Complex event flows in distributed systems

@berndruecker

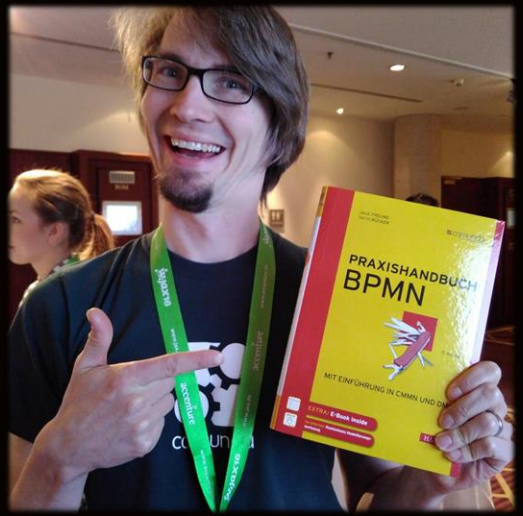
With thoughts from http://flowing.io
@berndruecker | @martinschimak

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



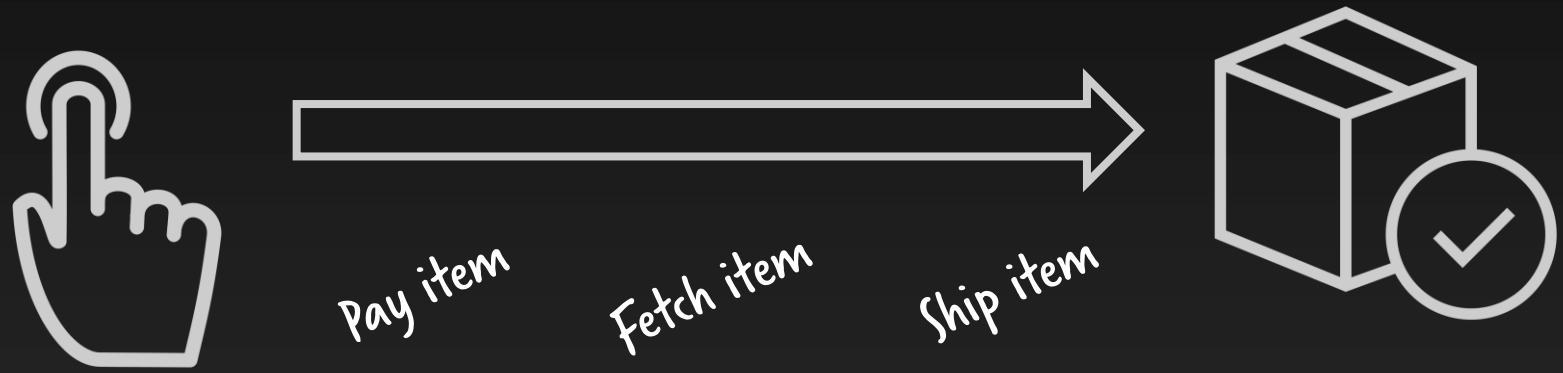
Berlin, Germany

bernd.ruecker@camunda.com
@berndruecker

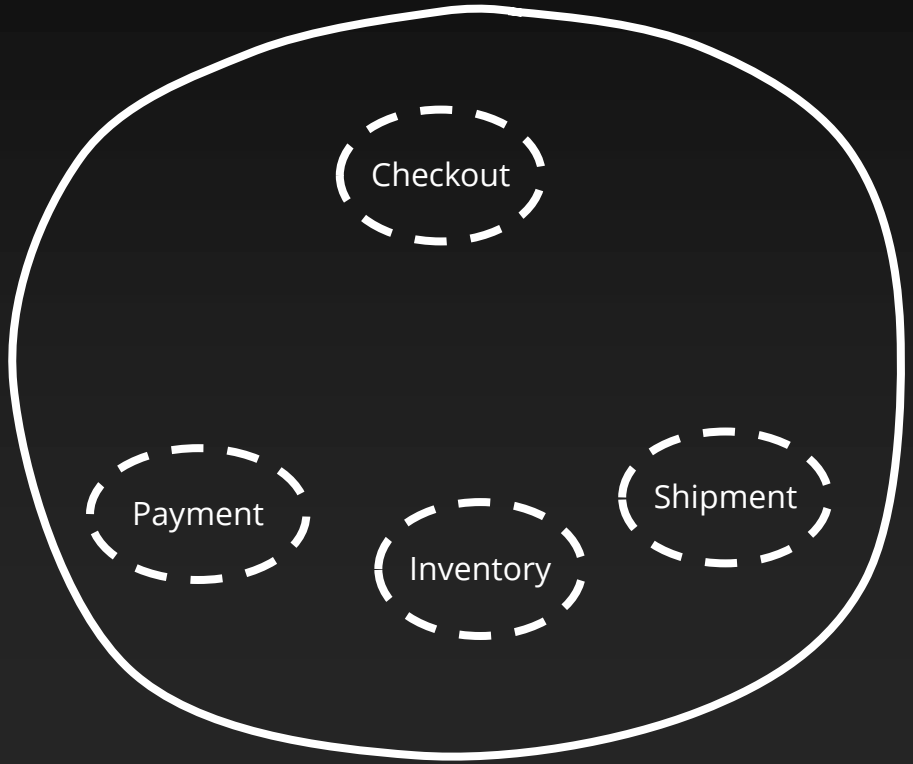

ORCHESTRATING A HIGHLY-SCALABLE FULFILLMENT PROCESS
JÖRN HORSTMANN
LUKAS NIEMEIER
2017-09-19
THE SHUTTLE
TECH INNOVATION LAB

Simplified example: dash button

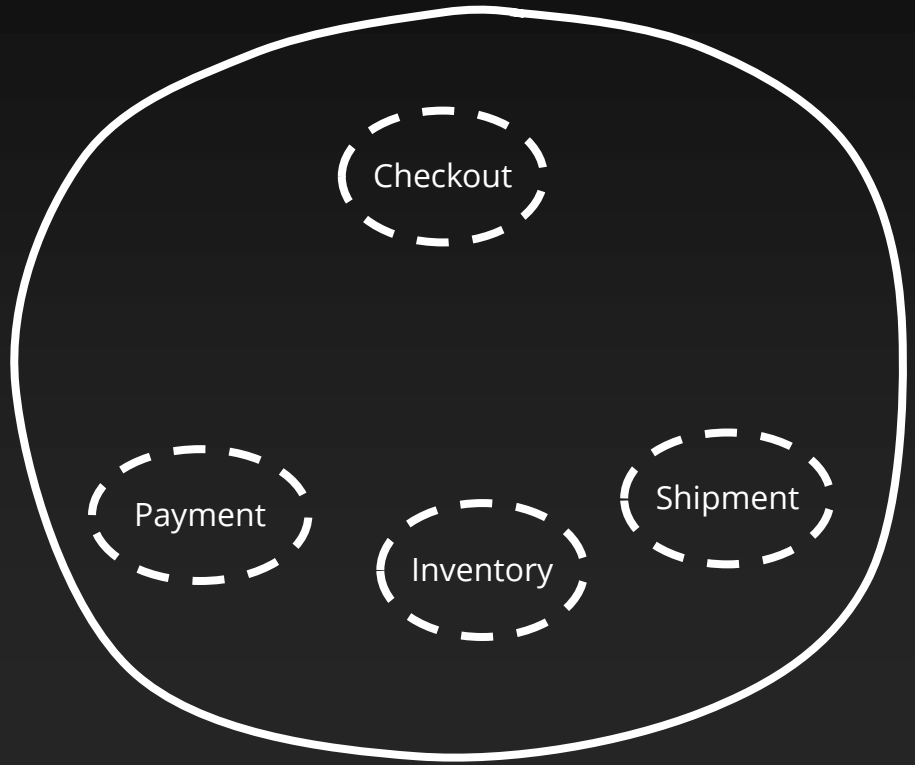
Three steps...



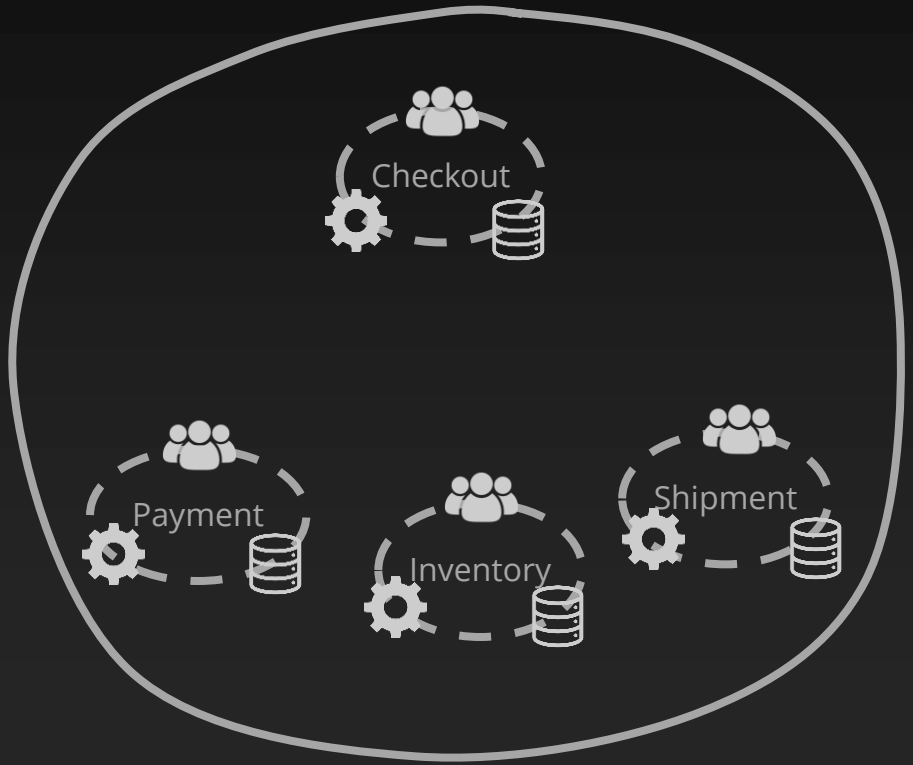
Who is involved? Some bounded contexts...



(Micro-)services



Autonomous (micro-)services



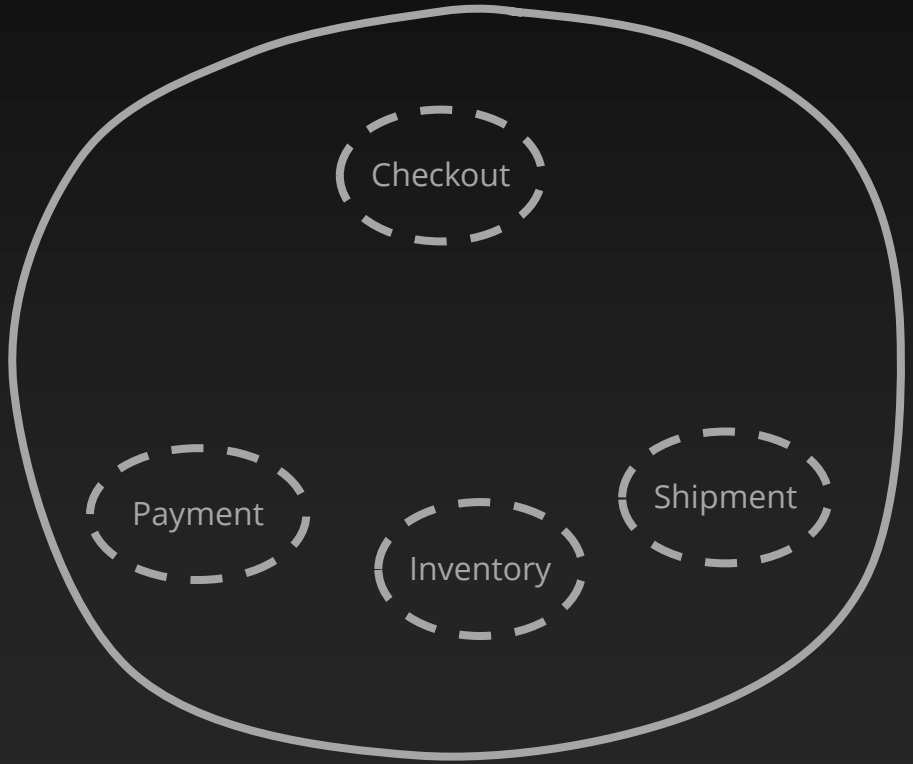
Dedicated Application Processes

Dedicated infrastructure

Dedicated Development Teams

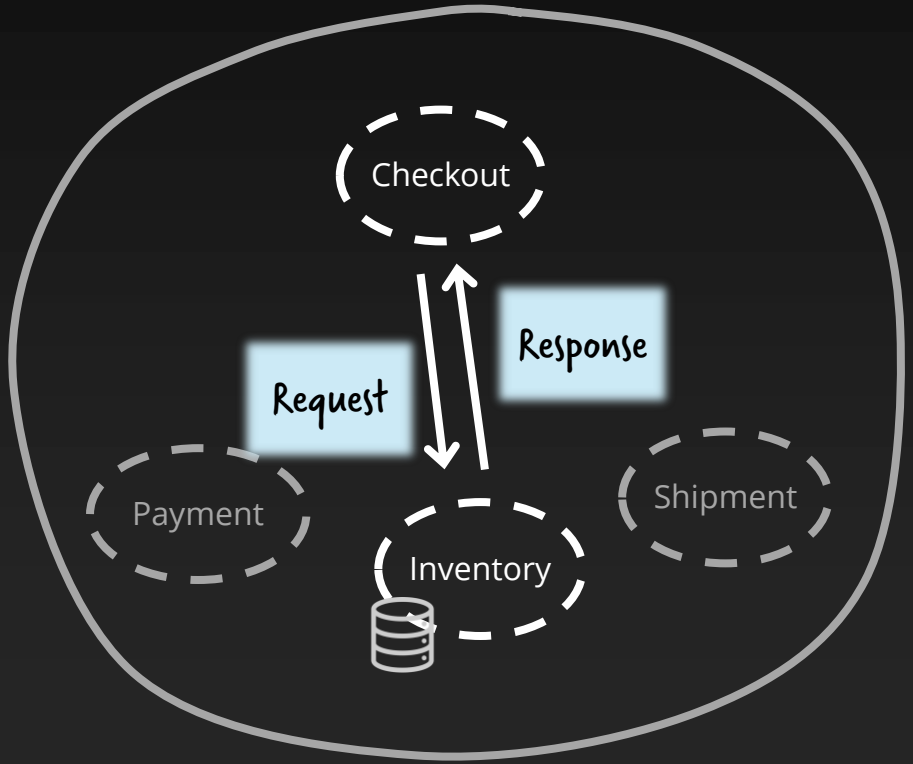
Events decrease coupling

Example



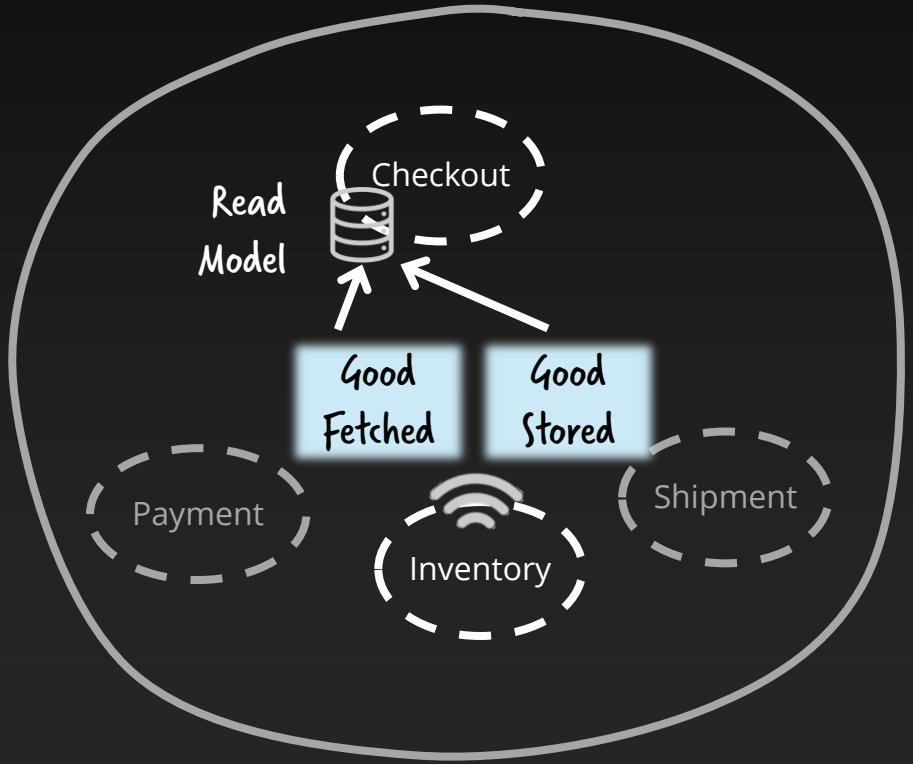
The button blinks if we can ship within 24 hours

Request/response: temporal coupling



The button blinks if we can ship within 24 hours

Temporal decoupling with events and read models



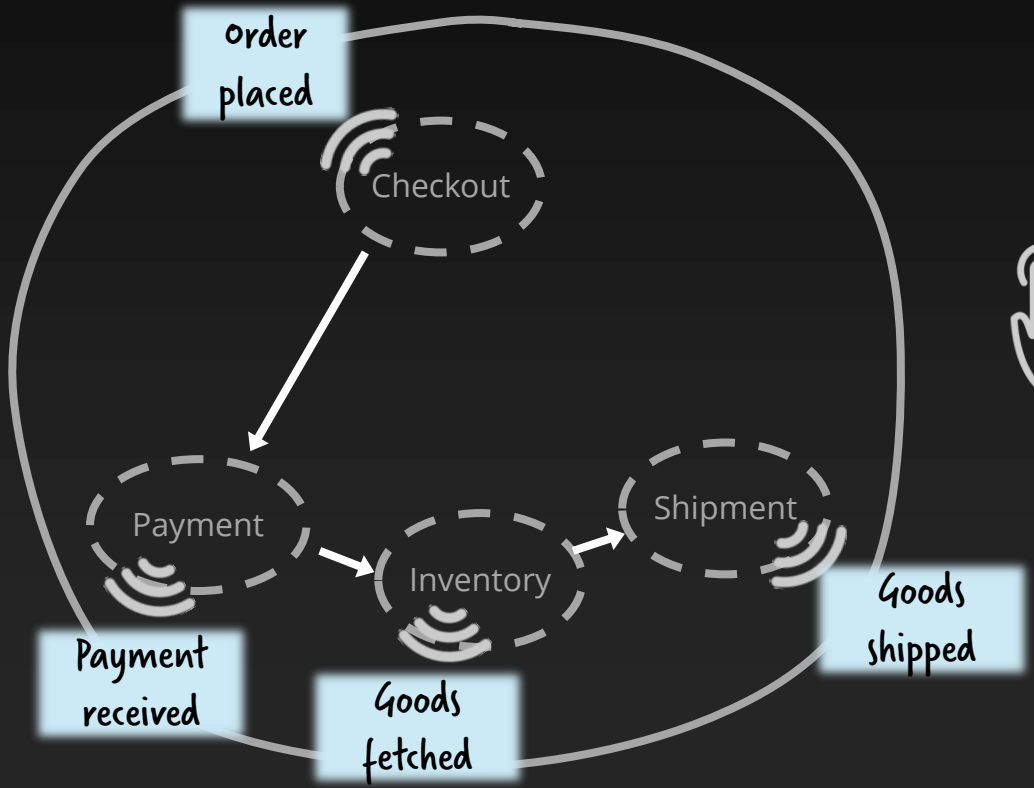
The button blinks if we can ship within 24 hours

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

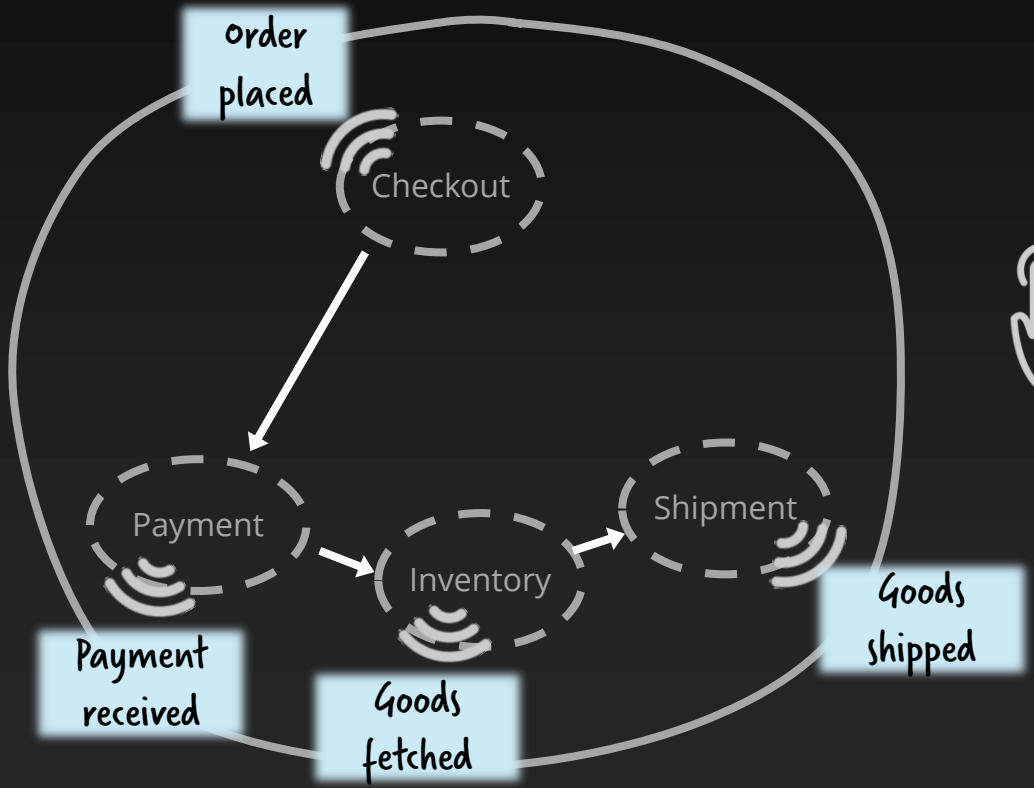
Events can decrease coupling*

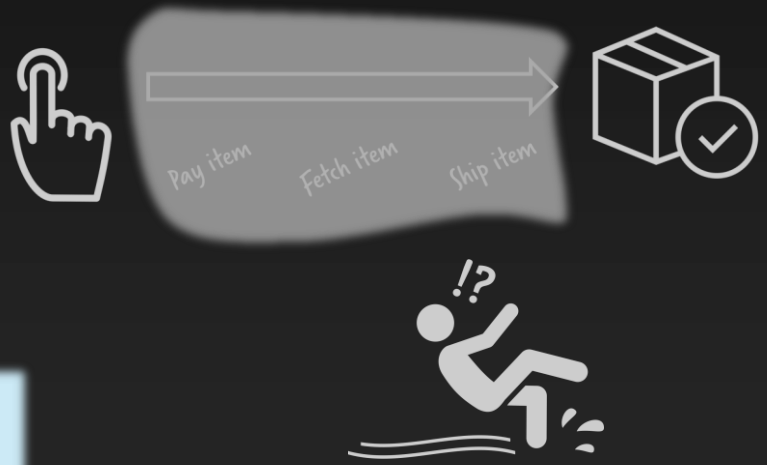
*e.g. decentral data-management, read models, extract cross-cutting aspects

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.

https://martinfowler.com/articles/201701-event-driven.html

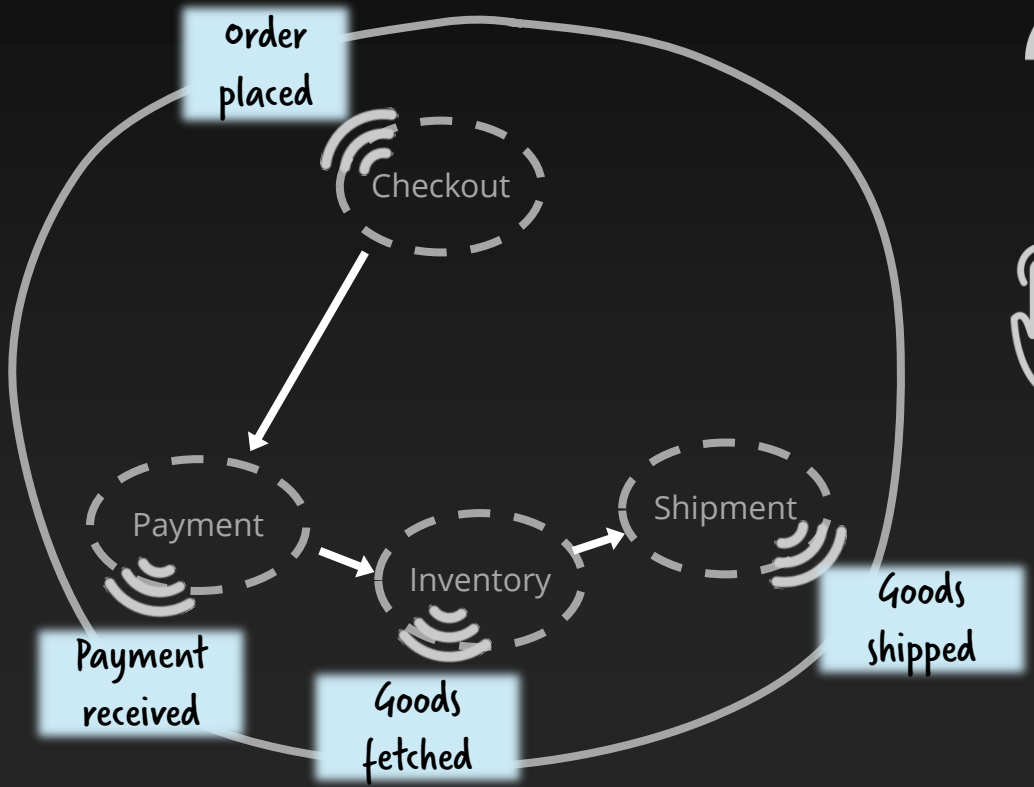
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html

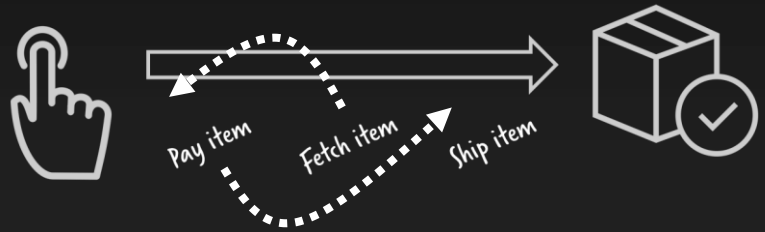
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus **set yourself up for trouble in future years**.

https://martinfowler.com/articles/201701-event-driven.html

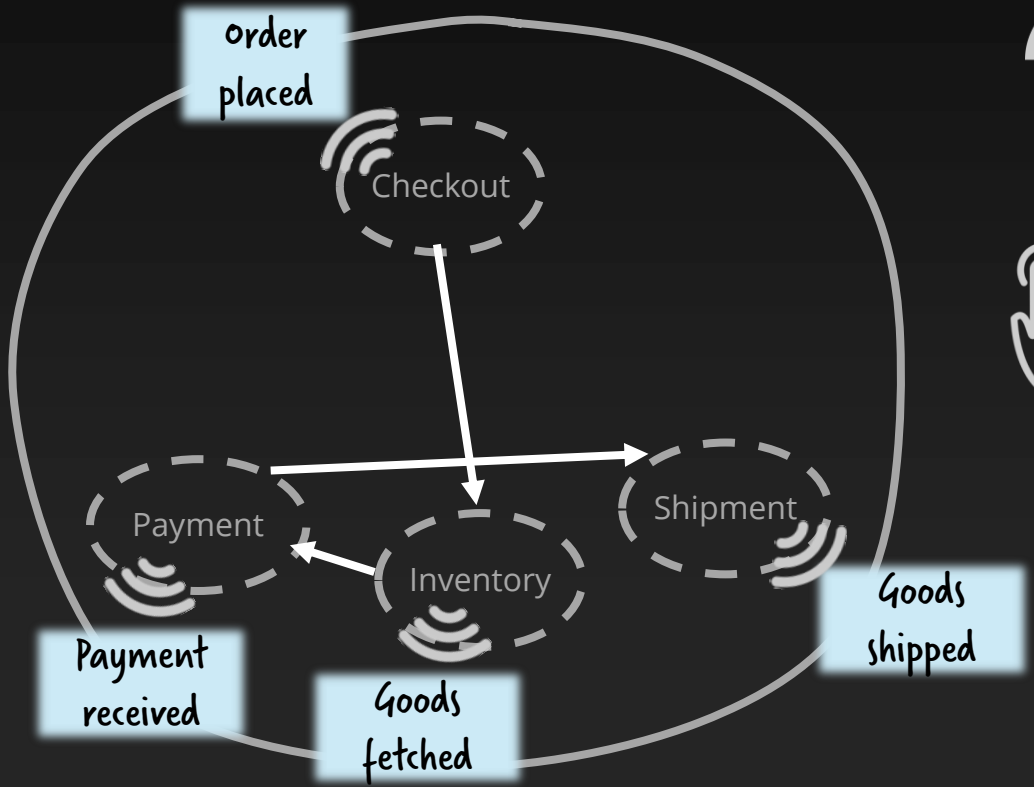
Peer-to-peer event chains



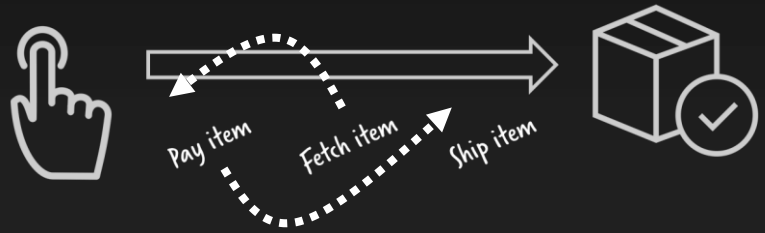
Fetch the goods **before** the payment



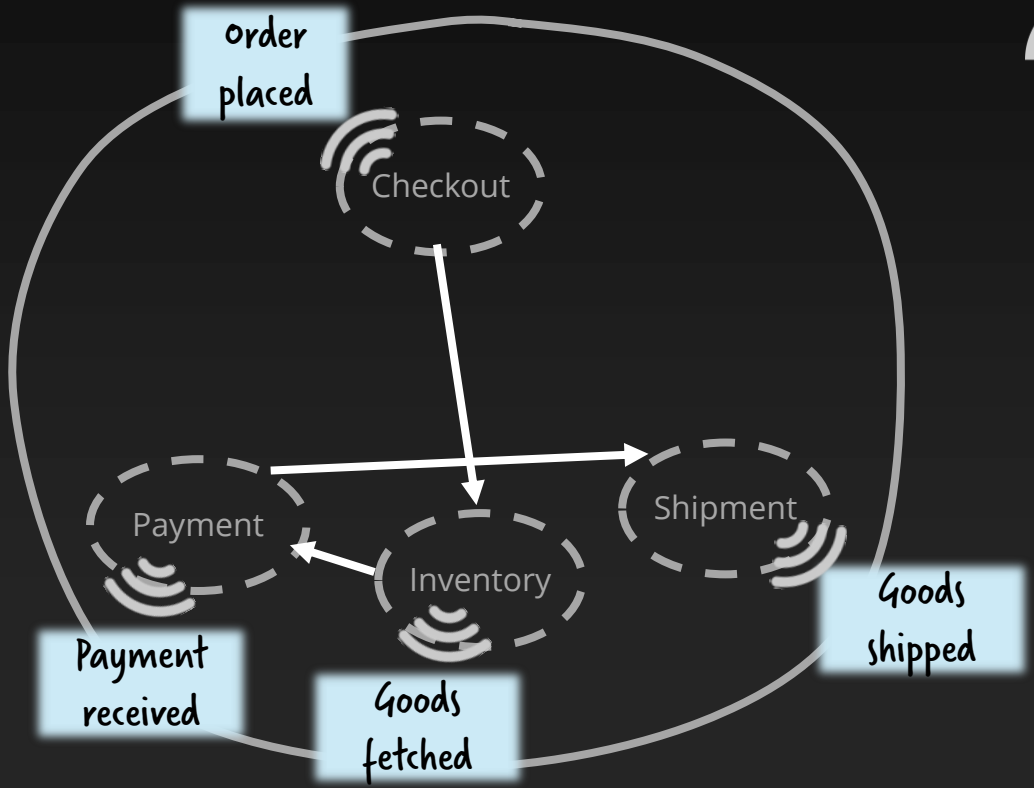
Peer-to-peer event chains



Fetch the goods **before** the payment



Peer-to-peer event chains

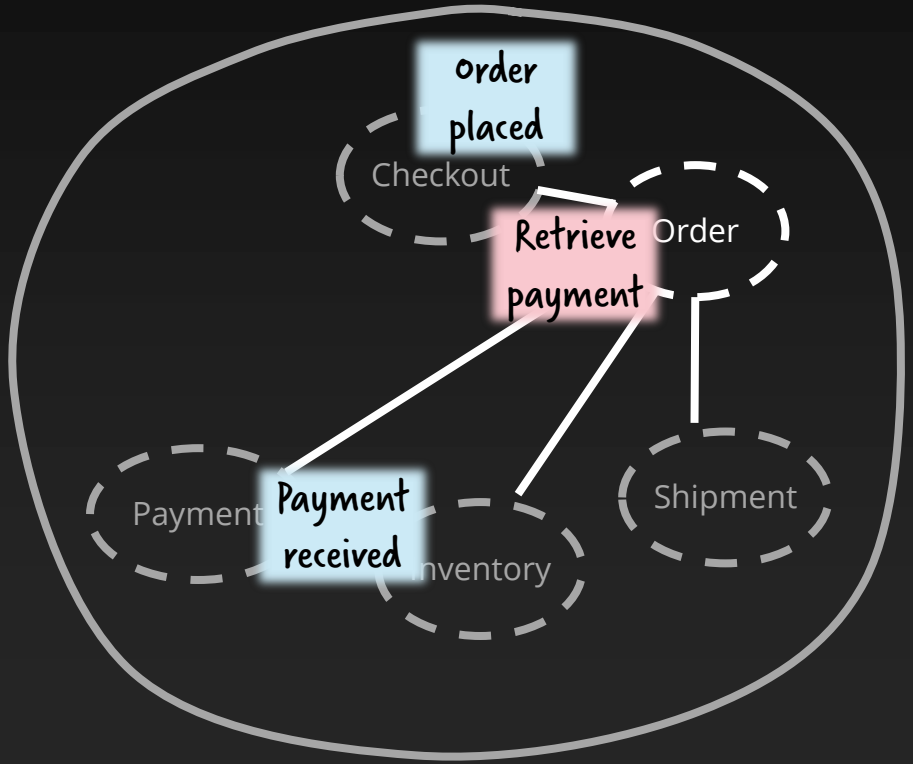


Fetch the goods **before** the payment

Customers can pay via invoice

...

Extract the end-to-end responsibility

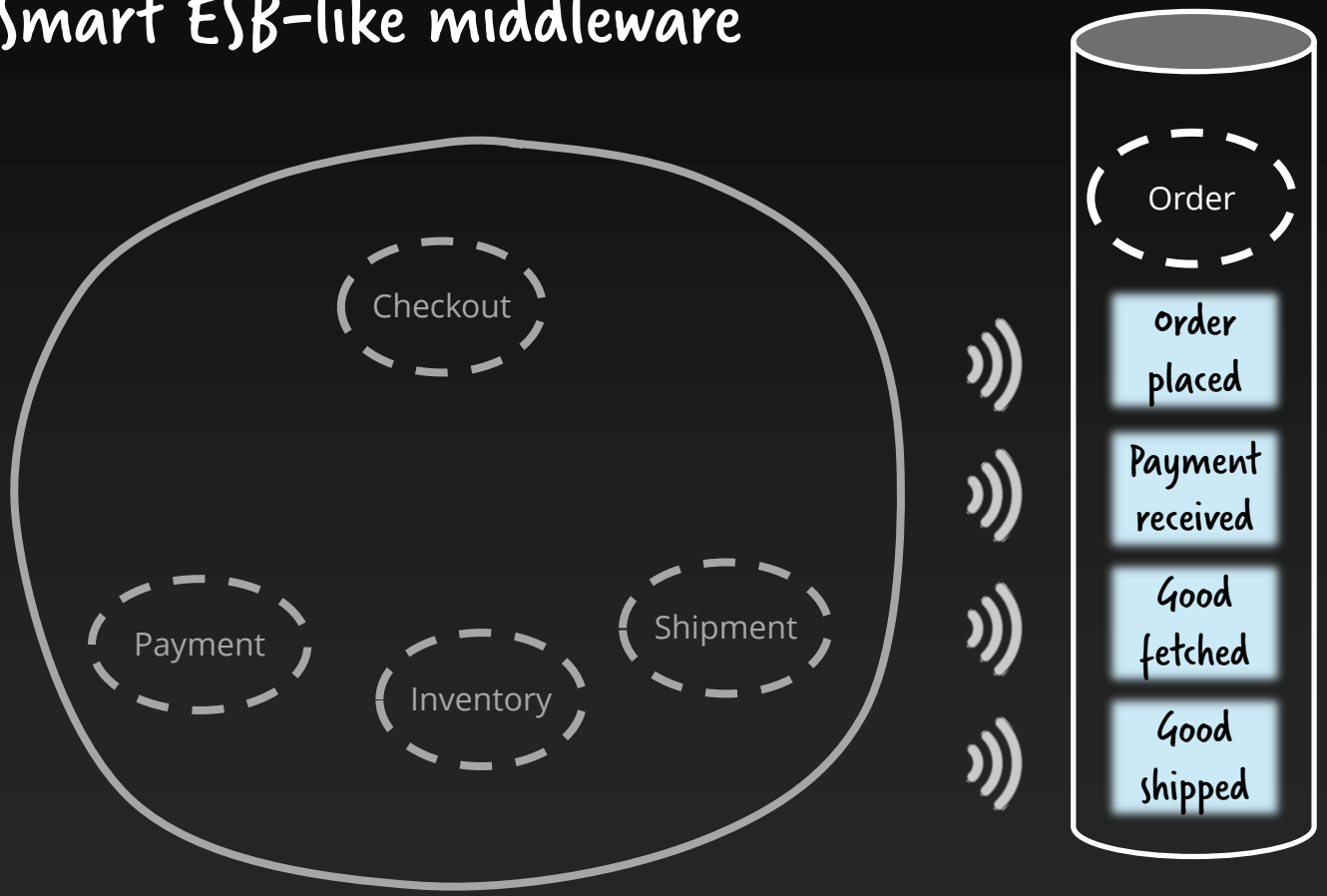


***Commands** have an intent about what needs to happen in the future

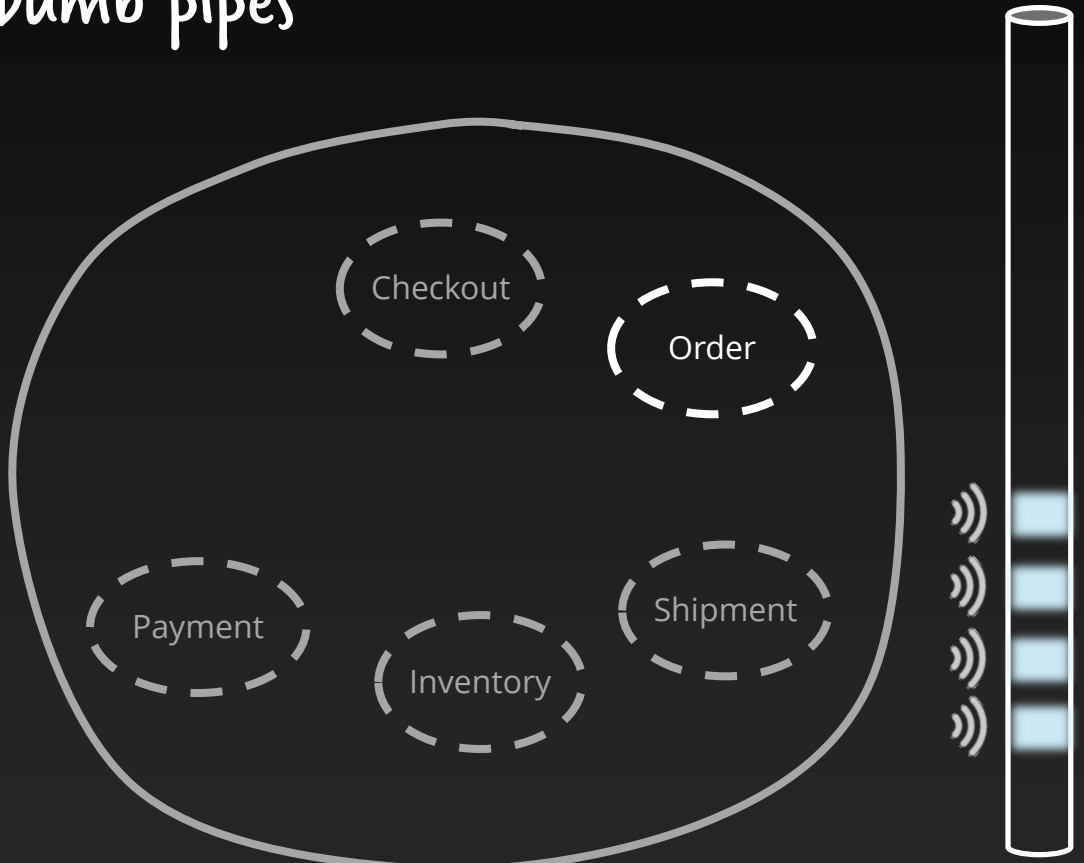
Commands help to avoid (complex) peer-to-peer event chains

orchestration needs to be avoided

Smart ESB-like middleware



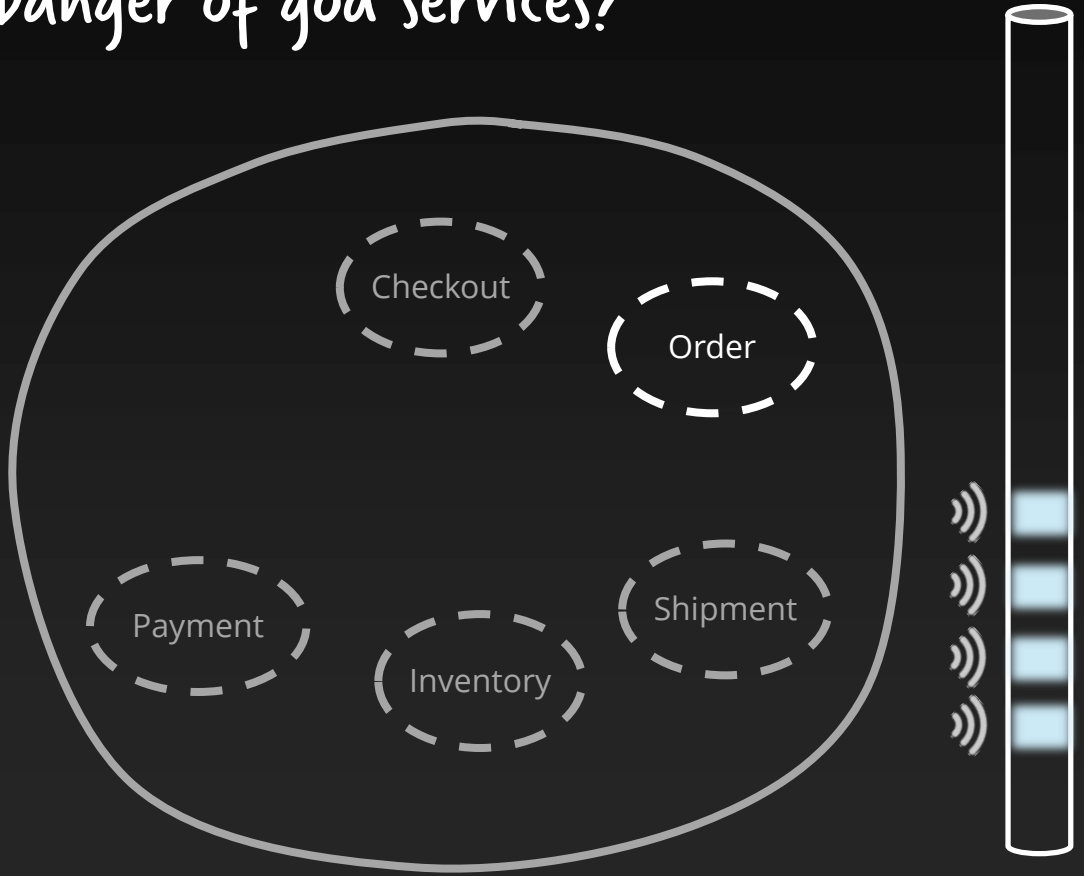
Dumb pipes



Martin Fowler

Smart endpoints
and **dumb pipes**

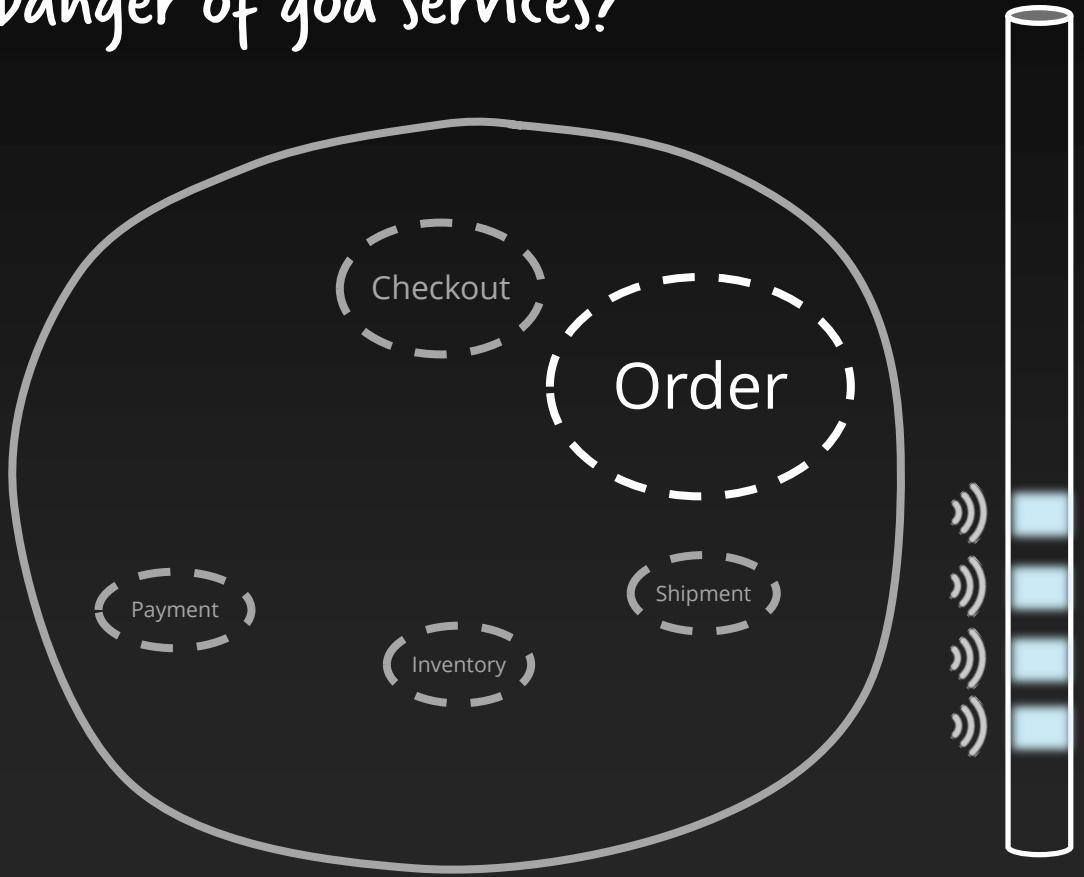
Danger of god services?



Sam Newmann

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

Danger of god services?



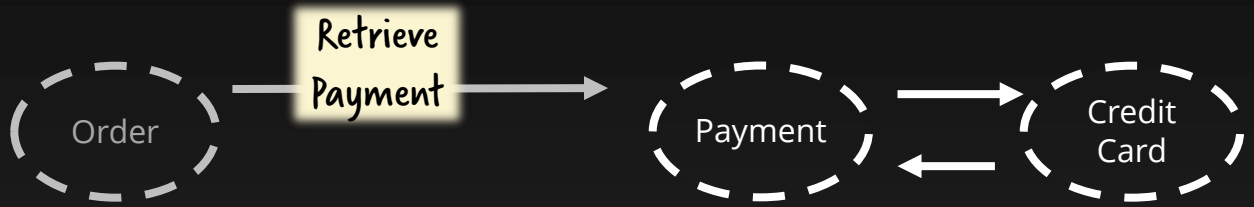
Sam Newmann

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

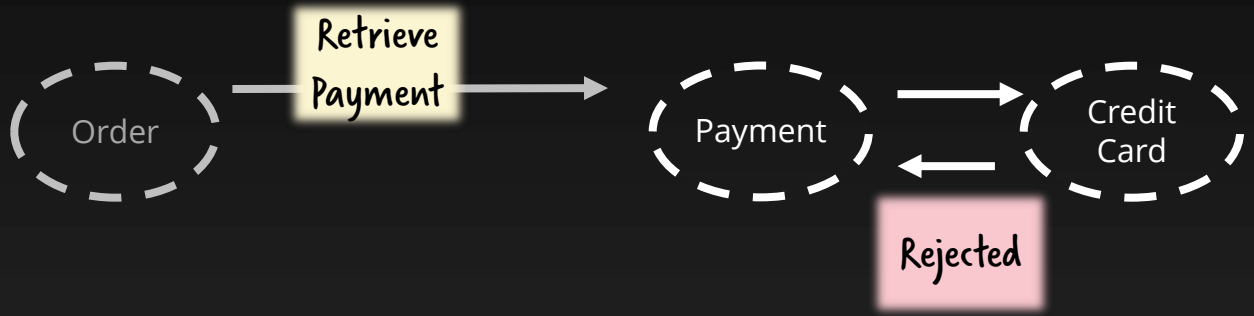
A god service is only created by bad API design!

Example

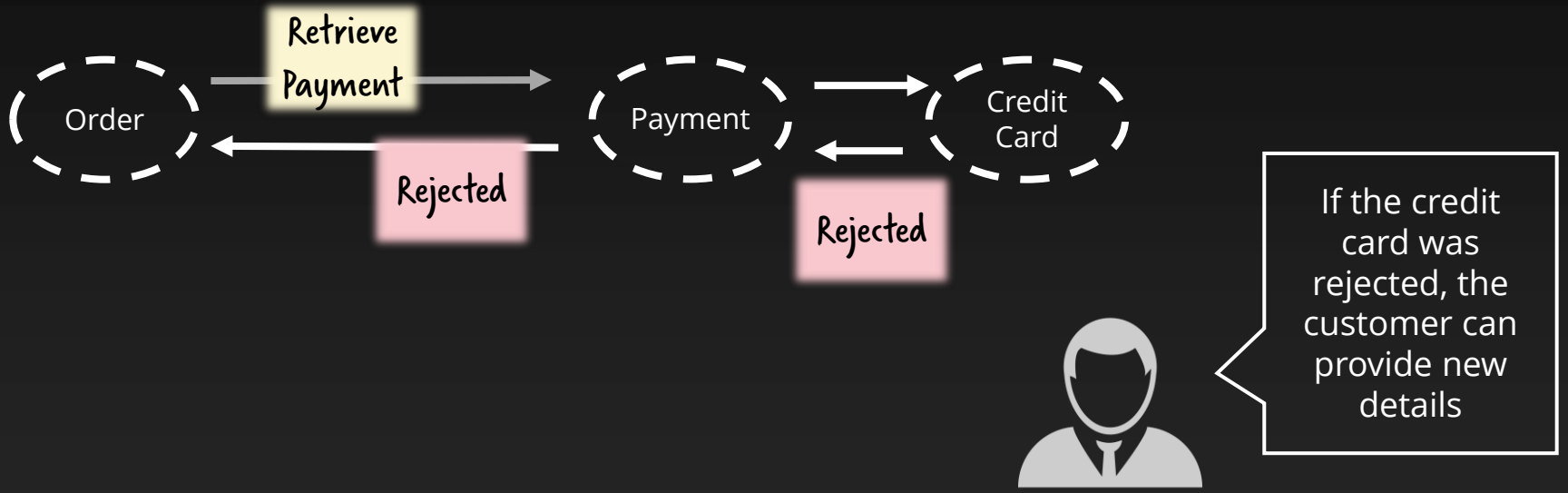
Example



Example

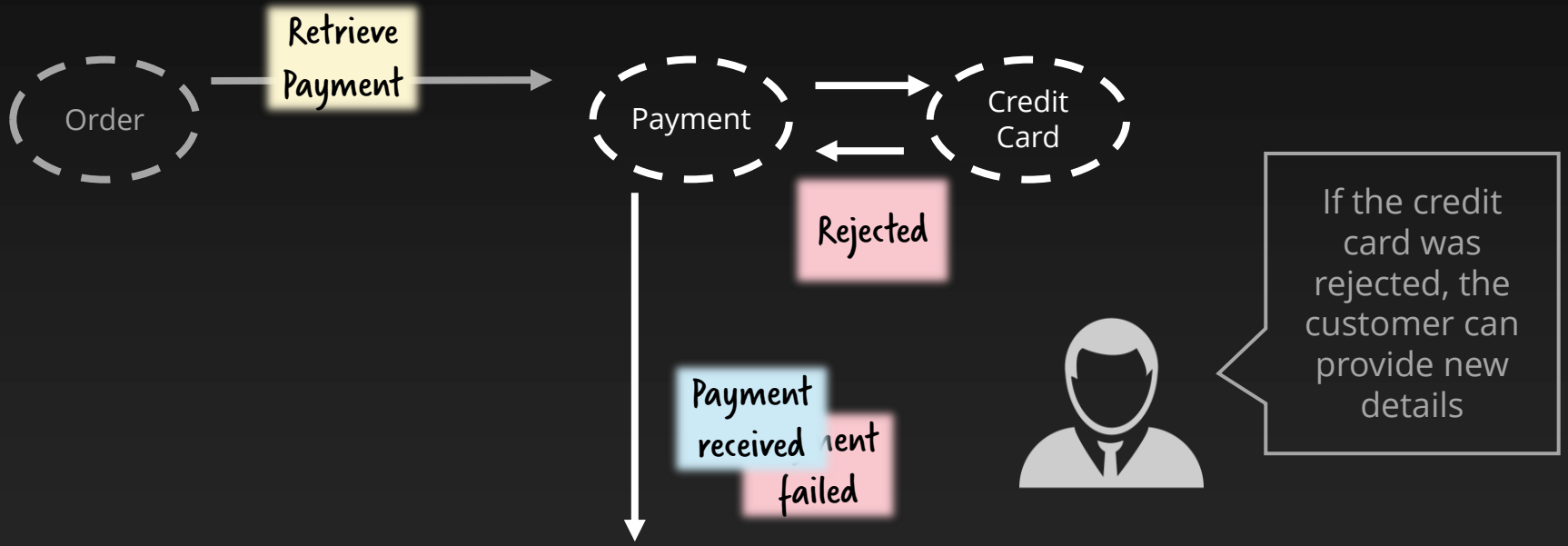


Example

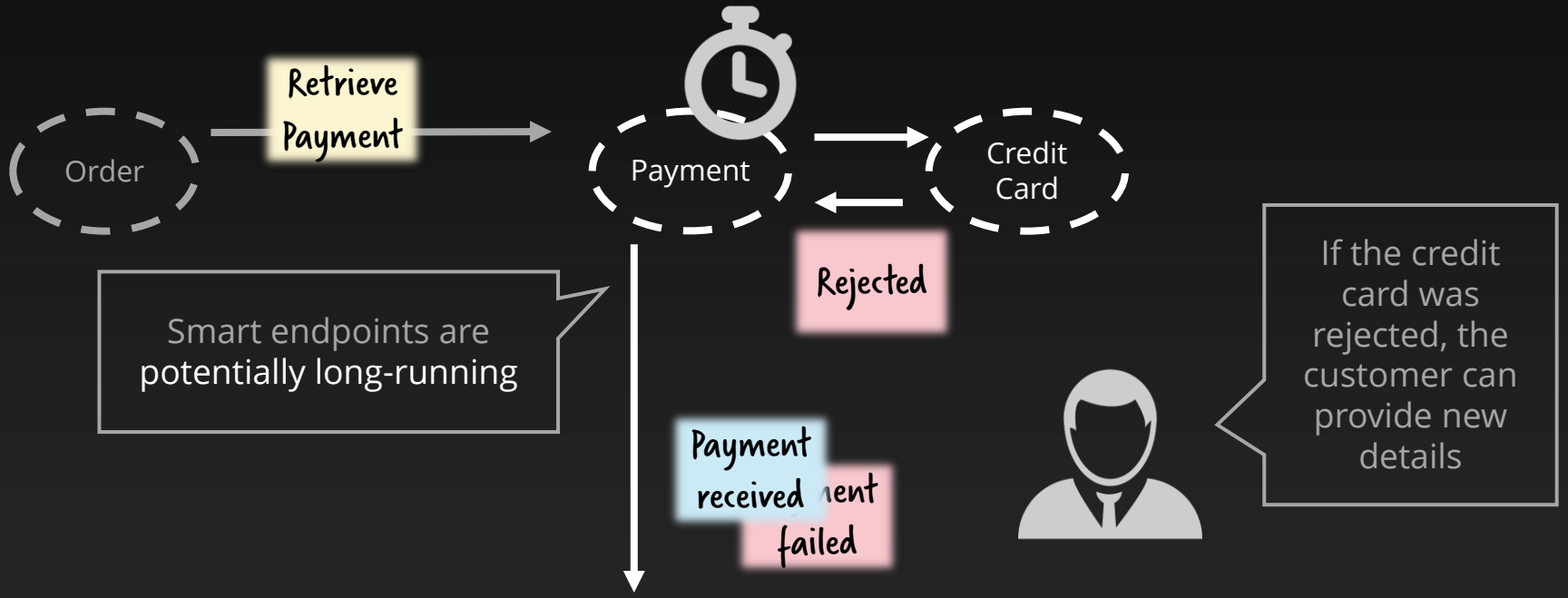


Client of **dumb endpoints** easily become a god services.

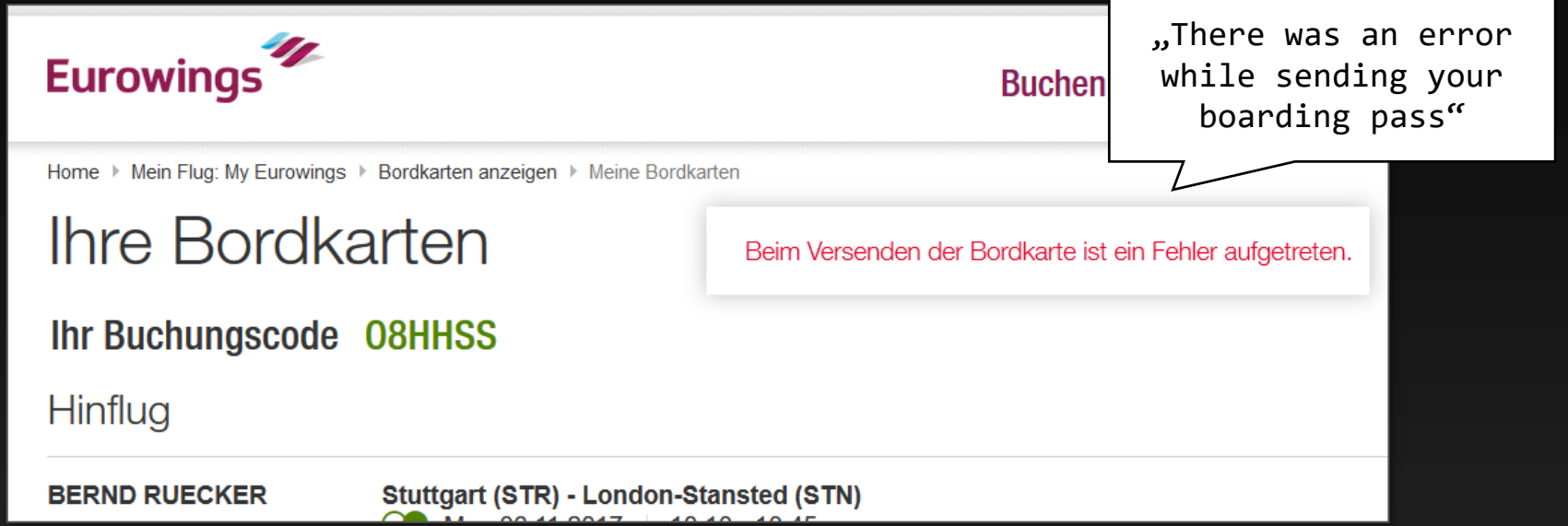
Who is responsible to deal with problems?



Who is responsible to deal with problems?

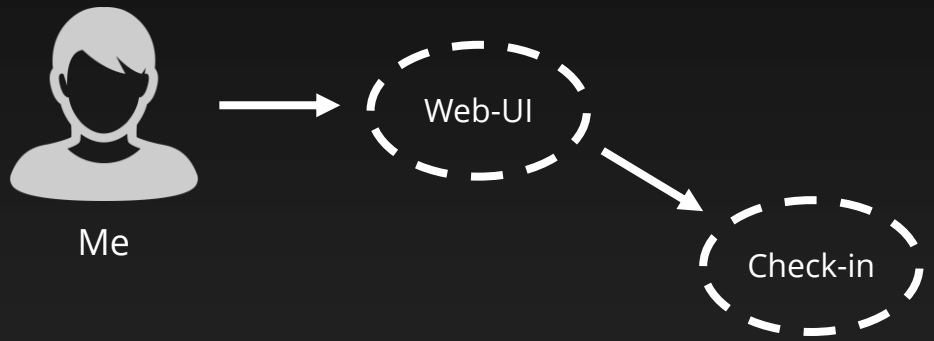


Clients of **smart endpoints** remains lean.

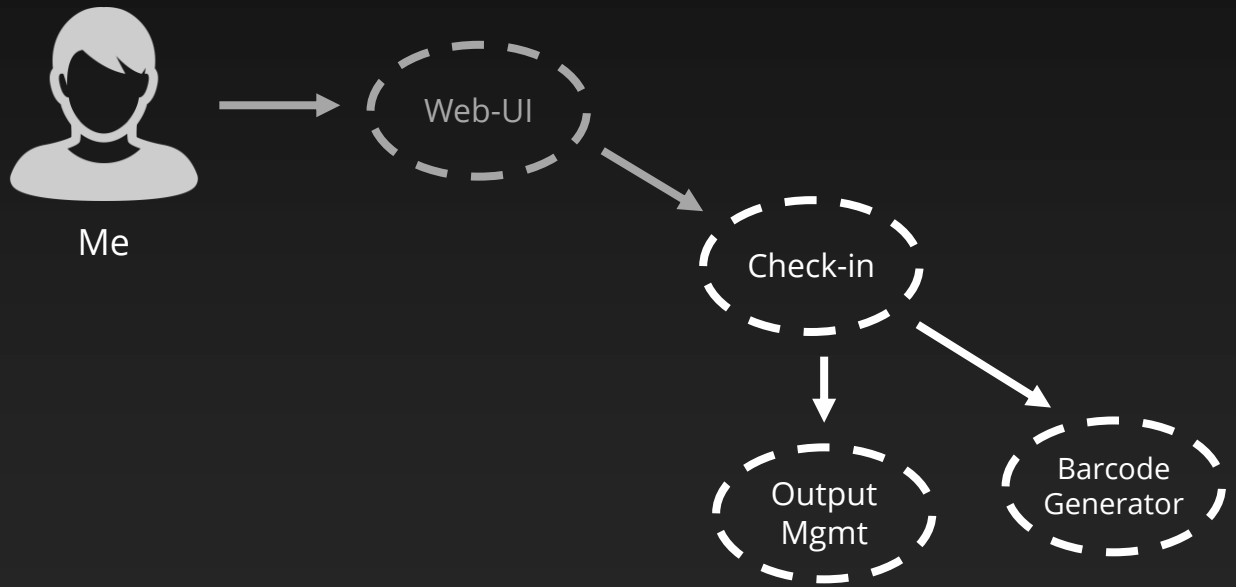


„There was an error while sending your boarding pass“

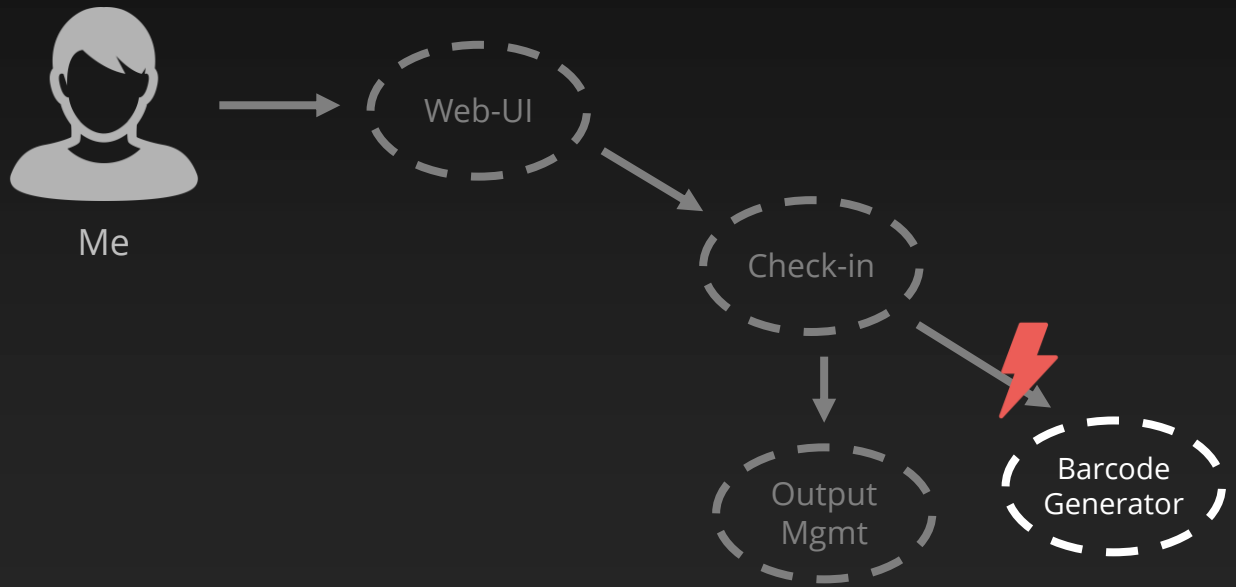
Current situation



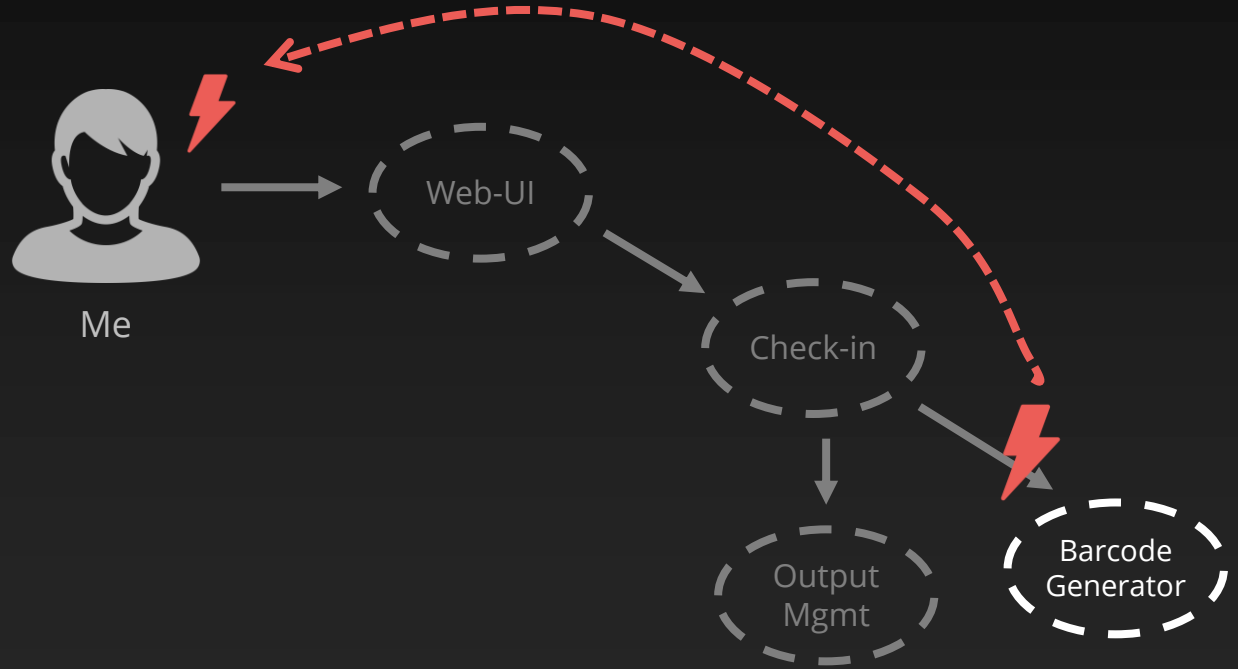
Current situation



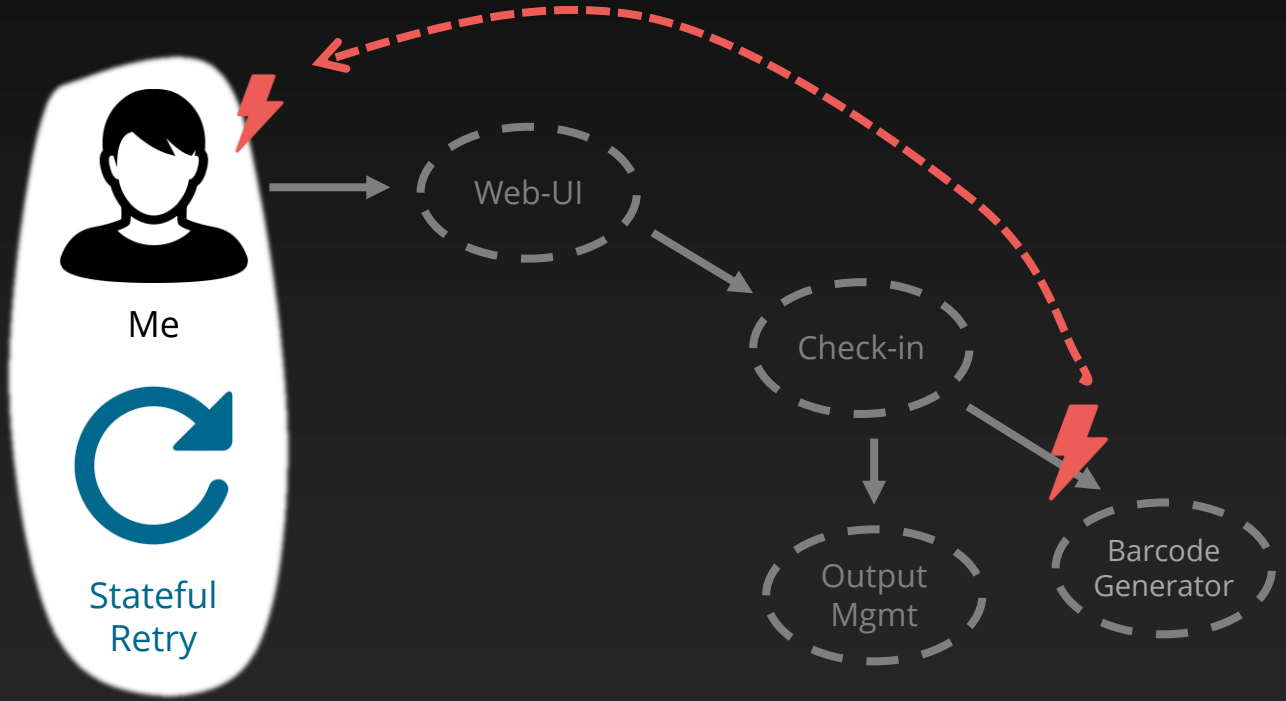
Current situation

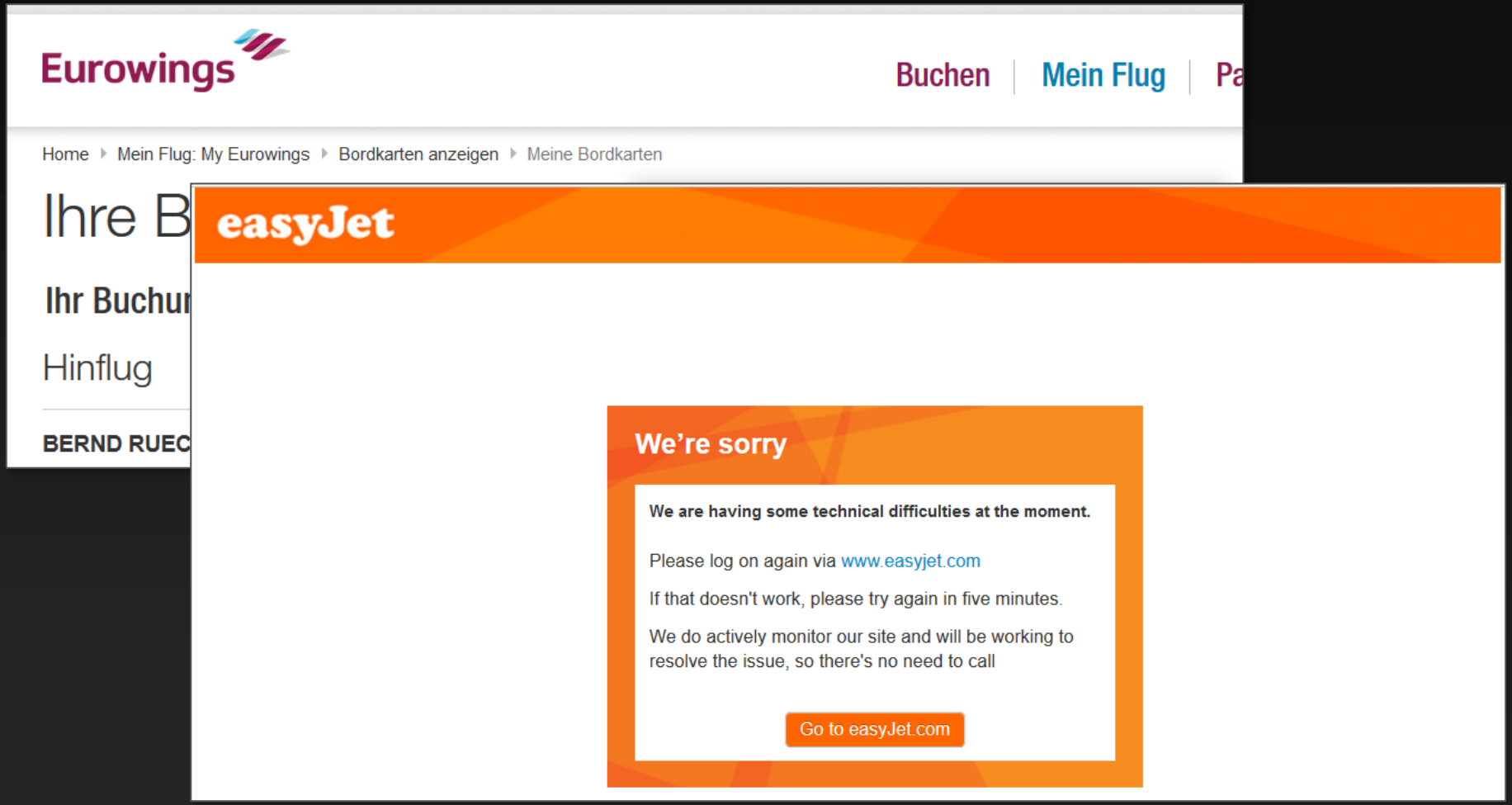


Current situation – the bad part



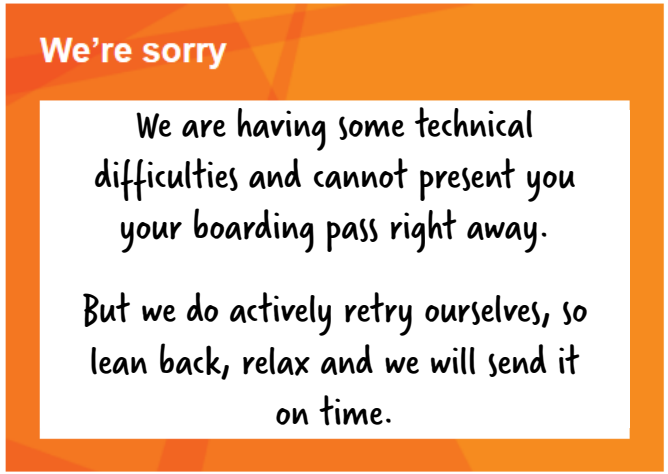
Current situation – the bad part



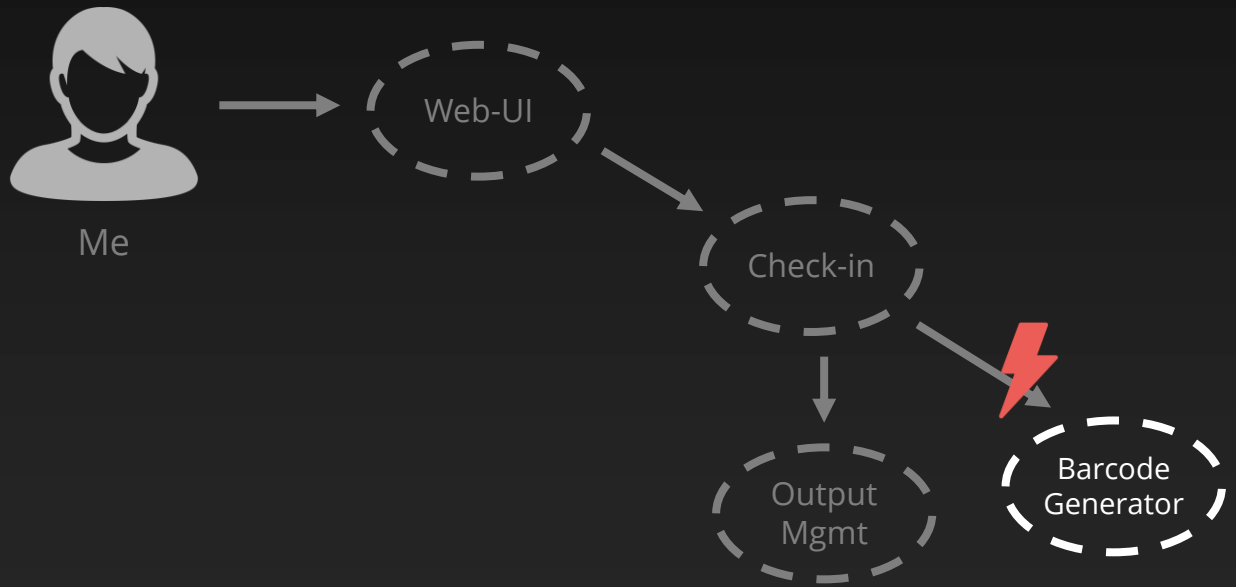

Eurowings
Buchen
Mein Flug
Home
Mein Flug: My Eurowings
Bordkarten anzeigen
Meine Bordkarten
Ihre B
Ihr Buchu
Hinflug
BERND RUEC
easyJet
We're sorry
We are having some technical difficulties at the moment.
Please log on again via www.easyjet.com
If that doesn't work, please try again in five minutes.
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

easyJet

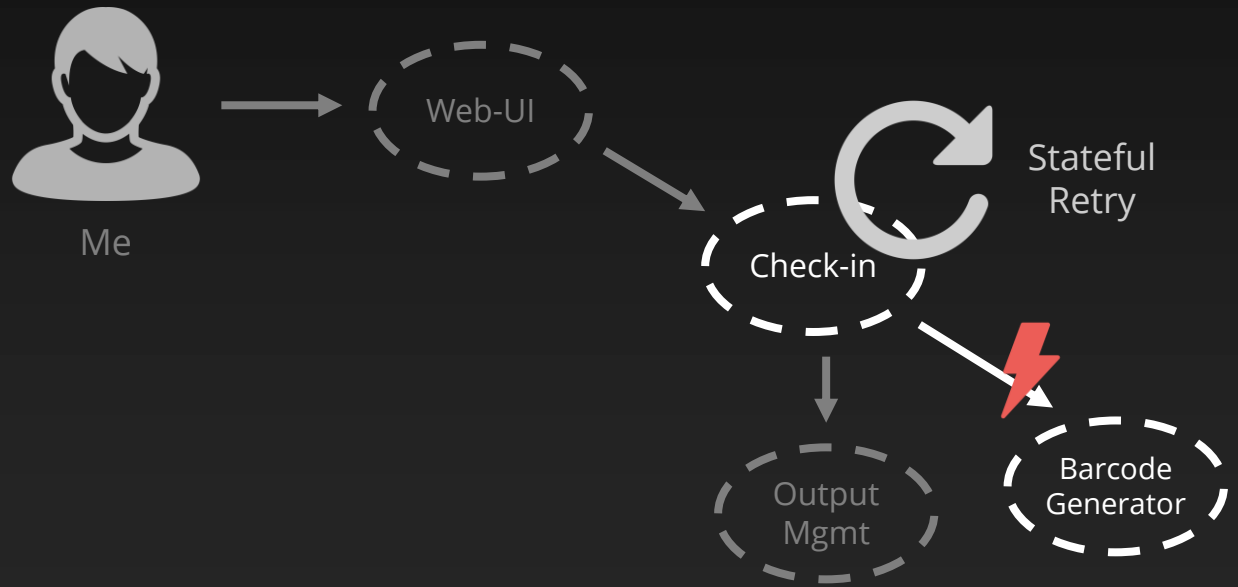
...I just made this up...



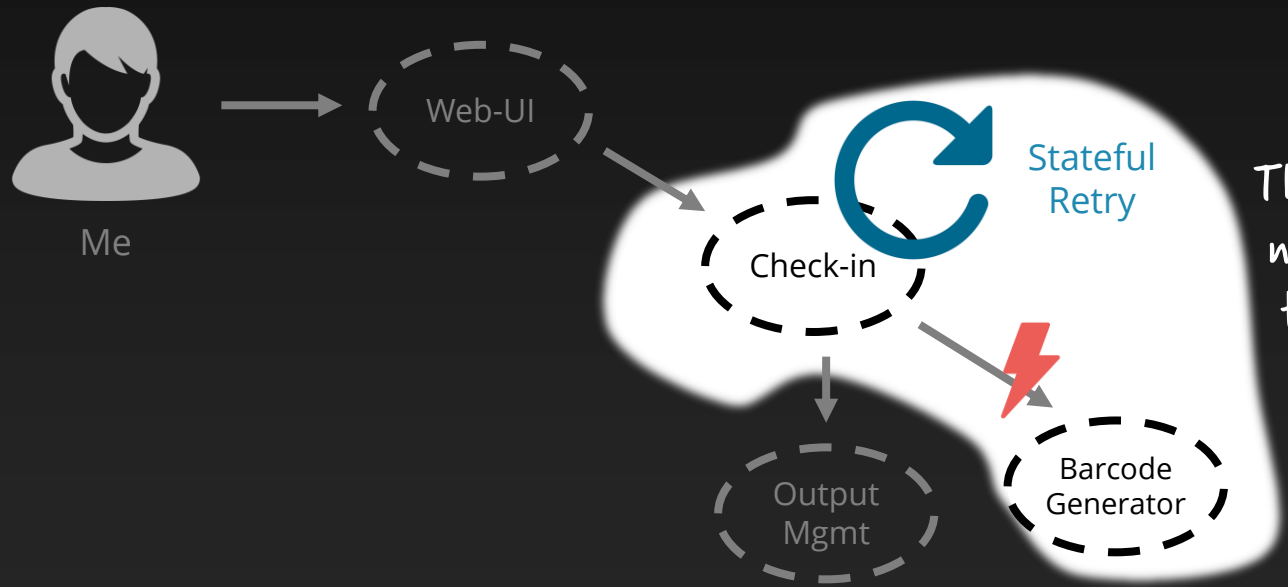
Possible situation – much better!



Possible situation – much better!

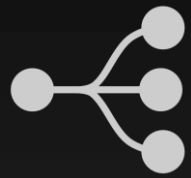


Possible situation – much better!



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

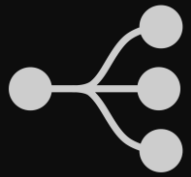
~~Complex, proprietary, heavyweight, central, developer adverse, ...~~

Avoid the wrong tools!

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

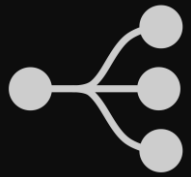
Death by properties panel

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



Workflow engines,
state machines

It is
relevant
in modern
architectures



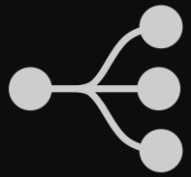
Workflow engines,
state machines

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

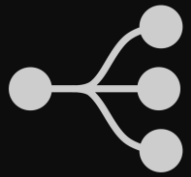
AWS Step
Functions

camunda

jBPM

Activiti™

conductor
NETFLIX OSS



Workflow engines,
state machines

UBER CADENCE

also at scale

zeebe
by Camunda

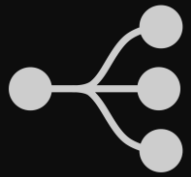
AWS Step Functions

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™

```
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.getRuntimeServi
     "flow", Variables.pu
 }
 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"));
   }
 }
```

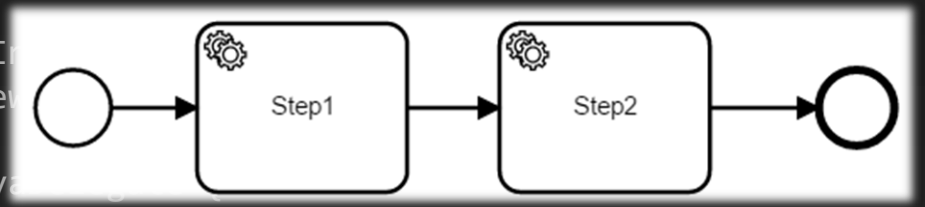
Build engine in one line of code (using in-memory H2)

```
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().startProcessI
     "flow", Variables.putValue("city", "Ne
 }
 public class SysoutDelegate implements Java
   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...

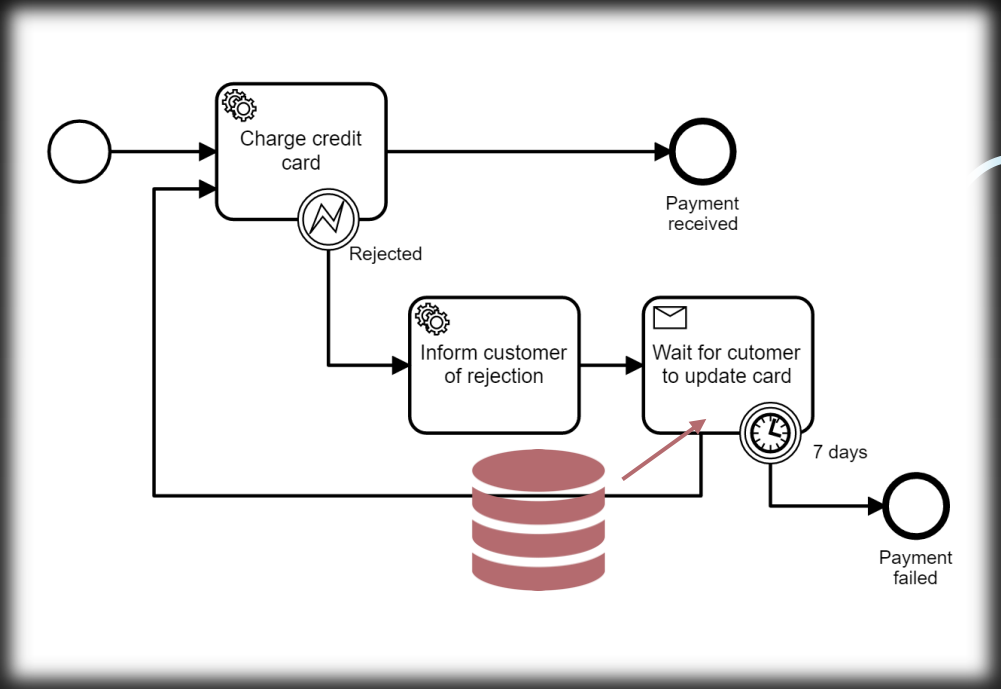
```
public static void main(String[] args) {
   ProcessEngine engine = new StandaloneInMemProcessEngineConfiguration()
     .buildProcessEngine();

   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"));
   }
 }
```

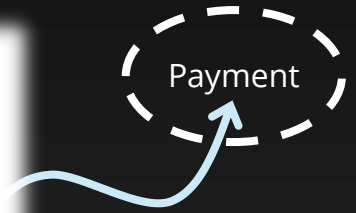
...that is called when workflow instances pass through

```
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"));
   }
 }
```

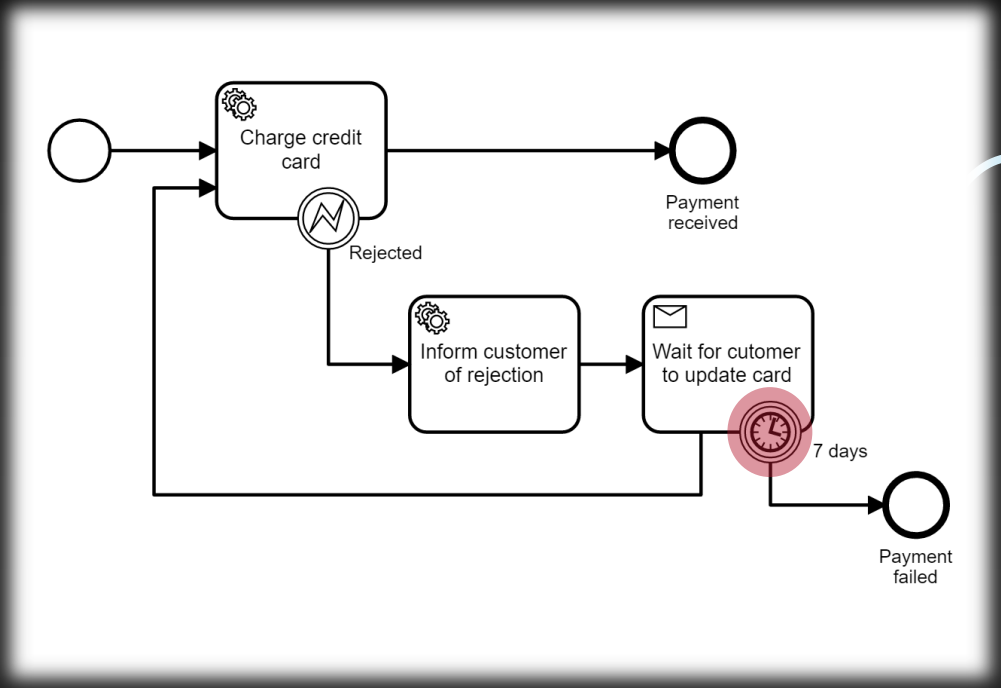
Start instances

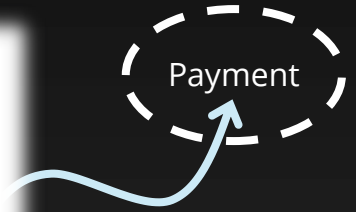
Now you have a state machine!



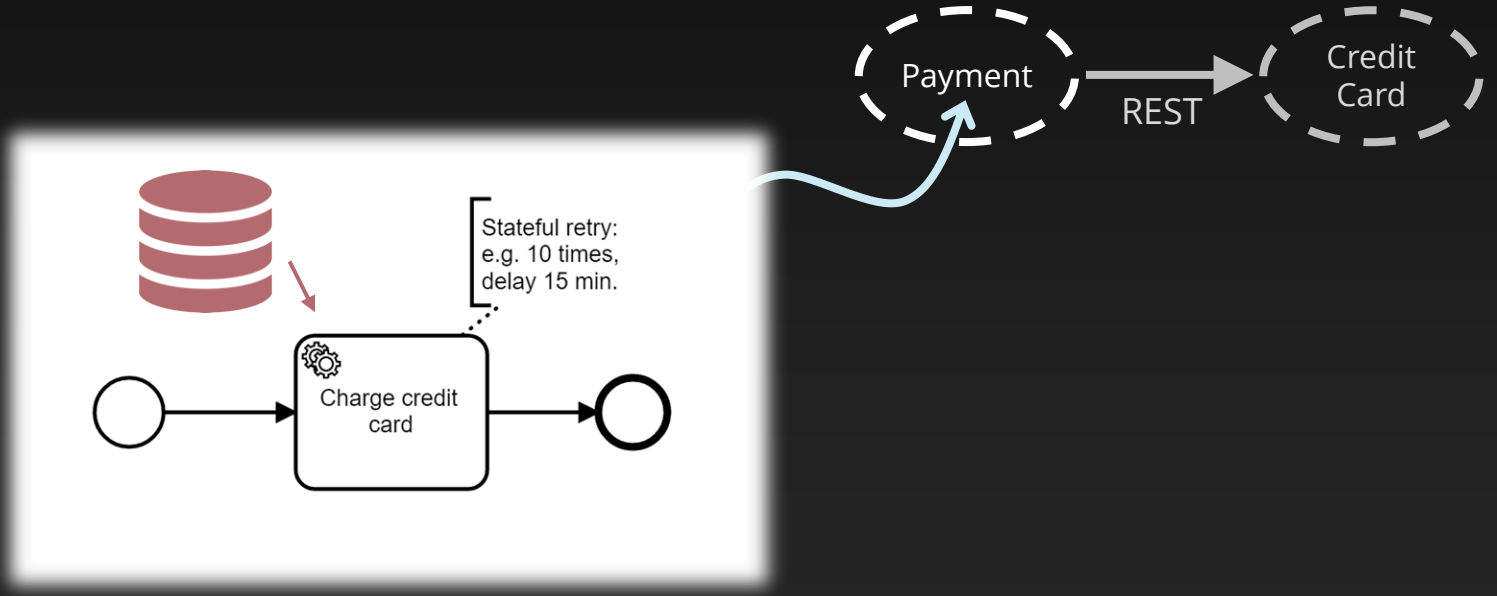


Easy to handle time

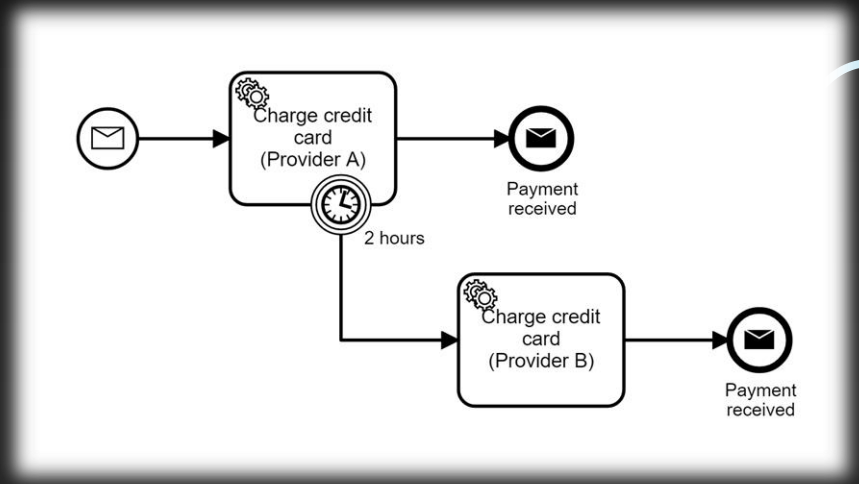




Stateful retries



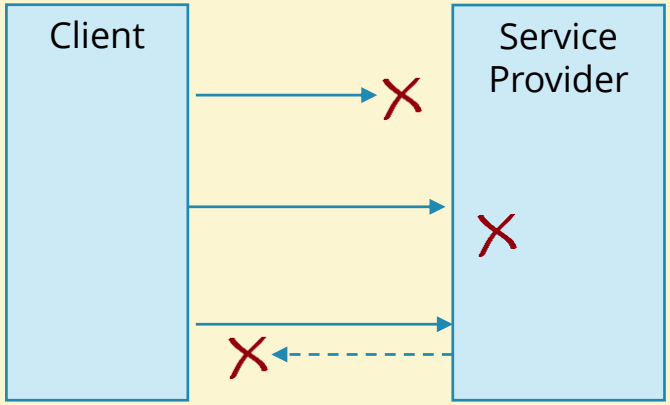
Fallbacks increase resilience



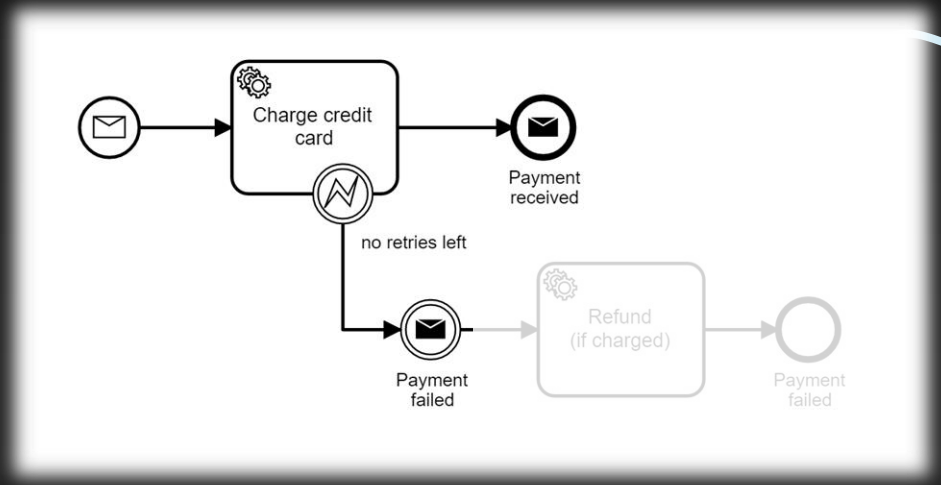
Distributed systems

It is impossible to differentiate certain failure scenarios.

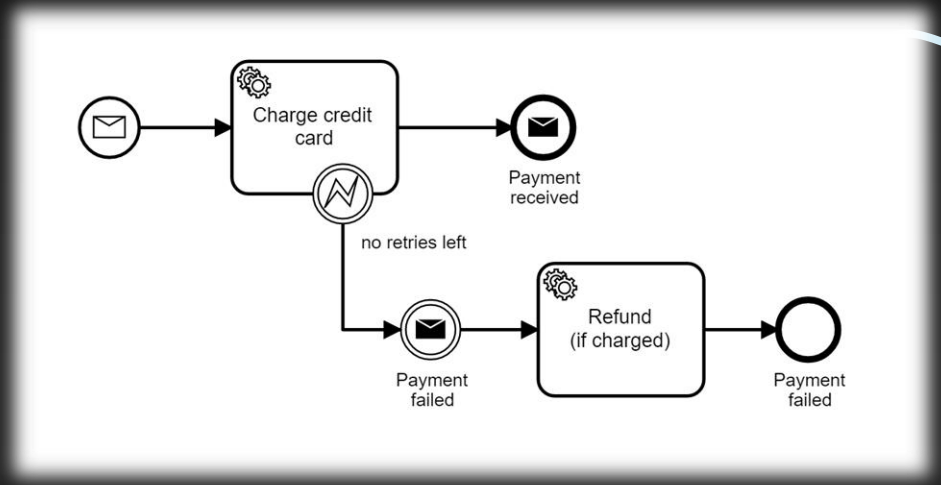
Independant of communication style!

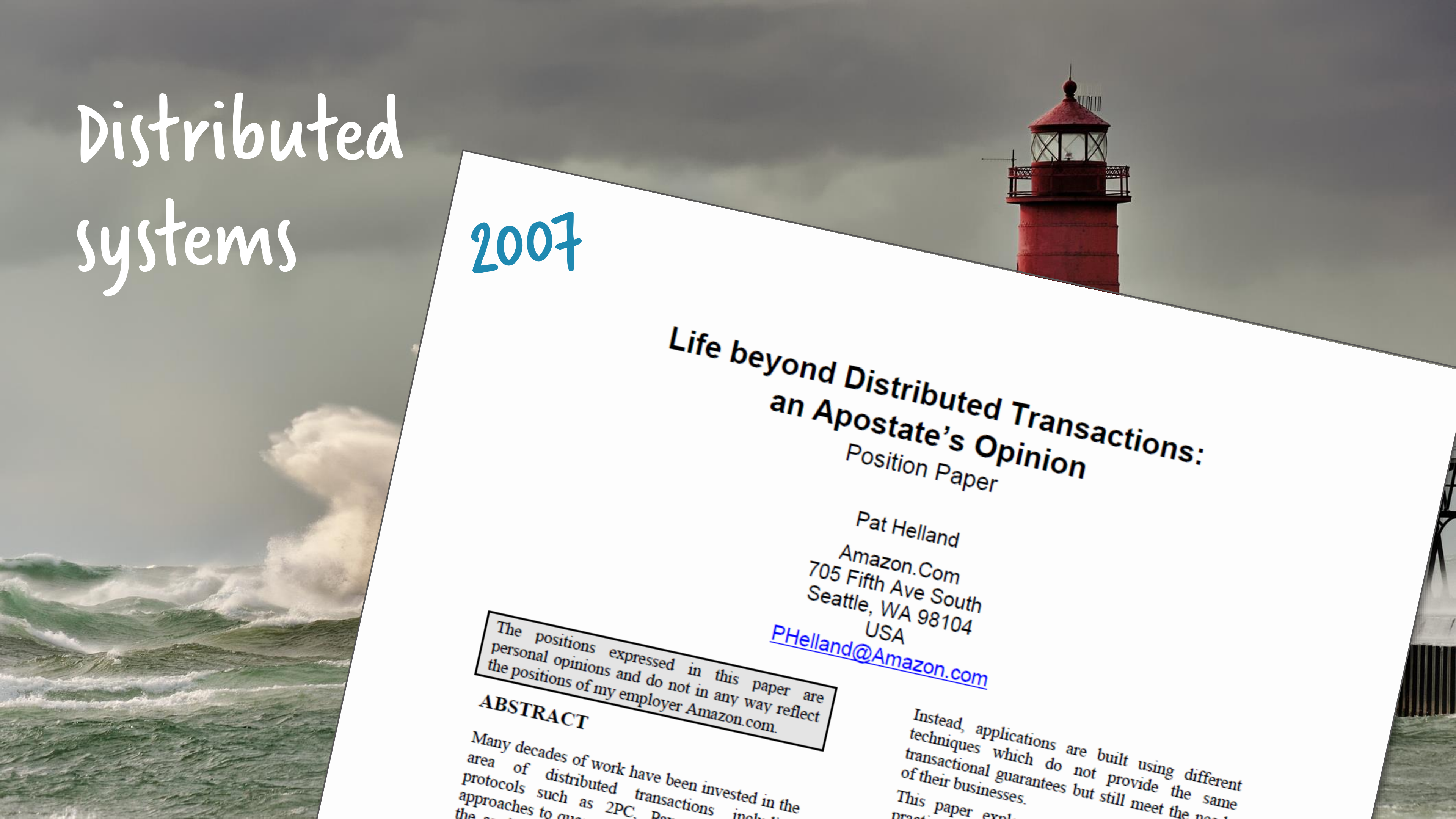


Distributed systems introduce complexity you have to tackle!



Distributed systems introduce complexity you have to tackle!





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

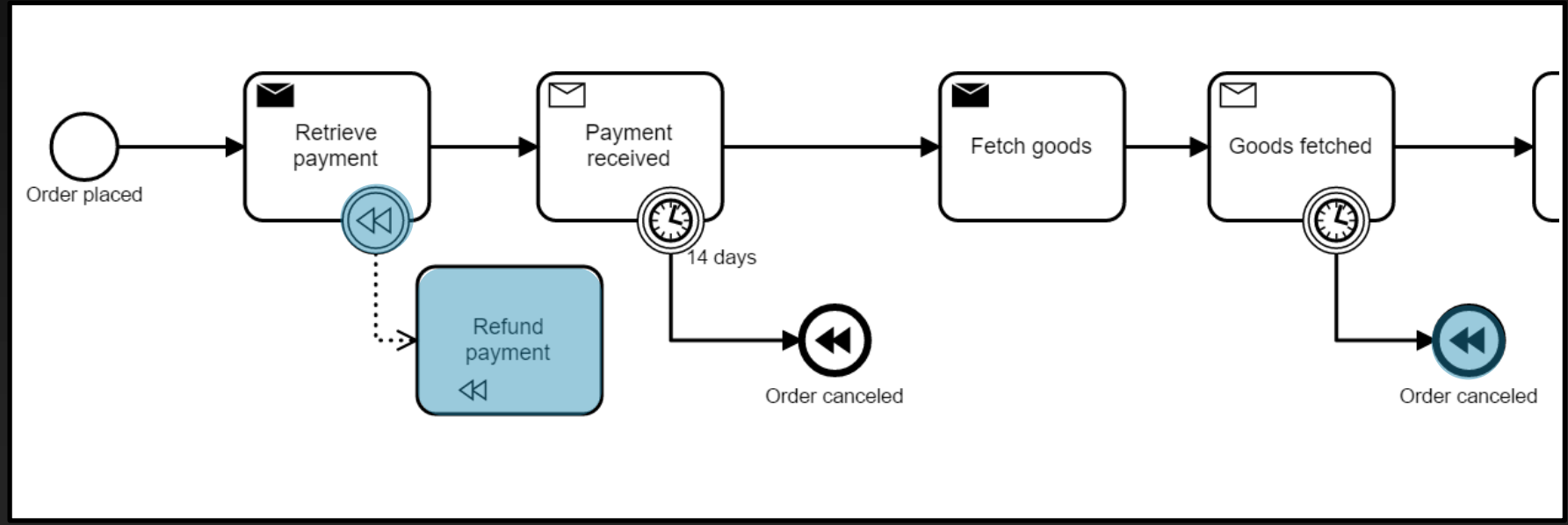
Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC, ...

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

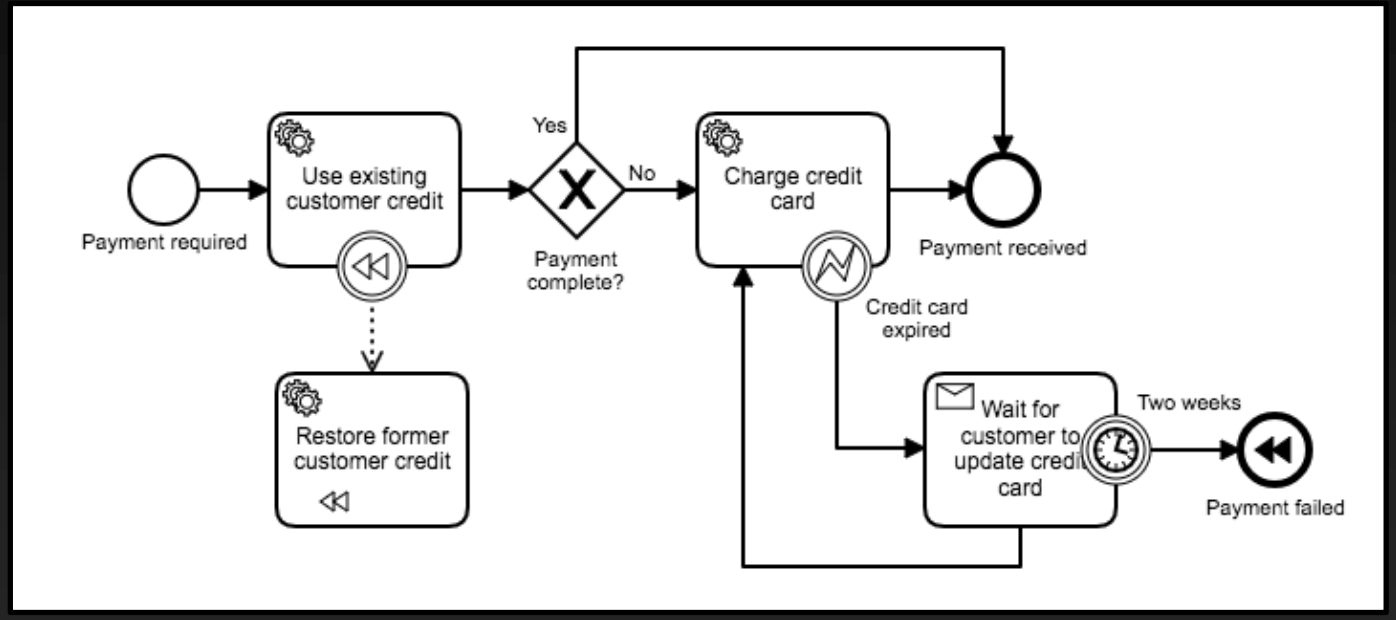
This paper ...

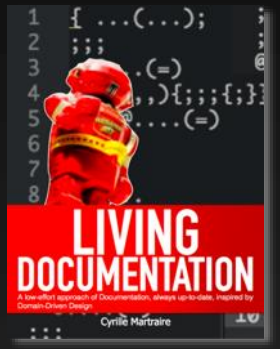
Distributed transaction with compensations*

*aka Saga-Pattern

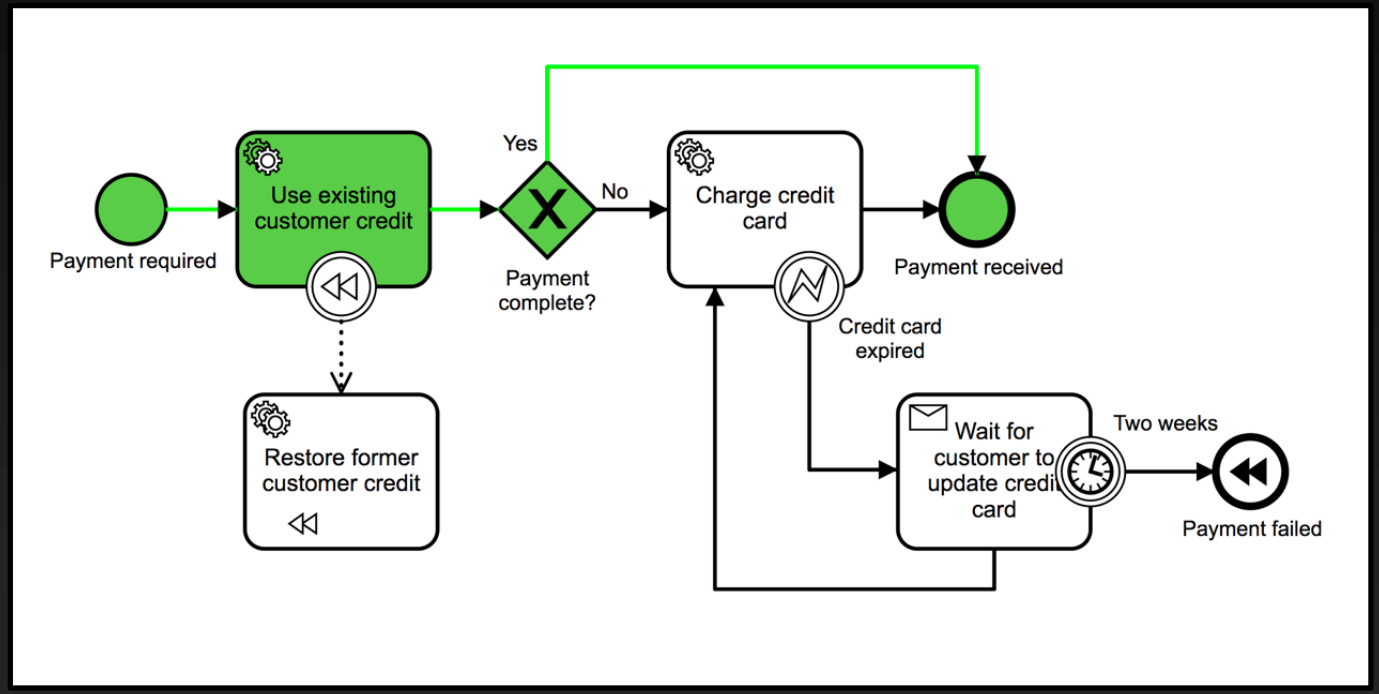


Living documentation for long-running behaviour



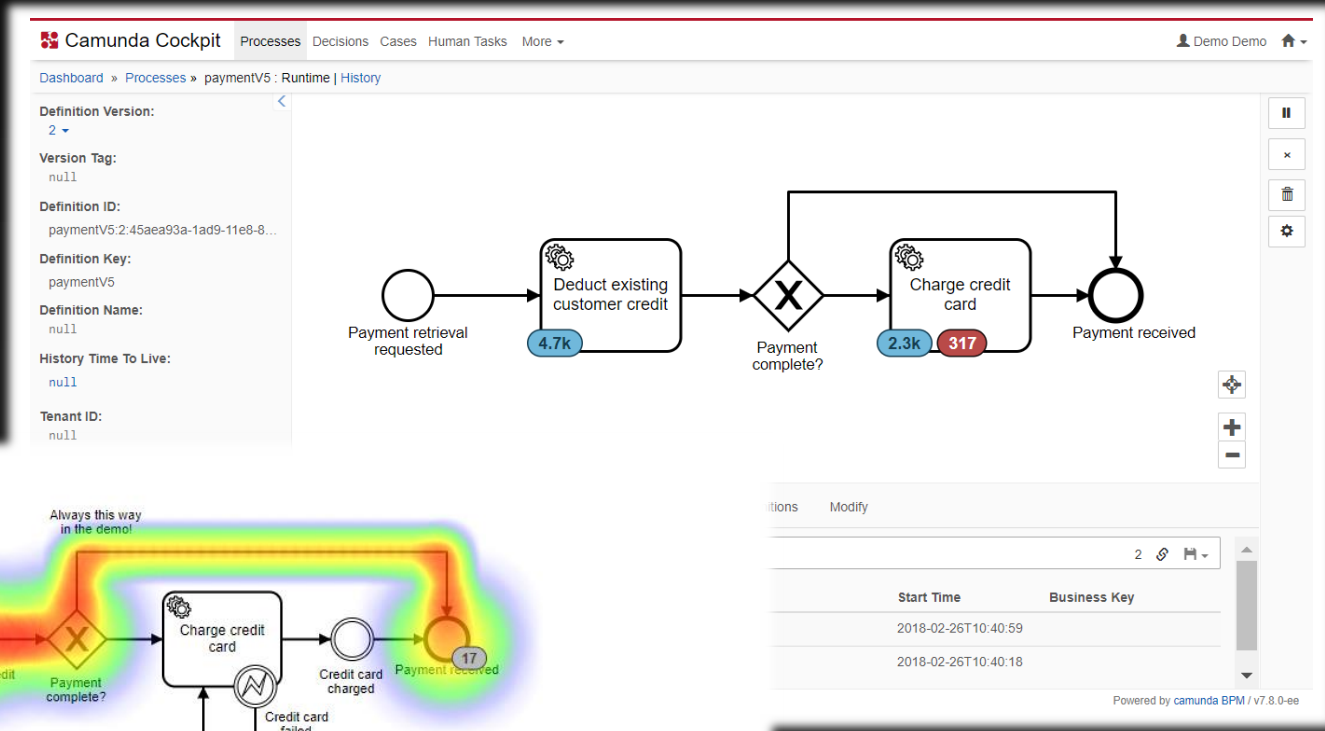


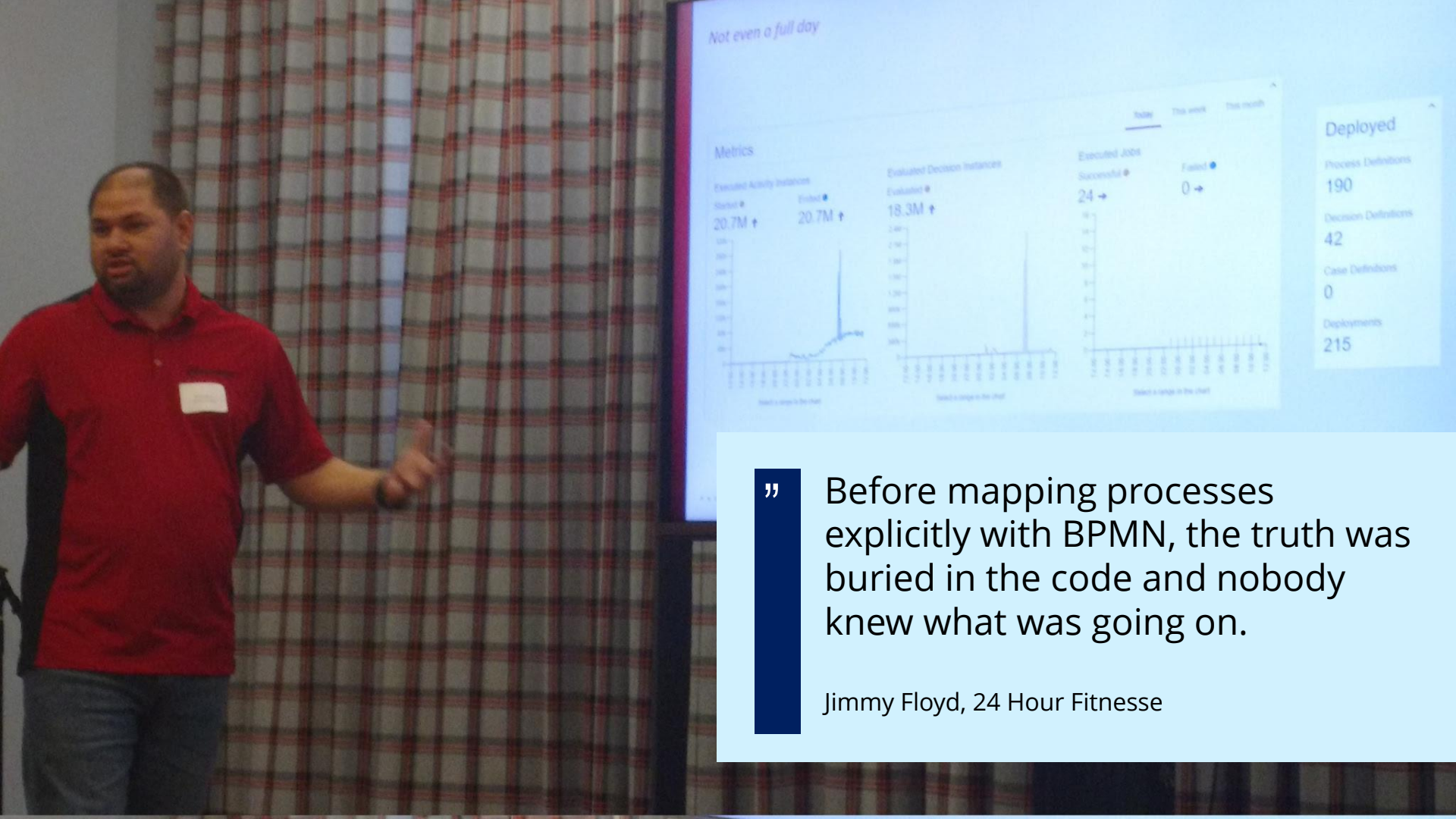
Visual HTML reports for test cases



Proper operations

Visibility + Context

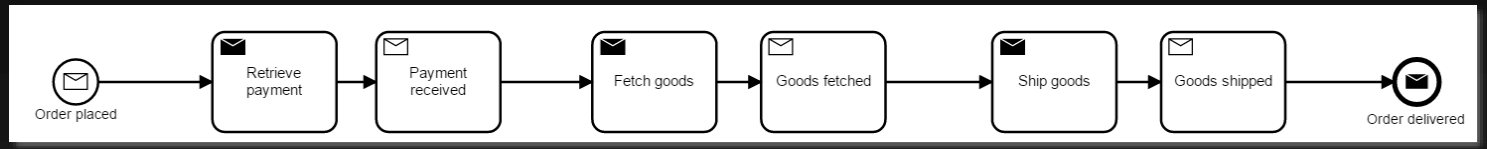


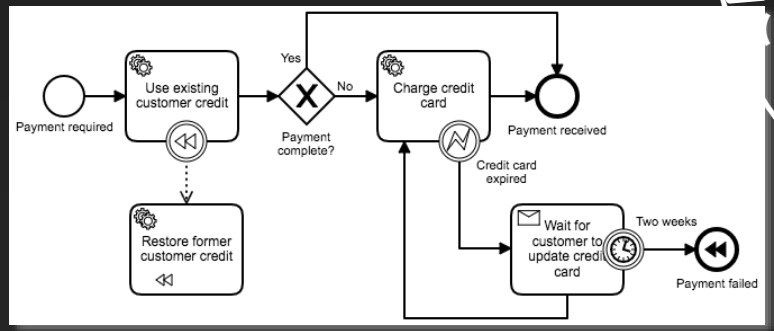


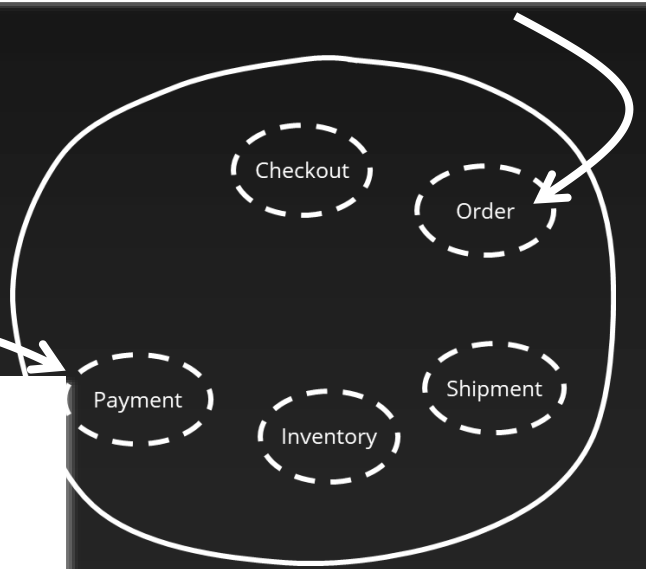
> Before mapping processes explicitly with BPMN, the truth was buried in the code and nobody knew what was going on.

Jimmy Floyd, 24 Hour Fitnesse

Workflows live inside service boundaries



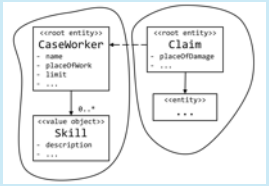


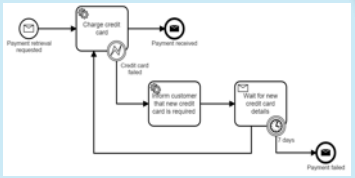


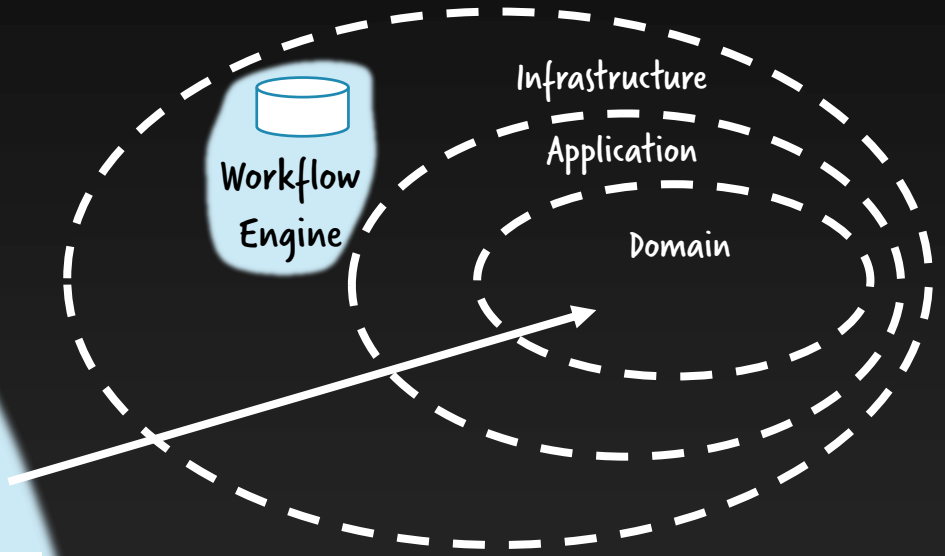
Explicit flows help separate domain and infrastructure

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

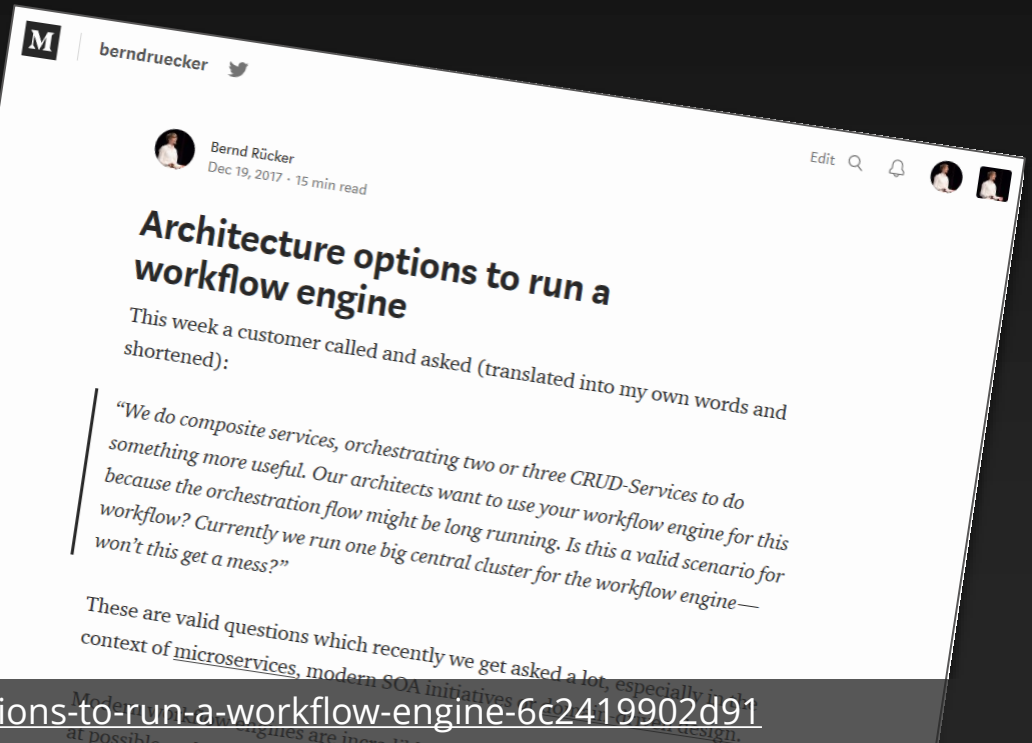
+ the flow





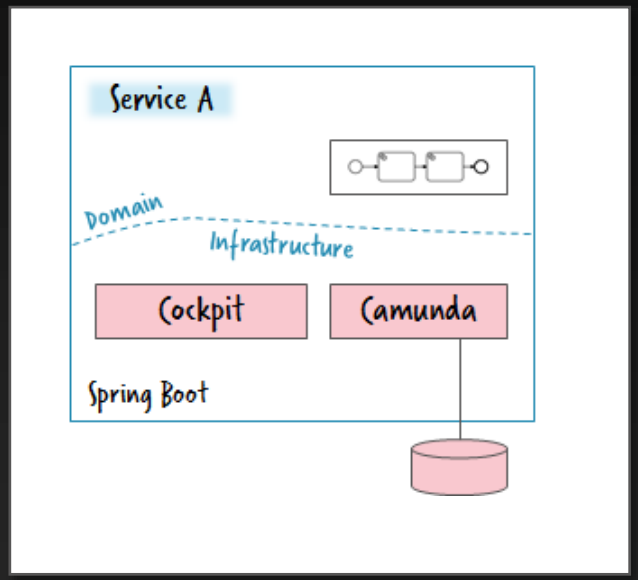


Manigfold architecture options



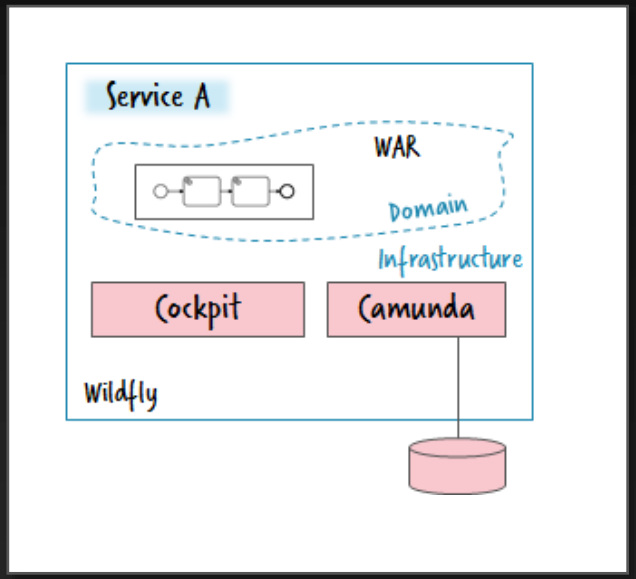
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Manigfold architecture options



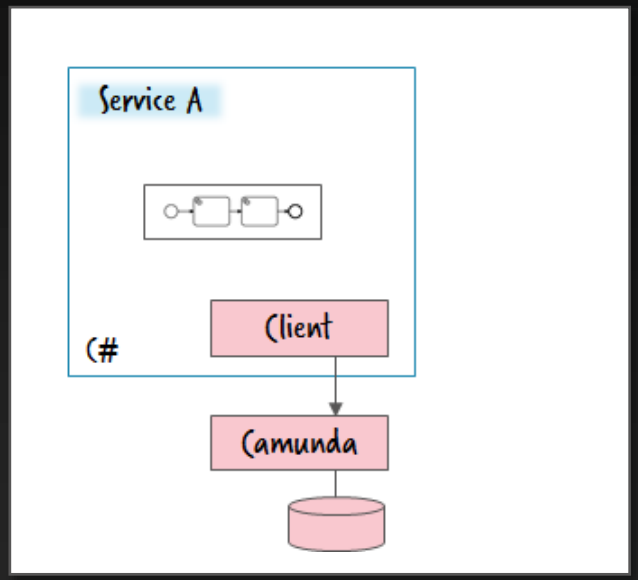
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Manigfold architecture options



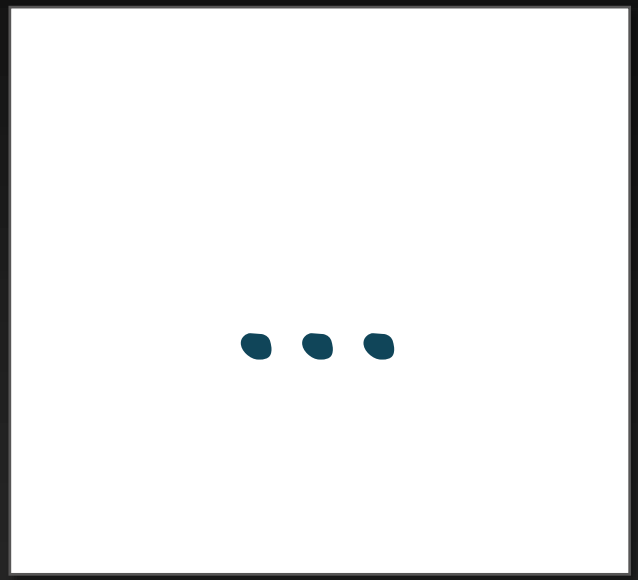
https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Manigfold architecture options



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

Manigfold architecture options

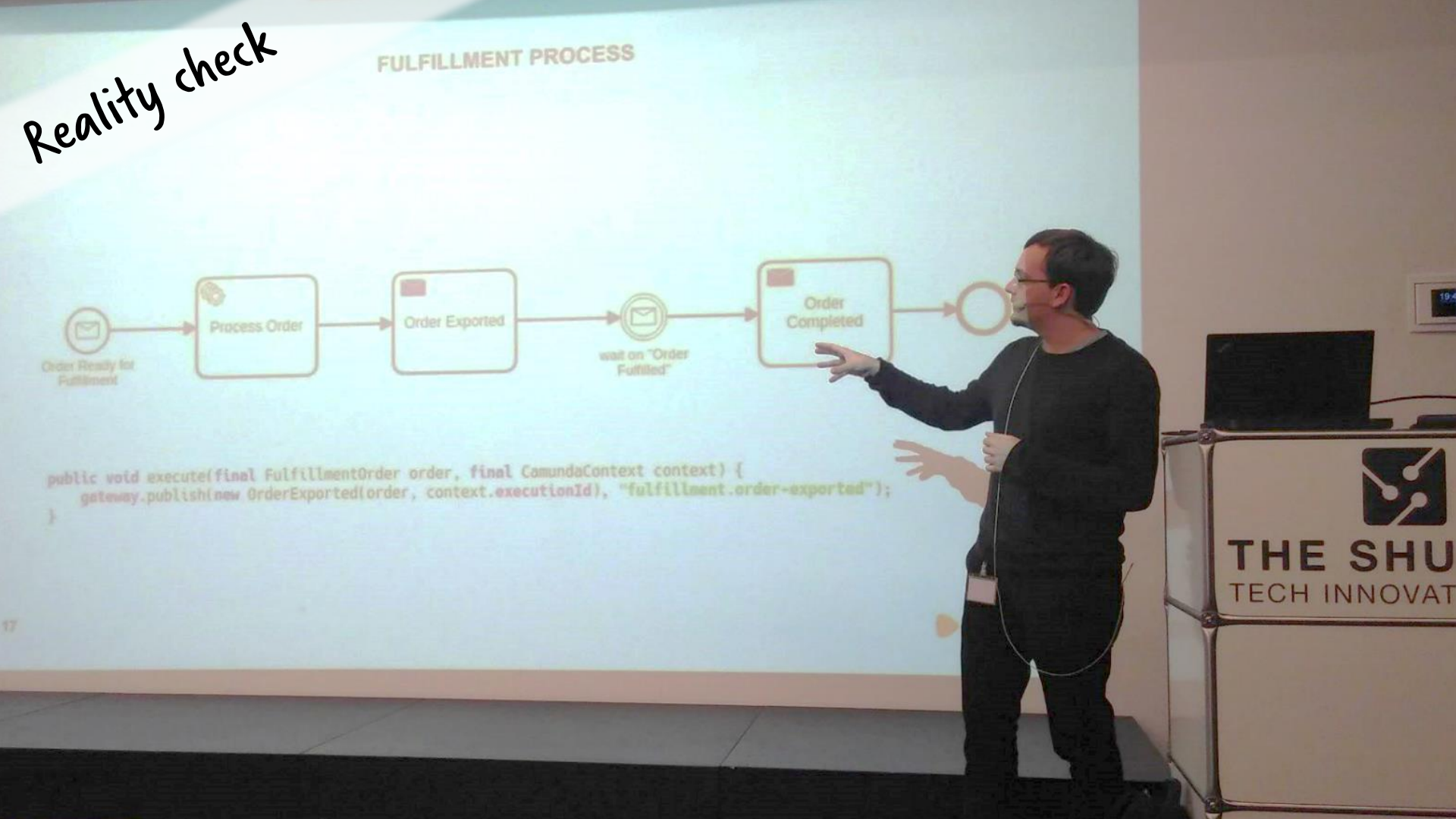


https://blog.bernd-ruecker.com/architecture-options-to-run-a-workflow-engine-6c2419902d91

Lightweight workflow engines are great – don't DIY*

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

Reality check



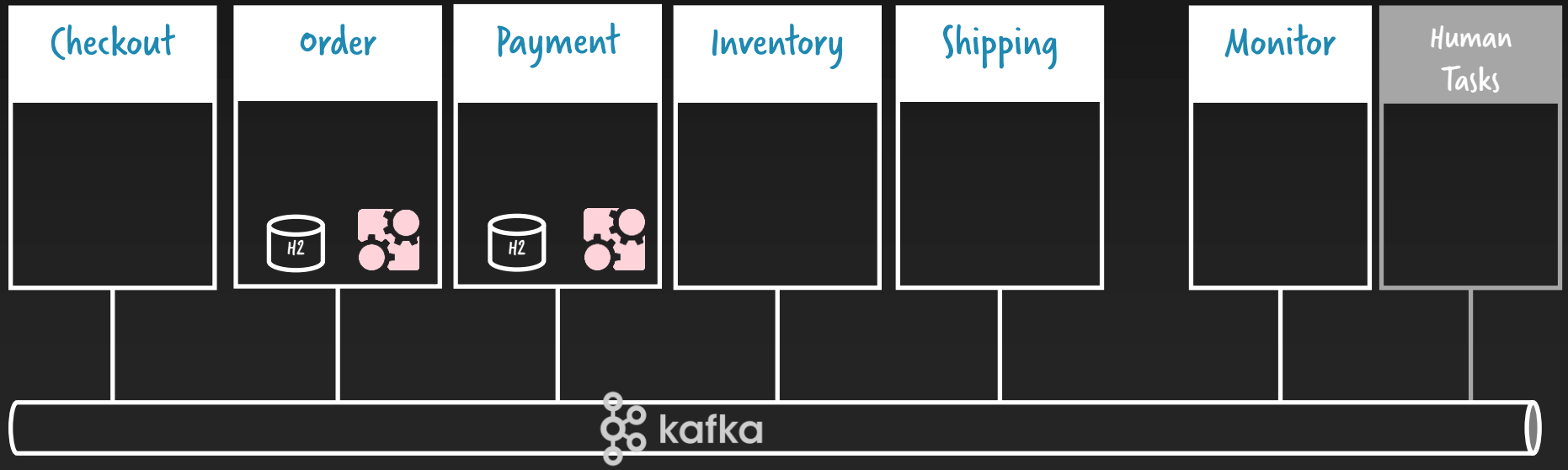
Zalando

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%

Code example & live demo



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

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-integrationand-how-to-avoid-them.html

InfoQ

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

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