SOACON 2007
Where IT Aligns With Business

www.eaipatterns.com

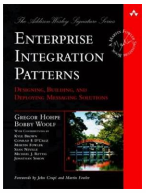
Gregor Hohpe | Google

Developing in a Service-oriented World

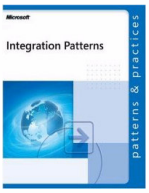
Who's Gregor?

- Distributed systems, enterprise integration, service-oriented architectures
- MQ, MSMQ, JMS, TIBCO, Web Services
- Write code every day. Share knowledge through patterns.

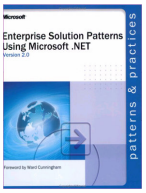
eaipatterns.com
- Patterns
- Articles
- Blog



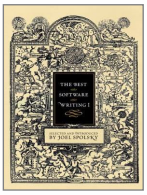
Enterprise Integration Patterns
Addison-Wesley



Integration Patterns
Microsoft Press



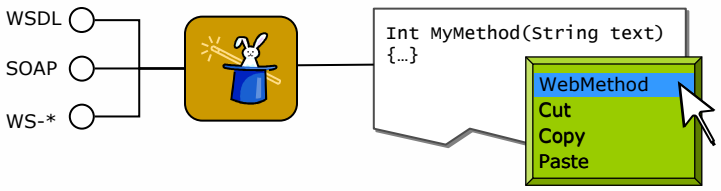
Enterprise Solution Patterns
Microsoft Press



Best Software Writing I
APress

SOA Experten-wissen
dpunkt Verlag

Could It Be So Easy?

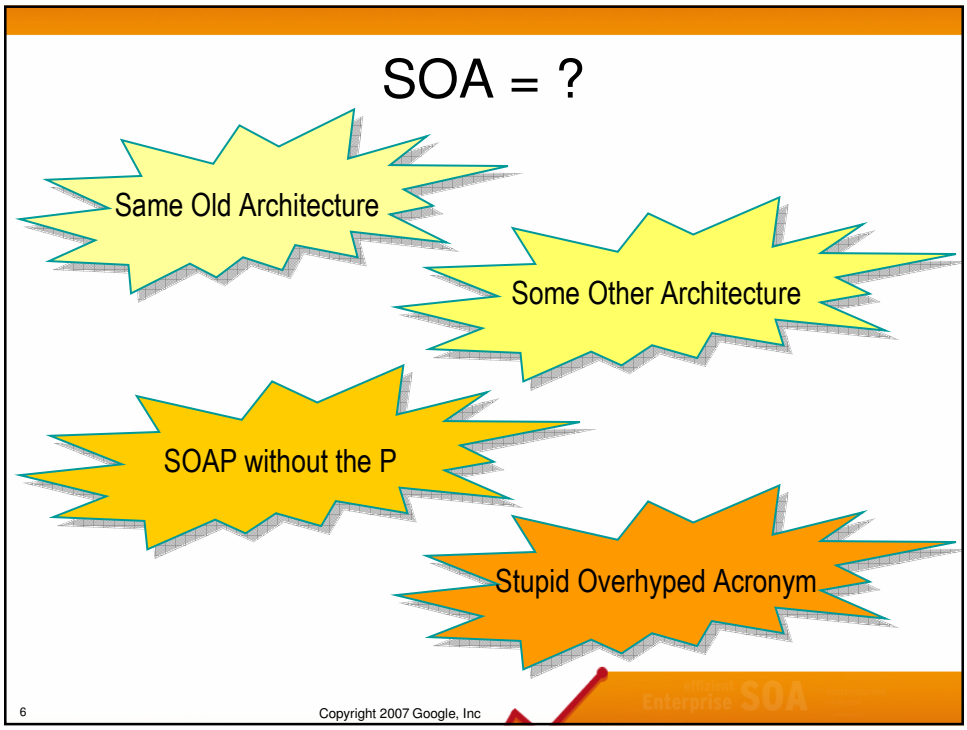


- Buzzword compliant, but not a service-oriented architecture
- Synchronous call stack mentality
- No interface-implementation separation

Advice for Aspiring SOA Developers

- Forget about SOAP
- Become good at PowerPoint
- Pay close attention to Starbucks
- Shred “Design Patterns” (or eBay it)
- PROLOG rocks
- Replace MDA with ADM


SOACON
2007
Where IT Aligns With Business
How Did We Get Here?
PART I


SOA = ?
Same Old Architecture
Some Other Architecture
SOAP without the P
Stupid Overhyped Acronym
6
Copyright 2007 Google, Inc

Service-Oriented Architecture

- Service
  - Well-defined, Self-contained
  - Independent of consumer context (mostly)
  - Universally accessible without individual deployment
- Service-Oriented Architecture
  - An architectural style
  - A simple, document-oriented interaction model
  - Loose(r) coupling
  - Interface contracts, registry
  - Functional assets reside in services, explicit orchestration across services

Distributed Component Architectures

- Main driver: transparency to developer
  - Remote code looks like local code
- The Distributed Object approach ignores:
  - Latency (network, marshalling, applications)
  - Disconnected or intermittently connected networks
  - Lack of shared memory access (pointers, references)
  - Partial failure and concurrency
  - Independent variability between systems (coupling)

Distributed Component Architectures

"The first law of distributed objects: Don't distribute your objects"
-- Martin Fowler

"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

"95% transparent is not good enough. In fact, it is worse because it deceives developers."
-- Werner Vogels

Service Oriented Integration

Defining Characteristics

- Simplicity of interaction.
- No notion of inheritance, polymorphism, call stack, references etc.
- No lifecycle control. Service provider manages instances / allocations internally to suit its needs.
- Pass fewer, more self-contained documents. A tree structure (e.g., XML) is well suited for this.
- More amenable to asynchronous interaction.

Service Oriented Integration

Considerations

- Progress through Regress?
- Is the simplified interaction model sufficient? (WS-*)
- Are the contracts expressive enough?
- Are we getting it right this time around?
- When is SOA not appropriate?

SOACON 2007
Where IT Aligns With Business

What Now?

PART II

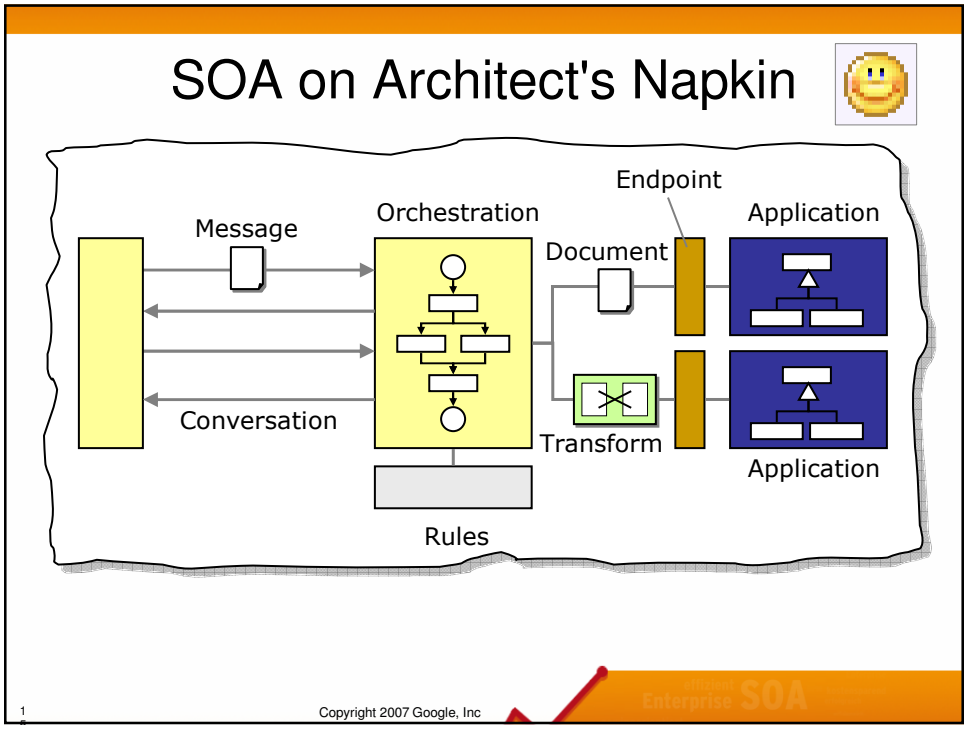
The Human Side of Service-Orientation

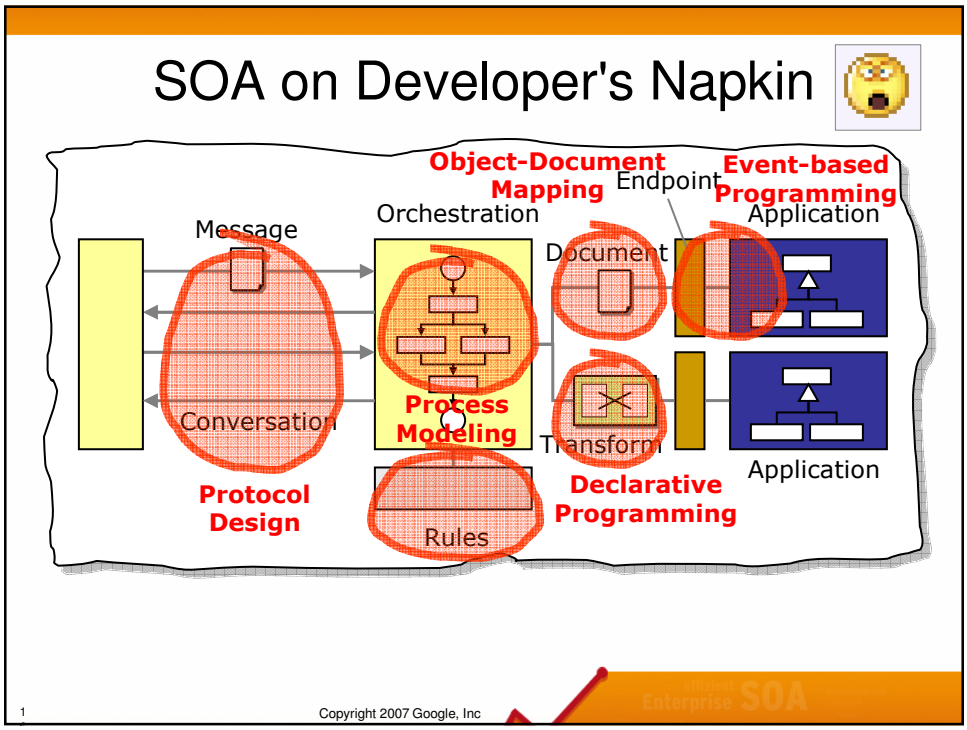
- Architectural style is based on patterns and intent, not technology selection.
- SOAP vs. Binary is only a very small part of the SOA puzzle.
- Conversation models, asynchrony, document-orientation, granularity, decoupling, management, etc. are much more important.

The Human Side of Service-Orientation

- Loose coupling means shared architectural vision and intent are critical.
- SOA is primarily an agreement on what *not* to do.
- Your compiler can't tell you if you violated SOA principles.
- In the near term, this means documentation. Yes, PowerPoint!

Copyright 2007 Google, Inc


SOA on Architect's Napkin
Endpoint
Orchestration
Application
Message
Document
Conversation
Transform
Application
Rules
Copyright 2007 Google, Inc

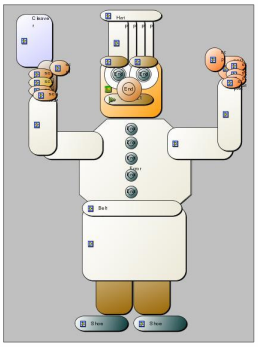

SOA on Developer's Napkin
Object-Document Mapping
Endpoint
Event-based Programming
Orchestration
Application
Message
Document
Process Modeling
Conversation
Transform
Application
Protocol Design
Declarative Programming
Rules
Copyright 2007 Google, Inc

New Programming Models in SOA

- Event-based, Asynchronous Programming
  - Explicit state management
  - Sequencing, timing uncertainty
- Declarative Programming
  - Execution path chosen at run-time
  - XSLT, Rules engines
- Object-Document Mapping
  - Analogous to O-R mapping: subtle, but important
- Process Modeling
  - Many concurrent, long-running instances
  - No two-phase-commit style transactions

"Doodleware" Only Limited Help

- For example
  - Graphical process editors
  - Graphical transformation editors
- We love pictures
- Programming in pictures tedious
  - Scalability issues
  - Diff, Merge mostly unsupported
- Often a thin veneer over a complex (or unfamiliar) programming paradigm



"EAI Art"

Understanding Technology

- Syntax
  - Basic language mechanism
  - Artefact of crude input devices
- Constructs
  - "Vocabulary": Objects, Classes, Interfaces, Inheritance, etc.
  - Easily explained but no guidance on good design
- Principles
  - Separation of Concerns, Open-Closed, etc.
  - Help evaluate a solution
- Patterns
  - Null Object, Decorator, Model-View-Controller
  - Concrete design guidance based on principles

Patterns – 10 Years After GoF

- “Mind sized” chunks of information (Ward Cunningham)
- Human-to-human communication
- Good solution to a common problem within a specific context
- Expresses intent (the “why” vs. the “how”)
- Observed from actual experience
- NOT:
  - A firm rule –always a time when not to use
  - Copy-paste
  - Isolated. Part of a Pattern Language

Why Revisit Patterns?

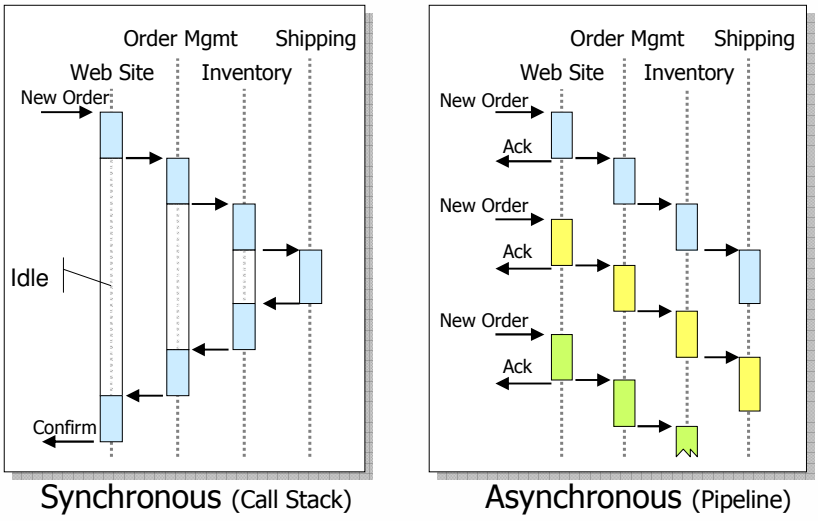
- New programming models bring new patterns.
- Patterns are expressed using the constructs of the underlying architectural style (e.g. OO, SOA).
- Patterns can help discover higher levels of abstraction.
- Ultimately some of these patterns can be implemented in the platform.
- This is usually an iterative process.

Focus on Interaction

- In the OO world interaction is essentially free
- Powerful structural mechanisms: inheritance, composition, aggregation
- In the SOA world more focus shifts to interaction. Structural composition mechanisms are limited.

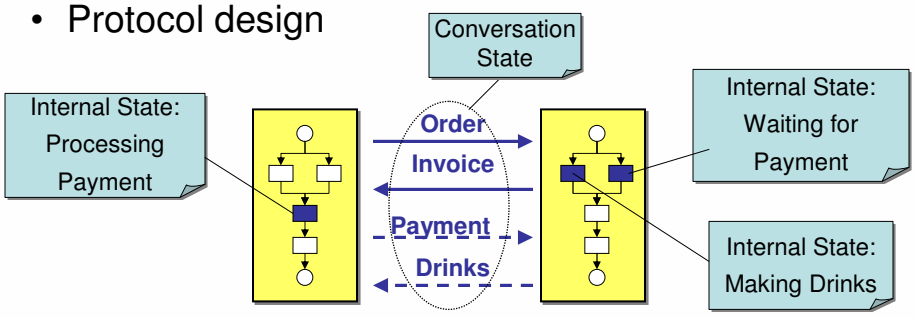
> "The lines are becoming boxes now."
> -- Ralf Westphal

Thinking Asynchronously



Conversations

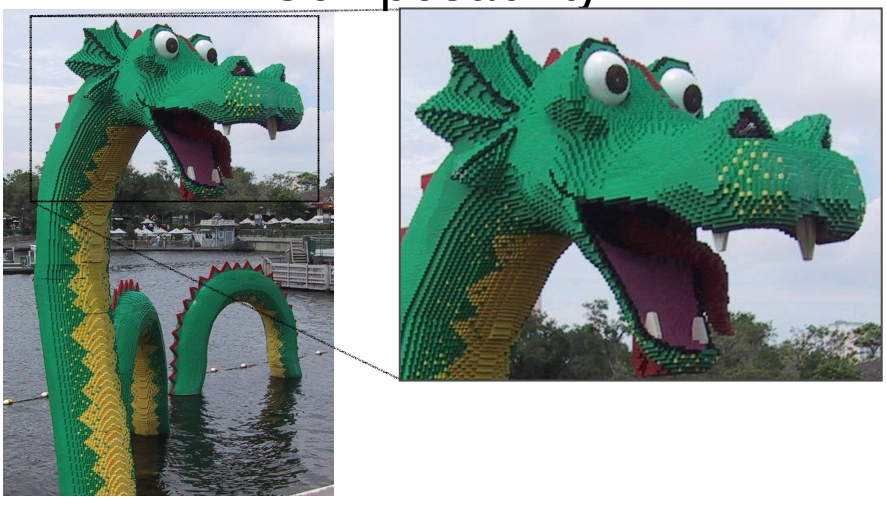
- Series of related messages between parties
- Choreography (e.g. WS-CDL)
- Describing conversation state and rules
- Protocol design



Exception Handling

- "Starbucks does not use two-phase commit"
  - Compensation
  - Retry
  - Write-off
- Throughput over latency
  - "Wider bridges, not faster cars"
- Optimize for happy day scenario

Composability



"The ability to build new things from existing pieces."

Composition Considerations

- Introduces a new layer into the system: the composition layer
- Deserves to be a 1st class citizen:
  - Language
  - Tools
  - Tests

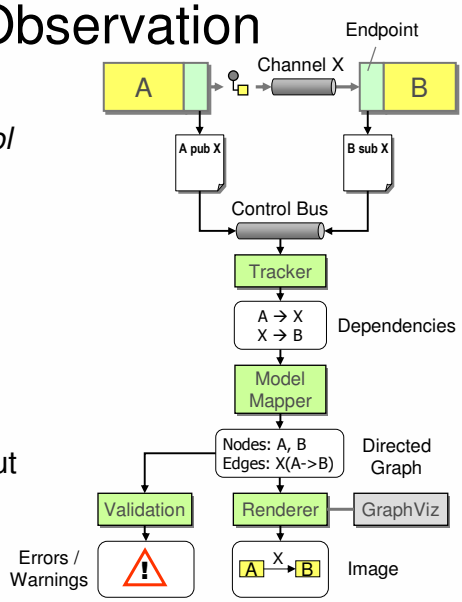
> *"Great composers are few and far in between."*
> -- Gregor's Ramblings

Bottom Up vs. Top Down

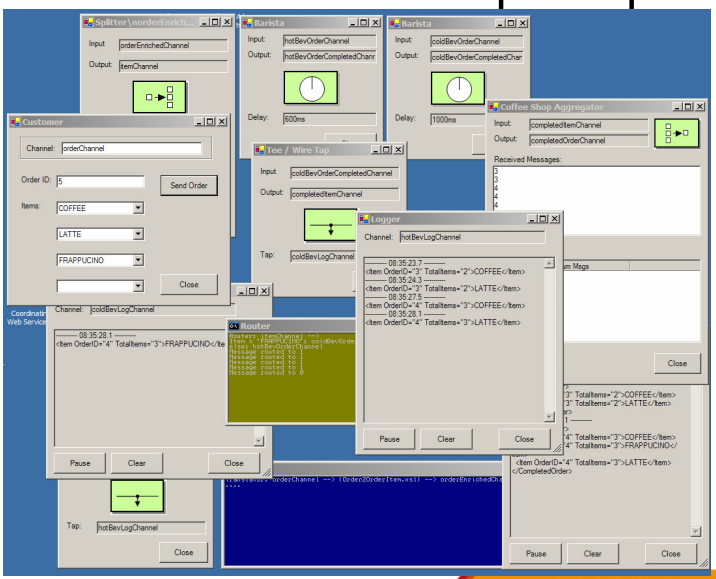
- Loosely coupled systems enable independent variability
- System can evolve locally without breaking
- Evolution can lead to surprises
- Therefore, extract accurate state of the system:
  - Design-time analysis
  - Run-time observation
- "Reverse" MDA

Run-time Observation

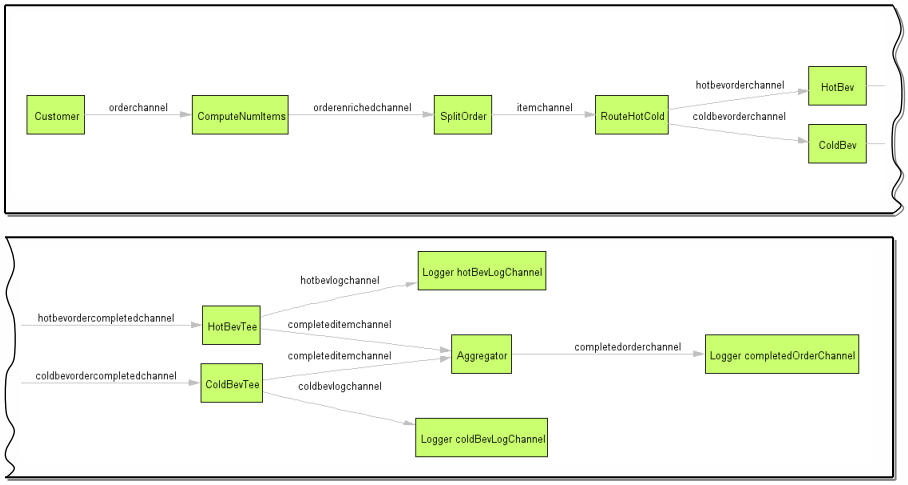
- Component endpoints send status messages to a *Control Bus*
- Invisible to applications
- Central component collects publication and subscription data
- Map onto a Directed Graph metamodel
- Use AT&T GraphViz to layout a visual representation



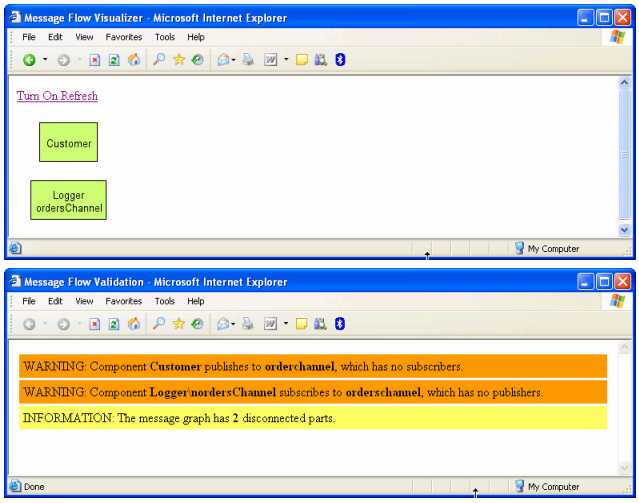
Visualization – Example Input

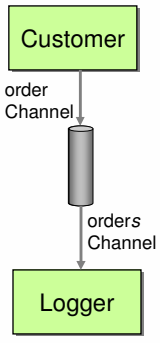


Visualization – Example Output



Model Validation





Domain-Specific Languages

- Finding generic languages to support these programming models is hard
- It also makes the languages complex and the learning curve steep (see XSLT)
- “Language Workbenches” may help us create our smaller domain-specific languages
- Intentional Programming
  - JetBrains Meta Programming System (MPS)
  - Visual Studio 2005
  - See article on http://www.martinfowler.com/

In Summary

- SOA brings new and unfamiliar:
  - Architectural Styles
  - Programming Models
  - Best Practices
  - Patterns
  - Testing Approaches
  - Management Approaches
- The collective learning cycle will take some time
- The vendors and specs are sometimes ahead (or amiss) of the real issues

Enterprise Integration Patterns

- Language of 65 patterns
- Consistent vocabulary and notation
- Focuses on asynchronous messaging
- Many more patterns to harvest:
  - Conversations
  - Orchestrations
  - Error Handling
  - Complex Transformations
  - Rules Engines
- www.EnterpriseIntegrationPatterns.com