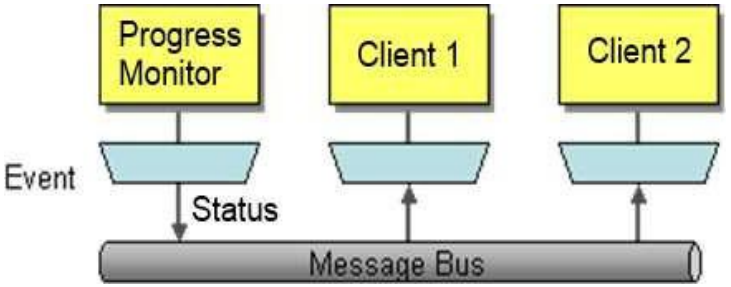
https://kodtodya.github.io/talks/

In Nova





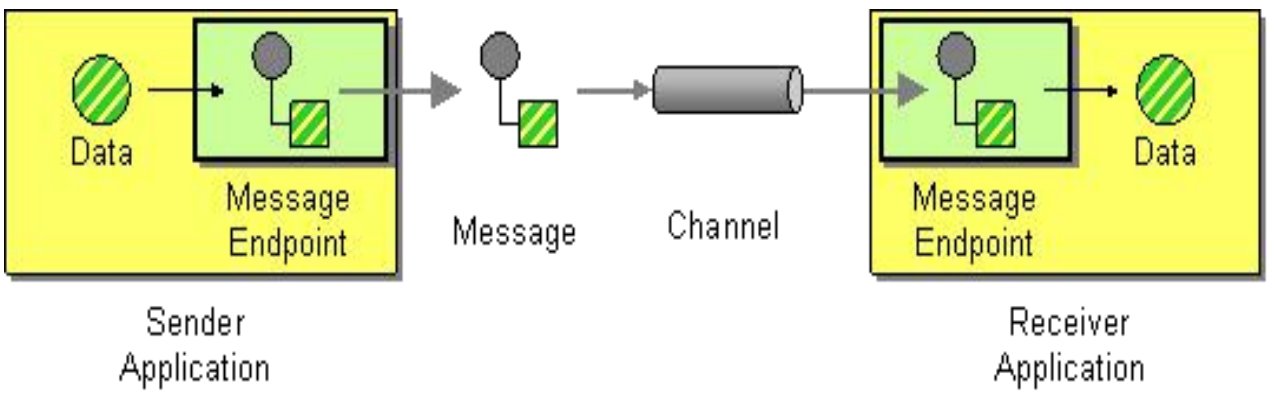




https://kodtodya.github.io/talks/

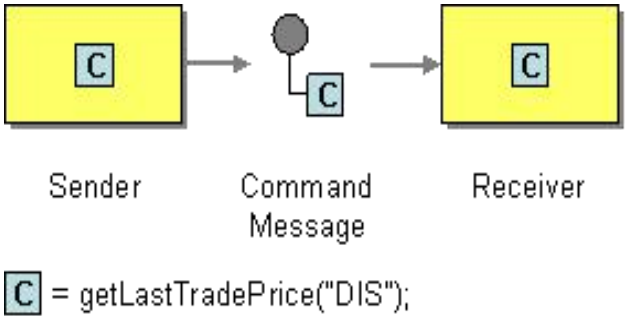
Main building blocks

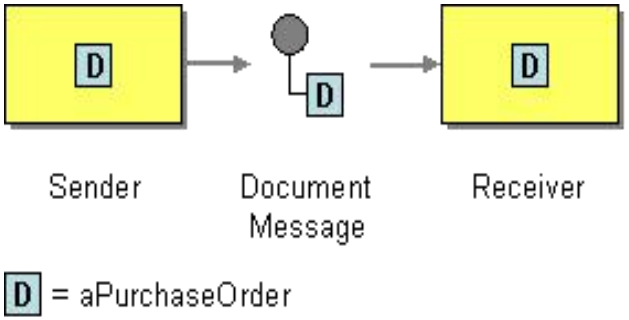
- **Endpoint**
- **Channel**
- **Message**

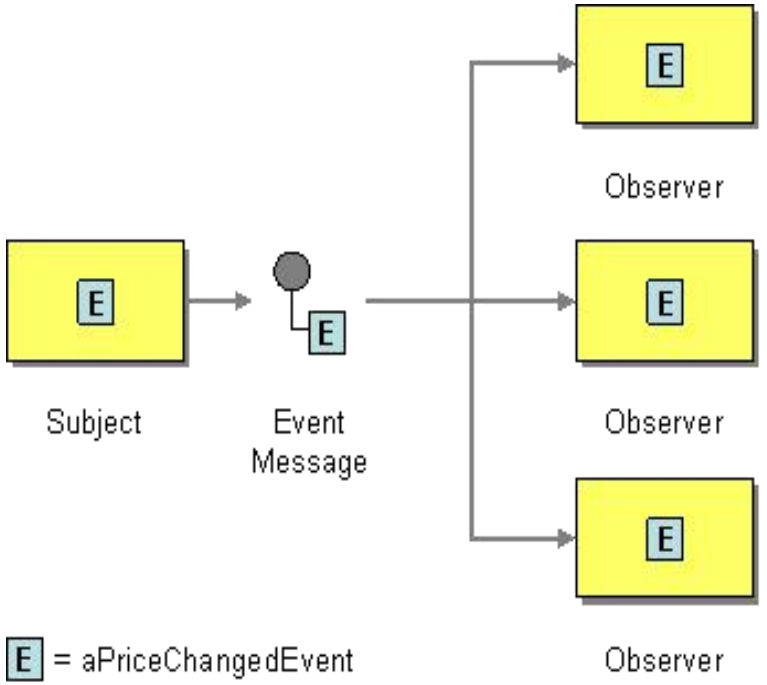


Messages

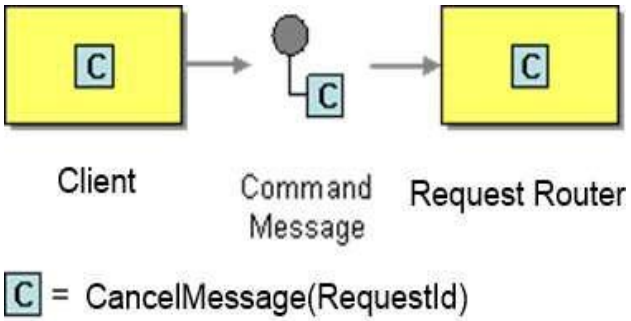
Message types

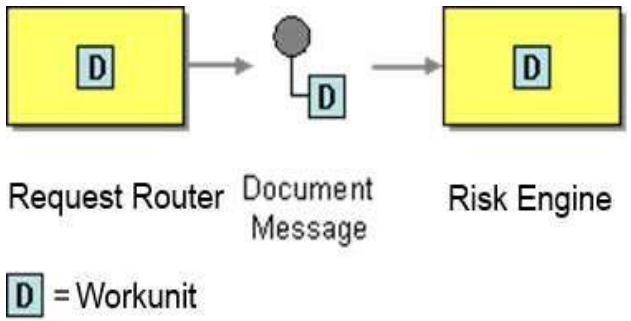


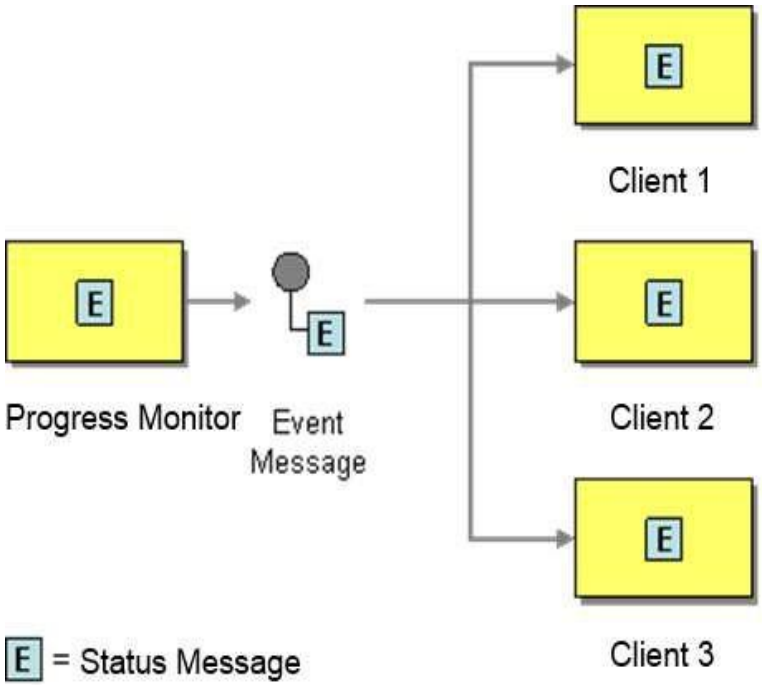




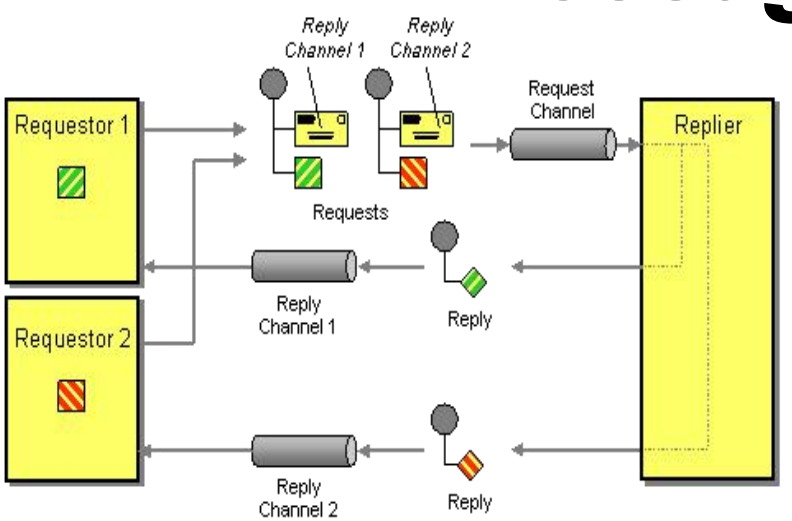
In Nova


C
C
C
Client
Command Message
Request Router
C = CancelMessage(RequestId)

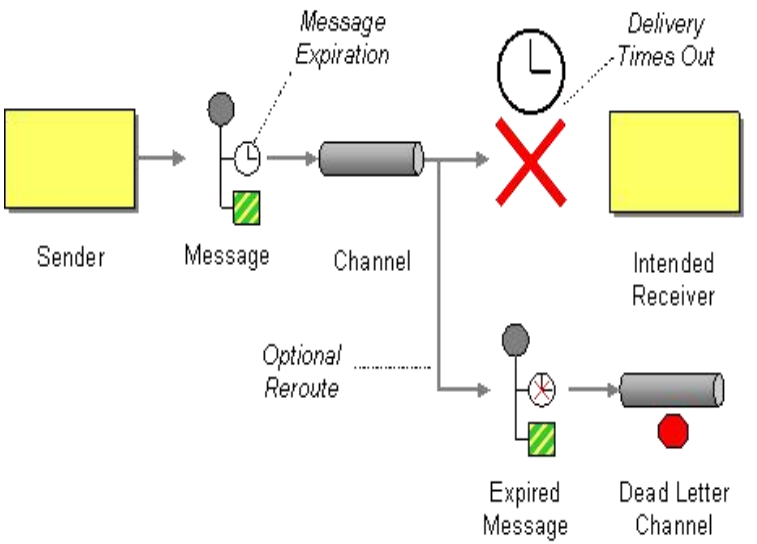

D
D
D
Request Router
Document Message
Risk Engine
D = Workunit


E
E
E
Progress Monitor
Event Message
E
E
E
Client 1
Client 2
Client 3
E = Status Message

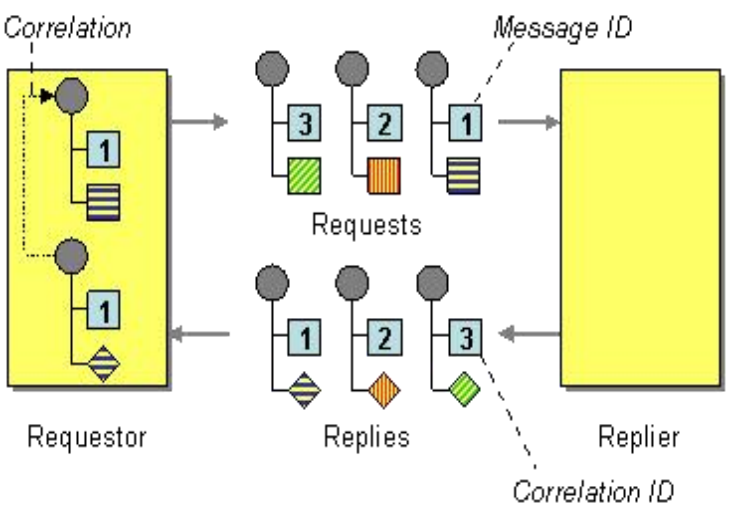
Message attributes



Return address



Expiration time

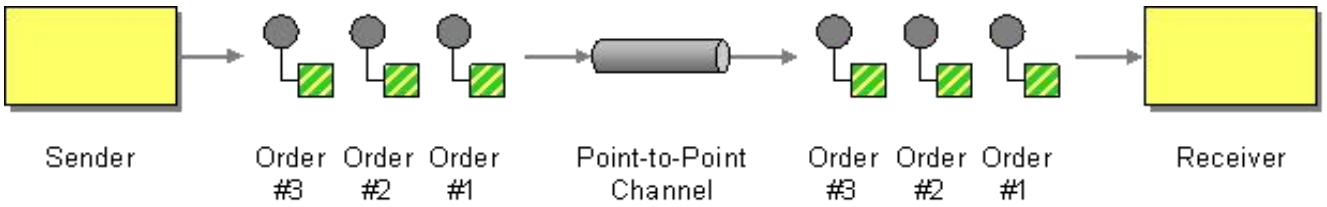


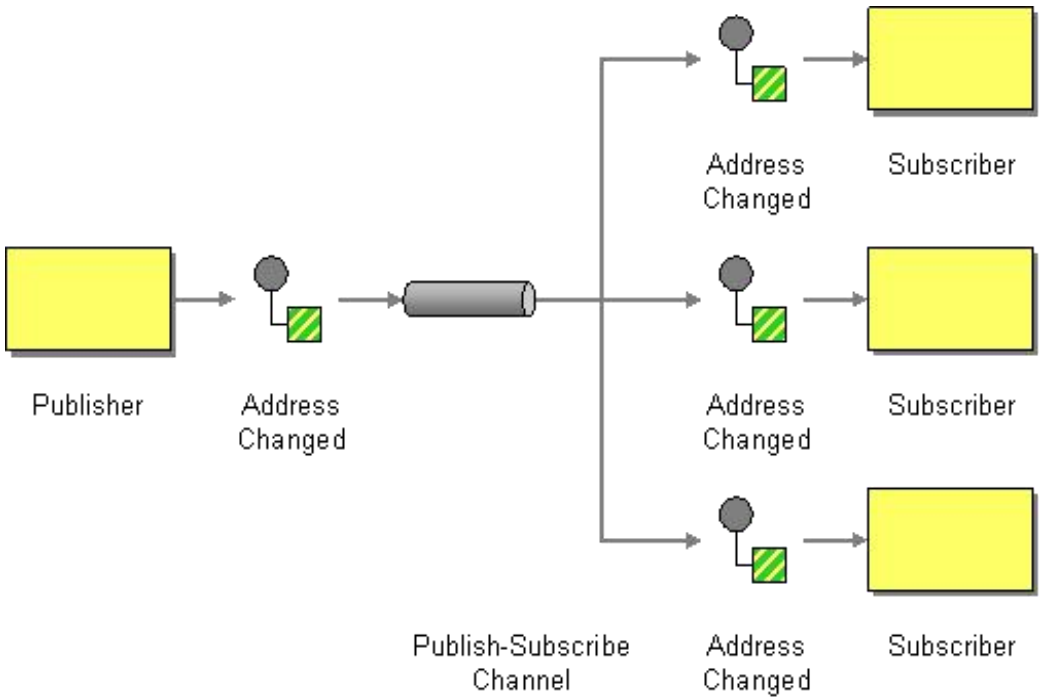
Correlation ID

https://kodtodya.github.io/talks/

Messaging Channels

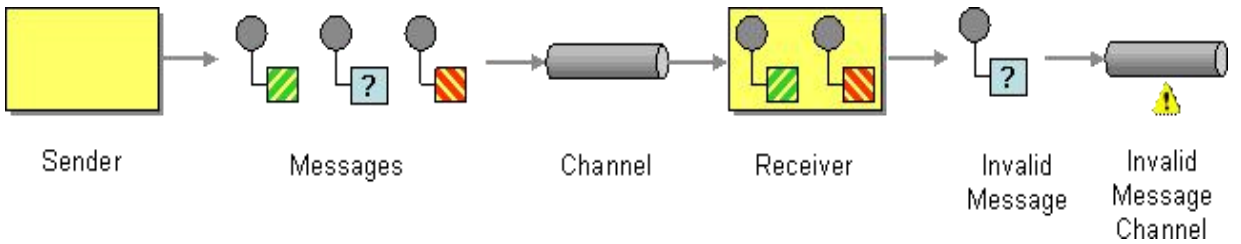
Message exchange styles



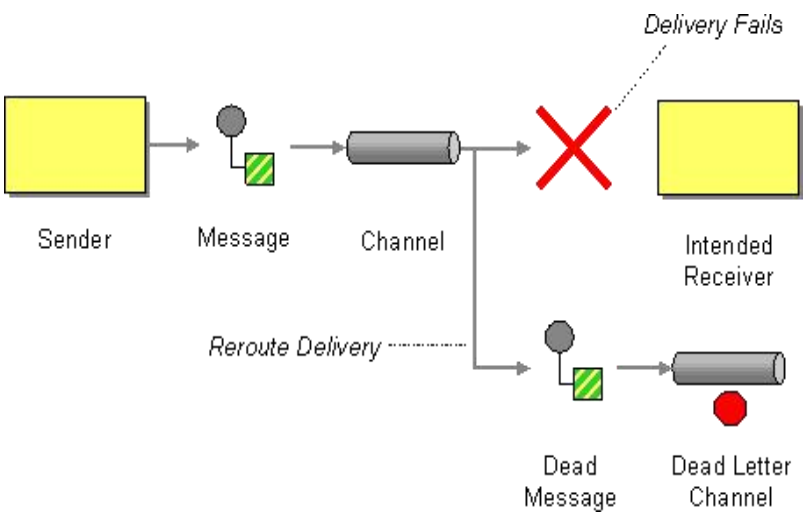


https://kodtodya.github.io/talks/

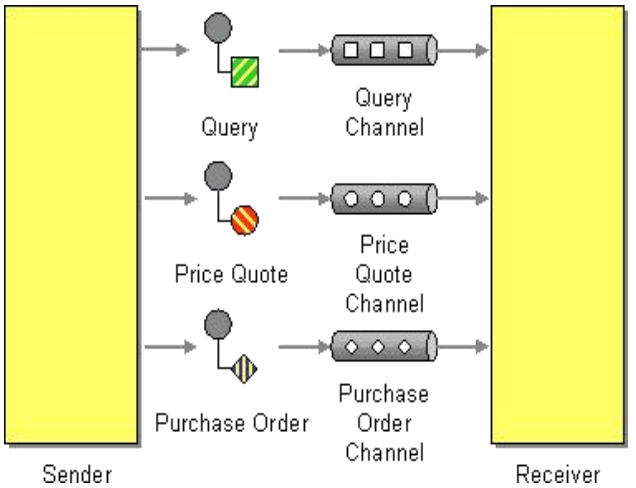
Invalid Message Channel



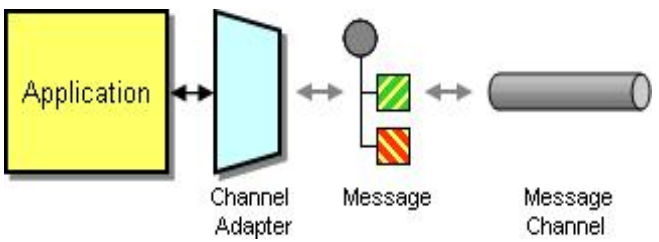
Dead Letter Channel



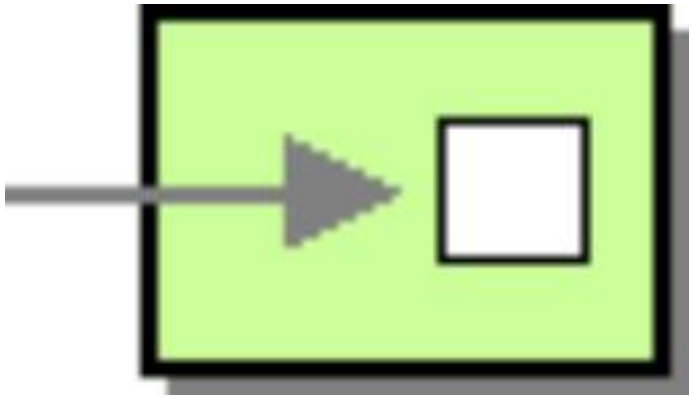
Datatype Channel



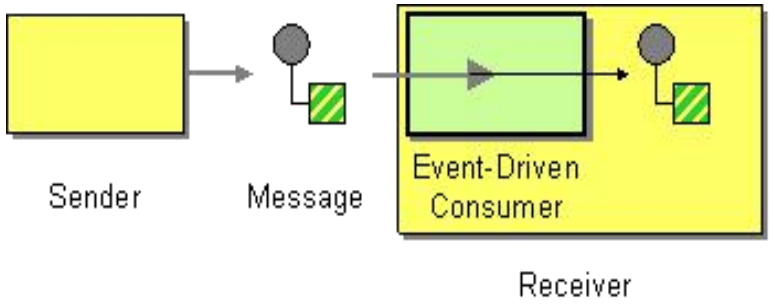
Channel Adapter

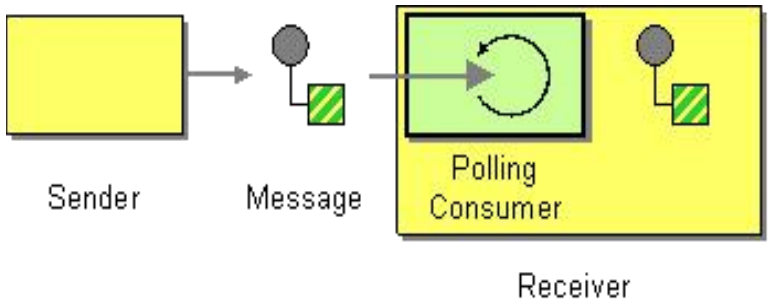


Message Endpoints

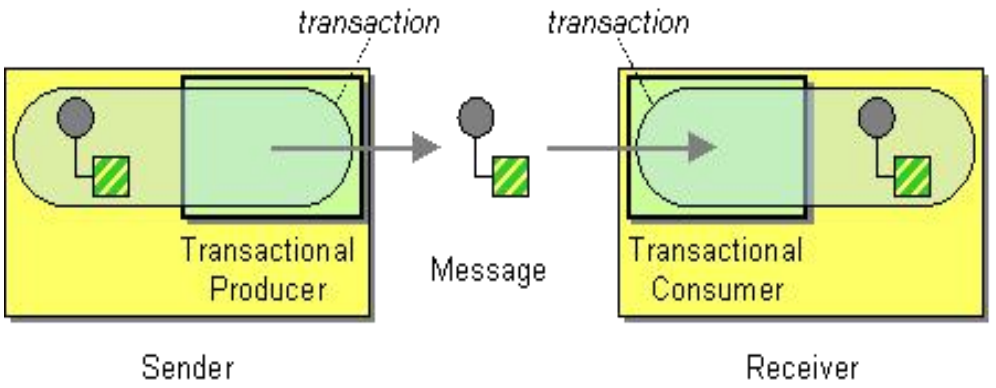


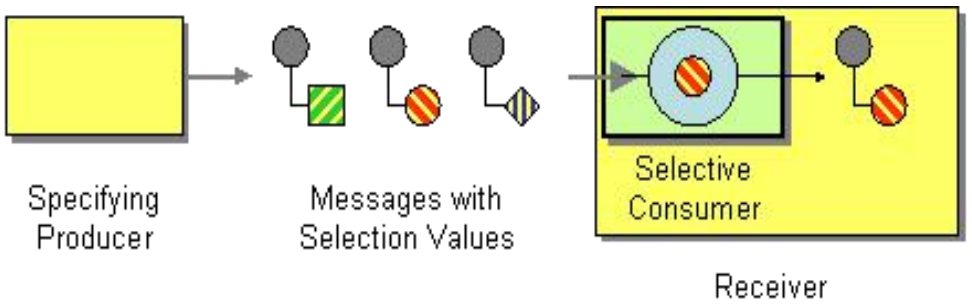
Messaging endpoints





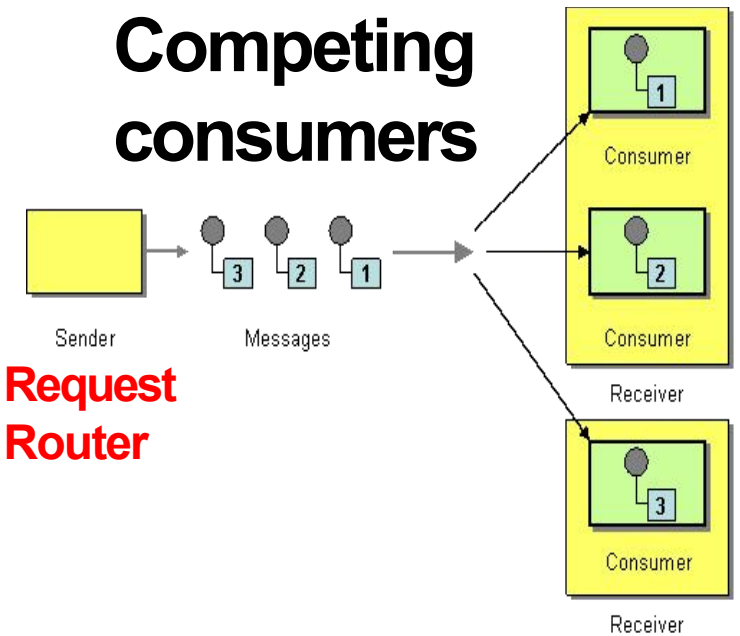
Messaging endpoints





Messaging endpoints

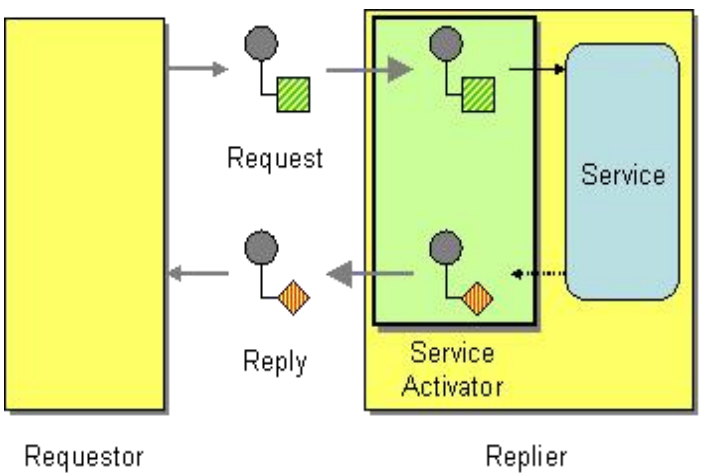
Competing consumers



Request Router

Risk Engine Farm

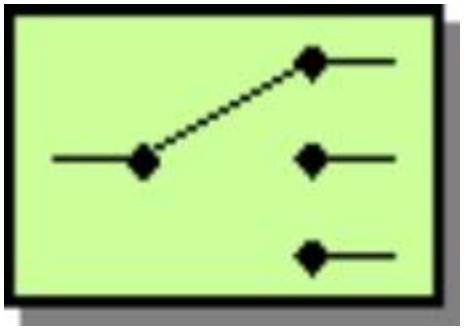
Service activator



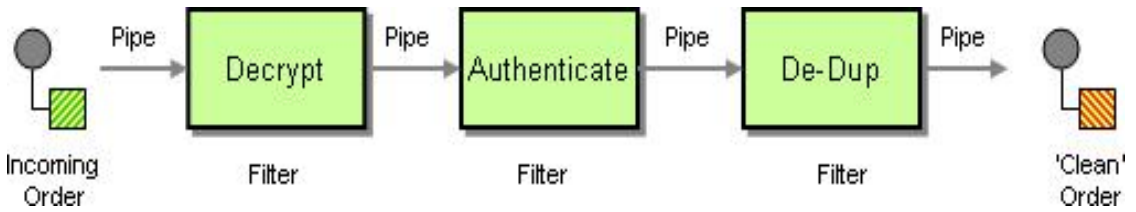
Pattern categories

- Message routing patterns
- Message transformation patterns
- Message management patterns

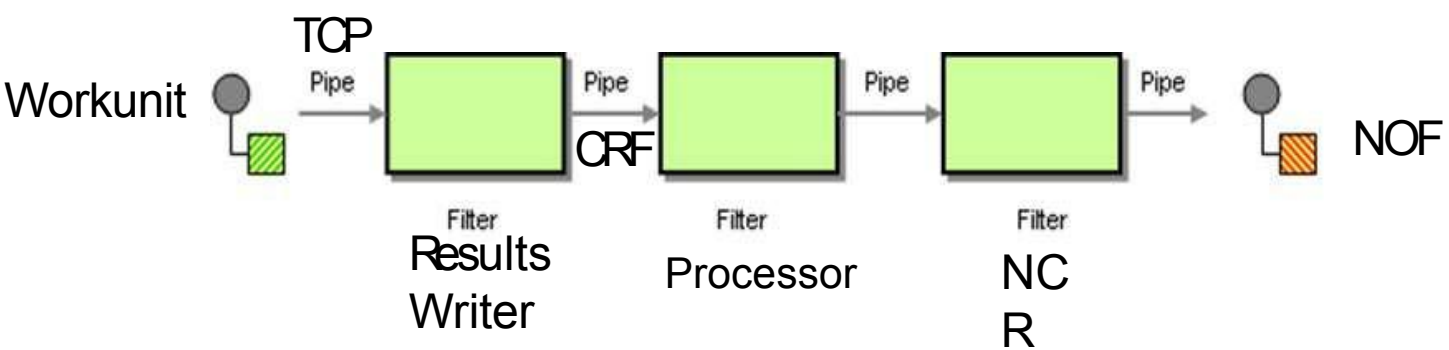
Message routing patterns



Pipes and filters

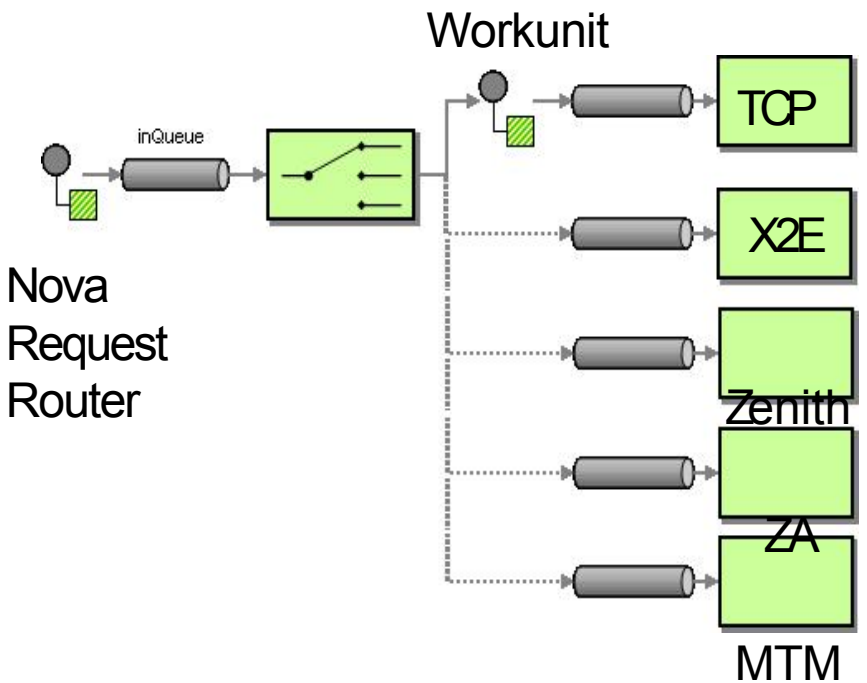


In Nova



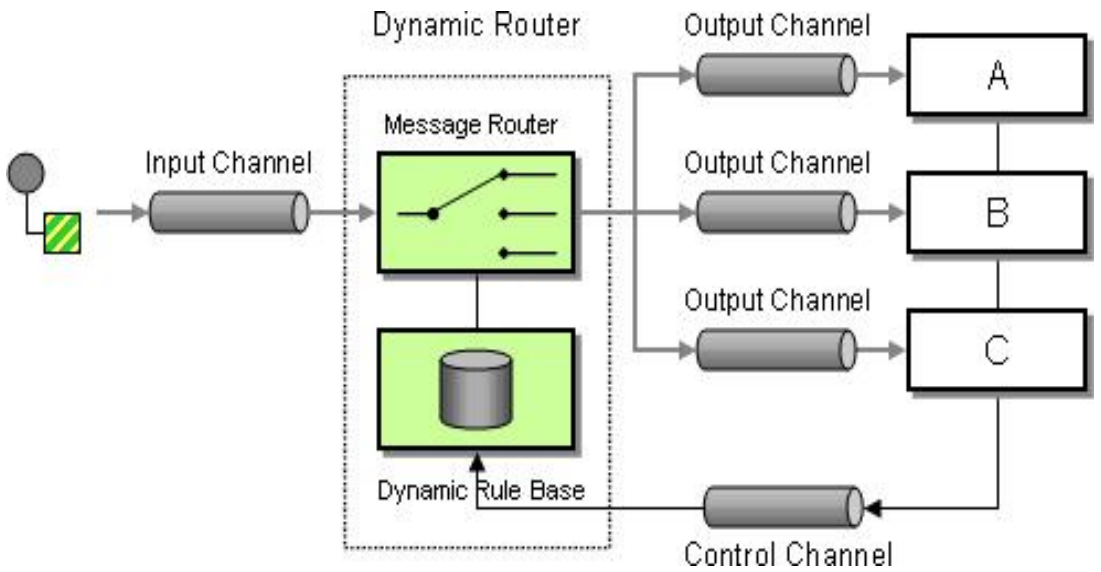
NCRF - Non-Cancelled Request Filter

Content-Based Router

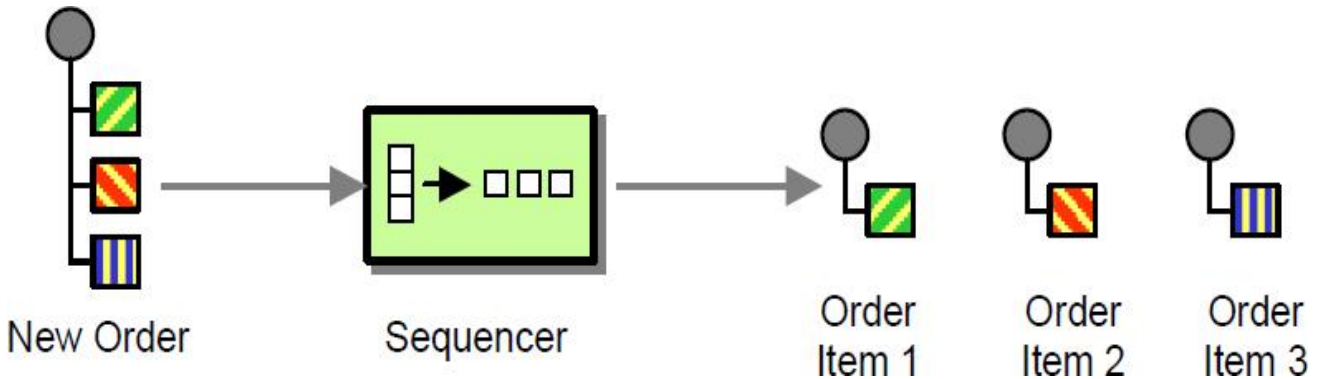


Instrument-Valuation Request

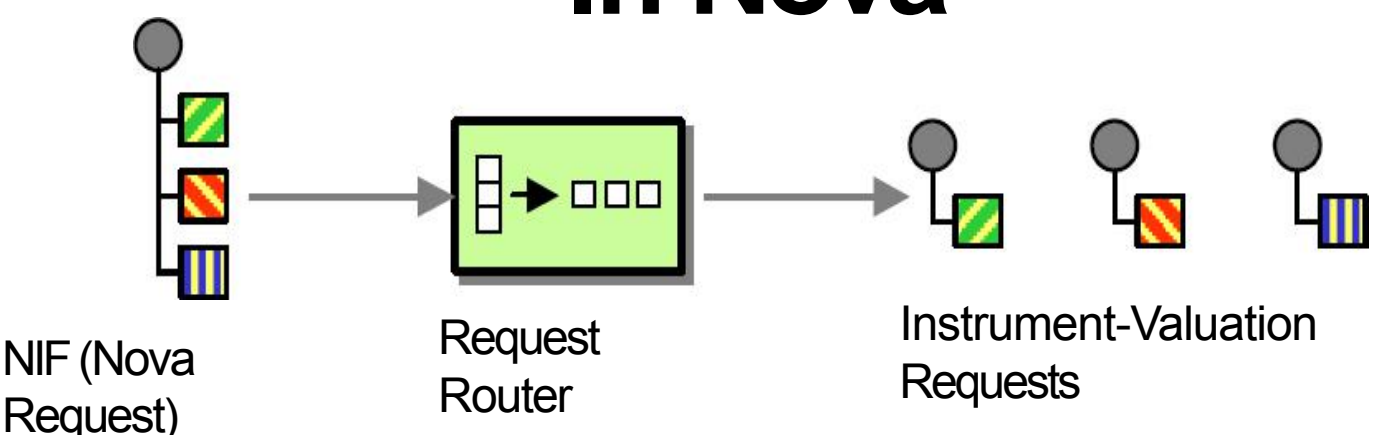
Dynamic Router



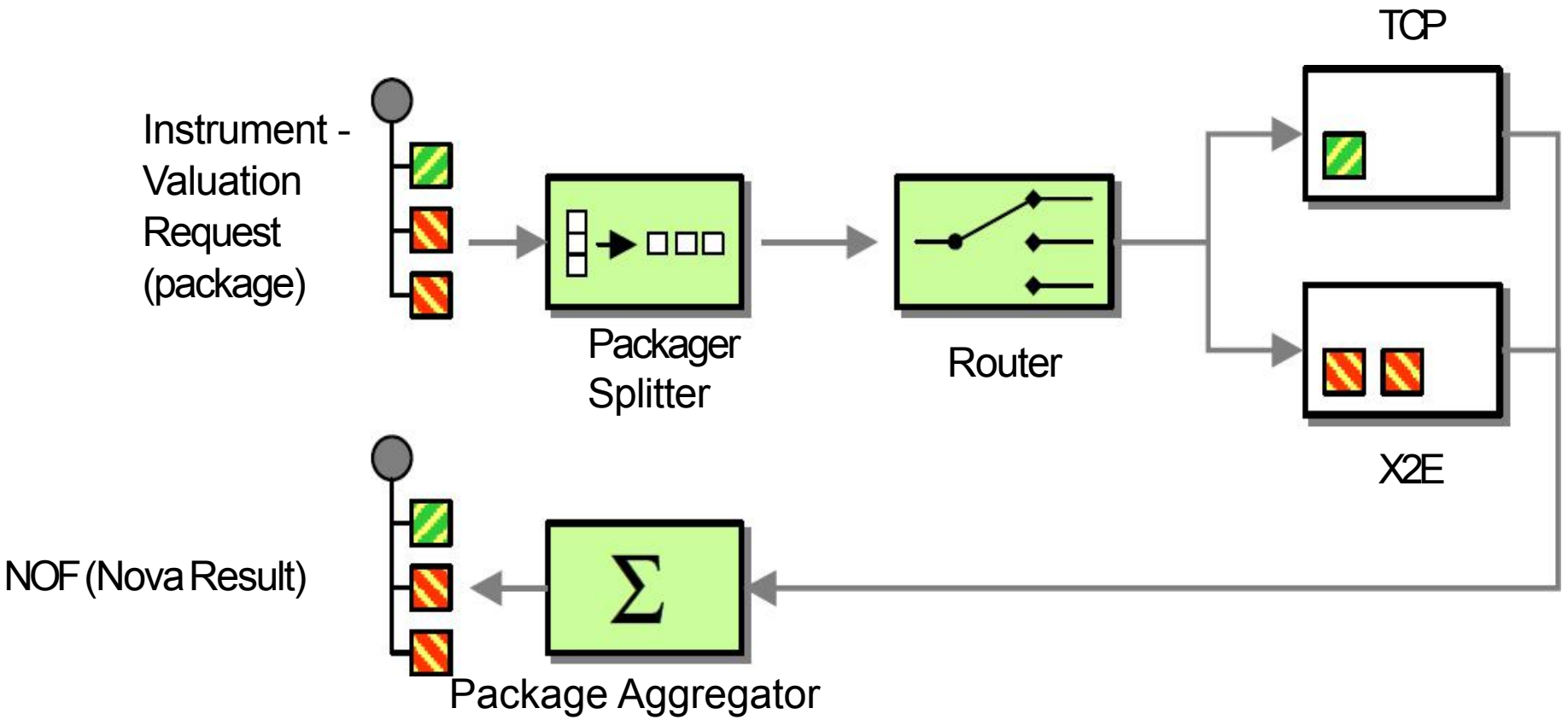
Sequencer (Splitter)



In Nova



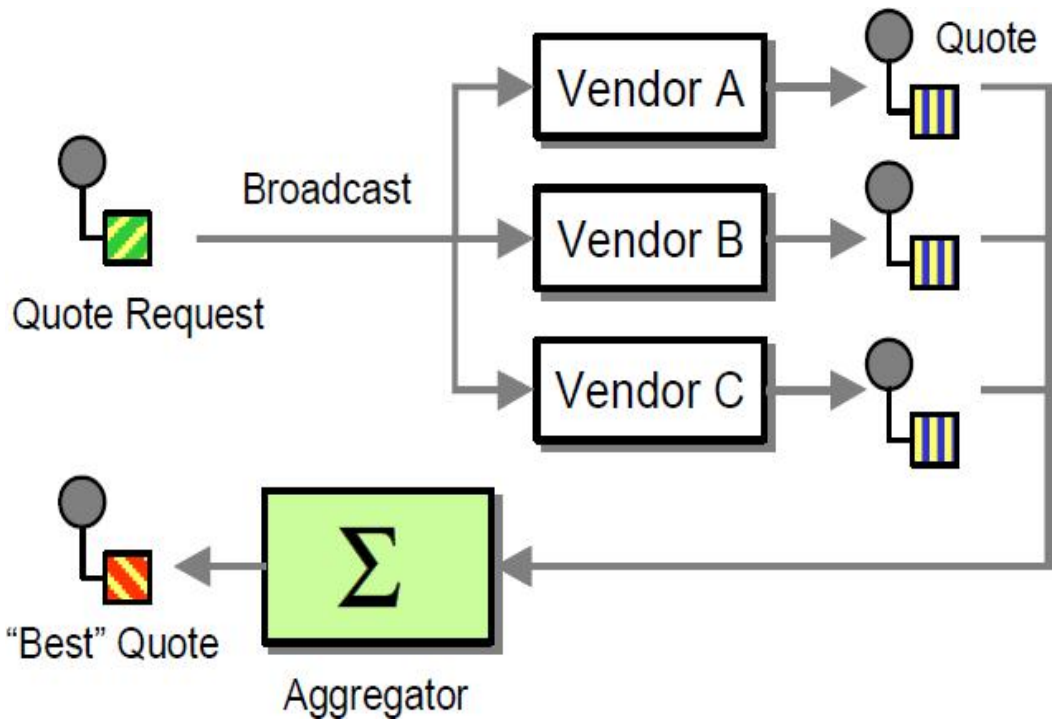
Aggregator


TCP
Instrument -
Valuation
Request
(package)
Packager
Splitter
Router
X2E
NOF (Nova Result)
Σ
Package Aggregator

In Nova - Batching

- ILS, Snap Service, Zenith
- Correlation ID = Request ID
- Completion = on time-out | | on max count

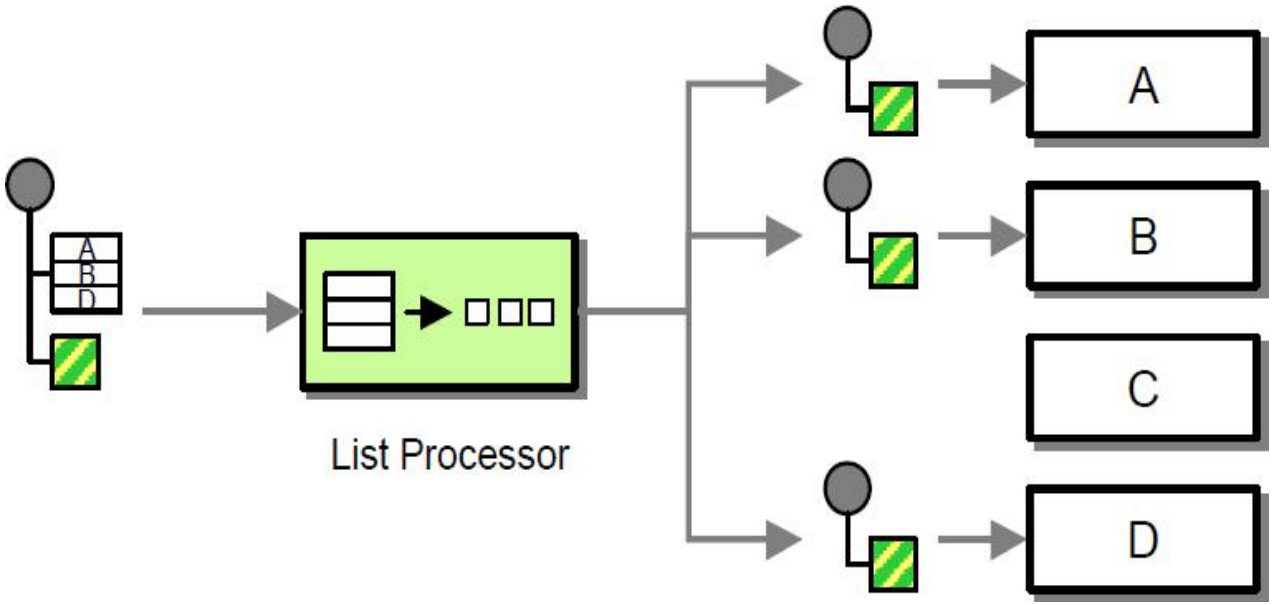
Broadcast with Aggregate Response



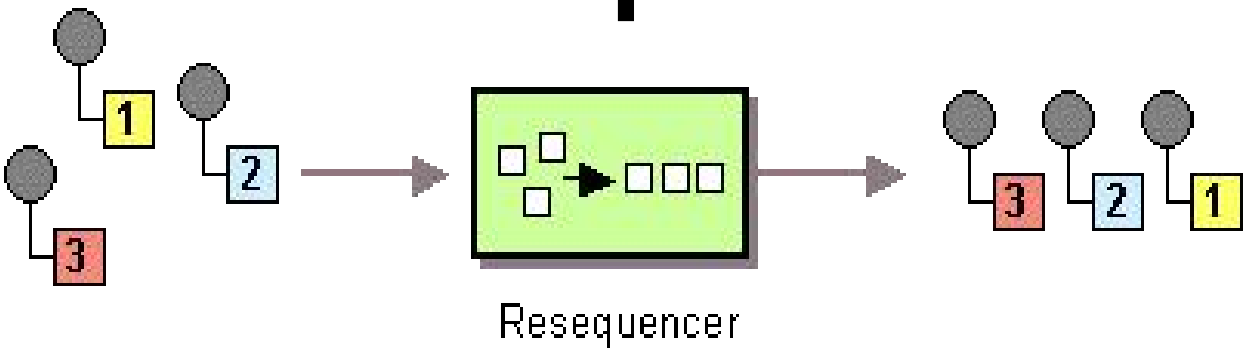
Completion criterion:

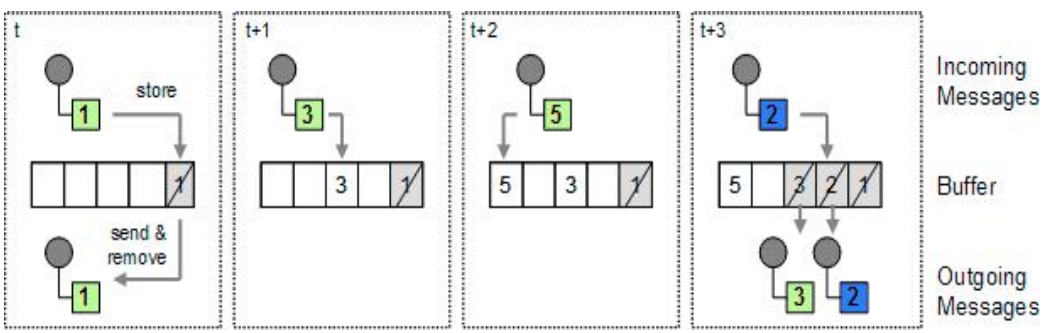
- Timeout
- Count
- External event

Recipient List



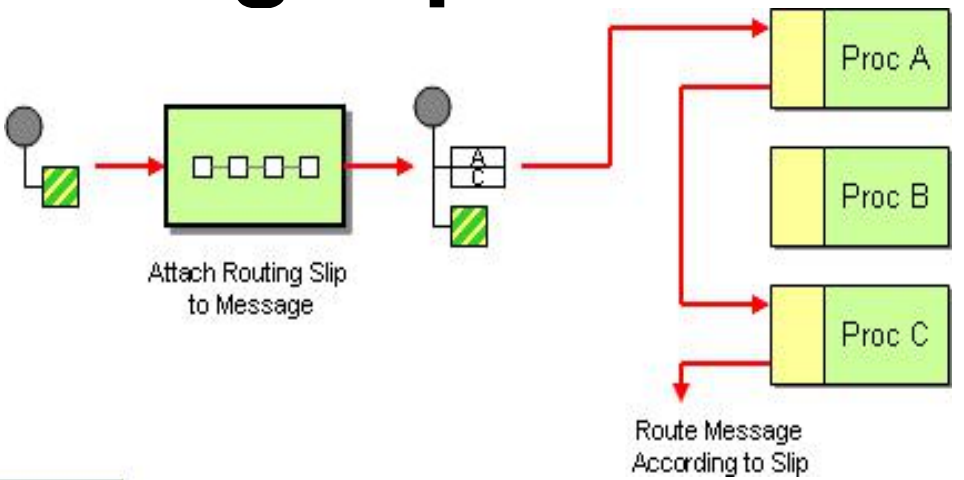
Resequencer



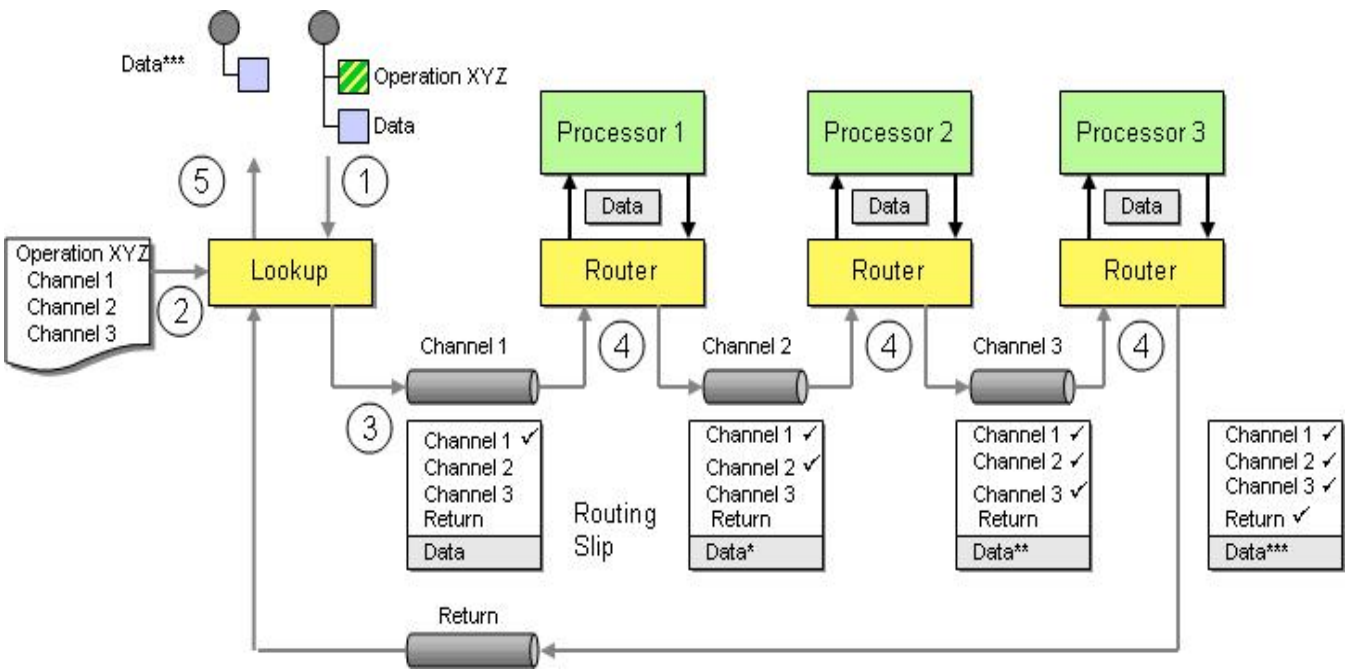


Example – TCP datagrams

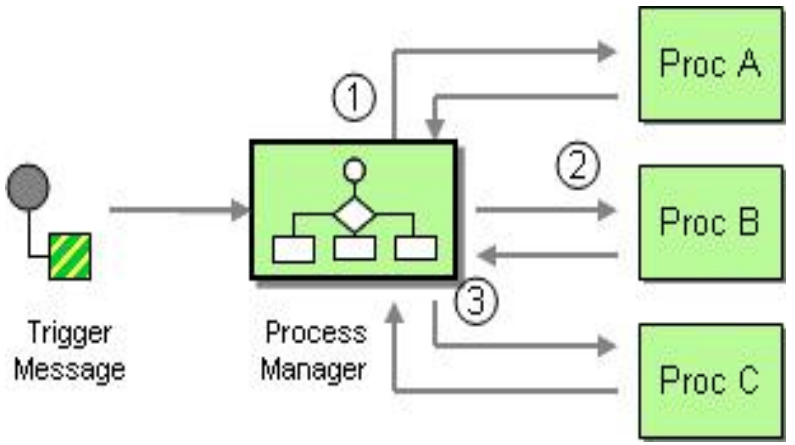
Routing Slip



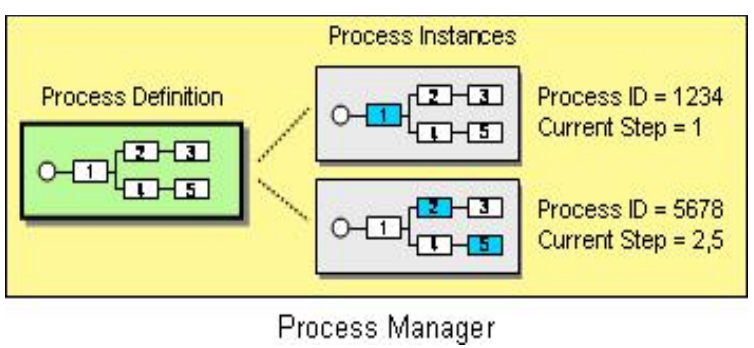
Linear flow

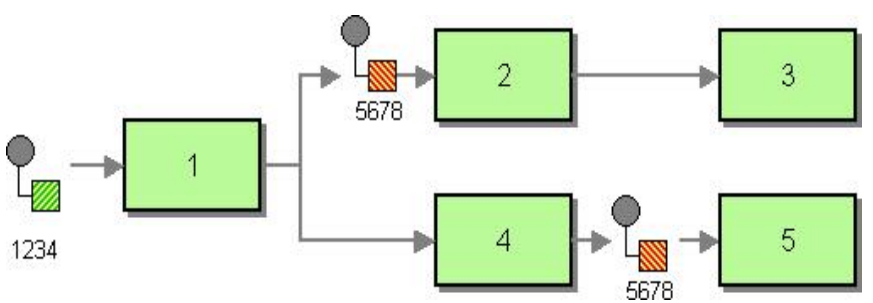


Process Manager



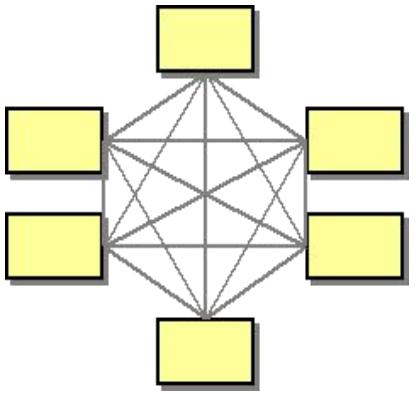
Complex message flow



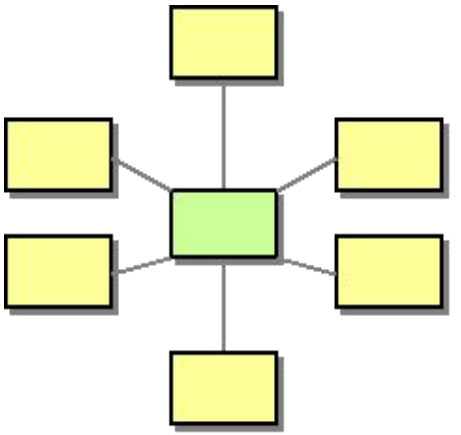


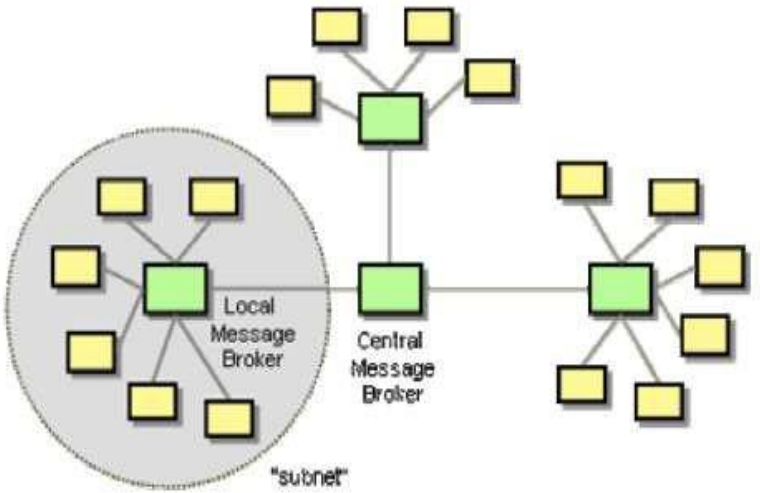
https://kodtodya.github.io/talks/

Message broker



Integration Spaghetti as a Result of Point-to-Point Connections



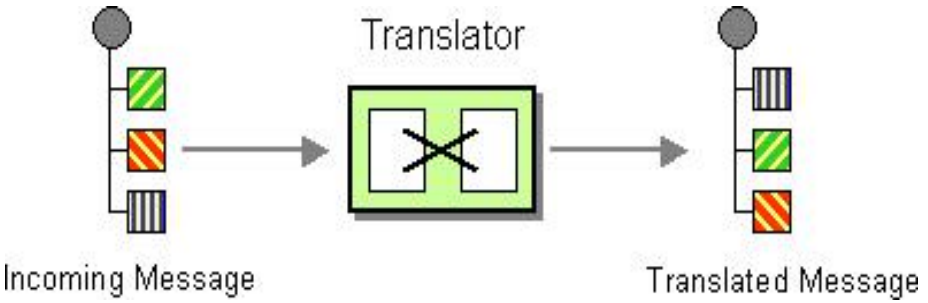


A Hierarchy of Message Brokers Provides Decoupling while Avoiding the "Über-Broker"

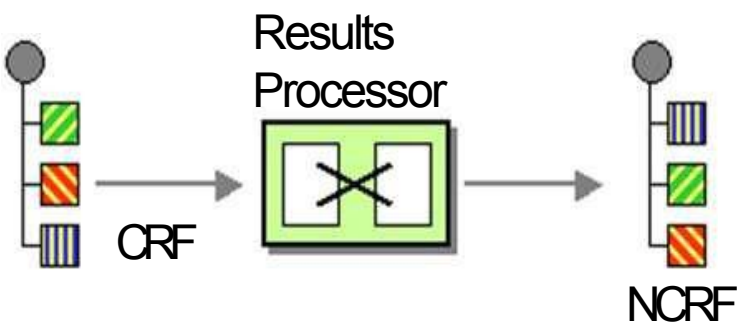
Message Transformation Patterns

https://kodtodya.github.io/talks/

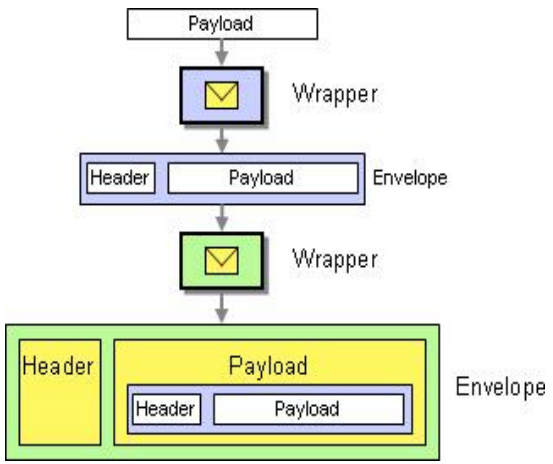
Message Translator

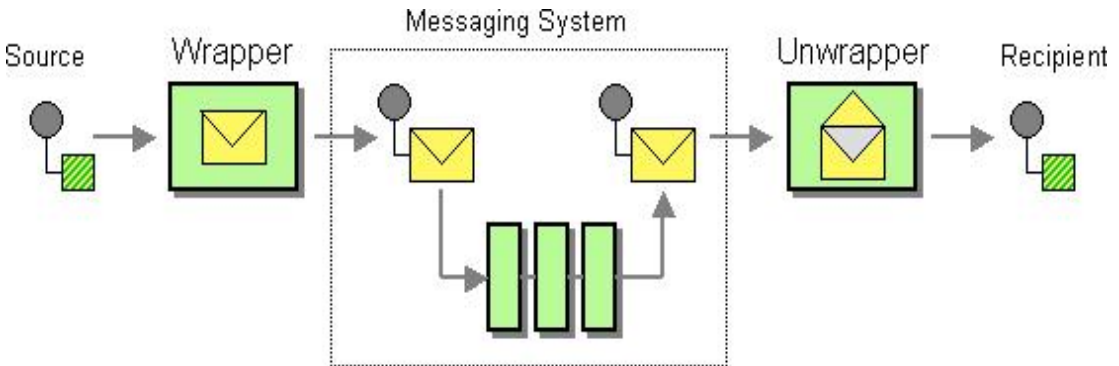


In Nova

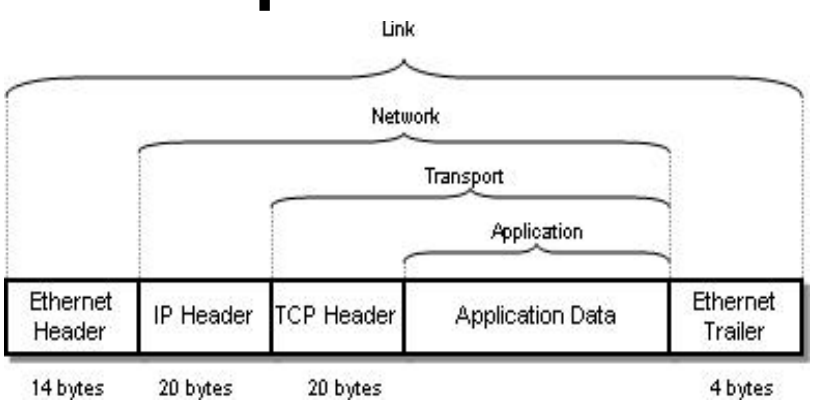


Envelope Wrapper / Un-wrapper



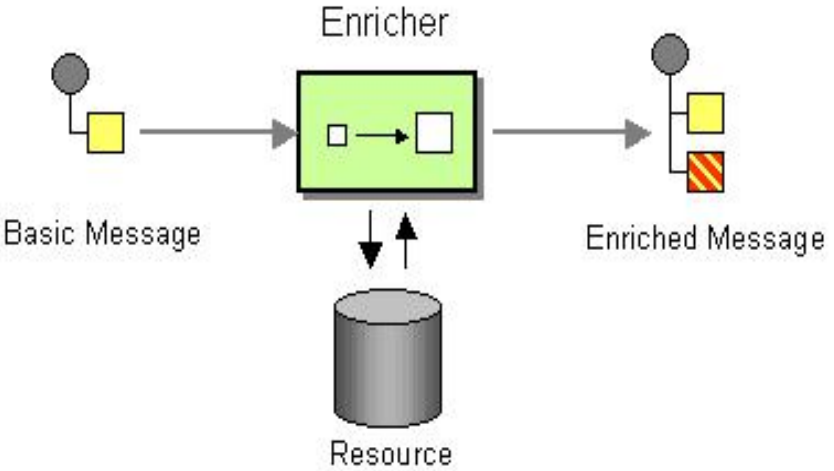


Example: TCP/IP

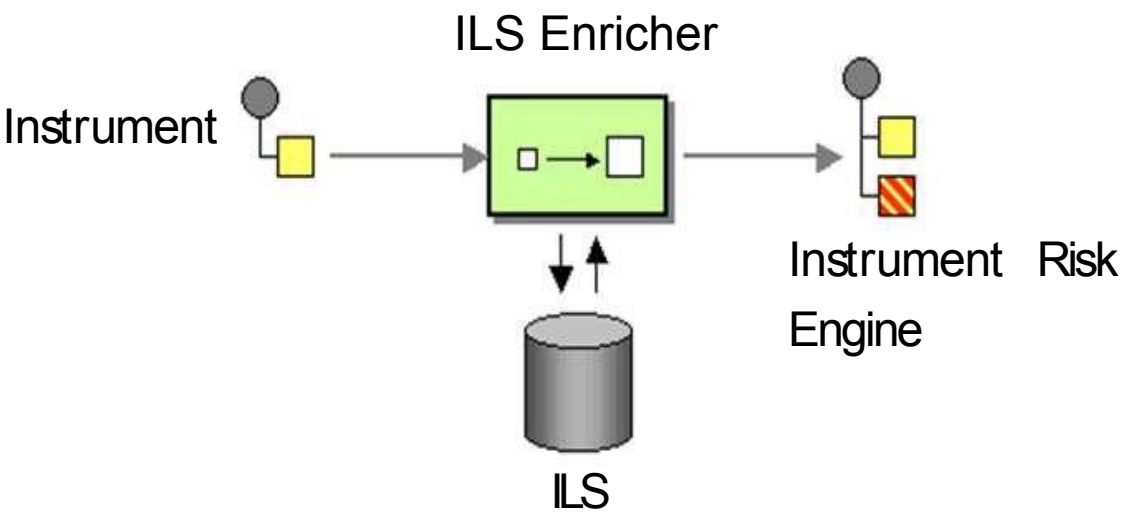


https://kodtodya.github.io/talks/

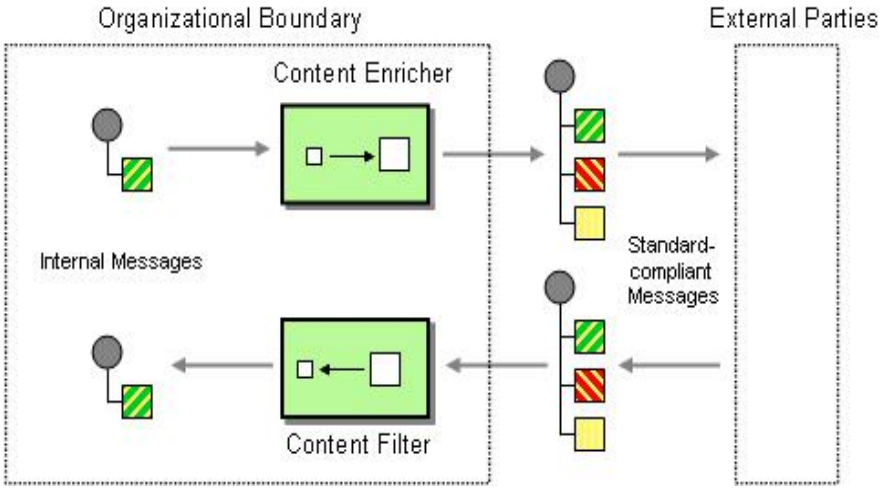
Content Enricher



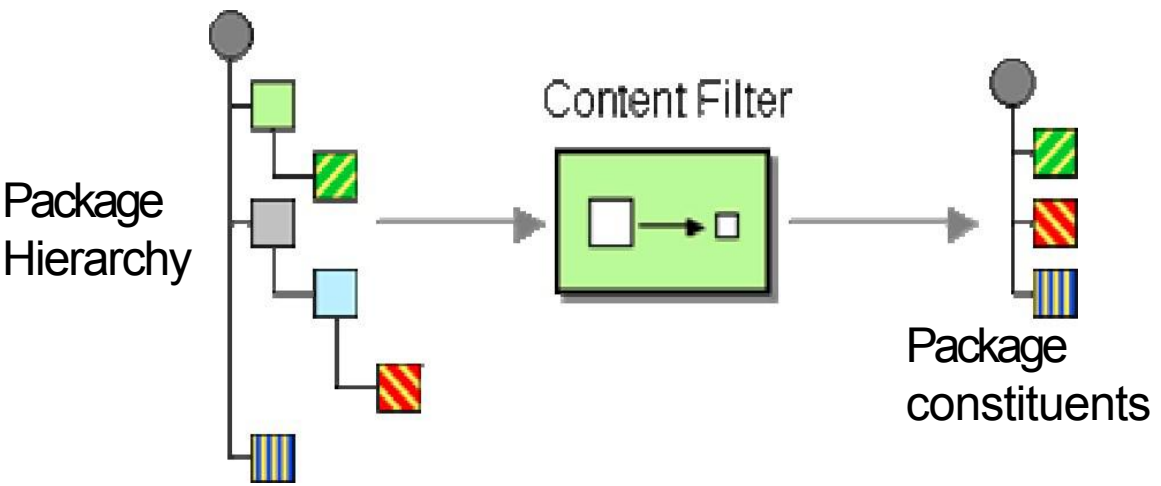
In Nova



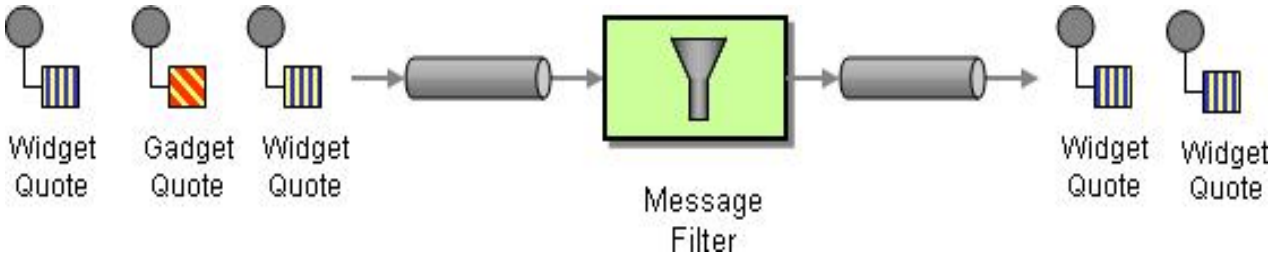
Content Filter



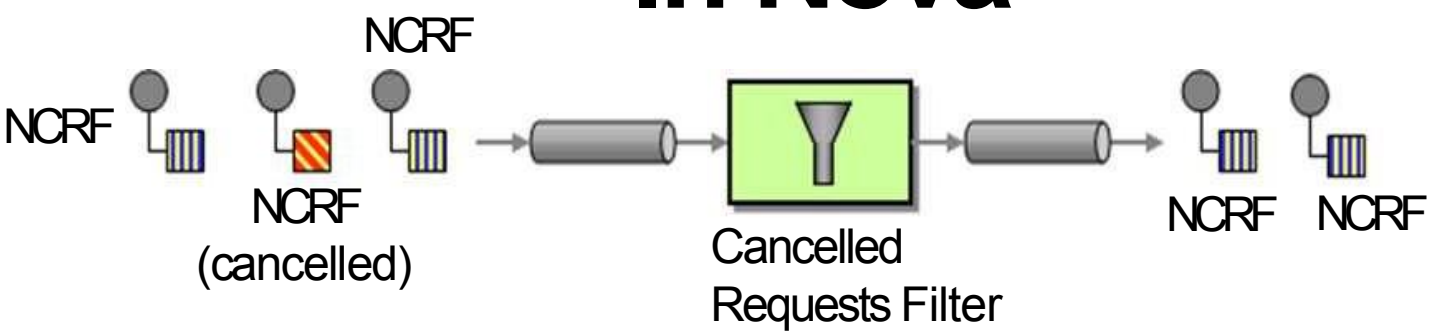
In Nova - Message Flattening



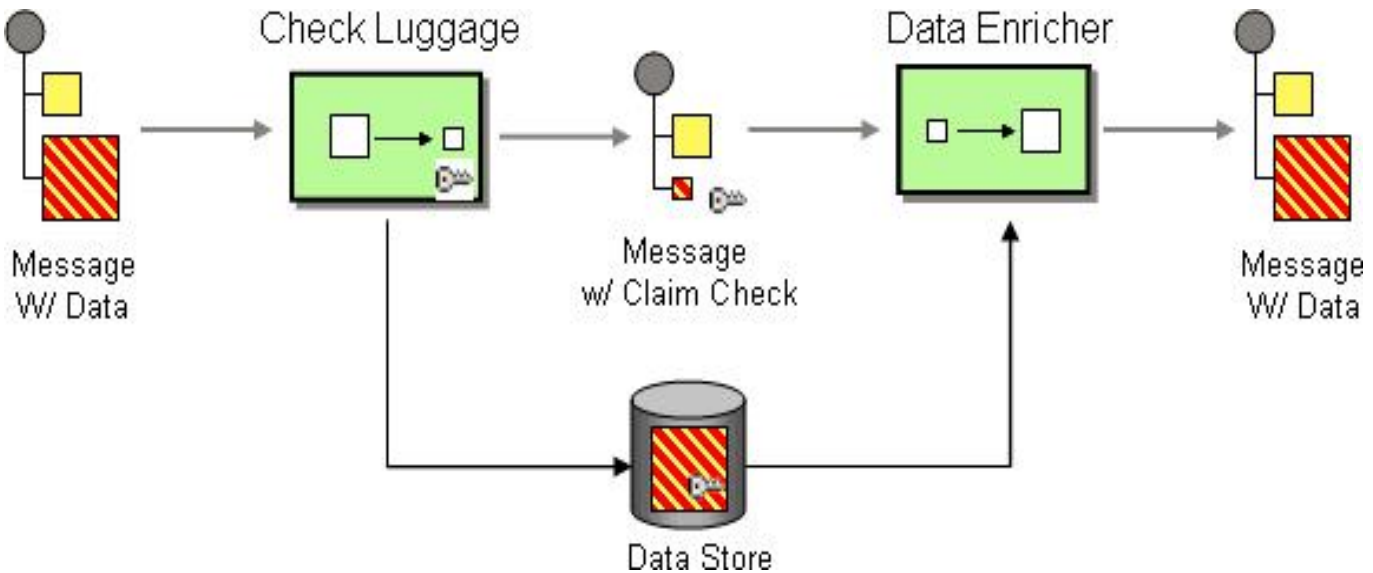
Message Filter



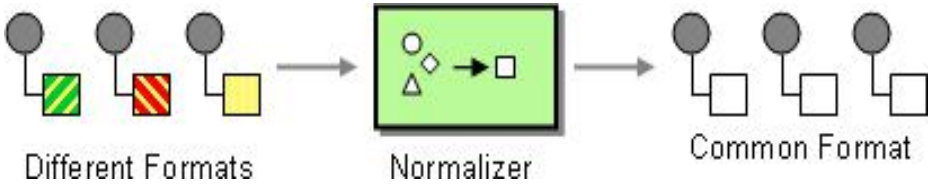
In Nova



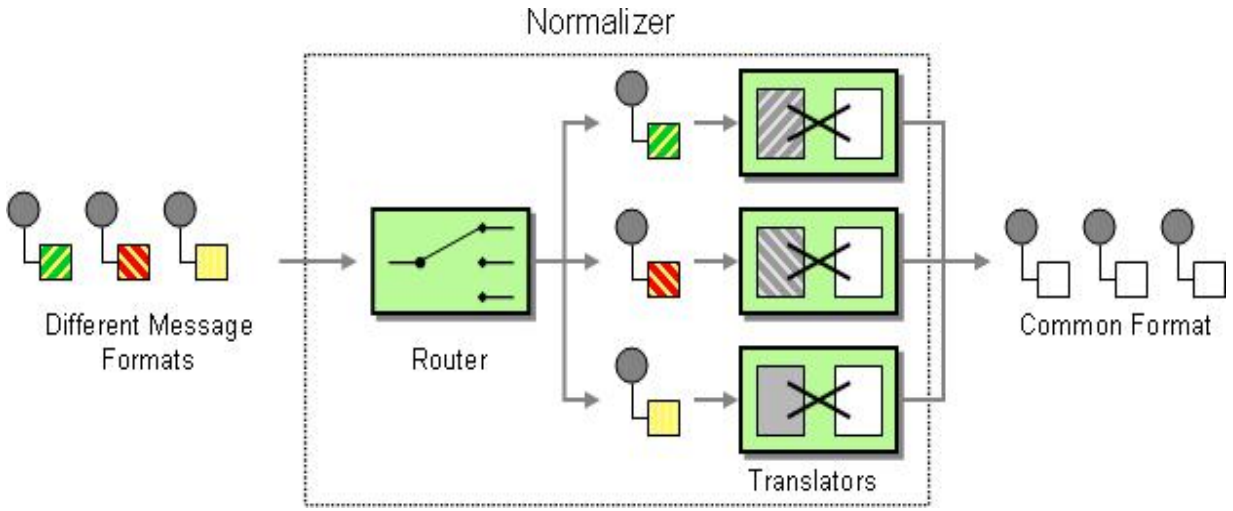
Claim Check



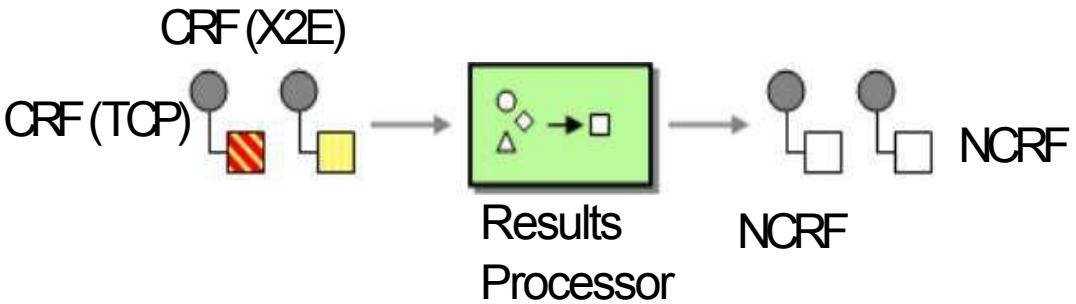
use the Claim Check to hide the sensitive portions of data

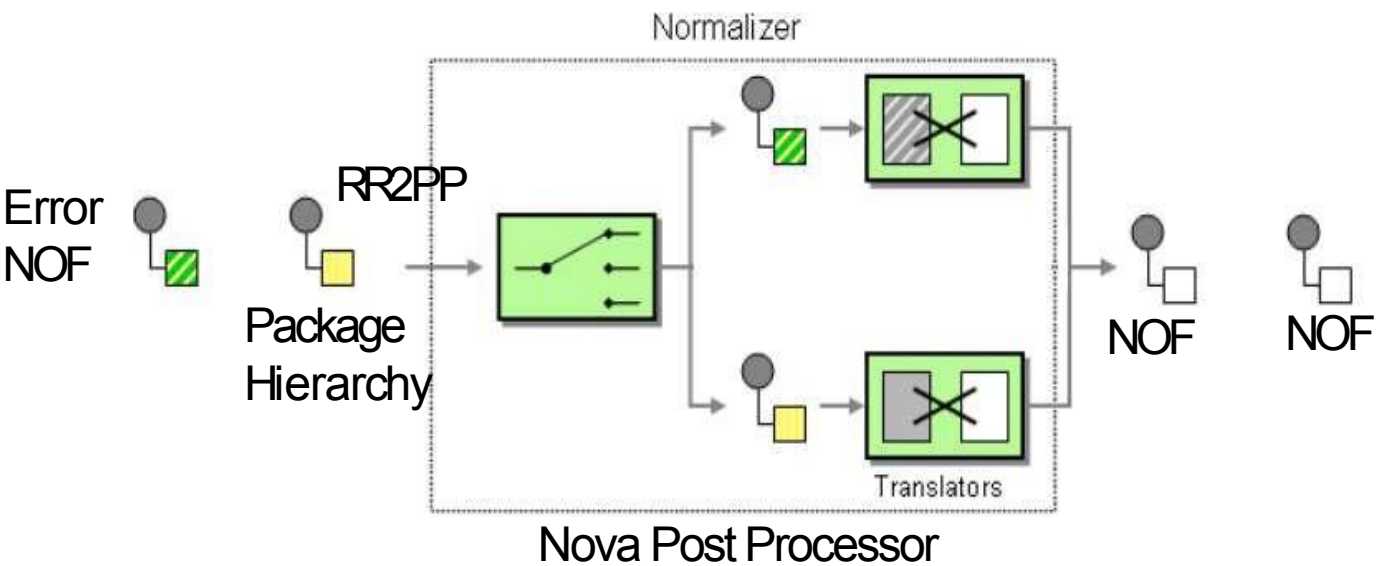

Different Formats
Normalizer
Common Format

Normalizer

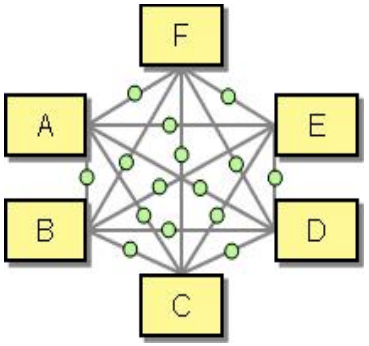

Normalizer
Different Message Formats
Router
Translators
Common Format

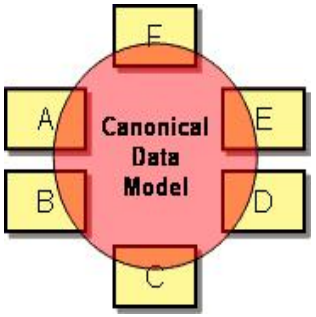
In Nova

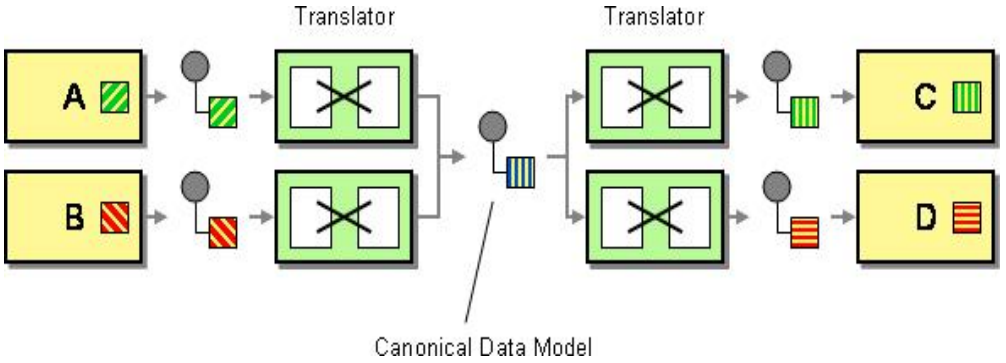

CRF (X2E)
CRF (TCP)
Results
Processor
NCRF
NCRF


Normalizer
Error
NOF
RR2PP
Package
Hierarchy
Translators
NOF
NOF
Nova Post Processor

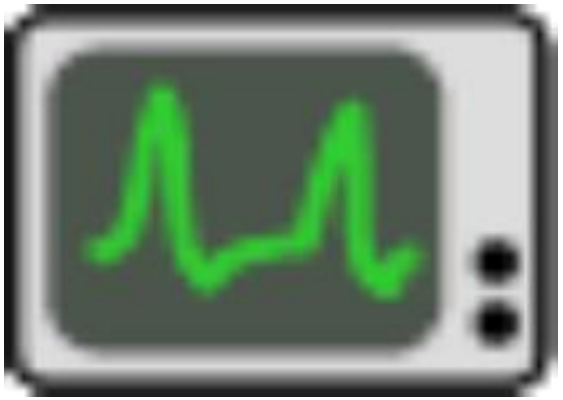
Canonical Data Model





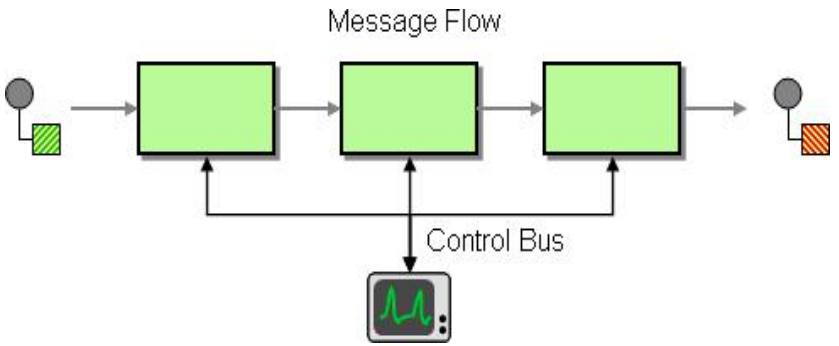


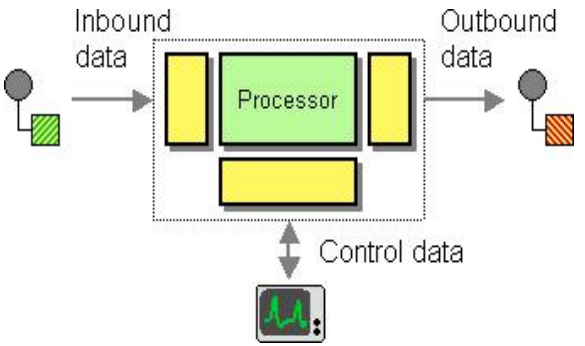
System Management Patterns



https://kodtodya.github.io/talks/

Control Bus





- Configuration
- Heartbeat
- Test Messages
- Exceptions
- Statistics
- Live Console

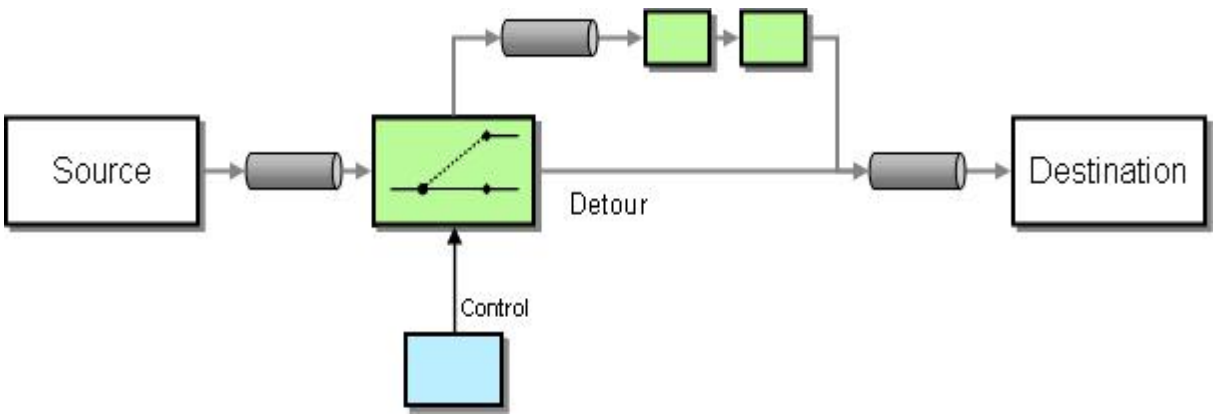
In Nova:

- Request Router – configuration
- Progress Monitor – Statuses

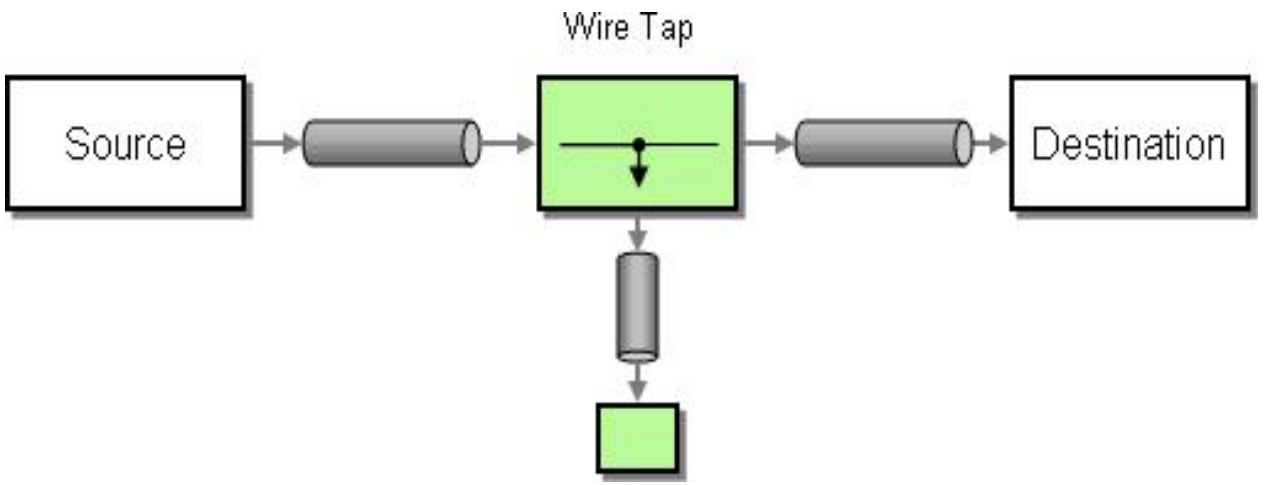
Detour

Purposes:

- **Debugging**
- **Validation**
- **Testing**
- **Inspection**

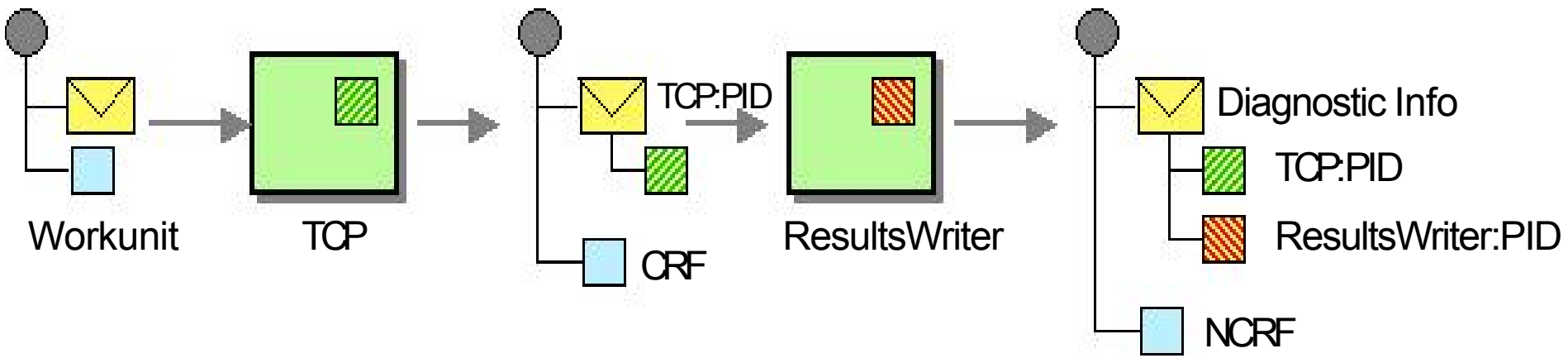


WireTap

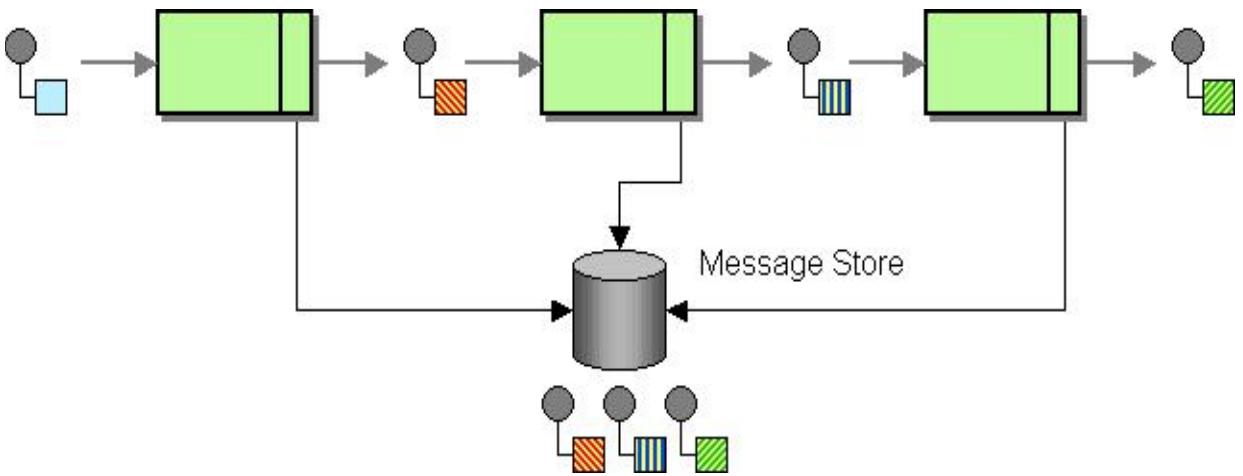


In Nova – logging messages to disk

Message History



Message Store

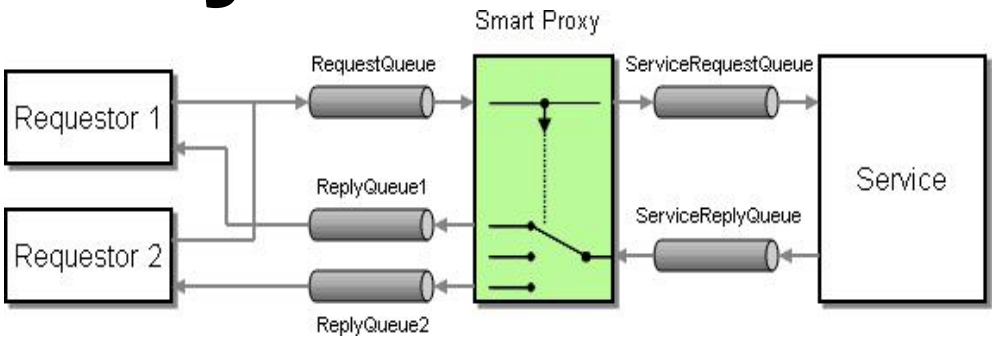


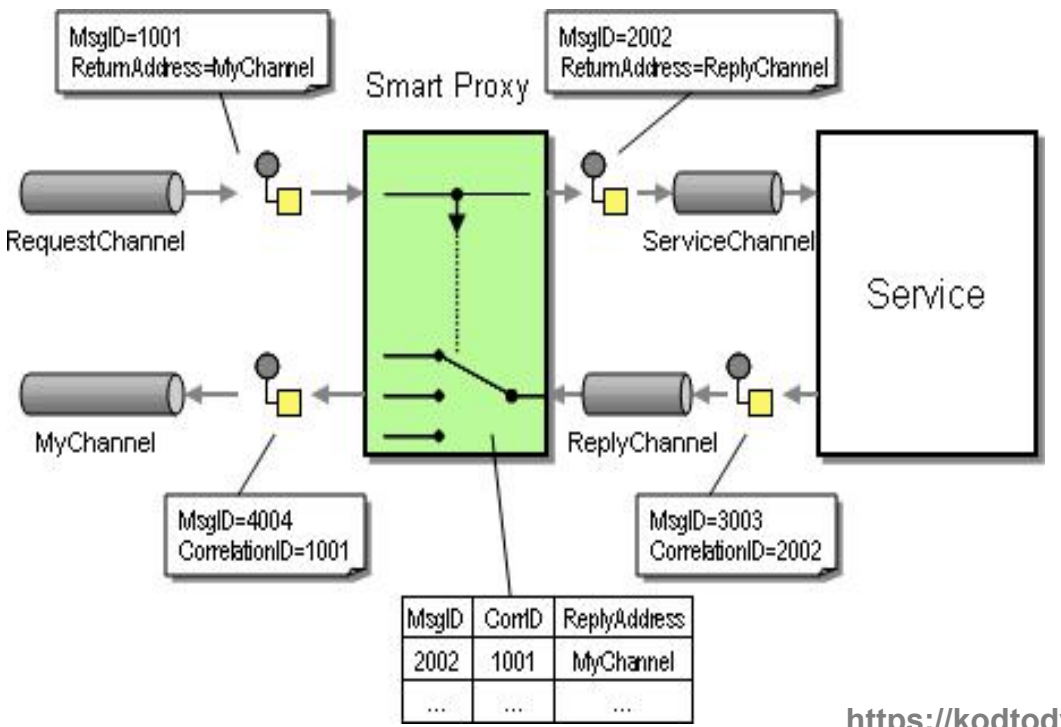
1) Xvis, Instrumentation DB
2) Message logging to disk in Nova

Smart Proxy

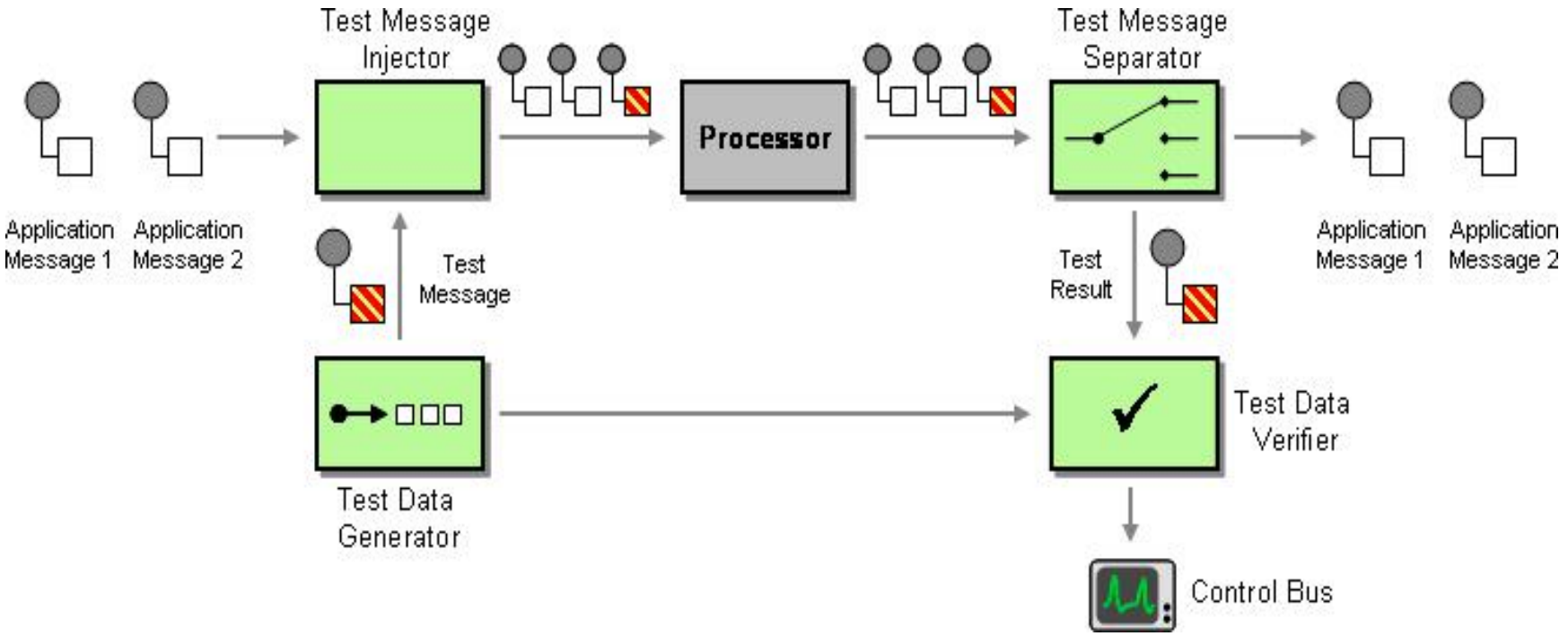
The *Smart Proxy* can store this data in two places:

- Inside the Message
- Inside the Smart Proxy



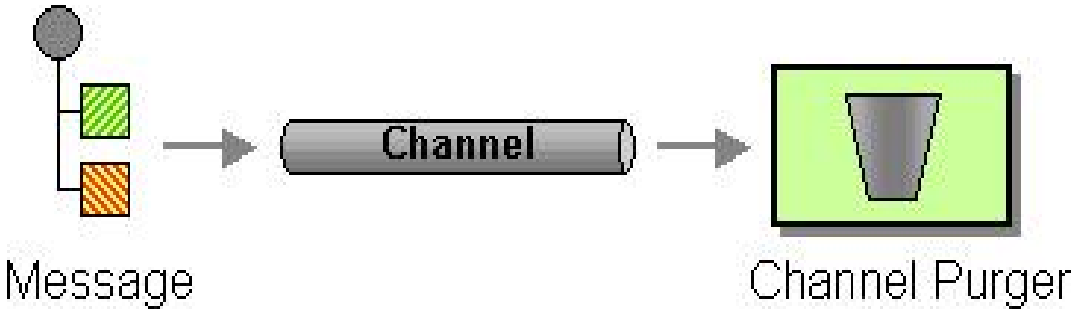


Test Message



https://kodtodya.github.io/talks/

Message Purger



Enterprise Message Brokers

IBM MQ

Tibco EMS

Frameworks that implement EIP

https://kodtodya.github.io/talks/

OSGi

OSGi to be covered in separate presentation...

Questions ?

Thank you..!!!

LinkedIn, GitHub, GitLab, Twitter: @kodtodya

https://kodtodya.github.io/talks/