# Service Oriented Architecture (Micro-)services & Integration

Sébastien Mosser

Lecture #0, 17.09.2018

"Anyone who claims that **integration is easy** must be incredibly **smart**, incredibly **ignorant** ...

… or **have a financial interest** in making you believe that **integration is easy**

**- Gregor Hophe [EIP]**

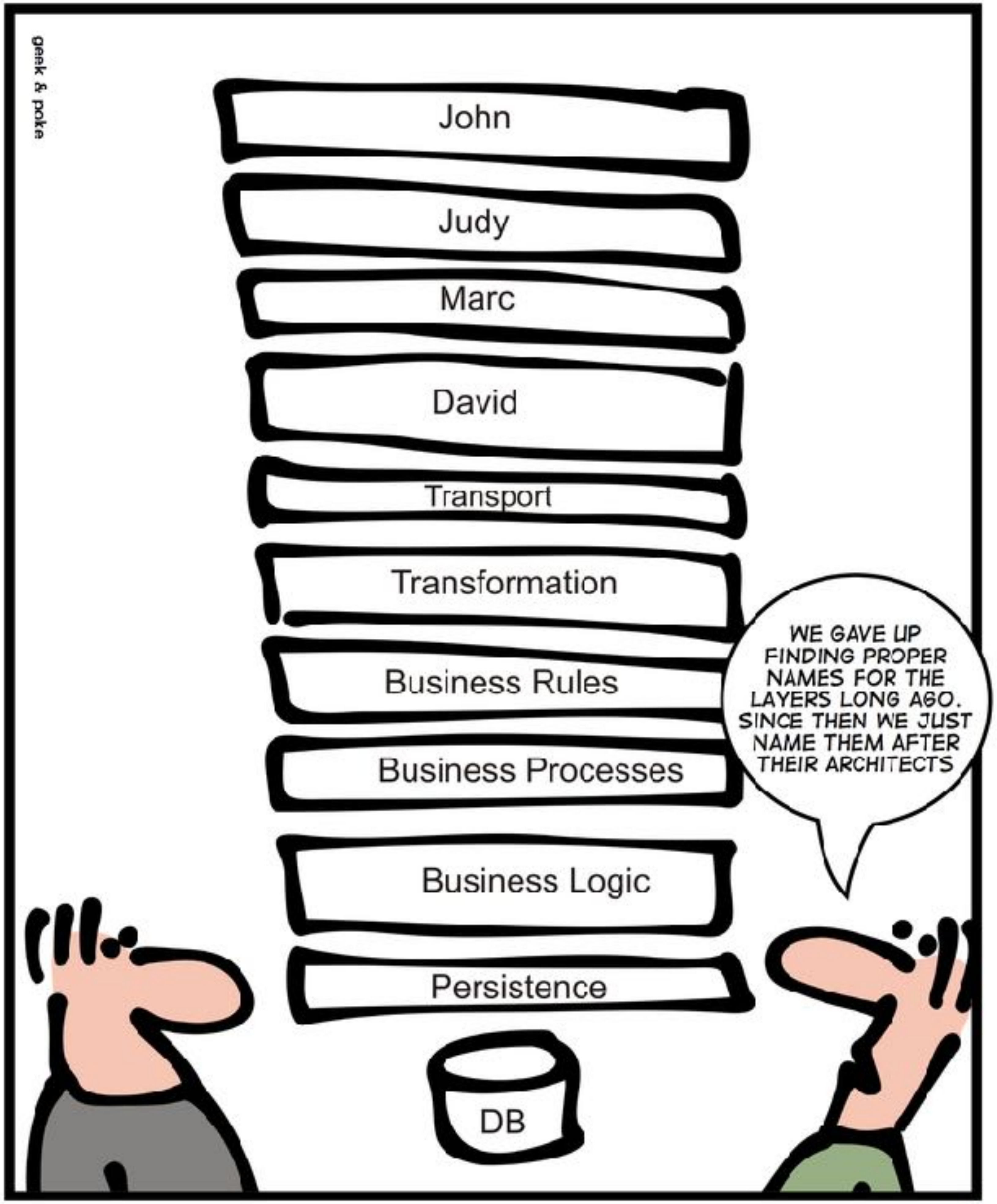
Integration is **not** about **designing beautiful** systems.

It is about **dealing with** **existing** and **crappy** ones.

**existing**

and

**crappy**



A GOOD ARCHITECT LEAVES A FOOTPRINT

http://geek-and-poke.com/geekandpoke/2013/7/13/foodprints

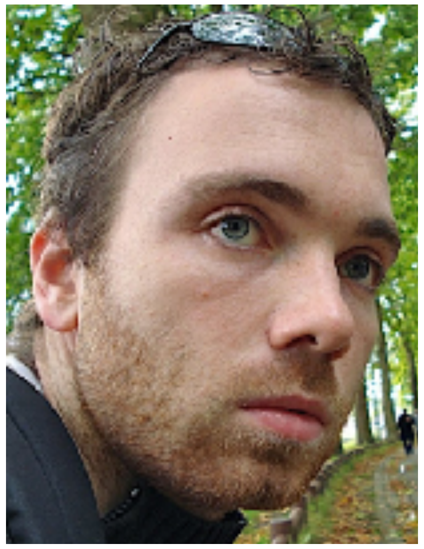
# 1 Course Organization

# Contract

- **You:**
  - **No computer** during lectures
  - **Be on time**
  - Project **involvement**
  - **Prepare** when asked for
- **Staff:**
  - **One-week latency** feedback
  - **Availability** (slack, meeting)

## Teaching Staff

Sébastien
**Mosser**
(former SAR)



Loic
**Gaillard**
(former AL)

Jean-Yves
**Delmotte**
(former HCI)

Main contact: mosser@i3s.unice.fr

DAAAAAD, I lost one of my microServices I can't play without it.
I told you he is not mature enough for it. We should have bought him a monolith.
I believe you.

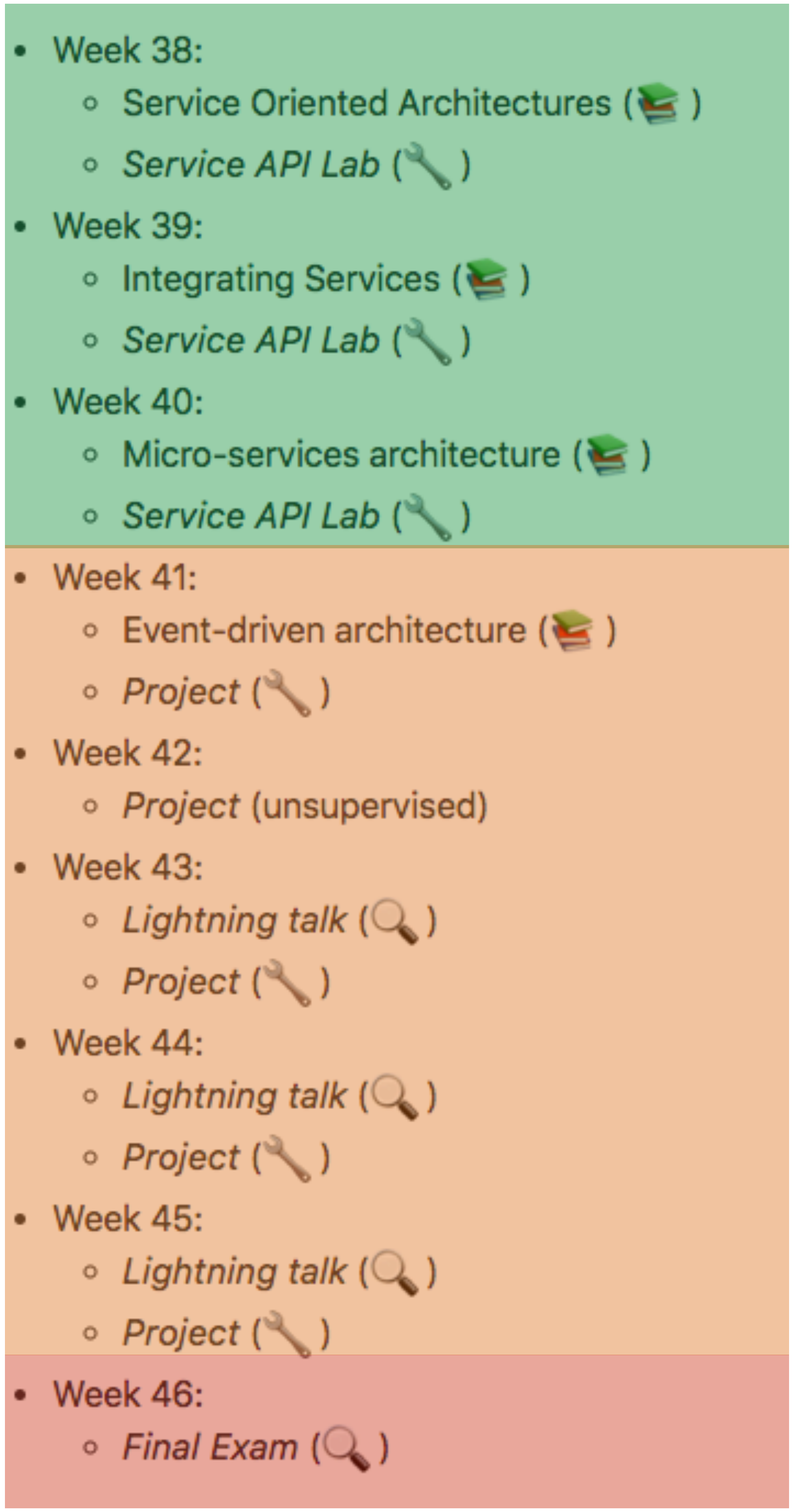
DANIEL STORI {TURNOFF.US}

# Agenda 2018

**Service API Design**

**Microservice Project**

**Final Exam**



- Week 38:
  - Service Oriented Architectures (📚)
  - *Service API Lab* (🔧)
- Week 39:
  - Integrating Services (📚)
  - *Service API Lab* (🔧)
- Week 40:
  - Micro-services architecture (📚)
  - *Service API Lab* (🔧)
- Week 41:
  - Event-driven architecture (📚)
  - *Project* (🔧)
- Week 42:
  - *Project* (unsupervised)
- Week 43:
  - *Lightning talk* (🔍)
  - *Project* (🔧)
- Week 44:
  - *Lightning talk* (🔍)
  - *Project* (🔧)
- Week 45:
  - *Lightning talk* (🔍)
  - *Project* (🔧)
- Week 46:
  - *Final Exam* (🔍)

## Communication: **No emails**!

**https://github.com/mosser/microservices/**

slack

# Technological Environment

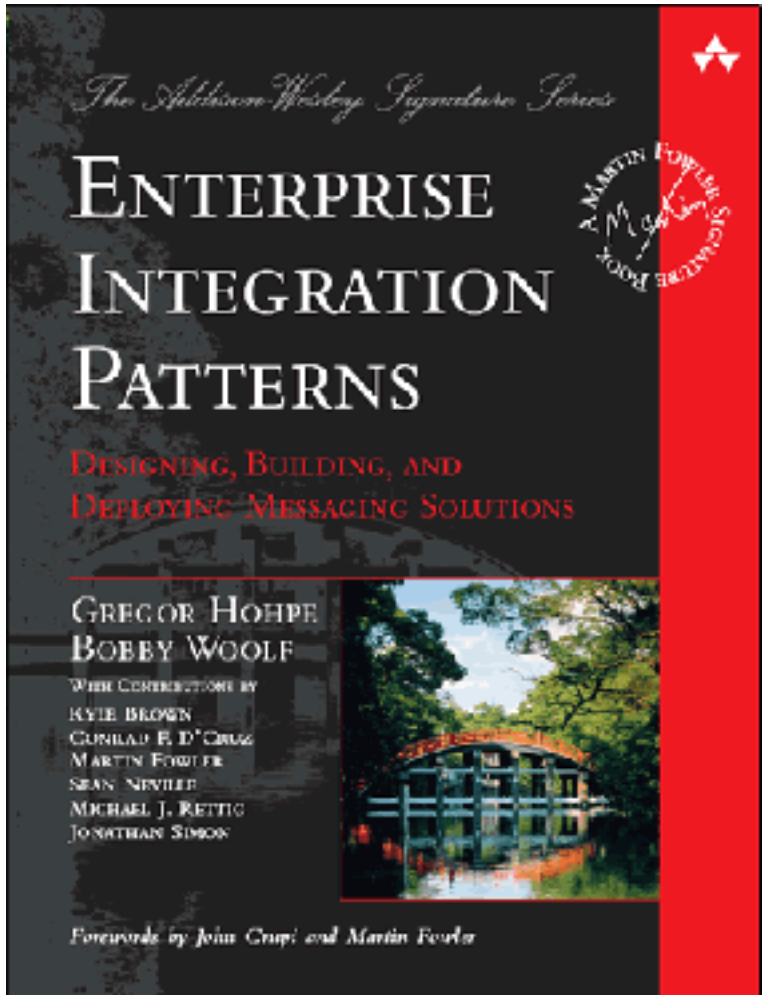
Gatling

LOAD TESTING

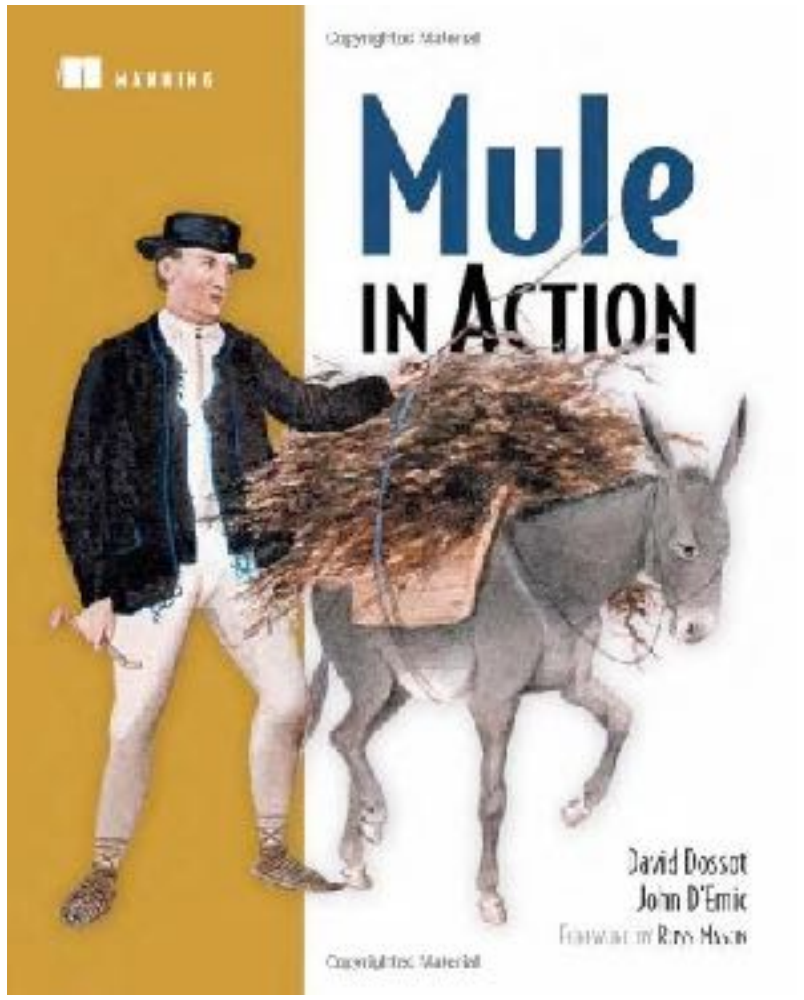
kafka

+ Your favorite Stack

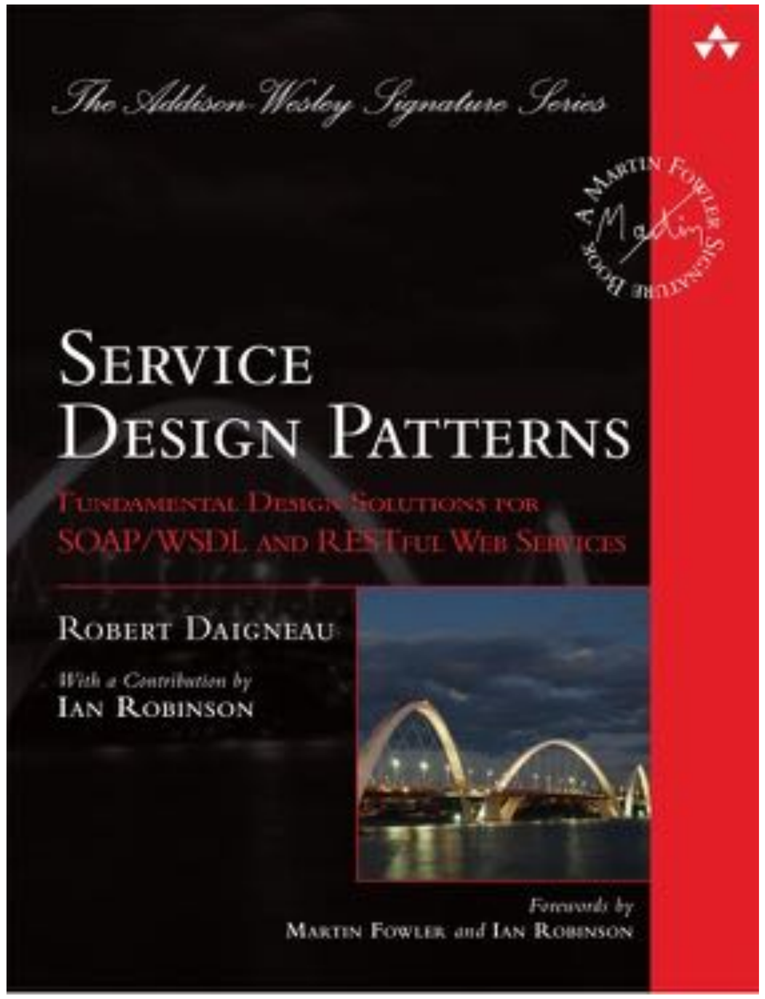
# Bibliography: Service & Integration


The Addison-Wesley Signature Series
ENTERPRISE INTEGRATION PATTERNS
DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS
GREGOR HOHPE
BOBBY WOOLF
Forewords by John Crupi and Martin Fowler

[EIP]

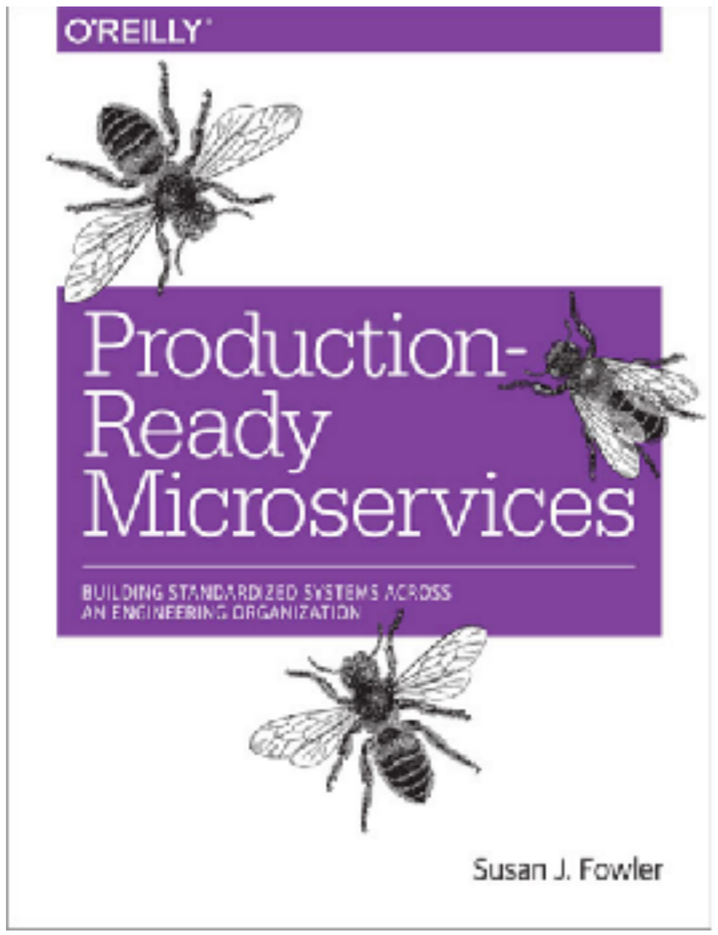


[MiA]


The Addison-Wesley Signature Series
SERVICE DESIGN PATTERNS
FUNDAMENTAL DESIGN SOLUTIONS FOR SOAP/WSDL AND RESTFUL WEB SERVICES
ROBERT DAIGNEAU
With a Contribution by IAN ROBINSON
Forewords by MARTIN FOWLER and IAN ROBINSON

[SDP]

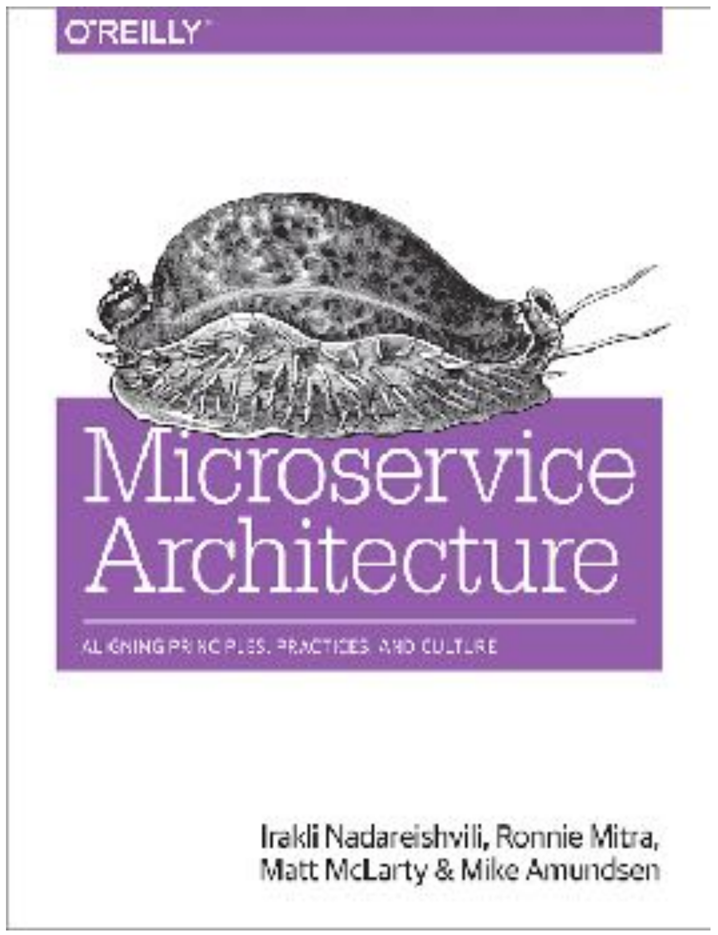
# Microservices bibliography


O'REILLY

Production-Ready Microservices

BUILDING STANDARDIZED SYSTEMS ACROSS AN ENGINEERING ORGANIZATION

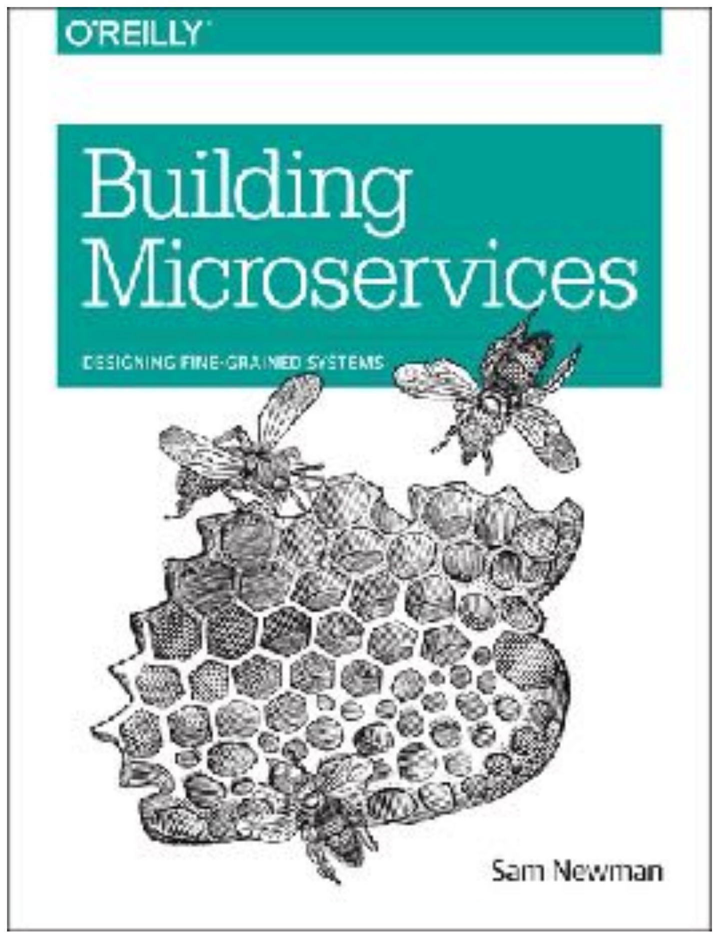
Susan J. Fowler


O'REILLY

Microservice Architecture

ALIGNING PRINCIPLES, PRACTICES, AND CULTURE

Irakli Nadareishvili, Ronnie Mitra, Matt McLarty & Mike Amundsen

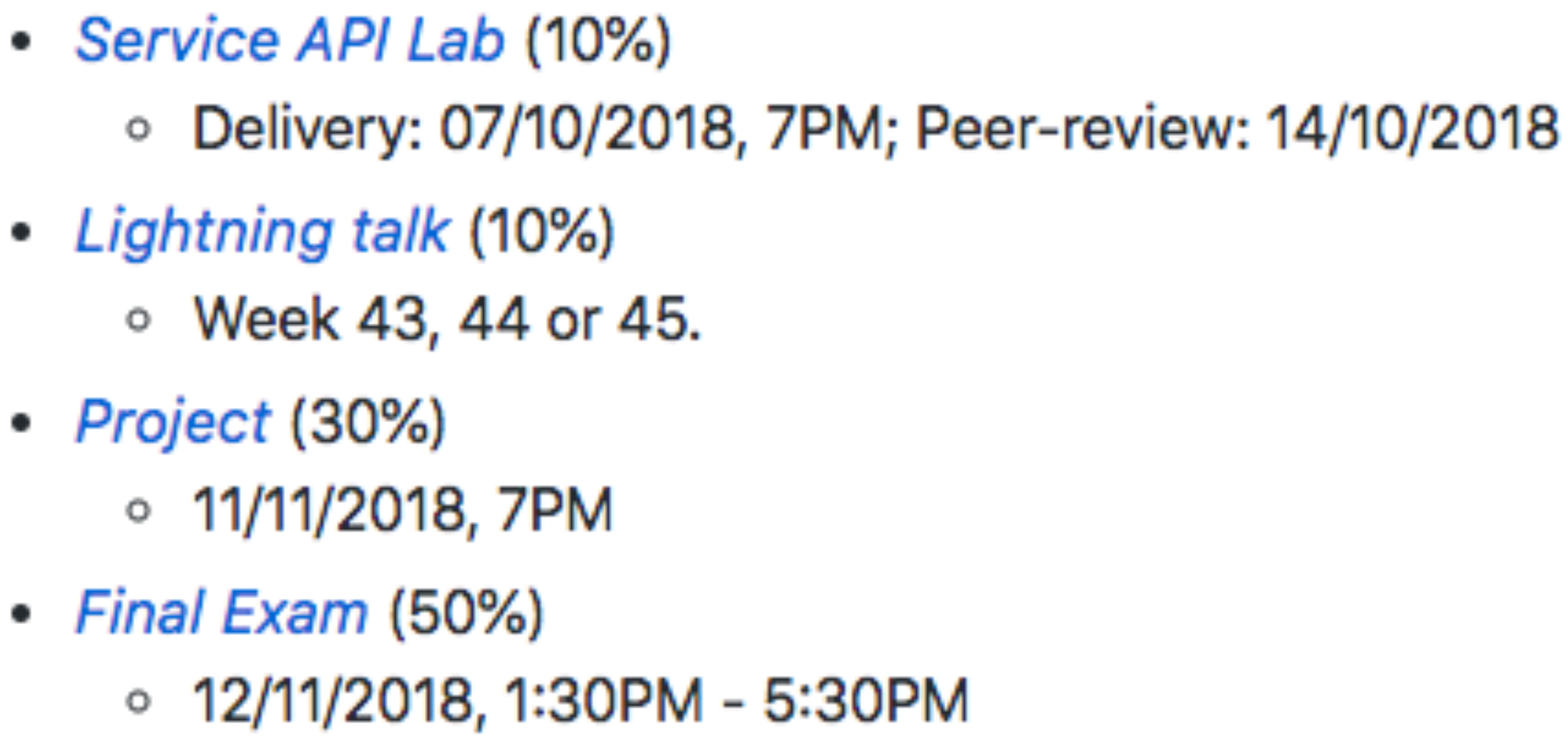

O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman

## Evaluation



- *Service API Lab* (10%)
  - Delivery: 07/10/2018, 7PM; Peer-review: 14/10/2018
- *Lightning talk* (10%)
  - Week 43, 44 or 45.
- *Project* (30%)
  - 11/11/2018, 7PM
- *Final Exam* (50%)
  - 12/11/2018, 1:30PM - 5:30PM

# Integration 101

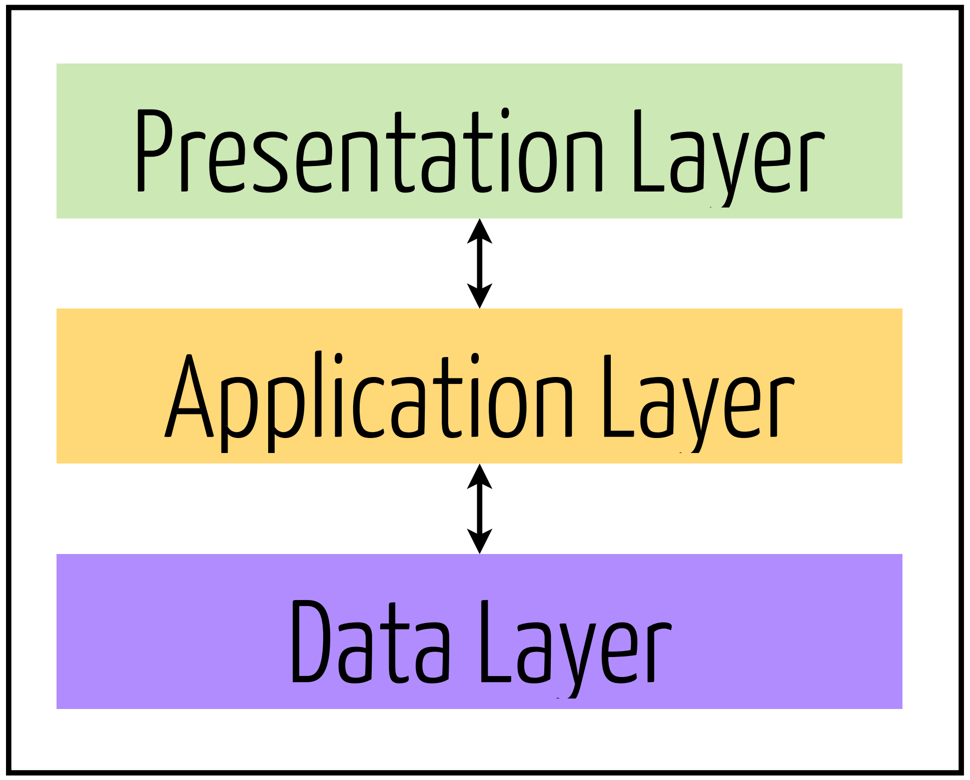
2

"

Interesting applications rarely live in isolation

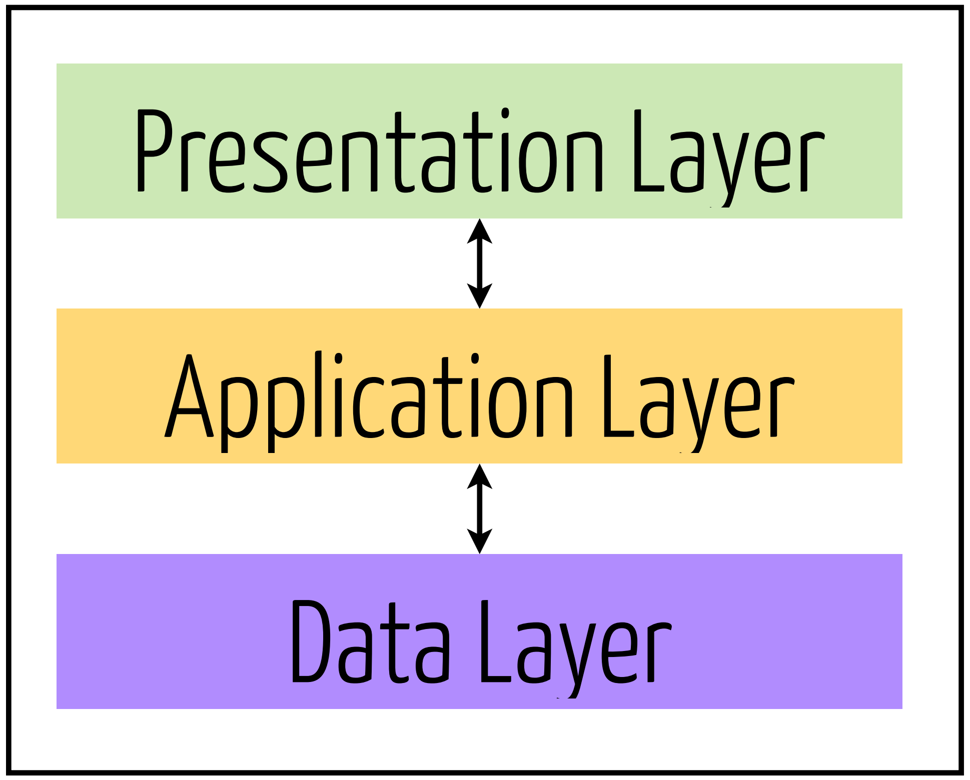
[EIP]

## Reminder: n-tiers architectures, where n=3



Application

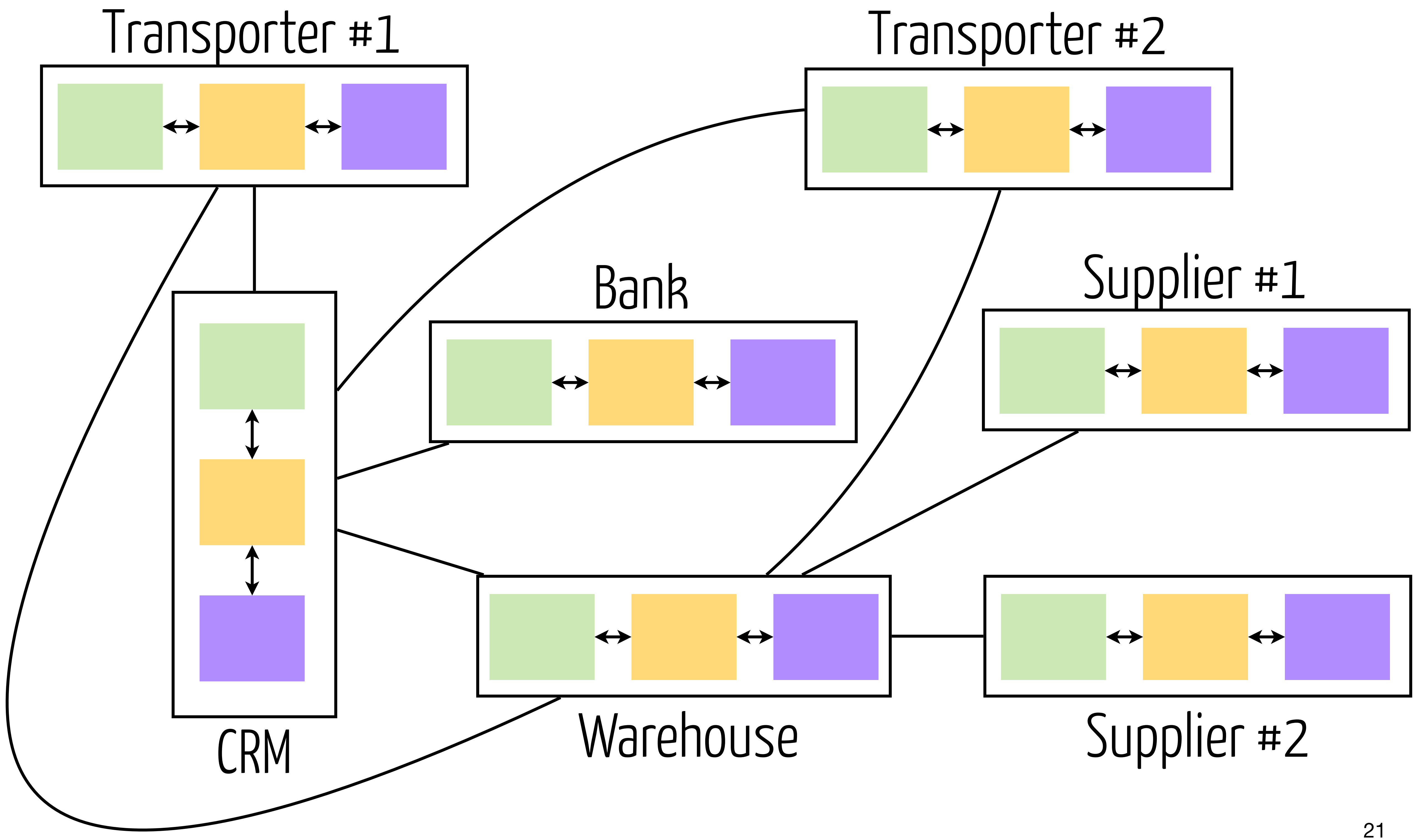
## Reminder: n-tiers architectures, where n=3



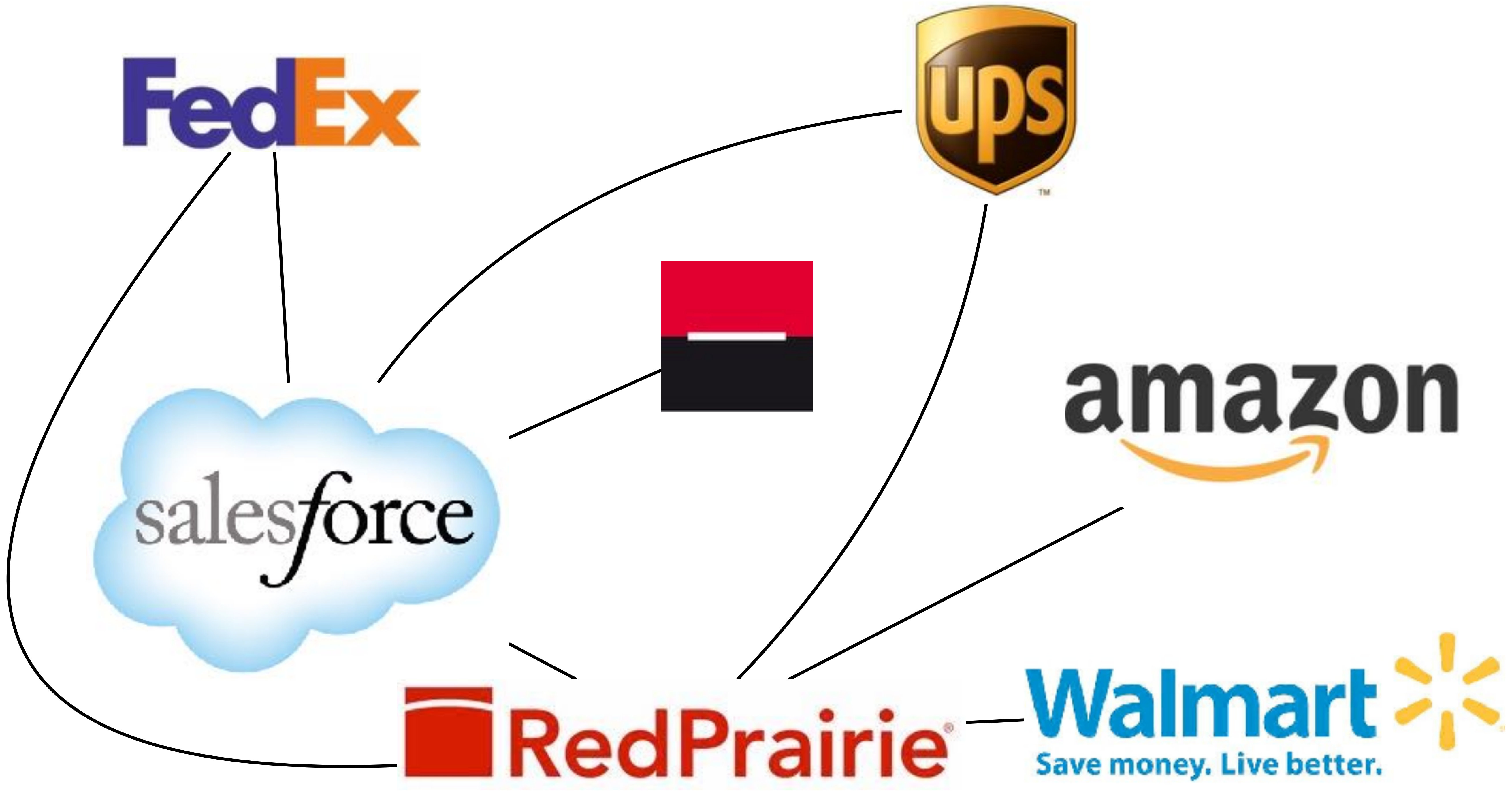
Application

**This is not interesting!**

# This is interesting!



# This is **even more** interesting



## Challenges for Integration

[EIP]

## Challenges for Integration

[EIP]

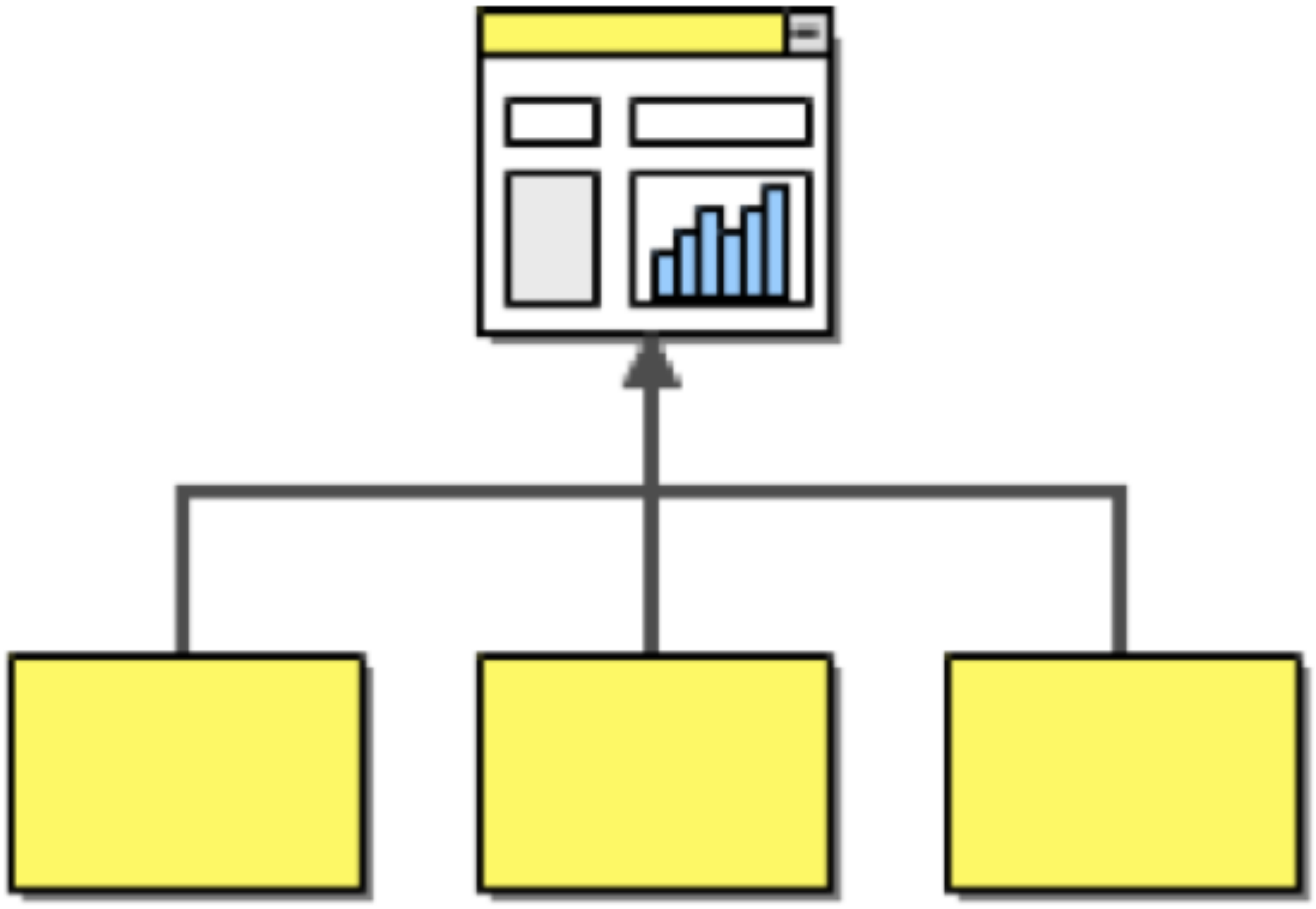
**Networks** are **unreliable**

**Networks** are **slow**

**Change** is **inevitable**

Any **two applications** are **different**

## Practical Integration

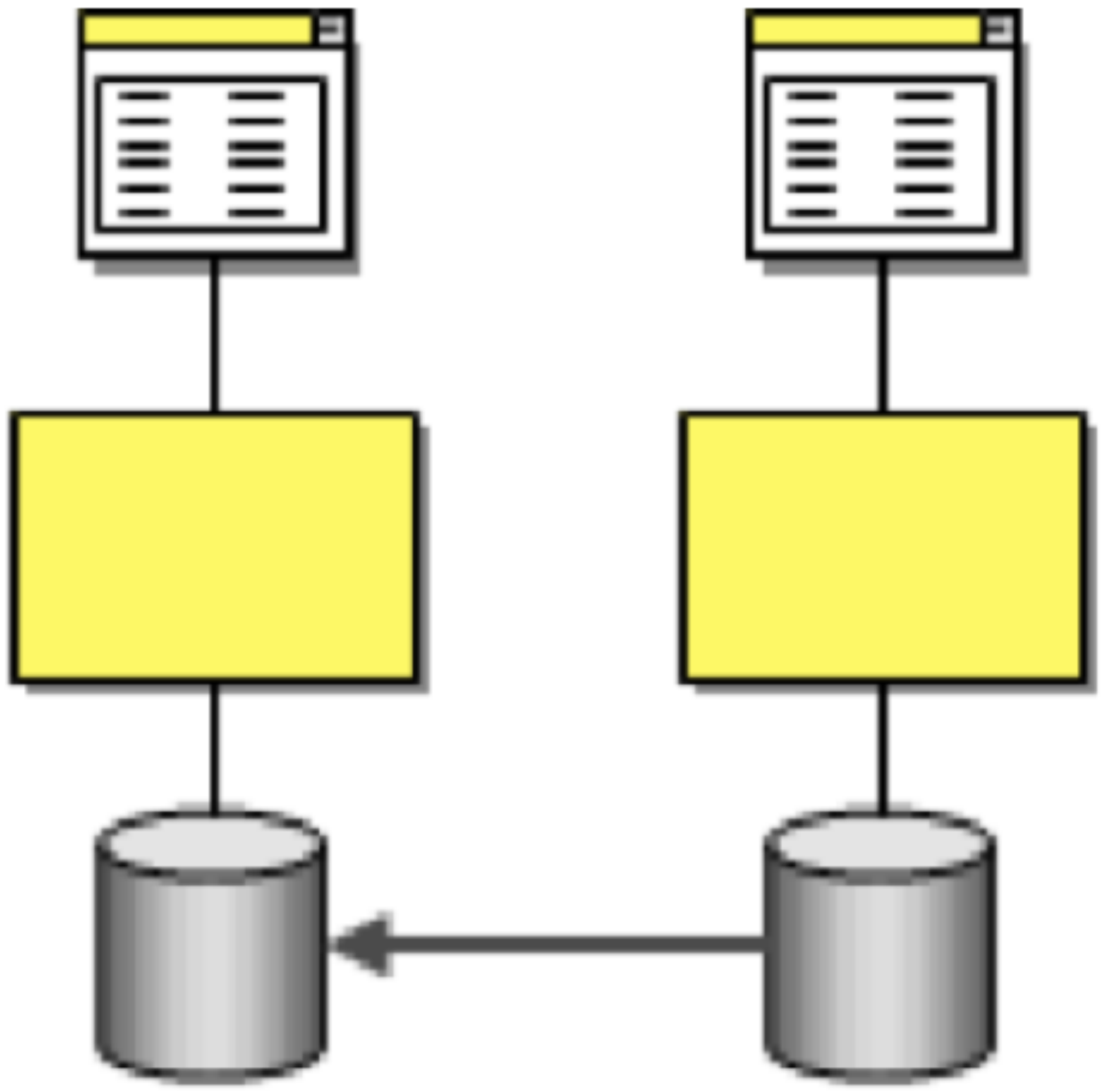


[EIP]

Information Portal

LIFERAY®
Enterprise. Open Source. For Life.

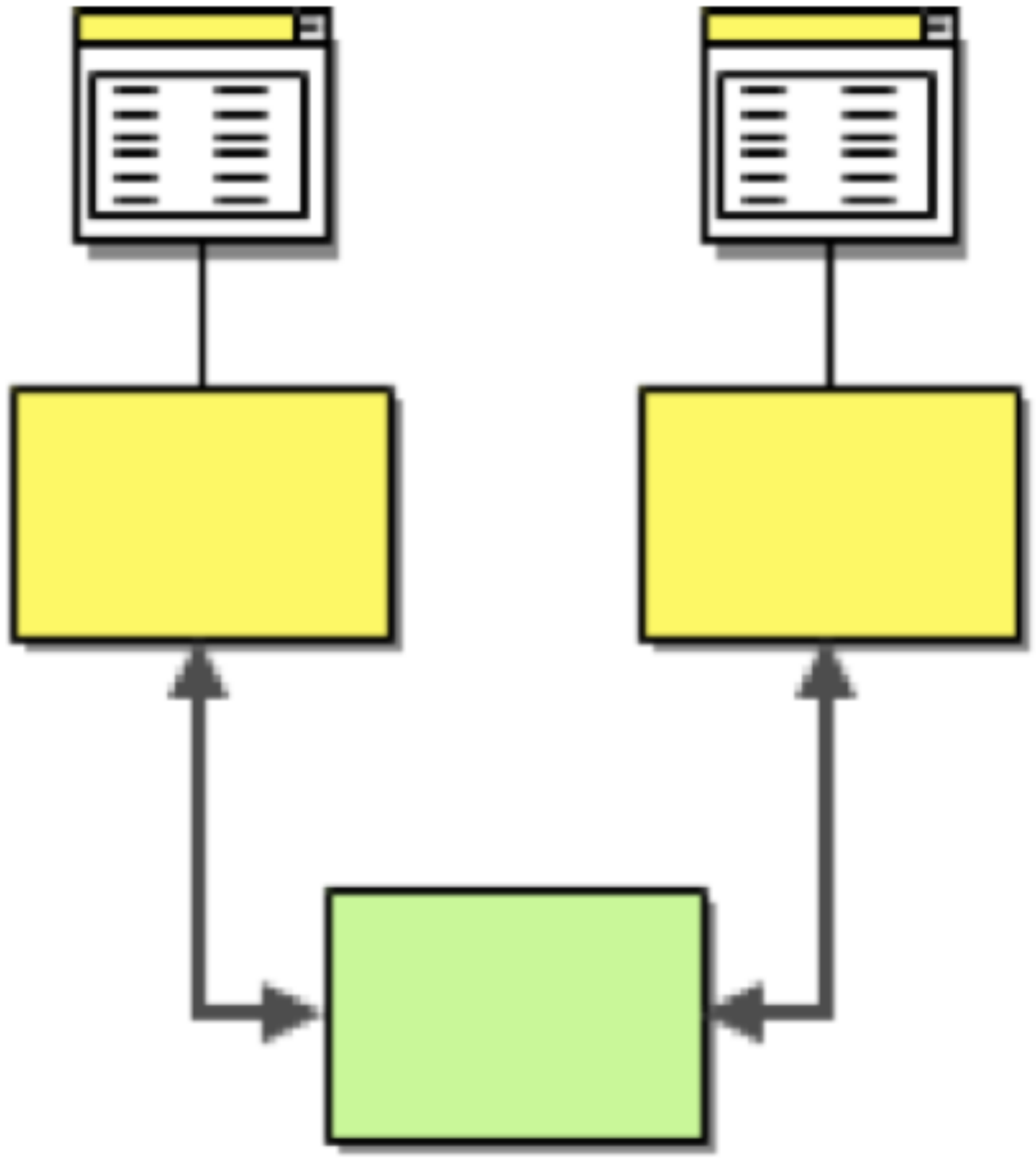
## Practical Integration



Data
Replication

[EIP]

## Practical Integration

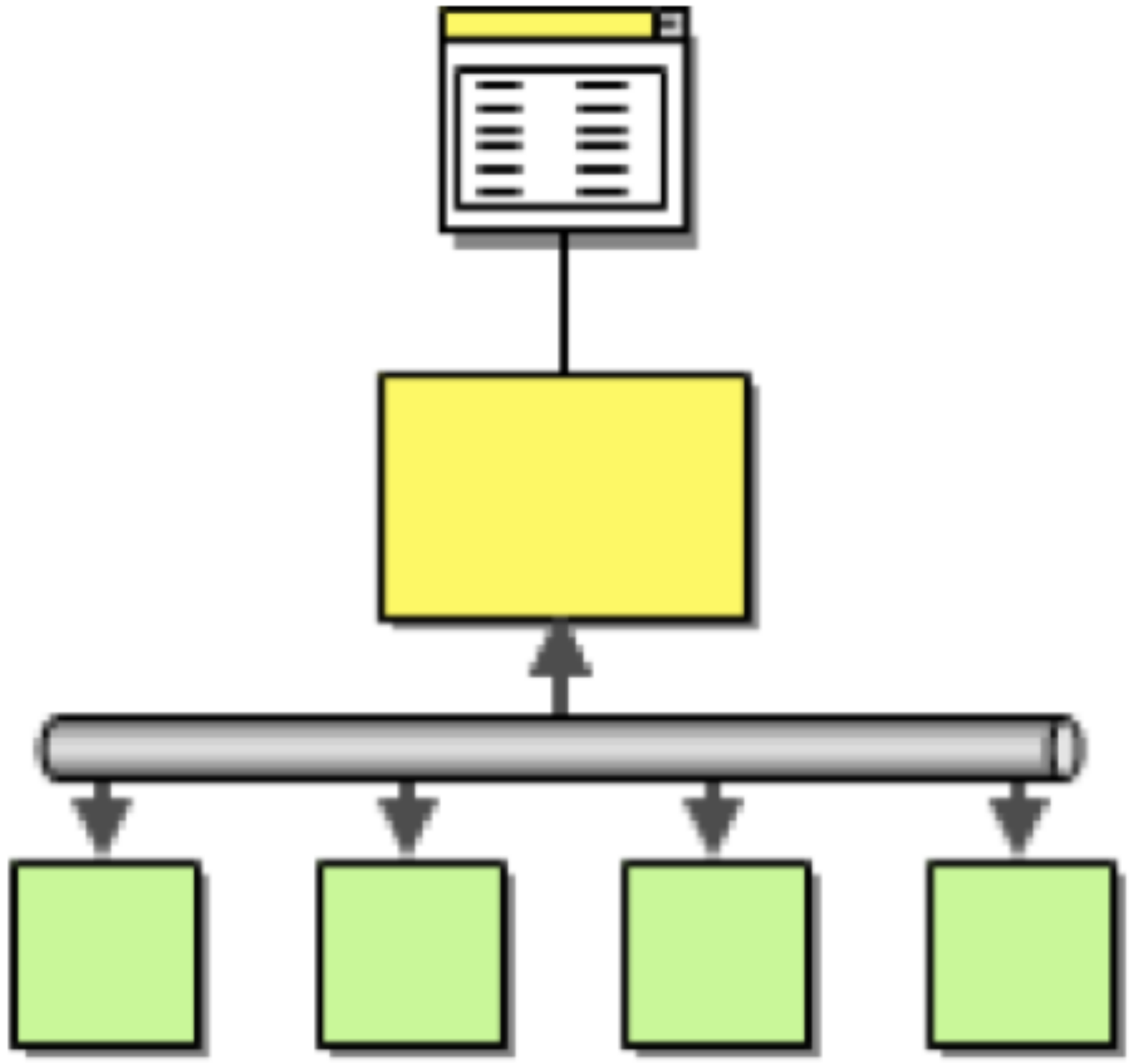


[EIP]

Shared Business Functions

(Remote Procedure Call)

# Practical Integration



[EIP]

Service
Oriented
Architecture

(Messaging)

# Loose Coupling

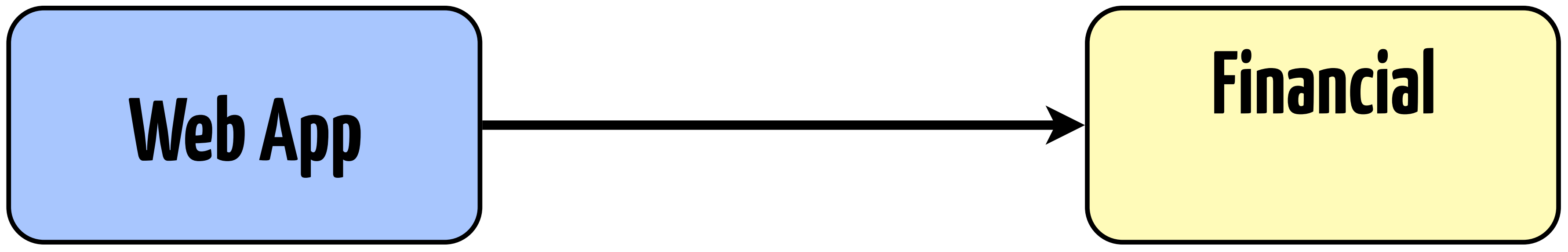
3

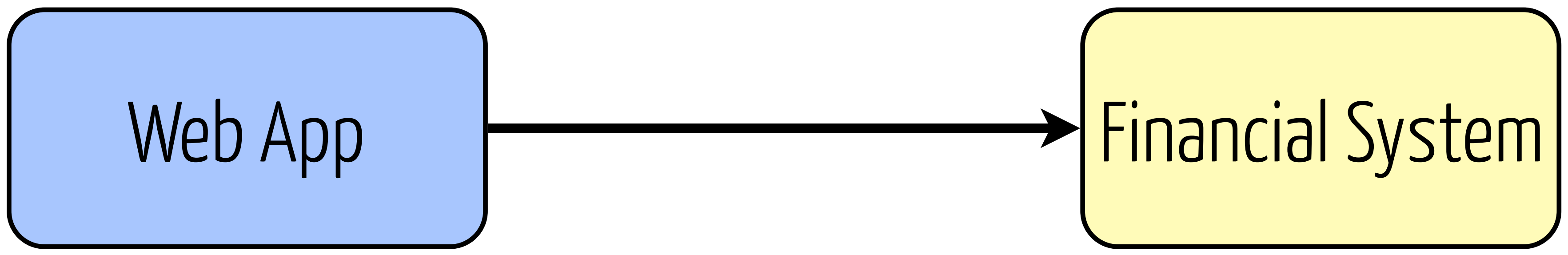
## Coupling?

"

**Assumptions** two parties make about **each others** when they **exchange** information

[EIP]


Web App
Financial



```
String hostName = "server.bank.com";
int port = 80;

IPHostEntry hostInfo = Dns.GetHostByName(hostName);
IPAddress address = hostInfo.AddressList[0];
IPEndPoint endpoint = new IPEndPoint(address, port);

Socket socket = new Socket(address.AddressFamily, SocketType.Stream,
                ProtocolType.Tcp);
socket.Connect(endpoint);

byte[] amount = BitConverter.GetBytes(1000);
byte[]name =Encoding.ASCII.GetBytes("Joe");

int bytesSent =  socket.Send(amount);
bytesSent     += socket.Send(name);

socket.Close();
```

## Coupling Measurement

```
String hostName = "server.bank.com";
int port = 80;

IPHostEntry hostInfo = Dns.GetHostByName(hostName);
IPAddress address = hostInfo.AddressList[0];
IPEndPoint endpoint = new IPEndPoint(address, port);

Socket socket = new Socket(address.AddressFamily, SocketType.Stream,
                ProtocolType.Tcp);
socket.Connect(endpoint);

byte[] amount = BitConverter.GetBytes(1000);
byte[]name =Encoding.ASCII.GetBytes("Joe");

int bytesSent =  socket.Send(amount);
bytesSent     += socket.Send(name);

socket.Close();
```

## Coupling Measurement

Location

```
String hostName = "server.bank.com";
int port = 80;

IPHostEntry hostInfo = Dns.GetHostByName(hostName);
IPAddress address = hostInfo.AddressList[0];
IPEndPoint endpoint = new IPEndPoint(address, port);

Socket socket = new Socket(address.AddressFamily, SocketType.Stream,
                ProtocolType.Tcp);
socket.Connect(endpoint);

byte[] amount = BitConverter.GetBytes(1000);
byte[]name =Encoding.ASCII.GetBytes("Joe");

int bytesSent =  socket.Send(amount);
bytesSent     += socket.Send(name);

socket.Close();
```

Availability

Data Encoding

Communication Protocol

## Data Encoding Assumption

Web App

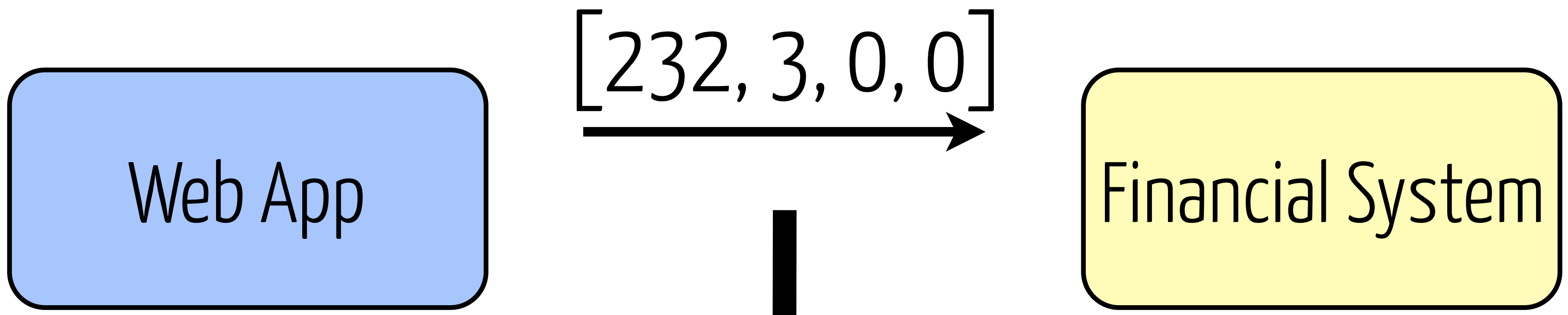
Financial System

Little Endian

$$[232, 3, 0, 0] \equiv 232 + 3 \times 2^8$$

$$= \mathbf{1000}$$

# Data Encoding Assumption

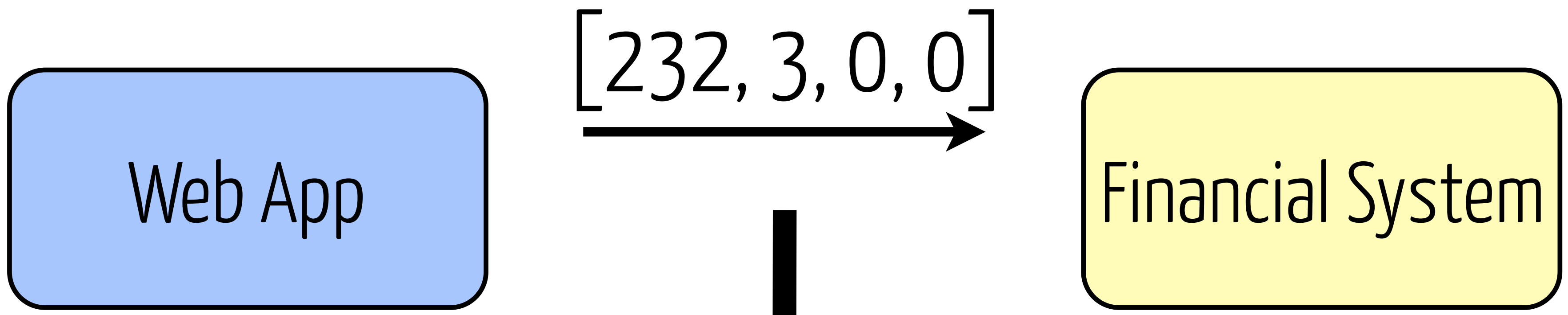


Little Endian

$[232, 3, 0, 0] \equiv 232 + 3 \times 2^8$

$= \mathbf{1000}$

# Data Encoding Assumption



Little Endian

$[232, 3, 0, 0] \equiv 232 + 3 \times 2^{8}$

$= \mathbf{1000}$

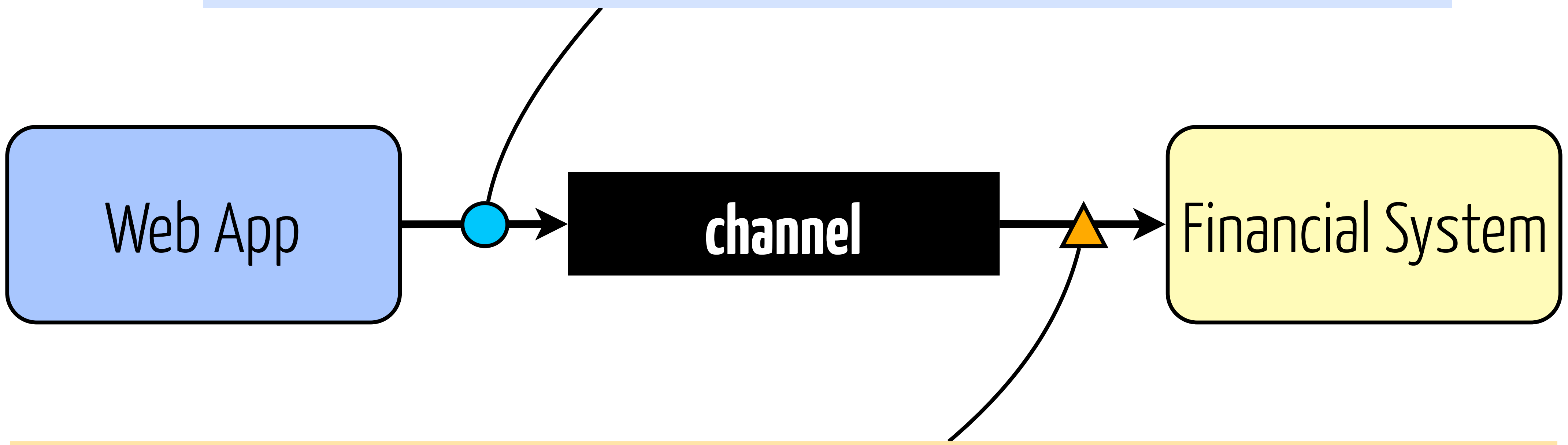
Big Endian

$[232, 3, 0, 0] \equiv 232 \times 2^{24} + 3 \times 2^{16}$

$= \mathbf{3{,}892{,}510{,}720}$

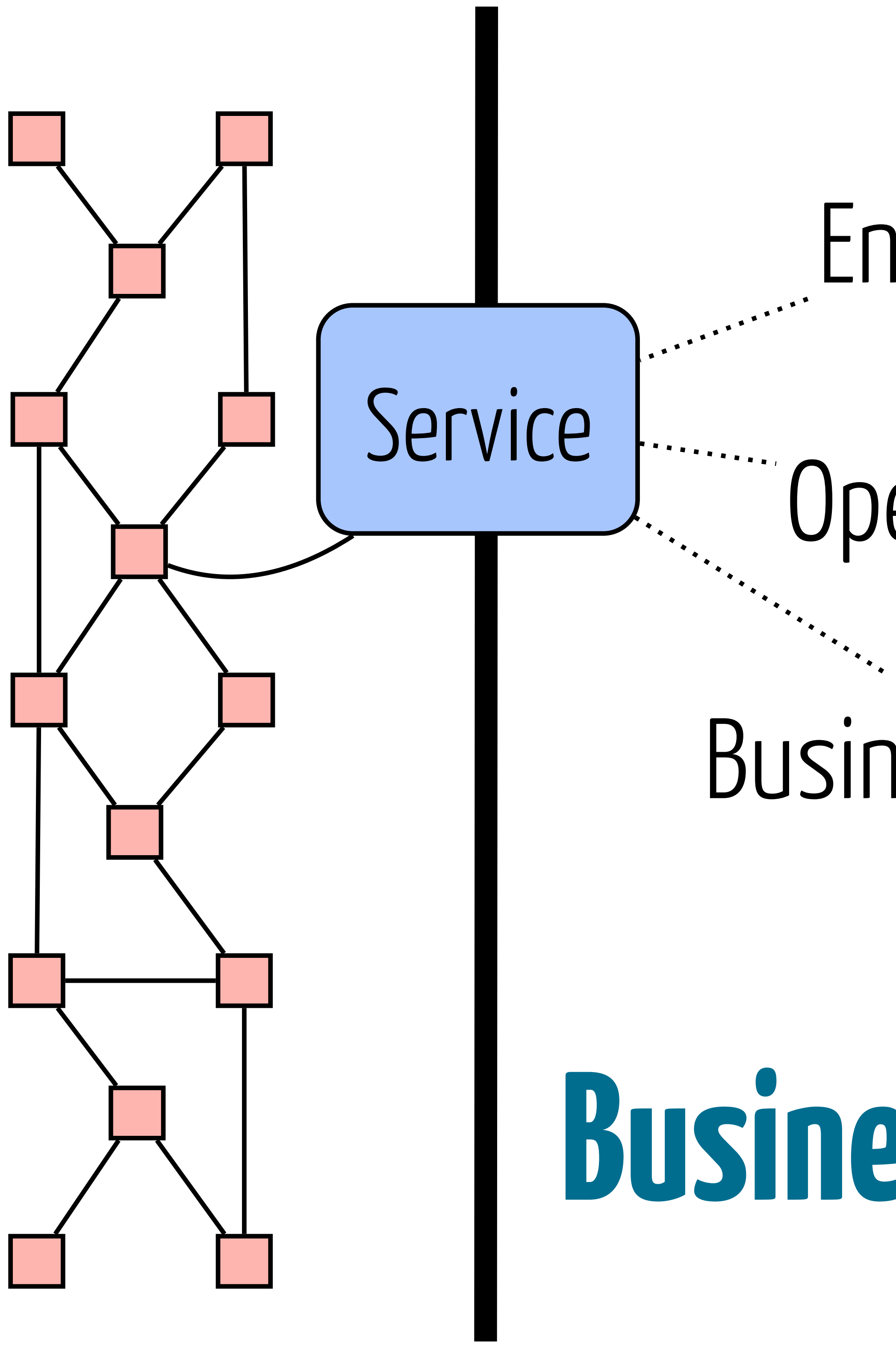
# Goal: Decouple artifacts

```
<deposit>
  <amt>1000</amt>
  <cust>Sebastien</cust>
</deposit>
```



```
{ «kind»: «DEP», «cId»:
«16118325358», «amount»: 1000 }
```

# Service in a nutshell



| | |
|---|---|
| Endpoint | **Where / How** |
| Operations | **Why** |
| Business Objects | **What** |

**Business** vision of the **internal** system

## Endpoint

**Transport**

Protocol (e.g., HTTP)

**Communication**

Protocol (e.g., SOAP)

**Address**

location (e.g., «http://server.bank.com/service»)

## Operations

**Business logic**
encapsulation

**Coarse-grained**
visibility

**Remote** interface

## Business Objects

**Messages**
exchanged

**Business**
data

**Format**
Description

"

**Loose-coupling** provides **important benefits** such as **flexibility** and **scalability**, ...

… but it introduces a more **complex programming model** and can make designing solutions **more difficult**.

[EIP]

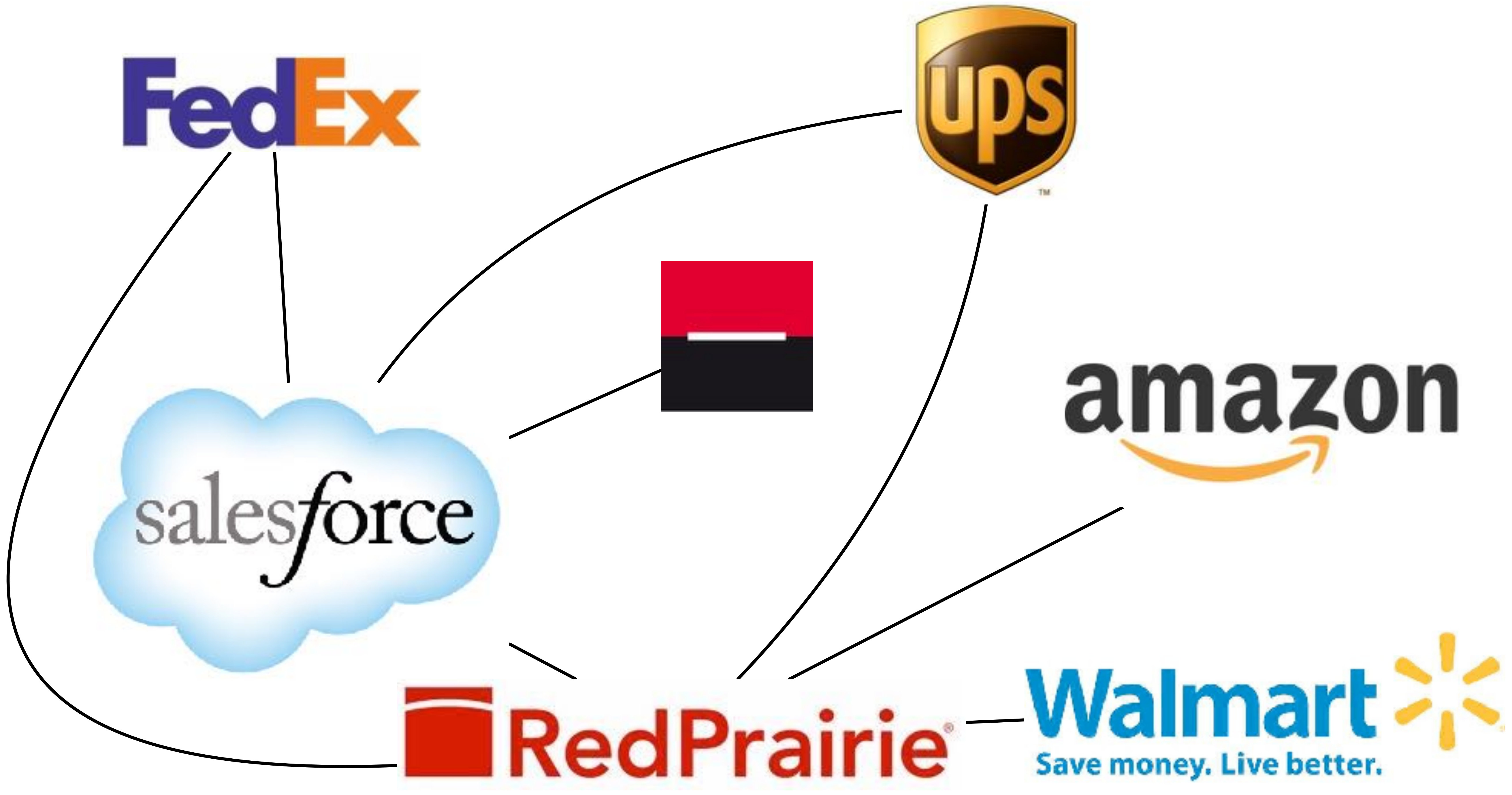
# Sum-up

4

Integration is **not** about **designing beautiful** systems.

It is about **dealing with existing** and **crappy** ones.

# Together, we'll build an empire …



**Loose-coupling** provides **important benefits** such as **flexibility** and **scalability**, but it introduces a more **complex programming model** and can make designing solutions **more difficult**.