What is Modeling?
Model Consistency

What's a Model?

Mathematics and Physics

Concept

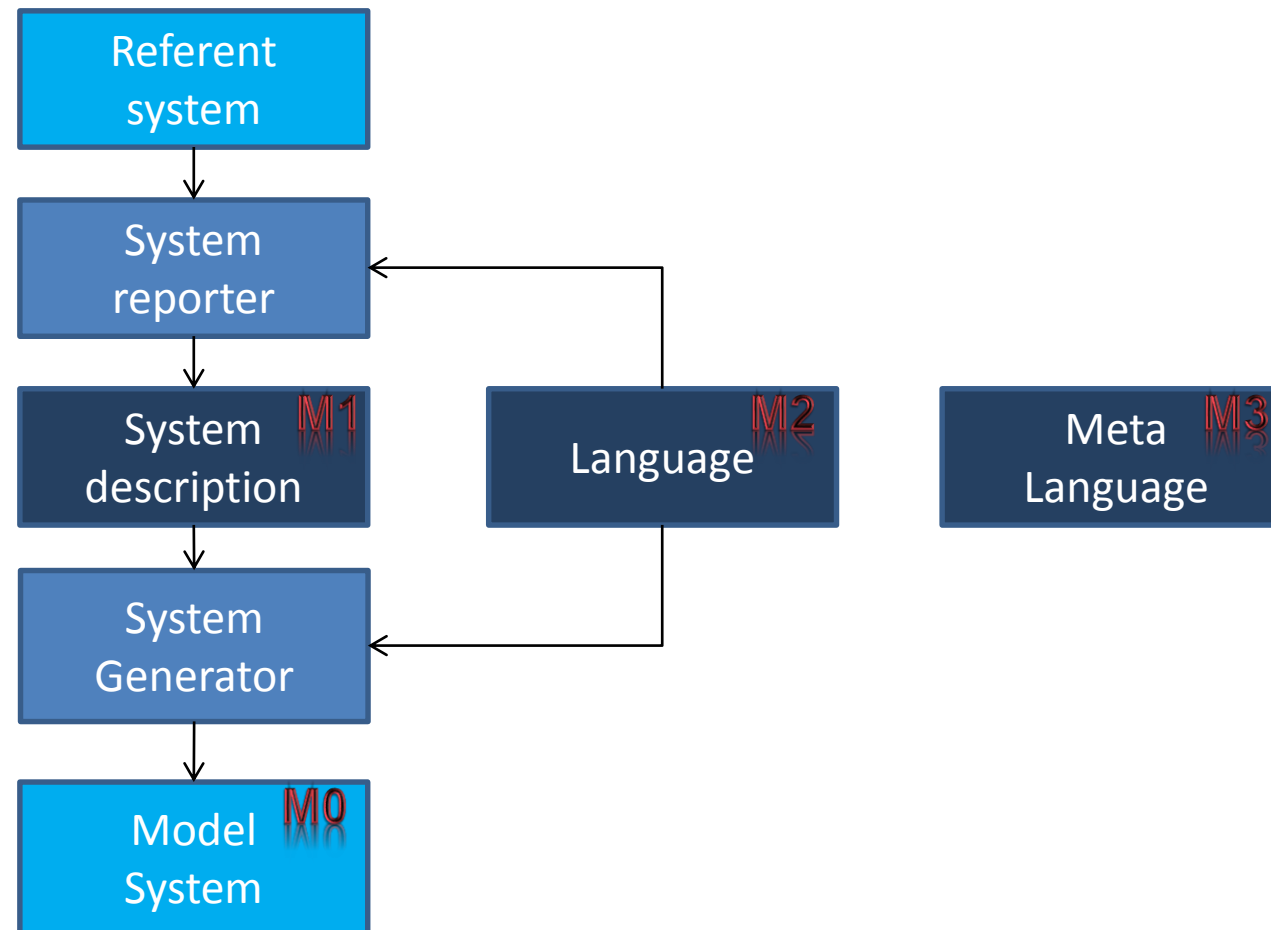
THOUGHT OR REFERENCE

CORRECT • Symbolises (a causal relation)

ADEQUATE • Refers to (other causal relations)

Model

?

SYMBOL

Stands for (an imputed relation)

• TRUE

REFERENT

Term

Phenomenon

UML1

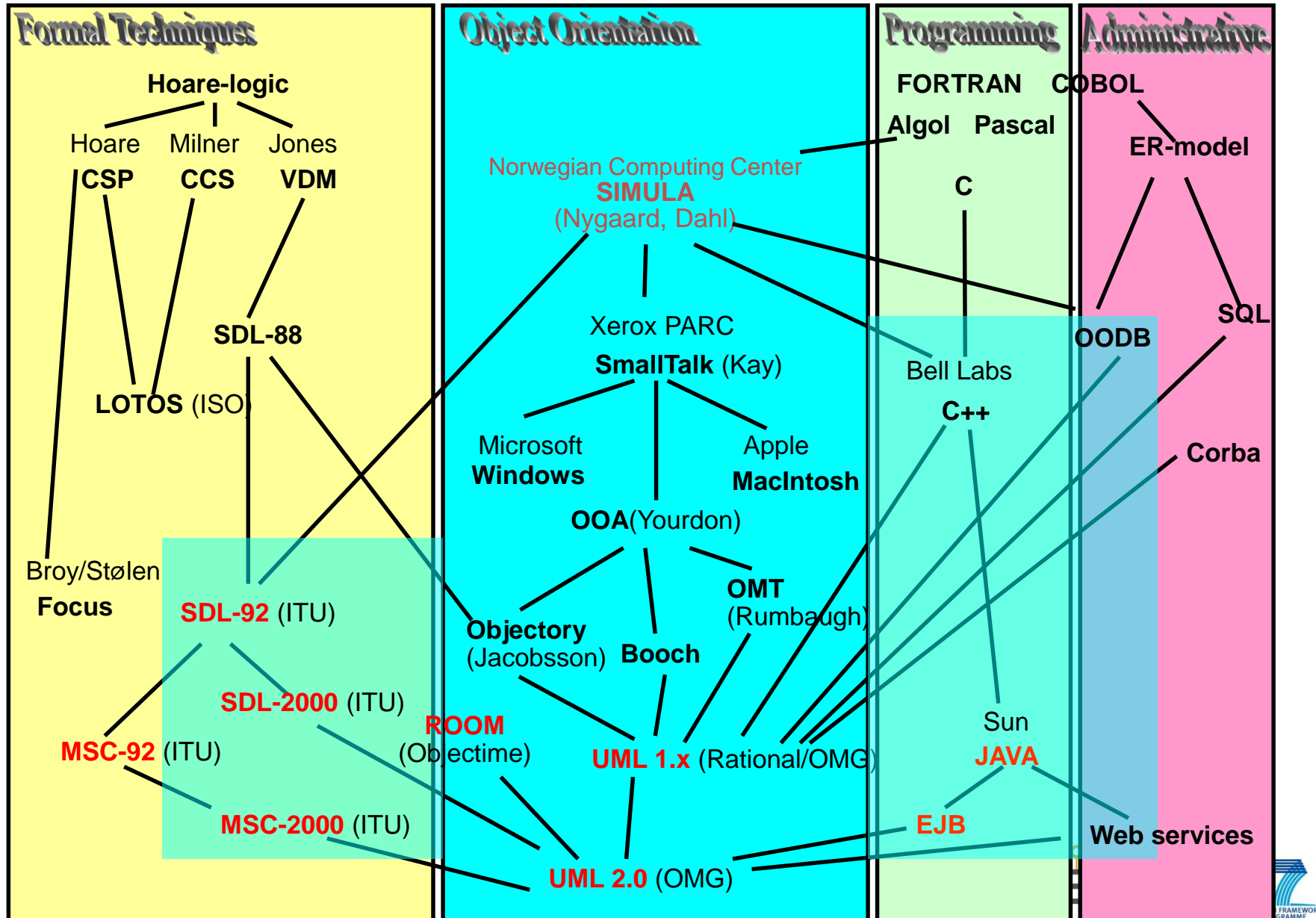Models@runtime

HEADS

http://heads-project.eu

2
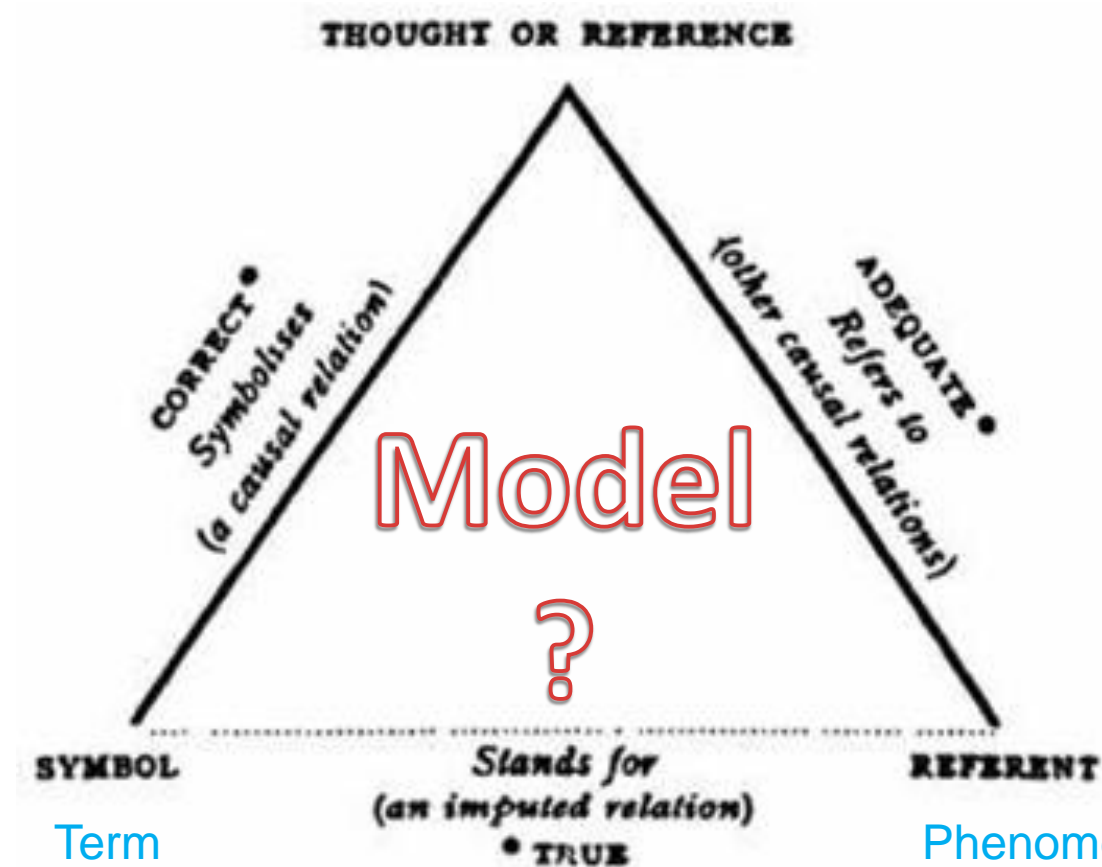
Modeling a system

- A system is a part of the world
  - which we choose to regard as a whole, separated from the rest of the world during some period of consideration, a whole which we choose to consider as containing a collection of components, each characterized by a selected set of associated data items and patterns, and by actions which may involve itself and other components
- Mental systems
  - Systems existing in the human mind, physically materialized as states of the cells of our brains
- Mental and manifest models
  - when a limited set of properties is selected from a system
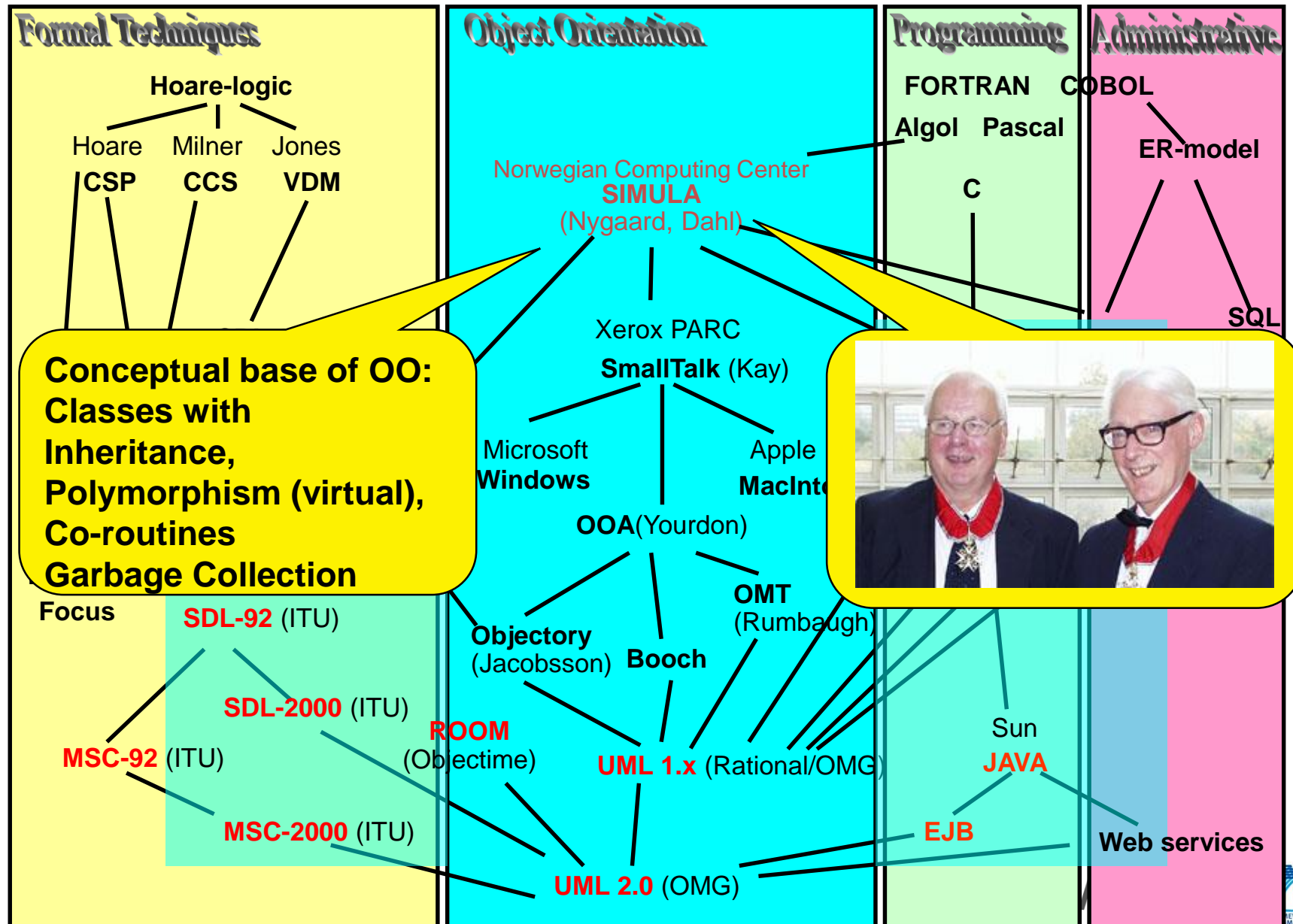- These definitions are from K. Nygaard and his DELTA team (in 1977)
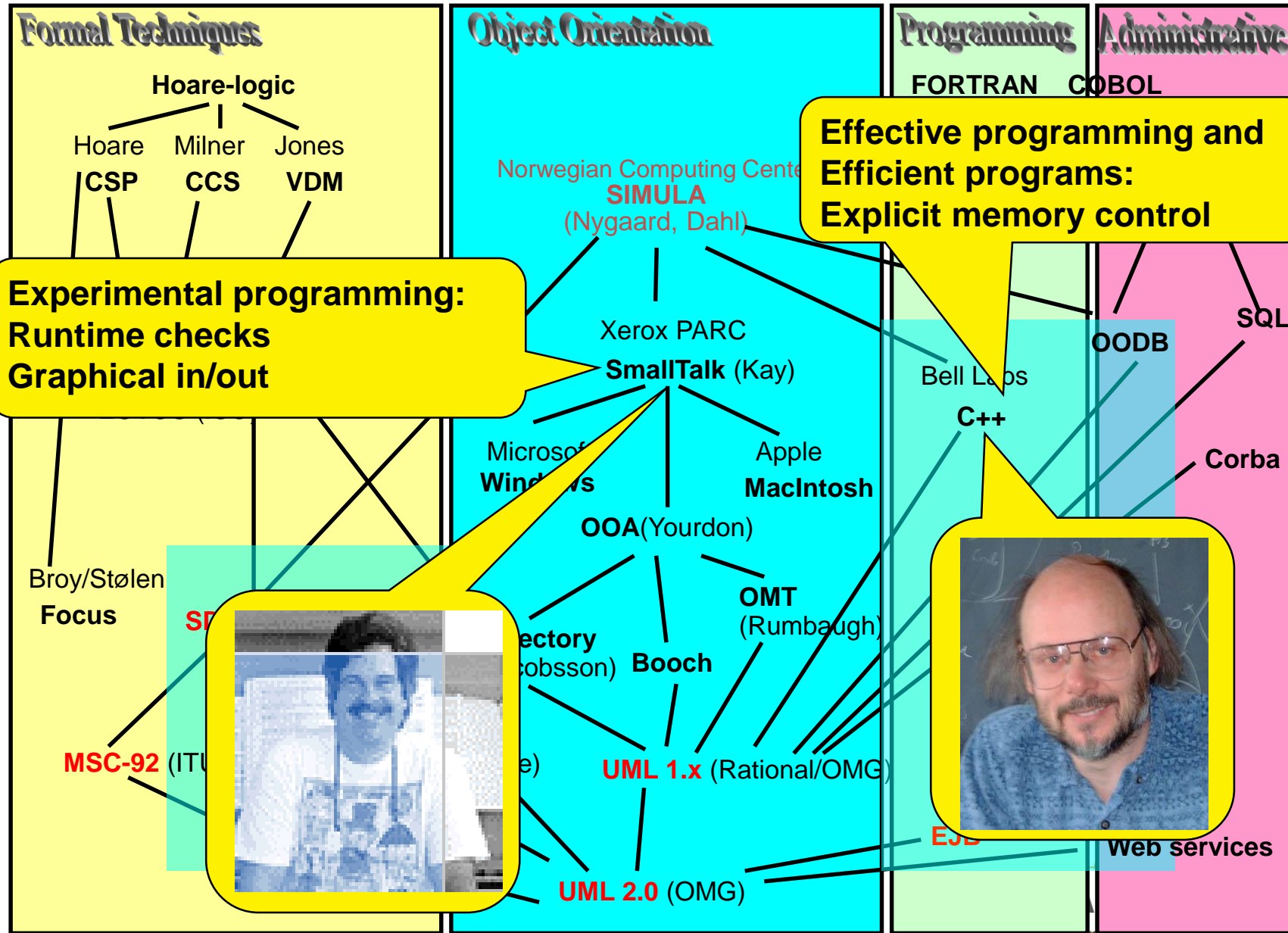
Modeling levels revisited
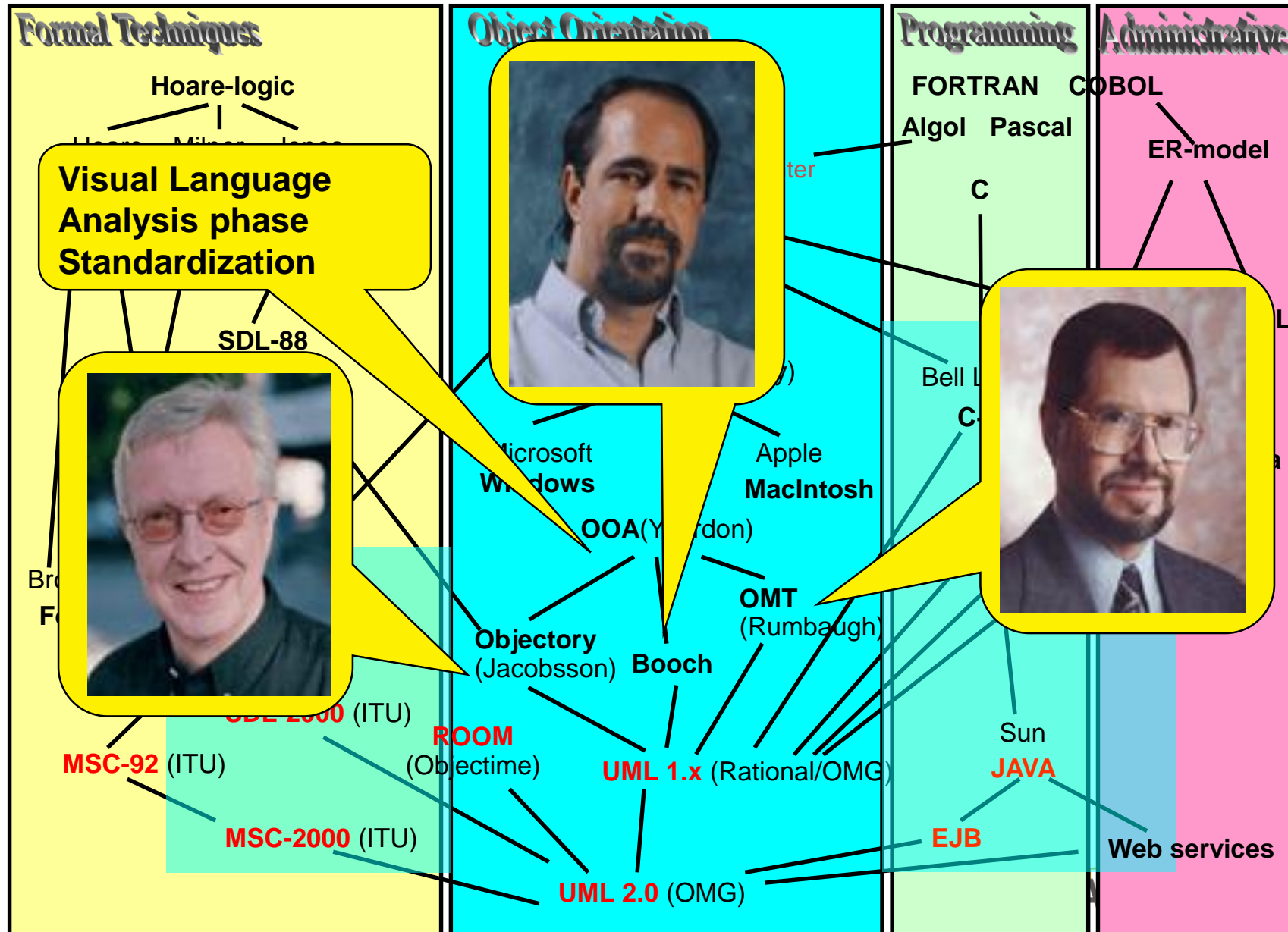
A history of modeling languages

Formal Techniques

Hoare-logic

Hoare   Milner   Jones
CSP   **CCS**   **VDM**

SDL-88

LOTOS (ISO)

Broy/Stølen
Focus

SDL-92 (ITU)

SDL-2000 (ITU)

MSC-92 (ITU)

MSC-2000 (ITU)

ROOM
(Objectime)

UML 2.0 (OMG)

Object Orientation

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Microsoft   Apple
Windows   **MacIntosh**

OOA(Yourdon)

OMT
(Rumbaugh)

Objectory
(Jacobsson)   **Booch**

UML 1.x (Rational/OMG)

Programming

FORTRAN   COBOL

Algol   **Pascal**

C

Bell Labs
C++

Sun
JAVA

EJB

Administrative

ER-model

SQL

OODB

Corba

Web services

The founding fathers

Formal Techniques

Hoare-logic

Hoare | Milner | Jones
CSP | **CCS** | **VDM**

Object Orientation

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Microsoft
Windows

Apple
MacInt...

OOA (Yourdon)

OMT
(Rumbaugh)

Objectory
(Jacobsson) | **Booch**

> **Conceptual base of OO:**
> **Classes with**
> **Inheritance,**
> **Polymorphism (virtual),**
> **Co-routines**
> **Garbage Collection**

Focus

SDL-92 (ITU)

SDL-2000 (ITU)

MSC-92 (ITU)

ROOM
(Objectime)

MSC-2000 (ITU)

UML 1.x (Rational/OMG)

UML 2.0 (OMG)

Programming

FORTRAN   **COBOL**

Algol   **Pascal**

C

Sun
JAVA

EJB

Web services

Administrative

ER-model

SQL

Making OO Popular and Commercial

Formal Techniques

Hoare-logic

Hoare — **CSP**
Milner — **CCS**
Jones — **VDM**

Broy/Stølen **Focus**

SDL

MSC-92 (ITU)

Object Orientation

Norwegian Computing Center
SIMULA
(Nygaard, Dahl)

Xerox PARC
SmallTalk (Kay)

Microsoft **Windows**   Apple **MacIntosh**

OOA (Yourdon)

OMT (Rumbaugh)

ectory (cobsson)   **Booch**

UML 1.x (Rational/OMG)

UML 2.0 (OMG)

Programming

FORTRAN   COBOL

Bell Labs
C++

Administrative

OODB   **SQL**

Corba

EJB   **Web services**

Experimental programming:
Runtime checks
Graphical in/out

Effective programming and
Efficient programs:
Explicit memory control

7

The Three Amigos

Influences on UML 2.0

Why make a language?

http://heads-project.eu

Domain Specific Language characteristics

- A language is a precise and well-defined way to describe an area of concern

  - We have a long tradition for making languages and for making supporting tools and frameworks for that

- There are domain specific languages wherever you turn

  - like the London metro map pioneered by Beck in 1931

  - like architectural drawings of buildings and train stations

- "Make everything as simple as possible, but not simpler"

 - General languages are just too much of everything

The business case for DSLs

- "Small is beautiful"
  - You have full control and rely on nobody
- Much easier to make code generation that can produce 100% of the code
  - because there are few elements and they are all well known to you
- Well documented company-wide understanding
  - good for bringing new employees into the company

# Why use a general standardized language?

- Common terms and interpretation
  - across persons, teams, companies and cultures
    - Experience SISU project
      - Very large SDL specification ported from Alcatel to Kongsberg
    - Experience MSC
      - We have a Korean translation of MSC 2000
  - across computers! portability
    - Experience Simula
      - We ported the exact same code on at least 5 machines without changing a single line of code around 1980
- Common teaching material
- Common libraries
- Common and open reviewing process

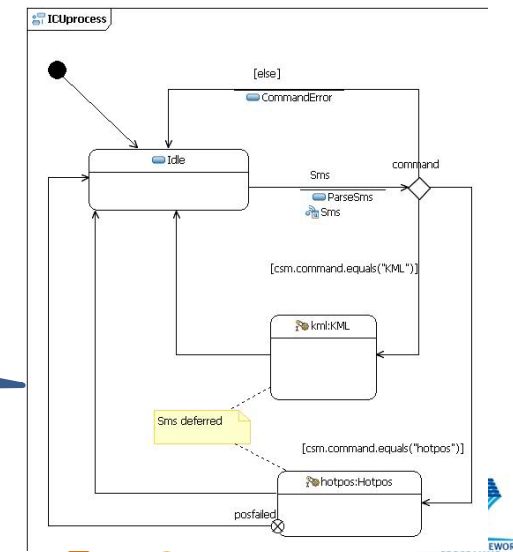# We will apply modeling for several purposes
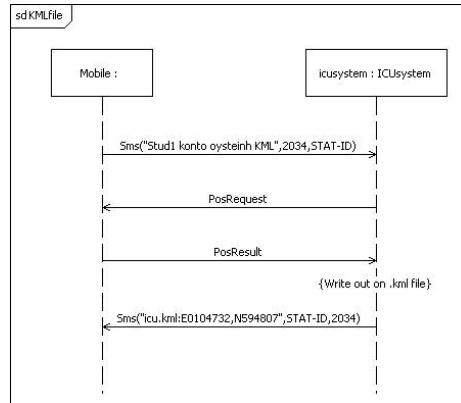
Behavioral Requirements of Interaction

Product Lines (Variability)

Executable models

http://heads-project.eu
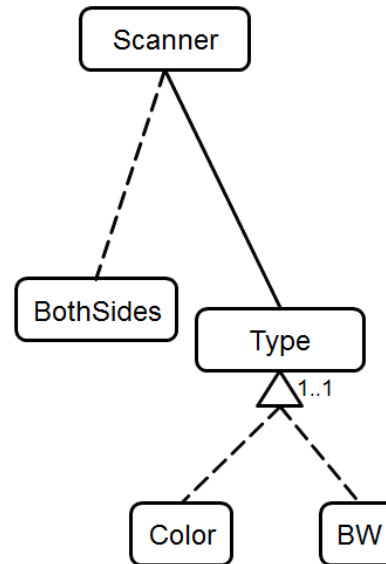
The CPS Modeling Methodolody

Agile modeling

- "agile"
  - = having a quick resourceful and adaptable character
- executable models!
- very stepwise approach
  - each step will have its specification and executable model
  - each step should be tested
- We shall use one example throughout the course
  - with many steps
  - intended to be mirrored by the project exercise model
- Every week a working program!

Manifesto for Agile Software Development

- We are uncovering better ways of developing software by doing it and helping others do it.

- Through this work we have come to value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- That is, while there is value in the items on the right, we value the items on the left more.

Dialectic Software Development

- Software Development is a process of learning
  - once you have totally understood the system you are building, it is done
- Learning is best achieved through conflict, not harmony
  - discussions reveal problematic points
  - silence hides critical errors
- By applying different perspectives to the system to be designed
  - inconsistencies may appear
  - and they must be harmonized
- Inconsistencies are not always errors!
  - difference of opinion
  - difference of understanding
  - misunderstanding each other
  - a result of partial knowledge
- Reliable systems are those that have already met challenges

Consistency

Consistency 1: Ports
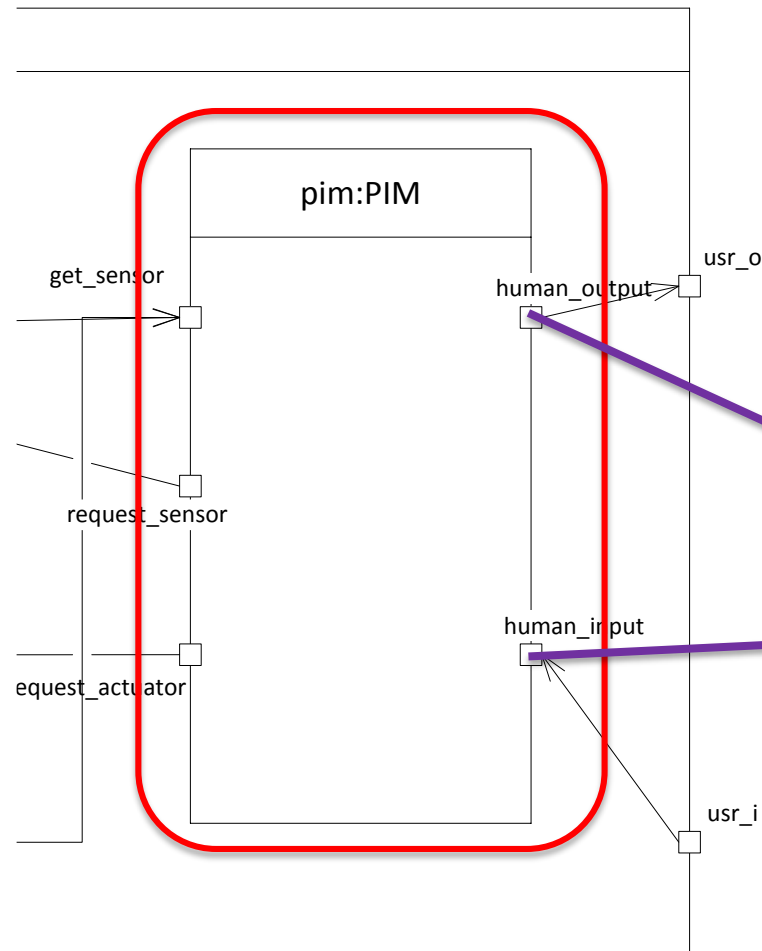

```
thing PIM includes GeneralMsg, TemperatureMsg, OnOffMsg {
    provided port get_sensor {
        receives temperature, sensorinfo, deviceinfo
    }
    required port request_sensor {
        sends add_thermometer
    }
    required port request_actuator{
        sends add_device, SwitchOn, SwitchOff
    }
    provided port human_input {
        receives add_thermometer, add_device, fetch_temp, fetch_
    }
    required port human_output {
        sends temperature, sensorinfo, deviceinfo
    }
}
```
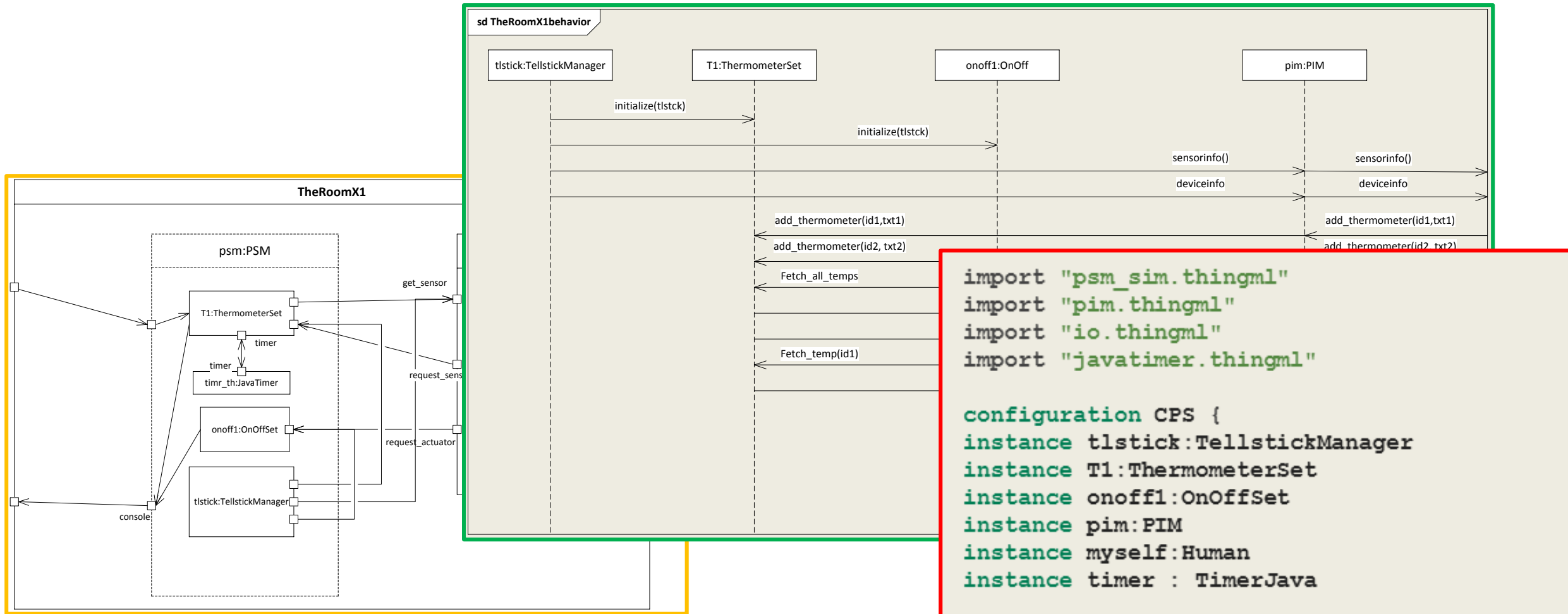
Consistency 2: Messages

```
thing PIM includes GeneralMsg, TemperatureMsg, OnOffMsg {
    provided port get_sensor {
        receives temperature, sensorinfo, deviceinfo
    }
    required port request_sensor {
        sends add_thermometer
    }
    required port request_actuator{
        sends add_device, SwitchOn, SwitchOff
    }
    provided port human_input {
        receives add_thermometer, add_device, fetch_temp, fetch_all_temps, SwitchOn, SwitchOff
    }
    required port human_output {
        sends temperature,  sensorinfo, deviceinfo
    }
}
```
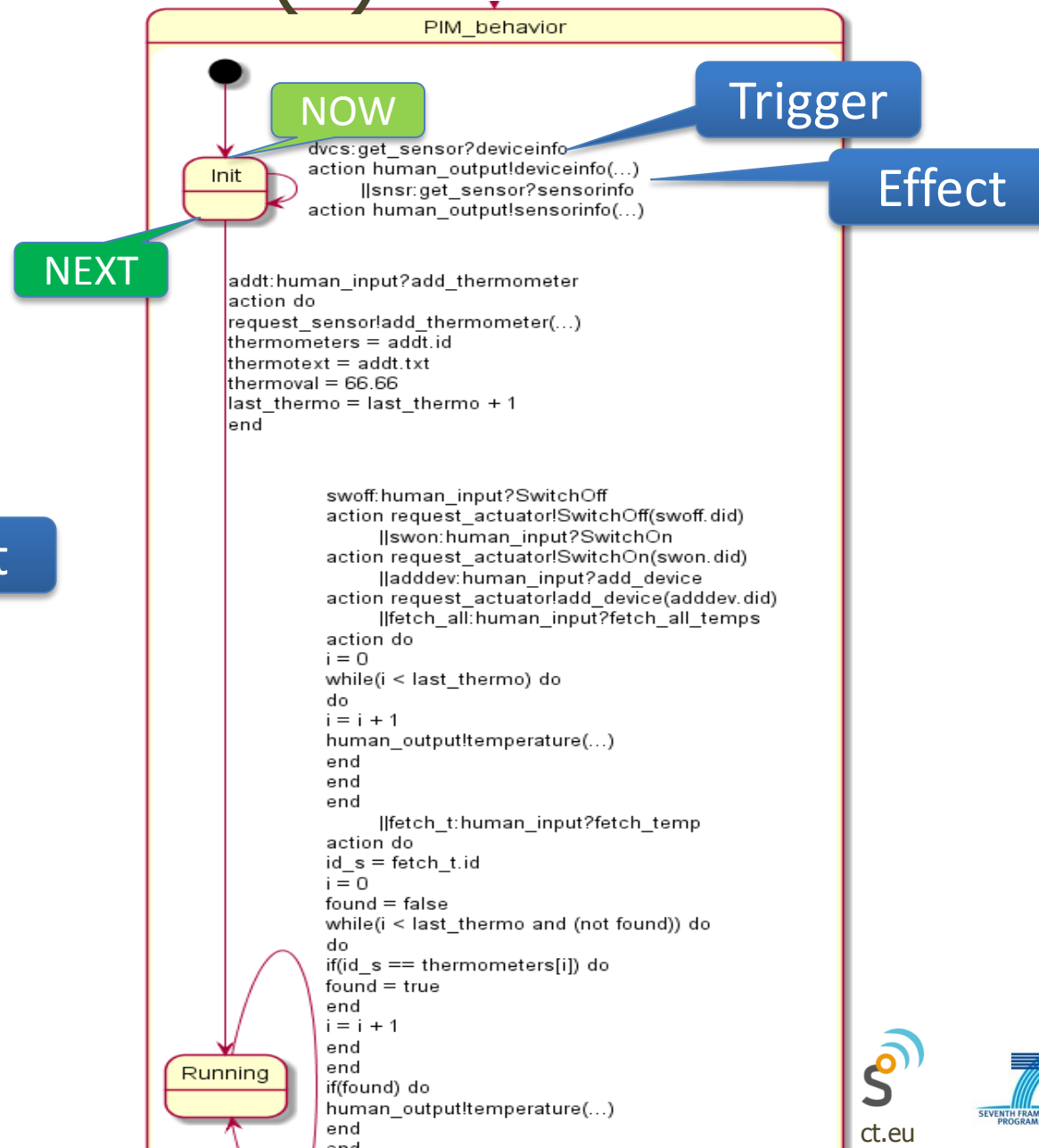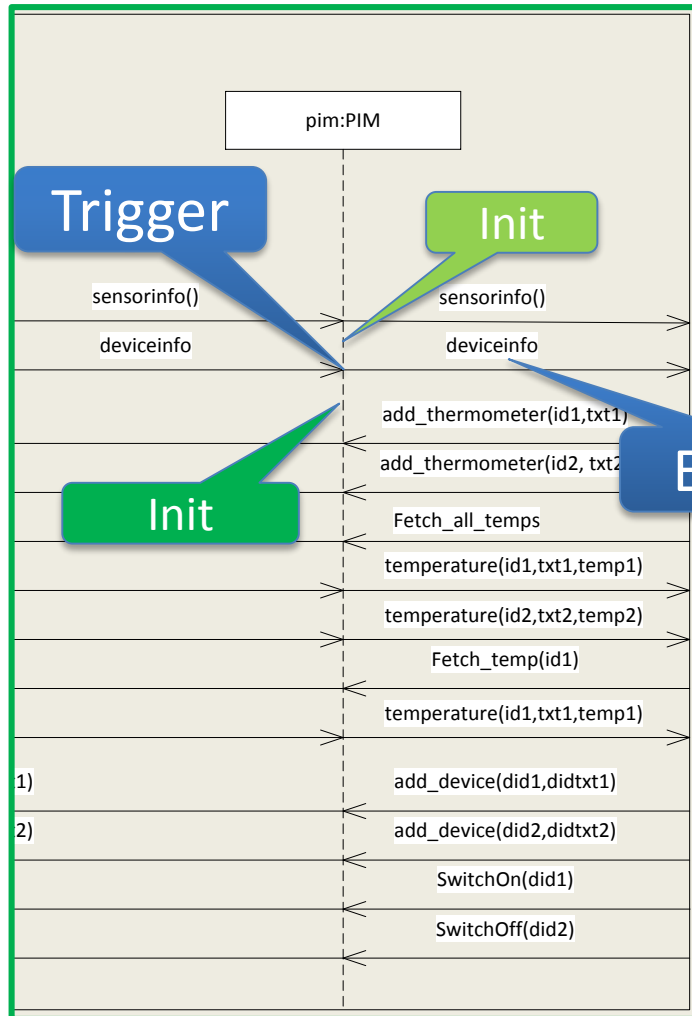
```
statechart PIM_behavior init Init {
    state Init {
        transition -> Init
        event snsr:get_sensor?sensorinfo
        action do
            human_output!sensorinfo(snsr.model,snsr.proto,snsr.sid,snsr.dataTypes,snsr.temperature,snsr.humidity,snsr.timeStamp)
        end
        transition -> Init
        event dvcs:get_sensor?deviceinfo
        action do
            human_output!deviceinfo(dvcs.did,dvcs.name,dvcs.model,dvcs.proto, dvcs.ttype,dvcs.meth,dvcs.lastCmd,dvcs.lastValue)
        end
```
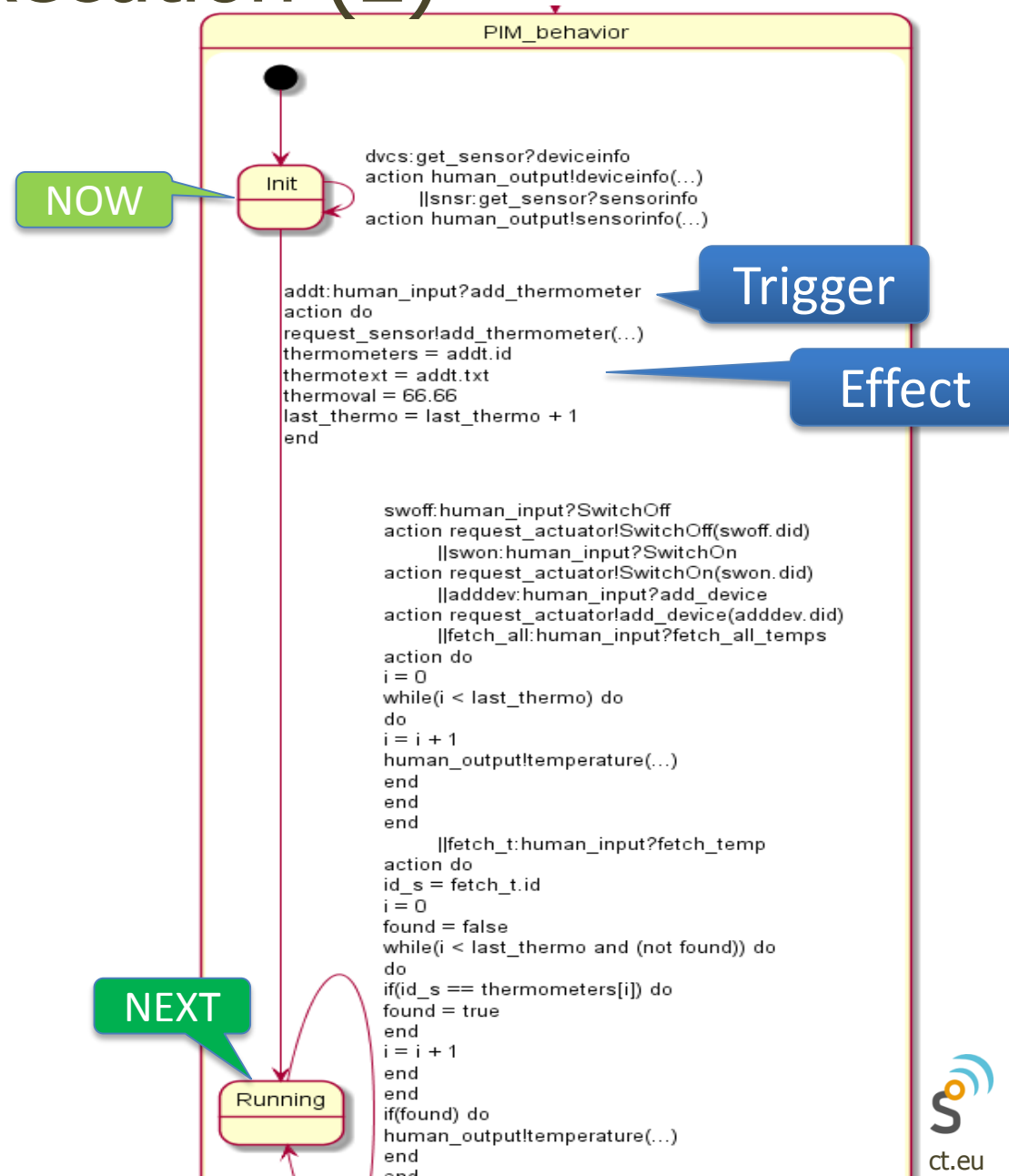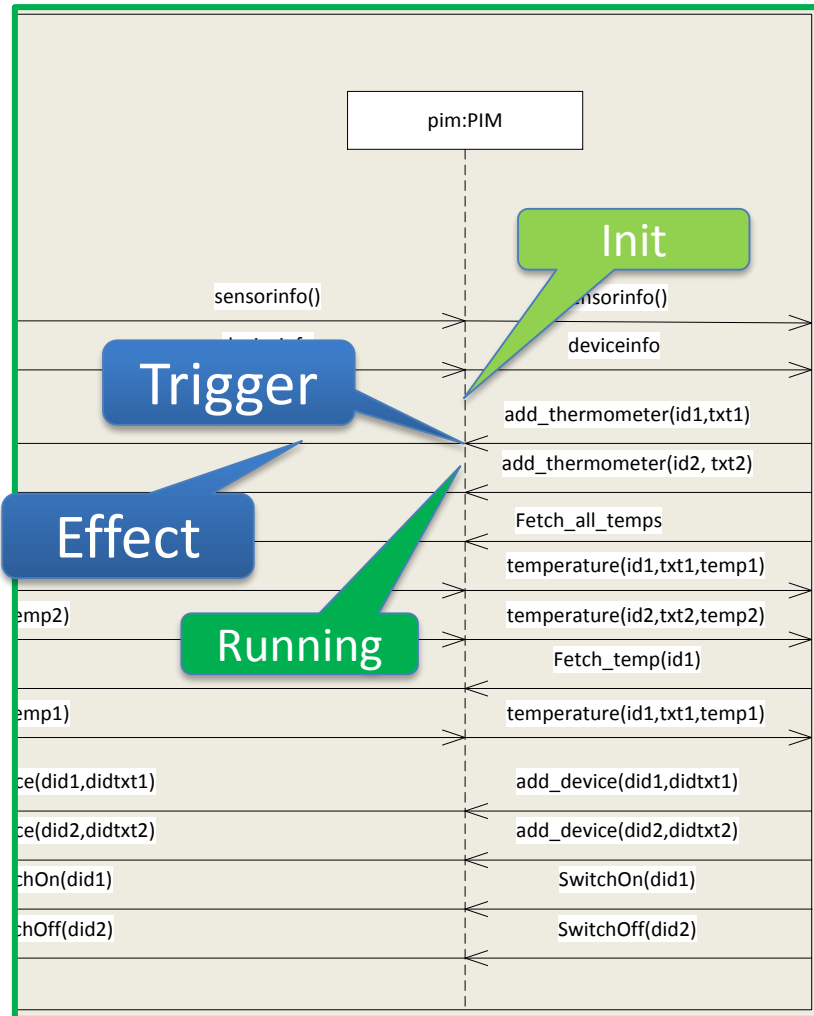
Consistency 3: Parts/instances and Lifelines (+connect)


```thingml
import "psm_sim.thingml"
import "pim.thingml"
import "io.thingml"
import "javatimer.thingml"

configuration CPS {
instance tlstick:TellstickManager
instance T1:ThermometerSet
instance onoff1:OnOffSet
instance pim:PIM
instance myself:Human
instance timer : TimerJava
```

http://heads-project.eu

Consistency 4: Execution (1)

Consistency 4: Execution (2)

Consortium