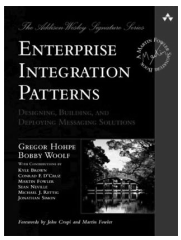
Hooking Stuff Together – Software Development in the Age of the Cloud

Gregor Hohpe
Google

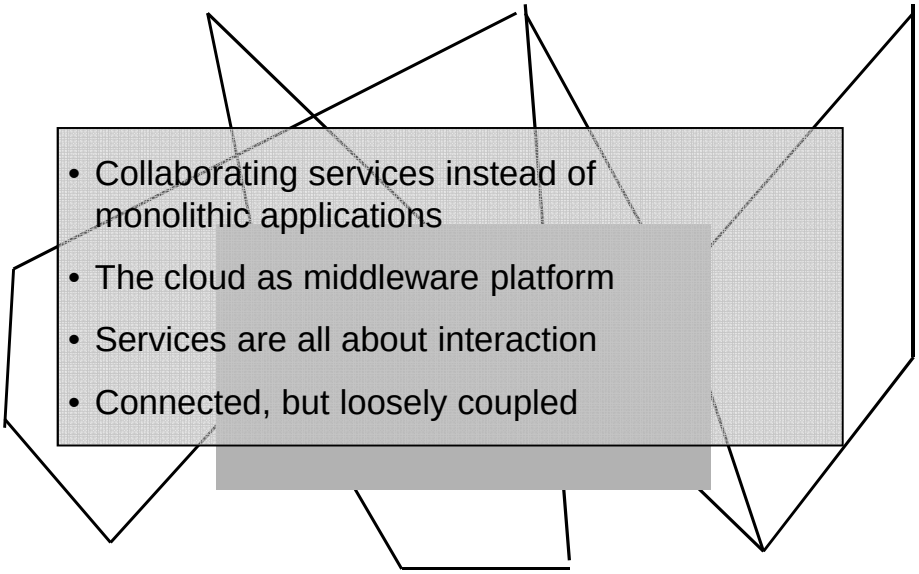
www.eaipatterns.com
www.conversationpatterns.com

Look Who's Talking!

- **Distributed Systems / EAI/ SOA / Messaging**
- **Software engineer at Google**
- **Enterprise Integration Patterns**
  (Addison-Wesley)
- **Enterprise Solution Patterns**
  (Microsoft Patterns & Practices)
- **Integration Patterns**
  (Microsoft Patterns & Practices)
- **SOA Expertenwissen**
  (dpunkt Verlag)
- **www.eaipatterns.com**



~~Yesterday's~~ Today's Software Environment



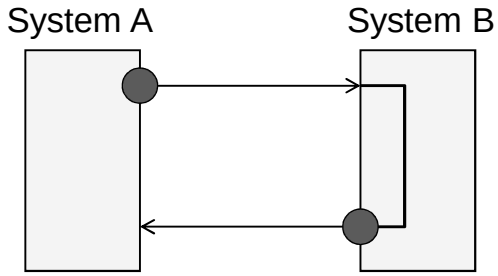
- Collaborating services instead of monolithic applications
- The cloud as middleware platform
- Services are all about interaction
- Connected, but loosely coupled

Less is More?

- **NO Call Stack**
- **NO Transactions**
- **NO Promises**
- **NO Certainty**
- **NO Ordering Constraints**
- **NO Assumptions**

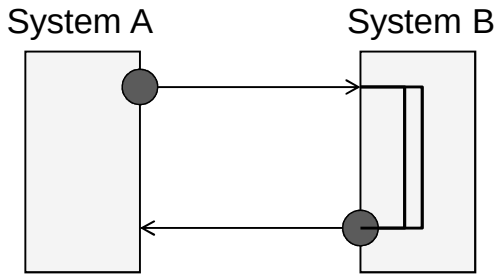
Scary?	Yes!
Cool?	Yes!
Way to go?	Yes!

A Simple Interaction



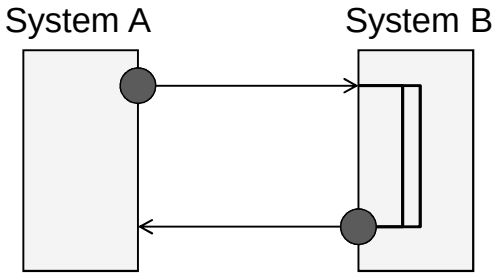
What if the response does not come?

Communication Problems



Lost Request?
Lost Response?
System B Crashed?
Retry?

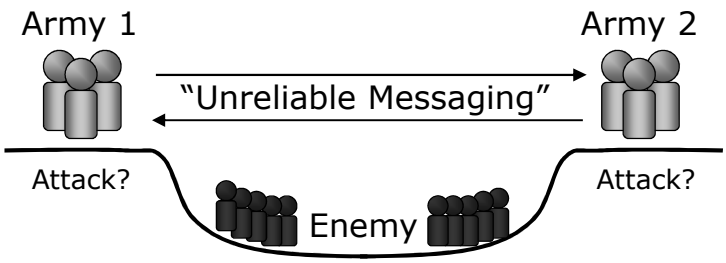
Delayed Response



Executed Once?
Executed Twice?

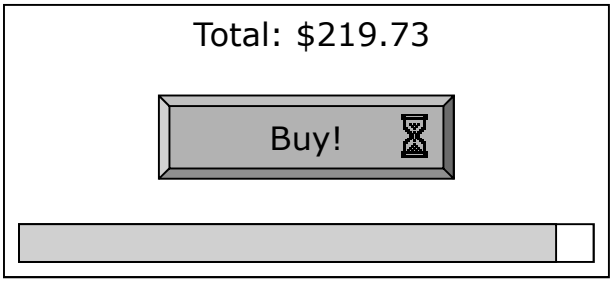
Inherent State Uncertainty

- **System A is never 100% sure what state System B is in**
- **This problem does not occur in a monolithic system**
- **Compare Byzantine General's Problem**



Still An Issue With HTTP

- **Hardware failure**
- **Network failure**
- **Time-outs**
- **Partial response**



8

Isn't This What Distributed Transactions Are For?

- **Require coordinator**
- **Even 2 Phase Commit has windows of uncertainty**
- **Not practical for long running interactions**
  - **Locks not practical / economical**
  - **Isolation not possible / practical**
- **Usually not supported**
- **Don't scale**

"Life Beyond Distributed Translations – an Apostate's Opinion"
--Pat Helland

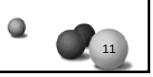
9

You Got My Attention. So Now What?

- **Live with the uncertainty**
- **Keep it as simple as possible**
- **Interaction steps in the foreground**
- **Asynchronous messaging rules**
- **Think differently – new programming models**
- **Run-time behavior is a big part of the game**

Living with Uncertainty: Idempotency

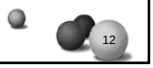
- **“F5 Strategy” – If it does not work, try again**
- **“At-least-once Delivery” a reality in loosely coupled systems**
- **Requires some state, e.g. “Which messages did I see before?”**



Living With Uncertainty: Failure Handling Strategies

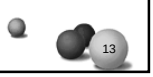
- **Compensation**
  - **“Execute” / “Compensate”**
  - **Everyone executes optimistically**
  - **Explicit back out strategy if errors occur**
- **Reservation / Tentative operation**
  - **“Try” / “Confirm” / “Cancel”**
  - **All activities are tentative; outcome is promised in good faith**
- **Write-Off**

“Starbucks does not use two-phase commit”



Keep it as simple as possible

- **Even simple things become complicated in a distributed environment**
- **If it looks complicated on paper it’s likely to be impossible in practice**
- **If you can’t understand it, other developers likely won’t either**
- **A well understood failure scenario can be better than an incomprehensible and unproven “failsafe” system**

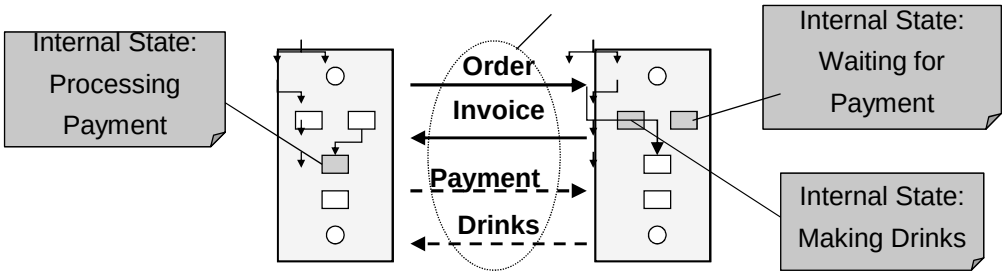


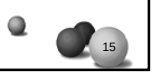
Focus on Interaction

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

Interactions: Conversations

- **Series of related messages between parties**
- **Not handled at lower layer**
- **Endpoints keep some conversation state**
- **Protocol design**

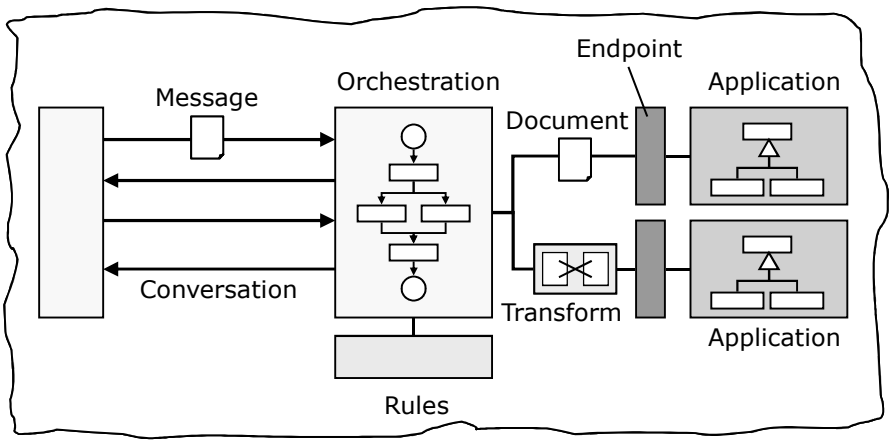




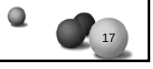
Asynchronous Messaging

- **Exchange through messages, not RPC**
- **Waiting for the results of an HTTP request is not a smart use of a 3 GHz processor**
- **Request and response message typically handled by different parts of your program, even if the same TCP connection**
- **Reduced assumptions about timing and state**

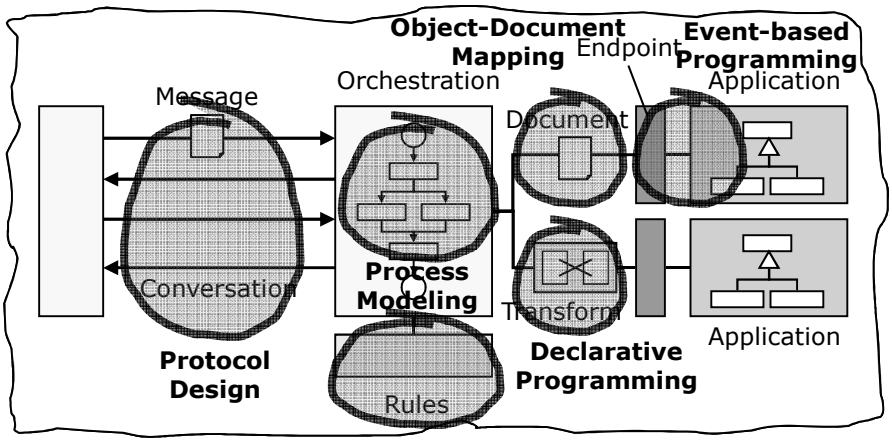
Think differently – New Programming Models



"Architect's Dream"



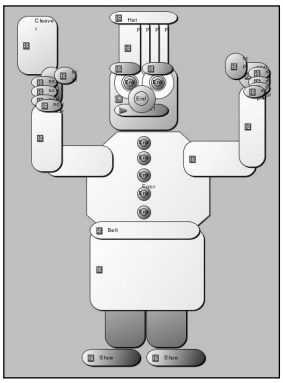
Think differently – New Programming Models



"Developer's Nightmare"

"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**



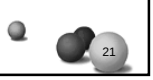
"EAI Art"

Patterns – 10 Years After GoF

- **New programming models bring new patterns.**
- **“Mind sized” chunks of information (Ward Cunningham)**
- **Human-to-human communication**
- **Expresses intent (the “why” vs. the “how”)**
- **Makes assumptions explicit**
- **Observed from actual experience**

Run-time Behavior is a Big Part of the Game

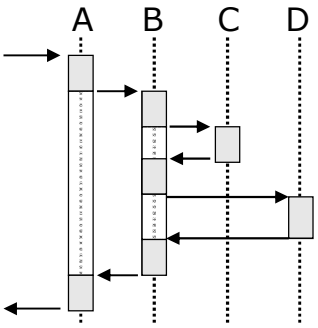
- **Some programming abstractions are great, e.g. MapReduce**
- **In a single-threaded call-stack machine, programming model and execution model match fairly closely**
- **In a highly distributed dynamic system, they are very different!**
- **Monitoring, run-time analysis, and visualization critically important**



Run-time Behavior is a Big Part of the Game

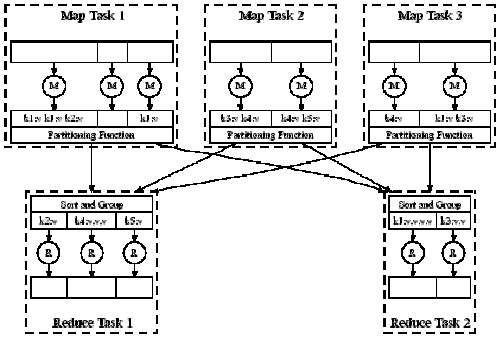
- **Call Stack**

```
void a() {
  b();
}

Void b() {
  c();
  d();
}
```



- **MapReduce**

```
map(in_key, data)
 → list(key, value)

reduce(key, list(values))
→ list(out_data)
```

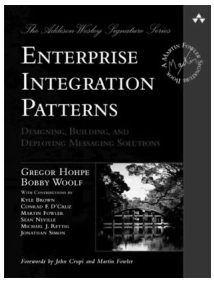


22

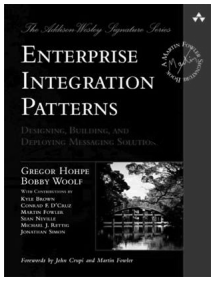
My Work

- **Messaging Patterns (65)**
  - **Messaging Systems**
  - **Messaging Channels**
  - **Message Construction**
  - **Message Routing**
  - **Message Transformation**
  - **Messaging Endpoints**
  - **System Management**
- **Conversation Patterns**
  - **Discovery**
  - **Establishing a Conversation**
  - **Multi-party Conversations**
  - **Reaching agreement**
  - **Resource Management**
  - **Error Handling**

www.eaipatterns.com

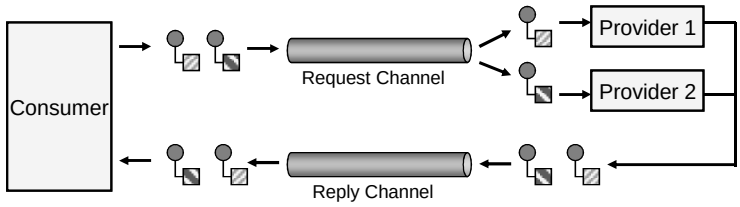


www.conversationpatterns.com



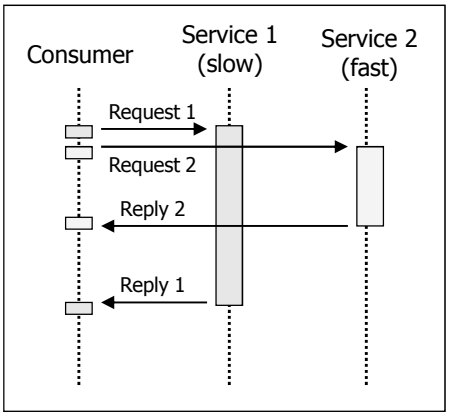
23

Multiple Service Providers



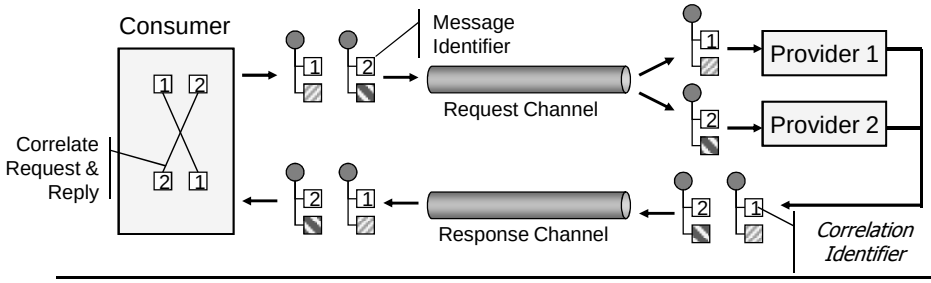
- **Request message can be consumed by more than one service provider**
- ***Point-to-Point Channel* supports *Competing Consumers*, only one service receives each request message**
- **Channel queues up pending requests**

Multiple Service Providers



- **Reply messages get out of sequence**
- **How to match request and reply messages?**
  - **Only send one request at a time → very inefficient**
  - **Rely on natural order → bad assumption**

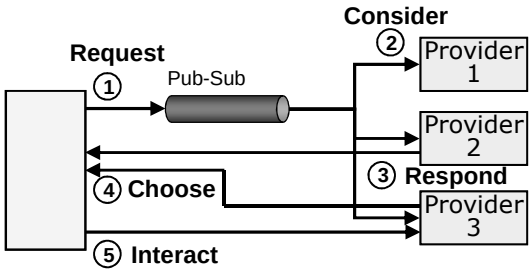
Pattern: *Correlation Identifier*



- **Equip each message with a unique identifier**
  - **Message ID (simple, but has limitations)**
  - **GUID (Globally Unique ID)**
  - **Business key (e.g. Order ID)**
- **Provider copies the ID to the reply message**
- **Consumer can match request and response**

26

Conversation Pattern: *Dynamic Discovery*



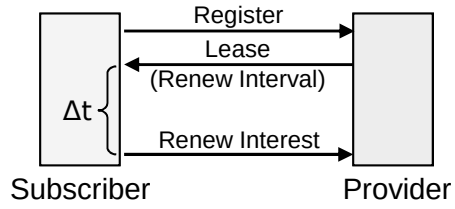
1. **Broadcast request**
2. **Provider(s) consider whether to respond (load, suitability)**
3. **Interested providers send responses**
4. **Requestor chooses "best" provider from responses**
5. **Requestor initiates interaction with chosen provider**

- **Examples: DHCP, TIBCO Repository discovery**

27

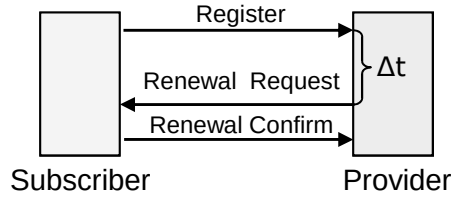
Conversation Pattern: *Renewing Interest*

Automatic Expiration



- **"Lease" model**
- **Heartbeat / keep-alive**
- **Subscriber has to renew actively**
- **Example: Jini**

Renewal Request



- **"Magazine Model"**
- **Subscriber can be simple**
- **Provider has to manage state for each subscriber**

Fin