Enterprise Integration Patterns

Gregor Hohpe
「グレガー　ホペ」
www.eaipatterns.com

パターンワーキンググループ

April 20, 2004

ThoughtWorks
The art of heavy lifting.

"As applications become more connected, asynchronous messaging will be the next big architectural trend.

Similar to the shift procedural to Object-Oriented Programming, we need patterns and best practices to help us master the new paradigm."

Copyright ThoughtWorks 2004. All rights reserved.

Coupling

ThoughtWorks® The art of heavy lifting™

- Coupling = Measure of dependency between applications
  - Technology Dependency
  - Temporal Dependency
  - Location Dependency
  - Data Format Dependency
- Coupling is a Range, not a Binary Property
- Coupling is not inherently good or bad

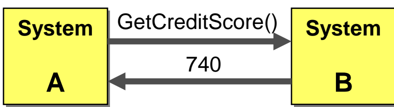
> "How do you make two systems loosely coupled? Don't connect them."
> -- David Orchard, BEA

Copyright ThoughtWorks 2004. All rights reserved.

Remote Procedure Calls (RPC)

ThoughtWorks® The art of heavy lifting™

- "Natural" extension of local programming model to remote applications



- Allows one application to invoke a method inside another application
- Details of remote communication are largely hidden from programmer
- Java RMI, .Net Remoting, DCE RPC

Copyright ThoughtWorks 2004. All rights reserved.

RPC's Can Be a Dangerous Illusion

ThoughtWorks® The art of heavy lifting™

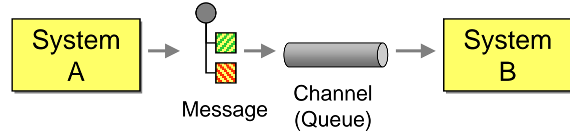
- Tight coupling just as local method call
- The RPC approach ignores:
  - Latency
  - Lack of shared memory access
  - Partial failure and concurrency

> "Objects that interact in a distributed system need to be dealt with in ways that are intrinsically different from objects that interact in a single address space."
>
> -- Waldo et al, 1994

Copyright ThoughtWorks 2004. All rights reserved.

Message-Oriented Communication

ThoughtWorks® The art of heavy lifting™



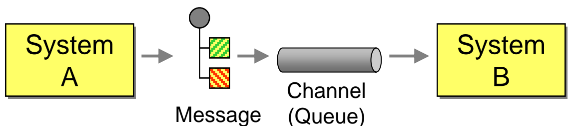
- Systems communicate via *Channels*
- Channels have logical (location-independent) addresses
- Data is encapsulated in messages in a technology-neutral format, e.g. XML
- The sending application places a message into the Channel and goes on to other work ("fire-and-forget")
- The Channel queues the data until the receiving application is ready to consume it (FIFO)

Copyright ThoughtWorks 2004. All rights reserved.

Message-Oriented Communication

ThoughtWorks® The art of heavy lifting™

Message-oriented communication provides loose coupling and reliability



- Channels are separate from applications
- Channels are asynchronous & reliable (queues)
- Use a common data representation, e.g. XML
- Messages are self-contained

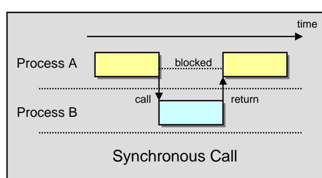
- ➢ Removes location dependencies
- ➢ Removes temporal dependencies
- ➢ Removes technology dependencies
- ➢ Removes data format dependencies

Copyright ThoughtWorks 2004. All rights reserved.

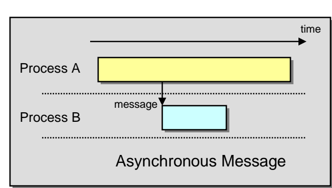
Asynchrony

ThoughtWorks® The art of heavy lifting™

- Most applications are synchronous
  - The caller waits until a method completes
  - The call stack manages local state and keeps track of the return address
- Messaging is inherently asynchronous
  - The calling program continues after the message is sent
  - Need to manage state manually
  - Results may come back at any time via an asynchronous message

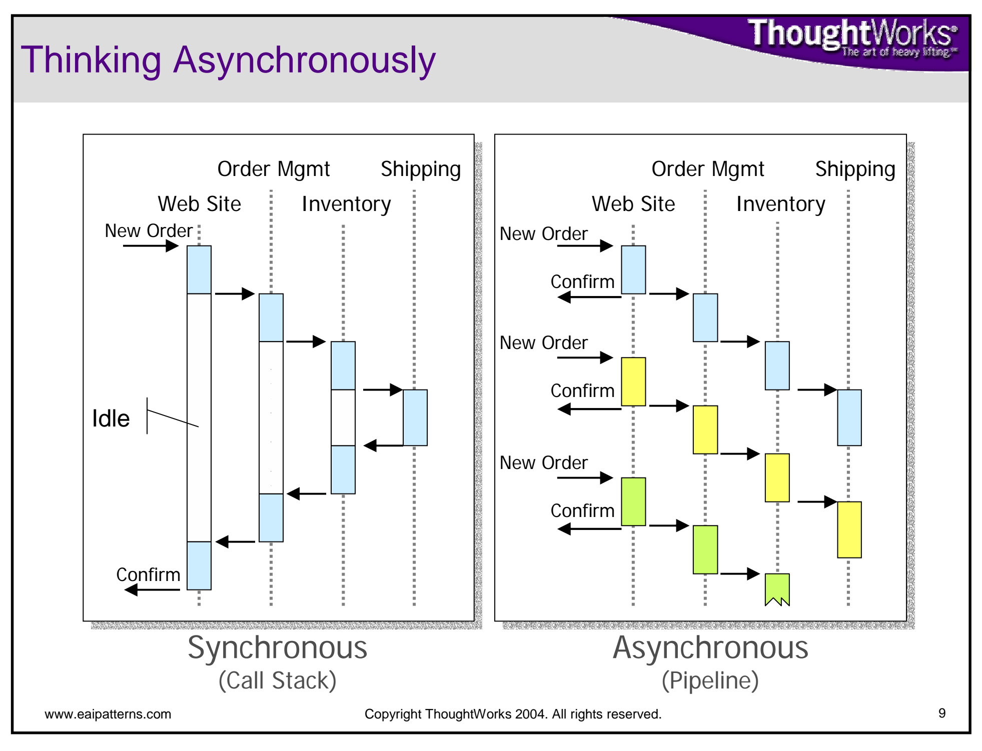


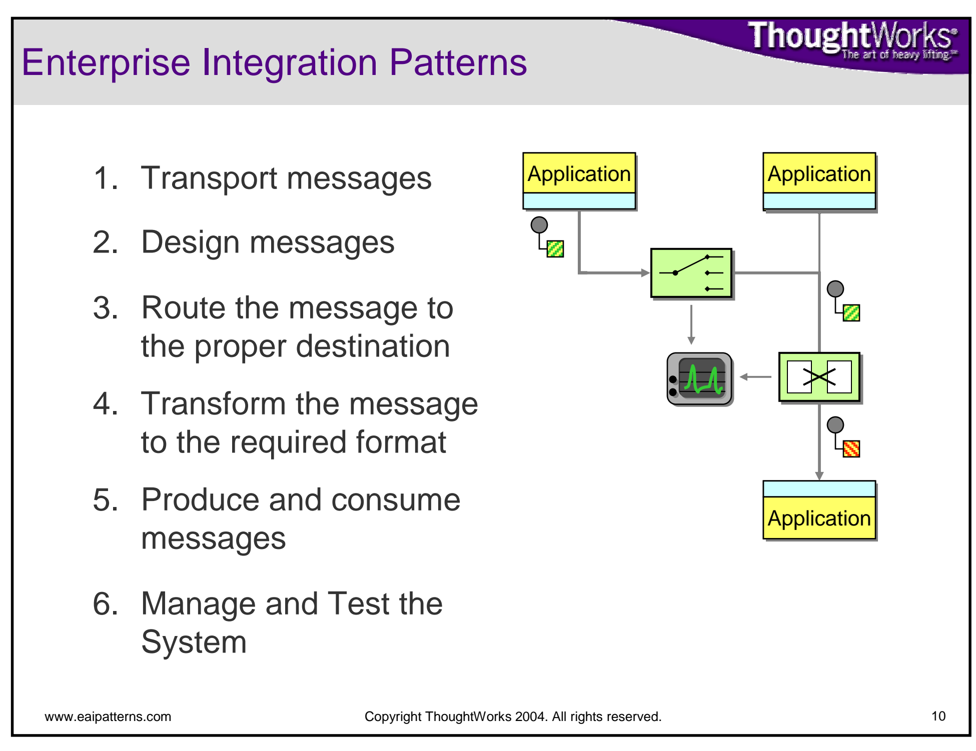
Synchronous Call

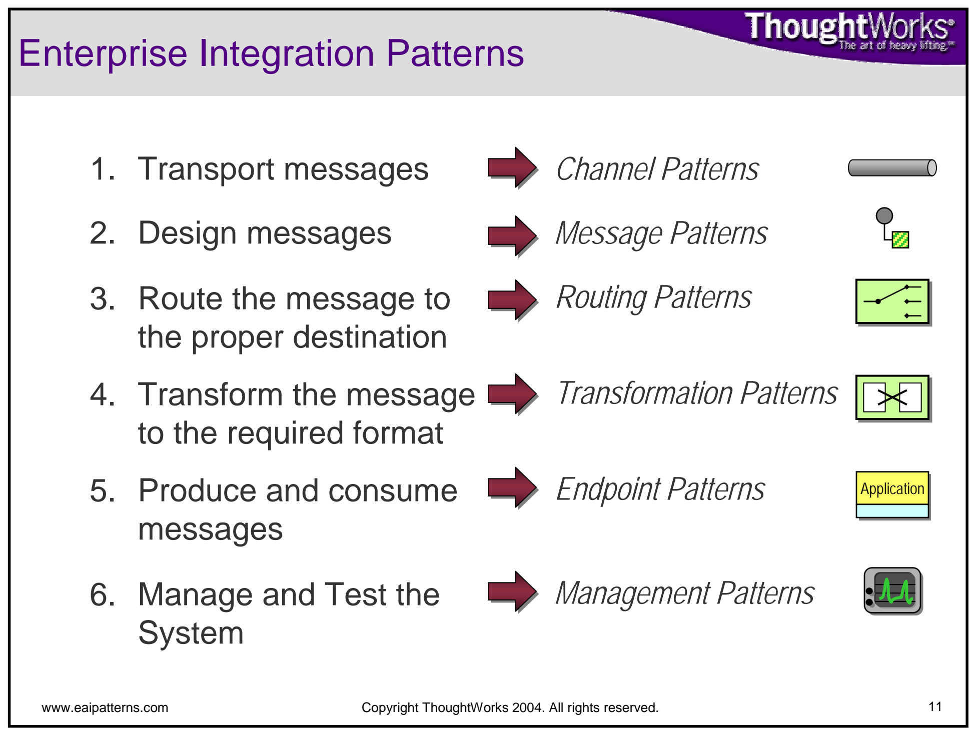


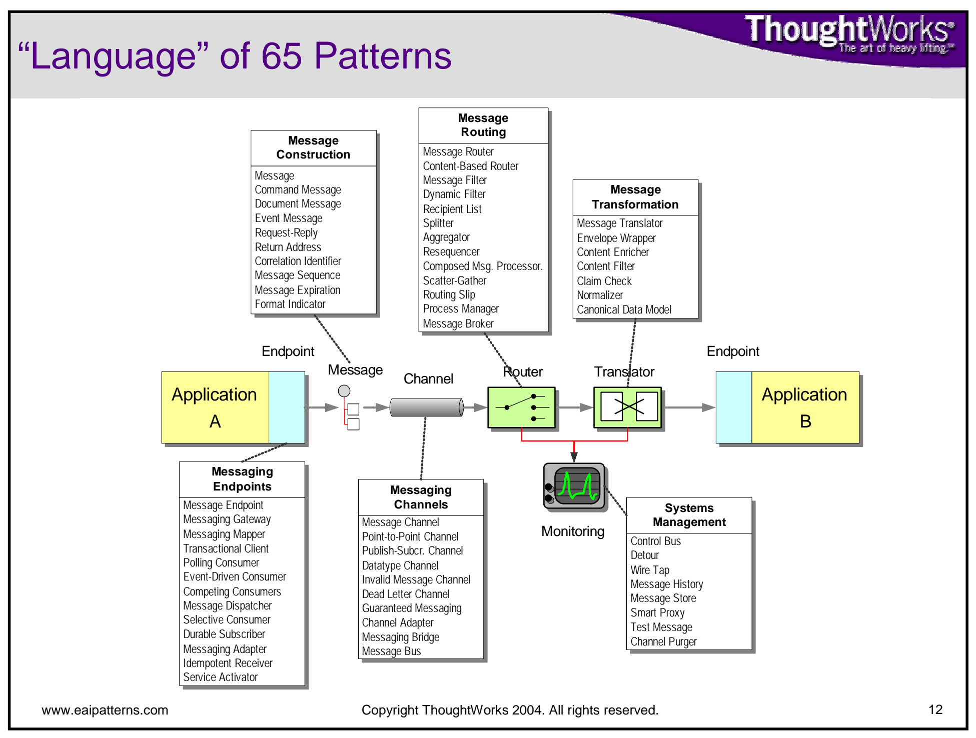
Asynchronous Message

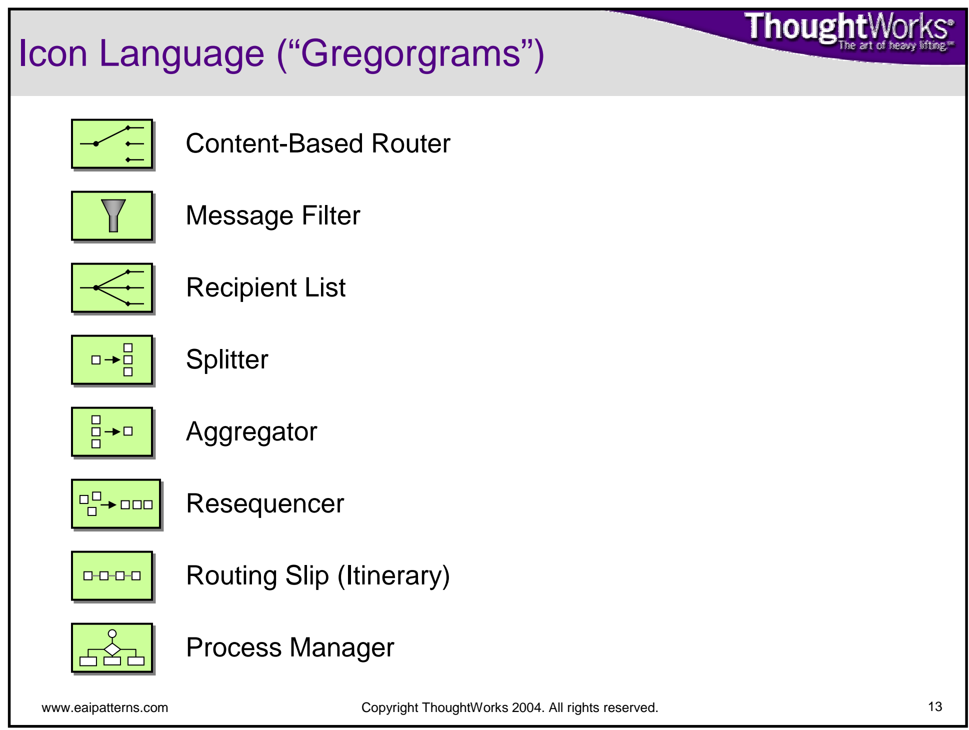
Copyright ThoughtWorks 2004. All rights reserved.


ThoughtWorks
The art of heavy lifting
Thinking Asynchronously
Order Mgmt
Shipping
Web Site
Inventory
New Order
Idle
Confirm
Order Mgmt
Shipping
Web Site
Inventory
New Order
Confirm
New Order
Confirm
New Order
Confirm
Synchronous
(Call Stack)
Asynchronous
(Pipeline)
www.eaipatterns.com
Copyright ThoughtWorks 2004. All rights reserved.
9


ThoughtWorks
The art of heavy lifting
Enterprise Integration Patterns
1. Transport messages
2. Design messages
3. Route the message to the proper destination
4. Transform the message to the required format
5. Produce and consume messages
6. Manage and Test the System
Application
Application
Application
www.eaipatterns.com
Copyright ThoughtWorks 2004. All rights reserved.
10


Enterprise Integration Patterns
ThoughtWorks
The art of heavy lifting
1. Transport messages
Channel Patterns
2. Design messages
Message Patterns
3. Route the message to the proper destination
Routing Patterns
4. Transform the message to the required format
Transformation Patterns
5. Produce and consume messages
Endpoint Patterns
Application
6. Manage and Test the System
Management Patterns
www.eaipatterns.com
Copyright ThoughtWorks 2004. All rights reserved.
11


"Language" of 65 Patterns
ThoughtWorks
The art of heavy lifting
Message Construction
Message
Command Message
Document Message
Event Message
Request-Reply
Return Address
Correlation Identifier
Message Sequence
Message Expiration
Format Indicator
Message Routing
Message Router
Content-Based Router
Message Filter
Dynamic Filter
Recipient List
Splitter
Aggregator
Resequencer
Composed Msg. Processor.
Scatter-Gather
Routing Slip
Process Manager
Message Broker
Message Transformation
Message Translator
Envelope Wrapper
Content Enricher
Content Filter
Claim Check
Normalizer
Canonical Data Model
Endpoint
Message
Channel
Router
Translator
Endpoint
Application A
Application B
Monitoring
Messaging Endpoints
Message Endpoint
Messaging Gateway
Messaging Mapper
Transactional Client
Polling Consumer
Event-Driven Consumer
Competing Consumers
Message Dispatcher
Selective Consumer
Durable Subscriber
Messaging Adapter
Idempotent Receiver
Service Activator
Messaging Channels
Message Channel
Point-to-Point Channel
Publish-Subcr. Channel
Datatype Channel
Invalid Message Channel
Dead Letter Channel
Guaranteed Messaging
Channel Adapter
Messaging Bridge
Message Bus
Systems Management
Control Bus
Detour
Wire Tap
Message History
Message Store
Smart Proxy
Test Message
Channel Purger
www.eaipatterns.com
Copyright ThoughtWorks 2004. All rights reserved.
12



Example: *Aggregator* (abbrev.)

ThoughtWorks
The art of heavy lifting

- Problem
  - How do you combine the results of individual, but related messages back into a single message?
- Forces
  - Messages may be out of sequence
  - Messages may be delayed, when to stop?
  - Avoid burdening receiver with these issues
- Solution
  - Use a stateful filter, an Aggregator, to collect and store individual messages until it receives a complete set of related messages. Then, the Aggregator publishes a single message distilled from the individual messages.
- Sketch

Item 1 Item 2 Item 3 → **Aggregator** → Validated Order

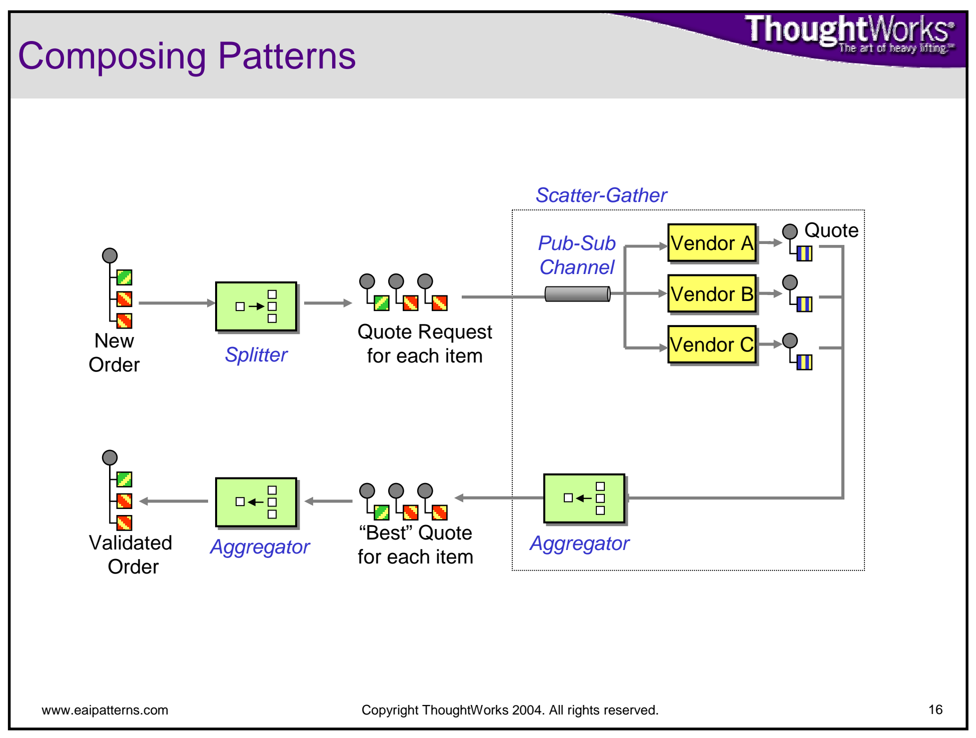
Copyright ThoughtWorks 2004. All rights reserved.

Aggregator Design Decisions

- Correlation
  - Which incoming messages belong together?
- Completeness Condition
  - Wait for all
  - Time out (absolute, incremental)
  - First best
  - Time box with override (Buy Now”)
  - External event
- Aggregation Algorithm
  - Single best answer
  - Condense data (e.g., average)
  - Concatenate data for later analysis

Copyright ThoughtWorks 2004. All rights reserved.

Composing Patterns

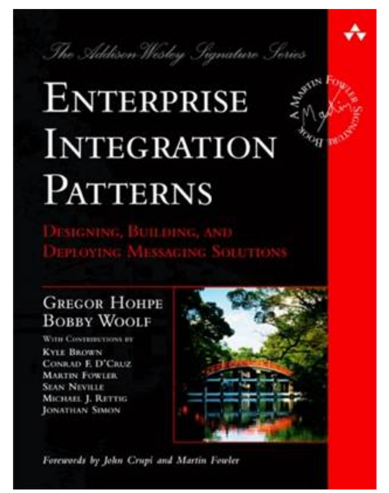


Copyright ThoughtWorks 2004. All rights reserved.

Enterprise Integration Patterns

ThoughtWorks® The art of heavy lifting™

- Language of 65 patterns
- Vocabulary and notation
- Focuses on asynchronous messaging
- Based on pipes-and-filters model of message flow
- Many more patterns to harvest:
  - Conversations
  - Orchestrations
  - Error Handling
  - Complex Transformations
  - Rules Engines



Copyright ThoughtWorks 2004. All rights reserved.

For more information…

ThoughtWorks® The art of heavy lifting™

ghohpe@thoughtworks.com

www.eaipatterns.com

www.thoughtworks.com

Copyright ThoughtWorks 2004. All rights reserved.