SPARX SYSTEMS CENTRAL EUROPE

ENTERPRISE ARCHITECT

Best Practice Tour 2023

Stockholm
Brussels
Munich
Zurich
Frankfurt
Paris
Warsaw
Berlin
Budapest
Gothenburg

Workshop 2 - Best Practice Tour 2023

March 23, 2023

DSLs for SMEs ☺

Bob Hruska

Principal Consultant

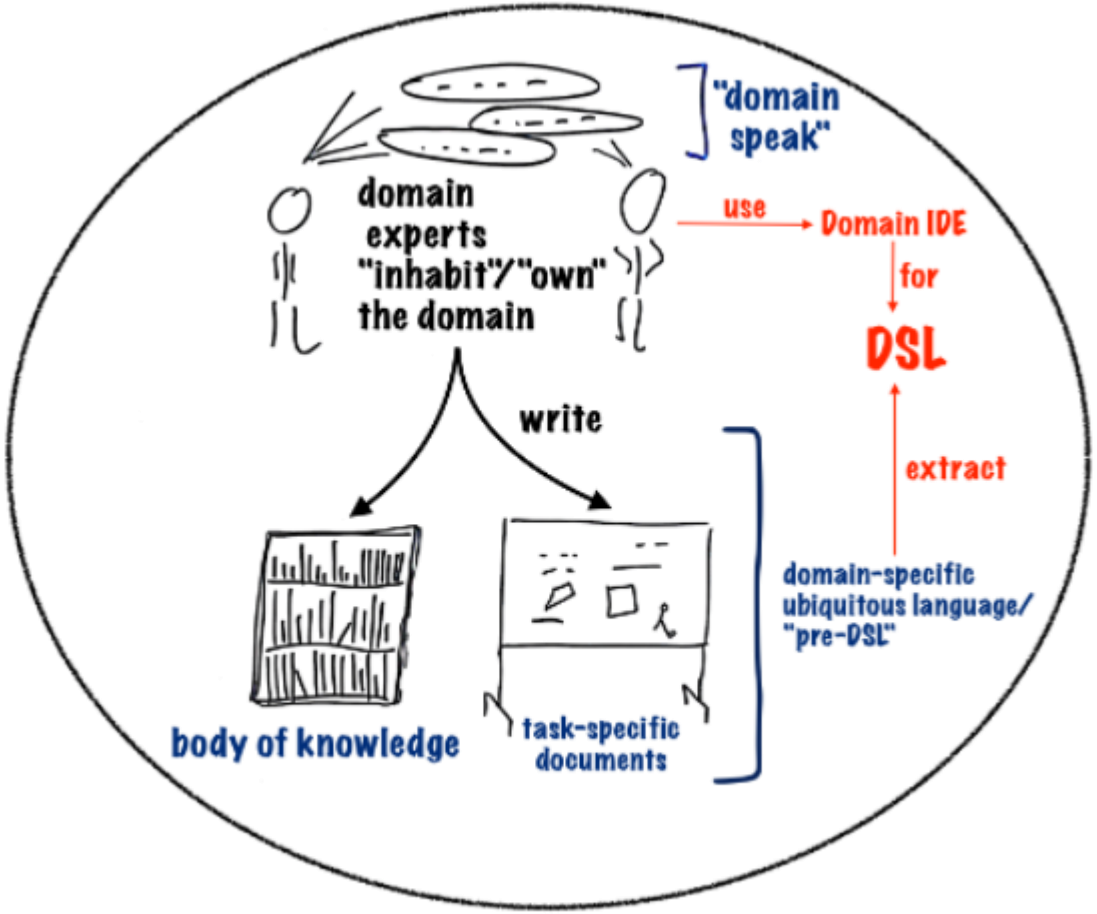
Let’s address these questions:

- Introduction to Domain Specific Languages (DSLs)
- How to establish a common language?
  - bridging GAPs of terms such as AI model, trained AI model, model, etc.
  - Identifying the right modeling approach
  - use of different model languages and how to combine them
- How to integrate the modeling workflow into the development process?
  - which model should be created when and by whom
- How to implement the right modeling approach as a framework into EA?
  - Exploring the real-life example of EU funding project EUREKA PENTA ECOMAI

SPARX SERVICES CENTRAL EUROPE

Domain Specific Languages

- Subject matter experts, or SMEs, are in charge of the knowledge and expertise that form the foundation of software
  - But too often this rich expertise is not captured in a structured way and gets lost when translating it for software developers



https://freecontent.manning.com/the-what-and-why-of-domain-specific-lanugages/

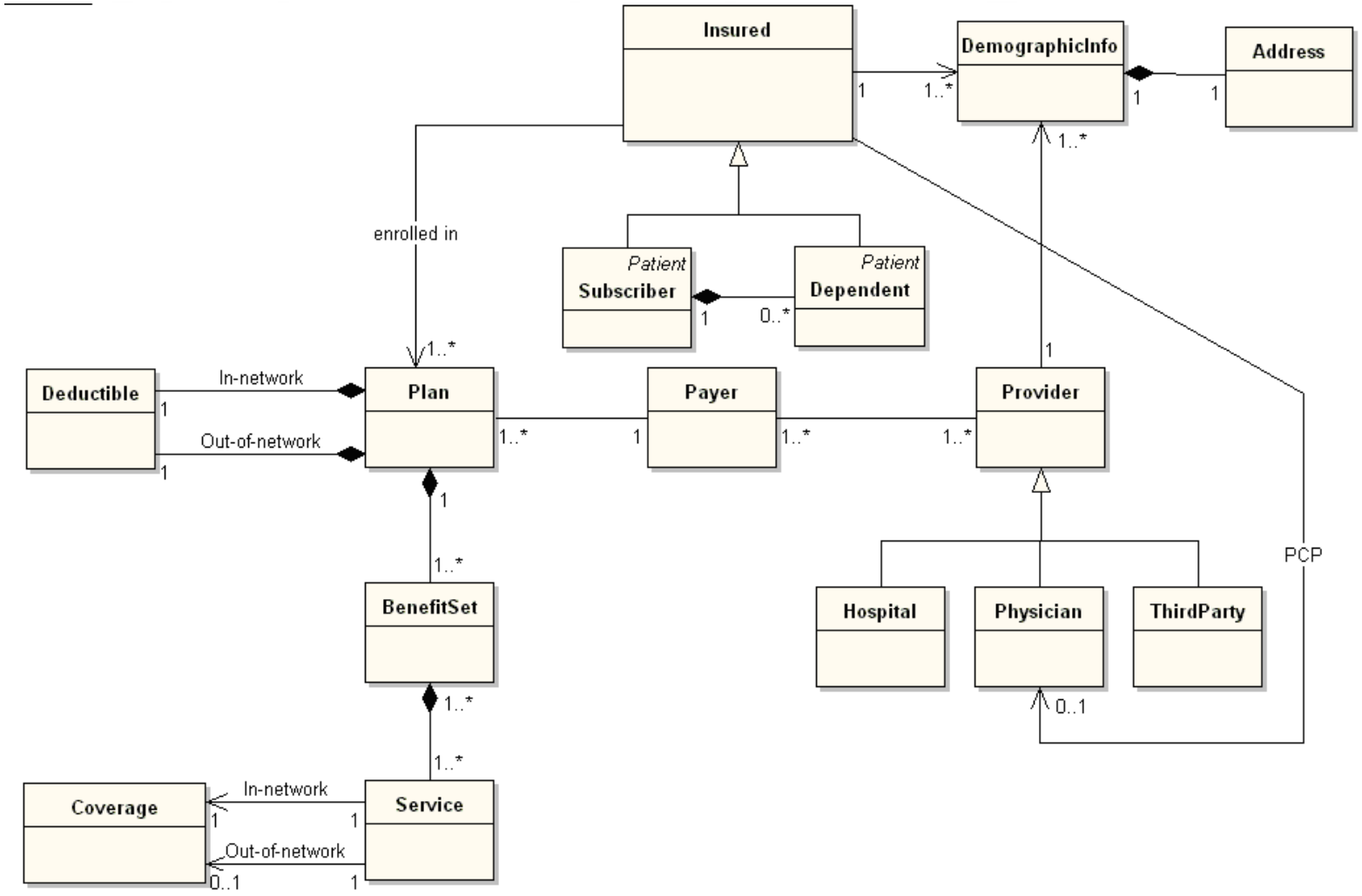
When is something a language?

Glossary

When is something a language?

Glossary

Structured Glossary

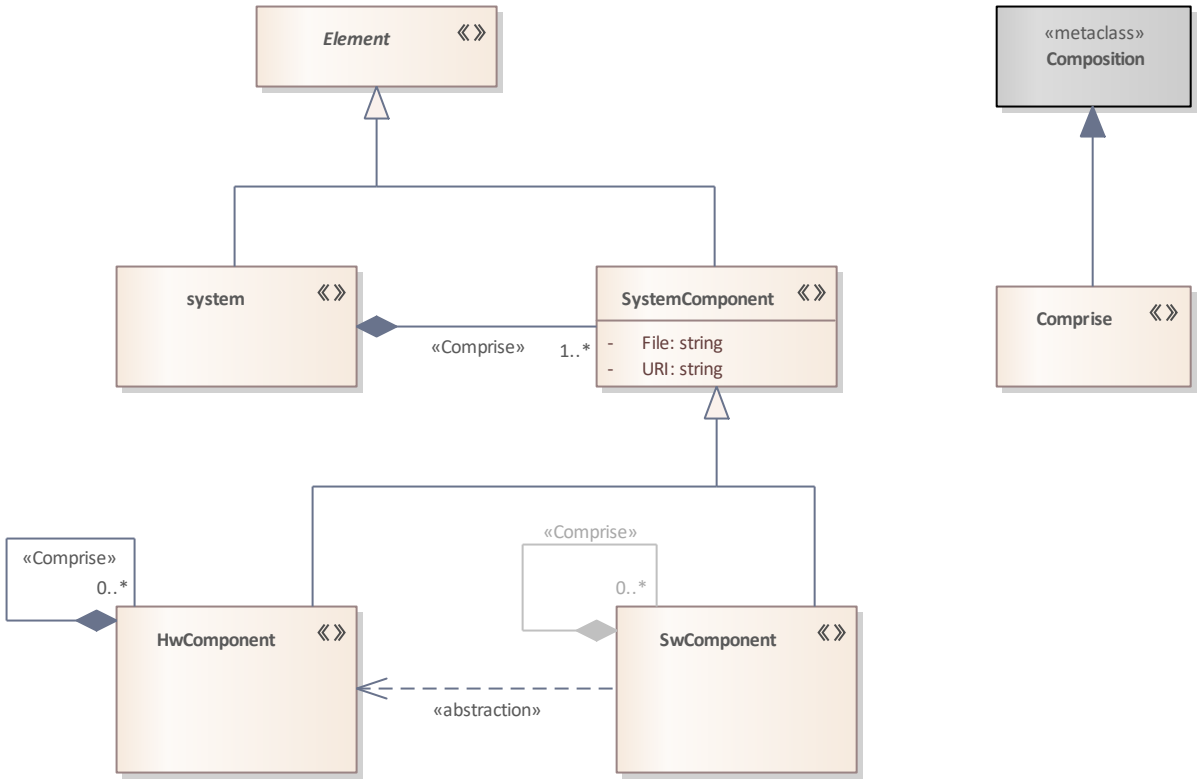


When is something a language?

Glossary

Structured Glossary

Metamodel



When is something a language?

Glossary

Structured Glossary

Metamodel

Validations

```
if
  the X contains a Y
then
  this A over there cannot have
  more than 2 children of type B.
```

When is something a language?

Glossary

Structured Glossary

Metamodel

Validations

Serialisation Format

```
FunCall name="myFun"
  arg: NumLit value="10"
  arg: PlusOp
         arg: NumLit value="4"
         arg: NumLit value="5"
```

```
<FunCall name="myFun">
  <arg><NumLit value="10"/></arg>
  <arg>
    <PlusOp>
      <arg><NumLit value="4"/></arg>
      <arg><NumLit value="5"/></arg>
    </PlusOp>
  </Arg>
</FunCall>
```

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- **Syntax**

```
myFun(10, 4 + 5)
```

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- Syntax
- **Type System**

```
+(int, int)    → int
+(int, real)   → real
+(real, int)   → real
+(real, real)  → real
+(string, *)   → string
+(*, string)   → string

val(<name>, <type>, <init>) → typeof(type)
  # typeof(type) > typeof(init)
```

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- Syntax
- Type System
- **Semantics**

Try typing the phrase "the answer to life the universe and everything" into google...)

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- Syntax
- Type System
- Semantics

Too informal.

When is something a language?

- Glossary
- **Structured Glossary**
- Metamodel
- Validations
- Serialisation Format
- Syntax
- Type System
- Semantics

Too informal.

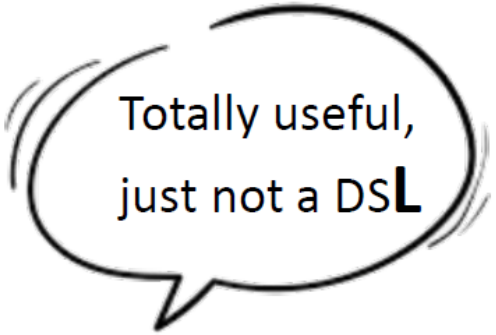
When is something a language?

- Glossary
- Structured Glossary
- **Metamodel**
- Validations
- Serialisation Format
- Syntax
- Type System
- Semantics

That's just a data model.
Or a domain model.
Or an OO structure.
Or a schema.

When is something a language?

- Glossary
- Structured Glossary
- **Metamodel**
- **Validations**
- Serialisation Format
- Syntax
- Type System
- Semantics

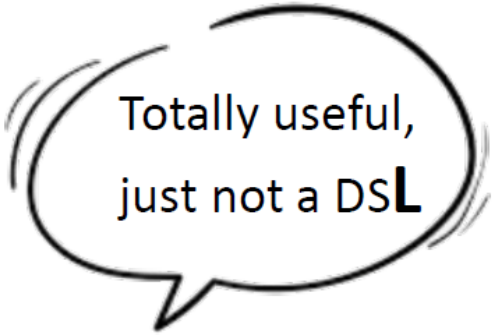


That's just a data model.
Or a domain model.
Or an OO structure.
Or a schema.

With Validations.

When is something a language?

- Glossary
- Structured Glossary
- **Metamodel**
- **Validations**
- **Serialisation Format**
- Syntax
- Type System
- Semantics



That's just a data model.
Or a domain model.
Or an OO structure.
Or a schema.

With Validations.
And a way to store.

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- Syntax
- Type System
- Semantics

Finally, a language!

It‘s about syntax, stupid!

When is something a language?

- Glossary
- Structured Glossary
- Metamodel
- Validations
- Serialisation Format
- Syntax
- Type System
- Semantics

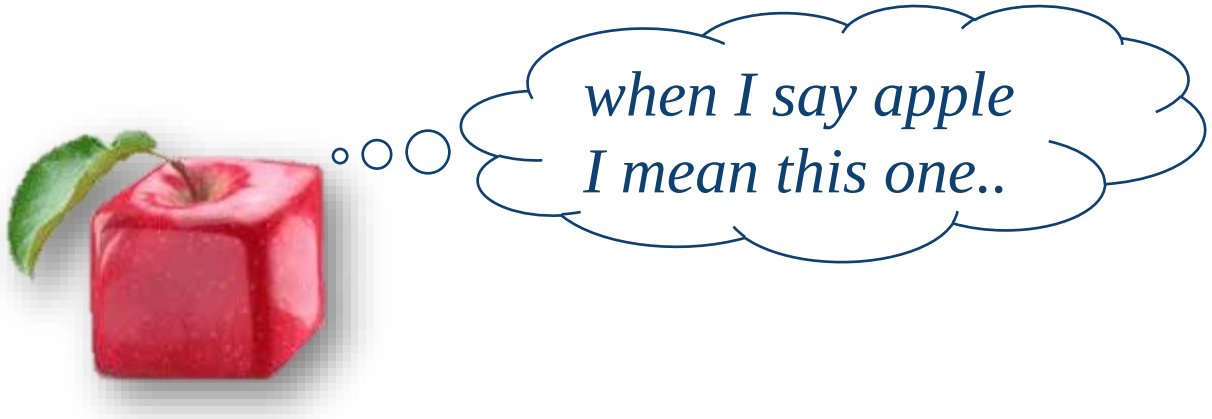
A serious language :-)

Find more details at:

https://medium.com/@markusvoelter/when-is-something-a-domain-specific-language-83b7eff79ed4

Establish the common language

- Create and use a glossary of project terms (**domain model**…)
  - Synonyms are excellent to reconcile different languages
- Use it consistently in all communication within/about the project
  - Inside the team …
  - With other partners …



Common language facilitates communication and avoids confusion

SPARX SERVICES CENTRAL EUROPE

Baseline Application Description

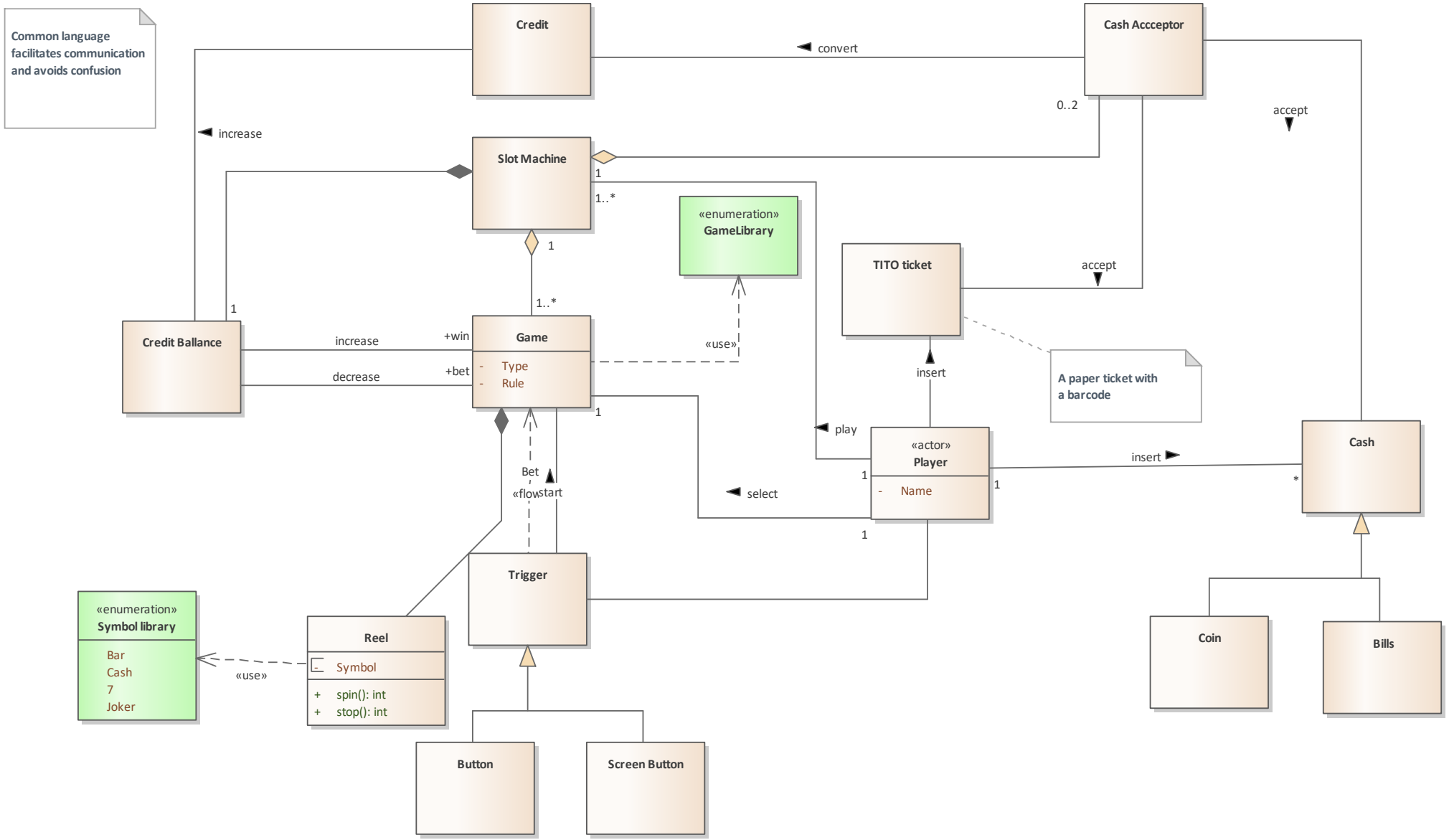
- A gaming machine allowing clients to buy slot machine credits the player cash or some other sort of value, win or lose, and cash in their credits.

SPARX SERVICES CENTRAL EUROPE

How to Create a Slot Machine Domain Model

- A **person playing** a **slot machine** can insert **cash** and insert **TITO tickets** (a **paper ticket** with a **barcode**), into a designated **slots** on the machine.
- The machine can support multiple games and the player can select which game to play
- The game is then activated by means of a button, or on newer machines, by pressing a touchscreen on its face.
- The game itself does not involved skill on the player's part
- The objective is to get the players to play
  - The game usually involves matching symbols, either on mechanical reels that spin and stop to reveal one or several symbols, or on simulated reels shown on a video screen.
  - Most games have a variety of winning combination of symbols, often posted on the face of the machine. If a player matches a combination according to the rules of the game, the slot machine credits the player, such as free spins or extra games.

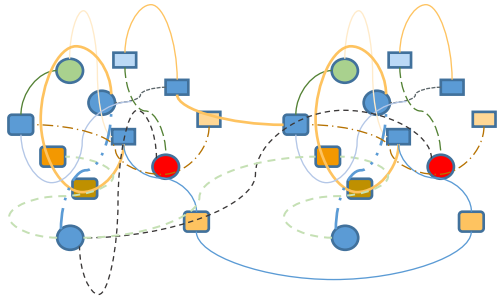
SPARX SERVICES CENTRAL EUROPE



SPARX SERVICES CENTRAL EUROPE

What the Modelling Approach consists of?

Models are structured data with a graphical representation!
What is the structure of this?



To prevent models like this, we **need a modelling approach**!

Who reads the model?

What information should be read from the model?

Modelling Approach

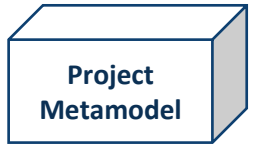
Method

Process

*** Which Artifact should be created**

*** When**

*** Who's responsible for that step**



What language elements should be used? *

What connectors should be used? *

What element should be connected with which connector? *

Define additional Stereotypes and Tags if required *

Approach specific subset of the **language & required extensions for the Domain (DSL)**

How to model structure (Packages) should look like? *

Which elements should be contained in which Package? *

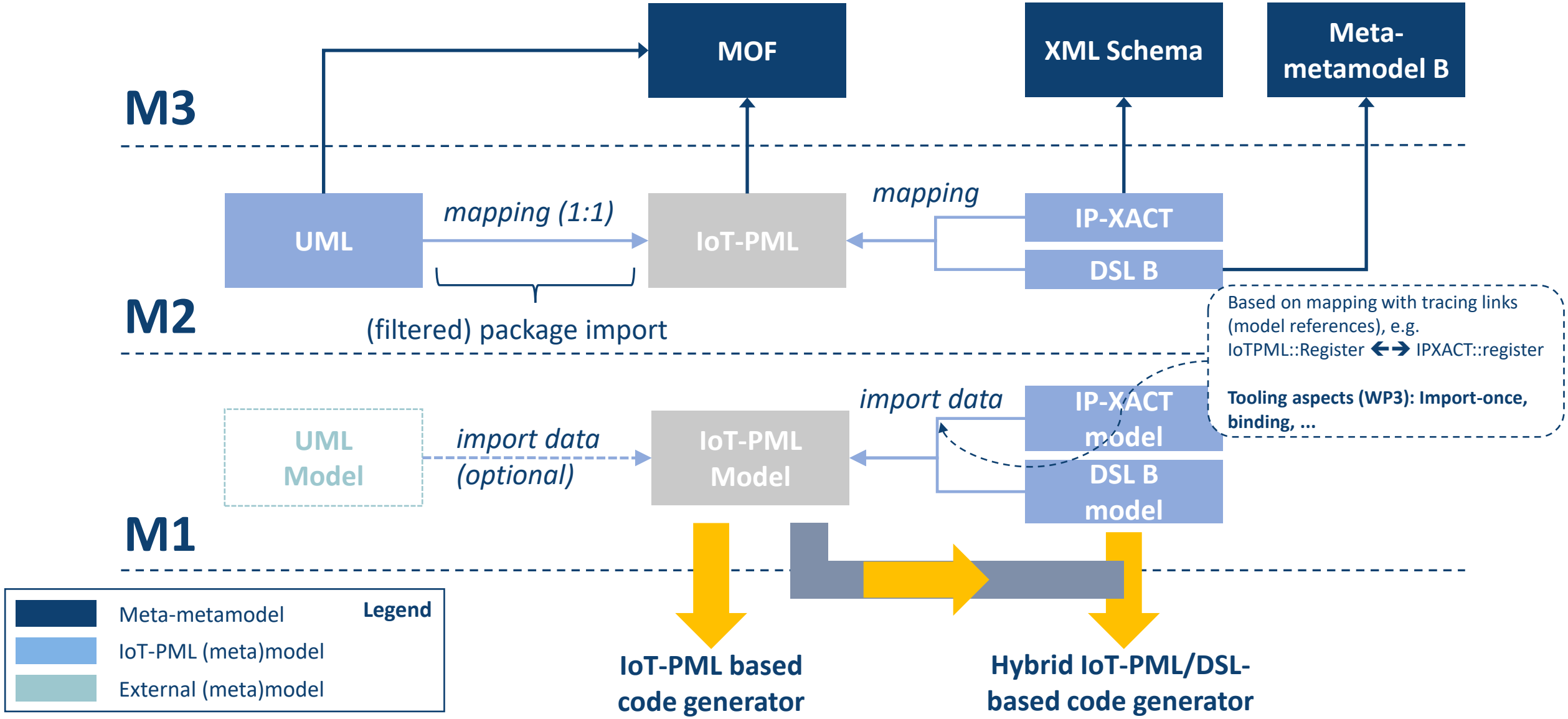
Approach specific **model structure**

Define additional rules, constraints and guidelines *

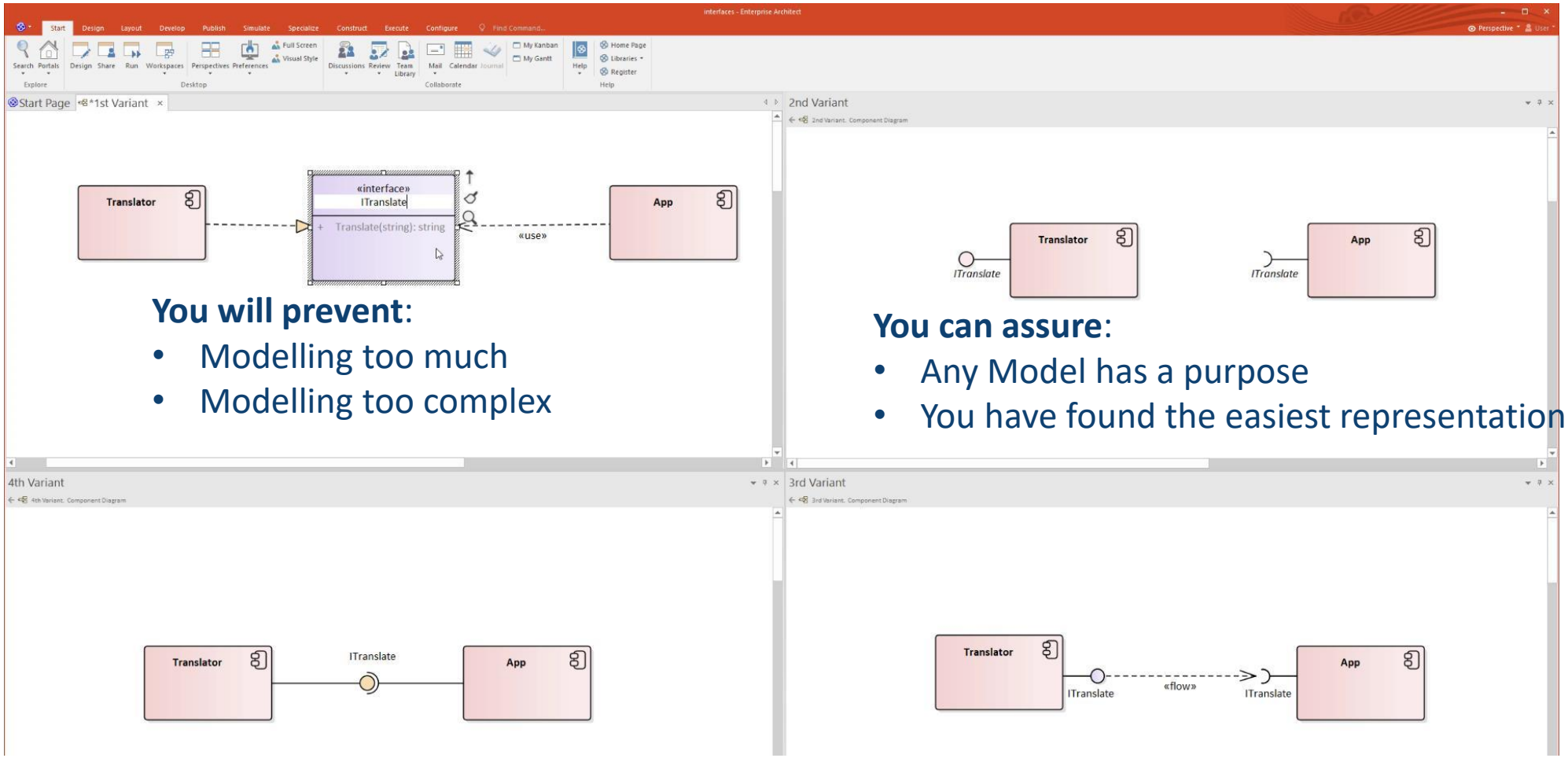
Approach specific **Governance**

SPARX SERVICES CENTRAL EUROPE

Identify the “right” modeling languages



Identify the right Modelling Approach



You will prevent:

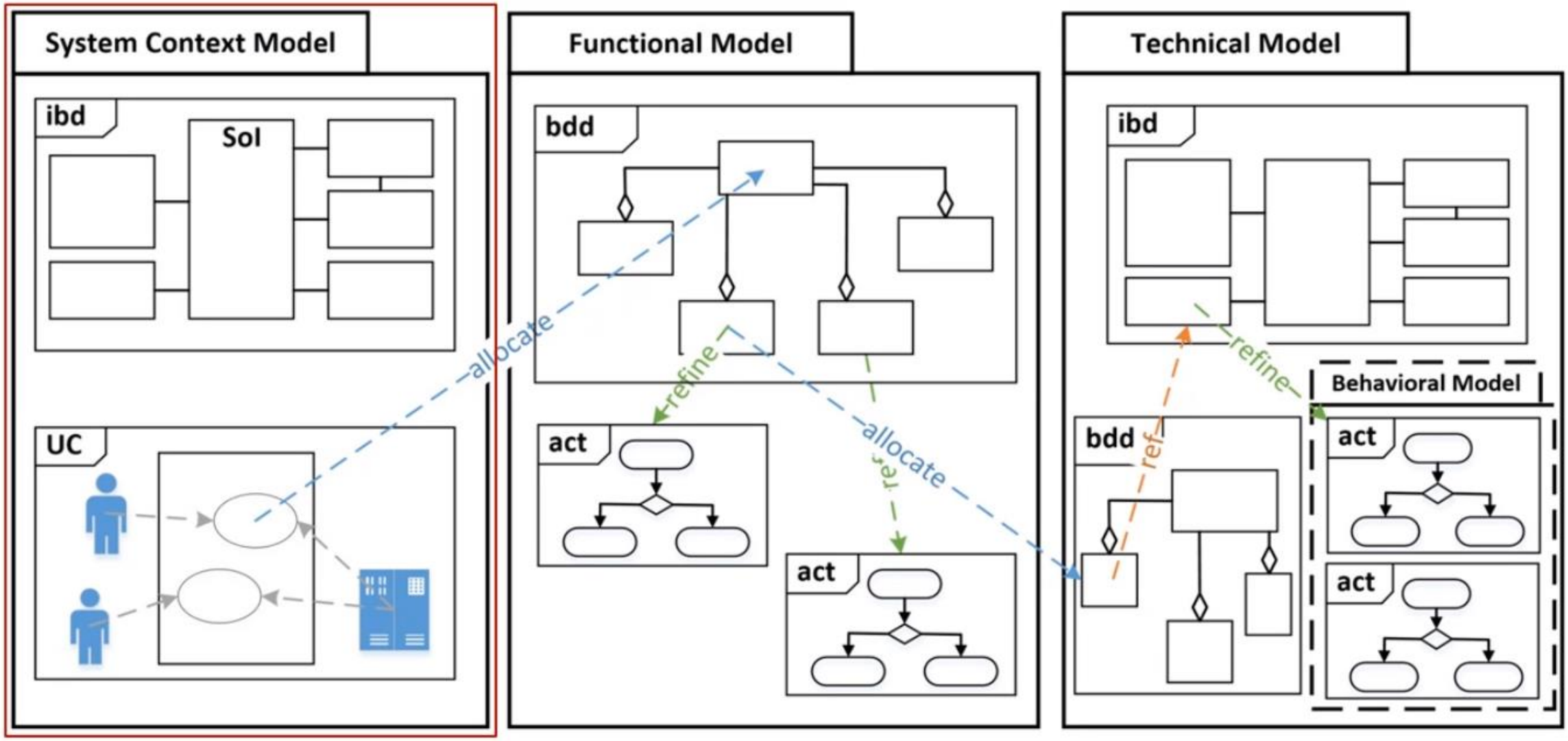
- Modelling too much
- Modelling too complex

You can assure:

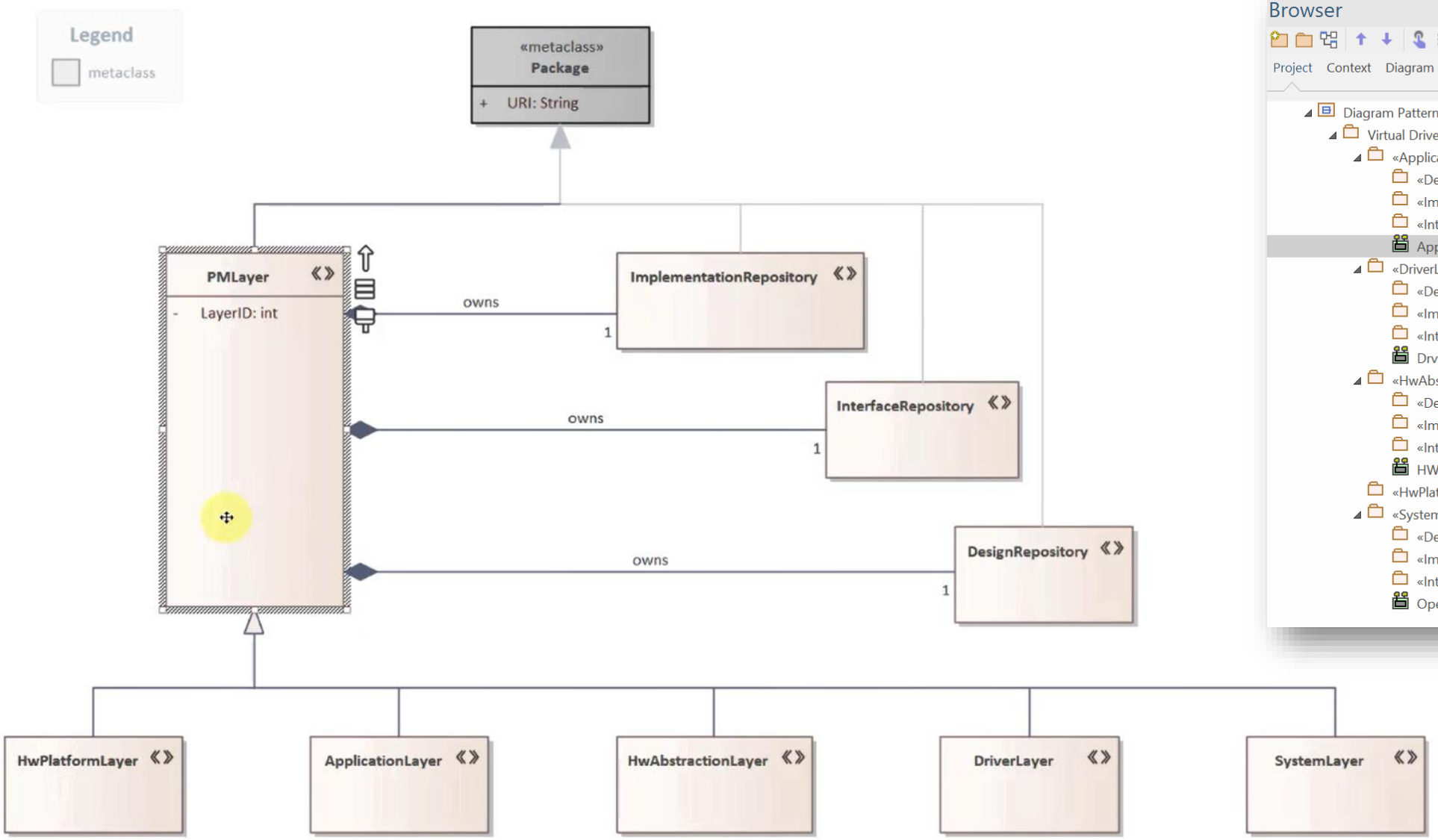
- Any Model has a purpose
- You have found the easiest representation

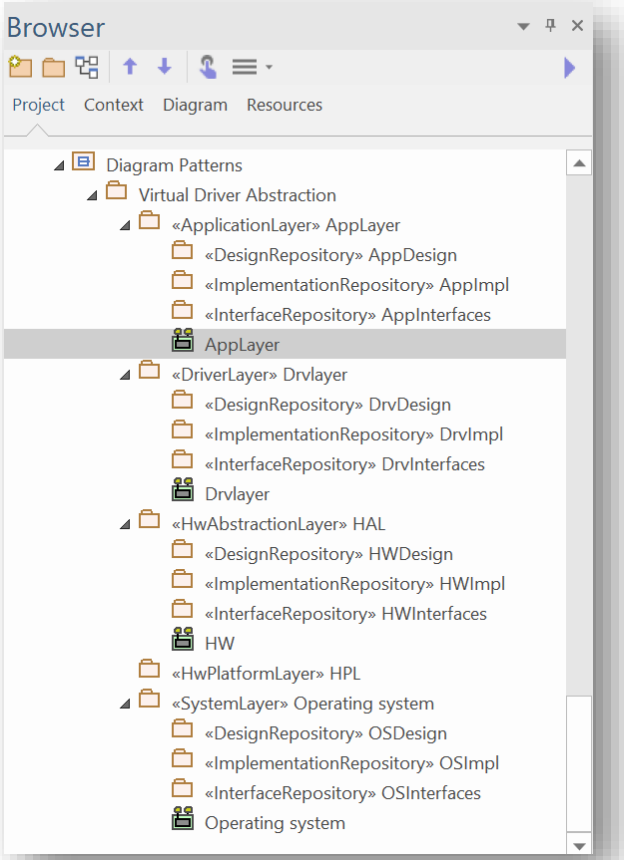
The more questions we have,
the easier it is to find the required language elements!

Identify the “right” modeling workflow

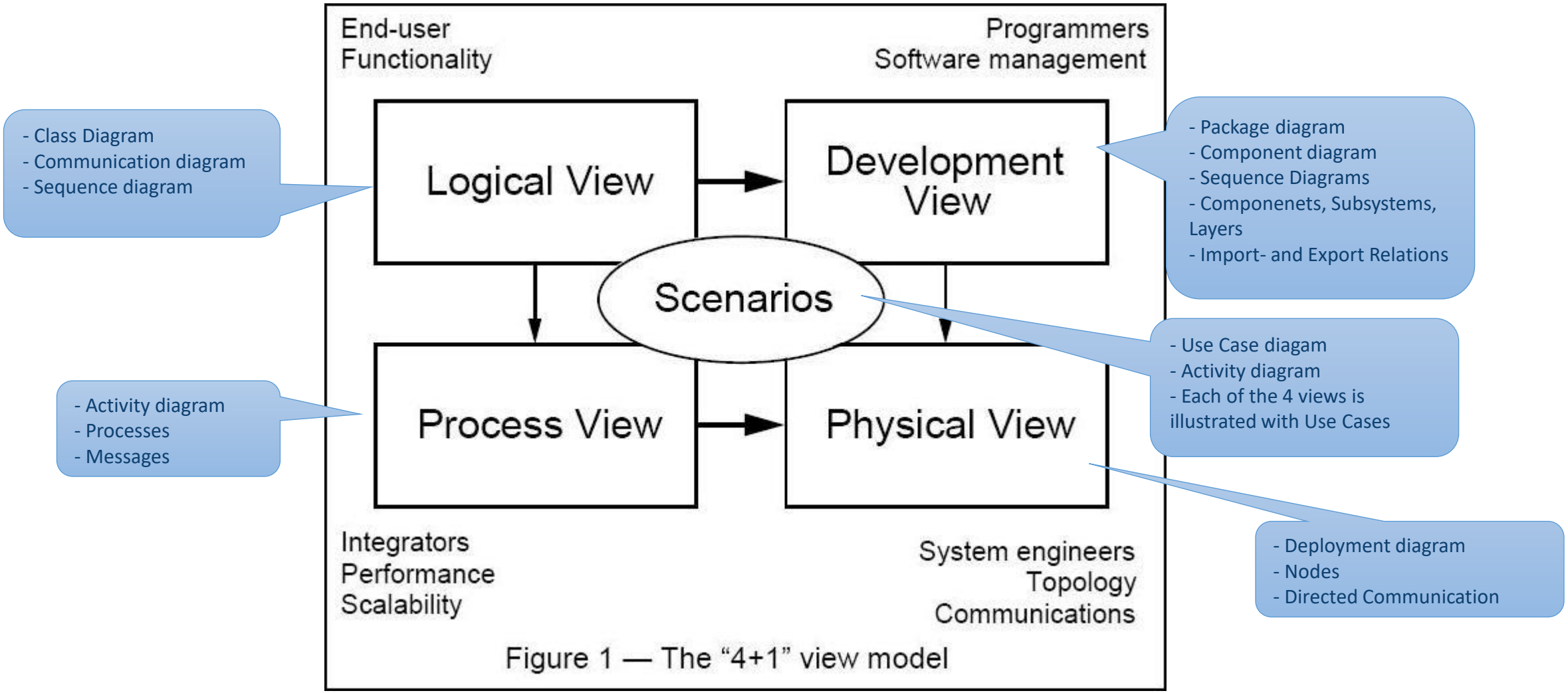


IoT-PML metamodel example





Let’s stay on course…



Philippe Kruchten: The "4 + 1" View Model of Software Architecture, IEEE Software 12 (6), November 1995

1st Iteration Architectural Viewpoints 1/3

Architectural View	Stakeholder Concerns (What does the view help to answer?)	Stakeholders						
		Architect	Developer	Integrator	Tester	Maintainer	Product Manager	Customer
Context View	• Who (users, external systems) is interacting with the system? • How does the system fit into the existing environment? • What kind of information is required and delivered by the system?	X	X	X	X	X	X	X
Conceptual View	• What are the functional capabilities of the system? • How does the system work on a high-level? • How is the system structured conceptually? • What are the operating modes of the system during a run-time phase? • What kind of information is required and delivered by the conceptual elements?					X	X	X
Subsystem View	• What subsystems (software, electronical, mechanical, etc.) compose the system? • What dependencies exist between subsystems?	X		X		X	X	

1st Iteration of Architectural Viewpoints 2/3

Architectural View	Stakeholder Concerns (What does the view help to answer?)	Stakeholders						
		Architect	Developer	Integrator	Tester	Maintainer	Product Manager	Customer
Subsystem Interaction View	• What kind of information is required and delivered by the subsystems? • What interaction points does each subsystem expose? • How do subsystems interact in order to provide a certain behavior or functionality? • How does the system work?	X	X	X	X	X	X	
Software Implementation View	• Which software structural elements will be developed in-house and which will be acquired or reused? • How are concepts like libraries, configuration files, assemblies, and executables used?	X	X	X	X	X	X	
Software Deployment View	• What is the run-time configuration of processing hardware nodes and the execution environment running on those notes? • How are the implementation elements distributed across physical hardware nodes and execution environment?	X	X	X	X	X	X	

1st Iteration of Architectural Viewpoints 3/3

Architectural View	Stakeholder Concerns (What does the view help to answer?)	Stakeholders						
		Architect	Developer	Integrator	Tester	Maintainer	Product Manager	Customer
Software Structural View	• What logical structural elements compose the software system? • What interfaces are provided and required by structural elements? • What structural elements are nested together? • Are all key usage scenarios covered by the identified structural elements? • Which structural elements need to be built first for better scheduling? • Which structural elements bring higher risk? • Which functional units need more attention? • What are the key integration points between functional units/modules? • How do different elements interact with each other to satisfy key usage scenarios? • What kind of information is required and delivered by the structural elements?	X	X	X	X	X		

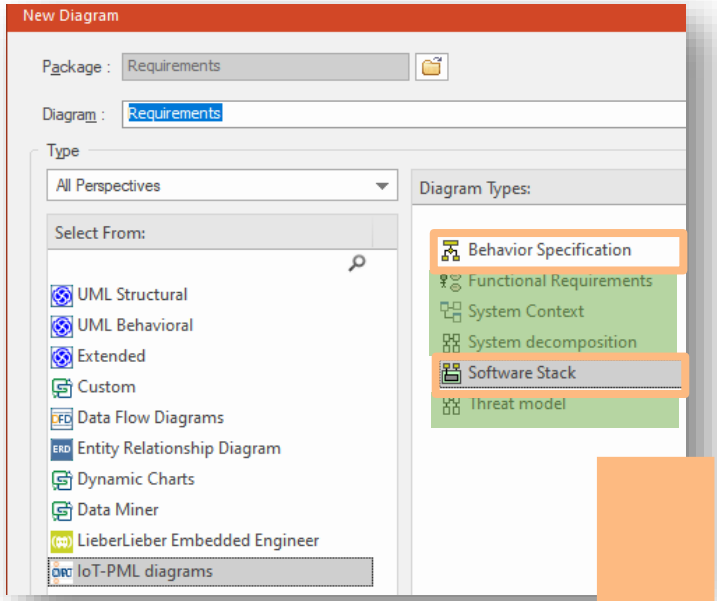
Final IoT-PML Architectural Viewpoints

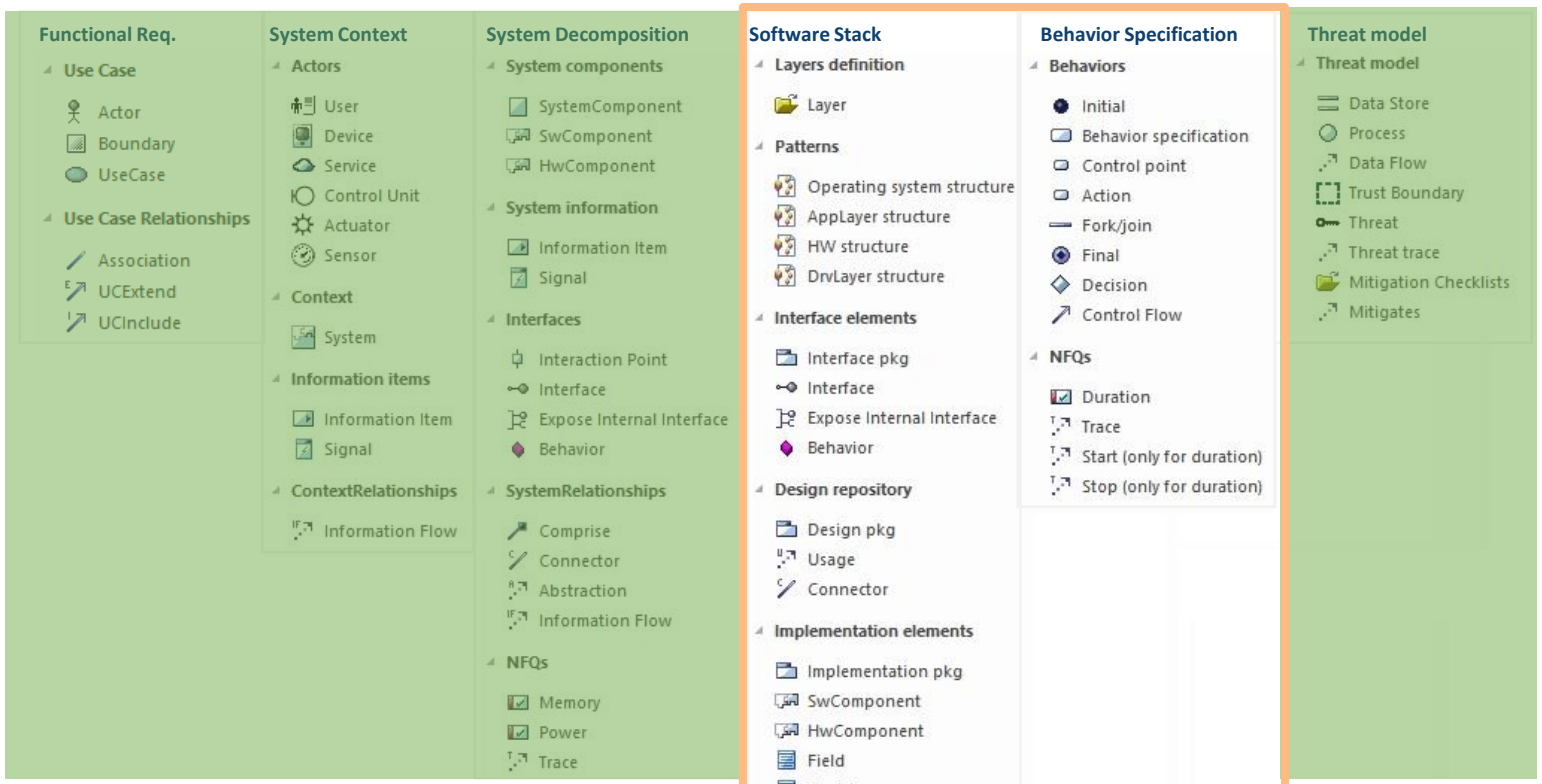
Architectural Viewpoint	Stakeholders							Note
	Architect	Developer	Integrator	Tester	Maintainer	Product Manager	Customer	
Requirements	X	X		X		X	X	Systems engineering
System Context	X	X	X	X	X	X	X	Systems engineering
System Decomposition					X	X	X	Systems engineering
Software Stack	X	X	X	X	X	X		IoT-PML
Cybersecurity	X	X					X	Threat modeling

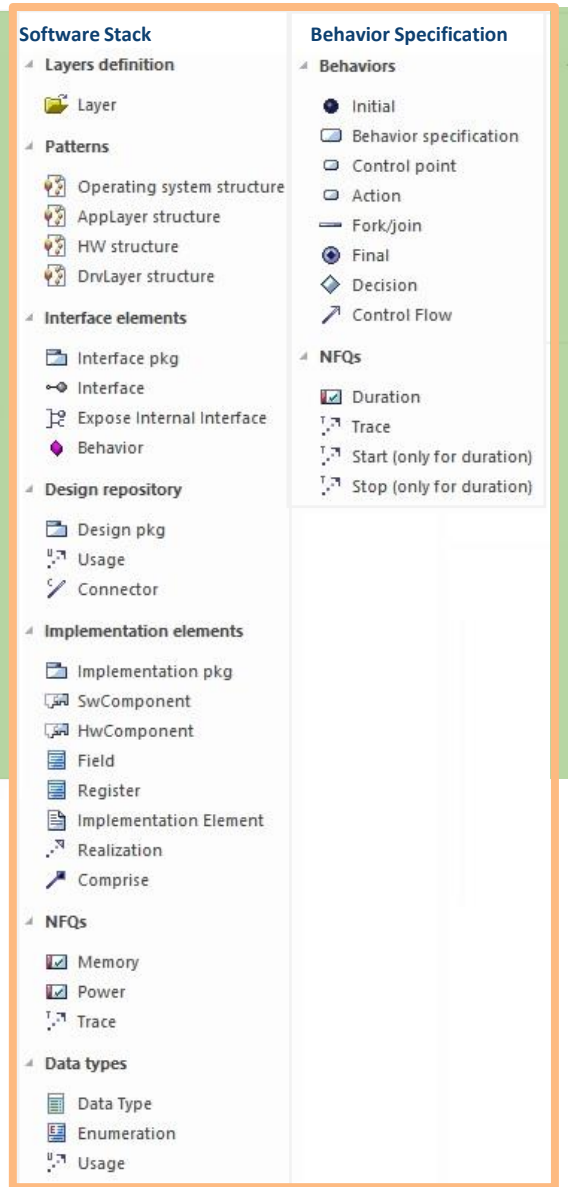
IoT-PML

SPARX SERVICES CENTRAL EUROPE

IoT-PML implementation in EA

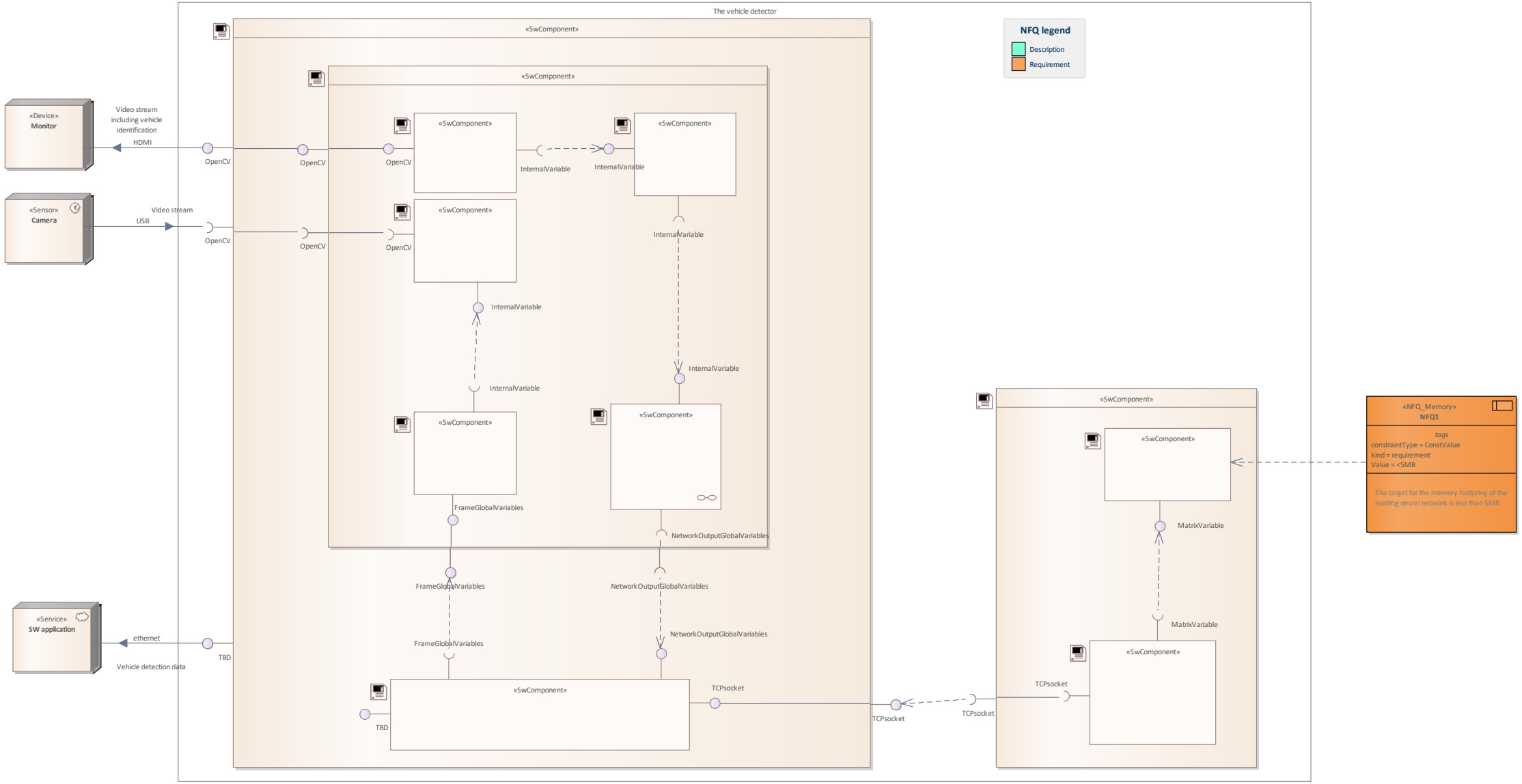






IoT-PML

IoT-PML example diagram

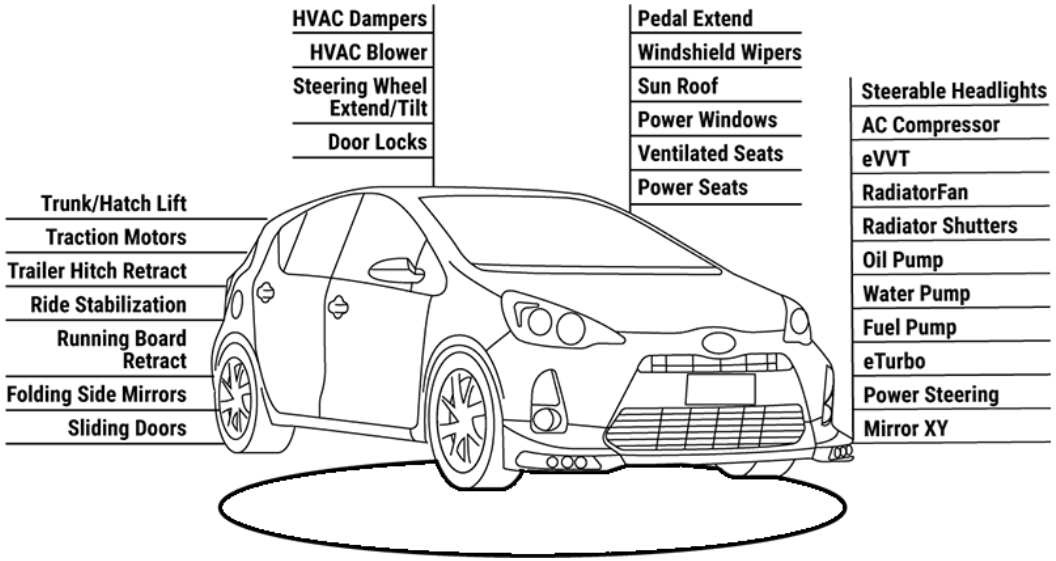


Ecological Motor Control and Predictive Maintenance with AI

ECOMAI has started in April 2022

Mission

- **ECOMAI project is developing technologies**
- to enhance electric motor drive systems with an embedded AI system running on a specialized AI hardware platform
- to optimize the efficiency and lifetime of electric motors, thereby reducing energy consumption and enabling development of more ‘ecological’ systems
- to lead to market opportunities for applications in numerous sectors including automotive, medical and transportation.



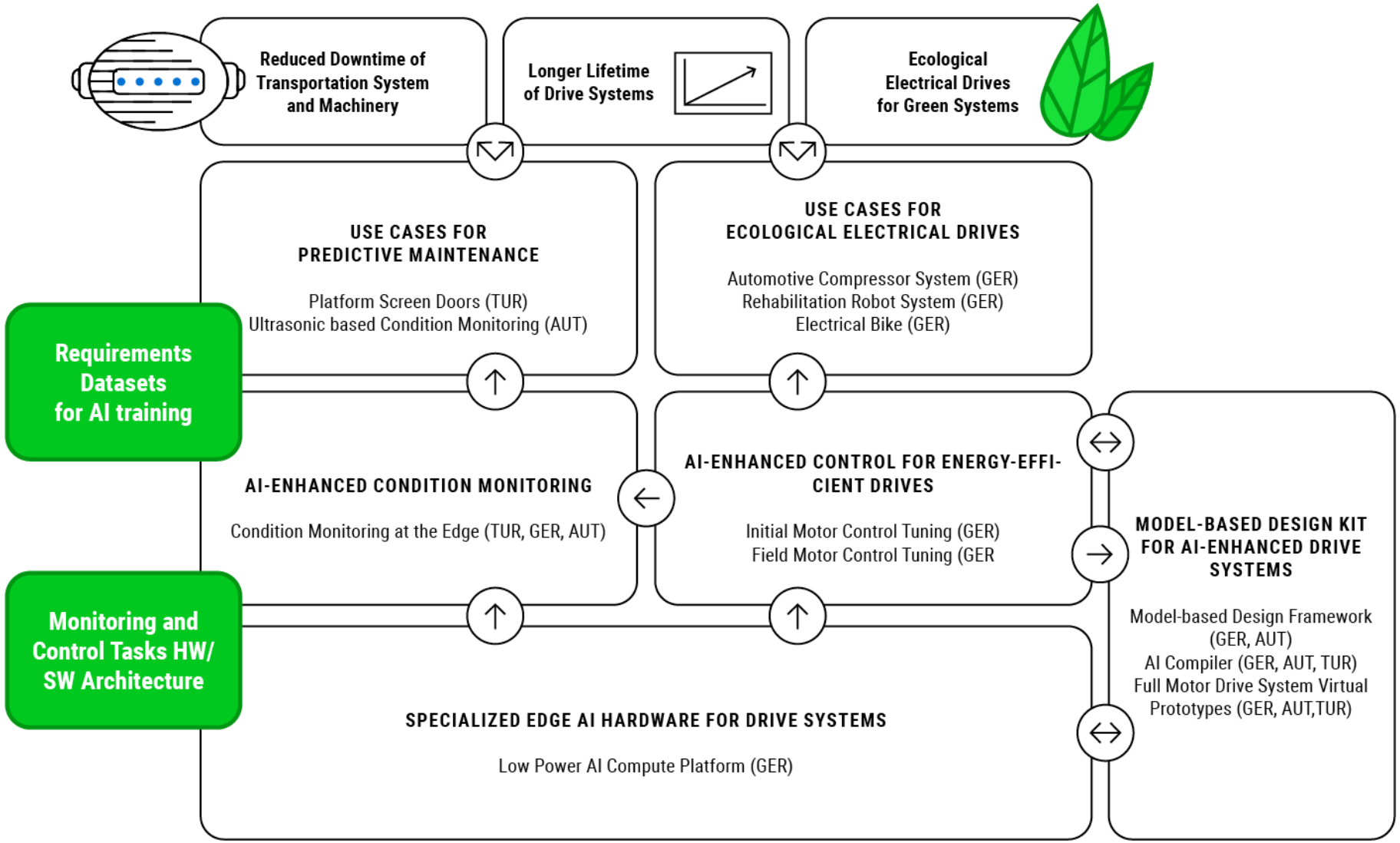
SPARX SERVICES CENTRAL EUROPE

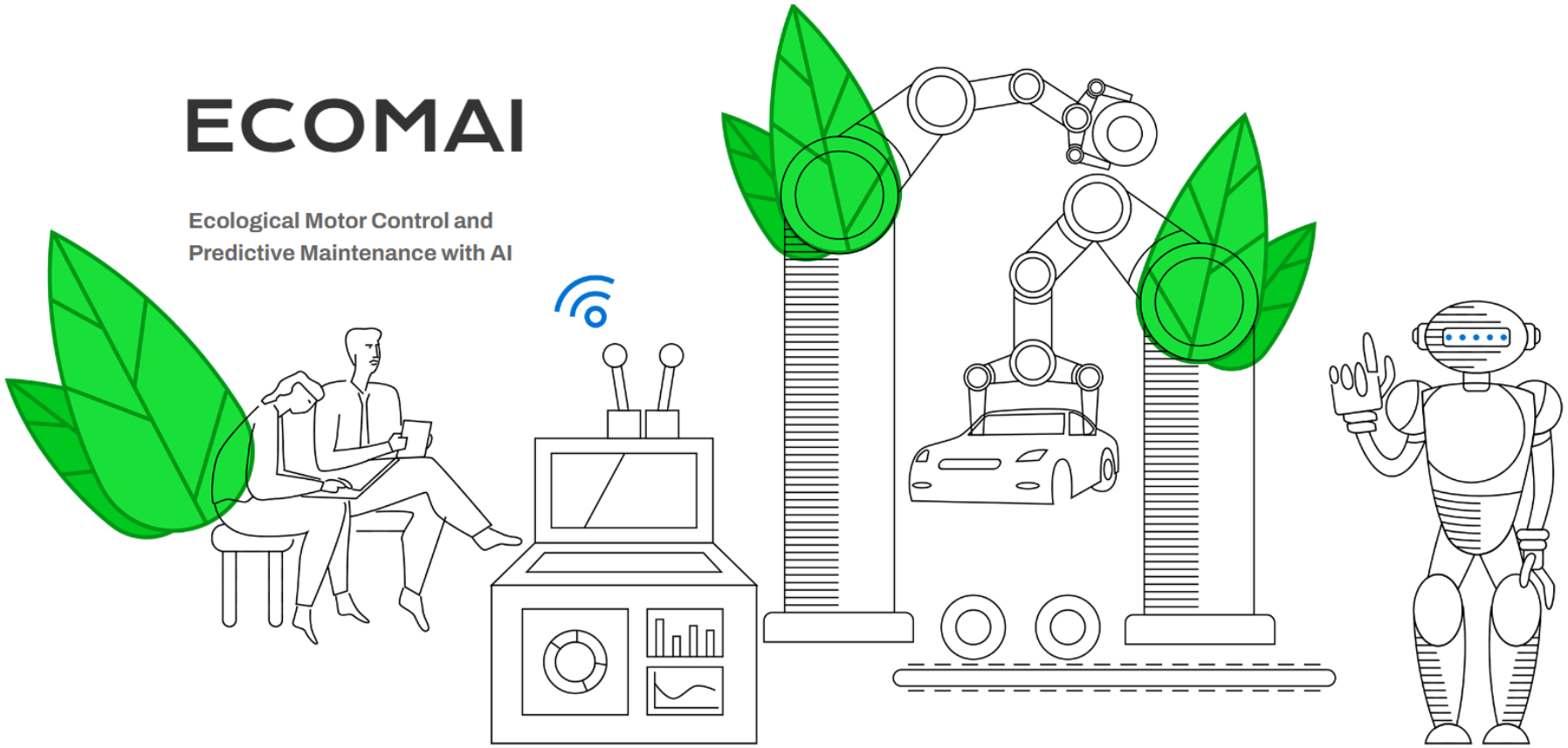
Consortia

- **Germany:**
  - Infineon Technologies AG (IFX)
  - MOTEON
  - *FEAAM*
  - Technical University of Munich
  - Technical University of Ilmenau
- **Austria**
  - SparxSystems (SPARX): Model-based Design
    - Subcontractor: Software Center Hagenberg GmbH (SCCH)
- **Turkey**
  - Albayrak ltd., Railway Platform Screen Doors,
    - Subcontractor: Eskisehir Osmangazi University, (Eyup Cinar eyup.cinar@ogu.edu.tr)

SPARX SERVICES CENTRAL EUROPE

Technology Value Chain and Use Cases





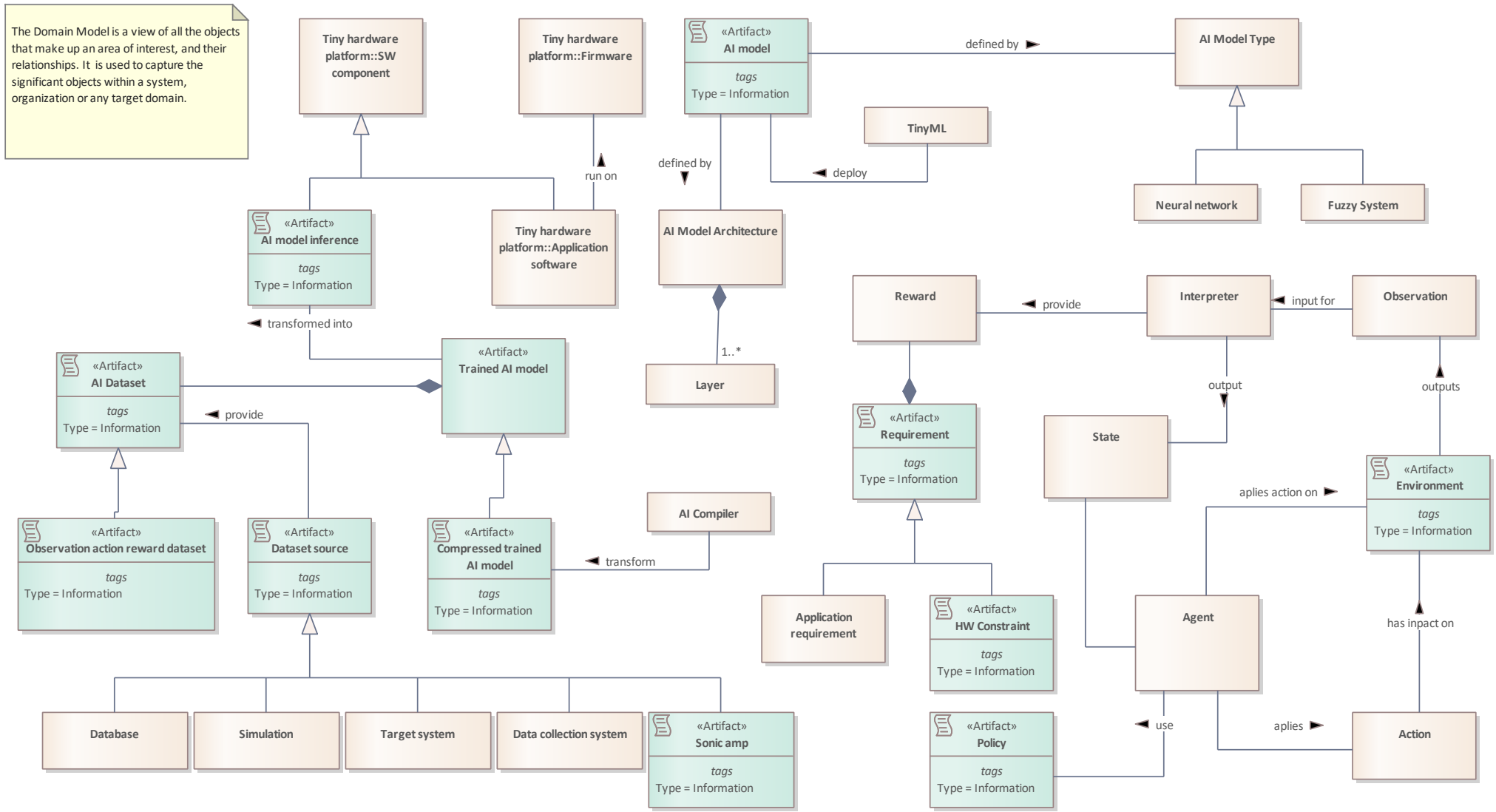
https://ecomai.eu

WP5: Embedded AI SW And Development Kit (SPARX)

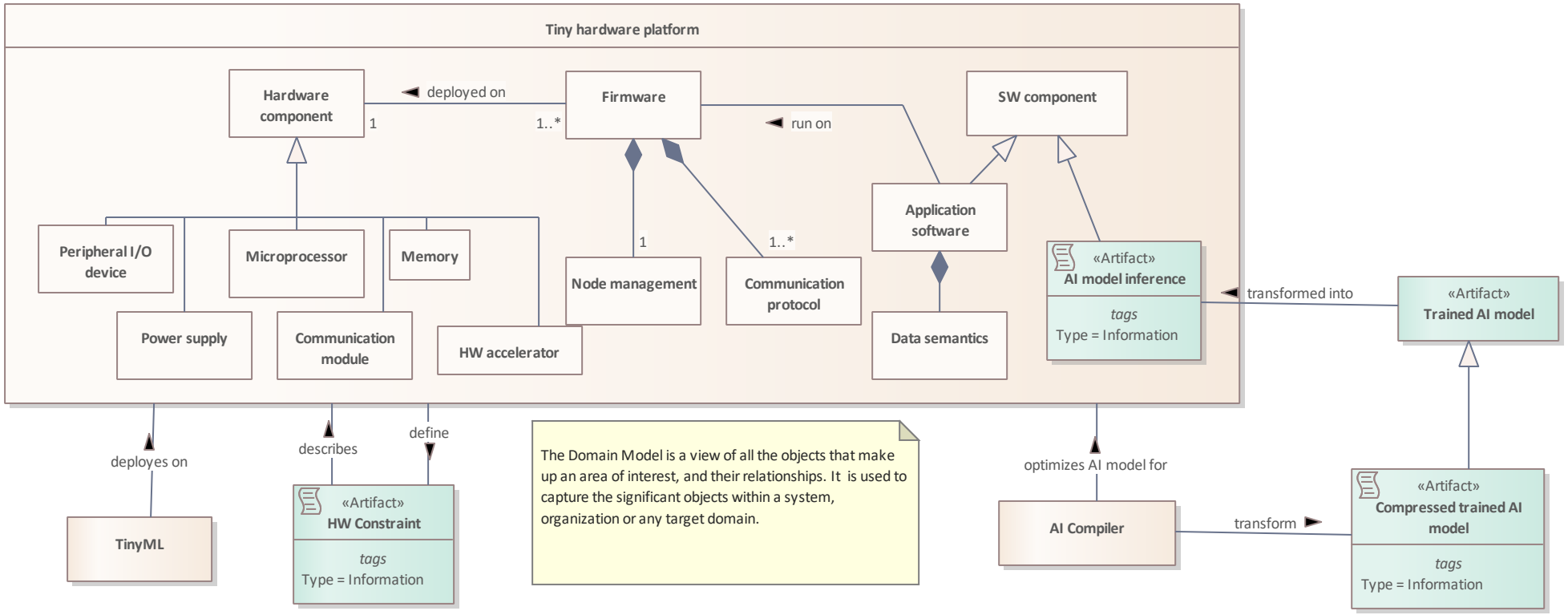
- T5.1 Software/AI Model Development for Condition Monitoring and Predictive Maintenance (Alb) (Alb, UsePAT, IFX)
- T5.2 Software/AI Model Development for Ecological Motor Drives (TUIL) (MTN, TUIL, IFX, FEAAM)
- T5.3 Model-based Design Environment for AI-enhanced Drive Systems (SPARX)

SPARX SERVICES CENTRAL EUROPE

ECOMAI taxonomy 1st iteration



ECOMAI taxonomy 1st iteration



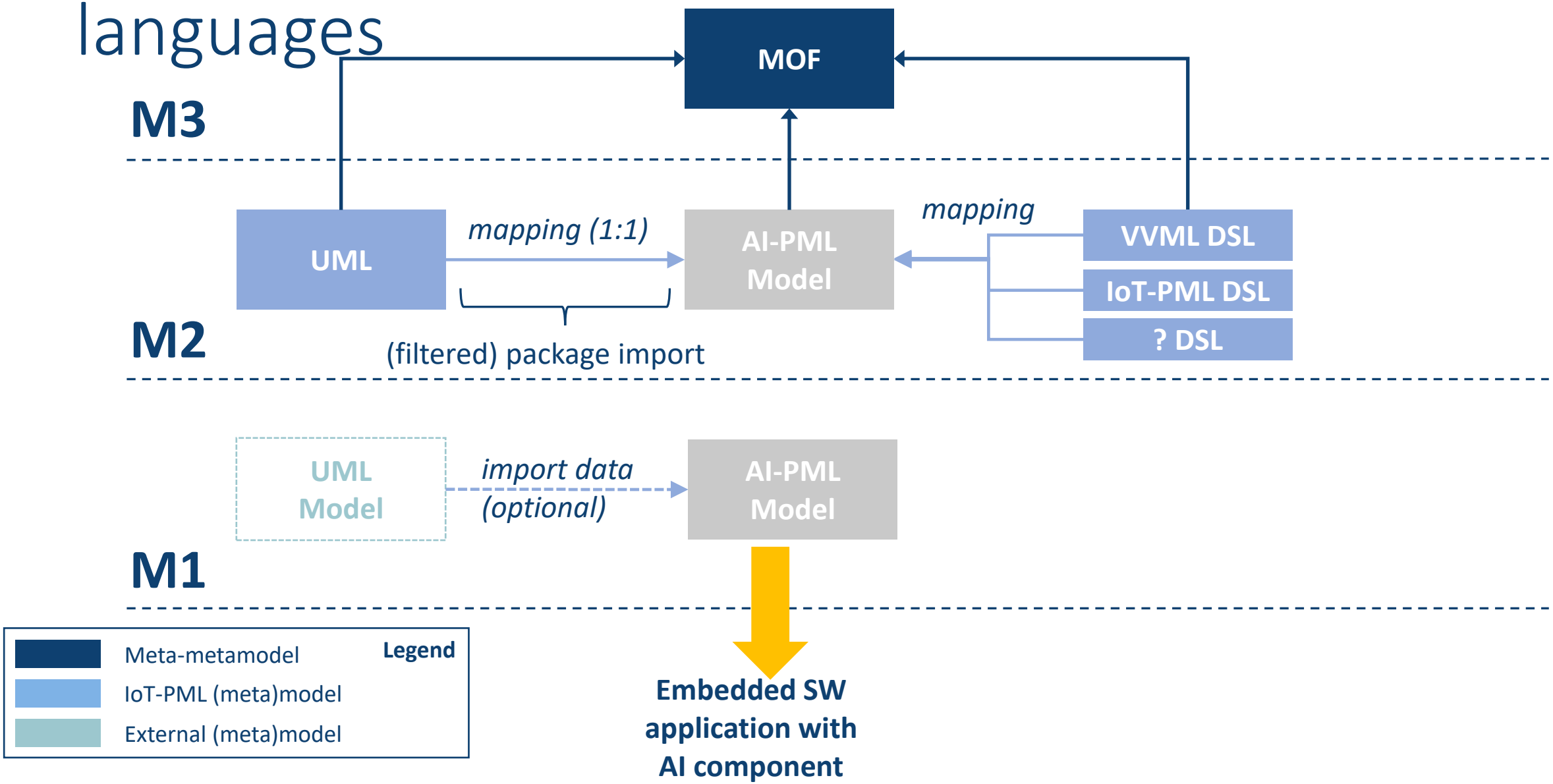
SPARX SERVICES CENTRAL EUROPE

Let’s not re-invent the wheel, shall we?

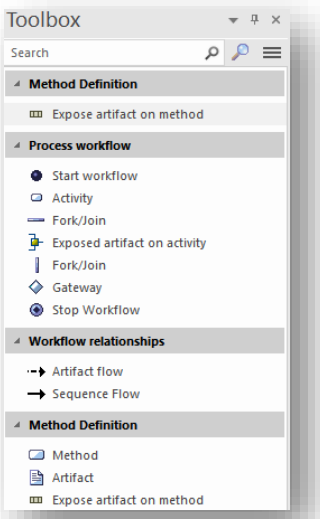
- **VVML (VALU3S)**
  - VVML is **domain-specific language** (DSL) for describing validation and verification activities
  - Design of re-usable workflow assets such as V&V activities and artifacts that are exchanged between workflows
  - 2 levels of modelling: method definition and workflow specification
- **IoT-PML (COMPACT)**
  - is a **DSL** suitable to **IoT nodes**, which is implemented as the UML profile.
  - IoT-PML supports both, top-down and bottom-up design flow or its combination.
- **Advantages**
  - Based on a **simple, known, standardized modelling notation**
  - Implemented into EA as a framework using MDG Technology

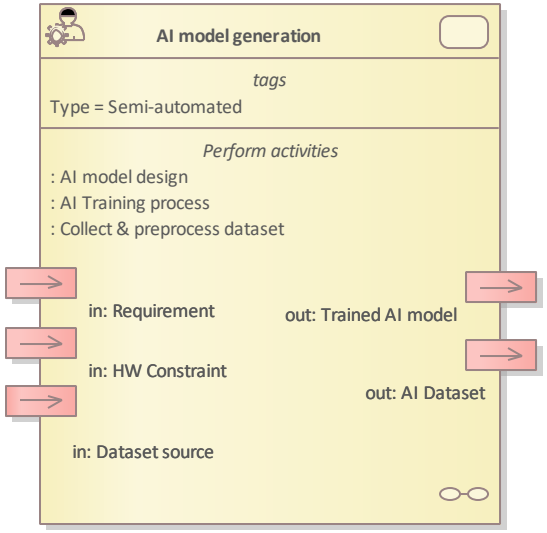
SPARX SERVICES CENTRAL EUROPE

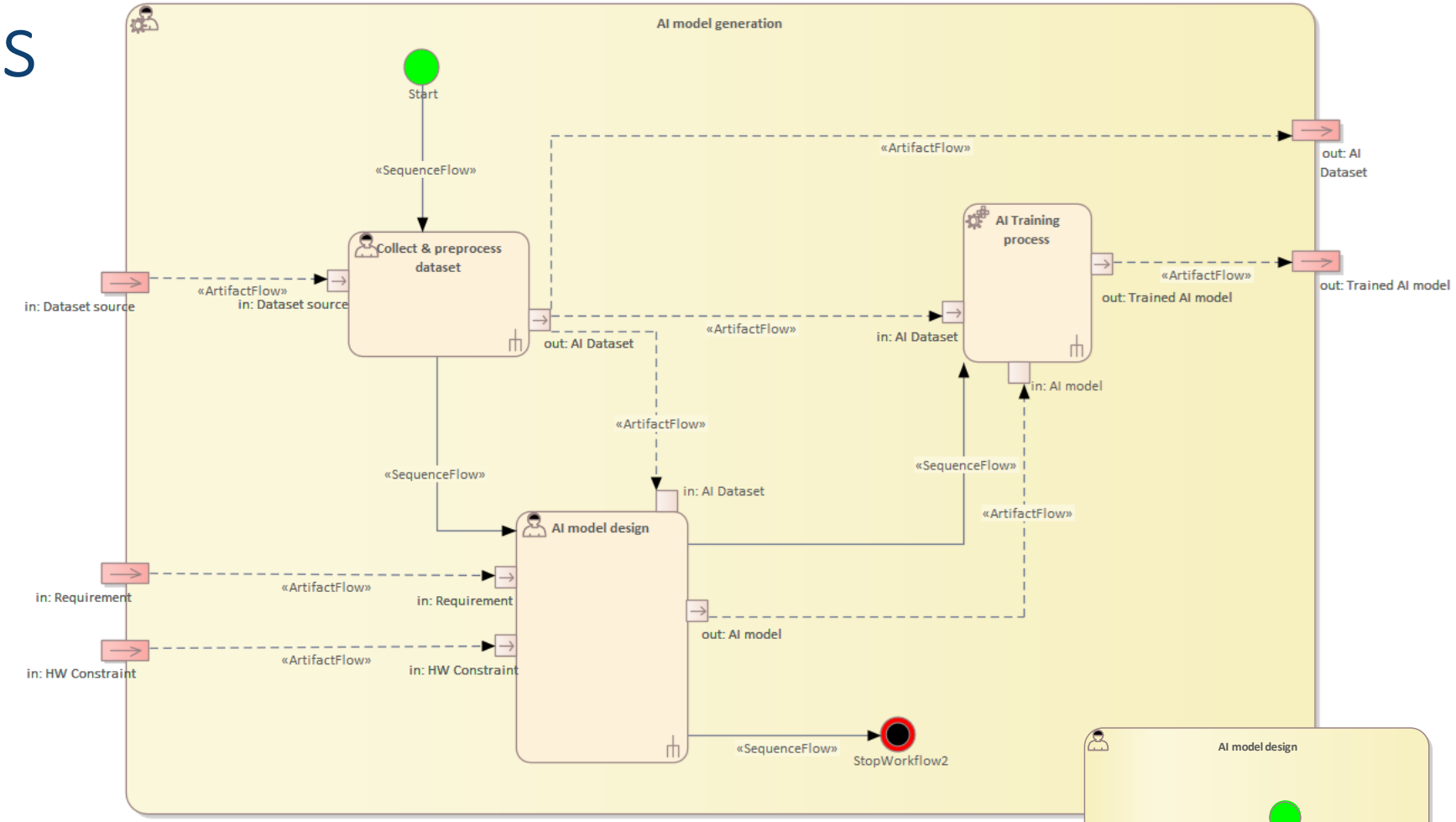
Identify the “right” modeling languages

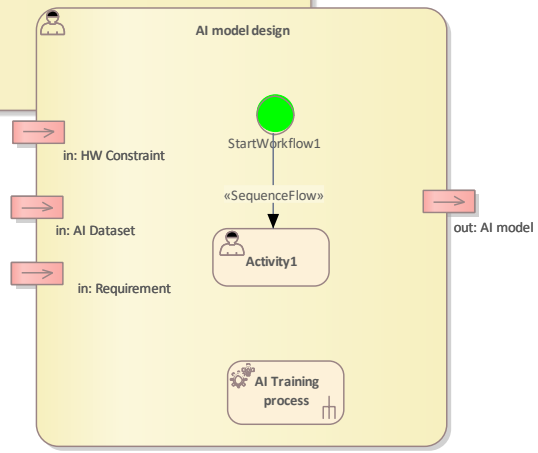


VVML Principles



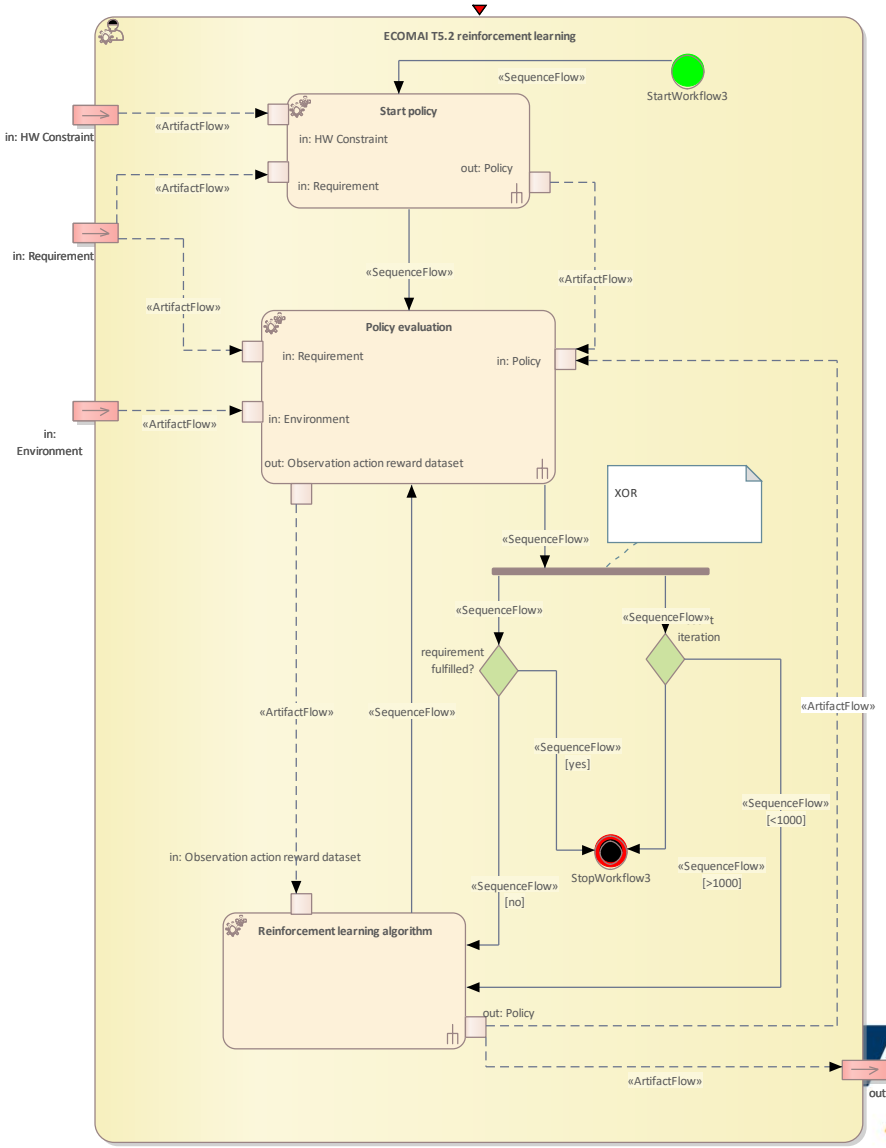


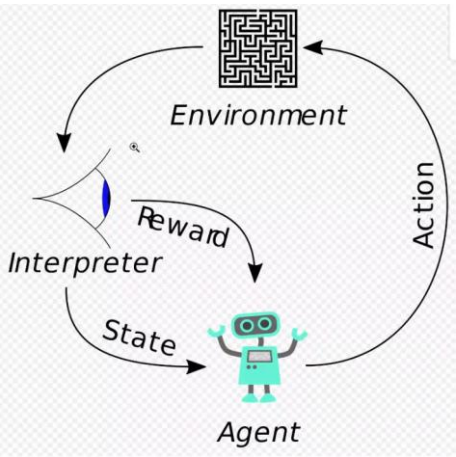


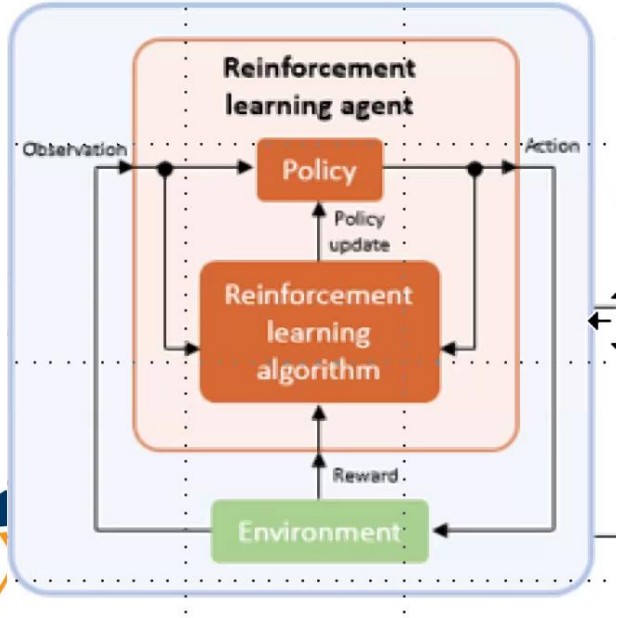


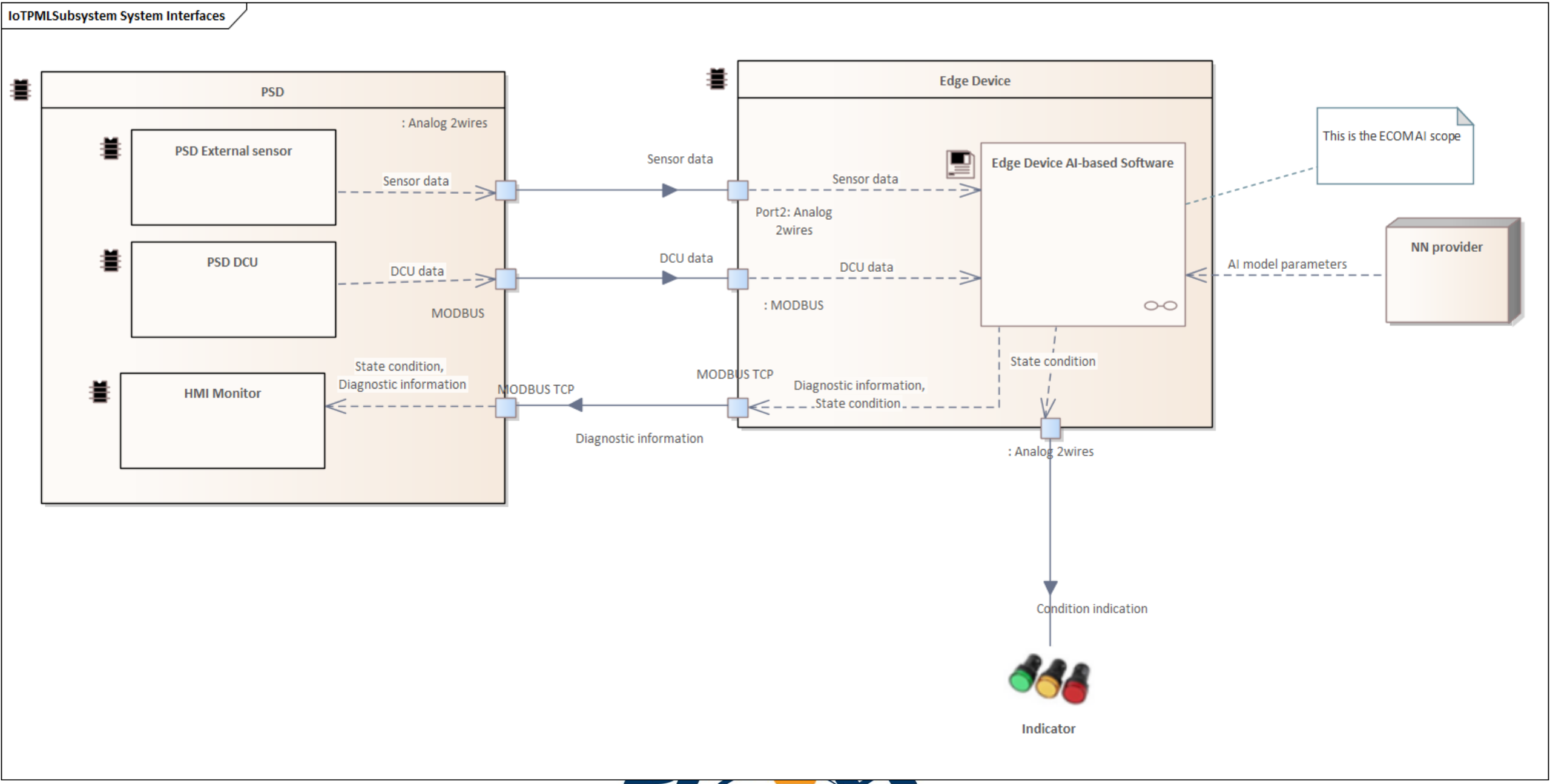
SPARX SERVICES CENTRAL EUROPE

Reinforcement Learning

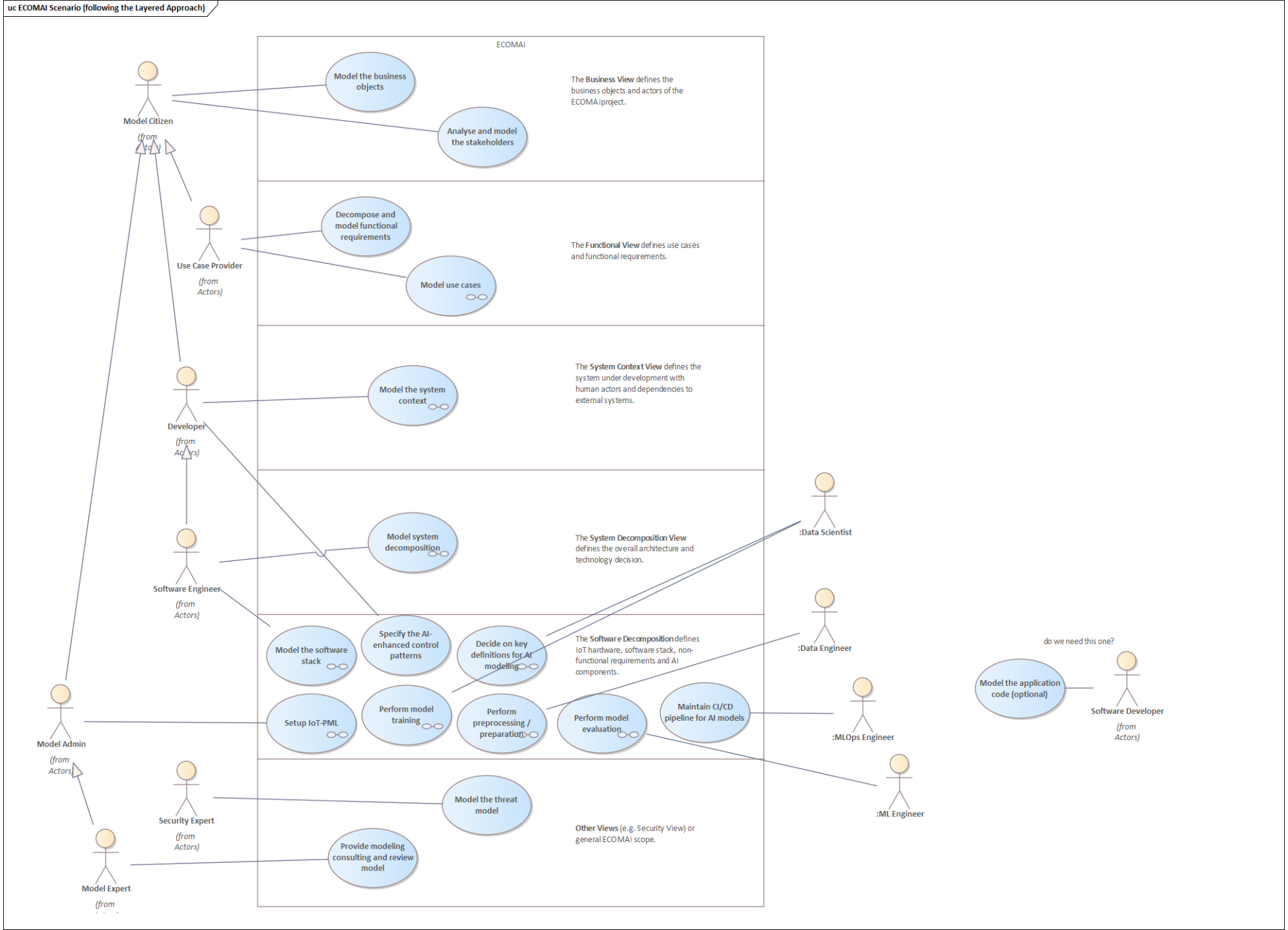


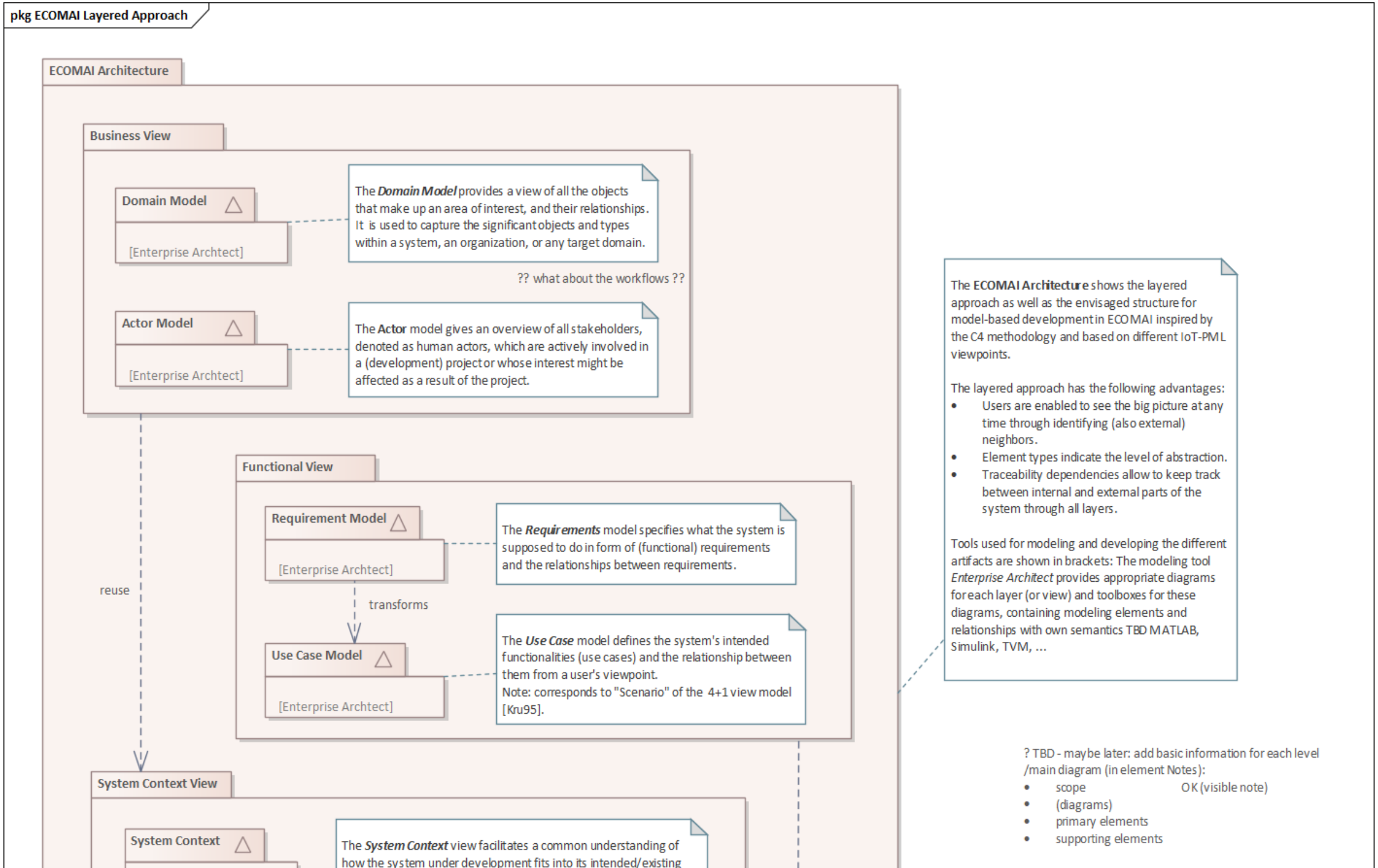





IoTPMLSubsystem System Interfaces
PSD
: Analog 2wires
PSD External sensor
Sensor data
PSD DCU
DCU data
MODBUS
HMI Monitor
State condition,
Diagnostic information
MODBUS TCP
Sensor data
DCU data
MODBUS TCP
Diagnostic information
Edge Device
Sensor data
Port2: Analog
2wires
DCU data
: MODBUS
Edge Device AI-based Software
State condition
Diagnostic information,
State condition
: Analog 2wires
This is the ECOM AI scope
NN provider
AI model parameters
Condition indication
Indicator

DSL Description


uc ECOMAI Scenario (following the Layered Approach)
ECOMAI
Model Citizen
(from Actors)
Use Case Provider
(from Actors)
Developer
(from Actors)
Software Engineer
(from Actors)
Model Admin
(from Actors)
Security Expert
(from Actors)
Model Expert
(from
Model the business objects
Analyse and model the stakeholders
The Business View defines the business objects and actors of the ECOMAI project.
Decompose and model functional requirements
Model use cases
The Functional View defines use cases and functional requirements.
Model the system context
The System Context View defines the system under development with human actors and dependencies to external systems.
Model system decomposition
The System Decomposition View defines the overall architecture and technology decision.
Model the software stack
Specify the AI-enhanced control patterns
Decide on key definitions for AI modeling
The Software Decomposition defines IoT hardware, software stack, non-functional requirements and AI components.
Setup IoT-PML
Perform model training
Perform preprocessing / preparation
Perform model evaluation
Maintain CI/CD pipeline for AI models
Model the threat model
Provide modeling consulting and review model
Other Views (e.g. Security View) or general ECOMAI scope.
:Data Scientist
:Data Engineer
:MLOps Engineer
:ML Engineer
do we need this one?
Model the application code (optional)
Software Developer
(from Actors)


pkg ECOMAI Layered Approach
ECOMAI Architecture
Business View
Domain Model
[Enterprise Archtect]
The Domain Model provides a view of all the objects that make up an area of interest, and their relationships. It is used to capture the significant objects and types within a system, an organization, or any target domain.
?? what about the workflows ??
Actor Model
[Enterprise Archtect]
The Actor model gives an overview of all stakeholders, denoted as human actors, which are actively involved in a (development) project or whose interest might be affected as a result of the project.
reuse
Functional View
Requirement Model
[Enterprise Archtect]
The Requirements model specifies what the system is supposed to do in form of (functional) requirements and the relationships between requirements.
transforms
Use Case Model
[Enterprise Archtect]
The Use Case model defines the system's intended functionalities (use cases) and the relationship between them from a user's viewpoint.
Note: corresponds to "Scenario" of the 4+1 view model [Kru95].
System Context View
System Context
The System Context view facilitates a common understanding of how the system under development fits into its intended/existing
The ECOMAI Architecture shows the layered approach as well as the envisaged structure for model-based development in ECOMAI inspired by the C4 methodology and based on different IoT-PML viewpoints.
The layered approach has the following advantages:
• Users are enabled to see the big picture at any time through identifying (also external) neighbors.
• Element types indicate the level of abstraction.
• Traceability dependencies allow to keep track between internal and external parts of the system through all layers.
Tools used for modeling and developing the different artifacts are shown in brackets: The modeling tool Enterprise Architect provides appropriate diagrams for each layer (or view) and toolboxes for these diagrams, containing modeling elements and relationships with own semantics TBD MATLAB, Simulink, TVM, ...
? TBD - maybe later: add basic information for each level /main diagram (in element Notes):
• scope OK (visible note)
• (diagrams)
• primary elements
• supporting elements

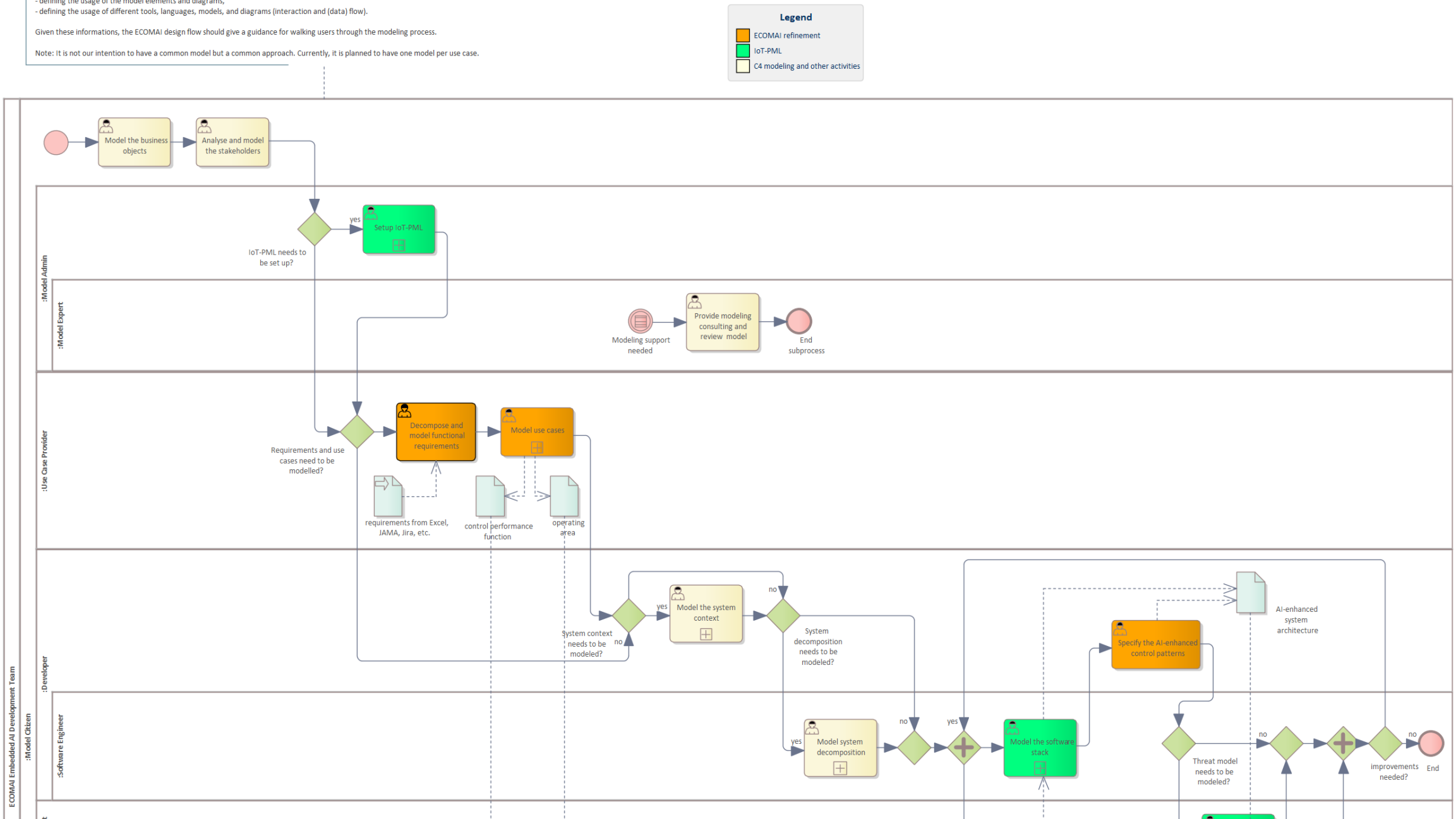
Business Process AI-PML

Describes the top-down ECOMAI design flow on how to build models using the "AI-PML" (methodology).

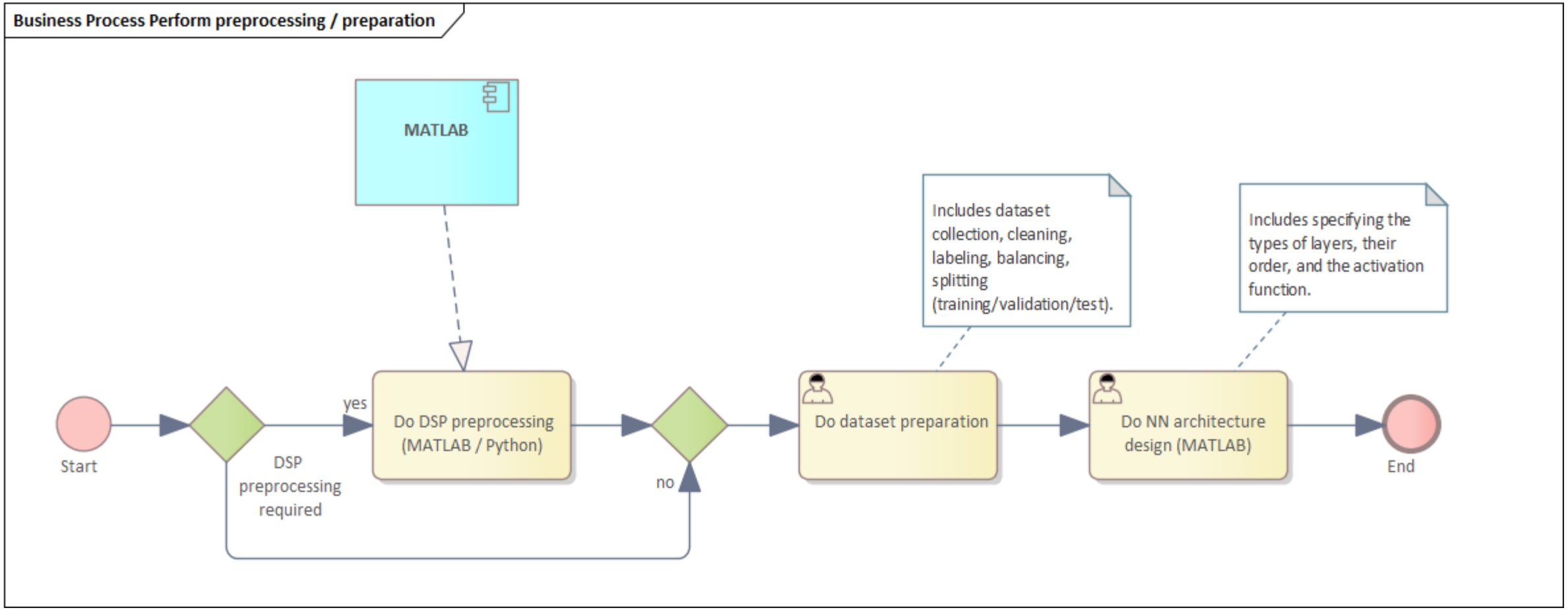
The goal of this sub-task is to document the ECOMAI development process, which includes
- defining the steps and responsibilities (user roles) for model-based development in the ECOMAI Design Kit,
- defining the usage of the model elements and diagrams,
- defining the usage of different tools, languages, models, and diagrams (interaction and (data) flow).

Given these informations, the ECOMAI design flow should give a guidance for walking users through the modeling process.

Note: It is not our intention to have a common model but a common approach. Currently, it is planned to have one model per use case.



ArchiMate + BPMN



SPARX SERVICES CENTRAL EUROPE

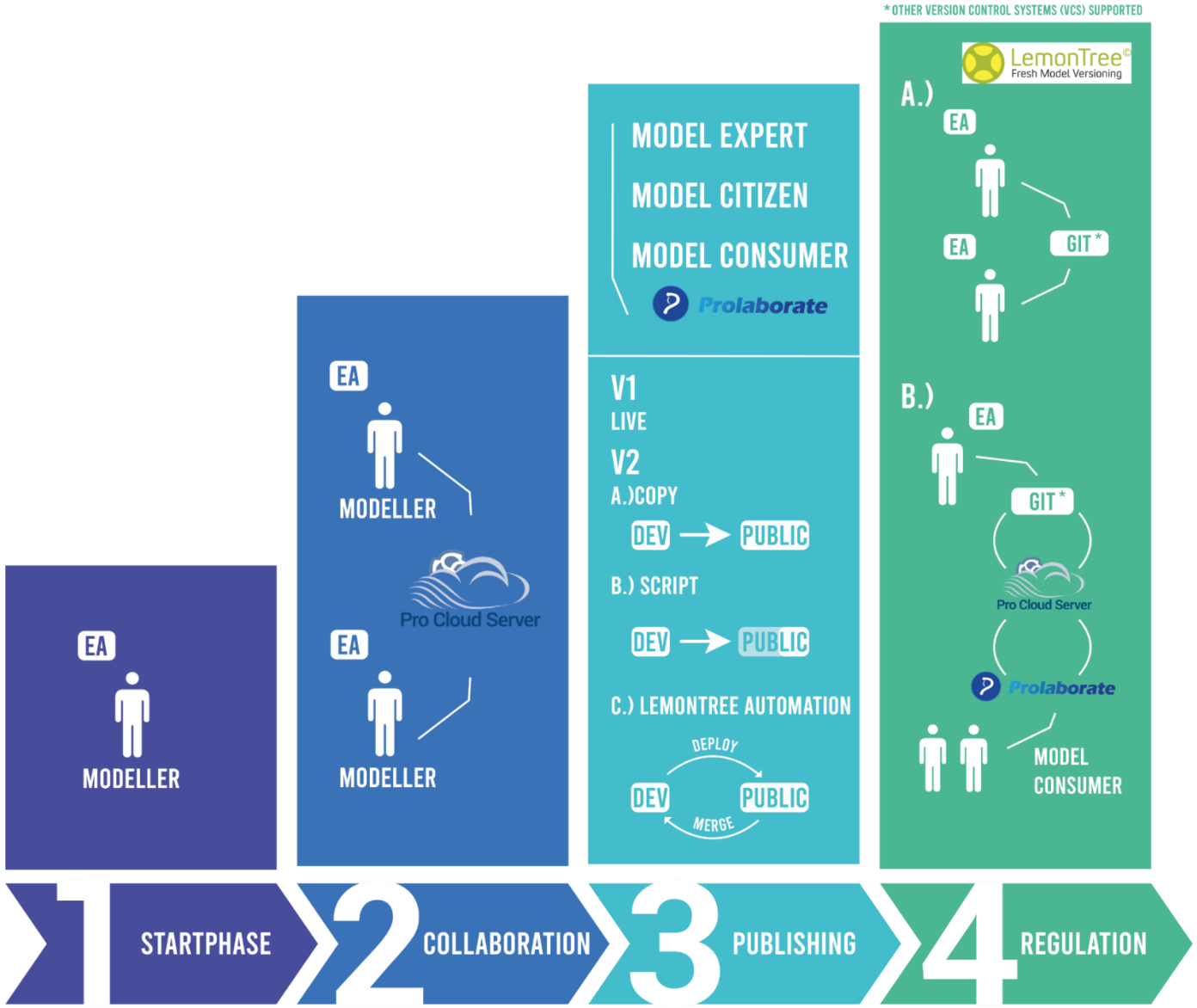
General Takeaway

IT Should Not Use BPMN To Capture Business Processes

- IT should not use BPMN to capture business processes as this will create chaos and no clear demarcation between business and the application layer.
- To avoid misunderstandings and political hick-hack, IT should use Archimate for application processes and UML Activity Diagrams for the creation of Application Process Flow Diagrams.
- BPMN should be left to Business Architects and the Business. In the BPMN Business Process flow diagrams, specify Business Services in the Pool and Business Roles and Business Collaborations in the Lanes.

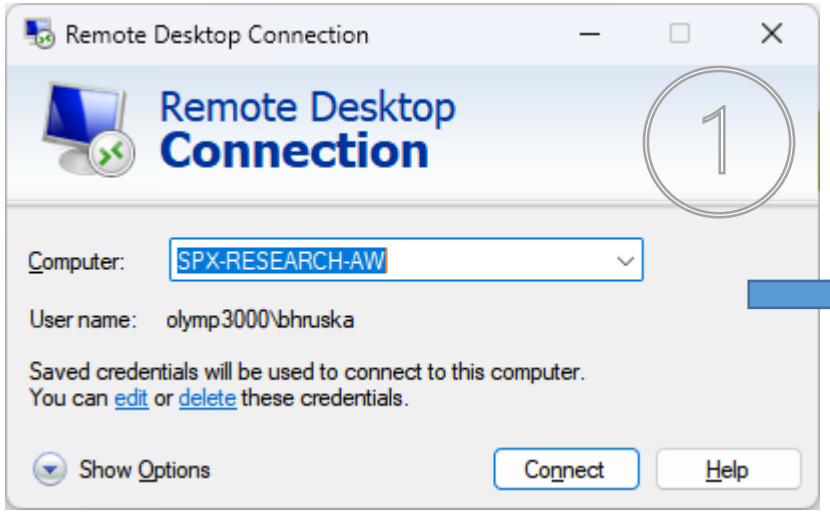
Representation of Application Components in BPMN and UML Activity Diagrams

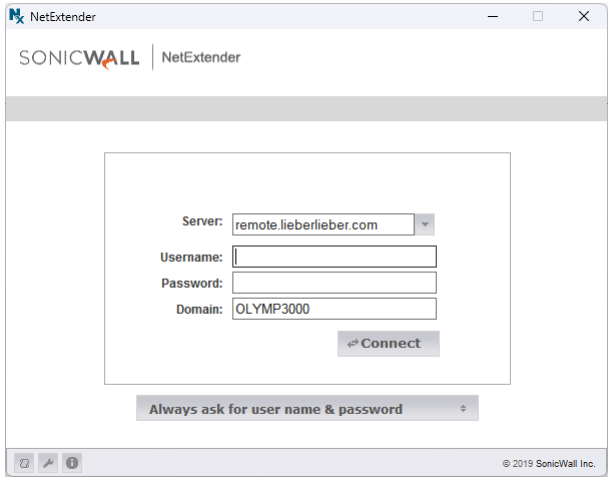
- In BPMN diagrams, Application Components such as User Interfaces may be represented as a "Supporting" architecture element.
- In UML Activity diagrams, generic business actors and application components from ArchiMate can be represented in the partitions or used as classifiers.
- Business Roles, Business Services, Business Processes, and Business Collaborations should not be used in order to maintain a clear separation.

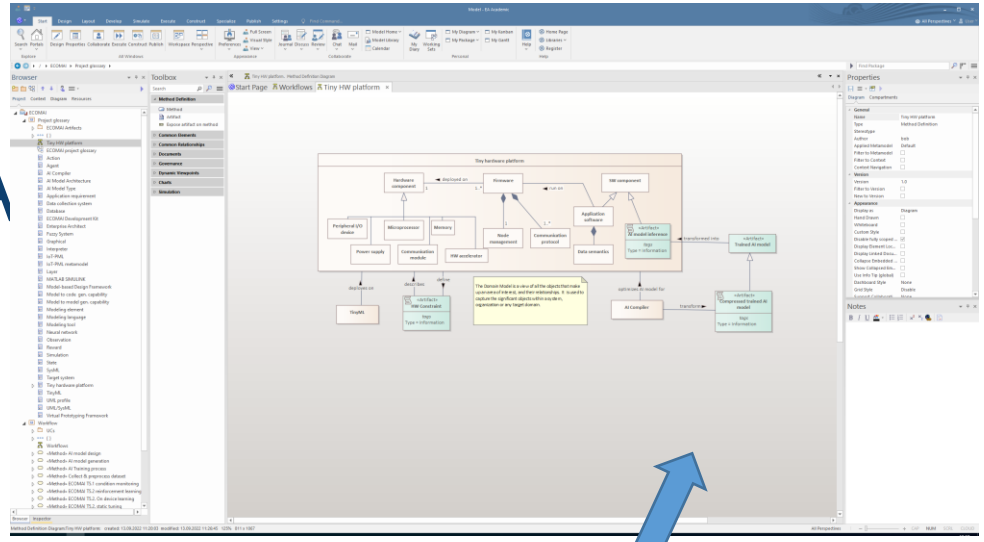


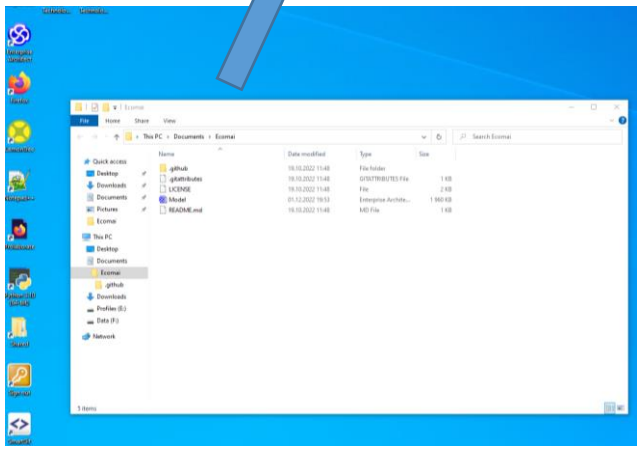
SPARX SERVICES CENTRAL EUROPE

Deployment of EA for ECOMA





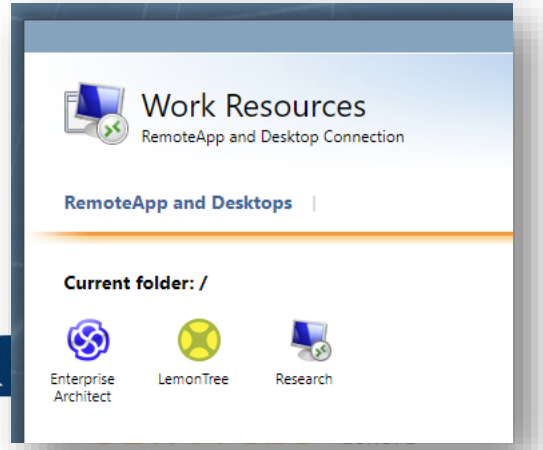




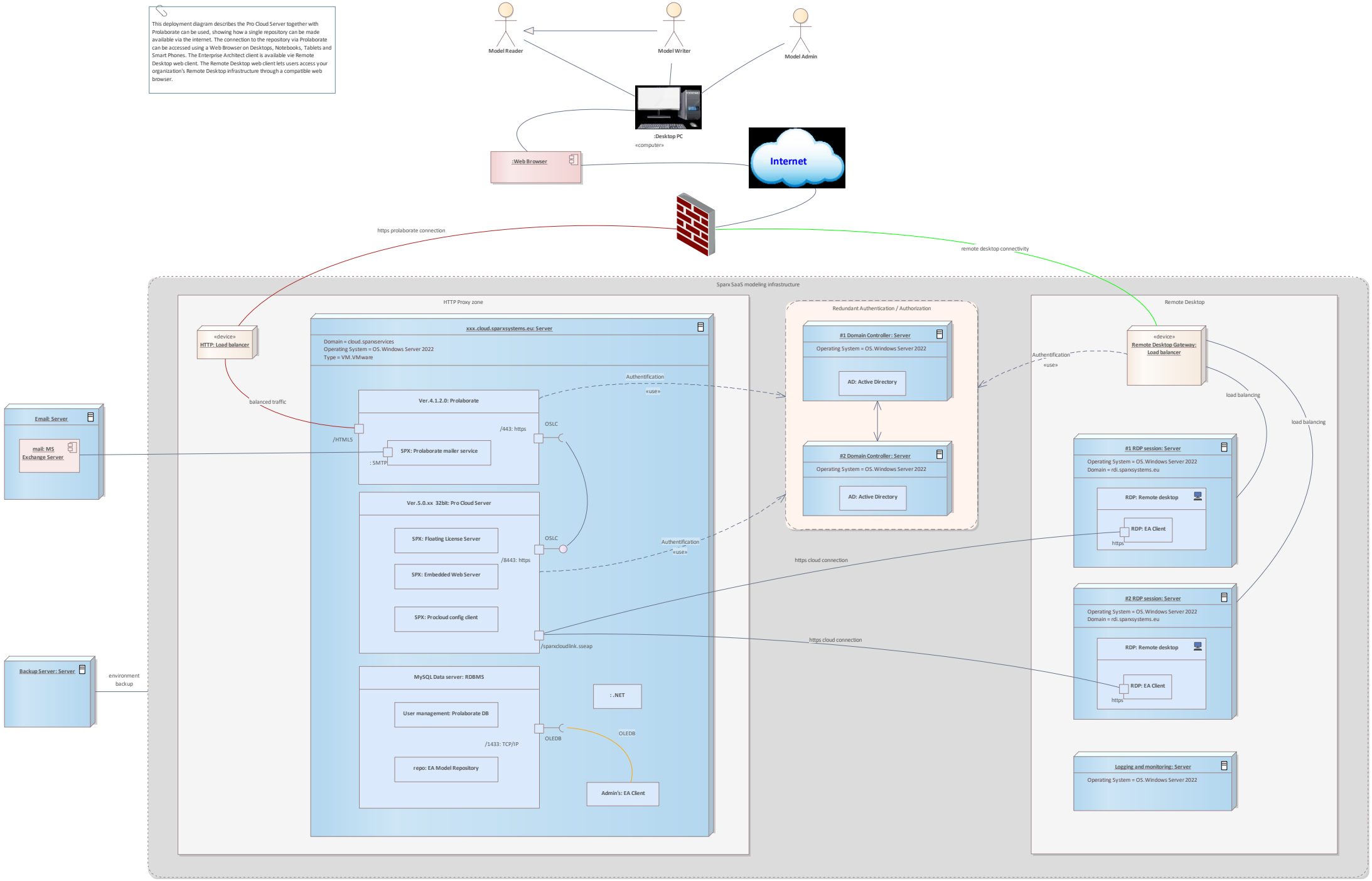
https://research.sparxsystems.eu/rdweb

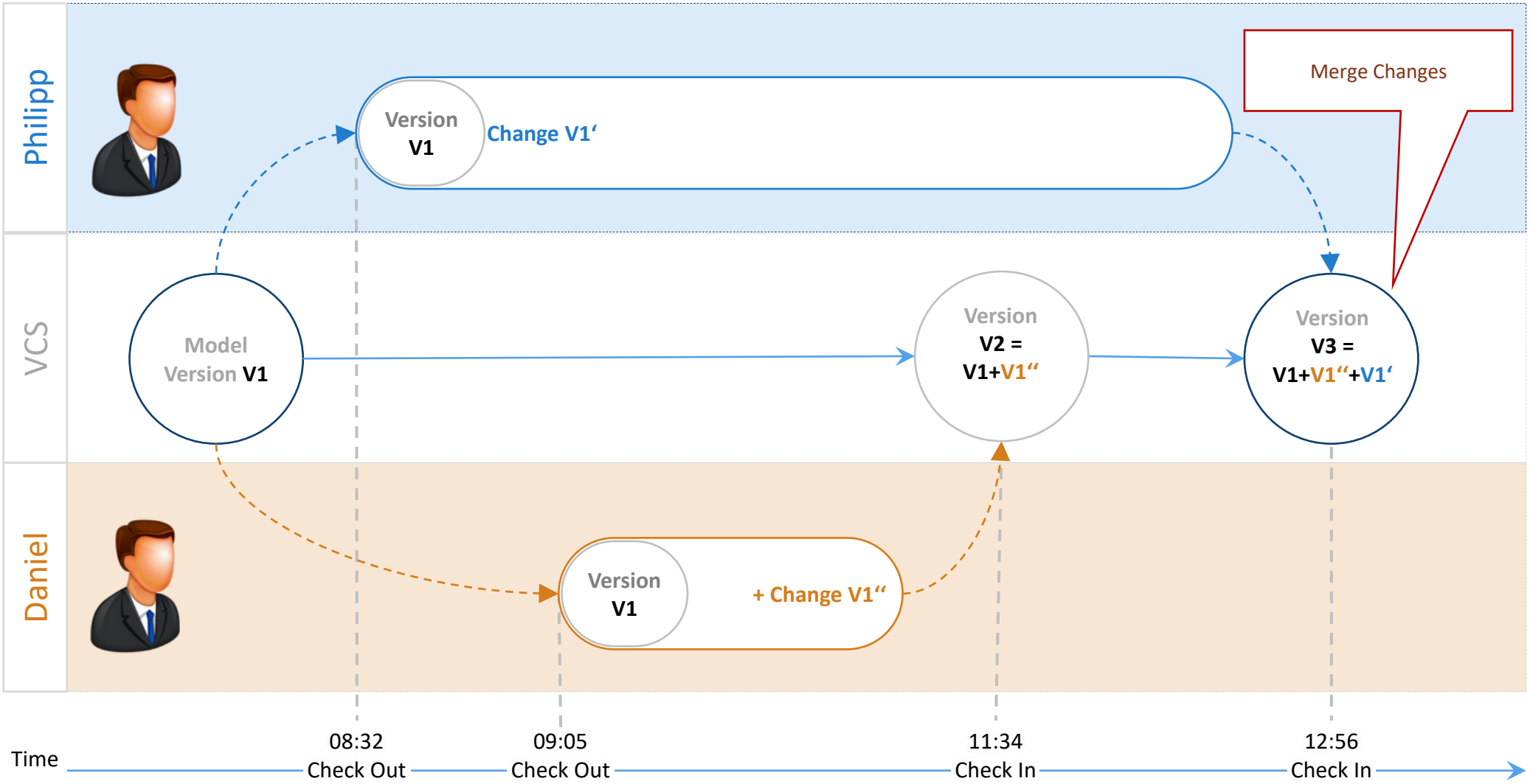
https://research.sparxsystems.eu/RDWeb/webclient

2



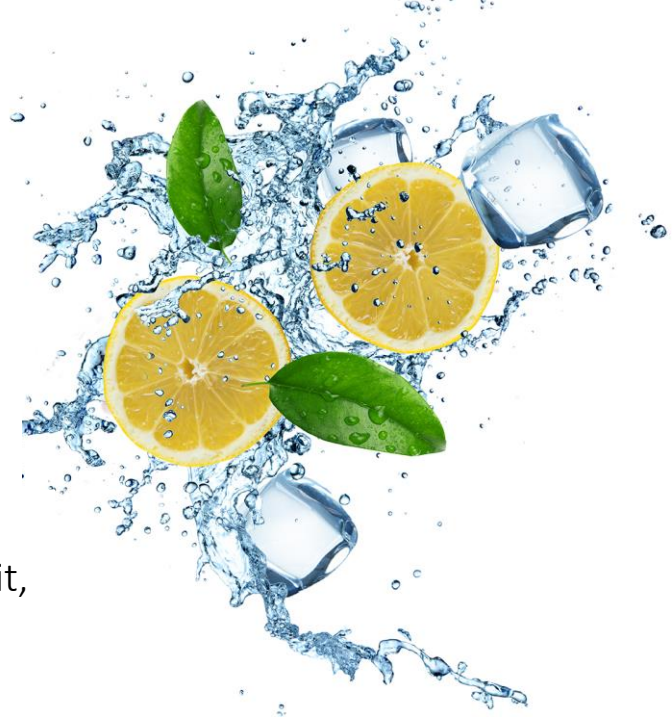
This deployment diagram describes the Pro Cloud Server together with Prolaborate can be used, showing how a single repository can be made available via the internet. The connection to the repository via Prolaborate can be accessed using a Web Browser on Desktops, Notebooks, Tablets and Smart Phones. The Enterprise Architect client is available vie Remote Desktop web client. The Remote Desktop web client lets users access your organization's Remote Desktop infrastructure through a compatible web browser.





SPARX SERVICES CENTRAL EUROPE

LemonTree © Highlights



Diff & Merge
3-way diffing and merging of Enterprise Architect models

Model Versioning
Parallel editing of models through optimistic model versioning

VCS Integration
Seamless integration with Git, Subversion, PTC, etc.

Branches of models
Parallel developments of versions and variants

Merge Preview
Diagram merge and merge preview

Review
Changes are visualized clearly and understandably for reviews

LemonTree©
Fresh Model Versioning
lemontree.lieberlieber.com

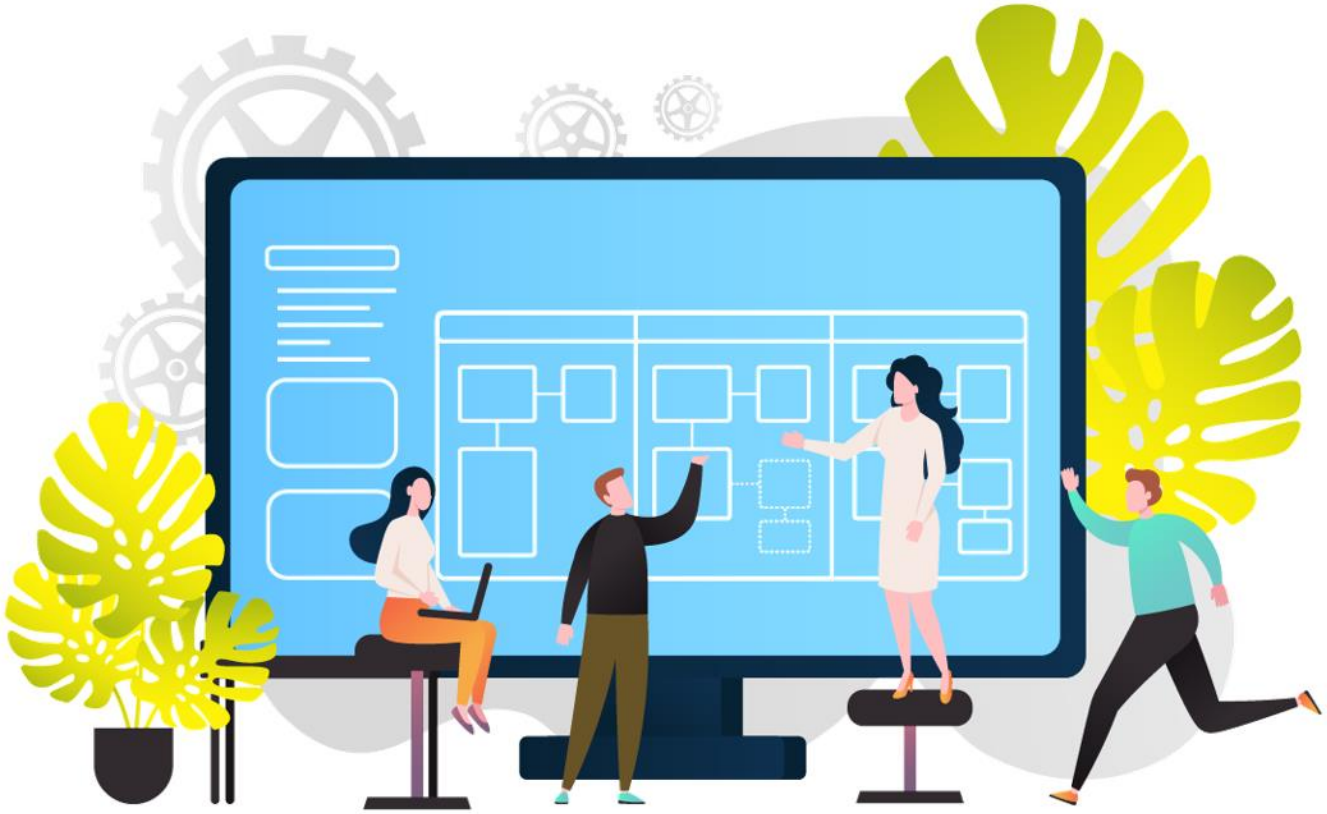
SPARX SERVICES CENTRAL EUROPE

1. Filtering
2. Diagram Versions
3. List of Impacted Diagrams
4. Changed Element Properties
5. Smart Grouping of Changes
6. Merge Preview

LemonTree©

Fresh Model Versioning

lemontree.lieberlieber.com



by LieberLieber