Messaging Systems

Systems Integration

PBA Softwareudvikling/BSc Software Development

Tine Marbjerg

Fall 2018

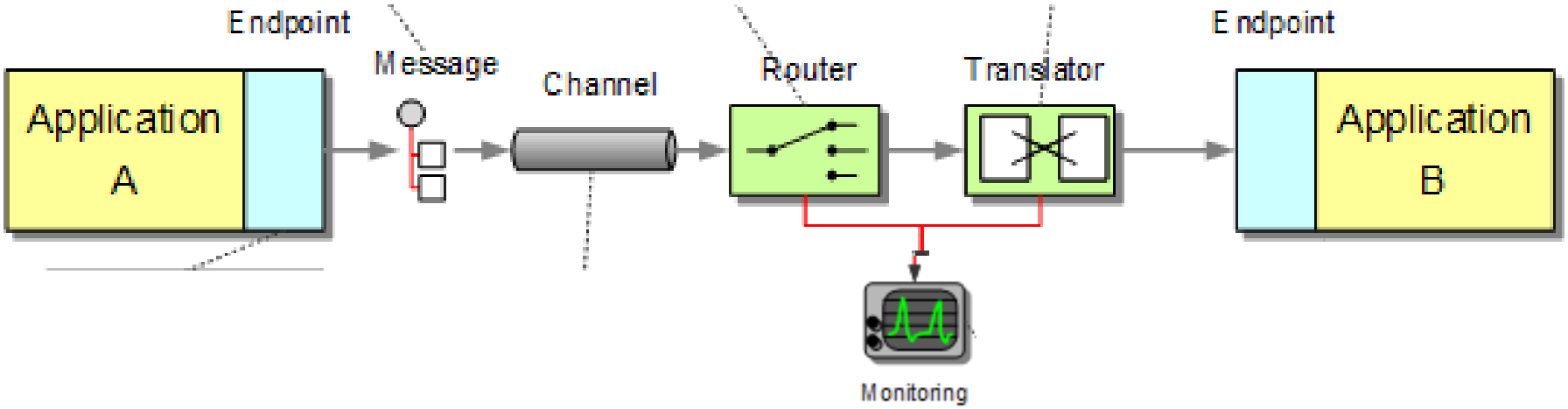
Basic Messaging Concepts (EIP chapter 3)

- Basic Concepts
  - Channel (60)
  - Message (66)
  - Pipes and Filters (70)
  - Message Router (78)
  - Message Translator (85)
  - Message Endpoint (95)

- MessageKit exercises
  - Made by EIP author Gregor Hohpe
  - To give you conceptual overview of messaging

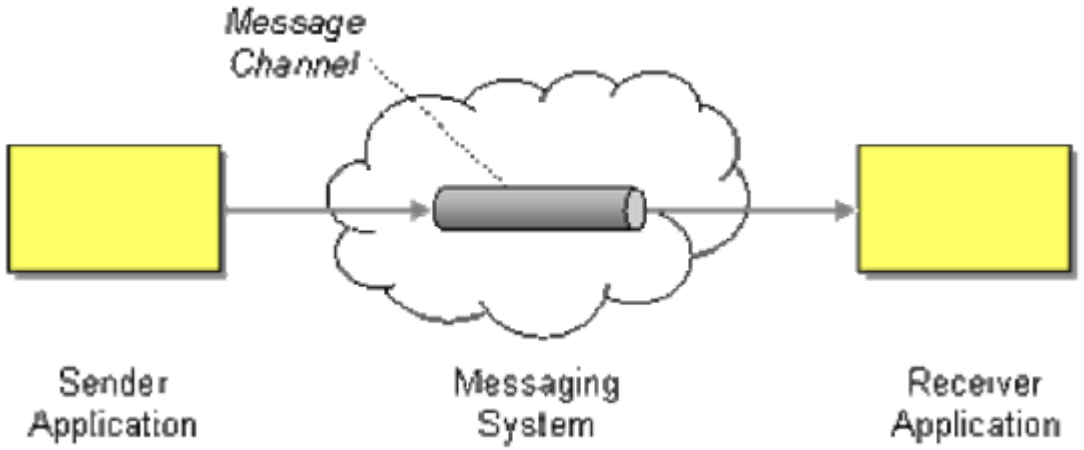
Basic Messaging Architecture (EIP chap. 3)

- Basic messaging concepts



Message Channel (60)

- ***How does one application communicate with another using messaging?***
- Connect the applications using a *Message Channel,* where one application writes information to the channel and the other one reads that information from the channel

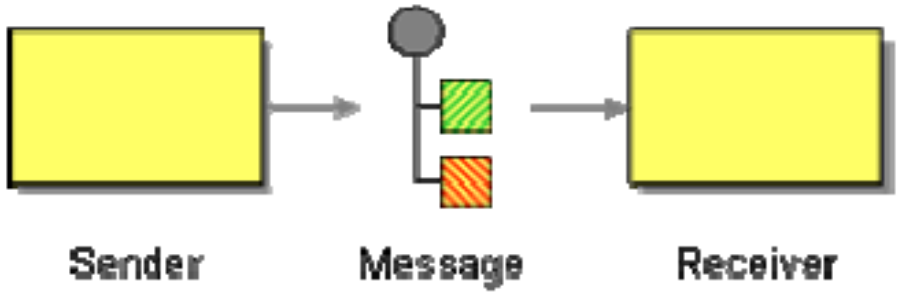


Message Channel Design

- Channels are logical addresses in the messaging system,
  - E.g. MSMQ path name syntax: **`ComputerName\QueueName`**
- Developers have to decide what channels they need for communication
- Number and purpose of channels tend to be fixed at deployment time
- A well-designed set of channels form a messaging API for a whole group of applications
- Pretty much like database design ☺

Message (66)

- ***How can two applications connected by a message channel exchange a piece of information?***
- Package the information into a *Message,* a data record that the messaging system can transmit through a message channel



Message – Basic Parts

Header

Information used by the messaging system that describes the data being transmitted; its origin; its destination, lifetime, priority etc.

Body

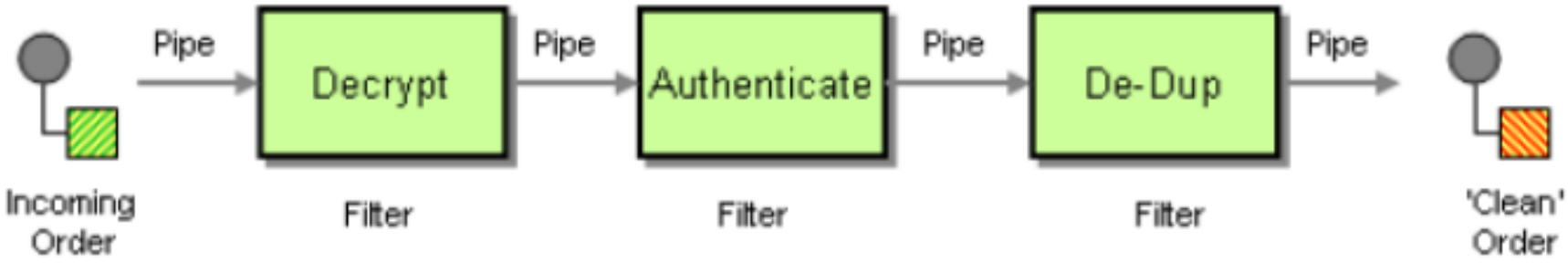
The data being transmitted

```
public void Send()
{
        Message requestMessage = new Message();
        requestMessage.Body = "Hello world.";
```

Generally ignored by the messaging system and simply transmitted as-is

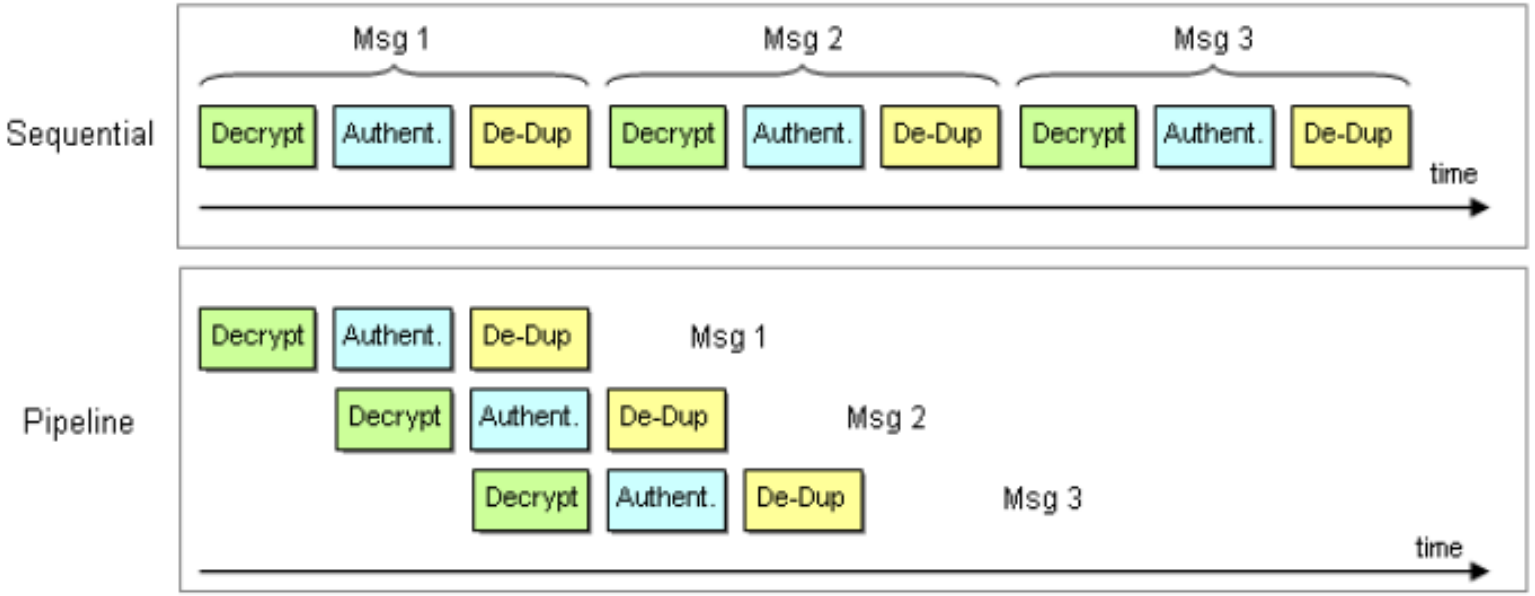
Pipes and Filters (70)

- ***How can we perform complex processing on a message while maintaining independence and flexibility?***
- Use the *Pipes and* Filters architectural style to divide a larger processing task into a sequence of smaller, independent processing steps (filters) that are connected by channels (pipes)



Pipeline Processing

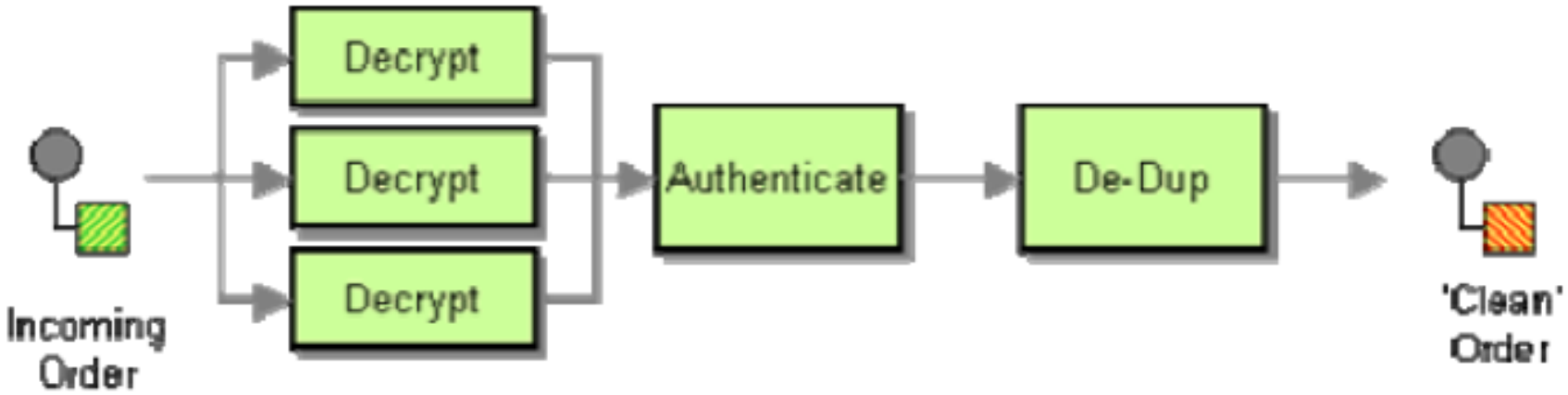
- *Pipes and Filters* allows multiple messages to be processed concurrently
- Can increase the overall system *throughput* (but not the throughput for each message)



Parallel Processing

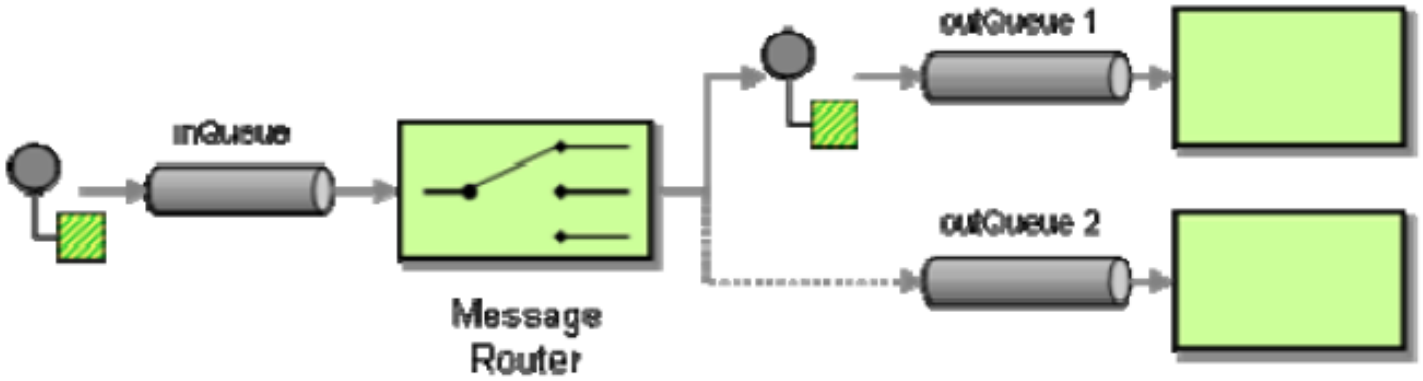
- The overall system throughput is limited by the slowest process in the chain.

- We can increase throughput with parallel processing:



Message Router (78)

- ***How can you decouple individual processing steps so that messages can be passed to different filters depending on a set of conditions?***
- Insert a special filter, a *Message Router,* which consumes a *Message* from one *Message Channel* and republishes it to a different *Message Channel,* depending on a set of conditions

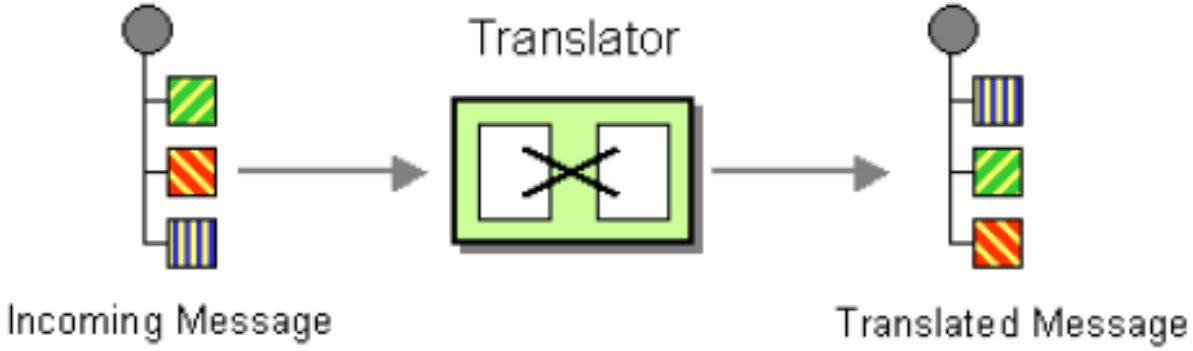


Simple Router with C# and MSMQ – (Example EIP p. 83)

```
class SimpleRouter
{
    protected MessageQueue inQueue;
    protected MessageQueue outQueue1;
    protected MessageQueue outQueue2;

    public SimpleRouter(MessageQueue inQueue, MessageQueue outQueue1, MessageQueue outQueue2)
    {
        this.inQueue = inQueue;
        this.outQueue1 = outQueue1;
        this.outQueue2 = outQueue2;

        inQueue.ReceiveCompleted += new ReceiveCompletedEventHandler(OnMessage);
        inQueue.BeginReceive();
    }

    private void OnMessage(Object source, ReceiveCompletedEventArgs asyncResult)
    {
        MessageQueue mq = (MessageQueue)source;
        Message message = mq.EndReceive(asyncResult.AsyncResult);

        if (IsConditionFulfilled())
            outQueue1.Send(message);
        else
            outQueue2.Send(message);

        mq.BeginReceive();
    }
```

Message Translator (85)

- ***How can systems using different data formats communicate with each other using messaging?***
- Use a special filter, a *Message Translator,* to translate one data format into another

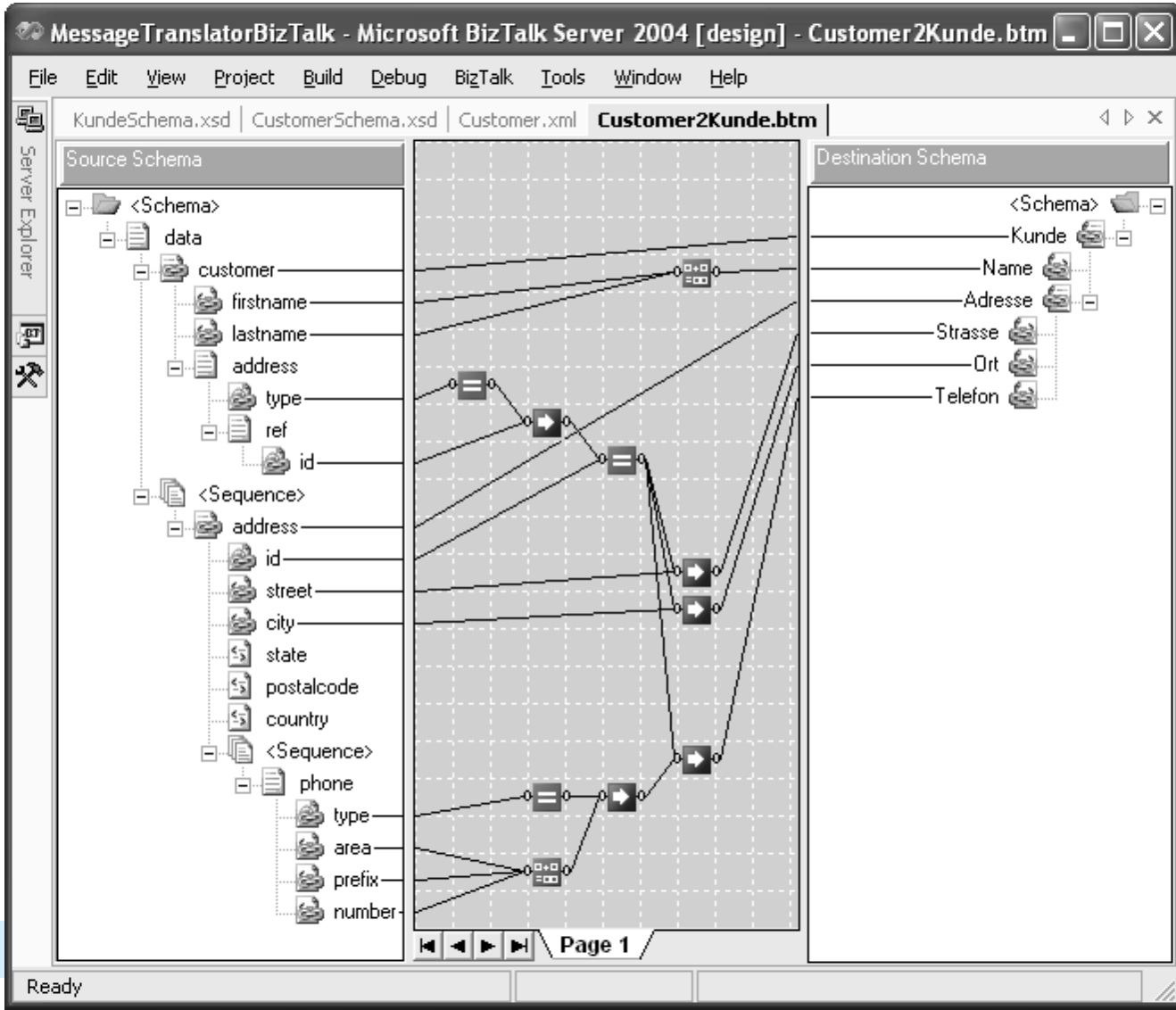


Levels of Transformation

Layer	Deals With	Transformation Needs (Example)	Tools/ Techniques
Data Structures (Application Layer)	Entities, associations, cardinality	Condense many-to-many relationship into aggregation.	Structural mapping patterns, custom code
Data Types	Field names, data types, value domains, constraints, code values	Convert ZIP code from numeric to string. Concatenate First Name and Last Name fields to single Name field. Replace U.S. state name with two-character code.	EAI visual transformation editors, XSL, database lookups, custom code
Data Representation	Data formats (XML, name-value pairs, fixed-length data fields, EAI vendor formats, etc.) Character sets (ASCII, UniCode, EBCDIC) Encryption/compression	Parse data representation and render in a different format. Decrypt/encrypt as necessary.	XML parsers, EAI parser/ renderer tools, custom APIs
Transport	Communications protocols: TCP/IP sockets, HTTP, SOAP, JMS, TIBCO RendezVous	Move data across protocols without affecting message content.	*Channel Adapter* (127), EAI adapters

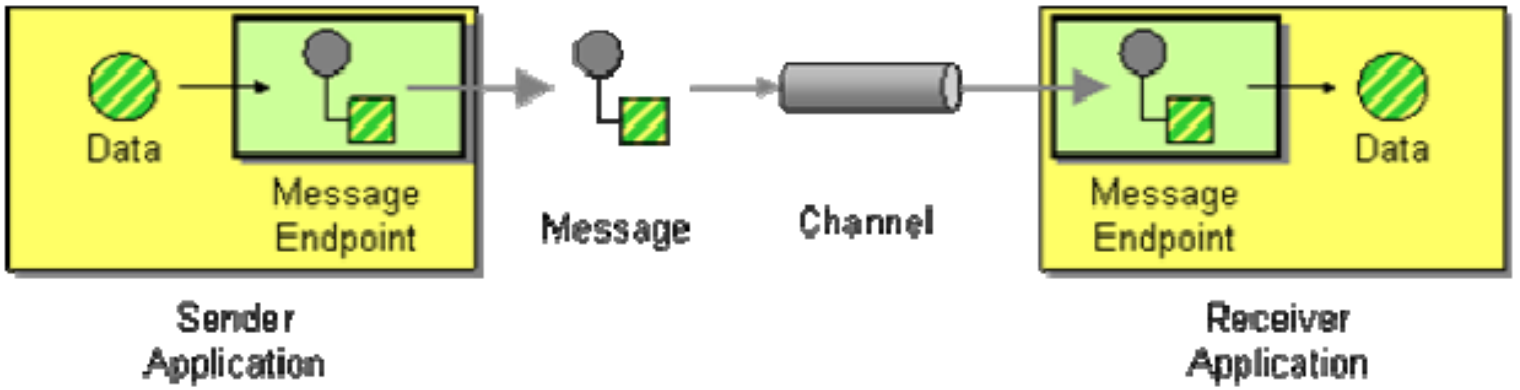
Visual transformation of data types – ex.

- Drag-Drop Style



Message Endpoint (95)

- ***How does an application connect to a messaging channel to send and receive messages?***
- Connect an application to a messaging channel using a *Message Endpoint,* a client of the messaging system that the application can then use to send or receive messages.



Exercise Time

- See in exercise folder: *"Intro Exercise"*