Scaling Applications with Microservices and NServiceBus

Microservices and the Service Bus

Roland Guijt

@rolandguijt | www.rmgsolutions.nl

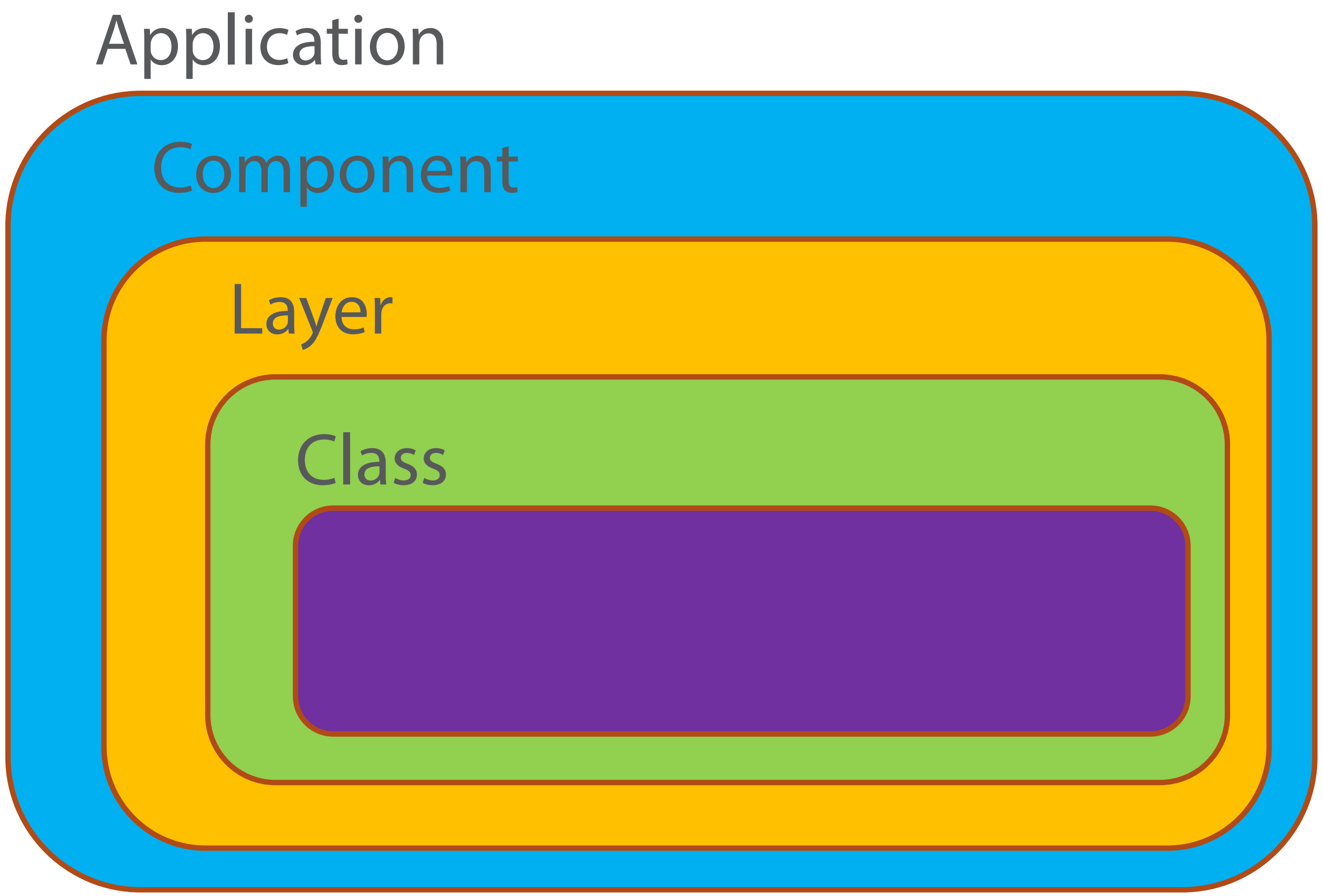
Overview

- Monolith
- Distributed applications
- Microservices
- Servicebus

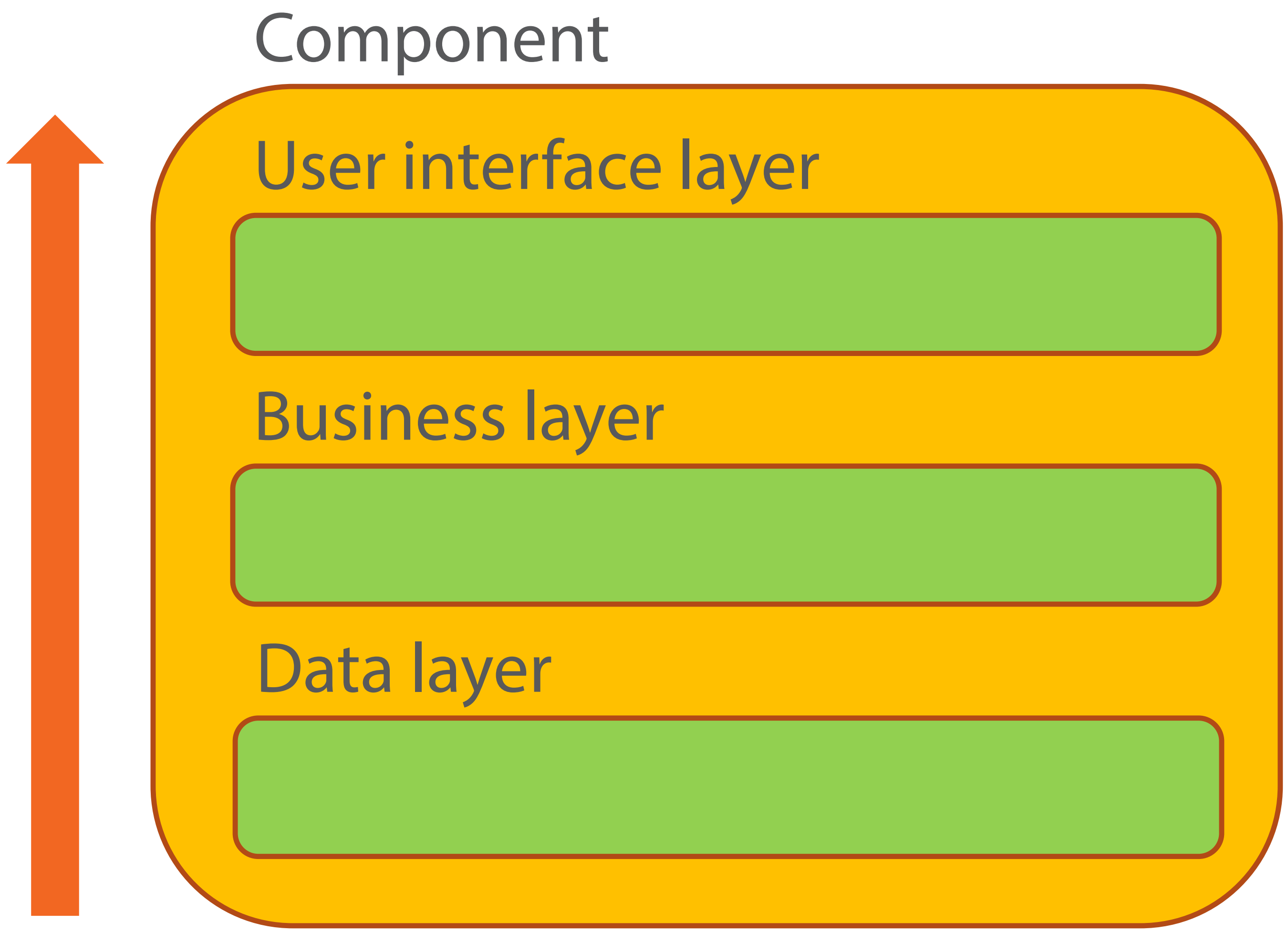
Monolithic application

A single-tiered software application in which the user interface and data access code are combined into a single program from a single platform

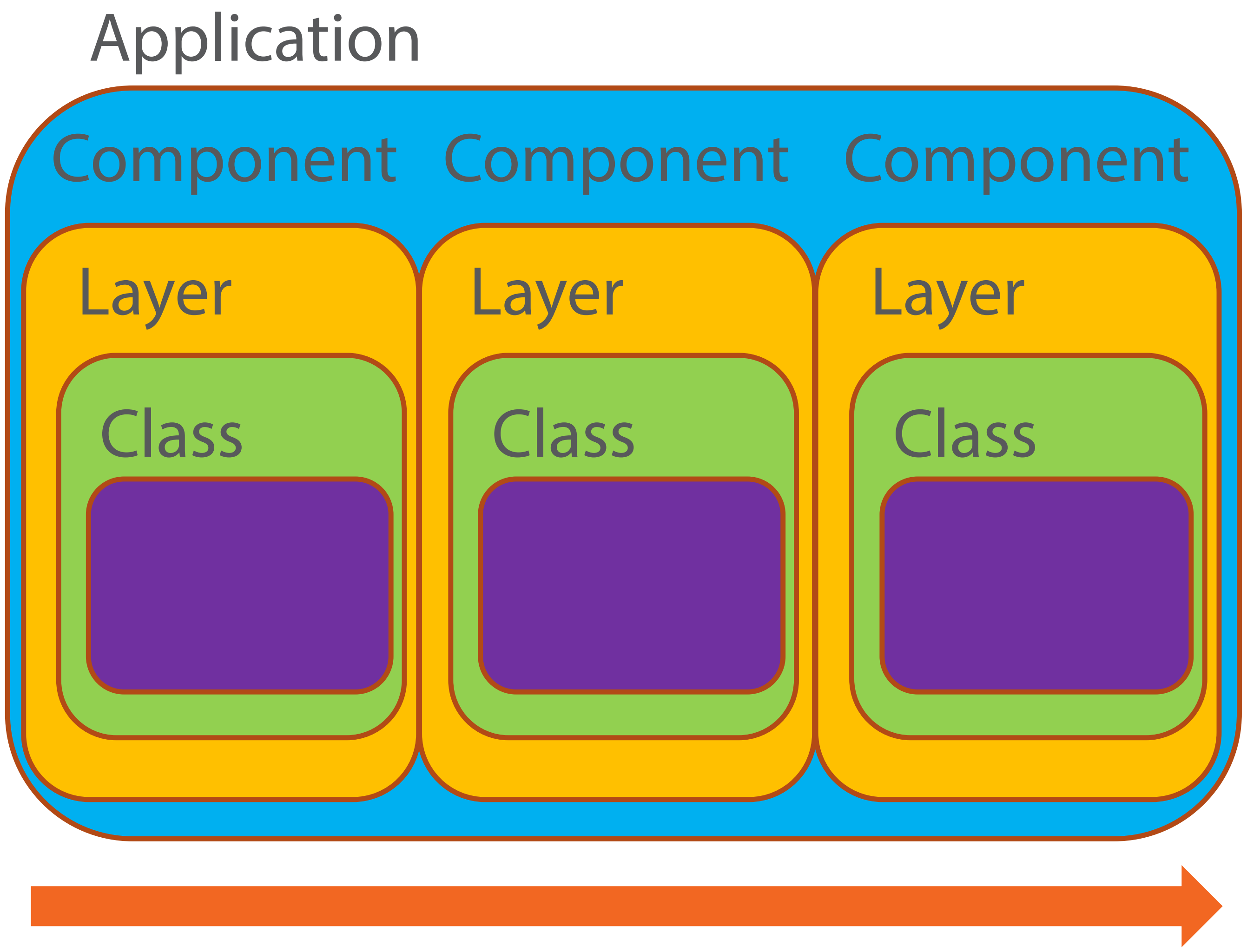
General Architecture of an App



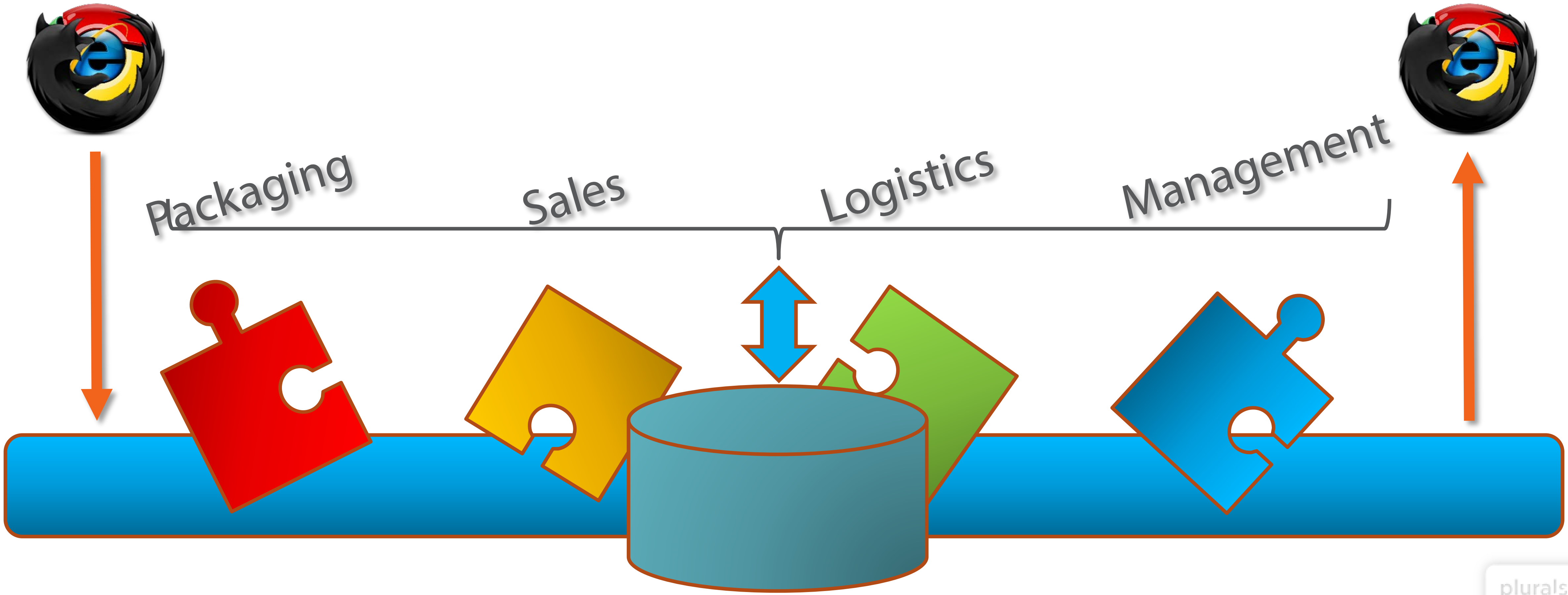
Vertical Coupling



Horizontal Coupling



Request Handling



Benefits of Monolithic Applications

Easy to deploy

Well known

No external dependencies

IDE friendly

Possible Problems

Downsides of Monolithic Applications

- If complex hard to maintain
- Tends to get complex
- Release hardening
- Performance
- Reliability
- One stack

Demo: Simplified Monolith

FIRE ON WHEELS

Startup in the package delivery business

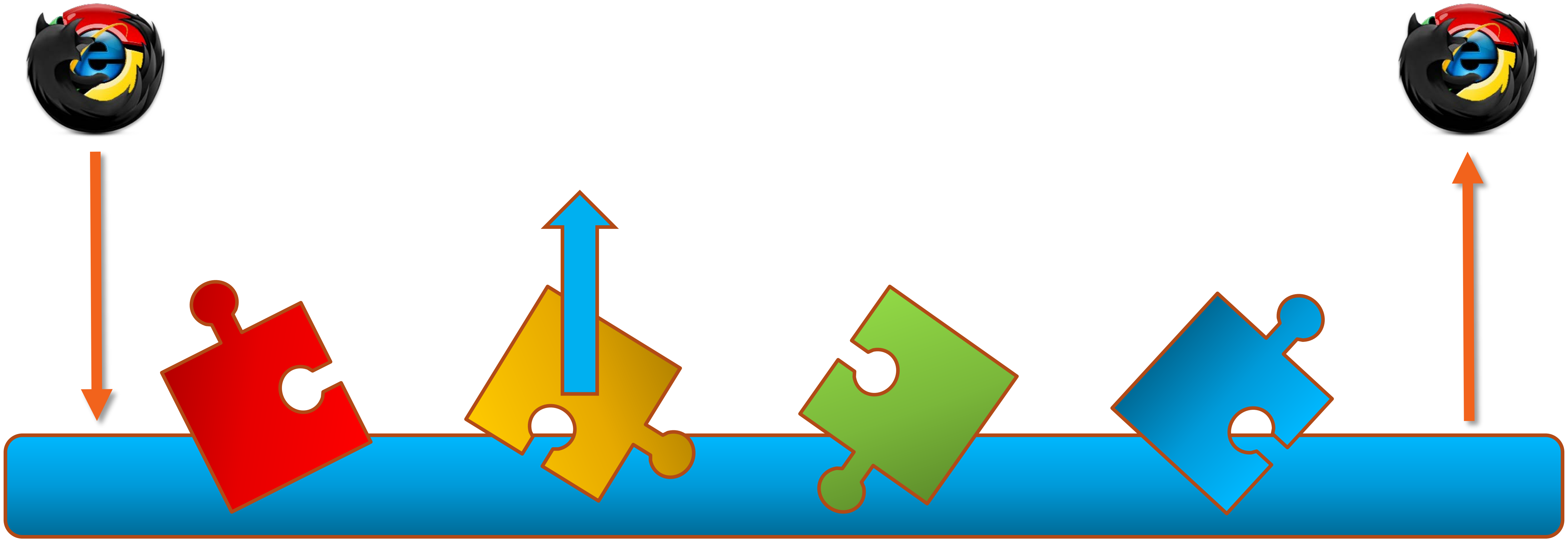
Customers can enter package delivery orders in a simple ASP.NET MVC web application

It sends an email to the person responsible for delivering the package

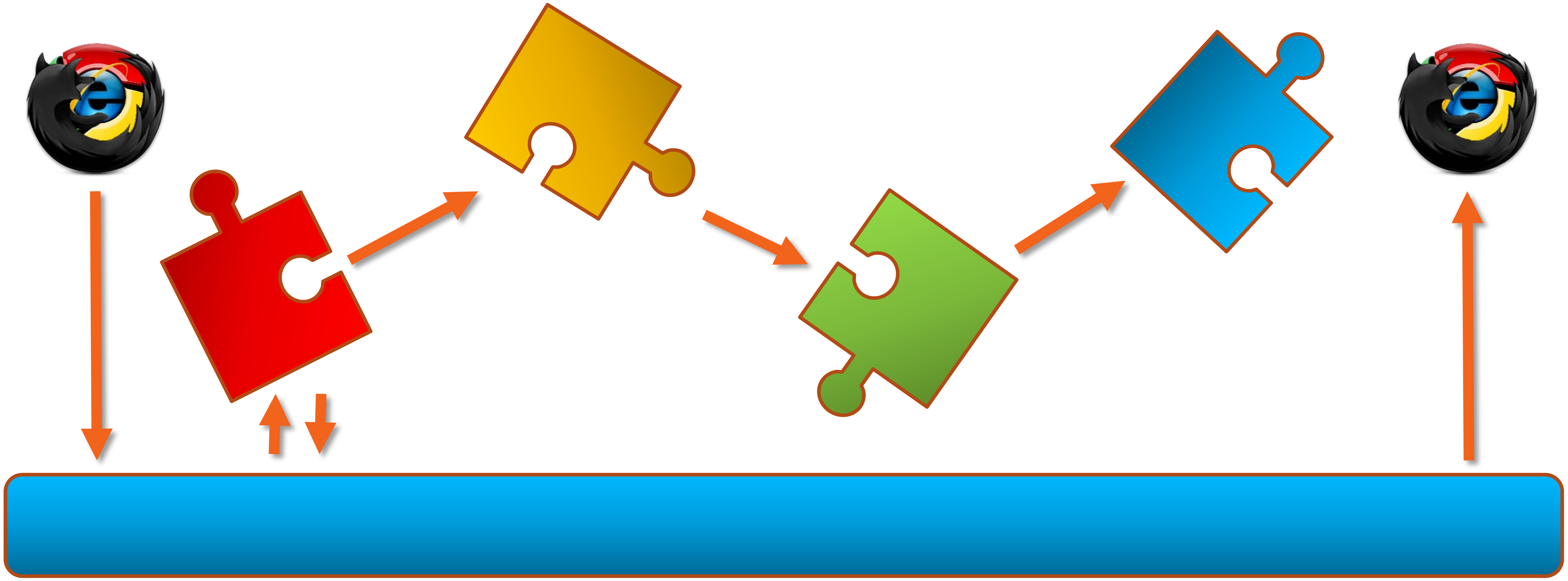
Distributed system

A distributed system is a software application in which components located on networked computers communicate and coordinate their actions by issuing calls or passing messages

Distributing Components



Service-oriented Architecture



The Fallacies of Distributed Computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology won’t change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

High Cohesion

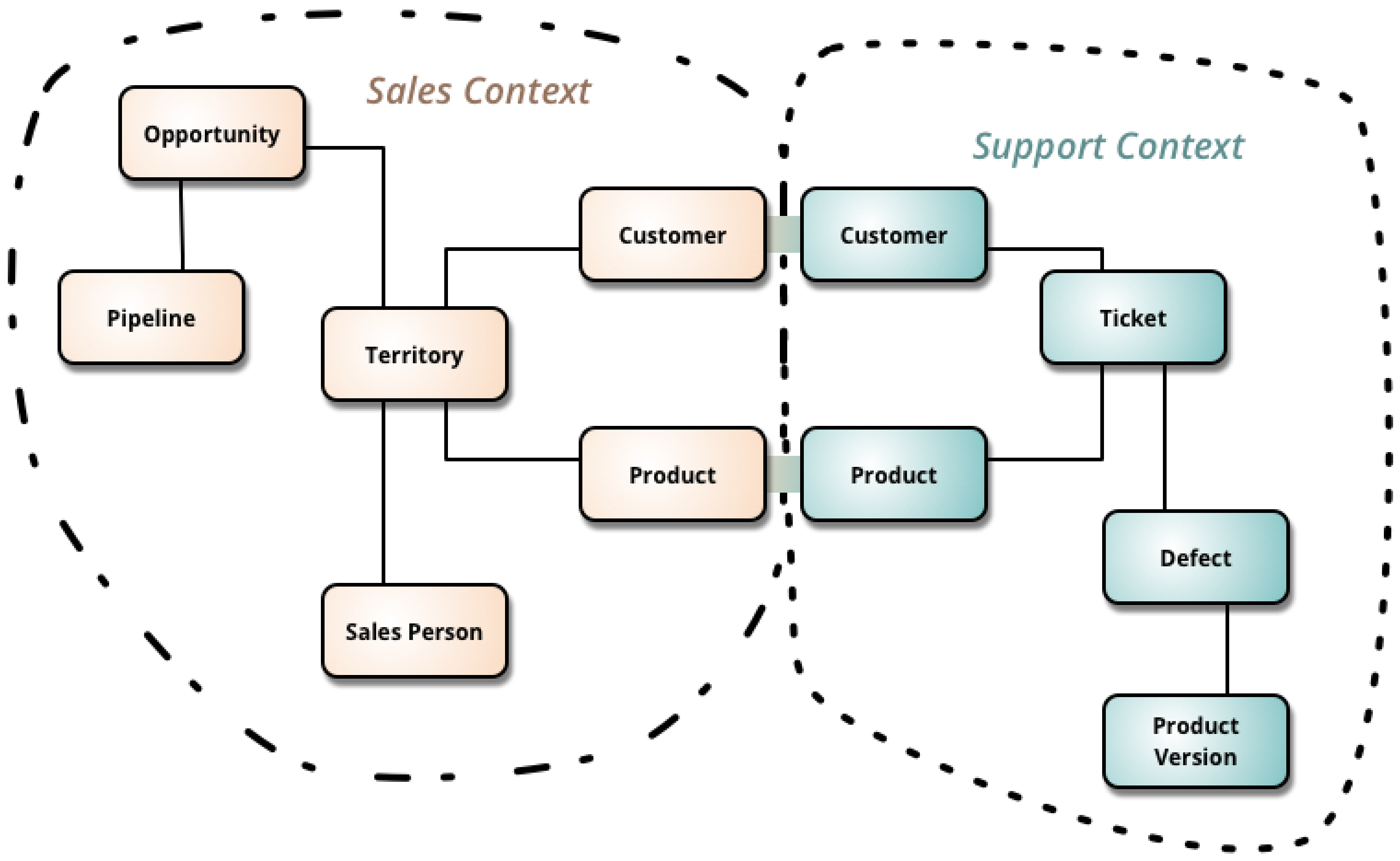
Related functionality together

Hide complexity

Before You Begin

- Make a diagram of your services
- Overlap of entities, business rules, functionality is unavoidable
- Include the boundaries of the services in your diagram
- Result: Best possible (start of) architecture up front

Bounded Context

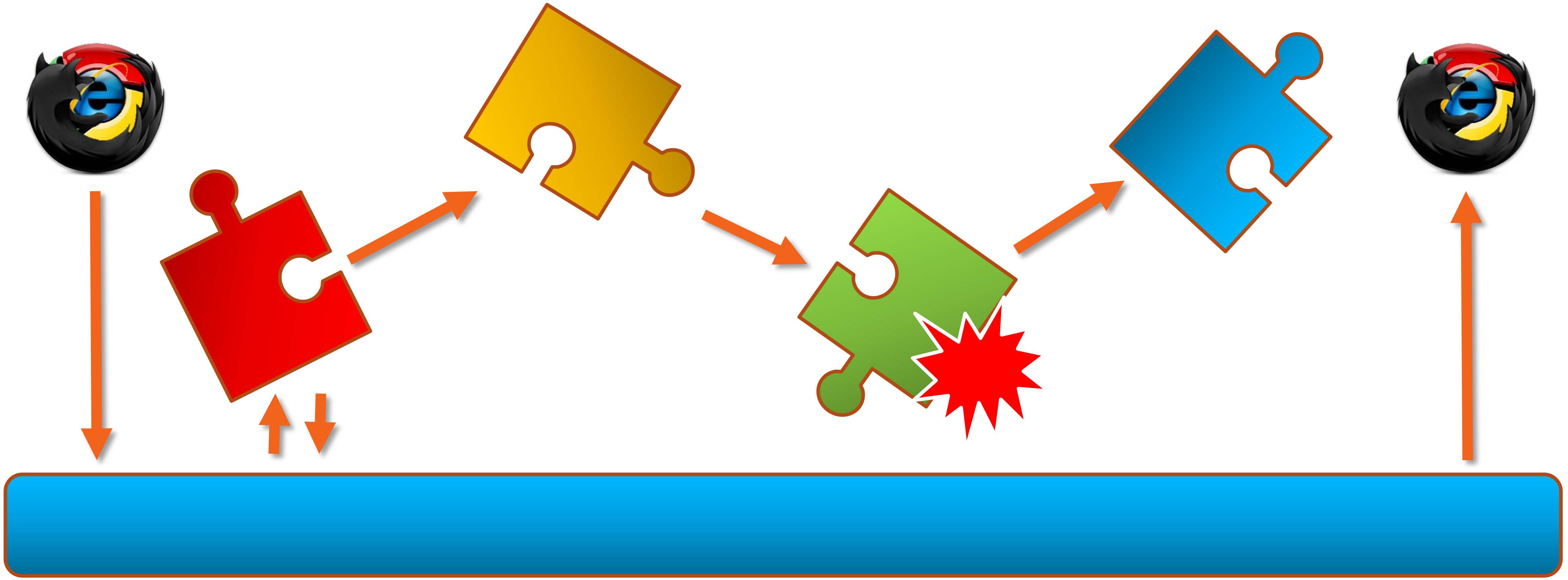


Picture by Martin Fowler

Coupling

Platform	Behavioral	Temporal

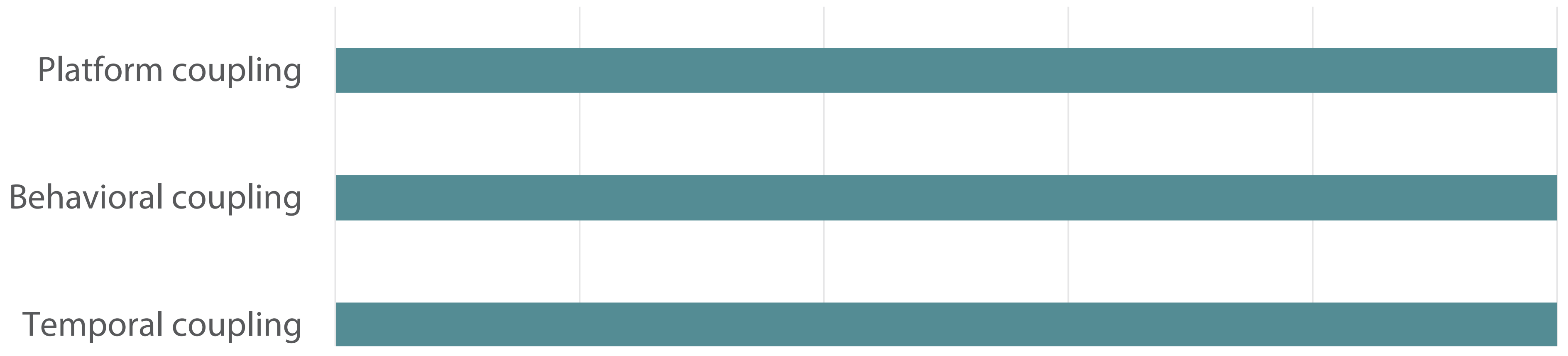
Service-oriented Architecture



Remote Procedure Call

Call a method over the wire

.NET Remoting/Java RMI



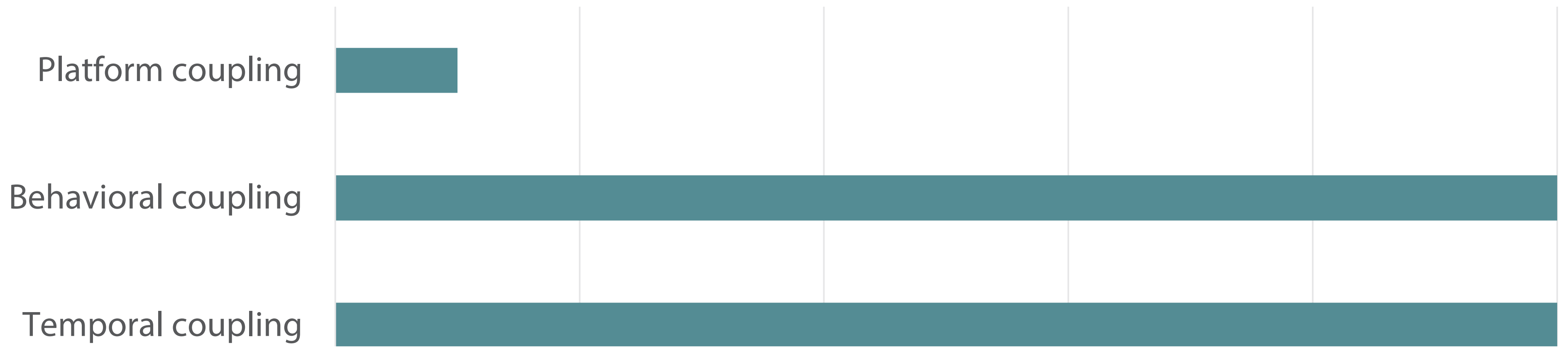
Remote Procedure Call

- Tends to generate proxy classes
- Beware of the fallacies of distributed computing!

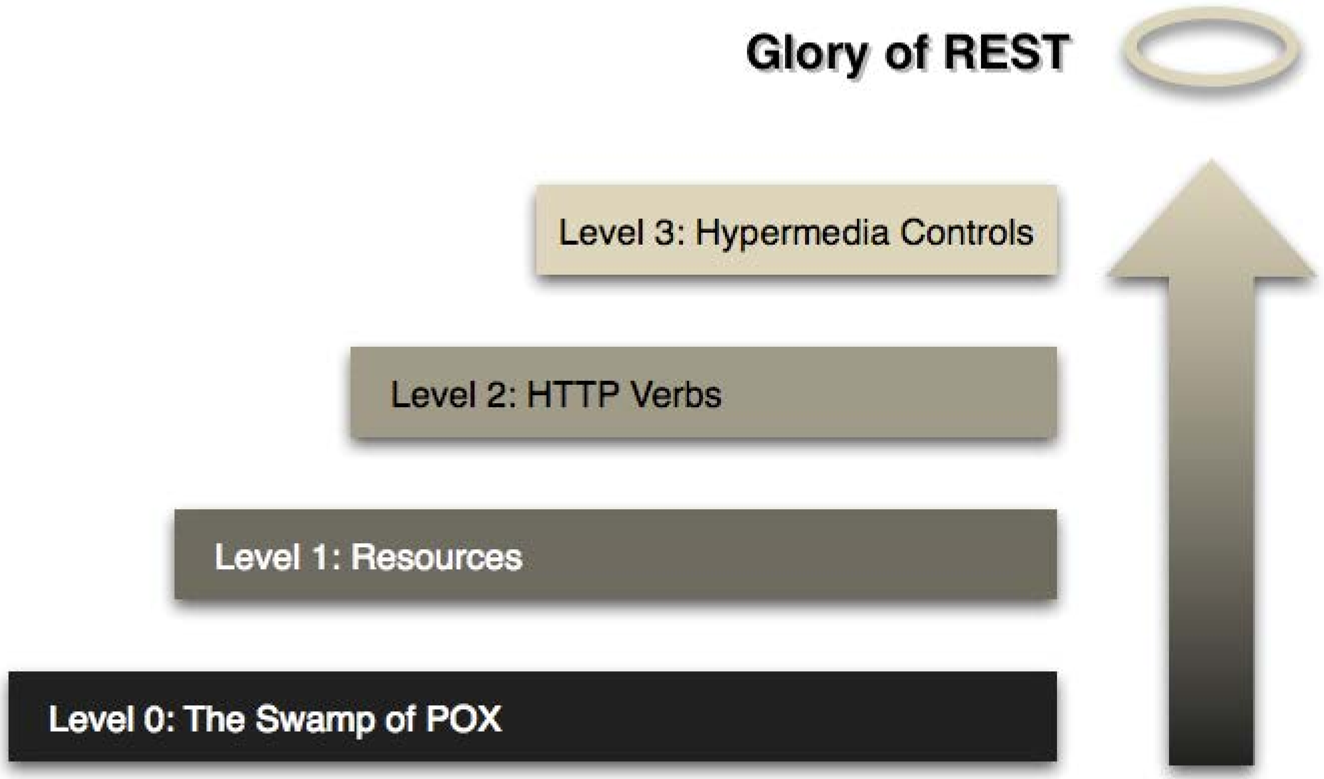
Remote Procedure Call: SOAP

Simple Object Access Protocol

Call method using standardized XML

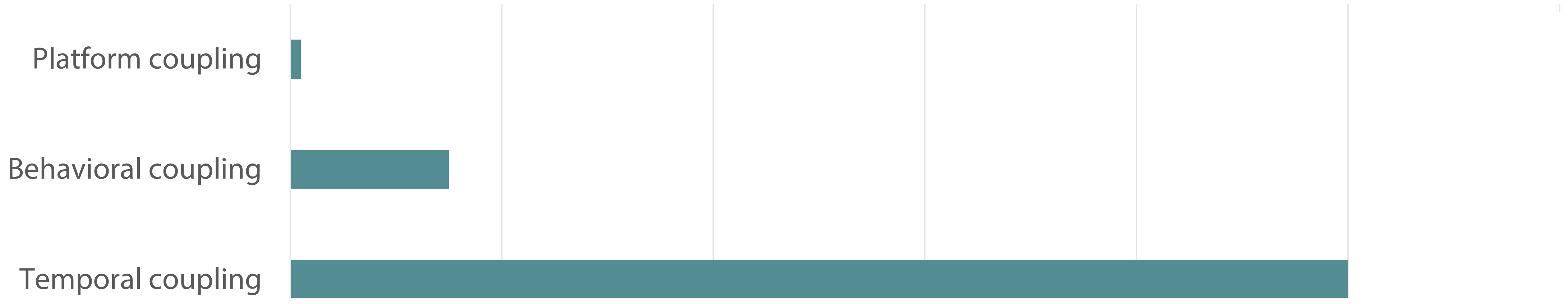


Representational State Transfer (REST)



Picture by Martin Fowler

Representational State Transfer



- Fallacies more apparent than with RPC

Demo: Simplified SOA

FIRE ON WHEELS

Business is great

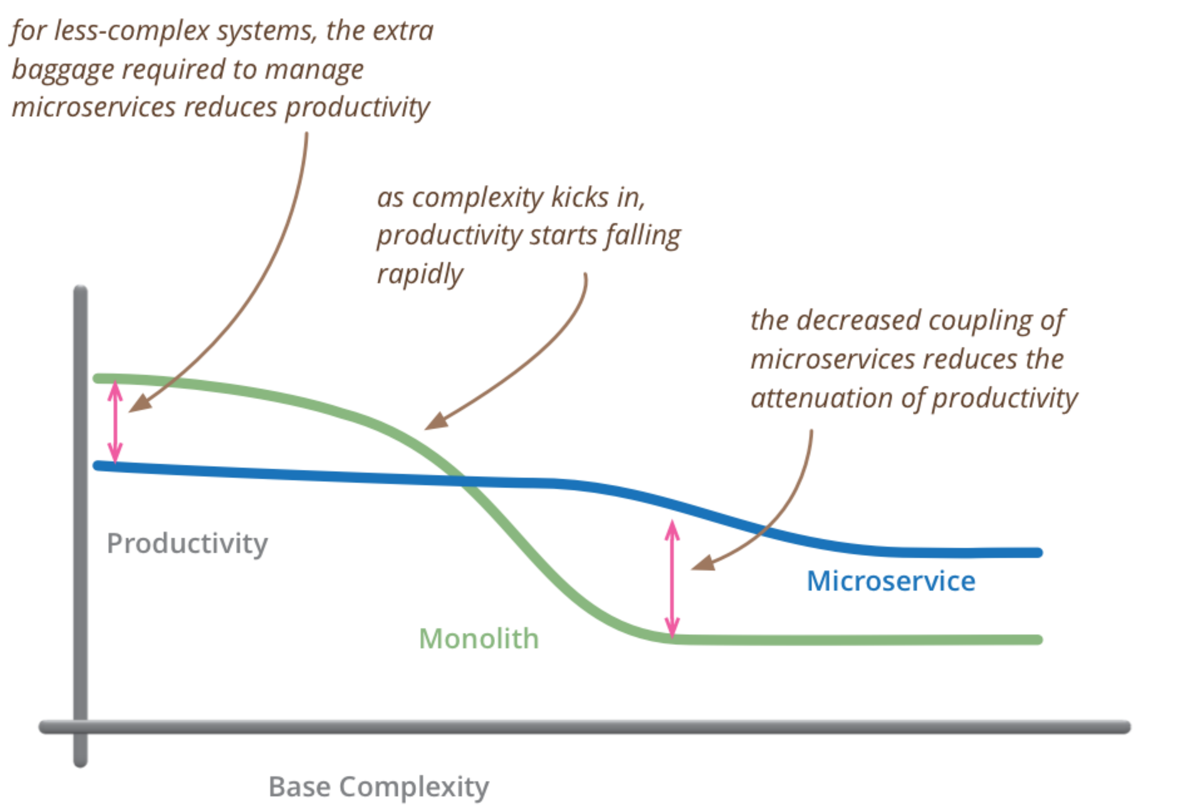
Amazon wants Fire On Wheels to deliver packages for them

But they don't want to fill out a web page!

Microservices

Microservices is a software architecture style, in which complex applications are composed of small, autonomous processes communicating with each other using language-agnostic APIs.

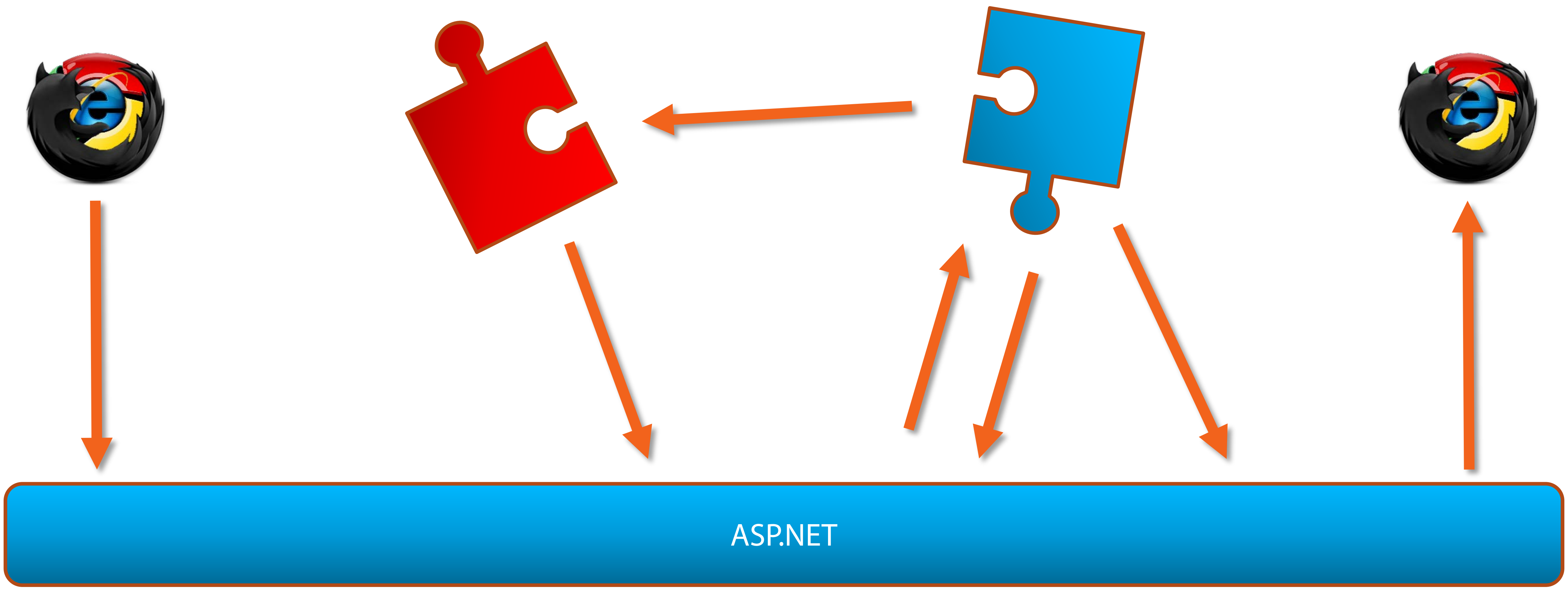
Should You?



but remember the skill of the team will outweigh any monolith/microservice choice

Picture by Martin Fowler

Microservices with Messaging



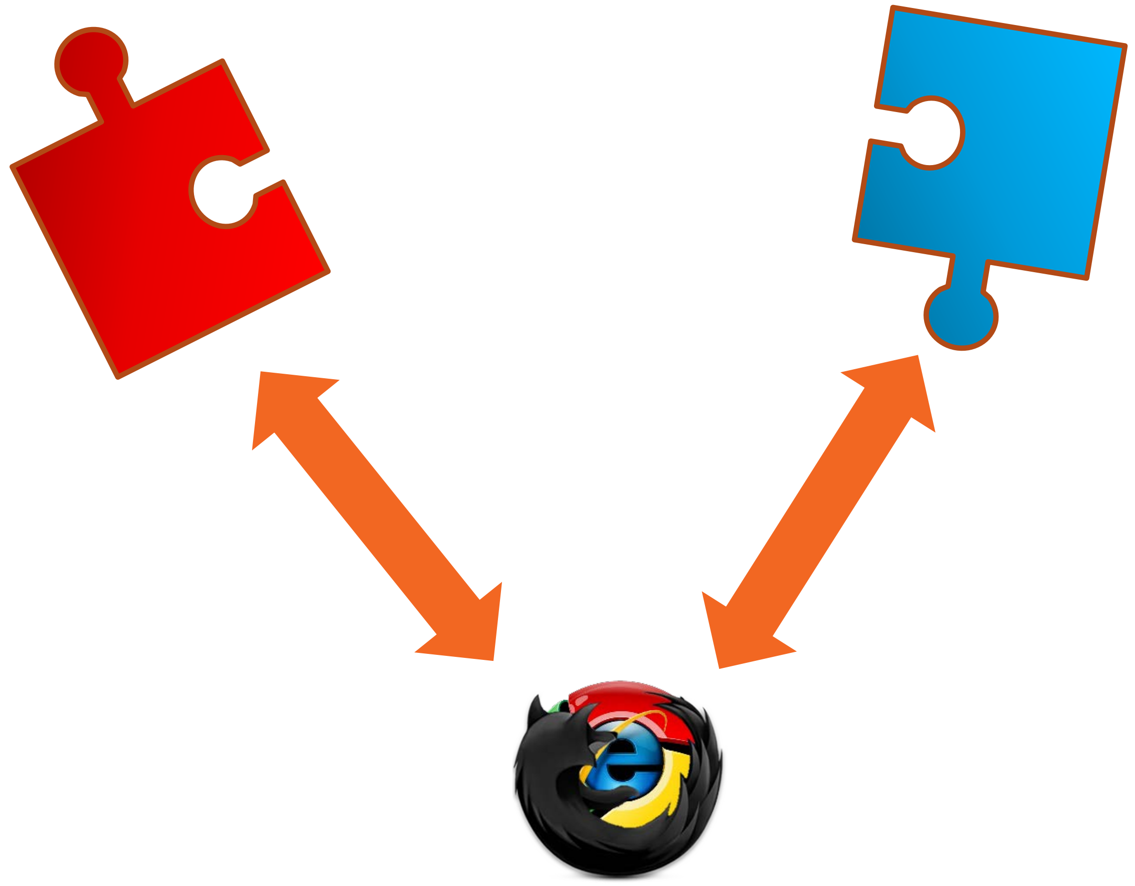
Microservices and UI

Each service should expose its own UI

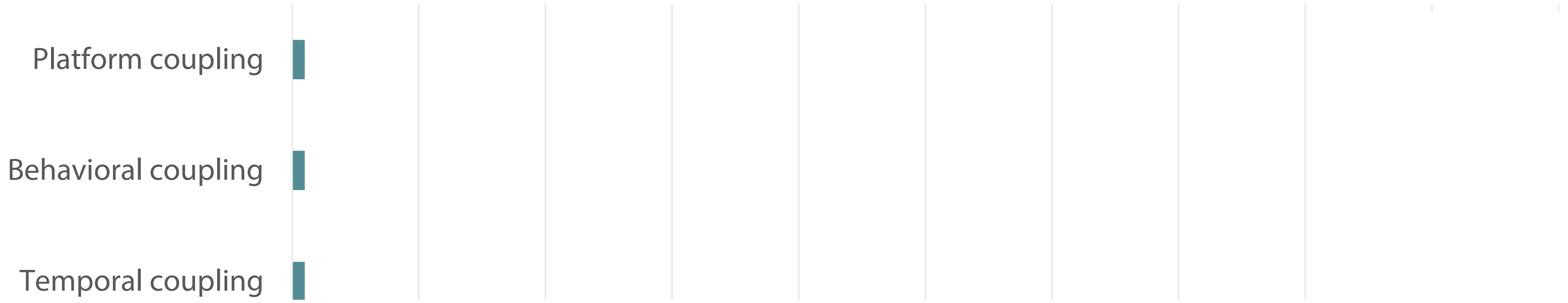
When one service fails the rest of the UI is shown

Needs UI composition

Microservices and SPAs



Microservices



- Fallacies still apply!

Properties of Microservices (1)

- Maintainable
- Versioning
- Each their own
- Hosting
- Failure isolation
- Observable

Properties of Microservices (2)

UI

Discovery

Security

Deployment

The downside of the microservice architecture is that it’s relatively complex

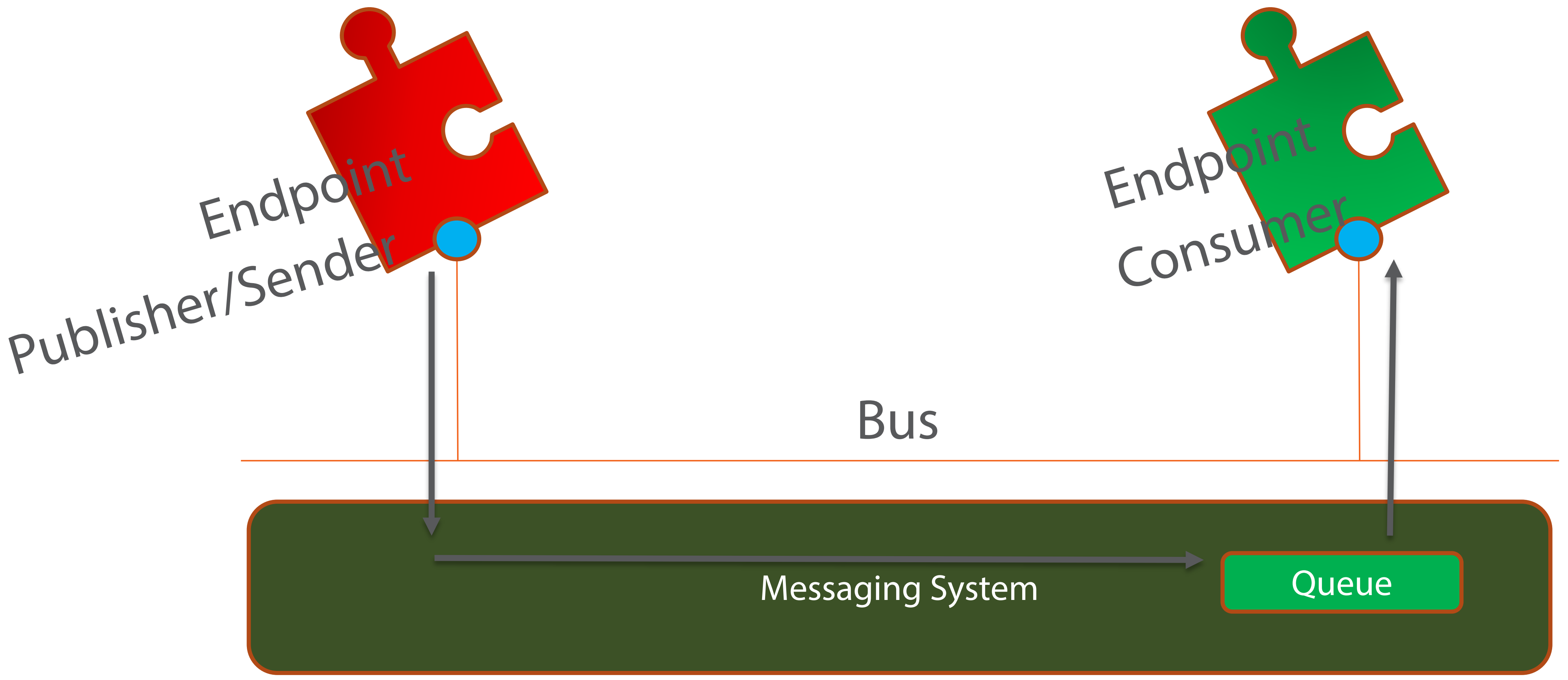
Reuse and Microservices

- Code is typically not shared among services
- But... How about DRY!? (Don't Repeat Yourself)
- Set yourself free from DRY, use source control

Databases and Microservices

- An entity can exist in multiple services and databases
- Data duplication is fine
- But... How about foreign keys?
- Set yourself free from data integrity

The Services



NServiceBus

- An implementation of a service bus
- Written for .NET services
- Non- .NET services can use other bus implementations compatible with the same messaging system.

ESB

- Example: Biztalk
- Don’t confuse with our service bus
- Logic in the bus
- Microservices = dumb pipes

Demos?

FIRE ON WHEELS

More technical explanation needed

In the next module!

Summary

Monoliths are great but become unmaintainable when they get more complex

SOA with RPC and REST has some distinctive downsides

Microservices solve these downsides but beware of the complexity

The service bus facilitates the communication between services through a messaging backend