Über die Semantik von Modellierungssprachen

und des UML-Standards

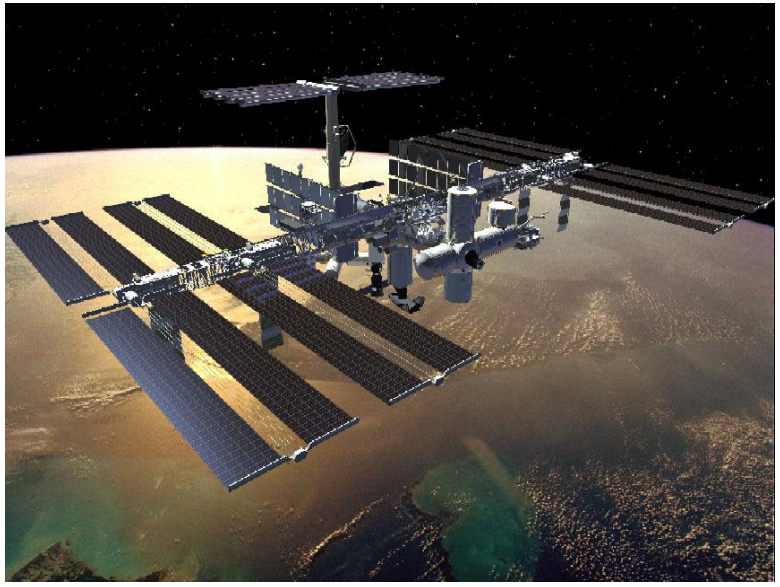
Prof. Dr. Bernhard Rumpe
Software Systems Engineering
Technische Universität Braunschweig

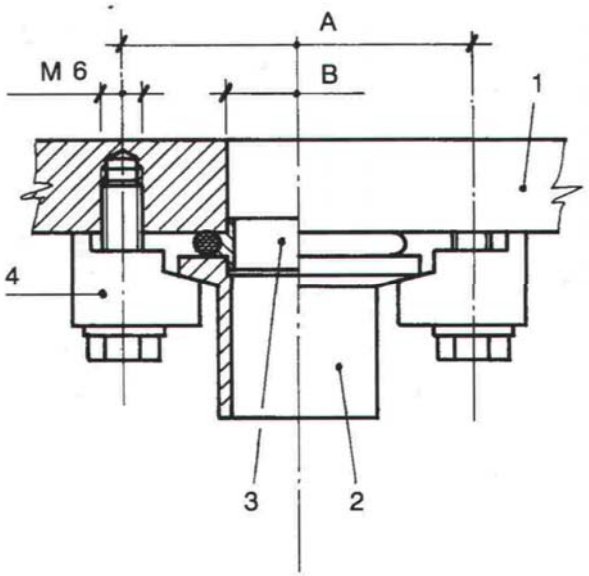
http://www.sse.cs.tu-bs.de/

What is a model?

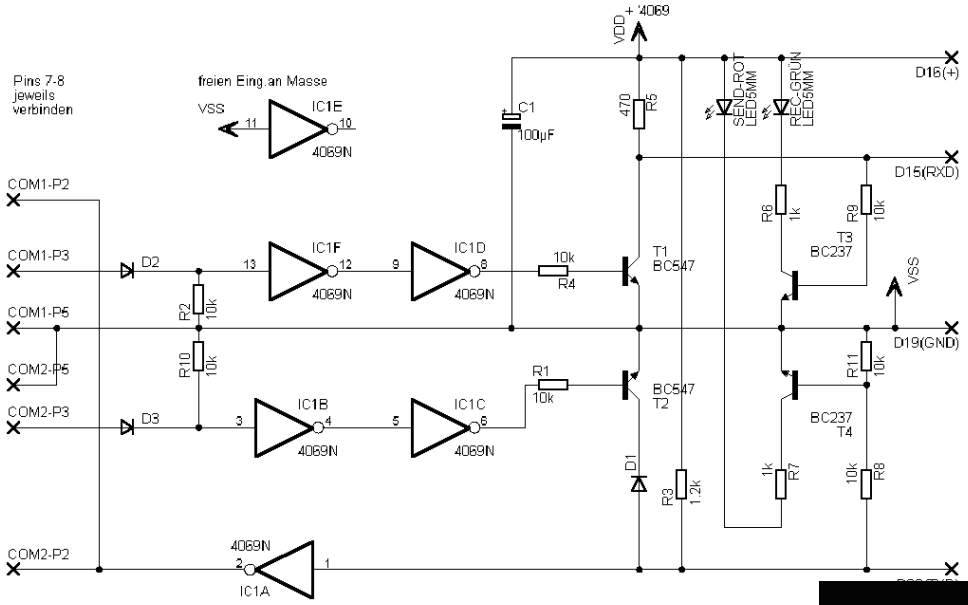
And why do we need modeling languages anyway?

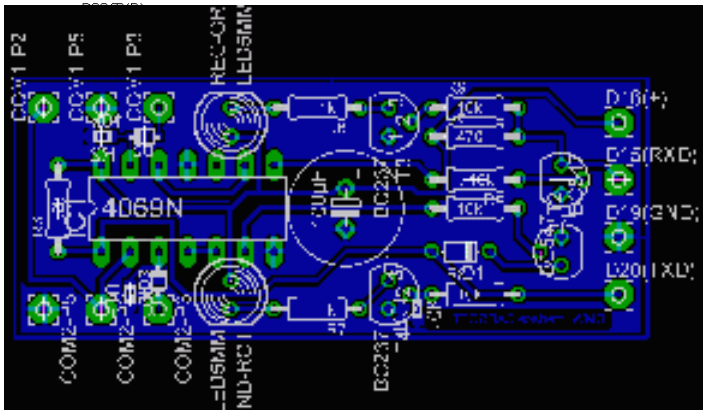
Mechanical Engineering: Models of Machines in ISO-Norms



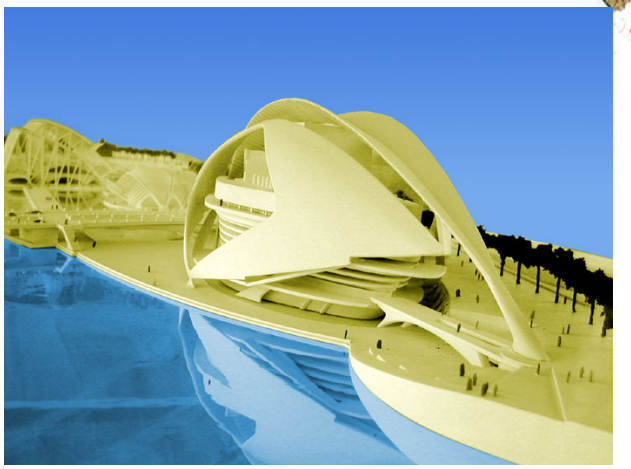


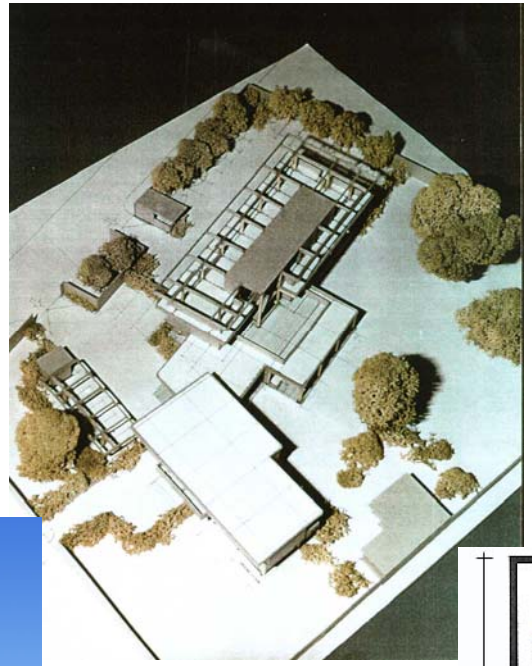
Electrical Engineering: Switches in ISO-Norms

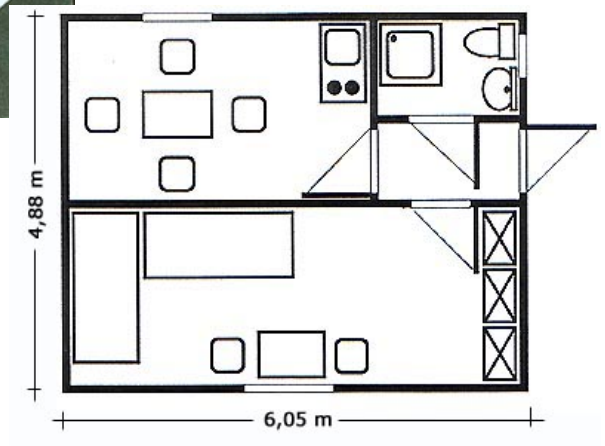




Architecture

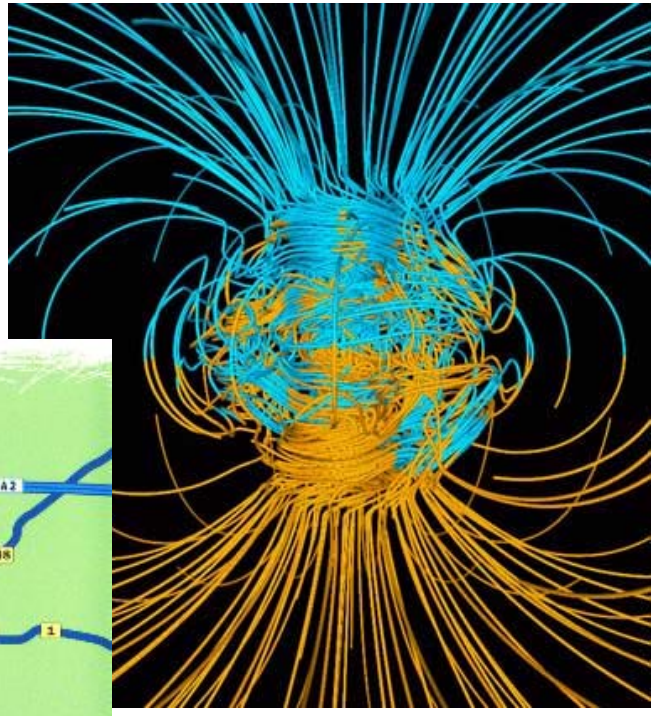


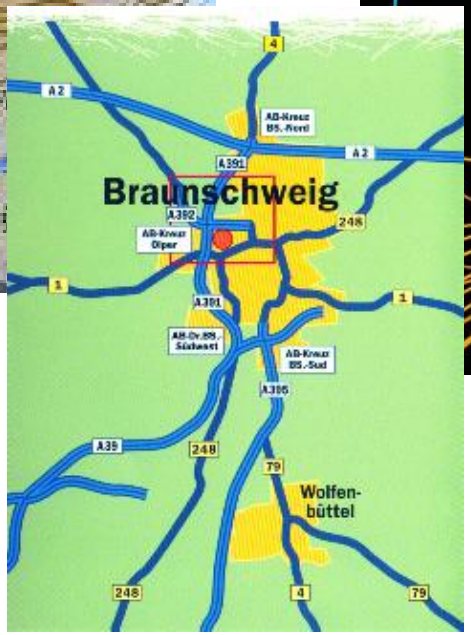




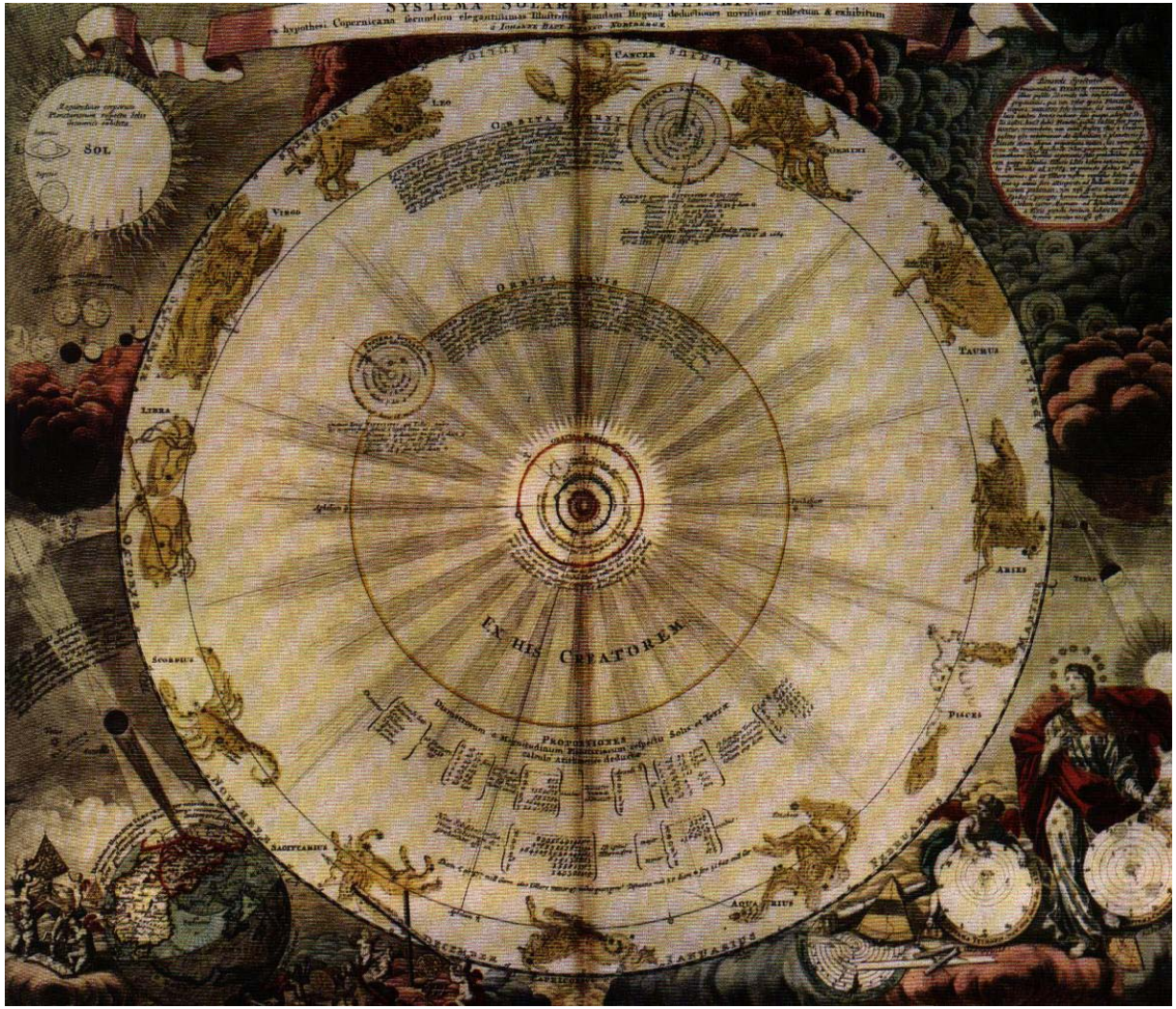
Geography


Windberg
nach Dresden



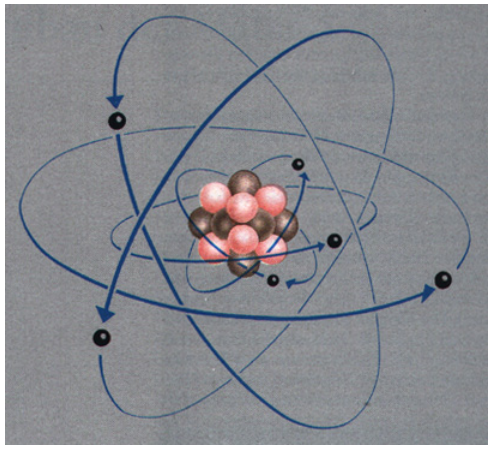

Braunschweig
Wolfen-
büttel
A2
A391
A392
A39
A395
248
1
4
79

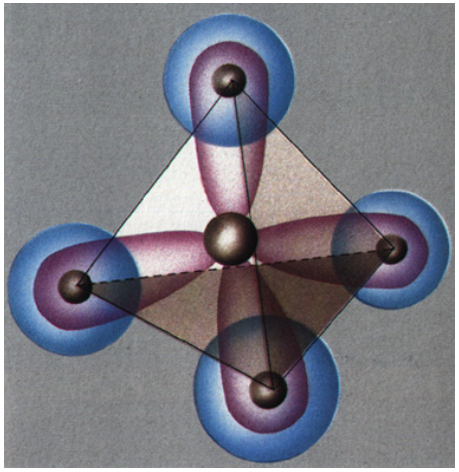
Astronomy: Geocentric model from Kopernikus



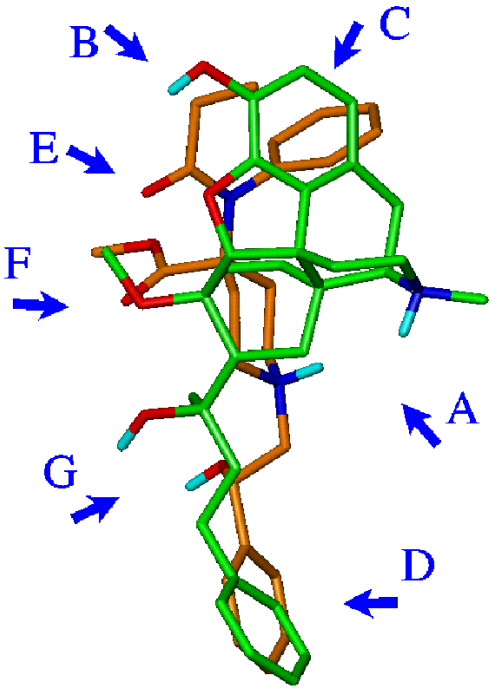
Physics

- Rutherford‘s and Bohr‘s atomic models
- Einstein‘s theory of relativity
- Model of Big Bang
- ...

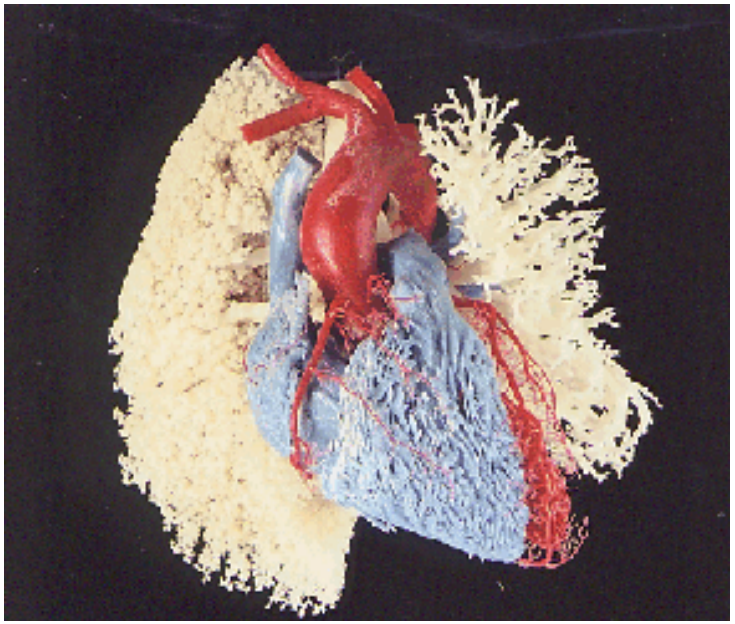




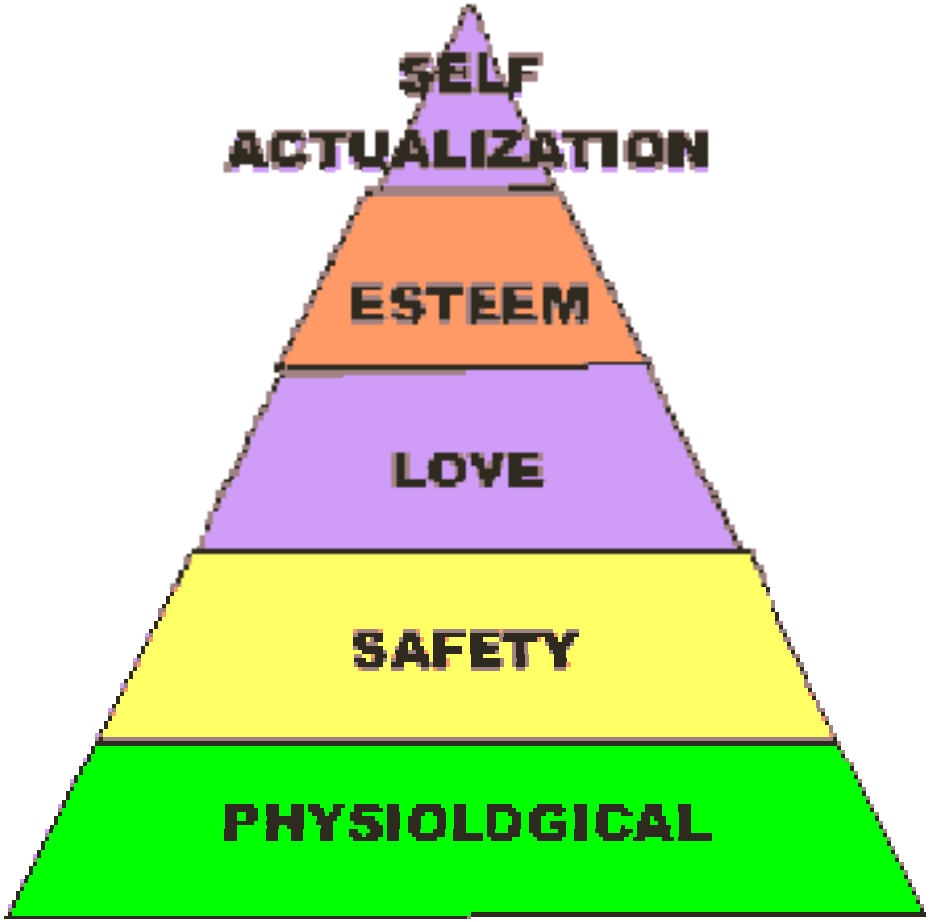
Biology:
Animals, molecules, their interaction ...



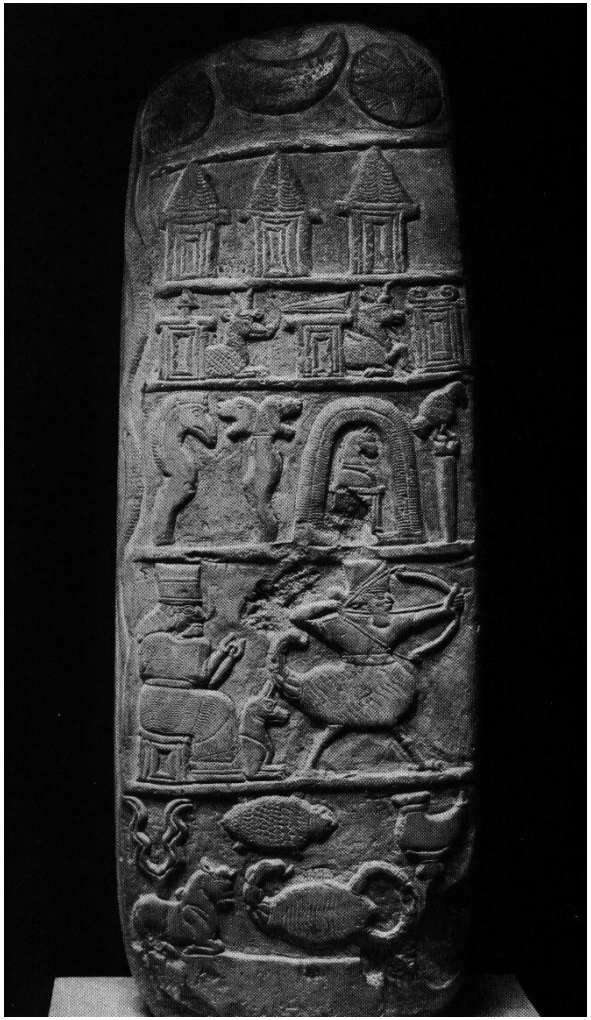
Medicine, safety engineering



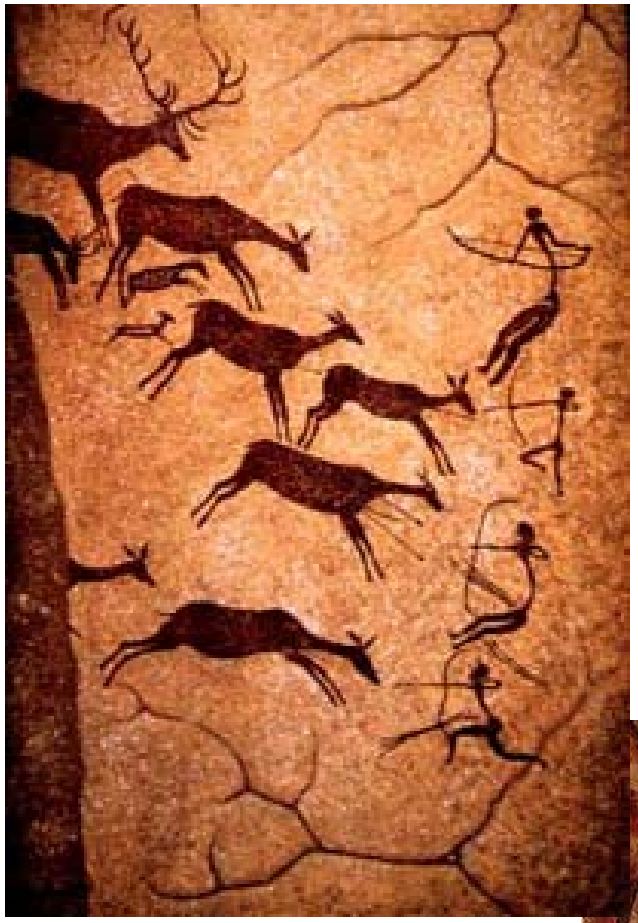
Sociology: Maslow‘s hierarchy of needs

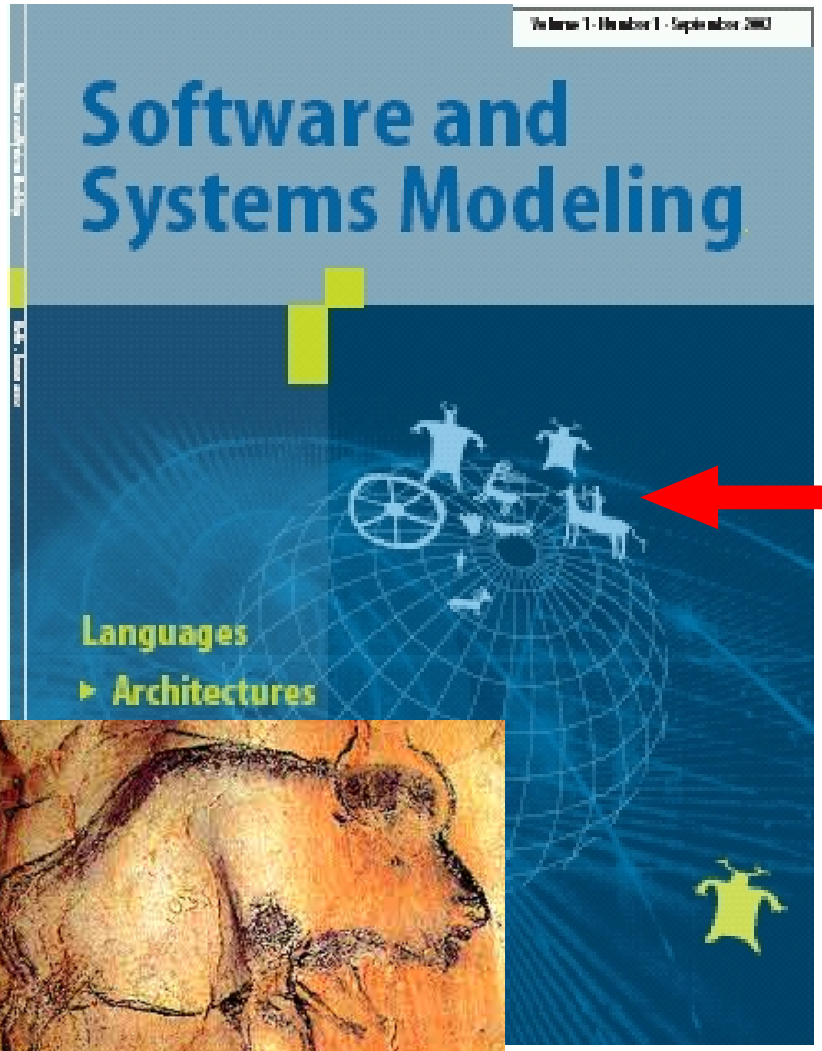


The first models: Hieroglyphs, early „languages“

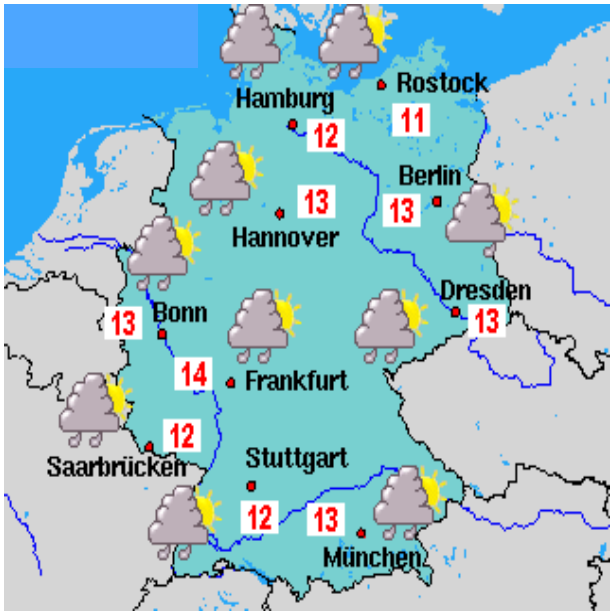


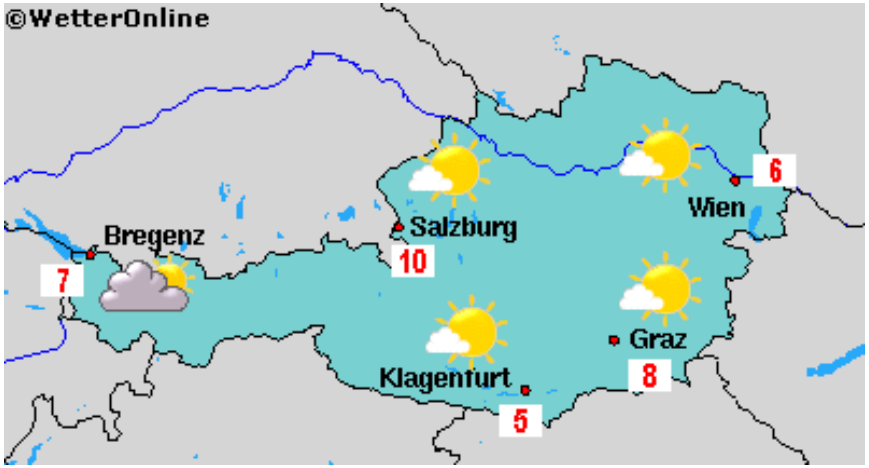
The really first (still existing) models: cave drawings





Our daily live: weather charts





Der Modellbegriff

Ein Modell ist seinem Wesen nach eine in Maßstab, Detailliertheit und/oder Funktionalität verkürzte beziehungsweise abstrahierte Darstellung des originalen Systems.

1. Es gibt ein Original
2. Abstraktion ist essentiell
3. Modelle werden mit einem Ziel erstellt und verwendet, um Eigenschaften des Originals zu studieren

(Stachowiak 1973)

Der Modellbegriff

Ein Modell ist seinem Wesen nach eine in Maßstab, Detailliertheit und/oder Funktionalität verkürzte beziehungsweise abstrahierte Darstellung des originalen Systems.

1. Es gibt ein Original
2. Abstraktion ist essentiell
3. Modelle werden mit einem Ziel erstellt und verwendet, um Eigenschaften des Originals zu studieren

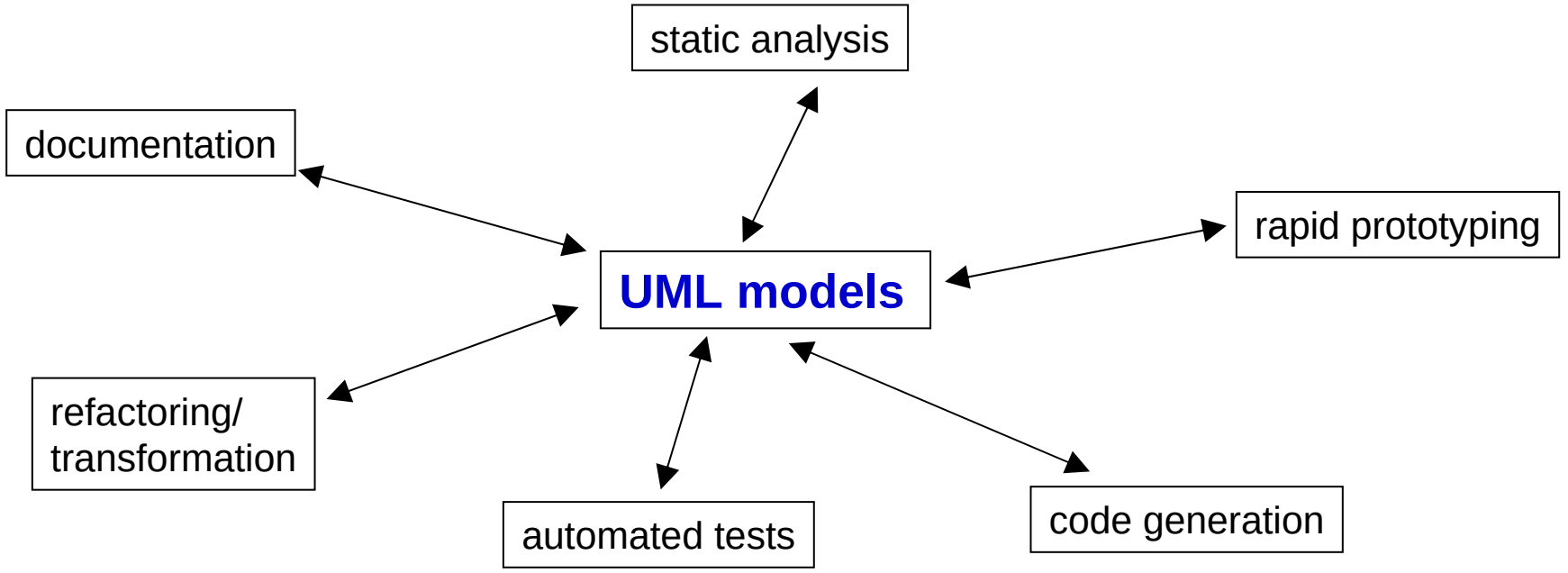
(Stachowiak 1973)

Die Softwaretechnik verwendet Modelle

- präskriptiv: Das Modell existiert vor dem Original (Softwaresystem)
- und oft konstruktiv: Das Original wird aus dem Modell generiert.

Model based development with UML

- Models as a central notation in the development process



- UML serves as central notation for development of software
- UML is programming, test and modelling language at the same time

Reasons for an explicit modeling language

- Many kinds of models do not need an explicit „language“ (let alone a formally defined one)
- Software engineering does, because:
  - Models have many kinds of uses in the development process (code generation etc.)
  - Software is an immaterial product; it is difficult to model
  - Relationships between model and original are closer that usual
    - Original as its own model
    - Model configures the original
    - Interpreters uses the „model“ directly
    - Presence of the model does affect the product!
  - Software is complex
    - It has views, therefore needs different kinds of models and they need to fit together

Models in Software Engineering

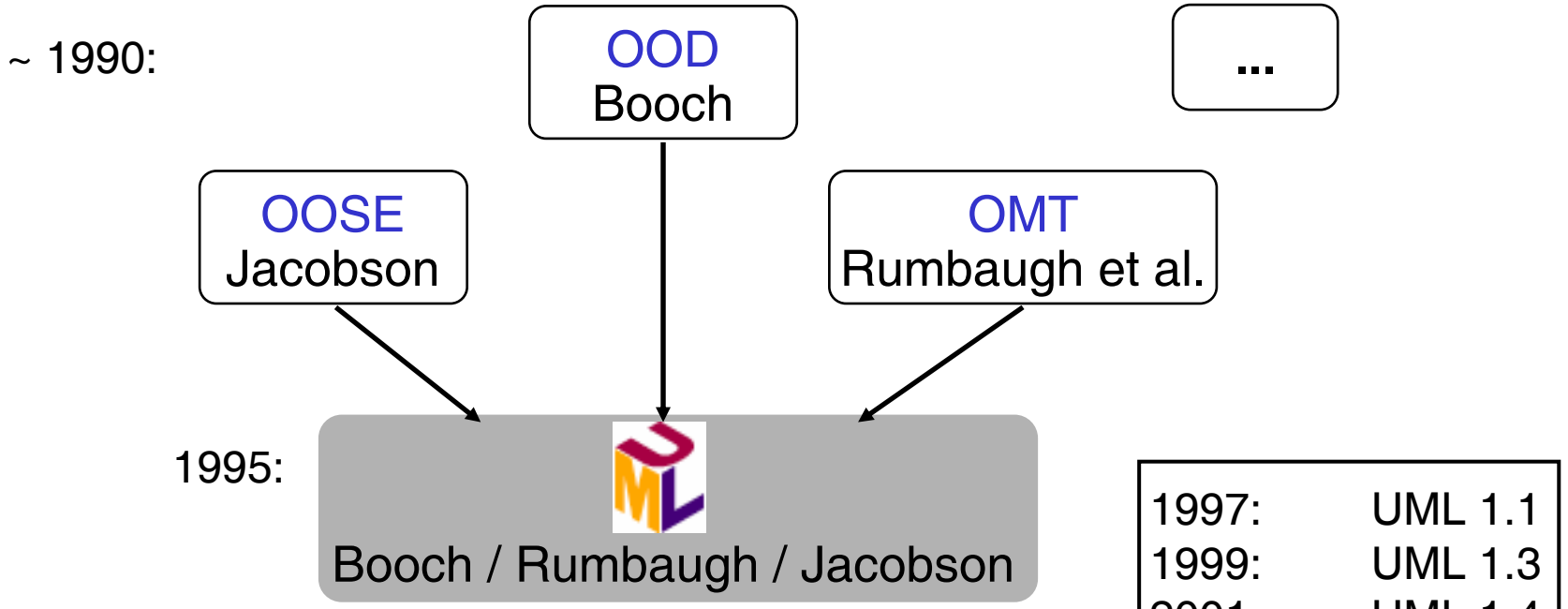
- Industry standard: Unified Modeling Language
  - 13 kinds of diagrams (class diagrams, Statecharts etc.)

- But beyond the UML:
  - Petri Nets
  - Logic
  - Relations
  - Dataflow diagrams
  - Nassi-Schneidermann diagrams
  - SDL
  - Finite automata
  - etc.

Algebraic Specifications
Entity/Relationship-Models
Jackson Structured Diagrams
Control flow diagrams

Grammars
Regular expressions

Unified Modeling Language UML

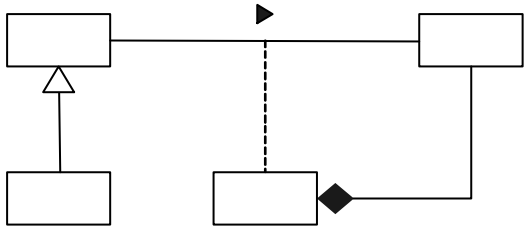


1997:	UML 1.1
1999:	UML 1.3
2001:	UML 1.4
2002:	UML 1.5
2004:	UML 2.0
2006:	UML 2.1

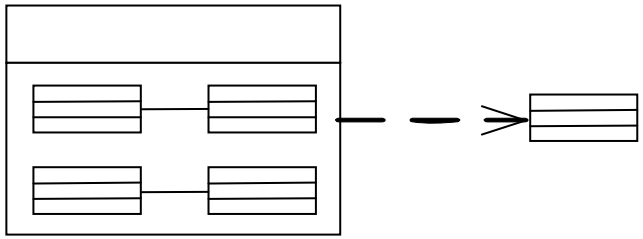
- UML is a second generation notation for object orientierted modelling

Structural diagrams of the UML

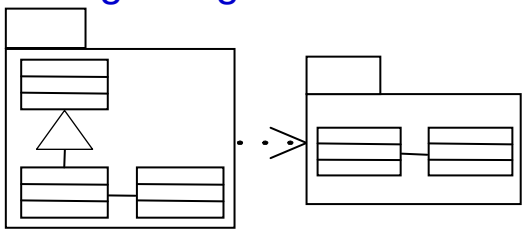
class diagram



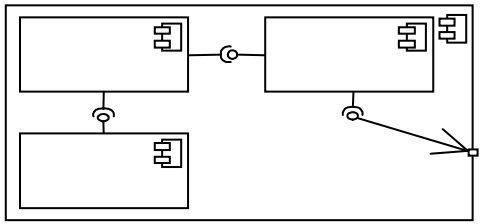
composition structure diagram



package diagram

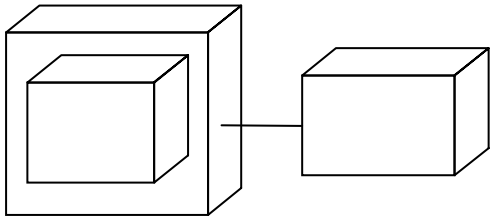


component diagram



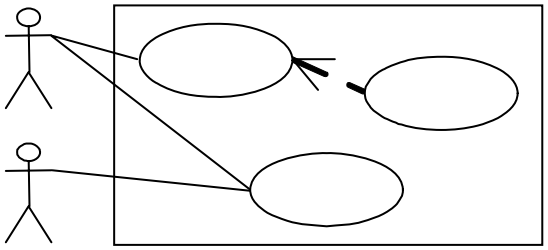
object diagram

deployment diagram

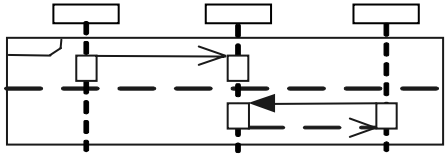


Behavioral diagrams of the UML

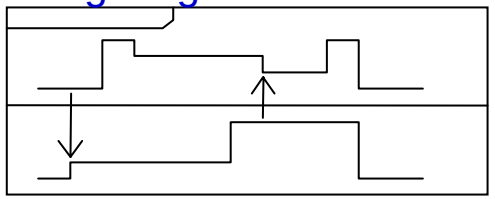
use case diagram



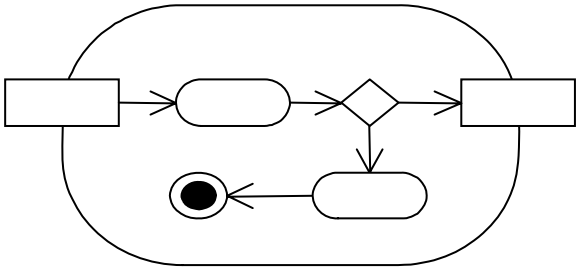
sequence diagram



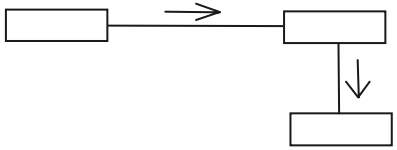
timing diagram



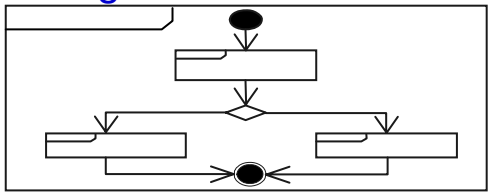
activity diagram



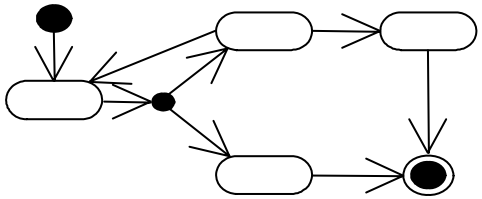
communication diagram



interaction overview diagram



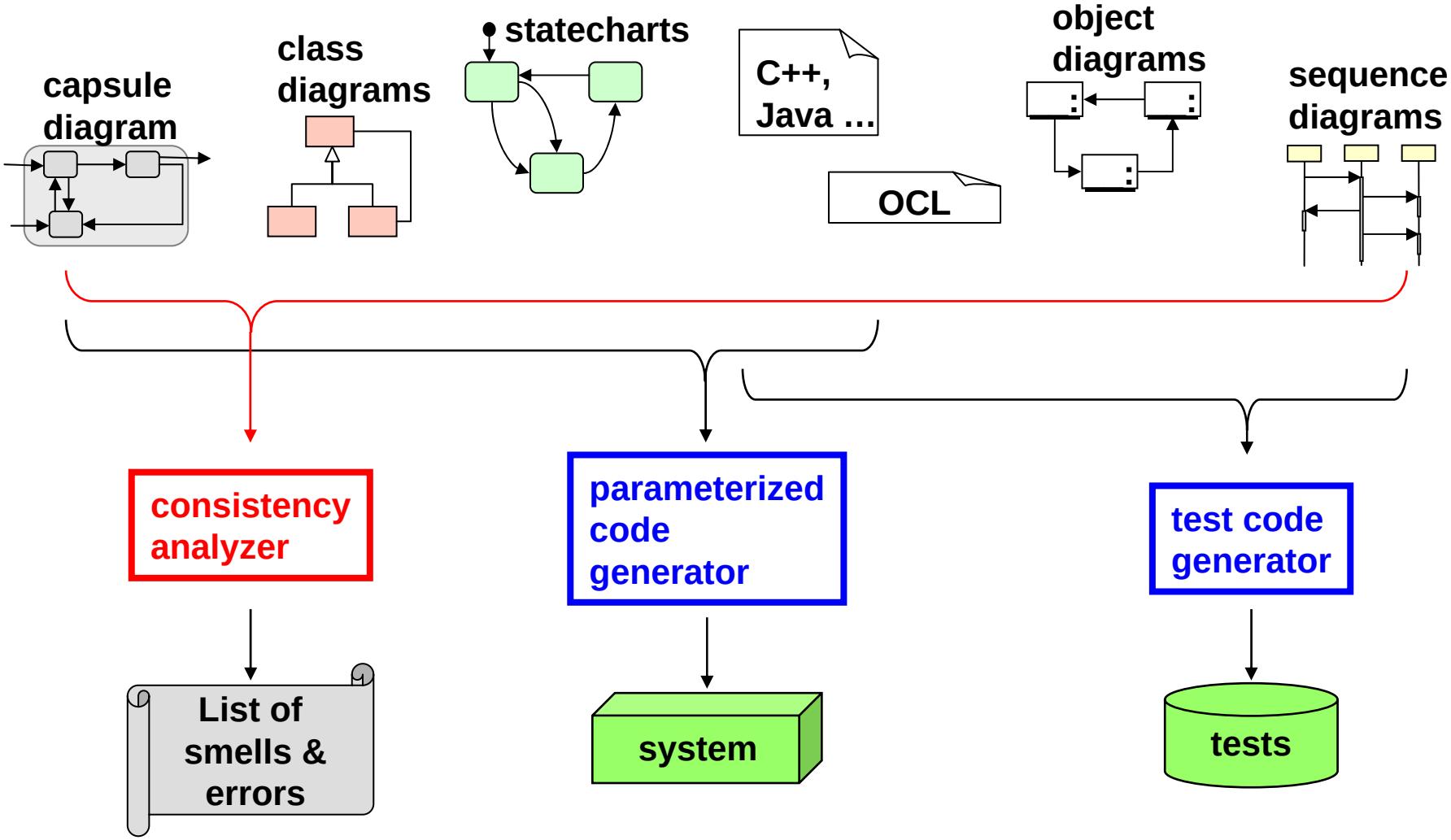
Statechart



+ textual part:
Object Constraint Language (OCL)

UML-based model engineering

- UML + code-parts enable us to model code & tests



What is semantics?

And what is it good for?

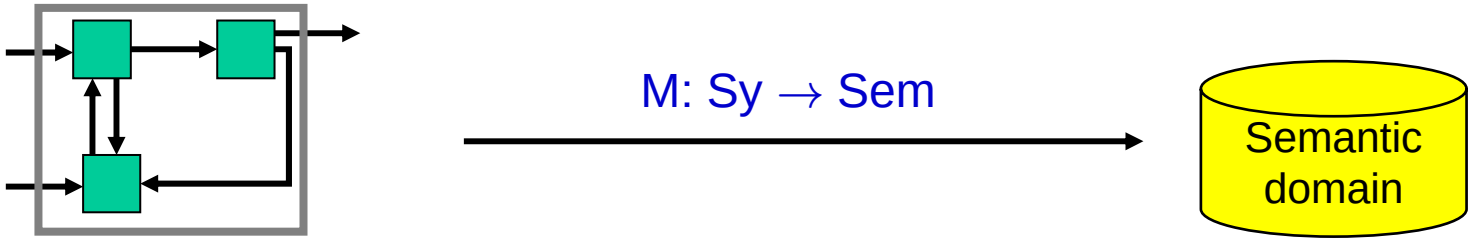
The structure of a language

- A language consists of
  - Syntax (notation)
    - Abstract syntax
    - Concrete (graphical or textual) representation
  - Semantics domain
  - Semantics mapping (explanation)
  - (and pragmatics ...)

- Semantics describes the meaning of a language.
- Computer science knows additionally “axiomatic semantics”:
  - “How to manipulate it” instead of “what it means”

Semantics

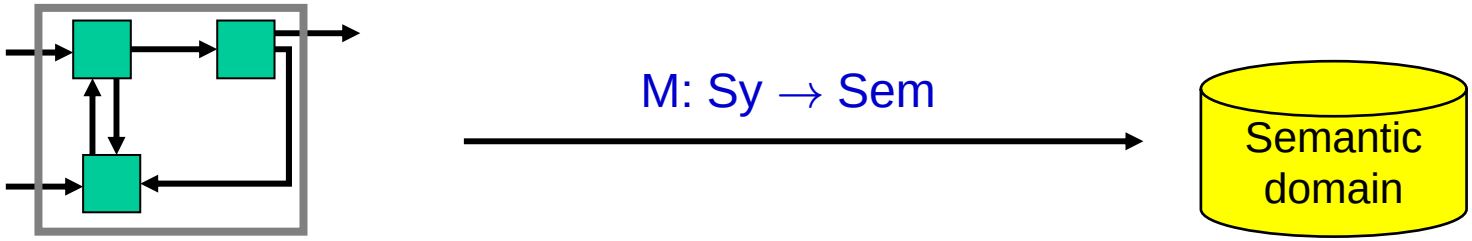
- Observation:
  semantics is a mapping from syntax to a semantic domain



- However various variations exist

Semantics: Choice of semantic domain

- Observation:
  semantics is a mapping from syntax to a semantic domain

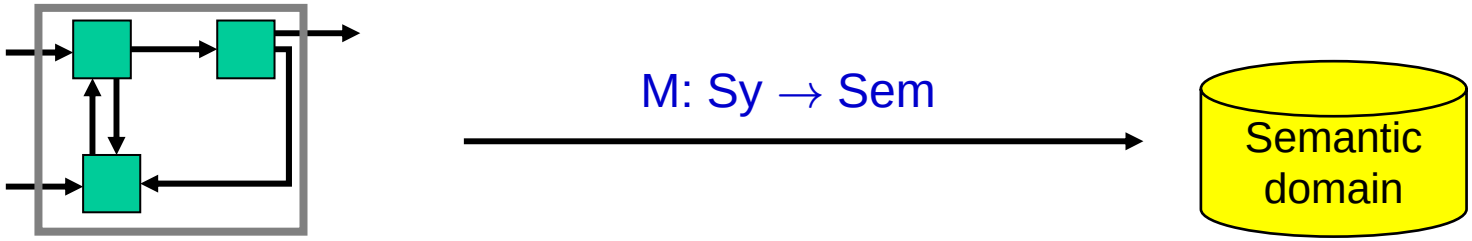


- However various variations exist e.g. for the semantic domain:
  - Explicit semantic domain: “System model” was specified
  - Implicit through use of a spec. language (CSP, Z, etc.)
  - Preciseness?
  - Detailedness?
  - Completeness to describe the language concepts?

  - Problem: Choice of semantic domain affects semantics implicitly

Semantics: Description of mapping

- Observation:
  semantics is a mapping from syntax to a semantic domain



- More variations exist e.g. for the semantic mapping:
  - Mapping given through examples (many “formalizations” do this)
  - Explicitness?
  - Preciseness?
  - Detailedness?

Preciseness of language vs. expression

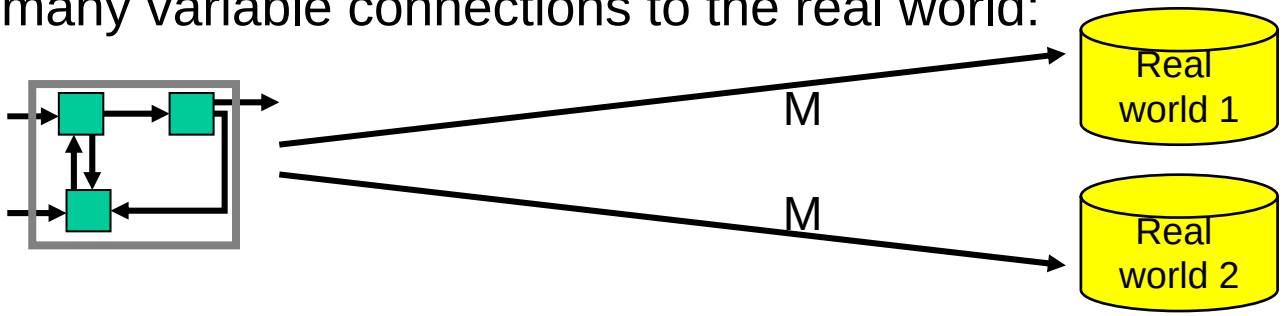
- Observation:
- Preciseness of language and detailedness of expressions are partly independent.

- Example: Mathematical expressions

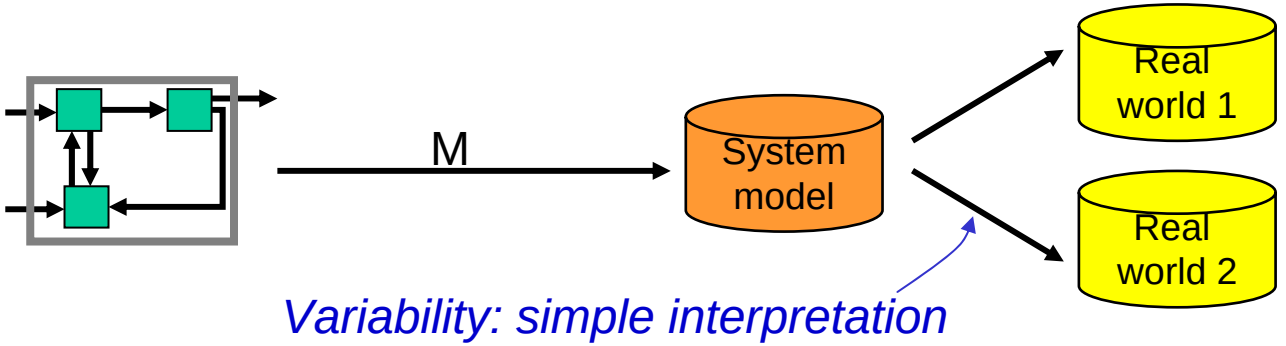
	English: imprecise language	Math: precise language
imprecise, not detailed expression	around 100	[13,2000]
precise, detailed expression	more than 98, at most 101	[99,101]

Semantics in the real world?

- Problem: UML is connected to the “real world” in various ways:
- A class can e.g. an entity of the real world or a software artifact.
- Solution:
  A precise semantics must be based on a precise abstraction of the artifacts to describe.
- Instead of many variable connections to the real world:

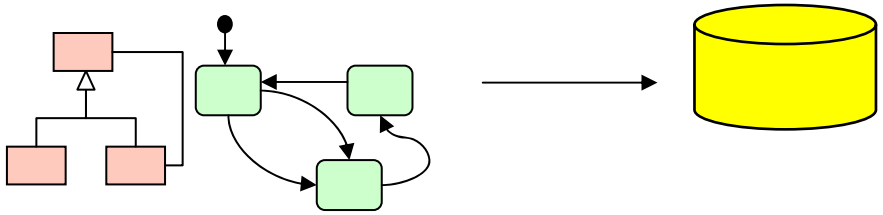


- an intermediate “system model” as abstraction of the “real world”

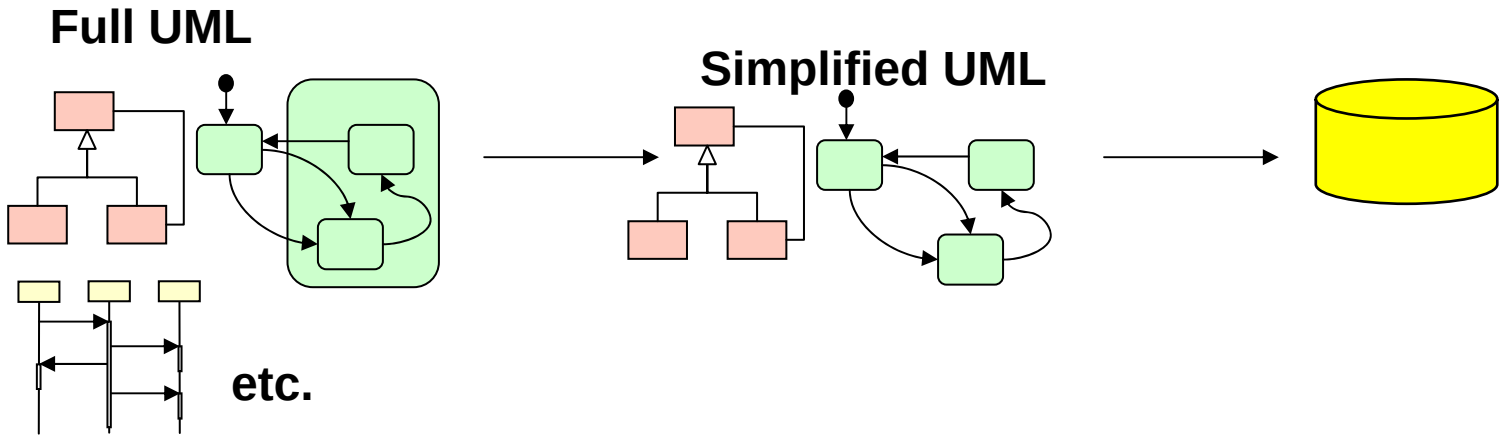


Chain of semantic mappings

- Problem:
  UML is complex: Many concepts are redundant.
- Solution:
  Reduce number of concepts via mapping into a core language
- Instead of a single mapping:

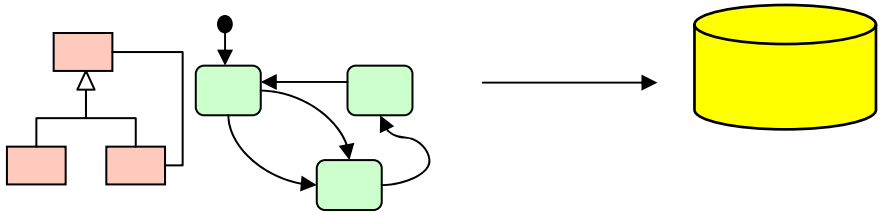


- a chain of mappings

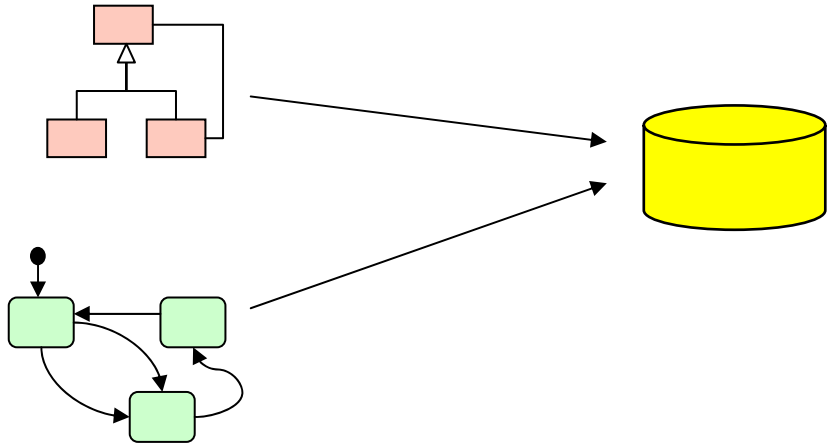


Decomposing the language

- Problem:
  UML is complex: It is combined of several languages.
- Solution:
  Define semantics for each-sublanguage individually
- Instead of a single mapping:

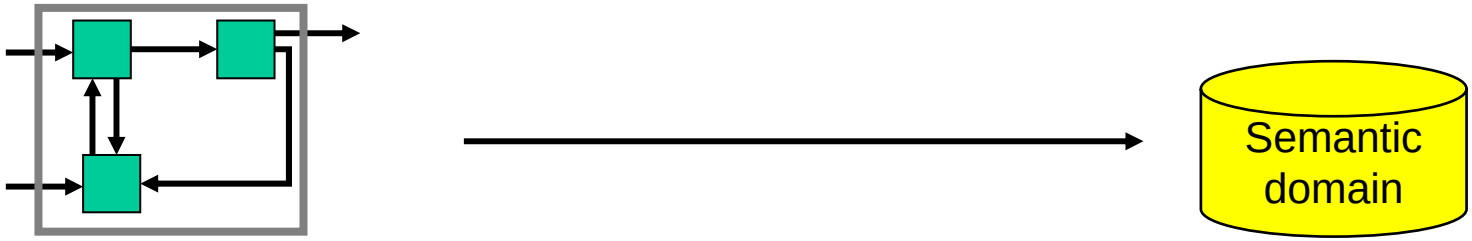


- a composed mapping



Underspecification and execution in UML

- Observation: UML is not a programming language, underspecification should be possible



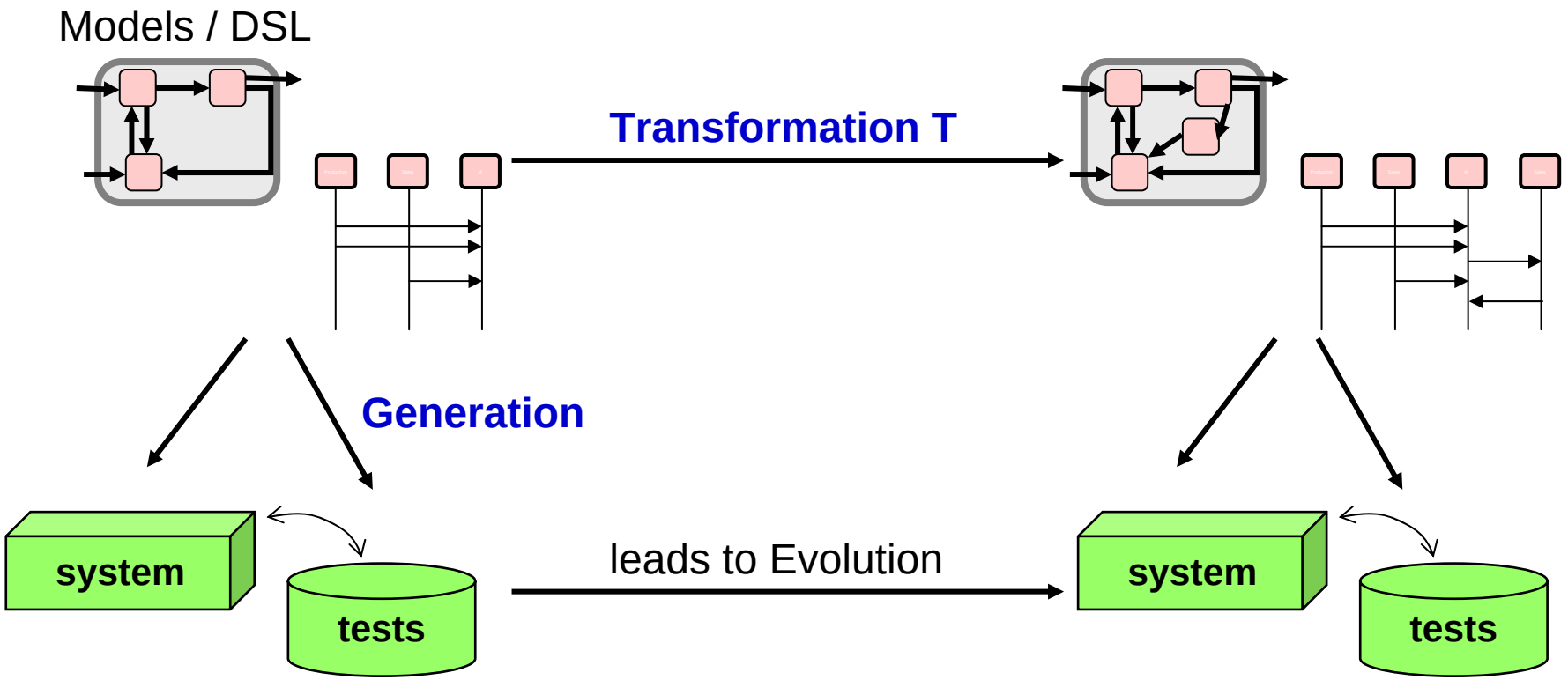
- Problem: No executable semantics for UML (in general)
- Solution: Using a "set-based" executable semantics.
  - The semantic mapping maps to a set of all possible implementations / systems
  - M: Sy $\rightarrow$ Powerset(Sem)

Underspecified: Set based semantics

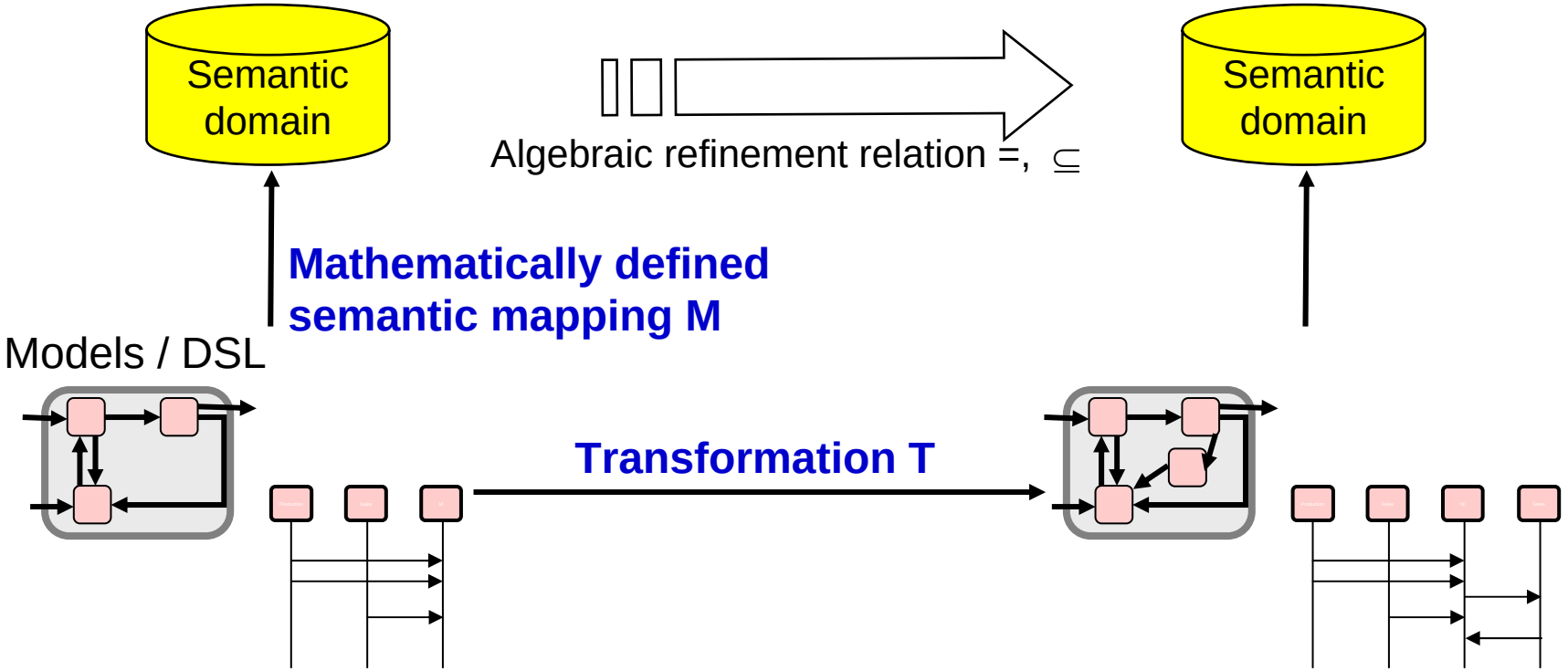
- Models are abstractions: they are underspecified:
  - We use “sets” of possible realizations as semantics
  - M: Sy $\rightarrow$ Powerset(Sem)
  - M(UML-document A) = { Set of realizations }

- Consequently:
  - A is well defined: $M(A) \neq \emptyset$
  - A is refinement of B: $M(A) \subseteq M(B)$
  - A and B are consistent: $M(A) \cap M(B) \neq \emptyset$

Transformations: Evolution of Models

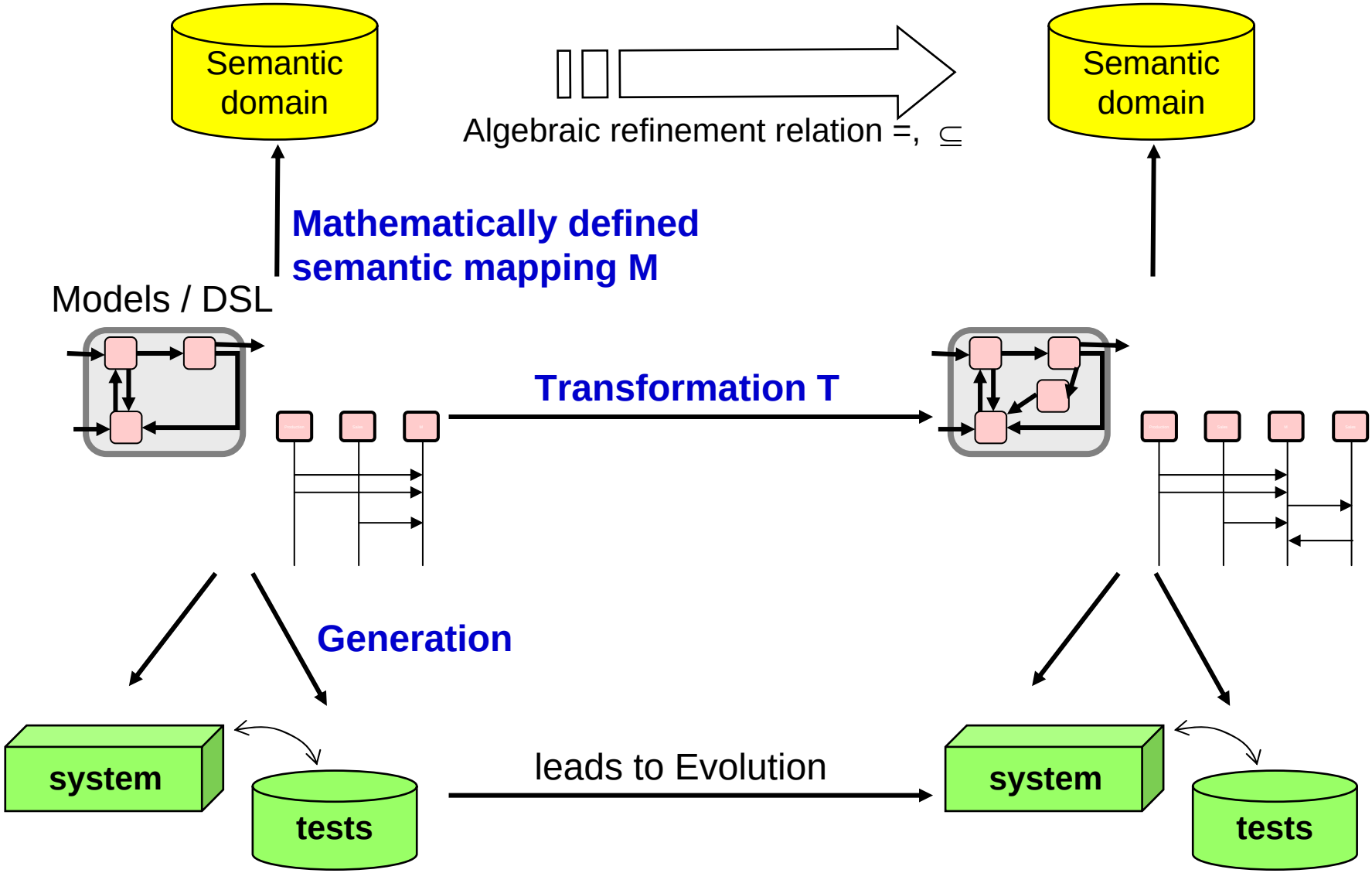


Transformations: Evolution of Models



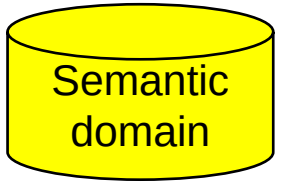
- Transformations T: Sy → Sy can be compatible to semantics:
  - T is a refinement if M(T(A)) ⊆ M(A)

Transformations: Evolution of Models



Dealing with variation points

- UML does have many variation points:
  - a) some are in the mappings
  - b) some are in the system model (for various domains)
- We use a descriptive style for property characterization: e.g.
- There are universes of
  - types UTYPE,
  - object identifiers UOID,
  - threads UTHREAD, ...



- Domain specializations may be:
  - there are real numbers: Real $\in$ UTYPE,
  - a single threaded system: |UTHREAD| = 1

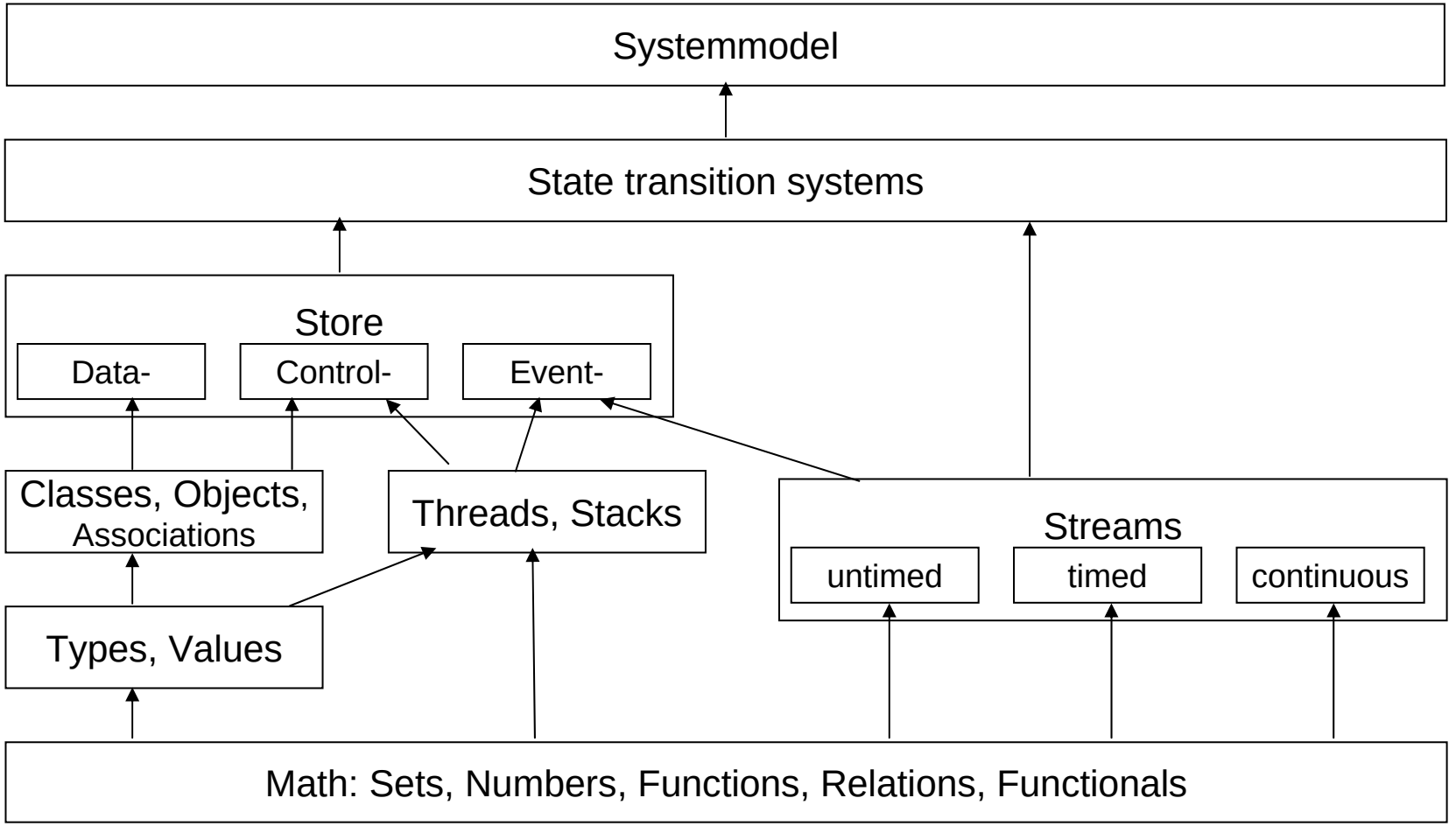
Putting it all together ...

- Our approach to semantics definition is:
  - Using pure math, because it is
  - most flexible, compact, allowing to underspecify,
  - being descriptive (denotational),
  - capture higher-order-concepts,
  - build layers of theories, ...

- We call the semantics domain a “system model”:
- It is a generic “model”, of how any OO systems is
  - structured and does its
  - behavioral interactions

Structure of the system model (Broy, Cengarle, Rumpe)

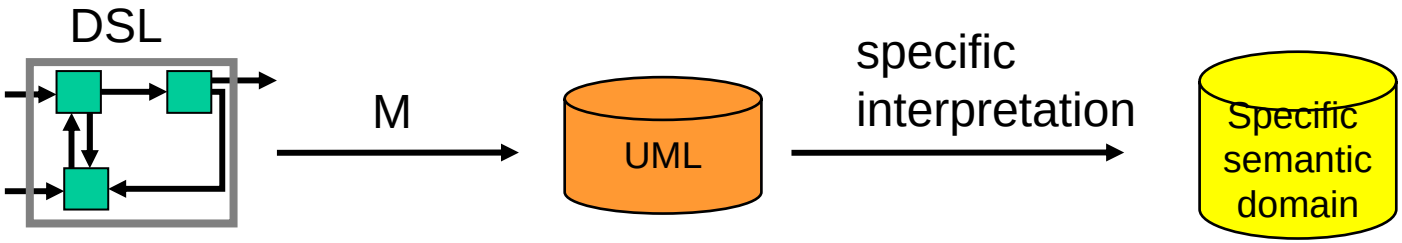
- Within math, we build layers of small theories:



Domain Specific Languages (DSL) ?

- Many DSL's need semantics too ...

- May be one possible approach is:
  - using UML to encode the "semantics domain" by
  - mapping DSL-concepts to UML-concepts
  - and specializing the interpretation of the UML-models to specific domains



Zusammenfassend ...

- Es gibt viele Wege zur Semantik einer Sprache
- UML-Semantik ist nicht fixiert
  - und es wird schwierig eine solche anerkennen zu lassen

- Für Domänenspezifische Sprachen wird es wieder richtig konfus.

- Besten Dank für Ihre Aufmerksamkeit.

Und noch etwas Werbung:

- UML-P - UML-Profil für agile Modellierung
  Tutorium:
  Code-Generierung, Testfälle, Testmuster,
  Refactoring, Evolution