Lost in transaction?

Über (In-)Konsistenz in verteilten Systemen

Benjamin Hoffmann

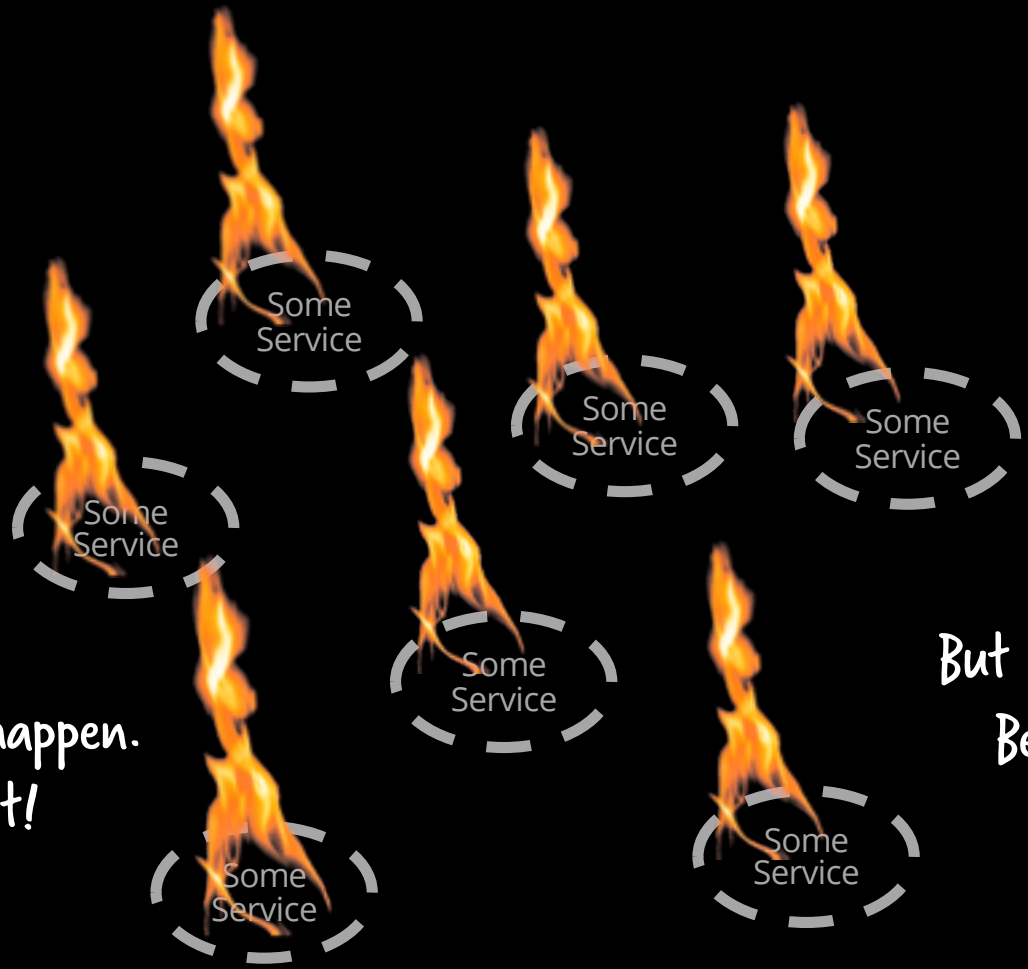
Technical Consultant
Camunda

Berlin, Germany

benjamin.hoffmann@camunda.com

REST, SOAP,
Cloud, Saas,
Microservices, SCS,
FaaS, Serverless,
...

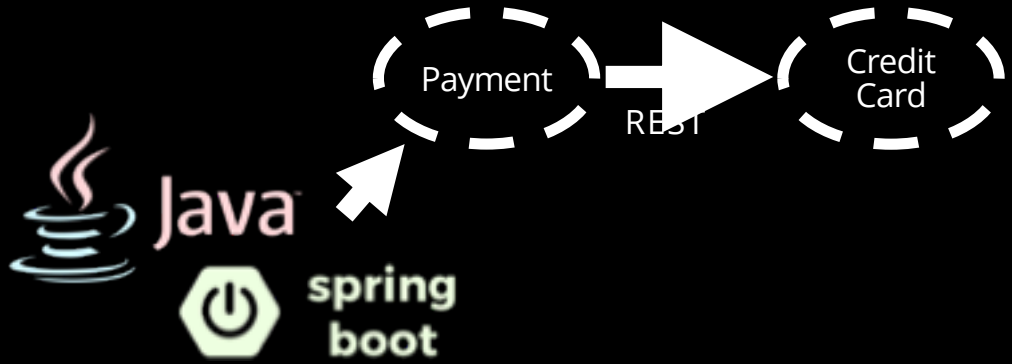
Distributed systems



Failure will happen.
Accept it!

But keep it local!
Be resilient.

Let's start with a simple example



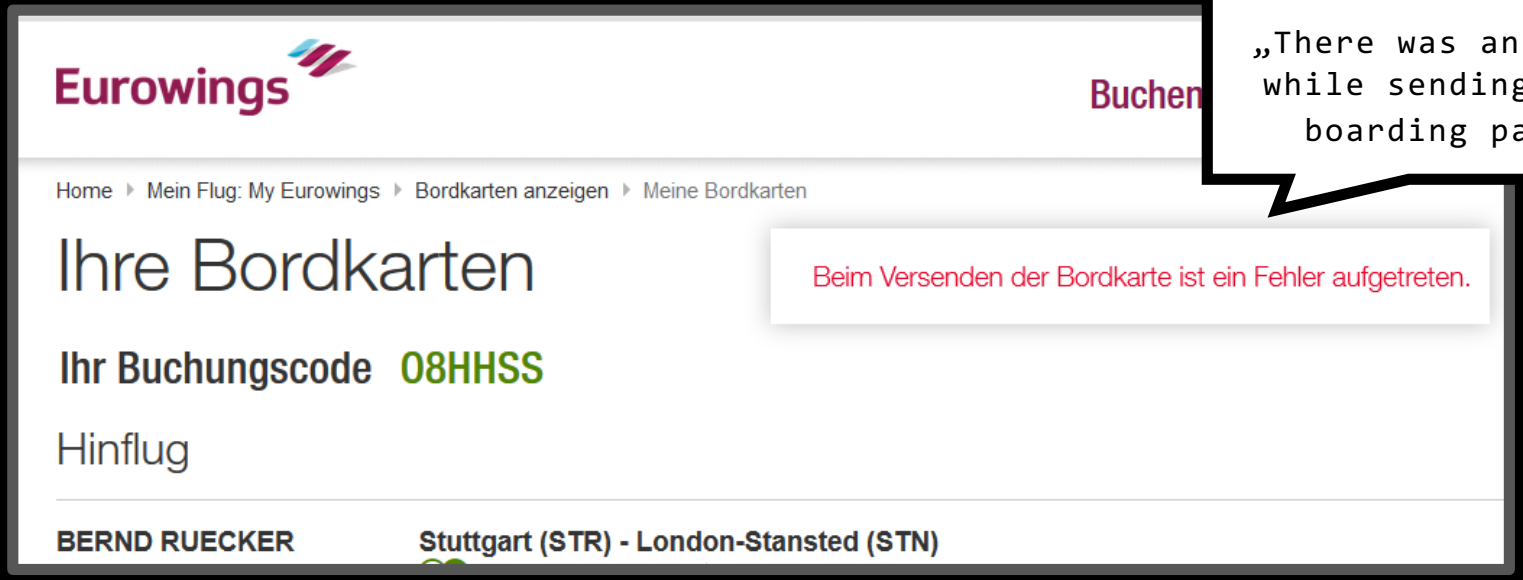

Live demo
https://github.com/flowing/flowing-retail/tree/master/rest

Circuit Breaker

Live demo

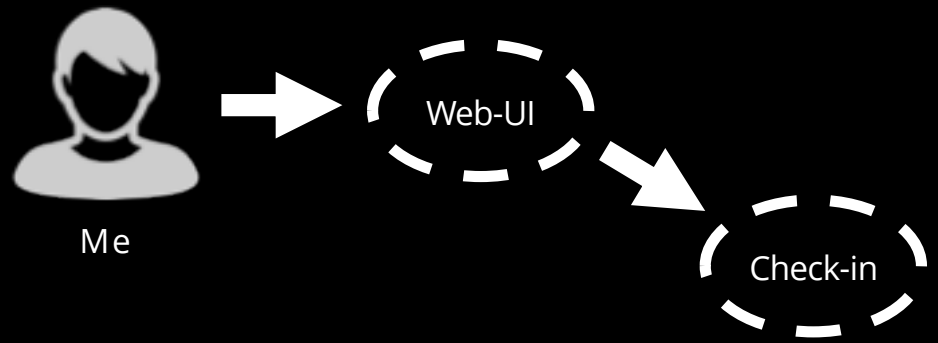
https://github.com/flowing/flowing-retail/tree/master/rest

Fail fast
is important

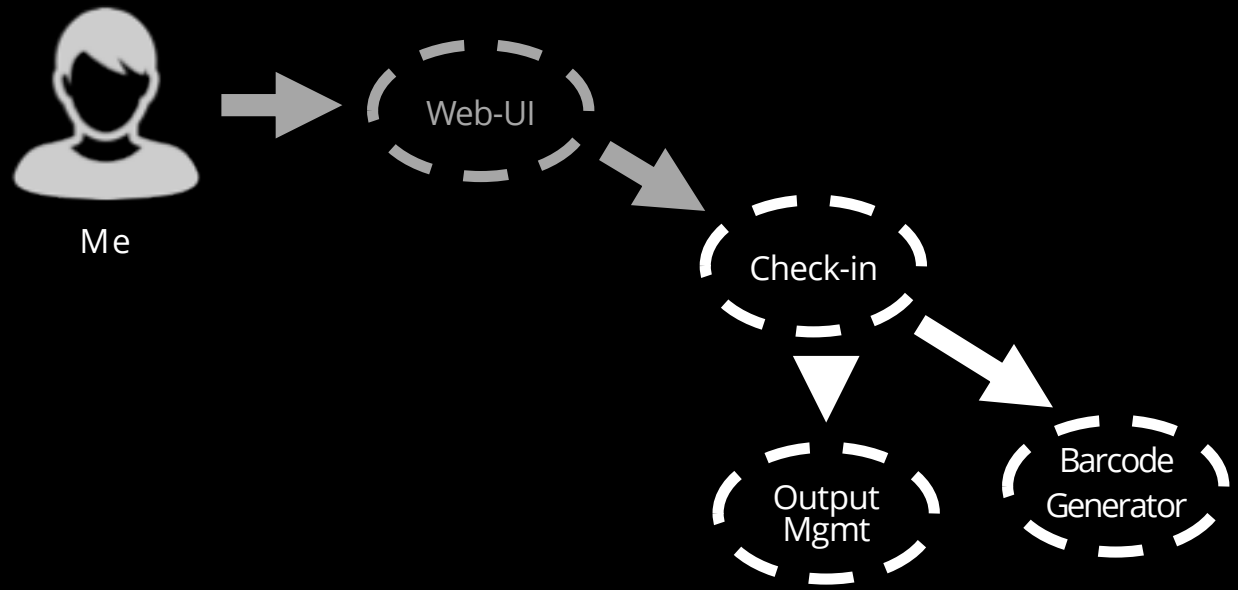


„There was an error while sending your boarding pass“

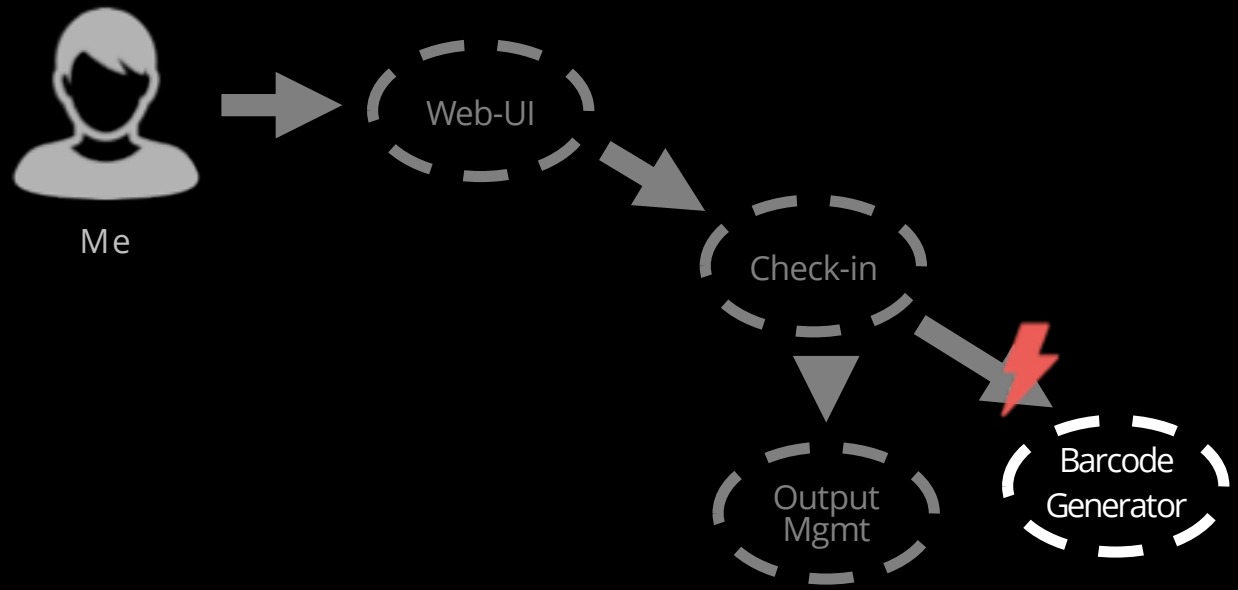
Current situation



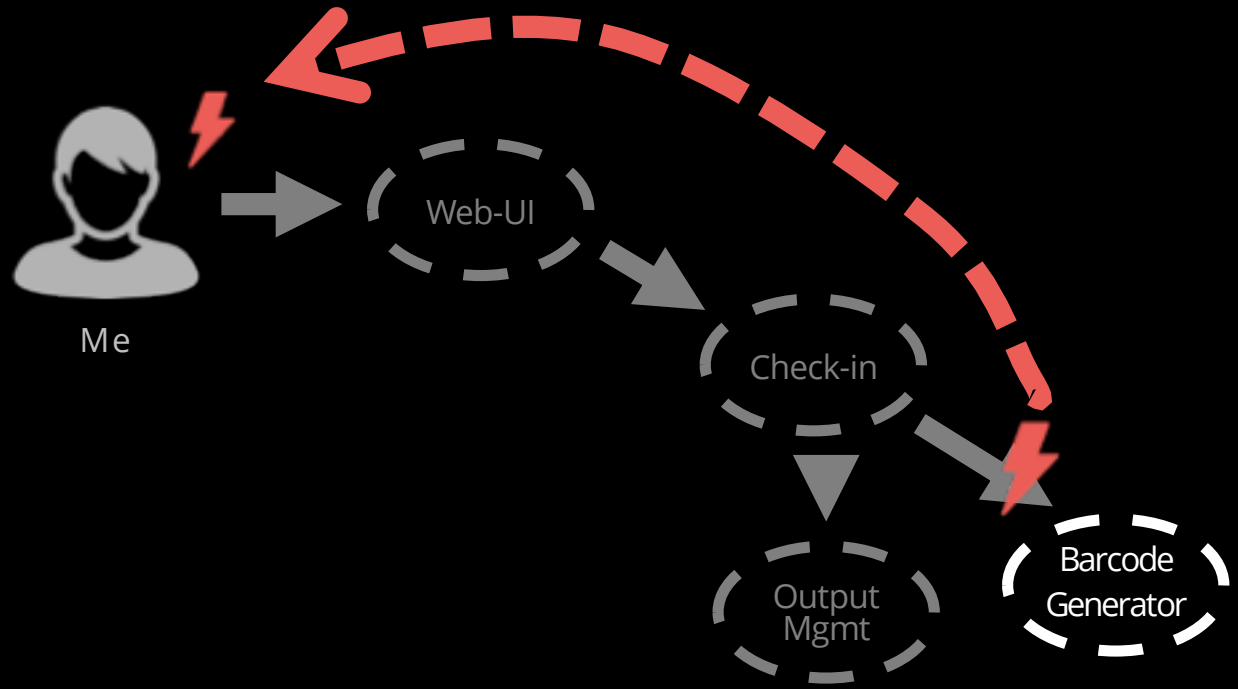
Current situation



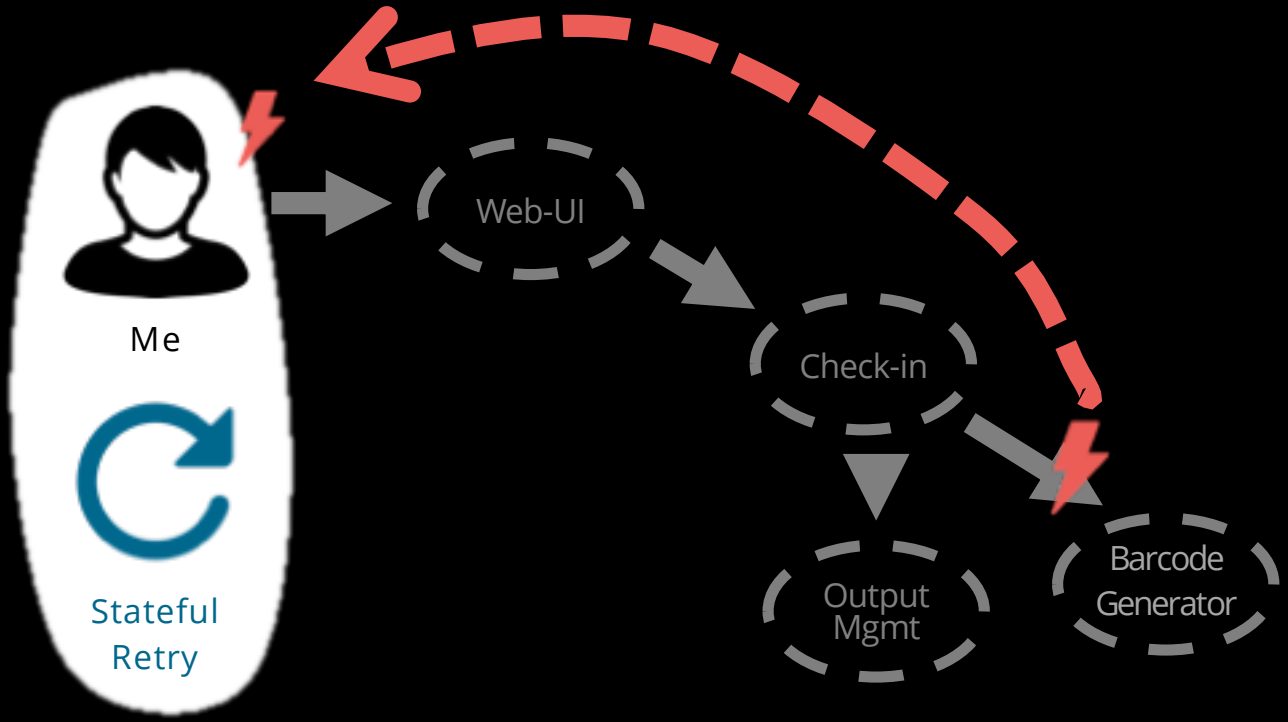
Current situation – the bad part

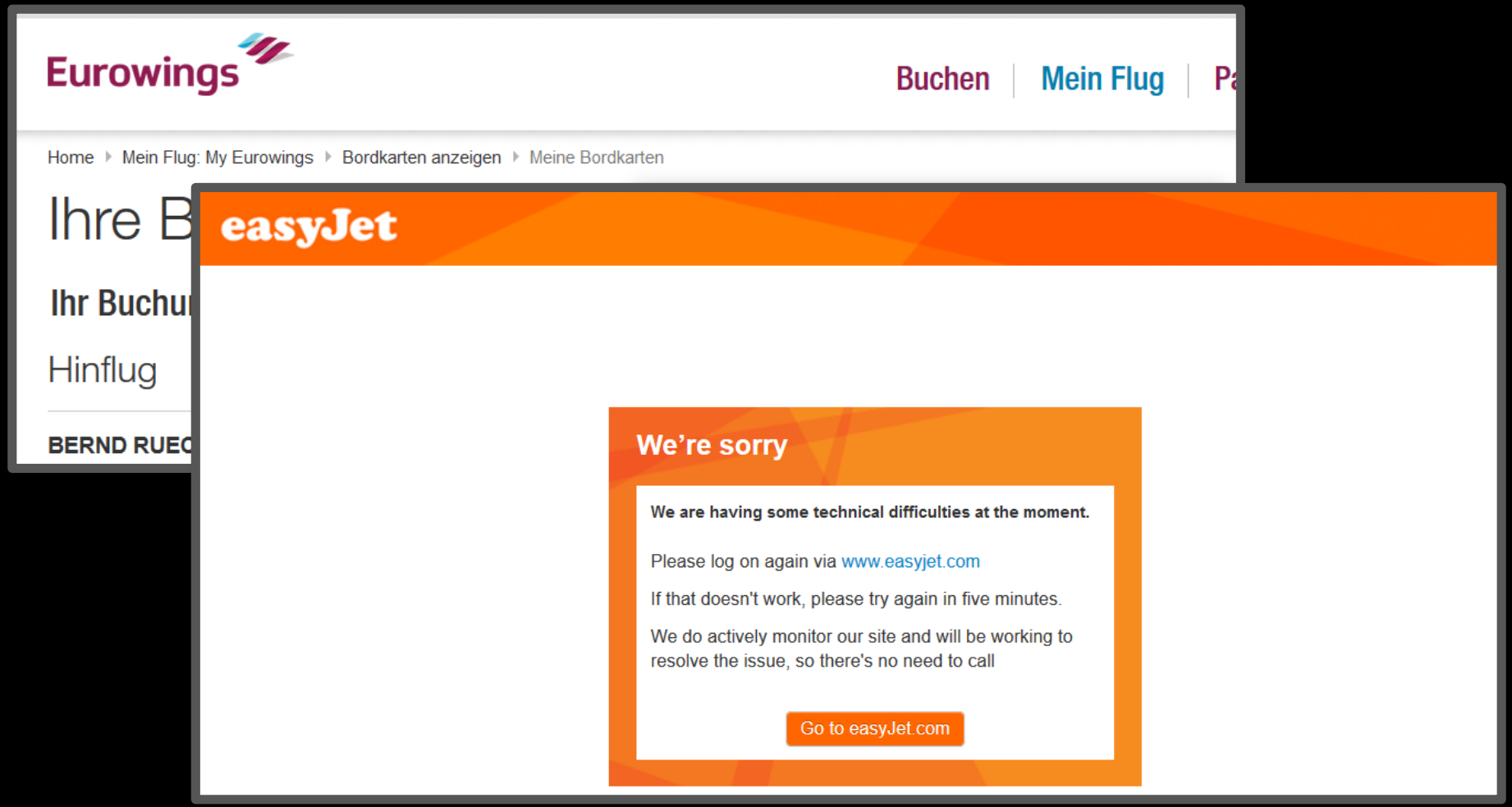


Current situation – the bad part



Current situation – the bad part




Eurowings
Buchen
Mein Flug
Home
Mein Flug: My Eurowings
Bordkarten anzeigen
Meine Bordkarten
Ihre B
Ihr Buchu
Hinflug
BERND RUEC
easyJet
We're sorry
We are having some technical difficulties at the moment.
Please log on again via www.easyjet.com
If that doesn't work, please try again in five minutes.
We do actively monitor our site and will be working to resolve the issue, so there's no need to call
Go to easyJet.com

Fail fast
is important

Fail fast
is important
but not enough!

easyJet

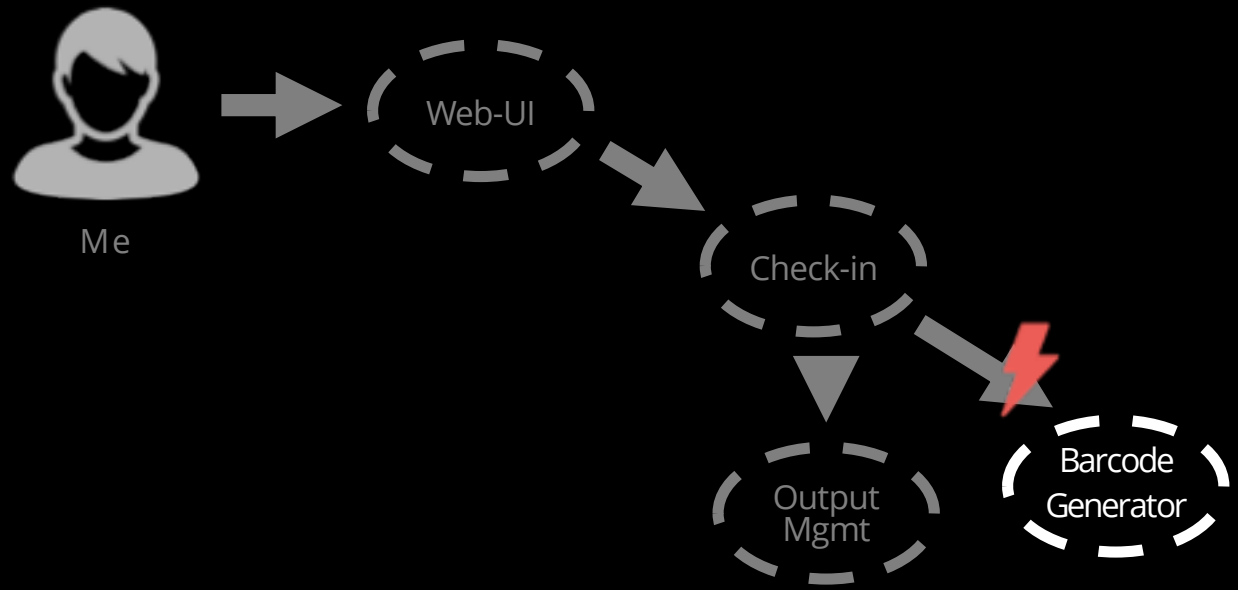
...I just made this up...

We're sorry

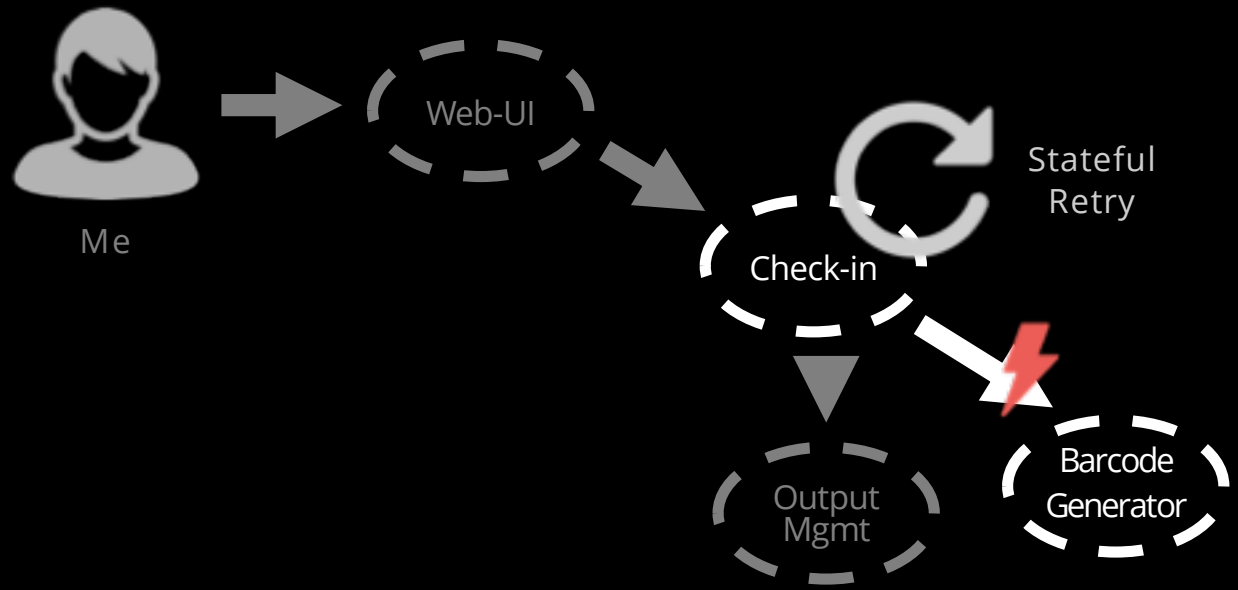
We are having some technical difficulties and cannot present you your boarding pass right away.

But we do actively retry ourselves, so lean back, relax and we will send it on time.

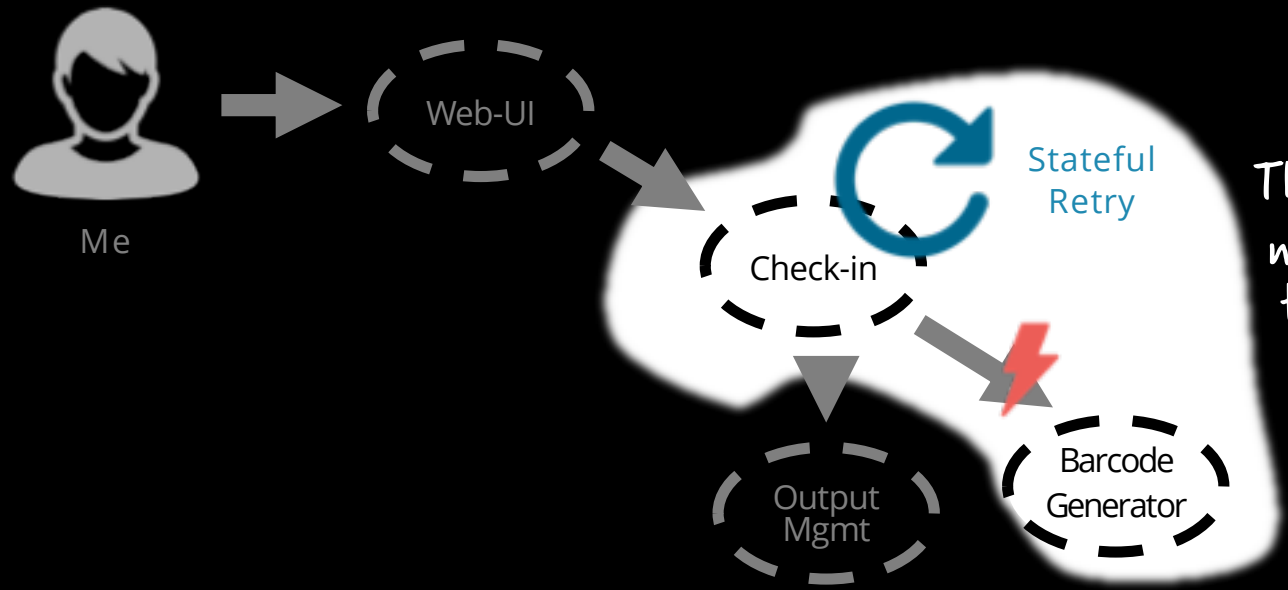
Possible situation – much better!



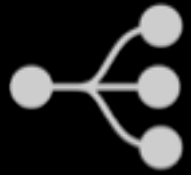
Possible situation – much better!



Possible situation – much better!



Handling State



Persist thing
(Entity, Document, Actor, ...)

State machine or
workflow engine

Typical concerns

DIY = effort, accidental complexity

Scheduling, Versioning, operating, visibility, scalability, ...

Complex, proprietary, heavyweight, slow, deployment, developer adverse

Workflow engines,
state machines

It is
relevant
in modern
architectures

Workflow engines,
state machines

UBER CADENCE

AWS Step Functions

Silicon valley
has recognized

conductor
NETFLIX OSS

Workflow engines,
state machines

UBER CADENCE

There are
lightweight open source
options

zeebe
by Camunda

AWS Step
Functions

camunda

jBPM

Activiti

conductor
NETFLIX OSS

Workflow engines,
state machines

UBER CADENCE

also at scale

zeebe
by Camunda

AWS Step Functions

camunda

jBPM

conductor
NETFLIX OSS

Activiti

Workflow engines,
state machines

UBER CADENCE

for todays demo

zeebe by Camunda

AWS Step Functions

camunda

jBPM

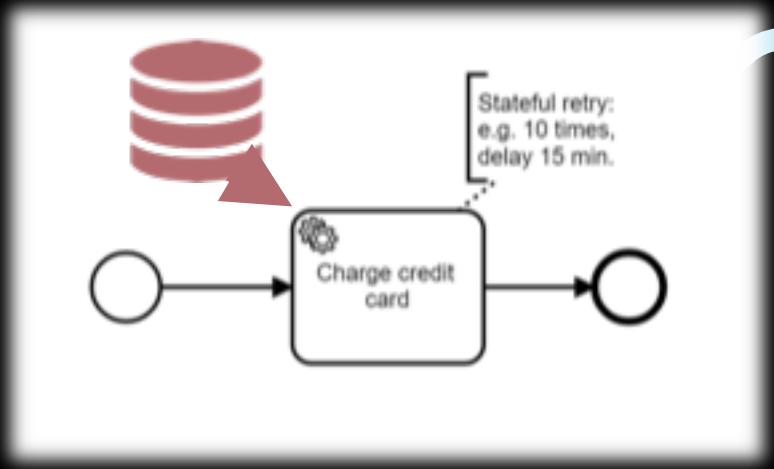
conductor
NETFLIX OSS

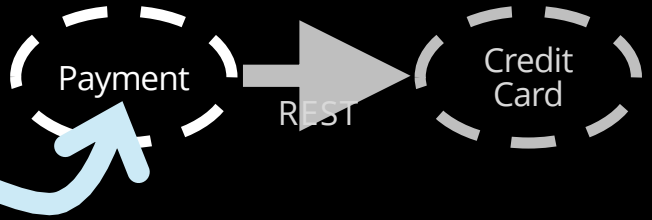
Activiti

Live demo

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

Now you have a state machine!



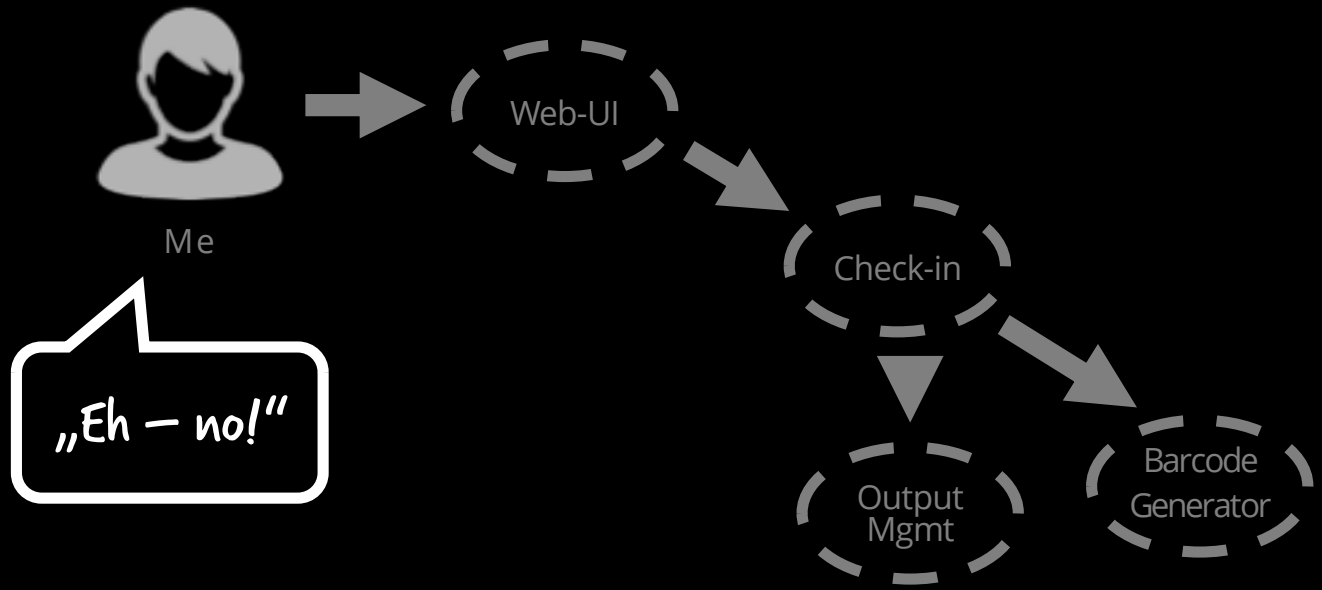


Client

has to implement

Retry

„The customer wants a synchronous response“

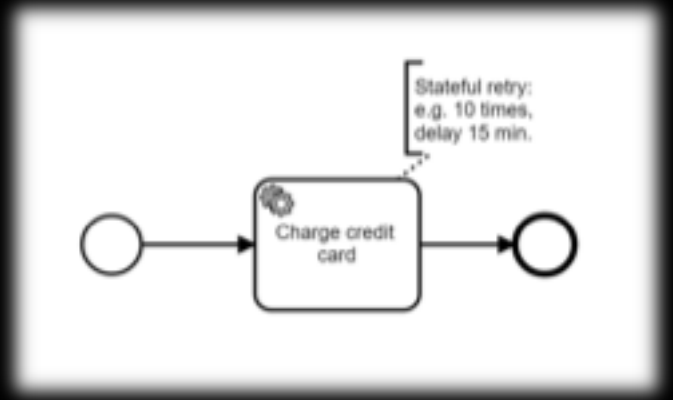


Payment

createPayment

HTTP 200 OK

HTTP 202 ACCEPTED



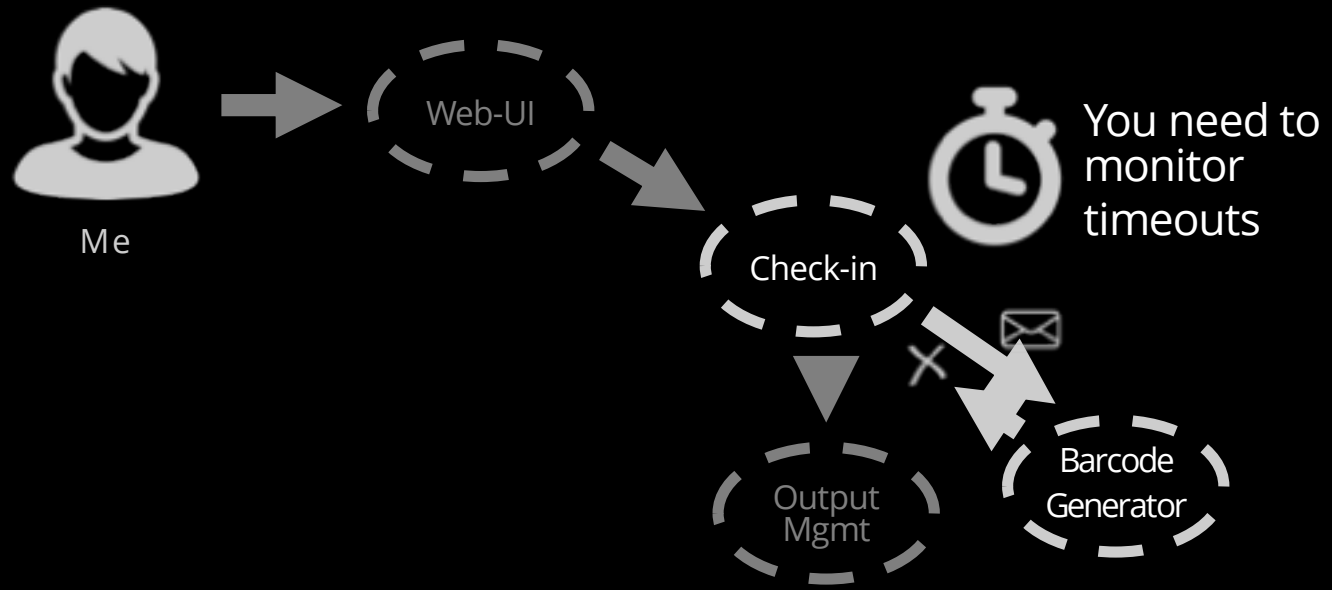
A synchronous response is possible in the happy case, otherwise it is switched to asynchronous processing.


Live demo
https://github.com/flowing/flowing-retail/tree/master/rest

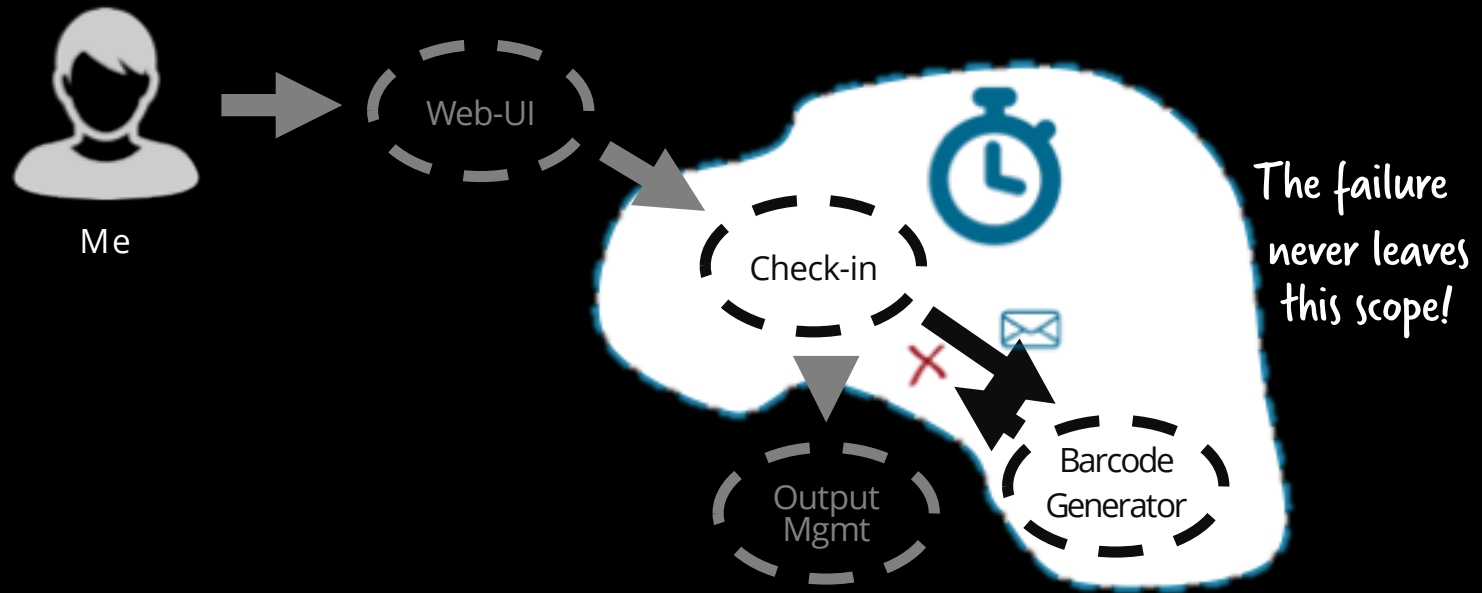
Synchronous communication is the crystal meth of distributed programming

Todd Montgomery and Martin Thompson
in "How did we end up here" at GoTo Chicago 2015

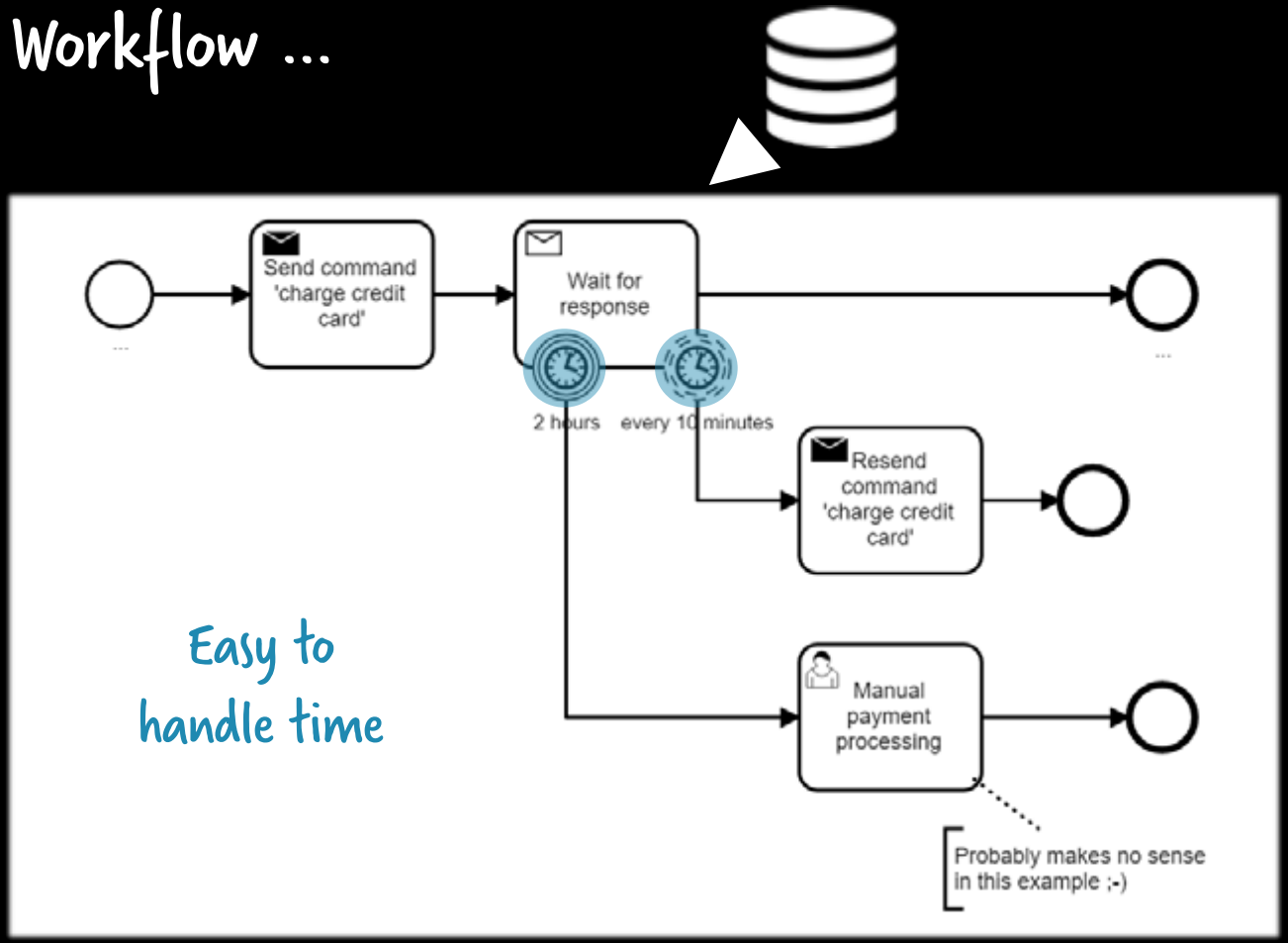
Asynchronous communication



Remember...


Me
Web-UI
Check-in
Output Mgmt
Barcode Generator
The failure never leaves this scope!

Workflow ...



Easy to handle time

BPMN

Business Process Model and Notation

ISO Standard

Client

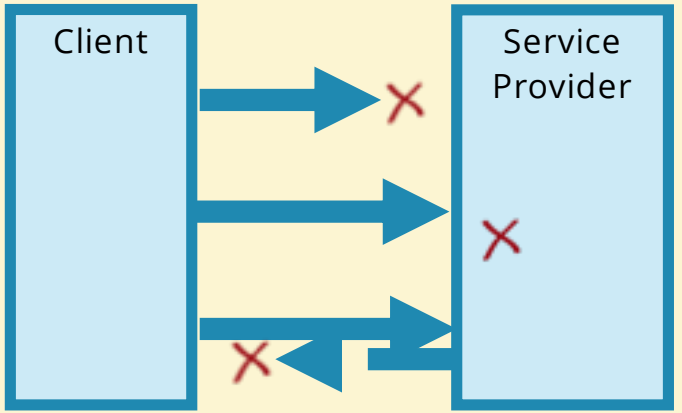
has to implement

Retry,
Timeout

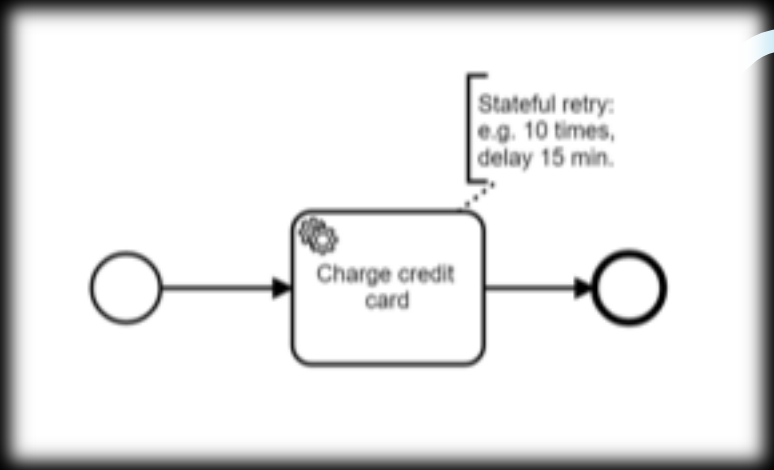
Distributed systems

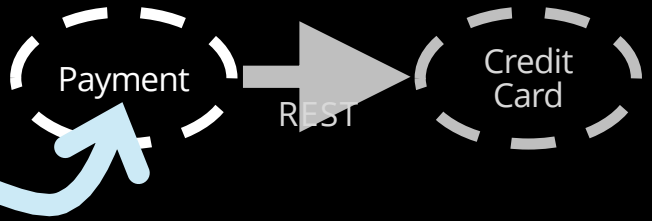
It is impossible to differentiate certain failure scenarios.

Independent of communication style!



What does it mean?





Duplicates
Duplicates

Client

has to implement

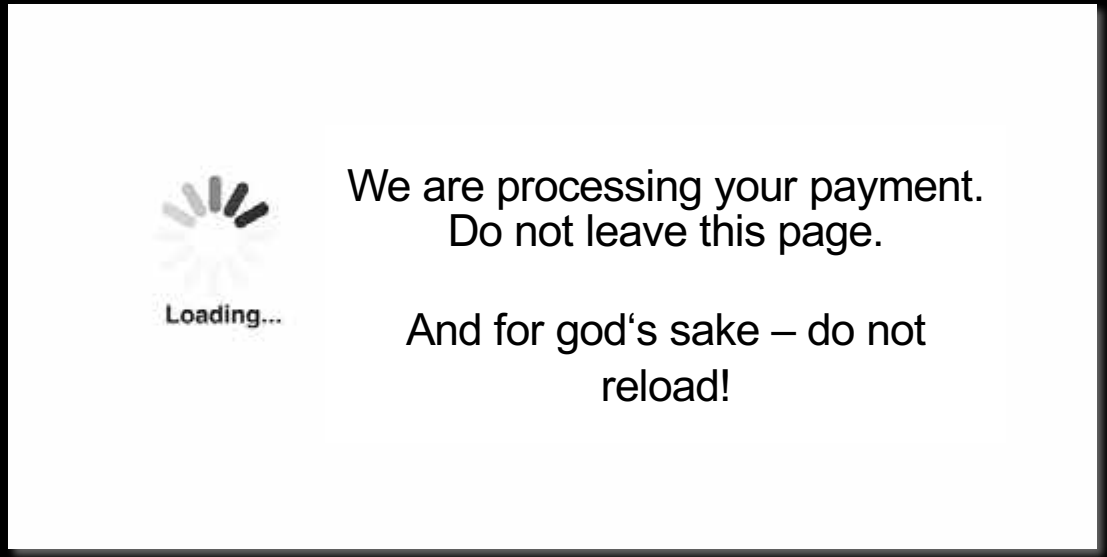
Retry,
Timeout

Service Provider

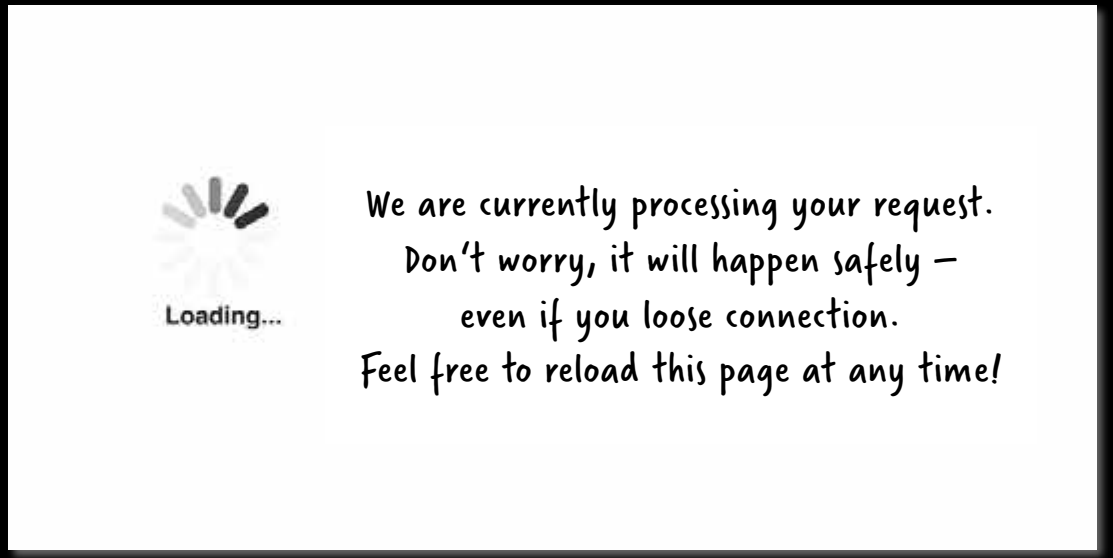
has to implement

Idempotency

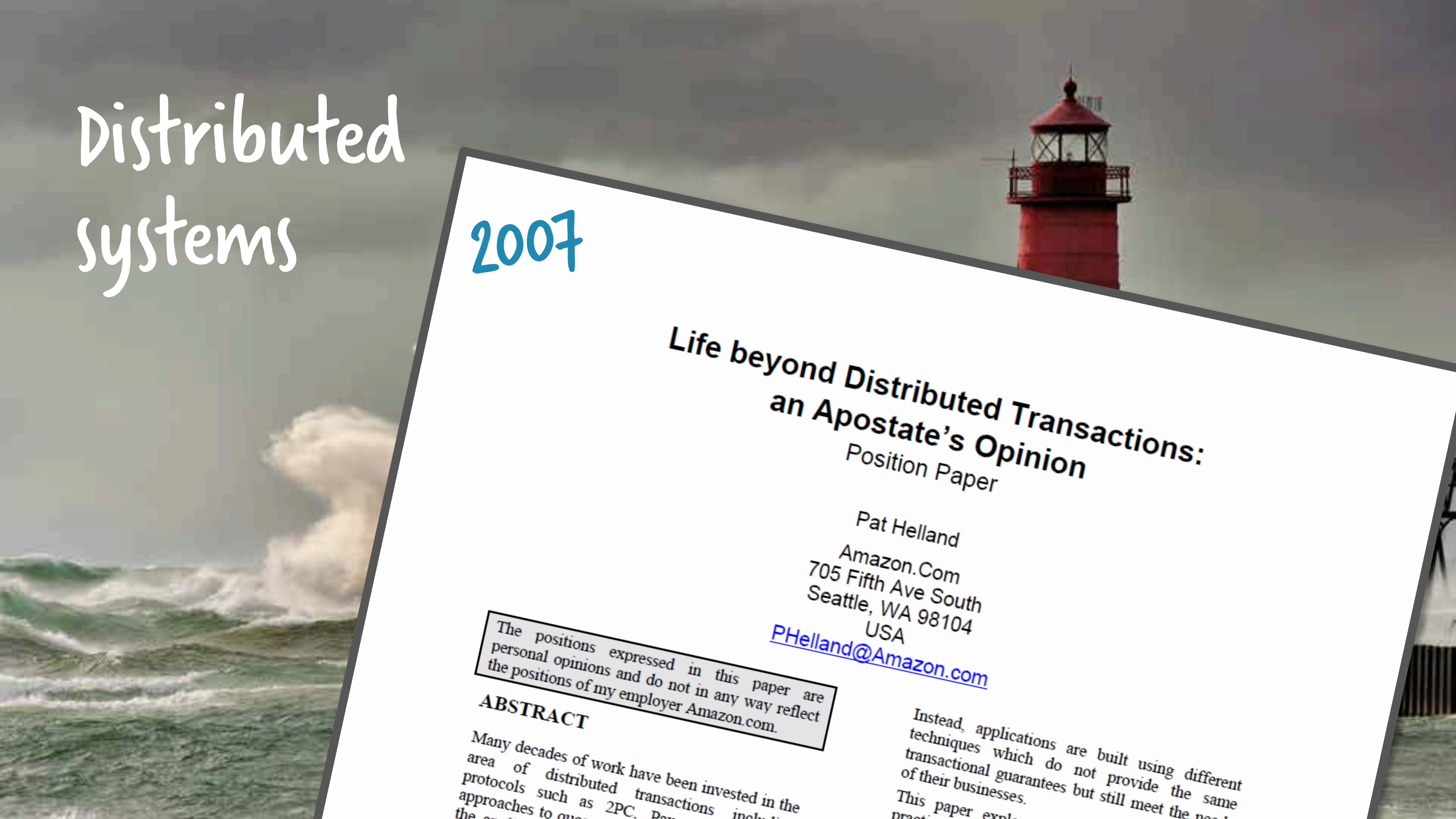
Idempotency is a business problem!



Idempotency is a business problem!



But we wanted to
talk about consistency!



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.

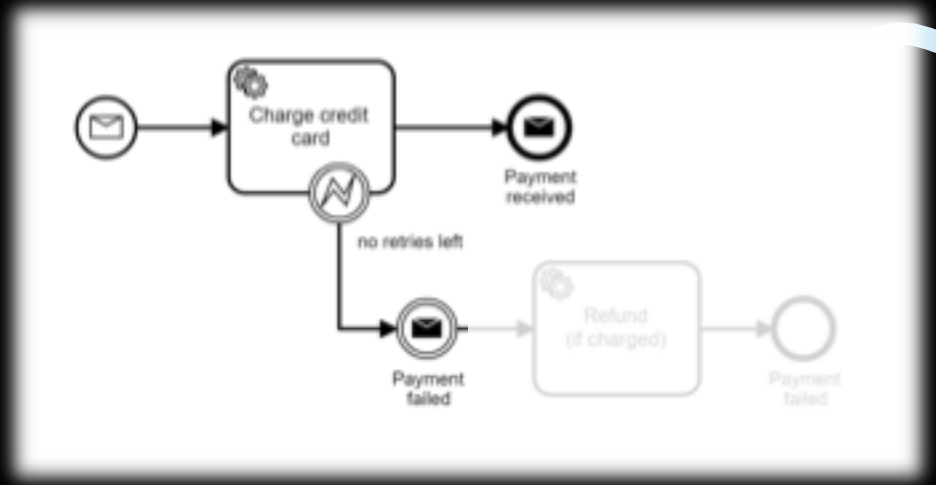
ABSTRACT

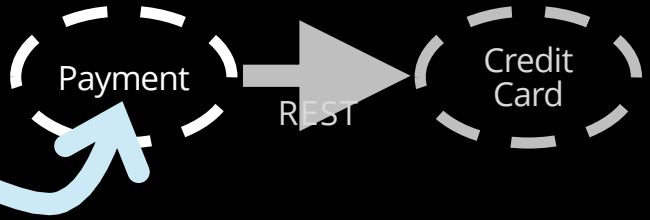
Many decades of work have been invested in the area of distributed transactions ... protocols such as 2PC, ... approaches to ... the ...

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

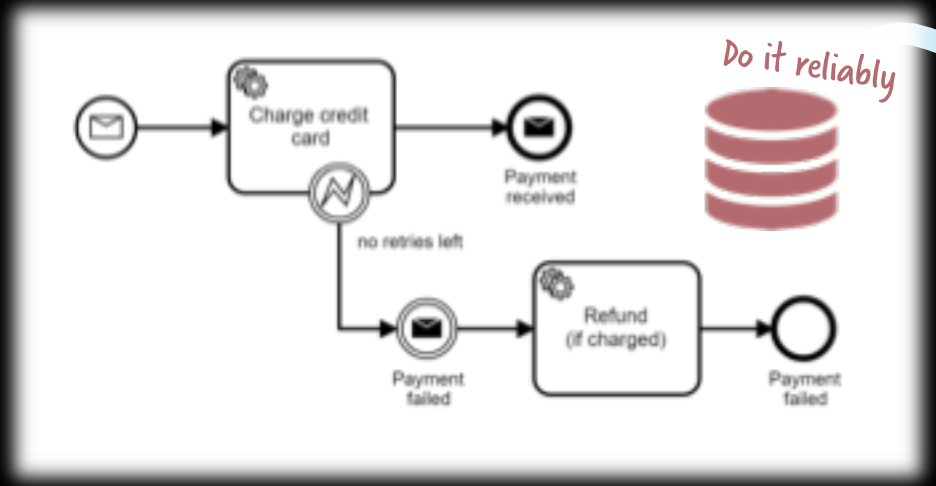
This paper ...

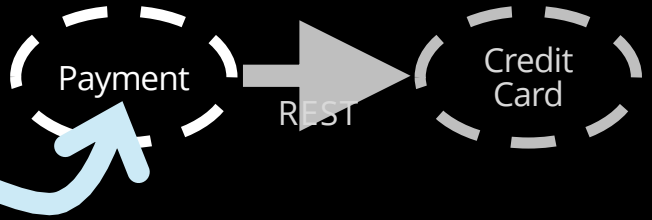
Distributed systems introduce complexity you have to tackle!





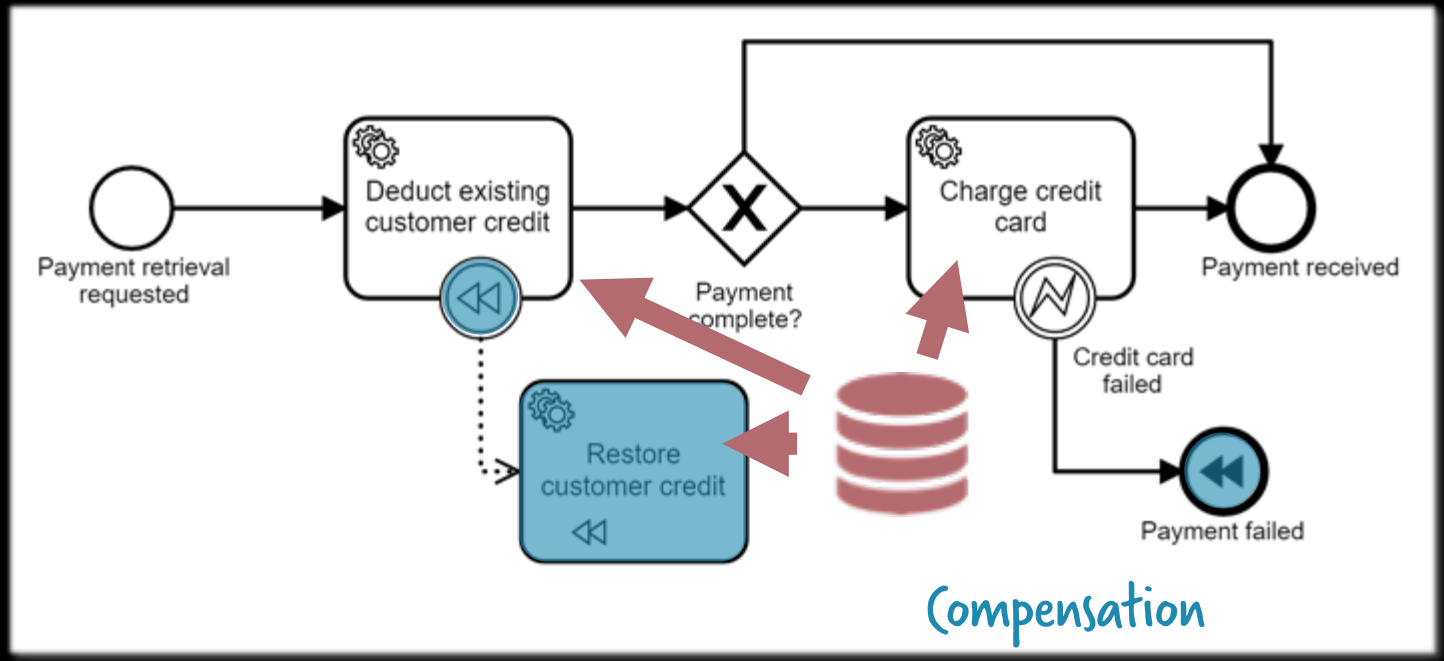
Distributed systems introduce complexity you have to tackle!





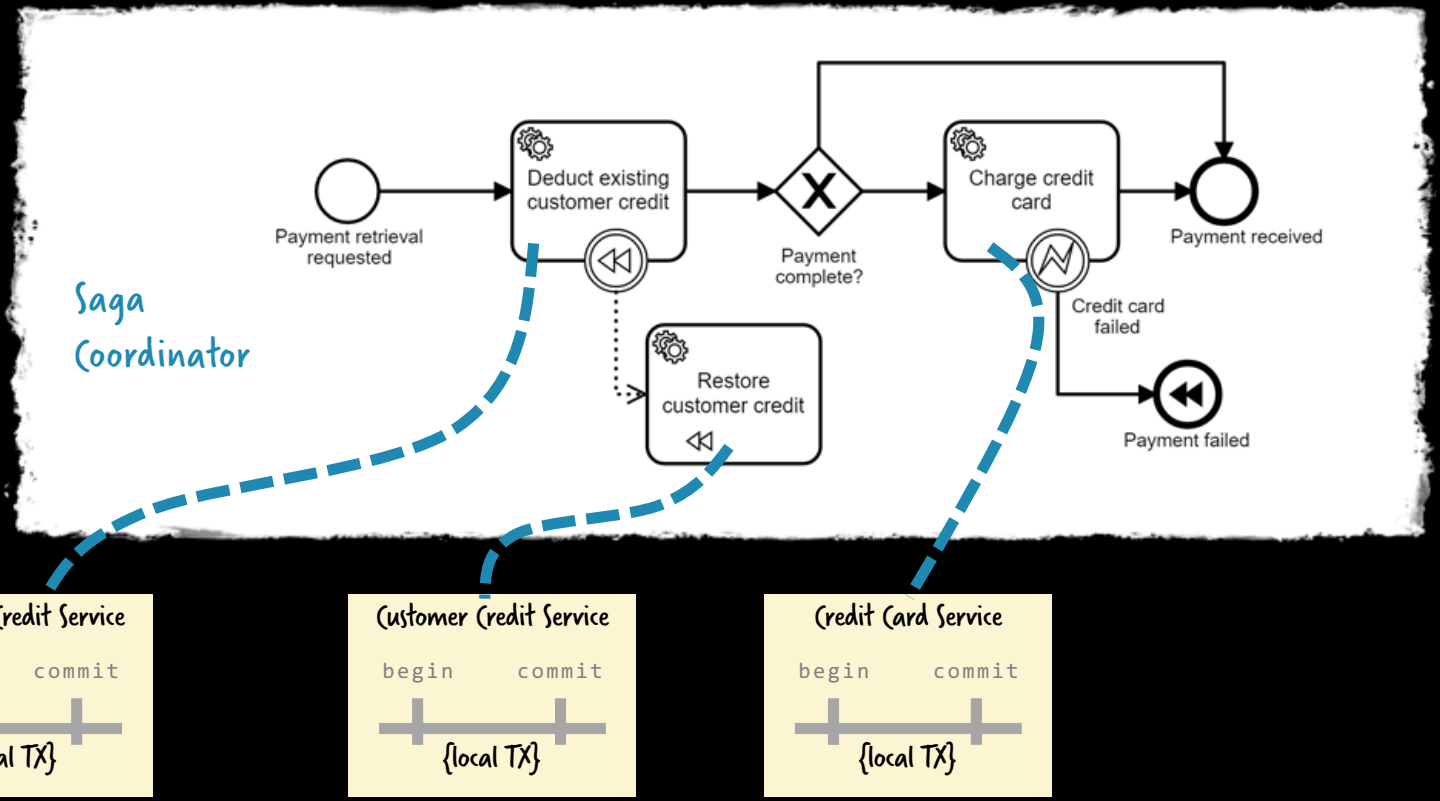
Distributed transactions using compensation *

*aka Saga pattern

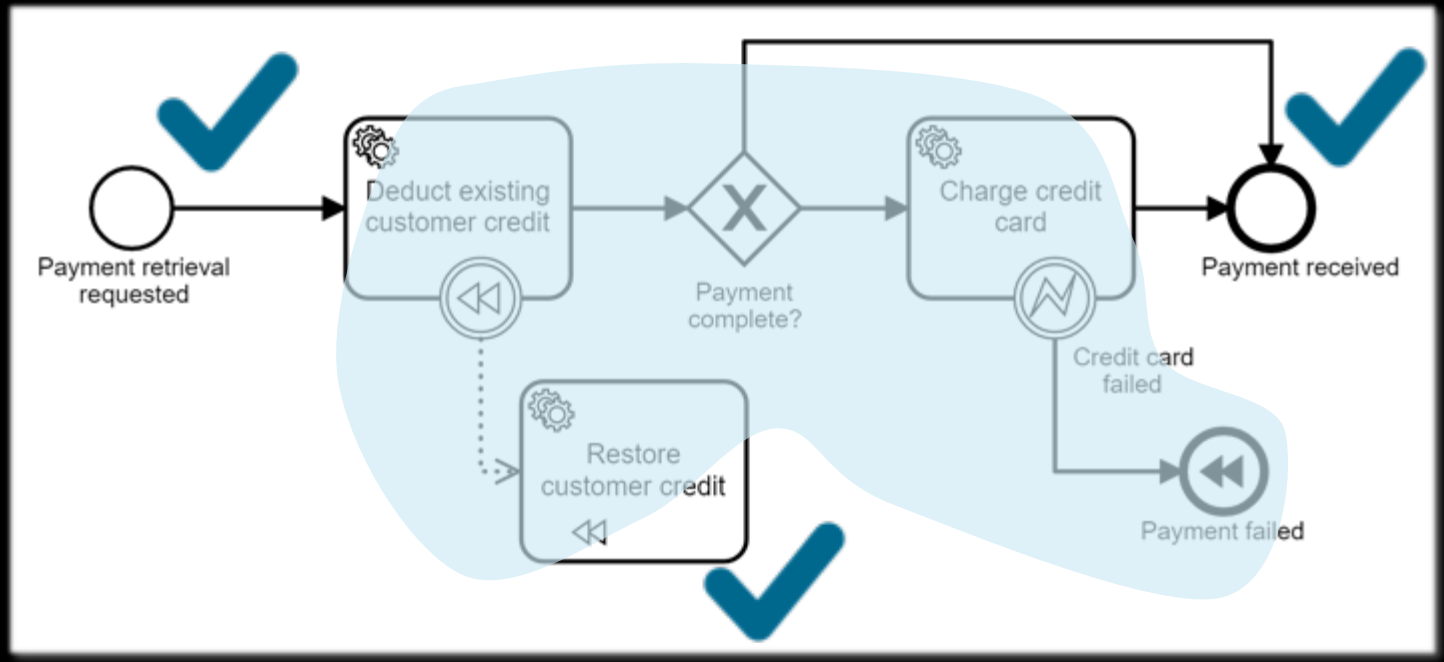



Live demo
https://github.com/flowing/flowing-retail/tree/master/rest

A workflow engine can serve as Saga coordinator



Eventual consistency



Temporarily inconsistent state

But only temporary

No Isolation (as in ACID)

Apologize!

https://blogs.msdn.microsoft.com/pathelland/2007/05/15/memories-guesses-and-apologies/

Client

has to implement

Timeout, Retry, Compensation

Service Provider

has to offer

Compensation

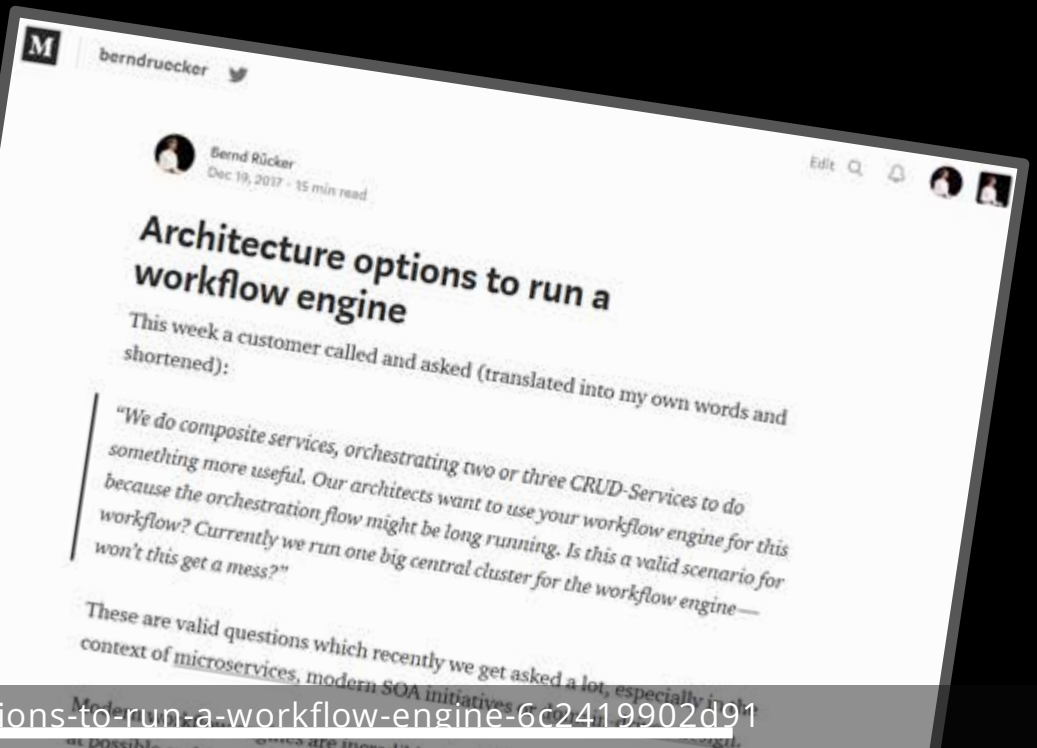
has to implement

Idempotency

And: don't forget to apologize sometimes...

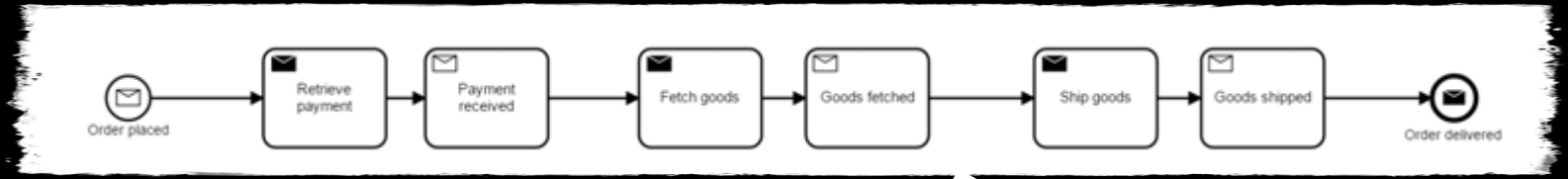

A few words
about architecture ...

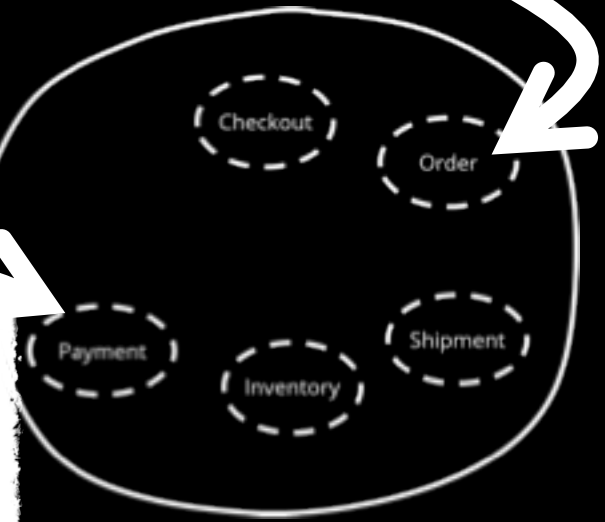
Architecture

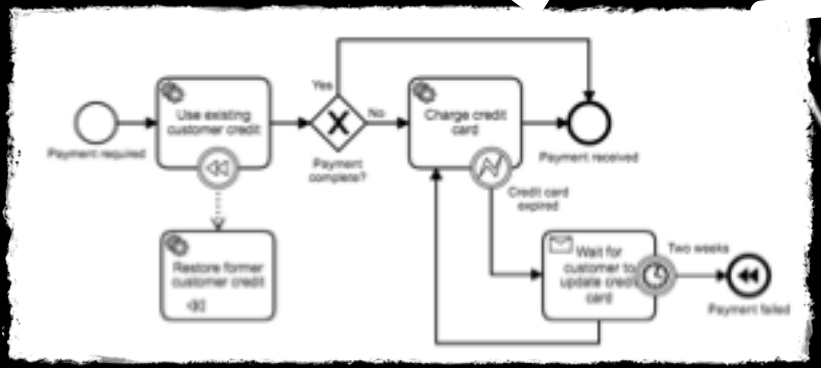


https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

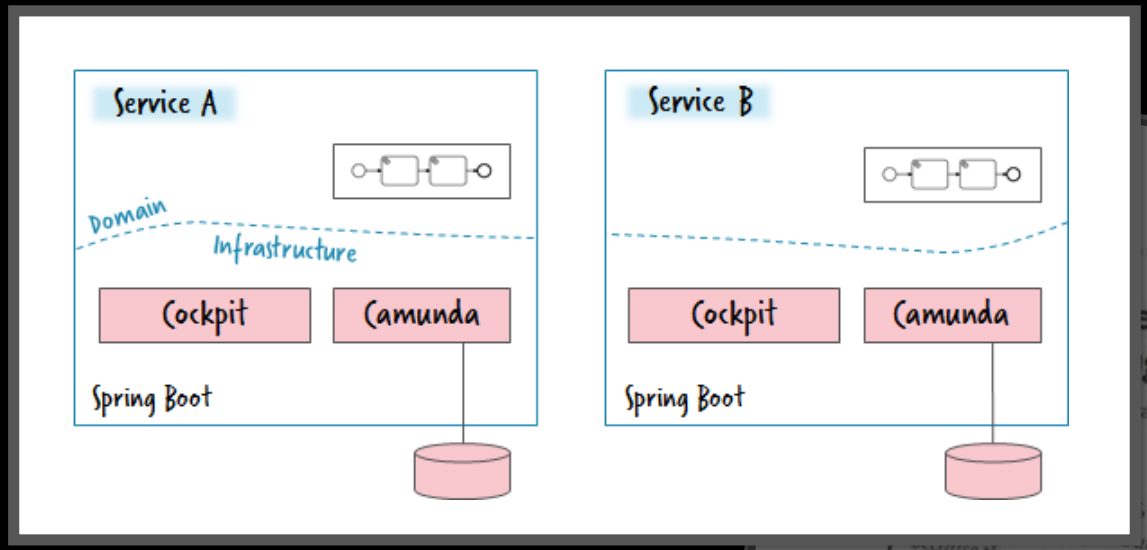
Workflows live inside service boundaries

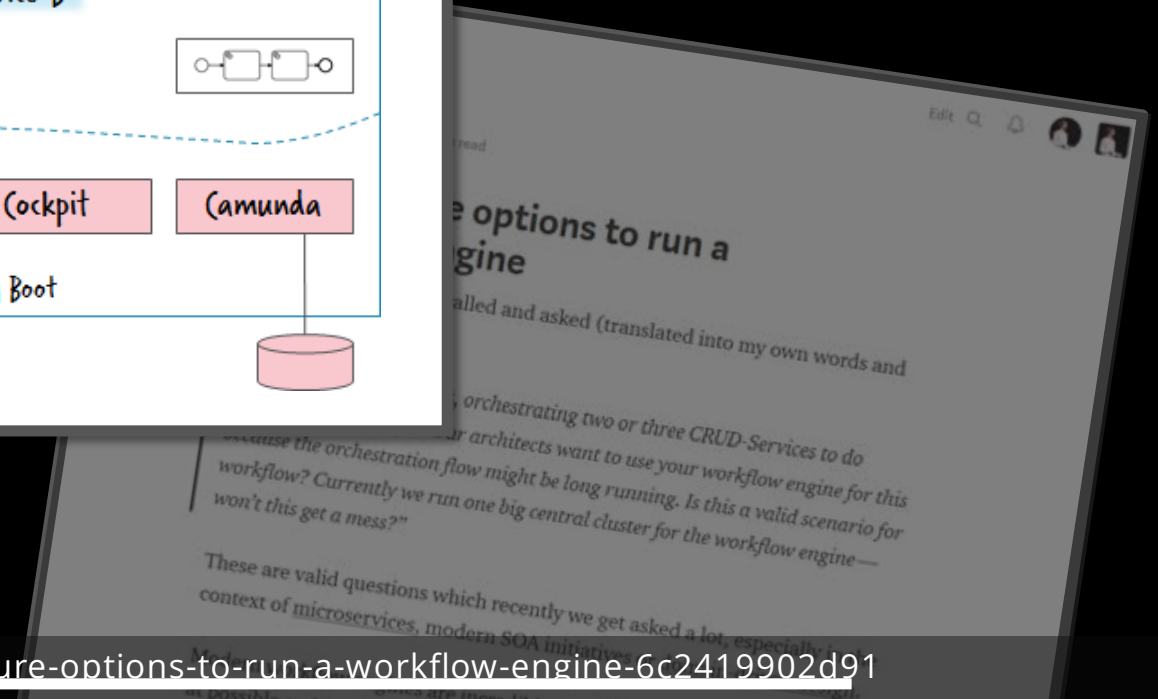






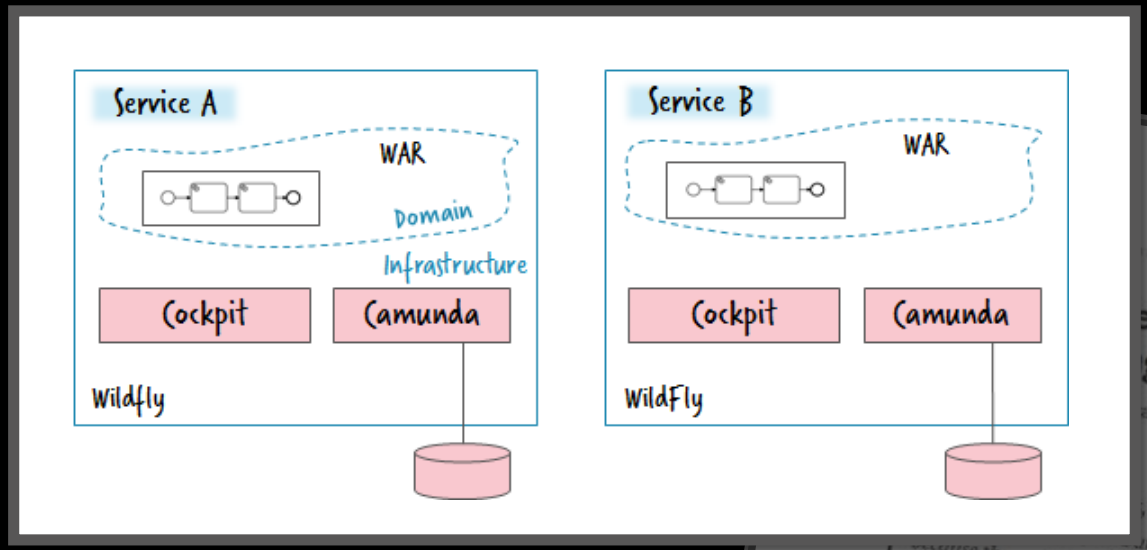
Architecture

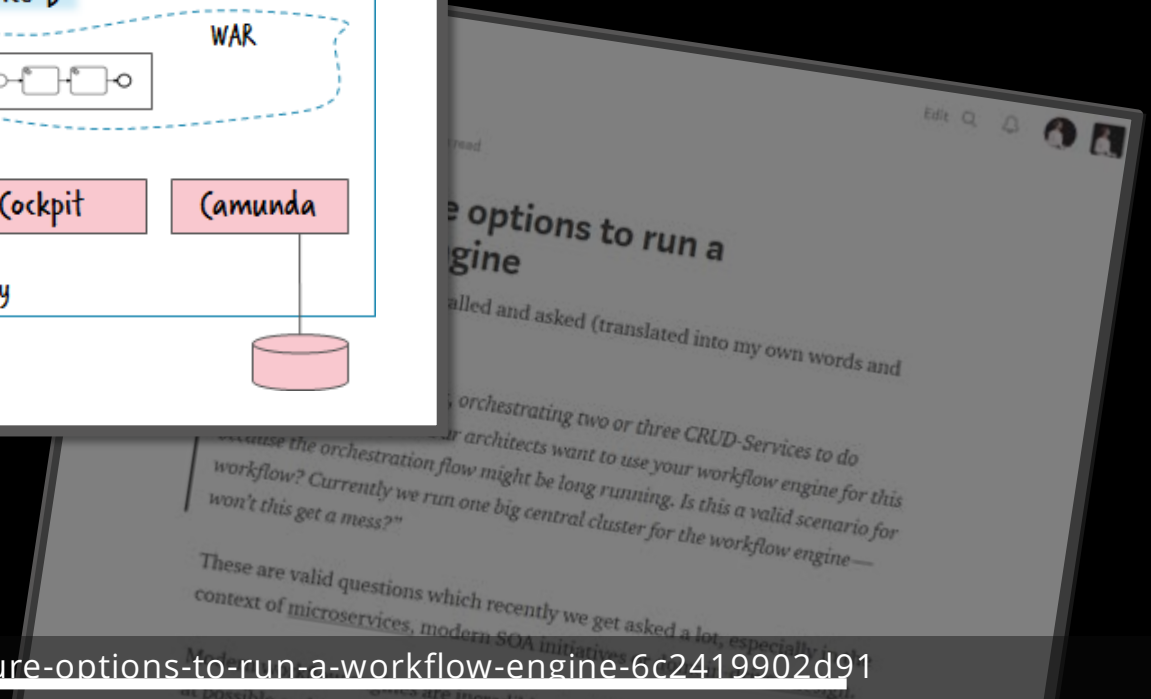




https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

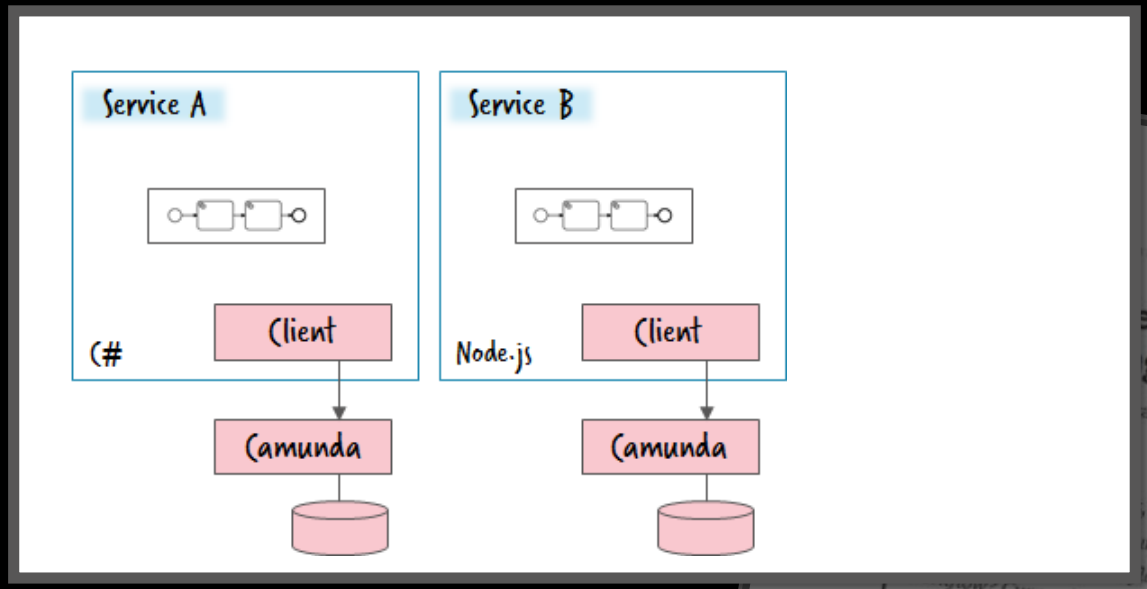
Architecture





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

Architecture



options to run a
gine

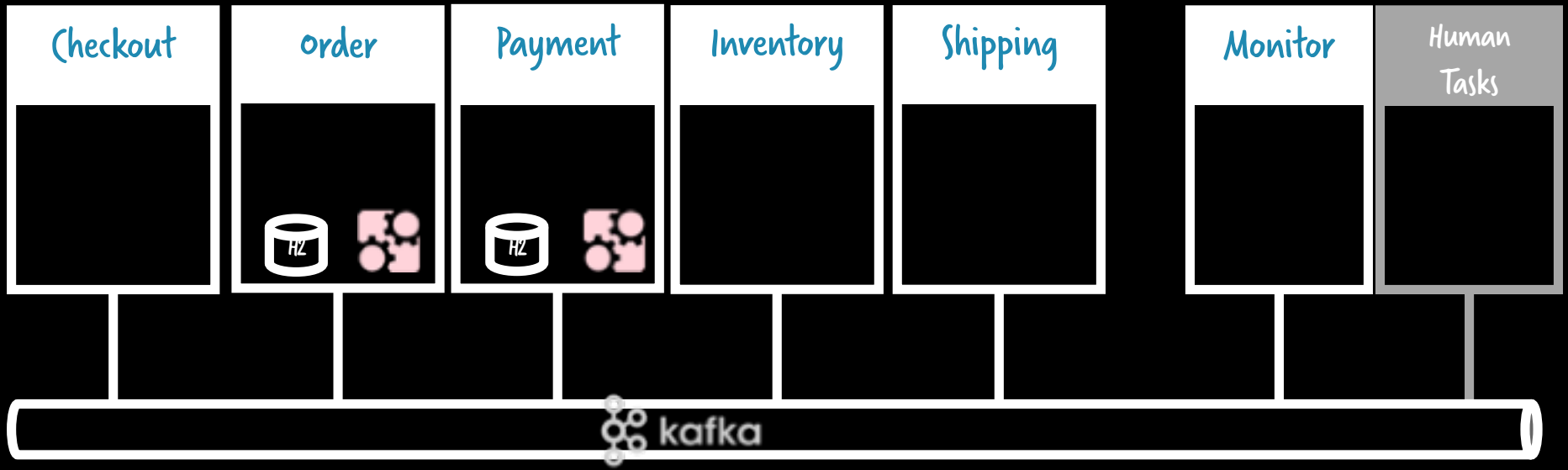
alled and asked (translated into my own words and

orchestrating two or three CRUD-Services to do
r architects want to use your workflow engine for this
low might be long running. Is this a valid scenario for
won't this get a mess?"

These are valid questions which recently we get asked a lot, especially

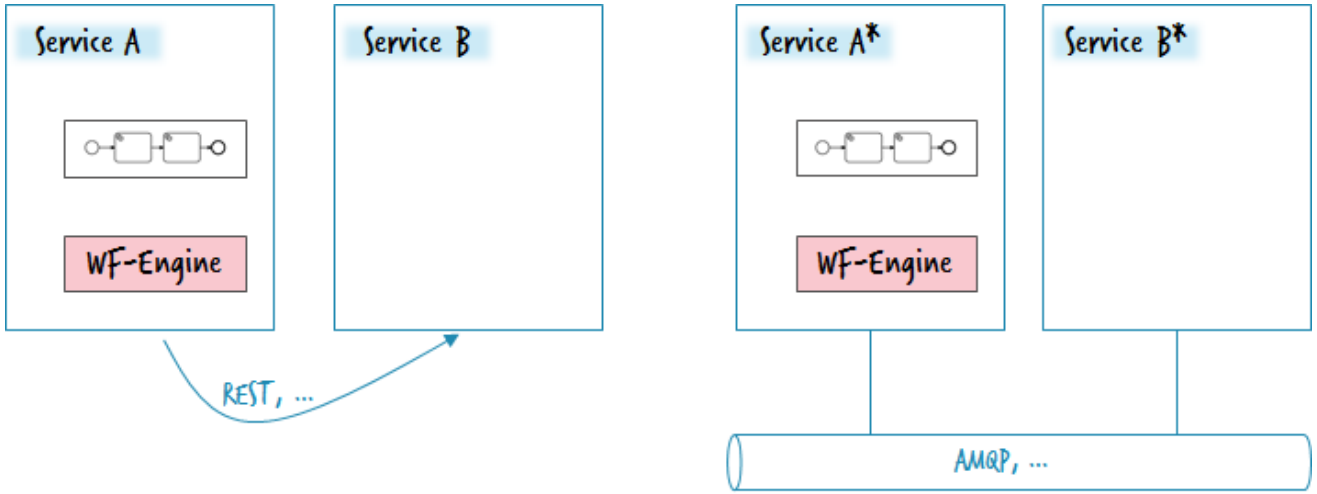
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Event-driven example also available



https://github.com/flowing/flowing-retail/

Architecture



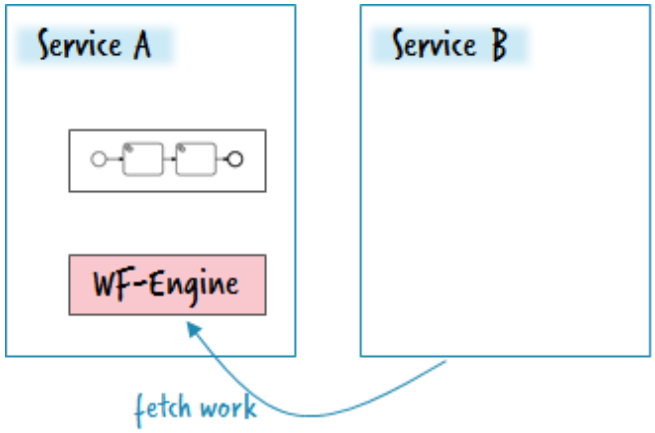
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Who uses a message bus?

Who has no problems operating a message bus?

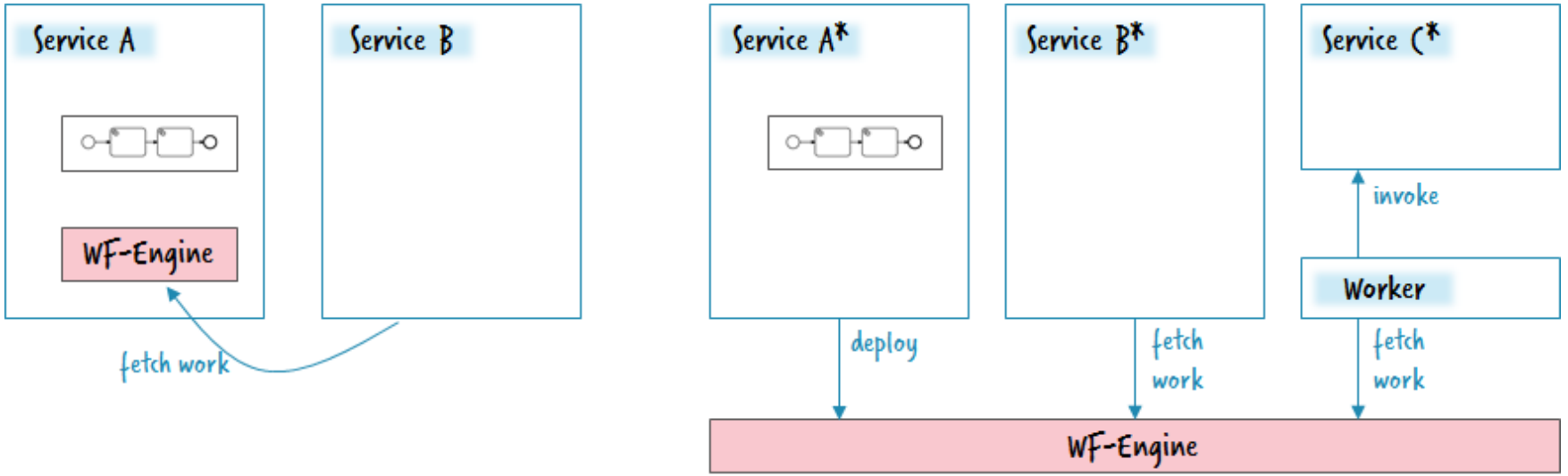
Dead messages | No context | Inaccesible payload | Hard to redeliver | Home-grown message hospitals | ...

Architecture



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

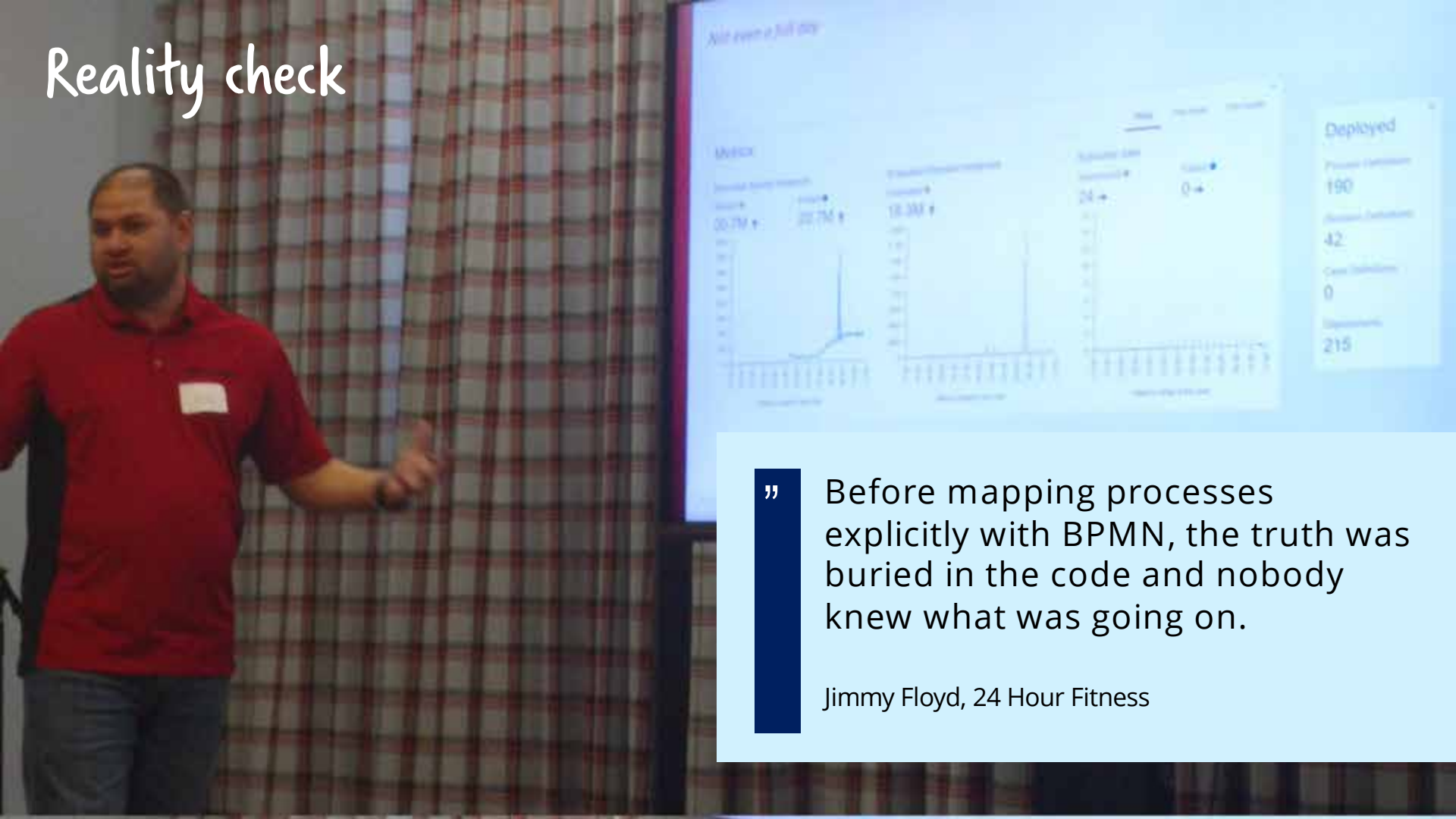
Architecture



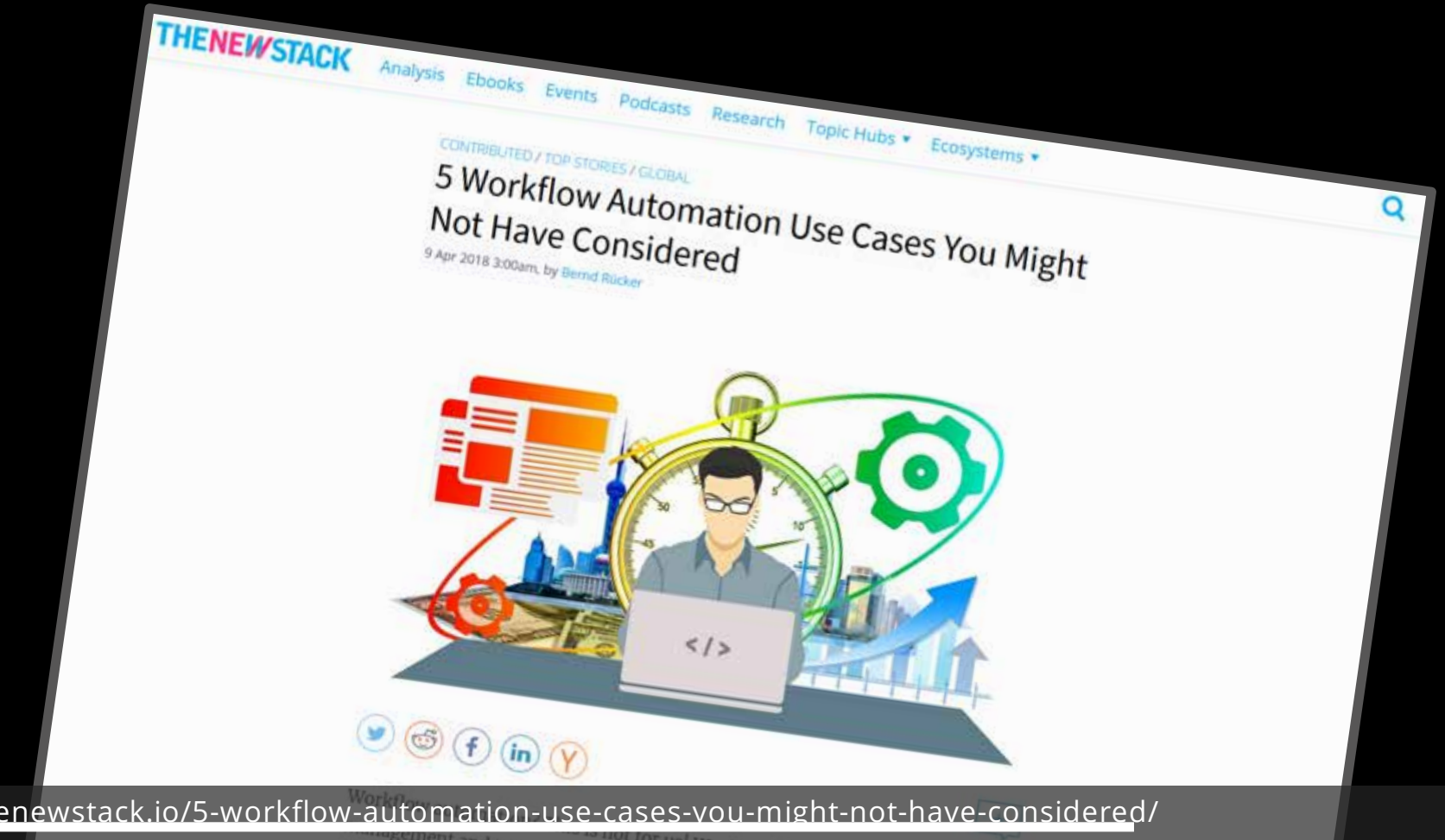
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91


Live demo
https://github.com/flowing/flowing-retail/tree/master/rest

Reality check

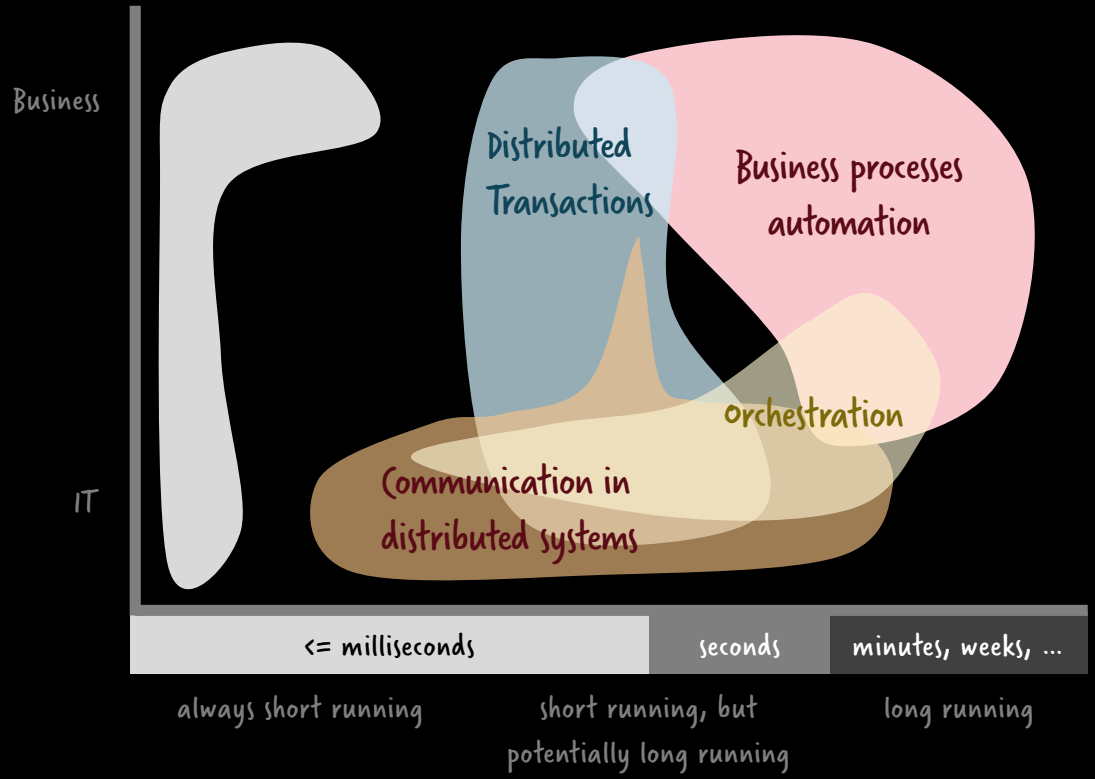


> Before mapping processes explicitly with BPMN, the truth was buried in the code and nobody knew what was going on.
>
> Jimmy Floyd, 24 Hour Fitness



https://thenewstack.io/5-workflow-automation-use-cases-you-might-not-have-considered/

Use cases for workflow automation



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)


Thank you!