Co-Design of Business and IT Services

A Tool-Supported Approach

Blagovesta Pirelli, Natalia Nessler, Anton Ragot,
Alain Wegmann

LAMS, EPFL, Switzerland

blagovesta.pirelli@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Problem

- Business requirements are often semi-formal
  - “As a participant, I want to upload my camera-ready paper"

- IT specifications have to be precise
  - In (authorId: String, paperId: String, …) -> Out (200 OK)

➔ Why not use the same modeling method?

Motivation: Design of Services

+ Services are used in business and IT
+ OpenAPI is prevalent for RESTful services

— No single service modeling method for business requirements and IT specifications

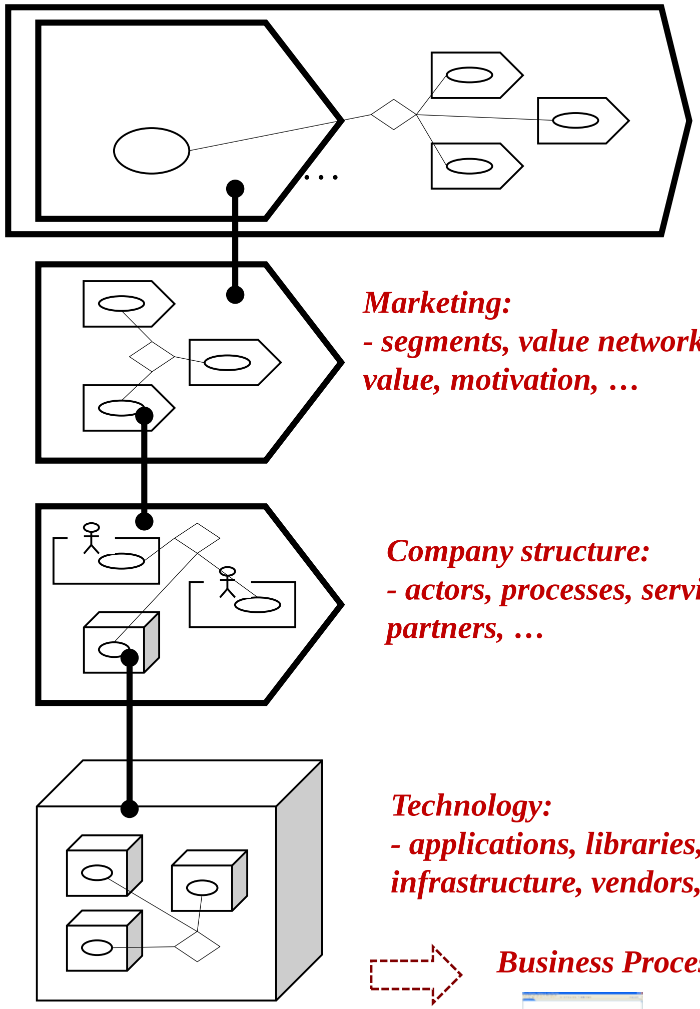
? *How do we extend existing service modeling to simultaneously design business and IT services?*

The SEAM Method

- Systemic
  (hierarchical complex systems, networked organizations)
- Concrete
  (project-based, story-telling, “examples”, …)
- Subject-based
  (viewpoints, goals)

- Slide credits: Alain Wegmann

as-is / to-be



Business Idea (TYM)

Marketing:
- segments, value network, customers, value, motivation, …

[CRM](#) [Config](#).

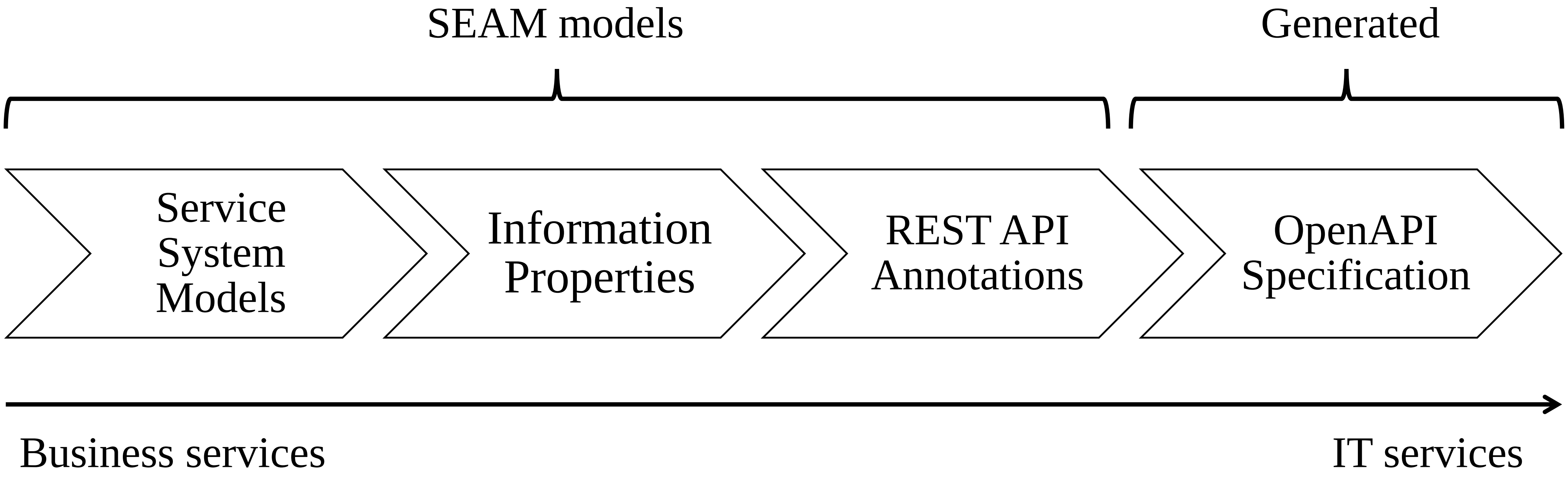
Company structure:
- actors, processes, services, partners, …

Project Proposal

Technology:
- applications, libraries, VMs, infrastructure, vendors, …

Business Process Specification

Modeling Process – Top-Down



SEAM models

Generated

Service System Models

Information Properties

REST API Annotations

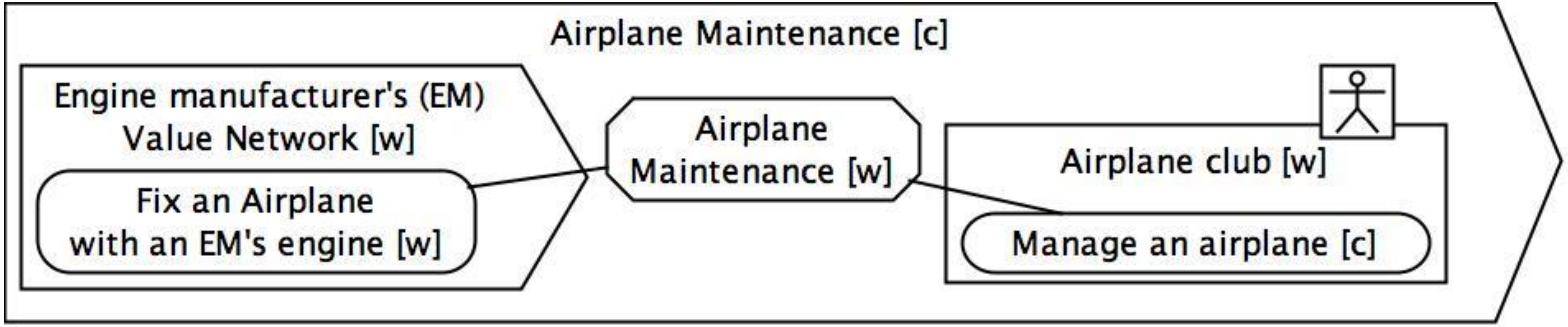
OpenAPI Specification

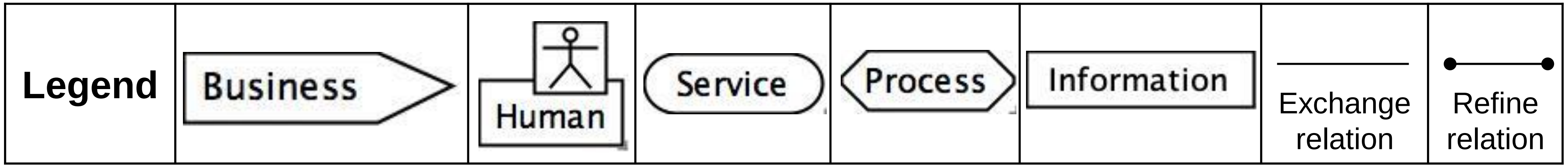
Business services

IT services

Modeling Process: Service Models

- Understand the business environment
- Model the services provided to the client

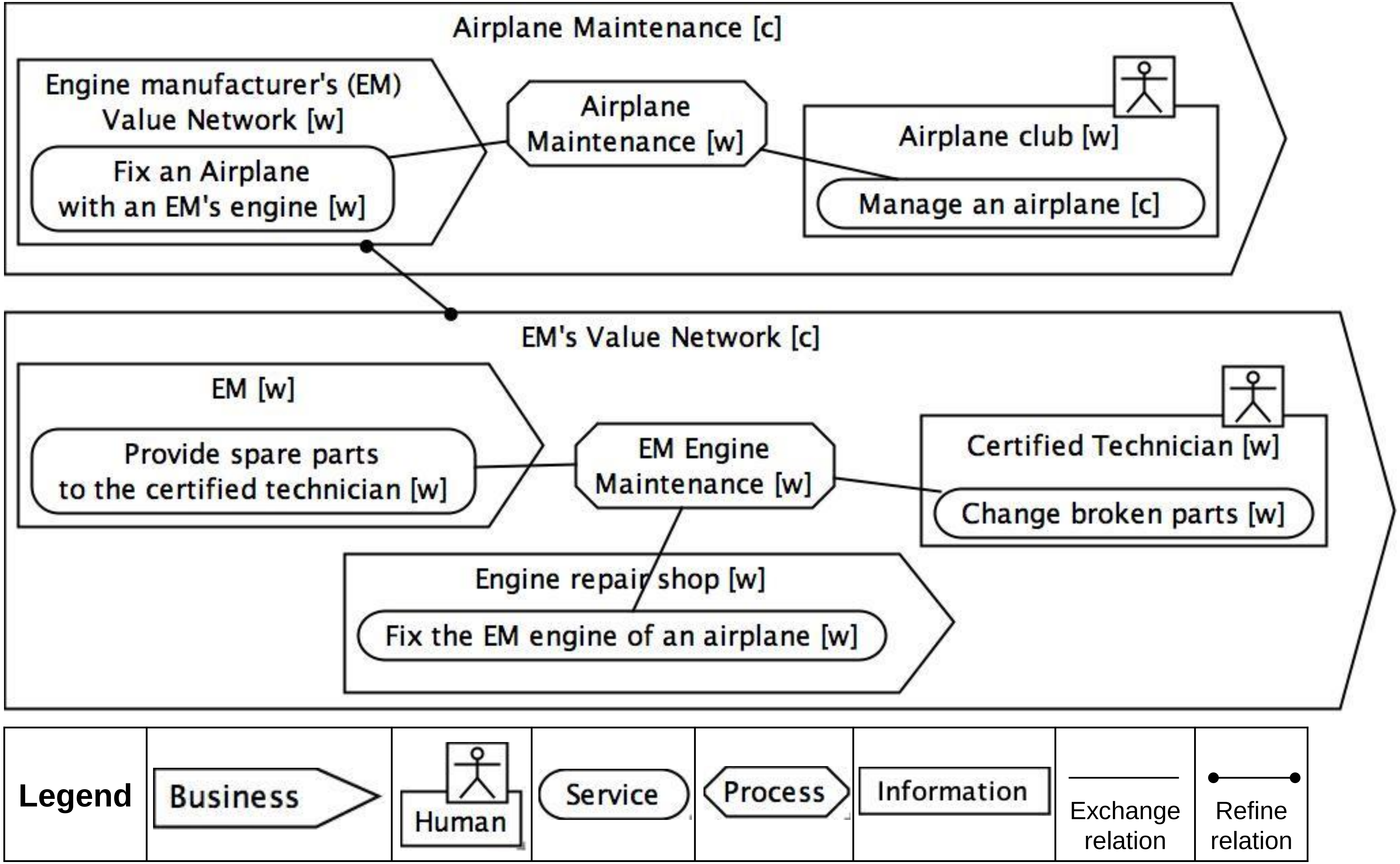

Airplane Maintenance [c]
Engine manufacturer's (EM) Value Network [w]
Fix an Airplane with an EM's engine [w]
Airplane Maintenance [w]
Airplane club [w]
Manage an airplane [c]


Legend
Business
Human
Service
Process
Information
Exchange relation
Refine relation

Modeling Process: Service Models

- Understand the internal organization (value network) of the service provider

- Model the services of the actors of the service provider’s value network

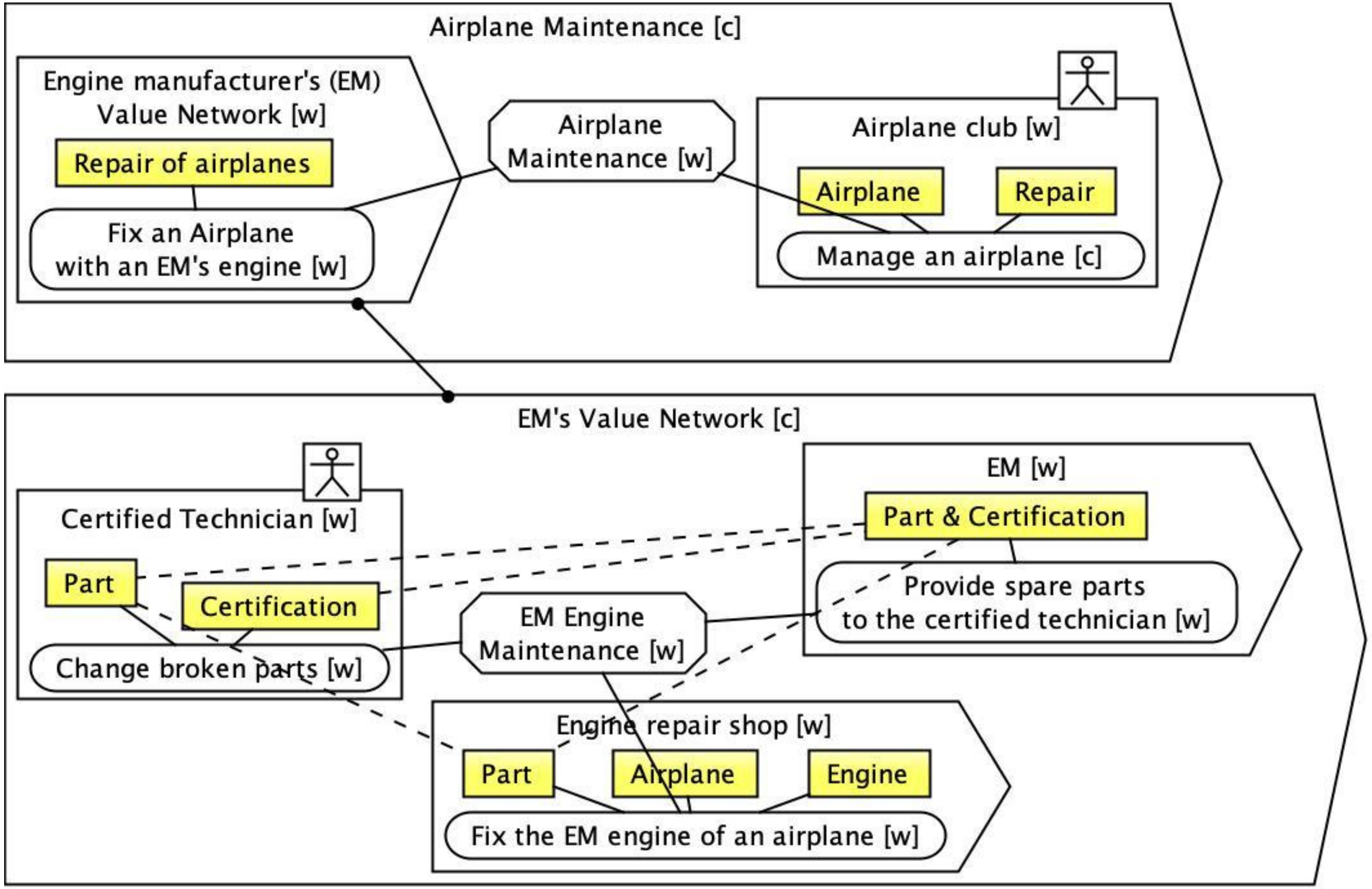

Airplane Maintenance [c]
Engine manufacturer's (EM) Value Network [w]
Fix an Airplane with an EM's engine [w]
Airplane Maintenance [w]
Airplane club [w]
Manage an airplane [c]
EM's Value Network [c]
EM [w]
Provide spare parts to the certified technician [w]
EM Engine Maintenance [w]
Certified Technician [w]
Change broken parts [w]
Engine repair shop [w]
Fix the EM engine of an airplane [w]
Legend
Business
Human
Service
Process
Information
Exchange relation
Refine relation

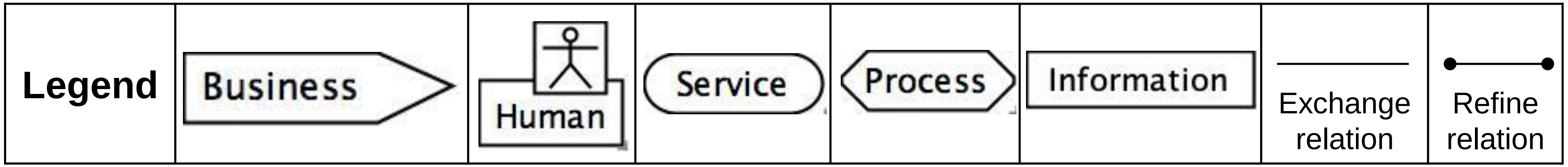
Modeling Process: Information

- Find the lingo of the actors

- Model the information actors have while participating in the service exchange

- Possible integration with DDD

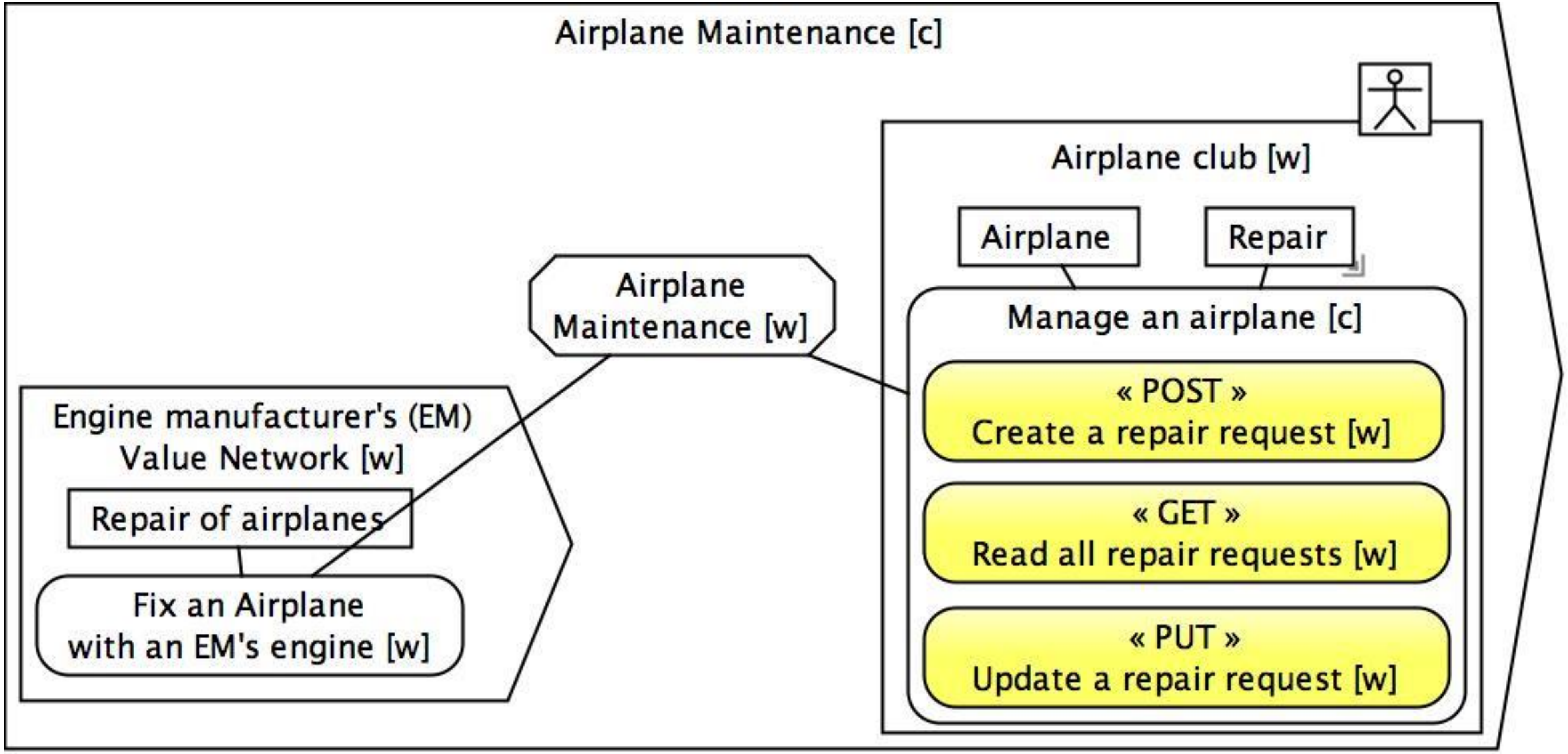

Airplane Maintenance [c]
Engine manufacturer's (EM) Value Network [w]
Repair of airplanes
Fix an Airplane with an EM's engine [w]
Airplane Maintenance [w]
Airplane club [w]
Airplane
Repair
Manage an airplane [c]
EM's Value Network [c]
Certified Technician [w]
Part
Certification
Change broken parts [w]
EM Engine Maintenance [w]
EM [w]
Part & Certification
Provide spare parts to the certified technician [w]
Engine repair shop [w]
Part
Airplane
Engine
Fix the EM engine of an airplane [w]


Legend
Business
Human
Service
Process
Information
Exchange relation
Refine relation

Modeling Process: REST Annotation

- Define CRUD operations on the information properties

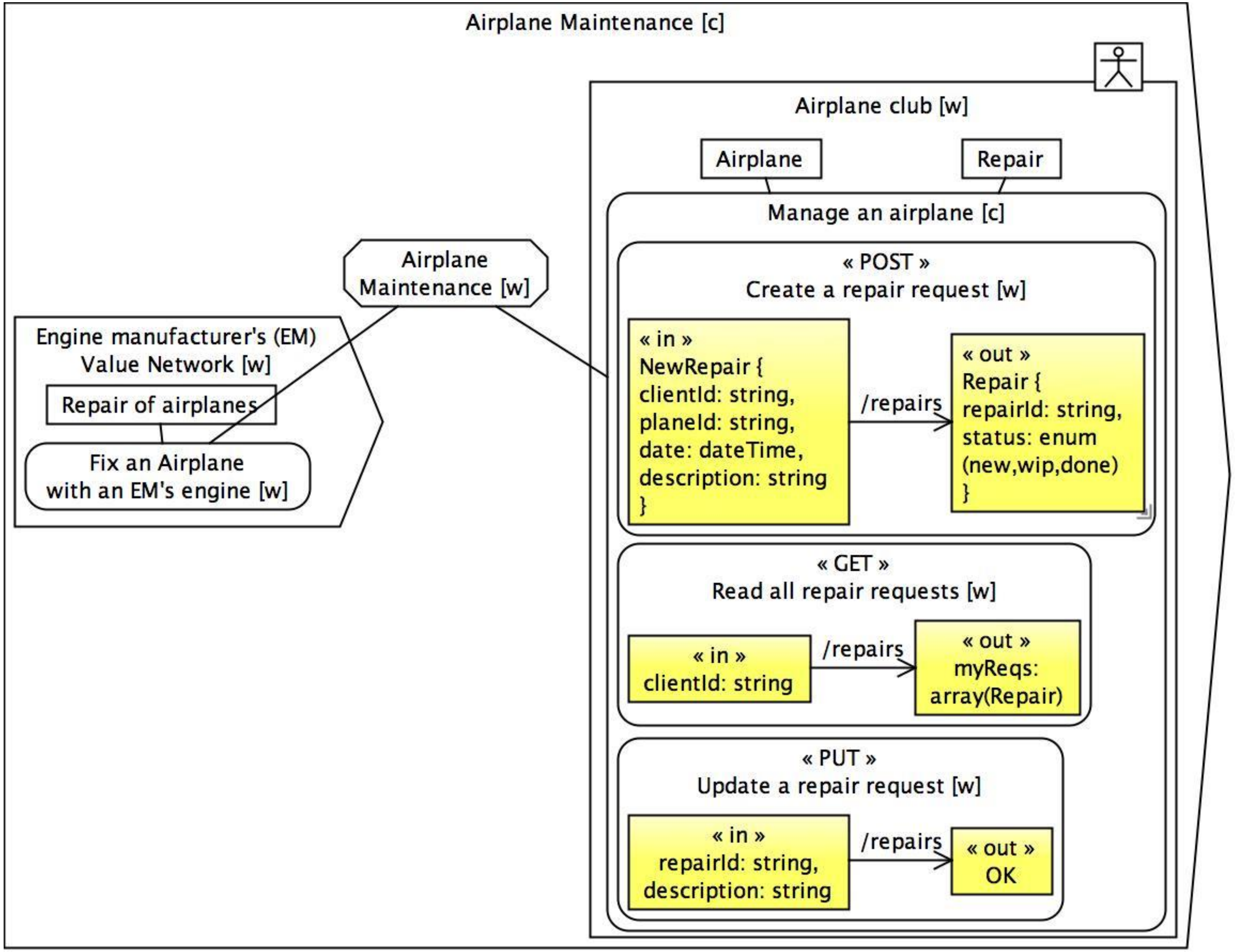
- Connect the RESTful services to the business service they support



Legend	Business	Human	Service	Process	Information	Exchange relation	Refine relation

Modeling Process: REST Annotation

- Specify the parameters and the path
- Possible: built-in, schema, enumeration, array


Airplane Maintenance [c]
Airplane club [w]
Airplane
Repair
Manage an airplane [c]
« POST »
Create a repair request [w]
« in »
NewRepair {
clientId: string,
planeId: string,
date: dateTime,
description: string
}
/repairs
« out »
Repair {
repairId: string,
status: enum
(new,wip,done)
}
« GET »
Read all repair requests [w]
« in »
clientId: string
/repairs
« out »
myReqs:
array(Repair)
« PUT »
Update a repair request [w]
« in »
repairId: string,
description: string
/repairs
« out »
OK
Airplane
Maintenance [w]
Engine manufacturer's (EM)
Value Network [w]
Repair of airplanes
Fix an Airplane
with an EM's engine [w]

Modeling Process: OpenAPI Specs

- Generate the OpenAPI specification with a tool

- Tool available on GitHub

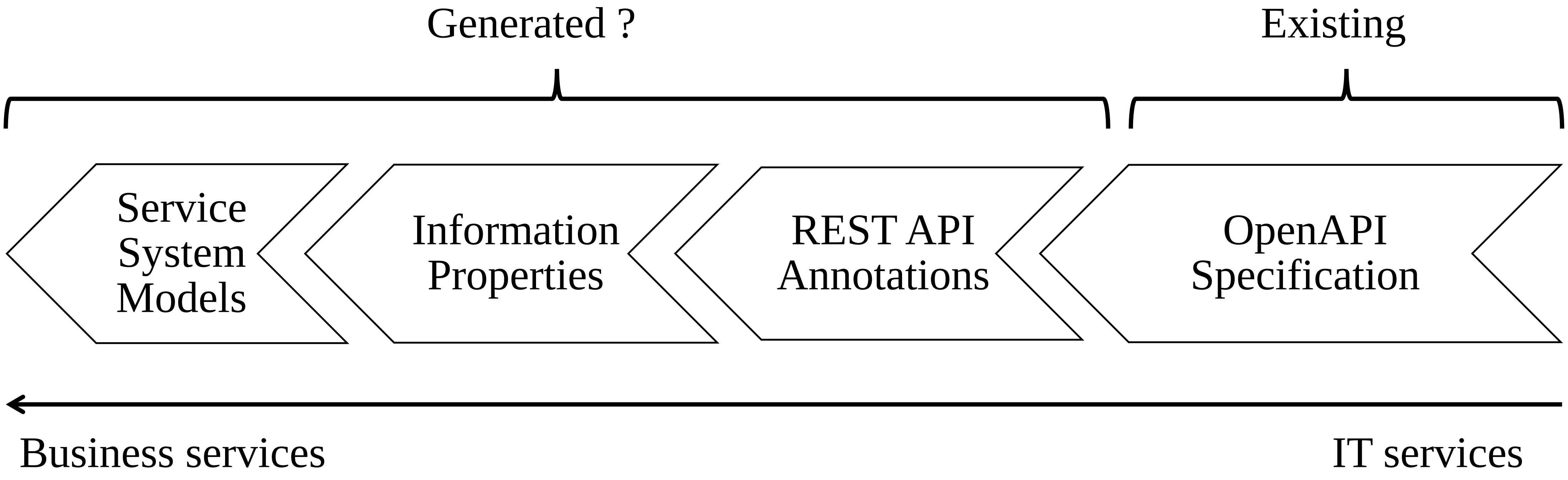
  => https://github.com/lams-epfl/gen-rest/

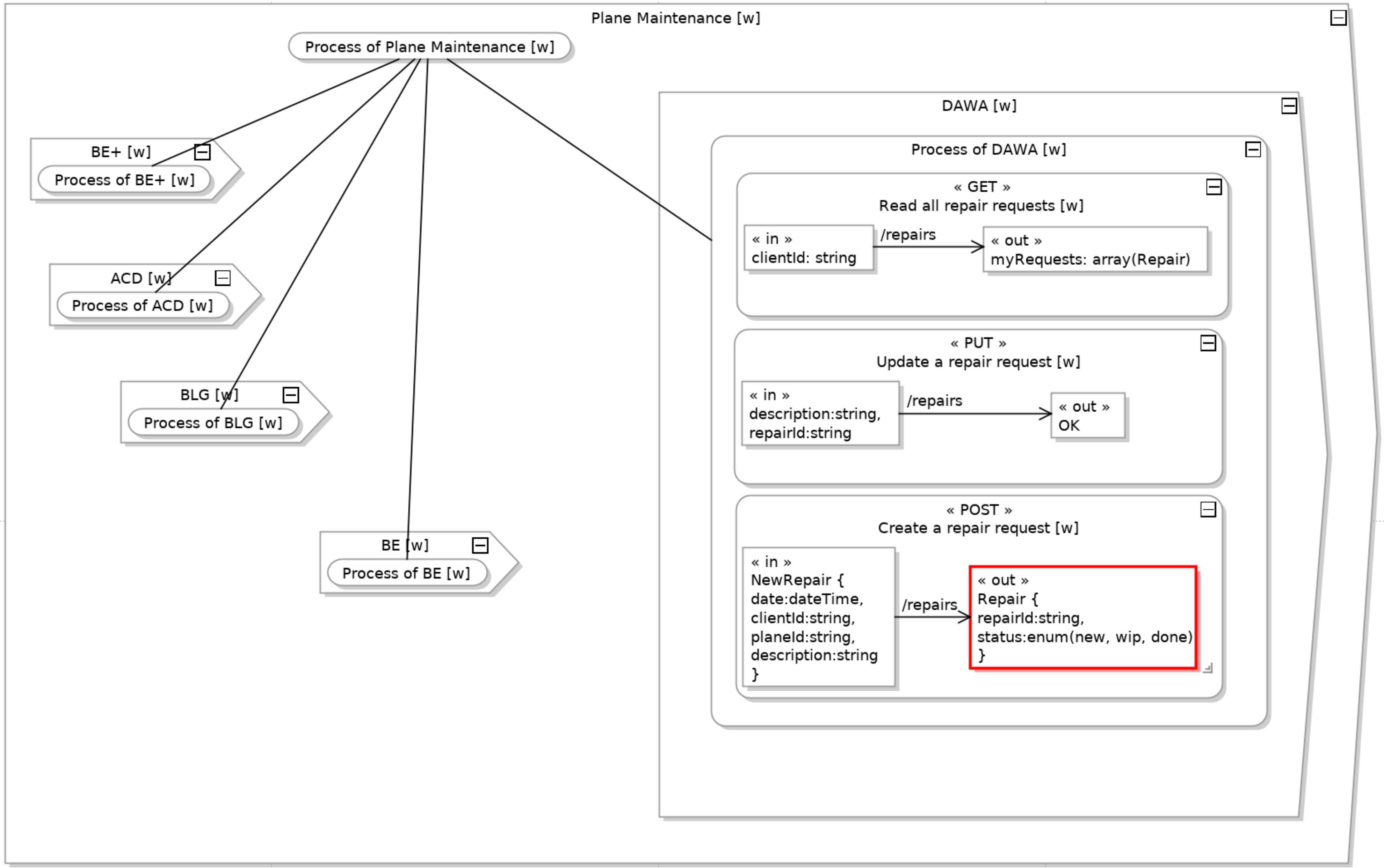
```yaml
openapi: 3.0.0
servers: []
info:
  version: 1.0.0
  title: Airplane maintenance
tags:
  - name: EM VN
  - name: Airplane club
  - name: Technician
  - name: Engine repair shop
  - name: EM
```

```yaml
paths:
  /repairs:
    post:
      tags:
        - Airplane club
      description: Create a repair request
      responses:
        '200':
          description: request successful
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Repair'
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/NewRepair'

components:
  schemas:
    NewRepair:
      type: object
      properties:
        clientId:
          type: string
        planeId:
          type: string
        date:
          type: string
          format: date time
        description:
          type: string
```

Modeling Process – Bottom-Up





Plane Maintenance [w]

Process of Plane Maintenance [w]

BE+ [w]

Process of BE+ [w]

ACD [w]

Process of ACD [w]

BLG [w]

Process of BLG [w]

BE [w]

Process of BE [w]

DAWA [w]

Process of DAWA [w]

« GET »
Read all repair requests [w]

« in »
clientId: string

/repairs

« out »
myRequests: array(Repair)

« PUT »
Update a repair request [w]

« in »
description:string,
repairId:string

/repairs

« out »
OK

« POST »
Create a repair request [w]

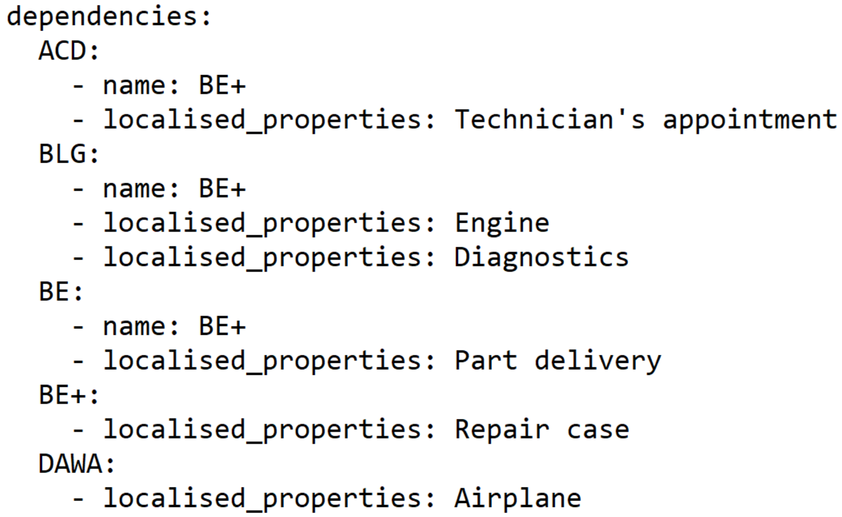
« in »
NewRepair {
date:dateTime,
clientId:string,
planeId:string,
description:string
}

/repairs

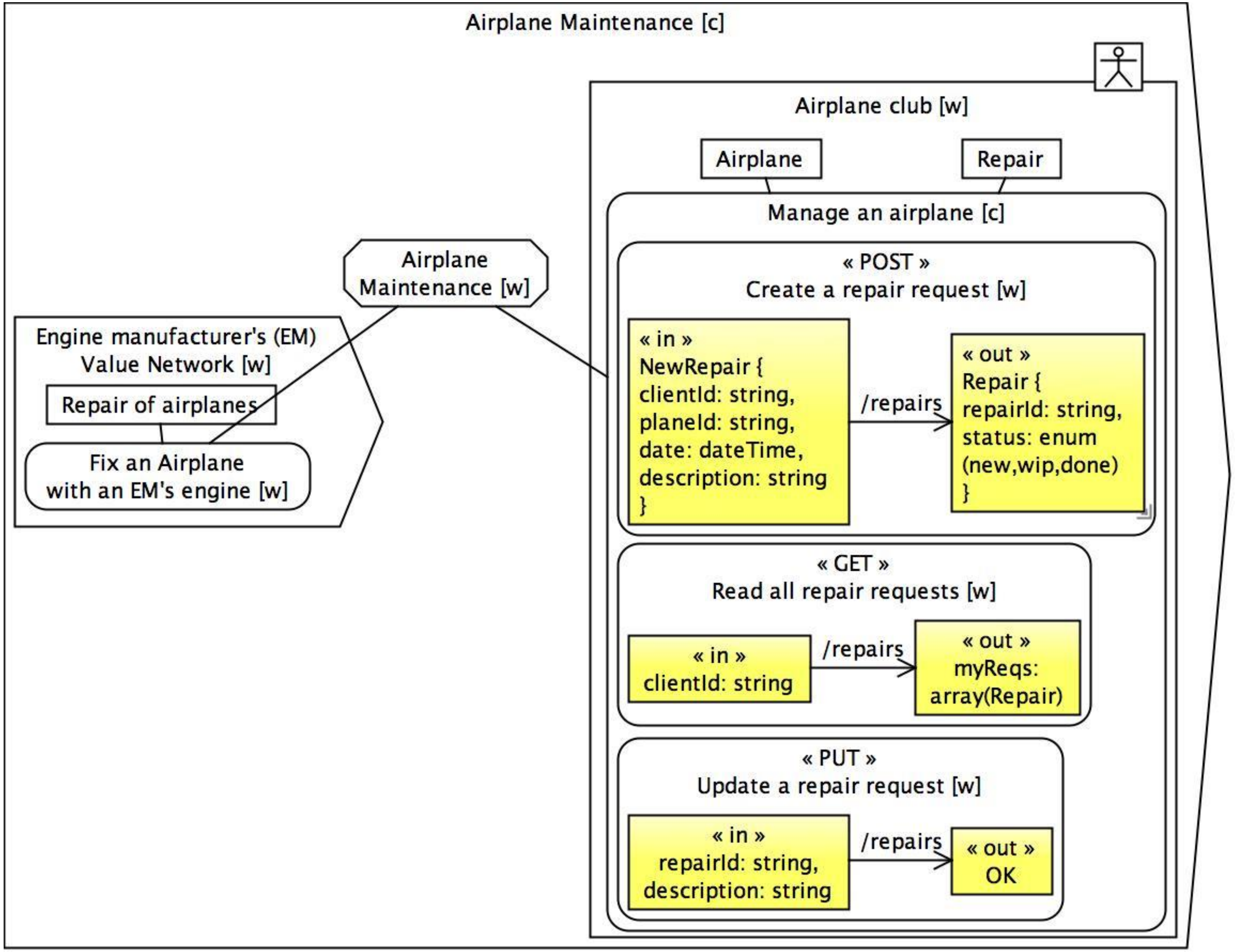
« out »
Repair {
repairId:string,
status:enum(new, wip, done)
}

Additional Annotations

- Biz-to-REST tool generates annotations



```
dependencies:
  ACD:
    - name: BE+
    - localised_properties: Technician's appointment
  BLG:
    - name: BE+
    - localised_properties: Engine
    - localised_properties: Diagnostics
  BE:
    - name: BE+
    - localised_properties: Part delivery
  BE+:
    - localised_properties: Repair case
  DAWA:
    - localised_properties: Airplane
```


Airplane Maintenance [c]
Airplane club [w]
Airplane
Repair
Manage an airplane [c]
Airplane Maintenance [w]
Engine manufacturer's (EM) Value Network [w]
Repair of airplanes
Fix an Airplane with an EM's engine [w]
« POST »
Create a repair request [w]
« in »
NewRepair {
clientId: string,
planeId: string,
date: dateTime,
description: string
}
/repairs
« out »
Repair {
repairId: string,
status: enum
(new,wip,done)
}
« GET »
Read all repair requests [w]
« in »
clientId: string
/repairs
« out »
myReqs:
array(Repair)
« PUT »
Update a repair request [w]
« in »
repairId: string,
description: string
/repairs
« out »
OK

Discussion

- Who do services belong to?
  - Provider, adopter, IT system
- How do we model the data?
- Model-driven anything: is it a good idea?

Next Steps

- Validation of the modeling method

Your Questions?

https://betty.github.io

Thank you!

@bettypirelli