Messaging Coffeeshop Exercises Follow Up

System Integration

PBA Softwareudvikling/BSc Software Development

Tine Marbjerg

Fall 2017

Today's Agenda

- Last week's Coffee Shop exercises
  - What did you learn?
- Integration Styles (EIP chap. 2) & Messaging Channels patterns (EIP chap. 4)
  - Moodle Multiple Choice Quiz
- Message Construction patterns (EIP chap. 5)
  - Programming exercises with MSMQ (in .NET)

Objectives for Coffee Shop Exercises

- Demonstrate the role of messaging in decoupling of applications so that they can be more **scalable**
- Illustrate some of the **challenges** that need to be addressed when adopting a decoupled architecture
- The role of messaging in addressing **non-functional requirements**
- Introduction to **pattern language** from EIP book

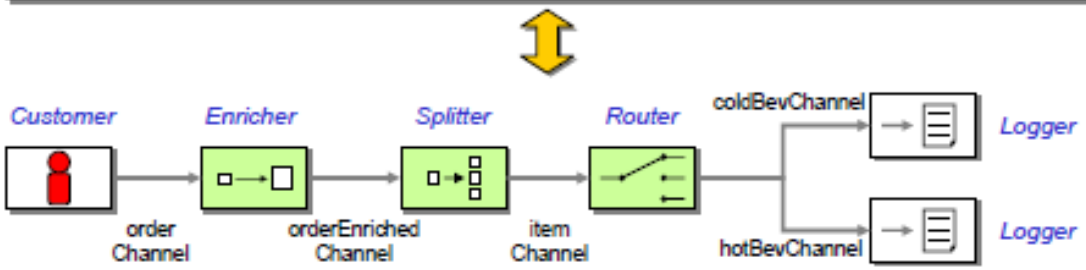
Coffee Shop Domain Specific Language

- Composition of solutions from predefined components (.bat files)
- Domain Specific Language listed in Tutorial Reference Chart

Language in use!

```
call Customer orderChannel
call Enricher orderChannel orderEnrichedChannel
call Splitter orderEnrichedChannel itemChannel "/Order/Item"
call Router itemChannel coldBevChannel "Item = 'FRAPPUCINO'" hotBevChannel
call Logger coldBevChannel
call Logger hotBevChannel
```

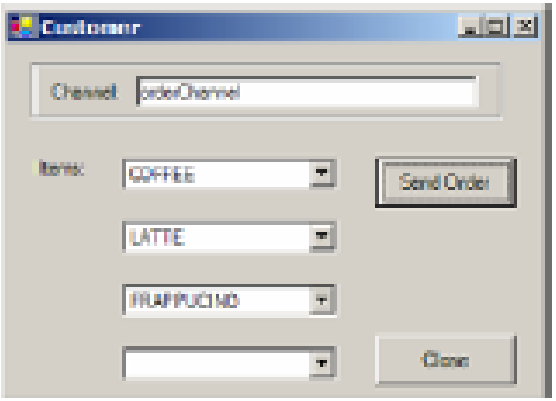
Graphic illustration of flow!



Convenience and Test Components

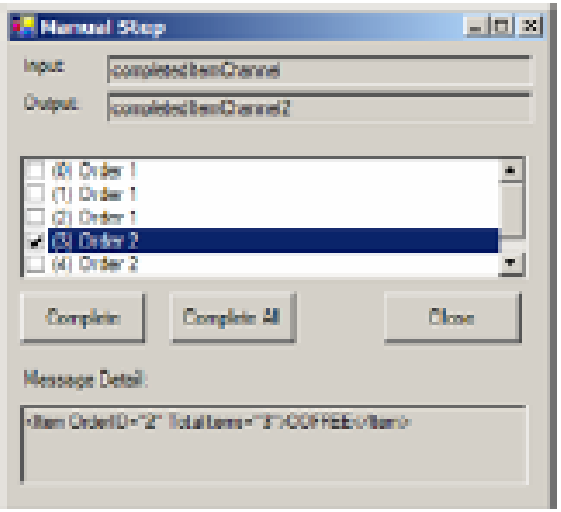
Examples:

Customer



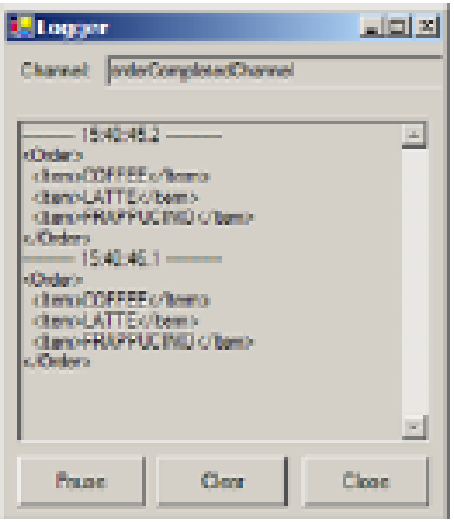
Sends order messages to specified channel

Manual Step



Allows inspection of messages and out-of-sequence completion

Logger



Display messages and time stamps

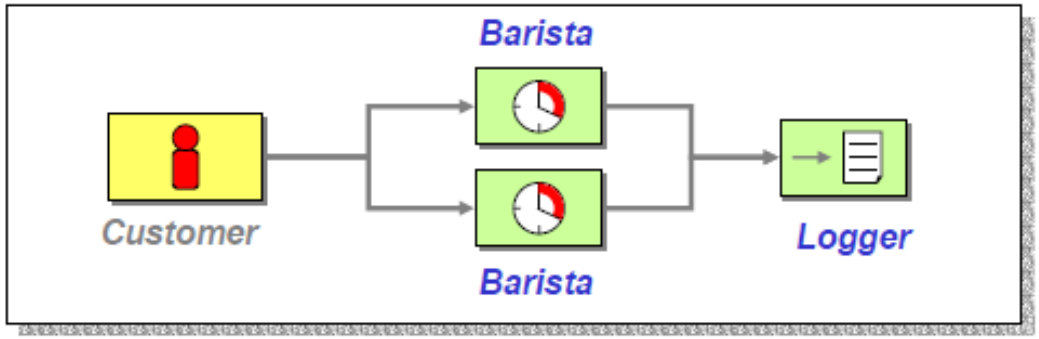
Coffee Shop Follow-up – Exercise 1

Problem: Scalability

Higher throughput with 2 baristas

1 barista: 1 coffee per second

2 baristas: 2 coffees per second



Observation

- Overall throughput is doubled
- Messaging architecture scales through ***Competing Consumers***
- Scalability: Adding more baristas <u>did not require changes</u> to the architecture or existing components

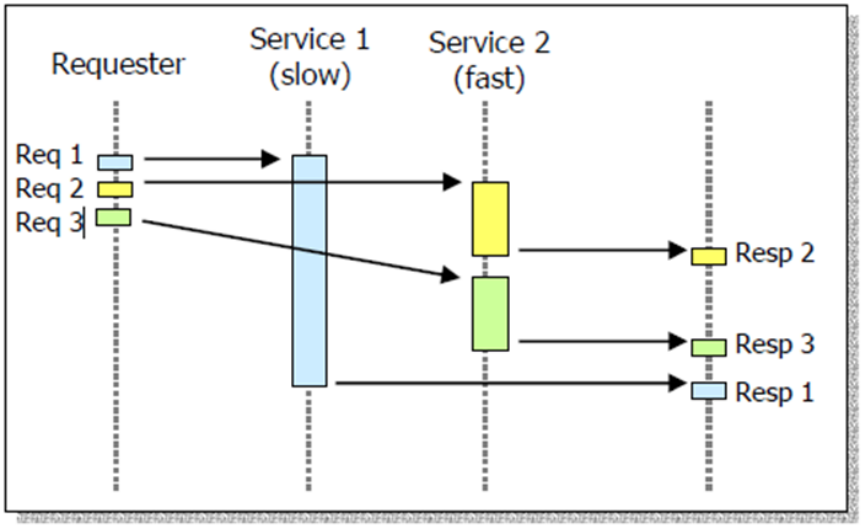
Example Solution for Exercise 1B

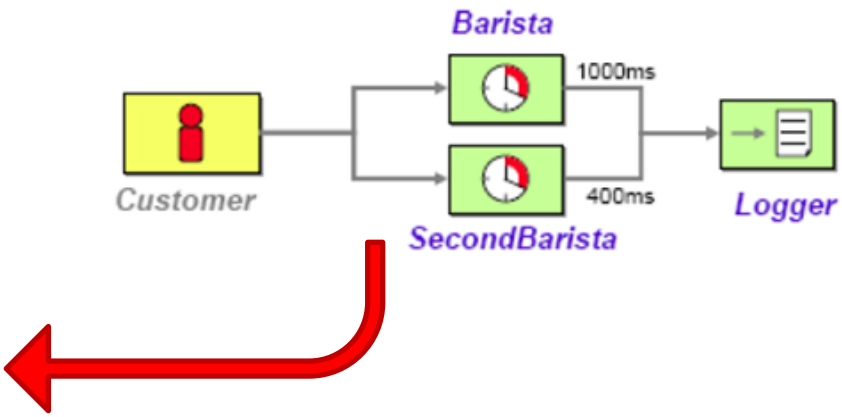
```
call Customer orderChannel

call Barista orderChannel orderCompletedChannel

call Barista orderChannel orderCompletedChannel

call Logger orderCompletedChannel
```

Coffee Shop Follow-up – Exercise 2

Problem: sequencing

Proper sequence (some components are faster than others)



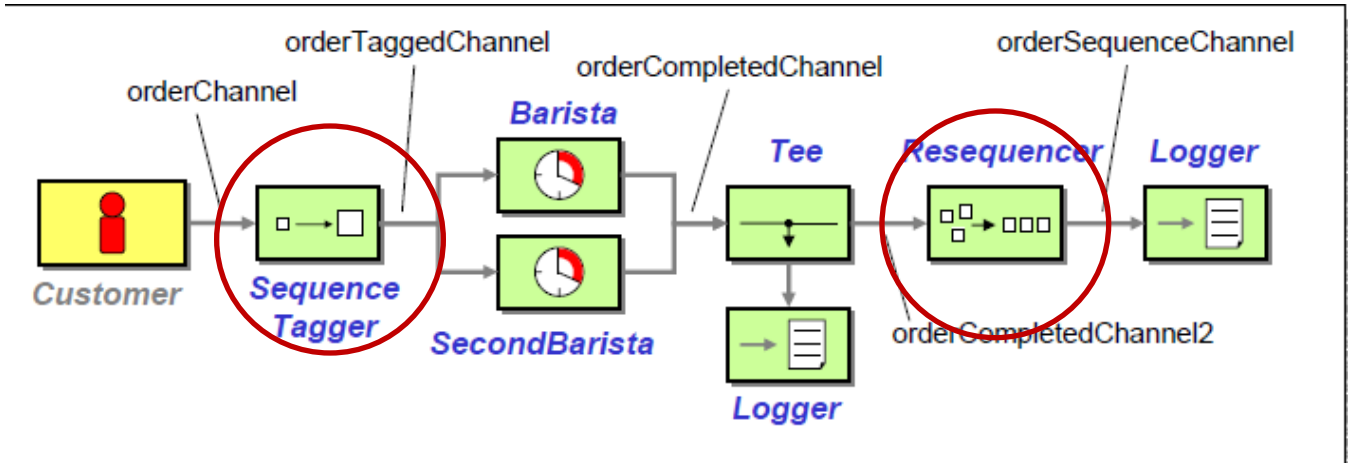


Observation

- Parallel processing may cause messages to get out of order
  - We need to give each message unique identity
  - We need to collect and re-order messages so that they can be published to an output channel in a specified order

Coffee Shop Follow-up – Exercise 2

Possible solution to sequencing problem:



- SequenceTagger (i.e. ***Content Enricher***) adds consecutive numbers to messages
- ***Resequencer*** brings messages back in order

Example Solution for Exercise 2

```
call Customer orderChannel
call SequenceTagger orderChannel orderTaggedChannel "/Order/@OrderID"
call Barista orderTaggedChannel orderCompletedChannel
call SecondBarista orderTaggedChannel orderCompletedChannel
call Tee orderCompletedChannel orderCompletedChannel2 logChannel
call Logger logChannel
call ManualStep orderCompletedChannel2 orderCompletedChannel3 "/Order/@OrderID"
call Resequencer orderCompletedChannel3 orderSequenceChannel "/Order/@OrderID"
call Logger orderSequenceChannel
```

Exercise 2 - Discussion

- Resequencing: Possible complications related to throughput, latency and robustness?

Brief discussion

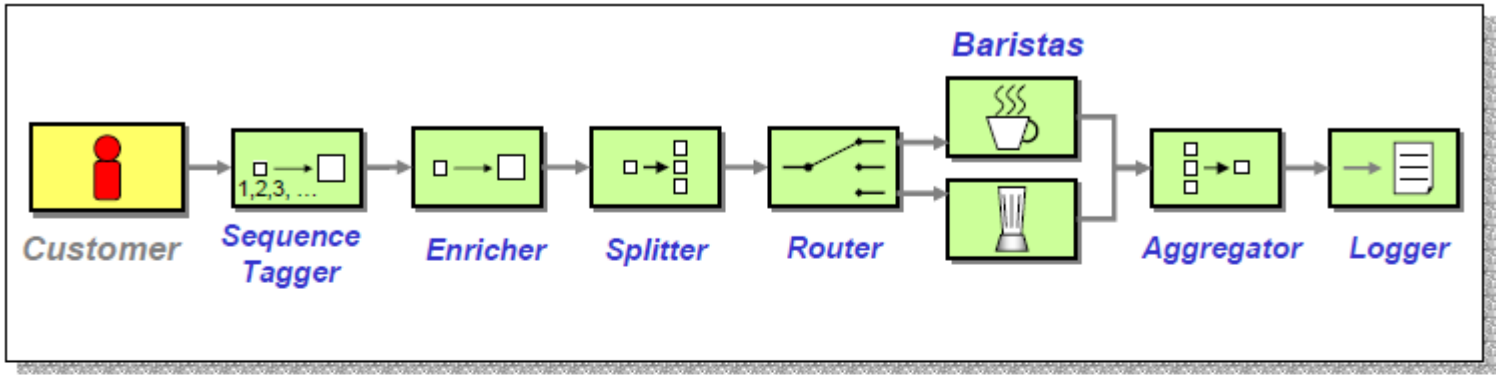
Coffee Shop Follow-up – Exercise 3

Problem

- Processing a whole order at one time limits our scaling options
- Creating a specialized Barista each for iced beverages and for hot beverages allows us to fine-tune baristas

Coffee Shop Follow-up – Exercise 3

Possible solution for exercise 3:




```
call Customer orderChannel
call SequenceTagger orderChannel orderTaggedChannel "/Order/@OrderID"
call Enricher orderTaggedChannel orderEnrichedChannel
call Tee orderEnrichedChannel orderEnrichedChannel2 logEnrichedChannel
call Logger logEnrichedChannel
call Splitter orderEnrichedChannel2 orderItemChannel "/Order/Item"
call Tee orderItemChannel orderItemChannel2 logItemChannel
call Logger logItemChannel
call Router orderItemChannel2 orderItemColdChannel "Item = 'FRAPPUCINO'" orderItemHotChannel
call ColdBevBarista orderItemColdChannel orderItemCompletedChannel
call HotBevBarista orderItemHotChannel orderItemCompletedChannel
call Aggregator orderItemCompletedChannel orderCompletedChannel
call Logger orderCompletedChannel
```

Coffee Shop Follow-up – Exercise 3

Observations

- Splitting allows different message types to be processed individually
- Separating tasks into smaller pieces can improve throughput for the application and support greater scalability
- Messages will get out of order and need to be re-aggregated
  - ***Aggregator*** combines individual, but related messages so they can be processed as a whole