

Complex event flows in distributed systems

@berndruecker



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

3 common hypotheses I check today:

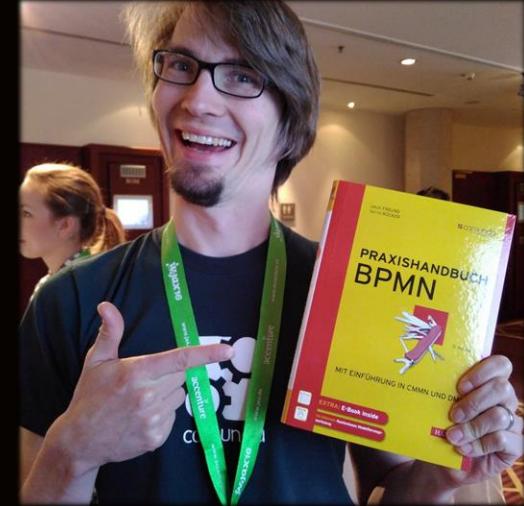
Events decrease coupling

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](https://twitter.com/berndruecker)





ORCHESTRATING A HIGHLY-SCALABLE FULFILLMENT PROCESS

JÖRN HORSTMANN
LUKAS NIEMEIER

2017-05-19



Simplified example: dash button



Photo by 0xF2, available under Creative Commons BY-ND 2.0 license. <https://www.flickr.com/photos/0xf2/29873149904/>

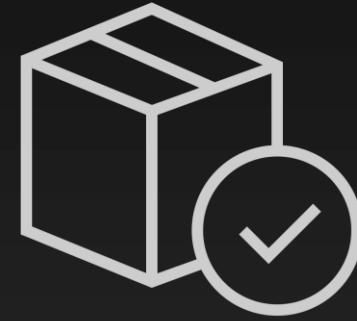
Three steps...



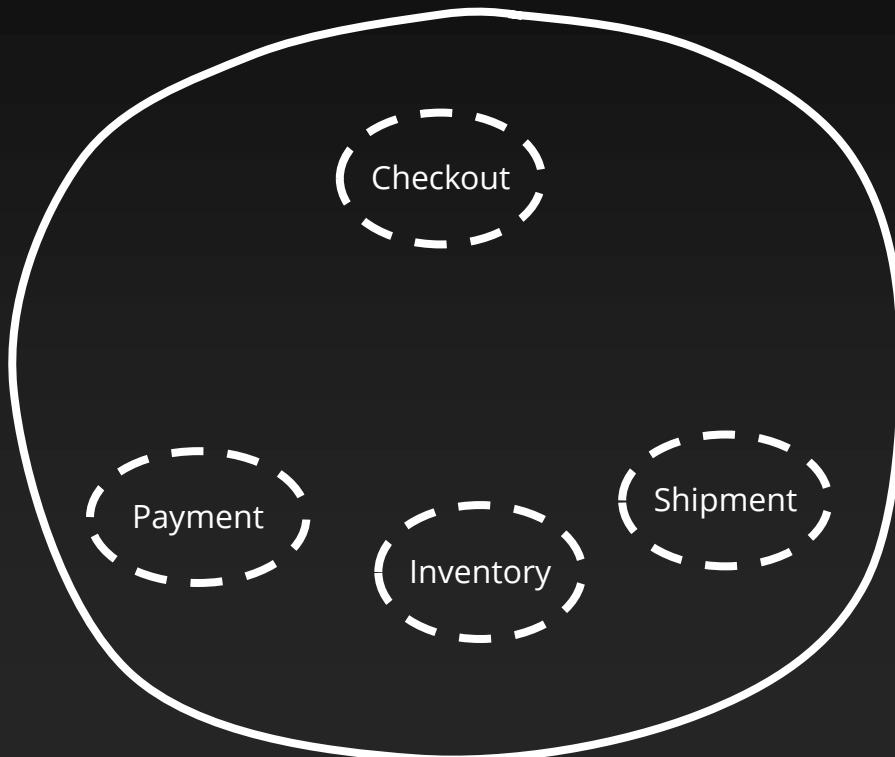
Pay item

Fetch item

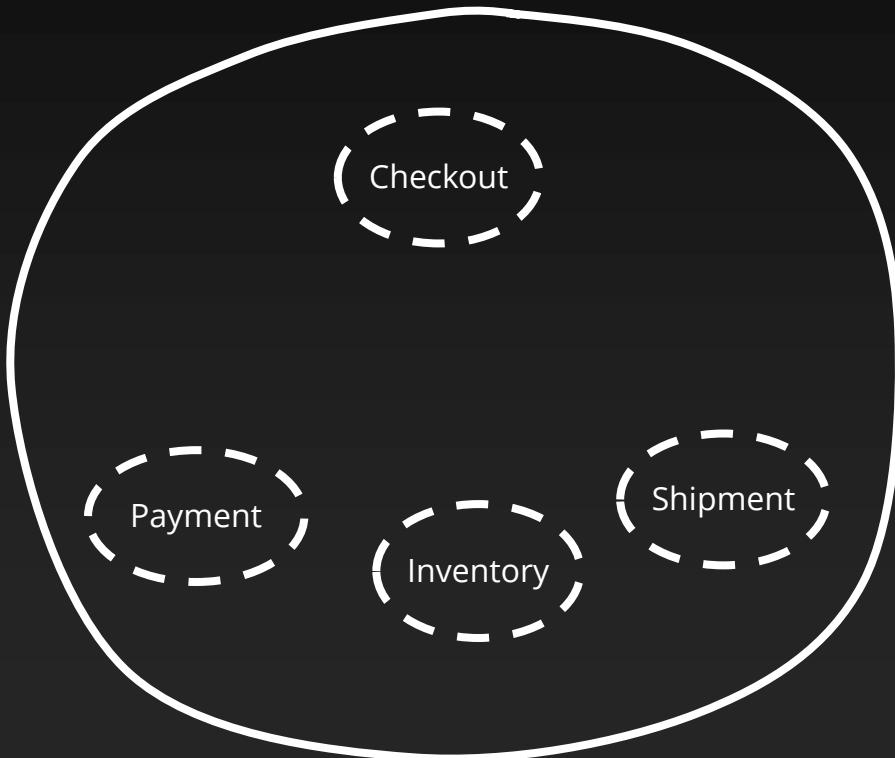
Ship item



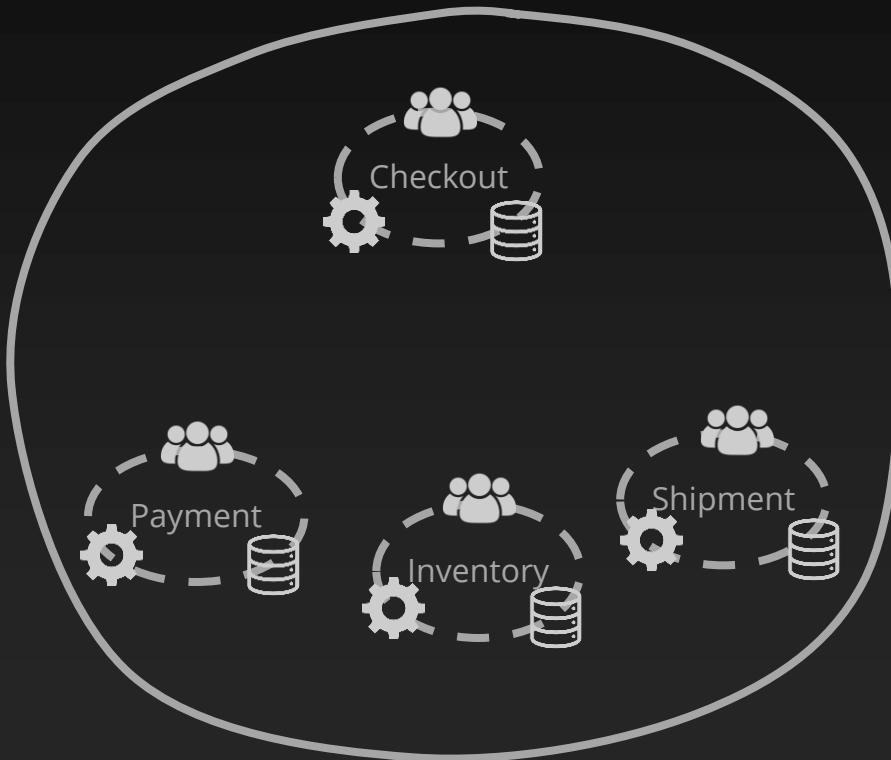
Who is involved? Some bounded contexts...



(Micro-)Services



Autonomous (micro-)services

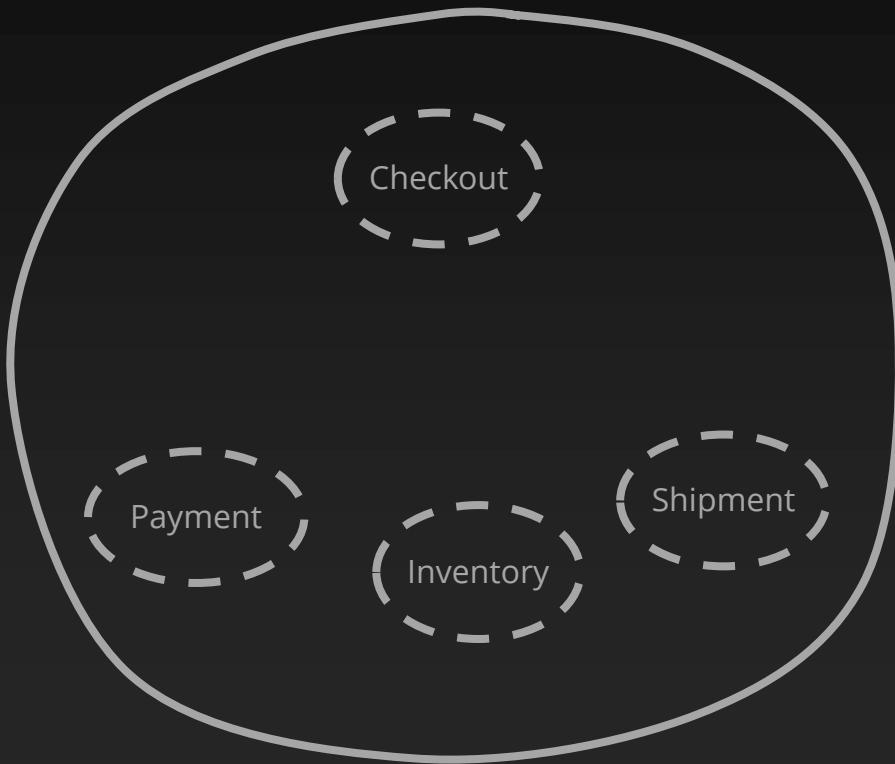


- Dedicated Application Processes
- Dedicated infrastructure
- Dedicated Development Teams

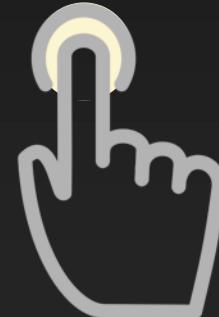
Events decrease coupling



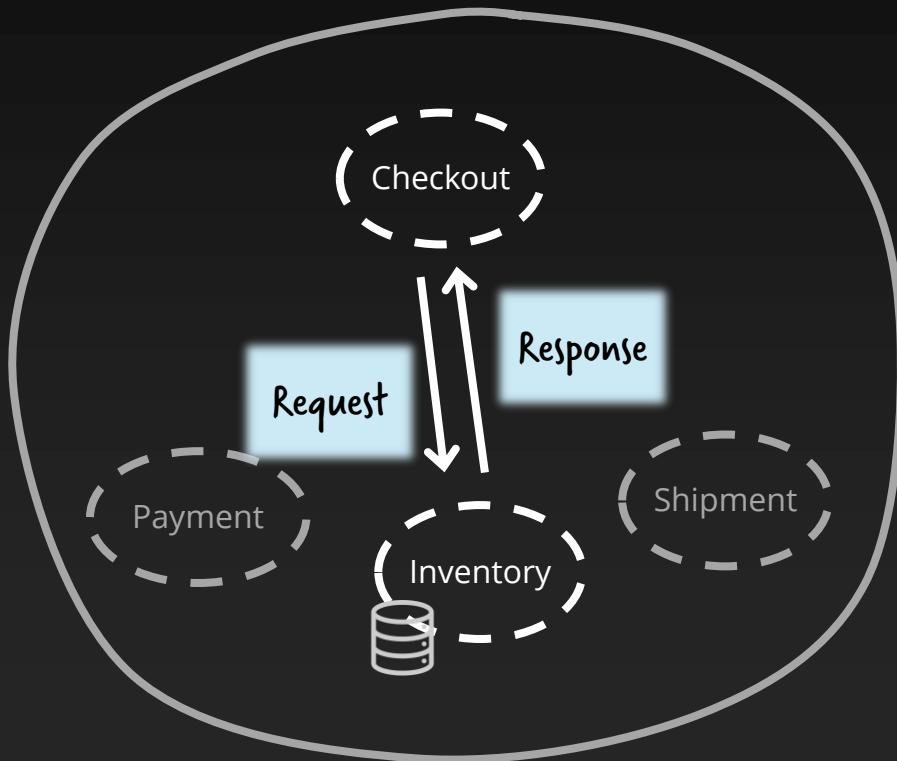
Example



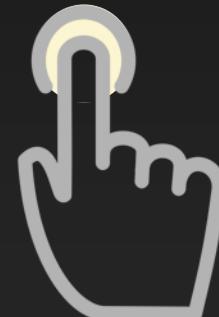
The button blinks if we can
ship within 24 hours



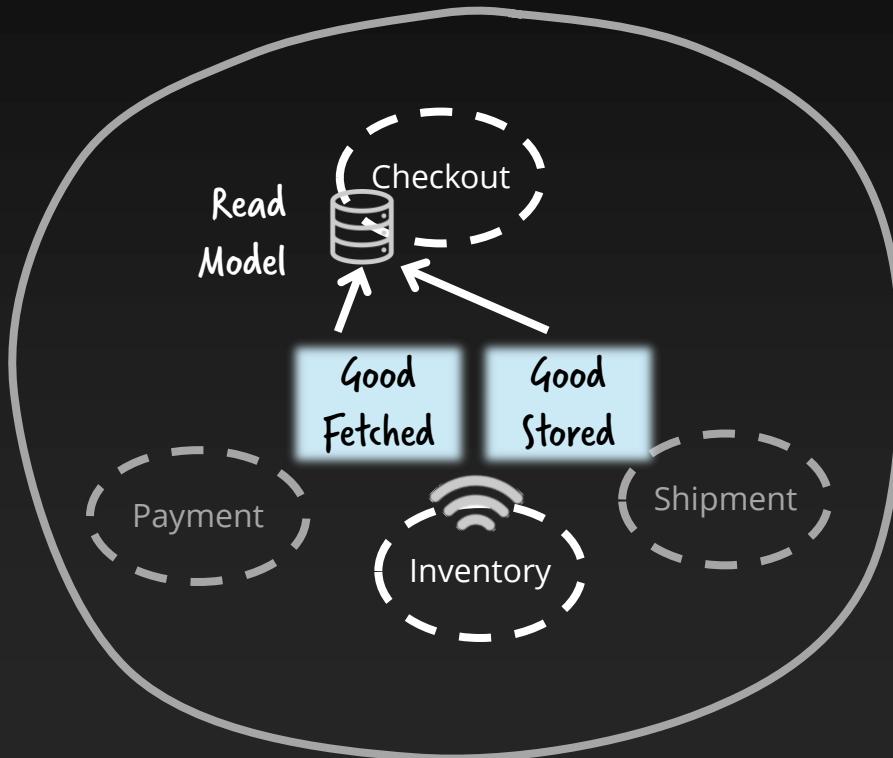
Request/response: temporal coupling



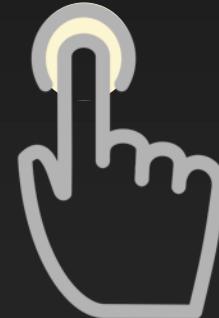
The button blinks if we can ship within 24 hours



Temporal decoupling with events and read models



The button blinks if we can
ship within 24 hours

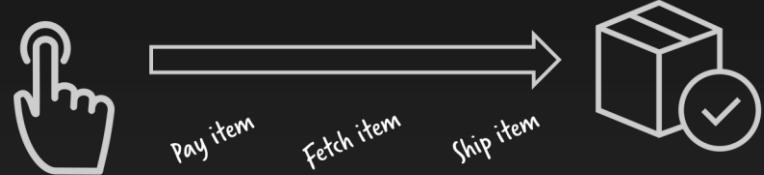
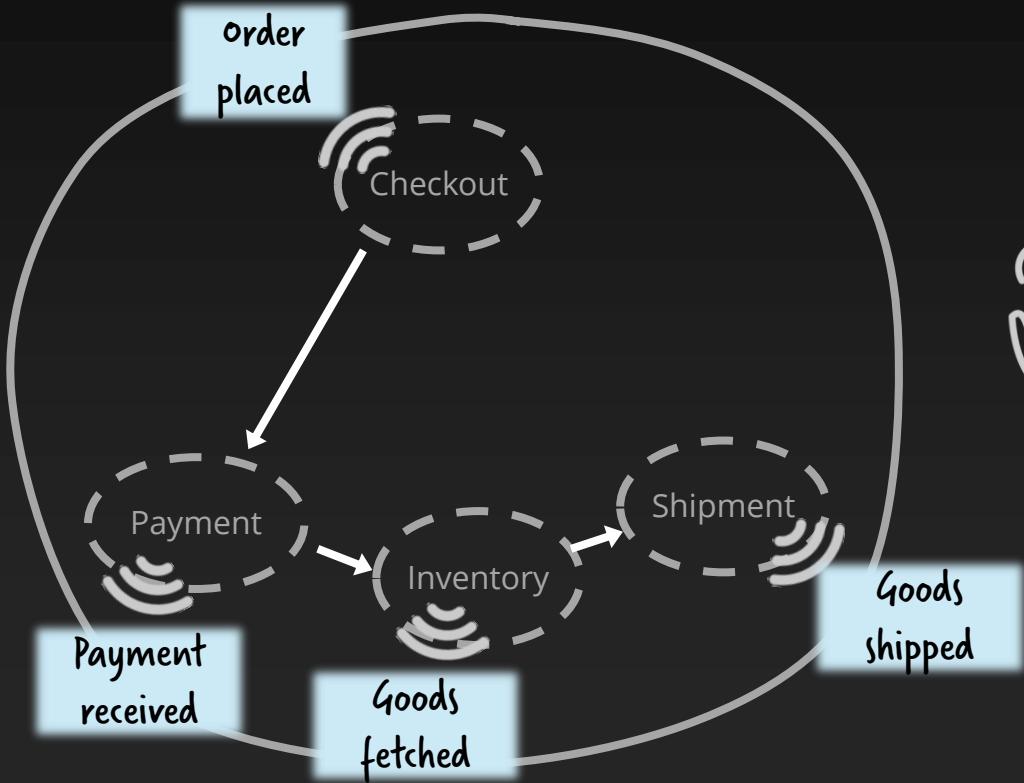


*Events are facts about what happened (in the past)

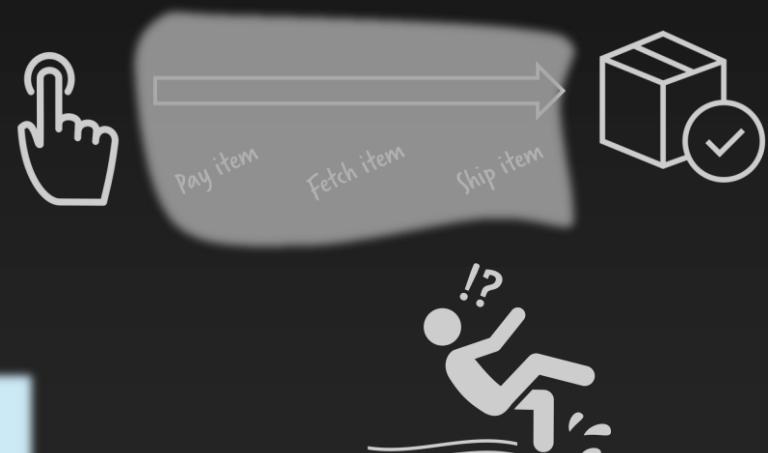
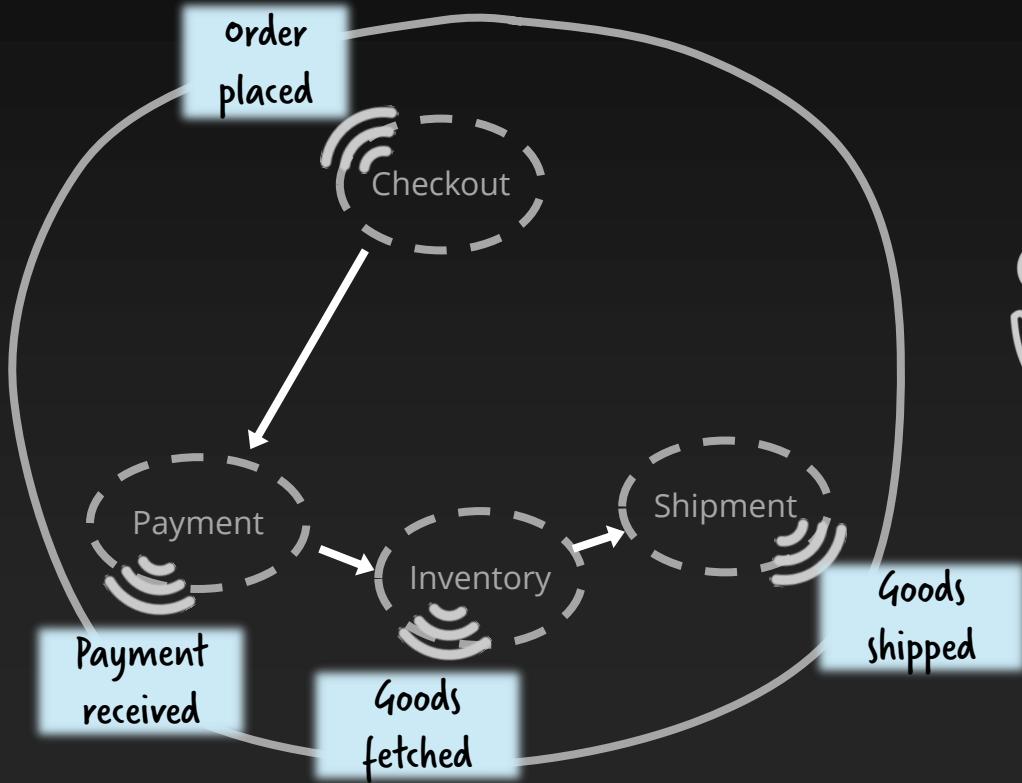
Events can decrease coupling*

*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.

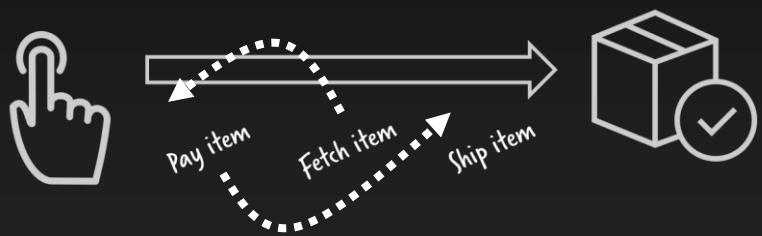
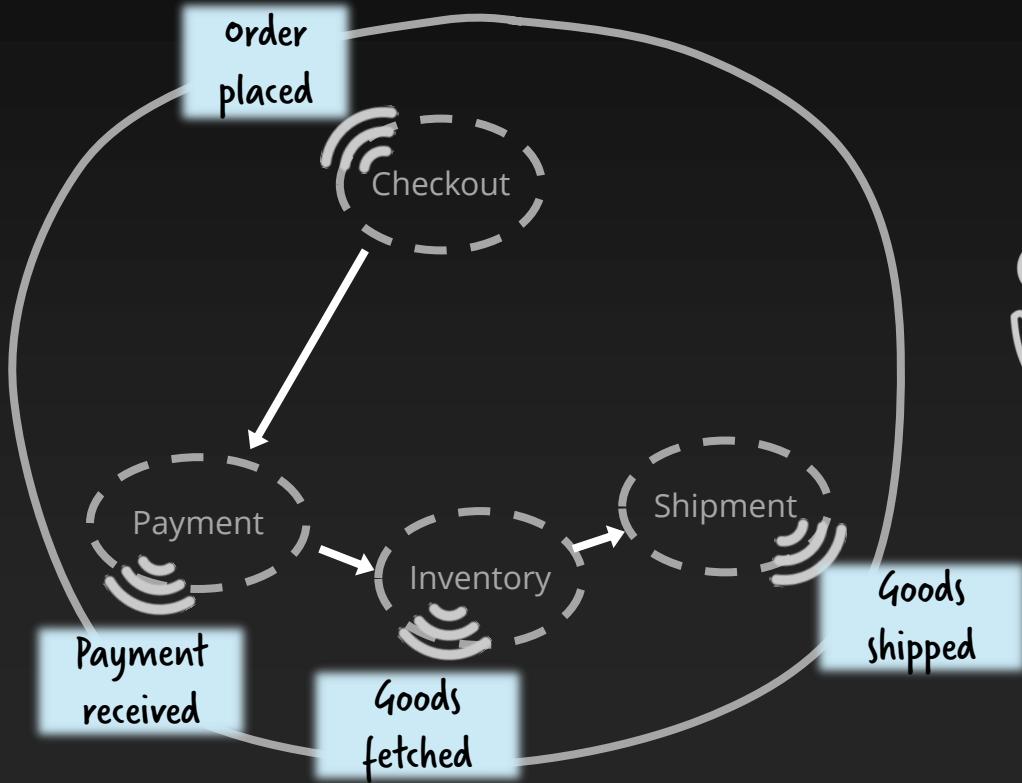


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

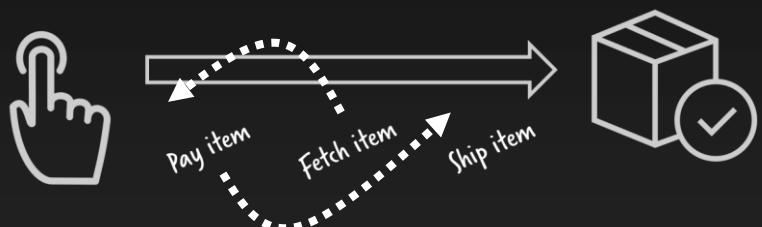
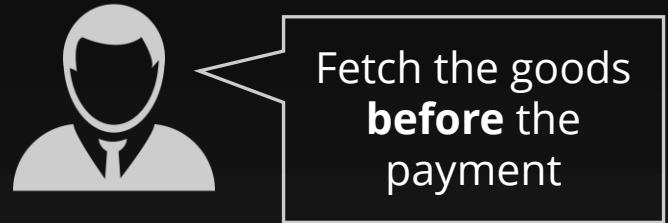
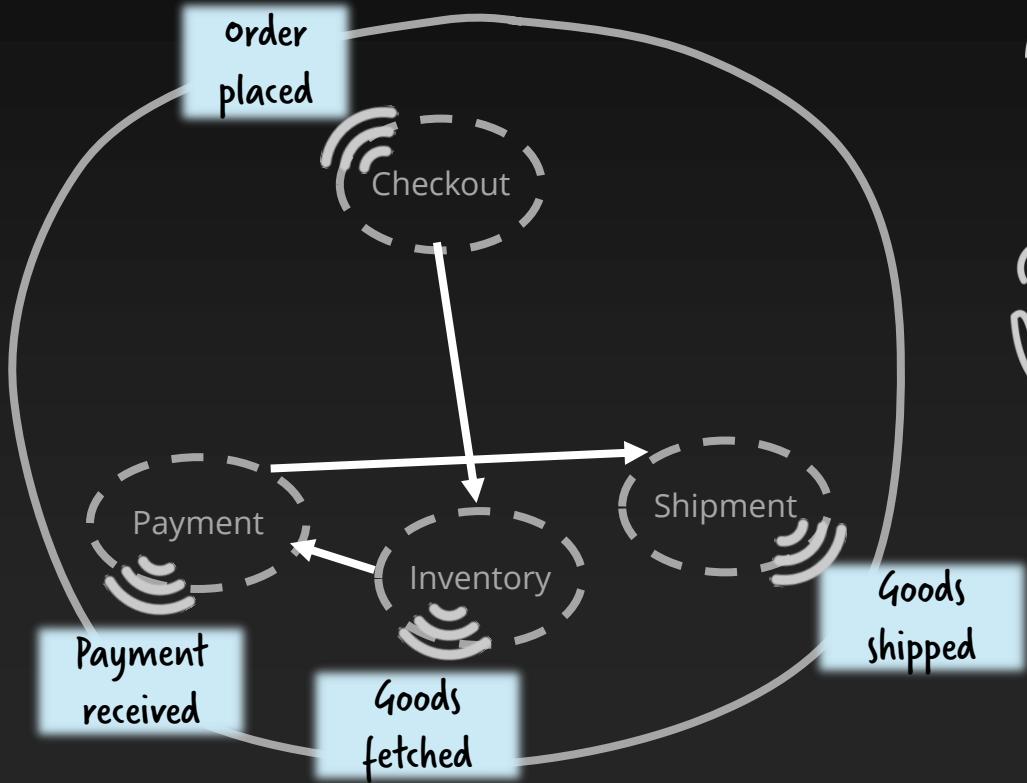


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

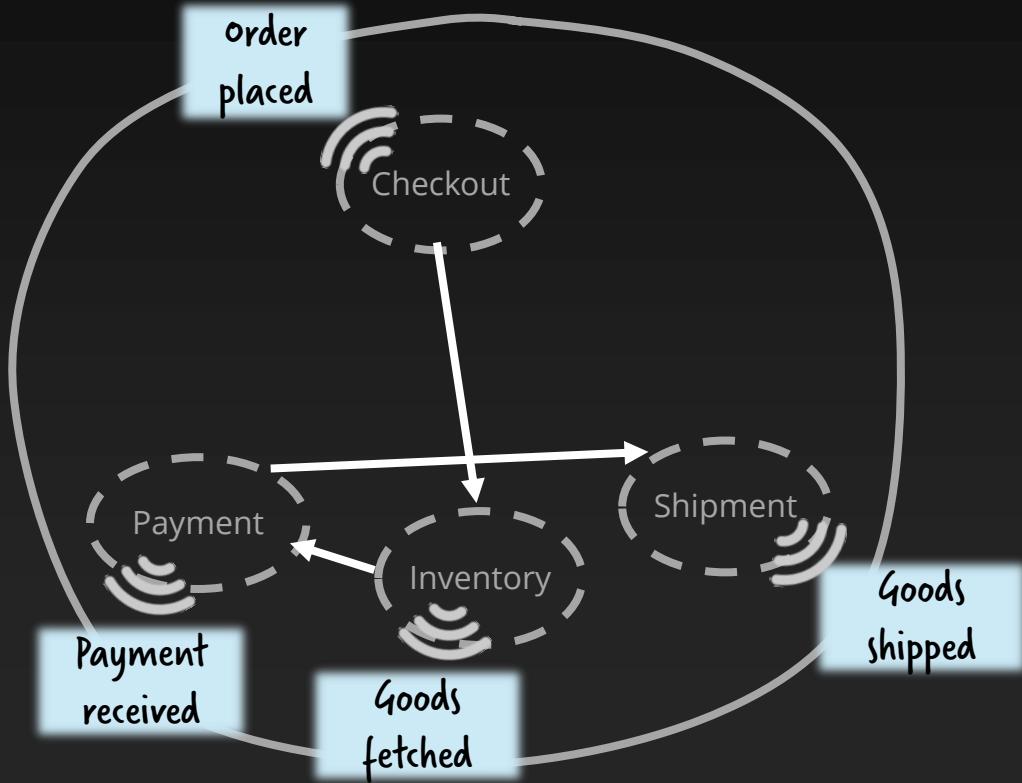
Peer-to-peer event chains



Peer-to-peer event chains



Peer-to-peer event chains



Fetch the goods
before the payment

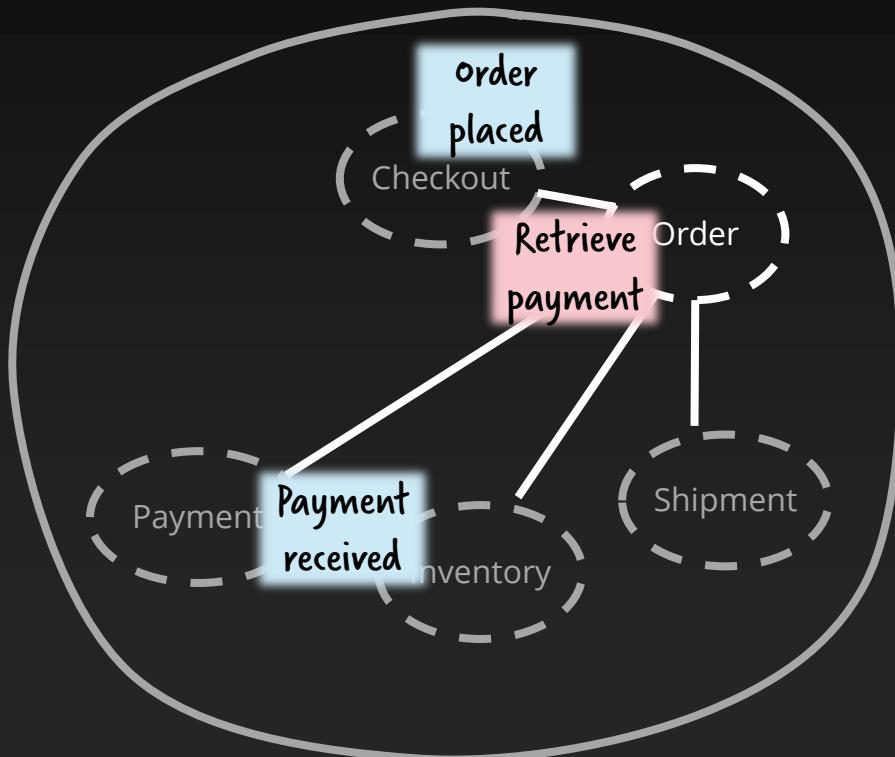
Customers can pay via invoice

...



Photo by born1945, available under [Creative Commons BY 2.0 license](#).

Extract the end-to-end responsibility



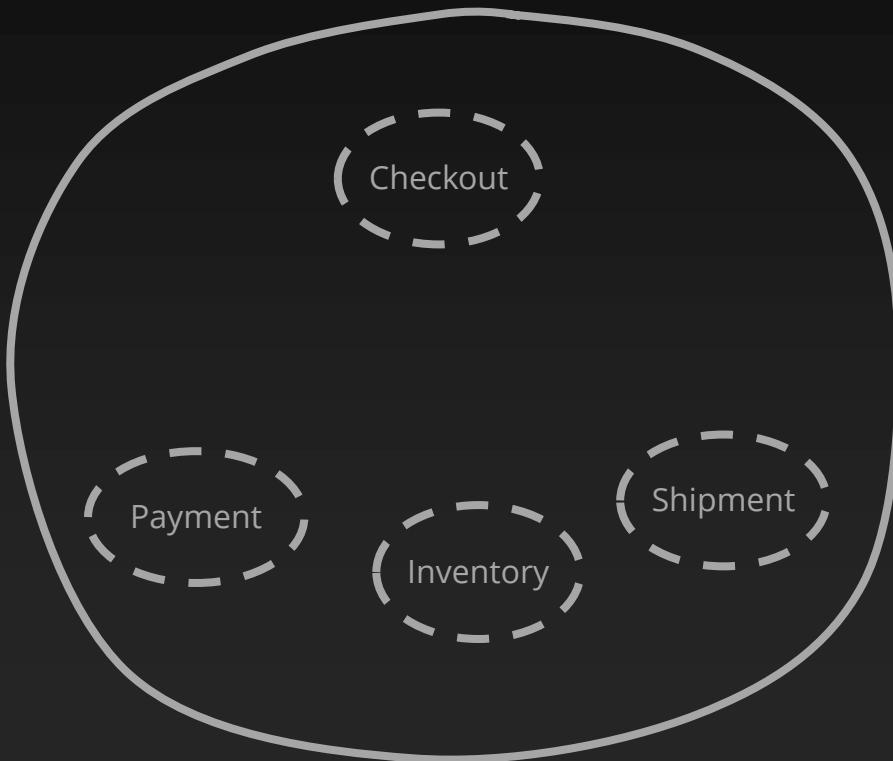
***Commands** have an intent about what needs to happen in the future

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

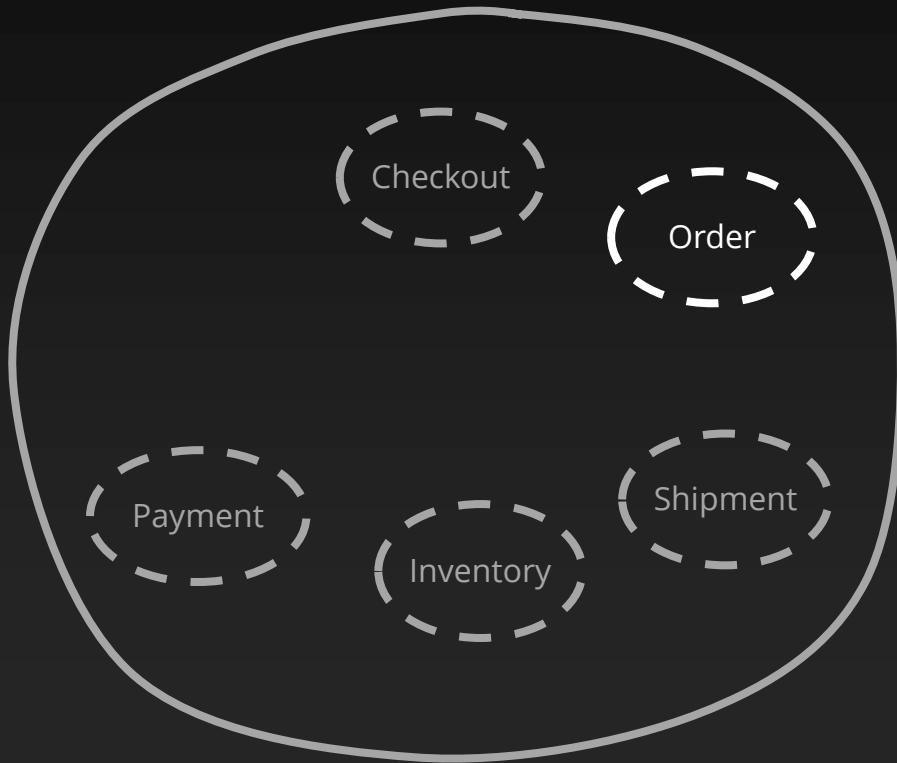


orchestration needs to be avoided

Smart ESB-like middleware



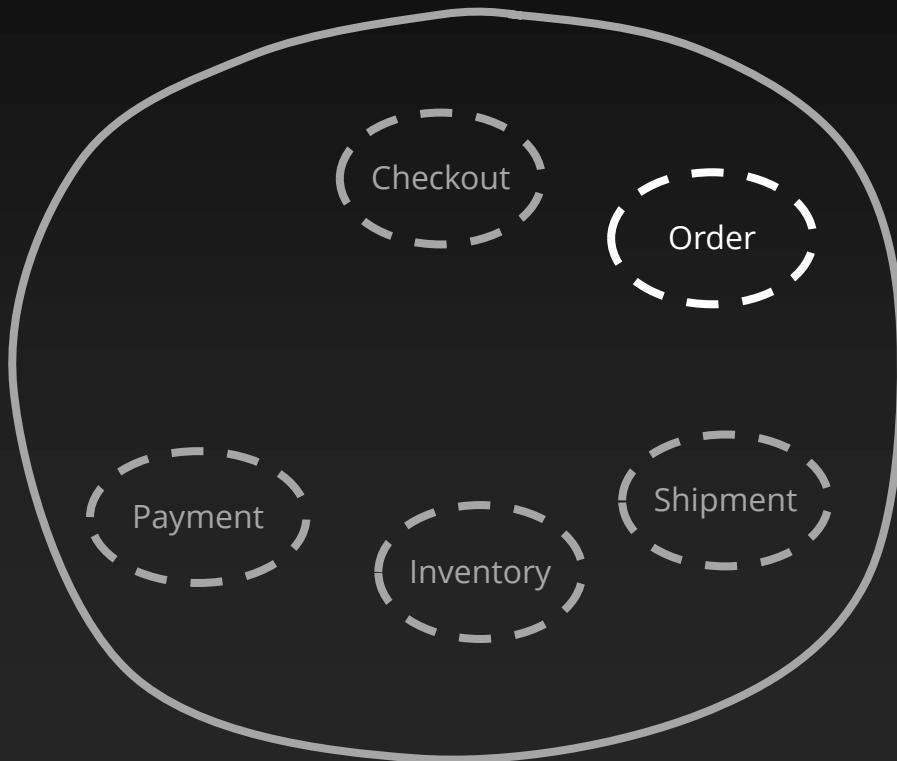
Dumb pipes



Martin Fowler

Smart endpoints
and **dumb pipes**

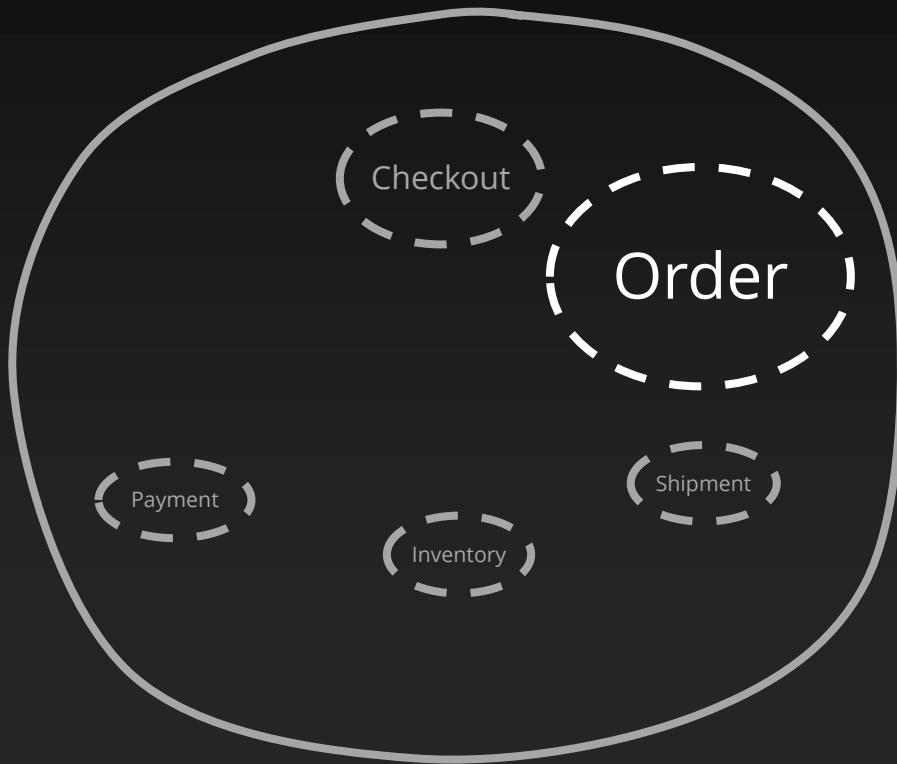
Danger of god services?



Sam Newmann

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

Danger of god services?



Sam Newmann

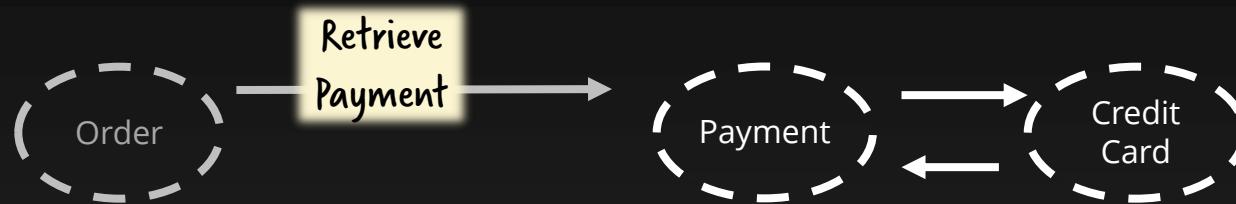
A few
smart god services
tell
anemic CRUD services
what to do

A god service is only created
by bad API design!

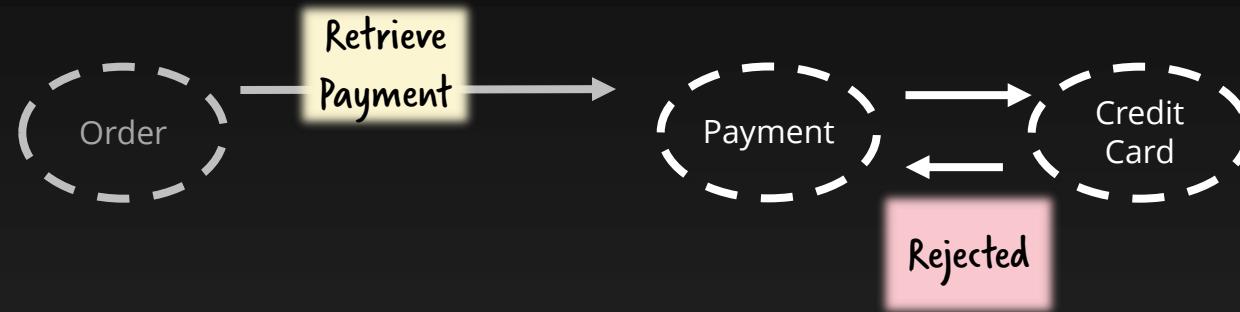
Example



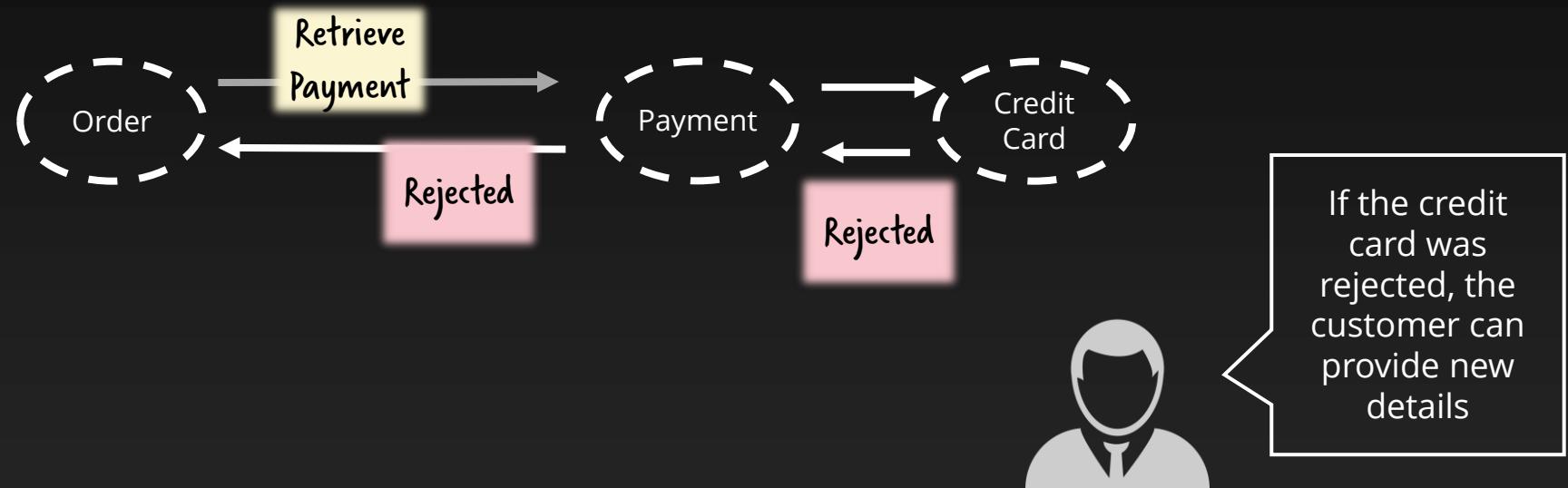
Example



Example

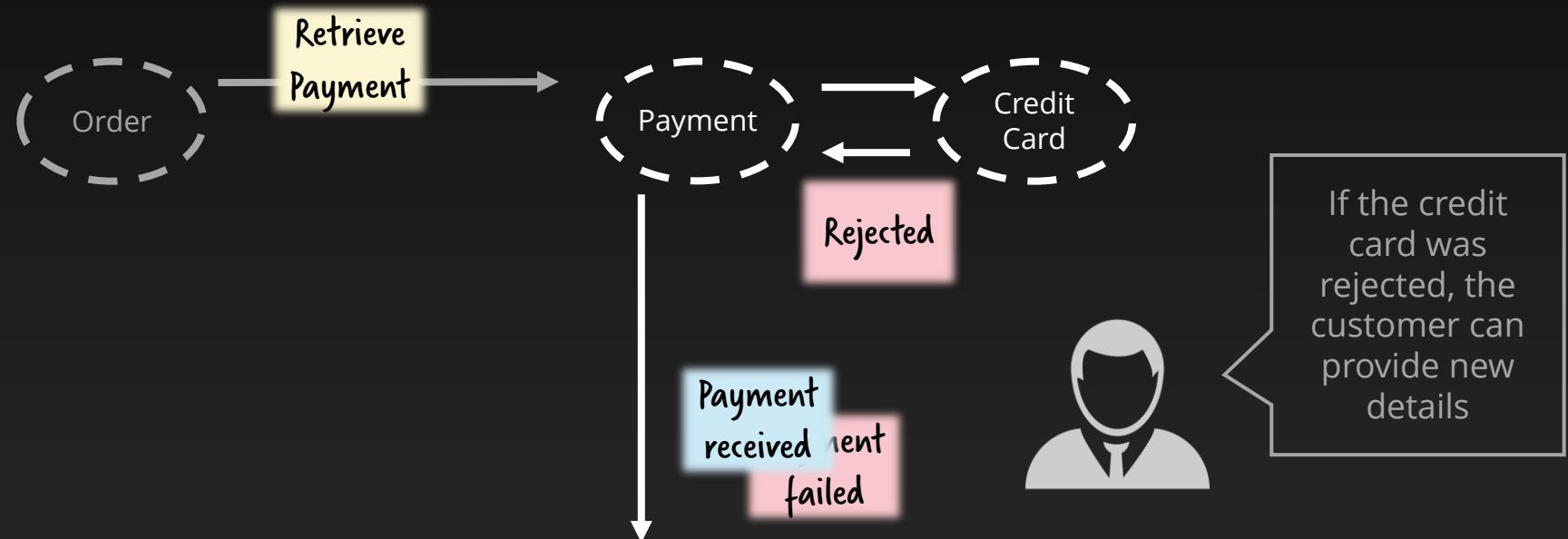


Example

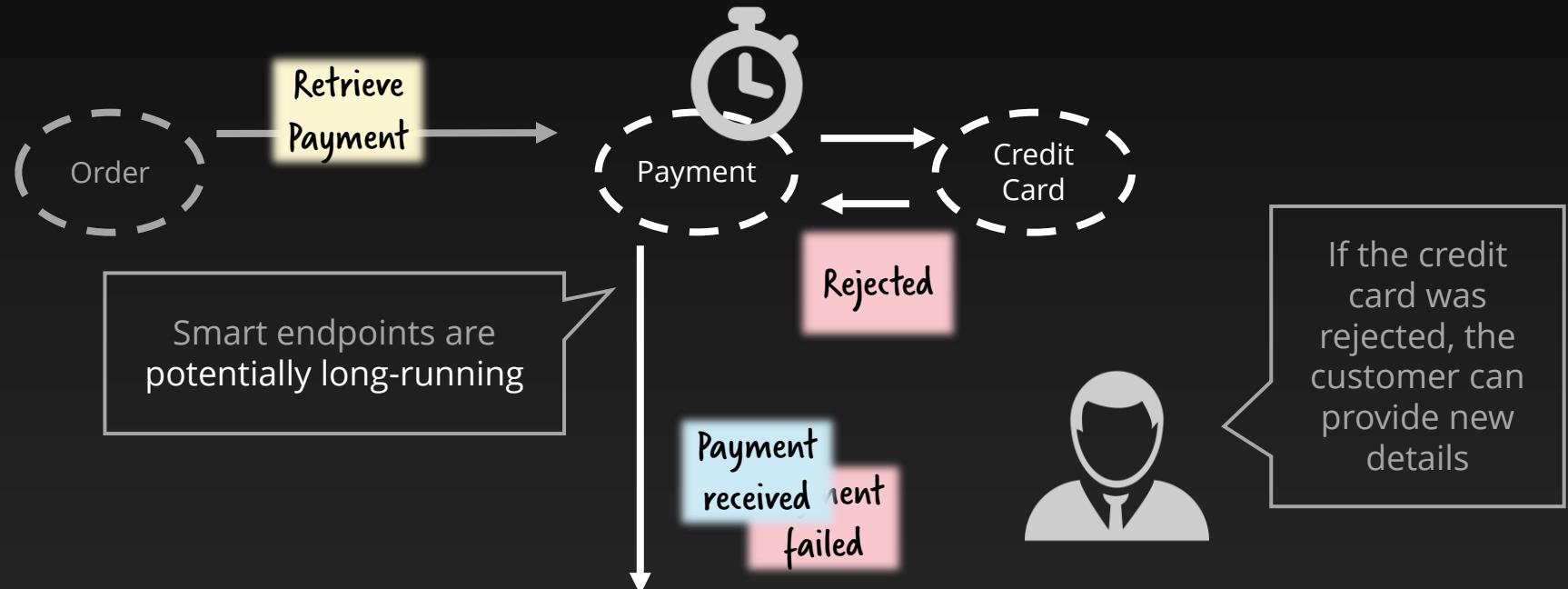


Client of **dumb endpoints** easily become a god services.

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](#).



Buchen

„There was an error
while sending your
boarding pass“

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

Ihre Bordkarten

Ihr Buchungscode **08HHSS**

Hinflug

BERND RUECKER

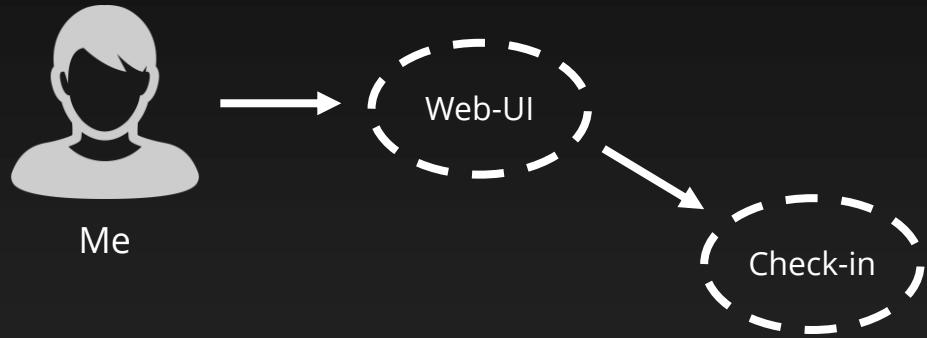
Stuttgart (STR) - London-Stansted (STN)



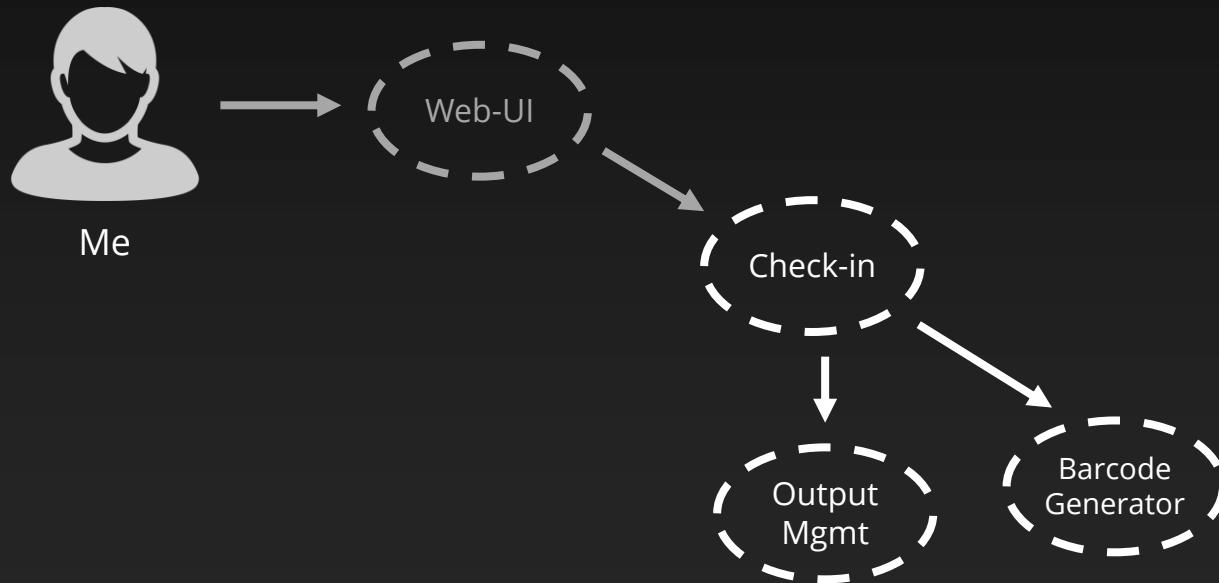
Mon 20.11.2017 10:10 - 10:45

Beim Versenden der Bordkarte ist ein Fehler aufgetreten.

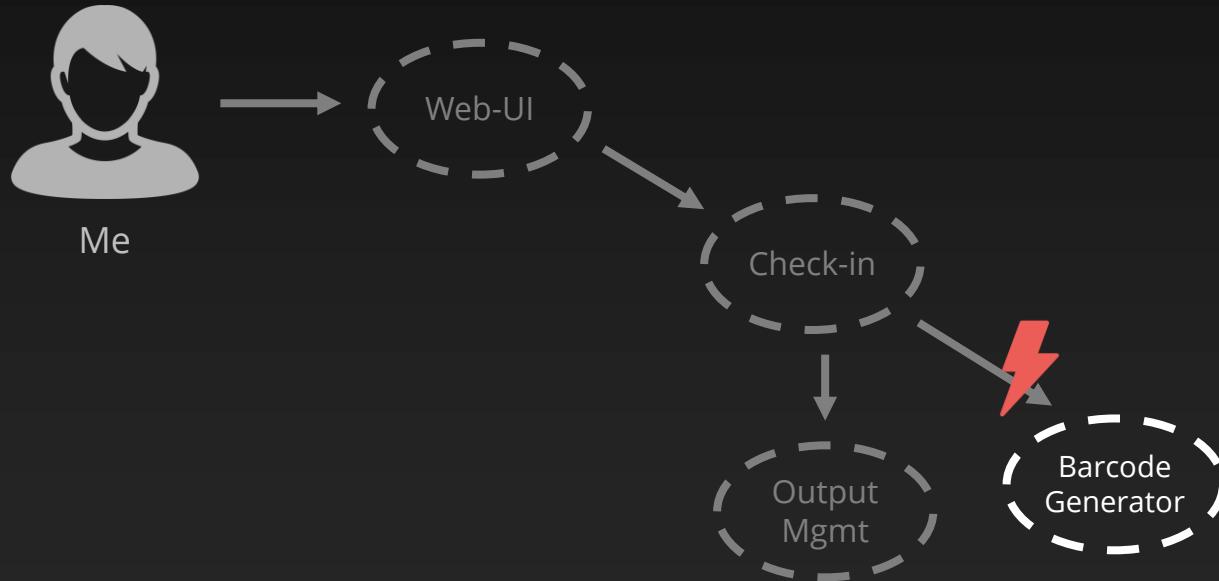
(urrent situation



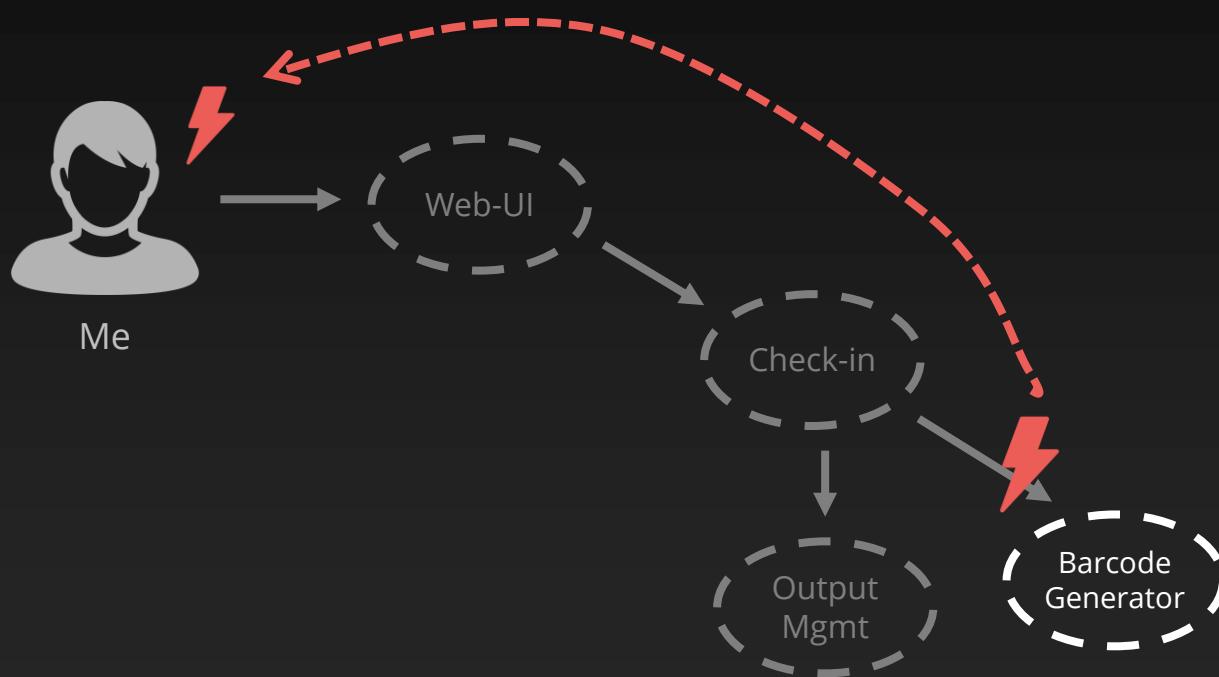
(current situation



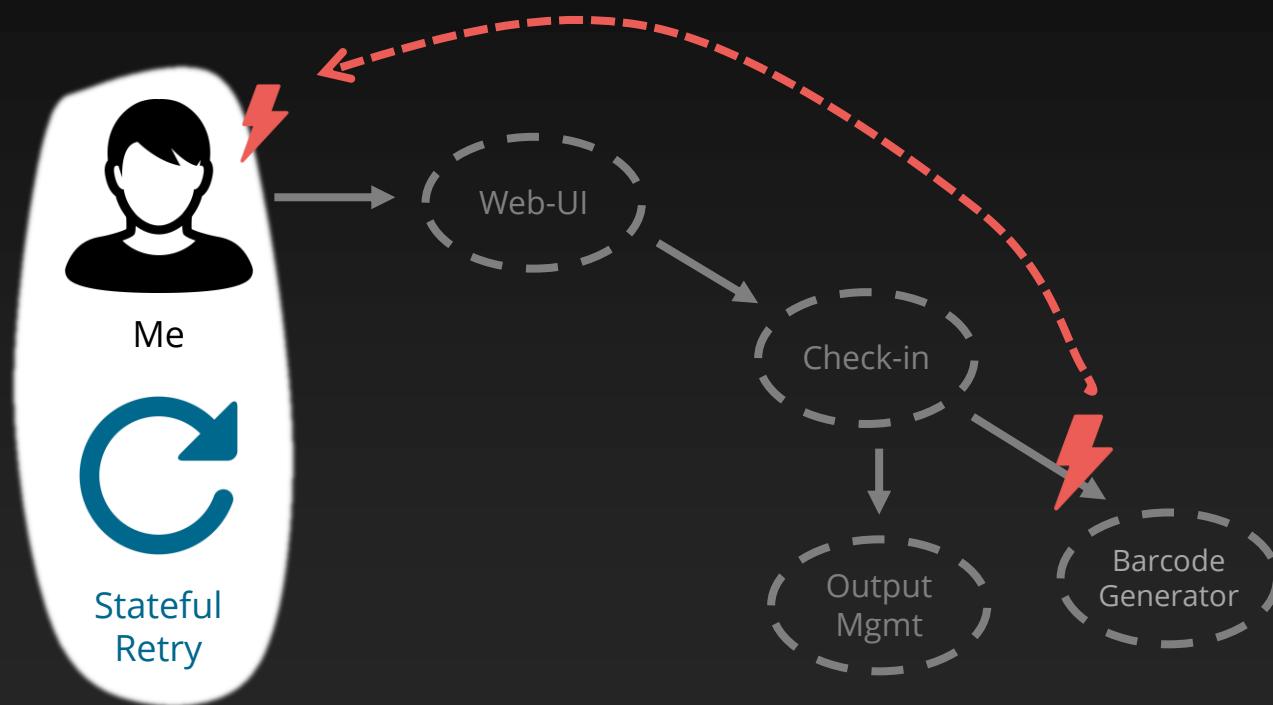
(current situation



(current situation – the bad part



(current situation – the bad part



Ihre B

easyJet

Ihr Buchun

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)



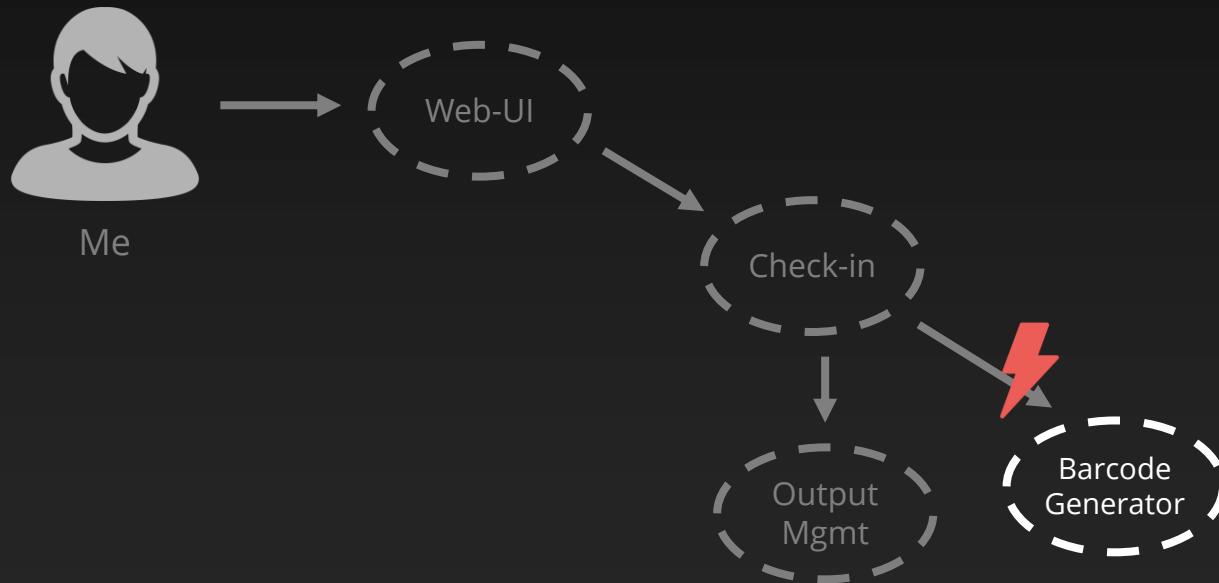
...I just made this up...

We're sorry

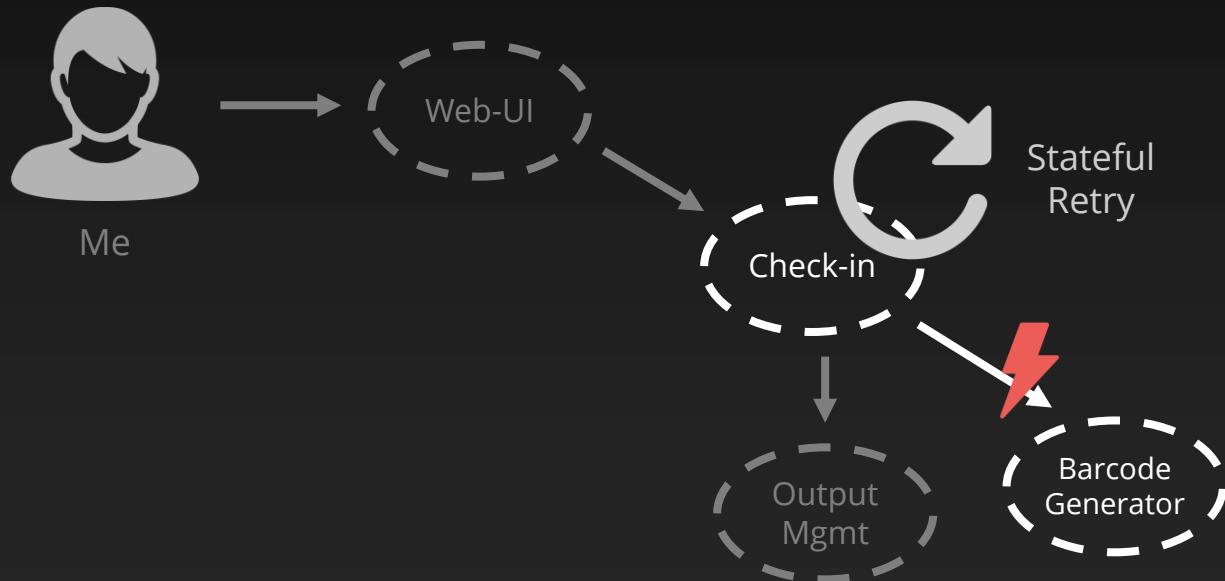
We are having some technical difficulties and cannot present you your boarding pass right away.

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

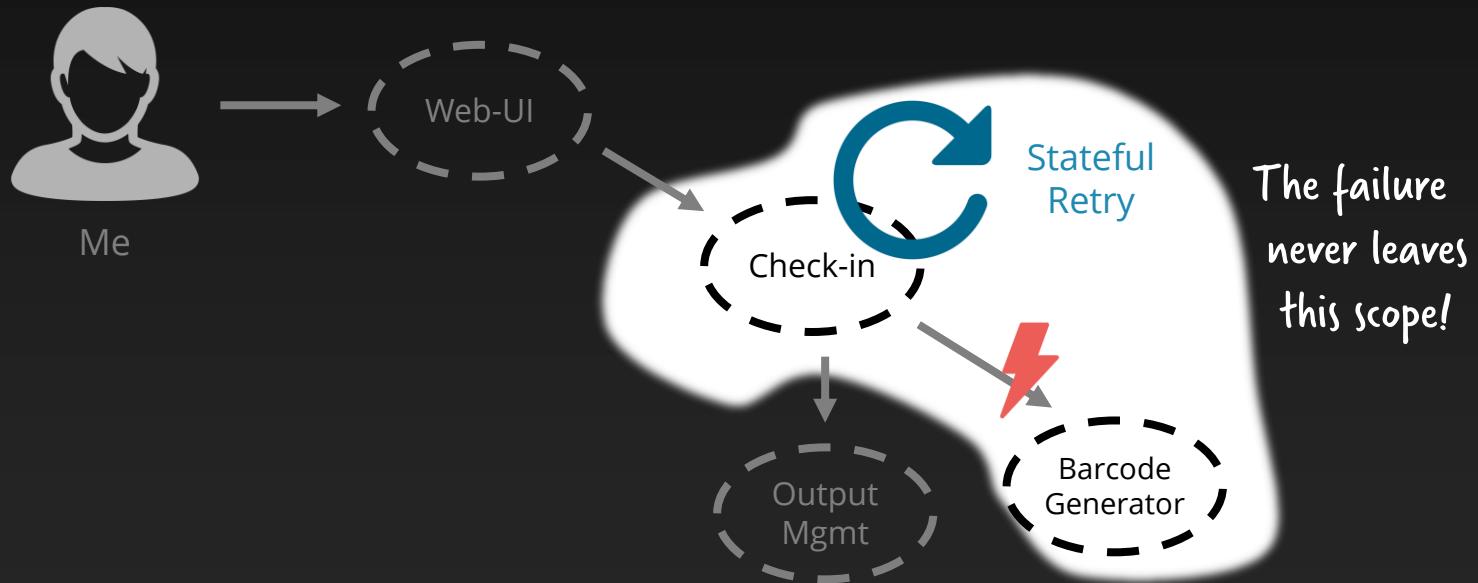
Possible situation – much better!



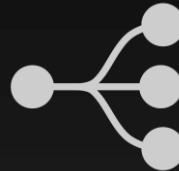
Possible situation – much better!



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!

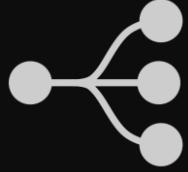


Low-code is great!
(You can get rid
of your developers!)



Death by properties panel

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

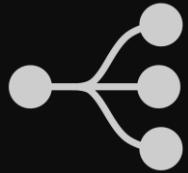


Workflow engines,
state machines



AWS Step
Functions

It is
relevant
in modern
architectures



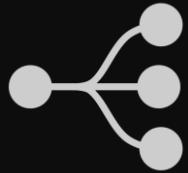
Workflow engines,
state machines



AWS Step
Functions

Silicon valley
has recognized





Workflow engines,
state machines



AWS Step
Functions

UBER

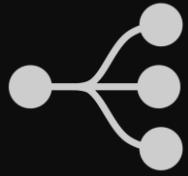
CADENCE

There are
lightweight open source
options



camunda





Workflow engines,
state machines



AWS Step
Functions

UBER

CADENCE

also at scale

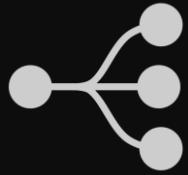
 zeebe
by Camunda

 camunda

 conductor
NETFLIX OSS

 Activiti™

 jBPM



Workflow engines,
state machines



for todays demo



AWS Step
Functions



camunda



```
public static void main(String[] args) {  
    ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()  
        .buildProcessEngine();  
  
    engine.getRepositoryService().createDeployment() //  
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow") //  
            .startEvent()  
            .serviceTask("Step  
            .serviceTask("Step  
            .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();

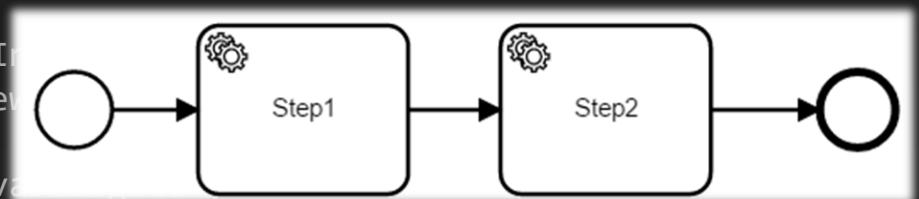
    engine.getRepositoryService().createDeployment() // Define flow
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow") e.g. in Java
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)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

    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"));
    }
}
```

```
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().startProcessInstance("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();

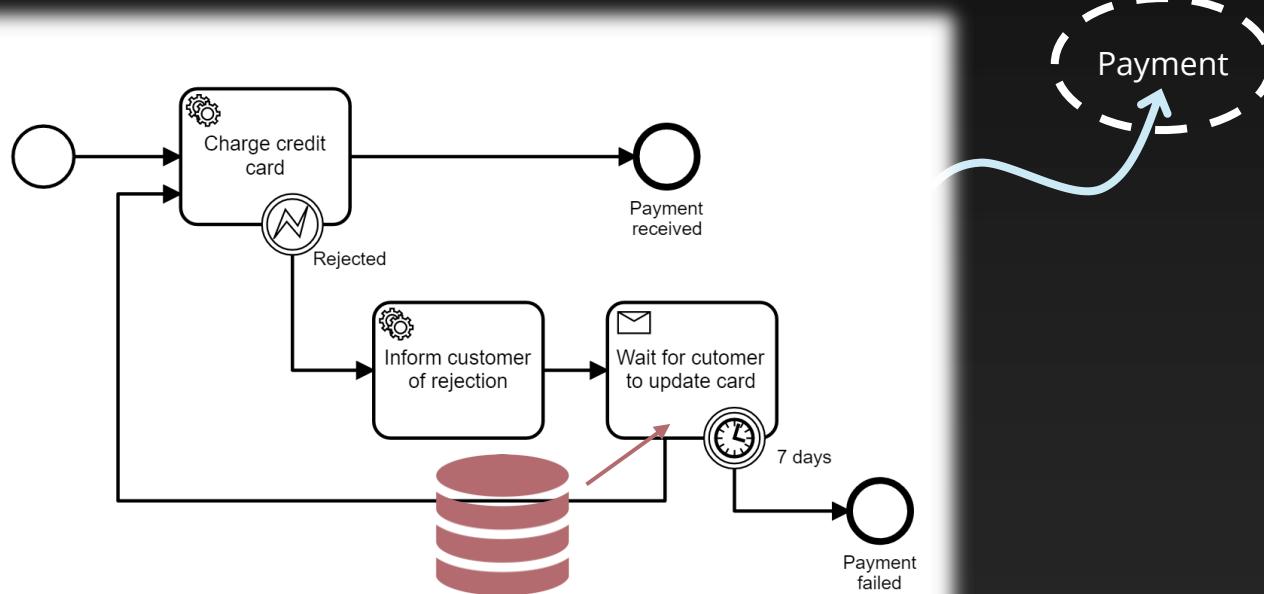
    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
        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

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

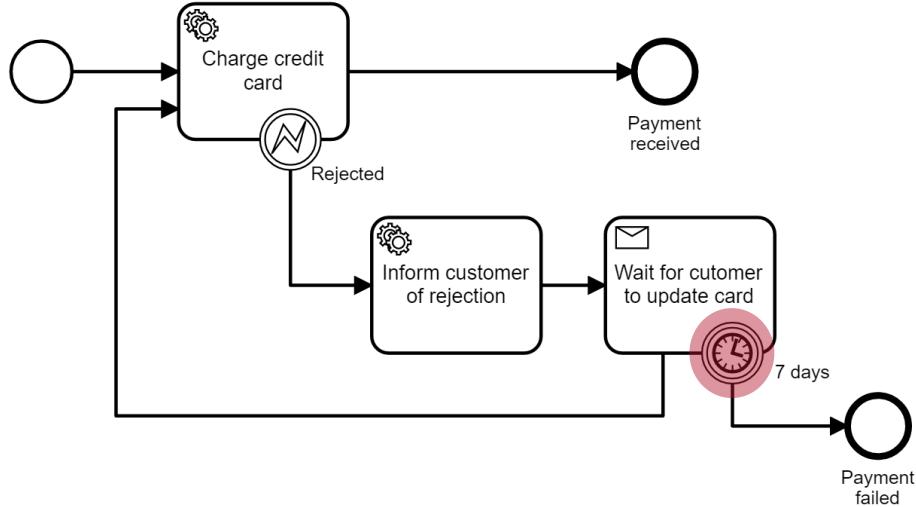
public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

Start instances

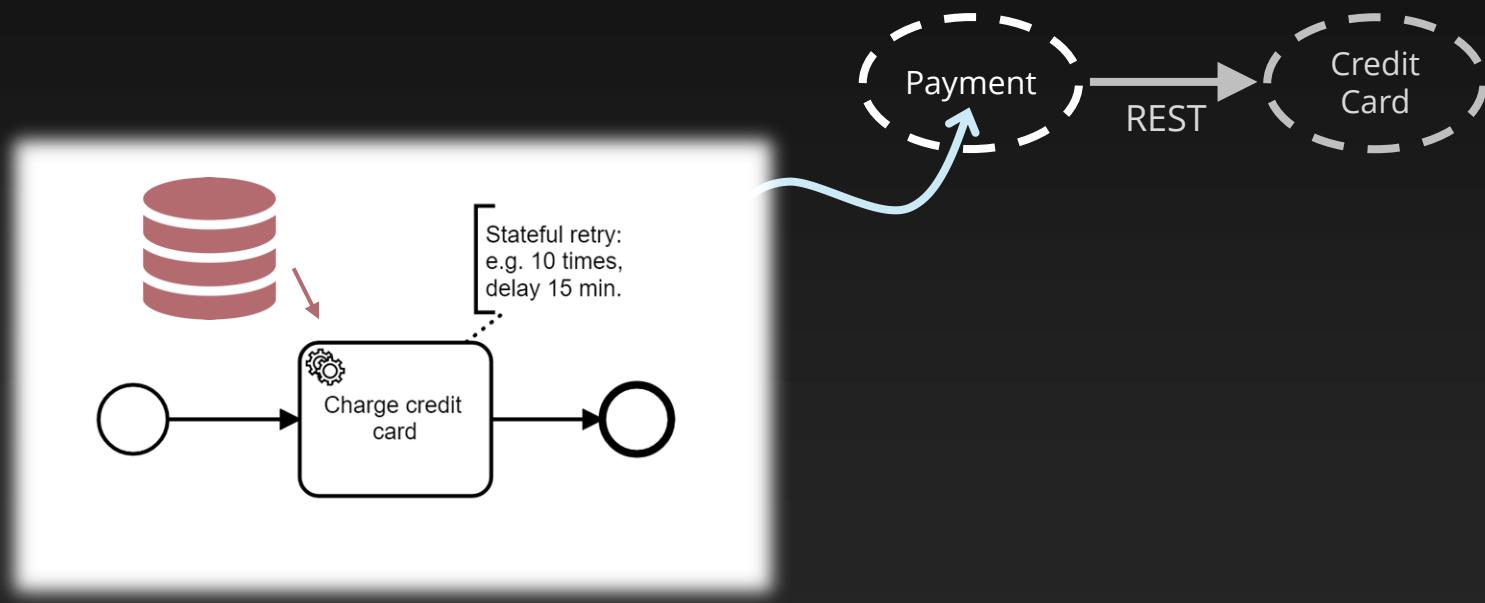
Now you have a state machine!



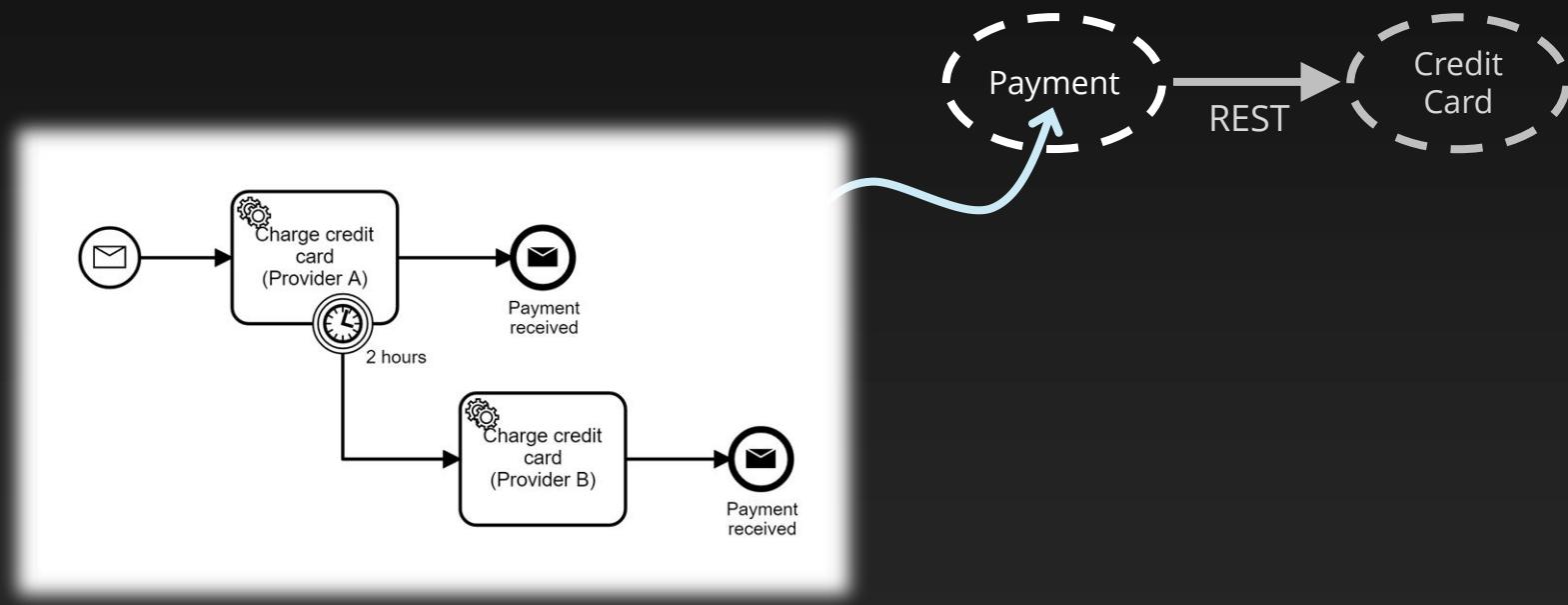
Easy to handle time



Stateful retries



Fallbacks increase resilience

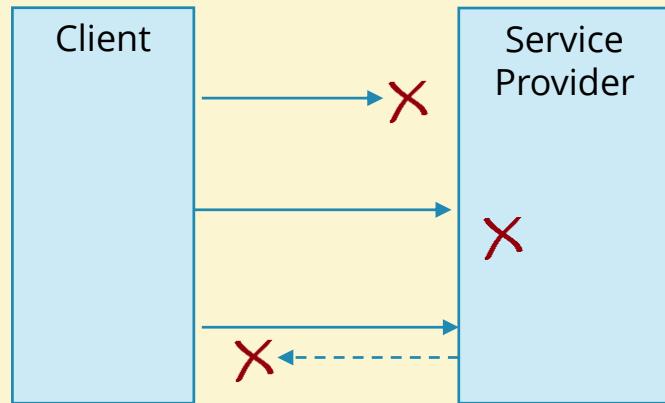


Distributed systems

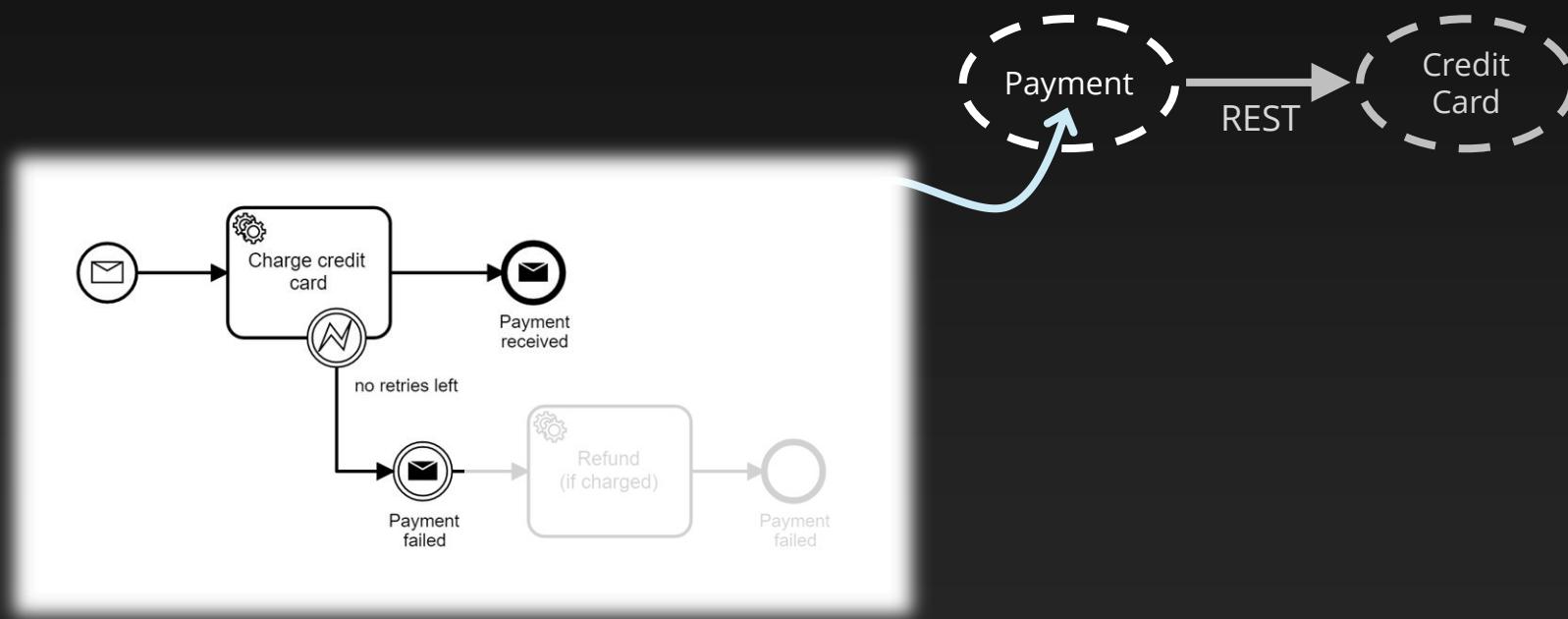


It is impossible to
differentiate certain
failure scenarios.

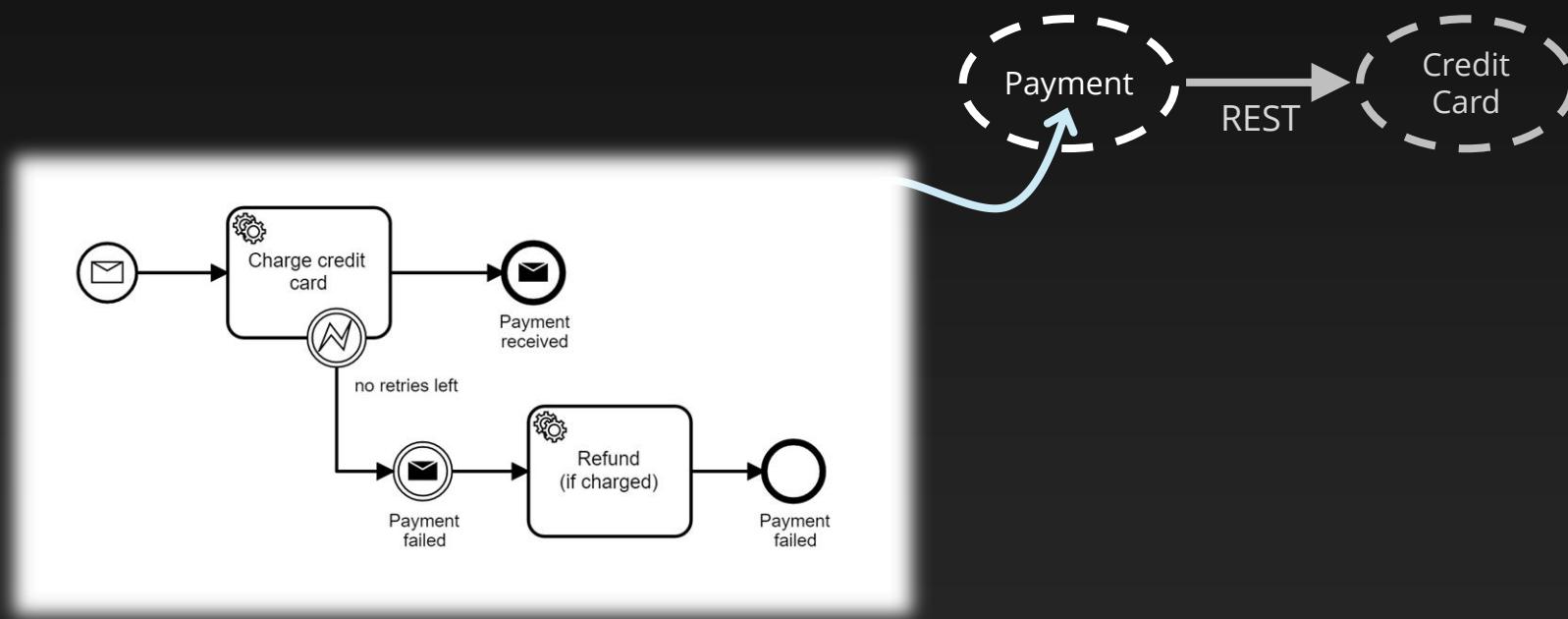
Independant of
communication style!



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. But there are other approaches to guarantee consistency in distributed systems.

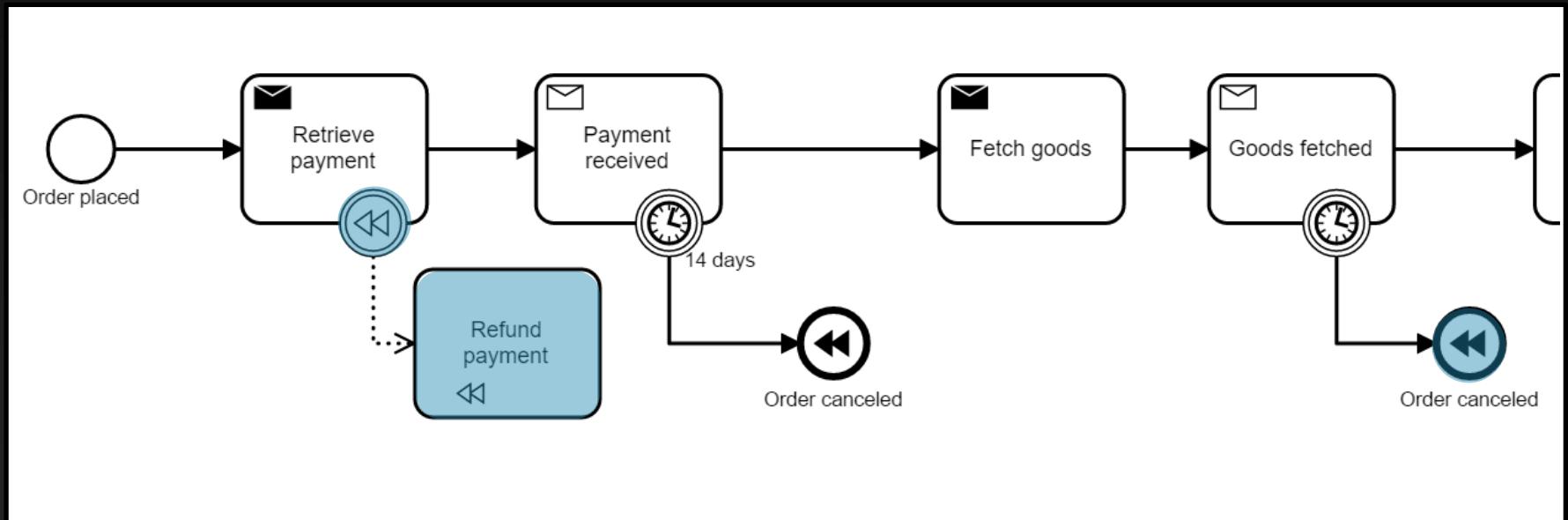
Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

This paper explores some of the practical

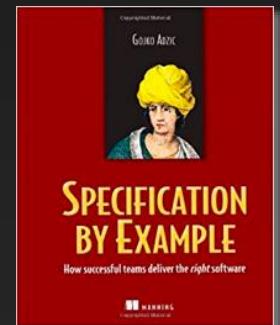
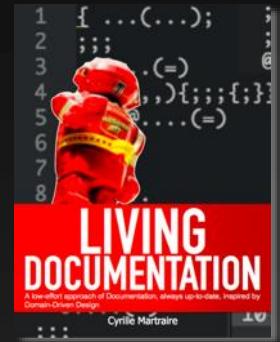
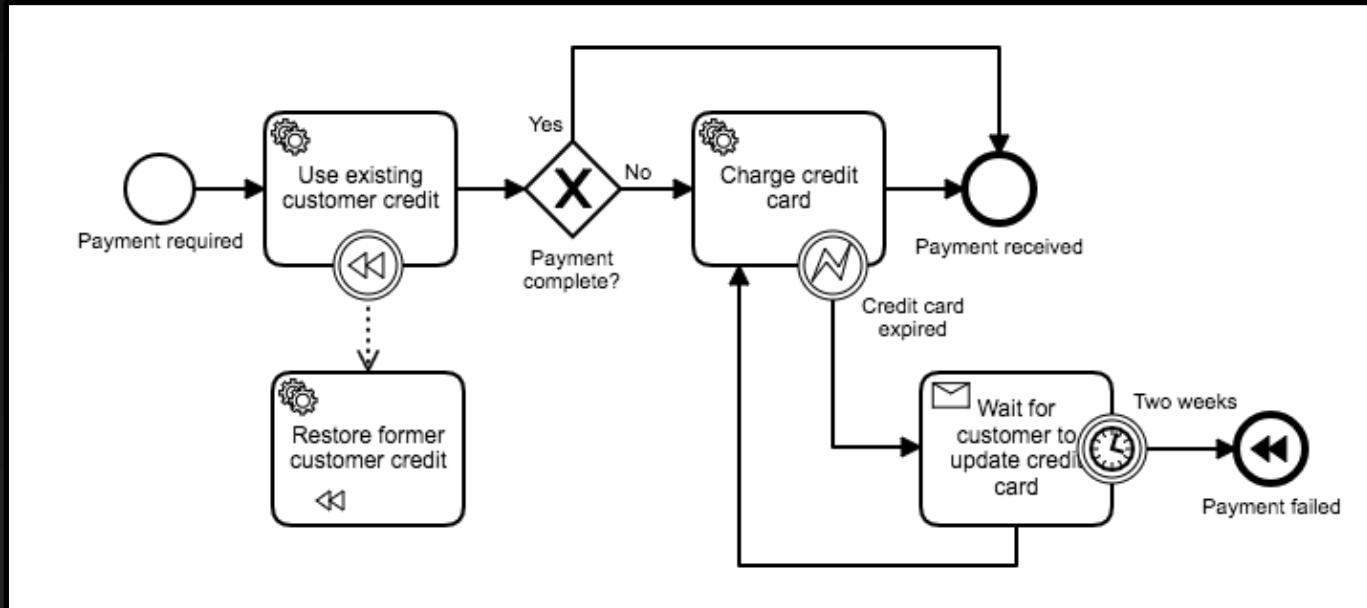


Distributed transaction with compensations*

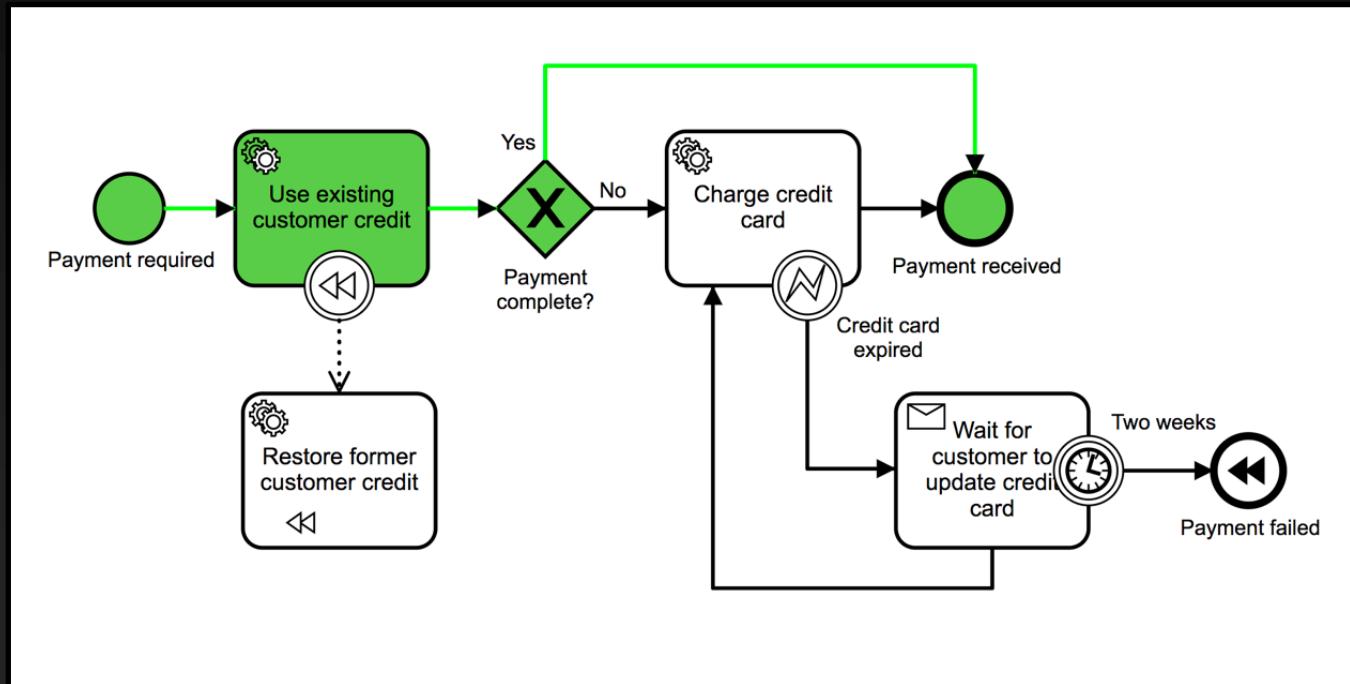
*aka Saga-Pattern



Living documentation for long-running behaviour

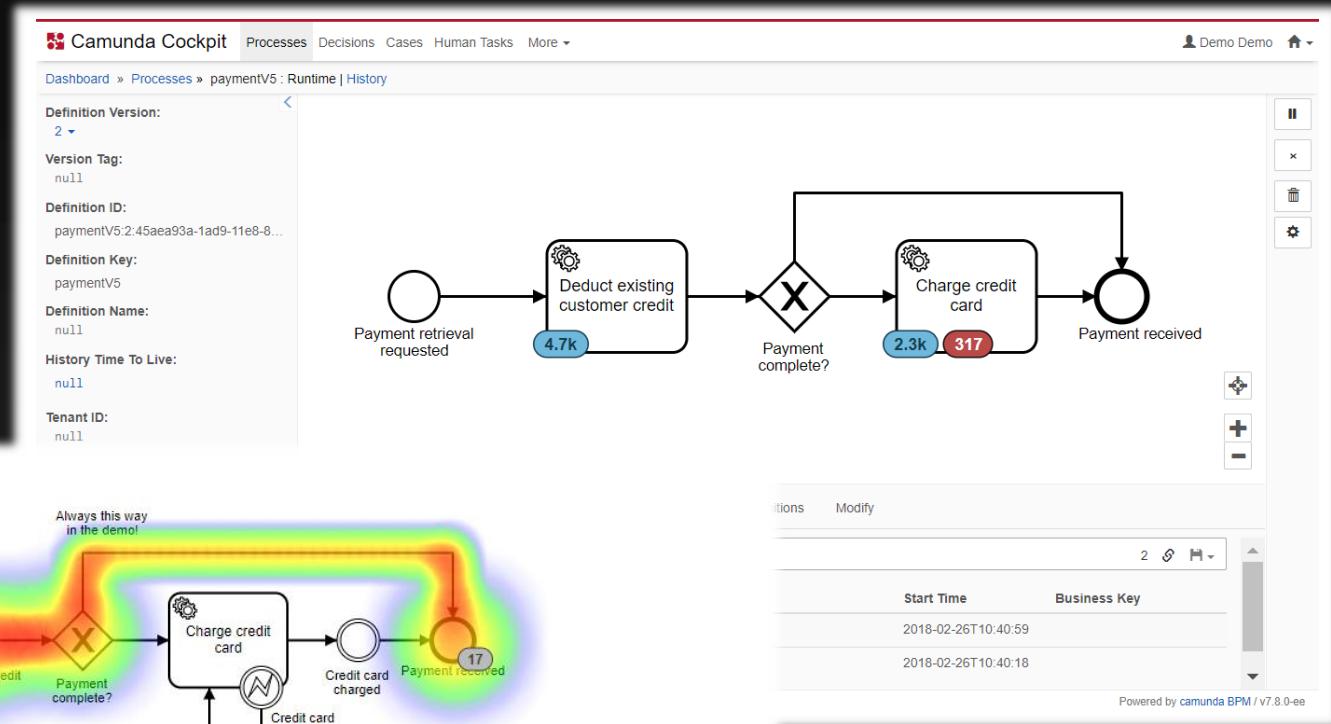
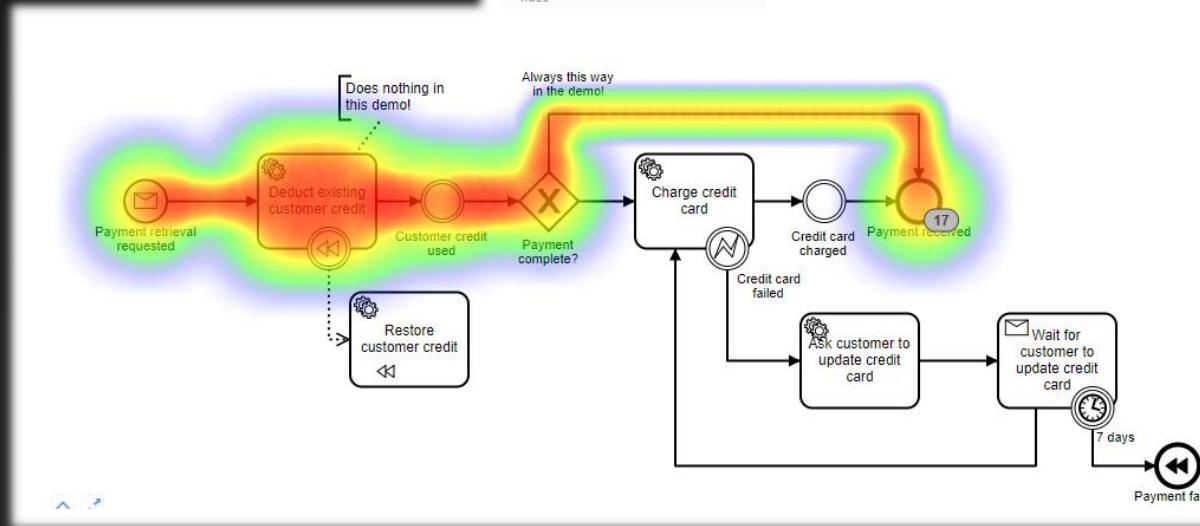


Visual HTML reports for test cases



Proper operations

Visibility + Context



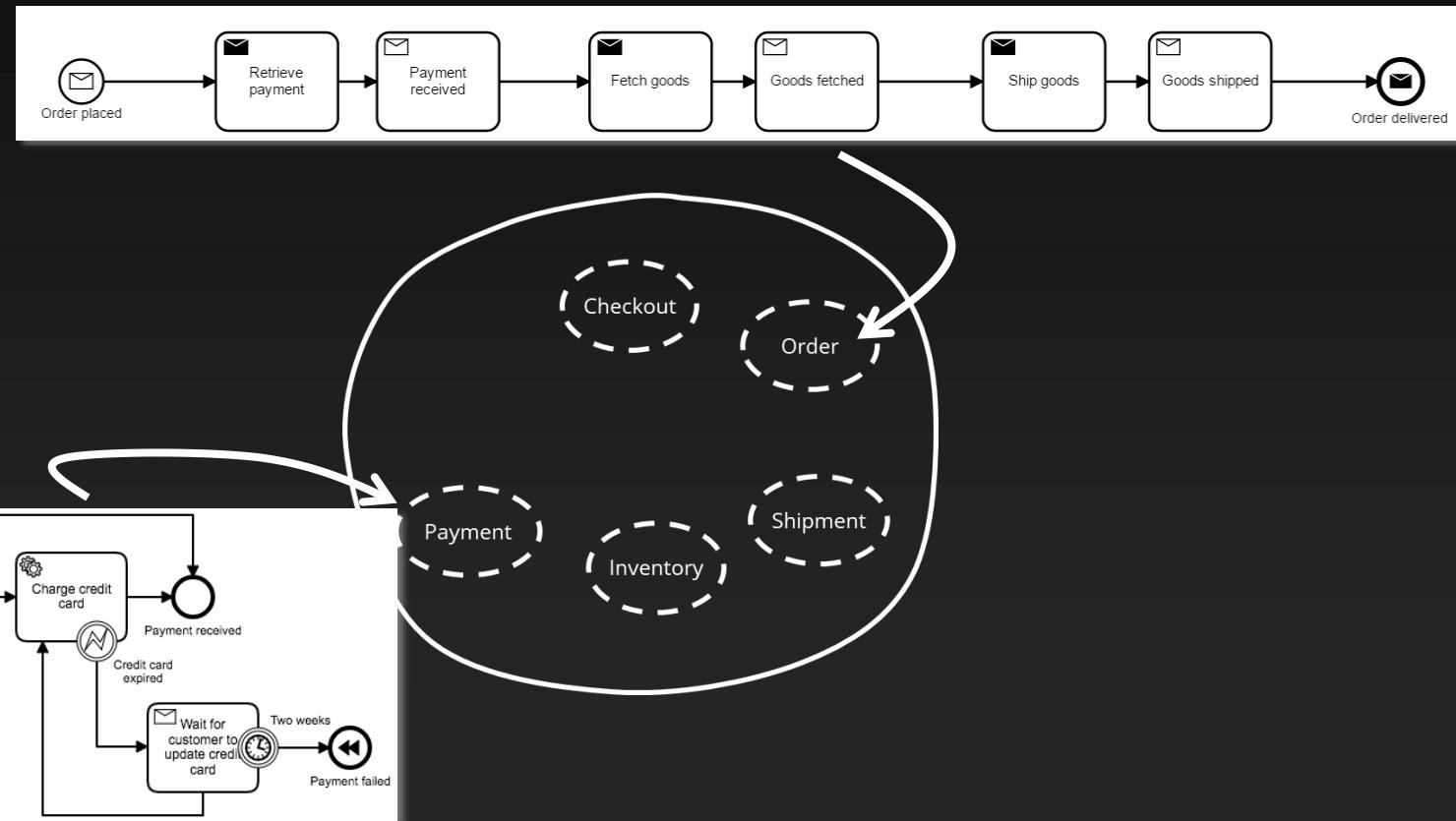


”

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

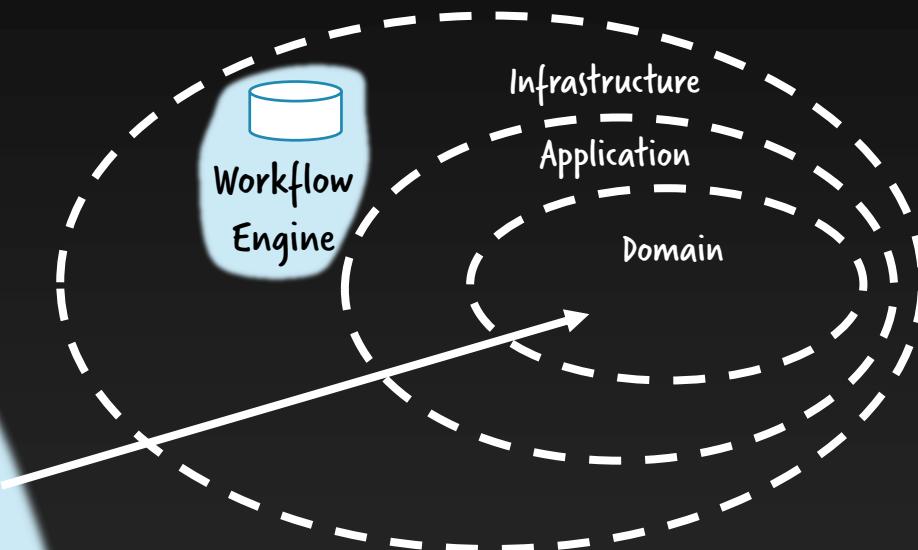
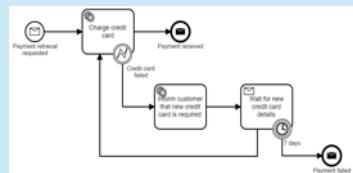
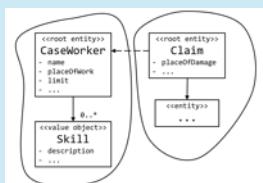
Workflows live inside service boundaries



Explicit flows help separate domain and infrastructure

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

+ the flow



Manifold architecture options

The screenshot shows a blog post on a white background. At the top left is a black square icon with a white letter 'M'. To its right is a vertical line, followed by the author's name 'berndruecker' and a small Twitter icon. On the far right are icons for 'Edit', a magnifying glass, a bell, and two user profiles.

Bernd Rücker
Dec 19, 2017 • 15 min read

Architecture options to run a workflow engine

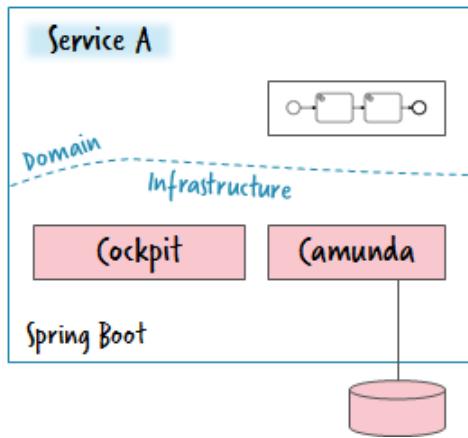
This week a customer called and asked (translated into my own words and shortened):

"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 context of microservices, modern SOA initiatives and API design. Modern workflow engines are increasingly distributed at possible...

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

Manifold architecture options



M | berndruecker

Bernd Rücker Dec 19, 2017 • 15 min read

Edit

Architecture options to run a workflow engine

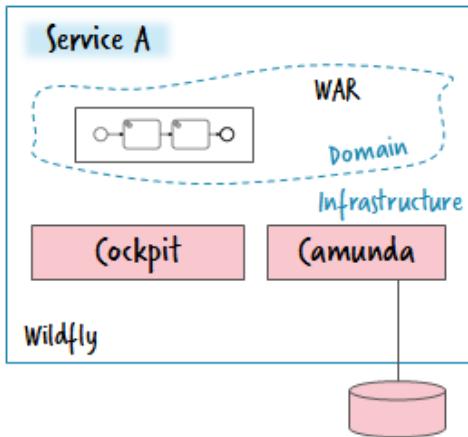
This week a customer called and asked (translated into my own words and shortened):

"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 context of microservices, modern SOA initiatives, etc. Many of these questions are interesting and I will try to answer them in this post.

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

Manifold architecture options



M | berndruecker

Bernd Rücker Dec 19, 2017 • 15 min read

Edit

Architecture options to run a workflow engine

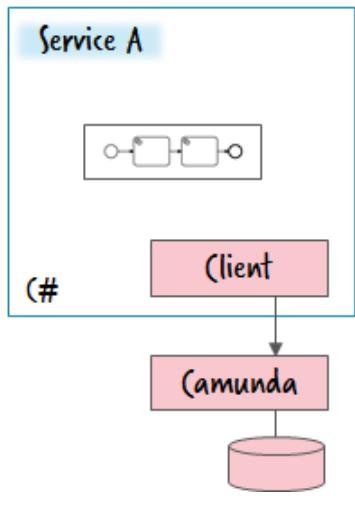
This week a customer called and asked (translated into my own words and shortened):

"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 context of microservices, modern SOA initiatives, and serverless design.

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

Manifold architecture options



M | berndruecker

Bernd Rücker Dec 19, 2017 · 15 min read

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 context of microservices, modern SOA initiatives, DevOps and cloud design. Many of these questions are interesting and I will try to answer them in this series of posts.

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

Manifold architecture options



The screenshot shows a blog post on a website. At the top left is a large 'M' logo, followed by a vertical line and the author's name 'berndruecker'. To the right of the name is a Twitter icon. Below the author information is a smaller profile picture of the author, 'Bernd Rücker', and the date 'Dec 19, 2017 · 15 min read'. On the far right of the header are 'Edit', 'Search', 'Bell', and user profile icons. The main title of the post is 'Architecture options to run a workflow engine'. Below the title is a subtitle: 'This week a customer called and asked (translated into my own words and shortened):'. A quote follows: '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?' At the bottom of the post, there is a note: 'These are valid questions which recently we get asked a lot, especially in context of microservices, modern SOA initiatives, ...'. The URL of the post is visible at the bottom of the image: <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

FULFILLMENT PROCESS



```
public void execute(final FulfillmentOrder order, final CamundaContext 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%

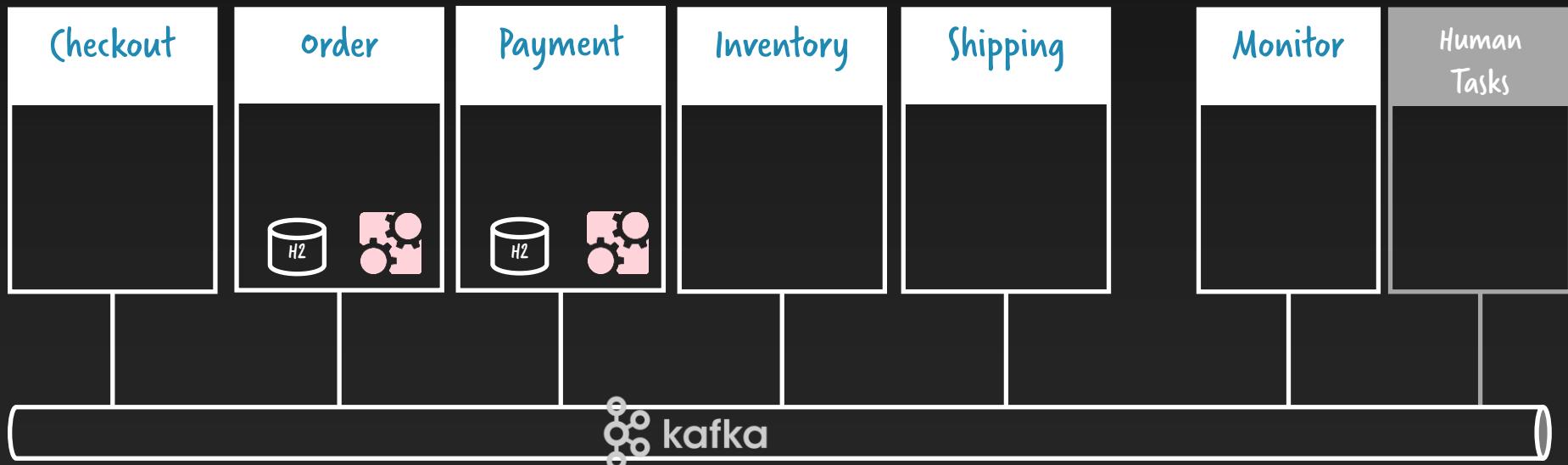
Bernd Ruecker
@berndruecker

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

A photograph showing a group of people sleeping in what appears to be a conference room or lecture hall. In the foreground, a man with a beard and grey hair is smiling while sleeping. Behind him, many other people are also asleep, leaning their heads on their hands or resting on their desks. The room has large windows with blinds in the background.

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>

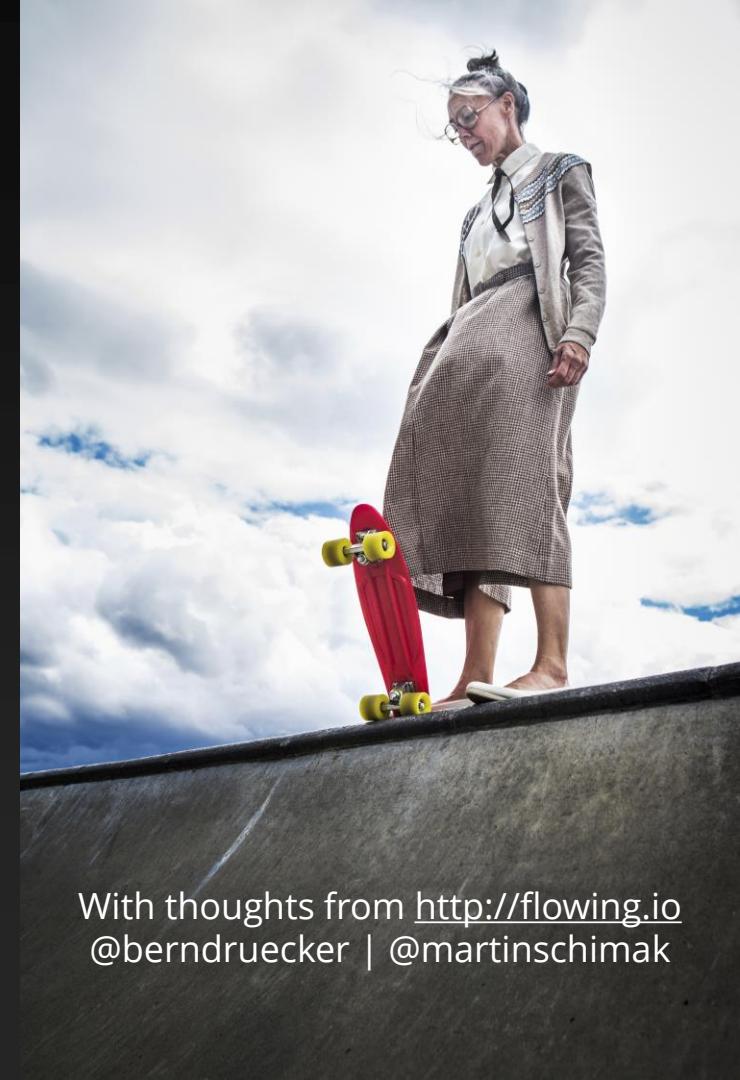
Code: <https://github.com/flowing>

InfoWorld
FROM IDG

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

InfoQ
ueue

[https://www.infoq.com/articles/
events-workflow-automation](https://www.infoq.com/articles/events-workflow-automation)



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