Conversations between Loosely Coupled Systems

Gregor Hohpe
gregor@hohpe.com

Today's Conversation

Coupling
Messaging
Conversations
Describing Conversations
Standards
Patterns
Resources
Questions
Coupling (Once Again)

Looser Coupling
= Fewer Dependencies
= More Independent Variability

“How do you make two systems loosely coupled? Don’t connect them.”
- David Orchard, BEA

Messaging

System A ➔ Message ➔ Channel (Queue) ➔ System B

The Poster Child of Loose Coupling

- Systems communicate via Channels ➔ Level of indirection
- Location-independent channel names ➔ Location decoupling
- “Fire-and-forget” ➔ Temporal decoupling
- Simplified interaction ➔ Conversational decoupling
So Far, So Good, But…

- Messaging is loosely coupled, composable, elegant, scalable etc.
- But is sending a simple message enough?
- Multi-step Interactions
  - quote, order, shipment, invoice, payment, refund
- Distributed Transactions
  - 2-phase commit, long-running transactions
- Error handling
  - compensation, retry

Conversation = Series of Related Messages

- Conversations can span seconds, hours, days
- Multiple conversation instances at the same time
- Messages belonging to one conversation are correlated (typically through identifier)
Why Do We Care?

- OOAD focuses on structural aspects of the solution (inheritance, composition, aggregation)
- SOA shifts attention to interaction (messaging, stateless servers)
- Messages are expensive
- Part of an expressive contract between parties
- Conversation rules impact coupling

Describing Conversations

- Message Exchange Patterns
- Endpoint Process
- Choreography
- Temporal Logic
Message Exchange Patterns

- Perspective of Service Provider
- Single Requestor
- Described as linear sequence of input-output elements
- Documented by Example
- Part of WSDL 1.1
- More in WSDL 2.0

One-way

Request-Response

Solicit-Response

(Fault)

Notification

Endpoint Processes

- A process supports the desired exchange of messages through send and receive activities
- Each conversation corresponds to one process instance
- Each participant has a (potentially different) process definition
Public and Private Processes

- The private process is more complex than the public process
- Details of the private process may not be shared

Public process sends & receives messages and invokes private processes through orchestration

Process "Inheritance" – verify compliance of a private process with a public process template (Wil v.d. Aalst)

Choreography

- Describe the interaction as a flow of messages: sequence, parallel, choice
- Conversational state machine
- Generate endpoint processes from the conversation model
- Not limited to 2 participants
Temporal Logic

- Describes rules of the conversation as predicates
- Operators X (next), G (globally), U(until), F(eventually)
- Declarative
- Can be used to verify message exchanges

G(Order ⇒ F(Invoice ∧ Drinks))
G(Invoice ⇒ F(Payment))

Standards – WS-BPEL

- Business Process Execution Language
- Process-oriented Composition
- Describe endpoint processes
- Executable through process engine
- Composite service exposed via WSDL
- Sending and receiving of messages, control constructs, control links
- Correlation, compensation, dead path elimination
Standards – WS-BPEL

- Choreography Description Language
- Used to describe conversation “activities”
- Declarative, not executable, can be used to validate
- Generate endpoint processes
- Global view of all interactions between participants
- Time-out, extracting tokens from messages

Standards – WS-CDL

- Roles
- Relationships
- Work Units
- Activities
  - Sequence
  - Parallel
  - Choice
- Interaction
  - Channel
  - From / To
  - Operation
Conversations Between Loosely Coupled Systems

Standards – WS-CDL

Model

Specification

"Compiler"

Implementation

Company A

Company B

SOAP Service Description Language

Multiple Protocol Frameworks

- MEP
- CSP
- Rules
- Sequencing Constraints

www.ssdl.org

Specification - SSDL

<rls:rule>
<ssdl:msgref ref="Invoice" direction="out" />
<rls:condition>
<rls:and>
<ssdl:msgref ref="Order" direction="in" />
<rls:not>
<ssdl:msgref ref="Invoice" direction="out" />
</rls:not>
</rls:and>
</rls:condition>
</rls:rule>
Description Languages

- Provide syntax and base vocabulary to express conversations
- Are intended for consumption by machines
- Enable a broad range of solutions
- Do not tell us how to design a "good" conversation
- Or for that matter, what distinguishes a "good" conversation from a "bad" one

Conversation Patterns

- Provide vocabulary at a higher level of abstraction
- Are intended for consumption by humans
- Investigate a specific usage scenario at a time
- Catalog common scenarios and offer solutions
- Focus on design intent and trade-offs

"A pattern is a mind-sized chunk of information"

--Ward Cunningham
Conversations Between Loosely Coupled Systems

Request-Reply

- Simplest conversation
- Single Conversation state: waiting for reply, complete
- More complicated once error conditions considered

Request-Reply with Retry

- Sender can repeat request \( n \) times
- Provider has to be idempotent
- Sender might receive responses after it gave up
- Example: RosettaNet Implementation Framework (RNIF)
Layered Conversation Protocols

Top Layer

- Request
- Awaiting Response
- Send Request
- Wait for Reply
- Max retry?
- [yes]
- [no]
- Timeout
- Received
- Done
- Fault

Bottom Layer

Dynamic Discovery

1. Broadcast request for provider
2. Provider(s) consider whether to respond (load, suitability)
3. Interested providers send responses
4. Requestor chooses “best” provider from responses
5. Requestor initiates interaction with chosen provider
   - Examples: DHCP, TIBCO Repository discovery
Subscribe-Notify (Multi-responses)

- Subscriber expresses interest in receiving notifications
- Subscriber receives messages until a stop condition is reached:
  - Subscriber sends a stop request
  - A deadline is reached without the subscriber renewing interest
  - Subscriber does not respond to requests from provider
  - Provider notifies subscriber of end of transmission
    Example: WS-Eventing, WS-Notification

Renewing Interest

- “Lease” model
- Heartbeat / keep-alive
- Subscriber has to renew actively
  Example: Jini

- “Magazine Model”
- Subscriber can be simple
- Provider has to manage state for each subscriber
Reaching Consensus

- Coordinated conversation
- Coordinator tracks responses from each participant, makes a decision and then broadcasts the decision
- Example:
  - Two-phase commit

Summary

- Interaction steps into the foreground
- New languages to describe conversations
- Process-focused or declarative
- New set of design patterns
- Focus on motivation and forces over language syntax
Resources - Print

- **Enterprise Integration Patterns**
  Gregor Hohpe, Bobby Woolf
  Addison-Wesley, 2004

- **Essential Business Process Modeling**
  Havey O'Reilly, 2005

- **Issues in Agent Communication**
  Dignum (Ed.)
  Springer, 2000

- **Web Services**
  Alonso et al.
  Springer, 2004

- **Service-Oriented Architecture**
  Erl
  Prentice Hall, 2005

- **Business Process Execution Language**
  Juric
  Packt, 2006

Resources - Online

- [www.eaipatterns.com/ramblings.html](http://www.eaipatterns.com/ramblings.html)
  – My Blog
- [www.serviceinteraction.com](http://www.serviceinteraction.com)
  – Animated interaction patterns
- [www.conversationpatterns.com](http://www.conversationpatterns.com)
  – To go live soon
- [www.ssdl.org](http://www.ssdl.org)