

Introduction on UML for Industrial Systems

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

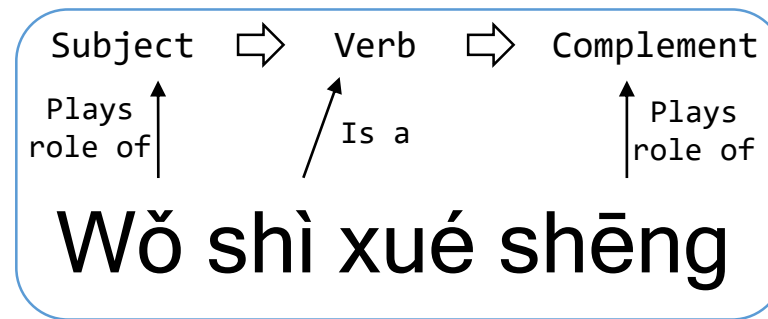
Acknowledgments – contains material from my CEA colleagues
Shuai Li, Jérémie Tatibouët, François Terrier, Sébastien Gérard

Domain specific modeling languages (DSLs)

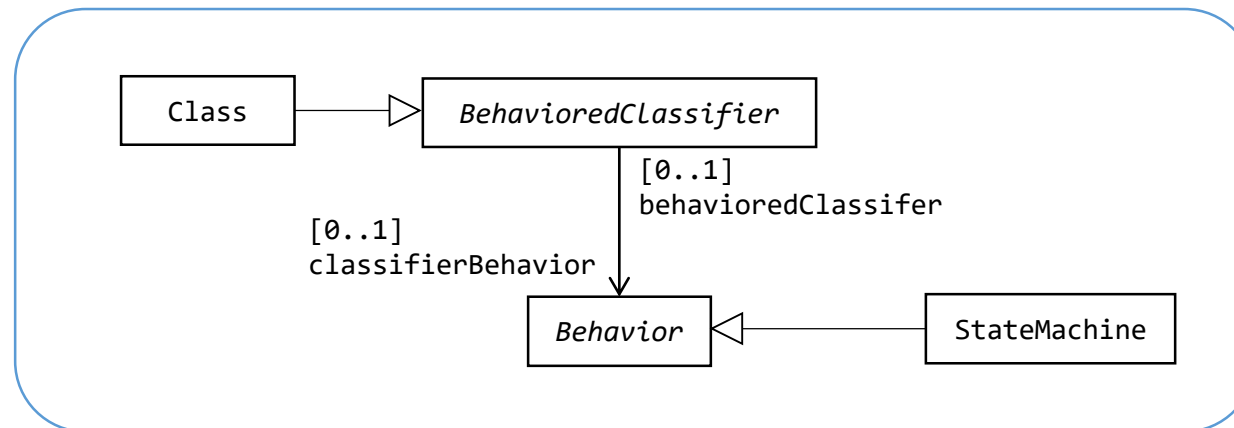
- 1. Recap – abstract syntax, notation, and semantics**
2. What are domain specific languages, what are their objectives?
3. Text-based DSLs
4. UML profile mechanisms
5. Define a simple profile

Abstract Syntax

The concepts defined by a language



Language concepts, i.e. set of syntactic elements, their relationships, and their constraints, provided by a language



Notation

The representation of language concepts

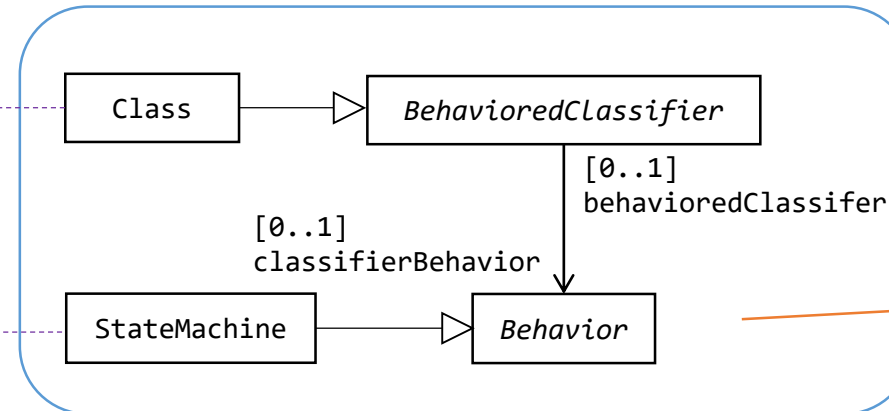
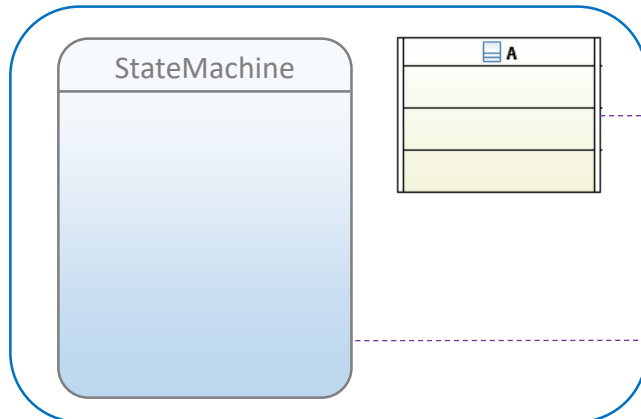
Symbolic representations associated to language concepts (i.e. elements of the abstract syntax)

我 是 学 生

Map notation symbols to syntactical elements

Subject \Rightarrow Verb \Rightarrow Complement
Plays role of \uparrow \uparrow Is a \uparrow Plays role of

我是学生
Wǒ shì xué shēng

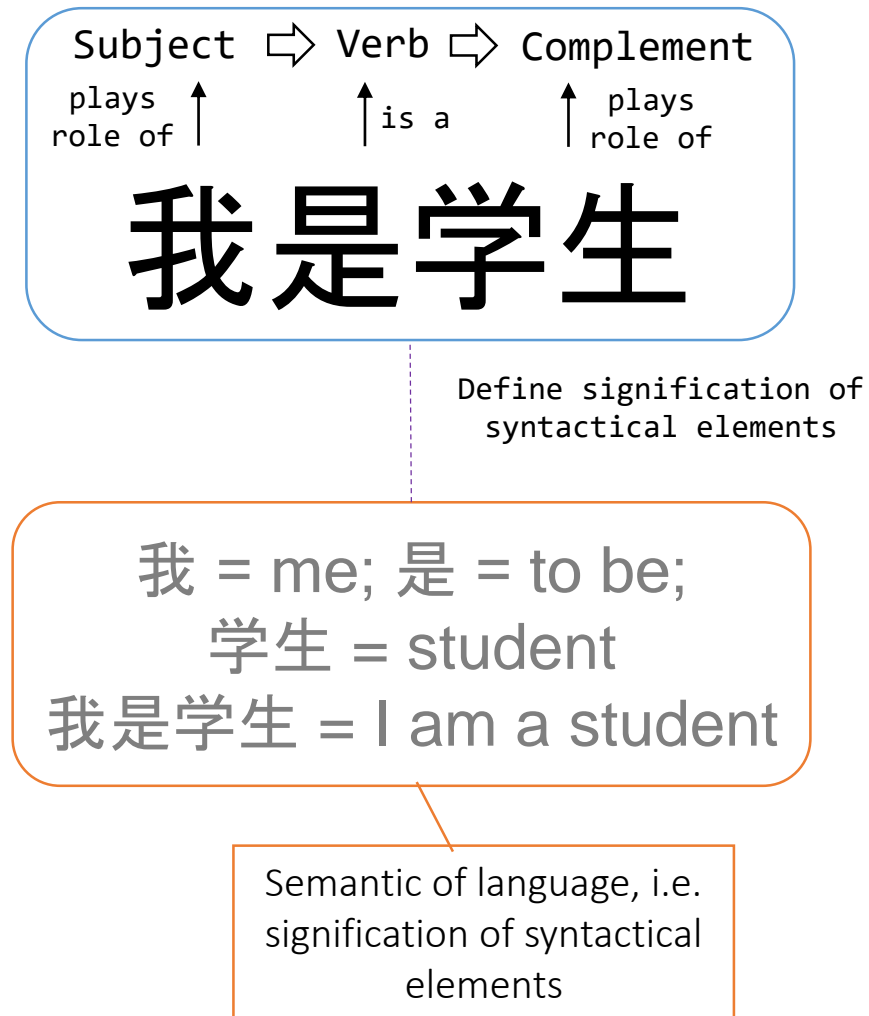


Note: some concepts in a language may not have a representation.

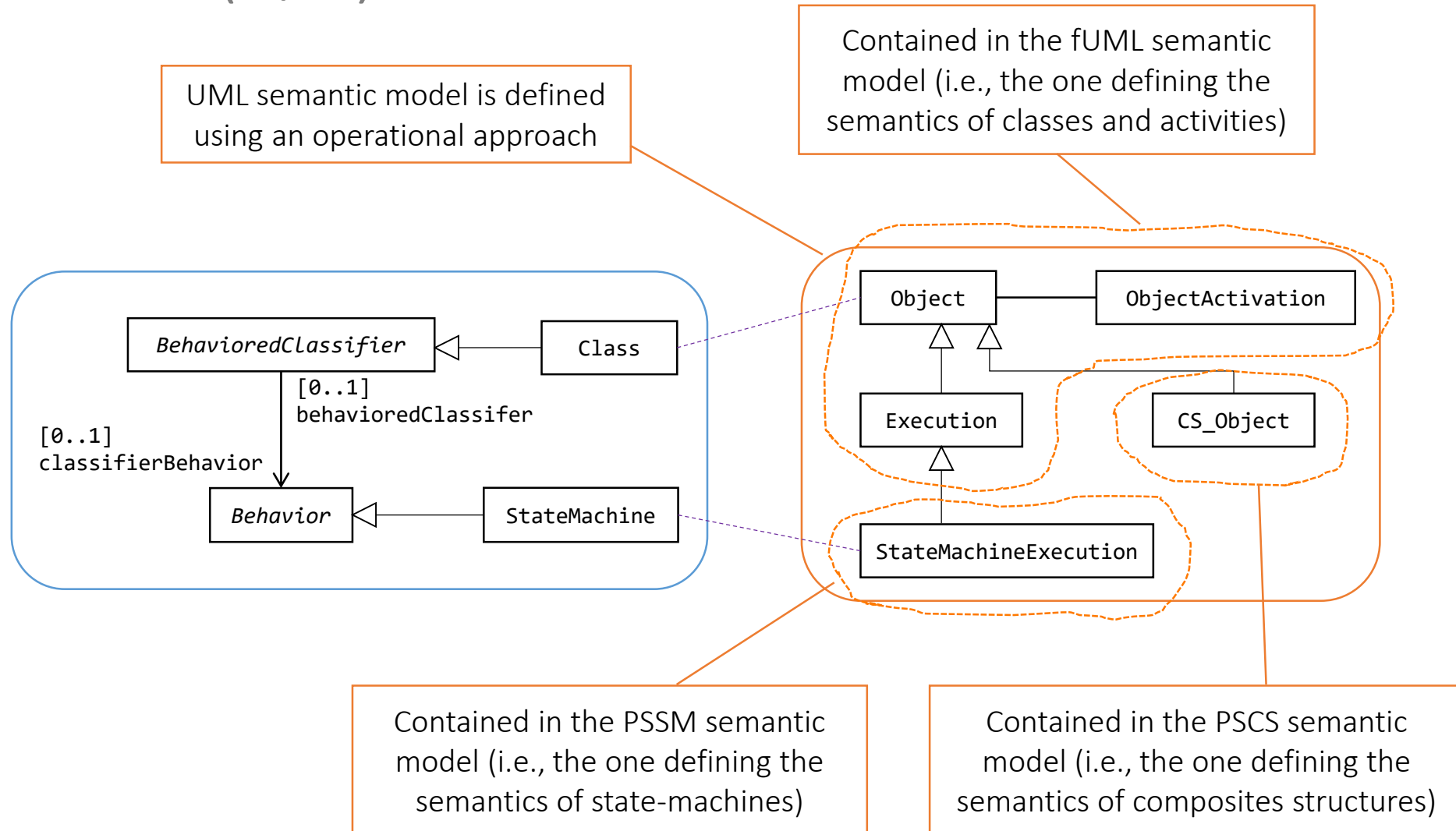
The meaning of the concept defined by a language semantics (1/2)

The semantics of a languages in computer science

- Defined by a semantic model
 - Captures the signification of concepts provided by the syntactical model
- Semantic model can be formalized using different approaches
 - Operational – semantics is defined by a program / algorithm
 - Axiomatic – semantics is defined as a set of axioms)
 - Denotational – semantics is defined as a set of denotations
 - Transformational – semantics is the one defined by the target language
- The form of the semantics defines what you can do with it
 - Execution/reasoning about valid execution traces for a model...



The meaning of the concept defined by a language semantics (2/2)



Abstract syntax, Notation, and semantics

WRAP UP

- The abstract syntax
 - The concepts that are offered by the language
 - Papyrus: you use it when you create UML elements
- The notation
 - The representation that is attached to language concepts
 - Papyrus: you use it when creating diagrams and graphical elements
- The semantics
 - The elements capturing the meaning of language concepts
 - Papyrus: you use it when you execute a model

Domain specific modeling languages (DSLs)

1. Recap – abstract syntax, notation, and semantics
2. **What are domain specific languages, what are their objectives?**
3. Text-based DSLs
4. UML profile mechanisms
5. Define a simple profile

What are domain specific languages (DSLs)?

- A language specialized to a particular application domain (contrast to a general-purpose language (GPL), which is broadly applicable across domains).
- Wide variety of DSLs, ranging from the quite generic HTML to very specific ones
- Can be further subdivided into specification/modeling languages and domain-specific programming languages.
- Special-purpose computer languages not new, term "domain-specific language" has become more popular due to the rise of domain-specific modeling.
- Separation between general-purpose and domain-specific not always sharp
 - Language may have features for particular domain but applicable more broadly
 - Conversely capable of broad application but in practice used primarily for specific domain
 - Examples: Perl and PostScript


Objectives of DSLs

1. **Use domain specific concepts and vocabular**
2. **Small, just the concepts that are required**

Domain specific modeling languages (DSLs)

1. Recap – abstract syntax, notation, and semantics
2. What are domain specific languages, what are their objectives?
- 3. Text-based DSLs**
4. UML profile mechanisms
5. Define a simple profile

Textual modeling languages

- Classical context free grammars
- Eclipse Xtext
-  langium
 - Open-source language tool supporting the Language Server Protocol
 - written in TypeScript and running in Node.js.

Context free grammars (CFGs)

- Grammar = set production rules describing all possible strings of a formal language.
- Production rules iof n the form $A \rightarrow \alpha$
- A – nonterminal symbol, α – string of terminals or non-terminals
- “context free” – production rules can be applied regardless of the symbols before/after.
- Example : $\text{Statement} \rightarrow \text{Id} = \text{Expr} ;$



Terminal

Classical tools

- Lex and yacc (free / enhanced variants flex and bison)
- Two steps
 - Lexical analysis
 - Syntactical analysis (parsing)

Regular epressions

| Meta character | Matches |
|----------------|---|
| . | any character except newline |
| * | zero or more copies of the preceding expression |
| + | one or more copies of the preceding expression |
| ? | zero or one copy of the preceding expression |
| a b | a or b |
| [] | character class |

[A-Za-z0-9]+ at least one alpha numerical character

[A-Za-z][A-Za-z0-9_]* start with a alphabetic character, followed by zero or more
alpha numerical characters (+ underscore)

Example

source code

a = b + c * d

Lexical Analyzer

Lex

patterns

tokens

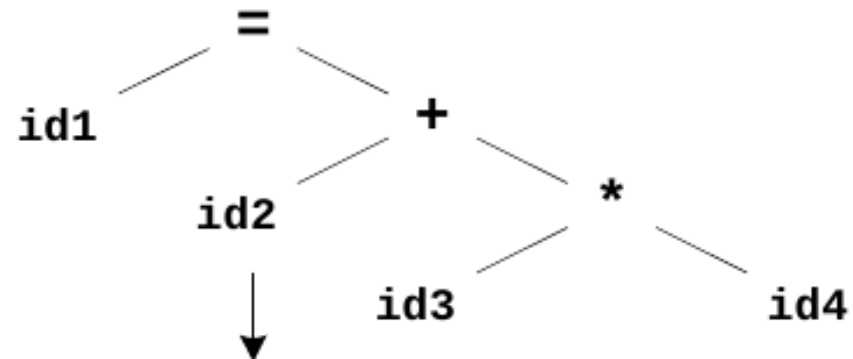
id1 = id2 + id3 * id4

Syntax Analyzer

Yacc

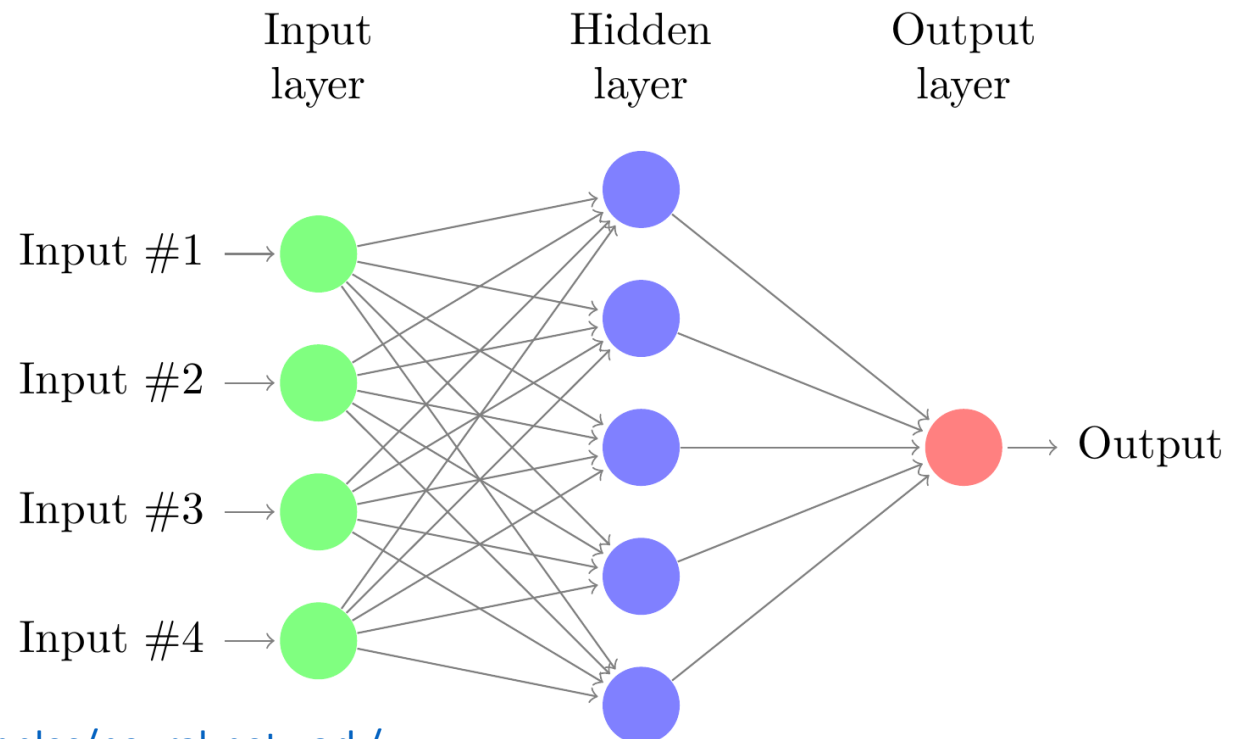
grammar

syntax tree



Other example – neural network descripton

- Keras <https://keras.io/>
- Python library/framework for describing and executing neural networks
- High level API
- Goal : Declarative description of neural network



Source: <https://texample.net/tikz/examples/neural-network/>

Neural network descripton

- Different layer types in Keras
- Each layer has a set of parameters
 - Simplification – focus on two layers and a subset of the parameters
- Convolutional layer Conv1D
 - filters: Integer, dimensionality of output space (i.e. number of output filters in the convolution).
 - kernel_size: An integer specifying the length of 1D convolution window.
- LSTM layer (Long Short-Term Memory layer – recurrent layer)
 - activation: Activation function to use.
Two options: **tanh** - hyperbolic tangent, **linear** - $a(x) = x$.
 - use_bias: Boolean (default True), whether the layer uses a bias vector.

Other example (from neural networks)

Possible textual description

```
network MyNN {  
  l1 : Conv (kernel_size = 2)  
  l2 : LSTM (use_bias = true, activation = tanh)  
  l3 : LSTM (activation = linear)  
}
```

Howto describe this editor formally?

Use Eclipse xtext

- <https://www.eclipse.org/Xtext/>

“Xtext is a framework for development of programming languages and **domain-specific languages**. ... define your language using a powerful grammar language. ... get parser, linker, typechecker, compiler as well as editing support for Eclipse, any editor that supports the Language Server Protocol and your favorite web browser.”

Xtext formalization

```
grammar org.xtext.example.mydsl.NNDsl with org.eclipse.xtext.common.Terminals
```

```
generate NNDsl "http://www.xtext.org/example/nndsl/NNDsl"
```

Model:

```
'network' name=ID '{'  
    (layers += Layer)*  
'}';
```

Layer:

```
name=ID ':' (  
    'Conv' options += COptions |  
    'LSTM' options += LOptions  
);
```

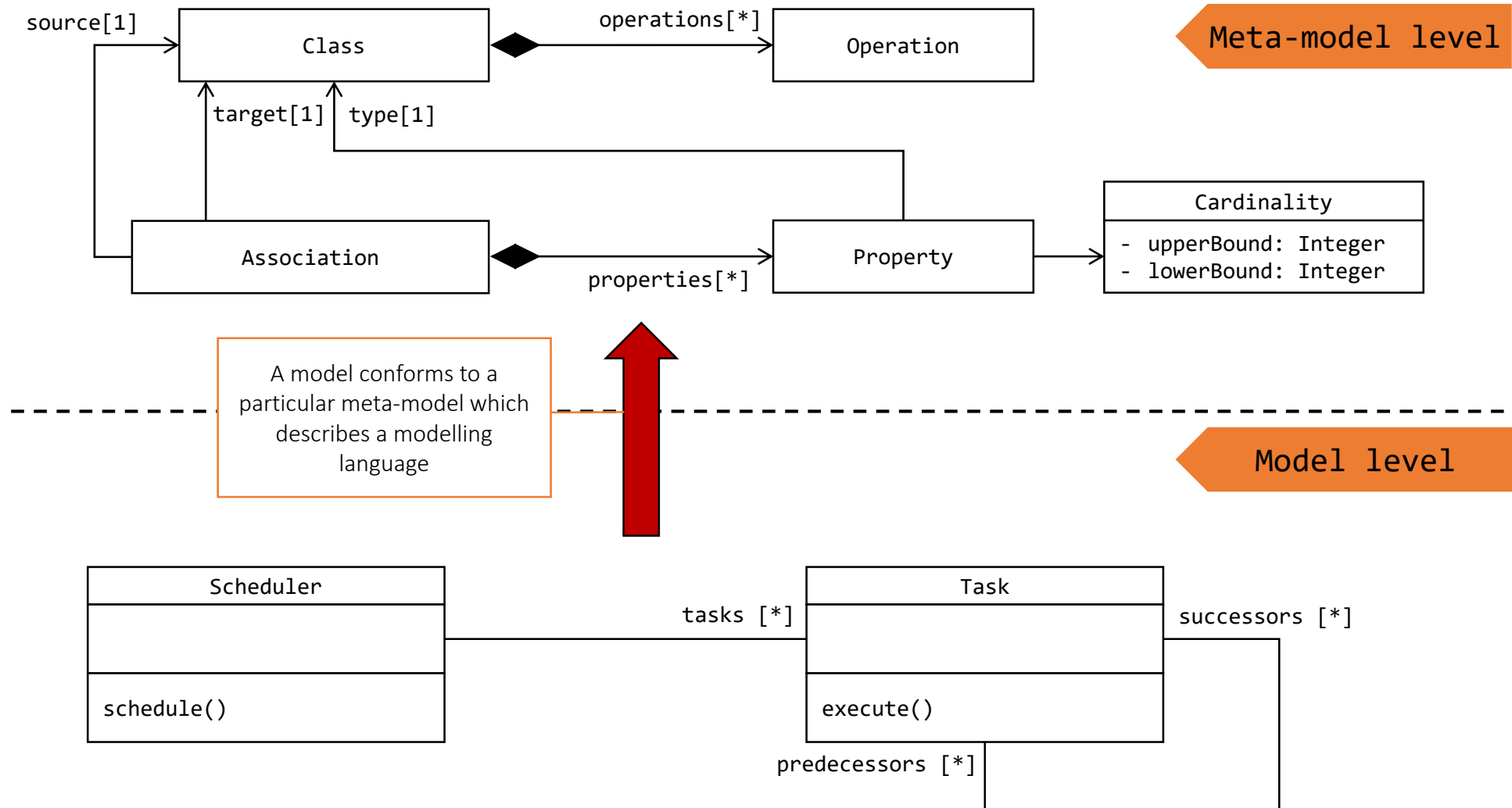
...

(see in Eclipse editor, exercise)

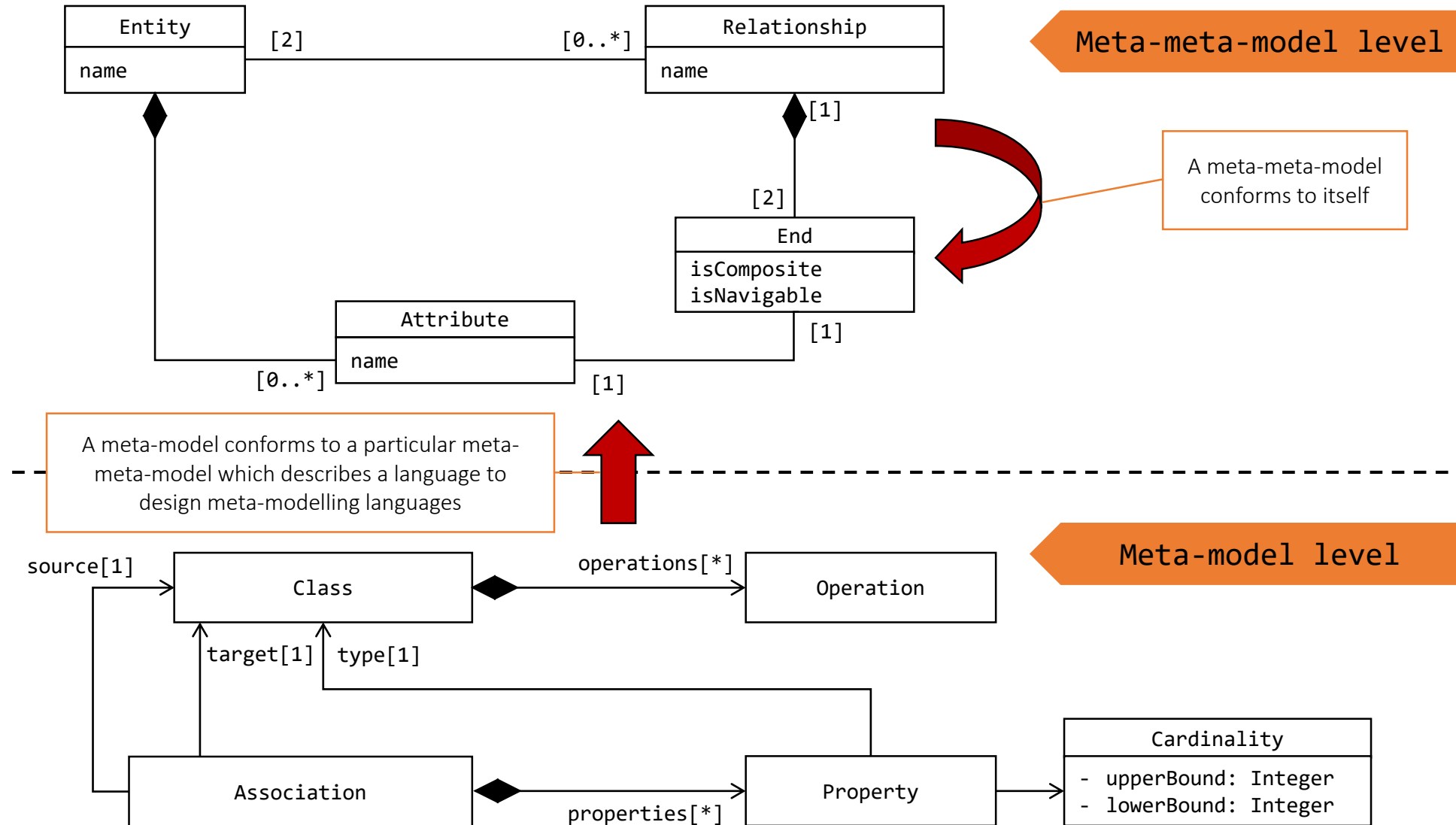
Domain specific modeling languages (DSLs)

1. Recap – abstract syntax, notation, and semantics
2. What are domain specific languages, what are their objectives?
3. Text-based DSLs
4. **UML profile mechanisms**
5. Define a simple profile

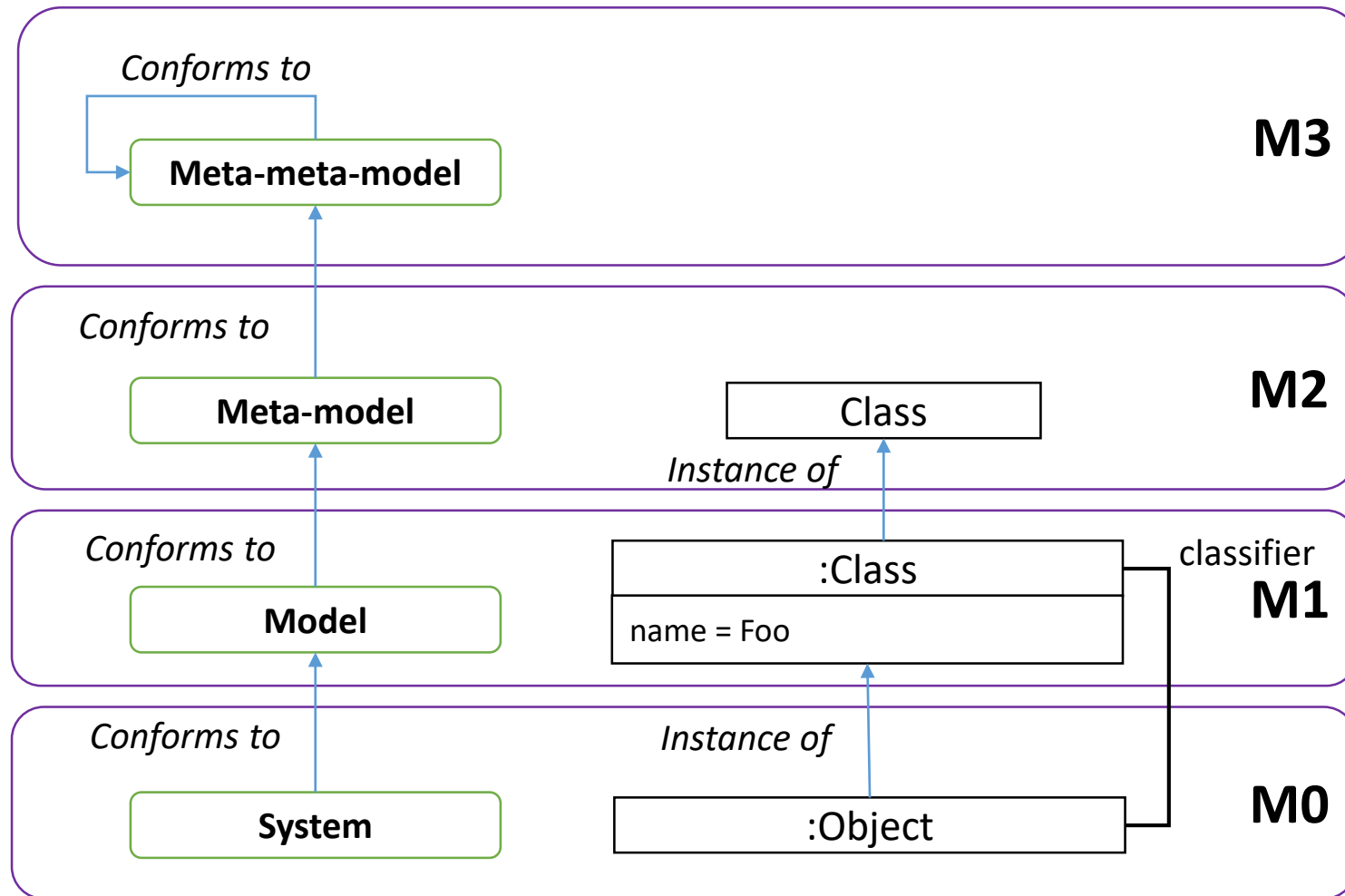
Model and metamodel – conformance relationship



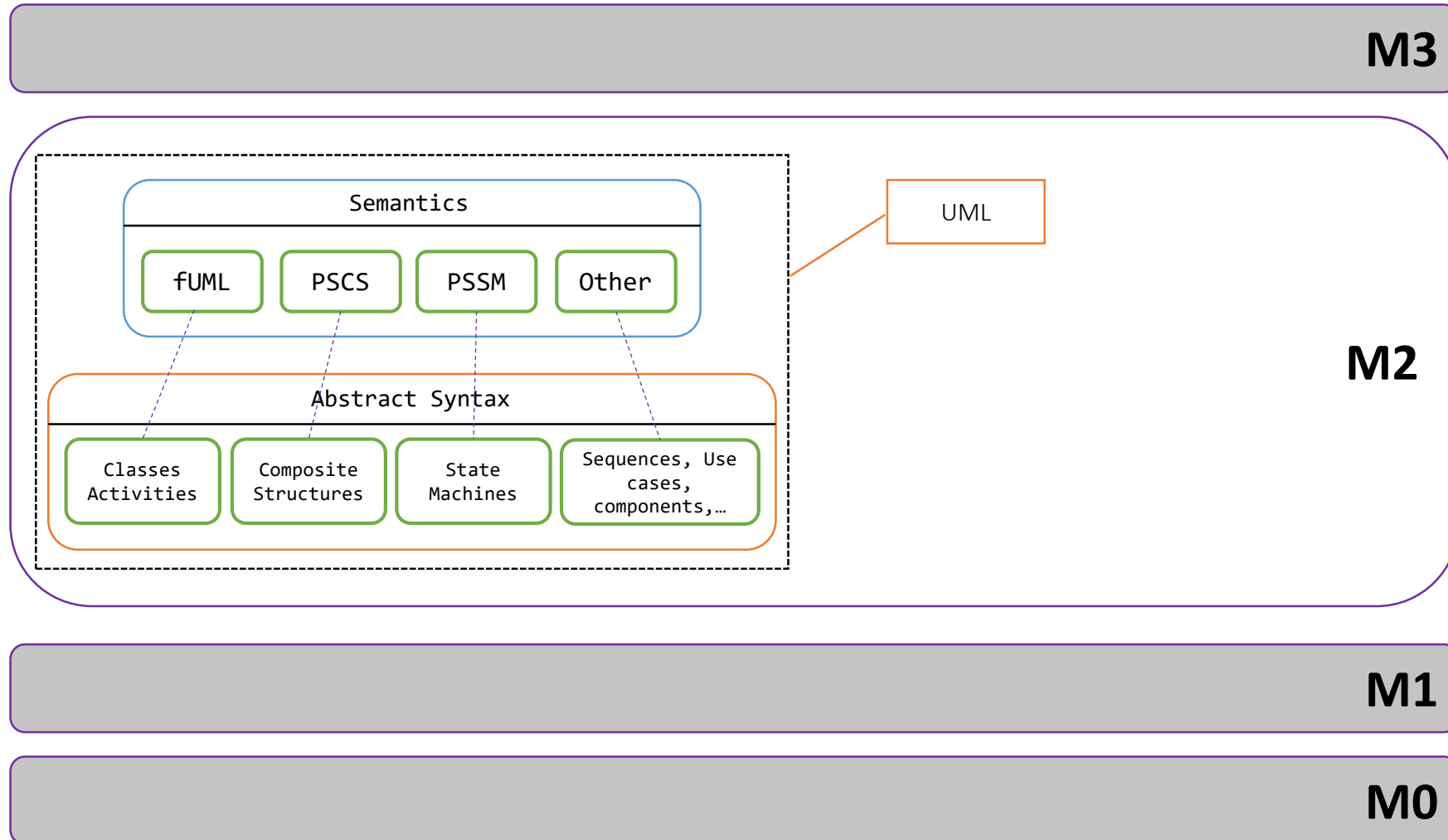
Metamodel and meta-meta-model – end of the loop



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



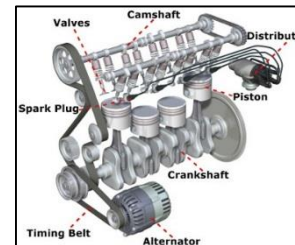
WHERE IS UML ?



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
 - But it is clearly software oriented
 - Many concepts come from object oriented programming languages
- Will you use it as it is 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/>

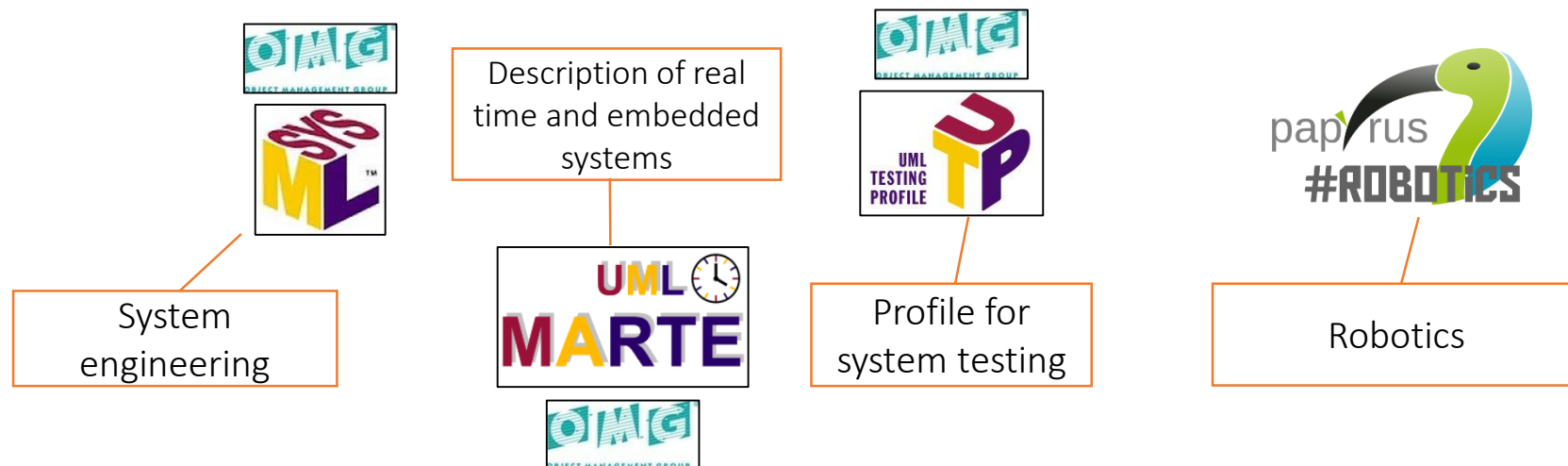


Points of interest: processor rate, the quantity of memory, energy consumption?

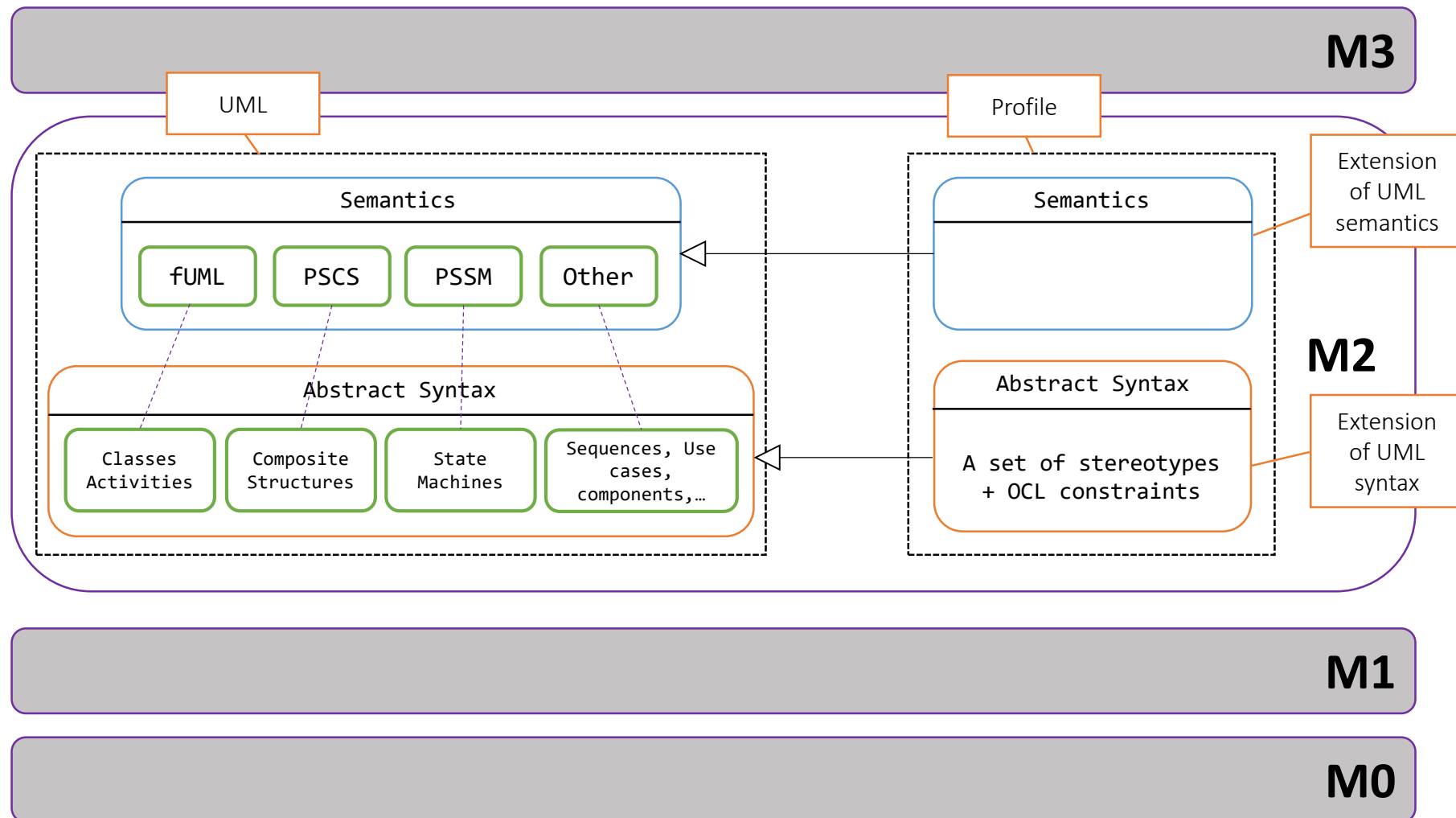
UML extensibility – the profile mechanism

Difficulty to capture concerns of specific domains

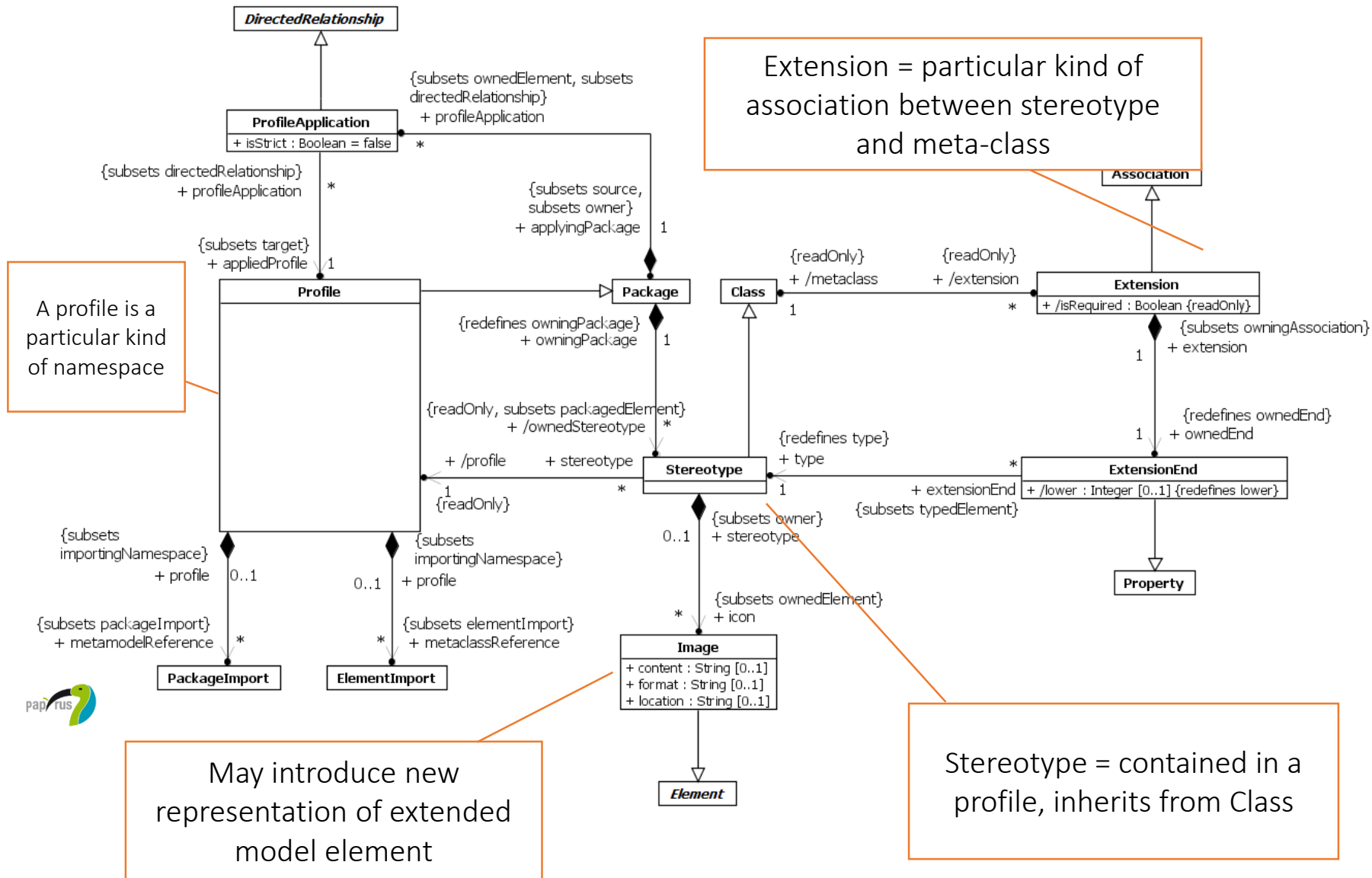
-such as mechanics, electronics, avionic, etc.
- Extend UML to capture the concerns of these domains
 - Enable design of UML based domain specific languages
 - Capitalize on syntax and semantics provided by UML
- UML profiles are widely used in industry



How does a profile contribute to UML?



Profiles - UML meta-model view

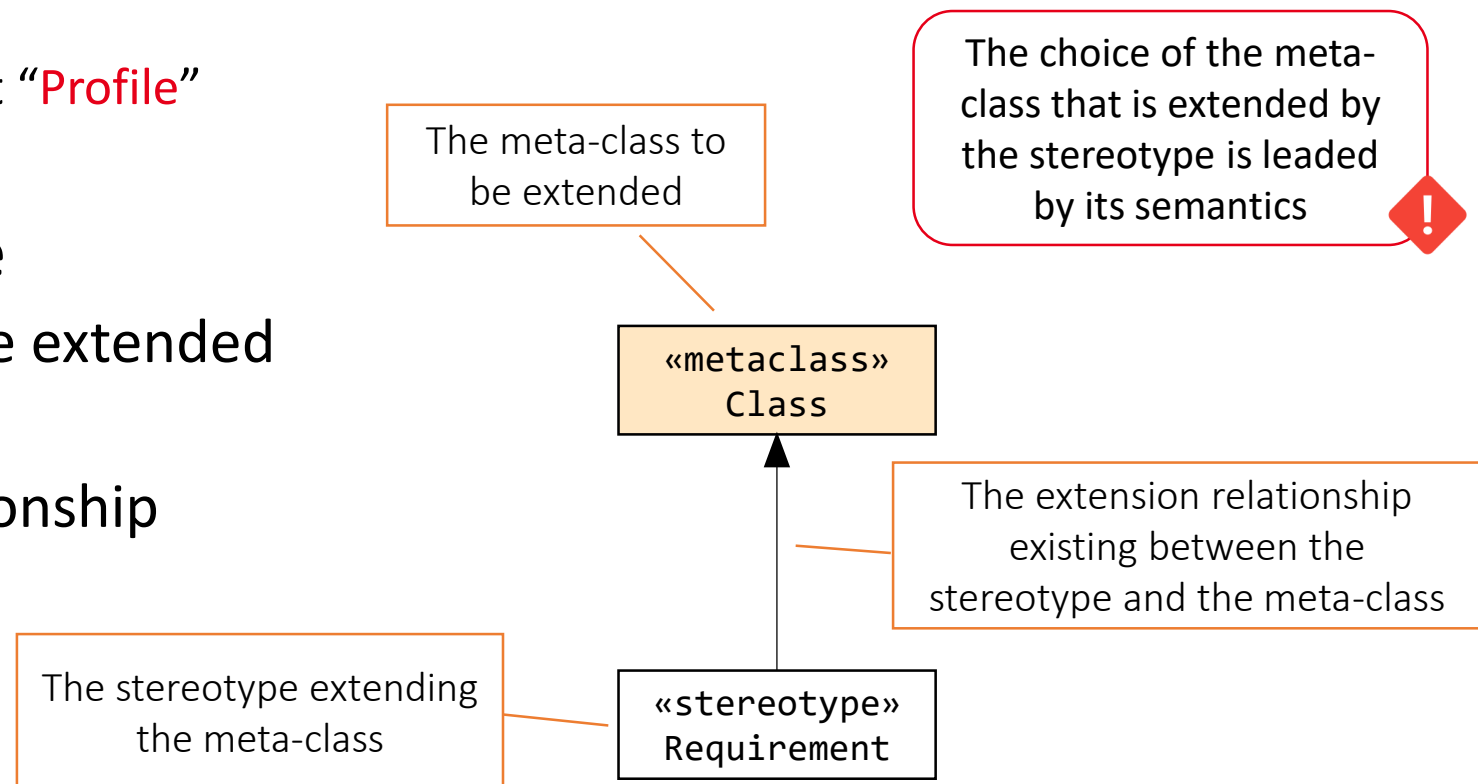


Example – a profile for requirement engineering

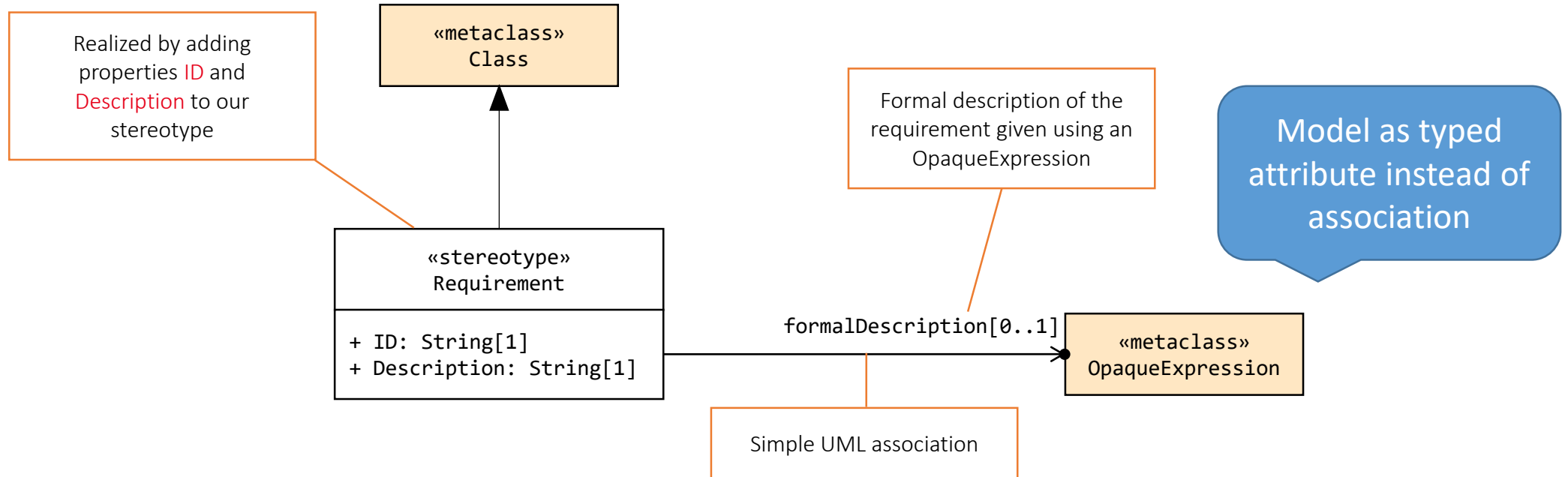
- Objective – Capture system requirements
- Analysis
 - In my system I can have
 - Functional requirements (actions that can be accomplished by the system)
 - Performance requirements (expected performance for the system)
 - Structural requirements (Identify the necessary structure of the system)
 - 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



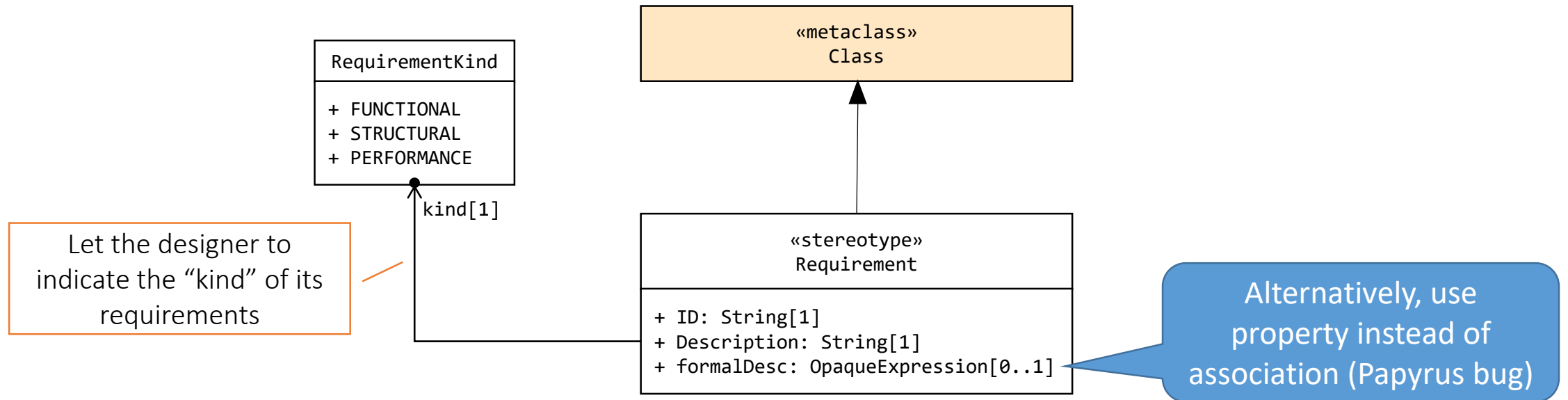
Step 2 – capture basic properties of a requirement



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?



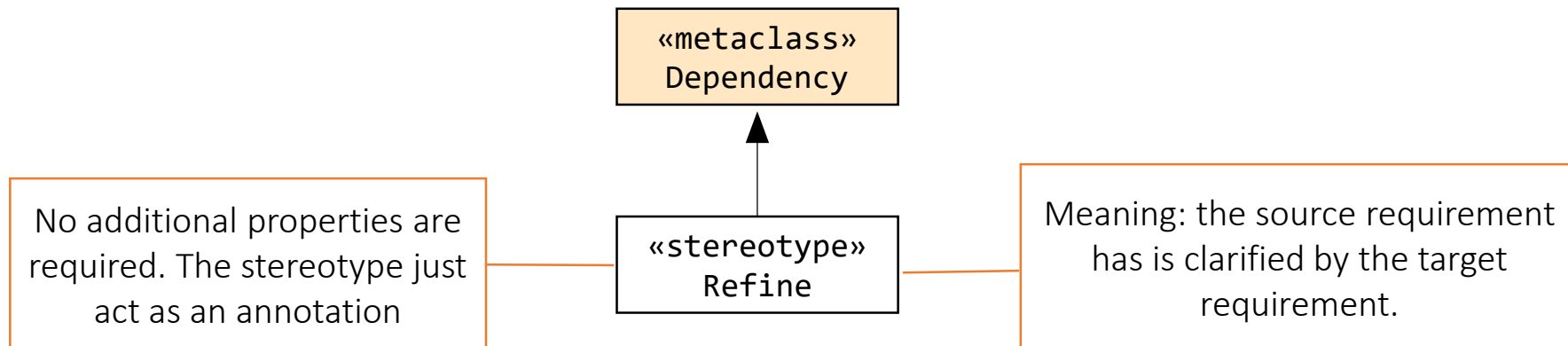
Step 3 – Refinement relationship

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 and How

Why? We do not need some features when specifying requirements

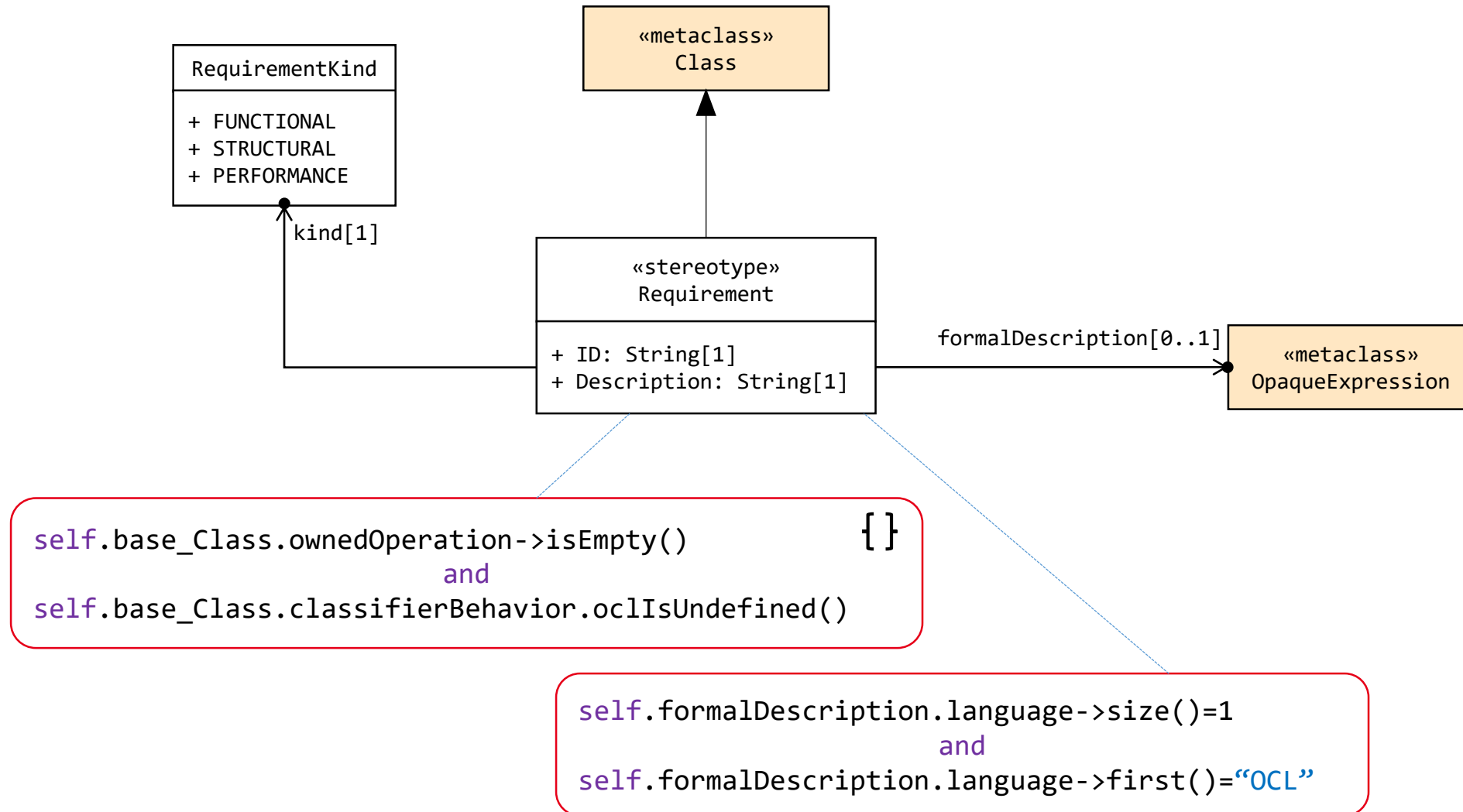


- Class
 - ~~A class can have operations~~
 - ~~A class can have a classifier behavior~~
 - ~~A class can be active~~
- Dependency
 - ~~The source and the target can be NamedElement~~

How? Via OCL constraints in our profile

- Object Constraint Language
- OCL is a formal language
- Specification of constraints on models (not only UML)
- The constraints that can be verified at any time of the model life

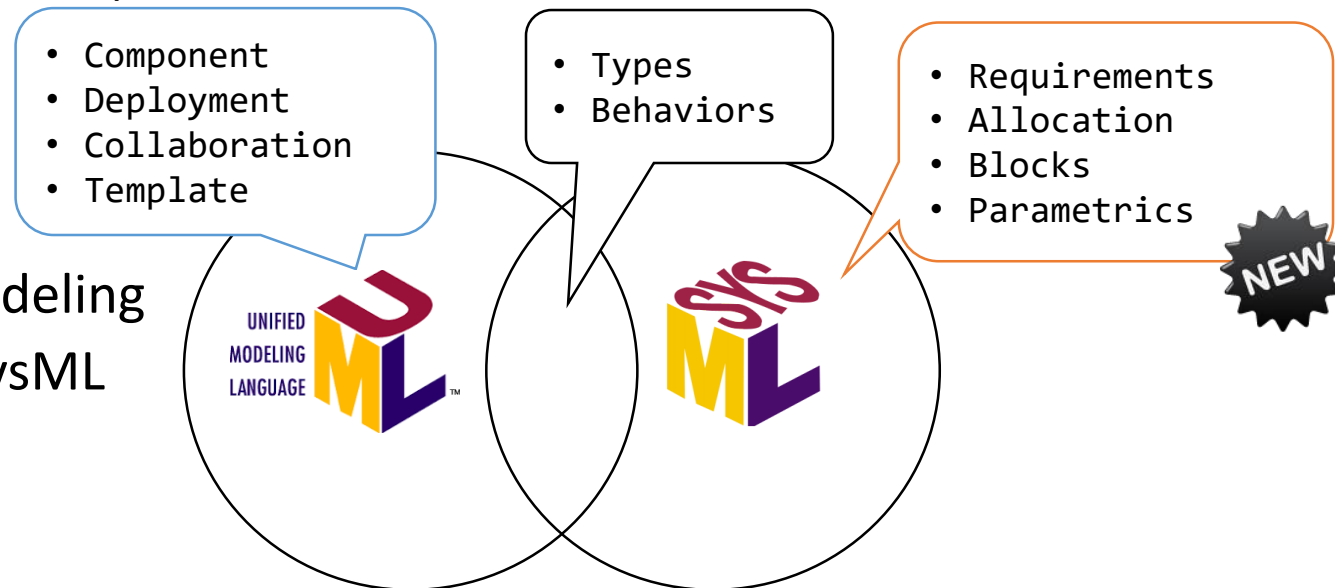
Reduce expressiveness – «Requirement» stereotype



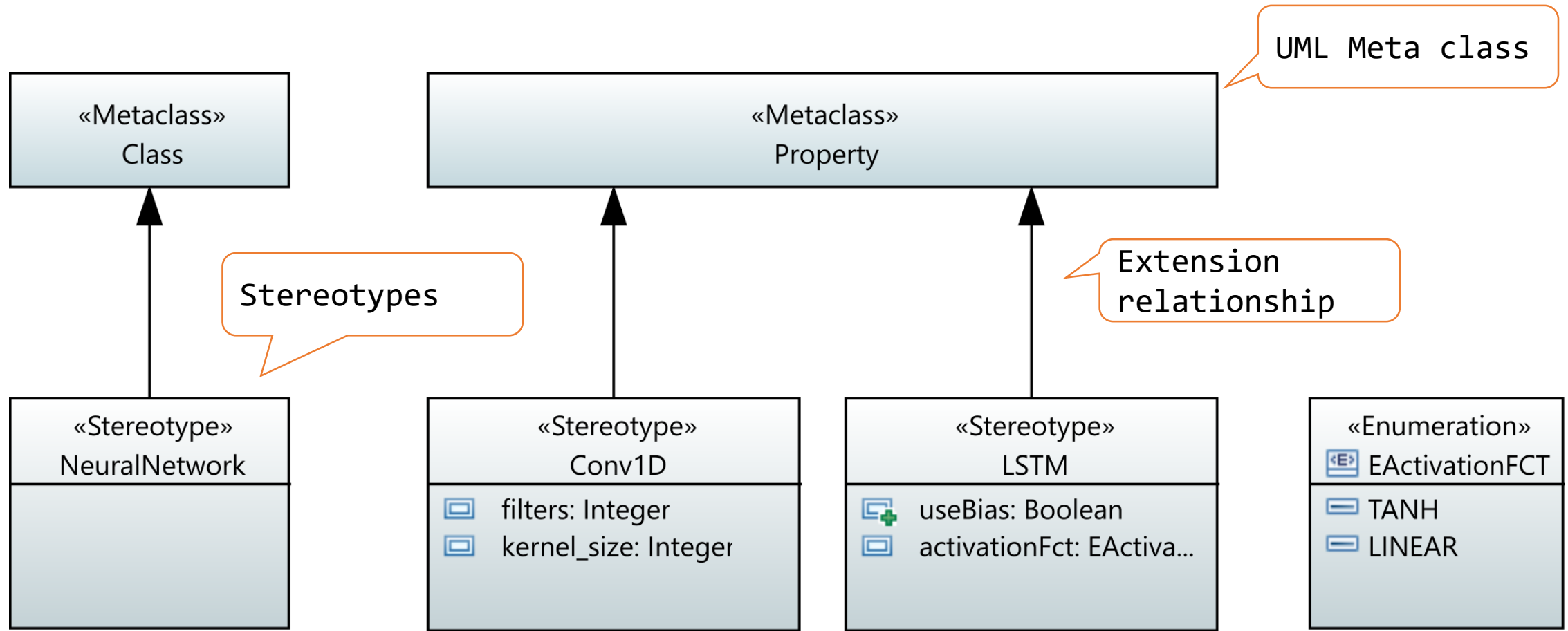
Interest requirement model / related work in SysML

- 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

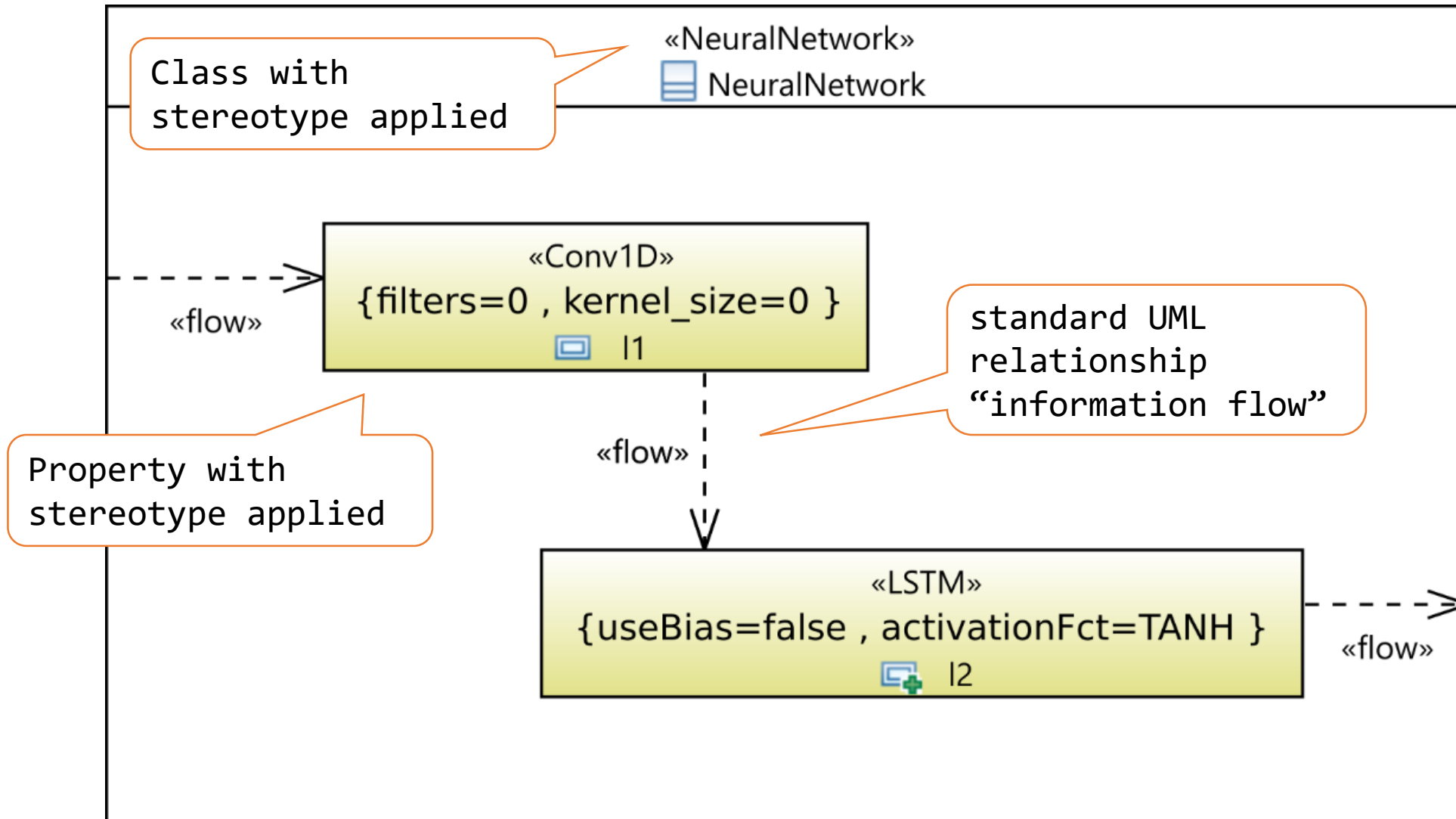


Simple Neural Network profile



Sample model applying the profile

Composite structure diagram
CSS (style sheet)
modifies appearance



Specify valid models

- Restrict UML meta-models having a lot of features
- Use OMG OCL (object constraint language)
 - <https://www.omg.org/spec/OCL/>
- 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(...)”

CSS – Cascading style sheet

```
Class {  
    fillColor: white;
```

Background color of
compsite class is white

```
}  
Property {  
    maskLabel : name;  
    fillColor: #e0e0a0;
```

Show only name as label
(not visibility,type, ...)

```
}  
Property > Compartment[kind="internalstructure"] {  
    visible : false;
```

Hide “internal structure”
compartment

```
}  
Compartment[type=StereotypeBrace] {  
    visible : true;
```

Stereotype attributes in Braces
{ ... } after stereotype name

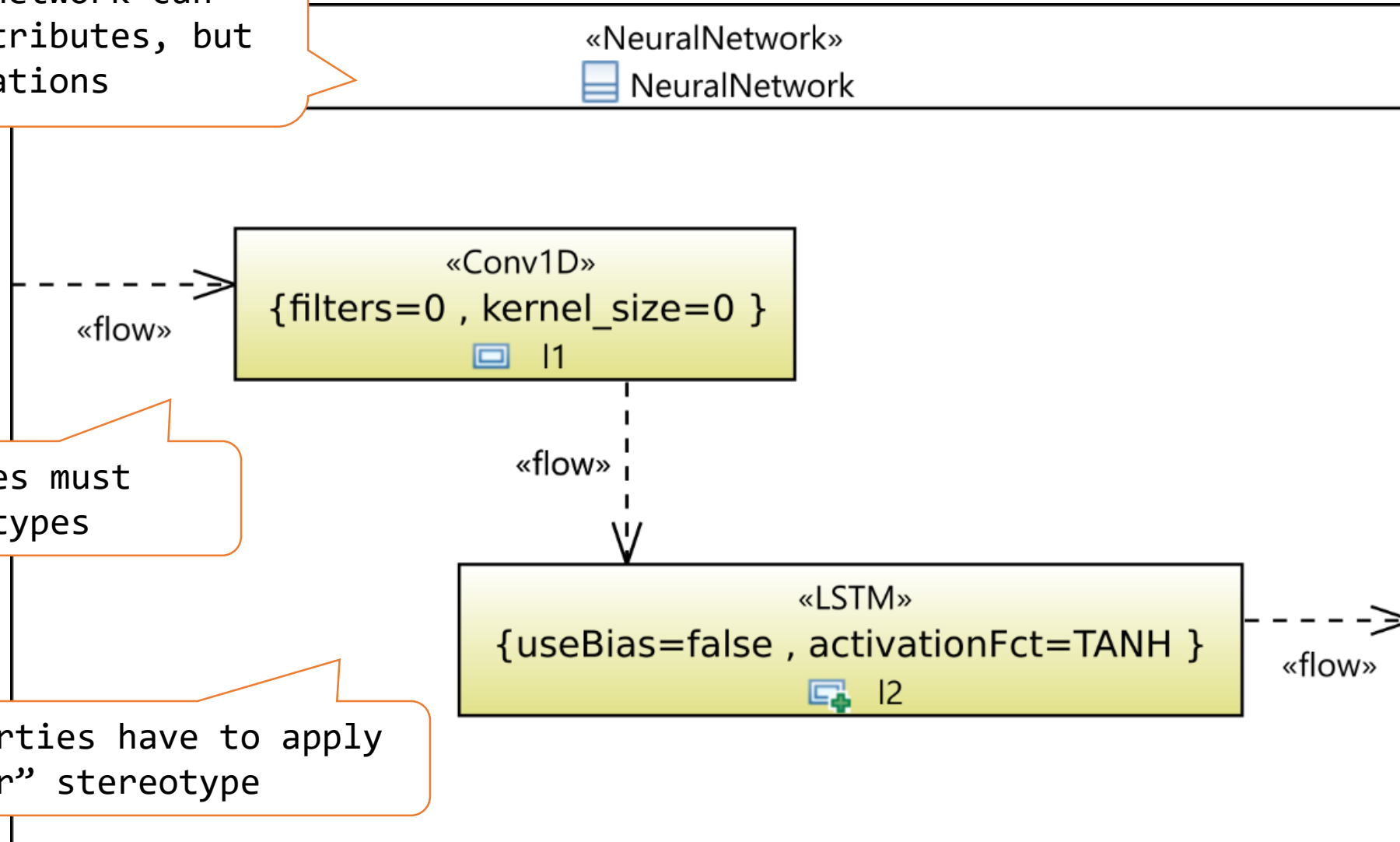
```
}
```

Sample model, constraints

Neural network can have attributes, but no operations

Properties must have no types

Properties have to apply "Layer" stereotype



Add constraints

Navigate to base_Class (extension),
use **size** list operator

- Neural networks must not have operations

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

- Parts should not be typed

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

Attribute of Class Meta-Model element

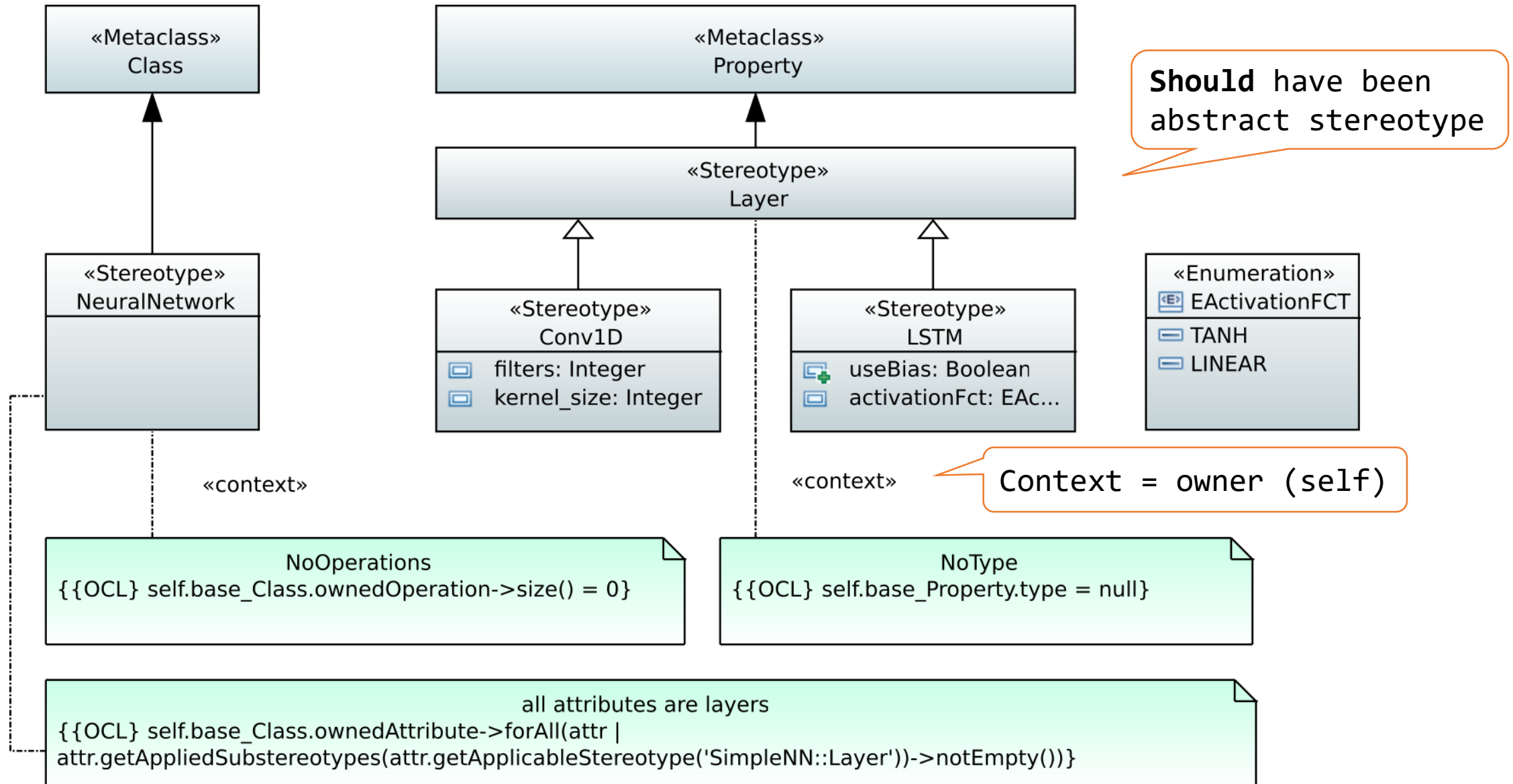
- Parts should apply either the Conv1D or LSTM stereotype

⇒ Introduce (abstract) «Layer» stereotype to facilitate extensibility

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

Each owned attribute must apply
abstract «Layer» stereotype

Resulting profile



Useful OCL (list) operations

- **exists**(boolean condition on list element) – true, if at least one element of the list fulfills the condition
- **forAll**(boolean condition on list element) – true, if all elements of the list fulfill the condition
- **select**(boolean condition on list element) – filtered list containing only elements with condition evaluating to **true** (**reject**(...) ... to **false**)
- **collect**(attribute of list element) – new list containing a collection of the attribute values
- **isEmpty**(), **notEmpty**(), **size**() – checks on list size

Exercise

- Create profile with Papyrus
- Create a new model, apply the profile and style-sheet
- Create the layers and flow