

The Microservice Dungeon 2.0

Große Softwaresysteme in der Programmier-Ausbildung

Stefan Bente

TH Köln
Cologne Institute for Digital Ecosystems (CIDE)
Software Architecture Lab (ArchiLab)



Technology
Arts Sciences
TH Köln

Agenda für diesen Vortrag

- Warum Microservices?
 - (in der Lehre und überhaupt)
- Was ist der Microservice Dungeon?
- Wofür eine Version 2.0?
 - (6 Design-Fehler)
- Fazit und Ausblick

Warum Microservices?

(in der akademischen Lehre und überhaupt)

Ein Absolvent frisch im Job ...

Ähhh ...

3 Mio Zeilen
Sourcecode

Viele neue
Features

... oder, mit Dunning-Kruger, ...

Easy 😊

Wie die Hochschule dafür vorbereitet ...



300 Zeilen
Sourcecode

ChatGPT,
programmiere
mir ...

Eine neue
Methode

Und was wir mit dem MSD wollten ...

100.000
Zeilen
Sourcecode

Wo liegt die
Wie stelle ich
API-Spec?
das Log-Level
ein?
Eine eigener
Service

Aber warum Microservices?

- Unabhängig deployte Services
- I.d.R. asynchrone Kommunikation



- **Entkoppeln** Entwicklungsteams
- Ermöglichen schnelles **Time2Market**
- Erlauben **Technologiefreiheit**

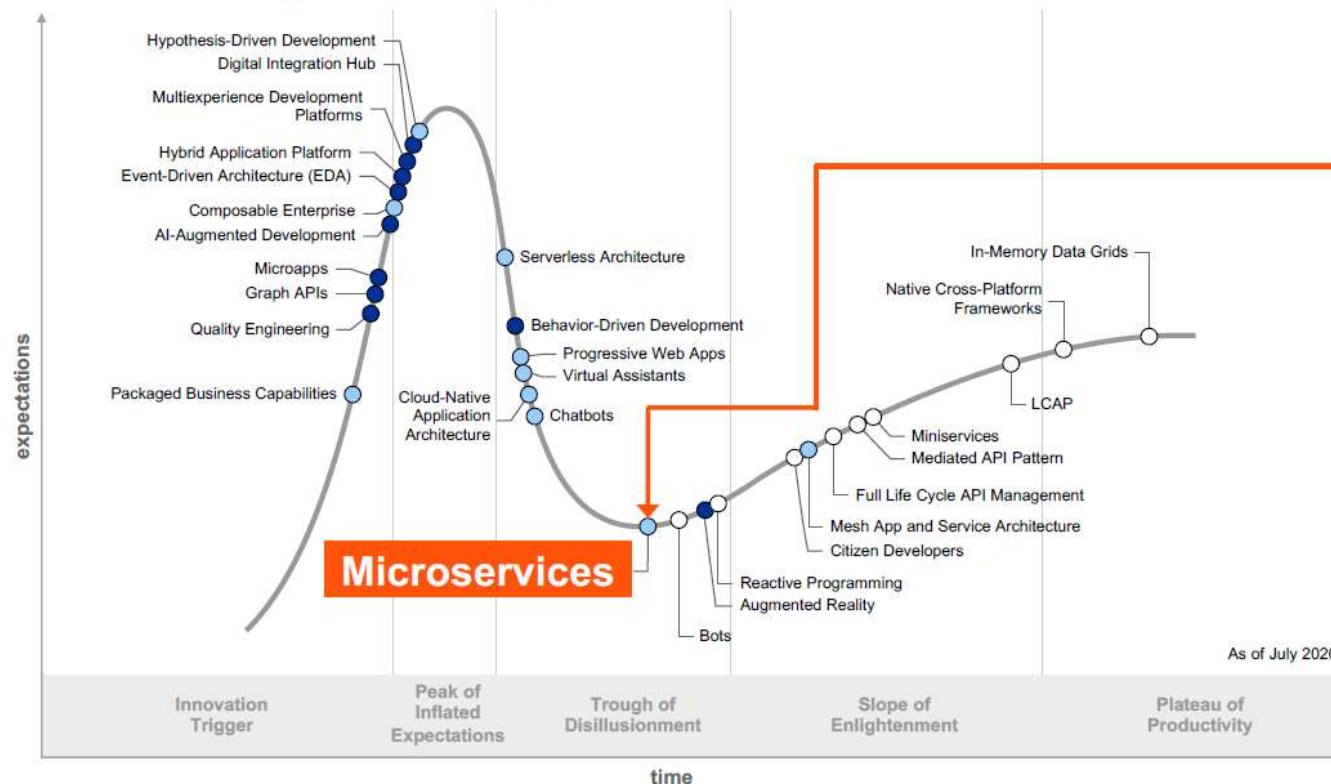
Popularität von Microservices

Google Trends (Topics 2010-2025)



Gartner Hype Cycle

Hype Cycle for Application Architecture and Development, 2020



Microservices are in the Trough of Disillusionment

Plateau will be reached:

○ less than 2 years ● 2 to 5 years ● 5 to 10 years ▲ more than 10 years ✕ obsolete before plateau

Source: [Hype Cycle for Application Architecture and Development, 2020](#), 31 July 2020 (G00450406)

ThoughtWorks Tech Radar

JAN
2014

Trial ?

We are seeing an uptick in adoption of microservices as a technique for distributed system design, both in Thoughtworks and in the wider community. Frameworks such as Dropwizard and practices like declarative provisioning point to a maturing of the technologies and tools. Avoiding the usual monolithic approach and being sympathetic to the need to replace parts of systems individually has important positive implications for the total cost of ownership of systems. We see this as having greatest impact in the mid-to-long term, specifically with respect to the two-to-five year rewrite cycle.

MAY
2013

Trial ?

OCT
2012

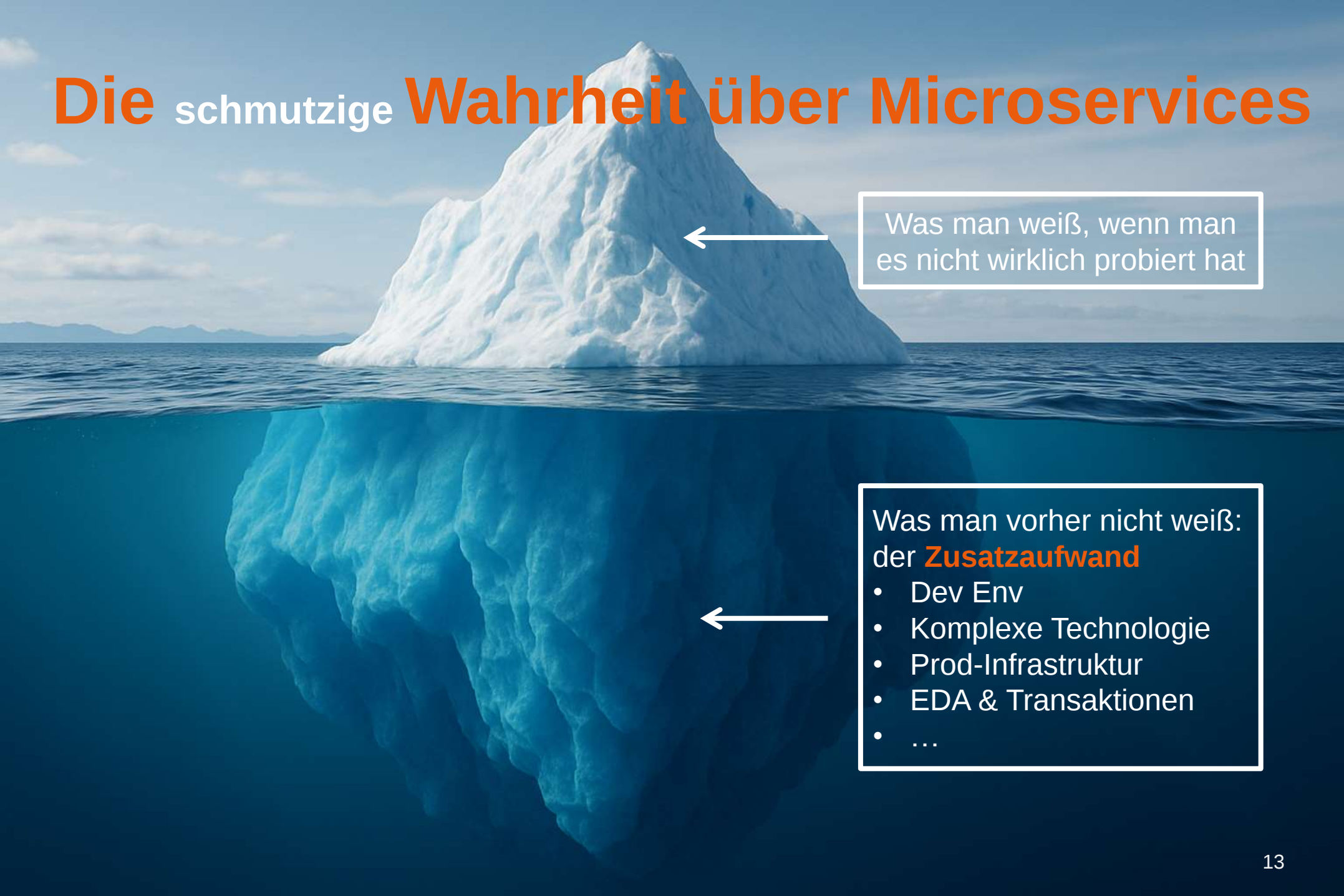
Trial ?

MAR
2012

Assess ?

Microservices, often deployed out-of-container or using an embedded HTTP server, are a move away from traditional large technical services. This approach trades benefits such as maintainability for additional operational complexity. These drawbacks are typically addressed using infrastructure automation and continuous deployment techniques. On balance, microservices are an effective way of managing technical debt and handling different scaling characteristics especially when deployed in a service oriented architecture built around business capabilities.

Die schmutzige Wahrheit über Microservices

An iceberg floating in the ocean. The visible tip represents the known aspects of microservices, while the much larger submerged part represents the hidden complexities and costs. Two white arrows point from text boxes to the respective parts of the iceberg.

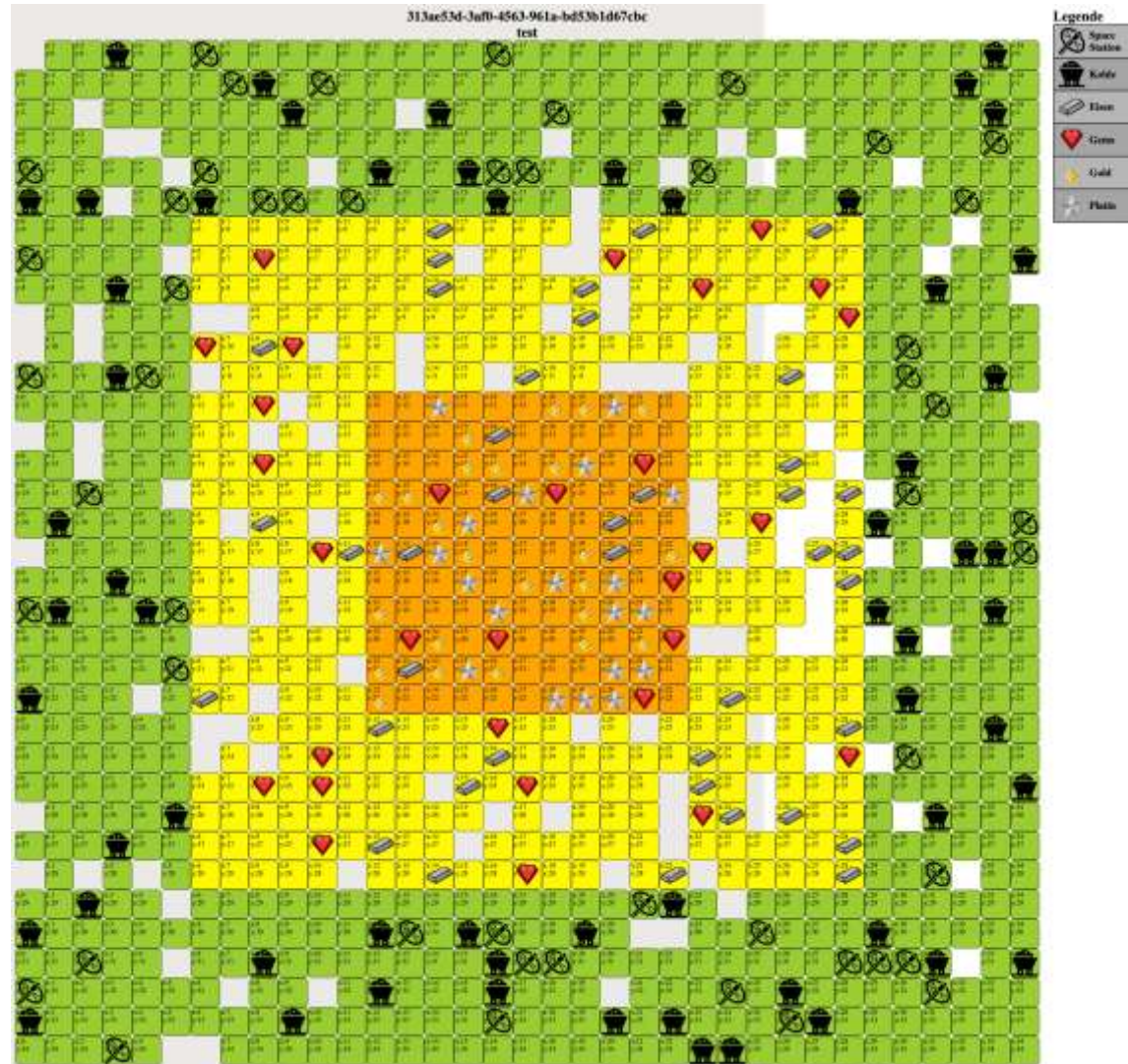
Was man weiß, wenn man
es nicht wirklich probiert hat

Was man vorher nicht weiß:
der **Zusatzaufwand**

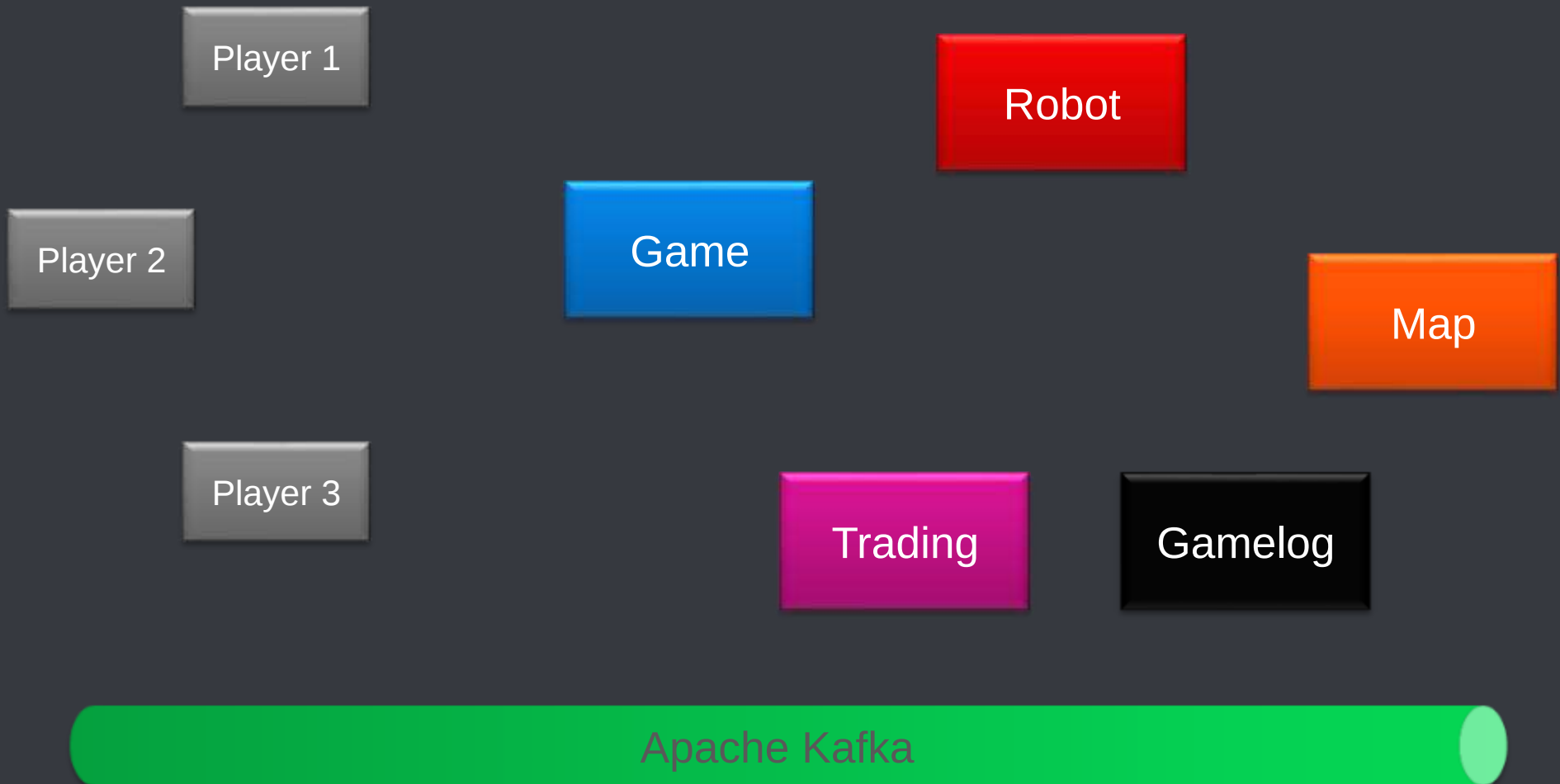
- Dev Env
- Komplexe Technologie
- Prod-Infrastruktur
- EDA & Transaktionen
- ...

Was ist der Microservice Dungeon?

Dungeon Game: Robots & Planeten



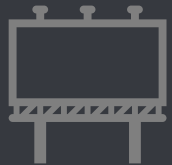
Aktuelle Architektur MSD 1.0



Entwickelt seit 2021



~175 Studierende



17 Veranstaltungen



3 Studiengänge



1 Peer-Reviewed Paper



Seit 2024 Teil von EFRE-Projekt

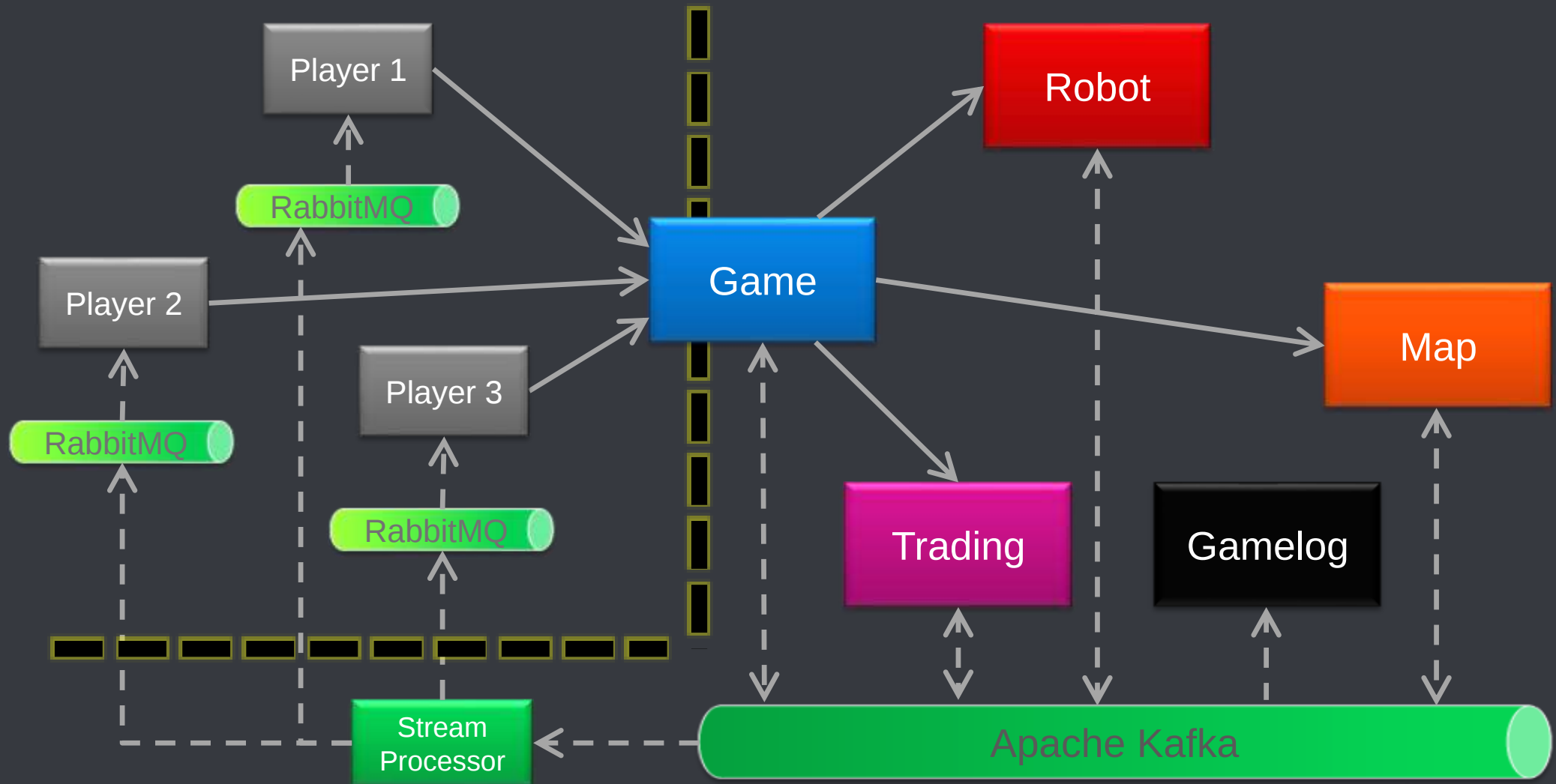
Wofür eine Version 2.0?

(6 Design-Fehler in V 1.0)

6 Fehler im MS Dungeon 1.0

1. Player-Services sind Bürger 2. Klasse
2. Fehlende Governance
3. Nicht genug über Domäne nachgedacht
4. Getaktete Event-Verarbeitung
5. Mix aus synchron & asynchron
6. „Wird schon alles gut gehen“

1) Player-Services: Bürger 2. Klasse



1) Player-Services: Bürger 2. Klasse

	Core Services	Player Services
Deployed in Kubernetes-Cluster	✓	✓
Überwacht von Infrastruktur-Monitoring	✓	✗
Senden Kommandos / Queries via REST	✓	(✓) nur an Game
Nehmen Kommandos / Queries via REST entgegen	✓	(✓) nur von eigenen UIs
Konsumieren Events von Kafka-Topic	✓	(✓) nur via Rabbit MQ
Produzieren Events für Kafka-Topic	✓	✗

2) Fehlende Governance



Viele Refactorings durch
verschiedene WMA

Entstanden aus
studentischen Projekten

Zahllose Branches für
explorative Praxisprojekte

2) FIX: Architekturprinzipien

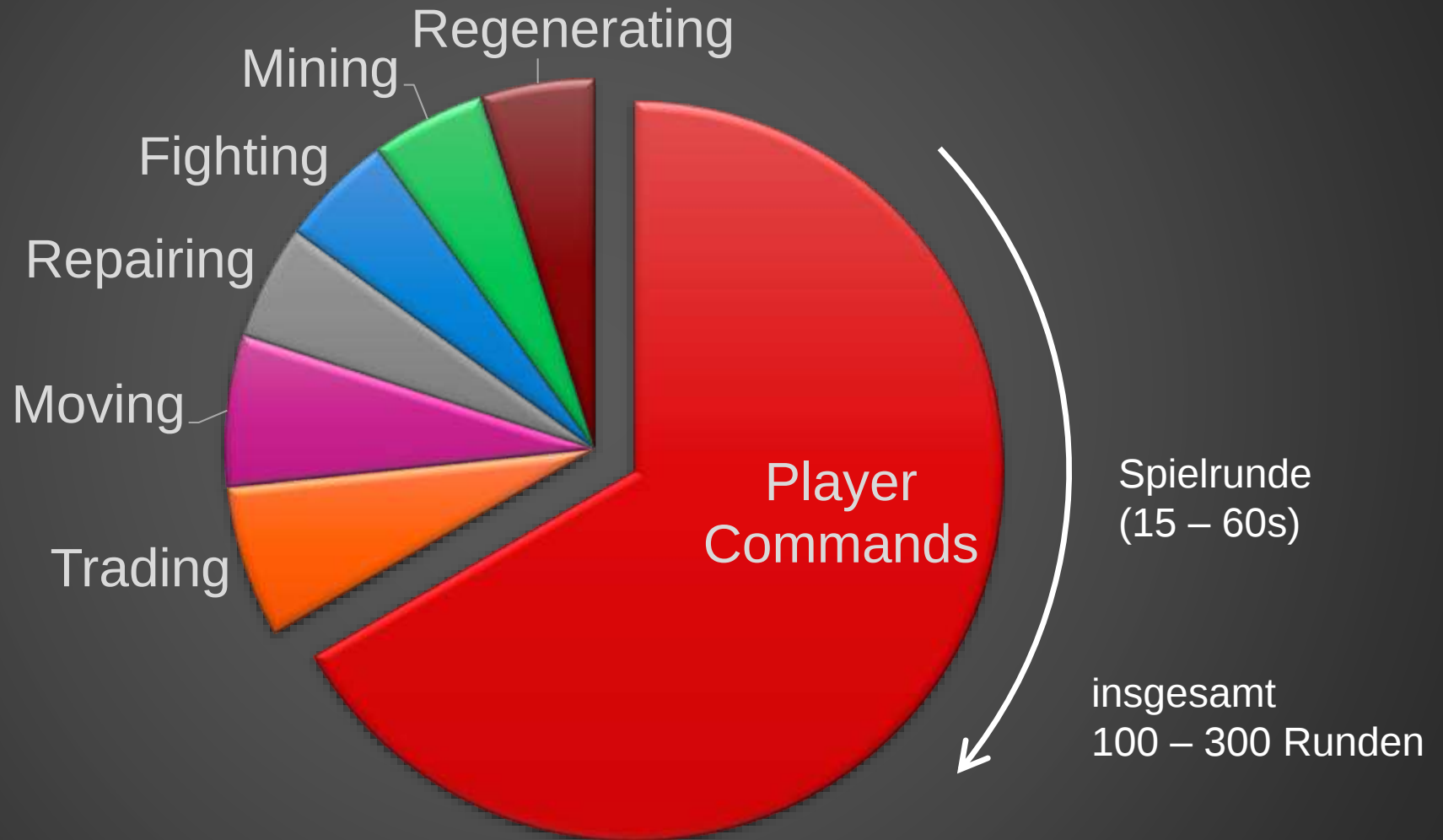
1. MSD resembles a **productive** microservice landscape, like e.g in e-commerce, as **closely as possible**.
2. MSD is supposed to be **fun** to work with. This is why it comes in the shape of a **game**.
3. There will be cases when (1) and (2) clash with each other. If so, **compromises** should rather be made in the area of **gaming rules**, rather than in the realism of its microservice architecture.

3) Nicht genug über Domäne nachgedacht

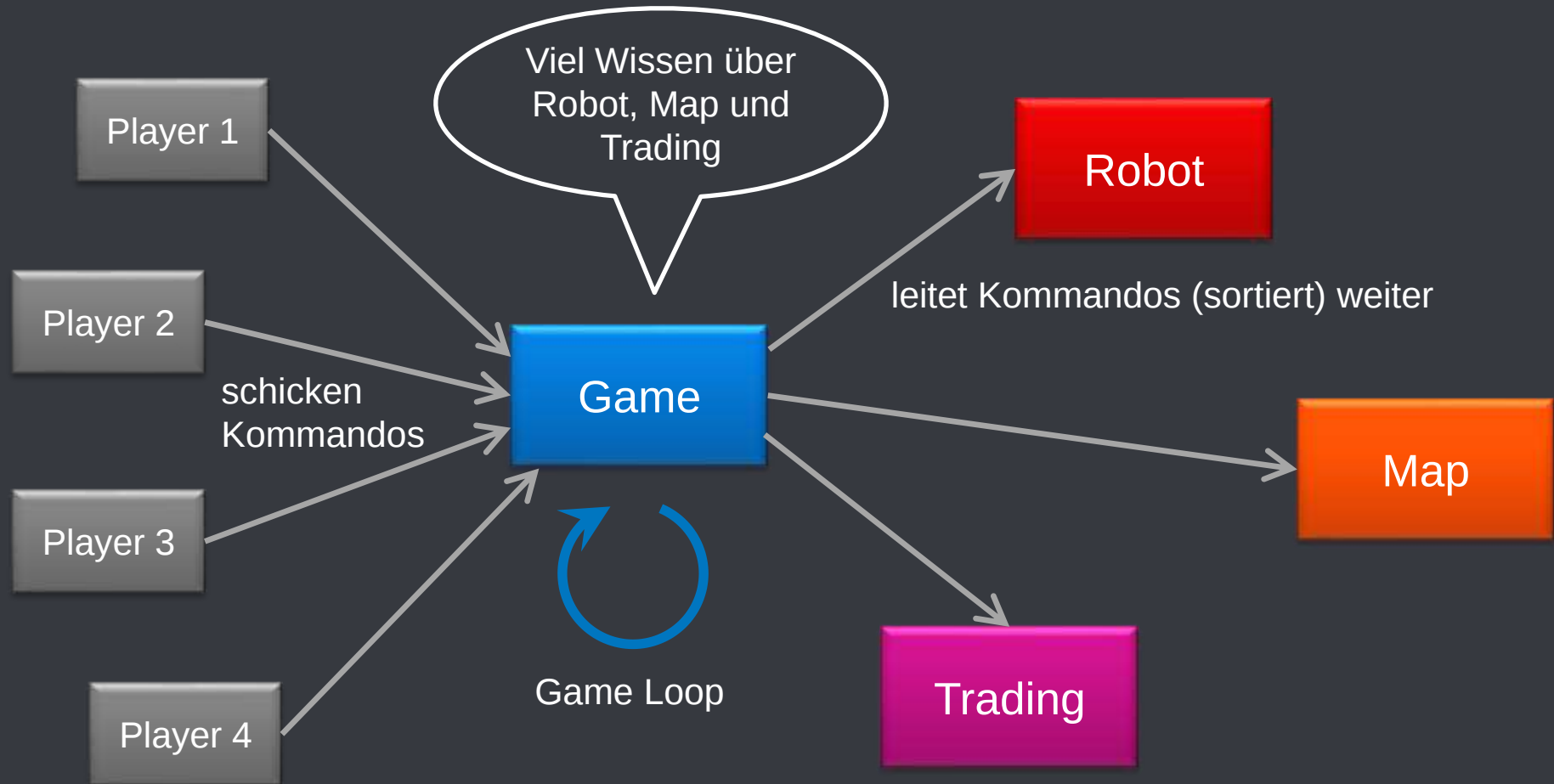


- => Erlaubt es wieder, gemäß DDD von der „Fachlichkeit“ aus zu denken

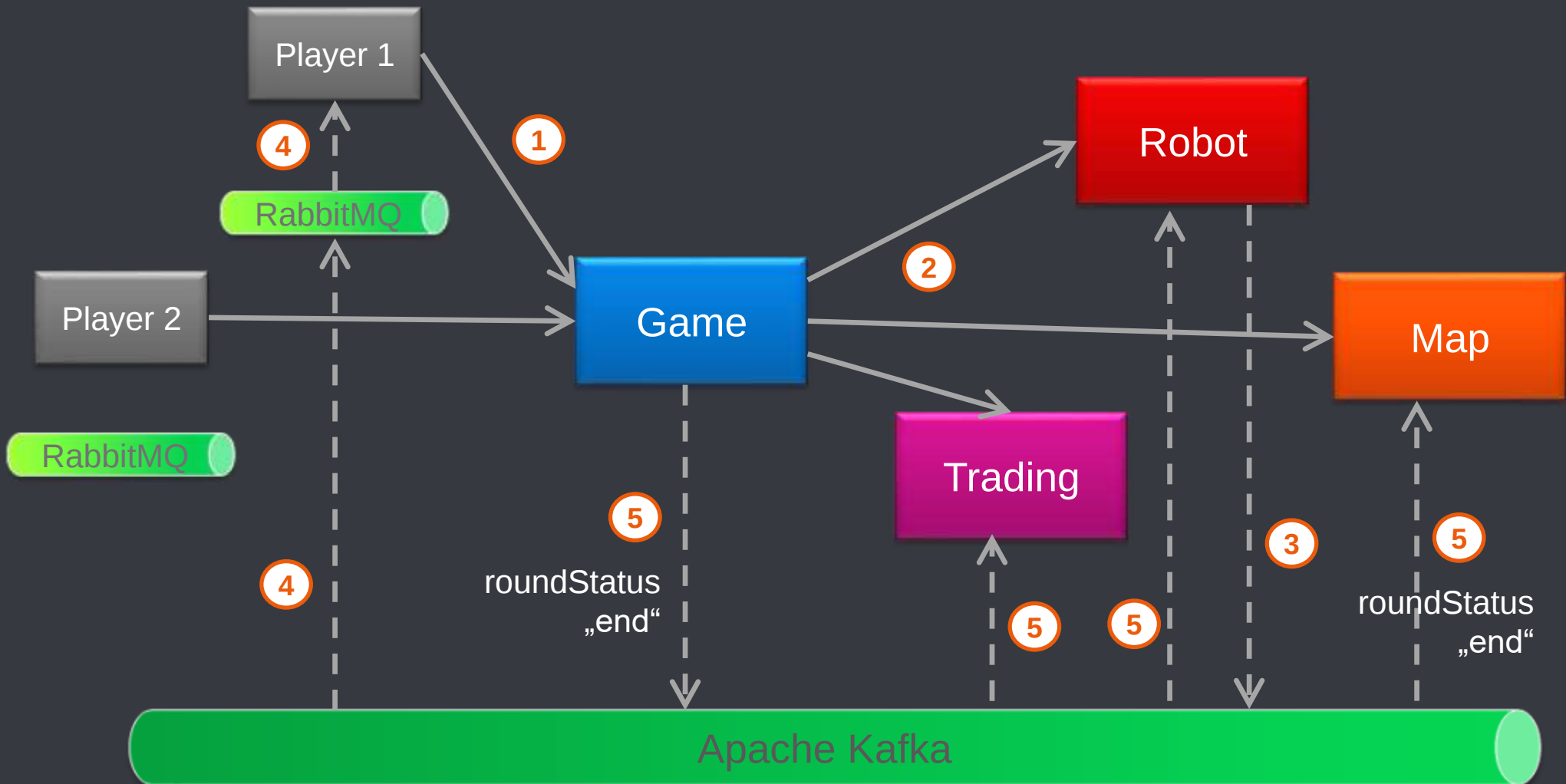
4) Getaktete Event-Verarbeitung ...



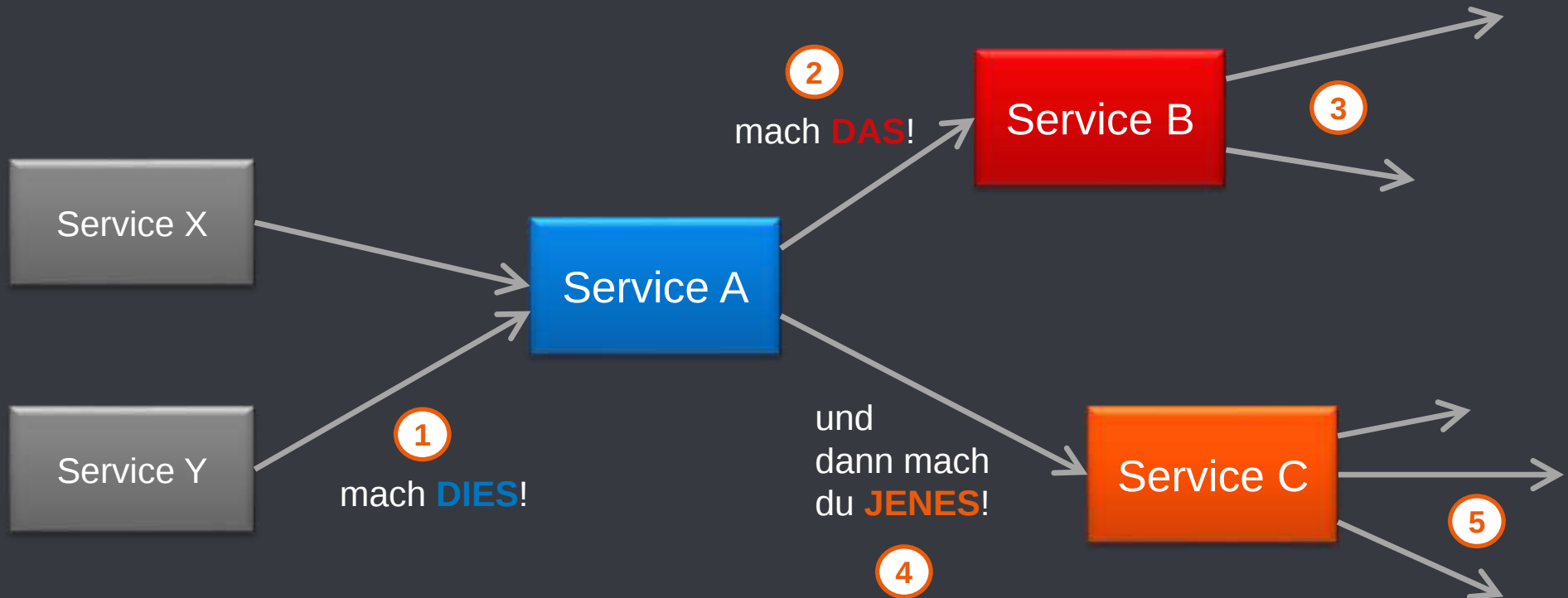
4) ... macht *Game* zum Orchestrator



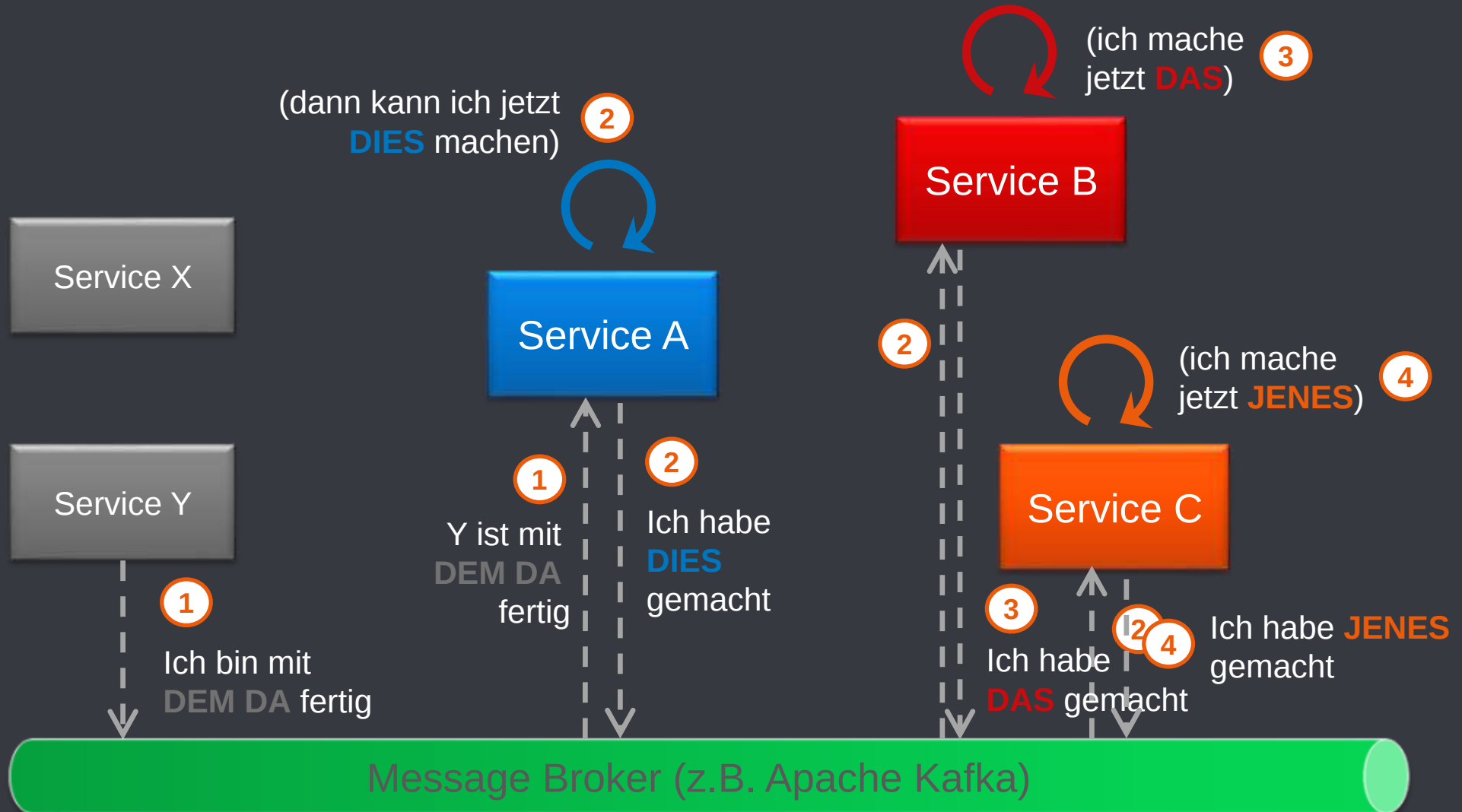
5) Mix aus synchron & asynchron



Wie geht synchrone Architektur?



Asynchrone / Event-getriebene Architektur

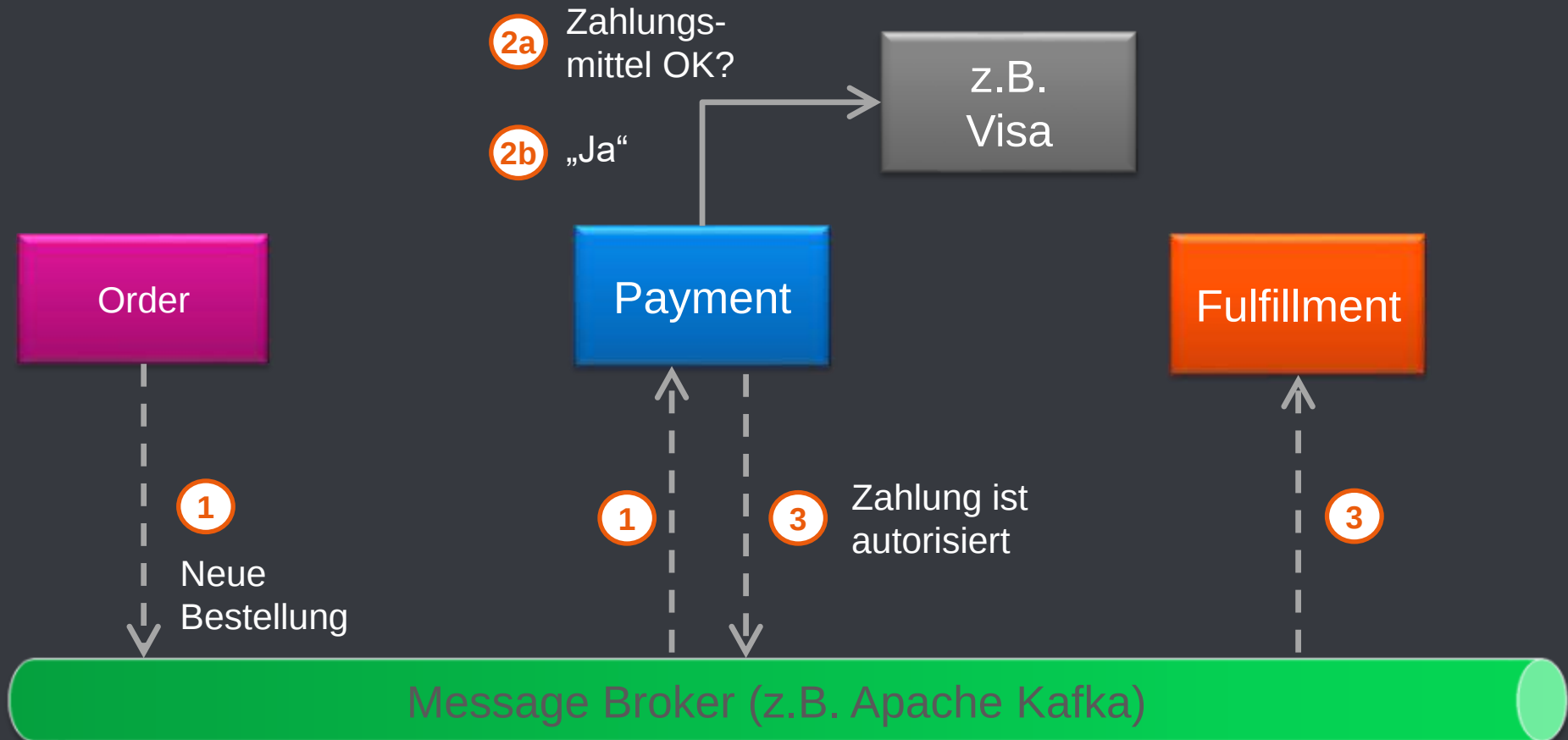


Probleme mit EDA (Event-Driven Architecture)

- Gewöhnungsbedürftige Denkweise
- Transaktionen dürfen keine Service-Grenzen überspannen
 - => Korrekturen sind aufwändig
- Saga-Pattern
 - Folge-Transaktionen & -Events als „Corrective Actions“

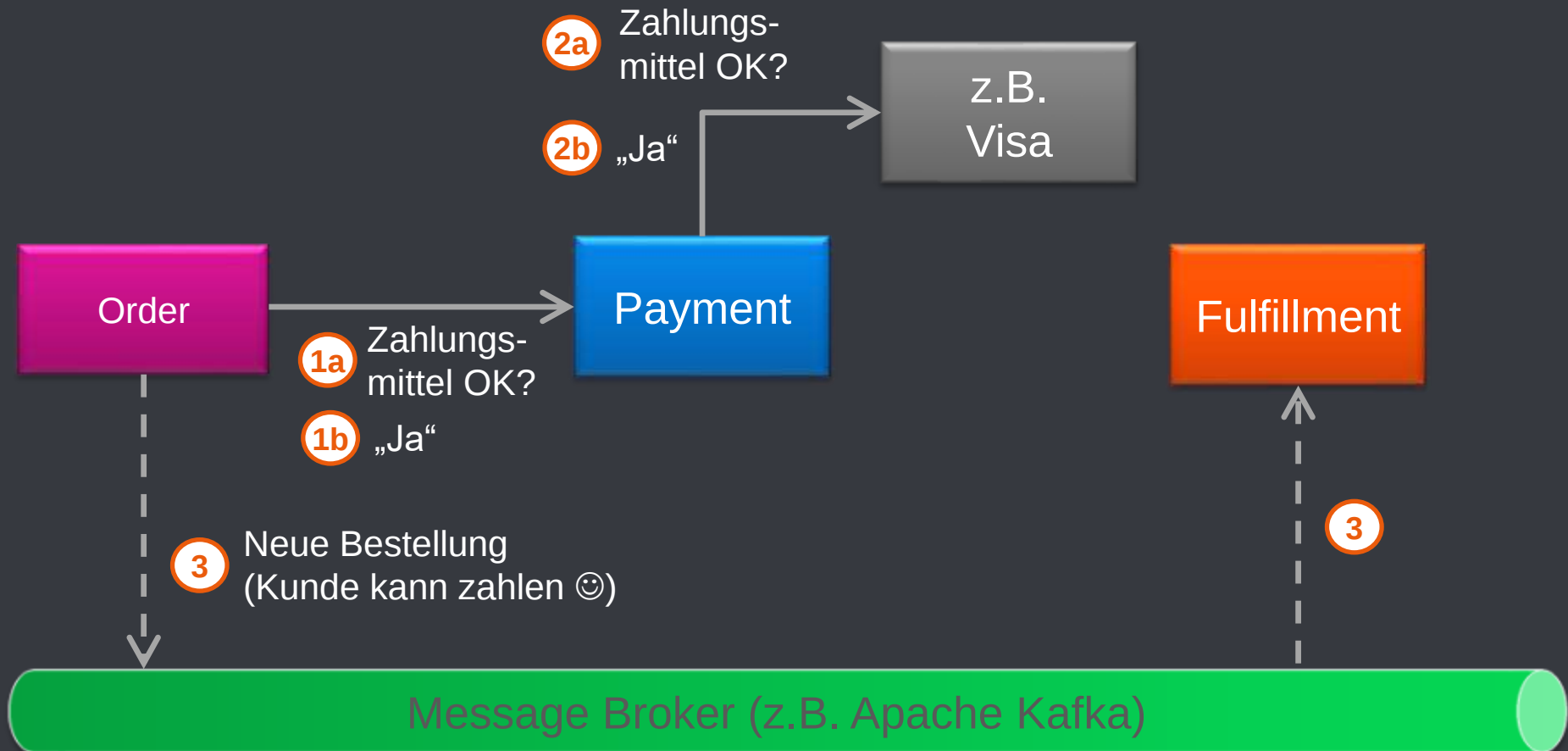
In der Industrie: SAGA vermeiden

1) Mix synchron / asynchron ...



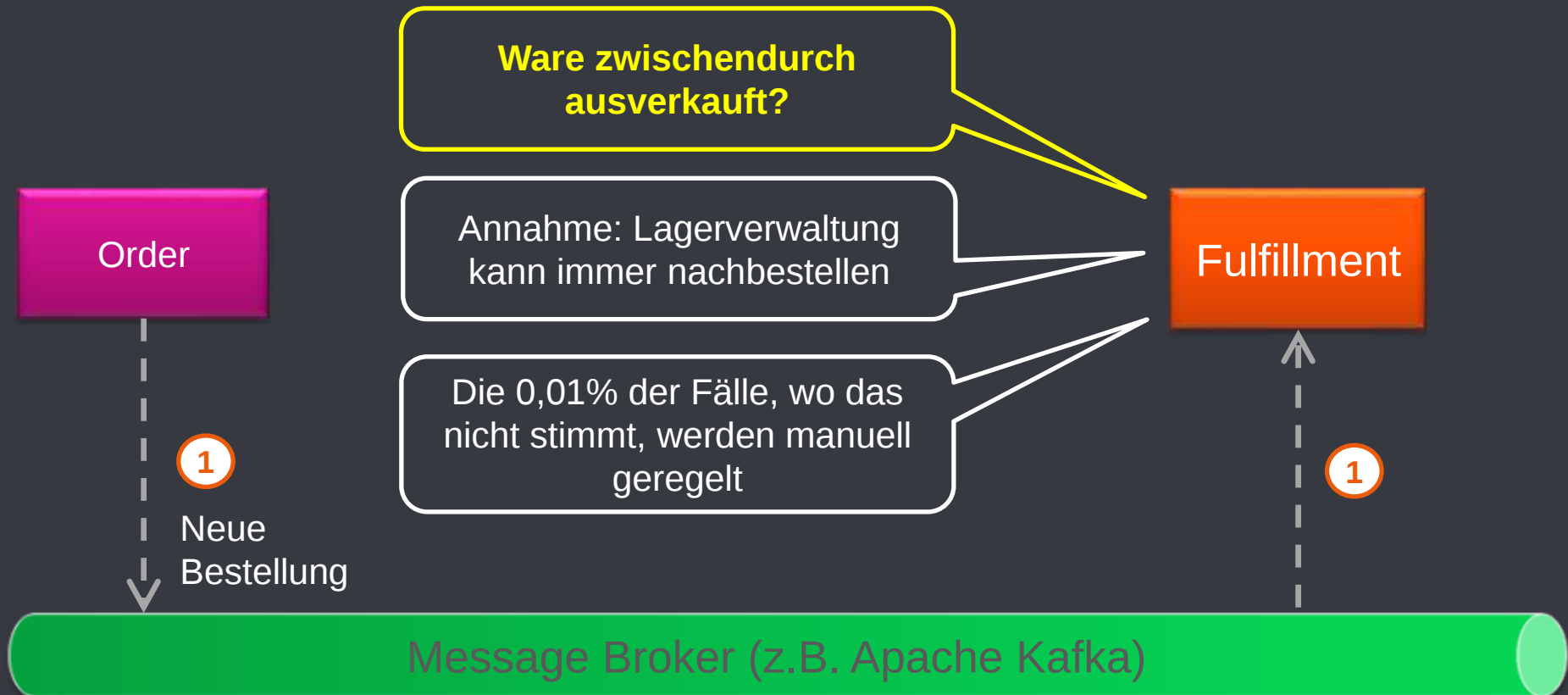
In der Industrie: SAGA vermeiden

1) ... oder so:

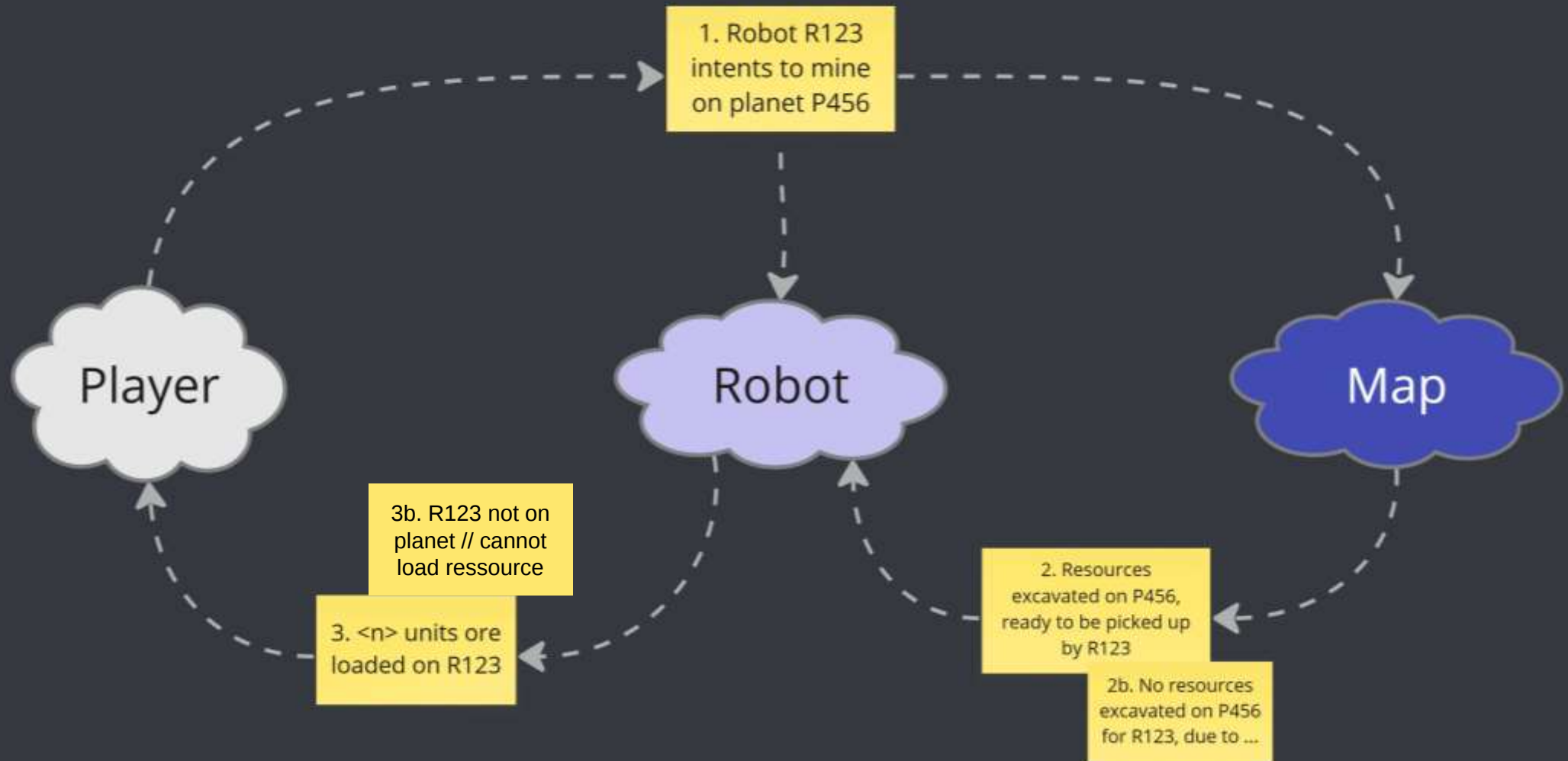


In der Industrie: SAGA vermeiden

2) Pragmatische Fachprozesse

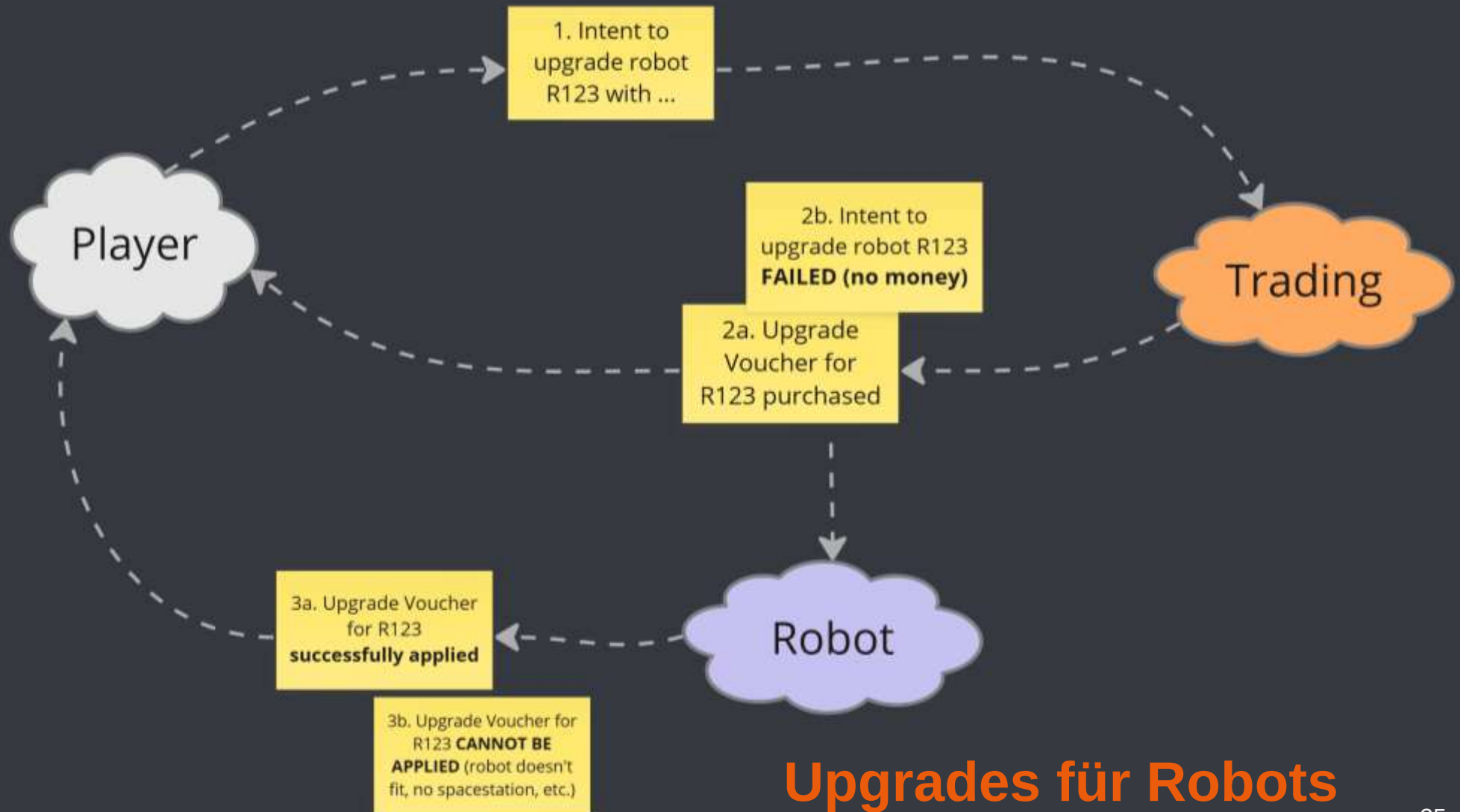


MSD 2.0: Weg 2) Pragmatismus ...

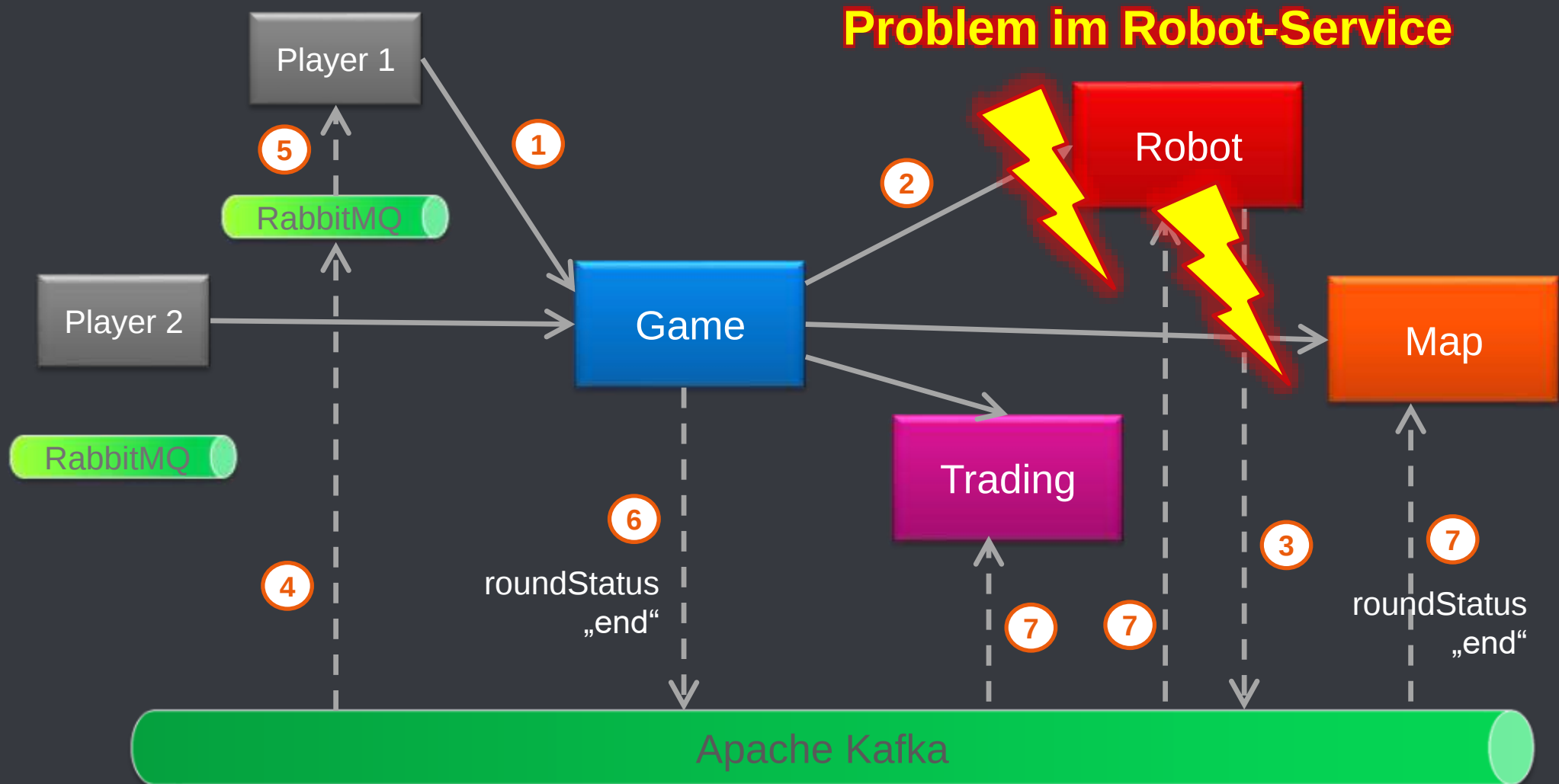


Mining von Ressourcen

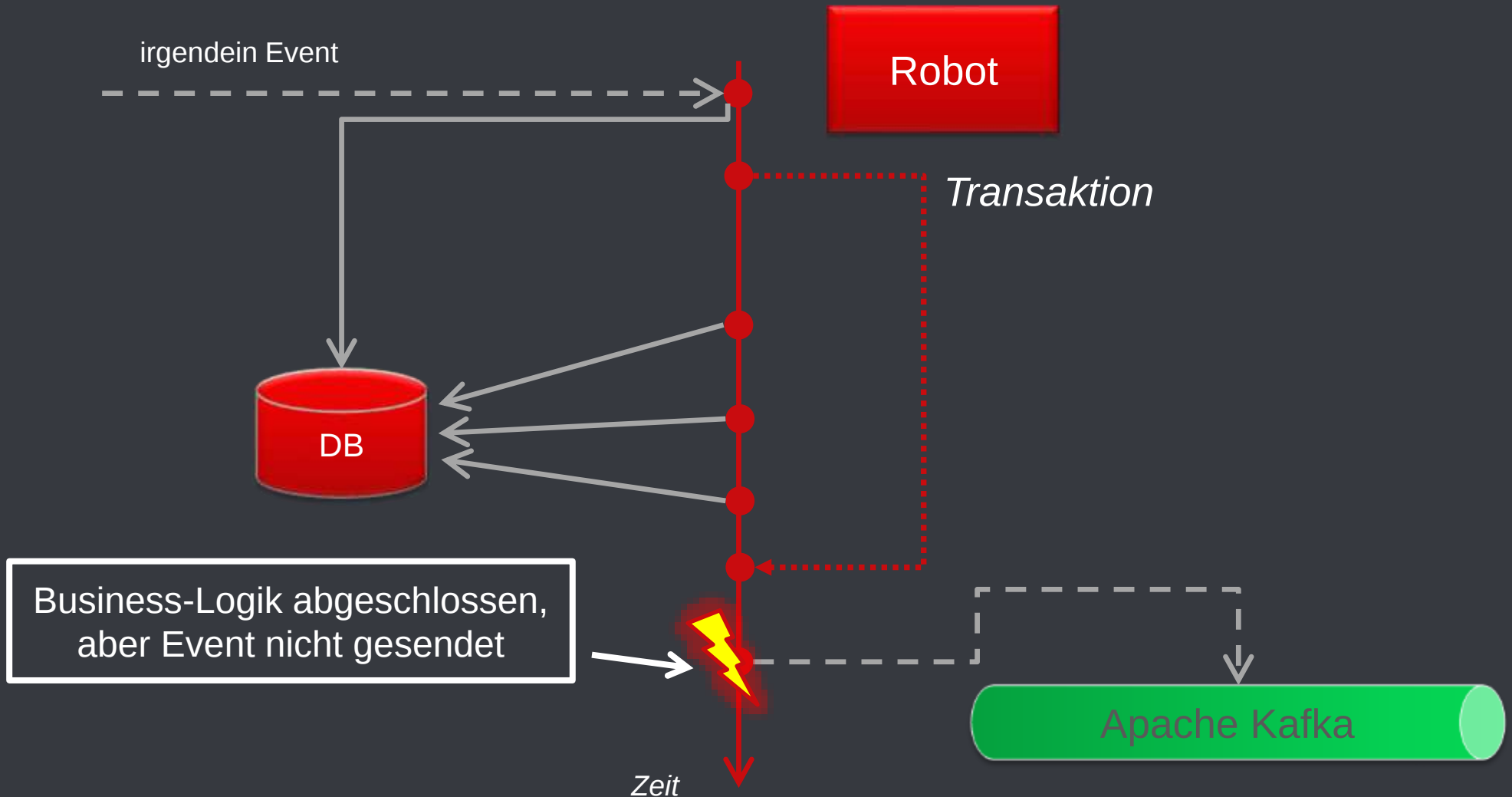
... und Anpassung der Fachlichkeit



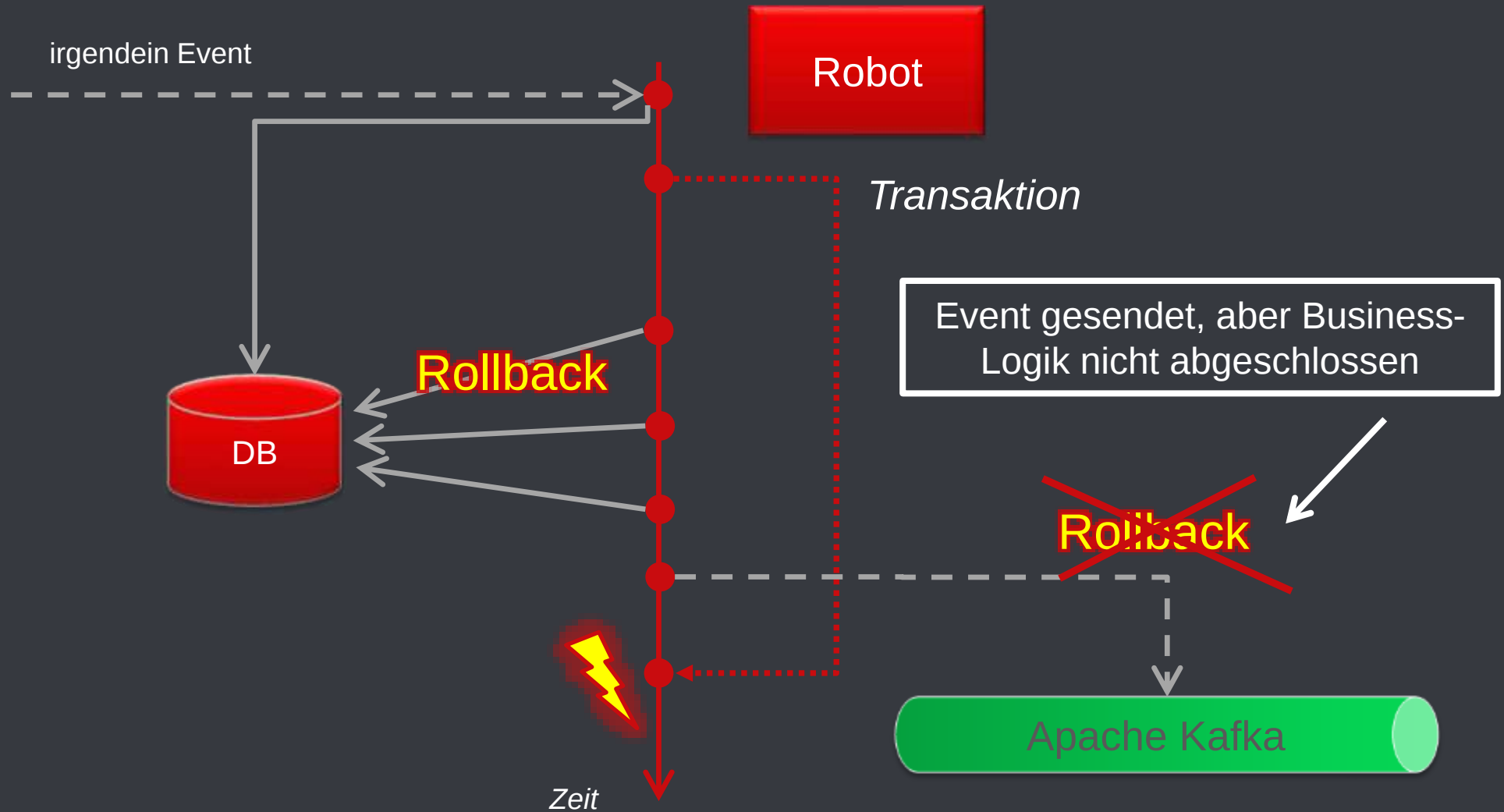
6) „Wird schon alles gut gehen“



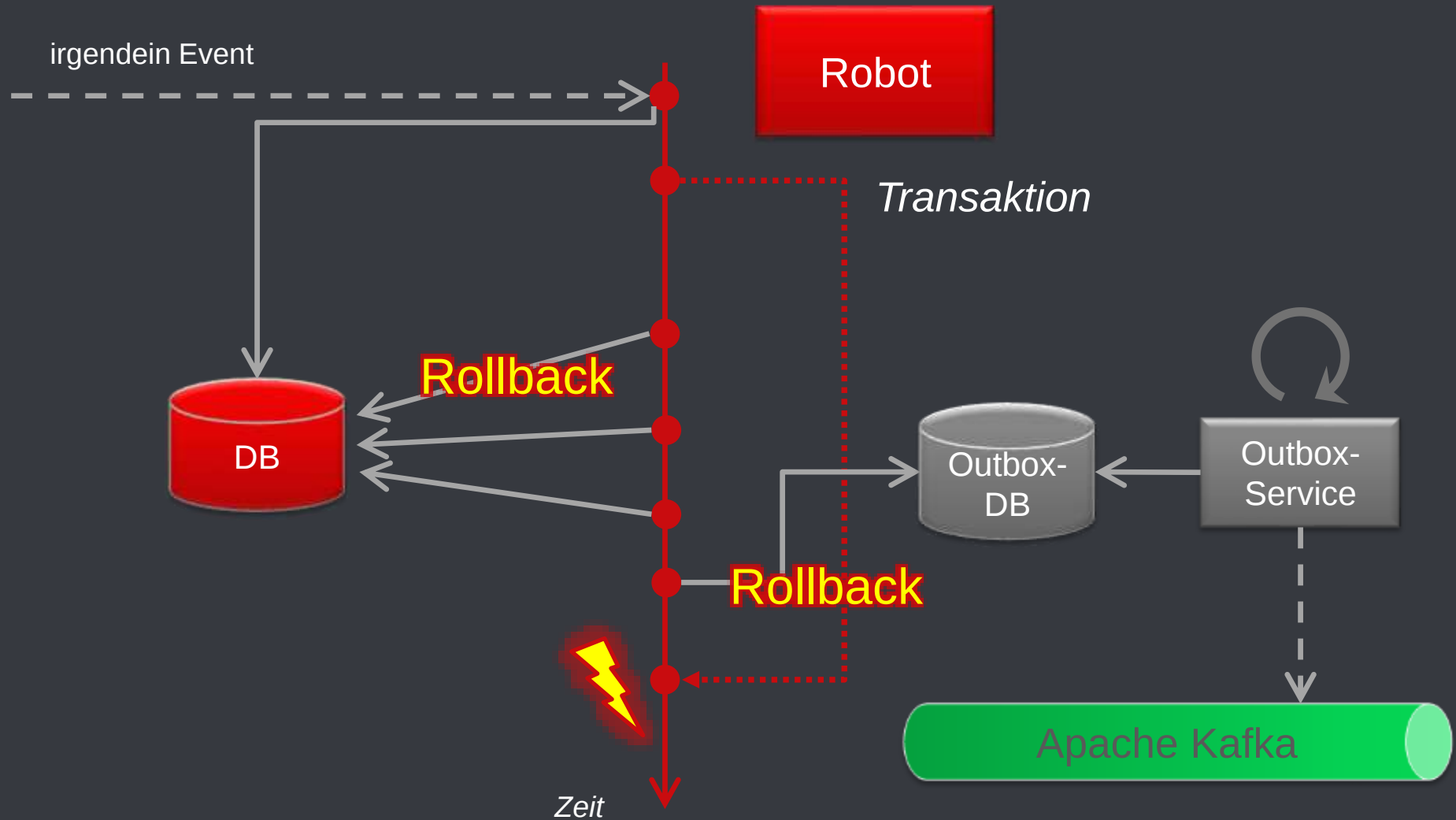
Kein Event versendet



Event voreilig gesendet



LÖSUNG: Outbox-Pattern



Fazit und Ausblick

1) Player-Services: (fast) gleichberechtigt

	Core Services	Player Services
Deployed in Kubernetes-Cluster	✓	✓
Überwacht von Infrastruktur-Monitoring	✓	✗
Senden Kommandos / Queries via REST	✗	✗
Nehmen Kommandos / Queries via REST entgegen	(✓) nur von eigenen UIs	(✓) nur von eigenen UIs
Konsumieren Events von Kafka-Topic	✓	✓
Produzieren Events für Kafka-Topic	✓	(✓) ohne Topic-Ownership

Design-Fehler 1) ... 6)

	Version 2.0
Player-Services sind Bürger 2. Klasse	(✓)
Fehlende Governance	✓
Nicht genug über Domäne nachgedacht	✓
Getaktete Event-Verarbeitung	✓
Mix aus synchron & asynchron	✓
„Wird schon alles gut gehen“	✓

Ausblick

- Implementierung läuft gerade
- Fertigstellung bis Anfang 2026 geplant
- Einsatz in ...
 - BA WASP „Microservices und Event-getriebene Architektur“
 - Innofaktur-Events
- **We'll keep you posted 😊**



www.archi-lab.io

© 2025 Prof. Dr. Stefan Bente