

# Schlankheitskur für die SOA?


07.11.2013, A. Arnold

1. Heavy SOA

2. Struktur – Domain Driven Design 4 SOA

3. Agile SOA? – Prozesse vereinfachen

4. Continuous Integration + DevOps



Big Ball of  
(Service) Mud

Abstimmung

Prozesse?  
Wasserfall

(zu viel)  
SOA Governance  
+ Politik

Komplexe  
Infrastruktur

Quelle: [http://www.laufexperten.de/files/inketten\\_verlauf\\_banner\\_575x200px.jpg](http://www.laufexperten.de/files/inketten_verlauf_banner_575x200px.jpg)

Modelle und DDD als Ansatz

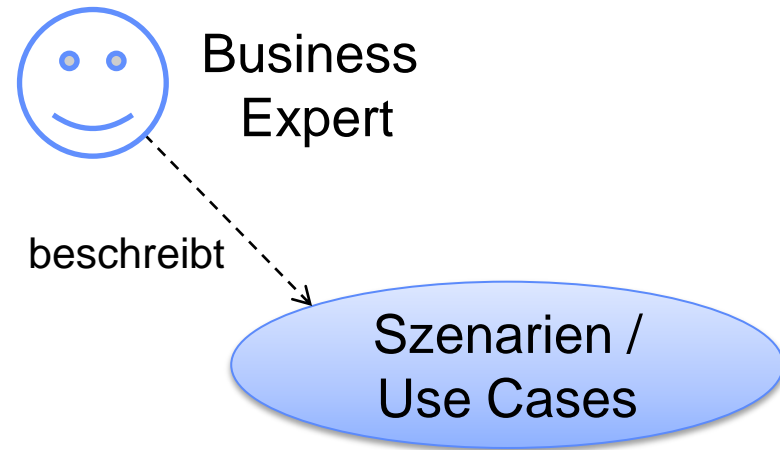
# DOMAIN DRIVEN DESIGN

# Was wollen wir erreichen?

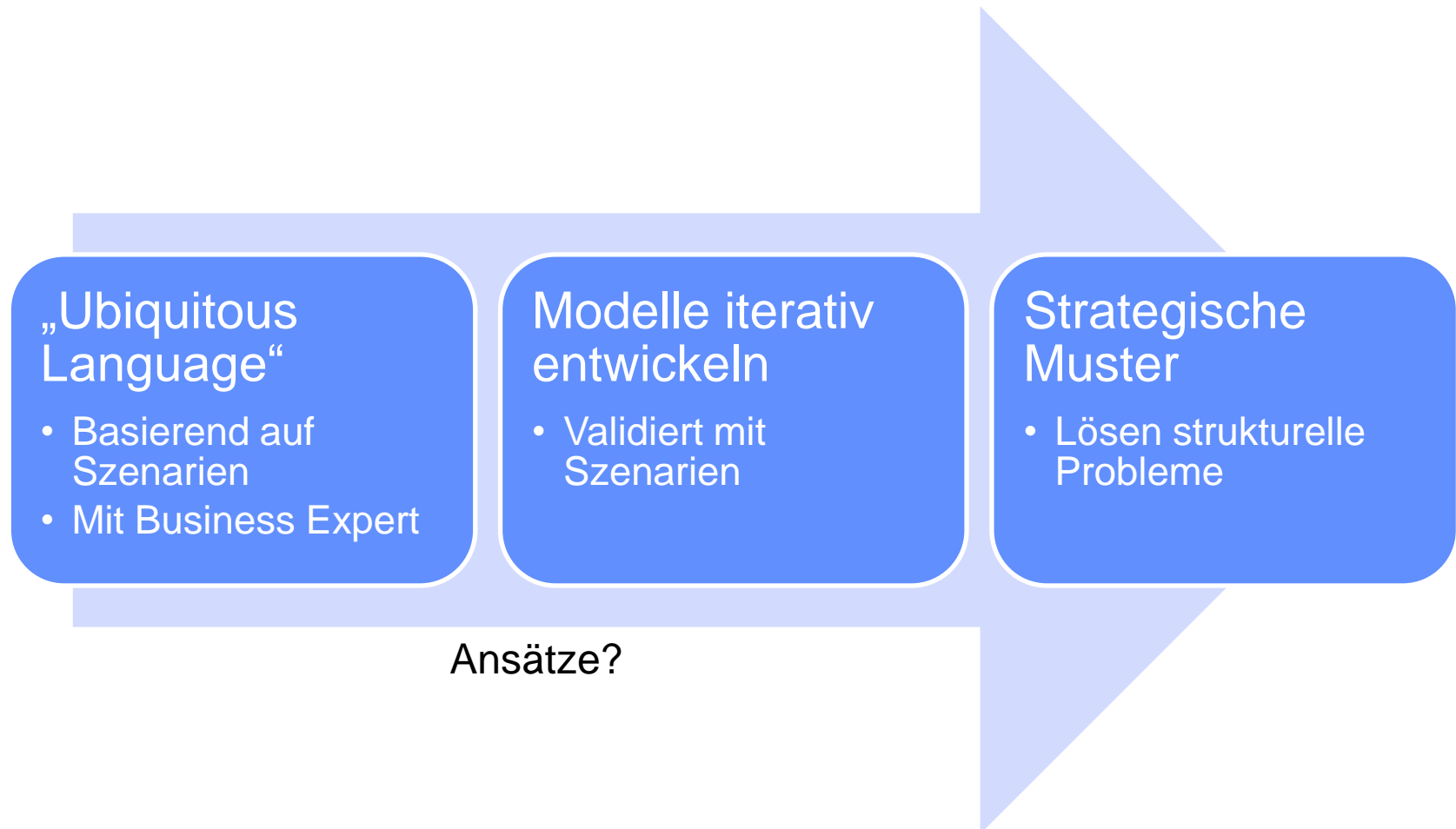
- Working software
- Geeignete Lösung des fachlichen Problems
- tragfähige ARCHITEKTUR

## Ausgangspunkt:

Das **Problem** und dessen  
**Kontext** verstehen



## Problem Domain im Mittelpunkt





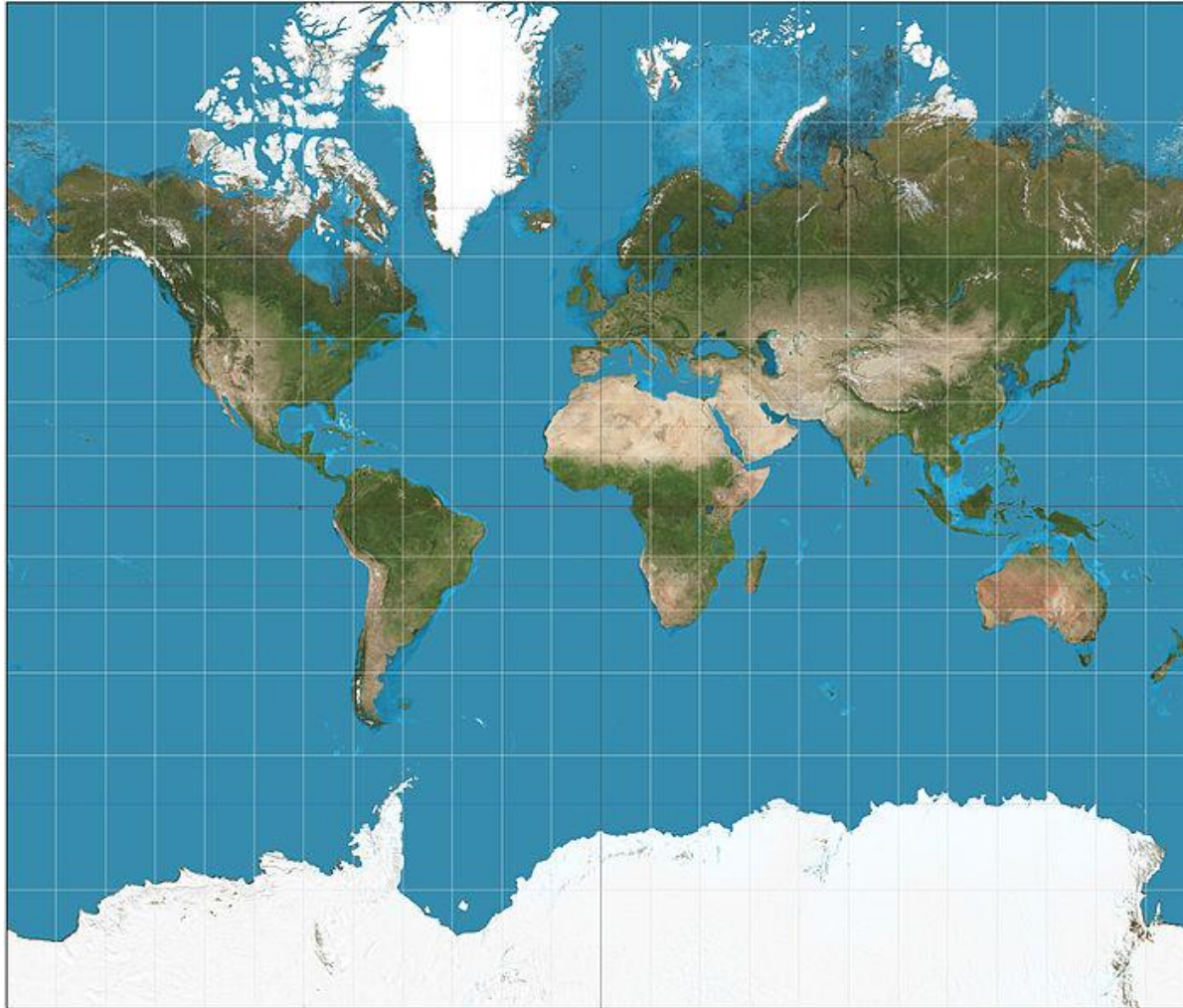
Quelle: <http://datendealer.s3.amazonaws.com/2013/2/28/world%20grey.png>

„Eine **Flugreise** von einem **Startort** zu einem **Zielort** setzt sich aus ein oder mehreren **Flügen** mit **Zwischenstopps** zusammen. “



## Exkurs: Was ist ein Modell?

# Weltkarte (Mercator-Projektion) *trust in competence*



 Quelle: [http://upload.wikimedia.org/wikipedia/commons/f/f4/Mercator\\_projection\\_SW.jpg](http://upload.wikimedia.org/wikipedia/commons/f/f4/Mercator_projection_SW.jpg)

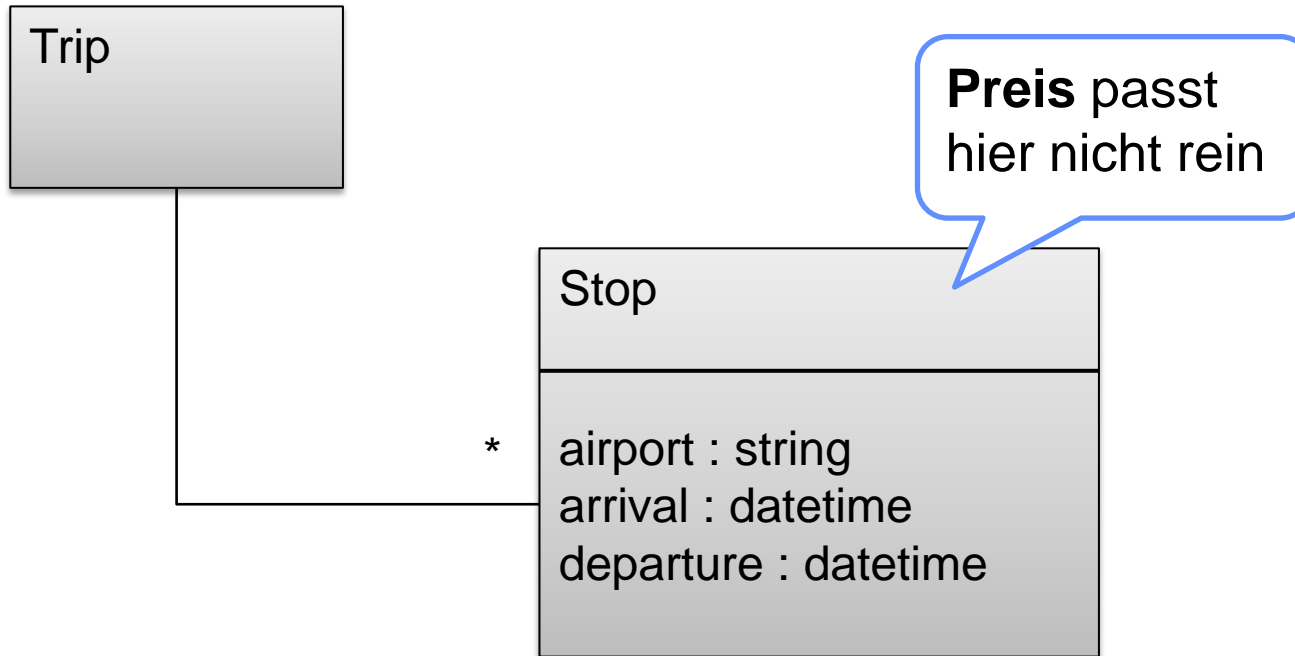




Quelle: <http://upload.wikimedia.org/wikipedia/commons/a/ac/World-airline-routemap-2009.png>

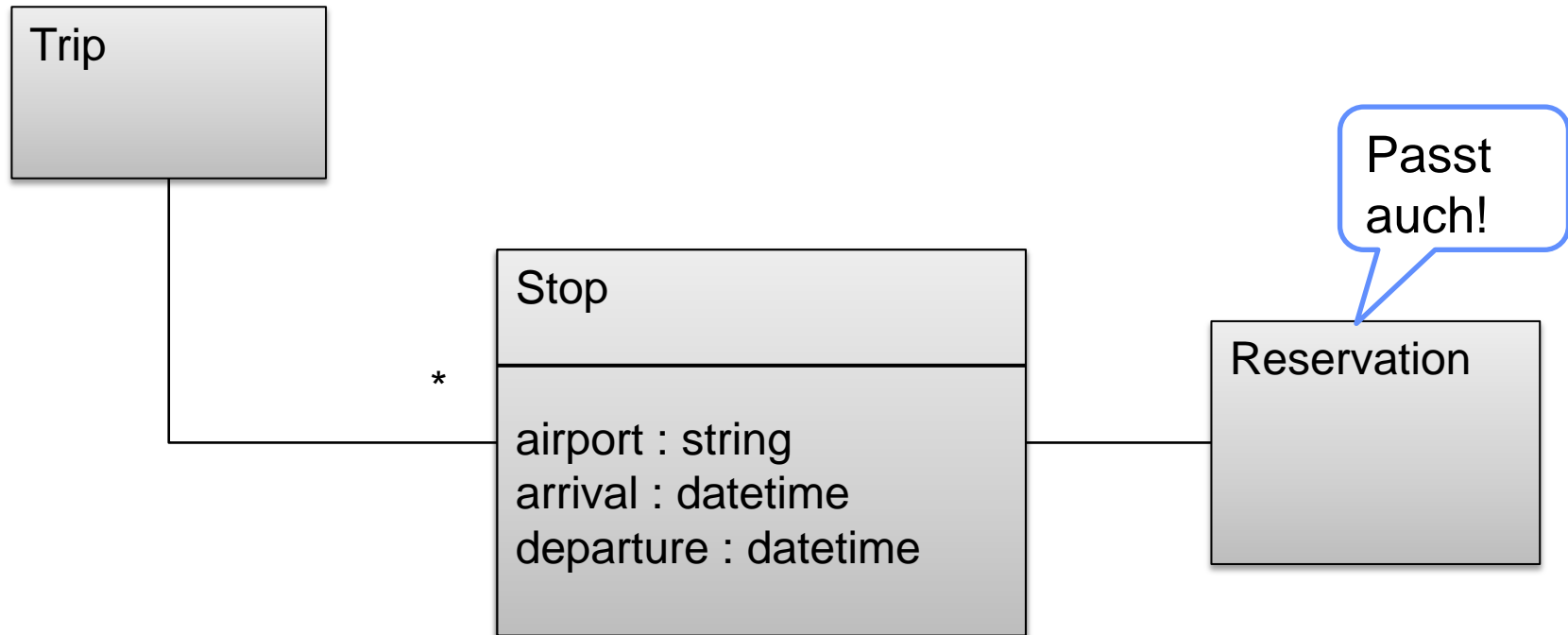
Eric Evans:

„A model is a **system of abstractions** that describes **selected** aspects of a domain of knowledge and can be **used** to solve problems **related to that** domain.“



## ● Neues Szenario:

„Ein Kunde interessiert sich für die schnellste oder die preiswerteste Reiseroute.“



- Und noch ein neues Szenario:

„Ein Kunde bucht eine Reise.“ Für jede Teilstrecke wird die erforderliche Zahl an Sitzplätzen **reserviert**.

# STRATEGISCHE MUSTER

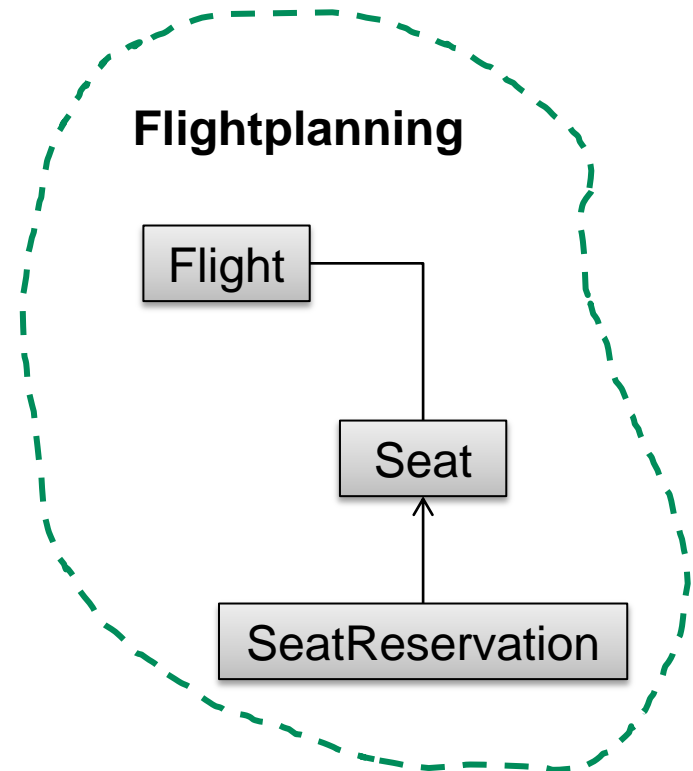
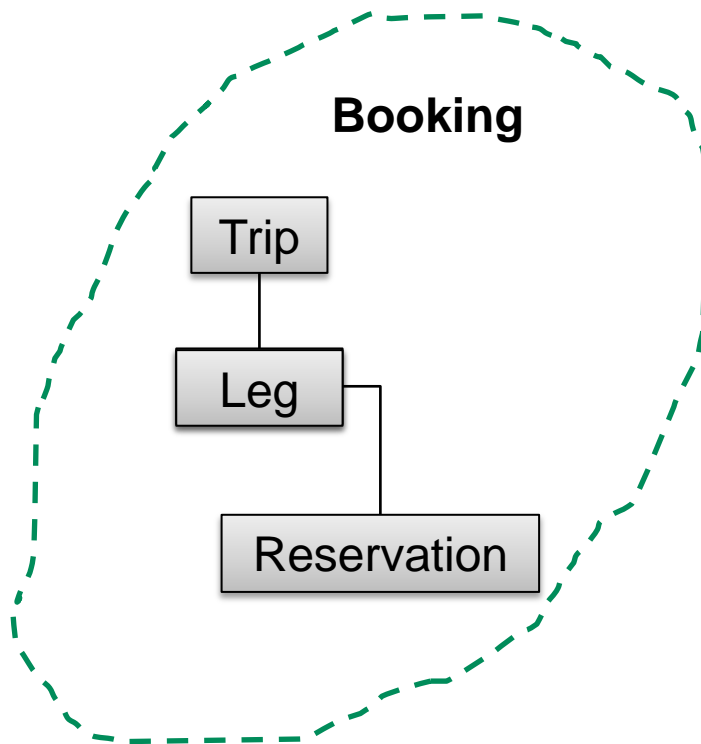
Lose Designkopplung im Modell

# BOUNDED CONTEXT



## Verschiedene „Bounded Contexts“

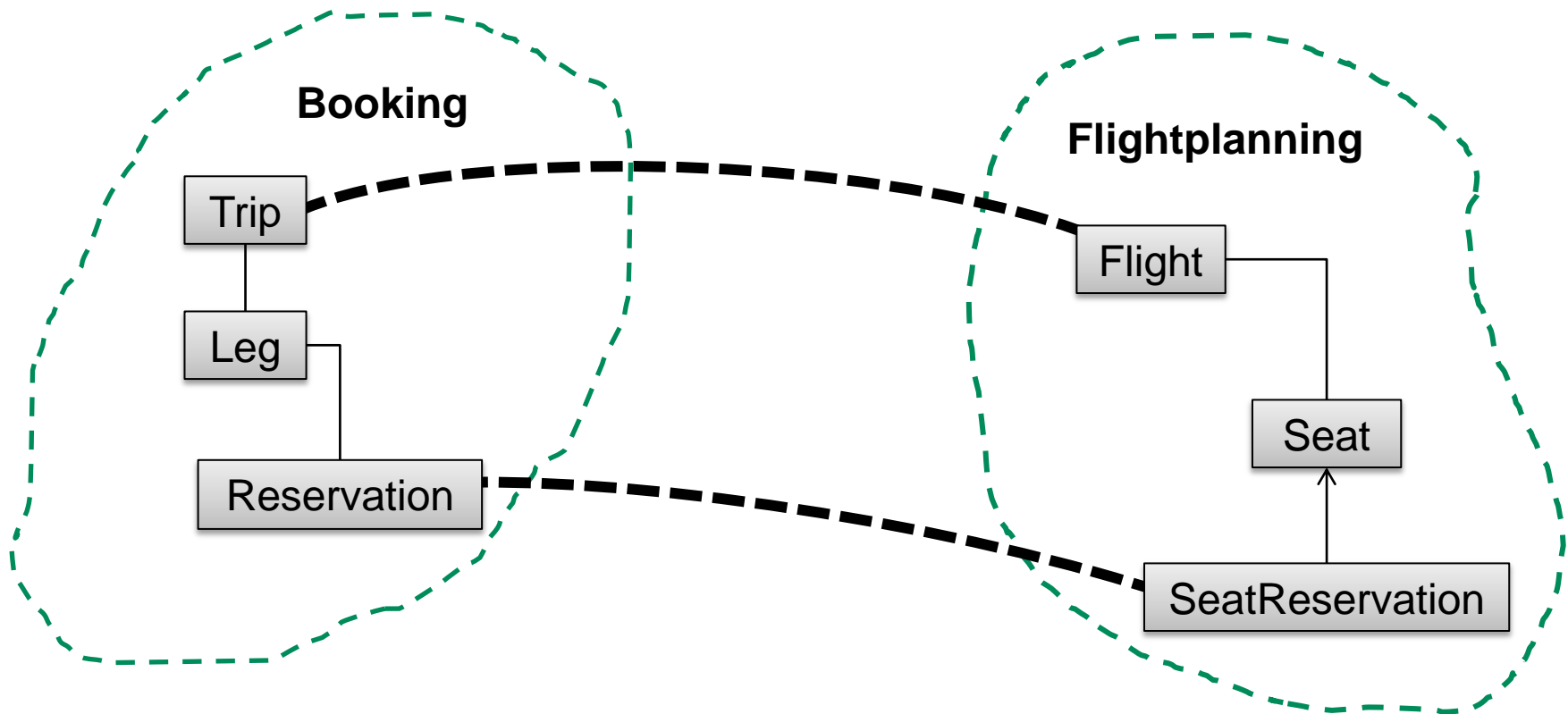
## Verschiedene „Ubiquitous Languages“

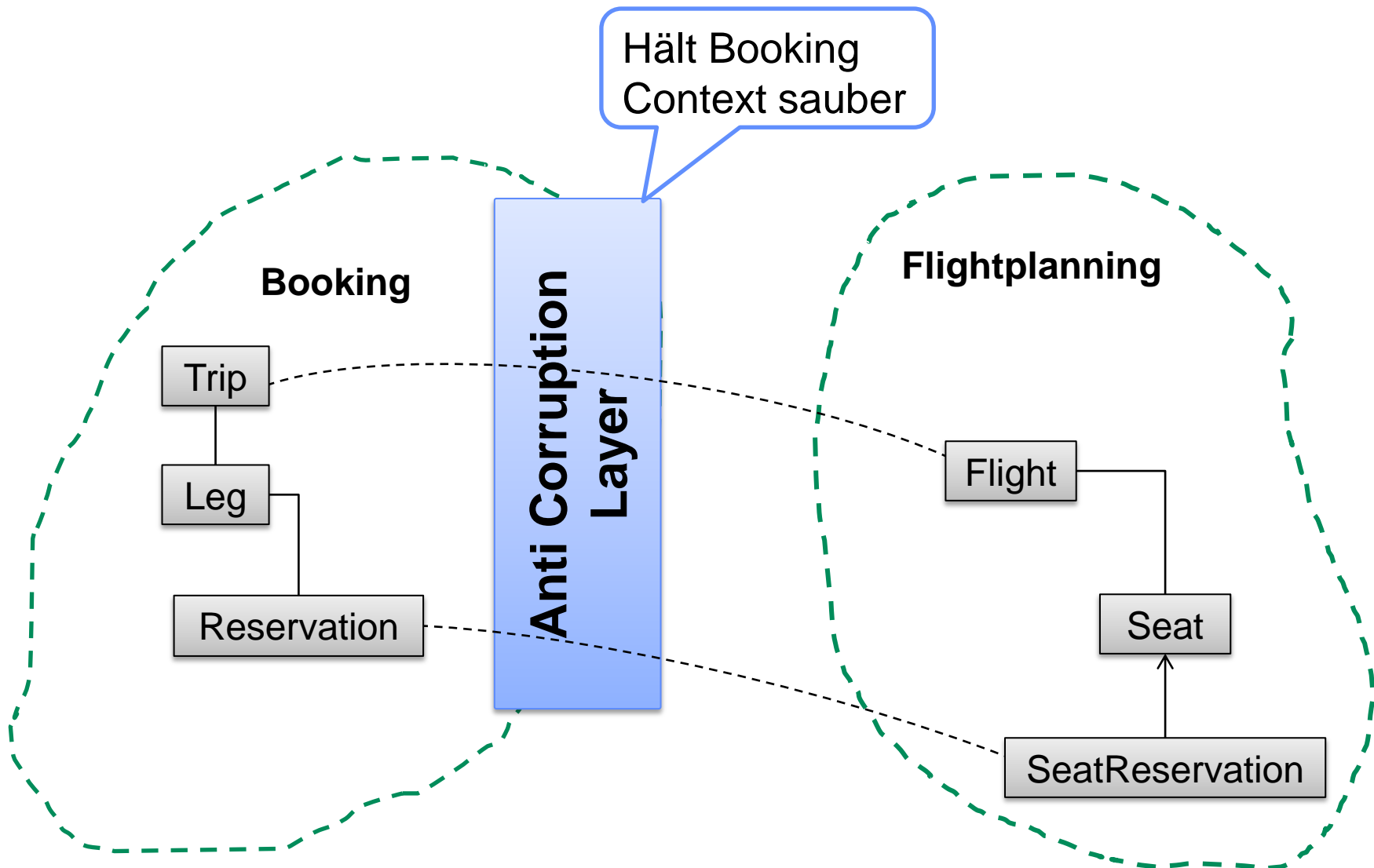


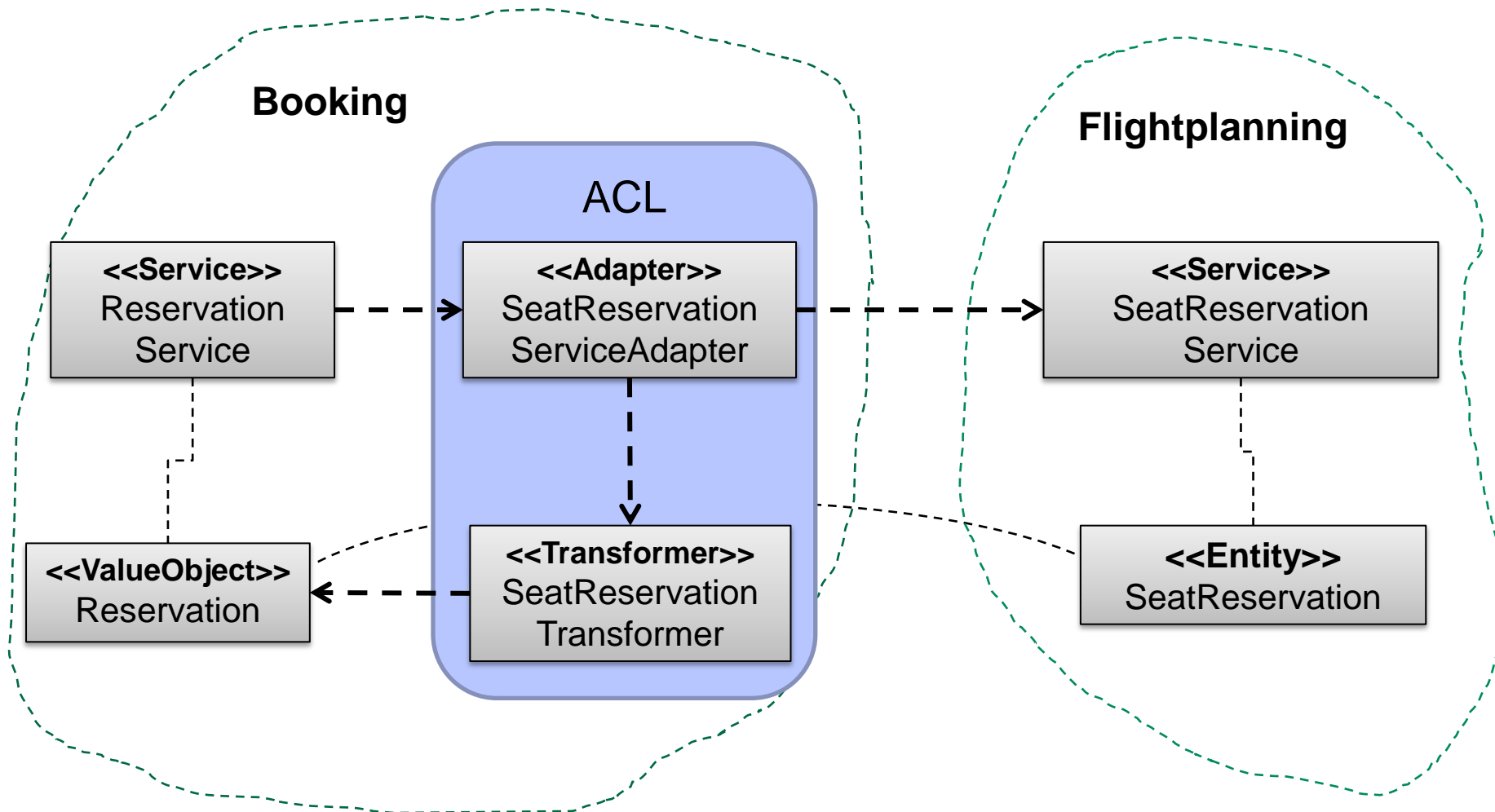
*"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations".*

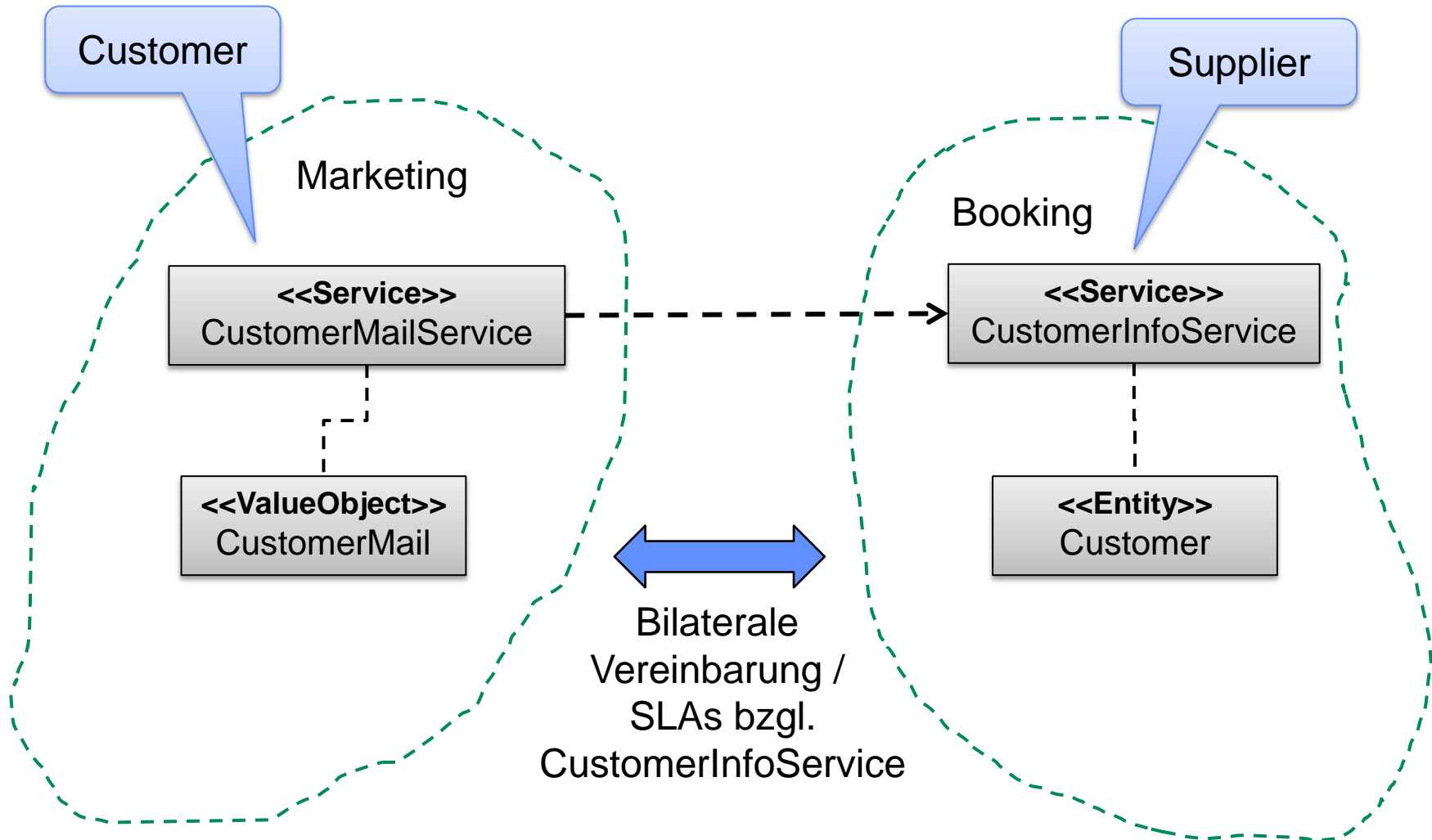
Quelle: [http://en.wikipedia.org/wiki/Conway%27s\\_law#cite\\_note-Conway-2](http://en.wikipedia.org/wiki/Conway%27s_law#cite_note-Conway-2)

## „Context Map“



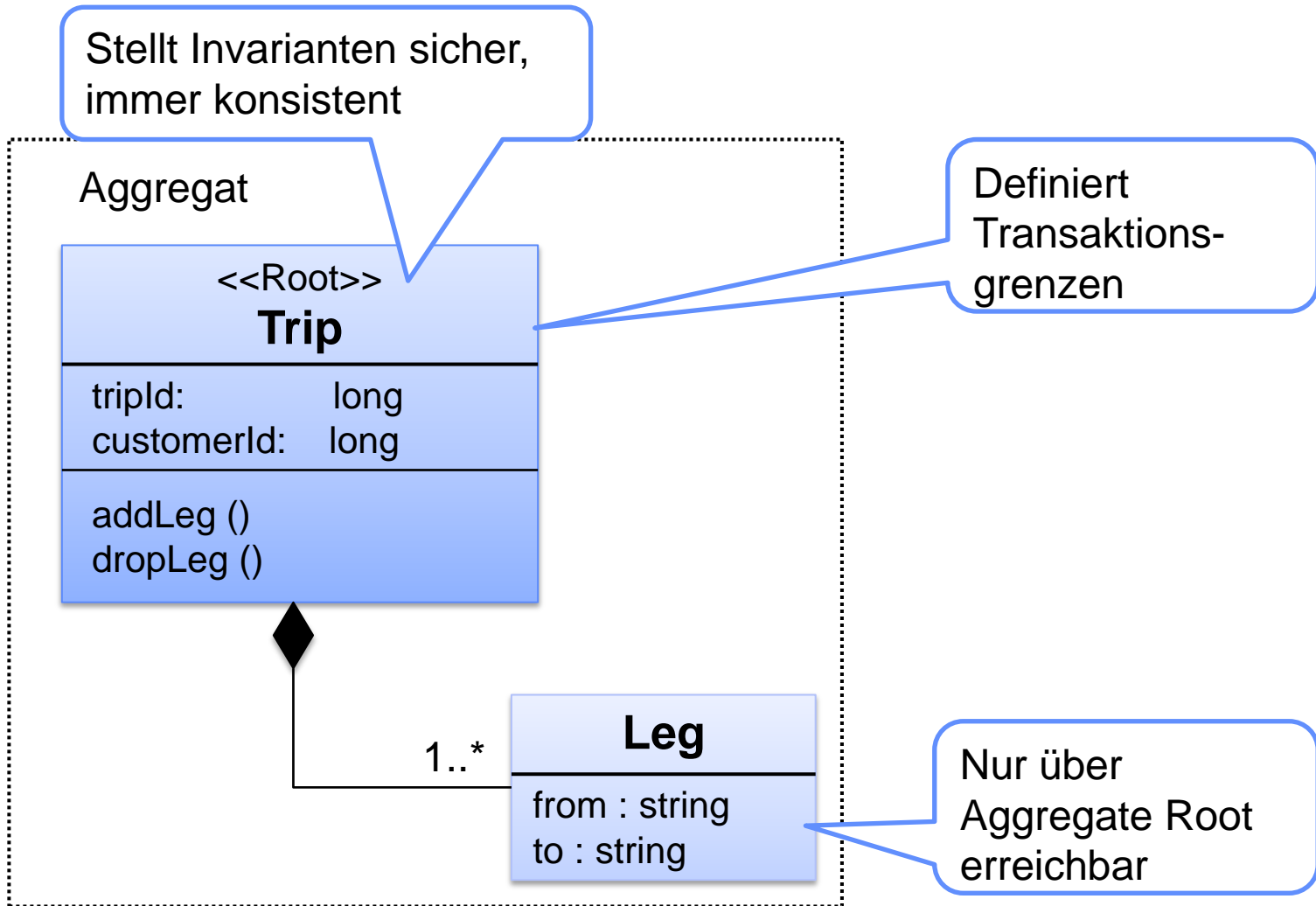






Mit Objektgraphen und Concurrency umgehen

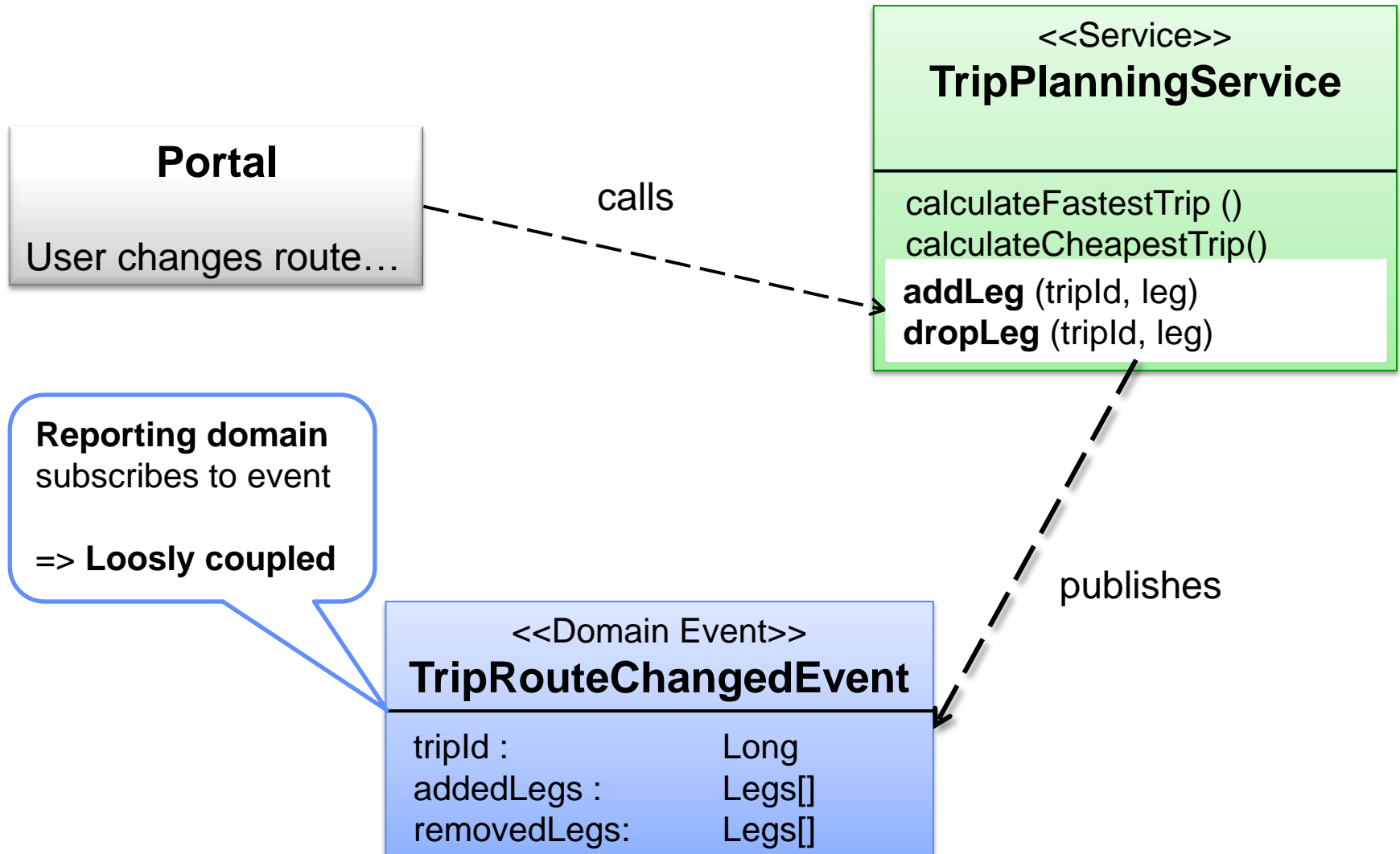
# AGGREGATES



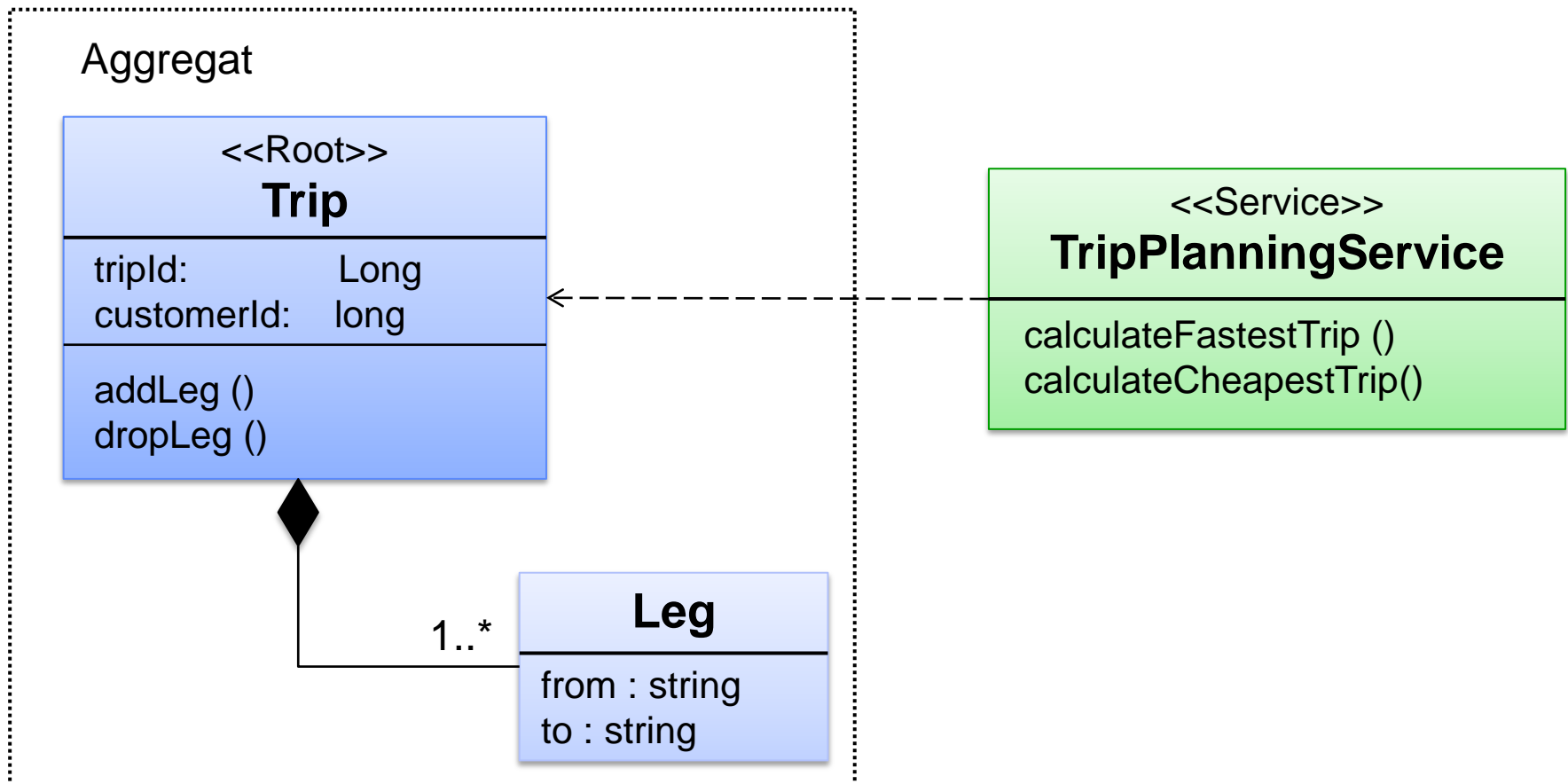


# DOMAIN EVENTS

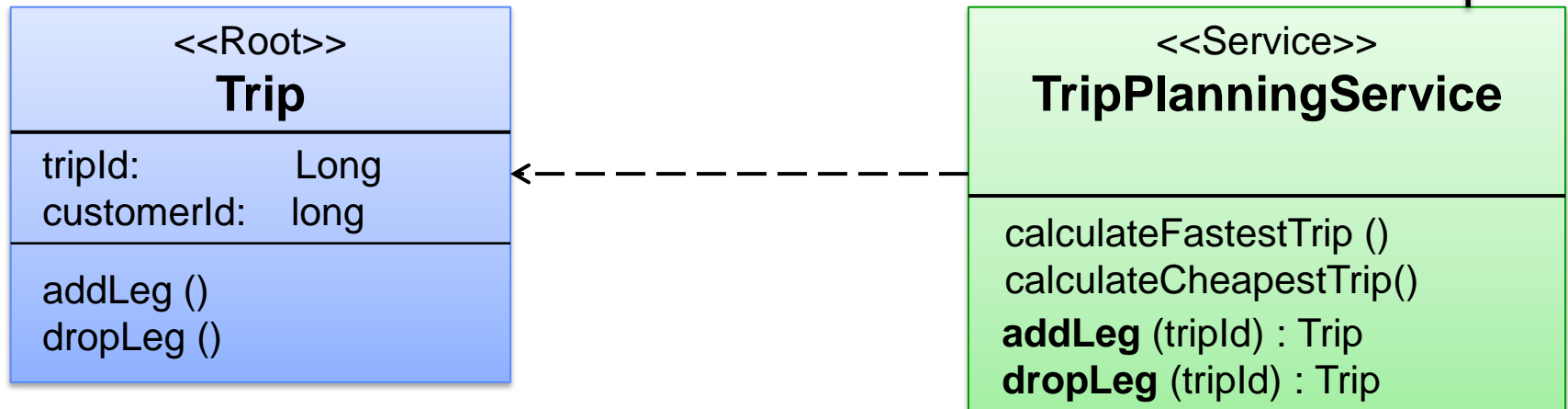
# Lose Kopplung per Event



# SERVICES



„Open Host Service“  
HTTP / SOAP o. REST



- Strukturtierung nach Use Cases
  - Zusammengehörende Use Cases => ein Service 😊
  - Problematisch:  
Strukturierung nach Entity => breite Services ☹️
- Erweiterbar
  - Versionierung / Modellevolution
- Services für Verwaltung von Aggregaten
  - Alle Zustandsänderungen
  - Transaktionen hier
  - Auch als Command implementierbar

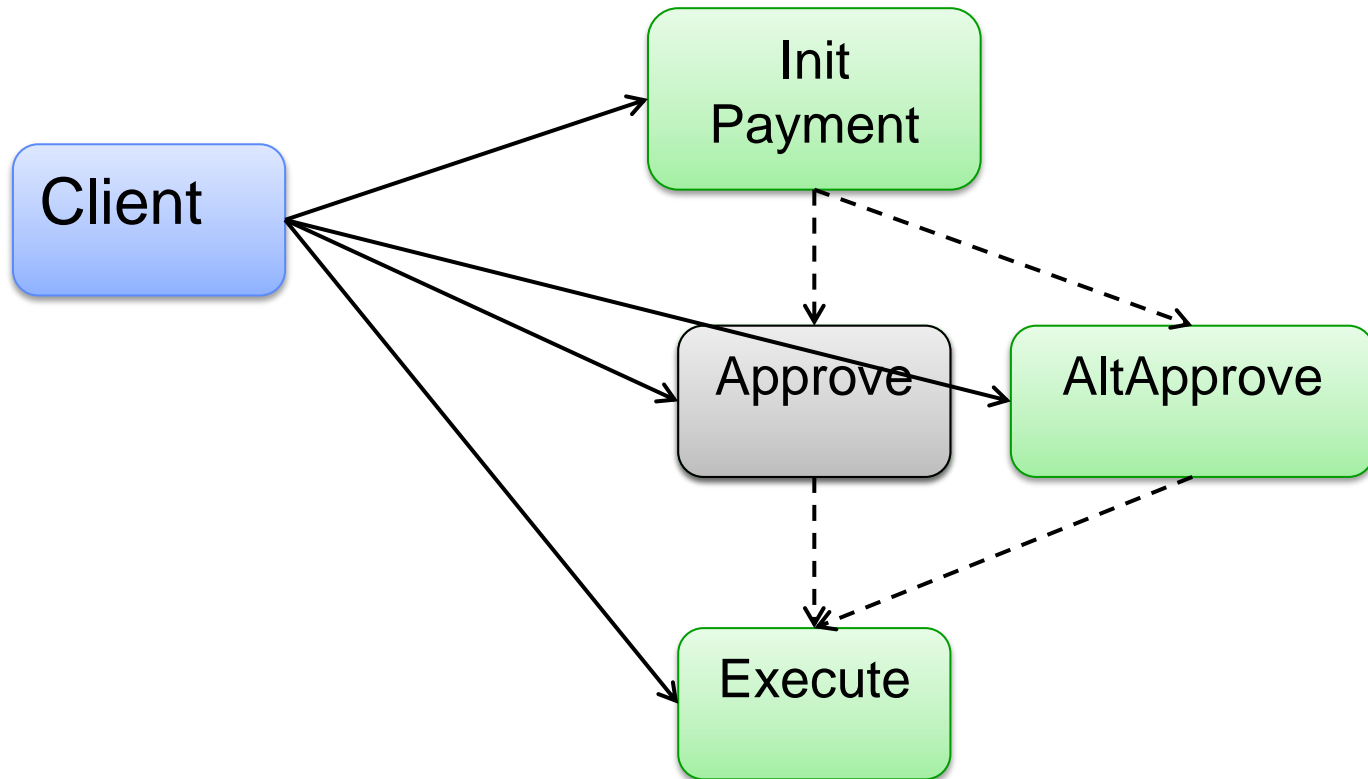
- Domain Language + Model iterativ entwickeln
- Strategic DDD Patterns zur Strukturierung
  - Bounded Contexts und Context Maps
  - Aggregates
  - Service und Open Host Services
- Voraussetzungen
  - Business Expert verfügbar
  - Iteratives Vorgehen

REST und HATEOAS

# SONSTIGE STRATEGIEN



## **HATEOAS :** **Hypermedia As The Engine Of Application State**



Paypal:

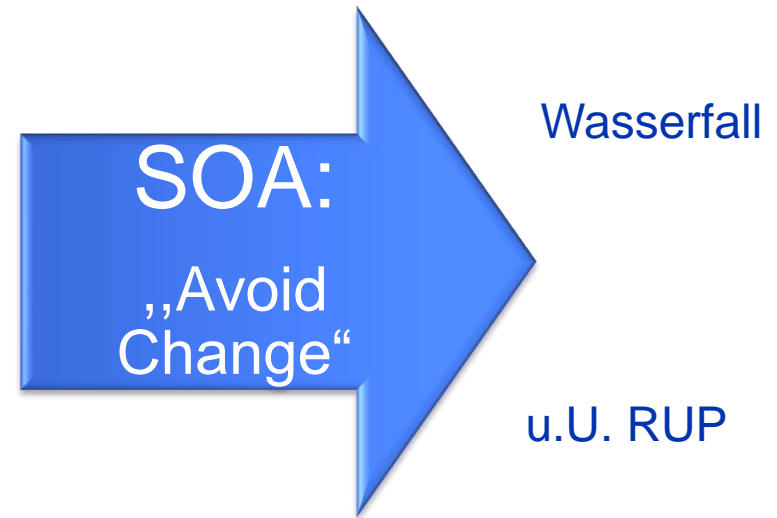
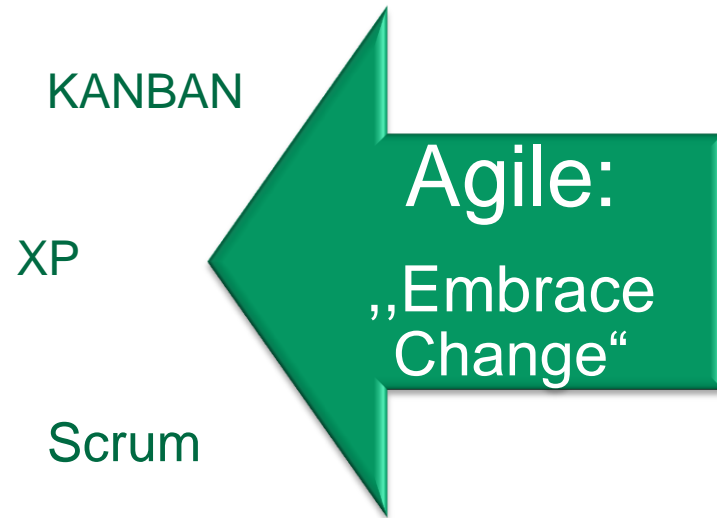
```
[  
  {  
    "href": "https://api.sandbox.paypal.com/v1/payments/payment/PAY...",  
    "rel": "self", "method": "GET"  
  },  
  
  {  
    "href": "https://www.sandbox.paypal.com/webscr?cmd=_express-checkout...",  
    "rel": "approval_url", "method": "REDIRECT"  
  },  
  
  {  
    "href": "https://api.sandbox.paypal.com/v1/payments/payment/PAY... /execute",  
    "rel": "execute", "method": "POST"  
  }  
]
```

Links

Semantik

Quelle: <https://developer.paypal.com/webapps/developer/docs/integration/direct/paypal-rest-payment-hateoas-links/>

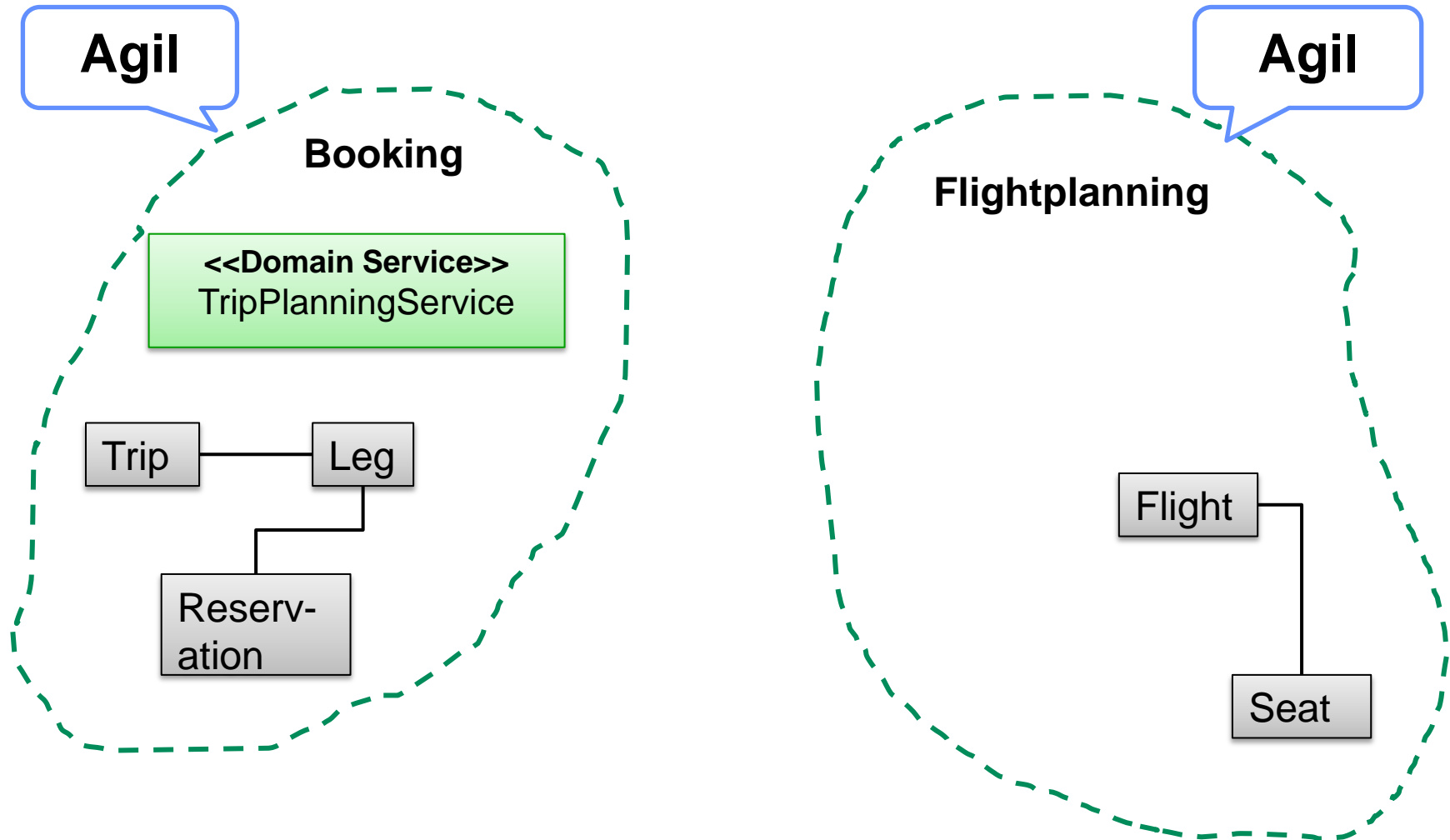
# AGILE SOA?



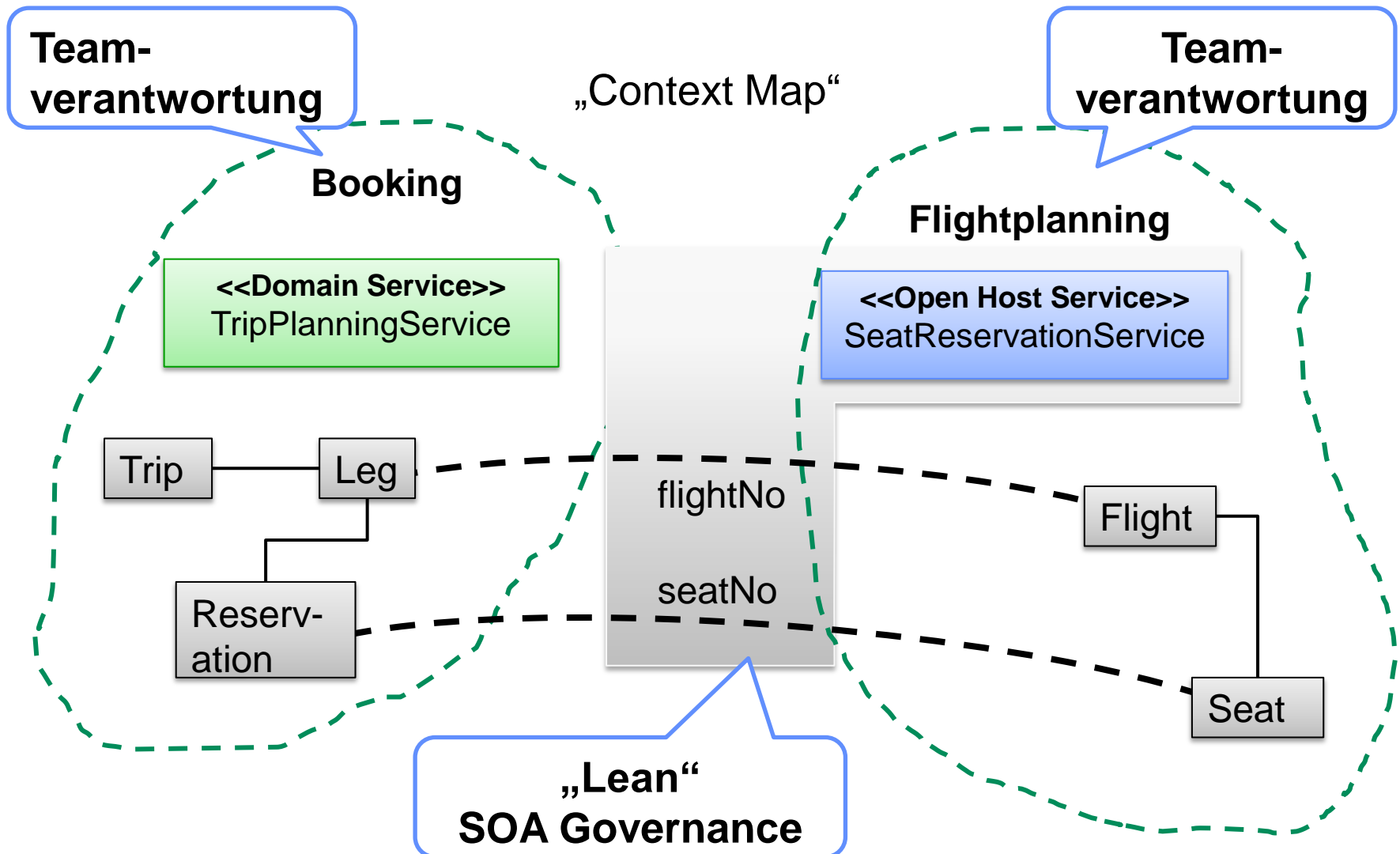
(auch) Schnittstellen ändern sich

In einer SOA gelten SLAs

- zw. Consumer + Provider
  - Erfordern Abstimmung
- ⇒ SOA Governance



# Was fällt unter Governance?



- **Automatisiert** für formale Regeln
- Möglichst viel Teamverantwortung
- Konventionen „ohne daran denken zu müssen,,

## Explizit

- Validierungsregeln
- z.B:
  - Namenskonventionen
  - Service ungenutzt?

## Implizit

- Contract First
- generative Ansätze (MDSD)
- z.B:
  - WSDL / XSD generieren

- Domain Specific Language für einen Zweck:

- Contract First mit DDD + Services
- Teilautomatisierung der SOA Governance

+ Validierung  
+ Codegenerierung  
(WSDL, XSD etc.)

```
1 businessObject Leg {  
2     metadata {  
3         version 1.0  
4     }  
5     flight : string  
6     departureAirport : string  
7     arrivalAirport : string  
8     departureTime : datetime  
9     arrivalTime : datetime  
10    price : Money majorVersion 1  
11 }
```

```
1 public service SeatReservationService {  
2     metadata {  
3         version 1.0  
4         lifecycle-state development  
5     }  
6     getAvailableSeats (flightId : string,  
7         numberOfSeats : int)  
8     returns seats : int [] {  
9  
10    }  
11 }  
12
```



- Sprache mit Grammatik definieren

```
1  Service:
2      visibility=Visibility 'service' name = ID '{'
3          'metadata' '{'
4              version = Version
5              'lifecycle-state' state = [LifecycleState | QualifiedName]
6              (governanceDecisions += GovernanceDecision)*
7              ('service-category'
8                  category = [ServiceCategory | QualifiedName])?
9              ('statefulness' statefulness = Statefulness)?
10             ('tags' (tags += [Tag]))+)?
11         '}',
12         (operations += Operation)*
13     '}';
```

- Highlighting, Contentassist, Outlines etc. „geschenkt“
- Regeln als Validatoren in Java
- Codegenerierung mit Xtend2

Continuous Integration und DevOps

# **BUILD + RELEASE**

- Etabliert
- Deployment auf ESB als Build
  - Meist nicht Out of the Box
  - Deployment von Stage zu Stage
- Tools
  - Maven / Ant / Gradle
  - JUnit / Fit
  - Maven Release Plugin
  - Jenkins & Co

- Entwicklung und Betrieb Hand-in-Hand
- „Infrastructure as code“
  - Versioniert in SVN/Git
  - wiederholbar

## Virtual Machines vom Image – Vagrant

- `$ vagrant init precise32 http://files.vagrantup.com/precise32.box`
- `$ vagrant up`

## Konfigurationsskripte – Puppet / Chef

- User/Rechte anlegen
- System Services bereitstellen (Tomcat, ActiveMQ etc.)
- MCollective für Serverfarm-Administration
- ...

## Struktur + Entflechtung

- Domain Driven Design für Services
- Flexible Versionierung
- Entflechtung der Kommunikation

## Agilität

- Agile Prozesse + Lean Governance
- Domain Specific Languages
- Möglichst selbständige Teams
- Continuous Integration / Continuous Delivery / DevOps
- Automatisierung

# Vielen Dank!

André Arnold

anderScore

<http://www.anderscore.com>

[andre.arnold@anderscore.com](mailto:andre.arnold@anderscore.com)

<http://andre-arnold.blogspot.de>

- Domain Driven Design Quickly  
<http://www.infoq.com/minibooks/domain-driven-design-quickly>
- Vaughn Vernon, Restful SOA and DDD:  
<http://www.infoq.com/presentations/RESTful-SOA-DDD>
- Udi Dahan  
<http://www.udidahan.com/>
- Xtext  
<http://www.eclipse.org/Xtext/>
- Xtext Service Repository  
<https://github.com/andearn/org.fornax.soa.xtextservicerepository>
- Vagrant  
<http://docs.vagrantup.com/v2/getting-started/index.html>
- Puppet  
<http://puppetlabs.com/>
- DevOps for Developers, Michael Hüttermann
- Karten: Wikimedia