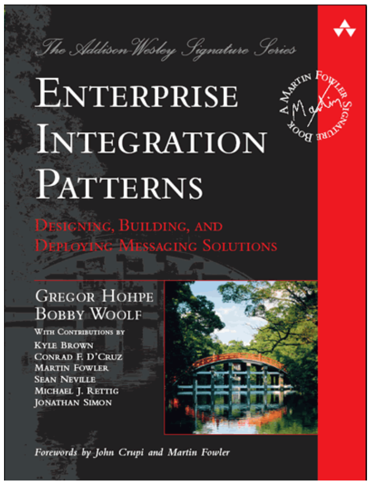
MESSAGING PATTERNS

52

52

Enterprise Integration Patterns

- A pattern language on messaging
- 65 patterns
- Several code examples

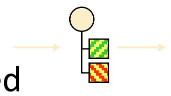

The Addison-Wesley Signature Series
ENTERPRISE INTEGRATION PATTERNS
DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS
A MARTIN FOWLER SIGNATURE BOOK
GREGOR HOHPE
BOBBY WOOLF
WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON
Forewords by John Crupi and Martin Fowler

53

53

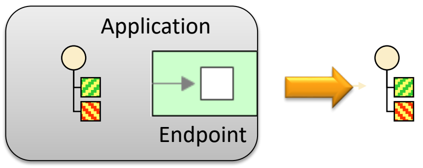
Basic Patterns

- Message
  - Self-contained



- Channel

  - Location-independent, separate from applications
  - Asynchronous and reliable
- Message Endpoint
- Concepts
  - Fire-and-forget
  - Store-and-forward

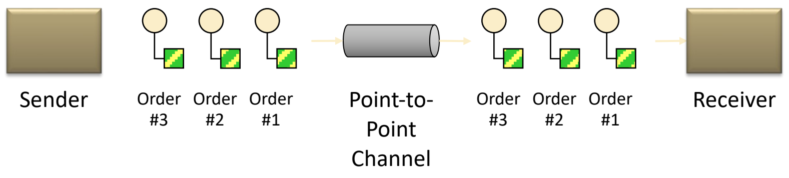


54

54

Point-to-Point Channel

- Make sure only one receiver will consume each message
- Send the message on a Point-to-Point Channel
  - Channel ensures only one receiver consumes a message

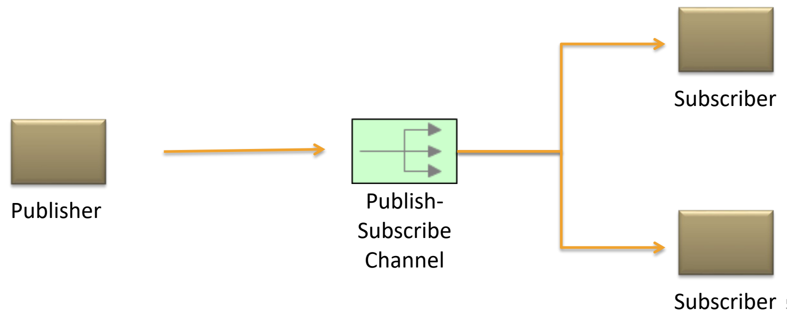


55

55

Publish-Subscribe Channel

- How can the sender broadcast an event to all interested receivers?
- Send event on a Publish-Subscribe Channel
  - Channel delivers copy to each subscriber

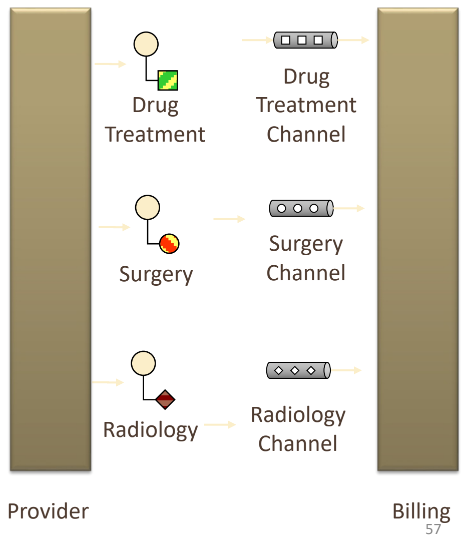


56

56

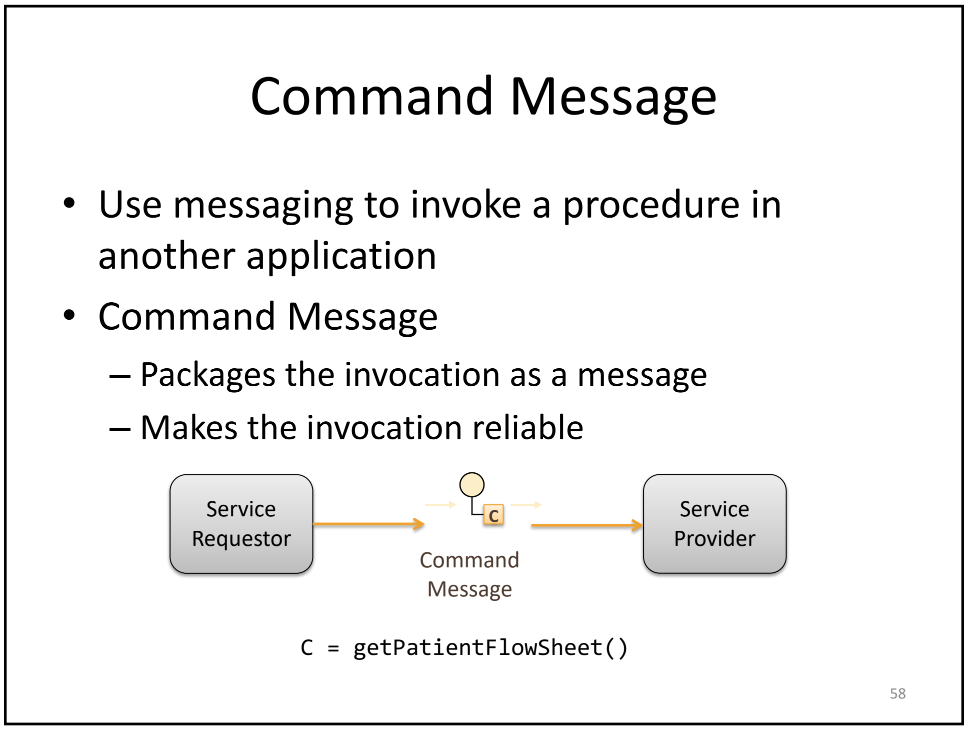
Datatype Channel

- Send a data item so the receiver knows how to process it
- Use a separate Datatype Channel for each data type
  - Data on a channel is all one type



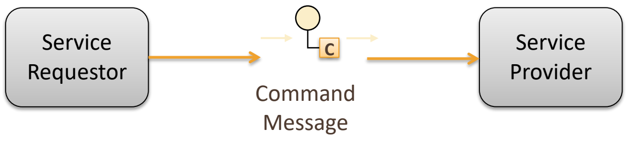
57

57



Command Message

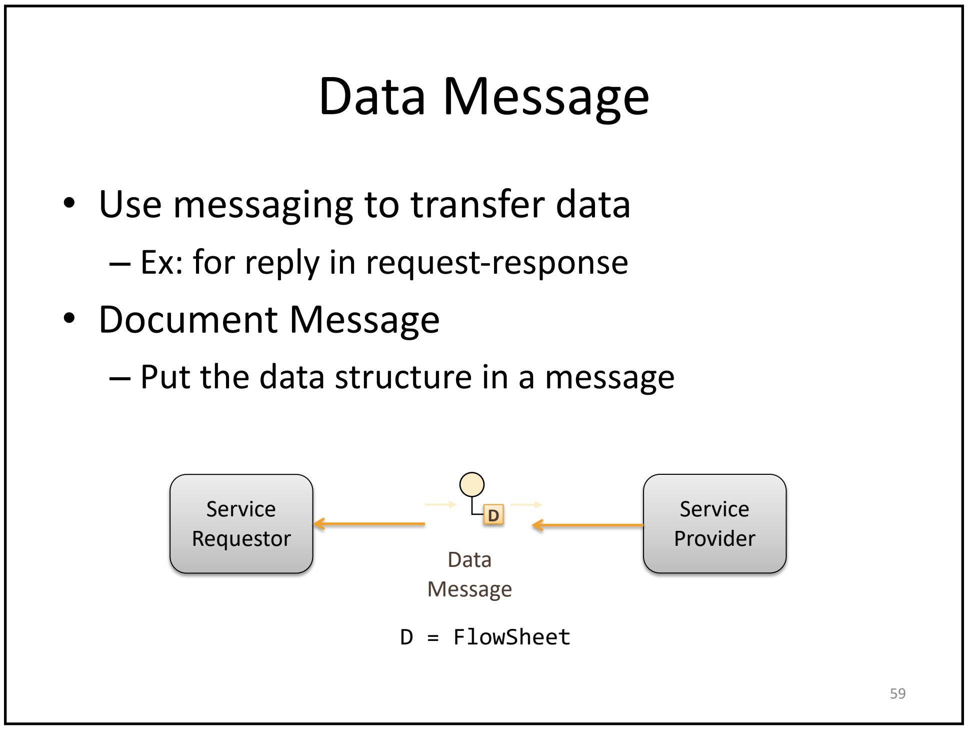
- Use messaging to invoke a procedure in another application
- Command Message
  - Packages the invocation as a message
  - Makes the invocation reliable



```
C = getPatientFlowSheet()
```

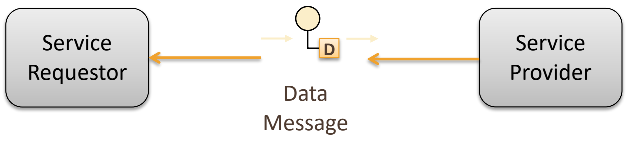
58

58



Data Message

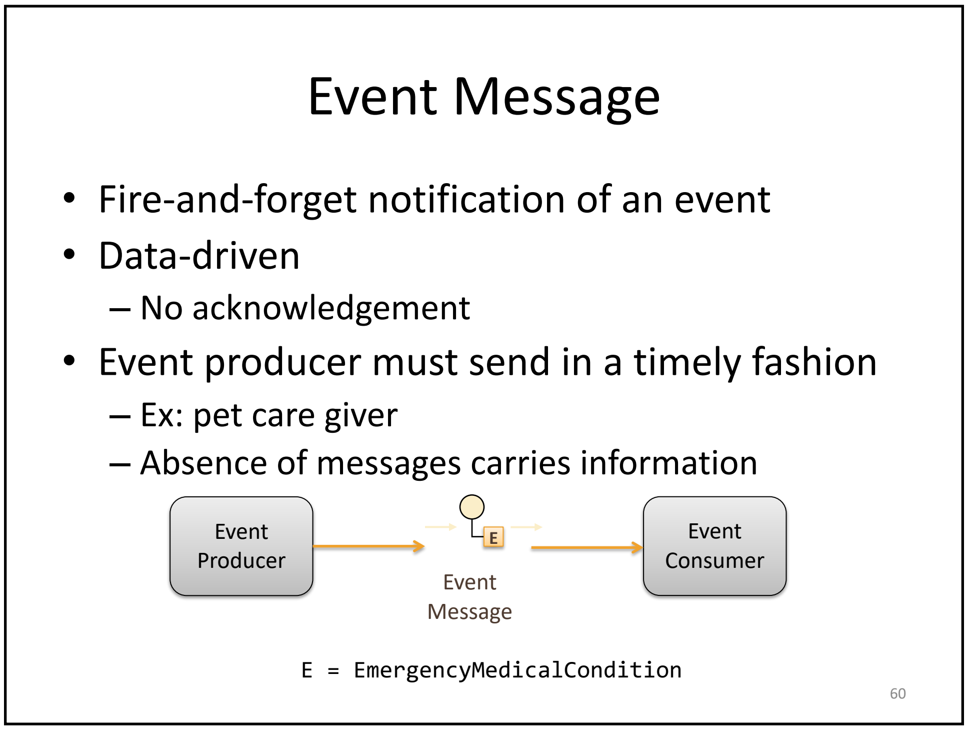
- Use messaging to transfer data
  - Ex: for reply in request-response
- Document Message
  - Put the data structure in a message



```
D = FlowSheet
```

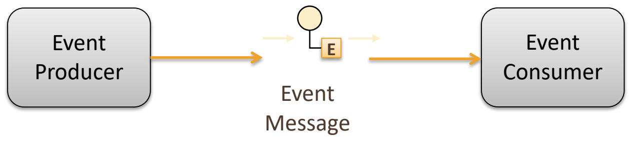
59

59



Event Message

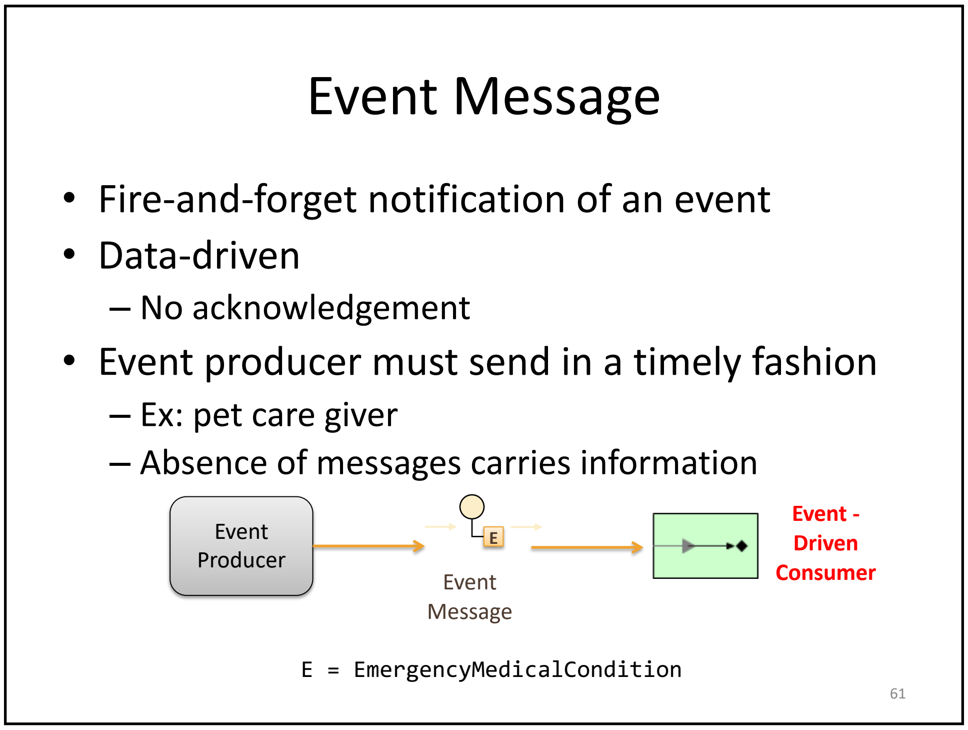
- Fire-and-forget notification of an event
- Data-driven
  - No acknowledgement
- Event producer must send in a timely fashion
  - Ex: pet care giver
  - Absence of messages carries information



E = EmergencyMedicalCondition

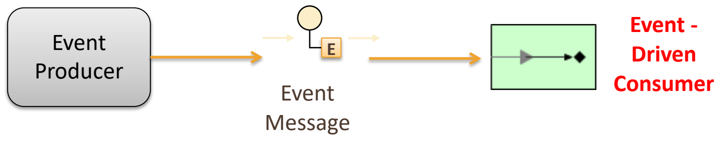
60

60



Event Message

- Fire-and-forget notification of an event
- Data-driven
  - No acknowledgement
- Event producer must send in a timely fashion
  - Ex: pet care giver
  - Absence of messages carries information



E = EmergencyMedicalCondition

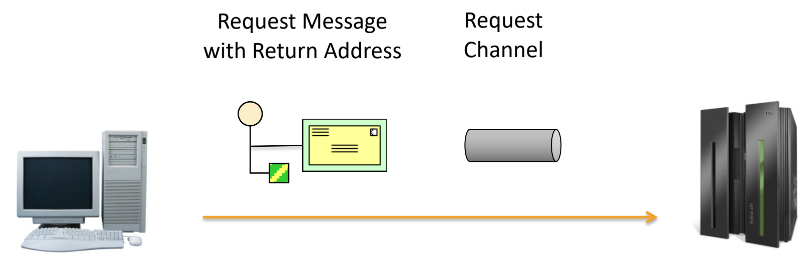
61

61

REQUEST-REPLY PATTERNS

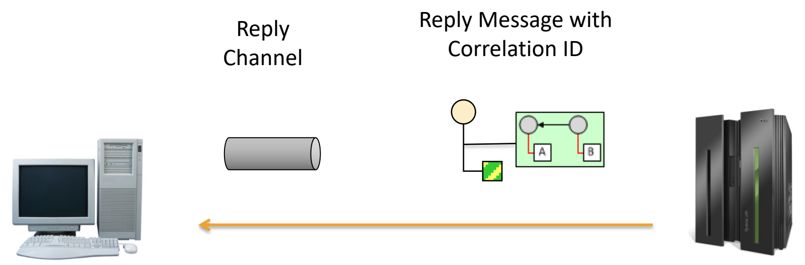
Request-Reply

- Separate request and reply channels
  - Request as command message
  - Return address pattern



Request-Reply

- Separate request and reply channels
  - Response as data message
  - Correlation ID pattern



64

64

Processing Request

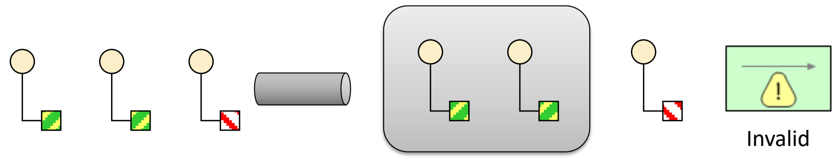
- Request message has several requirements
  - Proper datatype (Datatype Channel pattern)
  - Method to invoke
  - Parameters for method
  - Return address
- What if request message isn’t right?

65

65

Invalid Message Channel

- What if message has wrong format?
- Put message on an Invalid Message Channel
  - Channel for “unprocessable” messages
  - Don’t just dump bad messages back on queue

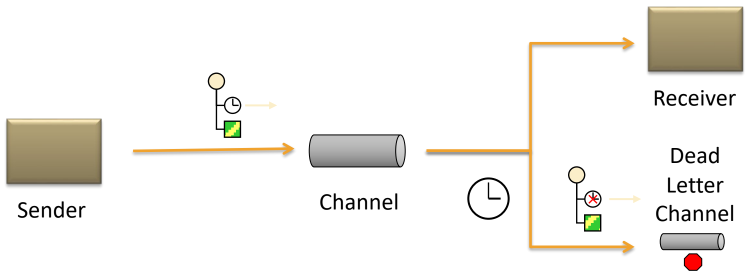


66

66

Dead Letter Channel

- Set expiration period on a message
- Move message to Dead Letter Channel if not delivered by expiration time



67

67

Dead Letter Channel vs Invalid Message Channel

- Dead Letter
  - One for every queue manager
  - Every place where messages may be stored
- Invalid Message
  - Error log
  - Global to the enterprise

68

68

Review of Request-Reply Example

- Use Request-Reply
  - Request is Command Message
  - Reply is Document Message
  - Request has Return Address
  - Reply has Correlation Identifier
  - Malformed requests/replies go to Invalid Message Channel
  - Requests can expire (Message Expiration)
  - Expired messages go to Dead Letter Channel

69

69

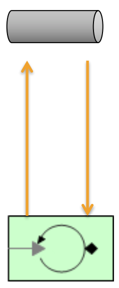
ENDPOINT PATTERNS

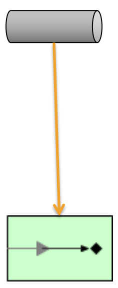
70

70

Consumer Endpoints

- Polling Consumer
- Event-Driven Consumer



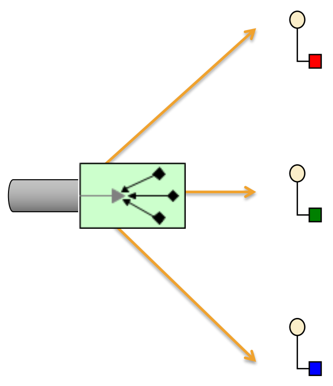


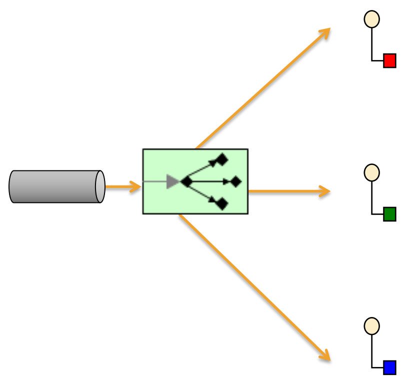
71

71

Consumer Endpoints

- Competing Consumers
- Message Dispatcher



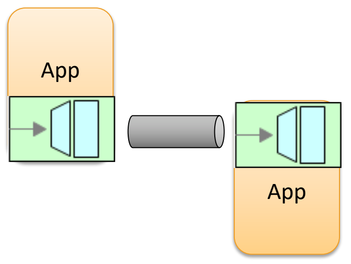


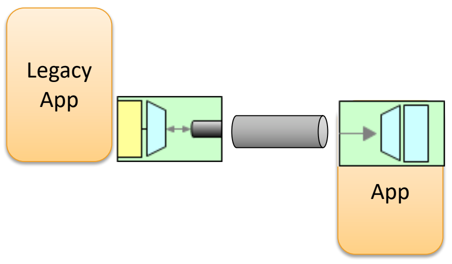
72

72

Consumer Endpoints

- Messaging Gateway
  - Encapsulate queuing logic
- Channel Adapter
  - Legacy wrapper

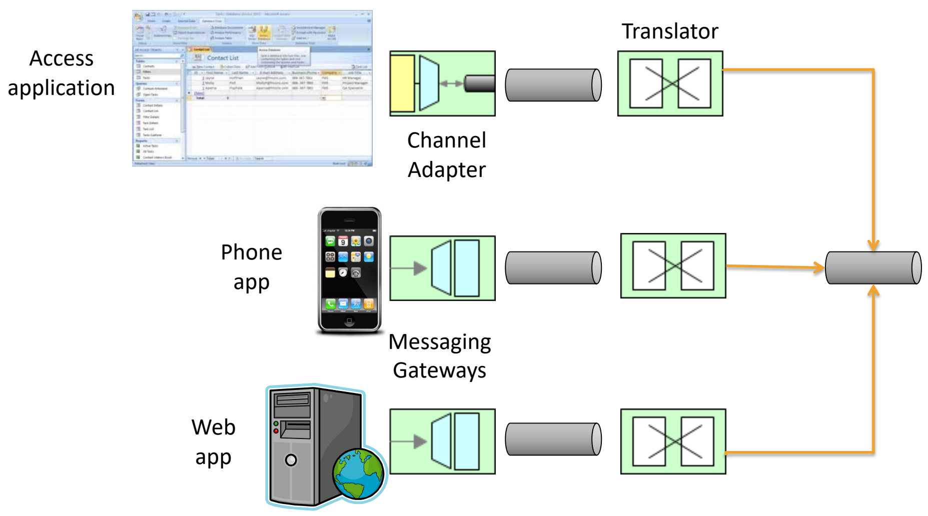




73

73

Multiple Entry Points



74

74

Summary of Entrypoint Patterns

- Polling Consumer
- Event-Driven Consumer
- Competing Consumers
- Message Dispatcher
- Messaging Gateway
- Channel Adapter
- Translater

75

75

ROUTING PATTERNS

76

76

Separate Messages

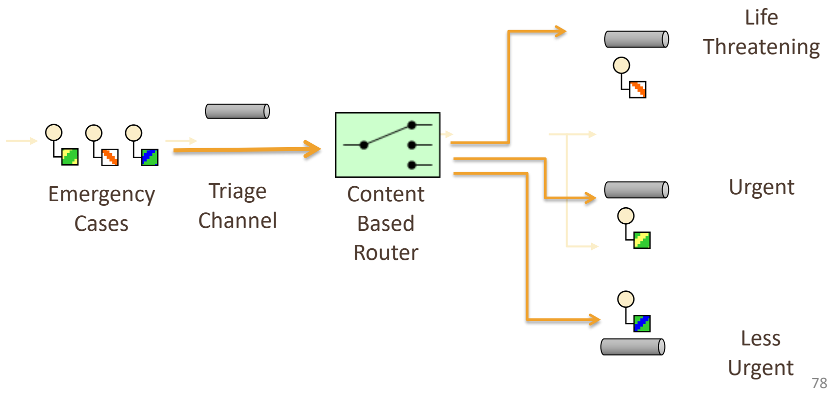
- Separate messages based on:
  - Attributes
    - Ex: urgent vs non-urgent patients
    - Ex: high vs low security
  - Types
    - Ex: treatment types (drugs, radiology, surgery, …)
    - Ex: prescription requests, appointments, consultations

77

77

Content Based Router

- Redirects messages based on attributes, types
- One input channel, multiple output channels

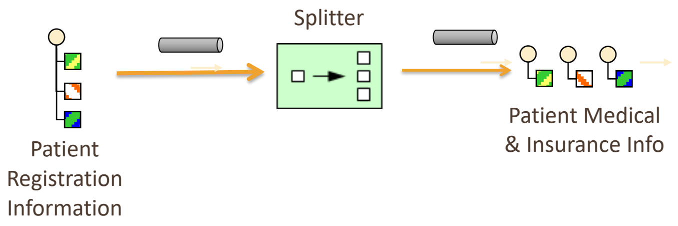


78

78

Splitter

- Break a compound message into smaller messages
- Typically followed by content-based router

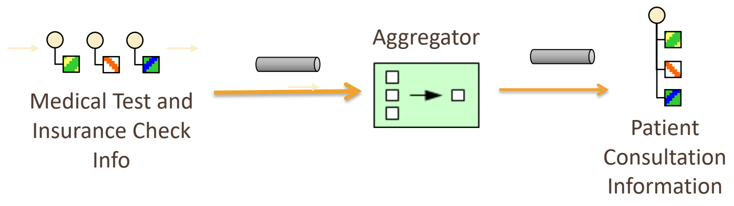

Splitter
Patient Registration Information
Patient Medical & Insurance Info

79

79

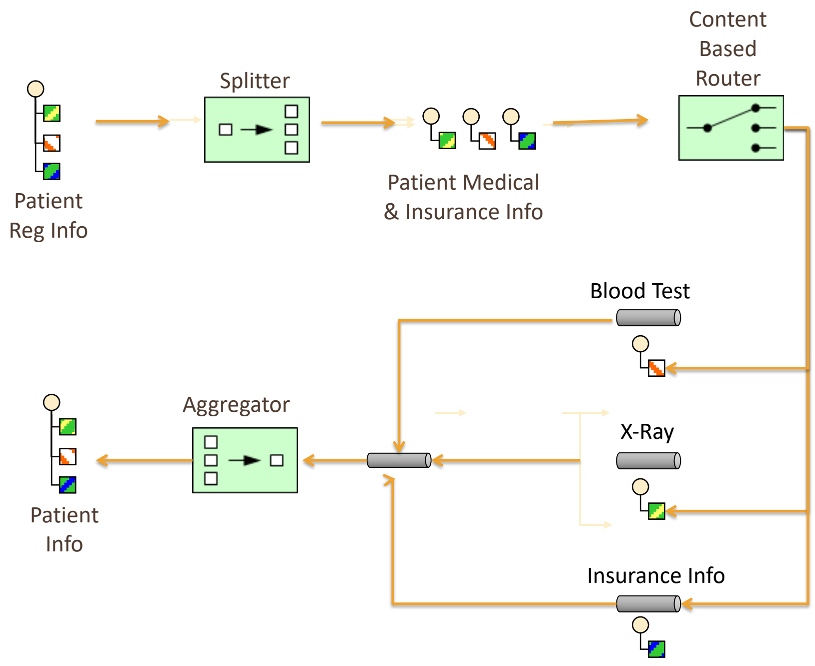
Aggregator

- Collects and stores messages until a complete set has been received
  - *Completeness condition*
  - *Aggregation algorithm*
- Typically preceded by Splitter or Publish-Subscribe


Medical Test and Insurance Check Info
Aggregator
Patient Consultation Information

80

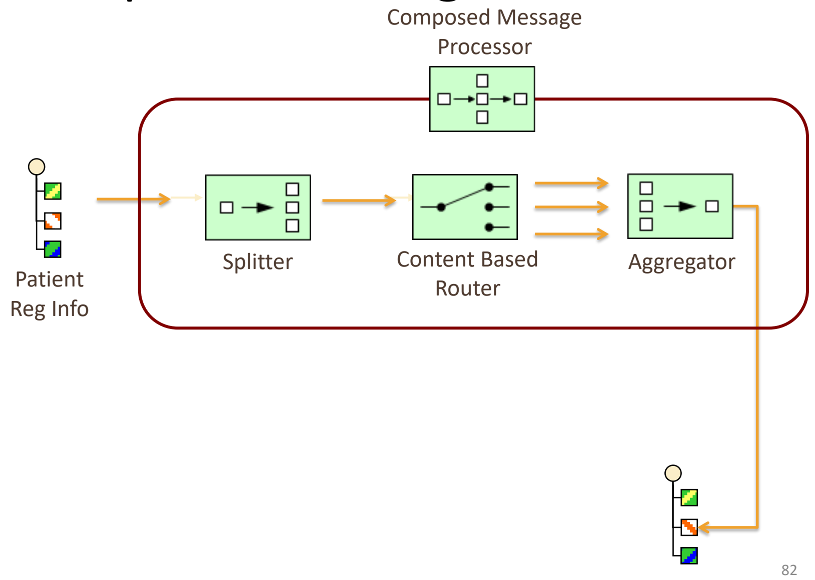
80



81

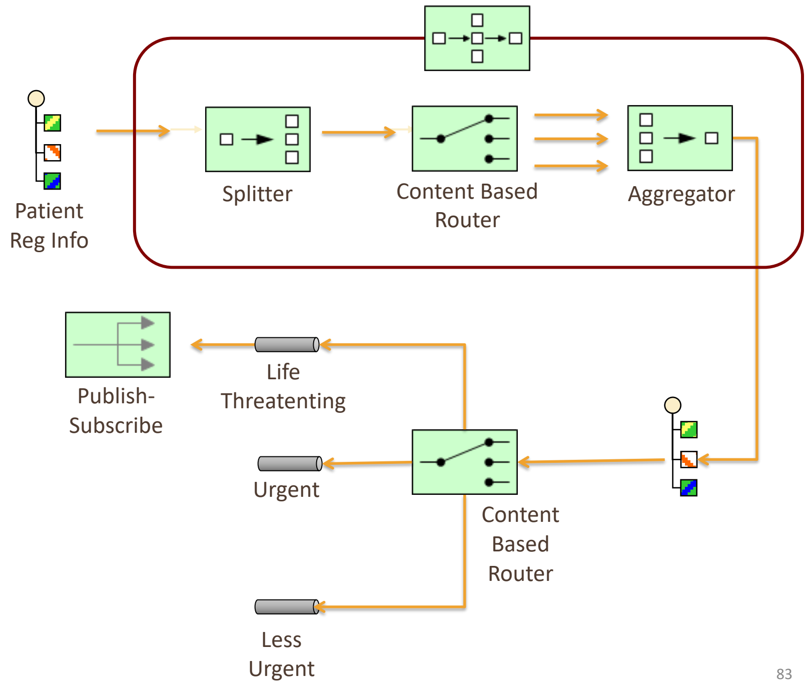
81

Composed Message Processor



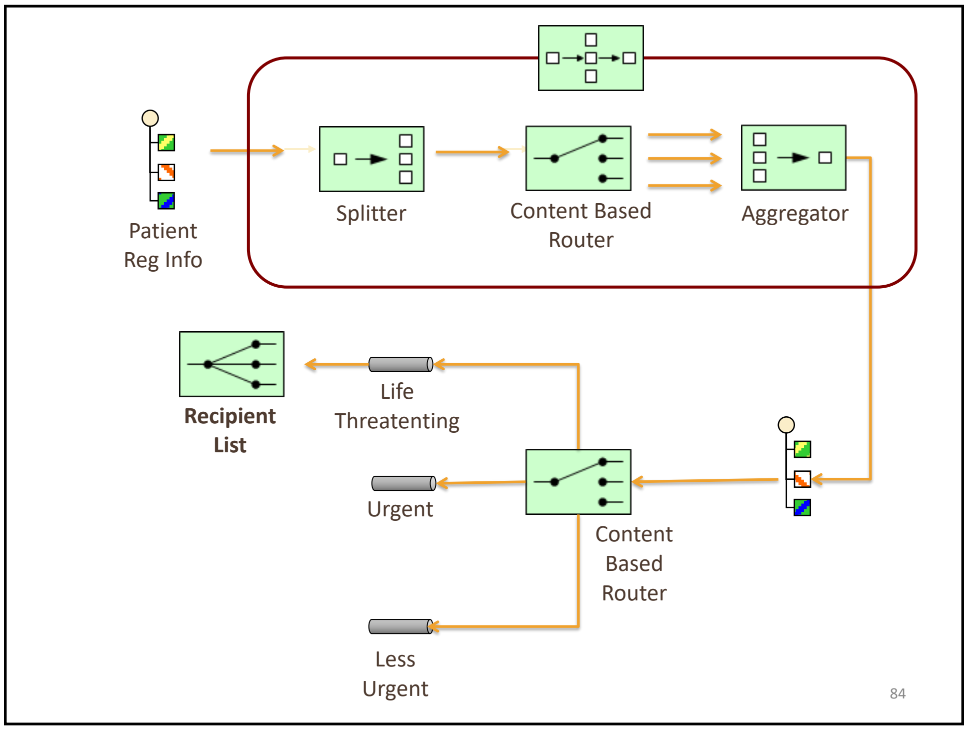
82

82

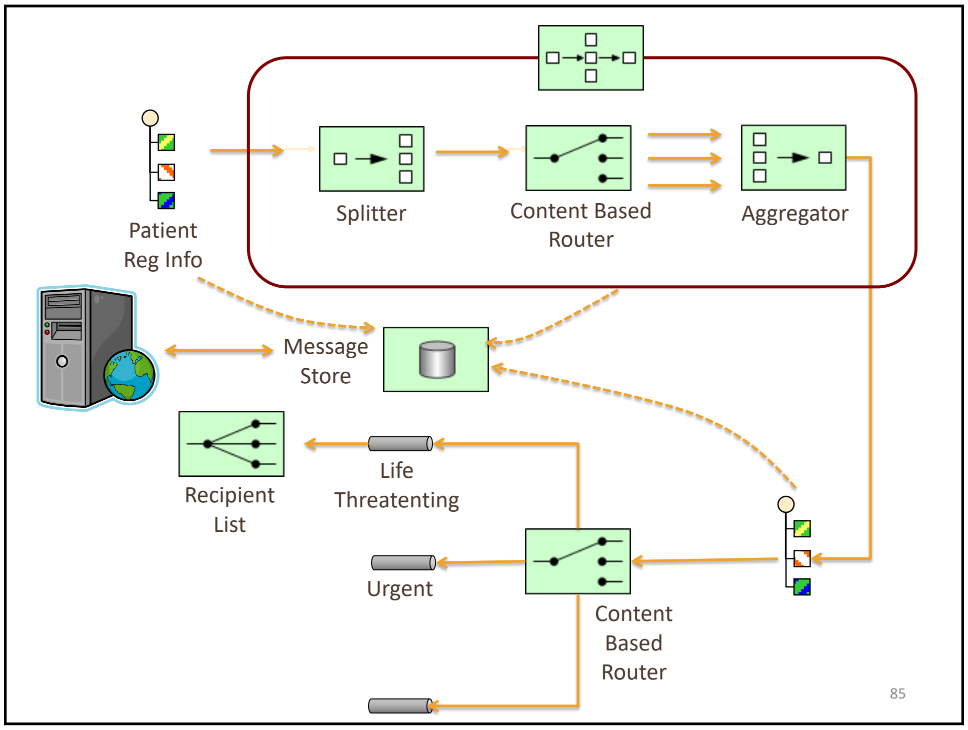


83

83

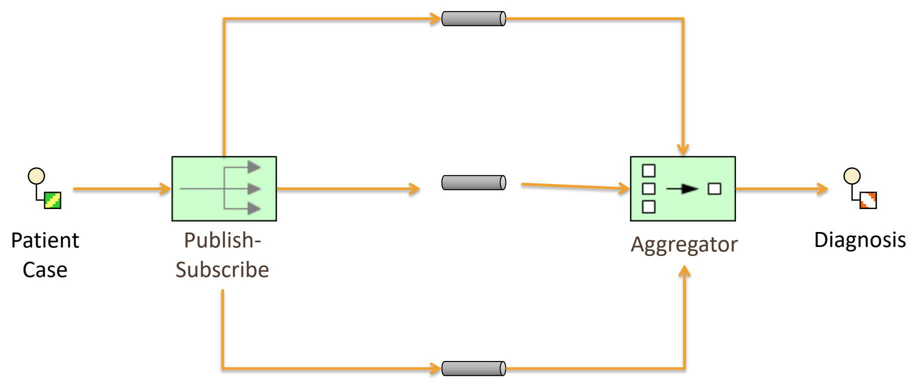

Patient
Reg Info
Splitter
Content Based
Router
Aggregator
Recipient
List
Life
Threatenting
Urgent
Content
Based
Router
Less
Urgent
84

84


Patient
Reg Info
Splitter
Content Based
Router
Aggregator
Message
Store
Recipient
List
Life
Threatenting
Urgent
Content
Based
Router
85

85

Pattern: Scatter-Gather



86

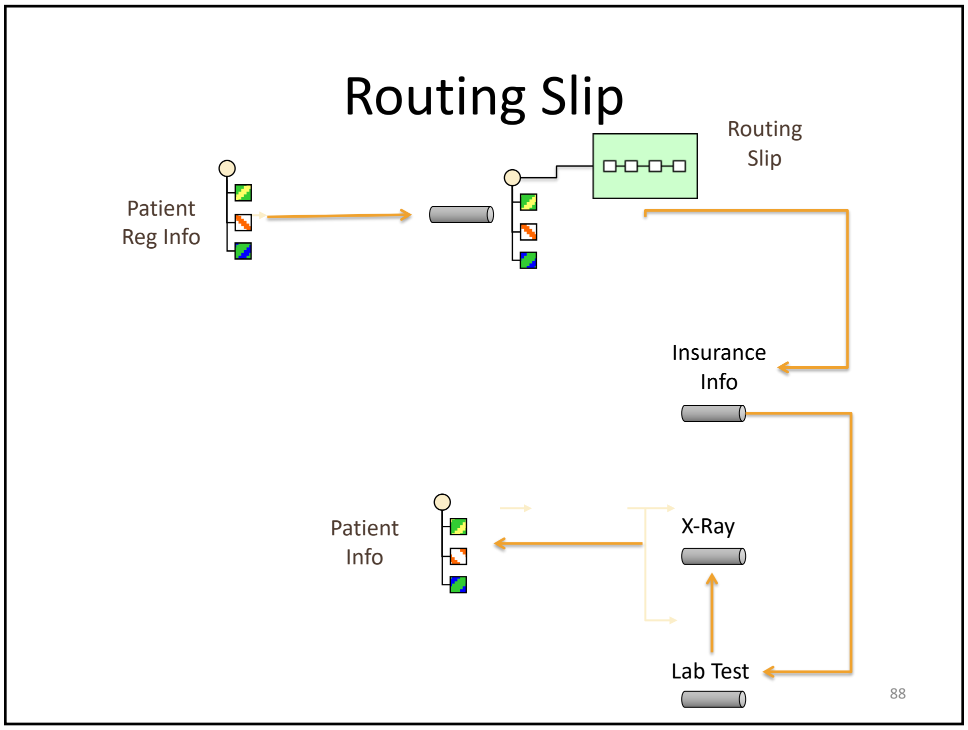
86

Routing Slip

- Publish-subscribe, Splitter & Content Router:
  - Process components in parallel
- Routing Slip:
  - Process sequentially
  - Ensure that prerequisites satisfied at each step
- Process Manager:
  - Runtime routing decisions
  - Dynamic logic for ensuring prerequisites

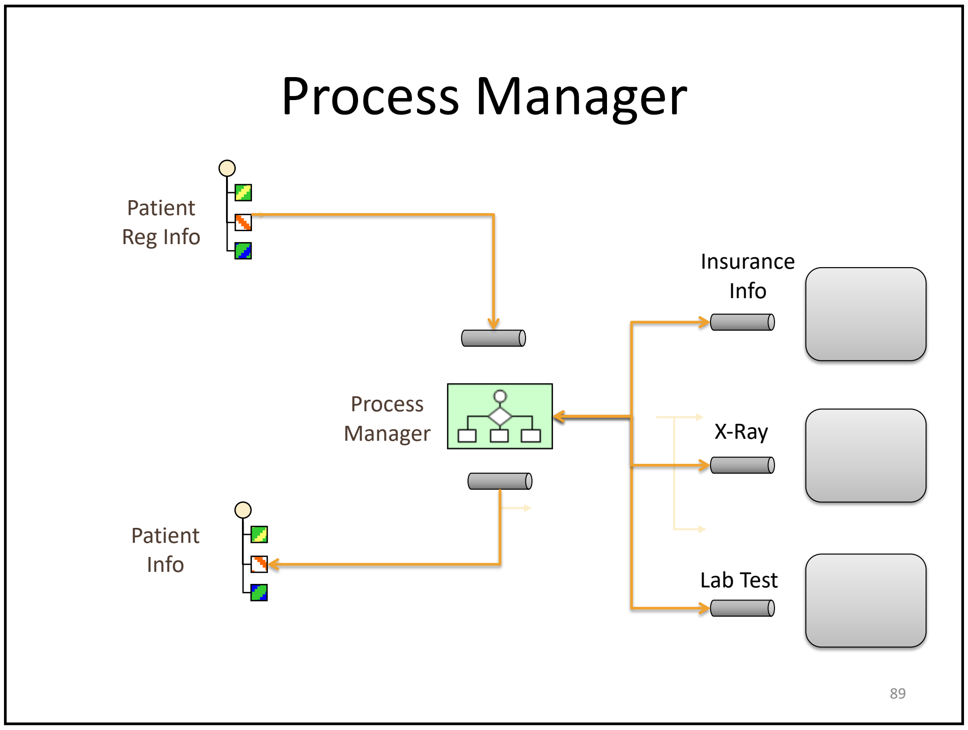
87

87



Routing Slip

88

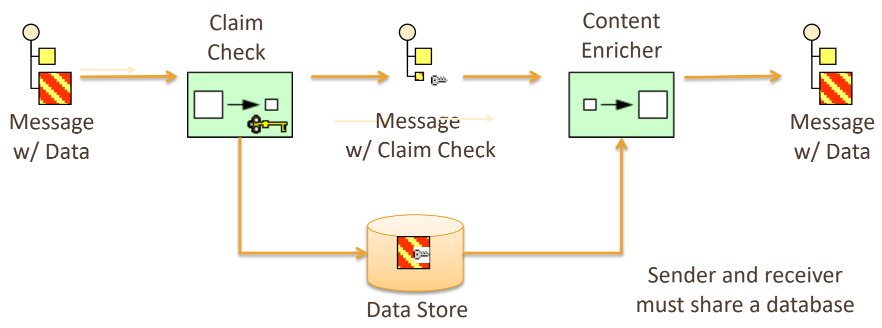


Process Manager

89

Claim Check

- Store the data, just transmit the key
  – Suitable for very large message contents
  – Reduces data volume without sacrificing content



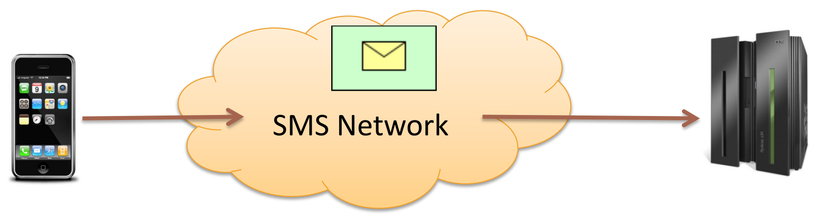
Sender and receiver must share a database

90

90

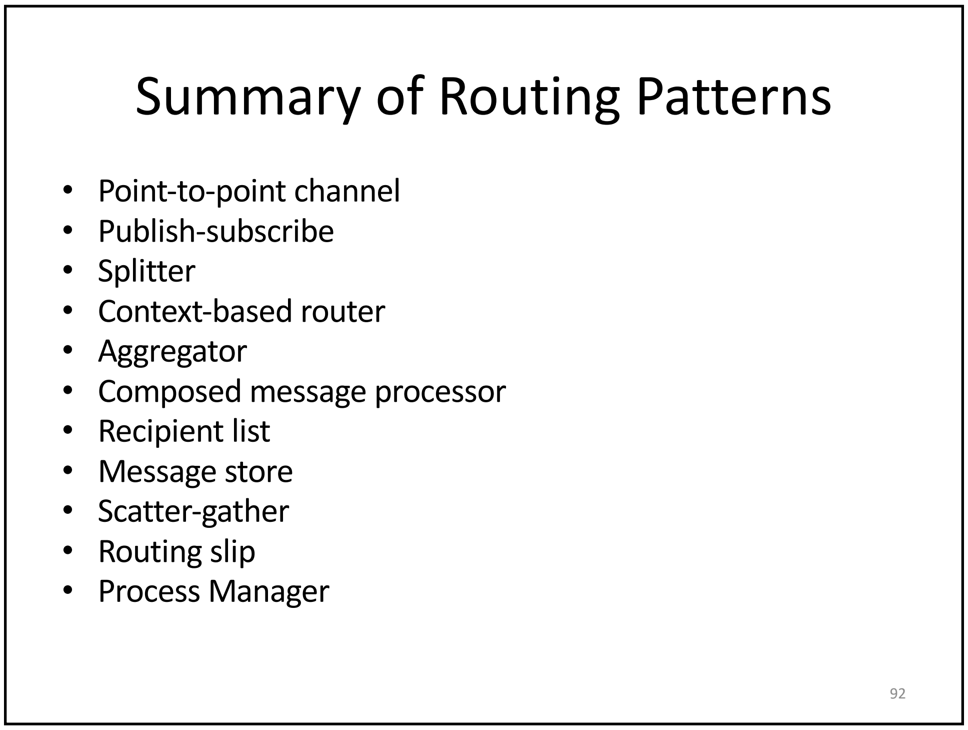
Wrapper Envelope

- Tunneling a message, with headers, as data through middleware



91

91

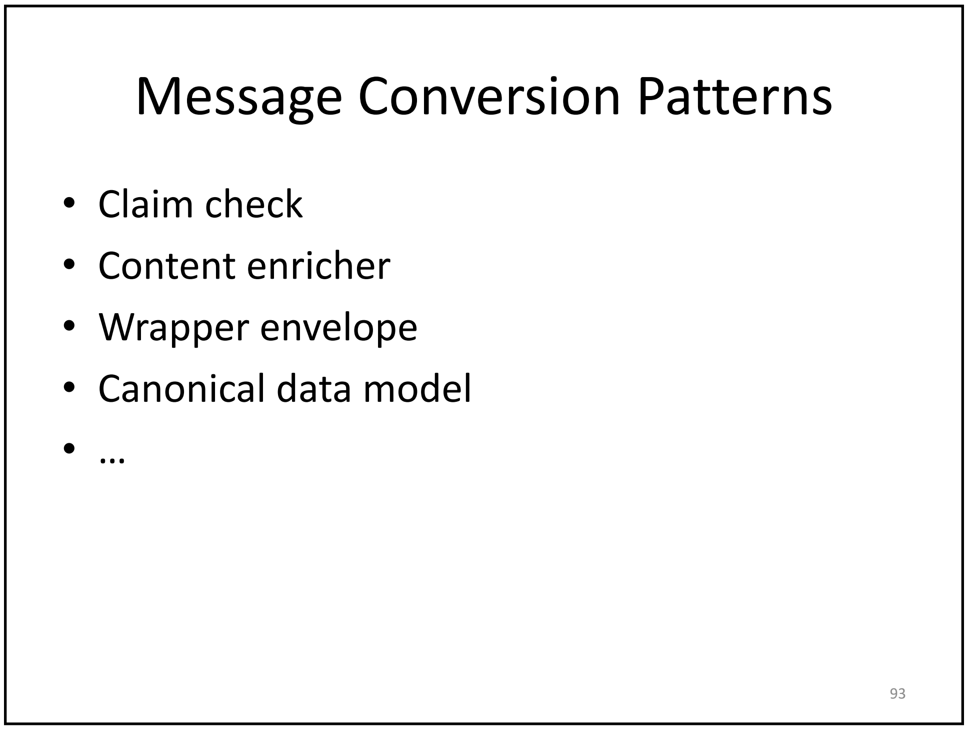


Summary of Routing Patterns

- Point-to-point channel
- Publish-subscribe
- Splitter
- Context-based router
- Aggregator
- Composed message processor
- Recipient list
- Message store
- Scatter-gather
- Routing slip
- Process Manager

92

92



Message Conversion Patterns

- Claim check
- Content enricher
- Wrapper envelope
- Canonical data model
- …

93

93