

Master Thesis - final presentation

# Interactive Visualizations for supporting the analysis of distributed services utilization

Daniel Graf Hoyos, 11.06.2018, Munich

Chair of Software Engineering for Business Information Systems (sebis)  
Faculty of Informatics  
Technische Universität München  
[www.matthes.in.tum.de](http://www.matthes.in.tum.de)

## Motivation and Background

- Transition from Monolithic Systems to Microservice Architectures
- Documentation by Distributed Tracing
- Architecture Discovery

## Research Questions

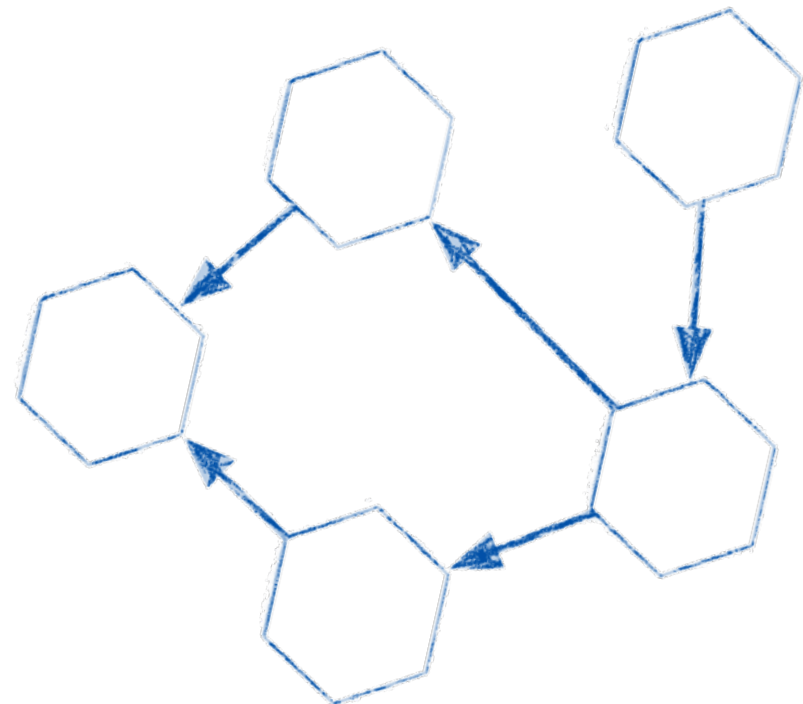
## Solution Approach

- Data Sources
- Test Environment

## Prototype Implementation

## Outlook

## Live Demo



### Monolithic System



- one system per product
- large complexity
- long development iterations
- difficult to scale
- hard to optimize

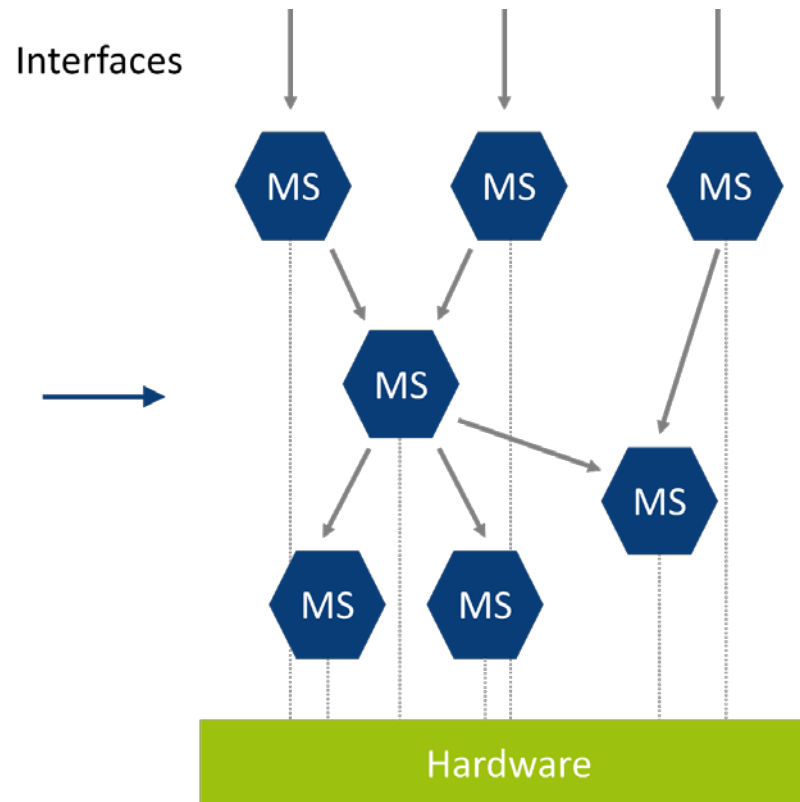
# Motivation and Background

## Microservice Architecture

### Monolithic System

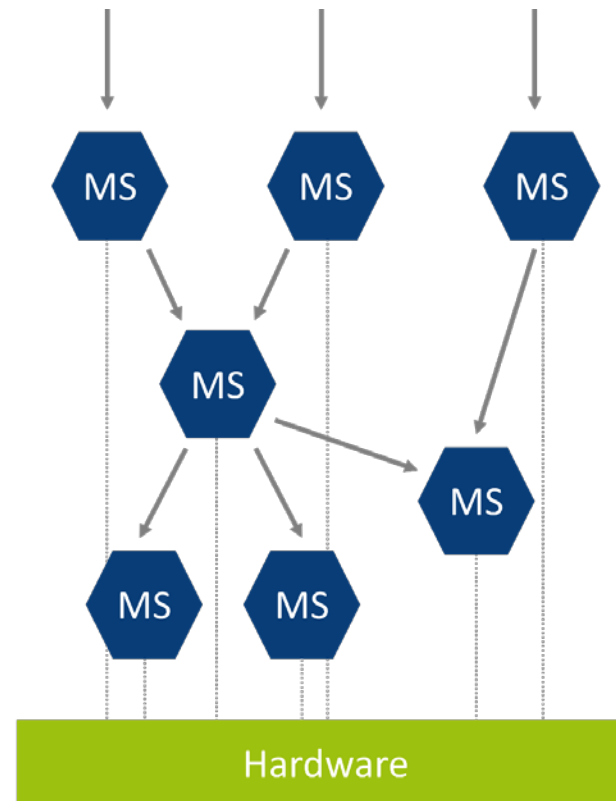


### Distributed Service Architecture



### Distributed Service Architecture

- low complexity
- agile development
- code reusability
- improved technology fit
- service replaceability



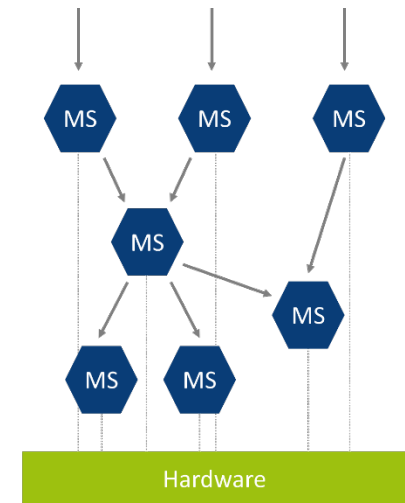
### Monolithic System



**inner  
complexity**



### Distributed Service Architecture



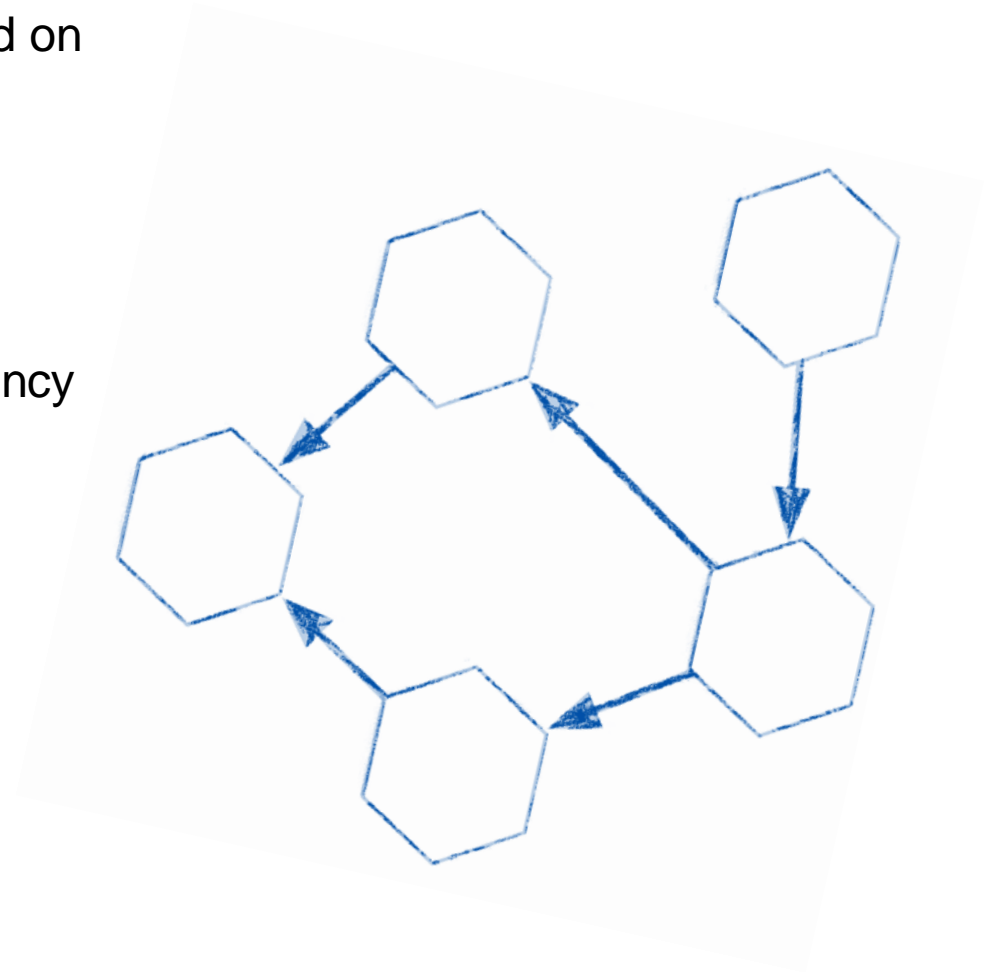
**outer  
complexity**

- Independent development of Microservices
    - Diverging versions
  - Microservice utilized from a different Product
    - Gap between developer and user
  - Multiple Products using the same Microservice
    - Differing requirements
  - Alignment with planned architecture
- In order to keep track: **Distributed Tracing**

Reconstruct architecture based on distributed tracing spans

### Design Goals

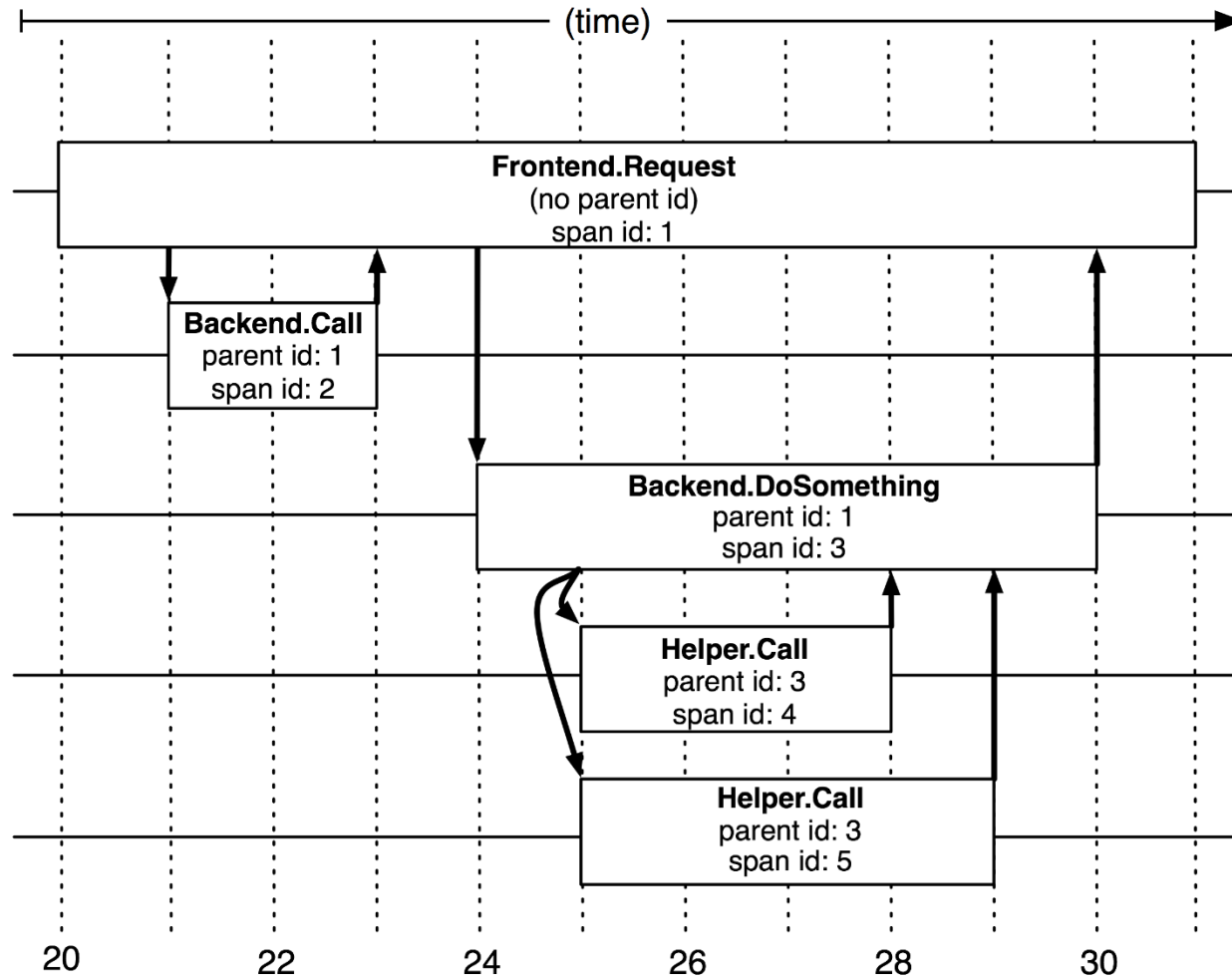
- Application Level Transparency
- Low overhead
- Scalability
- Realtime data availability





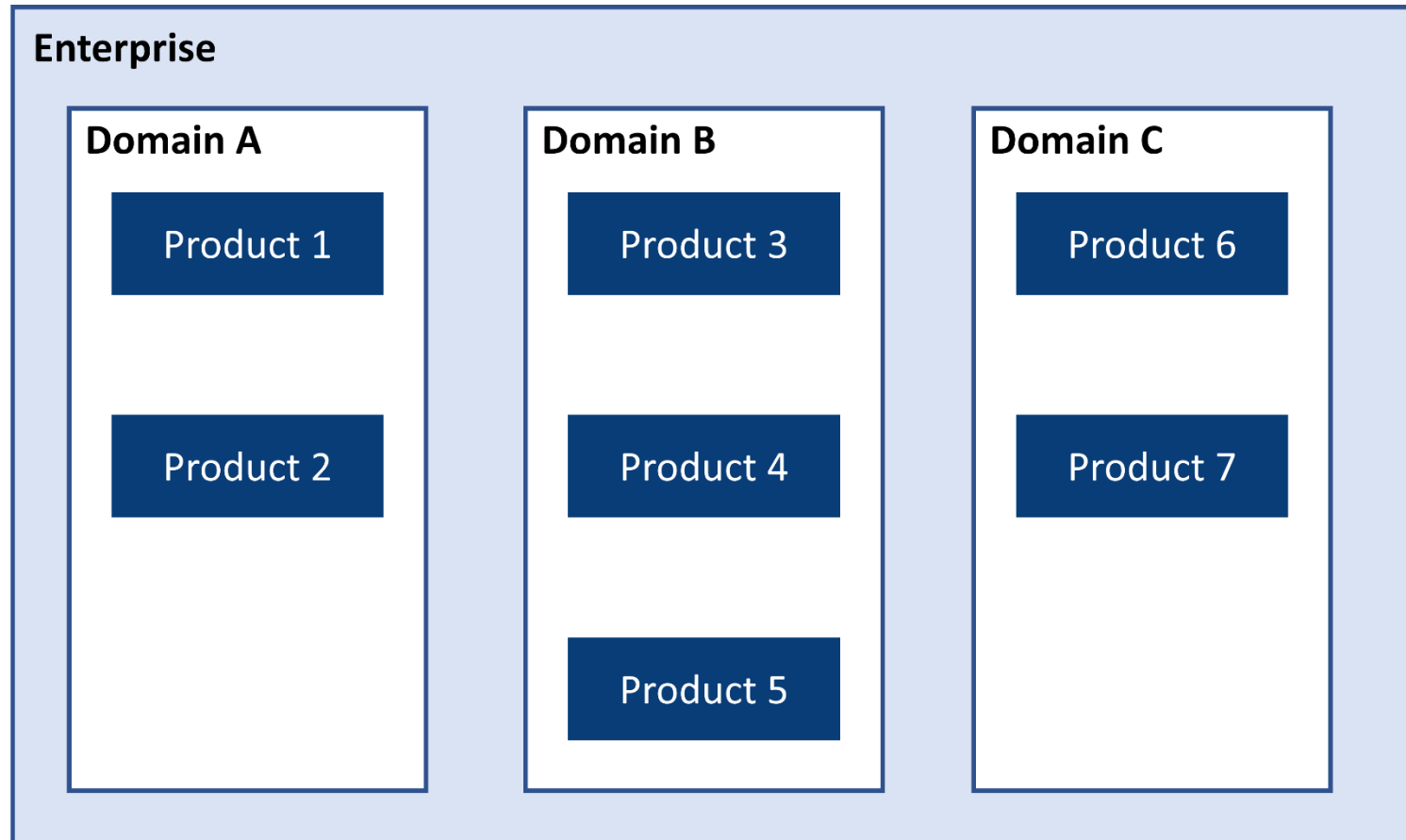
# Motivation and Background

## Distributed Tracing



# Motivation and Background

## Architecture Discovery: Context

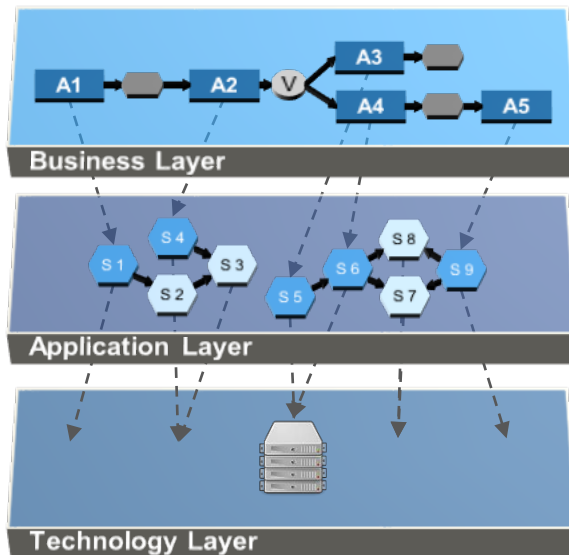


# Motivation and Background

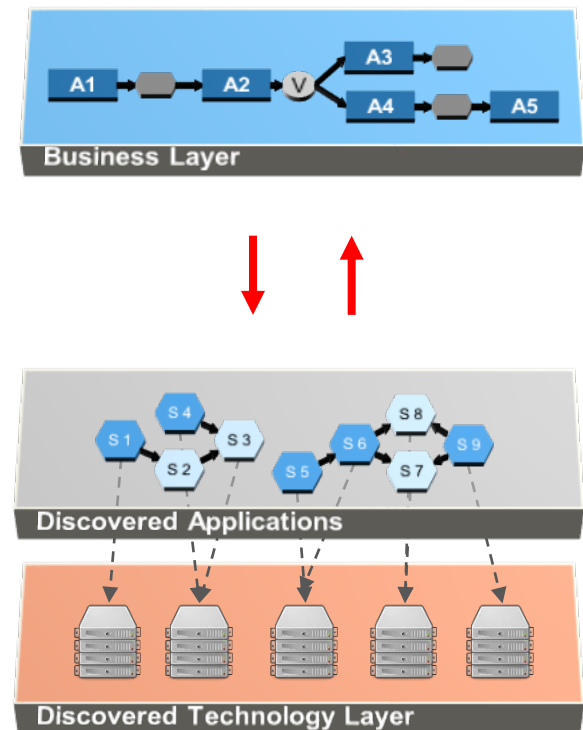
## Architecture Discovery: Overview

Realtime discovery of

- architecture components
- inter- and intra-layer dependencies
- revisions



Architecture  
Discovery



# Motivation and Background

## Architecture Discovery: Results





	P1	A1	A2	A3	A4	A5	S1	S2	S3	DEUTSCHEBAHN-MOBILITY-SERVICE	I4	I5	I6	I7	I8	I9	I10	I11	I12	H1	H2	H3	H4	
Route Booking	P1	-	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Book Route	A1	-		X		X																		
Login	A2		-			X														X		X		
Logout	A3			-																				
Search Route	A4	X			-				X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Select Provider	A5				X	-																		
ACCOUNTING-CORE-SERVICE	S1						-															X		
BUSINESS-CORE-SERVICE	S2							-														X		
DEUTSCHEBAHN-MOBILITY-SERVICE	S3								-														X	X
DRIVENOW-MOBILITY-SERVICE	S4																						X	X
EUREKA-SERVICE	S5																				X			
MAPS-HELPER-SERVICE	S6																						X	X
<b>TRAVELCOMPANION-MOBILITY-SERVICE</b>	<b>S7</b>								X	X													X	X
ZUUL-SERVICE	S8						X	X												X		X	X	X
ACCOUNTING-CORE-SERVICE (10.0.2.110:5000)	I1																					X		
BUSINESS-CORE-SERVICE (10.0.2.110:5000)	I2																						X	
DEUTSCHEBAHN-MOBILITY-SERVICE (10.0.2.110:5000)	I3																							X
DEUTSCHEBAHN-MOBILITY-SERVICE (10.0.2.110:5000)	I4																							X
DRIVENOW-MOBILITY-SERVICE (10.0.2.121:6003)	I5																						X	
DRIVENOW-MOBILITY-SERVICE (10.0.2.122:6003)	I6																							X
EUREKA-SERVICE (10.0.2.100:8761)	I7																				X			
MAPS-HELPER-SERVICE (10.0.2.121:7000)	I8																						X	
MAPS-HELPER-SERVICE (10.0.2.122:7000)	I9																							X
TRAVELCOMPANION-MOBILITY-SERVICE (10.0.2.110:5000)	I10																						X	
TRAVELCOMPANION-MOBILITY-SERVICE (10.0.2.110:5000)	I11																							X
ZUUL-SERVICE (10.0.2.110:9000)	I12																							X
10.0.2.100	H1																							
10.0.2.110	H2																							
10.0.2.121	H3																							
10.0.2.122	H4																							

**Relation Details**

Service **TRAVELCOMPANION-MOBILITY-SERVICE** uses Service **DEUTSCHEBAHN-MOBILITY-SERVICE**

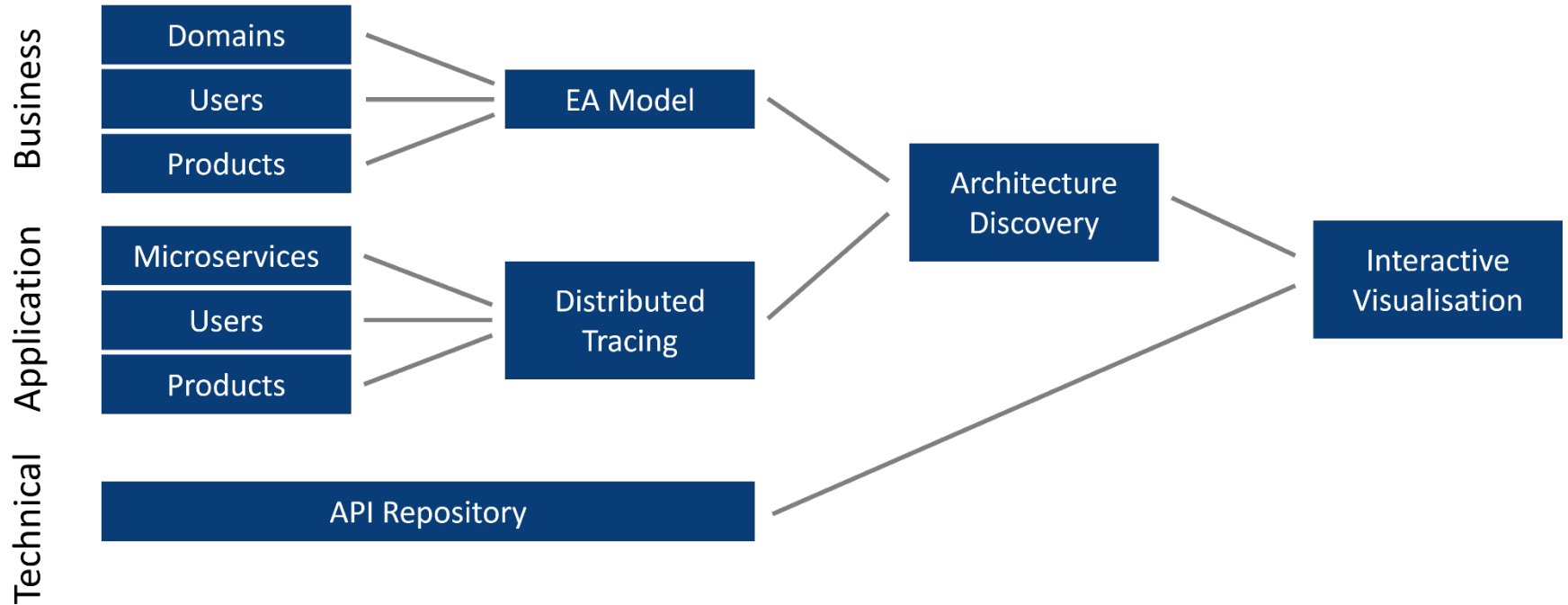
**Annotations**

- mvc.controller.class** DeutscheBahnController
- mvc.controller.method** findDistance
- spring.instance\_id** 10.0.2.121:deutschebahn-mobility-service:6003
- http.path** /getroutes
- http.url** http://localhost:6003/getroutes?origin=1&destination=3
- First seen** 2017-11-08 12:31:28
- Asynchronous** true
- http.method** GET
- http.host** localhost

-  How to effectively visualize the relations between Business Entities and Microservices?
-  Which criteria are suitable to position Microservices meaningful in a directed Graph?
-  How can the results of a root cause analysis and its impact be effectively displayed?
-  Which methods are suitable for displaying larger Microservice Networks in a manageable manner?  
(clustering, filtering, etc.)

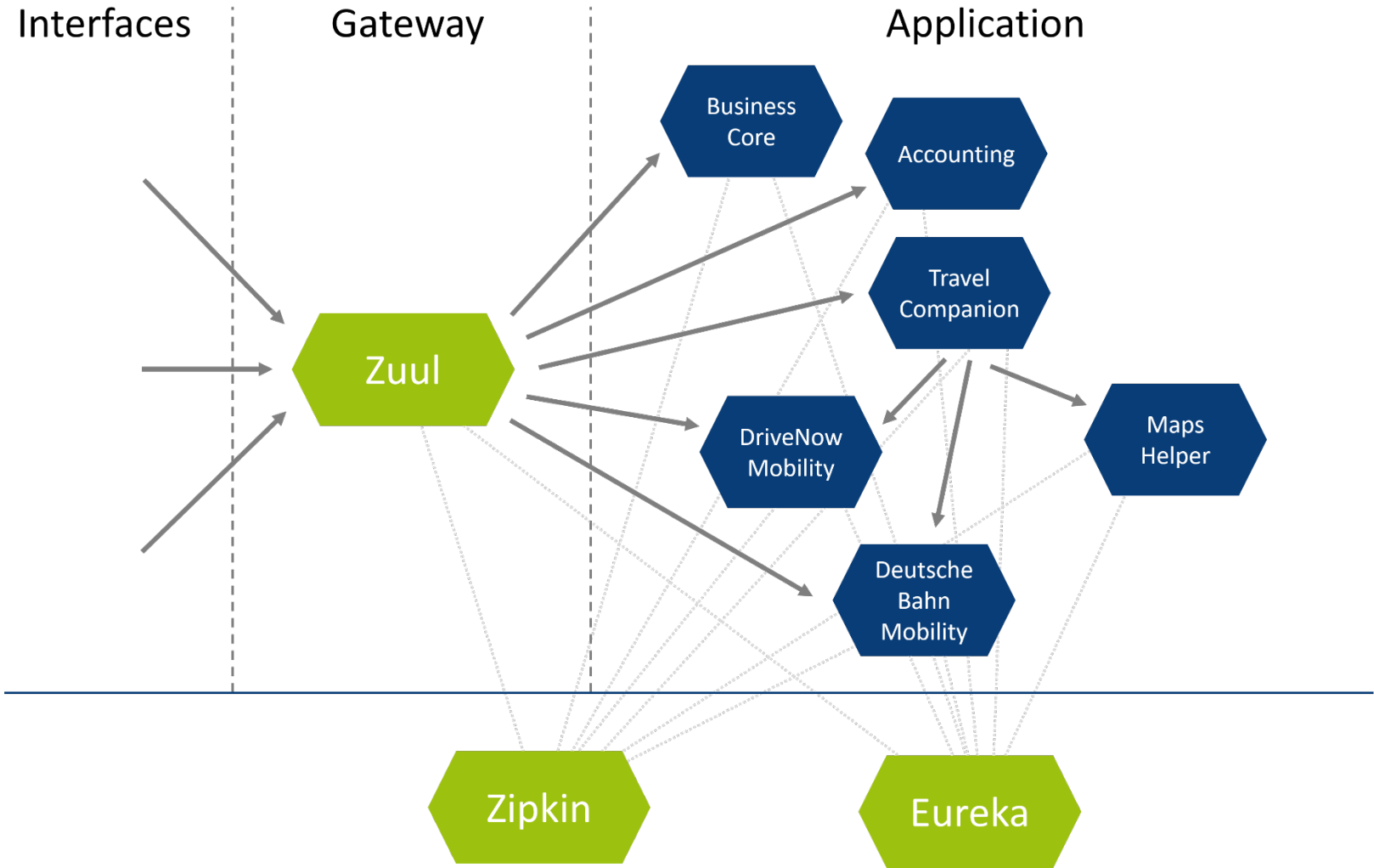
# Solution Approach

## Data Sources



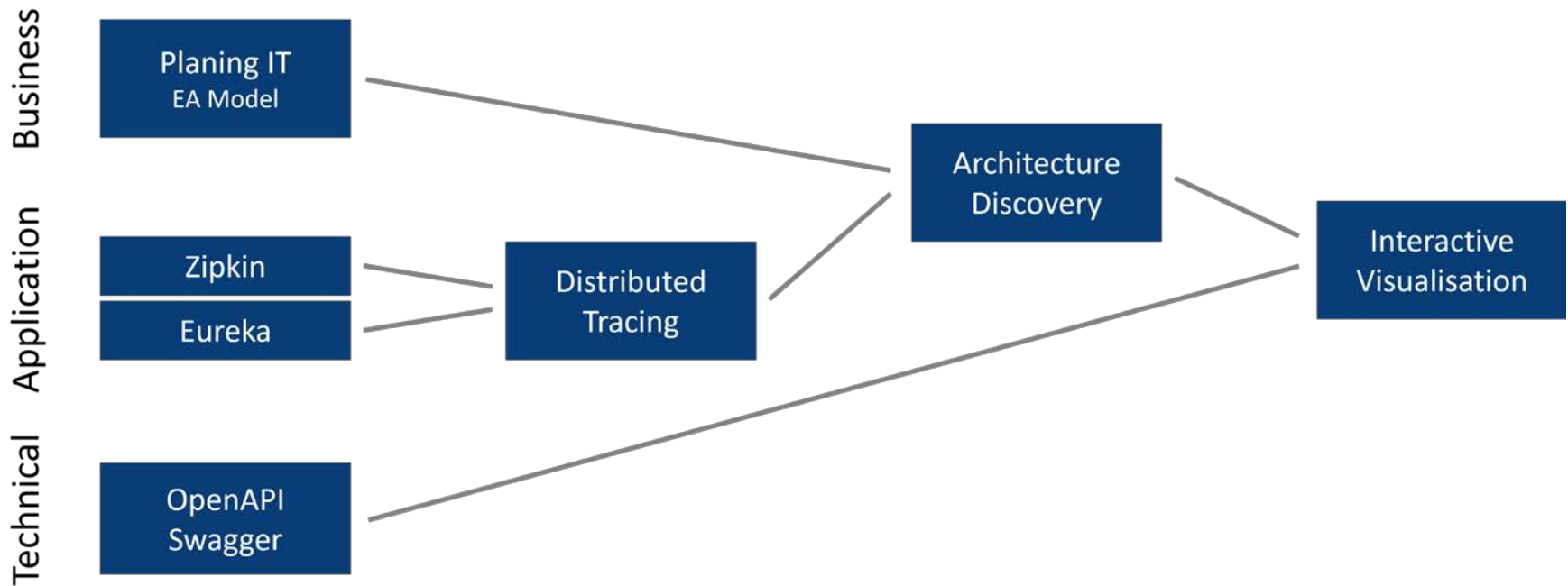
# Solution Approach

## Test Environment



# Solution Approach

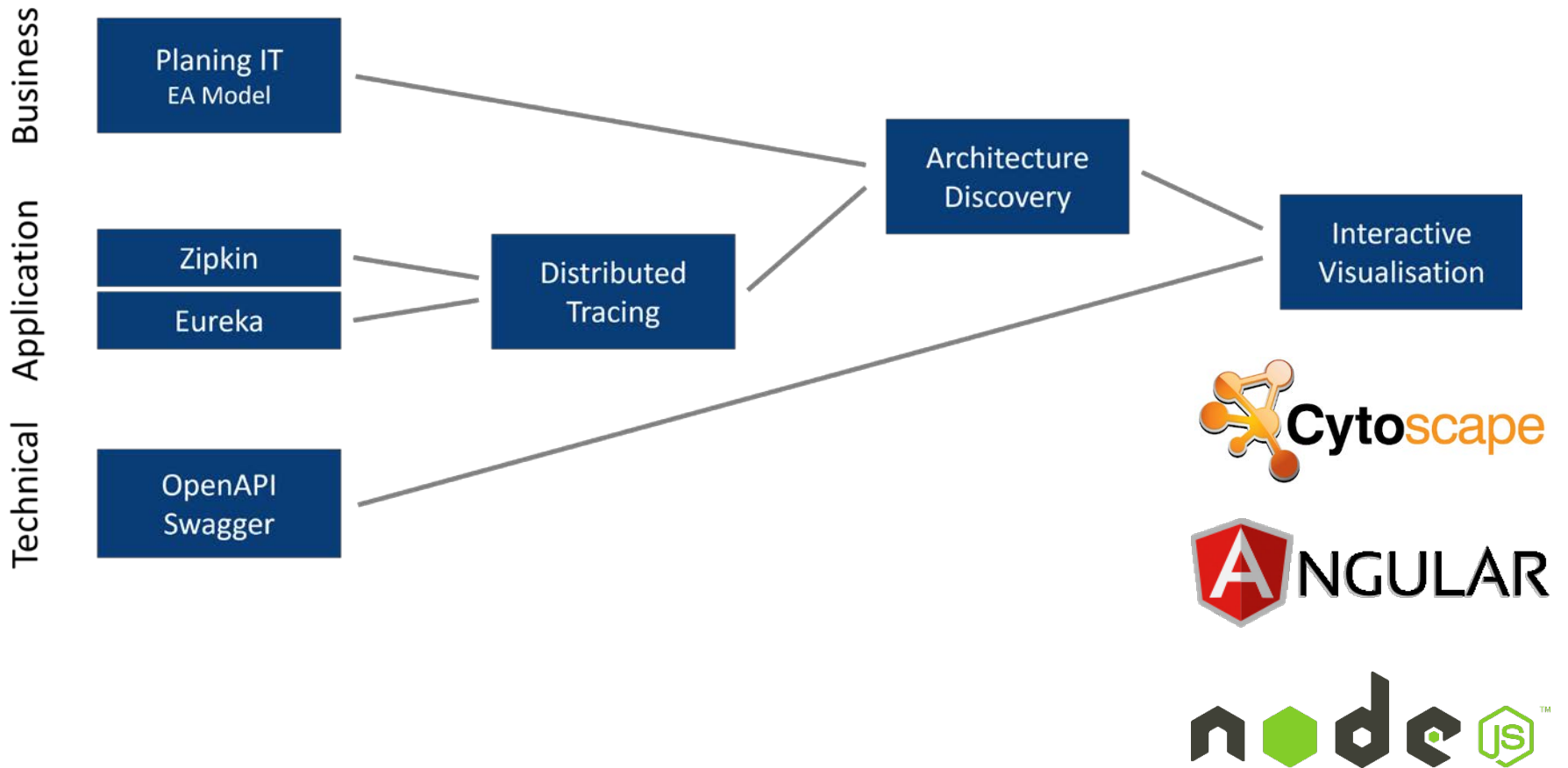
## Data Sources





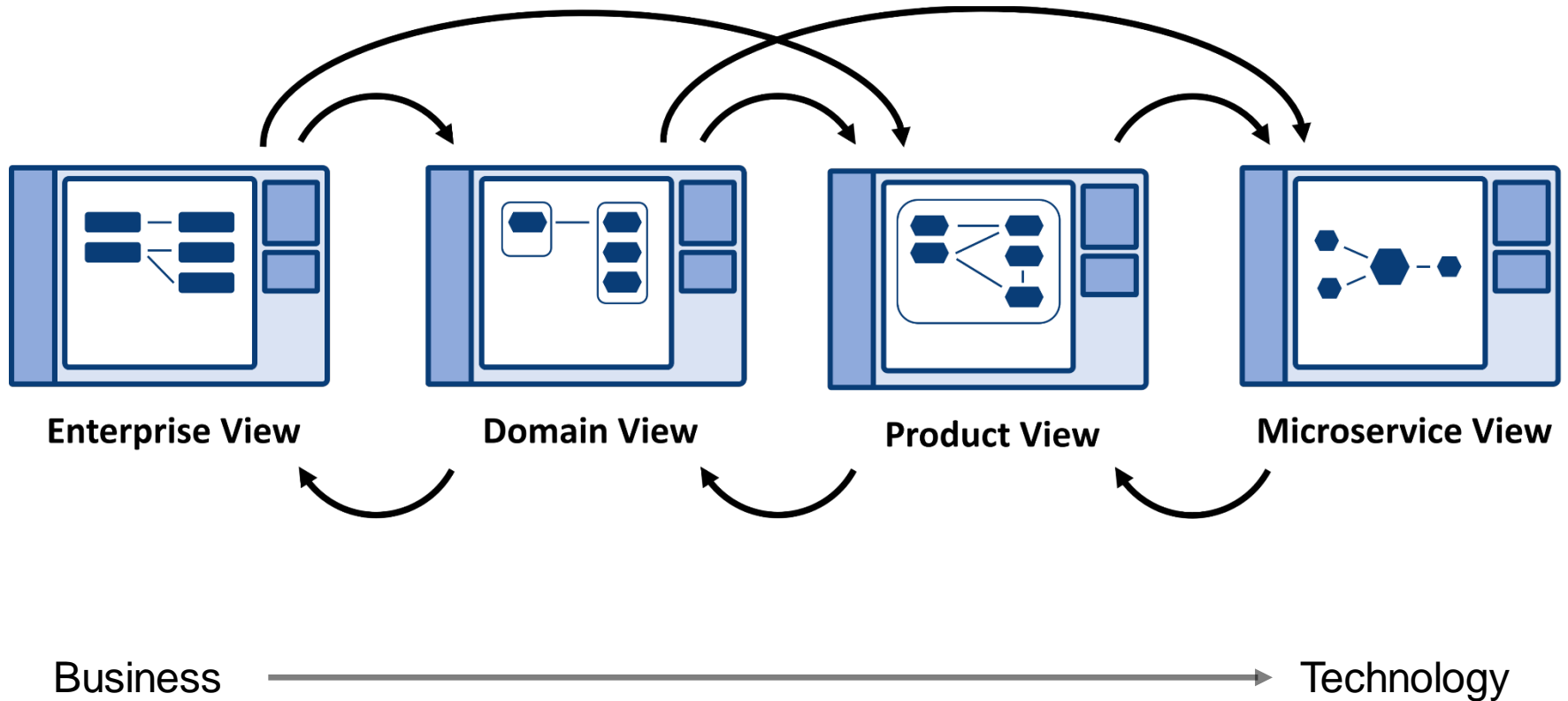
# Solution Approach

## Data Sources

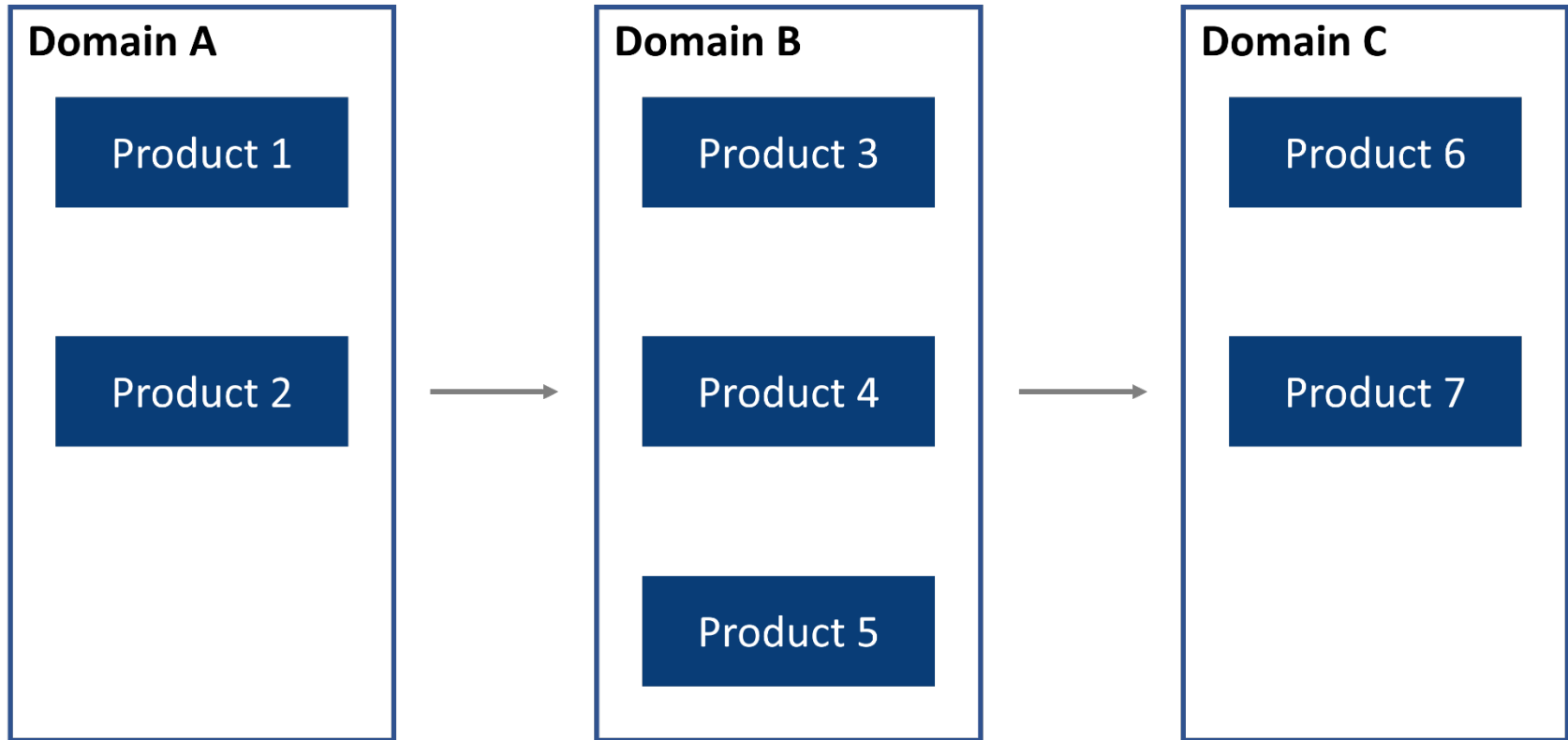
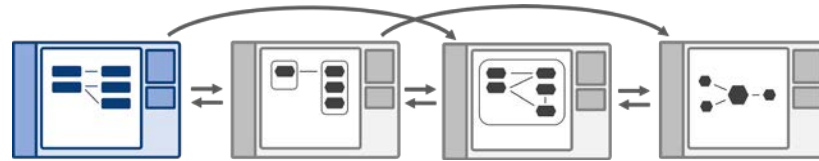


# Prototype Implementation

## Available Views



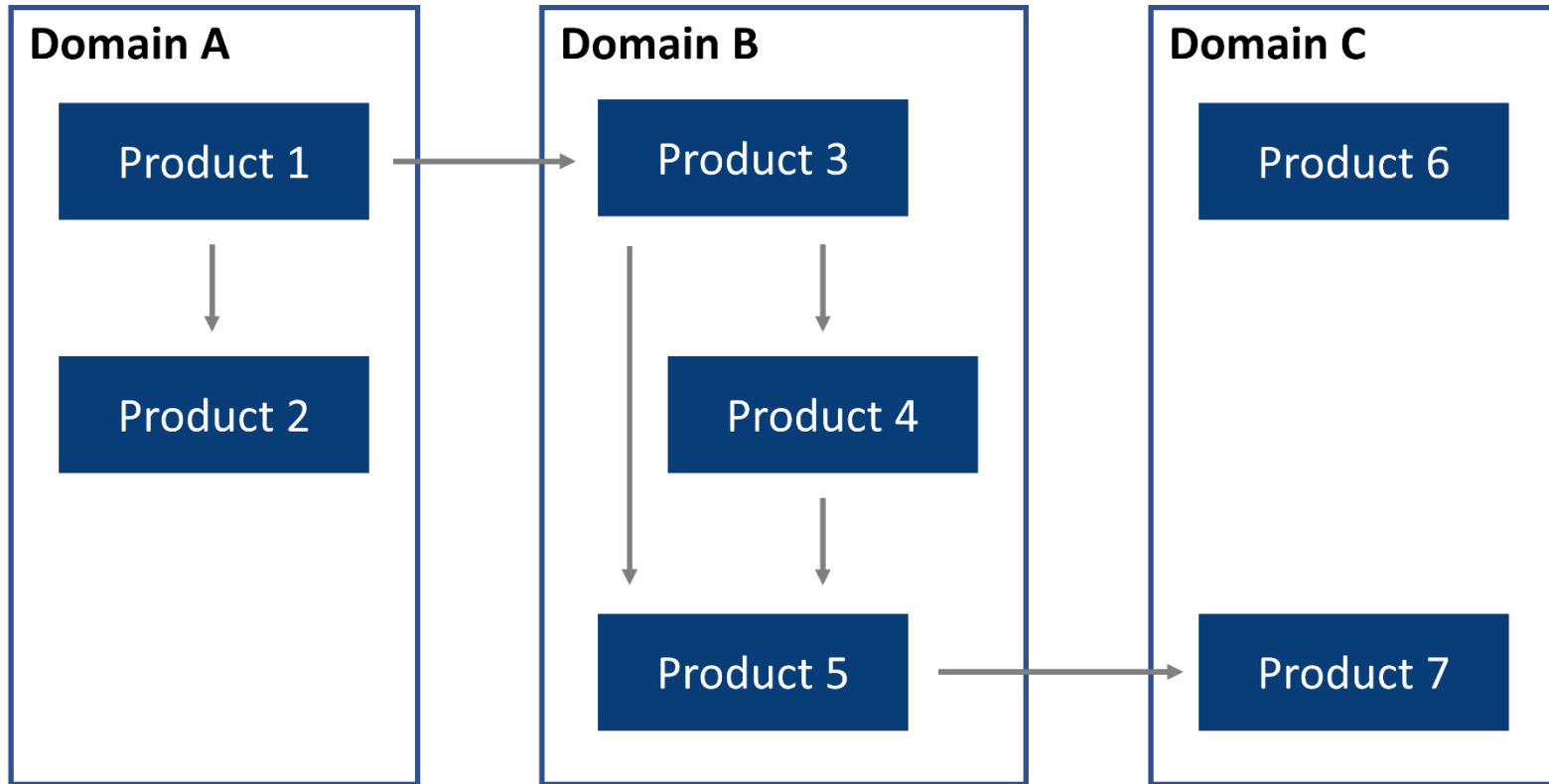
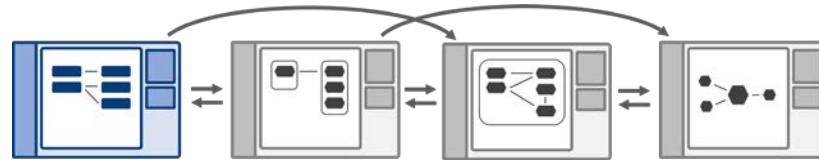
# Prototype Implementation Enterprise View



Useful for **Enterprise Architect**  
Switch between **domain-centered** and product-centered view

# Prototype Implementation

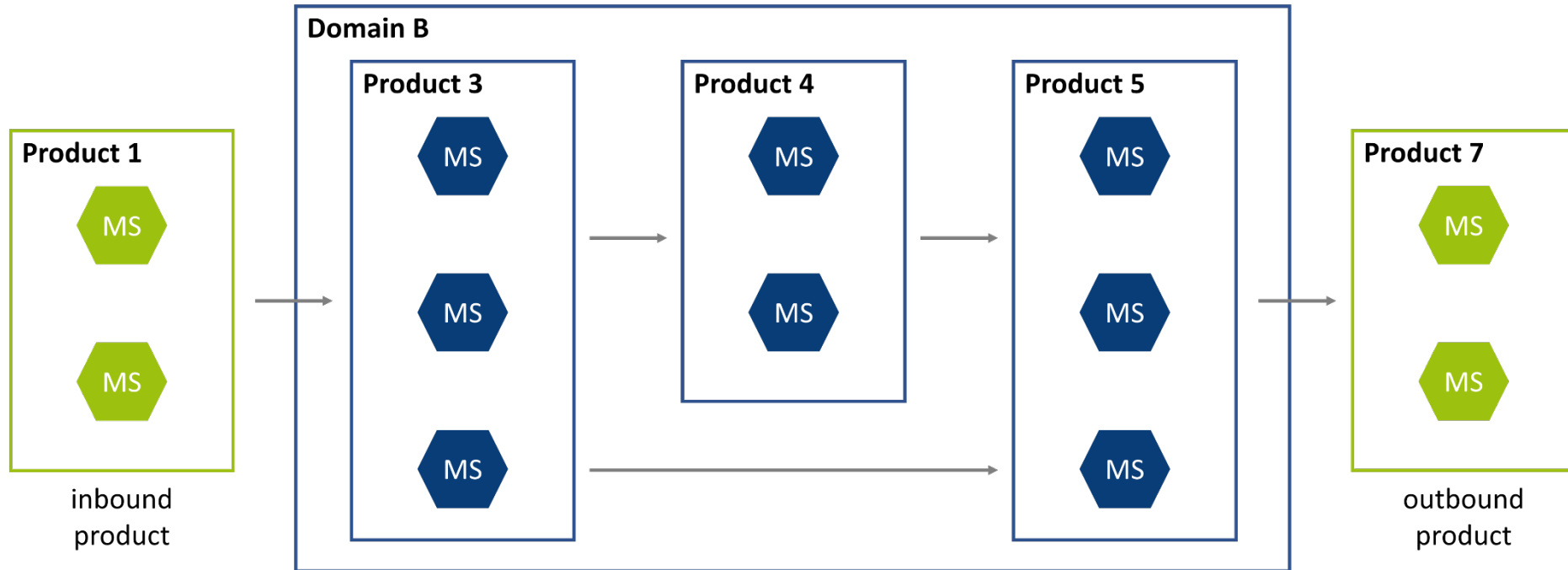
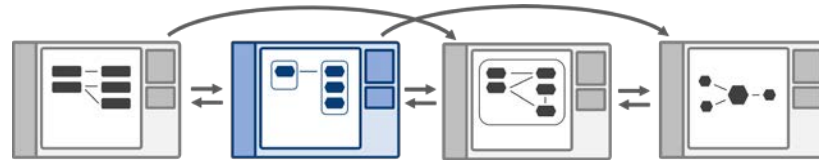
## Enterprise View



Useful for **Enterprise Architect**  
Switch between domain-centered and **product-centered** view

# Prototype Implementation

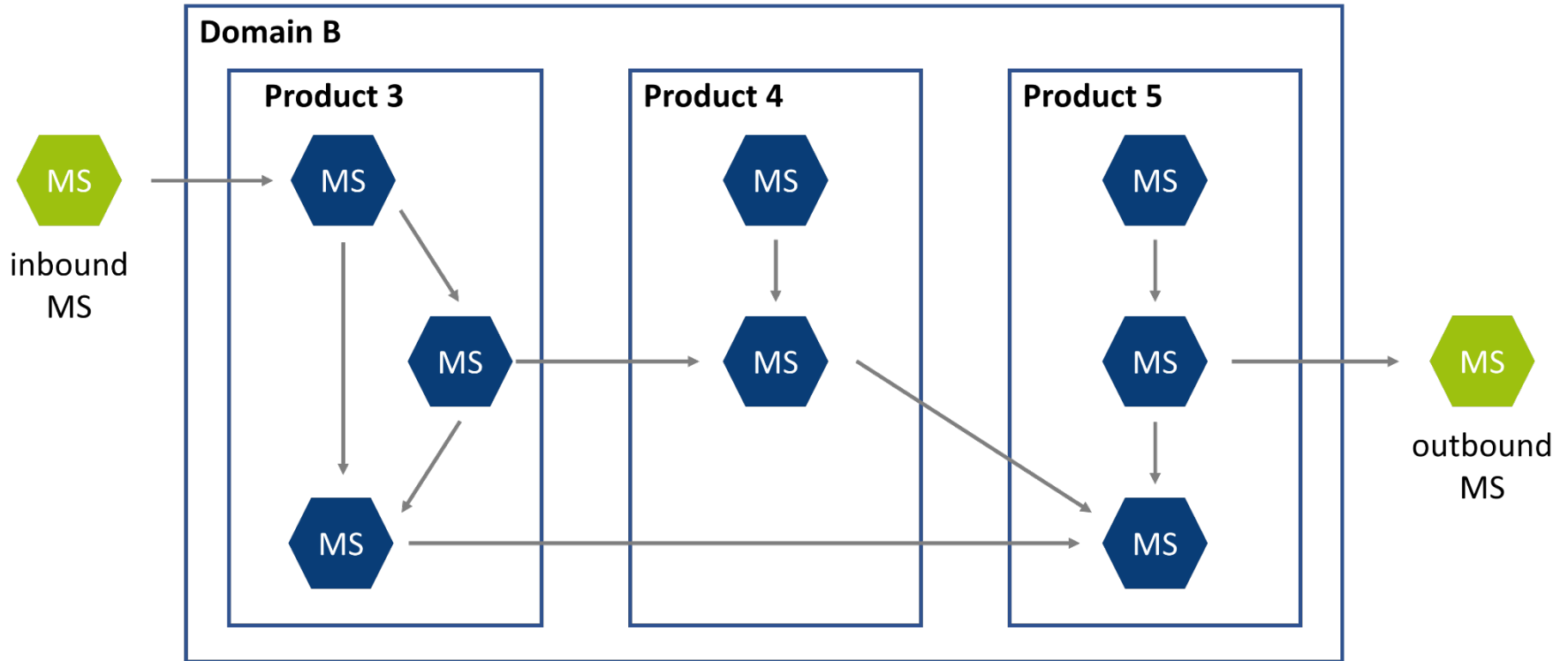
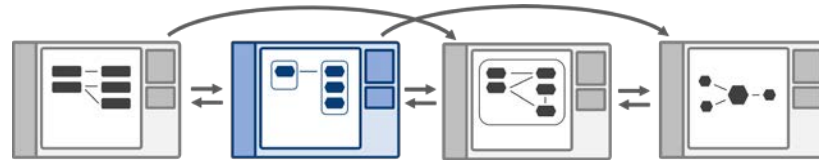
## Domain View



Useful for  
Switch between **Domain Owner** **product-centered** and microservice-centered view

# Prototype Implementation

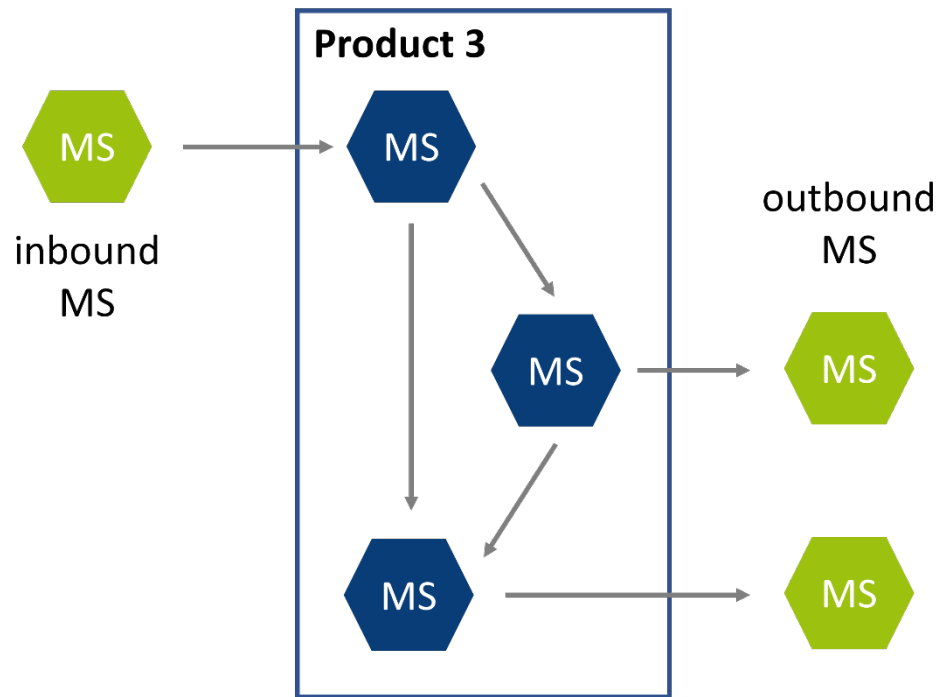
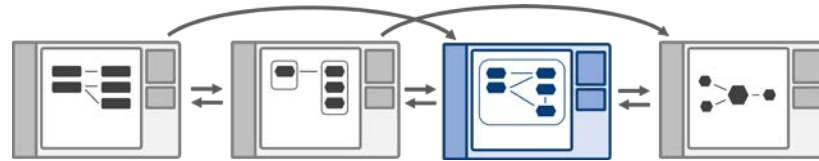
## Domain View



Useful for **Domain Owner**  
Switch between product-centered and **microservice-centered** view

# Prototype Implementation

## Product View

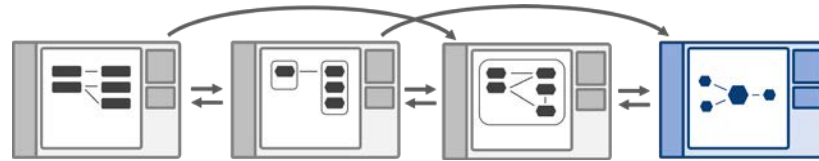


Useful for

**Software Architect, Product Owner**

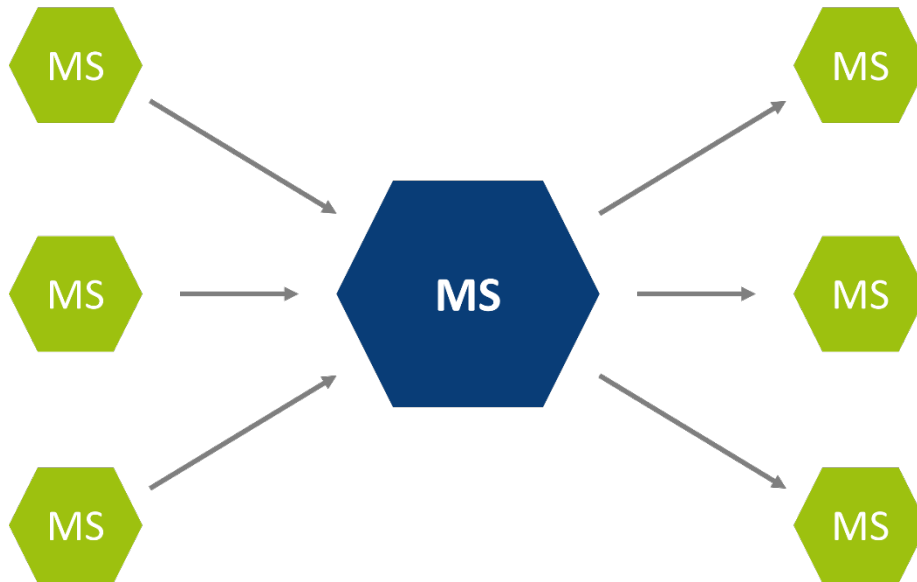
# Prototype Implementation

## Microservice View



**inbound**

**outbound**



Including technical data:

- Microservice Instances
- Microservice API

Possibly also:

- Microservice life data

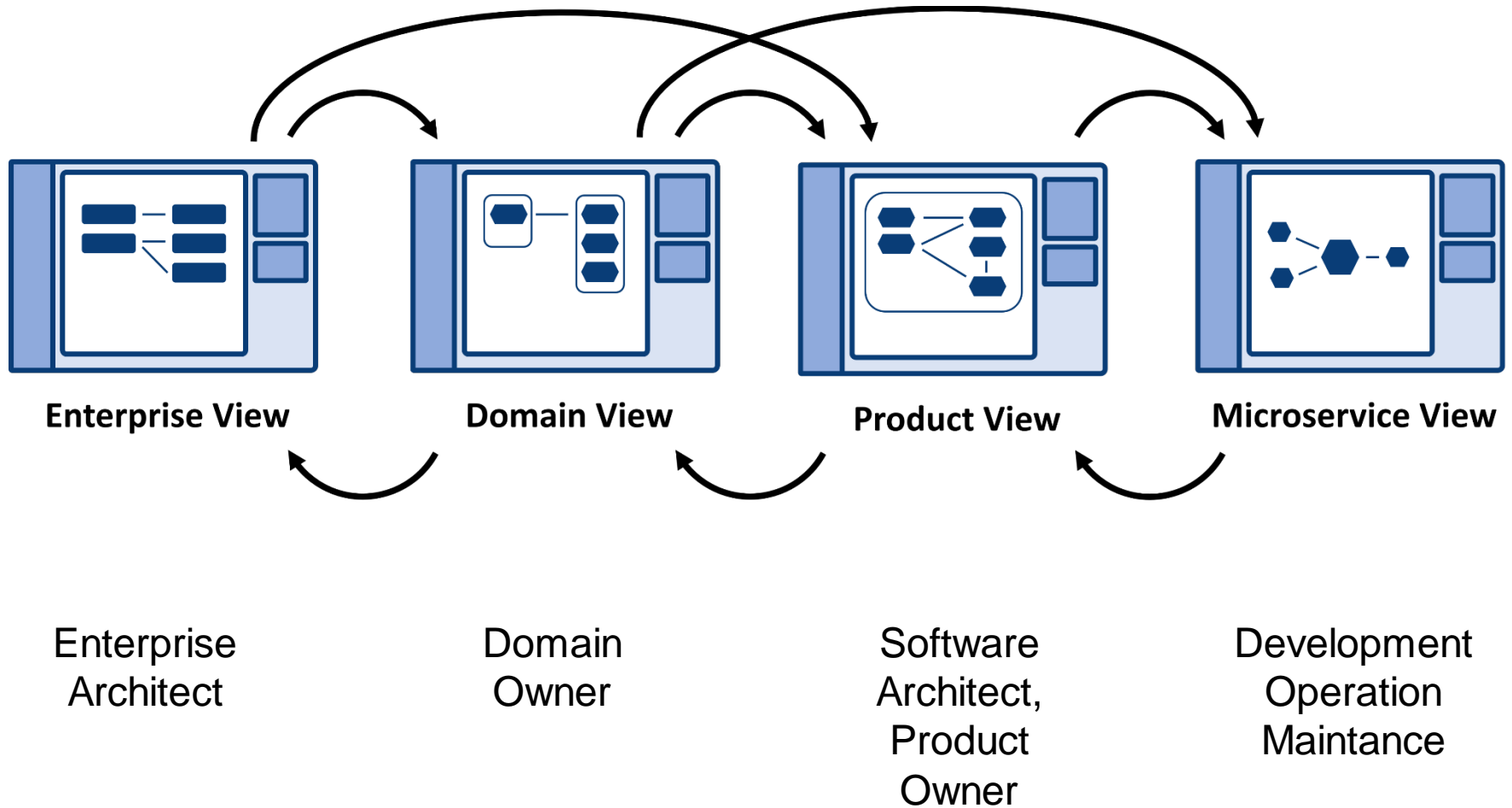
Useful for

**Development, Operation, Maintenance**

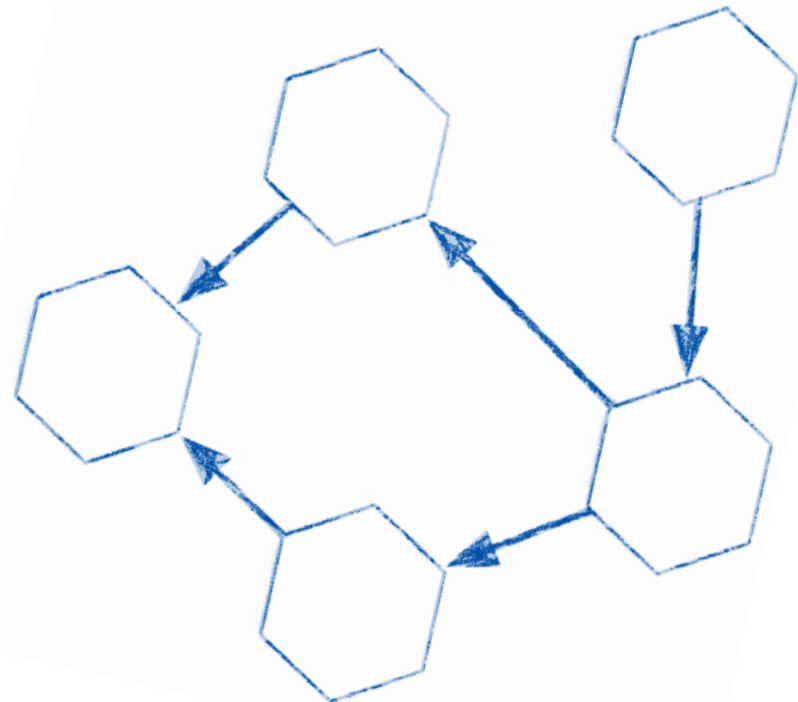


# Prototype Implementation

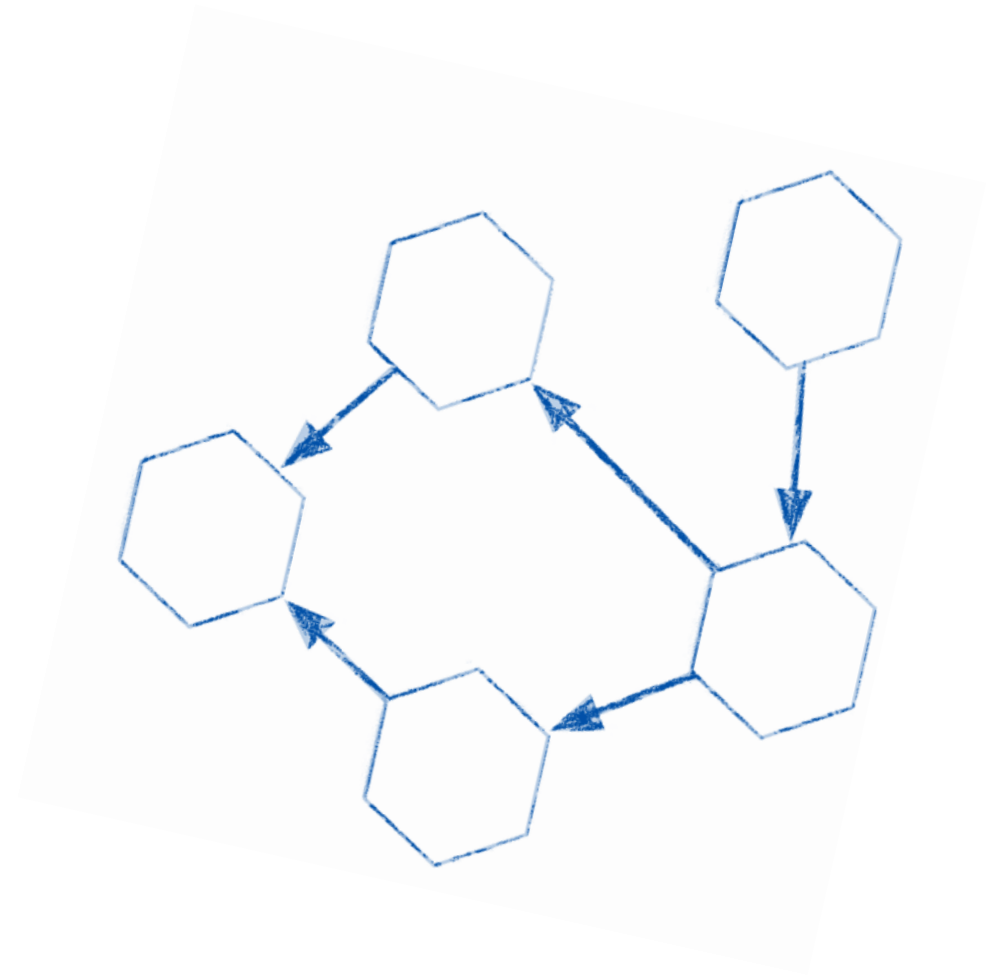
## Available Views



- Enhanced relation data
  - Weights to visualize flow
  - Synchronous / asynchronous communication
- Enhanced business information
  - Estimate business value of Microservice
  - Identify success critical Microservices
- Full root cause analysis integration



# Live Demo





B. Sc.

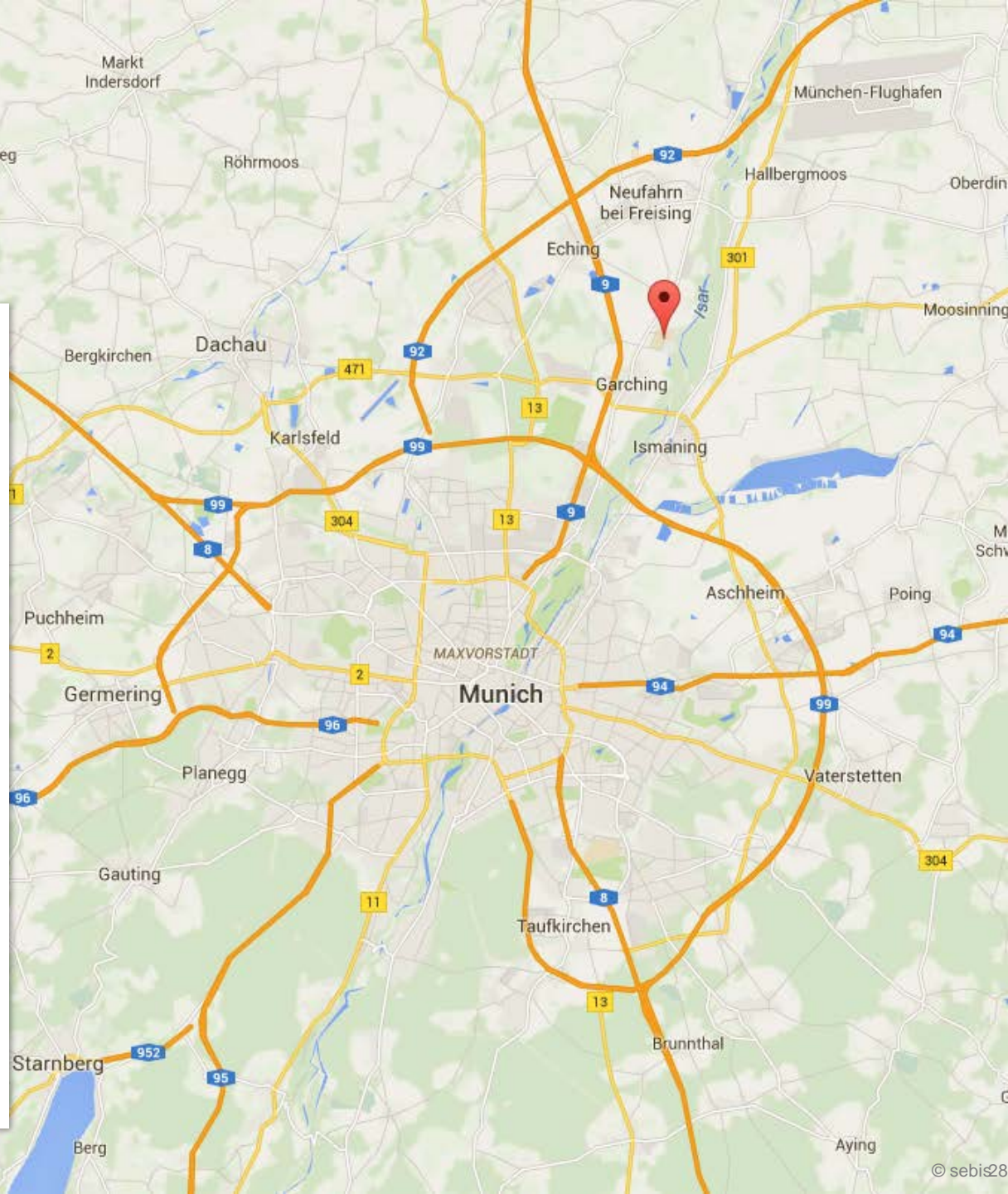
**Daniel Graf Hoyos**

Master Student Information Systems

Technische Universität München  
Faculty of Informatics  
Chair of Software Engineering for  
Business Information Systems

Boltzmannstraße 3  
85748 Garching bei München

[www.matthes.in.tum.de](http://www.matthes.in.tum.de)  
daniel.hoyos@tum.de

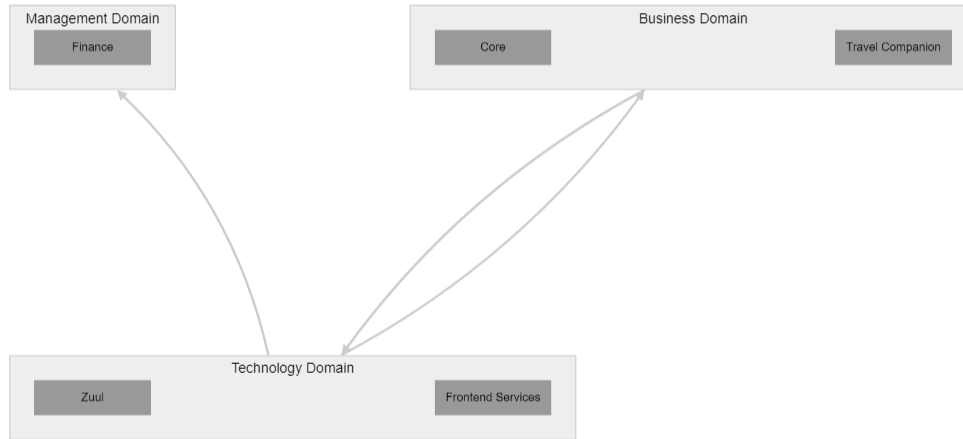


Architecture Explorer

- Dashboard
- Dependency Model
- Architecture Livelog
- Click Sequence
- Modelling

Home / Home / Microlyze Dashboard

## Enterprise View



2010/11/29

2011/11/29

## Settings

Aggregate Relations on *Domain level*

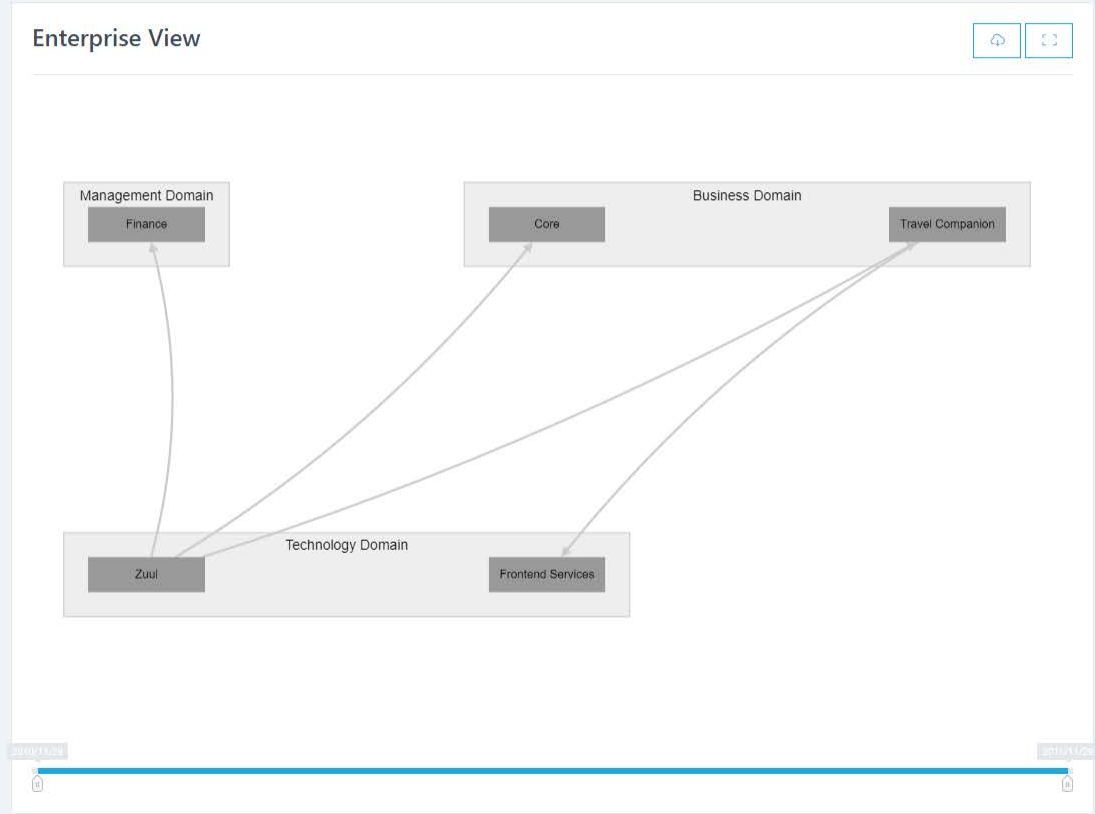
## Recent Problems

- Delivery Service **5 minutes ago**
- Mail Service **2 hours ago**
- Mail Service **5 days ago**
- Zuul Service **12 days ago**

Architecture Explorer

- Dashboard
- Dependency Model
- Architecture LiveLog
- Click Sequence
- Modelling

Home / Home / Microlyze Dashboard



### Settings

Aggregate Relations on Domain level

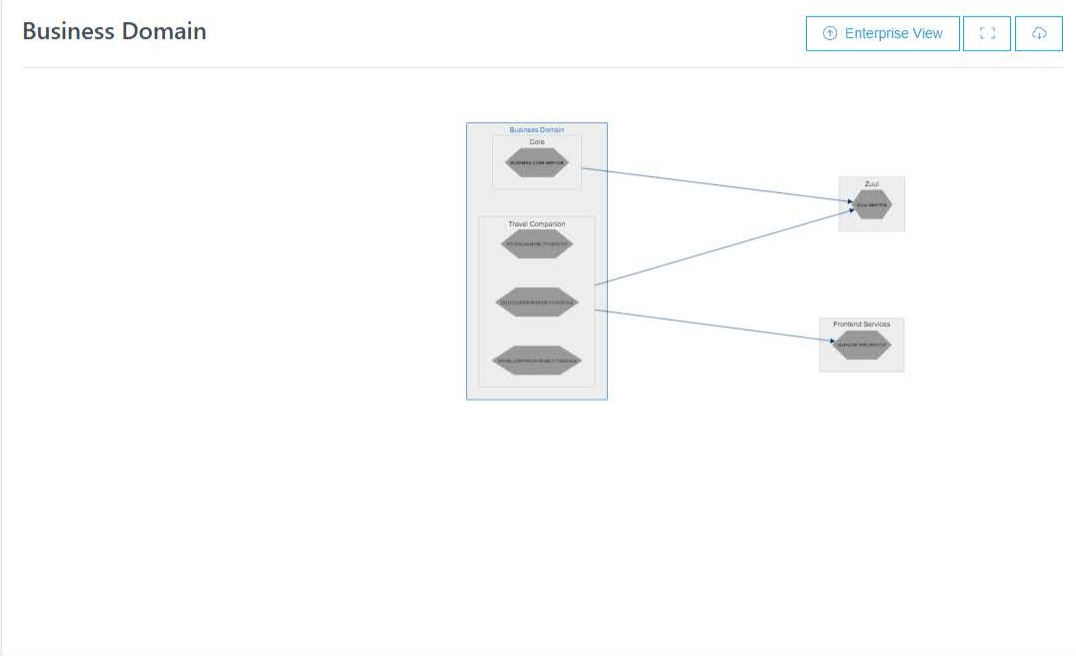
### Recent Problems

- Delivery Service **6 minutes ago**
- Mail Service **2 hours ago**
- Mail Service **5 days ago**
- Zuul Service **12 days ago**

Architecture Explorer

- Dashboard
- Dependency Model
- Architecture Livelog
- Click Sequence
- Modelling

Home / Home / Component View



### Settings

Aggregate Relations on Product level

### Entities

**Products**

- Core
- Travel Companion

**External Products**

- Zuul
- Frontend Services

**Services**

- BUSINESS-CORE-SERVICE
- DRIVENOW-MOBILITY-SERVICE
- DEUTSCHEBAHN-MOBILITY-SERVICE
- TRAVELCOMPANION-MOBILITY-SERVICE

**External Services**

- ZUUL-SERVICE
- MAPS-HELPER-SERVICE

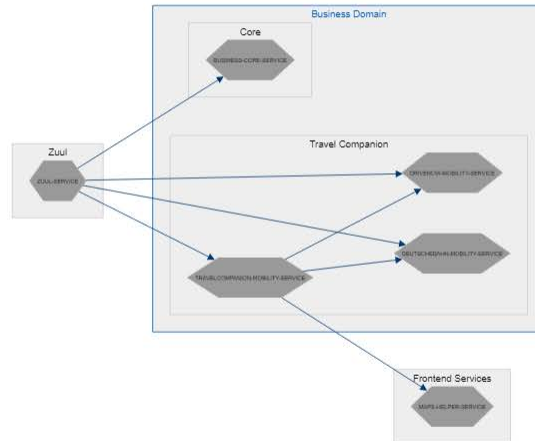
Architecture Explorer

- Dashboard
- Dependency Model
- Architecture Livelog
- Click Sequence
- Modelling

Home / Home / Component View

## Business Domain

Enterprise View



## Settings

Aggregate Relations on Product level

## Entities

### Products

Core  
Travel Companion

### External Products

Zuul  
Frontend Services

### Services

BUSINESS-CORE-SERVICE  
DRIVENOW-MOBILITY-SERVICE  
DEUTSCHEBAHN-MOBILITY-SERVICE  
TRAVELCOMPANION-MOBILITY-SERVICE

### External Services

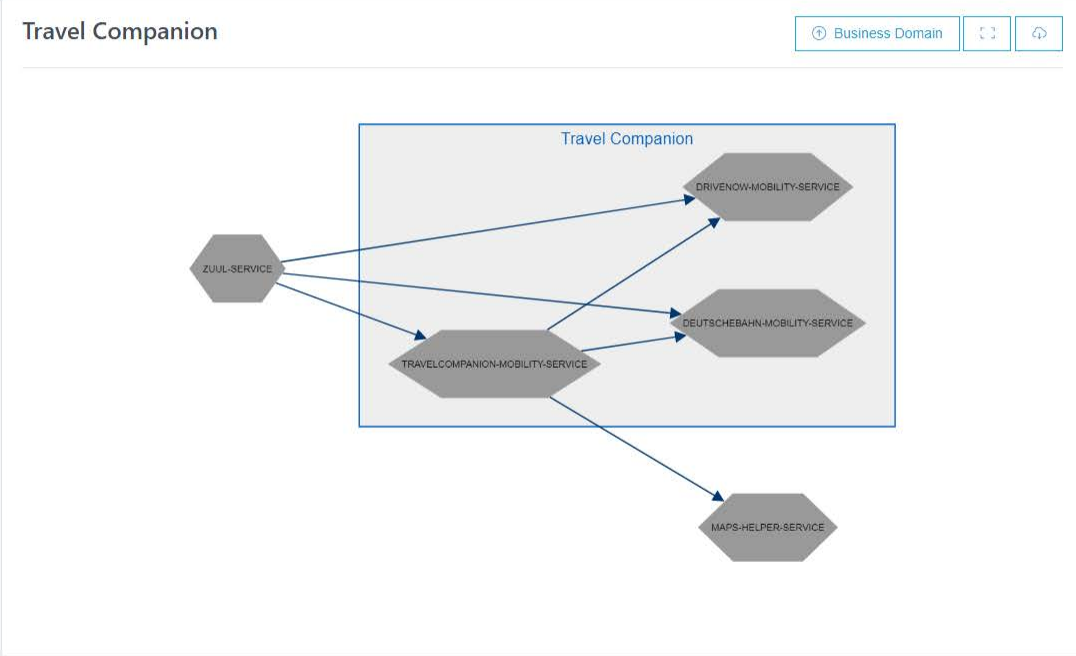
ZUUL-SERVICE  
MAPS-HELPER-SERVICE



Architecture Explorer

- Dashboard
- Dependency Model
- Architecture Livelog
- Click Sequence
- Modelling

Home / Home / Component View



### Entities

**Parent**  
Business Domain

---

**Microservices**  
DRIVENOW-MOBILITY-SERVICE  
DEUTSCHEBAHN-MOBILITY-SERVICE  
TRAVELCOMPANION-MOBILITY-SERVICE

---

**External Services**  
ZUUL-SERVICE  
MAPS-HELPER-SERVICE

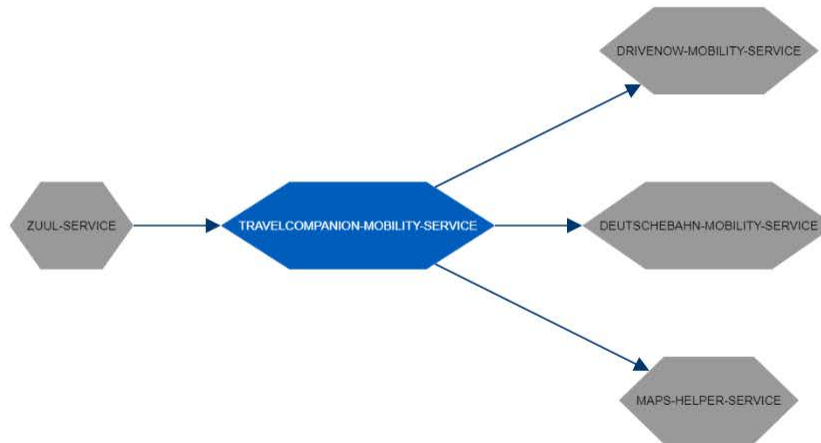
Architecture Explorer

- Dashboard
- Dependency Model
- Architecture Livelog
- Click Sequence
- Modelling

Home / Home / Component View

## TRAVELCOMPANION-MOBILITY-SERVICE

Travel Companion



### Entities

#### Parents

Travel Companion  
Business Domain

#### Inbound Services

ZUUL-SERVICE

#### Outbound Services

DRIVENOW-MOBILITY-SERVICE  
DEUTSCHEBAHN-MOBILITY-SERVICE  
MAPS-HELPER-SERVICE

### API

```
swagger : "2.0"  
+ info : Object {"version":"1.0.0","title":"Swagger Petstore","description":  
+ host : "petstore.swagger.io"  
+ basePath : "/api"  
+ schemes : Array[1] ["http"]  
+ consumes : Array[1] ["application/json"]  
+ produces : Array[1] ["application/json"]  
+ paths : Object {"pets":{"get":{"description":"Returns all pets from the sy  
+ definitions : Object {"Pet":{"type":"object","allOf":[{"$ref":"#/definition:
```