Introduction on UML for Industrial Systems

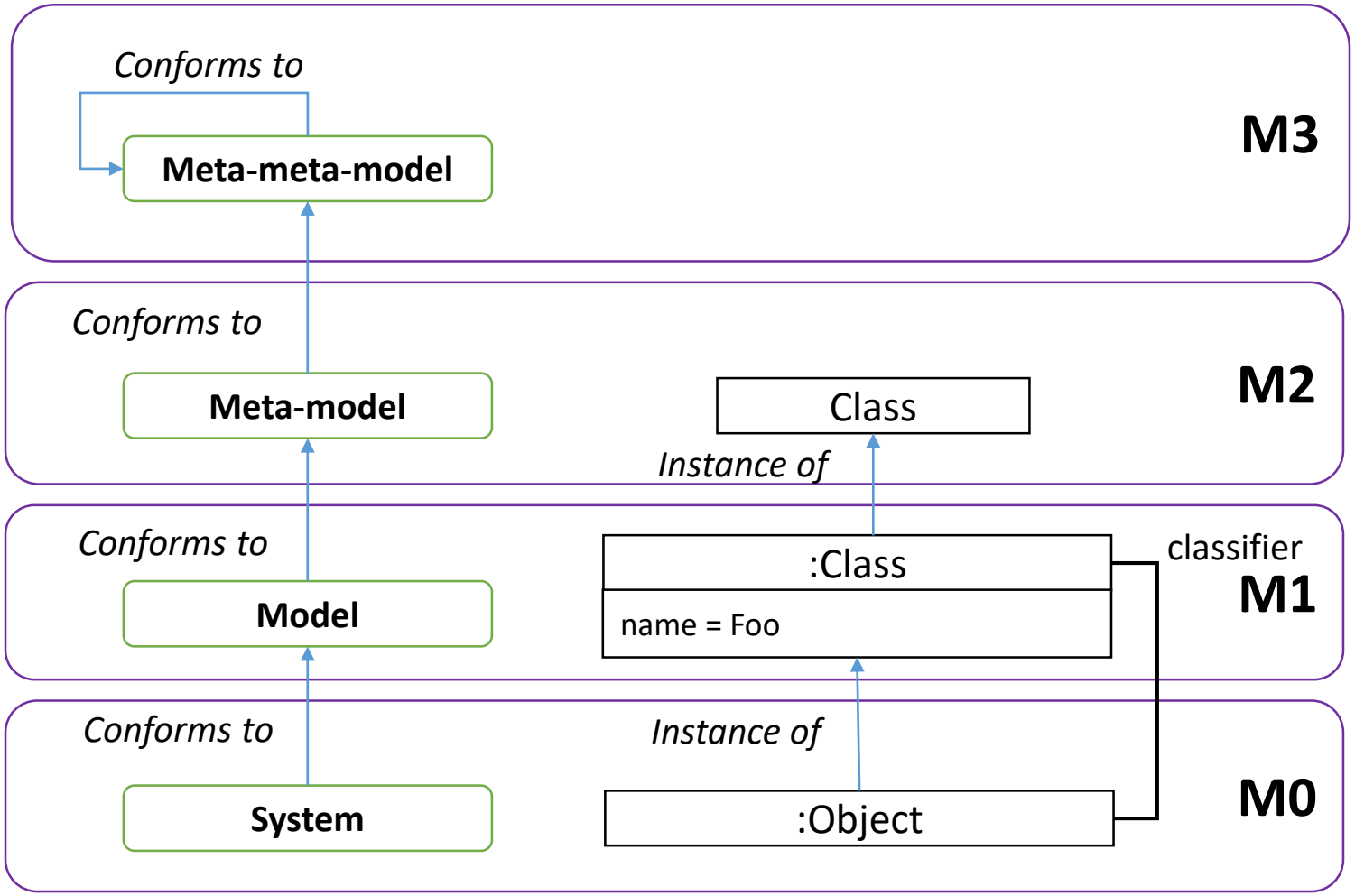
Ansgar Radermacher / Asma Smaoui
ansgar.radermacher@cea.fr

Acknowledgments – contains material from my CEA colleagues Shuai Li, Jérémie Tatibouët, François Terrier, Sébastien Gérard

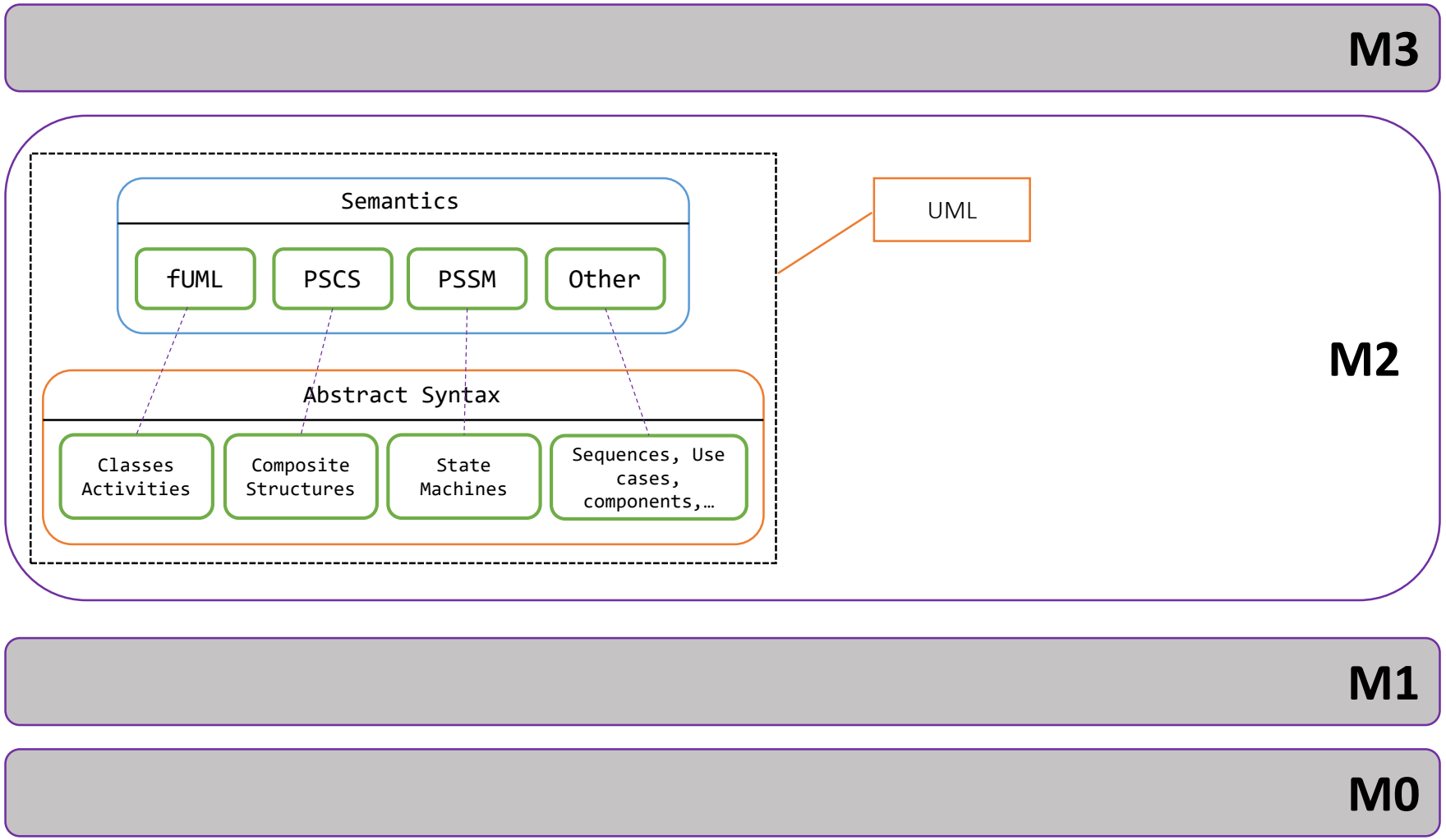
Adapt UML to a specific domain

1. **UML profile mechanisms**
2. **Define a simple profile for requirements**
3. **See how this has been done in SysML v1**

Wrap UP - Model, meta-model and meta-meta-model



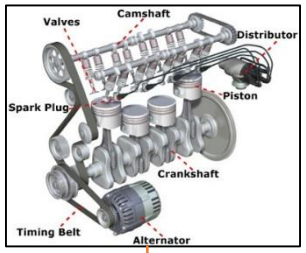
WHERE IS UML ?


M3
Semantics
fUML
PSCS
PSSM
Other
UML
Abstract Syntax
Classes Activities
Composite Structures
State Machines
Sequences, Use cases, components,…
M2
M1
M0

UML – richness & limitations

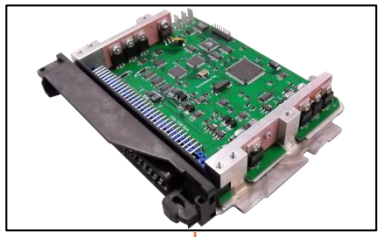
- Formalize structure of a system using different formalisms (statics)
  Formalize behavior of a system using different formalisms (dynamics)
  Capture system abstractions and user interactions
- Intended to be **agnostic** of a particular domain
  - Software-oriented, inspired from object-oriented programming languages
- Will you use it to describe the following kind of systems ?

© http://www.driving-test-success.com



Points of interest: piston diameter, cylinder volume, weight of components, etc.

© http://www.standaloneenginemanagement.com/

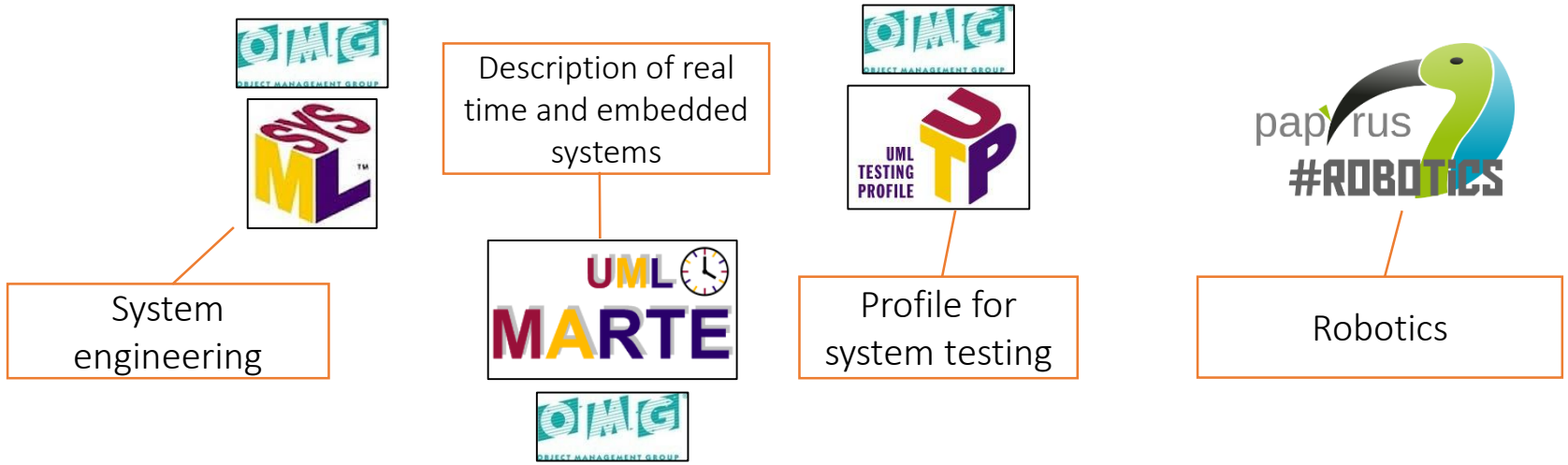


Points of interest: processor rate, the quantity of memory, energy consumption?

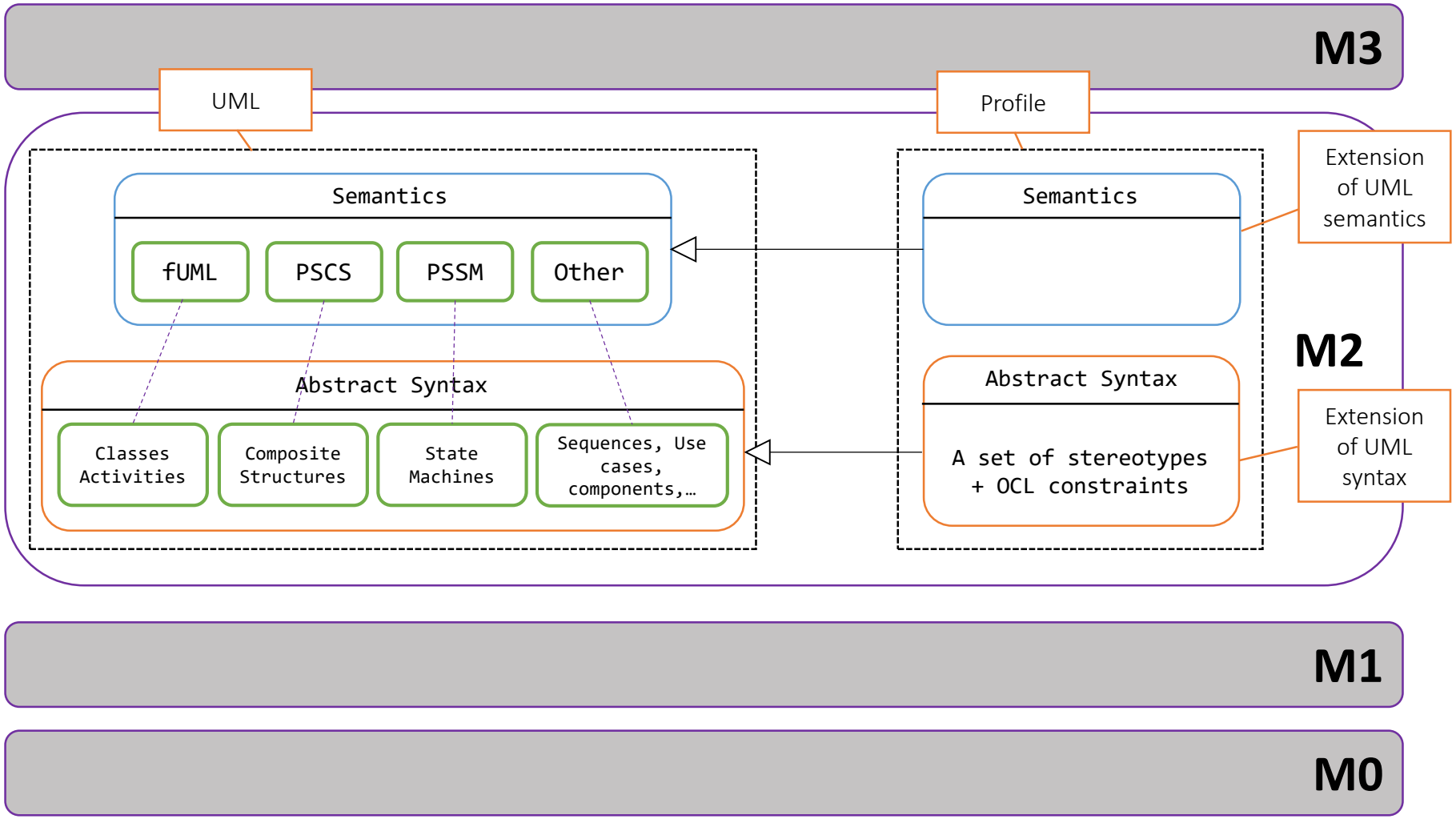
UML extensibility – the profile mechanism

Difficulty to capture concerns of specific domains

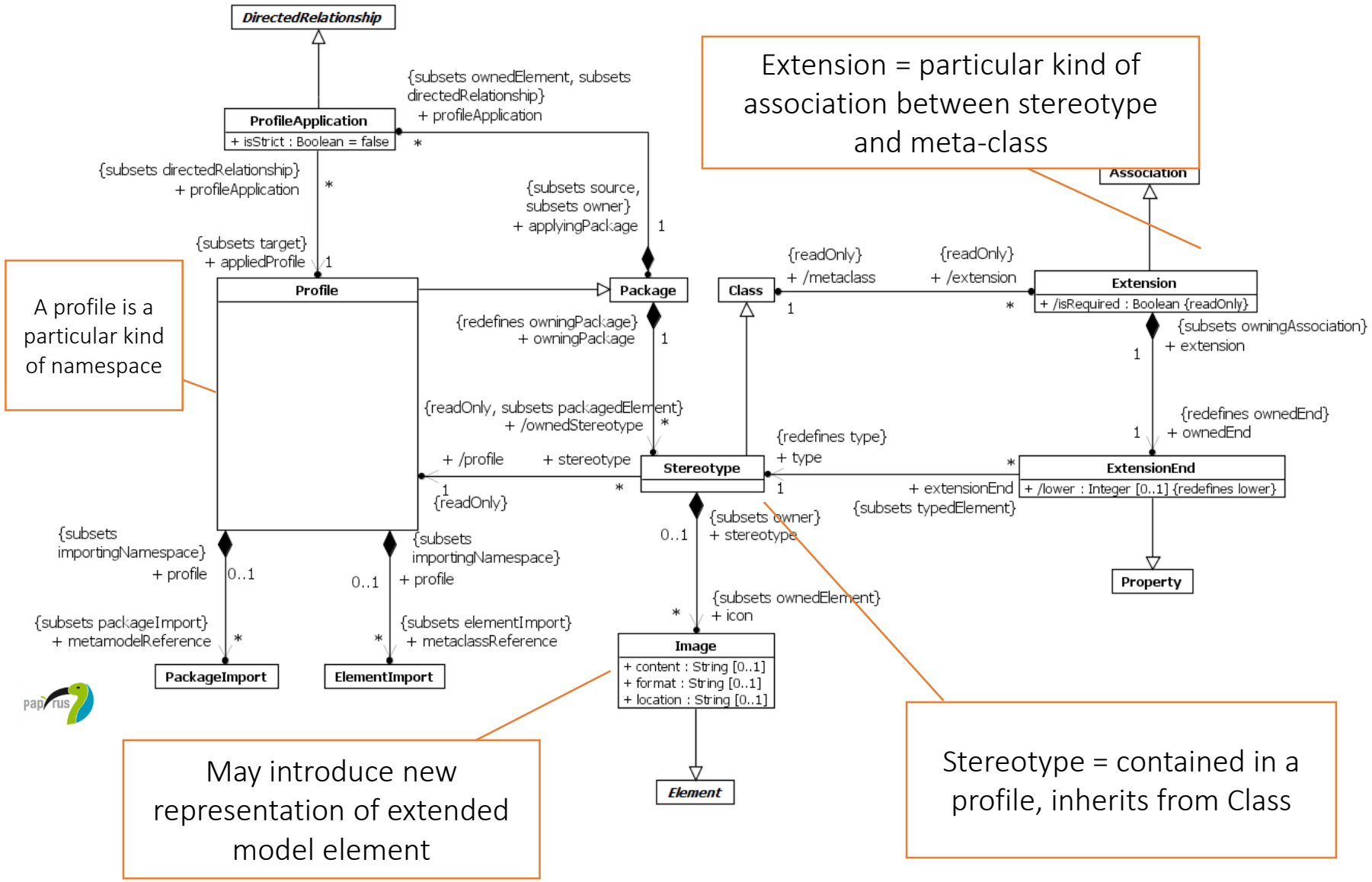
- … such as mechanics, electronics, avionic, etc.
- Extend UML to capture the concerns of these domains
  - Enable design of UML based domain specific languages
  - Capitalize on syntax and semantics provided by UML
- UML profiles are widely used in industry



How does a profile contribute to UML?



Profiles - UML meta-model view



Extension = particular kind of association between stereotype and meta-class

A profile is a particular kind of namespace

May introduce new representation of extended model element

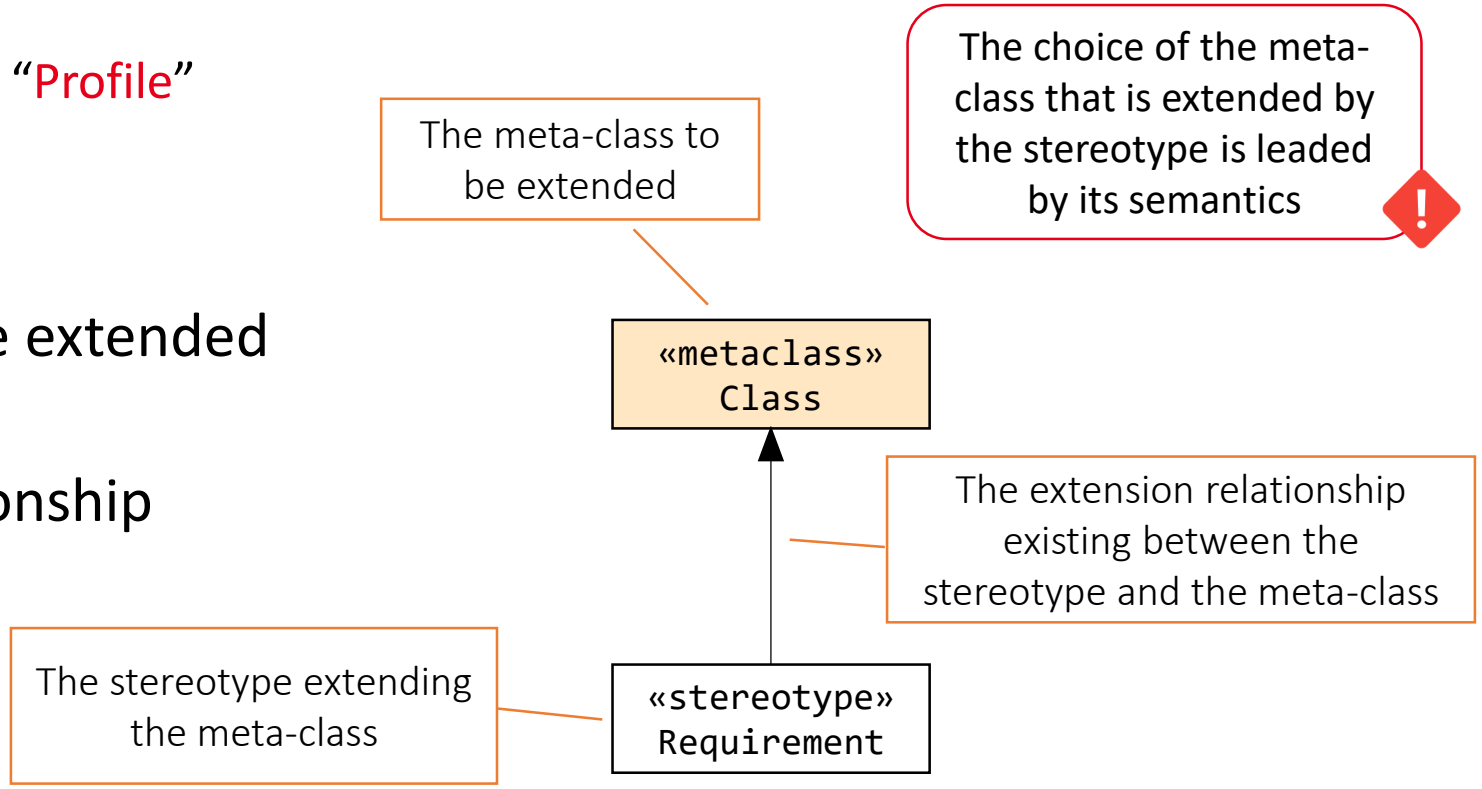
Stereotype = contained in a profile, inherits from Class

Example – a profile for requirement engineering

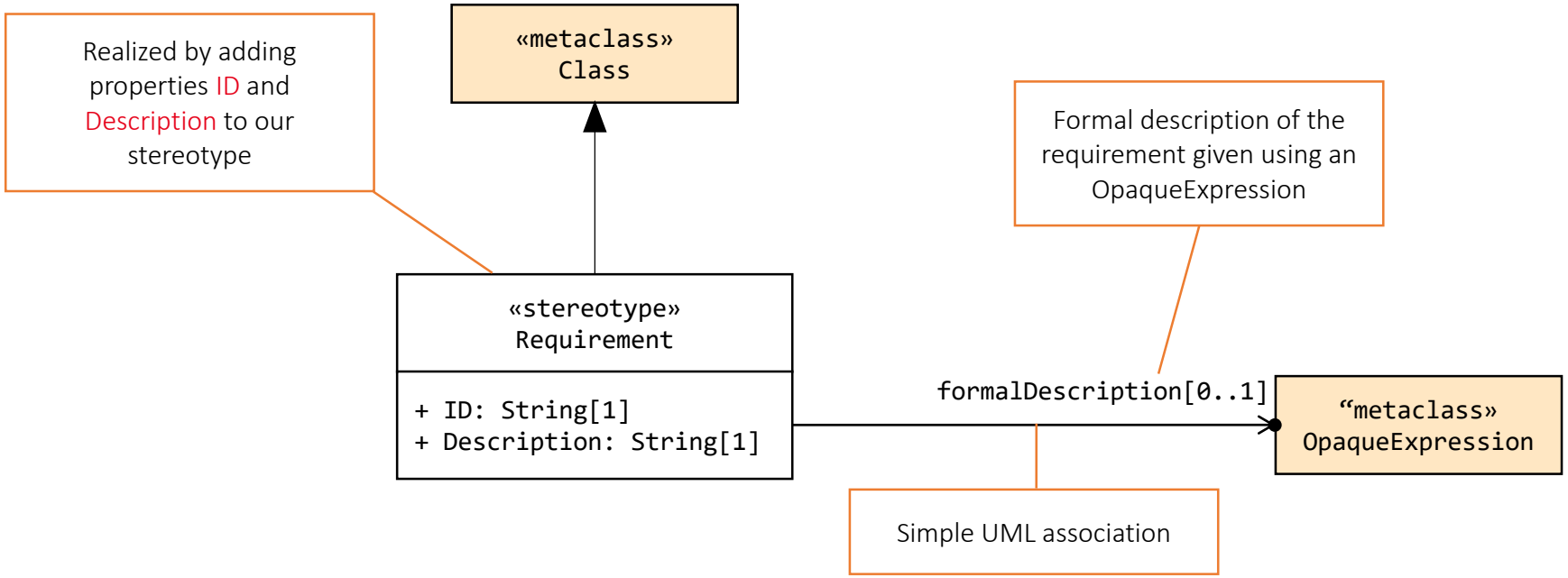
- Objective – Capture system requirements
- Analysis
  - In my system I can have
    - Functional requirements (actions that can be accomplished by the system)
    - Performance requirements (expected performance for the system)
    - Structural requirements (Identify the necessary structure of the system)
  - A base requirements has
    - An identifier
    - A text that describes a requirement
    - A formalization (i.e. an expression given in a formal language)
  - A general requirement can be refined (clarified) by a set of sub-requirements

Step 1 – define a «Requirement» stereotype

- In Papyrus
  - File – New – Papyrus project
    - Set the project name
    - Choose architecture context “Profile”
- Start to define your profile
  - Import the meta-class to be extended
  - Create a stereotype
  - Create the extension relationship



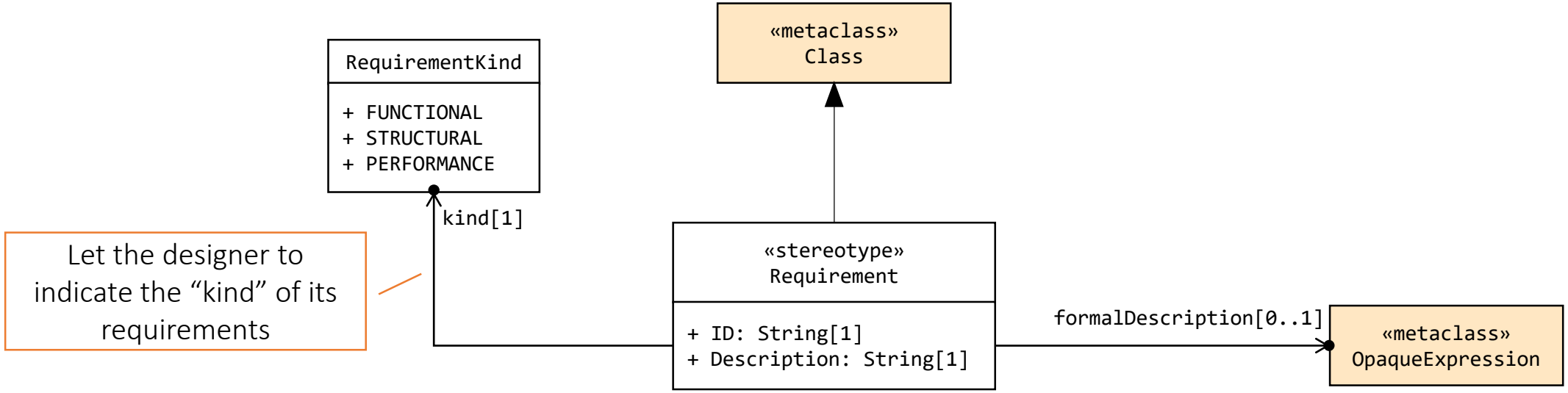
Step 2 – capture basic properties of a requirement



Why did we choose OpaqueExpression?

- Possibility to choose a formal language to define something
- The specification takes the form of a piece of text that is easily accessible

Step 2 – capture basic properties of a requirement



Question

- What is missing in the profile to make a requirement "refinable" by another requirement?

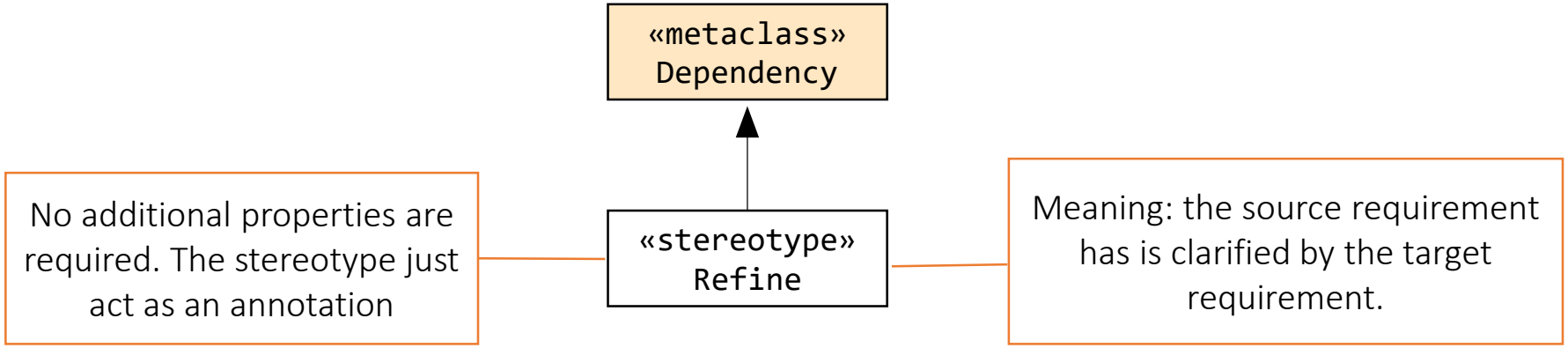
Step 3 – Refinement relationship

Objective

- Find meta-class that allows the specification of a relationship between two classes
- Create a stereotype that apply on this meta-class

Proposal

- Dependency (UML 2.5 – p. 36)
  - Source and target are NamedElements
  - A model element requires another model element for its specification



Reduce UML expressiveness – Why?

- UML meta-models have a lot of features for software modeling
- We do not need (do not want) some features in a specific context

- Requirements use case:
  - Class
    - ~~A class can have operations~~
    - ~~A class can have a classifier behavior~~
    - ~~A class can be active~~
  - Dependency
    - ~~The source and the target can be NamedElement~~

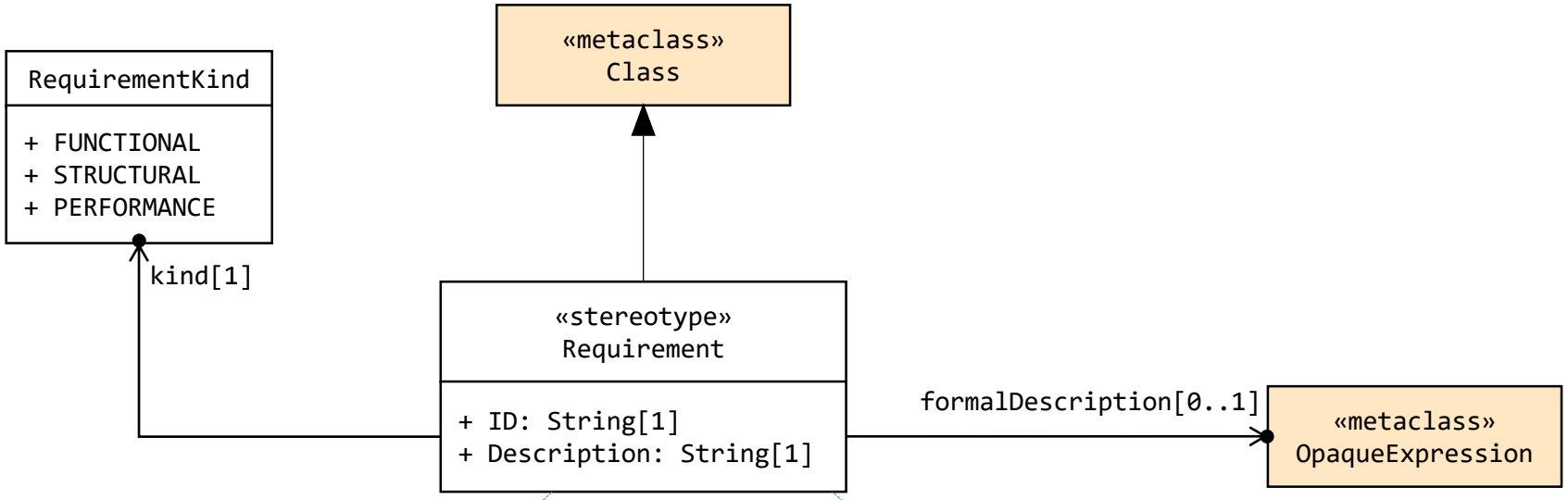
Reduce UML expressiveness – How?

- Use OMG OCL (object constraint language)
  - https://www.omg.org/spec/OCL/
  - OCL is a formal language for constraint specification (not only UML)
  - The constraints that can be verified at any time of the model life

- Associate it with stereotypes

- OCL access to meta-model or stereotype attributes
  - Use "`.`" in expressions for normal attributes, "`->`" for lists
  - List operators, e.g. "`forAll(attr | …)`", "`size()`" or "`sum(…)`"

Reduce expressiveness – «Requirement» stereotype

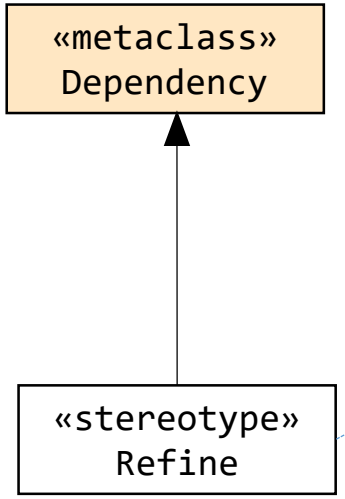



```
self.base_Class.getOwnedOperations()->isEmpty()
                                  and
self.base_Class.classifierBehavior->oclIsUndefined()
```

{}

```
self.formalDescription.language->size()=1
                                  and
self.formalDescription.language->first()="OCL"
```

Reduce expressiveness – «Refine» stereotype



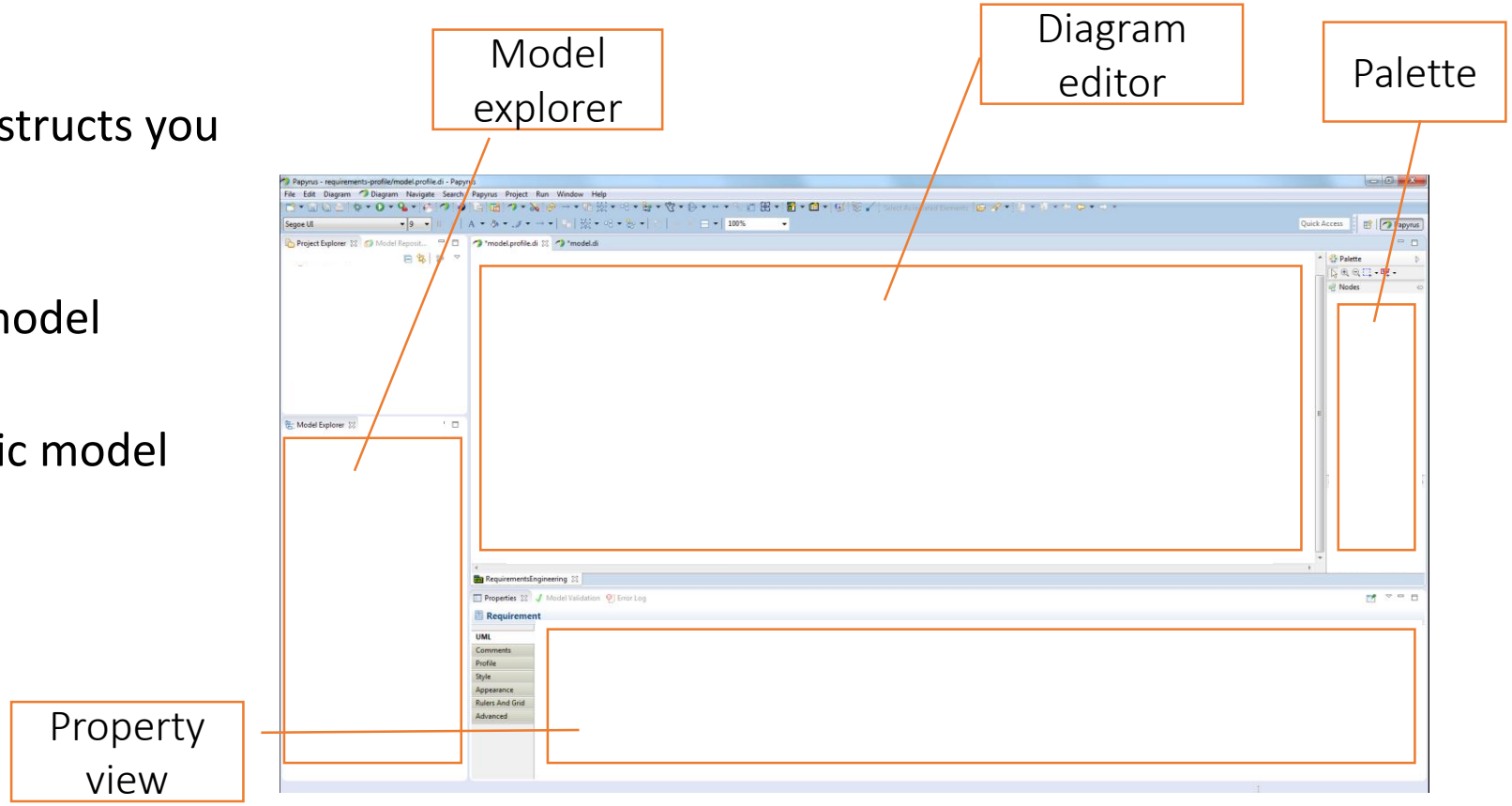

```
self.base_Dependency.supplier->
        getAppliedStereotype("RequirementProfile::Requirement")<>null
                                    and
self.base_Dependency.client->
        getAppliedStereotype("RequirementProfile::Requirement")<>null
```

Guarantee that the source and the target are requirements

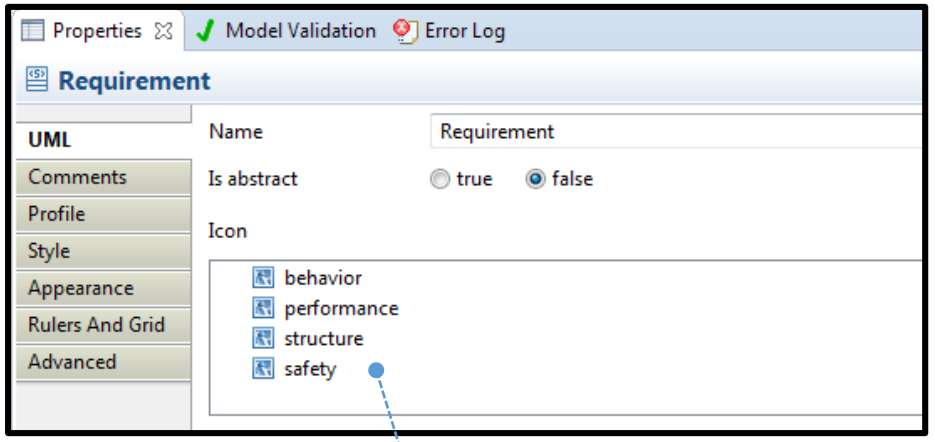
- Help to maintain a model correct by construction

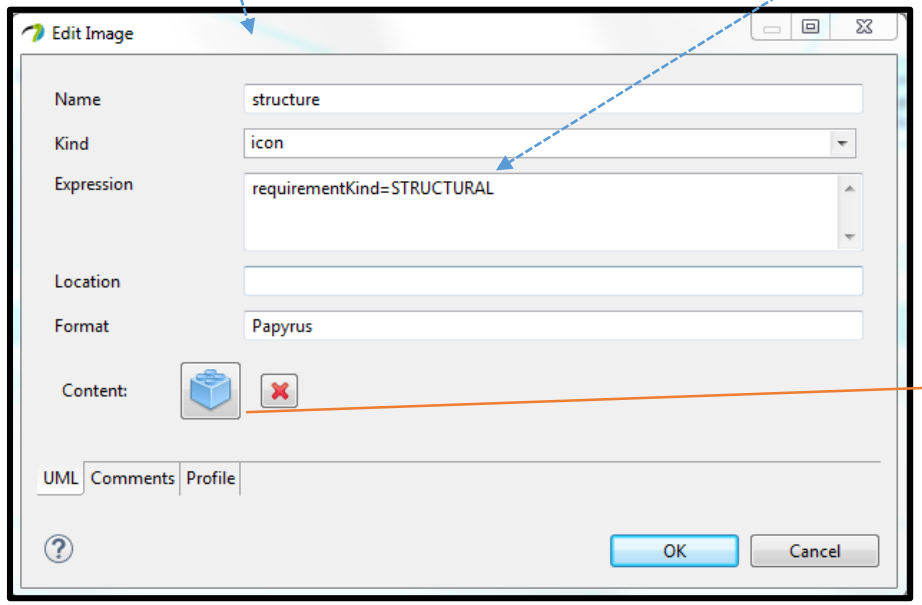
Customization – Papyrus tooling

- **Objective** – a modeler focused on a particular domain … based on a UML profile
- Customizable elements
  - **Palette**
    Provide the modeling constructs you can use in diagrams
  - **Model explorer**
    Current structure of the model
  - **Property view**
    Edit properties of a specific model element
  - **Diagram editor**



Customize stereotyped model elements





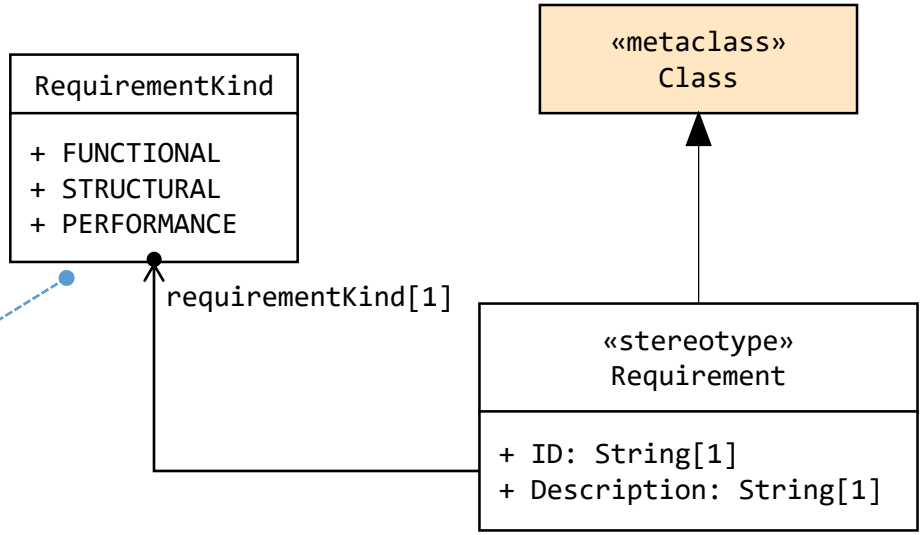


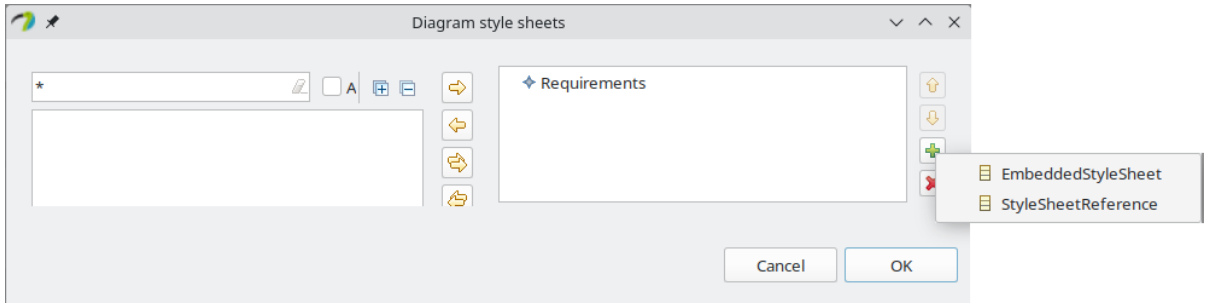
Image associated with representation when
(1) stereotype «Requirement» is applied and
(2) its "requirementKind" property = "STRUCTURAL"

[limitation – bitmap icons, use CSS for SVG shapes]

Customize diagrams via style sheets

Alternative customization via cascading style sheets (CSS)

Style => Diagram Style Sheets, “+” button, select “+” again / Embedded style sheet



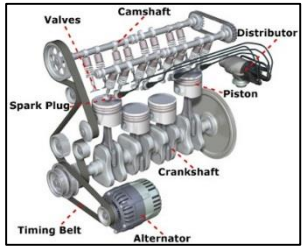

```
Class[appliedStereotypes~="Requirement"] {
   fillColor : yellow;
}
Class[appliedStereotypes~="Requirement"] > Compartment[kind="operations"],
Class[appliedStereotypes~="Requirement"] > Compartment[kind="nestedclassifiers"] {
   visible: false;
}
```

Background color of classes with stereotype Requirement is yellow

Hide compartments "operations" & "nestedclassifiers"

Apply a profile => tailored version of UML

After applying the profile on a model, we can capture requirements

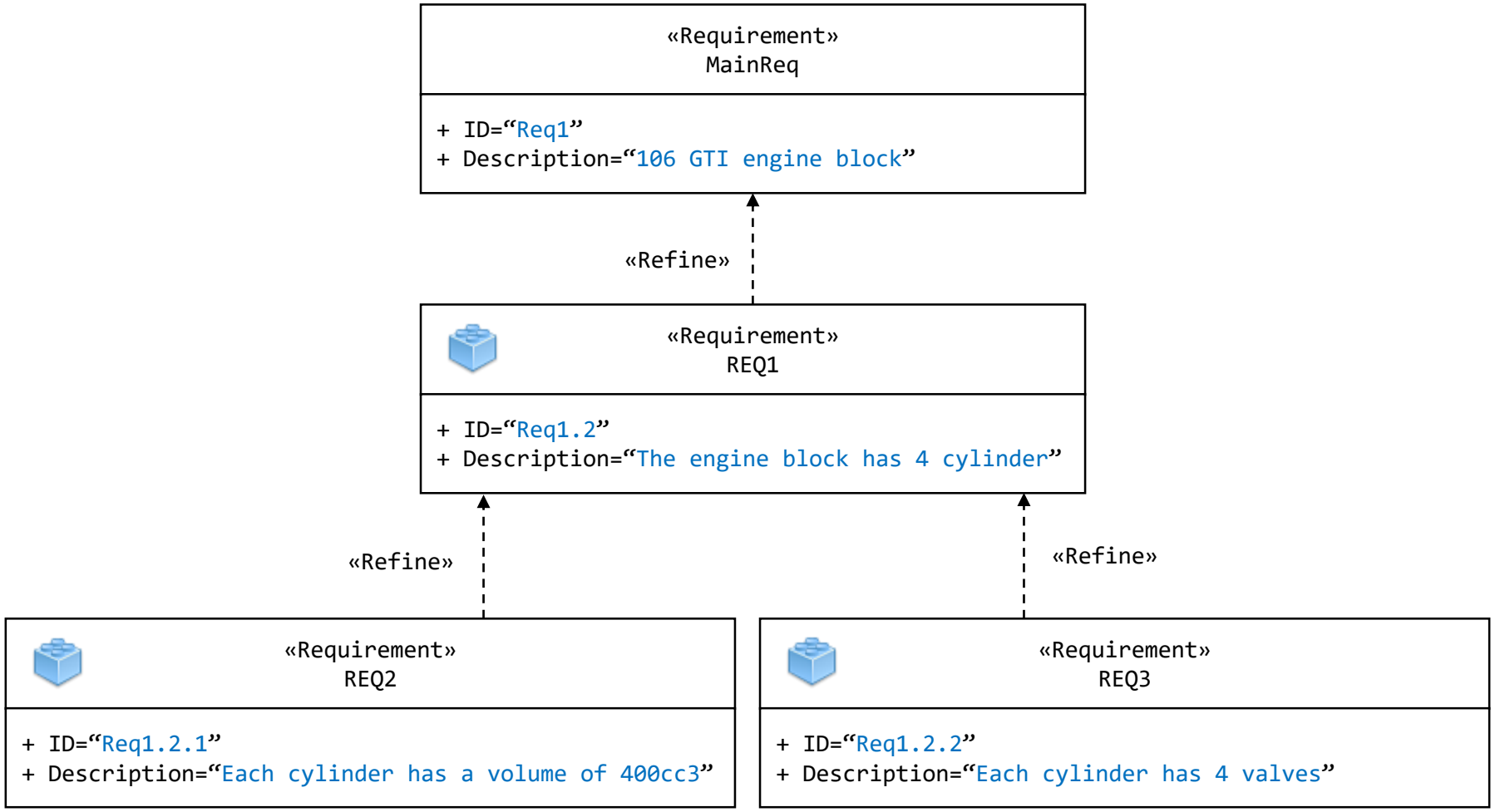


Lets consider the following specification

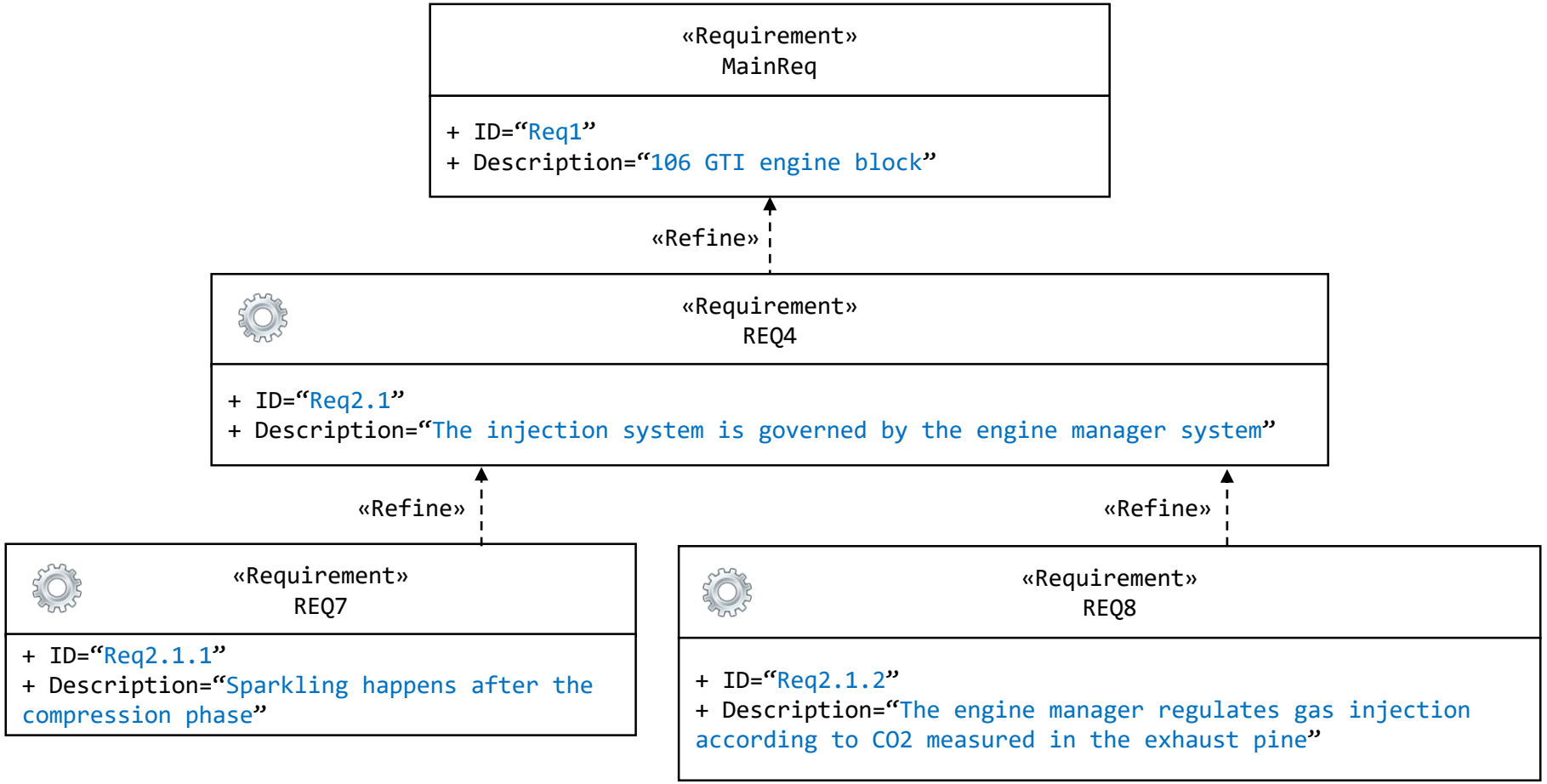
- REQ1 - The engine block of a 106 GTI has four cylinder
- REQ2 - The volume of the engine is 1600cc
- REQ3 - Each cylinder has four valves
- REQ4 - The injection system is governed by a the engine manager
- REQ5 - The engine provides 140ch at 6700 rpm/min
- REQ6 - The engine provides a torque of 150Nm at 5200 rpm/min
- …

Let’s build the requirement model corresponding to the specification

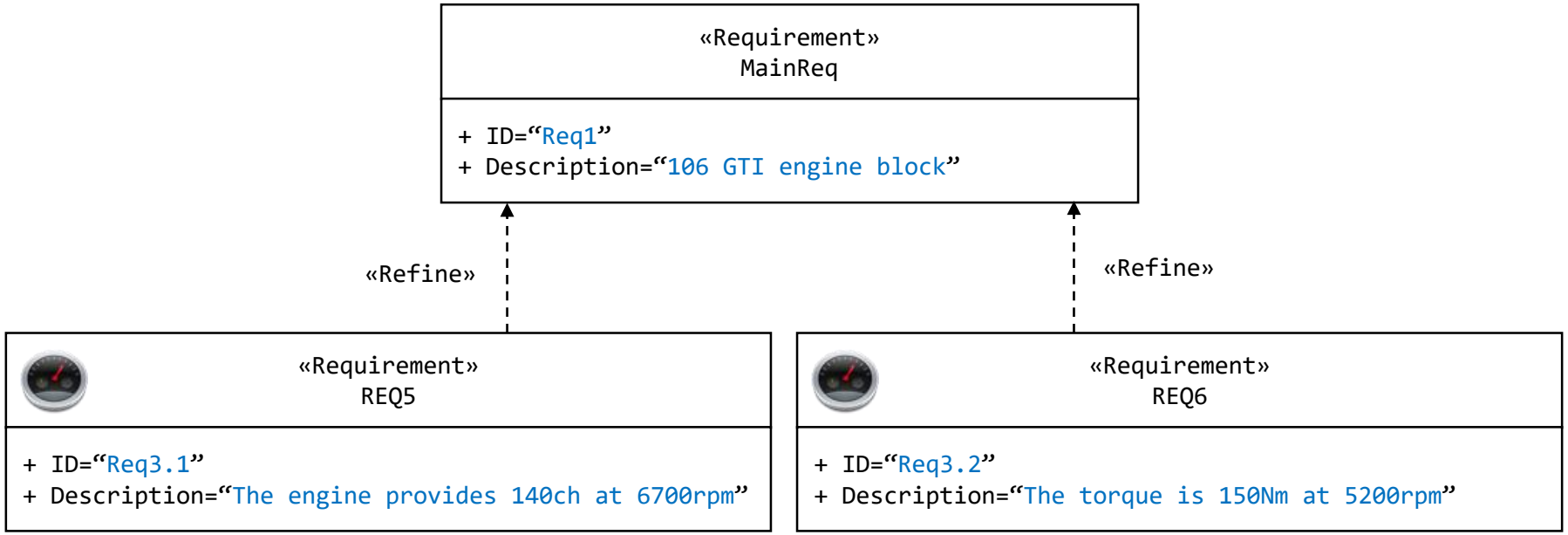
Capture *structural* requirements



Capture *functional* requirements



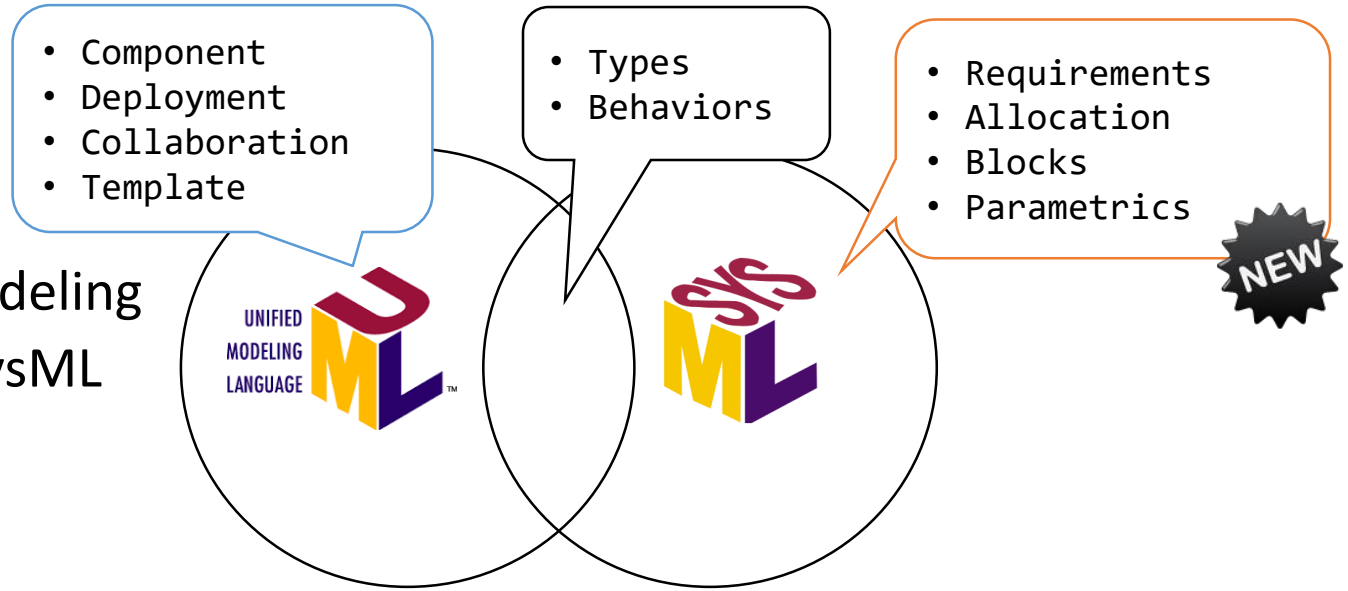
Capture *performance* requirements



Interest requirement model / related work in SysML

- Interest of the requirements model
  - Capture and organize efficiently the requirements of a system
  - Link the requirements to the model of the system
    - What are the part of the system that satisfy the requirements
  - Link the requirements to the test cases validating the system
    - What are the test cases that ensure the requirements are verified

- SysML
  - Normative UML profile
  - Specialization of UML for system modeling
  - The addition of UML compared to SysML



Add constraints

Navigate to base_Class (extension), use size list operator

- Neural networks must not have operations

```
self.base_Class.ownedOperation->size() = 0
```

Attribute of Class Meta-Model element

- Parts should not be typed

```
self.base_Property.type = null
```

- Parts should apply either the Conv1D or LSTM stereotype
  - ⇨ Introduce (abstract) «Layer» stereotype to facilitate extensibility

```
self.base_Class.ownedAttribute->forAll(attr |
   attr.getAppliedSubstereotypes(
        attr.getApplicableStereotype('SimpleNN::Layer')
   )->size() > 0))
```

Each owned attribute must apply abstract «Layer» stereotype

Exercise

- Create profile with Papyrus
- Create a new model, apply the profile (and optionally a style-sheet)