Xebia

Software Development Done Right

#OreillySACON

TUTORIAL

Designing a resource model for a Public API

O'Reilly Software Architecture Conference

New York – February 26th , 2018

Tom Höfte, IT Architect, Xebia

Marco van der Linden, IT Architect, Xebia

Agenda – 3.5 hours

Introduction & API Domain discovery

~ 20 minutes

API Domain discovery

~ 45 minutes

REST Resource modeling

~ 20 minutes

Coffee Break

~ 30 minutes (3pm)

REST Resource modeling

~ 60 minutes

Other Topics & Conclusion

~ 20 minutes (5pm)

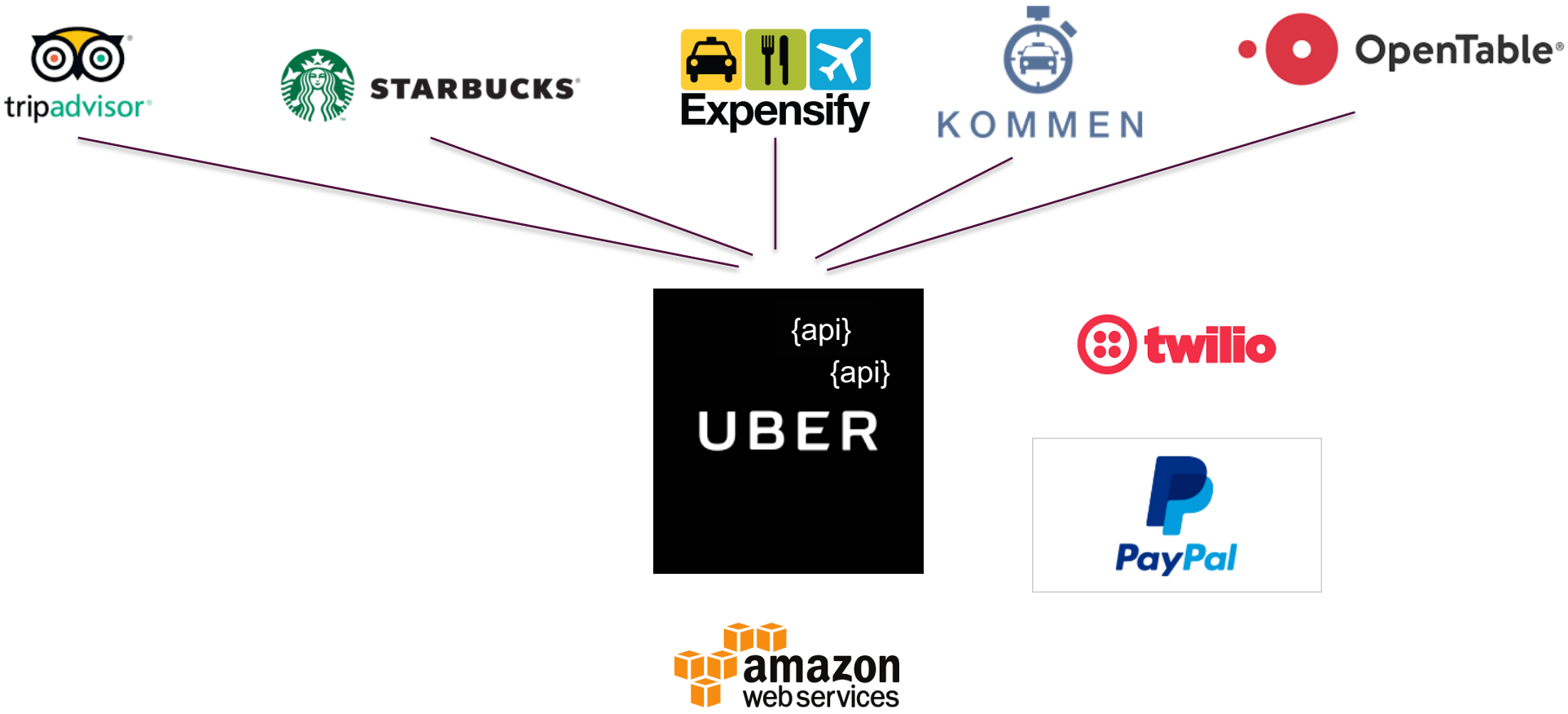
Xebia

Xebia

Software Development Done Right

Part 0 – Introduction

The API Economy

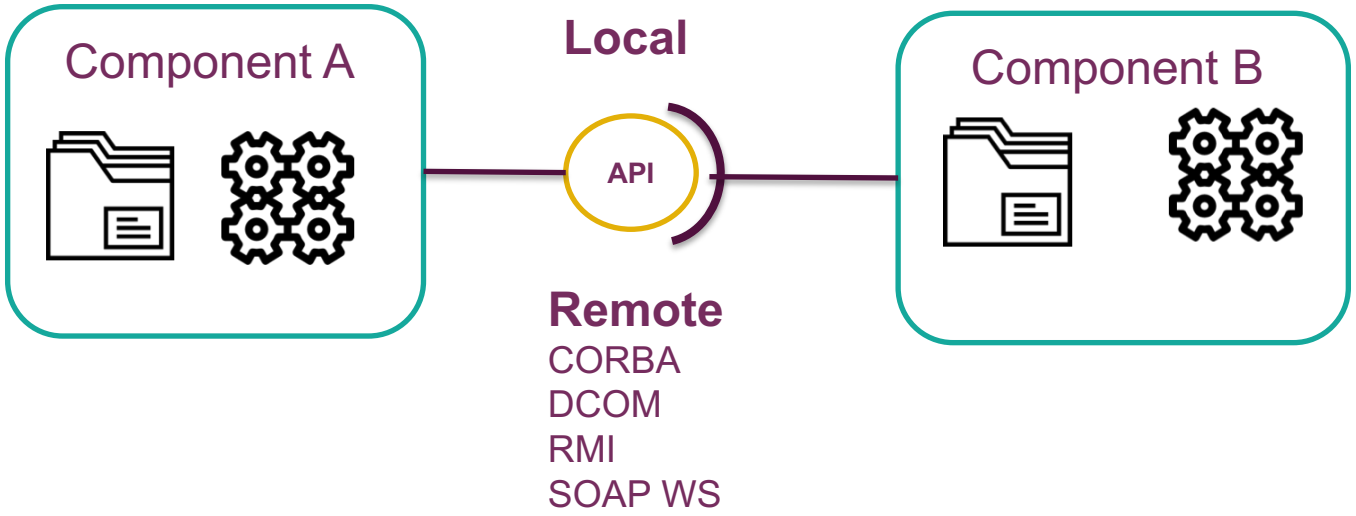


APIs allow companies to grow and extend businesses at unprecedented rates by sharing services with third parties – *source: Harvard Business Review (https://hbr.org/2015/01/the-strategic-value-of-apis)*

Xebia

Note: Slide picture inspired by an idea of Willem van Schieveen, Bol.com

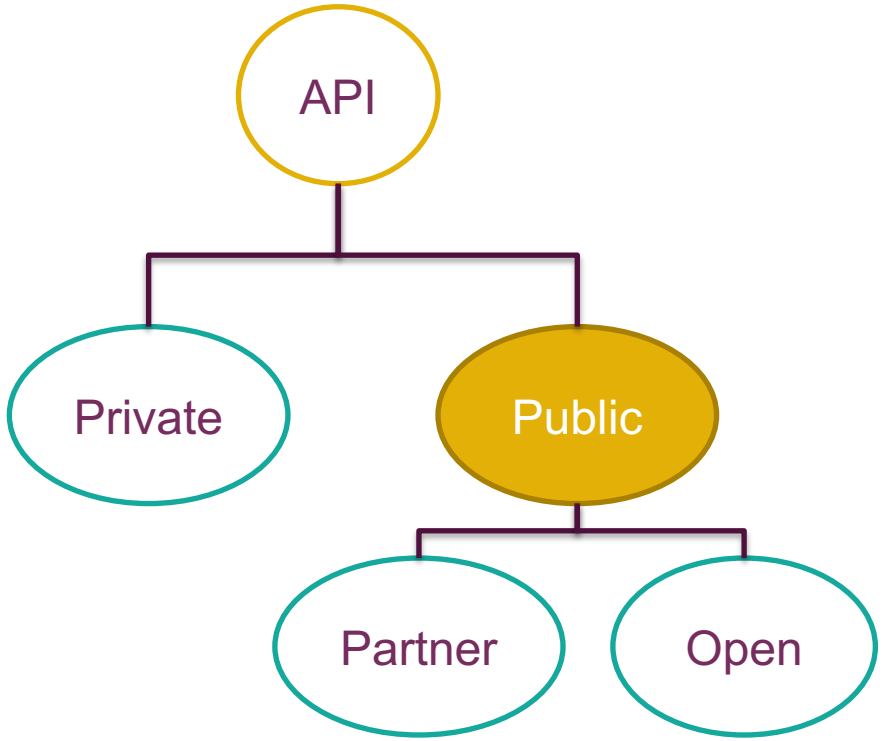
What we, ***techies***, used to think an API is



With the rise of web technologies, the **term API** is now in general use and a ***unique selling point*** on the **web**

Xebia

A public API is a business service with a web access point that is managed as a product

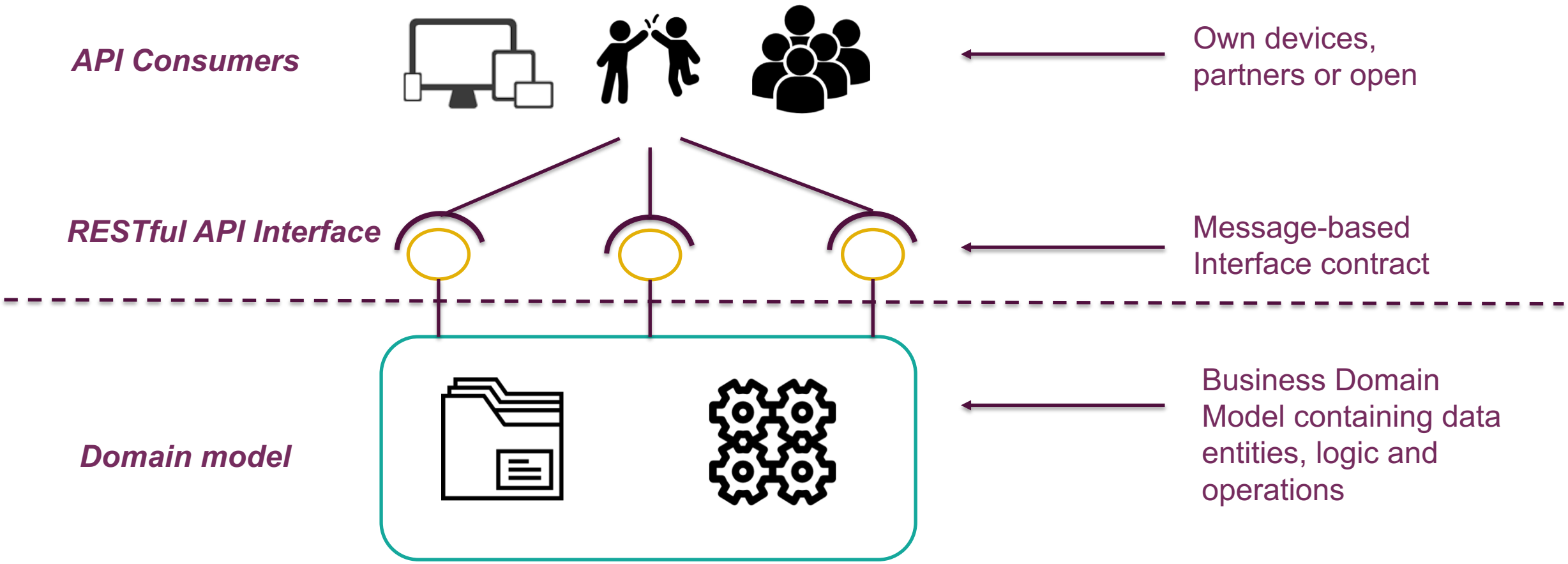


We think a **public API**

- is a open or partner **service**
- that exposes **business capabilities**
- through **consumer-friendly operations**
- that are used to built **new applications**
- accessible from a **web access point**
- and follows a **product life cycle**

Xebia

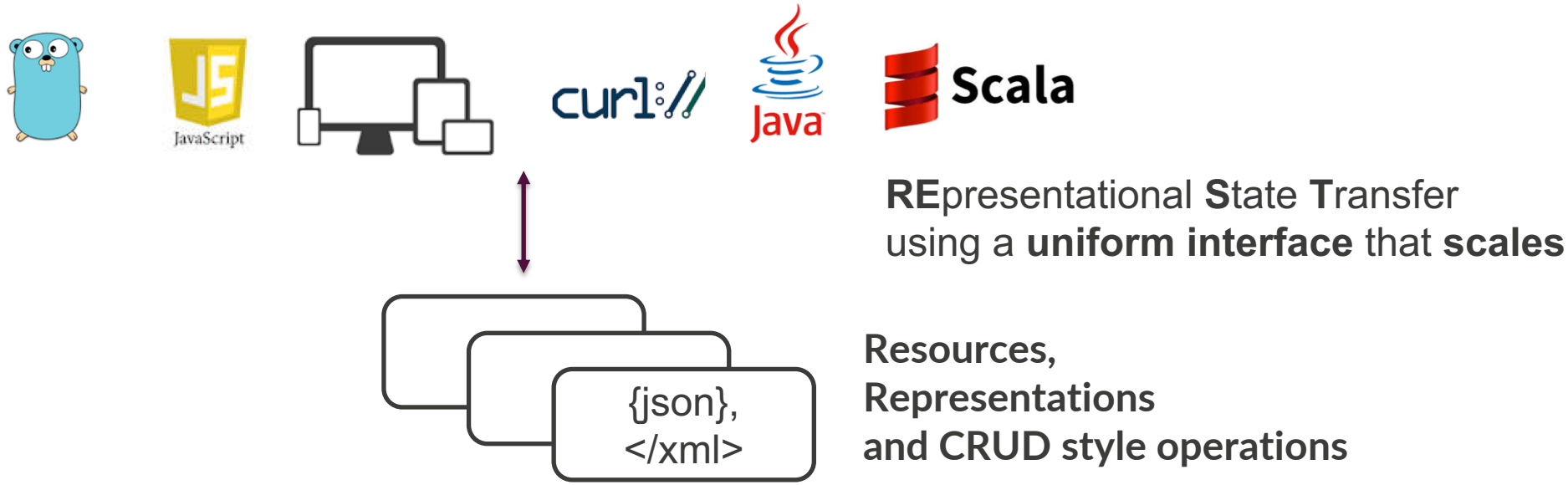
A typical API architecture: A RESTful API as the channel to your domain



But why REST?

Xebia

An API **needs** a **uniform interface, client-server decoupling** and **scalability. REST provides** that



REpresentational **S**tate **T**ransfer
using a **uniform interface** that **scales**

Resources,
Representations
and CRUD style operations

HTTP

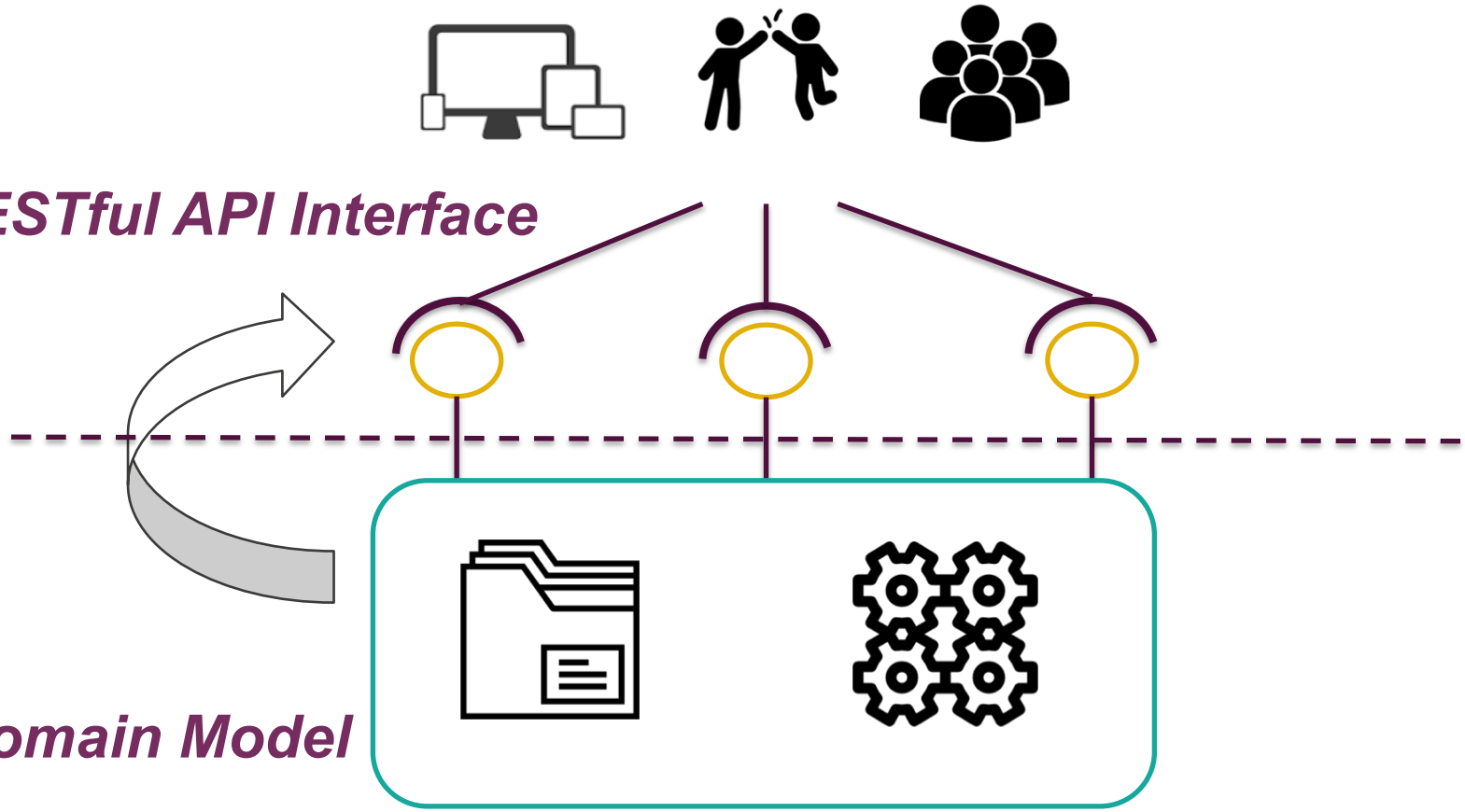
- URI Format
- Request Message Format
- Scalability
- Caching
- HTTP Methods
- Response Message Format

But…

Xebia

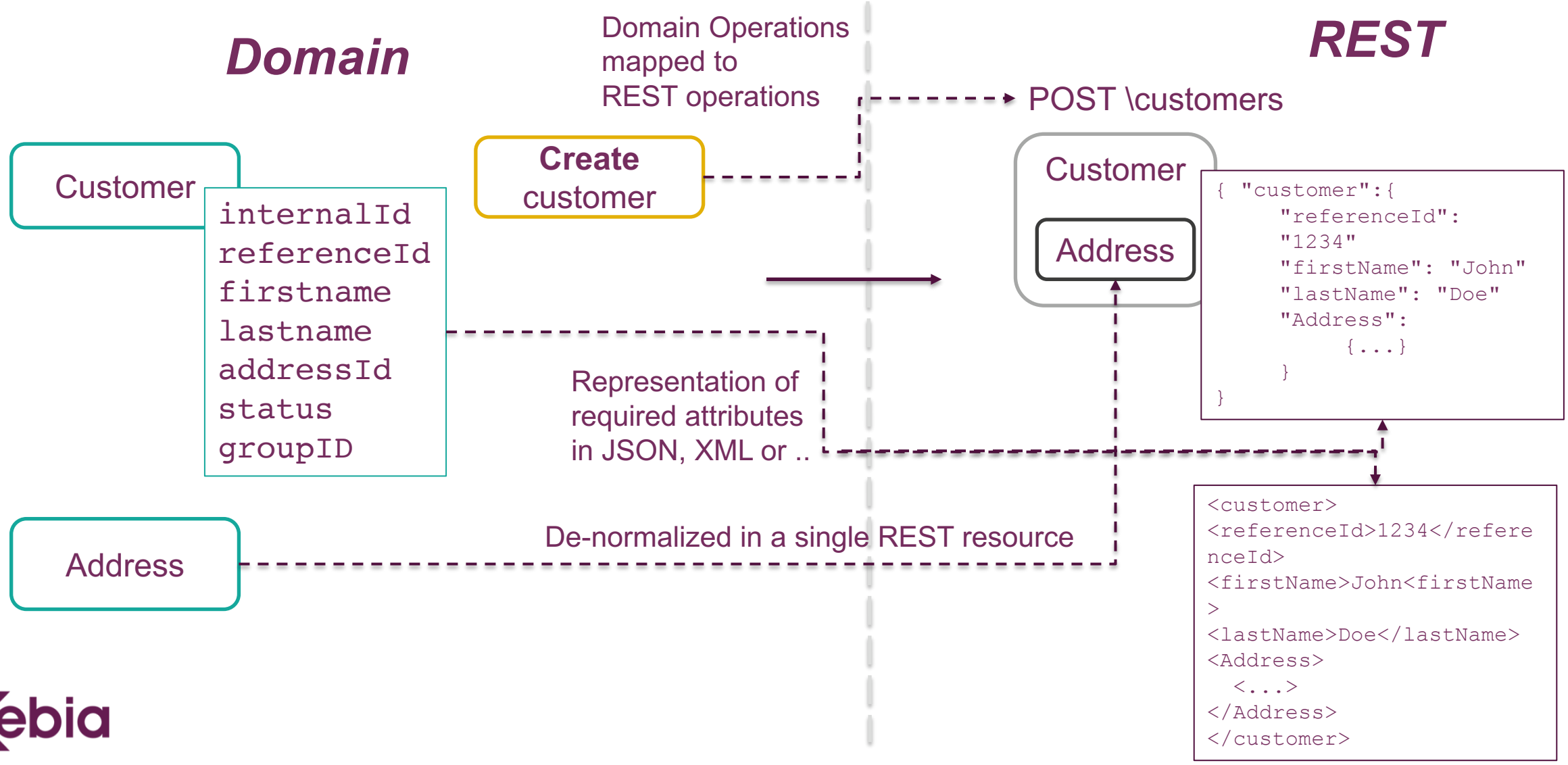
… how do you create a REST API contract for your domain?

*Describing **complex** **domain behavior** in a **simple REST manner** is **not straightforward***



Xebia

Example: Webshop



Xebia

So, resources are the *core* of a RESTful API…

Resource modeling is therefore a **key activity** to…

- Map your domain **operations** to REST operations
- Map your domain **data entities** to REST resource representations

…in order create an API that is user-friendly, extensible and maintainable

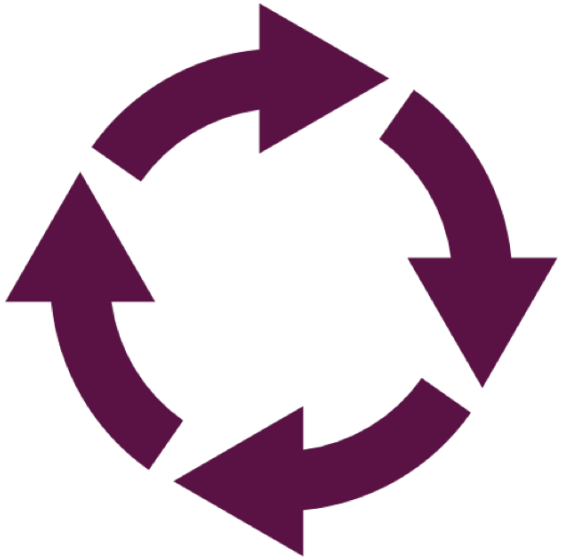
Xebia

Resource Modeling Process

Discover the API domain

Discover the REST resources & operations

Define the API interface



Xebia

Xebia

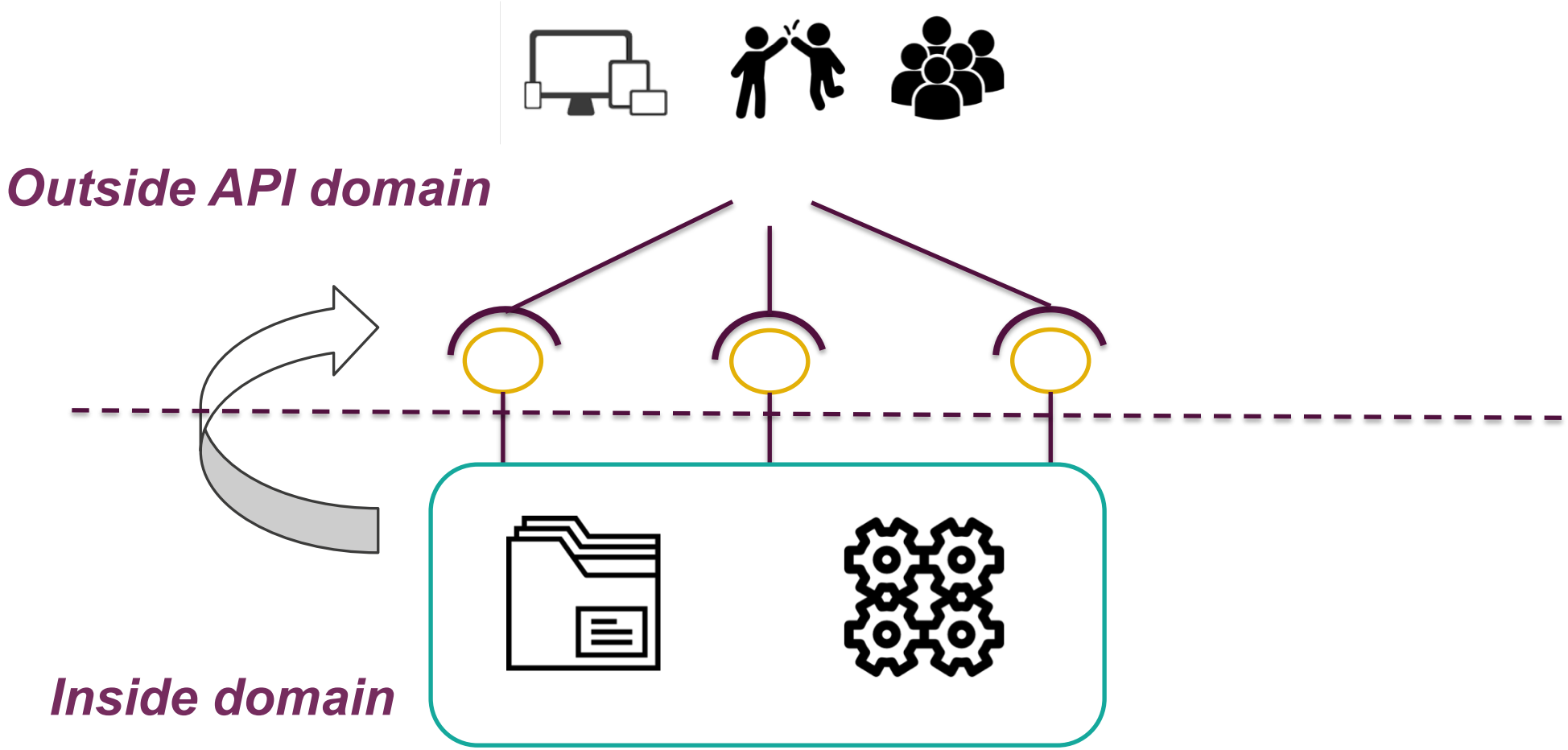
Software Development Done Right

Part 1 – Discover the API domain

Goal

- Practice how to discover API domain entities, relations and operations that *you need in your API*

Discover the API functionality



Xebia

Outside API Model ⊂ Inside Business Model

Use a Ubiquitous Language

Protest against any form of:

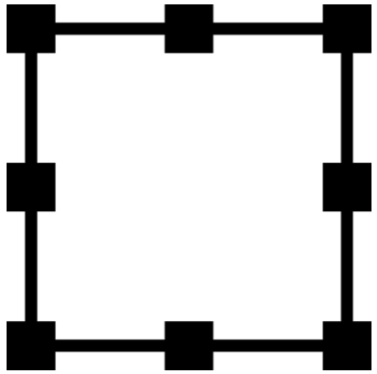
- Ambiguity
- Inconsistencies

Xebia

Create bounded context

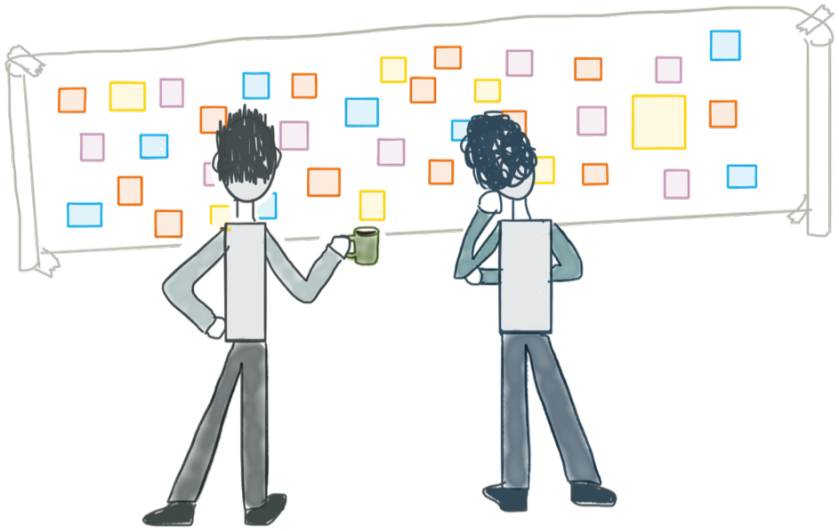
Bounded context helps:

- To logically group API behavior and form an API outline
- To structure the later implementation



Different methods

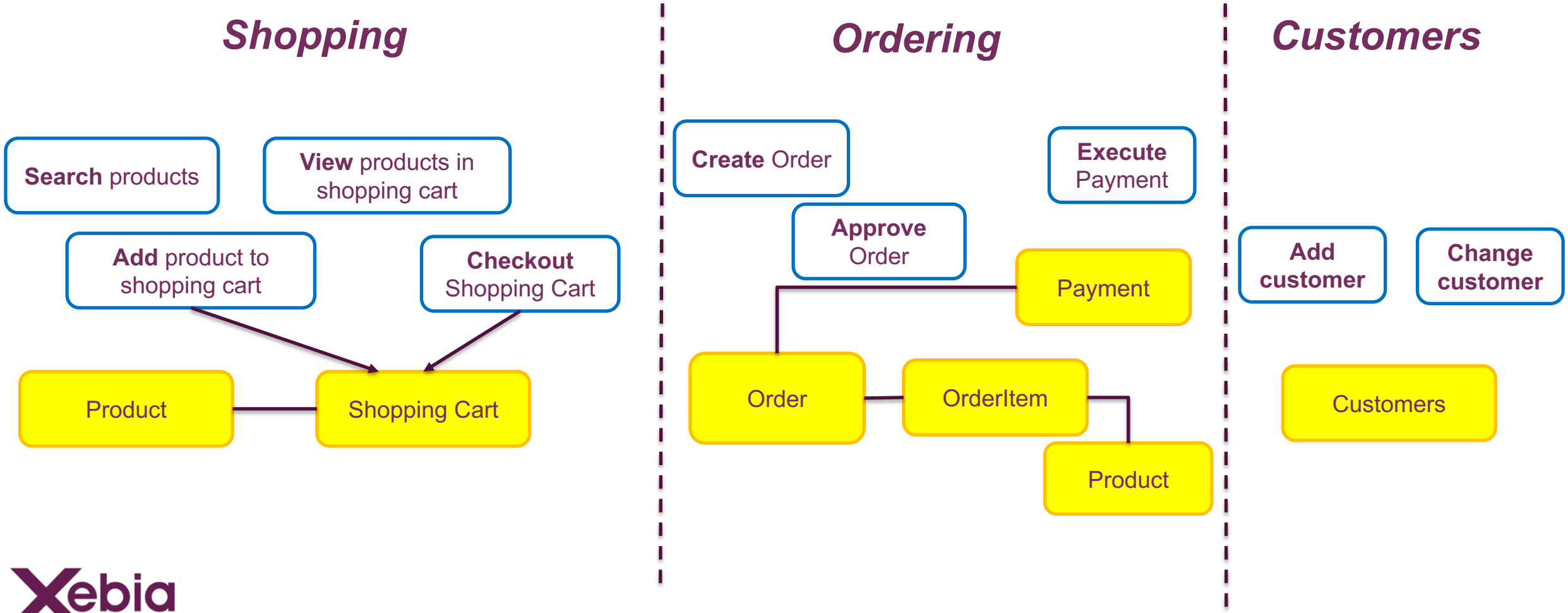
- Input:
  - Use case descriptions
  - Domain experts, external and internal
- Methods:
  - Event storming or other work shops
  - Text and domain analysis (verbs and nouns)
- Output
  - A logical domain model depicting
    - Set of nouns (entities) and verbs (operations) grouped in bounded contexts
    - Entity relations



Xebia

Discover your API model

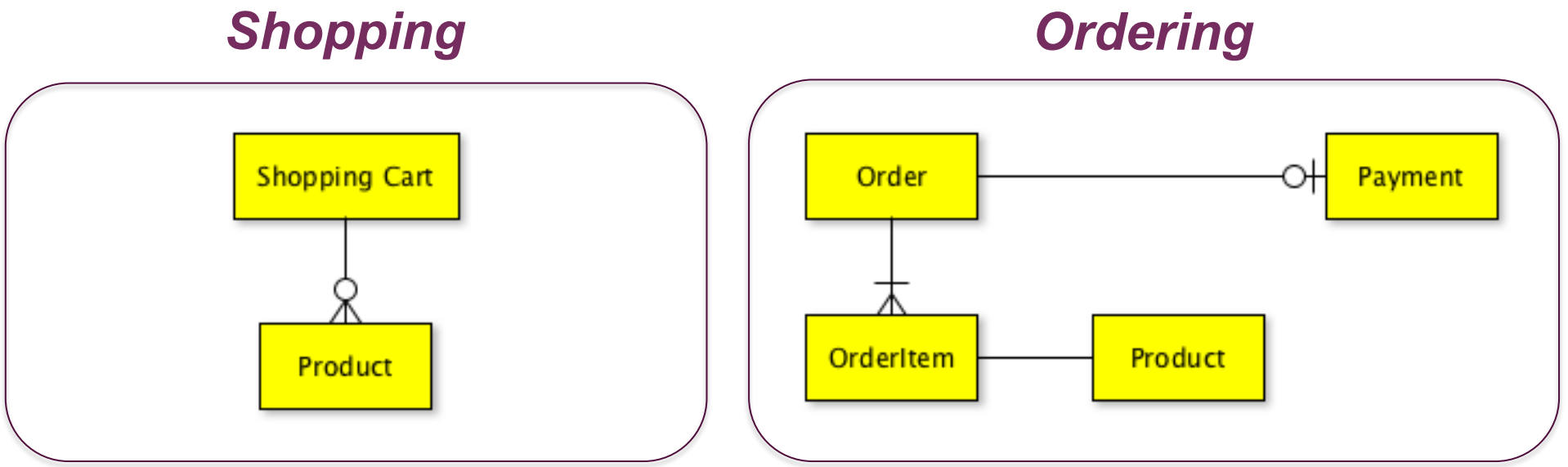
Example: Partial Event–Storming output Webshop



Xebia

Discover your API model

Example: Partial Webshop domain model



Xebia

Exercise 1: Discover the API Domain

Xebia

Exercise Input - Case Description: FlyWithMe airline

FlyWithMe is an airline that wants to increase sales by extending its current business model towards third parties. They plan to do this by providing a Public API.

You are an architect at FlyWithMe, responsible for designing the API.
You and your team decide to start with discovering the domain entities and operations

...more input is provided in the hand-outs

Xebia

Exercise outcome

- List of domain entities
- List of domain operations
- Grouped by bounded context
- Time left? Try to define the entity relations

Tips

- Don’t try to over-complicate the business domain; the goal is not to get a complete understanding of the domain.
- Make assumptions about the business domain where needed
- Focus on API consumer functionality
- Do not focus on the attributes of an entity, solely focus on the *nouns* and *verbs*

And now to work!

- Organize yourselves in groups of 3-4
- Work on exercise till 14:45
- *Make use of whiteboard, paper, flip overs (easel pad) etc. to visualize your answer.*

Case material: http://bit.ly/restfulapidesign

Xebia

Retro time

- What went well?
- What was difficult?

Xebia

Xebia

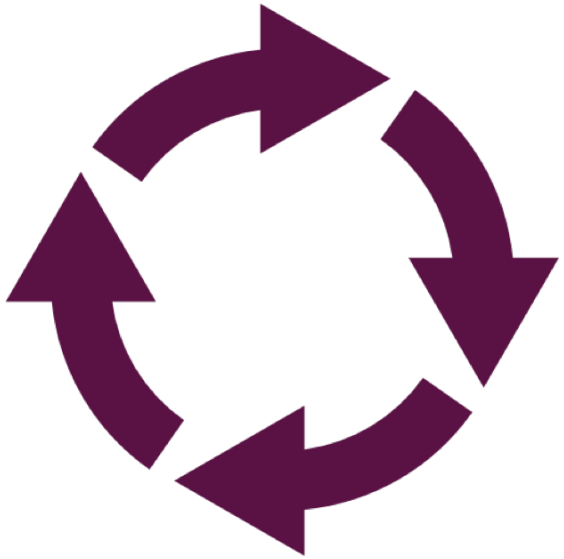
Software Development Done Right

Part 2 – REST resource modeling

Goal

- Learn how to map domain entities and operations to REST resources and operations
- Learn about API documentation options
- Practice by modelling basic use cases from the case study

REST API Resource Modeling Process

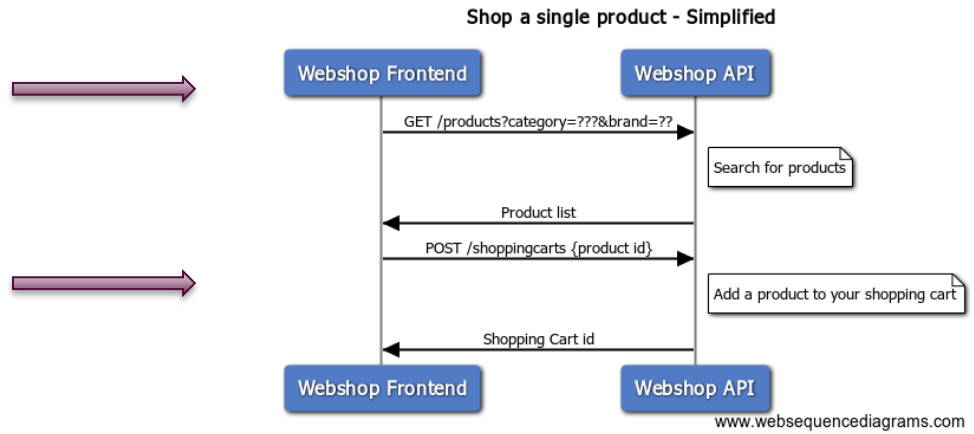


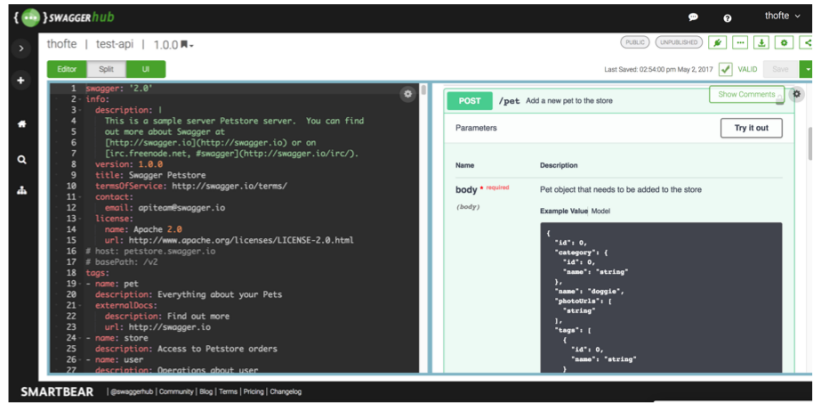
Xebia

From a domain model to a REST contract
A typical flow

Domain operations

Domain Entities





Step 1 – Define sequence diagrams

- Group related domain operations in user stories
- Map domain operations and entities to REST resources and operations

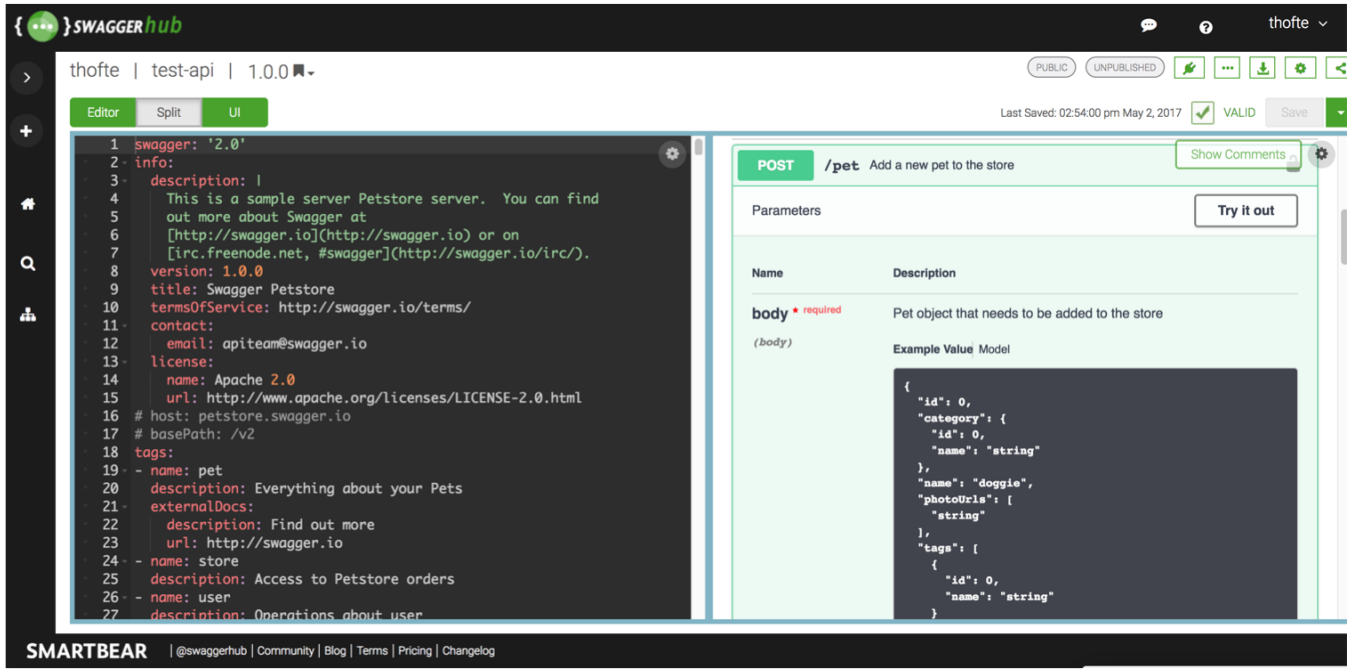
Step 2 – Define the API in a specification language

- *Determine the resource representations*
- *Operation granularity*

Xebia

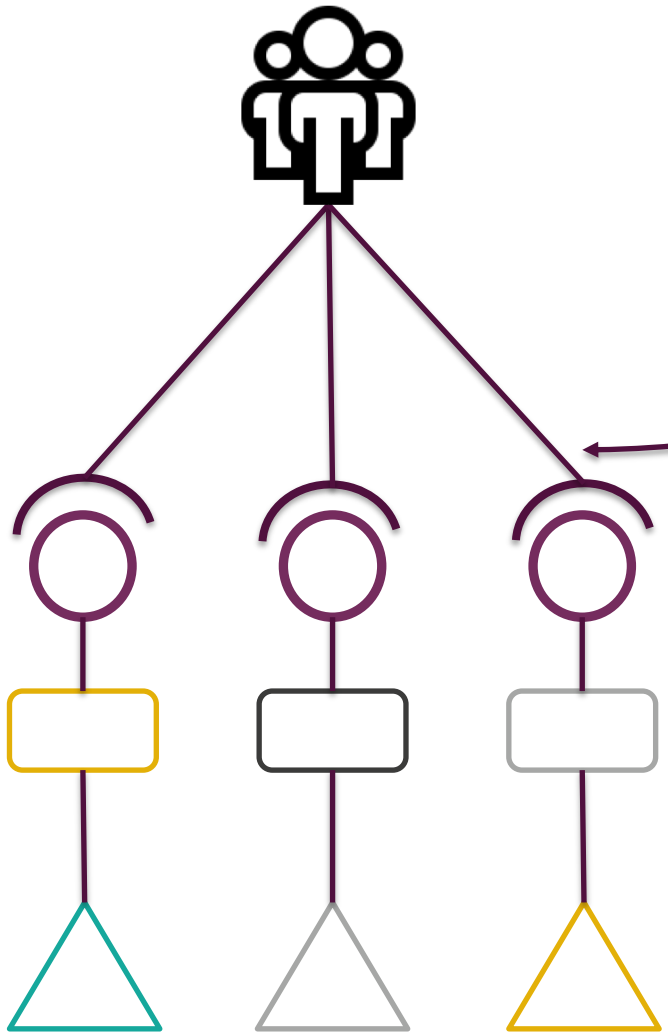
REST API specification language

- Start defining the API using a API description languages:
  - **Open API (fka Swagger)**
  - RAML
  - API Blueprint
- Kickstart your development with an API design tool
  - SwaggerHub
  - Apiary for API blueprint
  - Confluence + swagger.io plugin



Xebia

Anatomy of a RESTful API *REST= Representational State Transfer*



Resource location = URL

Resource Operations = VERBS = GET | POST | PUT | DELETE

Resources representations = NOUNS representations = Media Types (JSON, XML)

resources = NOUNS

Xebia

REST resource types

- **Document**: A single document with optional subresources
  - `https://api.flywithme.com/flightoffers/{id}`
- **Collection**: A server managed collection: a plural noun
  - `https://api.flywithme.com/airports`

REST Operations - Standard HTTP operations

Retrieve resources

```
GET https://api.flywithme.com/airports/{airportid}
```

Create a new resource

```
POST https://api.flywithme.com/bookings body: trip
```

Update an existing resource

```
PUT https://api.flywithme.com/bookings/{bookingid} body: updated trip
```

Delete an existing resource

```
DELETE https://api.flywithme.com/bookings/{bookingid}
```

Xebia

REST Operations – non-CRUD operations

- Sometimes it can be *difficult* to model a business capability spanning multiple resources with fine-grained HTTP CRUD operations.

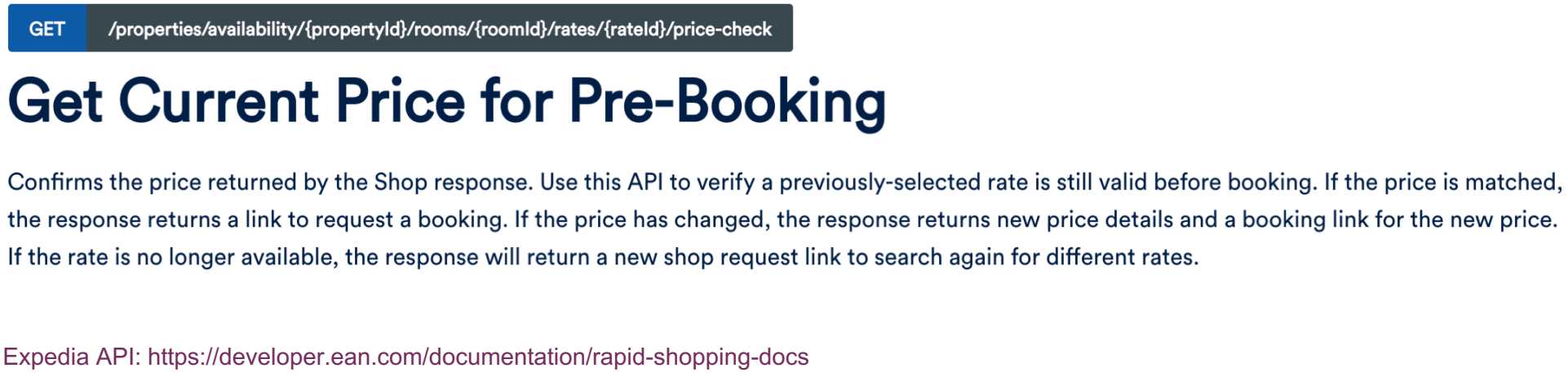
  - **A business capability is a *long-running* process**
  - **A business capability is a synchronous function**

- Add non-CRUD verbs or noun-ified verbs to your API

Xebia

REST Operations – non-CRUD Verbs

- A verb in your URL representing a synchronous action or function
  - Typically modelled with GET



Expedia API: https://developer.ean.com/documentation/rapid-shopping-docs

Xebia

REST Operations – Noun-ified verbs

- **A noun representing an asynchronous action**
  - Long running
  - Action can be monitored
  - Typically modelled with HTTP POST
    - REST without PUT / CQRS

Create a fork

Create a fork for the authenticated user.

```
POST /repos/:owner/:repo/forks
```

Parameters

Name	Type	Description
`organization`	`string`	Optional parameter to specify the organization name if forking into an organization.

Response

Forking a Repository happens asynchronously. Therefore, you may have to wait a short period before accessing the git objects. If this takes longer than 5 minutes, be sure to contact GitHub support.

Source: GitHub API

Xebia

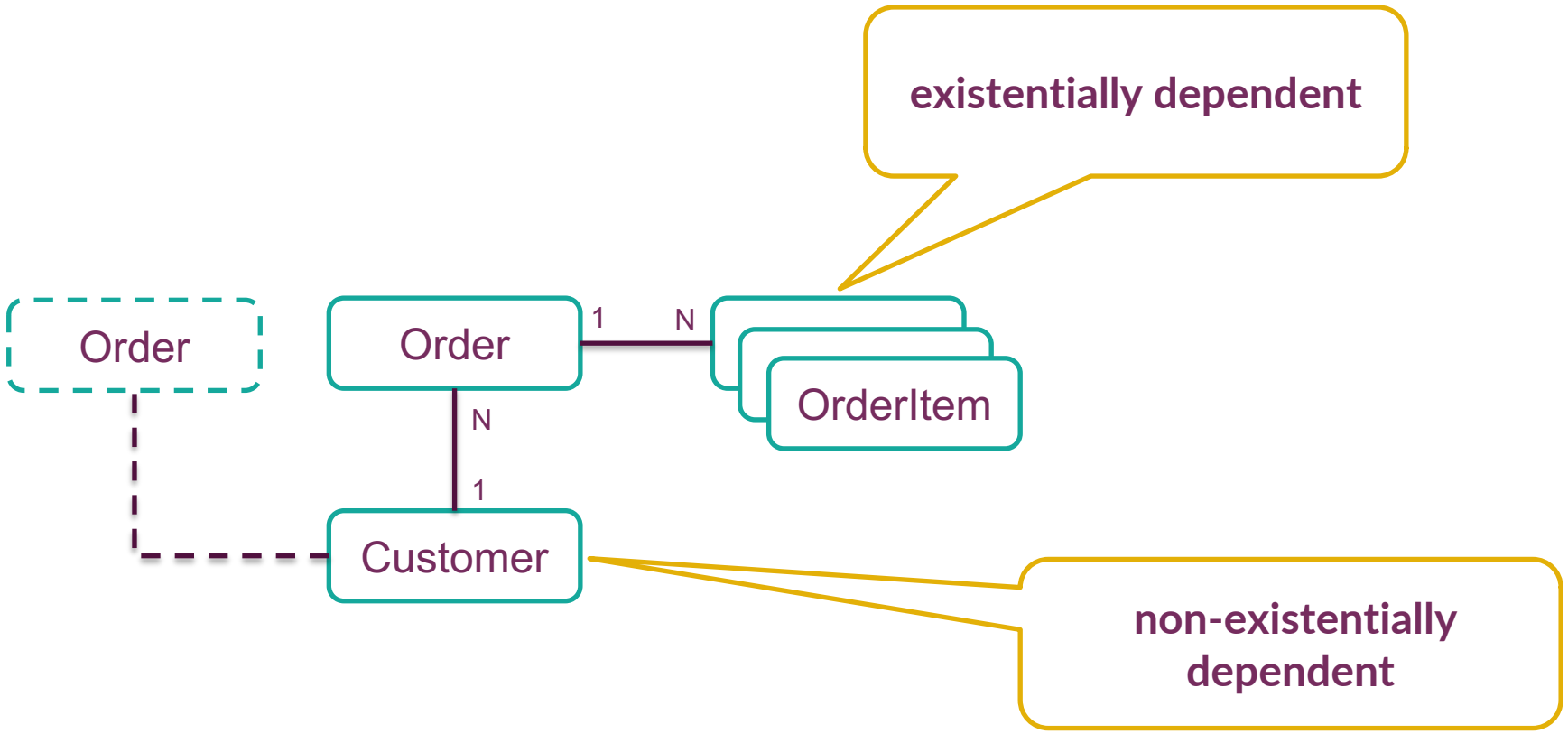
REST Operations – *Fine-grained HTTP CRUD operations or course-grained non-CRUD operations?*

It depends:

- ***CRUD** should be your first option*
- *Use **Non-CRUD verbs** for **synchronous operations** for which non-CRUD becomes clumsy*
- *Use **noun-ified verbs** for **long running actions** or events that must be monitored*

Xebia

REST Resource Relations



Xebia

REST Resource Relations - Modeling

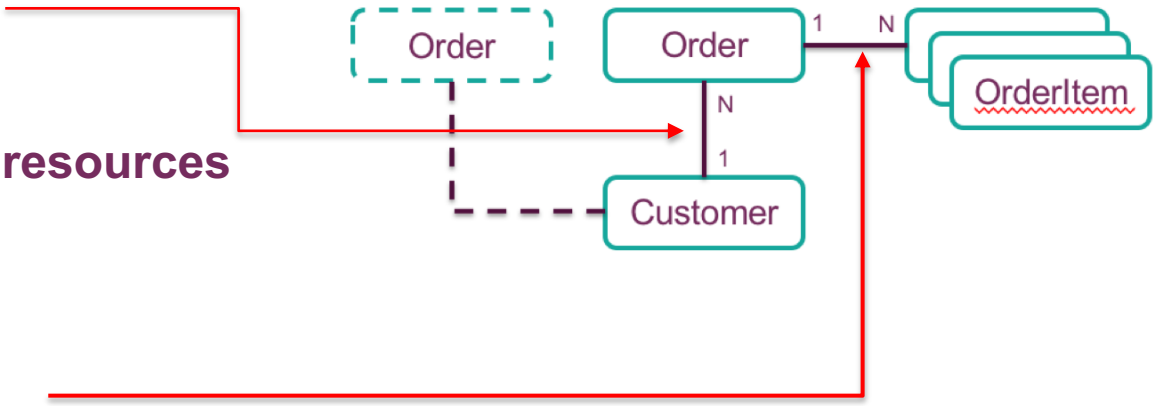
Resource modeling *options*:

- **Linked Resources**

```
{
   orderId: 1
    ...
   customerId: 1
    ...
}
```

- **Embedded resources**

```
{
   id: 1
   ...
   orderItems:
   [
     {
       id: 1
       product: "X"
       ..    }
    ]
}
```



Path modeling *rules*:

- **existentially dependent**

```
\orders\{orderid}\orderitems\{orderitemid}
```

- **non-existentially dependent**

```
\customers\{customerid}
```

Xebia

REST Resource Representations – *Media Types*

- Mime-types used most often in the Content-Type header:
  - application/json
  - application/xml
- But binary content is also possible:
  - Content-Type: application/pdf
- Or create your own custom type:
  - Content-Type: application/vnd.amazon.ebook

Xebia

REST operation response codes

Reuse HTTP response code semantics

- **Range 2xx** – Client message has been received, understood and processed succesfully
  - E.g. Use HTTP 201 when a resource is created
- **Range 3xx** – Location redirection
- **Range 4xx** – Request could not be processed due to *an error caused by the client*
- **Range 5xx** – Request could not be processed due *to an error caused by the server*

Exercise 2: Discover REST resources and operations

Xebia

Exercise input

- Take the output of exercise 1 and …
  - Take your team result
  - Or take our example result
- Look at the 2 user stories (*) and define a RESTful API by
  - Creating a sequence flow diagram
  - A Swagger file specification

(*) User stories will be provided in the exercise

Exercise outcome

For each of the 2 user stories:

- Sequence flow diagrams highlighting the REST operations and resources
- An initial swagger file specifying the REST operations and resources needed to implemented the user stories

And now to work!

- Work on the exercise till 16:30
- *Make use of*
  - *https://www.websequencediagrams.com*
  - *https://editor.swagger.io*
  - *https://swagger.io/specification/*

Case material: http://bit.ly/restfulapidesign

Xebia

Coffee Break

Xebia

Running out of material, but not out of time ?

You can add the following features:

- Notify partner websites with flight updates
- Update bookings

Xebia

Retro time

- What went well?
- What was difficult?

Xebia

Xebia

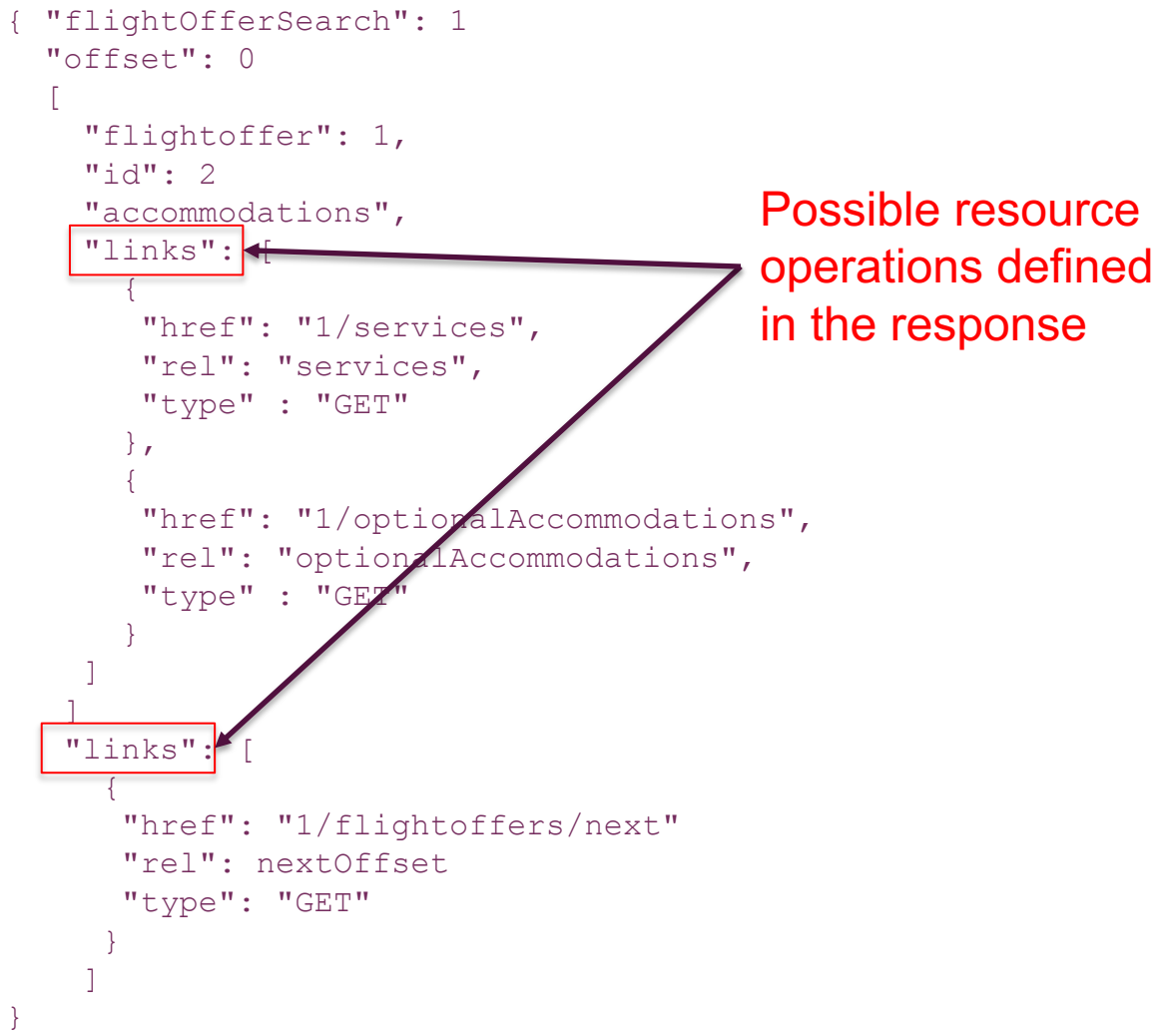
Software Development Done Right

Part 3 – Other *Important* Topics

Hypermedia

- **HATEOAS** (Hypermedia as the Engine of Application State) – Richardson Maturity Model – Level 3
- Self describing, discoverable API
- Changes do not break the contract

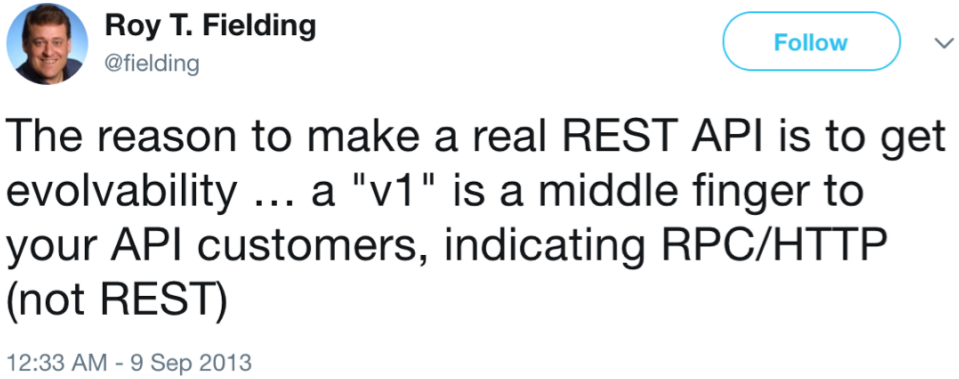
- Some frameworks / standardisation efforts:
  - Hypertext Application Language – HAL
  - Structured Interface for Representing Entities – SIREN
  - JSON-LD and Hydra
  - JSON-API



Xebia

Evolving the API

Changes will happen, *so design for them.*

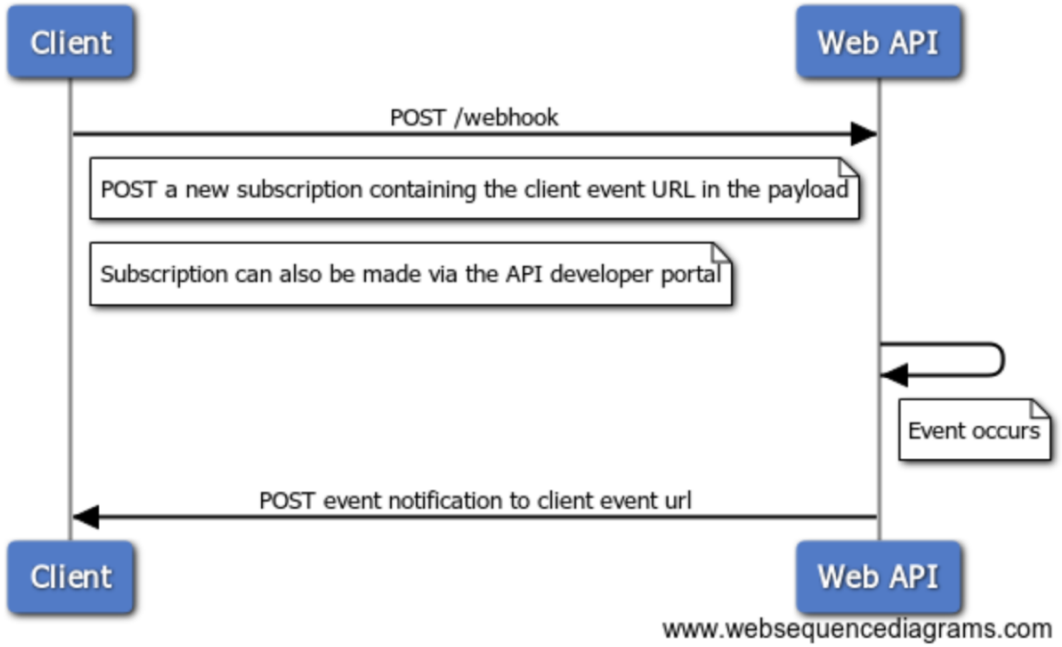


- **By using hypermedia or …**
- **Support versioning, but only on MAJOR versions**
  - **Make** backwards compatible changes and **avoid** breaking changes
  - **Deprecate** but leave in old functionality

Xebia

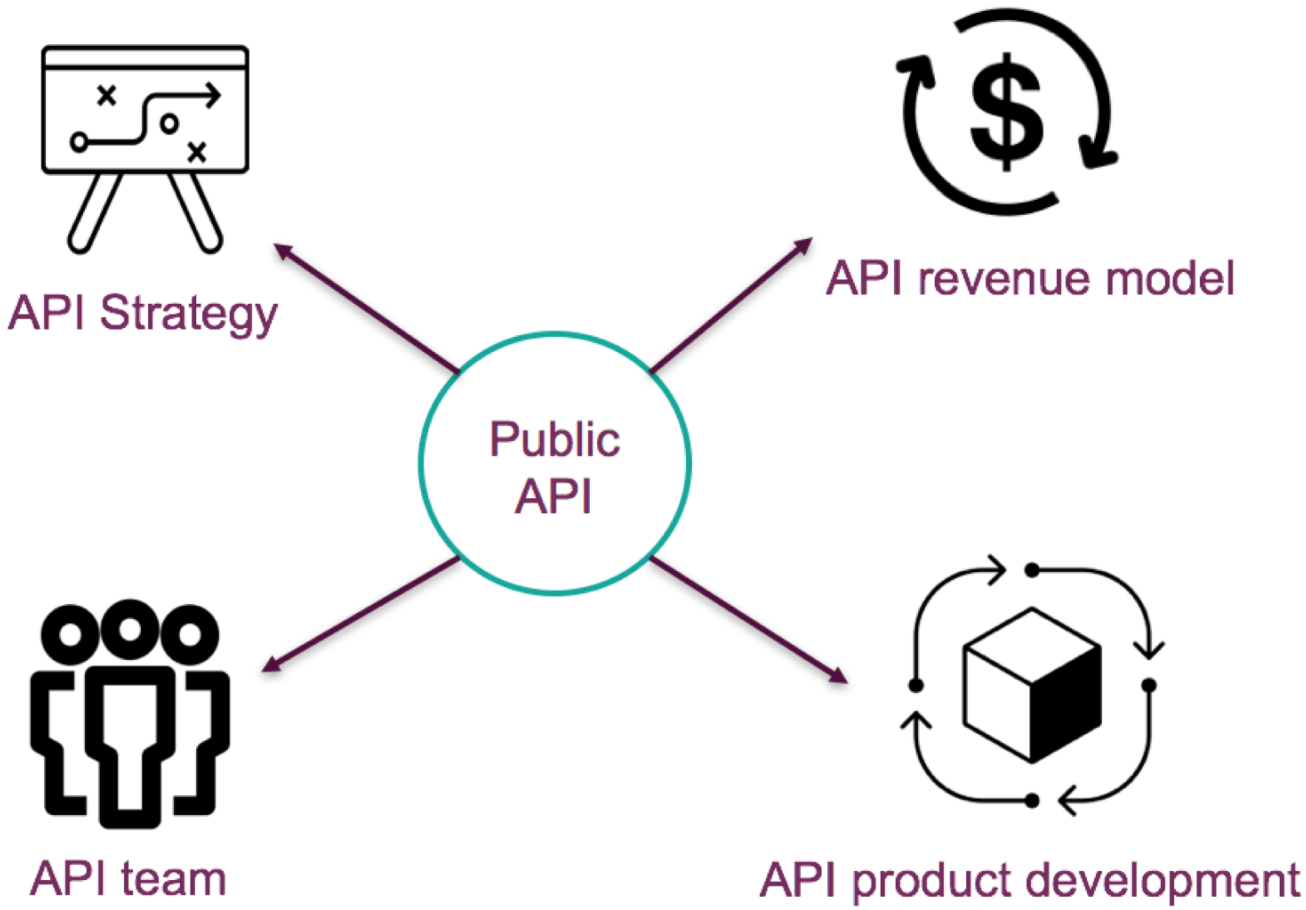
Webhook pattern

- Event-driven integration via server callbacks
  - Flight updates
  - Price changes
- Polling is bad for everyone
  - ~95% of polling requests are useless
  - Reduce load on server



Xebia

Process & People



Xebia

What have we learned?

Xebia

Learning points

- A RESTful API **interface <>** your domain **model**
- **Embrace non-CRUD** operations
- Don’t be **afraid** of changes, but **facilitate** them in your API **design** and in **communication** to your API consumers
- Be pragmatic and work iteratively
- Don’t forget about the **people** and the **process**

Xebia

Software Development Done Right

#OreillySACON

TUTORIAL

Thank you for attending!

Tom Höfte, IT Architect, Xebia, *thofte@xebia.com*

Marco van der Linden, IT Architect, Xebia, *mvanderlinden@xebia.com*