3 Common Pitfalls in Microservice Integration (Bonus : And how to avoid them ☺) credit to Bernd Ruecker

Patricio Zambrano

Technical Consultant, Camunda Inc.

Microservices Agenda

- Introduction
- 3 Common Challenges and How to Avoid Them
- Conclusion

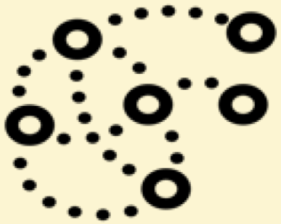
Raise your hand

- REST
- Microservices
- Java

Distributed systems

Distributed systems

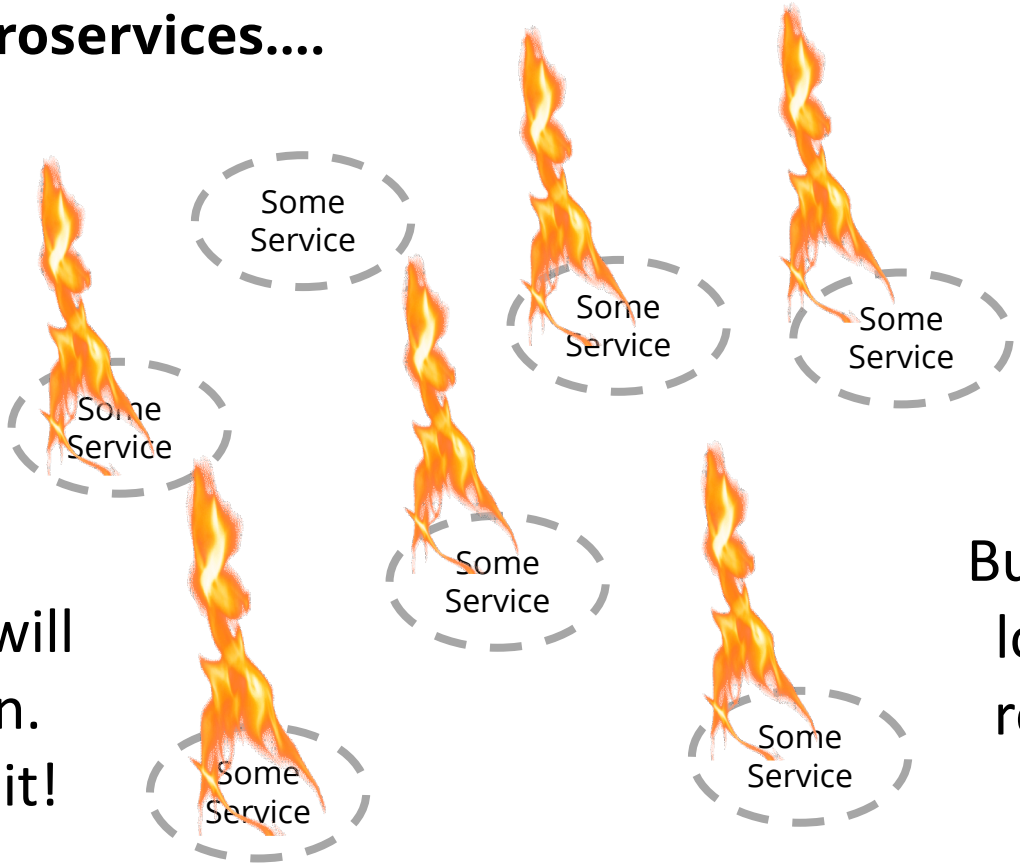
Communication is complex



Challenges of asynchronicity

Distributed Transactions

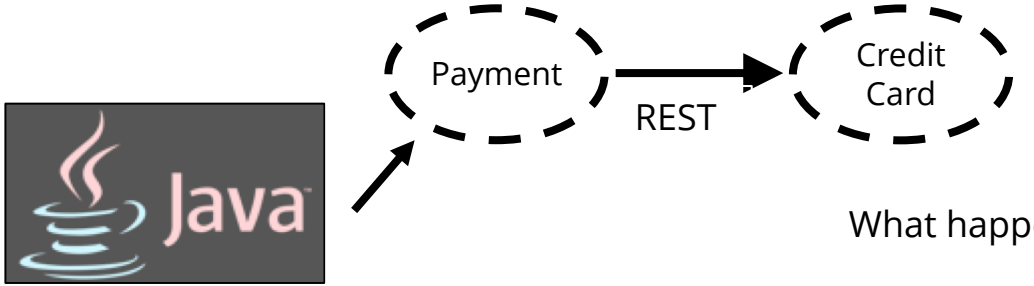
Some Microservices….



Failure will happen. Accept it!

But keep it local! Be resilient.

Let‘s start with a simple example



What happens if the Credit Card Service is Super Slow?

Payment Requestor Application

https://github.com/flowing/flowing-retail/blob/master/payment-rest/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHacksControllerV2.java

Circuit Breaker

Failing Fast is important....

..but not enough

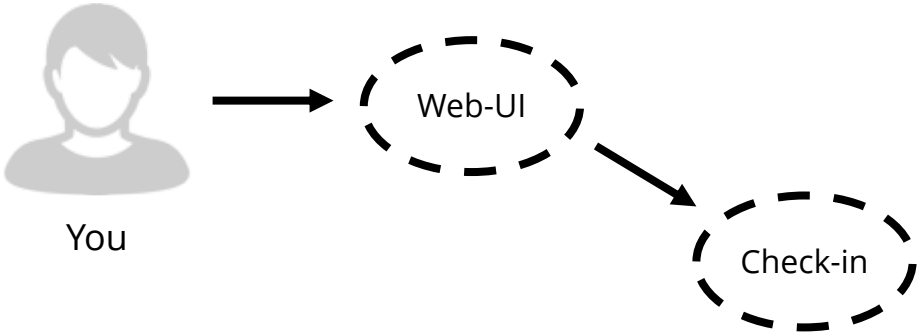
Photo by https://www.archdaily.com/560641/liverpool-insurgentes-department-store-rojkind-arquitectos

Internal Server Error - Read

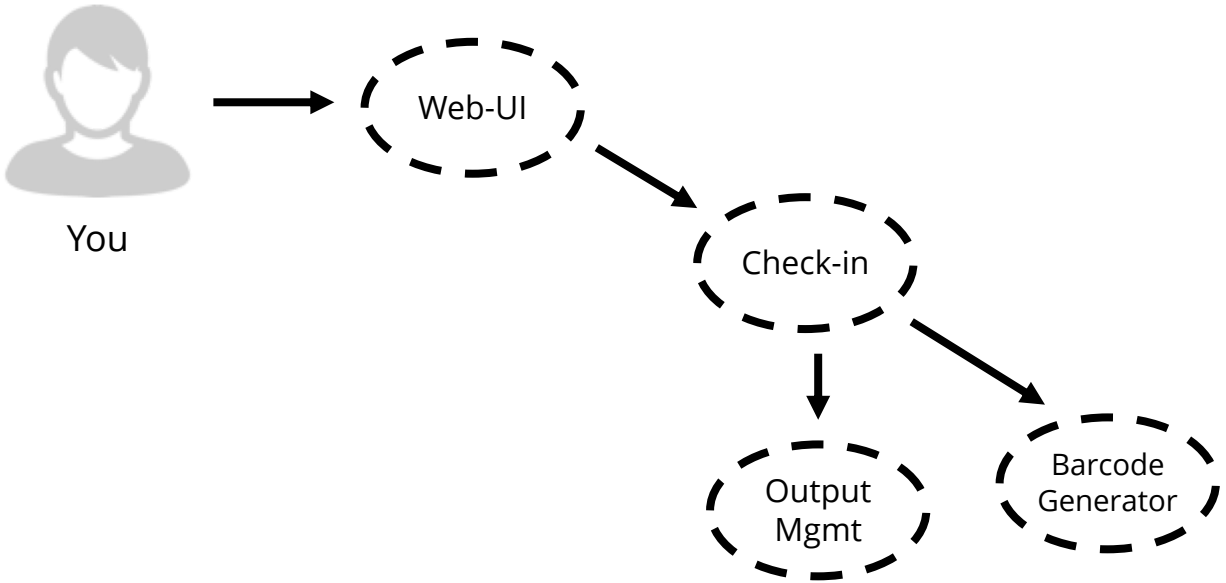
The server encountered an internal error or misconfiguration and was unable to complete your request.

Reference #3.1d079ccc.1519892932.9c55d68

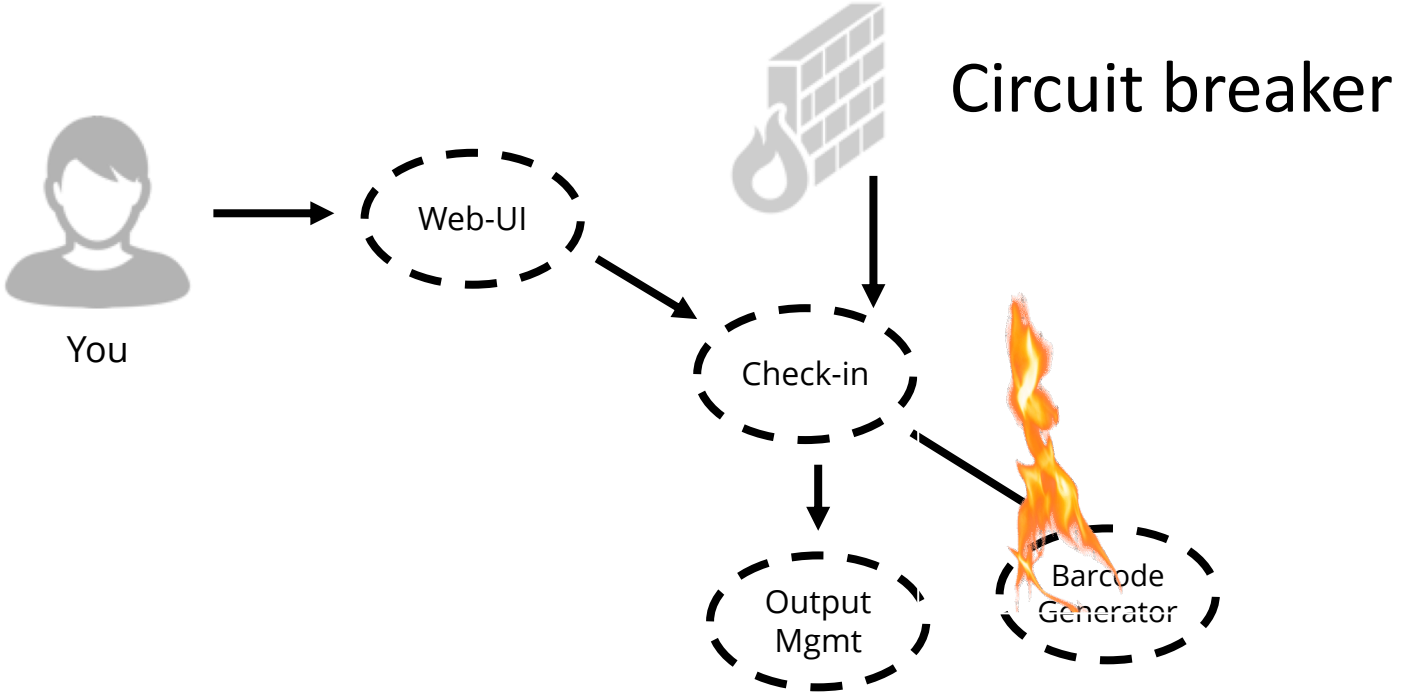
Current situation



Current situation

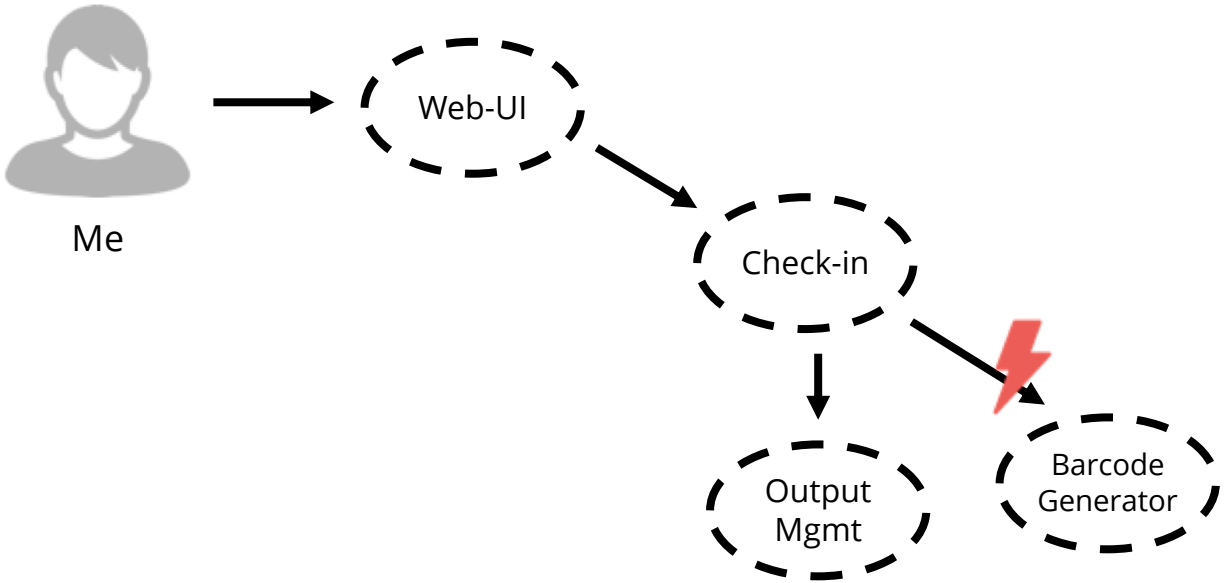


Current situation

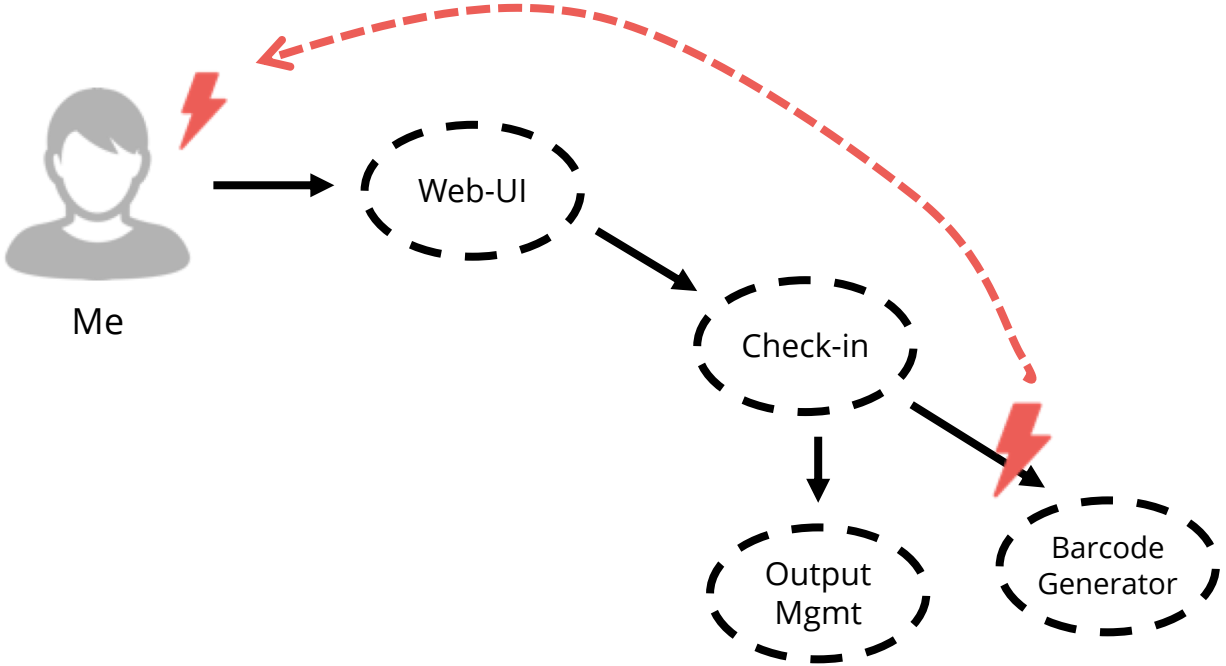


Another screenshot

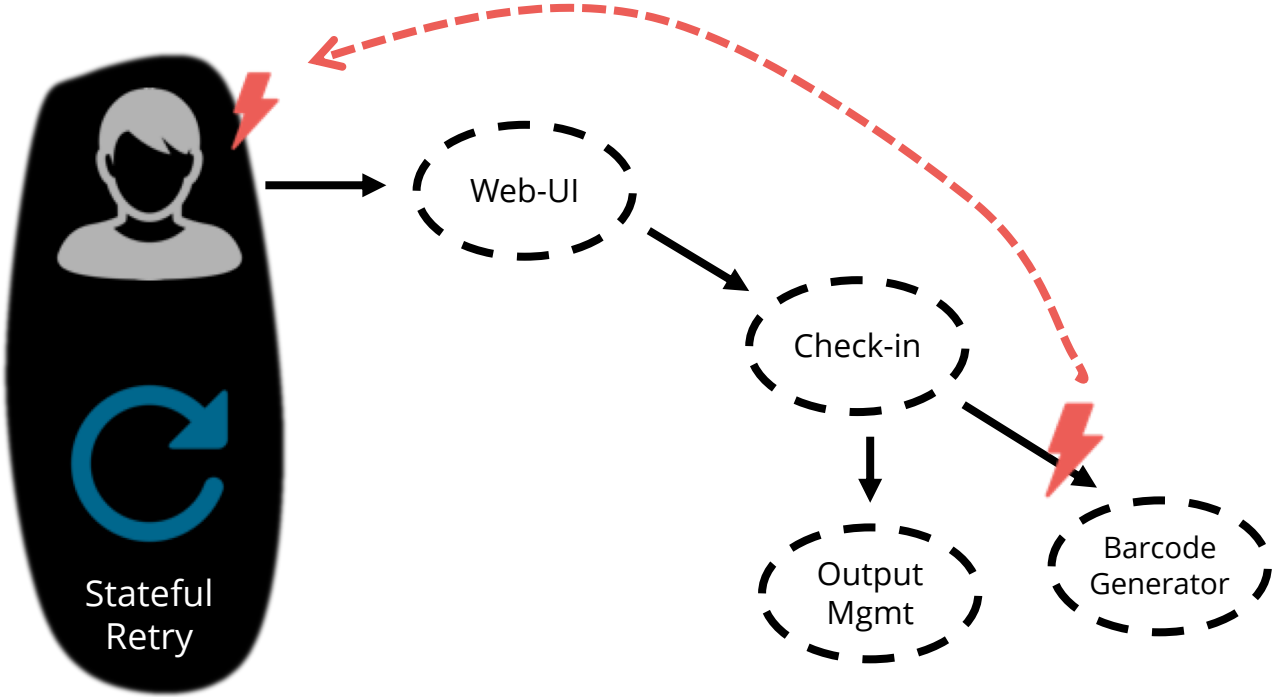
Current situation – the bad part



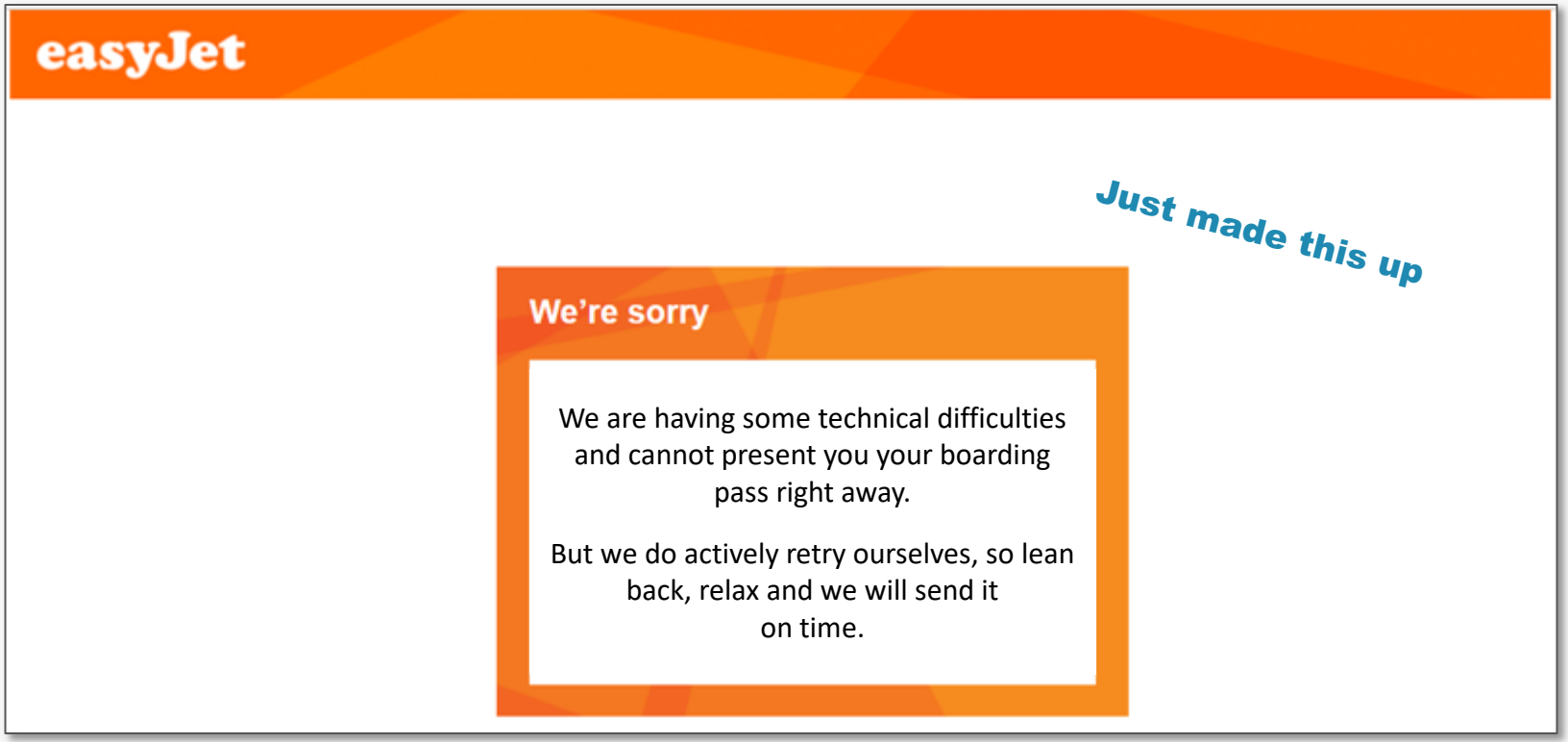
Current situation – the bad part



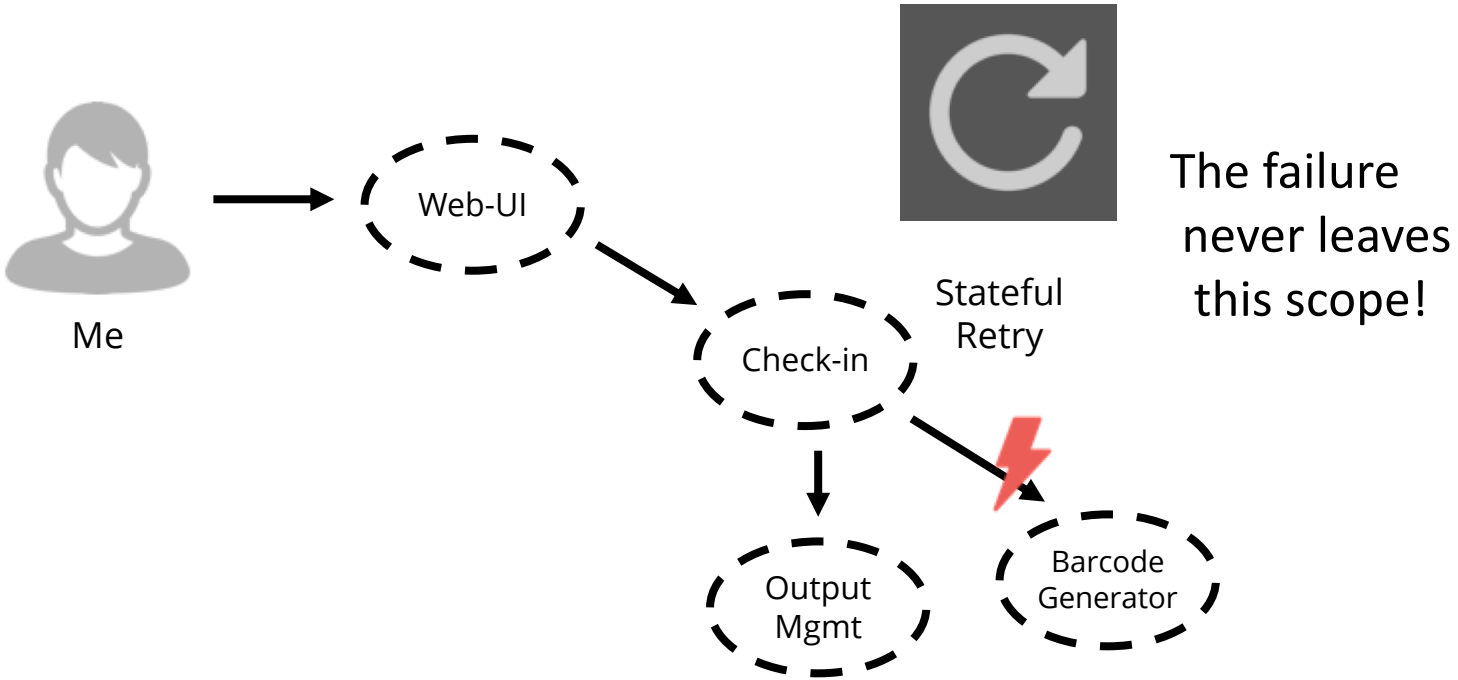
Current situation – the bad part



Another Example



Possible Solution – Much better?



Handling State

Persist thing (Entity, Document, Actor, …)

Typical concerns

DIY = effort, accidental complexity

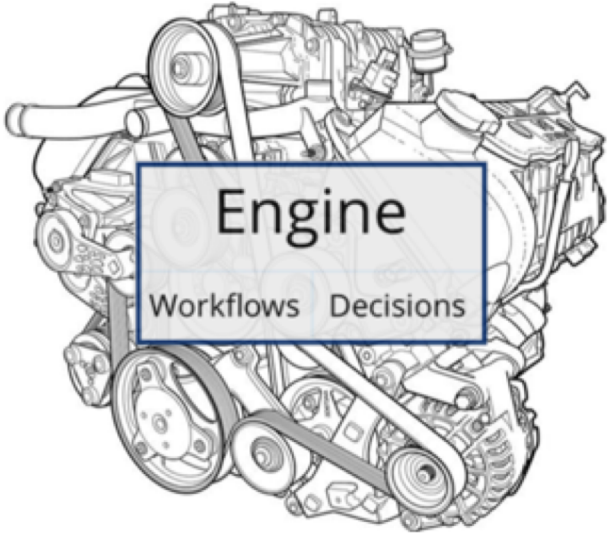
Scheduling, Versioning, operating, visibility, scalability, …

State machine or workflow engine

Typical concerns

Complex, proprietary, heavyweight, slow, don't scale, developer adverse

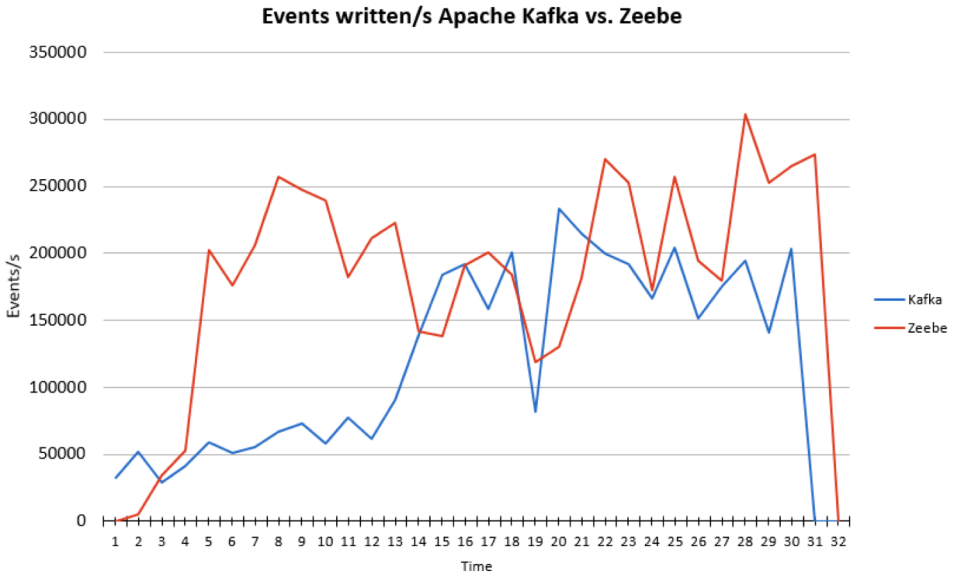
Current Players in the State Machine Market



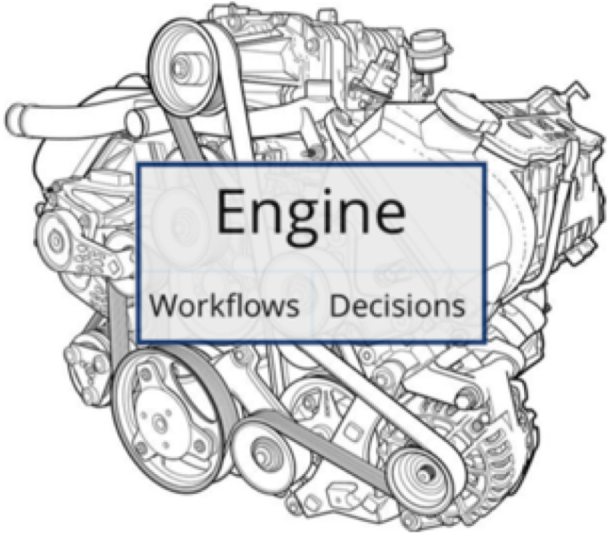
- AWS Step Function
- UBER Cadence
- Netflix Conductor
- Camunda ☺
- Zeebe ☺
- jBPM
- Activiti

Performance: Zeebe vs. Kafka

zeebe by Camunda **Vs.** Apache Kafka

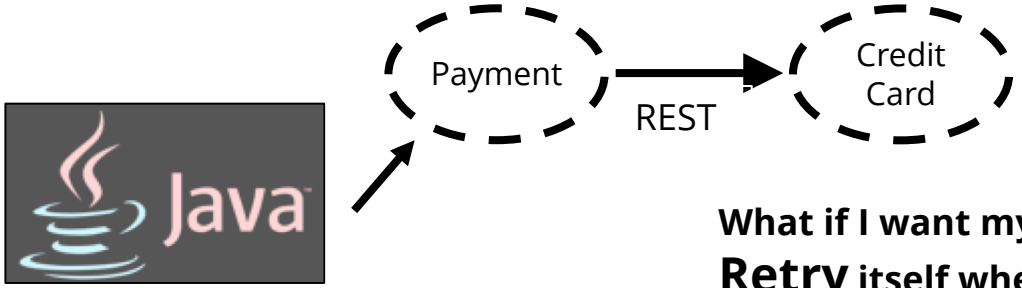


Current Players in the State Machine Market



- AWS Step Function
- UBER Cadence
- Netflix Conductor
- **Camunda ☺ (Raise of hand?)**
- Zeebe ☺
- jBPM
- Activiti

In the previous demo....

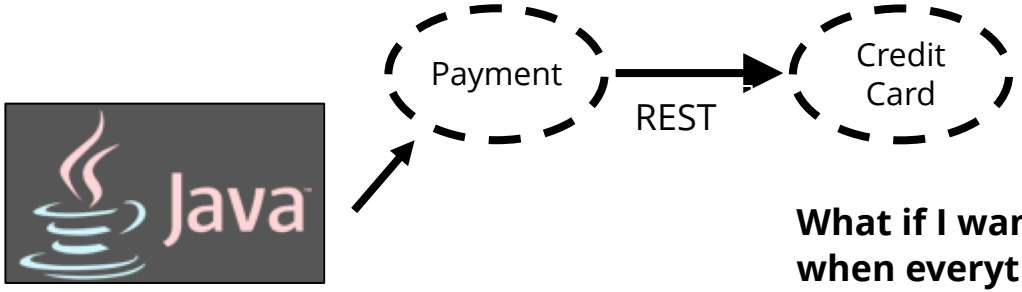


What if I want my Payment to be Asynchronous and Retry itself when my Credit Card Service Slow?

Payment Requestor Application

https://github.com/flowing/flowing-retail/blob/master/payment-rest/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHacksControllerV3.java

Demo

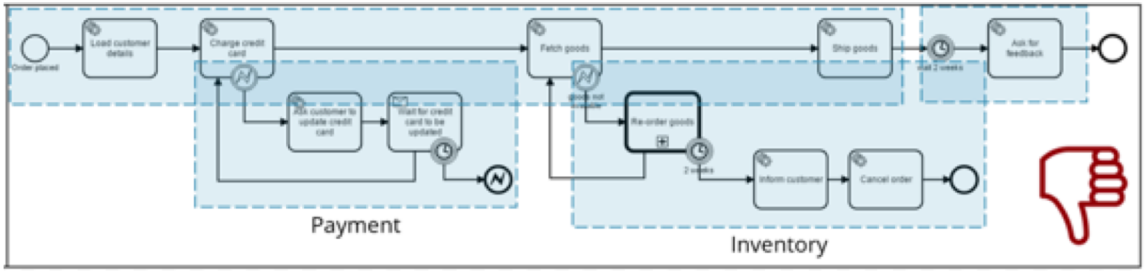


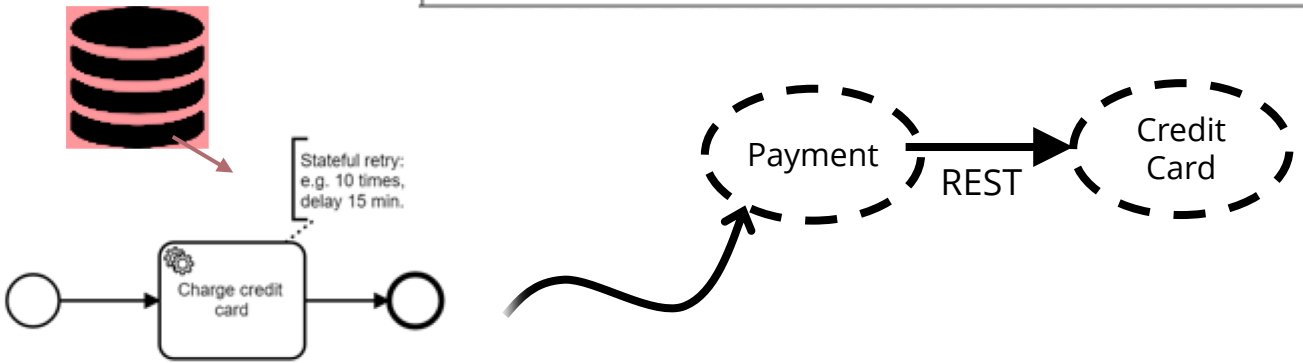
What if I want a Synchronous response when everything is fast?

Payment Requestor Application

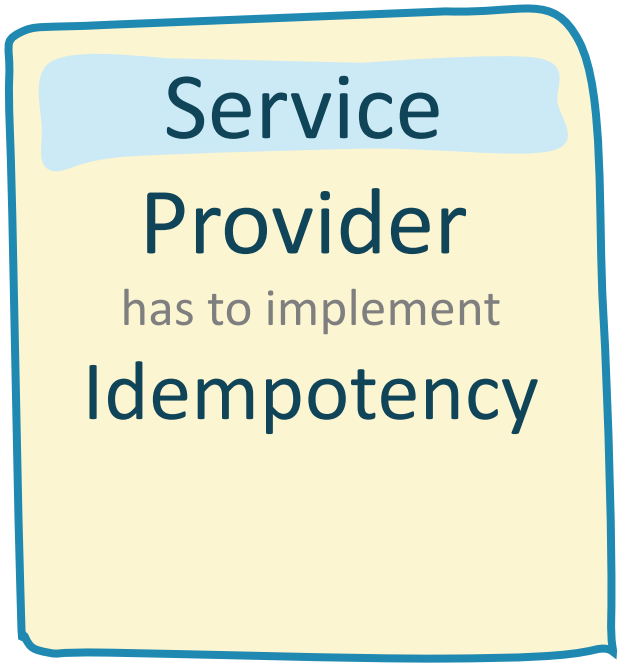
https://github.com/flowing/flowing-retail/blob/master/payment-rest/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHacksControllerV3.java

Now you have a state machine!





Most important factors to consider in distributed systems (so far..)



Bad Example..

It is a business problem anyway!

We are processing your payment.
Do not leave this page.

And for god sake – do not reload!

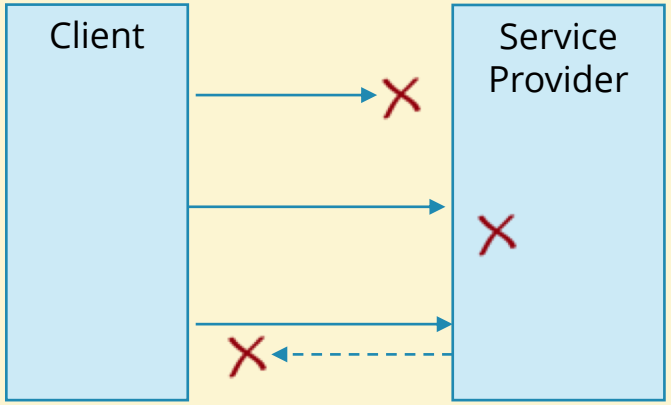
Better...

It is a business problem anyway!

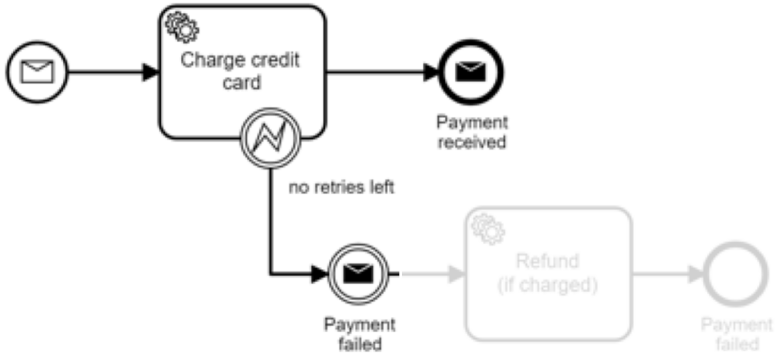
We are currently processing your request. Don‘t worry, it will happen safely –
even if you loose connection.
Feel free to reload this page any time!

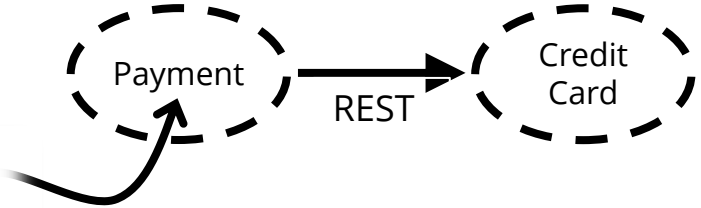
It is impossible to differentiate certain failure scenarios(and Code Exceptions).

Independant of communication style!

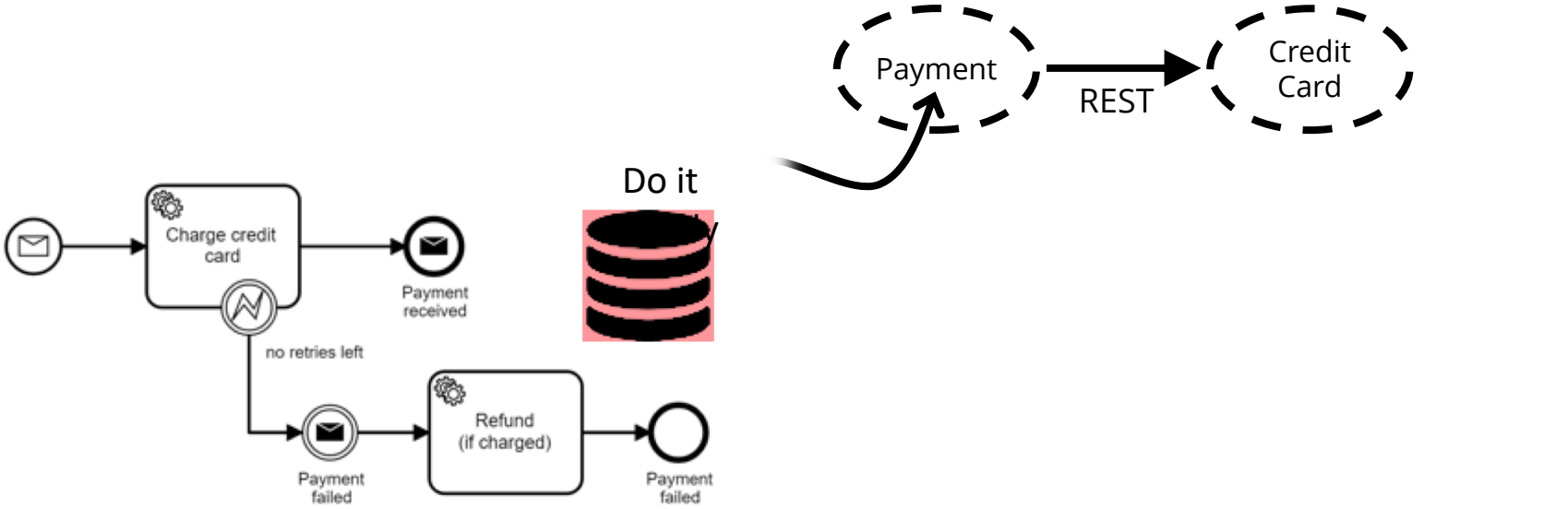


Distributed systems introduce complexity you have to tackle!

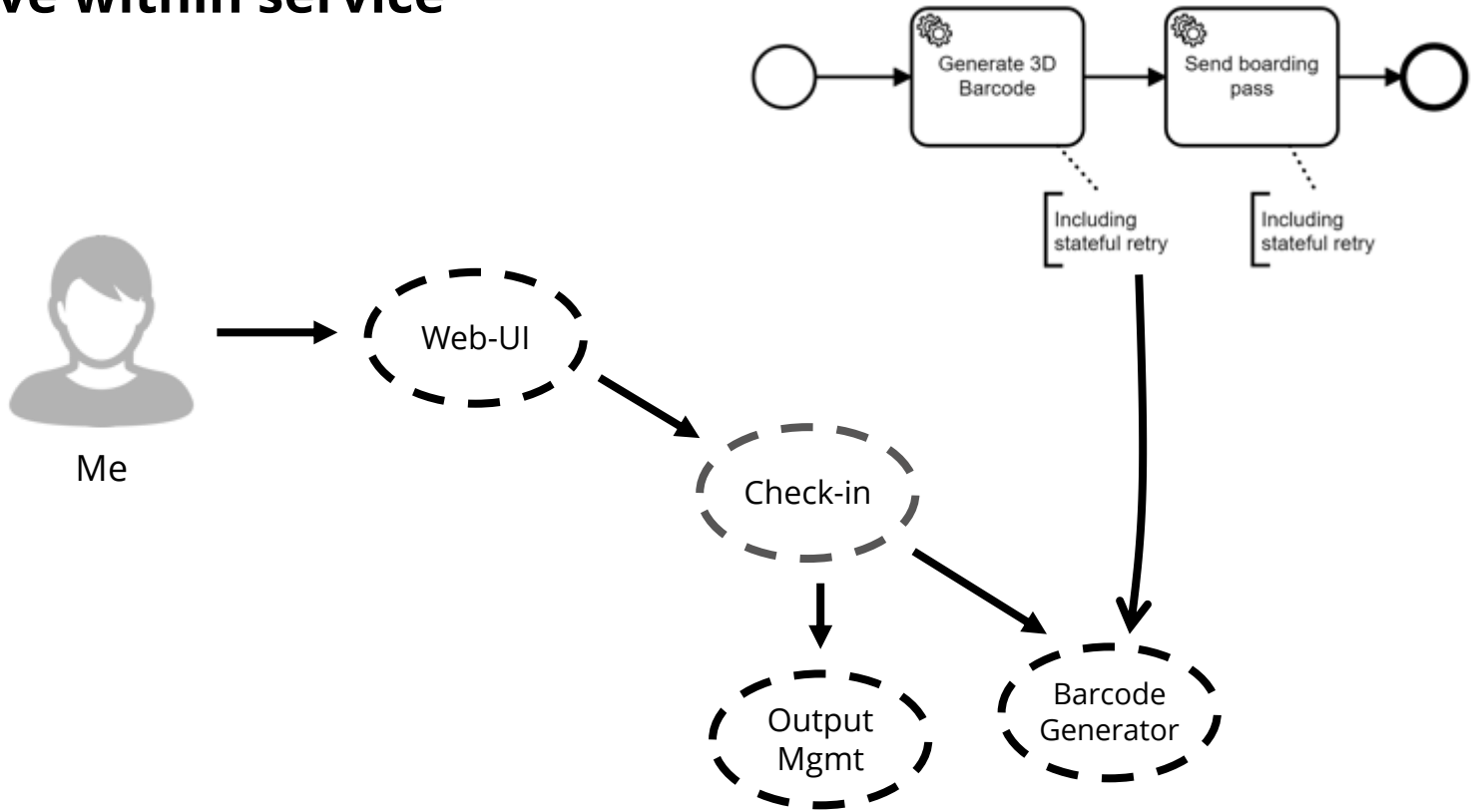




Distributed systems introduce complexity you have to tackle!



Workflows live within service boundaries



Different Architecture Options

berndruecker

Bernd Rücker
Dec 19, 2017 · 15 min read

Edit

Architecture options to run a workflow engine

This week a customer called and asked (translated into my own words and shortened):

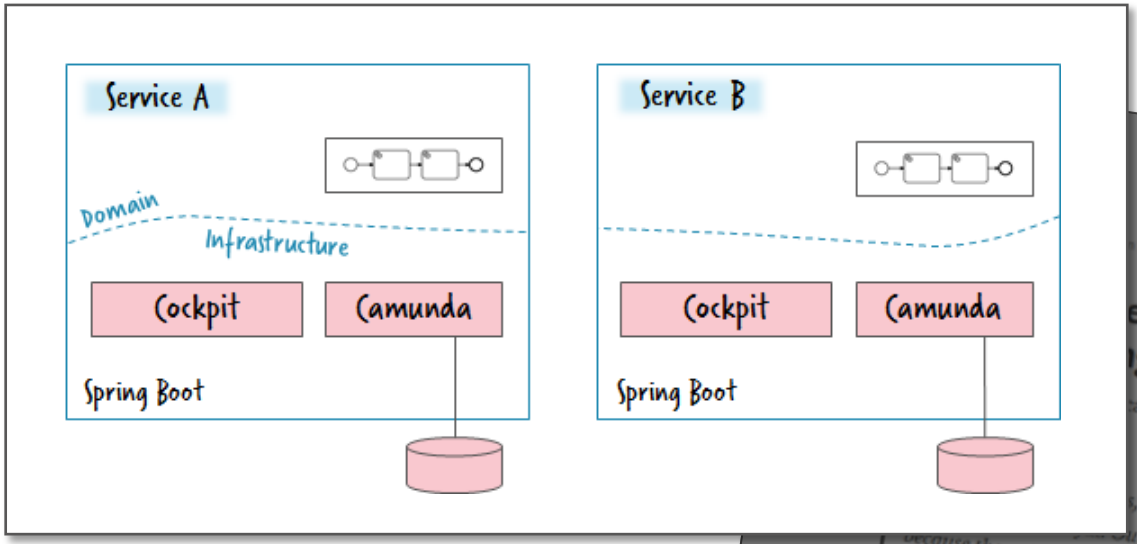
> *"We do composite services, orchestrating two or three CRUD-Services to do something more useful. Our architects want to use your workflow engine for this because the orchestration flow might be long running. Is this a valid scenario for workflow? Currently we run one big central cluster for the workflow engine—won't this get a mess?"*

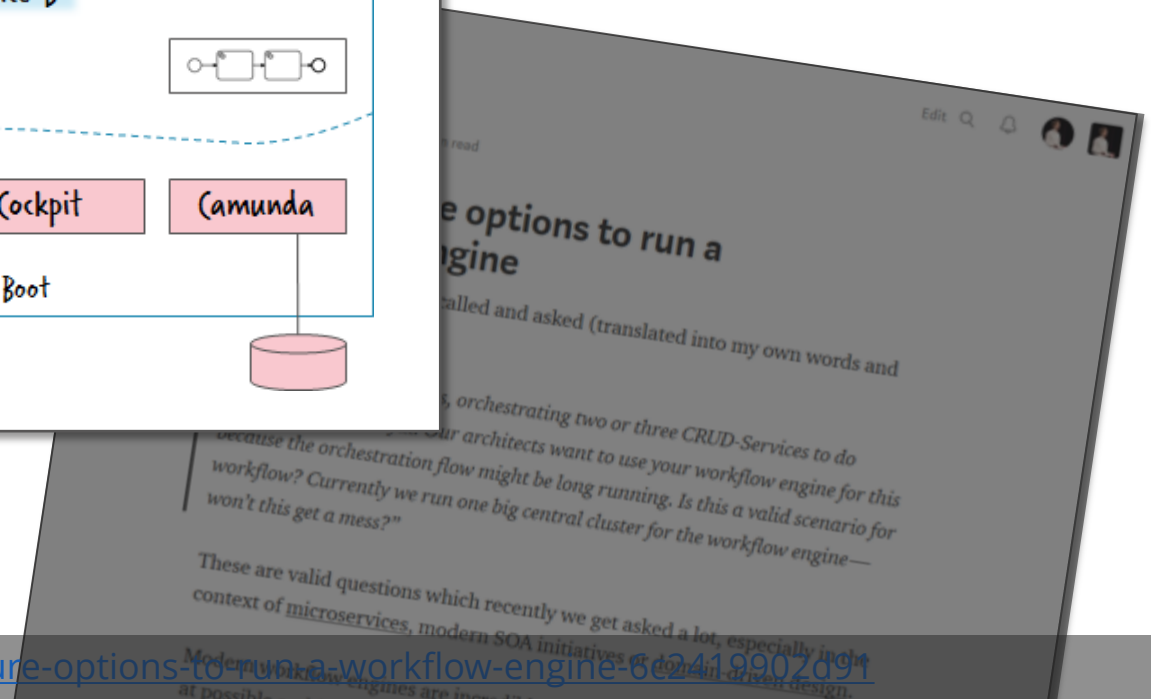
These are valid questions which recently we get asked a lot, especially in the context of microservices, modern SOA initiatives or domain-driven design.

Modern workflow engines are incredibly flexible. In this blog post I will look at possible architectures using them. To illustrate these architecture I use the open source products my company provides(Camunda and Zeebe).

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

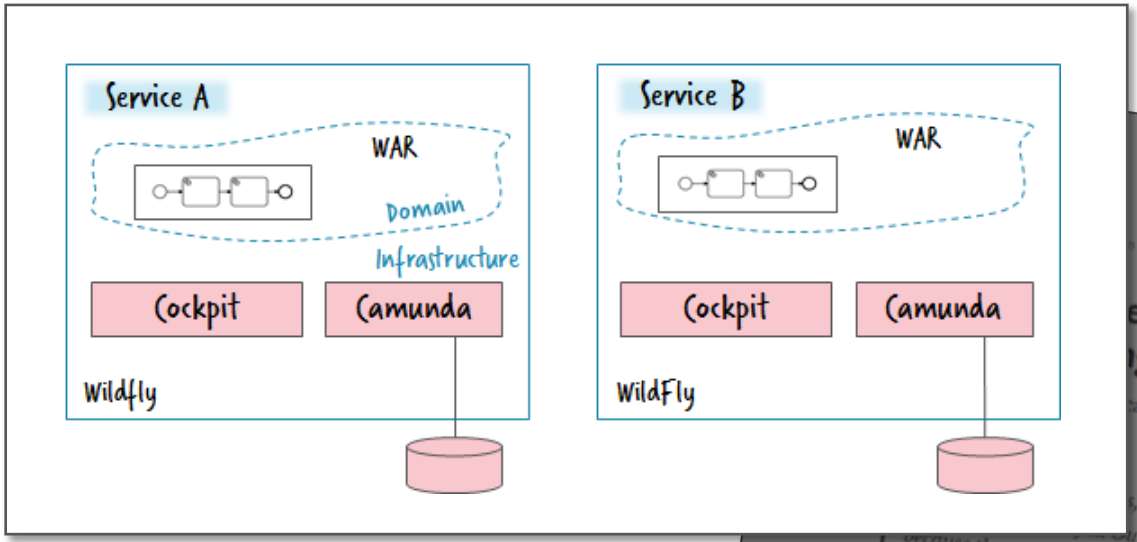
Different architecture options

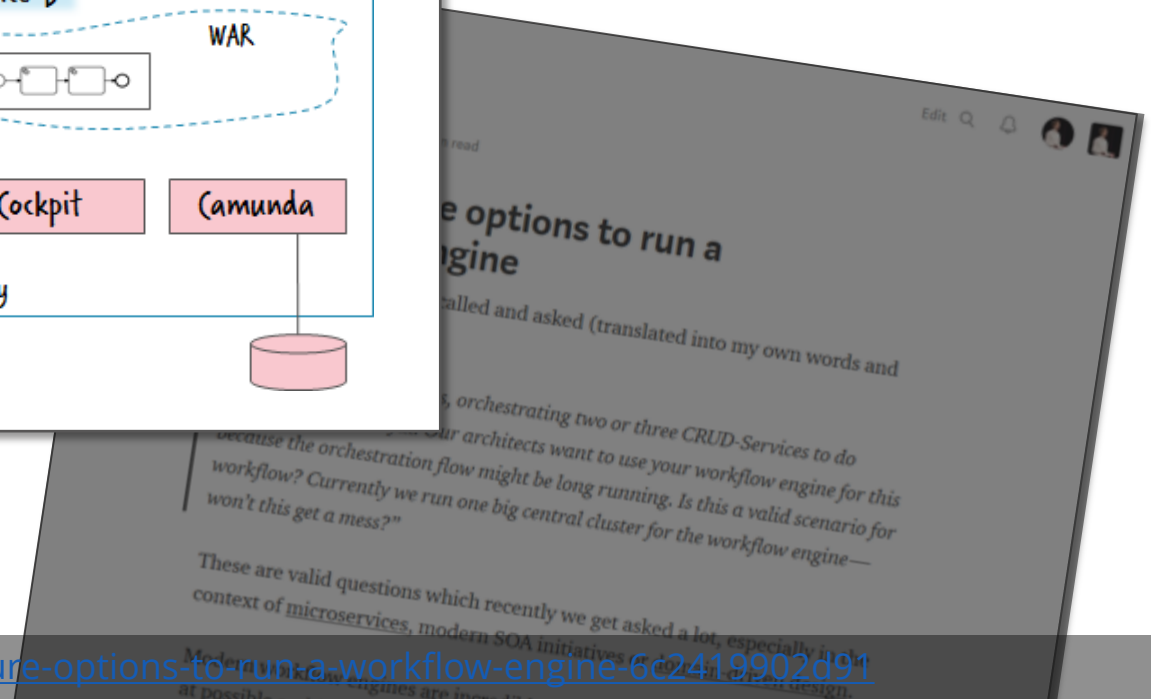




https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c24199021d91

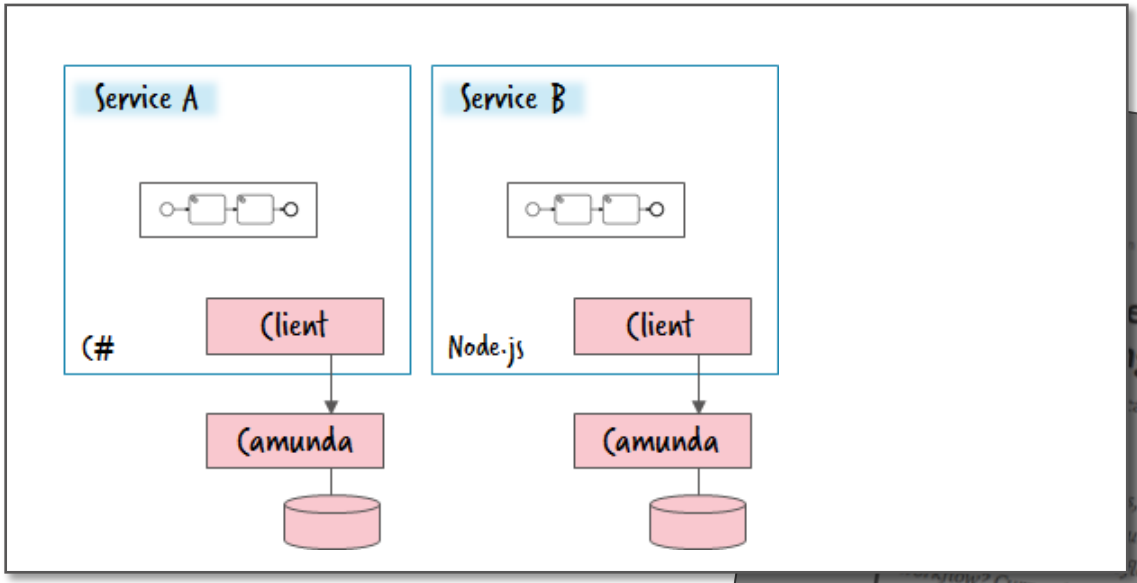
Different architecture options

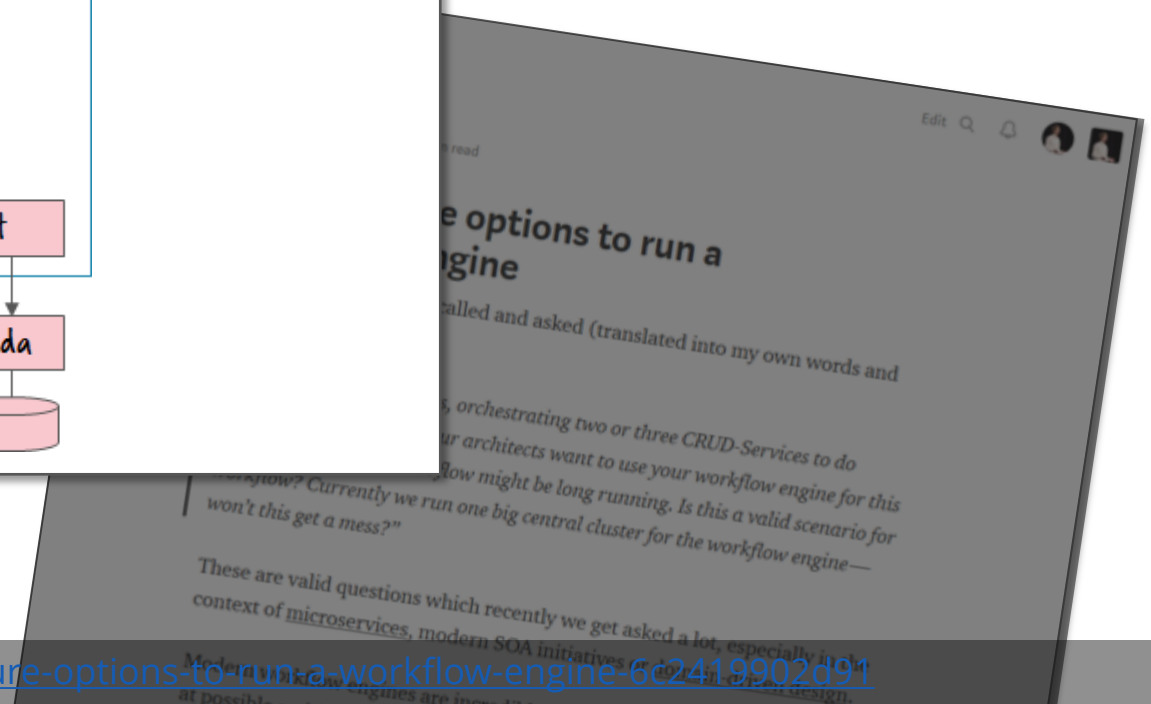




https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

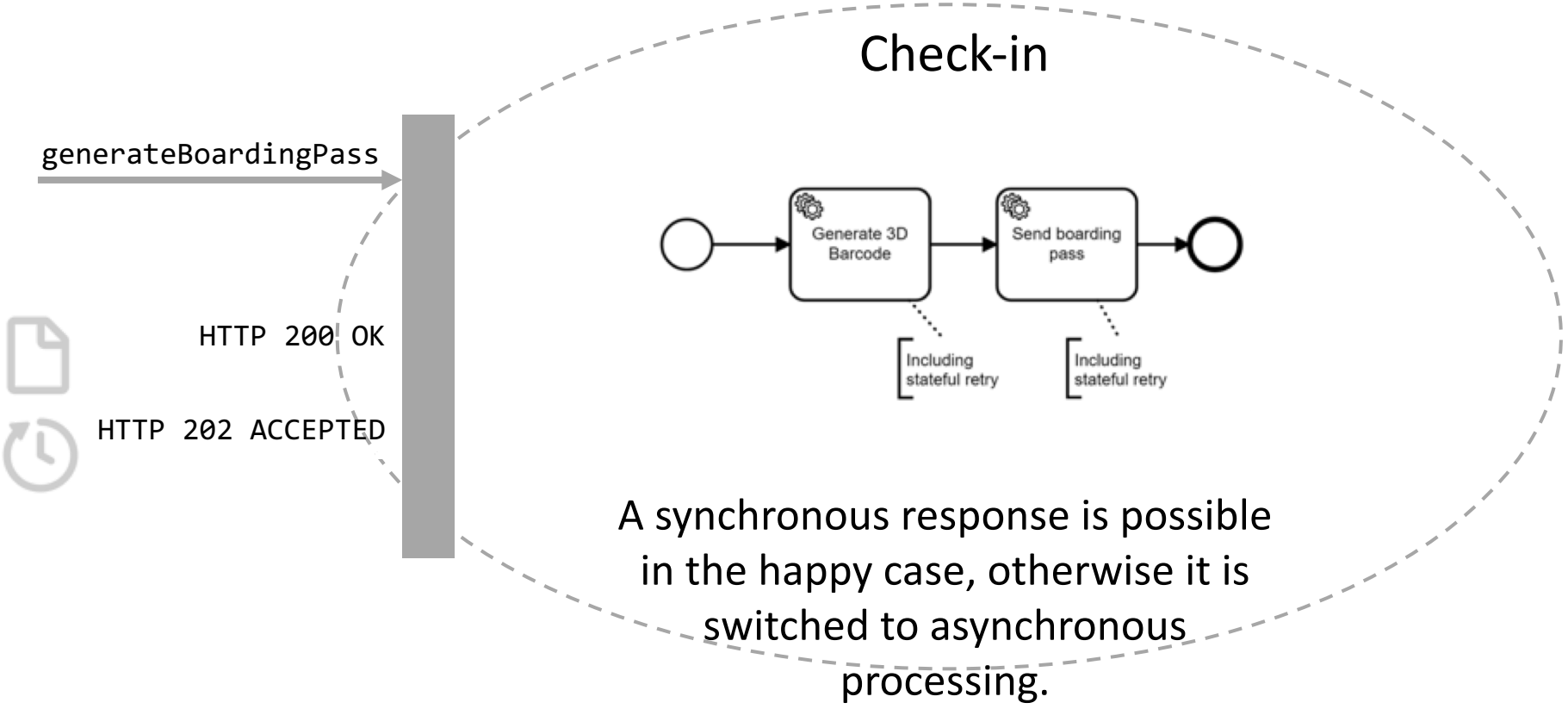
Different architecture options



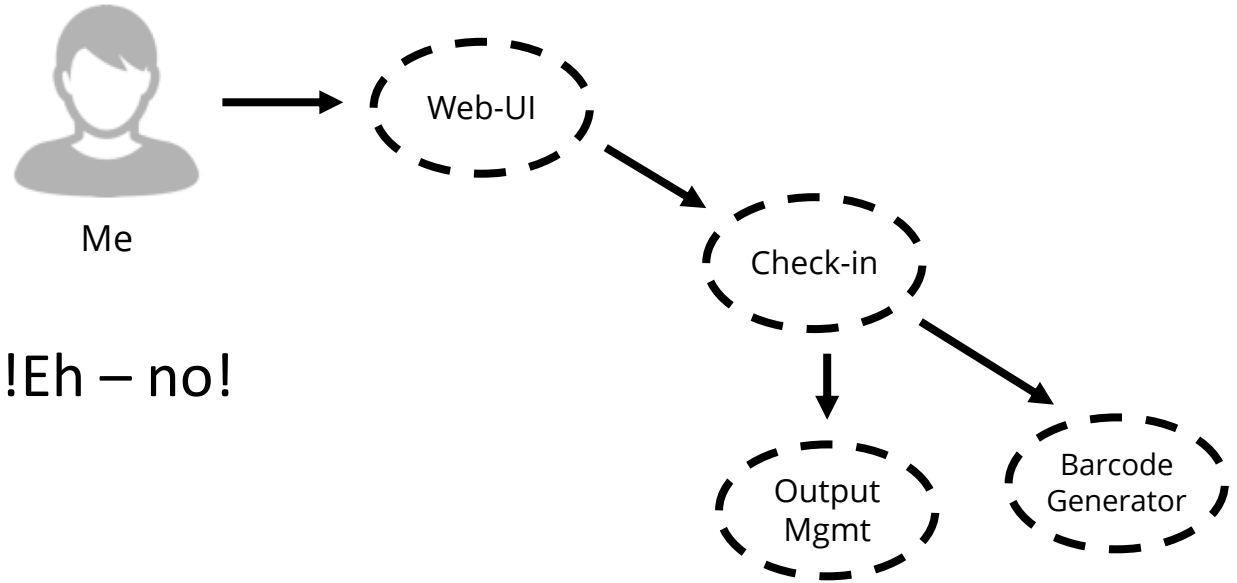


https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

First Sync then Async



The customer wants a synchronous response…



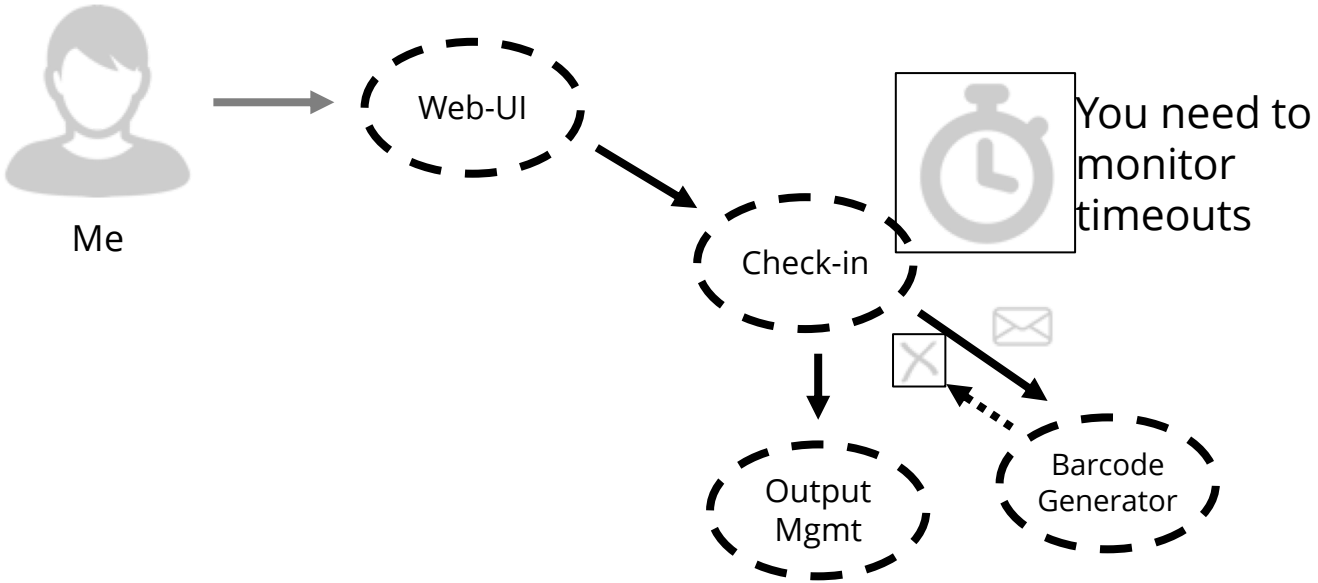
!Eh – no!

Synchronous communication is the crystal meth of distributed programming

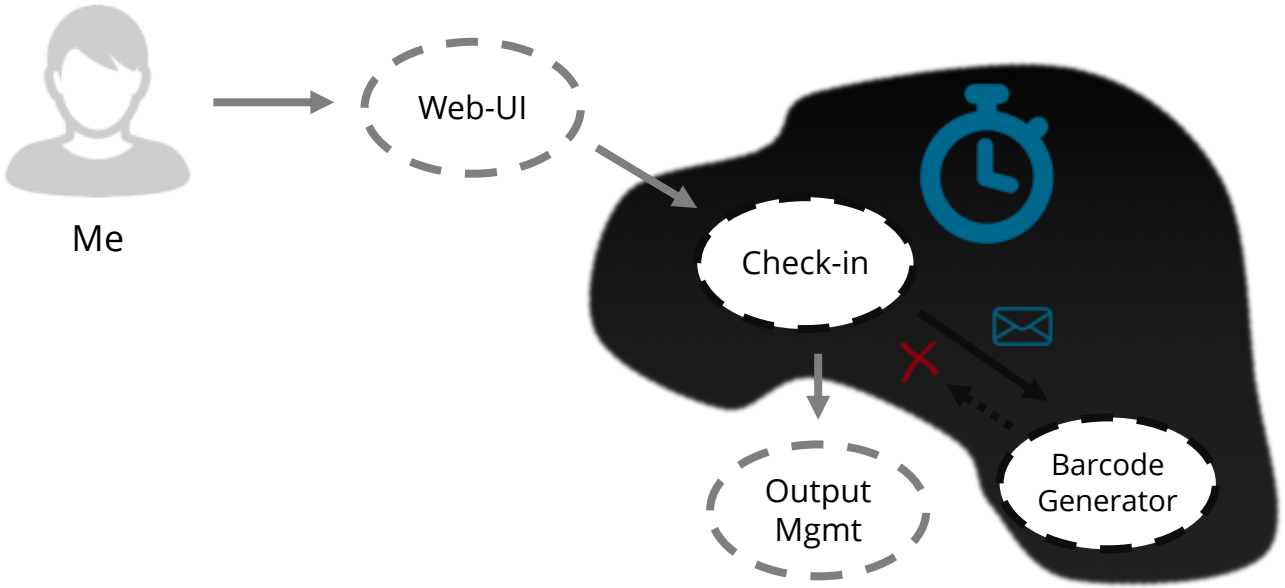
Todd Montgomery and Martin Thompson
in “How did we end up here” at GOTO Chicago 2015

Challenges of asynchronicity

Asynchronous communication

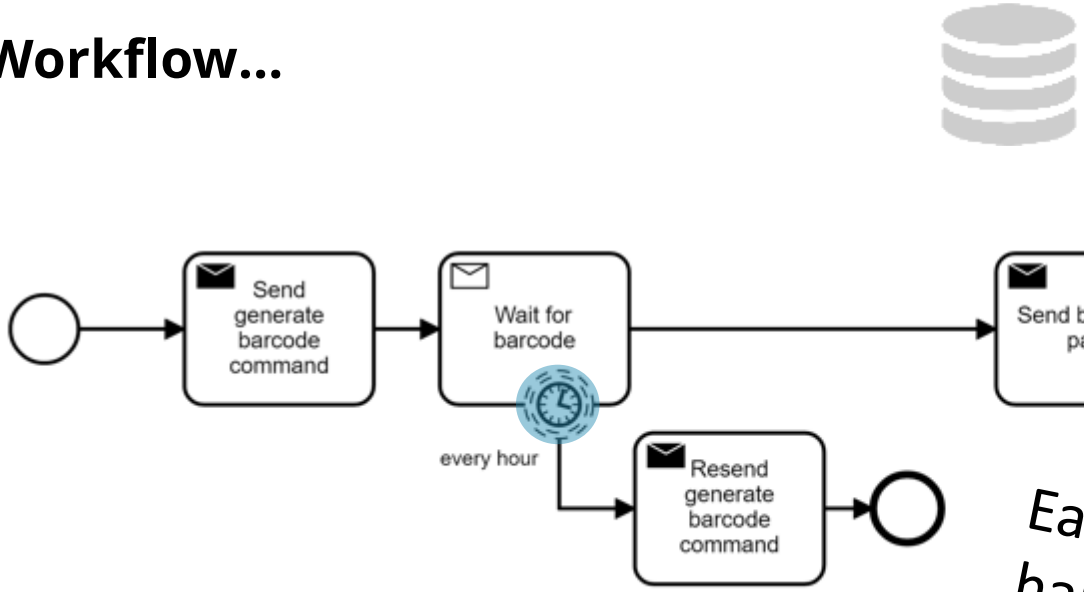


Remember…



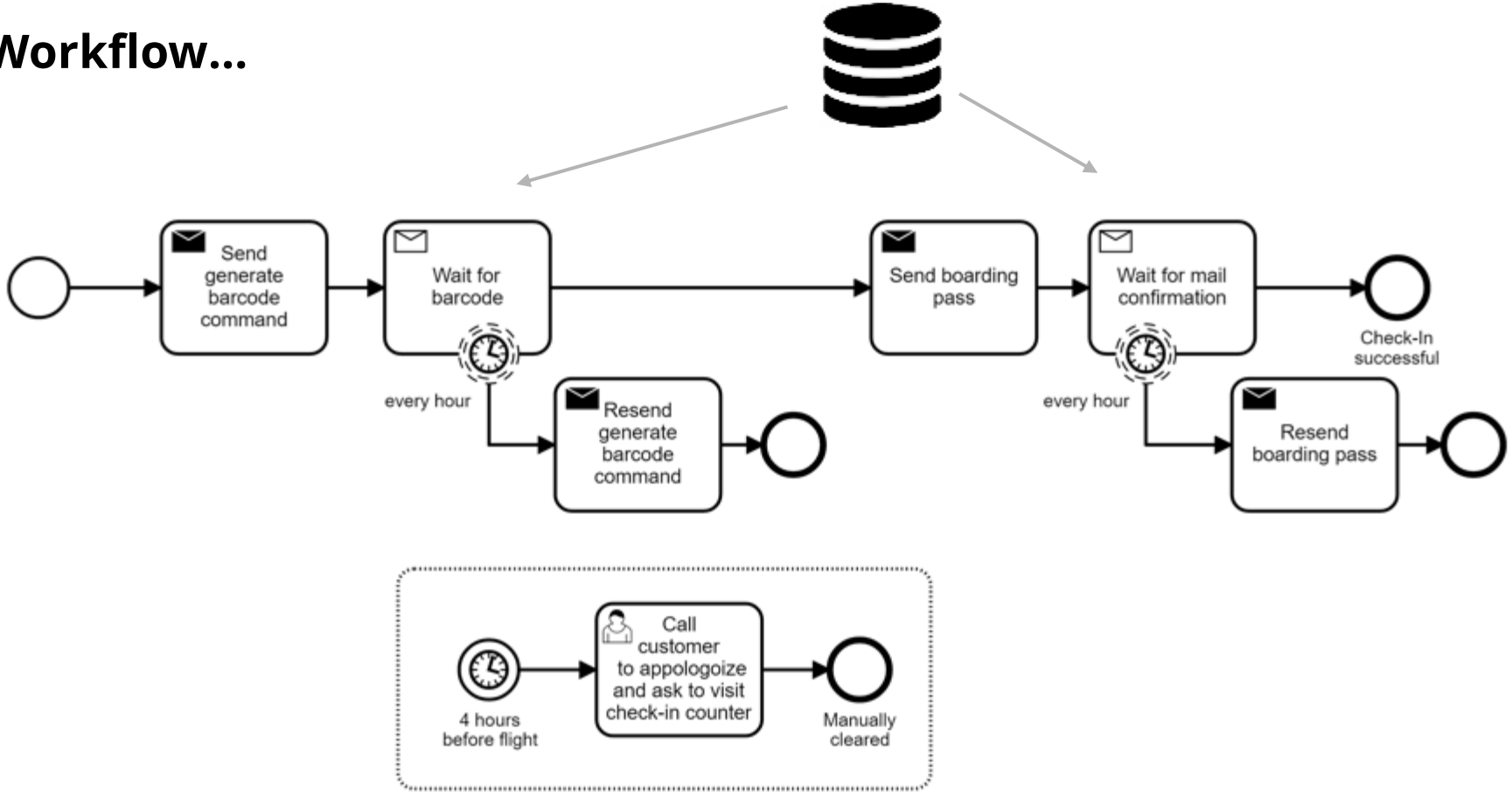
The failure never leaves this scope!

Workflow…



Easy to handle time

Workflow...


Send generate barcode command
Wait for barcode
every hour
Resend generate barcode command
Send boarding pass
Wait for mail confirmation
every hour
Resend boarding pass
Check-In successful
4 hours before flight
Call customer to appologoize and ask to visit check-in counter
Manually cleared

Client

has to implement

Timeout, Retry

Service

Provider

has to implement

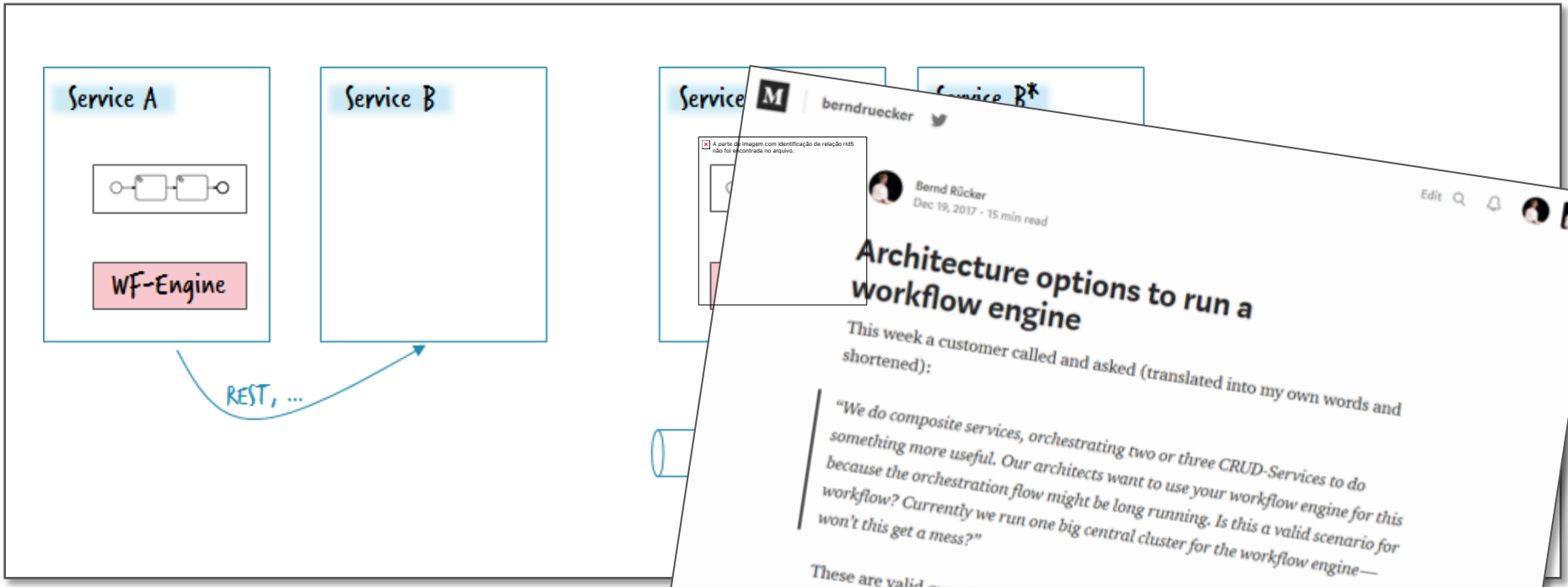
Idempotency

Who uses a message bus?

Who has no problems operating a message bus?

Dead messages | No context | Inaccesible payload | Hard to redeliver |
Home-grown message hospitals | …

Other Architecture options

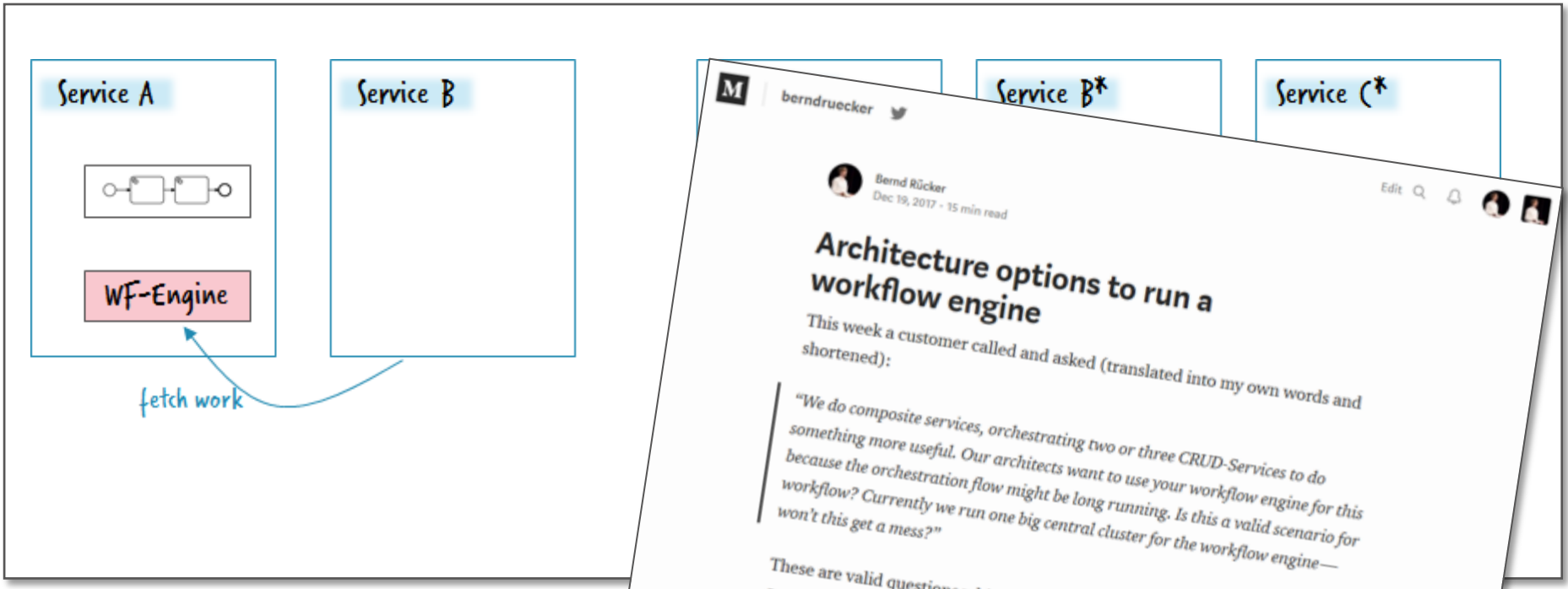


https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Other Architecture options

https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Other Architecture Options



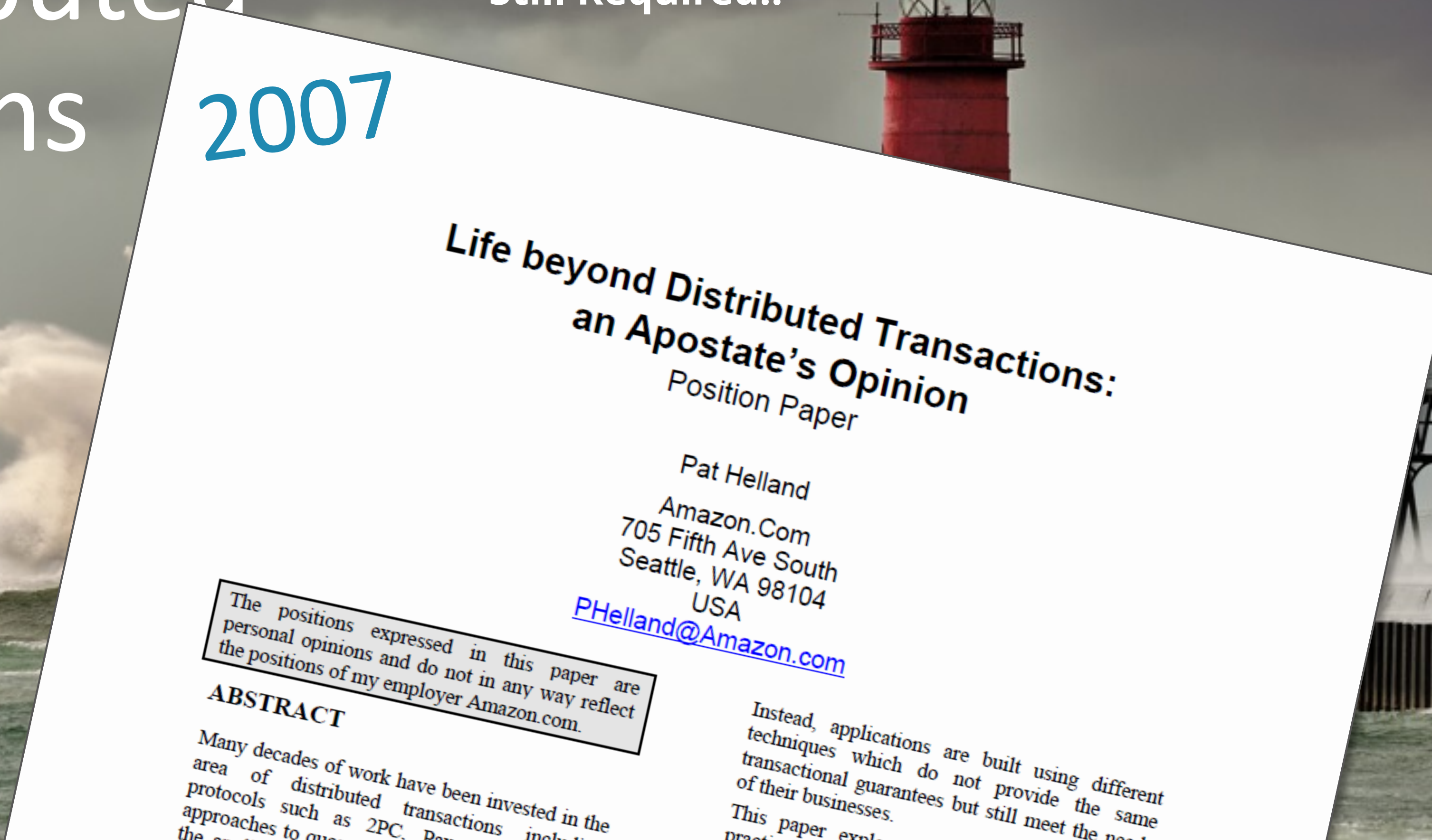
https://blog.bernd-ruecker.com/architecture-o

Distributed Transactions

Distributed systems

- ACID Transactions
- Scalability
- Troubleshooting TM
- Still Required!!

2007



Life beyond Distributed Transactions: an Apostate's Opinion

Position Paper

Pat Helland

Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

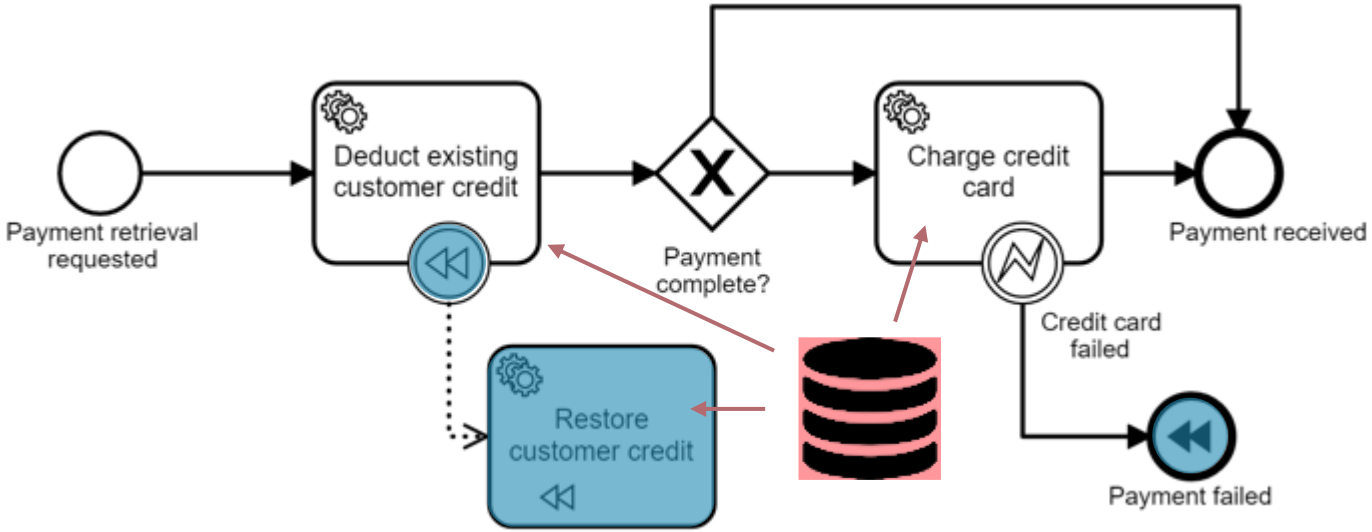
The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

ABSTRACT

Many decades of work have been invested in the area of distributed transactions ...

Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

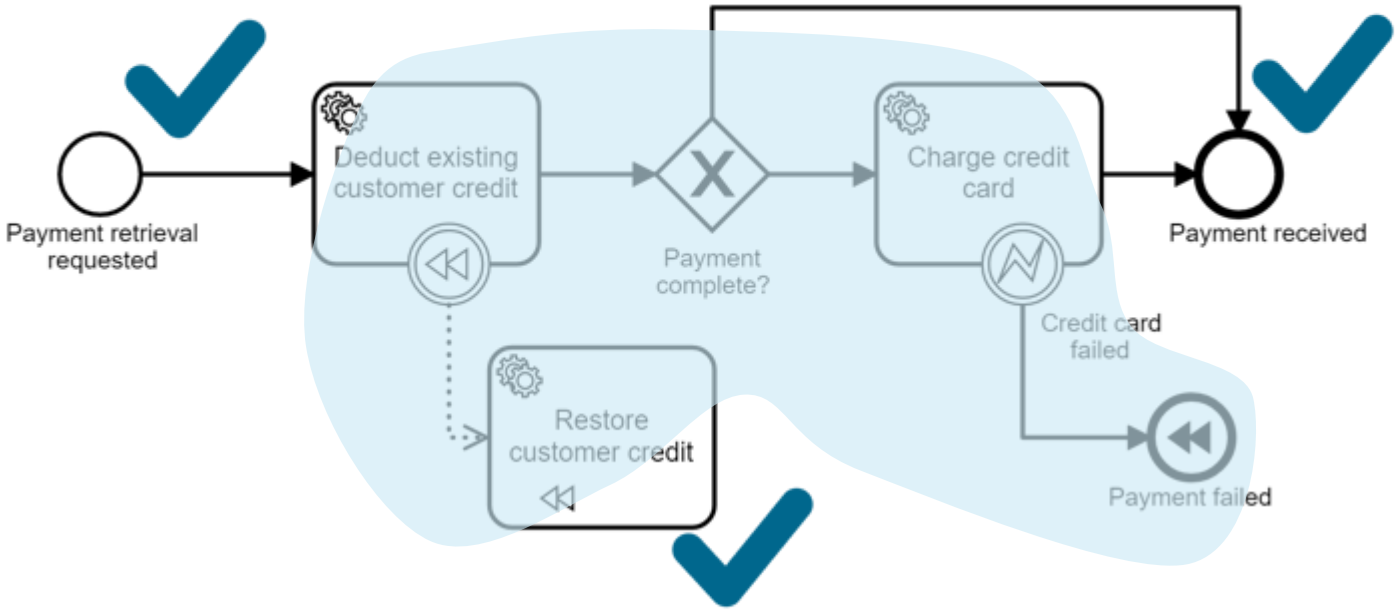
Distributed transactions using compensation *



Compensation

*aka Saga pattern

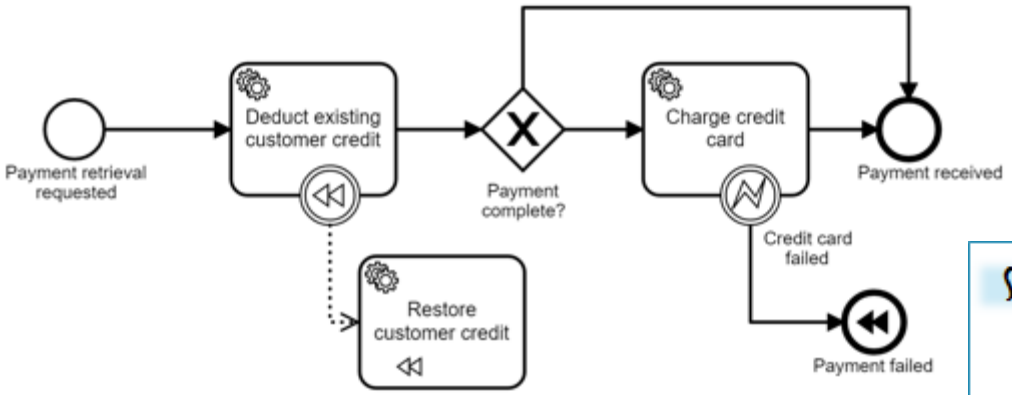
Eventual consistency

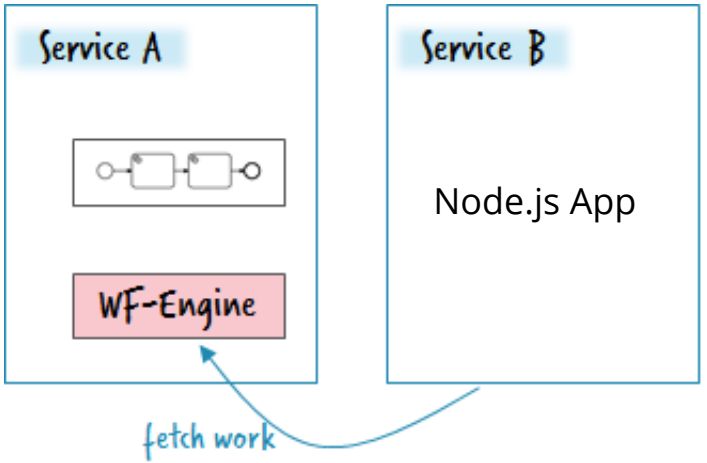


Temporarily inconsistent state

But only temporarily!

Demo Time





Payment Requestor Application

Live hacking

https://github.com/flowing/flowing-retail/blob/master/payment-rest/src/main/java/io/flowing/retail/payment/port/resthacks/PaymentRestHacksControllerV6.java

Client

has to implement

Timeout, Retry, Compensation

Service

Provider

has to offer

Compensation

has to implement

Idempotency

Don‘t forget about state

Client

has to implement

Timeout Retry

atic

Service

Provider

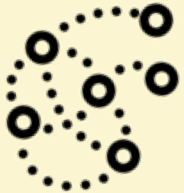
has to offer

Compensation

has to

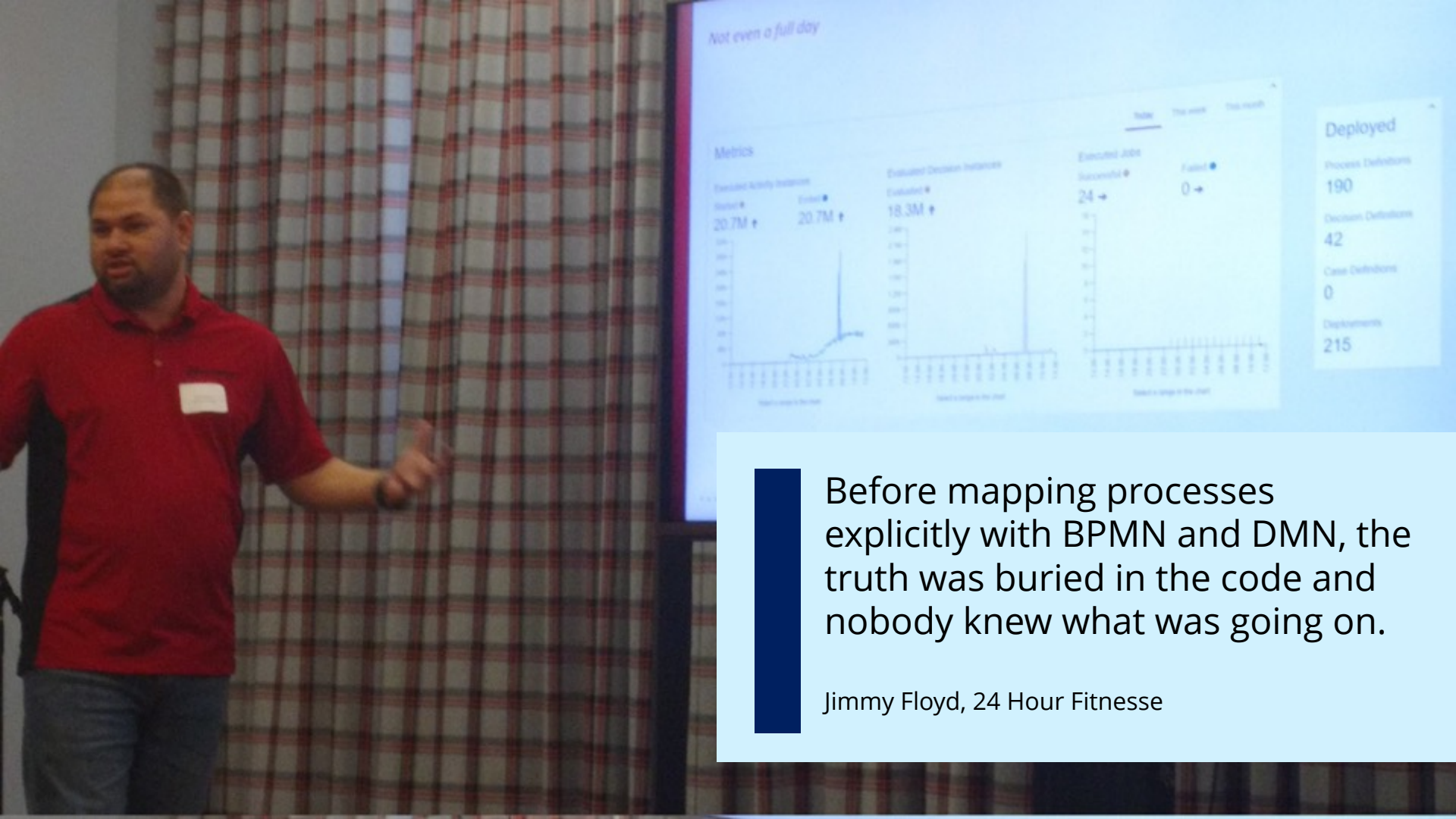
dem

Communication is complex



Challenges of asynchronicity

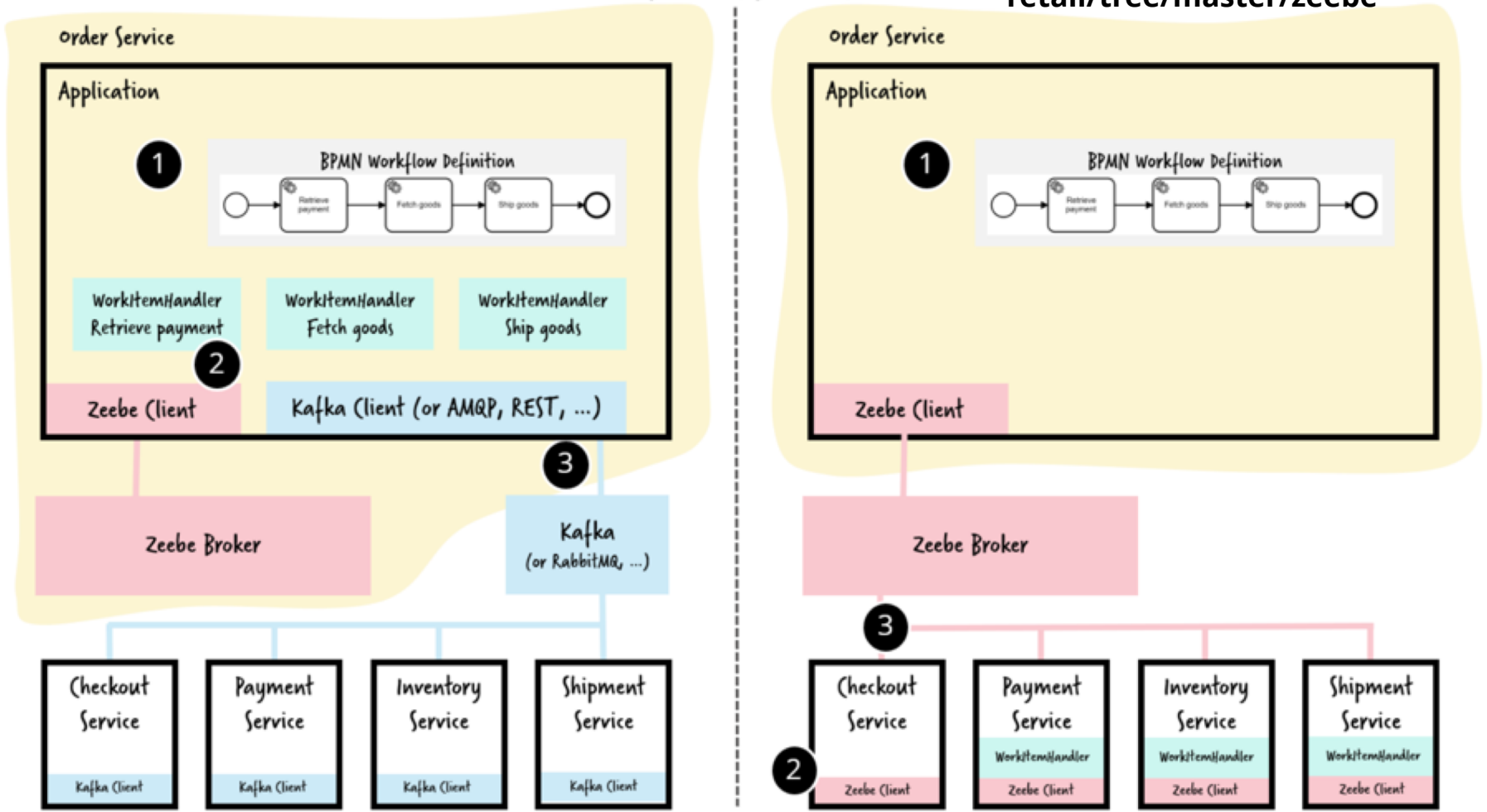
Distributed Transactions



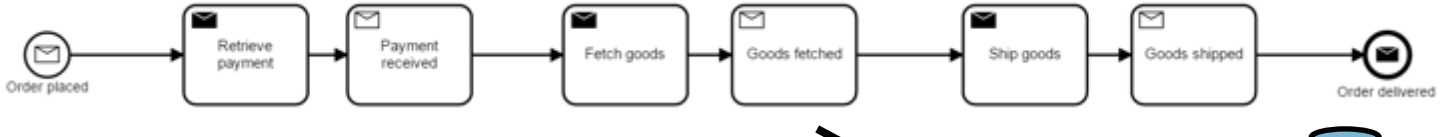
Before mapping processes explicitly with BPMN and DMN, the truth was buried in the code and nobody knew what was going on.

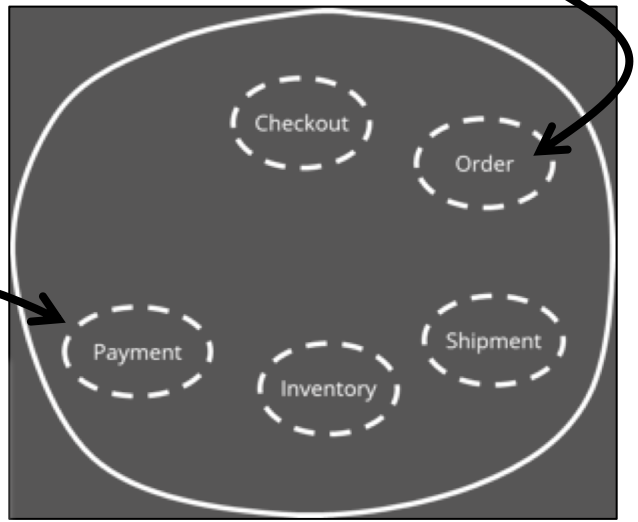
Jimmy Floyd, 24 Hour Fitnesse

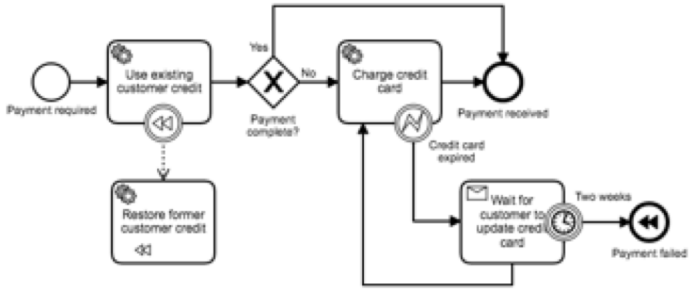
https://github.com/flowing/flowing-retail/tree/master/zeebe

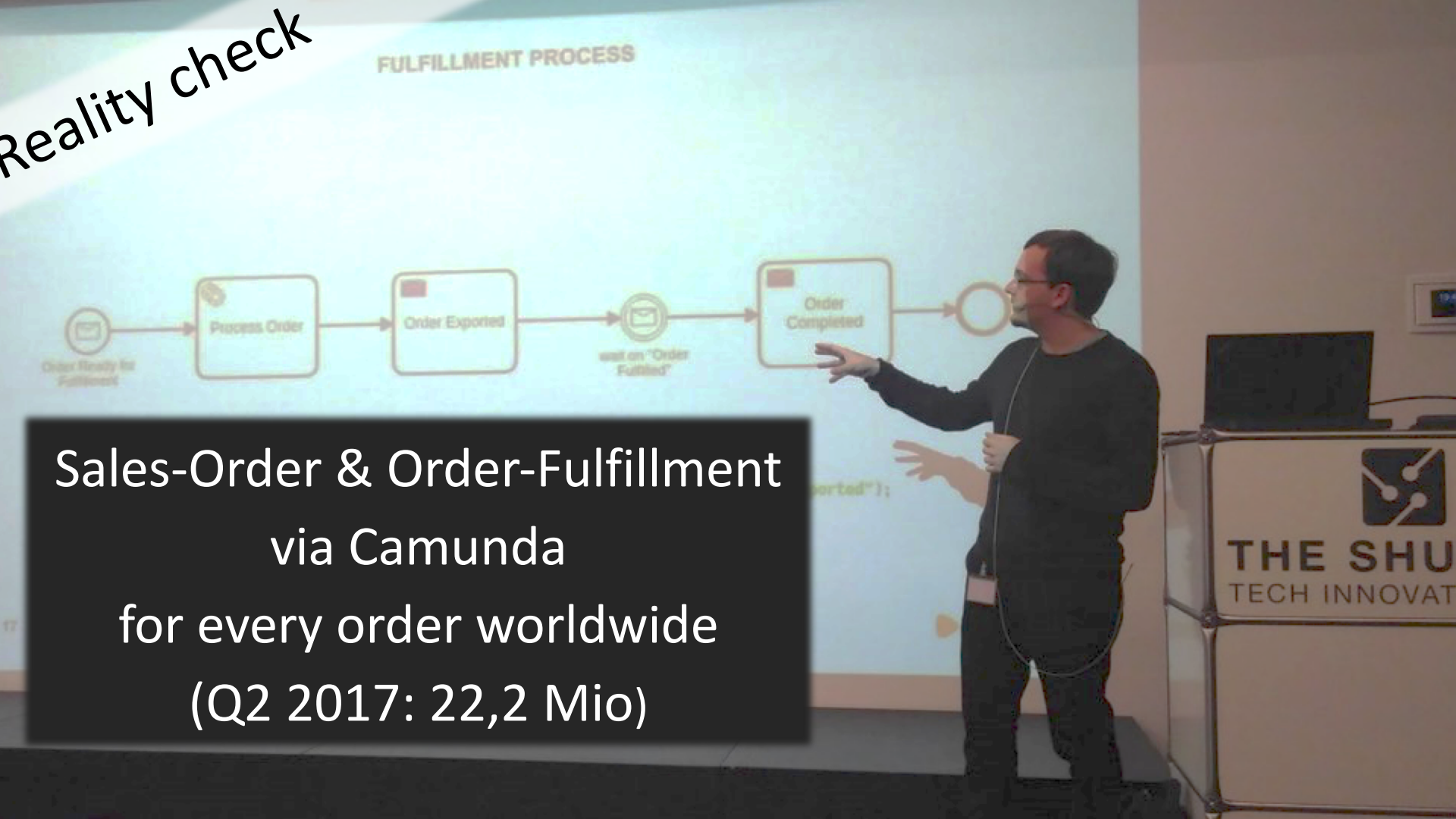


Workflows live inside service boundaries








Reality check
FULFILLMENT PROCESS
Process Order
Order Exported
Order Completed
Sales-Order & Order-Fulfillment
via Camunda
for every order worldwide
(Q2 2017: 22,2 Mio)

Some of the Workflow Engine Use Cases and… what we talked about

Camunda Value

Improving development, operations and visibility of automated workflows and decisions.

Technical Use Cases

Straight-Through Processing

Microservice Orchestration

Human Workflow Management

Business Rule Automation

Business Process Examples

E-Commerce: Order Execution

Finance: Stock Trading

Insurance: Claim Settlement

Telco: OSS/BSS

……..

Be aware of complexity of distributed systems

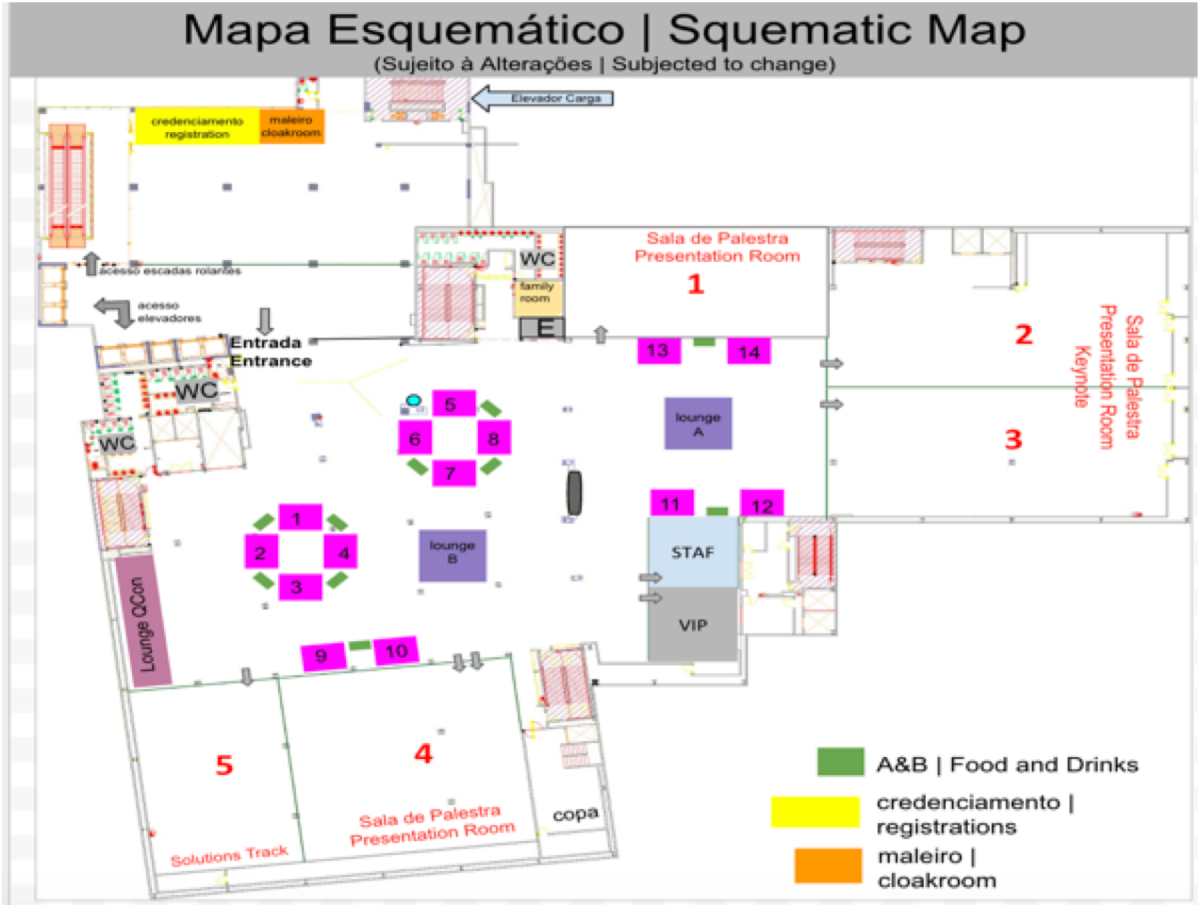
Know strategies and tools to handle it

e.g. Circuit breaker (Hystrix)

Workflow engine for stateful retry, waiting, timeout and compensation (Camunda)

https://www.infoworld.com/article/3254777/application-development/3-common-pitfalls-of-microservices-integrationand-how-to-avoid-them.html

Where to learn more



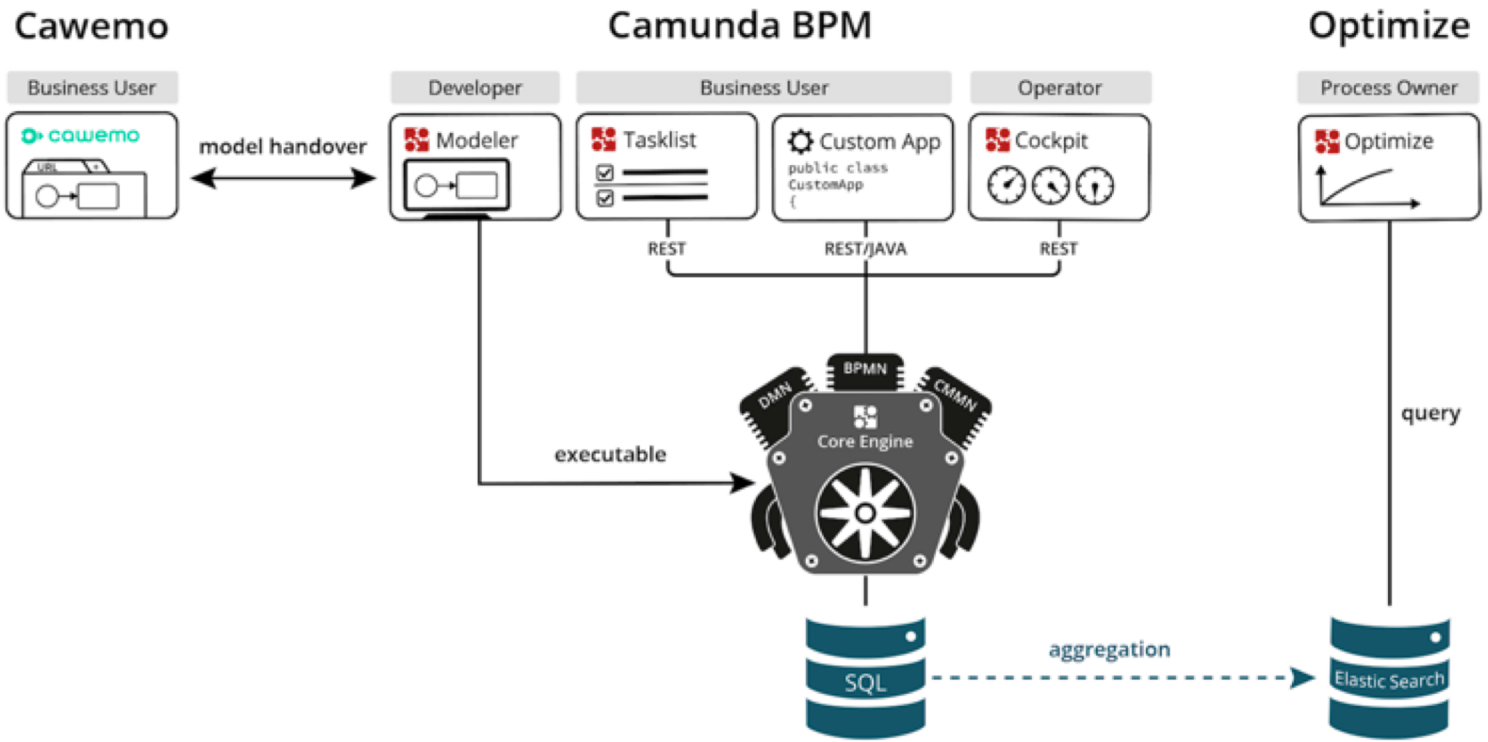
Contact Us

- Andreas Stange | International Sales
  - +49-172-862-2730 | Berlin

- Mauricio Bitencourt | Customer Delivery & Success
  - +55 51 984.087.798 | São Paulo

Q&A

Camunda Ecosystem



Model > **Execute** > **Improve**

What is ZEEBE?

- Zeebe scales orchestration of workers and microservices using visual workflows. Zeebe is horizontally scalable and fault tolerant so that you can reliably process all your transactions as they happen.