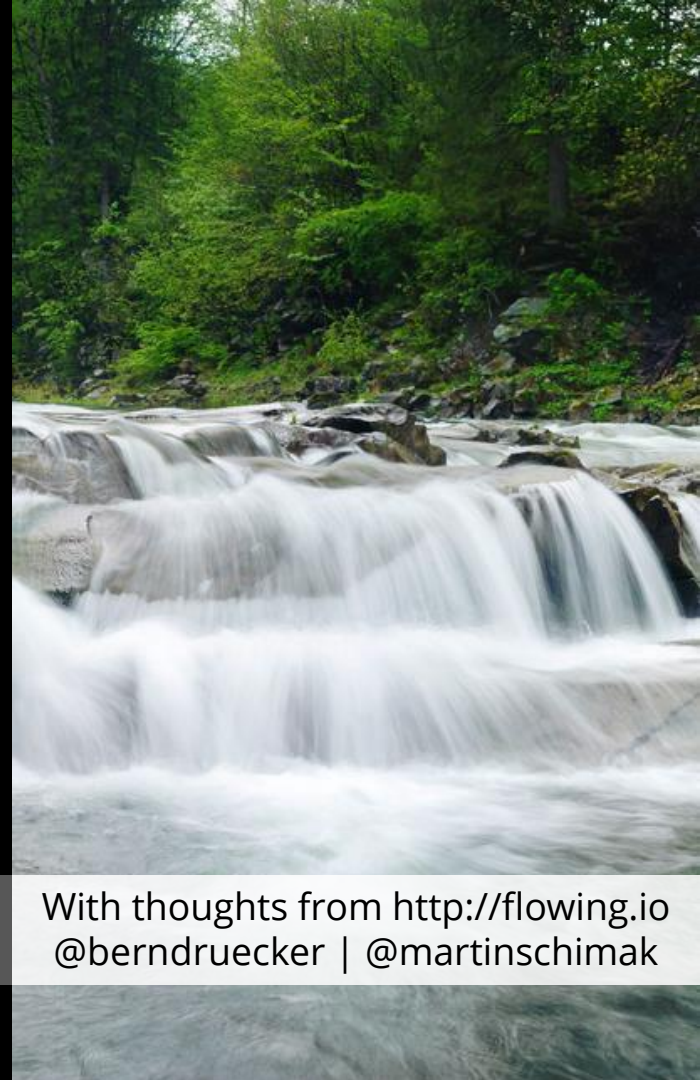


Complex event flows in distributed systems

@berndruecker

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



3 common hypotheses I check today:

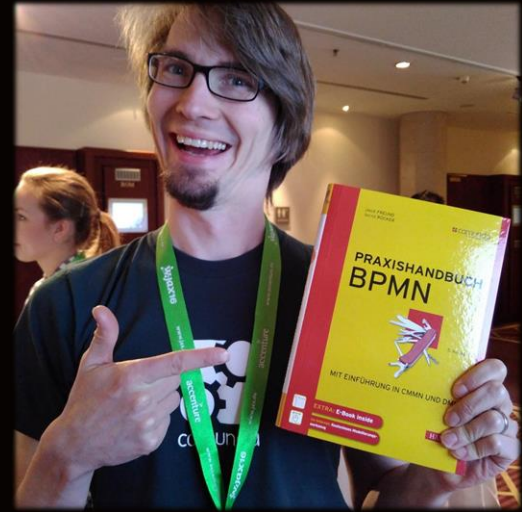
Events decrease coupling

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
LUCAS NIEMEIER

2017-09-19



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/>

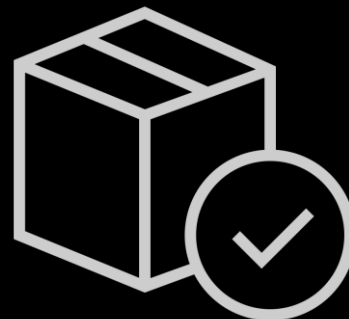
Three steps...



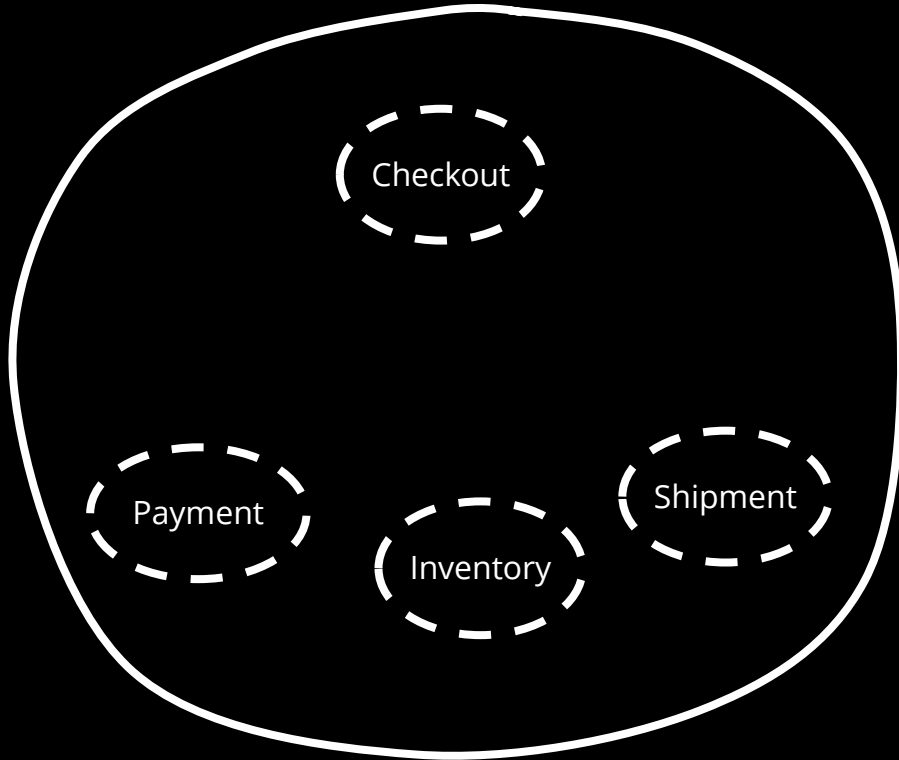
Pay item

Fetch item

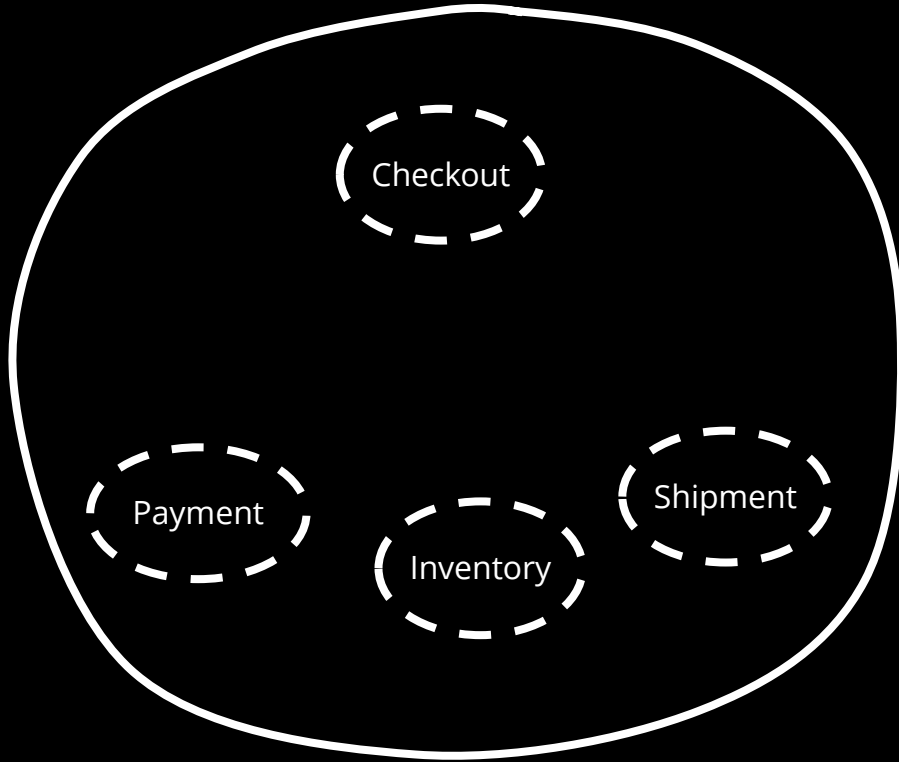
Ship item



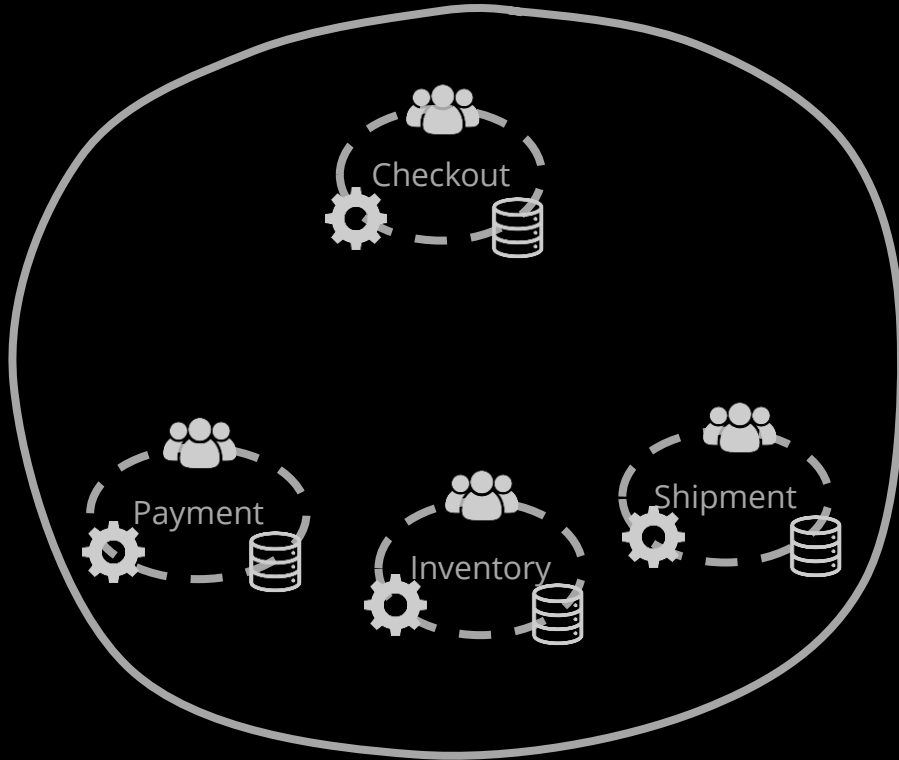
Who is involved? Some bounded contexts...



(Micro-)services



Autonomous (micro-)services



Dedicated Application Processes



Dedicated infrastructure

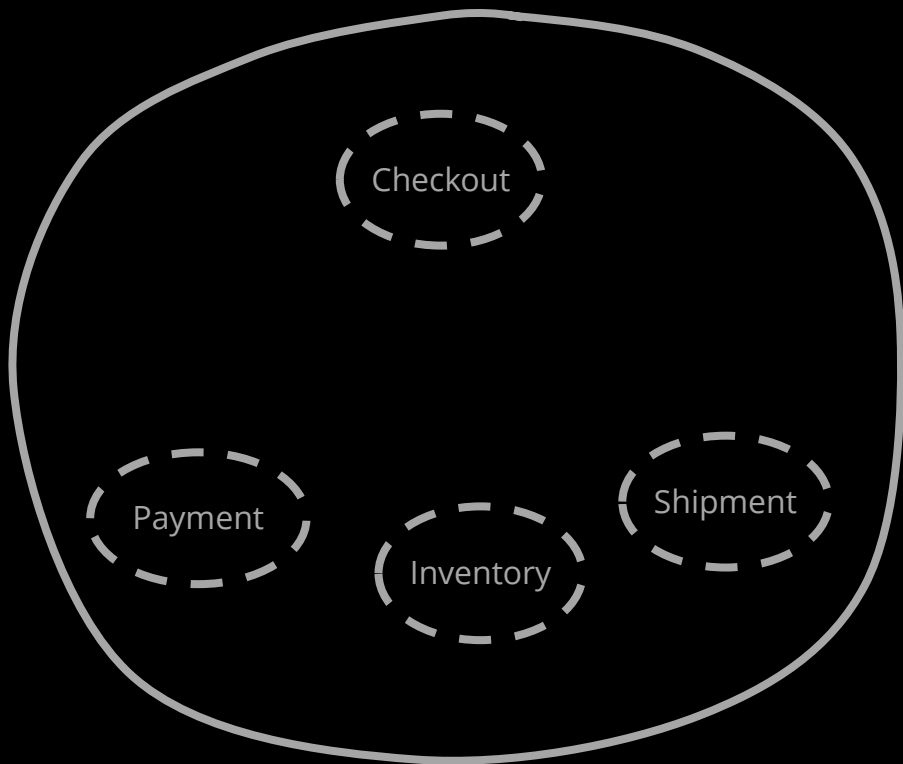


Dedicated Development Teams

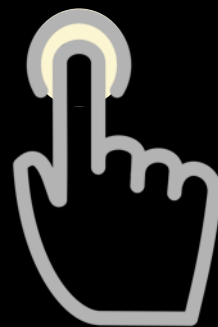


Events decrease coupling

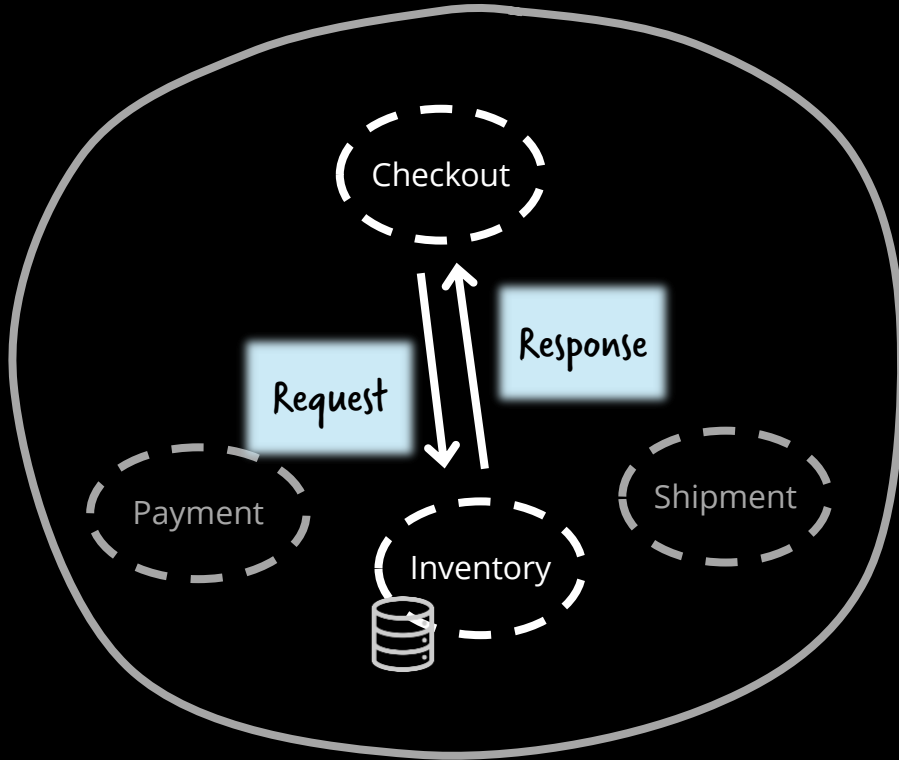
Example



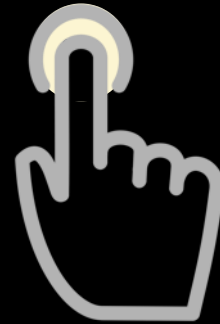
The button blinks if we can ship within 24 hours



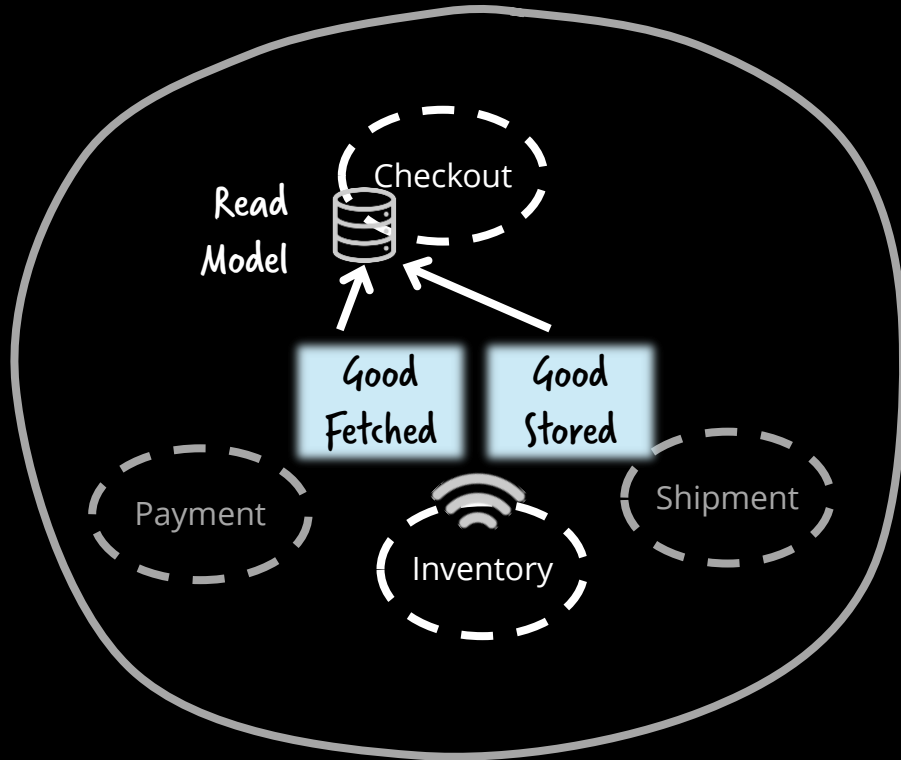
Request/response: temporal coupling



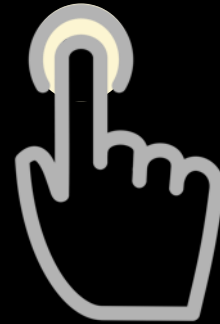
The button blinks if we can ship within 24 hours



Temporal decoupling with events and read models



The button blinks if we can ship within 24 hours

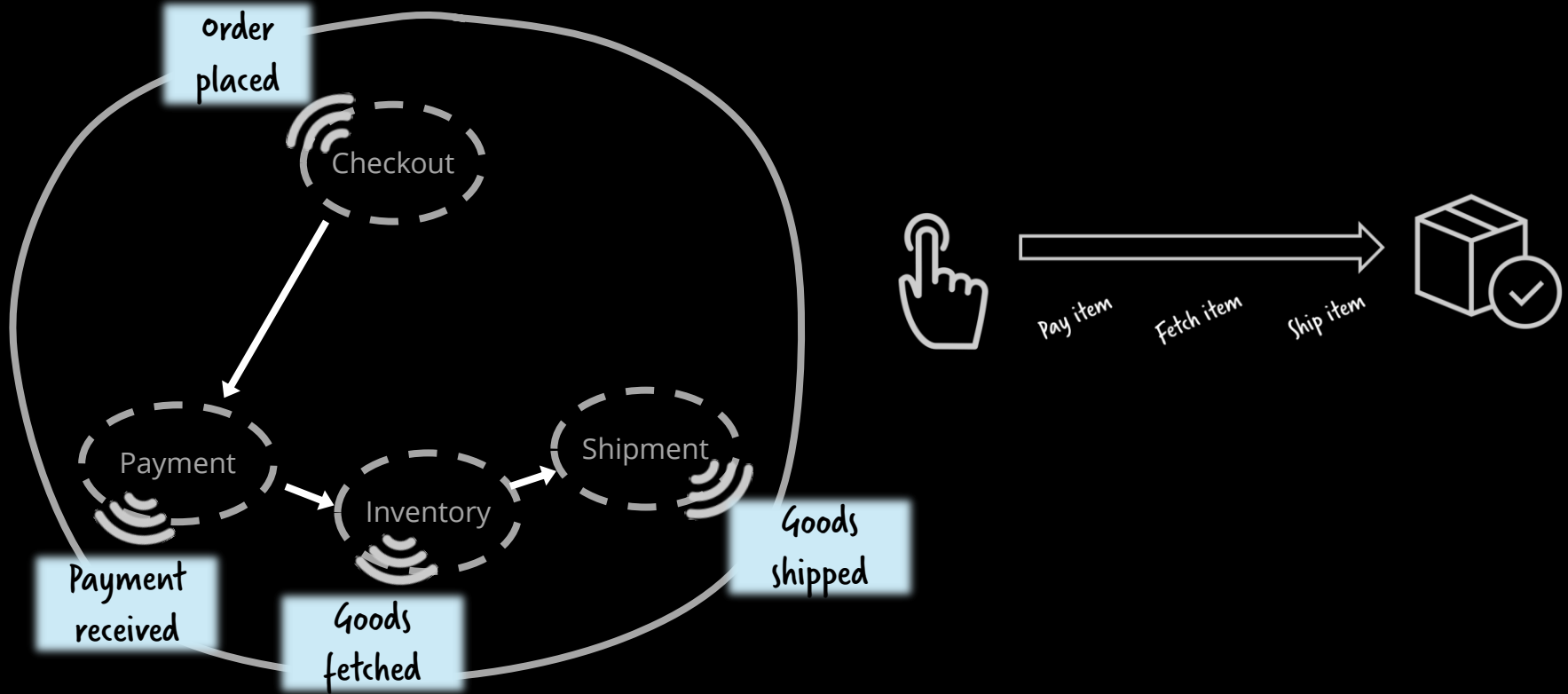


***Events** are facts about what happened (in the past)

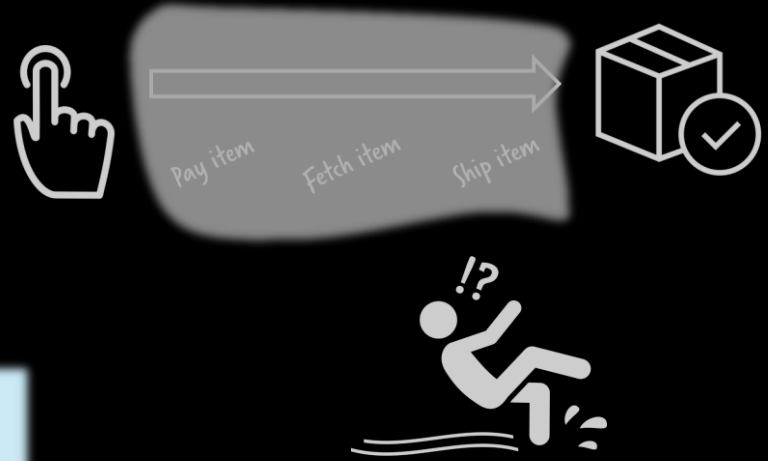
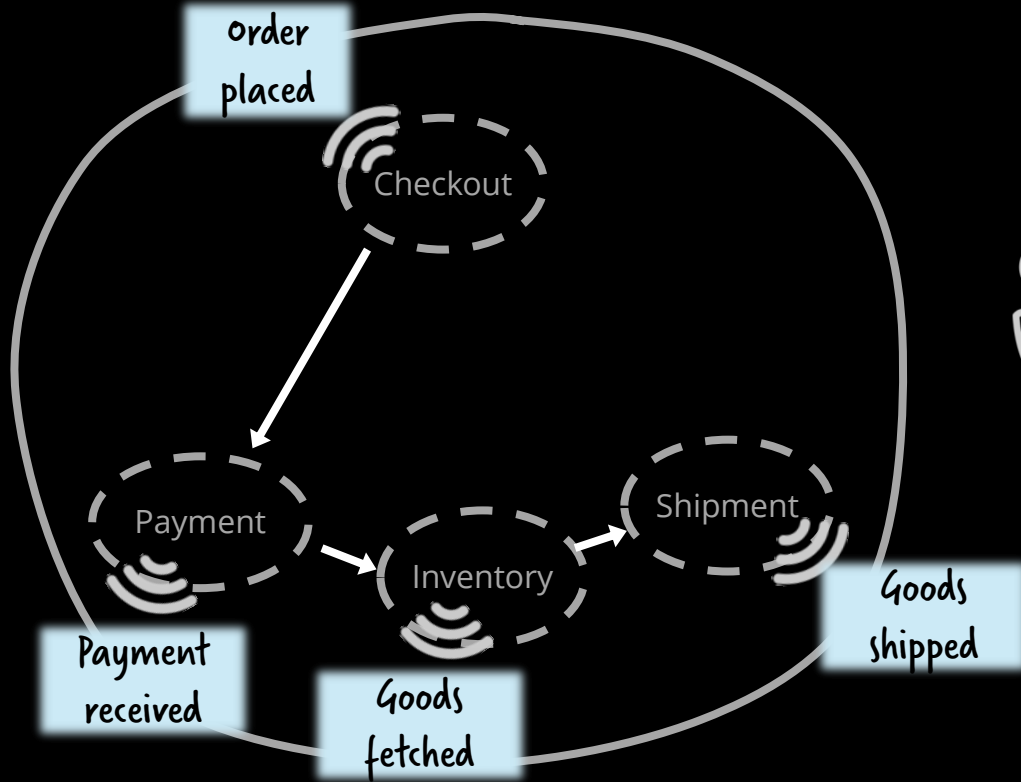
Events can decrease coupling*

*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.

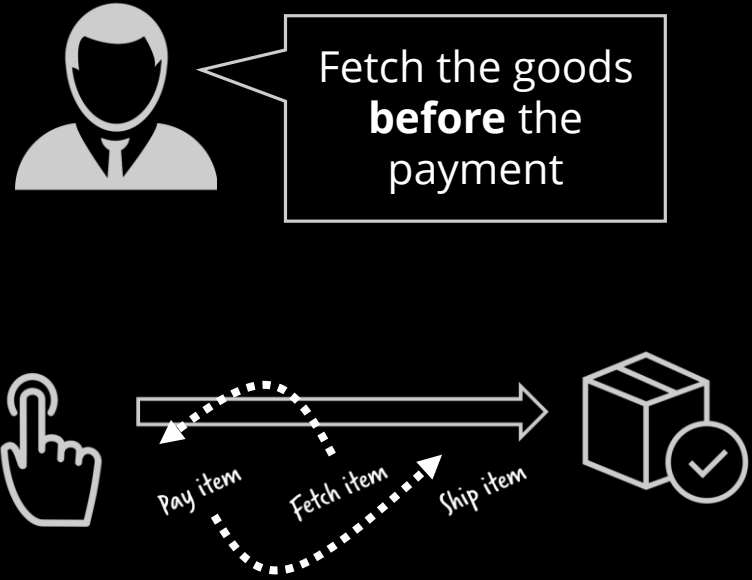
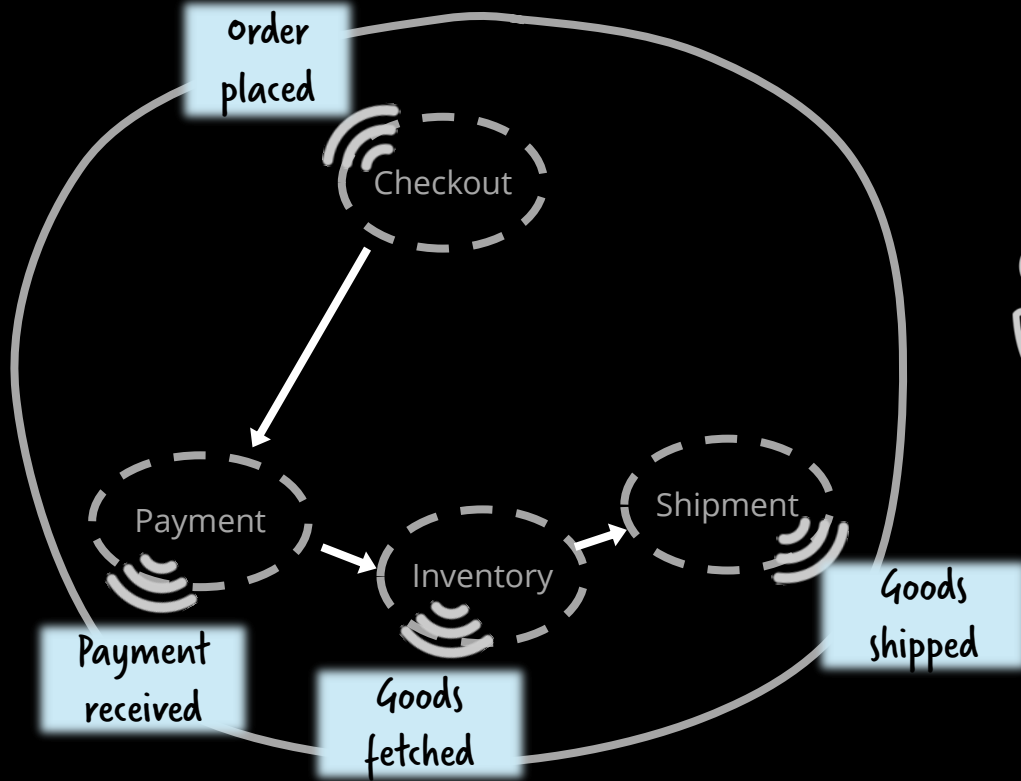


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

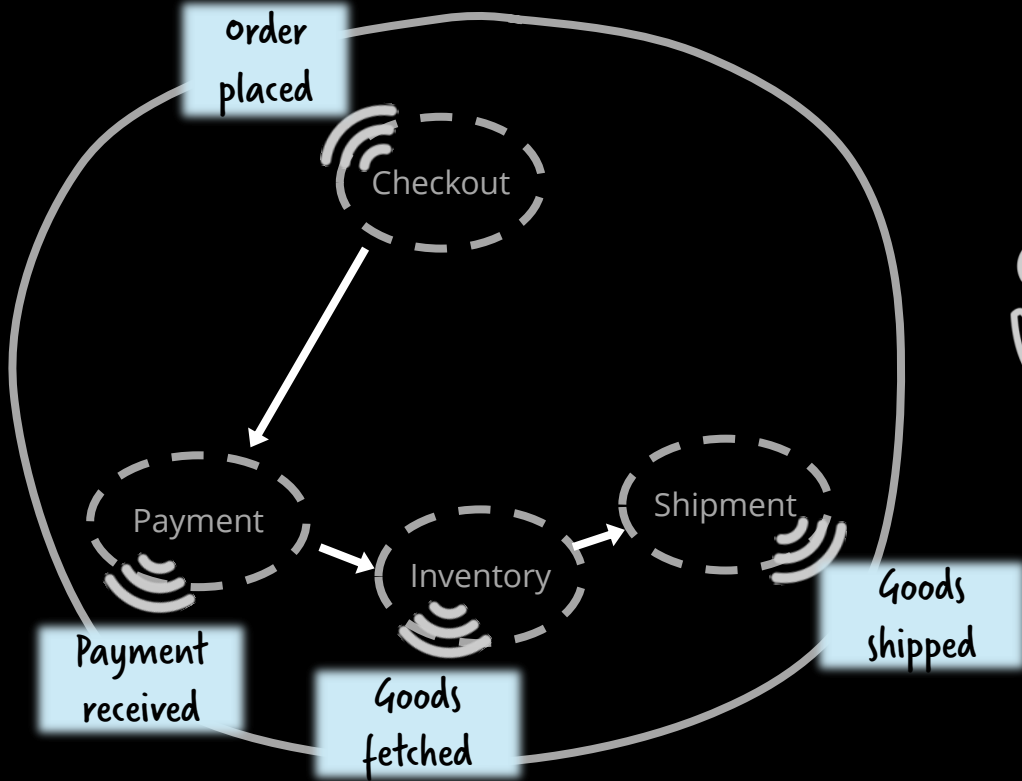


The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

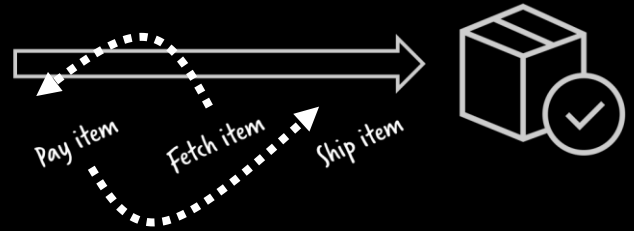
Peer-to-peer event chains



Peer-to-peer event chains



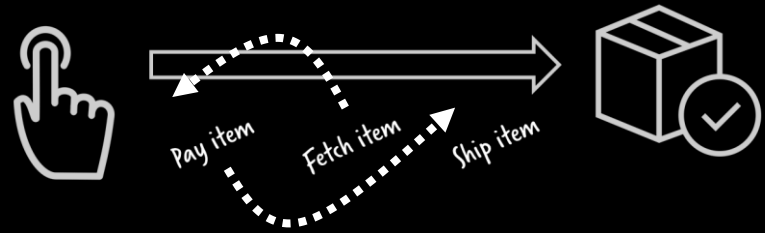
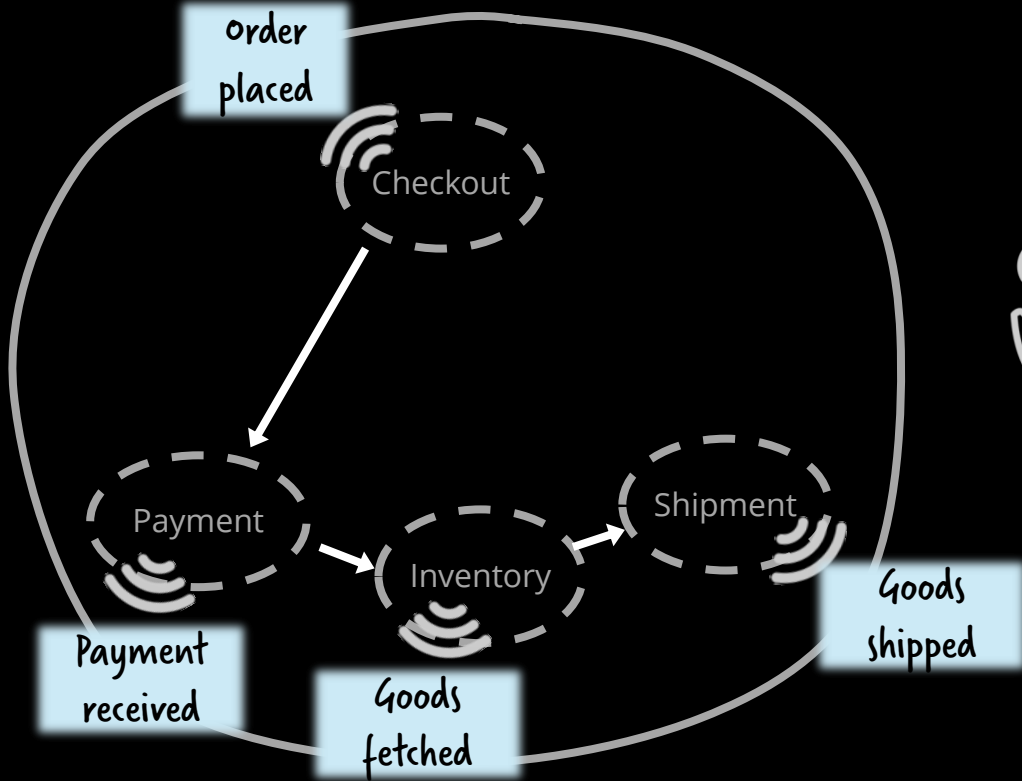
Fetch the goods **before** the payment



Customers can pay via invoice

...

Peer-to-peer event chains



Peer-to-peer event chains

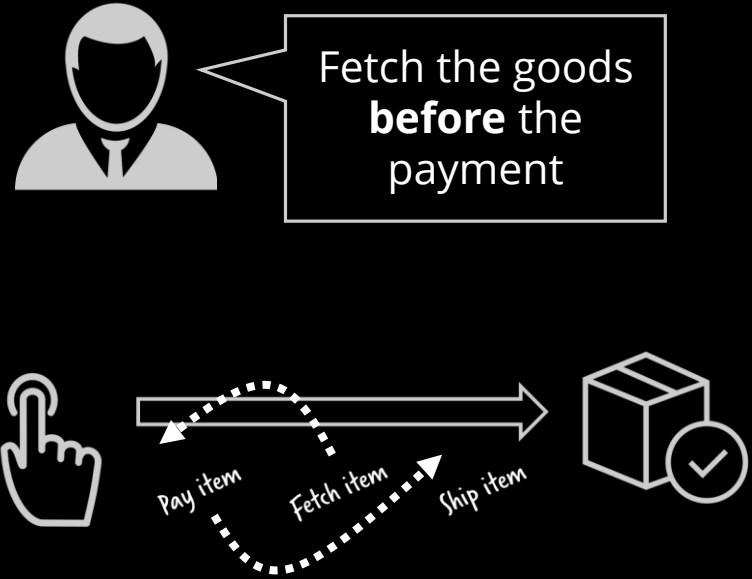
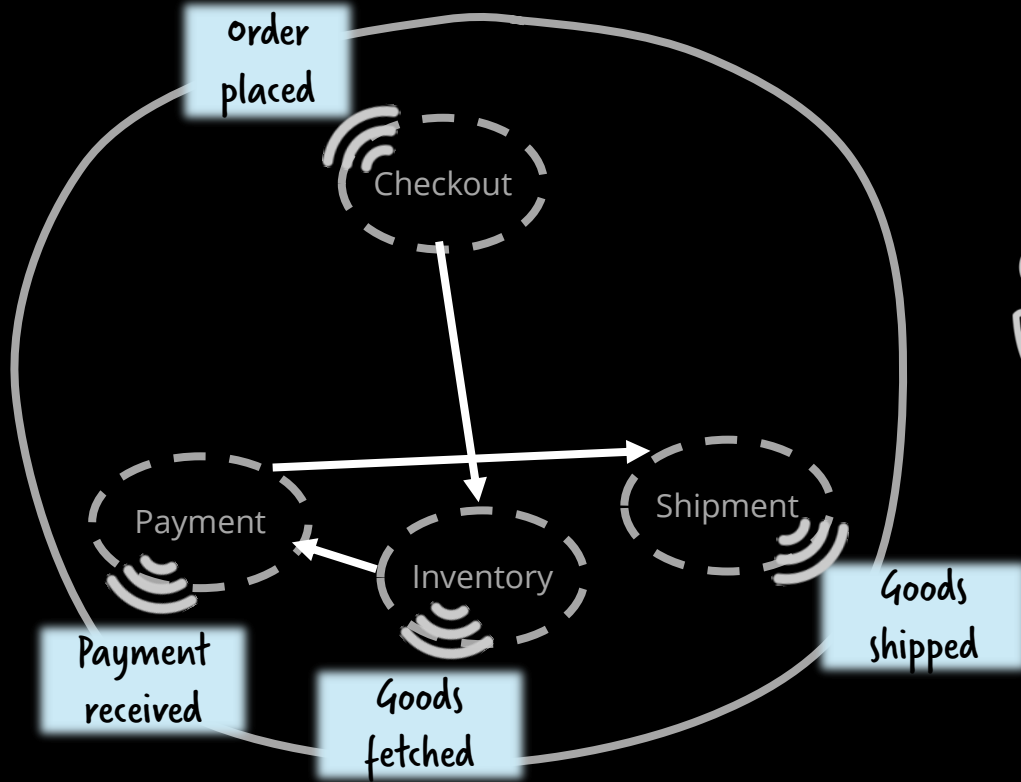
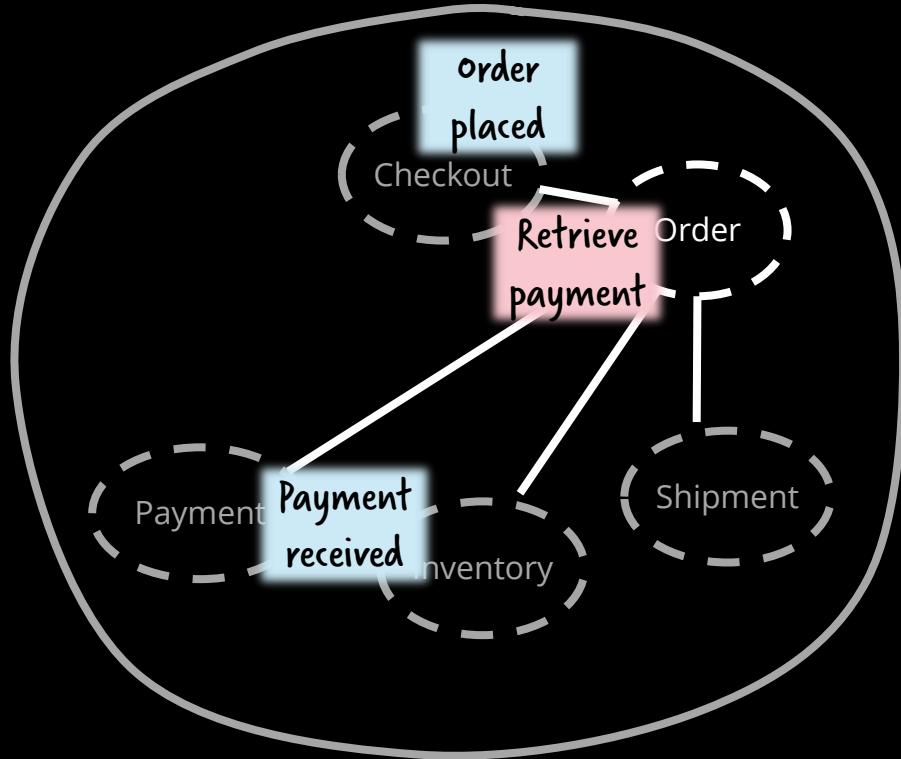




Photo by born1945, available under [Creative Commons BY 2.0 license](https://creativecommons.org/licenses/by/2.0/).

Extract the end-to-end responsibility



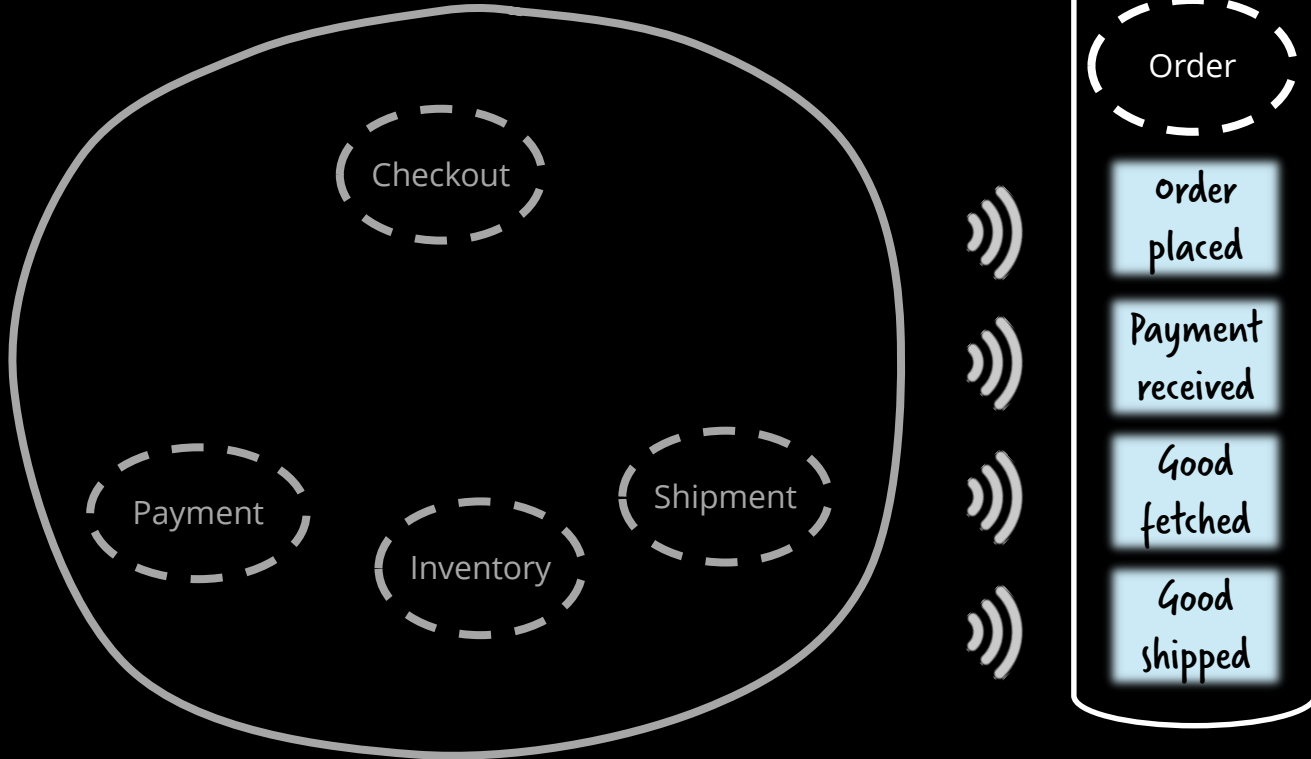
***Commands** have an intent about what needs to happen in the future

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

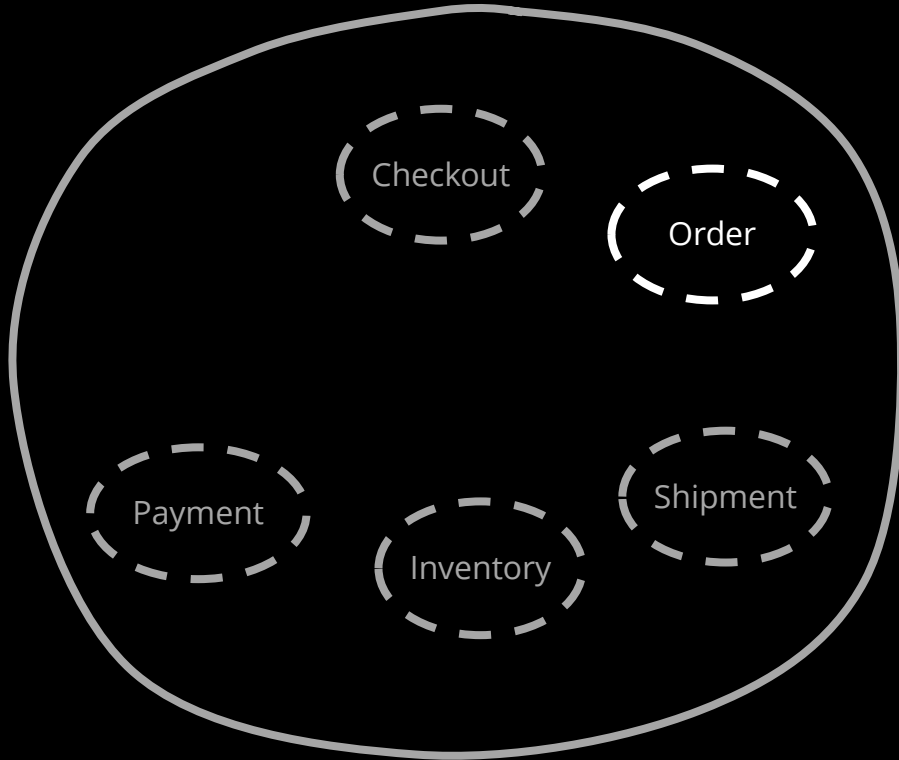


orchestration needs to be avoided

Smart ESB-like middleware



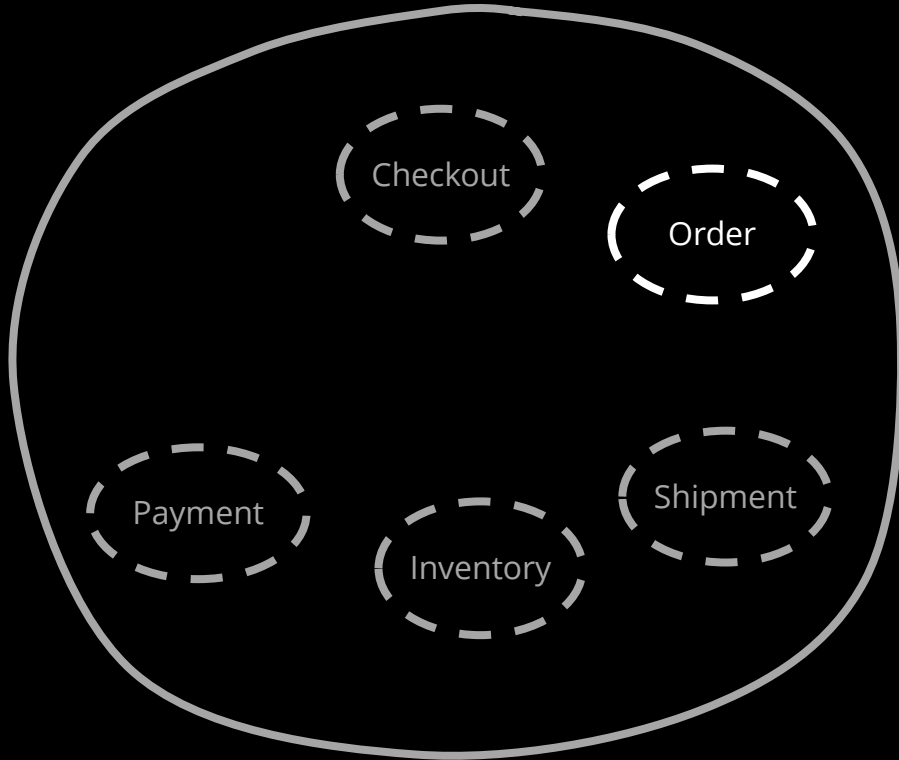
Dumb pipes



Martin Fowler

Smart endpoints
and **dumb pipes**

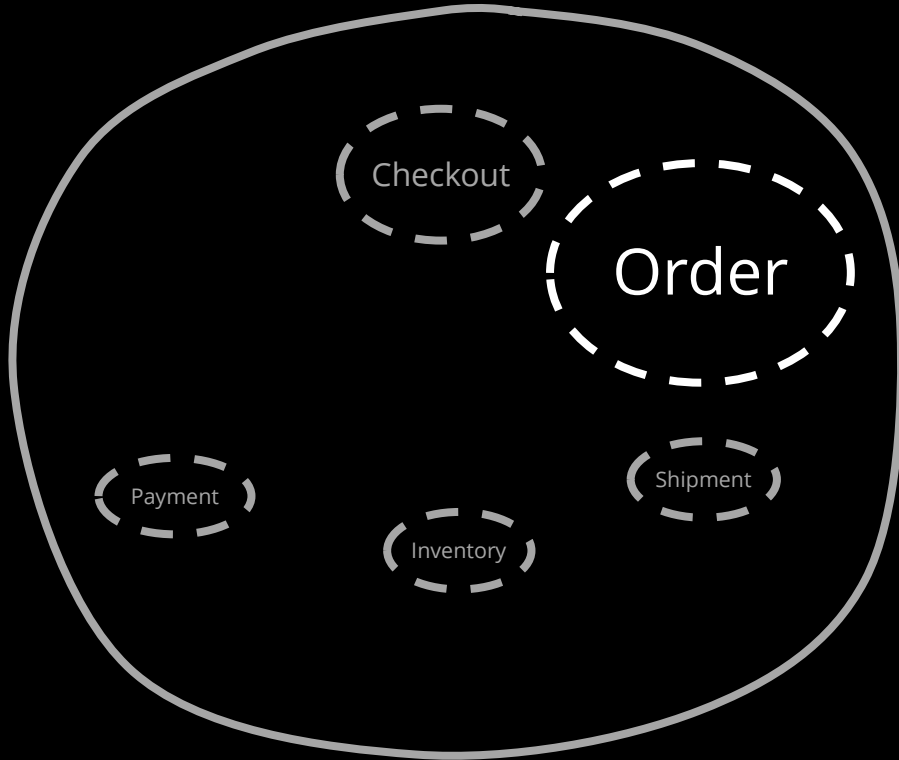
Danger of god services?



Sam Newmann

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

Danger of god services?



Sam Newmann

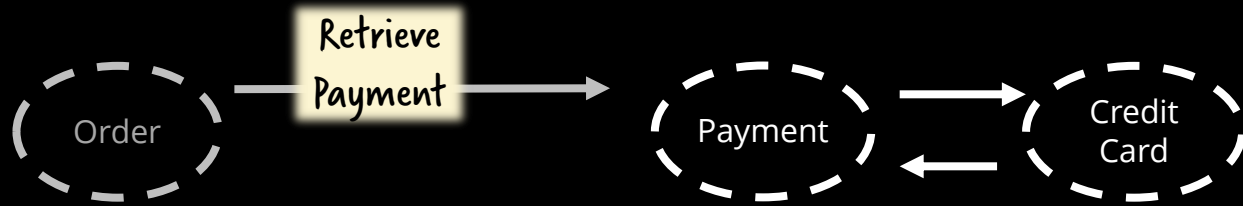
A few
smart god services
tell
anemic CRUD services
what to do

A god service is only created
by bad API design!

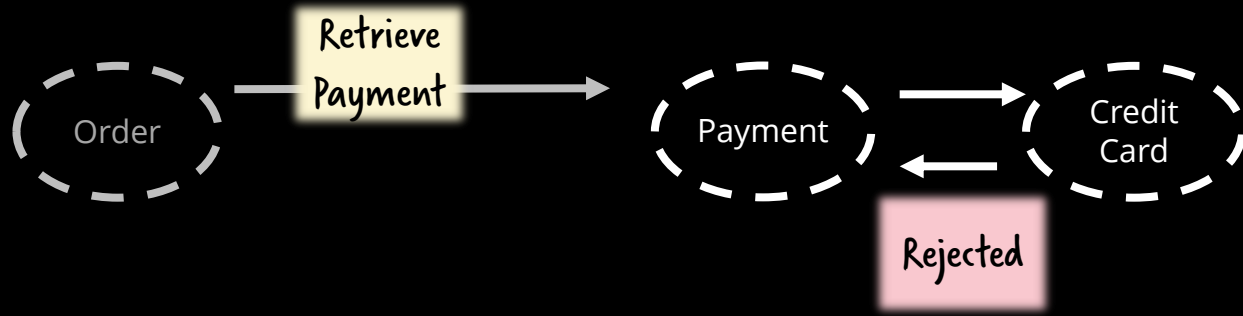
Example



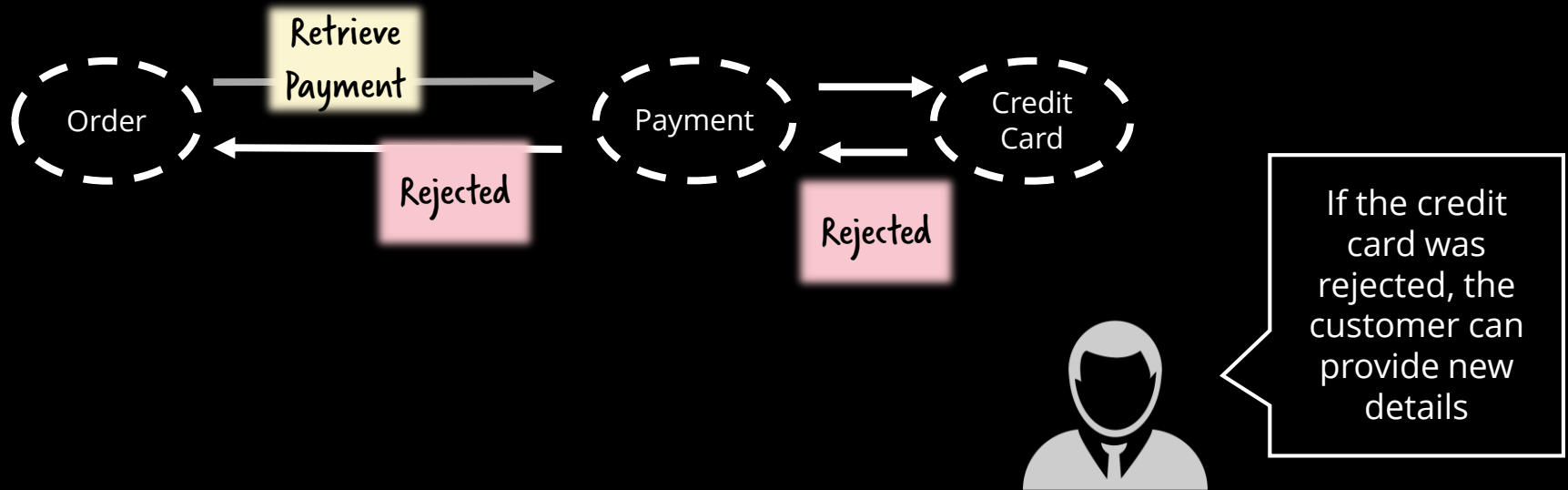
Example



Example

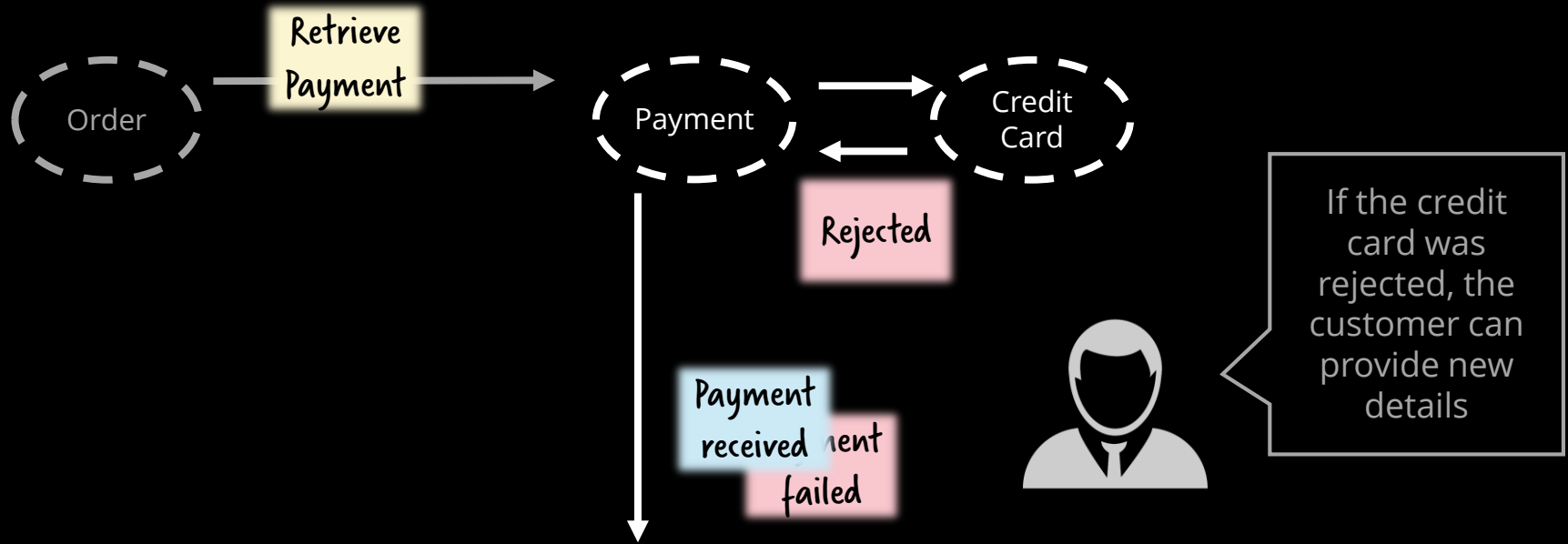


Example

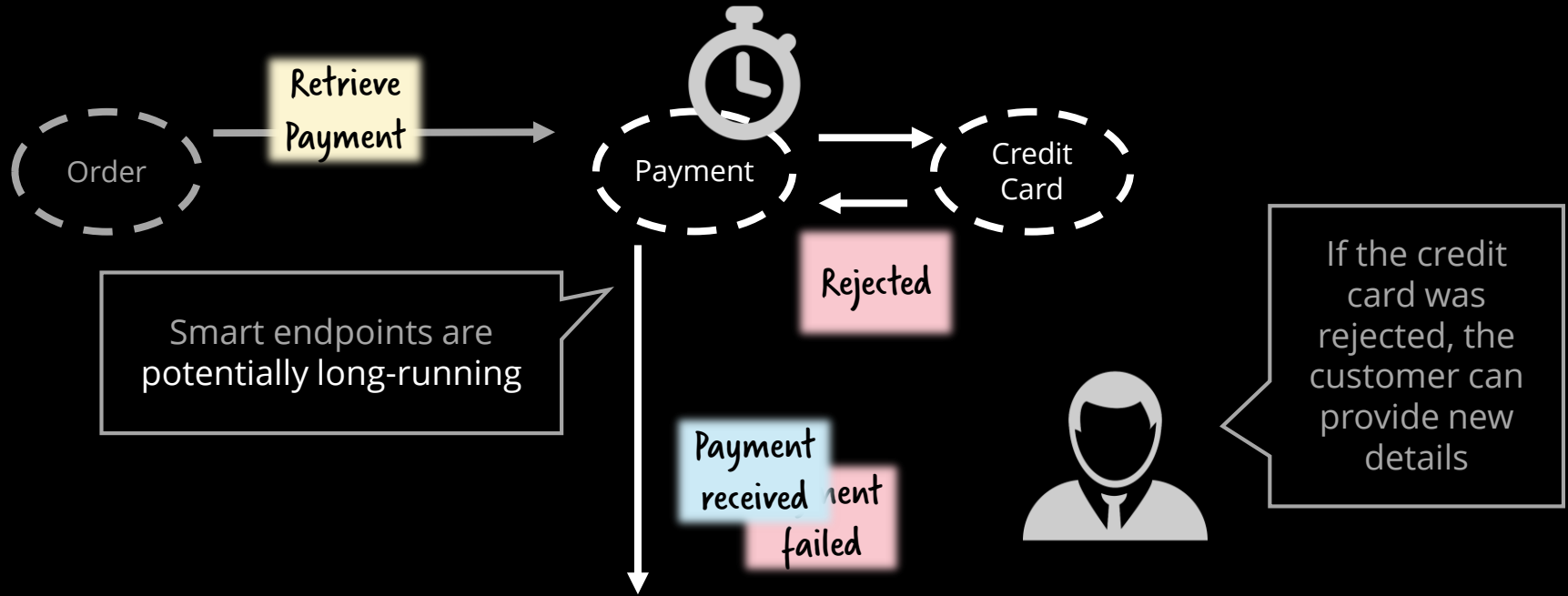


Client of **dumb endpoints** easily become a god services.

Who is responsible to deal with problems?



Long-running execution

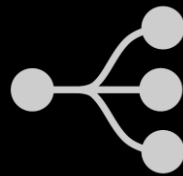


Clients of **smart endpoints** remains lean.

Handling State



Persist thing
(Entity, Document, 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!

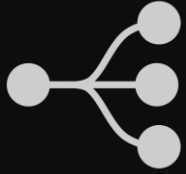


Low-code is great!
(You can get rid
of your developers!)



Death by properties panel

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

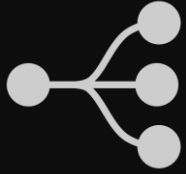


Workflow engines,
state machines



AWS Step
Functions

It is
relevant
in modern
architectures



Workflow engines,
state machines



Silicon valley
has recognized



Traditionally, some of these processes had been orchestrated in an ad-hoc manner using a combination of pub/sub, making direct REST calls, and using a database to manage the state. However, as the number of microservices grow and the complexity of the processes increases, getting visibility into these distributed workflows becomes difficult without a central orchestrator.



Netflix Technology Blog [Follow](#)

Learn more about how Netflix designs, builds, and operates our systems and engineering organizations

Dec 12, 2016 · 7 min read

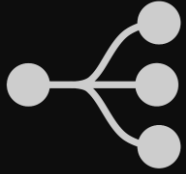
Netflix Conductor: A microservices orchestrator

The Netflix Content Platform Engineering team runs a number of business processes which are driven by asynchronous orchestration of tasks executing on microservices. Some of these are long running processes spanning several days. These processes play a critical role in getting titles ready for streaming to our viewers across the globe.

A few examples of these processes are:

- Studio partner integration for content ingestion
- [IMF](#) based content ingestion from our partners
- Process of setting up new titles within Netflix
- Content ingestion, encoding, and deployment to CDN

Traditionally, some of these processes had been orchestrated in an ad-hoc manner using a combination of pub/sub, making direct REST calls, and using a database to manage the state. However, as the number of microservices grow and the complexity of the processes increases, getting visibility into these distributed workflows becomes difficult without a central orchestrator.



Workflow engines,
state machines

UBER

CADENCE

There are
lightweight open source
options



AWS Step
Functions



camunda

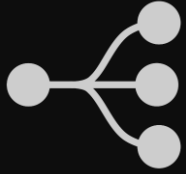


conductor

NETFLIX OSS



Activiti™



Workflow engines,
state machines

UBER

CADENCE

also at scale



zeebe

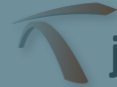
by Camunda



AWS Step
Functions



camunda



jBPM



conductor

NETFLIX

OSS



Activiti™

Submission

Medium



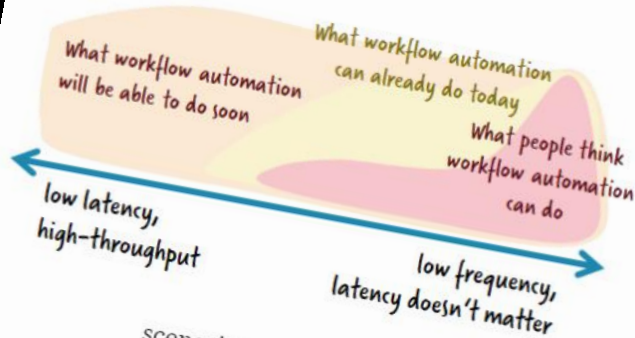
Bernd Rucker [Follow](#)
Draft · 16 min read

[Sign in](#)

[Get started](#)

How we built a highly scalable distributed state machine

Say hello to "big workflow"—part 2



In Zeebe.io—a highly scalable distributed workflow engine I described that Zeebe is a super performant, highly scalable and resilient workflow engine. I described that this allows to leverage workflow automation in a lot more use cases, also low lat...

scenarios. I revealed that Zeebe...
Kafka. I res...

```
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();

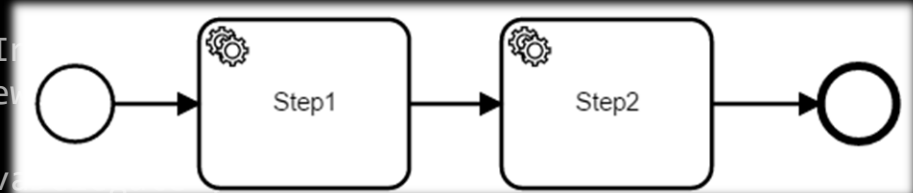
    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
        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

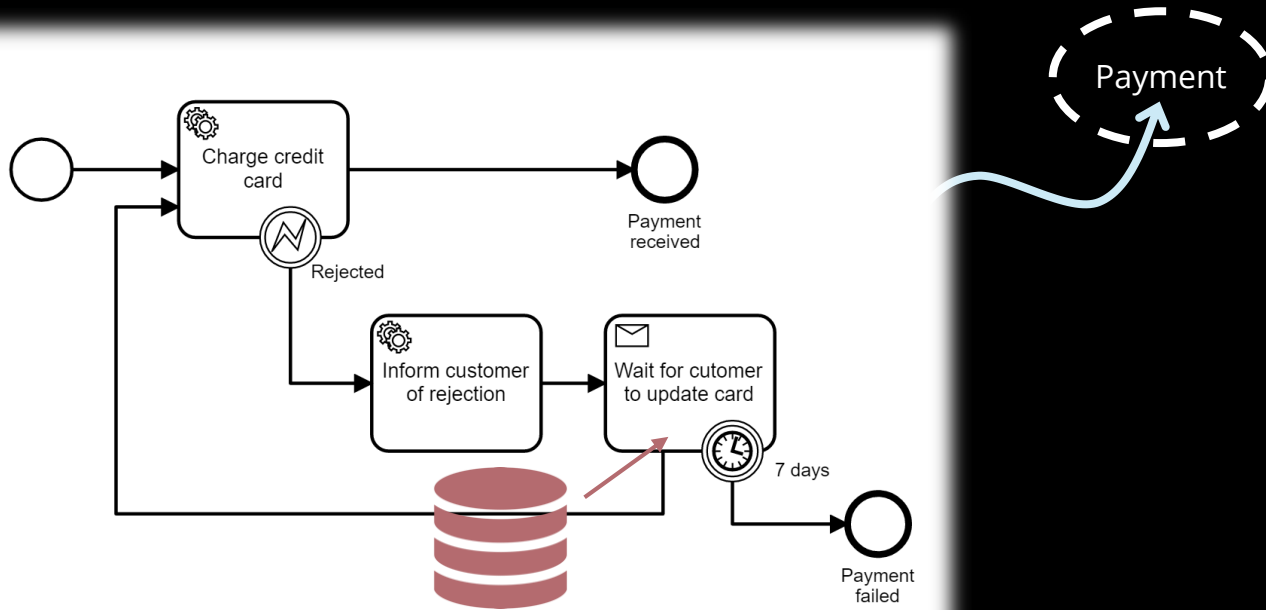
    engine.getRepositoryService().createDeployment() //
        .addModelInstance("flow.bpmn", Bpmn.createExecutableProcess("flow")
            .startEvent()
            .serviceTask("Step1").camundaClass(SysoutDelegate.class)
            .serviceTask("Step2").camundaClass(SysoutDelegate.class)
        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

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

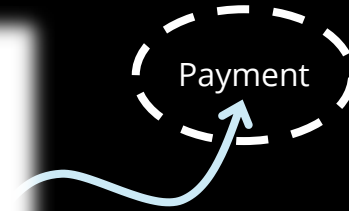
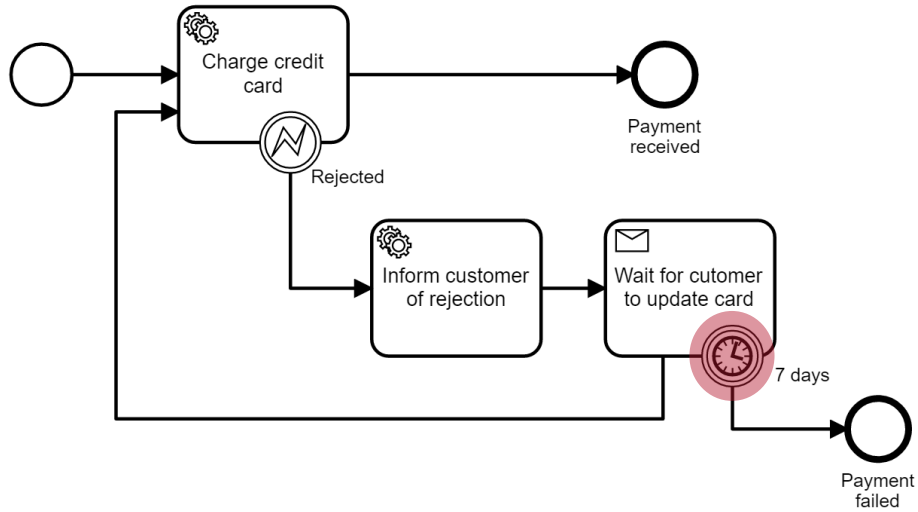
public class SysoutDelegate implements JavaDelegate {
    public void execute(DelegateExecution execution) throws Exception {
        System.out.println("Hello " + execution.getVariable("city"));
    }
}
```

Start
instances

Now you have a state machine!



Easy to handle time



Domain-Driven

DESIGN

Tackling Complexity in the Heart of Software



Eric Evans

Foreword by Martin Fowler

EVENT STORMING

Alberto Brandolini



Atomic vs. composite command execution

Place
order

order
placed

Atomic, trans-
actional execution

Typically we see a „command“ as the intent to change a write model...

Place
order

order
fulfilled

Composite, long-
running execution

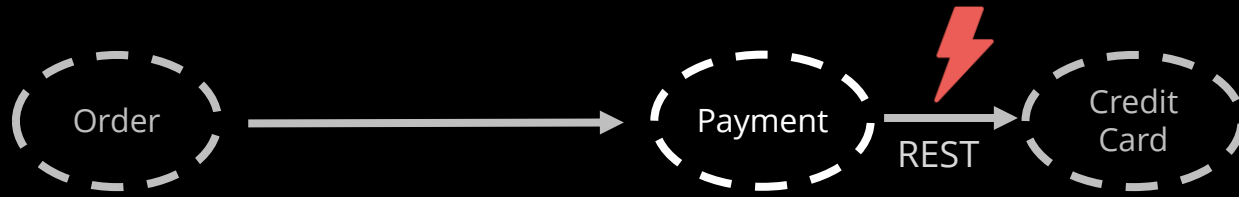
... but the customer's or service clients intent is often targeted at a more valuable business result, which needs many steps to be achieved.

martin.schimak@plexiti.com

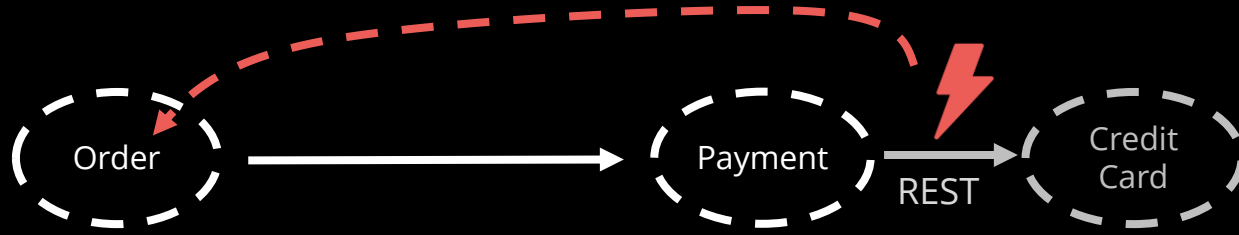
@martinschimak



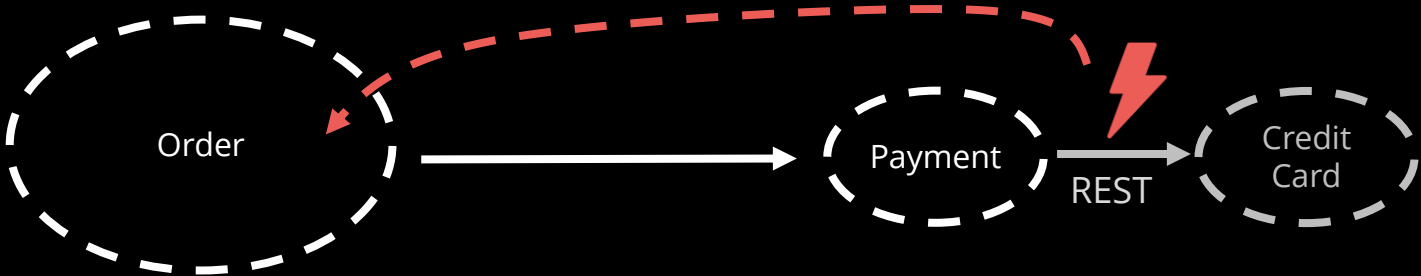
Synchronous communication



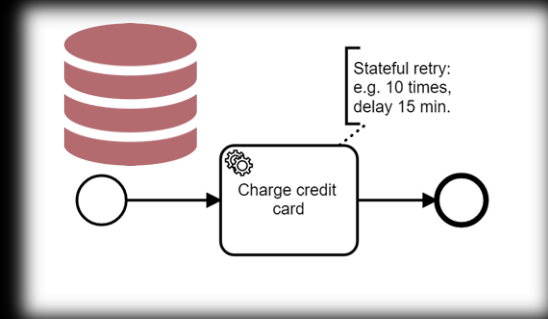
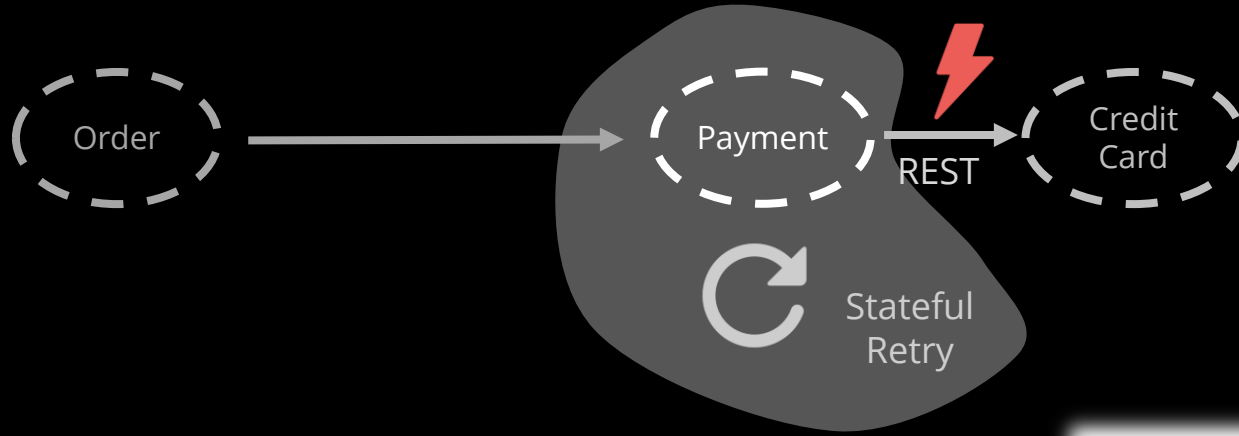
Synchronous communication



Synchronous communication



Synchronous communication

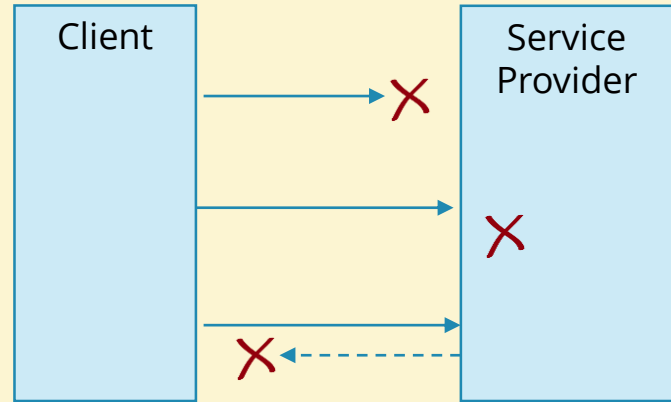


Distributed systems

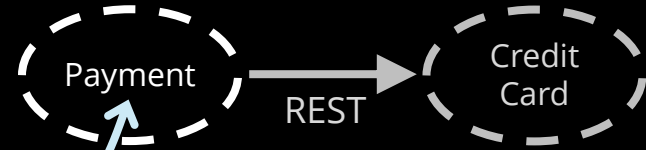
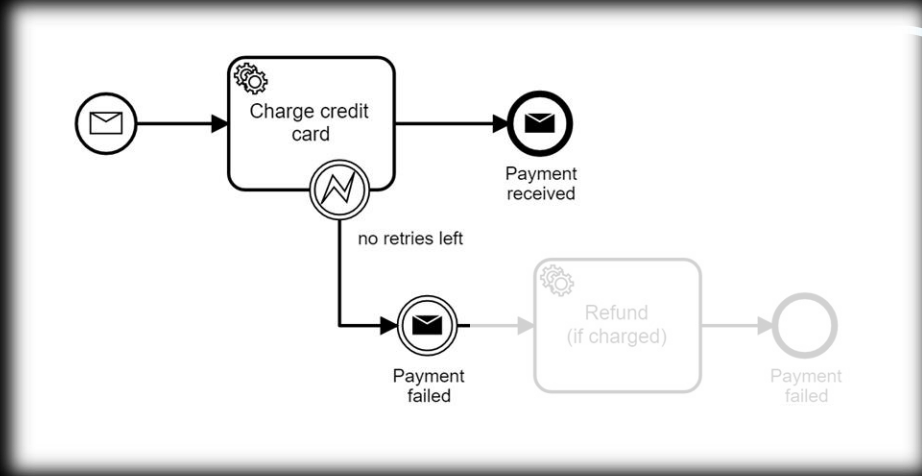


It is impossible to differentiate certain failure scenarios.

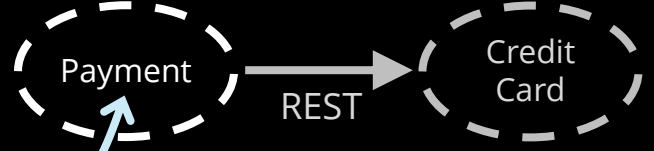
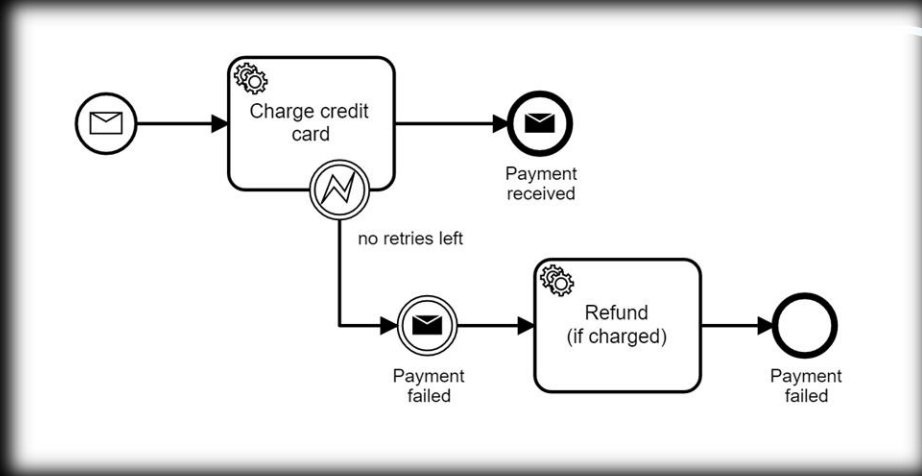
Independant of communication style!



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.

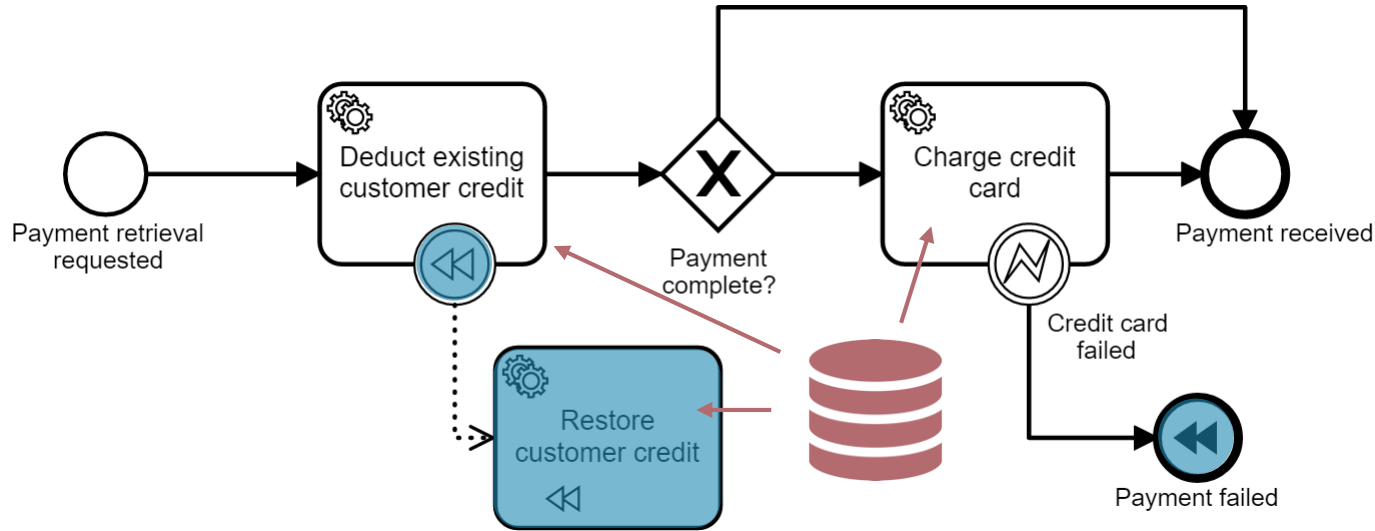
ABSTRACT

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

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

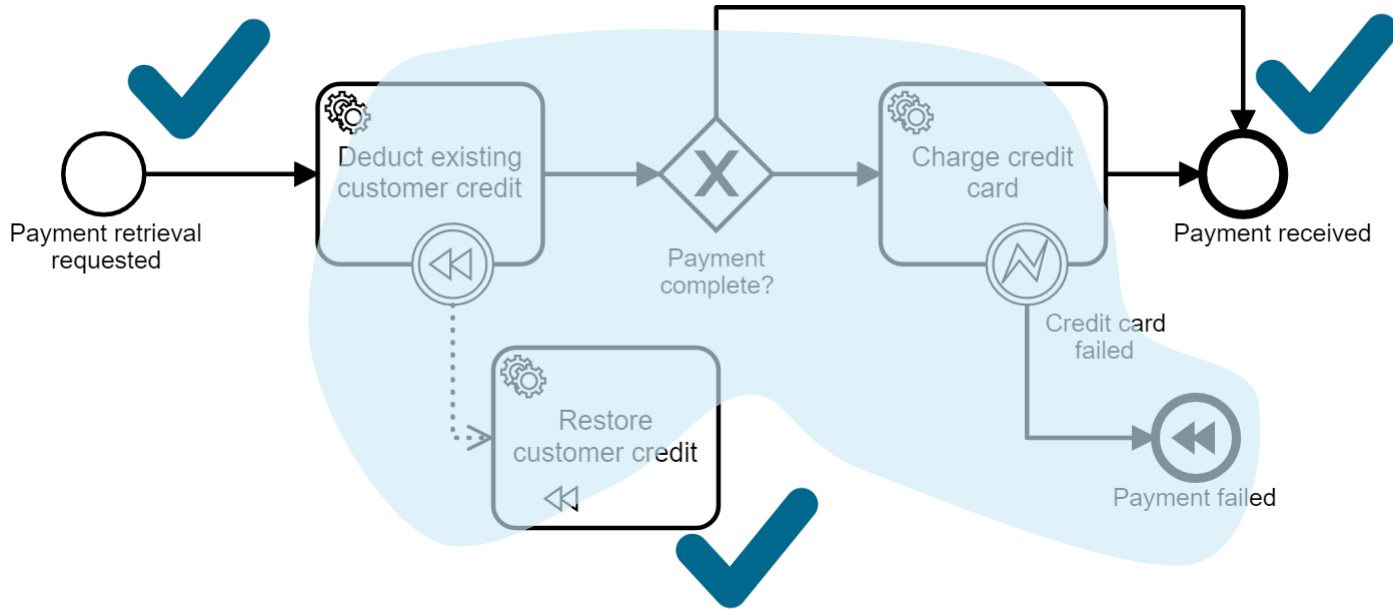
Distributed transactions using compensation *

* aka Saga pattern



Compensation

Relaxed consistency



Temporarily
inconsistent
state

But eventually
consistent

No Isolation
(as in ACID)

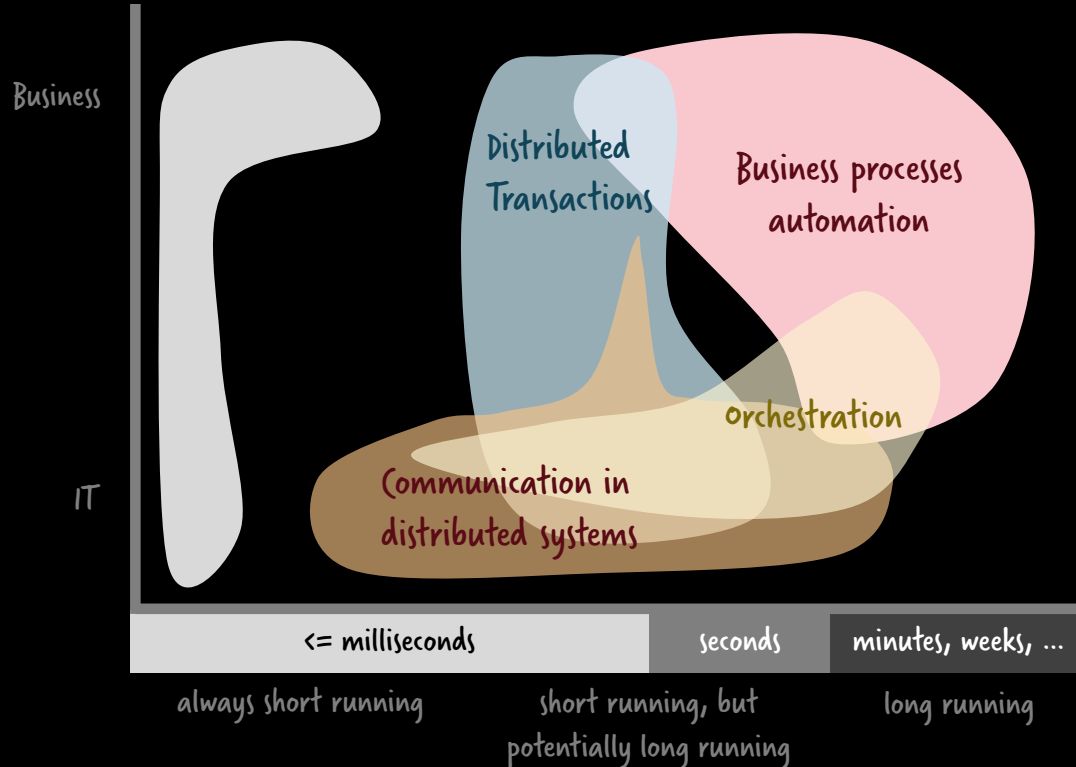
CONTRIBUTED / TOP STORIES / GLOBAL

5 Workflow Automation Use Cases You Might Not Have Considered

9 Apr 2018 3:00am, by Bernd Rucker



Use cases for workflow automation



Biz Dev ops

improve
communication

improve
communication

Biz Dev ops

improve
communication

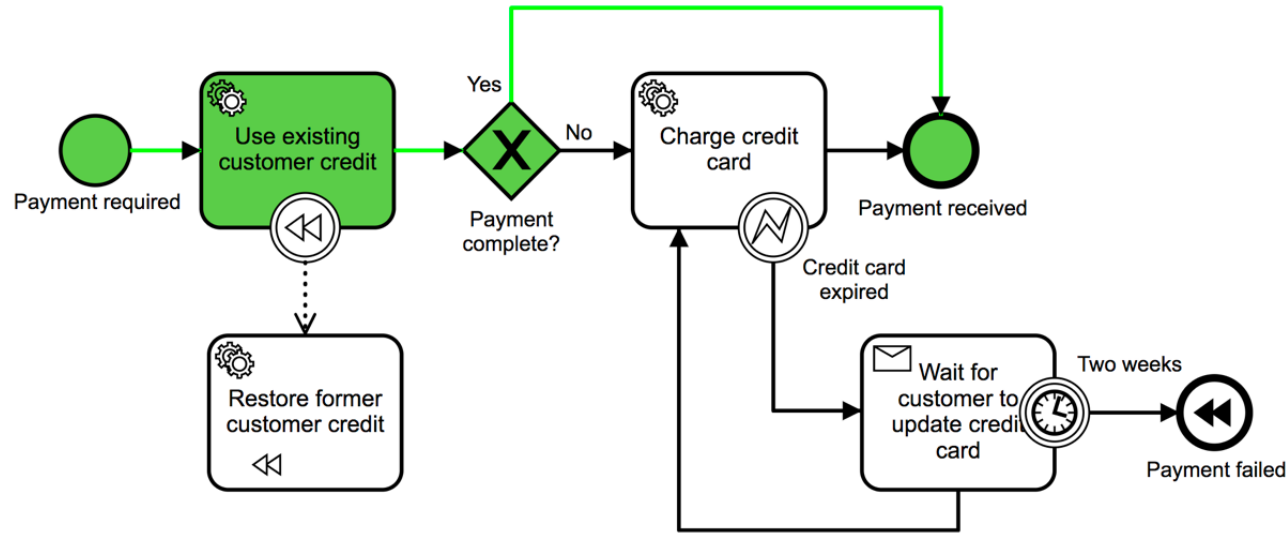
improve
communication

Leverage
state machine &
workflow engine

Living
documentation

Visibility in
testing

Visual HTML reports for test cases



Biz Dev ops

improve
communication

improve
communication

Understand and discuss
business processes

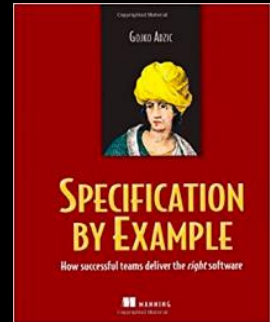
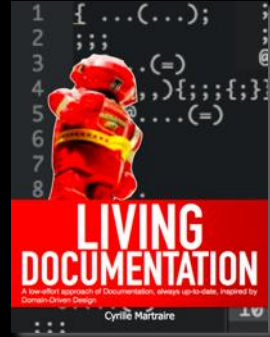
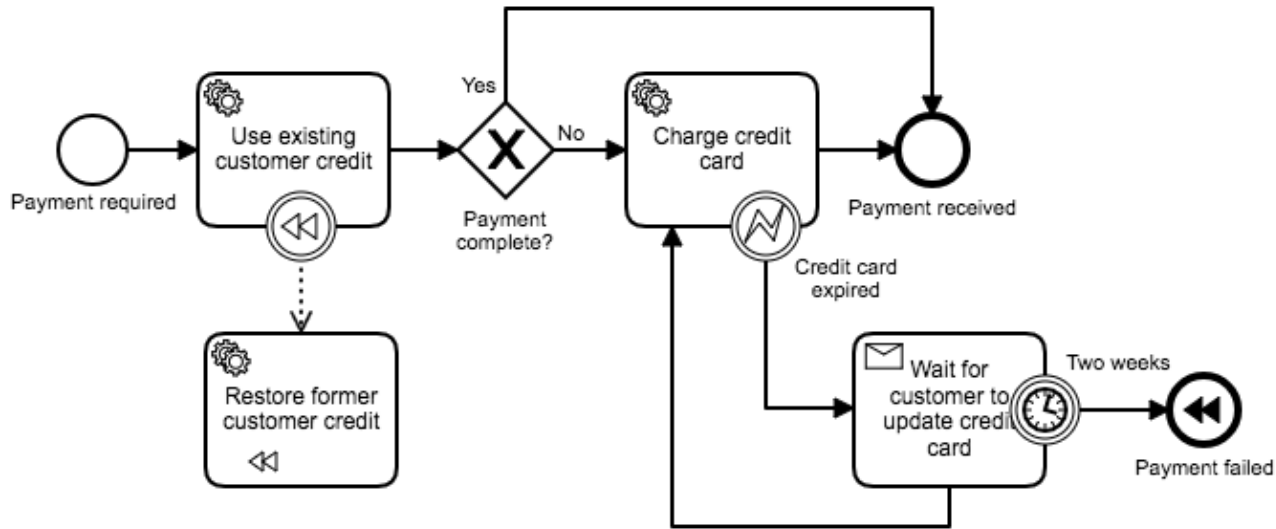
Leverage
state machine &
workflow engine

Evaluate optimizations
in-sync with
implementation

Living
documentation

Visibility in
testing

Living documentation for long-running behaviour



Biz Dev ops

improve
communication

improve
communication

Understand and discuss
business processes

Leverage
state machine &
workflow engine

Evaluate optimizations
in-sync with
implementation

Living
documentation

Visibility in
testing

Biz Dev ops

improve
communication

improve
communication

Understand and discuss
business processes

Leverage
state machine &
workflow engine

operate with visibility
and context

Evaluate optimizations
in-sync with
implementation

Living
documentation

Visibility in
testing

Proper operations

Visibility + Context

Camunda Cockpit Processes Decisions Cases Human Tasks More ▾

Dashboard > Processes > paymentV5 : Runtime | History

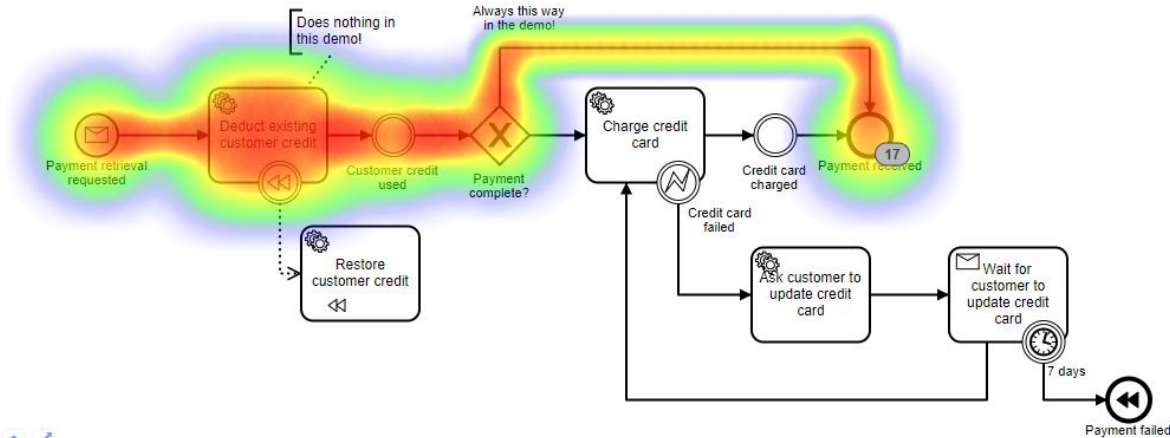
Definition Version: 2
Version Tag: null
Definition ID: paymentV5:2:45aea93a-1ad9-11e8-8...
Definition Key: paymentV5
Definition Name: null
History Time To Live: null
Tenant ID: null

BPMN Diagram:
Start: Payment retrieval requested (4.7k)
Task: Deduct existing customer credit
Decision: Payment complete?
Task: Charge credit card (2.3k, 317)
End: 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





Not even a full day

Metrics

Executed Activity Instances

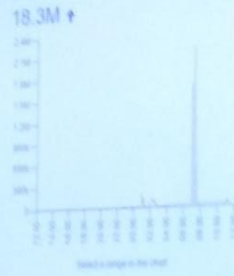
Started ● 20.7M ↑

Ended ● 20.7M ↑



Evaluated Decision Instances

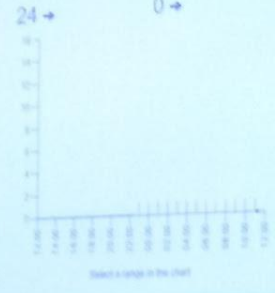
Evaluated ● 18.3M ↑



Executed Jobs

Successful ● 24 →

Failed ● 0 →



Deployed

Process Definitions

190

Decision Definitions

42

Case Definitions

0

Deployments

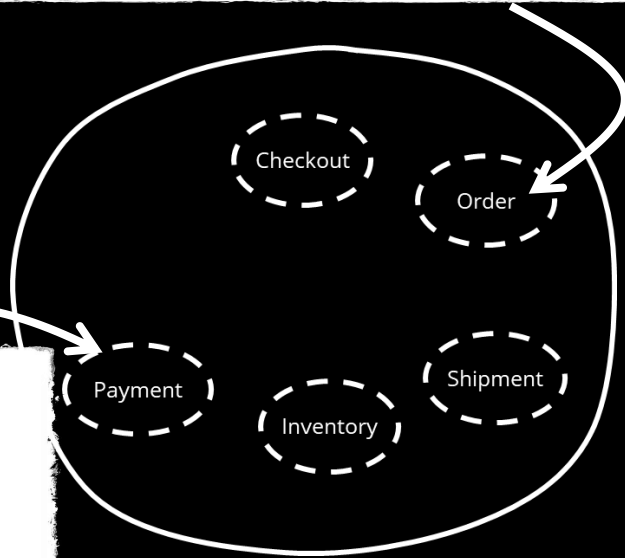
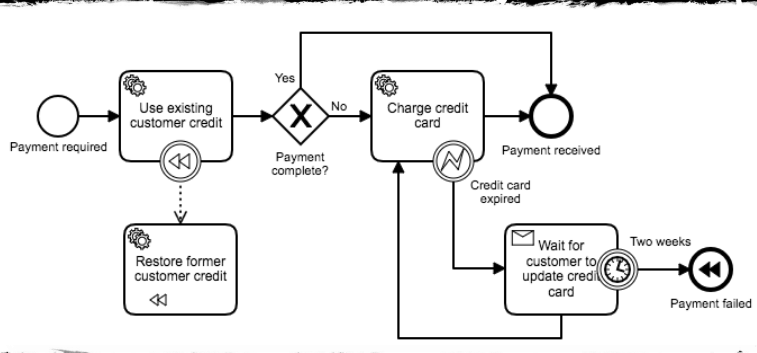
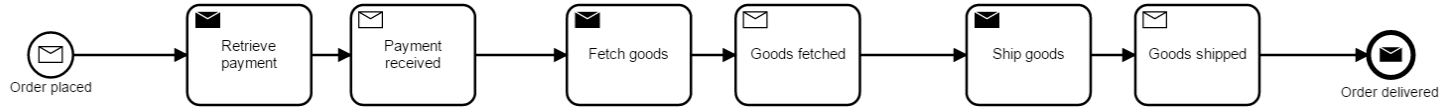
215

”

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



Manifold architecture options



berndruecker 



Bernd Rucker
Dec 19, 2017 · 15 min read

Edit    

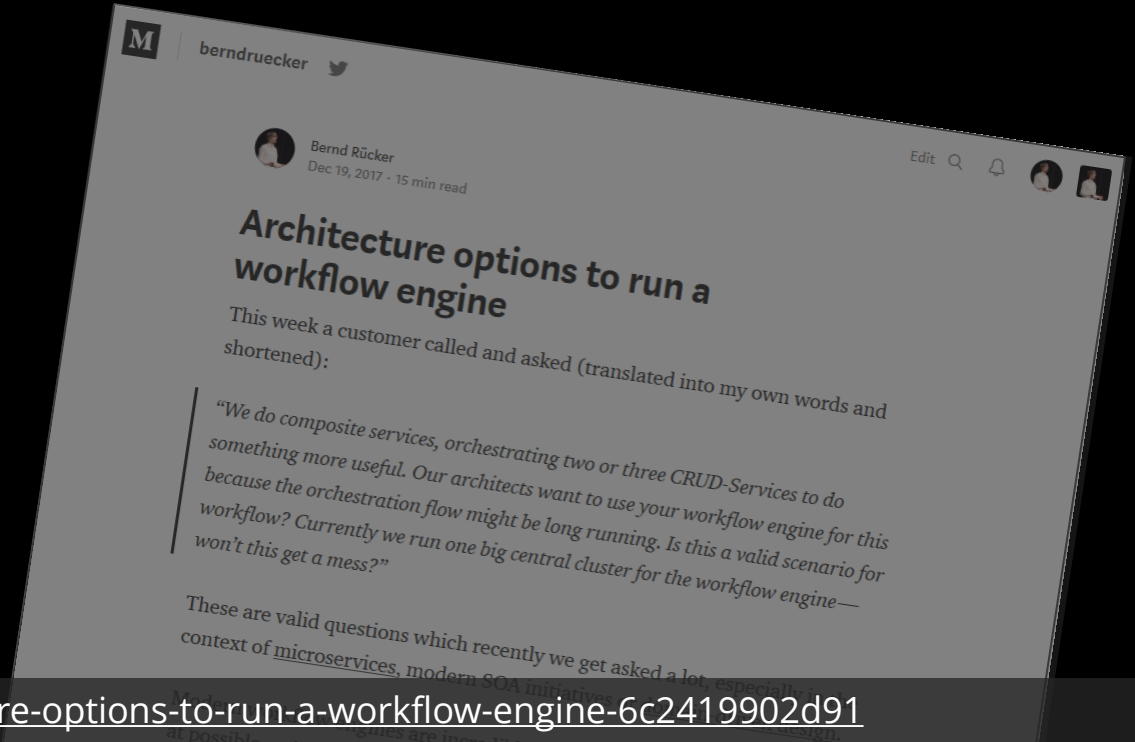
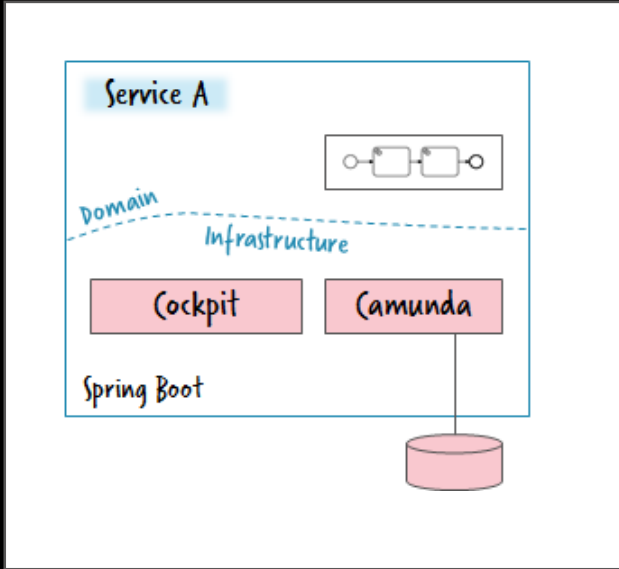
Architecture options to run a workflow engine

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

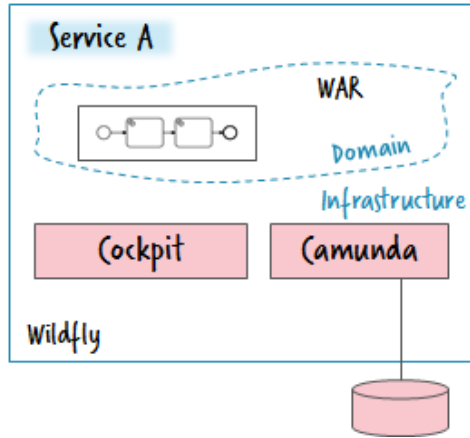
“We do composite services, orchestrating two or three CRUD-Services to do something more useful. Our architects want to use your workflow engine for this because the orchestration flow might be long running. Is this a valid scenario for workflow? Currently we run one big central cluster for the workflow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



berndruecker



Bernd Rucker
Dec 19, 2017 · 15 min read

Edit

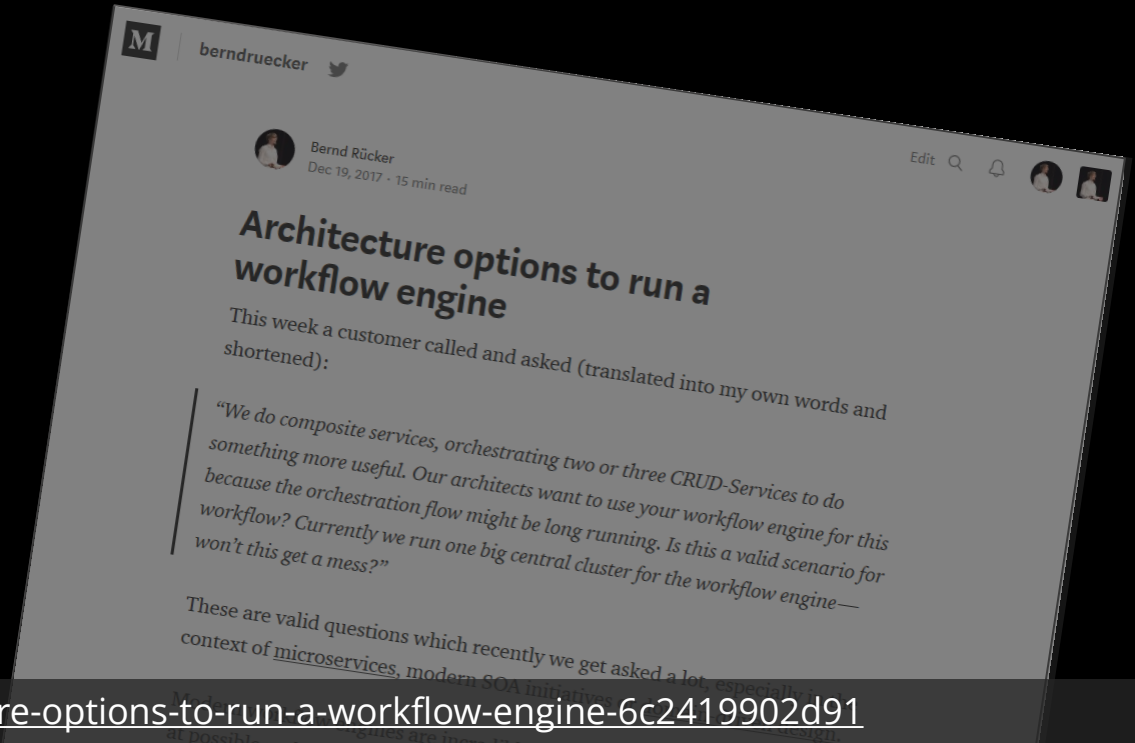
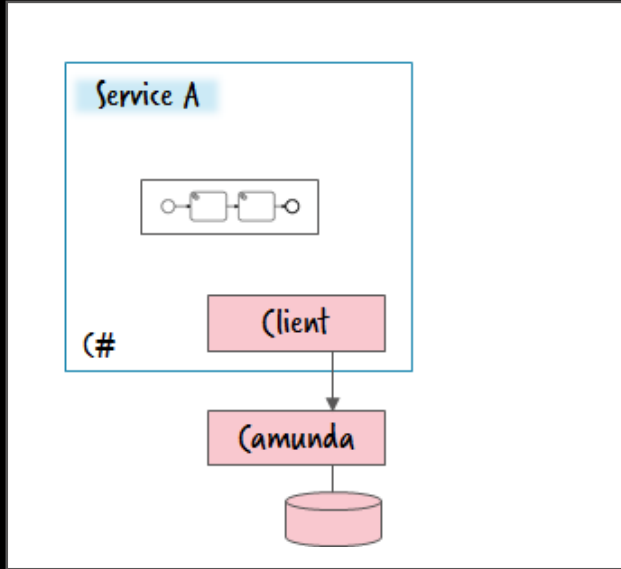
Architecture options to run a workflow engine

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

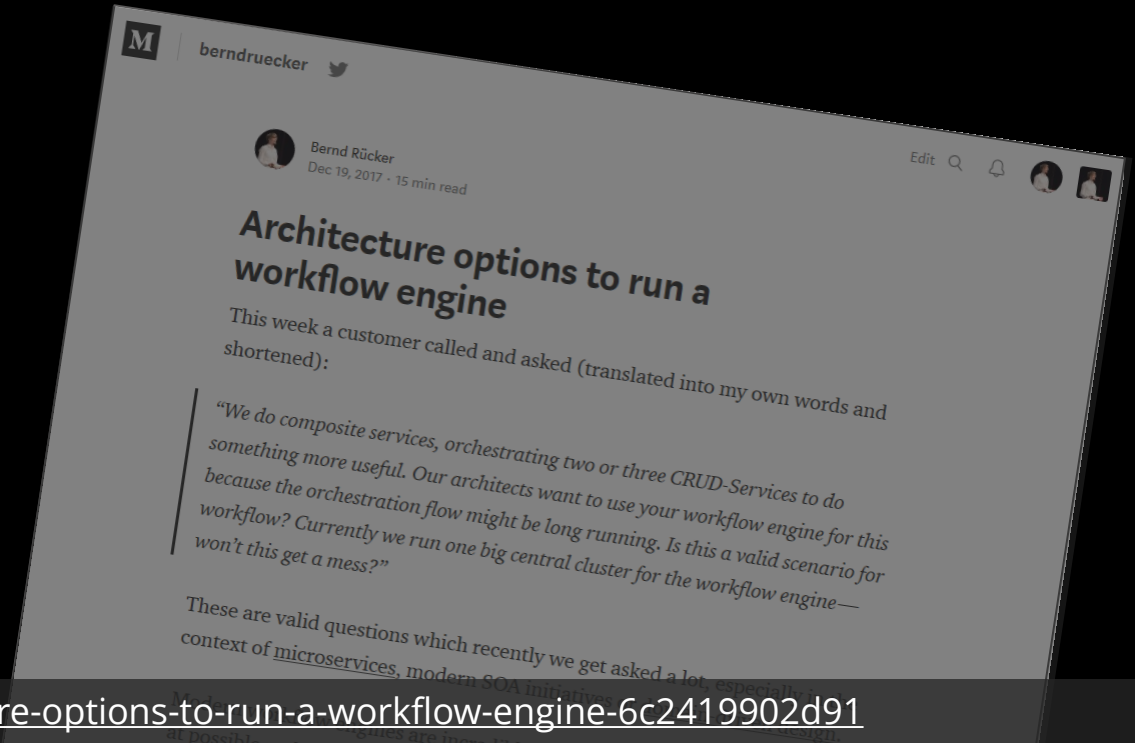
“We do composite services, orchestrating two or three CRUD-Services to do something more useful. Our architects want to use your workflow engine for this because the orchestration flow might be long running. Is this a valid scenario for workflow? Currently we run one big central cluster for the workflow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



Lightweight workflow engines are
great* – don't DIY

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

Reality check

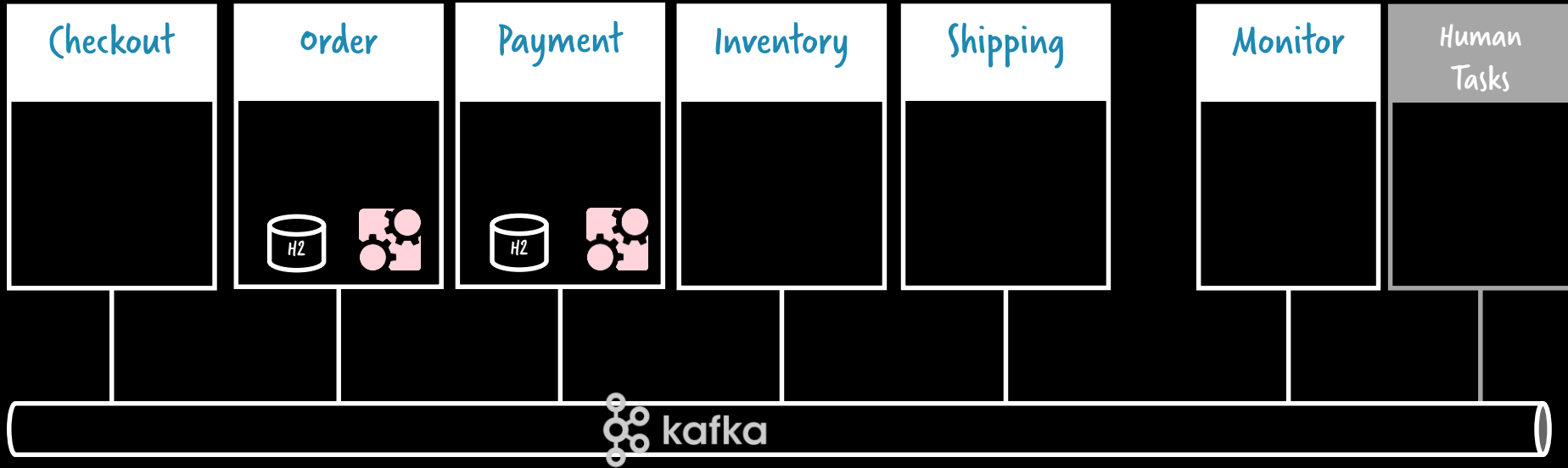
FULLFILLMENT PROCESS



Sales-order & order-Fulfillment
via Camunda
for every order worldwide
(Q2 2017: 22,2 Mio)


THE SHU
TECH INNOVAT

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!



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

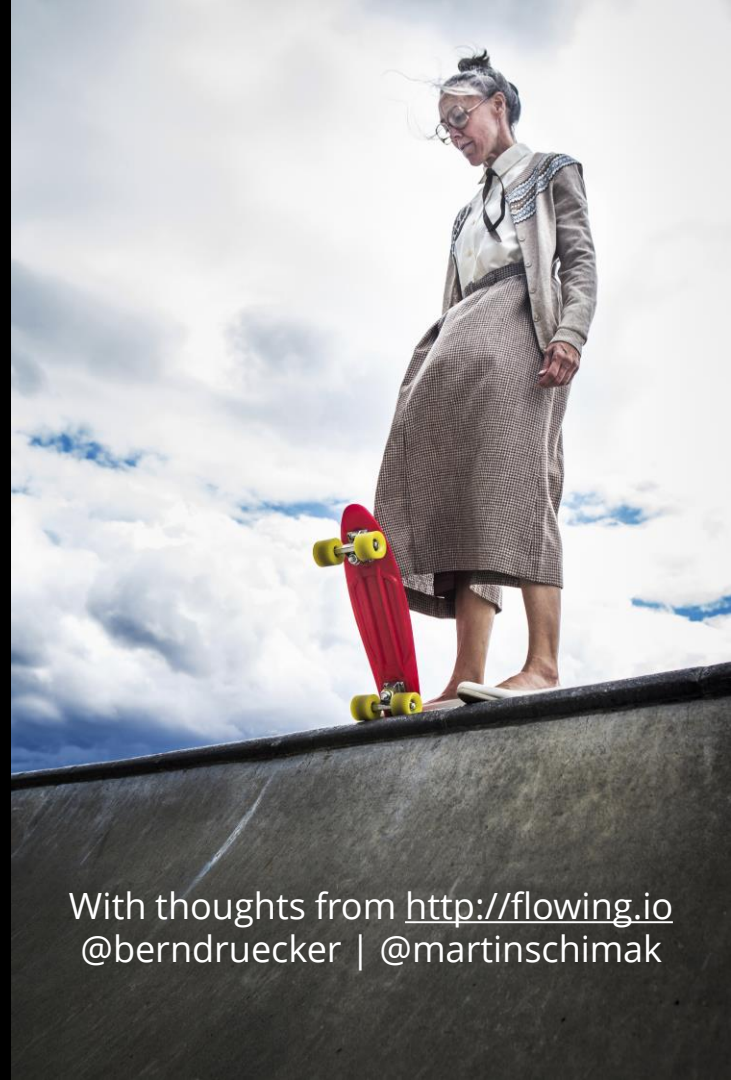
<https://www.infoworld.com/article/3254777/application-development/3-common-pitfalls-of-microservices-integration-and-how-to-avoid-them.html>

InfoQ
neue

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

THE NEW STACK

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



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