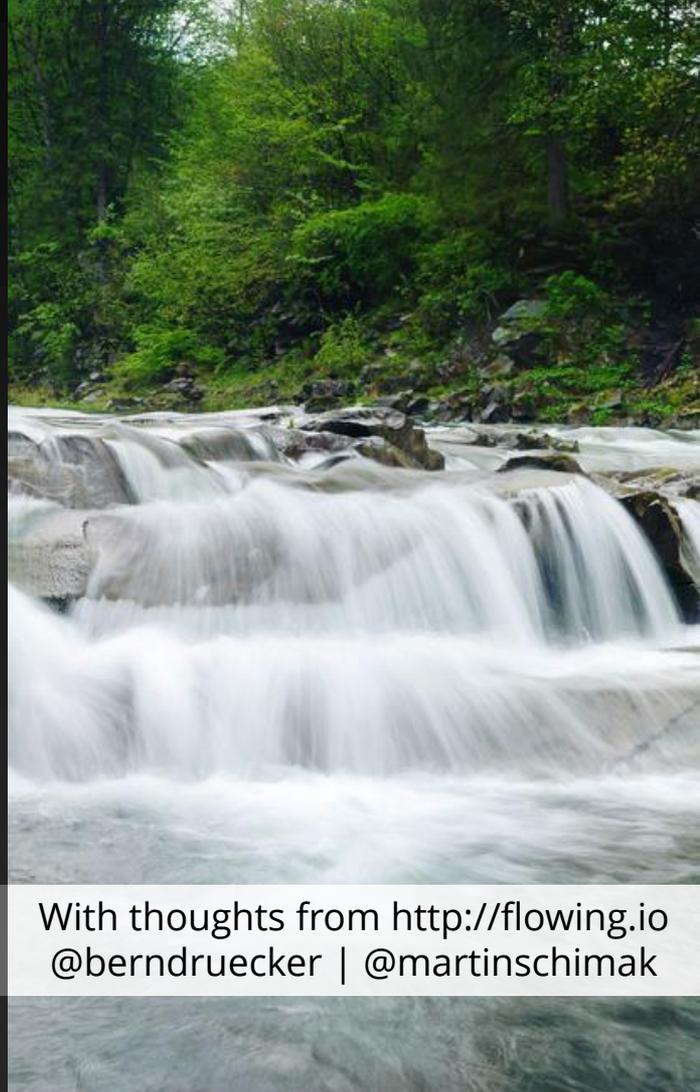


Complex event flows in distributed systems

@berndruecker

With thoughts from <http://flowing.io>
@berndruecker | @martinschimak



3 common hypotheses I check today:

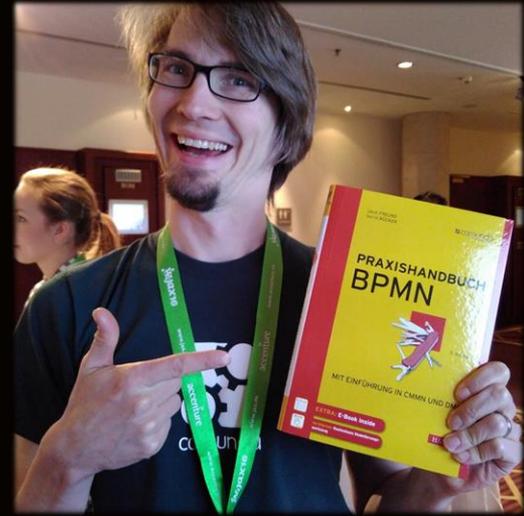
Events decrease coupling

orchestration needs to be avoided

Workflow engines are painful



Bernd Ruecker
Co-founder and
Developer Advocate of
Camunda



Berlin, Germany



bernd.ruecker@camunda.com
@berndruecker

ORCHESTRATING A
HIGHLY-SCALABLE
FULFILLMENT
PROCESS

JÖRN HORSTMANN
LUKAS NIEMEIER

2017-05-10



THE SHUTTLE
TECH INNOVATION LAB

Simplified example:
dash button



Photo by 0xF2, available under [Creative Commons BY-ND 2.0](https://creativecommons.org/licenses/by-nd/2.0/) license. <https://www.flickr.com/photos/0xf2/29873149904/>

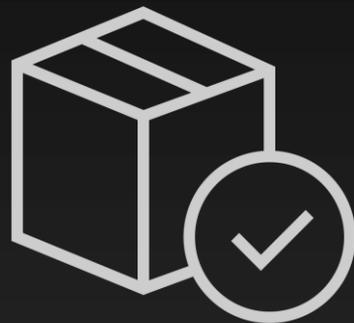
Three steps...



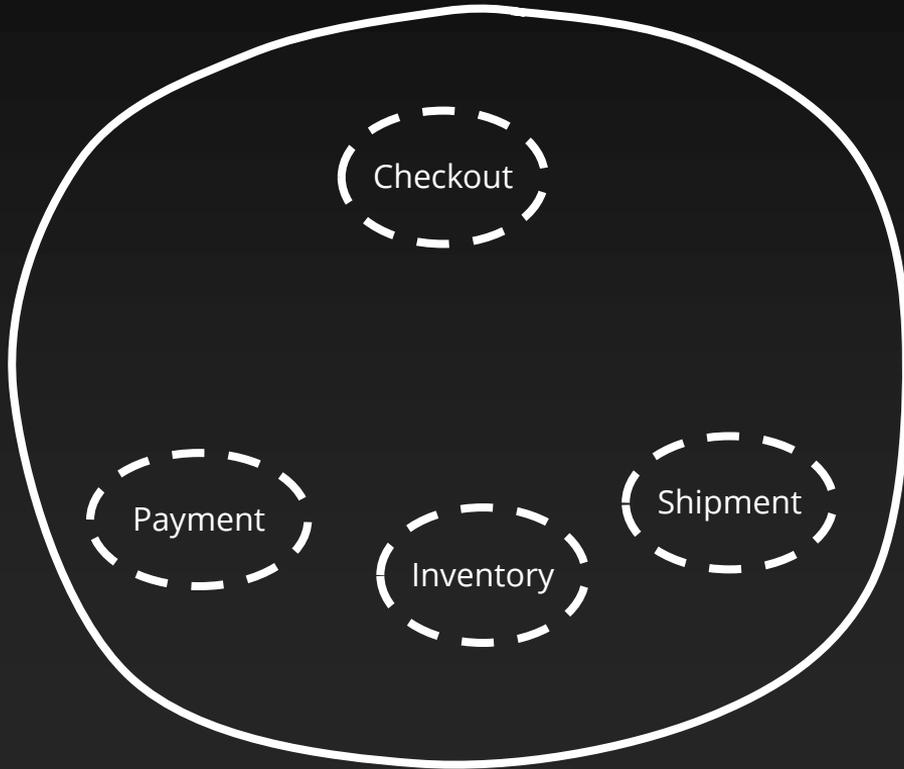
Pay item

Fetch item

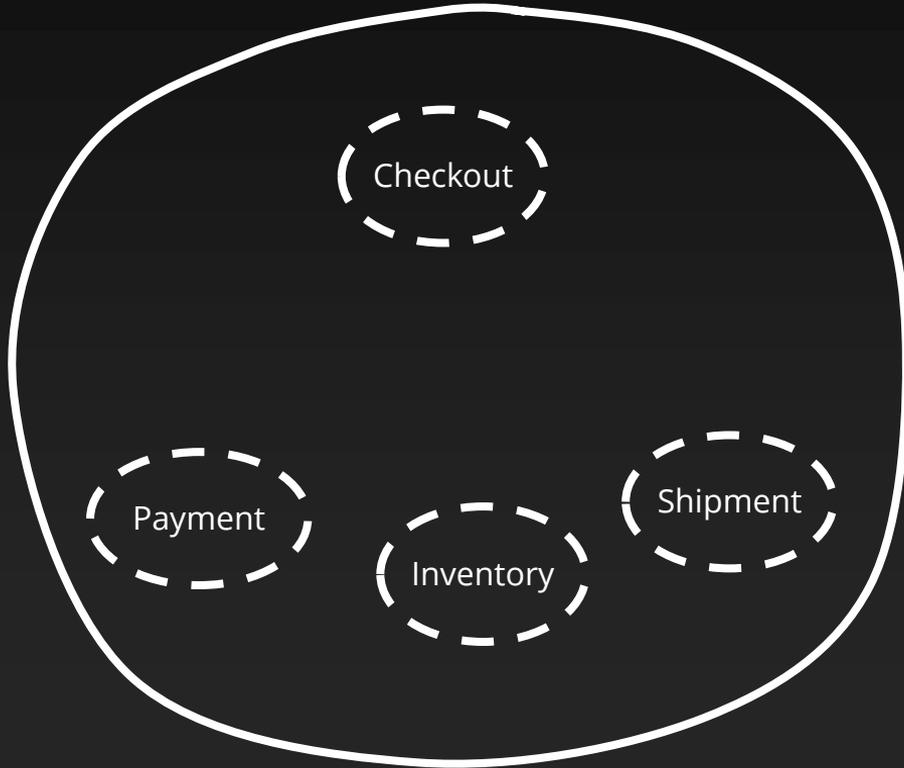
Ship item



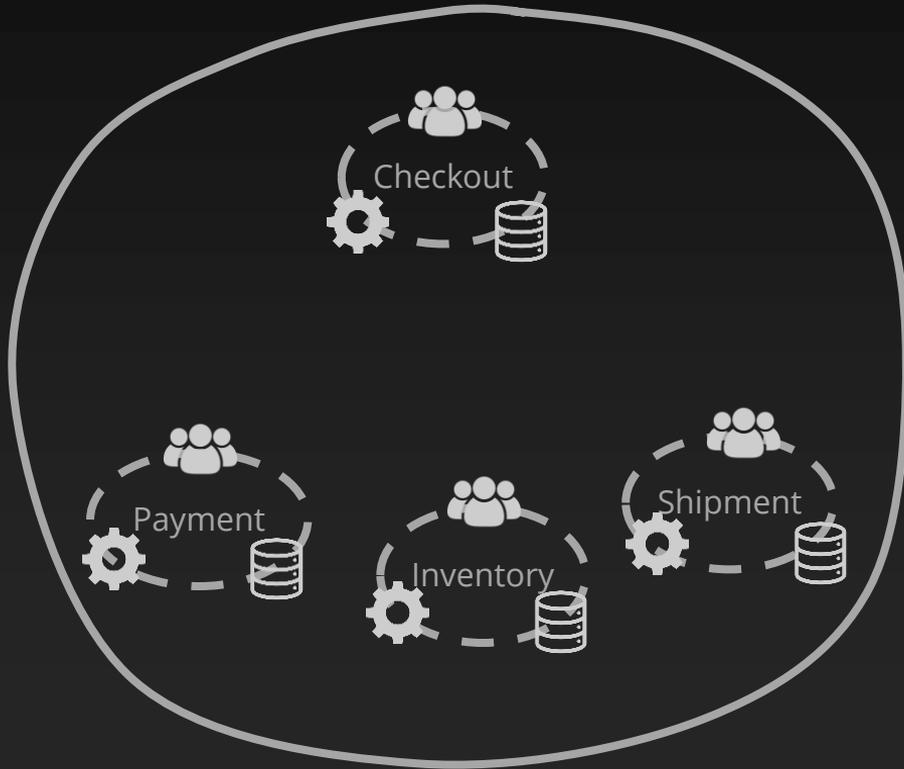
Who is involved? Some bounded contexts...



(Micro-)services



Autonomous (micro-)services



Dedicated Application Processes



Dedicated infrastructure

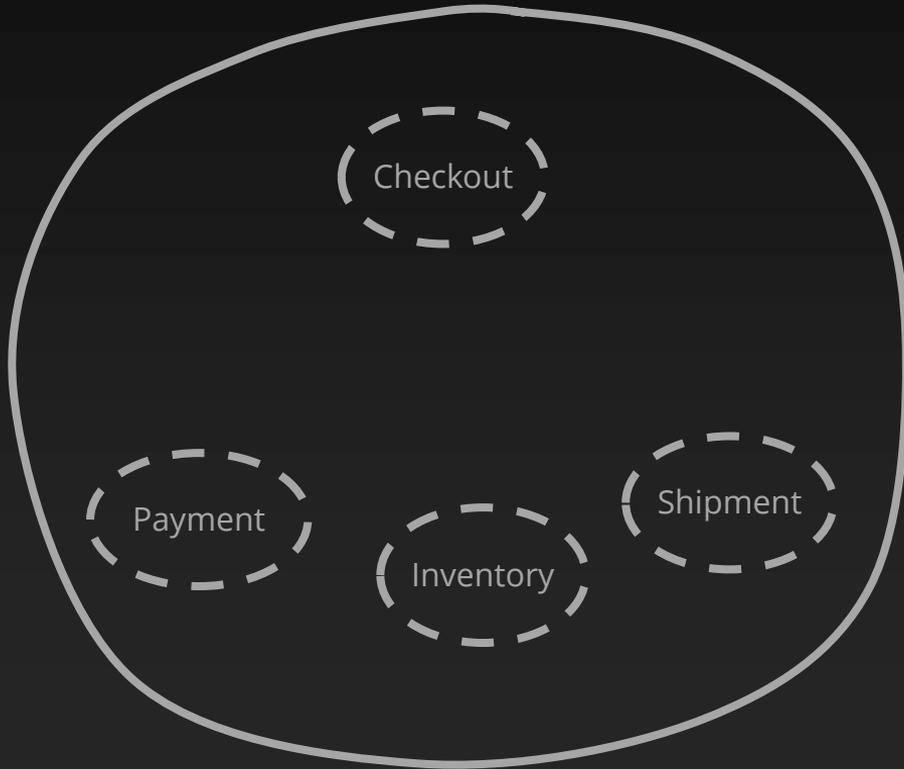


Dedicated Development Teams



Events decrease coupling

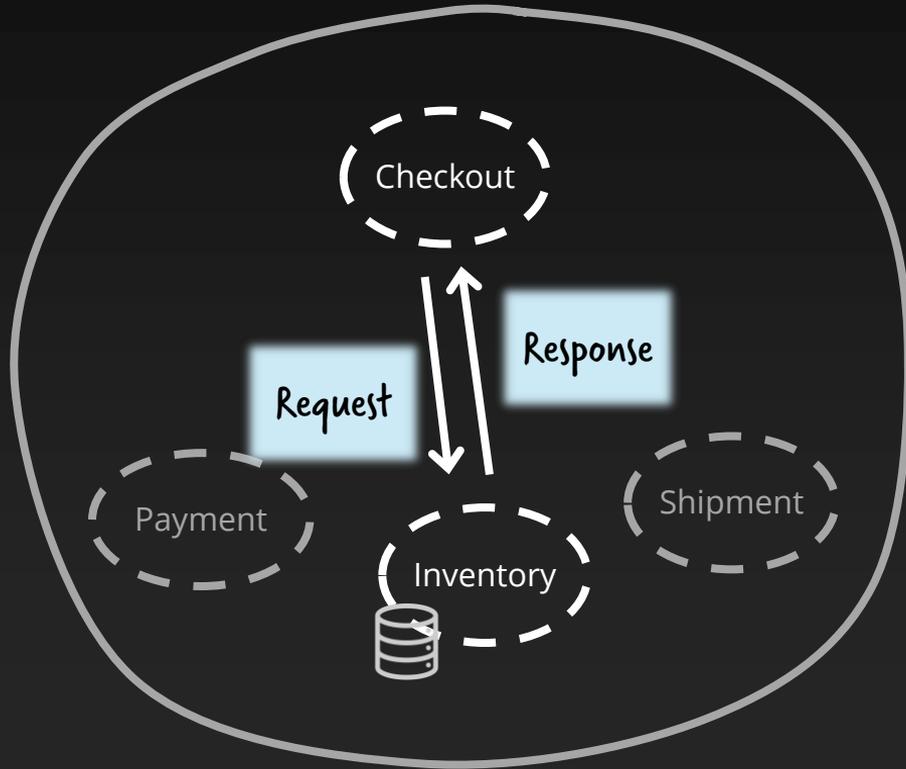
Example



The button blinks if we can ship within 24 hours



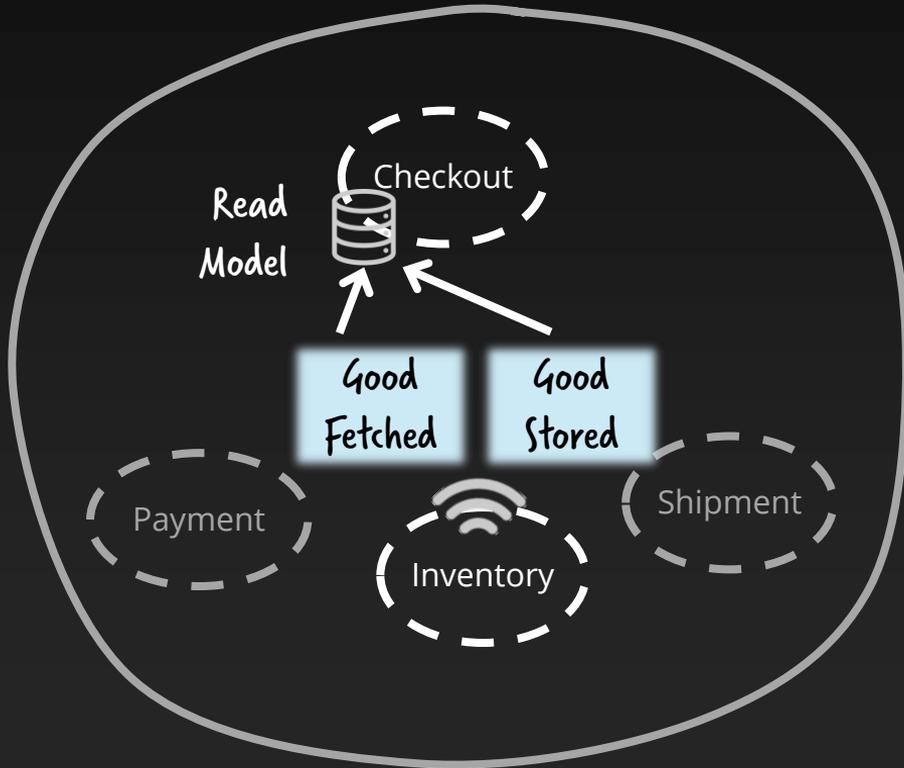
Request/response: temporal coupling



The button blinks if we can ship within 24 hours



Temporal decoupling with events and read models



The button blinks if we can ship within 24 hours

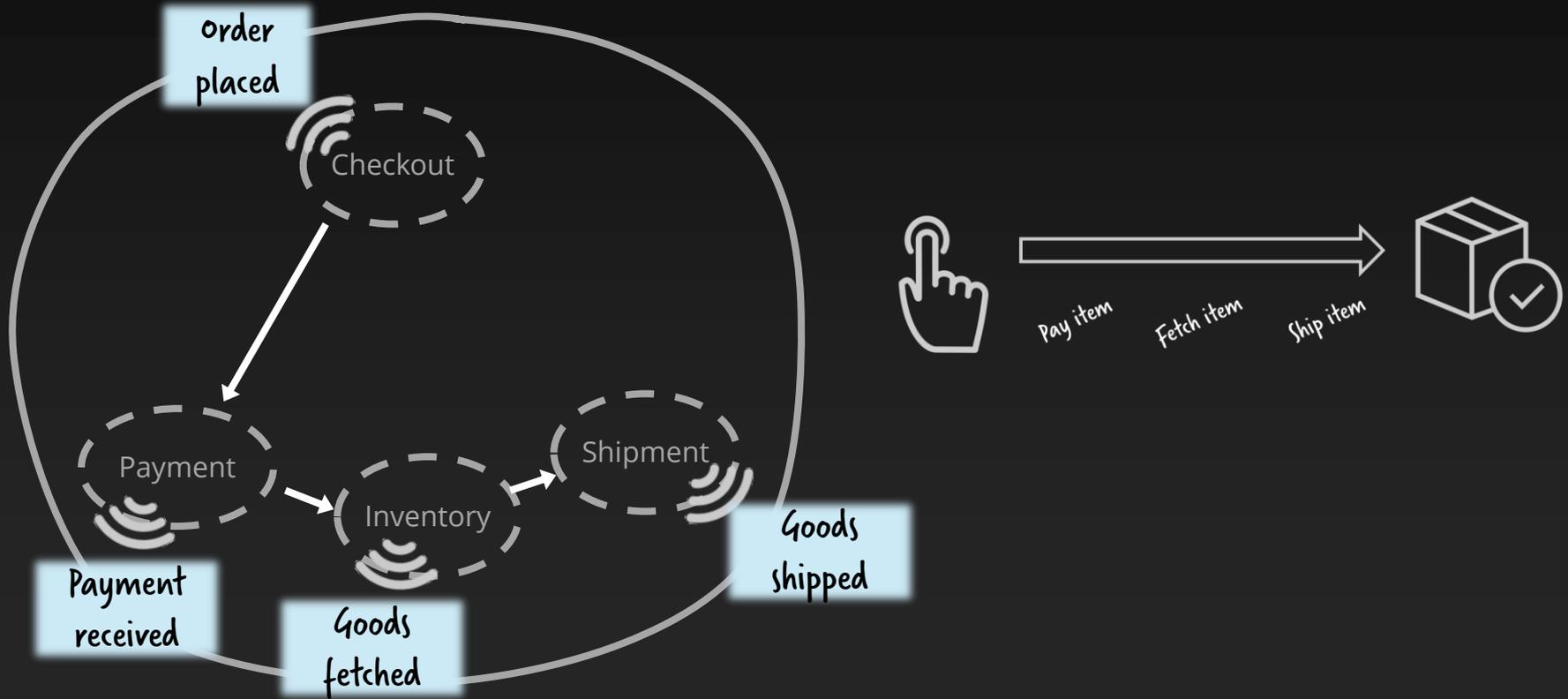


***Events** are facts about what happened (in the past)

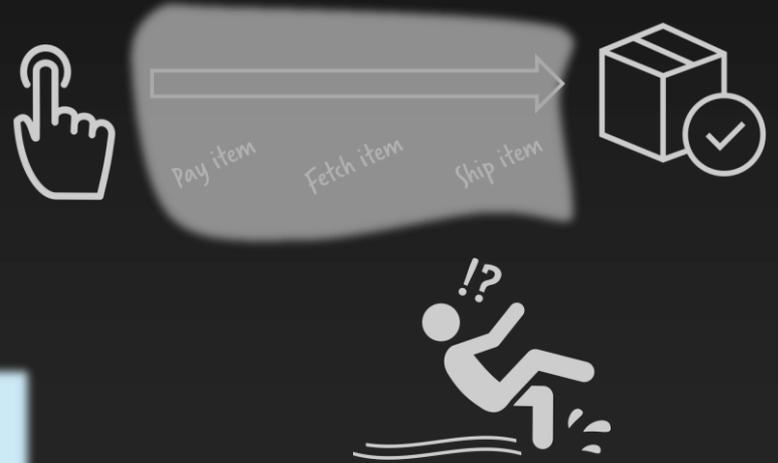
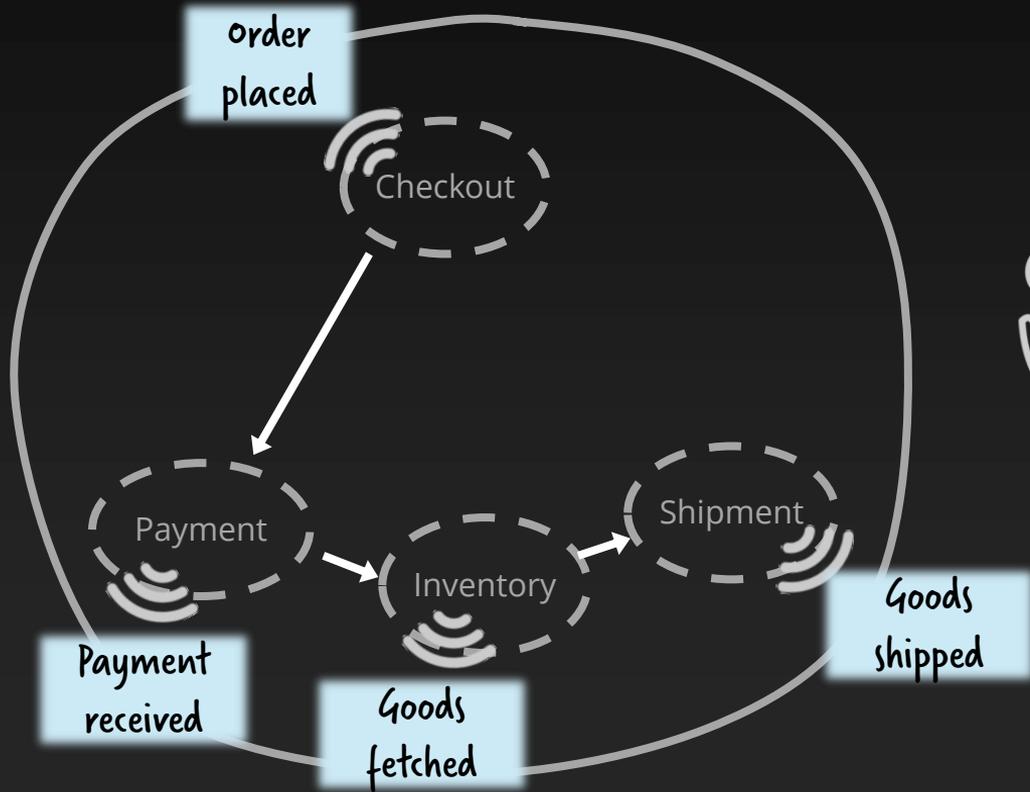
Events can decrease coupling*

*e.g. decentral data-management, read models,
extract cross-cutting aspects

Peer-to-peer event chains



Peer-to-peer event chains





The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

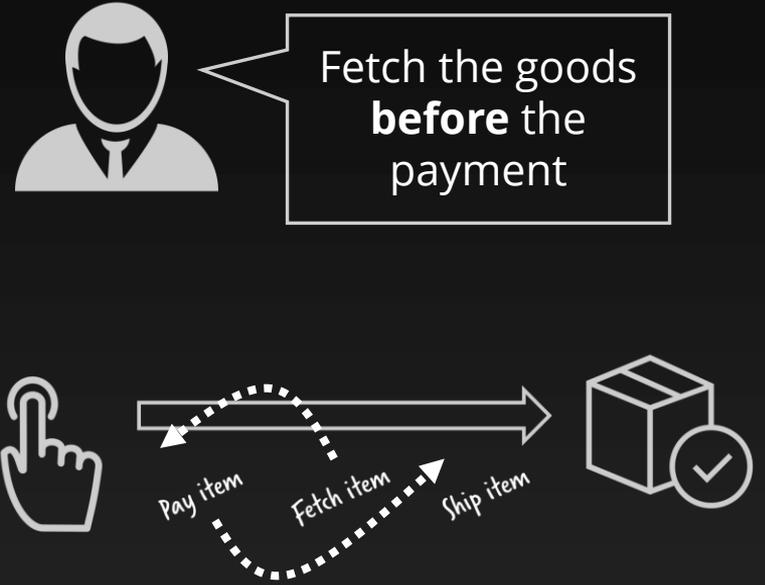
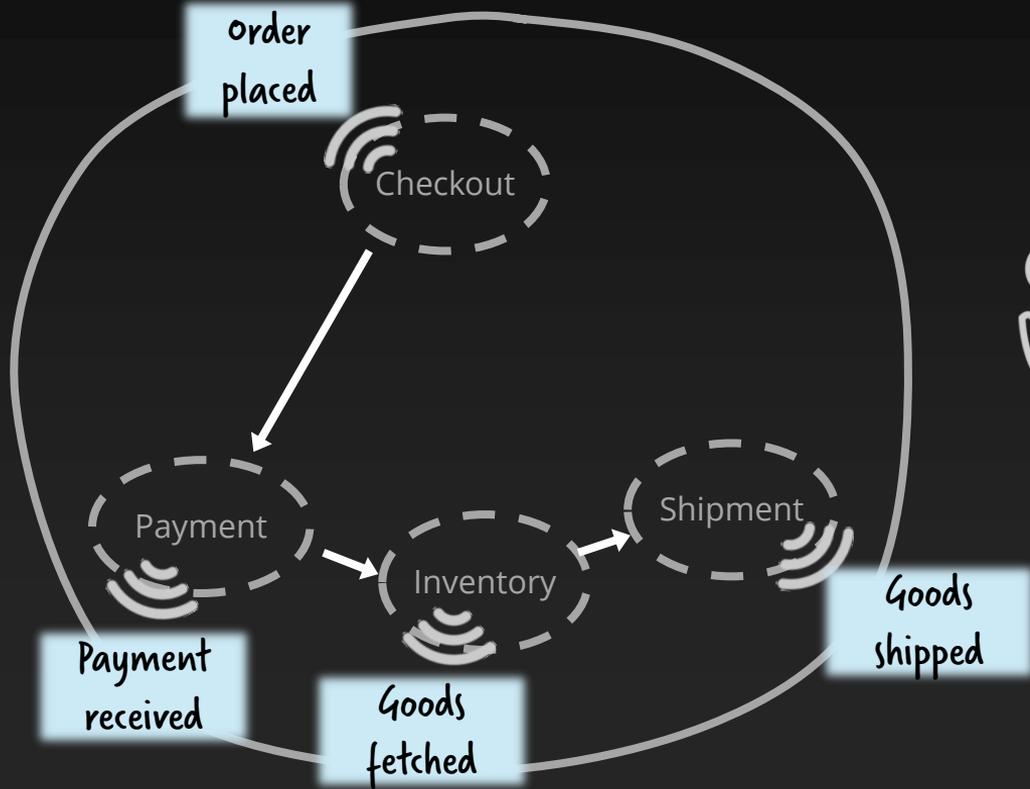


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

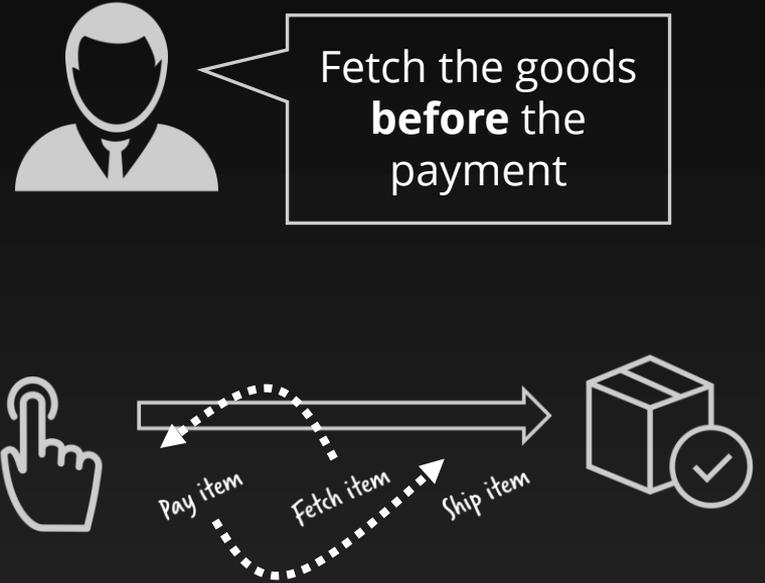
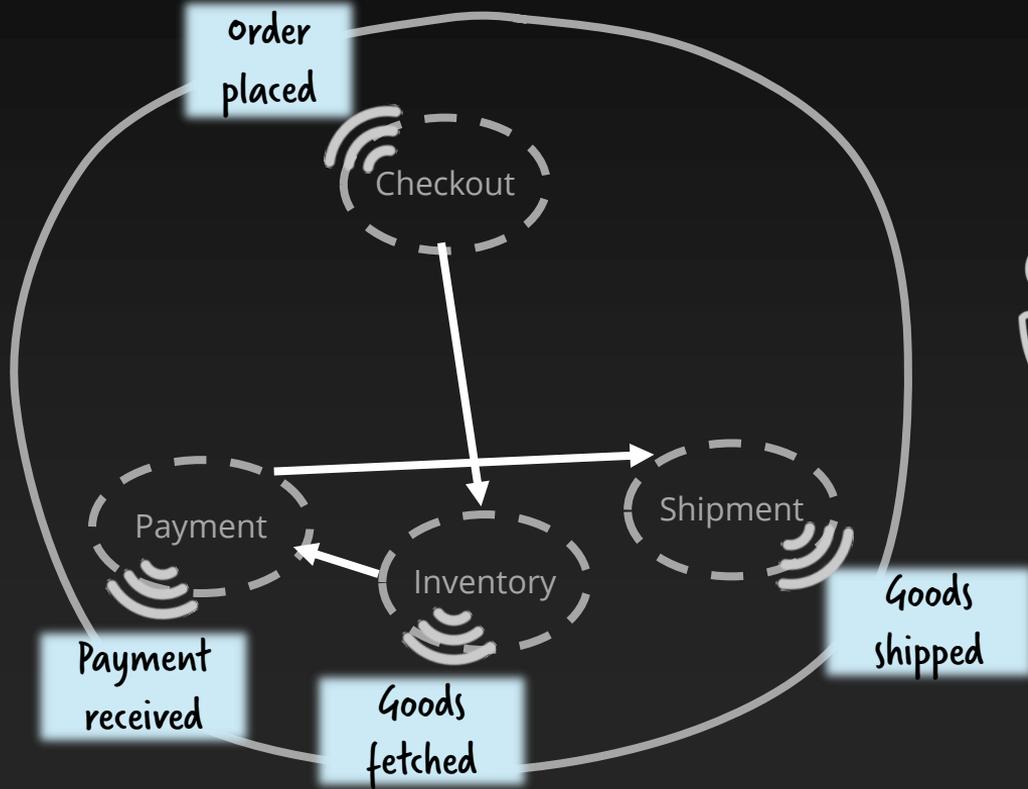


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

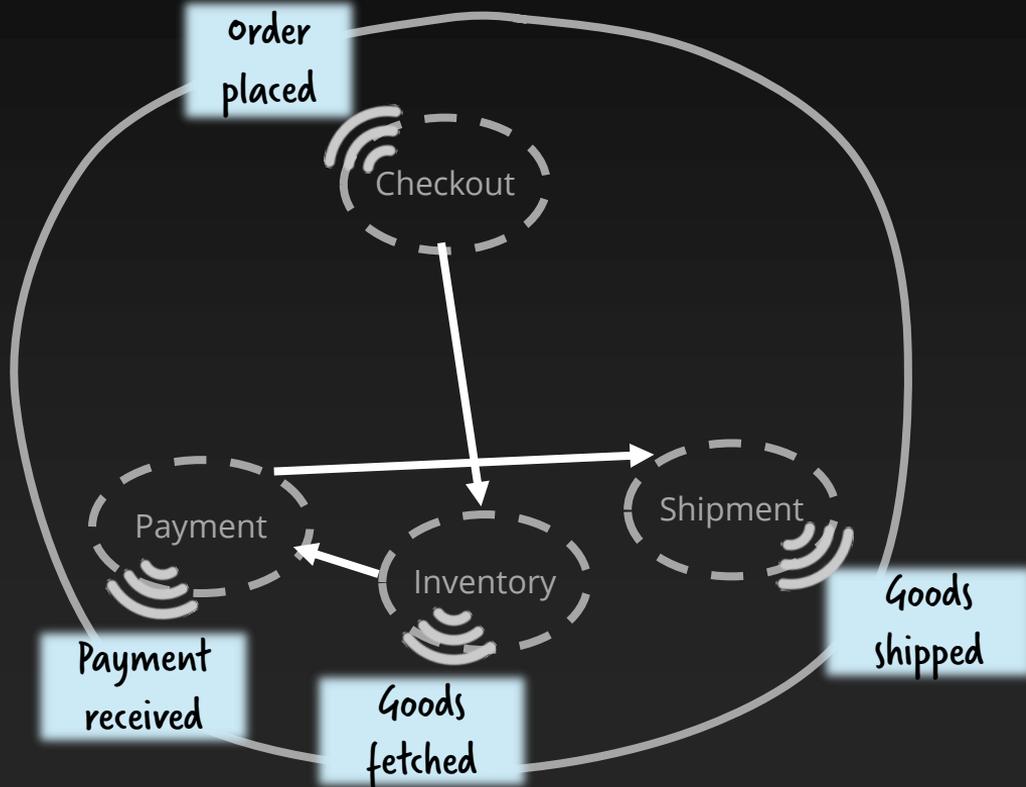
Peer-to-peer event chains



Peer-to-peer event chains



Peer-to-peer event chains



Fetch the goods **before** the payment

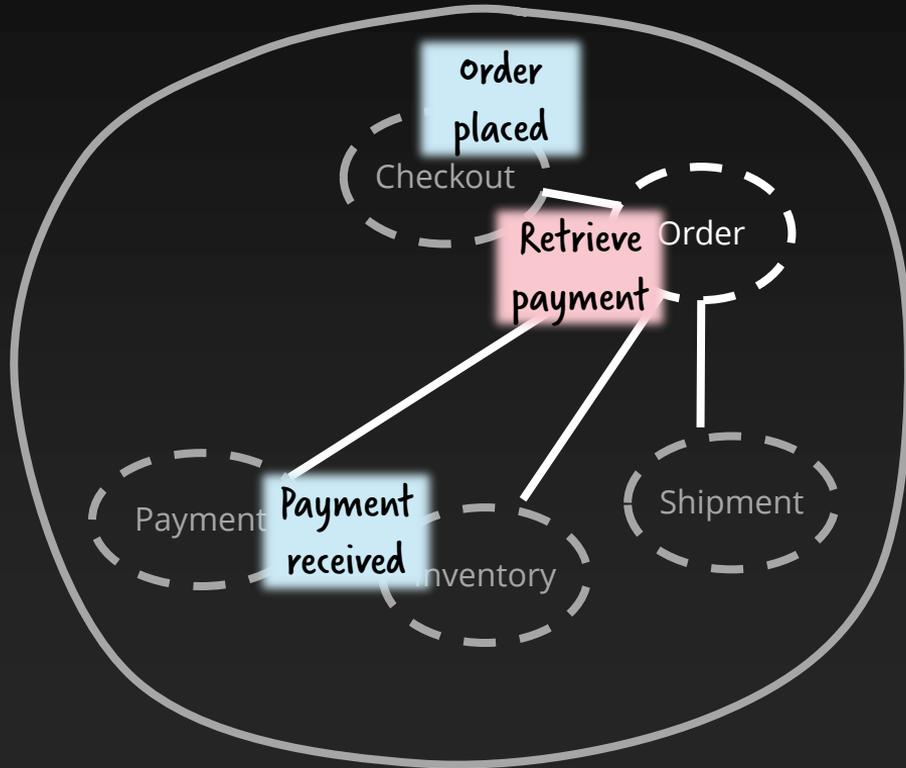
Customers can pay via invoice

...



Photo by born1945, available under [Creative Commons BY 2.0 license](https://creativecommons.org/licenses/by/2.0/).

Extract the end-to-end responsibility



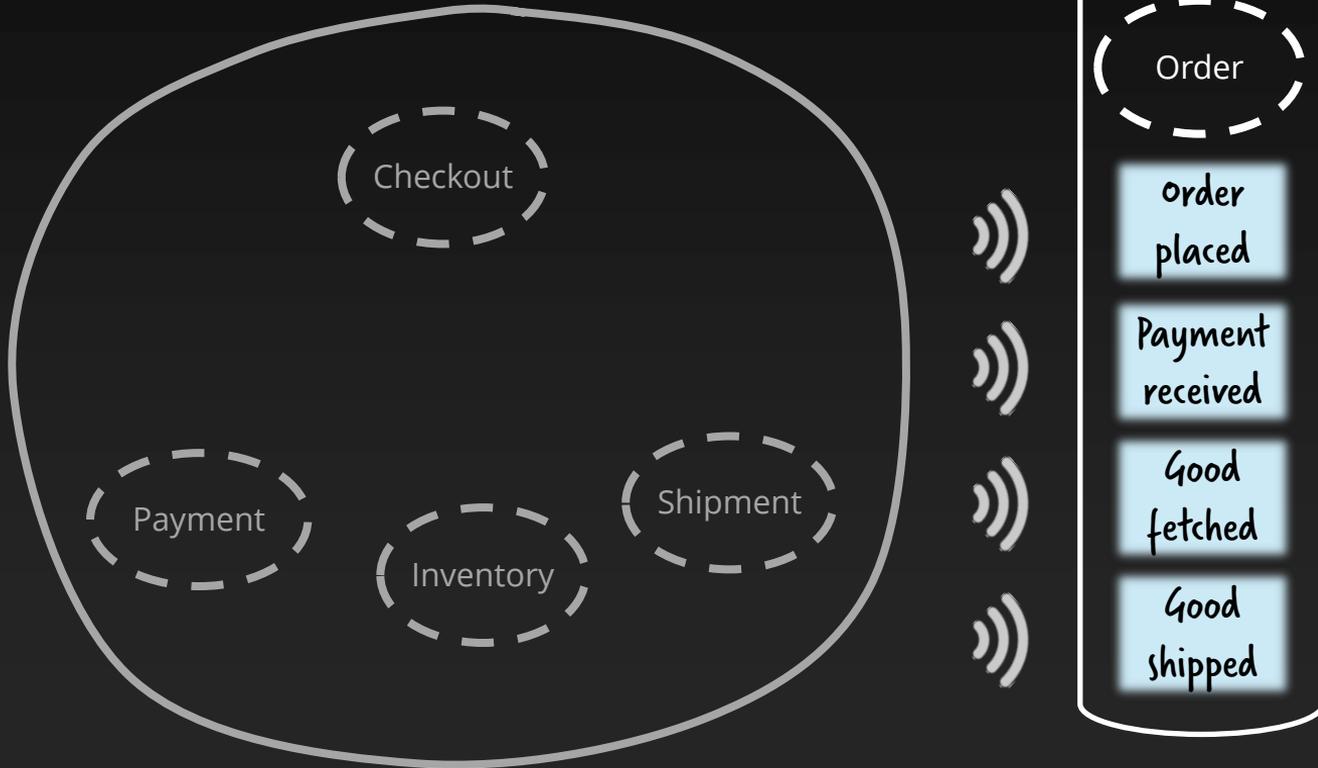
***Commands** have an intent about what needs to happen in the future

Commands help to avoid (complex)
peer-to-peer event chains

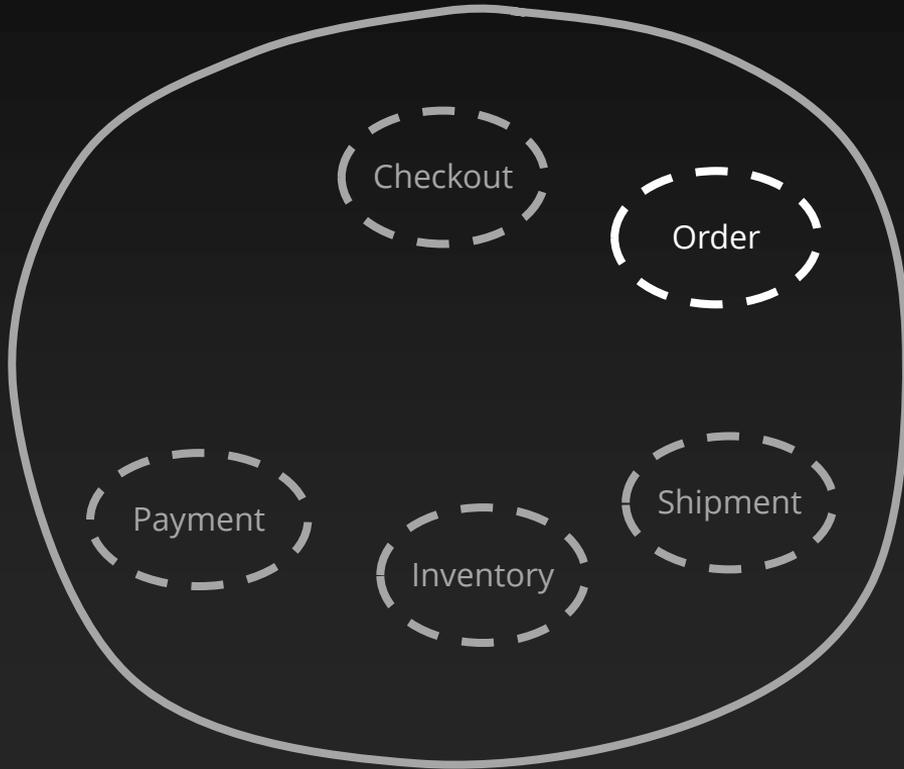


orchestration needs to be avoided

Smart ESB-like middleware



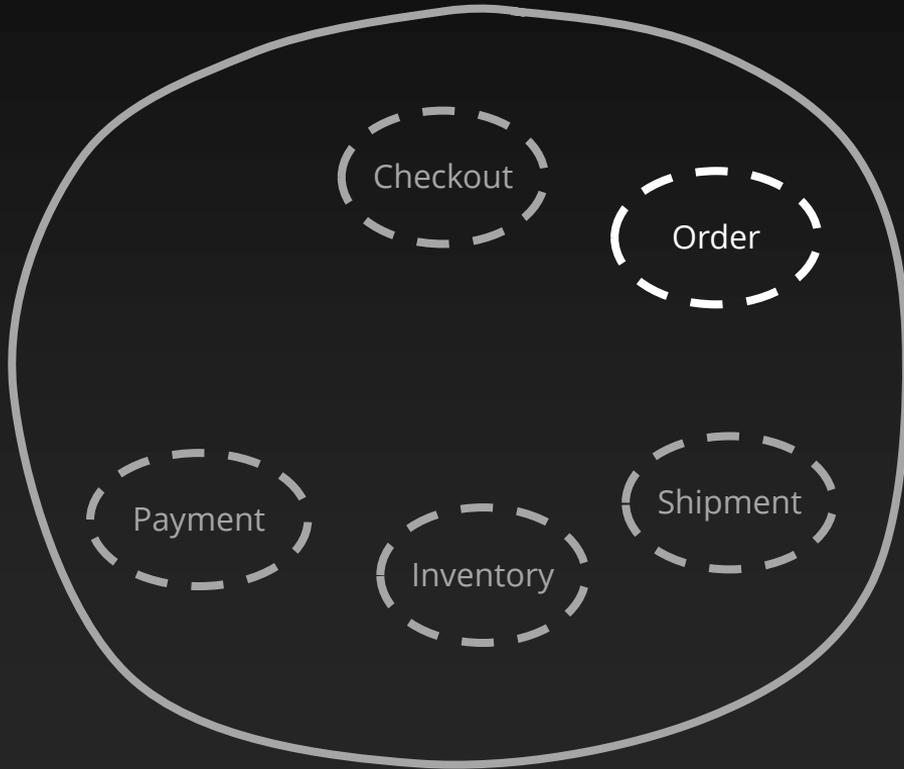
Dumb pipes



Martin Fowler

Smart endpoints
and **dumb pipes**

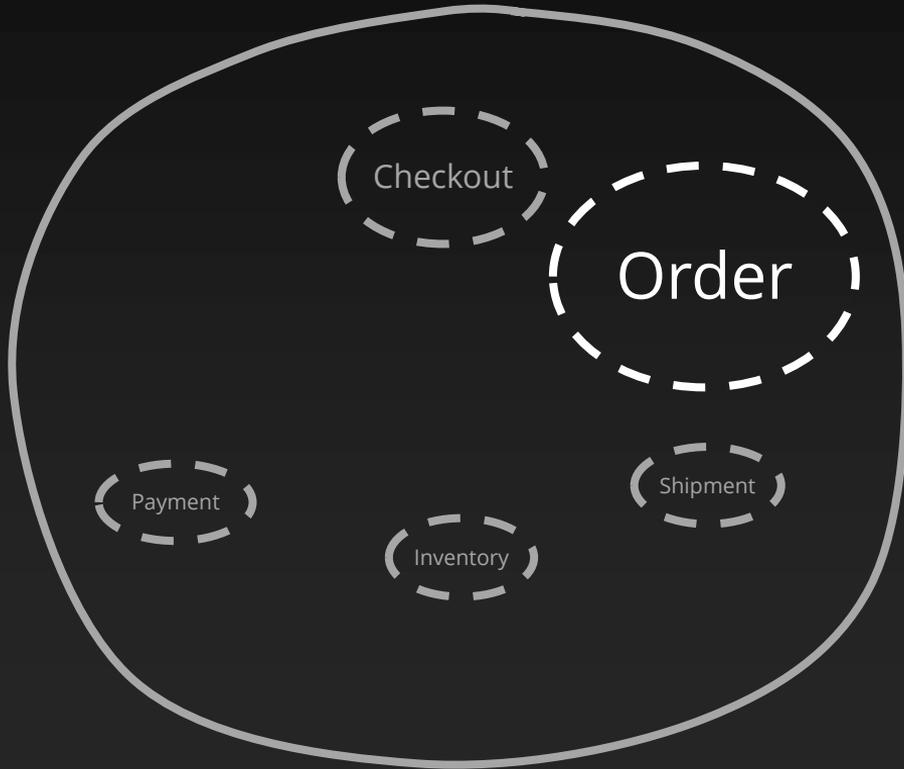
Danger of god services?



Sam Newmann

A few
smart god services
tell
anemic CRUD services
what to do

Danger of god services?



Sam Newmann

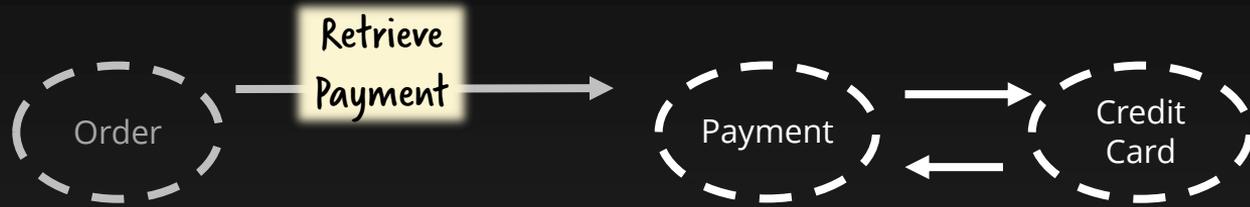
A few
smart god services
tell
anemic CRUD services
what to do

A god service is only created
by bad API design!

Example



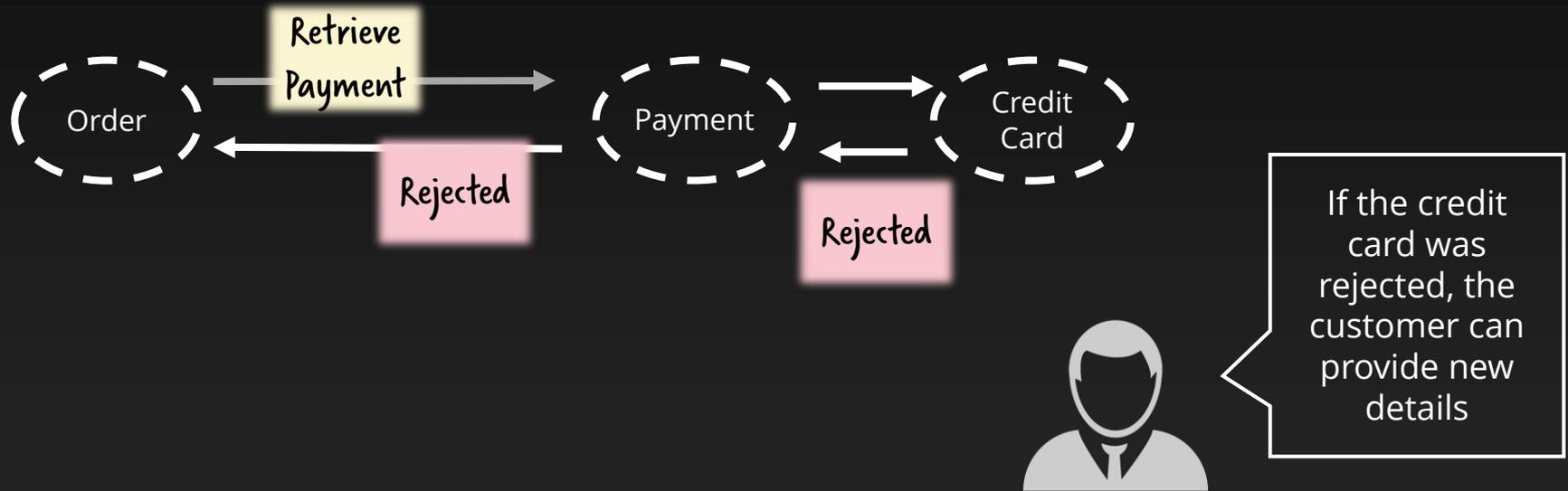
Example



Example

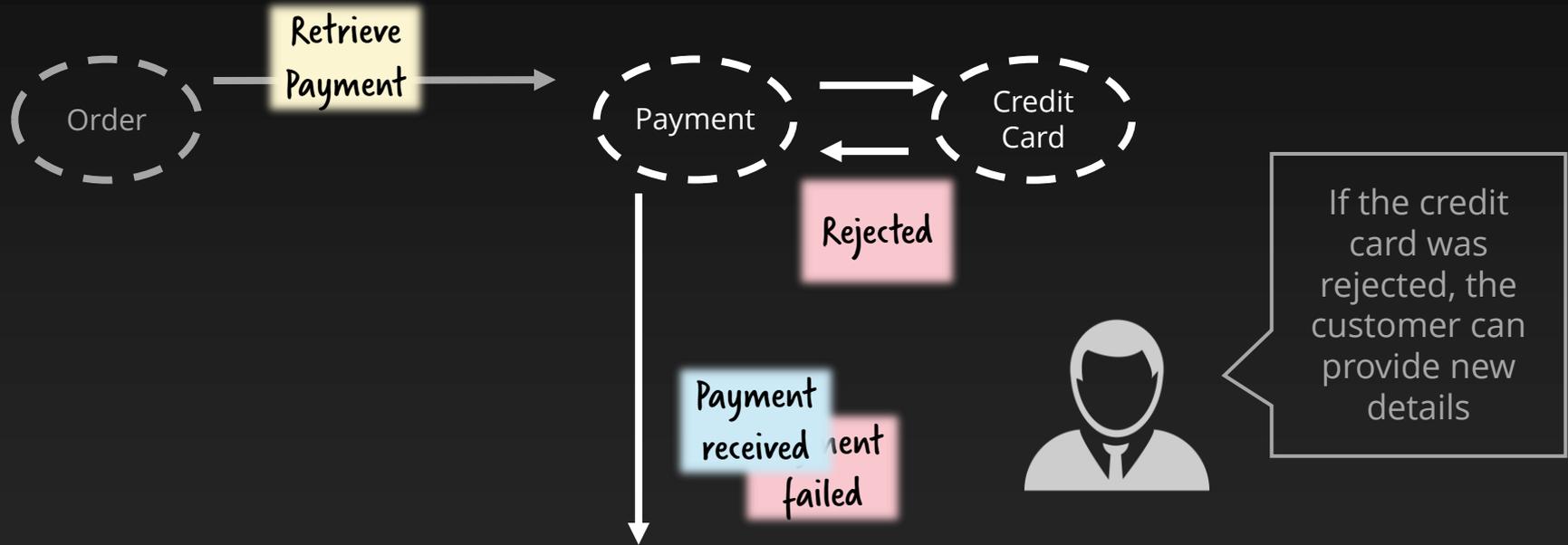


Example

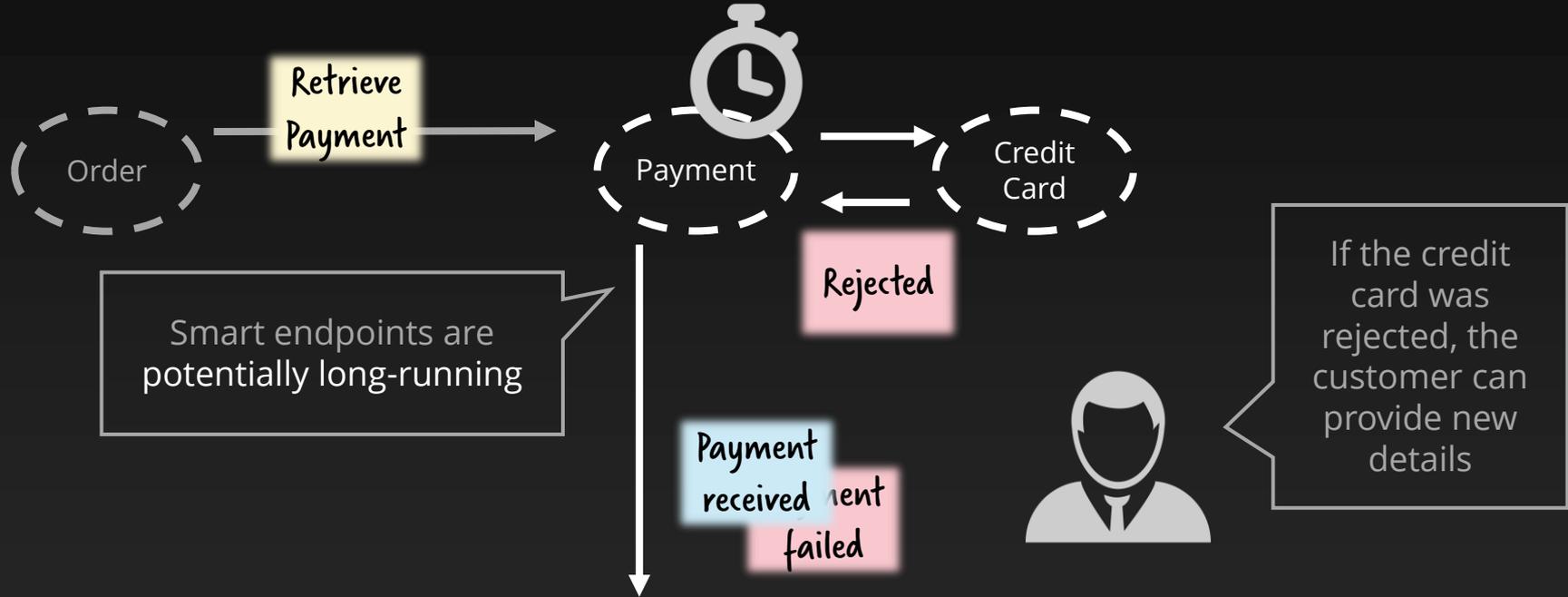


Client of **dumb endpoints** easily become a god services.

Who is responsible to deal with problems?



Who is responsible to deal with problems?



Clients of **smart endpoints** remains lean.



Photo by [Tookapic](#), available under [Creative Commons CC0 1.0 license](#).

„There was an error while sending your boarding pass“

Home ▶ Mein Flug: My Eurowings ▶ Bordkarten anzeigen ▶ Meine Bordkarten

Ihre Bordkarten

Beim Versenden der Bordkarte ist ein Fehler aufgetreten.

Ihr Buchungscode **08HHSS**

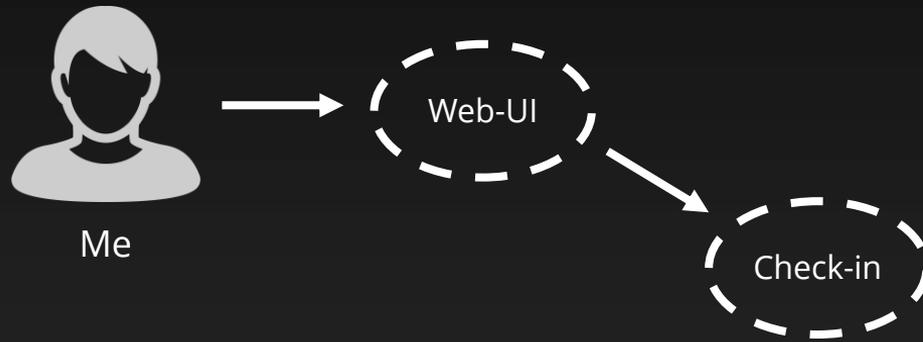
Hinflug

BERND RUECKER

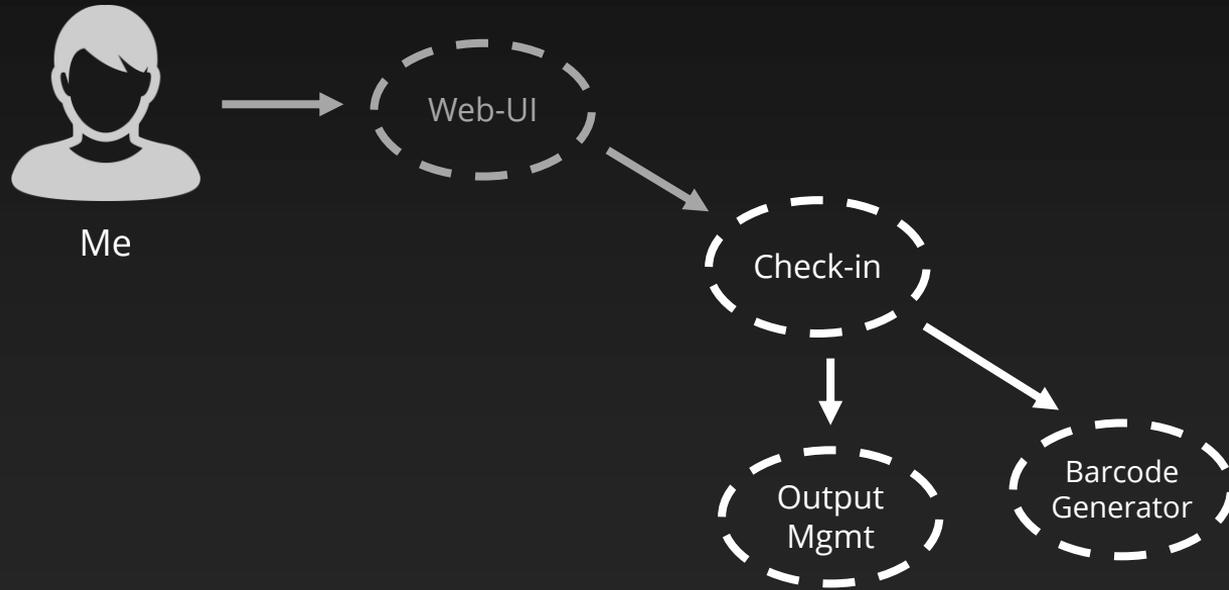
Stuttgart (STR) - London-Stansted (STN)

 M 08:11-09:17 - 10:10 - 10:15

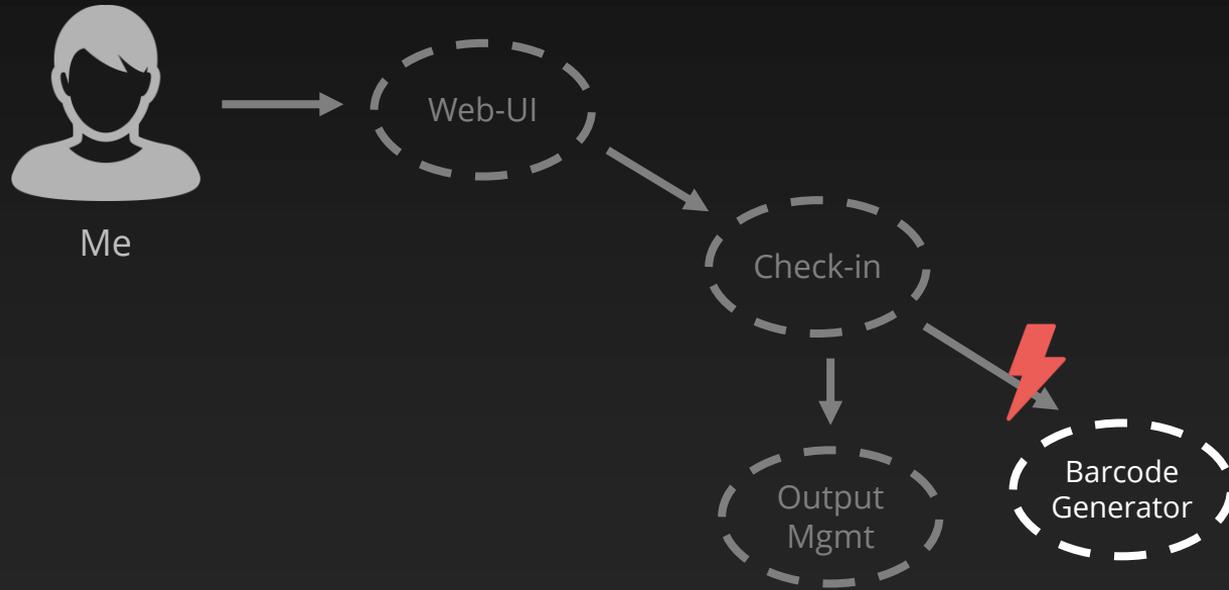
Current situation



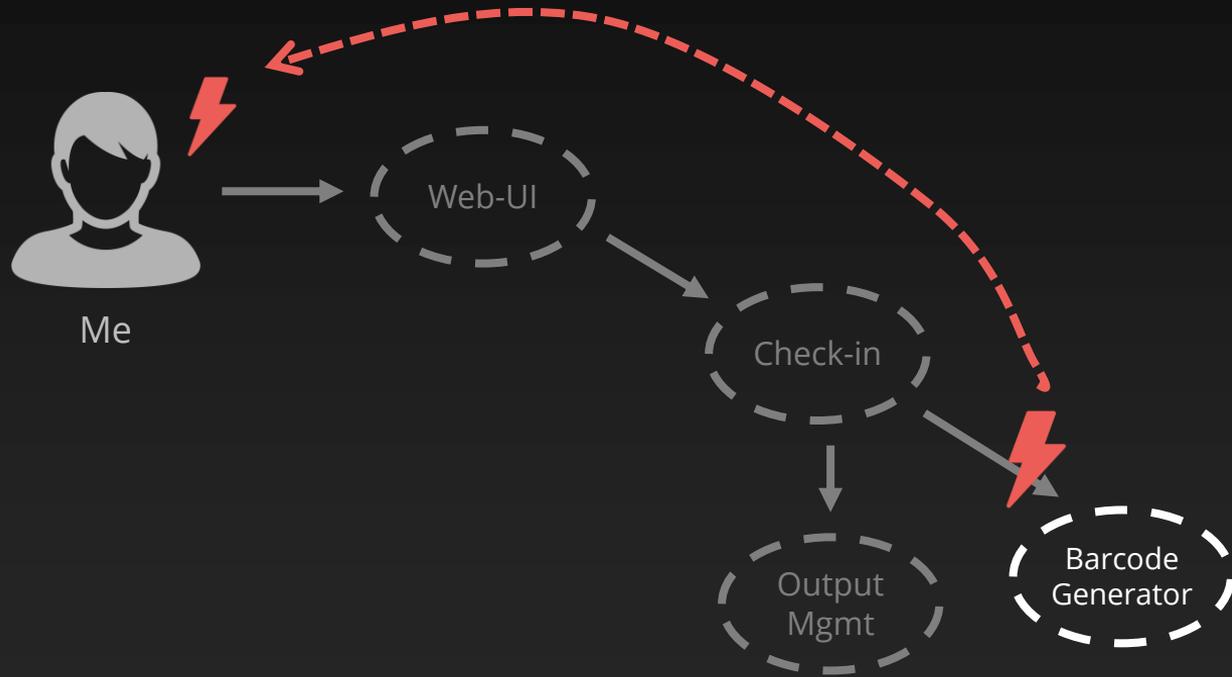
Current situation



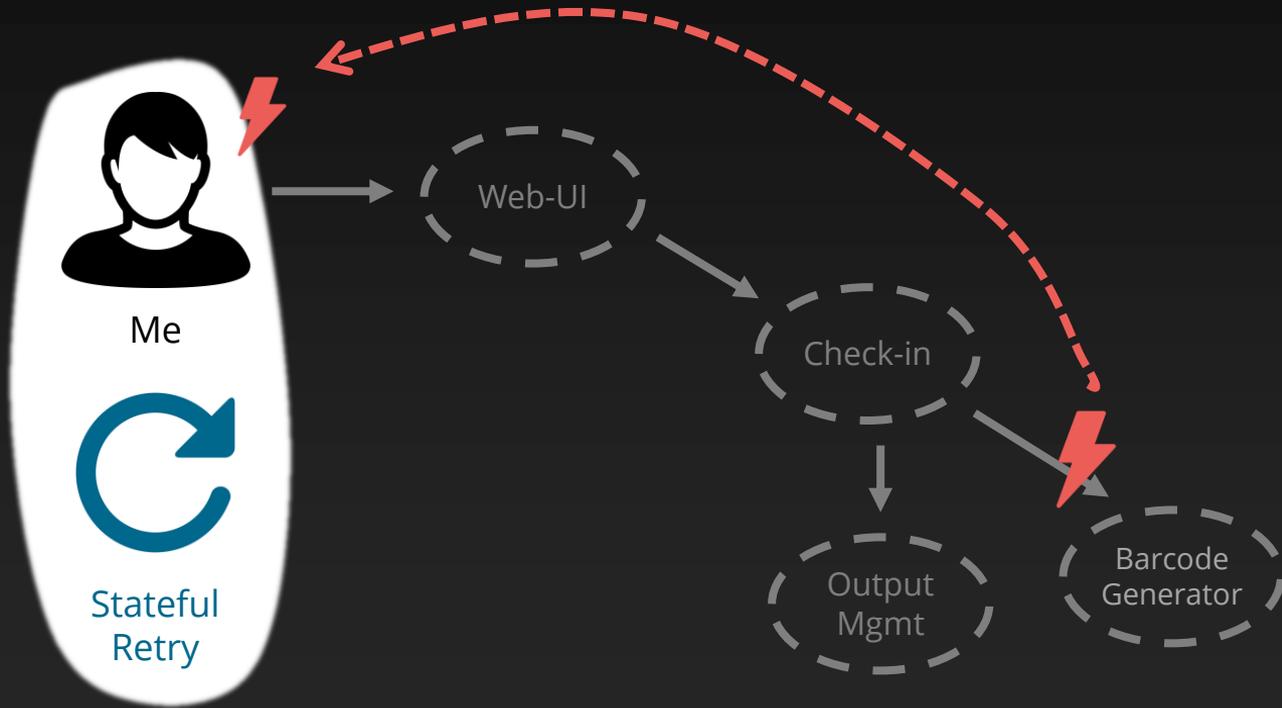
Current situation



Current situation – the bad part



Current situation – the bad part



Ihre B

easyJet

Ihr Buchung

Hinflug

BERND RUEC

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](http://www.easyjet.com)

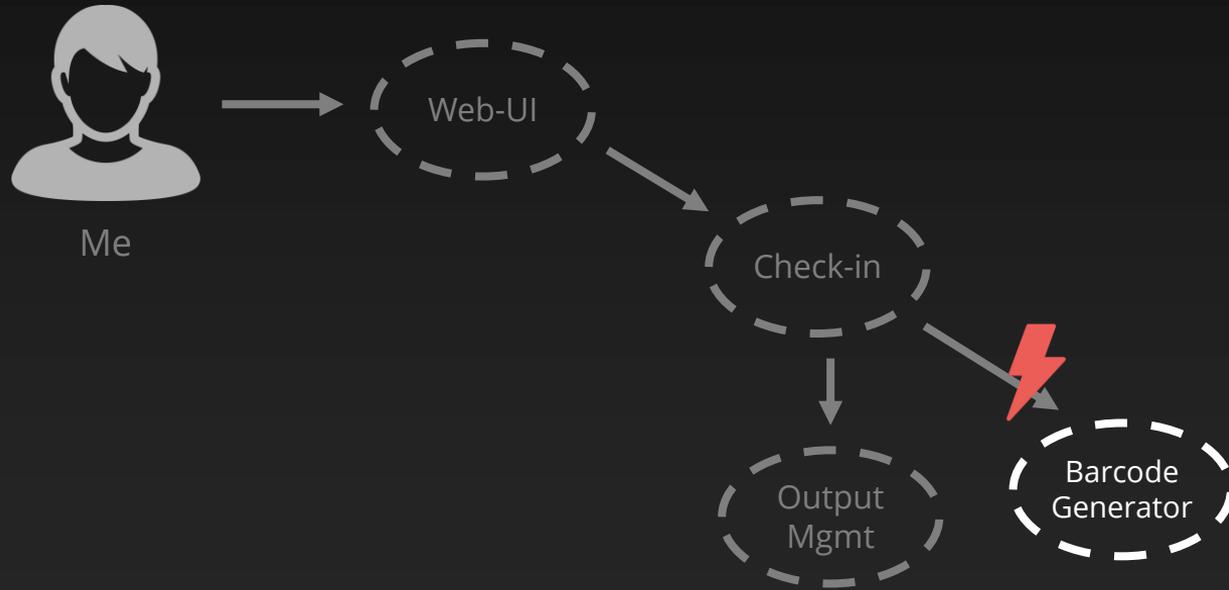
...I just made this up...

We're sorry

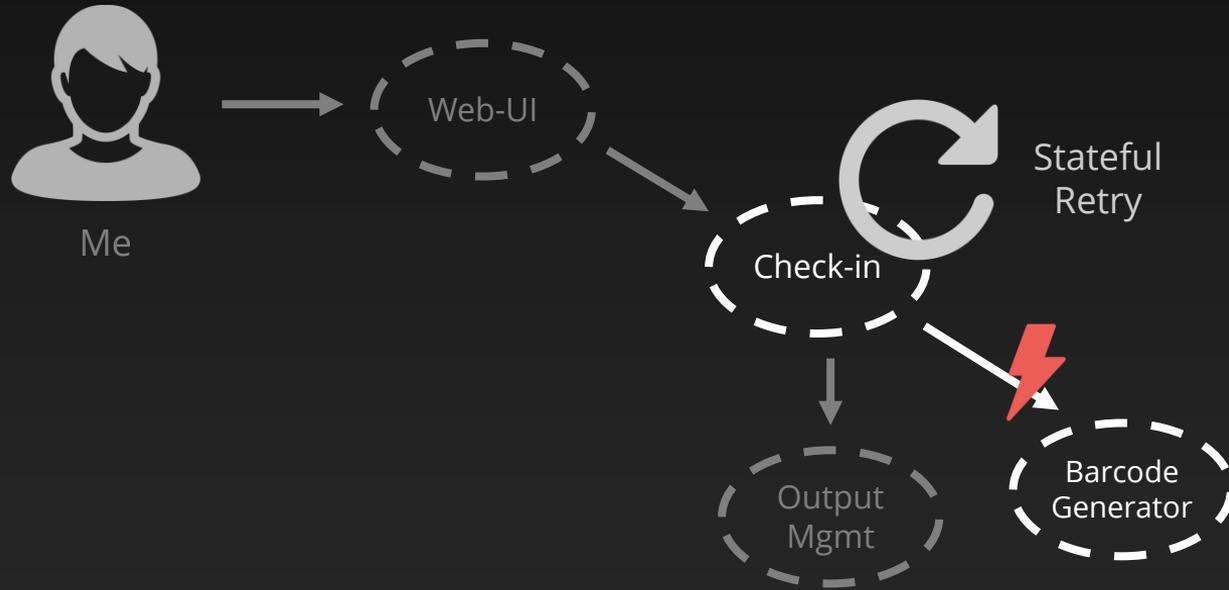
We are having some technical difficulties and cannot present you your boarding pass right away.

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

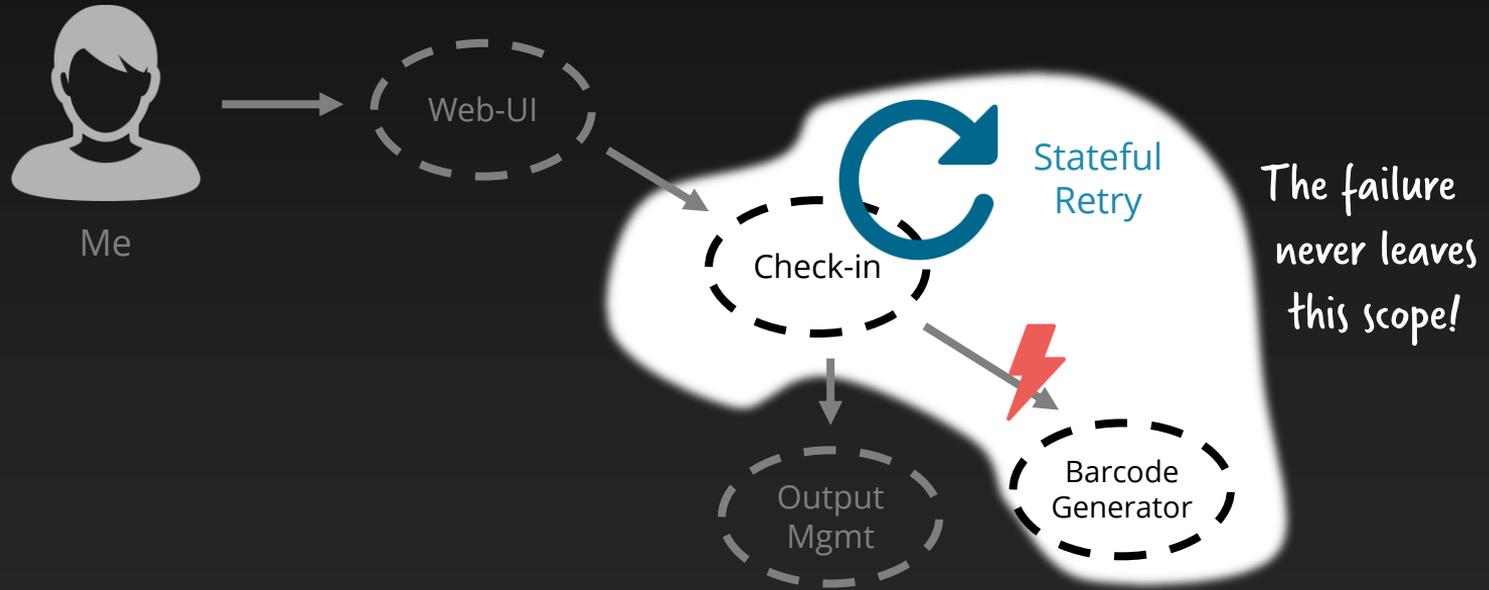
Possible situation – much better!



Possible situation – much better!



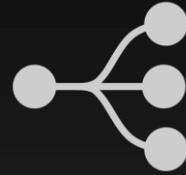
Possible situation – much better!



Handling State



Persist thing
(Entity, Actor, ...)



State machine or
workflow engine

Typical
concerns

DIY = effort,
accidental
complexity



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



Workflow engines are painful

~~Complex, proprietary, heavyweight, central, developer adverse, ...~~

Avoid the wrong tools!

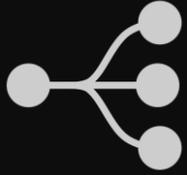


Low-code is great!
(You can get rid
of your developers!)



Death by properties panel

Complex, proprietary, heavyweight, central, developer adverse, ...

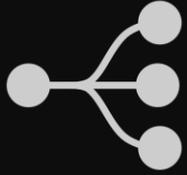


Workflow engines,
state machines



AWS Step
Functions

It is
relevant
in modern
architectures

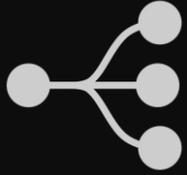


Workflow engines,
state machines



Silicon valley
has recognized





Workflow engines,
state machines

UBER

CADENCE

There are
lightweight open source
options



AWS Step
Functions



camunda

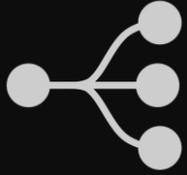


conductor

NETFLIX OSS



Activiti™



Workflow engines,
state machines

UBER

CADENCE

also at scale



zeebe

by Camunda



AWS Step
Functions



camunda



jBPM

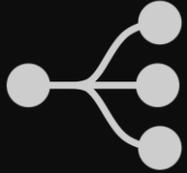


conductor

NETFLIX OSS



Activiti™



Workflow engines,
state machines

UBER CADENCE

for today's demo



AWS Step
Functions



camunda



jBPM



conductor

NETFLIX OSS



Activiti™

```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow") //
            .startEvent()
            .serviceTask("Step 1")
            .serviceTask("Step 2")
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().start("flow", Variables.putValue("city", "New York"));
}

public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

What do I mean by
„leightweight?“



```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow") //
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}
public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

Build engine
in one line of
code
(using in-
memory H2)

```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}

public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

Define flow
e.g. in Java
DSL

```

public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

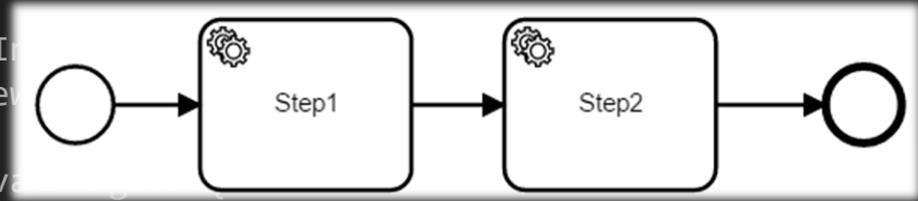
    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}

public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}

```

Define flow
e.g. in Java
DSL



BPMN

Business Process
Model and Notation

ISO Standard



```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}
public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

*We can attach
code...*

```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}

public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

...that is
called when
workflow
instances pass
through

```
public static void main(String[] args) {
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
        .buildProcessEngine();

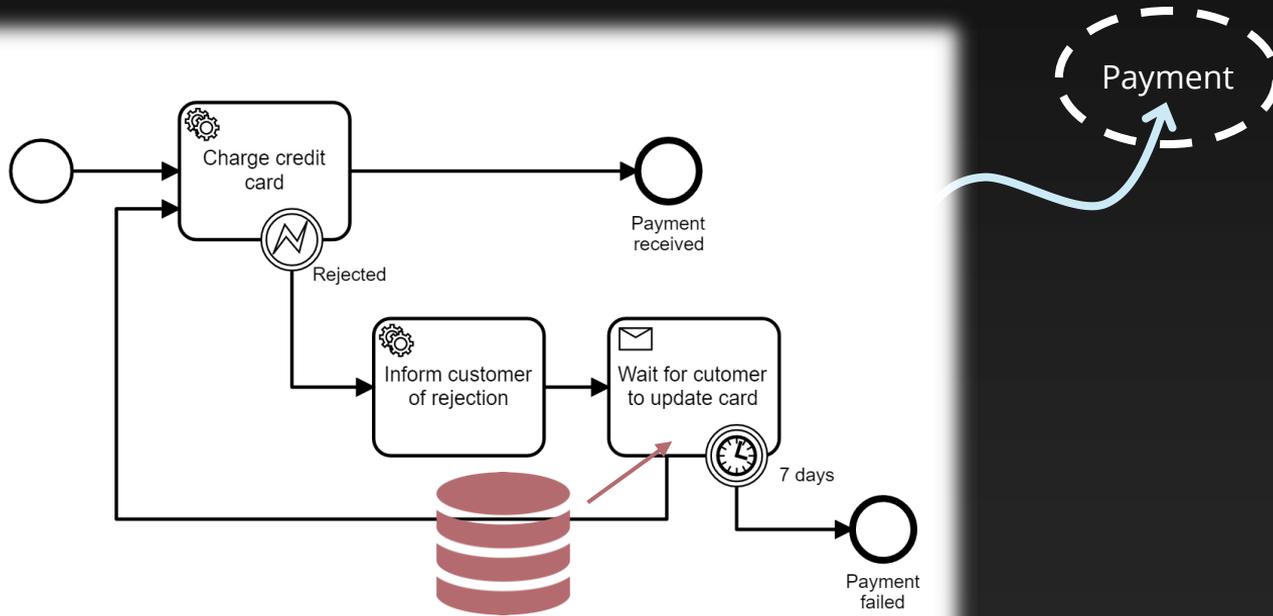
    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
            .endEvent()
            .done()
        ).deploy();

    engine.getRuntimeService().startProcessInstanceByKey(
        "flow", Variables.putValue("city", "New York"));
}

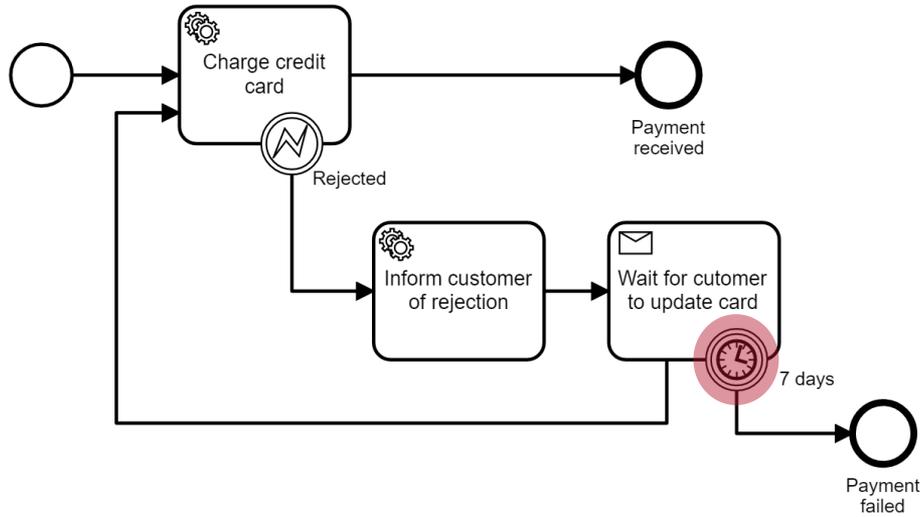
public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

Start
instances

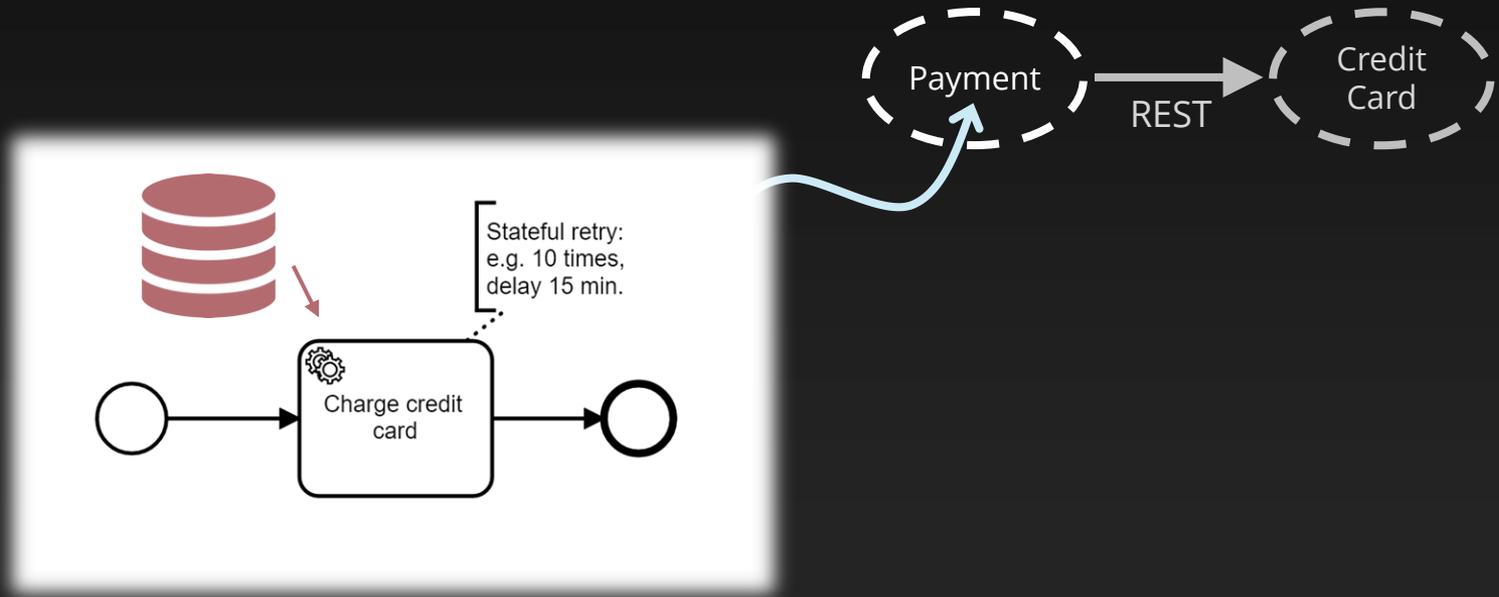
Now you have a state machine!



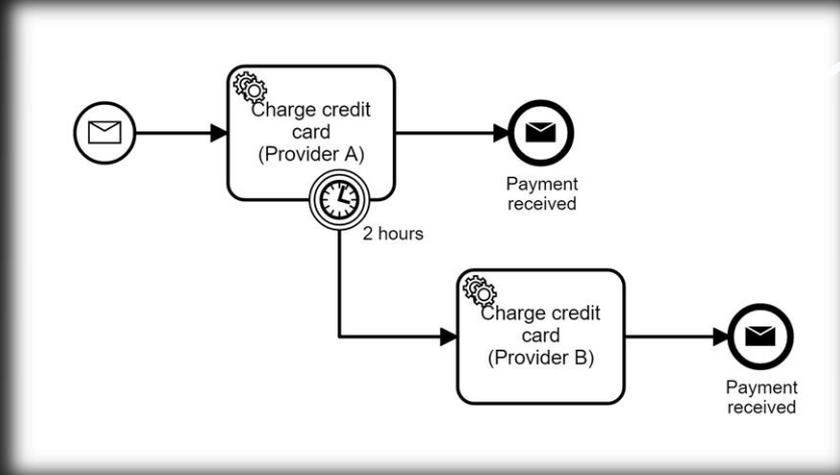
Easy to handle time



Stateful retries



Fallbacks increase resilience

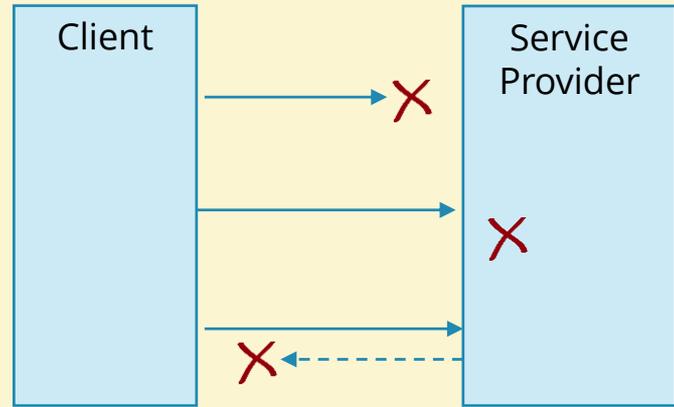


Distributed systems

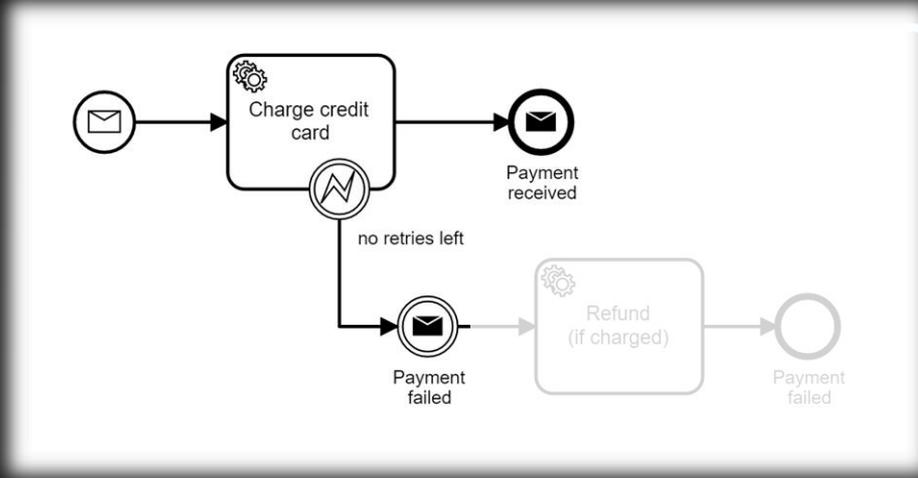


It is impossible to differentiate certain failure scenarios.

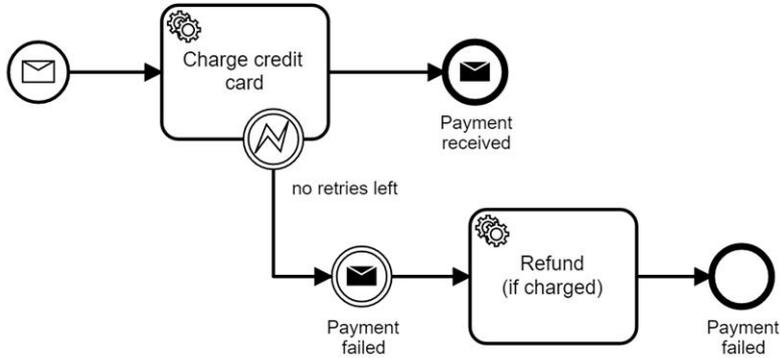
Independant of communication style!



Distributed systems introduce complexity you have to tackle!



Distributed systems introduce complexity you have to tackle!



Distributed systems

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.

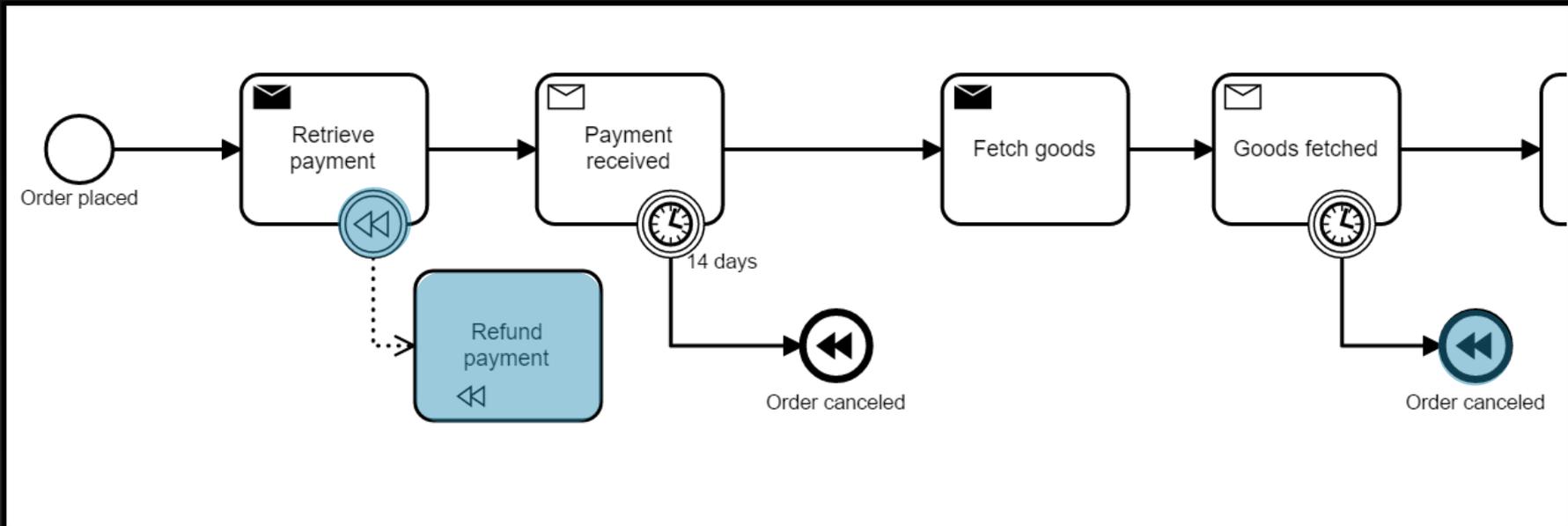
ABSTRACT

Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC. For many years, the approaches to guaranteeing consistency in the context of distributed transactions have been

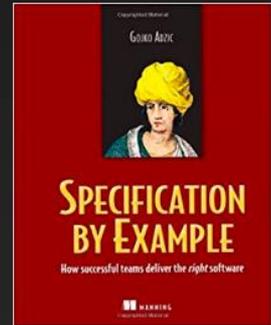
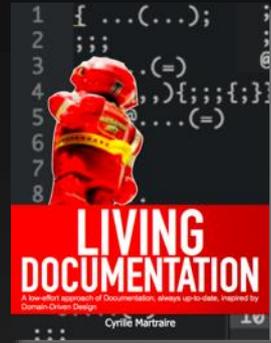
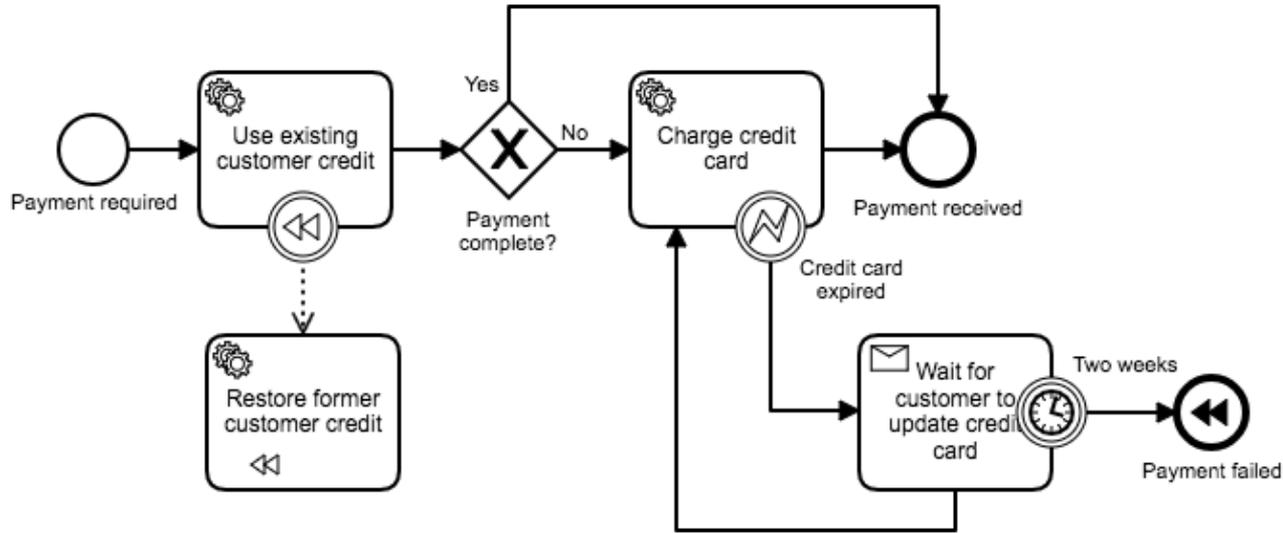
Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses. This paper explores practical alternatives to 2PC.

Distributed transaction with compensations*

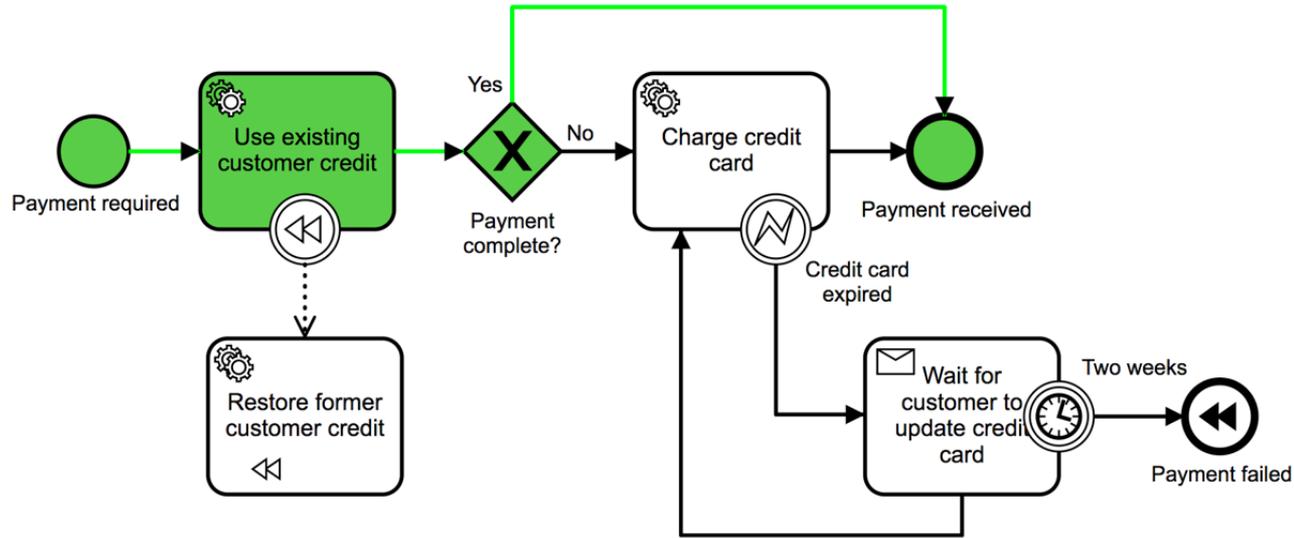
*aka Saga-Pattern



Living documentation for long-running behaviour



Visual HTML reports for test cases



Proper operations

Visibility + Context

Camunda Cockpit Processes Decisions Cases Human Tasks More ▾

Dashboard > Processes > paymentV5 : Runtime | History

Definition Version: 2 ▾

Version Tag: null

Definition ID: paymentV5:2:45aea93a-1ad9-11e8-8...

Definition Key: paymentV5

Definition Name: null

History Time To Live: null

Tenant ID: null

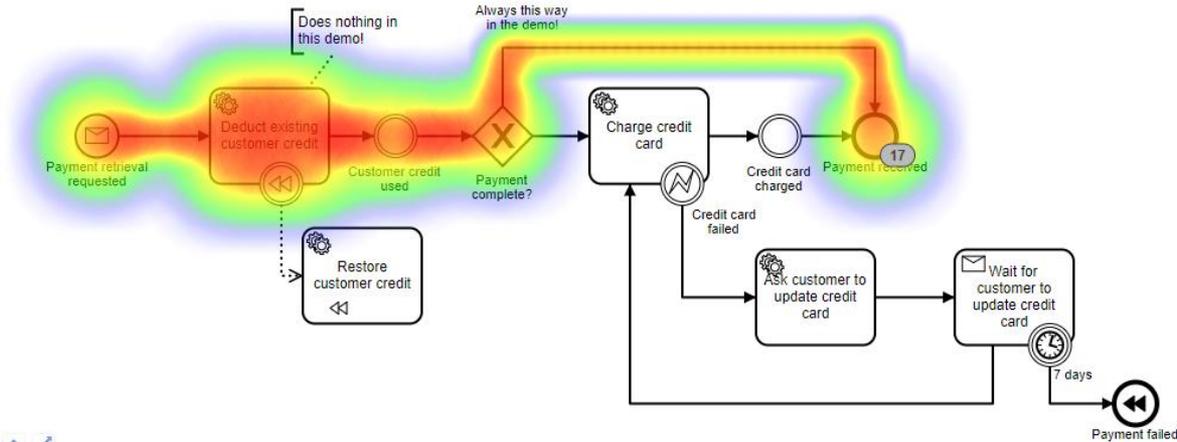
BPMN Diagram:

- Start Event: Payment retrieval requested
- Task: Deduct existing customer credit (4.7k)
- Decision: Payment complete? (X)
- Task: Charge credit card (2.3k, 317)
- End Event: Payment received

Table:

Start Time	Business Key
2018-02-26T10:40:59	
2018-02-26T10:40:18	

Powered by camunda BPM / v7.8.0-ee



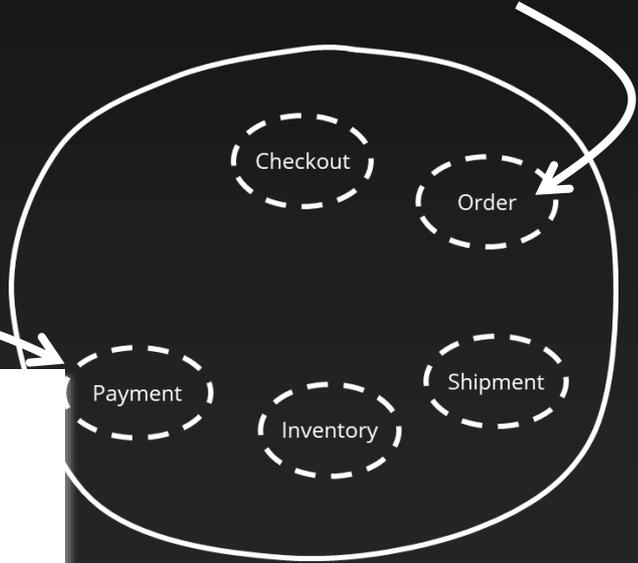
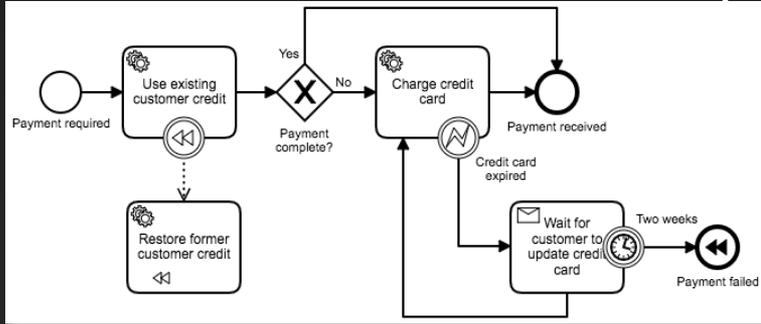
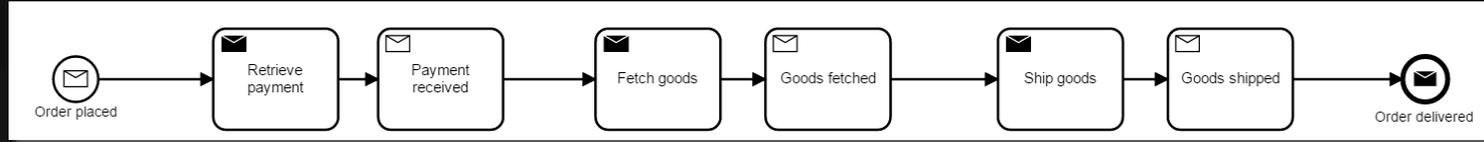


”

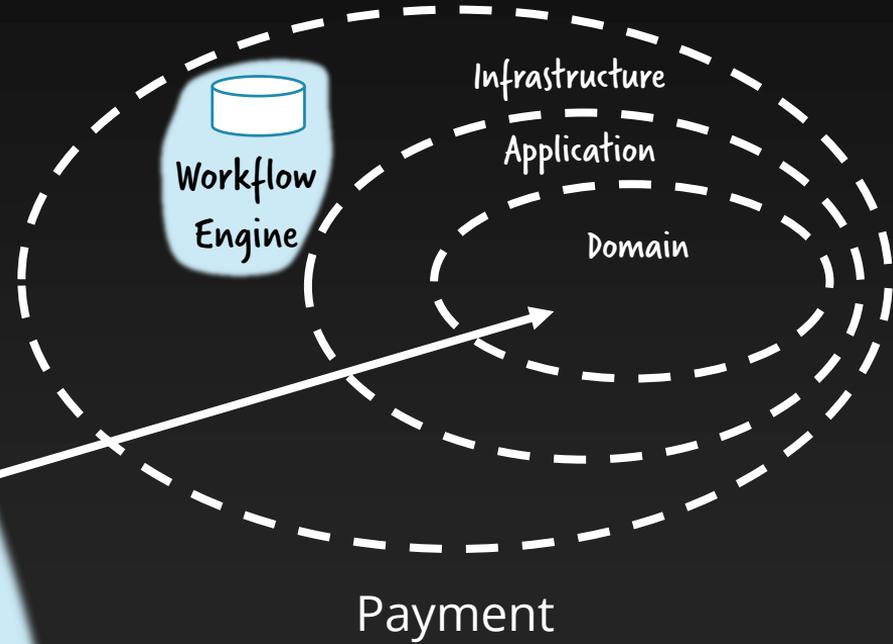
Before mapping processes explicitly with BPMN, the truth was buried in the code and nobody knew what was going on.

Jimmy Floyd, 24 Hour Fitness

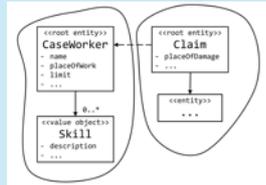
Workflows live inside service boundaries



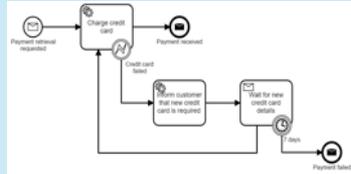
Explicit flows help separate domain and infrastructure



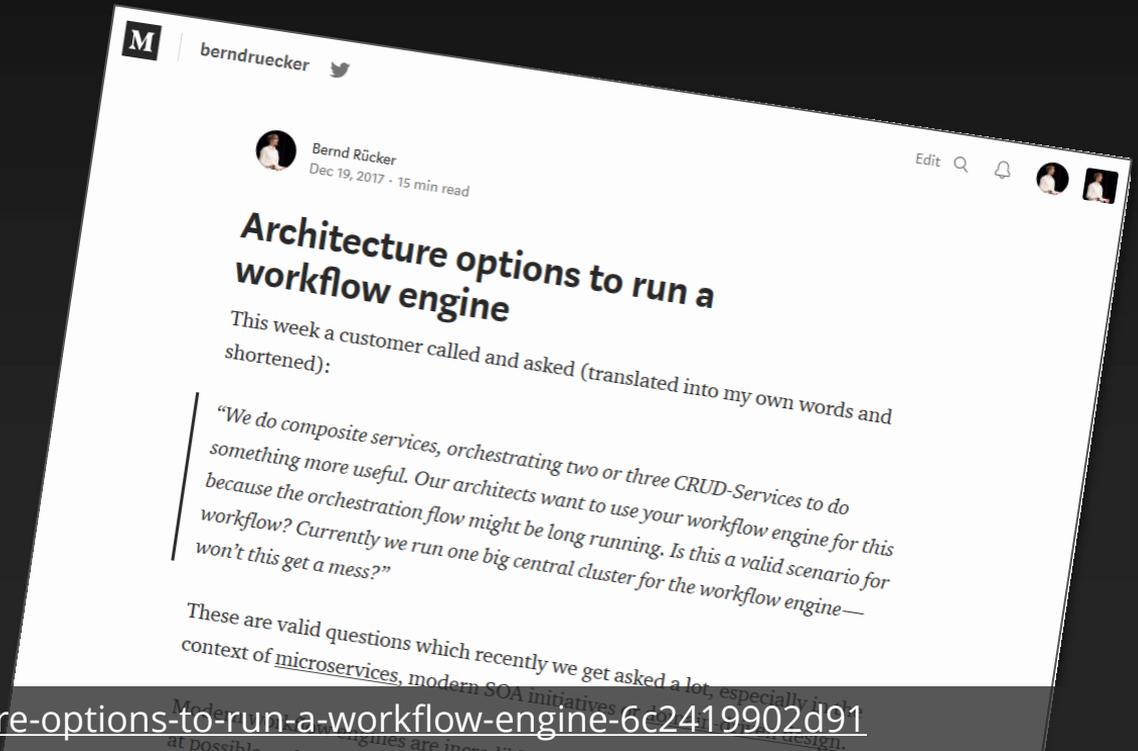
Aggregates,
Domain Events,
Domain Services,
etc ...



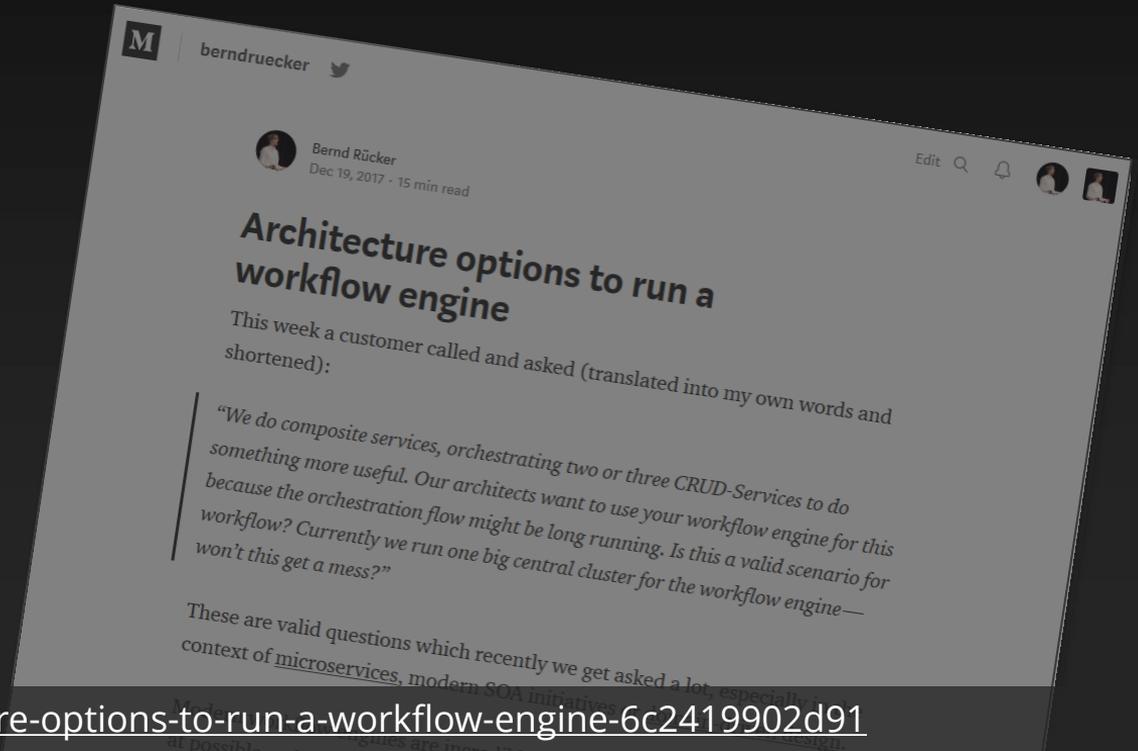
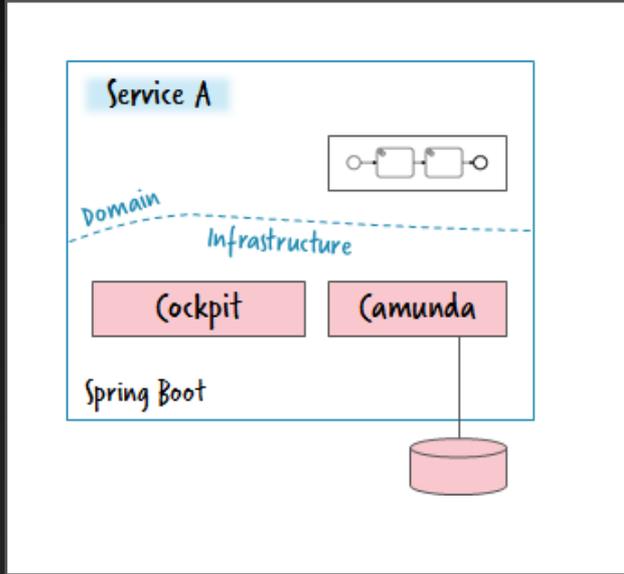
+ the flow



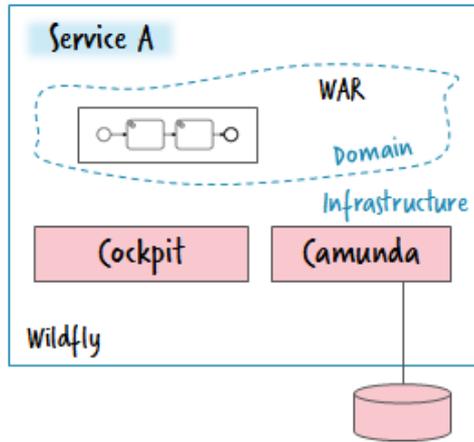
Manifold architecture options



Manifold architecture options



Manifold architecture options



berndruecker



Bernd Rucker
Dec 19, 2017 · 15 min read

Edit 🔍 🔔 👤

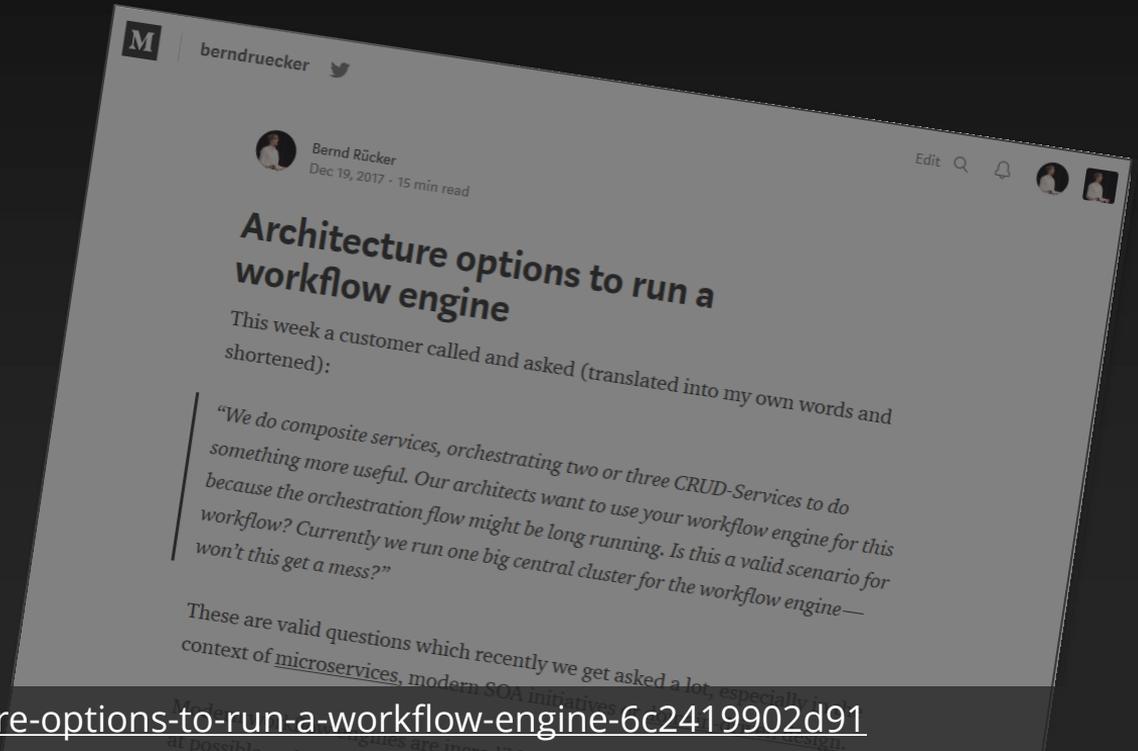
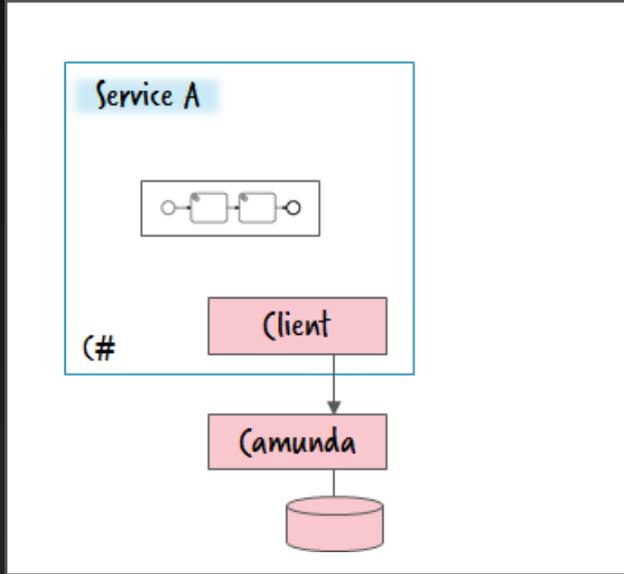
Architecture options to run a workflow engine

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

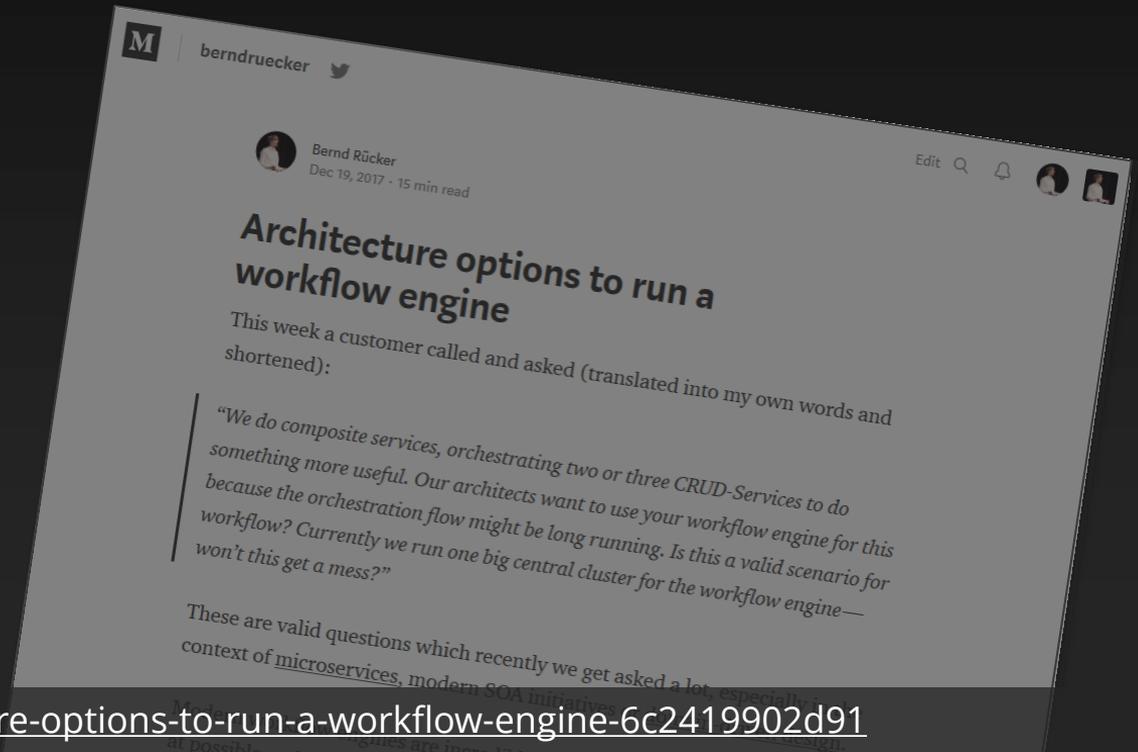
“We do composite services, orchestrating two or three CRUD-Services to do something more useful. Our architects want to use your workflow engine for this because the orchestration flow might be long running. Is this a valid scenario for workflow? Currently we run one big central cluster for the workflow 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.

Manifold architecture options



Manifold architecture options



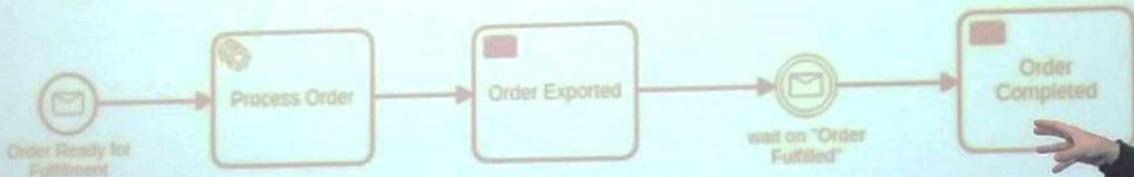
<https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91>

Lightweight workflow engines are
great – don't DIY*

*e.g. enabling potentially long-running services, solving hard
developer problems, can run decentralized

Reality check

FULLFILLMENT PROCESS



```
public void execute(final FulfillmentOrder order, final ComundaContext context) {  
    gateway.publish(new OrderExported(order, context.executionId), "fulfillment.order-exported");  
}
```



Zalando

Sales-Order and
Order-Fulfillment
via Camunda
for every order worldwide

Orders Q2-2017: 22,2 Mio.

Sales 2016: 3,6 Mrd. EUR
Growth 2016: 23%



Bernd Ruecker
@berndruecker

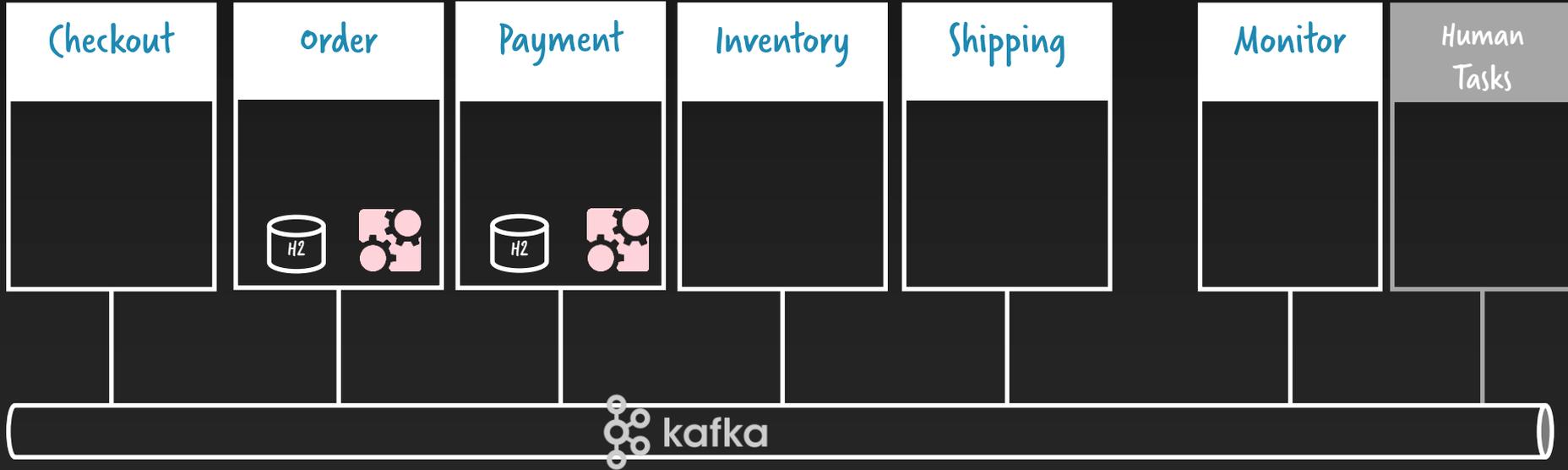


Cool! Zalando can handle more than 100k jobs per minute with [@CamundaBPM](#) and is safe for the load of the next 15 years :-)



11:07 AM - 19 Sep 2017

Code example & live demo



Events decrease coupling: sometimes

read-models, but no complex peer-to-peer event chains!

Orchestration needs to be avoided: sometimes

no ESB, smart endpoints/dumb pipes, important capabilities need a home

Workflow engines are painful: some of them

lightweight engines are easy to use and can run decentralized,
they solve hard developer problems, don't DIY

Thank you!

Meet me at
Meet the experts
Now!



Contact: bernd.ruecker@camunda.com
[@berndruecker](#)

Slides: <https://bernd-ruecker.com>

Blog: <https://blog.bernd-ruecker.com>

Code: <https://github.com/flowing>

InfoWorld
FROM IDG

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

InfoQ
neue

<https://www.infoq.com/articles/events-workflow-automation>



With thoughts from <http://flowing.io>
[@berndruecker](#) | [@martinschimak](#)