Metamodels for Object Role Modeling

MSDL

Hongyan Song

March 2005

References

- [1] Chapter II - Two Meta-Models for Object-Role Modeling, Information Modeling Methods and Methodologies by John Krogstie, Terry Halpin and Keng Siau, Idea Group Publishing © 2005
- [2] Modeling Approaches, Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design by Terry Halpin, Morgan Kaufmann Publishers © 2001
- [3] Chapter II - Comparing Metamodels for ER, ORM and UML Data Models, Advanced Topics in Database Research, Volume 3 by Keng Siau (ed), Idea Group Publishing © 2004
- [4] Halpin, T.A. (1998). ORM/NIAM object-role modeling. In P. Bernus, K. Mertins, & G. Schmidt (Eds.), *Handbook on information systems architectures* (pp. 81-101). Berlin: Springer-Verlag. Available online at: http:// **www.orm.net/pdf/springer.pdf**

Outline

- Introduction of ORM
- Two metamodels of ORM
  - Model A purely written in ORM
  - Model B written in ORM reusing some UML constructs
- Transform metamodel in ORM to MOF

Why Metamodel ORM

- A better way than ER and UML for conceptual information analysis
- A possible standard for business rules expression within the MDA architecture, and for use in ontology standards
- Mature technology (30 years history), supported by commercial industry tools (such as Visio Enterpise Architect), and Required for Microsoft Architect Certificate test
- Support these initiatives and the interchange of ORM model data between different software tools

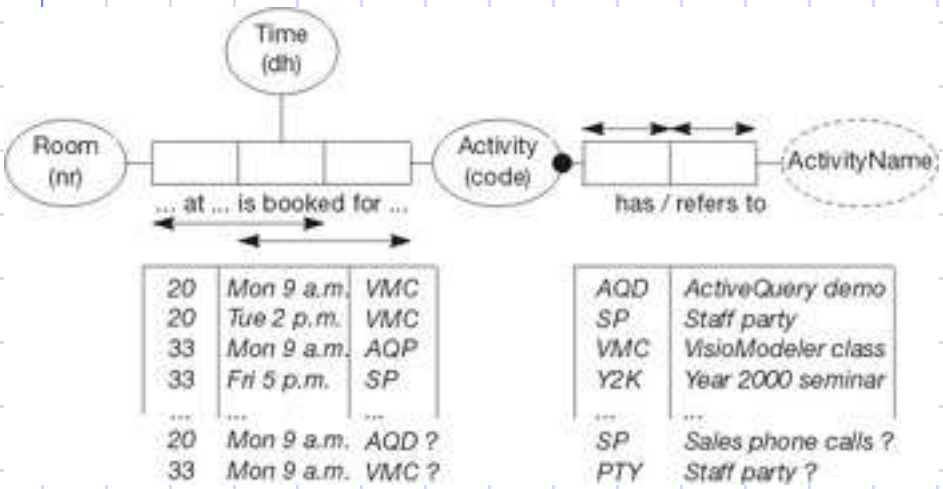
What is ORM

- Definition
  - A fact-oriented method for modeling an information system at the conceptual level
  - It pictures the world in terms of *objects*(entities or values) that play *roles*(parts in relationships)
- Working Principles
  - Whatever data use case, domain experts familiar with their meanings should be able to verbalize their information content in terms of natural-language sentences
  - Modelers transform that informal verbalization into a formal yet natural verbalization that is clearly understood by the domain expert
  - Sample data as fact instances are then abstracted to fact types. Constraints and perhaps derivation rules are then added and themselves validated by verbalization and sample fact populations

ORM Example

Room	Time	ActivityCode	ActivityName
20	Mon 9 a.m.	VMC	VisioModeler class
20	Tue 2 p.m.	VMC	VisioModeler class
33	Mon 9 a.m.	AQD	ActiveQuery demo
33	Fri 5 p.m.	SP	Staff party
...	...	...	...

(a) A Simple Room Scheduling Use Case

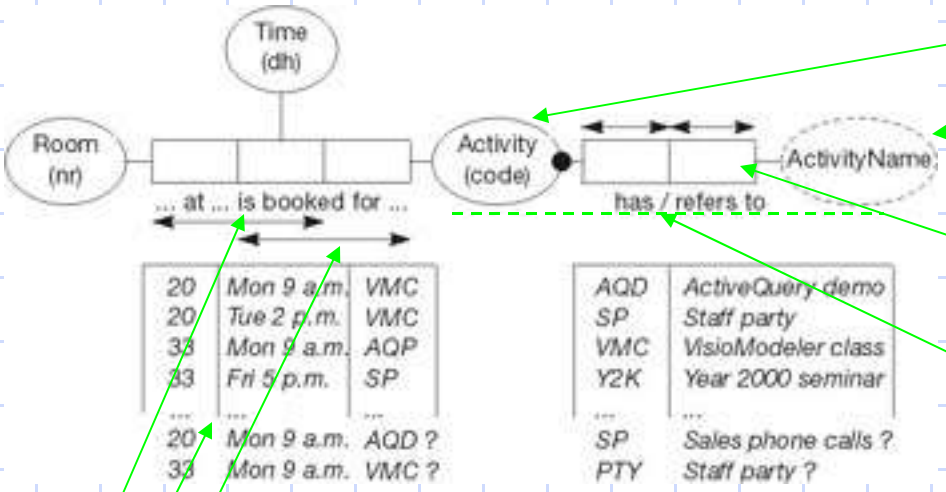


(b) A ORM Diagram for Room Scheduling

- The top row may be read by a domain expert as: **Room 20 at 9 a.m. Monday isbooked for the activity 'VMC' which has the name 'VisioModeler class'**
- A modeler may transform it into two elementary facts: **the Room numbered '20' at the Time with day-hour-code 'Mon 9 a.m.' is booked for the Activity coded 'VMC'; the Activity coded 'VMC' has the ActivityName 'VisioModeler class'**
- Once the domain expert agrees with this verbalization, *fact types* are abstracted from the fact instances, and an ORM diagram with population of sample data will be drawn

Figure from Ref[2]

Example Explanation



A ORM Diagram for Room Scheduling Figure from Ref[2]

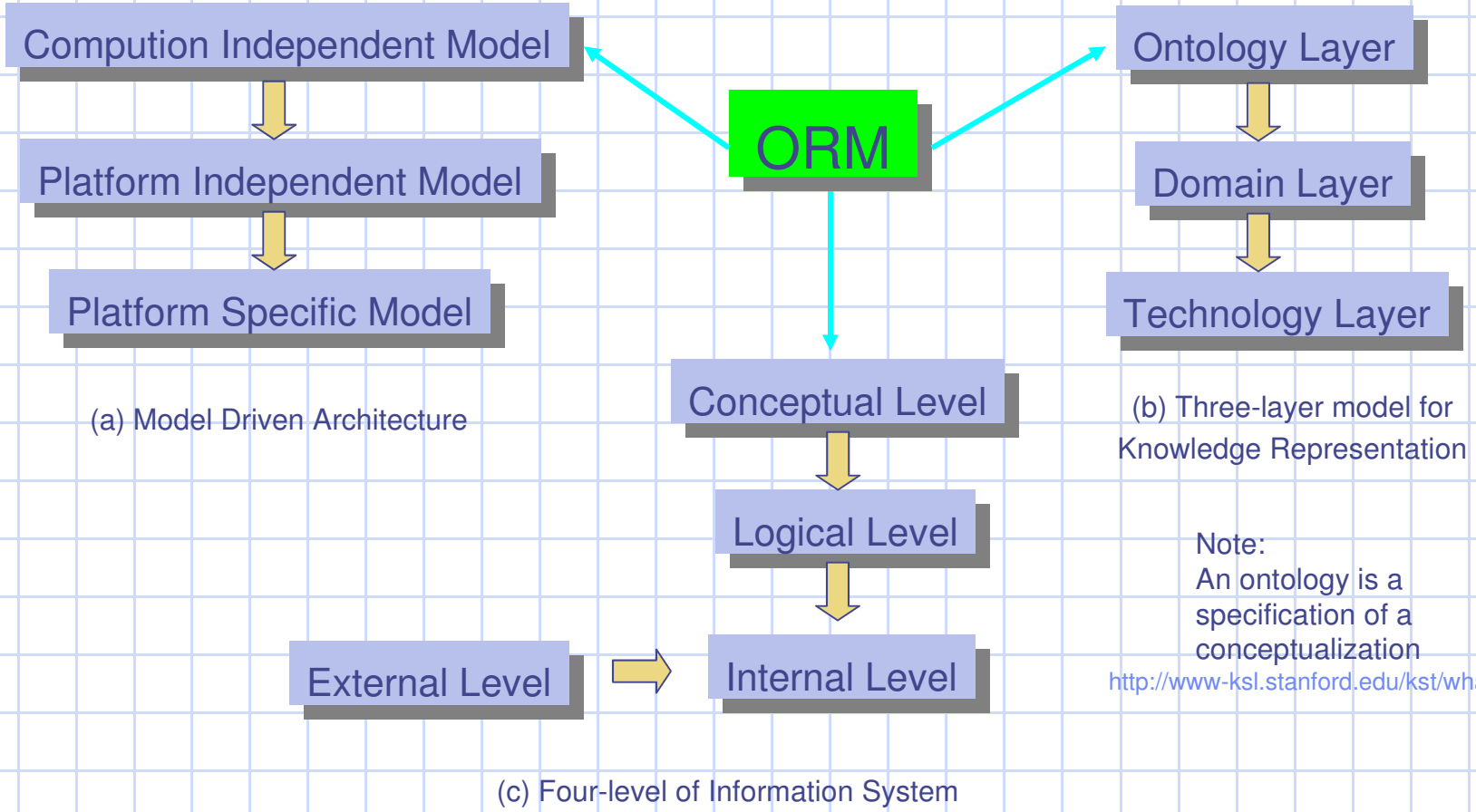
- Entity types are shown as named ellipses
- Value types are shown as named, dashed ellipses (e.g., ActivityName)
- A role is a part played in an association, and is depicted as a box
- An association is shown as a named sequence of one or more role boxes, each connected to the object type that plays it

- A predicate is a sentence with object holes in it
- A fact table may be added with a sample population to help validate the constraints
- The arrow-tipped bars are internal *uniqueness constraints*, indicating which roles or role combinations must have unique entry

Features

- Attributes free, all facts are represented in terms of objects (entities or values) playing roles
  - Leads to larger diagrams, has advantages including simplicity, stability, and ease of validation
- Associations of any arity, while ER only allows binary associations, and UML has no unary association
- Constraint primitives are orthogonal, and work properly with n-ary associations
- ORM schemas can be represented in either diagrammatic or textual form, and be easily understood and validated by domain experts
- ORM model can be transformed to ER or UML model manually or automatically

Where ORM fits in



(a) Model Driven Architecture

(b) Three-layer model for Knowledge Representation

(c) Four-level of Information System

Note:
An ontology is a specification of a conceptualization
http://www-ksl.stanford.edu/kst/what-is-an

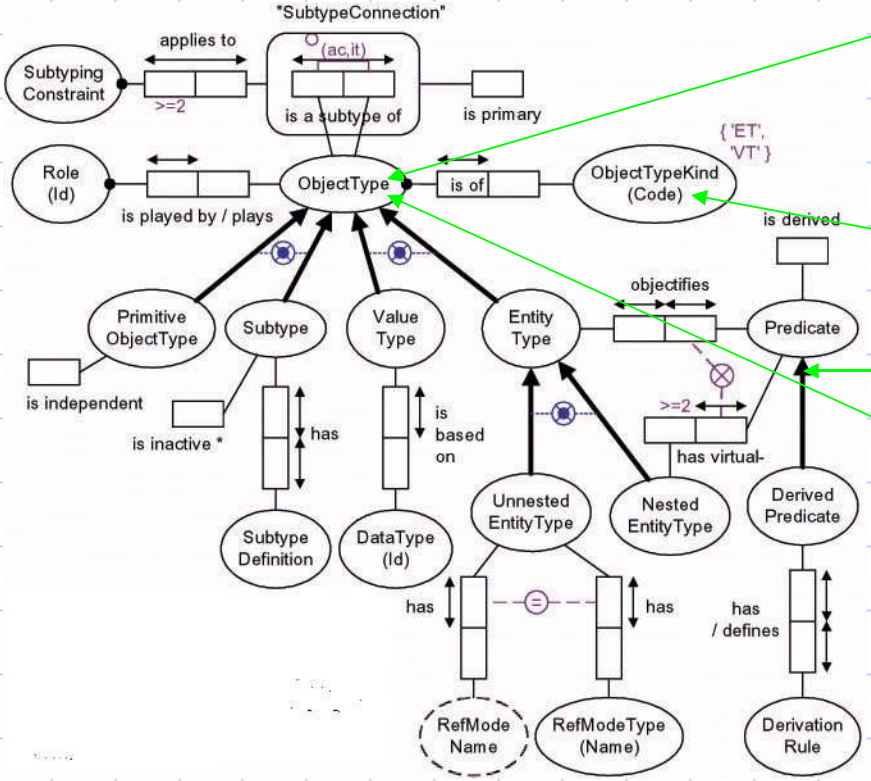
Seven Steps for Conceptual Modeling

1. Transform familiar information examples into elementary facts, and apply quality checks
2. Draw the fact types, and apply a population check
3. Check for entity types that should be combined, and note any arithmetic derivations
4. Add uniqueness constraints, and check arity of fact types
5. Add mandatory role constraints, and check for logical derivations
6. Add value, set comparison, and subtyping constraints
7. Add other constraints and perform final checks

Metamodeling ORM

- Orm components need to be meta-modeled:
  - Types, associations, instance data, and constraints
- Two Metamodels
  - A metamodel using ORM itself
  - B metamodel using ORM and UML

ORM Metamodel A – Main Types

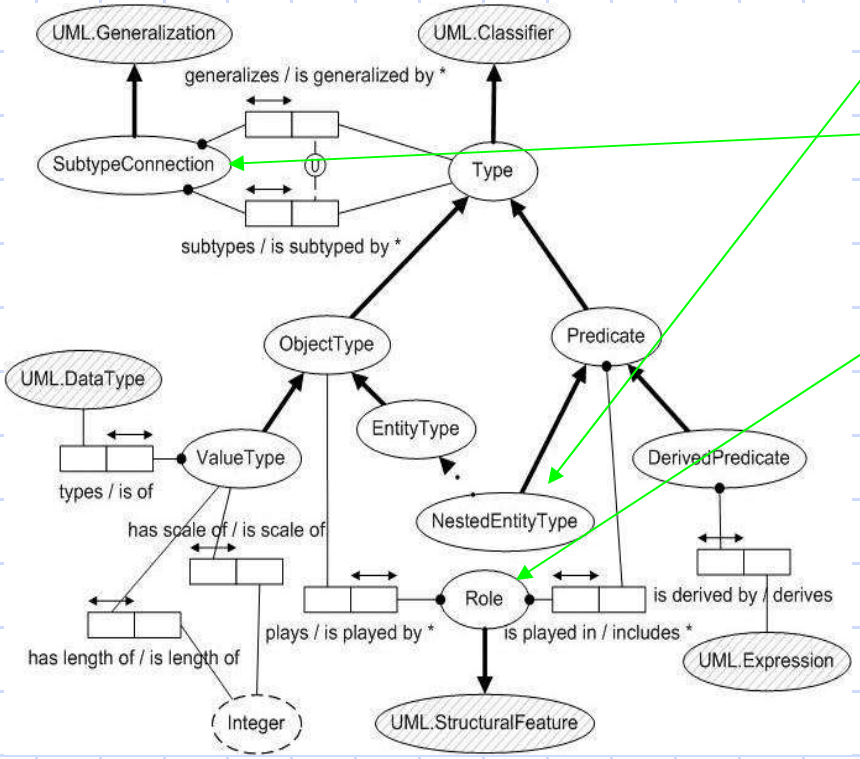


Main types in ORM meta-model A

- An object type is either an entity type or a value type
- Entity type is displayed as a solid named ellipse, and value type a dotted one
- A simple reference scheme may be abbreviated by reference mode in parentheses
- Subtyping relationships are depicted as solid arrows from subtype to supertype
- An object type either is *primitive* or is a *subtype*
- If subtypes collectively exhaust their supertype, this may be displayed as a circled dot. If subtypes are mutually exclusive, this may be displayed as a circled "X"
- An association may be *objectified*. For example, the meta-fact type `ObjectType is a subtype of ObjectType` is objectified as the entity type SubtypeConnection

Figure from Ref[1]

ORM Metamodel B – Main Types



Main types in ORM meta-model B

- NestedEntityType is treated as a subtype of both EntityType and FactType
- SubtypeConnection is inherited from Generalization in the UML meta-model
- Role is inherited from UML's StructuralFeature

Figure from Ref[1]

Metamodel A – Fact Type Reading

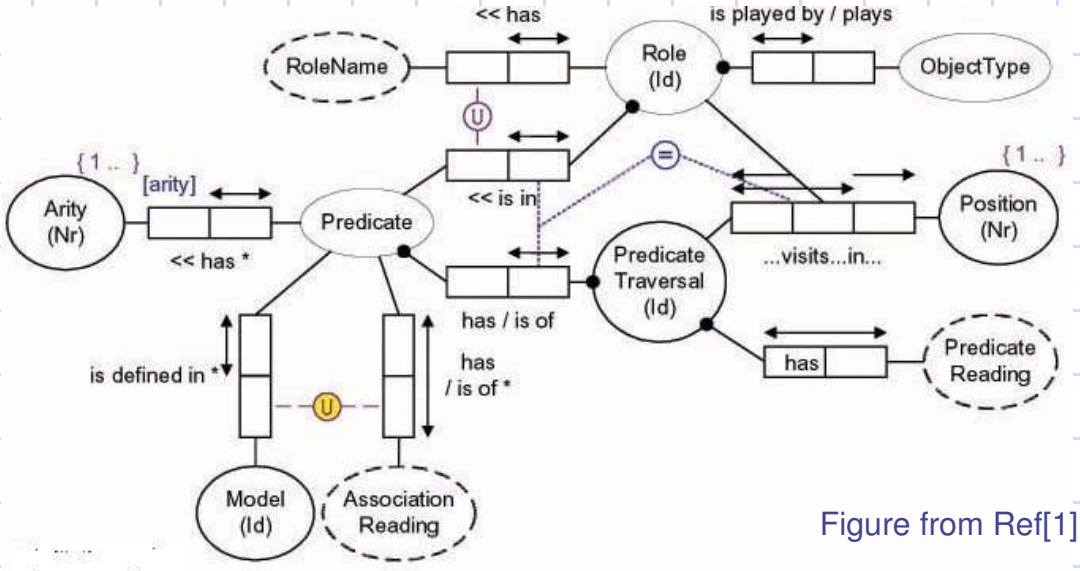
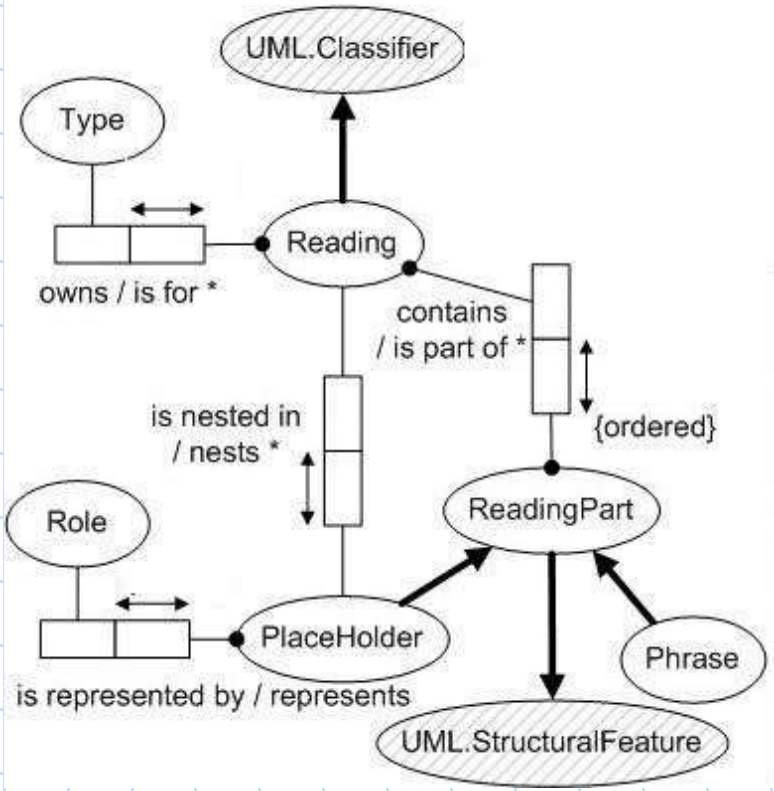


Figure from Ref[1]

Fact Type Reading in meta-model A

- A *role* is a part in an association and is depicted as a box
- The *arity* of an association is its number of roles
- An association is composed of a logical *predicate* and the object types that play the roles
- Predicates are displayed as sequences of role boxes, and have one or more *readings* depending on the order in which the roles are traversed.

Metamodel B – Fact Type Reading

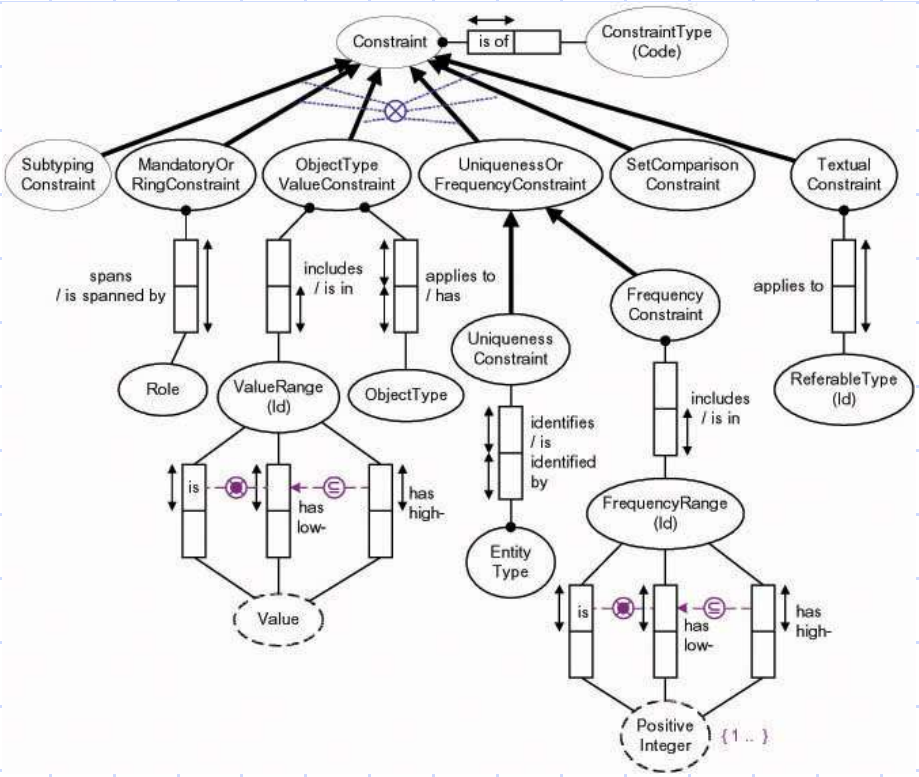


Fact Type reading in meta-model B

- A predicate reading is treated as a sequence of reading parts, each of which is either a placeholder "..." or a phrase

Figure from Ref[1]

Metamodel A - Constraints



ORM constraints in meta-model A

- Arrow-tipped bars over roles depict *internal uniqueness constraints*, indicating which roles or role combinations may have only unique entries.
- *External uniqueness constraints* apply to roles from different predicates and are depicted as a circled “u”
- A black dot depicts a *mandatory constraint*, indicating the role is mandatory for its object type
- A circled black dot is a disjunctive-mandatory (*inclusive-or*) constraint and applies to two or more roles

Figure from Ref[1]

Constraints – continued

- Possible values for an object type may be specified as a *value constraint* in braces
- A *frequency constraint* restricts the number of times an entry may appear in any given population of a role or role sequence
- *Set-comparison constraints* may apply between compatible role-sequences, to constrain their populations, and are of three kinds: *subset* (depicted as a circled "⊆"), *equality* (depicted as a circled "="), and *exclusion* (depicted as a circled "×")

Metamodel B - Constraints

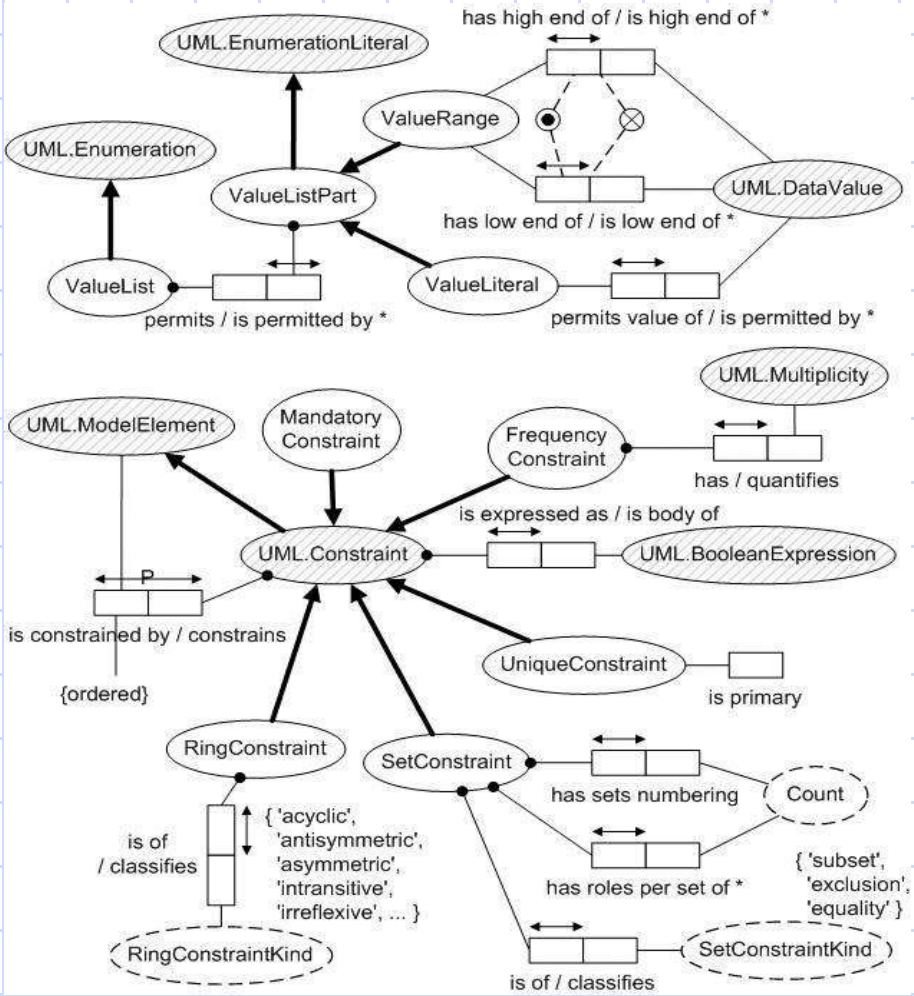
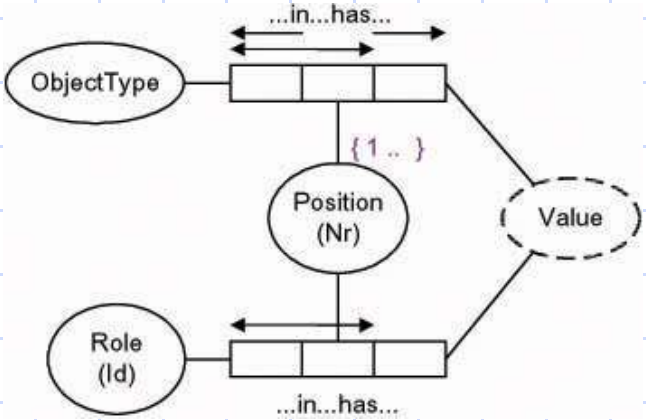


Figure from Ref[1]

ORM constraints in meta-model B

Metamodel A – Instance Data

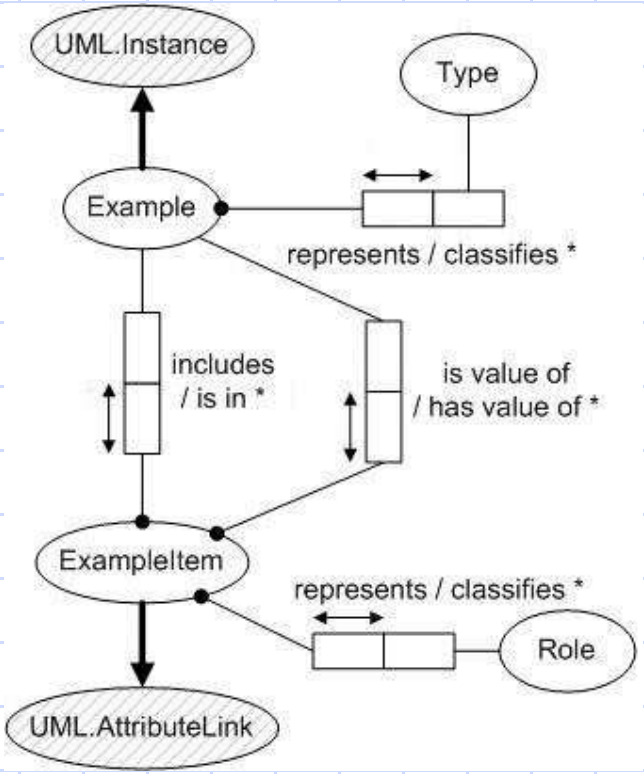


Modeling instances in meta-model A

- Sample populations may be provided for object types (see top ternary) and fact types (see bottom ternary)
- The position indicates a row number of the reference table or fact table

Figure from Ref[1]

Metamodel B – Instance Data



Modeling instances in meta-model B

Figure from Ref[1]

Metamodel - from ORM to MOF

- Transformations
  - Intersection classes and binary associations are generated for nested, ternary, and higher predicates
  - Binary predicates connecting entity types become associations connecting classes
  - Binary predicates connecting entity types with value types become simple attributes
  - Unary predicates become attributes of Boolean type

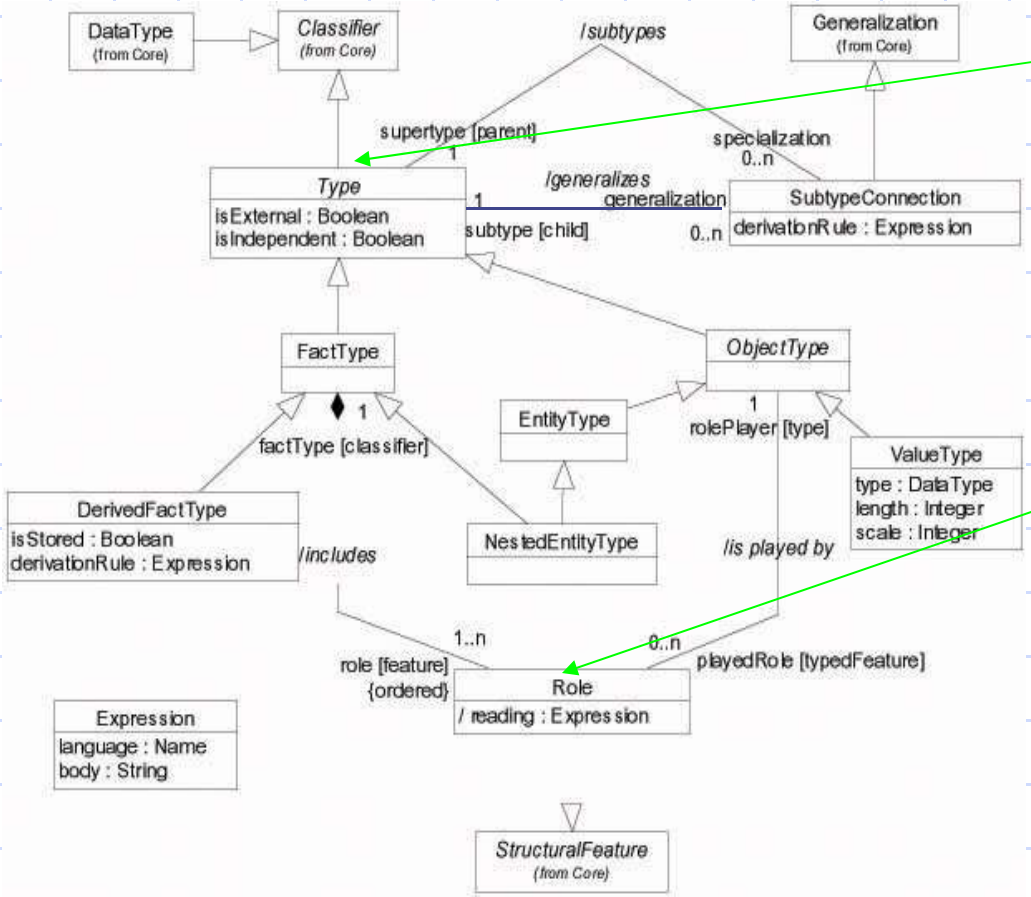
Mapping from ORM to MOF

ORM Constructs	MOF Constructs
Type	Classifier
Object Type	Classifier (through Type)
Entity Type	Classifier (through Object Type)
Value Type	Classifier (through Object Type)
Nested Entity	Classifier (through Entity Type and Fact Type)
Role	Structural Feature
Subtype Connection	Generalization
Reading	Classifier
Reading Part	Structural Feature
Phrase	Structural Feature (through Reading Part)
Place Holder	Structural Feature (through Reading Part)

Value List	Enumeration
Value List Part	Enumeration Literal
Value Literal	Enumeration Literal (through Value List Part)
Value Range	Enumeration Literal (through Value List Part)
Unique Constraint	Constraint
Mandatory Constraint	Constraint
Frequency Constraint	Constraint
Set Constraint	Constraint
Ring Constraint	Constraint
Example	Instance
Example Item	Attribute Link

Table 1: Mappings of ORM meta-classes to MOF meta-classes

Metamodel in MOF – Types

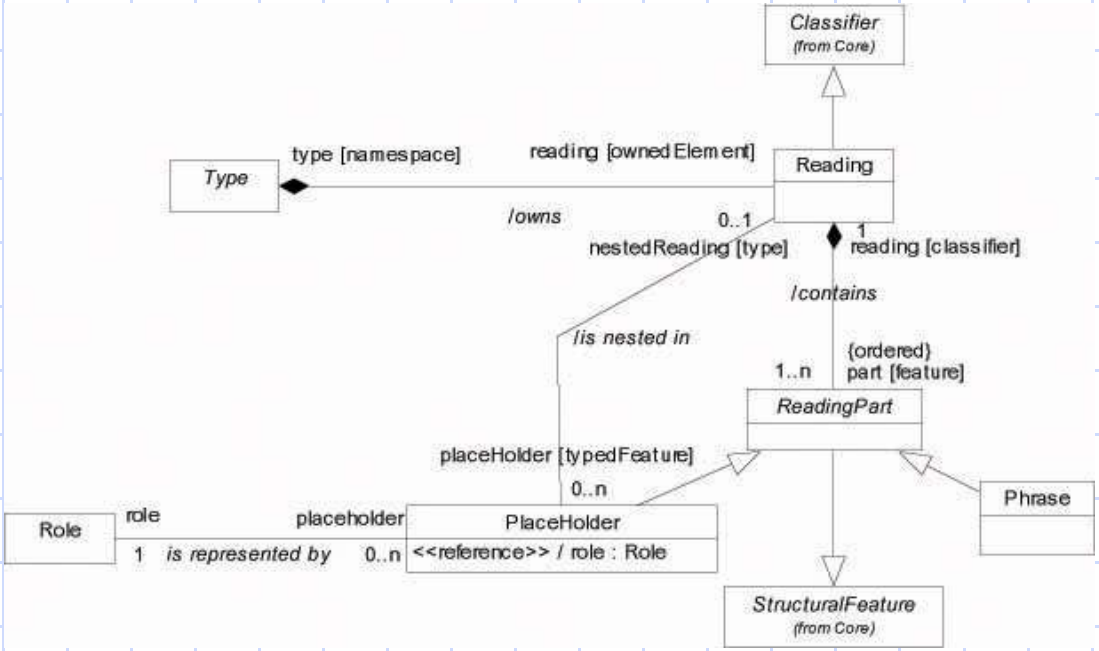


- The abstract meta-class "Type" was generated, from which both "Object Type" and "Fact Type" inherit
- The "Type" meta-class inherits from the UML "Classifier"
- "Role" inherits from the UML Structural Feature

Object types and Fact types

Figure from Ref[1]

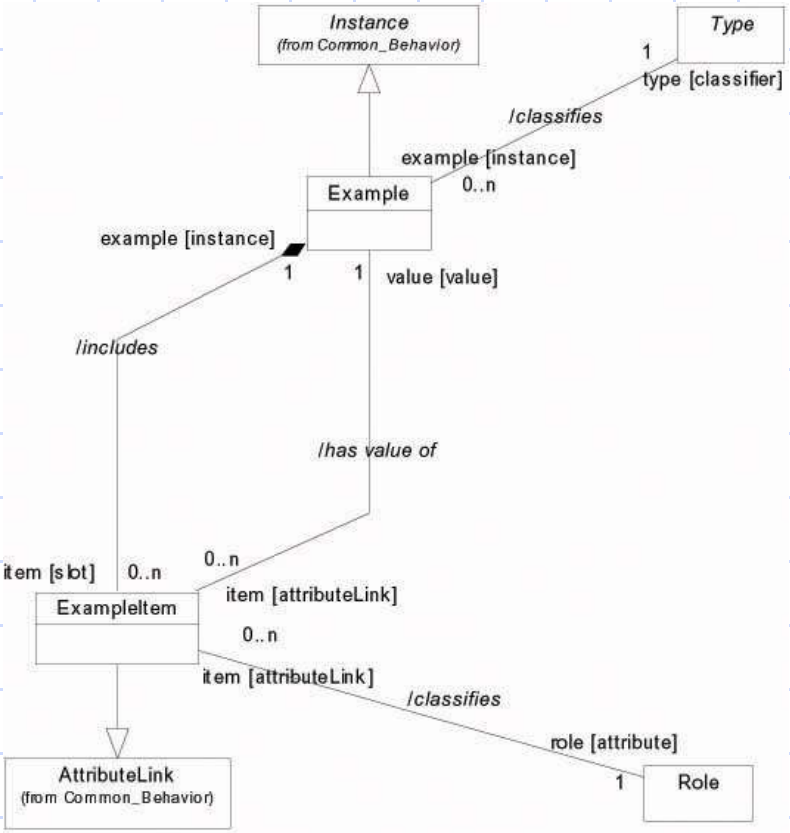
Metamodel in MOF – Fact Type Reading



Fact Type Reading

Figure from Ref[1]

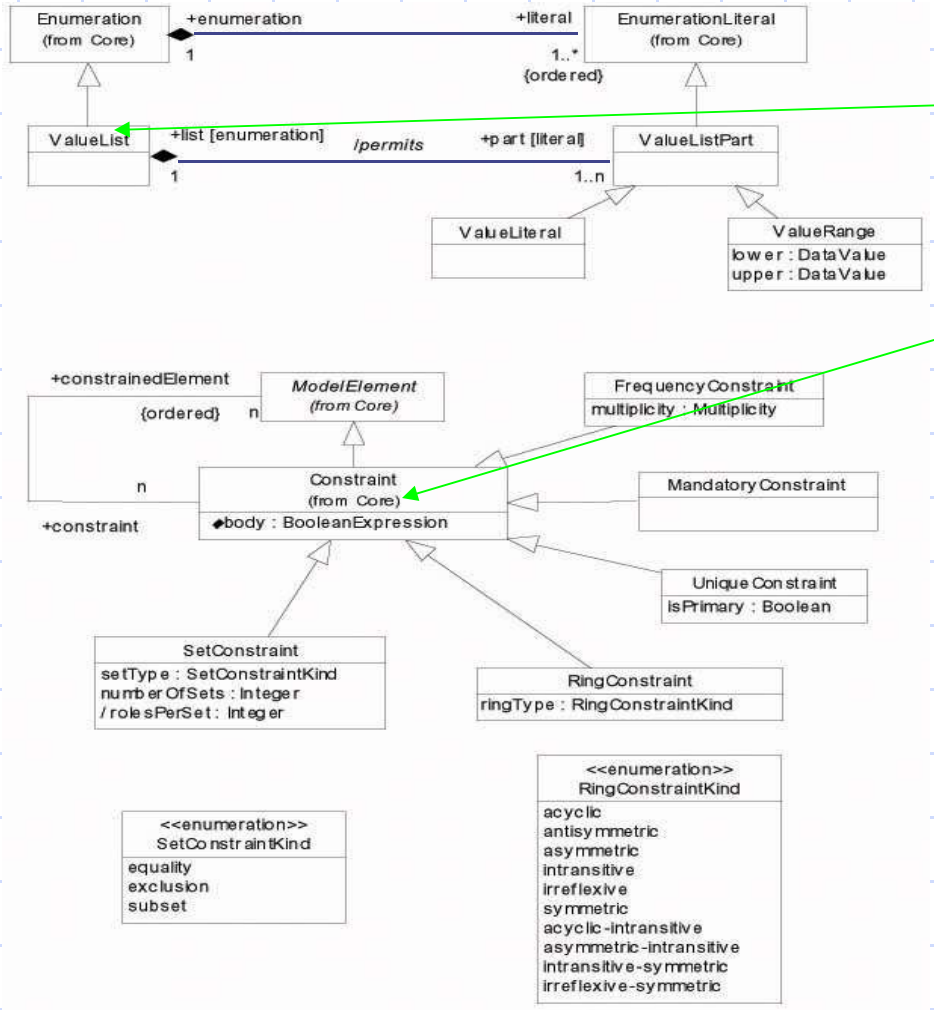
Metamodel in MOF – Instance Data



Sample Populations

Figure from Ref[1]

Metamodel in MOF - Constraints



- With the exception of "Value List," all ORM constraints inherit from the UML "Constraint"

Figure from Ref[1]

Constrants

Summary

- Introduction of ORM
  - What's ORM
  - An ORM example
  - Seven steps for Conceptual Schema Design Procedure (CSDP)
- Two metamodels of ORM
  - Metamodel purely written in ORM
  - Metamodel written in ORM and UML
- Transform metamodel expressed in ORM to MOF