

# ADVANCED DISTRIBUTED SYSTEMS DESIGN WITH SERVICE ORIENTED ARCHITECTURE

---

Udi Dahan – The Software Simplist  
Enterprise Development Expert & SOA Specialist

#ADSDcourse

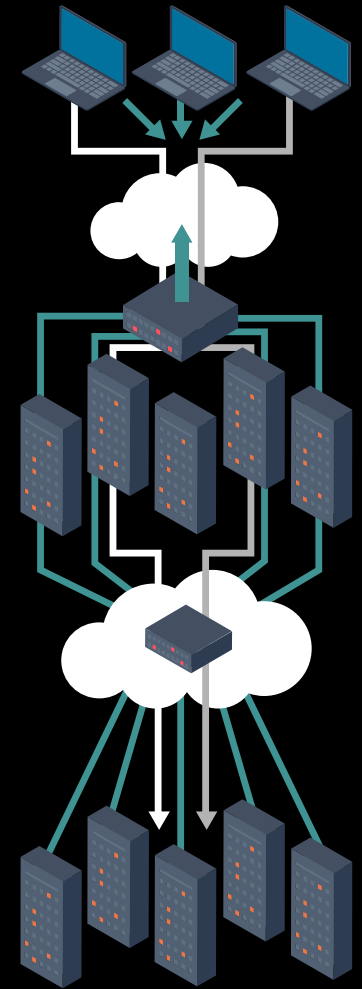
[www.UdiDahan.com](http://www.UdiDahan.com)



# DISTRIBUTED SYSTEMS THEORY

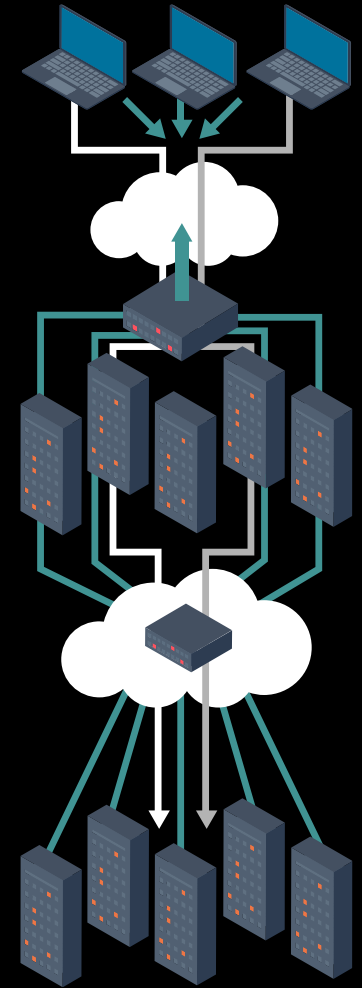
# SYSTEMS ARE NOT APPLICATIONS

- An application has a single executable and runs on a single machine
- Usually has a single source of information
- Applications don't know about "connectivity"



# SYSTEMS ARE NOT APPLICATIONS

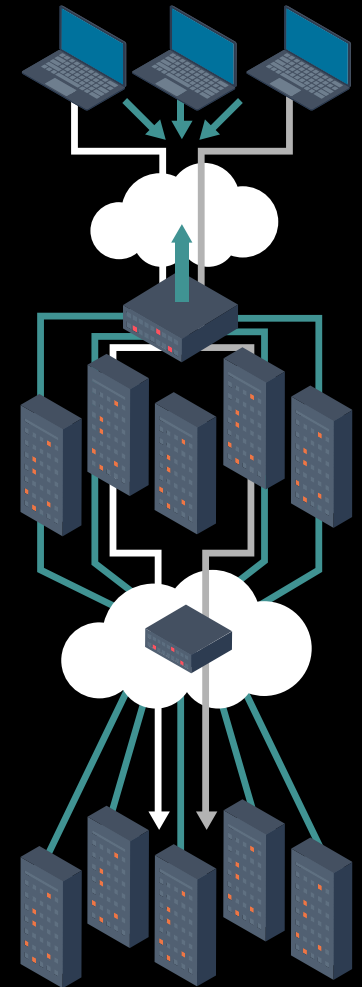
- A system can be made up of multiple executable elements on multiple machines
- Usually has multiple sources of information
- System must deal with “connectivity”





# SYSTEMS ARE NOT APPLICATIONS

- Each executable within a system is not an application
- Each executable must deal with “connectivity”



# "CONNECTIVITY" – THE NETWORK MATTERS

Common assumptions made by developers and architects in distributed systems

- The network is reliable
- Latency isn't a problem
- Bandwidth isn't a problem
- The network is secure
- The topology won't change
- The administrator will know what to do
- Transport cost isn't a problem
- The network is homogeneous




*Deutsch 94  
Gosling 97*

# "CONNECTIVITY" – THE NETWORK MATTERS

## "The 8 fallacies of distributed computing"

1. The network is reliable
2. Latency isn't a problem
3. Bandwidth isn't a problem
4. The network is secure
5. The topology won't change
6. The administrator will know what to do
7. Transport cost isn't a problem
8. The network is homogeneous



*Deutsch 94  
Gosling 97*

# 3 MORE FALLACIES

1. The network is reliable
2. Latency isn't a problem
3. Bandwidth isn't a problem
4. The network is secure
5. The topology won't change
6. The administrator will know what to do
7. Transport cost isn't a problem
8. The network is homogeneous
9. The system is atomic/monolithic
10. The system is finished
11. Business logic can and should be centralized



*Neward 06*

# #1. THE NETWORK IS RELIABLE

- Hardware, software, security can cause issues

```
var svc = new MyService();  
var result = svc.Process(data);
```

- How do you handle `HttpTimeoutException`?  
Data can get lost when sent over the wire

# #1. THE NETWORK IS RELIABLE

- Solutions:

Retry & Ack / Store & Forward / Transactions

Don't roll your own – too many edge cases

Use reliable messaging infrastructure

MSMQ / Sql Server 2005 Service Broker

- But doesn't provide a request/response synchronous method-centric model

## #2. LATENCY ISN'T A PROBLEM (==0)

- Time to cross the network in one direction
- Small for a LAN, WAN & internet can be large  
Many times slower than in-memory access
- Bad-old days of OO – remote objects  
Even accessing a property was a round-trip  
Now we use DTO's
- But what about lazy-loading with an ORM?

EVENT	LATENCY	SCALED
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-State disk I/O (flash memory)	50-150 $\mu$ s	2-6 days
Rational disk I/O	1-10 ms	1-12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: SanFrancisco to United Kingdom	81 ms	8 years
Internet: SanFrancisco to Australia	183 ms	19 years
TPC packet retransmit	1-3 s	105-317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization syatem reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia



## #2. LATENCY ISN'T A PROBLEM (==0)

- Solutions:

Don't cross the network if you don't have to  
Inter-object chit-chat shouldn't cross the network

If you have to cross the network,  
take all the data you might need with you

### #3. BANDWIDTH ISN'T A PROBLEM ( $\infty$ )

- Although bandwidth keeps growing, the amount of data grows faster
- When transferring lots of data in a given period of time, network congestion may interfere
- ORMs eagerly fetching too much data

# #3. BANDWIDTH ISN'T A PROBLEM ( $\infty$ )

- Solution:

Move time-critical data to separate networks

Can't eagerly fetch everything / can't lazy load everything

Might need to have more than one domain model to resolve forces of bandwidth and latency

# #4. NETWORK IS SECURE

- Unless you're on a separate network that will never, ever be connected to anything else...
- Well, not even then. Viruses, Trojans, etc can still be brought in by users on CDs, DVDs, DOKs, etc
- You can't be 100% safe from everything

# #4. NETWORK IS SECURE

- Solution:

Perform a threat model analysis

Balance costs against risks

Most importantly, talk about it. Include PR and legal.

## #5. THE TOPOLOGY WON'T CHANGE

- Unless a server goes down and is replaced
- Or is moved to a different subnet
- Or clients wirelessly connect and disconnect

Issues with WCF callback contracts

- What will happen to the system when those hard coded / config-file values change?

# #5. THE TOPOLOGY WON'T CHANGE

- Solution:

Don't hard-code addresses

Consider using resilient protocols (multicast)

Discovery mechanisms are cool, but hard to get right

- Will your system be able to maintain response-time requirements when this happens?

## #6. THE ADMIN WILL KNOW WHAT TO DO

- Possible in small networks  
Until they get ~~run over by a truck~~ promoted.  
Their replacement probably won't know what to do.
- If there are multiple admins, rolling out various upgrades and patches, will everything grind to a halt?  
Will client software be able to work with a new version of the server?
- High Availability while upgrading?



## #6. THE ADMIN WILL KNOW WHAT TO DO

- Solution:

Consider how to pinpoint problems in production

Some logging is helpful, too much can be harmful

Consider multiple versions running in parallel

Although backwards compatibility is hard

Enable the admin to take parts of the system down  
for maintenance without adversely affecting the rest

Queuing technology helps

## #7. TRANSPORT COST ISN'T A PROBLEM

- Serialization before crossing the network (and deserialization on the other side) takes time.  
In the cloud, it can be a big cost factor
- The hardware network infrastructure has upfront and ongoing costs.

# #7. TRANSPORT COST ISN'T A PROBLEM

- Solution:

The effect of serialization on performance further strengthen the argument to stay away from chatting over the network

Architects need to make trade-offs between infrastructure costs and development costs – upfront vs. ongoing.

## #8. THE NETWORK IS HOMOGENEOUS

- It used to be easier - .NET/Java interop works
- Now we've got Ruby, NoSQL, and stuff people hacked together over http (a.k.a REST)
- Semantic interoperability will always be hard, budget for it

# #9. THE SYSTEM IS ATOMIC

- Maintenance is hard in “big balls of mud”  
Changing one part of the system affects other parts
- Integration through the DB creates coupling  
It gets worse with XML in the DB
- If the system wasn't designed to scale out to multiple machines, doing so may actually hurt performance

# #9. THE SYSTEM IS ATOMIC

- Solution:

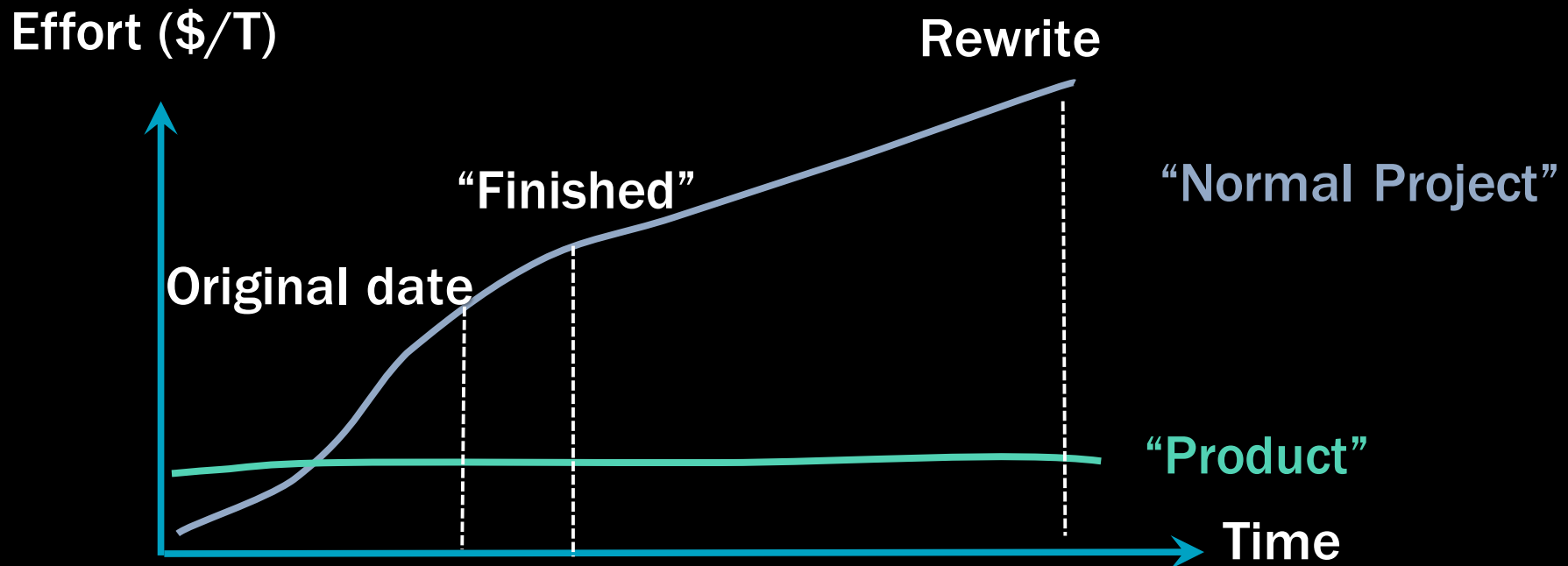
Internal loose coupling

Modularize

Design for scale out in advance, or you just may end up being stuck with scale up.

# #10. THE SYSTEM IS FINISHED

- Maintenance costs over the lifetime of a system are greater than its development costs



- The system is never "finished"

# #10. THE SYSTEM IS FINISHED

- Solution:

There's no such thing as a "maintenance programmer"

Projects are a poor model for software development  
Long-lived products are better

Beware the rewrite that will solve everything



# A BETTER DEVELOPMENT PROCESS

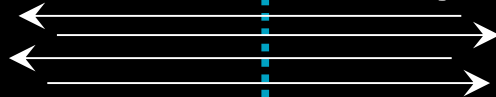
The Business

IT

~~Requirements~~

workarounds

Rapid Prototyping



So you wish X could be better/cheaper/faster

Exactly!

(still not a requirement)

Business Analyst

# A BETTER DEVELOPMENT PROCESS

The Business

IT

Estimate

Architect

Too broad a range

Let me do a POC for T

Go ahead with POC

New Estimate

Estimate accepted

Now it's a requirement

Estimate:

Given a well-formed team of size  $S$   
that is not working on anything else  
I'm  $C\%$  confident work will take between  $T1$  &  $T2$

# A BETTER DEVELOPMENT PROCESS

The Business

IT

So it'll be ready in T2?

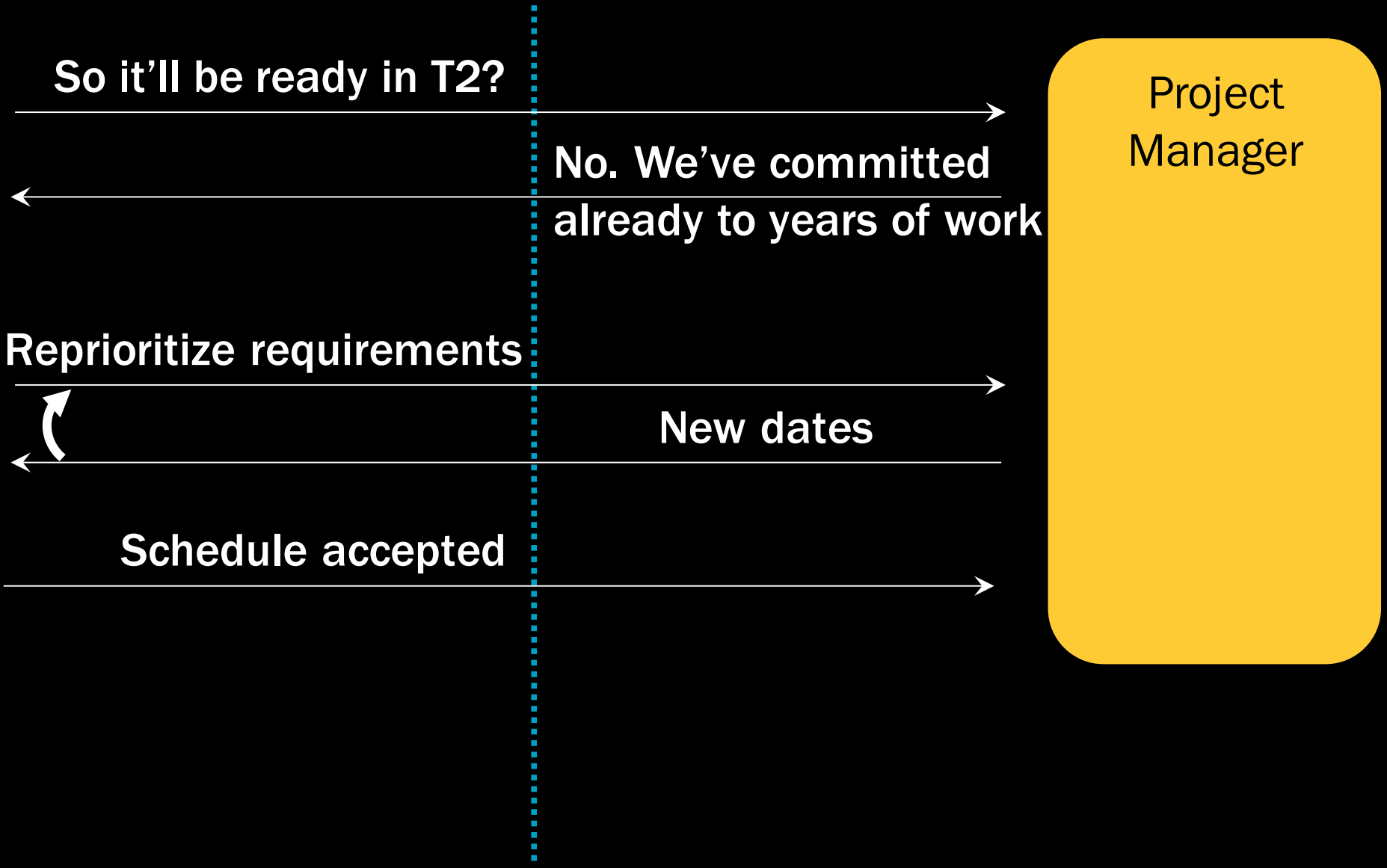
No. We've committed  
already to years of work

Project  
Manager

Reprioritize requirements

New dates

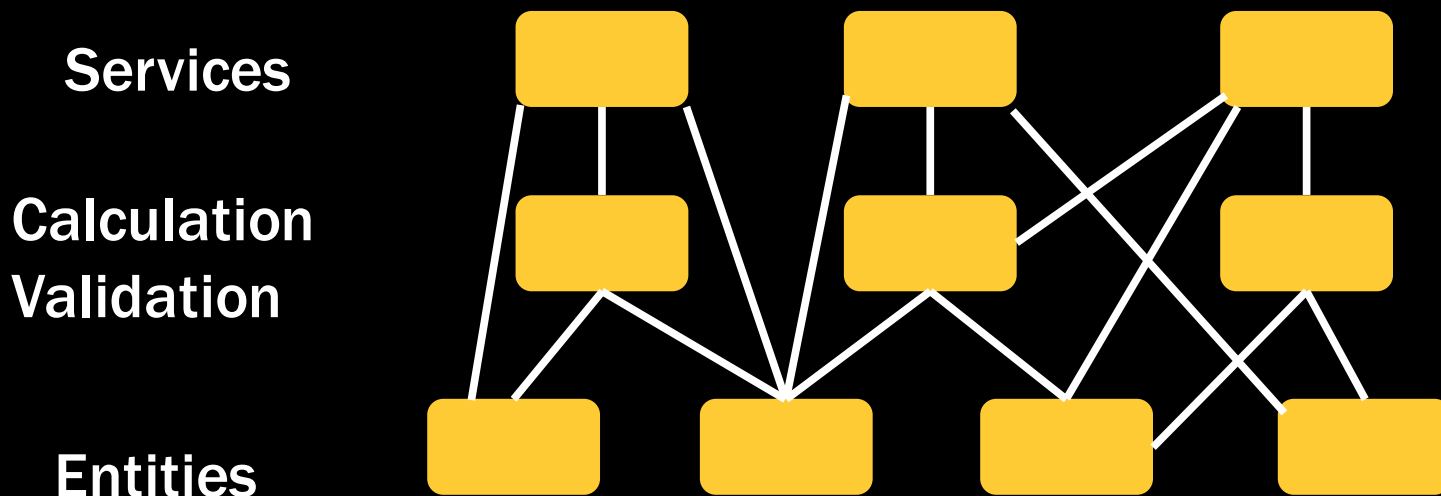
Schedule accepted



# #11. BUSINESS LOGIC CAN AND SHOULD BE CENTRALIZED

- “First name must be less than 40 characters”  
Enforce in the UI? BL? DB? Everywhere?

What about when the business rules change?



# #11. BUSINESS LOGIC CAN AND SHOULD BE CENTRALIZED

- Solution:

Logic will be physically distributed

Can still “centralize” in the development view

[more reading] 4+1 views of software architecture

“Tag” source code by feature implemented

Enables finding all code by feature

Even if its in multiple files

# SUMMARY

- Best practices have yet to catch up to “best thinking”
- Technology cannot solve all problems
- Adding hardware doesn't necessarily help

# COUPLING IN DISTRIBUTED SYSTEMS

# WHAT IS COUPLING?

- A measure of dependencies
- If X depends on Y,  
there is coupling between them
- 2 kinds of coupling: Afferent (Ca), Efferent (Ce)



# WHAT IS COUPLING?

- Afferent coupling ( $C_a$ ) – who depends on you  
Incoming coupling
- Efferent coupling ( $C_e$ ) – on who you depend  
Outgoing coupling

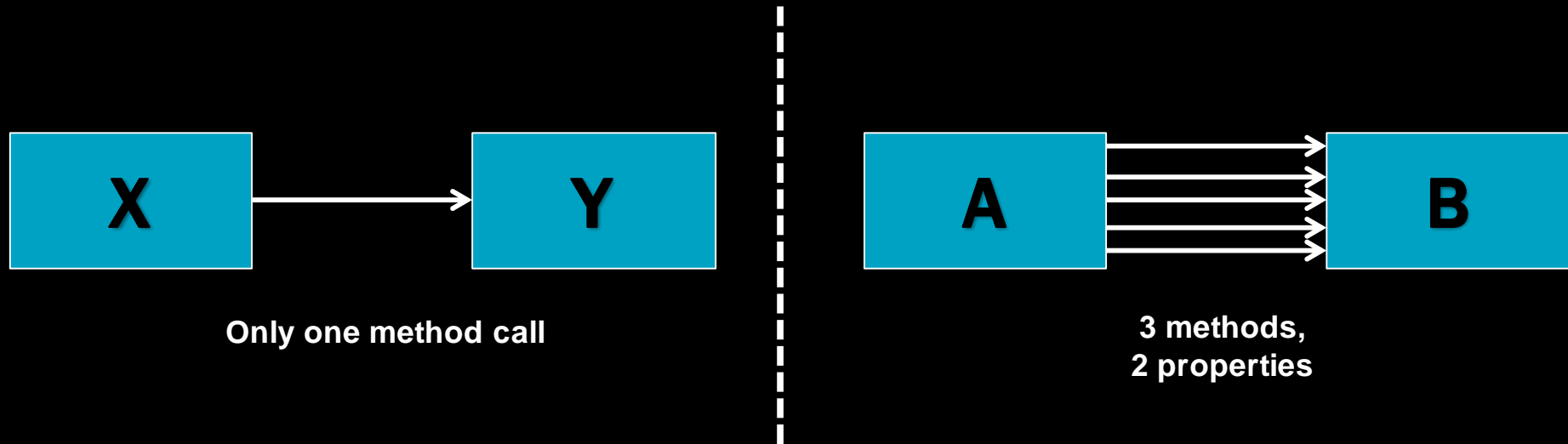
# WHAT IS COUPLING?

- If X depends on Y then:
- X is efferently coupled to Y
- Y is afferently coupled to X

# COUPLING – WHICH KIND IS WORSE?

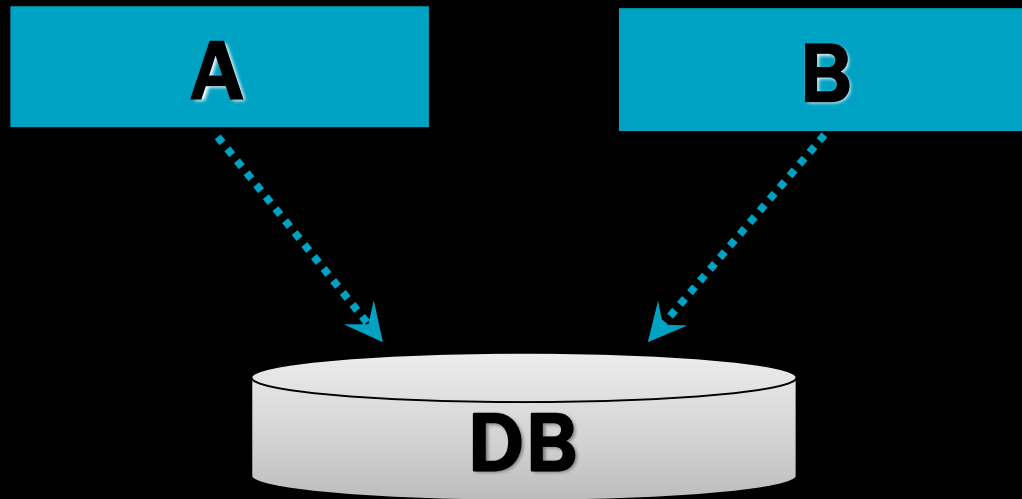
	Afferent (Incoming)	Efferent (Outgoing)
A	5	0
B	0	5
C	2	2
D	0	0

# HOW TO COUNT COUPLING?



**Same amount of coupling?  
Different?**

# BEWARE SHARED RESOURCES



**They hide the coupling that  
otherwise would be visible**

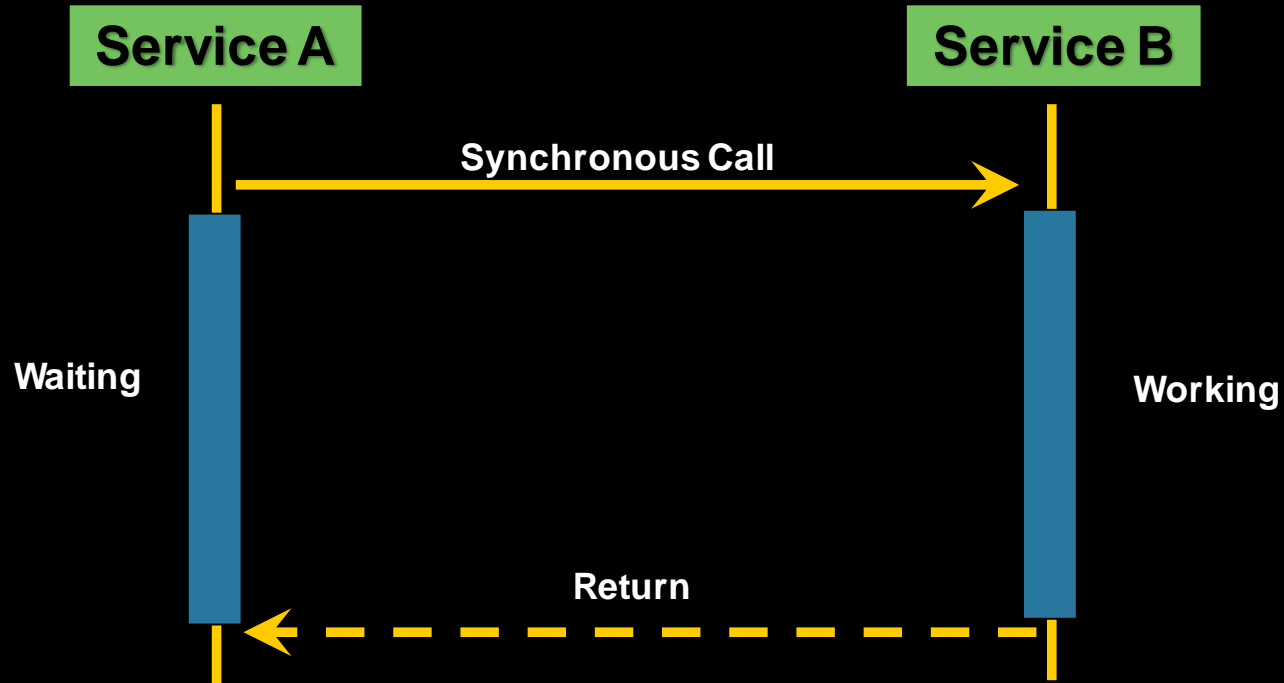
# LOOSE COUPLING AT THE SYSTEMS LEVEL

- Minimize afferent and efferent coupling  
But not mechanically
- Zero coupling isn't really possible
- 3 Different aspects of coupling for systems:  
Platform  
Temporal  
Spatial

# COUPLING ASPECT #1: PLATFORM

- Also known as “Interoperability”
- Using protocols only available on one platform  
Remoting, Binary Serialization, etc
- One of the 4 Tenets of Service Orientation:  
“Share contract and schema, not class or type”

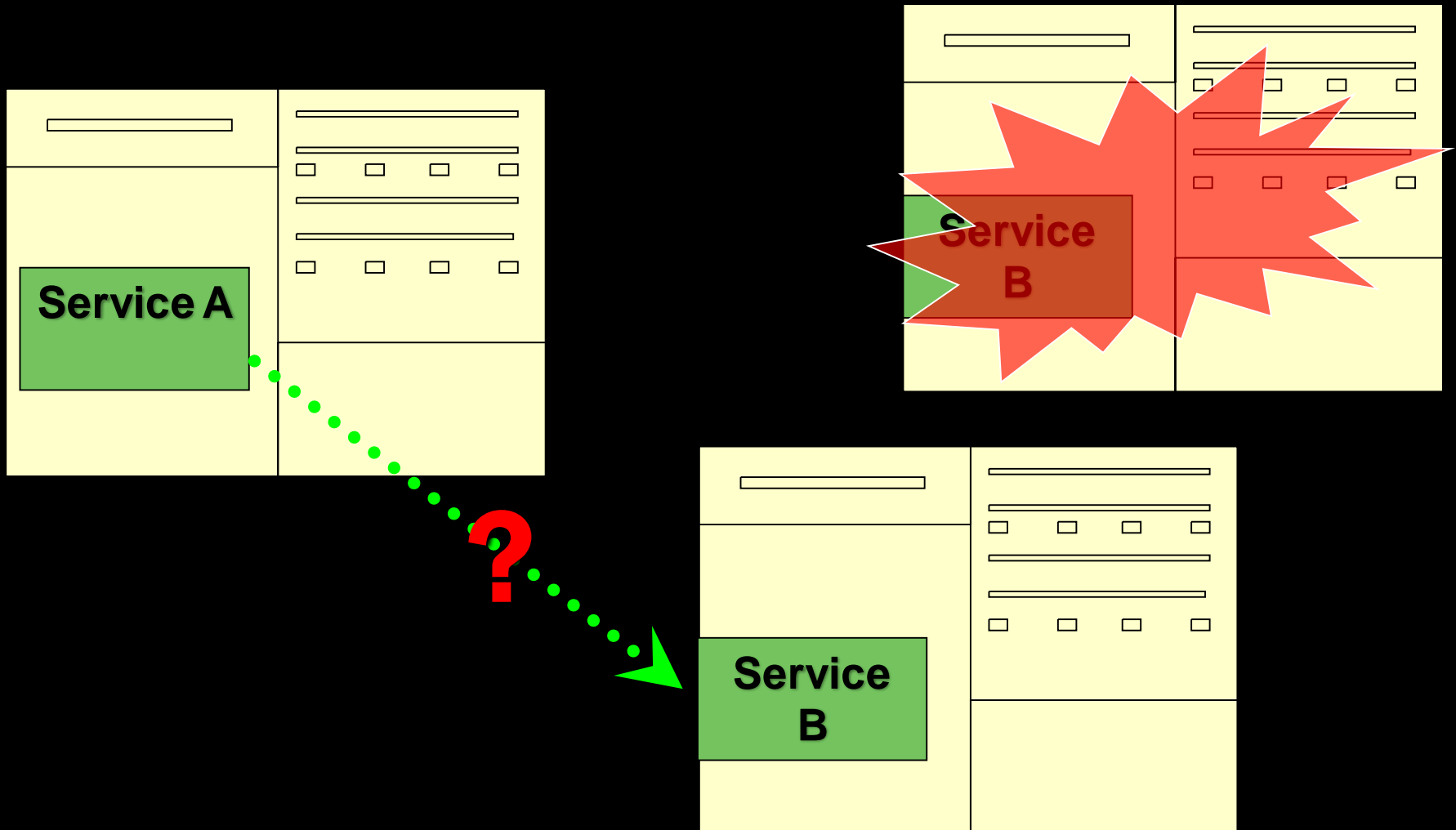
# COUPLING ASPECT #2: TEMPORAL



Processing time of Service B affects that of A



# COUPLING ASPECT #3: SPATIAL



Can communication automatically continue?

# COUPLING ASPECTS: SOLUTIONS

# COUPLING ASPECT #1: PLATFORM

- Many options possible for interoperability.  
Text-based representation on the wire (XML/JSON)  
With or without schema

Use standards based transfer protocol like http  
Or SMTP, UDP, etc

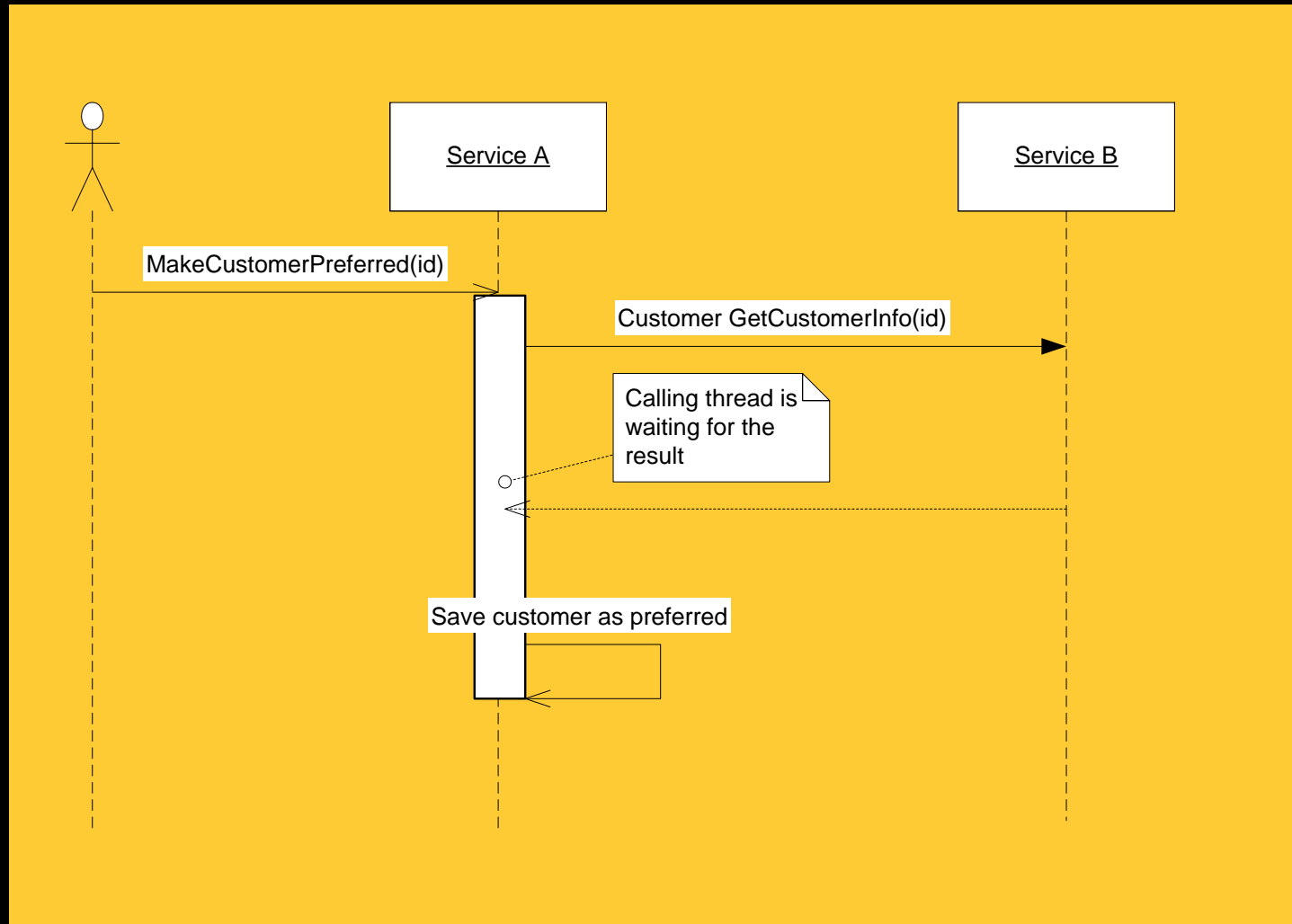
SOAP / WSDL / REST

# ADDITIONAL PLATFORM SOLUTIONS

- Running Java code in-process on the CLR
- Running .Net code in-process on the JVM

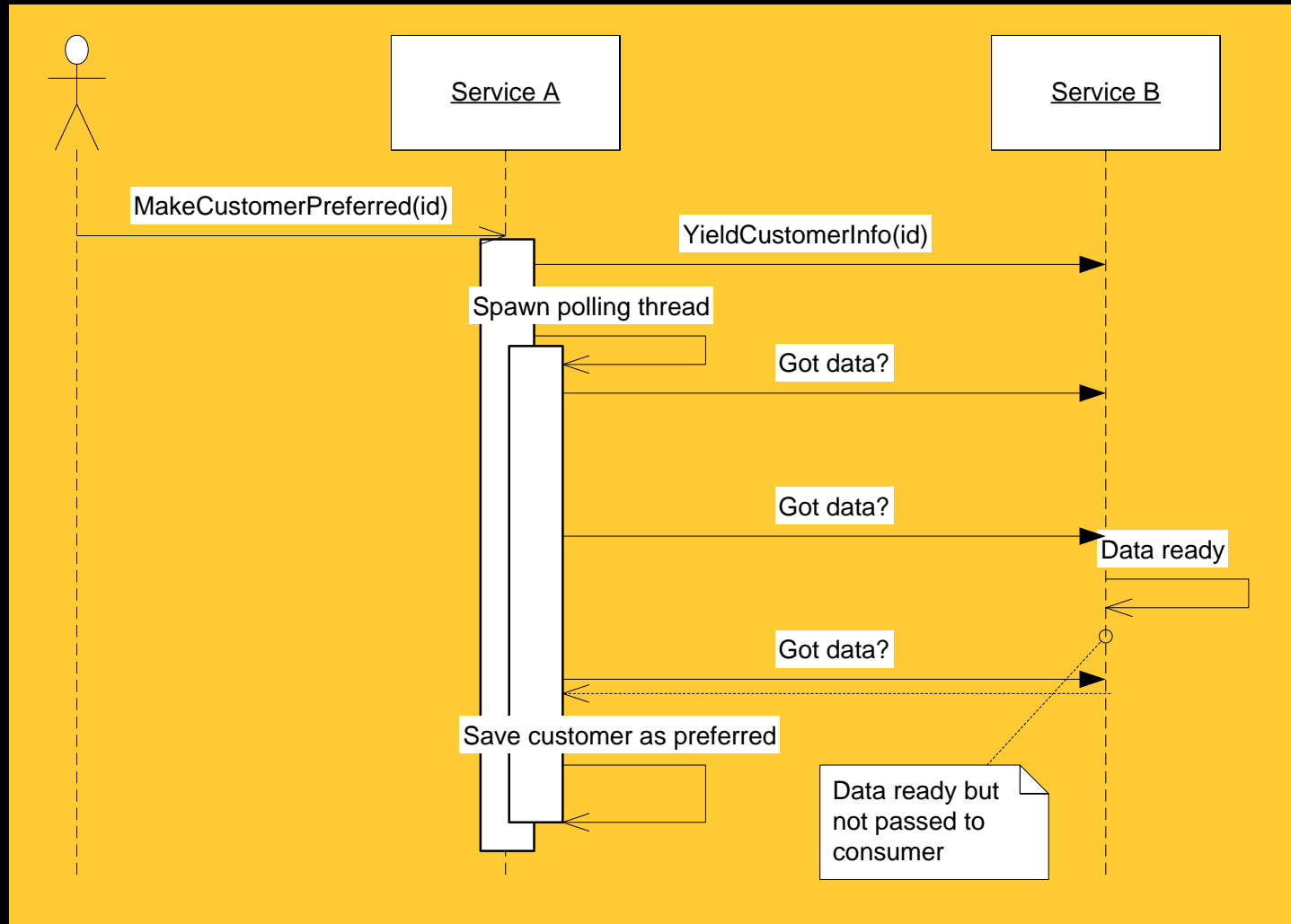


# COUPLING ASPECT #2: TEMPORAL - 1



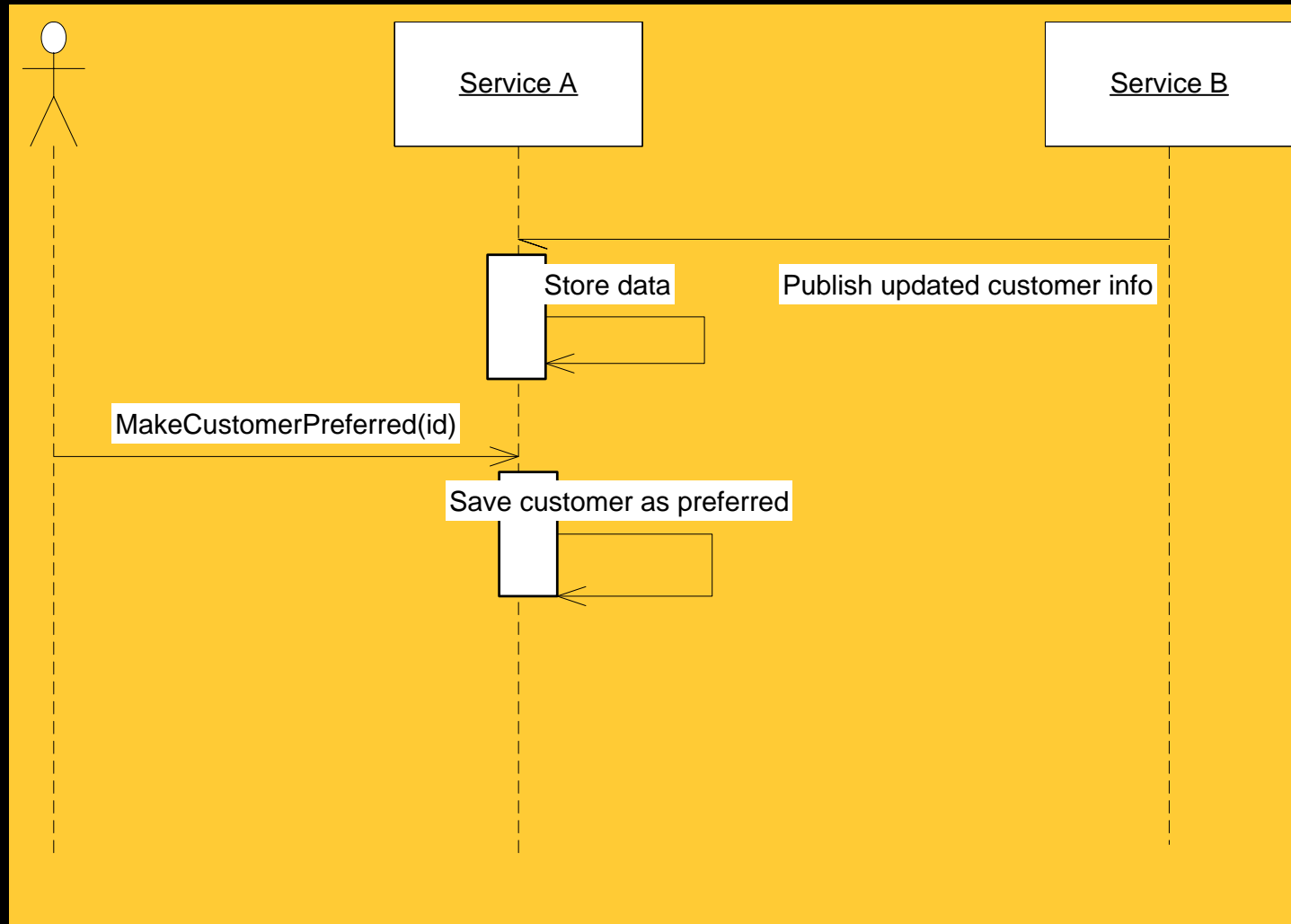
Resources are held while waiting

# COUPLING ASPECT #2: TEMPORAL - 2



Resources are held while waiting. Increased load on service B per consumer (impacted by polling interval)

# COUPLING ASPECT #2: TEMPORAL - FINAL



Good. By separating (in time) the inter-service communication and the request handling

# PUB/SUB TEMPORAL CONSTRAINTS

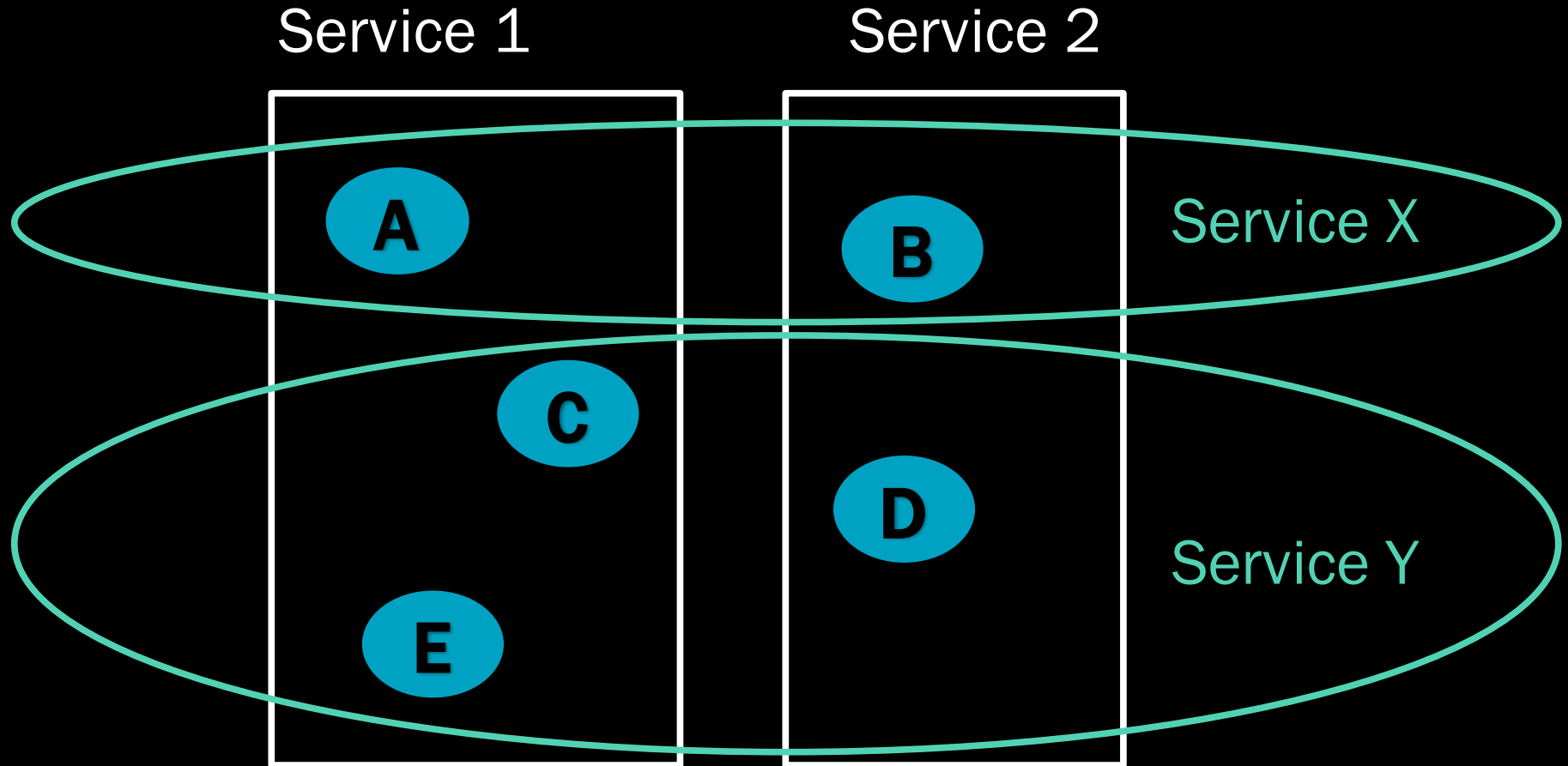
- Subscriber must be able to make decisions based on somewhat stale data
- Requires a strong division of responsibility between publishers and subscribers
- Only one logical publisher should be able to publish a given kind of event



# HOW TO DESIGN EVENTS

- Avoid requests/commands  
Bad: "SaveCustomerRequested"
- State something that happened (past tense)  
Subscribers shouldn't be able to invalidate this  
Good: "OrderAccepted"
- If you have to talk about data, state its validity  
ProductPriceUpdated { Price: \$5, ValidTo: 1/1/15 }

# WHERE (AND WHY) NOT TO DO PUB/SUB



when business requirements demand consistency

# COUPLING ASPECT #3: SPATIAL

- Application level code should not need to know where cooperating services are on the network
- Delegate communications to lower layer – the service agent pattern  
`myAgent.Send(message);`
- How does the agent know which destination to send the message to?

# LOAD BALANCING

- Clients talking to servers through a load balancer don't know which physical server is handling the request...
- ... as long as logically the server CAN handle the request
- Routing is first logical, and second physical

# COUPLING ASPECT #3: SPATIAL

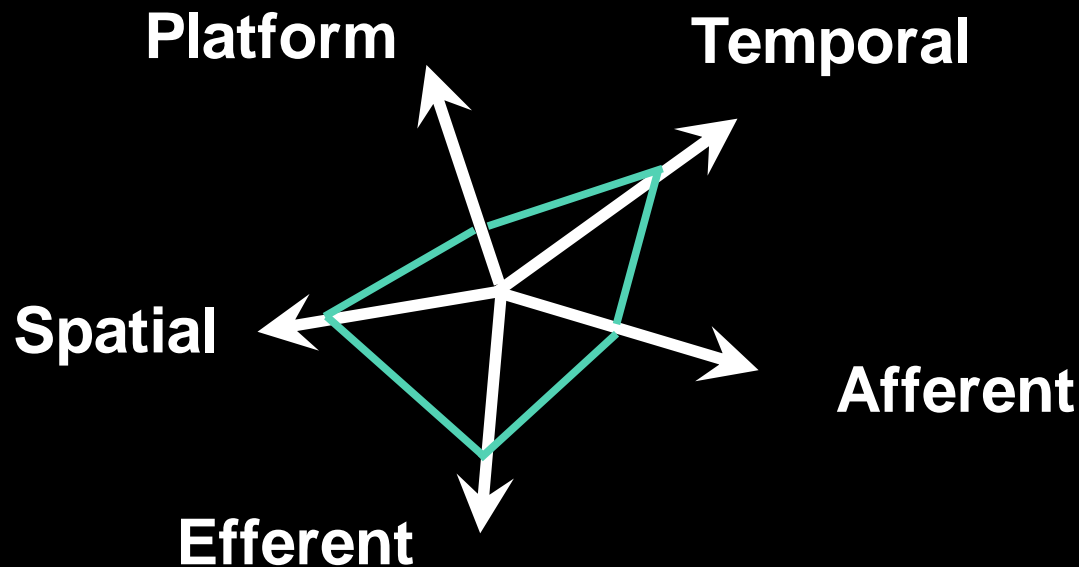
- But if the application code doesn't tell the agent which *logical* destination to send the message to, how would the agent know?
- If there was a direct mapping from message type to logical destination, then specifying the type of message being sent/published would be enough

# MESSAGE TYPE = LOGICAL DESTINATION

- AddCustomerMessage:  
Sent by clients to one logical server  
Multiple physical servers behind a load balancer
- OrderCancelledEventMessage:  
Published by one logical server  
Multiple physical servers can publish the same
- Strongly-typed messages simplify routing  
vs document-centric messaging

# SUMMARY

- Loose coupling is more than just a slogan
- Coupling is a function of 5 different dimensions



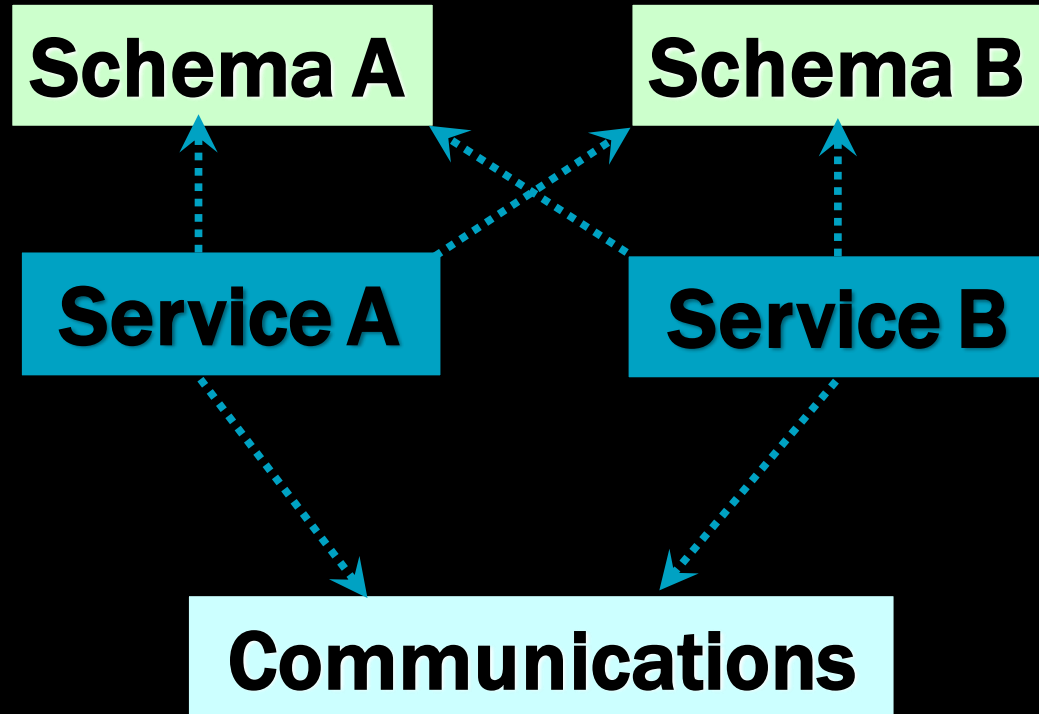
# MESSAGING PATTERNS



# WHY MESSAGING?

- Reduces afferent and efferent coupling while increasing autonomy
- Reduces coupling
  - Use JSON/XML + AMQP for platform coupling
  - Use asynchronous messaging for temporal coupling

# MESSAGING, COUPLING, & AUTONOMY



Service A and B don't directly depend on each other

# ASYNCHRONOUS MESSAGING

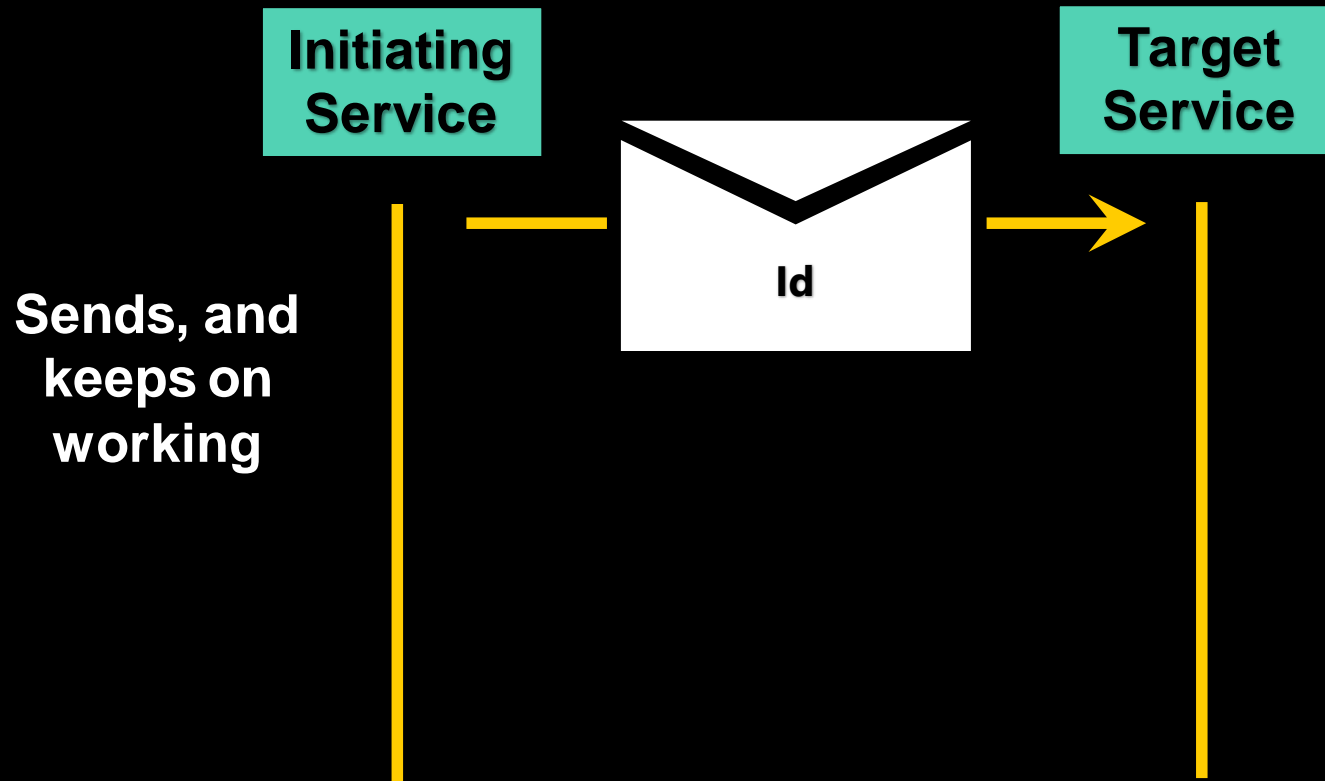
- It's all about one-way, fire & forget messages
- Everything is built on top of it

Return Address pattern

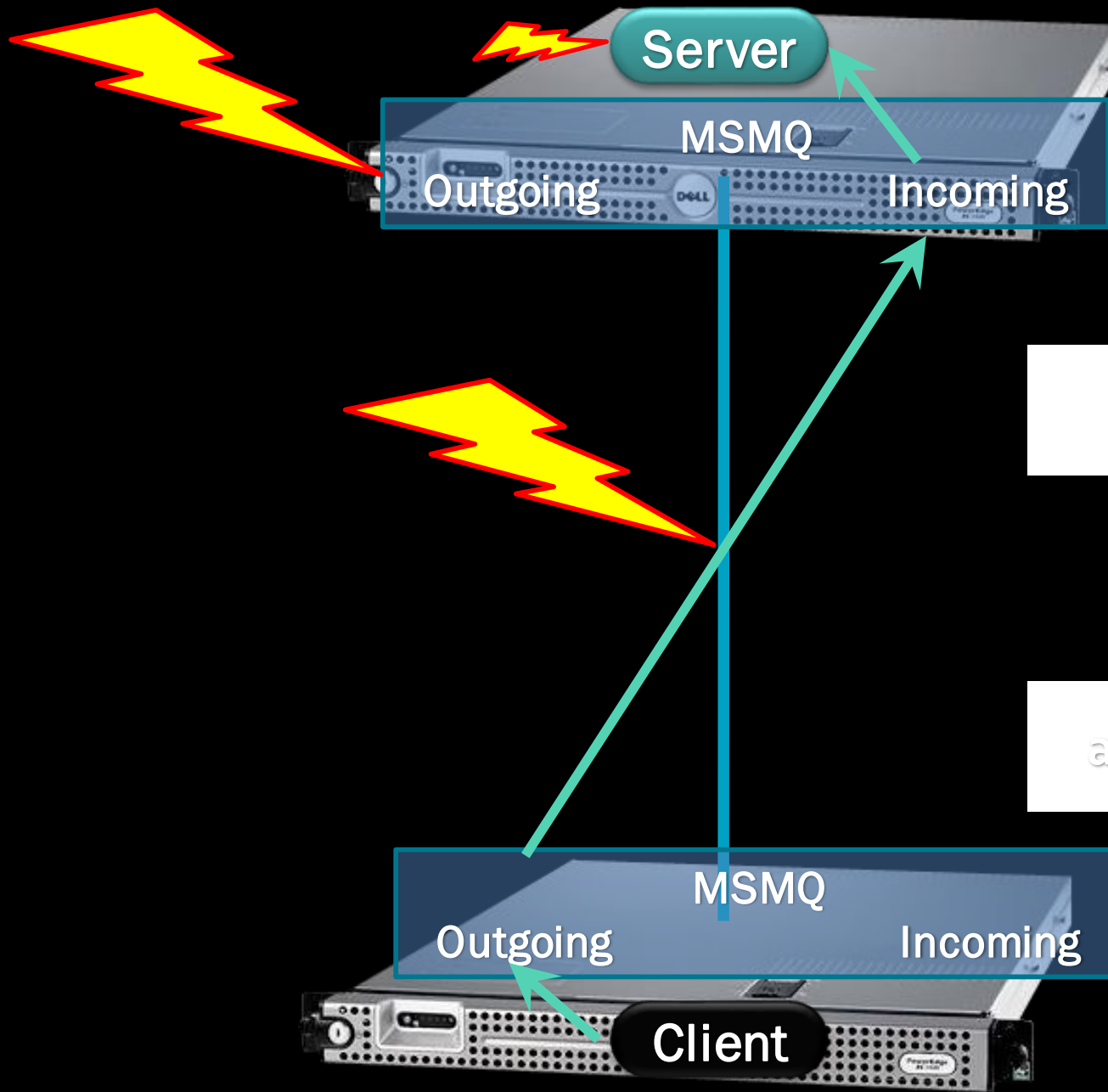
Correlated Request/Response

Publish/Subscribe

# ONE-WAY, FIRE & FORGET MESSAGING



Each message has an Id.  
Seems simple, but there's more to it.

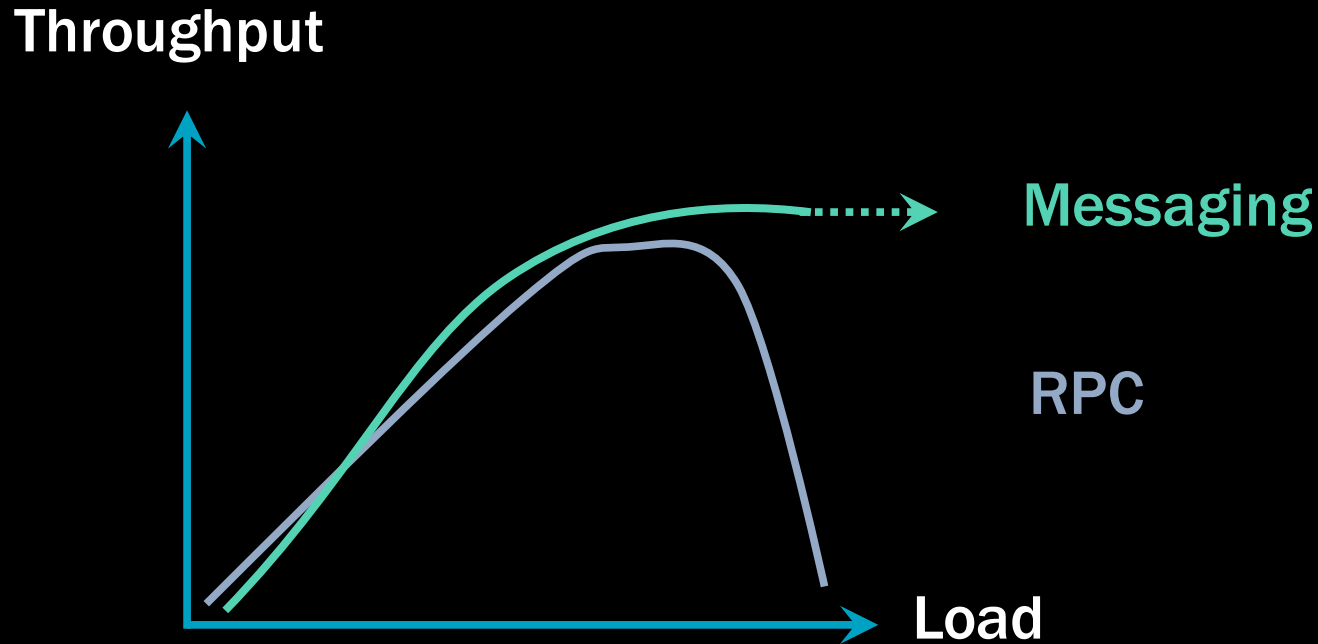


Store and  
Forward



adds resilience

# PERFORMANCE – RPC vs MESSAGING



- With RPC, threads are allocated with load  
With messaging, threads are independent  
Difference due to synchronous blocking calls
- Memory, DB locks, held longer with RPC

# STANDARD SERVICE INTERFACES

## Customer Service

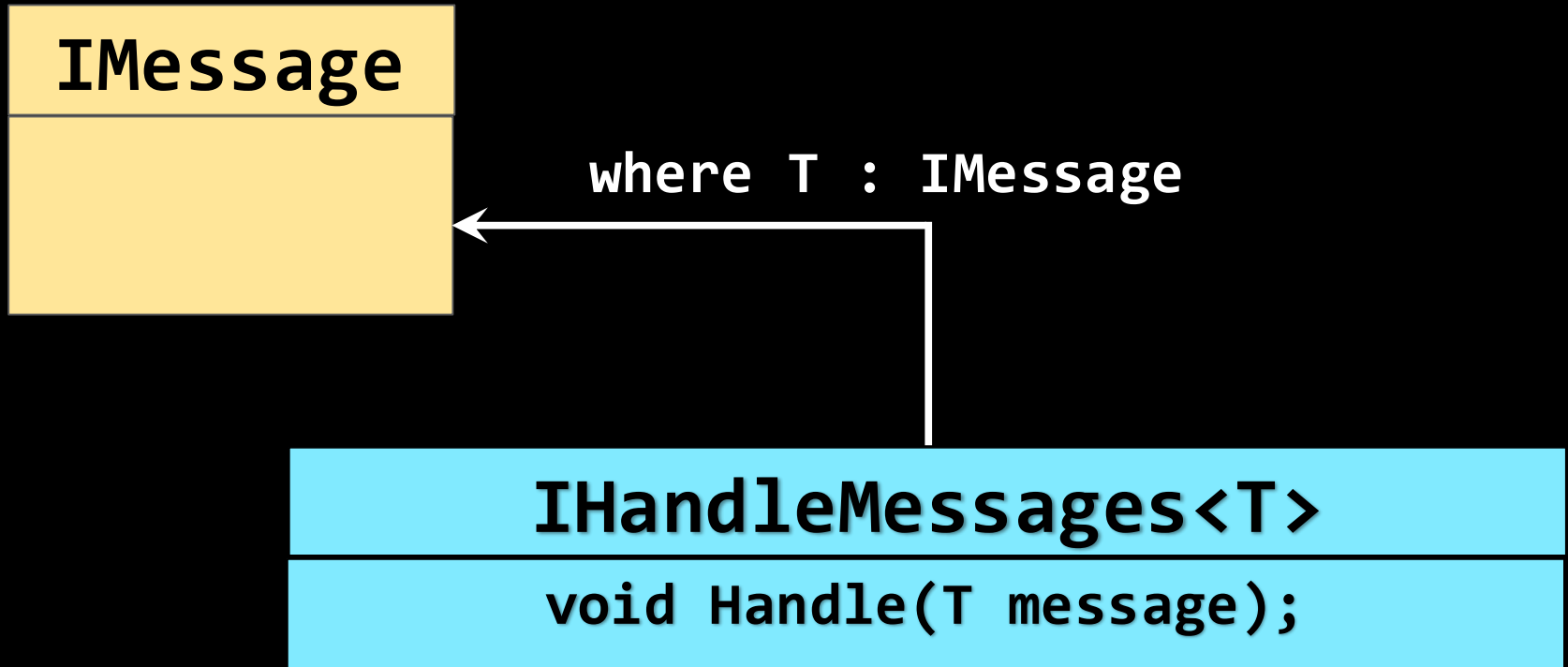
```
void Change_Address(Guid id, Address a);
```

```
void Make_Preferred(Guid id);
```

```
void Change_Credit(Guid id, Credit c);
```

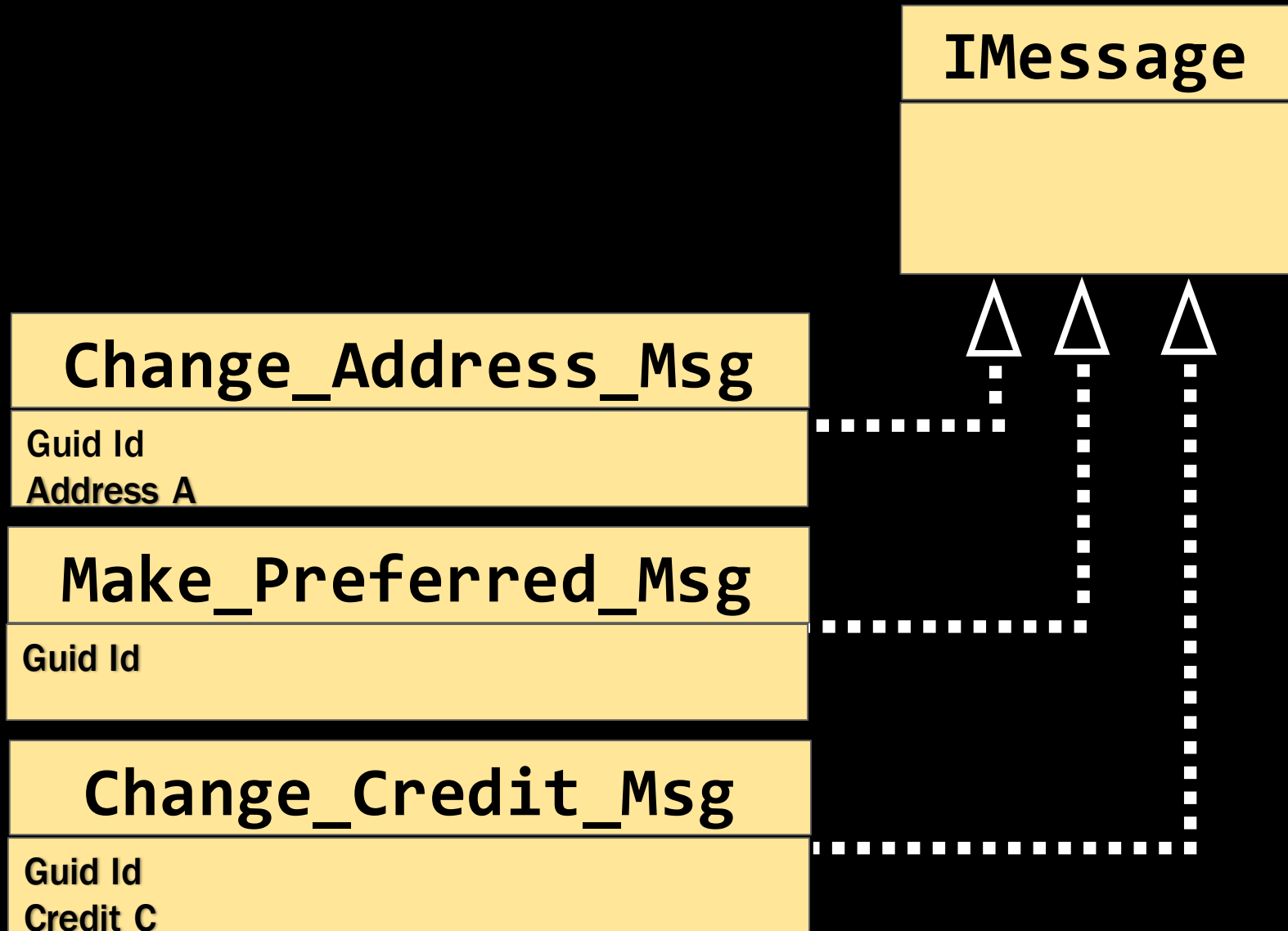
- Problem is that service layers get too large
- Difficult for multiple developers to collaborate
- Difficult to reuse logging, authorization, etc

# EXPLOIT STRONGLY-TYPED MESSAGES





# REPRESENT METHODS AS MESSAGES



# HANDLING LOGIC SEPARATED

**IHandleMessages<T>**

**void Handle(T message);**

**H1 : IHandleMessages<Change\_Address\_Msg>**

**H2 : IHandleMessages<Make\_PREFERRED\_Msg>**

**H3 : IHandleMessages<Change\_Credit\_Msg>**



# MULTIPLE HANDLERS PER MESSAGE

**H1** : IHandleMessages<Change\_Address\_Msg>

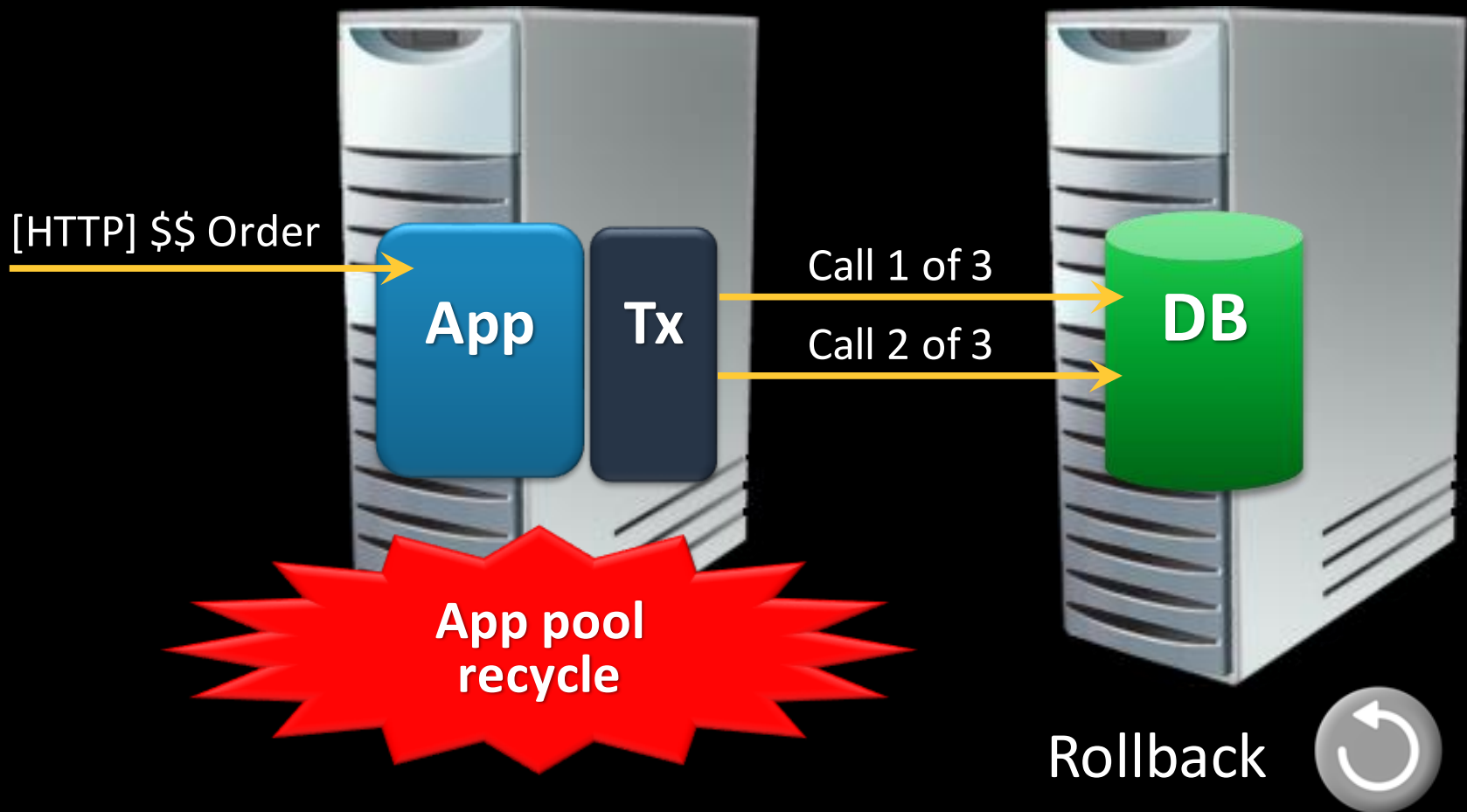
**H4** : IHandleMessages<Change\_Address\_Msgv2>

- Dispatch based on type polymorphism
- Allows for pipeline of handler invocation

# FAULT-TOLERANCE - SCENARIOS

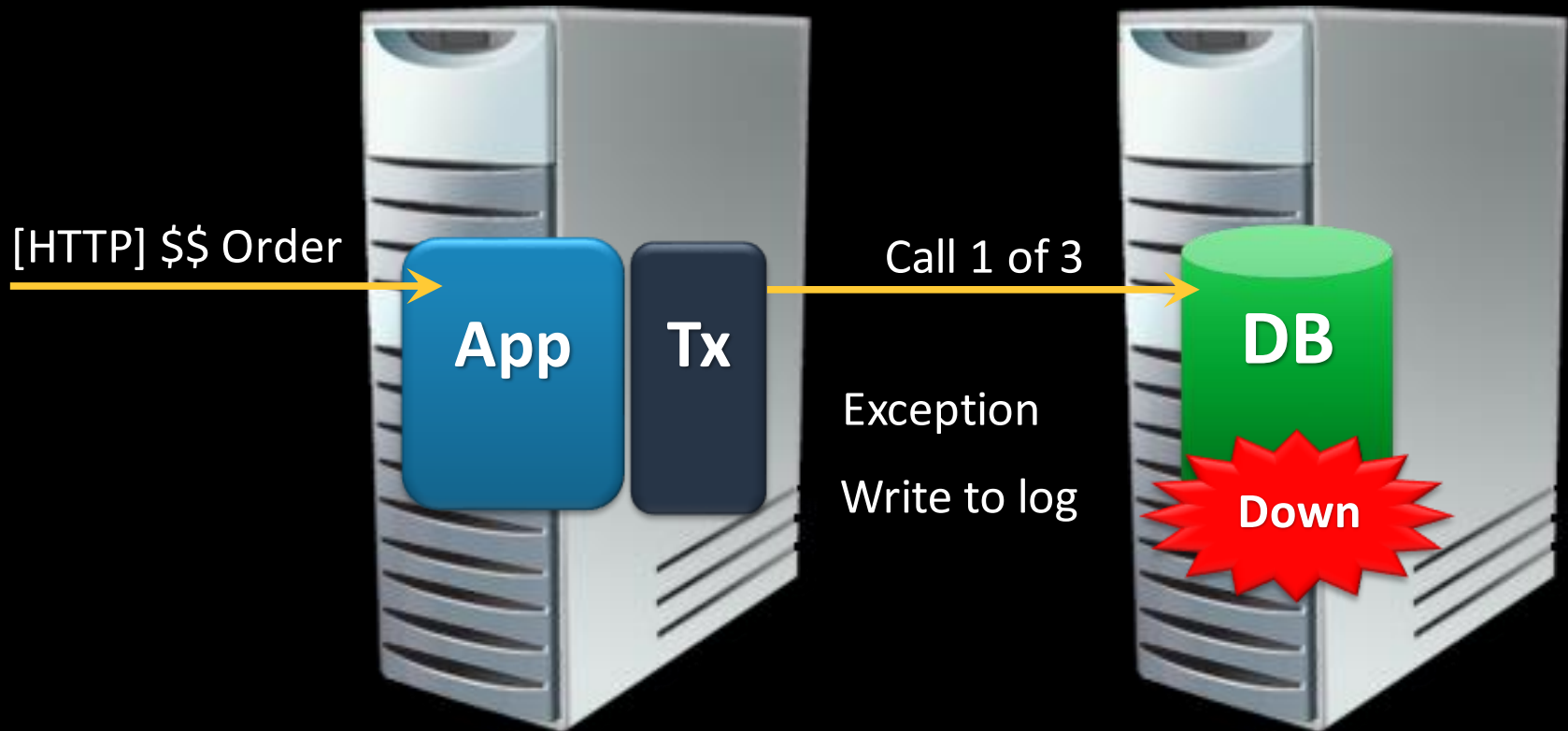
- When servers crash
- When databases are down
- When deadlocks occur in the database

# WHEN SERVERS CRASH



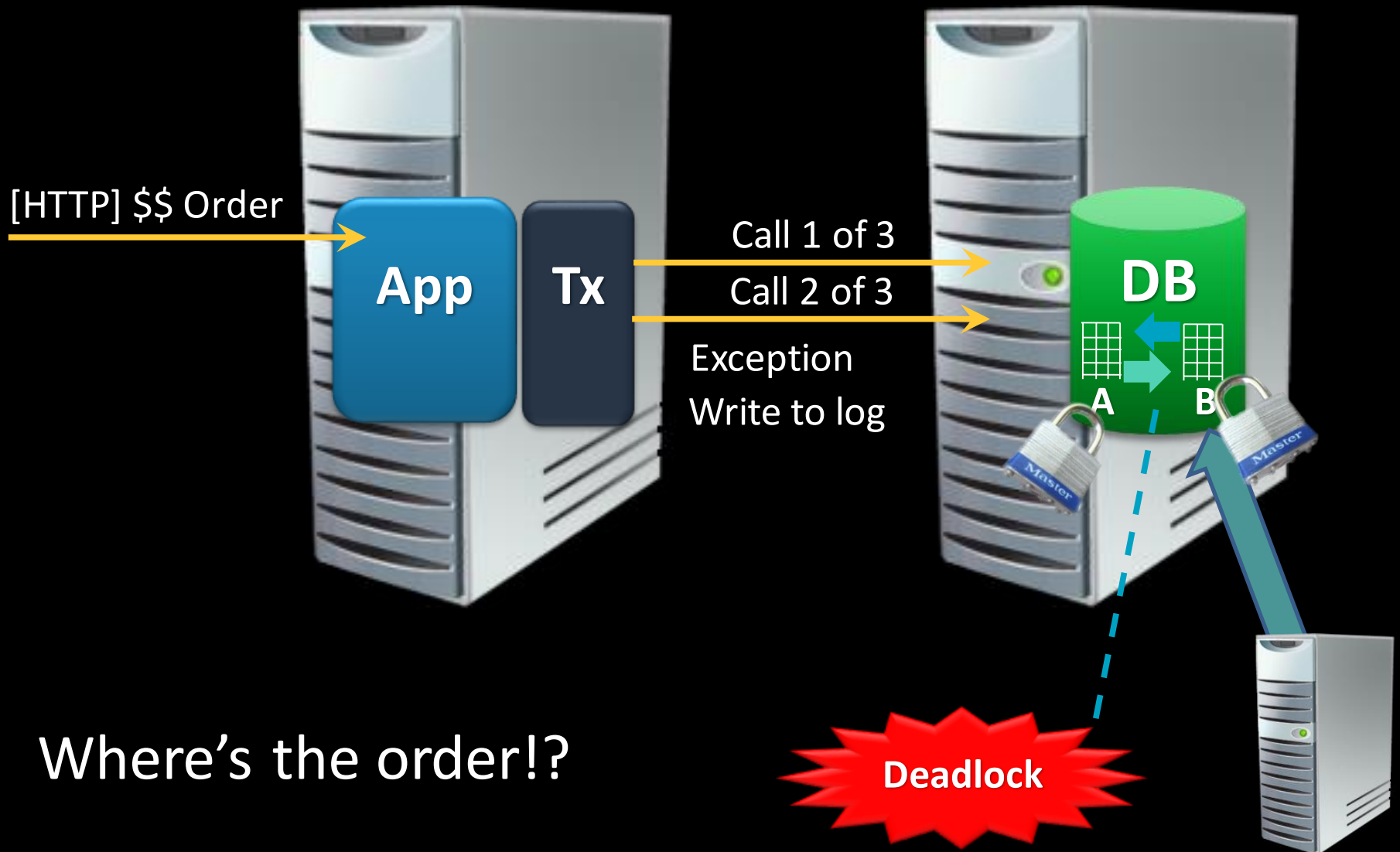
Where's the order!?

# WHEN DATABASES ARE DOWN



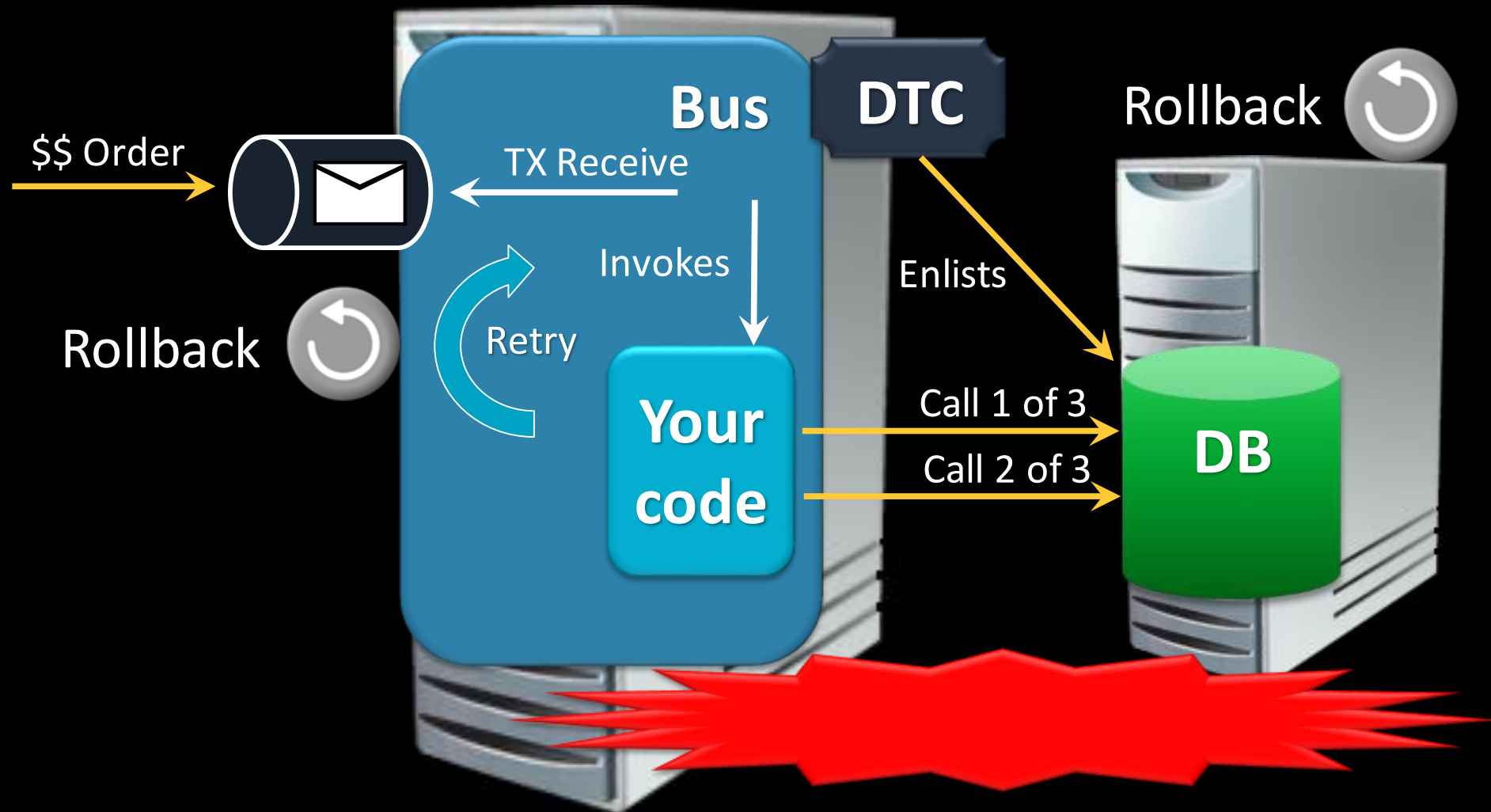
Where's the order!?

# WHEN DEADLOCKS HAPPEN



Where's the order!?

# HOW DOES MESSAGING HELP?



The order is back in the queue

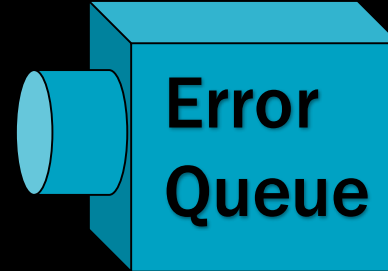


# AFTER ALL RETRIES EXHAUSTED



**Append exception info  
to headers\***

**Moved failed message**



**Notify  
admin**

**a.k.a “poison letter queue”**

**\* NServiceBus feature – not done by all queues natively**

# MONITORING

## Dashboard

### SYSTEM STATUS



3

Heartbeats



50

Failed Messages



1

Custom Checks

ServicePulse v1.2.0 ([update available](#))

ServiceControl v1.9.0 ([update available](#))

### LAST 10 EVENTS



Endpoint has failed to send expected heartbeat to ServiceControl. It is possible that the endpoint could be down or is unresponsive. If this condition persists, you might want to restart your endpoint.

a minute ago

# AUDITING / JOURNALING

- Sends a copy of the message to another queue when it is processed

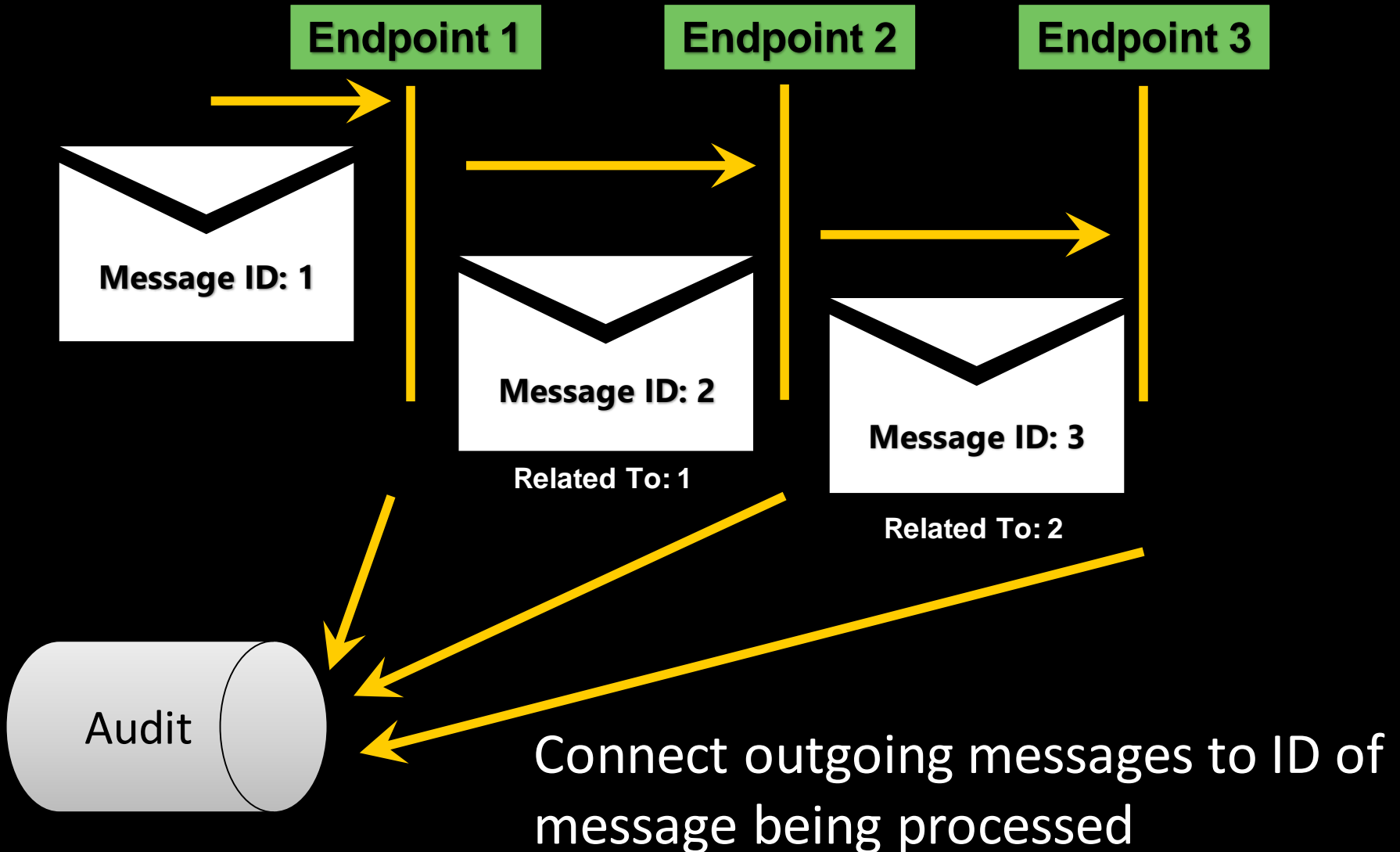
Supported out-of-the-box by most queues

Extract to longer-term storage

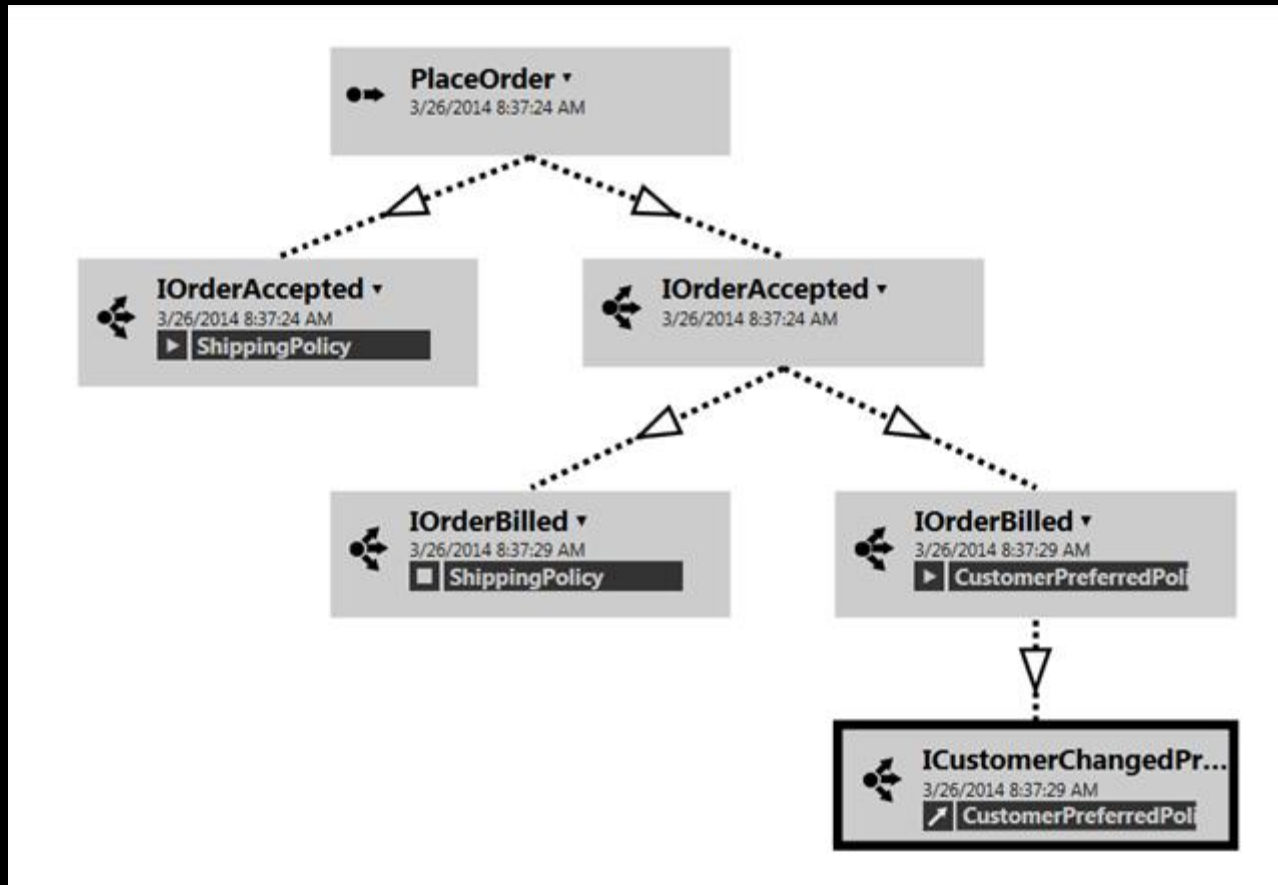
So the queue doesn't "explode"

- A central log of everything that happened
- Can be difficult to interpret by itself

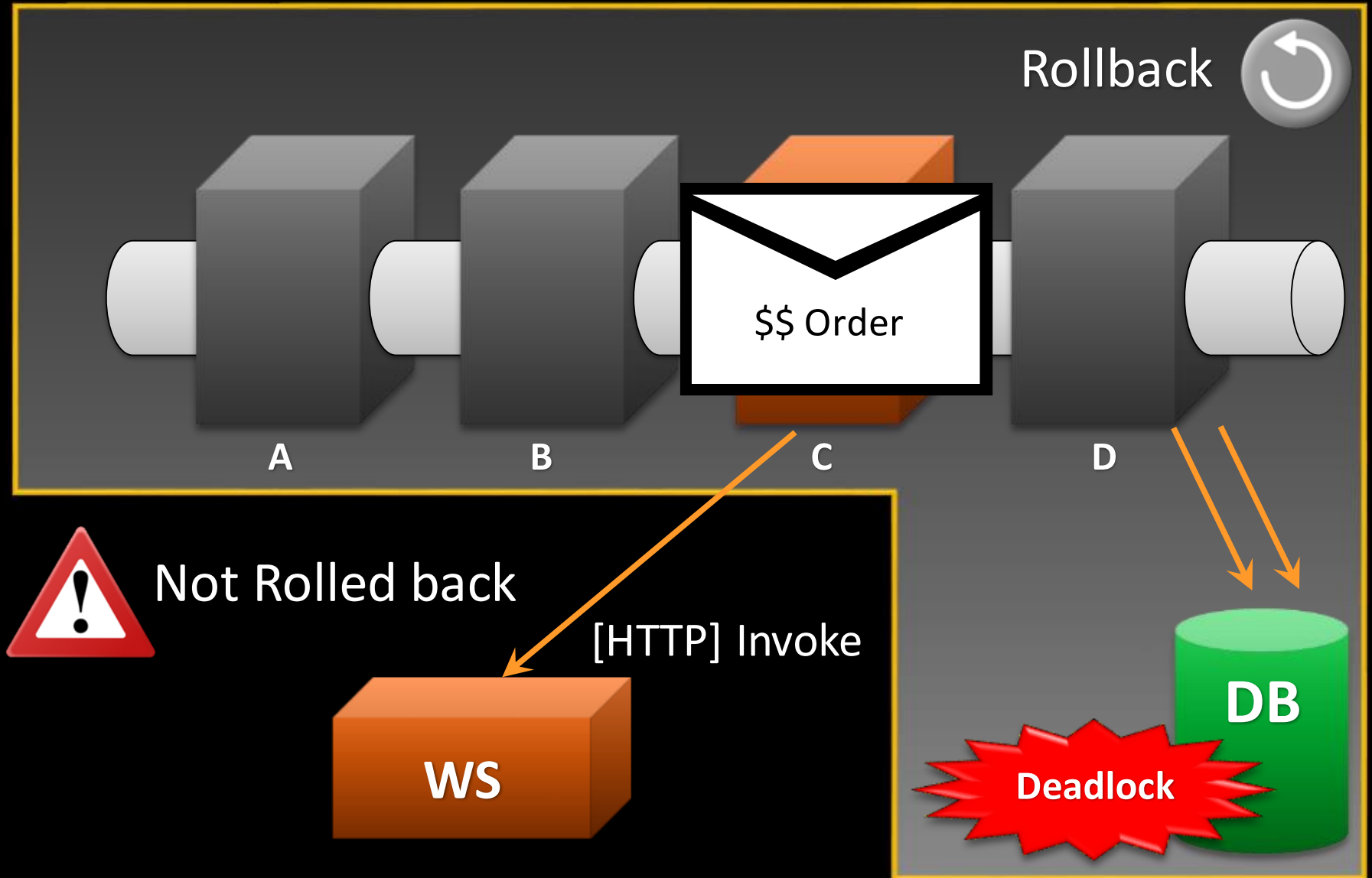
# LEVERAGING MESSAGE HEADERS



# VISUALIZING THE AUDIT STORE

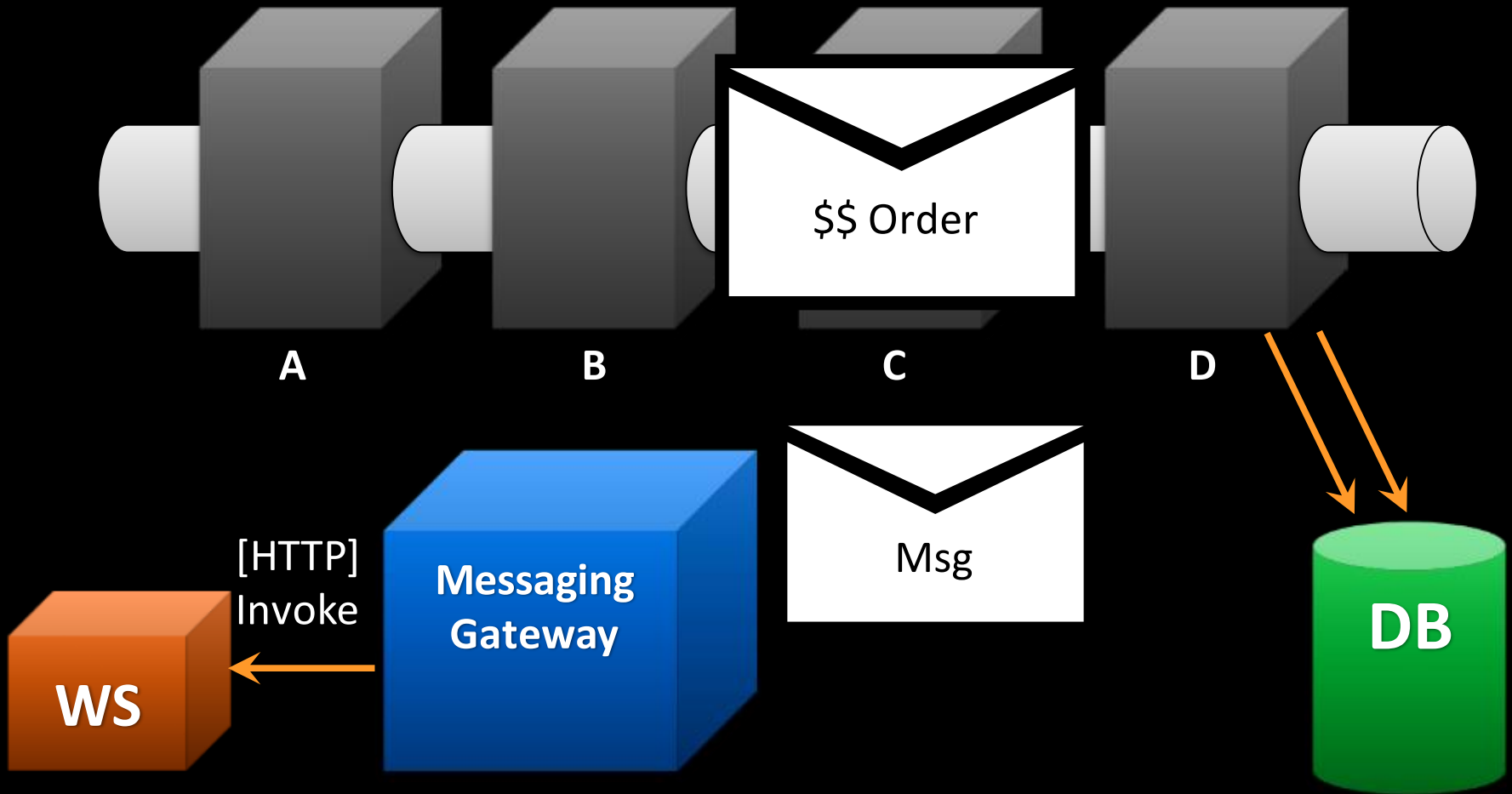


# CALLING WEB SERVICES

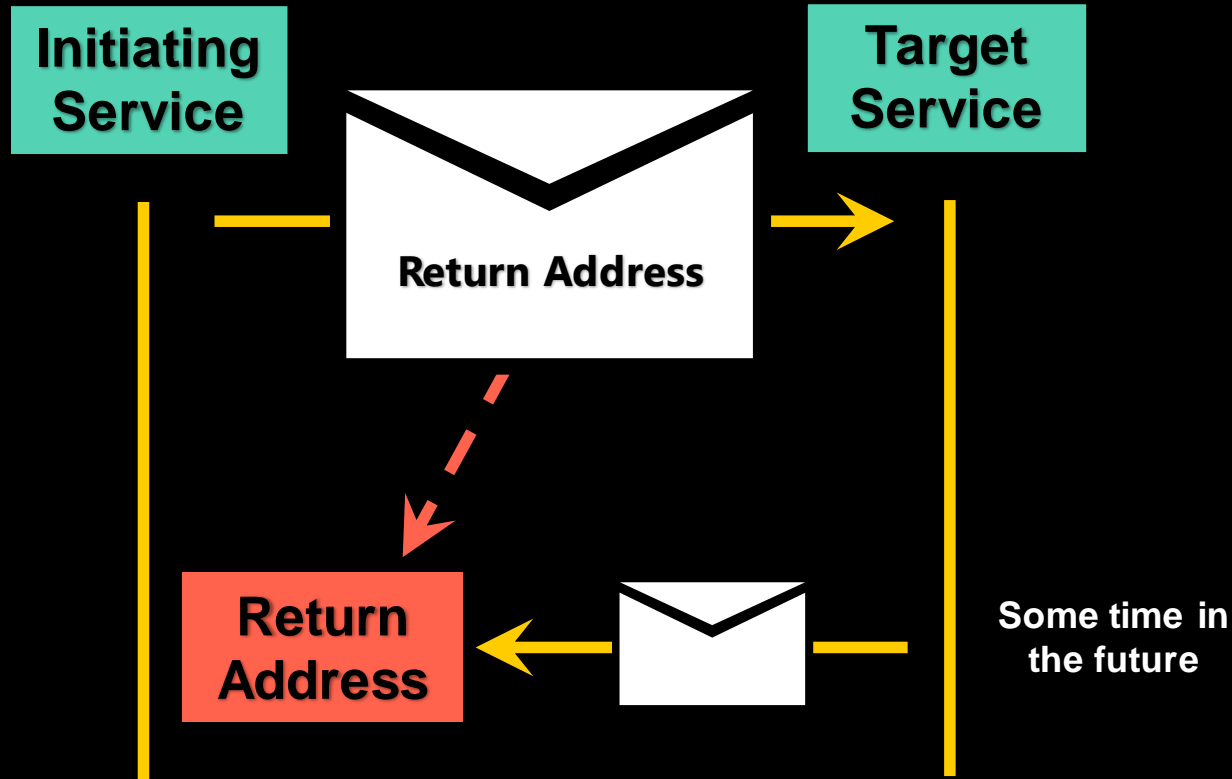


# WEB SERVICES WITH MESSAGING

The message won't be sent if there's a failure

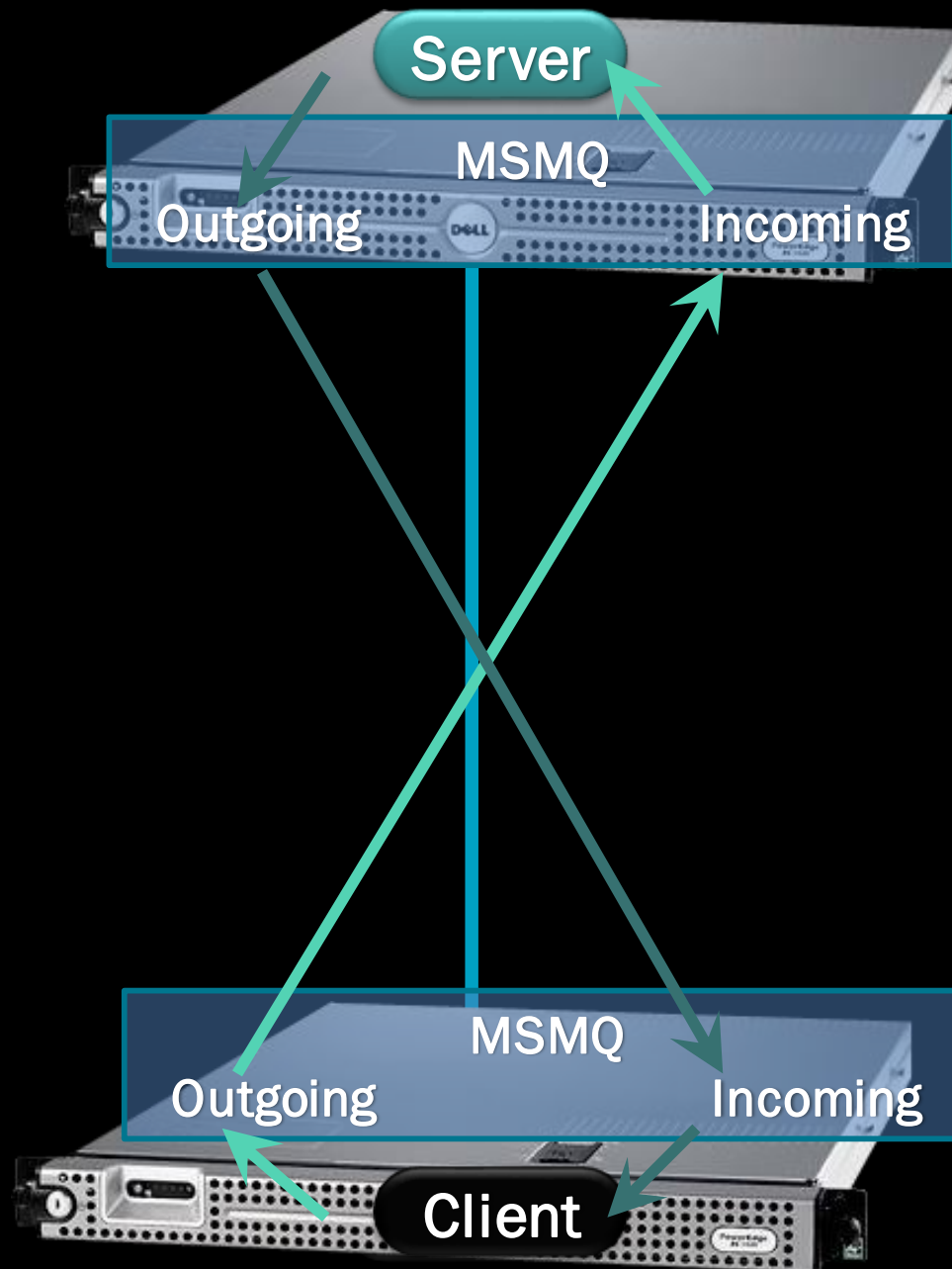


# RETURN ADDRESS PATTERN



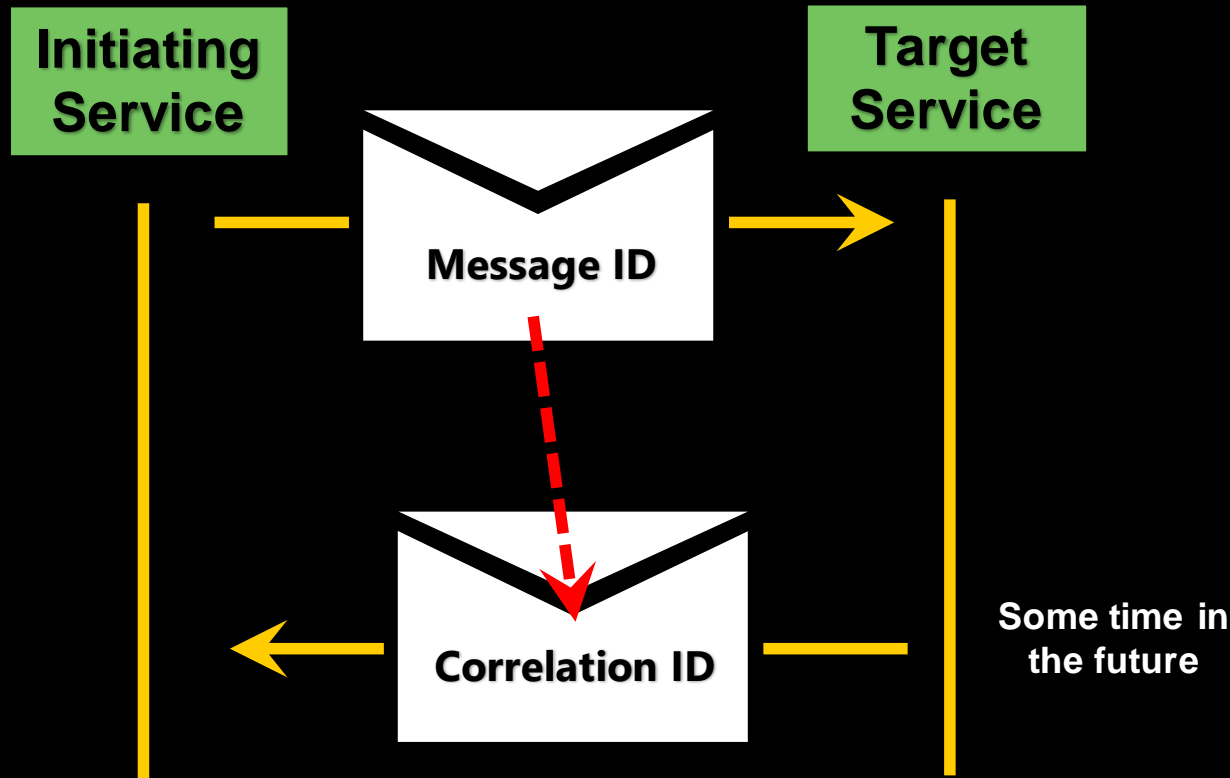
2 Channels: one for requests, one for responses





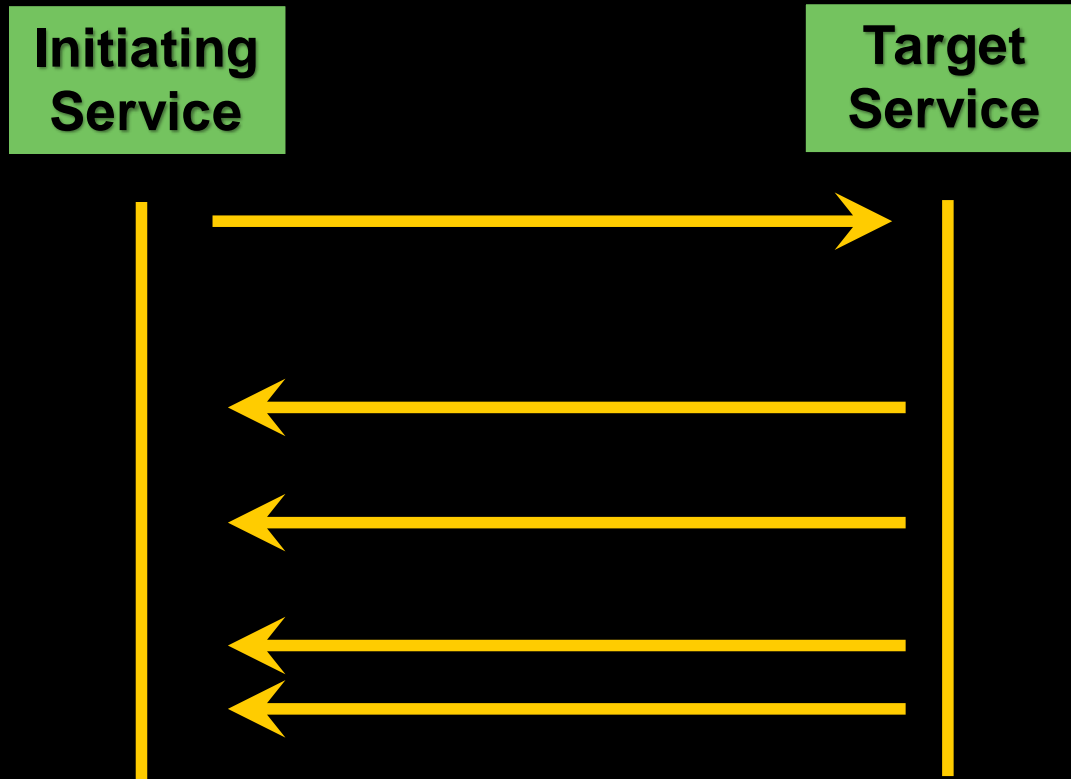
# CORRELATED REQUEST/RESPONSE

Based on Return Address



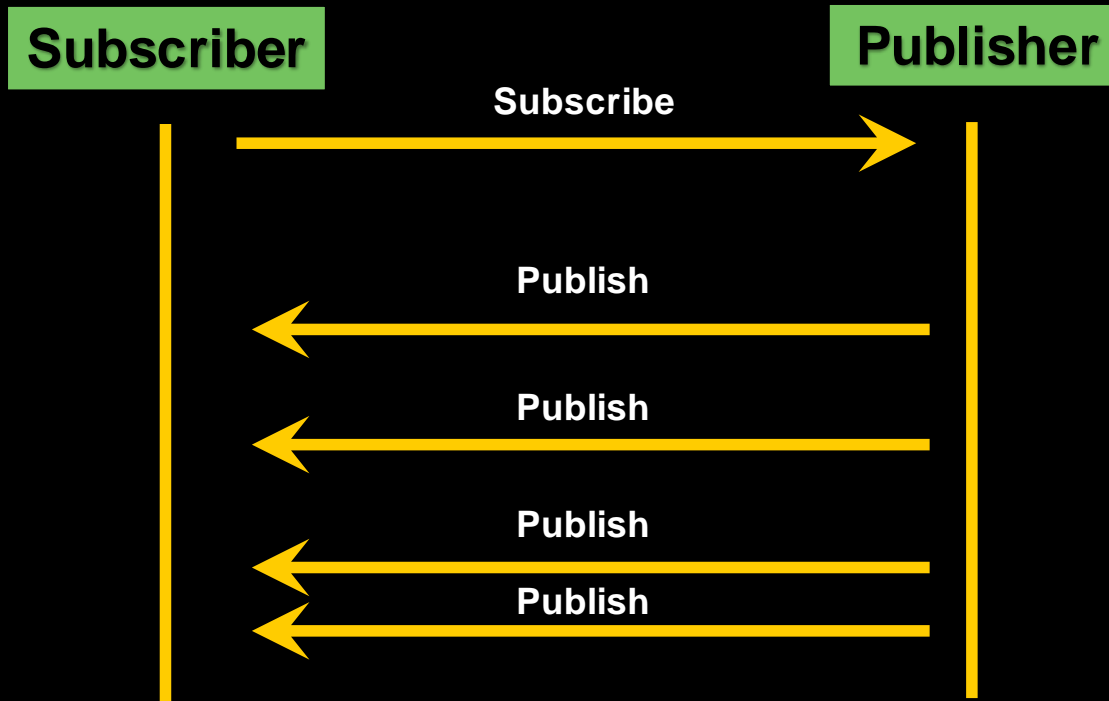
In the header of the response message, there is a correlation id equal to the request message id

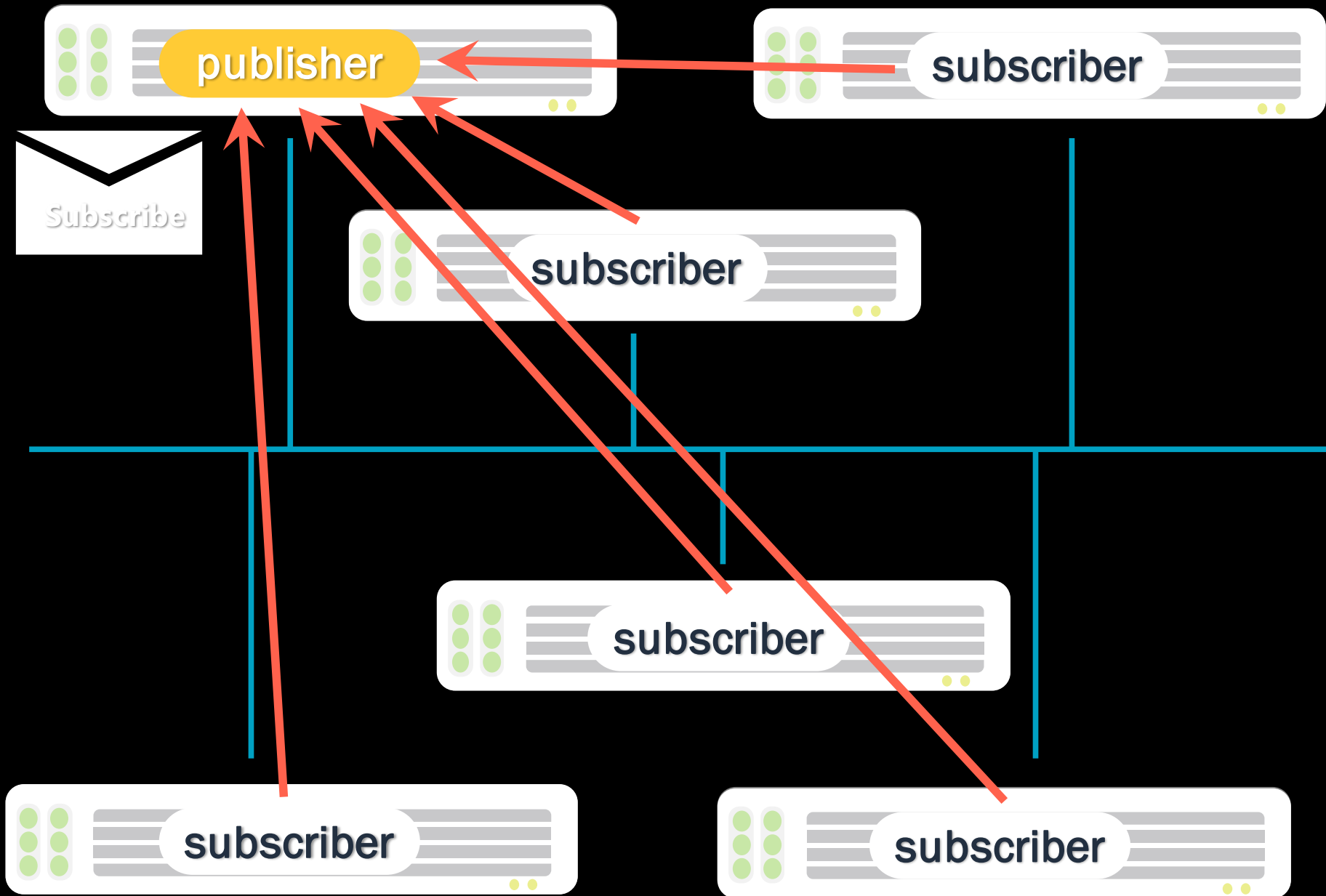
# REQUEST / MULTI RESPONSE

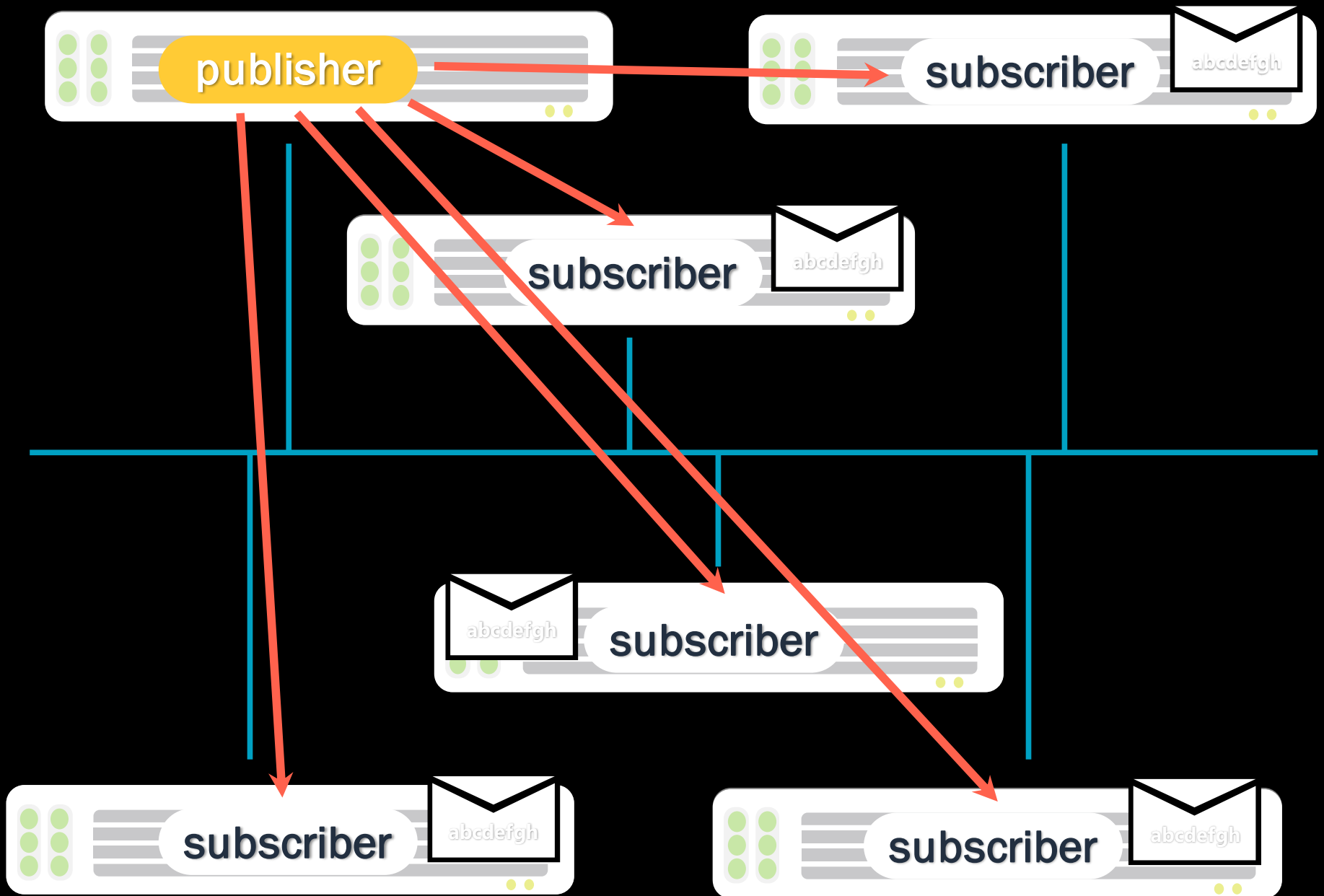


Responses can be of different types

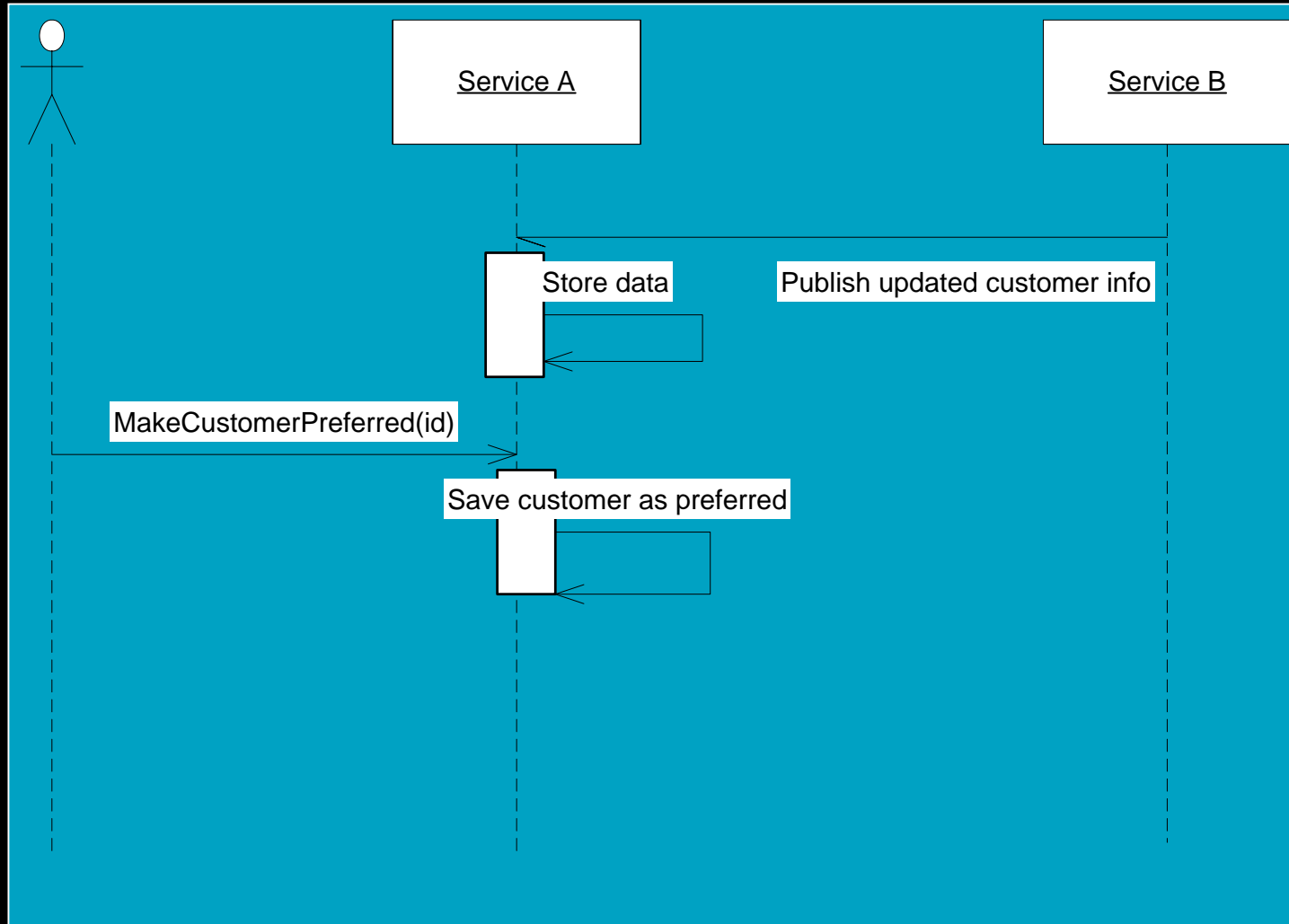
# SUBSCRIBE / PUBLISH

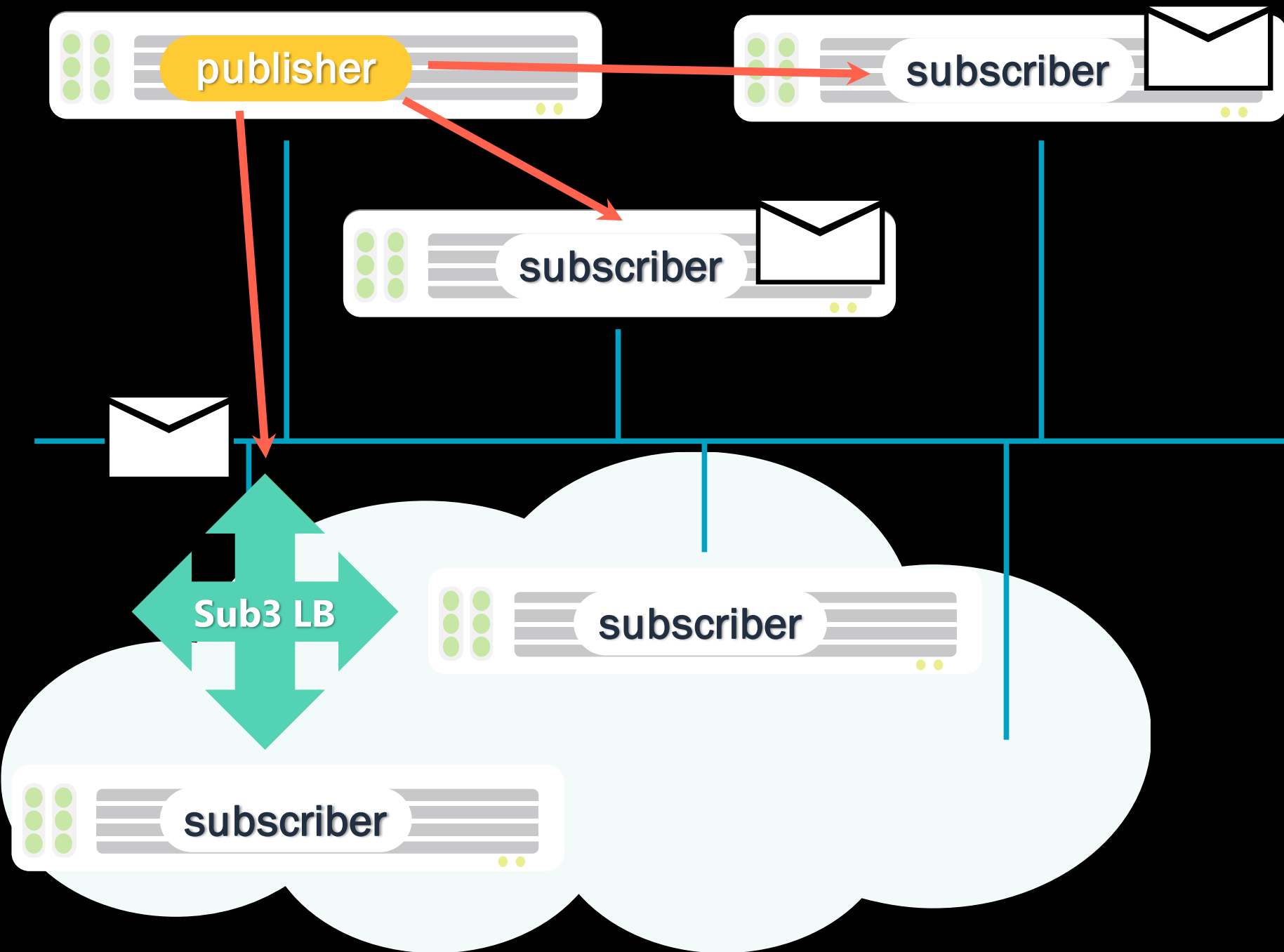






# DON'T FORGET CONSISTENCY BOUNDARIES







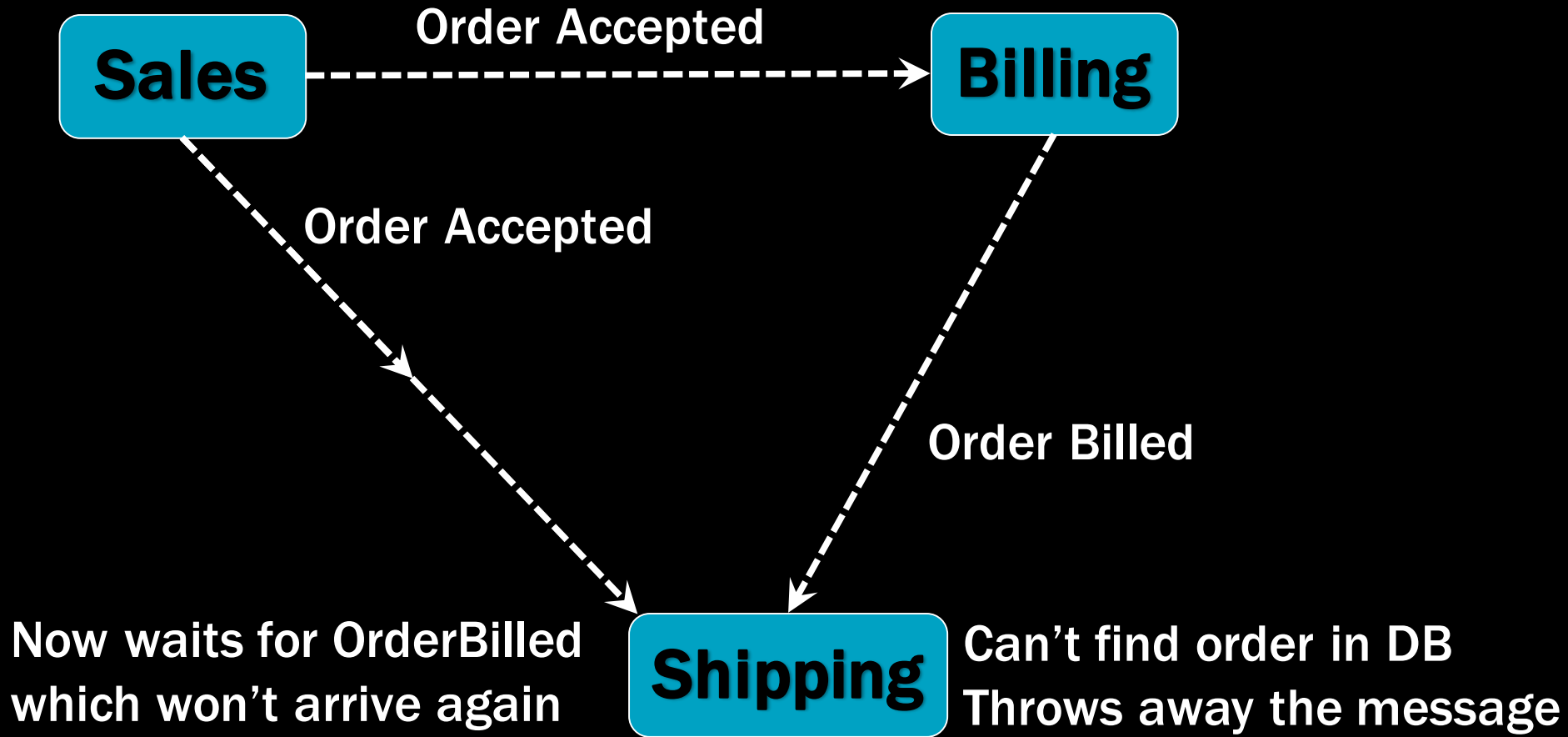
# TOPIC HIERARCHIES & POLYMORPHISM

- Subscribe to "Products", "Products.InStock", "Products.InStock.PricedToClear"
- Multiple-inheritance even more interesting  
Publishing an event A which inherits B, C, and D  
Can subscribe to any or all A, B, C, or D  
Must use interfaces (not classes)  
    Might not be supported by standard serializers

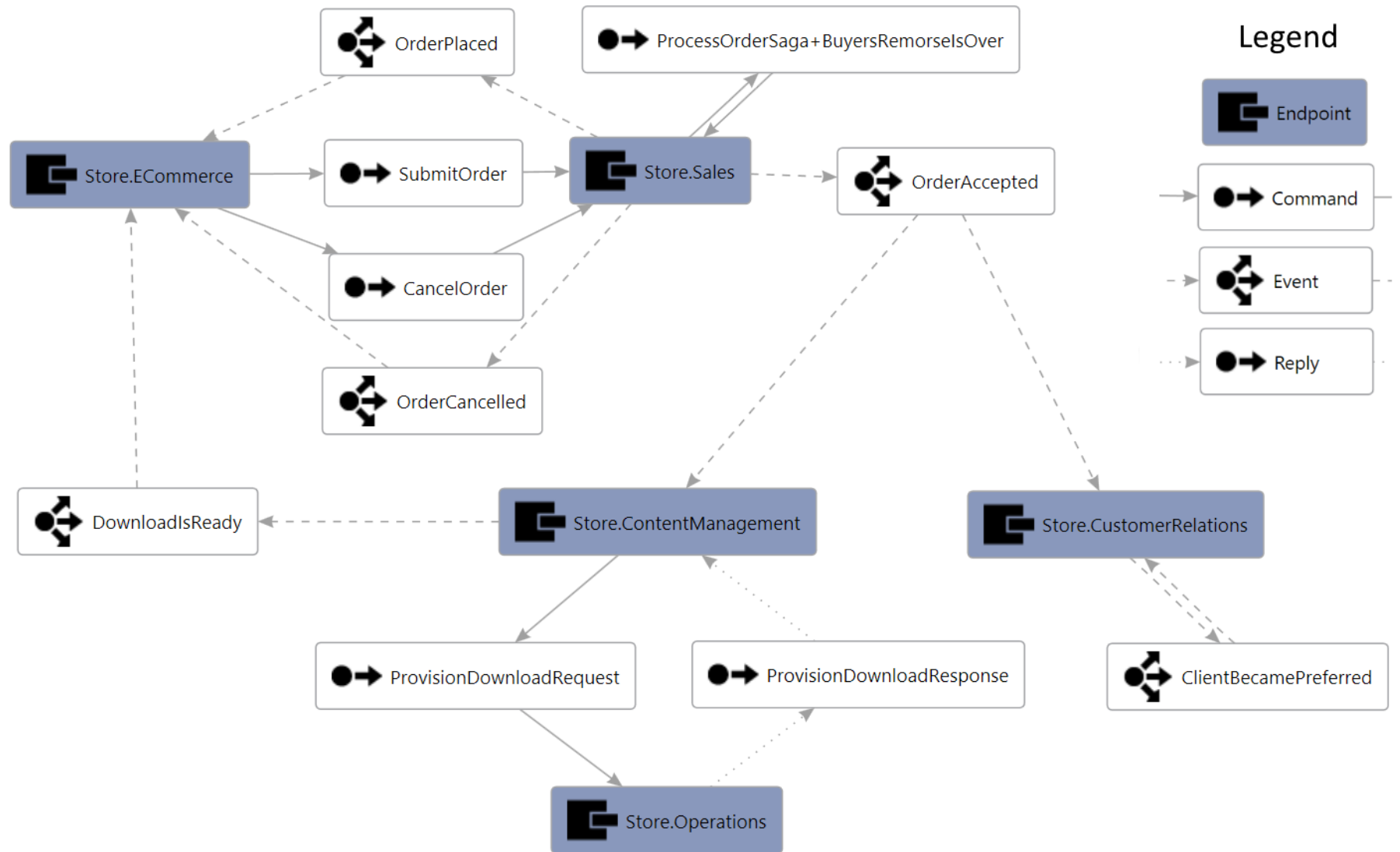
# EVENTS: IN-PROCESS VS. DISTRIBUTED

- In-memory, synchronous invocation  
Publisher can know when all subscribers up to date
- Distributed, asynchronous invocation  
Publisher (and other subscribers) can't know

# OUT-OF-ORDER EVENTS



# VISUALIZATION WITH MESSAGING



# SUMMARY

- Building blocks are simple
  - IMessage
  - IMessageHandler
  - Send, Reply, and Publish
- Identifying boundaries is most important

# EASING CORPORATE ADOPTION

- People are afraid of change
- Meet them where they are
- Consider using database tables under a message-driven API
- Diffuses admin/backup/monitoring objections
- Message-driven code is a good first step

# ARCHITECTURAL STYLES

## BUS & BROKER

# WHAT IS AN "ARCHITECTURAL STYLE"?

An **architectural style** is a coordinated set of architectural constraints that restricts the roles/features of architectural elements and the allowed relationships among those elements within any architecture that conforms to that style.

*Fielding 2000*



# WHAT IS AN "ARCHITECTURAL STYLE"?

In plain English:

What is and isn't allowed in an architecture

Doesn't say "there can be only one"

Should expect multiple styles in a project

- Layering, MVC, pipes & filters, etc.

# WHAT TO USE WHEN?

# Layers

# CQRS

# Pipes & Filters

# EDA

# DDD

# MVC

# REST

# SOA

# SOA AS AN ARCHITECTURAL STYLE

- SOA likely to be founded on messaging
- It is best to first understand current styles also founded on messaging before going to SOA

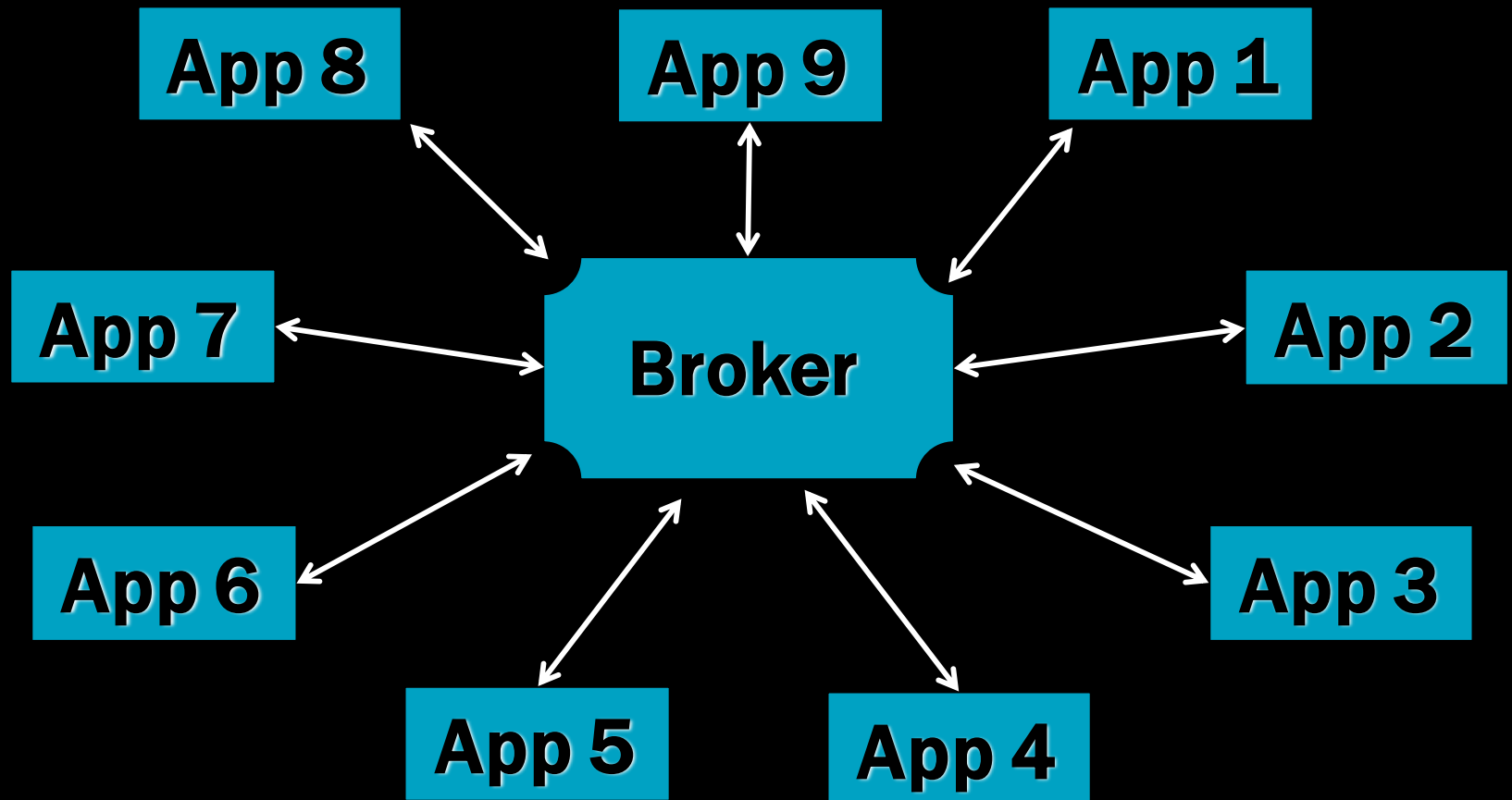
# BUS & BROKER COMMONALITIES

- Attempt to handle spatial coupling

# BROKER ARCHITECTURAL STYLE

Also known as “Hub and Spoke” and “Mediator”

Designed to avoid having to change apps – EAI



# BROKER CHARACTERISTICS

- Broker is physically separate
- All communication goes through the broker
- Broker handles fail over, routing
- The broker is a single point of failure, must be robust and performant.

# BROKER TECHNOLOGY

- BizTalk / WebSphere / Sonic ESB
- MS Sql Service Broker
- BPEL Engines
- CORBA
- UDDI







# BROKER ADVANTAGES

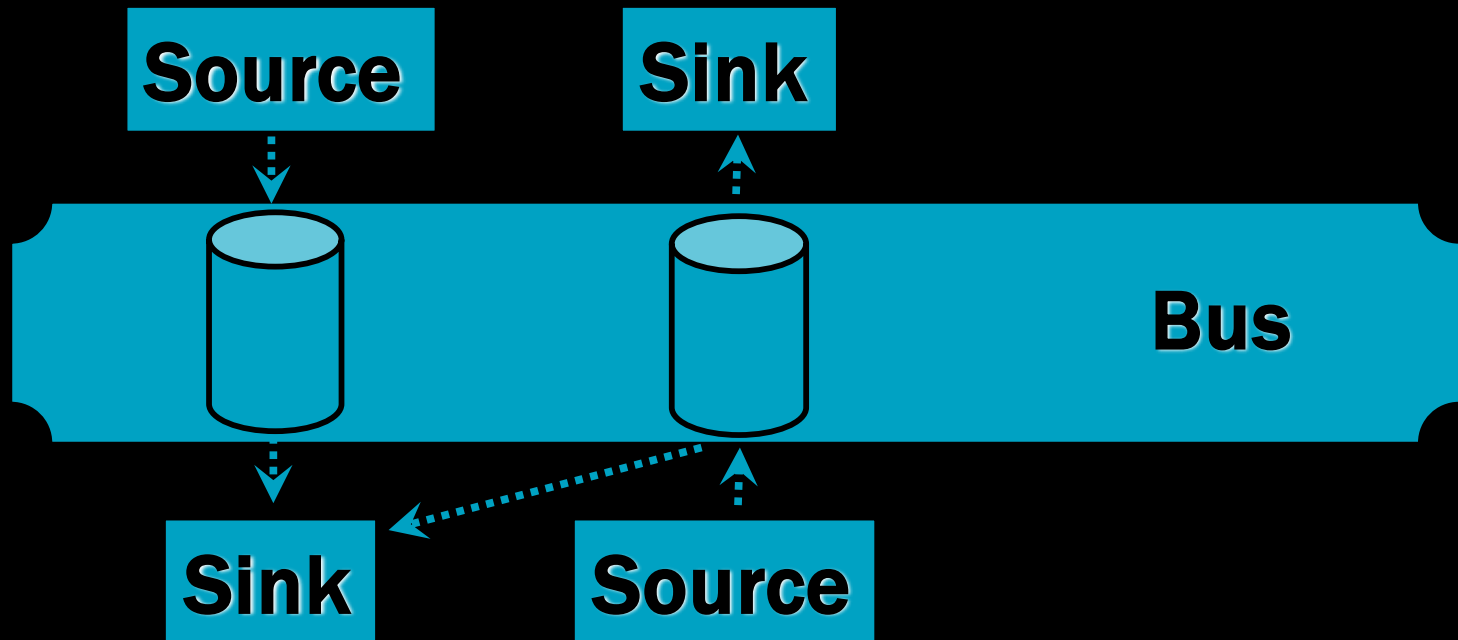
- Concentrating all communications to a single logical entity, enables central management
- Enables “intelligent” routing, data transformation, orchestration
- Doesn't require changes to surrounding apps

# BROKER DISADVANTAGES

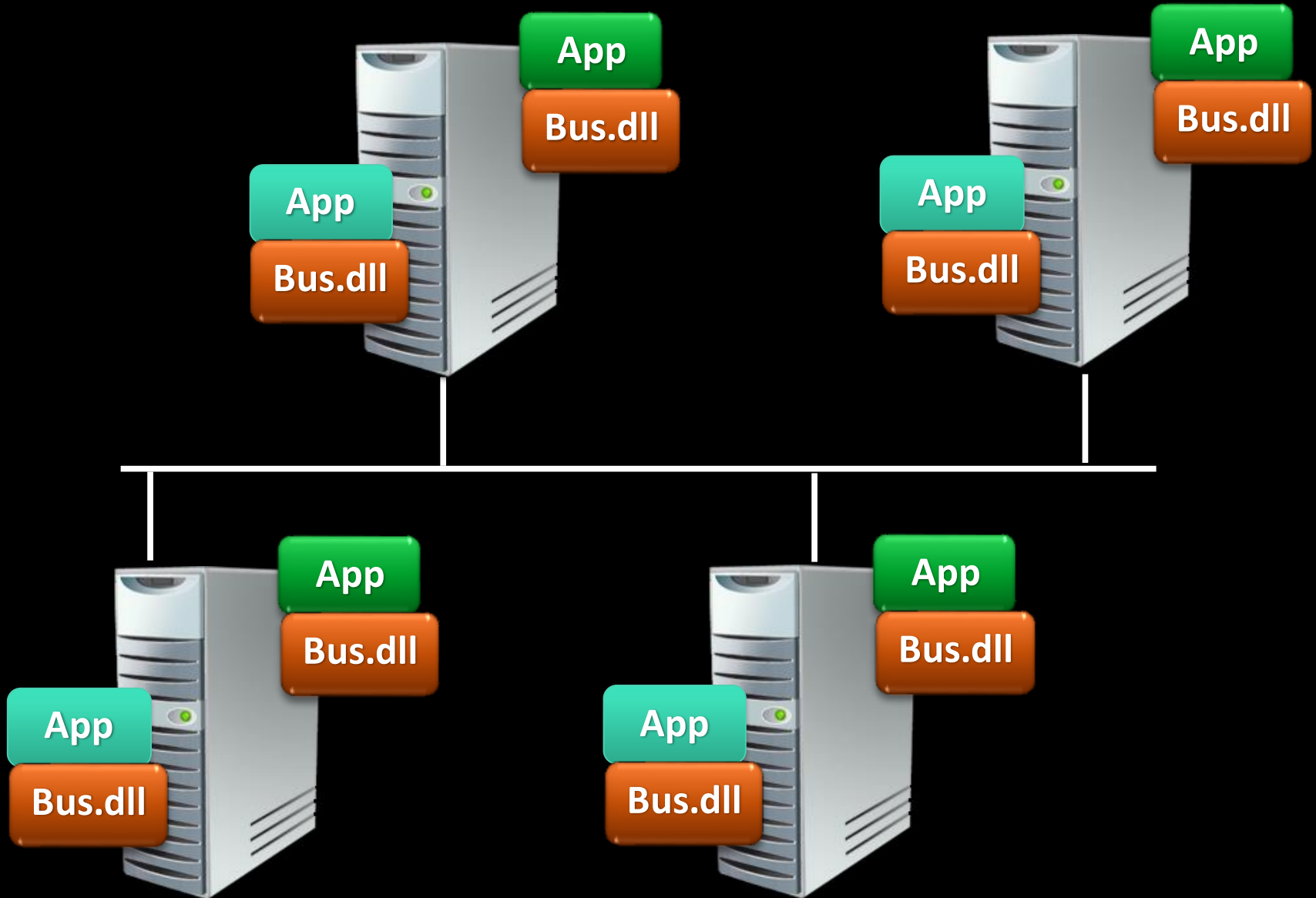
- Embodies the 11<sup>th</sup> fallacy:  
  
“Business logic can and should be centralized”
- Procedural programming at a large scale  
Without good unit testing or source control
- Prevents apps from gaining autonomy

# BUS ARCHITECTURAL STYLE

- Event source and sinks use bus for pub/sub
- Designed to allow independent evolution of sources and sinks



# BUS TOPOLOGY

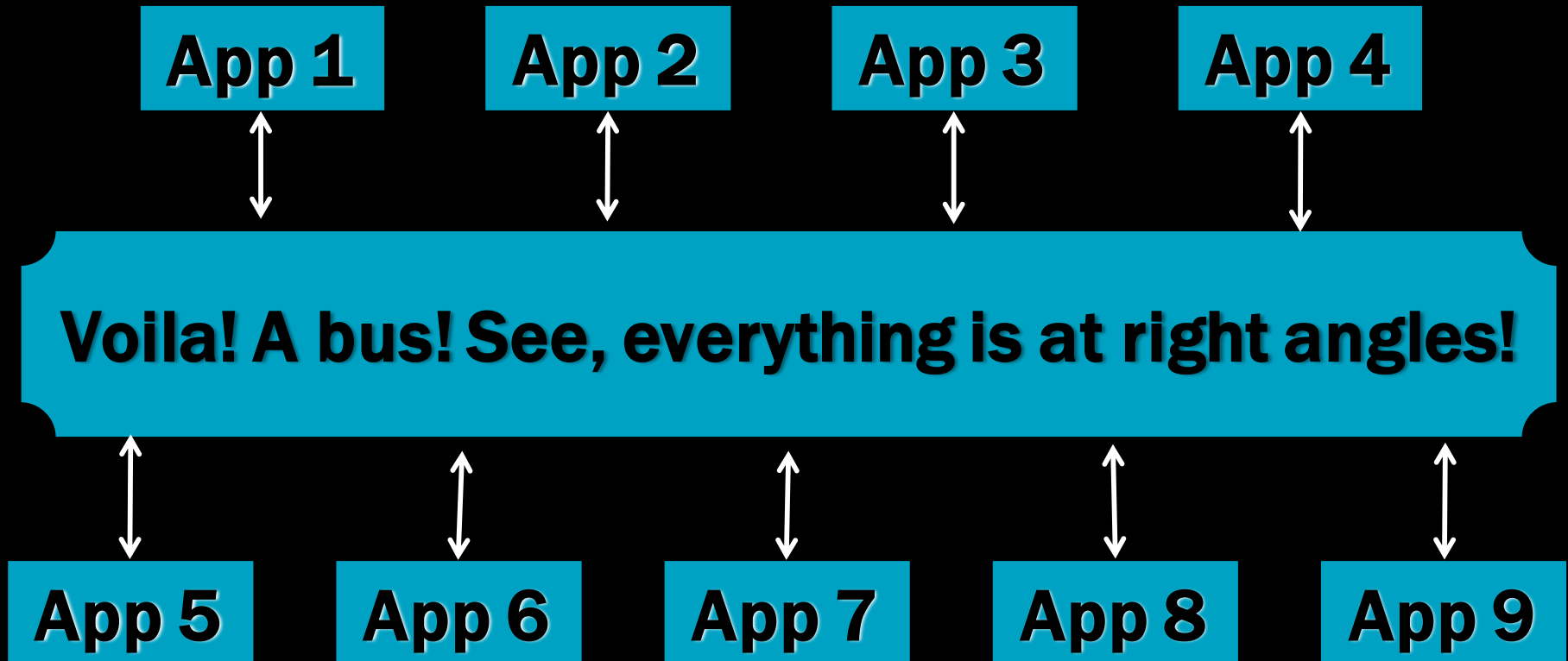


# BUS CHARACTERISTICS

- Bus is not necessarily physically separate
- Communication is distributed  
No single point of failure
- Bus is simpler – no content-based routing or data transformations
- Orthogonal to the broker style

# BROKER TECHNOLOGIES CALLED ESBS

- Some "New" ESB products actually brokers  
WebSphere, Mule, Sonic



# BUS TECHNOLOGY

- Open-source on the Microsoft platform  
NServiceBus, MassTransit, Rhino Service Bus
- “Old” JMS implementations / Federated AMQP  
Tibco Rendezvous, RabbitMQ, Qpid  
Not all support distributed / XA transactions

# BUS ADVANTAGES

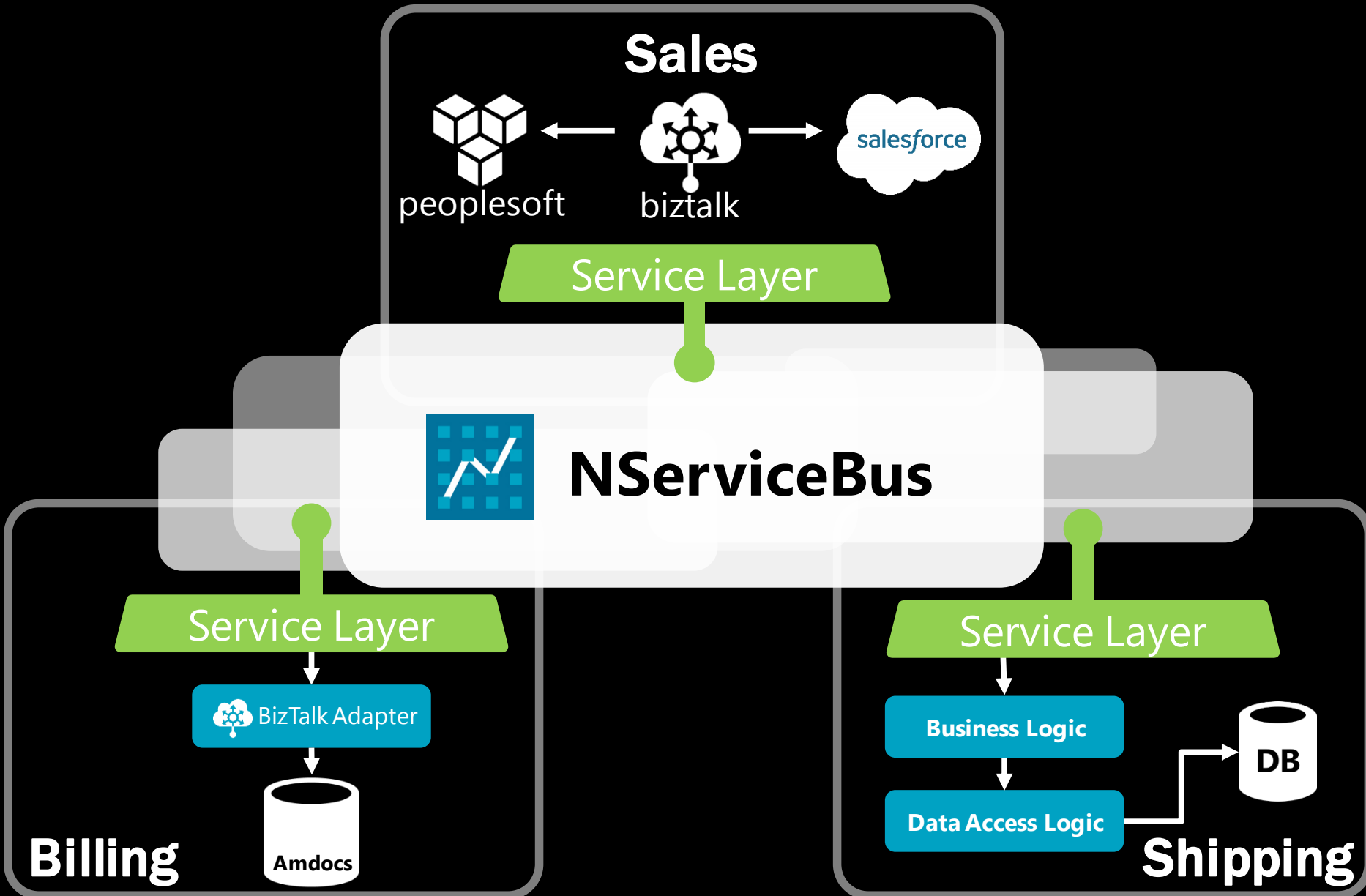
- No single point of failure
- Doesn't break service autonomy



# BUS DISADVANTAGES

- More difficult to design distributed solutions than centralist ones

# THE BEST OF BOTH WORLDS



# 3 ELEMENTS OF INTEGRATION

- Data transformation



XML, DB, Flat file, EDI  
XBRL, WS, etc

- Protocol Bridging



IP\*Works  
Ftp, SMTP, POP, SMS  
LDAP, DNS, etc

- Business Logic

# SUMMARY

- Feature-rich broker products less suited to distributed systems than robust bus products.
- Projects will likely use a combination of both bus and broker

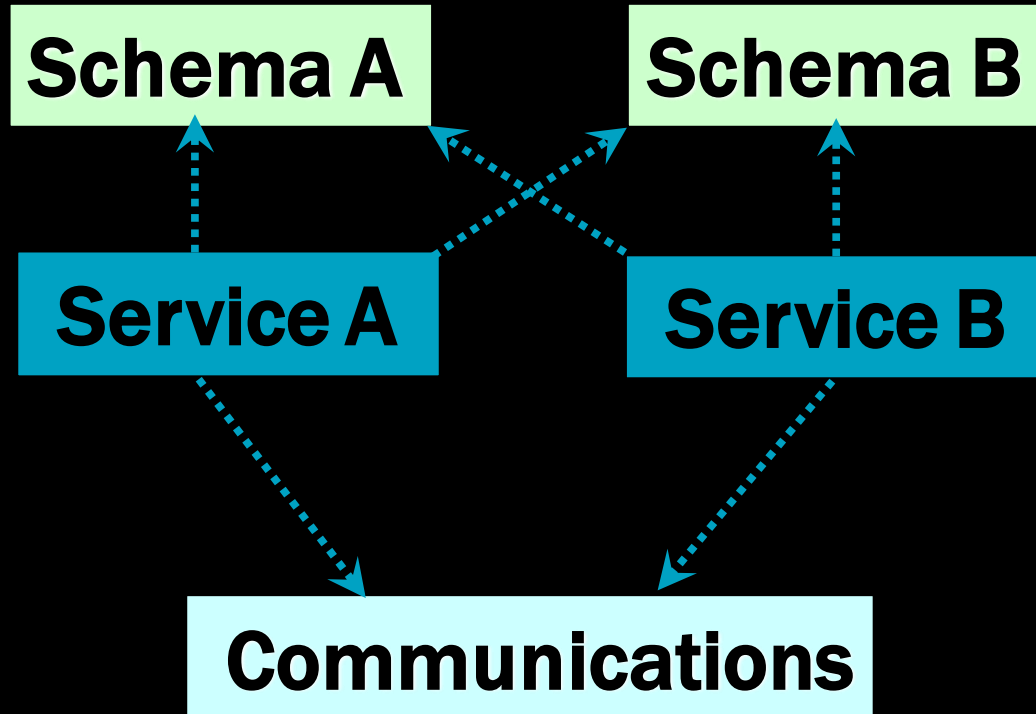
# SOA BUILDING BLOCKS

# WHAT IS A SERVICE?

Tenets of Service Orientation:

1. Services are autonomous.
2. Services have explicit boundaries
3. Services share contract & schema,  
not class or type
4. Service interaction is controlled by policy.

# SERVICE ORIENTATION



# WHAT IS A SERVICE?

A service is the technical authority for a specific business capability.

All data and business rules reside within the service.

Nothing is “left over” after identifying services

Everything must be in some service



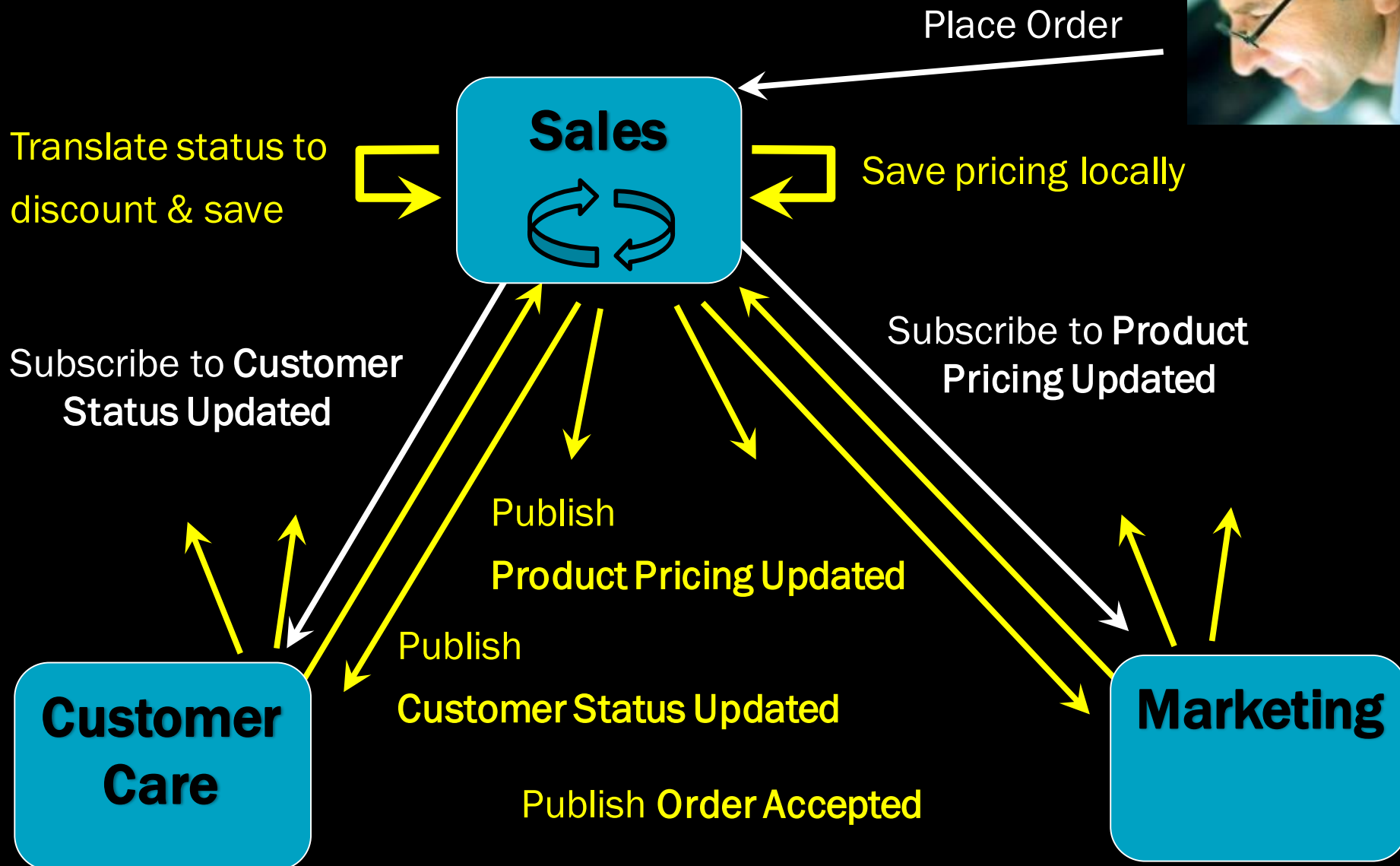
# WHAT A SERVICE IS NOT

- A service that has only functionality is a function not a service.  
Like calculation, validation
- A service that has only data is a database, not a service.  
Like [create, read, update, delete] entity
- WSDL / REST doesn't change logical responsibility

# 4+1 VIEWS OF SOFTWARE ARCHITECTURE

- Services are in the logical view
- Mapping to the development view, a service could be a source control repository

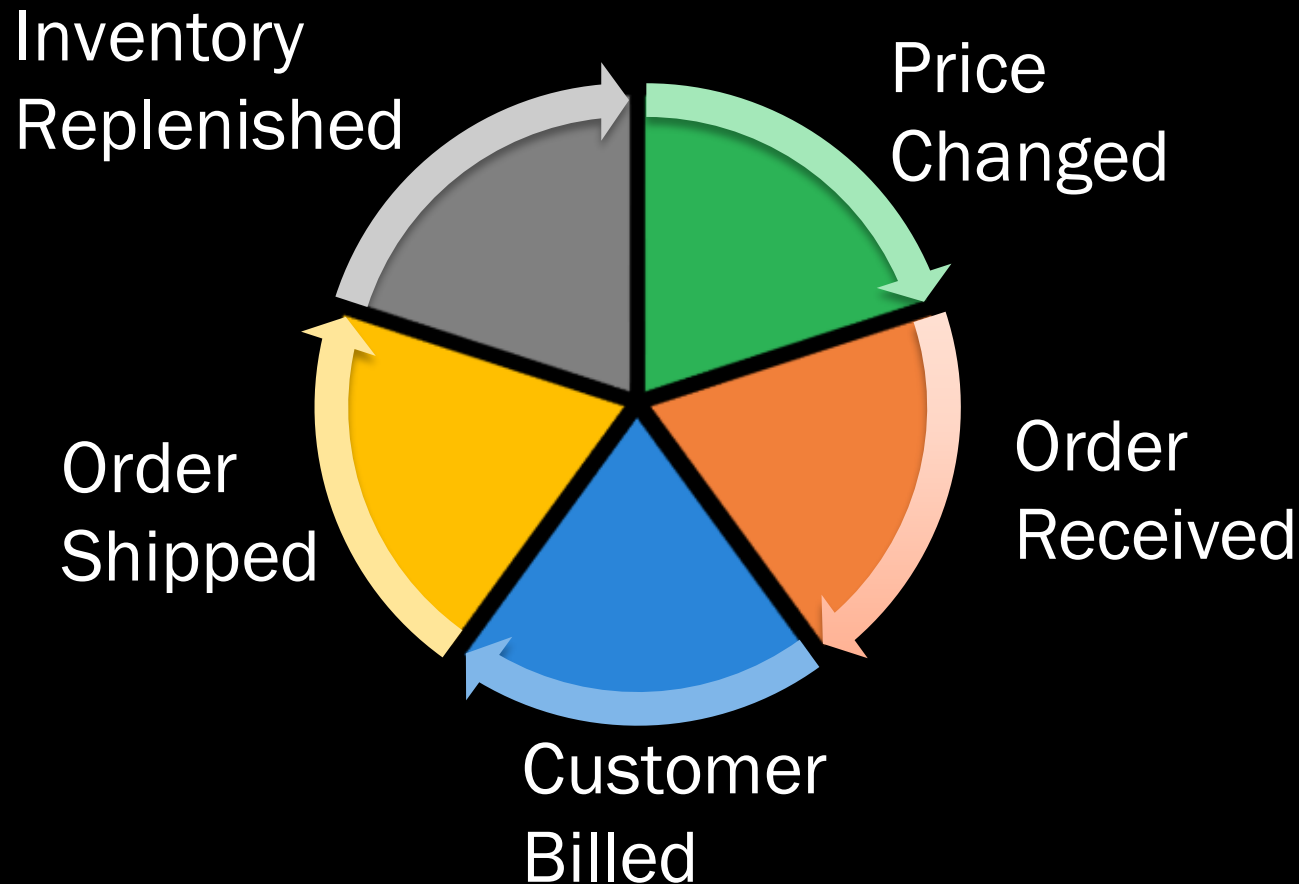
# SERVICE EXAMPLES



# SERVICE EVENT LIFECYCLES

Business processes remain within services

Cascading events give rise to enterprise processes



# WHICH SERVICE OWNS THIS PAGE?



Hello, BAT-SHEVA DAHAN. We have [recommendations](#) for you. ([Not BAT-SHEVA?](#))

[BAT-SHEVA's Amazon.com](#) |  [Today's Deals](#) | [Gifts & Wish Lists](#) | [Gift Cards](#)

[Shop All Departments](#)  Search

[Books](#) | [Advanced Search](#) | [Browse Subjects](#) | [New Releases](#) | [Bestsellers](#) | [The New York Times](#)



[Click to LOOK INSIDE!](#)

**PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE**

MARTIN FOWLER  
With introductions by:  
Dennis Flack,  
Marcelo Francisco,  
Eugene Hurley,  
Robert Mills, and  
Robert Szymanski

**Patterns of Enterprise Application Architecture (Hardcover)**  
~ [Martin Fowler](#) (Author)  
★★★★★  ([65 customer reviews](#))

---

List Price: ~~\$69.99~~  
Price: **\$45.87** & this item ships for **FREE with Super Saver Shipping**. [Details](#)  
You Save: **\$24.12 (34%)**

**In Stock.**  
Ships from and sold by **Amazon.com**. Gift-wrap available.

**Want it delivered Thursday, April 8?** Order it in the next **14 hours and 39 minutes**, and choose **One-Day Shipping** at **\$3.99**.  
[32 new](#) from **\$36.96** | [18 used](#) from **\$36.64**

Formats	Amazon Price	New from	Used from
Kindle Edition	<b>\$43.28</b>	--	--
Hardcover	<b>\$45.87</b>	<b>\$36.96</b>	<b>\$36.64</b>

[Share your own customer images](#)  
[Look inside this book](#)

**Frequently Bought Together**

Customers buy this book with [Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions](#) by Gregor Hohpe



**Price For Both: \$87.79**

[Show availability and shipping details](#)

**Customers Who Bought This Item Also Bought**



[Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions](#)



[Domain-Driven Design: Tackling Complexity in the Design of Software](#)



[Refactoring: Improving the Design of Existing Code](#)



[Applying Domain-Driven Design and Event Storming](#)



[Microsoft .NET Architecting: Designing and Building Enterprise Applications](#)



[Design Patterns: Elements of Reusable Object-Oriented Software](#)

# WHICH SERVICE OWNS THIS PAGE?

amazon.com Hello, BAT-SHEVA DAHAN. We have [recommendations](#) for you. ([Not BAT-SHEVA?](#))  
BAT-SHEVA's Amazon.com | [Today's Deals](#) | [Gifts & Wish Lists](#) | [Gift Cards](#)

[Shop All Departments](#) Search Books

Books Advanced Search Browse Subjects New Releases Bestsellers The New York Times

Click to **LOOK INSIDE!**

**PATTERNS OF ENTERPRISE APPLICATION ARCHITECTURE (Hardcover)**  
~ [Martin Fowler](#) (Author)  
★★★★★ (65 customer reviews)

List Price: ~~\$69.99~~  
Price: **\$45.87** & this item ships for **FREE with Super Saver Shipping**. [Details](#)  
You Save: **\$24.12 (34%)**  
**In Stock**

**None**

**Price For Both: \$87.79**  
[Add both to Cart](#) [Add both to Wish List](#)  
[Show availability and shipping details](#)

**Customers Who Bought This Item Also Bought**

[Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions](#) by Gregor Hohpe  
[Domain-Driven Design: Tackling Complexity in the Product Domain](#) by Eric Evans  
[Refactoring: Improving the Design](#) by Martin Fowler  
[Applying Domain-Driven Design and Design Patterns](#) by Vaughn Vernon  
[Microsoft .NET Architecting Applications for the Enterprise](#) by Jeffrey Richter  
[Design Patterns: Elements of Reusable Object-Oriented Software](#) by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

# SERVICE DEPLOYMENTS

- Many services can be deployed to the same box
- Many services can be deployed in the same app
- Many services can cooperate in a workflow
- Many services can be mashed up  
in the same page

# SAME PAGE COMPOSITION



**Product Catalog**

**Pricing**

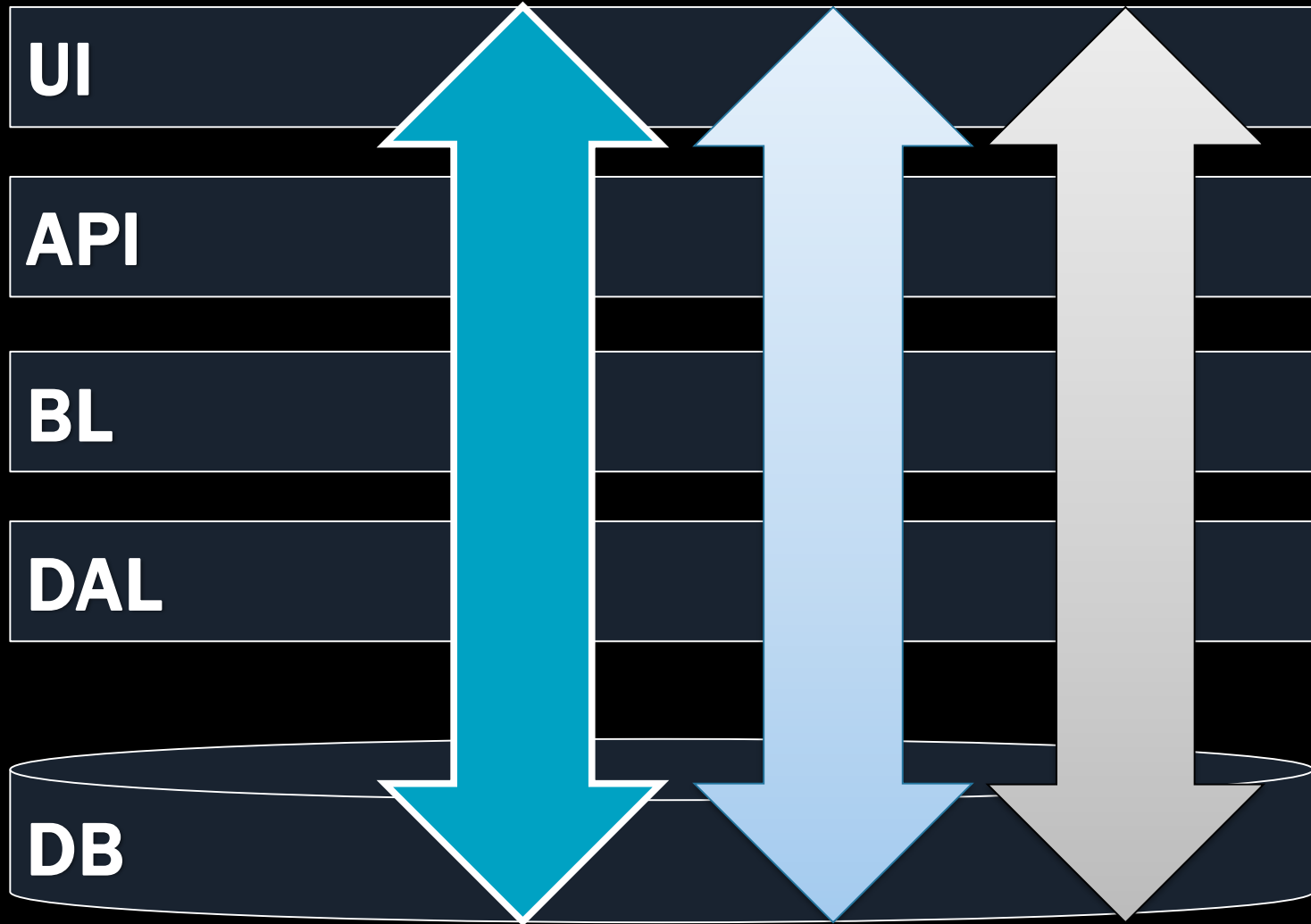
**Inventory**

**Cross Sell**

**Server**



# TOP-TO-BOTTOM SERVICES



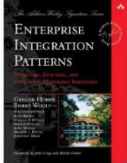

# DEMO






## ASP.NET MVC CompositeUI

[bit.ly/particular-microservices](http://bit.ly/particular-microservices)

# HOW TO MAKE A GRID

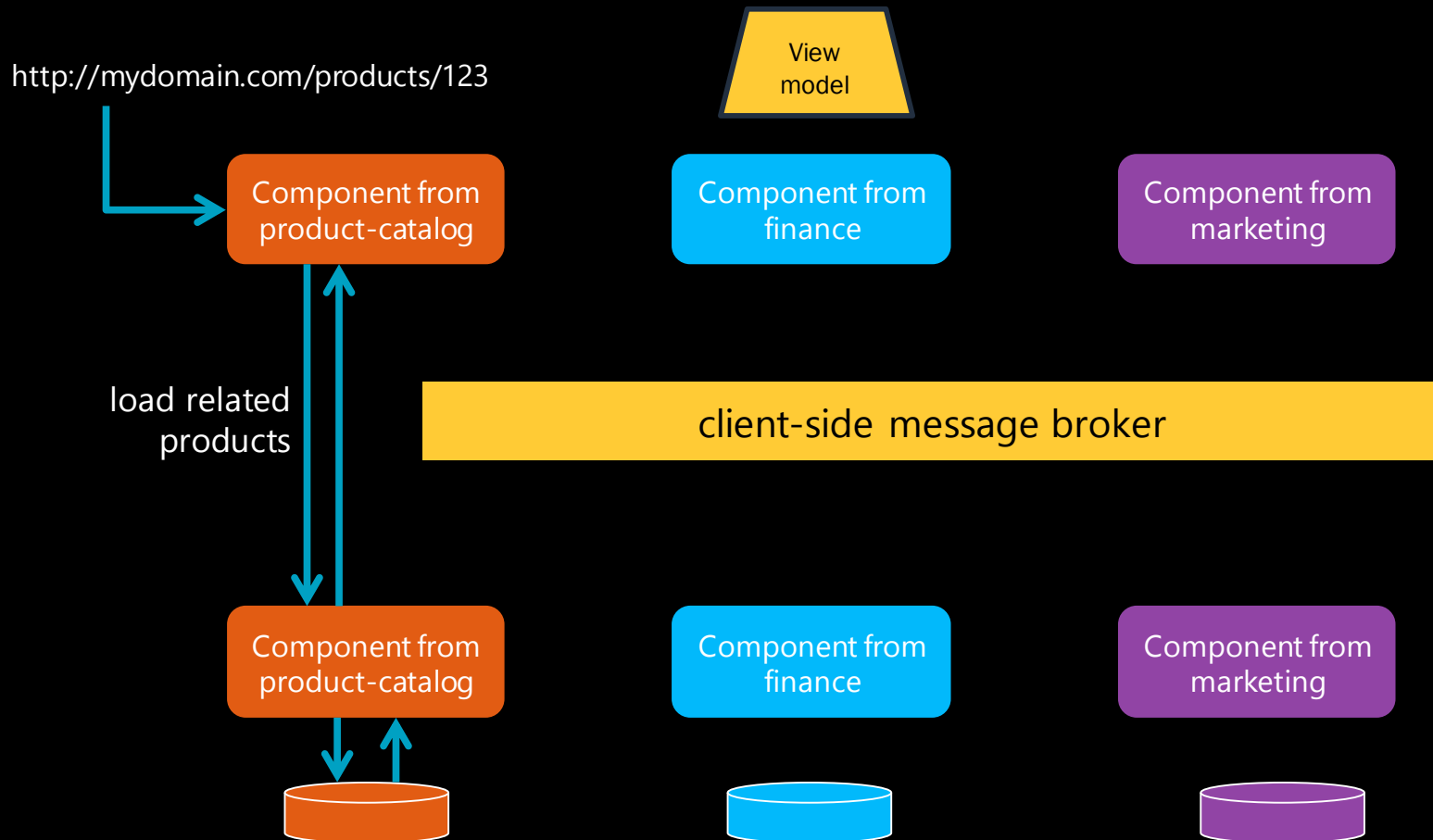
Customers Who Bought This Item Also Bought Page 1 of 17



<p>Enterprise Integration Patterns: Designing, Building, and Deploying...</p> <p>› Gregor Hohpe</p> <p>★★★★★ 78</p> <p>Hardcover</p> <p>\$44.39 </p>	<p>Refactoring: Improving the Design of Existing Code</p> <p>› Martin Fowler</p> <p>★★★★★ 204</p> <p>Hardcover</p> <p>\$40.68 </p>	<p>Domain-Driven Design: Tackling Complexity in the Heart of Software</p> <p>› Eric Evans</p> <p>★★★★★ 99</p> <p>Hardcover</p> <p>\$50.48 </p>	<p>Design Patterns: Elements of Reusable Object-Oriented Software</p> <p>› Erich Gamma</p> <p>★★★★★ 467</p> <p><b>#1 Best Seller</b> in Computer Vision &amp; Pattern...</p> <p>Hardcover</p> <p>\$37.48 </p>	<p>Clean Code: A Handbook of Agile Software Craftsmanship</p> <p>› Robert C. Martin</p> <p>★★★★★ 436</p> <p><b>#1 Best Seller</b> in Software Testing</p> <p>Paperback</p> <p>\$39.97 </p>	<p>Service Design Patterns: Fundamental Design Solutions for SOAP...</p> <p>› Robert Daigneau</p> <p>★★★★★ 32</p> <p>Hardcover</p> <p>\$46.39 </p>
---	---	---	--	---	---

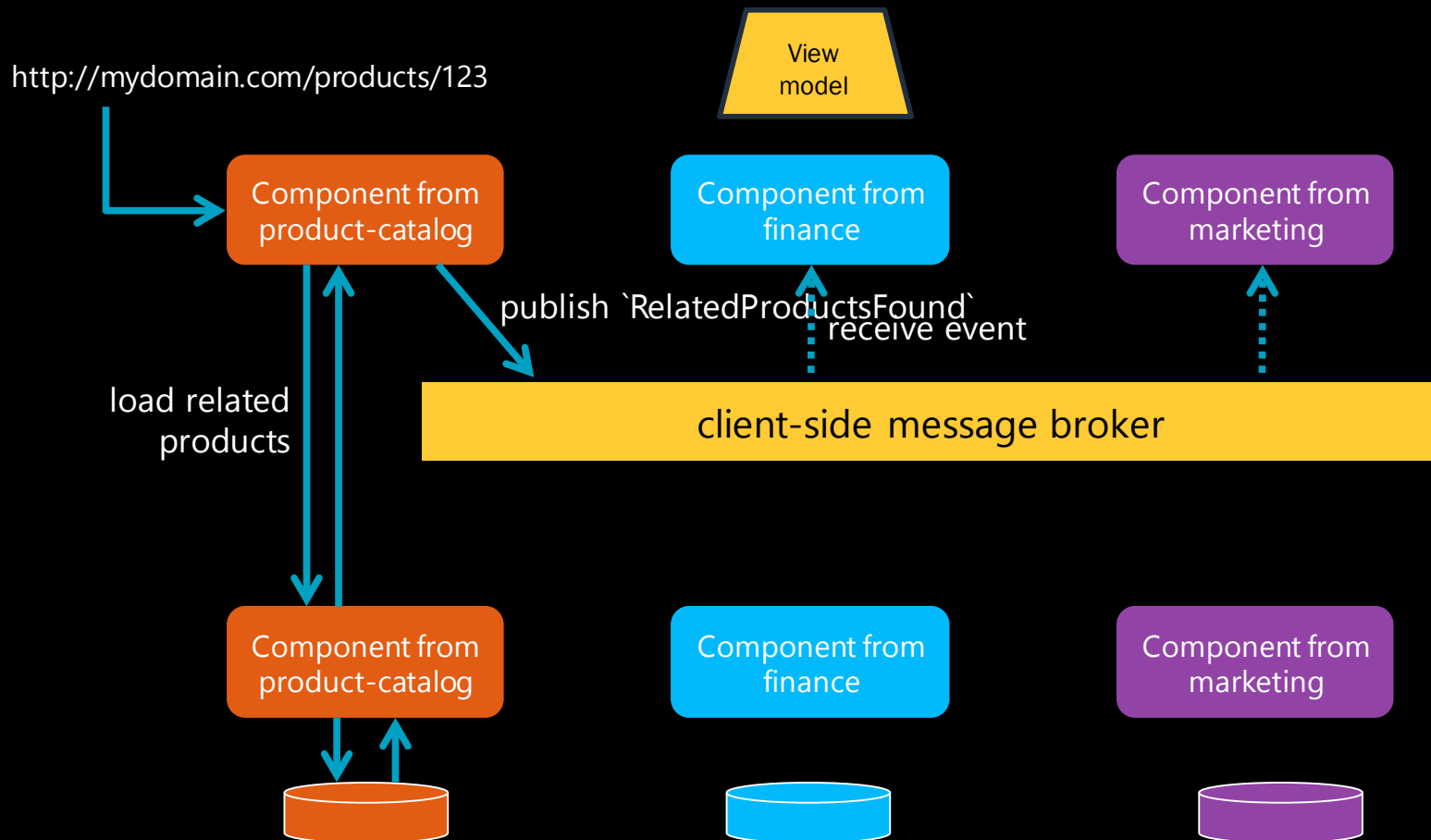
# HOW TO MAKE A GRID

ProductNameA	ProductNameB	ProductNameC	ProductNameD
<div>cover image A</div>	<div>cover image B</div>	<div>cover image C</div>	<div>cover image D</div>
AuthorNameA ★★★★★	AuthorNameB ★★★★★	AuthorNameC ★★★★★	AuthorNameD ★★★★★
€ 20.00	€ 20.00	€ 20.00	€ 20.00



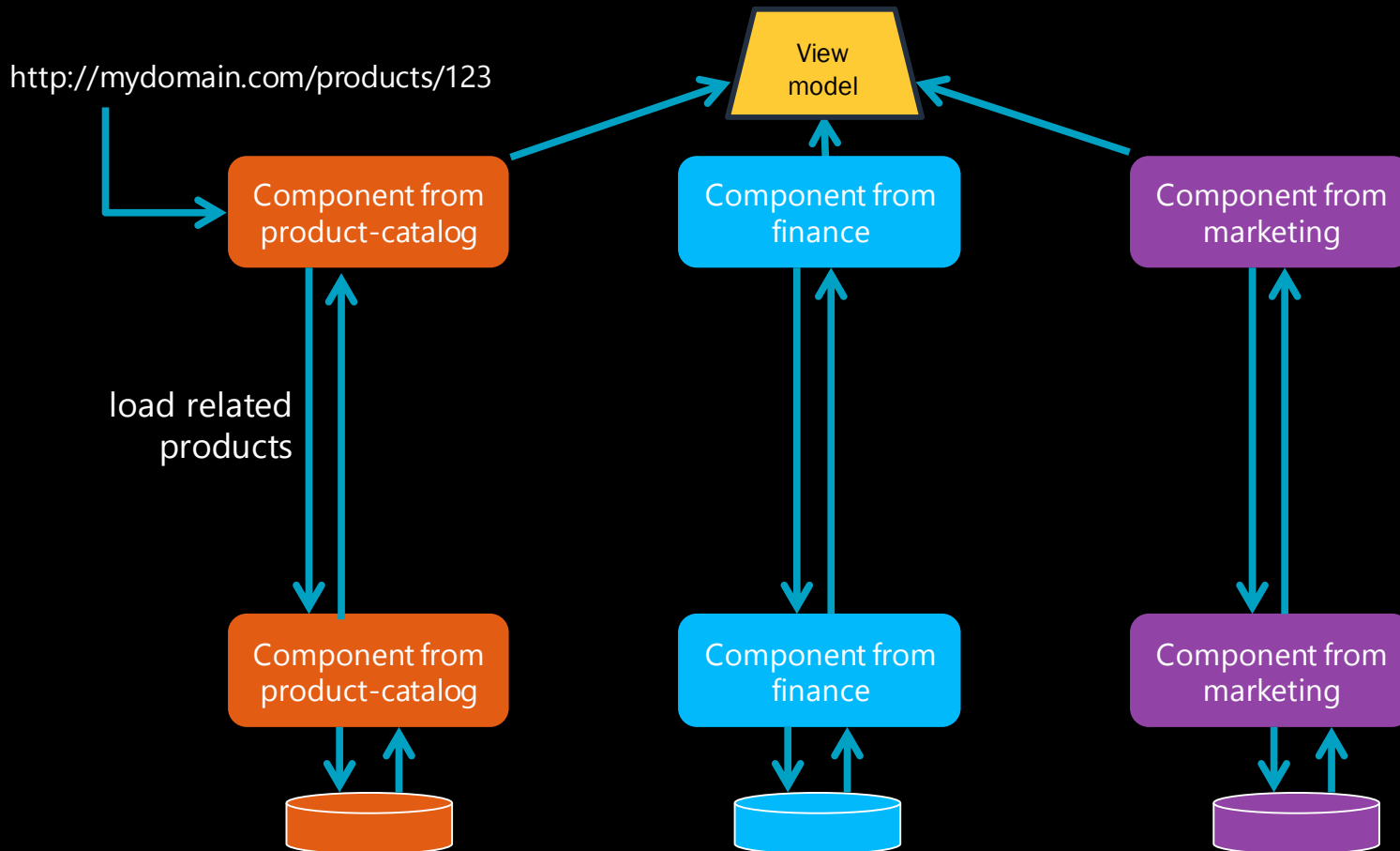
# HOW TO MAKE A GRID

ProductNameA cover image A AuthorNameA ★★★★★ € 20.00	ProductNameB cover image B AuthorNameB ★★★★★ € 20.00	ProductNameC cover image C AuthorNameC ★★★★★ € 20.00	ProductNameD cover image D AuthorNameD ★★★★★ € 20.00
--	--	--	--



# HOW TO MAKE A GRID

ProductNameA	ProductNameB	ProductNameC	ProductNameD
<div>cover image</div> <div>A</div>	<div>cover image</div> <div>B</div>	<div>cover image</div> <div>C</div>	<div>cover image</div> <div>D</div>
AuthorNameA ★★★★★	AuthorNameB ★★★★★	AuthorNameC ★★★★★	AuthorNameD ★★★★★
€ 20.00	€ 20.00	€ 20.00	€ 20.00



# OTHER COMMON ELEMENTS

- Color scheme, layout, fonts, CSS, images, etc
- All communicate the “corporate brand”
- The responsibility of the “branding” service

# LAYOUT IN THE BRANDING SERVICE

- View Models created with Whatever.js  
Angular / React / Knockout / Backbone / etc
- Can also be done server-side
- Each service binds its model to part of the view model



# OR LEVERAGE CSS CLASSES

```
<div class="price">$49.99</div>
```

Makes it bold, red, large

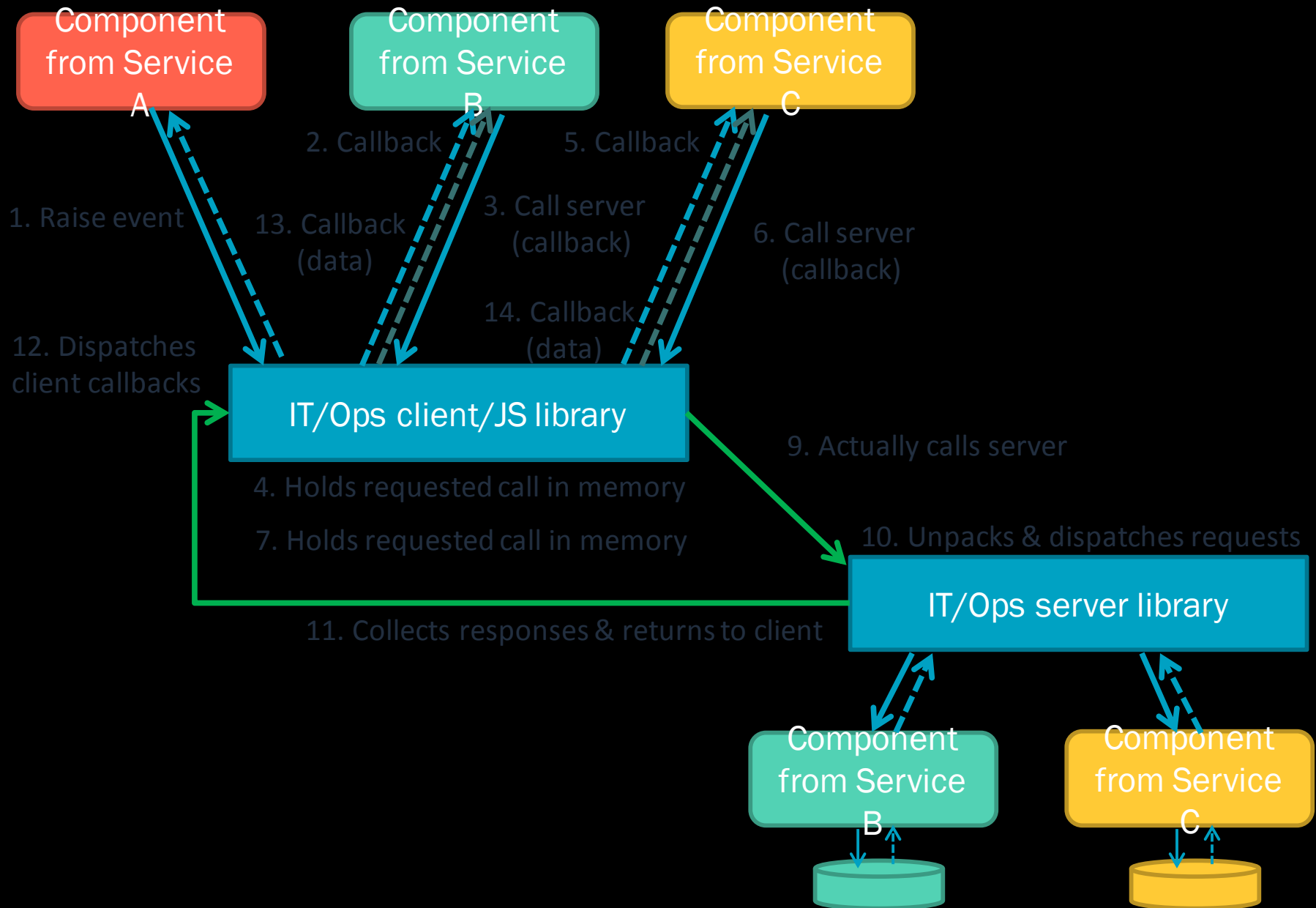
```
<div class="inventory.InStock">12 left</div>
```

Makes it bold and green

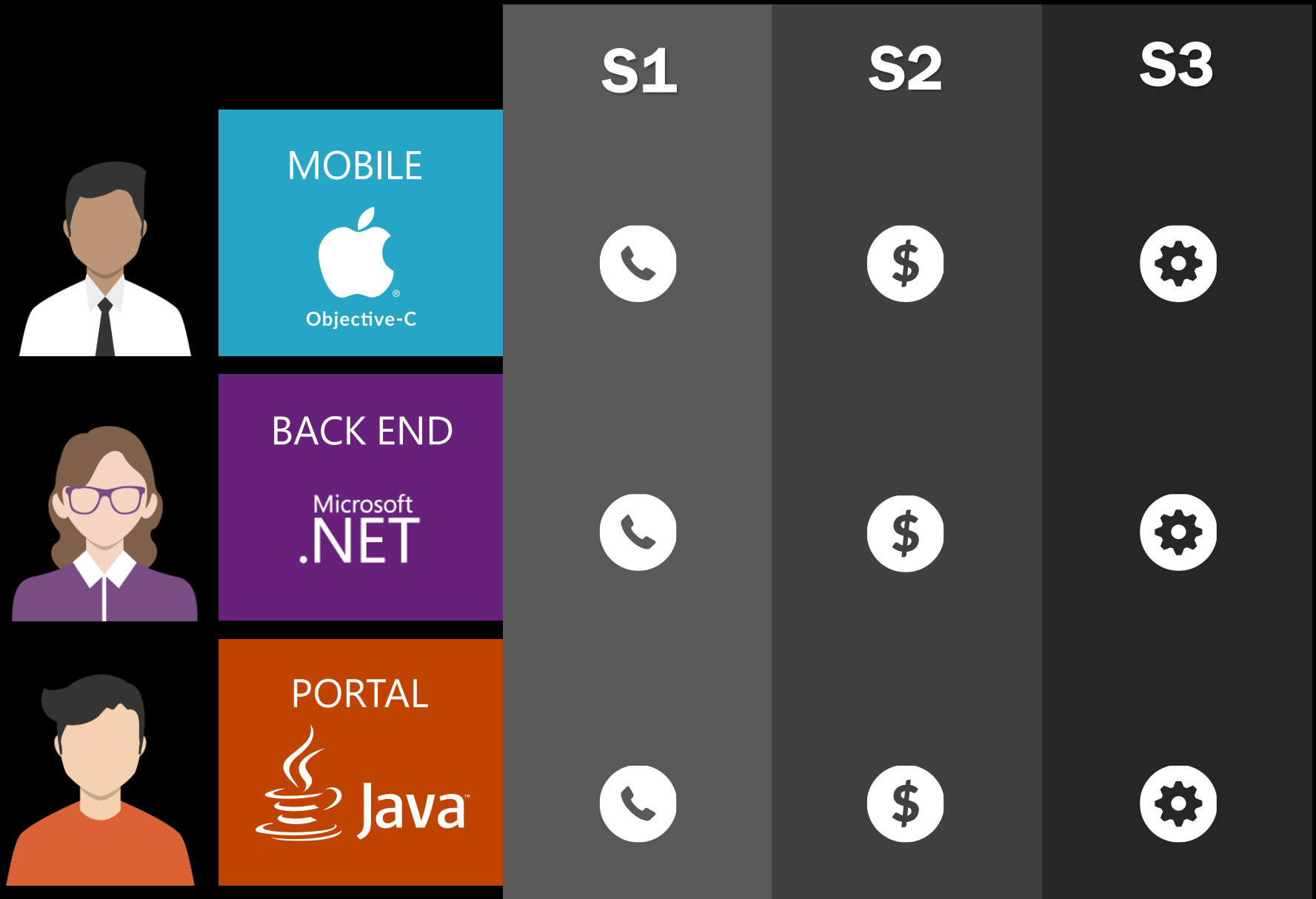
Go even farther with JS CSS preprocessors:

Mustache, LESS, Sass

# PERFORMANCE OPTIMIZATION



# ACROSS THE ENTERPRISE SERVICES



# CONFIGURATION MANAGEMENT

## an example



<http://go.particular.net/octopus>

<http://go.particular.net/octopus-script>

# AMAZON.COM CHECKOUT WORKFLOW

## Select a shipping address

Is the address you'd like to use displayed below? If so, click

### Udi Dahan

20 Uri Tzvi Greenberg street  
Haifa, 34757  
Israel  
Phone: +972-522888426

Ship to this address

Edit

Delete

amazon.com

SIGN IN

SHIPPING & PAYMENT

GIFT OPTIONS

PLACE ORDER



## Select a payment method

### Your credit and debit cards



MasterCard ending in

Name on card

Udi Dahan

Expires on



Continue

You can review this order before it's final.

Estimated delivery: Jan. 12, 2017 - Jan. 19, 2017



### Patterns of Enterprise Application Architecture

by Martin Fowler

**\$43.22**

Quantity: 1 [Change](#)

Sold by: Amazon Export Sales LLC



Add a gift receipt

and see other gift options

### Choose a delivery option:



**averages 9-14 business days**

\$7.98 - AmazonGlobal Standard Shipping



**averages 7-10 business days**

\$20.98 - AmazonGlobal Expedited Shipping



**averages 2-5 business days**

\$49.98 - AmazonGlobal Priority Shipping

Place your order in EUR

By placing your order, you agree to Amazon's [privacy notice](#) and [conditions of use](#).

Amazon Currency Converter is Enabled. [Learn more](#)

### Order Summary

Items: EUR 43,19

Shipping & handling: EUR 7,98

Total before tax: EUR 51,17

Estimated tax to be collected: EUR 0,00

**Order total: EUR 51,17**

### Selected payment currency

☒ EUR ☐ USD

[\(Change card currency\)](#)

### Applicable Exchange Rate

1 USD = 0.9994103744 EUR

(includes all Amazon fees and charges)

Please note that your country may charge import duties, taxes and fees that you may have to pay ahead of delivery. [Learn more](#)

[How are shipping costs calculated?](#)

[Why didn't I qualify for free shipping?](#)

# WHICH SERVICE OWNS THIS FLOW?

## Select a shipping address

Is the address you'd like to use displayed below? If so, click

### Udi Dahan

20 Uri Tzvi Greenberg street  
Haifa, 34757  
Israel  
Phone: +972-522888426

Ship to this address

Edit

Delete

amazon.com

SIGN IN

SHIPPING & PAYMENT

GIFT OPTIONS

PLACE ORDER



## Select a payment method

### Your credit and debit cards

Name on card

Expires on



MasterCard ending in

Udi Dahan

Continue

You can review this order before it's final.

Estimated delivery: Jan. 12, 2017 - Jan. 19, 2017



### Patterns of Enterprise Application Architecture

by Martin Fowler

**\$43.22**

Quantity: 1 [Change](#)

Sold by: Amazon Export Sales LLC



Add a gift receipt

and see other gift options

### Choose a delivery option:

- ☒ **averages 9-14 business days**  
\$7.98 - AmazonGlobal Standard Shipping
- ☐ **averages 7-10 business days**  
\$20.98 - AmazonGlobal Expedited Shipping
- ☐ **averages 2-5 business days**  
\$49.98 - AmazonGlobal Priority Shipping

Place your order in EUR

By placing your order, you agree to Amazon's [privacy notice](#) and [conditions of use](#).

Amazon Currency Converter is Enabled. [Learn more](#)

### Order Summary

Items:	EUR 43,19
Shipping & handling:	EUR 7,98
Total before tax:	EUR 51,17
Estimated tax to be collected:	EUR 0,00

**Order total: EUR 51,17**

### Selected payment currency

☒ EUR ☐ USD

[\(Change card currency\)](#)

### Applicable Exchange Rate

1 USD = 0.9994103744 EUR

(includes all Amazon fees and charges)

Please note that your country may charge import duties, taxes and fees that you may have to pay ahead of delivery. [Learn more](#)

[How are shipping costs calculated?](#)

[Why didn't I qualify for free shipping?](#)

# WHICH SERVICE OWNS THIS FLOW?

## Select a shipping address

Is the address you'd like to use displayed below? If so, click

### Udi Dahan

20 Uri Tzvi Greenberg street  
Haifa, 34757  
Israel  
Phone: +972-522888426

amazon.com

SIGN IN

SHIPPING & PAYMENT

GIFT OPTIONS

PLACE ORDER

Place your order in EUR

By placing your order, you agree to Amazon's  
[privacy notice](#) and [conditions of use](#).

Amazon Currency Converter is

# None

Estimated delivery: Jan. 12, 2017 - Jan. 19, 2017



### Patterns of Enterprise Application Architecture

by Martin Fowler

**\$43.22**

Quantity: 1 [Change](#)

Sold by: Amazon Export Sales LLC

Add a gift receipt

and see other gift options

### Choose a delivery option:

- ☒ **averages 9-14 business days**  
\$7.98 - AmazonGlobal Standard Shipping
- ☐ **averages 7-10 business days**  
\$20.98 - AmazonGlobal Expedited Shipping
- ☐ **averages 2-5 business days**  
\$49.98 - AmazonGlobal Priority Shipping

☒ EUR ☐ USD  
([Change card currency](#))

### ▼ Applicable Exchange Rate

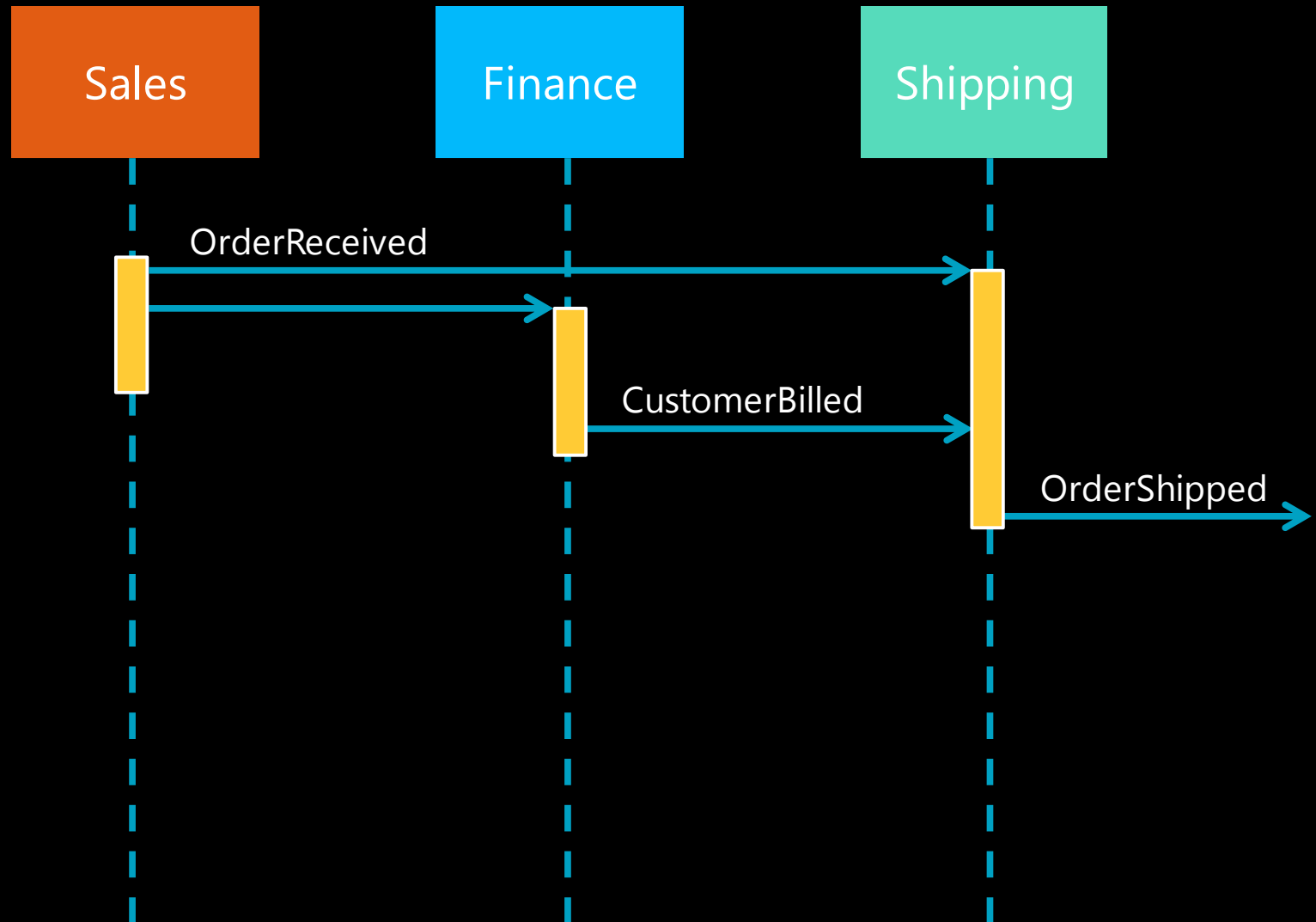
1 USD = 0.9994103744 EUR  
(includes all Amazon fees and charges)

Please note that your country may charge import duties, taxes and fees that you may have to pay ahead of delivery. [Learn more](#)

[How are shipping costs calculated?](#)

[Why didn't I qualify for free shipping?](#)

# WORKFLOW EVENT COMPOSITION





# WORKFLOW EVENT COMPOSITION

The screenshot shows the Amazon Shopping Cart interface. At the top, the Amazon logo and navigation links are visible. The cart contains one item: "Patterns of Enterprise Application Architecture" by Martin Fowler, priced at \$50.04. The subtotal is also \$50.04. A red box highlights the "Proceed to checkout" button, and an orange arrow points to it from the SetOrderId text below. The page also displays shipping information, a sign-in prompt, and a recommendation for another book, "Design Patterns" by Erich Gamma.


amazon  
Try Prime

All ▾

Departments ▾ Browsing History ▾ Dennis's Amazon.com Today's Deals Gift Cards & Registry Sell Help

Hello, Dennis  
Account & Lists ▾ Orders Try Prime ▾ Cart

### Shopping Cart

	Price	Quantity
 <p><b>Patterns of Enterprise Application Architecture</b> by Martin Fowler Hardcover In Stock Eligible for FREE Shipping <input type="checkbox"/> This is a gift <a href="#">Learn more</a> <a href="#">Delete</a> <a href="#">Save for later</a></p>	<b>\$50.04</b>	1 ▾

**Subtotal (1 item): \$50.04**

The price and availability of items at Amazon.com are subject to change. The Cart is a temporary place to store a list of your items and reflects each item's most recent price.  
[Learn more](#)  
Do you have a gift card or promotional code? We'll ask you to enter your claim code when it's time to pay.

✓ Your order qualifies for FREE Shipping  
Choose this option at checkout. [See details](#)


**Subtotal (1 item): \$50.04**  
☐ This order contains a gift

**Proceed to checkout**

or  
[Sign in to turn on 1-Click ordering.](#)

Estimate your shipping and tax ▾

Customers Who Shopped for Patterns of Enterprise Application Architecture Also Shopped for



**Design Patterns:...**  
Erich Gamma  
★★★★☆ 469  
Hardcover  
\$37.48 ✓ Prime  
[Add to Cart](#)

SetOrderId = 022032ba-1337-43a5-90c9-d48b58742c7

# WORKFLOW COMPOSITION

## Shipping

Is the address you'd like to use displayed below? If so, click the address to select it.

### Udi Dahan

20 Uri Tzvi Greenberg street  
Haifa, 34757  
Israel  
Phone: +972-522888426

Ship to this address

Edit

Delete

amazon.com

SIGN IN

SHIPPING & PAYMENT

GIFT OPTIONS

PLACE ORDER



## Sales

Place your order in EUR

By placing your order, you agree to Amazon's privacy notice and conditions of use.

Amazon Currency Converter is Enabled. [Learn more](#)

## Billing

### Order Summary

Subtotal EUR 43,19  
Shipping & handling EUR 7,98

Total before tax: EUR 51,17  
Estimated tax to be collected: EUR 0,00

**Order total: EUR 51,17**

### Selected payment currency

☒ EUR ☐ USD  
(Change card currency)

### Applicable Exchange Rate

1 USD = 0.9994103744 EUR  
(Includes all Amazon fees and charges)

Please note that your country may charge import duties, taxes and fees that you may have to pay ahead of delivery. [Learn more](#)

[How are shipping costs calculated?](#)

[Why didn't I qualify for free shipping?](#)

## Billing

Your credit and debit cards

☒ MasterCard ending in:

Name on card

Udi Dahan

Expires on

Continue

You can review this order before it's final.

## Sales



**Patterns of Enterprise Application Architecture**  
by Martin Fowler  
**\$43.22**

Quantity: 1 [Change](#)

Sold by: Amazon Export Sales LLC

Add a gift receipt

and see other gift options

## Shipping

Choose a delivery option.

- ☒ **averages 9-14 business days**  
\$7.98 - AmazonGlobal Standard Shipping
- ☐ **averages 7-10 business days**  
\$20.98 - AmazonGlobal Expedited Shipping
- ☐ **averages 2-5 business days**  
\$49.98 - AmazonGlobal Priority Shipping

# PARENT VS CHILDREN

Orders table

Id	ShippingId	FinanceId	Etc
123	1337	42	...
124	1338	43	...

VS

Sales

Orders table

Id	ProductId
123	ABC
124	ACD

Shipping

Shipping table

OrderId	Address
123	Haifa, Israel
124	Rotterdam, Holland

Finance

Invoices table

OrderId	Status
123	Paid
124	Overdue

# PARENT VS CHILDREN

## Orders table

Id	ShippingId	FinanceId	Etc
123	1337	42	...
124	1338	43	...

VS

## Sales

### Orders table

Id	ProductId
123	ABC
124	ACD

## Shipping

### Shipping table

OrderId	Address
123	Haifa, Israel
124	Rotterdam, Holland

### eBookShipping table

OrderId	UrlIdentifier
1234	03630b562df15c6
1235	4c77a8e12cb1c216

## Finance

### Invoices table

OrderId	Status
123	Paid
124	Overdue

### BitcoinInvoices table

OrderId	BitcoinId
4242	1d7e565784907
4243	6561433f9245710



# WHAT ABOUT THE TECHNICAL STUFF?

- Authentication, Authorization, etc
- A service should own this  
shouldn't it?

# THE IT/OPERATIONS SERVICE

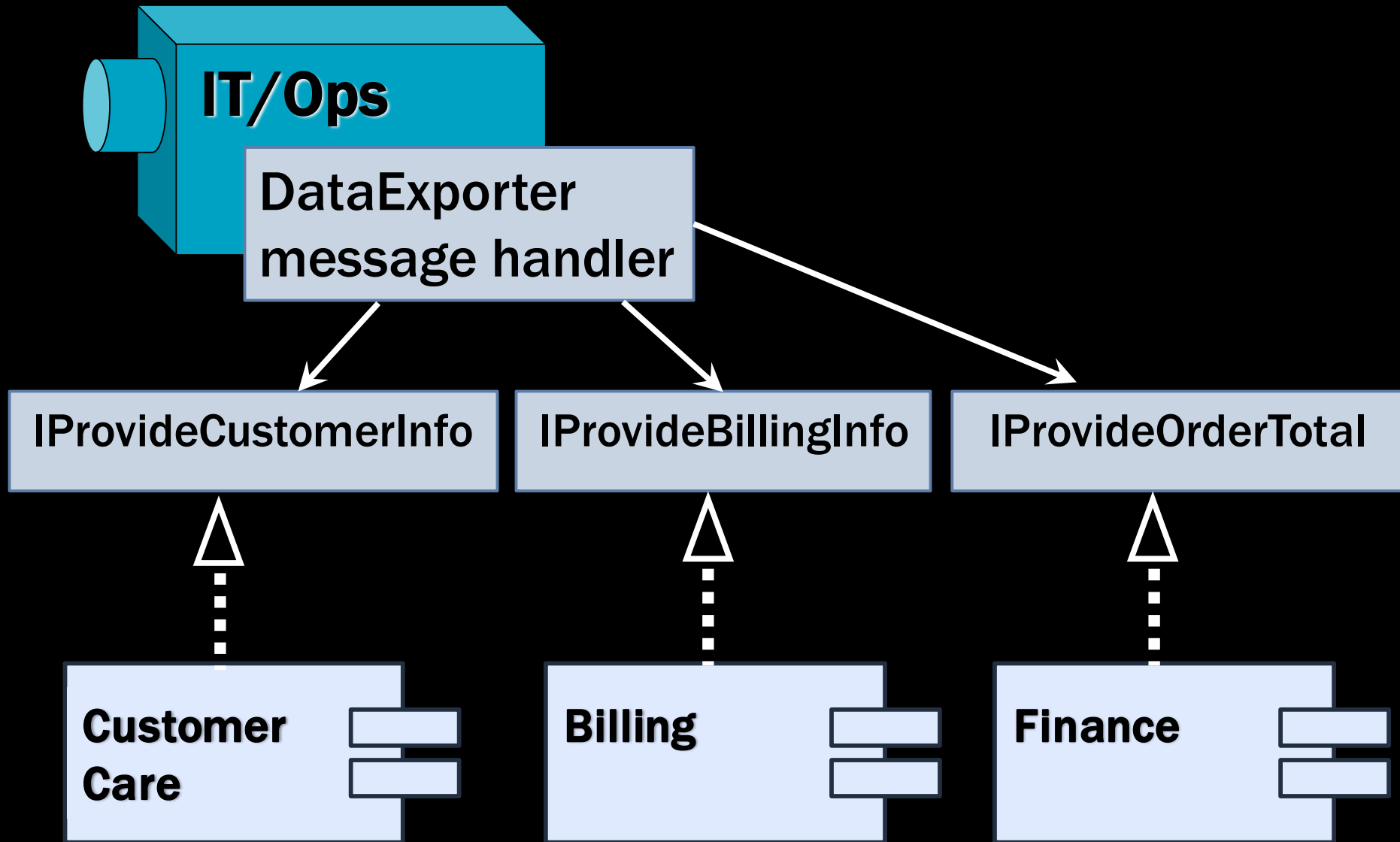
- Responsible for keeping information flowing (and secure) in the enterprise
- Focused on the deployment & physical views:  
Responsible for hosting (web servers, DBs, etc)  
Owns connection strings, queue names

Authentication, authorization – LDAP/AD access

# IT/OPS HARDLY EVER SUBSCRIBES

- Doesn't really do pub/sub with other services
- One (rare) case  
HR – employee hired, employee fired  
Provisions / De-provisions machines, accounts, etc  
Don't introduce this unless necessary

# INTEGRATION ENDPOINTS ( + EMAIL )





# REPORTING – CAN BE HARDER

- But not that different from Composite UI grids
- But there is a better way:
- Regular reports:
  - Find the pattern users are looking for in the report
  - Model constants as domain concerns
  - Implement as event-correlation & send email
- Research / Data Science

# REFERENTIAL INTEGRITY

- Inserting “children” without a “parent” across service boundaries

Solved by eventual consistency

- Deleting – but not “cascading”

Private data – wholly inside a service. OK

Public data – shared between services. Not OK.

Eg. Deleting a product. What about orders, inventory?

HOW DO YOU  
FIND THE BOUNDARIES?

# DECOMPOSING A DOMAIN

```
public class Customer
{
    FirstName
    LastName
    Status
    // etc
}
```

```
public class Product
{
    Name
    Description
    Price
    // etc
}
```

Customer  
FirstName  
LastName

Customer  
Status

Product  
Price

Product  
Name  
Description

# HEALTHCARE

## Dentistry Service

PatientId  
Orthodontical data  
Medication

## Psychology Service

## General Practitioner

Basic patient information  
Basic doctor visits

## Surgery Service

PatientId  
Laboratory data  
Allergies information  
Medication

VS

## Doctors Service

## Patient Service

## Medication Service

## Allergies Service

# INSURANCE

## Home Insurance

Home owner  
Coverage  
PolicyId  
ClaimId

## Travel Insurance

Insured persons  
Coverage  
PolicyId  
ClaimId

## Automotive Insurance

Insured vehicle  
Coverage  
PolicyId  
ClaimId

VS

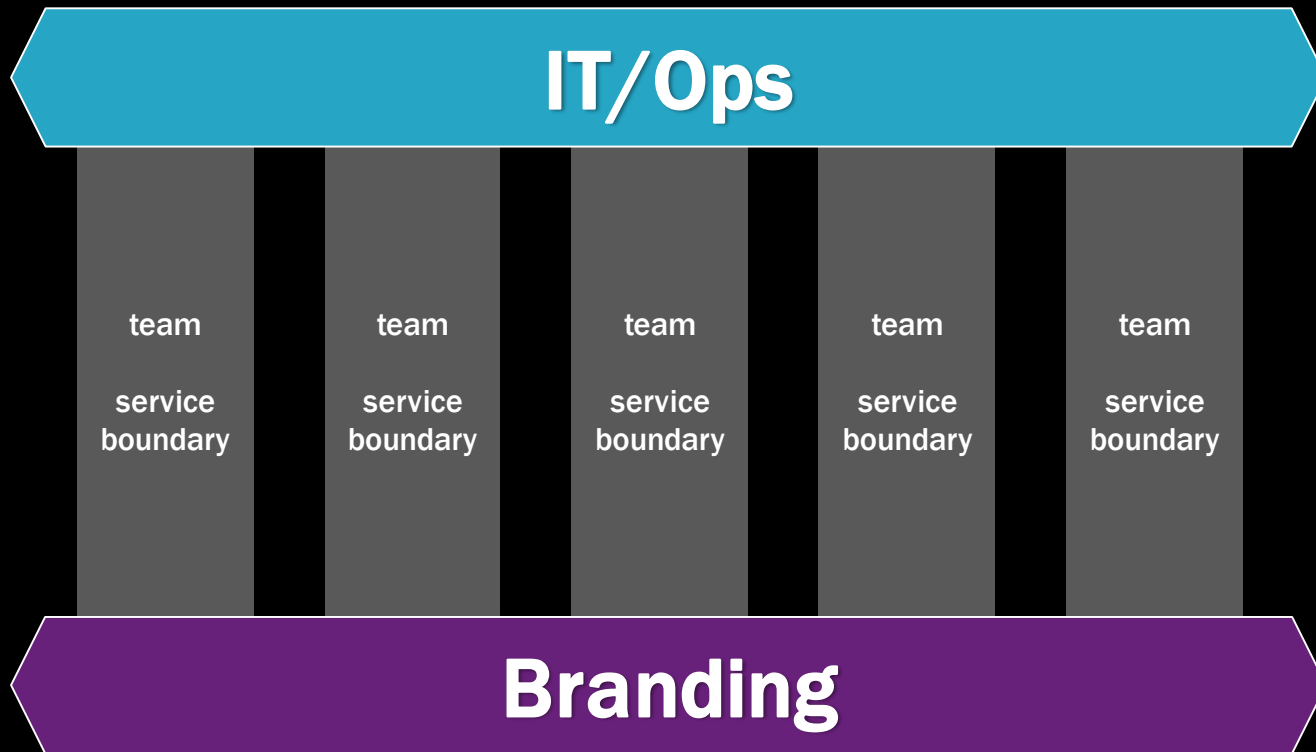
## Policy Service

Home insurance  
Travel insurance  
Automotive insurance

## Claims Service

Home insurance  
Travel insurance  
Automotive insurance

# TEAM STRUCTURE FOR SOA




**\* Task-forces as a new/alternate model**

# WHEN NOT TO DO SOA

- Startups
- Generic / extensible “platforms”



# SOA HOMEWORK



Sydney Harbour Marriott Hotel at Circular Quay

30 Pitt Street Sydney New South Wales 2000 Australia

185 Reviews +61-2-9259 7000 Photos


Your Stay

Sat, 17 Sep, 2016 - Fri, 23 Sep, 2016

1 Rooms, 1 Guests

EDIT STAY

CONTINUE



VIEW PHOTOS

Our Hotel

Photos

Rooms

Deals

Dining

Fitness

Local Area

Map

Meetings

Weddings

# YOUR MISSION

- Identify the service boundaries
- What data does each service own?
- What part of which UI does it own?
- What events does it publish / subscribe to?

# THE USE CASES:

- For a single hotel
- With only 1 guest per reservation  
1 room per reservation
- No loyalty program / sign-in

# ONLINE #1

- Search for availability

Check Rates & Availability

Sat, 17 Sep, 2016

Fri, 23 Sep, 2016

Special Rates

☐ My dates are flexible

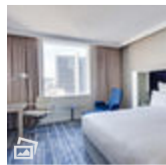
1 Room

1 Guest/room

☐ Use Points

What's this?

VIEW RATES



### City View, Guest room, 1 King, Skyline view

The rate for this room changes on 18/09/16 to 273 (AUD) per night

[Rate details](#) • [Room details](#)

**398** AUD/night

Select



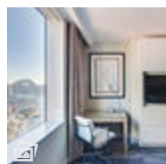
### Premium City View, Guest room, 1 King, Skyline view

The rate for this room changes on 18/09/16 to 293 (AUD) per night

[Rate details](#) • [Room details](#)

**417** AUD/night

Select



### Bridge view, Guest room, 1 King or 2 Double, Partial Bridge view

The rate for this room changes on 18/09/16 to 303 (AUD) per night

[Rate details](#) • [Room details](#)

**427** AUD/night

Select


# ONLINE #2

## ▪ Make a booking

### Review Reservation Details

#### 1. Your selection

**Sydney Harbour Marriott Hotel At Circular Quay**  
30 Pitt Street Sydney, 2000 Australia





**Check in:** Saturday, 17 September 2016  
**Check out:** Friday, 23 September 2016


**Room(s):** 1  
**Guest(s) per room:** 1  
**Room type(s):** City View, Guest room, 1 King, Skyline view  
[Edit](#) - [Room details](#)


#### 2. Your requests


Make requests for accessibility, early check-in and more.  
Another benefit of booking direct on Marriott.com


 Accessibility

 Early check-in

 Extra towels

 Rollaway/crib

 Feather-free room

 Room location

[Make Request](#)

#### 3. Summary of Charges

1 room(s) for 6 night(s)	Prices in AUD
Saturday, 17 September 2016	398.00
Sunday, 18 September 2016	273.00
Monday, 19 September 2016	273.00
Tuesday, 20 September 2016	381.00
Wednesday, 21 September 2016	352.00
Thursday, 22 September 2016	342.00

Total cash rate2,019.00

[Total taxes and fees](#)Included

Total for stay in hotel's currency2,019.00 AUD


[FREE Cancellation](#) You may cancel your reservation for no charge until 16 September 2016 (1 day[s] before arrival). [Learn more](#)

#### 4. Confirm details

**About this reservation:**  
[FREE Cancellation](#) You may cancel your reservation for no charge until 16 September 2016 (1 day[s] before arrival).

Continue

Next, you'll provide your information



# ONLINE #2.1

- Make a booking
- Fill in guest info
- Fill in credit card

Reservation Step 2 of 3

Your Stay	Room(s)	Room Preferences
<a href="#">edit</a> Check in: Saturday, 17 September 2016 Check out: Friday, 23 September 2016 Rooms 1 Total guests: 1	<a href="#">edit</a> <a href="#">City View, Guest room, 1 King, Skyline view.</a> 1 night at 398.00 AUD, 2 nights at 273.00 AUD, 1 night at 381.00 AUD, 1 night at 352.00 AUD, 1 night at 342.00 AUD 2,019.00 AUD Total hotel currency (incl. est. taxes) <a href="#">Rate details</a>	<a href="#">edit</a> No room preferences were selected.

[Sign In for faster booking](#)

**Title**

**\* First name**

**\* Last name**

**\* Email address**

**Rewards number**

**Company name**

**Address**  
**\* Country**

**\* Address**

**\* City**

**State/Province**

**Post code**

[For travel agents and planners](#)

**Credit/Debit Card Information** [Why we ask for this?](#)

**Card holder name**

**\* Credit/Debit card number**

☒ No fees for using your credit card.

**\* Expiration Month**

**\* Expiration Year**

**Free Wi-Fi + mobile check-in + our lowest member rates all the time**  
☒ Instantly join Rewards and enjoy the perks with each stay.

**Password**

**Confirm Password**

☒ Remember me (recommended for private computers only). [What's This?](#)

☒ I would like to receive account updates, program news and offers via email and direct mail.

☐ I would like to receive exclusive offers from select third parties.

By signing up, I agree to Marriott's [Terms of Use](#), [Privacy Statement](#) and [Data Protection Clause](#).

[Book Now](#)

**\* Required**

**Make reservations by phone**  
Dial toll-free 1 800 251 259 in Australia, toll-free 0800 264 333 in New Zealand, or view our [regional telephone reservation numbers](#).

**Rest assured.**  
Your credit card is safe. State-of-the-art security. Another plus of booking on [Marriott.com.au](#)

# ONLINE #2.2

- Make a booking – additional requirements:
- First, authorize credit card for cancellation \$\$
- If successful, see if a room is still available  
If not, release authorization, tell user “no room”
- Although email confirmation should be sent, result of booking should be on next screen

# ONLINE #2.2

- Make a booking – important domain info:
- Room number not allocated at time of booking
- Rooms not “locked” while booking in-process
- Capacity MUST be respected  
Can't have 50 people all book the last room



# FRONT DESK #3

- Check-in  
Find booking  
by last name

Verify info

Authorize \$\$  
for full stay

Allocate room



# FRONT DESK #4

- Check-out  
Night before:  
print out bill  
  
Guest leaves room  
Person verifies  
  
Card is charged  
amount authorized

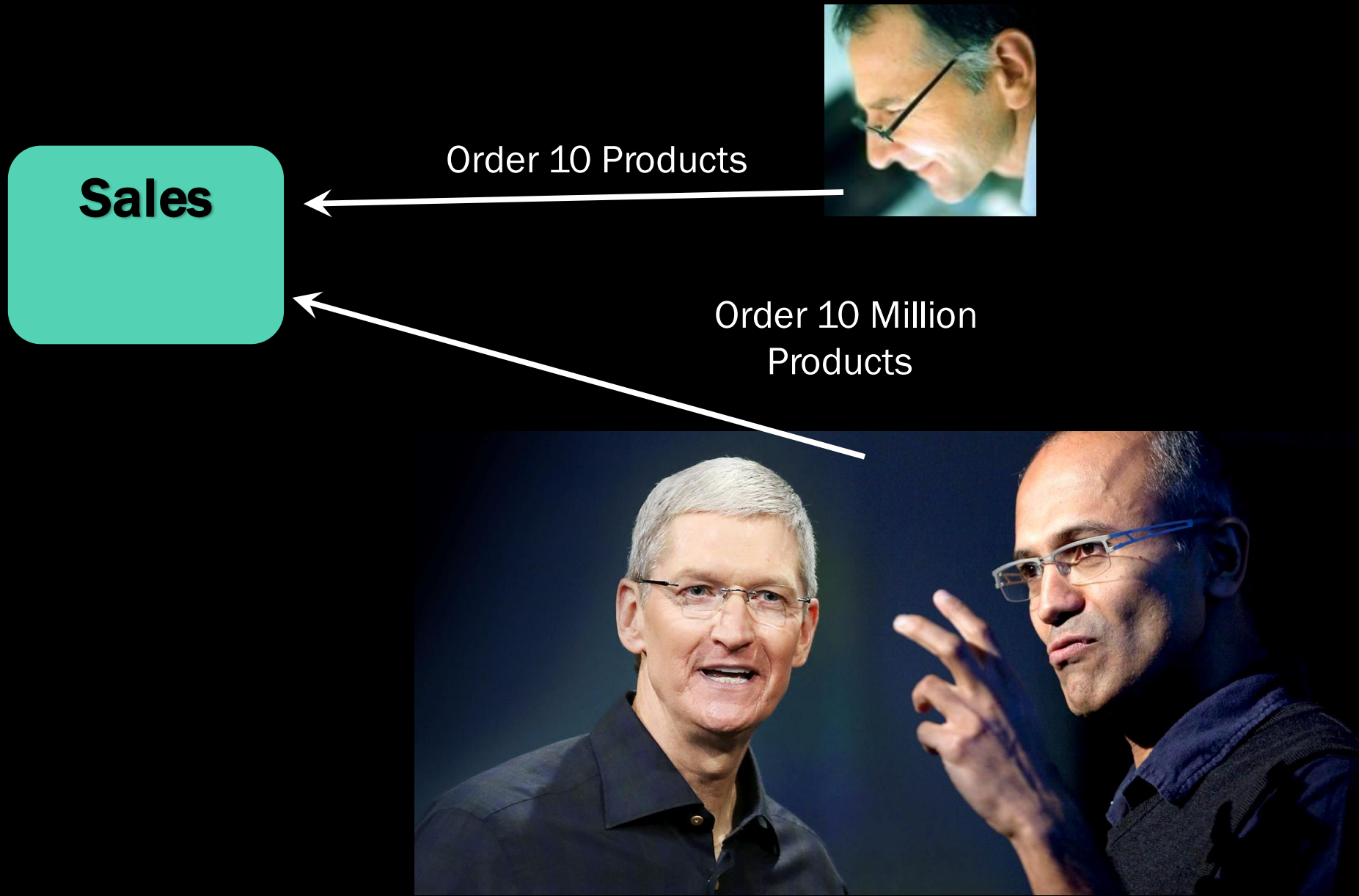


# BUSINESS & AUTONOMOUS COMPONENTS

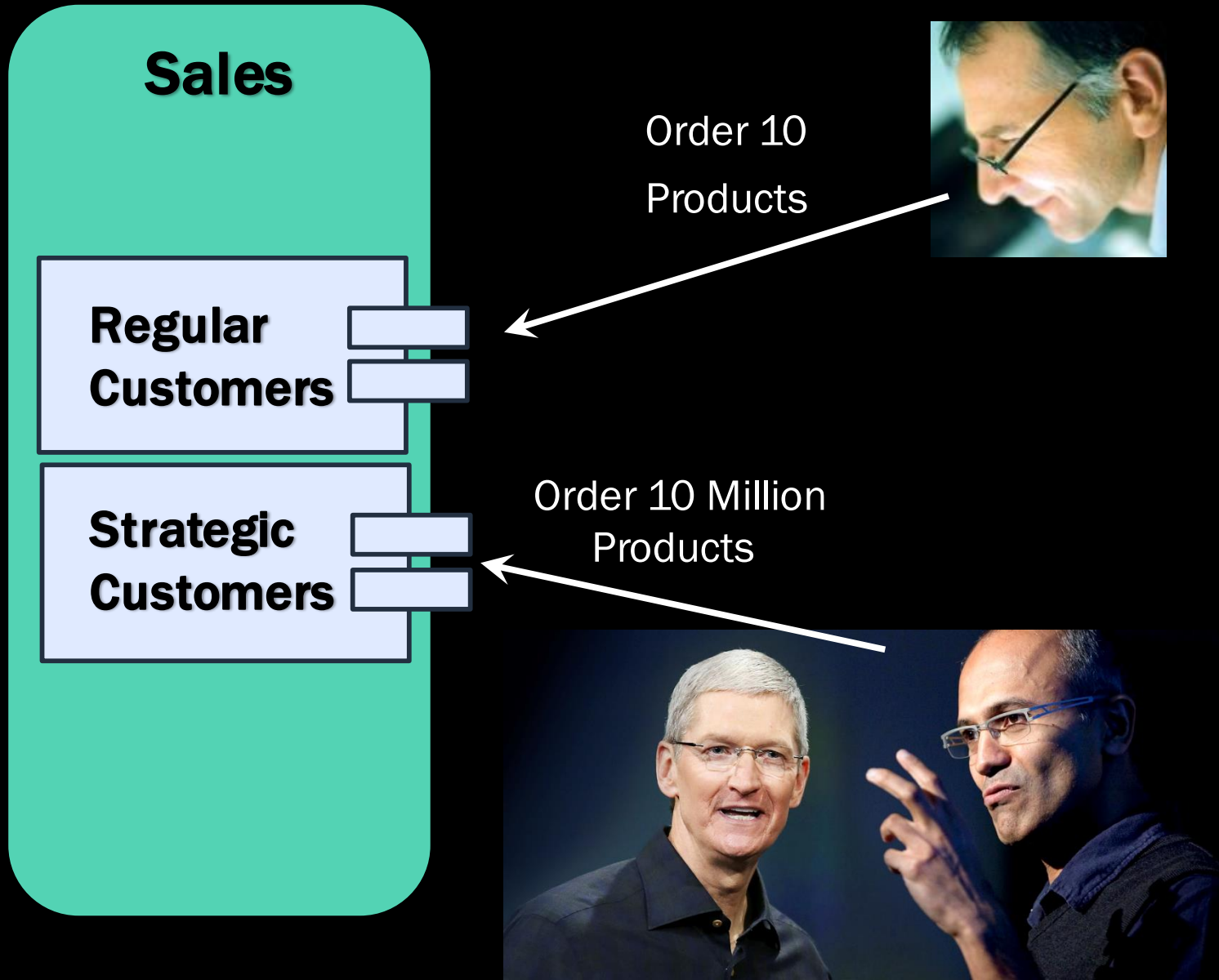
# SERVICE DECOMPOSITION

- The large-scale business capability that a service provides can be further broken down
- “Business Components” refer to the technical elements that implement the capability’s constituent parts

# QUALITY OF SERVICE REQUIREMENTS



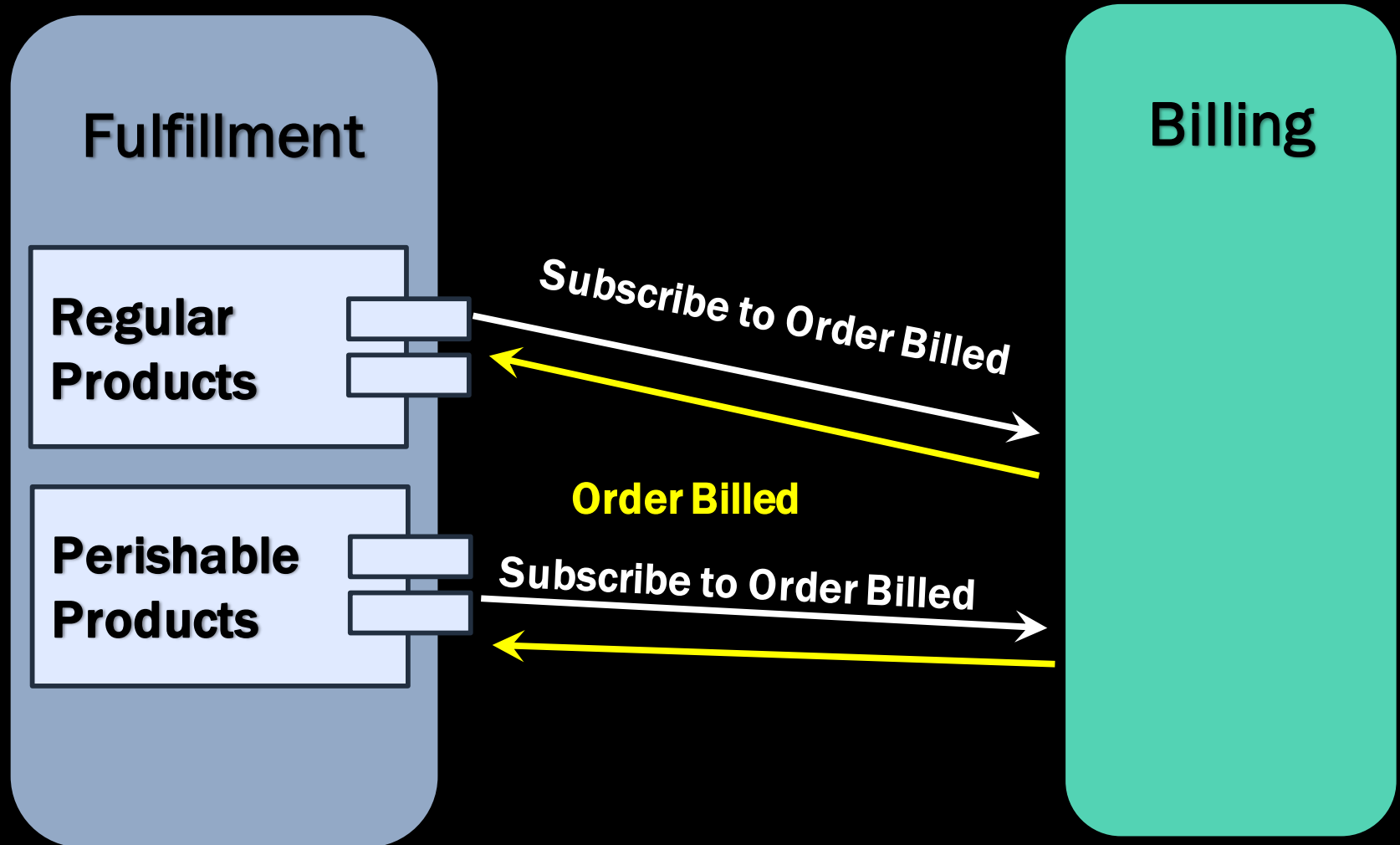
# BUSINESS COMPONENTS



# MORE EXAMPLES

- Airlines
  - Different counters for business class customers
- Shipping
  - Need to track temperatures for frozen goods / meat
  - Can use regular trucks for toilet paper
- Billing
  - Monthly invoicing for customers with an agreement
  - Others billed directly to credit card

# BUSINESS COMPONENTS & PUB/SUB





# TRANSACTIONS

- Some kinds of messages don't change state
- Other kinds of messages can get lost in case of failure (volatile data like stock quotes)
- We can divide a Business Component along transactional lines for improved performance, into "Autonomous Components"

Transactions stay within the boundary of an AC

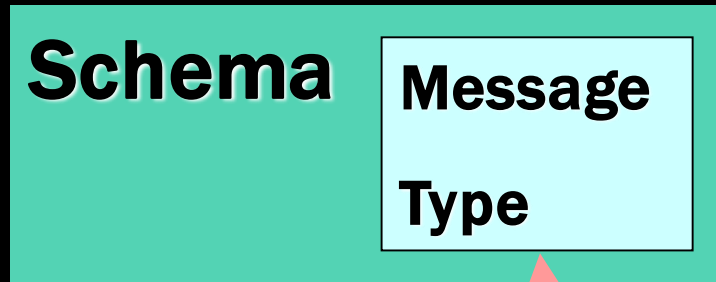
# AUTONOMOUS COMPONENTS

- A Business Component is composed of one or more Autonomous Components
- An Autonomous Component is responsible for one or more message types

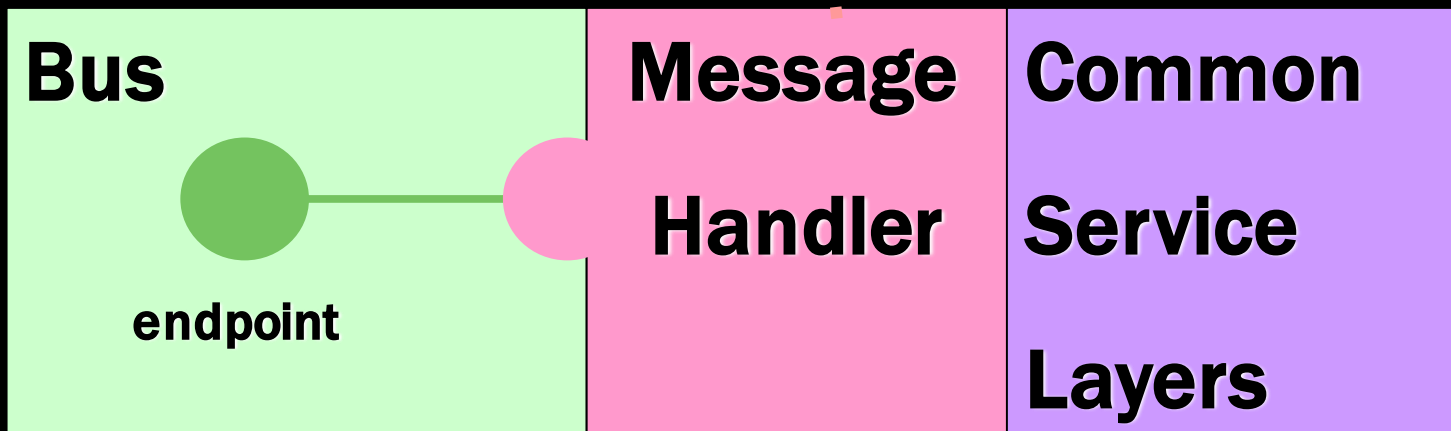
Composed of one or more message handlers and the rest of the layers in the service

**Is an independent package (NuGet)**

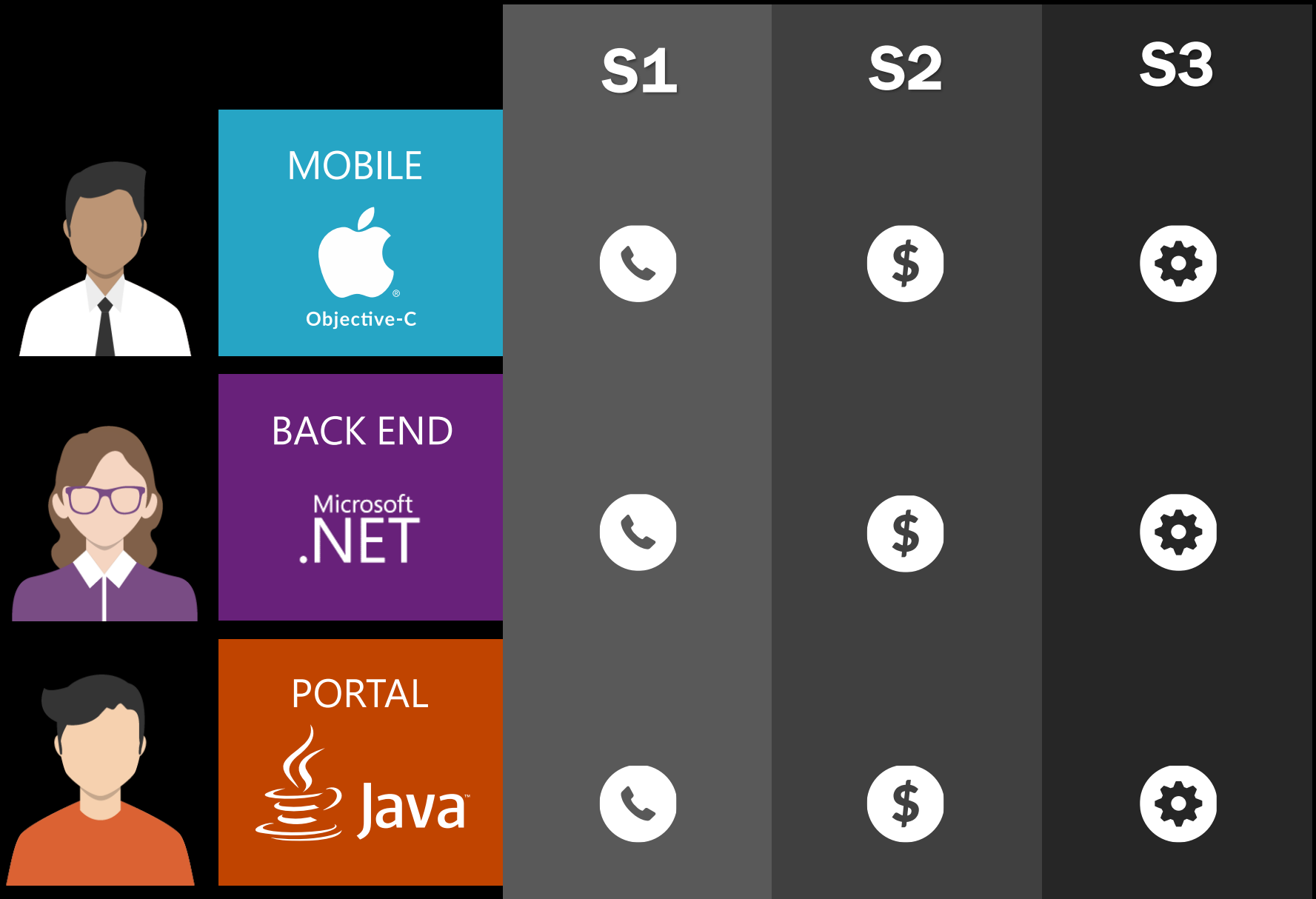
# AUTONOMOUS COMPONENT MAKEUP



Multiple ACs can be hosted together

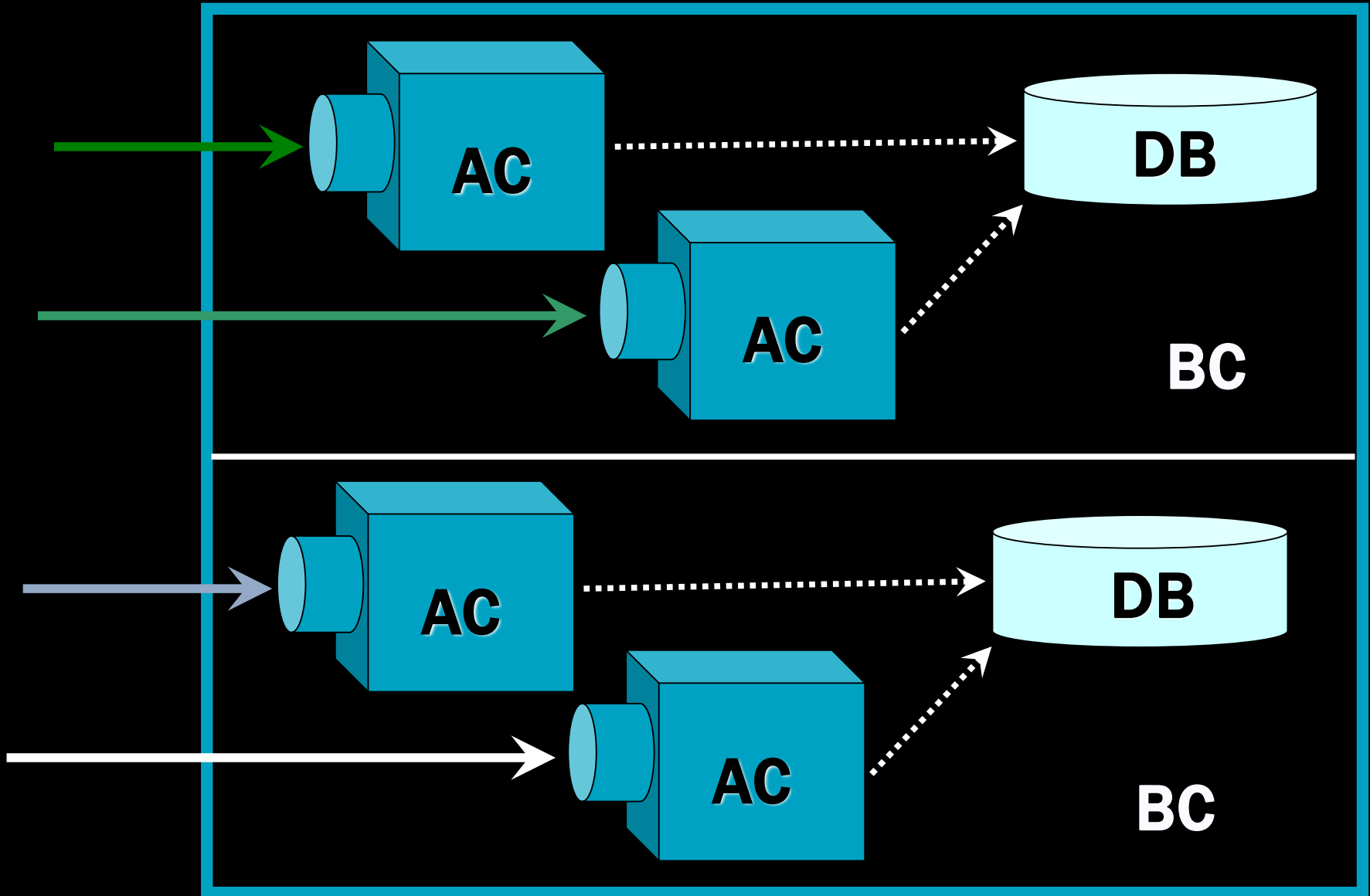


# AC DEPLOYMENT INTO SYSTEMS



# AC'S , BC'S, AND SERVICES

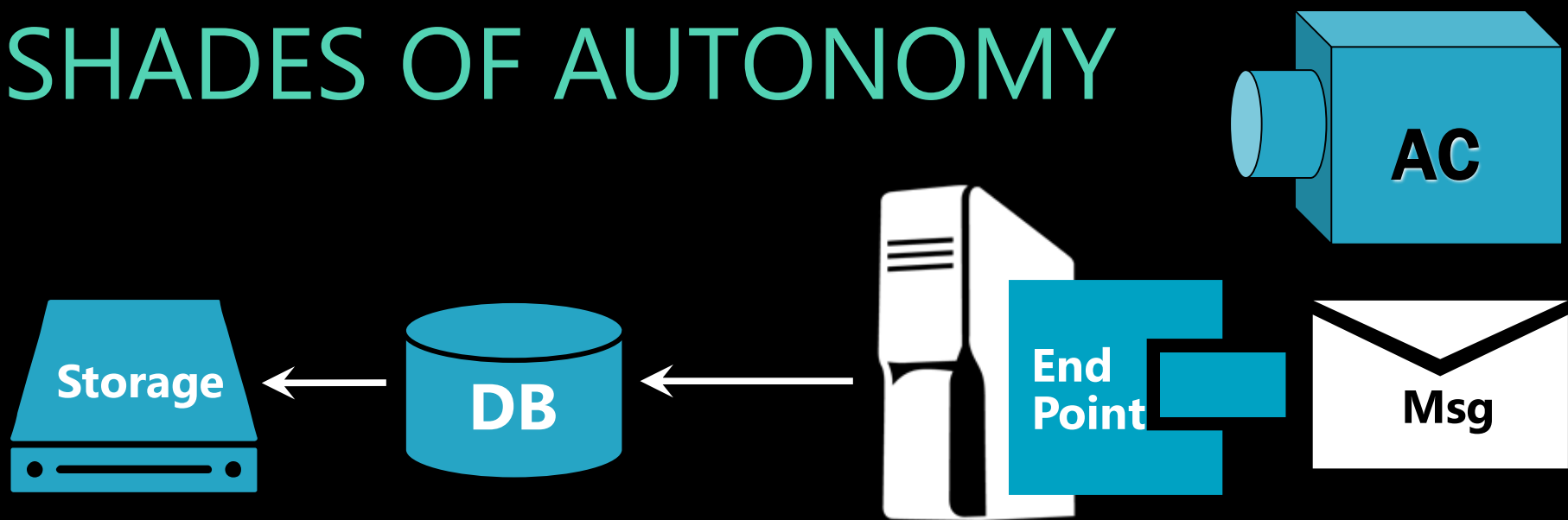
Service



# COUPLING BETWEEN ACS

- Don't try to reuse code between ACs
  - Strive for "disposable" code
  - Solve for today's problem – not tomorrow's
- JFHCI

# SHADES OF AUTONOMY



*Shared*

*Shared*

*Shared*

*Shared*

*Shared*

**Own**

*Shared*

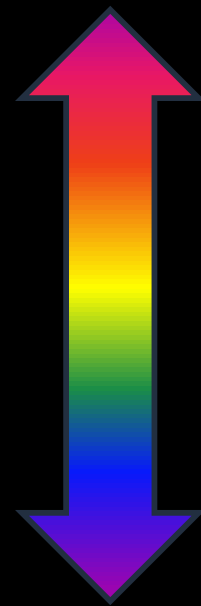
**Own**

**Own**

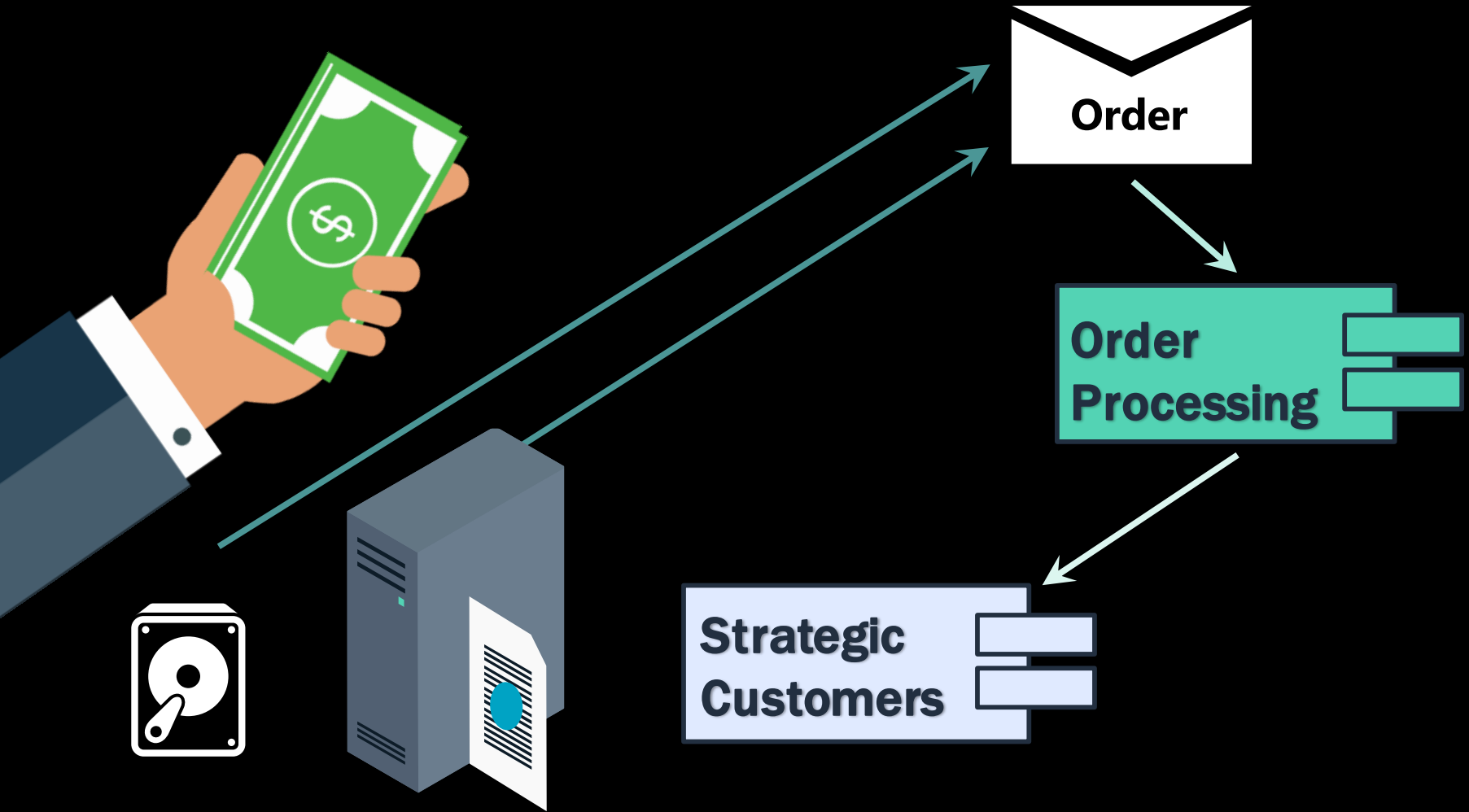
**Own**

**Own**

**Own**



# SOA => BUSINESS VALUE





# SUMMARY

- Autonomous Components are the unit of packaging in SOA – deployed into Systems.
- An AC takes responsibility for a specific set of message types in the service (ideally 1).
- Strive to use the bus style between ACs.

# SERVICE STRUCTURE

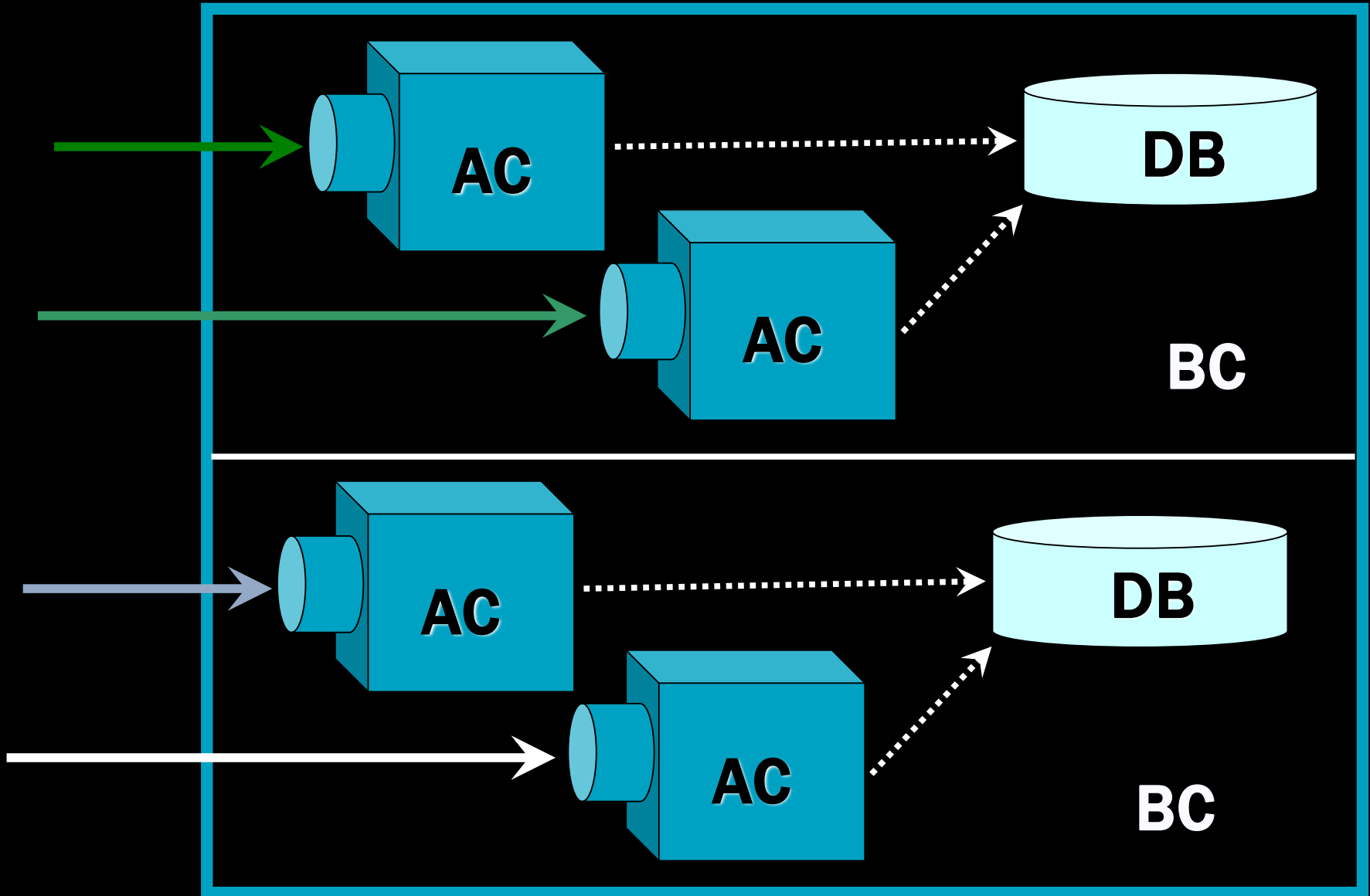
COMMAND/QUERY RESPONSIBILITY SEGREGATION



SERVICES

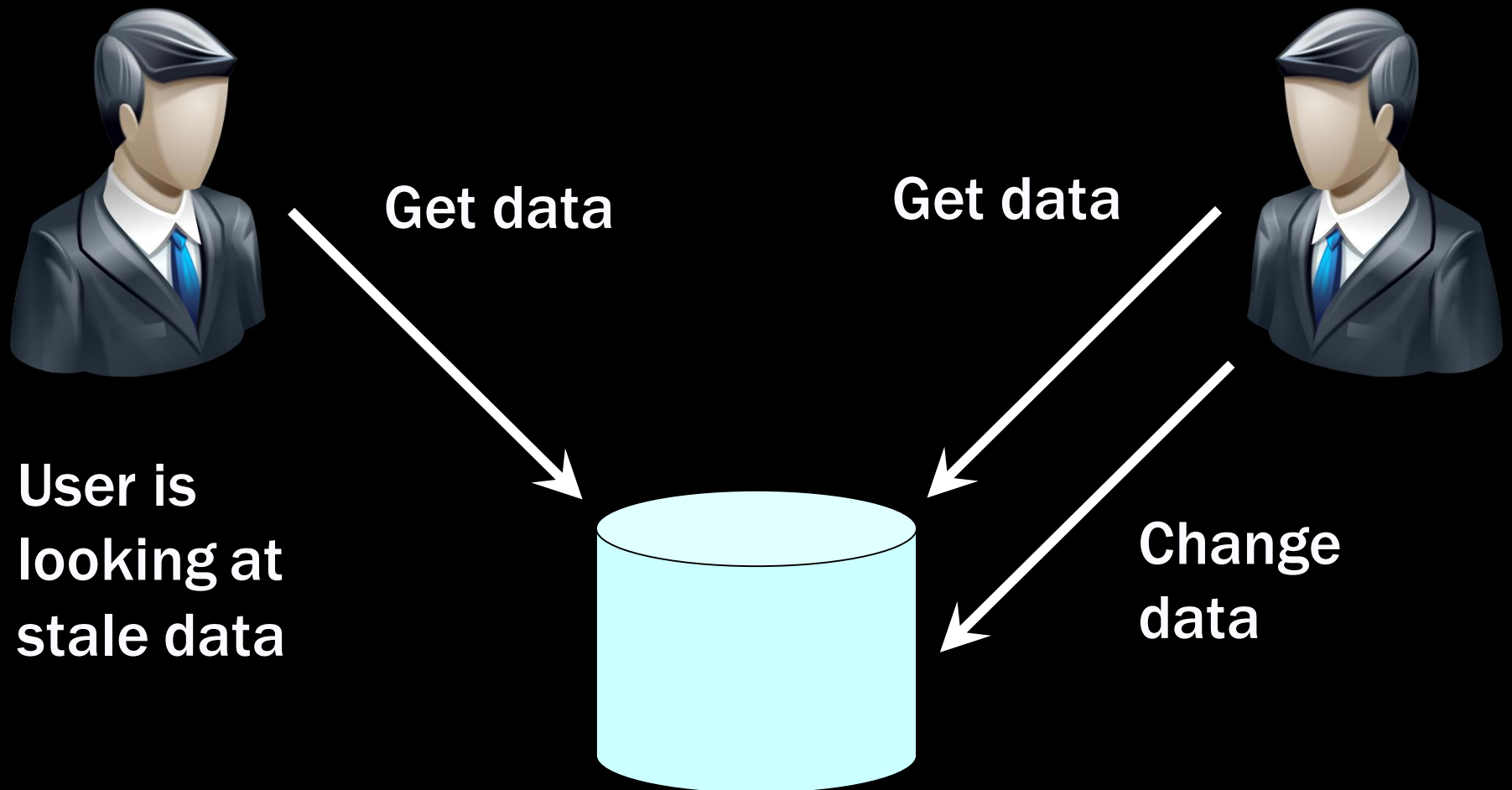
# AC'S , BC'S, AND SERVICES

Service

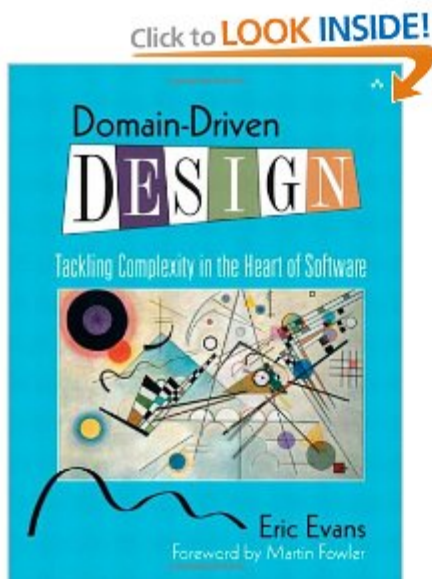


# MULTI-USER COLLABORATION

Need to assume users are looking at stale data



# HOW MUCH DATA IS COLLABORATIVE?



[Share your own customer images](#)

[Search inside this book](#)

## Domain-Driven Design: Tackling Complexity in the Heart of Software [Hardcover]

[Eric Evans](#) ☒ (Author)

★★★★★ ☒ (150 customer reviews) |  (41)

List Price: ~~\$69.99~~

Price: **\$48.96** & this item ships for **FREE with Super Saver Shipping**. [Details](#)

You Save: **\$21.03 (30%)**

[Special Offers Available](#)

**In Stock.**


Ships from and sold by **Amazon.com**. Gift-wrap available.

[31 new](#) from \$44.97    [32 used](#) from \$36.00

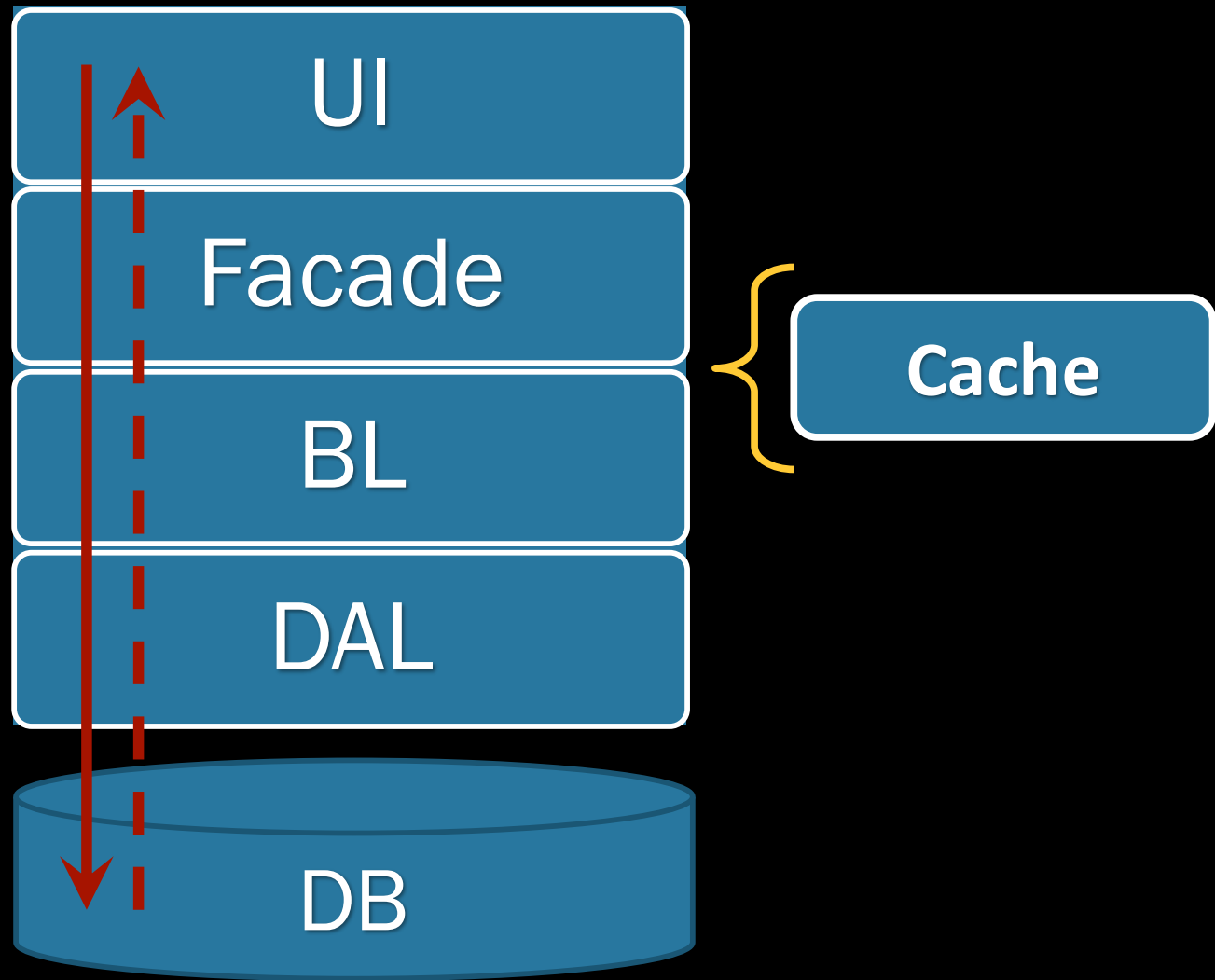


FREE Two-Day Shipping for students on millions of items. [Learn more](#)

### Formats

	Amazon Price	New from	Used from
Kindle Edition	\$23.99	--	--
Hardcover	\$48.96	\$44.97	\$36.00
 Paperback	--	--	--

# SIMPLEST SOLUTION? HARDLY.



# WHY TRANSFORM BETWEEN TIERS?

Keeping the cache up to date is even more work



Map from DTOs &  
WS to view model

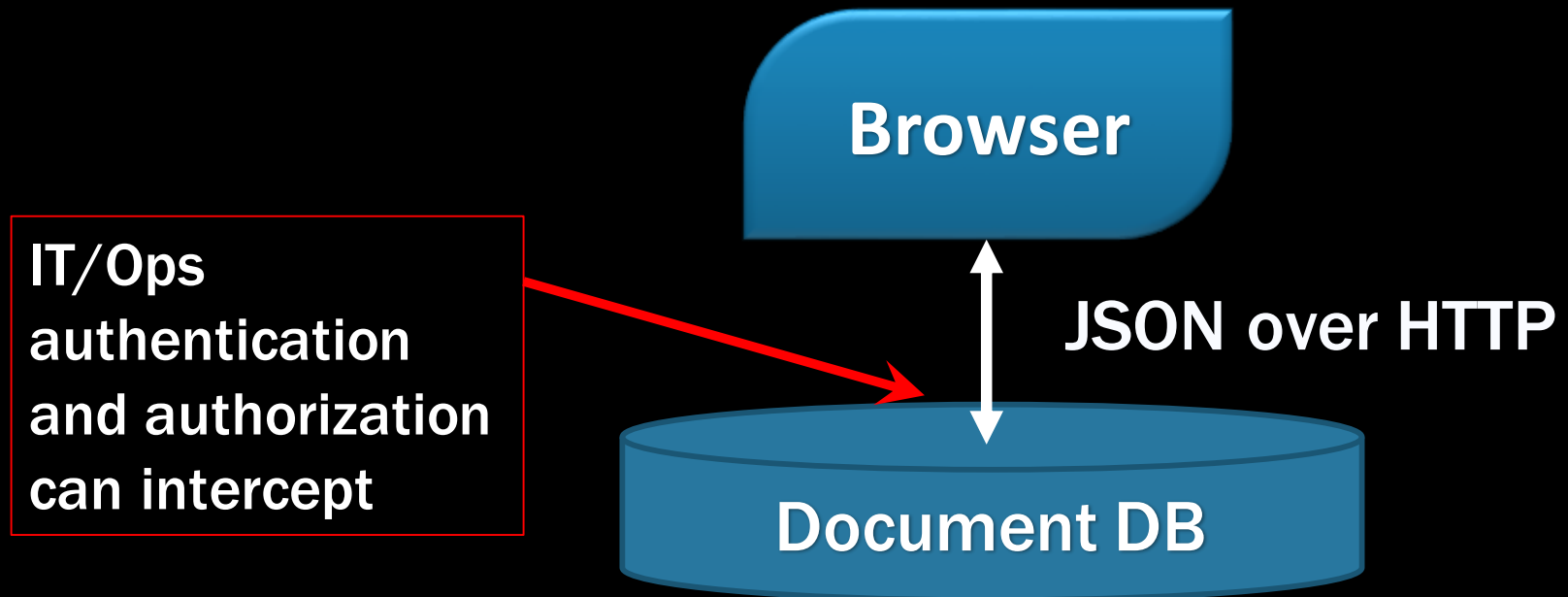
Map from DTOs  
and WS to  
domain objects

Use ORM to map  
from tables to  
domain objects



# SIMPLER SOLUTIONS

- Datasets UI through to DB
- Ruby on Rails



# FACETED SEARCH

Search:

▼ watches

## Brand

- ☐ FitBit
- ☐ Casio

## Style

- ☐ Casual
- ☐ Luxury

## Price

- ☐ Under \$25
- ☐ \$25 to \$100

search results

### Product A



Consectetur adipiscing elit. Etiam in laoreet ante. Etiam rutrum neque nec dui consequat, in pulvinar leo porttitor. Mauris congue, arcu et semper lacinia,.

### Product B



Consectetur adipiscing elit. Etiam in laoreet ante. Etiam rutrum neque nec dui consequat, in pulvinar leo porttitor. Mauris congue, arcu et semper lacinia,.

### Product C



Consectetur adipiscing elit. Etiam in laoreet ante. Etiam rutrum neque nec dui consequat, in pulvinar leo porttitor. Mauris congue, arcu et semper lacinia,.

### Product D

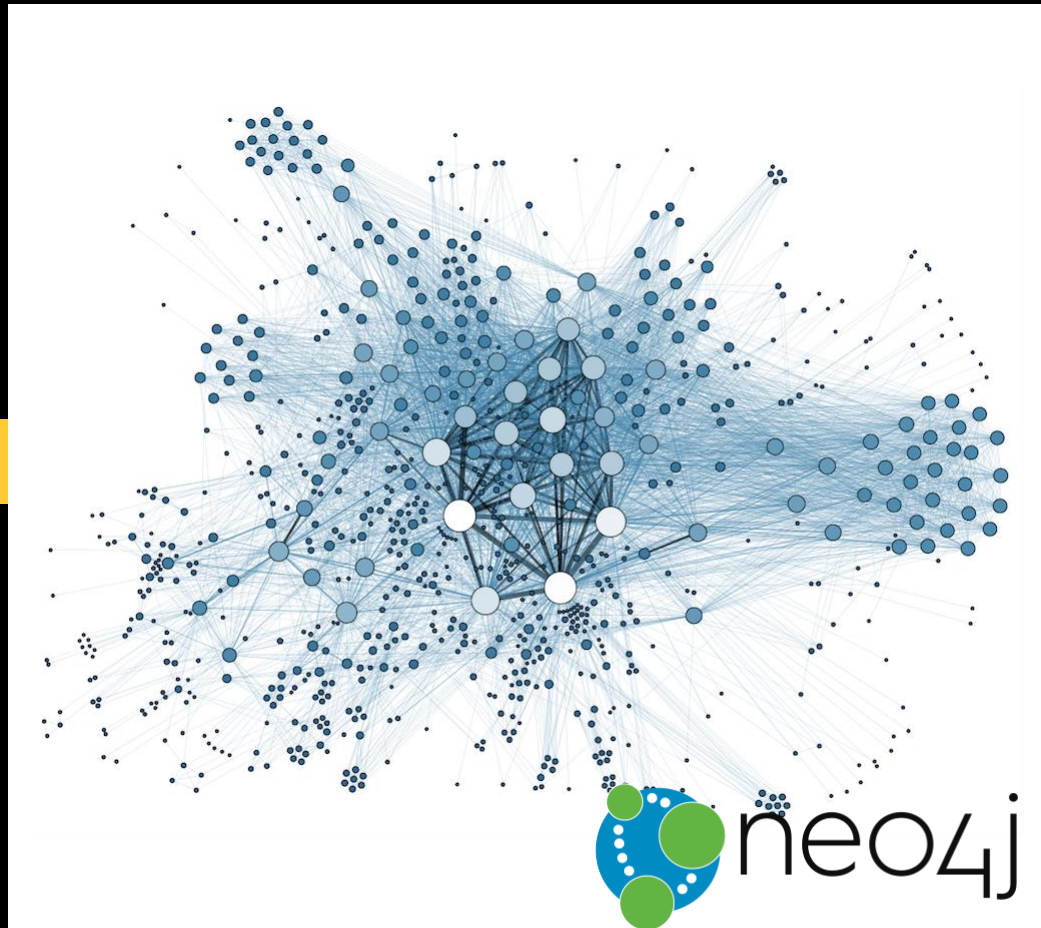
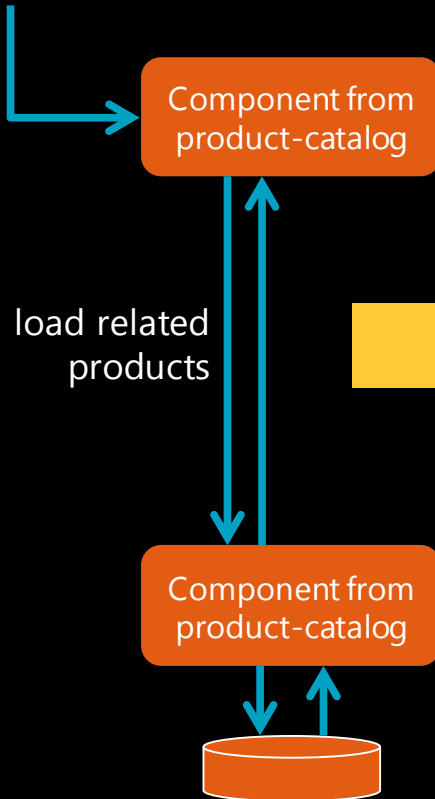


Consectetur adipiscing elit. Etiam in laoreet ante. Etiam rutrum neque nec dui consequat, in pulvinar leo porttitor. Mauris congue, arcu et semper lacinia,.

# RECOMMENDATION ENGINES

<div>ProductNameA</div> <div><div>cover image A</div><div>AuthorNameA ★★★★★</div><div>€ 20.00</div></div>	<div>ProductNameB</div> <div><div>cover image B</div><div>AuthorNameB ★★★★★</div><div>€ 20.00</div></div>	<div>ProductNameC</div> <div><div>cover image C</div><div>AuthorNameC ★★★★★</div><div>€ 20.00</div></div>	<div>ProductNameD</div> <div><div>cover image D</div><div>AuthorNameD ★★★★★</div><div>€ 20.00</div></div>
---	---	---	---

<http://mydomain.com/products/123>



# HIGH CONTENTION DOMAINS

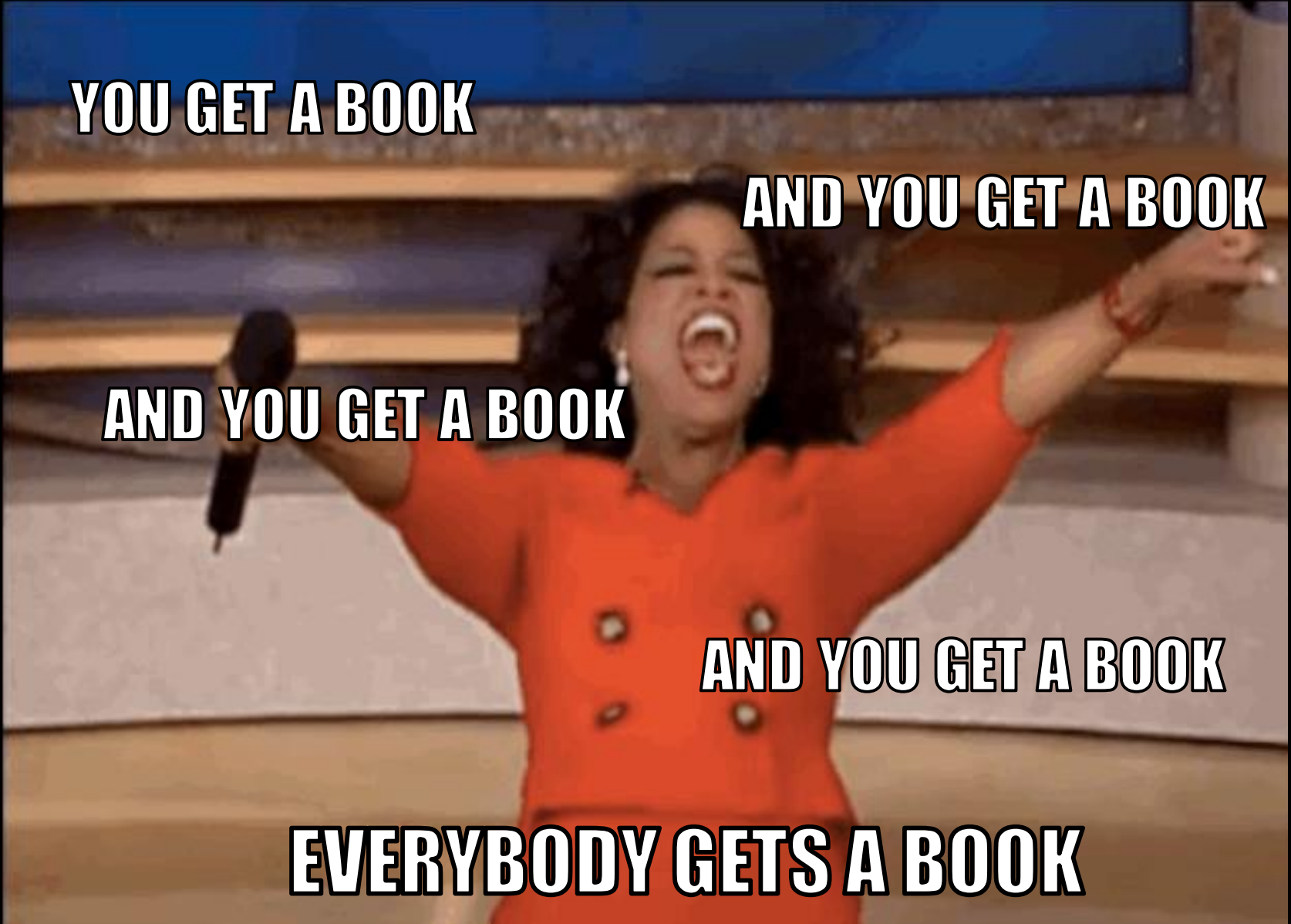
**YOU GET A BOOK**

**AND YOU GET A BOOK**

**AND YOU GET A BOOK**

**AND YOU GET A BOOK**

**EVERYBODY GETS A BOOK**



# REGULAR LOGIC CHOKES

## Inventory table

Id	Name	Quantity
42	Harry Potter 1	7
1337	Harry Potter 2	1000



```
begin transaction
    var quantity = select Quantity from Inventory
                    where Id = @ProdId
    if (quantity >= quantityRequested)
        update Inventory
        set Quantity = quantity - quantityRequested
        where Id = @ProdId
commit transaction
```

# CONNECTION POOL DRIES UP



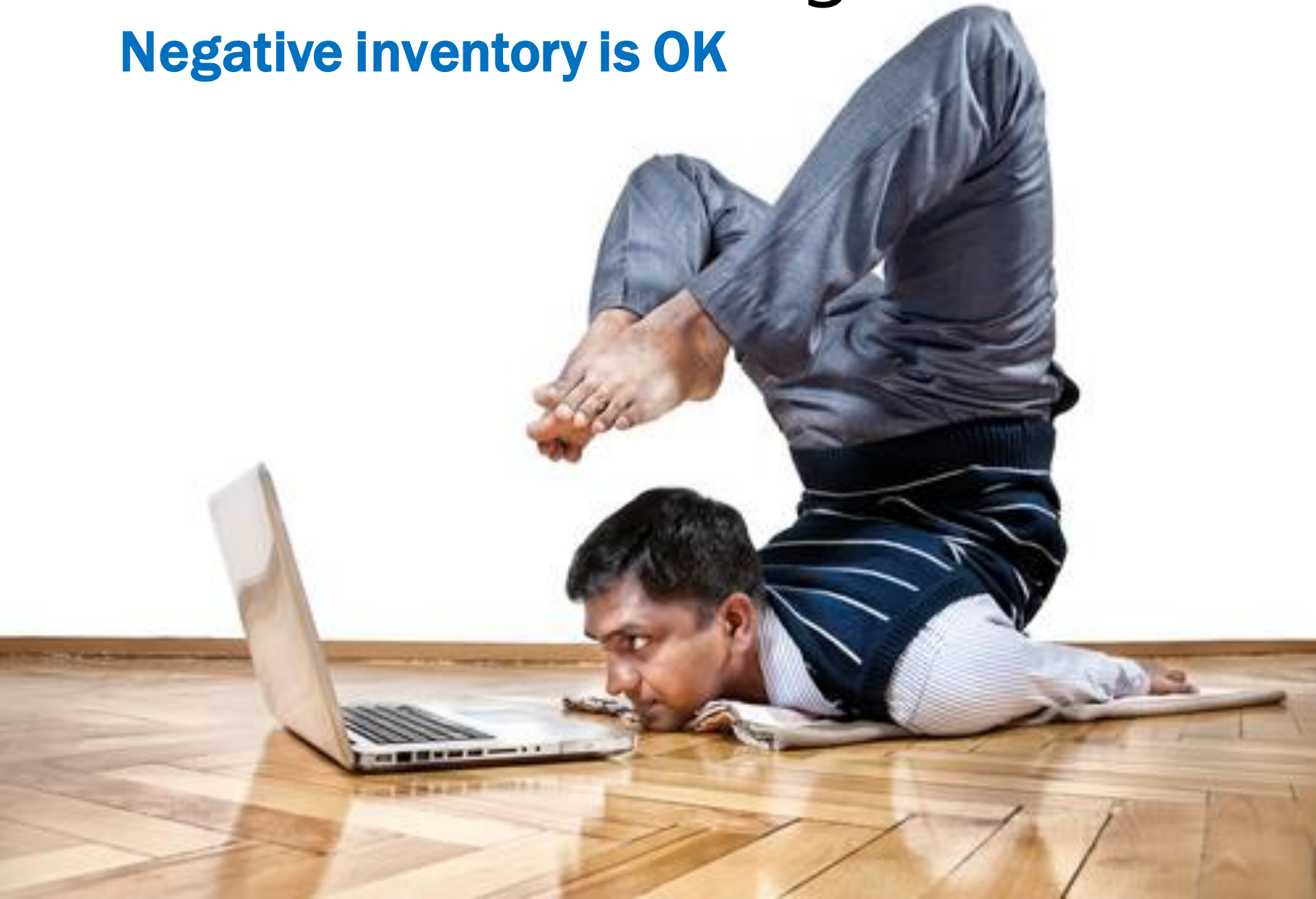
**sorry, the site is temporarily  
unavailable.**

We're working hard to fix the problem, and will have the site back up as soon as possible. We apologize for the inconvenience - thanks for your patience.



# Business needs to get flexible

**Negative inventory is OK**



# APPEND ONLY DATAMODELS

- ⤵ EventSourcing? Not quite.
- ⤵ No locking
- ⤵ Responsibility shifted

ProductId	Delta	Timestamp
1337	-5	09:03:22 17-1-2016
1337	-3	09:03:24 17-1-2016
1337	-4	09:03:25 17-1-2016
1337	-1	09:03:27 17-1-2016
1337	+250	09:03:28 17-1-2016
1337	-4	09:03:30 17-1-2016

`select sum(Delta) where...`

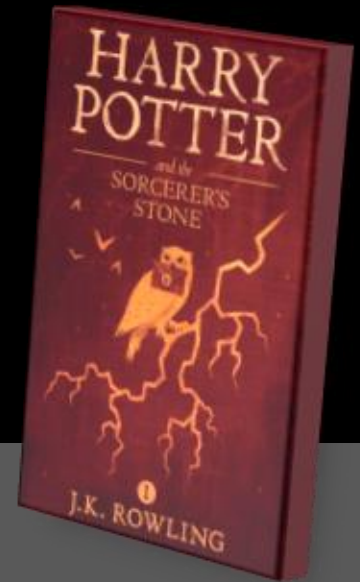
**data will expand fast!**





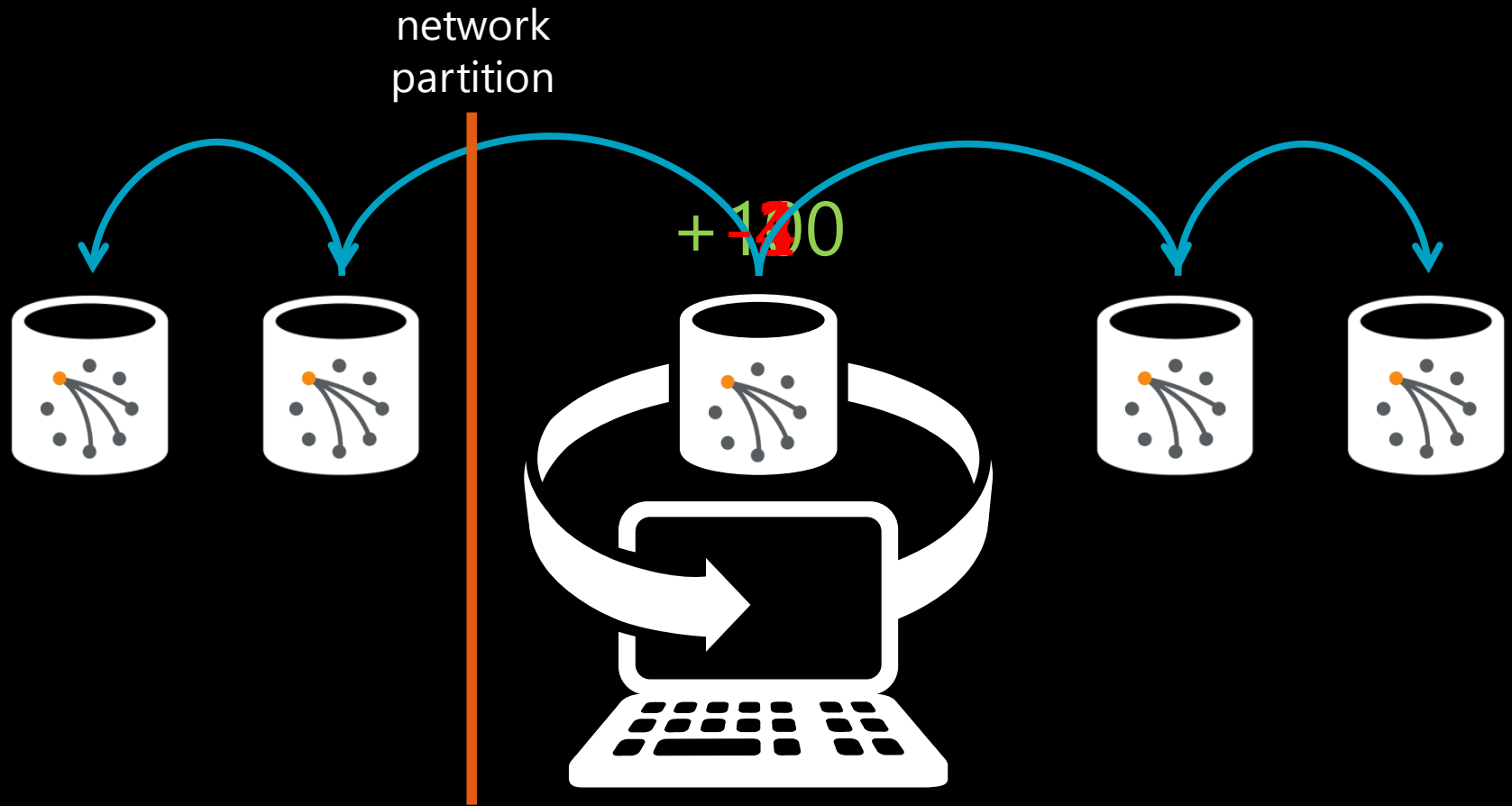
# ADD SNAPSHOTTING

ProductId	Delta	Timestamp
1337	+250	09:03:28 17-1-2016



```
begin transaction
    select @quantity = sum(Delta),
           TimeStamp.Before(5.min.ago)
           from Inventory where Id = @ProdId
    delete from Inventory where Id = @ProdId
           and TimeStamp.Before(5.min.ago)
    insert into Inventory @ProdId, @quantity, now()
commit transaction
```

# ...OR ATOMIC INCREMENT



# CQRS THEORY

?

QUERIES

# BE UP FRONT ABOUT QUERY STALENESS

**Your account**

---

---

---

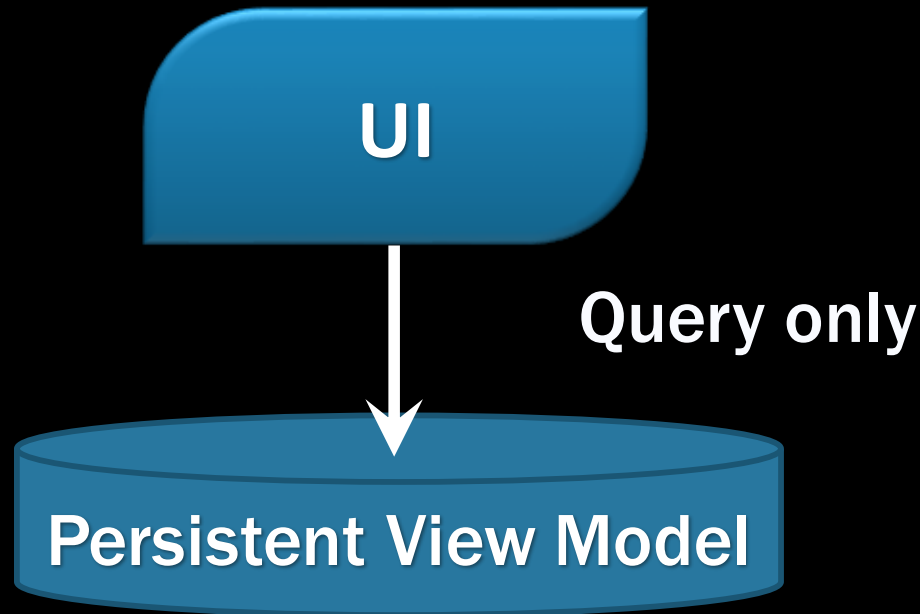
---

---

**Balance correct as  
of 10 minutes ago**

# KEEP QUERIES SIMPLE

2 Layers == 2 Tiers



For each view in the UI,  
have a view/table in the DB

```
SELECT * FROM MyTable (WHERE ID = @ID)
```

# DATA DUPLICATED, NO RELATIONSHIPS, DATA PRE-CALCULATED

## Customer Service Rep view

### List of customers

ID	Name	Phone

## Supervisor view

### List of customers

ID	Name	Phone	Lifetime value

### Rep\_Customers\_Table

ID	Name	Phone

### Supervisor\_Customers\_Table

ID	Name	Phone	Lifetime Value

# DEPLOYMENT AND SECURITY

- Deploy the persistent view model DB to the web tier (only SELECT is permitted)

Don't have to go through the firewall – faster

- Role-based security

Different screens for different roles go to different tables – SELECT permissions per role

- Just as secure as in-memory caches

If not more so



# USE FOR PRELIMINARY VALIDATION

- Before going to submit data, check if it already exists in the persistent view model
- Uniqueness  
Can expose to user (user signup)
- Related Entity Existence  
Address validation – existence of street name
- Results in less commands being rejected

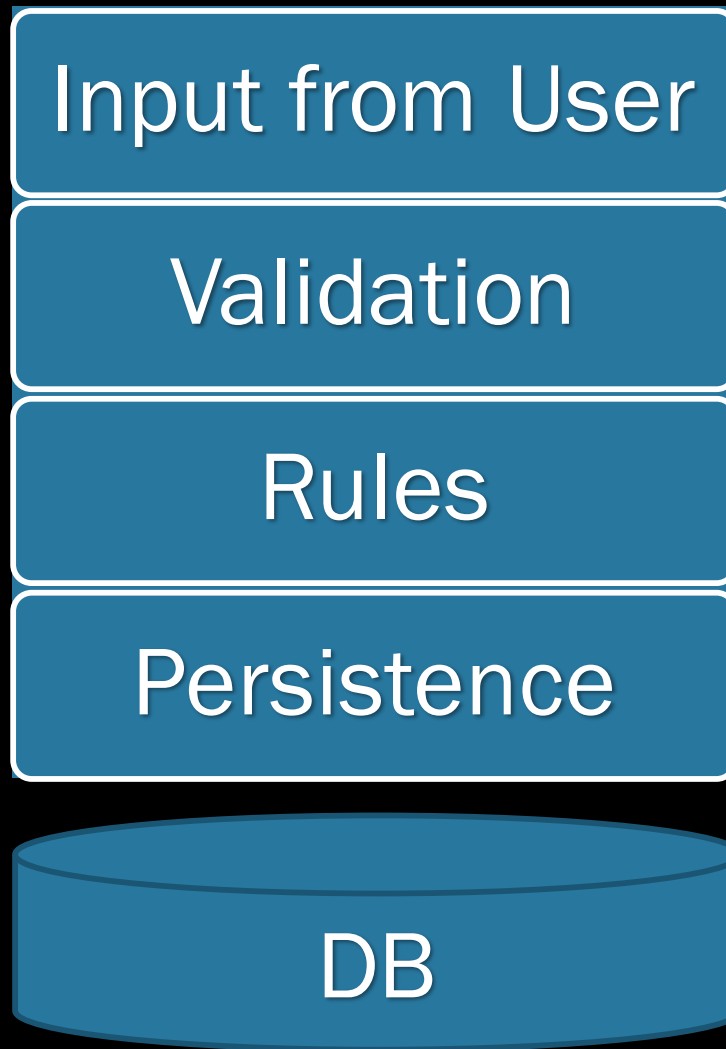


# COMMANDS

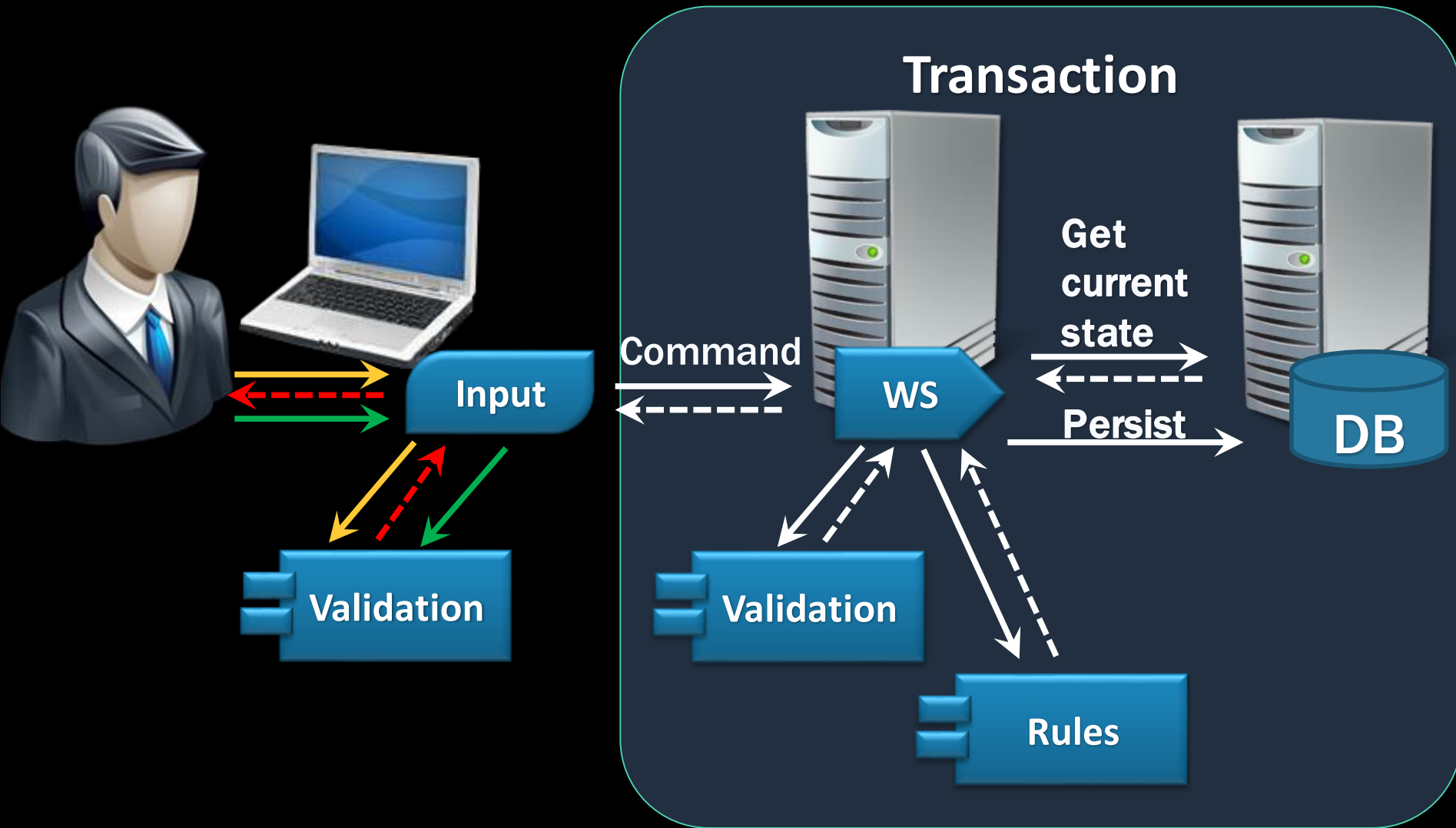
# VALIDATION AND BUSINESS RULES

- Validation: Is the input potentially good?  
Structured correctly?  
Ranges, lengths, etc
- Rules: Should we do this?  
Based on current system state  
What the user saw is irrelevant

# COMMAND PROCESSING LAYERS



# COMMAND PROCESSING TIERS



# AMAZON.COM "ADD TO SHOPPING CART"

The screenshot shows the Amazon.com homepage. At the top, the Amazon logo is on the left, and the user's name "BAT-SHEVA DAHAN" is in the center. To the right of the name are links for "Today's Deals", "Gifts & Wish Lists", and "Gift Cards". Further right are links for "Your Account" and "Help". Below the header is a navigation bar with "Shop All Departments" and a search bar containing the word "Books". To the right of the search bar are buttons for "GO", "Cart", and "Wish List". Below the navigation bar is a banner for "Your order qualifies for free shipping!" with a link to "See details". Below the banner is a section for "Get the Amazon.com Rewards Visa® Card and Get \$40 Back". To the right of this section is a "YOUR SHOPPING CART" sidebar. The sidebar contains a "Proceed to Checkout" button, a checkbox for "Show gift options during checkout", and a list of items added to the cart. The first item is "Patterns of Enterprise Application Architecture - Martin Fowler Hardcover", with a price of \$45.87 and a quantity of 1. Below the item list is a "subtotal = \$45.87" and an "Edit shopping cart" button. At the bottom of the sidebar is another "Proceed to Checkout" button and a note about 1-Click ordering. A red arrow points from the "Add to Cart" button of the book "Patterns of Enterprise Application Architecture" to the "YOUR SHOPPING CART" sidebar.

amazon.com Hello, BAT-SHEVA DAHAN. We have [recommendations](#) for you. (Not BAT-SHEVA?) FREE 2-Day Shipping: [See details](#)

BAT-SHEVA's Amazon.com [Today's Deals](#) [Gifts & Wish Lists](#) [Gift Cards](#) [Your Account](#) [Help](#)

Shop All Departments Search Books GO Cart Wish List

Books Advanced Search Browse Subjects New Releases Bestsellers The New York Times® Bestsellers Libros En Español Bargain Books Textbooks

Your order qualifies for free shipping! (Some restrictions apply)  
Make sure to select **FREE Super Saver Shipping** as your shipping speed at checkout.

Get the Amazon.com Rewards Visa® Card and **Get \$40 Back**

Sign up now

Customer

Domain-Driven Design: Tackling Complexity in the Heart of Software by Eric Evans  
Price: ~~\$69.99~~ \$46.34 [★★★★☆](#)  
Used & new from \$28.50 [Add to Cart](#)

Refactoring: Improving the Design

Applying Domain-Driven Design and Patterns by Eric Evans  
Price: ~~\$69.99~~ \$46.44 [★★★★☆](#) (19)  
Used & new from \$42.24 [Add to Cart](#)

Customers Who Shopped For

Microsoft .NET: Architecting Applications for the Enterprise

chitecture Also Shopped

ANALYSIS PATTERNS by Martin Fowler

**YOUR SHOPPING CART**

[Proceed to Checkout](#)

☐ Show gift options during checkout [?](#)

**Added to your Shopping Cart:**

[Patterns of Enterprise Application Architecture](#) - Martin Fowler  
**Hardcover**  
Condition: New  
**\$45.87**  
- quantity: 1

**subtotal = \$45.87**

[Edit shopping cart](#)

[Proceed to Checkout](#)

[Sign in](#) to turn on 1-Click ordering.  
Items in your Shopping Cart always reflect the most recent price displayed on their product pages.

**Shopping cart not actually shown!**

**Item added shown (from cmd)**

# SHOULD WE DO WHAT THE USER ASKED?

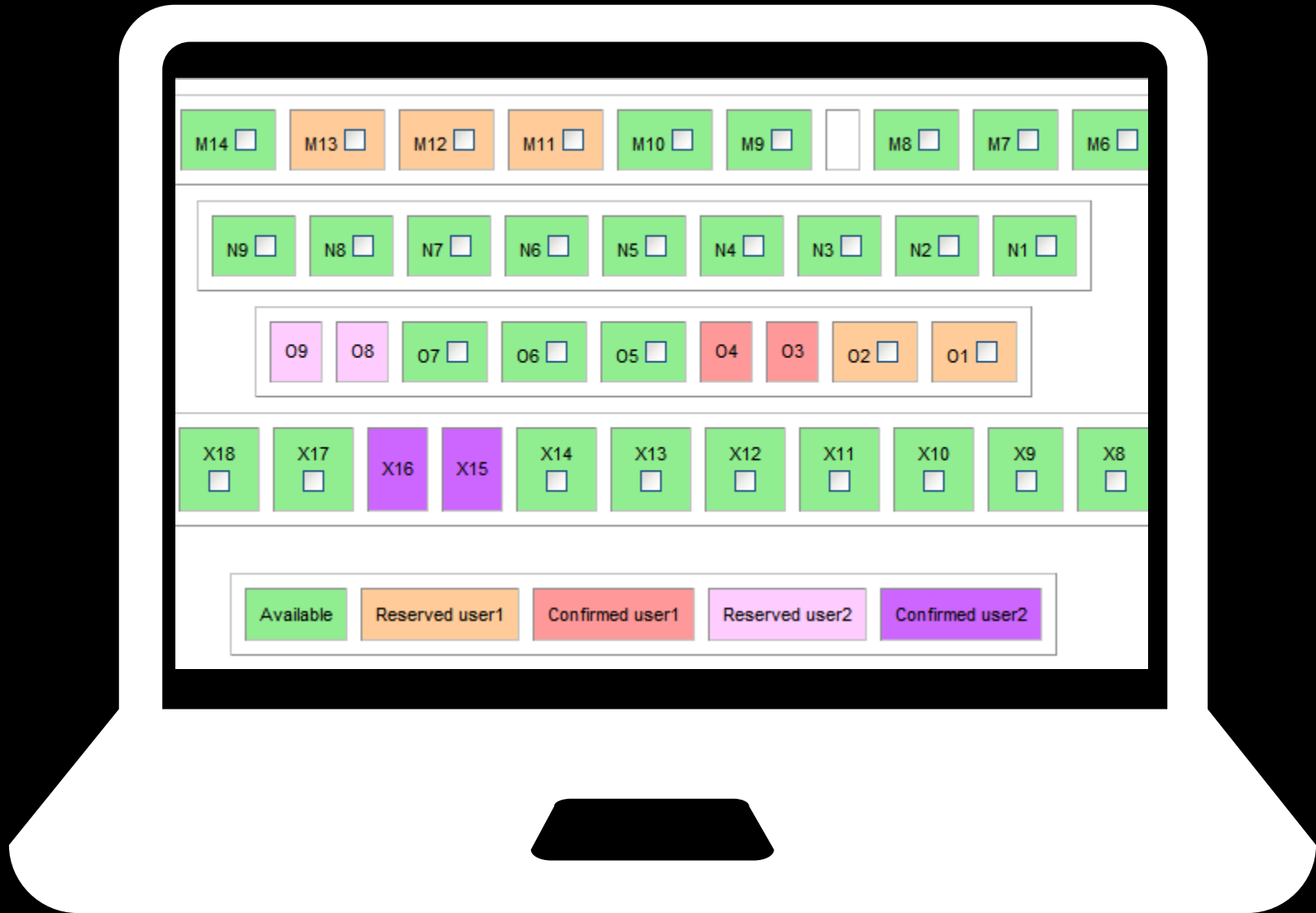
## Orders

ID	Total	Date	Shipped	Account	etc	etc	etc
317	\$37.87	1/9/09	Yes	A17T5			
318	\$99.99	3/7/09	Yes	A17T5			
319	\$100.11	4/8/09	Yes	P313Z			
320	\$69.47	9/9/09	No	P599Z			

Save

Cancel

# RESERVATION SYSTEMS





# RESERVATION SYSTEMS



# UI BORN OF SINGLE USER THINKING



**Customers line up at ticket counters**

**Line provides fairness**

**Line is invisible on the internet – we can be “unfair”**

# NOT CAPTURING USER INTENT

- In a traditional UI – the checkbox
- Why do users select multiple seats?  
Because they're reserving for a family / friends
- But then, concurrency happens  
Somebody else got in first on one of the seats
- Try to find a block of seats somewhere else

# CAPTURING USER INTENT

- Group reservations: people want to sit together
  - Enter number of people
  - Enter preferred seat **type** – indicates cost
  - Emails back when reservation can be filled
- Include waiting list functionality

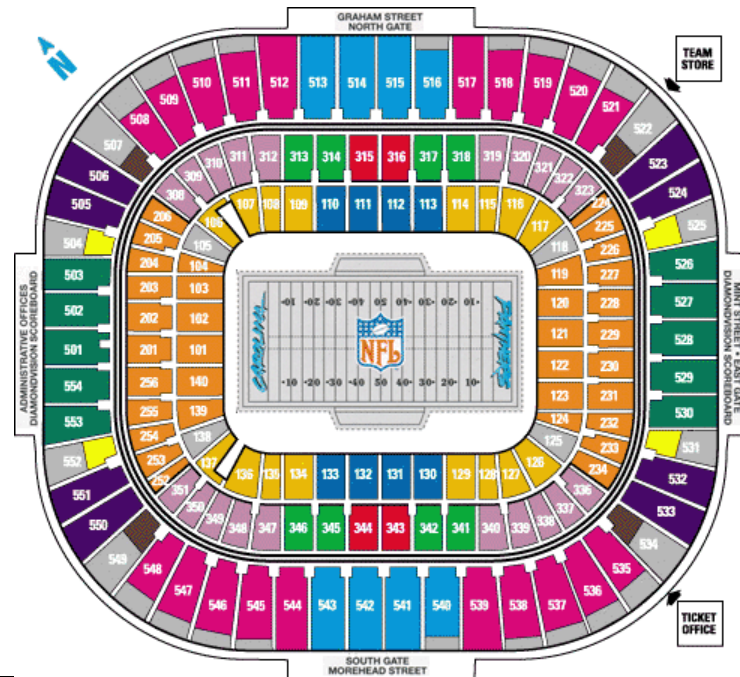
# USER POPULATION PARTITIONING

- VIP customers – first window of time  
Want the best seats in the house  
Willing to pay extra for the privilege
- Groups – next window of time  
Want to sit together
- Everybody else – last window of time  
Will take any free seat

# SCALABILITY BENEFITS

Thousands of seats, hundreds of thousands of requests

**No need to show actual status**



# WHAT IS A GOOD COMMAND?

- The kind you can reply with:

“Thank you.

Your confirmation email will arrive shortly”

OR

Just fake it in the UI

- Inherently asynchronous
- Not really related to an entity

# COMMANDS VERSUS ENTITIES

- It's easier to validate the command

Less data

More specific

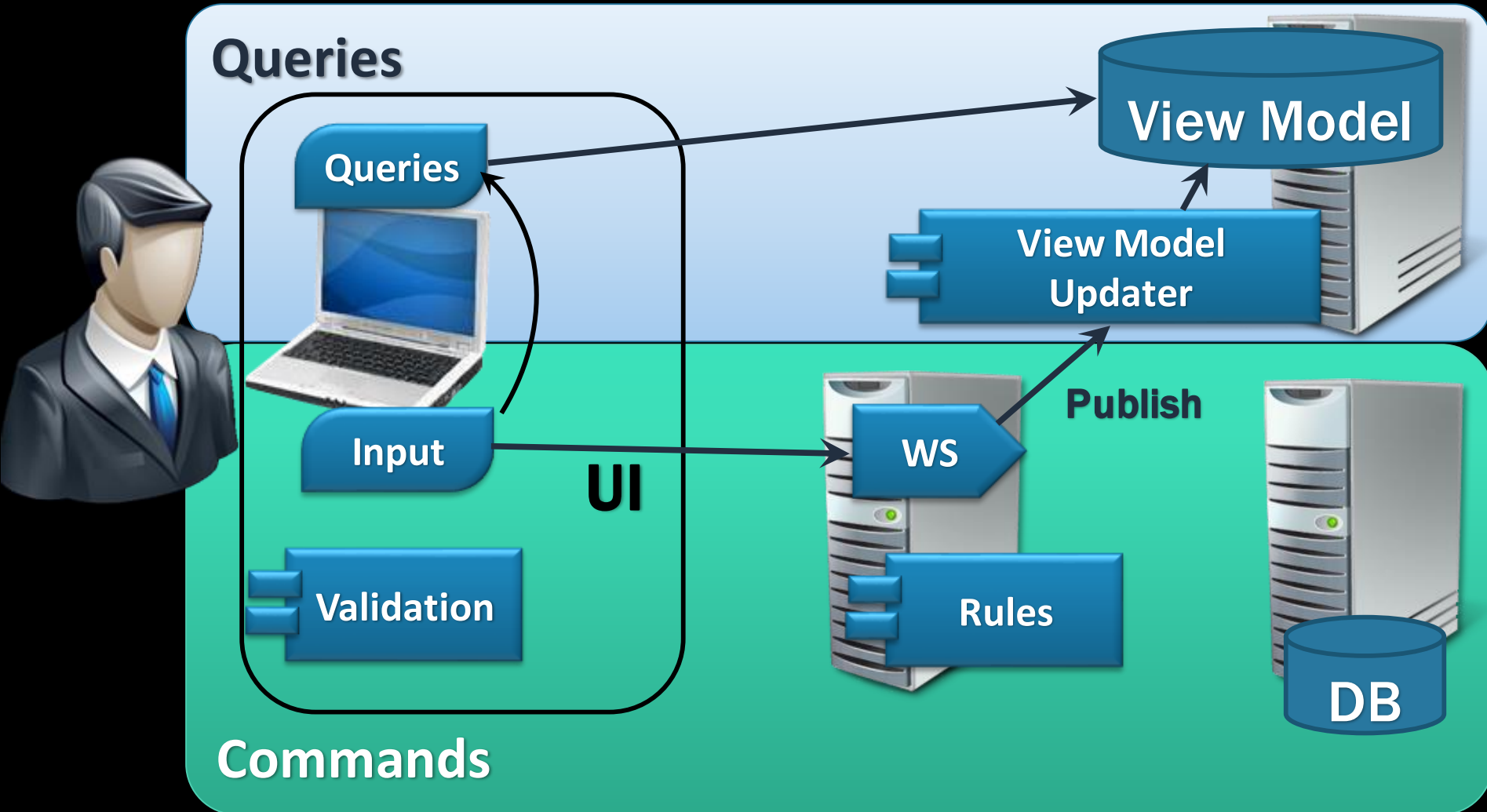
Is this *potentially good*

- Validating large entities is complex



# CQRS IN ACTION

Data from input immediately overlaid on queries

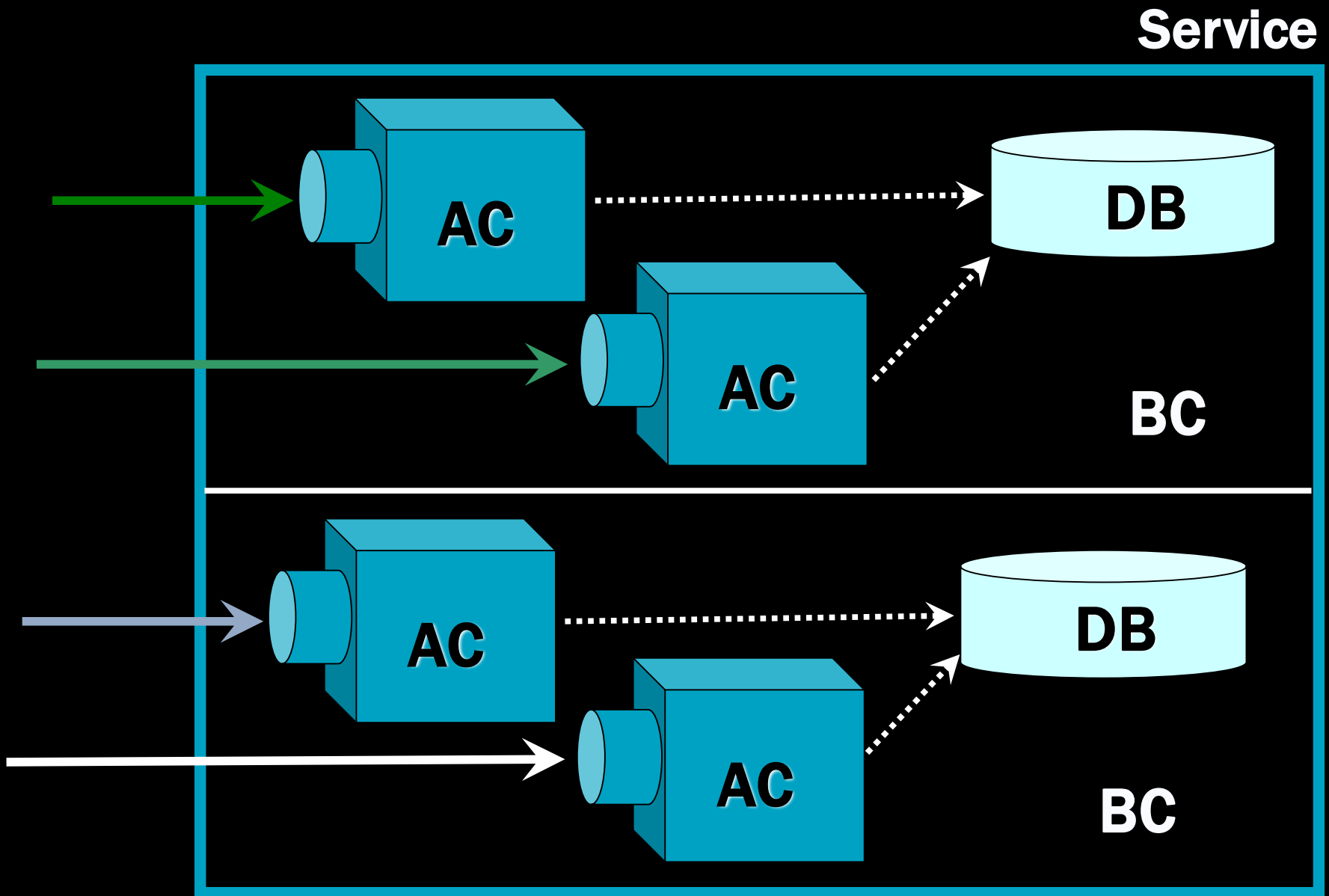


# SUMMARY

- Think of reads and writes differently
- Reflect that difference in the schemas
- Design commands so that they almost can't fail

# SCALABILITY AND FLEXIBILITY MONITORING AND MANAGEMENT

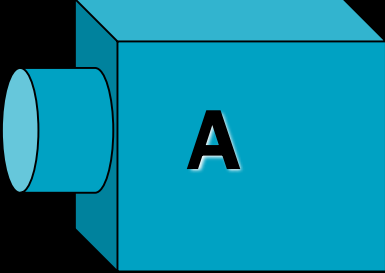
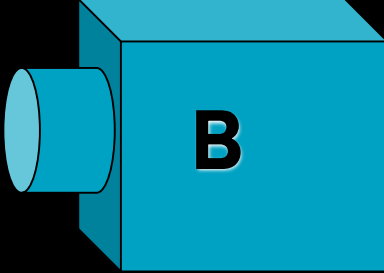
# NAMING QUEUES AND PROCESSES



# MONITORING QUEUE-BASED SYSTEMS

- Visibility into the number of messages in various queues provides insight
- Error queue notifies admins of problems

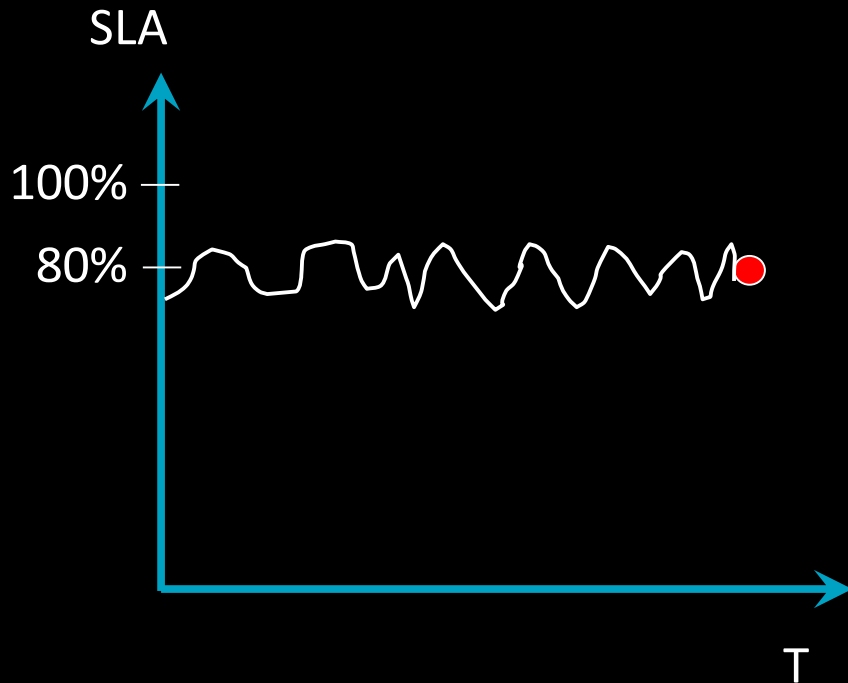
# IDENTIFYING BOTTLENECKS

# of messages	1000		500	
Throughput	500 msgs/s		50 msgs/s	
Total msg processing time	2 s		10 s	

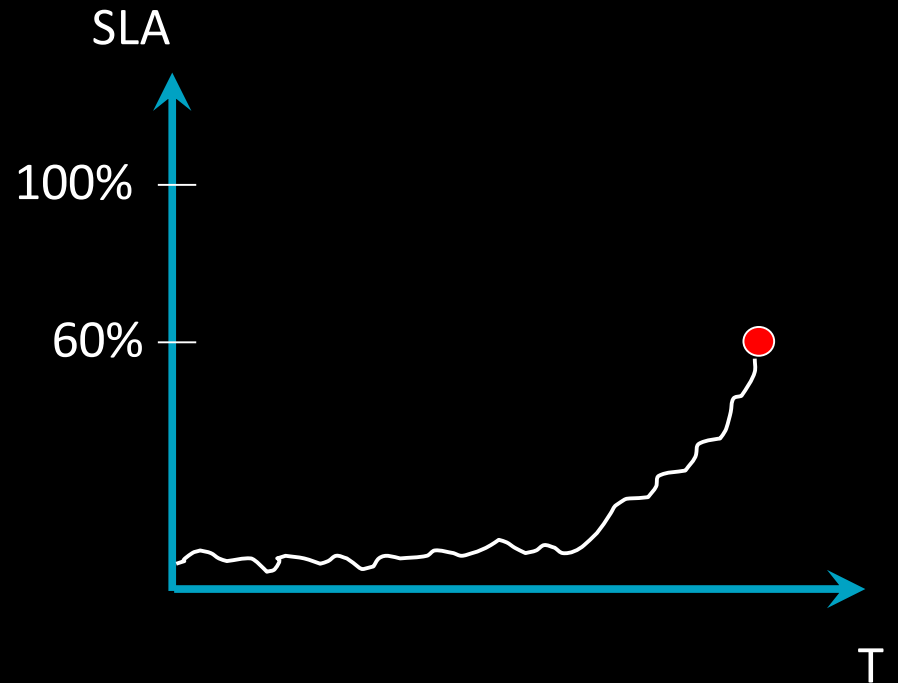
**Performance Counters**

# SO, WHICH IS WORSE?

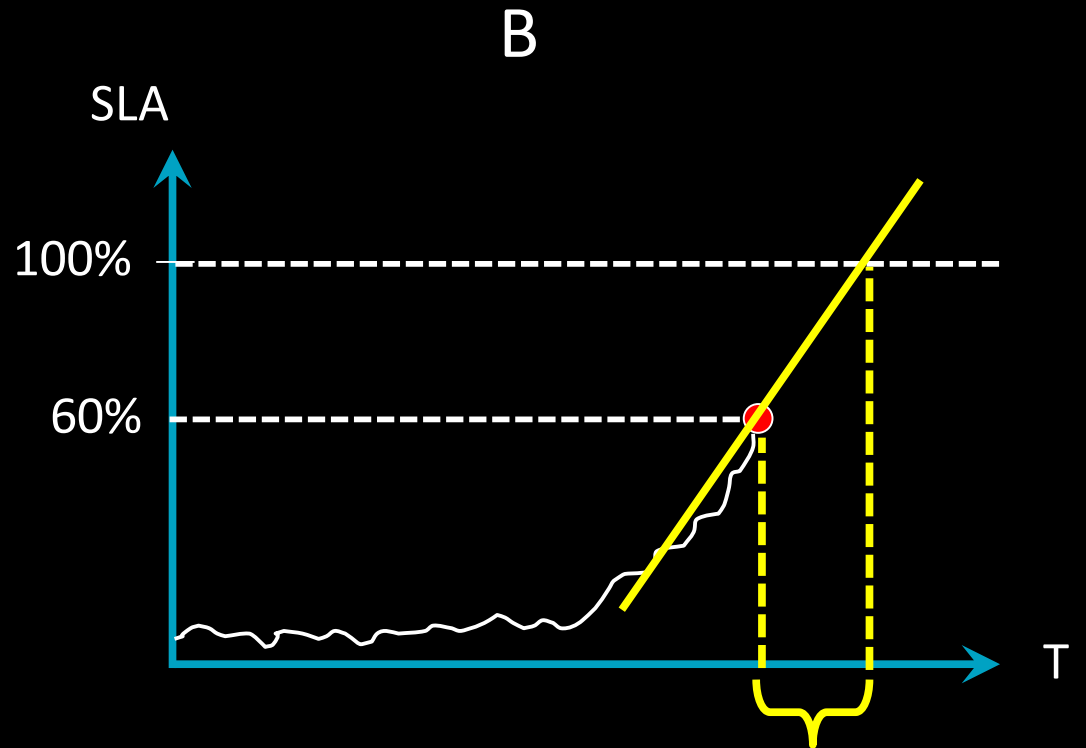
A



B



# THE MOST IMPORTANT METRIC



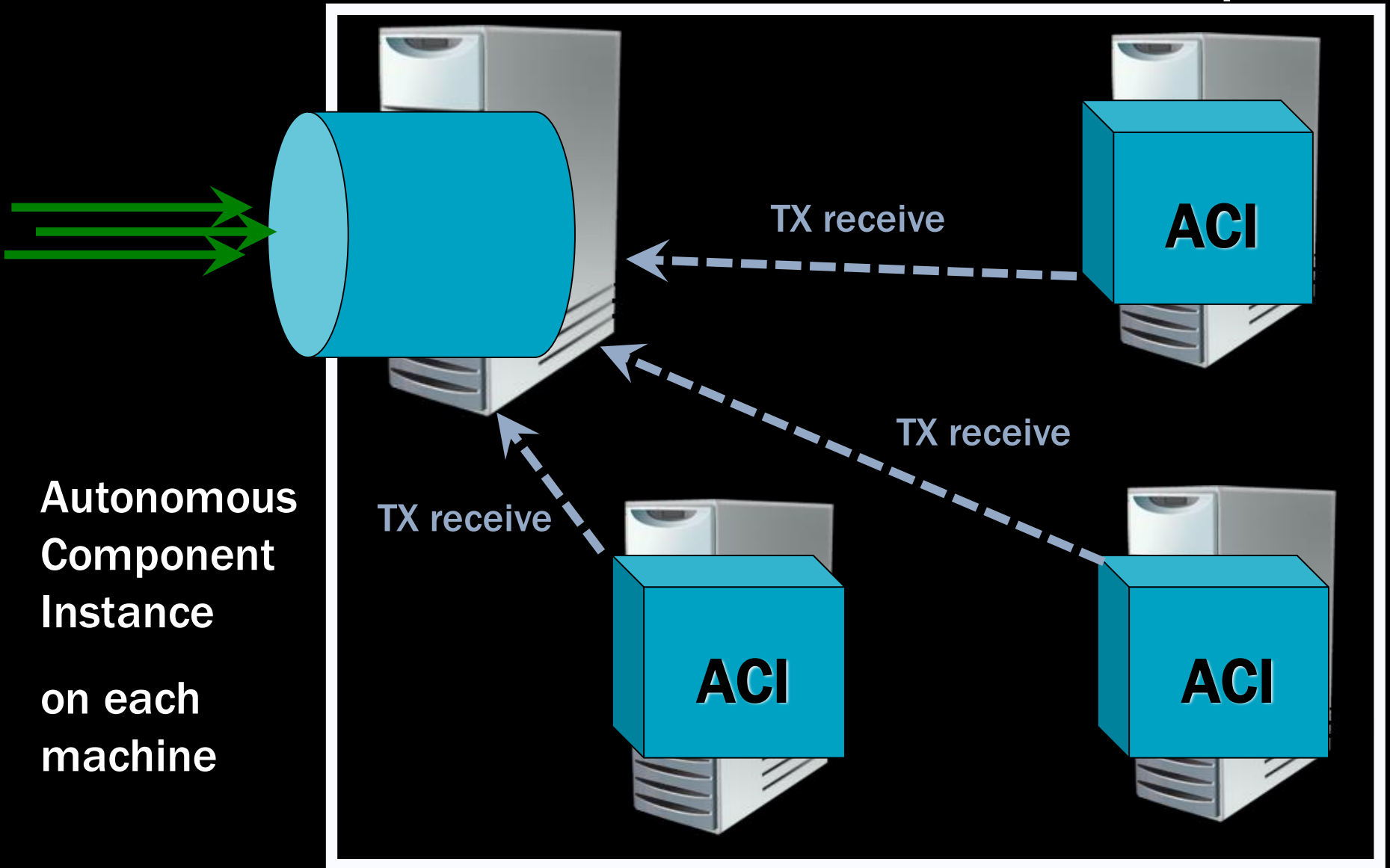


# SCALABILITY

- We can have a number of servers each running an instance of the same autonomous component
- Called the “Competing Consumers” pattern

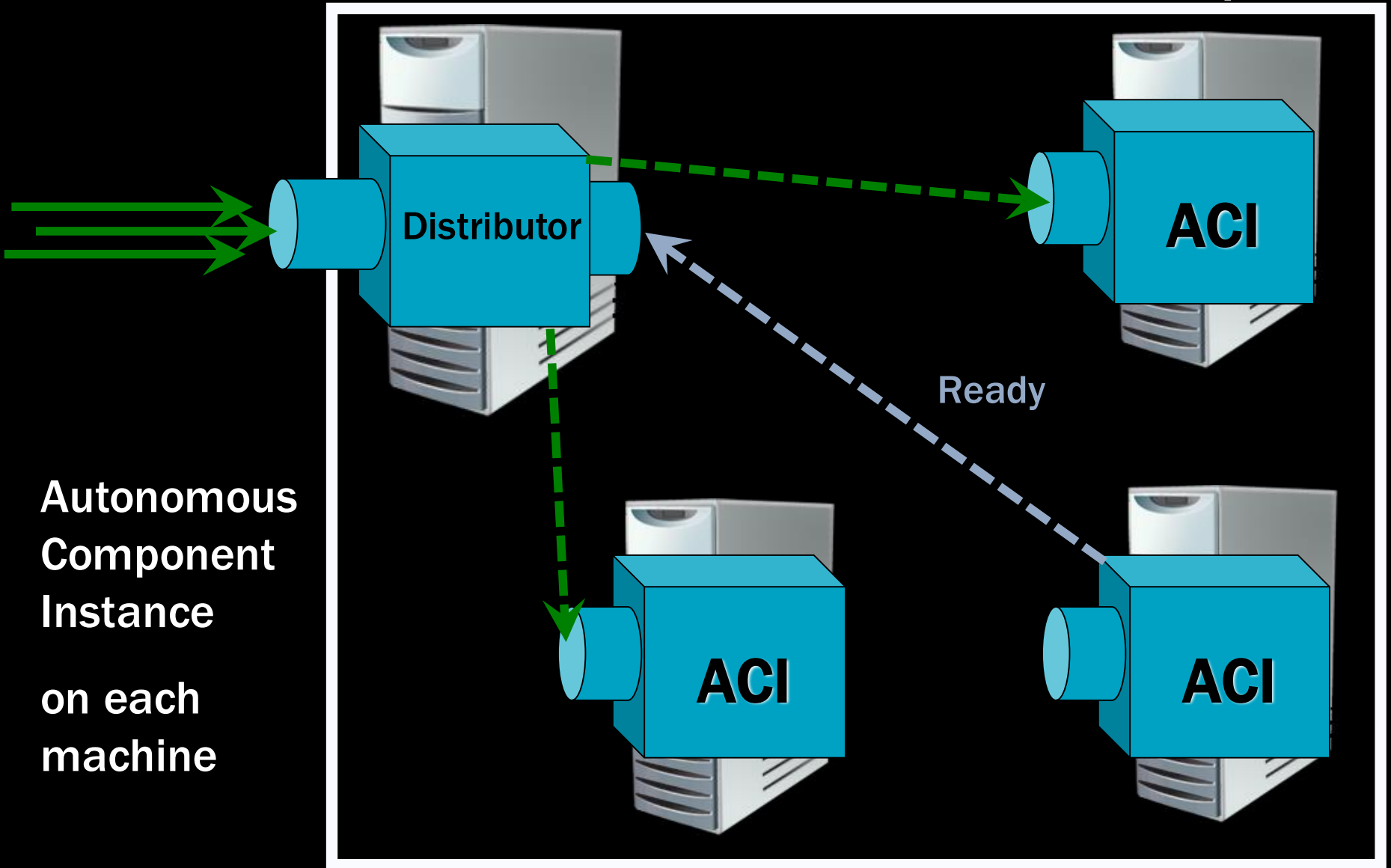
# TRADITIONAL COMPETING CONSUMER

Autonomous Component

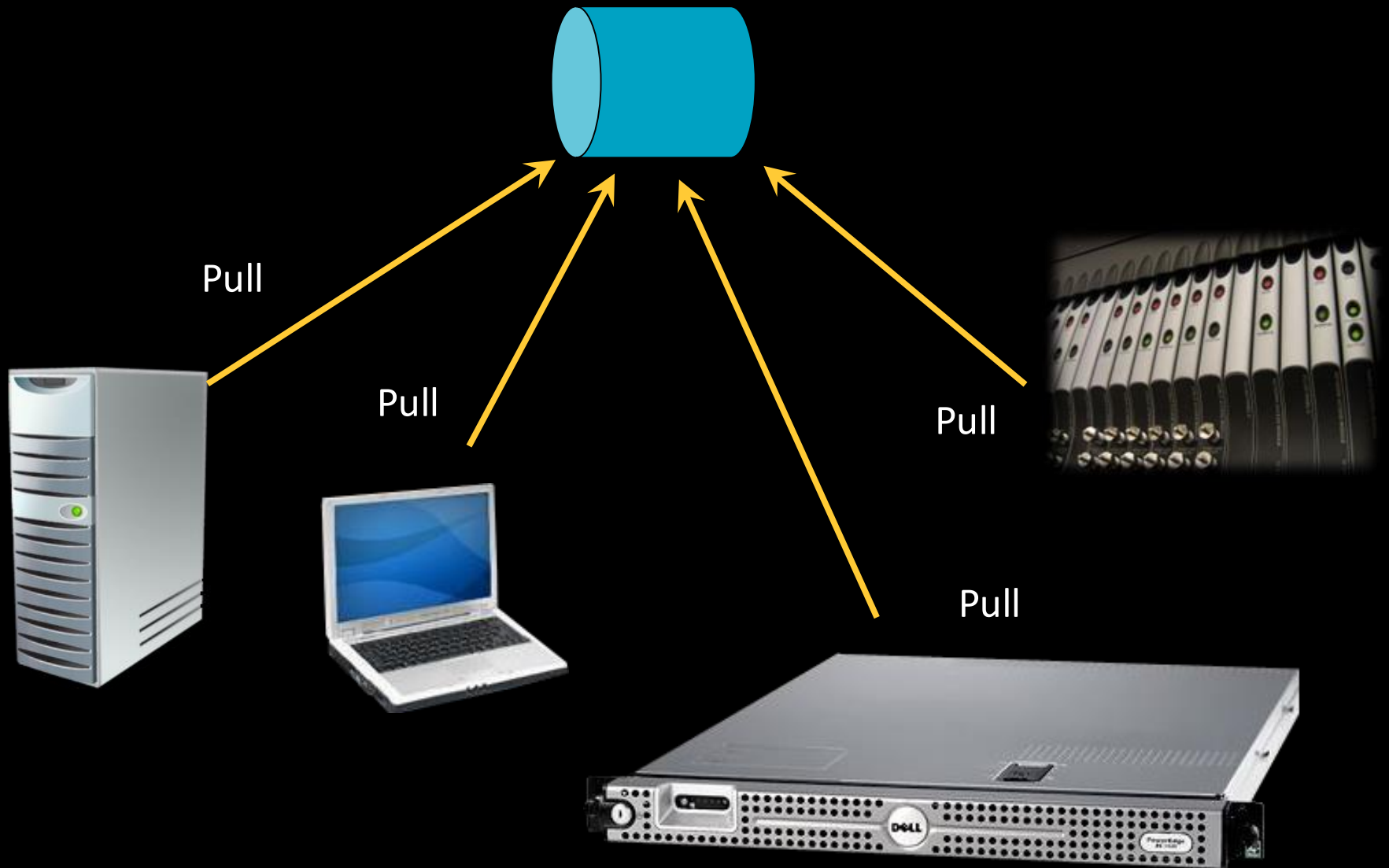


# COMPETING CONSUMER 2: DISTRIBUTOR

Autonomous Component



# HETEROGENEOUS SERVER FARMS



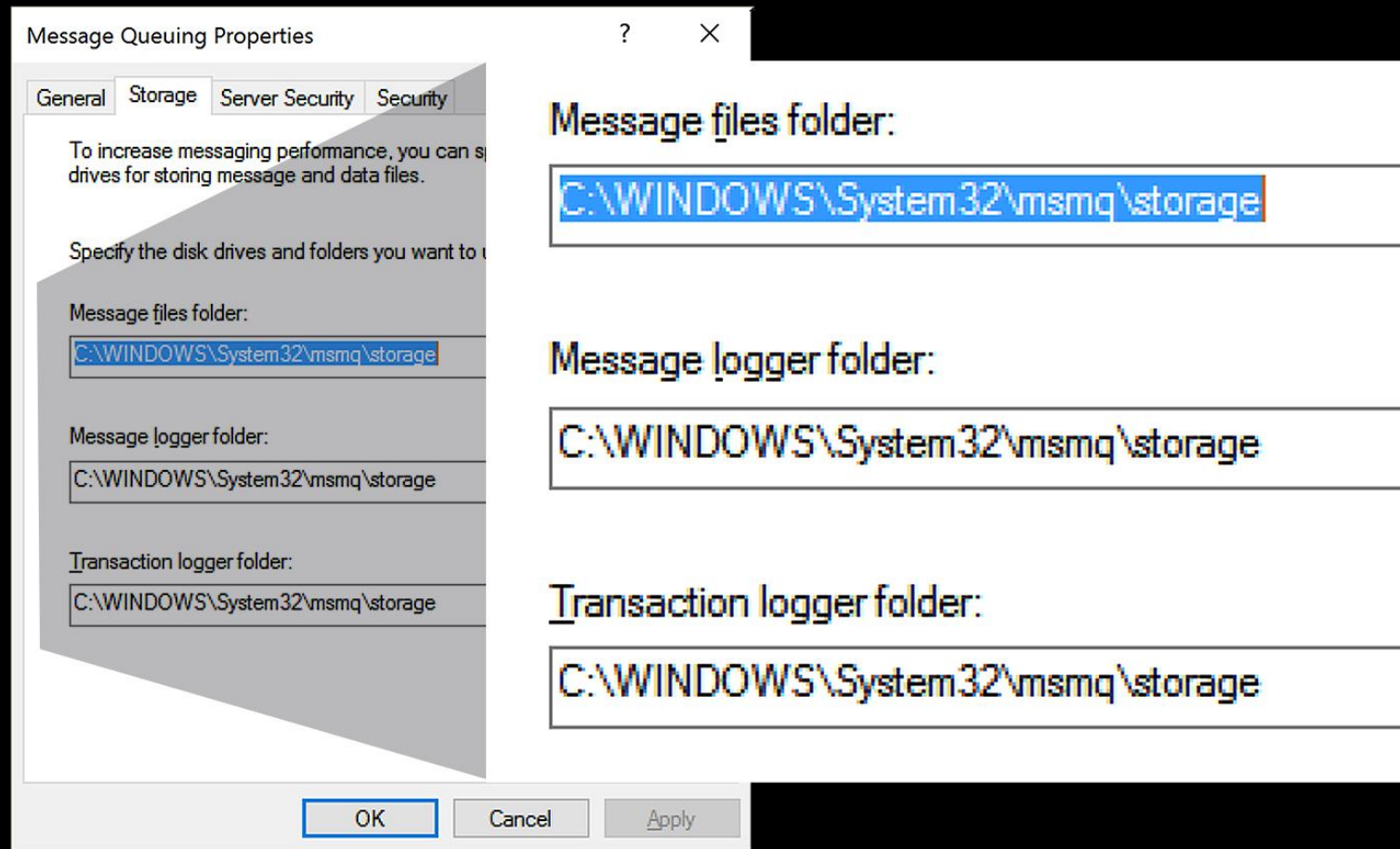
# VIRTUALIZATION – PART 1

- Connecting monitoring and scaling
- When time to violate SLA decreases below  $T$ , call API to provision another vServer
- When time to violate SLA increases above  $T$ , call API to deprovision a vServer
- Self-scaling systems are possible, given enough capacity

# FAULT TOLERANCE – HIGH LEVEL

- Any number of active backups
- Automatic load balancing

# FAULT TOLERANCE – DETAILS



- In a virtualized environment, the C drive is actually stored in a file on the SAN

# VIRTUALIZATION – CONTINUED

- VMWare vSphere handles failover of nodes  
Both for workers and distributor
- Consider fault-tolerant hardware ([Stratus.com](http://Stratus.com)) for greater than 99.999 availability



# BACKUPS & DISASTER RECOVERY

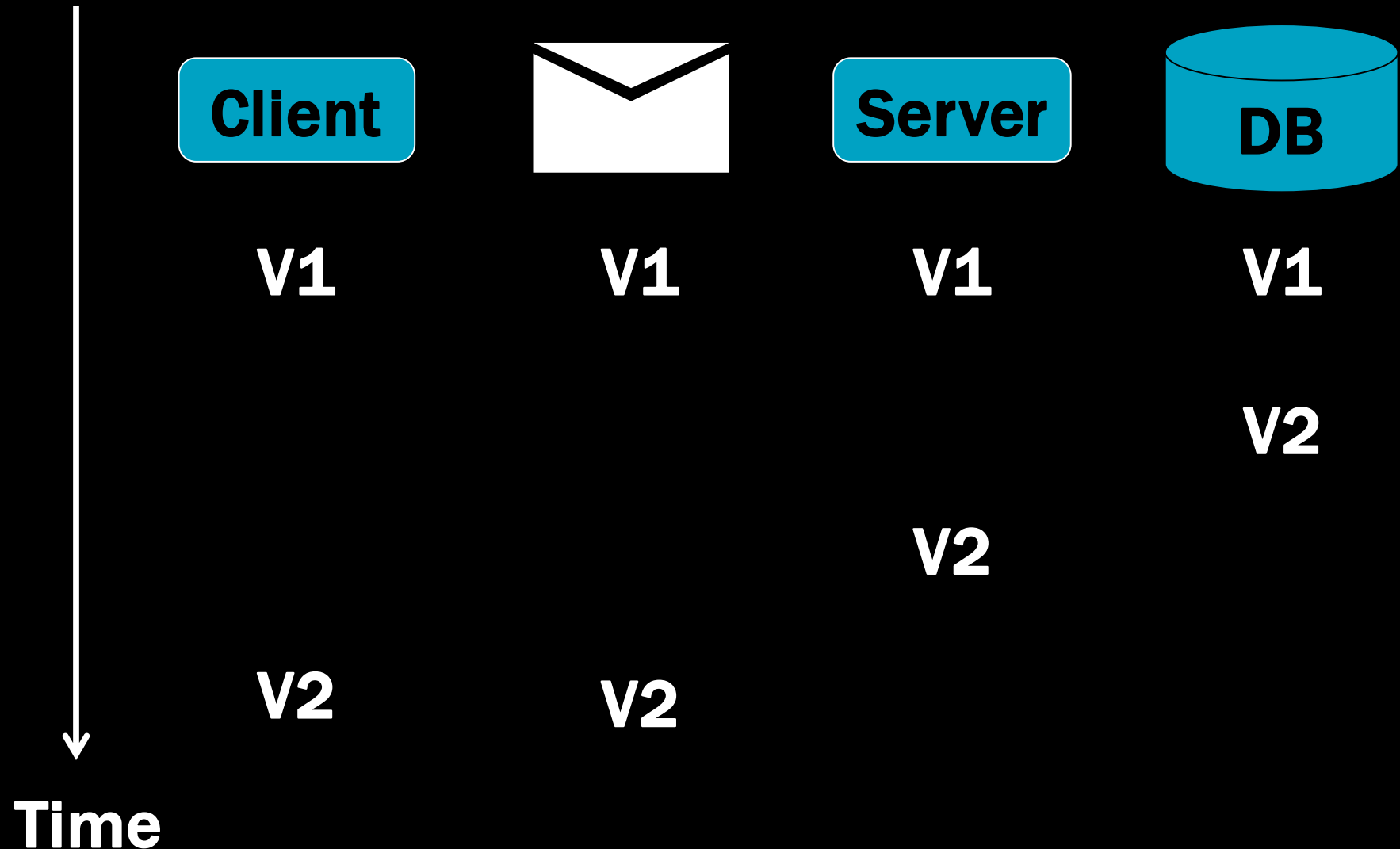
- If you're using a SAN:  
Database data will be stored there  
Queue data will be stored there
- Can use SAN Snapshots to get a fully consistent system-wide backup
- Ship the snapshot to a disaster recover site from time to time

# VERSIONING

- Clients not bound to a specific server implementation by a proxy
- Queues provide temporal separation
- Easy to swap out autonomous component implementation without affecting clients

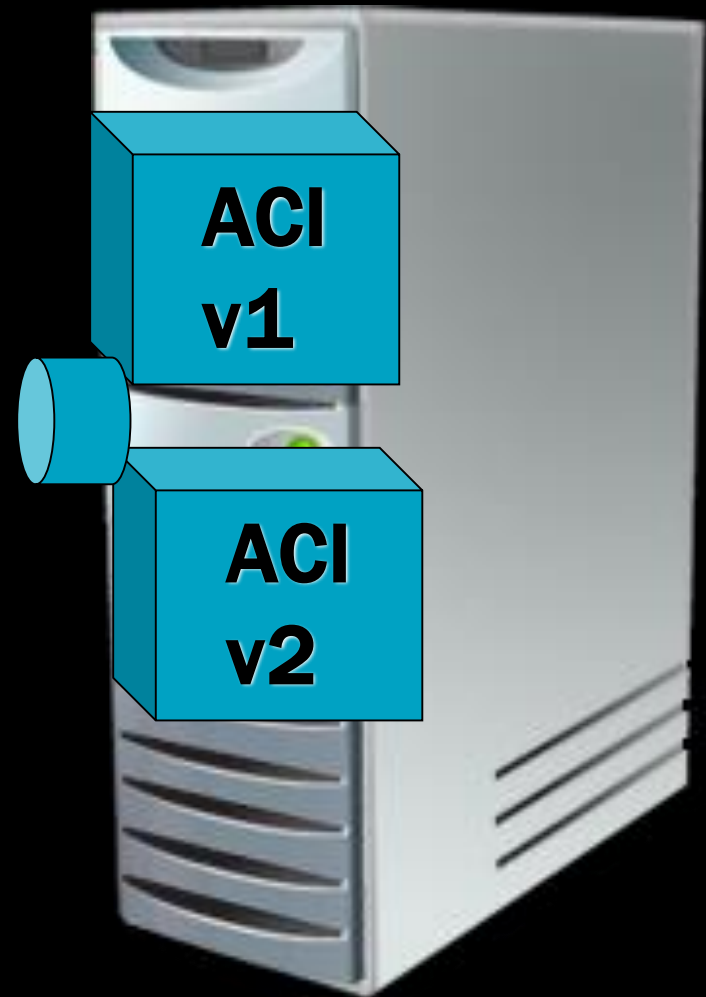
# VERSIONING – STRATEGY

Version from “back to front”



# ZERO-DOWNTIME UPGRADES

- Install v2 process alongside v1
- Both feeding off the same Q
- Check error queue
- Uninstall v1 process
- Go to next machine
- Automate with scripts
- Consider hooking into CI



# SUMMARY

- Configuration of autonomous components enables rich system capabilities

Flexibility

Scalability

Monitoring

Versioning

# LONG RUNNING PROCESSES

# WHAT IS "PROCESS"?

A process can be described as a set of activities that are performed in a certain sequence as a result of internal and external triggers.

- The most basic process control is: if-then
- More complex processes include state machines

# WHAT IS "LONG RUNNING PROCESS"?

A long running process is a process whose execution lifetime exceeds the time to process a single external event or message.

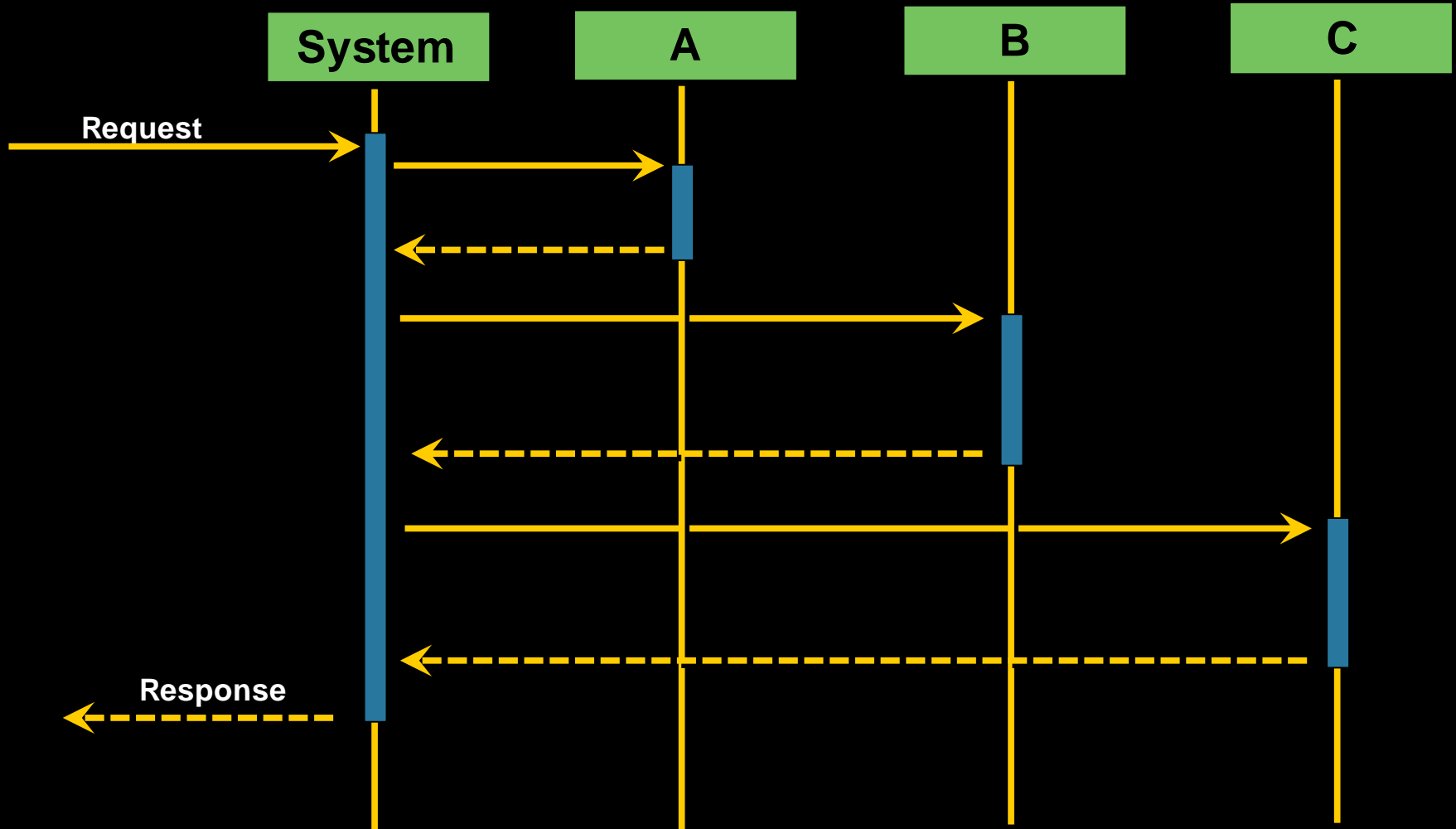
- Long running means that multiple external events/triggers are handled by the same process instance – is **Stateful**
- Derived from "long-lived transactions" work in the late 80's and early 90's



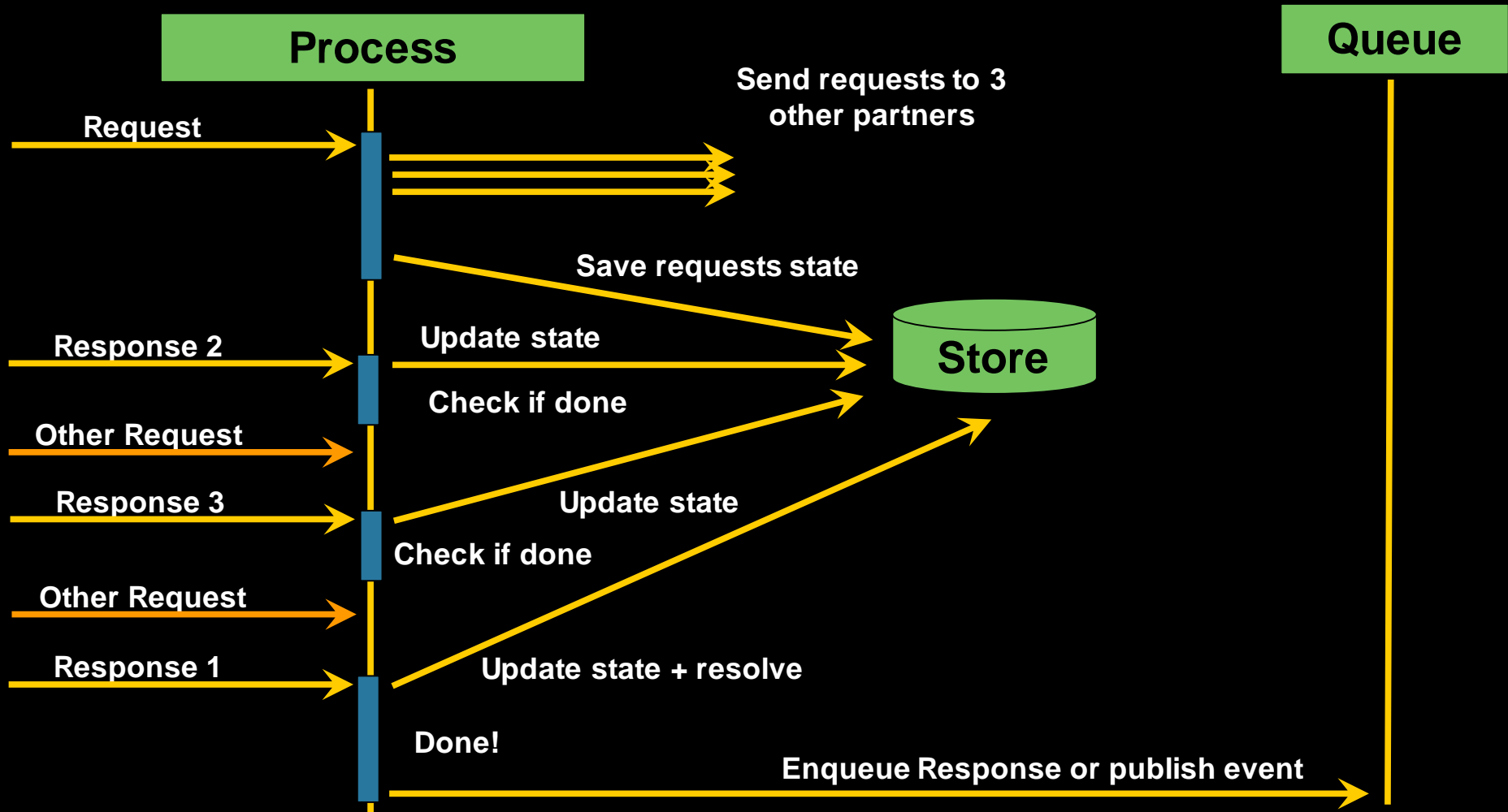
# WHY USE LONG RUNNING PROCESS?

- Long running processes provide a state management facility that enables a system to encapsulate the logic and data for handling an external stream of events.
- It's just good OO programming.

# INTEGRATION EXAMPLE

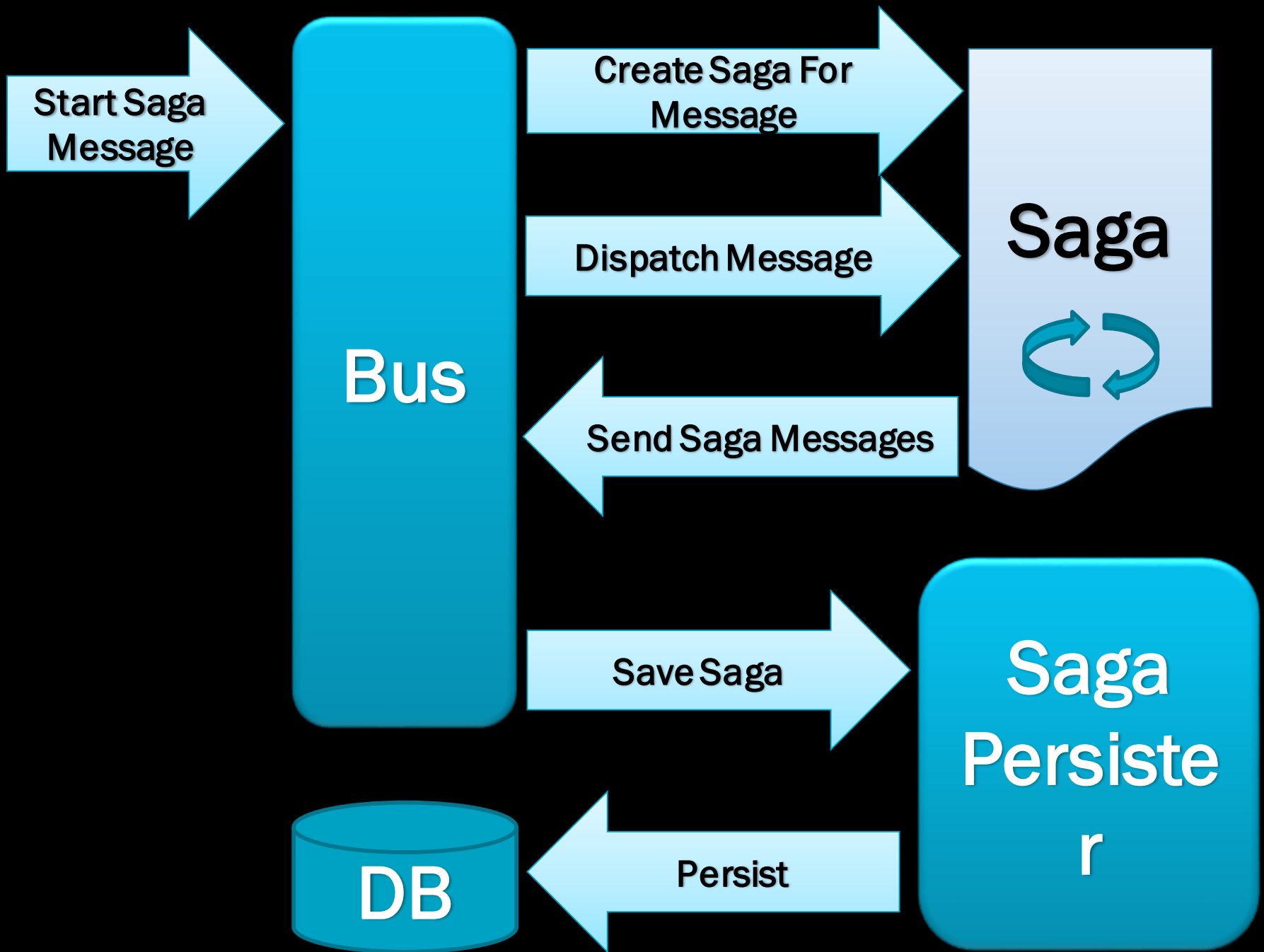


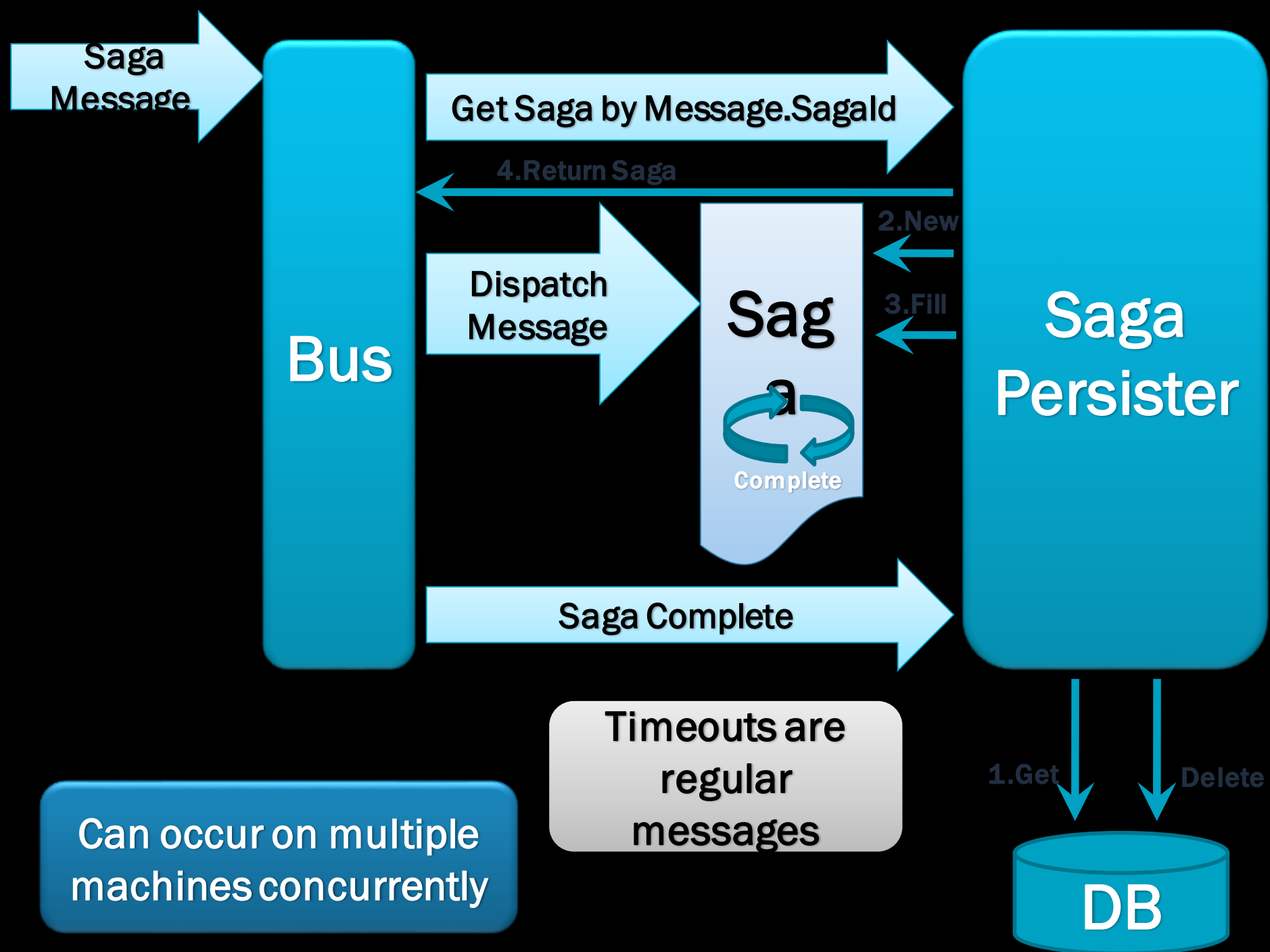
# LONG RUNNING PROCESS IMPLEMENTATION



# SAGAS

- Triggers are messages
- Similar to message handlers
  - Can handle a number of different message types
- Different from message handlers
  - Have state, message handlers don't

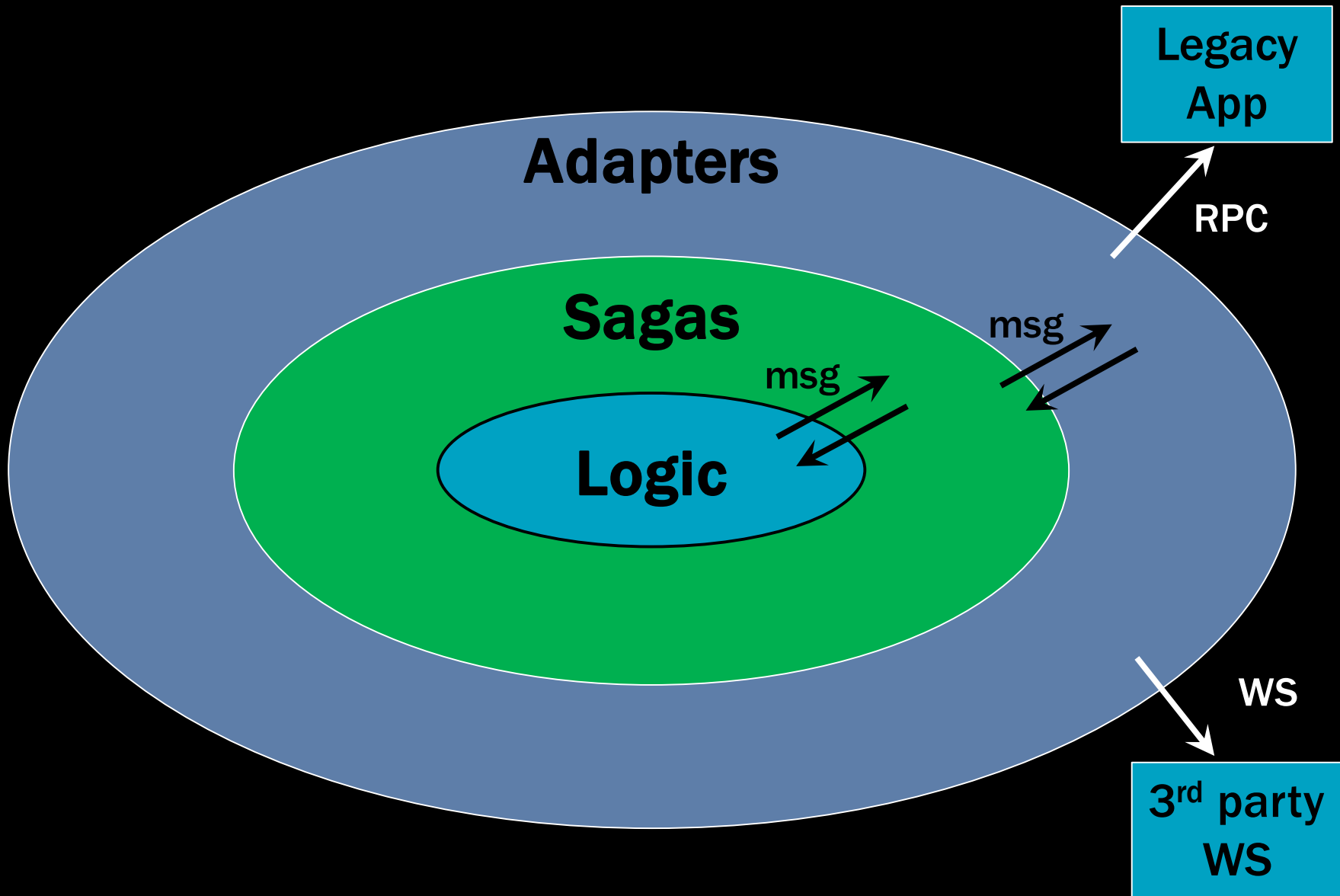




# THE HARD PART

- The easy part is using the building blocks
- The hard part is analyzing the business processes to identify what the steps should be.
- When interacting with legacy systems, use a saga to manage the flow, a separate adapter for the integration.

# SIMILAR TO HEXAGONAL ARCHITECTURE





# WORKFLOW & ORCHESTRATION

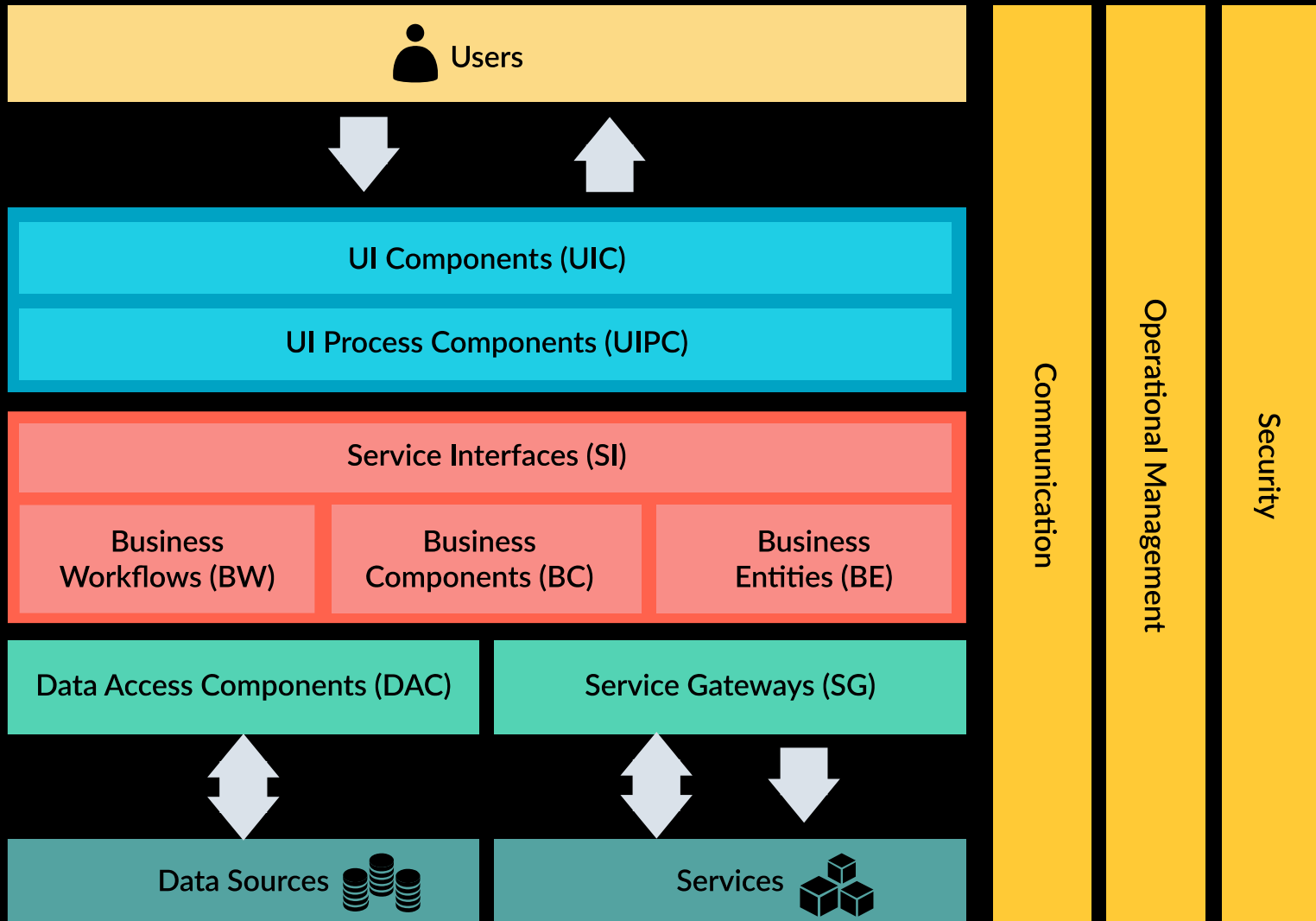
- **Orchestration is not a service by itself.**
- Divide up workflows/orchestrations along service boundaries
  - Events are published at the end of the sub-flow in a service
  - Events trigger a sub-flow in other services
- Sagas can be used for CEP/ESP:  
complex event processing, event-stream proc.

# SUMMARY

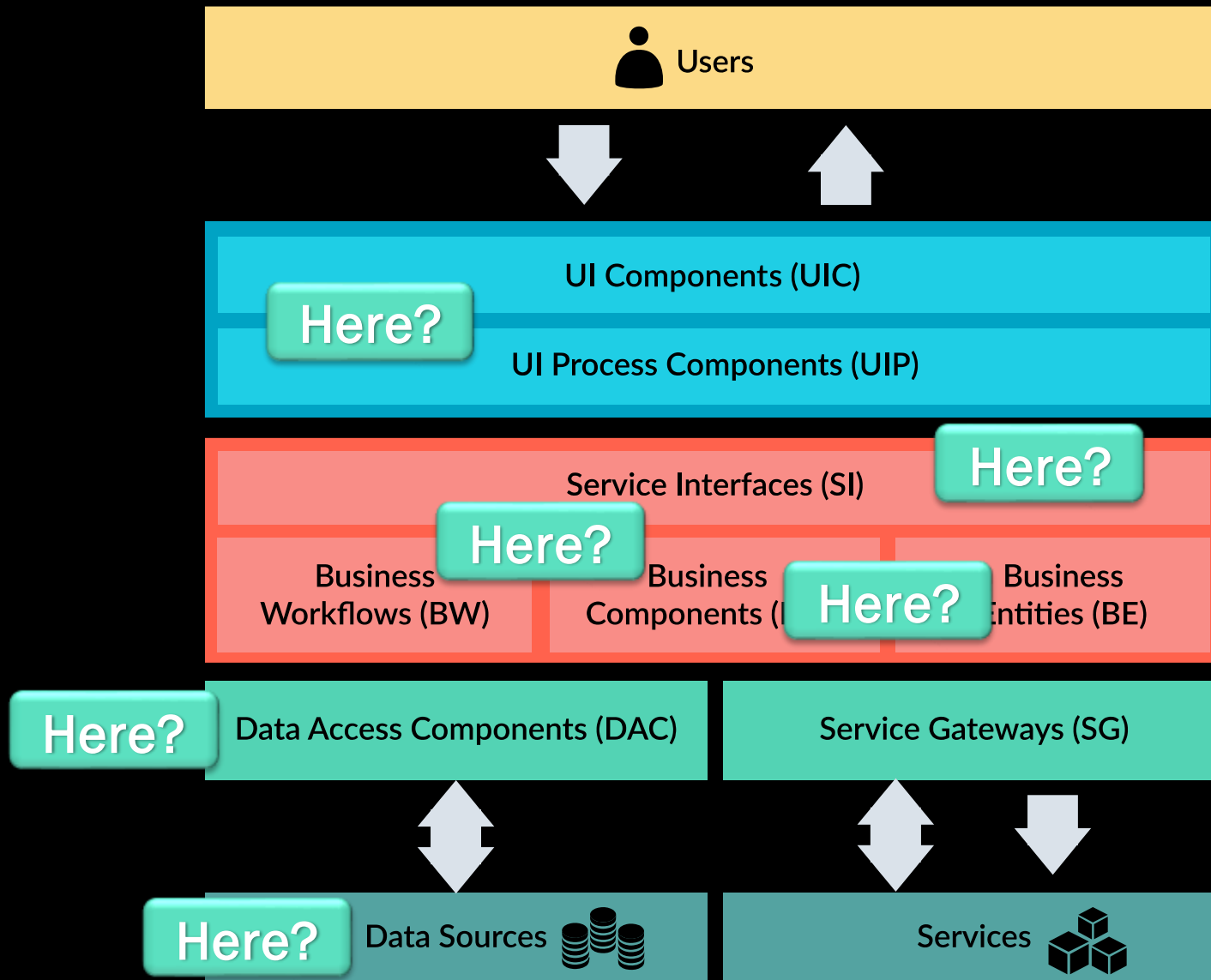
- Use messaging building blocks to support long running processes.
- Unit testing is critical for time-bound processes
- Keep service boundaries explicit

# SERVICE LAYER DOMAIN MODEL INTERACTIONS

# TRADITIONAL ARCHITECTURE



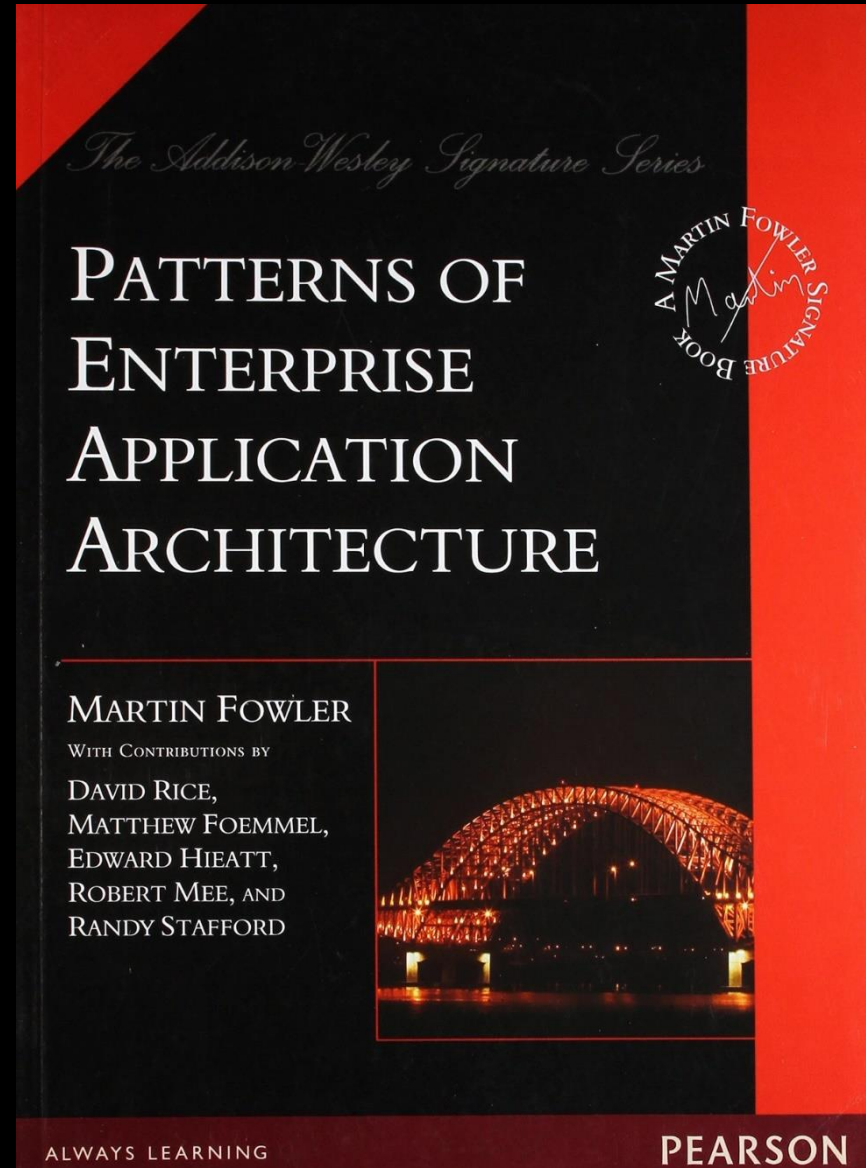
# BUT WHERE DO BUSINESS RULES GO?



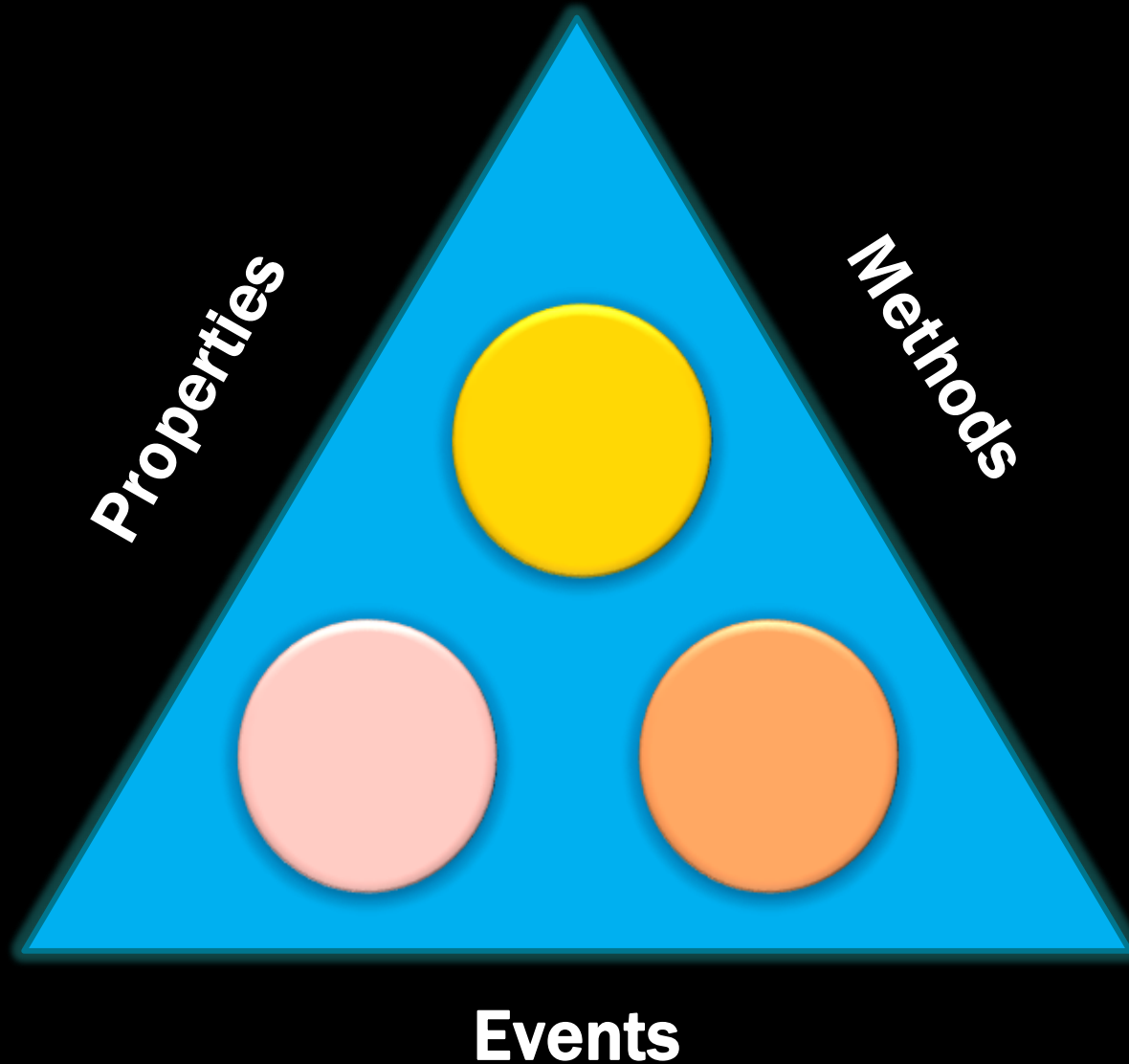
# THE BOOK

that changed everything

... and the pattern that  
battles complexity...



# THE DOMAIN MODEL PATTERN



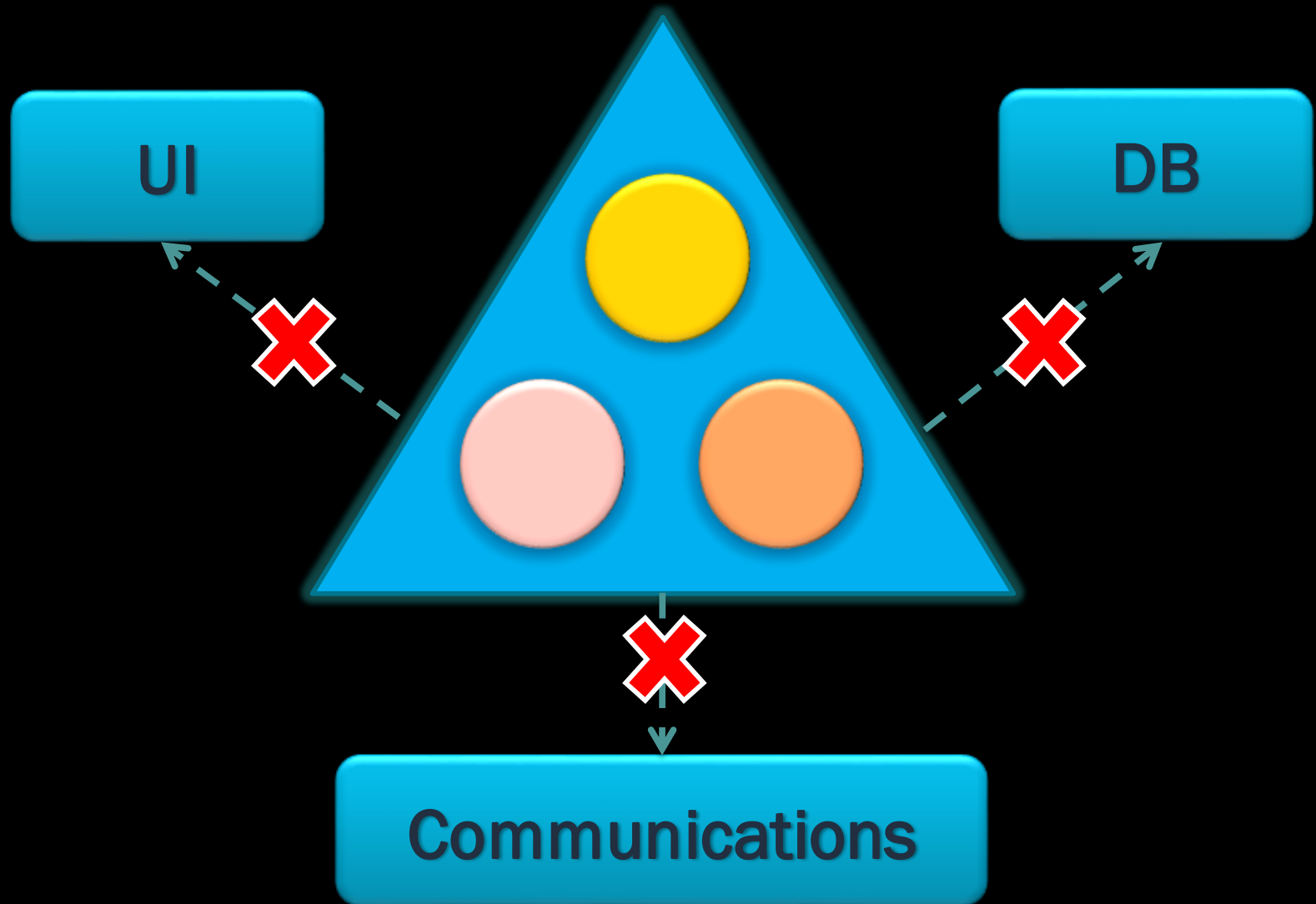
# WHEN TO USE IT, WHEN NOT TO

- “If you have complicated and everchanging business rules...”
- “If you have simple not-null checks and a couple of sums to calculate, a Transaction Script is a better bet”

p119 Patterns of Enterprise Application  
Architecture



# INDEPENDENT OF ALL CONCERNS



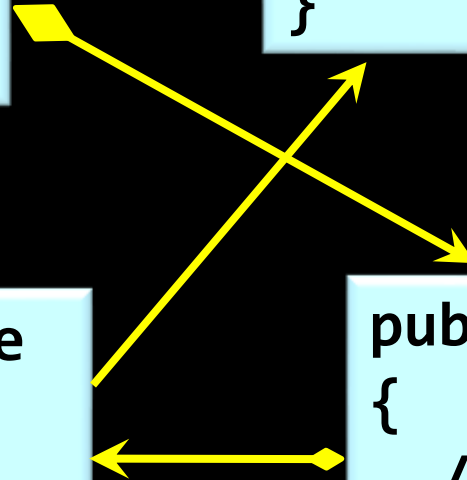
# DOMAIN MODELS MADE OF "POJO"S

```
public class Customer
{
    // properties
    // methods
}
```

```
public class Product
{
    // properties
    // methods
}
```

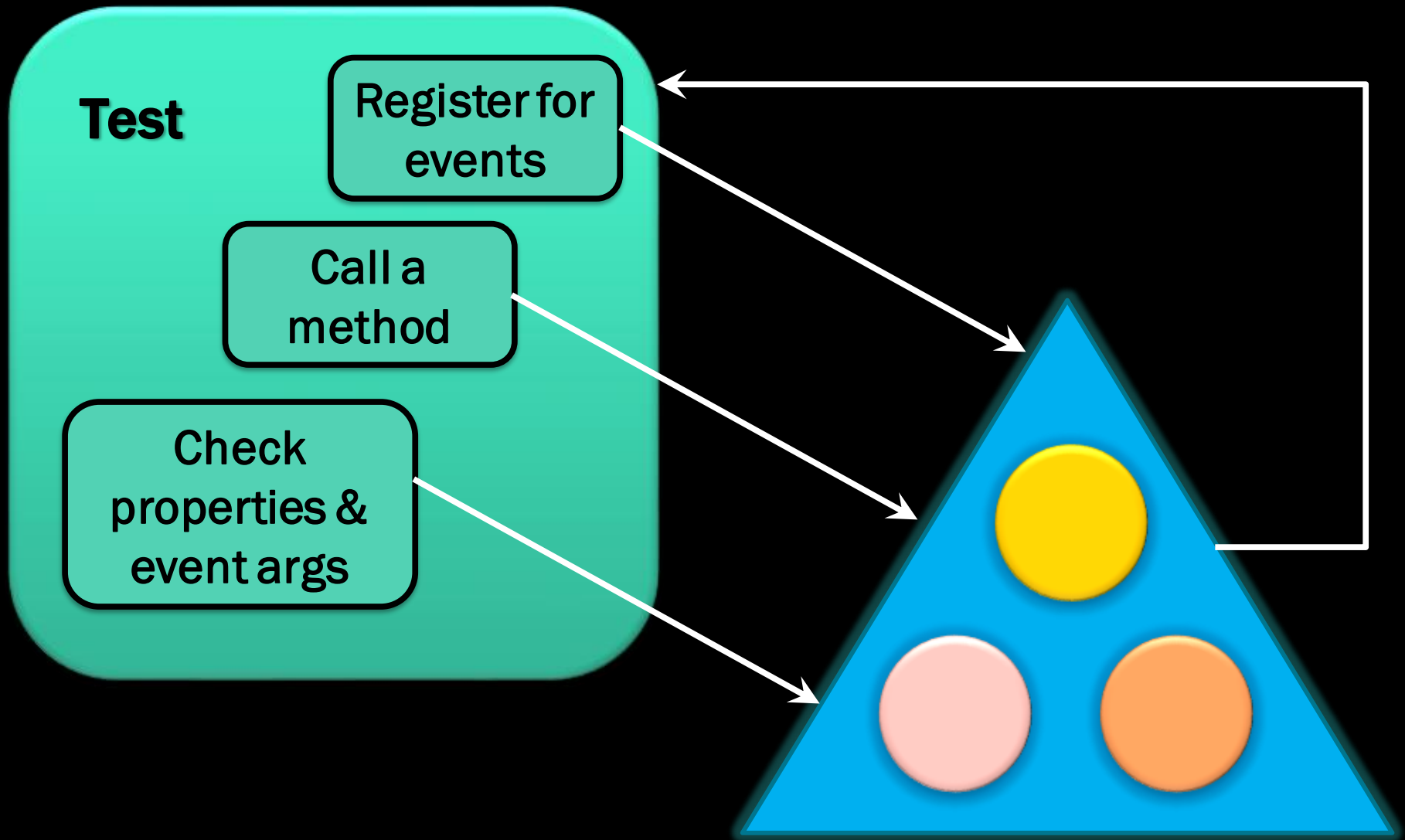
```
public class OrderLine
{
    // properties
    // methods
}
```

```
public class Order
{
    // properties
    // methods
}
```



**But not every bunch of POJOs is a domain model**

# HIGHLY TESTABLE



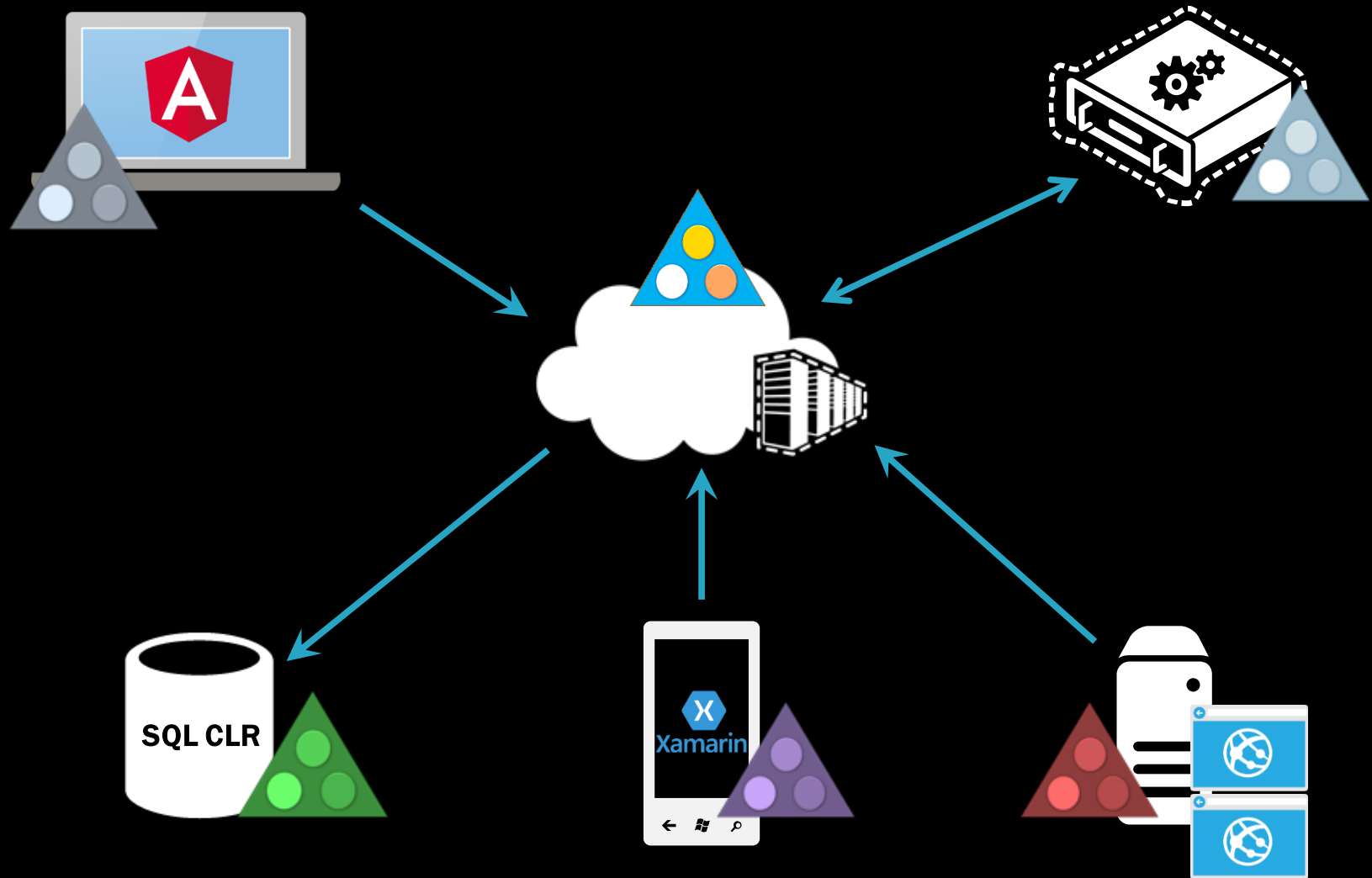
# UNIT TEST SAMPLE

```
[TestMethod]
public void CreateOrderShouldConnectToCustomer()
{
    Customer c = new Customer();
    Order o = c.CreateOrder();

    Assert.AreEqual(c, o.Customer);
}
```

# CAN BE DEPLOYED MULTI-TIER

Different types of domain models for different tiers



# CONCURRENCY MODELS



**Optimistic**



**Realistic**

**Pessimistic**



# MULTI-TABLE SPANNING TRANSACTIONS

Time	TX 1	TX 2	TX 3
T0	Begin TX	Begin TX	Begin TX
T1	Update A	Read A	Update B
T2		Read B	
T3	Commit	Use values of A and B to Update C	Commit
T4		Commit	

**Transactions can get an inconsistent picture!**

# REALISTIC CONCURRENCY

Need to get a current set of data  
before changing anything

Get Domain Object

Ask it to update itself

Domain Object runs business rules

If successful, updates its own state





# SERVICE LAYER SAMPLE CODE

```
public void Handle(MakeCustomerPreferredMsg m)
{
    Customer c = s.Get<Customer>(m.CustomerId);
    c.MakePreferred();
}
```

Necessary to address concurrency



# DOMAIN MODEL SAMPLE : CUSTOMER

```
public void MakePreferred()  
{  
    foreach(Order o in this.UnshippedOrders)  
        foreach(Orderline ol in o.OrderLines)  
            ol.Discount(10.Percent);  
}
```



**Entity relationships expose us to possible inconsistency!**

# REALISTIC CONCURRENCY

Some changes can be concurrent, others can't (?)

- ❖ You change the customer's address
- ❖ I update the customer's credit history
- ❖ You cancel an order
- ❖ I try to ship the order



**Race condition**

# EVEN IF THEY'RE EASY TO IMPLEMENT

```
public class Order
{
    public void Cancel()
    {
        if (status != OrderStatusEnum.Shipped
            //cancel
        )
    }

    public void Ship()
    {
        if (status != OrderStatusEnum.Cancelled
            //ship
        )
    }
}
```

# REMEMBER

- In CQRS, commands don't fail
- Race conditions don't exist in business
- A microsecond shouldn't change business objectives

# FIND UNDERLYING BUSINESS OBJECTIVES

Rules:

1. Cannot cancel shipped orders

Why?

Because shipping costs money

So?

That money would be lost  
if the customer cancelled

Why?

Because we refund the customers money

2. Don't ship cancelled orders

Refund Policies

# ANALYZE

- When an order is cancelled, does the refund need to be given immediately?

No

- Can we give a partial refund?

Yes

# DIG DEEPER

- What does a customer have to do in order to get a refund?

Return the products

\* Most orders cancelled soon after being made



# CONSIDER SERVICE BOUNDARIES

Cancel Order

**Sales**

Ship Products  
Products Returned

**Shipping**

Refund Policy

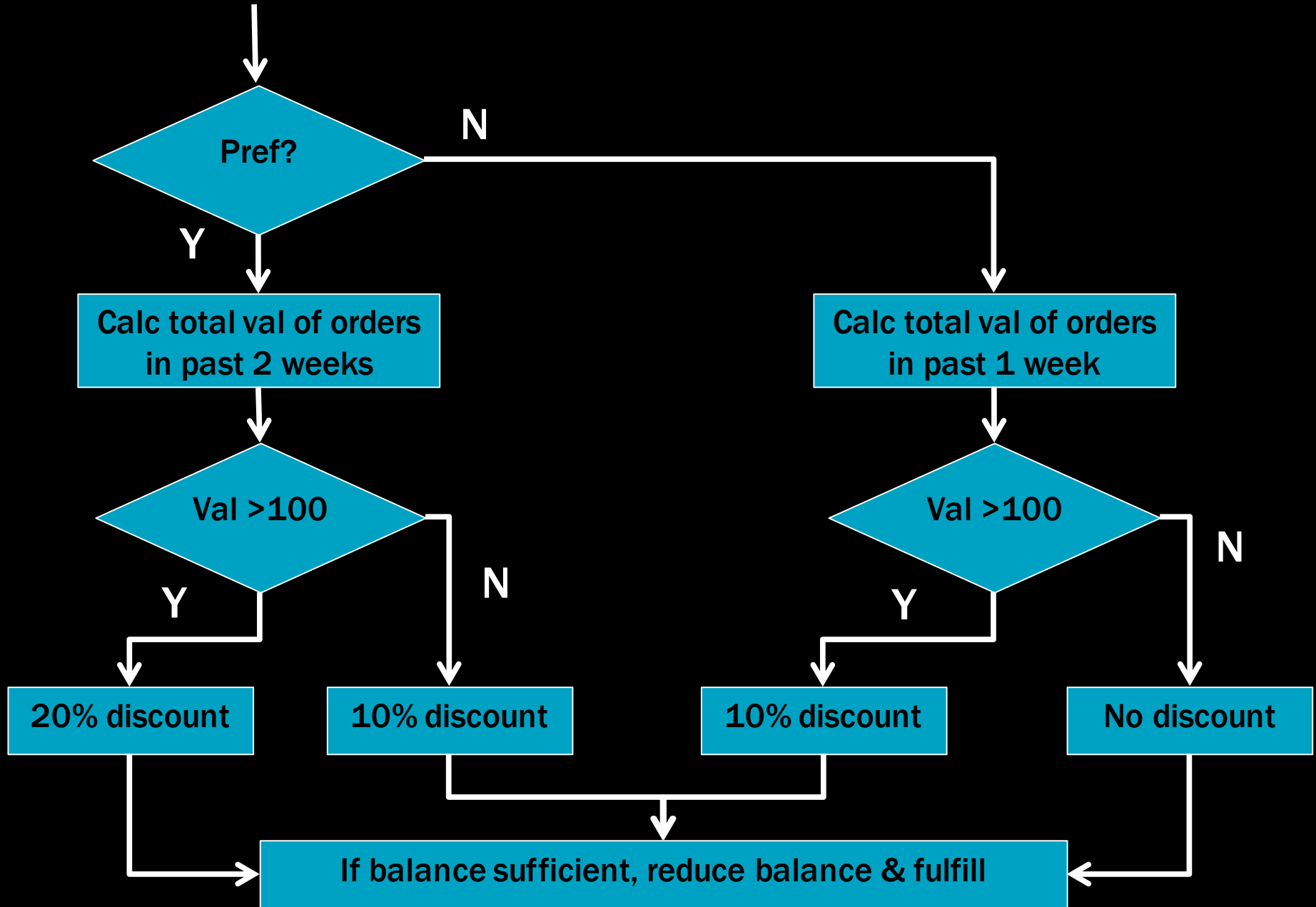
**Billing**

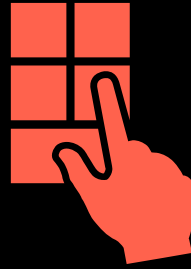
**Which services require a domain model?**

**Refund policy driven by multiple messages**

DOMAIN MODELS ARE SAGAS

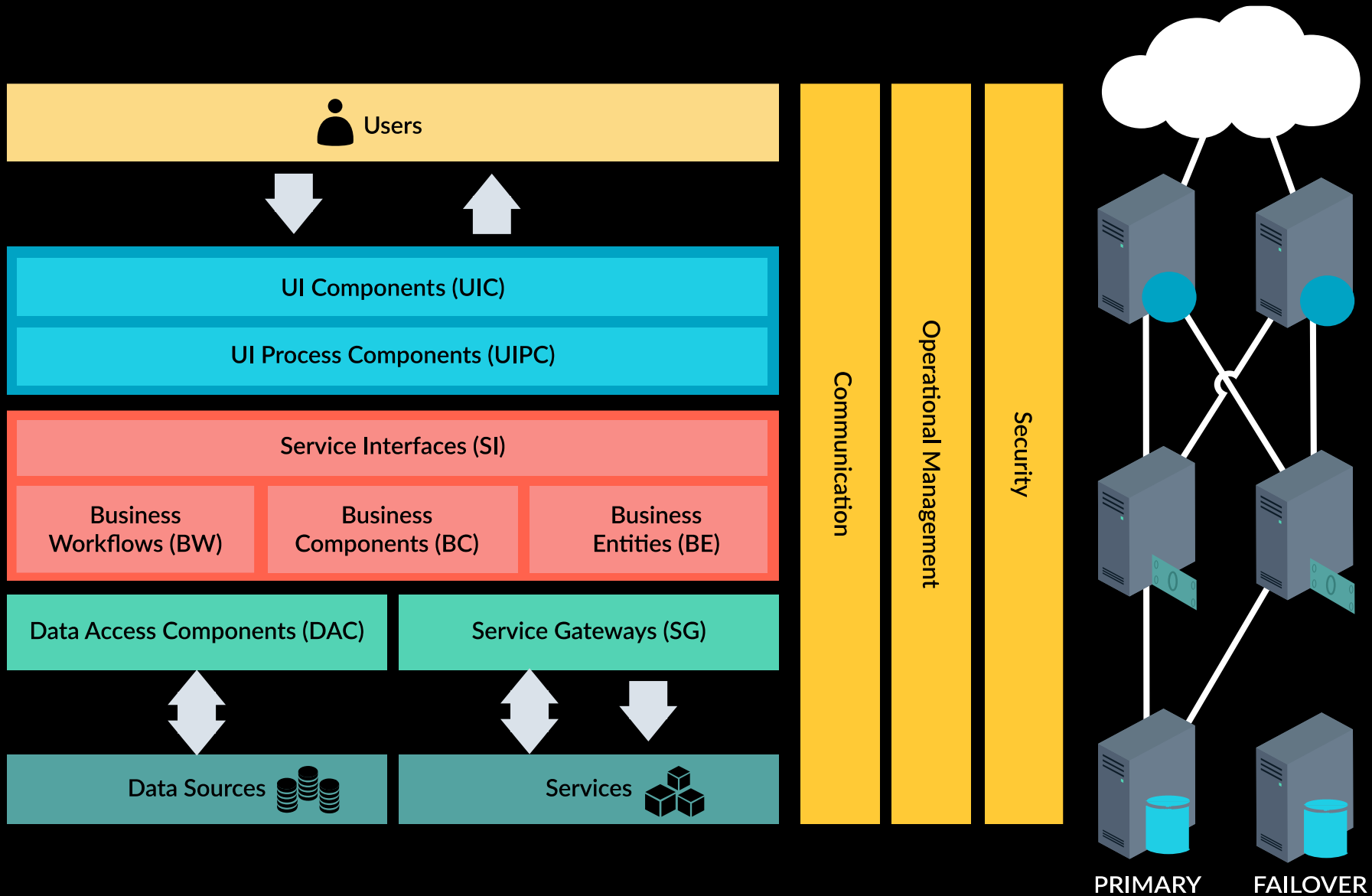
# BUY ITEM (IN-APP PURCHASE)



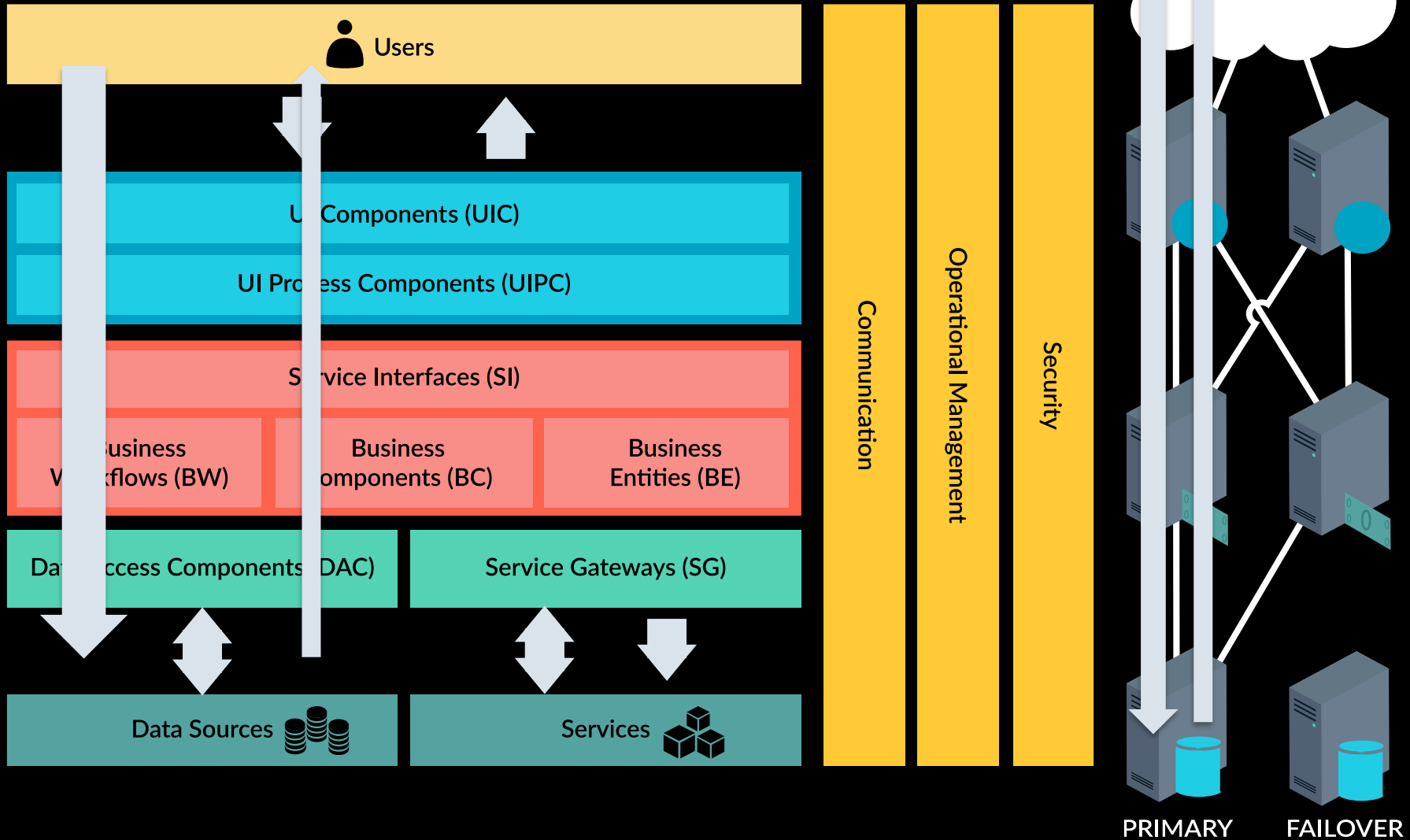


# WEB SERVICES & USER INTERFACES

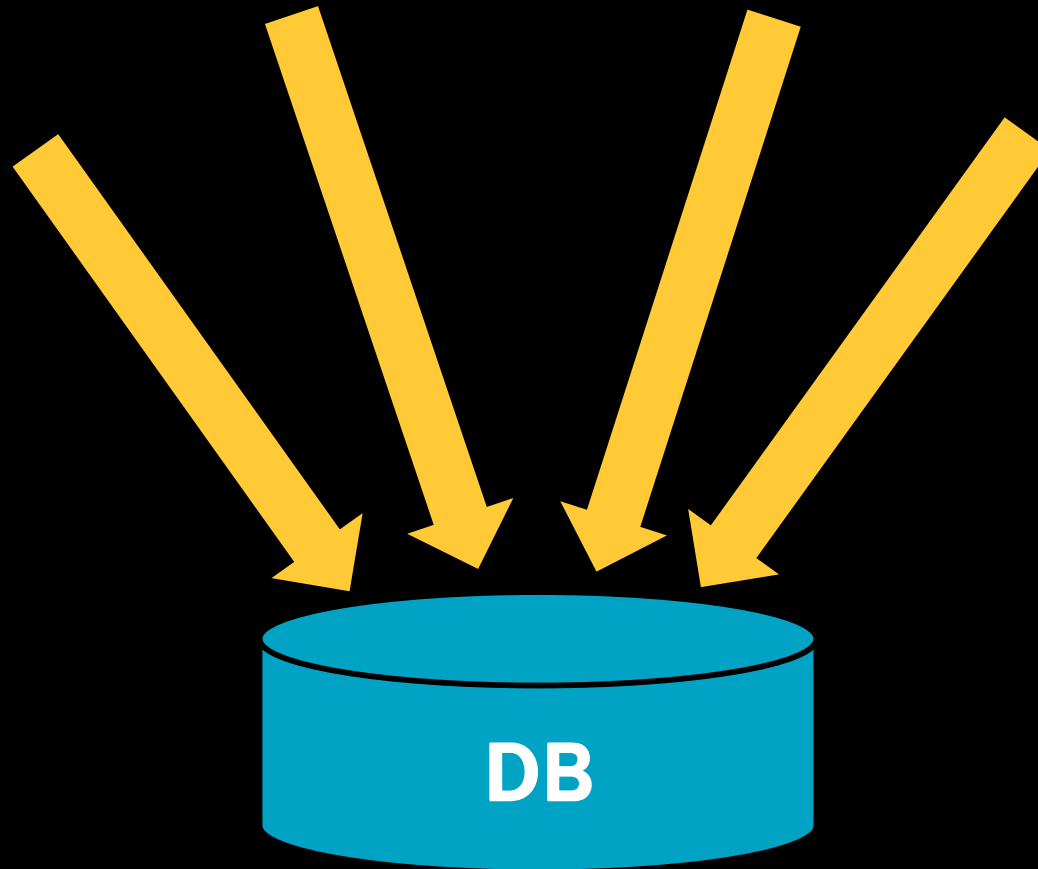
# COMMON WEB ARCHITECTURE



# SYNC REQUEST/RESPONSE



# SCALING OUT THE WEB TIER

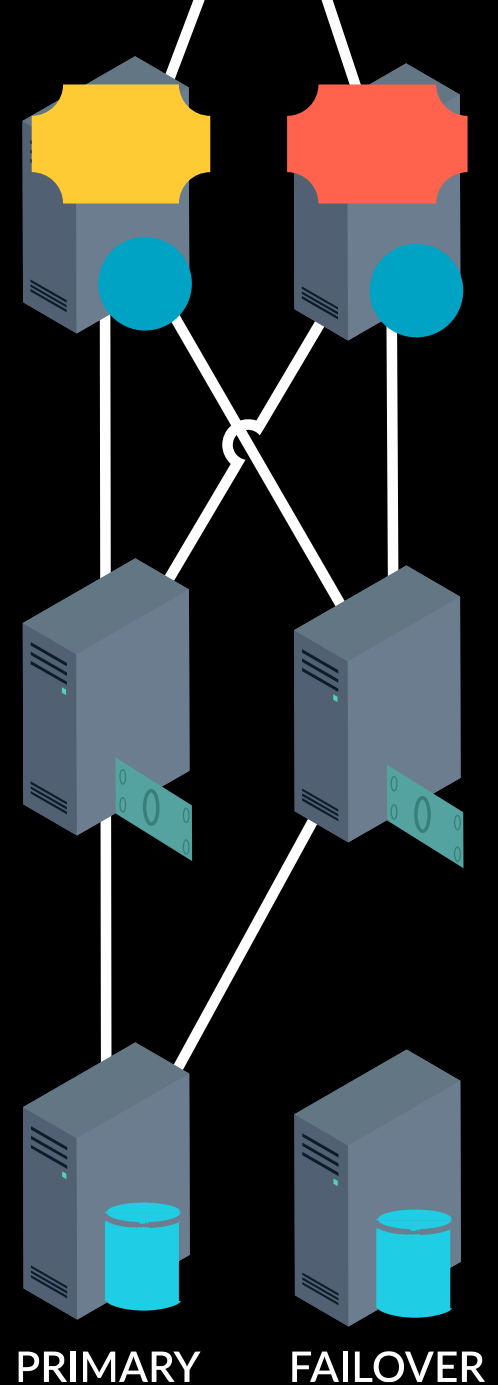


Just shows that the DB is the bottleneck

# CACHING – IN PROCESS

Keeping the cache up to date  
across farms is challenging

Often requires “sticky sessions”  
and that undermines load balancing





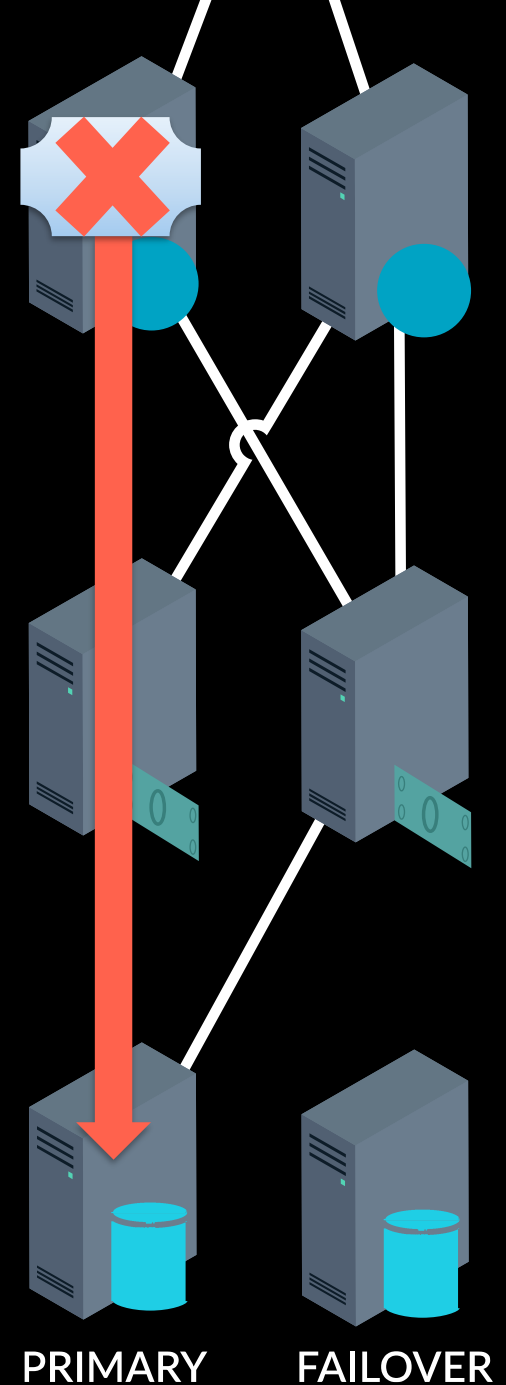
# DISTRIBUTED CACHING

Not all data is kept on each node

Beware the temptation to “loop”

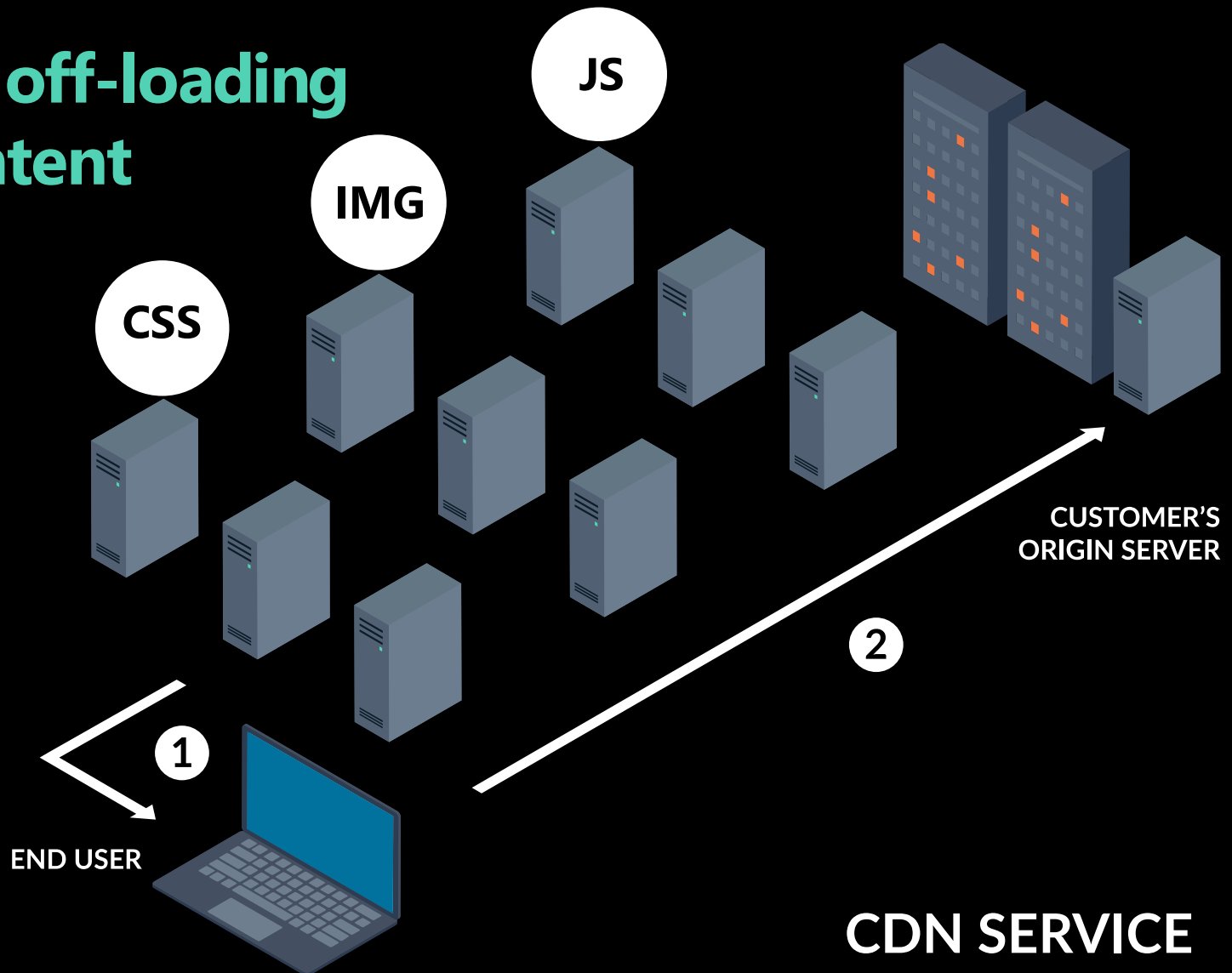
# CACHE INVALIDATION

Reads can interfere with writes  
and that hurts performance



# CONTENT DELIVERY NETWORKS

Great for off-loading  
static content



# ALMOST THERE...



Don't use a smart donkey when what you need is a heavy ox

# AMAZON.COM

It's all dynamic

It's all static

amazon.com

Hello. Sign in to get [personalized recommendations](#). New customer? [Start here](#).

FREE 2-Day Shipping, No Minimum Purchase

Your Amazon.com



Today's Deals

Gifts & Wish Lists

Gift Cards

Your Account | Help

Search All Departments

GO

Cart

Your Lists

Shop All Departments

- Books
- Movies, Music & Games
- Digital Downloads
- Computers & Office
- Electronics
- Home & Garden
- Grocery, Health & Beauty
- Toys, Kids & Baby
- Apparel, Shoes & Jewelry
- Sports & Outdoors
- Tools, Auto & Industrial

Check This Out



Winterize Your Vehicle

Get ready for winter road travel.



Intel Centrino 2 Laptops Store

High performance laptops for a mobile lifestyle.

Selling on Amazon



This summer, Oprah received a gift that she says changed her life. "I'm telling you, it is absolutely my new favorite thing in the world," she says.

Meet Amazon Kindle, a wireless portable reading device with instant access to more than 190,000 books, blogs, newspapers and magazines.

amazonkindle [Learn more](#)

Amazon Daily BLOG

11 posts since yesterday [Read posts](#)

Buy 2 DS Games, Get a 3rd Free

3-for-2 image

Buy two essential games for Nintendo's DS and get number three for free at checkout. Taxes and shipping may apply.

[Learn more](#)

ADVERTISEMENT

introducing

mino  
now playing: you™

Sophisticated Style for Men



[Omega Men's Speedmaster Professional](#)



[Sterling Silver Black Accent Rectangular Cufflinks](#)



[8mm Black Titanium Band](#)



# THE STATIC DYNAMIC WEB

\* Don't do this on your home

page

Layout.CSS

Layout.JS

Dept.XML

Kindle.XML

The image is a screenshot of the Amazon.com homepage from around 2005. It features a top navigation bar with the Amazon logo, a search bar, and links for sign-in, account, and cart. Below the navigation bar, the main content area is divided into several sections. On the left, a vertical sidebar lists various product categories. The main content area includes a large advertisement for the Amazon Kindle, a section for 'Sophisticated Style for Men' featuring watches, cufflinks, and a ring, and a section for 'Check This Out' with links to winterize a vehicle and Intel Centrino 2 laptops. On the right side, there is a 'Blog' section and an 'Advertisement' for the Mino video player. Several blue labels are overlaid on the image, pointing to specific sections: 'Layout.CSS' points to the top navigation bar, 'Layout.JS' points to the top right area, 'Dept.XML' points to the left sidebar, and 'Kindle.XML' points to the Kindle advertisement. A yellow box highlights the left sidebar, and another yellow box highlights the Kindle advertisement. A third yellow box highlights the 'Check This Out' section. A fourth yellow box highlights the 'Sophisticated Style for Men' section. A fifth yellow box highlights the 'Blog' section. A sixth yellow box highlights the 'Advertisement' section.

amazon.com

Hello. Sign in to get personalized recommendations

Your Amazon.com

Search All Departments

GO

Cart

Your Lists

Shop All Departments

Digital Downloads

Computers & Office

Electronics

Home & Garden

Grocery, Health & Beauty

Toys, Kids & Baby

Apparel, Shoes & Jewelry

Sports & Outdoors

Tools, Auto & Industrial

Kindle

Meet Amazon Kindle, a wireless portable reading device with instant access to more than 190,000 books, blogs, newspapers and magazines.

amazonkindle

Learn more

Sophisticated Style for Men

Omega Men's Speedmaster Professional

Sterling Silver Black Accent Rectangular Cufflinks

8mm Black Titanium Band

Check This Out

Winterize Your Vehicle

Get ready for winter road travel.

Intel Centrino 2 Laptops Store

High performance laptops for a mobile lifestyle.

Selling on Amazon

Amazon Daily BLOG

11 posts since yesterday

Read posts

Buy 2 DS Games, Get a 3rd Free

3-for-2 image

Buy two essential games for Nintendo's DS and get number three for free at checkout. Taxes and shipping may apply.

Learn more

ADVERTISEMENT

introducing mino now playing: you

flip video



# WHAT HAPPENS OVER THE WEB?

**WHAT HAPPENS OVER THE WEB?**

**Cached by browser**

- Layout.CSS
- Layout.JS

**Cached by ISP**

- Dept.XML

**Cached by CDN**

- Kindle.XML

The diagram illustrates the flow of data and caching across the web. A central network graph is overlaid with a large red 'X', indicating a point of failure or a specific network event. Arrows point from various sources to the network graph, representing data flow and caching. The sources include:

- Cached by browser:** Layout.CSS, Layout.JS
- Cached by ISP:** Dept.XML
- Cached by CDN:** Kindle.XML

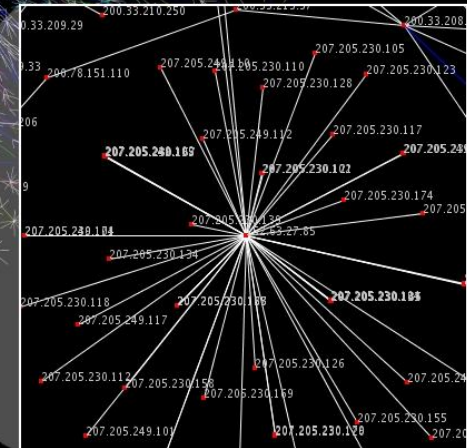
The network graph shows a dense web of connections between nodes, with a large red 'X' marking a specific point of interest. An inset image in the bottom right corner shows a detailed view of a network graph with many nodes and edges, representing a complex network structure.

Layout.CSS  
Layout.JS



## Dept.XML

## Kindle.XML



**Nobody  
even comes  
knocking**





# PERSONALIZATION

## The weather widget

- Look up location by IP
- Look up weather by location

2 back-to-back remote calls to  
costly 3<sup>rd</sup> party services

📍 LONDON 23 MAY 2016

MONDAY  
HUMIDITY 20%  
~23° ~27°  
**23°**  
CLEAR



TUESDAY ☁ 20°  
WEDNESDAY ☁ 18°  
THURSDAY ☁ 15°  
FRIDAY ⚡ 14°  
SATURDAY ☁ 16°  
SUNDAY ☁ 19°

Leverage cookies or browser-local storage

# THANK YOU

Udi Dahan – The Software Simplist

Enterprise Development Expert & SOA Specialist

