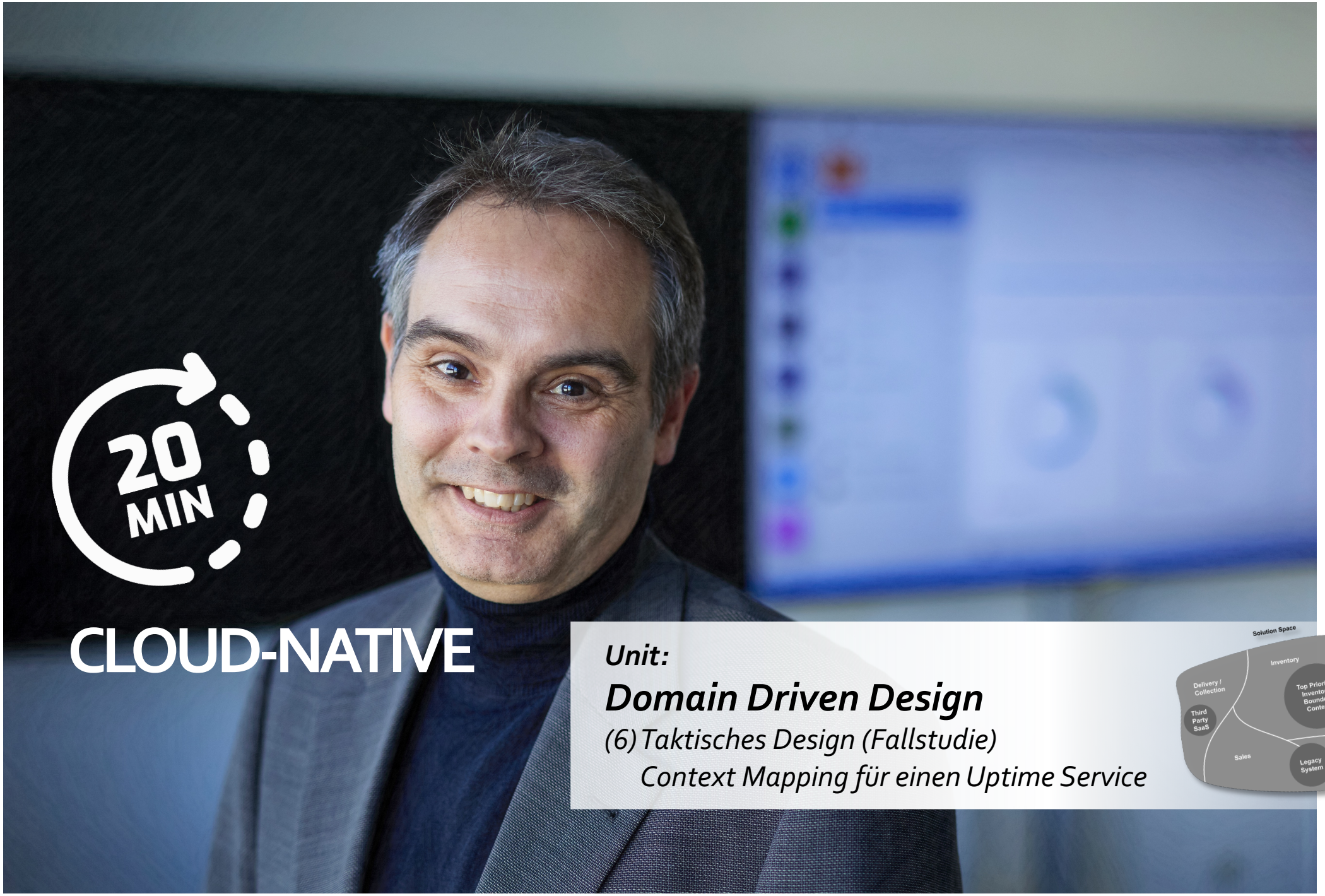



CLOUD NATIVE COMPUTING

# CLOUD-NATIVE

*Unit:*

## *Domain Driven Design*

*(6) Taktisches Design (Fallstudie)*

*Context Mapping für einen Uptime Service*

creative commons

CLOUD NATIVE COMPUTING

*Urheberrechtshinweise*

Diese Folien werden zum Zwecke einer praktikablen und pragmatischen Nutzbarkeit im Rahmen der **CCo 1.0 Lizenz** bereitgestellt.


Sie dürfen die Inhalte also kopieren, verändern, verbreiten, mit eigenen Inhalten mixen, auch zu kommerziellen Zwecken, und ohne um weitere Erlaubnis bitten zu müssen.

Eine Nennung des Autors ist nicht erforderlich (aber natürlich gern gesehen, wenn problemlos möglich).


Diese Folien sind insb. für die Lehre an Hochschulen konzipiert und machen daher vom **§51 UrhG (Zitate)** Gebrauch.

Die CCo Lizenz überträgt sich nicht auf zitierte Quellen. Hier sind bei der Nutzung natürlich die Bedingungen der entsprechenden Quellen zu beachten.

Die Quellenangaben finden sich auf den entsprechenden Folien.

# KAPITEL 14

*Domain Driven Design*


## 14.1 Fachlichkeit, Fachlichkeit, Fachlichkeit

## 14.2 Strategisches Design

- Subdomänen
- Ubiquitous Language
- Bounded Context
- Context Mapping

## 14.3 Taktisches Design

- Oft genutzte Pattern für Geschäftslogik
  (ETL, Active-Record, Domain Model, Event-Sourcing)
- Oft genutzte Architektur-Pattern
  (Layered Architecture, Ports & Adaptor, CQRS)

## 14.4 Zusammenfassung

# INHALTE

## Domain Driven Design

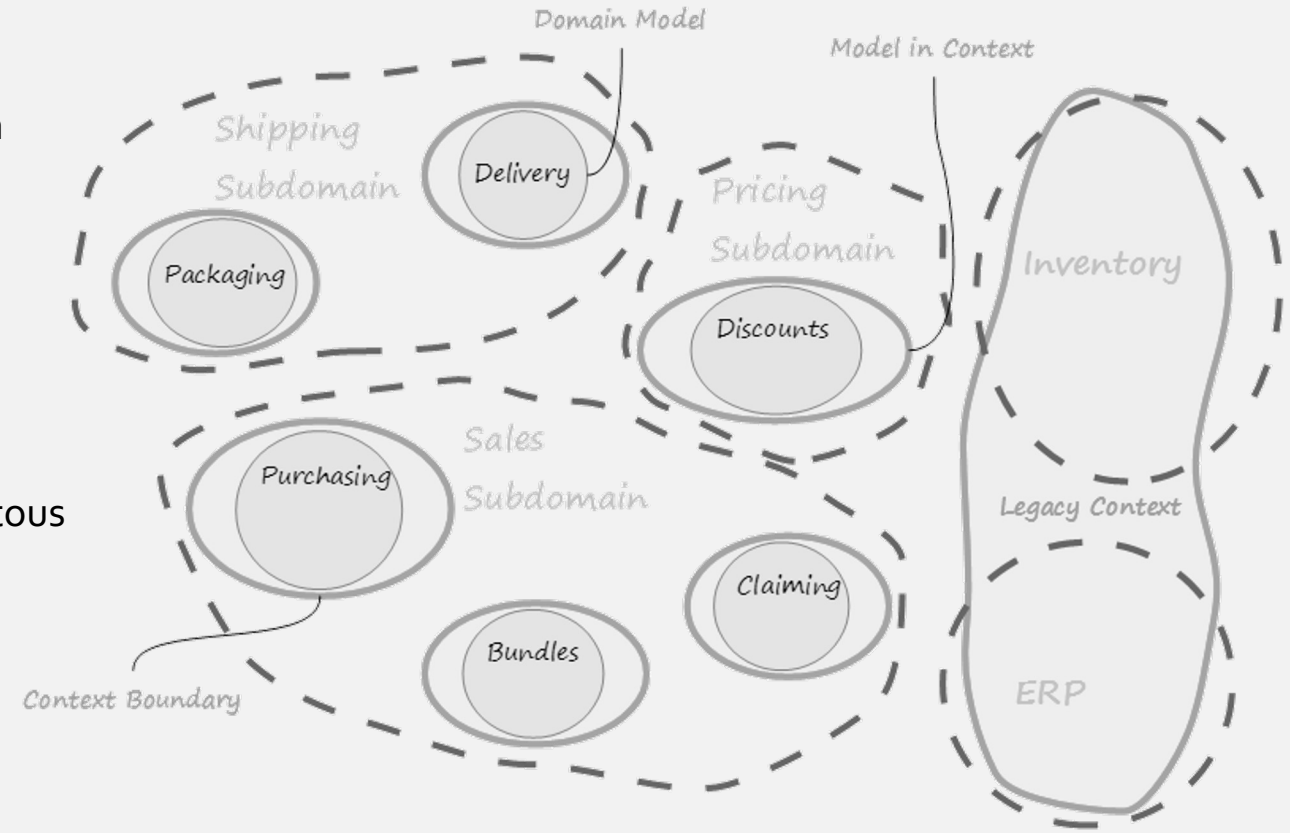
- Was ist das?
- Effektives Software Design
- Strategisches Design
- Taktisches Design

## Strategisches Design

- Subdomains
- Bounded Context + Ubiquitous Language
- Context Mapping

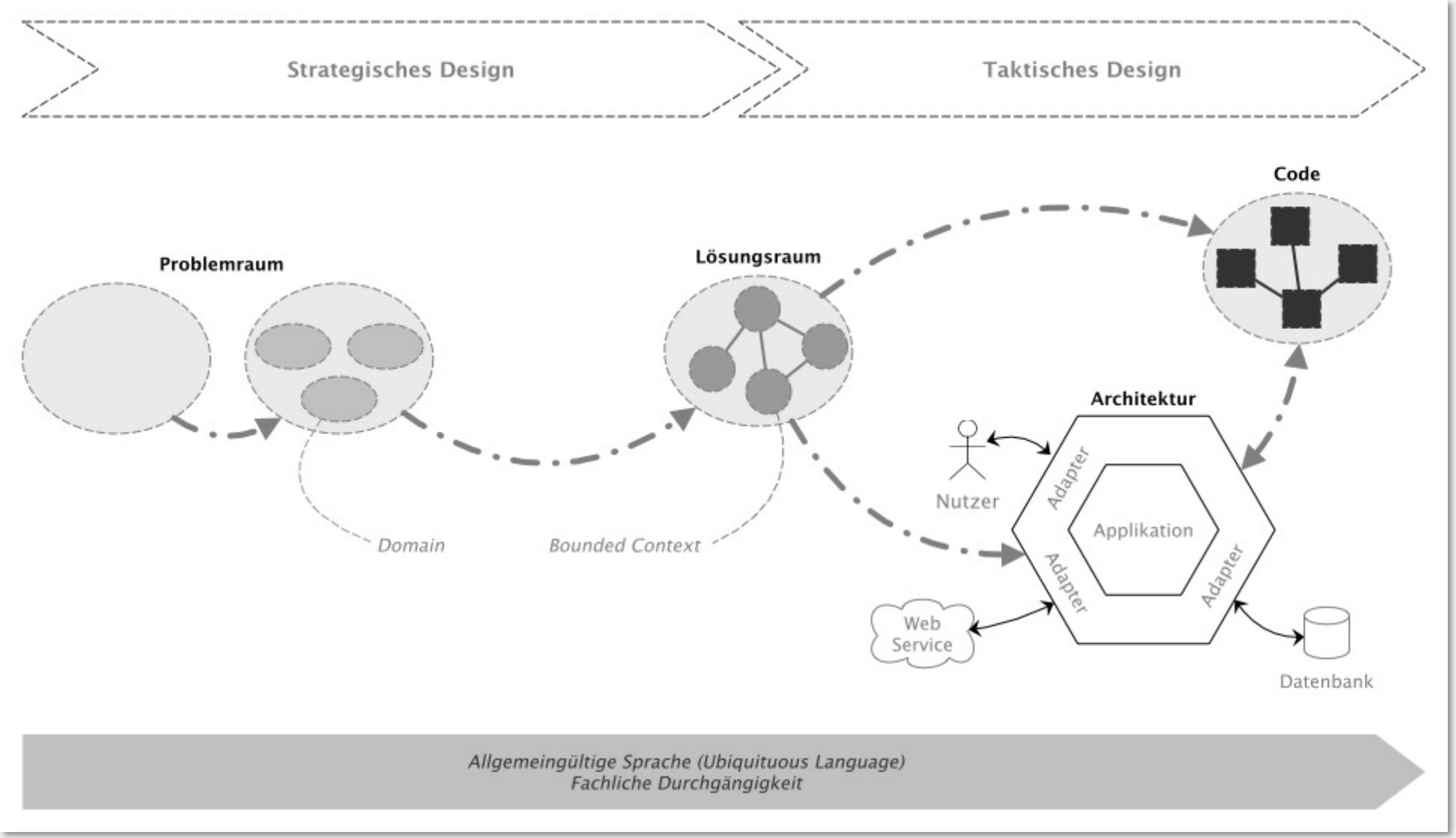
## Taktisches Design

- Business Logic Pattern
- Architectural Pattern
- Fallstudie: Entwurf eines Uptime-Service



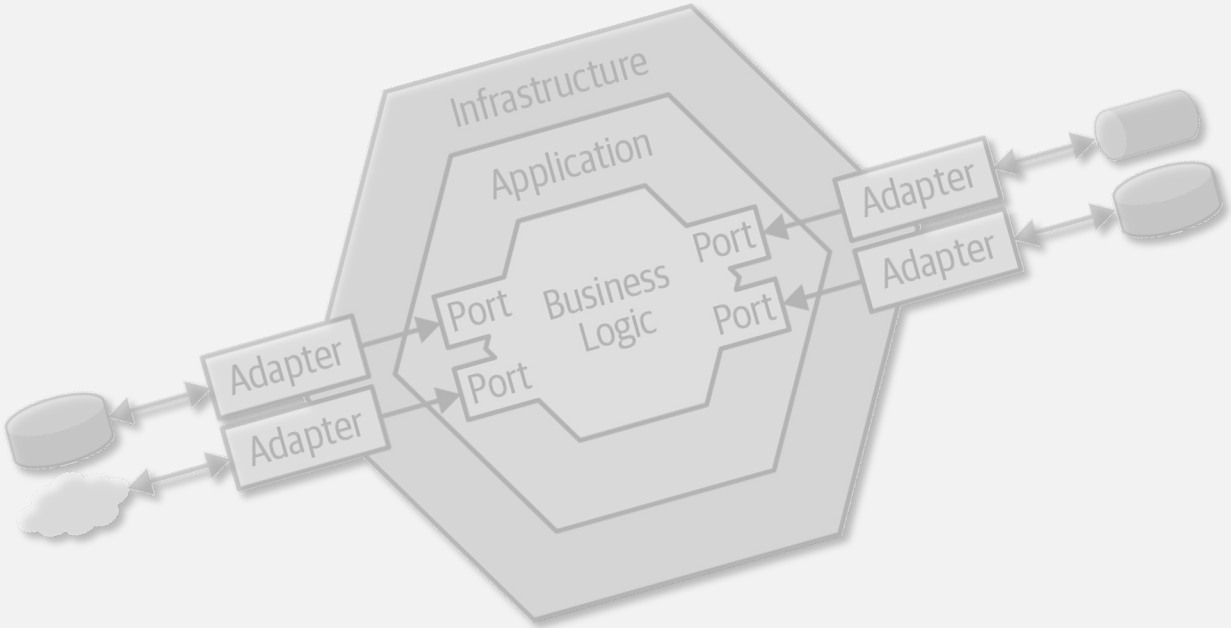
# DOMAIN DRIVEN DESIGN

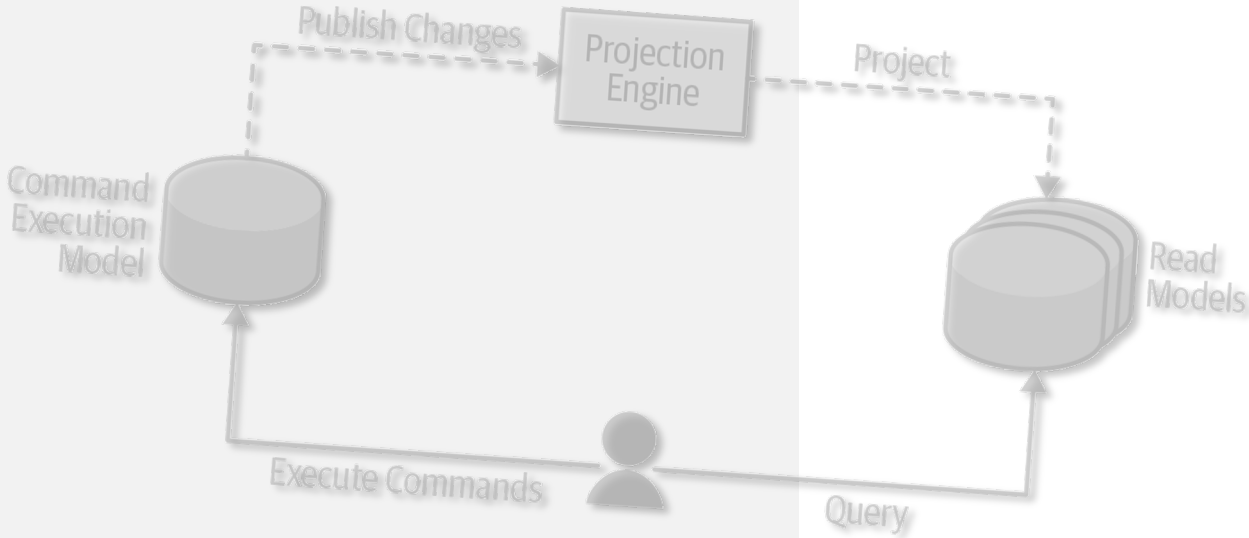
*Fachlichkeit als Treiber der Softwareentwicklung*



# ARCHITEKTUR-PATTERN

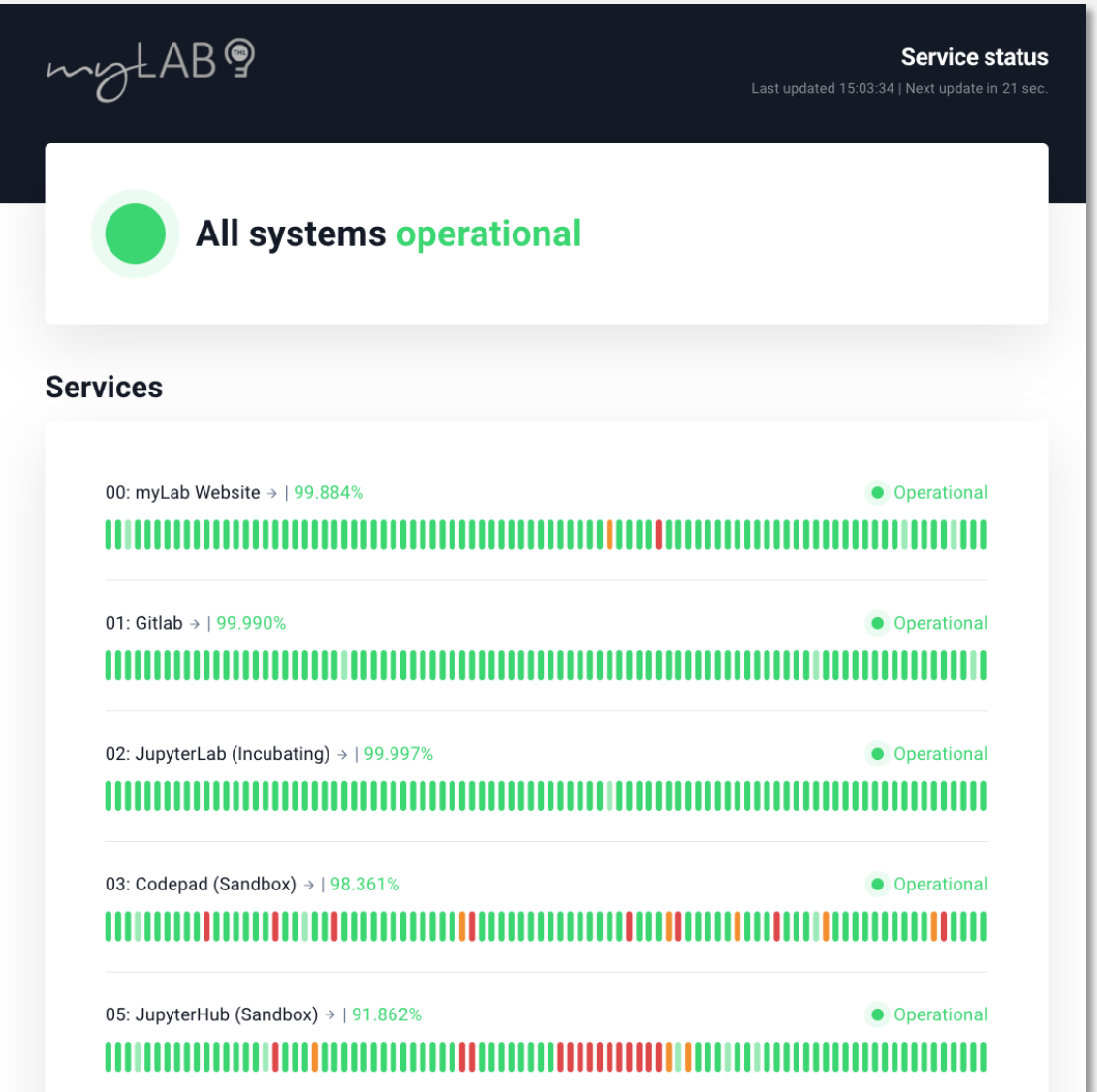
*Am Beispiel eines Use Cases: Entwurf eines Cloud-nativen Uptime Services*

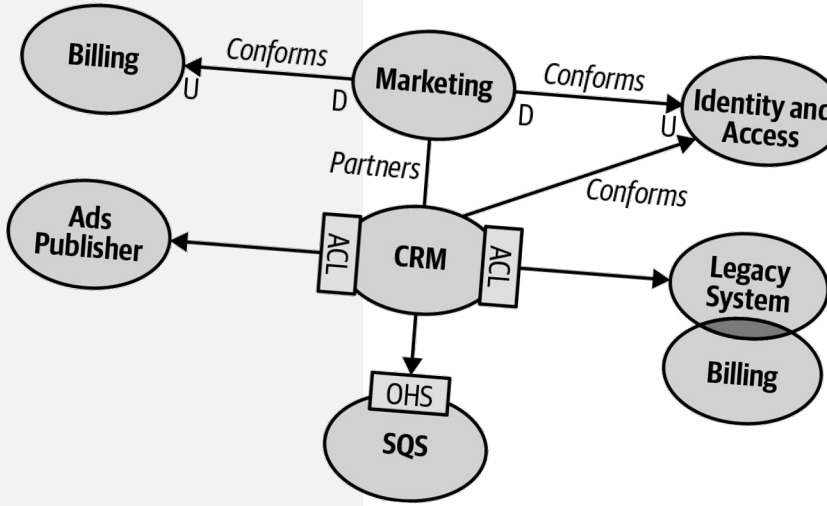



# AUFGABE:

*Entwickeln Sie eine Context-Map für einen Uptime-Service*



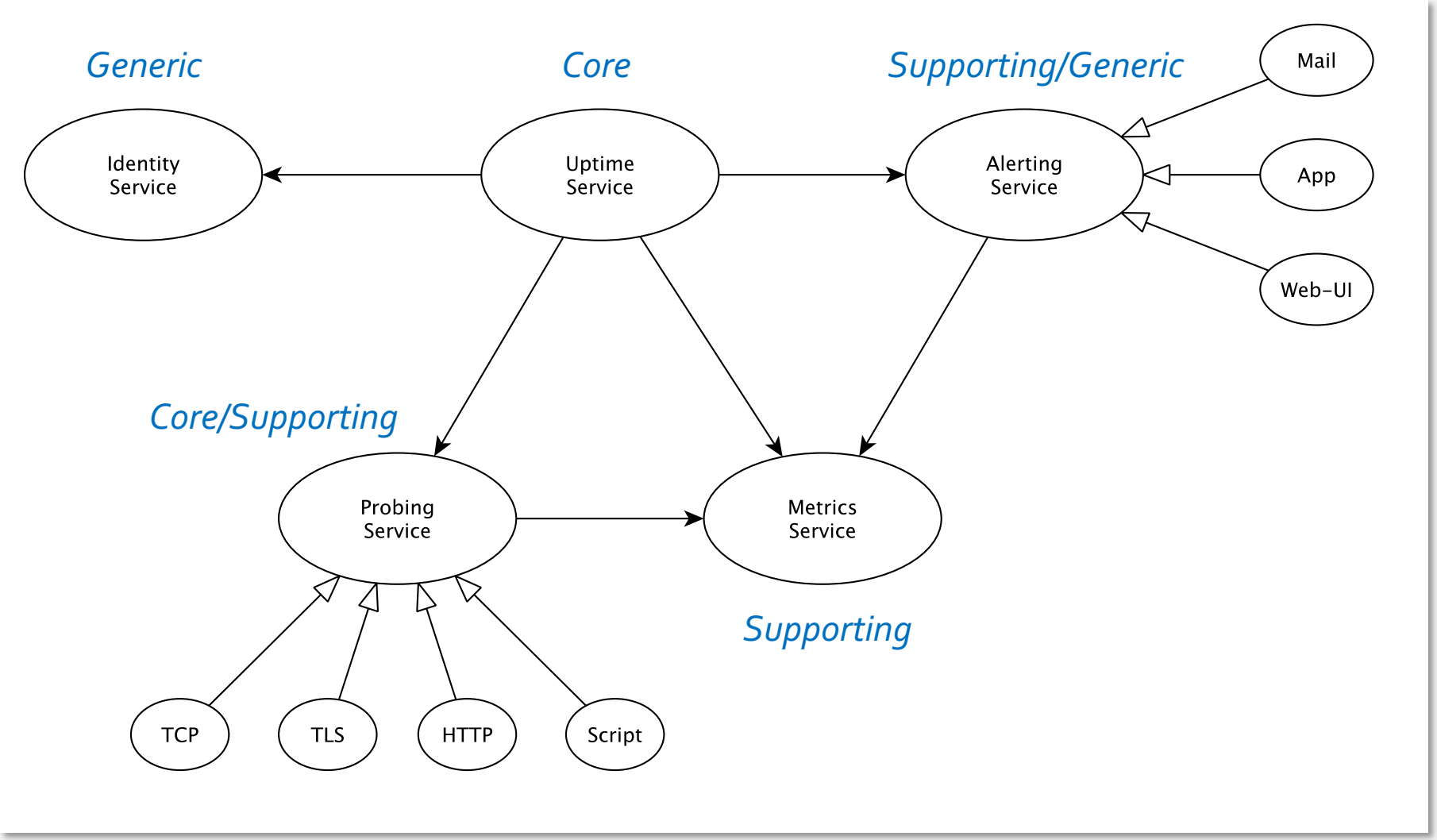

- **Core, Supporting, Generic**
  Subdomains entdecken
- **Bounded Contexts**
  in Subdomains definieren
- **Kooperationspattern**
  - Partner
  - Shared Kernel
- **Customer-Supplier Pattern**
  - Konformist
  - Anti Corruption Layer
    (ACL, auf Downstream-Seite)
  - Open Host Service
    (OHS, auf Upstream-Seite)




**Beispiel:** https://uptime.com

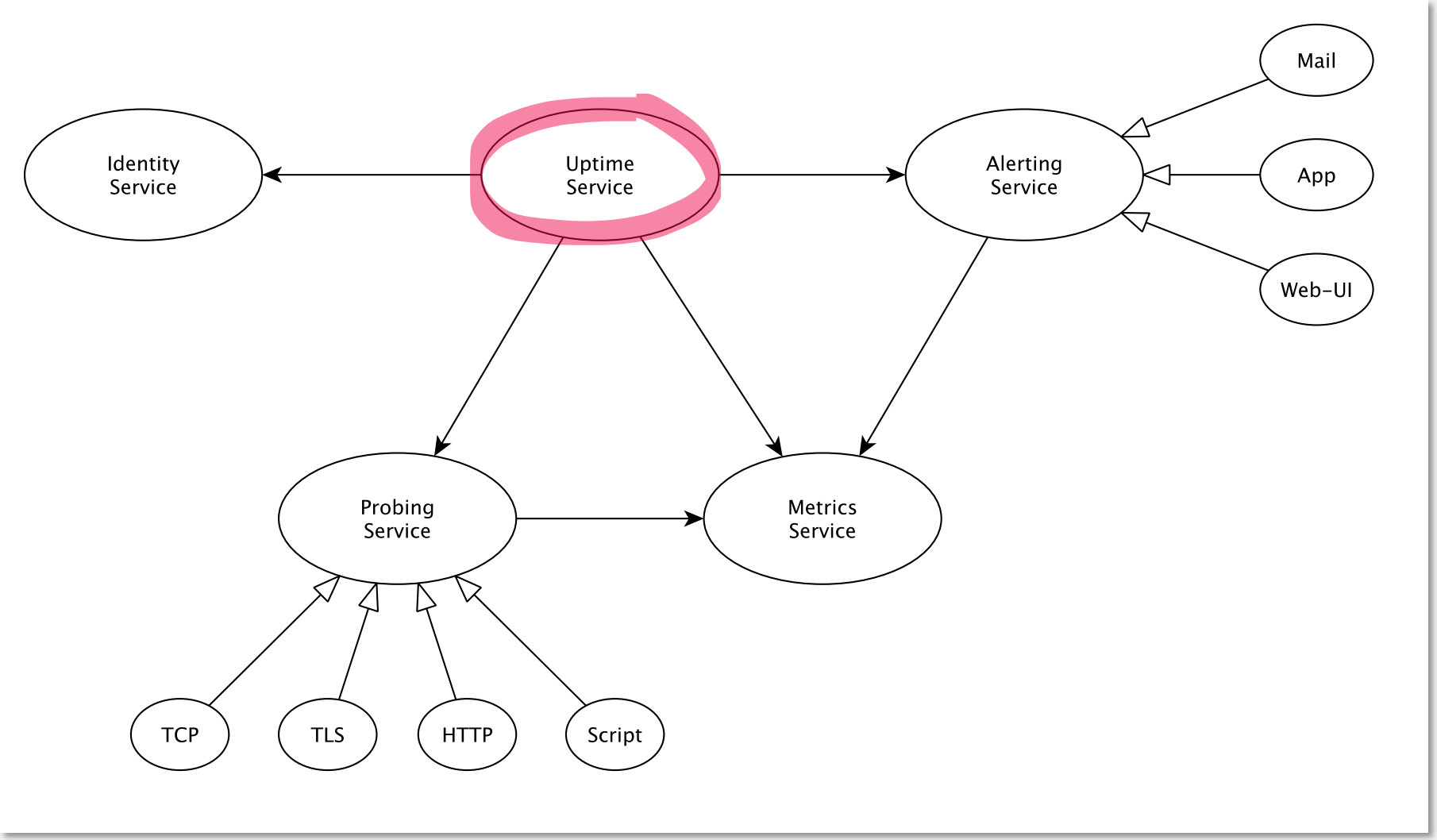
# DESIGN-VORSCHLAG 1:

*Frage: Welcher Service gehört in welche Domäne (Core, Supporting, Generic)?*



# SCHÜTZE SCHWACHE SERVICES

*Frage: Welcher Service ist der schwächste Service?*

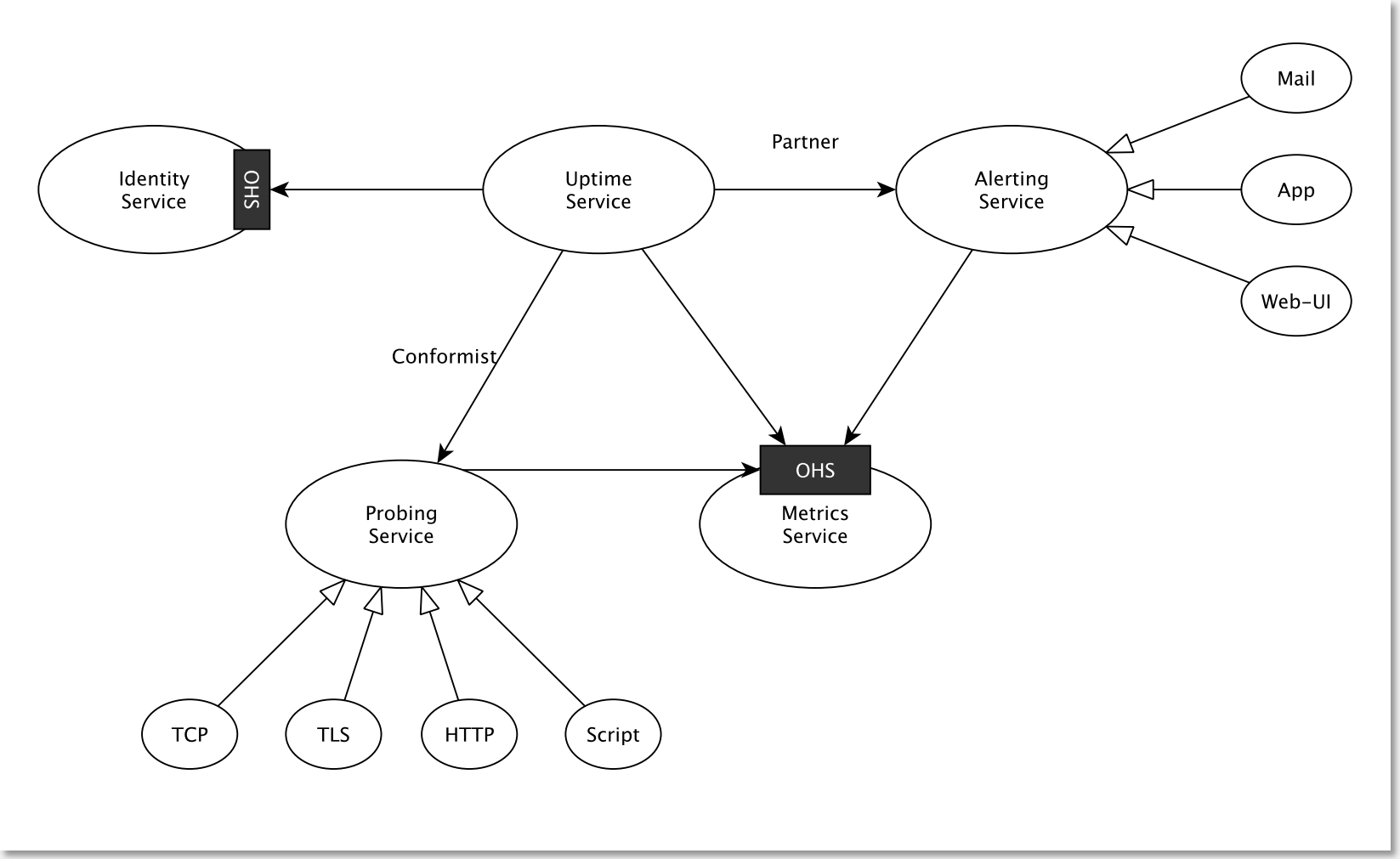



„Schwach" im Sinne von Machtgefälle.

**Hinweis:** Kooperationspfeile zeigen vom Downstream- zum Upstream-Service (D → U)

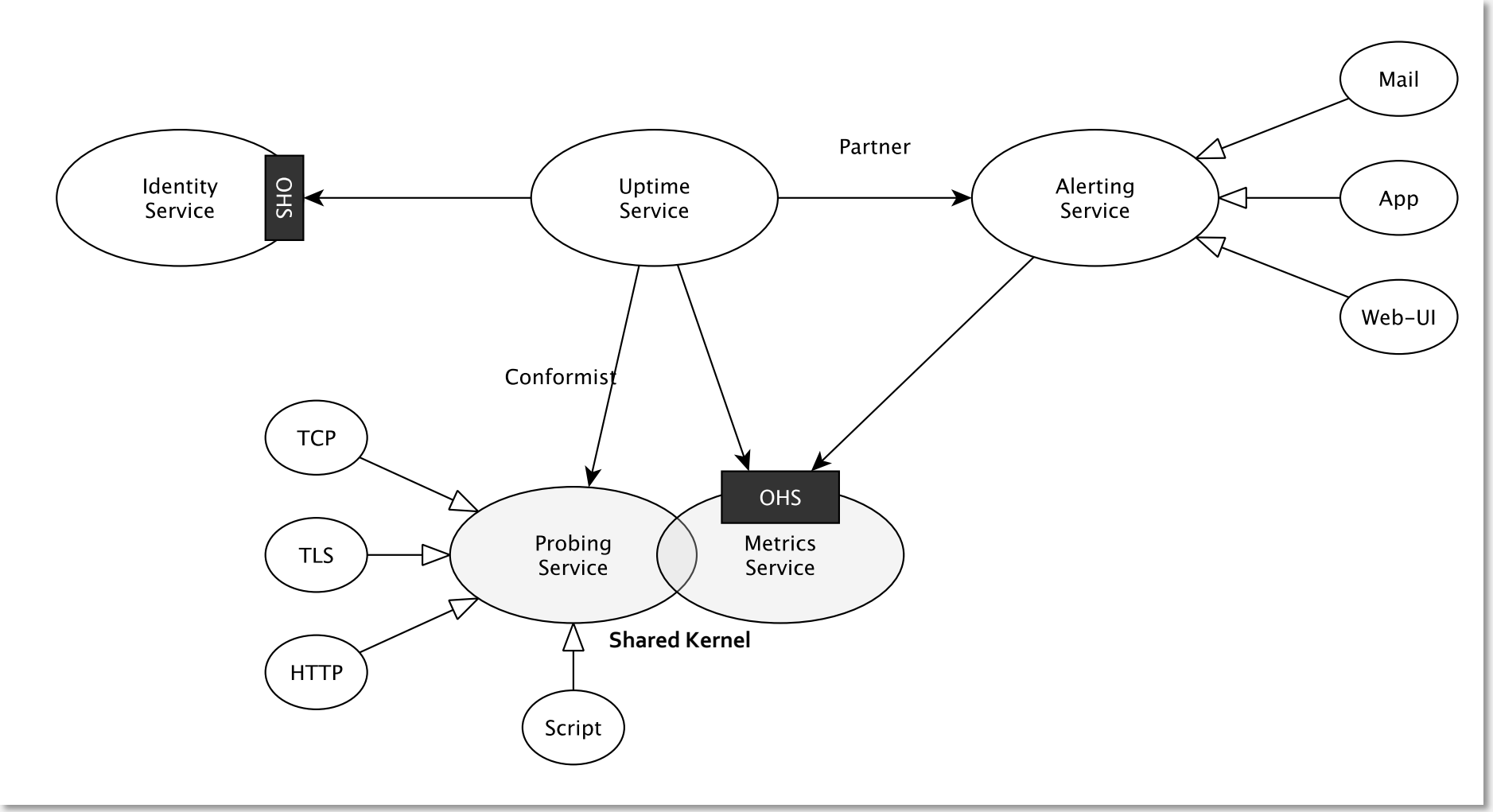
# DESIGN-VORSCHLAG 2:

*Schützen von schwachen Services und klären von Beziehungen verbleibender Abhängigkeiten*



# DESIGN-VORSCHLAG 3:

*Zusammenfassen von eng zusammenhängenden Services (Wer sieht den Nachteil?)*

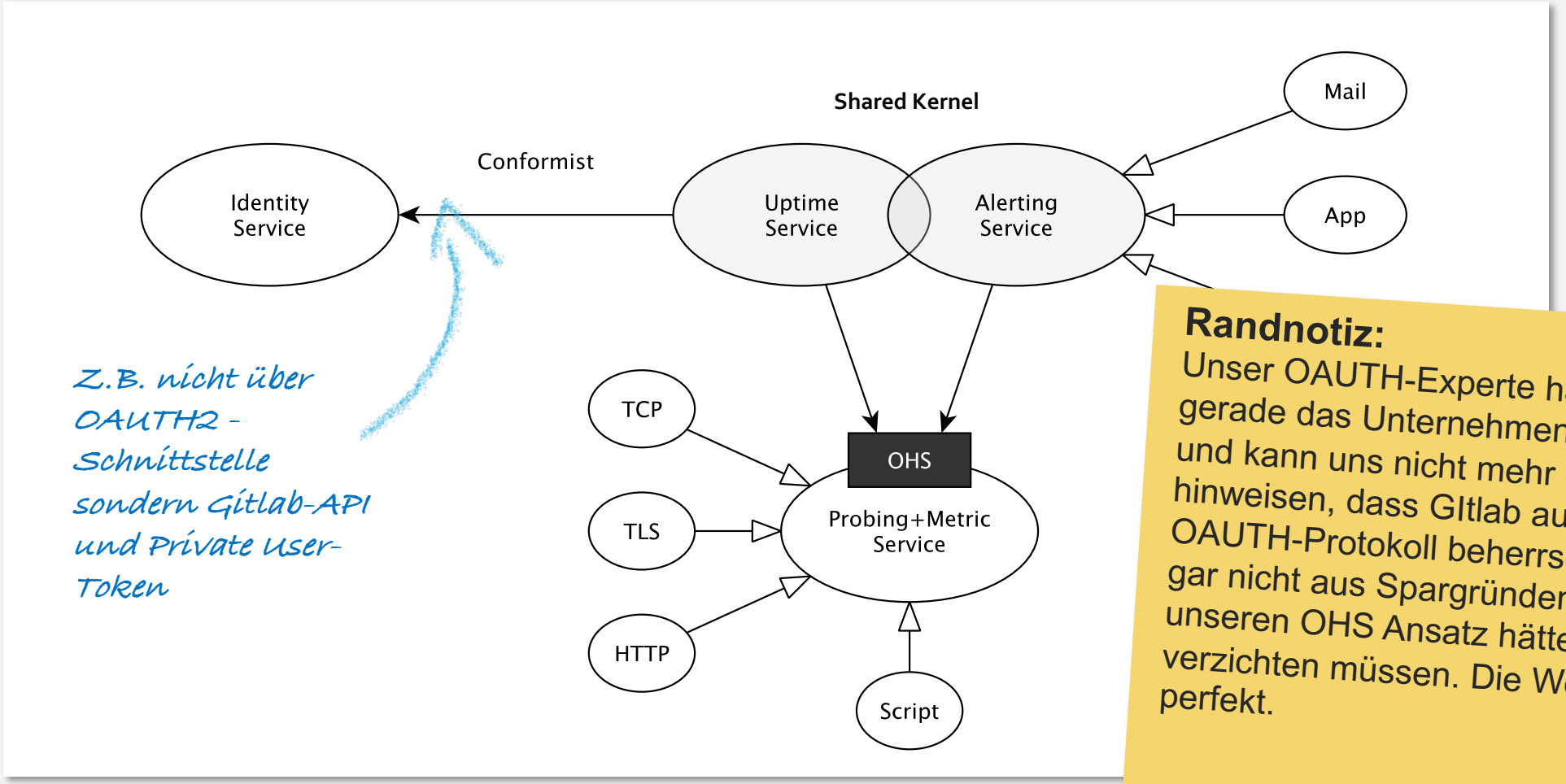



*Nachteil hier: Metrics Service ist Stateful.*

*Und Stateful Services sollten klar isoliert sein.*

# DESIGN-VORSCHLAG 4:

*Andere Anbindung an Identity Service (und Zusammenfassung von Probing + Metrics)*



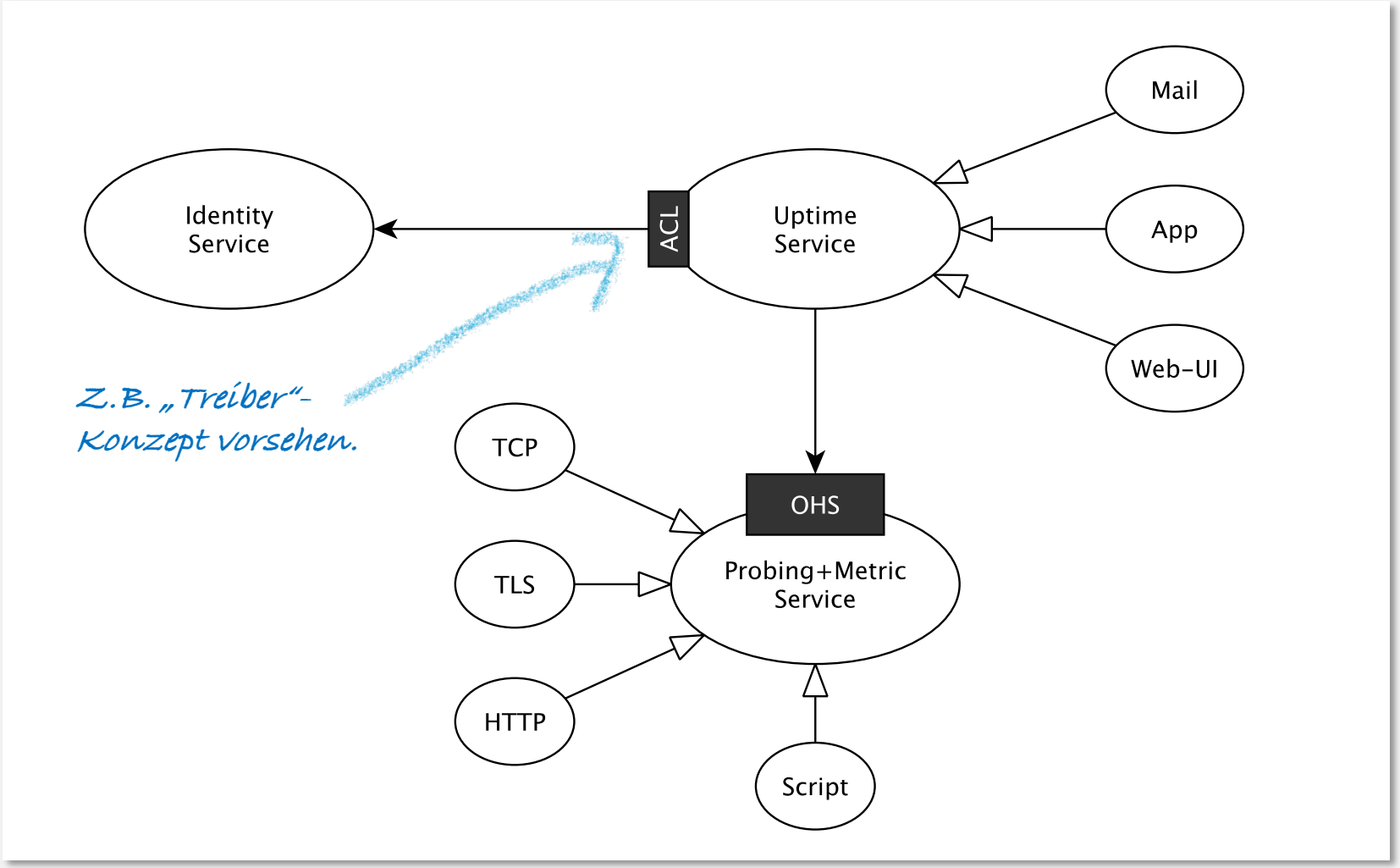

Z.B. nicht über OAUTH2 - Schnittstelle sondern Gitlab-API und Private User-Token

**Randnotiz:**
Unser OAUTH-Experte hat leider gerade das Unternehmen verlassen und kann uns nicht mehr darauf hinweisen, dass GItlab auch das OAUTH-Protokoll beherrscht und wir gar nicht aus Spargründen auf unseren OHS Ansatz hätten verzichten müssen. Die Welt ist nicht perfekt.

Nachteil hier: Wenn Gitlab-API sich ändert, müssen Sie sich anpassen!

# DESIGN-VORSCHLAG 5:

*Identity Service als Legacy Service handhaben*
*(und weitere Zusammenfassung von Uptime + Alerting)*

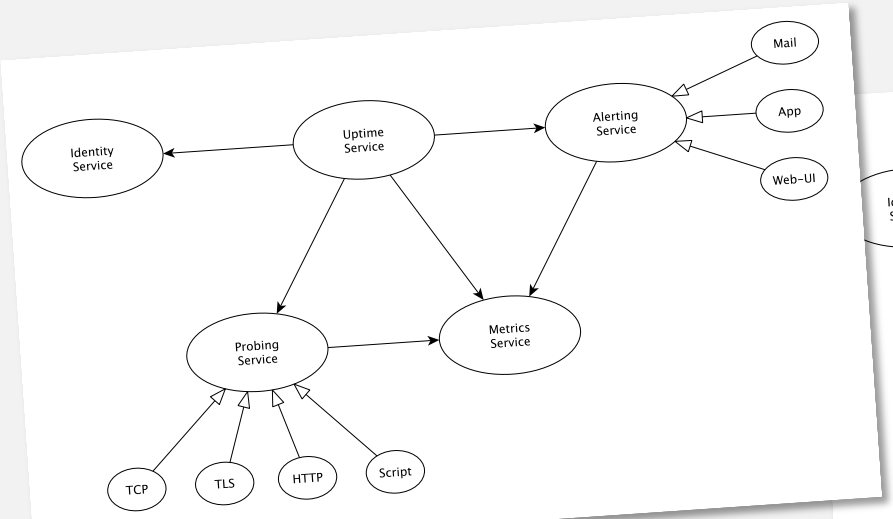


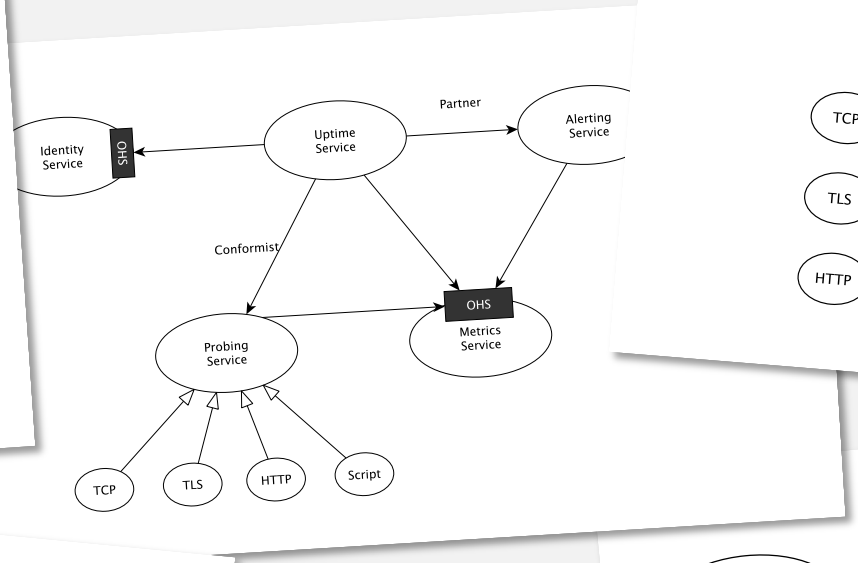

Nachteil hier: Alerting, Probing, und Metric sind jetzt nicht mehr gut individuell skalierbar.

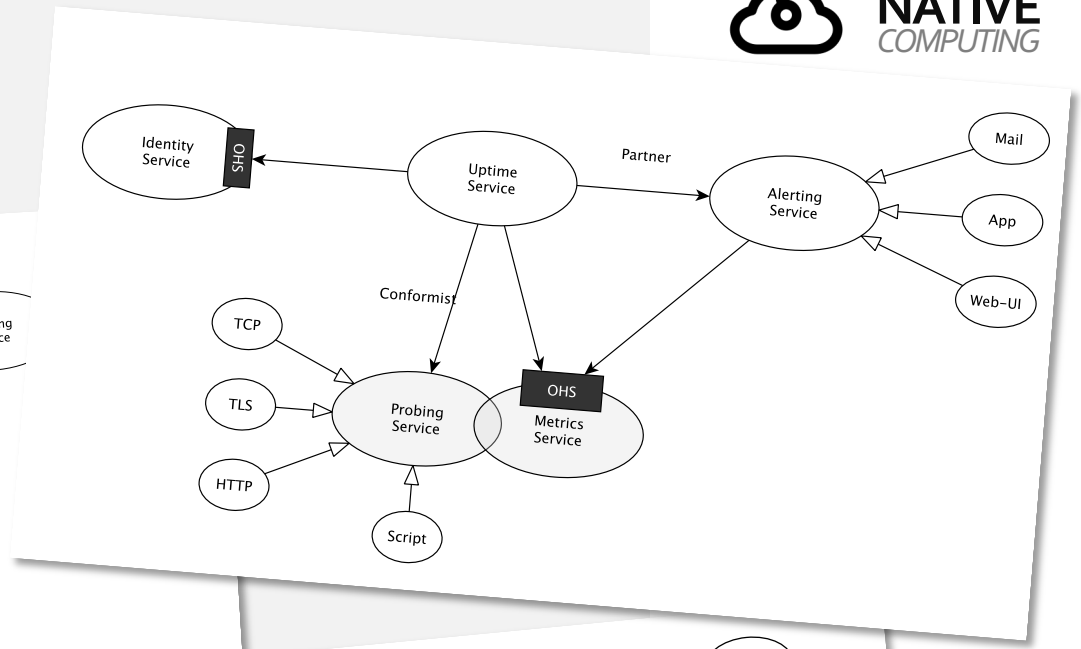
Rechnen Sie daher mit Skalierbarkeits-problemen, wenn der Service intensiv genutzt werden sollte.

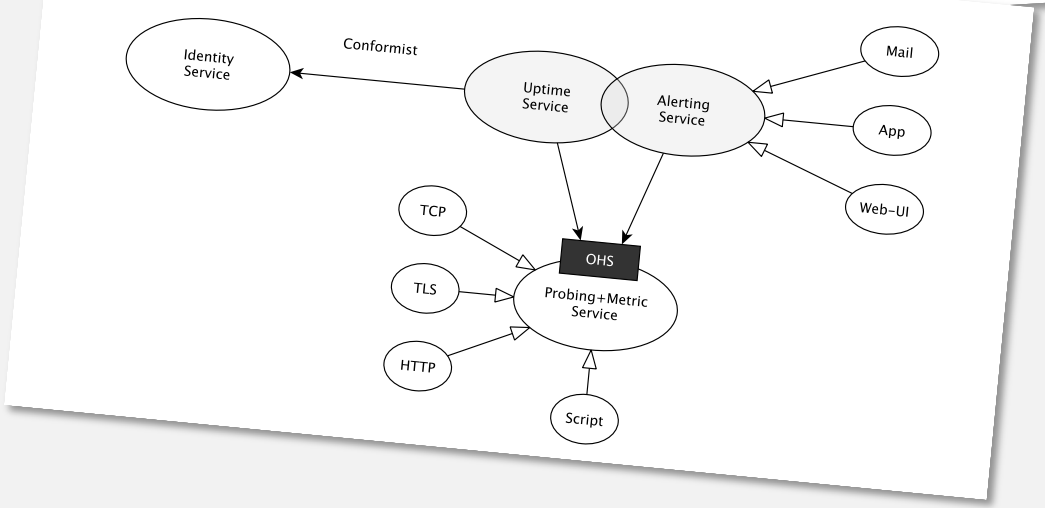
# OK

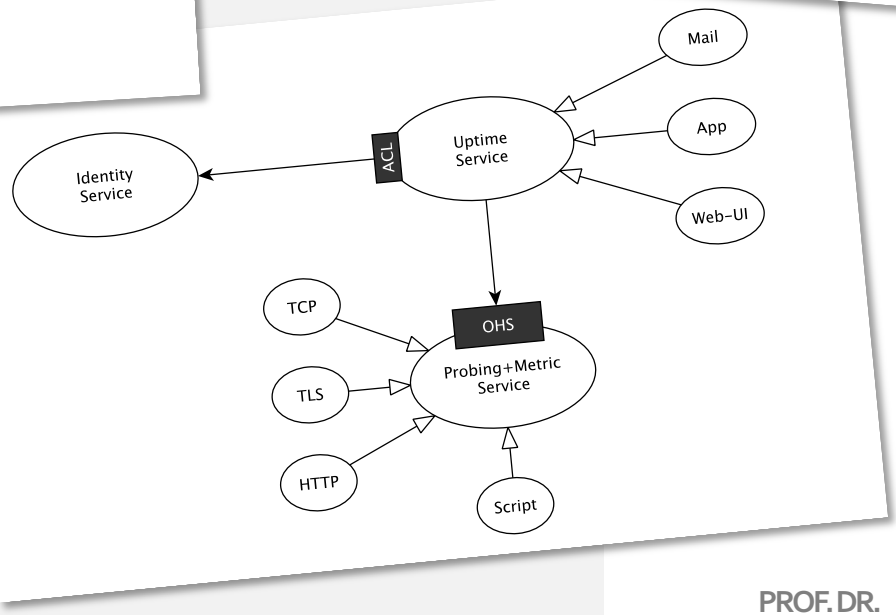
*Es gibt wohl mehr als eine Lösung …*



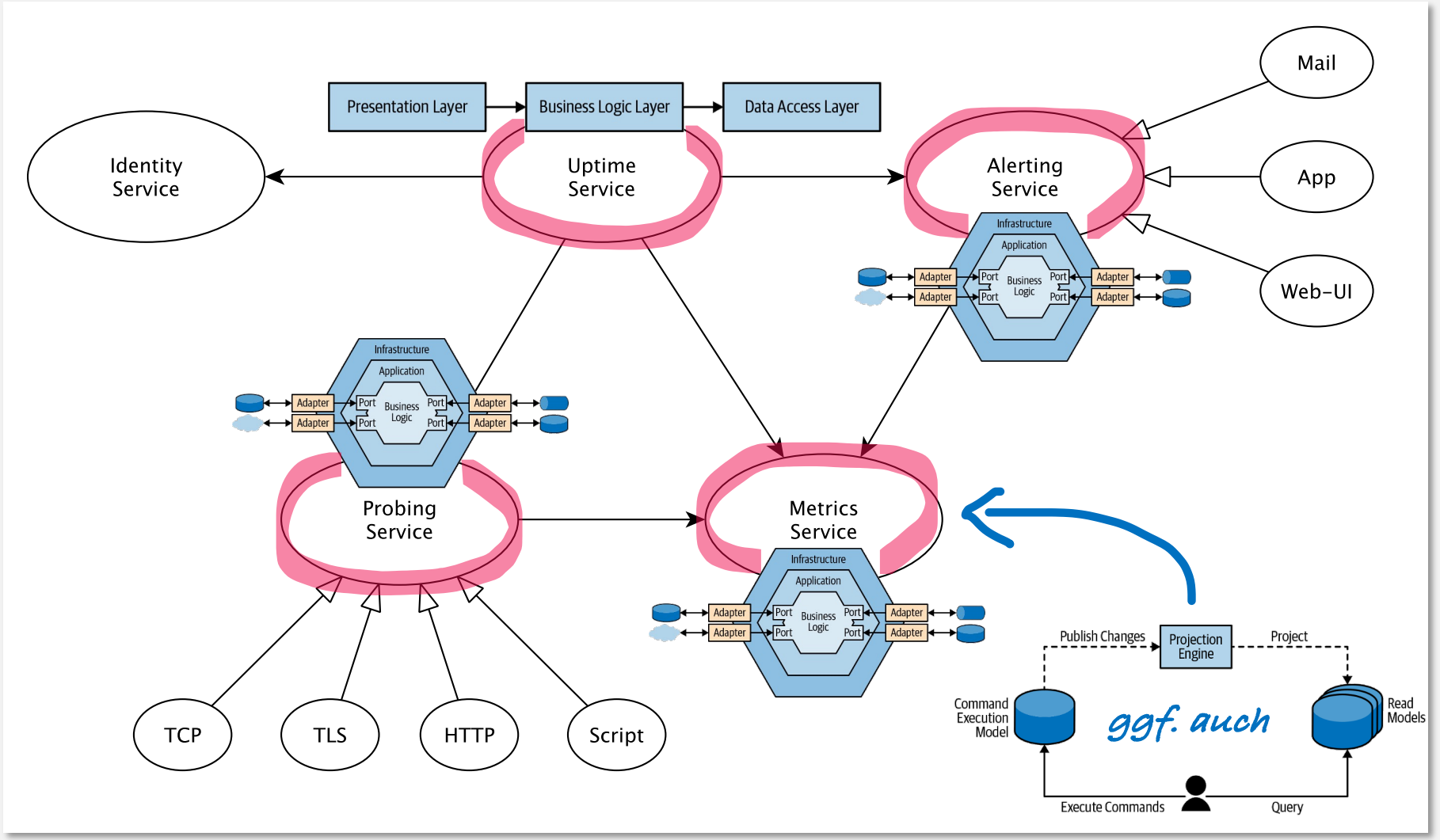



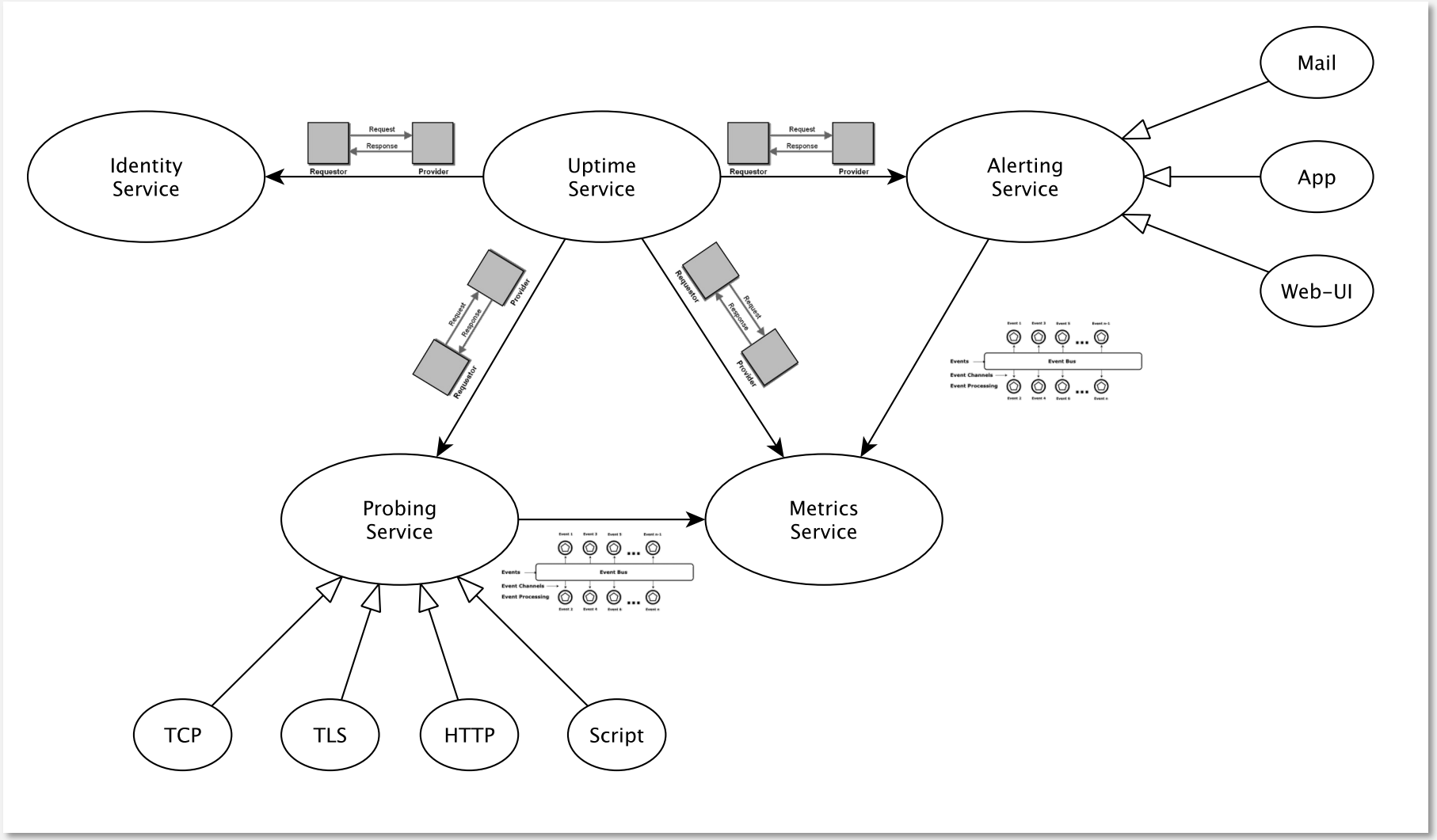
# TACTICAL DESIGN

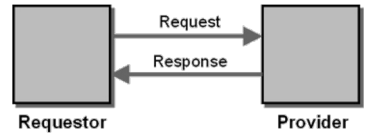
*Architecture Pattern (Wo anwenden?)*

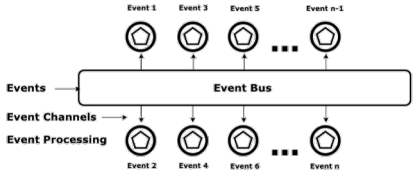


# TACTICAL DESIGN

*Abschlussfrage (Wo Req/Resp, wo Messaging?)*



# KONTAKT

*Disclaimer*

**Nane Kratzke**

+49 451 300-5549

nane.kratzke@th-luebeck.de

kratzke.mylab.th-luebeck.de