HY 351: Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
CS 351: Information Systems Analysis and Design

HY351:
Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Information Systems Analysis and Design

Εννοιολογική Μοντελοποίηση με Σημασιολογικά Δίκτυα
Περίπτωση: Η γλώσσα Telos και το Semantic Index System

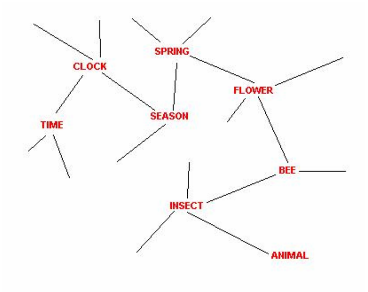
Γιάννης Τζίτζικας

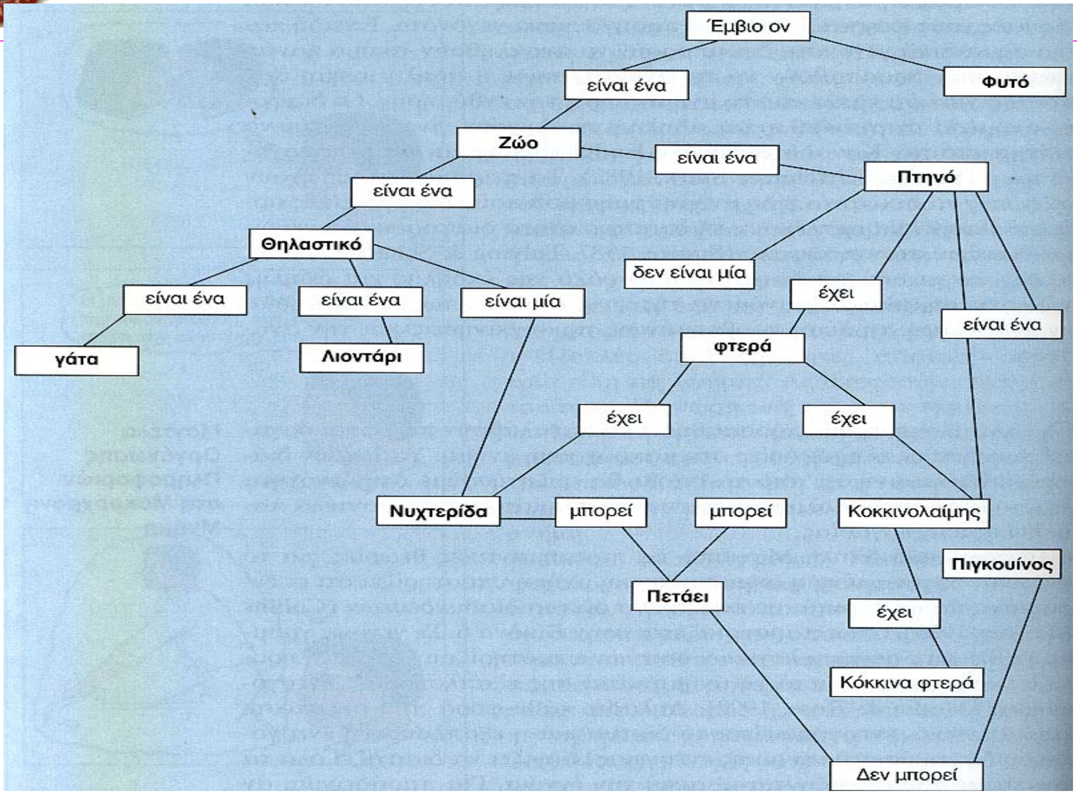
Διάλεξη :
Ημερομηνία :

Διάρθρωση

- Σύντομη εισαγωγή στα Σημασιολογικά Δίκτυα
- Η γλώσσα SIS-Telos και το Semantic Index System
  - Παραδείγματα δομικής μοντελοποίσης

Εισαγωγή στα Σημασιολογικά Δίκτυα (Semantic Networks)





Short introduction to Semantic Networks

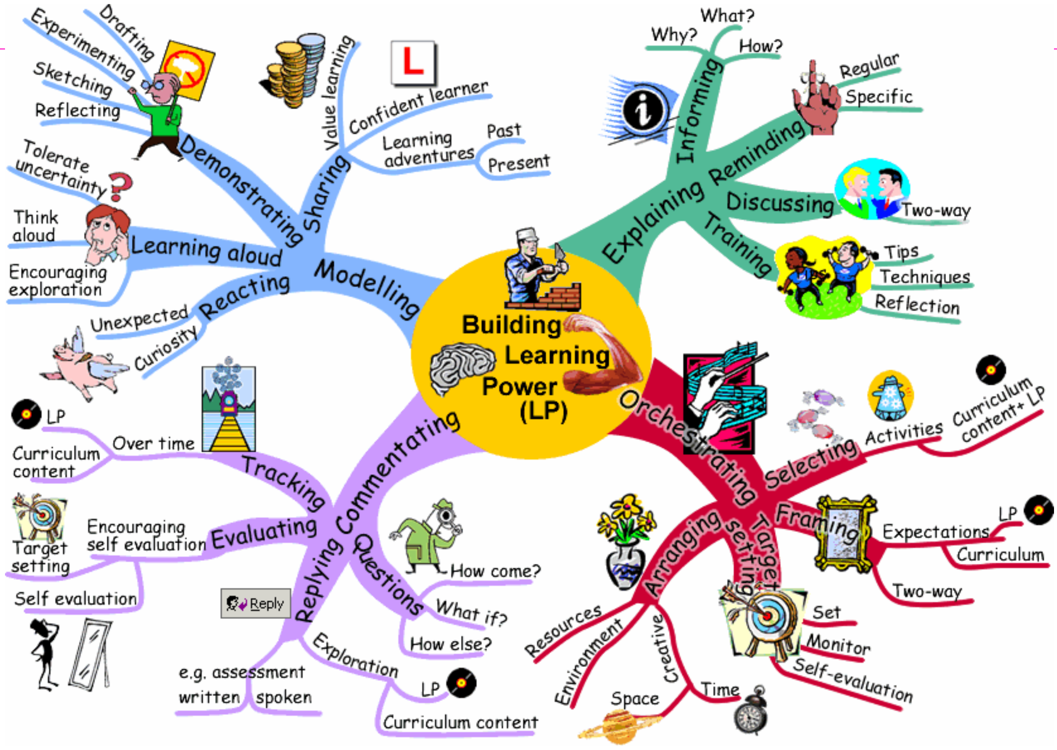
- A **semantic network** is often used as a form of knowledge representation.
- It is a *directed graph* consisting of *vertices*, which represent concepts, and *edges*, which represent semantic relations between the concepts.
- Indicative semantic relations:
  - Meronymy (A is part of B, i.e. B has A as a part of itself)
  - Holonymy (B is part of A, i.e. A has B as a part of itself)
  - Hyponymy (or troponymy) (A is subordinate of B; A is kind of B)
  - Hypernymy (A is superordinate of B)
  - Synonymy (A denotes the same as B)
  - Antonymy (A denotes the opposite of B)
- An example of a semantic network is *WordNet*, a lexical database of English. Such networks involve fairly loose semantic associations that are nonetheless useful for human browsing.

Short introduction to Semantic Networks (cont)

- It is possible to represent logical descriptions using semantic networks such as the **Existential Graphs** (of Charles S. Peirce) or the related **Conceptual Graphs** (of John F. Sowa). These have expressive power equal to or exceeding standard first-order predicate logic.
  - Unlike WordNet or other lexical or browsing networks, semantic networks using these can be used for reliable automated logical deduction. Some automated reasoners exploit the graph-theoretic features of the networks during processing.
- "Semantic Nets" were first invented for computers by Richard H. Richens of the Cambridge Language Research Unit in 1956 as an "interlingua" for machine translation of natural languages.
  - They were developed by Robert F. Simmons at System Development Corporation, Santa Monica, California in the early 1960s and later featured prominently in the work of M. Ross Quillian in 1966.

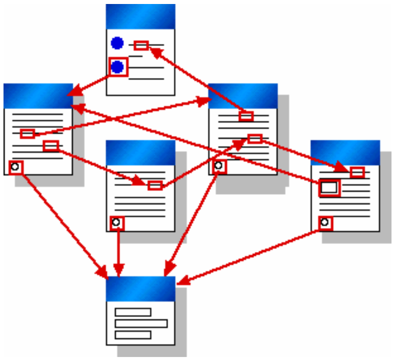
Short introduction to Semantic Networks (cont)

- One can consider a ***mind map*** to be a very free form variant of a semantic network. By using colors and pictures the emphasis is on generating a semantic net which evokes human creativity. However, a fairly major difference between mind maps and semantic networks is that the structure of a mind map, with nodes propagating from a centre and sub-nodes propagating from nodes, is hierarchical, whereas semantic networks, where any node can be connected to any node, have a more heterarchical structure.



Short introduction to Semantic Networks (cont)

- In the 1960s to 1980s the idea of a semantic link was developed within *hypertext systems* as the most basic unit, or edge, in a semantic network. These ideas were extremely influential, and there have been many attempts to add typed link semantics to HTML and XML.
  - latest attempt: the Semantic Web



SIS-Telos

For more see

http://www.ics.forth.gr/isl/r-d-activities/sis.html

Γιατί να μιλήσουμε για αυτήν τη γλώσσα;

- Για να δούμε
  - περισσότερα παραδείγματα δομικής μοντελοποίησης
  - τους βασικούς μηχανισμούς δόμησης σημασιολογικών δικτύων
  - δομικά μοντέλα που δεν μπορούν να παρασταθούν με τους δομικούς μηχανισμούς που μας δίνουν οι δημοφιλείς γλώσσες αντικειμενοστρεφούς προγραμματισμού ή τα δομικά μοντέλα της UML

SIS-Telos

SIS-Telos: A knowledge representation language supporting a structurally object oriented data model

No operations

Features:

- Every object has a unique system identifier and a unique logical name
- Objects are structured along three main hierarchies:
  - the classification hierarchy — **instanceOf** ⇢
  - the generalization/specialization hierarchy — **isA** ⇒
  - the aggregation(attribute) hierarchy — **attribute** →

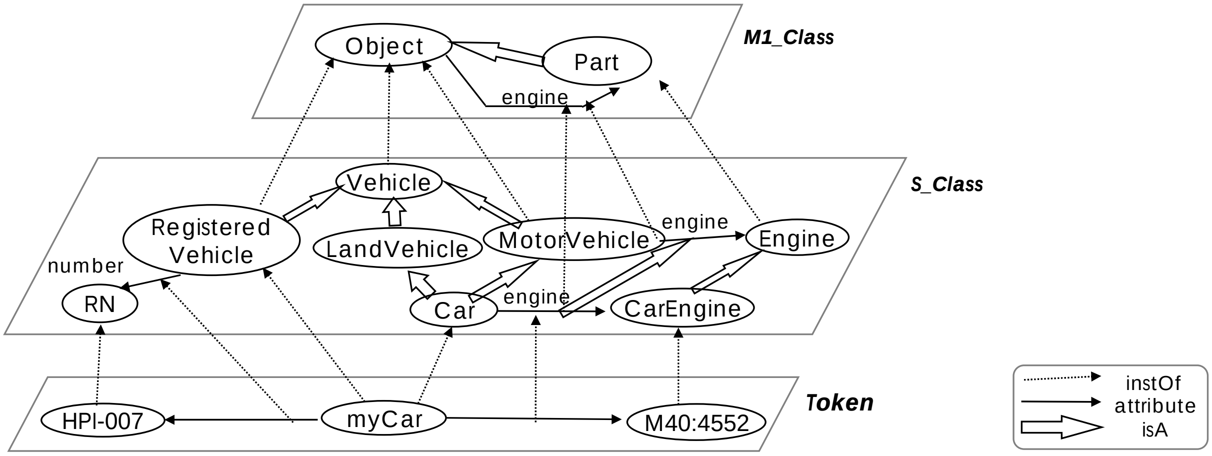
SIS-Telos and SIS

SIS-Telos

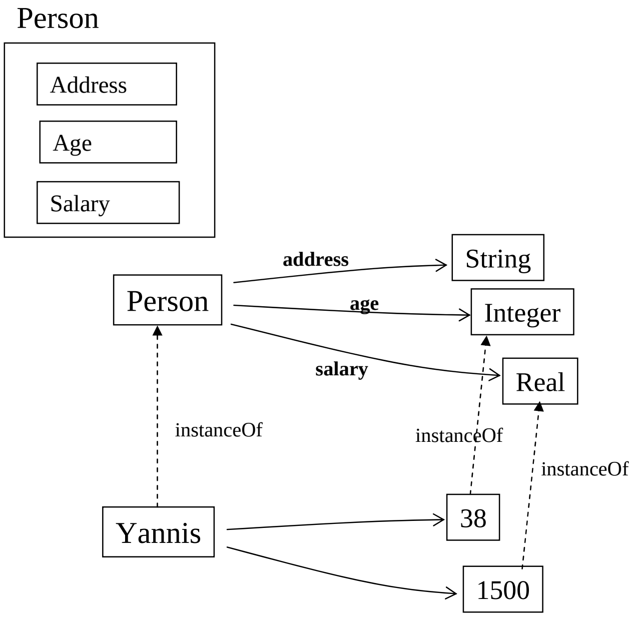
- object naming
- classification
- generalization
- attribution

Semantic Index System (SIS)

- DBMS functionality
- bidirectional link storage
- recursive (navigational) queries
- schema evolution at run-time



From tables to networks

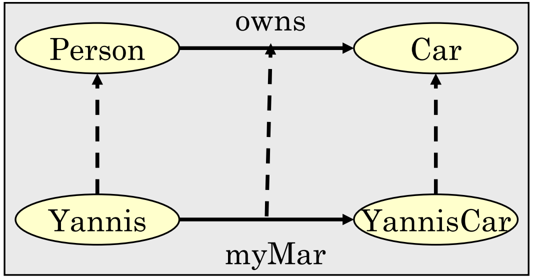


RETELL Individual *Person* in S_Class
with attribute
address : String ;
age : Integer ;
salary : Real
end

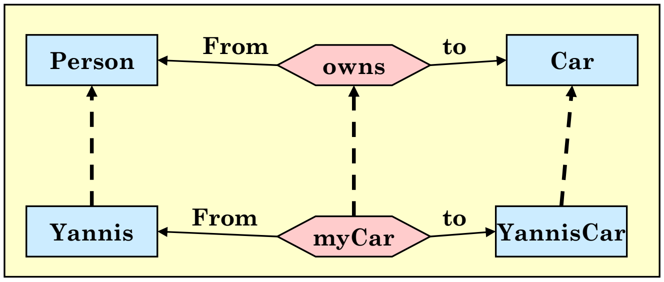
RETELL Individual *Yannis* in Token, *Person*
with
age : 38;
salary : 1500
end

Objects are partitioned to **Individuals** and **Attributes**

Conceptually



SIS-Telos Implementation



Individual Object 🟦

Attribute Object ⬡

Logical Names:

Individual: **"Yannis"**

Attribute: **"owns from Person"**
"myCar from Yannis"

Primitive Data Types:

Telos_String
Telos_Integer
Telos_Real
Telos_Time

Instantiation/Classification

- - - - - - - - - →

- Every object (individual or attribute) should be classified to one of the instantiation levels:
  - **Token** (objects here denote atomic objects)
  - **S_Class** (objects here denote classes, i.e. sets of atomic objects)
  - **M1_Class** (objects here denote metaclasses, i.e. sets of sets of objects)
  - …
- Instantiation has set-membership semantics
  - a Token object can be classified to a S_Class object
  - a S_Class object can be classified to a M1_Class object
- Multiple Classification: an object can be classified to one or more classes

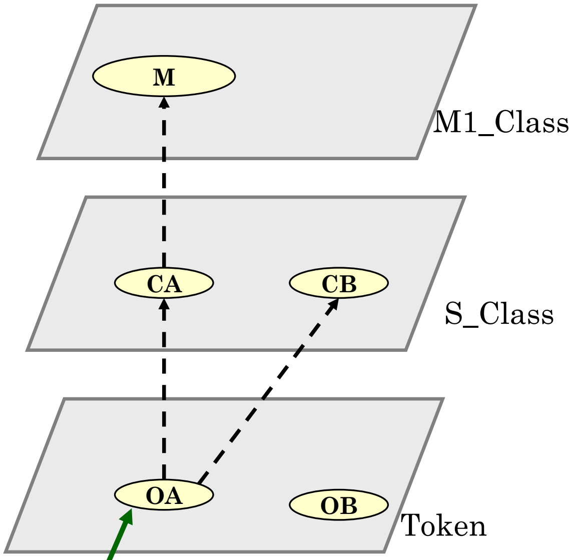
InstanceOf between Individuals

```
Tell Individual M  in M1_Class
end
```

```
Tell Individual CA  in S_Class, M
end
```

```
Tell Individual OA  in Token, CA, CB
end
```

```
Tell Individual OB  in Token
end
```



Multiple Classification

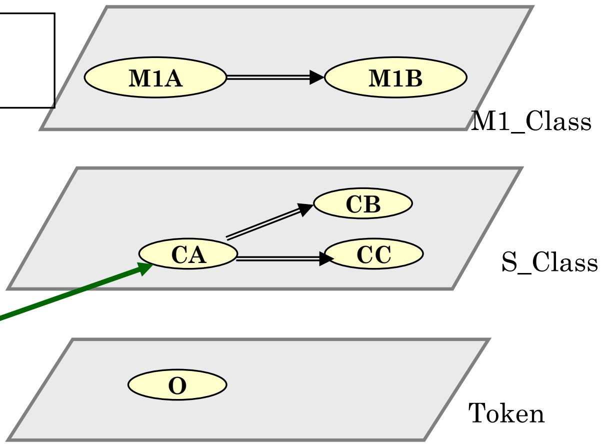
Gereralization/Specialization (IsA) relationships

- IsA links can relate objects of the <u>same instantiation level</u> (except Tokens) and same <u>type</u>
  - individuals with individuals
  - attributes with attributes
- The specialization has <u>subset-semantics</u>
- <u>Multiple</u> Specialization/Generalization
  - Integrity Constraint: The IsA lattice must me acyclic
- Inheritance
  - A subclass inherits all the attributes of its superclasses
  - A subclass may refine the range of an inherited attribute by specializing it

IsA between Individuals

```
Tell Individual M1A   in M1_Class isA M1B
end
```

```
Tell Individual CA   in S_Class isA  CB,CC
end
```

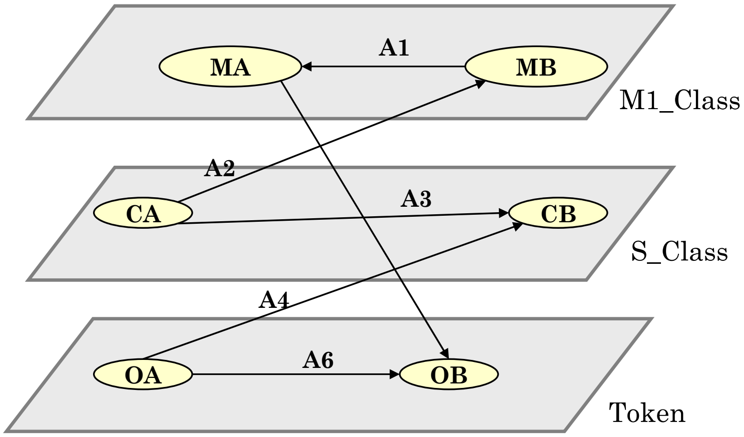


Multiple Inheritance

Attribution

- Attributes are first class objects thus can be structured along classification, generalization and attribution.
- Attributes may relate
  - an Individual with an Individual
  - an Attribute with an Individual
- The instantiation level of an attribute should be less or equal to the minimum of the instantiation levels of its ends.

Attributes between Individuals



Implicit declaration of attributes

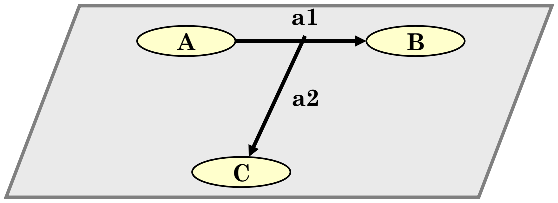
```
Tell Individual OA   in  Token
   with attribute
         A4: CB
         A6: OB
end
```

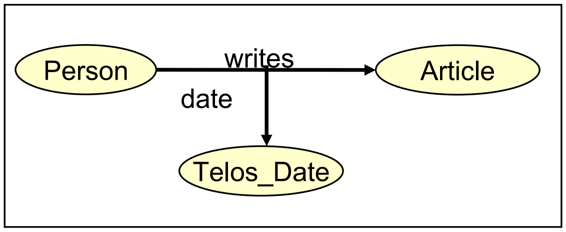
Explicit declaration of attributes

```
Tell Attribute A4
   from   : OA
   to        :CB
   in Token
end
```

Attributes from Attributes to Individuals

```
Tell Attribute a1
     from :A
     to      :B
  in Token
     with attribute
             a2: C
end
```





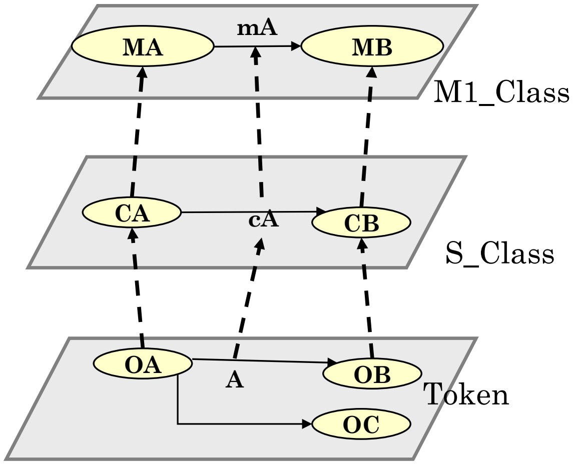
InstanceOf between Attributes

An attribute A1 might be InstanceOf- A2 provided that the origin and the destination of A1 are instances of the origin and the destination of A2

```
Tell Individual MA  in M1_Class
  with attribute
           mA : MB
end
```

```
Tell Individual CA  in S_Class, MA
  with mA
      cA  : CB
end
```

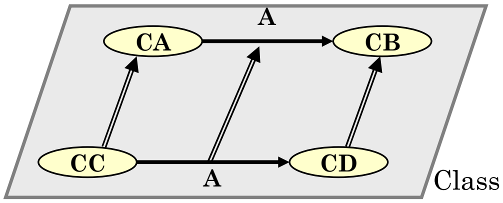
```
Tell Individual OA  in Token, CA
  with attribute
          : OC
  with cA
      A  : OB
 end
```



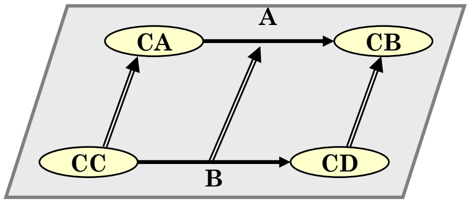
IsA between Attributes

Attributes classes might be Isa-related provided that the origin and the destination object of the subclass are subclasses of the origin and the destination object of the relevant superclass

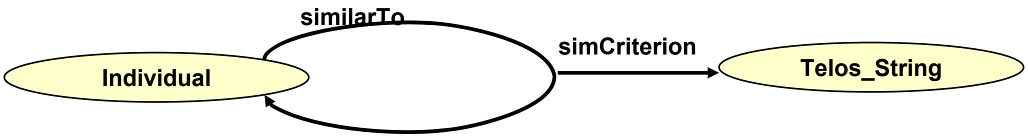
```
Tell Individual CC    in S_Class isA  CA
  with attribute
           A : CD
end
```



```
Tell Individual CC    in S_Class isA  CA
end

Tell AttributeClass B
    from :CC
    to     :CD
  in S_Class isA **A from CA**
end
```



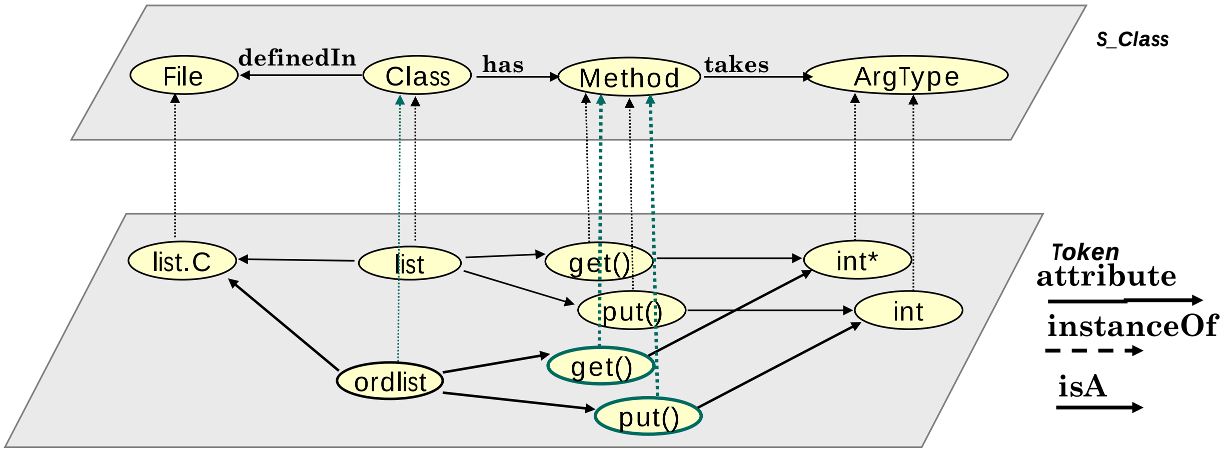
“Omega” classes



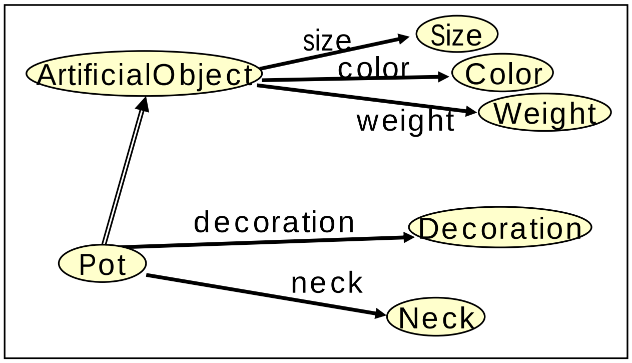
```
Tell AttributeClass similarTo
    from :Individual
    to     :Individual
  in S_Class
     with attribute
          simCriterion: Telos_String
end
```

Modeling Example:
Static Analysis of Software

Multivalued attributes



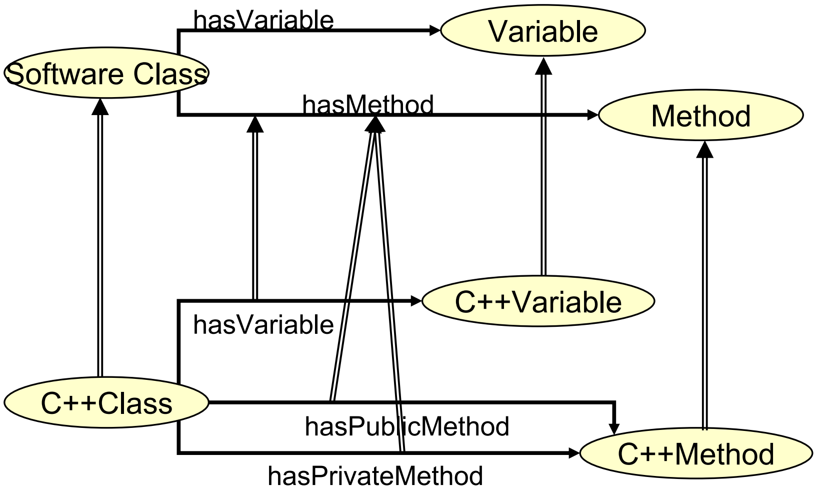
Modeling Example: Inheritance of Attributes



```
Tell Individual  Pot in S_Class
                    isA ArtificialObject
  with attribute
       decoration : Decoration
       neck          : Neck
end
```

```
Tell Individual  Pot11 in Token, Pot
with size
          : Size11
      color
           : Brown
      weight:
           : weight11
      decoration
          :  Minoan
      neck
          :  NeckWideOpen
end
```

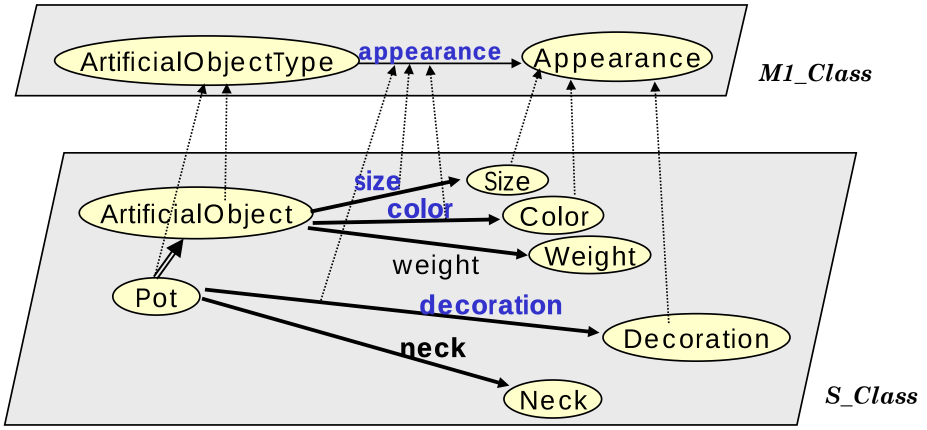
Specialization of Inherited Attributes



```
Tell Individual  C++Class
      in S_Class isA SoftwareClass
  with attribute
        hasVariable : C++Variable
 end
```

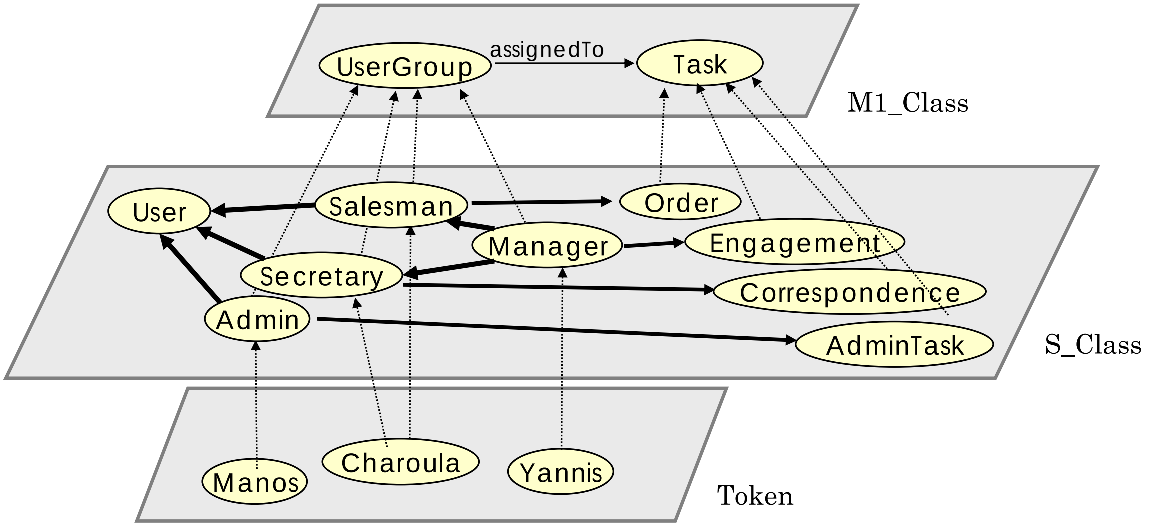
```
Tell Attribute  hasPublicMethod  in S_Class
   from   : C++Class
   to       : C++Method
in S_Class,  isA  hasMethod from SoftwareClass
end
```

Metacategories (attribute metaclasses)

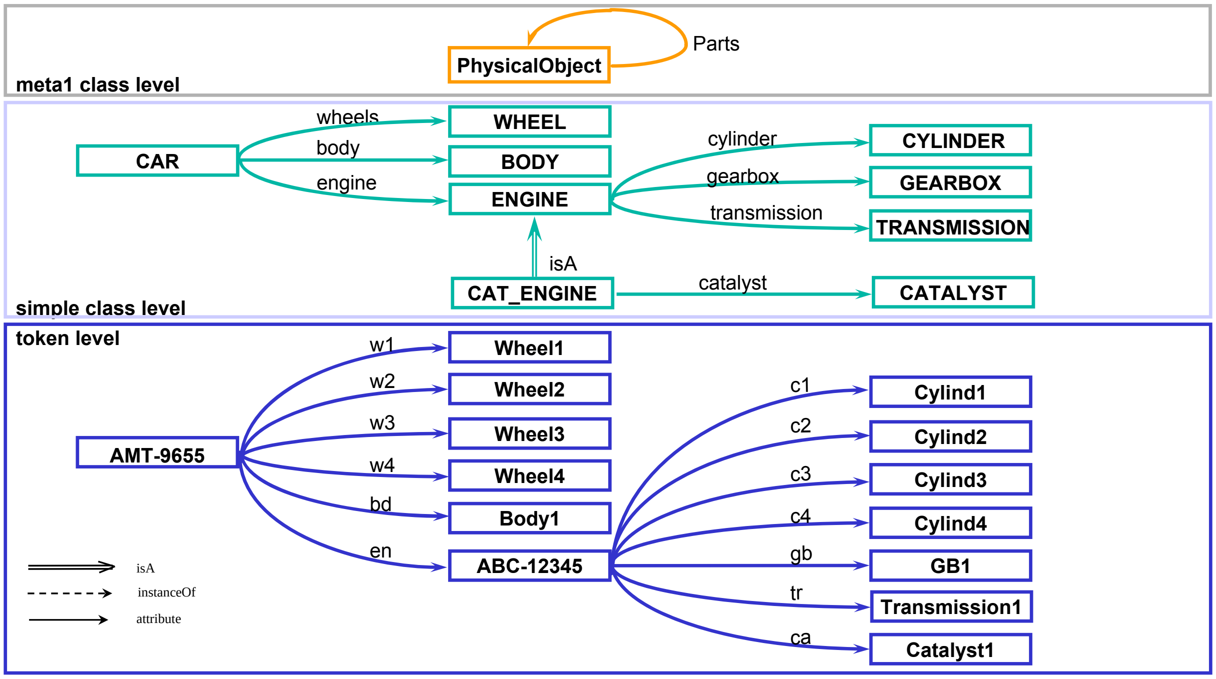


```
Tell Individual  Pot in S_Class, ArtificialObjectType  isA ArtificialObject
   with appearance
          decoration : Decoration
   with attribute
          neck        : Neck
end
```

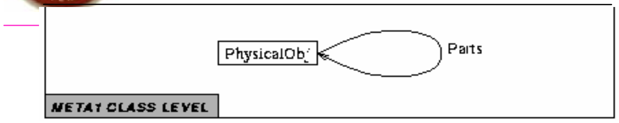
Example

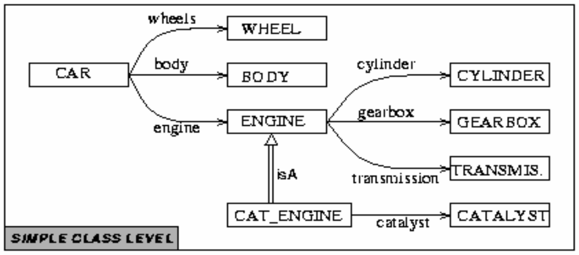


SIS-TELOS
The CAR example [1]



SIS-TELOS
The CAR example[2]

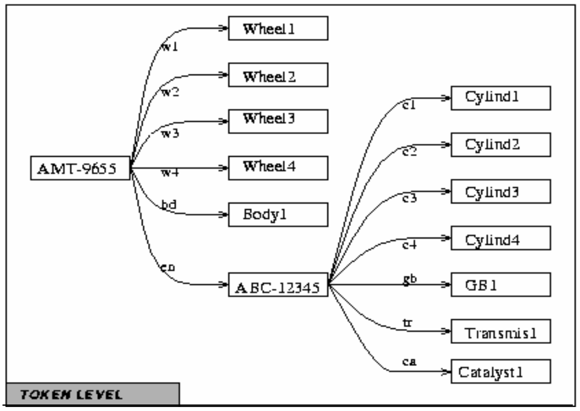




```
RETELL Individual CAR in S_Class, PhysicalObj
with Parts
      wheels : WHEEL;
      body   : BODY;
      engine : ENGINE
end CAR

RETELL Individual WHEEL in S_Class, PhysicalObj
end WHEEL

RETELL Individual BODY in S_Class, PhysicalObj
end BODY
```

```
RETELL Individual PhysicalObj in M1_Class
with attribute
      Parts : PhysicalObj
end PhysicalObj

RETELL Individual ENGINE in S_Class, PhysicalObj
with Parts
      cylinder : CYLINDER;
      gearbox  : GEARBOX;
      transmission : TRANSMISSION
end ENGINE

RETELL Individual CYLINDER in S_Class, PhysicalObj
end CYLINDER

RETELL Individual GEARBOX in S_Class, PhysicalObj
end GEARBOX

RETELL Individual  TRANSMISSION in S_Class, PhysicalObj
end TRANSMISSION

RETELL Individual CAT_ENGINE in S_Class, PhysicalObj
  isA ENGINE
with  Parts
         catalyst : CATALYST
end CAT_ENGINE

RETELL Individual CATALYST in S_Class, PhysicalObj
end CATALYST
```

SIS-TELOS
The CAR example[3]



RETELL Individual (*AMT-9655*) in Token, *CAR*
with *wheels*
w1 : *Wheel1*;
w2 : *Wheel2*;
w3 : Wheel3;
w4 : *Wheel4*
with *body*
bd : *Body1*
with *engine*
en : (*ABC-12345*)
end (*AMT-9655*)

RETELL Individual *Wheel1* in Token, *WHEEL*
end *Wheel1*

RETELL Individual *Wheel2* in Token, *WHEEL*
end *Wheel2*

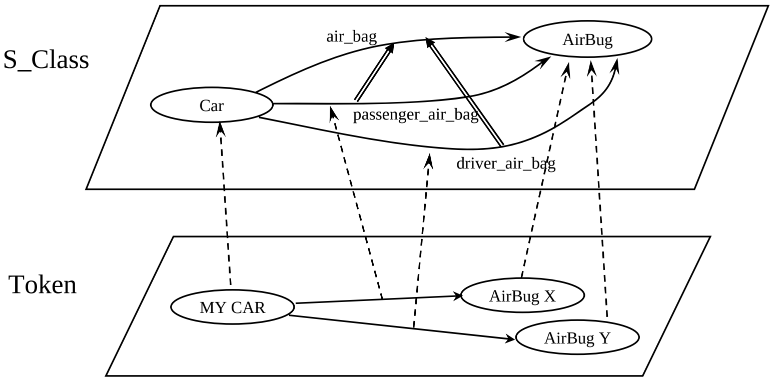
RETELL Individual *Wheel3* in Token, *WHEEL*
end *Wheel3*

RETELL Individual *Wheel4* in Token, *WHEEL*
end *Wheel4*

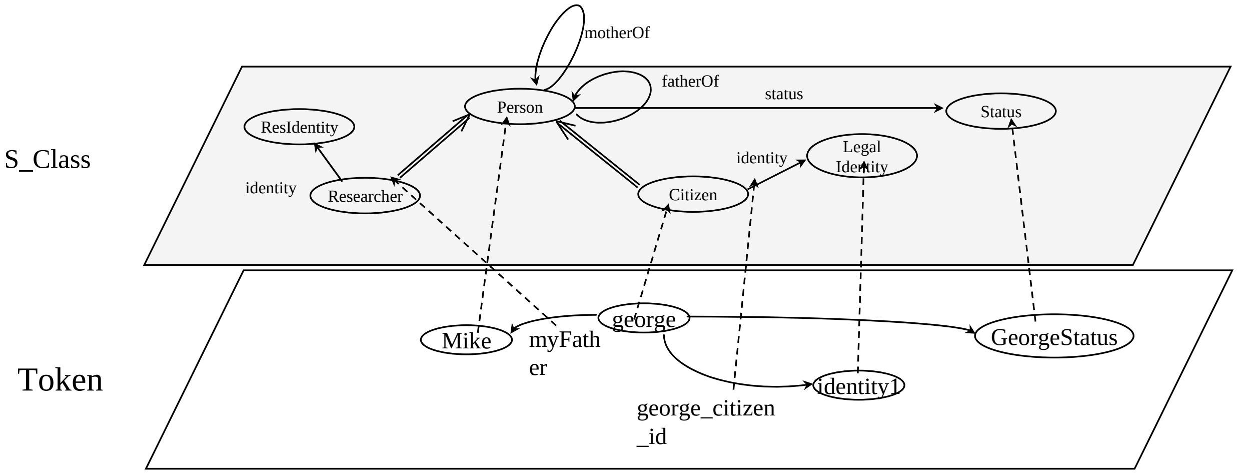
RETELL Individual *Body1* in Token, *BODY*
end *Body1*

RETELL Individual (*ABC-12345*) in Token,*CAT_ENGINE*
with *cylinder*
c1 : *Cylind1*;
c2 : *Cylind2*;
c3 : *Cylind3*;
c4 : *Cylind4*
with *gearbox*
gb : *GB1*
with *transmission*
tr : *Transmis1*
with *catalyst*
ca : *Catalyst1*
end (*ABC-12345*)

Example of isA-related attributed



Example of multiple classification


motherOf
fatherOf
status
Person
Status
ResIdentity
S_Class
identity
Legal Identity
identity
Researcher
Citizen
Token
george
Mike
myFath er
GeorgeStatus
identity1
george_citizen _id