

# INFS4206/7206 Advanced Topics in Database

2006 Semester 2  
Part 1  
Advanced Topics

## Last Week



- ❖ **Model-Driven Architecture** main driving application for OMG.
- ❖ Most interesting aspect is **mapping** from one metamodel to another.
- ❖ Most interesting mappings **require more information** than exists in bare source metamodel.
- ❖ Many open problems.

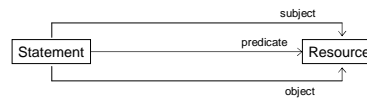
## MOF not the only way to metamodel

- ❖ The World-Wide Web consortium (W3C), in its Resource Description Framework (RDF), RDF Schema (RDFS) and Web Ontology Language (OWL) uses a metamodeling system very different from, and not equivalent to, the MOF.
- ❖ RDFS and OWL use RDF with built-in resources as its native metamodel.

## RDF Statement

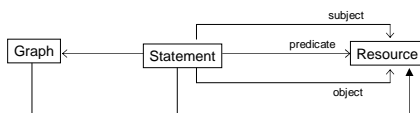
- ❖ Core of RDF is the statement, or triple, consisting of a subject, predicate and object all of which refer to resources.

- INFS4206 sameCourseAs INFS7206
- INFS4206 isPartlyAbout metamodeling



## Statements and Graphs are Resources

- ❖ A resource is something one can say something about (or something to say). So a statement can be about another statement.
- ❖ A graph is a collection of statements.



MOF is fine for all this

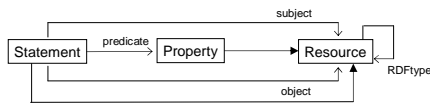
## Introducing Classes

- ❖ So far, RDF only models individual and their relationships.
- ❖ There is a special kind of resource called Property, defined as any resource used as the predicate of a statement



## Introducing Classes

- ❖ But this definition does not allow properties which don't happen to be used in a particular model instance, possibly incomplete.
- ❖ However, there is a built-in property `rdf:type` whose subject is defined as an instance of its object regarded as a class.



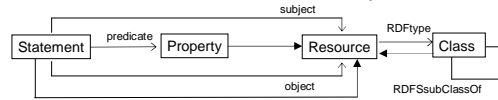
INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

7

## Introducing Classes

- ❖ Handy to distinguish resources which are classes from those not.
- ❖ And allow for empty classes
- ❖ And the subclass relationship
  - built-in property `rdfs:subClassOf`
  - Both a meta-association and an instance of a metaclass
  - As is `rdf:type`
  - Can rationalise this as abstract vs concrete syntax



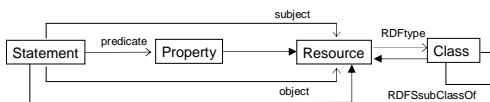
INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

8

## Statements about classes

- ❖ We want to make statements about Resource, Class, Property and their relationships
- ❖ Declare properties not yet used
- ❖ Introduce built-in classes `rdf:Property`, `rdfs:Class`, `rdfs:Resource`
- ❖ Violates the metalevel separation of MOF
- ❖ Not possible to fully model in MOF



INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

9

## RDFS weakly typed

- ❖ Much of the structure of RDFS model instance inferred from minimum evidence
- ❖ That a resource P is used as the predicate of a statement allows the inference
  - P `rdf:type` `rdfs:Property`
- ❖ The triple
  - S `rdfs:subClassOf` C
- ❖ Warrants the inferences
  - S `rdf:type` `rdfs:Class`
  - C `rdf:type` `rdfs:Class`
- ❖ Not possible to make (detect) a type error

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

10

## RDFS vs MOF

- ❖ RDFS is self-referential, MOF is not
- ❖ RDFS model mixes MOF metalevels
- ❖ RDFS weakly typed, MOF strongly typed
- ❖ RDFS could be used to model MOF, but not the reverse.

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

11

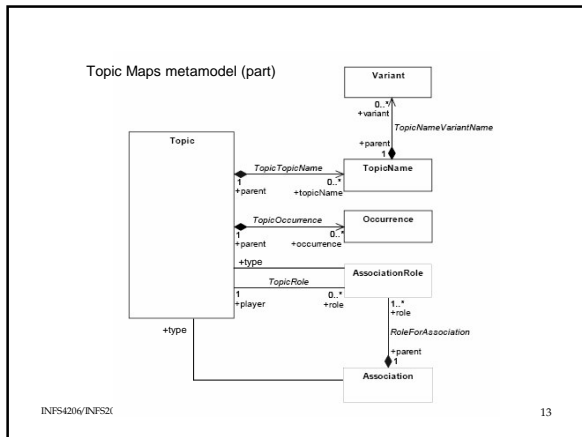
## Object Constraint Language

- ❖ UML does allow inference, though, via OCL.
- ❖ OCL is a specification language
- ❖ No side effects.
- ❖ No programming logic. Not directly executable
- ❖ Strongly typed
- ❖ Assumed instantaneously executed, so model can't change state during execution.
- ❖ We will describe by illustration from the Topic Maps metamodel

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

12



### Class-instance in TM

- ❖ Uses built-in instances of association

```

classDiagram
    class inst:Topic
    class Topic
    class asr:AssociationRole
    class Association

    inst:Topic --> Topic : roles
    Topic "1" -- "0..n" asr:AssociationRole : type
    asr:AssociationRole "1" -- "1" Association : parent
    Topic "1" -- "0..n" asr:AssociationRole : type
  
```

```

function StructuralType(top:Topic): Boolean
// True if topic is a structural type
{
  if top.subjectIdentifiers->exists(t |
    t = 'tmcore.type-instance' or
    t = 'tmcore.superType-subType')
  then true else false
  endif;
} // StructuralType
  
```

. Used to navigate model  
-> denotes a set  
Exists denotes non-empty set  
Variable scope whole statement

INFS4206/INFS206 part 1 week 4, 19 July 2006 Bob Colomb 14

### Definition of Type in TM

```

function TypeTopic(top: Topic) : Boolean
{
  // True if a topic instance is considered a class, as an OCL expression on the TM metamodel.
  if ((top.typeOf->exists(assoc : Association) and not StructuralType (top))
    // Topic is a type of an association, but not a structural type
    OR
    (top.roles->exists(asr | (asr.type.subjectIdentifiers = 'tmcore.type') and
    (asr.parent.type.subjectIdentifiers = 'tmcore.type-instance'))) OR
    // condition in Figure 2
    top.roles->exists(asr | asr.parent.type.subjectIdentifiers = 'tmcore.supertype-subtype')
  // Topic is either a subtype or a supertype, therefore a type.
  ) then true
  else false
  endif;
} // TypeTopic
  
```

INFS4206/INFS206 part 1 week 4, 19 July 2006 Bob Colomb 15

### Definition of Type in TM

```

classDiagram
    class Topic
    class TypeTopic
    class Association

    Topic --|> TypeTopic : subtype
    Topic --|> TypeTopic : supertype
    TypeTopic -- Association : type
  
```

Subclass of Topic defined as invariant  
context TypeTopic  
inv: TypeTopic(self)

INFS4206/INFS206 part 1 week 4, 19 July 2006 Bob Colomb 16

### Can constrain subclasses

```

classDiagram
    class A
    class B
    class C
    A --|> B
    A --|> C
  
```

- ❖ B and C are disjoint
  - context A
  - inv: not self.oclIsTypeOf(B) and self.oclIsTypeOf(C)
- ❖ B and C cover A
  - context A
  - inv: self.oclIsTypeOf(B) or self.oclIsTypeOf(C)

INFS4206/INFS206 part 1 week 4, 19 July 2006 Bob Colomb 17

### Iterators

- ❖ -> navigates to a set
- ❖ Iterators are operators defined over a set
  - Exists says a member exists satisfying condition
  - forall : all members satisfy condition
- ❖ Several other iterators defined

INFS4206/INFS206 part 1 week 4, 19 July 2006 Bob Colomb 18

## OCL

- ❖ OCL not clearly documented in OCL 2.0 spec
- ❖ Can get idea of use by looking in UML Infrastructure or Superstructure

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

19

## Mapping Among Metamodels

- ❖ We sometimes want to transform an instance of one metamodel to another metamodel
  - UML to OWL, say
- ❖ Extension of MDA mapping from PIM to PSM
- ❖ But different metamodels often have very different philosophies
  - Eg UML vs RDFS
- ❖ General mappings often not satisfactory
- ❖ Need to take into account characteristics of particular model instances being mapped

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

20

## Kinds of Mapping Failures

- ❖ *structure conflation*
  - Two constructs in one system map to a single construct in the other. In this case, a general-purpose mapping doesn't round trip. UML binary associations and class-valued attributes map to OWL properties, for example. In topic maps, three different kinds of identifier map to one kind in OWL.
  - But there is nothing to stop a particular project from specifying naming conventions so there is a record in the target of what construct the source was, and from maintaining that convention in subsequent development.

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

21

## Kinds of Mapping Failures

- ❖ *structure loss*
  - A complex construct is mapped to a collection of simpler constructs. Hard to reverse map.
  - Examples here are UML N-ary associations and association classes, which get mapped to a class and a collection of properties. In Topic Maps, the Association construct is typed itself and has N typed roles. The association maps to a class and the typed roles to properties. It is in general impossible to reliably map the reverse.
  - A particular project can use naming conventions or annotations to retain a memory of the structure, and maintain those conventions in subsequent maintenance so as to be able to reverse map.
  - A TM project could decide to limit itself to binary associations, making possible mapping associations directly to properties in that particular case.

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

22

## Kinds of Mapping Failures

- ❖ *trapdoor mappings*
  - A kind of construct in the source is mapped to a very specific arrangement of a general structure in the target. The analogy is with cryptography, where the encryption function takes any plaintext into an encrypted text, but almost no encrypted texts map back to plaintexts.
  - In topic maps, this occurs with the mapping of scope and variant names to specific properties in OWL identified with TM URIs. OWL properties map to TM associations with specific roles named with OWL URIs. Unless the source for a reverse mapping happened to maintain these conventions, it would be impossible to reverse in a sensible way.

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

23

## Kinds of Mapping Failures

- ❖ *feature lack*
  - The target metamodel lacks a feature present in the source. In this case there is no apparent general way to map the feature from the source. But in a particular project the feature may for example be used in a particular way leading to a mapping to target features particularized by naming conventions. OWL restriction classes relative to UML or Topic Map are of this kind.

INFS4206/INFS206 part 1 week 4, 19 July 2006

Bob Colomb

24

## Kinds of Mapping Failures

### ❖ *incompatible structural principles*

- The different metamodels are organized very differently. UML is organized around classes, with instances as subordinate objects. OWL has both classes and individuals typed only by a universal superclass. In Topic Maps, a Topic instance can be either typed or not. But a particular project might use a particular discipline in its use of these structures leading to mappings not otherwise feasible.

## Summary: Key Terms



- ❖ There are very different metamodeling systems
- ❖ MOF has a constraint language called OCL.
- ❖ There are not generally satisfactory mappings among different metamodels

## Resources

- ❖ OCL 2.0
  - On course web site as pdf file.
- ❖ RDF Concepts
- ❖ RDF Schema
  - On course web site as html file.