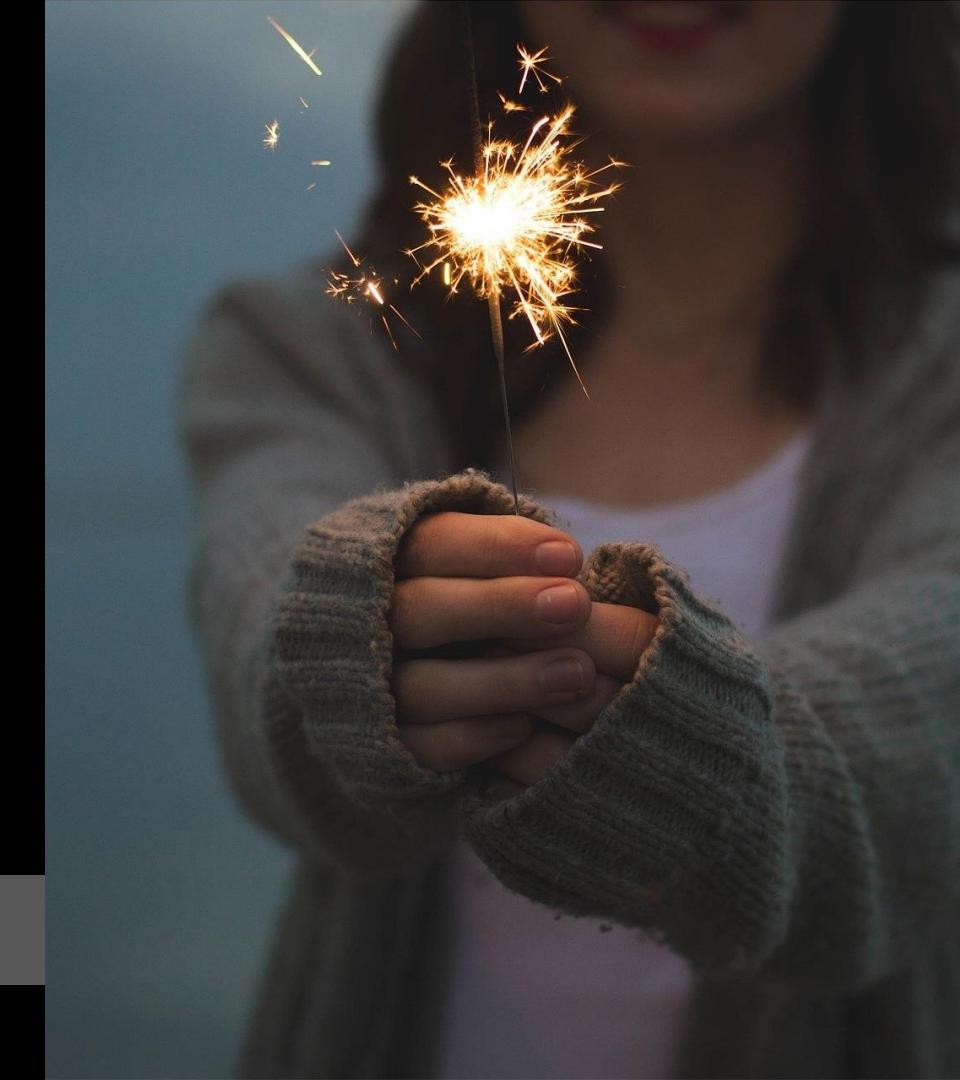


How Process Orchestration Increases Agility Without Harming Architecture

@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)

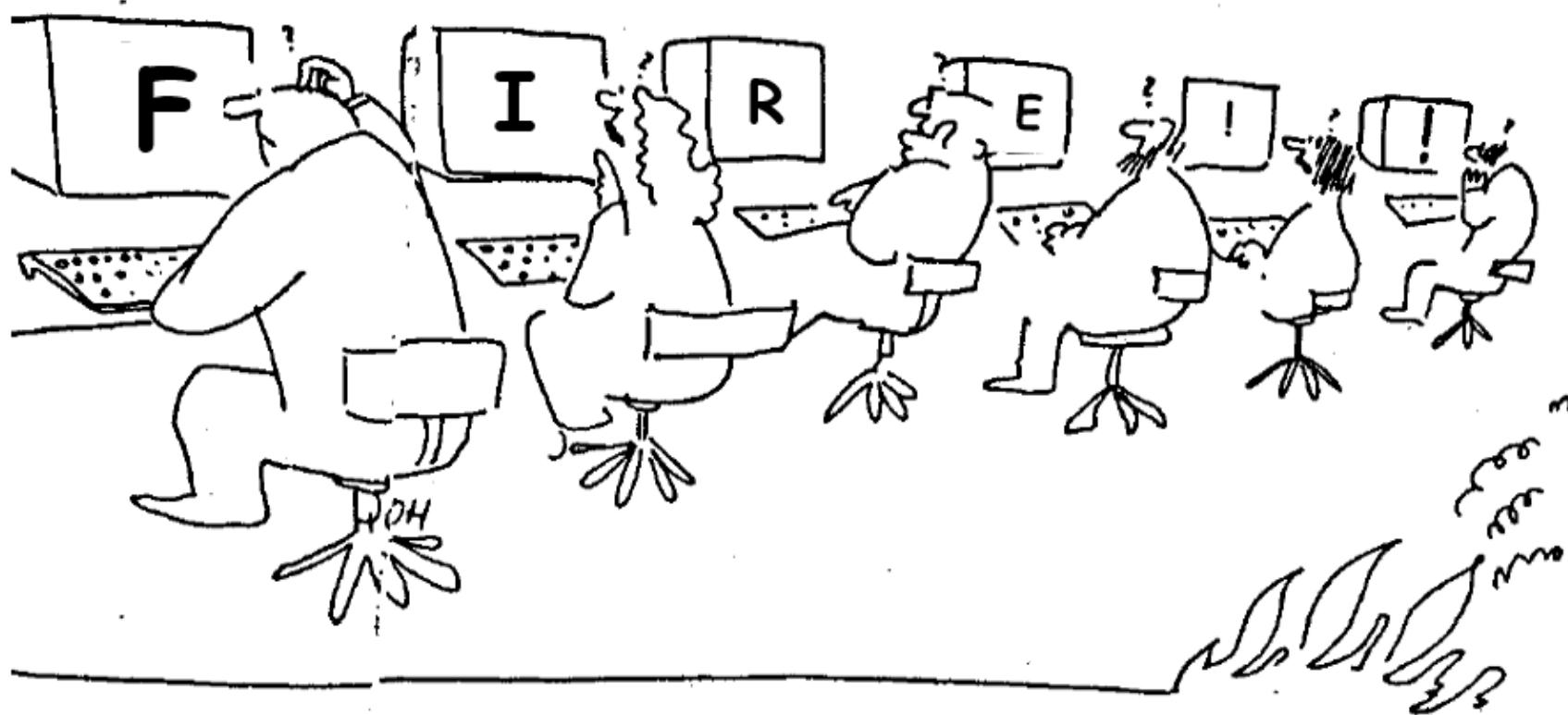


Processes span across Silos

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**.



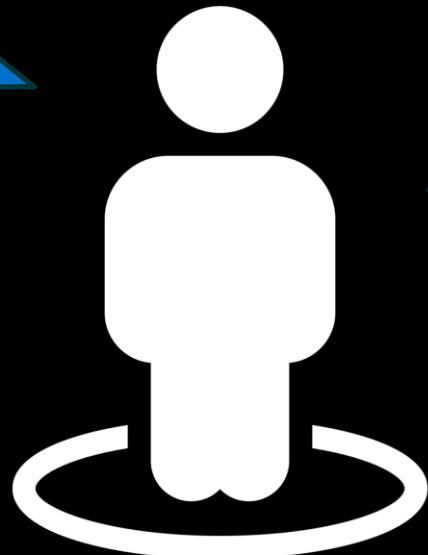


The customer doesn't care

Please open
a bank account
for me

Wow, that
was...

...slow 😞





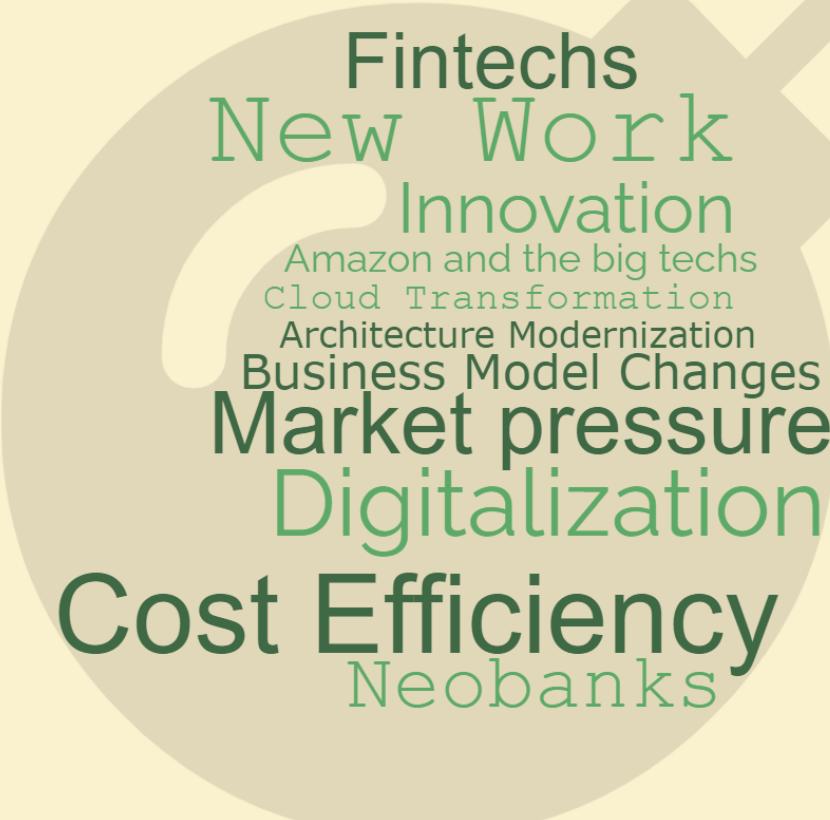
Fintechs
New Work
Innovation

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

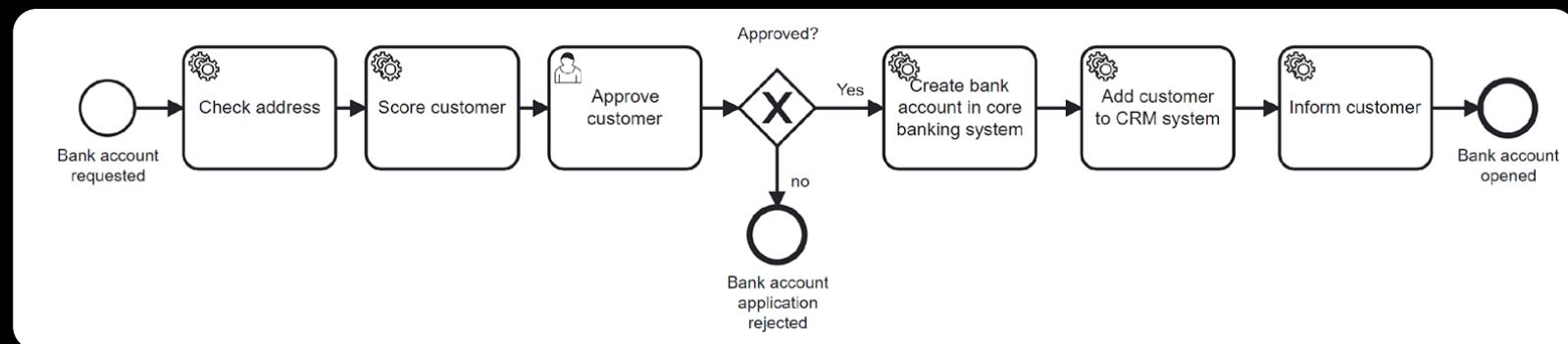


Process orchestration

C

Clerks

Website



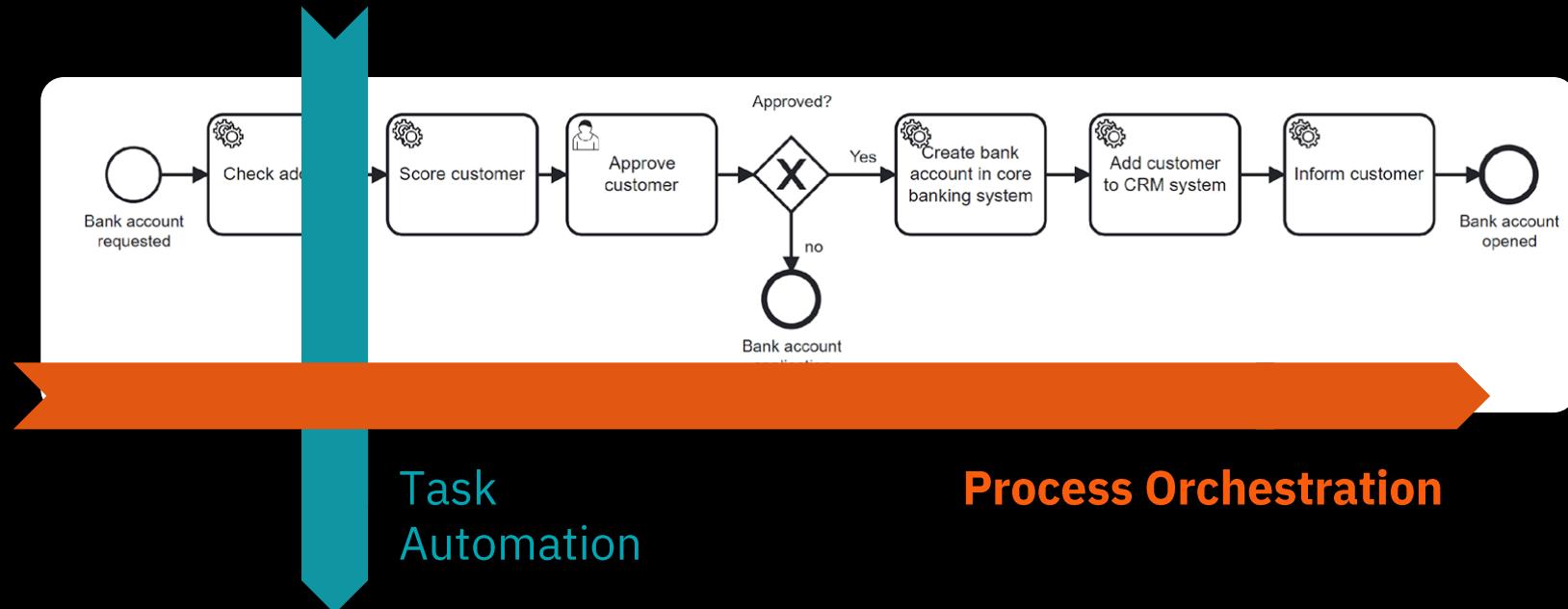
Core Banking System

CRM System

Scoring System

Address Check

Process automation: Task automation vs. process orchestration



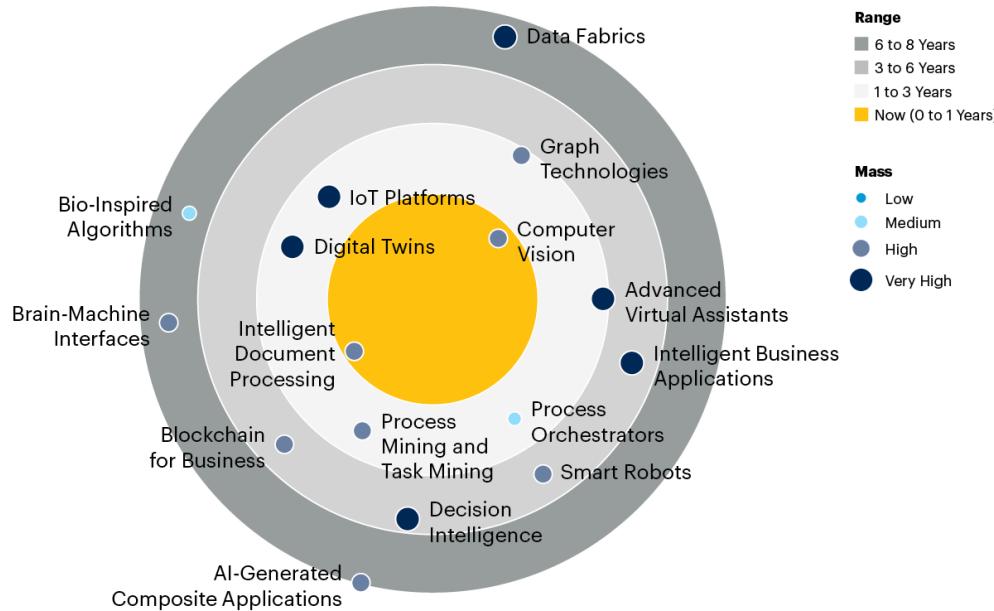
Task
Automation

Process Orchestration

Gartner - Process Orchestration



Impact Radar for Hyperautomation



Source: Gartner
772074_C

"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





BOTS & PROCESS IMPROVEMENT AT THE SAME TIME?

OUR AUTOMATION JOURNEY
@ DEUTSCHE TELEKOM SERVICE

Marco Einacker
Christoph Anzer

Bonn | 08.10.2020

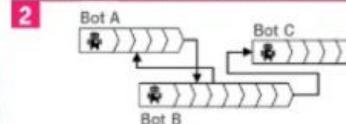
3: FROM FRONTEND AUTOMATION TO BACKEND AUTOMATION

Manual process



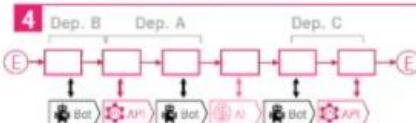
Short time-to-market results in "quick & dirty" process design
→ Complex processes including workarounds

RPA / Frontend Automation



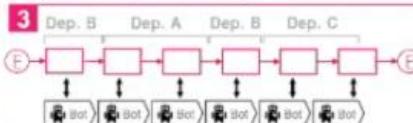
Robotic process automation imitates the human way of working
→ Complex "Spaghetti Bot" automation

Backend Automation



Shift from Bots (Front-End) to APIs (Back-End) and other technologies better fit for purpose
→ Enlarged scope for automation + higher efficiency

Separation process layer

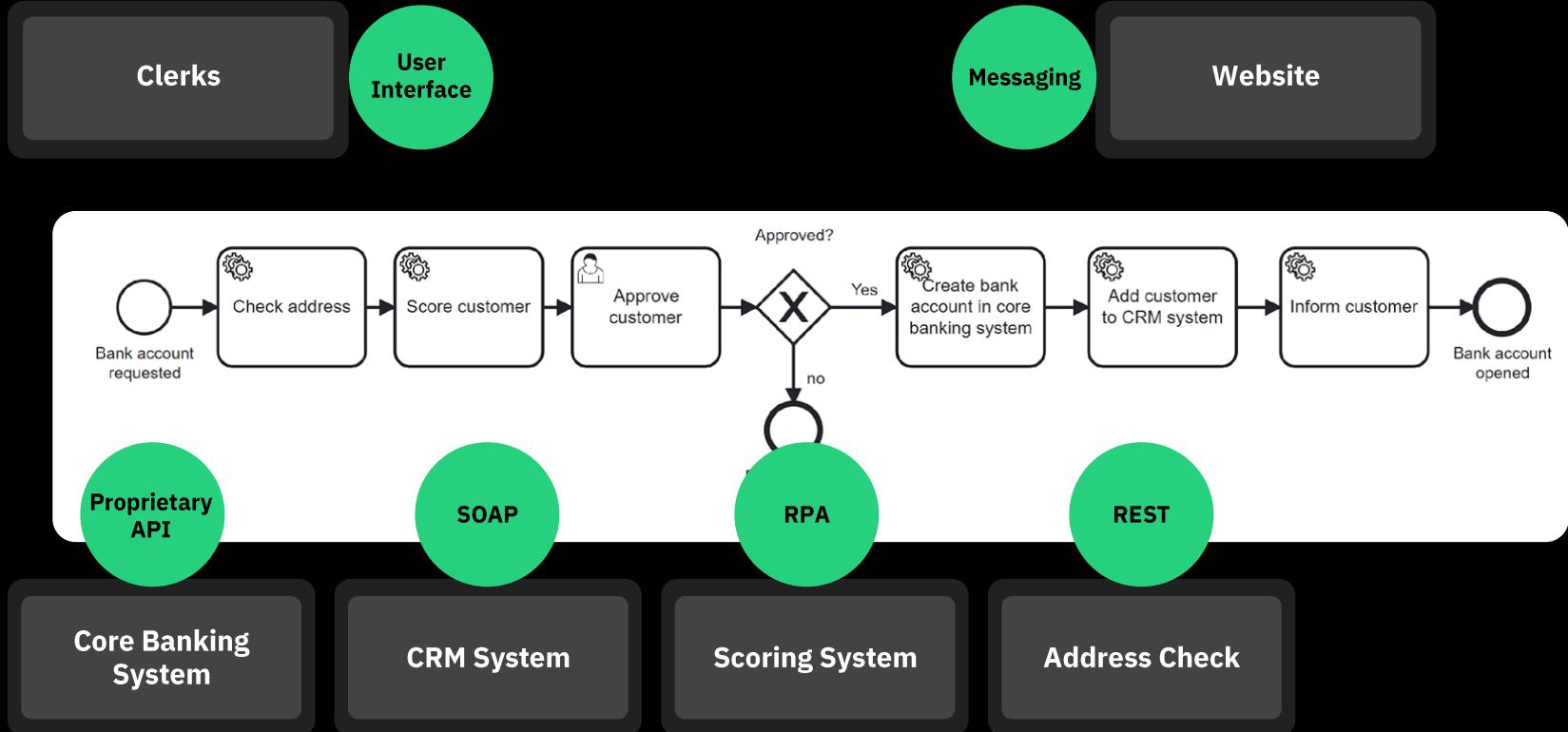


Separation of **Process Layer** (Bot Orchestration) and **Bot Layer**
→ Increased process transparency and optimization

"Spaghetti bot automation"



Integrating diverse endpoints



```

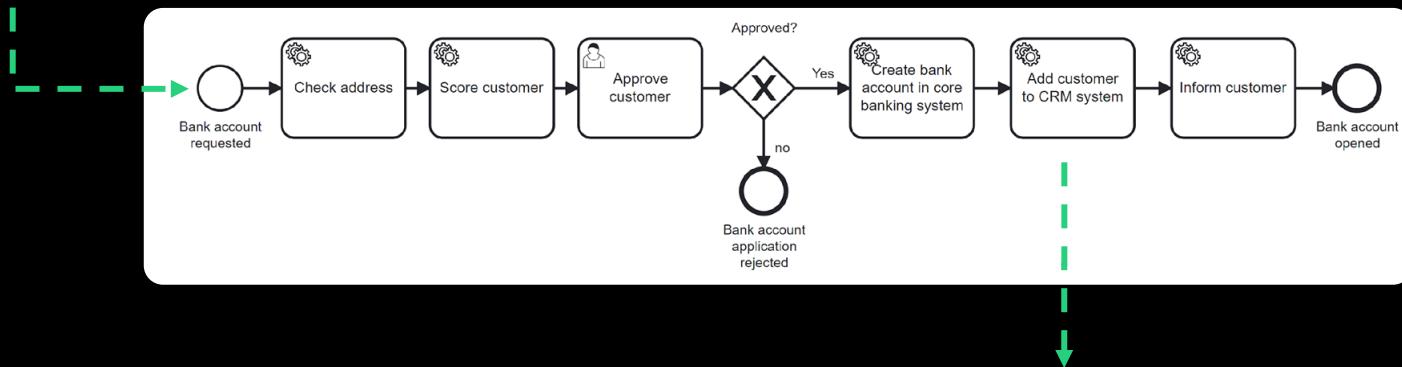
@PutMapping("/customer")
public ResponseEntity<CustomerOnboardingResponse> onboardCustomer(ServerWebExchange exchange) {
    HashMap<String, Object> variables = new HashMap<>();
    variables.put("automaticProcessing", true);
    variables.put("someInput", "yeah");

    client.newCreateInstanceCommand() //
        .bpmnProcessId("customer-onboarding") //
        .latestVersion() //
        .variables(variables) //
        .send().join();

    return ResponseEntity.status(HttpStatus.ACCEPTED).build();
}

```

Your code to provide a REST endpoint



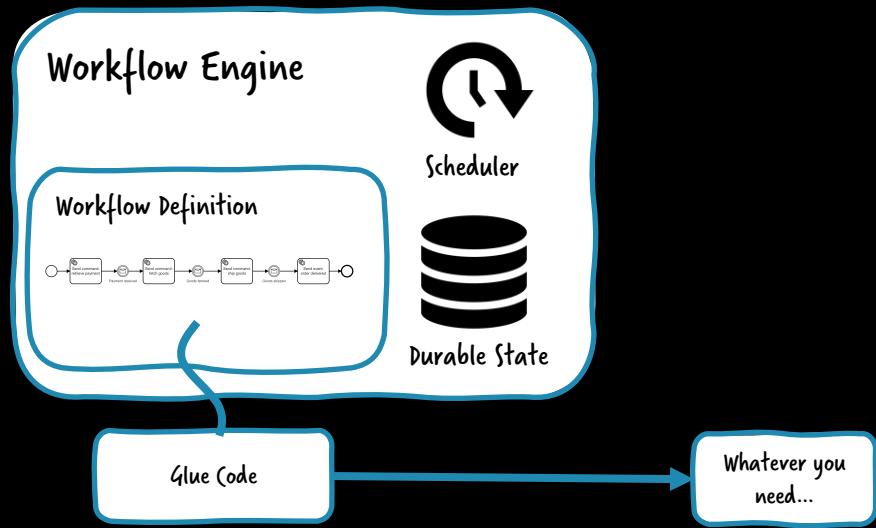
Your code to implement the REST call

```

@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/](https://github.com/berndruecker/customer-onboarding-camunda-8-springboot/)

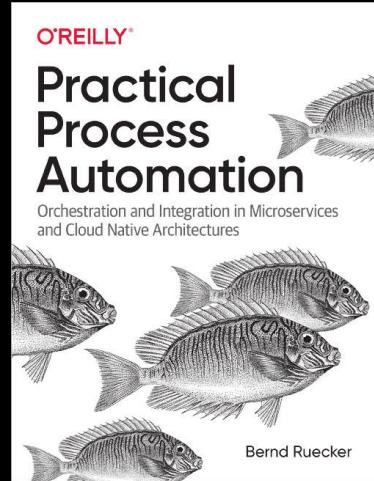


**Warning:
Contains Opinion**

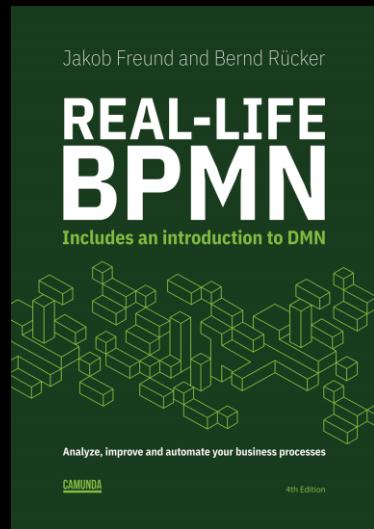


Bernd Ruecker
Co-founder and
Chief Technologist of
Camunda

mail@berndruecker.io
[@berndruecker](https://berndruecker.com)
<http://berndruecker.io/>



Jakob Freund and Bernd Rücker



Do you have to program all integration logic?

The screenshot shows a user interface for a low-code integration platform. On the left, there is a canvas with a single component: a rounded rectangle labeled "Charge credit card" with a blue border. A small "R" icon is in the top-left corner of the box. To the right of the component are several small icons: a circle, a diamond, a square, a circular arrow, a dashed line, three dots, a wrench, a trash can, and a gear. An arrow points from the left towards the component. On the right side of the screen is a configuration panel with the following sections:

- REST CONNECTOR** Charge credit card
- Authentication**: A dropdown menu with two options.
- HTTP Endpoint**: A dropdown menu with two options.
- Method**: A dropdown menu set to **GET**.
- URL**: A text input field containing `http://localhost:8099/charge`.
- Query Parameters**: A text input field with a "fx" button.
- HTTP Headers**: A text input field with a "fx" button.

Below the configuration panel, there are two descriptive text blocks:

- Map of query parameters to add to the request URL
- Map of HTTP headers to add to the request

Out-of-the-box Connectors



{...} REST API



Kafka Producer



GitLab



GitHub

 SendGrid



Amazon SQS



OpenAI



Asana

 Slack



AWS Lambda



Camunda
Operate



Google Maps

 Microsoft Teams



Amazon SNS



MessageBird



UiPath

 Google Drive



RabbitMQ



Twilio



Microsoft Power
Automate

 Automation
Anywhere



GraphQL



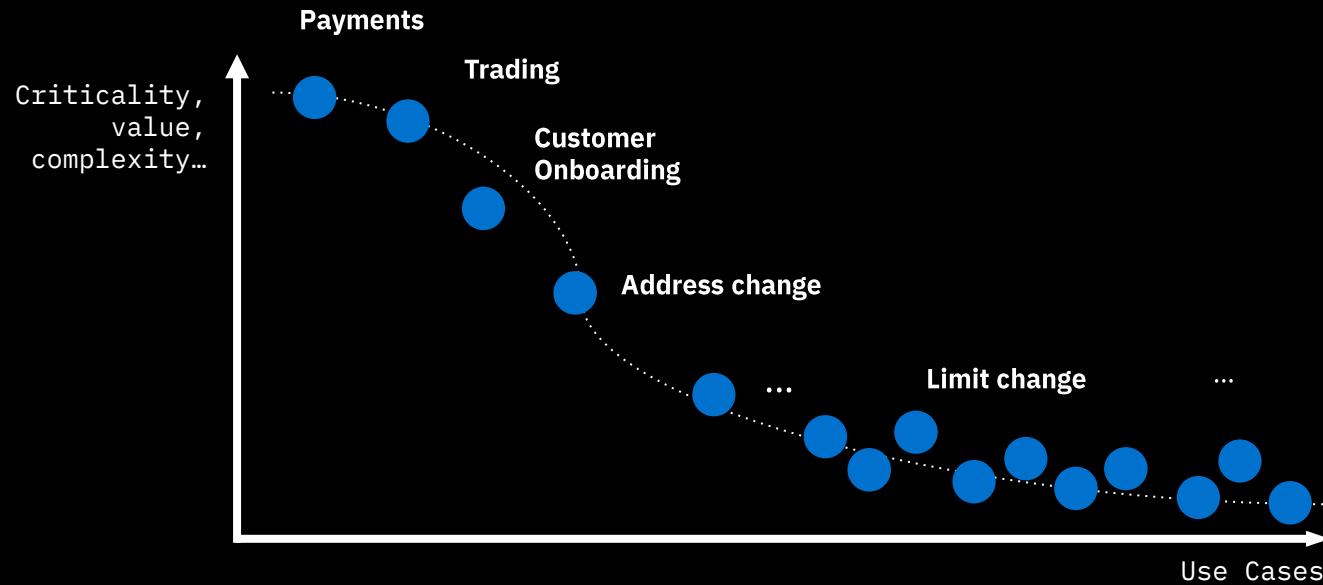
EasyPost

... 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



Yellow

Guided

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



Red

Professional Development

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



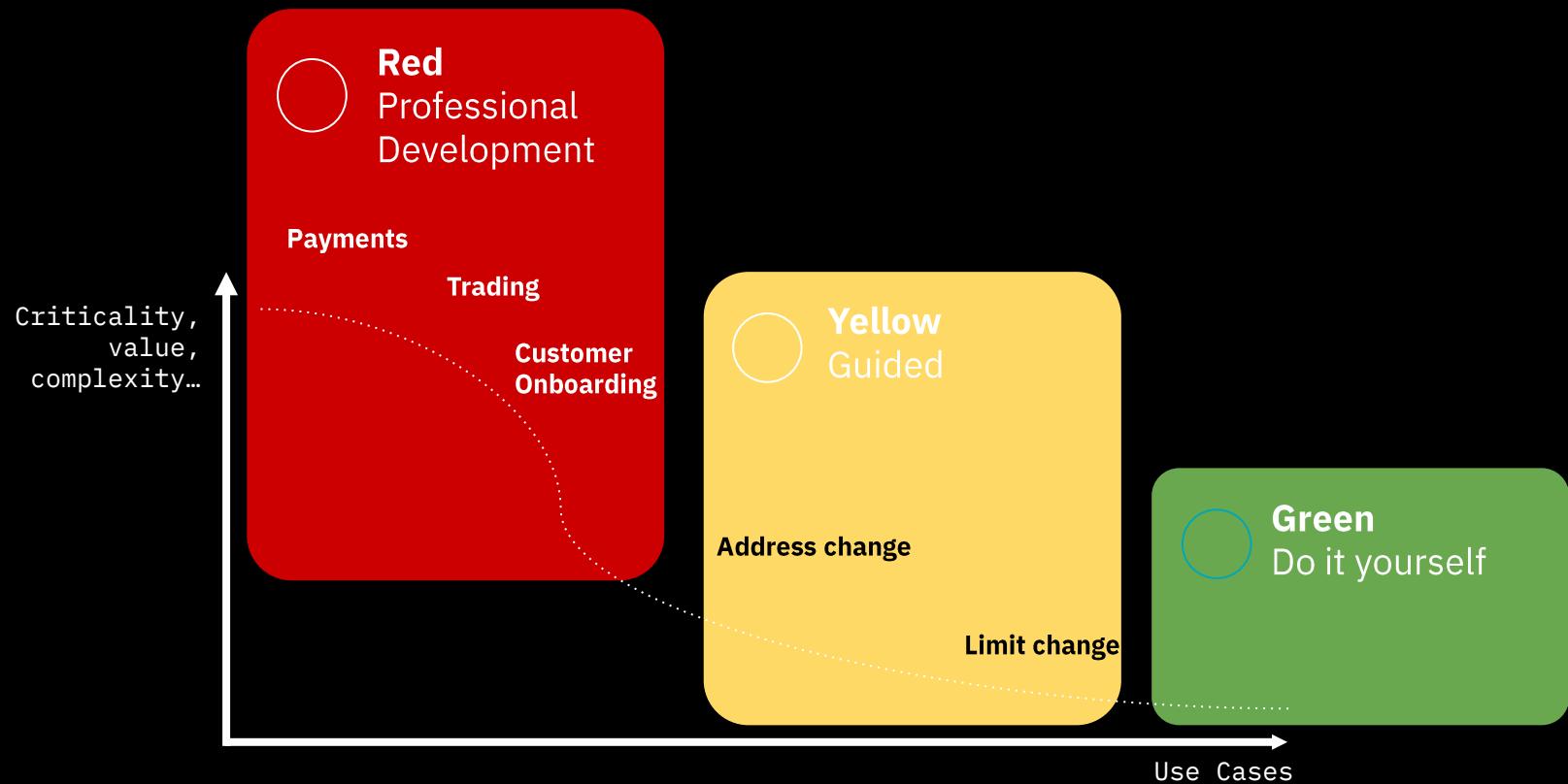
Citizen
Developer



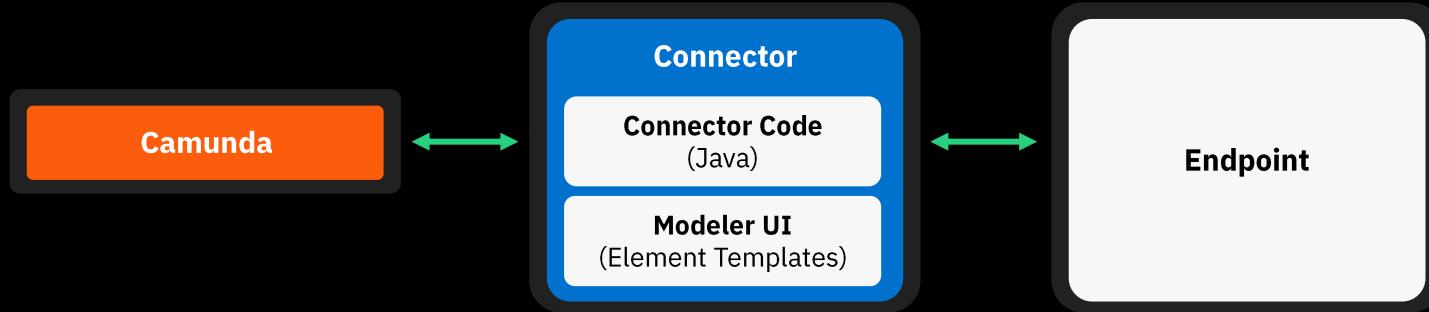
Anything in
between



Professional
Developer



What's a Connector?



```
@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": [  
                "value": "get",  
                "label": "GET",  
                "selected": true  
            ],  
            "description": "The method used for the HTTP request."  
        },  
        {  
            "id": "url",  
            "label": "URL",  
            "group": "endpoint",  
            "type": "Text",  
            "value": "https://github.com/camunda/connectors-bundle/tree/main/connectors/http-json",  
            "description": "The base URL for the REST endpoint."  
        },  
        {  
            "id": "queryParameters",  
            "label": "Query Parameters",  
            "group": "endpoint",  
            "type": "Text",  
            "value": "",  
            "description": "A map of query parameters to add to the request URL."  
        },  
        {  
            "id": "httpHeaders",  
            "label": "HTTP Headers",  
            "group": "endpoint",  
            "type": "Text",  
            "value": "",  
            "description": "A map of HTTP headers to add to the request."  
        },  
        {  
            "id": "connectTimeout",  
            "label": "Connect Timeout",  
            "group": "endpoint",  
            "type": "Text",  
            "value": "20",  
            "description": "The connect timeout for the HTTP connection."  
        }  
    ]  
}
```

REST CONNECTOR
Make a request

General

Template Applied

Authentication

Type None

Choose the authentication type. Select 'None' if no authentication is necessary

HTTP Endpoint

Method GET

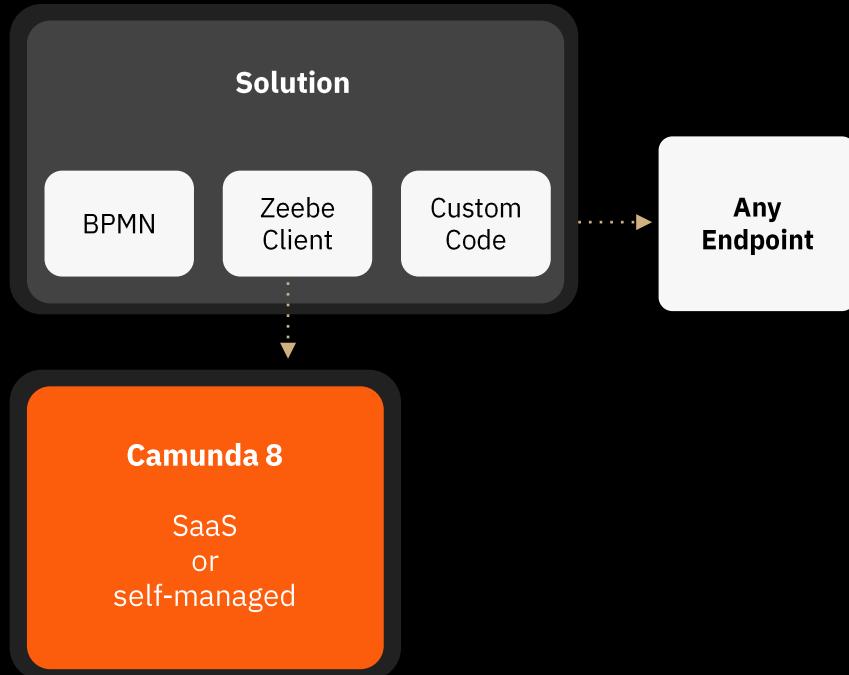
URL Must not be empty.

Query Parameters

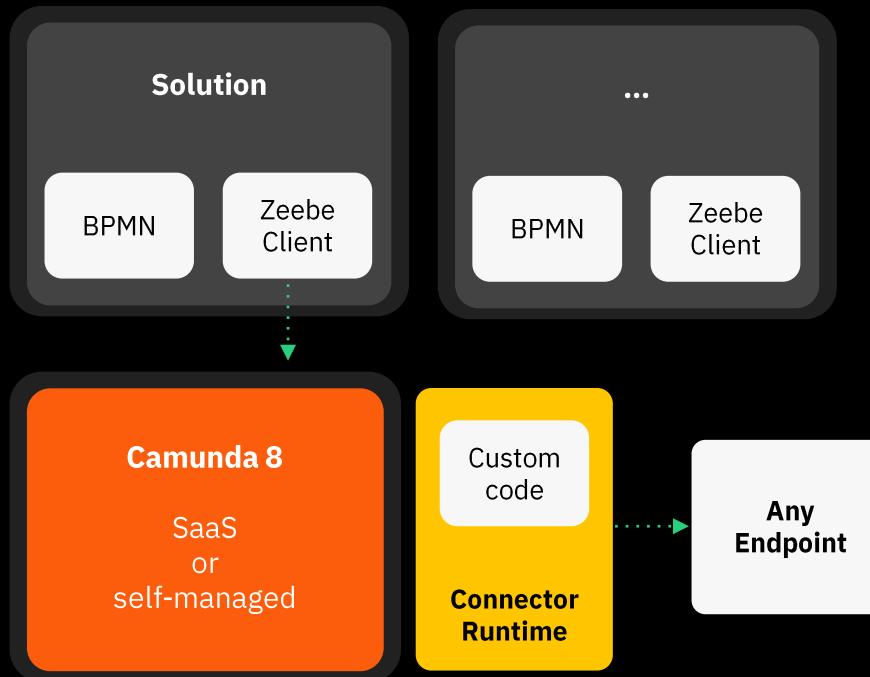
HTTP Headers

Connect Timeout Connection Timeout 20

Solution architecture example



Solution architecture example



Runs anywhere

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

Protocol Connectors

The screenshot shows the configuration interface for a REST CONNECTOR. The top bar has a 'REST CONNECTOR' icon and a 'Make a request' button. The main area is divided into sections: General, Template (selected), Authentication, Type (None), HTTP Endpoint, Method (GET), URL (empty field with error message), Query Parameters, and HTTP Headers. At the bottom are fields for Connect Timeout and Connection Timeout.

REST CONNECTOR
Make a request

General

Template **Applied**

Authentication

Type
None

Choose the authentication type. Select 'None' if no authentication is necessary

HTTP Endpoint

Method
GET

URL Must not be empty.

Query Parameters

HTTP Headers

Connect Timeout

Connection Timeout
20

Java &
JSON

Protocol > Generic System Connector

The diagram illustrates the transition from a REST connector configuration to a generic system connector configuration.

REST CONNECTOR Configuration:

- General:** REST Connector, Make a request.
- Template:** Applied (selected).
- Authentication:** Type: None. Note: Choose the authentication type. Select 'None' if no authentication is necessary.
- HTTP Endpoint:** Method: GET, URL: (redacted). Note: Must not be empty.
- Query Parameters:** Map of query parameters to add to the request URL.
- HTTP Headers:** Map of HTTP headers to add to the request.
- Connect Timeout:** Connection Timeout: 20.

Java & JSON (highlighted in orange)

TWILIO ServiceTask Configuration:

- General:** TWILIO ServiceTask.
- Template:**
- Operation:** Operation type: Send a SMS, Get message, List Messages, Authentication type: (dropdown).
- Response Mapping:** Result Variable: (dropdown), Result Expression: (dropdown). Notes: Name of variable to store the response in. Details in the documentation.
- Error Handling:** Connection Timeout: 20. Notes: Sets the timeout in seconds to establish a connection. For an infinite timeout, set to -1. Error Expression: (dropdown).

JSON (highlighted in orange)

Protocol > Generic > Specific Connectors

REST CONNECTOR
Make a request

General

Template **Applied**

Authentication

Type
None
Choose the authentication type. Select 'None' if no authentication is necessary

HTTP Endpoint

Method
GET

URL
Must not be empty.

Query Parameters
Map of query parameters to add to the request URL

HTTP Headers
Map of HTTP headers to add to the request

Connect Timeout
Connection Timeout
20

Java & JSON

TWILIO
ServiceTask

General

Template

Operation

Operation type
Send a SMS
Get message
List Messages
Authentication type

Response Mapping

Result Variable
Name of variable to store the response in. Details in the [documentation](#)

Result Expression
Expression to map the response into process variable
Details in the [documentation](#)

Error Handling

Connection Timeout
20
Sets the timeout in seconds to establish a connection. If set to 0 or higher, it will wait for an infinite timeout

Error Expression

JSON

SEND SMS
ServiceTask

General

Template

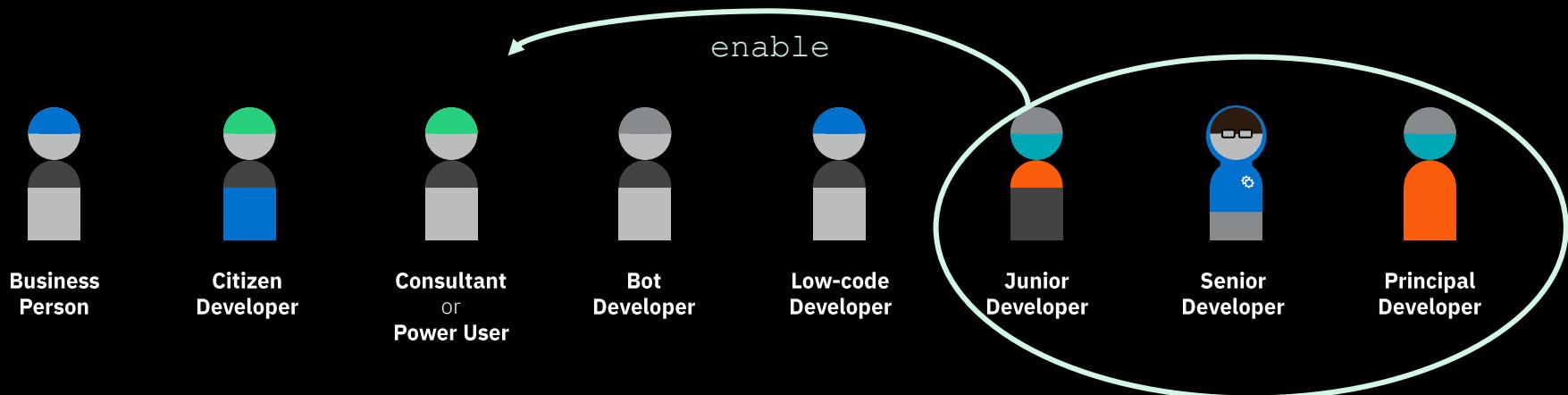
Input

Message Text
Must not be empty.
The content of the message that will be sent

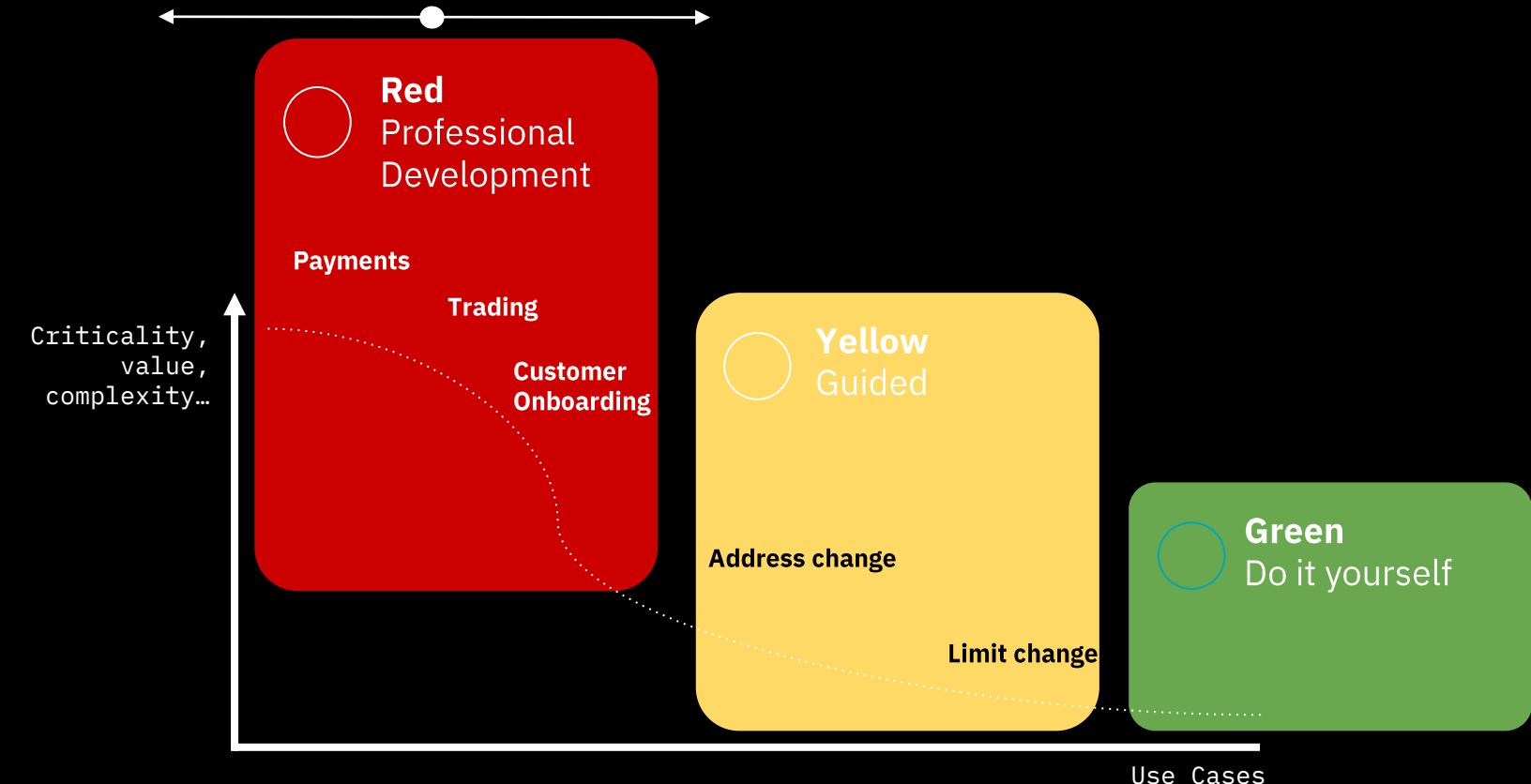
To Number
Must not be empty.
The recipient's phone number

JSON

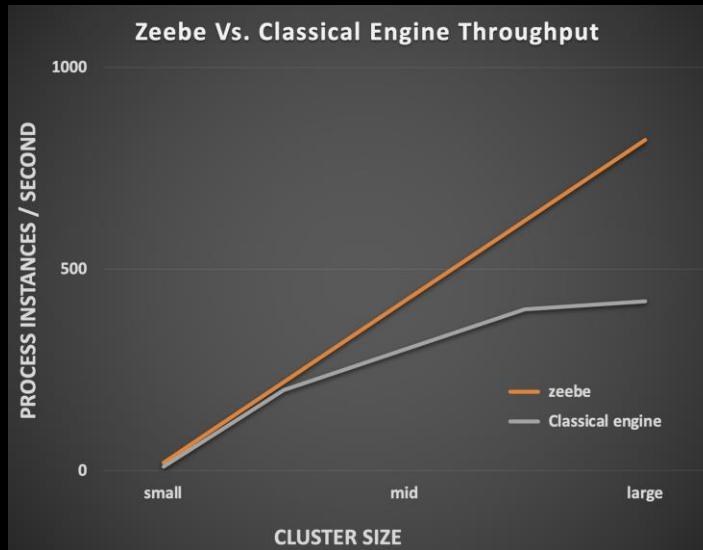
Enabling more roles to participate



Unlocking more use cases



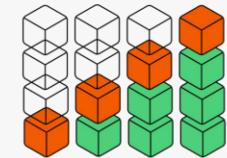
But also unlock more red use cases



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

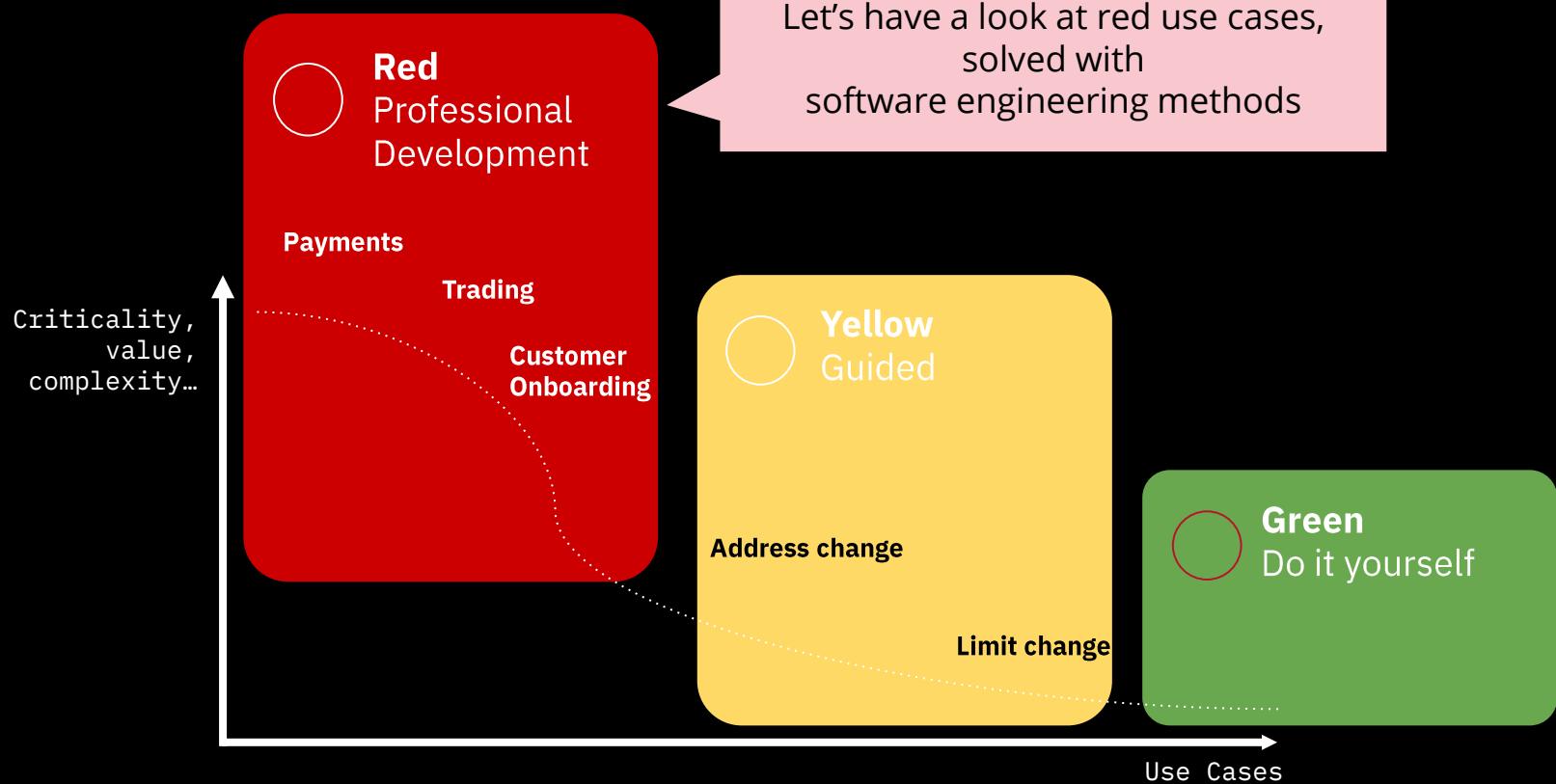


Pricing flexibility and transparency

Download the report:



The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester's call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave™. Information is based on best available resources. Opinions reflect judgment at the time and are subject to change.



Recently in software engineering...



Recently in software engineering...

Microservices,
Serverless, ...

Domain Driven Design
(DDD)

Cloud, SaaS, ...

Product mindset

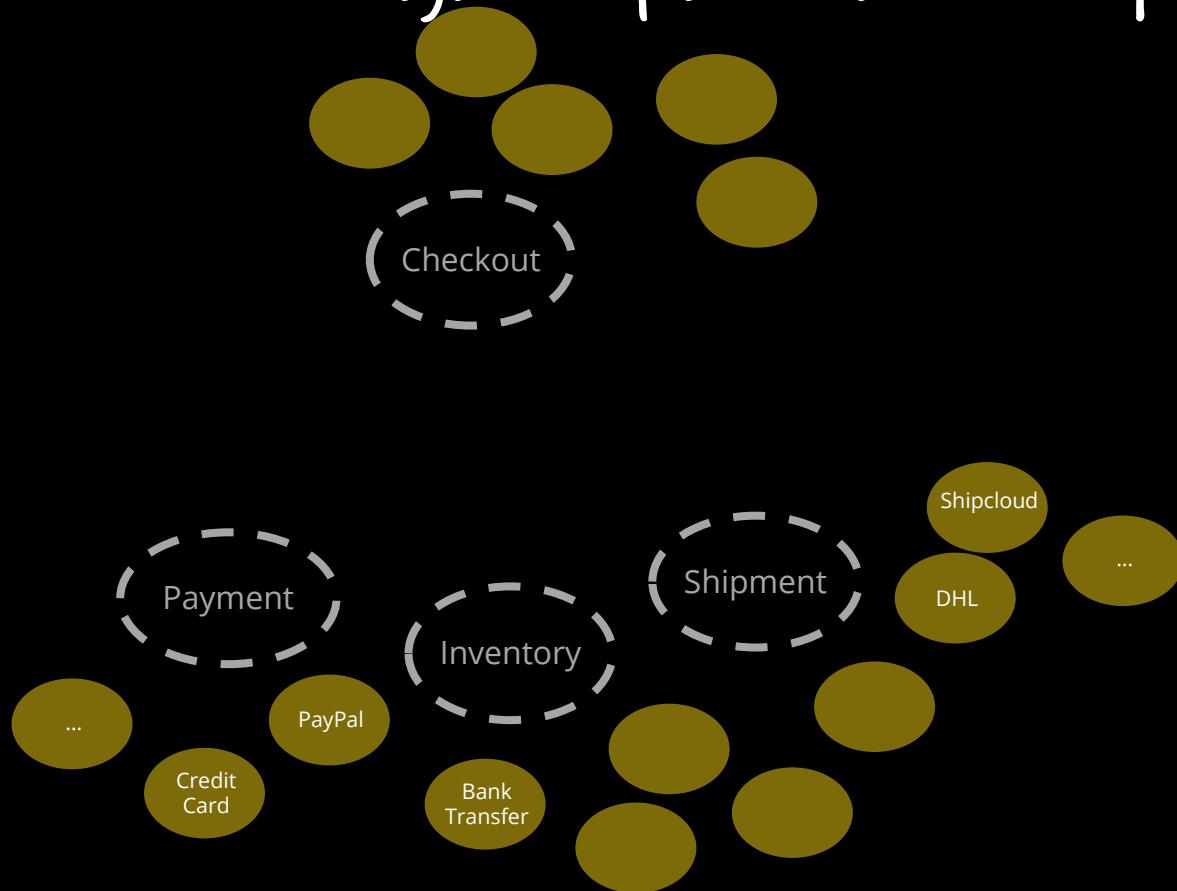
Agile, DevOps, ...

Team topologies

Building smaller components...



...with an ecosystem of services and components



Autonomy to speed up development



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

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 lously 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

(horeography is great!



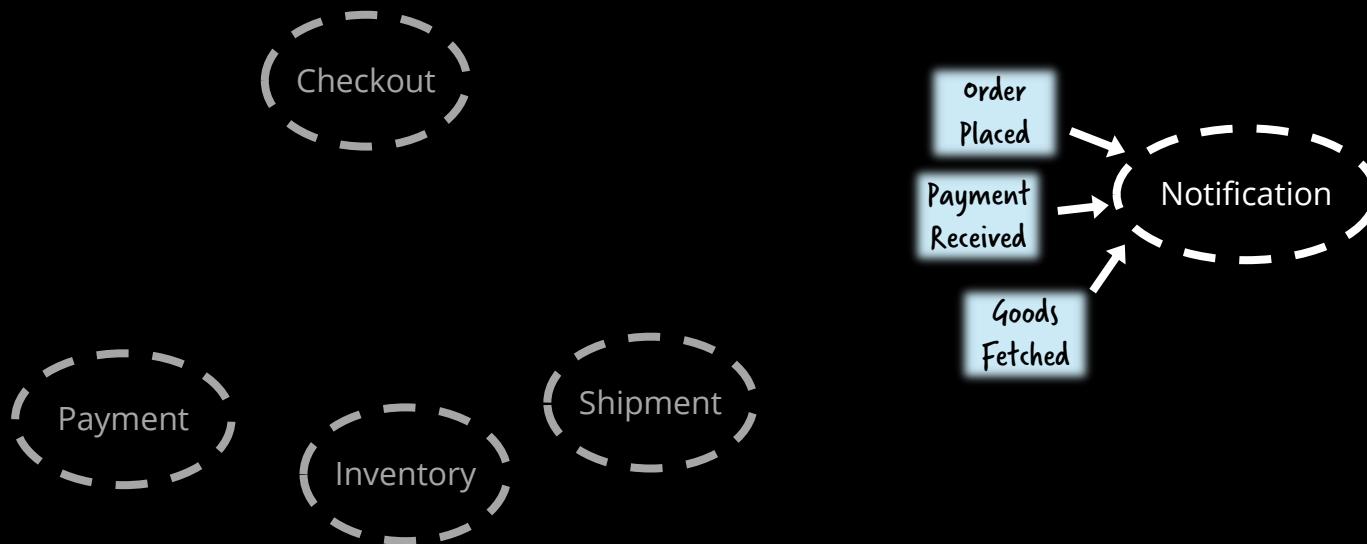
Photo by Lijian Zhang, under Creative Commons SA 2.0 License

What we wanted

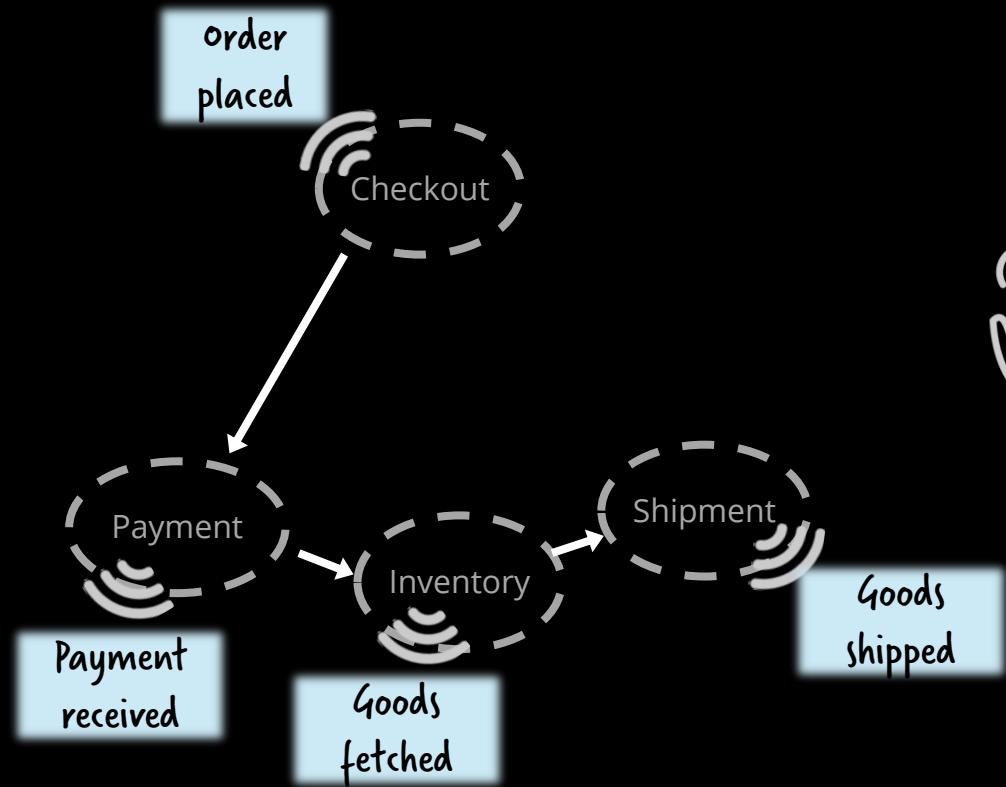


vs. what we got

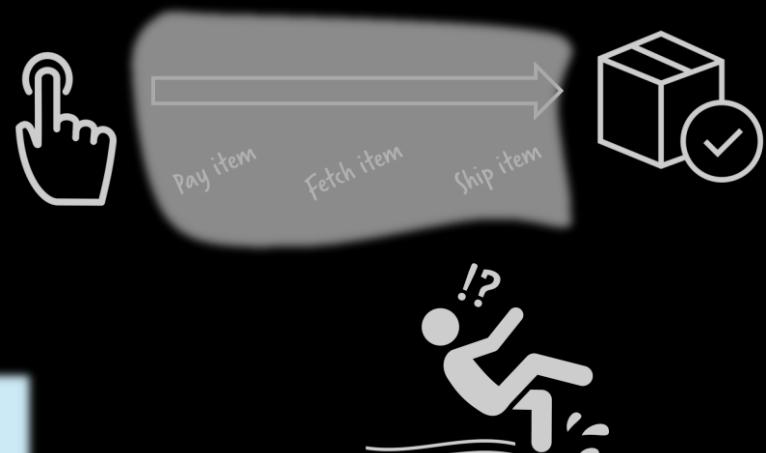
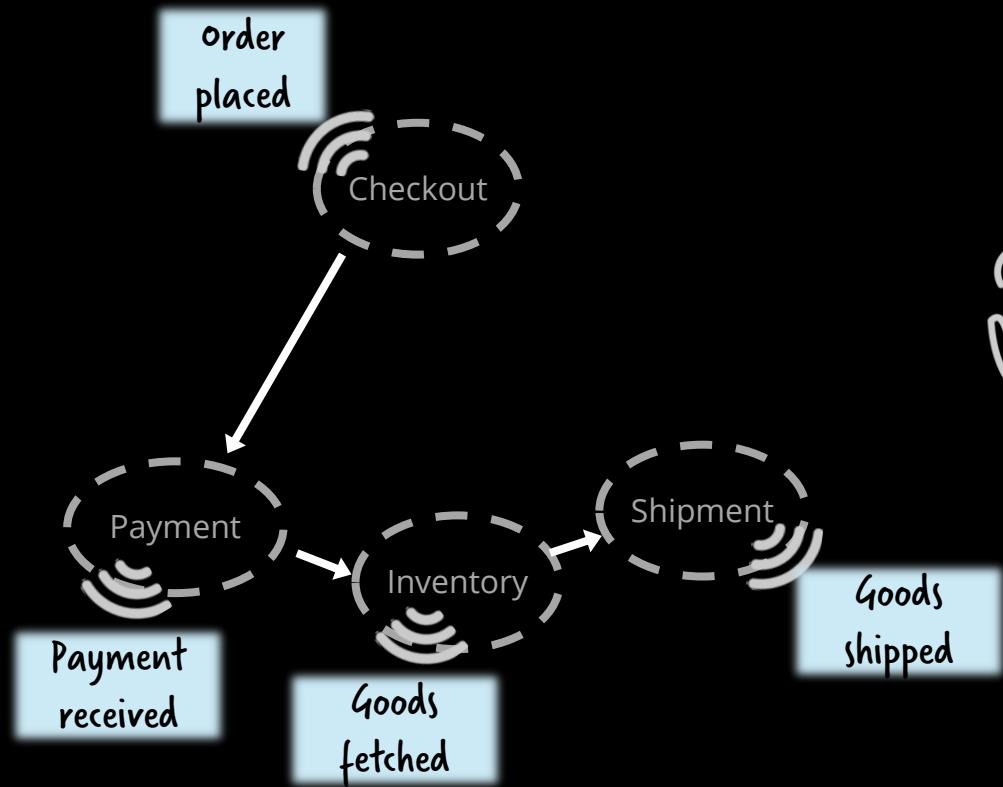
Event-driven



Peer-to-peer event chains



Peer-to-peer event chains



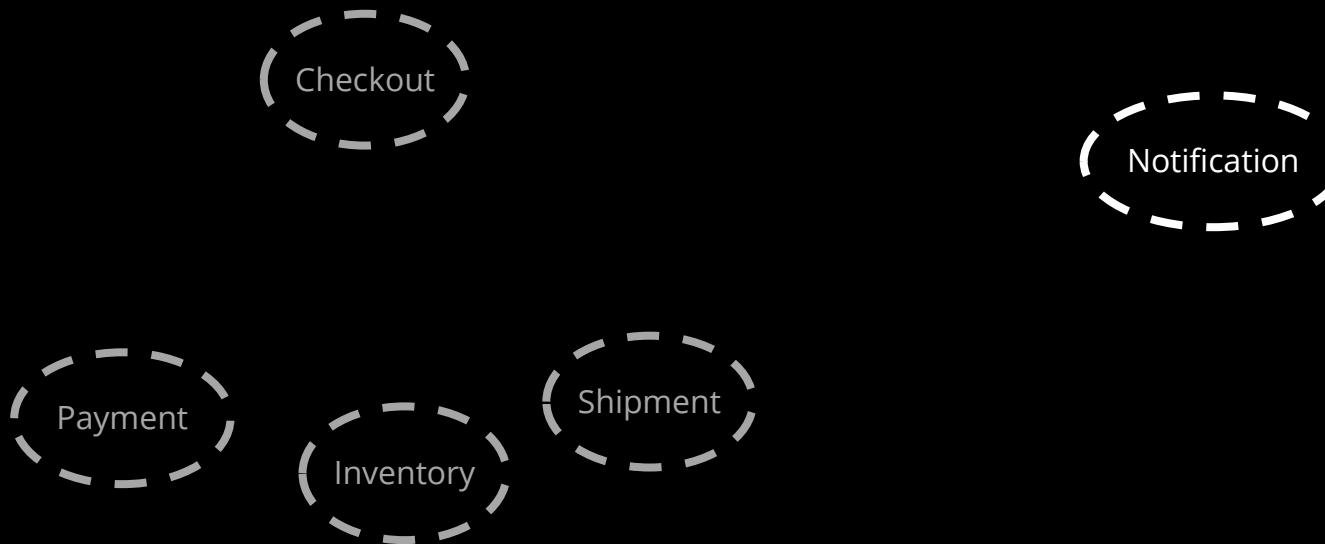
We were suffering from
Pinball machine Architecture



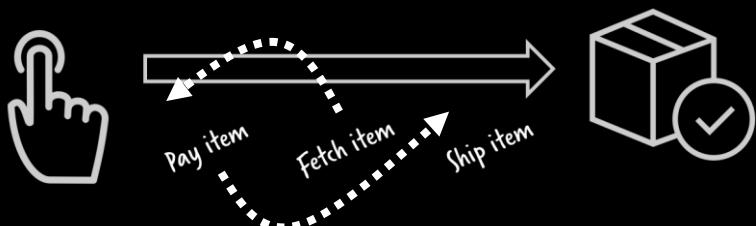
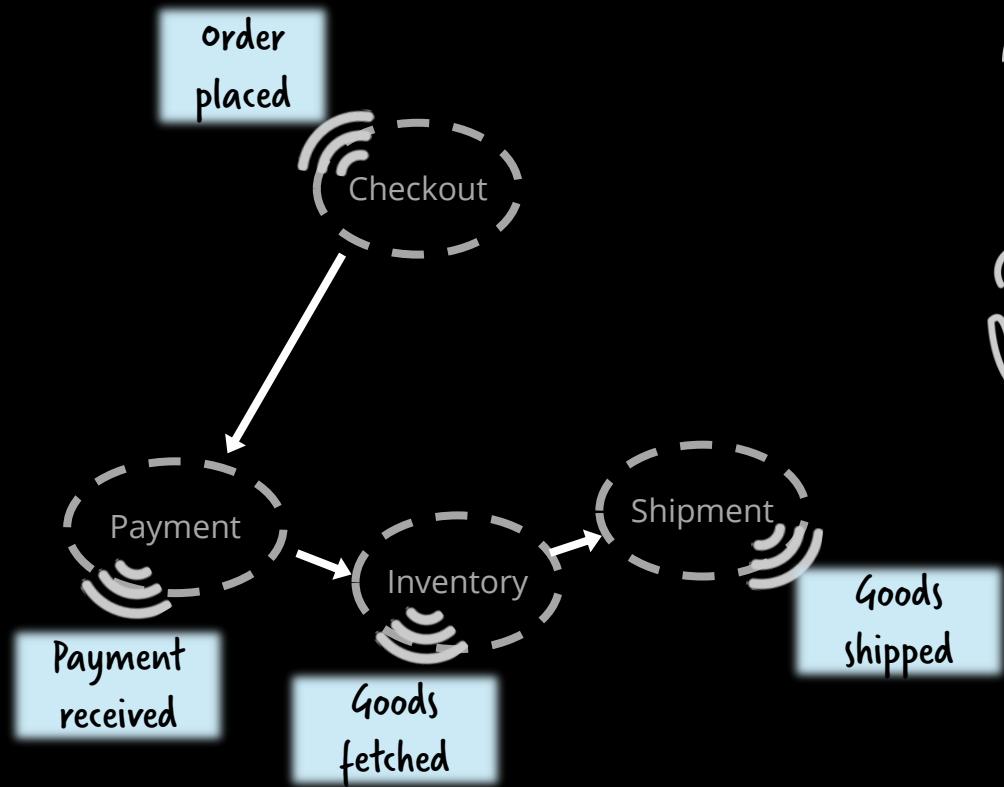
Pinball Machine Architecture



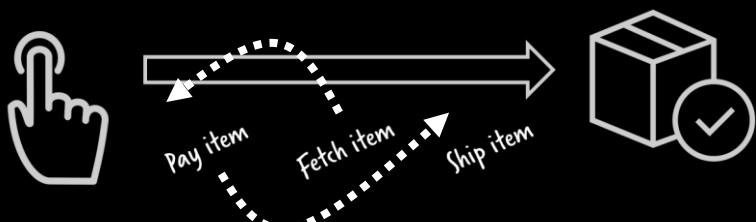
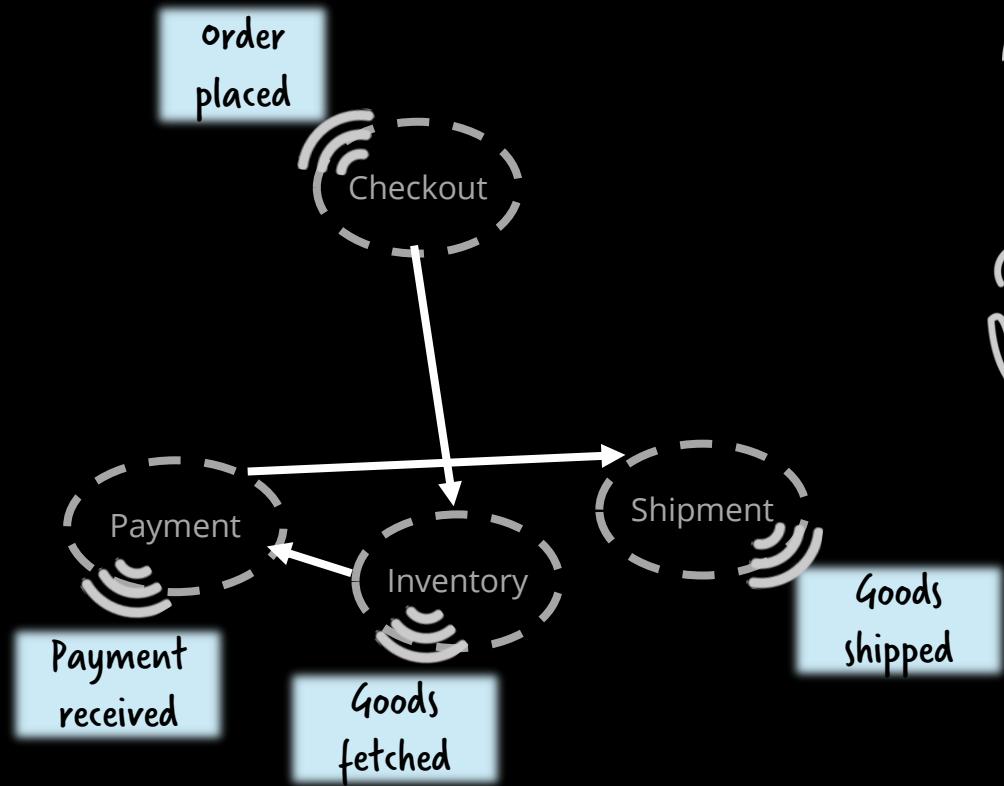
„What the hell just happened?“



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.

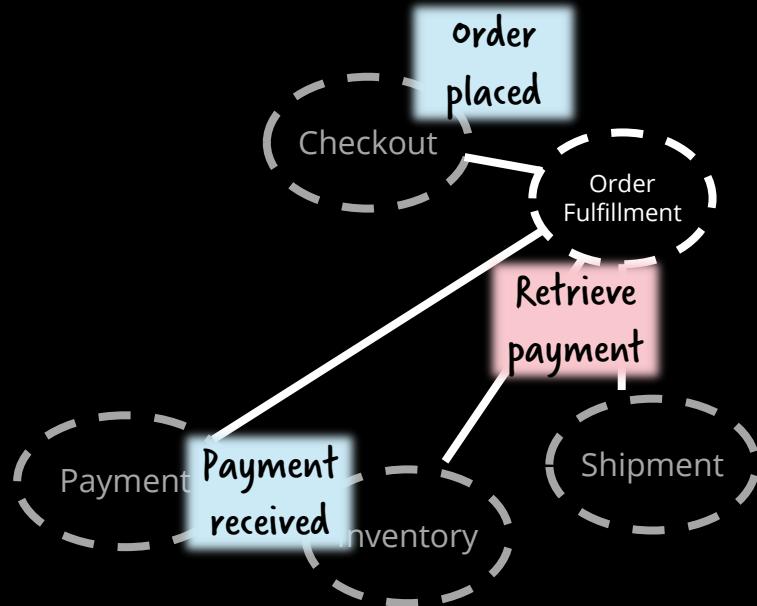


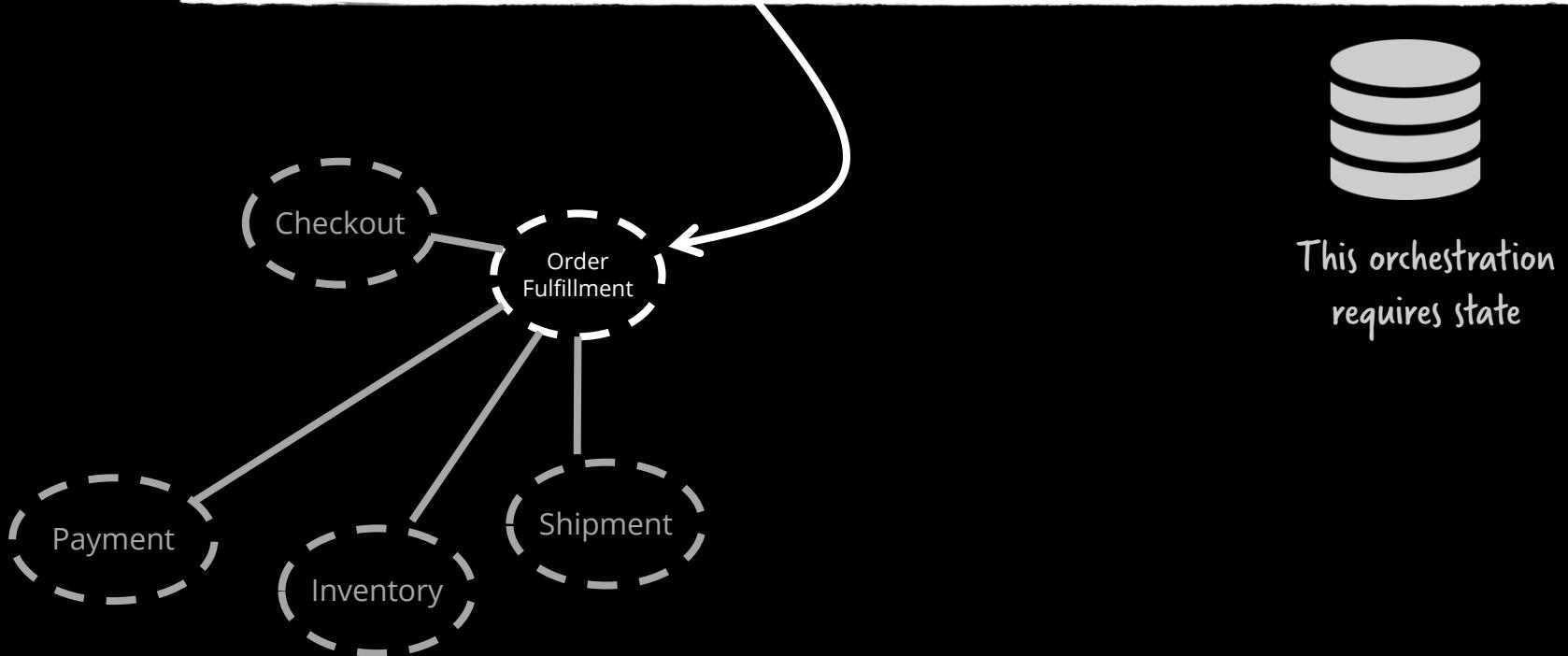
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.



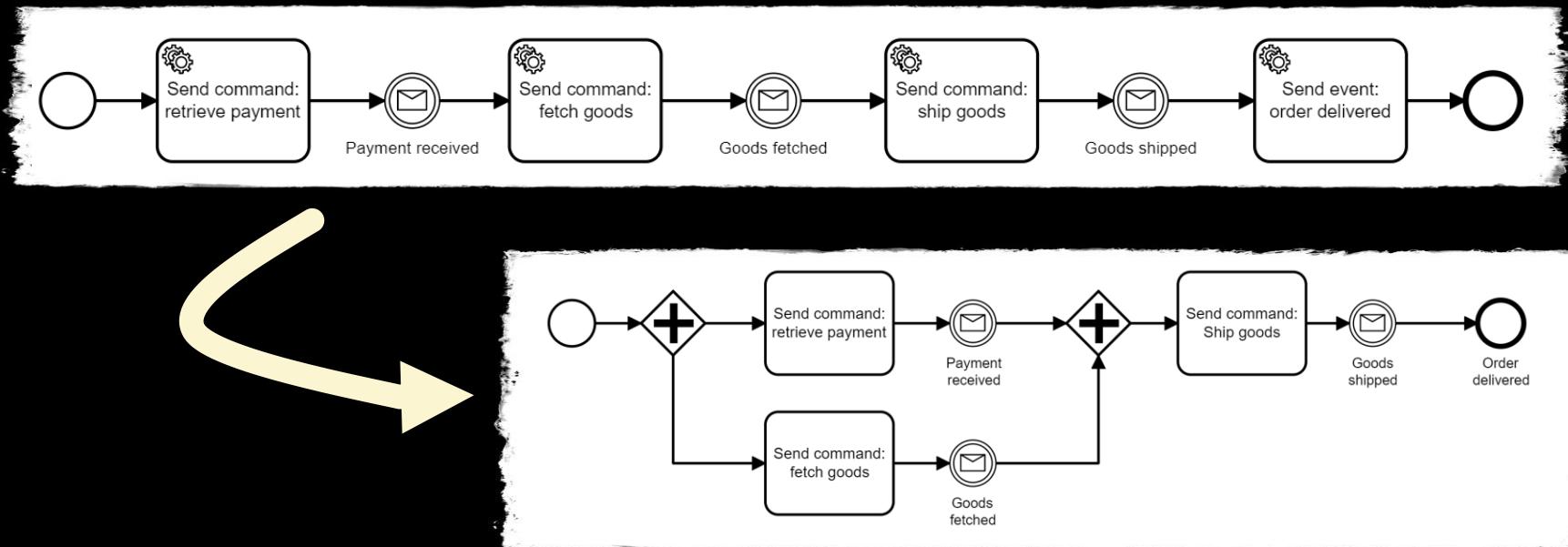
The danger is that it's very easy to make nicely decoupled systems with event notification, without realizing that you're losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

Extract the domain logic around order fulfillment

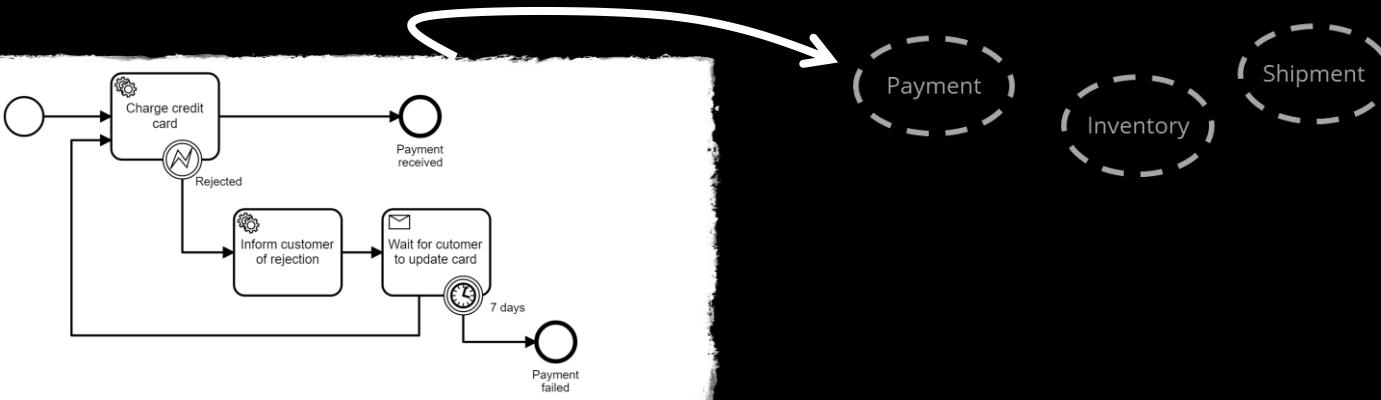
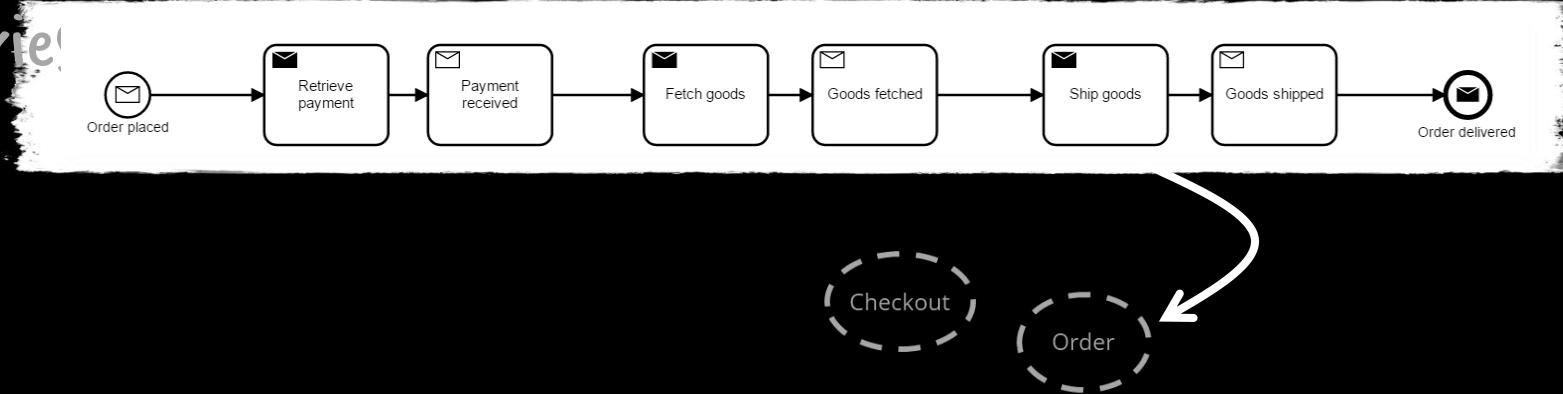




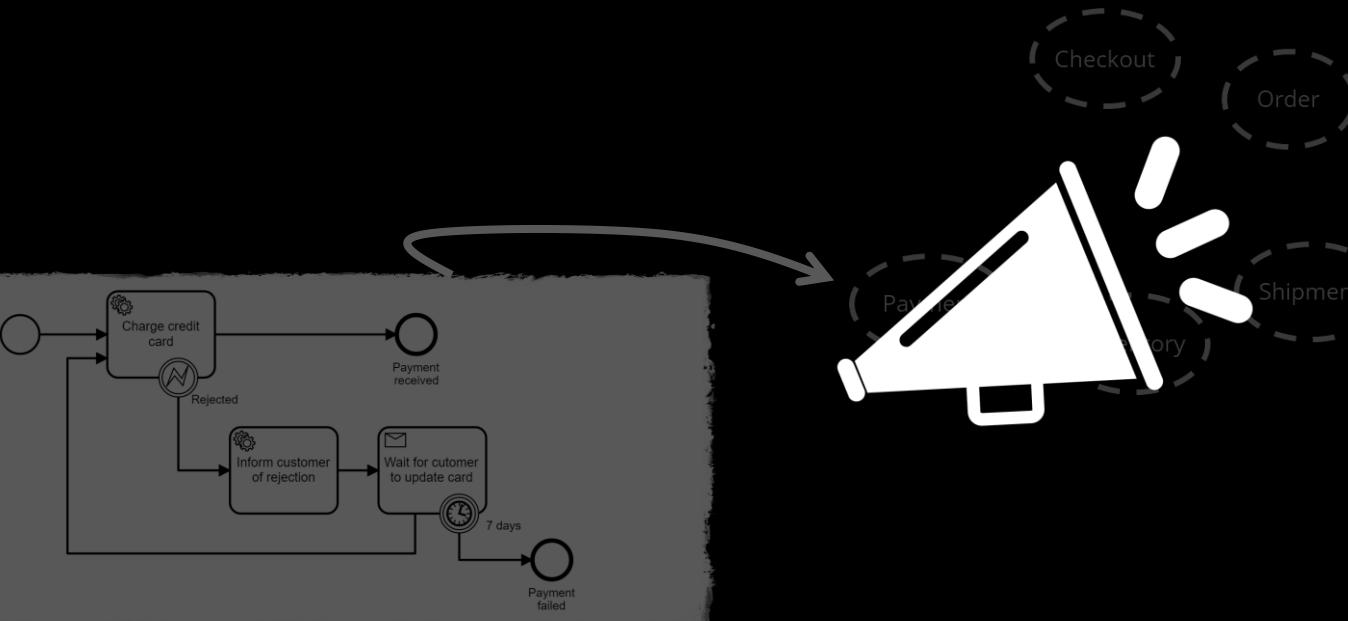
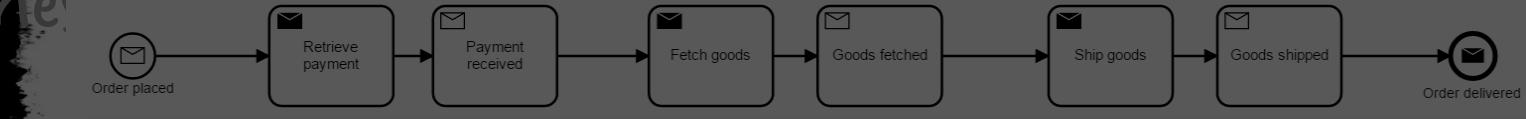
Now it is easy to change the orchestration logic



Processes are domain logic and live inside service boundaries!



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 lously 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



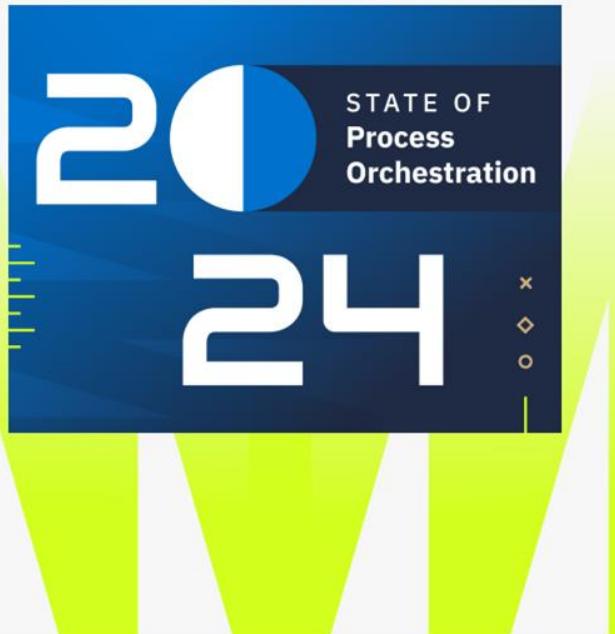
There is trouble in software engineering land!

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.

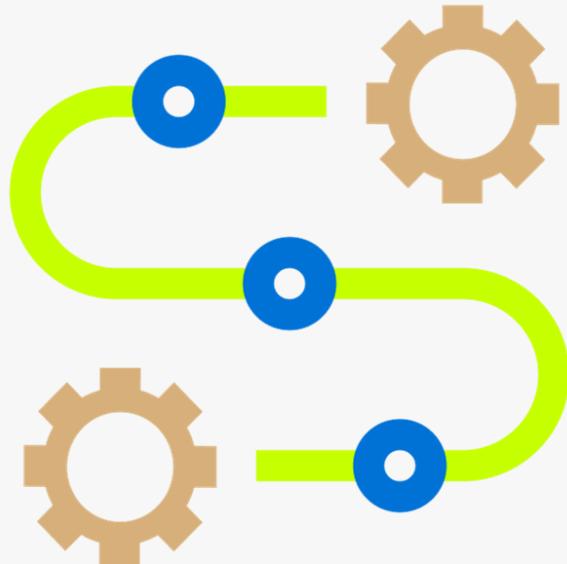
This is the fourth year we've produced this 'State Of' report, which also compares shifts from year to year and how organizations have matured in their automation initiatives.

**Download
the report**



<https://bit.ly/sopo-2024>

Processes are getting more complex



51%

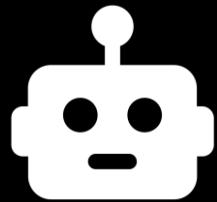
of IT decision makers say processes spanning multiple systems is a reason for process automation complexity

↗ (up from 41% in 2023)

60%

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

Is everything a Microservice?



RPA & Bots

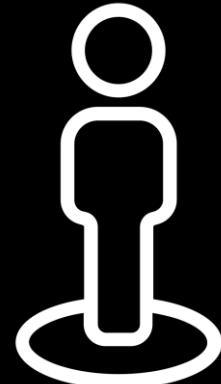
SaaS Services



Legacy Systems

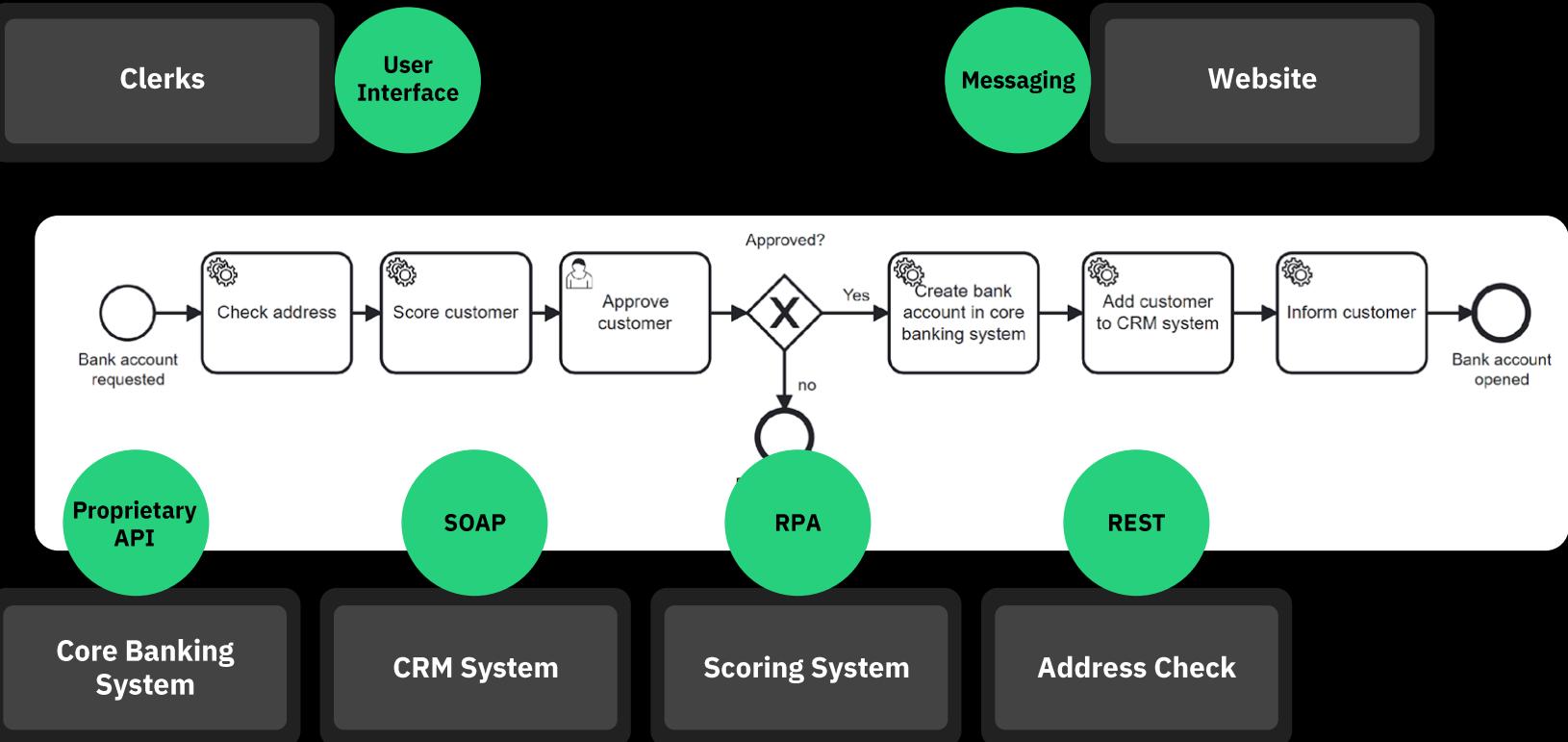


Front-end
applications

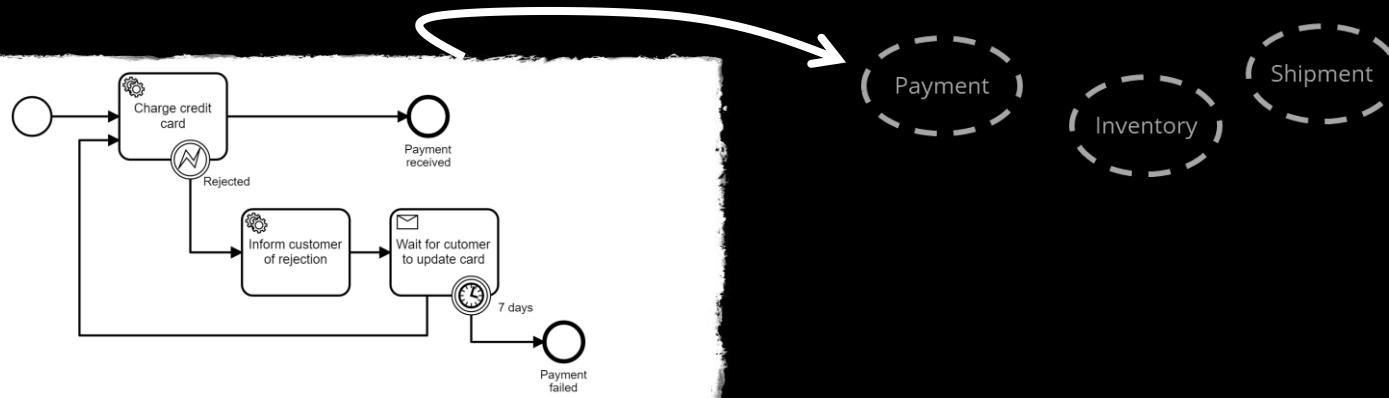
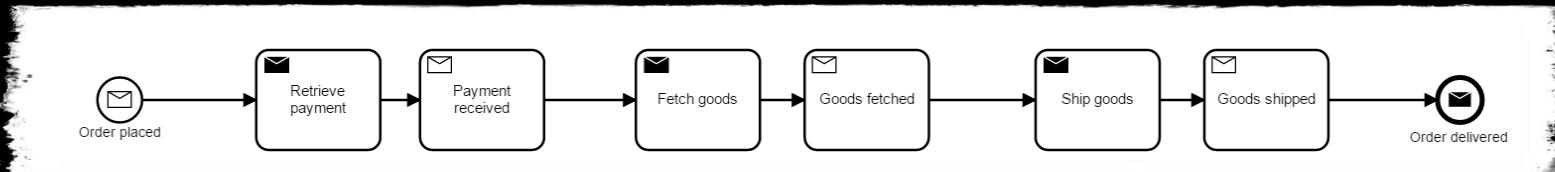


Humans

You need to integrate all endpoints!



Processes can still live in microservices



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!



Photo by born1945, available under Creative Commons BY 2.0 license.



Projects and Groups

Members

Certified partners and providers

Serverless

Wasm

VIEW MODE

Grid

Card

ZOOM



The flipside...

HELM CNCF GRADUATED	Backstage CNCF INCUBATING	Buildpacks.io CNCF INCUBATING	dapr CNCF INCUBATING	KubeVela CNCF INCUBATING
KubeVirt CNCF INCUBATING	OPERATOR FRAMEWORK CNCF INCUBATING	codezero	CARVEL HABITAT	KubePlus
Eclipse One*	fencel	Gefyra	Gradle	itopia
KUBERMINER	KubeOrbit	KUDU	lagoon	Kaniko
Knative	KONVEYOR	ko	MICROCKS	mirrorm
Nocatson	Open Application Model	OPENAPI INITIATIVES	Packer	plural
QUARKUS	raftt	sealer	ServiceComb	SHIPWRIGHT
TIL*	VMware Application Catalog		SKAFFOLD	squash
			Tanka	Terraform

KV CNCF GRADUATED	Vitess CNCF GRADUATED	ClickHouse	Cockroach Labs	Ignite	ArangoDB	BIGCHAINDB	cassandra
Databend	dgraph	druid	EDB	Foundries	IgniteDB	CRUX	Crunchy Data
DB2	Igz	Infinispan	InterSystems	Kudu	MariaDB	SQL Server	MongoDB
MySQL	neo4j	NuoDB	OceanBase	opencaxdb	OpenDemini	Oracle	OrientDB
Percona	Pilosa	PostgreSQL	Presto	Drabant	Dubole	Redis	RethinkDB
Scylla	Seata	Timescale	SingleStore	AWS SQL	SoftLayer	SAP	ScalrDB
Tarantool	TDengine	Timescale	Valid	Vertical	VoltDB	Weaviate	YDB
YugabyteDB							

argo CNCF GRADUATED	flux CNCF GRADUATED	keptn CNCF INCUBATING	OpenKruise CNCF INCUBATING
AppVeyor	CI	Bamboo	Brigade
CircleCI	Jenkins	Codefresh	Buildkite
Catturahub	GitLab	GitNoss	Travis CI
CircleCI	Go	Codefresh	Harness
Jenkins	Helm	Maven	TeamCity
Kubernetes	Ubi	Mergify	OpenShift
Oceanus	k6	Liquibase	TeamCity
Ozone	Kepler	Mergify	OpenShift
Piped	Raze	Semaphore	Spacelift
Tube-Buster	Testfire	Travis CI	Werf
			TeamCity

agol	CloudEvents
Akuity	NiFi
Beam	Apache Beam
Fluvio	Apache Beam
Kafka	Apache Kafka
Kafka-MQ	Apache Kafka
Polyflow	Apache Beam
Pravega	Apache Beam
TeamCity	Apache Beam

Flink	Apache Flink
Kubernetes	Apache Kubernetes
KuberMQ	Apache KuberMQ
Polyflow	Apache Polyflow
TeamCity	Apache TeamCity
Tremor	Apache Tremor
Userver	Apache Userver

gRPC	Apache gRPC
TARS	Apache TARS

CoreDNS	etcd
Kong	Apache etcd

KEDA CNCF GRADUATED	kubernetes CNCF GRADUATED	Crossplane CNCF INCUBATING	KARMADA CNCF INCUBATING	knative CNCF INCUBATING	VOLCANO CNCF INCUBATING	MESOS	AWSVPC	capsule	AWS Lambda
KCP	Kodestr	kube-green	Kubewharf	kubernetes	Kubelet	Nomad	Open Cluster Management	Prefect	Serverless Devs
Upbound	WasmCloud		Slack	Slack	Slack	ClusterHQ	Amazon EKS	AWS Lambda	AWS Lambda
AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda	AWS Lambda

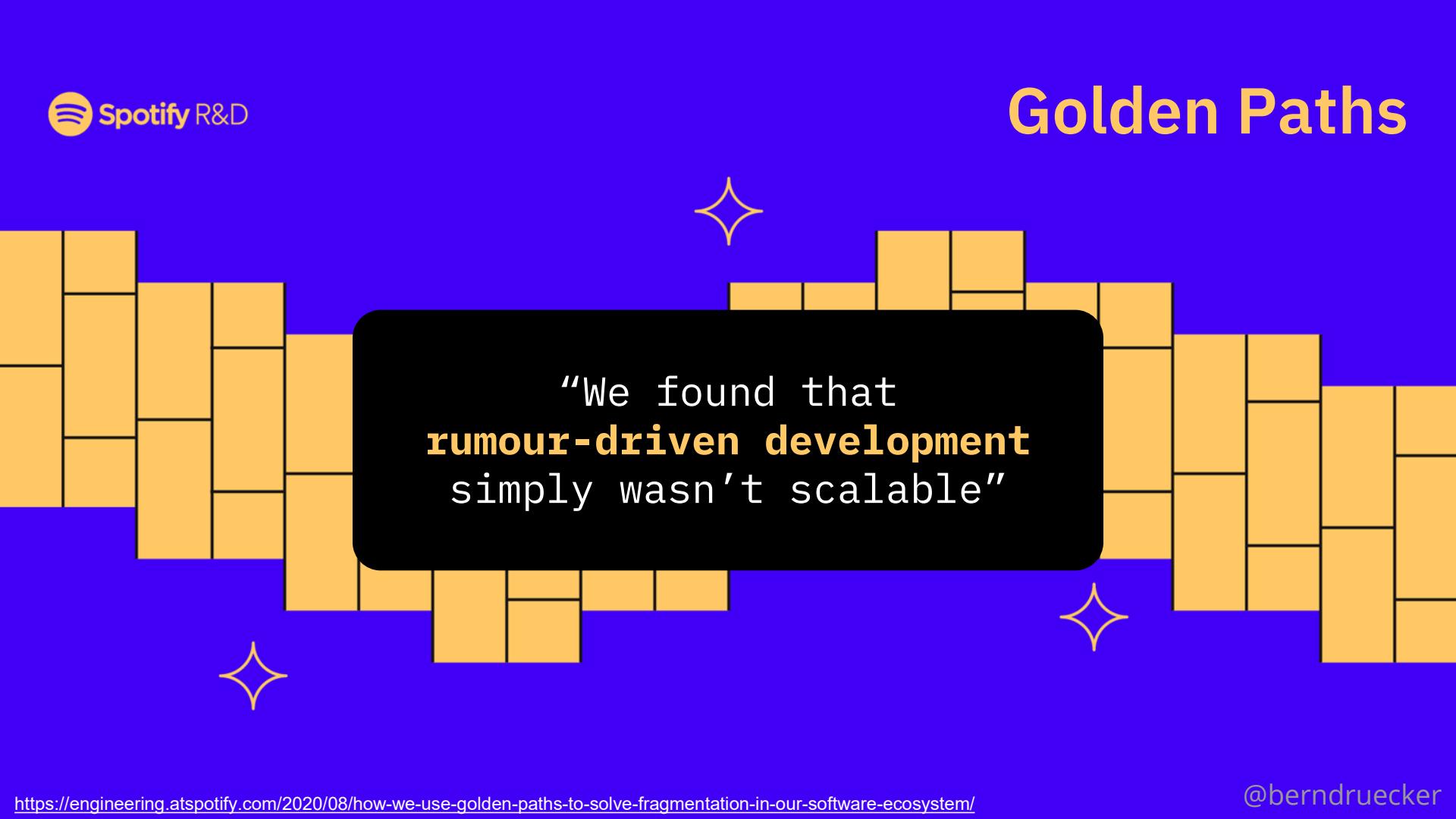
Istio CNCF GRADUATED	Linkerd CNCF GRADUATED	Kiali	Consul	AWS App Mesh	AWS Service Mesh
Nasp	Open-Serviwo	AWS App Mesh	AWS Service Mesh	AWS App Mesh	AWS Service Mesh
OpenTelemetry	Sentinel	AWS App Mesh	AWS Service Mesh	AWS App Mesh	AWS Service Mesh
SlimEx	TSB	AWS App Mesh	AWS Service Mesh	AWS App Mesh	AWS Service Mesh

Envoy CNCF GRADUATED	Contour CNCF INCUBATING	AVI Network	BFE	Caddy	Citrix	HAProxy	Inlets	MetalLB	MOSN	NGINX
OpenELB		GreenWelly	Sangfor Shinx	Sentinel	Skipper	NOVA	Tengine	Traefik proxy		

Emissary Ingress CNCF INCUBATING	oscale	Akana	APIoak	APISIX	Espresso	Saaras	Globus	Hongli	Kigress
Kong	KrakenD	Kubernetes Gateway	Kubernetes Gateway	Lunardev	MuleSoft	ngrok	TiKV	WS-O2	

XLine	CoreDNS CNCF GRADUATED	etcd CNCF GRADUATED
-------	---------------------------	------------------------

Golden Paths



“We found that
rumour-driven development
simply wasn’t scalable”

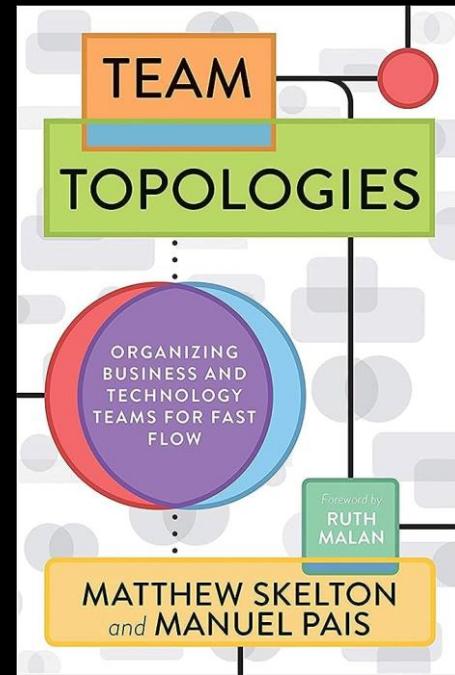


The Speed Paradox

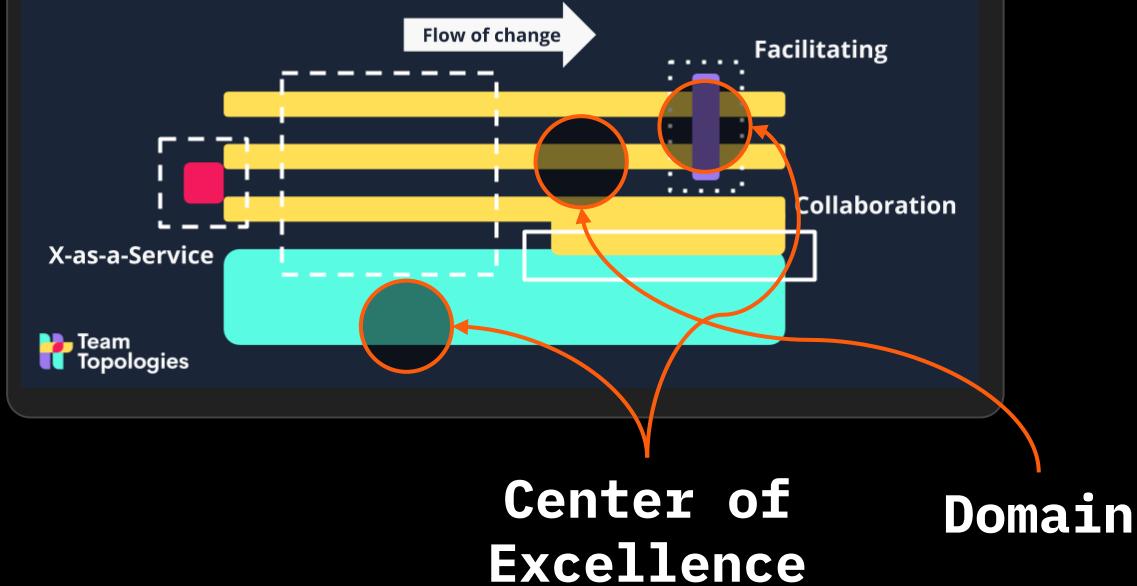
At Spotify, we've always believed in the speed and ingenuity that comes from having autonomous development teams. But as we learned firsthand, the faster you grow, the more fragmented and complex your software ecosystem becomes. And then everything slows down again.

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.

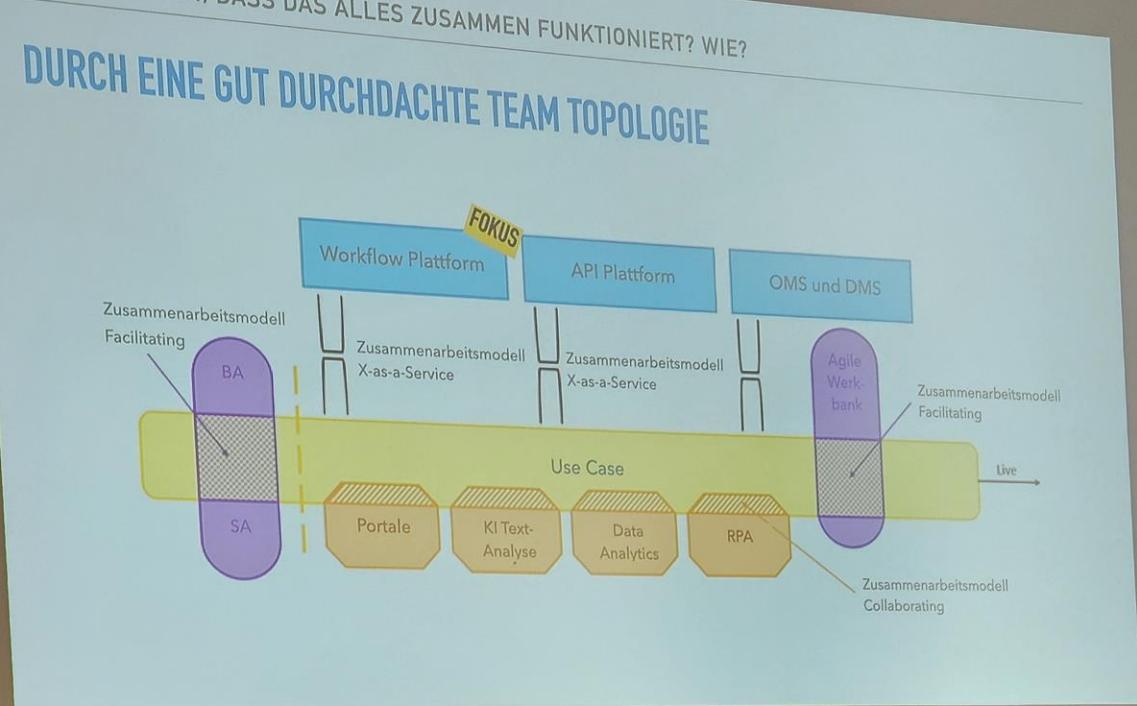


3 core interaction modes



Gestern in München

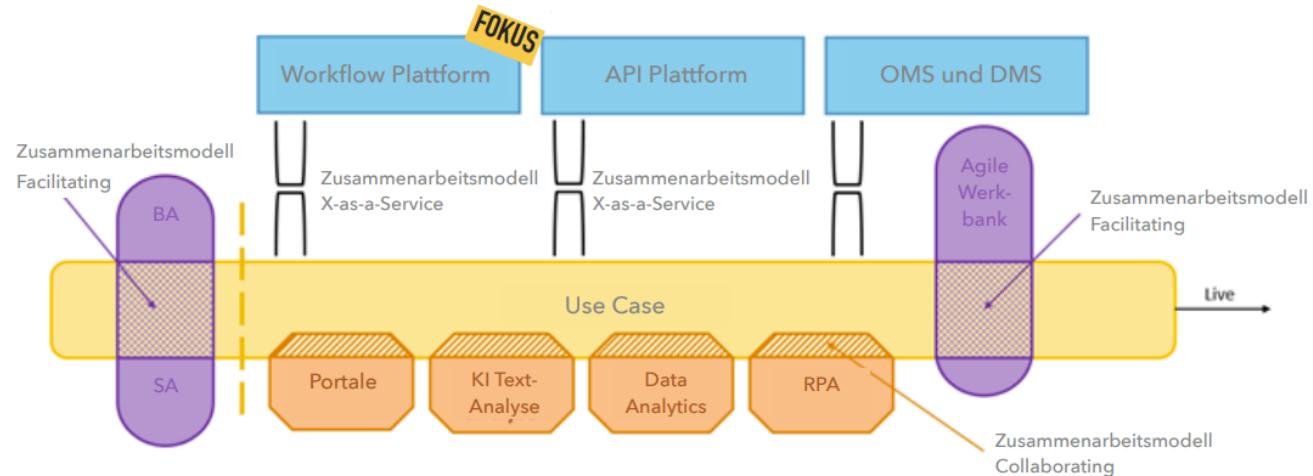
CAMUNDA
Der universelle
Prozess-
Orchestrator



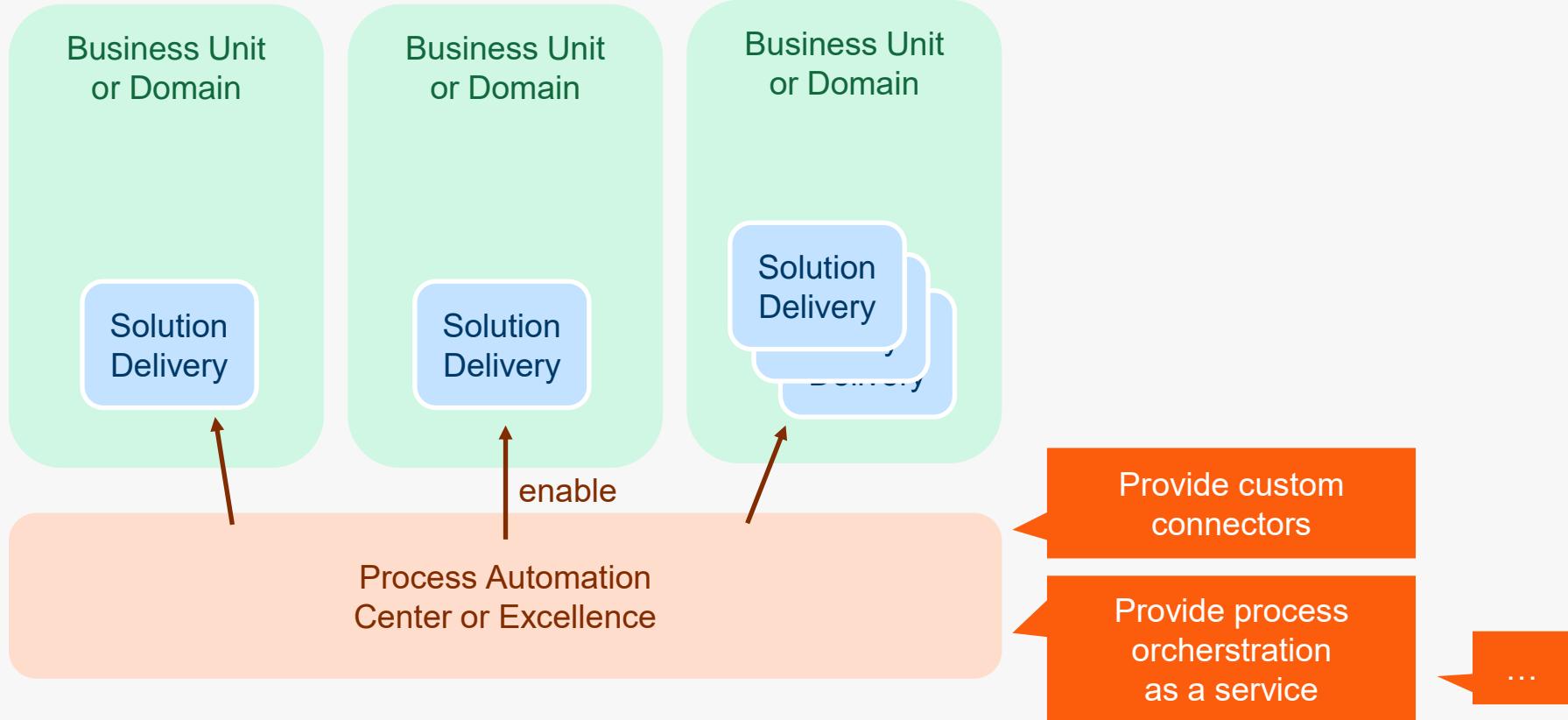
DIGITALE ENABLER – WORKFLOW COE IN A NUTSHELL

DAFÜR SORGEN, DASS DAS ALLES ZUSAMMEN FUNKTIONIERT? WIE?

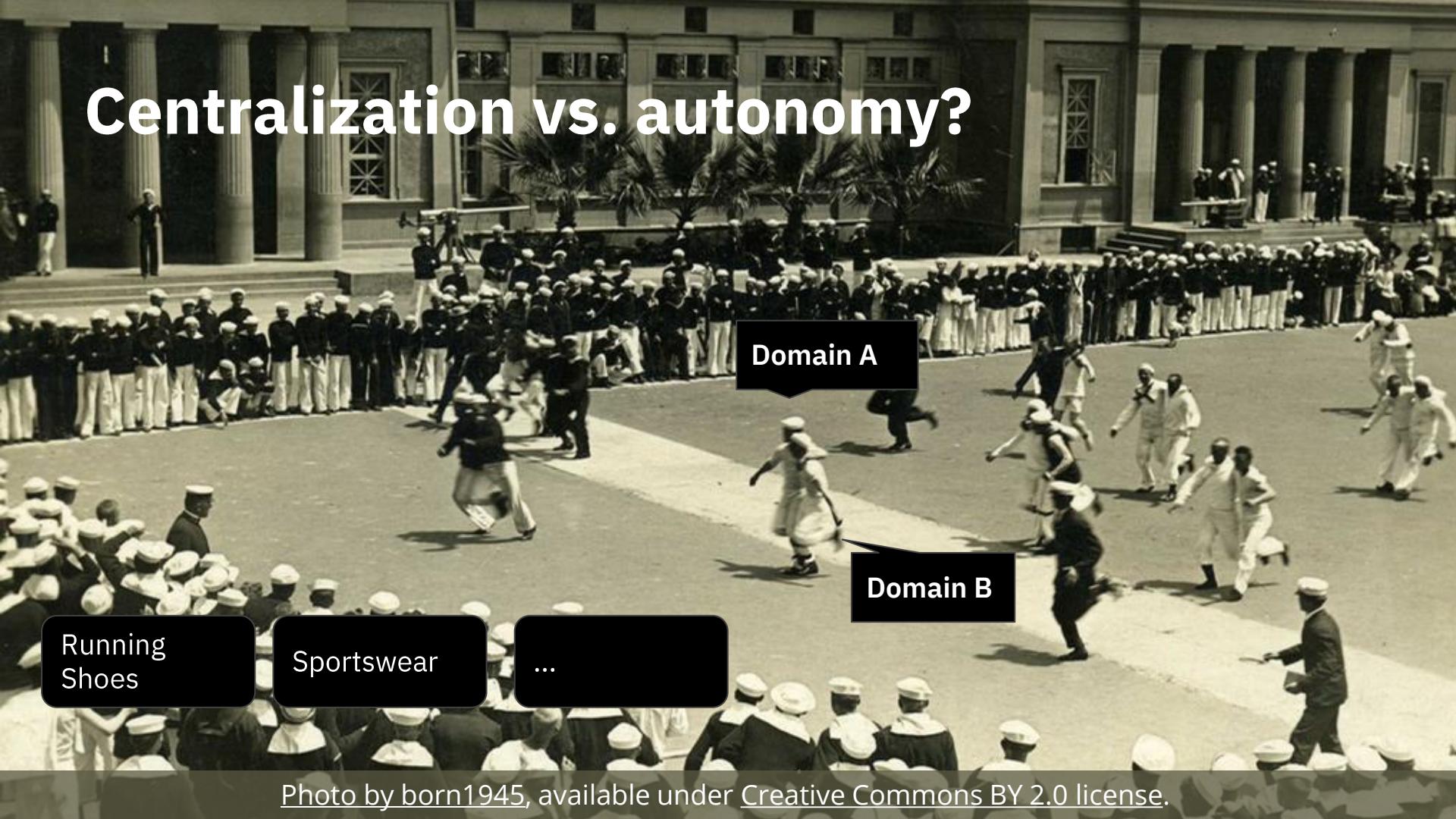
DURCH EINE GUT DURCHDACHTE TEAM TOPOLOGIE



Scaling adoption



Centralization vs. autonomy?



Running
Shoes

Sportswear

...

Domain A

Domain B

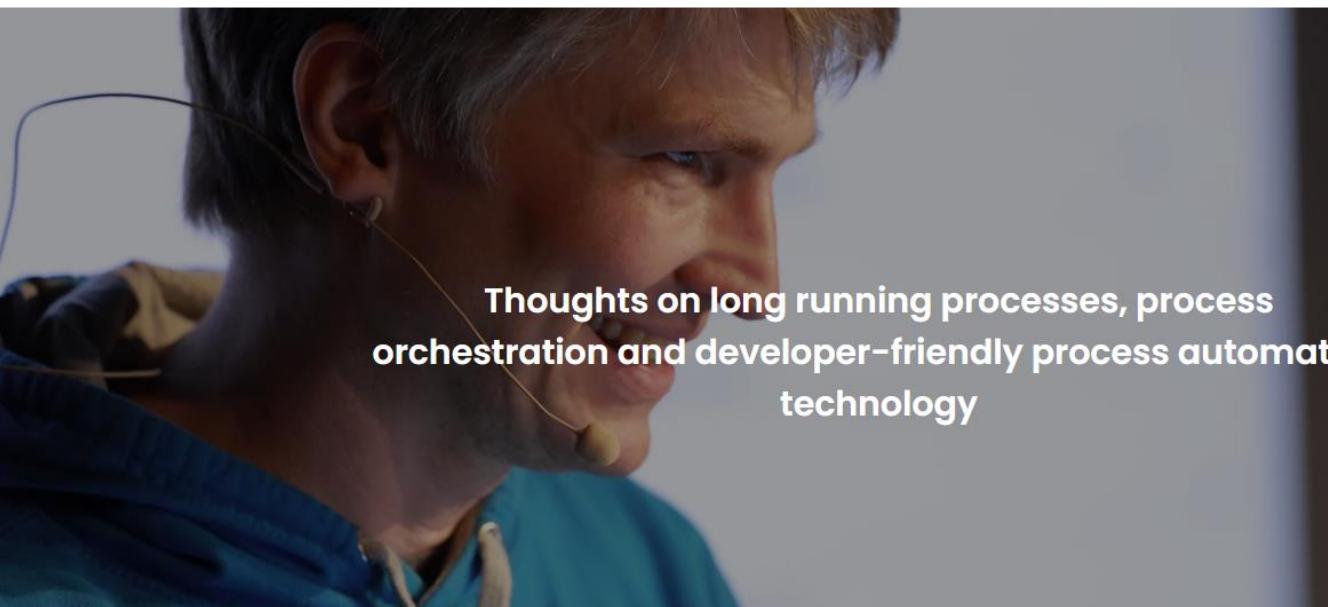
- # Process orchestration is an essential building brick for automation
- # Process automation = process orchestration + task automation
- # Red, yellow, and green processes have different requirements
- # Include more roles in solution creation
- # Low code and developer friendliness are not mutually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

Want To Know More?

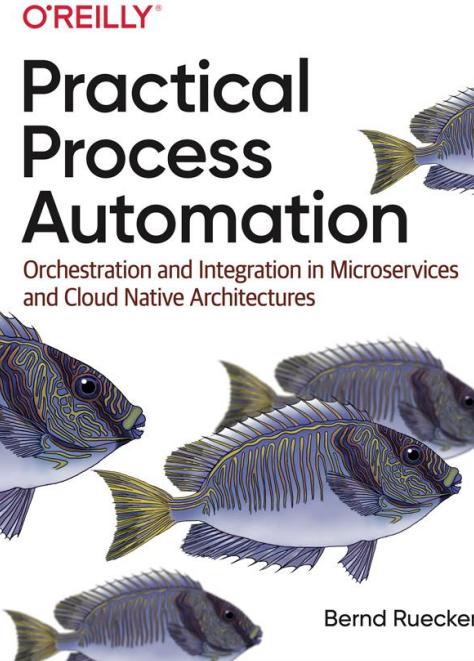
<https://berndruecker.io/>

Bernd Ruecker

Home About me My books My talks



Thoughts on long running processes, process
orchestration and developer-friendly process automation
technology





THE PROCESS ORCHESTRATION CONFERENCE

**199€* für
Teilnahme in Berlin**

(* Ticketpreis bis 15. März 2024)

**CAMUNDA
CON 2024**

<https://www.camundacon.com/>

Jetzt anmelden

BERLIN + ONLINE 15. & 16. Mai

Thank you!



Contact: bernd.ruecker@camunda.com
[@berndruecker](https://twitter.com/berndruecker)

Slides: <https://berndruecker.io>

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

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

