Description Languages

Sandro Morasca

Università degli Studi dell'Insubria

Dipartimento di Scienze Teoriche e Applicate

Via Ottorino Rossi 9 – Padiglione Rossi

21100 Varese, Italy

sandro.morasca@uninsubria.it

Models

- Reasoning about complex systems can be quite hard
- Building complex systems can be quite hard too
- How can we evaluate important aspects of a system or the effectiveness of building solutions without having to deal with the entire complexity of a real-life system?

Models

- ➢ **Models**
- Properties
- Languages
- Styles

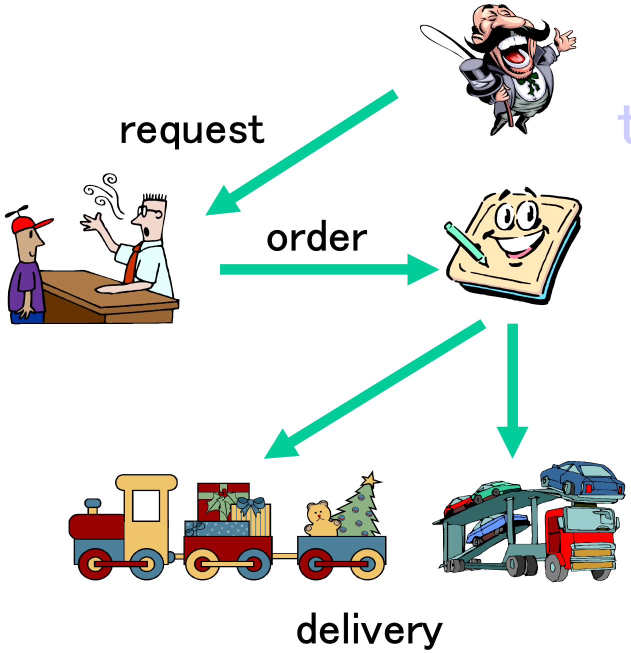
- Models are the language of the designers
  - they represent the system (to be built or already built)
  - they are a communication vehicle
  - they visually describe a system
- Models help tame complexity
- Models help analyze specific system qualities

Visual Modeling

- **Models**
- Properties
- Languages
- Styles

"A model captures the essential parts of a system"

James Rumbaugh



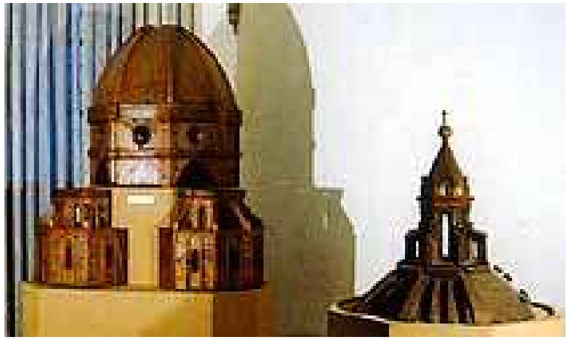
Business Process

Computer System

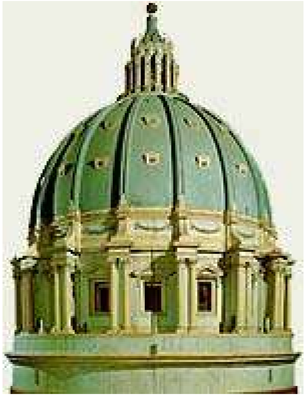
Model
An Engineering Idea

➢ **Models**
Properties
Languages
Styles

- The use of model is not a new idea
- Brunelleschi’s Dome



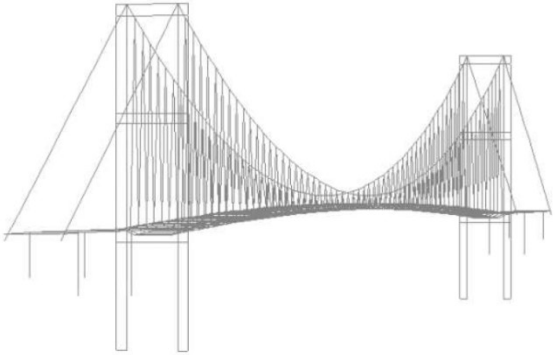
- Michelangelo’s Dome



Model
An Engineering Idea

- **Models**
- Properties
- Languages
- Styles

- Bosphorus bridge



Abstraction

- A real-life system has many elements
  - often, only few of them are "interesting"
    - what is "interesting" depends on the context
- A model can and should ignore all irrelevant details
- Abstraction = description of a system (or a part thereof) that only includes its relevant characteristics

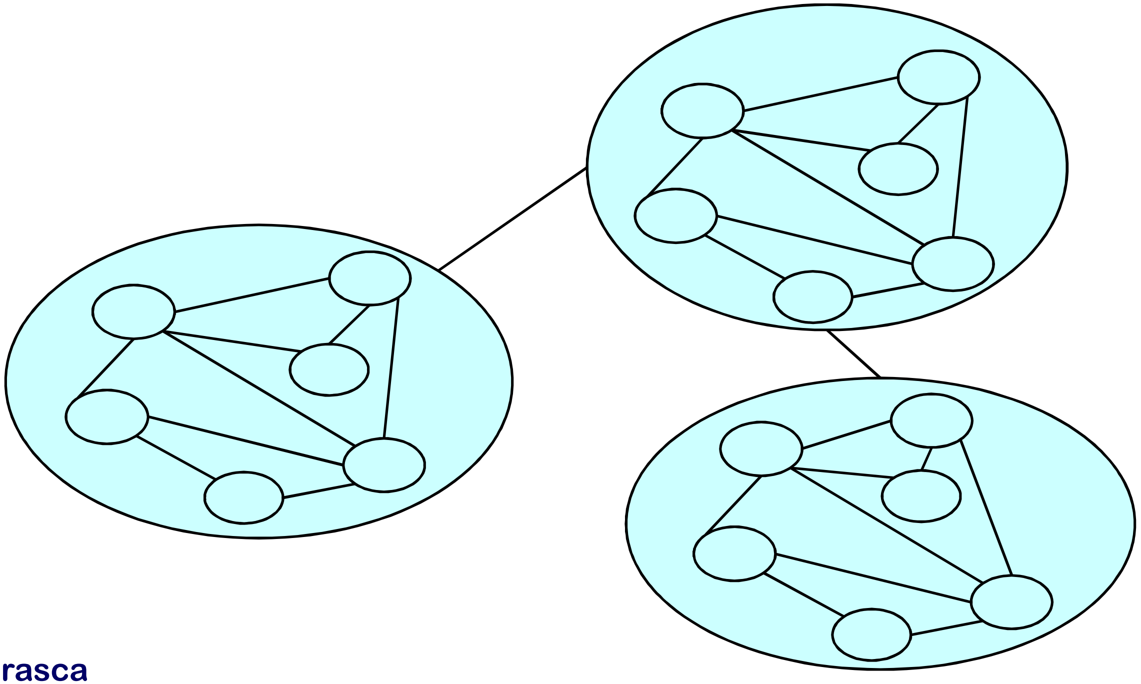
Abstraction

- Characteristics of a person
  - name
  - address
  - age
  - shoe size
  - blood group
  - social security number
  - soccer team supported
  - income
  - …
- In virtually all contexts, only a subset of these characteristics are relevant

Modularity

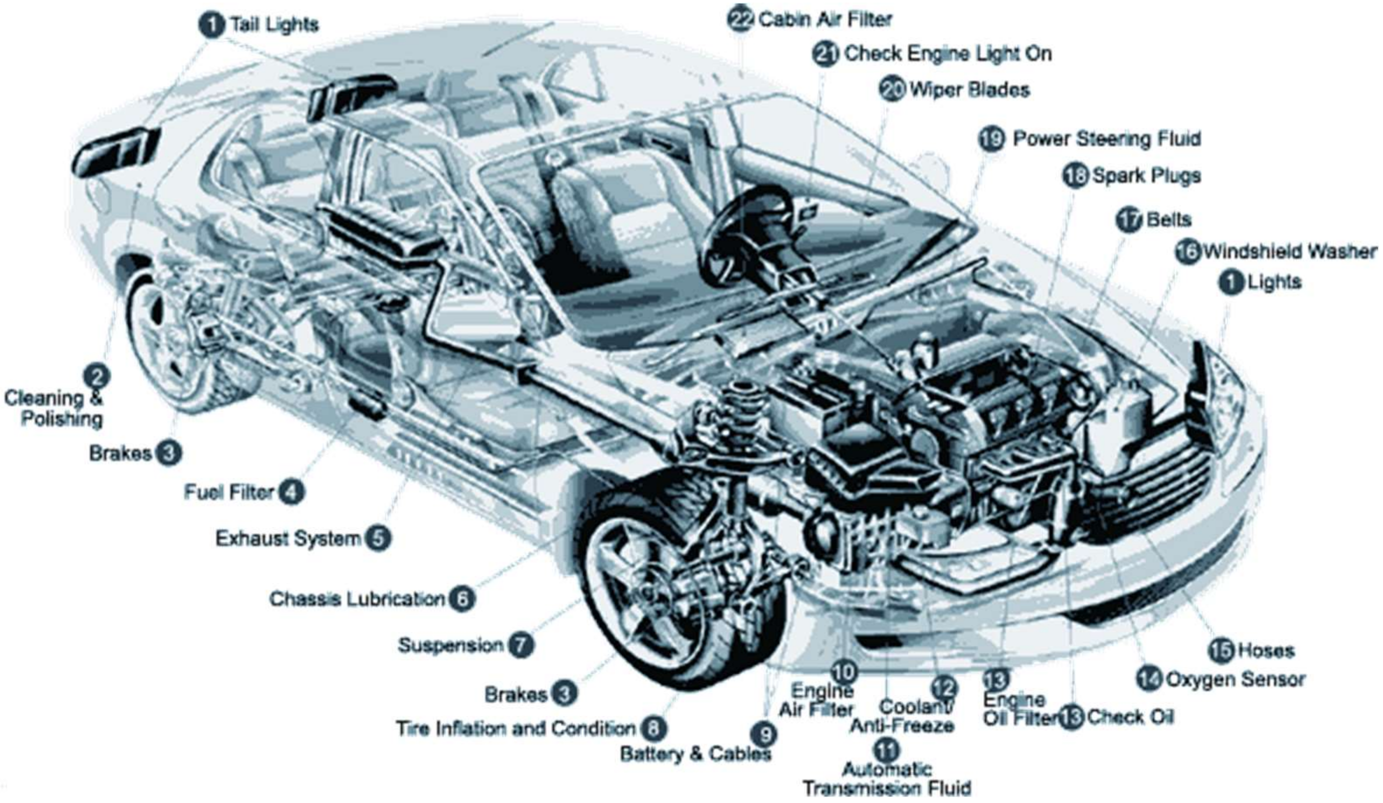
Models
➢ Properties
Languages
Styles

- Divide et impera (divide and conquer)
- A system is modular if it is decomposed into parts that are substantially autonomous and inceract as little as possible with each other
  - modules are sparsely connected and strongly cohesive



Models
➢ Properties
Languages
Styles

- Componenti di un’automobile



Modularity: Criteria

- Modular decomposability
  - decompose a large problem in smaller sub-problems that can be dealt with more easily
- Modular composability
  - aggregate existing module to solve new problems
  - reusability
  - example: Lego
- Modular comprehension
  - understand a module by observing only the module itself and the bordering ones

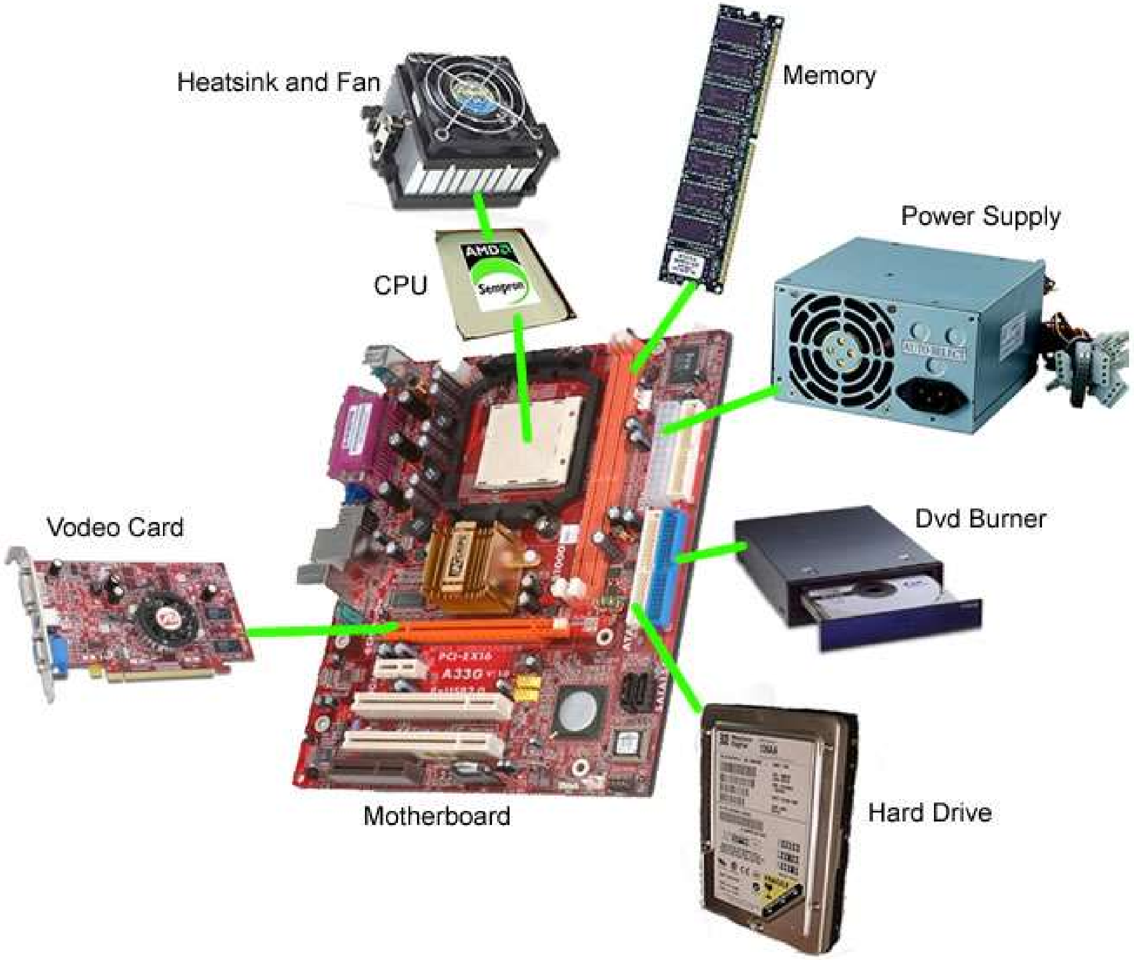
Modularity: Criteria

- Modular continuity
  - a small change in specifications implies small changes in one (or few) modules
  - extendibility
- Modular protection
  - faults only affect the module itself or at most propagate to the bordering ones

Modularity: Methods

- Modular language units
- Few communications: a module should communicate with the least possible number of modules
  - in a system with n modules, the number of communications should be close to (n-1) and far from n(n-1)/2
- Small interfaces: a module should exchange the smalles amount possible of information with the others
- Explicit interfaces: the communications between two modules A and B should be evident from the code of both A and B
- Information hiding: a module must make a clear and motivated distinction between public and private information

Modularity: Methods

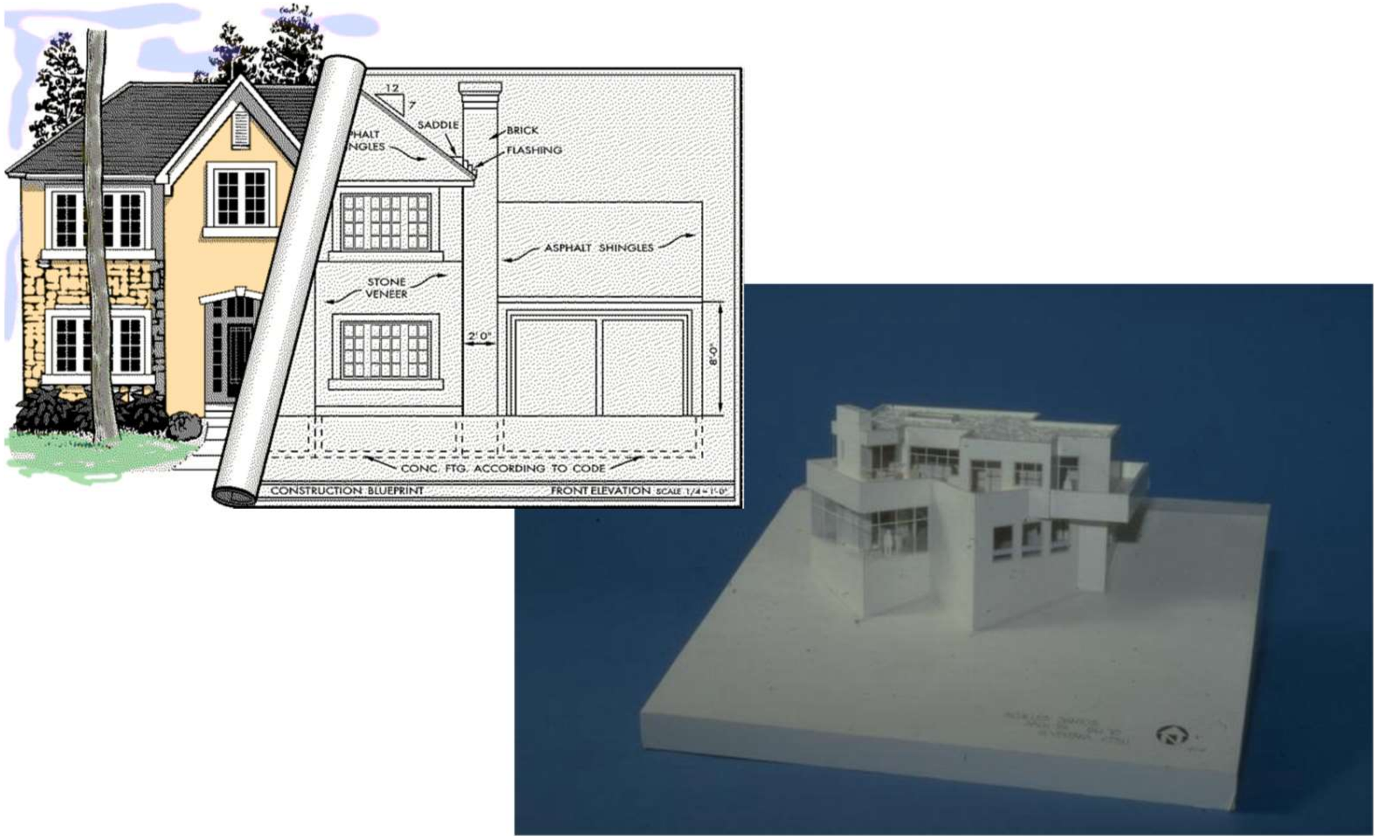


Views

- A complex system has multiple aspects that must be described to represent the system adequately
- It is not easy to have a single model that can represent diverse pieces of information like
  - static structure
  - dynamic behavior
  - logical organization
  - physical distribution
  - etc.
- Solution
  - not just one model, but several models, each specialized to provide information on a specified aspect

Views
An Engineering Idea

- Views are not a new idea



Models
Properties
➢ Languages
Styles

- Software development produces several artifacts:
  - problem definition
  - system design
  - implementation
  - documentation
  - ...
- All artifacts are written in some “language”
  - programming language (C++, Java, HTML)
  - natural language (Italian, English)
  - description languages

Description Languages

- Descriptions are produced in several process phases
  - description: set of statements about a portion of reality of interest
  - problem definition: description of the problem to solve
  - system design: among other things, description of the behavior of the program to build
- Description languages are used to write descriptions

Description Languages Characteristics

- Completeness: The language must provide all tools to describe all aspects of interest
- Accuracy: It must be possible to build a precise description
- Consistency: The language should help avoid contradictions in different representations of the same portion of reality
- Understandability: The description must be easily understandable for all those who must interpret it (to modify it, to use it as reference, etc.)
- Formality: It is the level of rigor with which the language’s syntax and semantics are defined
- Style: The system aspect (behaviors or properties) that is easier to describe with the language

Degree of Formality

- Informal: typically, natural language (Italian, English)
  - often used because the custemer can understand it
- Semi-formal: its syntax is perfectly defined, but its semantics is not
  - easy to understand, but it can be ambiguous
- Formal: both syntax and semantics are rigorously defined
  - it has logical-mathematical foundations
    - eliminates ambiguities
  - it makes it possible to reason on and check properties, possibly with (semi)automatated tools
    - N.B. Not all properties are formally verifiable

Styles

- Descriptive style: it defines desired properties, e.g.,
  - the curve satisfies equation $x^2 + y^2 - c = 0$.
- Operational style: it defines the desired behavior (a “machine”), e.g.,
  - take a compass
  - put its pin in the origin
  - open the compass to width $c^{1/2}$
  - lower the lead down to the sheet in point $(0, c^{1/2})$
  - draw in a continuous fashion clockwise
  - stop when the lead is back in $(0, c^{1/2})$

There are many possible variations in the description

Descriptive Style

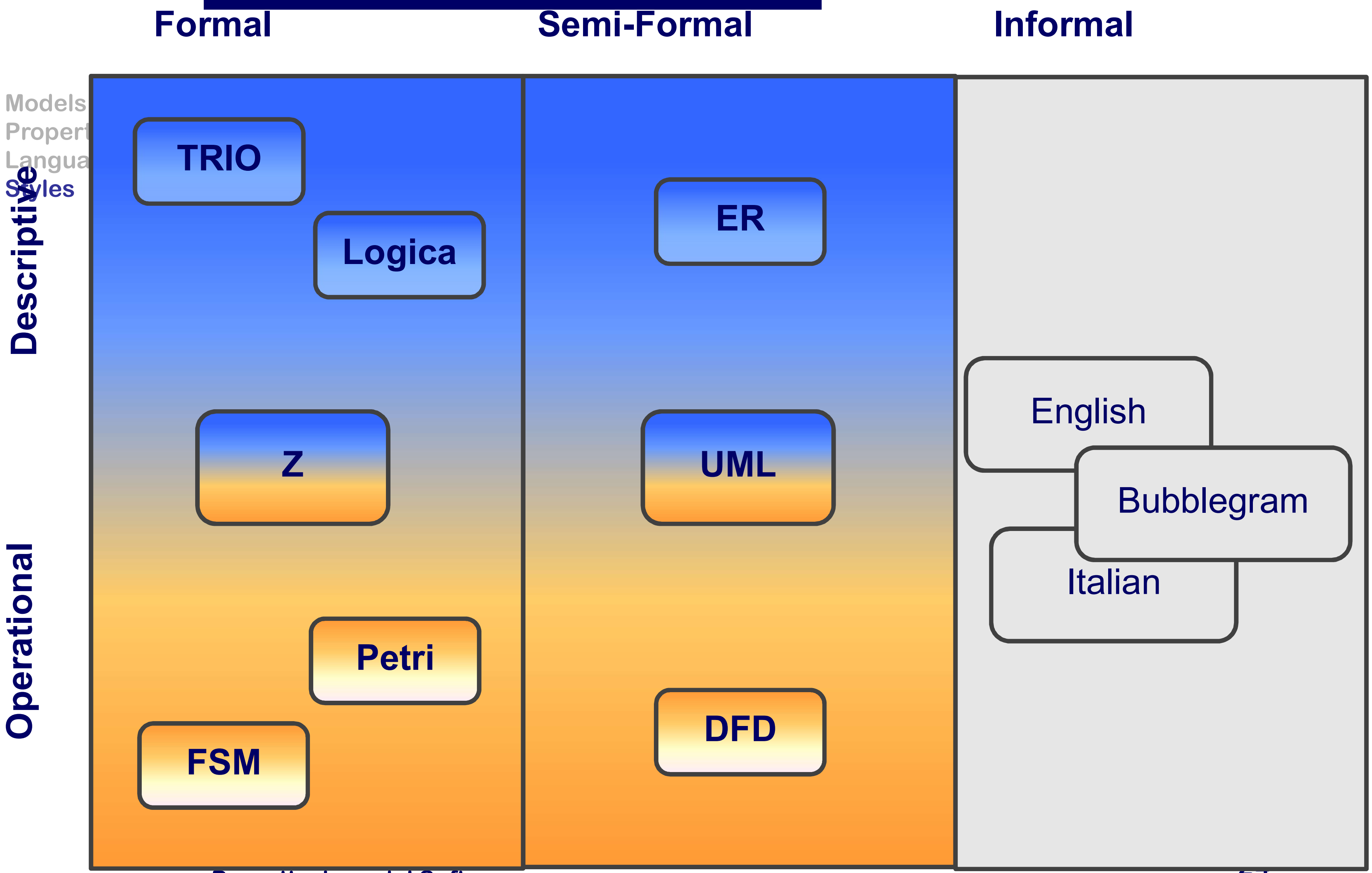
- The descriptive style does not focus on behaviors, but on system properties
- More compact description
- More abstract description: it does not describe a possible implementation
- It makes it more difficult to understand system behaviors
- It makes it easier to reason about the system
  - formally proving some property
  - modifying the description so that a property holds

Operational Style

- It describes behaviors via an abstract machine
- It suggests a possible implementation
  - but it may not be the best one (e.g., the most efficient one)
- It can be easily executed
- It is possible to build a prototype and check how close it is to the description
- It leads towards some specific implementation
- It makes it more difficult to state and prove system properties

Description Languages

Models
Propert
Langua
➢ Styles



Formal

Semi-Formal

Informal

Descriptive

Operational

TRIO

Logica

Z

Petri

FSM

ER

UML

DFD

English

Bubblegram

Italian