L4. DDD and MDE

S2. Metamodelling with MOF

Dr A Boronat

Table of Contents

Metamodel-Centric Language Design

- Metamodelling refers to the definition of modelling languages, with which models can be defined.
  - UML itself is defined in terms of a metamodel, called the MOF (Meta-Object Facility). The MOF reuses part of the UML as its representation
- A metamodel constitutes the abstract syntax of a modelling language
  - Concepts and relationships among them
  - Independent of the concrete syntax
- Model validation: models are validated against constraints defined in the metamodel
  - OCL as the language of choice for defining constraints
- Metamodels are employed
  - constructively by using the metamodel as a set of production rules for building models
  - analytically by using the metamodel as a set of constraints that need to be satisfied by a model in order to conform to its metamodel

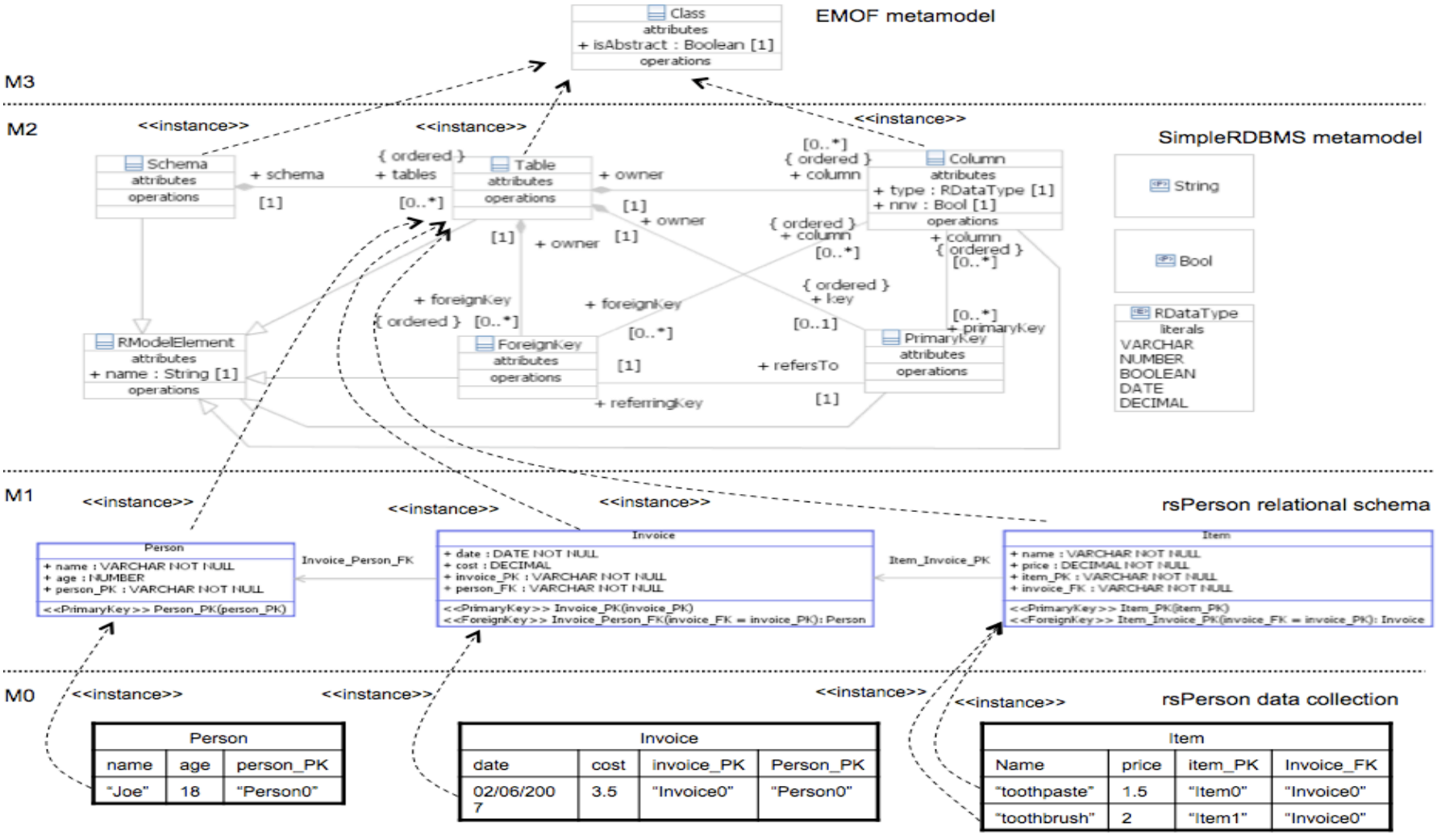
Benefits of Metamodelling

- Precise language definition
  - Models become unambiguous if they are formally defined
  - Models can become more succinct and more expressive if suitable metamodels are defined and used
  - Model exchange and interpretation is only possible if they are based on a formal metamodel
- Accessible language definition
  - The core of UML is used to define metamodels
- Evolvable language definition
  - OO extension mechanisms are available for adapting metamodels
- Metamodelling frameworks:
  - exchange formats: support to serialize/deserialize models into XML format to support metadata exchange
  - model repositories: models may be stored to and retrieved from a model repository
  - model editors for defining and validating models

Meta-Object Facility (MOF)

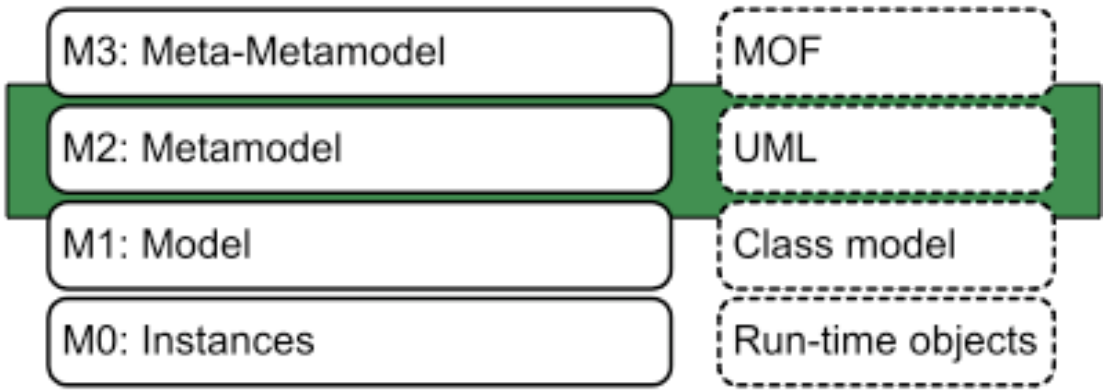
- MOF is a closed metamodelling architecture; it defines an M3-model, which conforms to itself.
- MOF allows a strict meta-modelling architecture; every model element on every layer is strictly in correspondence with a model element of the layer above.
- MOF only provides a means to define the structure, or abstract syntax of a language or of data.
- For defining metamodels, MOF plays exactly the role that EBNF plays for defining programming language grammars. MOF is a DSL used to define metamodels as much as EBNF is a DSL for defining grammars.
  - A grammar defines all valid sentences of a language
  - A metamodel defines all valid models of a modelling language
- Similarly to EBNF, MOF is defined in MOF.

MOF Framework



Where do we find metamodels?

- Metamodelling happens on the M2 Metamodel layer of the MOF framework



- The four-layer metamodelling stack assumes that a model on layer M conforms to a model on layer $M + 1$:
  - Models (M1) are represented as a collections of instances of classes defined in a metamodel (M2)
  - Changes to the metamodel have an impact on representation of the instances of this metamodel (i.e. the model itself)

MOF Modelling Primitives: classes

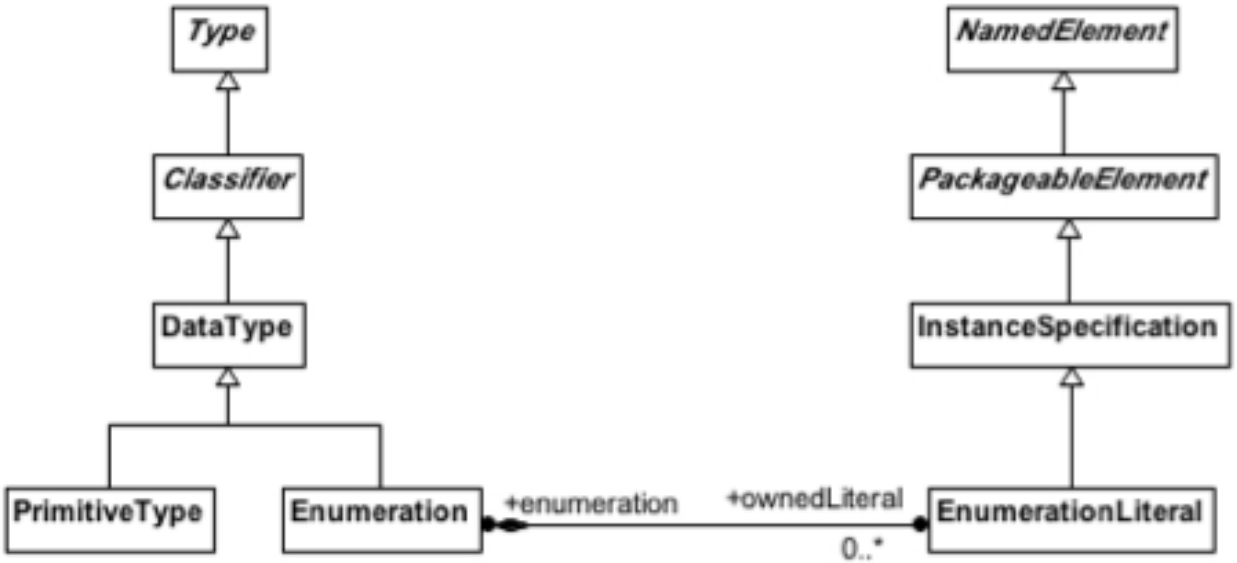
- A class is an abstraction modelling an entity in the application or solution domain
  - the class name is the only mandatory information
  - abstract classes are never instantiated but they are useful together with generalization relationships
- Attributes characterize a class
  - properties typed with a data type
  - basic syntax: *name : type*
  - default values: initial value for an attribute: deadline : Date = today()
  - derived attribute: their value can be determined from the values of other attributes
- Multiplicity: to indicate the number of separate values that an attribute could hold

n..m : n is the lowest value, m is the greatest value. middlename: String[0..2]
n : to indicate a fixed number of attributes
{ordered} : the different values are ordered address: String[1..*]{ordered}
{unique} : no duplicate values phoneNumber: Integer[1..*]{unique}

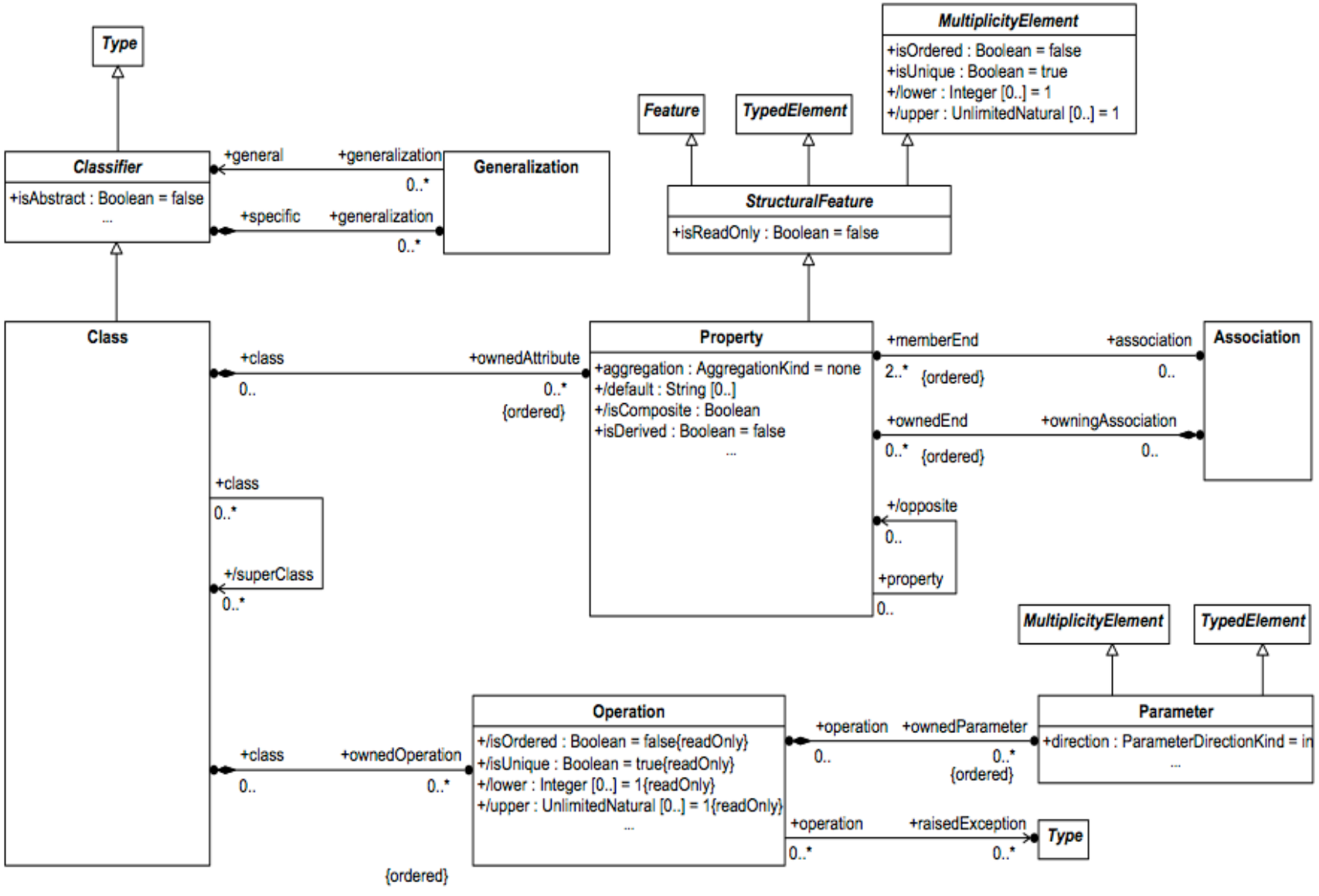
MOF Modelling Primitives: relationships

- superClass:
  - to define generalizations
  - multiple inheritance is allowed
- references (using EMF terminology)
  - properties typed with a class
  - implement one end of an association
  - lower and upper bounds: shape the cardinality of the association end
  - isUnique: all referenced elements must be different
  - isOrdered: all the elements that are referenced are ordered
  - isComposite:
    - to indicate the container of a particular concept
    - cyclic containment is invalid
    - only one container property may be non-null
  - opposite:
    - to define bidirectional associations

MOF Modelling Primitives: data types



MOF Modelling Primitives



DSL Design: Domain Analysis

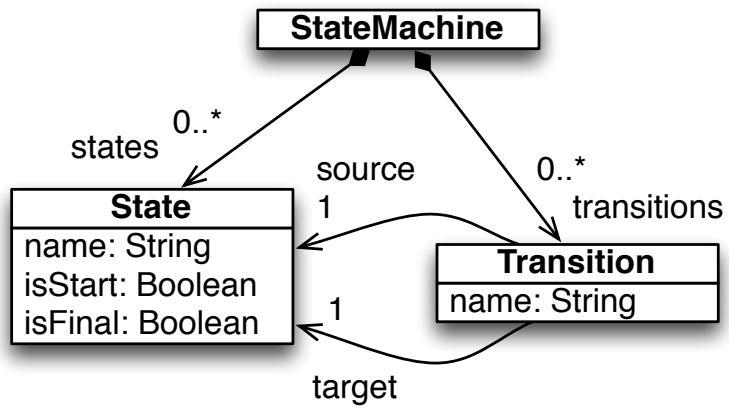
1. Identify the purpose of the language
   - the kind of models that can be defined
   - what kind of queries should they support?
2. Identify the users of the language and their terminology
   - this involves eliciting information from domain experts
     - from existing programs: find reoccurring patterns in the program code
     - analyse a particular domain: analysis documents or expert interviews
3. Define the contents
   - decide on a vocabulary of enumerations and classes, and their properties
   - define a list of concepts and their defining properties
     - intrinsic properties: have only primitive data values
     - extrinsic properties: represent relationships between modelling constructs
4. Define a model using the metamodel
5. Refine the metamodel

Exercise

- Model the abstract syntax for defining state machines as follows:
  - A state machine consists of states and transitions, both labelled with a name.
  - A state machine has one initial and it may have many final states.
  - A transition has one source state and one target state, which may be the source one.
- Solution:

Exercise

- Model the abstract syntax for defining state machines as follows:
  - A state machine consists of states and transitions, both labelled with a name.
  - A state machine has one initial and it may have many final states.
  - A transition has one source state and one target state, which may be the source one.
- Solution:

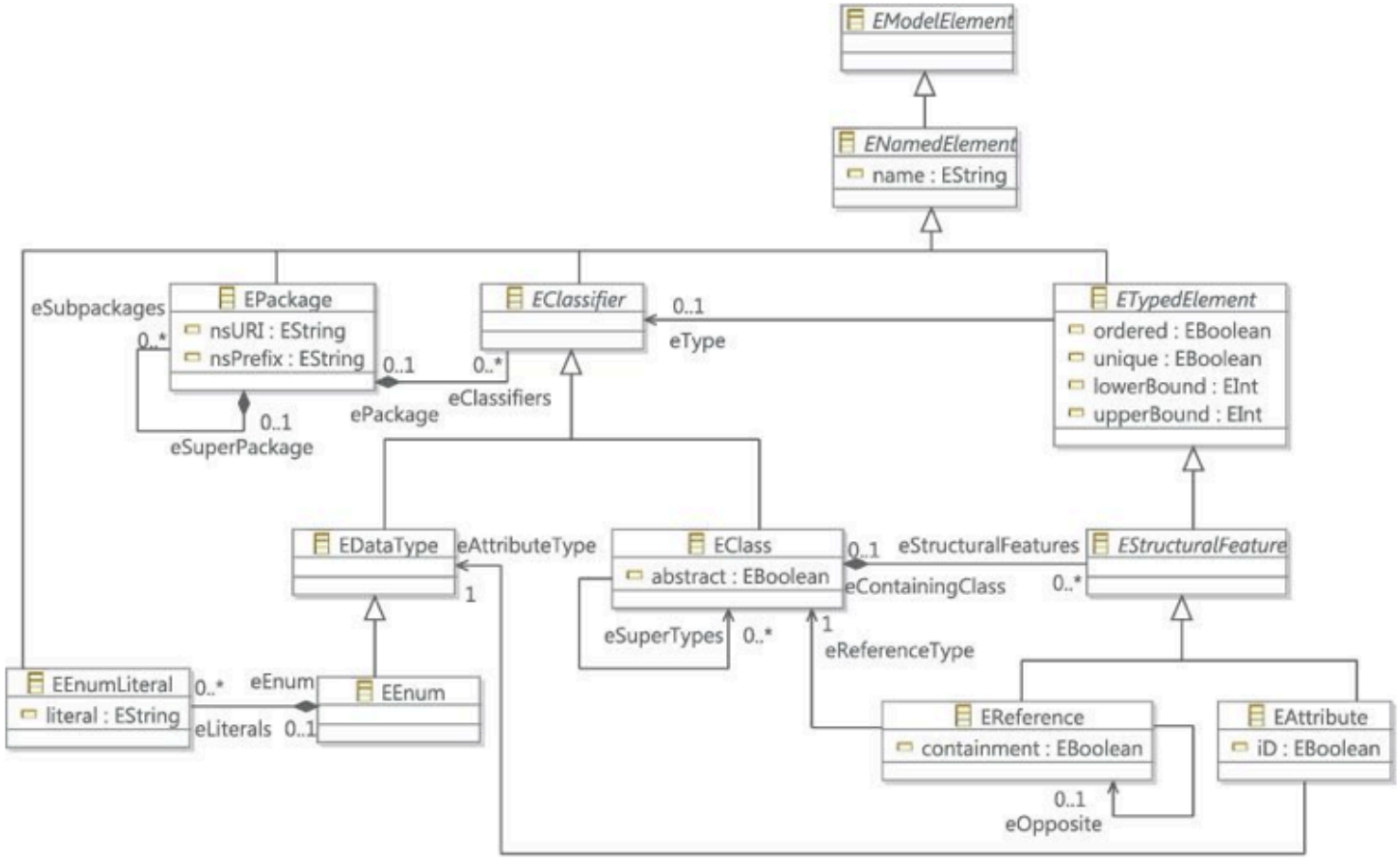


Model representation

- Metamodels define the types that can be used in models
  - Data types
  - Object types
- A model is represented by a collection of objects that are typed with the object types that are defined in a metamodel
  - Models can be represented as object diagrams
  - Models can be viewed as graphs: objects are nodes and references are unidirectional edges
- A model conforms to a metamodel iff it is syntactically well-formed, i.e. if
  - it is defined by means of the modelling primitives (types) that are provided in the metamodel
  - it satisfies the metamodel constraints

Eclipse Modelling Framework

- Metamodeling framework atop the Eclipse platform.
- Ecore metamodel is the realization of the MOF meta-modeling language:



OCL and Metamodelling

- OCL provides a way to develop more precise metamodels using MOF
- An OCL constraint is a rule attached to a UML element restricting its semantics.
  - In a modelling language, OCL constraints restrict what can be modelled with the metamodel.
  - An invariant is a property that must remain true for all the instances of a particular classifier
    - restrict the semantics of a particular class in the metamodel
- OCL constraints are:
  - declarative: they specify what must be true, not what must be done
  - have no side effects: evaluating an OCL expression does not change the model under study
  - have formal syntax and semantics: their interpretation is unambiguous

Types of Constraints

- OCL relies on the types (e.g. classes, datatypes, etc.) defined in a metamodel.
- The context of a constraint defines the link between an entity in the metamodel and an OCL expression.
- The contextual type is the type (e.g. a class, component, interface, data type...) of the object for which the expression will be evaluated:
  - when the context is a type, the context is the contextual type
  - when the context is an operation, attribute, or association end, the contextual type is the type for which the corresponding feature is defined
- Constraint expression:

  `context <contextual-type>`
  `<constraint-type>: <OCL expr>`
  - `<contextual-type>`: model element that is constrained
  - `<constraint-type>`:
    - `inv`: (invariant, contextual type is a classifier)
  - `<OCL-expression>`: boolean OCL expression
- OCL expressions that share the same context can be combined under the same `context` clause.

Example

- Given the following model

- Model the following constraints
  - A state cannot be initial and final in a state machine.
  - Direct loops are not allowed.

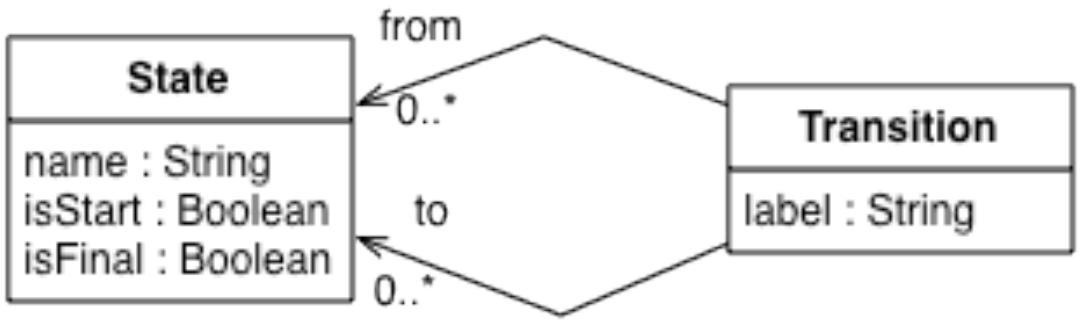
Collection Types

- When the multiplicity of an association end (reference) denotes more than one object, an object is linked to a collection of objects of the associated class
  - Traversing the association results in a collection of objects
- OCL has a number of collection operations to write expressions in such situations
  - Notation `c->f()`: apply the OCL operation f to the collection c as a whole (as opposed to each member of the collection)

Collection Types

- Set
  - In a set, each element may occur only once
  - When you navigate an association end marked unique (and not ordered)
- OrderedSet
  - In an ordered set, each element may occur only once and the elements are ordered
  - When you navigate an association end marked unique and ordered
- Bag
  - In a bag, elements may be present more than once
  - When you navigate an association end marked not unique and not ordered
- Sequence
  - Sequence is a bag in which elements are ordered
  - When you navigate an association end marked not unique and ordered

Examples



- A transition must have one source and at least one target state.
- All transitions with label ‘done’ lead to final states.

Concluding Remarks and Best Practices

- A DSL is a communication tool between developers and domain practitioners.
- The abstract syntax of a DSL captures the main constructs that can be used to specify knowledge in a domain.
- Design metamodels by considering the models that should be defined with it.
- DSL design is always iterative.