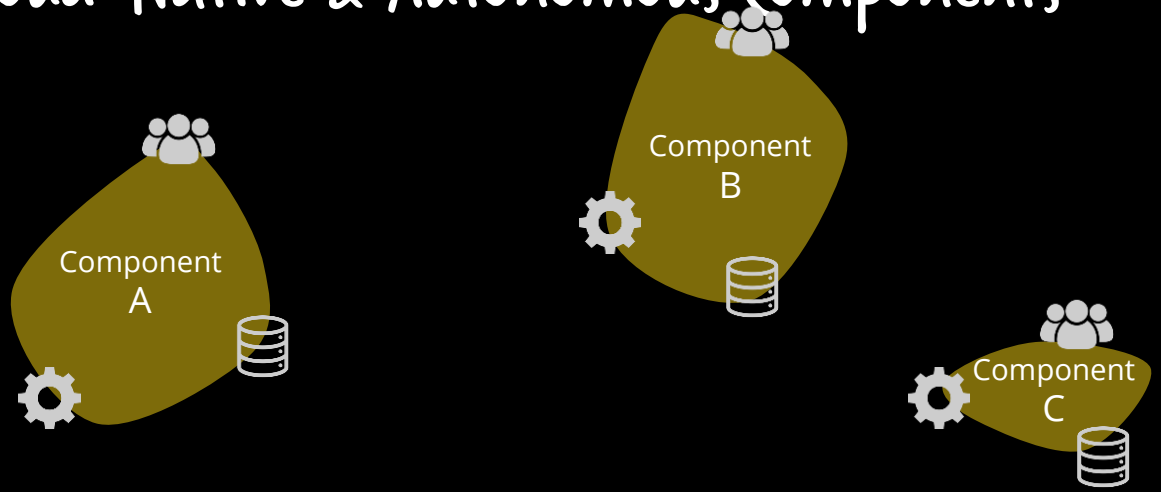
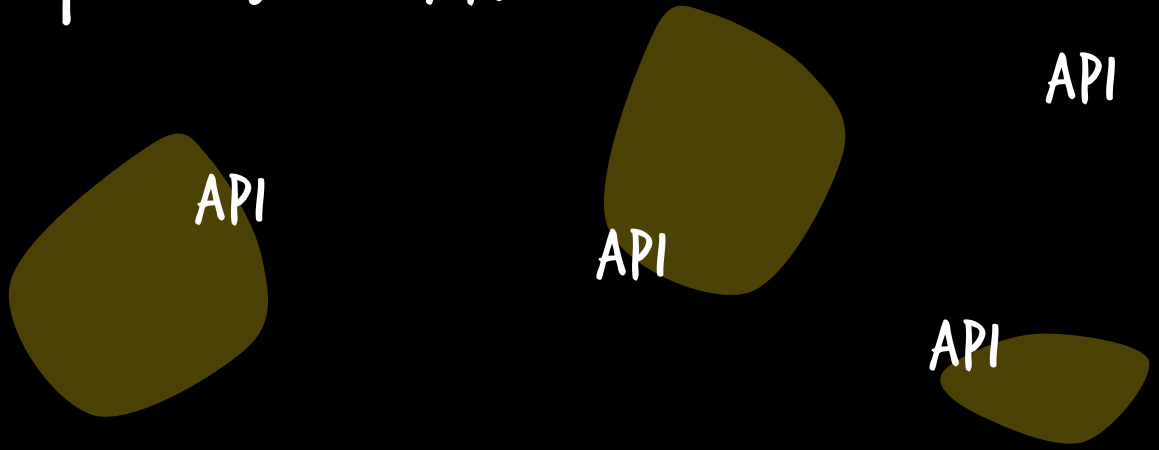
Automating Processes in Cloud-Native Architectures

@berndruecker

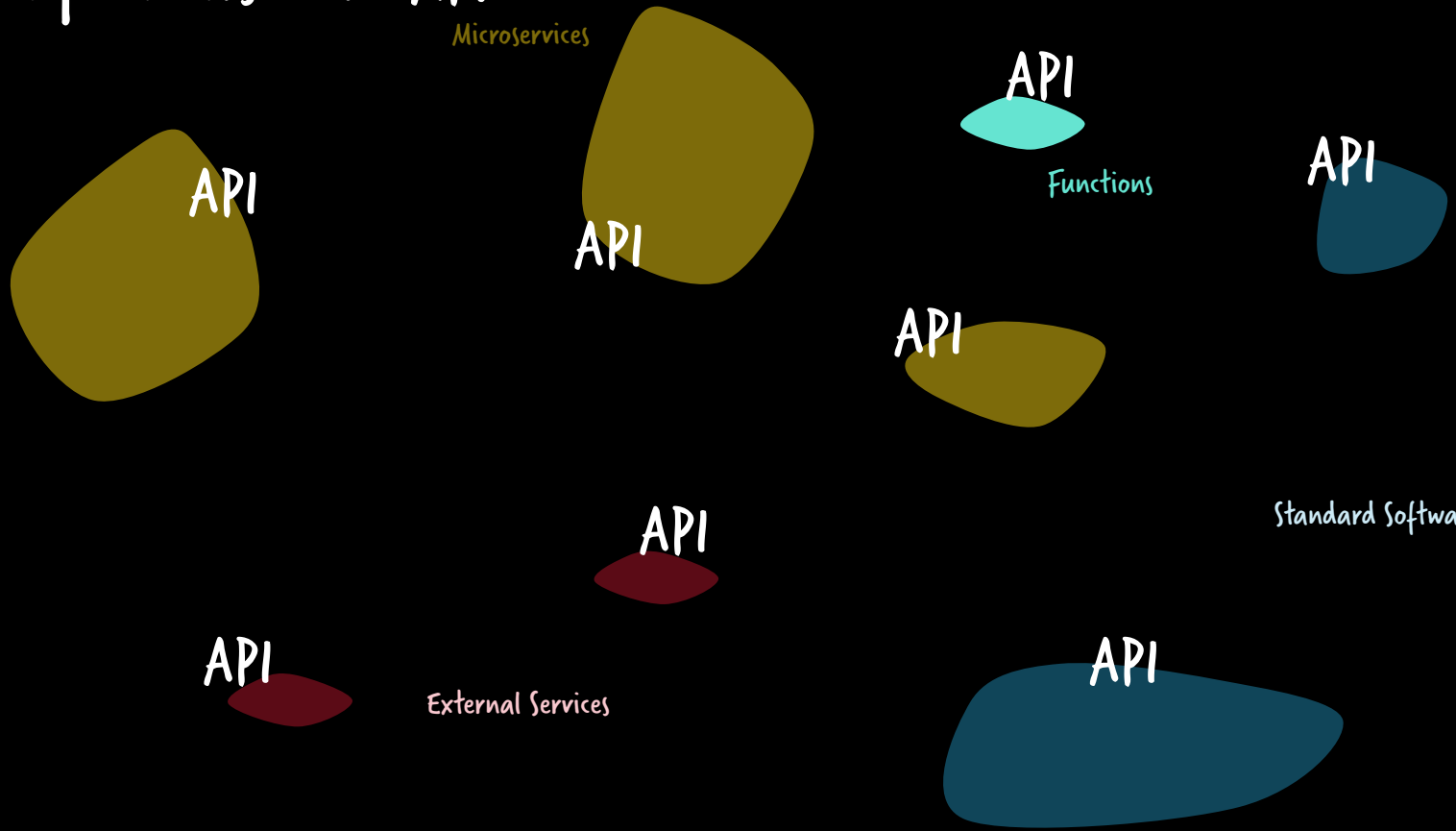
Cloud-Native & Autonomous Components



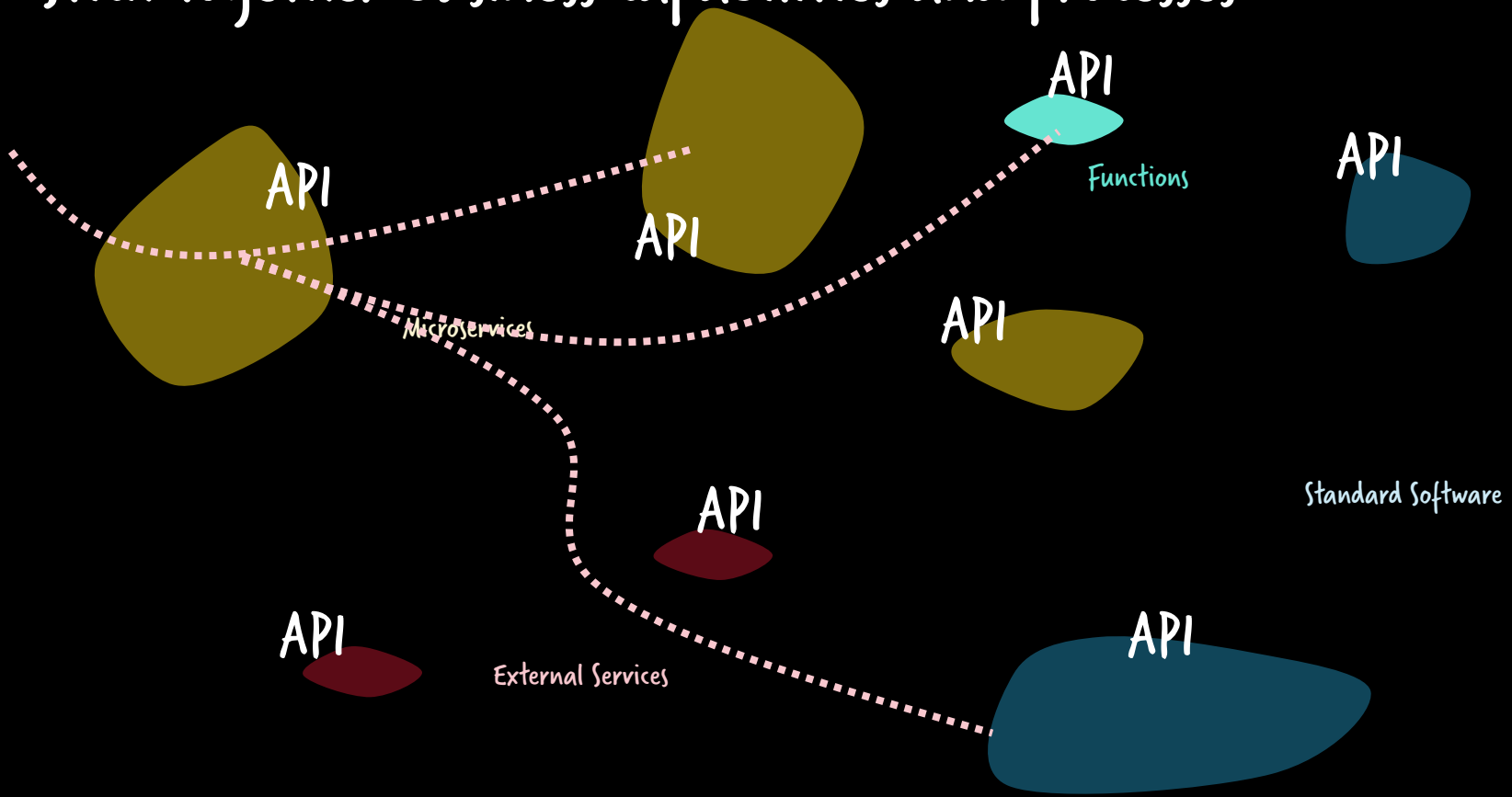
Components with API

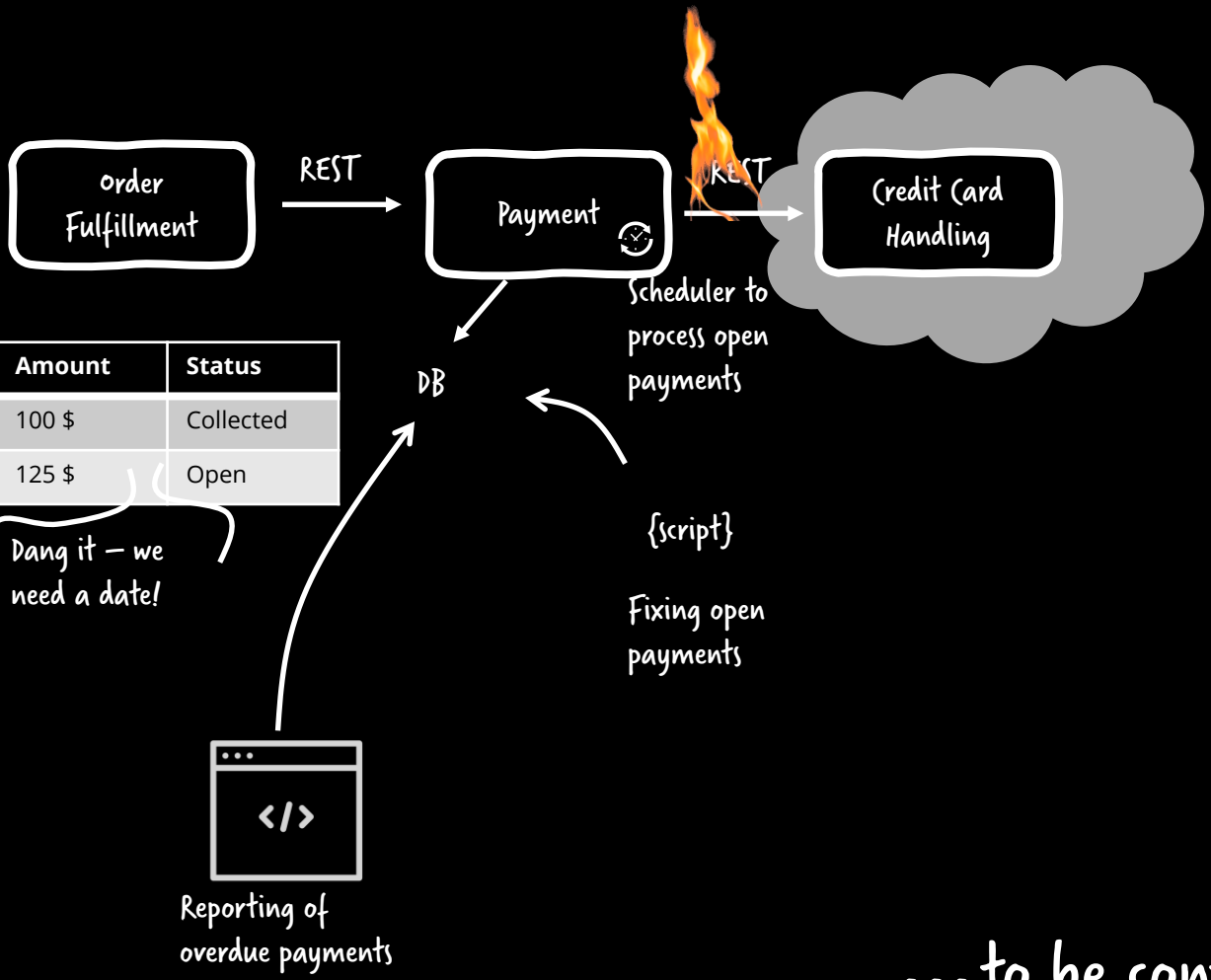


Components with API



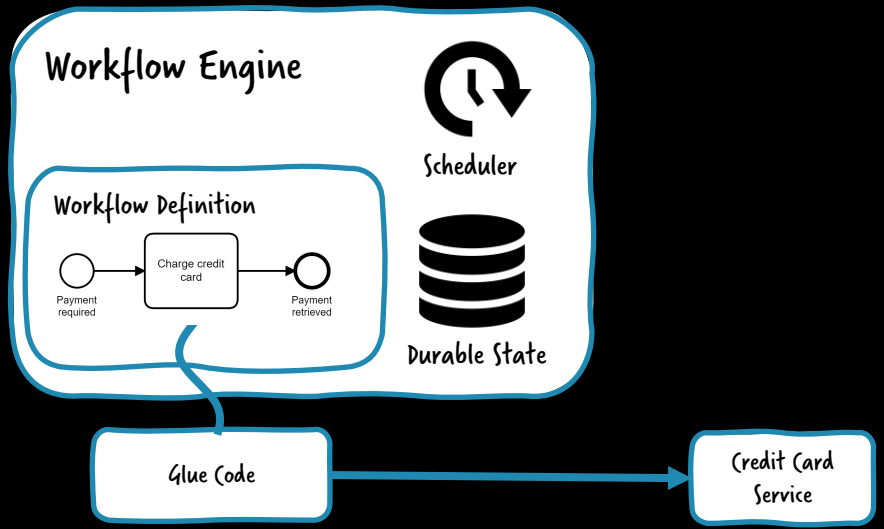
Stich together business capabilities and processes





...to be continued...

Using a workflow engine



Workflow Engine:

Is stateful

Can wait
Can retry
Can escalate
Can compensate

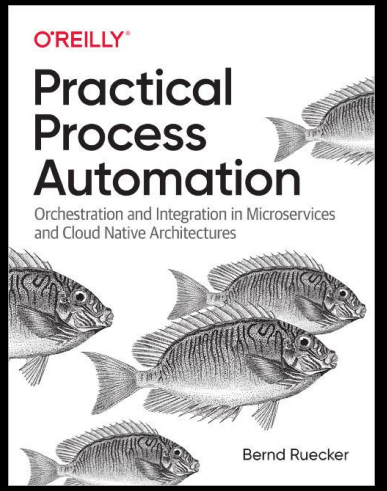
Provides visibility

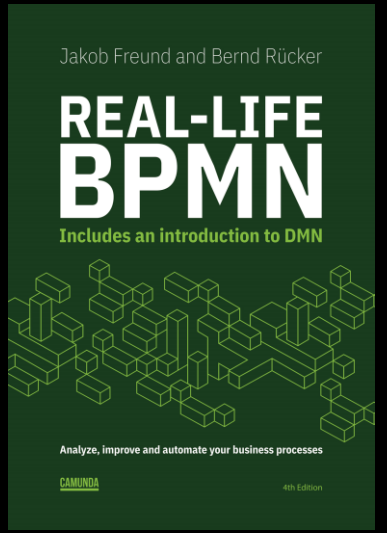
Warning:
Contains Opinion

Bernd Ruecker

Co-founder and
Chief Technologist of
Camunda

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

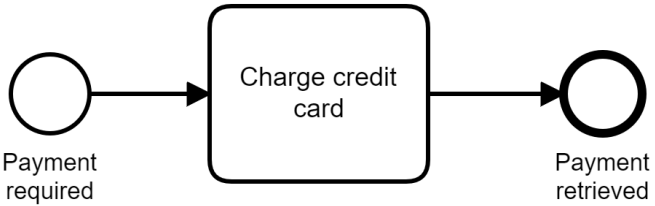




Camunda

- Source-Available & Developer Friendly
- Supports BPMN
- Mature tooling, widely adopted
- On-prem or cloud
- http://camunda.com/

```java
@RestController
public class PaymentRestHacksControllerV3 {

  @Autowired
  private ProcessEngine camunda;

  @PostConstruct
  public void createFlowDefinition() {
    BpmnModelInstance flow = Bpmn.createExecutableProcess("paymentV3") //
        .startEvent() //
        .serviceTask("stripe").camundaDelegateExpression("#{stripeAdapter}") //
          .camundaAsyncBefore().camundaFailedJobRetryTimeCycle("R3/PT1M") //
        .endEvent().done();

    camunda.getRepositoryService().createDeployment() //
      .addModelInstance("payment.bpmn", flow) //
      .deploy();
  }

  @Component("stripeAdapter")
  public static class StripeAdapter implements JavaDelegate {

    @Autowired
    private RestTemplate rest;
    private String stripeChargeUrl = "http://localhost:8099/charge";

    public void execute(DelegateExecution ctx) throws Exception {
      CreateChargeRequest request = new CreateChargeRequest();
      request.amount = (long) ctx.getVariable("amount");

      CreateChargeResponse response = new HystrixCommand<CreateChargeResponse>(HystrixCommandGroupKey.Factory.asKey("stripe")) {
        protected CreateChargeResponse run() throws Exception {
            return rest.postForObject( //
              stripeChargeUrl, //
              request, //
              CreateChargeResponse.class);
        }
      }.execute();
```



https://github.com/berndruecker/flowing-retail/blob/master/rest/java/payment-camunda/src/main/java/io/flowing/retail/payment/resthacks/PaymentRestHacksControllerV3.java

```java
@RequestMapping(path = "/payment/v3", method = PUT)
public String retrievePayment(String retrievePaymentPayload, HttpServletResponse response) throws Exception {
  String traceId = UUID.randomUUID().toString();
  String customerId = "0815"; // get somehow from retrievePaymentPayload
  long amount = 15; // get somehow from retrievePaymentPayload

  chargeCreditCard(customerId, amount);

  response.setStatus(HttpServletResponse.SC_ACCEPTED);
  return "{\"status\":\"pending\", \"traceId\": \"" + traceId + "\"}";
}

public void chargeCreditCard(String customerId, long remainingAmount) {
  ProcessInstance pi = camunda.getRuntimeService() //
      .startProcessInstanceByKey("paymentV3", //
          Variables.putValue("amount", remainingAmount));
}
```

Long Running Capabilities

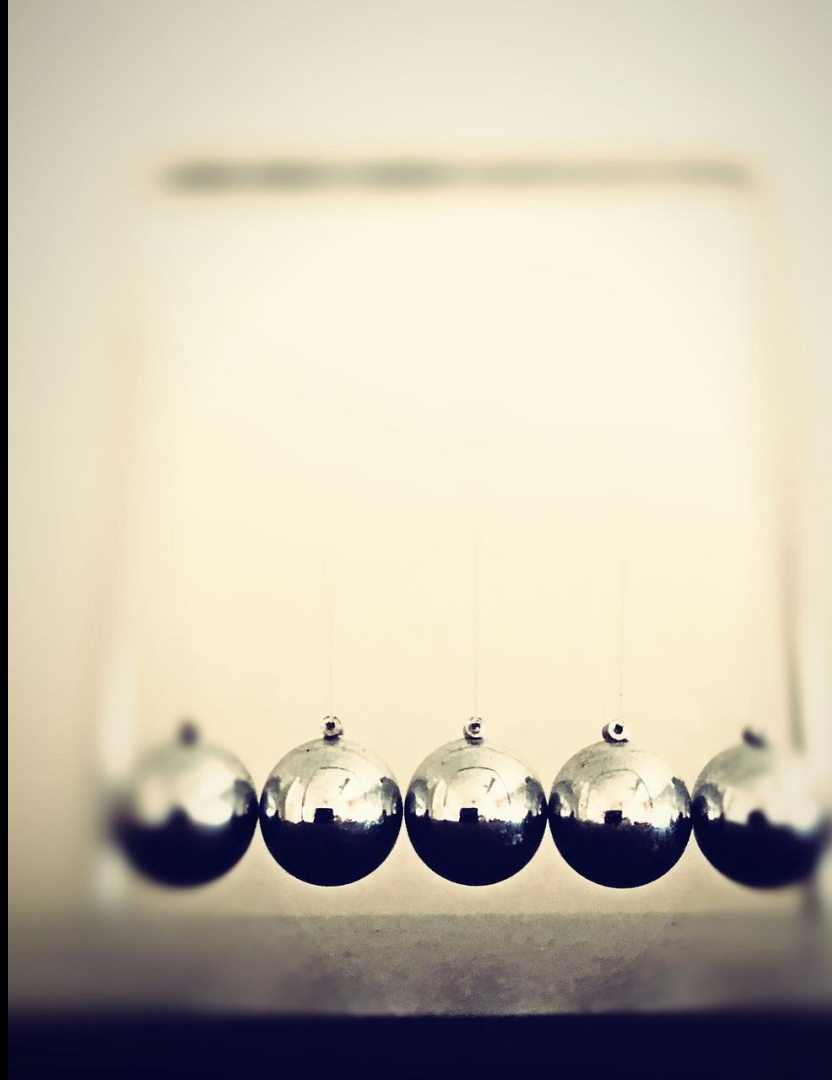
Process Automation

Solve Problems

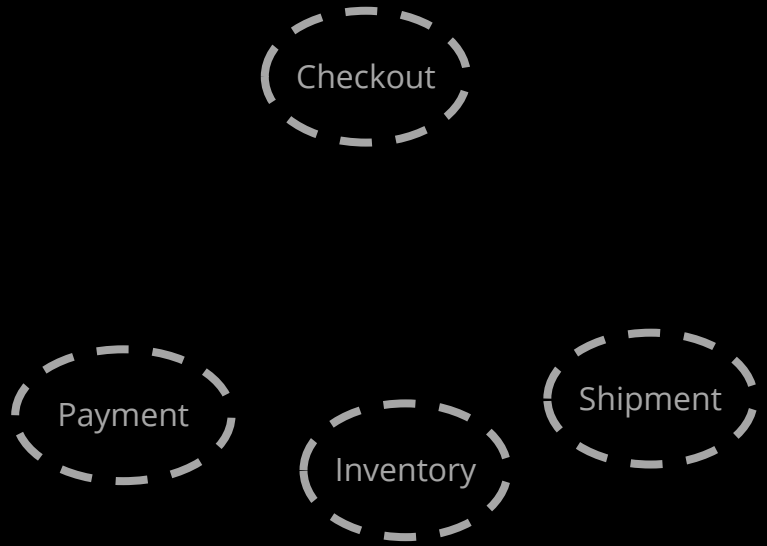
Long Running

Technical

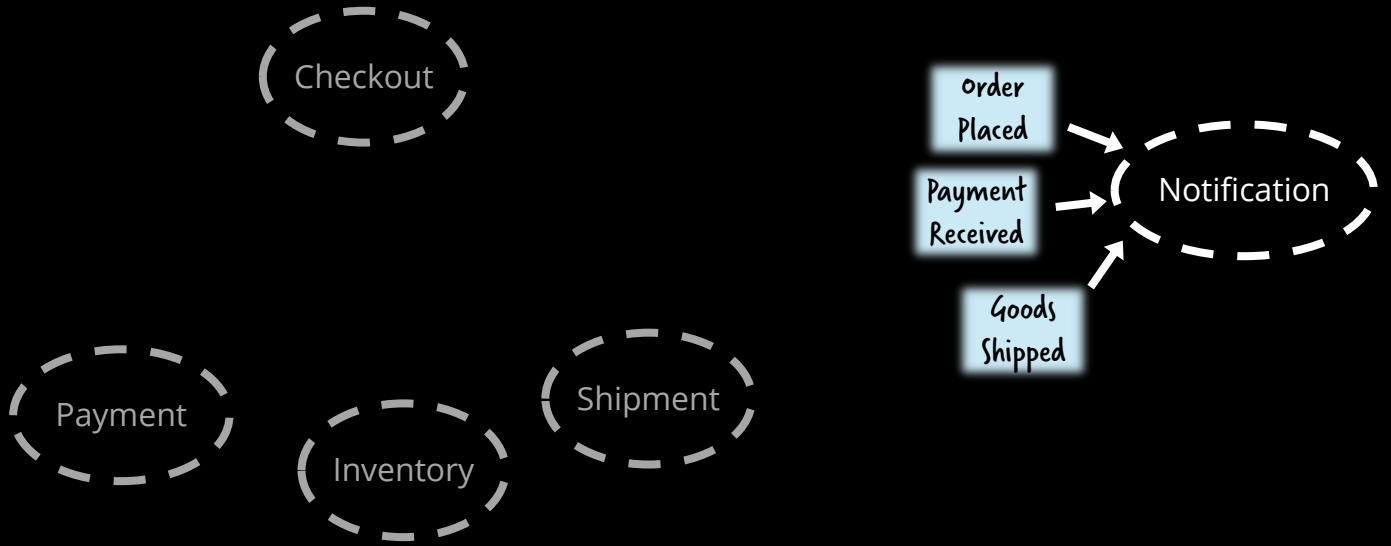
Going reactive!



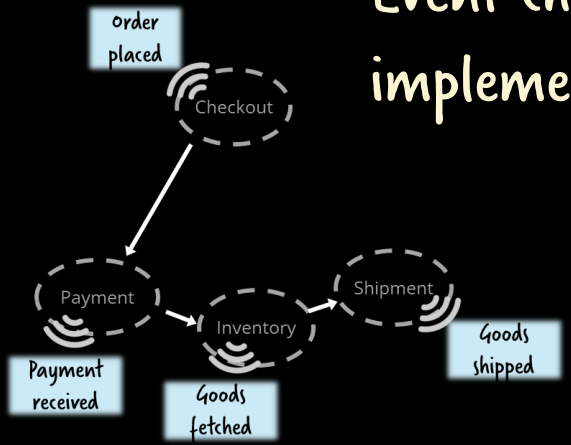
Example



Event-driven & Reactive



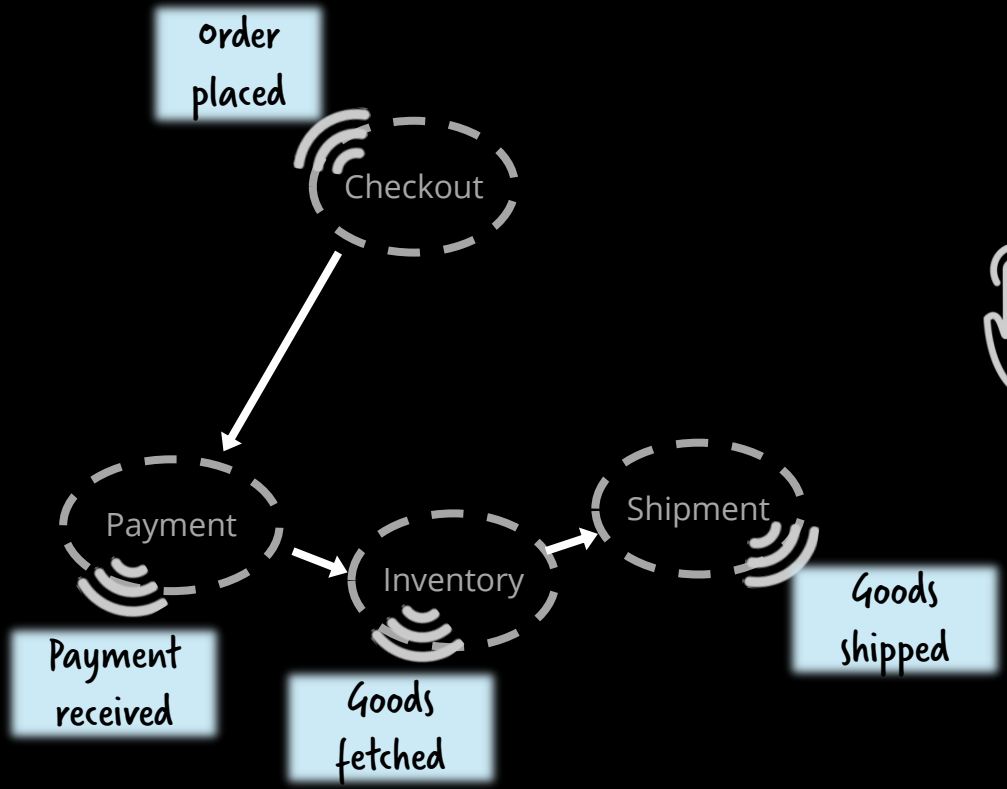
But there is also trouble in the event-driven land!



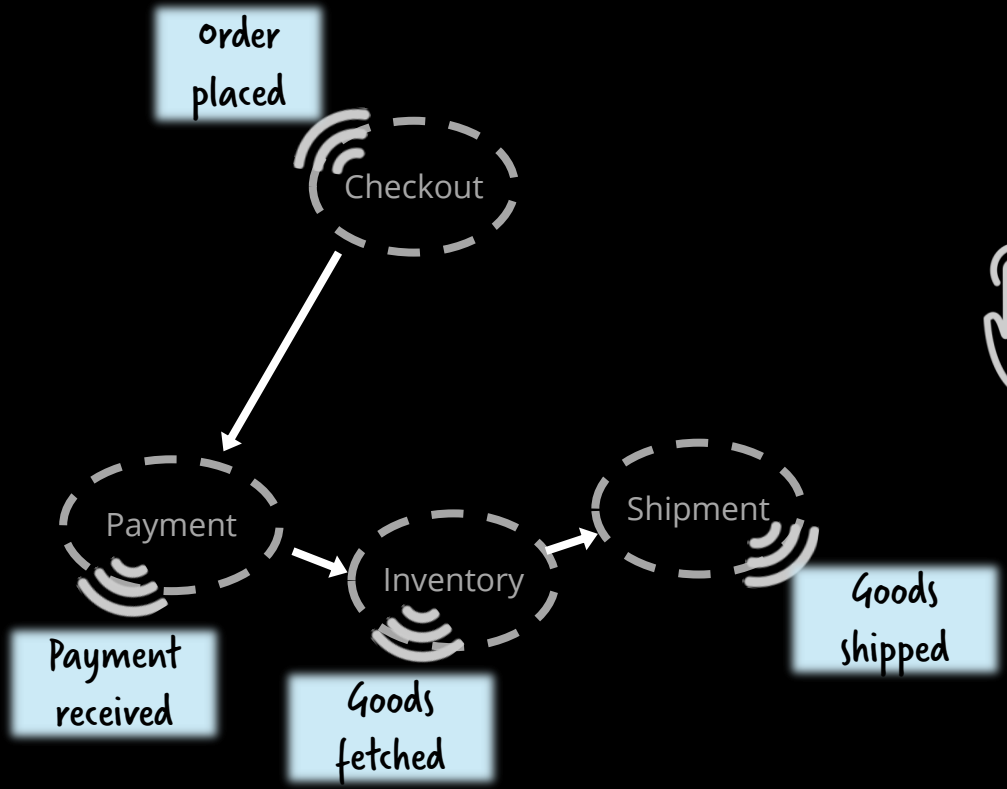
Event chains that implement processes

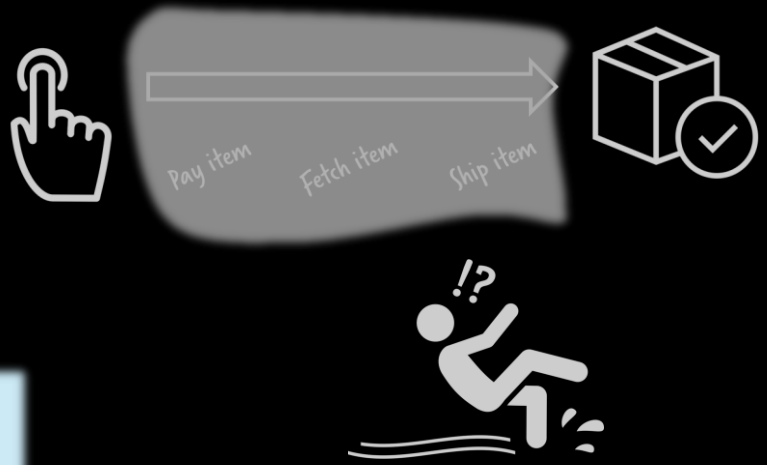
Implementing long running behavior

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.

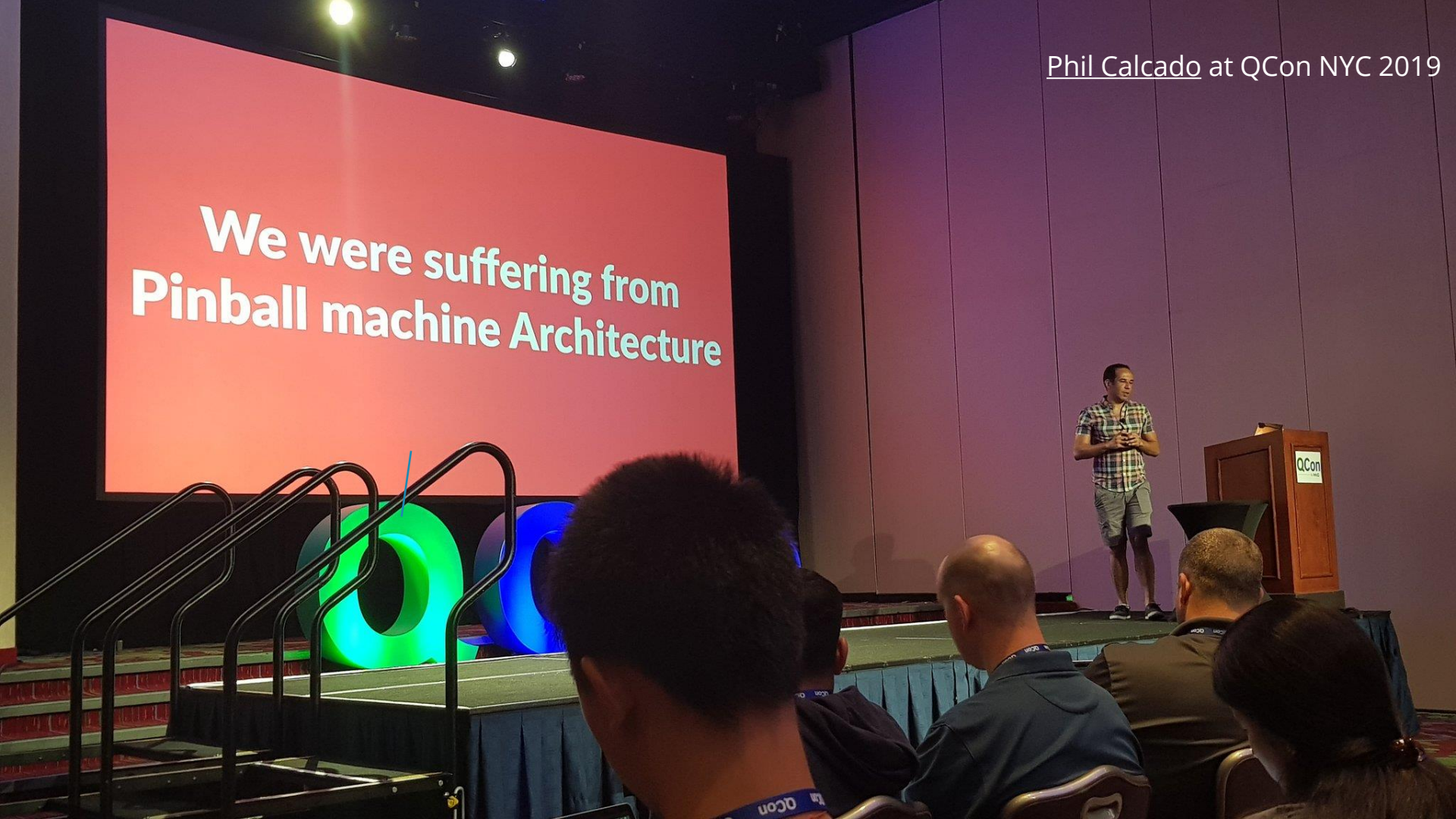
https://martinfowler.com/articles/201701-event-driven.html

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

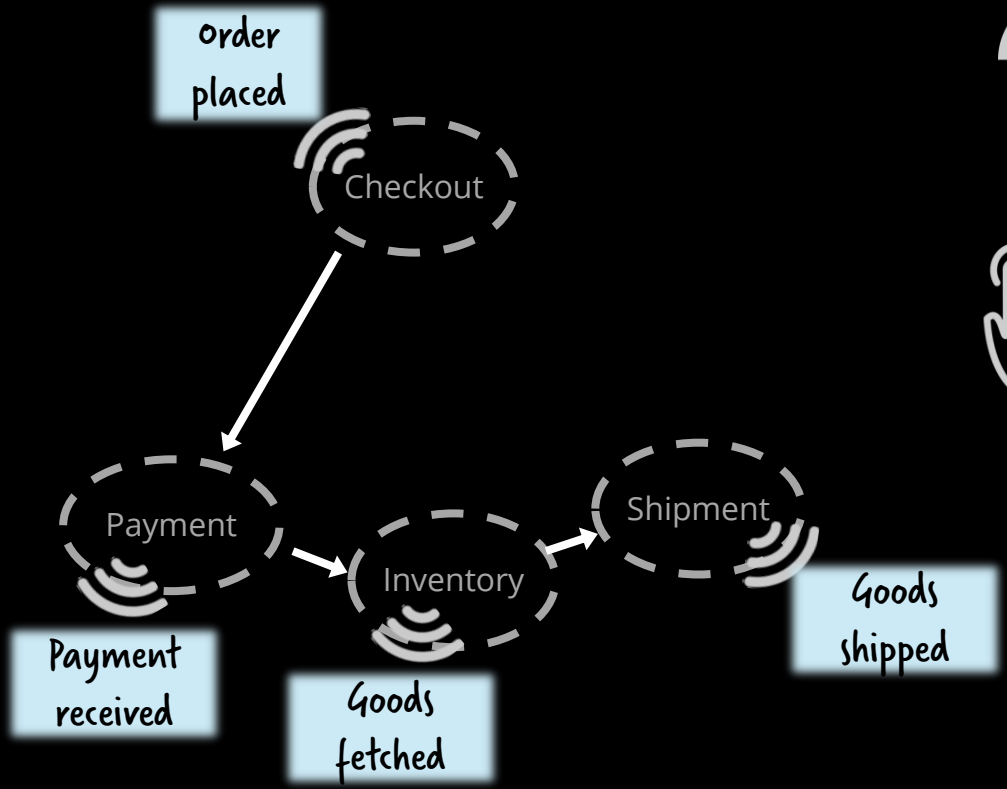


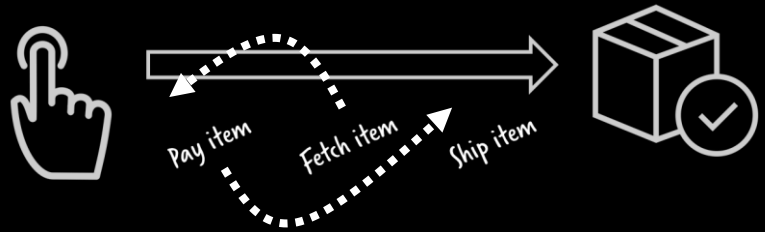
Phil Calcado at QCon NYC 2019

Pinball Machine Architecture

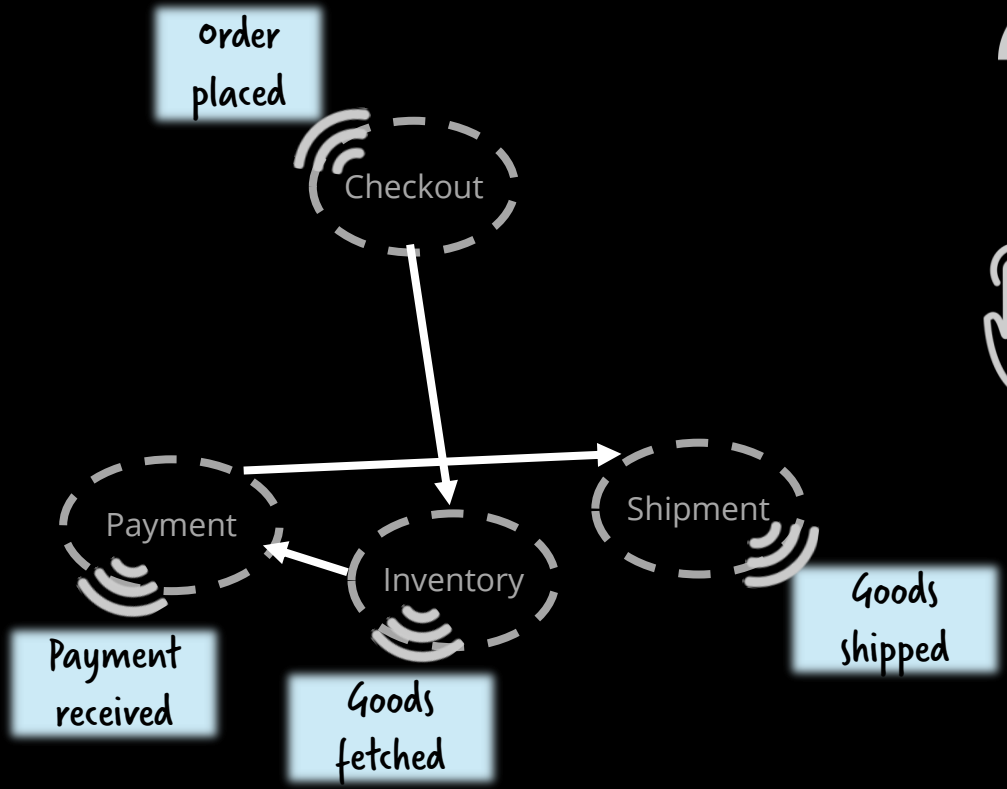
„What the hell just happened?“

Peer-to-peer event chains

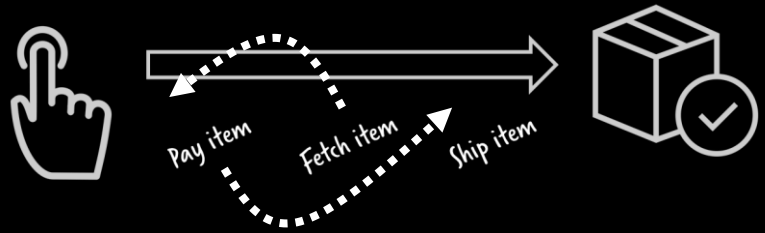




Peer-to-peer event chains



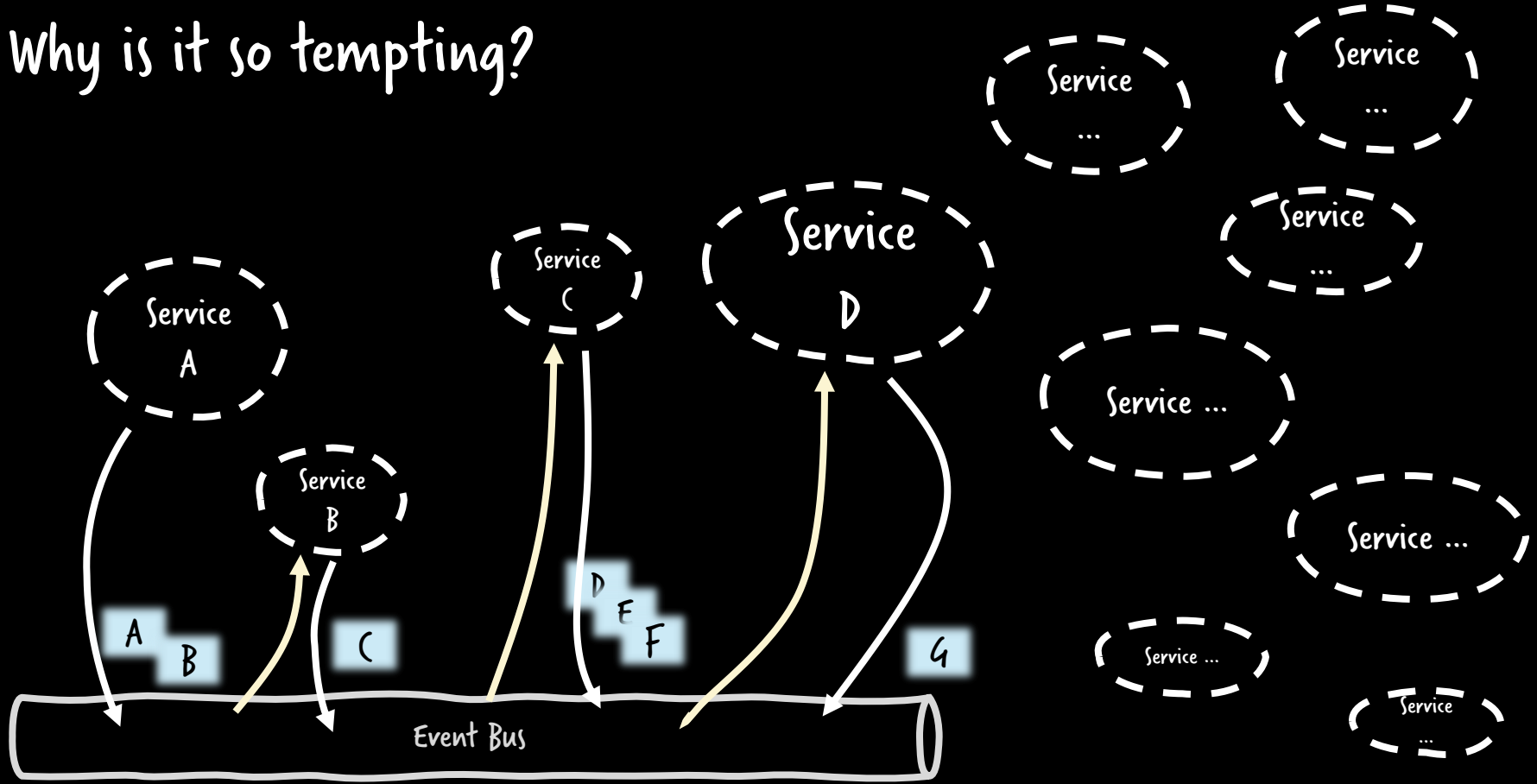
Fetch the goods **before** the payment



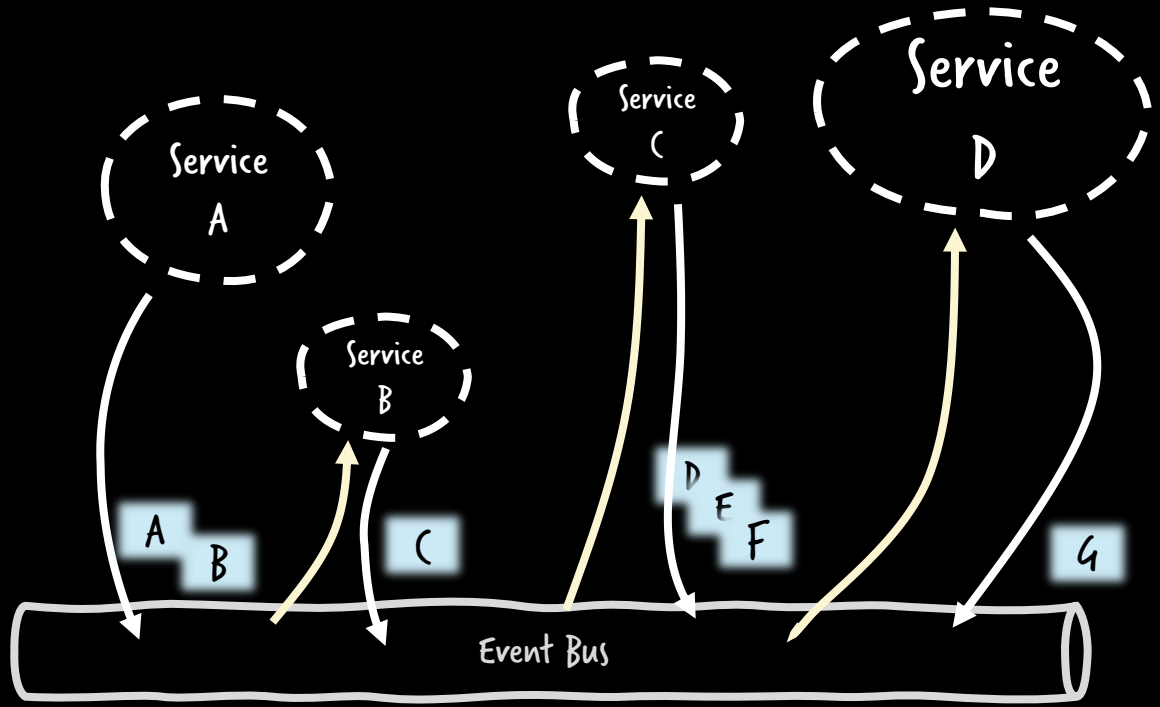
What we wanted

vs. what we got

Why is it so tempting?



Why is it so tempting?



Service
...

Service
...

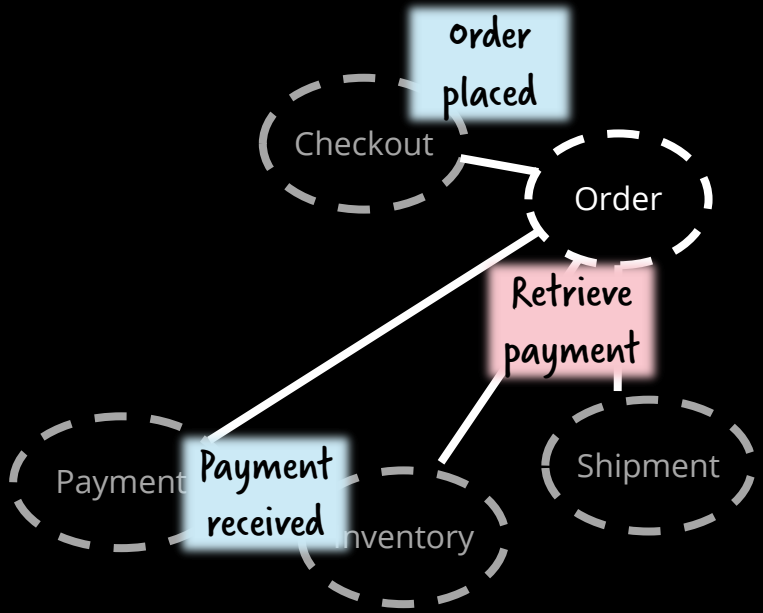
Adding is easy!

You can „buy" a shorter initial time-to-value by choreography.

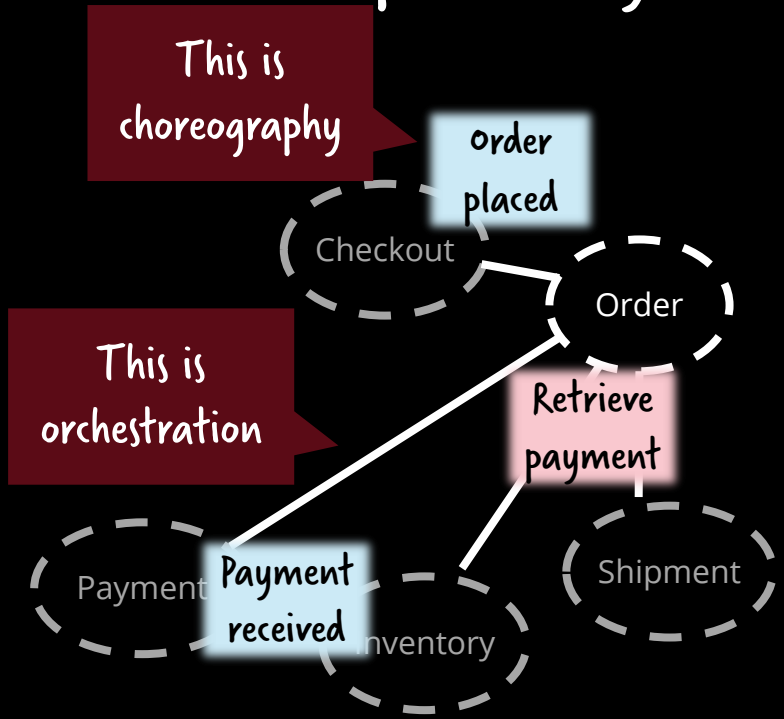
It yields in technical debt.

Service
...

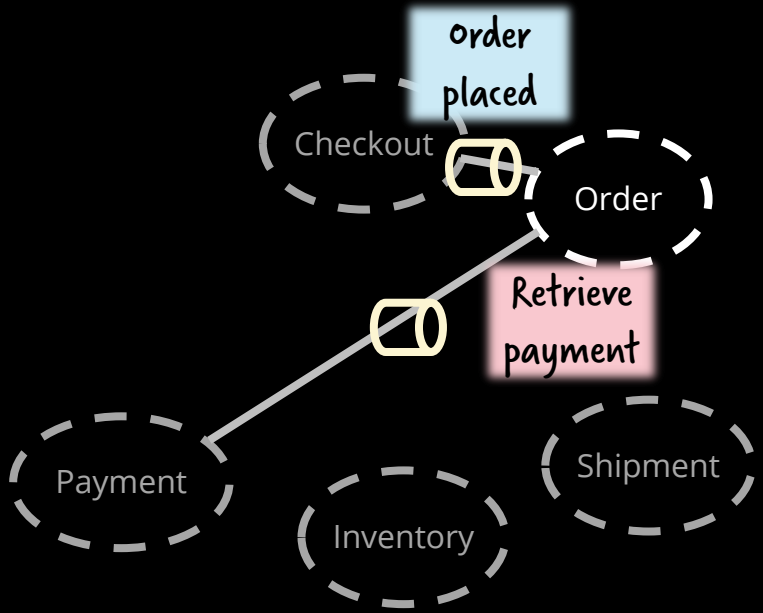
Decide about responsibility



Decide about responsibility

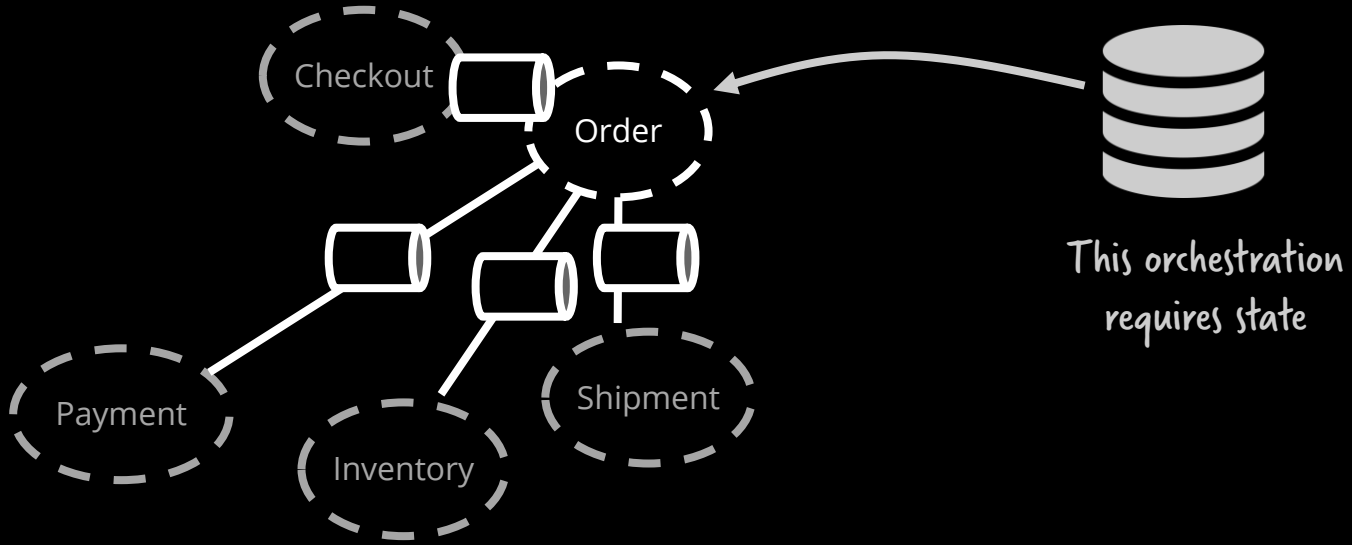


Decide about responsibility

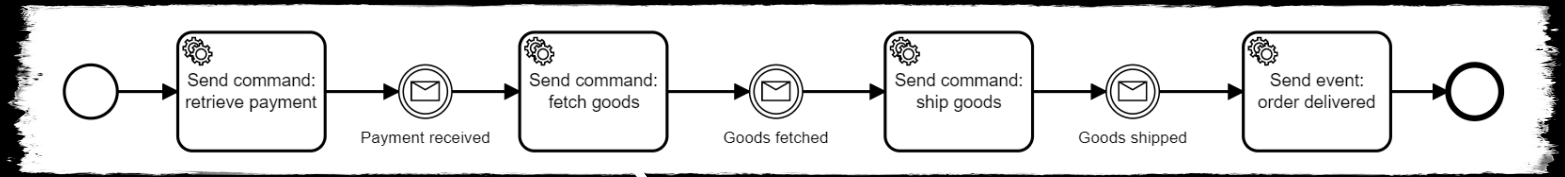


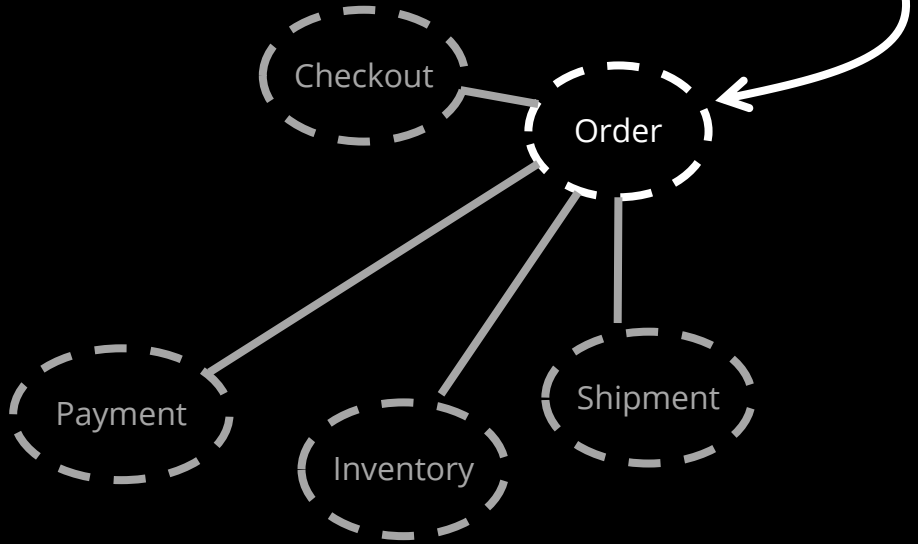
It can still be messaging!

Stateful orchestration

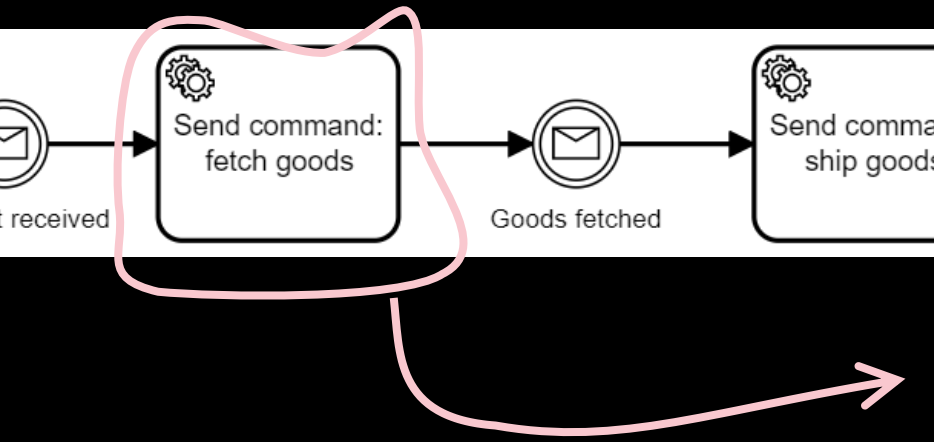


Long Running Capabilities


Send command: retrieve payment
Payment received
Send command: fetch goods
Goods fetched
Send command: ship goods
Goods shipped
Send event: order delivered


Checkout
Order
Payment
Inventory
Shipment

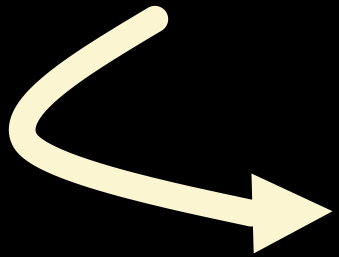
Glue code (e.g. Java)

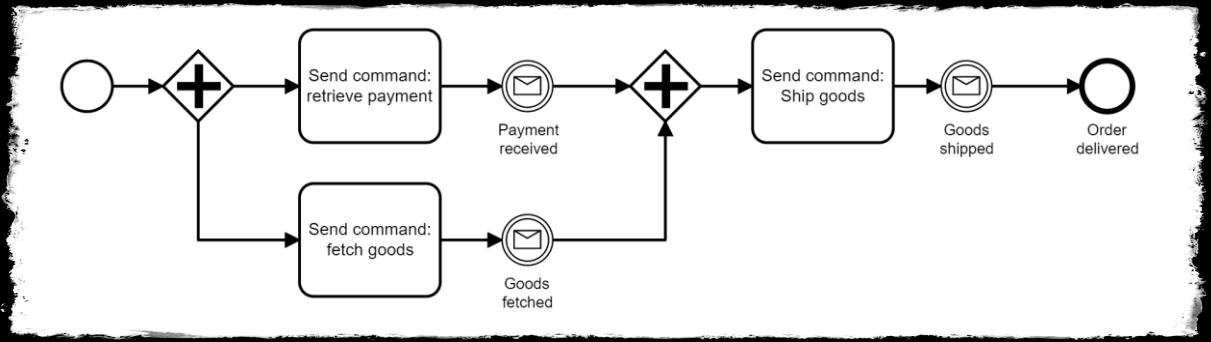


https://github.com/berndruecker/flowing-retail/blob/master/kafka/java/order-zeebe/src/main/java/io/flowing/retail/kafka/order/flow/FetchGoodsAdapter.java

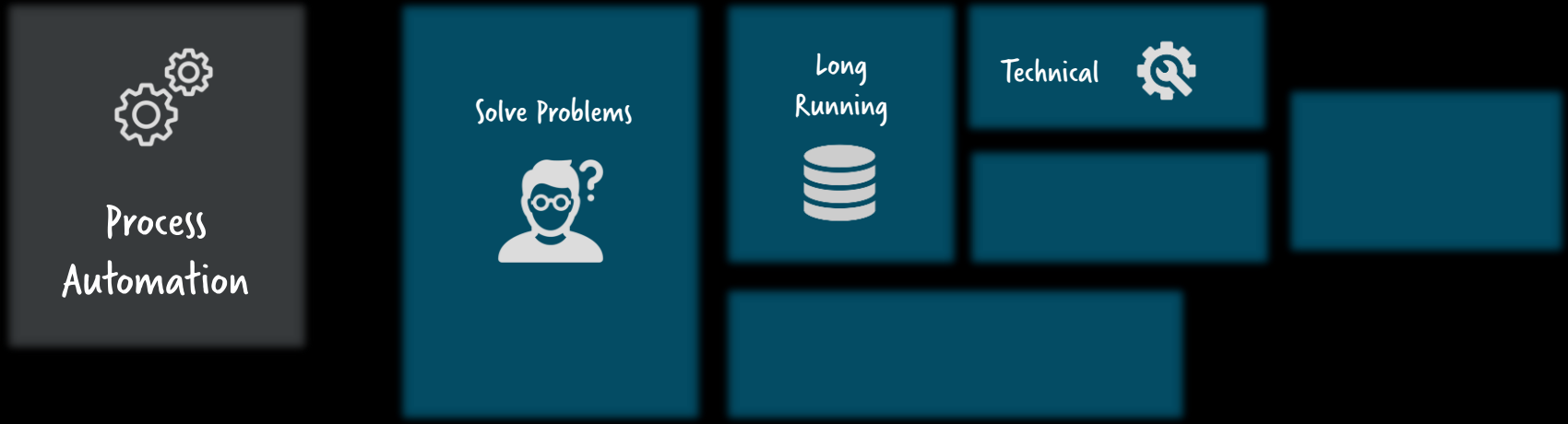
```java
@Component
public class FetchGoodsAdapter {

  @Autowired
  private MessageSender messageSender;

  @Autowired
  private OrderRepository orderRepository;

  @ZeebeWorker(type = "fetch-goods")
  public void handle(JobClient client, ActivatedJob job) {
    OrderFlowContext context = OrderFlowContext.fromMap(job.getVariablesAsMap());
    Order order = orderRepository.findById( context.getOrderId() ).get();

    // generate an UUID for this communication
    String correlationId = UUID.randomUUID().toString();

    messageSender.send(new Message<FetchGoodsCommandPayload>( //
            "FetchGoodsCommand", //
            context.getTraceId(), //
            new FetchGoodsCommandPayload() //
              .setRefId(order.getId()) //
              .setItems(order.getItems())) //
        .setCorrelationid(correlationId));

    client.newCompleteCommand(job.getKey()) //
      .variables(Collections.singletonMap("CorrelationId_FetchGoods", correlationId))
      .send().join();
  }

}
```

Now it is easy to change the process flow



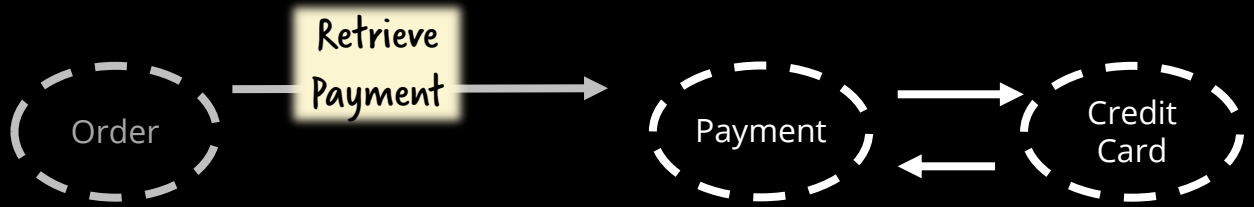


Long Running Capabilities

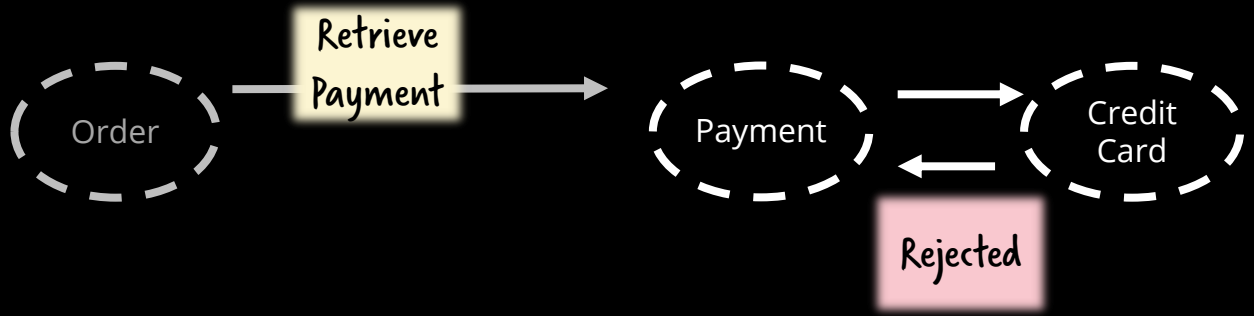


Long Running Capabilities

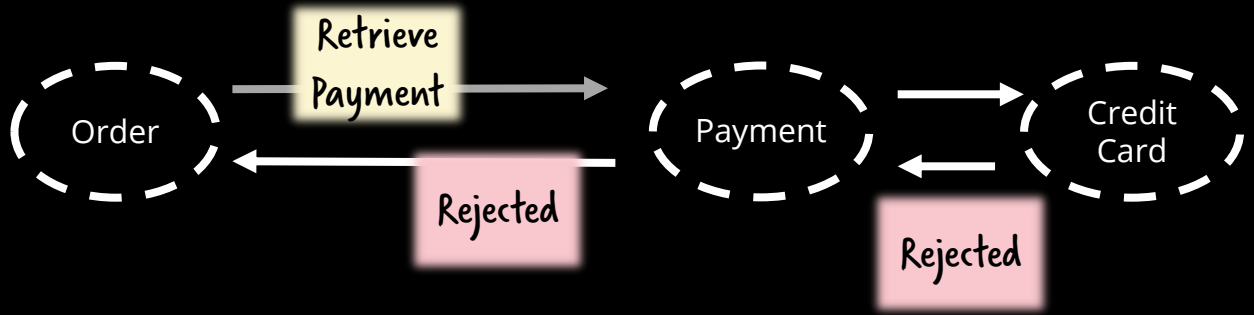
Example



Example

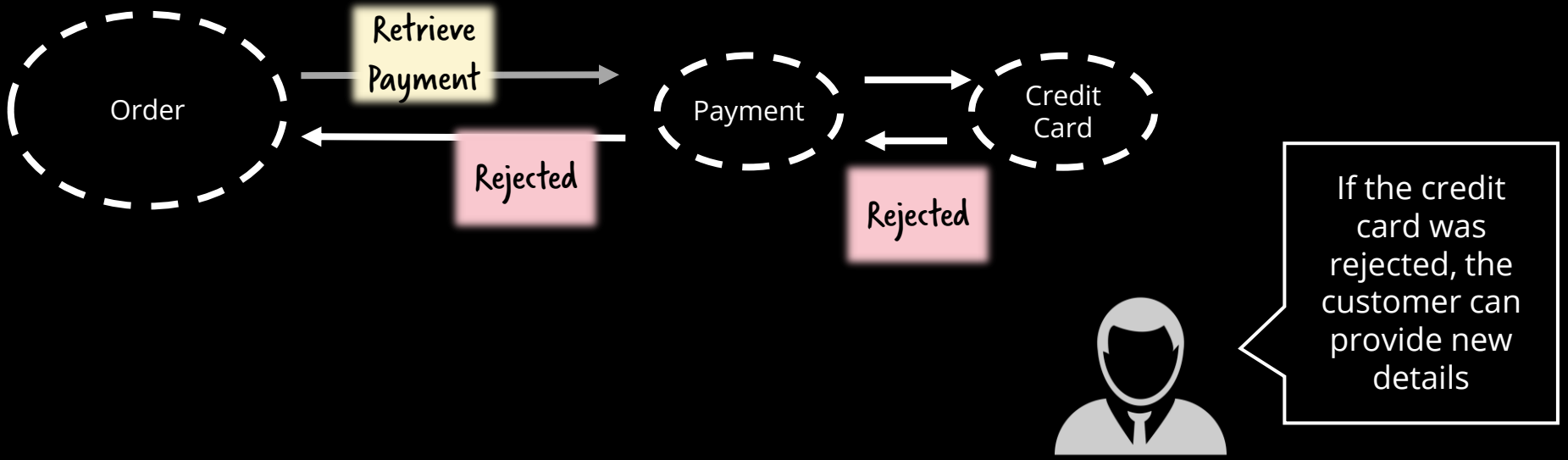


Example

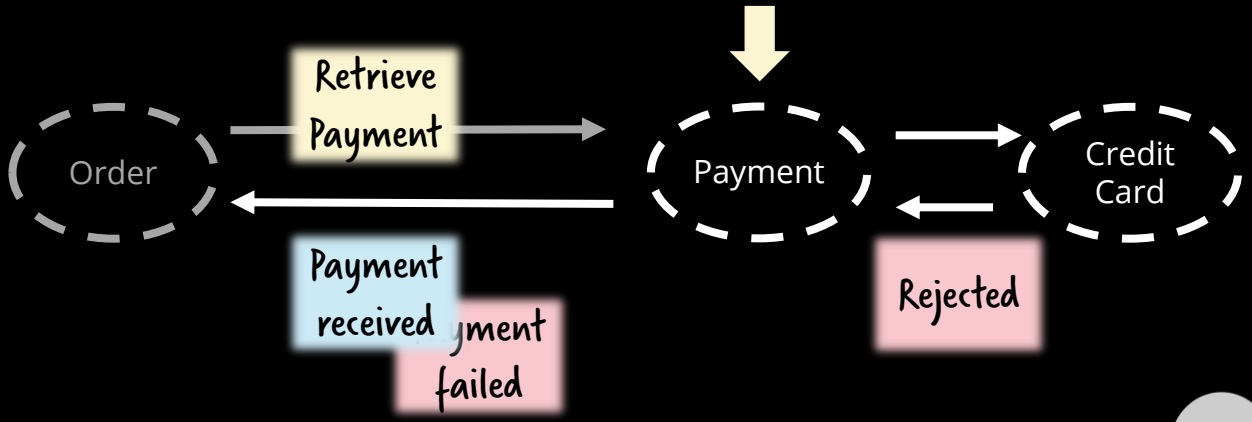


If the credit card was rejected, the customer can provide new details

Example

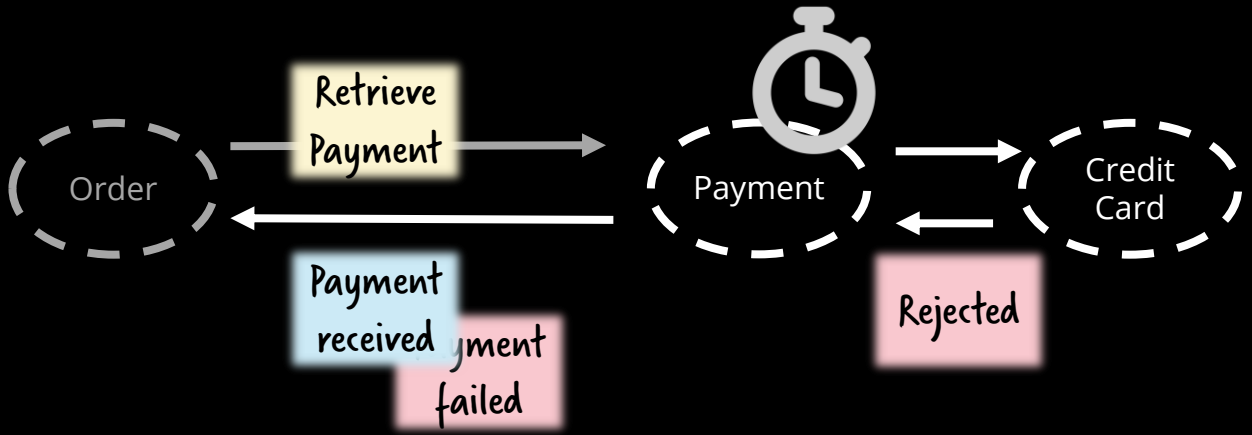

Order
Retrieve Payment
Rejected
Payment
Credit Card
Rejected
If the credit card was rejected, the customer can provide new details

Who is responsible?

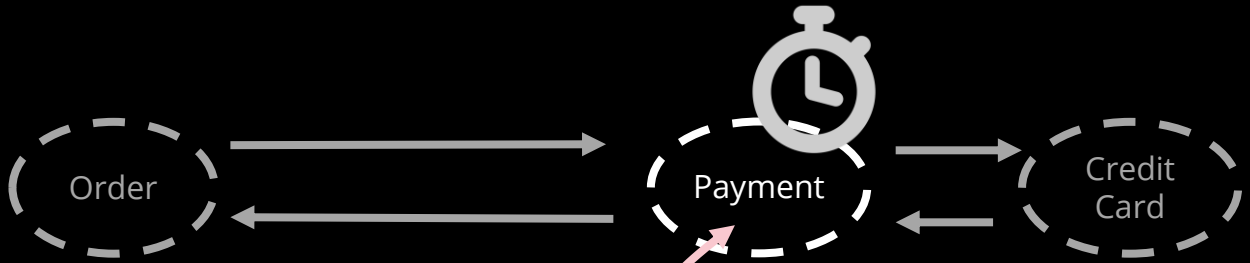


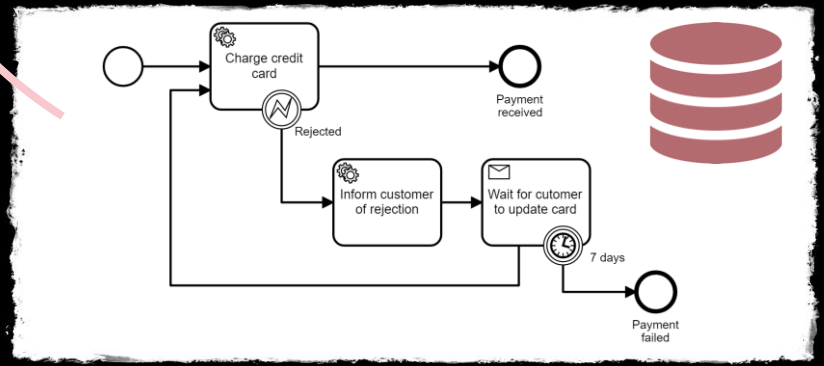
If the credit card was rejected, the customer can provide new details

Long running services

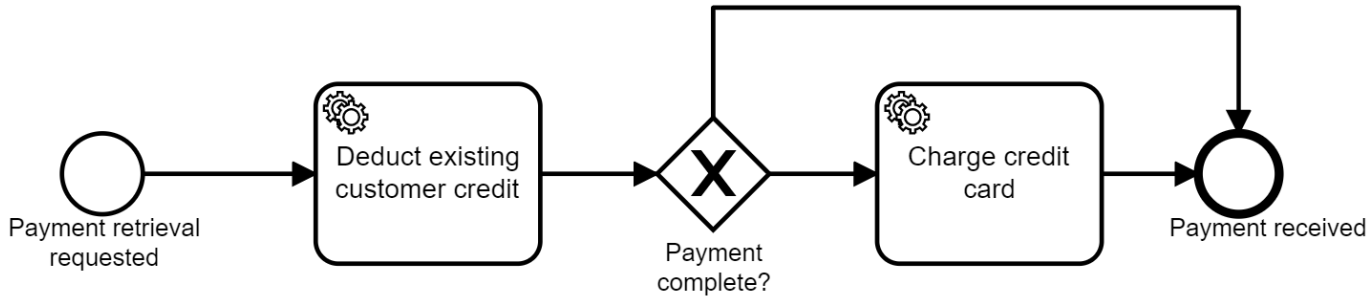


Long running services

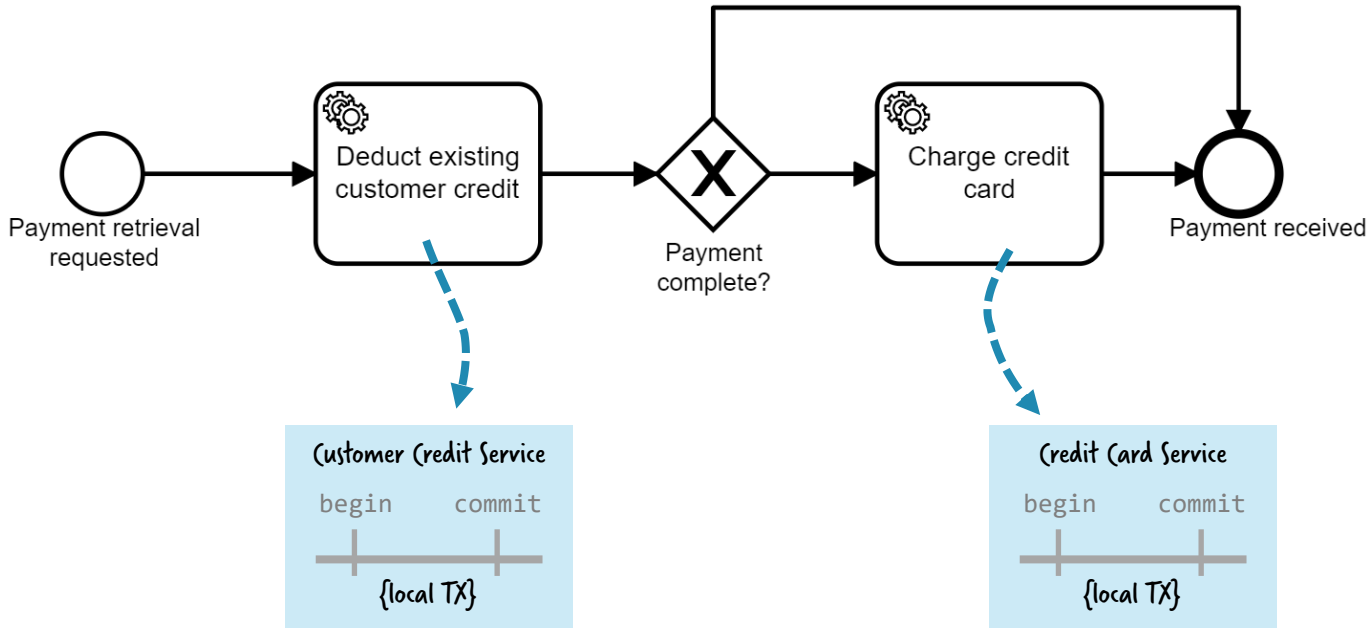




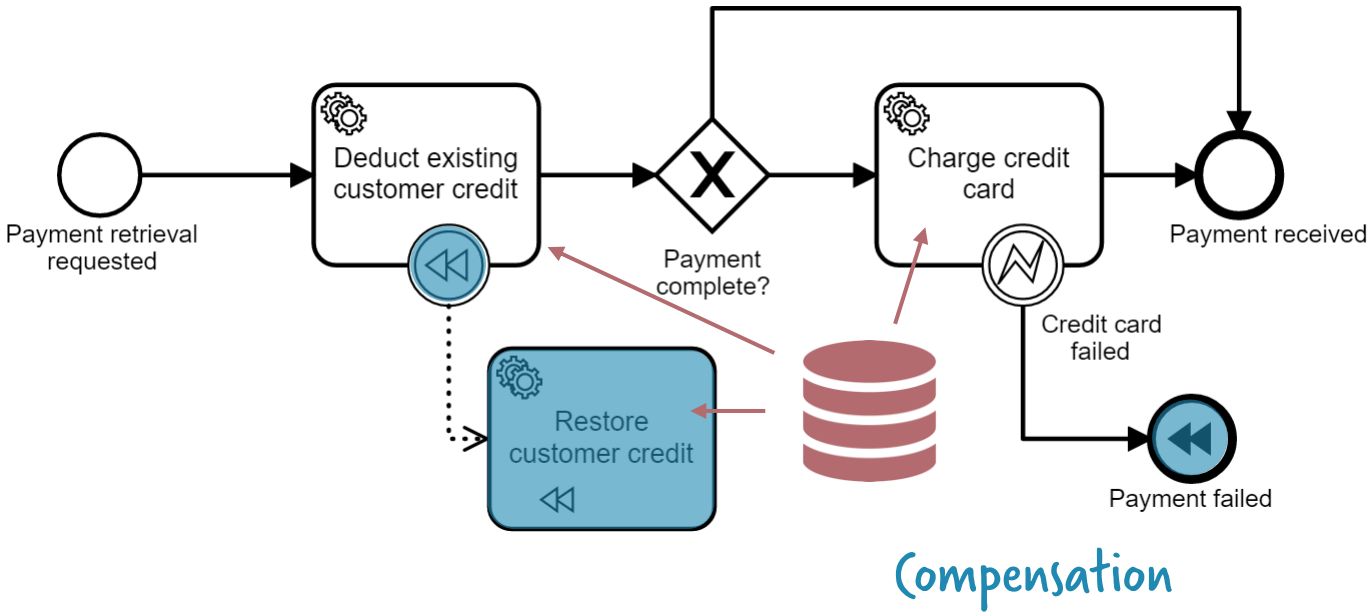
Extending Payment options



Extending Payment options



Distributed transactions using compensation *



*aka Saga pattern

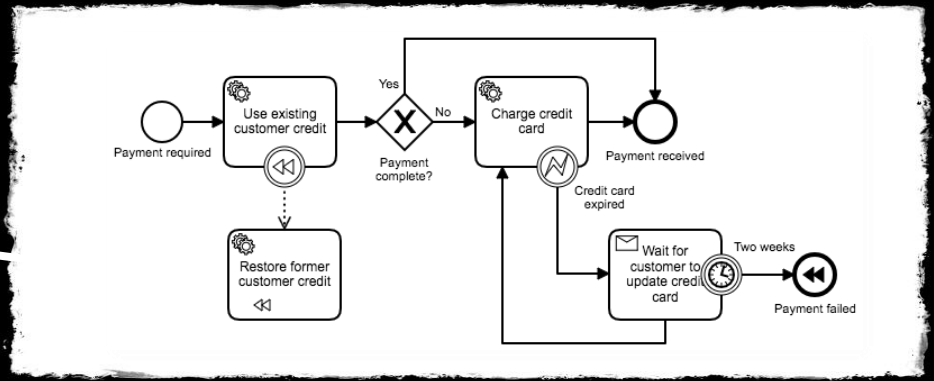
Long Running Capabilities

The workflow is domain logic as part of the service

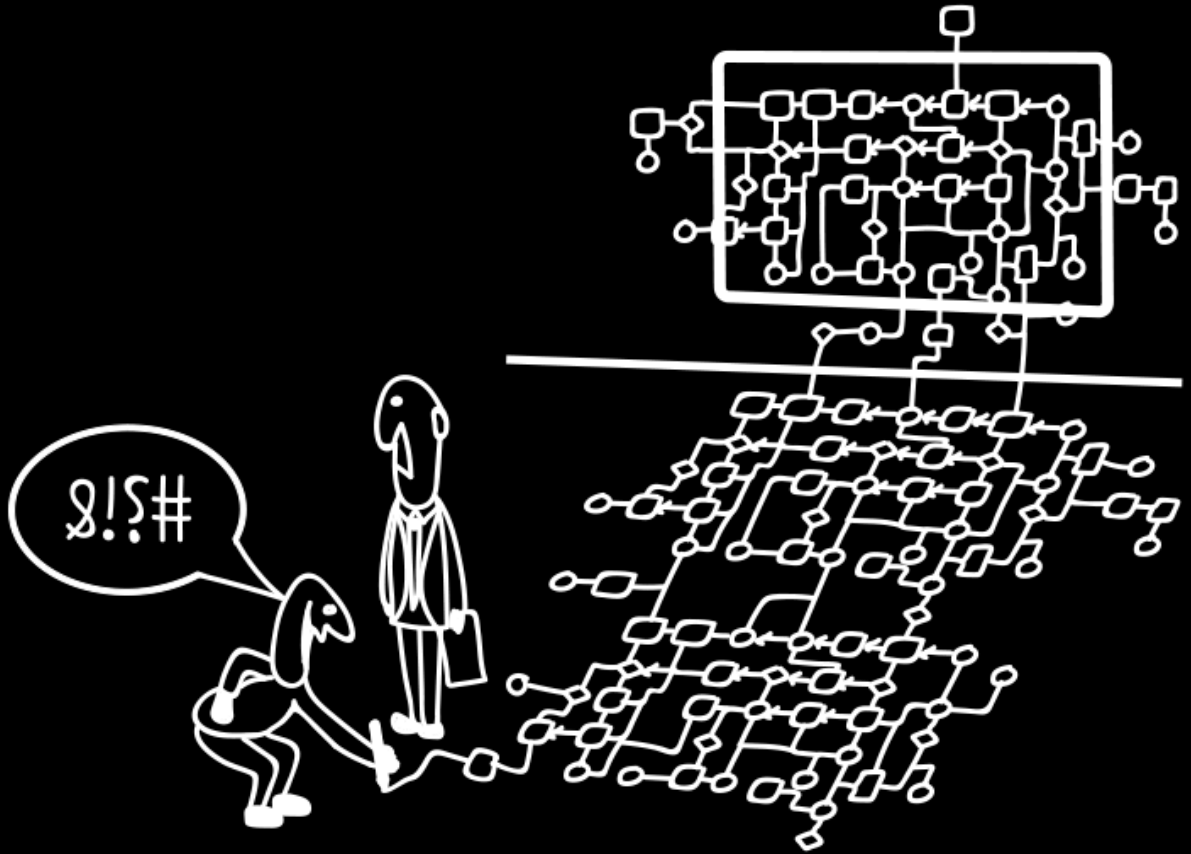
Checkout

Order

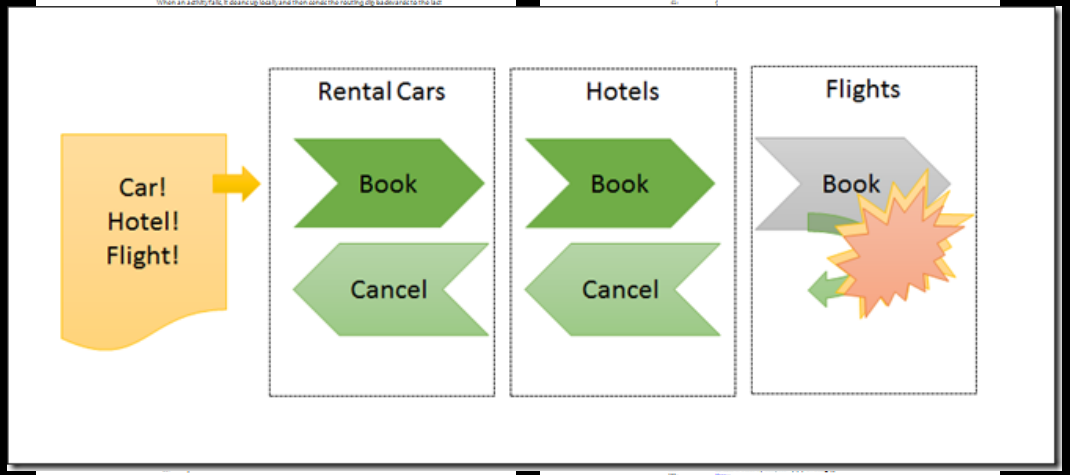
Payment

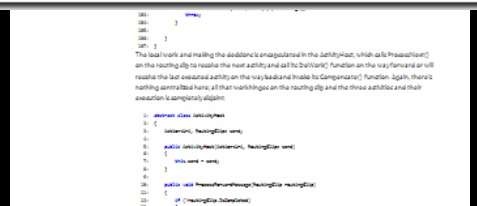


Graphical models?



Clemens Vasters
Architect at Microsoft





http://vasters.com/archive/Sagas.html

Clemens Vasters
Architect at Microsoft

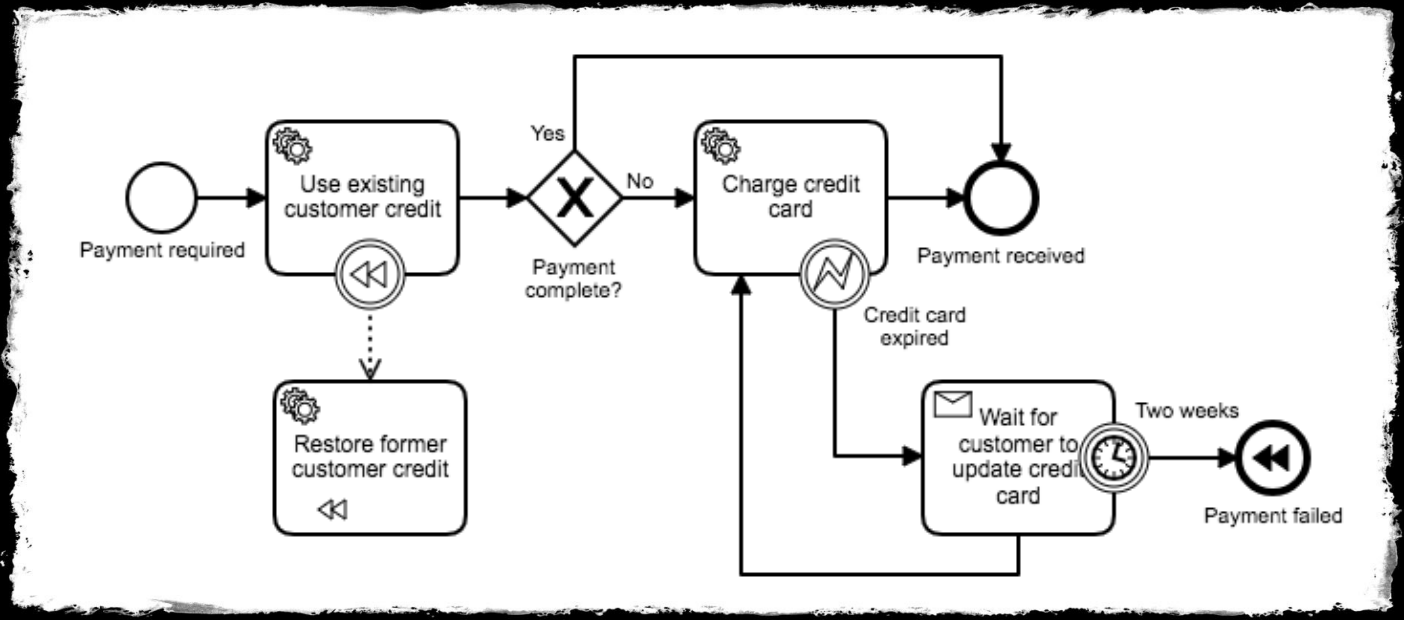
http://vasters.com/archive/Sagas.html

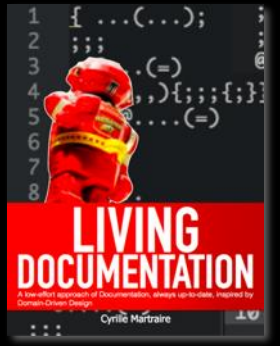
BPMN

Business Process Model and Notation

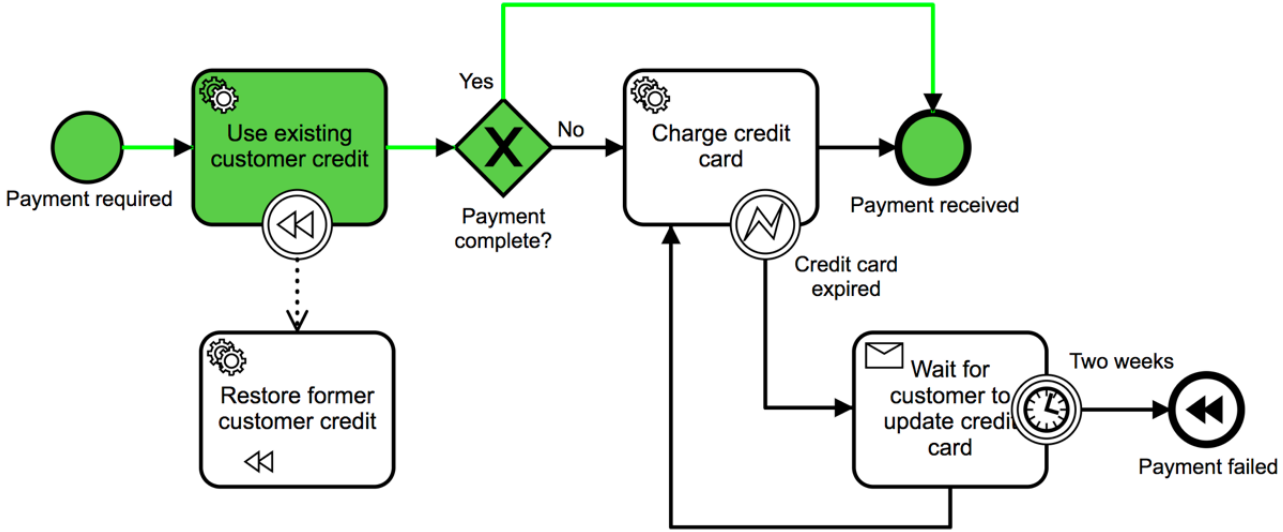
ISO Standard

Living documentation for long-running behaviour

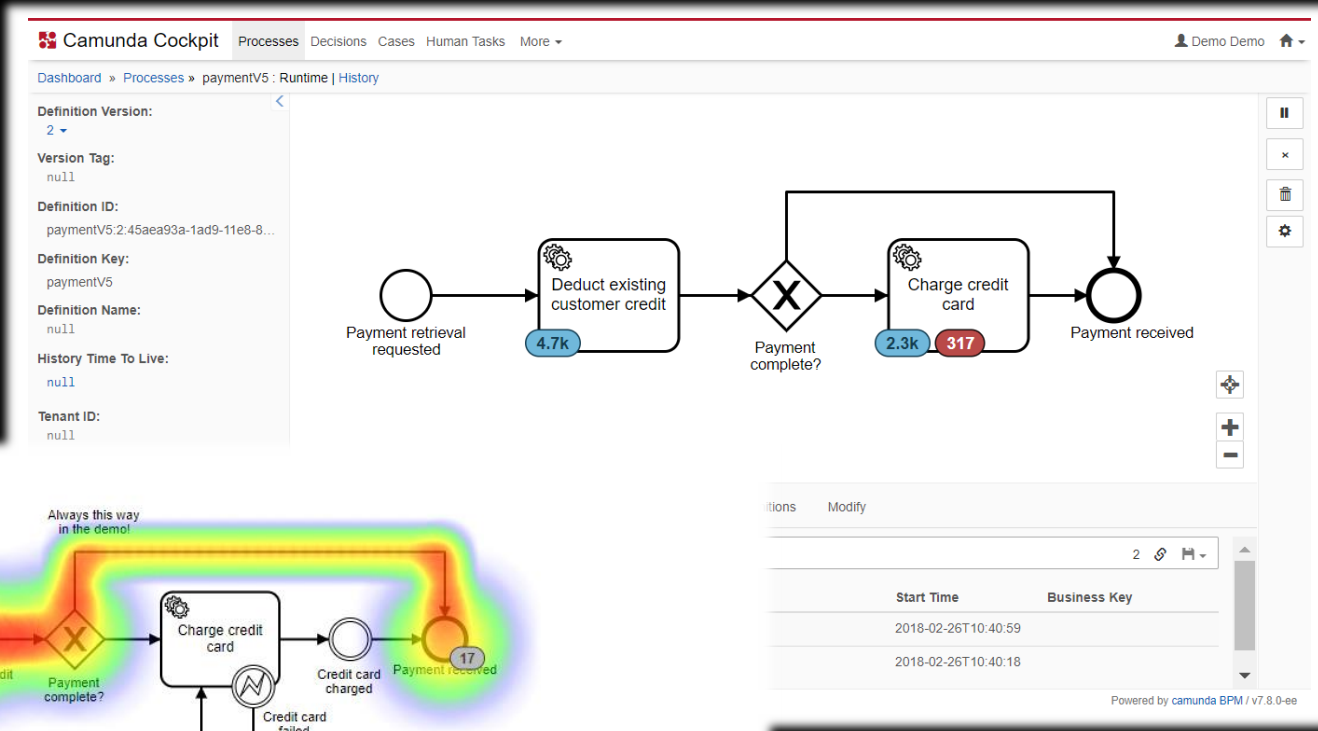




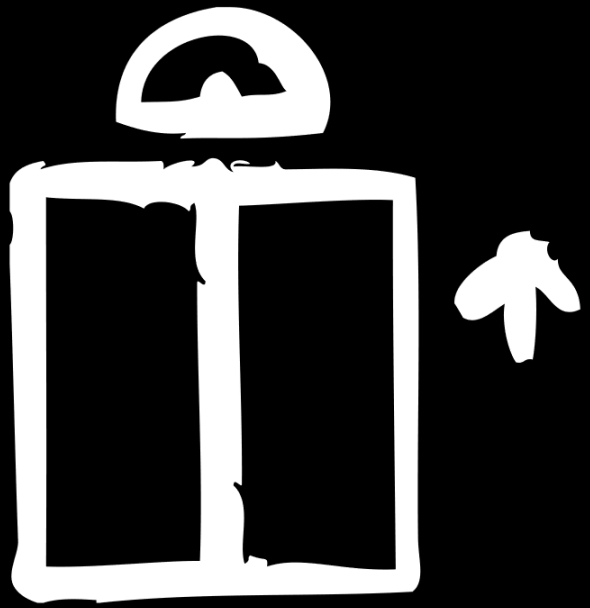
Visual HTML reports for test cases



BizDevOps

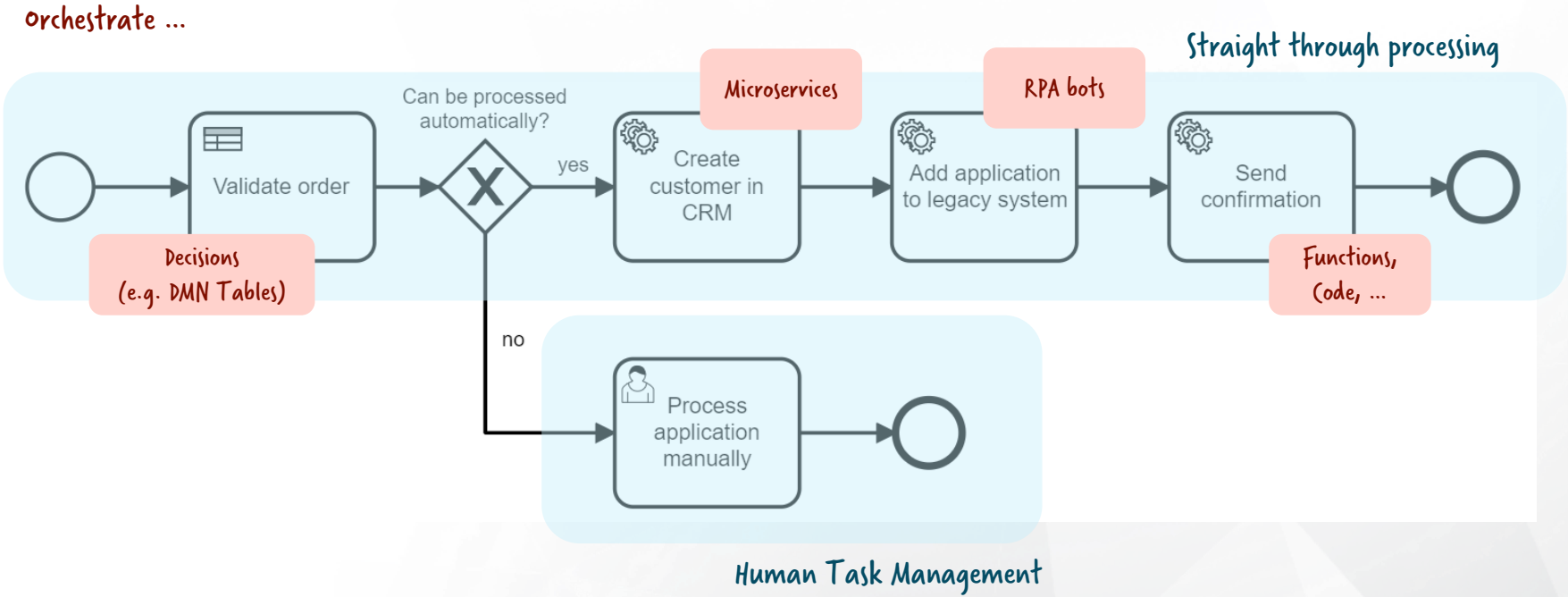

Camunda Cockpit
Processes
Decisions
Cases
Human Tasks
More
Demo Demo
Dashboard » Processes » paymentV5 : Runtime | History
Definition Version:
2
Version Tag:
null
Definition ID:
paymentV5:2:45aea93a-1ad9-11e8-8...
Definition Key:
paymentV5
Definition Name:
null
History Time To Live:
null
Tenant ID:
null
Payment retrieval requested
Deduct existing customer credit
4.7k
Payment complete?
Charge credit card
2.3k
317
Payment received
Modify
Start Time
Business Key
2018-02-26T10:40:59
2018-02-26T10:40:18
Powered by camunda BPM / v7.8.0-ee
Does nothing in this demo!
Always this way in the demo!
Payment retrieval requested
Deduct existing customer credit
Customer credit used
Payment complete?
Charge credit card
Credit card charged
Payment received
17
Credit card failed
Restore customer credit
Ask customer to update credit card
Wait for customer to update credit card
7 days
Payment failed

Long Running Capabilities



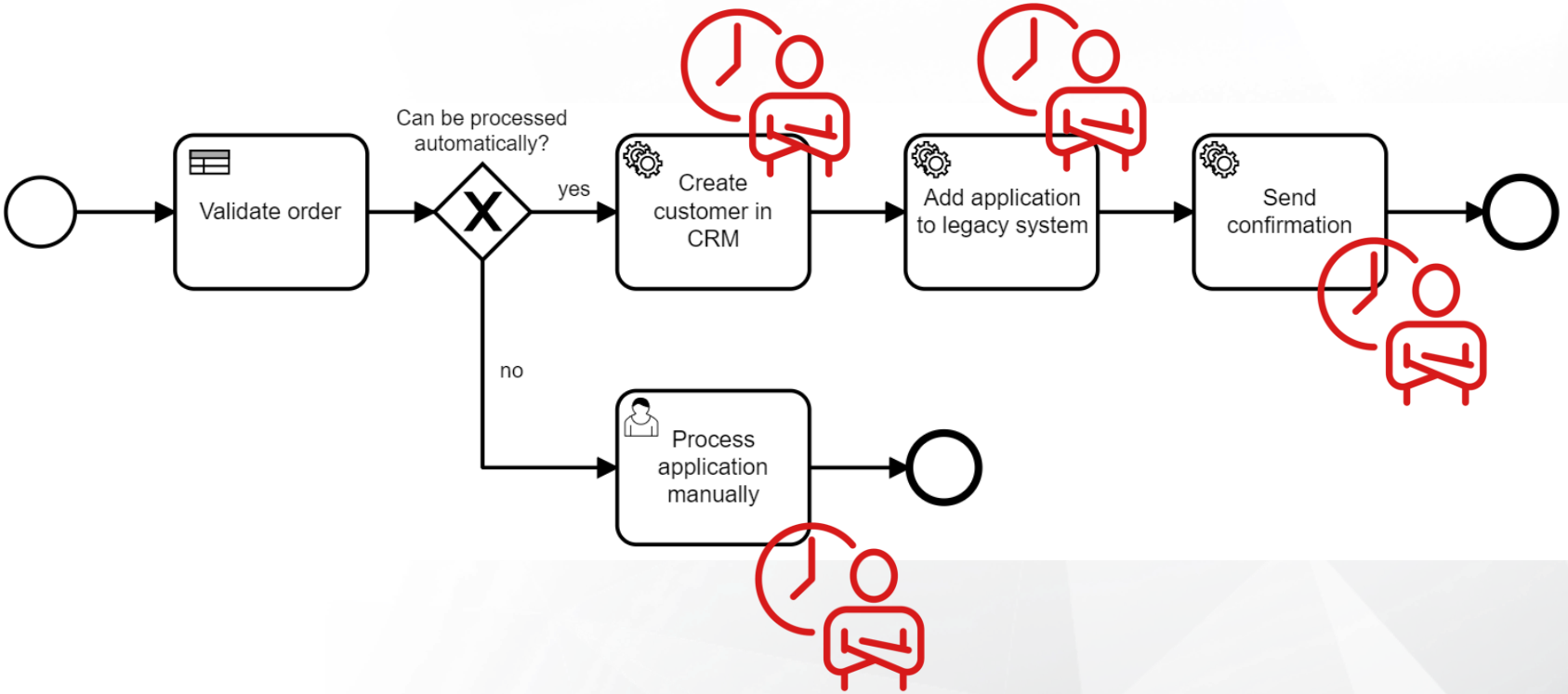
Process & consistency decisions need to be elevated to the business level!

Recap: Workflow engines orchestrate anything



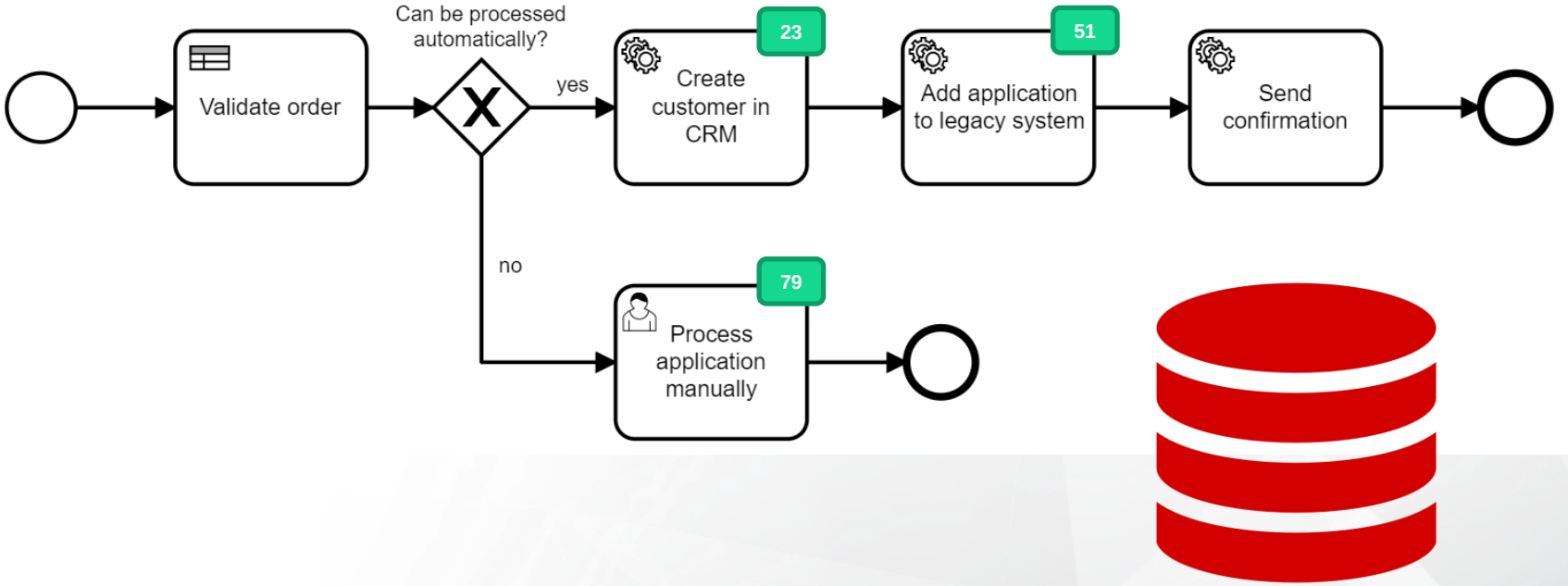
CAMUNDA

Recap: The challenge are long running tasks = waiting



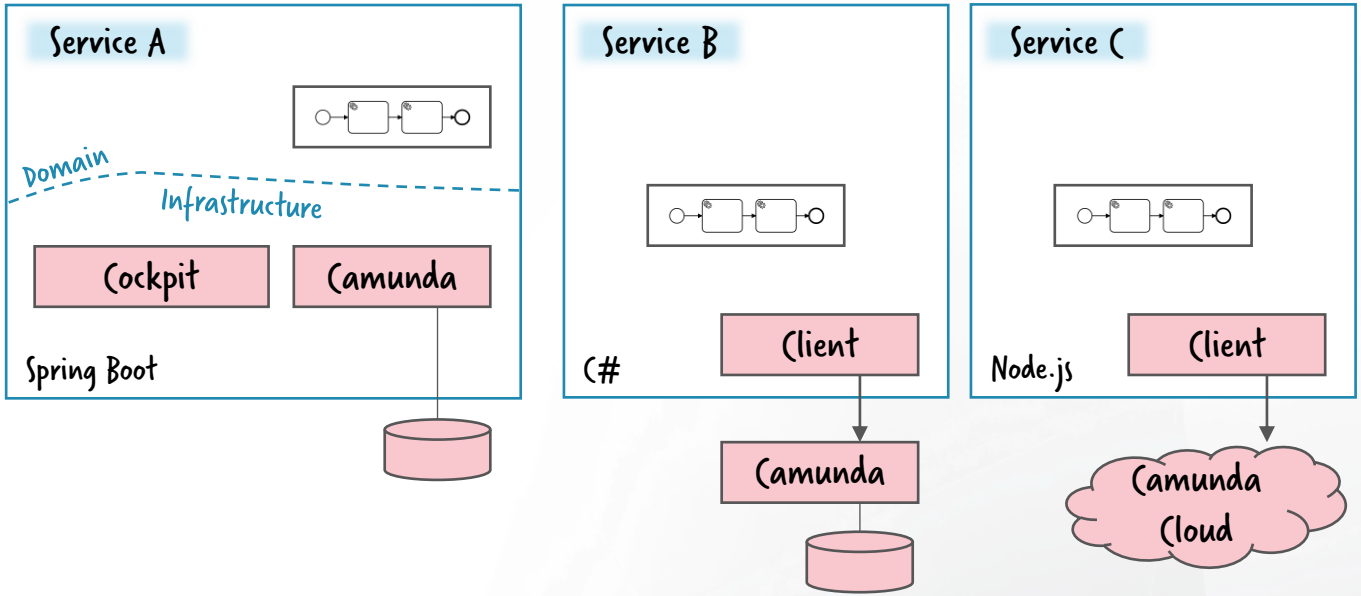
CAMUNDA

Recap: A workflow engine persists state



CAMUNDA

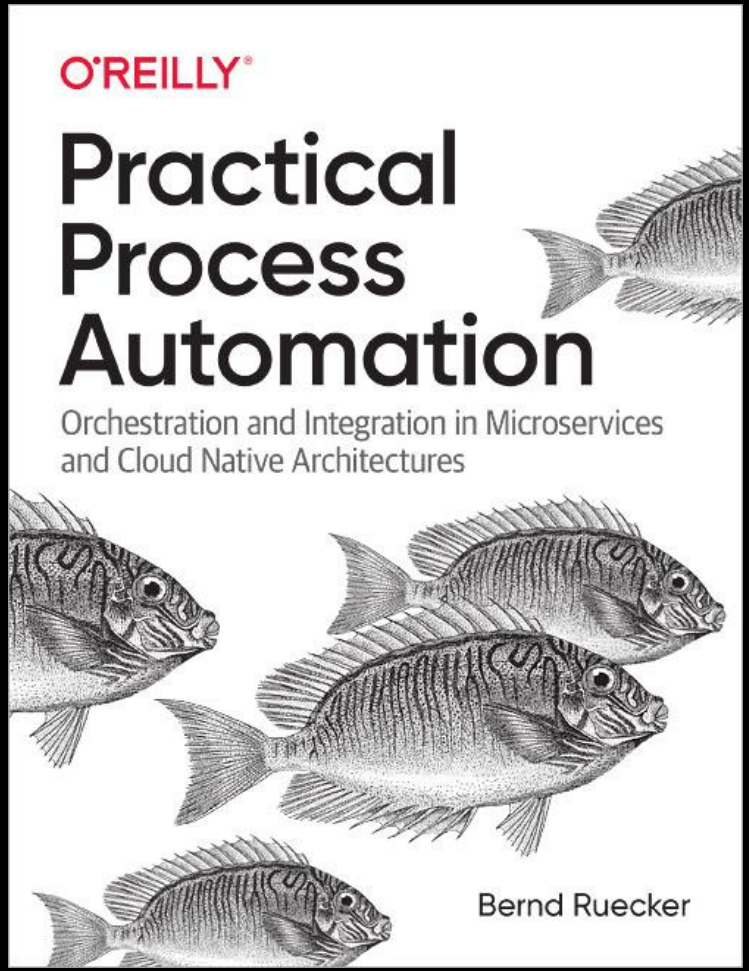
Decentralized workflow engines with flexible architecture options



Recap

- You need capabilities for long running behavior for technical and business reasons
- Workflow engines are a great fit – make sure you use a developer friendly one
- Balance orchestration (coordination) and choreography (reactive)

Want to learn more?



https://learning.oreilly.com/get-learning/?code=PPAER20

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/