




Zwischen innovativer Freiheit und Zufallsarchitektur

*Die Gratwanderung einer
modernen (Fach-)Architektur*

Stefan Bente

Technology
Arts Sciences
TH Köln

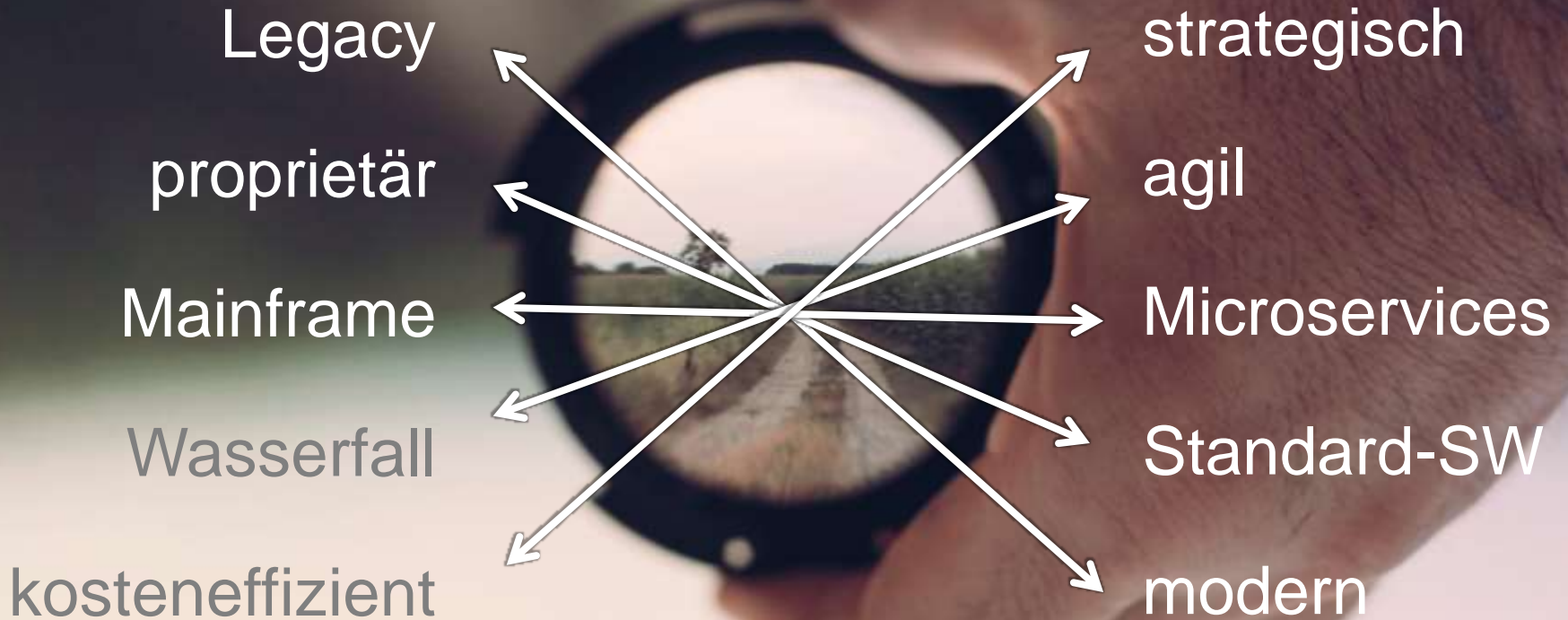
- 
- Architektur noch nötig?
 - Unternehmensarchitektur noch nötig?
 - Facharchitektur noch nötig?

Drei Thesen zur aktuellen Situation der (Fach-) Architektur

Versuch der Einordnung der Versicherungs-IT

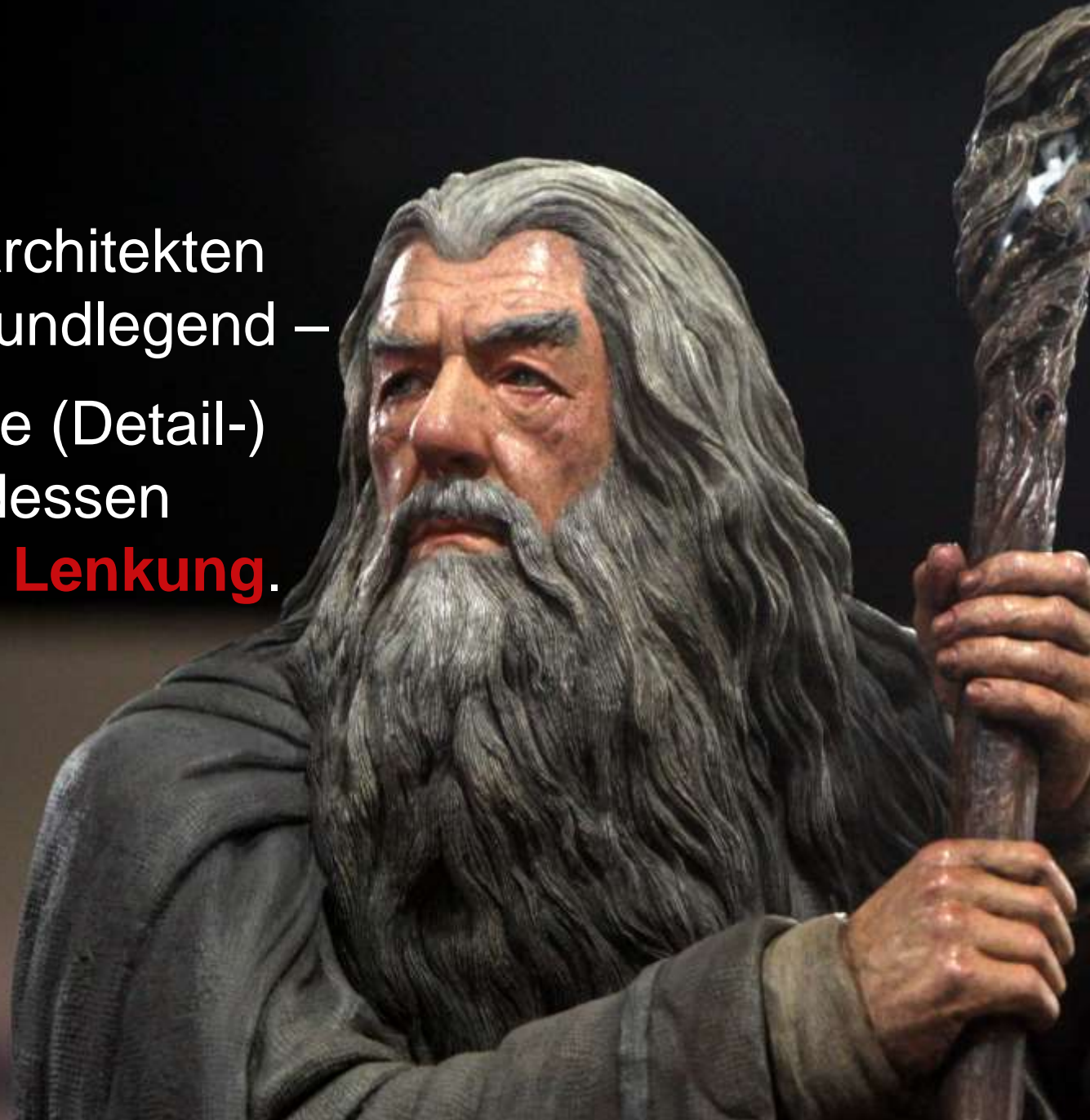
	eCommerce, Internet	Versicherung	Öffentliche Verwaltung
Alter	<ul style="list-style-type: none"> ▪ Junge IT-Landschaft, wenig Legacy 	<ul style="list-style-type: none"> ▪ Mischung aus neuer IT und Legacy 	<ul style="list-style-type: none"> ▪ Viel Legacy
Durchmischung	<ul style="list-style-type: none"> ▪ Proprietäre Lösungen, viel Open Source 	<ul style="list-style-type: none"> ▪ Mischung aus proprietärer IT und Standard-SW 	<ul style="list-style-type: none"> ▪ Nahezu ausschließlich proprietäre IT
Treiber	<ul style="list-style-type: none"> ▪ Expansion statt Kostendruck 	<ul style="list-style-type: none"> ▪ Sowohl Kostendruck wie strategische Weiterentwicklung 	<ul style="list-style-type: none"> ▪ Deutlicher Kostendruck
Entwicklung	<ul style="list-style-type: none"> ▪ Agile Entwicklung dominiert 	<ul style="list-style-type: none"> ▪ Nebeneinander von Agil und Wasserfall 	<ul style="list-style-type: none"> ▪ Wasserfall dominiert

Versicherungs-IT im Fokuspunkt aktueller IT-Herausforderungen



These 1

Die Rolle des Architekten wandelt sich grundlegend – weniger zentrale (Detail-) Planung, statt dessen mehr **indirekte Lenkung**.



- Weg vom Gespenst der schwerfälligen, bürokratischen IT
- Kleine, kreative, unabhängige Organisationseinheiten
 - Bimodale IT
 - agile Methoden
 - ausgegliederte Startup-Inkubatoren



Conway's Law

“Jede Organisation, die [IT-]Systeme entwirft, produziert ein Design, das die Kommunikationsstruktur der Organisation widerspiegelt.” (*)

Microservices als Blaupause der Architektur

- entkoppelte, dezentrale Architektur
- Im hohen Maß unabhängige, autarke Teams
- Kompromisse bei
 - Datenredundanz und -Integrität
 - Konzeptueller Integrität

(*) Quelle: Conway, Melvin. "How Do Committees Invent?" *Datamation*, 1968. http://www.melconway.com/Home/Committees_Paper.html.
Übersetzung durch den Vortragenden. Englische Originalformulierung: "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure."

These 1: Die Rolle des Architekten wandelt sich – weniger zentrale Planung, mehr indirekte Lenkung

Zukünftig muss Architektur anders gemanaged werden!



Leitplanken definieren

statt detailliert Top-Down planen

Wissen vermitteln

statt Spezifikationen vorgeben

In Projekten mitarbeiten

statt (nur) in reinen Architektur-teams diskutieren

“That is, while there is value in the items on the right, we value the items on the left more.” (*)

These 2

Die Software-Entwicklung erlebt eine **Renaissance der Fachlichkeit.**

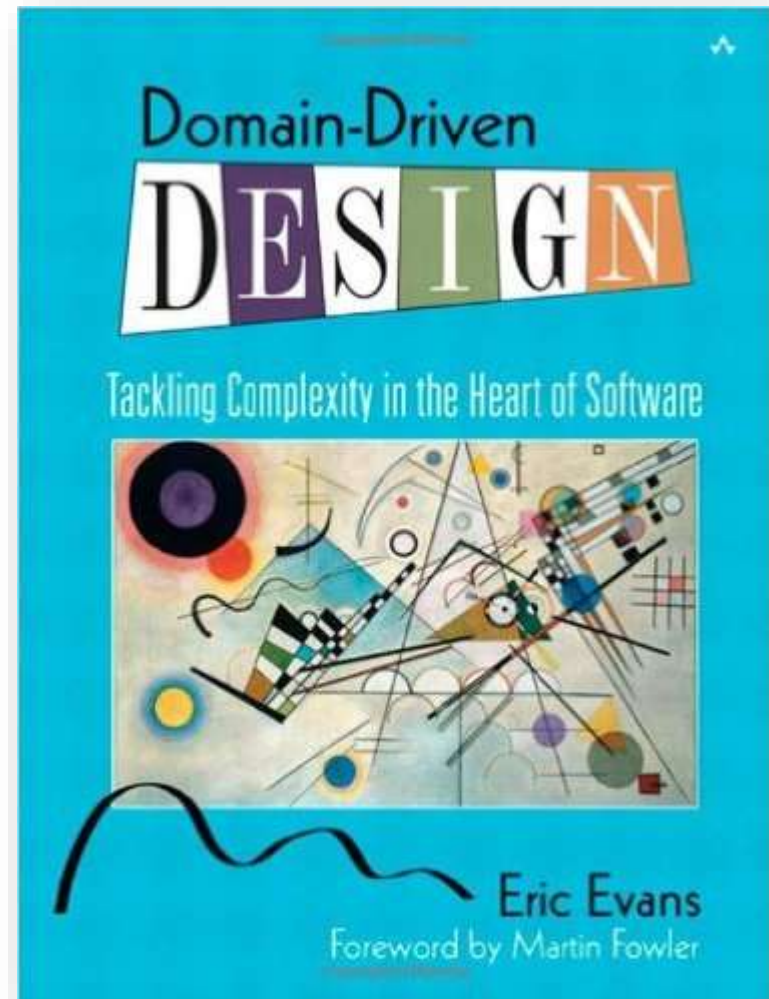
Dies stellt eine große Chance für die Facharchitektur dar – wenn sie **technisch genug** denkt.



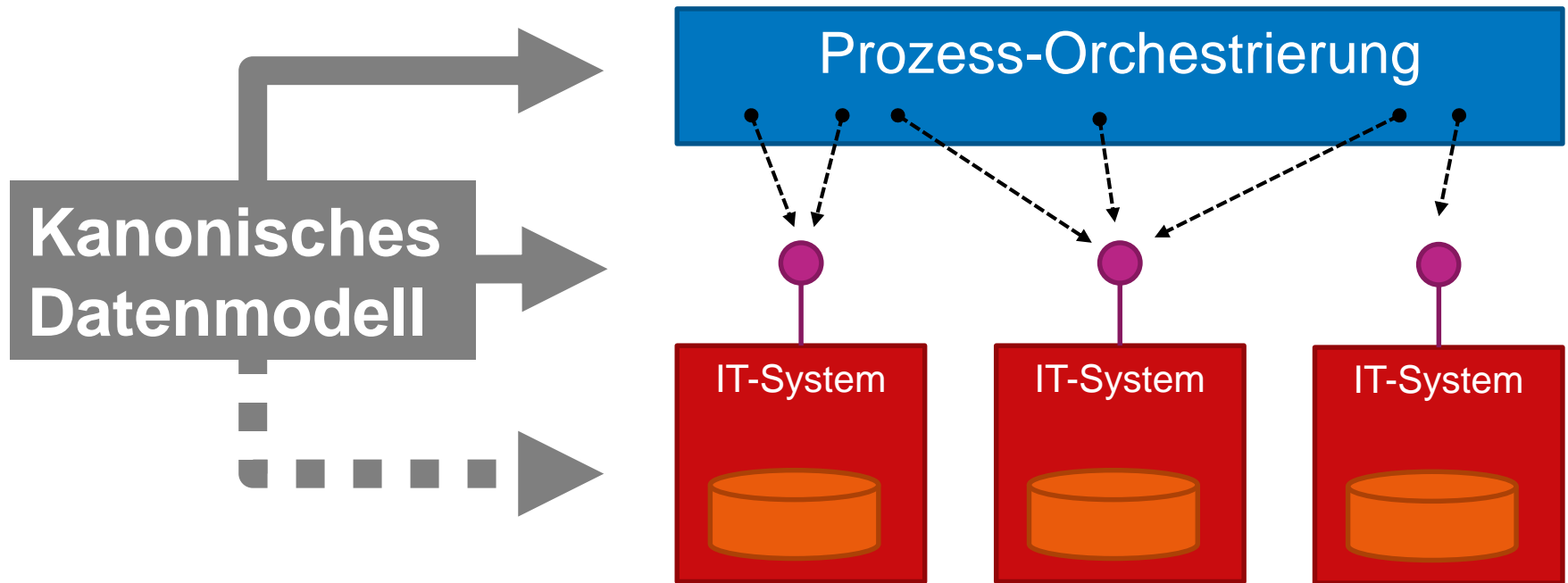
Domain-Driven Design = Rückbesinnung auf Fachlichkeit

Drei wesentliche Konzepte:

1. Domäne
2. Entität
3. Value Object



Klassische SOA: kanonisches Datenmodell bestimmt Prozesse und Datenaustausch



Komplexität des kanonischen Datenmodells

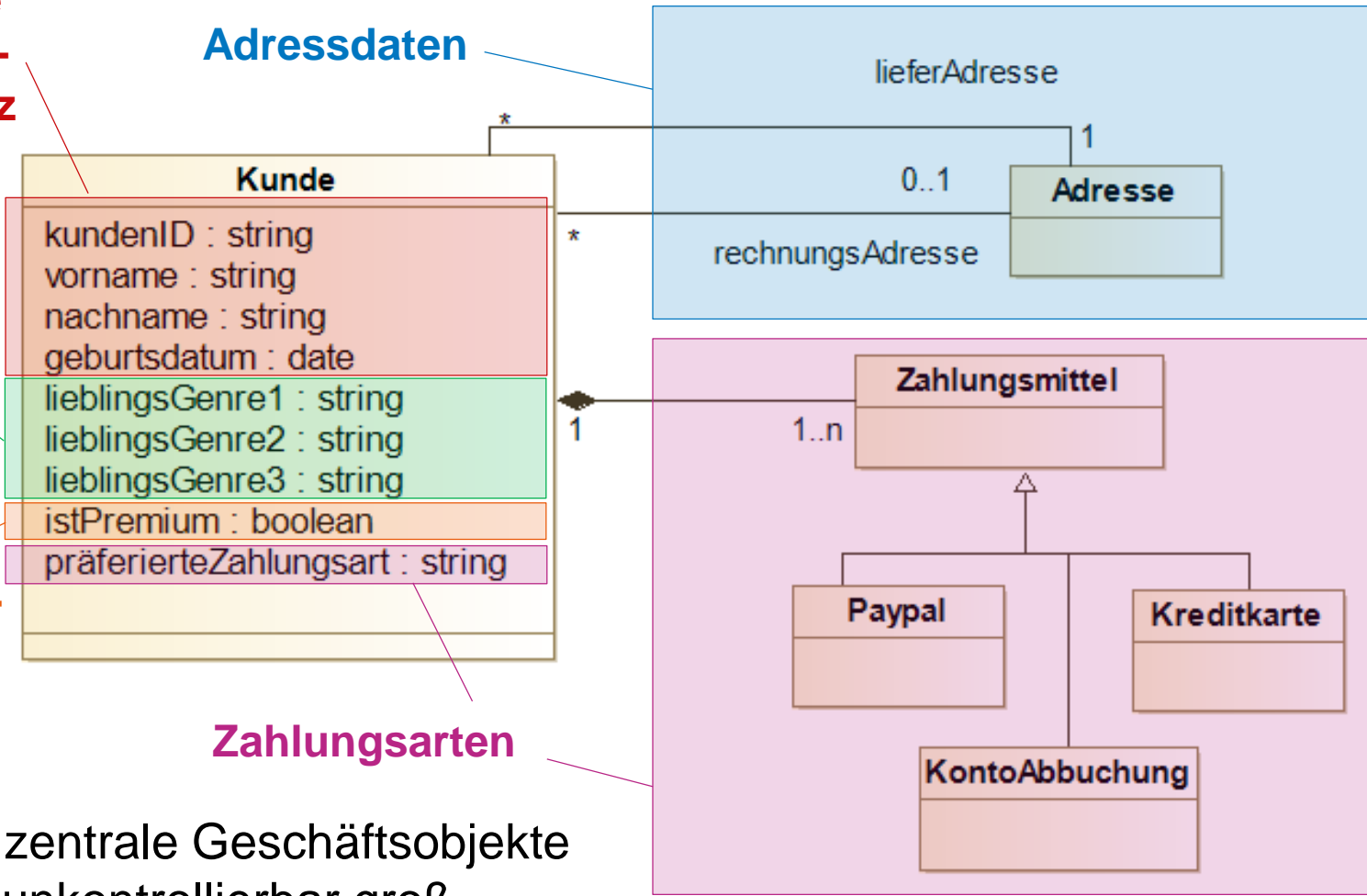
Zentrale Kunden-Referenz

Adressdaten

Präferenz

Kunden-rating

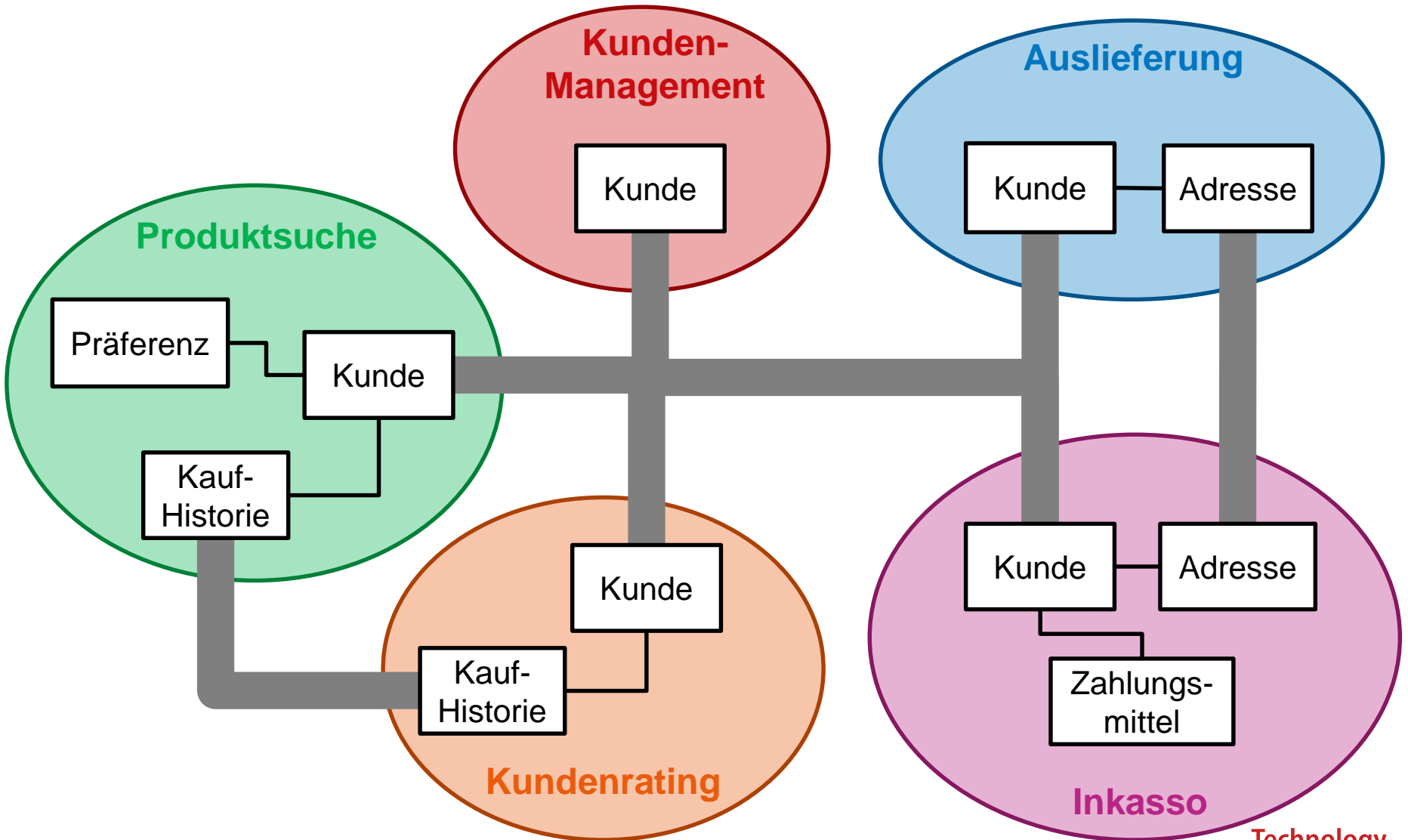
Zahlungsarten



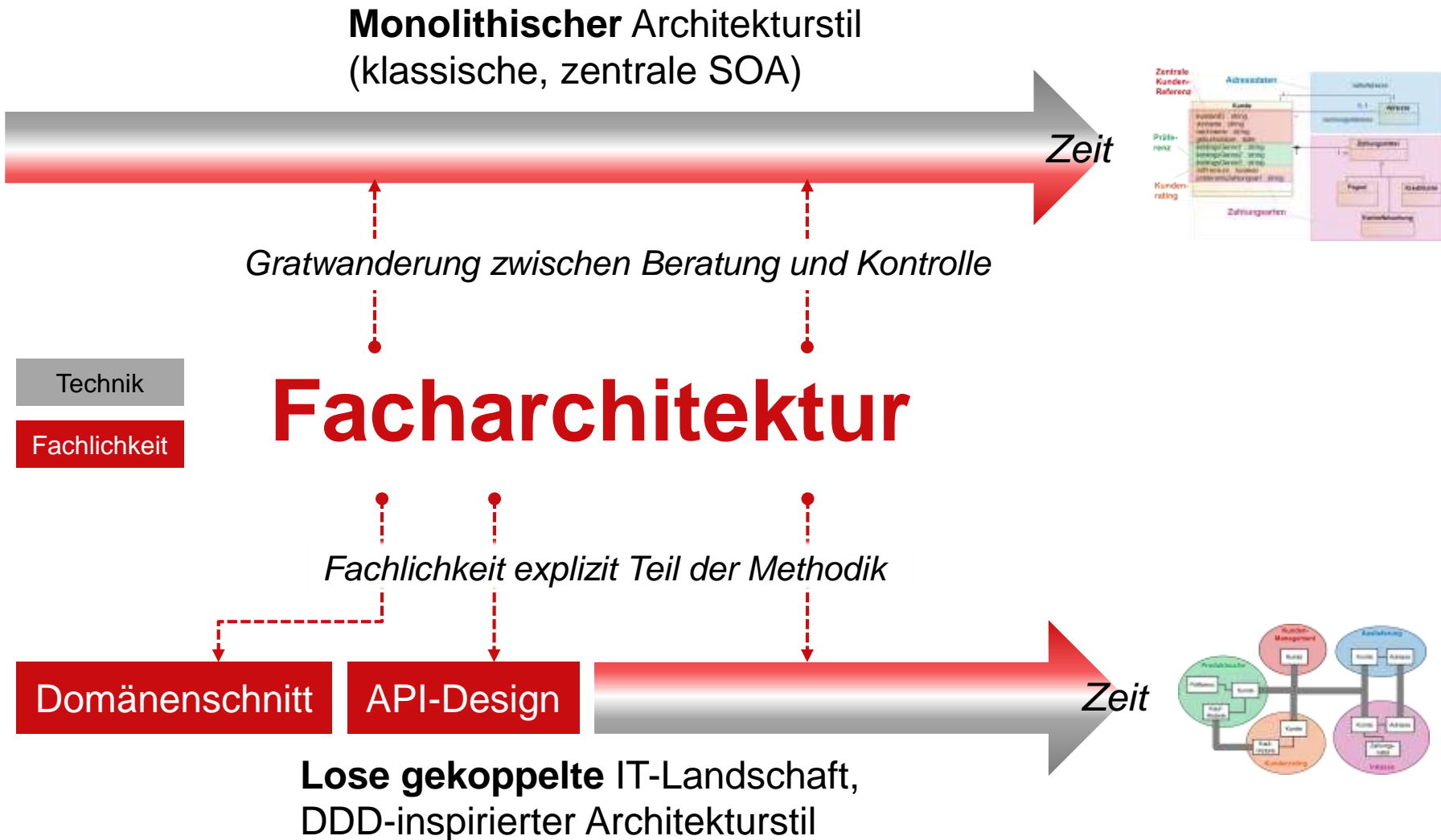
Risiko: zentrale Geschäftsobjekte werden unkontrollierbar groß

(... und das ist schon eine **stark verkürzte** Modellierung!)

DDD: Entkoppelte Domänen mit **Bounded Contexts**

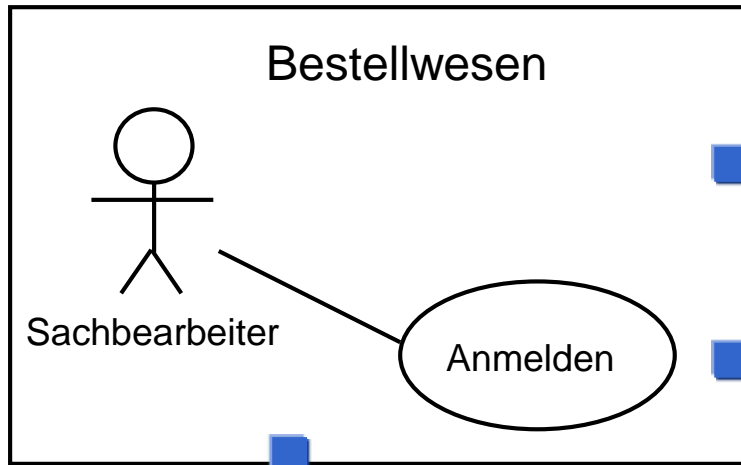


Chancen für die Facharchitektur beim Domänenschnitt



OOSE 2005: Fachlogik in spezialisierte Klassen, anämische Domänenklassen

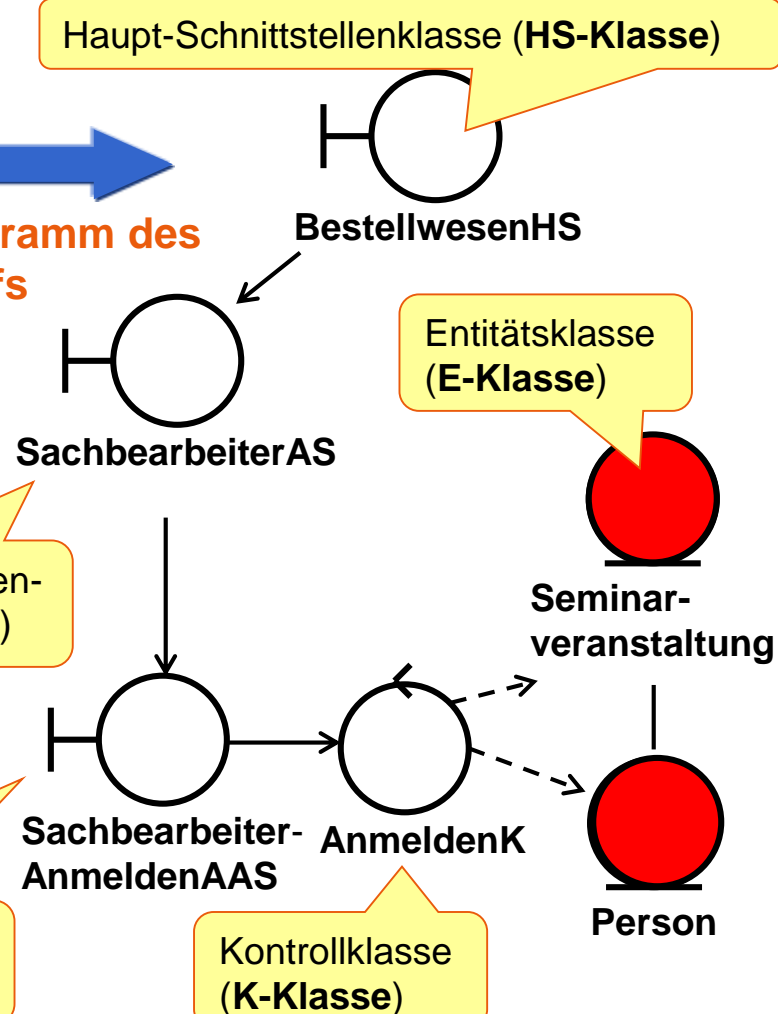
Anwendungsfalldiagramm



Klassendiagramm der Softwarespezifikation

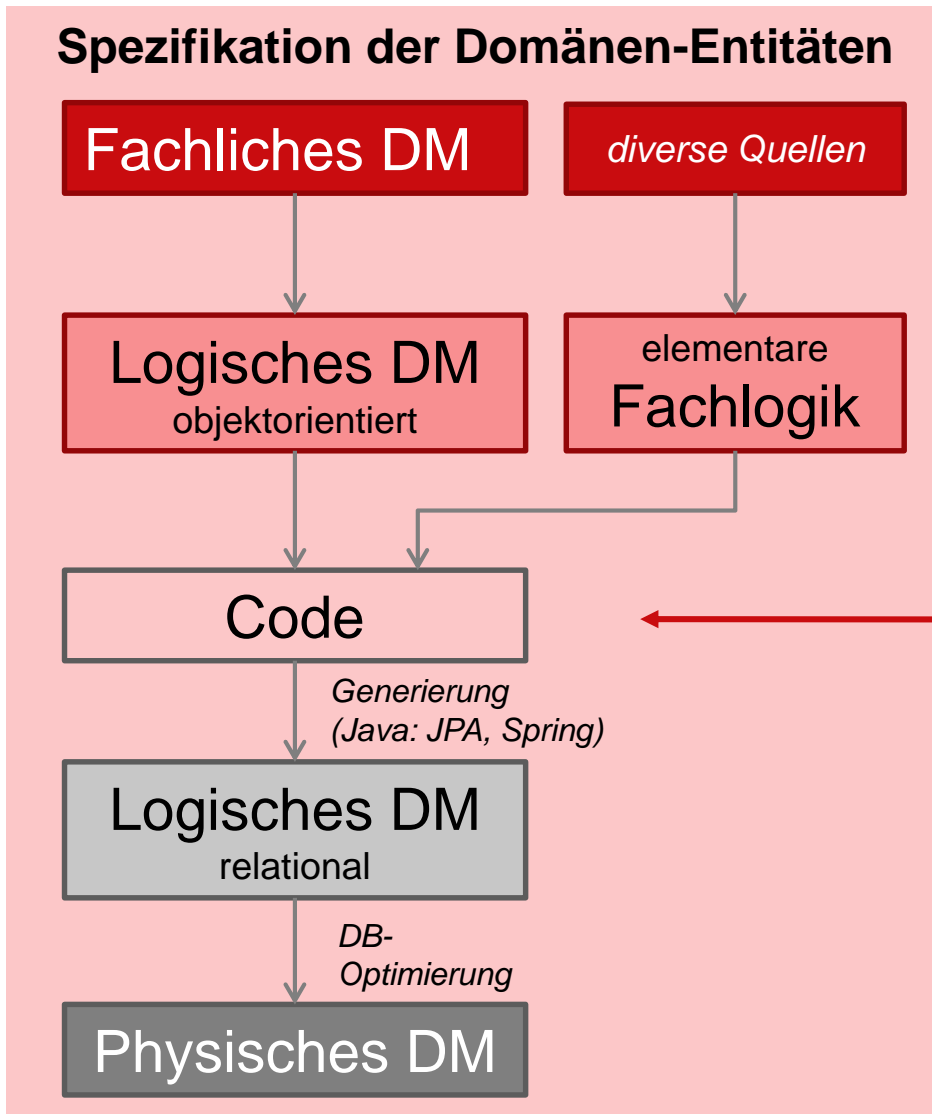


Klassendiagramm des Grobentwurfs



Quelle: M. Winter (2005), Methodische objektorientierte Software-Entwicklung (MOOS), Abbildung aus Vorlesung Softwaretechnik an der TH Köln, 2015.

OOSE heute: **Entität** als „reiche“ Domänenklassen

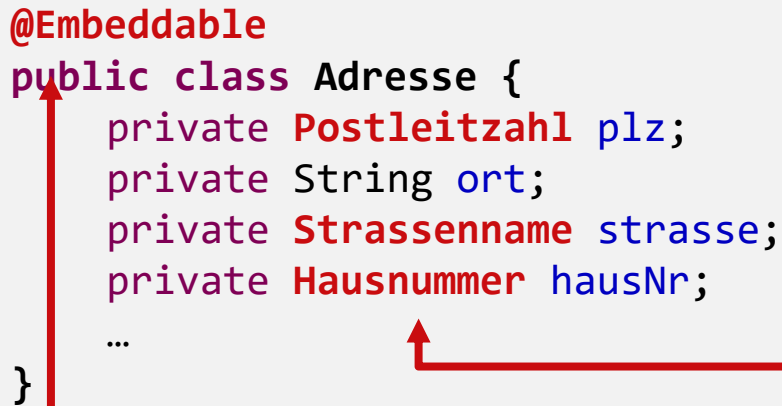


```
@Entity
public class Kunde {
    @Id
    private long id;
    private String name;
    private Date geburtsdatum;
    @OneToOne
    Adresse adresse;
    ...
}
```

```
@Entity
public class Adresse {
    @Id
    private long id;
    private String plz;
    private String ort;
    private String strasse;
    private int hausNr;
    ...
}
```


Value Objects in einem Rich Domain Model als Träger einer Mikro-Fachlichkeit

```
@Embeddable  
public class Adresse {  
    private Postleitzahl plz;  
    private String ort;  
    private Strassenname strasse;  
    private Hausnummer hausNr;  
    ...  
}
```

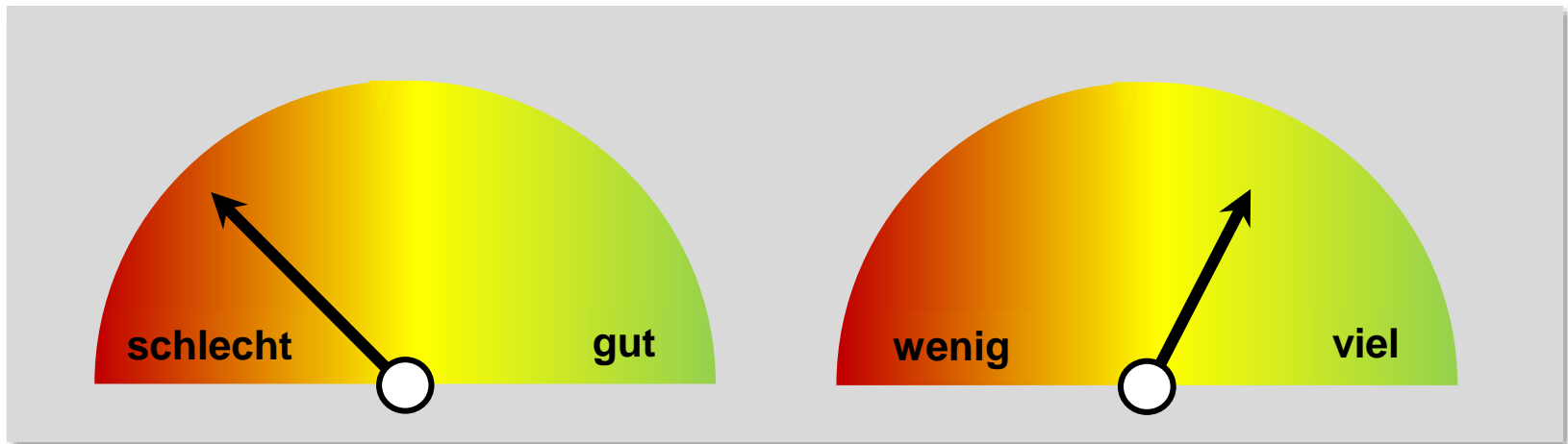


- Einheitlichkeit
- Validierung
- Konvertierung
- Formatierung
- Komposition

Nutzen Sie als Facharchitektur die
Gelegenheit der Mitwirkung auf einer
(vermeintlich) sehr technischen Ebene!

These 3

Je höher die Flughöhe einer Architektur, desto wichtiger sind **klare Ziele** und **gute Messsysteme**.



(Und um so seltener ist dieser Bedarf in der Praxis angemessen erfüllt.)

Hohe Autonomie

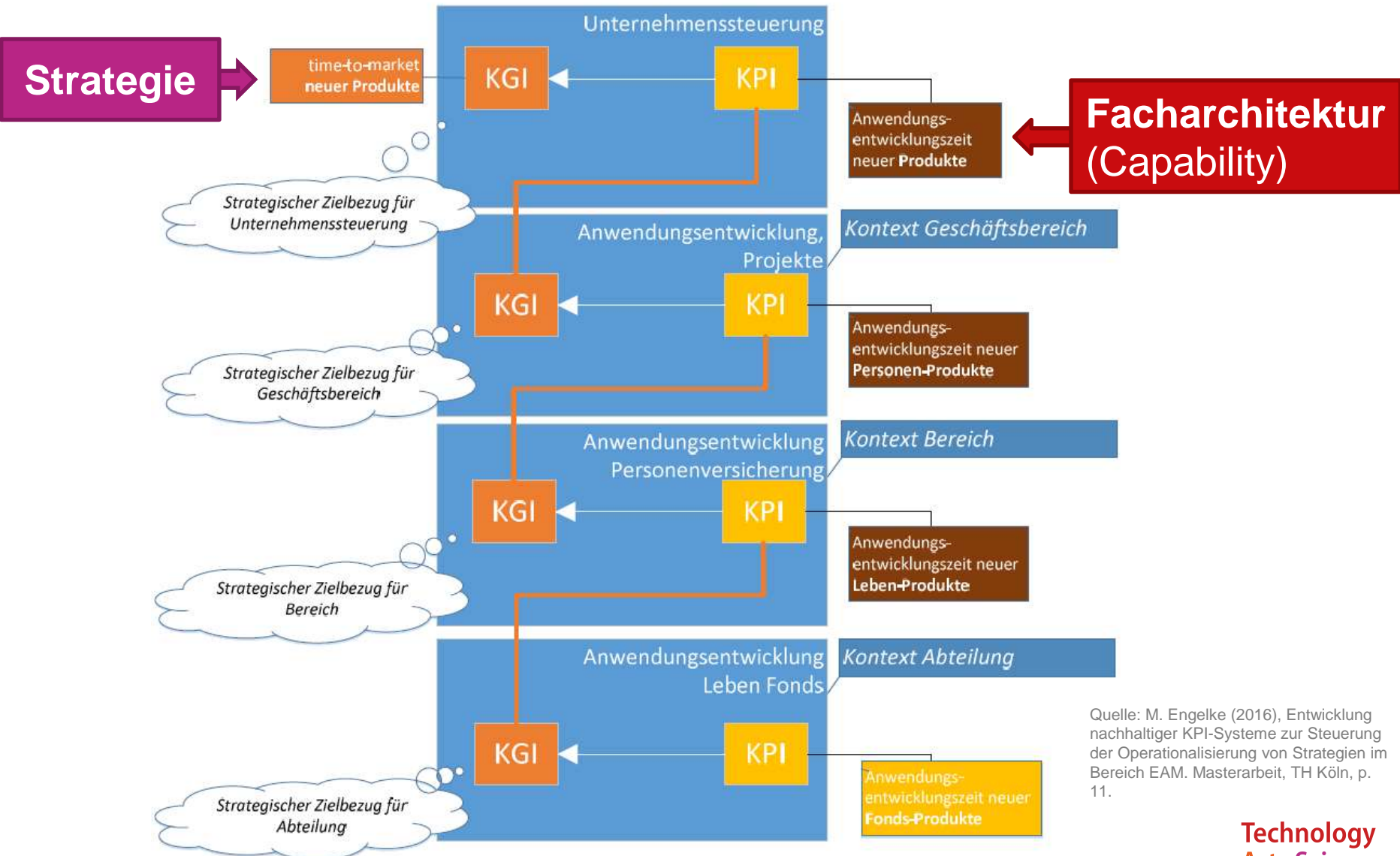
⇔ klare Zielvorgaben

⇔ Messung der Zielerreichung



These 3: Je höher die Flughöhe einer Architektur, desto wichtiger sind klare Ziele und gute Messsysteme

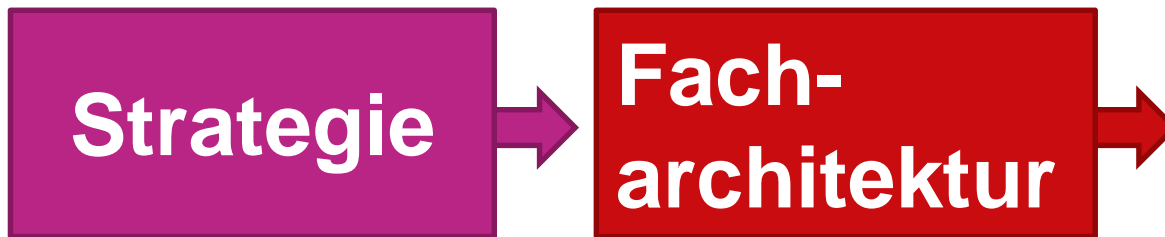
Dekomposition strategisch verankerter KPI-Systeme



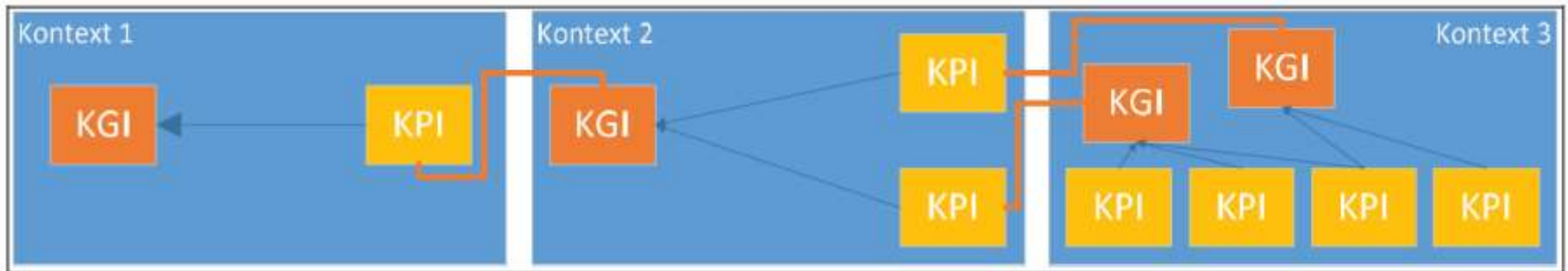
Quelle: M. Engelke (2016), Entwicklung nachhaltiger KPI-Systeme zur Steuerung der Operationalisierung von Strategien im Bereich EAM. Masterarbeit, TH Köln, p. 11.

These 3: Je höher die Flughöhe einer Architektur, desto wichtiger sind klare Ziele und gute Messsysteme

Rolle der Facharchitektur bei der KPI-Dekomposition



- Applikationsarchitektur
- Technische Architektur
- Betrieb
- Anforderungsmanagement
- Changemanagement
- ...



Quelle: ebd., p. 12.



Fazit