Performance Analysis of Domain Specific Models

Antonio Vallecillo

[Javier Troya, Francisco Durán, J.E. Rivera]

Atenea Research Group

IPCE 2011, Karlsruhe, Germany. March 2011

Preliminaries

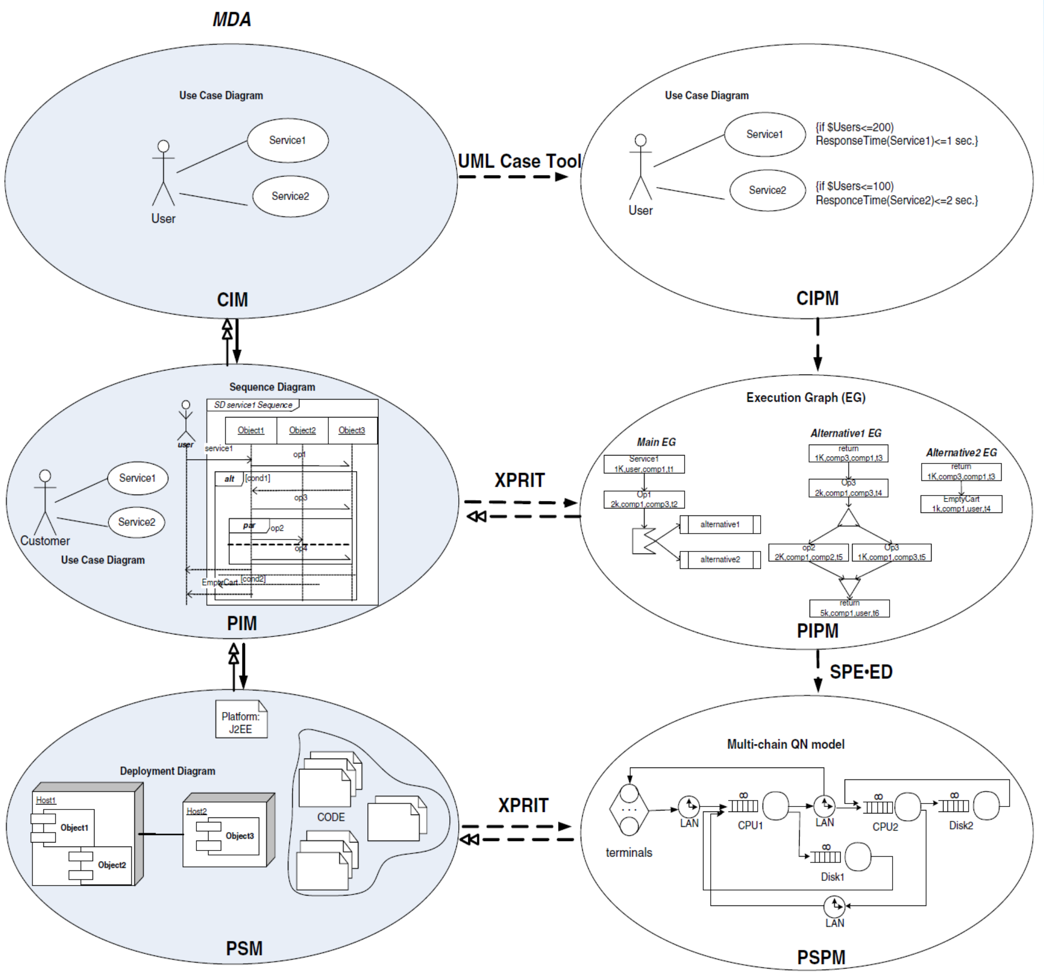
- **Model (of a <X>)**: A representation or specification of a <X> from a given point of view and with a particular purpose
- **Prototype model**: A functional model of a <X>, where the emphasis in on testing – e.g., to verify the design
- **Domain Specific Model**: A model written in a domain specific language
- **Domain Specific Language**: A language which offers concepts and notations closer to the domain experts, at an appropriate level of abstraction, and with a particular purpose
- **Model Transformation**: An algorithmic specification (declarative or operational) of the relationship between models

Preliminaries

- **Performance Analysis:** the process of evaluating how a particular system is functioning (or will work), with the aims to
  - ensure that the system is working at optimum efficiency;
  - identify and correct issues that may negatively impact that efficiency;
  - help the engineer adjust components so that they make the best use of available resources.

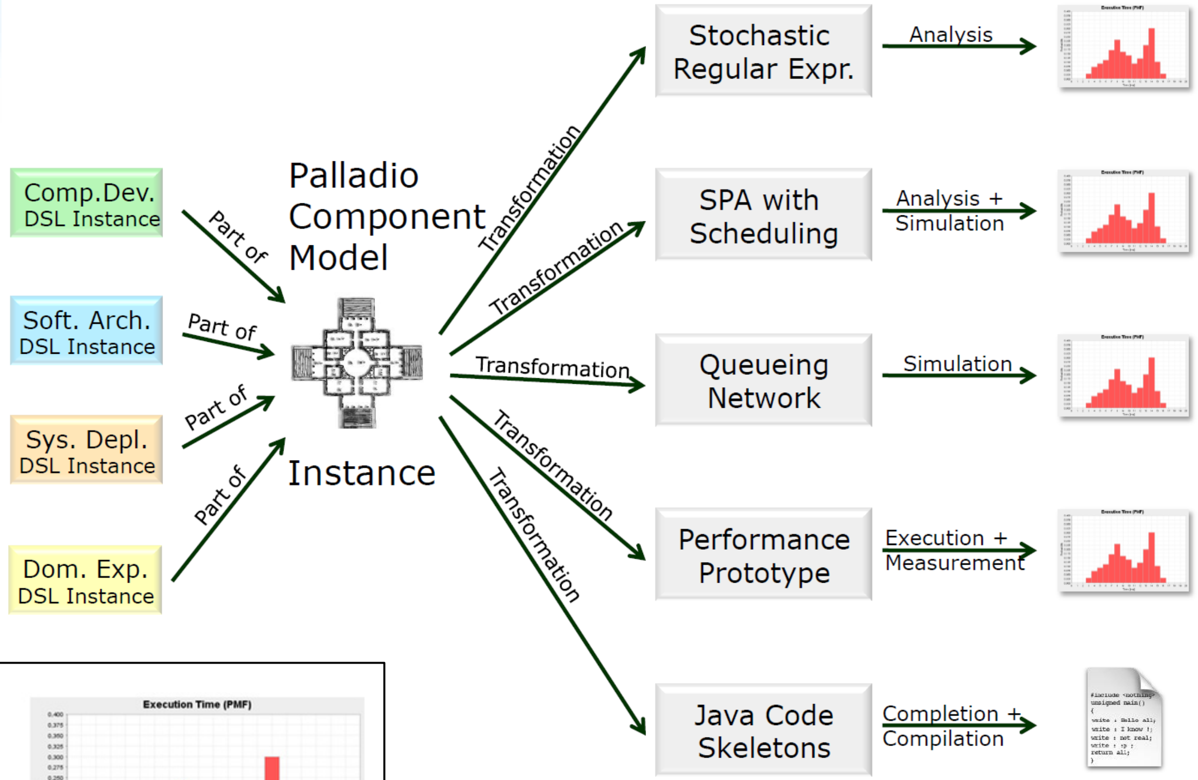
Model-Driven Performance Analysis

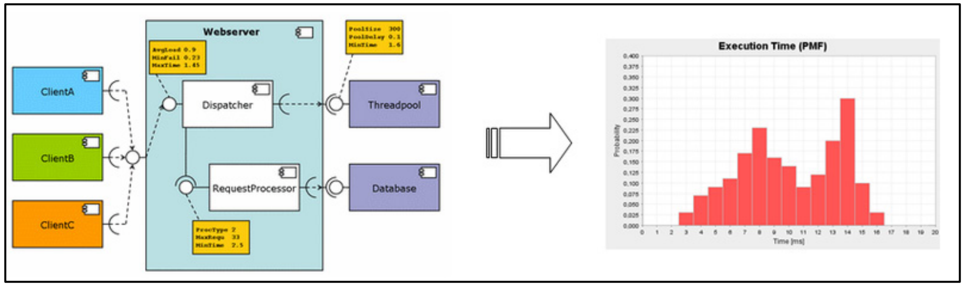
- PRIMA-UML/KLAPER



Model-Driven Performance Analysis

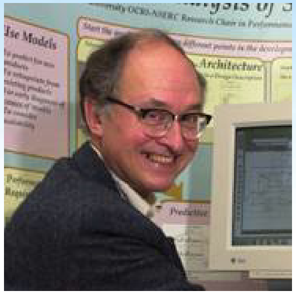
- Palladio Component Model

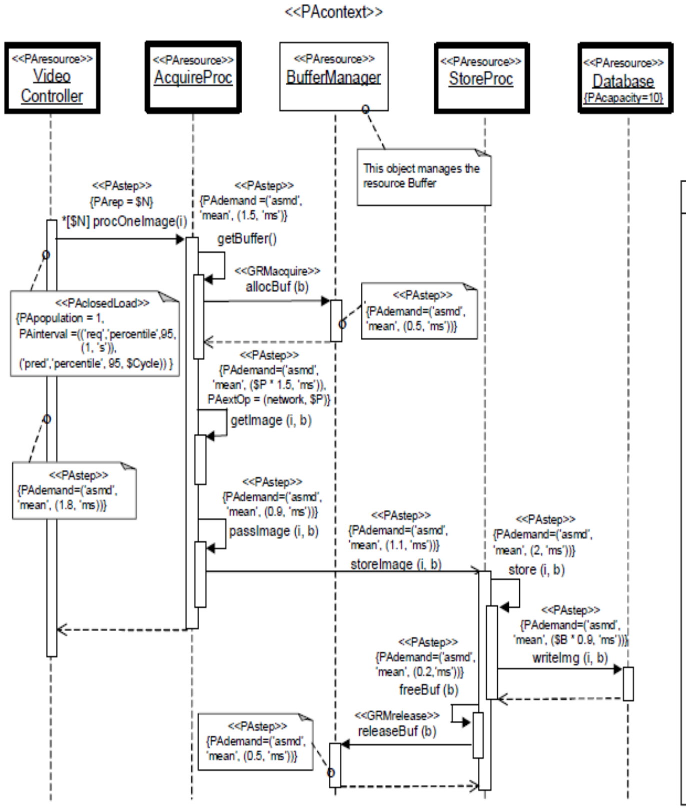


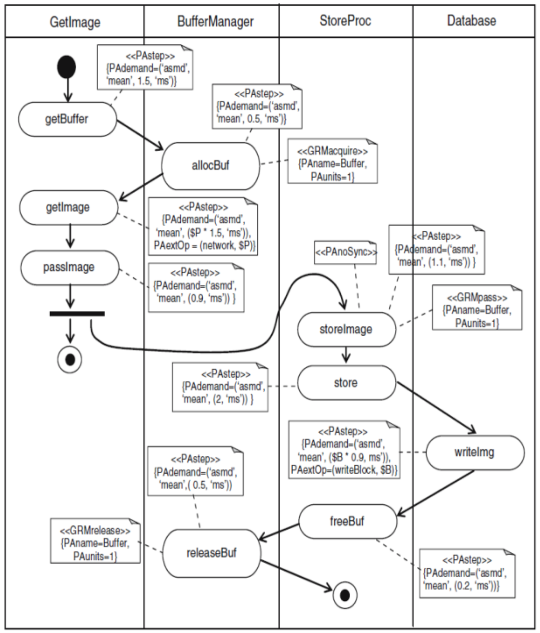


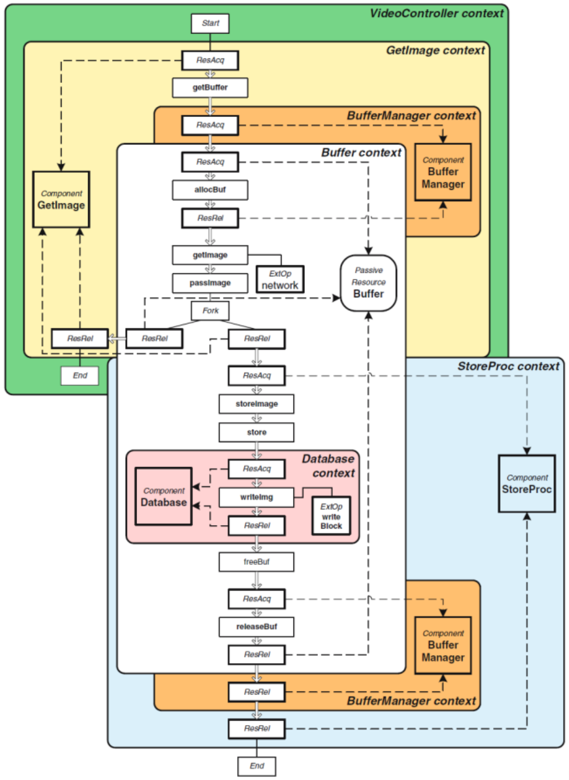
Model-Driven Performance Analysis

- Core Scenario Models





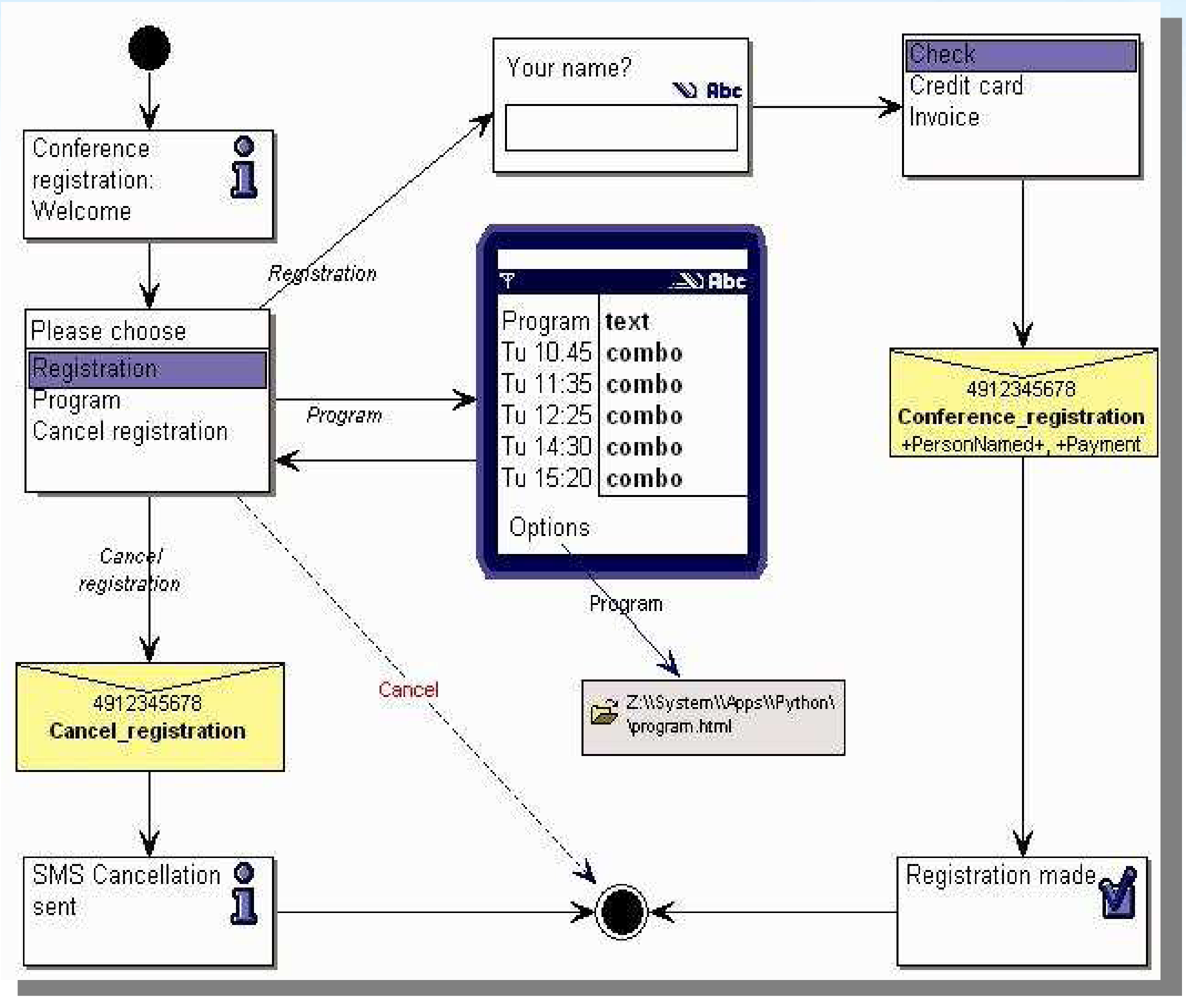




Domain Specific Modeling Languages (DSML)

- Languages for representing different **views** of a system in terms of models
- Higher-level **abstraction** than general purpose languages
- Closer to the **problem domain** than to the implementation domain
- Closer to the **domain experts**, allowing modelers to perceive themselves as working directly with domain concepts
- Domain **rules can be included into the language** as constraints, disallowing the specification of illegal or incorrect models.

An example of a DSM



The design for a conference application intended to run on a Symbian/S60 phone.

[Borrowed from Juha-Pekka Tolvanen, “Domain-specific Modeling: Making Code Generation Complete” http://www.devx.com.]

Visual DSMLs

- VDSMLs tend to offer substantial gains over conventional textual languages
  - Formal studies show significant benefits for novices
  - Increasing number of VDSMLs being defined

- But not a panacea: every notation has advantages and disadvantages
  - [Not the subject of this talk]

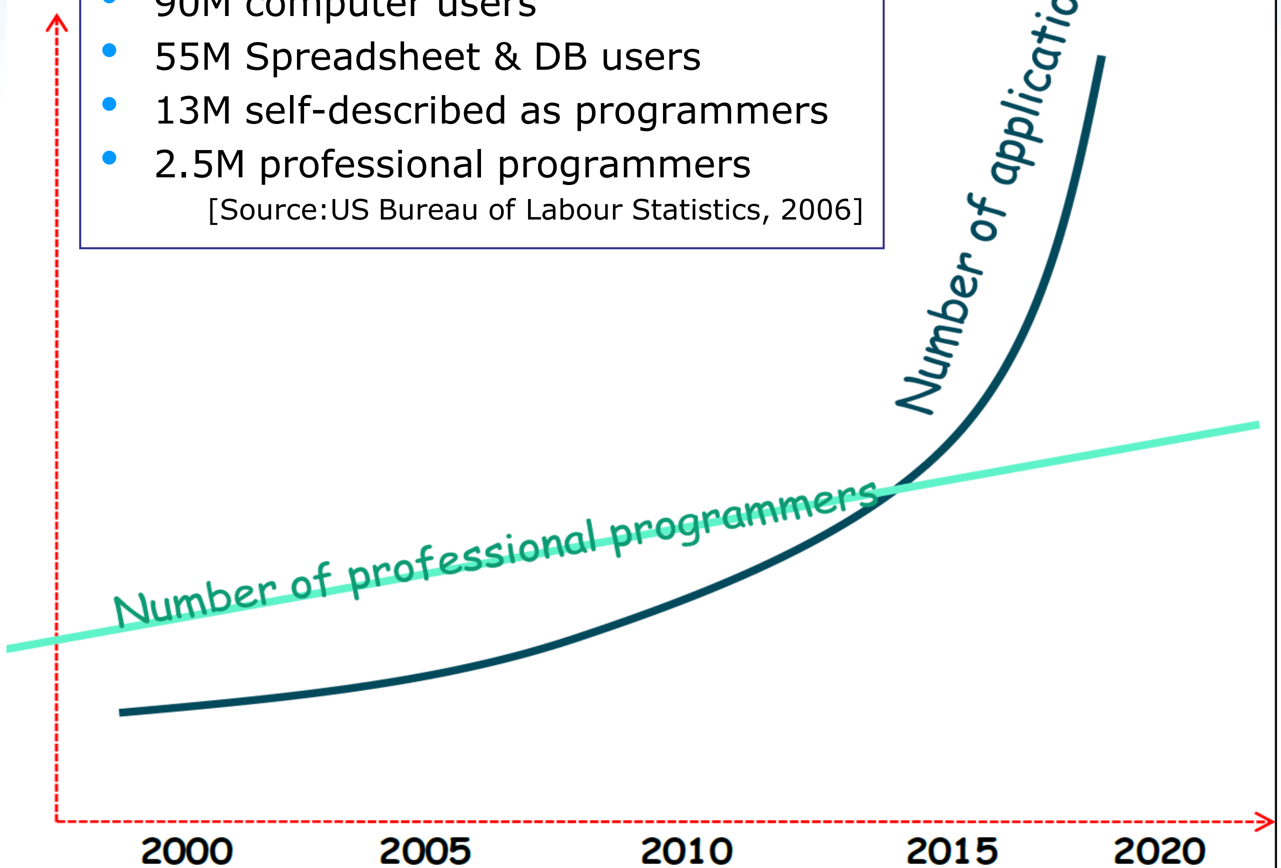
[T. Green, M. Petre “Usability Analysis of Visual Programming Environments: a ‘cognitive dimensions’ framework” *JVLC*, 1996]
[Kirsten whitley “Visual programming languages and the empirical evidence for and against”, *JVLC* 1997]
[R. Navarro-Prieto, J. Cañas “Are visual programming languages better? The role of imagery in program comprehension”, *IJHCS* 2001]


[Thomas R.G. Green & Marian Petre]

“The impossible equation”

USA (estimates for 2012):
- 90M computer users
- 55M Spreadsheet & DB users
- 13M self-described as programmers
- 2.5M professional programmers

[Source:US Bureau of Labour Statistics, 2006]



[J. Bezivin, Keynote at JISBD 2009]

End-user Programming[Modeling]

- Most software creators are not software professionals

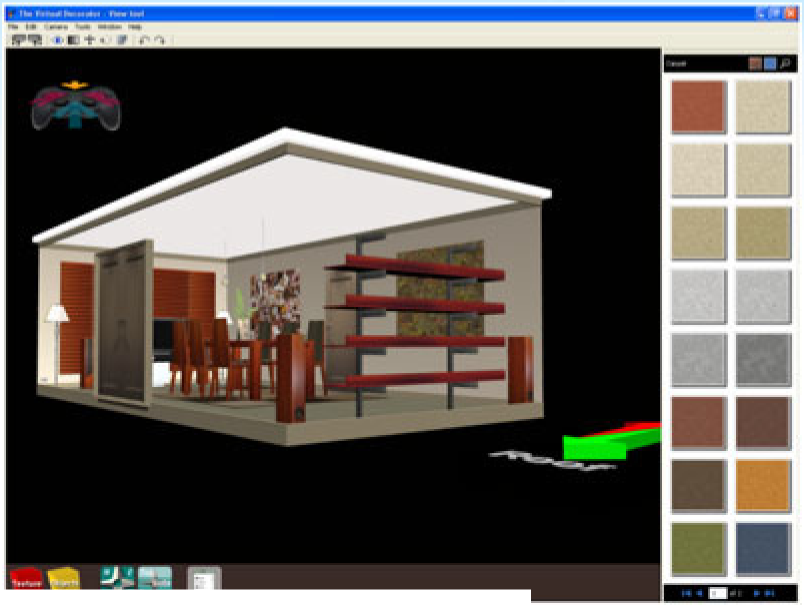
  - End users are participants and developers, not passive consumers
  - They do not reason about software like professionals

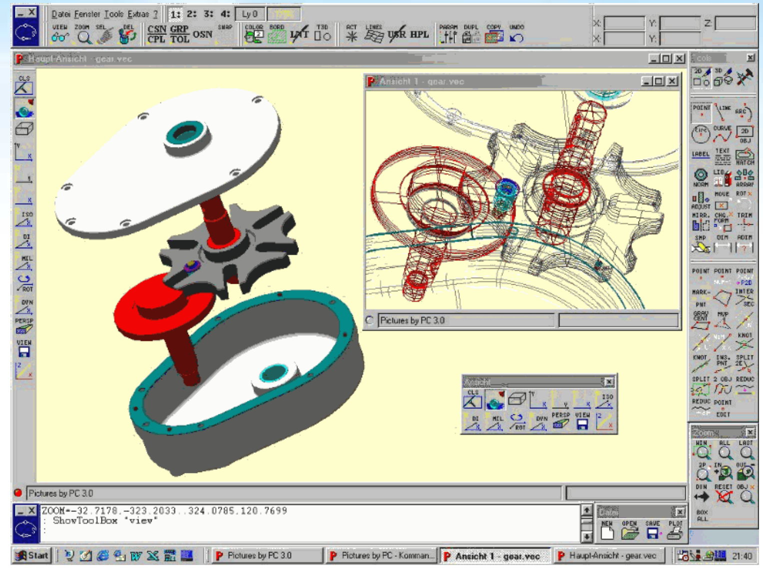
  [Mary Shaw, *The Challenge of Pervasive Software to the Conventional Wisdom of Software Engineering, ESEC-FSE09*]

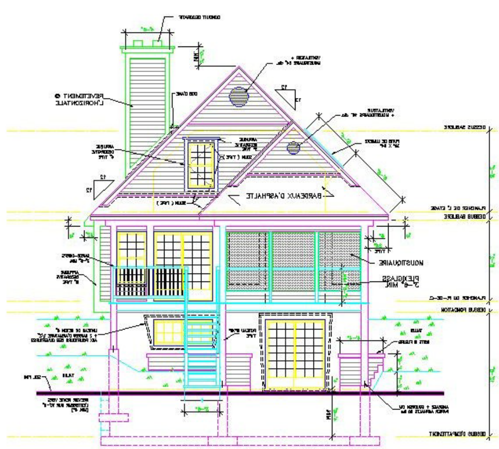
- End users are not "casual," "novice" or "naive" users; they are people such as chemists, librarians, teachers, architects, and accountants, who have computational needs and want to make serious use of computers, but who are not interested in becoming professional programmers.

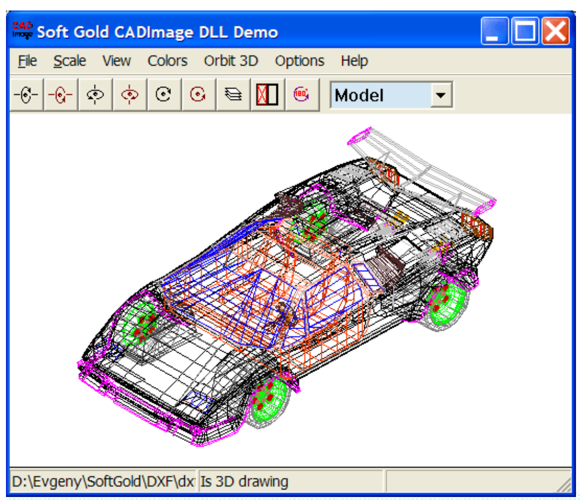
[Bonnie A. Nardi, *A Small Matter of Programming*. MIT Press, 1993]

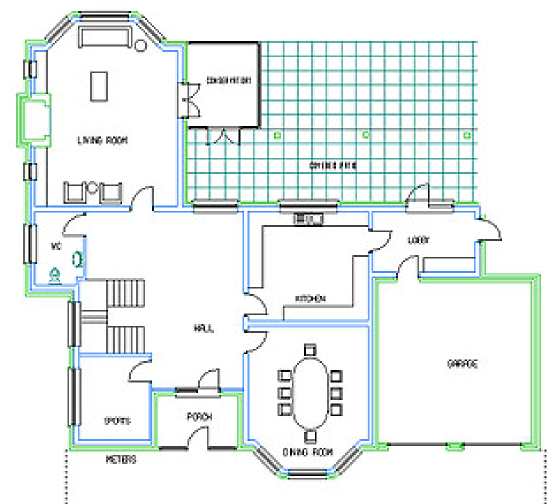
End-user (Visual) Modeling



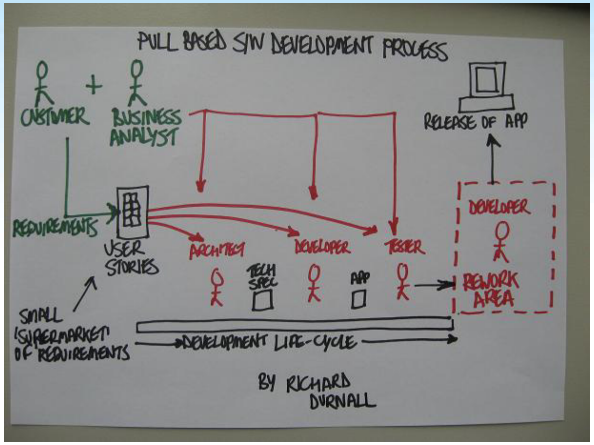


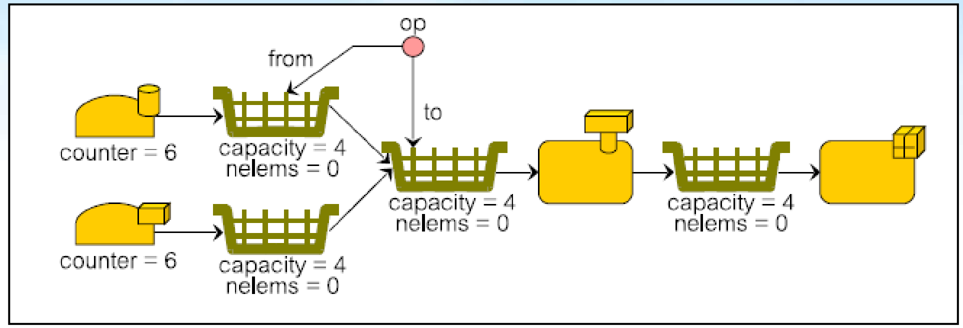


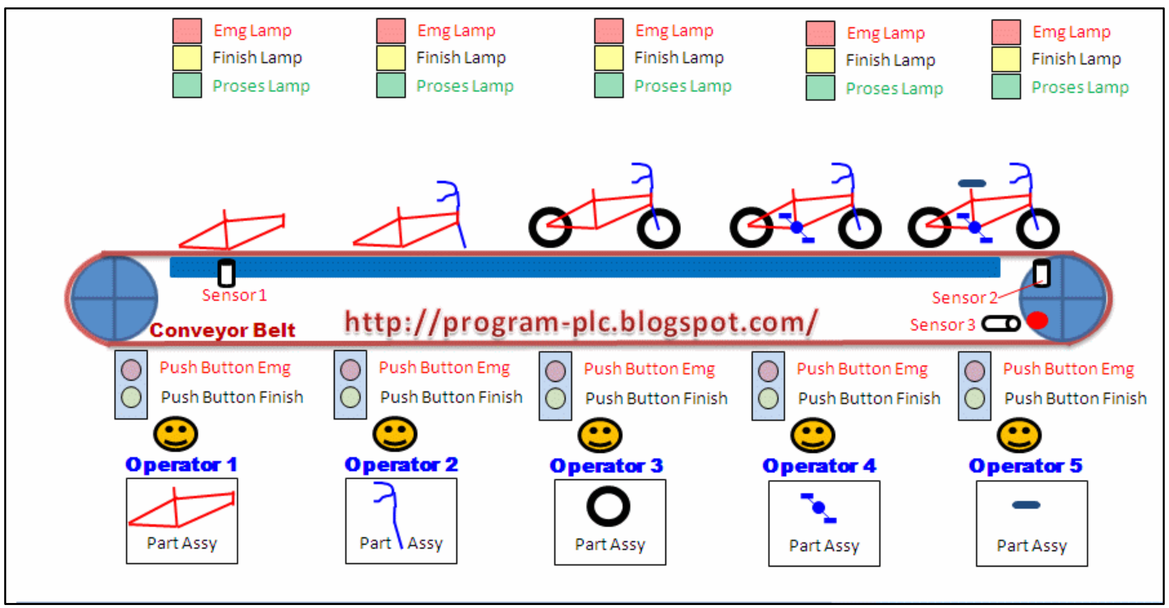




Production systems







DSMLs are starting to proliferate

- They allow users to model their systems at an appropriate level of abstraction
- Some of them allow more than “documentation”
  - Code generation
  - Animation
  - Simulation
  - …
- **Very few allow specification and analysis of the Quality Properties (NFPs) of modeled systems**
  - QoS usage and management constraints: performance, reliability, resource consumption and allocation, etc.

How to conduct
Performance Analysis on

> High-Level,
> Domain-specific,
> End-user defined

models?

Requirements for PA of DSM

- Notations for describing systems must be:
  - Simple and intuitive
  - Close to the problem domain
  - Close to the domain experts’ language
- Models must be:
  - Abstract, yet precise
  - Executable (to, at least, prototype systems)
- QoS notations must be:
  - Simple and precise, yet expressive
- Analysis results and feedback must be:
  - Understandable and easy to manage

“Being abstract is something profoundly different from being vague... The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise.”

Edsger Dijkstra

Current notations for DSMs

- Notations for describing systems ~~must be~~ normally are:
  - ~~Simple and intuitive~~ Complex
  - ~~Close to the problem domain~~ Close to the solution domain
  - ~~Close to the domain experts' language~~ General purpose
- Models ~~must be~~ normally are:
  - ~~Abstract, yet precise~~ Too detailed, imprecise
  - ~~Executable~~ Non-executable
- QoS notations ~~must be~~ normally are:
  - ~~Simple and precise, yet expressive~~ Too complex and low level (more than needed for most end-user DSMLs)
- Analysis results and feedback ~~must be~~ normally are:
  - ~~Understandable and easy to manage~~ Tough to deal with!

"Close to the problem domain"

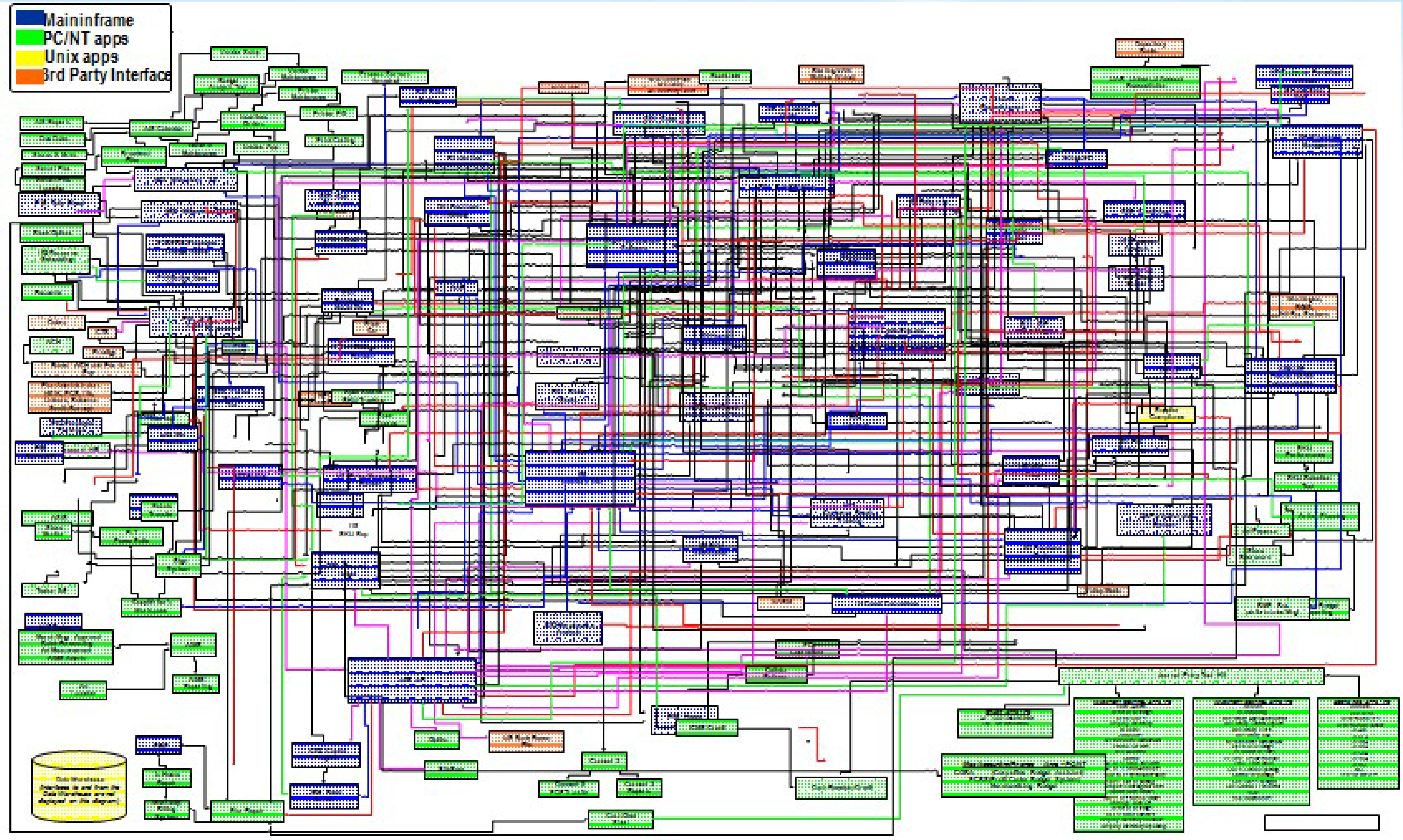
Once upon a time, there was a team leader that was going on holidays. Before leaving, she made the last recommendation to her small team of three young engineers: "For the ongoing project, do not start coding in Java before the UML model is completely finished and you all agree on the model."

On the Monday morning, as soon as she left, one of the engineers told the others about a wonderful discovery he made while twittering in the weekend: a very powerful tool that generates UML diagrams from code. The decision was rapidly taken and all three started coding the problem in Java.

Some days before the end of the leader's holidays, all the Java code was used to generate UML diagrams and both the code and the UML diagrams were handled to the group leader.

She was quite impressed about the level of detail of the UML model and the narrow correspondence between the code and the model.

[Borrowed from J. Bezivin]


Maininframe
PC/NT apps
Unix apps
3rd Party Interface

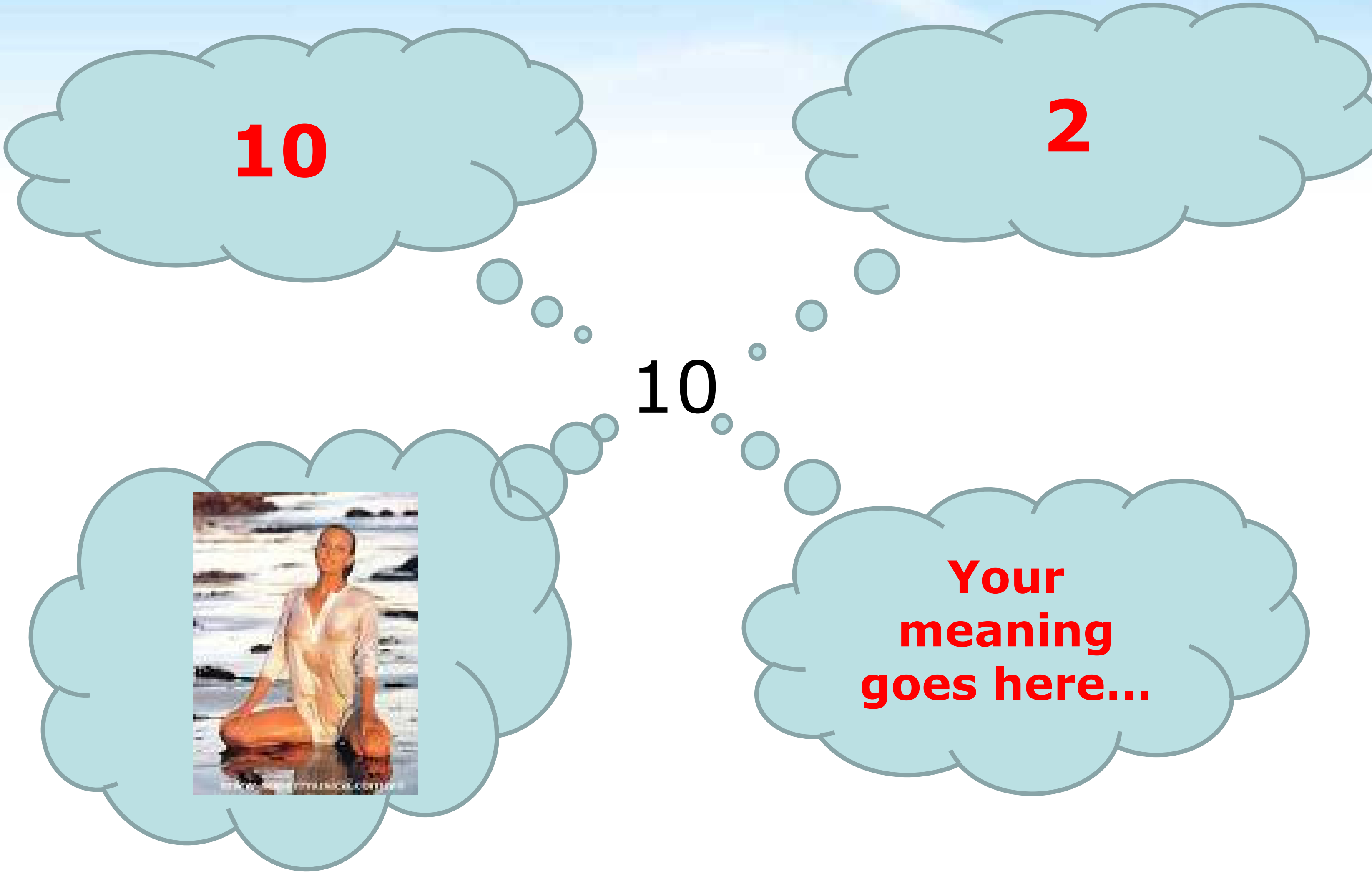
The *precise* meaning of models

10

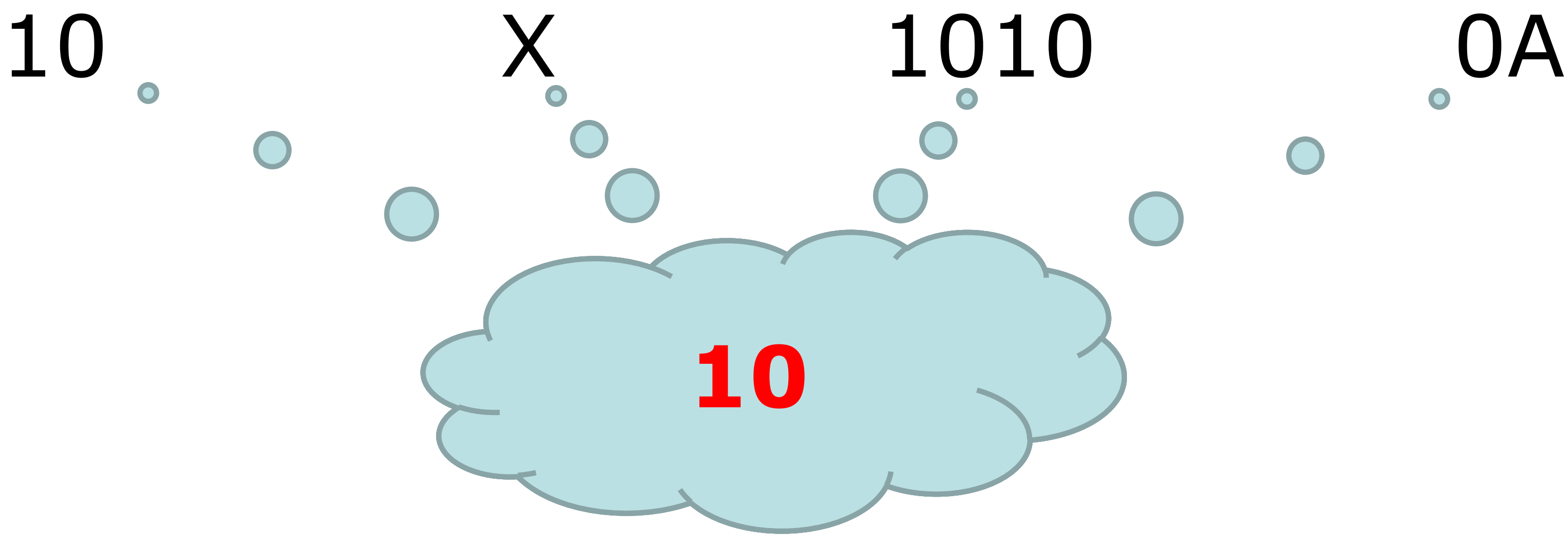
Sure?

“There are only 10 types of people in the world: Those who understand binary, and those who don't”

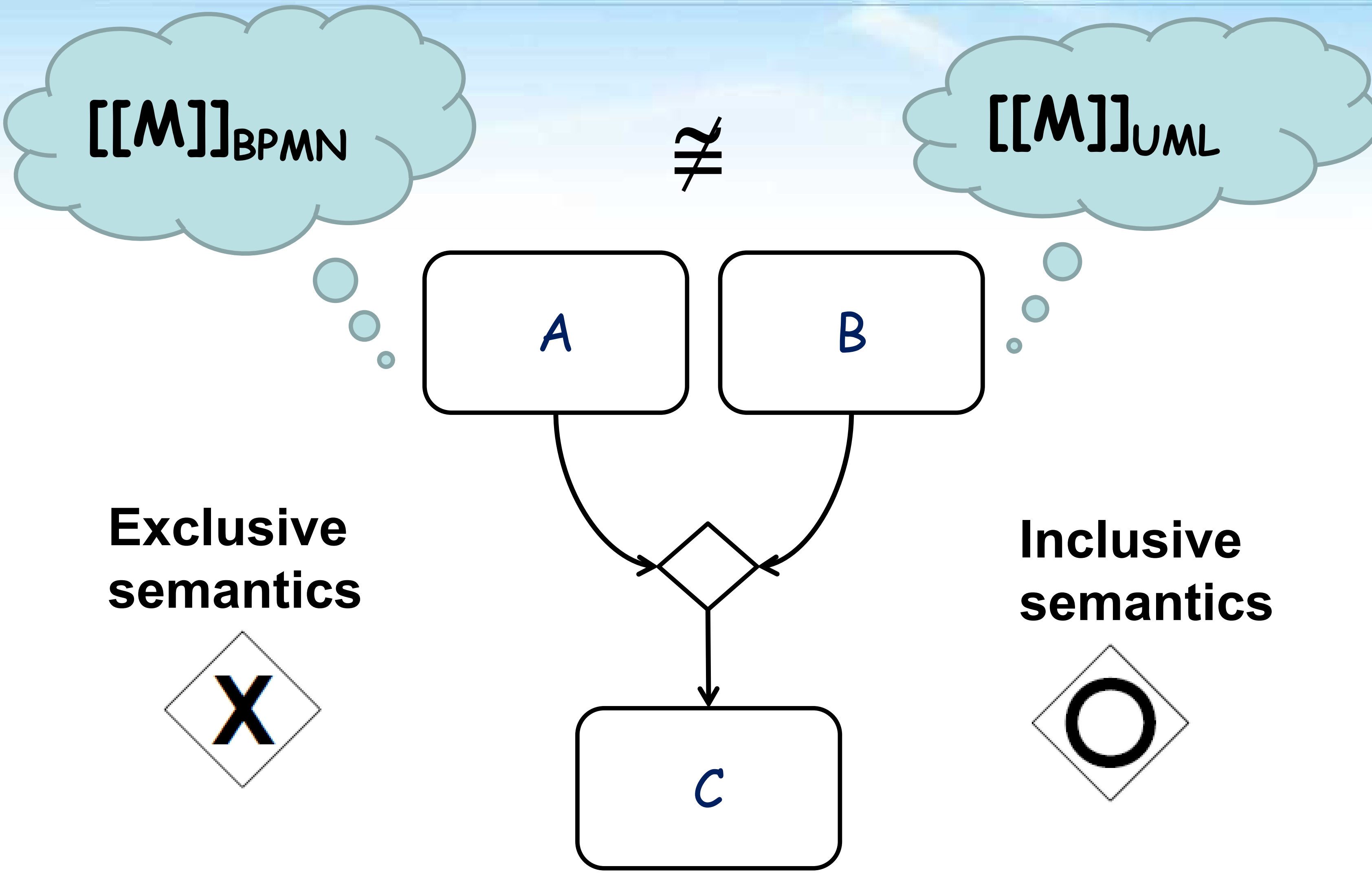
Same model for different concepts



Different models for the same concept

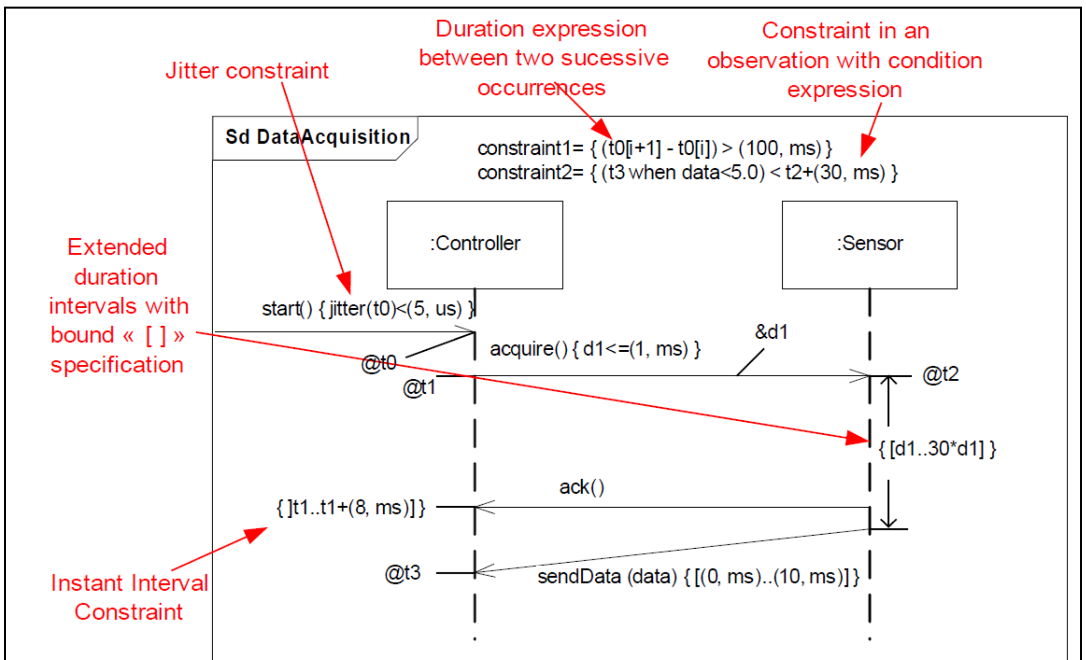


What does this model means?



Current notations for expressing QoS

- Annotations to existing (OO) models
- Very detailed and precise
- Provide connections with analysis models and tools (QNM, SPN, SPA)
- Excellent for modeling systems at certain levels of abstraction

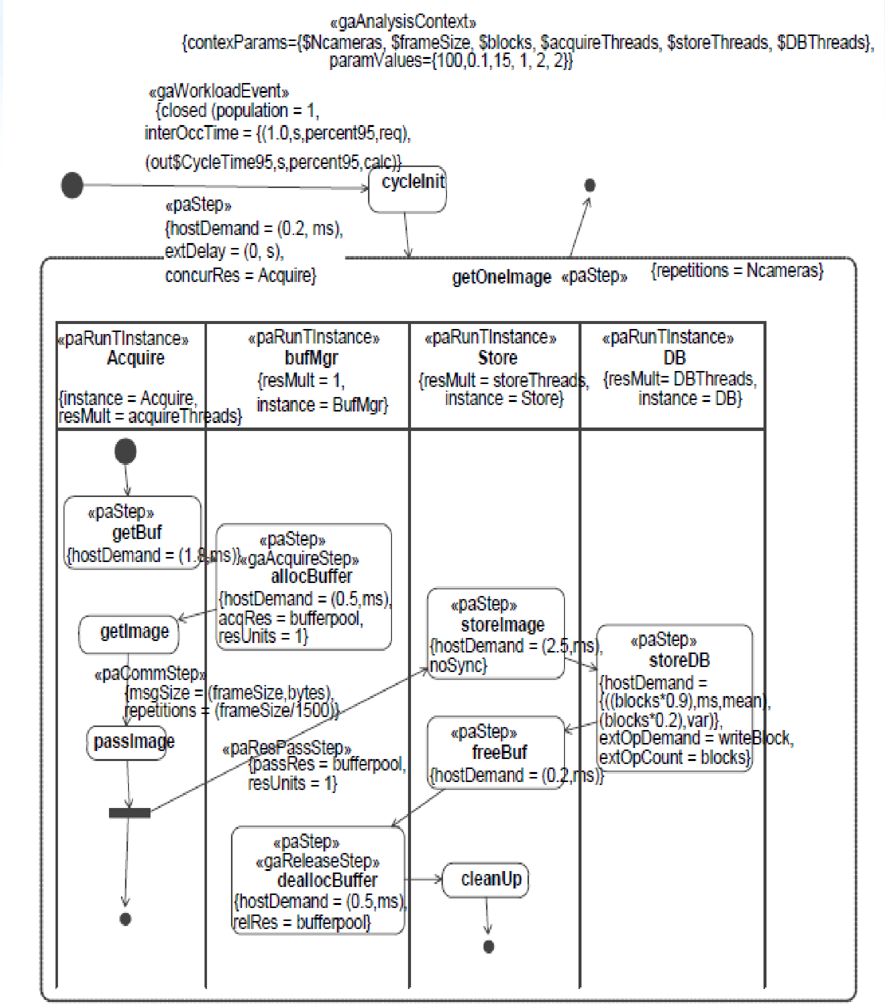


Slide courtesy of Sebastien Gerard, CEA-LETI

[Borrowed from Selic, 2007]

However…

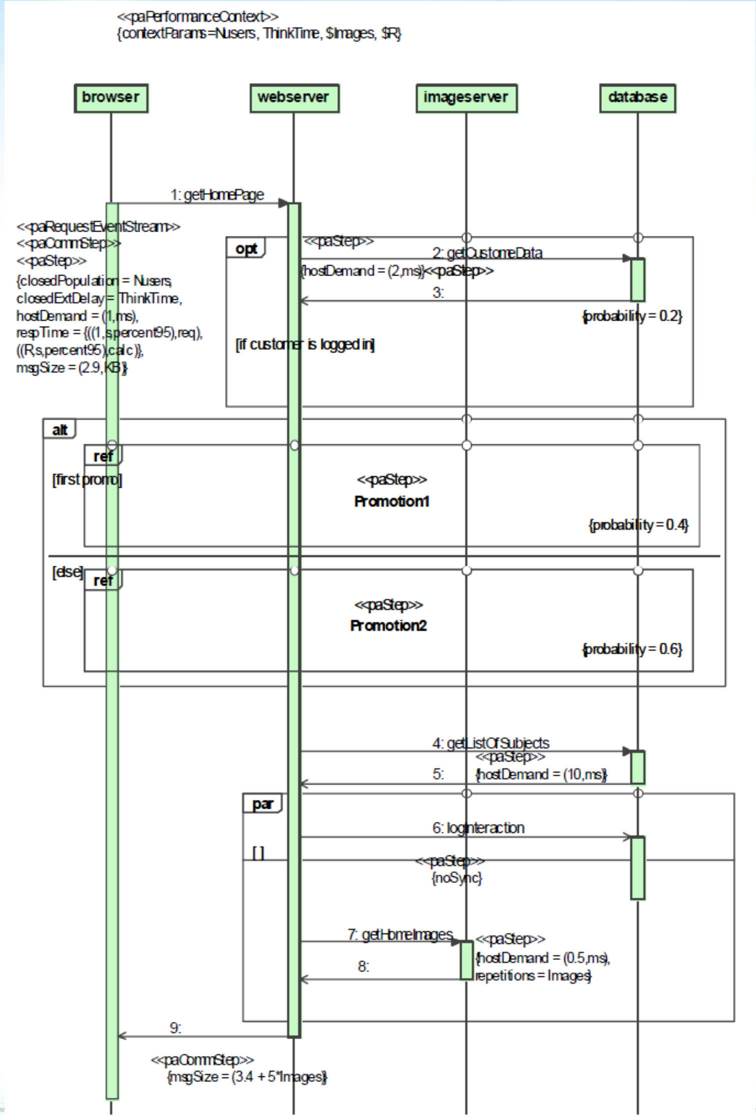
- At a lower level tan needed for most end-user DSMLs
- Complex (to read, write and maintain)
- Tedious, error-prone
- General-purpose
- Object-oriented



However…

- At a lower level tan needed for most end-user DSMLs
- Complex (to read, write and maintain)
- Tedious, error-prone
- General-purpose
- Object-oriented
- No “dynamic” management of QoS (contracts)
  - Negotiation of QoS?
  - Adaptation?
  - End-to-end QoS reqs?

UML MARTE



Require complex quality models

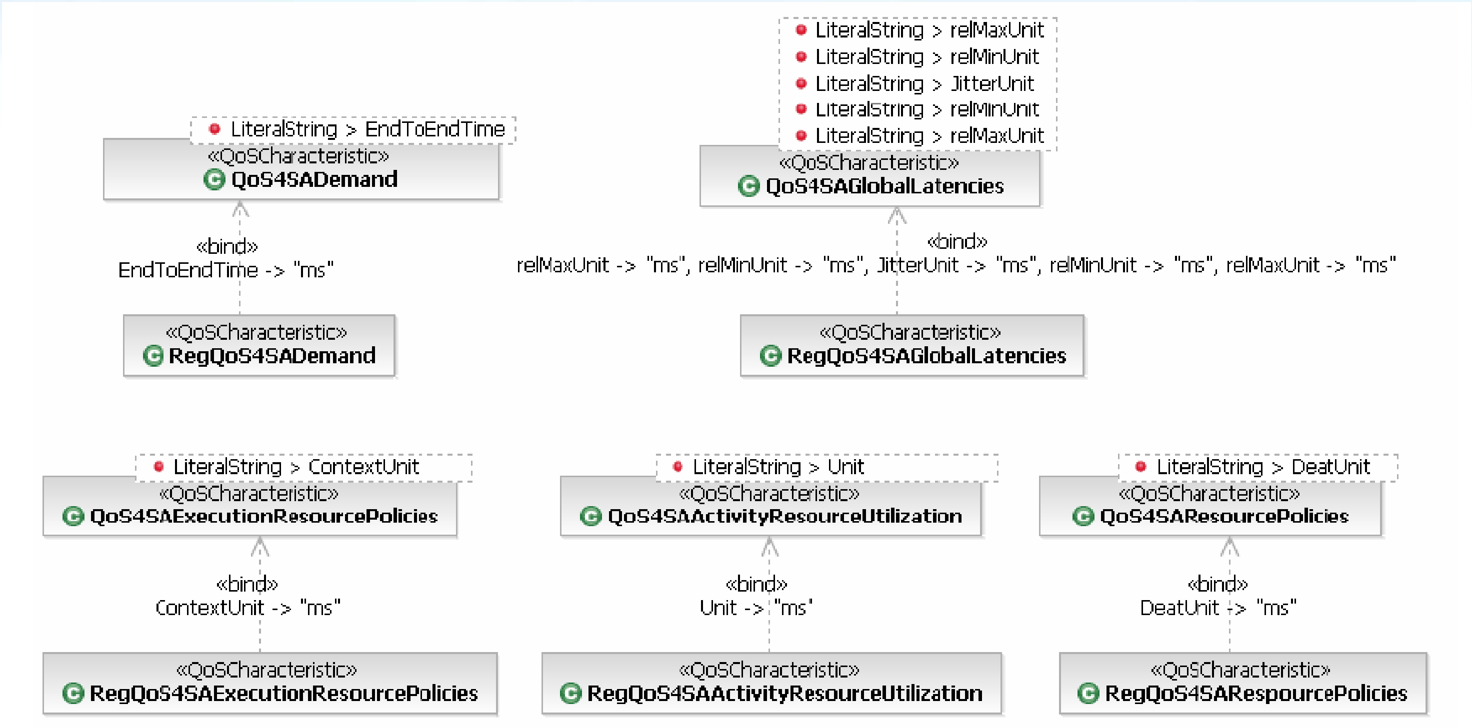


Fig. 6. The QoS Model for the Speed Regulator example

[Borrowed from Espinoza *et al.*]

Easy-to-read specifications?

- Notations to express QoS are strongly dependent on the notations used to express behavior

(Probably this is why it gets so complex in UML?)

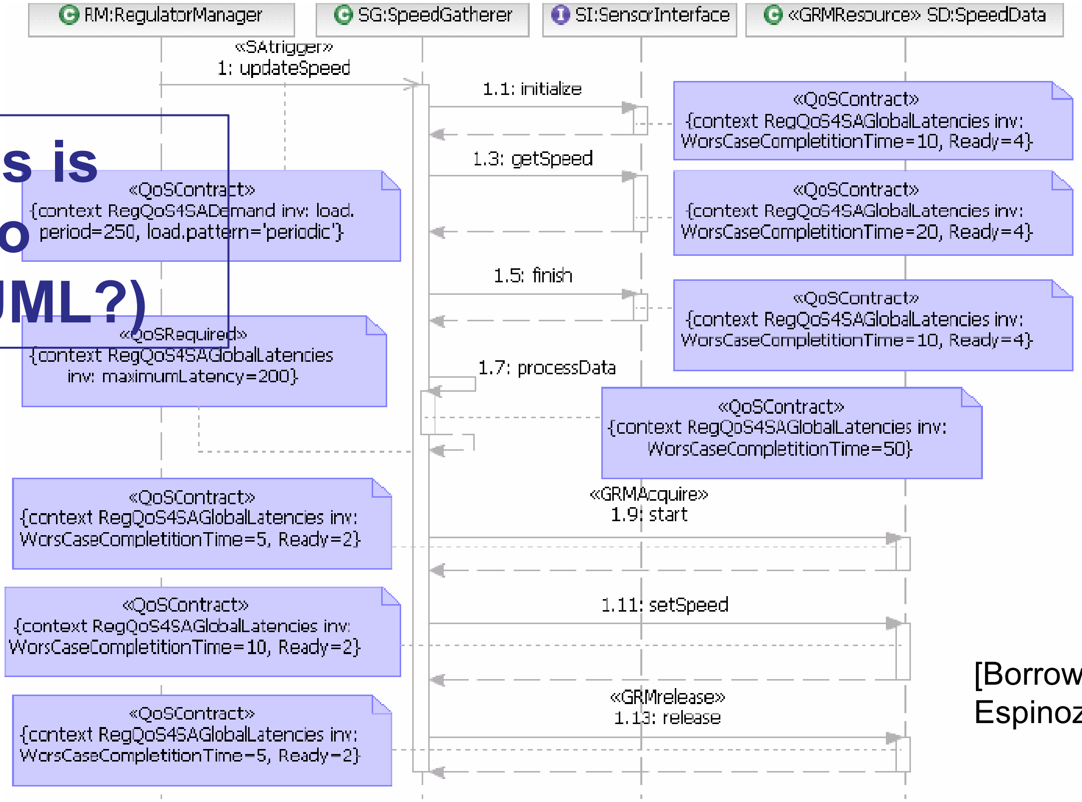


Fig. 7. Sequence Diagram for *updateSpeed* annotated with the UML profile for QoS & FT

[Borrowed from Espinoza *et al.*]

Requirements on QoS specifications

[ISO WD 15935:1998]

- QoS constraints should be **modular** enough to be attached to individual objects.
  - It should be possible to derive the QoS of a composition of objects from the QoS of its component objects.
- The level of QoS should be **observable** so as to allow the development of **monitoring** applications.
  - Through observation, applications become "QoS-aware" and can operate a feedback control loop on the supporting computing or network resources.
- QoS should be **guaranteed** at certain periods
  - The nature of the guarantees can range from deterministic "hard" real-time guarantees, through weaker probabilistic guarantees or "best-efforts" policies.
- QoS should be **negotiable** so that, during the life of the system, some users can quit an application whereas others can appear with different needs.
  - The framework should be flexible enough to allow such QoS management policies as graceful degradation.

How can we specify DSMs?

- How do we express in a precise and abstract manner:
  - Structure
  - Behavior
  - Time-dependent functionality
  - Quality properties (QoS,…)

- Which is the best **notation** for each of those aspects?
  - It depends on the **purpose** of the model…
  - …must have a **precise** meaning
  - …and must allow the **analysis** of the models

Each notation is more apt for a task

MCMLXVII	1.967
+ DLXXIX	+ 579
???	

Each notation is more apt for a task

MCMLXVII		1.967
+ DLXXIX		+ 579
???		2.546

Each notation is more apt for a task

MCMLXVII	1.967
+ DLXXIX	+ 579
MMDXLVI	2.546

How do you solve this problem?

- A 40-years-old man has a daughter and a son. If the difference of age between the kids is 4 years, and the sum of their ages is half of the age of the father, how old are they?

$$
\begin{array}{rl}
 & x - \cancel{y} = 4 \\
+ & x + \cancel{y} = 20 \\
\hline
 & 2x = 24
\end{array}
$$

$$
\begin{aligned}
x &= 12 \\
y &= 8
\end{aligned}
$$

Solution: the older is 12 and the younger is 8

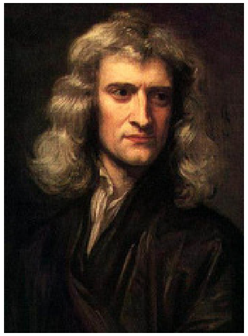
Problems, Notations, Solutions

- An invariant through the history of mature disciplines is the search for notations that allow formulating problems in a language that allows their easy solution

$$\frac{\partial f}{\partial x_i}(a_1, \ldots, a_n) = \lim_{h \to 0} \frac{f(a_1, \ldots, a_i + h, \ldots, a_n) - f(a_1, \ldots, a_n)}{h}.$$

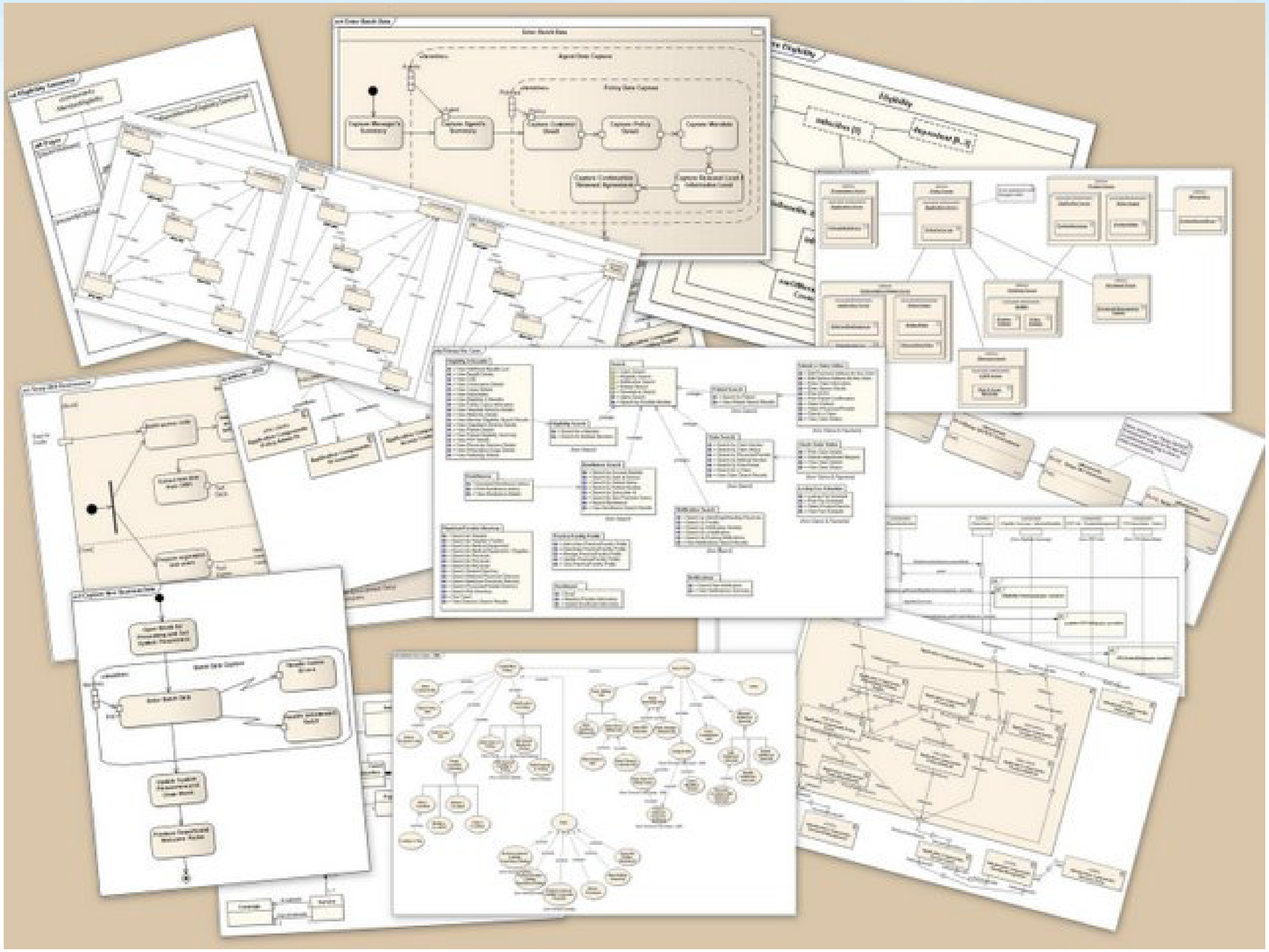
$$\vec{F} = \frac{\mathrm{d}}{\mathrm{d}t}(m\vec{v}) \qquad \int_{-N}^{N} f(x)\,dx \qquad \sum_{n=1}^{\infty} \frac{1}{n^2} \qquad \Box(p \to q) \to (\Box p \to \Box q)$$

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \nabla \cdot \mathbb{T} + \mathbf{f},$$



http://en.wikipedia.org/wiki/History_of_mathematical_notation
http://en.wikipedia.org/wiki/Temporal_logic

Our current software modeling notations



The UML way…

Sauron’s approach to metamodeling
(e.g., OMG’s UML metamodel)

The lord of the Metamodels

(obviously, adapted)

Three notations for the *Structure modelers* under the sky,
Seven for the *Behavior modelers* in their halls of stone,
Tree for *mortal Packagers* doomed to die,
One for the Designer of the Whole system on his dark throne
In the Land of Mof where the shadows lie.
One Metamodel to rule them all, One Metamodel to find them,
One Metamodel to bring them all and in the darkness bind them

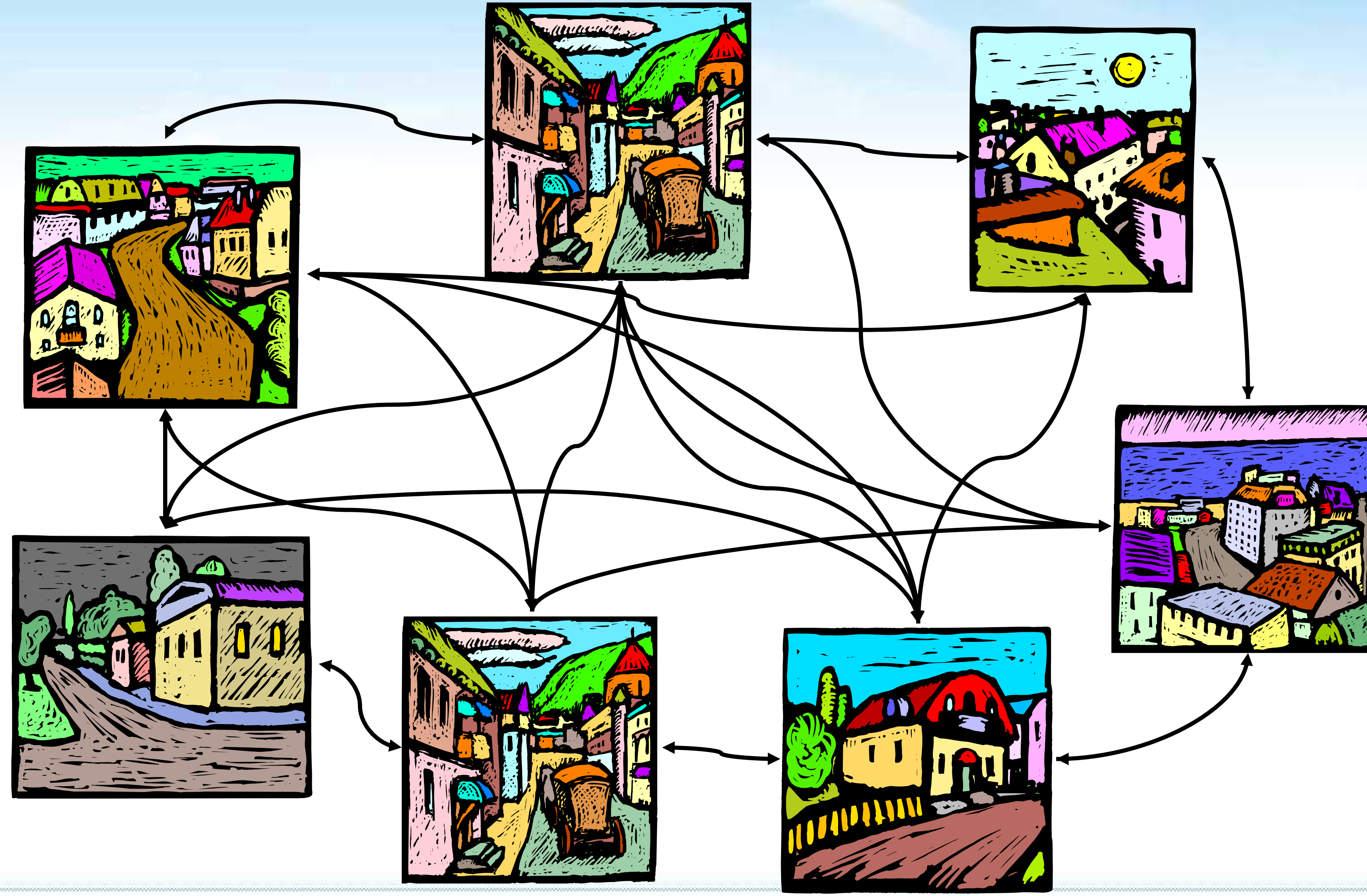
In the Land of Mof where the shadows lie.

- No general purpose language can express all different semantics without becoming a monster
  - Especially under the presence of antagonist semantics (Discrete & continuous; synchronous & asynchronous;...)

"More general does not mean Better. Heterogeneity may be better than generality.
...***Useful semantics imply constraints on designers***."

Edward A. Lee

The village metaphor



Semantic (or "Meaningful") Domains

(vs. "meaningless" Models)

- The meaning of a model **M** is defined by its **interpretation** in a **meaningful** semantic domain **D**.

- Each Semantic Domain has
  - Precise semantics
  - A set of (equivalent) notations
  - A set of analysis tools
  - Underlying logic

- **Semantic Bridges** connect Semantic Domains

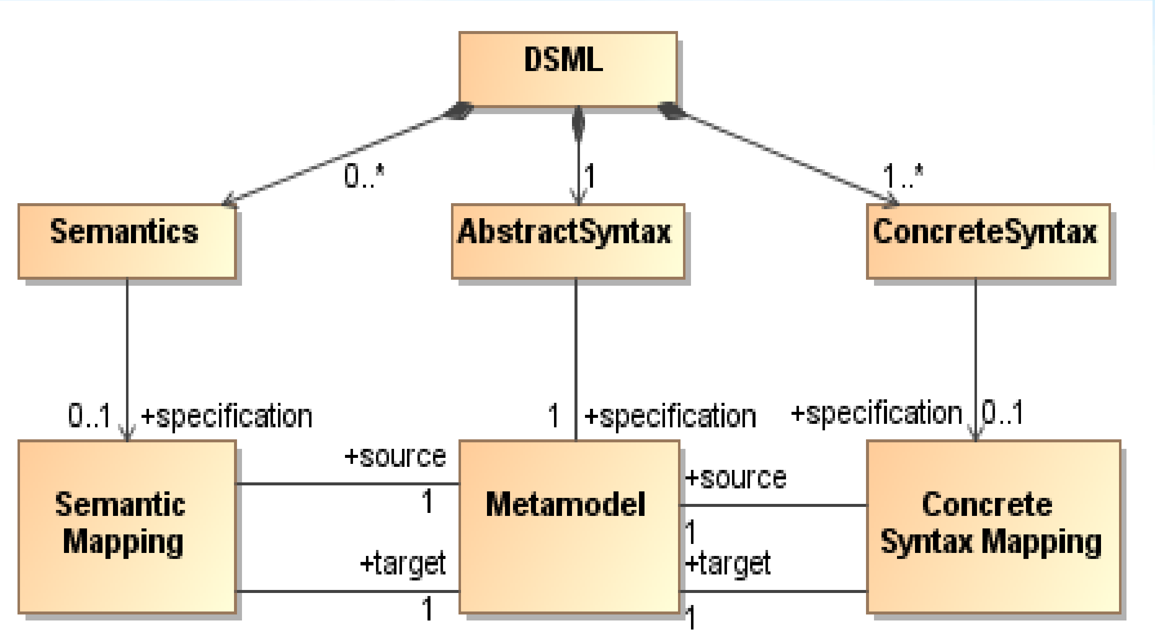
- The Prolog village
- The QNM village
- The Petri Net village
- The Coq village
- The Process Alg village
- The Maude village
- The Z village
- The Modellica village
- The B village
- ...

Expressing semantic bridges

- As **Model Transformations**
  - Possible if correspondences can be expressed as functions
  - Pairwise consistency can be formally studied
    - One form of consistency involves a set of correspondence rules to steer a transformation from one language to another. Thus given a specification S_1 in viewpoint language L_1 and specification S_2 in viewpoint language L_2, a transformation T can be applied to S_1 resulting in a new specification $T(S_1)$ in viewpoint language L_2 which can be compared directly to S_2 to check, for example, for behavioral compatibility between allegedly equivalent objects or configurations of objects [RM-ODP, Part 3]
- As **Weaving Models**
  - Possible if correspondences are just mappings

Semantic Mappings as Model Transformations

- Types
  - Domestic
  - Horizontal
  - Vertical
  - Abstracting
  - Refining
  - Pruning
  - Forgetful
  - …

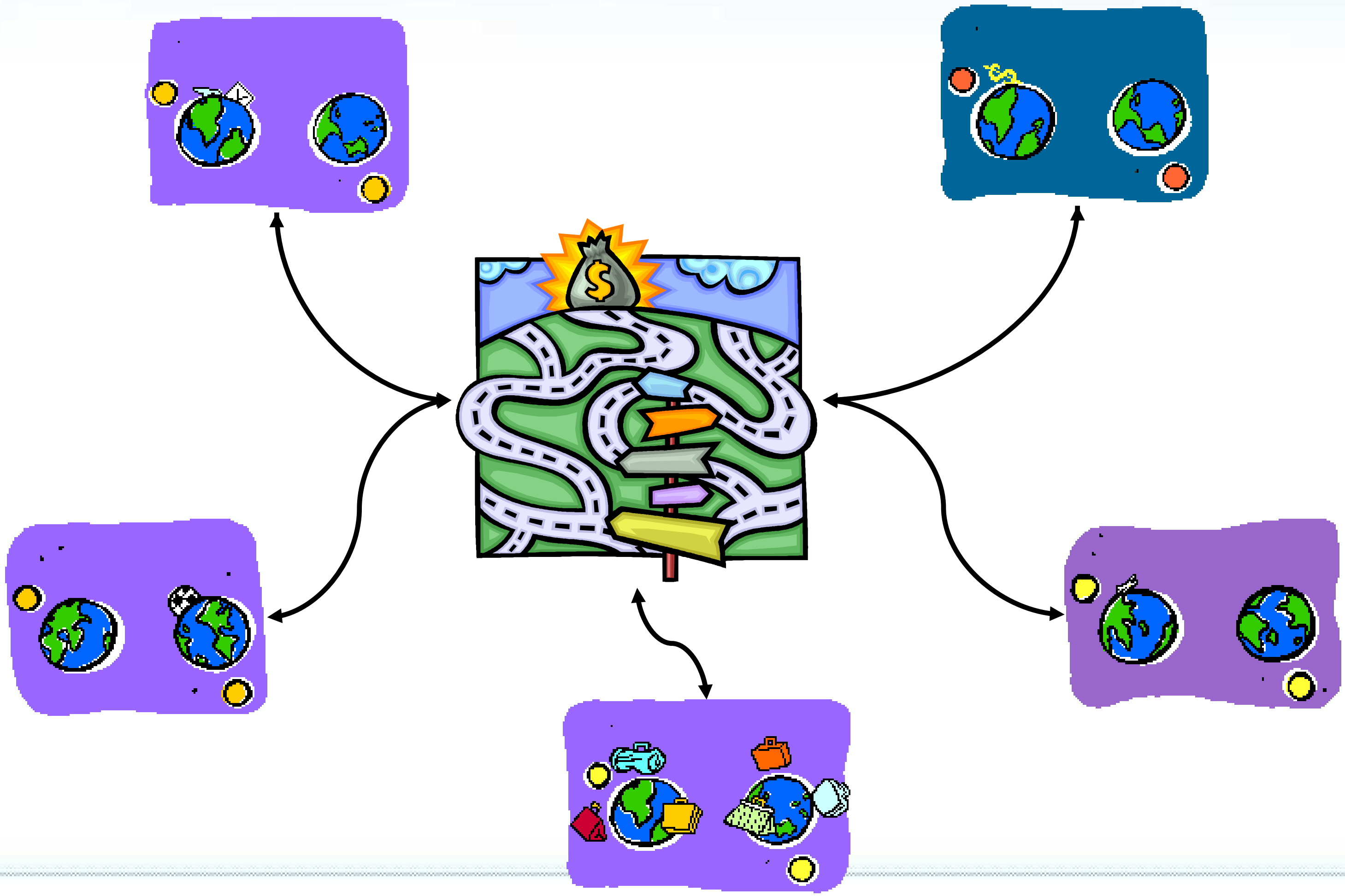


- The relationship between domains **D** and **D'** is defined by a model transformation **T:D->D'**

$$\boxed{[[M]]_{D'} := [[T(M)]]_{D'}}$$

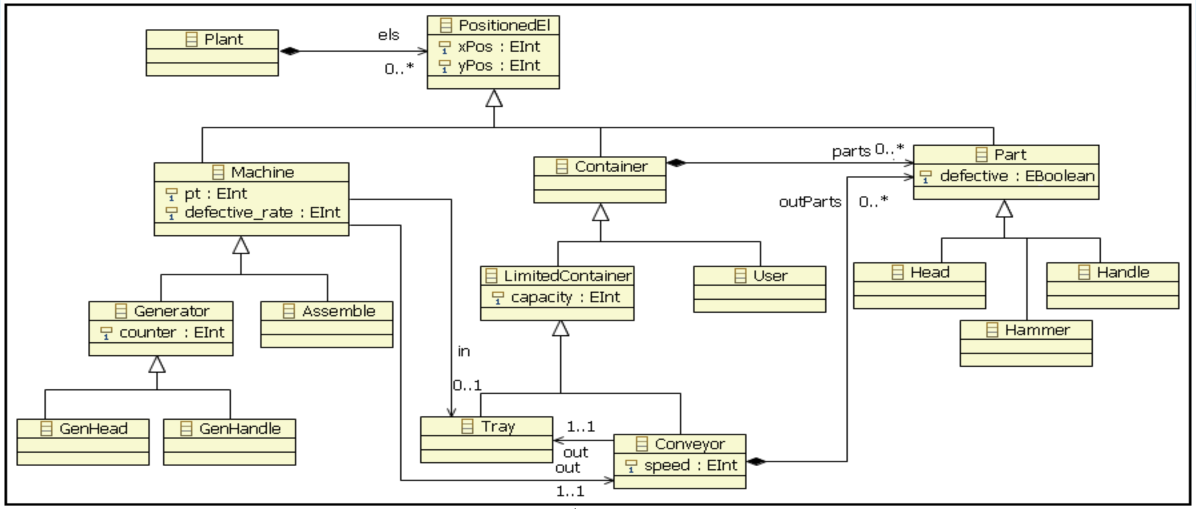
How do we analyse models?

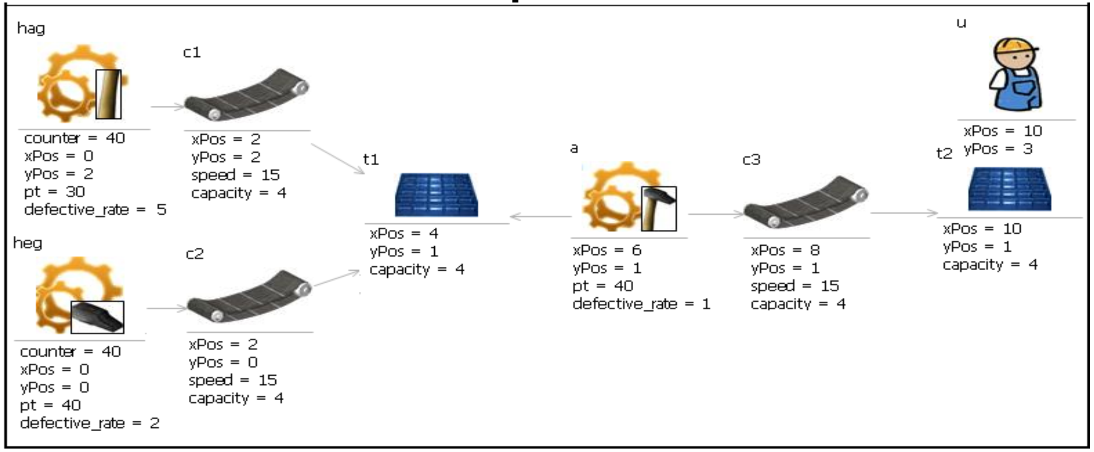
- Crossing the bridges!!!



Some initial experiments…

A Production System Example

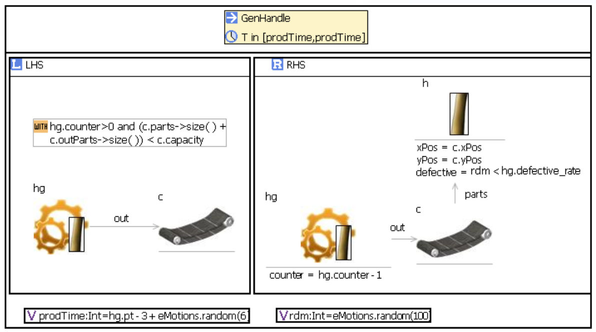


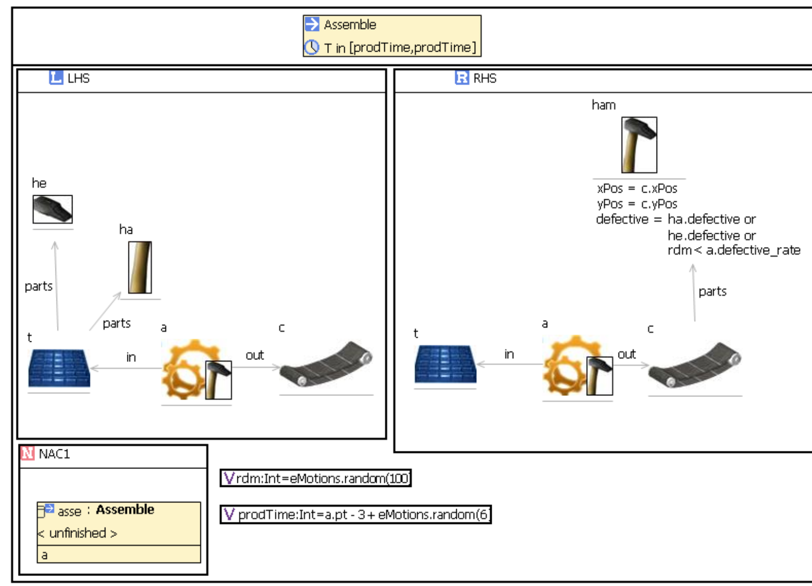


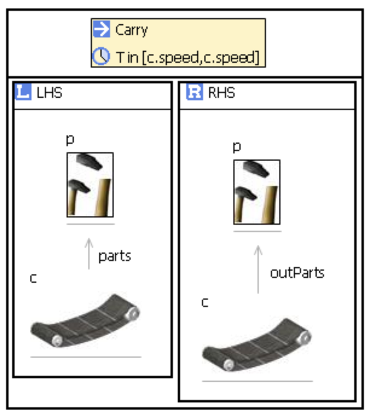
Behavioral semantics

- Using in-place model transformations (Graph Transf.)

$$l\!:\![NAC] \times LHS \rightarrow RHS$$



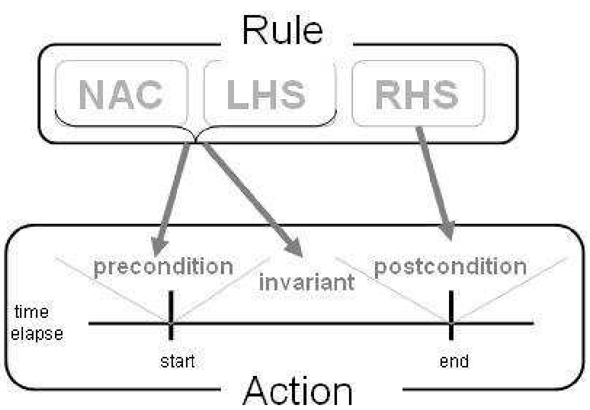




Some essential additions

- **Time**
  - Rule duration 
  - Periodicity, soft scheduling
  - Ongoing rules
  - Access to the Global Time Elapse
  - Time stamps, scheduled actions

$$l:[NAC] \times LHS \xrightarrow{t} RHS$$

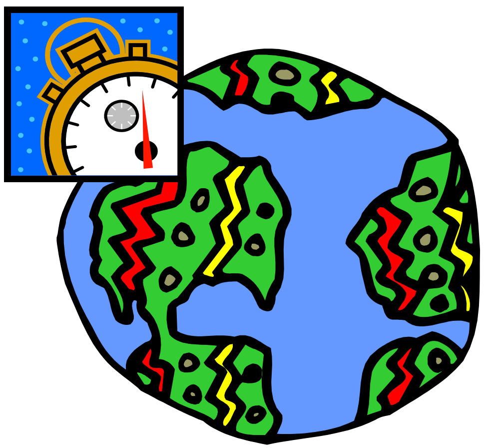


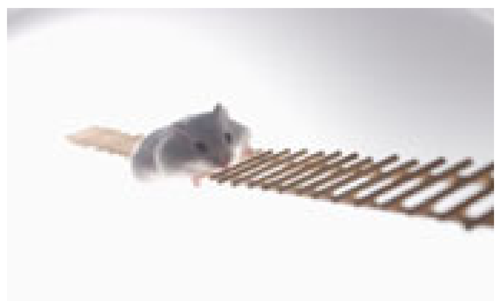
- Specification of **action executions**
  - Without the need of unnaturally modify the metamodel

- **OCL** for attribute calculations and rule conditions

Precise Semantics

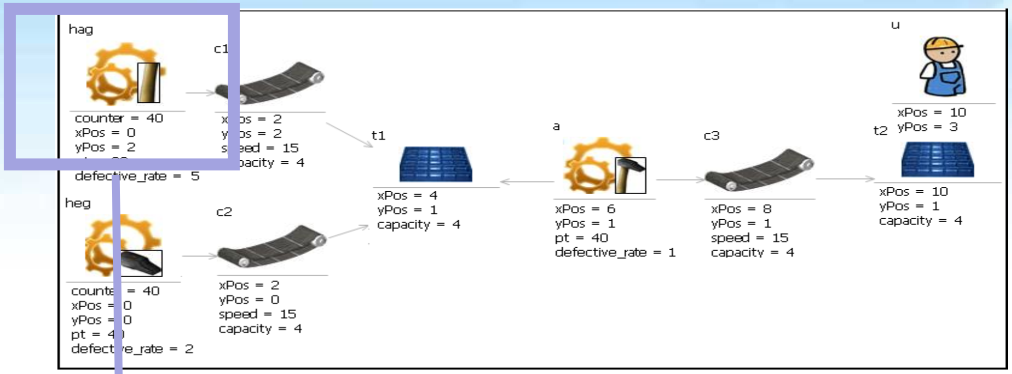
- Defined by a Semantic Mapping to Real-Time Maude





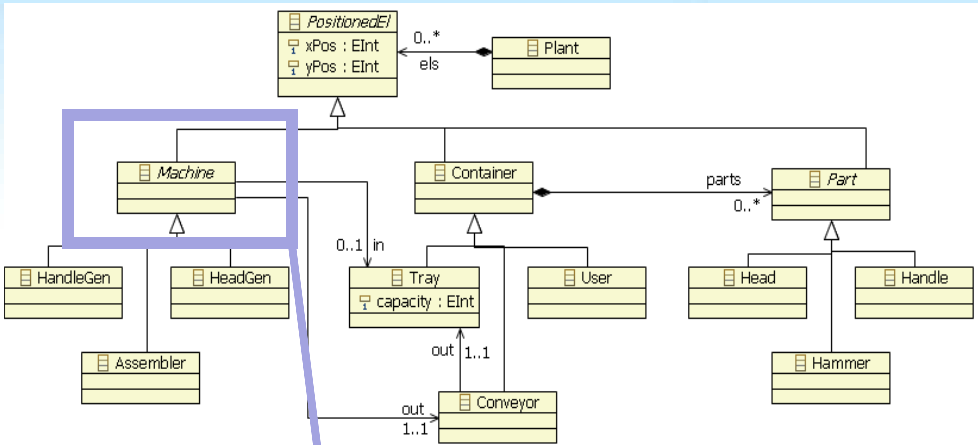
- This makes models amenable to simulation and to formal analysis using the Real-Time toolkit!
- **Implementable** (by a set of ATL model transformations)

Representing Models with Maude




```
ProductionSystem {
< 'p : Plant | els : 'heg 'hag 'c1 'c2 't1 'a 'c3 't2 'u >
< 'hag : HandleGen | in : null, out : 'c2, xPos : 1, yPos : 1 >
< 'heg : HeadGen | in : null, out : 'c1, xPos : 1, yPos : 3 >
< 'c1 : Conveyor | outParts : nil, out : 't1, xPos : 2, yPos : 3 >
< 'c2 : Conveyor | outParts : nil, out : 't1, xPos : 2, yPos : 1 >
< 't1 : Tray | parts : nil, capacity : 4, xPos : 3, yPos : 2 >
< 'a : Assembler | in : 't1, out : 'c3, xPos : 4, yPos : 2 >
< 'c3 : Conveyor | outParts : nil, out : 't2, xPos : 5, yPos : 2 >
< 't2 : Tray | parts : nil, capacity : 4, xPos : 6, yPos : 2 >
< 'u : User | parts : nil, xPos : 6, yPos : 3 >
}
```

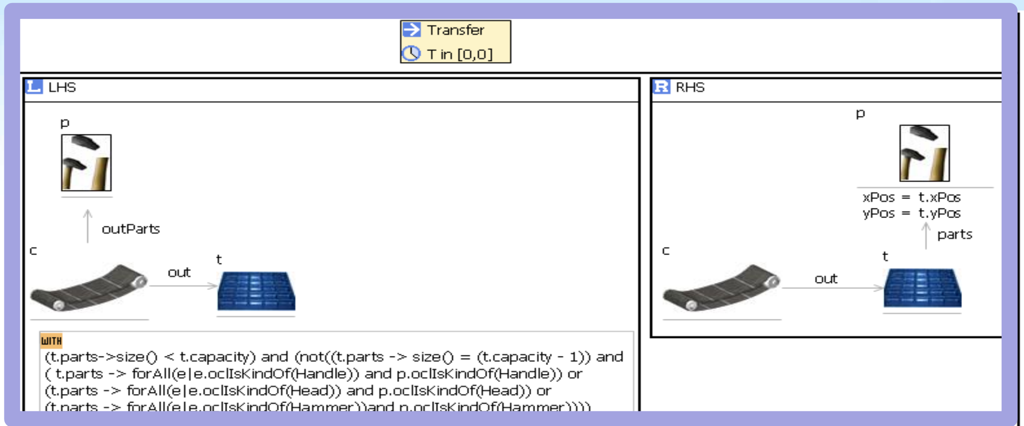
Representing Metamodels with Maude




```
eq isAbstract(Machine) = true .
...
eq type(in) = Tray .
eq lowerBound(in) = 0 .
eq upperBound(in) = 1 .
...
eq type(out) = Conveyor .
eq opposite(out) = null .
eq lowerBound(out) = 1 .
eq upperBound(out) = 1 .
```

```
op ProductionSystem : -> @Metamodel .

op PS : -> @Package .

sort PositionedEl .
subsort PositionedEl < @Class .
op PositionedEl : -> PositionedEl .
  op xPos : -> @Attribute .
  op yPos : -> @Attribue .

sort Container .
subsort Container < PositionedEl .
op Container : -> Container .
  op parts : -> @Reference .

sort Machine .
subsort Machine < PositionedEl .
op Machine : -> Machine .
  op in : -> @Reference .
  op out : -> @Reference .

...
```

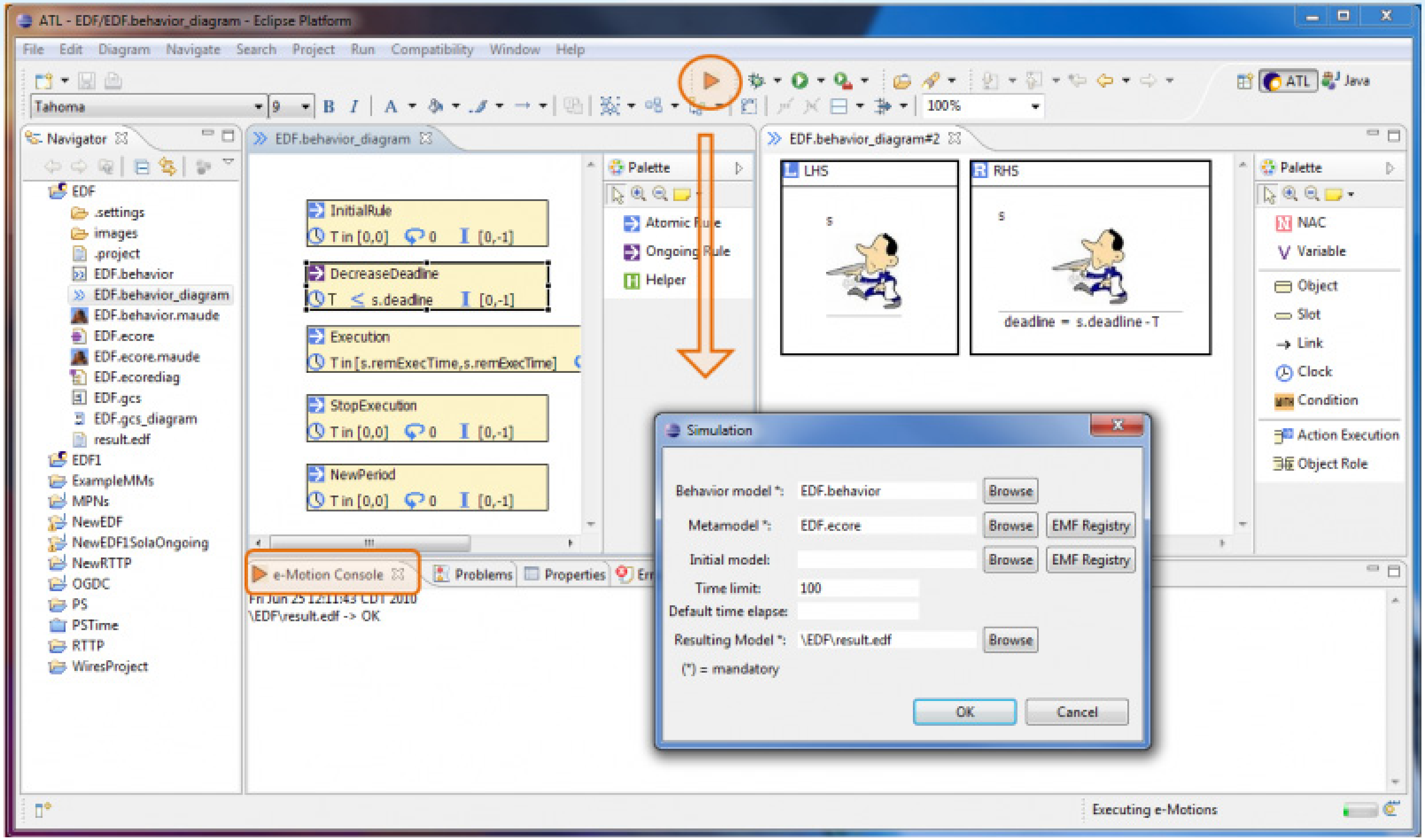
Representing Behavior with Maude




```
rl [Transfer] :
  ProductionSystem {
    < p : P:Part | xPos : XPOS, yPos : YPOS, SFS >
    < c : Conveyor | OutParts : (p PARTS), out : t, SFS' >
    < t : Tray | xPos : XPOS', yPos : YPOS', parts : PARTS', SFS" >
    OBJSET }
=>
  ProductionSystem{
    < p : P:Part | xPos : XPOS', yPos : YPOS',SFS >
    < c : Conveyor | outParts : PARTS, out : t, SFS' >
    < t : Tray | xPos : XPOS', yPos : YPOS', parts : (p PARTS'), SFS" >
    OBJSET }
```

http://atenea.lcc.uma.es/e-Motions

e-Motions



Model Simulation and Analysis

[Simulation 2009]

- Simulation/Execution of specifications

```
(trew initModel in time <= 20 .)
```

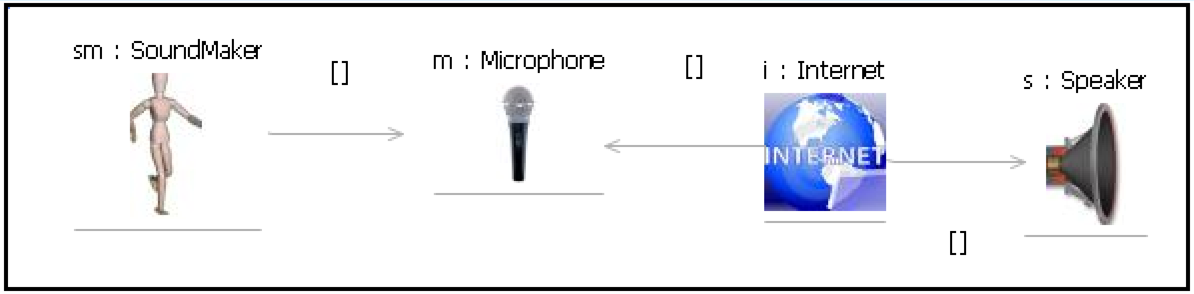
- Reachability Analysis
  - Deadlock
  - Invariants
  - Others

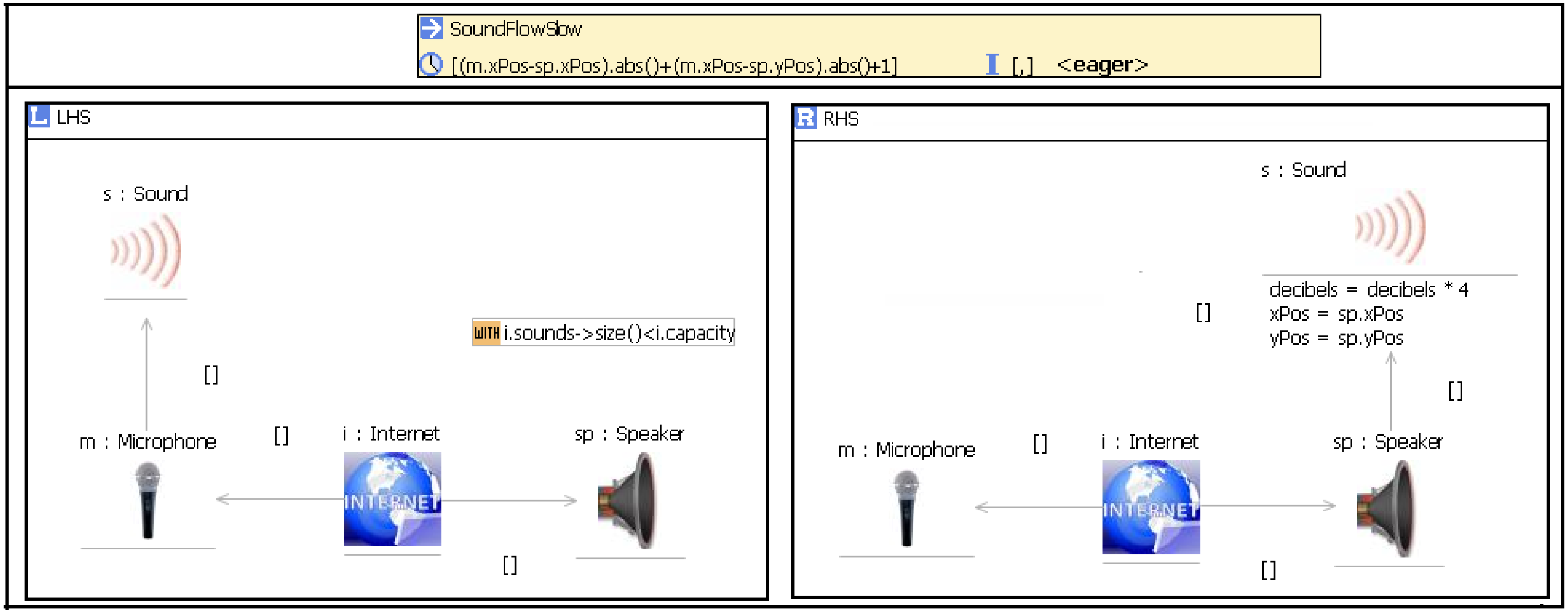
```
search initModel =>*
  ProductionSystem {
    < O : Tray | capacity : CAP, parts : PARTS, SFS >
    OBJSET }
(find earliest {initModel} =>* {ProductionSystem {
< T : ActionExec | rule : "Collect", value : null,
SFS@T > OBJSET }} .)
```

- LTL Model checking
  - Liveness properties

```
(mc {initModel} |=t
  [](ensembled('he10.ha10) -> collected('he10.ha10))
in time <= 100 .)
```

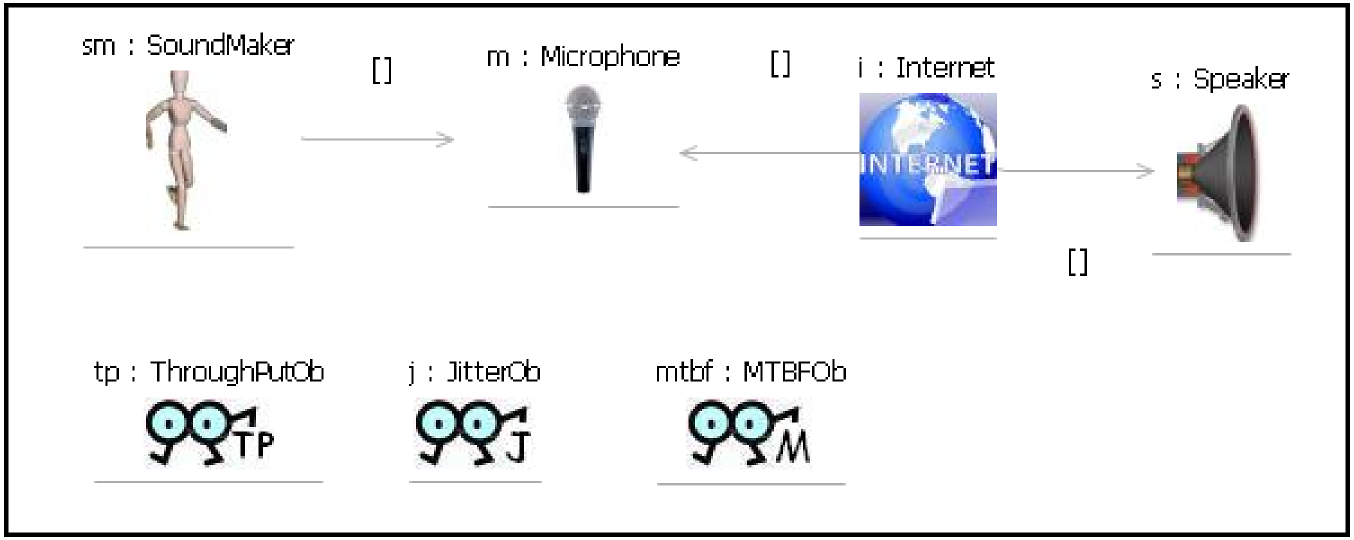
Adding NFPs to DSMLs





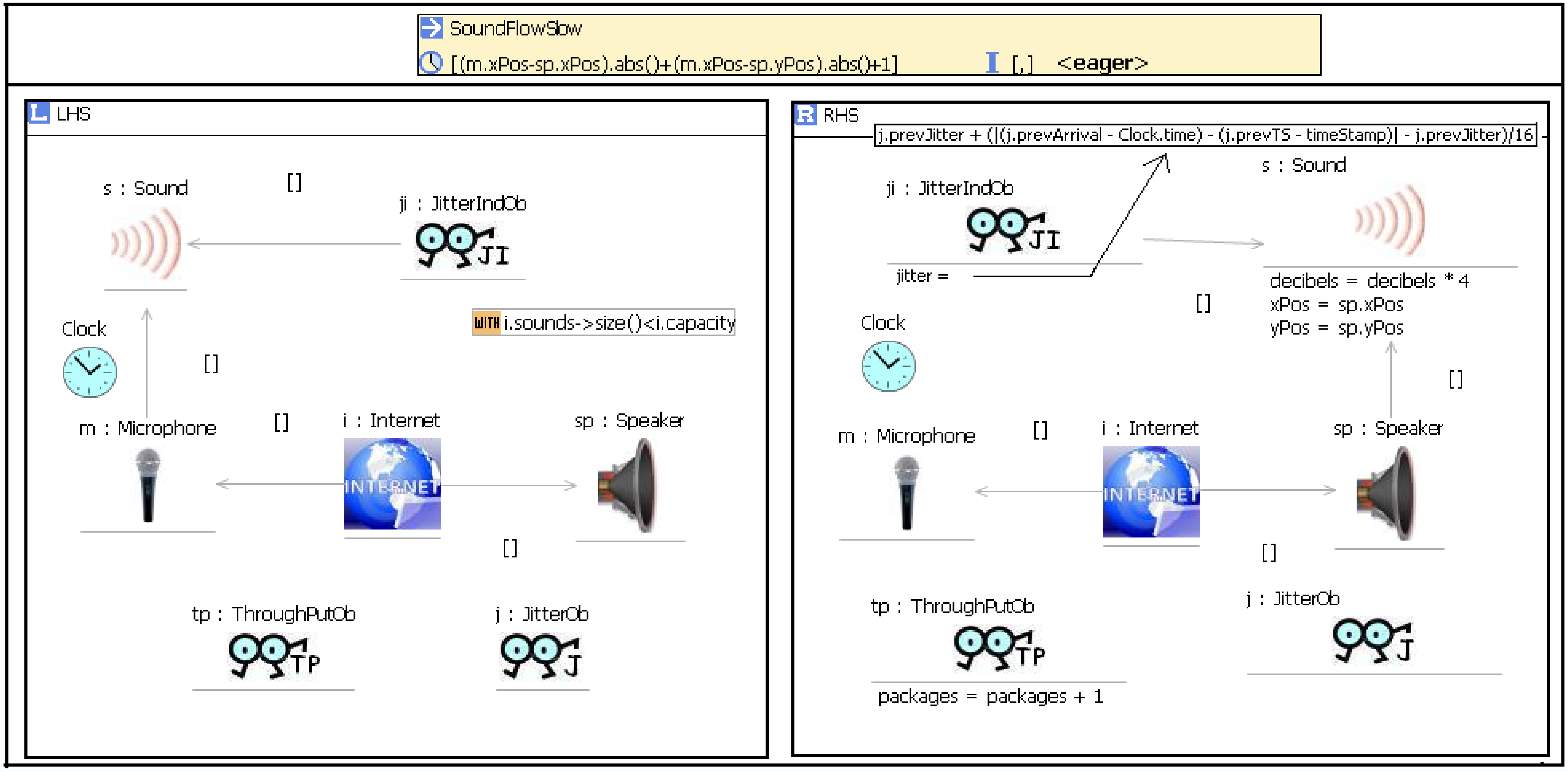
Adding NFPs to DSMLs

- Use of “observers”



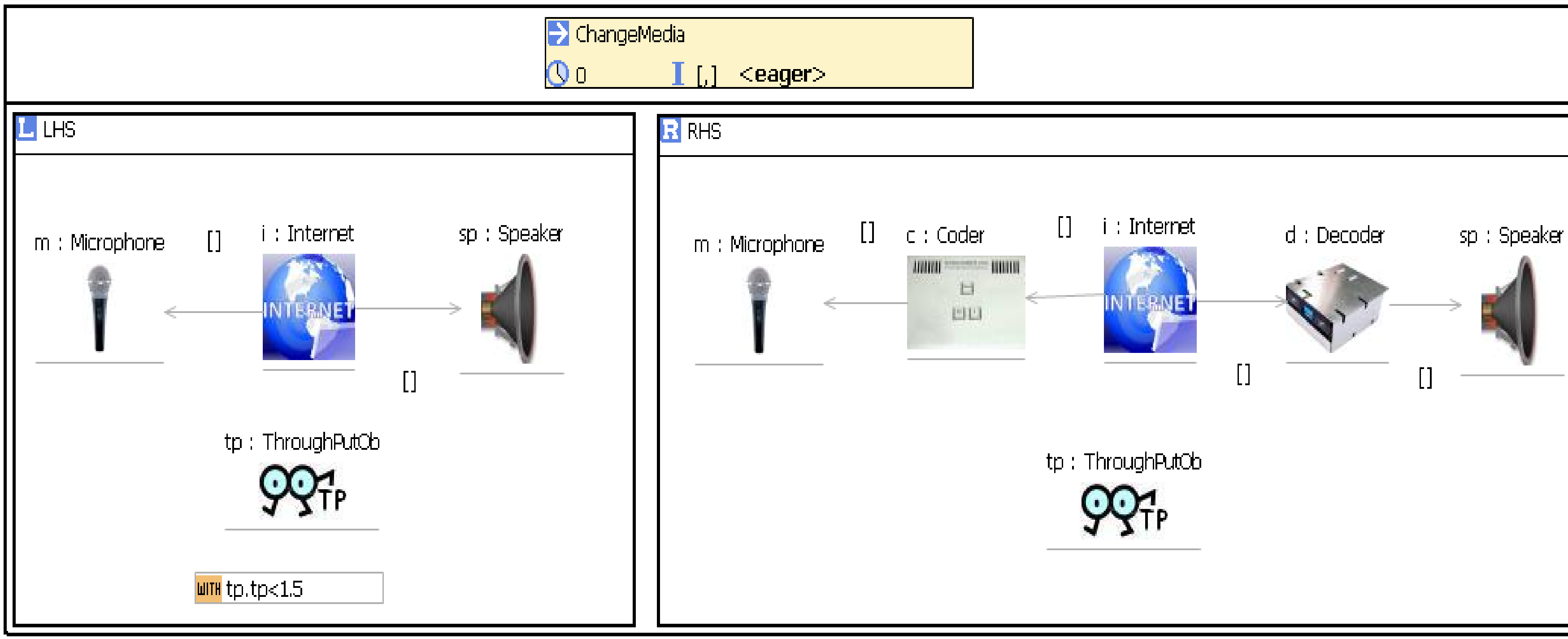
Adding NFPs to DSMLs

- Observers capture the state of the NFPs and monitor their progress



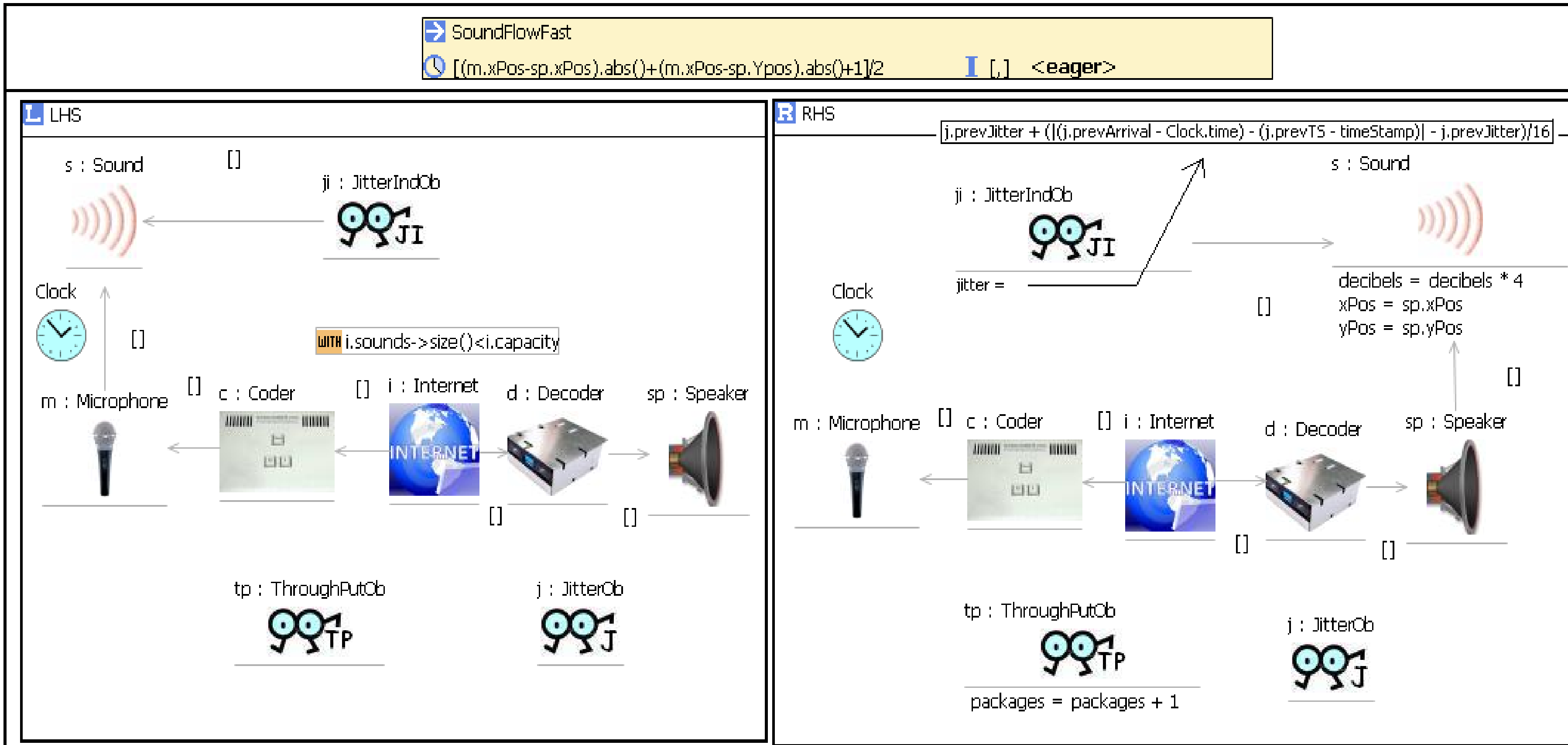
Making use of the Observers

- The system can self-adapt under certain conditions

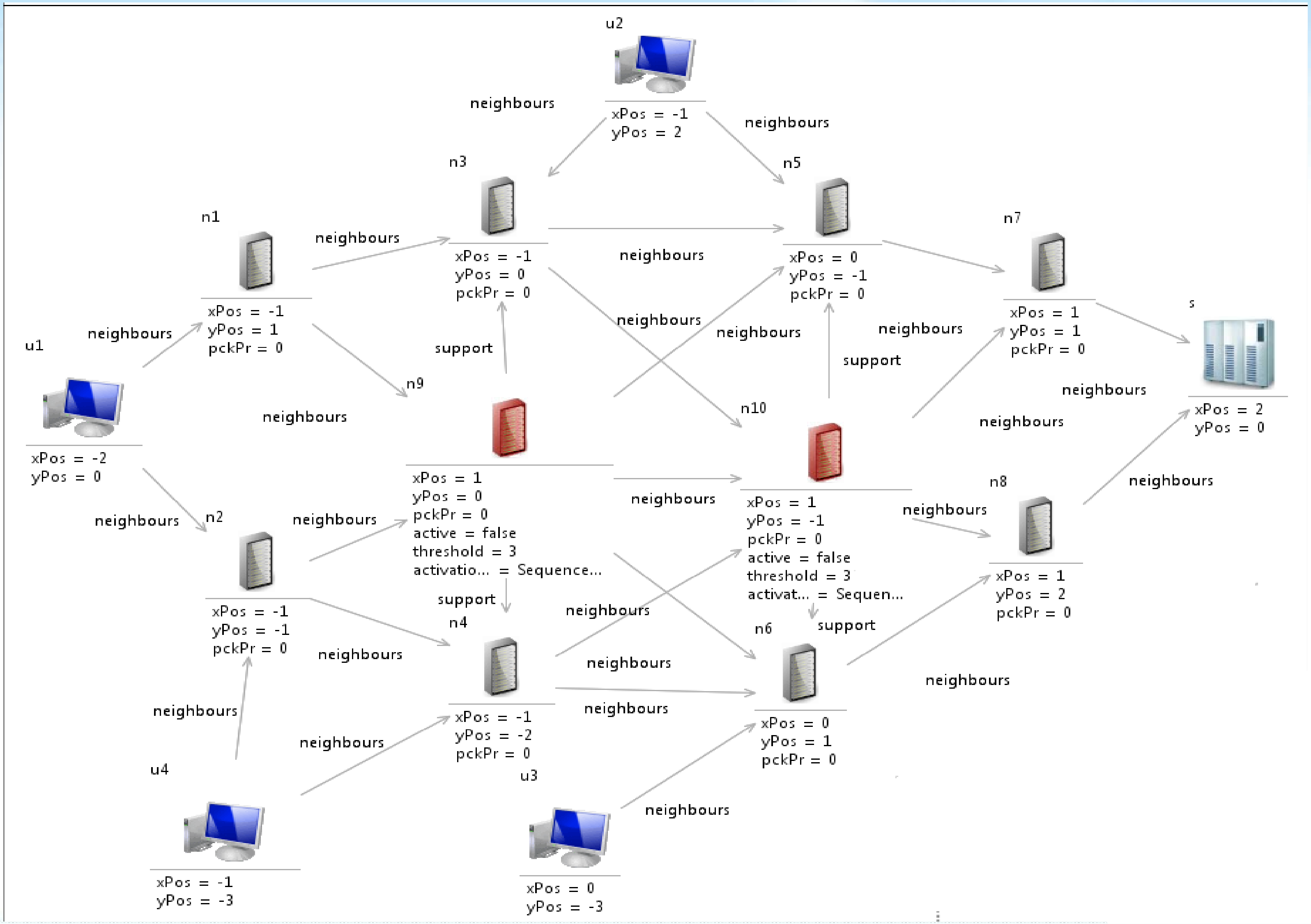


Making use of the Observers

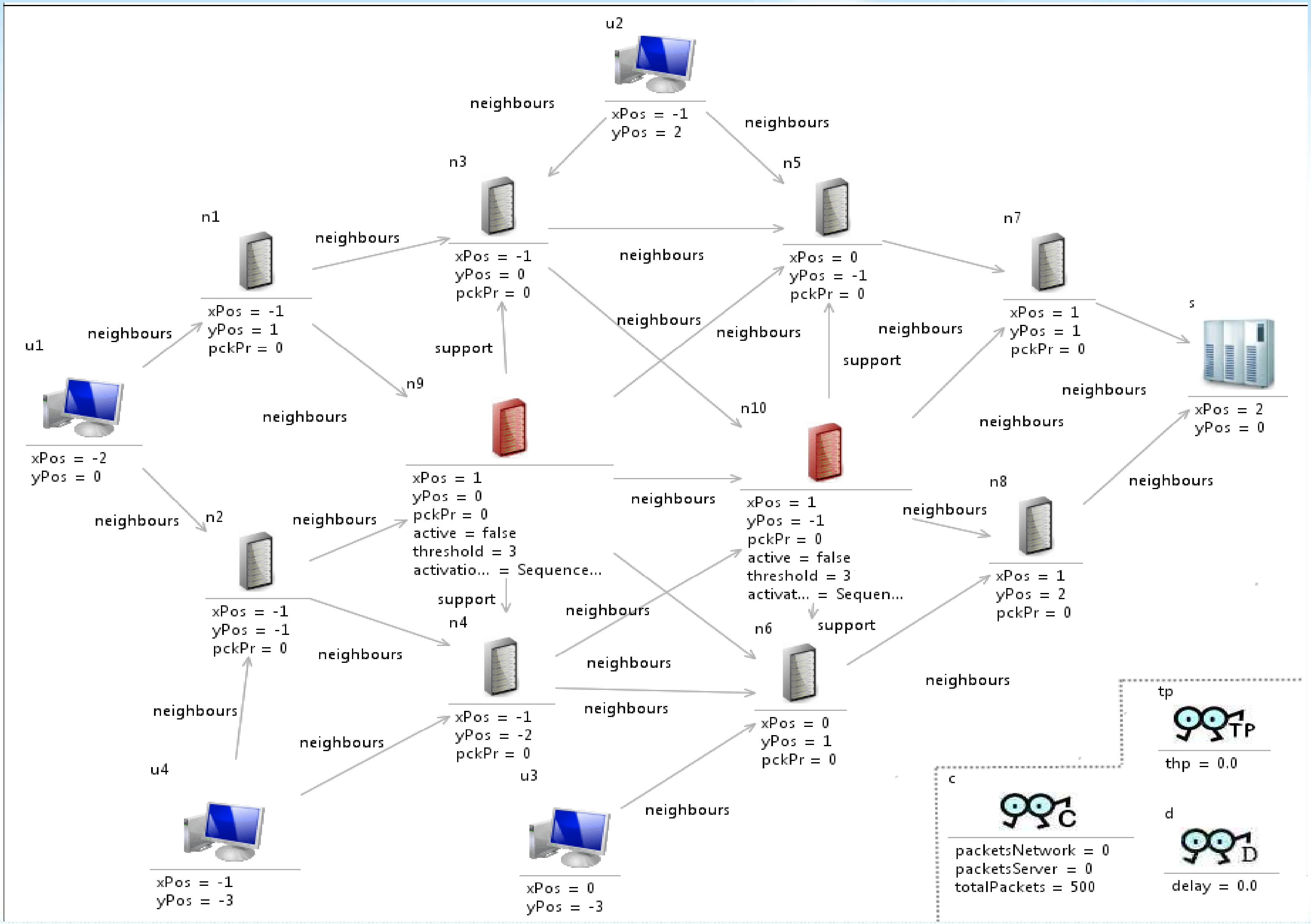
- With the new configuration, the system transmits sounds in a faster way

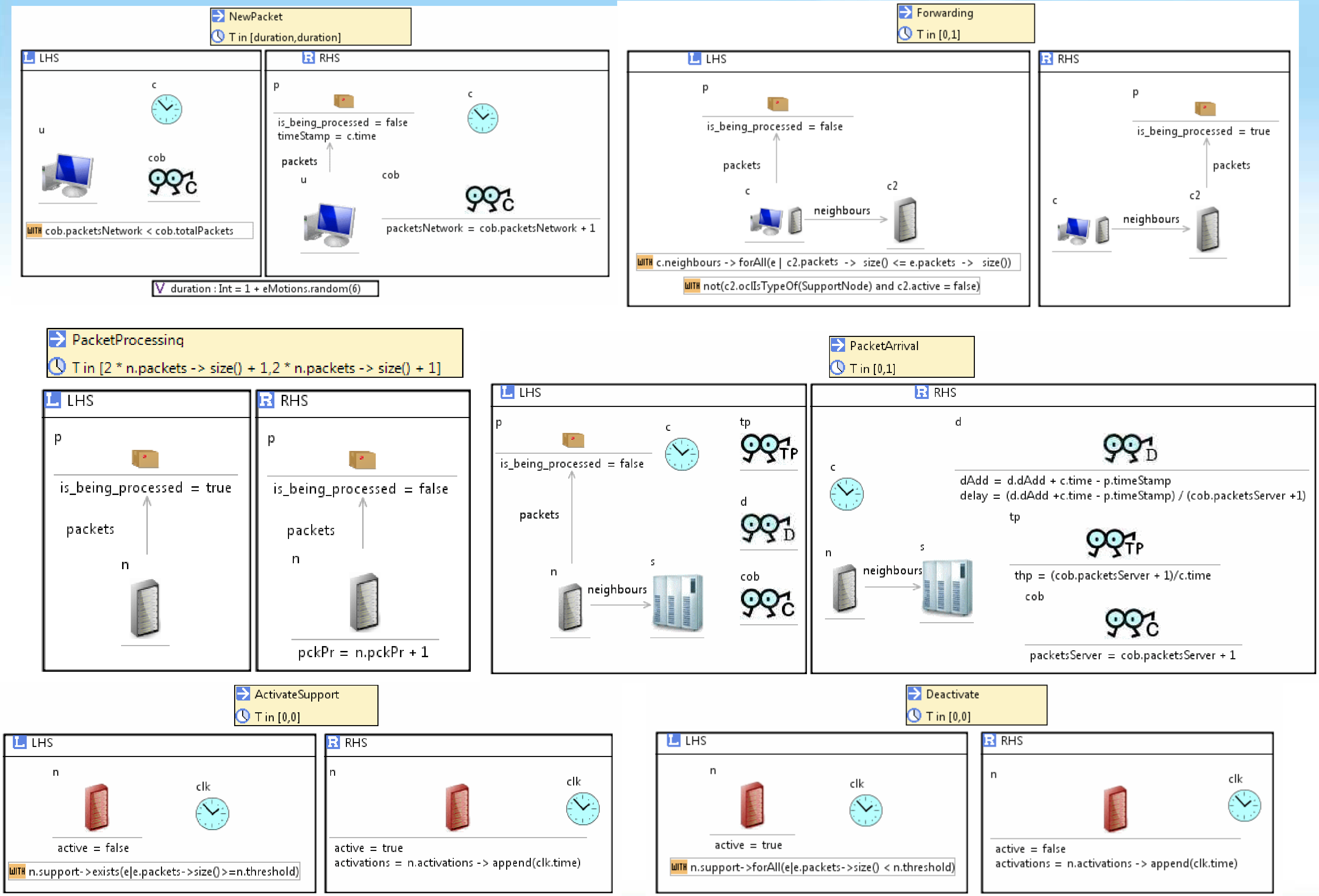


A more complex example



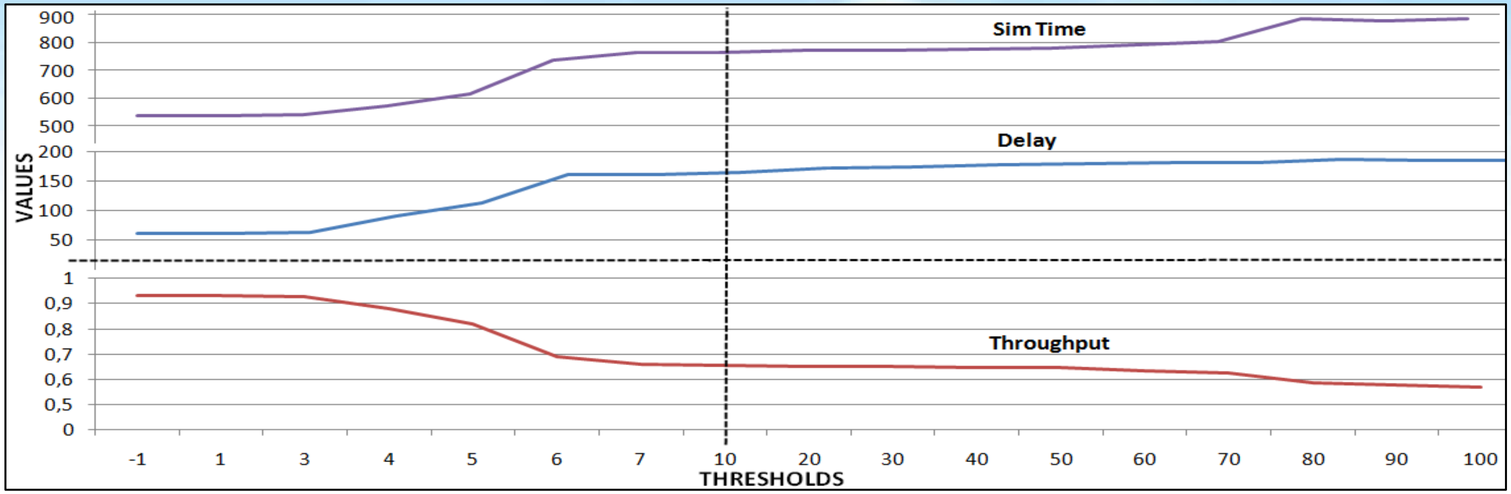
A more complex example

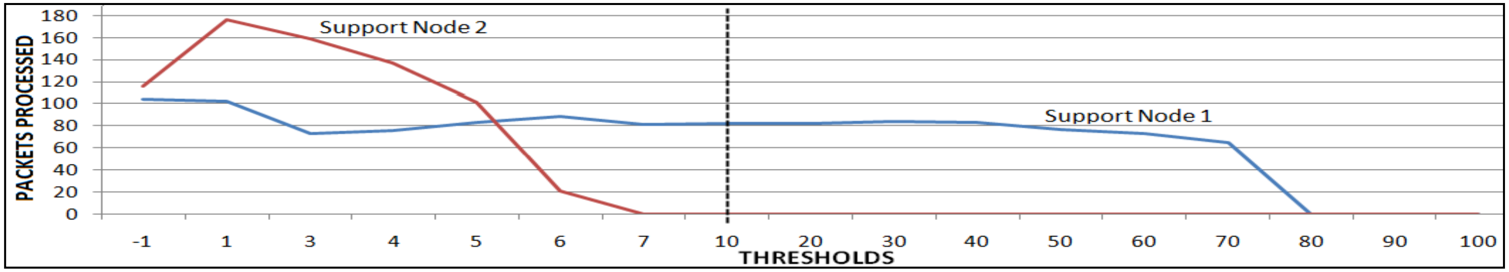


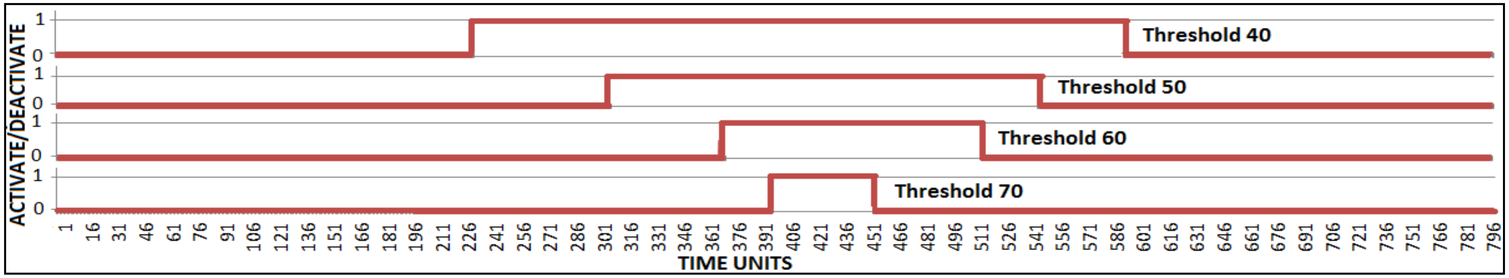


NewPacket
T in [duration,duration]
LHS
c
u
cob
cob.packetsNetwork < cob.totalPackets
RHS
p
is_being_processed = false
timeStamp = c.time
packets
u
c
cob
packetsNetwork = cob.packetsNetwork + 1
duration : Int = 1 + eMotions.random(6)
Forwarding
T in [0,1]
LHS
p
is_being_processed = false
packets
c
c2
neighbours
c.neighbours -> forAll(e | c2.packets -> size() <= e.packets -> size())
not(c2.oclIsTypeOf(SupportNode) and c2.active = false)
RHS
p
is_being_processed = true
packets
c
c2
neighbours
PacketProcessing
T in [2 * n.packets -> size() + 1,2 * n.packets -> size() + 1]
LHS
p
is_being_processed = true
packets
n
RHS
p
is_being_processed = false
packets
n
pckPr = n.pckPr + 1
PacketArrival
T in [0,1]
LHS
p
is_being_processed = false
c
tp
d
cob
packets
n
neighbours
s
RHS
d
dAdd = d.dAdd + c.time - p.timeStamp
delay = (d.dAdd +c.time - p.timeStamp) / (cob.packetsServer +1)
c
tp
thp = (cob.packetsServer + 1)/c.time
n
neighbours
s
cob
packetsServer = cob.packetsServer + 1
ActivateSupport
T in [0,0]
LHS
n
clk
active = false
n.support->exists(e|e.packets->size()>=n.threshold)
RHS
n
clk
active = true
activations = n.activations -> append(clk.time)
Deactivate
T in [0,0]
LHS
n
clk
active = true
n.support->forAll(e|e.packets->size() < n.threshold)
RHS
n
clk
active = false
activations = n.activations -> append(clk.time)

Analysing the results







Pros and Cons

- **Advantages**
  - Addition of observers independently of the system
  - Simple modelling of QoS properties
  - Ability to monitor QoS properties
  - Results obtained in easy-to-manipulate format
  - More expressive than other notations (SPA, SPN, QNM,…)
    (generalized distributions, OCL expressiveness, dynamic topologies, action executions as 1st class citizens,…)
- **Limitations**
  - Efficiency of simulations
  - Difficult to debug
  - Not for every problem or domain
  - More expressive than other notations (SPA, SPN, QNM,…)
    (Difficulties for defining semantic bridges due to large gaps/chasms)

We are not alone…

LIGHTWEIGHT MODELING

DEFINITION

- constructing a very abstract model of the core concepts of a system
- using an analysis tool based on exhaustive enumeration to explore its properties

WHY IS IT "LIGHTWEIGHT"?

- because the model is very abstract in comparison to a real implementation, and focuses only on core concepts, it is small and can be constructed quickly
- because the analysis tool is "push-button", it yields results with little effort

in contrast,
theorem proving is not "push-button"

WHAT IS ITS VALUE?

- it is a design tool that reveals conceptual errors early

decades of research on software engineering proves that the cost of fixing a bug rises exponentially with the delay in its discovery

- it is a documentation tool that provides complete, consistent, and unambiguous information to implementors and users
- it is easy (at least to get started) and surprising (you get the result of scenarios you would *never* expect)

"If you like surprises, you will love lightweight modeling."
—Pamela Zave

- EASY + SURPRISING = FUN

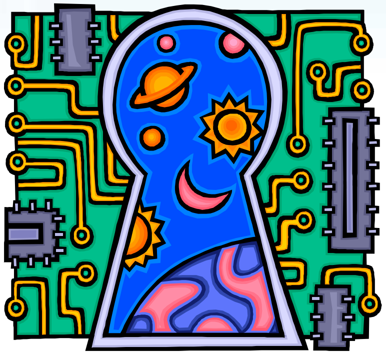
[Pamela Zave, keynote talk at MODELS 2010]

Recap

- One challenge for software engineers now is to provide **end-users** with **Modeling Languages** (and associated tools) that allow them to model their systems in a cheap, quick and useful way, and to analyse them using *push-button* approaches.
- Current widely-used general-purpose modeling notations (especially behavioural and QoS) do not seem to be really up to the job
- *Integrating heterogeneous notations and their associated tools using model transformations seems to be one promising way to go*

Challenges

- Definition of new languages for behavioural descriptions, which allow easy specification of Quality Properties and their analysis



- Improved languages for QoS specification
- Semantic bridges to other domains
  - Better connection with analysis tools
- Improved traceability mechanisms
  - Improve understandability of results
- Better feedback to users
  - E.g., Performance anti-patterns (!)

Thanks!

Acknowledgements:

Atenea
Modeling Software Systems

GOBIERNO DE ESPAÑA

MINISTERIO DE CIENCIA E INNOVACIÓN