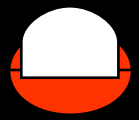
Zawansowane Modelowanie i Analiza Systemów Informatycznych

(l-7)



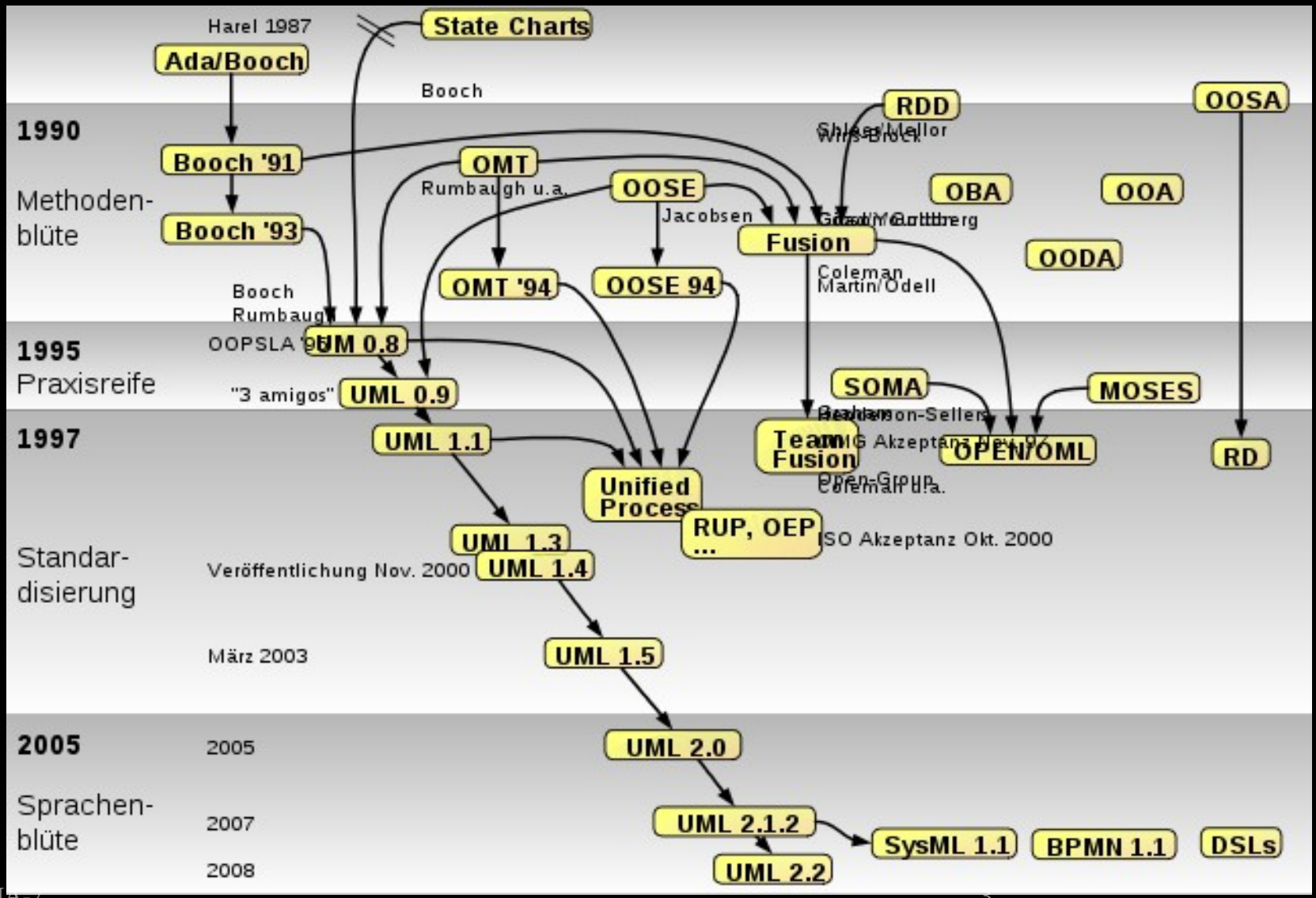
Polsko-Japońska Wyższa Szkoła Technik Komputerowych
Katedra Systemów Informacyjnych
2013

UML vs ORM

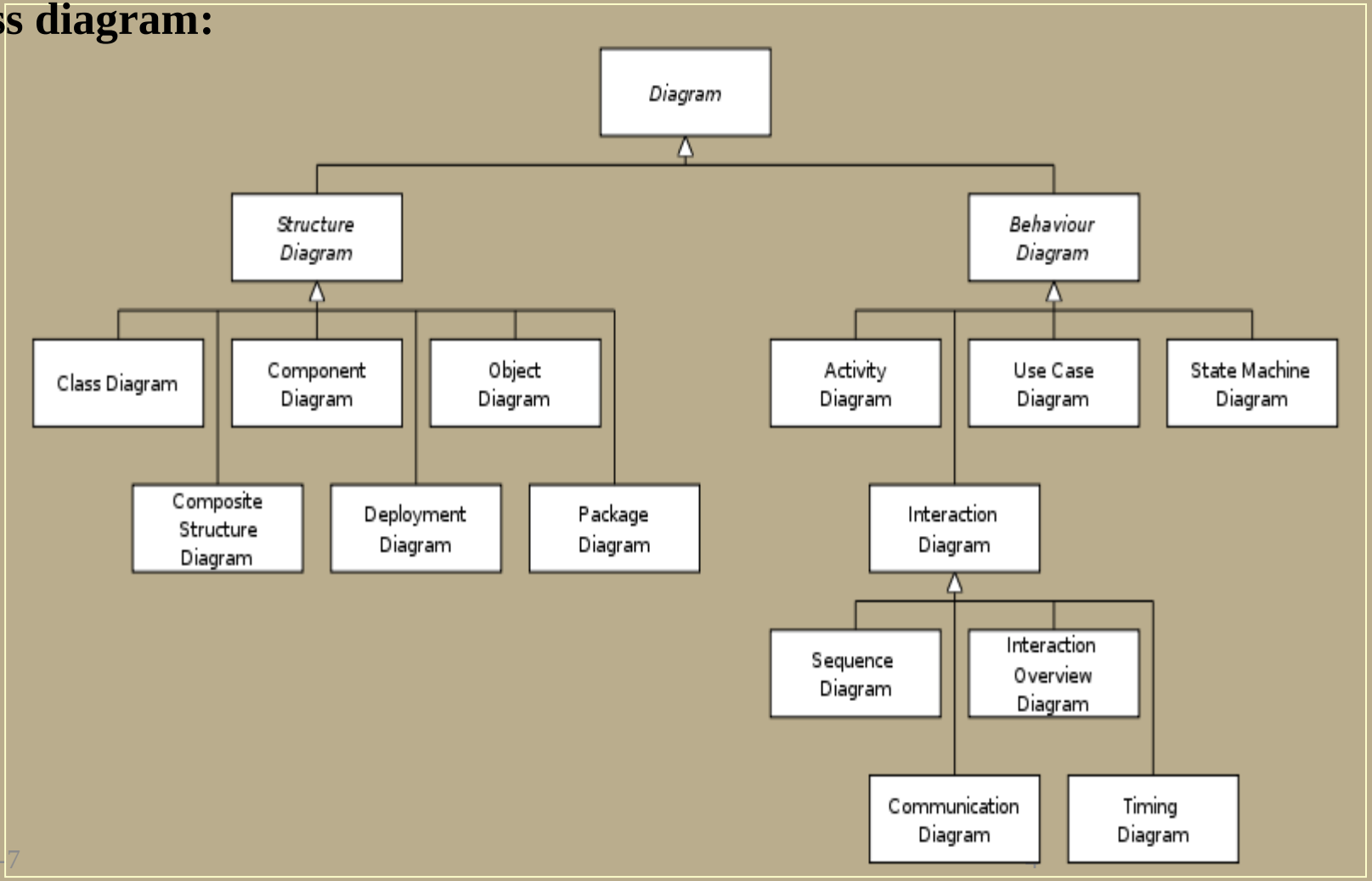
- The goal of this lecture is to identify only the baisc diffrences of the two languages in terms of modeling expressivness of the static data aspects.

- We will illustrate the selected aspects of the models by examples - easy to be generalised for further considerations.

http://en.wikipedia.org/wiki/Unified_Modeling_Language#History



UML 2.0 has 13 types of diagrams divided into three categories: Six diagram types represent *static* application structure, three represent general types of *behavior*, and four represent different aspects of *interactions*. These diagrams can be categorized hierarchically as shown in the following Class diagram:



UML class diagram

- Most popular UML diagram type
- Includes useful OO implementation features
- (e.g. attribute visibility, association navigability).
- But inferior to other approaches for conceptual data modeling
- (e.g. ORM or ER)
- UML graphical primitives are far less expressive than ORM's
- ORM constraints are orthogonal and unambiguous. **Orthogonality** allows use of an expression wherever its meaning or value can be used. ORM constructs were designed from the ground to be orthogonal. For example ORM constraints can be used and combined whenever this is meaningful. This is not true of languages like UML.

Comparing any two methods

The following criteria provide a useful basis for evaluating conceptual modeling methods:

- Expressibility
- Clarity
- Semantic stability
- Semantic relevance
- Validation mechanisms
- Abstraction mechanisms
- Formal foundation

The *expressibility of a language is a measure of what it can be used to say. Ideally, a* conceptual language

- The ***expressibility of a language*** *is a measure of what it can be used to say. Ideally, a* conceptual language should be able to completely model all details about the application domain that are conceptually relevant. This is called *the 100% Principle*.
- The ***clarity of a language*** *is a measure of how easy it is to understand and use.*
- ***Semantic stability*** *is a measure of how well models or queries expressed in the language retain their* original intent in the face of changes to the application.
- ***Semantic relevance*** *requires that only conceptually relevant details need be modeled. Any aspect* irrelevant to the meaning (e.g. implementation choices, machine efficiency) should be avoided. This is called the conceptualization principle
- ***Validation mechanisms*** *are ways in which domain experts can check whether the model* matches the application. For example, static features of a model may be checked by verbalization and multiple instantiation, and dynamic features may be checked by simulation.

- ***Abstraction mechanisms*** *allow unwanted details to be removed from immediate consideration* . This is very important with large models.
- ***A formal foundation*** is needed to ensure unambiguity and executability (e.g. to automate the storage, verification, transformation and simulation of models).

Recommended reading;

http://www.orm.net/pdf/orm-emm98.pdf

„A comparison of UML and ORM for data modeling”

by Dr. Terry Halpin

Director of Database Strategy, Visio Corporation

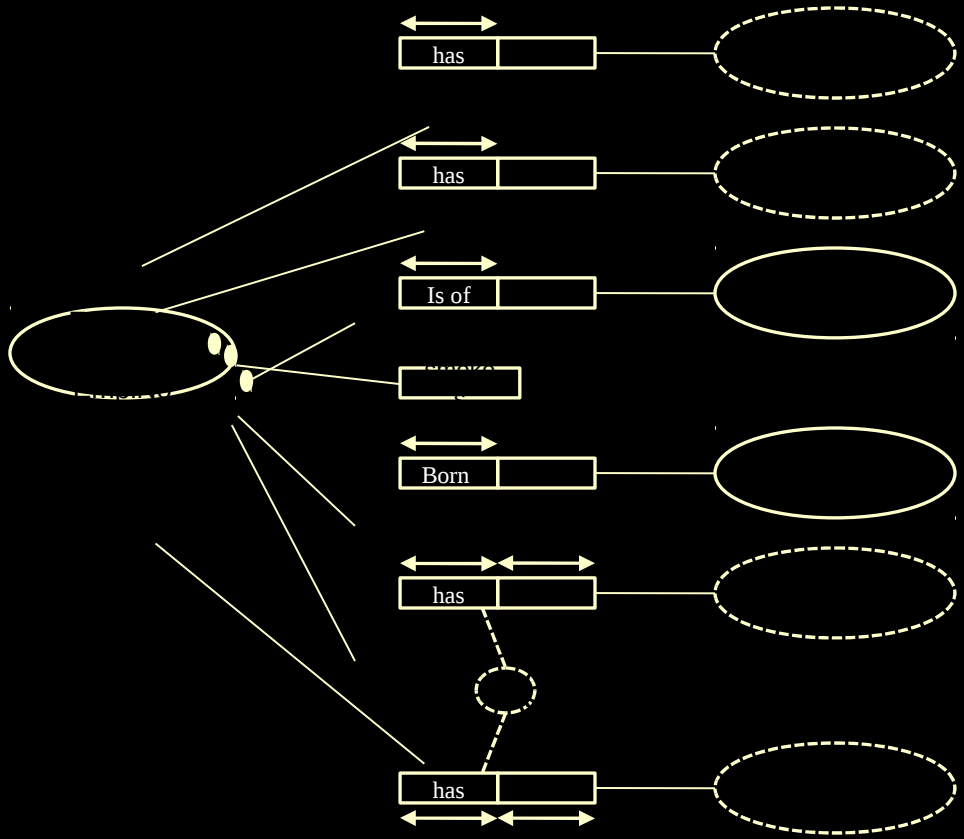
and Dr. Anthony Bloesch

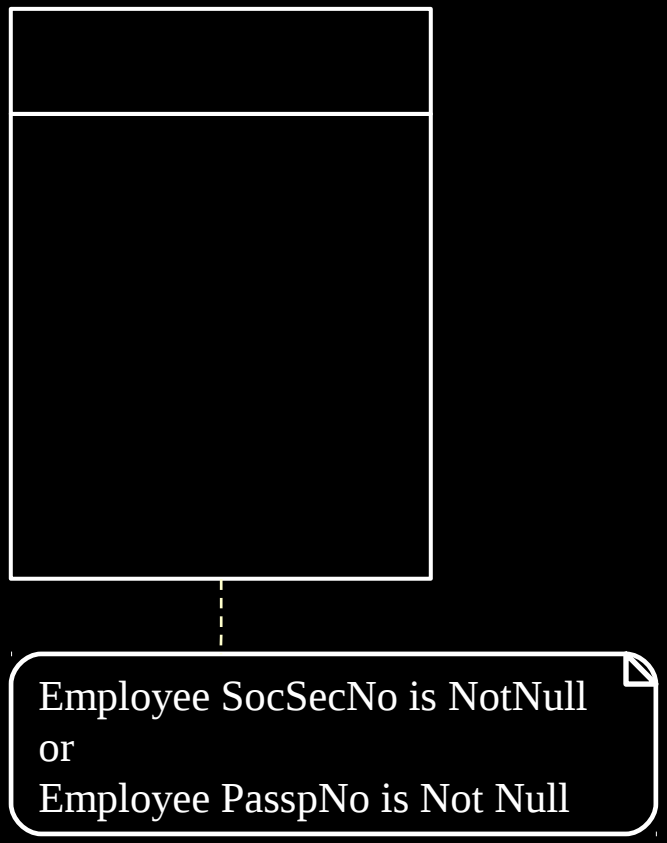
Director of Database Software Modeling, Visio Corporation

Some observations

- UML allows relationships to be modeled as attributes.
- ORM models the world in just objects and roles. Only one data structure – the relationship type is needed. Wherever an attribute is used in UML, ORM uses a realtionship instead.
- In UML attributes are mandatory by default.
- UML does not support unary relationships.
- UML does not have a graphic notation for disjunctive mandatory roles – textual expression – informal.
- UML does not have a standard graphica notation for the attribute uniqueness constraints.
- Many more differences can be identified.

Example – single valued attributes

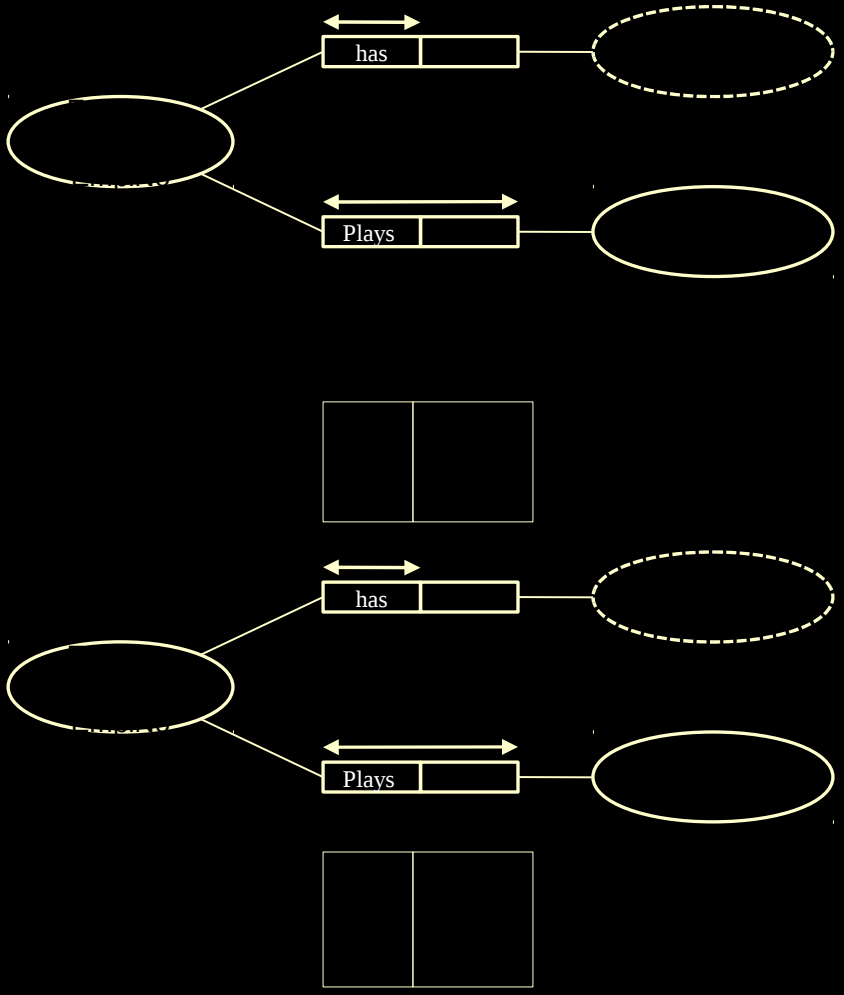




Benefits of Attributes-free Models

- Attributes –free models are;
  - More stable,
  - Easy to populate with multiple instacnes,
  - Facilites verbalization in sentences,
  - Highlight connectedness through semantic domains,
  - Are simpler and more uniform,
  - Make it easier to specify contraints,
  - Avoid arbitrary modeling decisions,

Example – multivalued attributes



EmplNo (P) EmplName **Sports** [0..*]

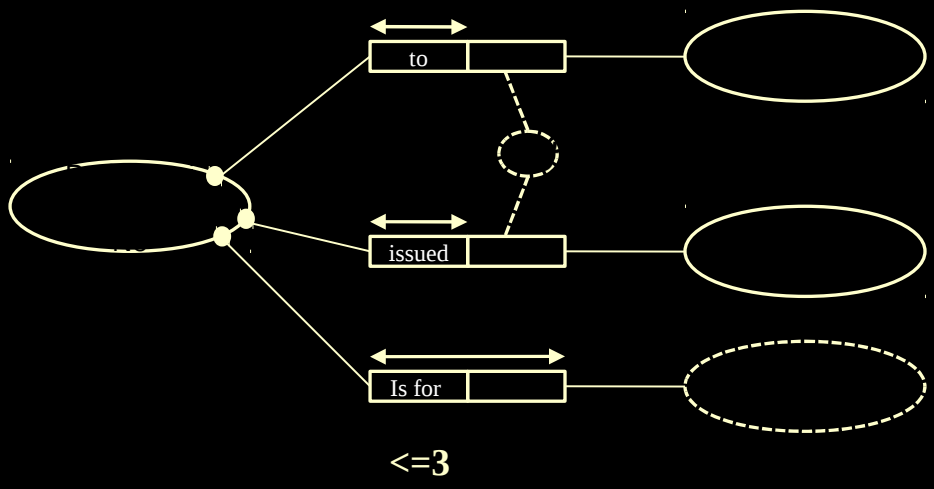
101: Employee
EmplNo = 101 EmplName = ‘Smith J’ Sports = null

102: Employee
EmplNo = 102 EmplName = ‘Jones P’ **Sports** = (‘judo’, ‘soccer’)

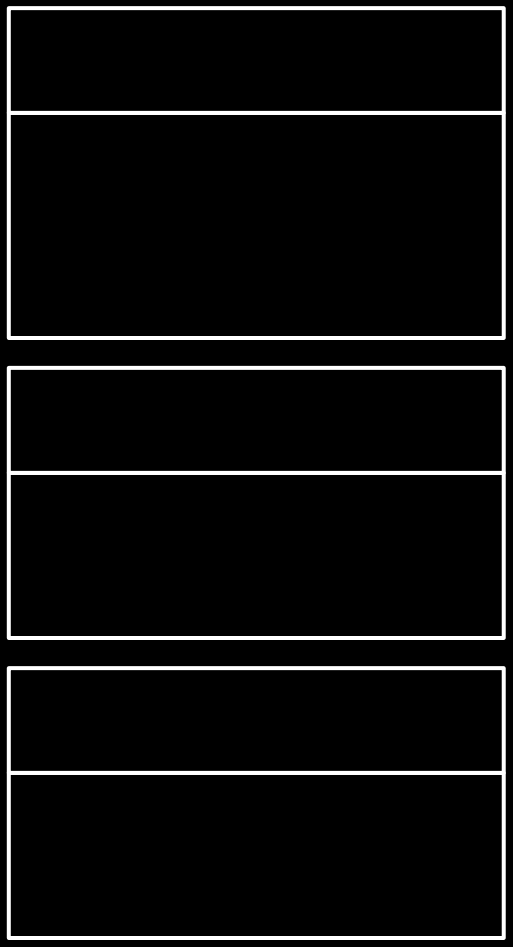
103: Employee
EmplNo = 103 EmplName = ‘Smith P’ Sports = (‘judo’, ‘netball’)

- UML standrad uses a null value for optional ‘sports’ in the previouse example – this is an implementation isssue not conceptual one!!!
- Instead of using fact tables for instantiation UML provides object diagrams essentially class diagrams where each object is shown as a separate class instance with data values supplied for its attributes.

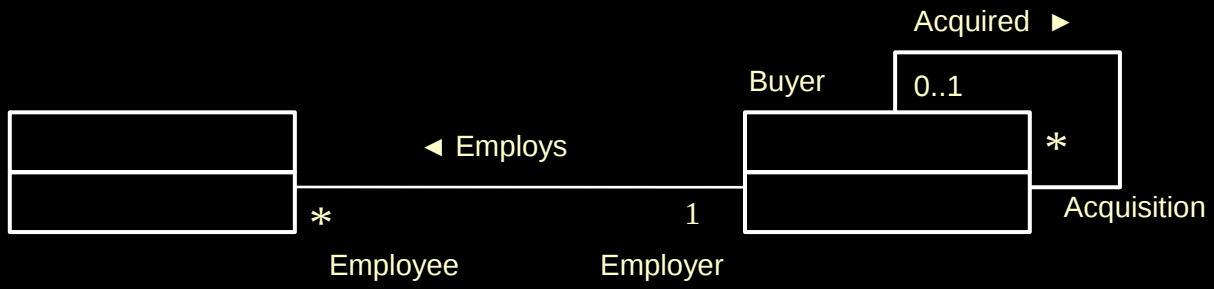
Let compare the two models of the same UoD

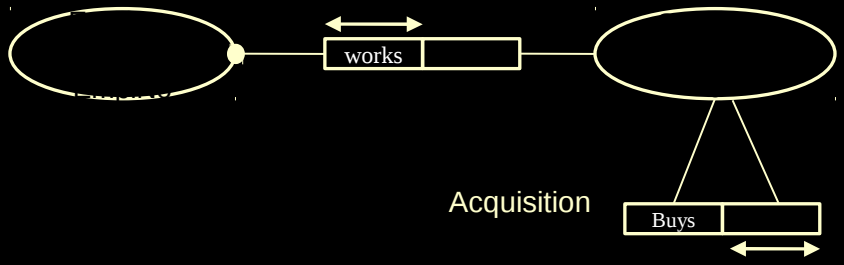


<=3

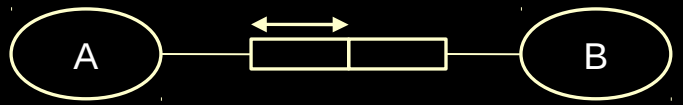
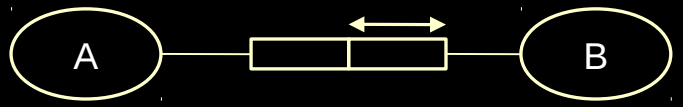
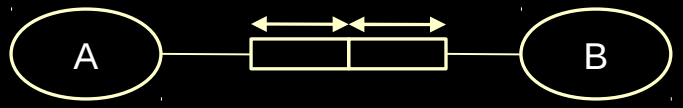
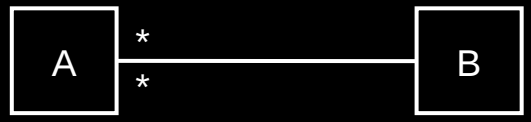
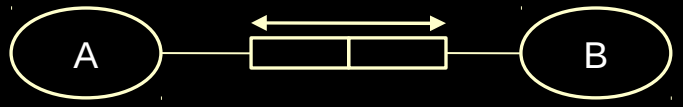
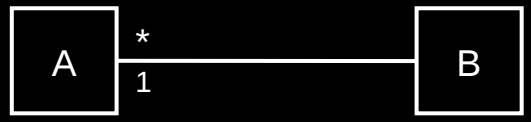
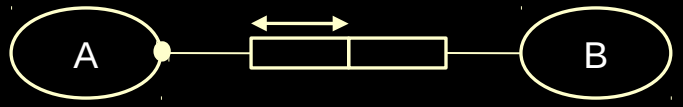
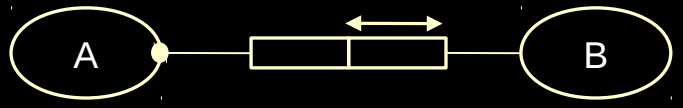
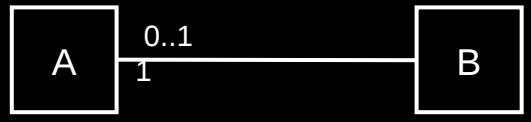
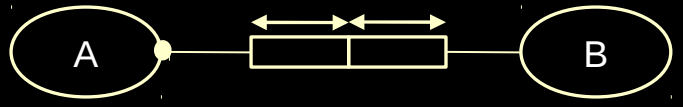
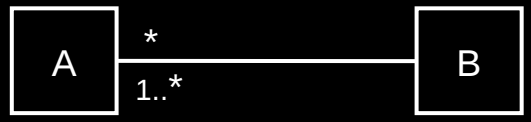
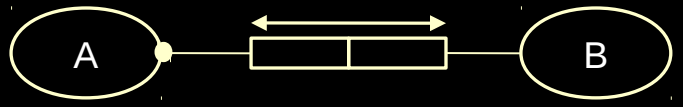


Associations





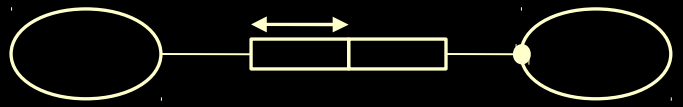
Equivalent Constraints patterns

	UML	ORM
N:1 Both roles optional		
1:N Both roles optional		
1:1 Both roles optional		
m:n Both roles optional		
N:1 First role mandatory		
1:N First role mandatory		
1:1 First role mandatory		
M:N First role mandatory		

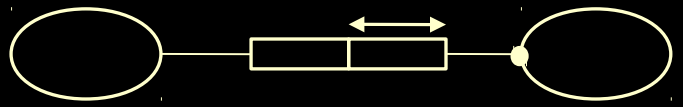
UML

ORM

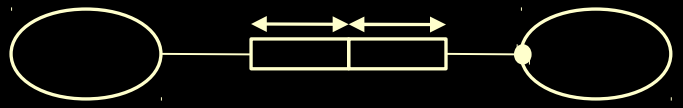
N:1
Second role mandatory



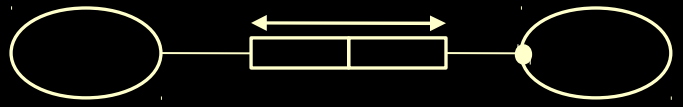
1:N
Second role mandatory



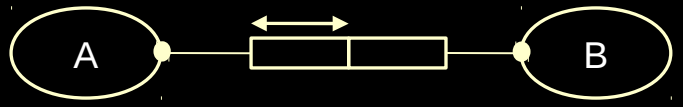
1:1
Second role mandatory



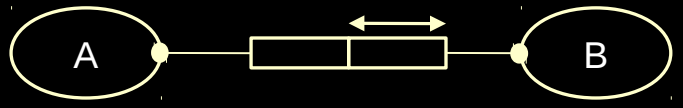
m:n
Second role mandatory



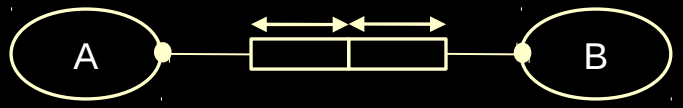
N:1
Both roles mandatory



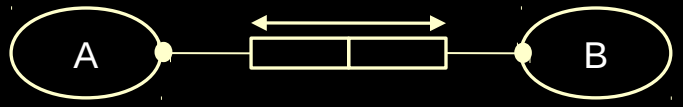
1:N
Both roles mandatory



1:1
Both roles mandatory



M:N
Both roles mandatory



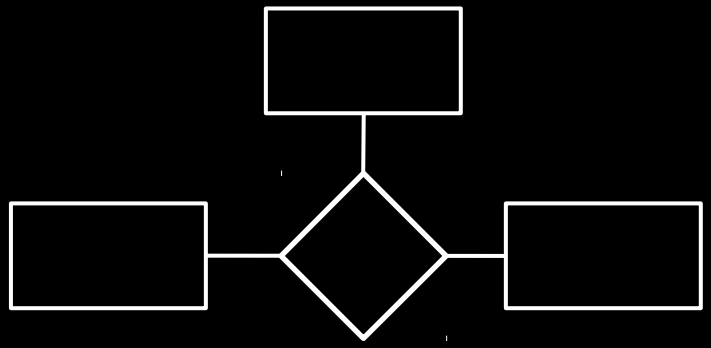
N-ary associations

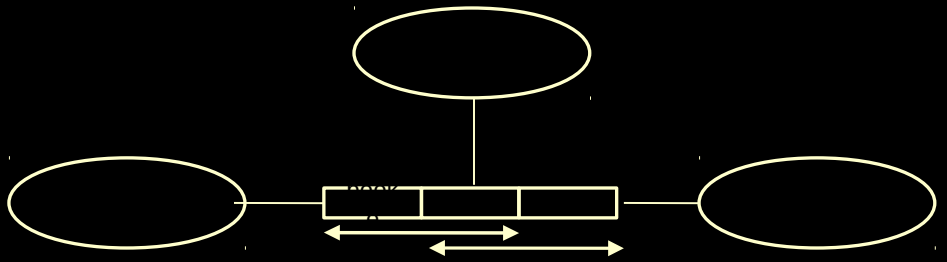
- In ORM, for an elementary n-ary association, each internal UC must span at least n-1 roles. In UML, a multiplicity constratint on a role of an n-ary association effectively constarints the population of the other roles combined.

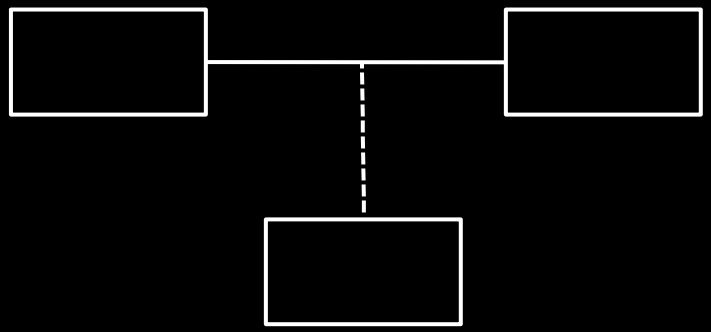
- There are cases where UML n-ary associations are incapable of capturing even a simple mandatory role constructions, or a minimum occurrence frequency constraint obove one.

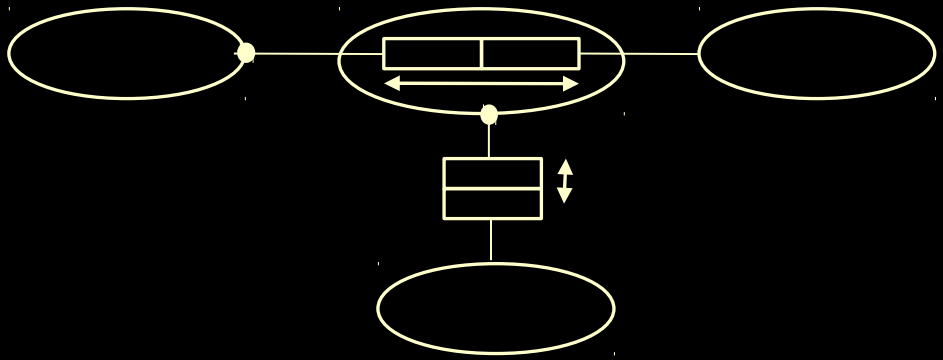
  ORM is far richer in this regard.

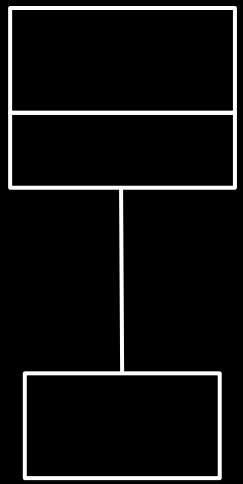
Some examples

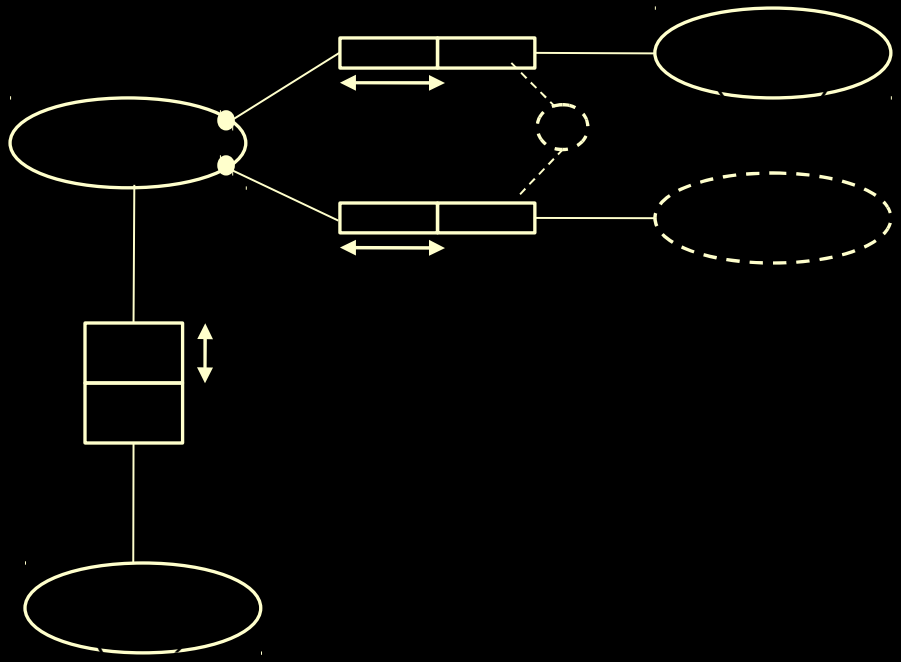






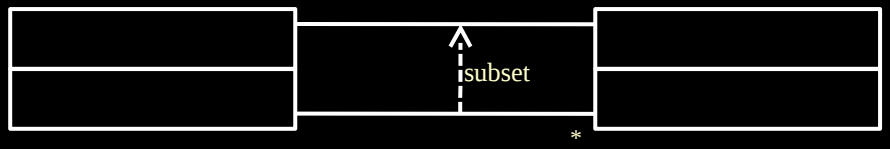




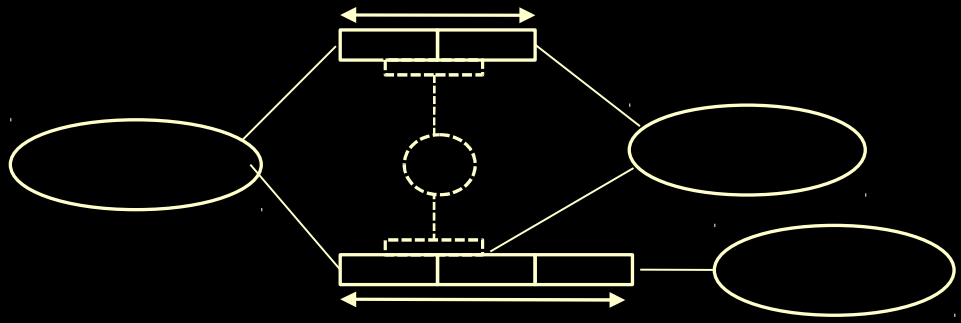


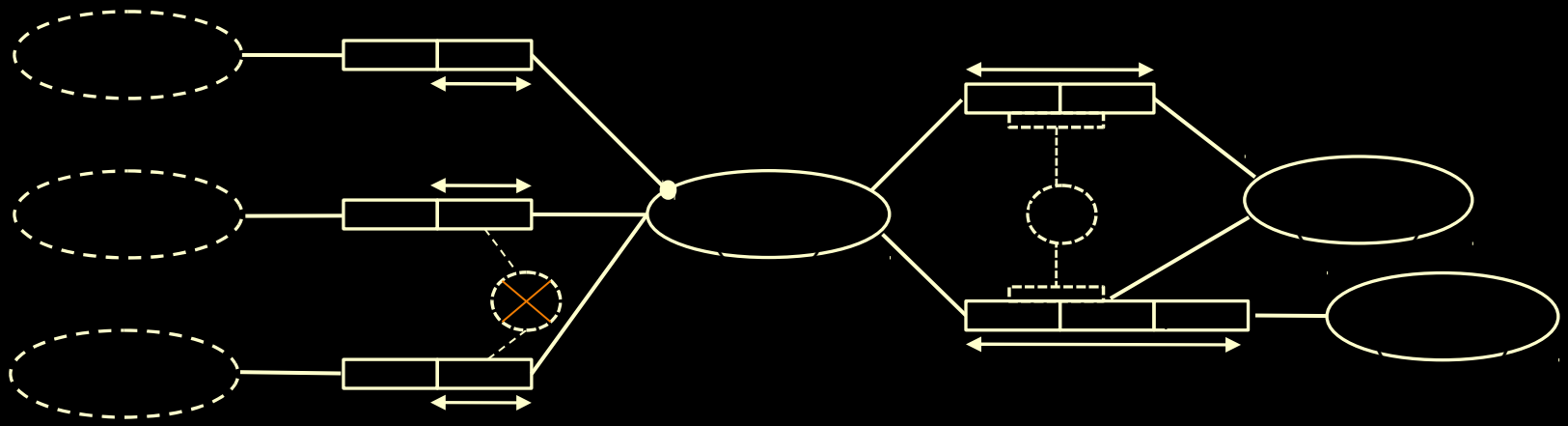
Set-Comparison Constraints

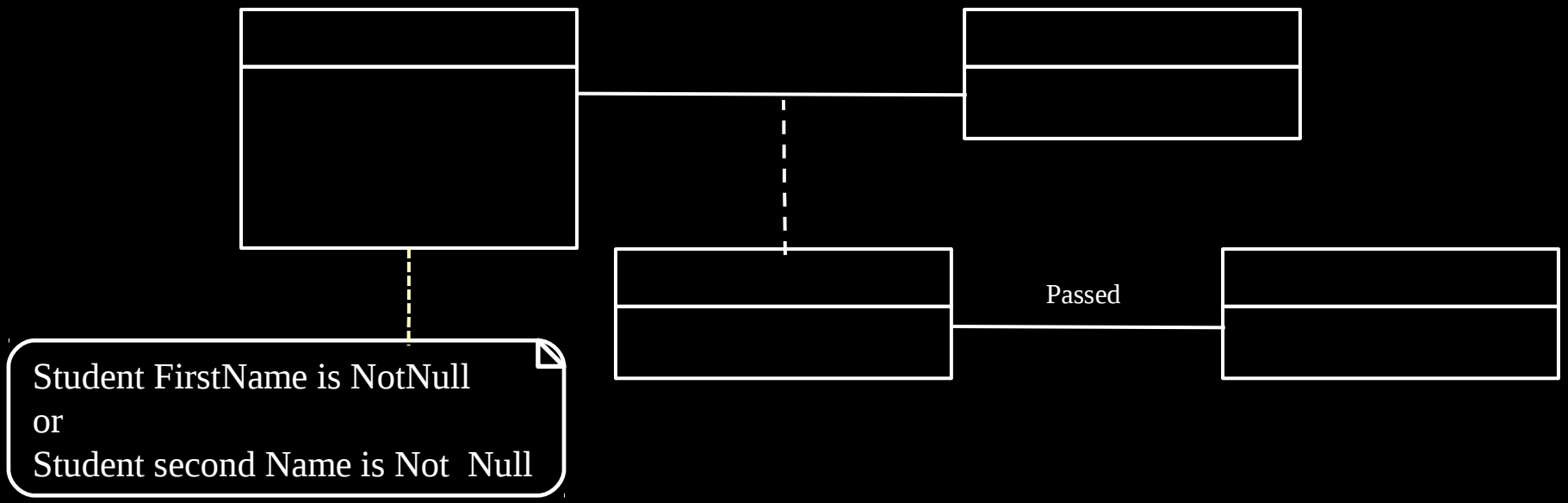
UML allows subset constraints to be specified between whole associations.

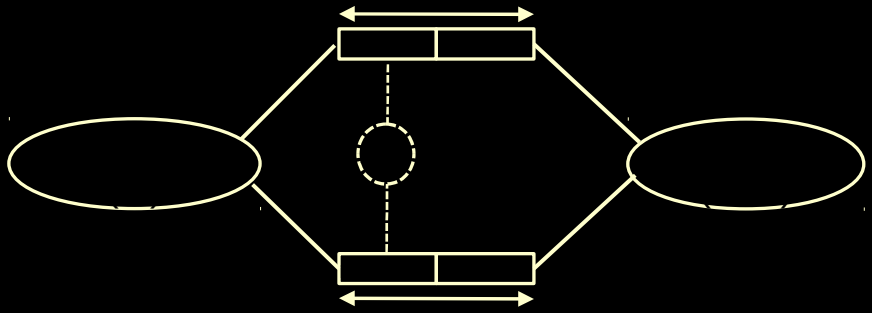


UML does not provide notation for subset constraints between single roles or parts of associations

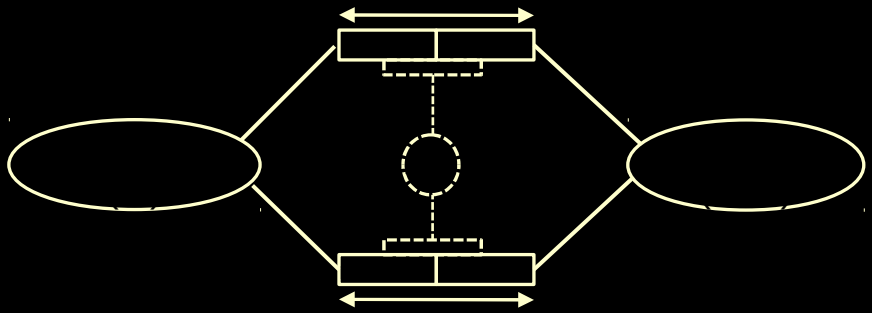




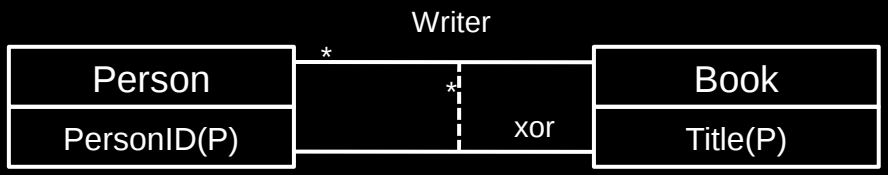

Passed
Student FirstName is NotNull
or
Student second Name is Not Null



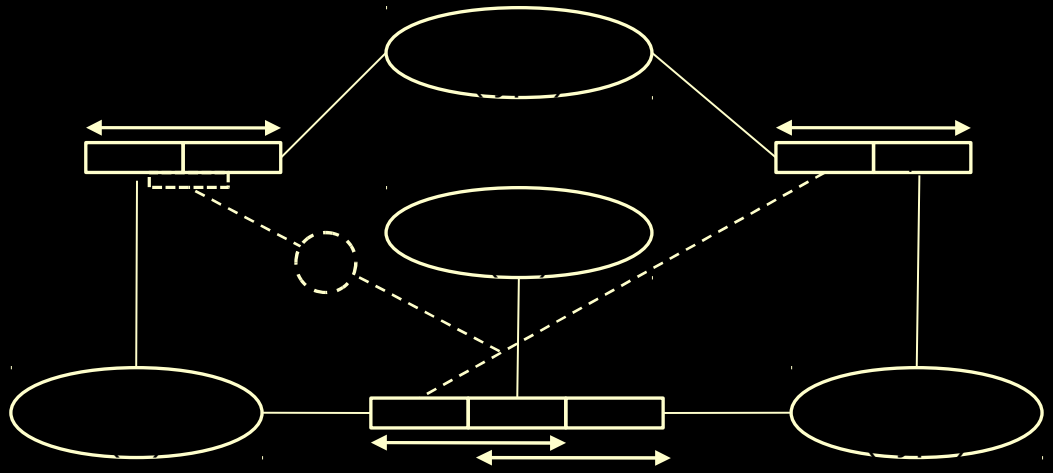
Meaning : A writer is never a reviewer

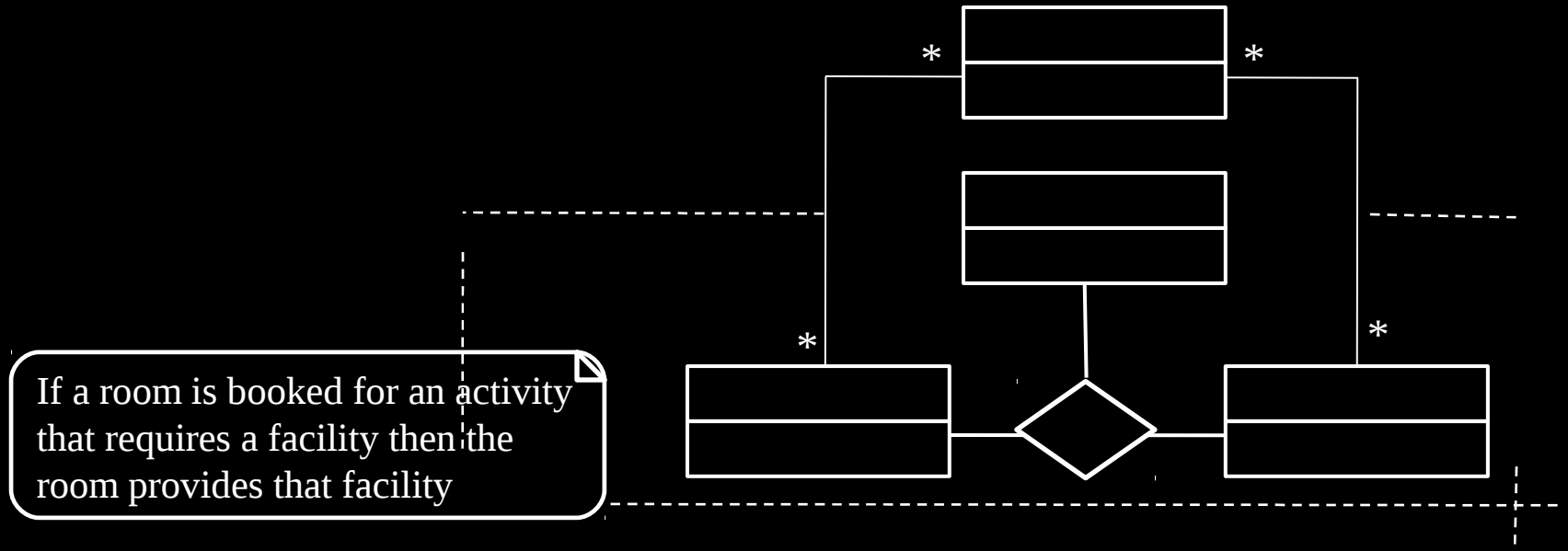


Meaning : A writer is never reviewing his/her books



Meaning : A writer is never reviewing own books. ULM does not provide graphical indication that would capture that difference




*
*
*
*
If a room is booked for an activity that requires a facility then the room provides that facility

Summary

- We have covered a selected aspects of the topic to indicate some differences of the expressiveness offered by the two conceptual languages – UML and ORM. The detailed comparison is presented in Terry's Halpin papers available on the Web;

UML Data Models From an ORM Perspective: Part 1 – Part 10

http://www.orm.net go to the bottom margine and follow *'UML and ORM'* link.

http://www.orm.net/pdf/ORM2_TechReport1.pdf