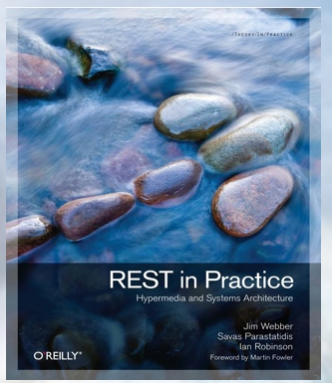
Fast-Track to RESTful Microservices

Dr. Jim Webber
@jimwebber



Logistics

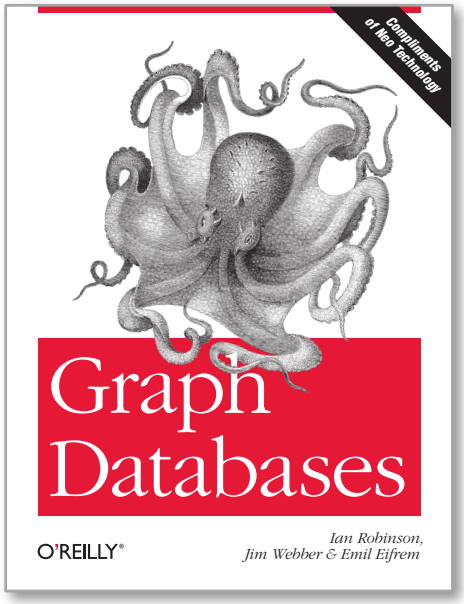
- Wireless
  - SSID: `CCTCBarbican`
  - Password: `CCTvw1f1` (case sensitive)

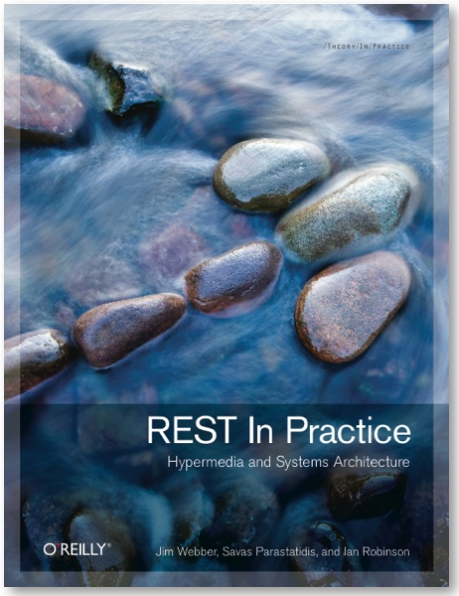
- 8:45 breakfast until 10:30am (4th floor restaurant)
- 11am coffee (coffee area)
- 12 Lunch (4th floor restaurant)
- 3pm coffee (coffee area)

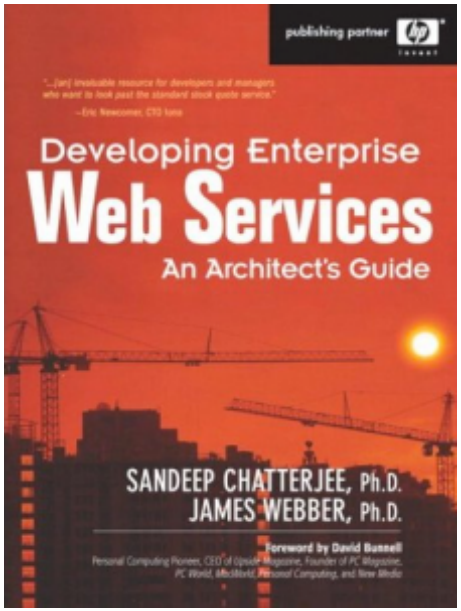
- Toilets:
  - 1st Floor: Gents
  - 2nd Floor: Ladies
  - Ground floor: unisex
- **Fire alarm test at 11am (do not evacuate)**

About Jim Webber

ThoughtWorks®







http://jimwebber.org @jimwebber

Day 1 – Technical Theme

- Web as platform
- Hypermedia APIs
- CRUD and Hypermedia Client Design Exercises

Day 2 – Architectural and Computer Science Themes

- Hypermedia Deep dive
- Scaling
- Event-driven systems
- Security considerations
- Distributed systems considerations

Day 3 – Ops and Governance Theme

- Migrating monoliths
- Operating systems of microservices
- Governance and planning
- Epilogue and Retrospective

Introduction

About

- This course covers two important styles in modern systems development
  - Microservices: A set of practices intended to provide a responsive delivery model
  - REST: An architectural style suited to the construction of dependable distributed systems
- These styles are mutually complementary and can be combined to provide agile delivery of robust large-scale distributed systems.

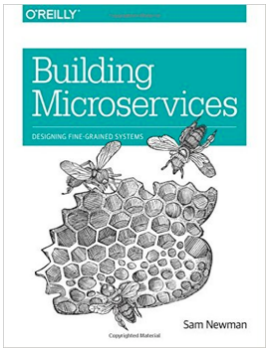
Overview

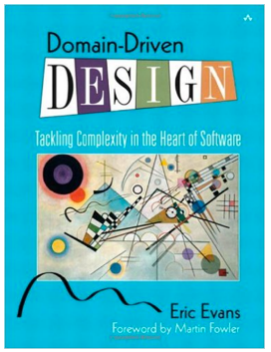
- Microservices:
  - Architectural style or delivery practices?
- REST
  - Architectural style or religion?

Over the next three days we're going to show how these two techniques are highly complementary and powerful

Microservices

- It's not about size, It's about context!
  - Sam Newman (Building Microservices) states that a service should be a bounded (business) context
  - Based on Eric Evans' DDD work
- So the size of the service isn't pre-defined?
  - Most contexts are relatively small and can be well defined:
    - Routing, pricing, ledger, storage, etc





Microservices are a delivery pattern, not an architectural style

Definitions of Microservices

Term describes at least 2 (not mutually incompatible) approaches:

- Service design
  - Implement ***business capabilities***
  - Scoped to ***bounded context***
  - Unix philosophy
- Convergence of modern delivery practices
  - Evolutionary architecture
  - Agile/Lean
  - Continuous Delivery
  - “Dev Ops”
  - Containerization

How big is “micro”?

Although “microservice” has become a popular name for this architectural style, its name does lead to an unfortunate focus on the size of service, and arguments about what constitutes “micro”.

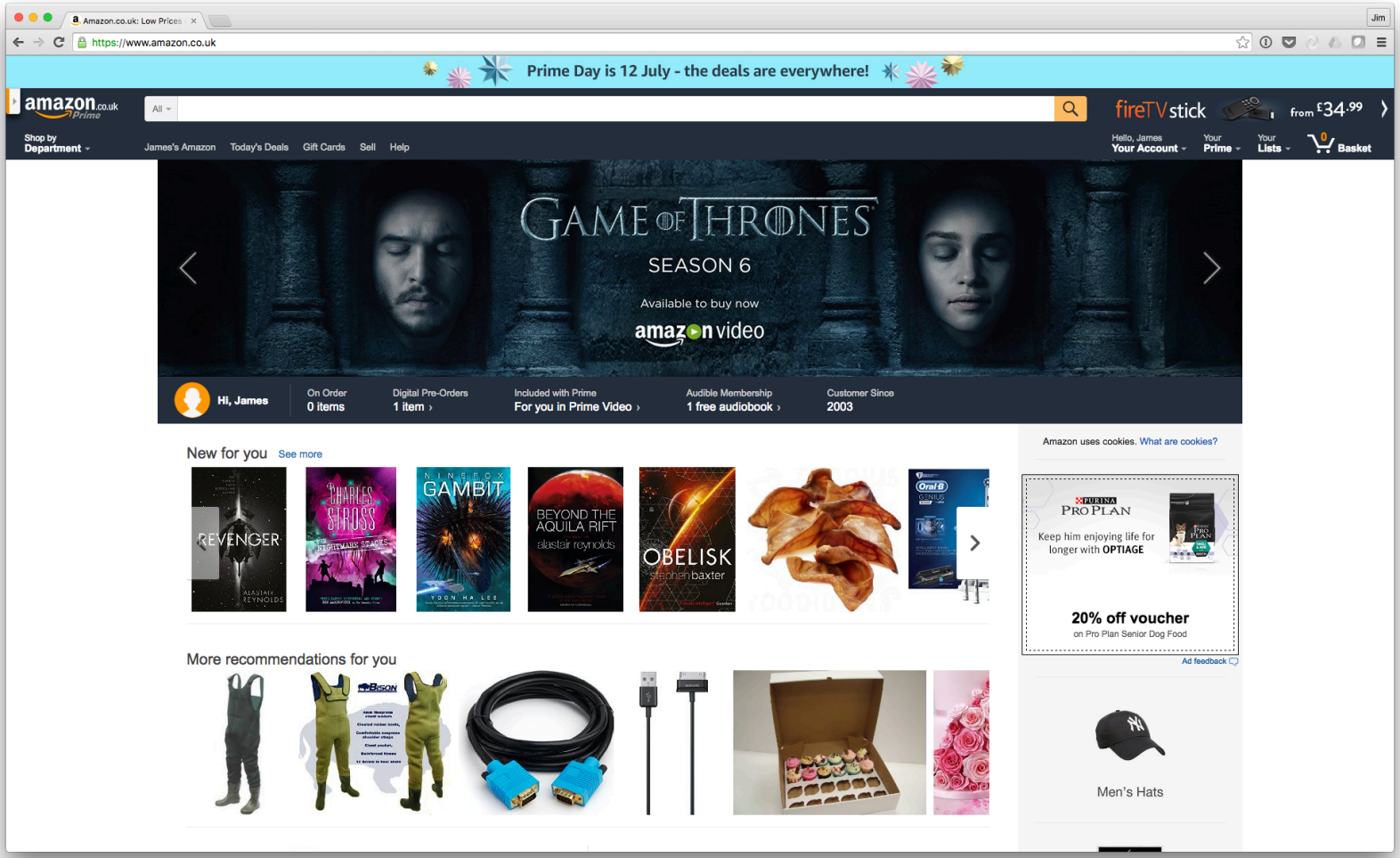
In our conversations with microservice practitioners, we see a range of sizes of services. The largest sizes reported follow Amazon's notion of the Two Pizza Team (i.e. the whole team can be fed by two pizzas), meaning no more than a dozen people.

On the smaller size scale we've seen setups where a team of half-a-dozen would support half-a-dozen services. This leads to the question of whether there are sufficiently large differences within this size range that the service-per-dozen-people and service-per-person sizes shouldn't be lumped under one microservices label.

At the moment we think it's better to group them together, but it's certainly possible that we'll change our mind as we explore this style further.

http://martinfowler.com/articles/microservices.html

~70 Services here!

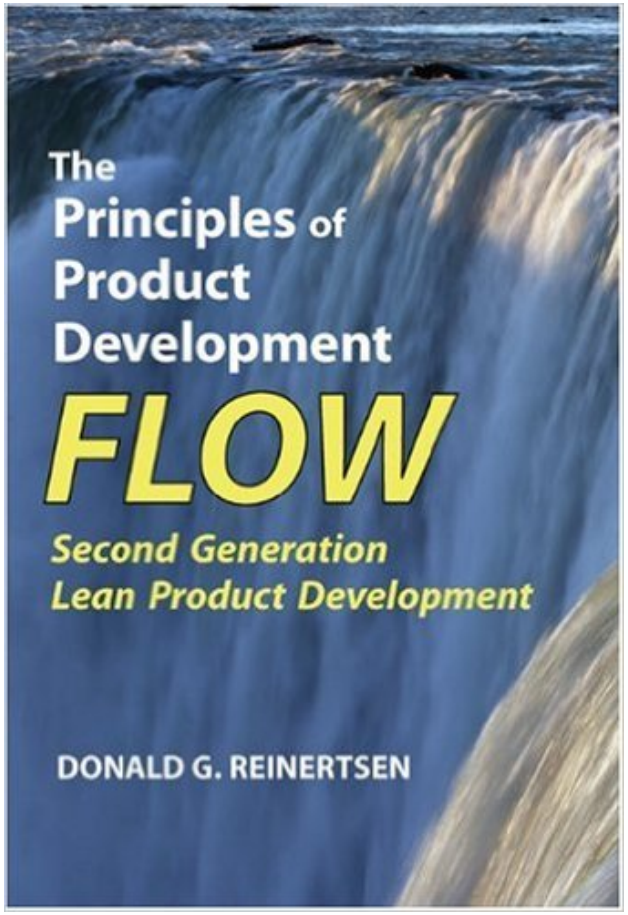


Microservices are not the panacea

Microservice Design Necessitates Delivery Practices

To succeed, the division of the software estate into many self-contained, self-managed services requires:

- Decentralized control
  - Local governance and decision-making
  - Economic model
- Fast feedback
  - Monitoring
  - Automated testing
- Lots of communication
  - Explicit context
  - Traceability, from need to delivery
- Repeatable processes
  - Automated deployment
  - “Cattle not pets”



Web as a Platform

- Distributed systems are hard to build and debug
- Much of what we do is the same-old, same-old
- Much of what we need is already described in the Web architecture
  - Distribution, coordination, state management, fault-tolerance…
  - And it's easy to work with

The Web provides a solid platform for building networks of microservices

Web as Platform: 3 Approaches

Behind the Web

Web as Platform: 3 Approaches

On the Web

Web as Platform: 3 Approaches

Of the Web

Of the Web

- Services are good "Web Citizens"
  - Fit within the Web's architecture
    - Used in combination with other parts of the architecture
  - Use the Web's technologies the way they were intended
    - **HTTP** Document transfer protocol
    - **URI** Identity and addressing
  - Reuse well-understood representation formats
  - Network effect
    - Networked components
    - Interlinked with other components

Be of the Web

If we want to build platform-like distributed systems with clean APIs, we need to:

- Understand what each service should do
  - Capabilities
  - Bounded contexts
- Build Web APIs that are sympathetic to the Web as platform
  - Web architecture
  - Web technologies – HTTP, URIs, hypermedia

Service Modeling

Identifying Services

- What should our service do?
  - What should it ***not*** do?
- Identify an organizing principle
  - Technology? Organizational unit? Business process?
  - But technologies change, organizations change, and processes change

Building Systems to Support a Changing Business

- Every business can be described in terms of a set of relatively stable abstractions
  - We call these abstractions ***business capabilities***
  - We should aim to build software aligned with these long-lived abstractions

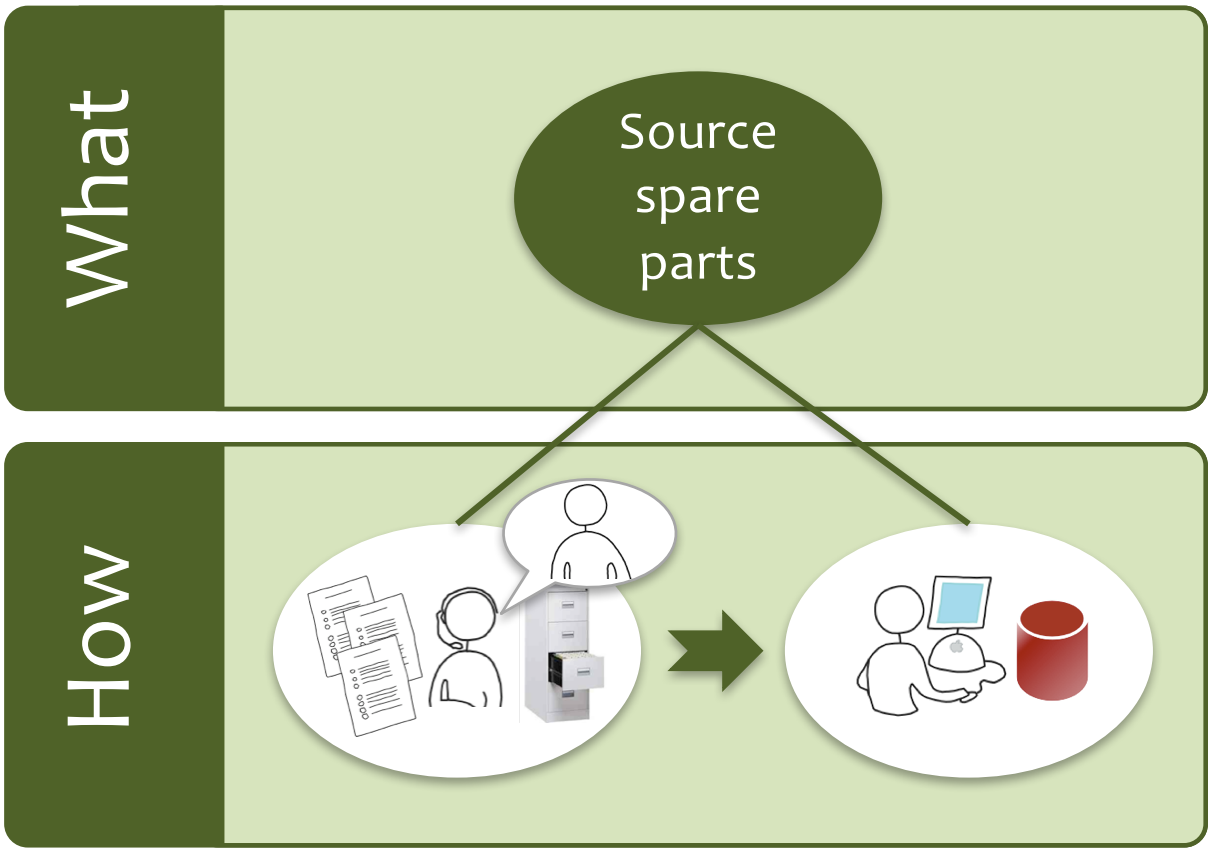
Business Capabilities: A Definition

- "A company's ability to deliver a core business outcome"
- Business function encapsulating some mix of ***people*** + ***processes*** + ***technology***
  - Inputs and outputs – capital, materials, expertise, data
  - Quality of service

Characteristics of a Business Capability

- Describe ***what*** a business does, but not ***how***
  - ***What*** a business does changes far less frequently than ***how*** it does things
  - Capability models are more stable than models based on a company's products, services, tools, technologies, strategy or organizational structure

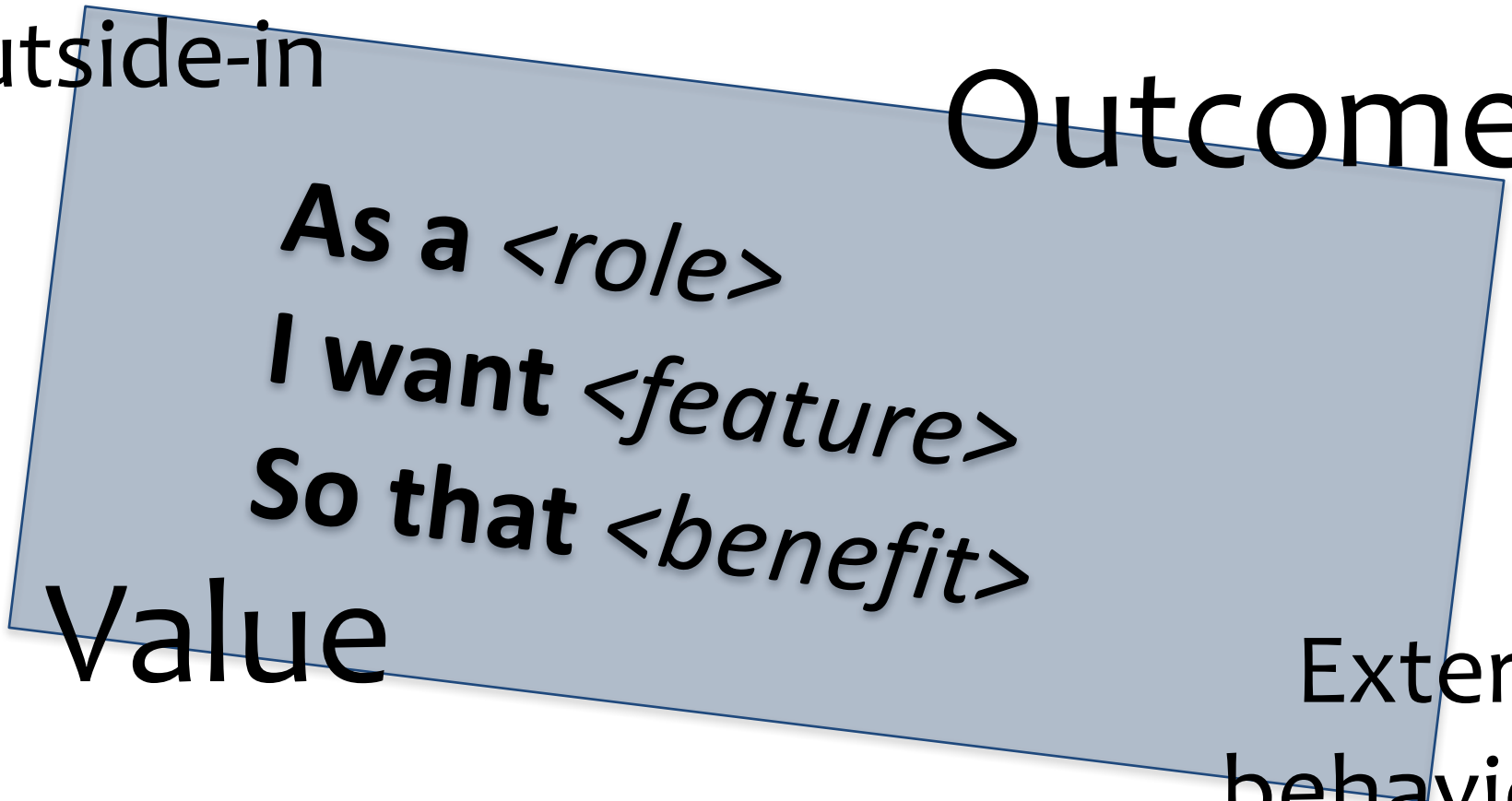
People Change, Processes Change, Technologies Change…



Hacking a Business Capability Model

- "To do X we need to be able to do Y"
  - Iterate, to build a capability map
- "How (well) do we implement this capability today?"
  - "Is it core to our business?"
  - "Does it differentiate us from our competitors?"
  - "Who's responsible for its execution when systems fail?"
- "What events in the wider world are we interested in?"
  - "When should others know that something has changed in our world?"

Identify Capabilities with User Stories

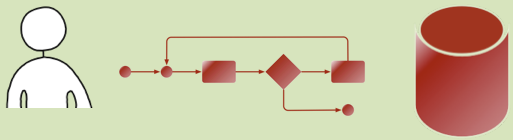


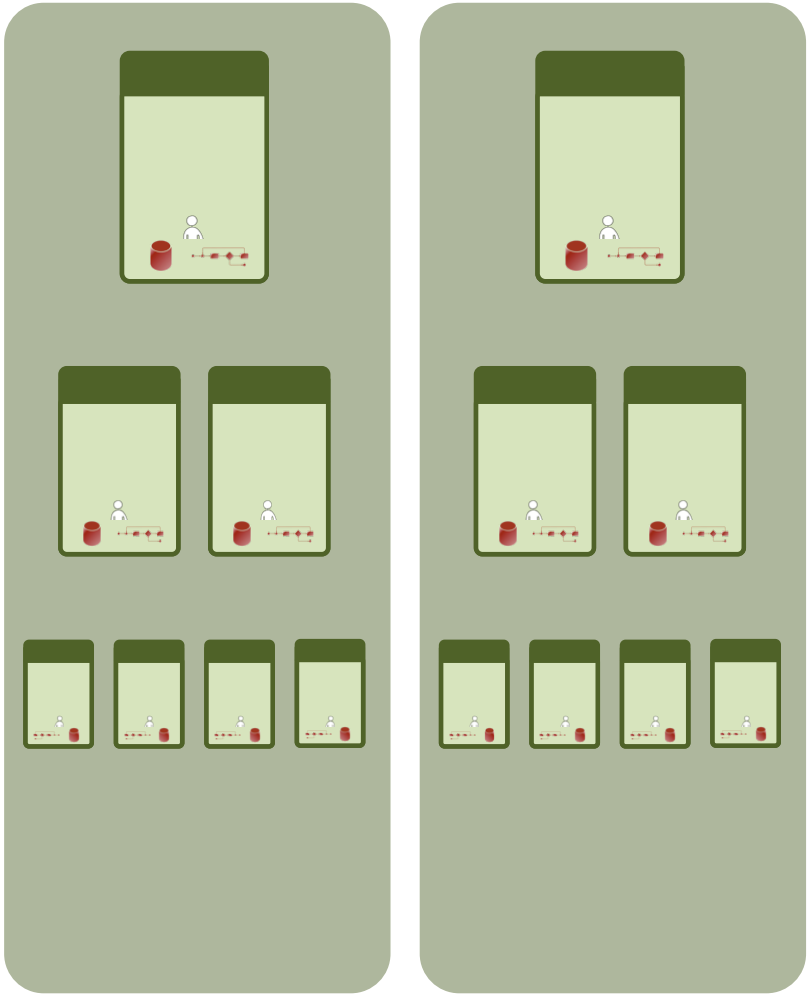
Capability Hierarchies

Outcome
[verb + noun]

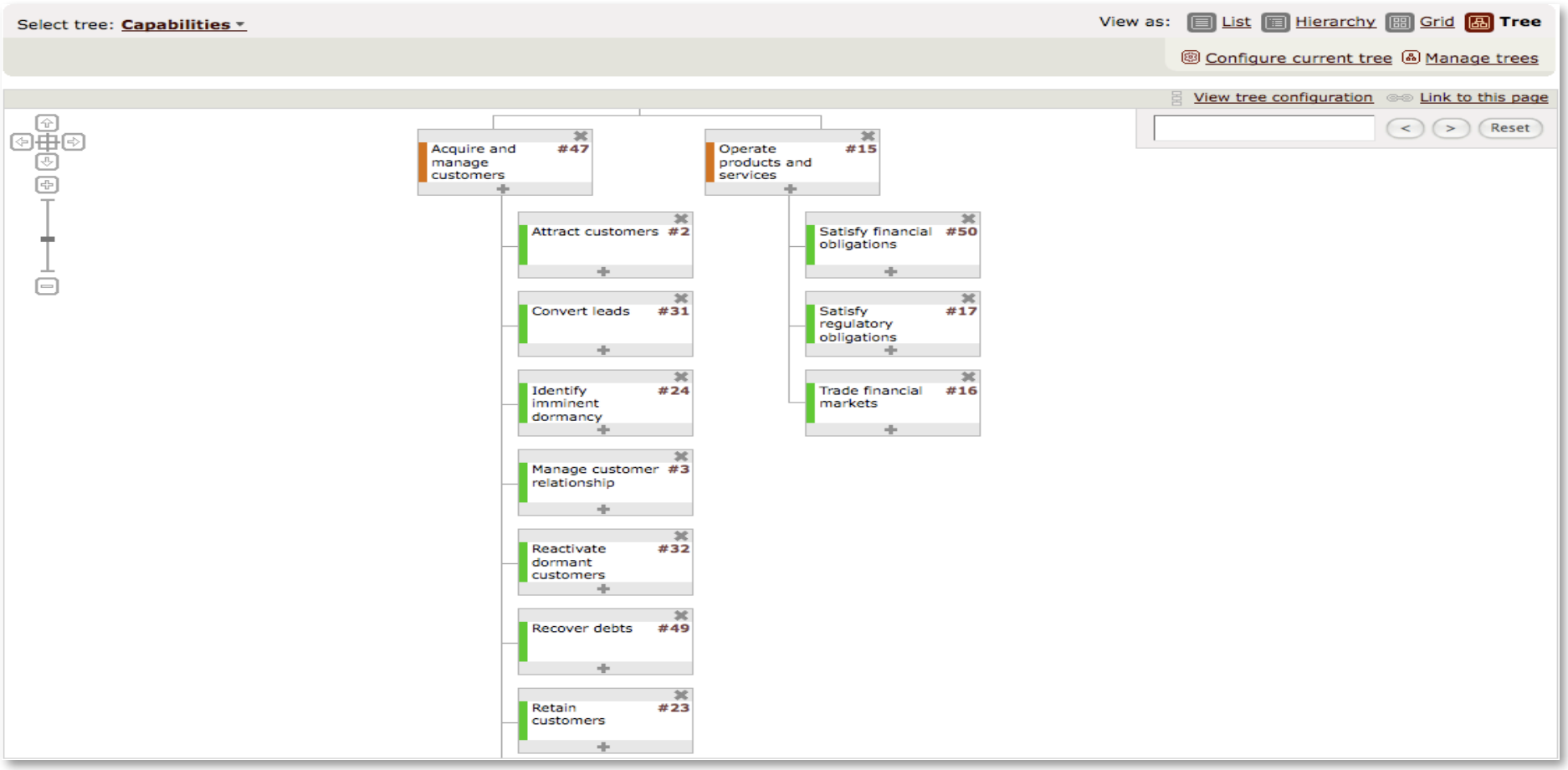
Autonomous/connected
Inputs/outputs
Events
Owner/consumer
Core/supporting
Differentiating/commoditized

Performance
Governance
(Regulatory) constraints

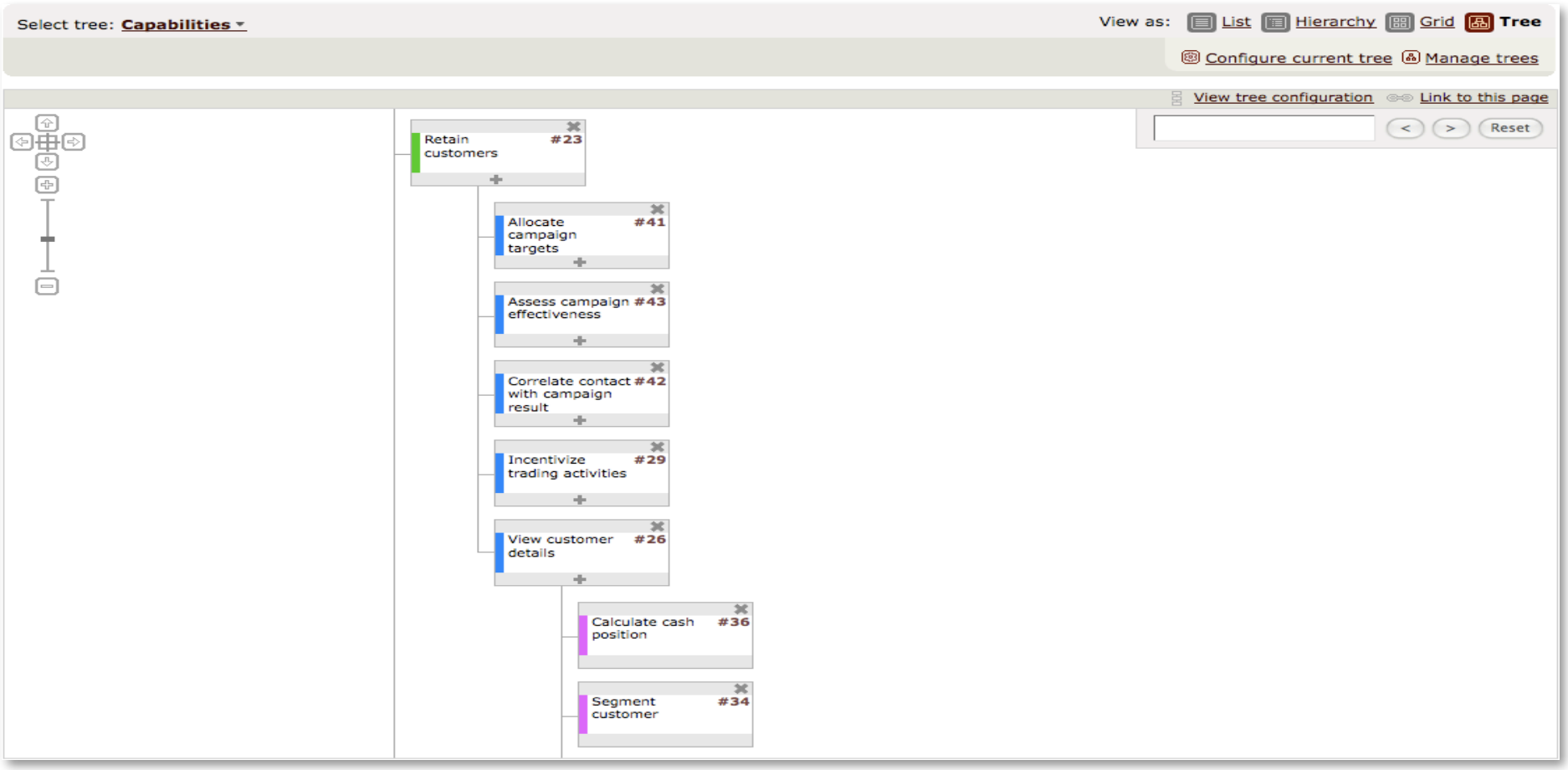




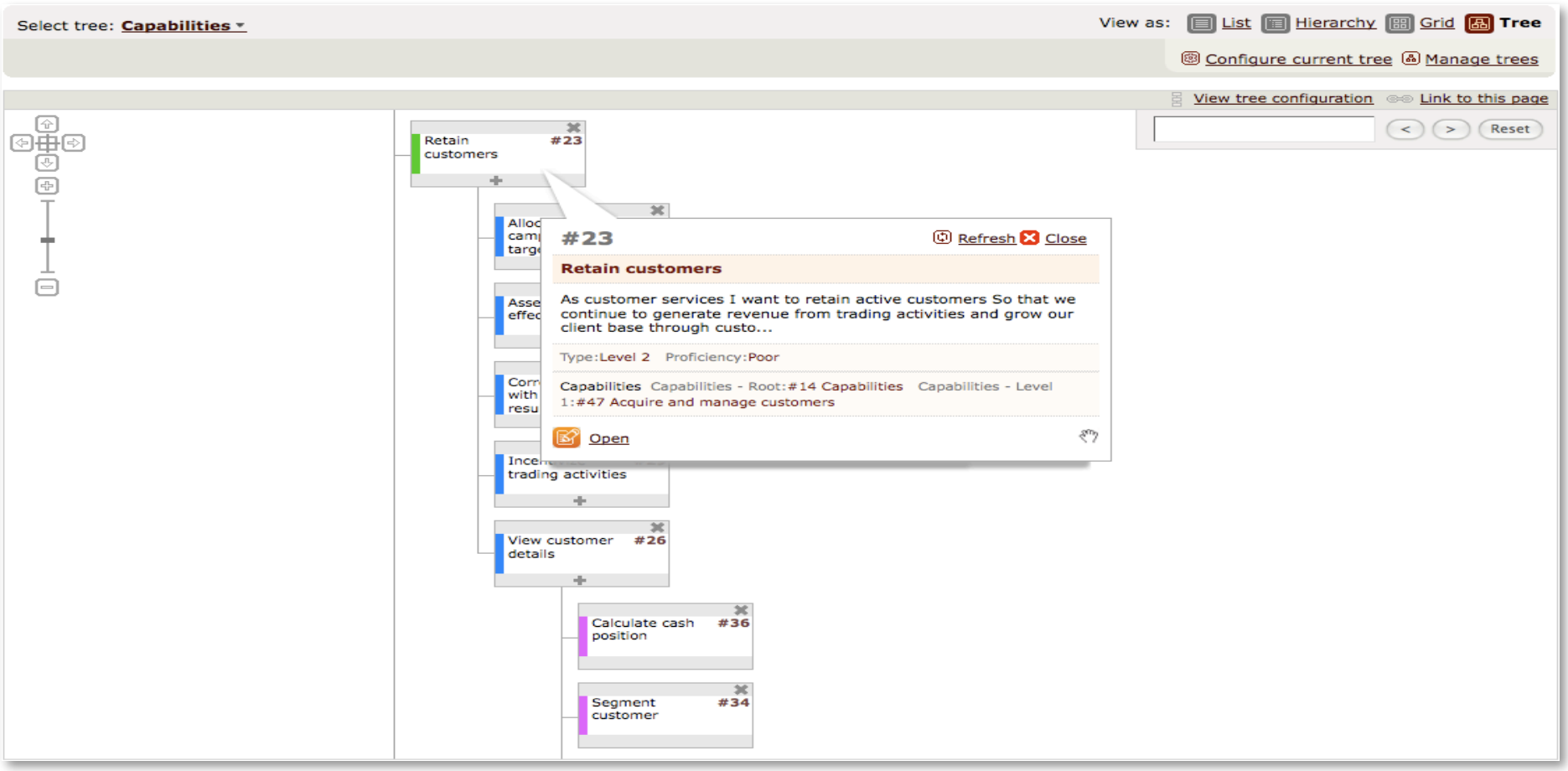
Capability Map



Capability Map

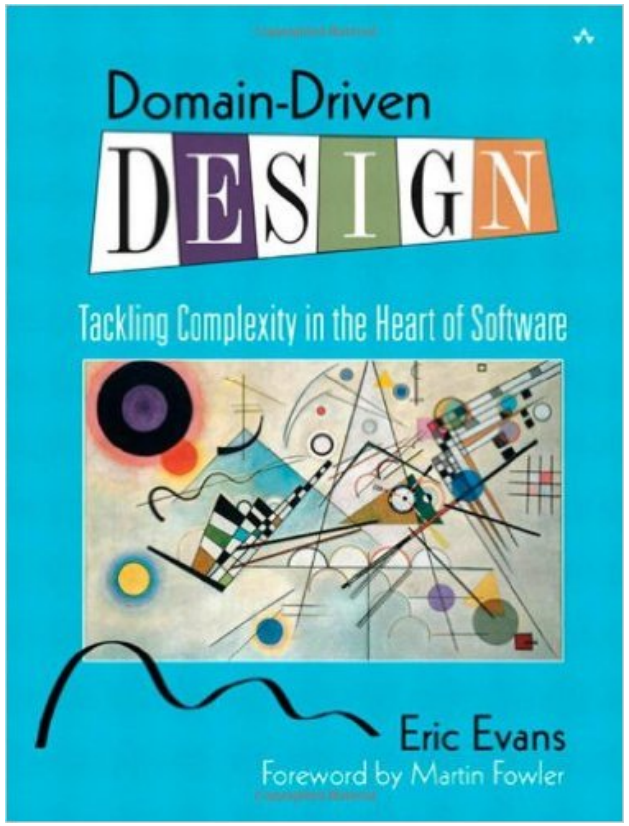


Capability Map

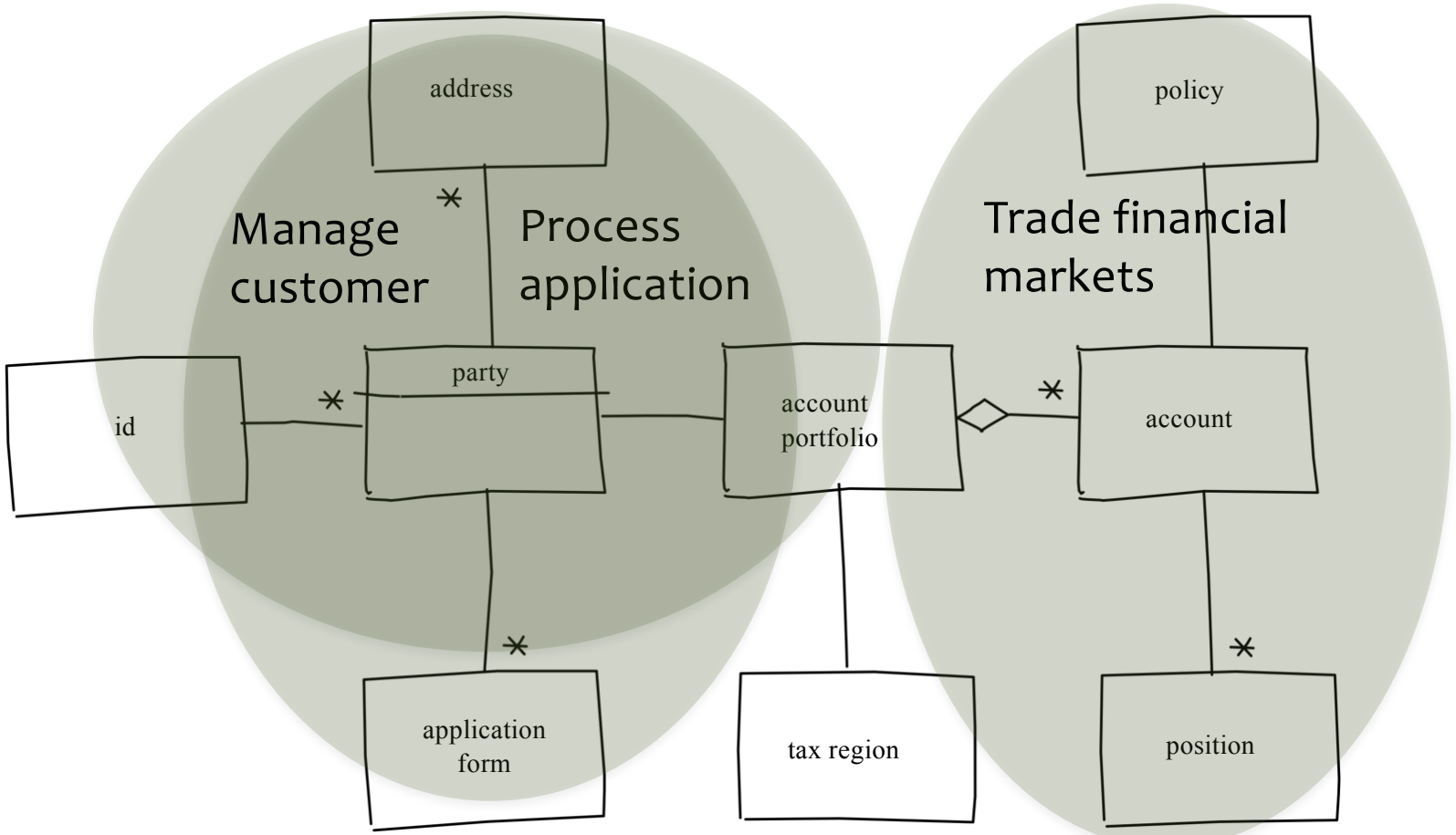


Bounded Context

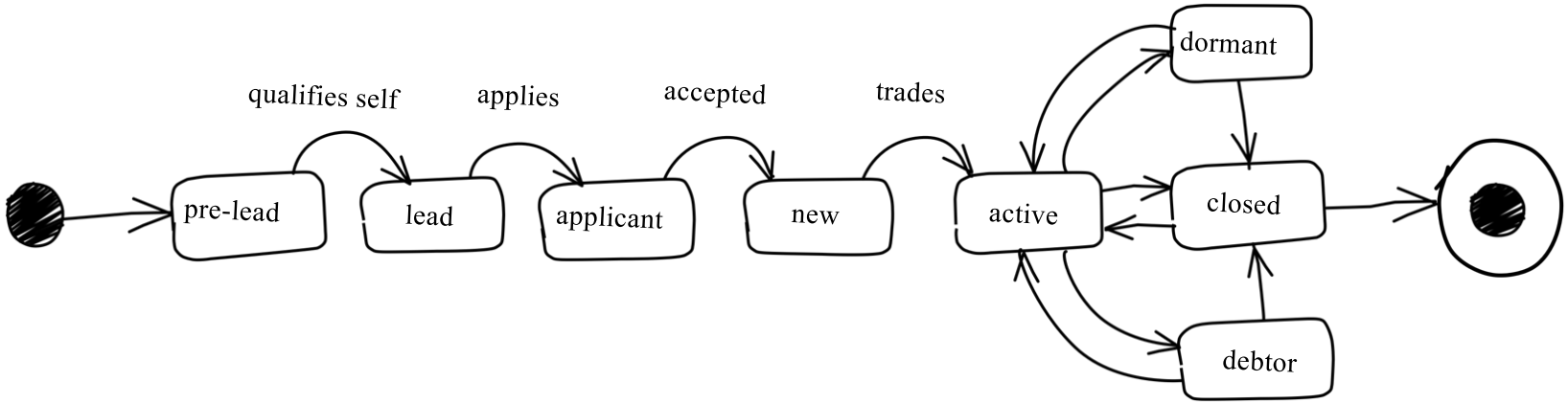
- Internally consistent domain model
  - Explicit boundary
- Organised around a business meaningful solution space
  - Business capability
- Entities may appear in several bounded contexts
  - Different representations and functions



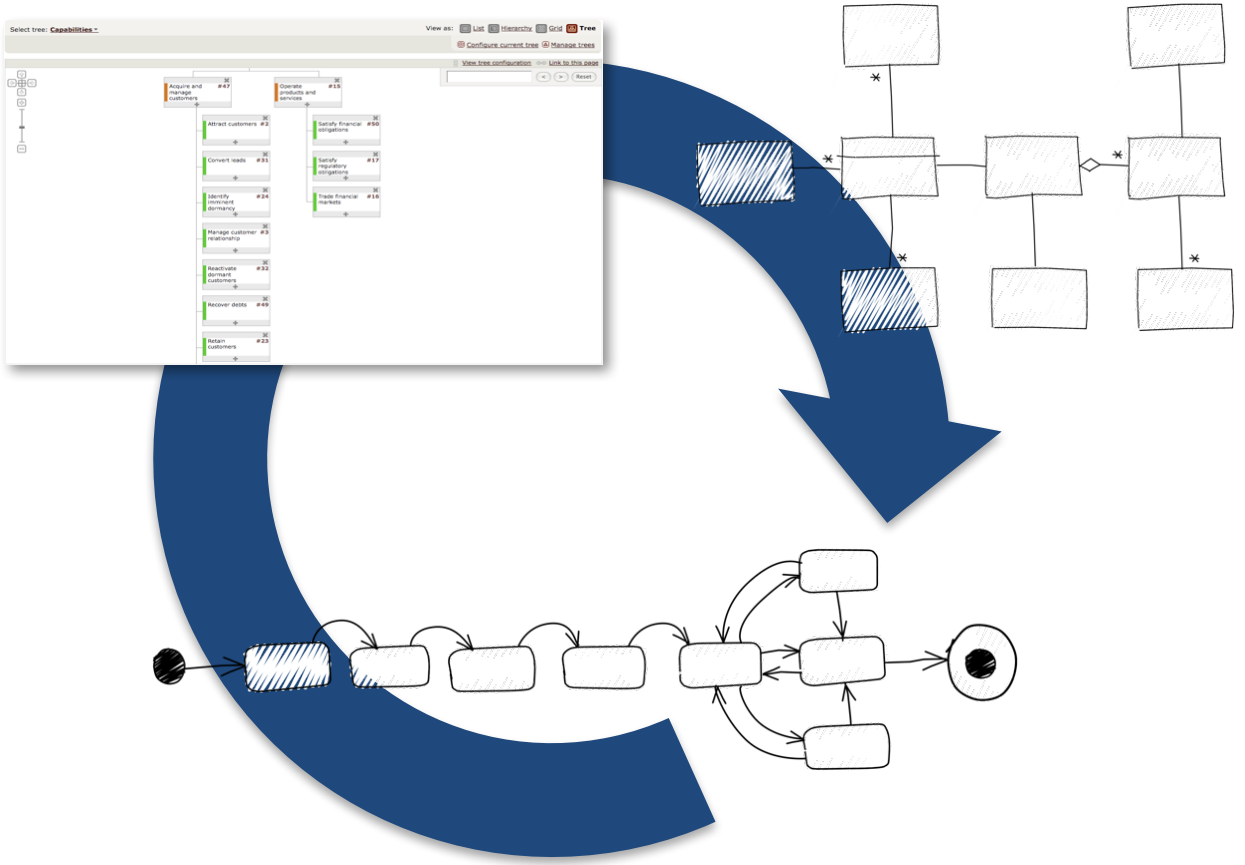
Bounded Contexts, Domain Models and Capabilities

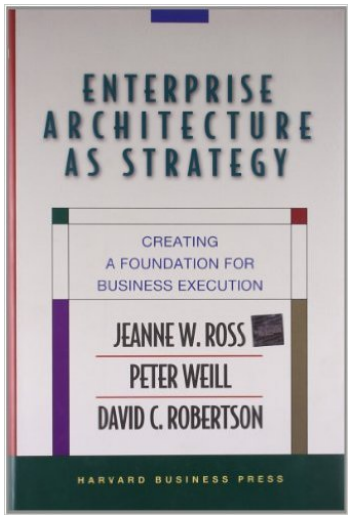


Entity Lifecycles



Capabilities Drive the Entity Lifecycle





Anticorruption Layer

- Means of linking two bounded contexts
  - Explicit interactions between different bounded contexts
  - Changes to the internal implementation of a service don't leak across boundaries
  - "Neutral zone"
- Pat Helland, "Data on the Outside versus Data in the Inside"
  - *http://www.cidrdb.org/cidr2005/papers/P12.pdf*
  - Coordinated exchanges of documents between services versus a service's manipulation of its internal state

Putting It All Together

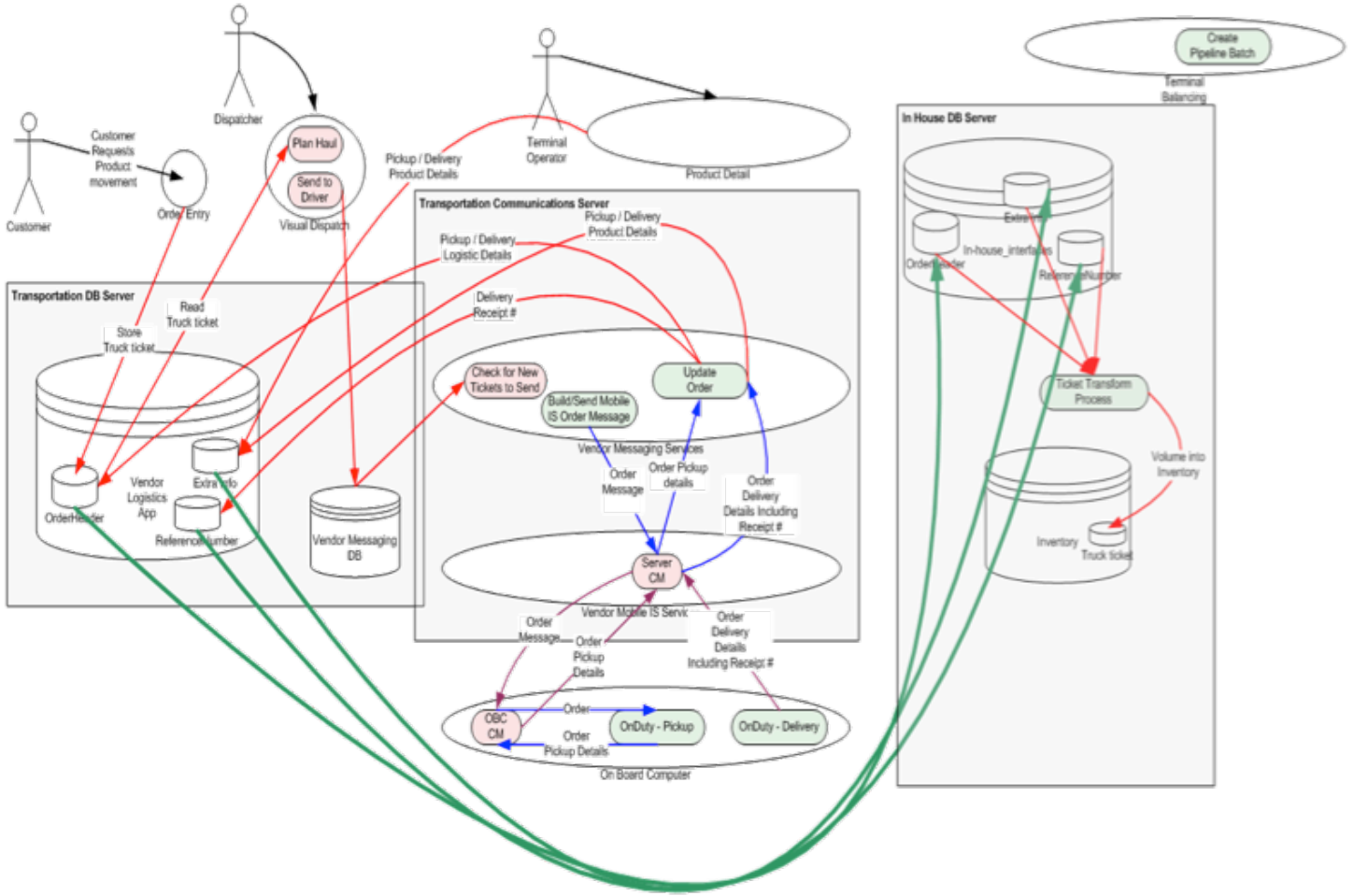
- Capabilities provide stable description of business functions
  - Services implement one or more capabilities
- Bounded contexts establish domain model boundaries
  - Context boundaries determine service boundaries
- Capabilities operate on domain entities
  - Transition entities through their lifecycle
  - Release business value
- Web APIs act as anticorruption layers
  - Adapt internal domain models for external consumption

Summary

Capability	Implemented by services
Bounded Context	Service boundary Internal domain model
Anticorruption Layer	Resources, which adapt domain model for external consumers

Case Study

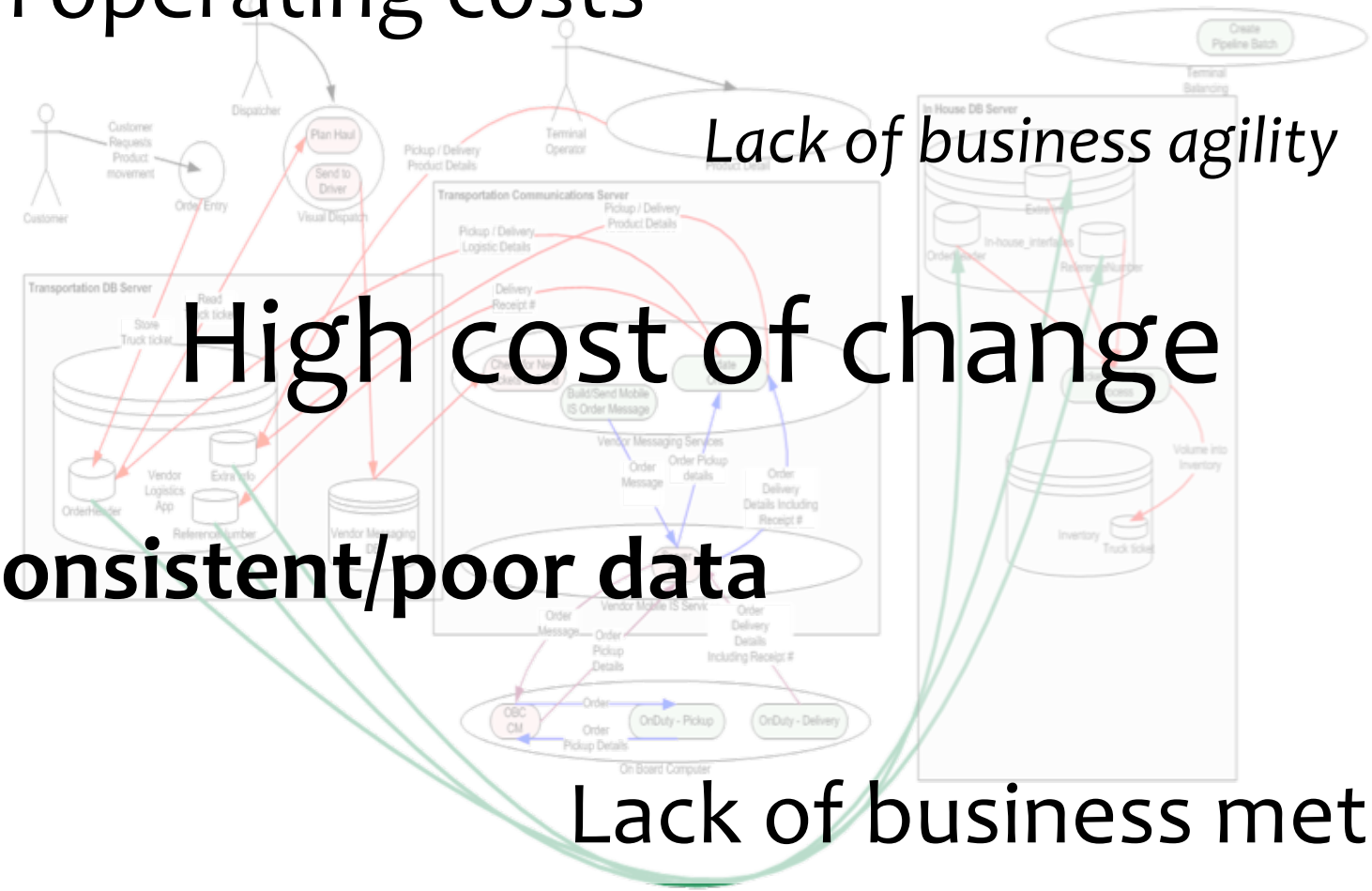
Midstream Oil and Gas



Problems With Existing Integration Architecture

High operating costs

Lack of business agility

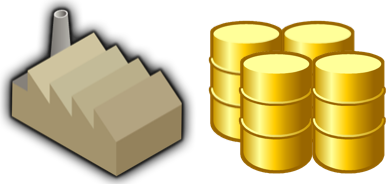


High cost of change

Inconsistent/poor data

Lack of business metrics

Entities, Actors and Actions

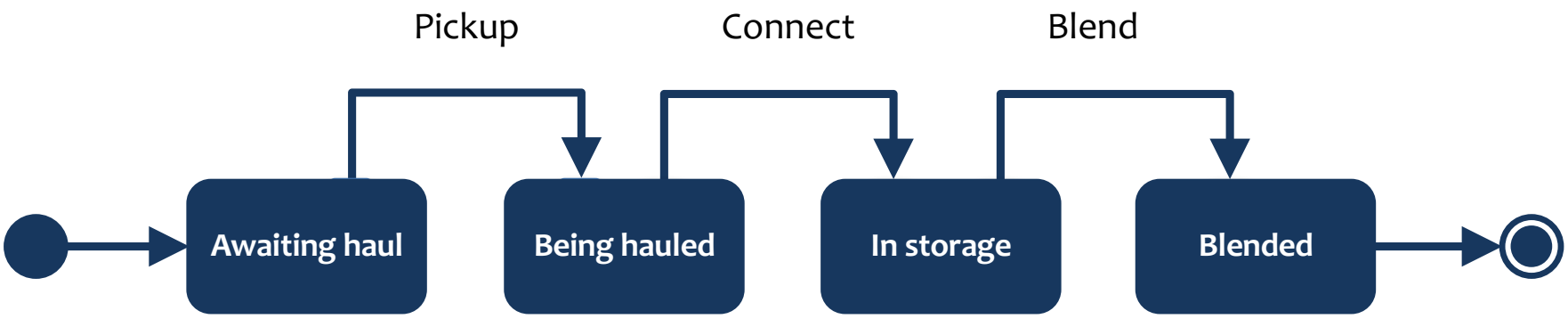


Production, Diluent, Truck, Terminal

Dispatcher, Driver, Operator

Dispatching, Hauling, Analyzing, Blending

Production States and Events

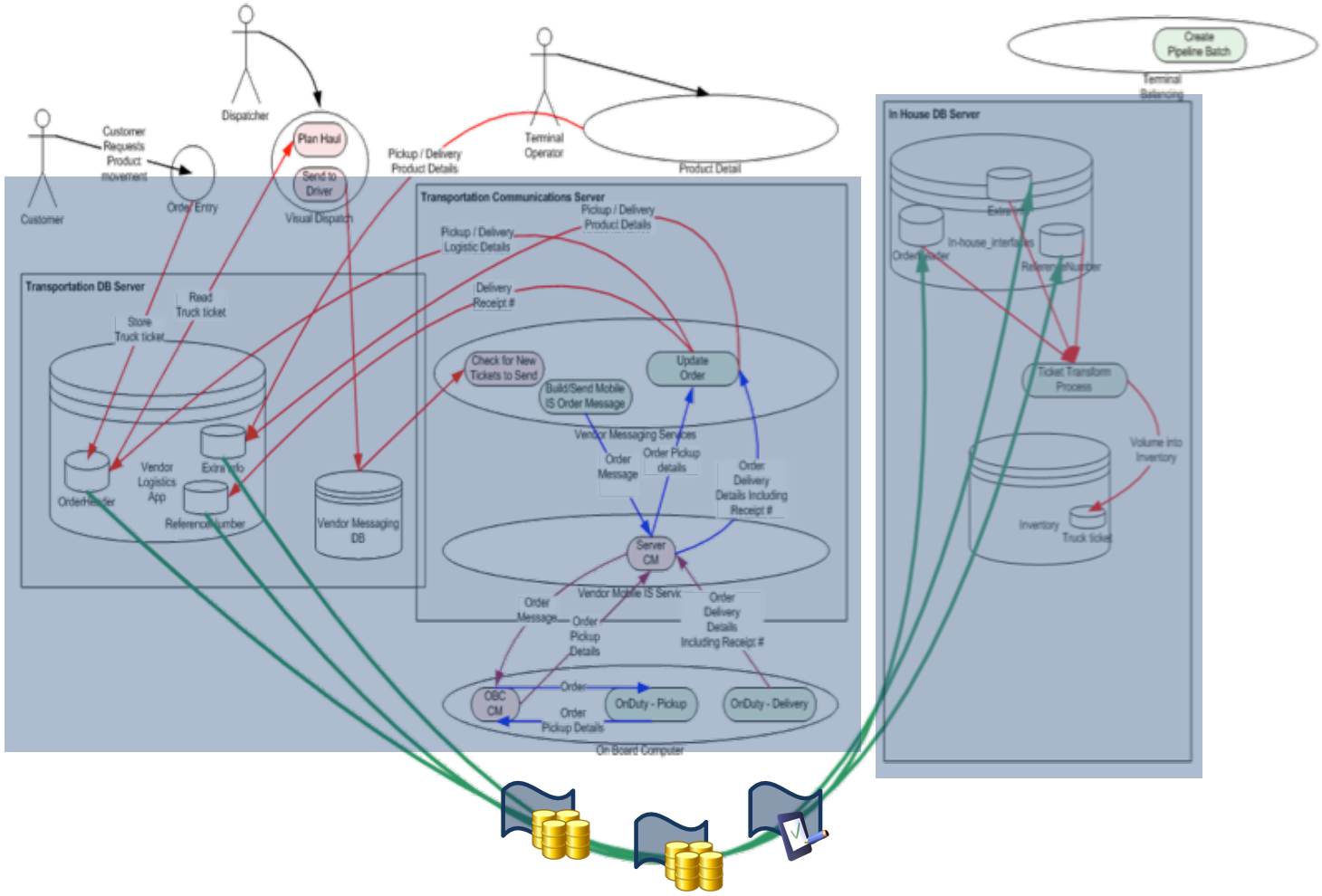


Crude connected to terminal

Diluent connected to terminal

Lab analysis available

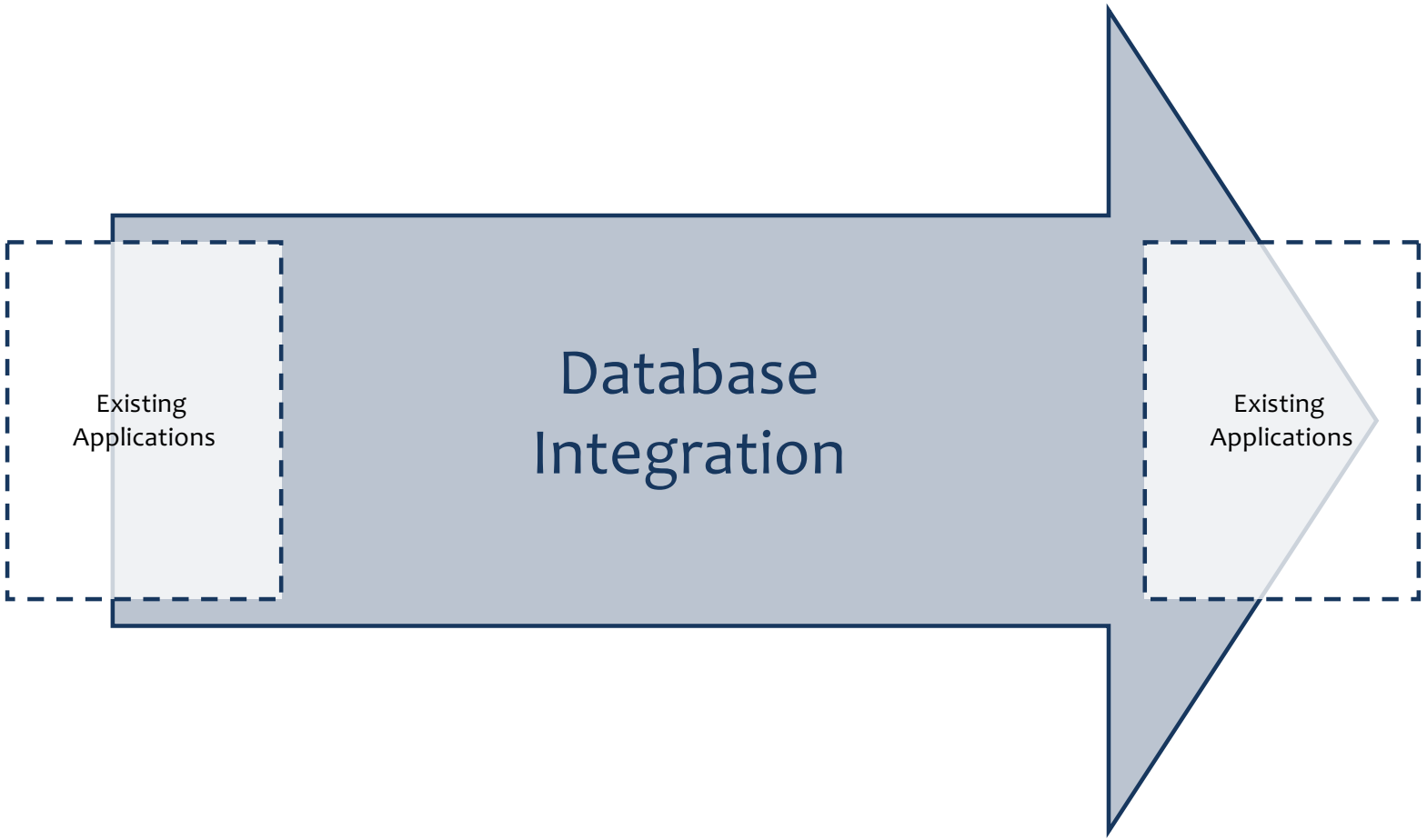
Mapping Events onto Existing Integration Architecture



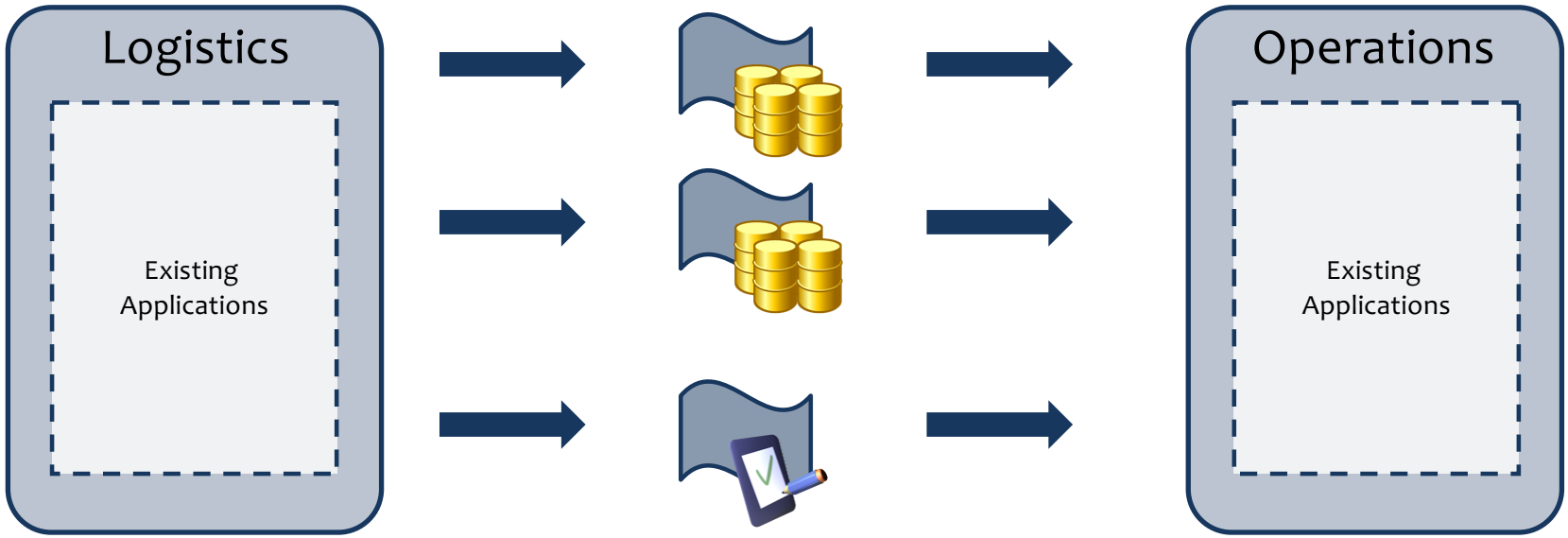
Quality of Service

When	Haul of crude production to terminal completed
Consumers	Terminal Balancing
Information	Order number Terminal Tank Volume Receipt number
Timeliness	< 5 minutes
Frequency	Approx 15,000 per month
Peak periods	7am to 8pm
Reliability	Guarantee delivery

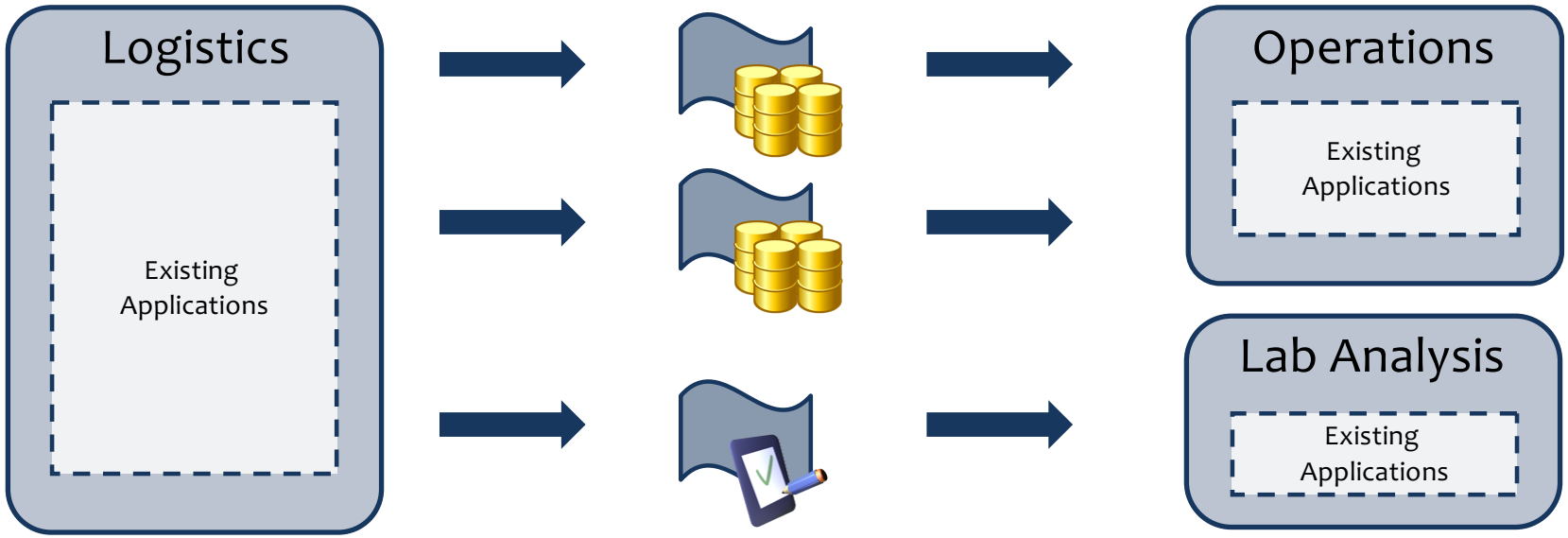
Existing Replication Strategy



Event-Based Integration



Later – Breaking Out More Services



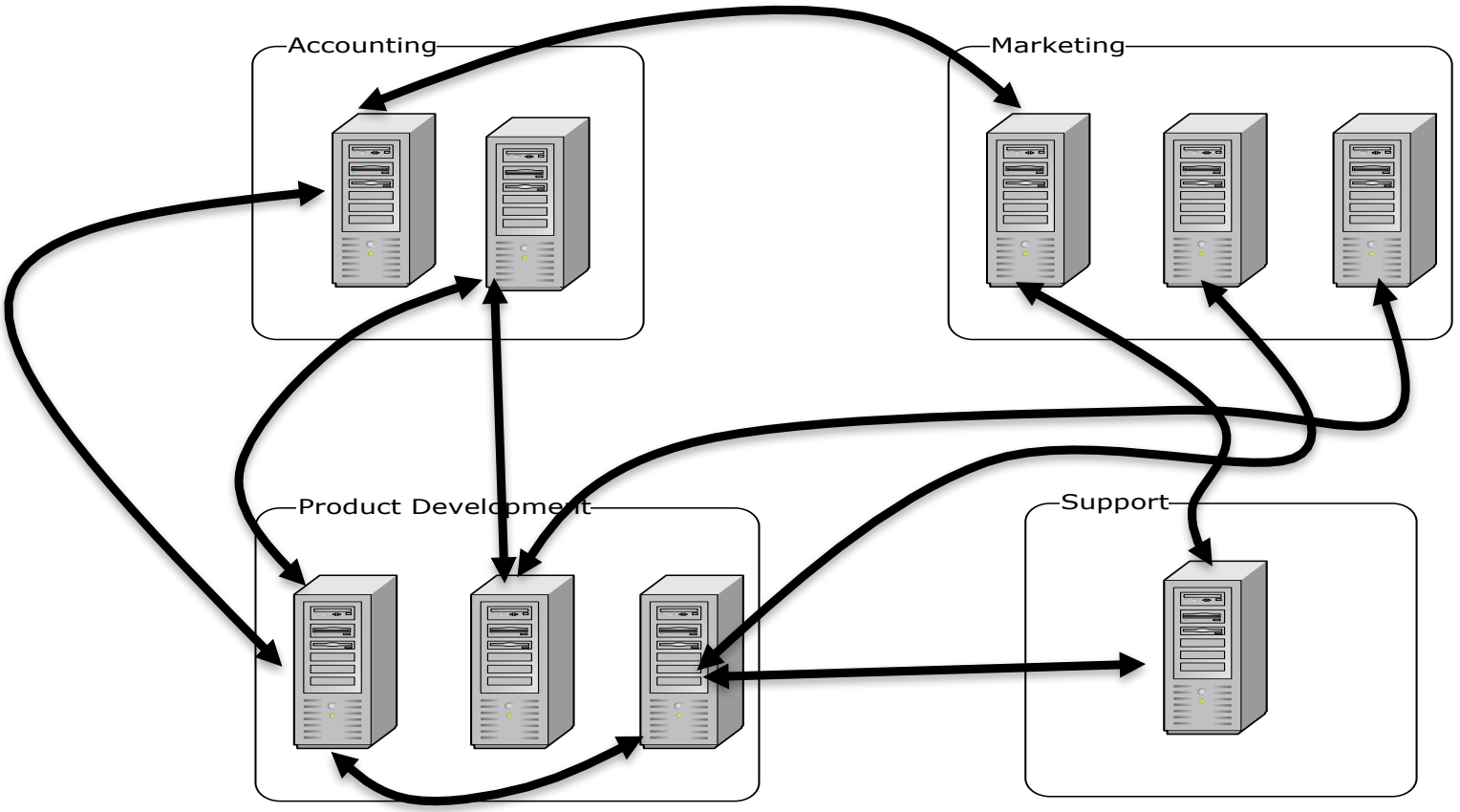
Motivating Example: Designing Systems of Services

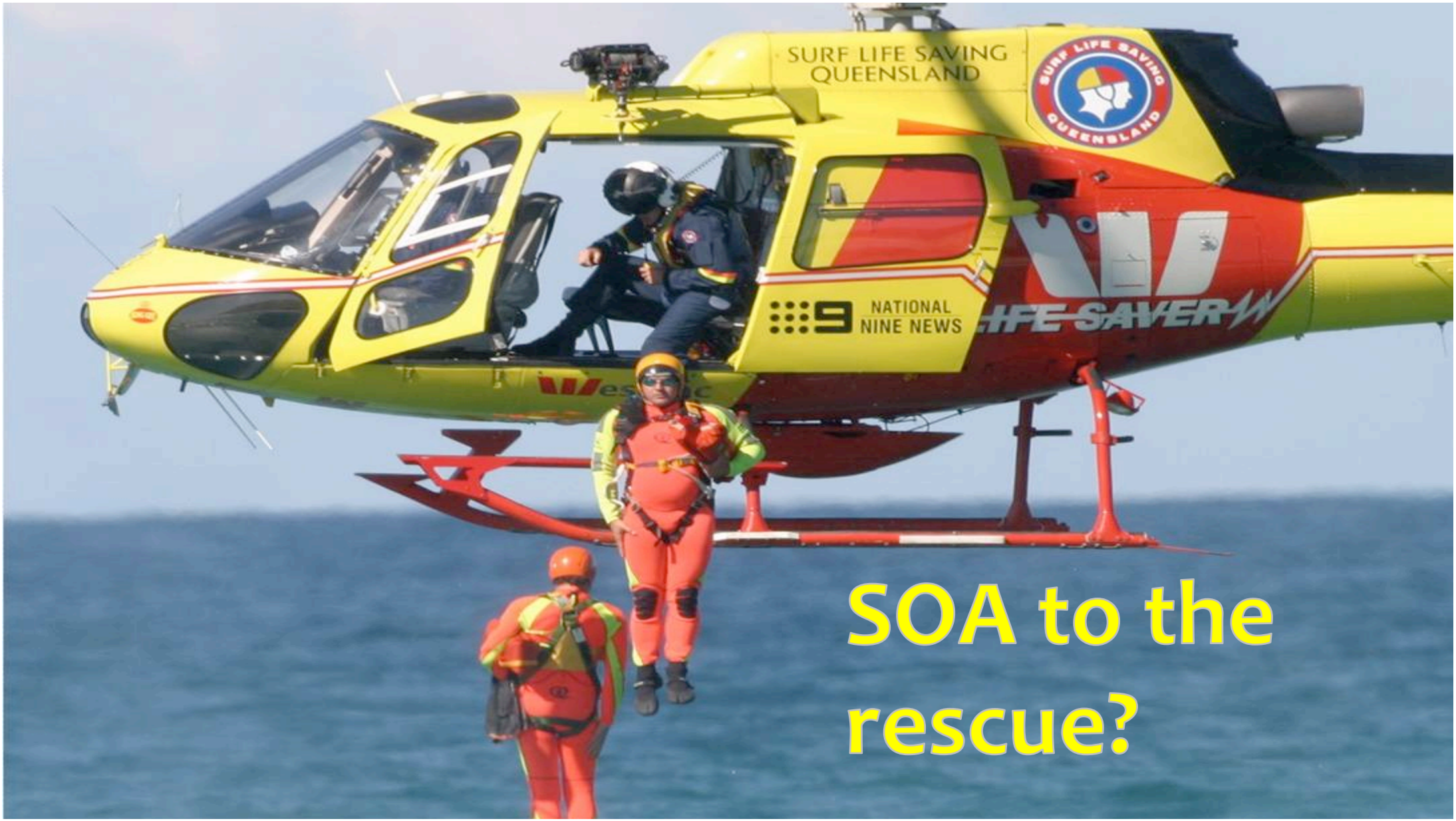
Incremental Service Delivery

- It is unfortunately common for architects to design impractical cathedrals for integrated systems
  - Latent, impractical, but look great on PowerPoint
- Conversely, we think that services emerge naturally from business domains
  - We believe that bounded contexts are great candidates for services
- In this session we'll discuss how to incrementally and iteratively integrate systems of services
  - This is a technique from the SOA world known as *Guerrilla SOA.*


I WANT TO
BELIEVE

Today's Architecture: Duct-taped Monoliths

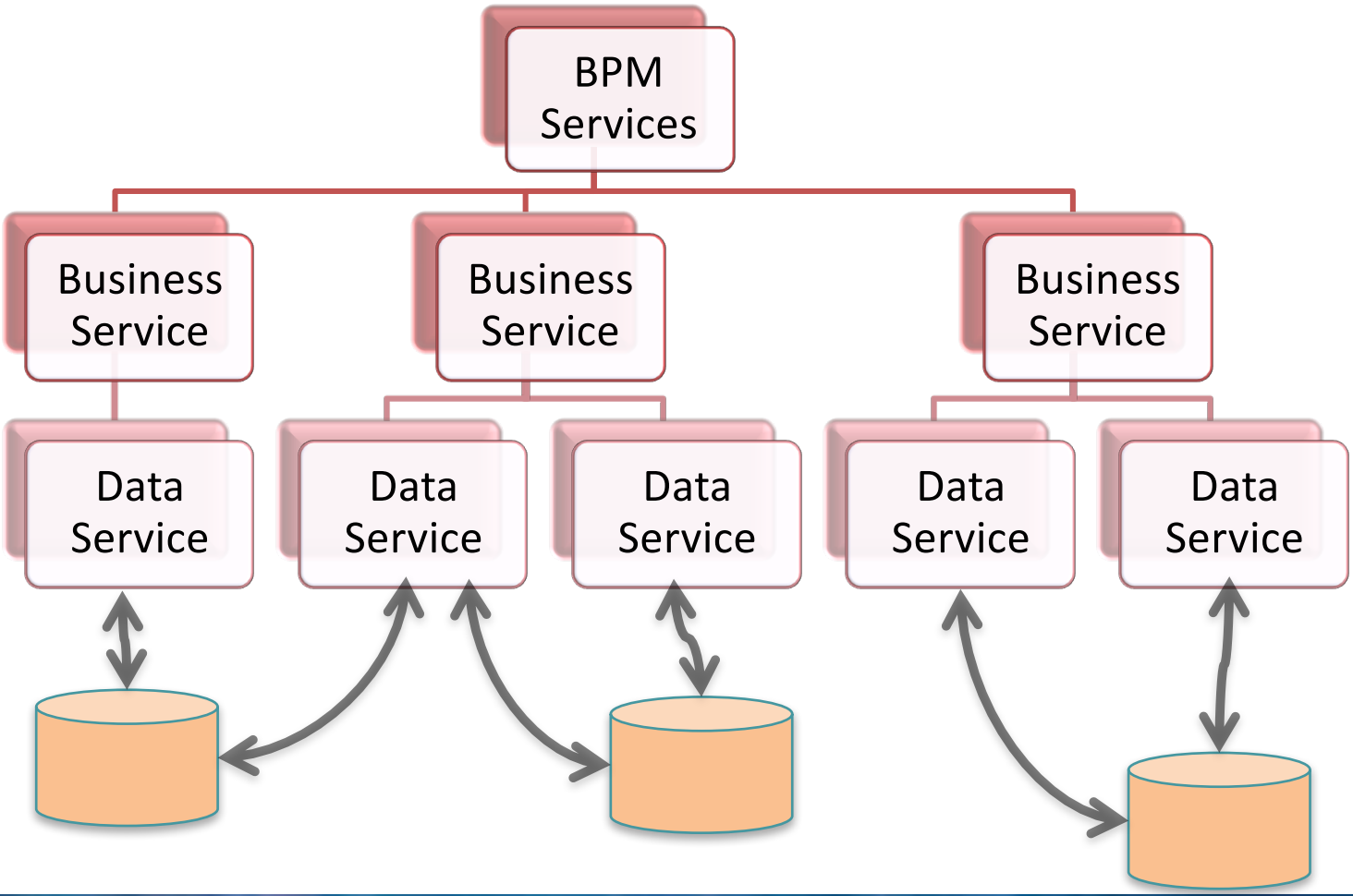



SURF LIFE SAVING
QUEENSLAND
SURF LIFE SAVING QUEENSLAND
9 NATIONAL NINE NEWS
SOA to the rescue?

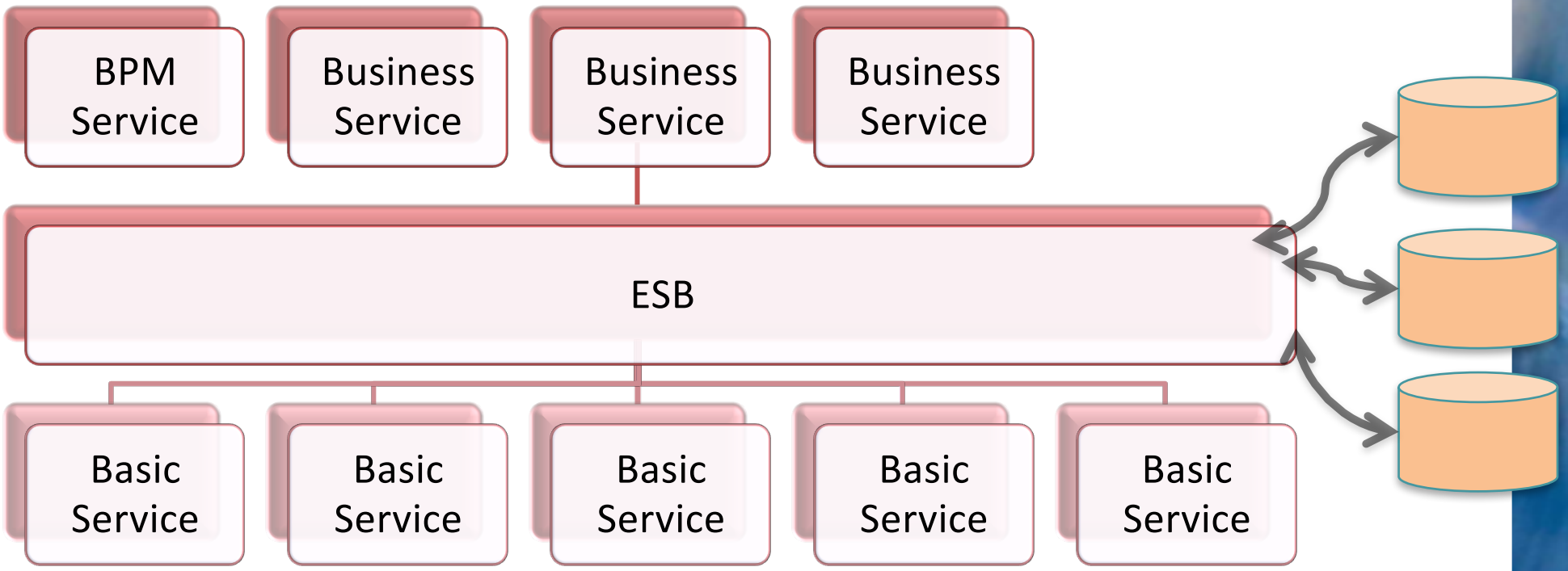
"SOA: Principles of Service Design *underwent a thorough technical review involving over* ***60 reviewers from different vendors****, organizations, and professions across North America, Europe, and Asia. The book has been* ***formally endorsed*** *by members of* ***major SOA vendors****, including IBM, Microsoft, Oracle, BEA, and Intel."*

-- Thomas Erl

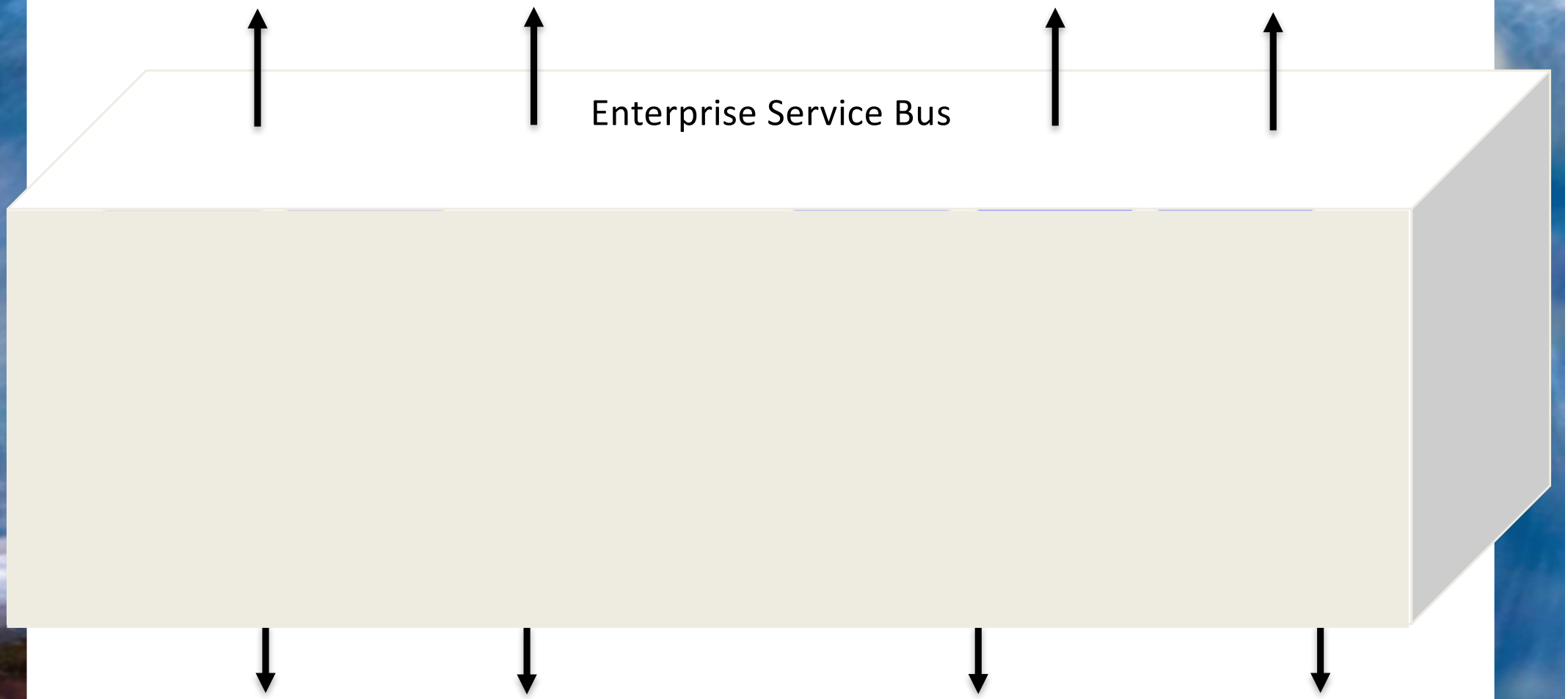
Tomorrow's Enterprise Architecture



Nirvana Enterprise Architecture, or...



ESB - Erroneous Spaghetti Box?




Architectural Fantasy
THE LORD OF THE RINGS
THE RETURN OF THE KING
WWW.LORDOFTHERINGS.NET


Ungovernable


Doesn’t Scale

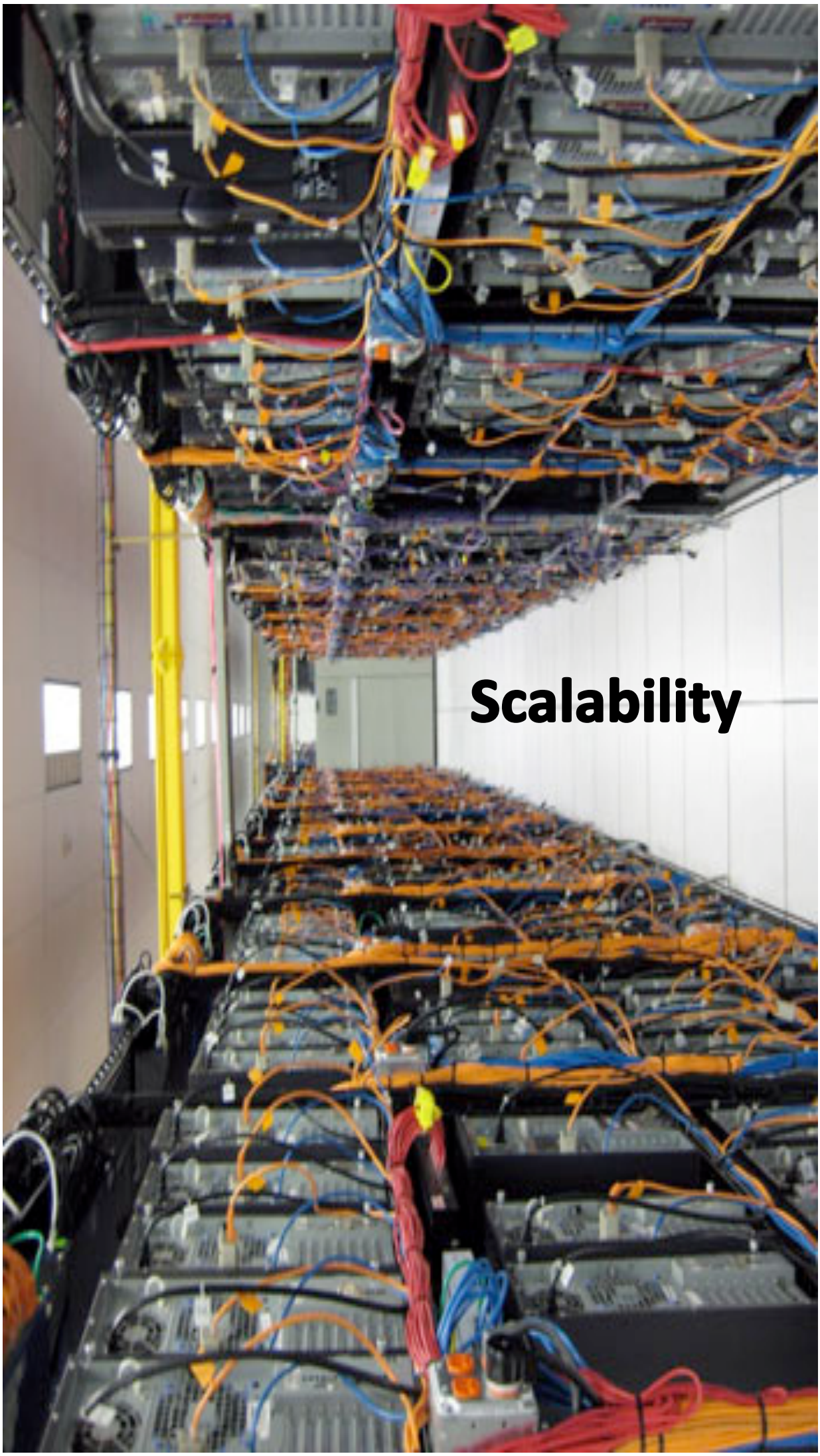

But we still do it.


Why?
Because it’s “less risky”


Why?
Because that’s what the market does


Why?
Because we need the -ilities

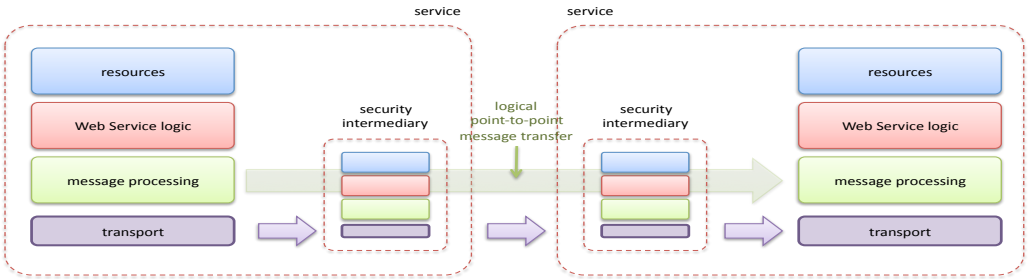
So let’s talk about those -ilities


Scalability

Enterprise Scalability?

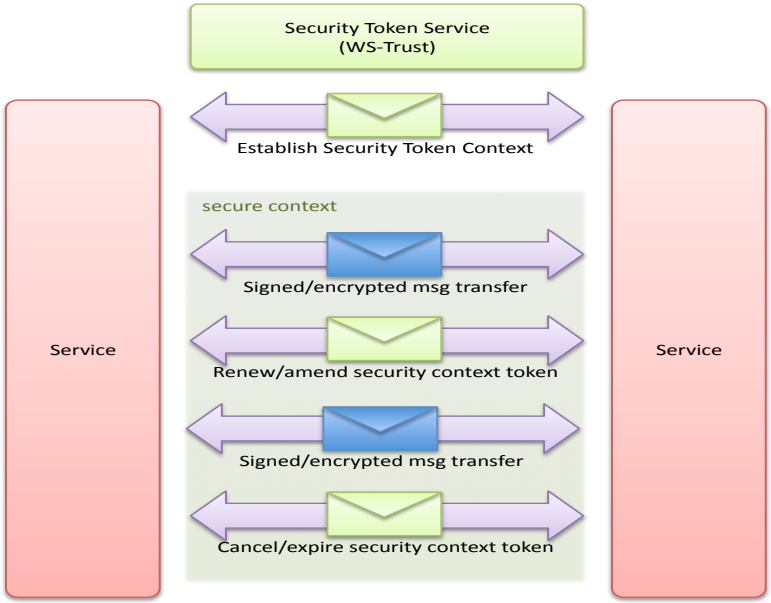
Web Scale!

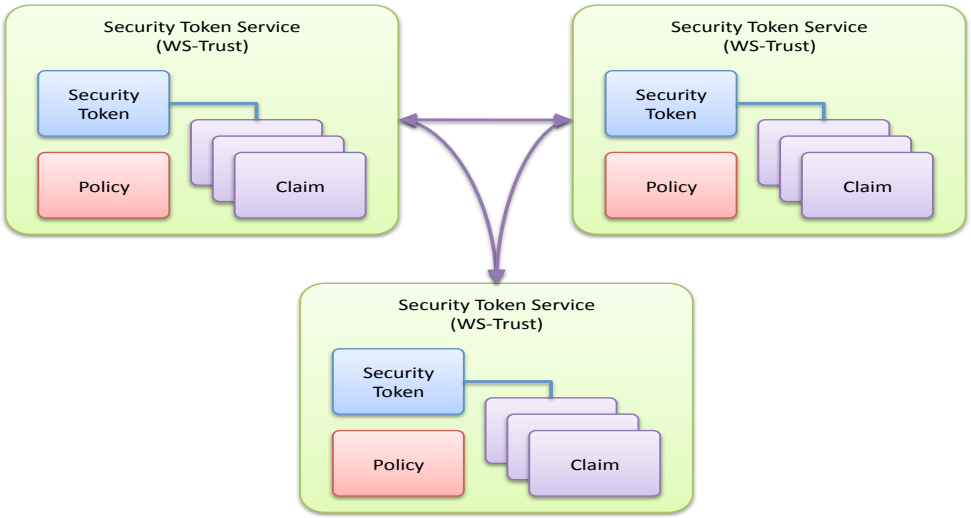
Traditional Enterprise Security



End to End Secure Messaging

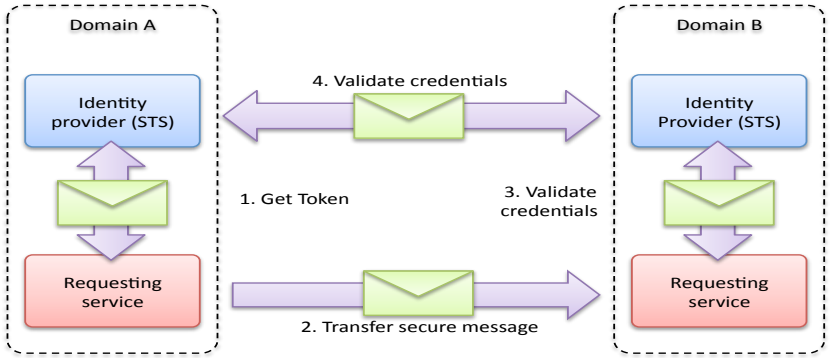
End to End Secure Conversations





Security Tokens and claims

Federating access with tokens

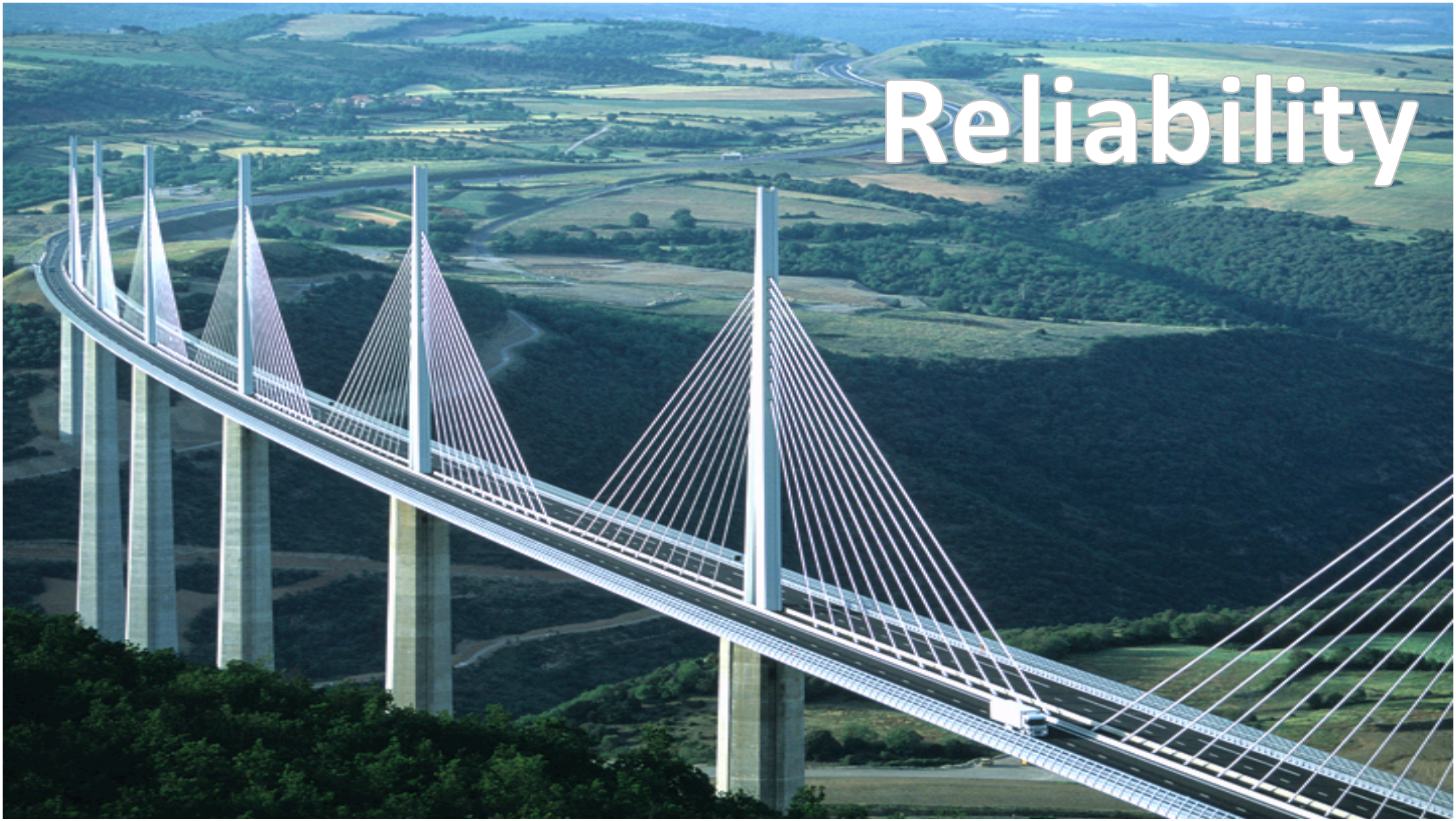


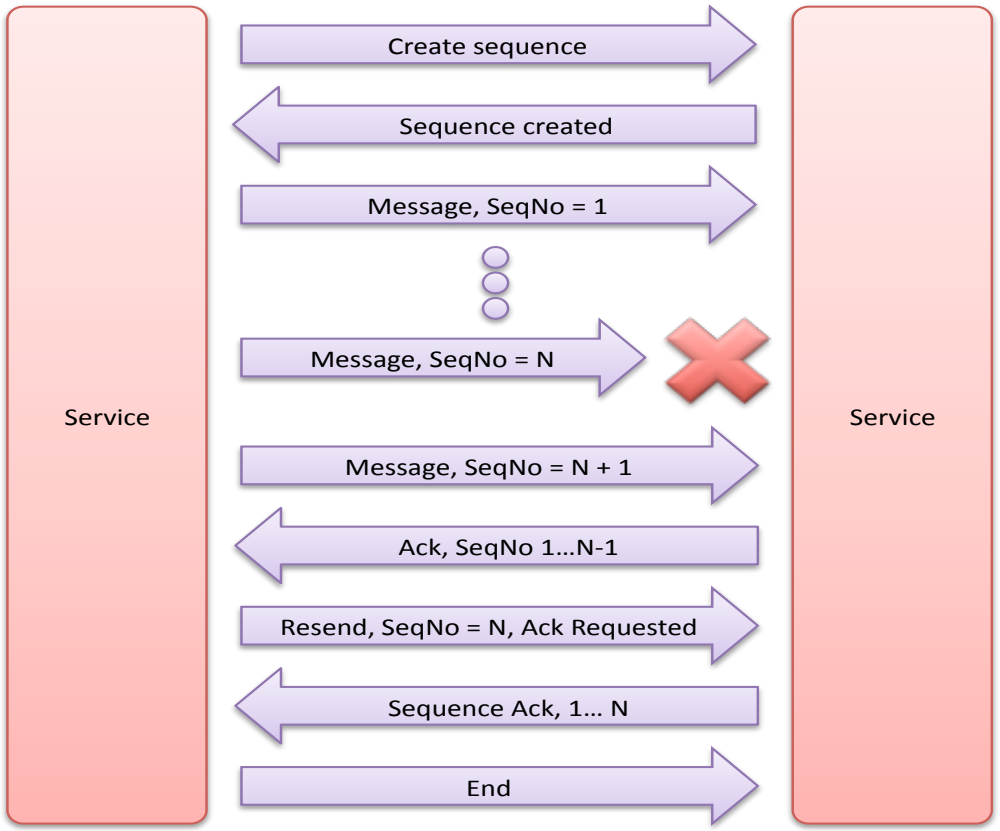

Enterprise security is
awesome, but…

…you wouldn't use it at home!

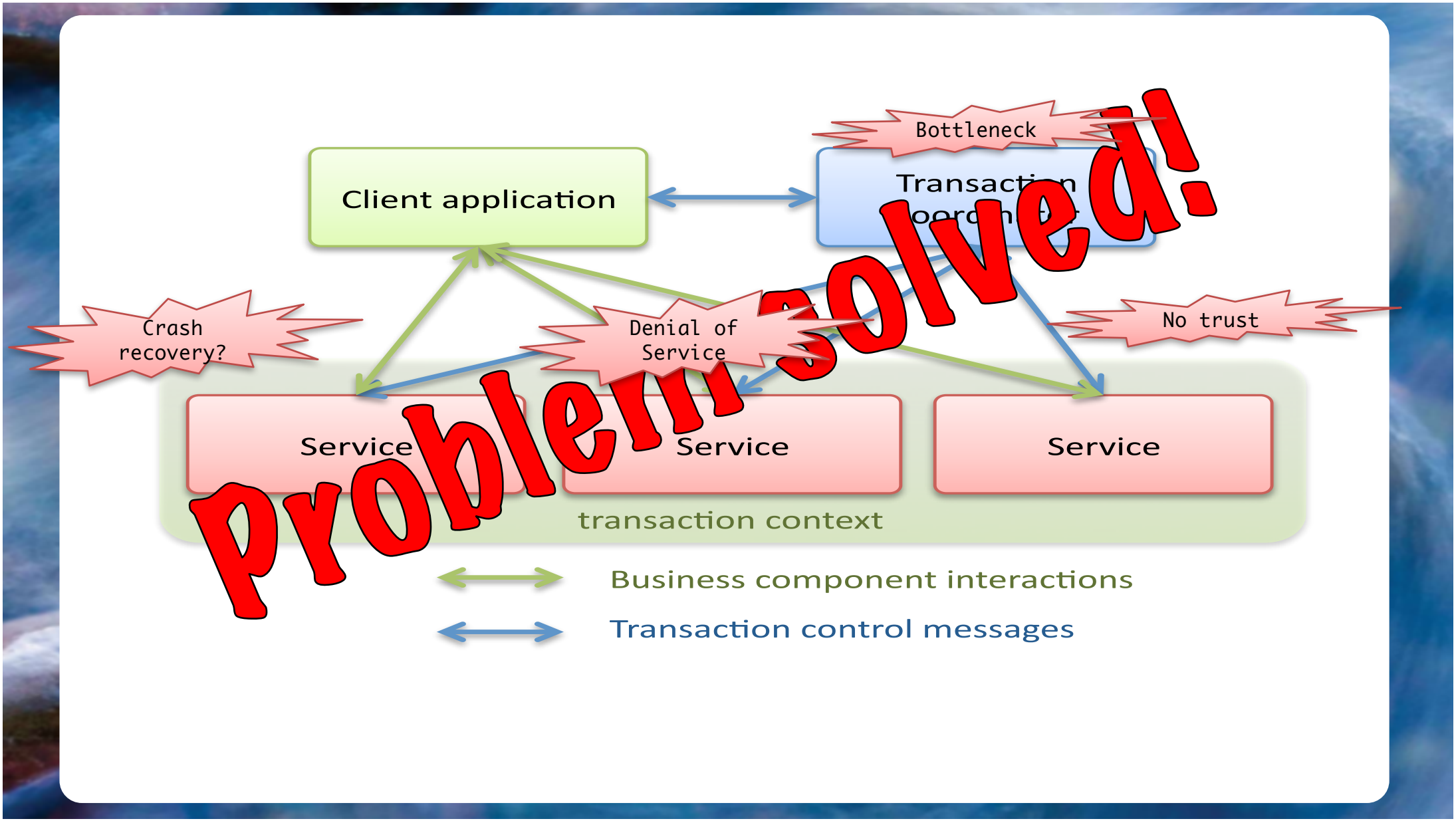
OpenID

OAUTH


Reliability


Service
Create sequence
Sequence created
Message, SeqNo = 1
Message, SeqNo = N
Message, SeqNo = N + 1
Ack, SeqNo 1...N-1
Resend, SeqNo = N, Ack Requested
Sequence Ack, 1... N
End
Service

Transactions


Bottleneck
Client application
Transaction
Crash
recovery?
Denial of
Service
No trust
Service
Service
Service
transaction context
Business component interactions
Transaction control messages
Problem solved!

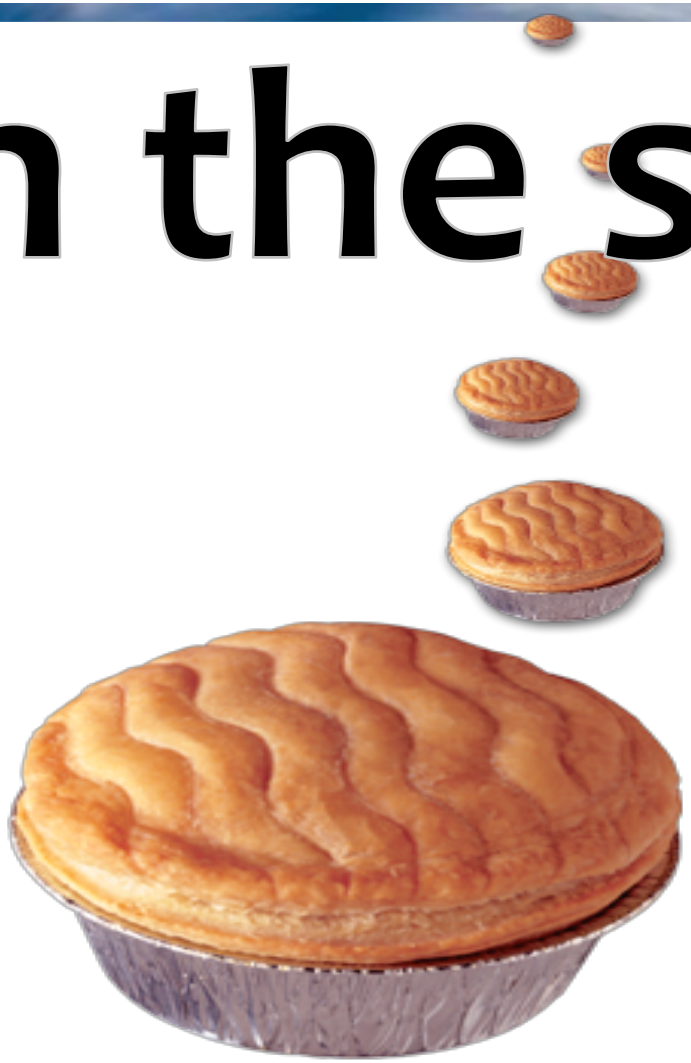

Contracts

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
   targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="sayHello">
         <soap:operation soapAction="sayHello"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </input>
         <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://localhost:8080/soap/servlet/rpcrouter"/>
      </port>
   </service>
</definitions>
```

Hello World!

The Web is a coordination platform

Pie in the sky?



Not at all

A Case Study

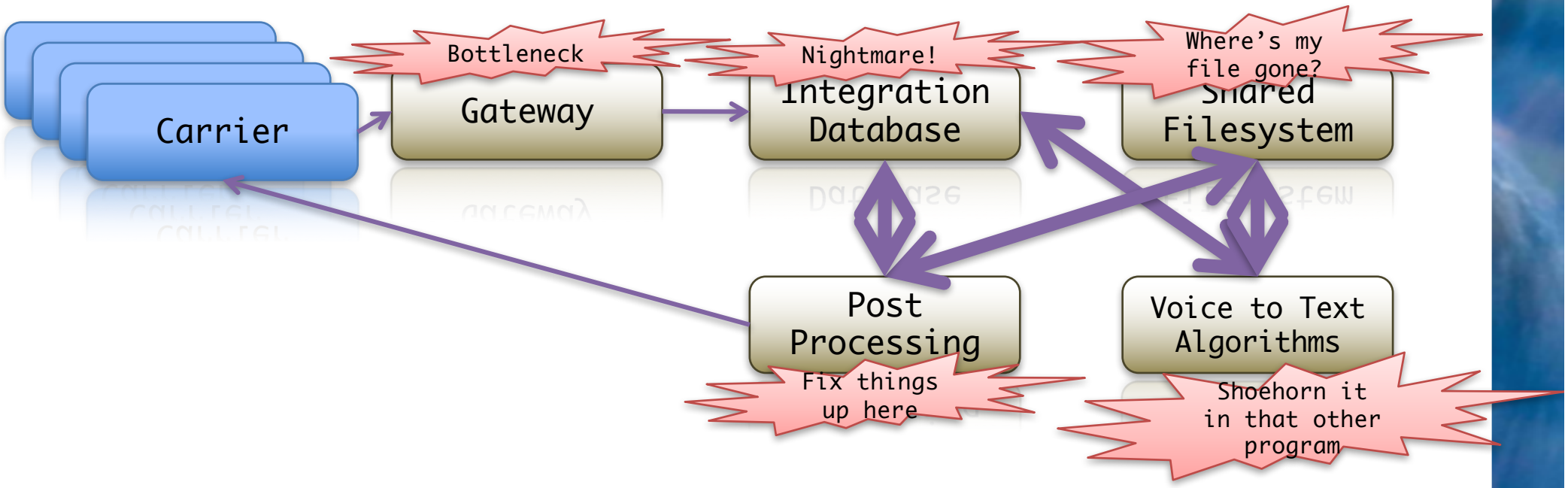
Client

- UK-based telecoms service provider
- Global customers (no easy downtime)

Problem

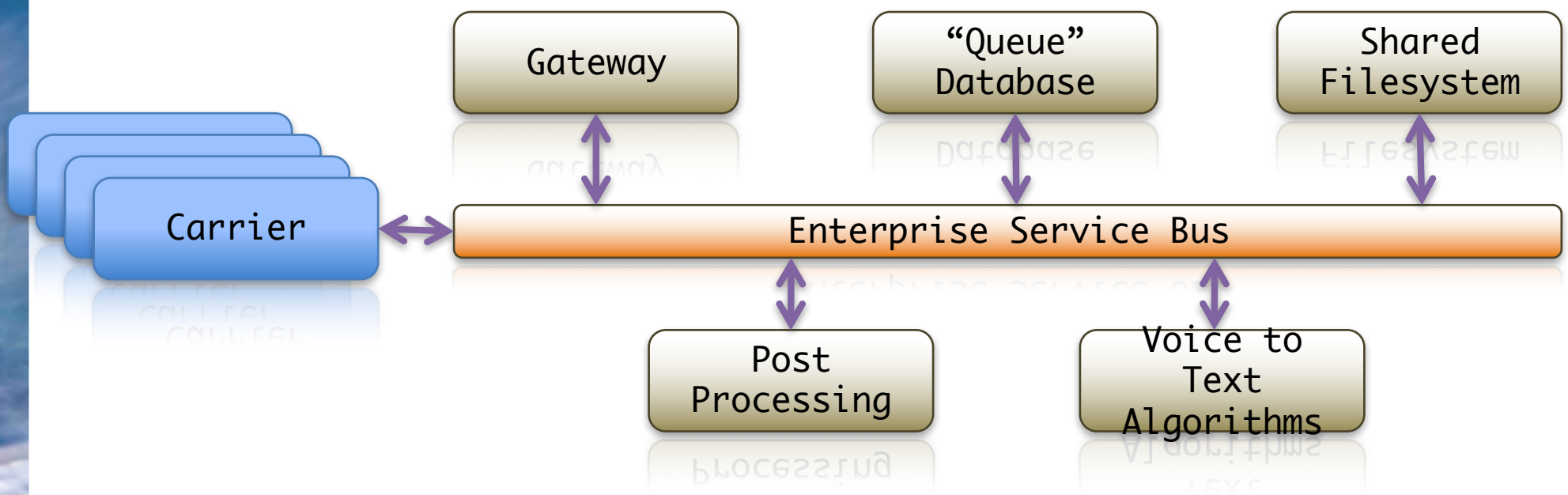
- Scalability and system resilience issues
- Huge growth curve to from a million to a billion messages per month
- Costs

Platform Architecture

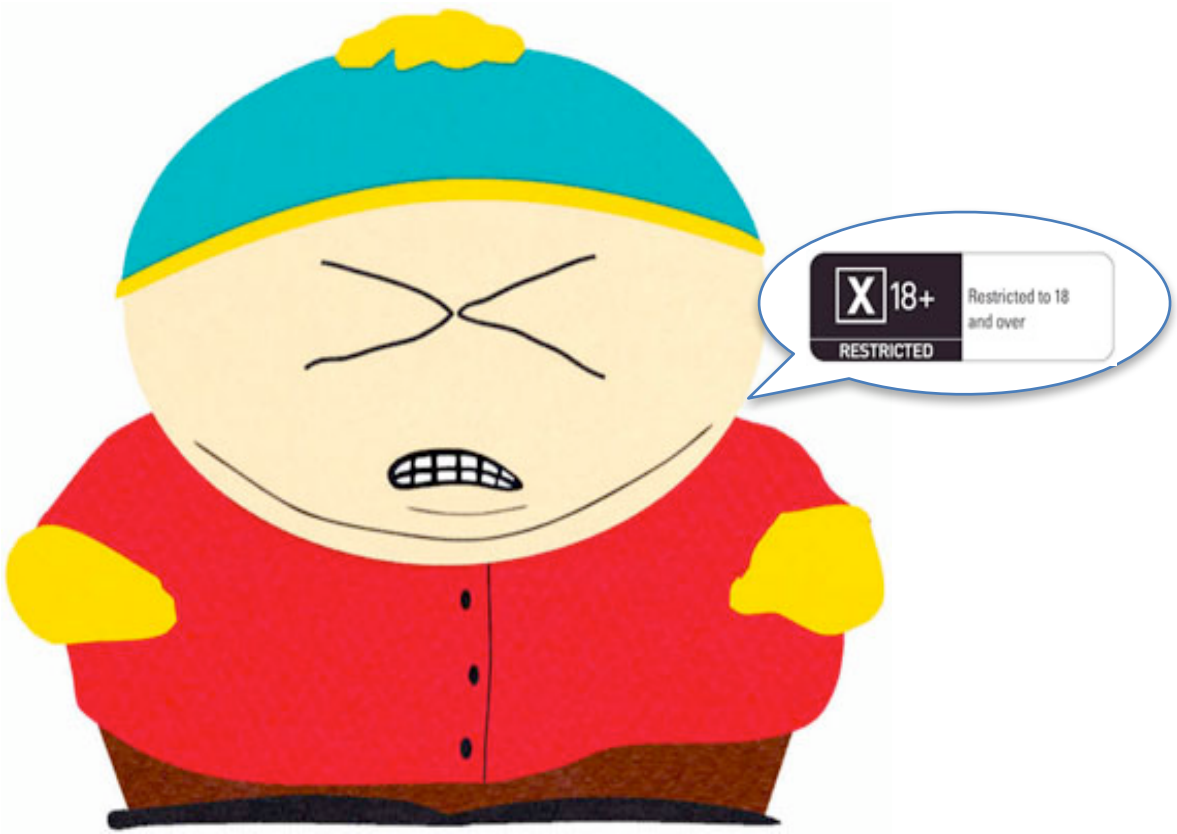



Three Months' Enterprise Consulting Later...

The £10 Million Solution *



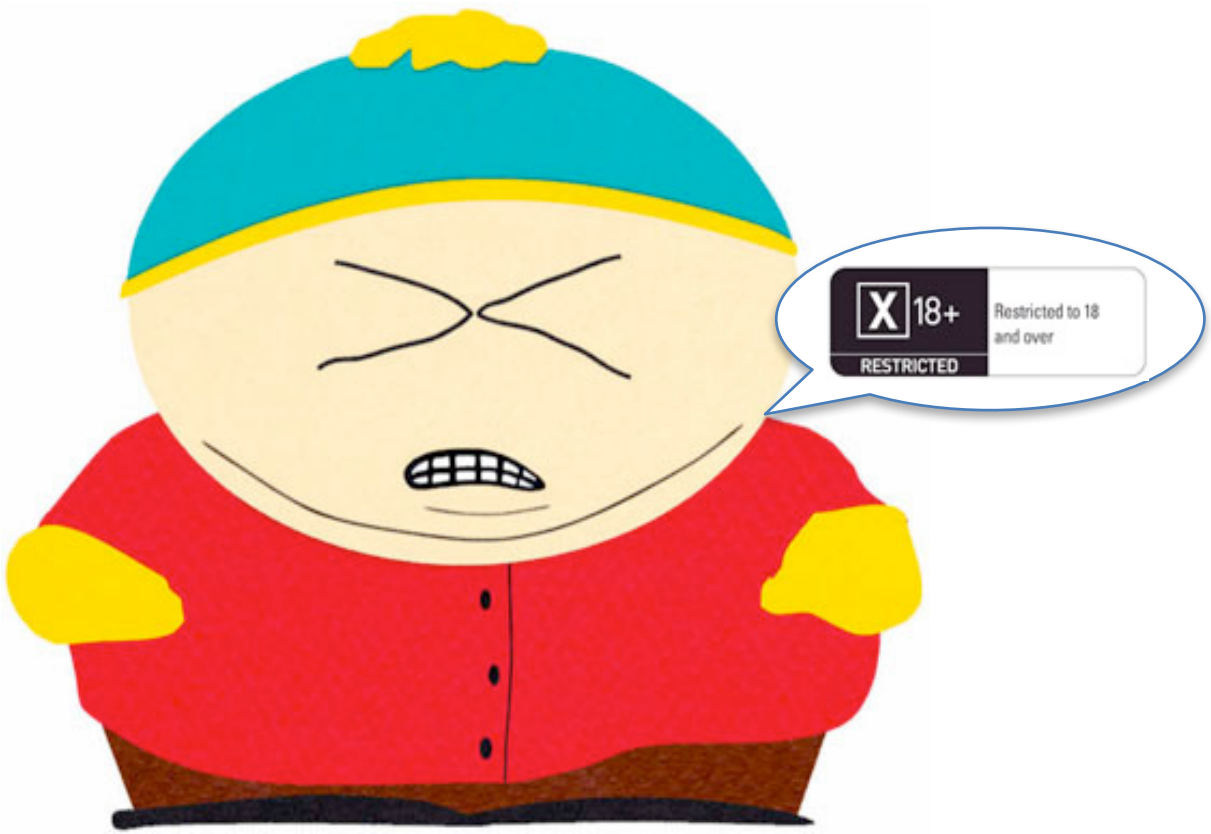
* Solution does not include actual implementation



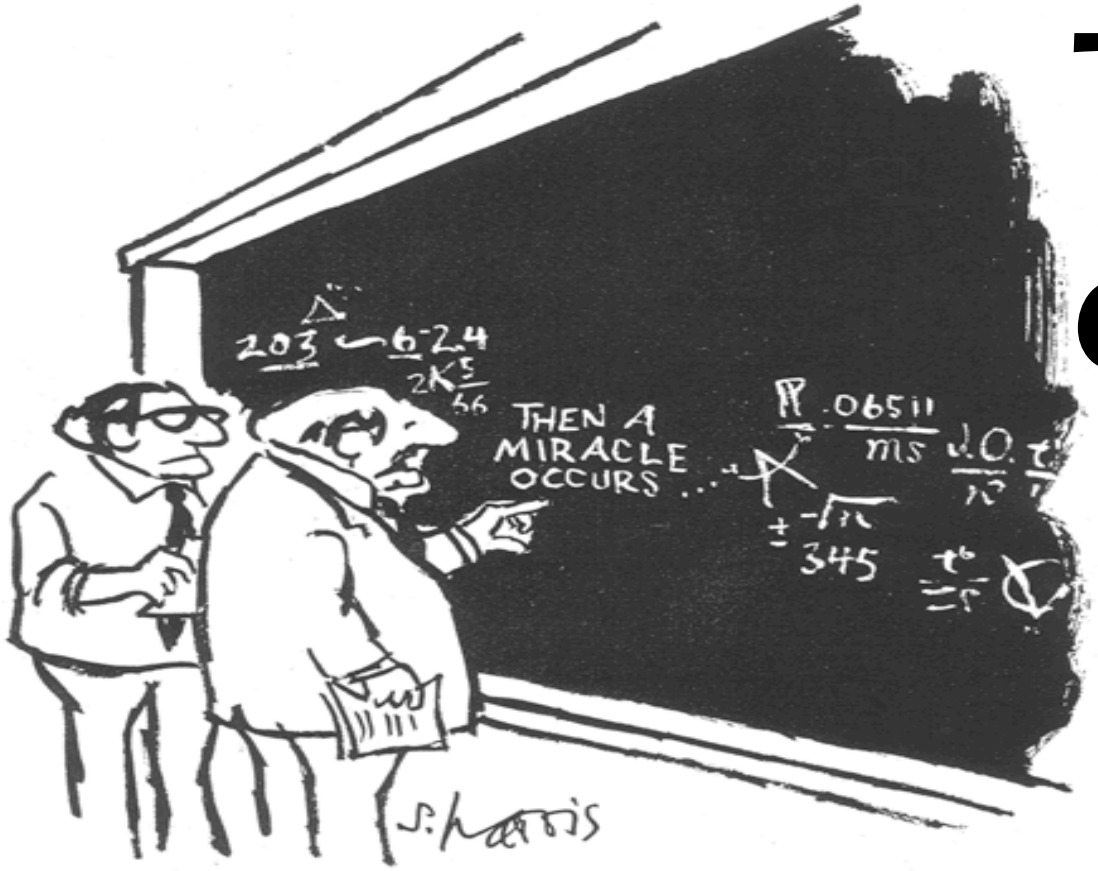
CEO's Response

We were given output from a previous enterprise consultancy…

…and told to “do that, but cheaper”

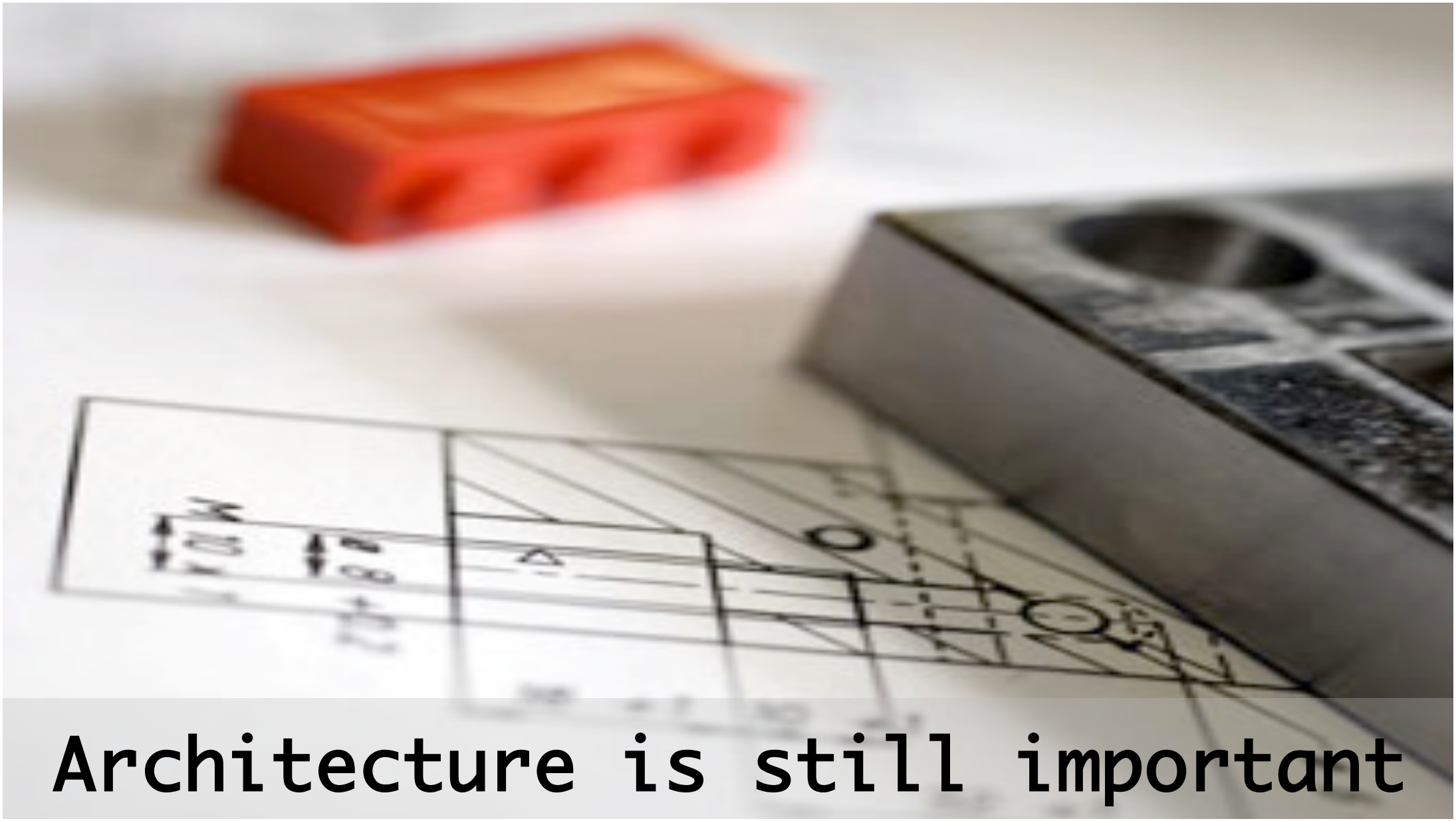


Our Response

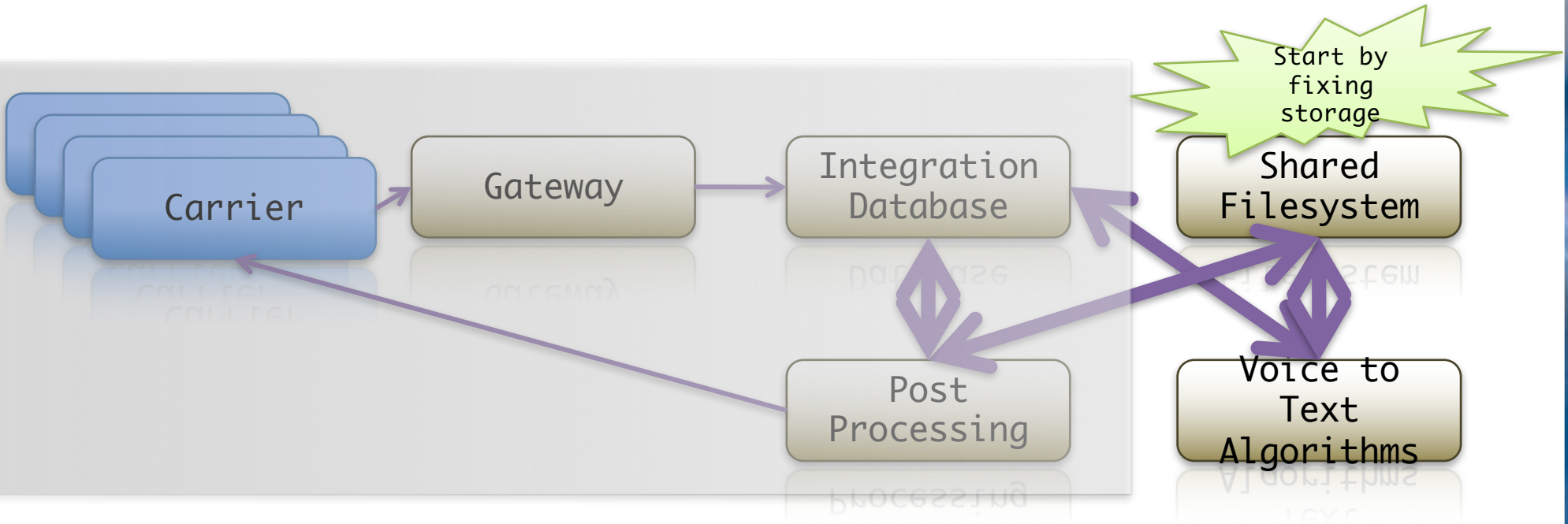


Think differently!

(or even just stop and think)


Architecture is still important

Agile EA: Prioritise Service Delivery




56
57
58
59
12
13
14
15
16
1


MODERN
SCIENCE
the
world
us
all
This is how risk is mitigated:
Not by buying
middleware up-front!

Storage Manager Project Delivery

Small team

3 Week Inception

14 iterations

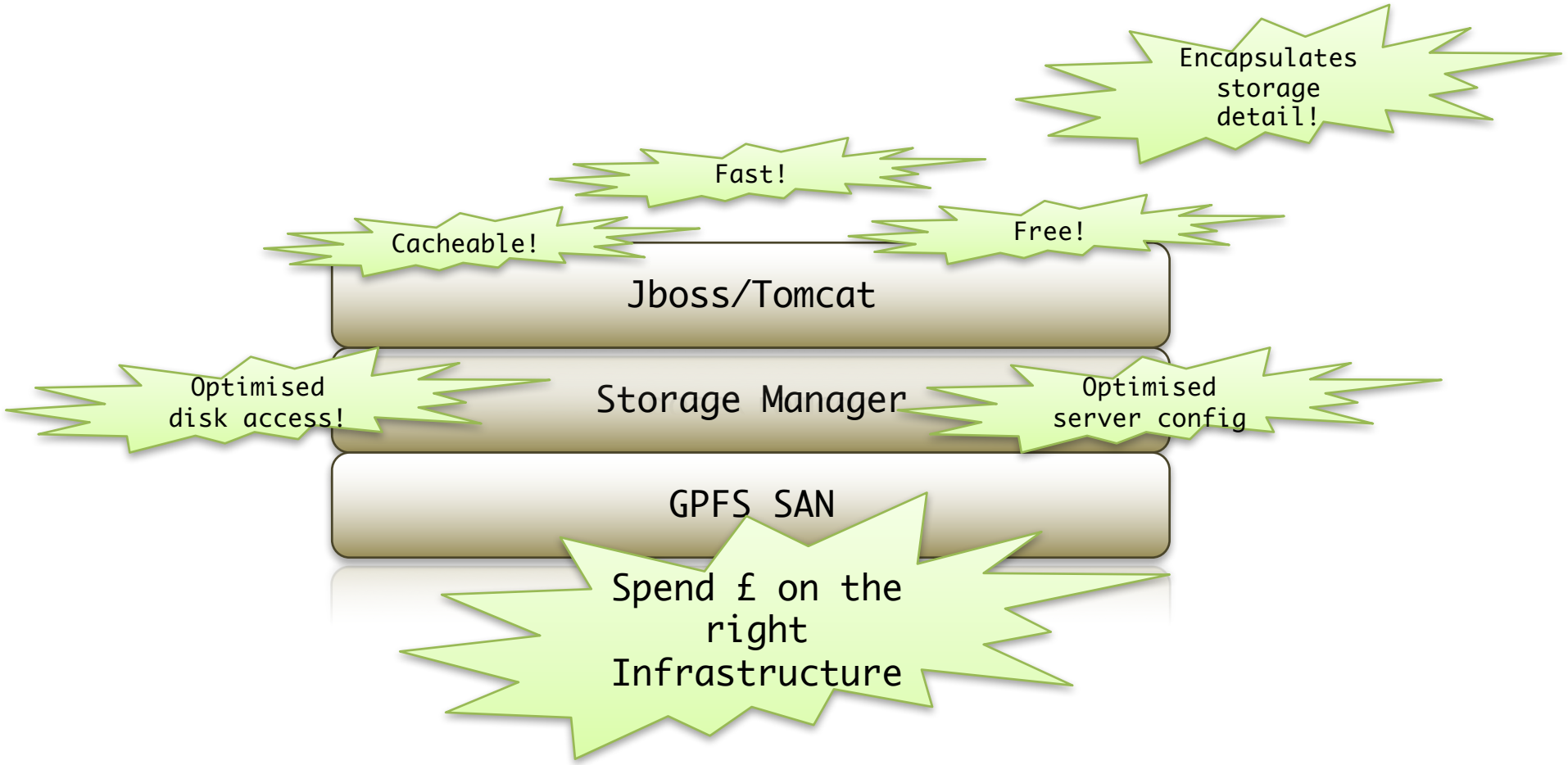
Java solution

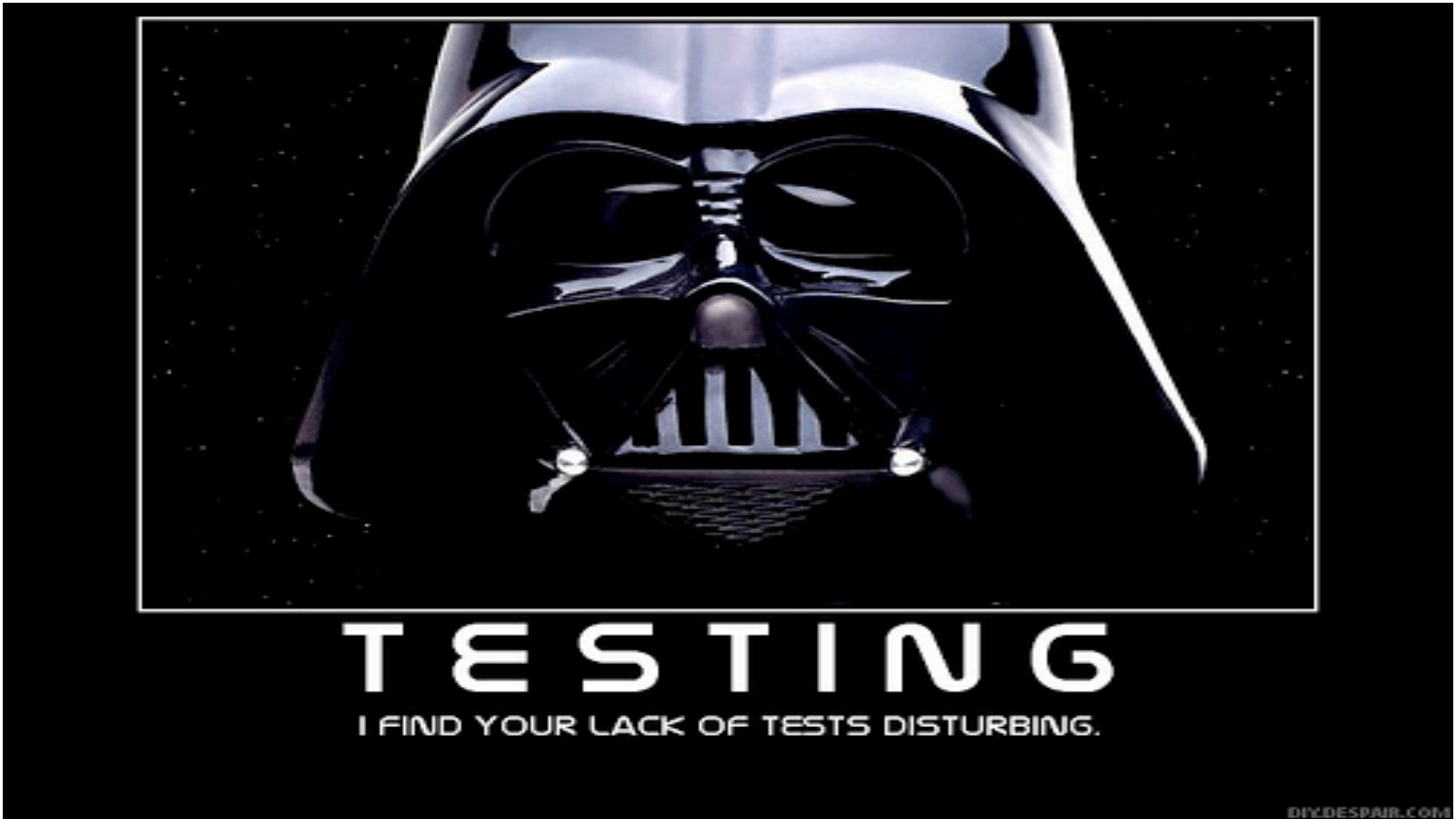
Performance testing throughout

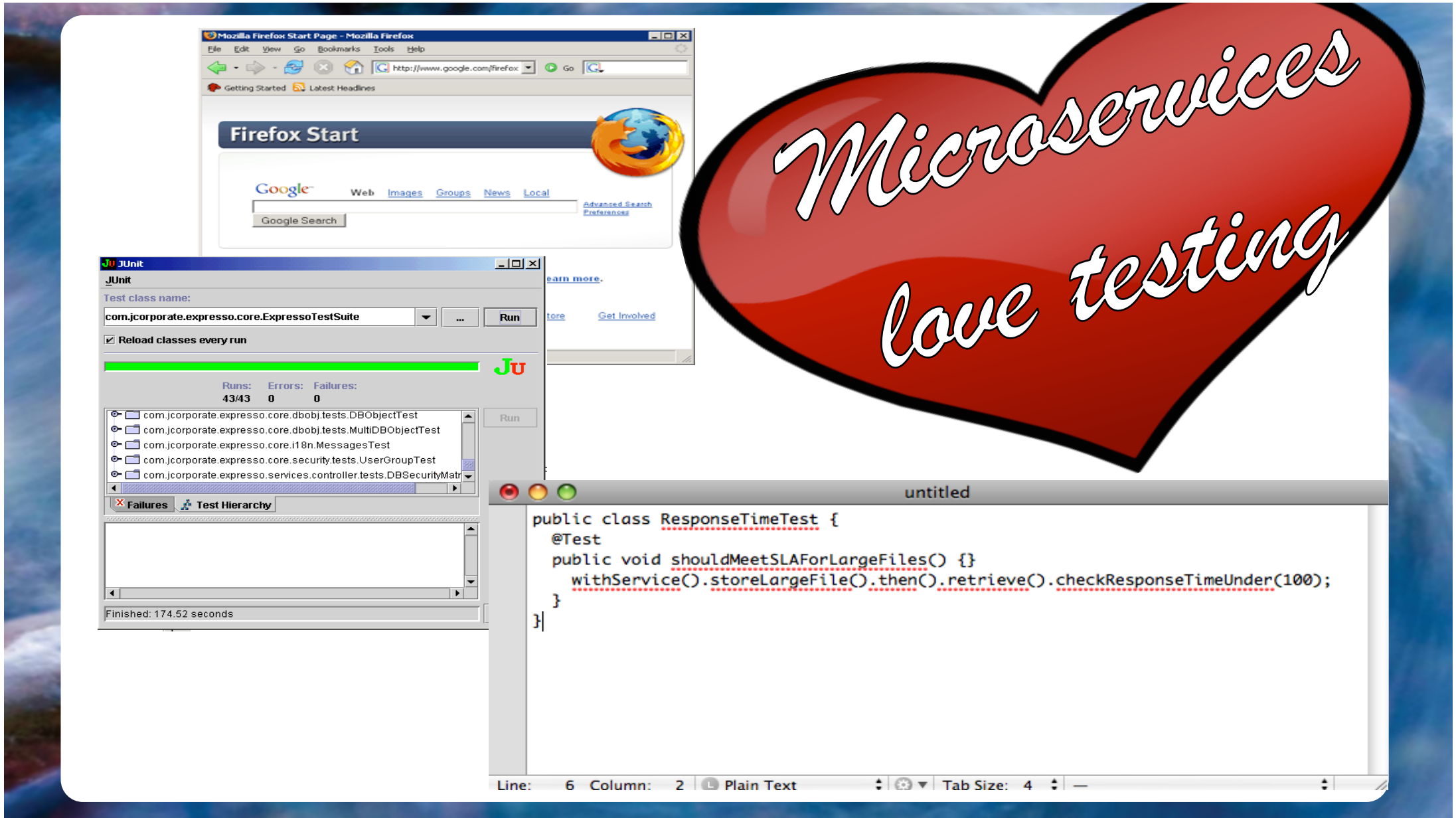
- See Jones and Kua paper, Agile 2009

Continuously built and deployed for testing

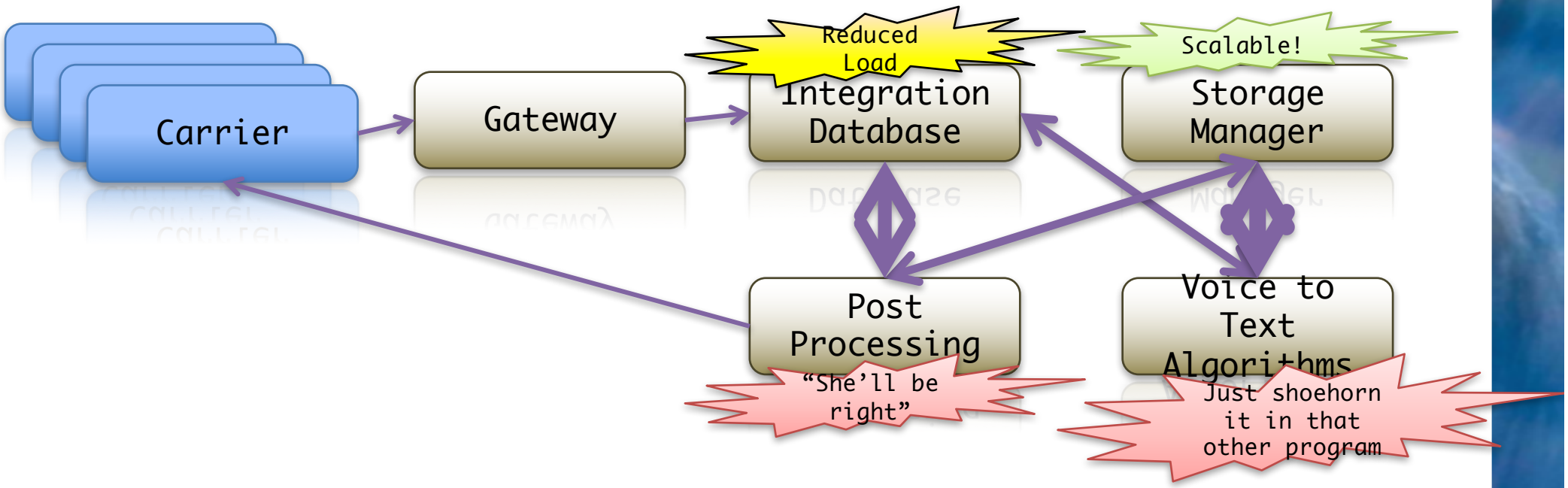
Web-friendly Architecture
(mostly wallet-friendly too!)




TESTING
I FIND YOUR LACK OF TESTS DISTURBING.
DIYDESPAIR.COM


Mozilla Firefox Start Page - Mozilla Firefox
Firefox Start
Google Search
JUnit
Test class name:
com.jcorporate.expresso.core.ExpressoTestSuite
Run
Reload classes every run
Runs: 43/43
Errors: 0
Failures: 0
com.jcorporate.expresso.core.dbobj.tests.DBObjectTest
com.jcorporate.expresso.core.dbobj.tests.MultiDBObjectTest
com.jcorporate.expresso.core.i18n.MessagesTest
com.jcorporate.expresso.core.security.tests.UserGroupTest
com.jcorporate.expresso.services.controller.tests.DBSecurityMatr
Failures
Test Hierarchy
Finished: 174.52 seconds
Microservices
love testing
untitled
public class ResponseTimeTest {
@Test
public void shouldMeetSLAForLargeFiles() {}
withService().storeLargeFile().then().retrieve().checkResponseTimeUnder(100);
}
}
Line: 6 Column: 2 Plain Text Tab Size: 4

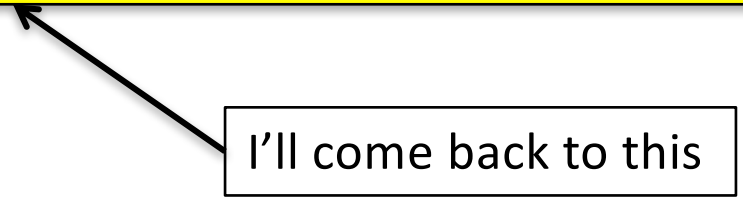
Improving Enterprise Architecture



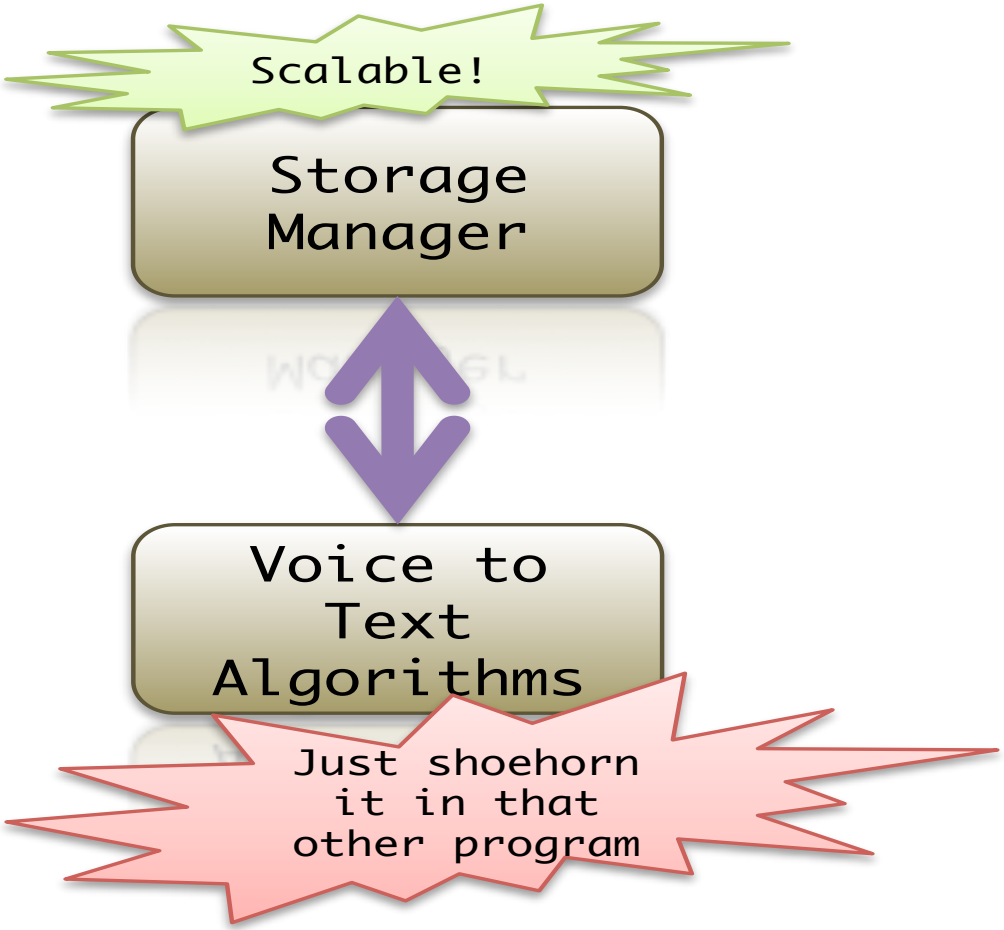
Storage Manager Benefits

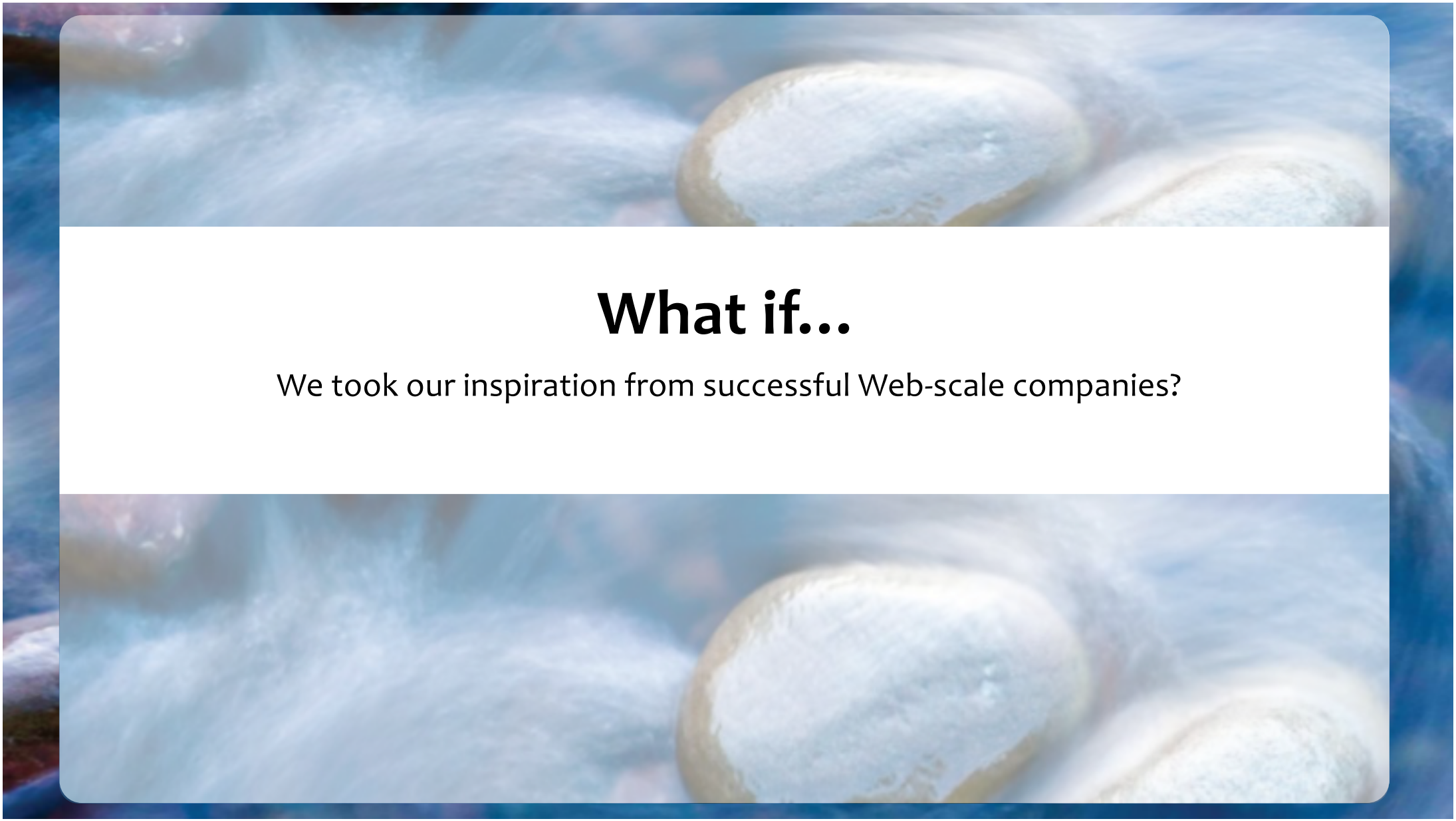
- Immediate reduced load on integration database
- Enabled removal of code from systems
- **Delivered business value by keeping customers happy**

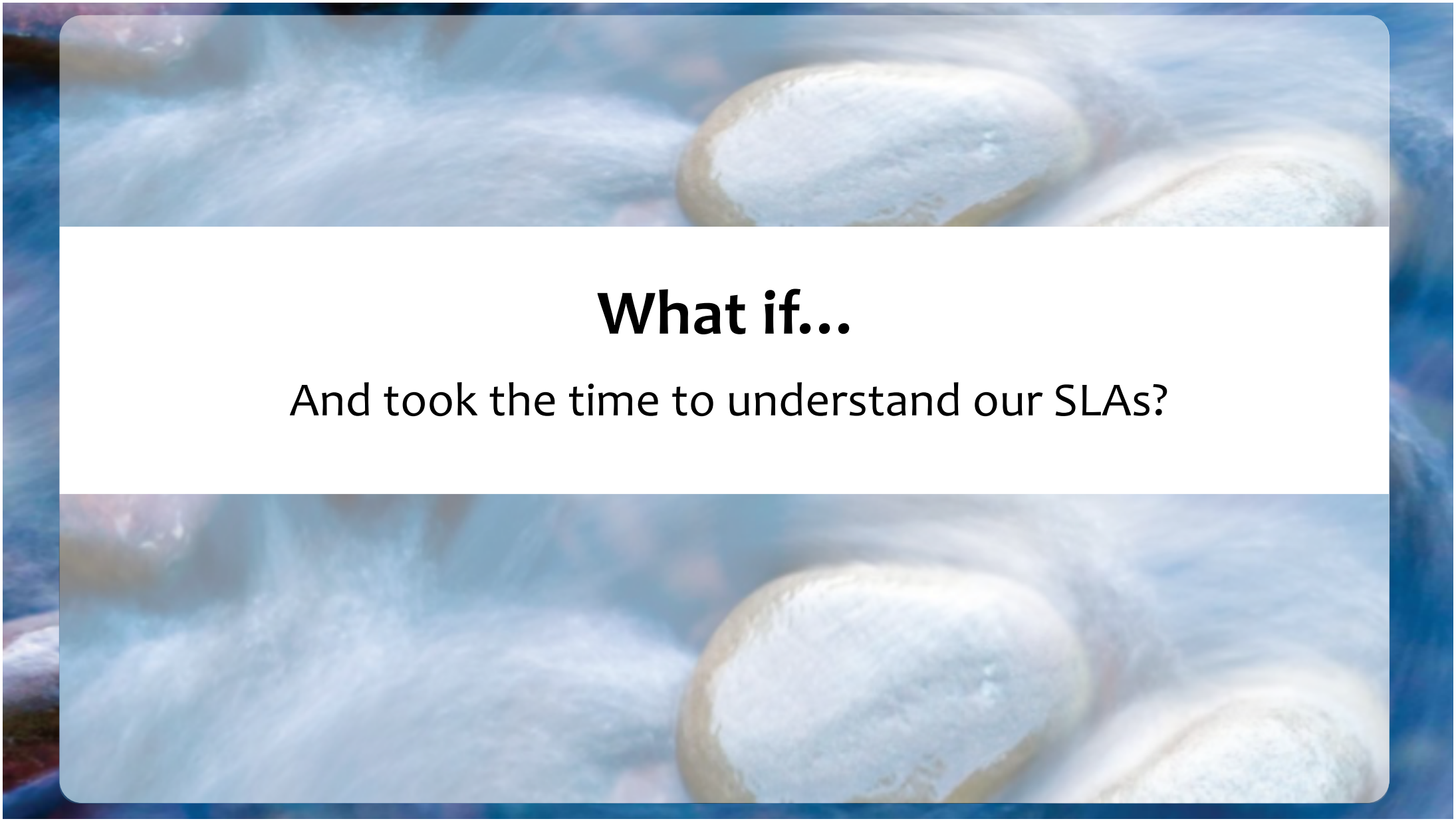
- "The best software we have ever deployed"
  - Operations director

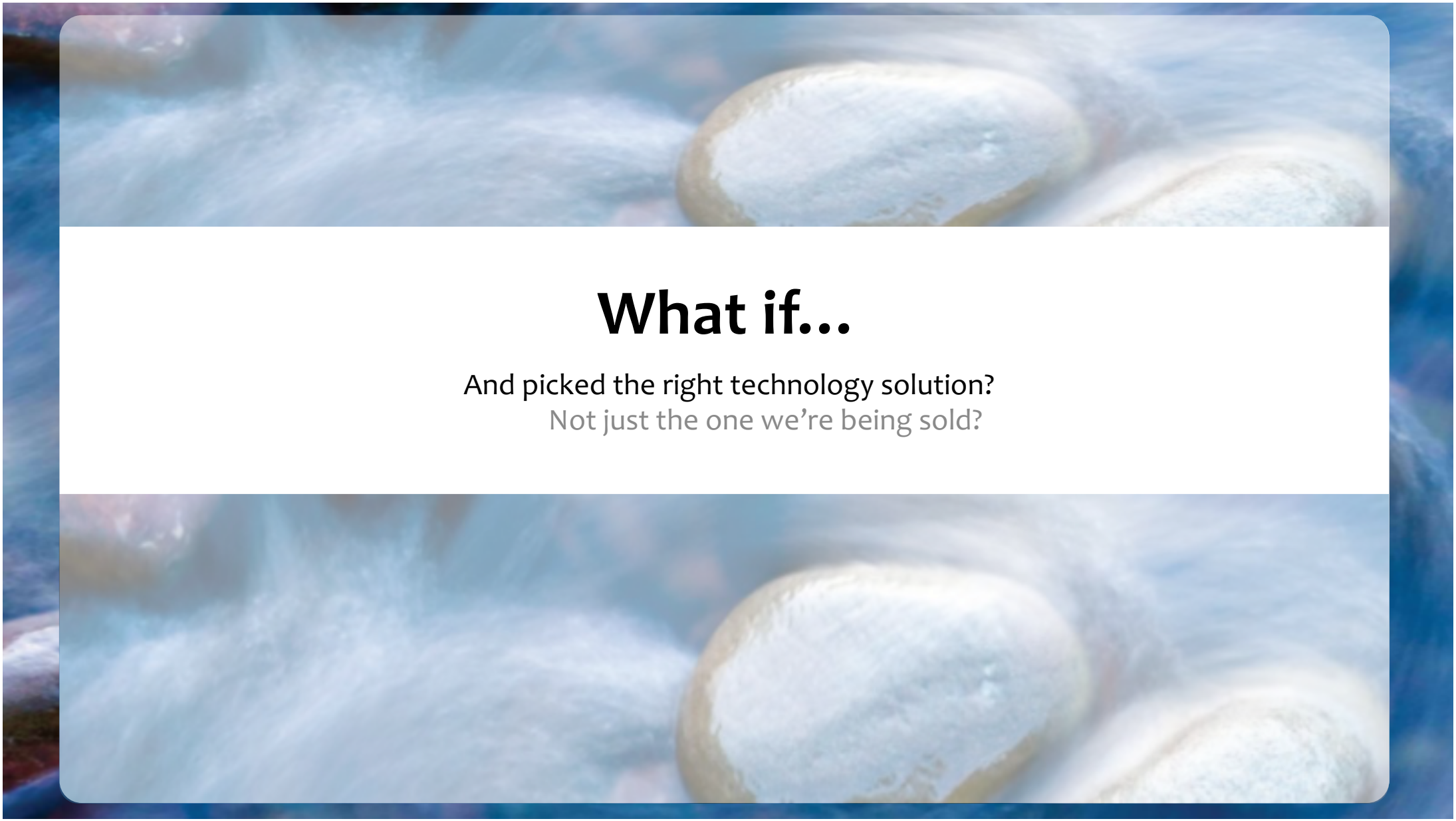


What's wrong with this picture?




What if…
We took our inspiration from successful Web-scale companies?


What if…
And took the time to understand our SLAs?


What if…
And picked the right technology solution?
Not just the one we're being sold?


What if…
We built a scalable compute platform?

Grid Project Delivery

Larger team

3.5 Week Inception

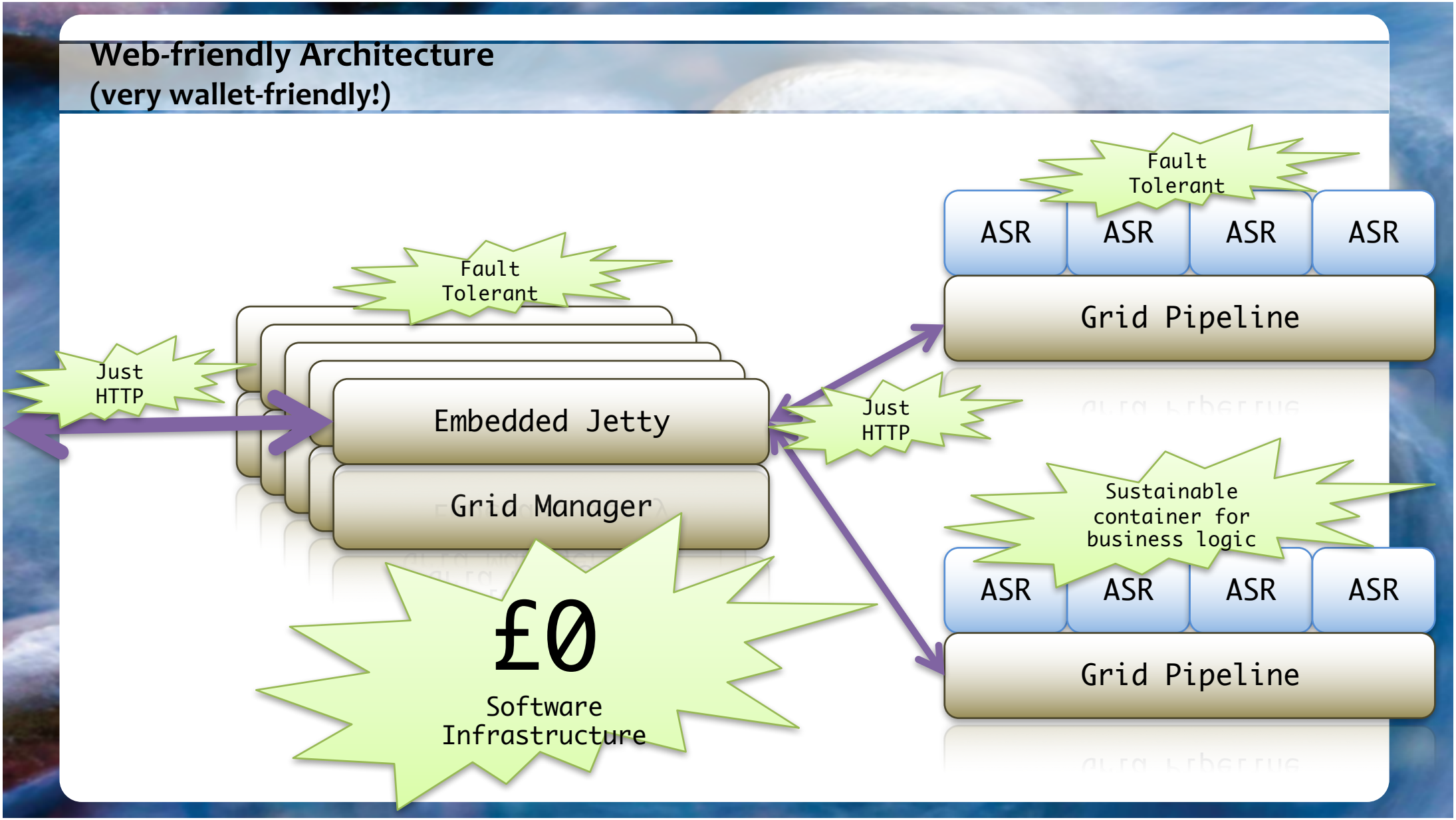
18 iterations

Java solution

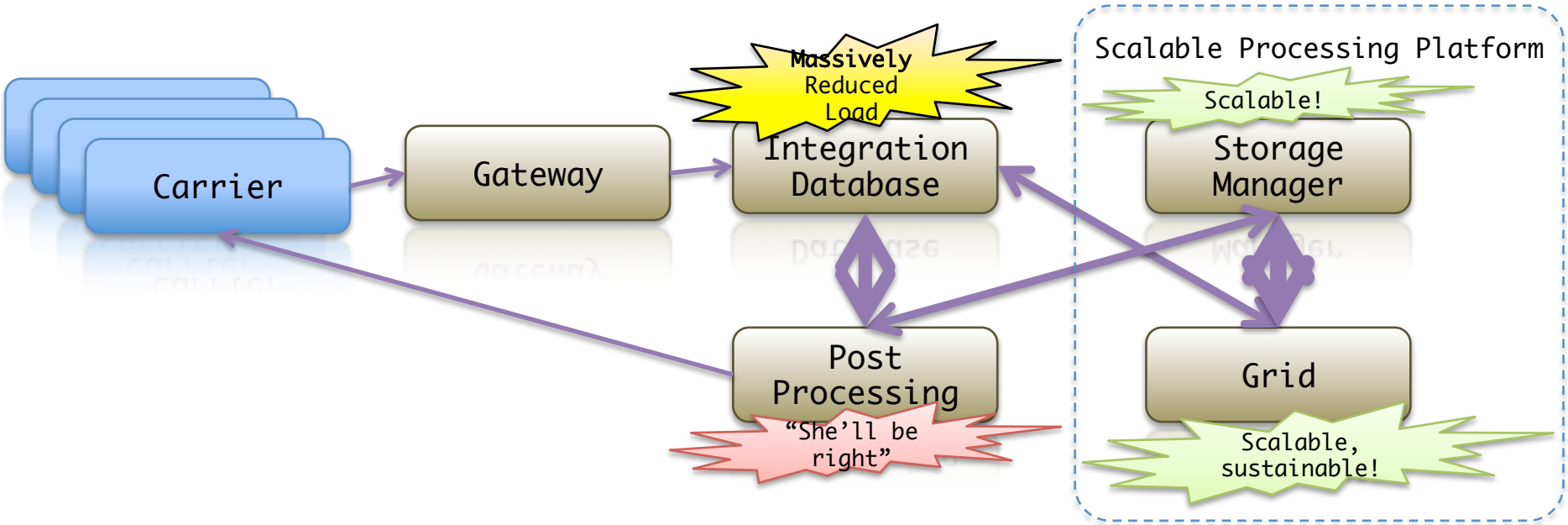
Performance testing throughout

Continuously built and deployed for testing

Web-friendly Architecture
(very wallet-friendly!)



Improving Enterprise Architecture



Grid Benefits

- Scalable, resilient platform
- Aligned with business goals
- Further reduced load on integration database
- Set architectural patterns
- **Deliver business value by processing more messages at lower cost than ever before**

Better every time!

- "The best software we have ever deployed"
  - Operations director


£10,000,000
Up-Front Cost of Bus Architecture


£1,000,000
Actual Cost of Completed Project


£0
Cost of Middleware


Microservices Work
And the Web is the most sensible underlay for them

Summary

- Be of the Web
- Model services as bounded contexts
- Join across contexts with Hypermedia
- Defy enterprise orthodoxy for fun and profit