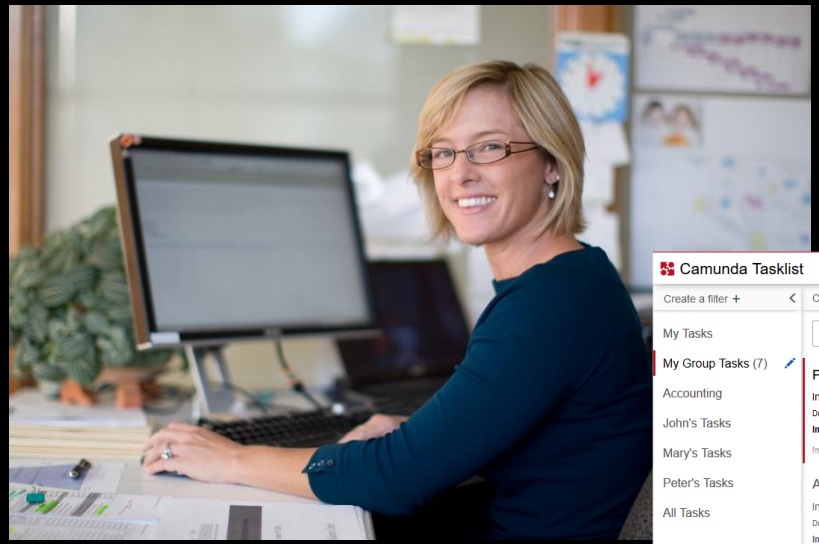
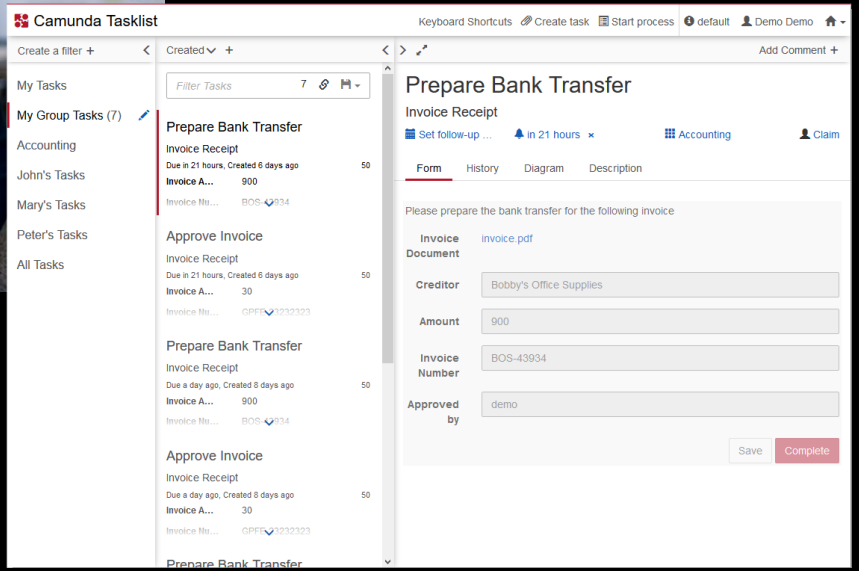
Workflow Automation Reinvented

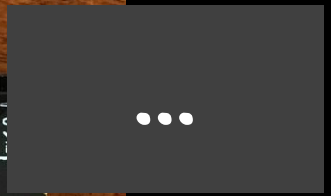
@berndruecker

What people think, when I say „Workflow Automation"...





What I think, when I say „Workflow Automation“



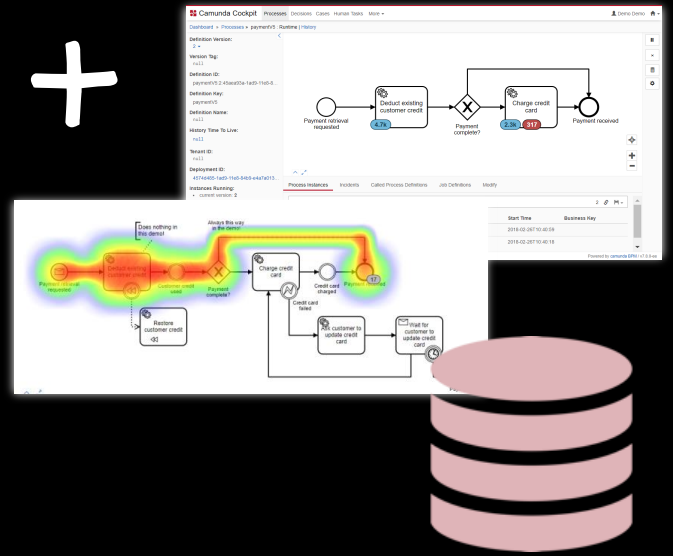
What people think, when I say „BPM“...

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

Death by properties panel

What I think, when I say „BPM“

+

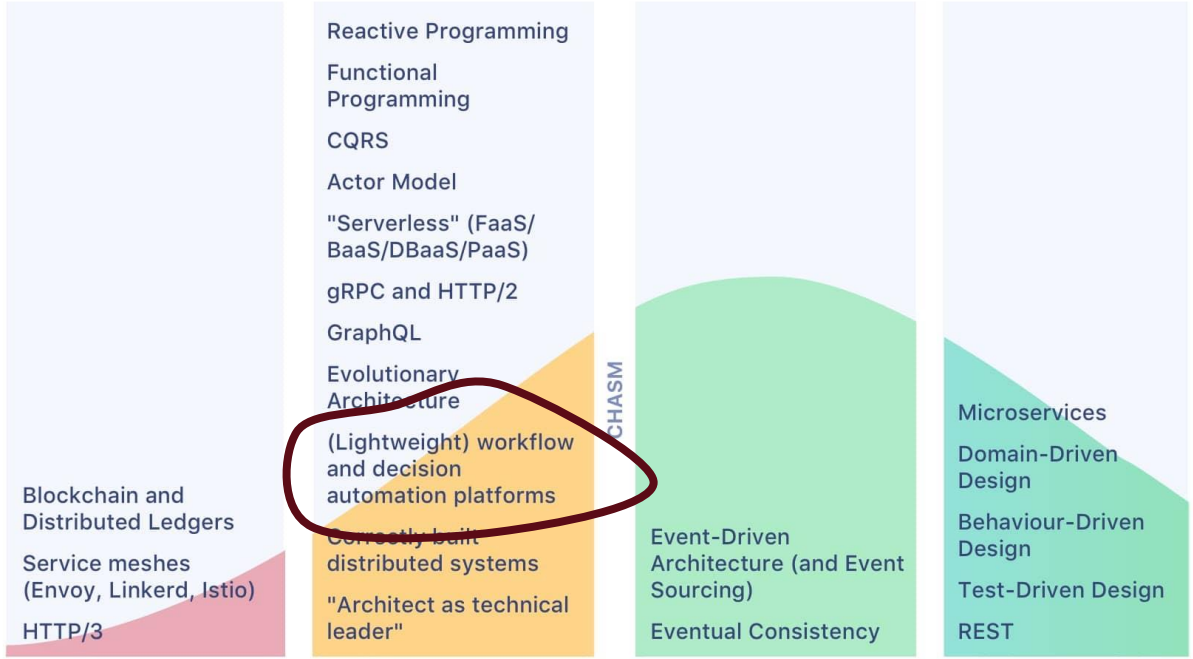


Software Development

Architecture and Design 2019 Q1 Graph

http://infoq.link/architecture-trends-2019

InfoQ



Why workflow automation?

THENEWSTACK Ebooks Podcasts Events Newsletter

Architecture Development Operations

MICROSERVICES

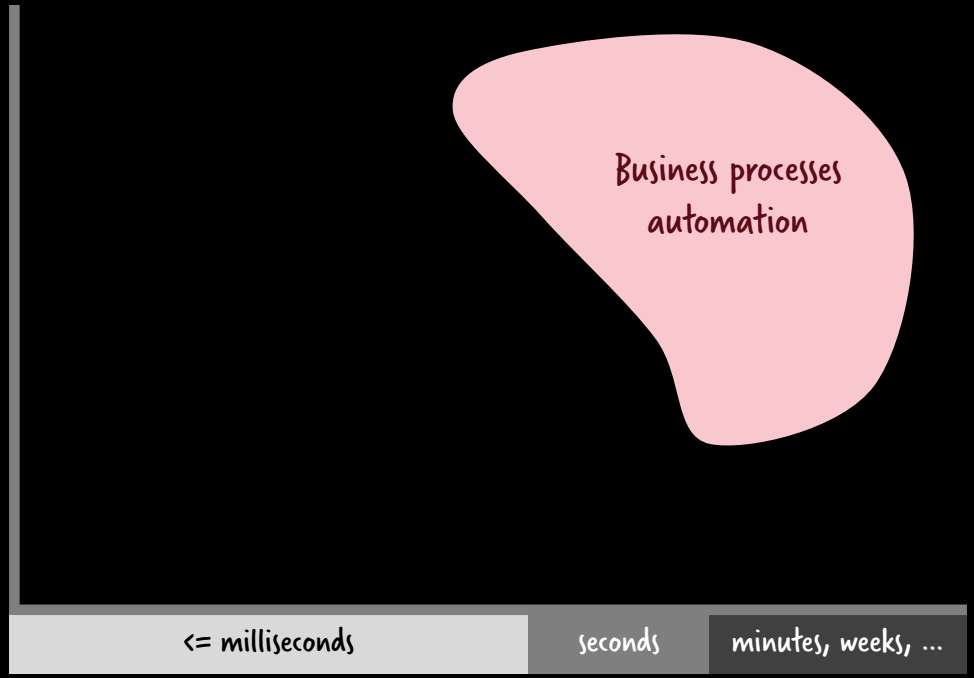
5 Workflow Automation Use Cases You Might Not Have Considered

9 Apr 2018 3:00am, by Bernd Rücker

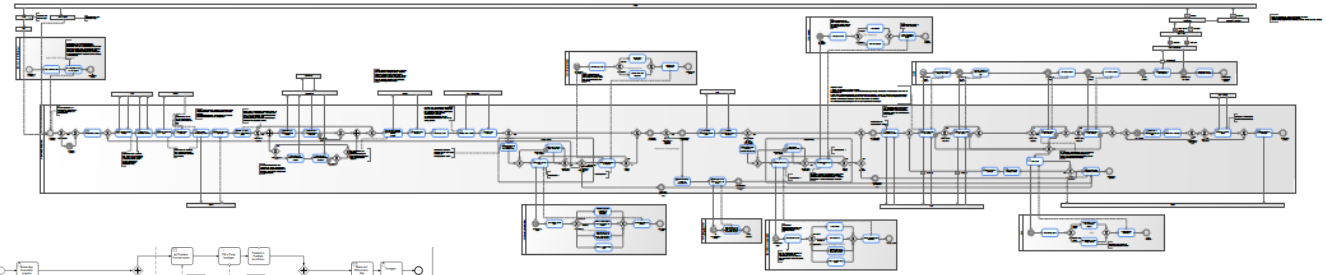
https://thenewstack.io/5-workflow-automation-use-cases-you-might-not-have-considered/

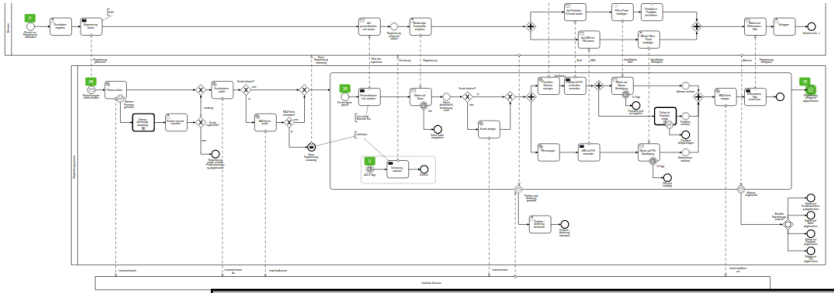
Use cases for workflow automation

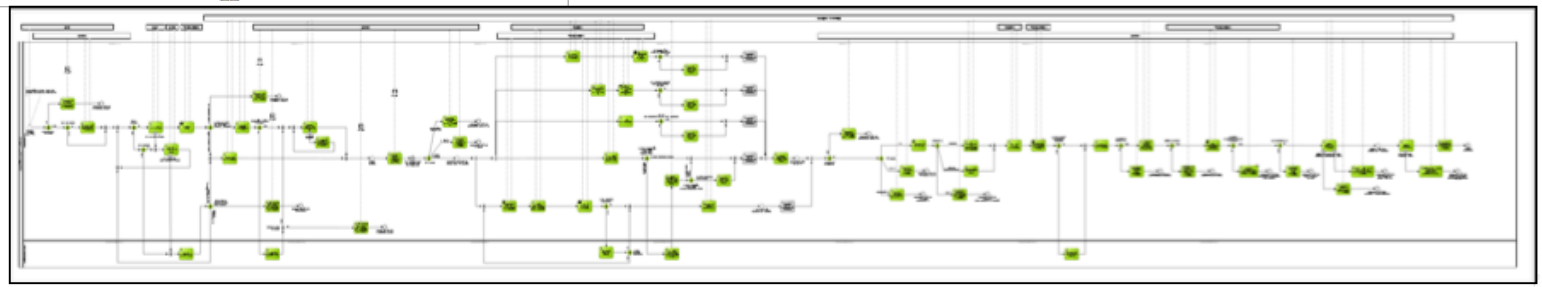
Use cases for workflow automation



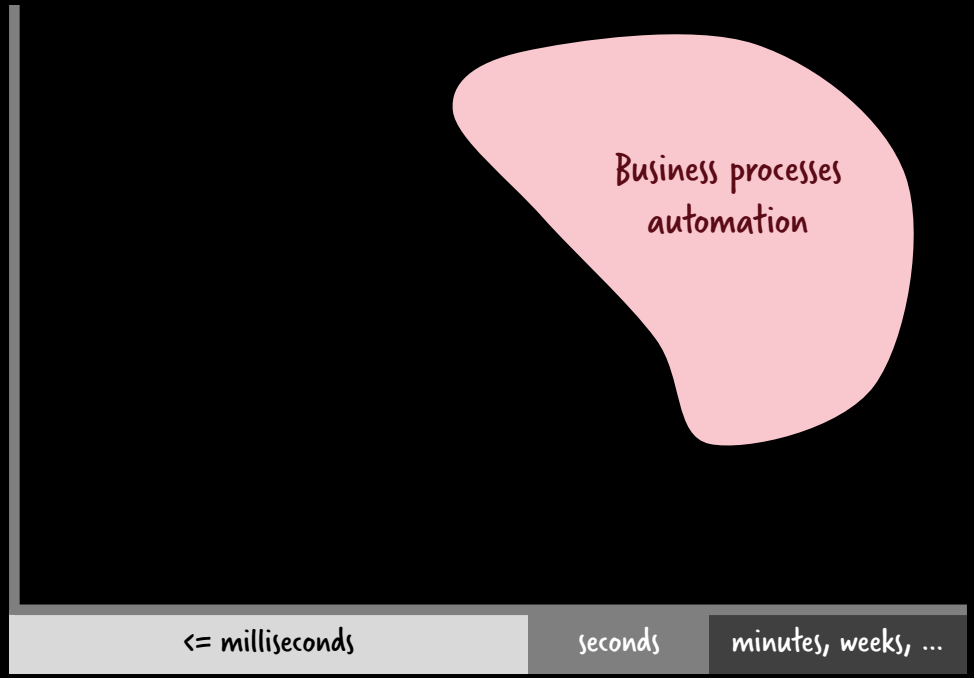
Real-life examples



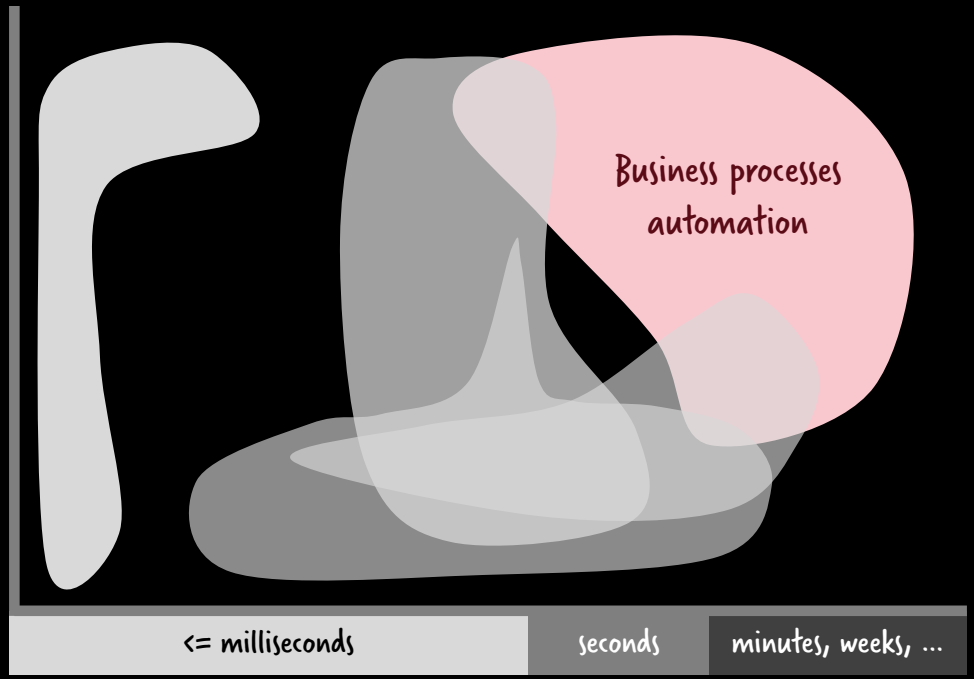


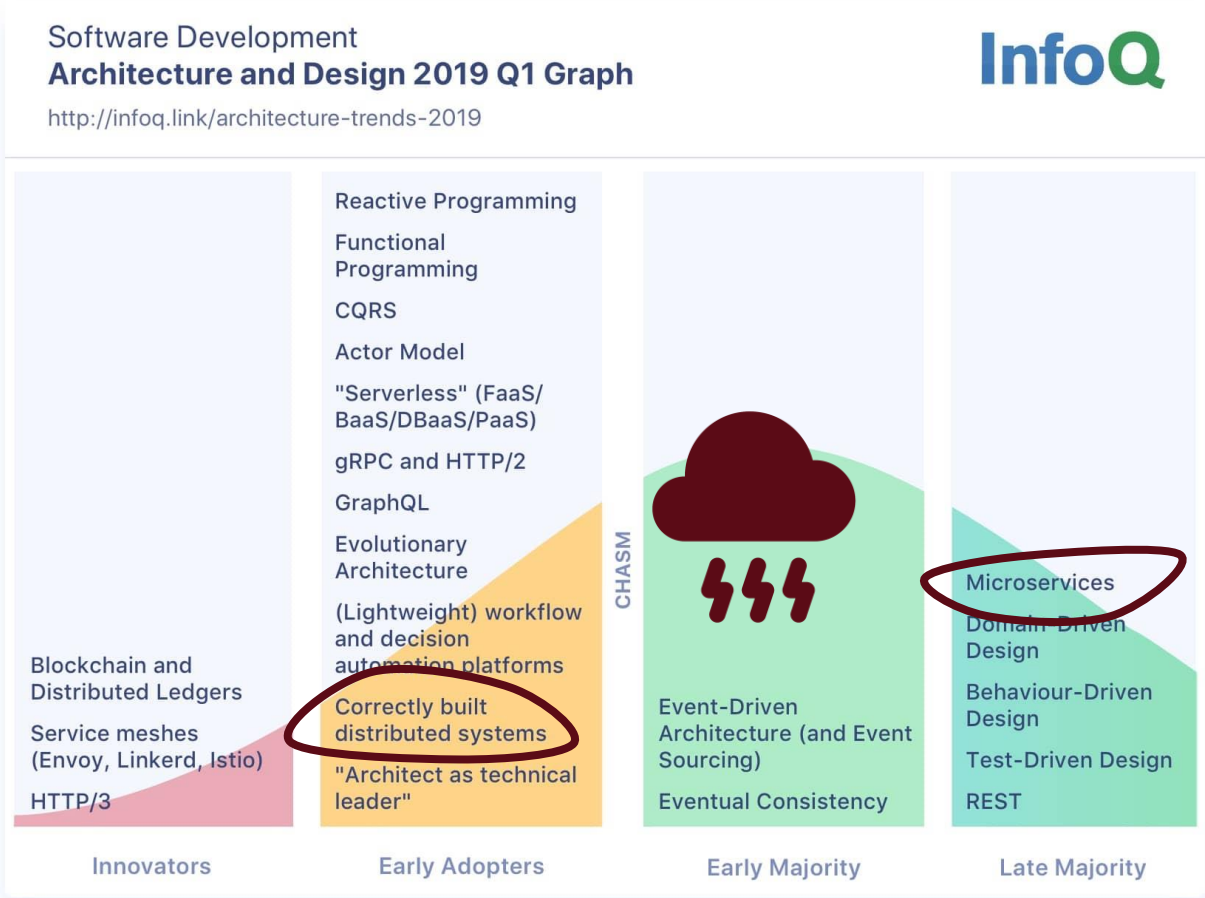


Use cases for workflow automation

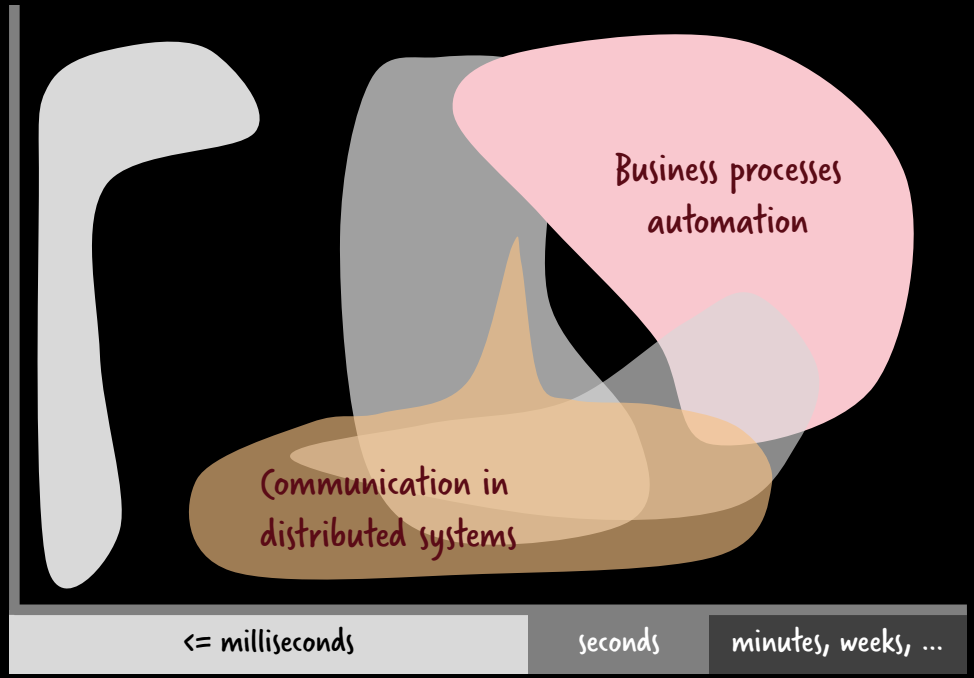


Use cases for workflow automation

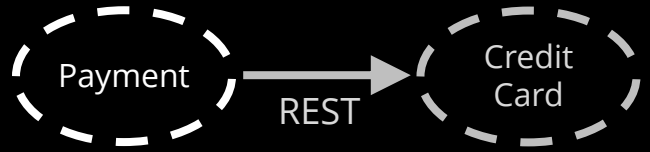



Software Development
Architecture and Design 2019 Q1 Graph
http://infoq.link/architecture-trends-2019
InfoQ
Blockchain and Distributed Ledgers
Service meshes (Envoy, Linkerd, Istio)
HTTP/3
Reactive Programming
Functional Programming
CQRS
Actor Model
"Serverless" (FaaS/ BaaS/DBaaS/PaaS)
gRPC and HTTP/2
GraphQL
Evolutionary Architecture
(Lightweight) workflow and decision automation platforms
Correctly built distributed systems
"Architect as technical leader"
CHASM
Event-Driven Architecture (and Event Sourcing)
Eventual Consistency
Microservices
Domain-Driven Design
Behaviour-Driven Design
Test-Driven Design
REST
Innovators
Early Adopters
Early Majority
Late Majority

Use cases for workflow automation

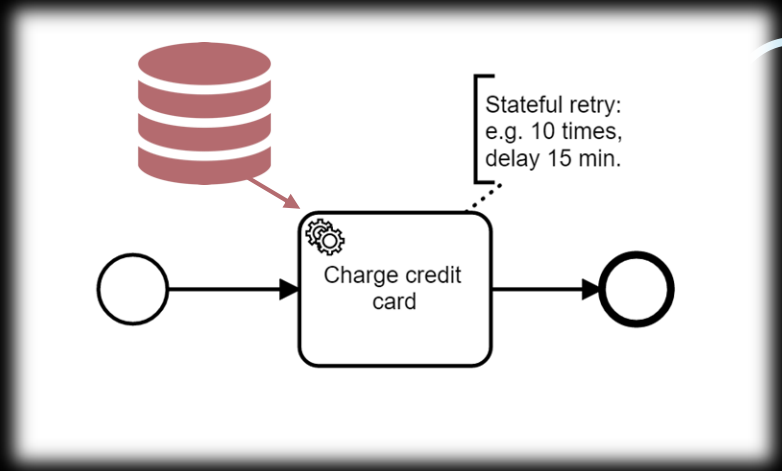


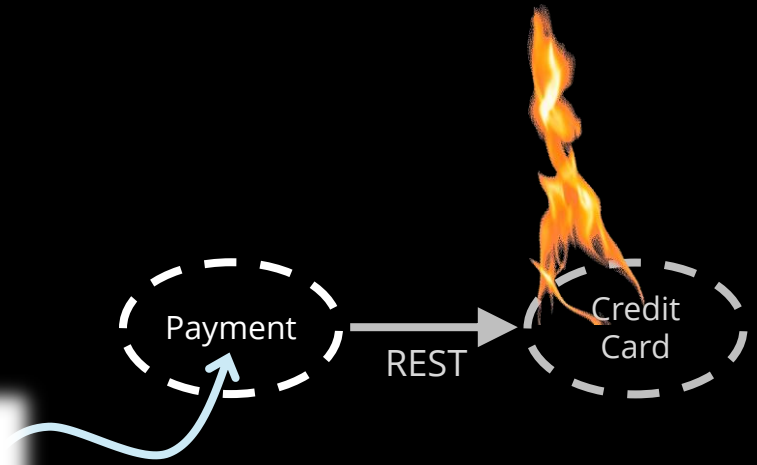
Ever called a REST API?



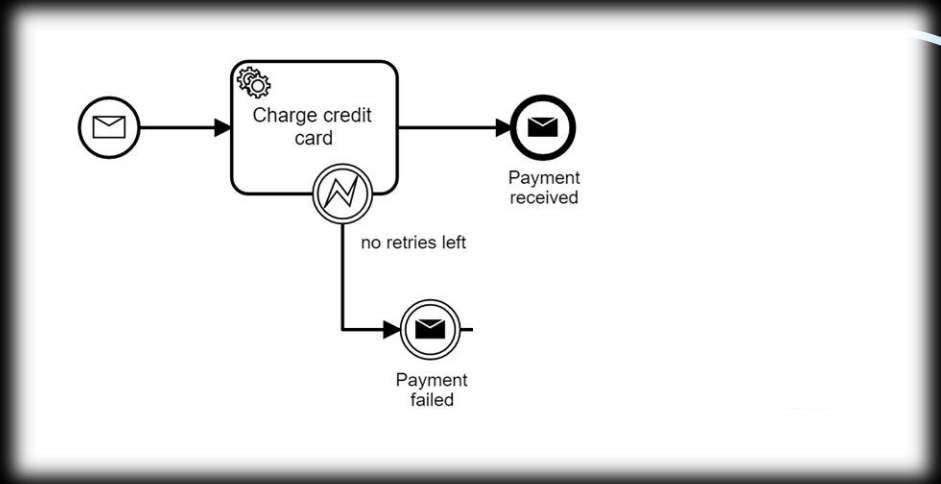
Distributed systems

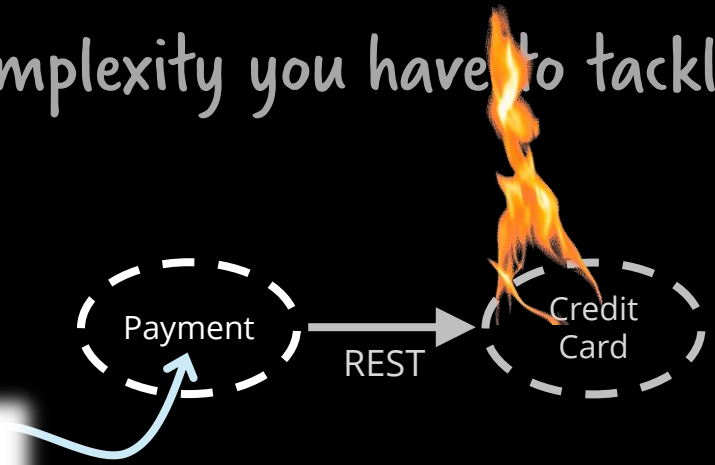
Stateful retry





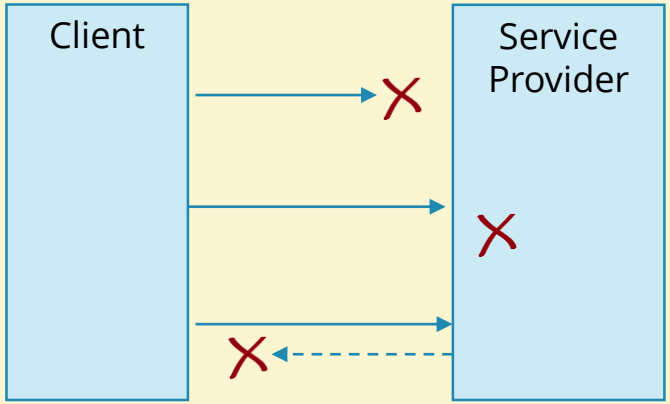
Distributed systems introduce complexity you have to tackle!



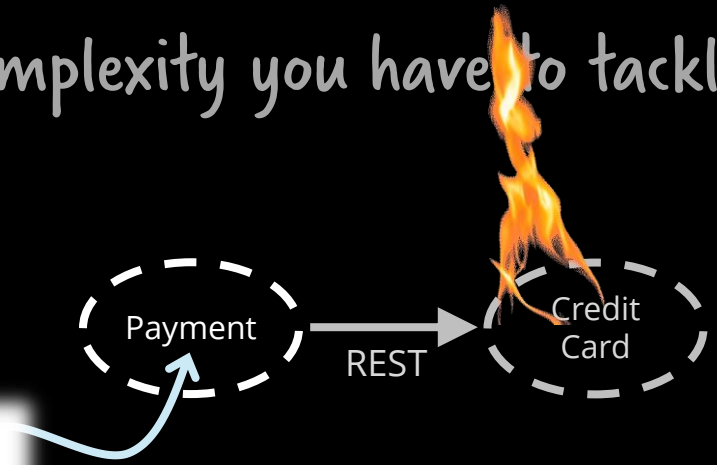


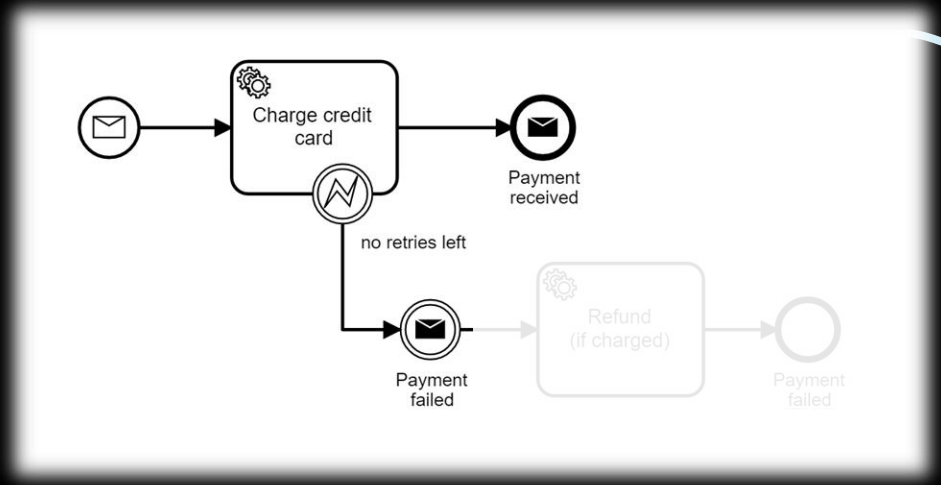
It is impossible to differentiate certain failure scenarios.

Independant of communication style!

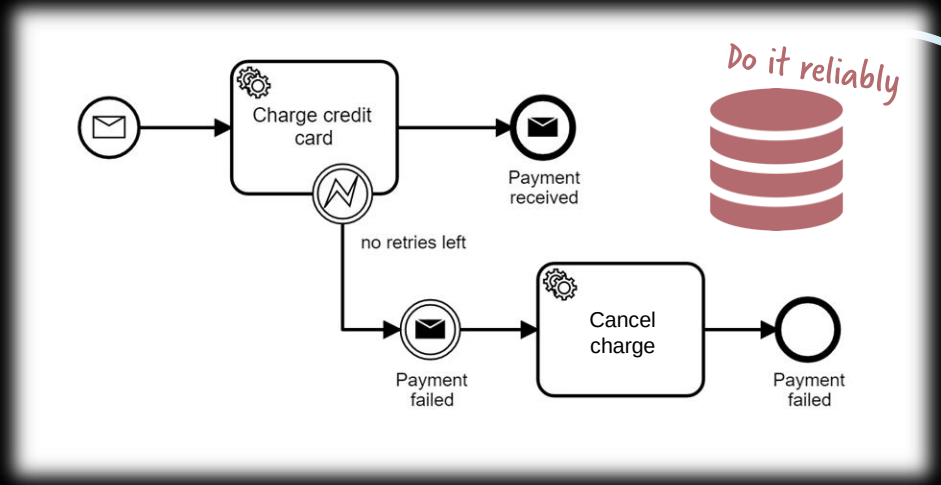


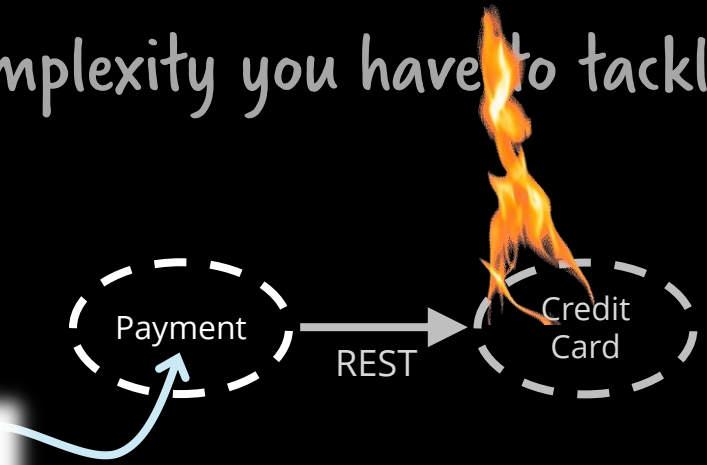
Distributed systems introduce complexity you have to tackle!





Distributed systems introduce complexity you have to tackle!



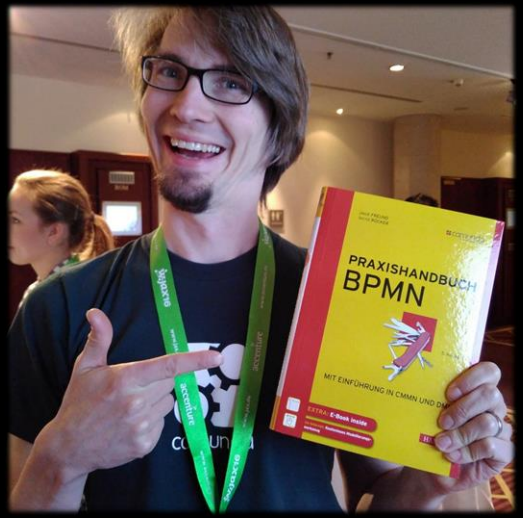


Live hacking

Warning:
Contains Opinion

Bernd Ruecker

Co-founder and
Chief Technologist of
Camunda

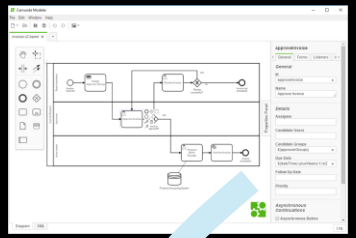


Berlin, Germany

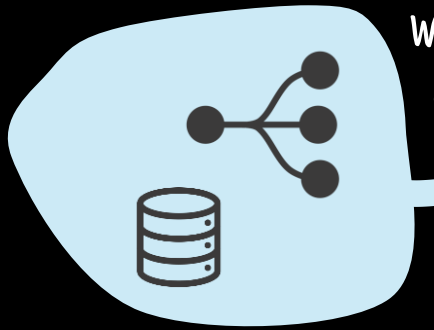
mail@berndruecker.io
@berndruecker


camunda
Live hacking
https://github.com/berndruecker/flowing-retail/tree/master/rest/java/payment-camunda

BPMN – Business Process Model and Notation
ISO Standard

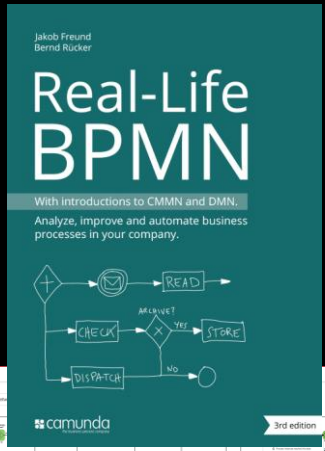


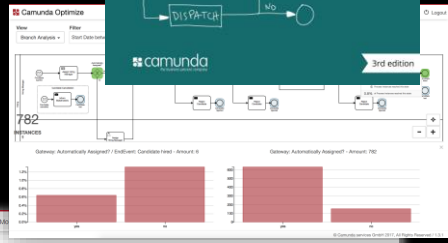
BPMN XML

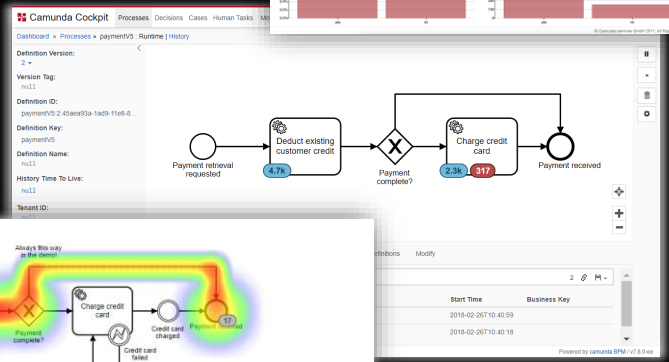


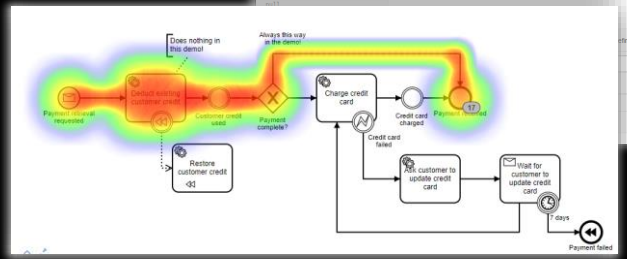
Workflow engine

+ Code, UI, ...







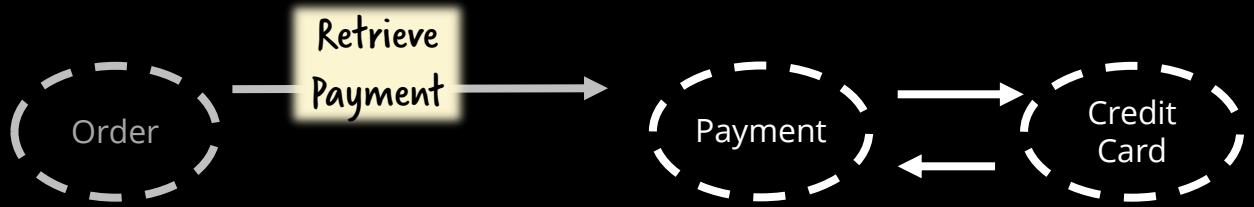


Long running services* provide a better API!

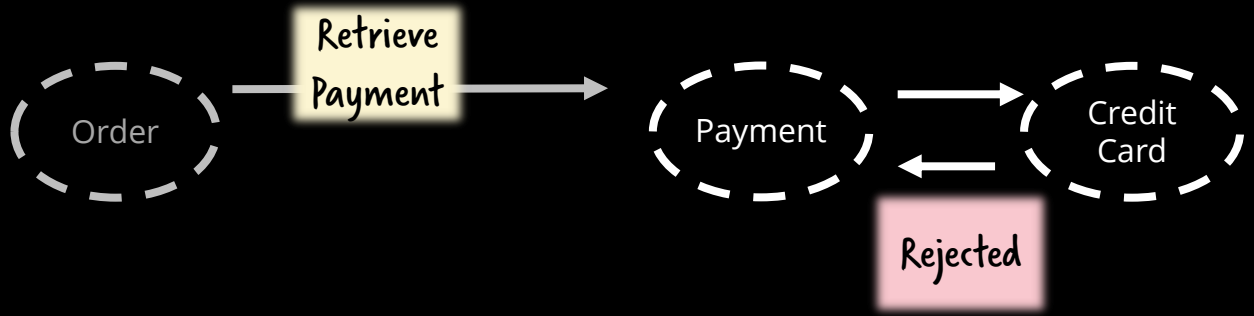
* Services that have the capability to keep state and thus can be potentially long running

Example

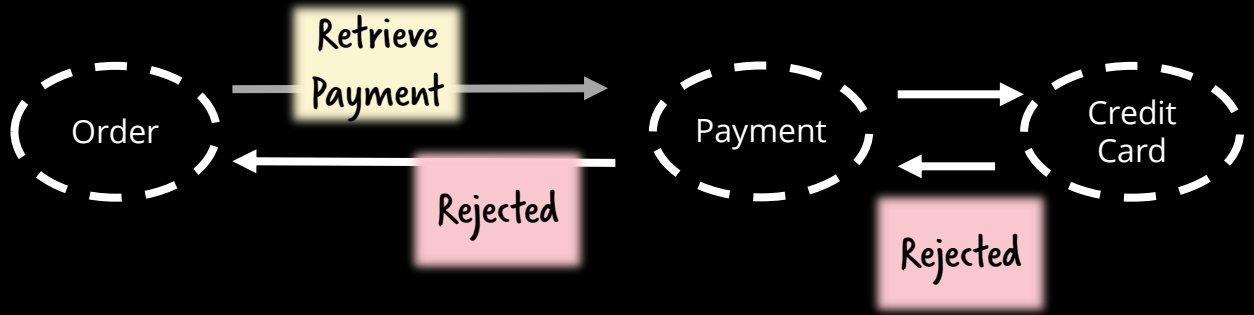
Example



Example

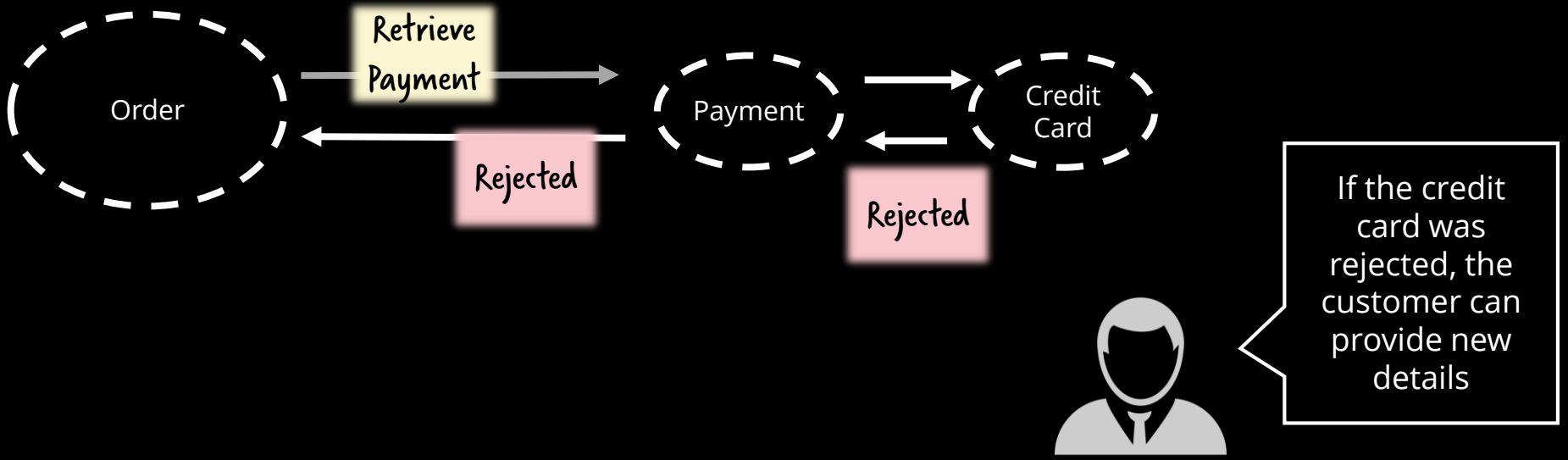


Example



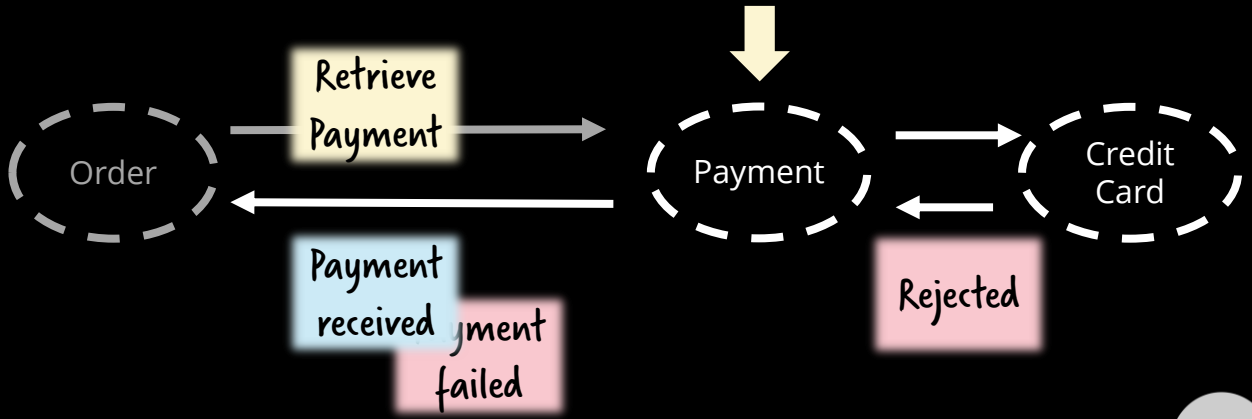
If the credit card was rejected, the customer can provide new details

Example



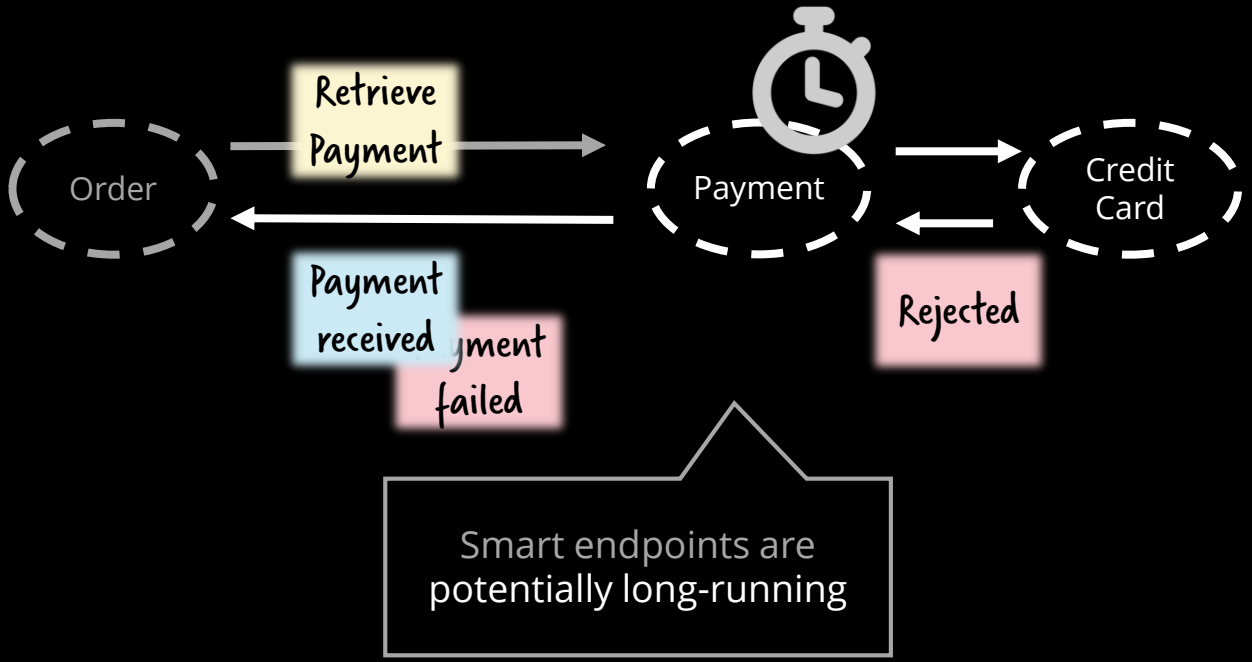
Client of **dumb endpoints** easily become a god services.

Who is responsible to deal with problems?

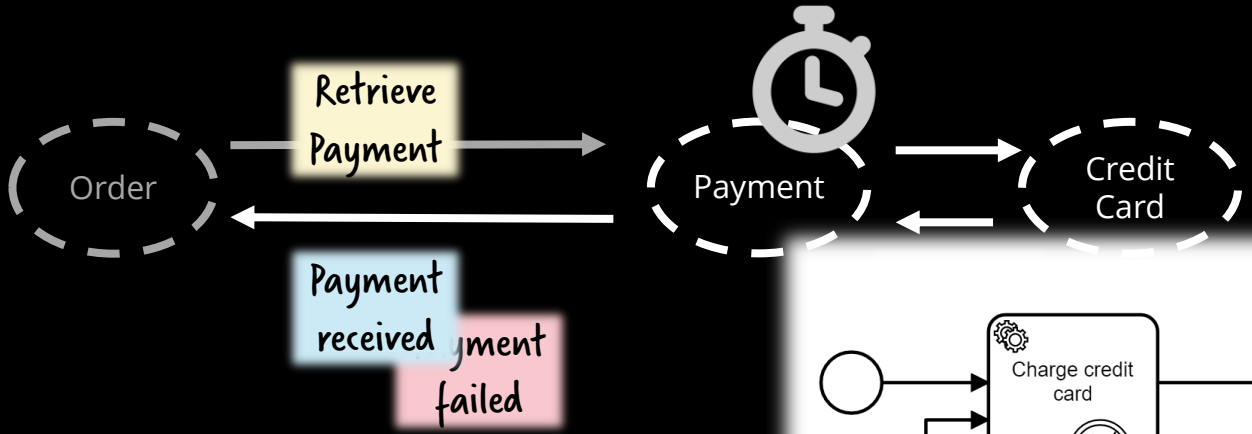


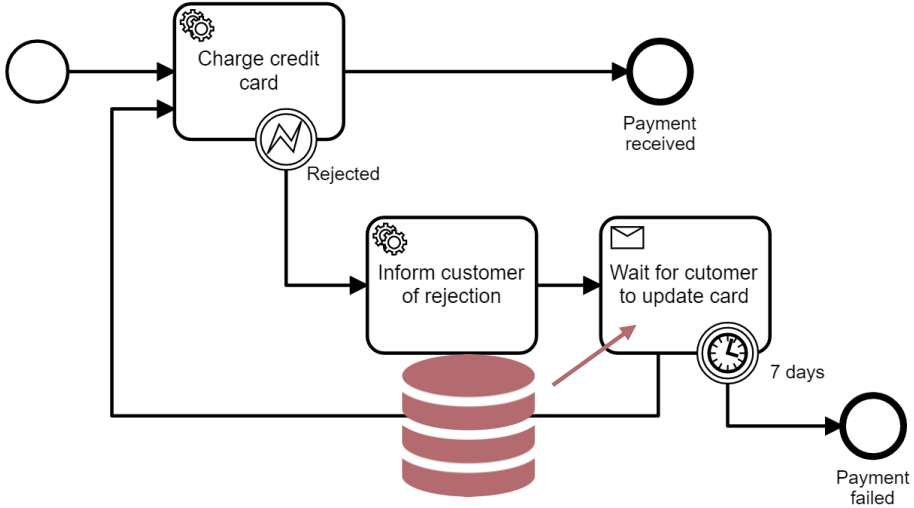
If the credit card was rejected, the customer can provide new details

Long running services

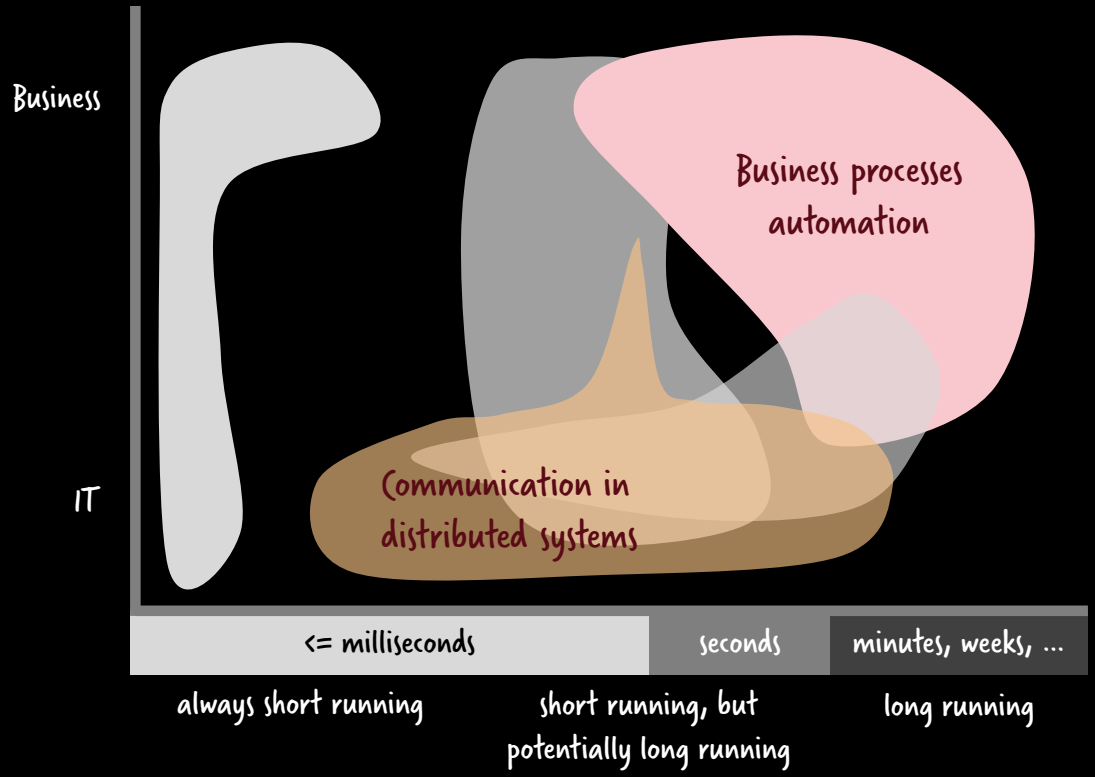


Long running services

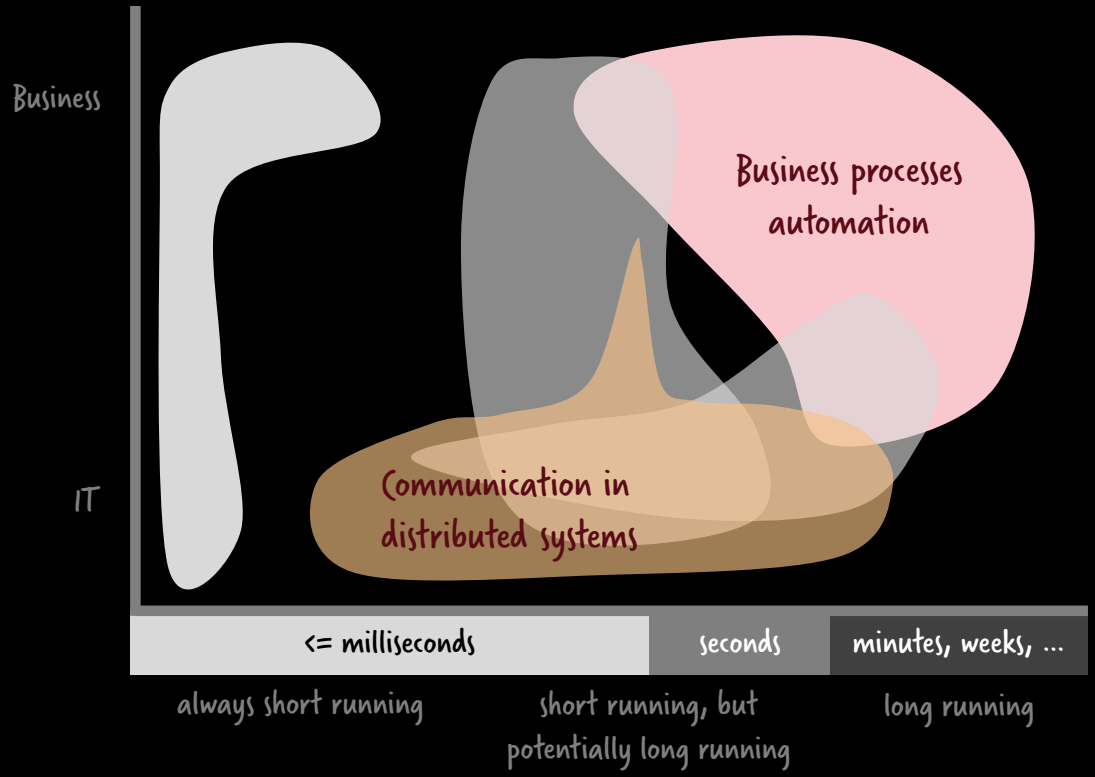




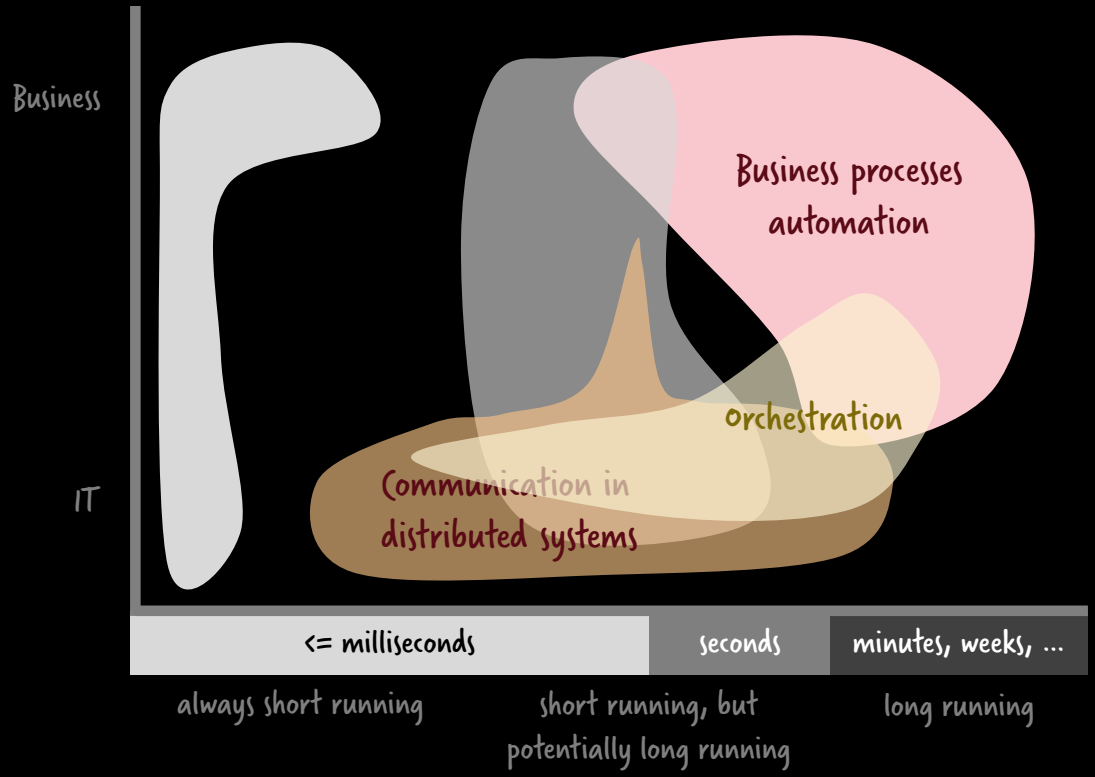
Use cases for workflow automation



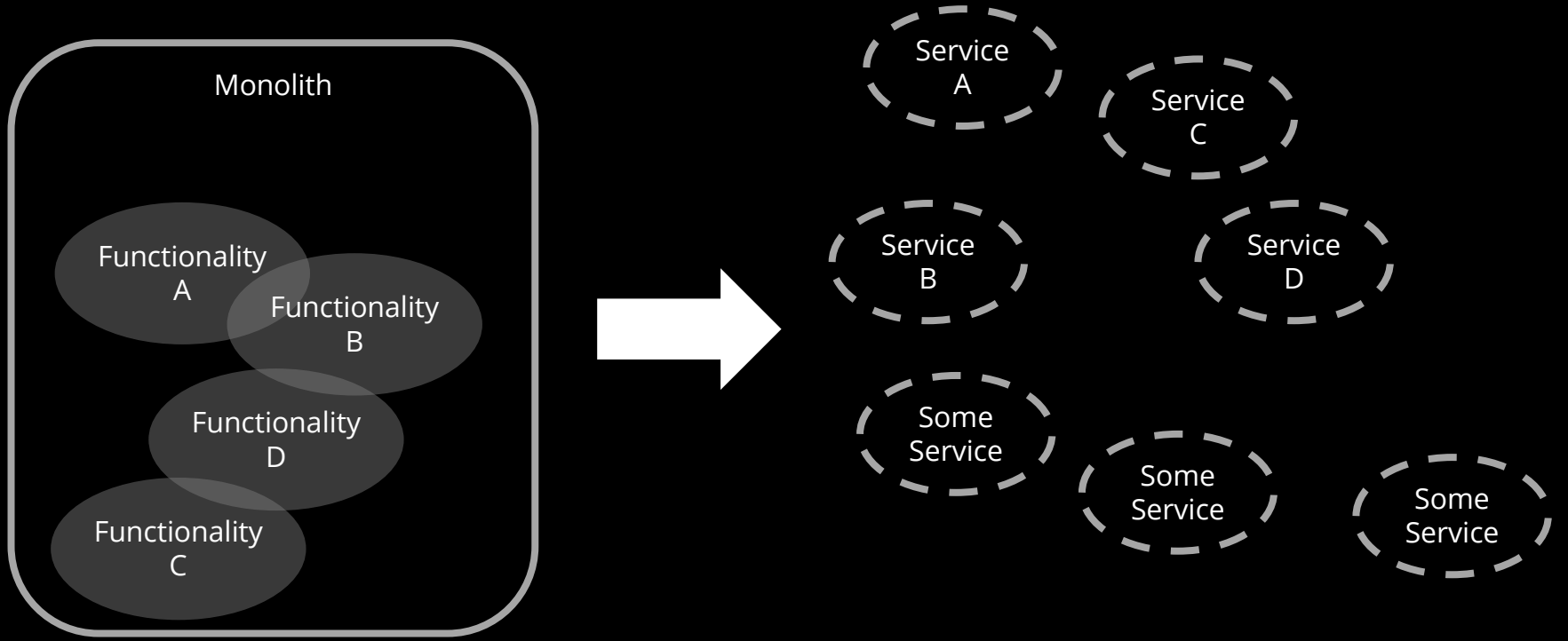
Use cases for workflow automation



Use cases for workflow automation

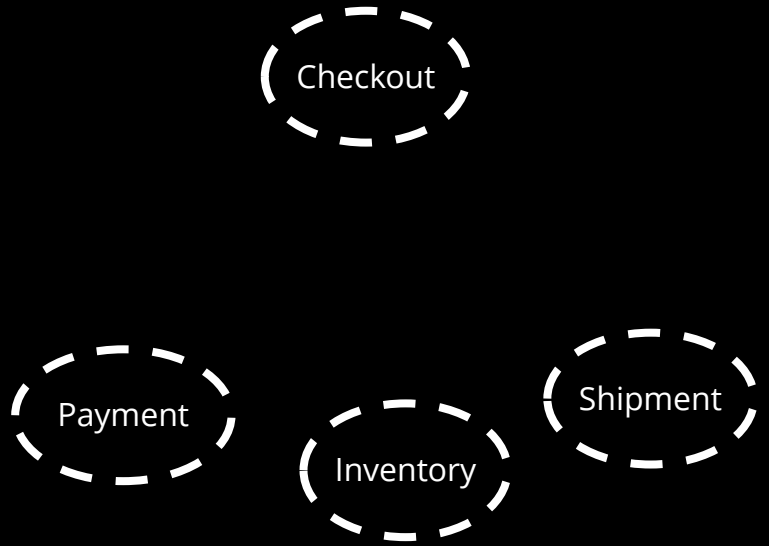


Microservices...

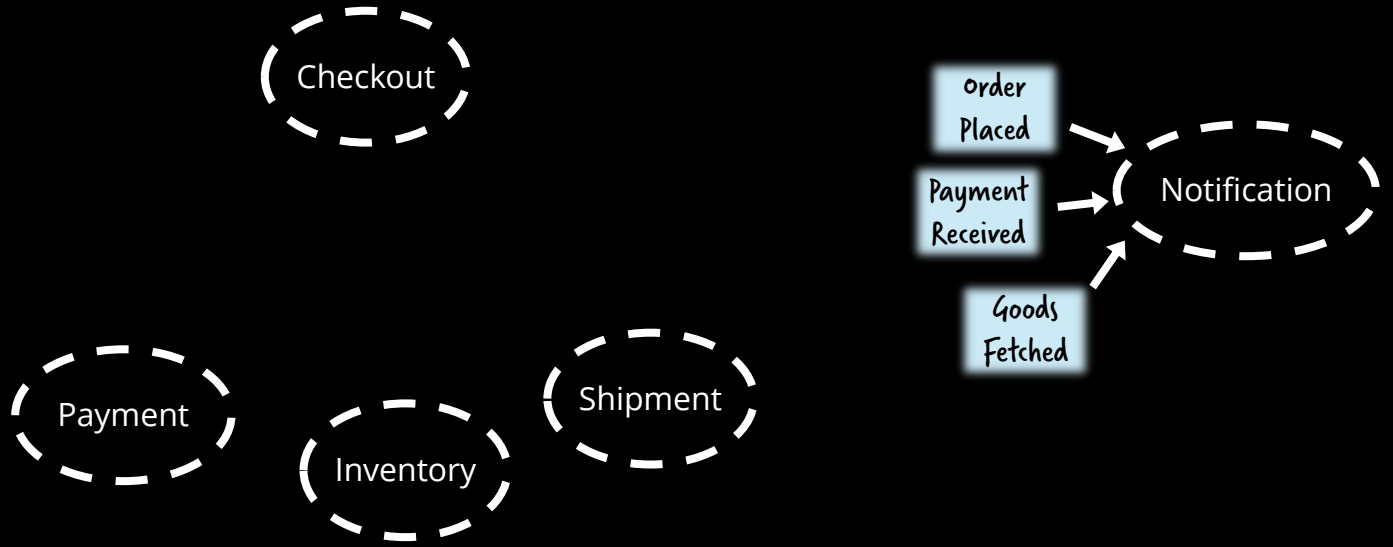


order fulfillment example: dash button

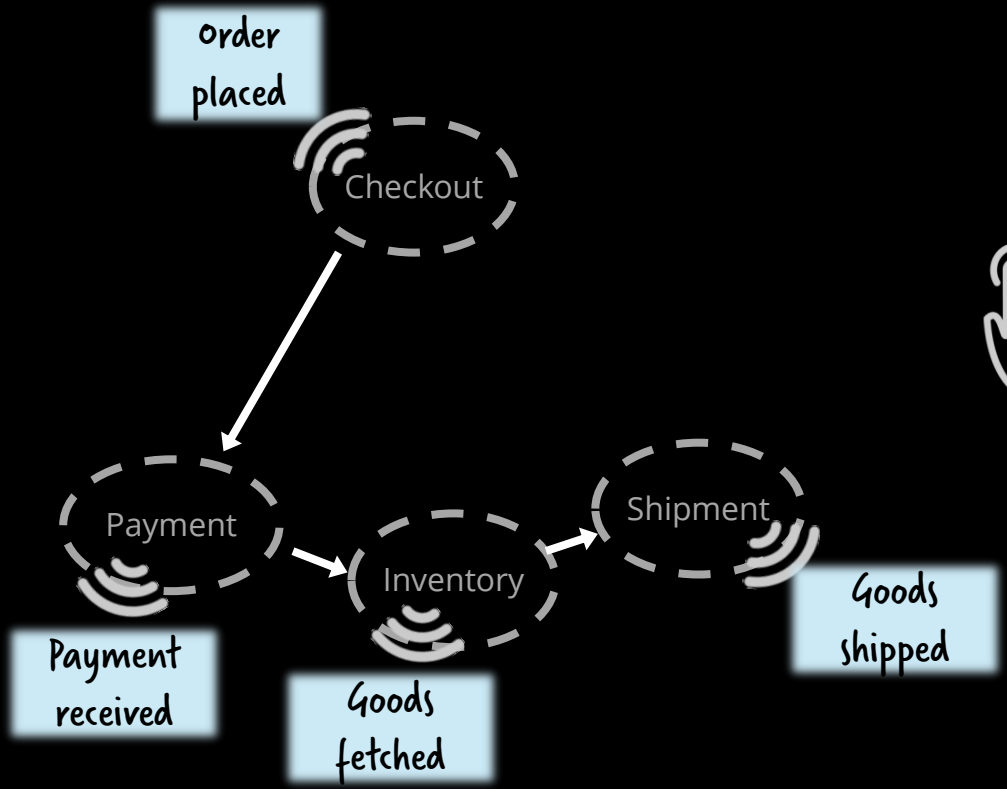
(Micro-)services



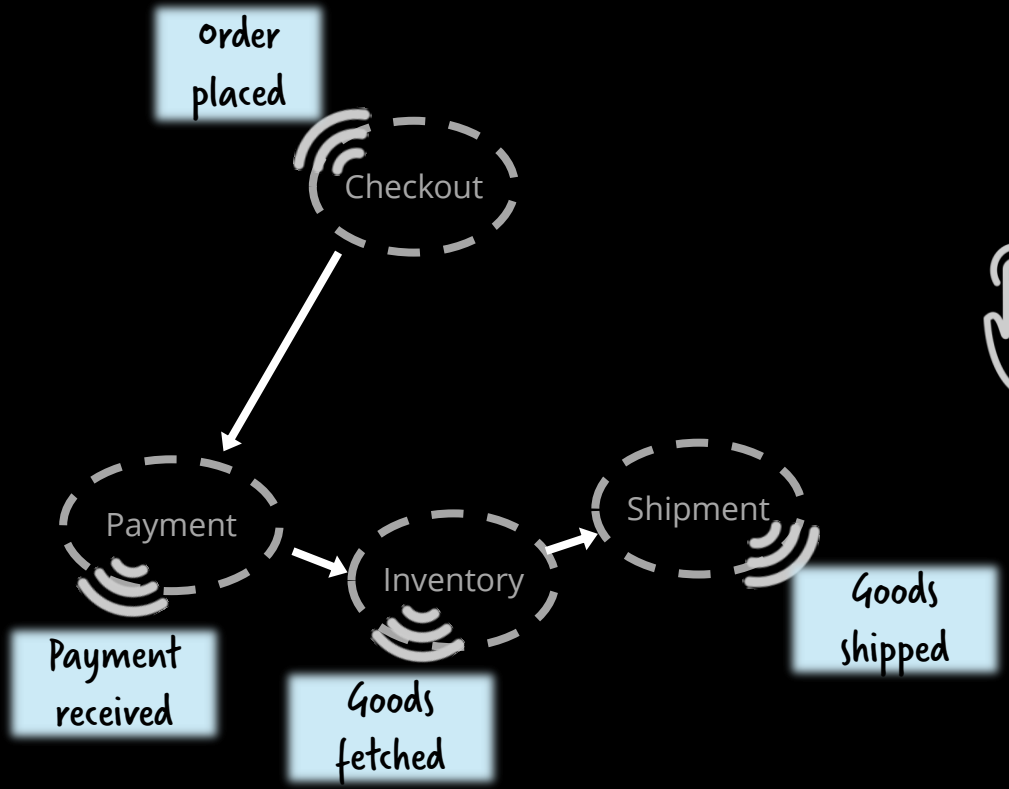
Event-driven architecture

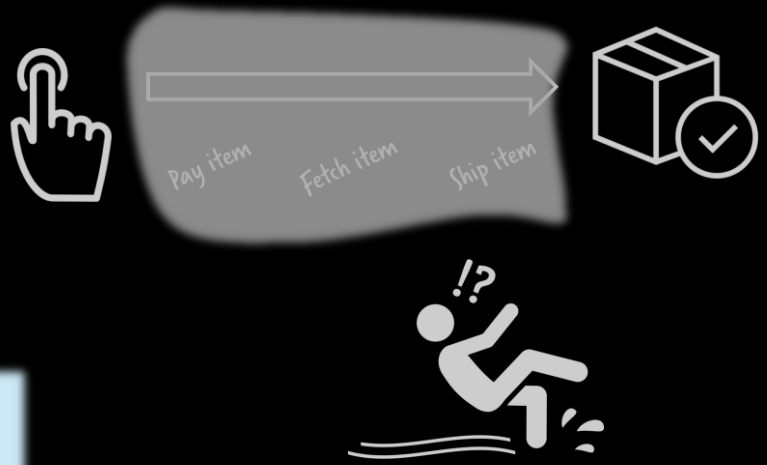


Peer-to-peer event chains



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

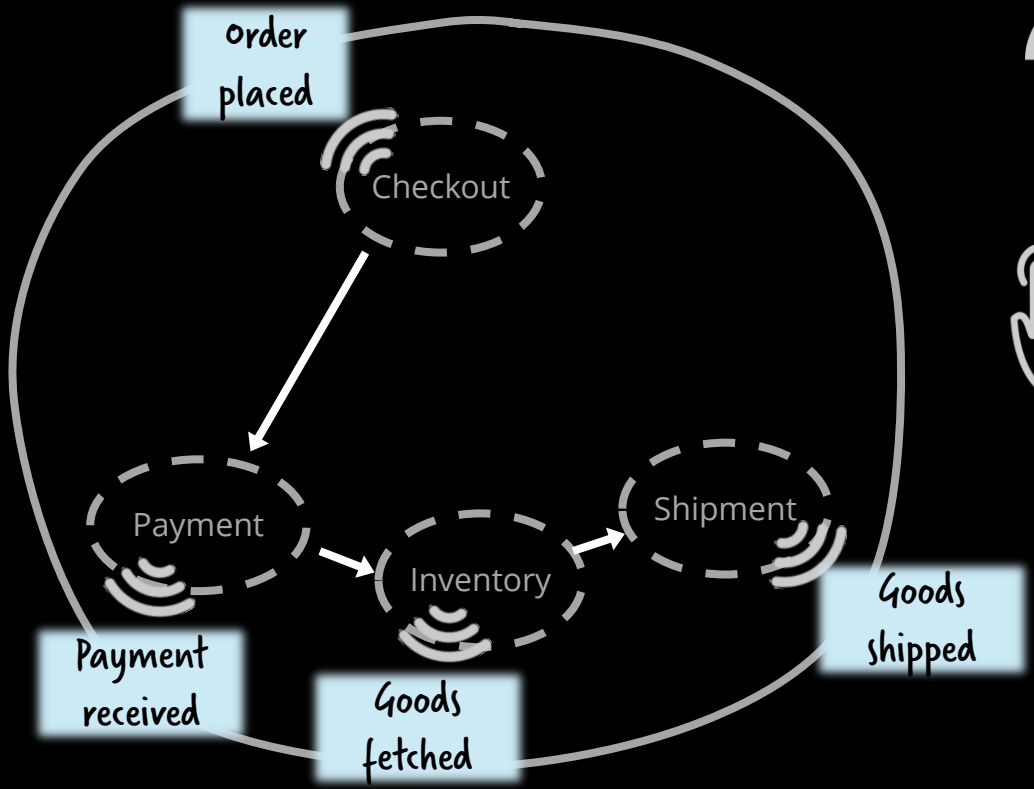
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

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

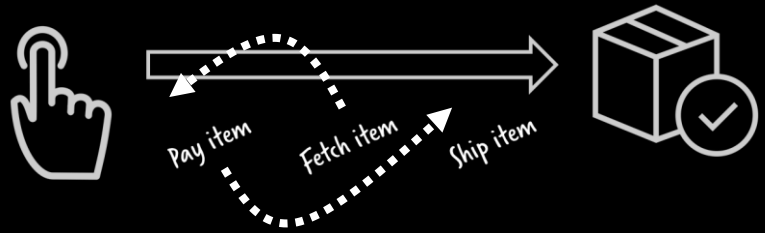
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

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

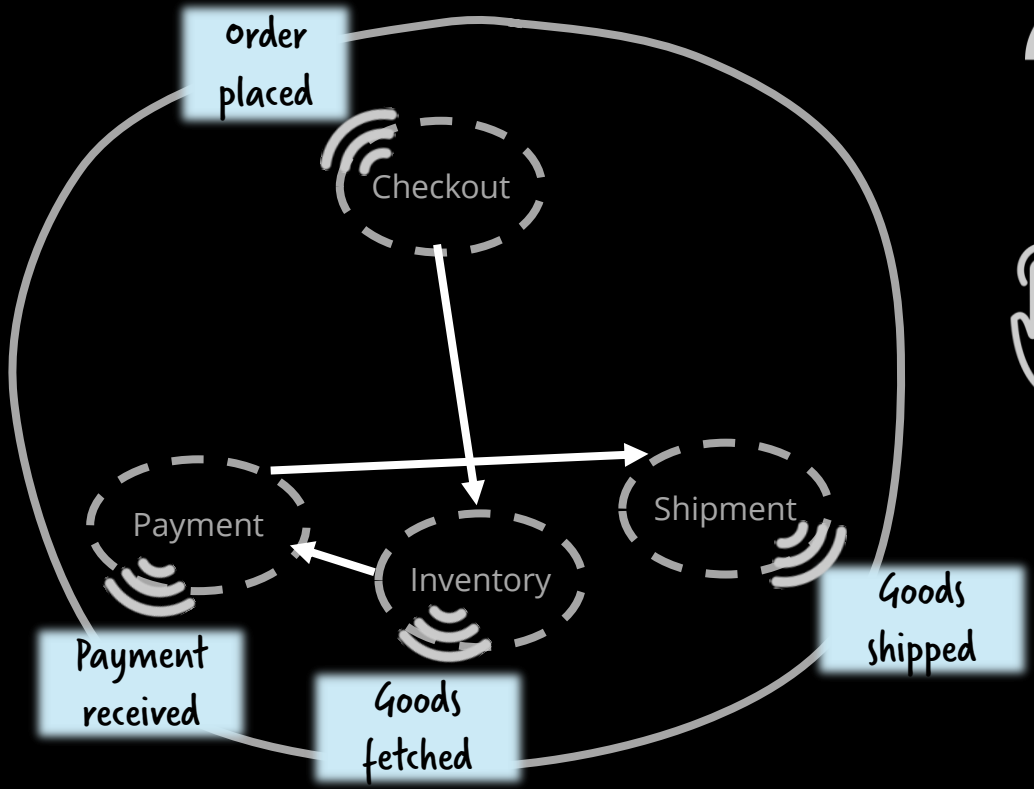
Peer-to-peer event chains



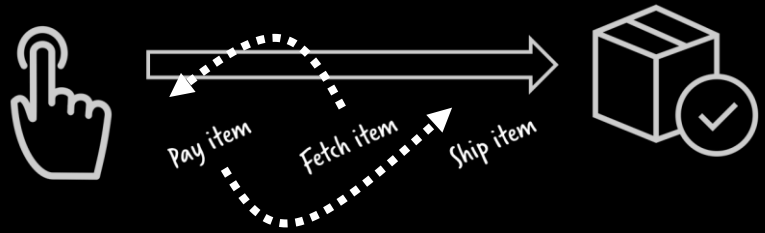
Fetch the goods **before** the payment



Peer-to-peer event chains



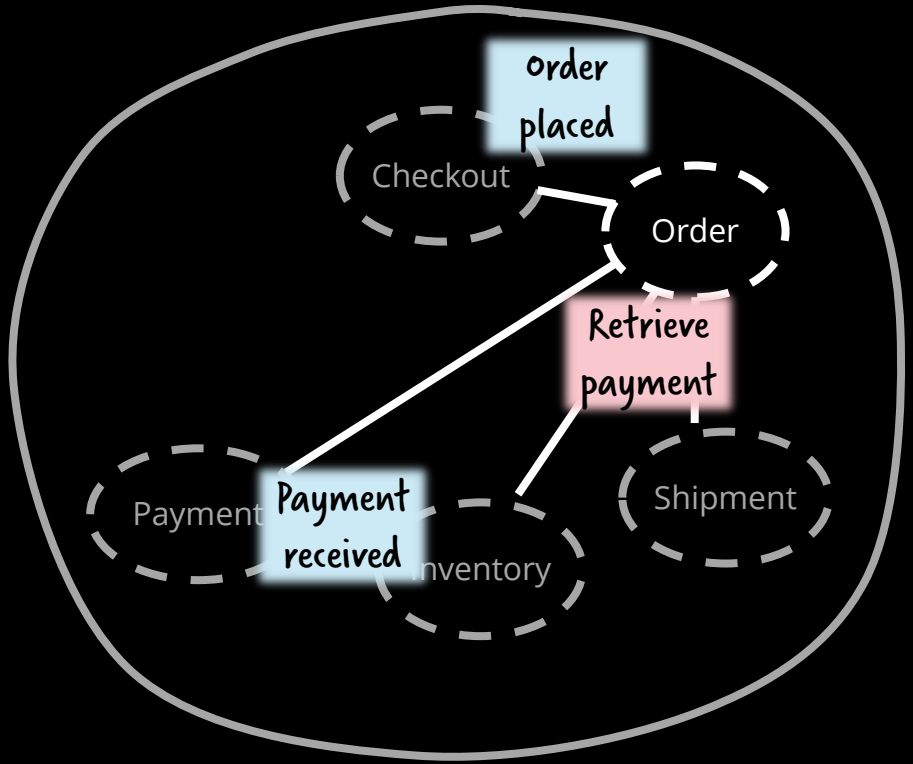
Fetch the goods **before** the payment



What we wanted

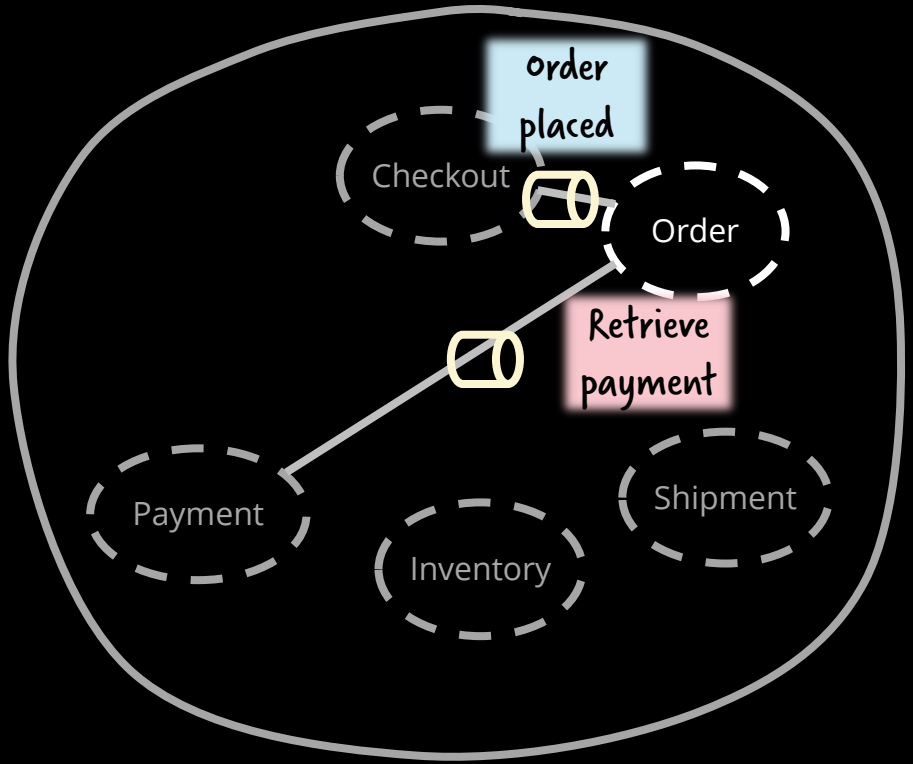
vs. what we got

Extract the end-to-end responsibility

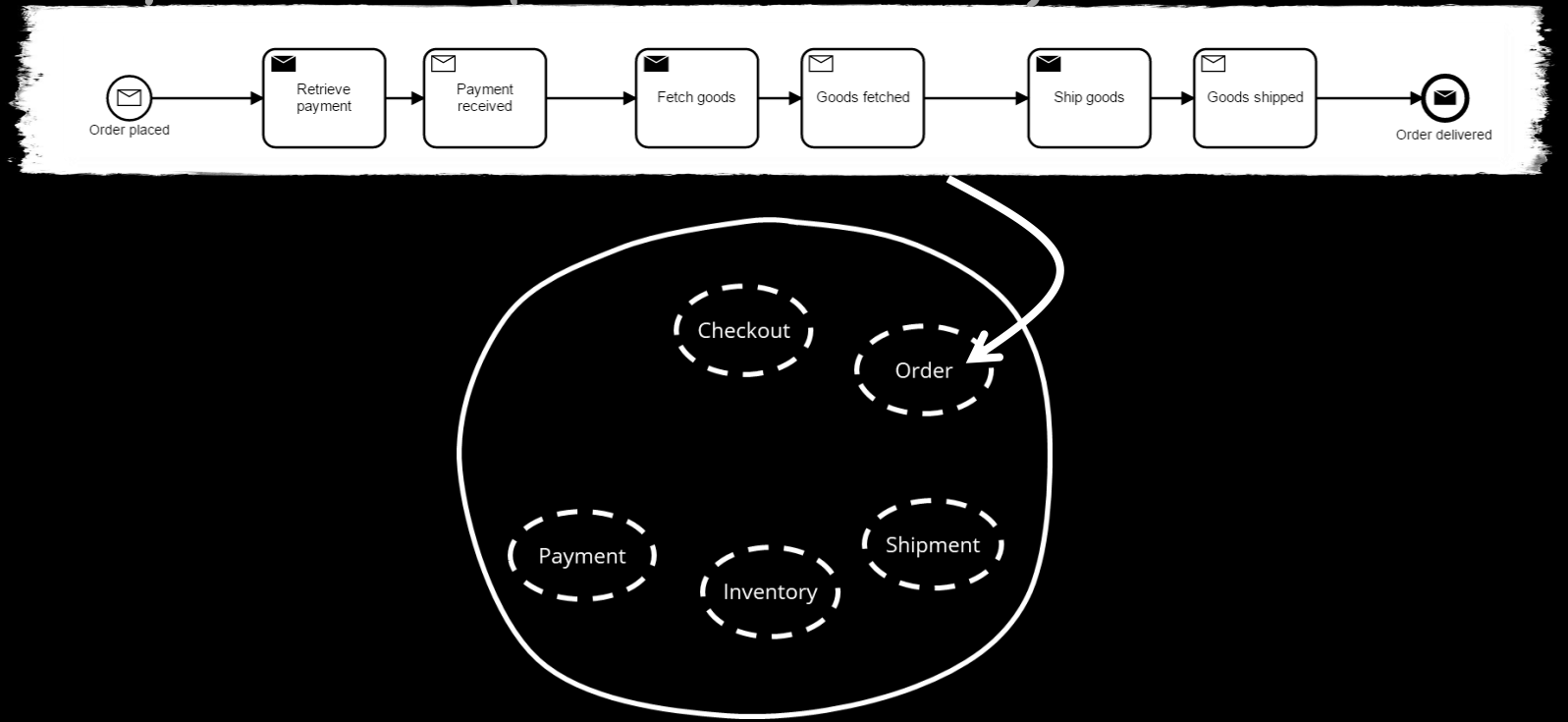


***Commands** have an intent about what needs to happen in the future

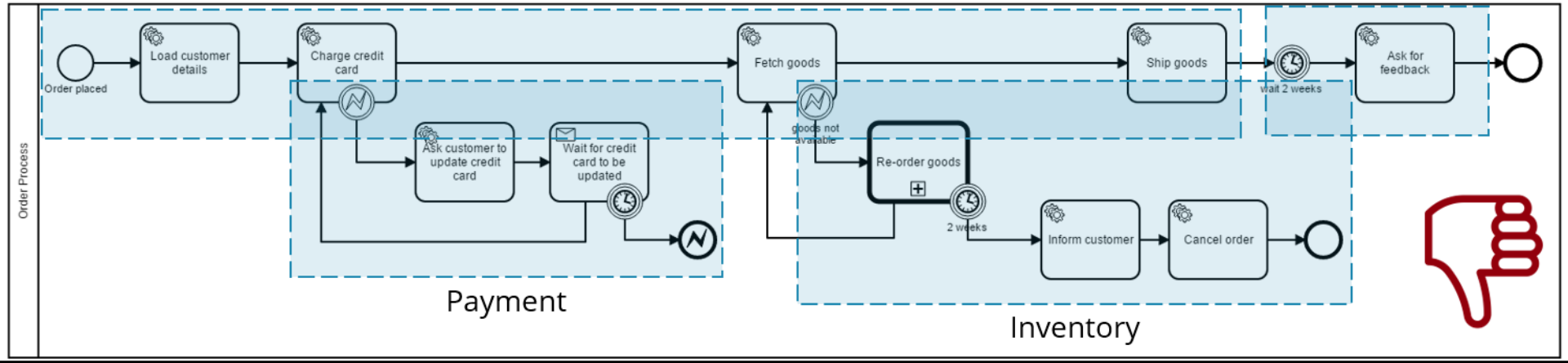
It still can be messaging!



Workflows implement stateful orchestration logic

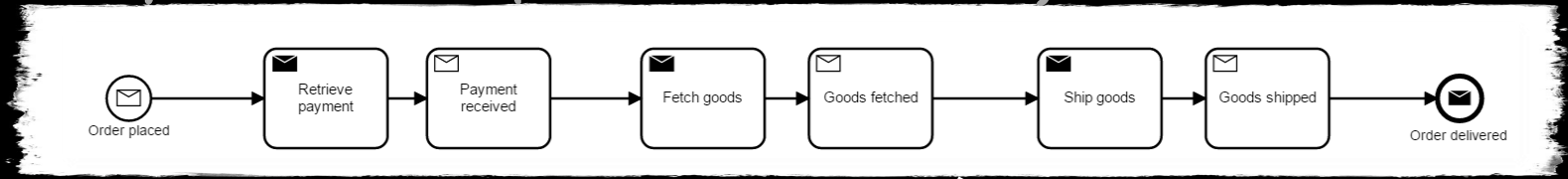


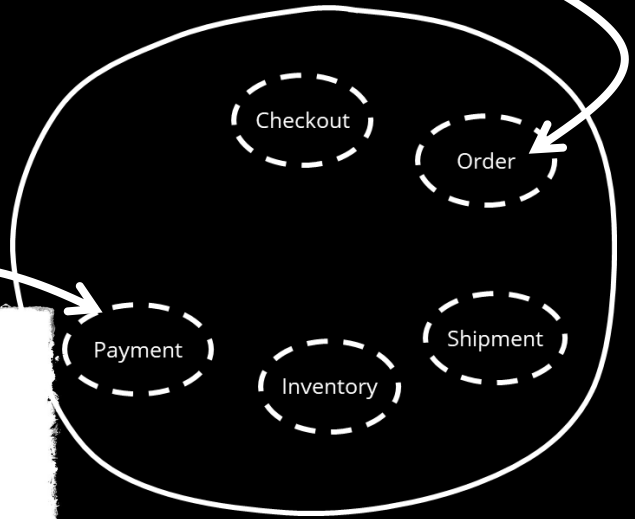
No BPM(N) monoliths

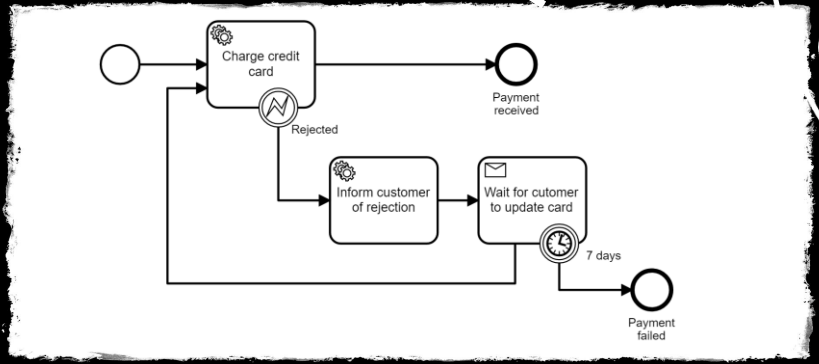


https://blog.bernd-ruecker.com/avoiding-the-bpm-monolith-when-using-bounded-contexts-d86be6308d8

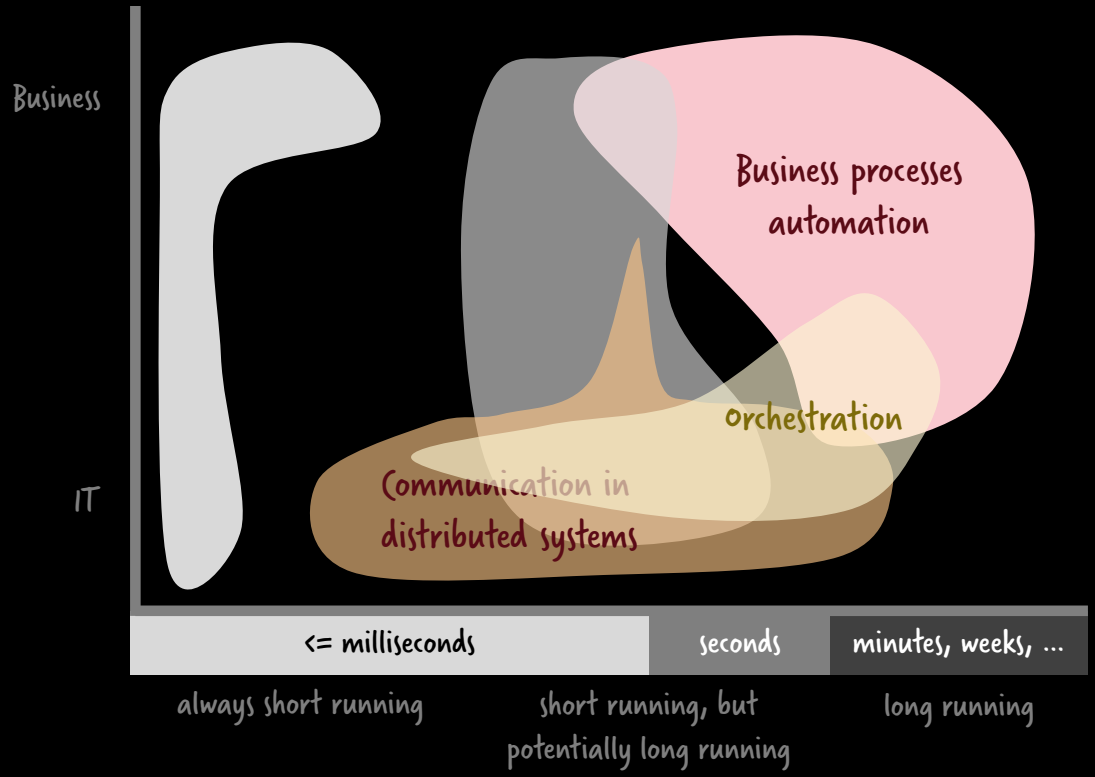
Workflows implement stateful orchestration logic



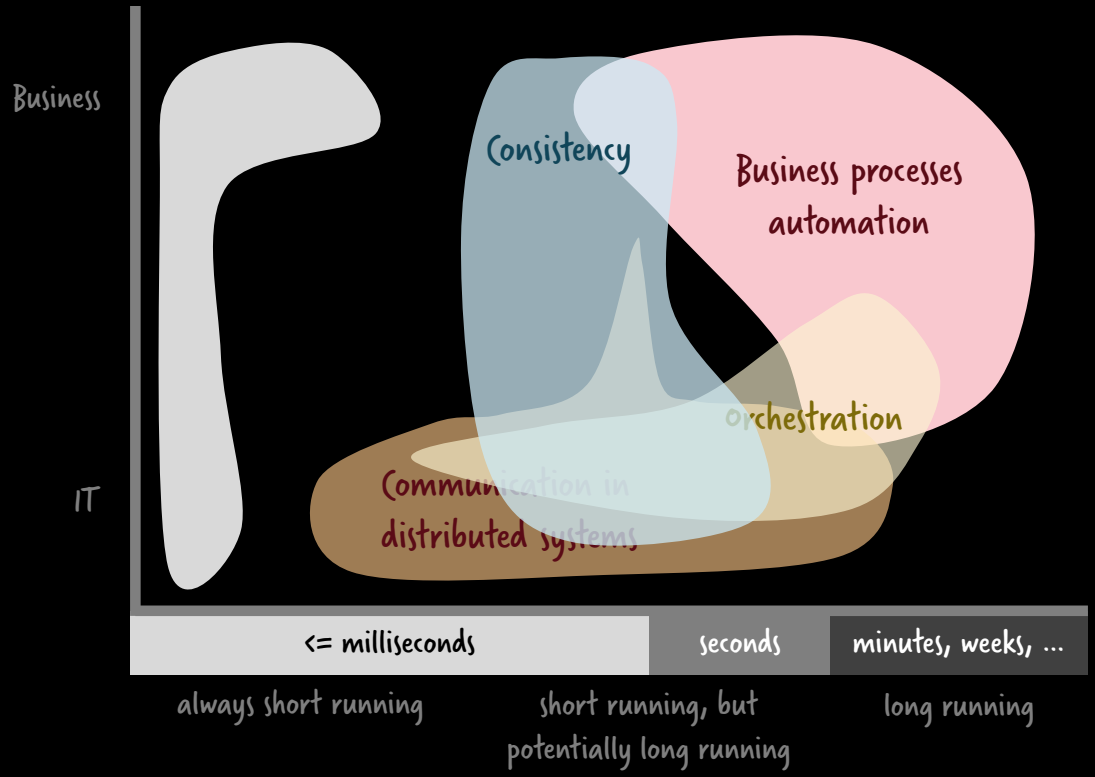




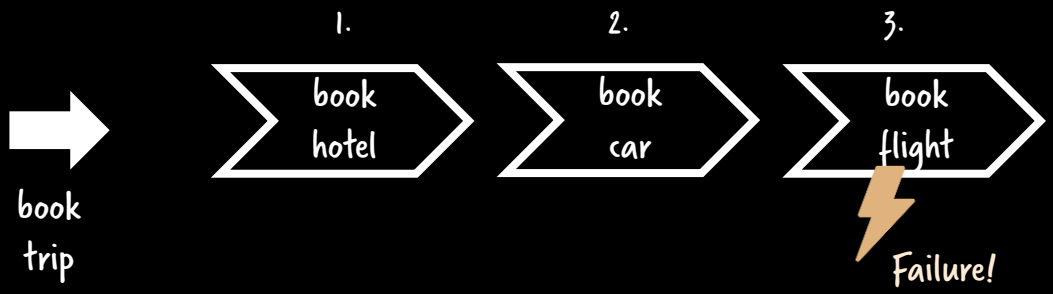
Use cases for workflow automation



Use cases for workflow automation



The classical example



Pat Helland

Distributed Systems Guru
Worked at Amazon,
Microsoft & Salesforce

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, Paxos, and various approaches to quorum. These protocols provide the application programmer a façade of global serializability. Personally, I have invested a non-trivial portion of my career as a strong advocate for the implementation and use of platforms

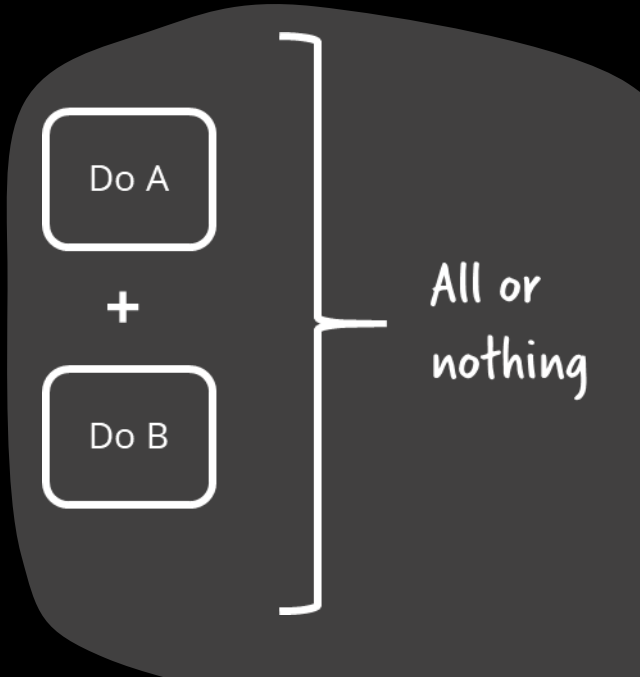
Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses.

This paper explores and names some of the practical approaches used in the implementations of large-scale mission-critical applications in a world which rejects distributed transactions. We discuss the management of fine-grained pieces of application data which may be repartitioned over time as the application grows. We also discuss the design patterns used in sending messages between these repartitionable pieces of data

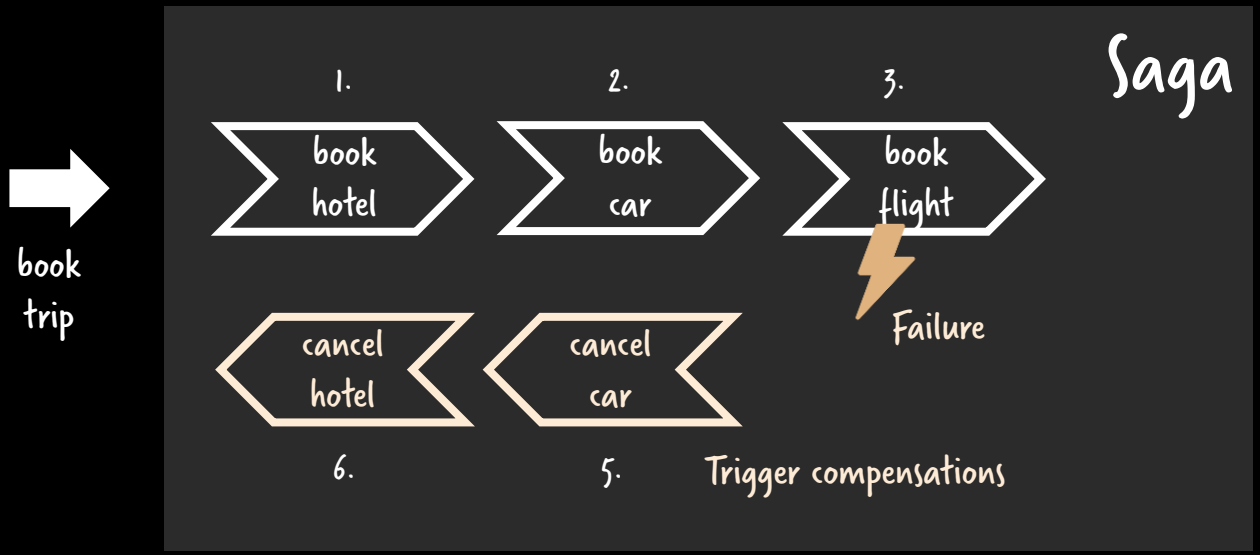
"Grown-Ups Don't Use Distributed Transactions

Pat Helland

Distributed Systems Guru
Worked at Amazon,
Microsoft & Salesforce

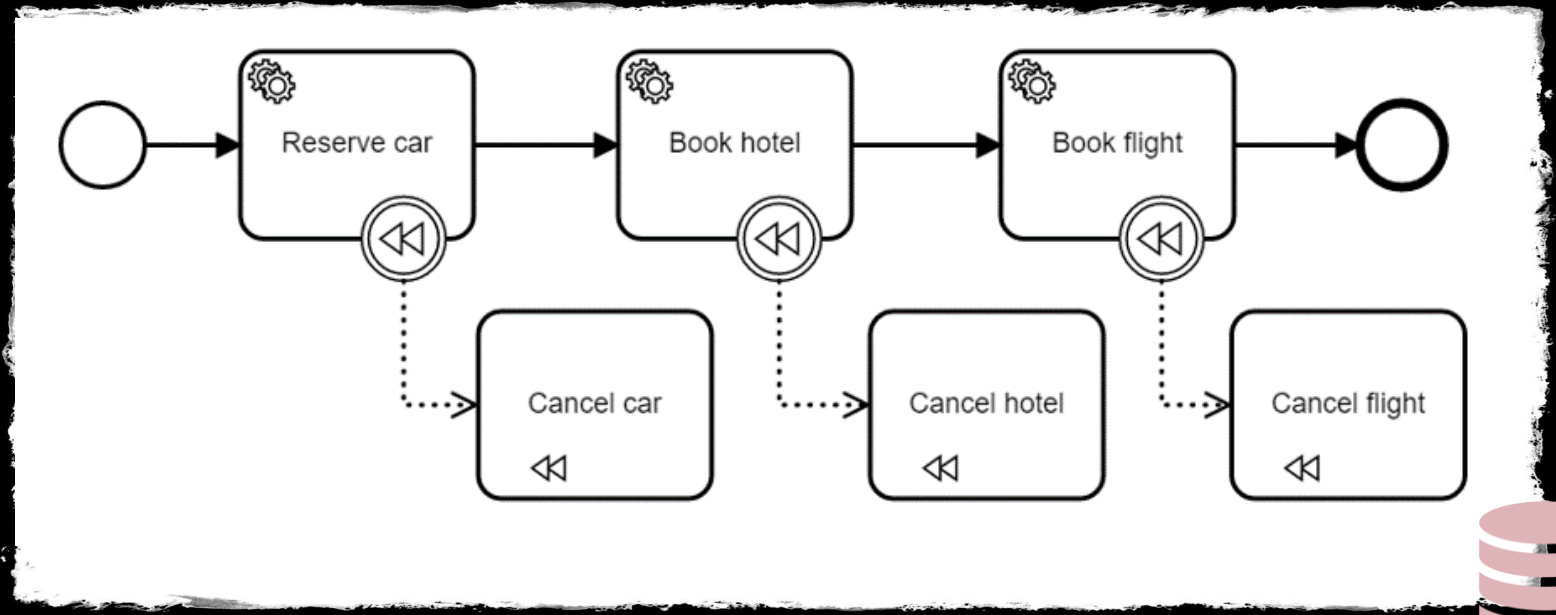


The classical example



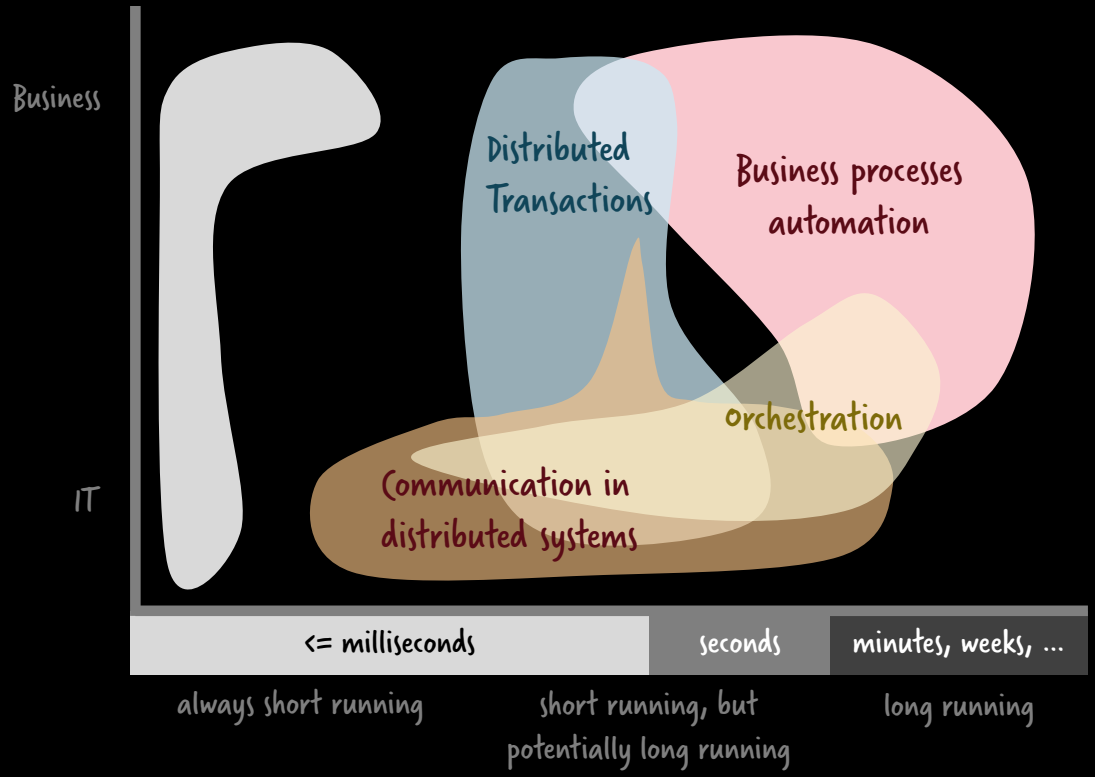
BPMN

Saga Pattern
(implemented by BPMN compensation)

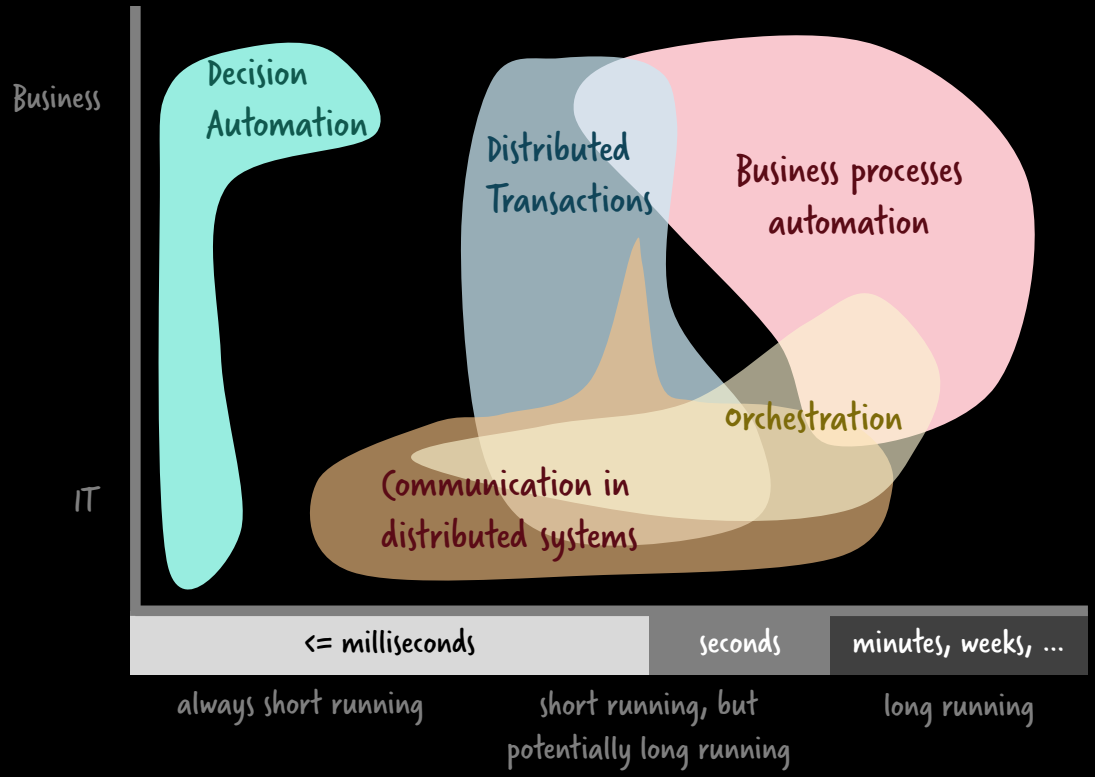


https://github.com/berndruecker/trip-booking-saga-java

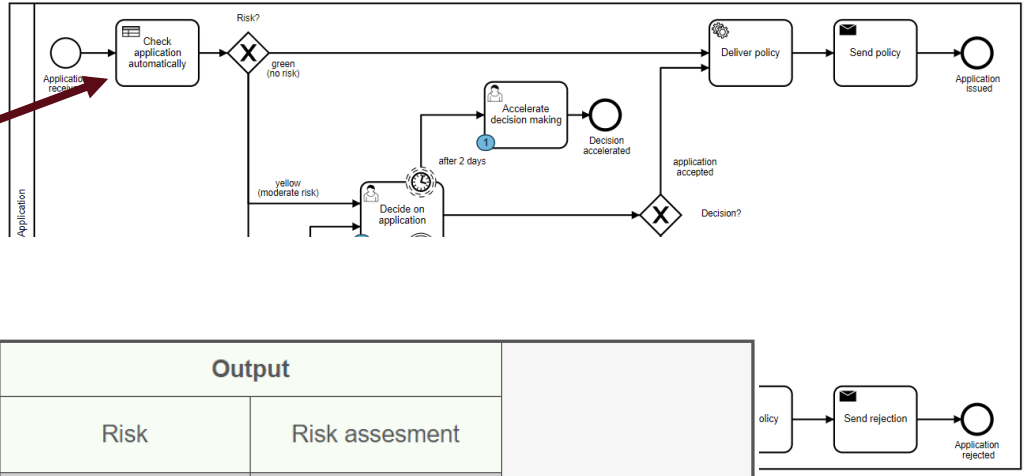
Use cases for workflow automation



Use cases for workflow automation



Decisions with DMN



Risk Assessment

C	Input			Output		Annotation
	Age = **37**	Car manufacturer = **Porsche**	Car type = **911**	Risk	Risk assesment	
	integer	string	string	string	string	
1	<= 21	-	-	"Beginner"	"yellow"	
2	<= 25	"Porsche"	-	"Young and too fast"	"red"	No sorry - that's too risky!
3	<= 30	"BMW"	-	"Young and fast"	"yellow"	
4		"Porsche"	"911"	"Fast and furious" = **Fast and furious**	"yellow" **= yellow**	
5		"BMW"	"X3"	"High value vehicle"	"yellow"	

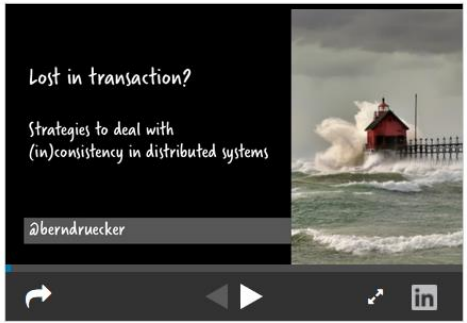
See more on http://berndruecker.io/

Lost in transaction? Strategies to manage consistency in distributed systems

You probably work on a distributed system. Even if you don't yet face a serverless microservice architecture using fancy NoSQL databases, you might simply call some remote services via REST or SOAP. This leaves you in charge of dealing with consistency yourself. ... read more ...

All occurances on conferences...

Slides

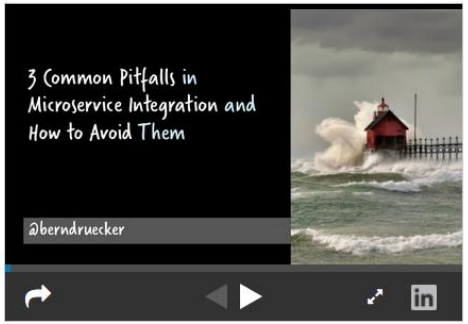


3 common pitfalls in microservice integration and how to avoid them

Integrating microservices and taming distributed systems is hard. In this talk I will present three challenges I've observed in real-life projects and discuss how to avoid them. I will not only use slides but also demonstrate concrete source code examples available on GitHub. ... read more ...

All occurances on conferences...

Slides

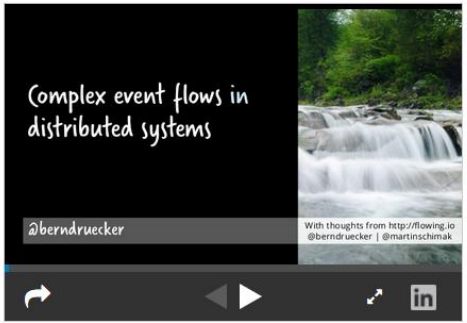


Complex event flows in distributed systems

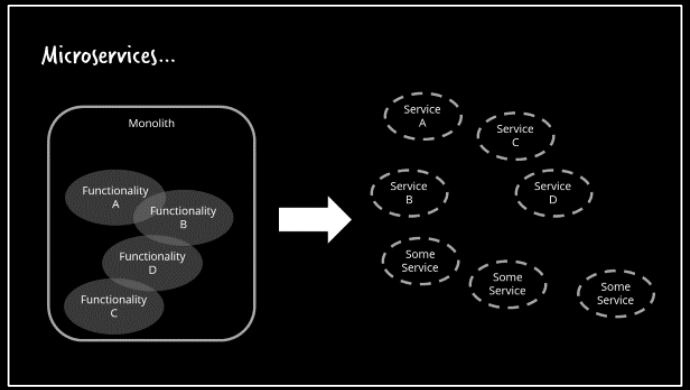
Event-driven architectures enable nicely decoupled microservices and are fundamental for decentral data management. However, using peer-to-peer event chains to implement complex end-to-end logic crossing service boundaries can accidentally increase coupling ... read more ...

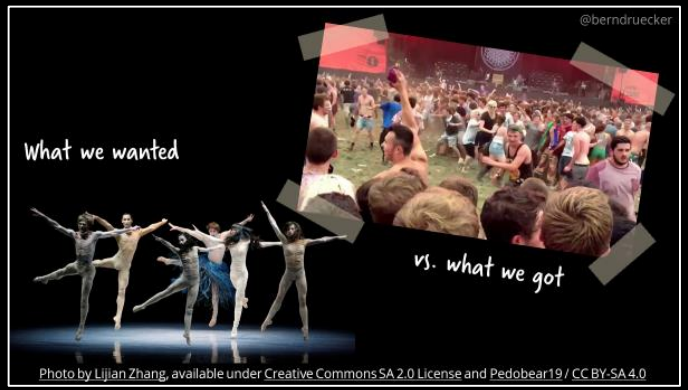
All occurances on conferences...

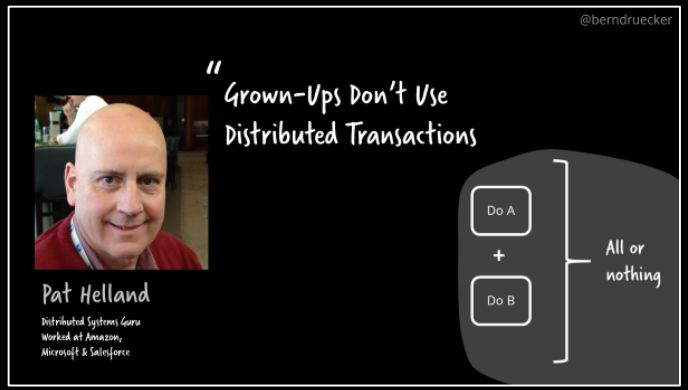
Slides



Workflow Automation is important in modern architectures!

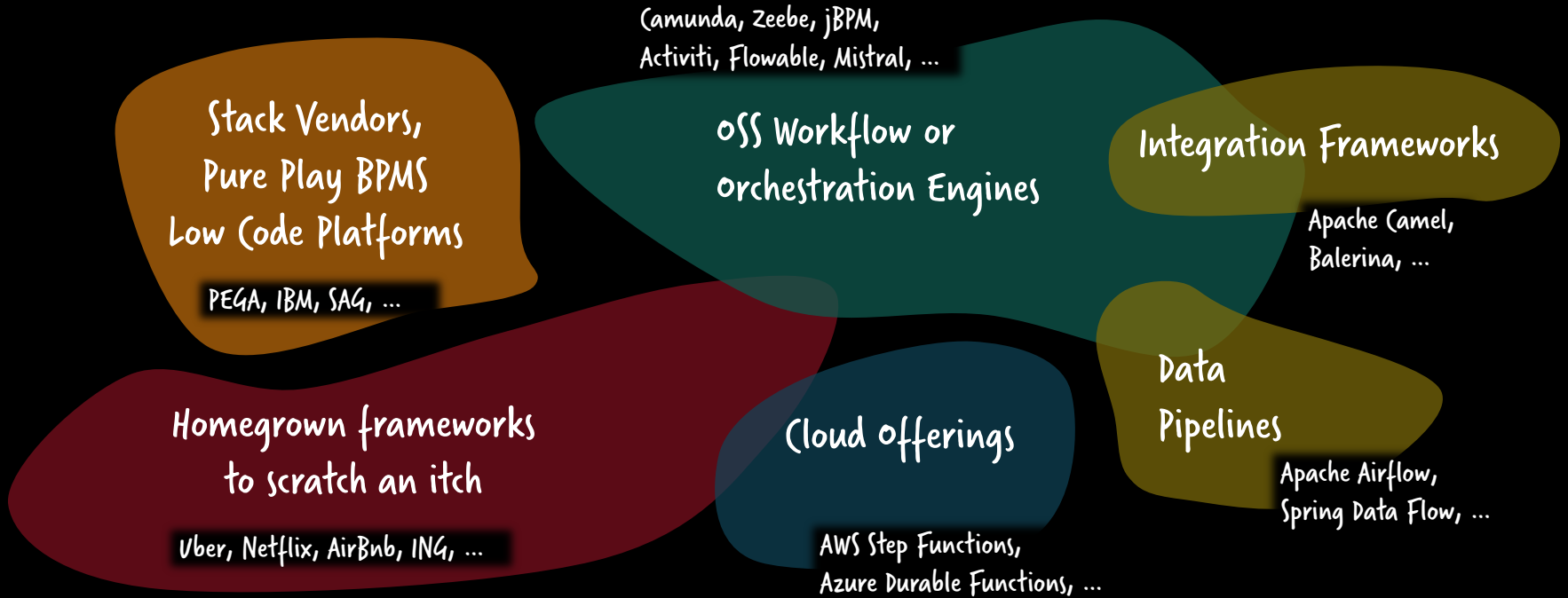


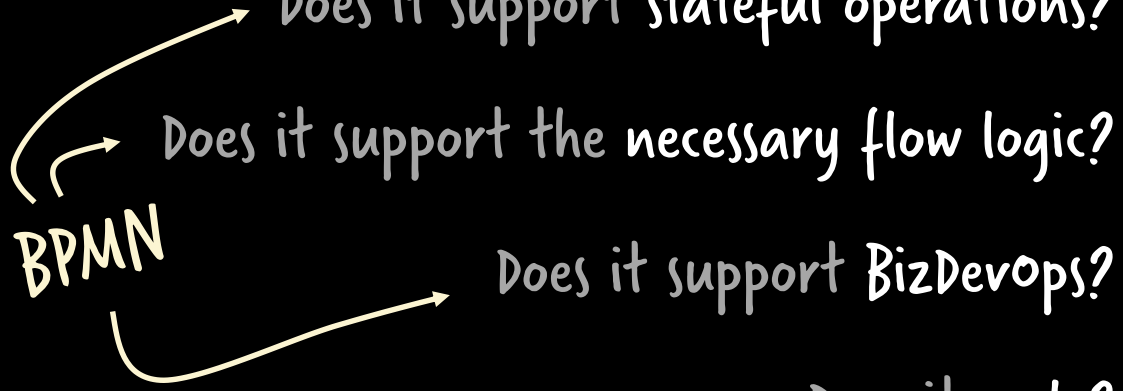




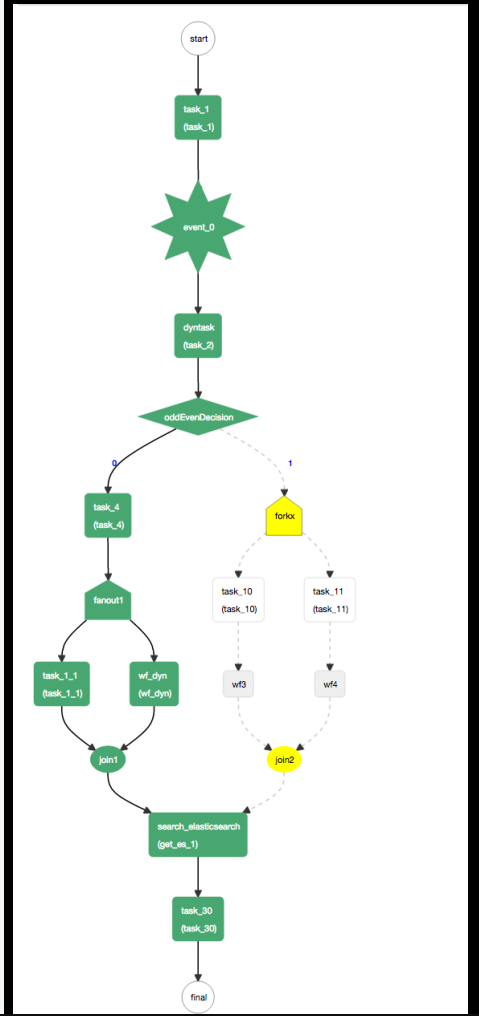
Thoughts on the state machine | workflow engine market

Thoughts on the state machine | workflow engine market



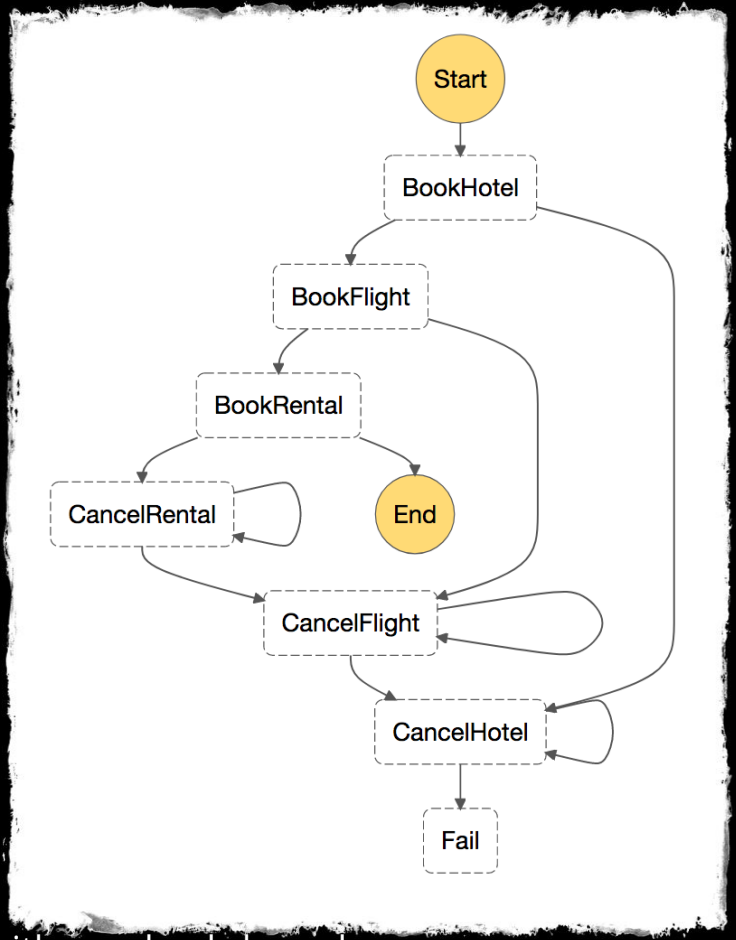

Does it support stateful operations?
Does it support the necessary flow logic?
BPMN
Does it support BizDevOps?
Does it scale?


```
{
  "name": "kitchensink",
  "description": "kitchensink workflow",
  "version": 1,
  "tasks": [
    {
      "name": "task_1",
      "taskReferenceName": "task_1",
      "inputParameters": {
        "mod": "${workflow.input.mod}",
        "oddEven": "${workflow.input.oddEven}"
      },
      "type": "SIMPLE"
    },
    {
      "name": "event_task",
      "taskReferenceName": "event_0",
      "inputParameters": {
        "mod": "${workflow.input.mod}",
        "oddEven": "${workflow.input.oddEven}"
      },
      "type": "EVENT",
      "sink": "conductor"
    },
    {
      "name": "dyntask",
      "taskReferenceName": "task_2",
      "inputParameters": {
        "taskToExecute": "${workflow.input.task2Name}"
      },
      "type": "DYNAMIC",
      "dynamicTaskNameParam": "taskToExecute"
    },
```



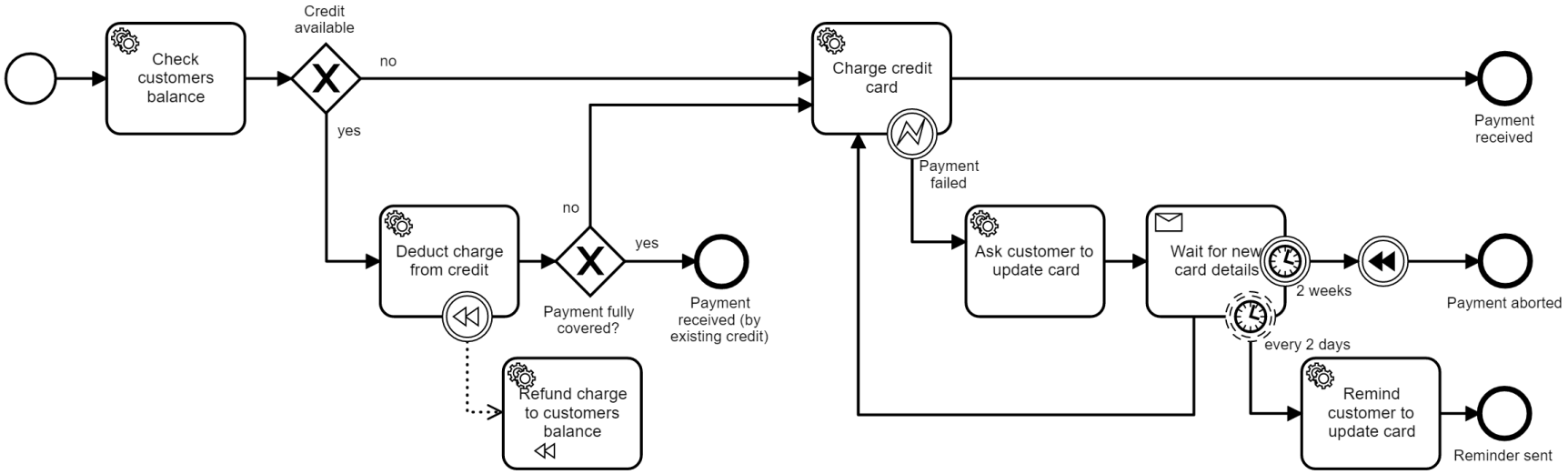
https://netflix.github.io/conductor/metadata/kitchensink/

AWS Step Functions



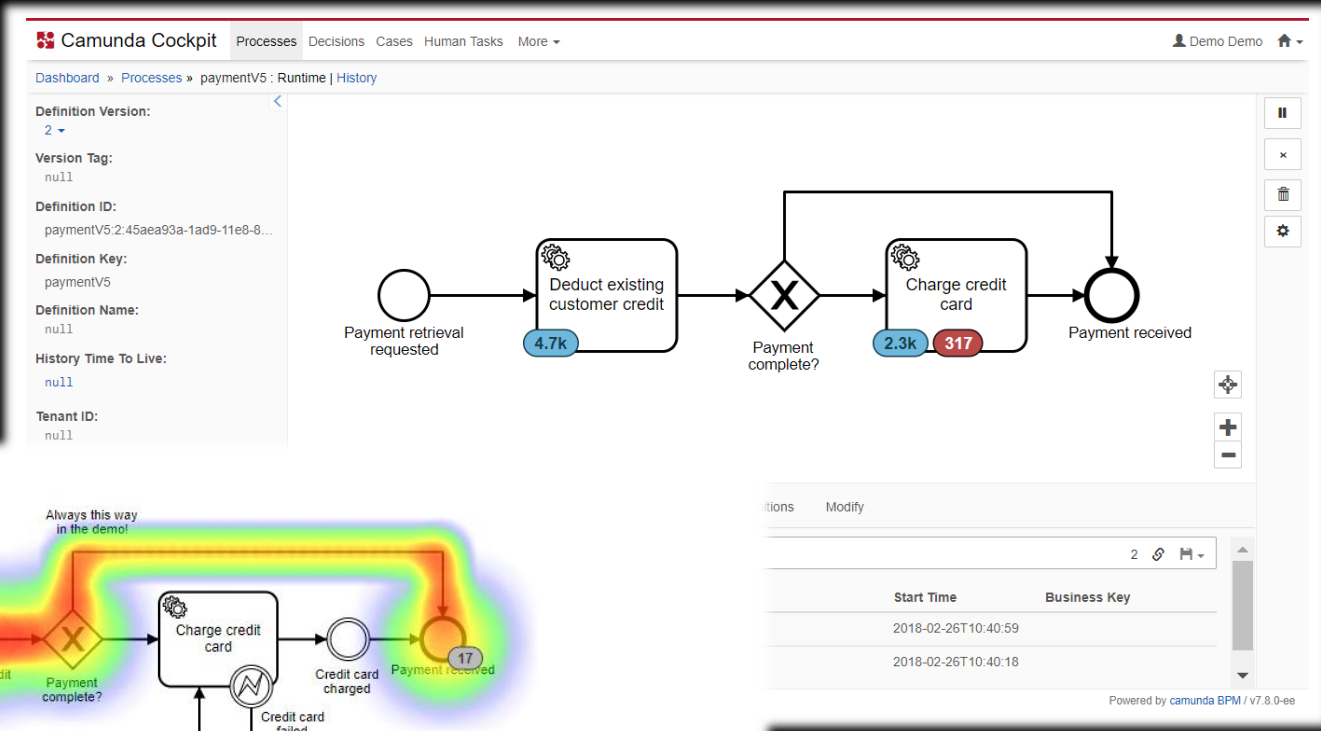
https://read.acloud.guru/how-the-saga-pattern-manages-failures-with-aws-lambda-and-step-functions-bc8f7129f900

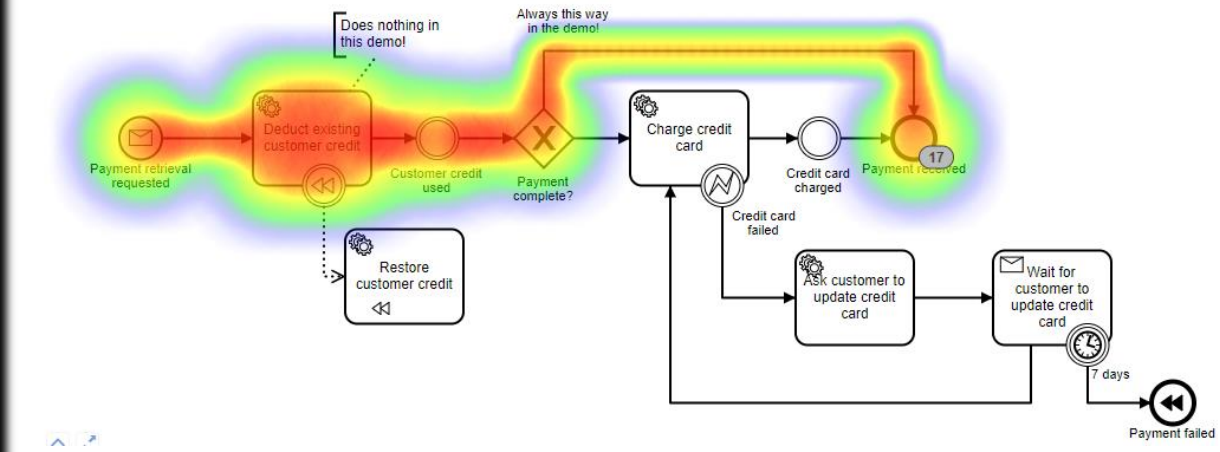
Flow language is important!
Think of more complicated scenarios...

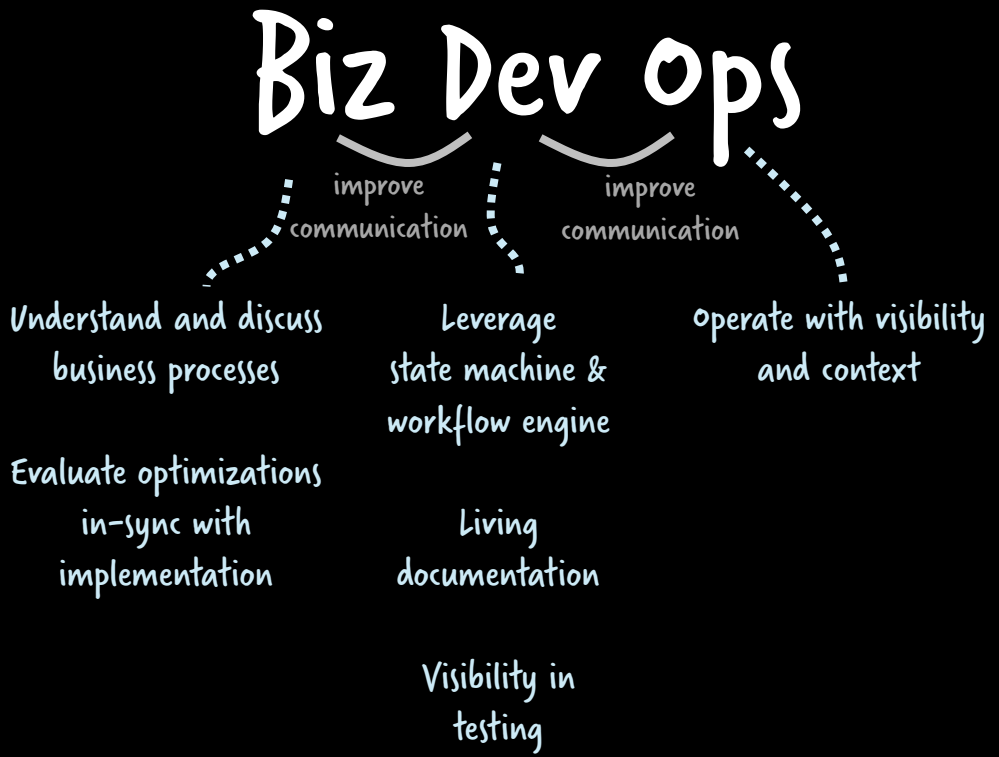


Proper operations

Visibility + Context






Biz Dev Ops
improve communication
improve communication
Understand and discuss business processes
Evaluate optimizations in-sync with implementation
Leverage state machine & workflow engine
Living documentation
Visibility in testing
Operate with visibility and context

Example: Storage

camunda

Spring Statemachine

zeebe by Camunda

Persistent State

28. Persisting State Machine

Traditionally an instance of a state machine is used as is within a running program.

28.2 Using StateMachinePersister

Building a `StateMachineContext` and then restoring a state machine from it has always been a little bit of a black magic if d

`StateMachinePersister` aims to ease these operations by providing *persist* and *restore* methods. Default implementation c

`DefaultStateMachinePersister`

Usage of a `StateMachinePersister` is easy to demonstrate by following a snippets from tests. We start by creating to two s

`machine1` and `machine2`. We could build different machines for this demonstration using various other ways but this serves

```
static class InMemoryStateMachinePersist implements StateMachinePersist<String, String, String> {

    private final HashMap<String, StateMachineContext<String, String>> contexts = new HashMap<>();

    @Override
    public void write(StateMachineContext<String, String> context, String contextObj) throws Exception {
        contexts.put(contextObj, context);
    }

    @Override
    public StateMachineContext<String, String> read(String contextObj) throws Exception {
        return contexts.get(contextObj);
    }
}
```


Source: Spring StateMachine docs

Example: Storage

camunda

Spring Statemachine

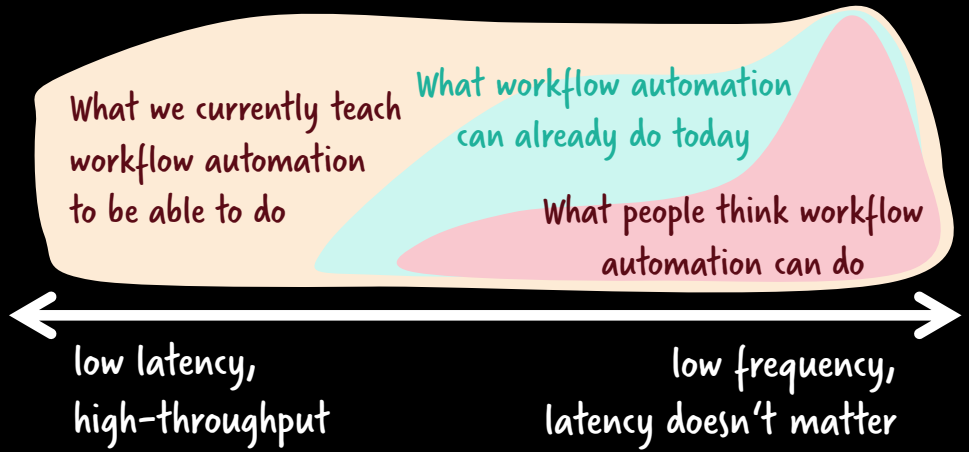
zeebe by Camunda

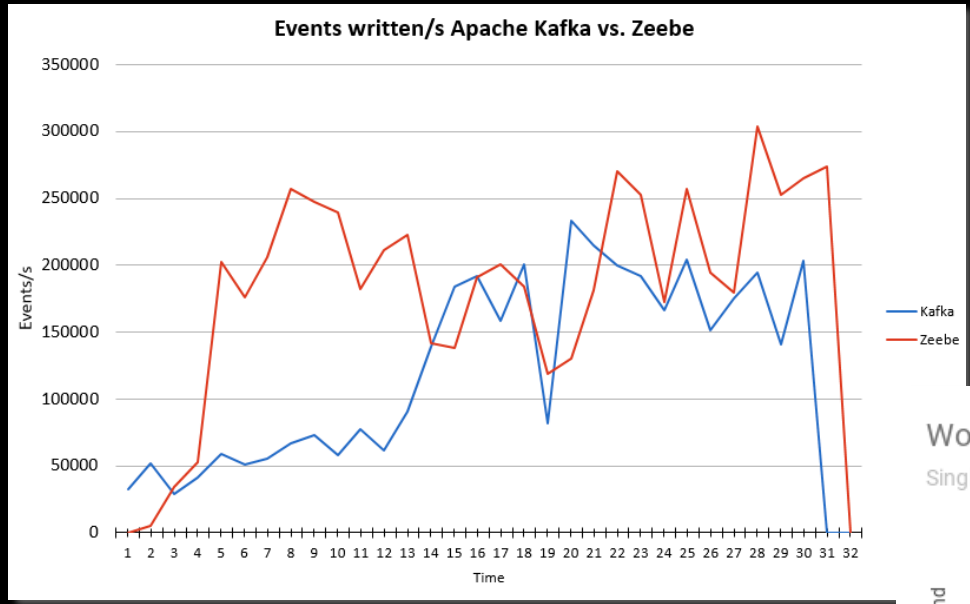
Persistent State

Do it yourself

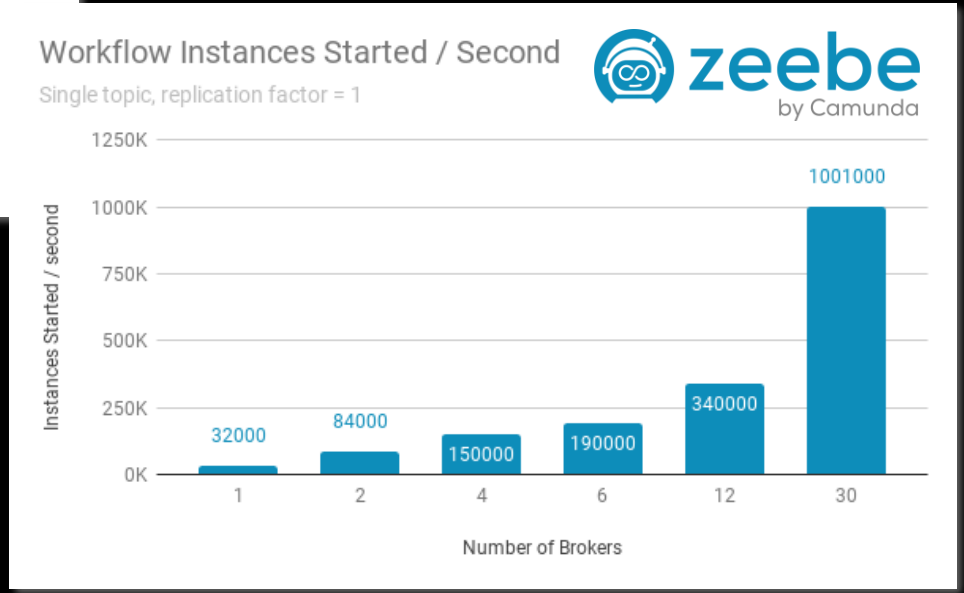
Persistent change

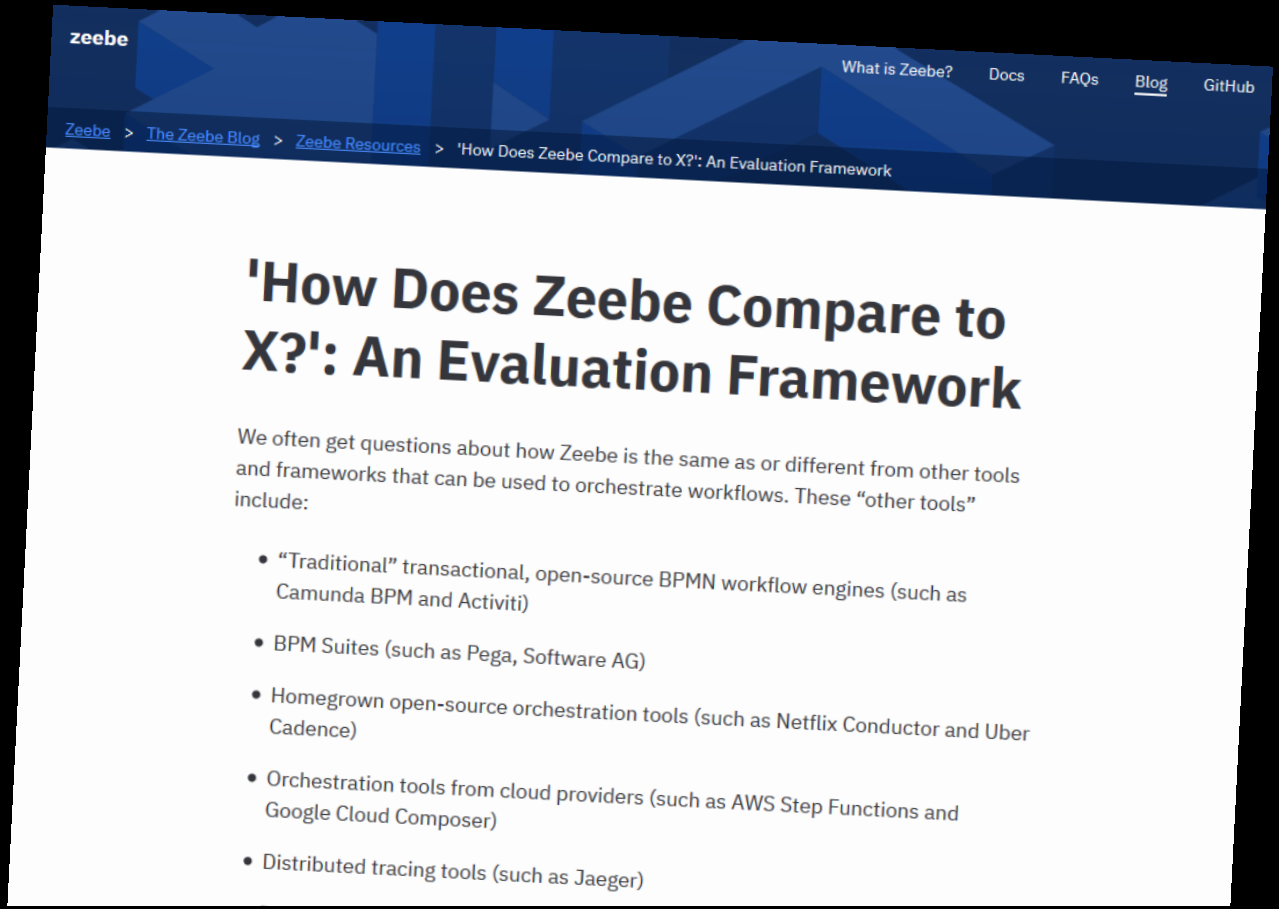
Workflow automation at scale!





Why Zeebe?
Horizontally scalable and resilient





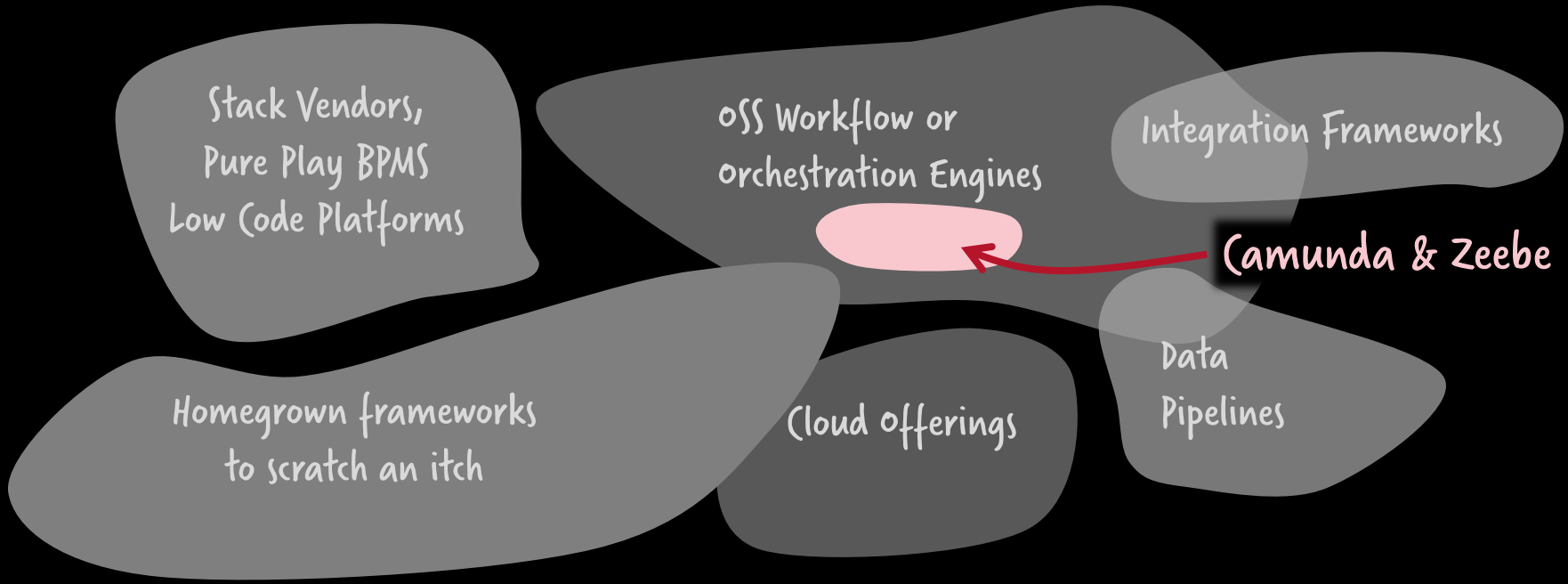
https://zeebe.io/blog/2019/03/zeebe-comparison-evaluation-criteria/

Aspects to consider

	Camunda	Spring State Machine
Workflow Definition	BPMN 2.0 (graphical, XML, Model API)	Java API, UML-Generator
Visual?	YES	No
Tooling	Modeler, Cockpit, Optimize	-
Storage Runtime	RDMS	up to you
Storage History	RDMS	-
Scalability	Stateless Engine, RDMS is limit, Sharding possible	up to you
Fault tolerance	If RDMS is HA	up to you
Supported programming languages	Java, REST, Language Clients (JS, C#)	Java

• • •

My personal pro-tip for a shortlist ;-)



Thank you!

Contact: mail@berndruecker.io
@berndruecker

Slides: https://berndruecker.io

Blog: https://medium.com/berndruecker

Code: https://github.com/berndruecker

InfoWorld FROM IDG

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

THENEWSTACK

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