Association Types

Motivation, Specification and Implementation with the XModeler^{ML}©

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Ulrich Frank

Daniel Töpel

LE4MM

Motivation

- Growing need for the possibility to specify association types
  - Different kinds of associations show clear **semantic differences**
  - Improve comprehensibility by distinguishing different kinds of associations in the **concrete syntax** of a DSML
  - Association types may help with **model analysis** (vs. analyzing by name)
- Also: recurring pattern of dependencies between associations

Illustration

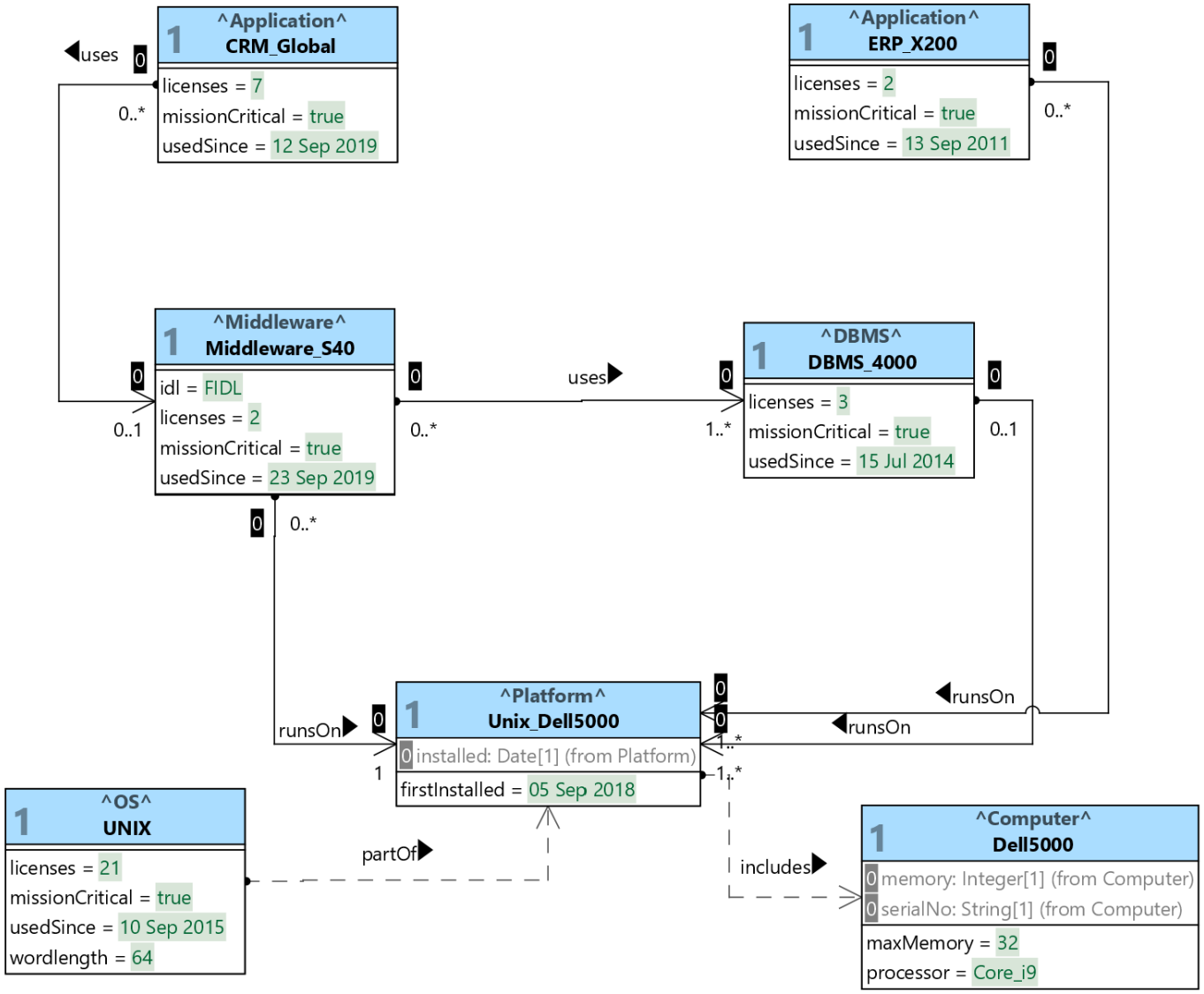
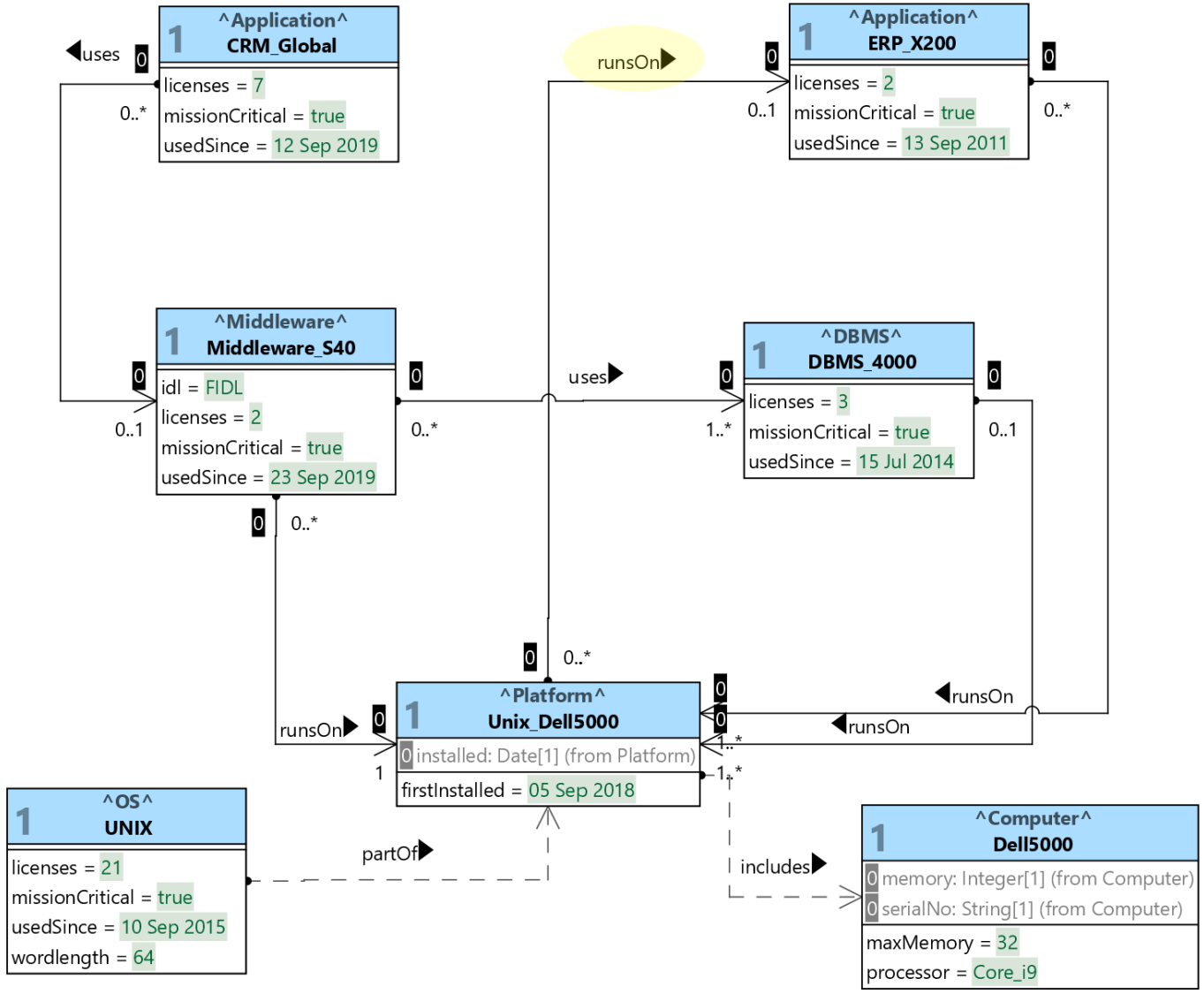


Illustration: Without AssociationType



Foundational Assumptions

- Intrinsic associations are not sufficient
- Association types as an abstraction over the properties shared by a set of similar associations
- hence, enable domain-specific association types
- For simplicity, (in contrast to UML), we focus on binary associations and stateless links.

How many (meta) levels of association types are needed?

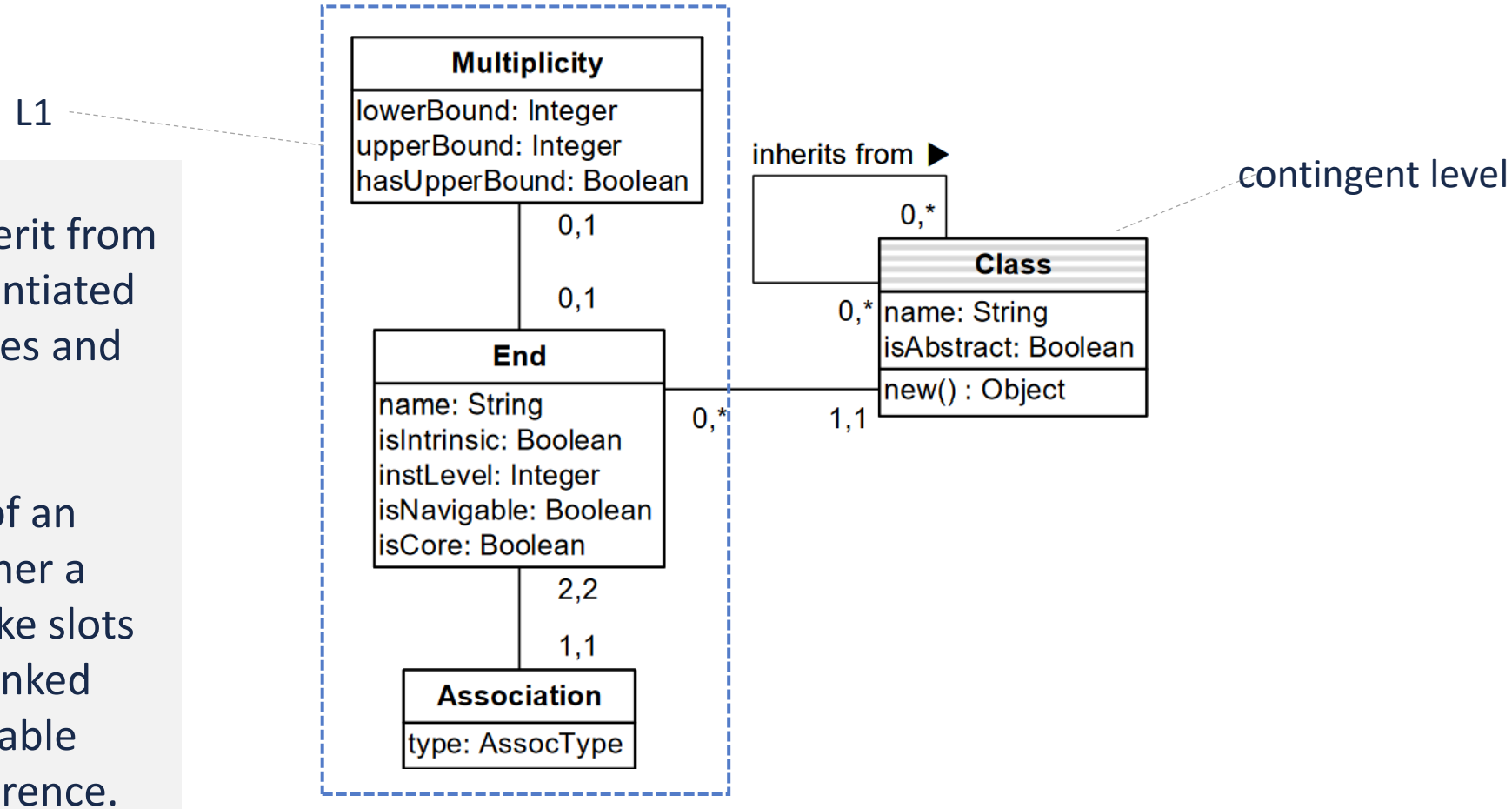
Selected Requirements

- **S2**: Restrict permissible set of classes
- **S3**: (Under-)specification of multiplicities should be possible
- **S4/5**: allow for restricting set of permissible associations by user-defined constraints that refer to the state of associated classes / linked objects
- **S6**: enable the definition of dependencies between associations

Previous Implementation of Associations

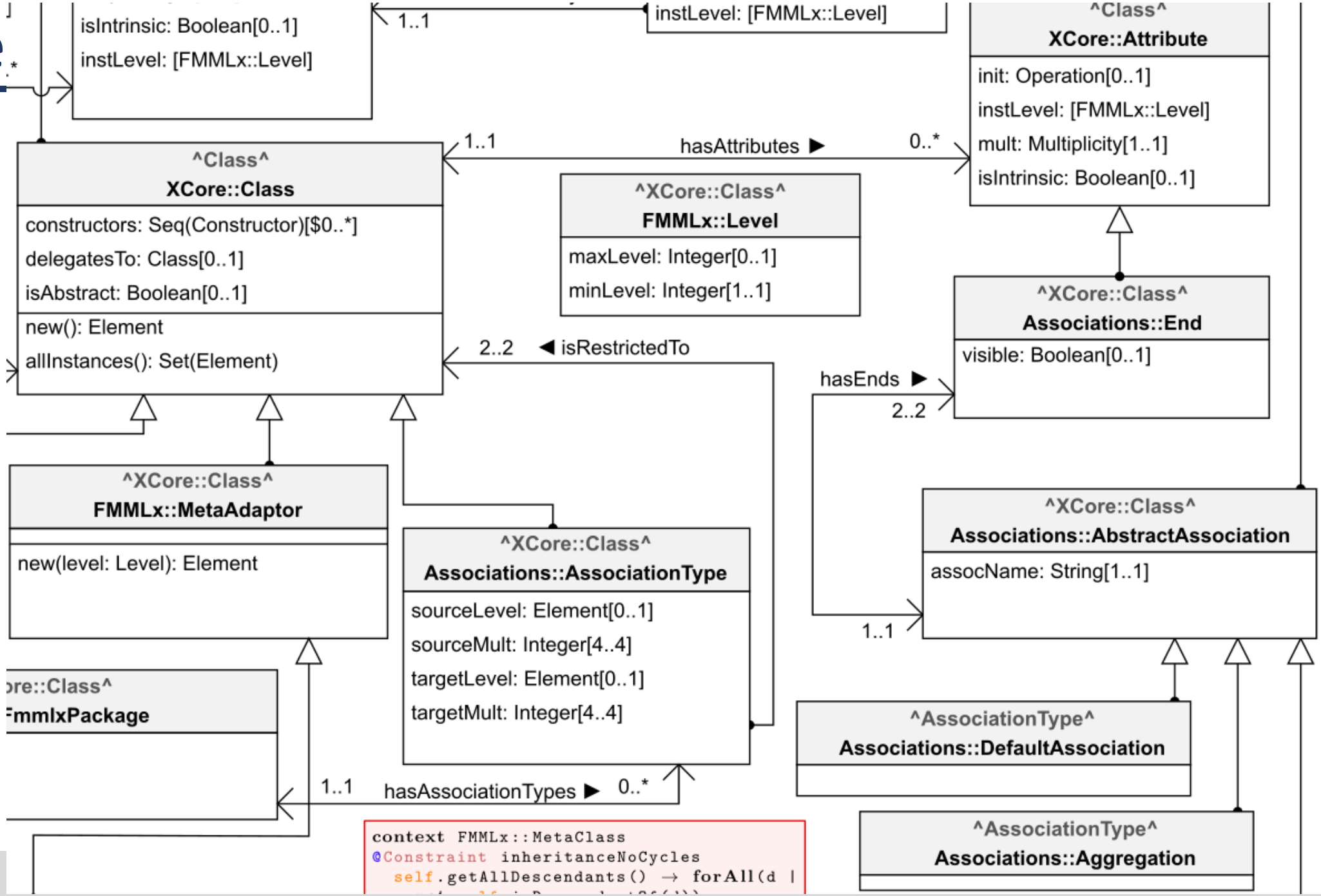
Association does not inherit from **Class**. Therefore the instantiated associations are not classes and not instantiable.

A **link** is not an instance of an association. There is neither a class Link. Links behave like slots with a **reference** on the linked object. For the non-navigable direction, there is no reference.



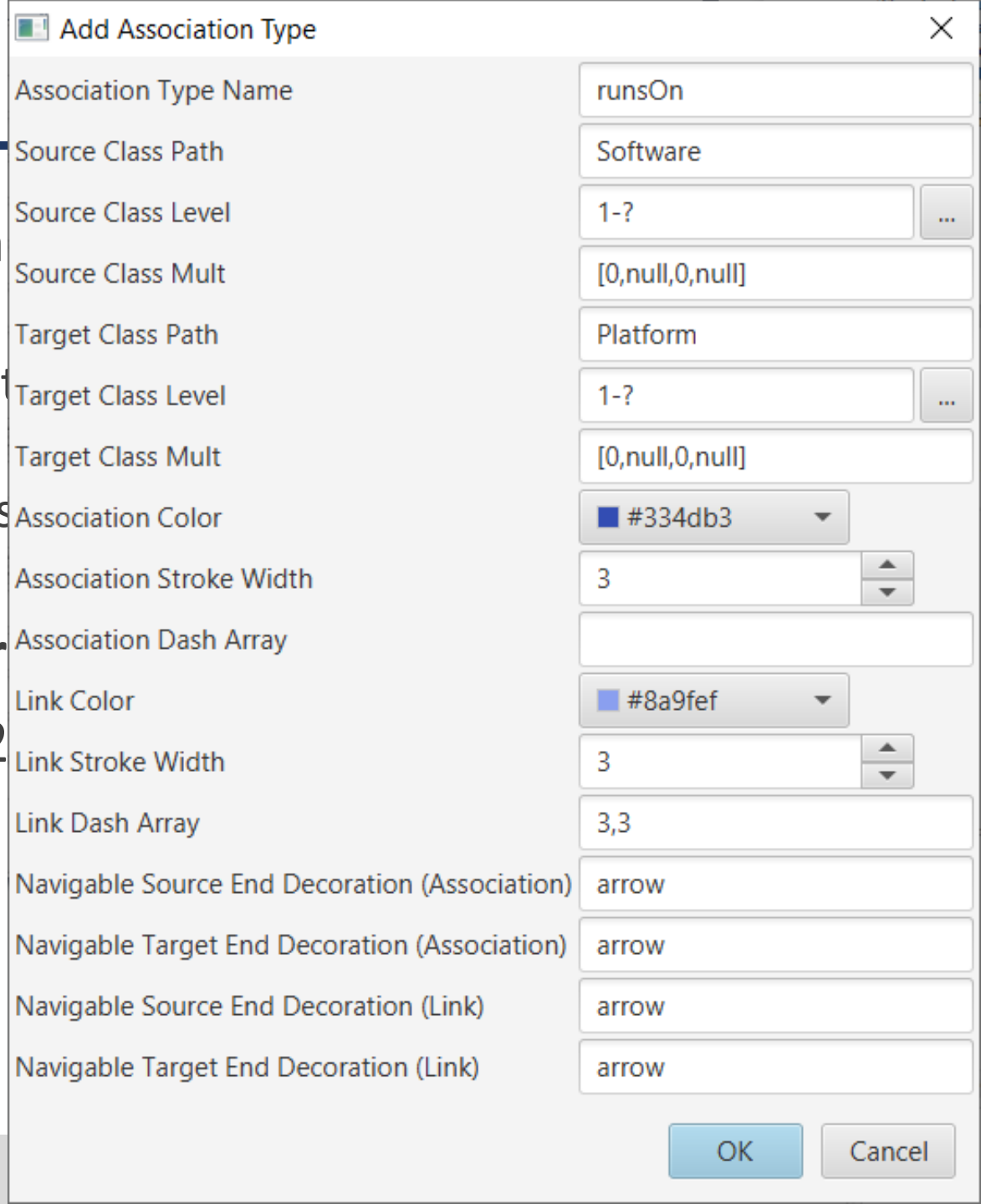
Design

- preliminary decision: at first, support for association types at AL3 only
  - AL3 (meta-level, the class AssociationType)
  - AL2 (association types)
  - AL1/AL0 unchanged
- Note that ALn corresponds to Ln-1
- However, convenient extension of further levels of association types should be possible (req. S7).
- Association levels do not necessarily imply instantiation
- Association levels independent from class levels
- The existing class Association (AL2) is migrated to new Association Type “DefaultAssociation” on AL2

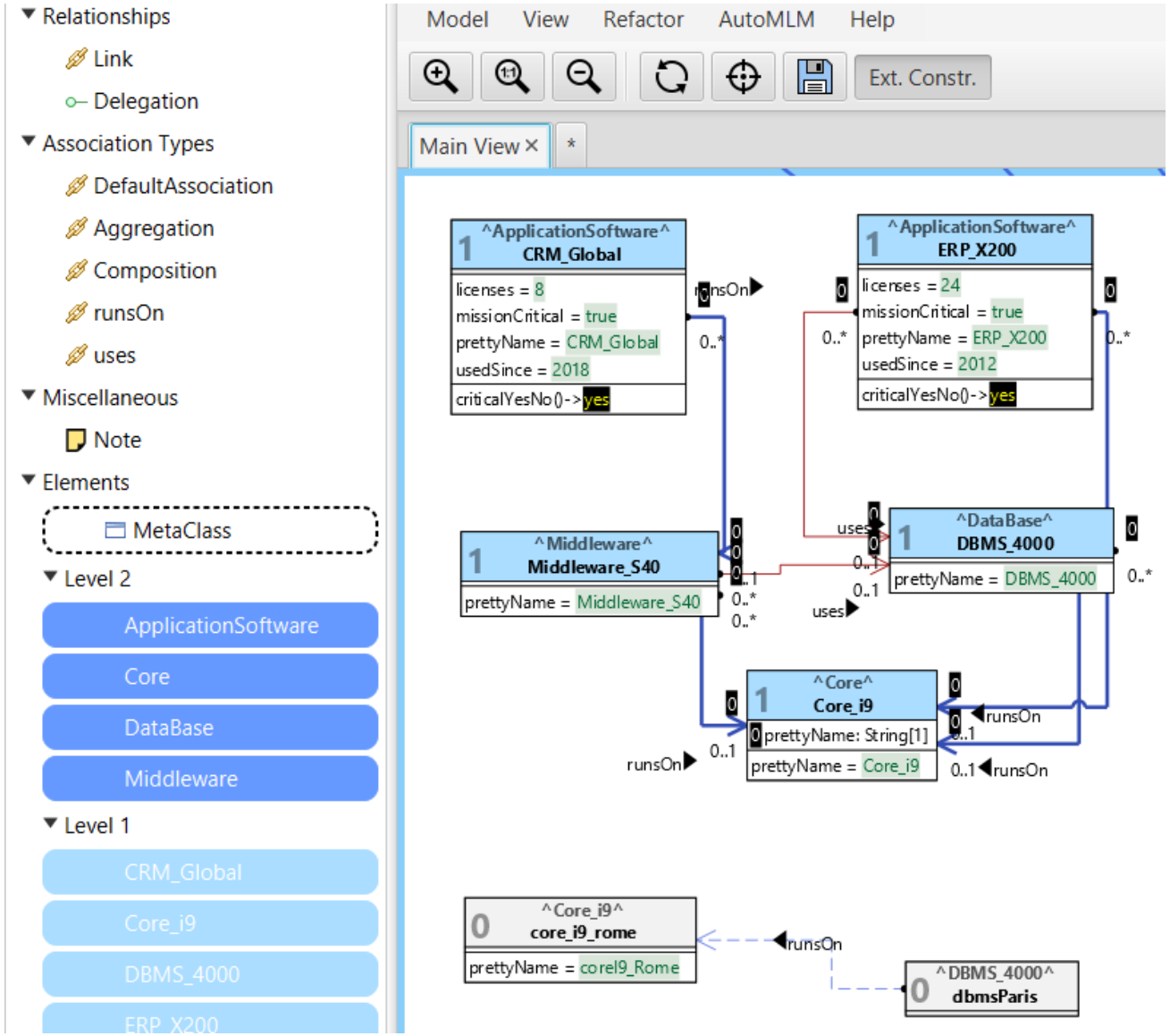

isIntrinsic: Boolean[0..1]
instLevel: [FMMLx::Level]
1..1
instLevel: [FMMLx::Level]
^Class^
XCore::Attribute
init: Operation[0..1]
instLevel: [FMMLx::Level]
mult: Multiplicity[1..1]
isIntrinsic: Boolean[0..1]
1..1
hasAttributes ▶
0..*
^Class^
XCore::Class
constructors: Seq(Constructor)[$0..*]
delegatesTo: Class[0..1]
isAbstract: Boolean[0..1]
new(): Element
allInstances(): Set(Element)
^XCore::Class^
FMMLx::Level
maxLevel: Integer[0..1]
minLevel: Integer[1..1]
^XCore::Class^
Associations::End
visible: Boolean[0..1]
2..2
◀ isRestrictedTo
hasEnds ▶
2..2
^XCore::Class^
FMMLx::MetaAdaptor
new(level: Level): Element
^XCore::Class^
Associations::AssociationType
sourceLevel: Element[0..1]
sourceMult: Integer[4..4]
targetLevel: Element[0..1]
targetMult: Integer[4..4]
^XCore::Class^
Associations::AbstractAssociation
assocName: String[1..1]
1..1
ore::Class^
FmmlxPackage
1..1
hasAssociationTypes ▶
0..*
^AssociationType^
Associations::DefaultAssociation
^AssociationType^
Associations::Aggregation
context FMMLx::MetaClass
@Constraint inheritanceNoCycles
self.getAllDescendants() → forAll(d |

Implementation

- New Association Types on AL2 restrict th
  according to requirements (S2/3)
  - Participating **classes** can be restricted to inst
    on a given **level**
  - **Cardinality** can be restricted by a 4-tuple, as
    be restricted individually
- Each restriction can be set to a **non-restr**
- A **concrete syntax** can be defined on AL2
  (associations) and AL0 (links)



Demo (introductory example with assoc types)



Conclusions

- The introduction of association types ultimately reflects the core idea of multi-level modeling, which is to promote reuse, integrity, and flexibility through additional abstraction.
- Current implementation provides a testbed for further investigating the utility of association types and for refining respective requirements.
- Implementation issues
  - currently specification of graphical notation rather restricted
  - also, definition of constraints could be in part more user-friendly
- Future Work:
  - If Association Types are defined, should it be prohibited to use default associations?
  - Change operations need to be added

Download of the XModeler^{ML} and Screencast

http://www.le4mm.org