How Process Orchestration Increases Agility Without Harming Architecture

@berndruecker

The common reality

Manual work
Ad-hoc problem solving

Legacy Systems

Website

Core Banking System

CRM System

Scoring System

Address Check

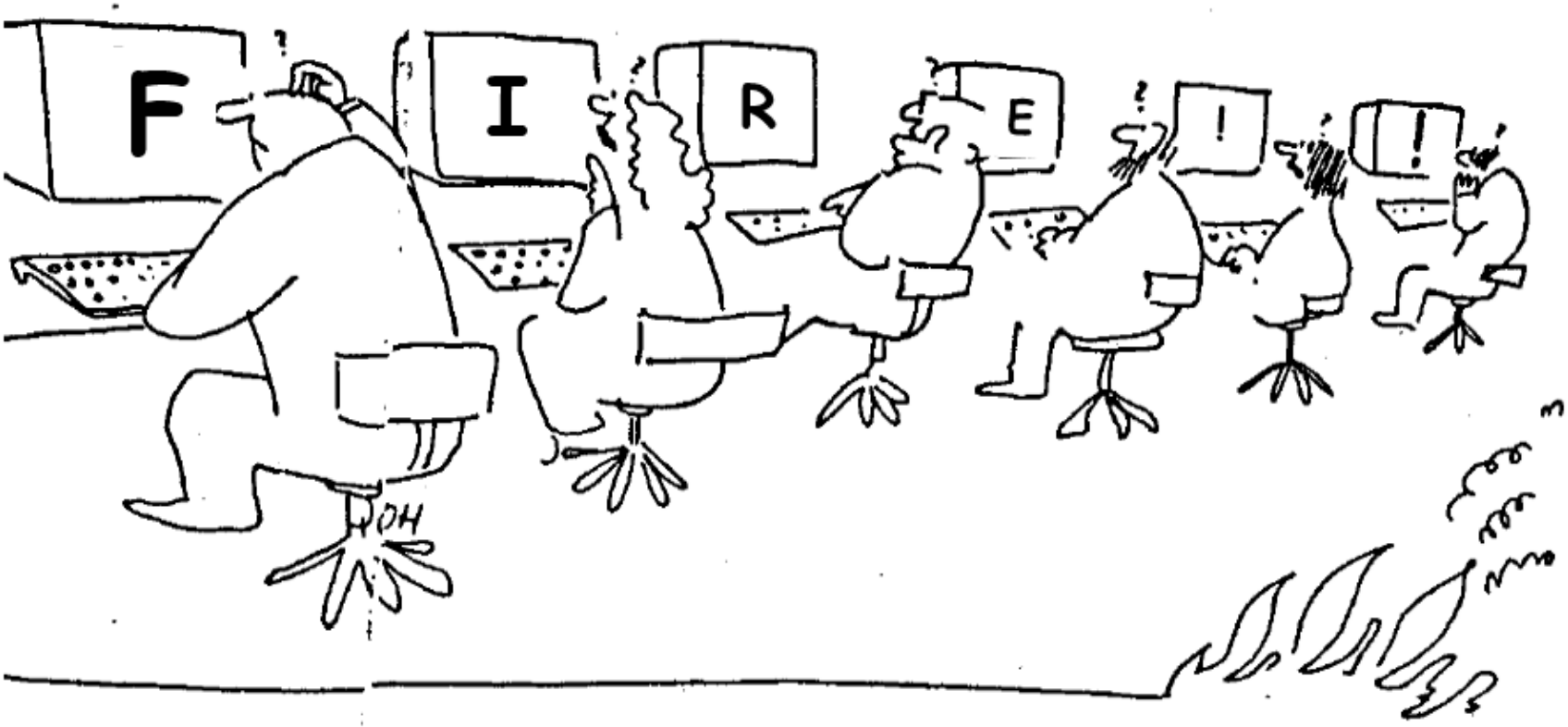
...

Point to point integrations (aka Spaghetti)

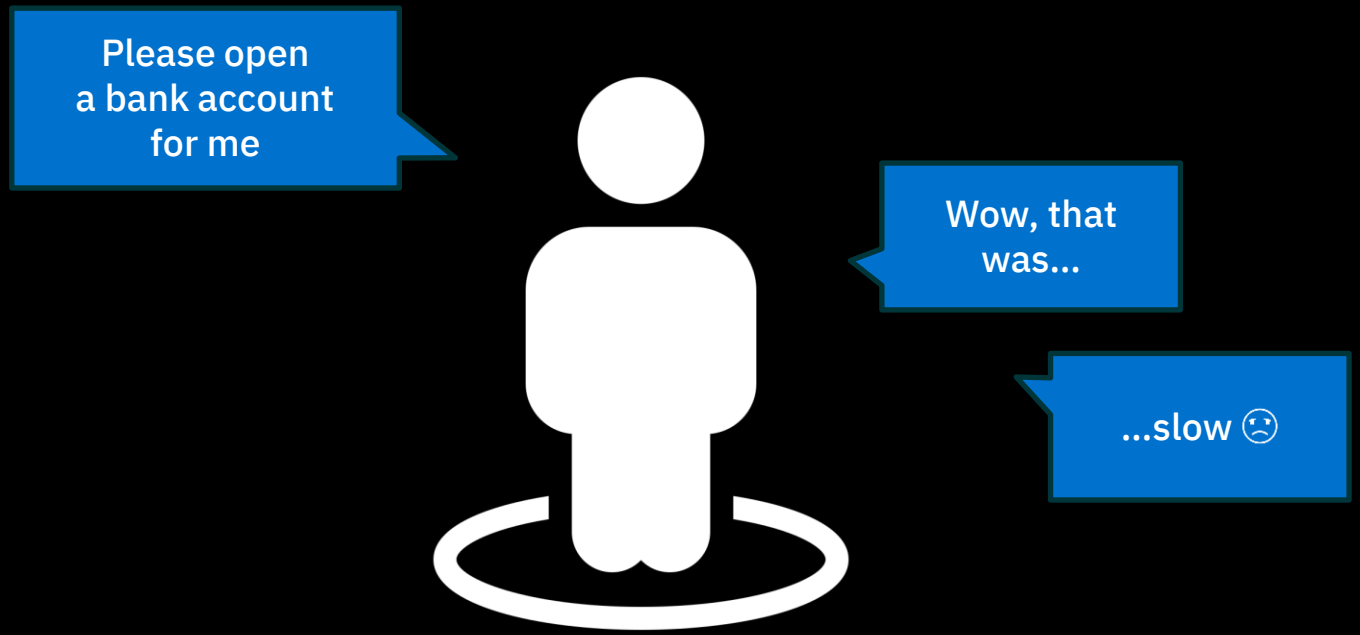
Processes span across Silos

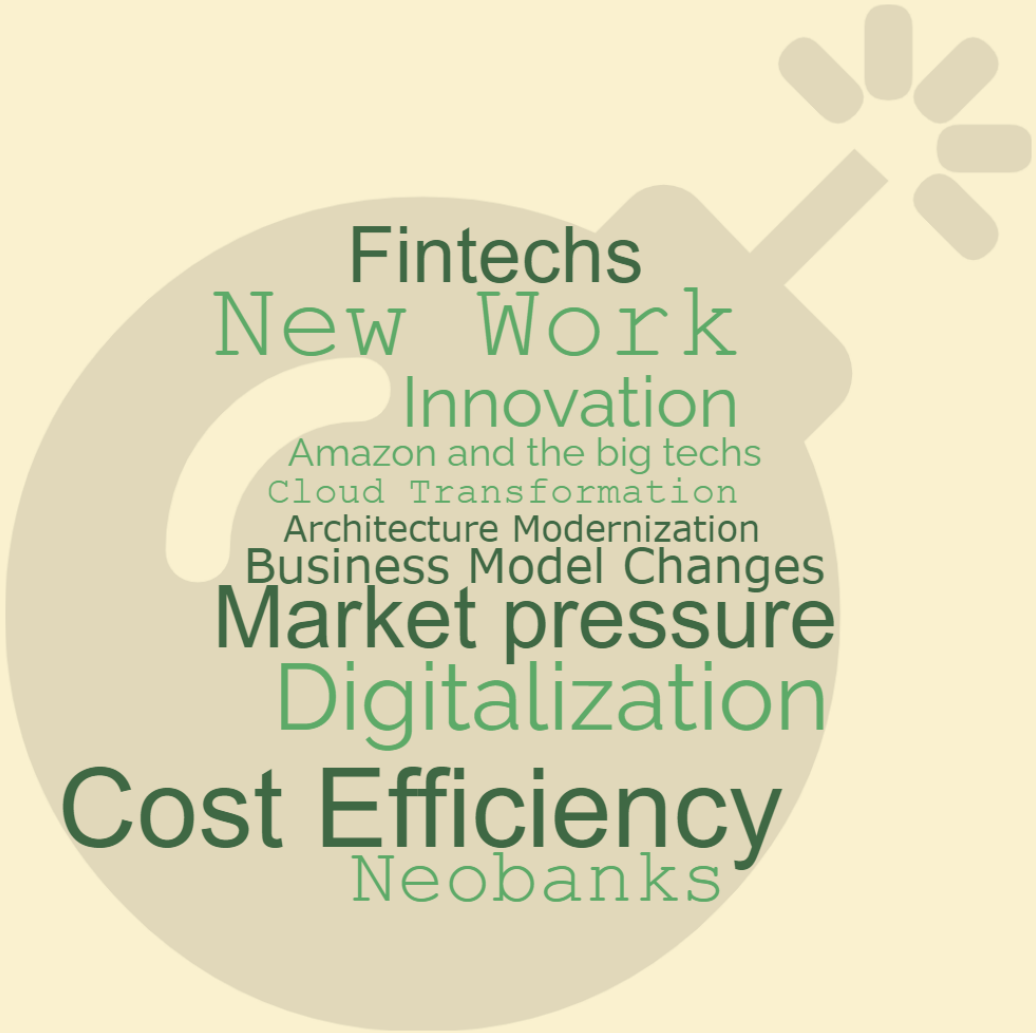
According to the 2023 State of Process Orchestration report, 72% of organizations found that their real-world, **mission critical processes were becoming more complex** to maintain.

The top reason for process complexity was having to **span multiple systems**.


F
I
R
E
!
!
OH

The customer doesn’t care




Fintechs
New Work
Innovation
Amazon and the big techs
Cloud Transformation
Architecture Modernization
Business Model Changes
Market pressure
Digitalization
Cost Efficiency
Neobanks

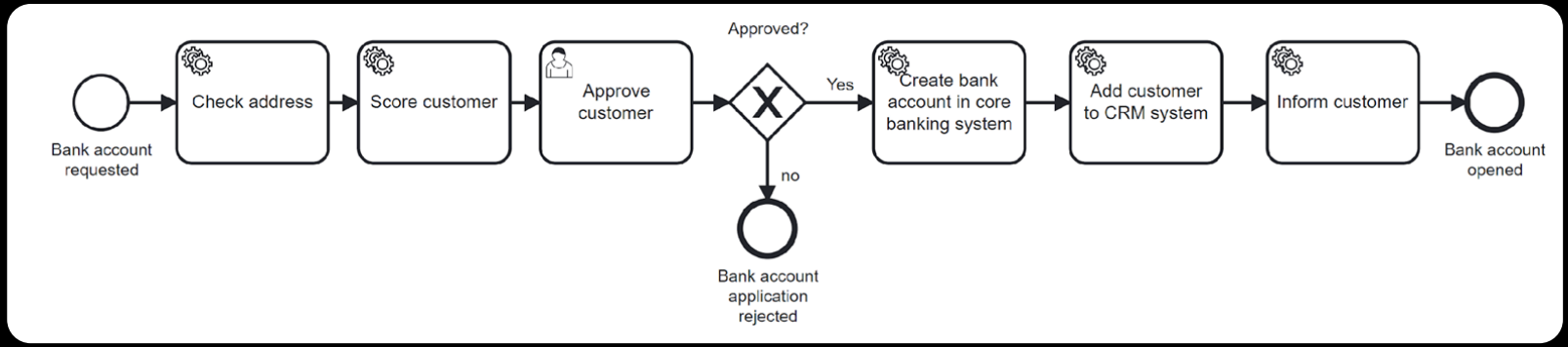
Process orchestration

Fintechs
New Work
Innovation
Amazon and the big techs
Cloud Transformation
Architecture Modernization
Business Model Changes
Market pressure
Digitalization
Cost Efficiency
Neobanks

Process orchestration

Clerks

Website



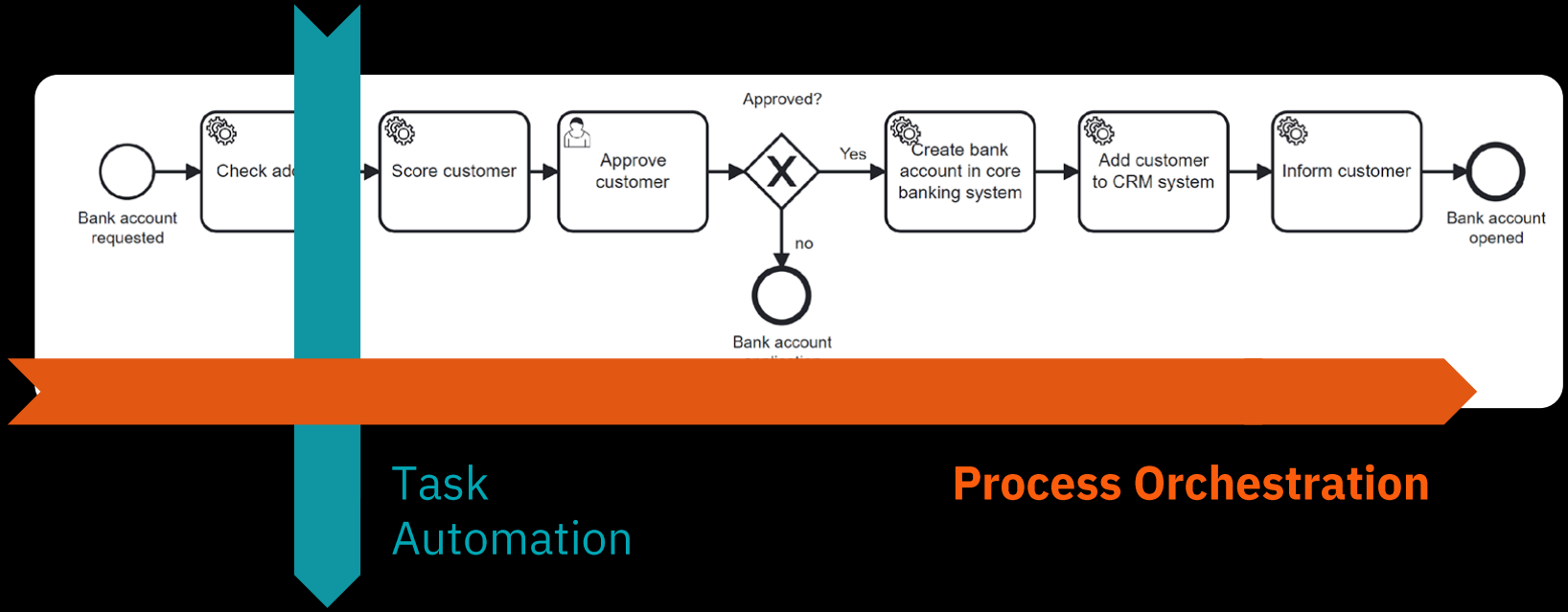
Core Banking System

CRM System

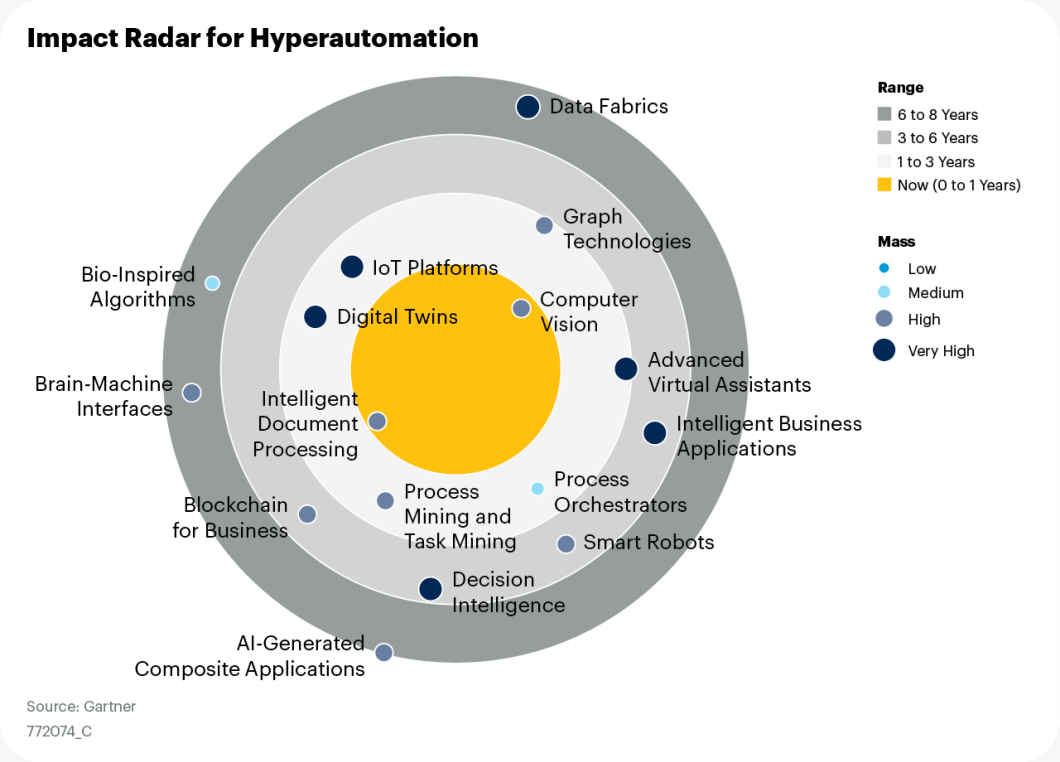
Scoring System

Address Check

Process automation:
Task automation vs. process orchestration



Gartner - Process Orchestration



"Process orchestration is critical both to manage end-to-end customer journeys and to provide consistency of experience to the human workforce."

— Gartner®

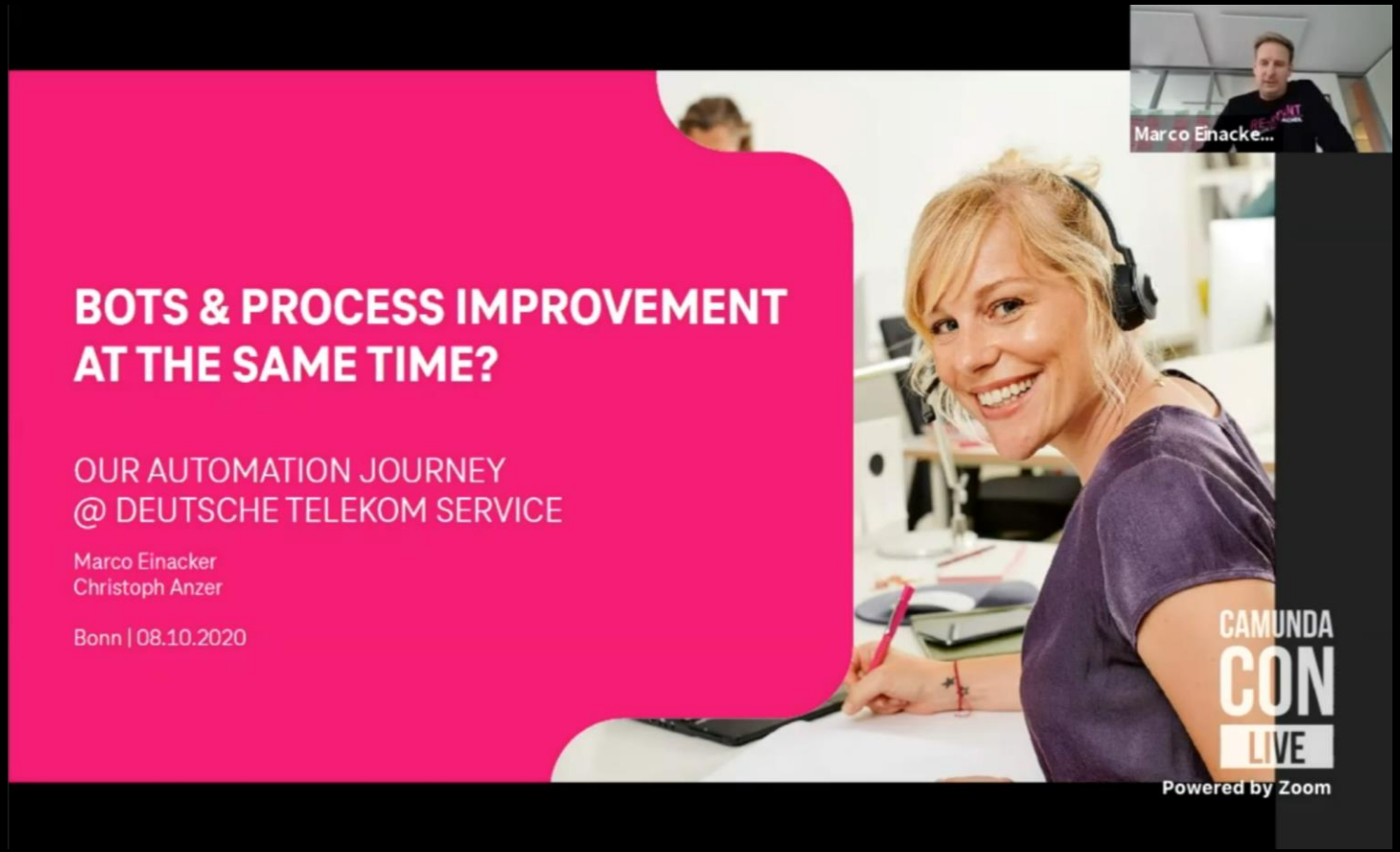
Why Process Orchestration Maturity matters

Organizations who are *not* implementing process orchestration across these silos often experience:

- ❌ Broken or inefficient customer experiences
- ❌ Unnecessary inefficiency due to poorly identified, implemented, executed, and maintained processes
- ❌ An inability to measure effectiveness or continuously improve automated processes

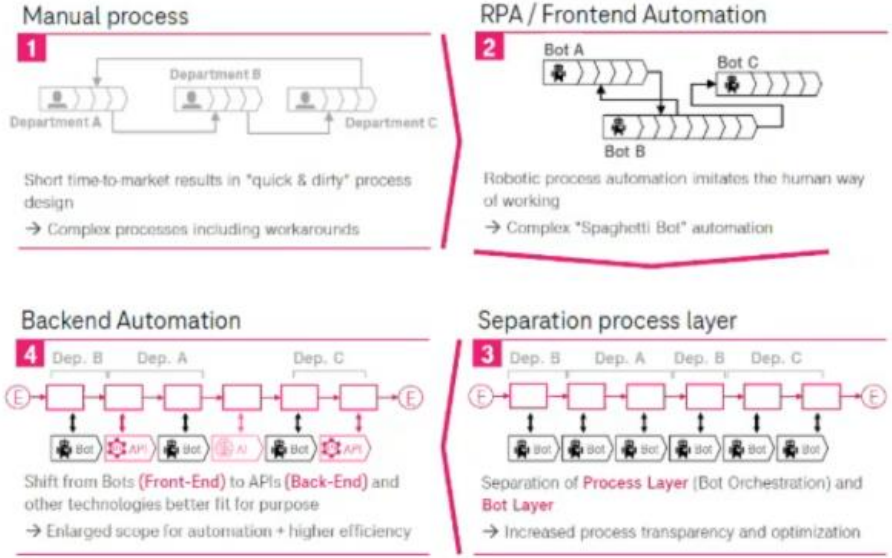
Organizations that are highly mature in their process orchestration:

- Marked improvements in customer experience, driving revenue opportunity
- Greater internal efficiency, lowering costs
- A higher degree of overall automation, driving digital transformation objectives



Source: Talk at Camunda Con 2020 (https://blog.bernd-ruecker.com/process-automation-in-harmony-with-rpa-720effdb0513)

3: FROM FRONTEND AUTOMATION TO BACKEND AUTOMTATION



"Spaghetti bot automation"

Source: Talk at Camunda Con 2020 (https://blog.bernd-ruecker.com/process-automation-in-harmony-with-rpa-720effdb0513)

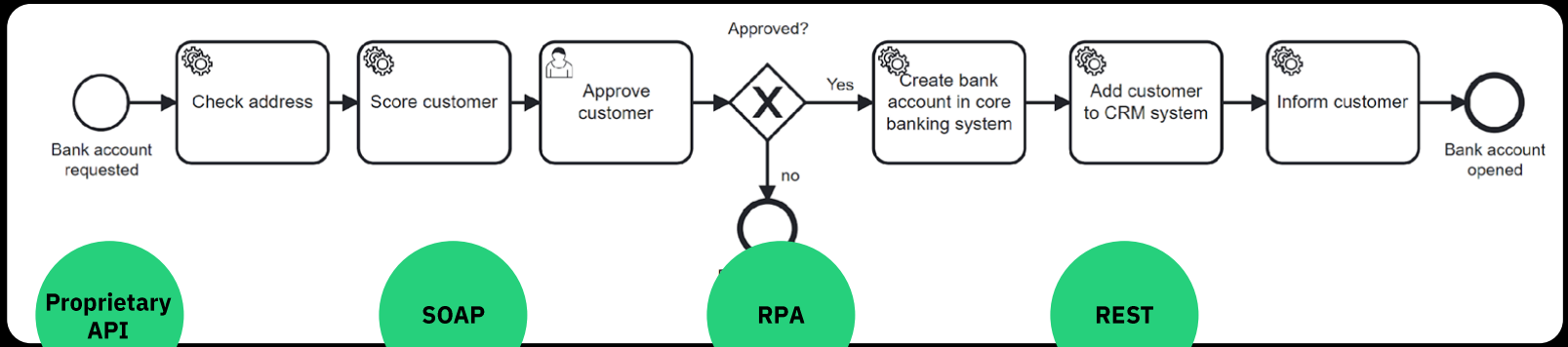
Integrating diverse endpoints

Clerks

User Interface

Messaging

Website



Proprietary API

SOAP

RPA

REST

Core Banking System

CRM System

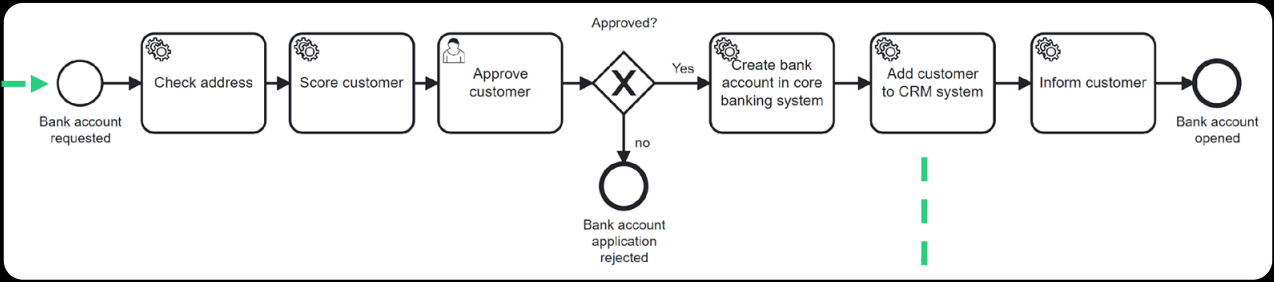
Scoring System

Address Check

```java
@PutMapping("/customer")
public ResponseEntity<CustomerOnboardingResponse> onboardCustomer(ServerWebExchange exchange) {
  HashMap<String, Object> variables = new HashMap<~>();
  variables.put("automaticProcessing", true);
  variables.put("someInput", "yeah");

  client.newCreateInstanceCommand() //
        .bpmnProcessId("customer-onboarding") //
        .latestVersion() //
        .variables(variables) //
        .send().join();
  return ResponseEntity.status(HttpStatus.ACCEPTED).build();
}
```

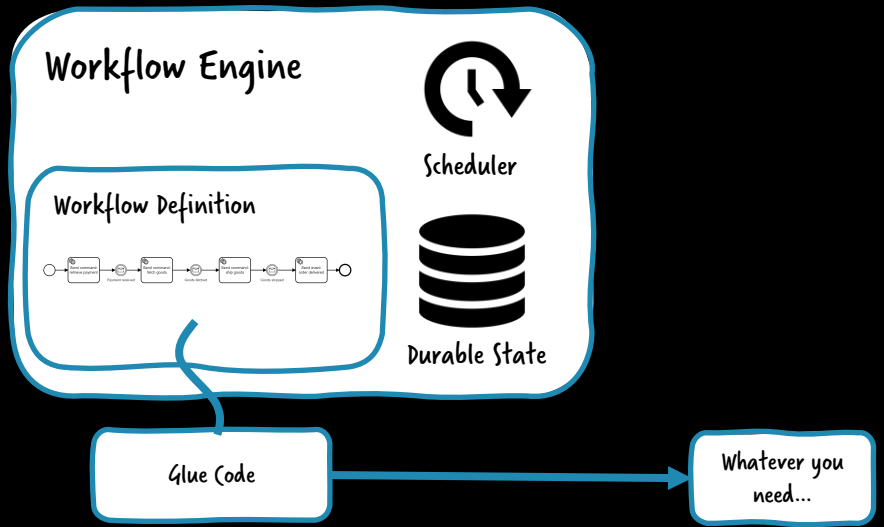
Your code to provide a REST endpoint



Your code to implement the REST call

```java
@ZeebeWorker(type = "addCustomerToCrm", autoComplete = true)
public void addCustomerToCrmViaREST(final ActivatedJob job) throws IOException {
    String request = "someData";
    restTemplate.put(ENDPOINT_CRM, request);
}
```

Using a workflow engine



Workflow Engine:

Is stateful

Can wait
Can retry
Can escalate
Can compensate

Provides visibility

Example

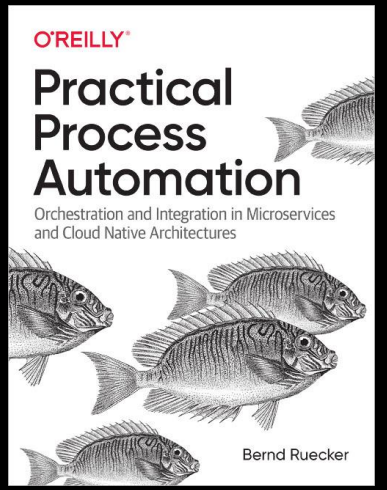
https://github.com/berndruecker/customer-onboarding-camunda-8-springboot/

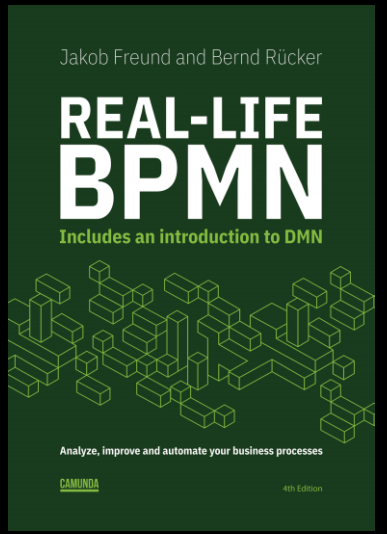
Warning:
Contains Opinion

Bernd Ruecker

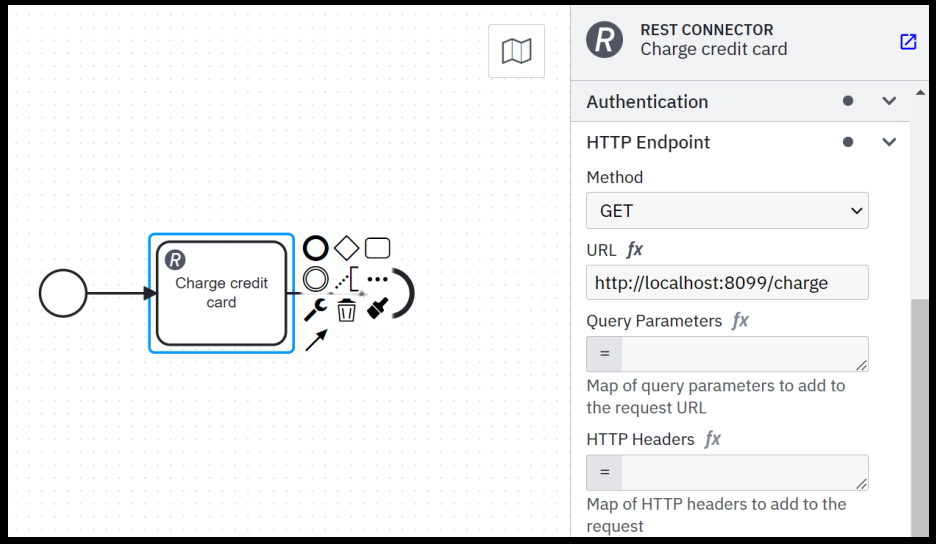
Co-founder and
Chief Technologist of
Camunda

mail@berndruecker.io
@berndruecker
http://berndruecker.io/





Do you have to program all integration logic?

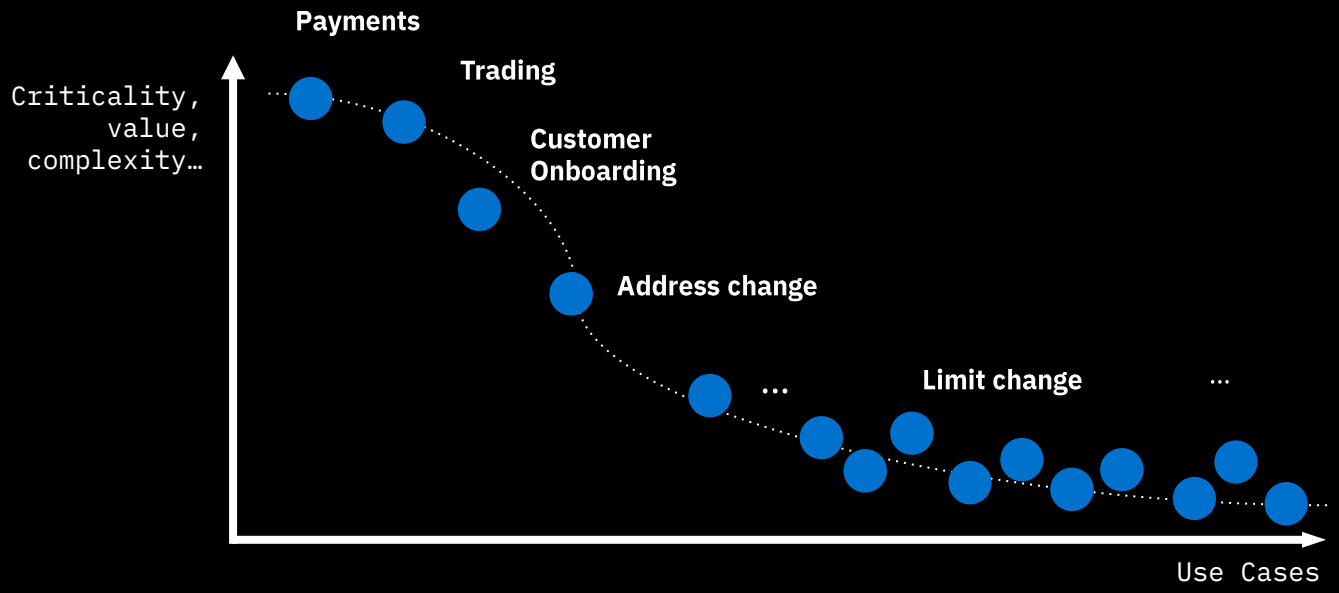


Out-of-the-box Connectors

- REST API
- SendGrid
- Slack
- Microsoft Teams
- Google Drive
- Automation Anywhere
- Kafka Producer
- Amazon SQS
- AWS Lambda
- Amazon SNS
- RabbitMQ
- GraphQL
- GitLab
- OpenAI
- Camunda Operate
- MessageBird
- Twilio
- EasyPost
- GitHub
- Asana
- Google Maps
- UiPath
- Microsoft Power Automate
- ***... and more***

Low-code?

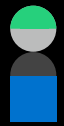
Process != process



Categorize your use case

Green
Do it yourself

- simple
- local automations with little criticality
- no governance or quality assurance



Citizen Developer

Yellow
Guided

- medium complexity
- medium criticality
- some governance required
- some guidance necessary

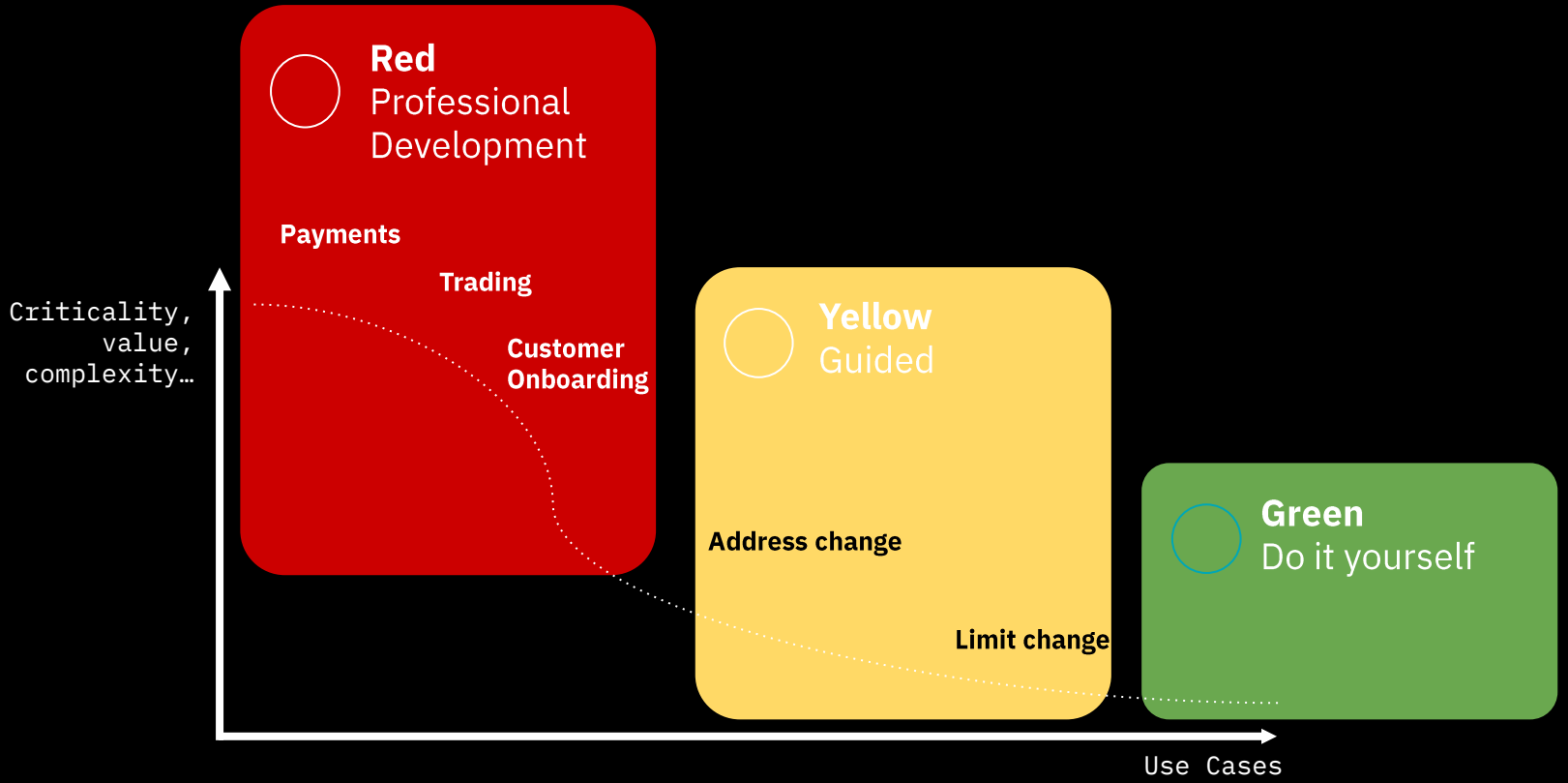
Anything in between

Red
Professional Development

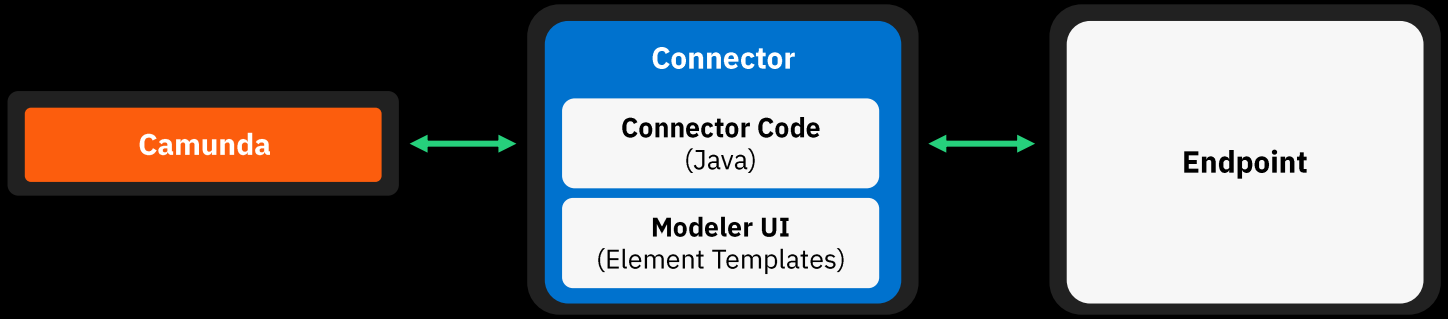
- high complexity
- high criticality
- compliance and regulatory requirements
- version control
- automated testing
- CI / CD

Professional Developer

Inspired also by https://www.forrester.com/blogs/how-shell-led-a-citizen-developer-movement/


Red
Professional
Development
Payments
Trading
Customer
Onboarding
Yellow
Guided
Address change
Limit change
Green
Do it yourself
Criticality,
value,
complexity…
Use Cases

What's a Connector?

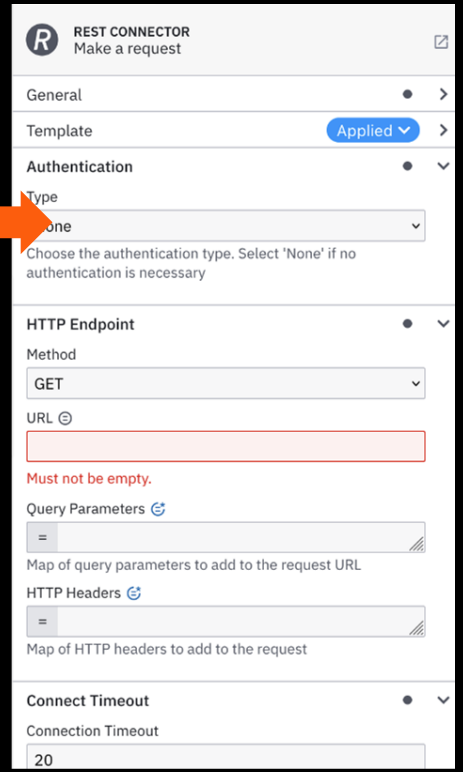


```java
@OutboundConnector(
     type = "io.camunda:http-json:1", inputVariables = {"method", ...},
)
public class HttpJsonFunction implements OutboundConnectorFunction {

  public Object execute(final OutboundConnectorContext context) throws Exception {
    final var json =;
    final var request = createRequest(context);
    return httpService.executeConnectorRequest(request);
  }
}
```

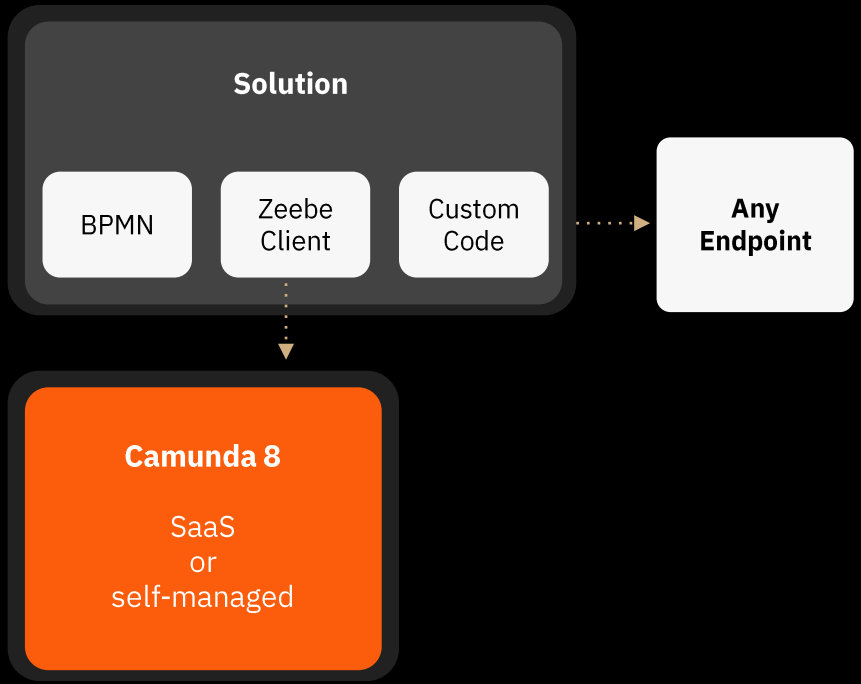
+

```
{
  "name": "REST Connector",
  "properties": [
    {
      "type": "Hidden",
      "value": "io.camunda:http-json:1",
      "binding": {
        "type": "zeebe:taskDefinition:type"
      }
    },
    {
      "id": "method",
      "label": "REST Method",
      "group": "endpoint",
      "type": "Dropdown",
      "value": "get",
      "choices": [
```

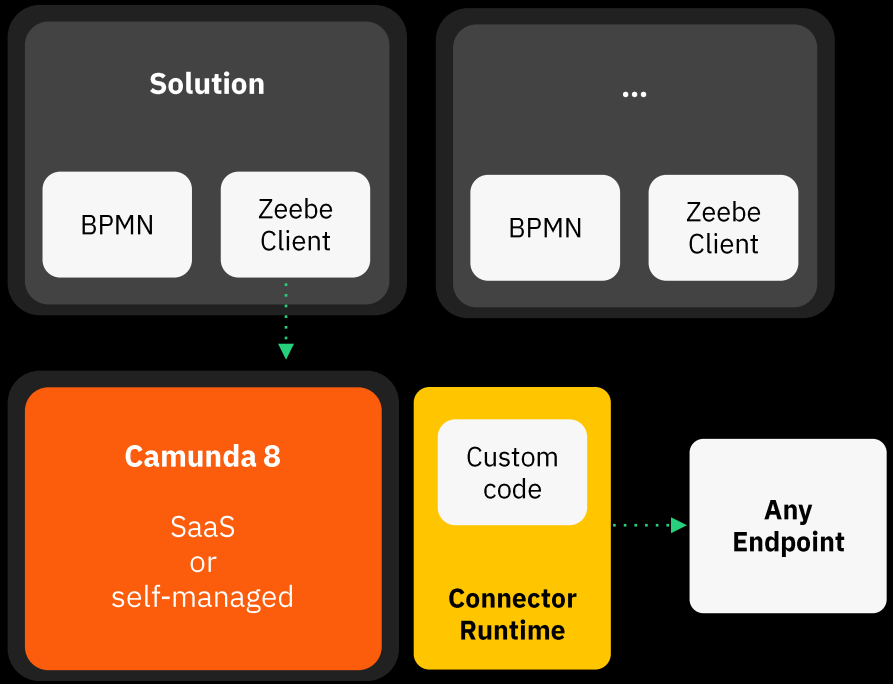


https://github.com/camunda/connectors-bundle/tree/main/connectors/http-json

Solution architecture example



Solution architecture example

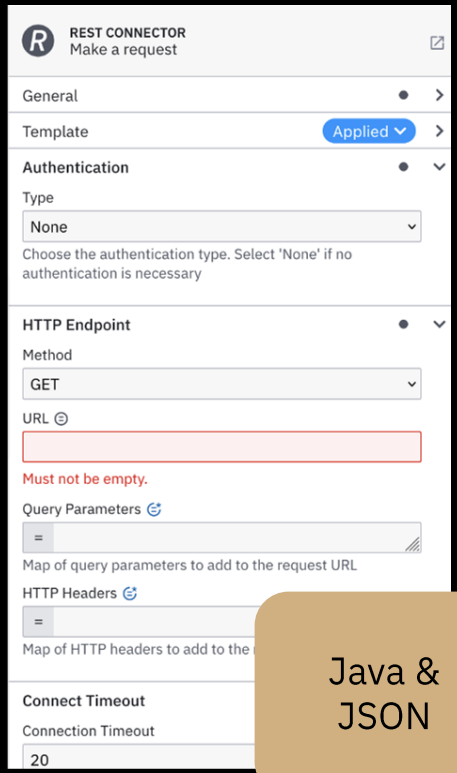


Runs anywhere

- Multiple connector runtime options, including SaaS, self-managed, co-located, and local installations

Protocol Connectors

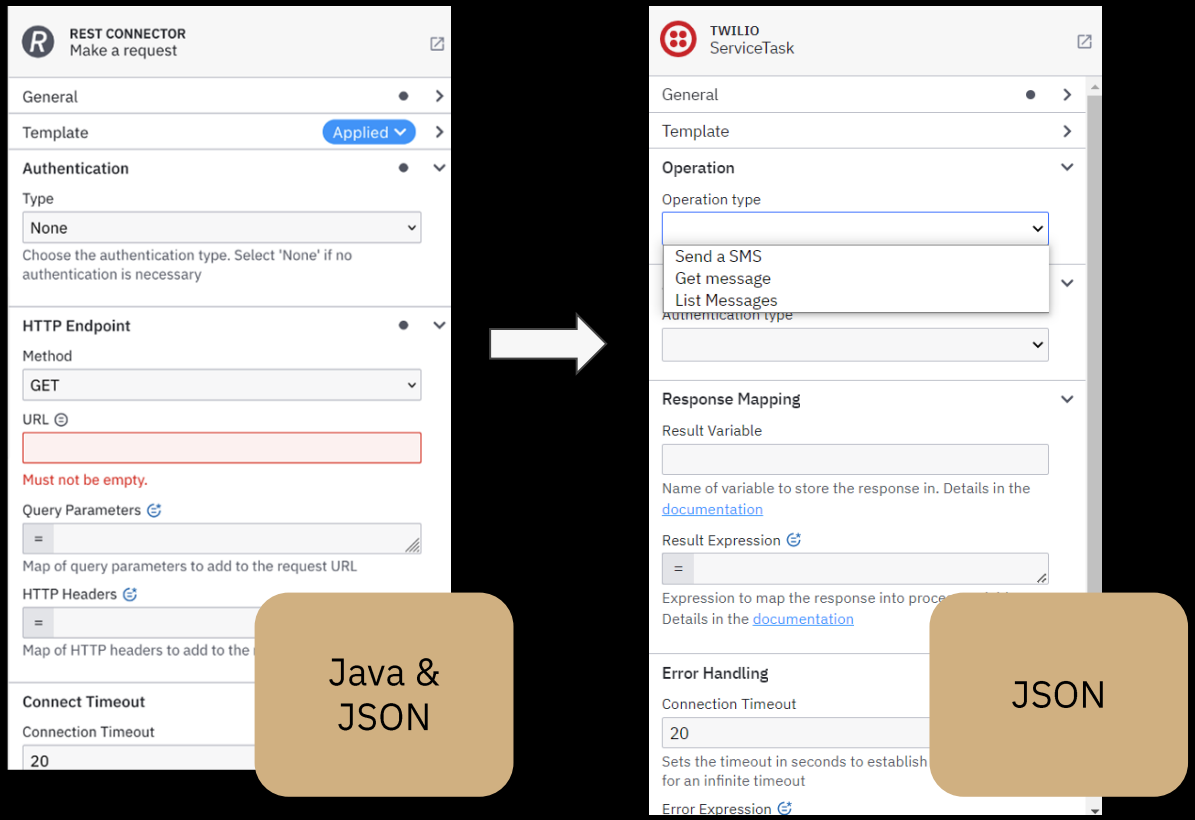
CAMUNDA



Java &
JSON

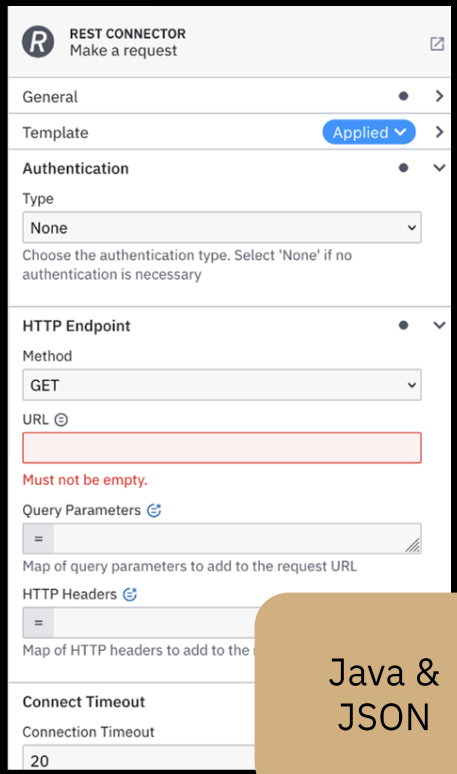
Protocol > Generic System Connector

CAMUNDA

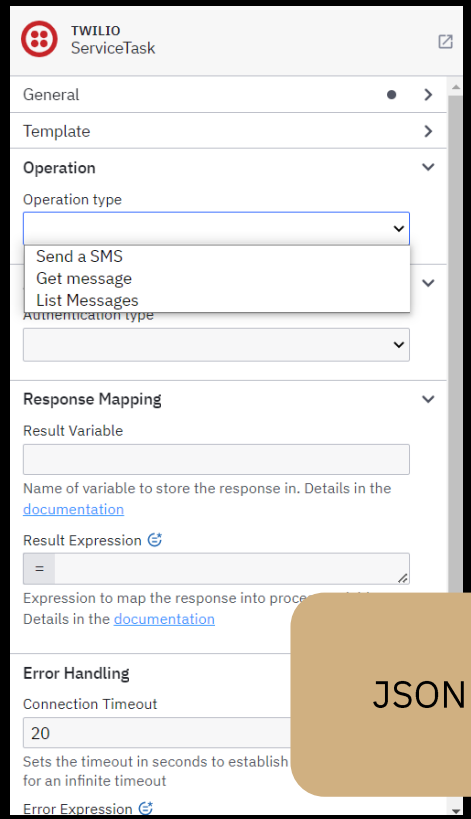


Protocol > Generic > Specific Connectors

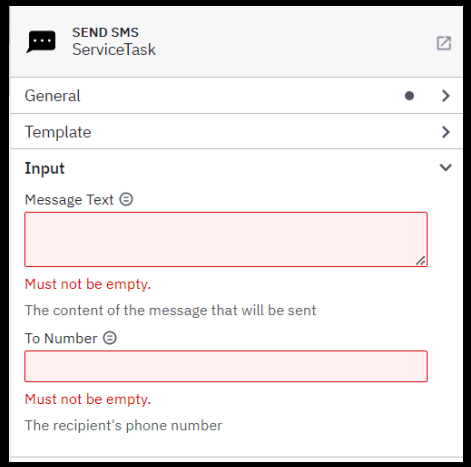
CAMUNDA



Java & JSON

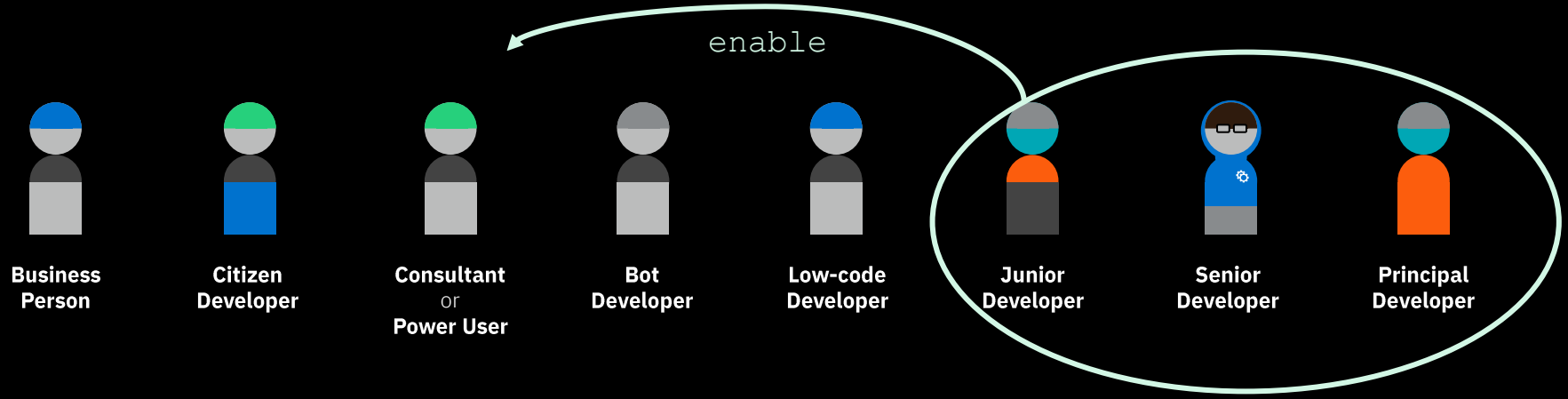


JSON

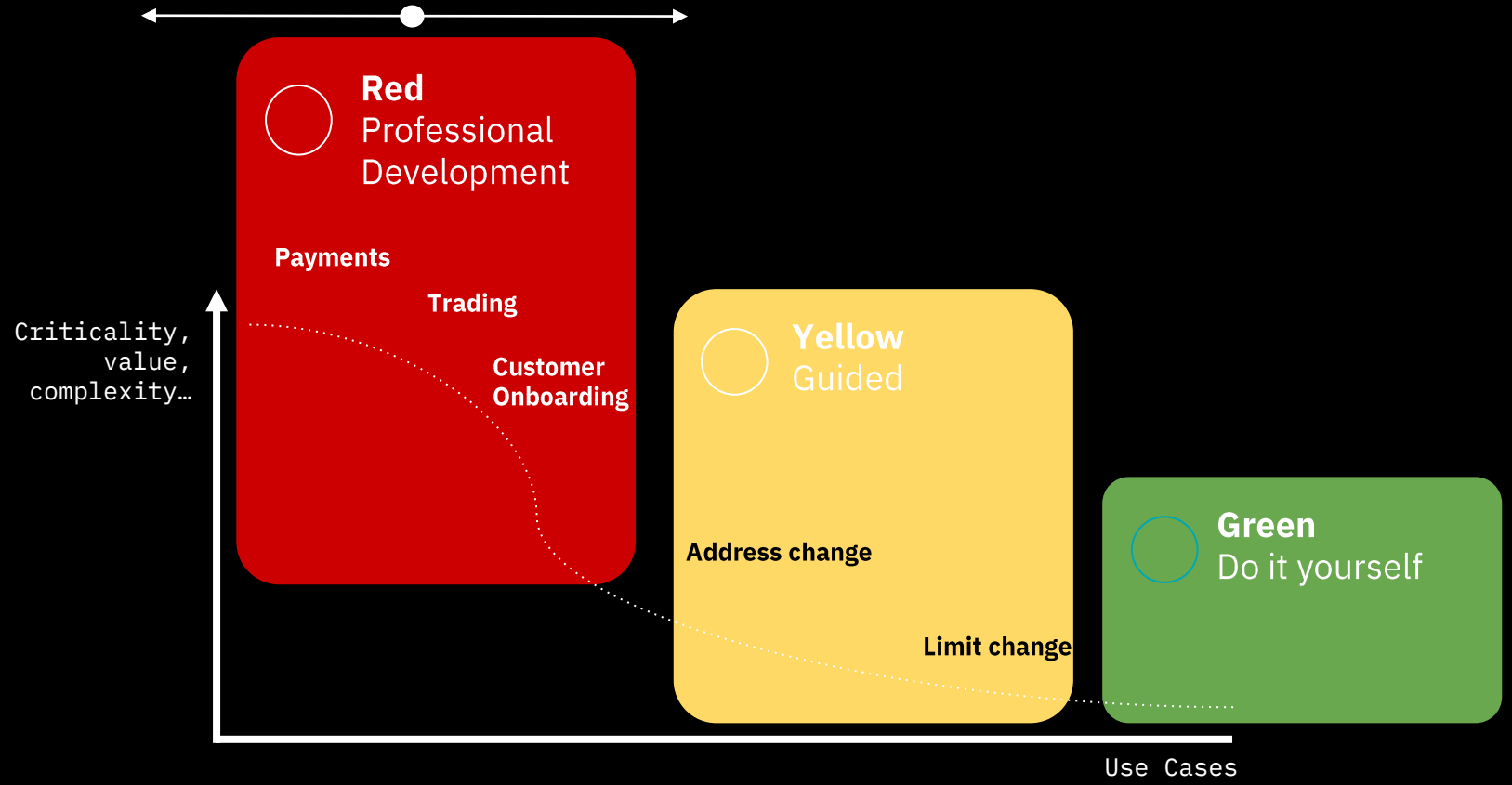


JSON

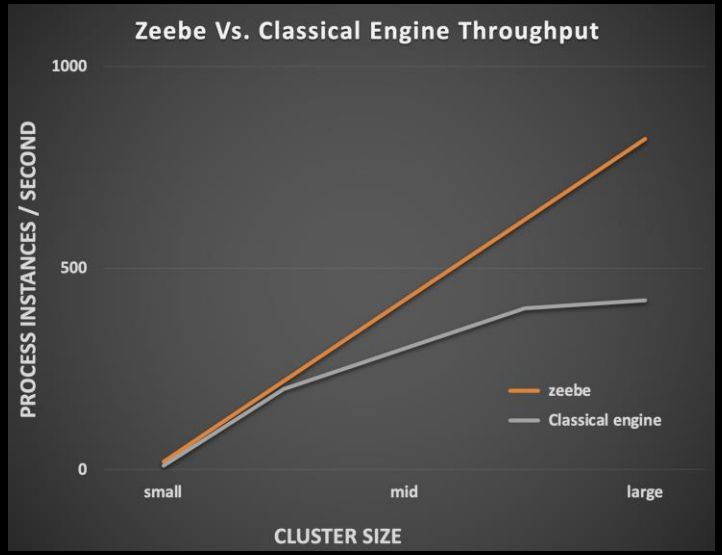
Enabling more roles to participate



Unlocking more use cases



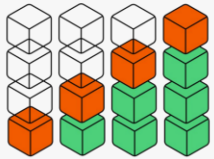
But also unlock more red use cases



CAMUNDA

Benchmark Performance with Camunda Platform's Zeebe Engine

February 2023



Workload Characteristics of Customers

Throughput (PI/s)	Process size (#tasks)	Latency (ms)	Multi-Region Setup
10,000	8 tasks	500 ms	active-passive east-west 60ms
500	3 tasks + 2 messages + 2 call activities	1,000 ms	active-active 10ms avg / 35ms max
2,400	10 tasks	1,200 ms	active-passive 52ms one way
1,700	10 tasks	120,000 ms	active-active-passive 2x east coast + 1x central
800	8 tasks	200 ms	active-passive 62ms
3,000	3 tasks	300 ms	single-region replication factor = 1

[Webinar] Benchmark Performance with Camunda Platform's Zeebe Engine

Low-code as an accelerator

Dial-in low-code as much as you need

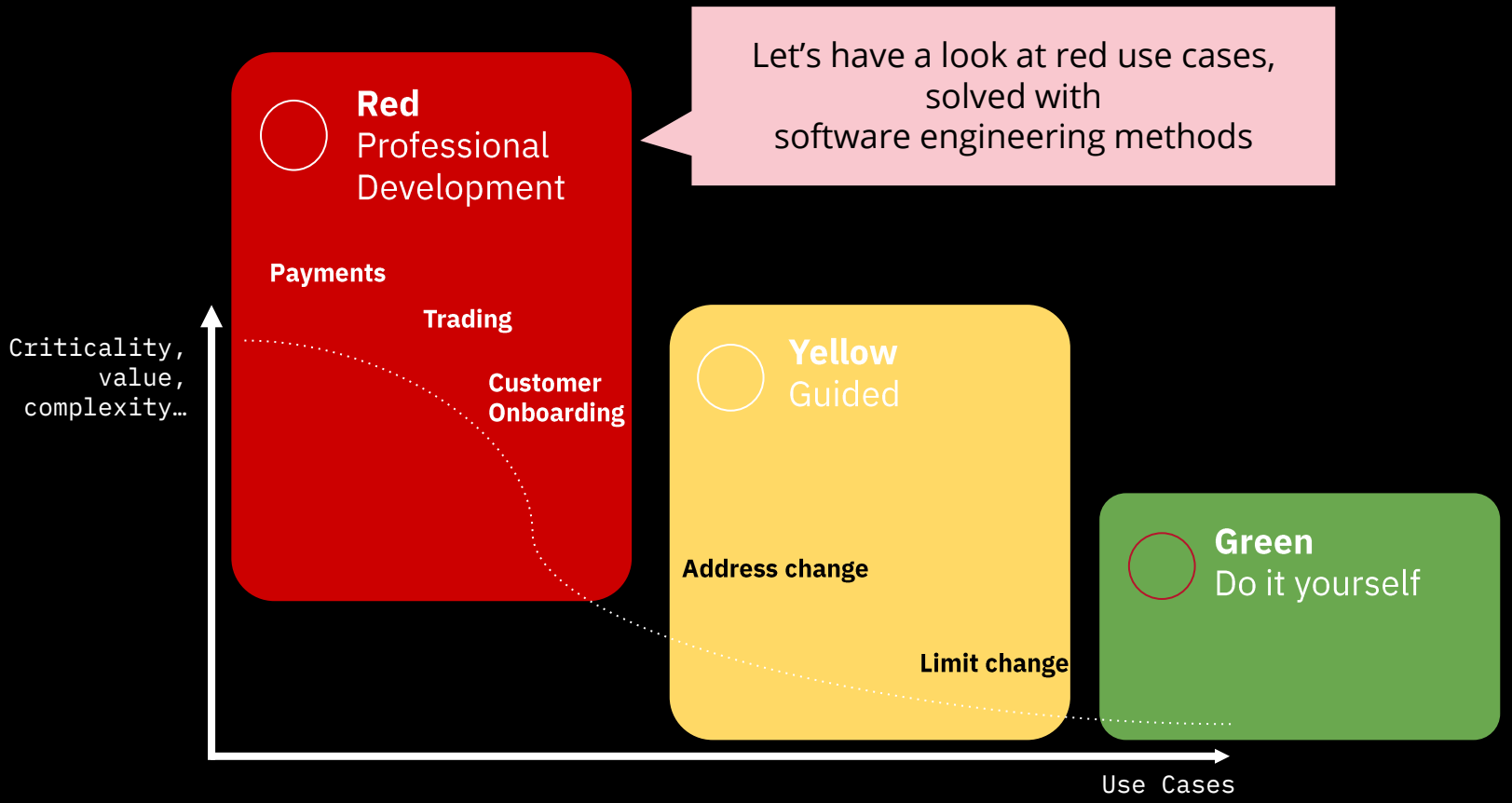
Camunda Recognized as A Strong Performer

The Forrester Wave™
Digital Process Automation Software, Q4 2023

Highest possible scores for the following criteria:

- End-to-end orchestration
- Data-driven automation
- Vision
- Innovation
- Adoption
- Pricing flexibility and transparency

Download the report:


Let’s have a look at red use cases,
solved with
software engineering methods
Red
Professional
Development
Payments
Trading
Customer
Onboarding
Yellow
Guided
Address change
Limit change
Green
Do it yourself
Criticality,
value,
complexity…
Use Cases

Recently in software engineering...

Recently in software engineering...

Microservices, Serverless, …

Domain Driven Design (DDD)

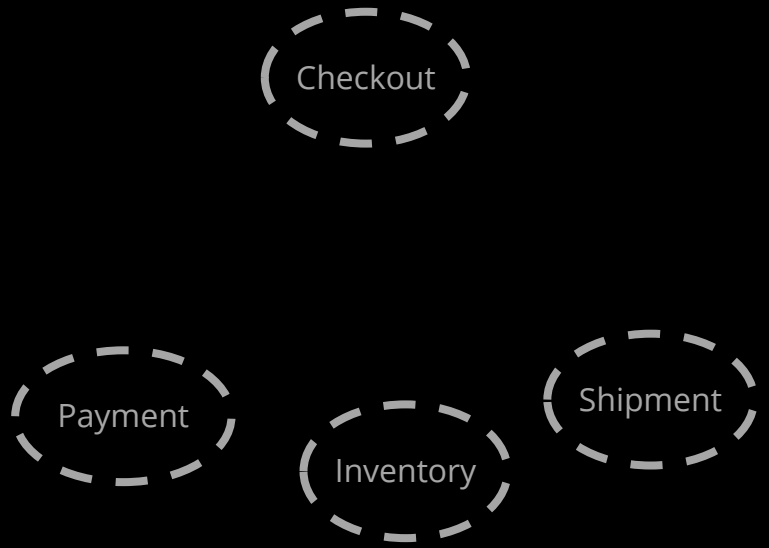
Cloud, SaaS, …

Product mindset

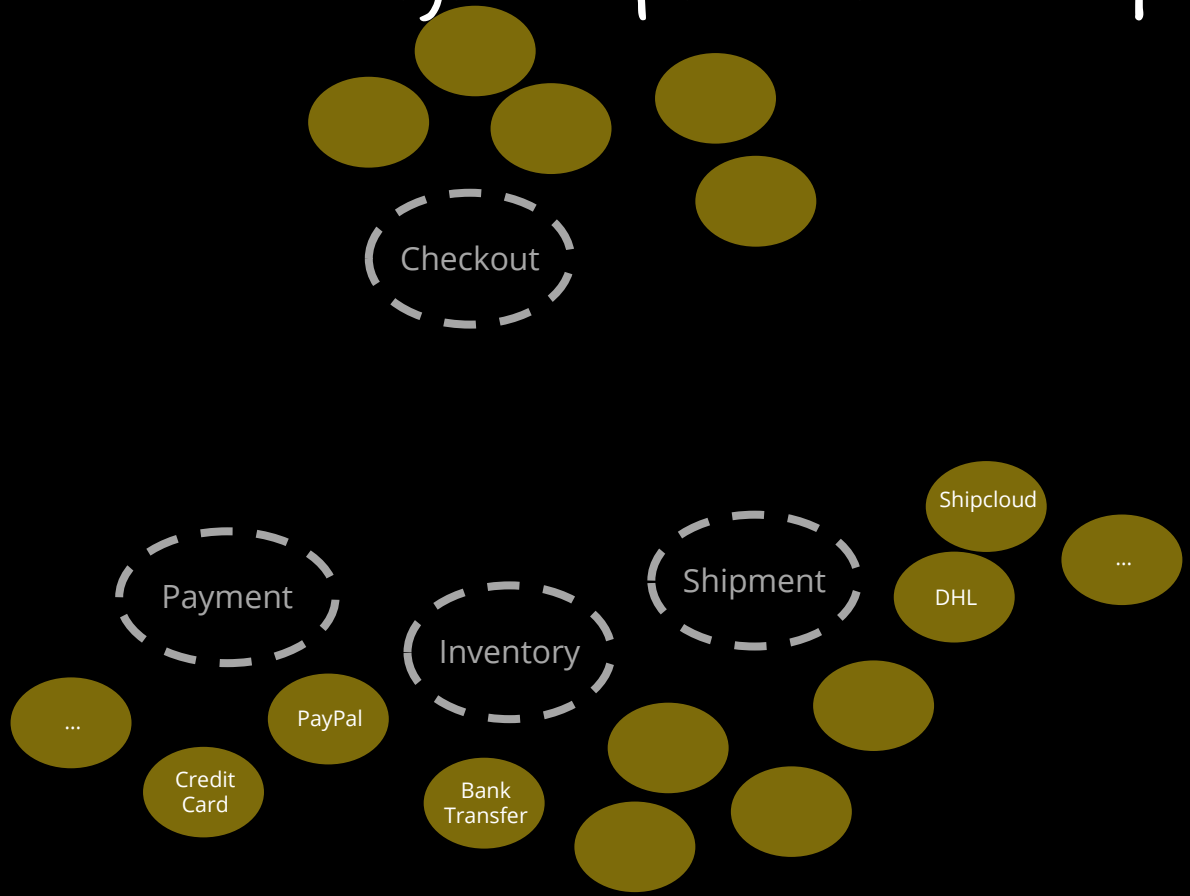
Agile, DevOps, …

Team topologies

Building smaller components...



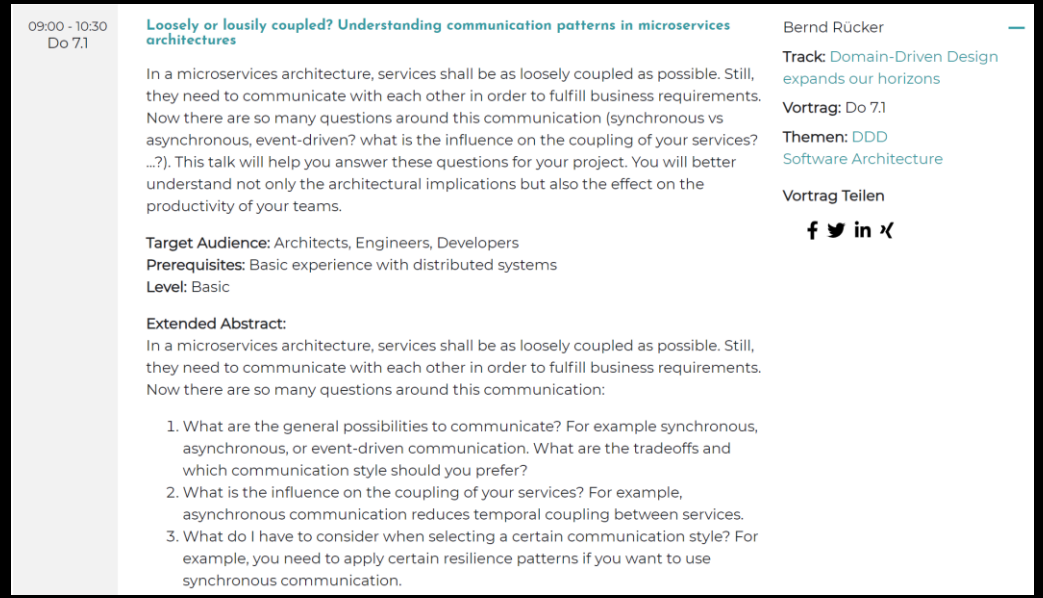
...with an ecosystem of services and components



Autonomy to speed up development

There is trouble in software engineering land!

Confusion around communication and collaboration styles
(REST, Messaging, Events)



09:00 - 10:30
Do 7.1

Loosely or lousily coupled? Understanding communication patterns in microservices architectures

In a microservices architecture, services shall be as loosely coupled as possible. Still, they need to communicate with each other in order to fulfill business requirements. Now there are so many questions around this communication (synchronous vs asynchronous, event-driven? what is the influence on the coupling of your services? ...?). This talk will help you answer these questions for your project. You will better understand not only the architectural implications but also the effect on the productivity of your teams.

Target Audience: Architects, Engineers, Developers
Prerequisites: Basic experience with distributed systems
Level: Basic

Extended Abstract:
In a microservices architecture, services shall be as loosely coupled as possible. Still, they need to communicate with each other in order to fulfill business requirements. Now there are so many questions around this communication:

1. What are the general possibilities to communicate? For example synchronous, asynchronous, or event-driven communication. What are the tradeoffs and which communication style should you prefer?
2. What is the influence on the coupling of your services? For example, asynchronous communication reduces temporal coupling between services.
3. What do I have to consider when selecting a certain communication style? For example, you need to apply certain resilience patterns if you want to use synchronous communication.

Bernd Rücker

Track: Domain-Driven Design expands our horizons

Vortrag: Do 7.1

Themen: DDD Software Architecture

Vortrag Teilen

There is trouble in software engineering land!

Confusion around communication and collaboration styles
(REST, Messaging, Events)

A wrong understanding of decoupling and the resulting event-driven chaos

Not thinking beyond Microservices

Missing out on (a healthy level of) centralization

There is trouble in software engineering land!

Confusion around communication and collaboration styles
(REST, Messaging, Events)

A wrong understanding of decoupling and the resulting event-driven chaos

Not thinking beyond Microservices

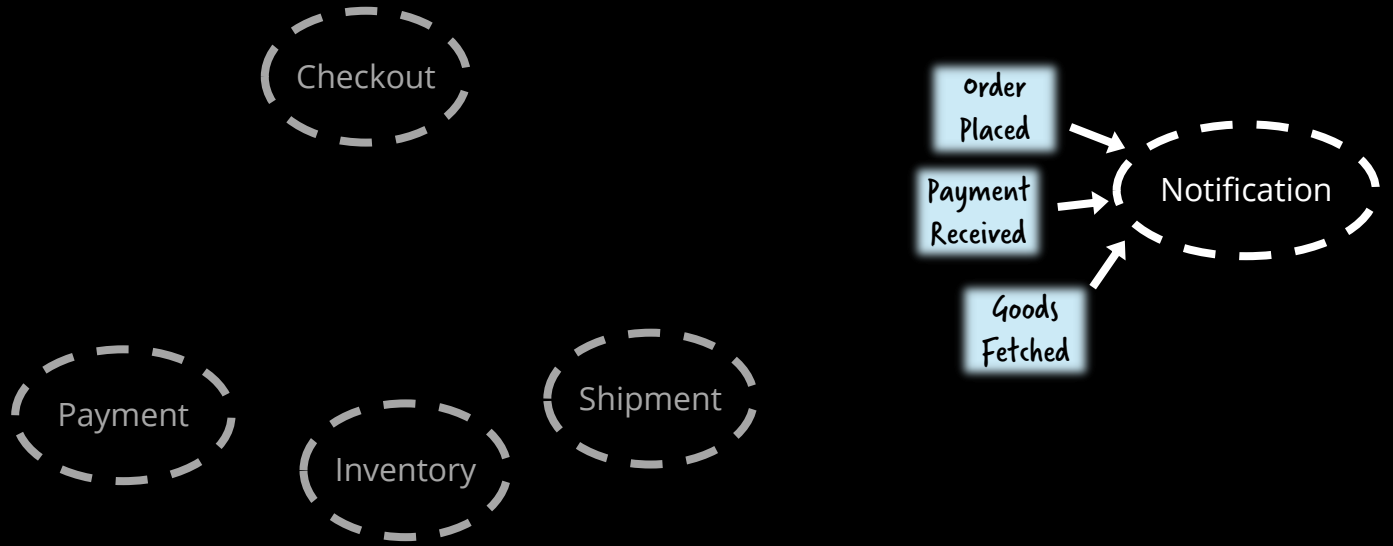
Missing out on (a healthy level of) centralization

Choreography is great!

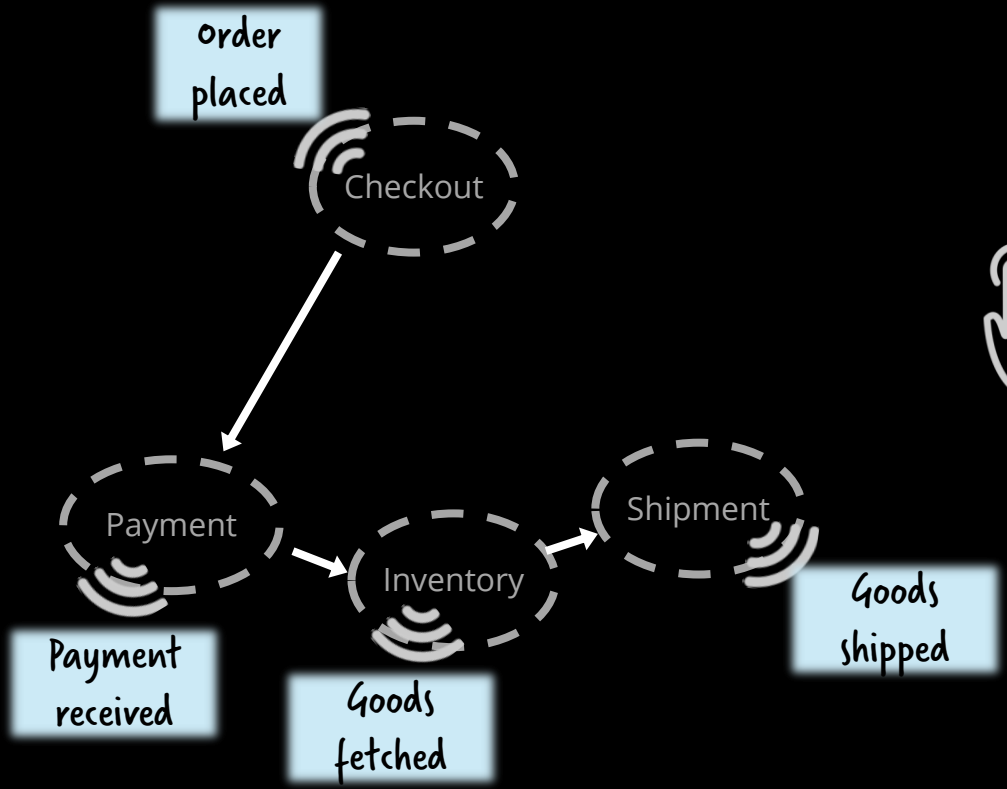
What we wanted

vs. what we got

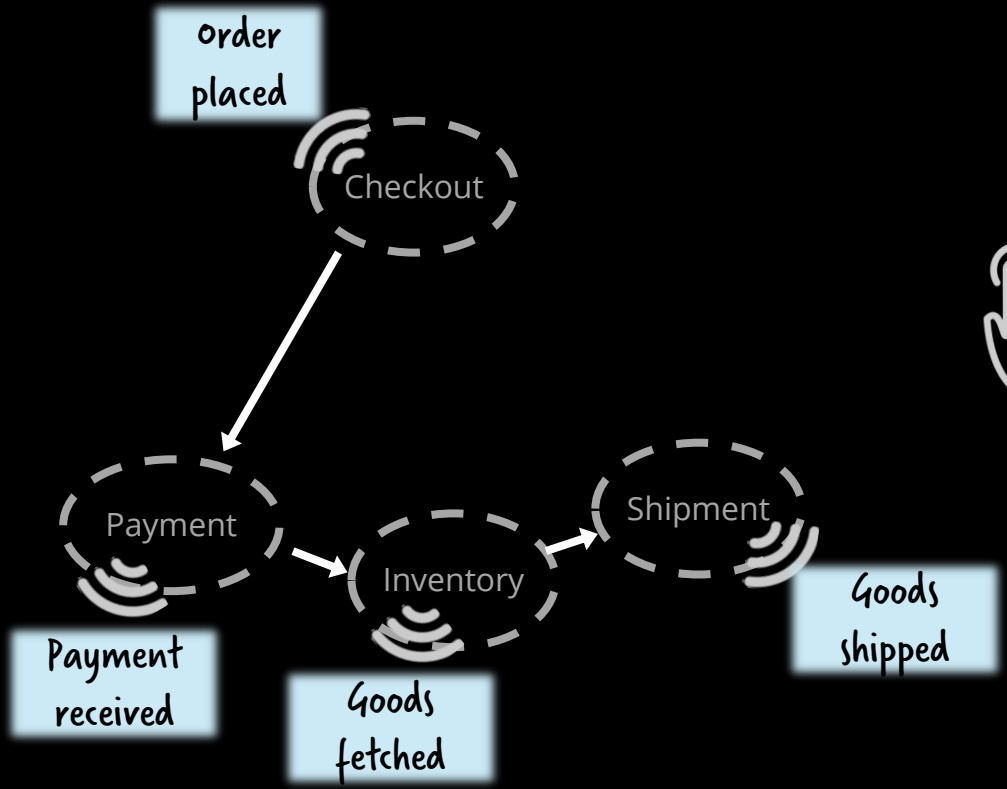
Event-driven

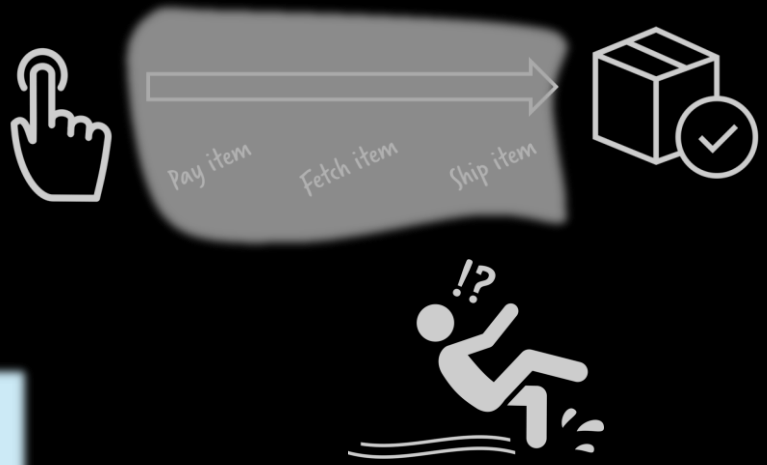


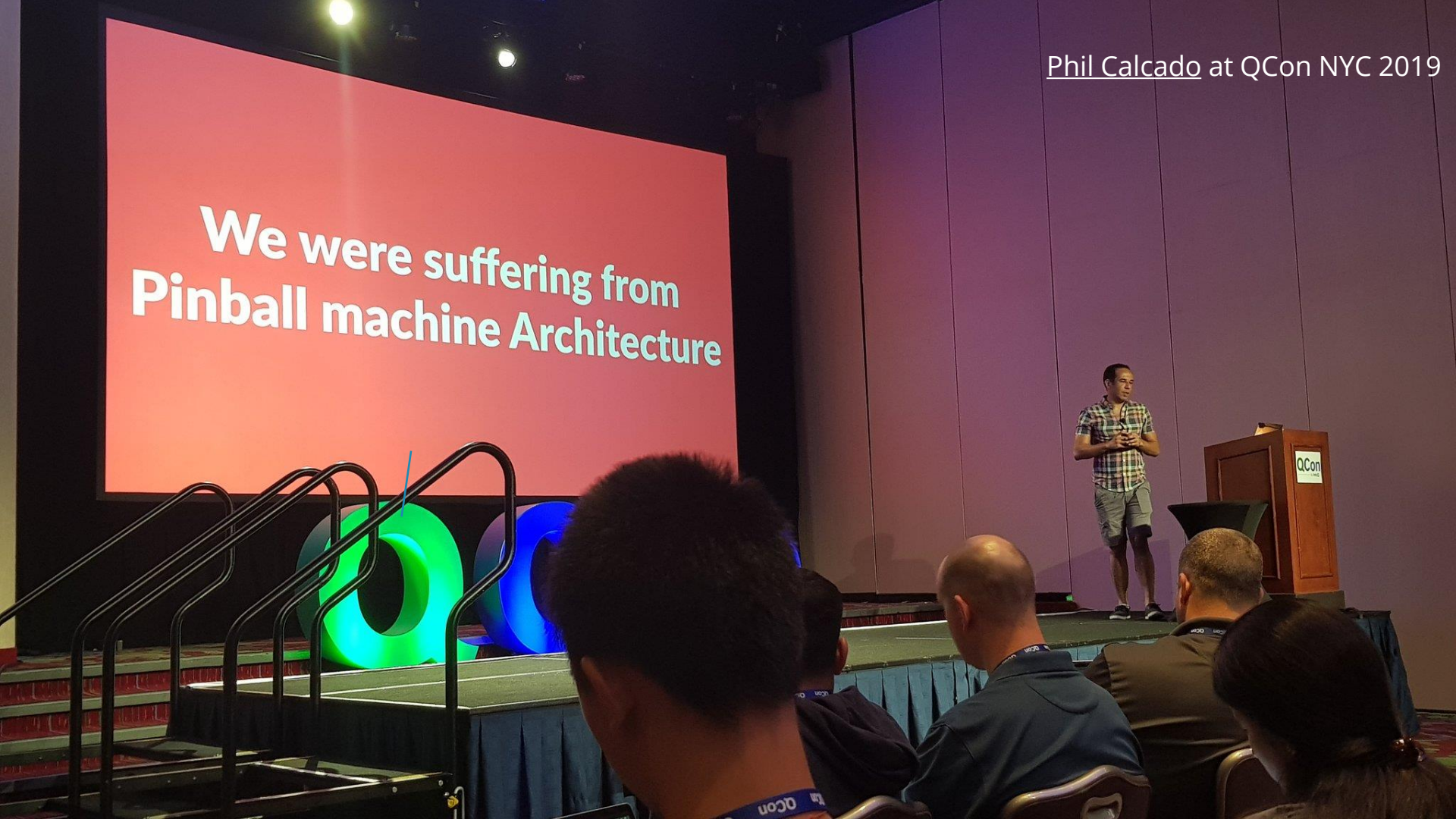
Peer-to-peer event chains



Peer-to-peer event chains





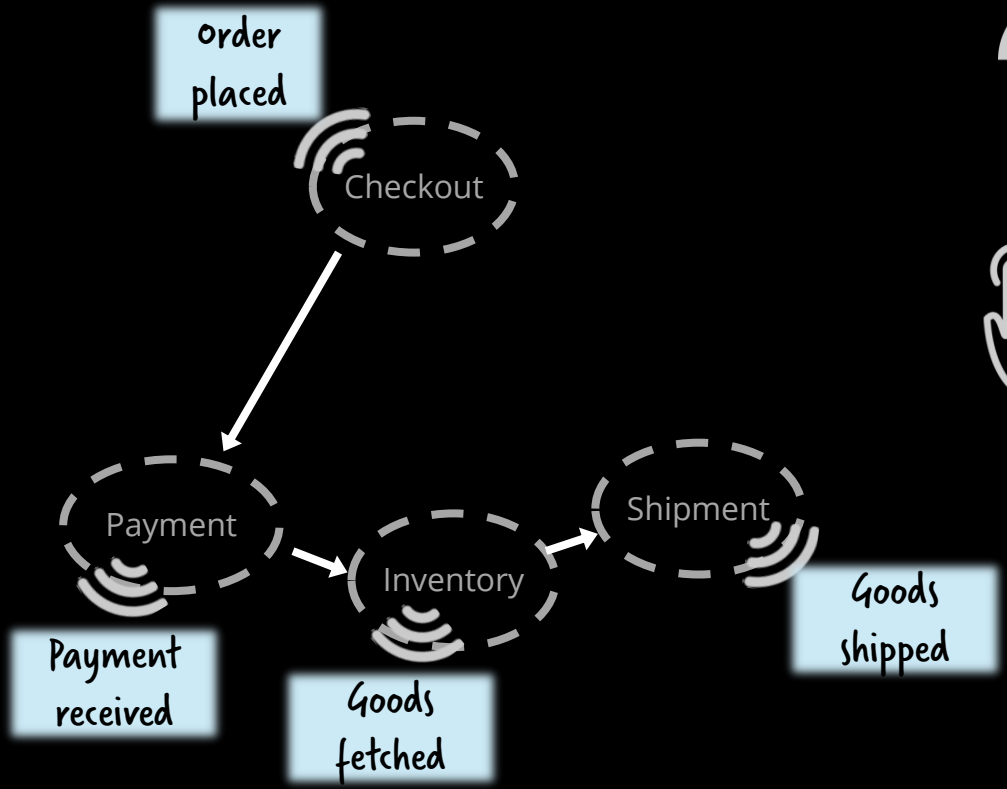


Phil Calcado at QCon NYC 2019

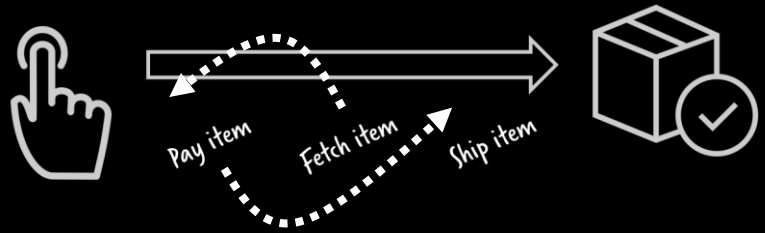
Pinball Machine Archite

„What the hell just happened?“

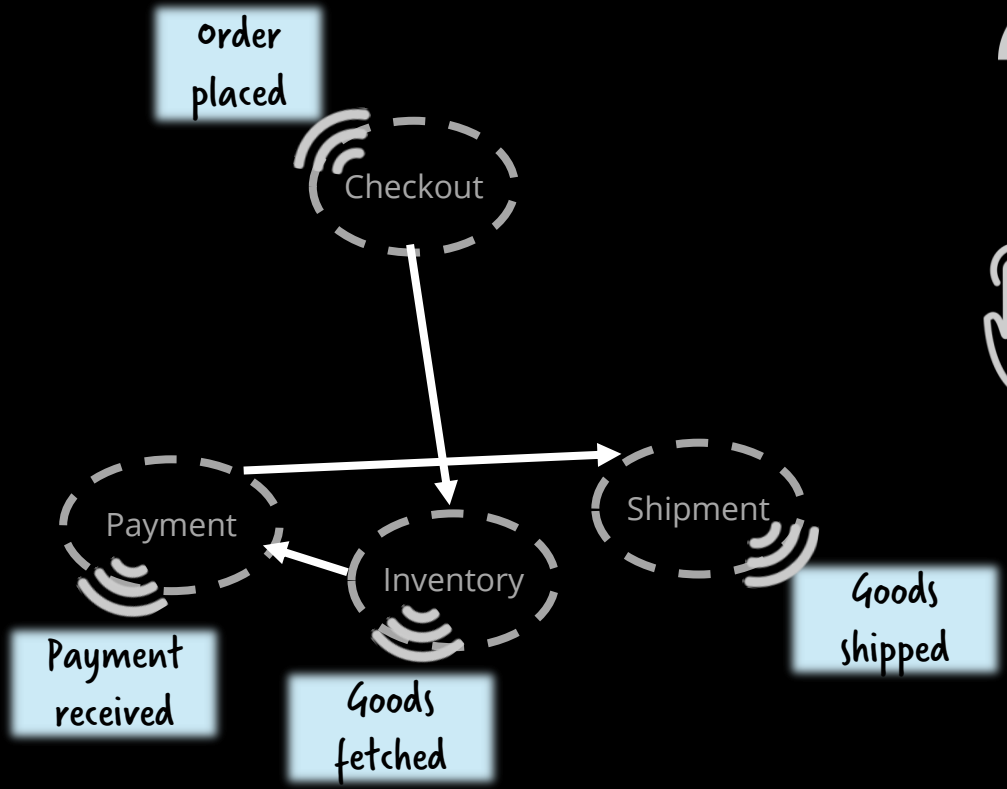
Peer-to-peer event chains



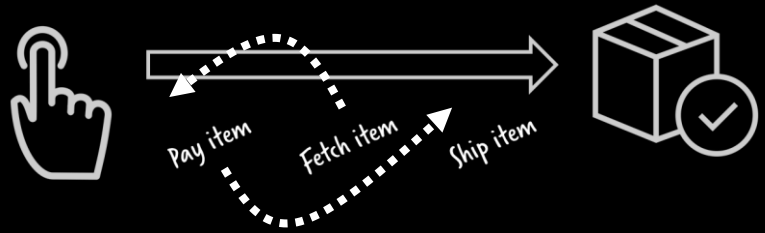
Fetch the goods **before** the payment



Peer-to-peer event chains



Fetch the goods **before** the payment



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

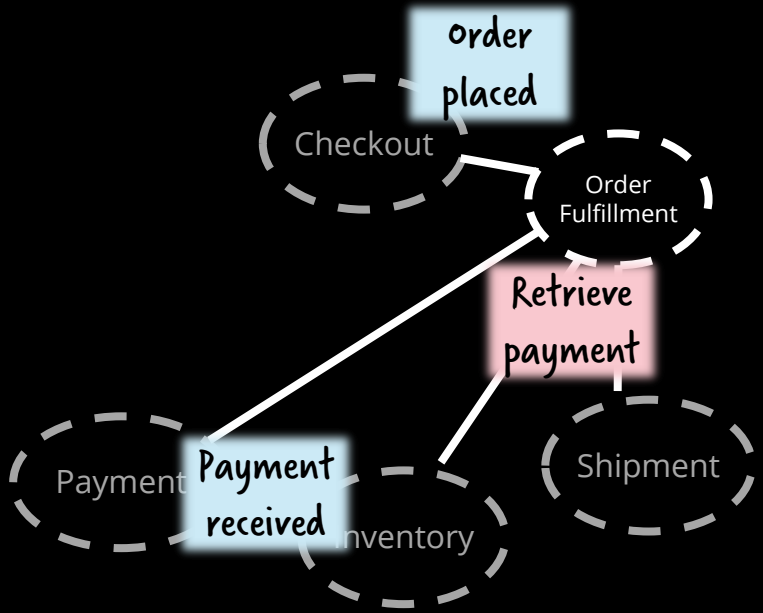
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

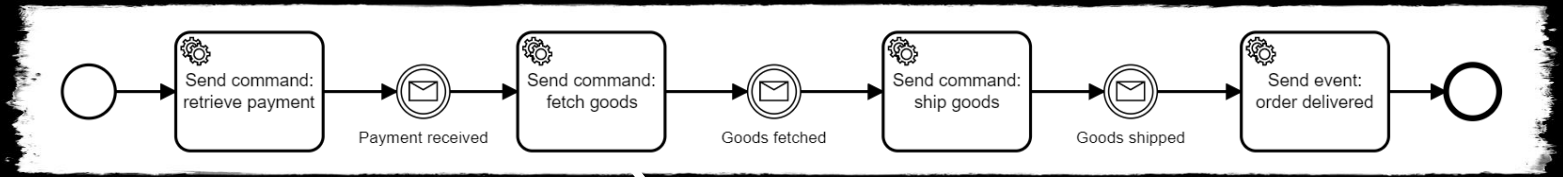
https://martinfowler.com/articles/201701-event-driven.html

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.

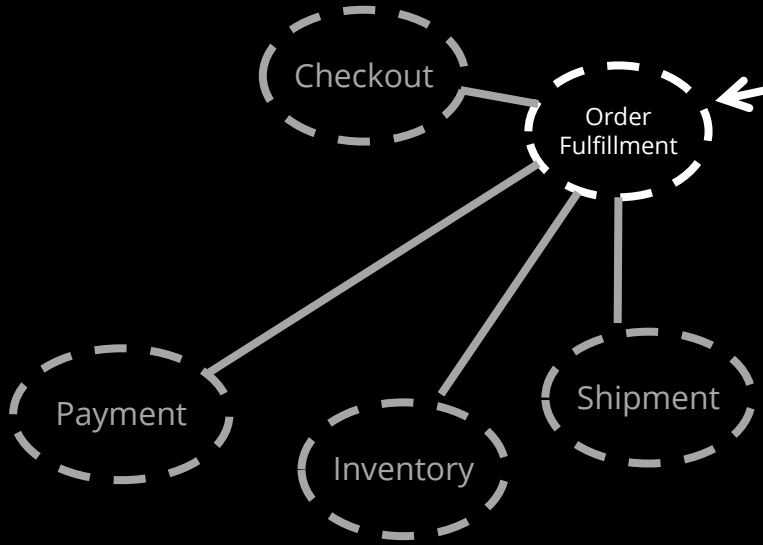
https://martinfowler.com/articles/201701-event-driven.html

Extract the domain logic around order fulfillment

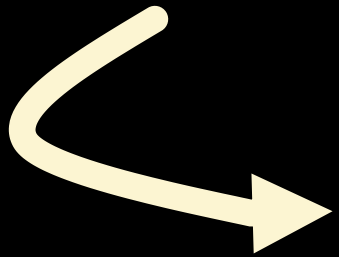


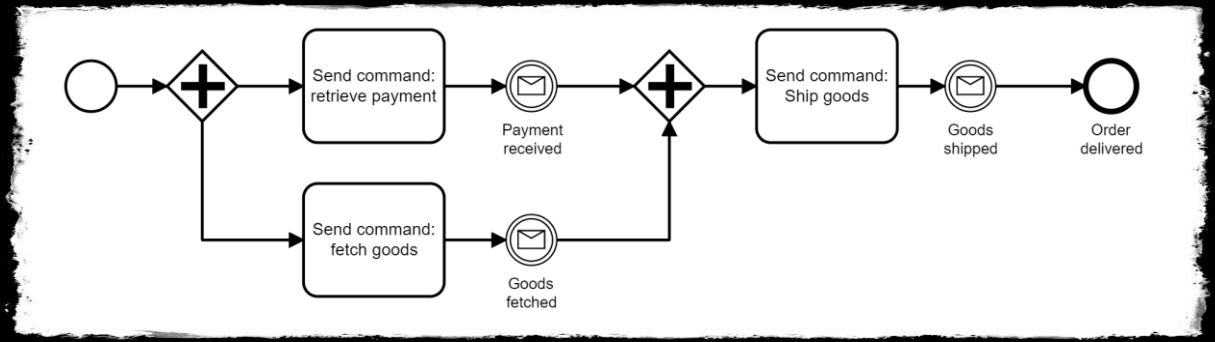


This orchestration requires state

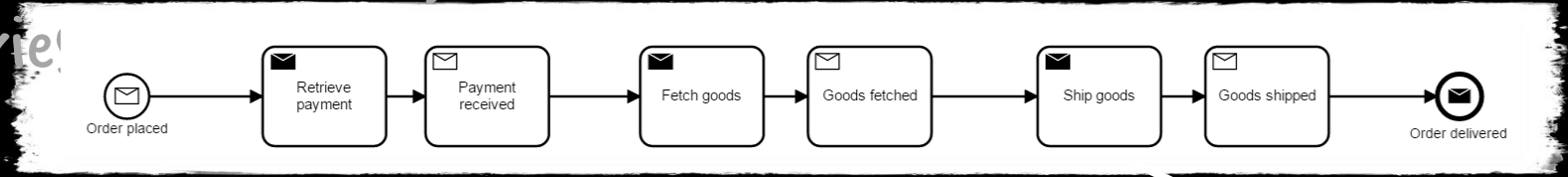


Now it is easy to change the orchestration logic





Processes are domain logic and live inside service boundaries



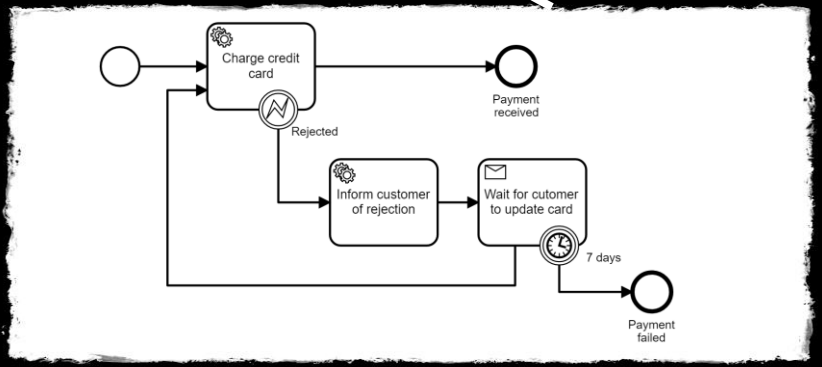
Checkout

Order

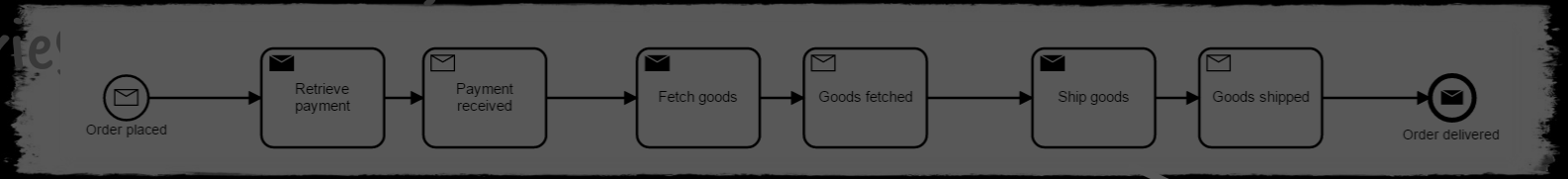
Payment

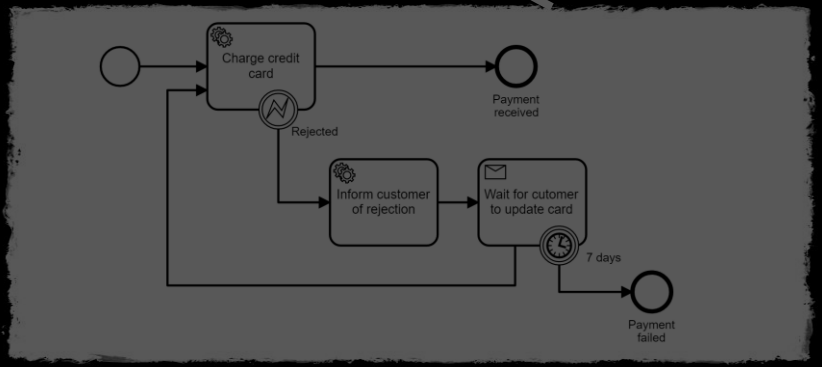
Inventory

Shipment



Processes are domain logic and live inside service boundaries

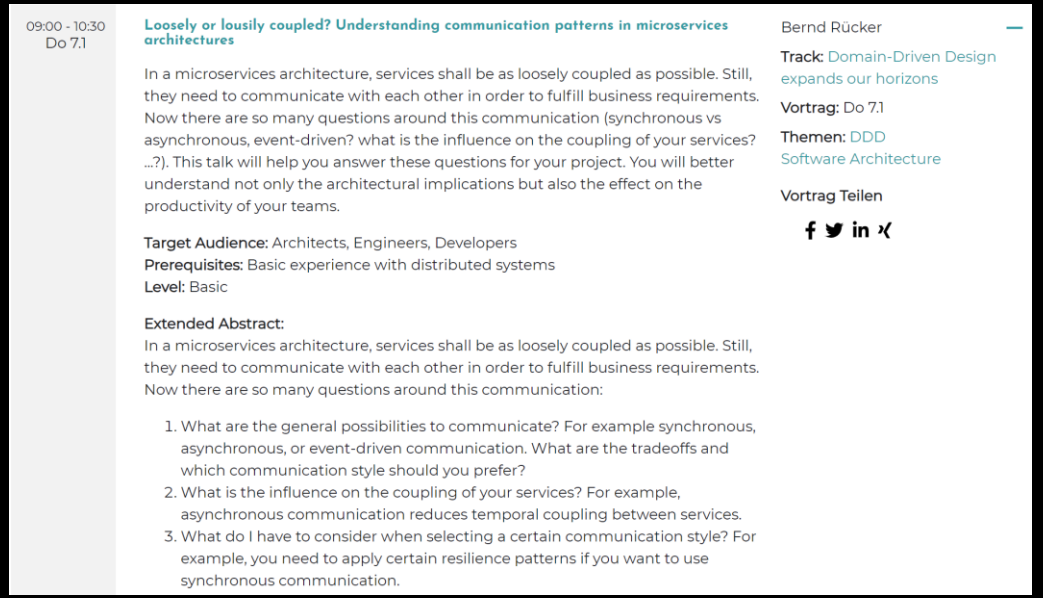




orchestration does **not** need to be central

More tomorrow if you are interested

Confusion around communication and collaboration styles
(REST, Messaging, Events)



09:00 - 10:30
Do 7.1

Loosely or lousily coupled? Understanding communication patterns in microservices architectures

In a microservices architecture, services shall be as loosely coupled as possible. Still, they need to communicate with each other in order to fulfill business requirements. Now there are so many questions around this communication (synchronous vs asynchronous, event-driven? what is the influence on the coupling of your services? ...?). This talk will help you answer these questions for your project. You will better understand not only the architectural implications but also the effect on the productivity of your teams.

Target Audience: Architects, Engineers, Developers
Prerequisites: Basic experience with distributed systems
Level: Basic

Extended Abstract:
In a microservices architecture, services shall be as loosely coupled as possible. Still, they need to communicate with each other in order to fulfill business requirements. Now there are so many questions around this communication:

1. What are the general possibilities to communicate? For example synchronous, asynchronous, or event-driven communication. What are the tradeoffs and which communication style should you prefer?
2. What is the influence on the coupling of your services? For example, asynchronous communication reduces temporal coupling between services.
3. What do I have to consider when selecting a certain communication style? For example, you need to apply certain resilience patterns if you want to use synchronous communication.

Bernd Rücker

Track: Domain-Driven Design expands our horizons

Vortrag: Do 7.1

Themen: DDD Software Architecture

Vortrag Teilen

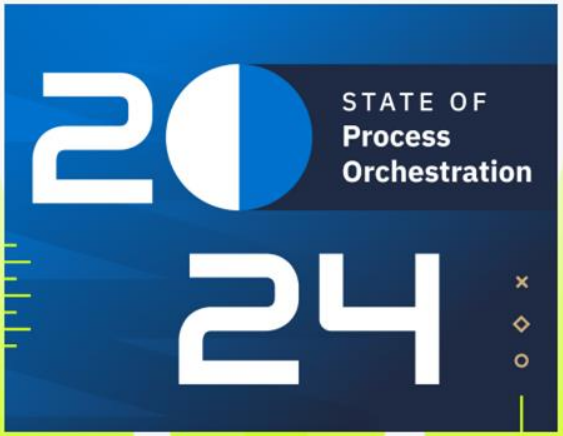
There is trouble in software engineering land!

Confusion around communication and collaboration styles
(REST, Messaging, Events)

A wrong understanding of decoupling and the resulting event-driven chaos

Not thinking beyond Microservices

Missing out on (a healthy level of) centralization



State of Process Orchestration Report 2024

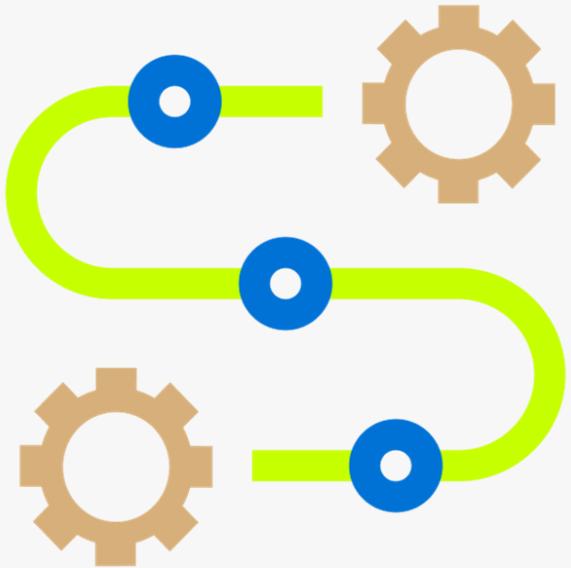
This report compiles findings from a survey of IT professionals across North America and Europe to better understand the current business drivers and opportunities for automating business processes.

This is the fourth year we've produced this 'State Of' report, which also compares shifts from year to year and how organizations have matured in their automation initiatives.

Download the report

https://bit.ly/sopo-2024

Processes are getting more complex



51%

of IT decision makers say processes spanning multiple systems is a reason for process automation complexity

↗ (up from 41% in 2023)

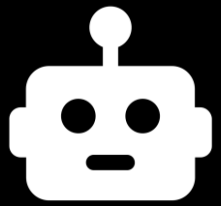
State of Process Orchestration 2024

60%

of business and IT decision-makers estimate that 26 or more systems are involved in their organization's automation implementation

State of Process Orchestration 2024

Is everything a Microservice?

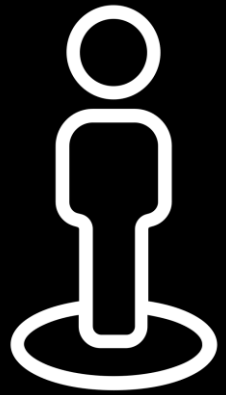


RPA & Bots

SaaS Services

Legacy Systems

Front-end applications



Humans

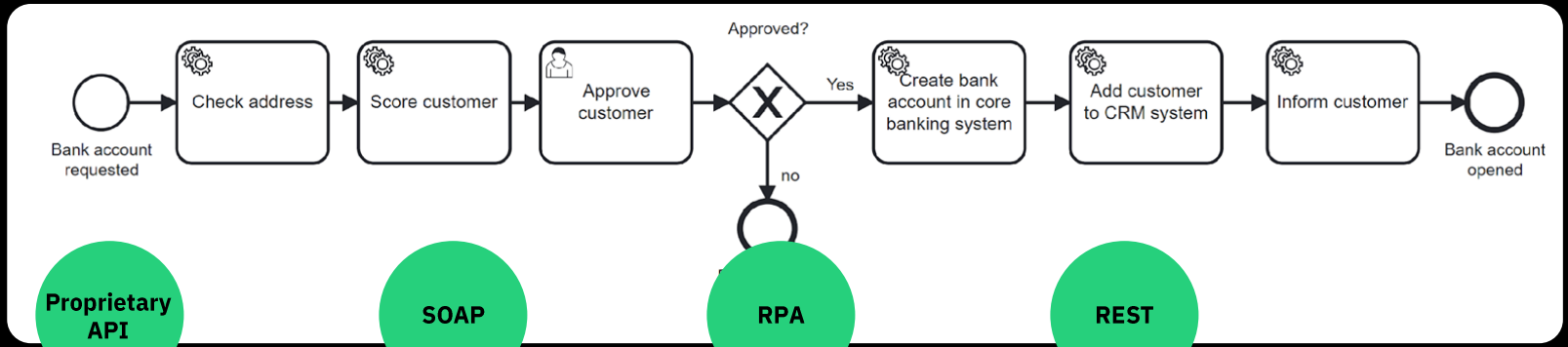
You need to integrate all endpoints!

Clerks

User Interface

Messaging

Website



Proprietary API

SOAP

RPA

REST

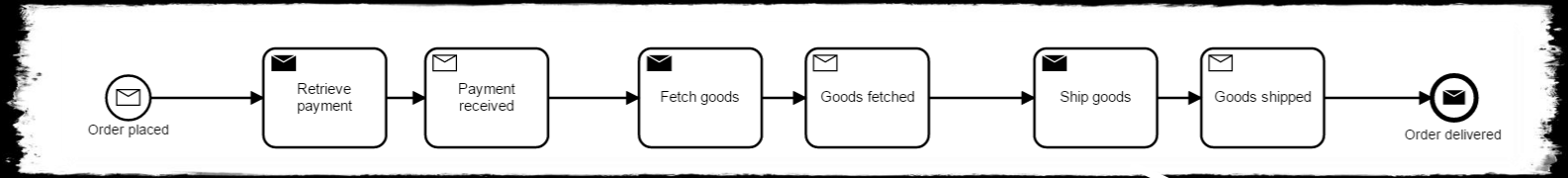
Core Banking System

CRM System

Scoring System

Address Check

Processes can still live in microservices



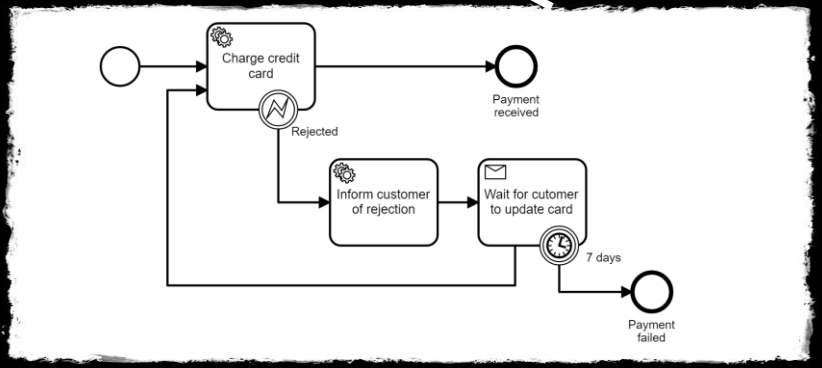
Checkout

Order

Payment

Inventory

Shipment



There is trouble in software engineering land!

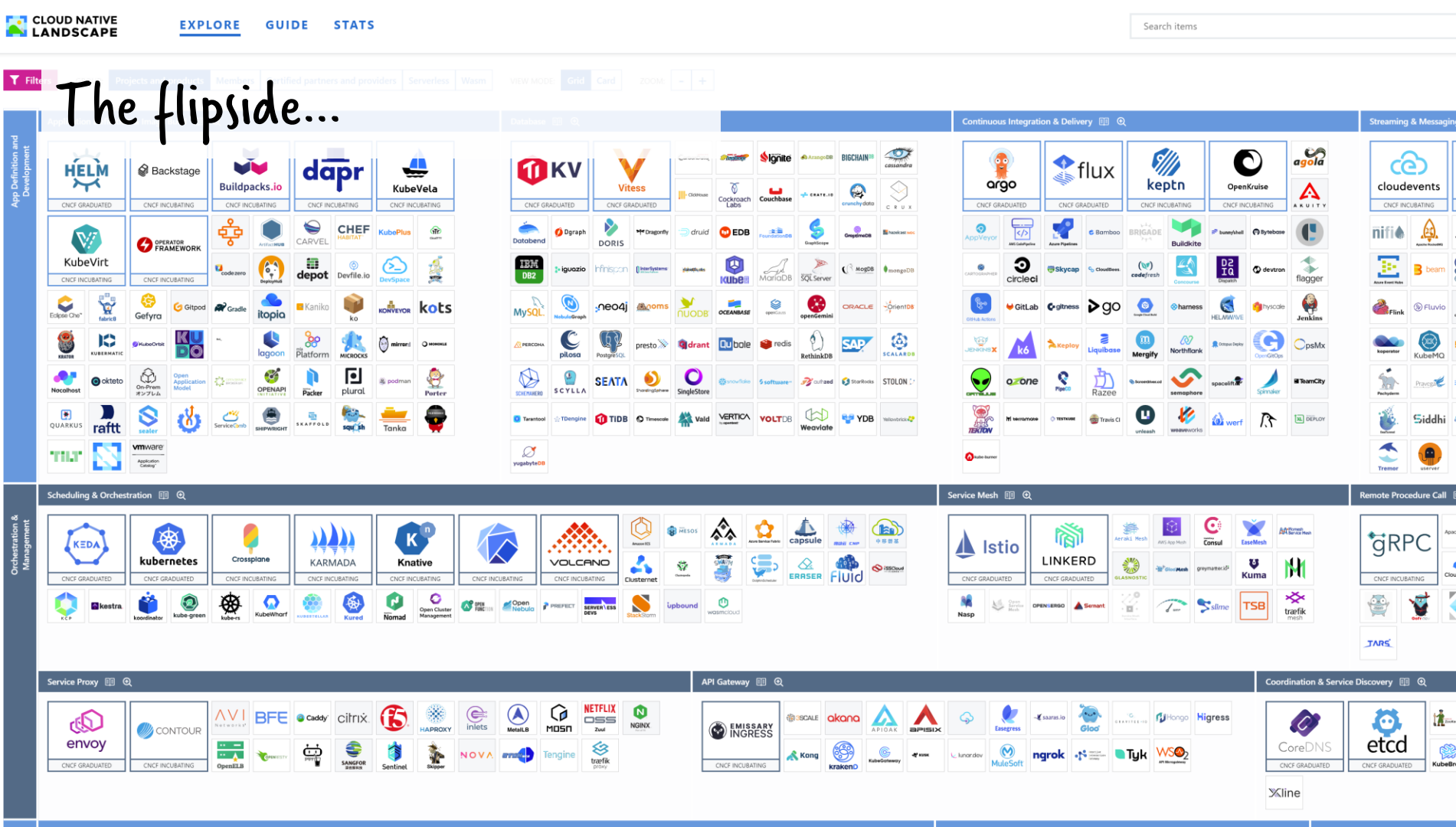
Confusion around communication and collaboration styles
(REST, Messaging, Events)

A wrong understanding of decoupling and the resulting event-driven chaos

Not thinking beyond Microservices

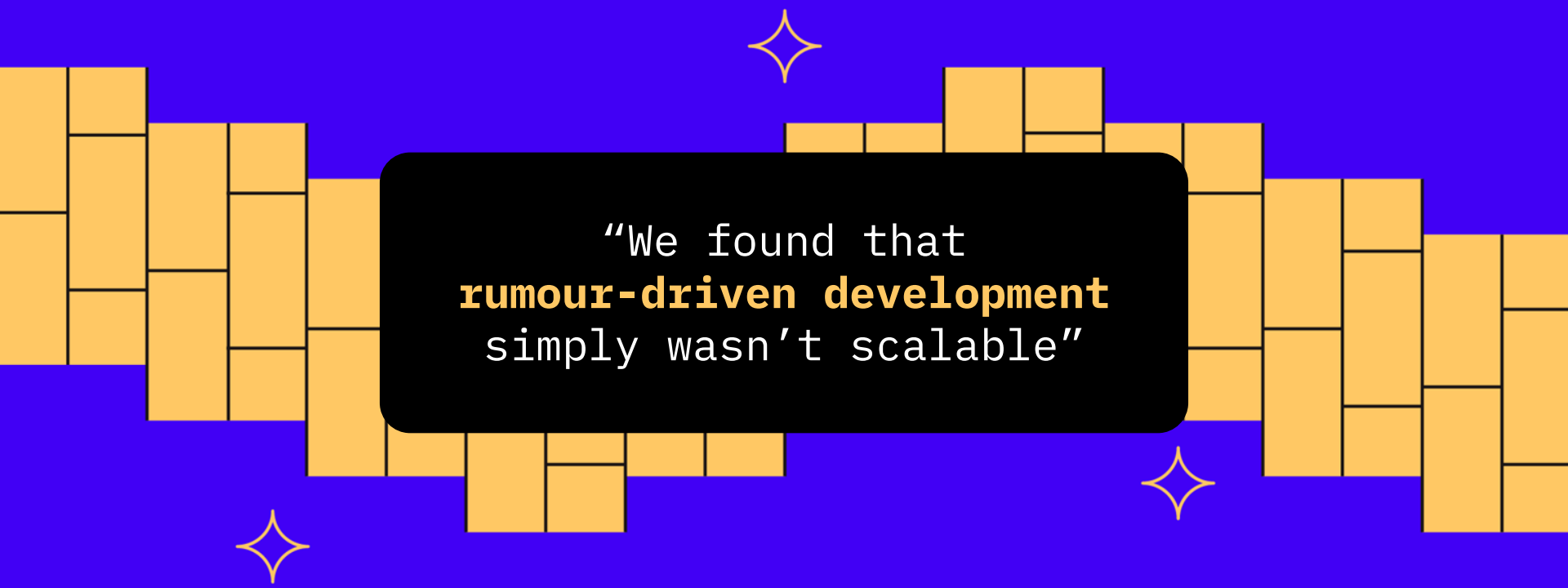
Missing out on (a healthy level of) centralization

Autonomy!


CLOUD NATIVE LANDSCAPE
EXPLORE
GUIDE
STATS
Search items
The flipside...
App Definition and Development
Database
Continuous Integration & Delivery
Streaming & Messaging
Orchestration & Management
Scheduling & Orchestration
Service Mesh
Remote Procedure Call
Service Proxy
API Gateway
Coordination & Service Discovery
HELM
Backstage
Buildpacks.io
dapr
KubeVela
KubeVirt
OPERATOR FRAMEWORK
KV
Vitess
argo
flux
keptn
OpenKruise
cloudevents
KEDA
kubernetes
Crossplane
KARMADA
Knative
VOLCANO
Istio
LINKERD
gRPC
envoy
CONTOUR
HAPROXY
BFE
Caddy
citrix
NGINX
EMISSARY INGRESS
Kong
CoreDNS
etcd
CNCF GRADUATED
CNCF INCUBATING

Spotify R&D

Golden Paths



https://engineering.atspotify.com/2020/08/how-we-use-golden-paths-to-solve-fragmentation-in-our-software-ecosystem/

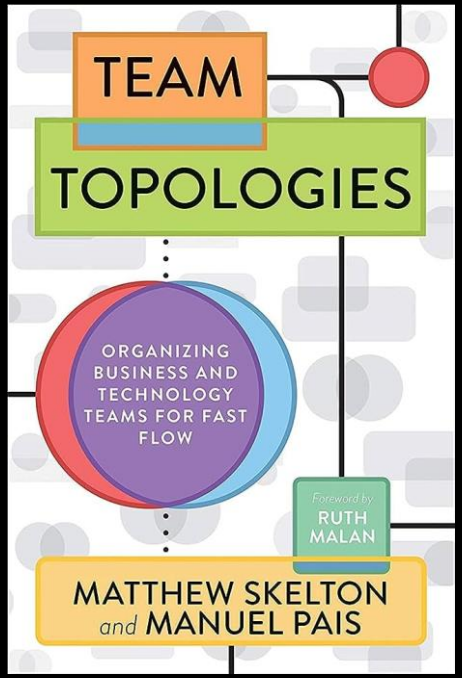
The Speed Paradox

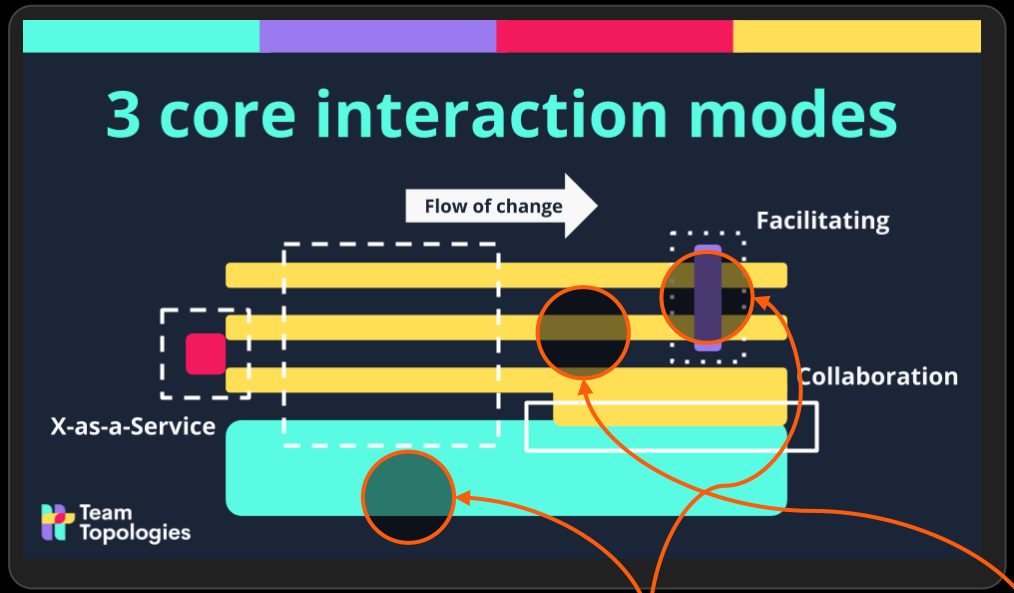
At Spotify, we've always believed in the speed and ingenuity that comes from having autonomous development teams. But as we learned firsthand, the faster you grow, the more fragmented and complex your software ecosystem becomes. And then everything slows down again.

The Standards Paradox

By centralizing services and standardizing your tooling, Backstage streamlines your development environment from end to end. Instead of restricting autonomy, standardization frees your engineers from infrastructure complexity. So you can return to building and scaling, quickly and safely.

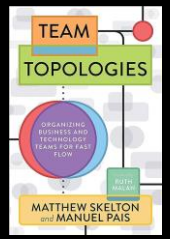
https://backstage.io/ (OSS, Made with 💔 at Spotify)


TEAM
TOPOLOGIES
ORGANIZING BUSINESS AND TECHNOLOGY TEAMS FOR FAST FLOW
Foreword by
RUTH MALAN
MATTHEW SKELTON
and MANUEL PAIS



Center of Excellence

Domain



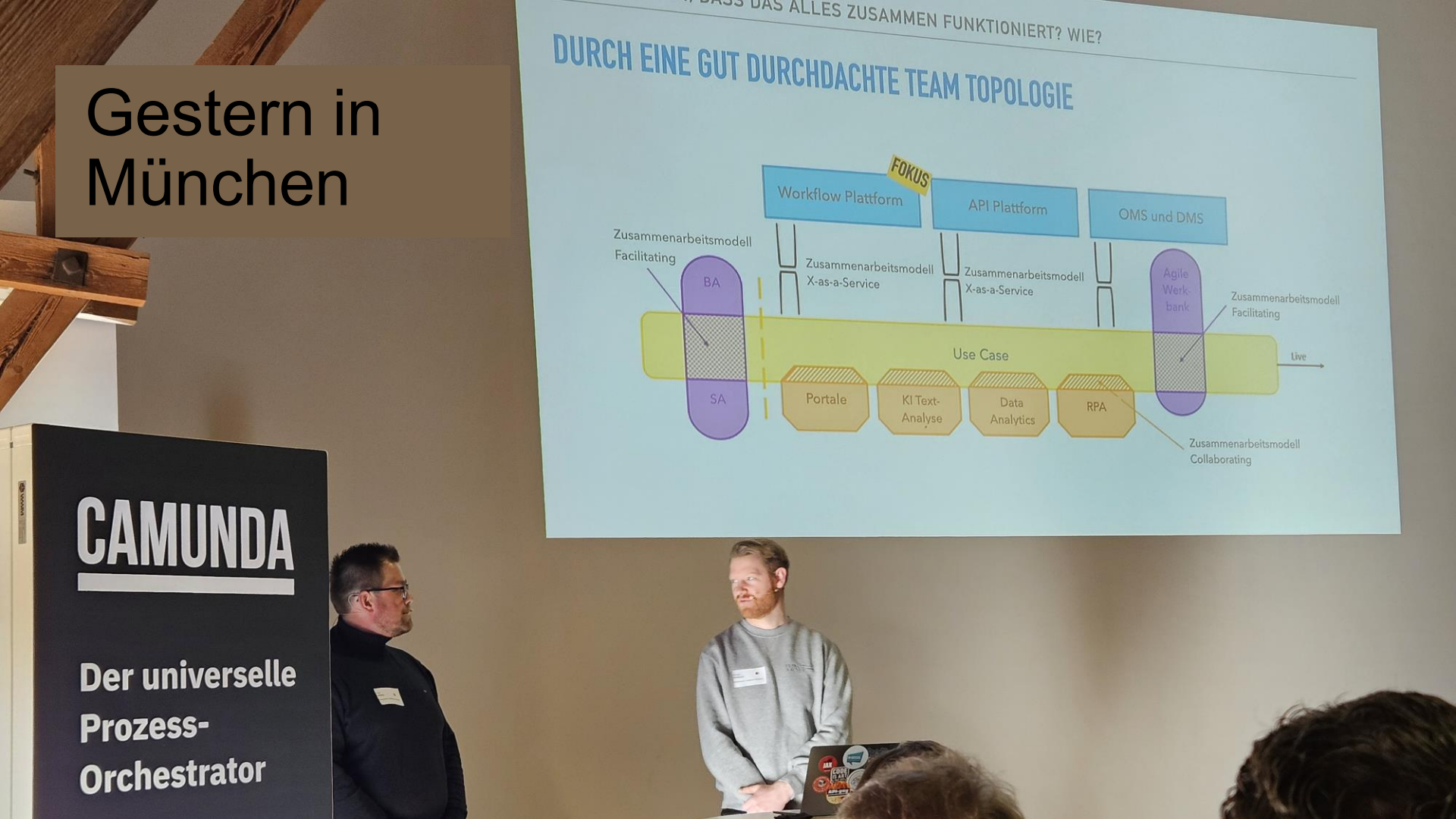
Stream-aligned team

Enabling team

Complicated Subsystem team

Platform team

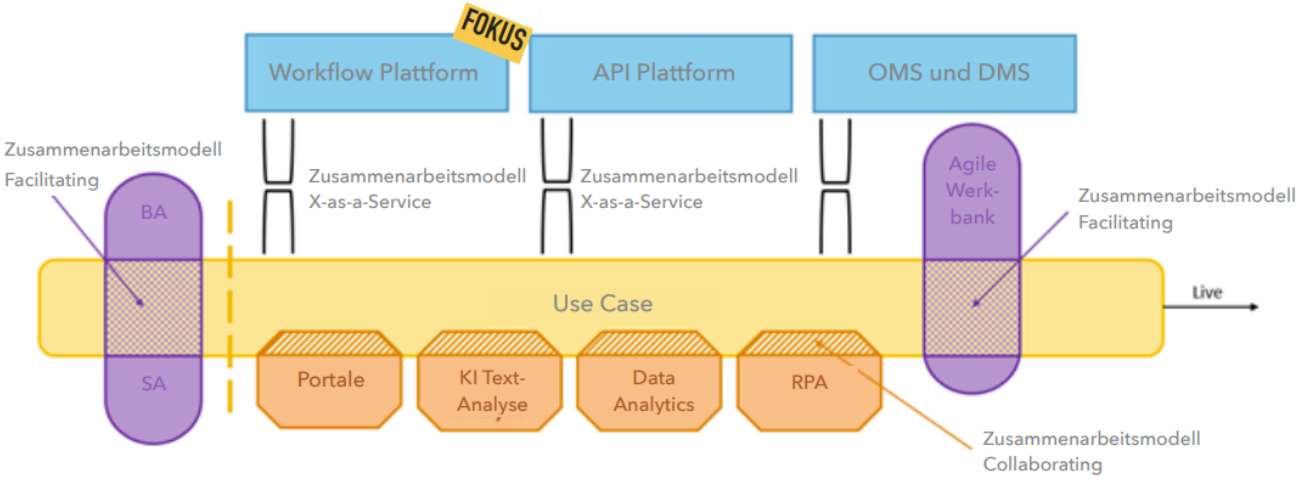
Gestern in München



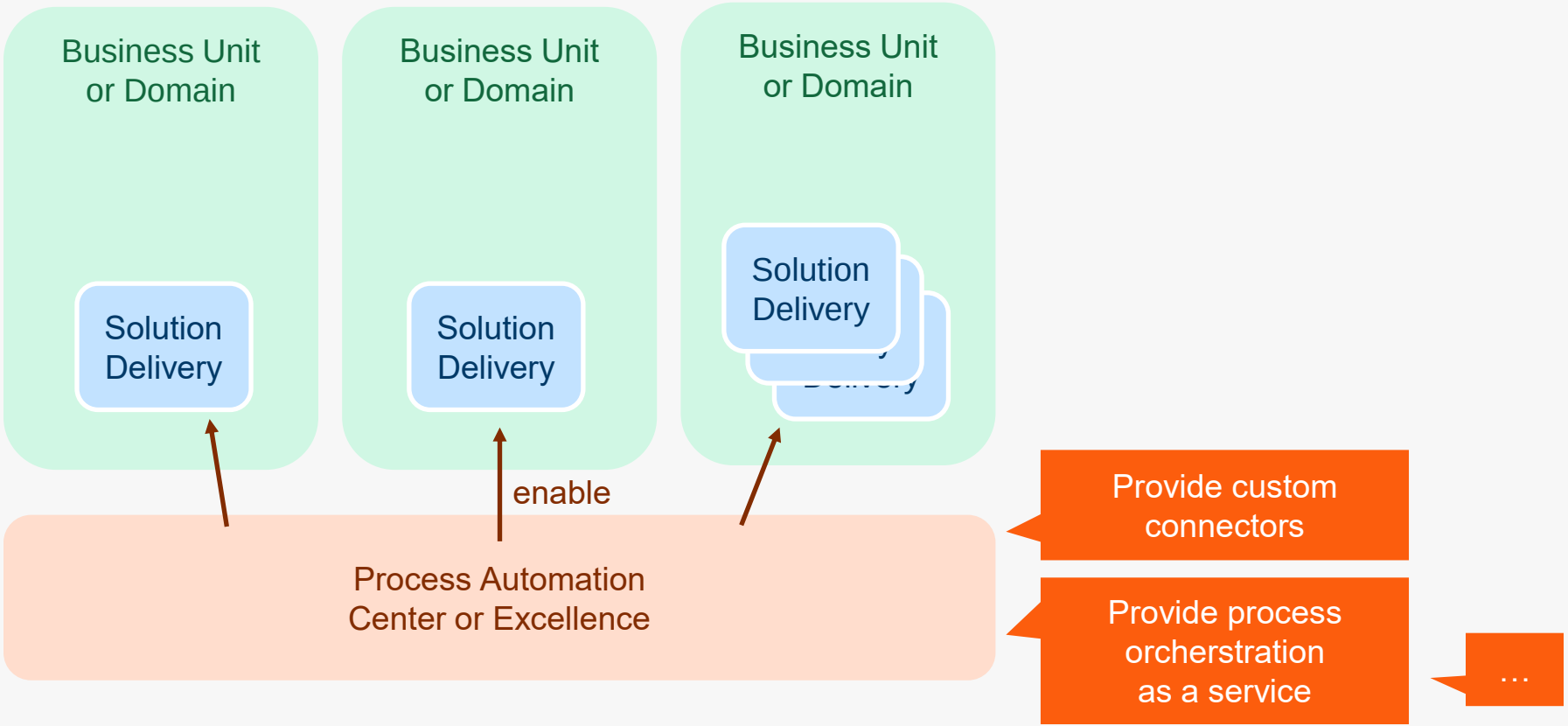
DIGITALE ENABLER – WORKFLOW COE IN A NUTSHELL

DAFÜR SORGEN, DASS DAS ALLES ZUSAMMEN FUNKTIONIERT? WIE?

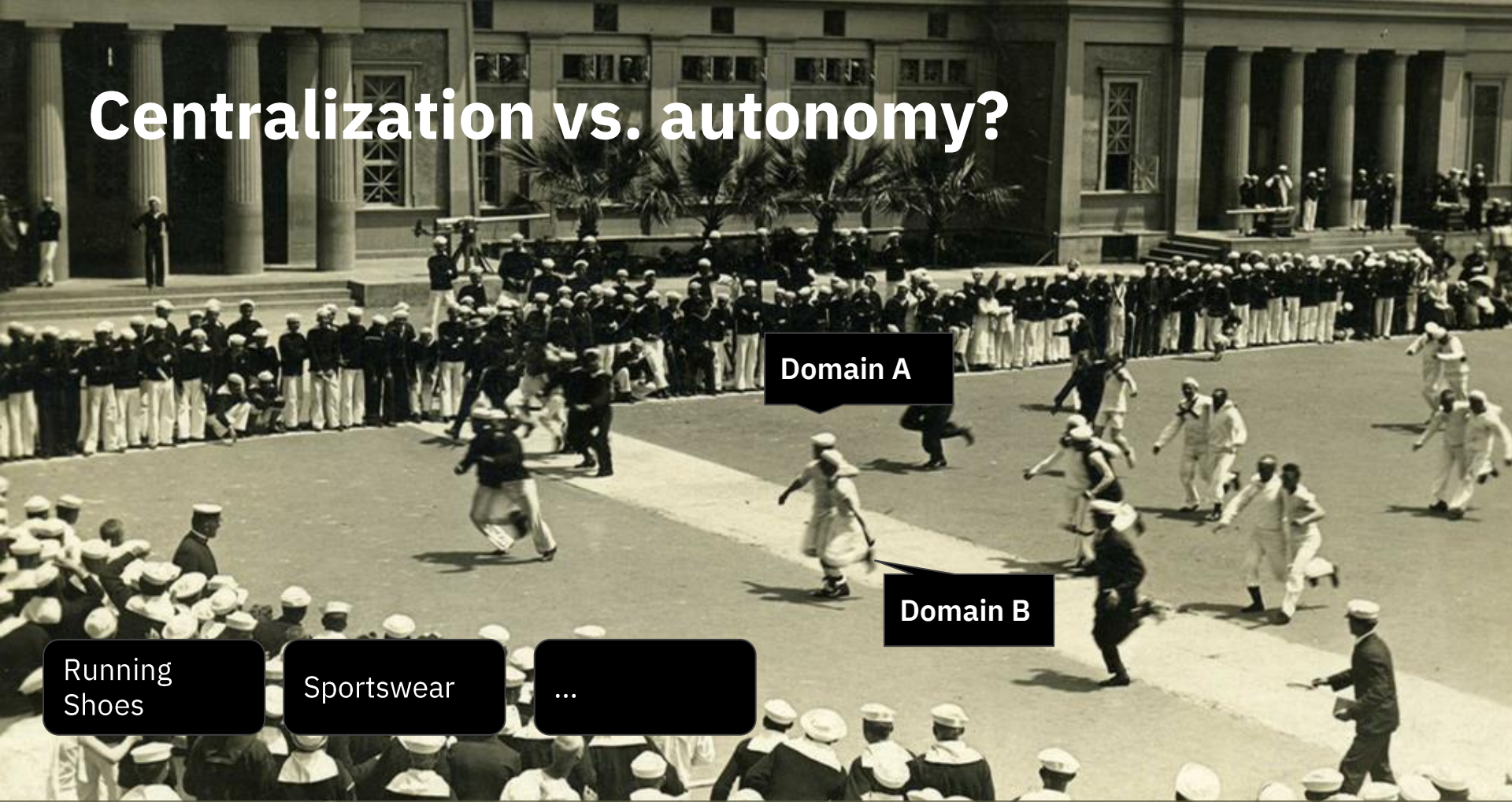
DURCH EINE GUT DURCHDACHTE TEAM TOPOLOGIE



Scaling adoption



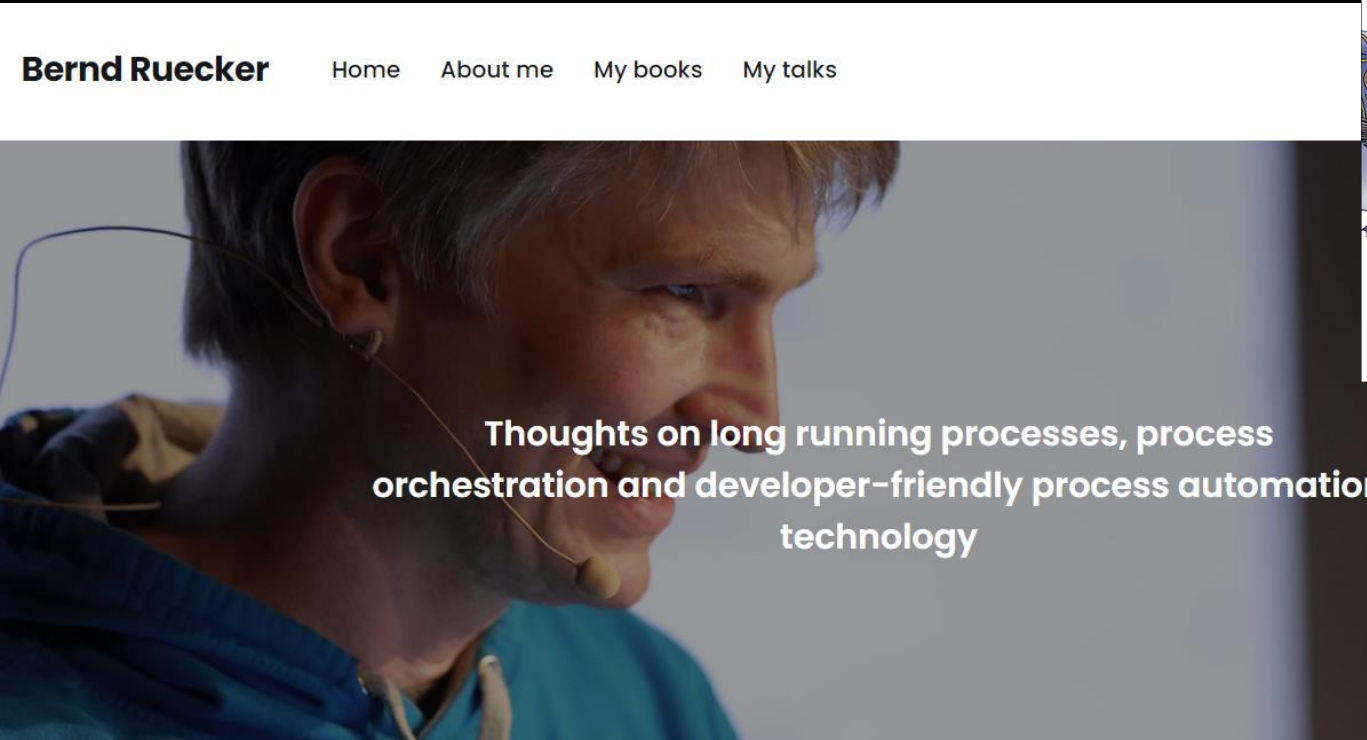
Centralization vs. autonomy?

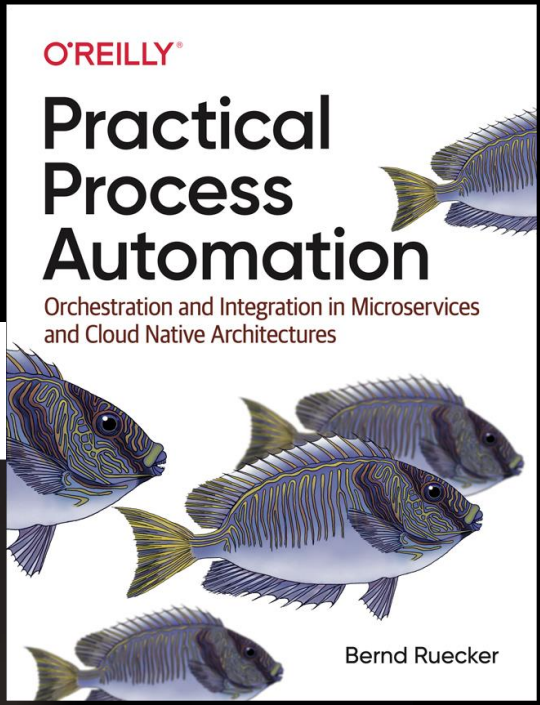


Process orchestration is an essential building brick for automation
Process automation = process orchestration + task automation
Red, yellow, and green processes have different requirements
Include more roles in solution creation
Low code and developer friendlyness are not mutually exclusive
Process orchestration can be part of software engineering, but
  # Don't fall in the evnt-driven chaos trap
  # Think beyond microservices
Establish a healthy level of centralization (Platform teams)

Want To Know More?

https://berndruecker.io/





THE PROCESS ORCHESTRATION CONFERENCE

199€* für
Teilnahme in Berlin

(* Ticketpreis bis 15. März 2024)

Jetzt anmelden

CAMUNDA CON 2024

https://www.camundacon.com/

BERLIN + ONLINE 15. & 16. Mai

Thank you!

Contact: bernd.ruecker@camunda.com
@berndruecker

Slides: https://berndruecker.io

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

Code: https://github.com/berndruecker