LEZIONI 45-46
METAMODELLAZIONE, METACLASSI, E REFLECTION
… UML e Java

Ingegneria del Software e Progettazione Web

Università degli Studi di Tor Vergata - Roma

Guglielmo De Angelis
guglielmo.deangelis@isti.cnr.it

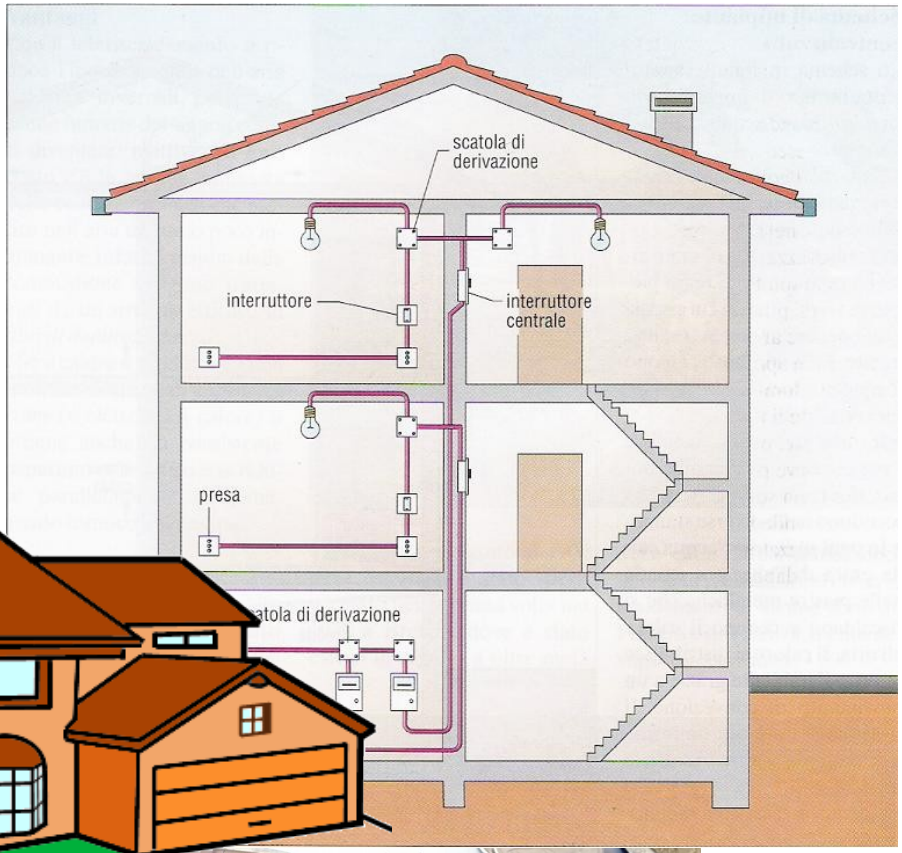
linguaggi di modellazione … “at a glance”

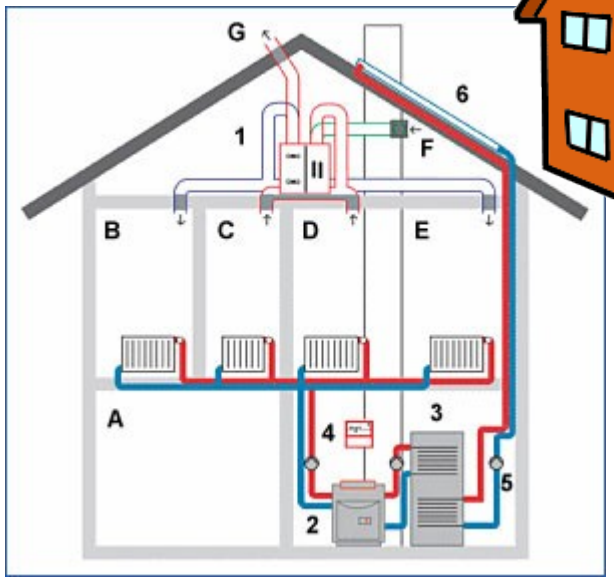
- i linguaggi di modellazione sono spesso framework concettuali
- il loro obiettivo è supportate i progettisti nella
  - formalizzazione dei loro pensieri
  - rappresentare la realtà (o una sua particolare visione)

linguaggi di modellazione … “at a glance”

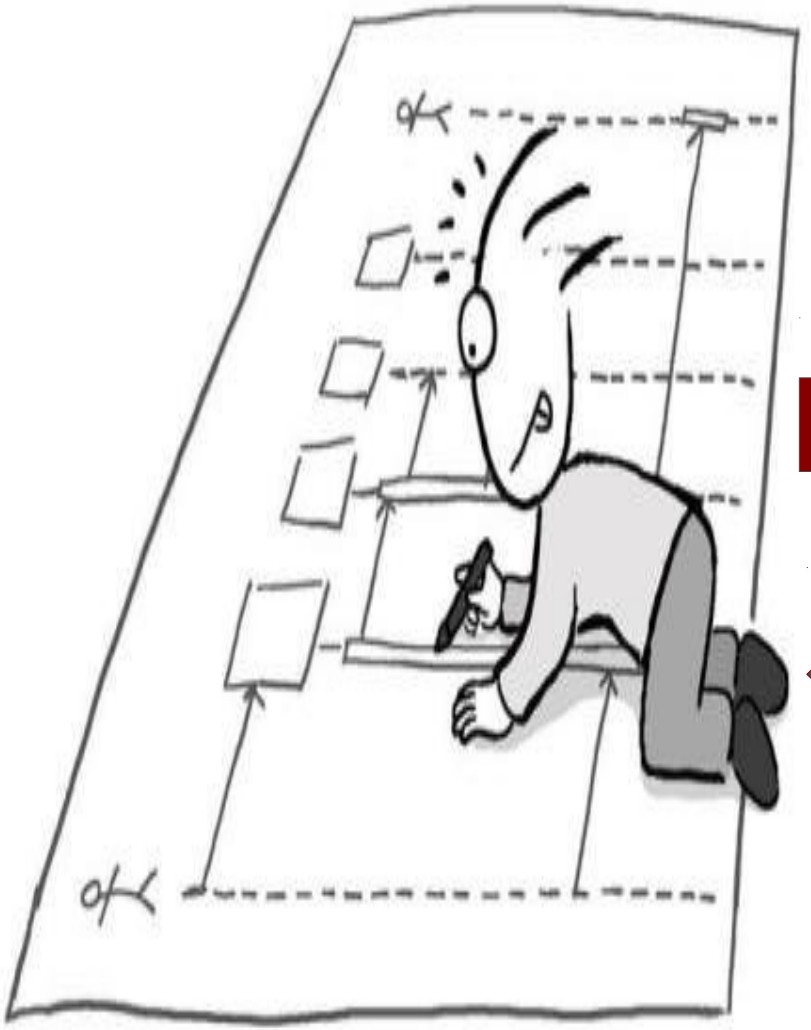
- in generale consentono di modellare vari aspetti di un sistema
  - e.g. offrendo una combinazione di viste linkabili
- tipicamente ogni aspetto può essere descritto con uno o più specifiche/diagrammi che possono
  - usare vari simboli/notazioni
    - formale o semi-formale
    - testuale, grafica, o entrambe
  - focalizzarsi su differenti rappresentazioni del problema e della soluzione
- comunemente questi aspetti sono classificati in
  - statici (o strutturali) : elementi e loro relazioni
  - dinamici : azioni o interazioni relative agli elementi modellati

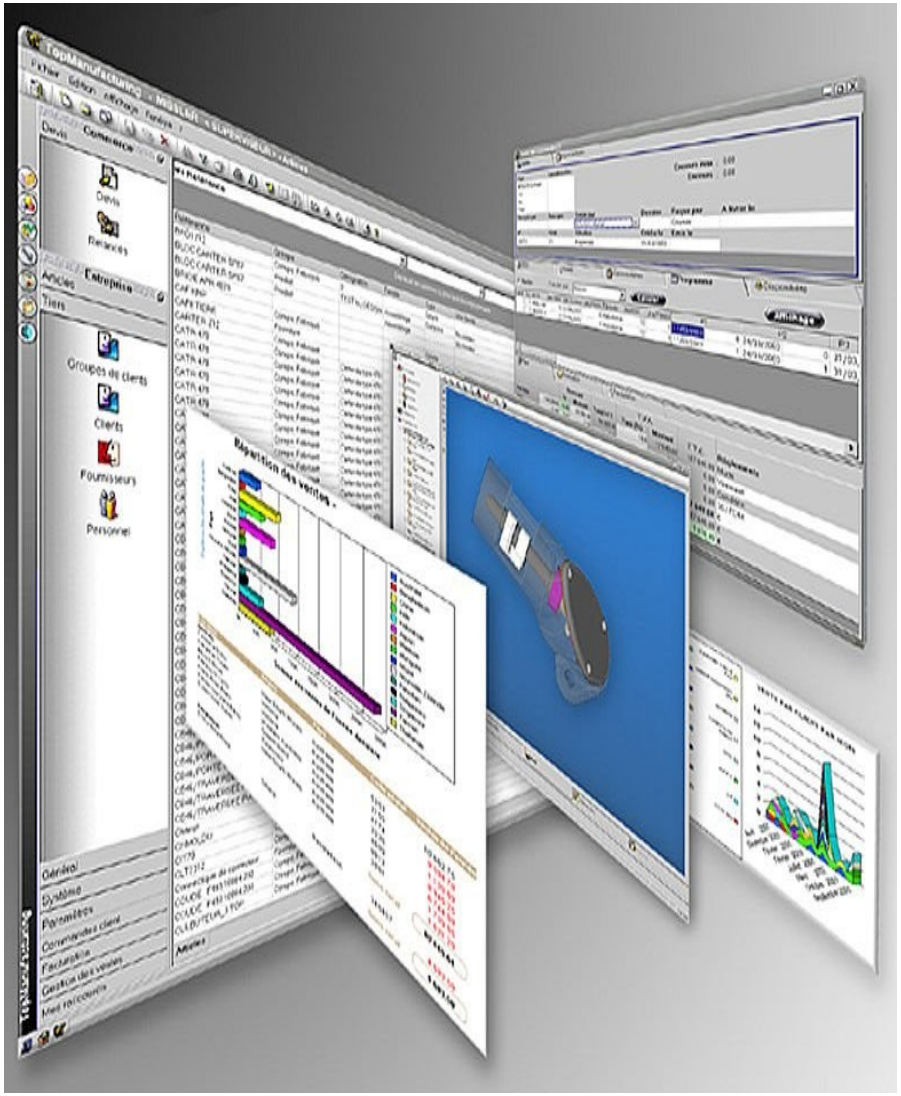
torniamo all'esempio della casa ...



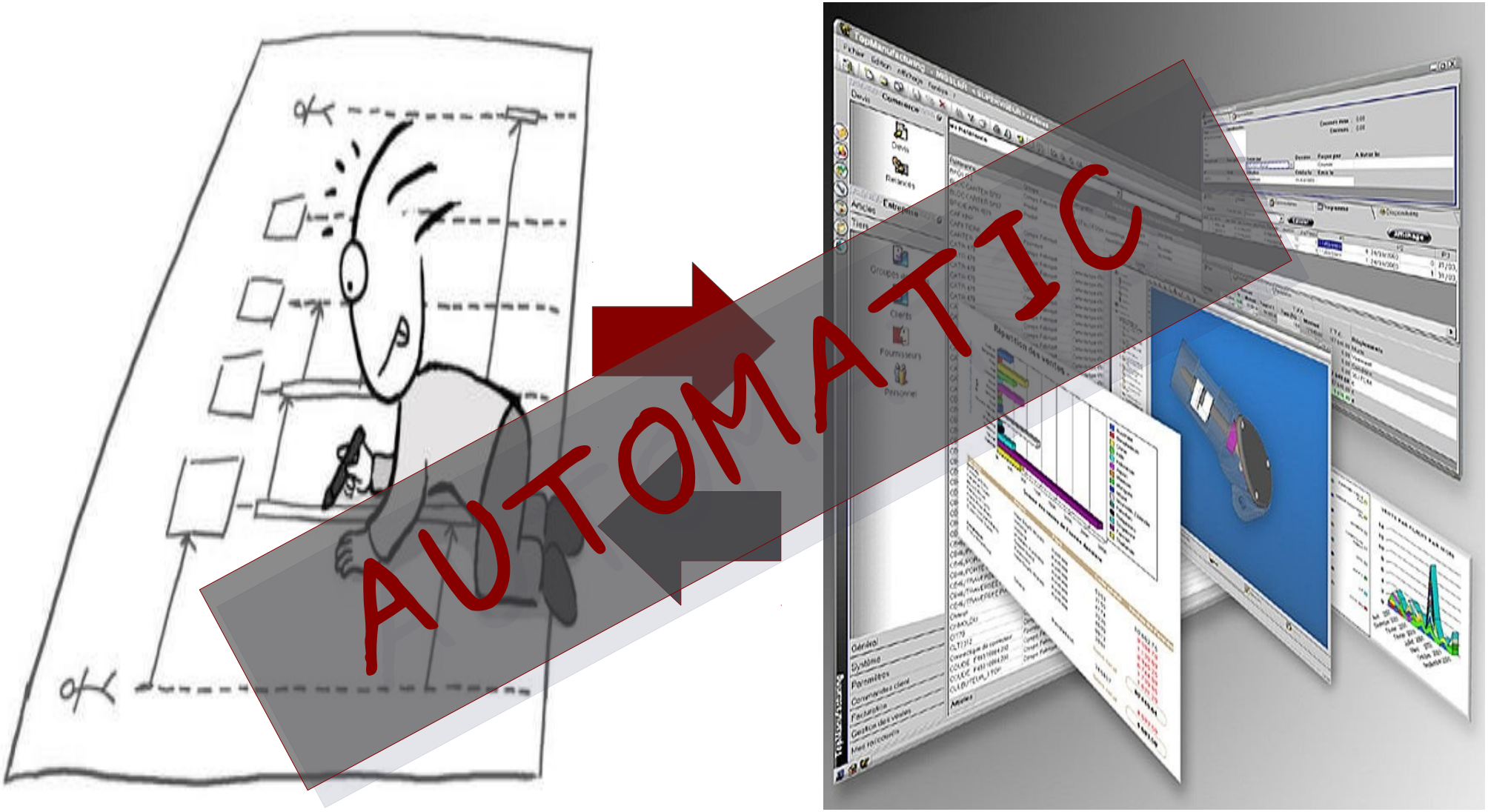


IdS && modelli : rappresentare, analizzare, sintetizzare





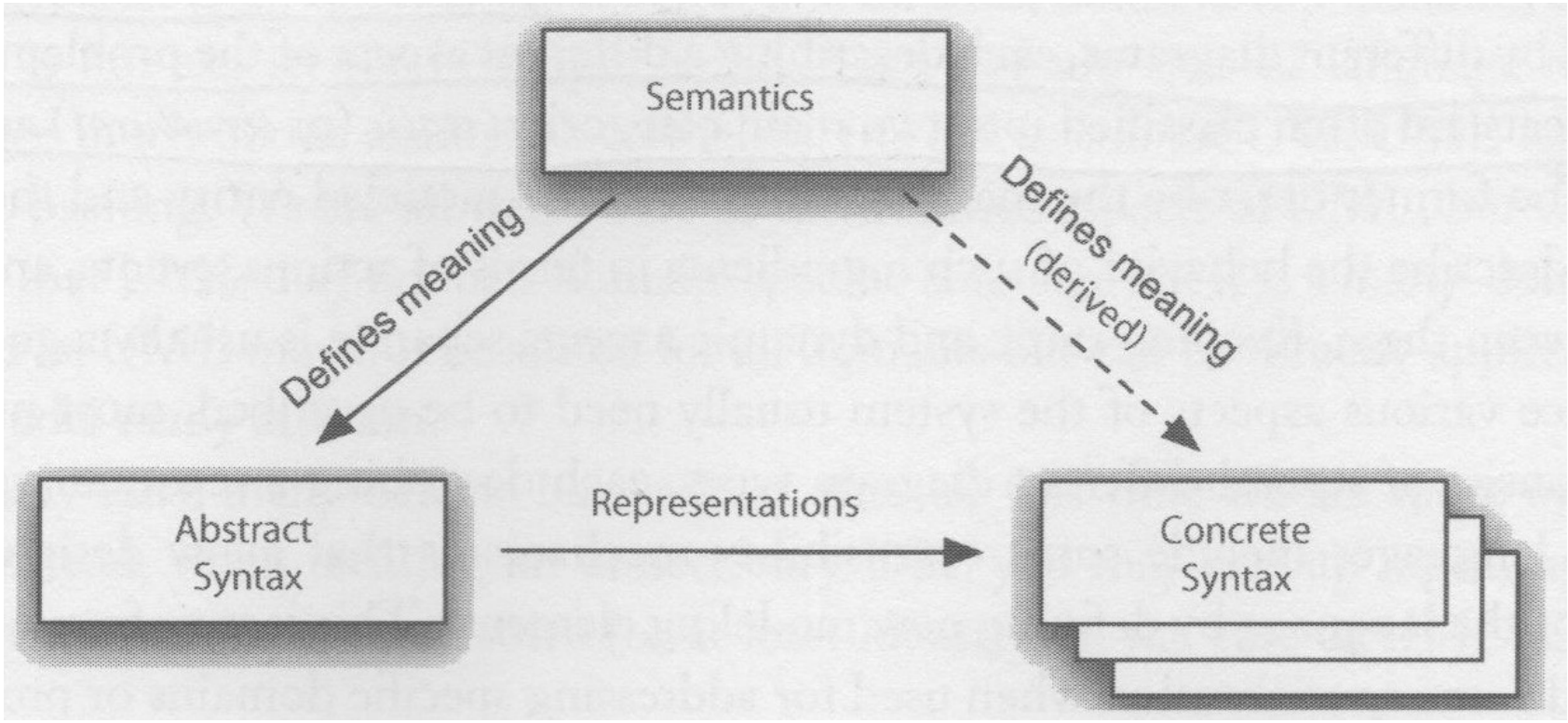
IdS && modelli : automazione



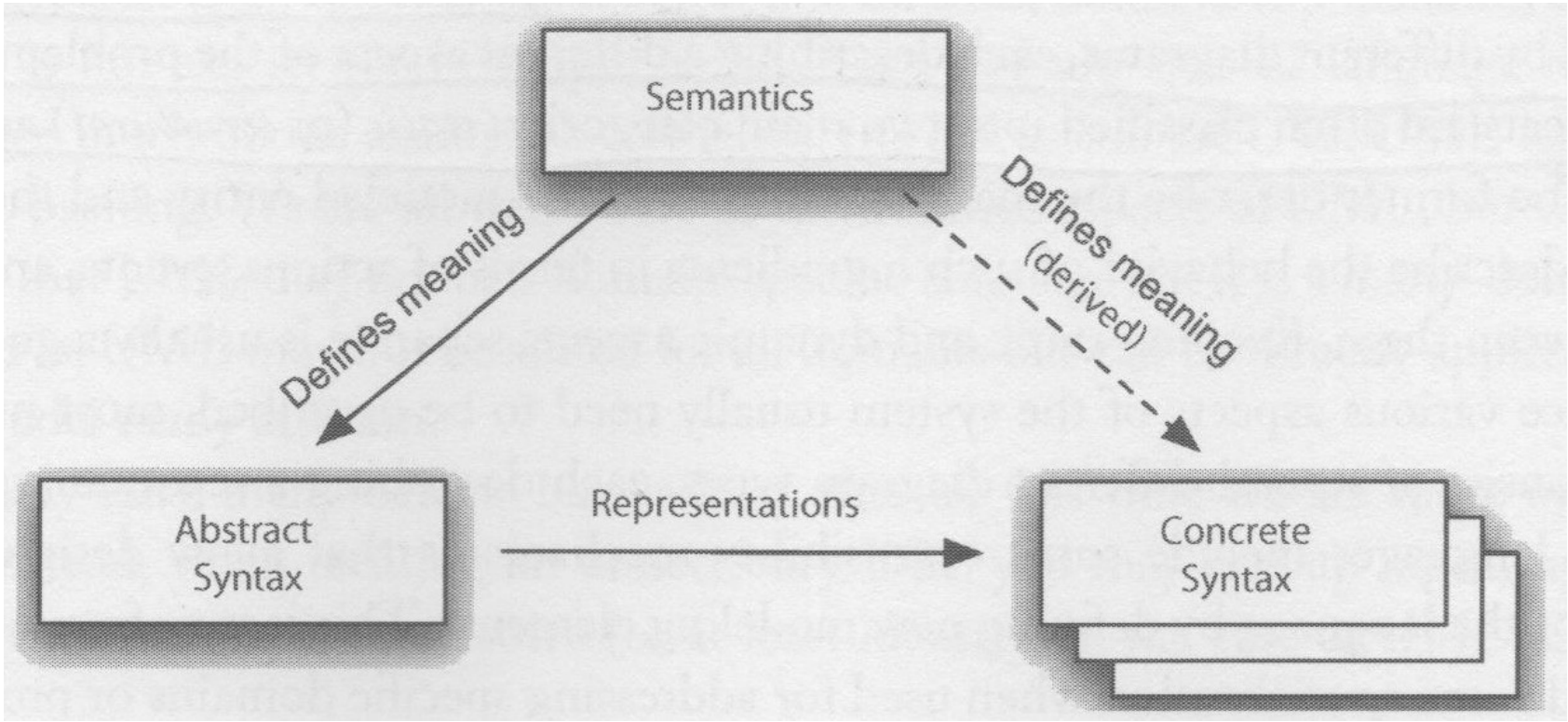
- come manipolare in modo automatico modelli software?
  - … …
  - … …

anatomia dei linguaggi di modellazione

anatomia dei linguaggi di modellazione

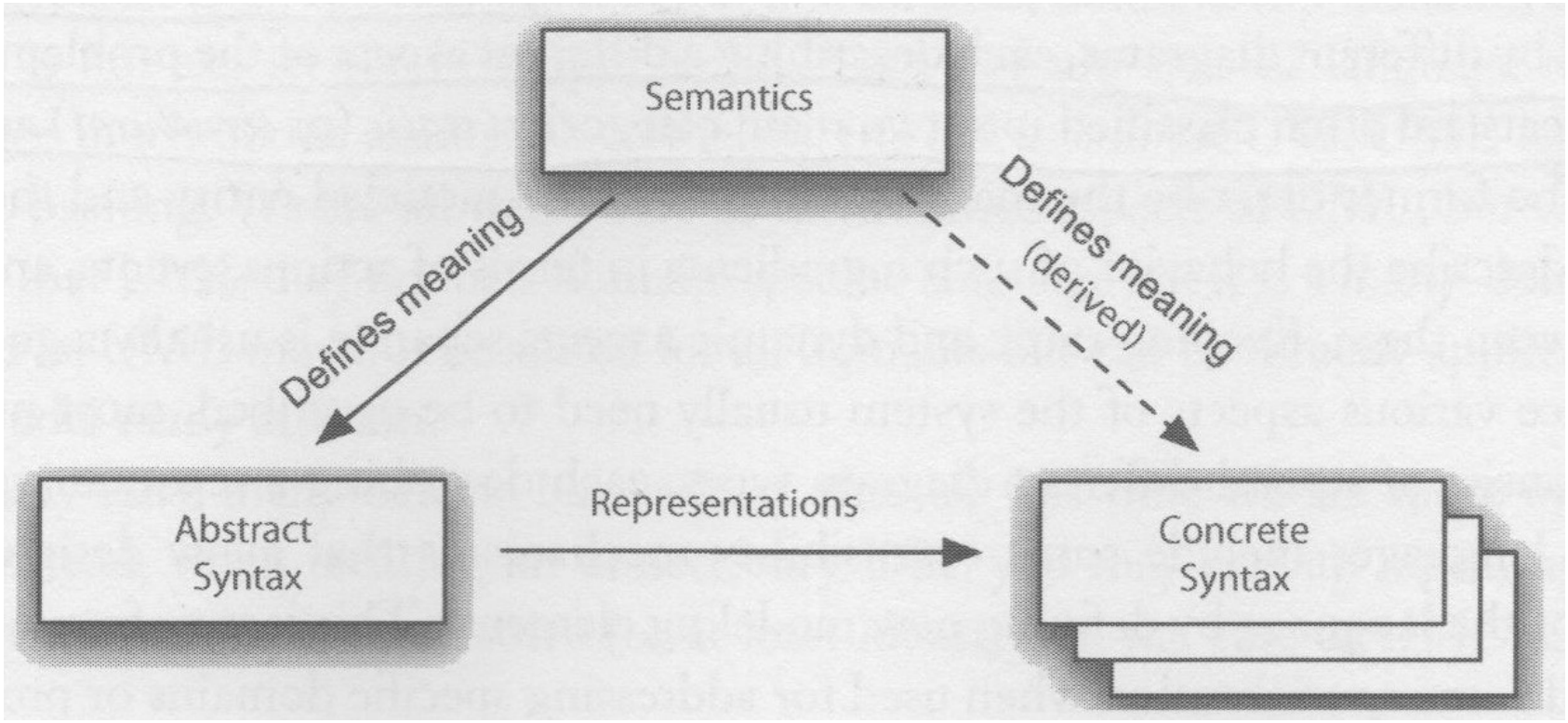


anatomia dei linguaggi di modellazione



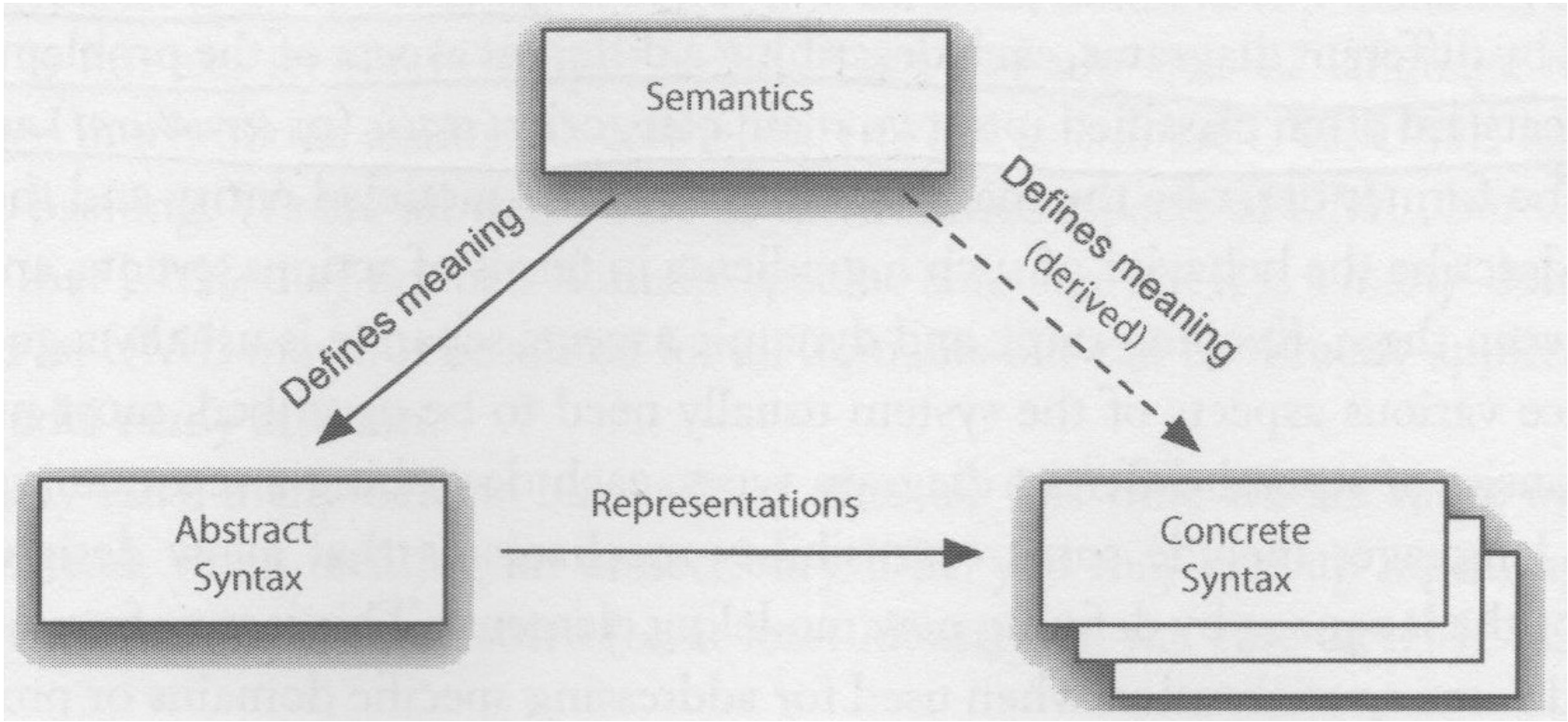
- abstract syntax: descrive la struttura del linguaggio, ed il modo in cui le primitive possono essere combinate tra loro. E' indipendente da ogni tipo di rappresentazione/ codifica

anatomia dei linguaggi di modellazione



- concrete syntax: descrive la specifica rappresentazione di un linguaggio; la notazione può essere sia testuale che grafica. E' utilizzata dai progettisti per creare modelli.

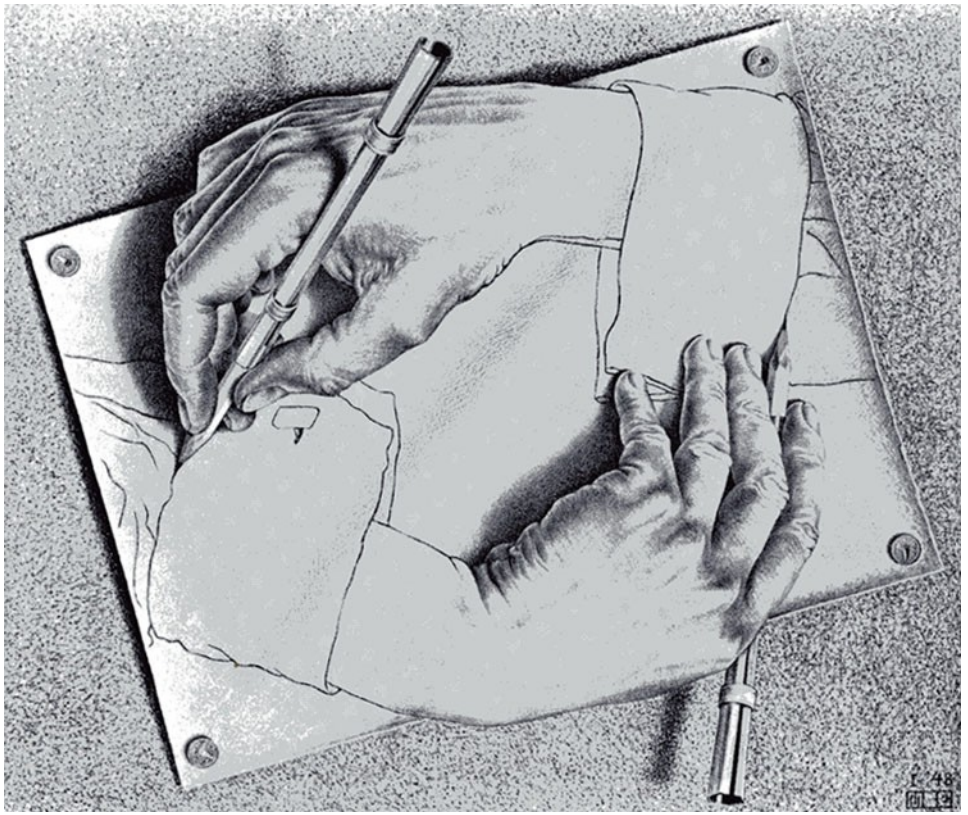
anatomia dei linguaggi di modellazione

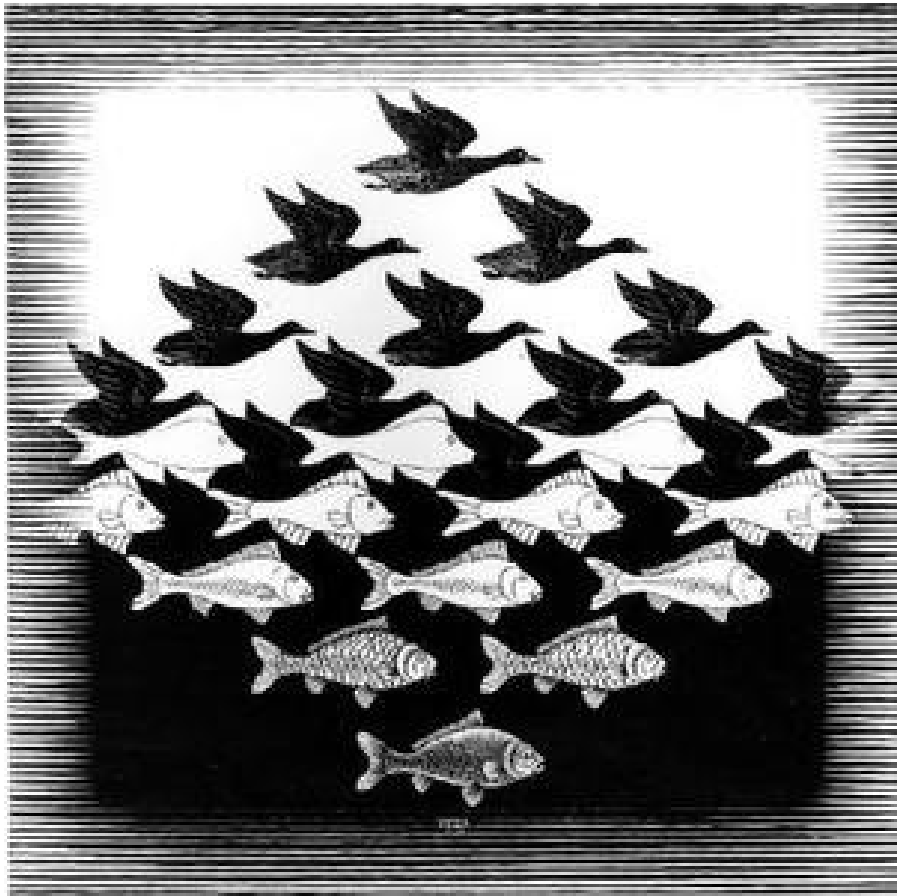


- semantics: definisce il significato di ogni elemento del linguaggio. Può essere una definizione sia formale che semi-formale. Una parziale o scorretta specifica della semantica causa incomprensioni ed usi scorretti del linguaggio.

- come manipolare in modo automatico modelli software?
  - … …
  - … …

- come manipolare in modo automatico modelli software?
  - meta-modellazione

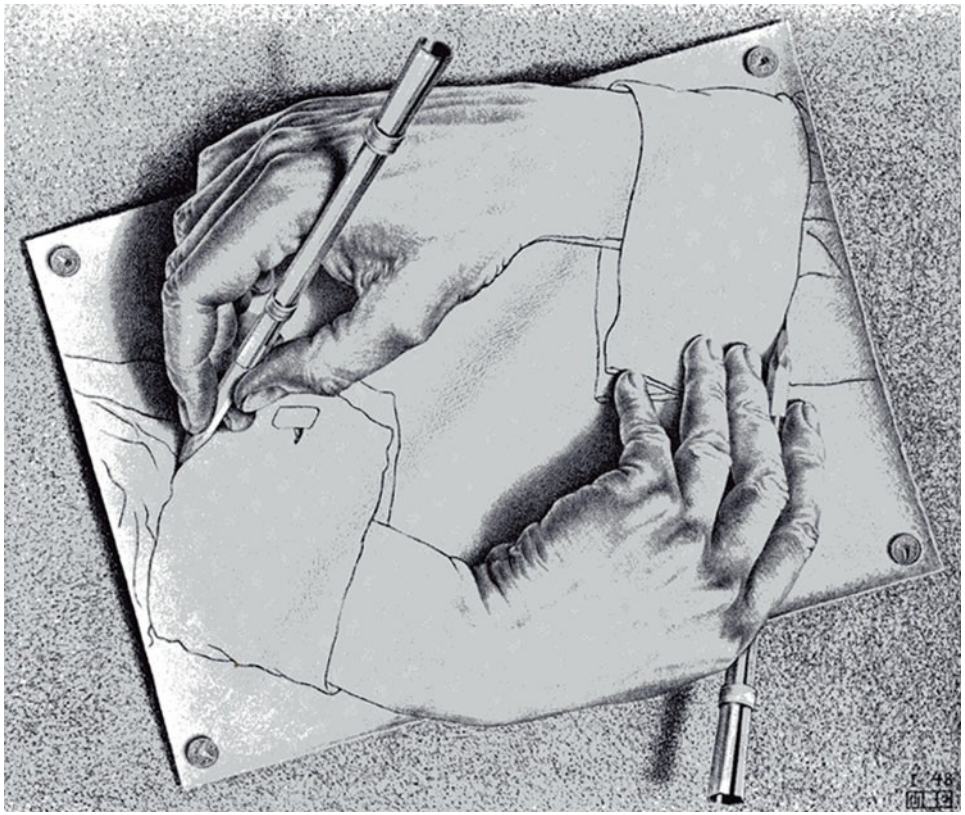


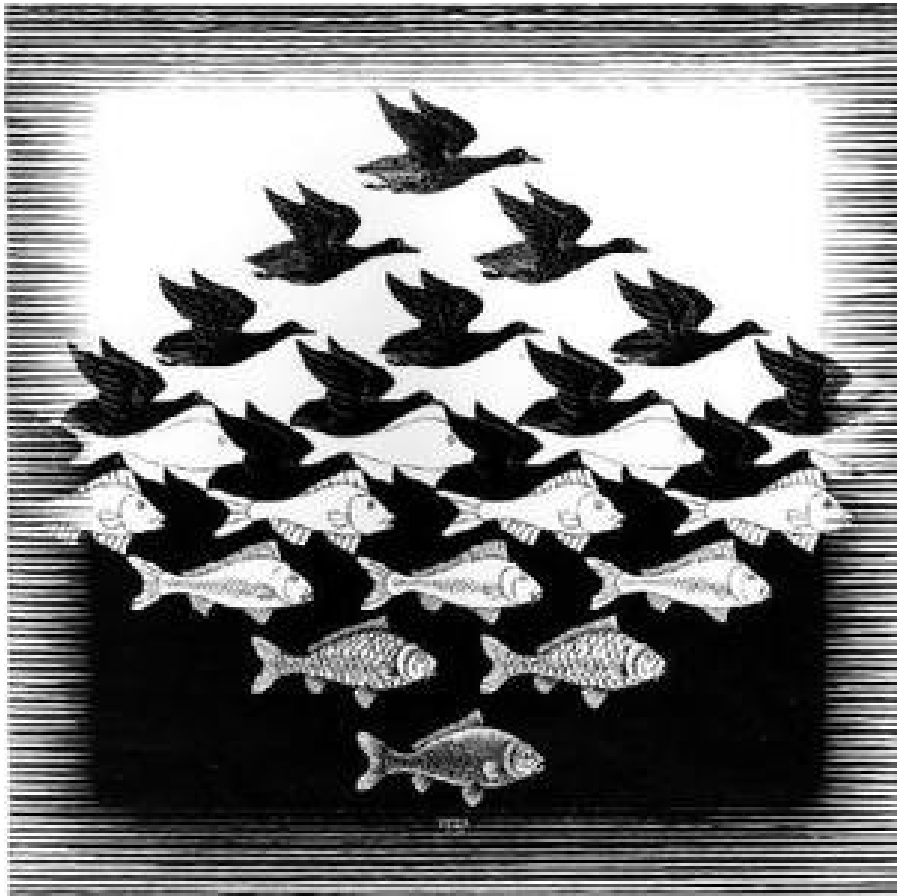


  - trasformazioni (automatiche)

- come manipolare in modo automatico modelli software?

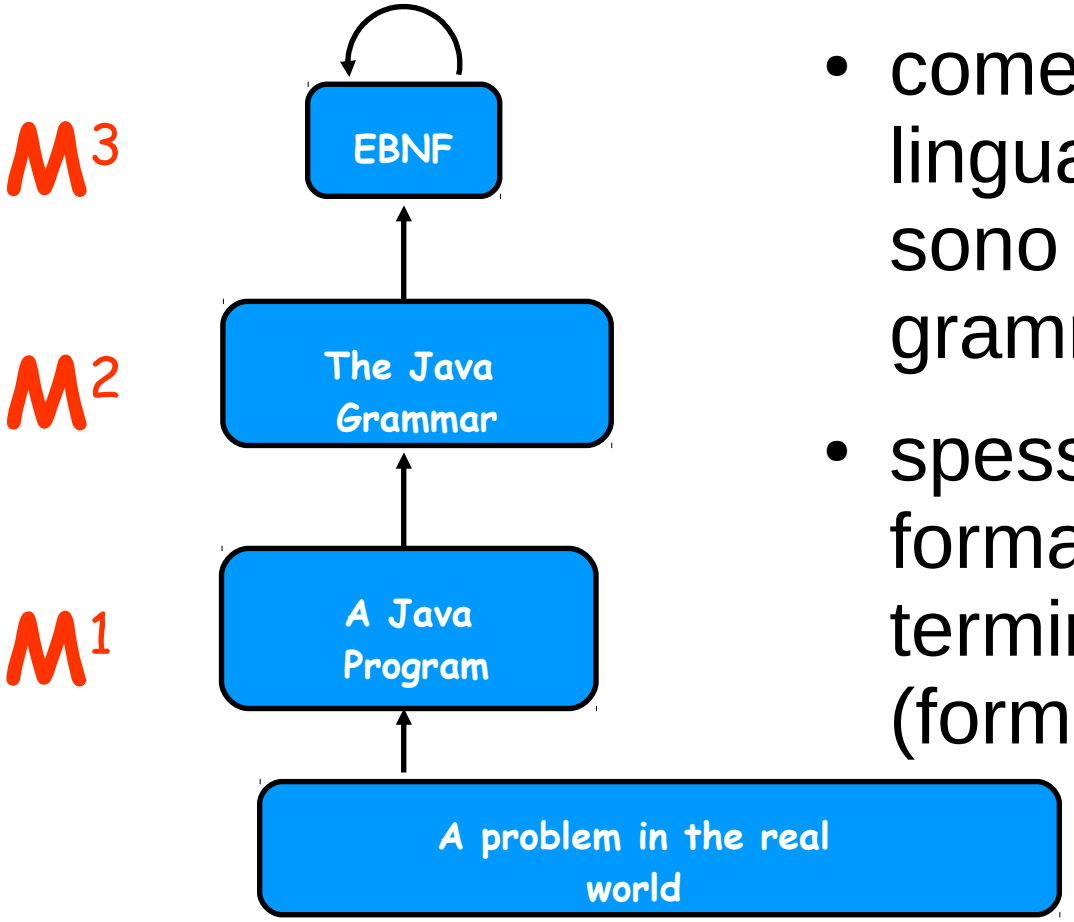
  - meta-modellazione





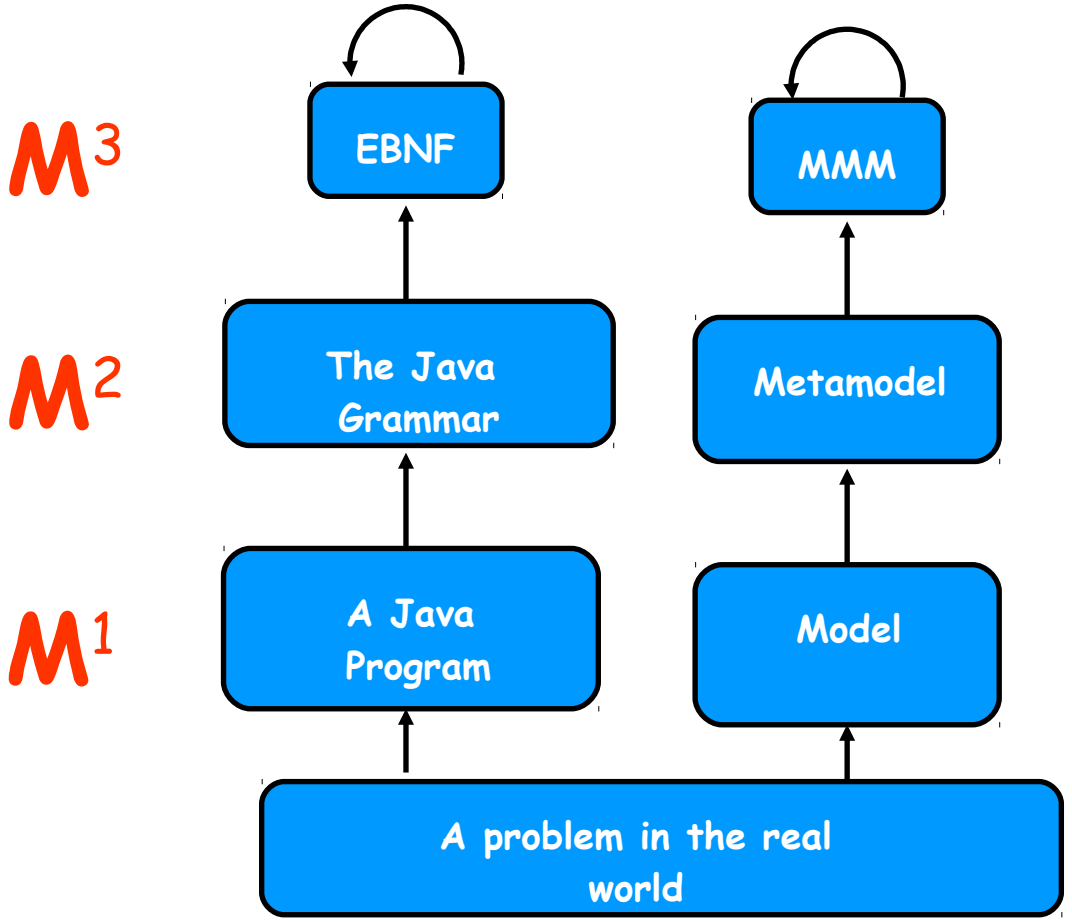
  - trasformazioni (automatiche)

dalla teoria dei linguaggi … - 1 -



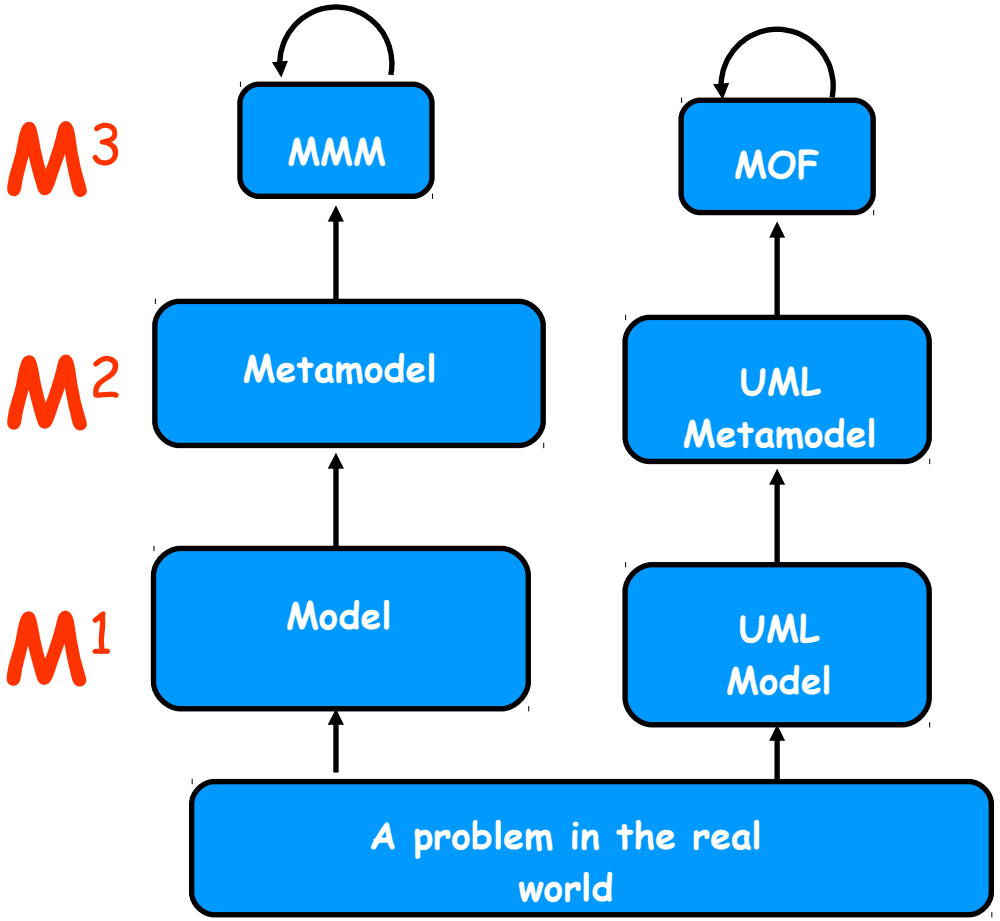
- come in linguaggi naturali, i linguaggi di programmazione sono definiti attraverso grammatiche formali
- spesso, queste grammatiche formali sono espresse in termini di altri linguaggi (formali)

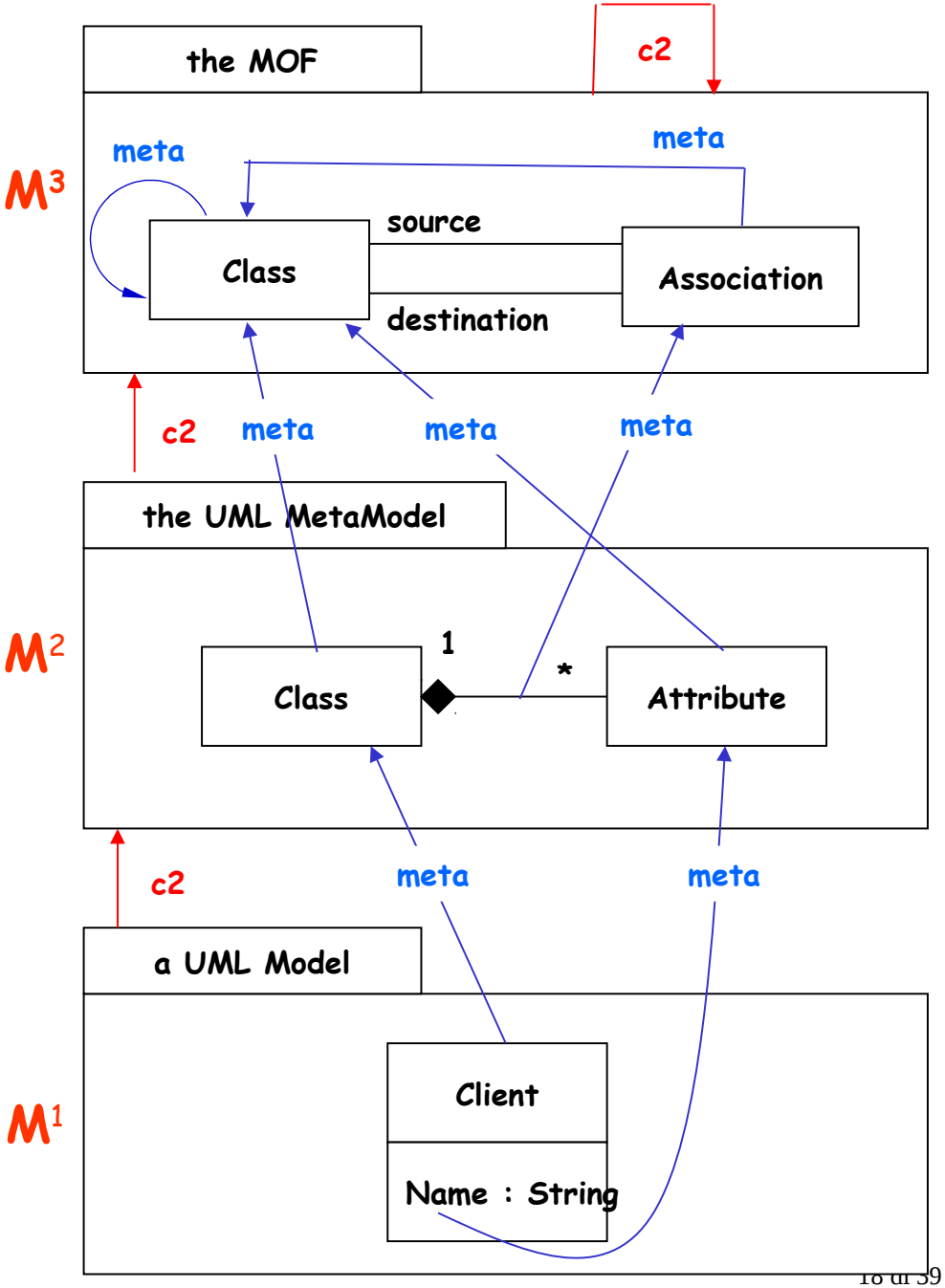
… la stessa cosa avviene per i linguaggi di modellazione



- i modelli sono conformi alla “grammatica” del linguaggio (i.e. metamodello)
- il metamodello dichiara (almeno) la sintassi astratta del linguaggio di modellazione
- i metamodelli sono definiti per mezzo di altri linguaggi chiamati : meta-metamodelli

metamodeling

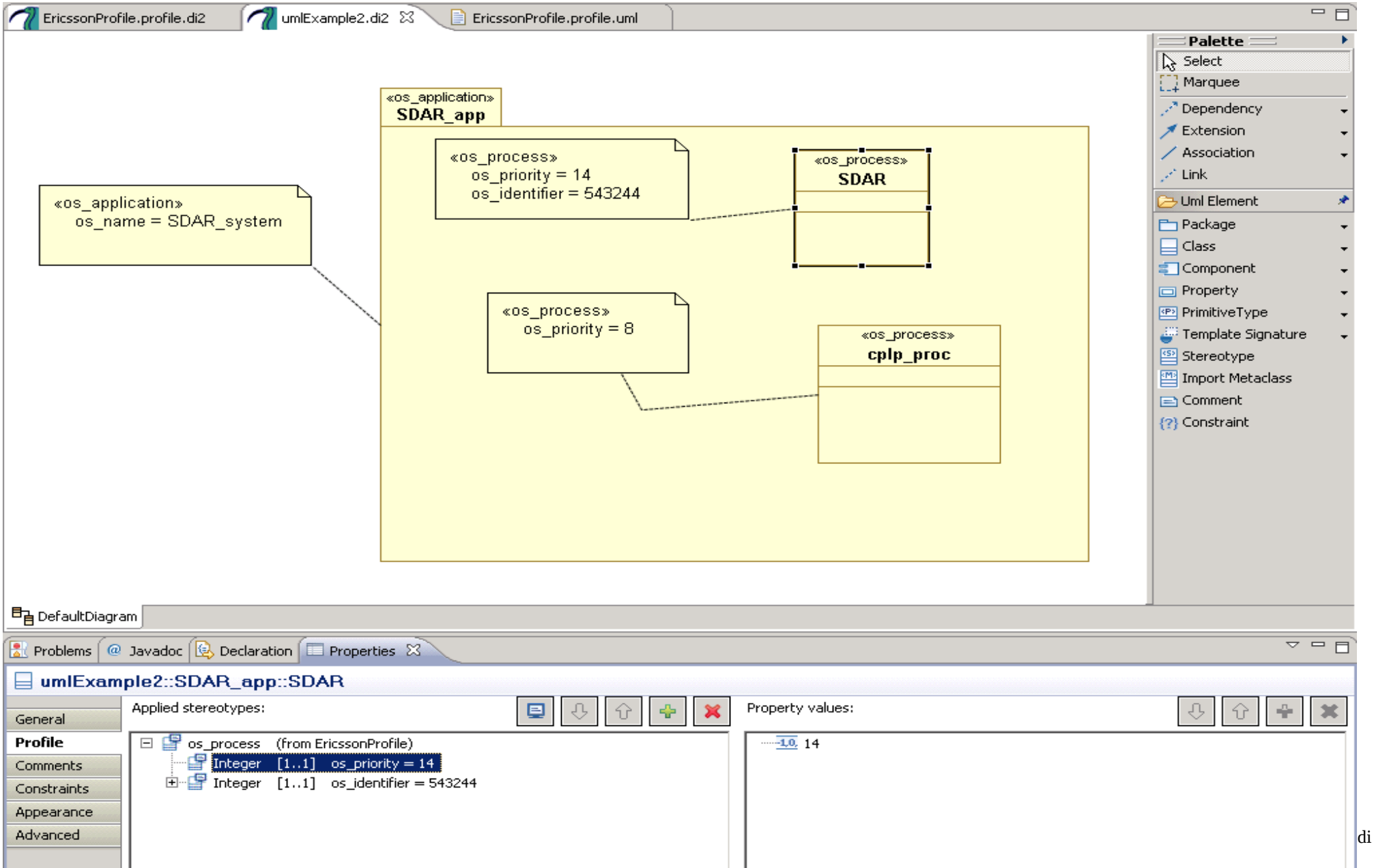




disegnare VS modellare

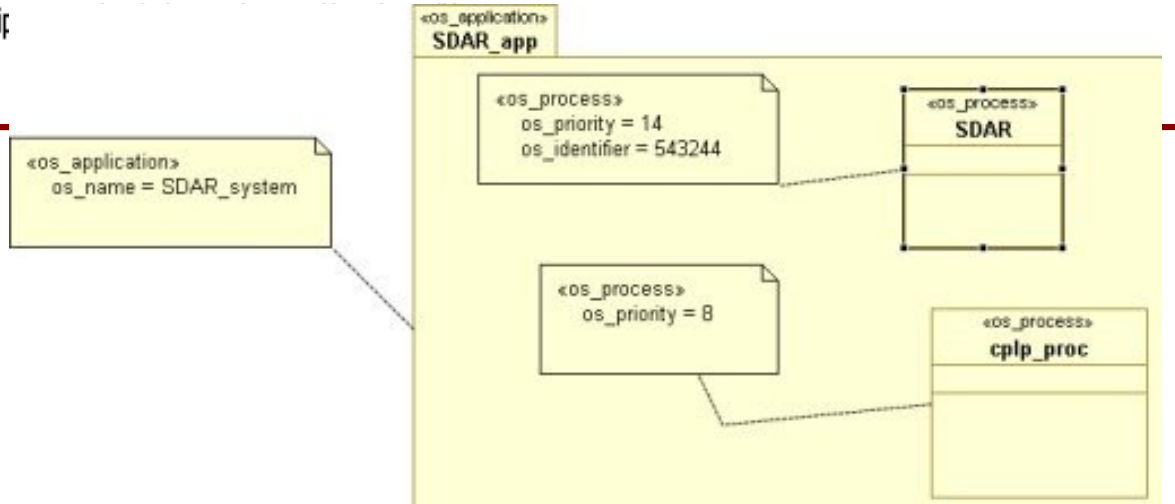
- strumenti per disegnare
  - non riferiscono ad alcuna grammatica per i modelli trattati
  - non è associato nessun significato agli elementi modellati
  - gestiscono la rappresentazione grafica degli elementi del modello
    - forme, linee, frecce
    - non c'è controllo sulle relazioni tra gli elementi modellati
- strumenti per modellare
  - rappresentano elementi secondo una grammatica
  - definiscono relazioni tra elementi solo se previste dalla grammatica

drawing VS modeling – example : graphic notation



drawing VS modeling – example : textual notation

```
<contents xmi:type="ecore:EPackage" xmi:id="_jxPwEZ3MEdyEi-YIP6R6ew" name="EricssonProfile" nsURI="http:///schemas/EricssonProfile/_jxPwEJ3MEdyEi-YIP6R6
 <eClassifiers xmi:type="ecore:EClass" xmi:id="_jxPwEp3MEdyEi-YIP6R6ew" name="os_process">
  <eAnnotations xmi:id="_jxPwE53MEdyEi-YIP6R6ew" source="http://www.eclipse.org/umi2/2.0.0/UML" references="_ilN3QJ3LEdyEi-YIP6R6ew"/>
  <eStructuralFeatures xmi:type="ecore:EAttribute" xmi:id="_jxPwFJ3MEdyEi-YIP6R6ew" name="os_priority" ordered="false" unique="false" lowerBound="1">
   <eType xmi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eStructuralFeatures>
  <eStructuralFeatures xmi:type="ecore:EReference" xmi:id="_jxPwFp3MEdyEi-YIP6R6ew" name="base_Class" ordered="false" unique="false" lowerBound="1">
   <eType xmi:type="ecore:EClass" href="http://www.eclipse.org/uml2/2.1.0/UML#//Class"/>
  </eStructuralFeatures>
  <eStructuralFeatures xmi:type="ecore:EAttribute" xmi:id="_jxPwGJ3MEdyEi-YIP6R6ew" name="os_identifier" ordered="false" unique="false" lowerBound="1">
   <eType xmi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eStructuralFeatures>
 </eClassifiers>
 <eClassifiers xmi:type="ecore:EClass" xmi:id="_jxPwGp3MEdyEi-YIP6R6ew" name="os_application">
  <eAnnotations xmi:id="_jxPwG53MEdyEi-YIP6R6ew" source="http://www.eclipse.org/uml2/2.0.0/UML" references="_i_cVIJ3LEdyEi-YIP6R6ew"/>
  <eStructuralFeatures xmi:type="ecore:EAttribute" xmi:id="_jxPwHJ3MEdyEi-YIP6R6ew" name="os_name" ordered="false" unique="false" lowerBound="1">
   <eType xmi:type="ecore:EDataType" href="http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eStructuralFeatures>
  <eStructuralFeatures xmi:type="ecore:EReference" xmi:id="_jxPwHp3MEdyEi-YIP6R6ew" name="base_Package" ordered="false" unique="false" lowerBound="1">
   <eType xmi:type="ecore:EClass" href="http://www.eclip
  </eStructuralFeatures>
 </eClassifiers>
</contents>
```



- è possibile sfruttare queste nozioni nei linguaggi di modellazione/programmazione

- come?

reflection

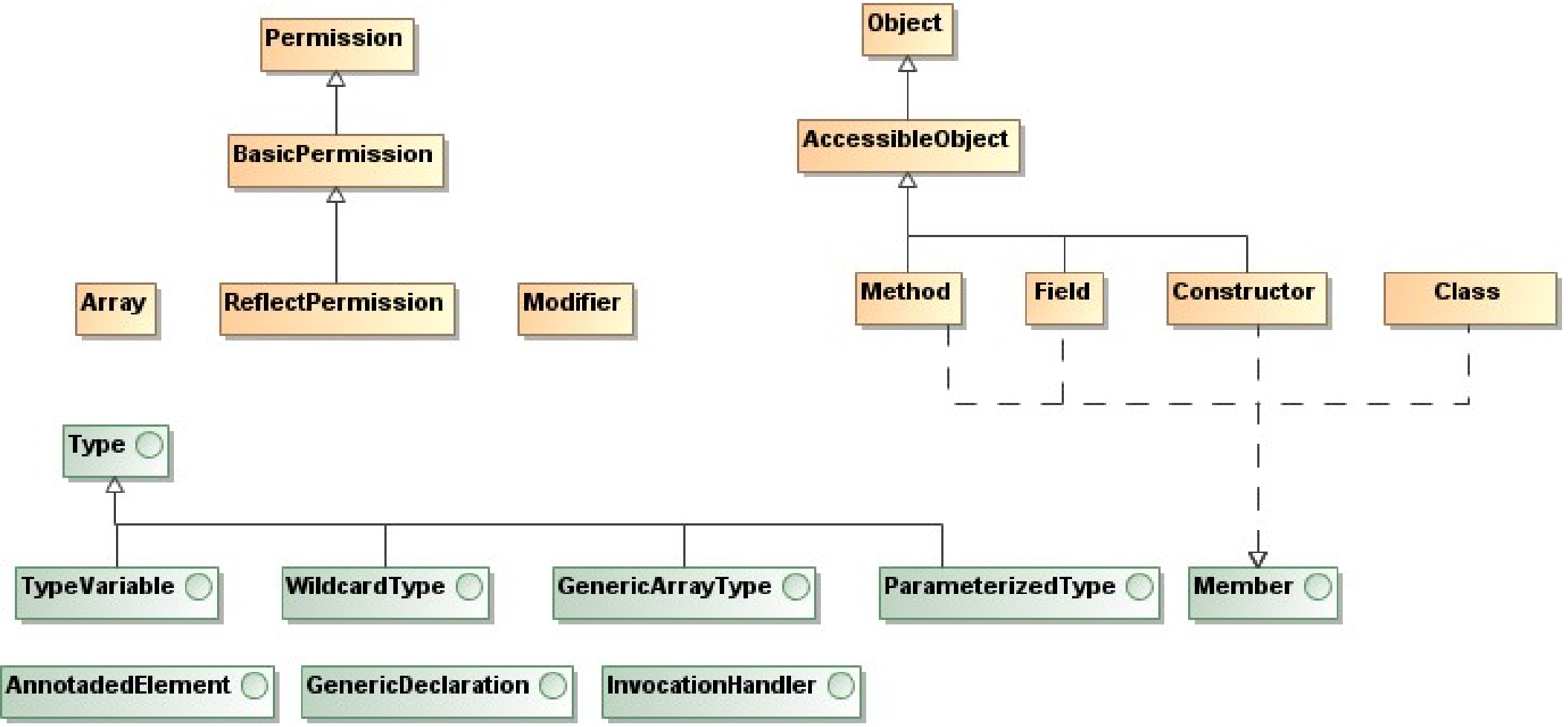
- meccanismo abilitato a tempo di esecuzione che consente :
  - analizzare la struttura di una classe
  - richiedere la creazione dinamica di oggetti
  - richiedere dinamicamente l'invocazione di metodi

come si sfrutta in Java?

la *classe* Object

- IN JAVA TUTTO (o quasi) è un OGGETTO!!
- Object è una classe
  - è il tipo base di ogni classe definita in Java
- se una classe non estende esplicitamente da un'altra classe, allora implicitamente estende Object
  - non c'è bisogno di “extends Object”
  - quindi ogni classe “is-a-kind-of” Object
- Object offre metodi basilari che sono ereditati da tutte le classi
  - `boolean equals(Object other); /* checks whether two object are the same from a semantics standpoint (to check if they are exactly the same object in memory, use == */`
  - `Object clone(); // instantiates a new object with the same values`
  - `void finalize(); // protected, cleans up memory and resources`
  - `int hashcode(); // unique (almost) identifier of the object`
  - `String toString(); // returns the class name + object hash code`

reflection model in Java



la *classe* Class – 1

- “istanze” della classe Class rappresentano classi ed interfacce in esecuzione in una applicazione Java
  - una enumerazione Java (i.e. enum) è rappresentato da un oggetto di tipo Class
  - una annotazione Java (i.e. @) è rappresentato da un oggetto di tipo Class
  - un array è rappresentato da un oggetto di tipo Class (una istanza per coppia [tipo-array,dimensione])
  - i tipi primitivi di Java (boolean, byte, char, short, int, long, float, and double), e la keyword “void” are rappresentato da oggetti di tipo Class

la *classe* Class – 2

- Class non definisce un costruttore pubblico:
  - i.e. non si può fare la “new” si un “Class”
- gli oggetti di tipo Class sono instanziati automaticamente dalla JVM e caricati in memoria
  - su questo aspetto ci sono alcune controversie che contribuiscono a non far considerare Java un linguaggio puramente orientato agli oggetti

esempi reflection – 1

VEDERE MATERIALE ALLEGATO ALLA LEZIONE :

ReflectionExample.java