Workflow Automation Reinvented
What people think, when I say „Workflow Automation”...
What I think, when I say „Workflow Automation“
What people think, when I say „BPM“...

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

Death by properties panel
What I think, when I say „BPM“
Software Development
Architecture and Design 2019 Q1 Graph

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

Innovators
- Blockchain and Distributed Ledgers
- Service meshes (Envoy, Linkerd, Istio)
- HTTP/3

Early Adopters
- Reactive Programming
- Functional Programming
- CQRS
- Actor Model
- "Serverless" (FaaS/BaaS/DBaaS/PaaS)
- gRPC and HTTP/2
- GraphQL
- Evolutionary Architecture
- (Lightweight) workflow and decision automation platforms

Early Majority
- Event-Driven Architecture (and Event Sourcing)
- Eventual Consistency

Late Majority
- Microservices
- Domain-Driven Design
- Behaviour-Driven Design
- Test-Driven Design
- REST
Why workflow automation?
5 Workflow Automation Use Cases You Might Not Have Considered

9 Apr 2018 3:00am, by Bernd Rücker

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

[Graph with time units: <= milliseconds, seconds, minutes, weeks, ...]
Use cases for workflow automation

- Business processes automation

| <= milliseconds | seconds | minutes, weeks, ... |
Real-life examples
Use cases for workflow automation

Business processes automation

| <= milliseconds | seconds | minutes, weeks, ... |
Use cases for workflow automation

<table>
<thead>
<tr>
<th>&lt;= milliseconds</th>
<th>seconds</th>
<th>minutes, weeks, ...</th>
</tr>
</thead>
</table>

Business processes automation
Use cases for workflow automation

- Business processes automation
- Communication in distributed systems

Time scale:
- $\leq$ milliseconds
- seconds
- minutes, weeks, ...
Ever called a REST API?
Stateful retry
Distributed systems introduce complexity you have to tackle!
It is impossible to differentiate certain failure scenarios. Independant of communication style!
Distributed systems introduce complexity you have to tackle!
Distributed systems introduce complexity you have to tackle!
Live hacking
Warning: Contains Opinion
Bernd Ruecker
Co-founder and Chief Technologist of Camunda

Berlin, Germany

mail@berndruecker.io
@berndruecker
Live hacking

https://github.com/berndruecker/flowing-retail/tree/master/rest/java/payment-camunda
BPMN - Business Process Model and Notation
ISO Standard
Long running services* provide a better API!

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

Order → Retrieve Payment → Payment
Example

If the credit card was rejected, the customer can provide new details.
Client of **dumb endpoints** easily become a god services.

If the credit card was rejected, the customer can provide new details.
Who is responsible to deal with problems?

If the credit card was rejected, the customer can provide new details.
Long running services

Order → Retrieve Payment → Payment received → Payment failed → Order

Payment → Credit Card → Rejected → Payment

Smart endpoints are potentially long-running
Long running services

Order → Retrieve Payment → Payment → Credit Card

Payment received → Payment failed

Charge credit card → Rejected

Inform customer of rejection → Wait for customer to update card → Payment received → 7 days → Payment failed
Use cases for workflow automation

- Business processes automation
- Communication in distributed systems

- Always short running
- Short running, but potentially long running
- Long running
Use cases for workflow automation

- **Business processes automation**
- **Communication in distributed systems**

- **<= milliseconds**
  - always short running
  - short running, but potentially long running
- **seconds**
- **minutes, weeks, ...**
  - long running
Use cases for workflow automation

- Business processes automation
- Communication in distributed systems
- Orchestration

Time intervals:
- <= milliseconds: always short running
- seconds: short running, but potentially long running
- minutes, weeks, ...: long running
Microservices...

Monolith

Functionality A
Functionality B
Functionality C
Functionality D

Service A
Service B
Service C
Service D
Some Service
Some Service
Some Service
Some Service
order fulfillment example: dash button
(Micro-)services

- Checkout
- Payment
- Inventory
- Shipment
Event-driven architecture
Peer-to-peer event chains

- Order placed → Checkout
- Payment received → Payment
- Goods fetched → Inventory
- Goods shipped → Shipment
- Pay item → Ship item
- Fetch item
Peer-to-peer event chains

- Order placed
- Checkout
- Payment received
- Inventory
- Goods fetched
- Shipment
- Goods shipped
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
The danger is that it’s very easy to make nicely decoupled systems with event notification, without realizing that you’re losing sight of that larger-scale flow, and thus set yourself up for trouble in future years.

https://martinfowler.com/articles/201701-event-driven.html
Peer-to-peer event chains

Order placed → Checkout → Payment → Inventory → Shipment → Goods fetched → Goods shipped

Fetch the goods **before** the payment
Peer-to-peer event chains

- Order placed
- Payment received
- Goods fetched
- Goods shipped
- Checkout
- Inventory
- Shipment

Fetch the goods before the payment
What we wanted

vs. what we got

Photo by Lijian Zhang, available under Creative Commons SA 2.0 License and Pedobear19 / CC BY-SA 4.0
Extract the end-to-end responsibility

*Commands* have an intent about what needs to happen in the future
It still can be messaging!
Workflows implement stateful orchestration logic.
No BPM(N) monoliths

https://blog.bernd-ruecker.com/avoiding-the-bpm-monolith-when-using-bounded-contexts-d86be6308d8
Workflows implement stateful orchestration logic

Order placed → Retrieve payment → Payment received → Fetch goods → Goods fetched → Ship goods → Goods shipped → Order delivered

Checkout

Order

Payment

Inventory

Shipment
Use cases for workflow automation

- Business processes automation
- Communication in distributed systems
- Orchestration

Time intervals:
- <= milliseconds: always short running
- seconds: short running, but potentially long running
- minutes, weeks, ...: long running
Use cases for workflow automation

- Consistency
- Business processes automation
- Orchestration
- Communication in distributed systems

- Business
- IT

- <= milliseconds
- seconds
- minutes, weeks, ...

- always short running
- short running, but potentially long running
- long running
The classical example

1. book hotel
2. book car
3. book flight

Failure!
Life beyond Distributed Transactions: an Apostate’s Opinion
Position Paper

Pat Helland
Amazon.Com
705 Fifth Ave South
Seattle, WA 98104
USA
PHelland@Amazon.com

The positions expressed in this paper are personal opinions and do not in any way reflect the positions of my employer Amazon.com.

ABSTRACT

Many decades of work have been invested in the area of distributed transactions including protocols such as 2PC, Paxos, and various approaches to quorum. These protocols provide the application programmer a façade of global serializability. Personally, I have invested a non-trivial portion of my career as a strong advocate for the implementation and use of platforms.

Instead, applications are built using different techniques which do not provide the same transactional guarantees but still meet the needs of their businesses. This paper explores and names some of the practical approaches used in the implementations of large-scale mission-critical applications in a world which rejects distributed transactions. We discuss the management of fine-grained pieces of application data which may be partitioned over time as the application grows. We also discuss the design patterns used in sending messages between these non-transactional pieces of the application.

Pat Helland
Distributed Systems Guru
Worked at Amazon, Microsoft & Salesforce
“Grown-Ups Don’t Use Distributed Transactions”
The classical example

1. book hotel
2. book car
3. book flight
   
4. cancel hotel
5. cancel car
6. Trigger compensations
BPMN

Saga Pattern

(implemented by BPMN compensation)

https://github.com/berndruecker/trip-booking-saga-java
Use cases for workflow automation

- Distributed Transactions
- Business processes automation
- Communication in distributed systems
- Orchestration

Time intervals:
- ≤ milliseconds: always short running
- seconds: short running, but potentially long running
- minutes, weeks, ...: long running
Use cases for workflow automation

- Decision Automation
- Distributed Transactions
- Business processes automation
- Communication in distributed systems
- Orchestration

- Always short running
- Short running, but potentially long running
- Long running

- ≤ milliseconds
- Seconds
- Minutes, weeks, ...
# Decisions with DMN

## Risk Assessment

<table>
<thead>
<tr>
<th>C</th>
<th>Input</th>
<th>Output</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Age = 37, Car manufacturer = Porsche, Car type = 911</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>&lt;= 21, integer</td>
<td>Risk = &quot;Beginner&quot;</td>
<td>&quot;yellow&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;= 25, string = &quot;Porsche&quot;</td>
<td>Risk = &quot;Young and too fast&quot;</td>
<td>&quot;red&quot;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;= 30, string = &quot;BMW&quot;</td>
<td>Risk = &quot;Young and fast&quot;</td>
<td>&quot;yellow&quot;</td>
</tr>
<tr>
<td>4</td>
<td>&quot;Porsche&quot;, &quot;911&quot;</td>
<td>Risk = &quot;Fast and furious&quot; = &quot;yellow&quot;</td>
<td>&quot;yellow&quot; = yellow</td>
</tr>
<tr>
<td>5</td>
<td>&quot;BMW&quot;, &quot;X3&quot;</td>
<td>Risk = &quot;High value vehicle&quot;</td>
<td>&quot;yellow&quot;</td>
</tr>
</tbody>
</table>
Lost in transaction? Strategies to manage consistency in distributed systems

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

All occurrences on conferences...

Slides

3 common pitfalls in microservice integration and how to avoid them

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

All occurrences on conferences...

Slides

Complex event flows in distributed systems

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

All occurrences on conferences...

Slides
Workflow Automation is important in modern architectures!

Microservices...

Distributed systems

What we wanted

Grown-Ups Don't Use
Distributed Transactions

Pat Helland
Distributed Systems Guru
Worked at Amazon
Microsoft & Salesforce
Thoughts on the state machine / workflow engine market
Thoughts on the state machine/.workflow engine market

- **Stack Vendors, Pure Play BPMS, Low Code Platforms**
  - PEGA, IBM, SAG, ...

- **OSS Workflow or Orchestration Engines**
  - Camunda, Zeebe, jBPM, Activiti, Flowable, Mistral, ...

- **Integration Frameworks**
  - Apache Camel, Balerina, ...

- **Homegrown frameworks to scratch an itch**
  - Uber, Netflix, AirBnb, ING, ...

- **Cloud Offerings**
  - AWS Step Functions, Azure Durable Functions, ...

- **Data Pipelines**
  - Apache Airflow, Spring Data Flow, ...
Does it support stateful operations?
Does it support the necessary flow logic?
Does it support BizDevOps?
Does it scale?
AWS Step Functions

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

Visibility + Context
Biz Dev Ops

- Understand and discuss business processes
- Leverage state machine & workflow engine
- Operate with visibility and context

Improve communication

- Evaluate optimizations in-sync with implementation
- Living documentation
- Visibility in testing

@berndruecker
Example: Storage

Persistent State

RDMS
28. Persisting State Machine

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

28.2 Using StateMachinePersister

Building a StateMachineContext and then restoring a state machine from it has always been a little bit of a black magic if done manually. The StateMachinePersister aims to ease these operations by providing `persist` and `restore` methods. Default implementation of the StateMachinePersister is used automatically.

Usage of a StateMachinePersister is easy to demonstrate by following a snippet from tests. We start by creating two StateMachinePersisters named `machine1` and `machine2`. We could build different machines for this demonstration using various other ways but this serves the purpose.

```java
static class InMemoryStateMachinePersist implements StateMachinePersist<String, String, String> {
    private final HashMap<String, StateMachineContext<String, String>> contexts = new HashMap<>();

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

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

Persistent State

RDMS

Do it yourself

Persistent change

camunda

Spring Statemachine

zeebe by Camunda
Workflow automation at scale!

What we currently teach workflow automation to be able to do

What people think workflow automation can do

What workflow automation can already do today

low latency, high-throughput

low frequency, latency doesn’t matter
Why Zeebe?
Horizontally scalable and resilient
‘How Does Zeebe Compare to X?': An Evaluation Framework

We often get questions about how Zeebe is the same as or different from other tools and frameworks that can be used to orchestrate workflows. These “other tools” include:

- “Traditional” transactional, open-source BPMN workflow engines (such as Camunda BPM and Activiti)
- BPM Suites (such as Pega, Software AG)
- Homegrown open-source orchestration tools (such as Netflix Conductor and Uber Cadence)
- Orchestration tools from cloud providers (such as AWS Step Functions and Google Cloud Composer)
- Distributed tracing tools (such as Jaeger)
## Aspects to consider

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Camunda</th>
<th>Spring State Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Definition</td>
<td>BPMN 2.0 (graphical, XML, Model API)</td>
<td>Java API, UML-Generator</td>
</tr>
<tr>
<td>Visual?</td>
<td>YES</td>
<td>No</td>
</tr>
<tr>
<td>Tooling</td>
<td>Modeler, Cockpit, Optimize</td>
<td>-</td>
</tr>
<tr>
<td>Storage Runtime</td>
<td>RDMS</td>
<td>up to you</td>
</tr>
<tr>
<td>Storage History</td>
<td>RDMS</td>
<td></td>
</tr>
<tr>
<td>Scalability</td>
<td>Stateless Engine, RDMS is limit, Sharding possible</td>
<td>up to you</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>If RDMS is HA</td>
<td>up to you</td>
</tr>
<tr>
<td>Supported programming languages</td>
<td>Java, REST, Language Clients (JS, C#)</td>
<td>Java</td>
</tr>
</tbody>
</table>
My personal pro-tip for a shortlist ;-)
Thank you!