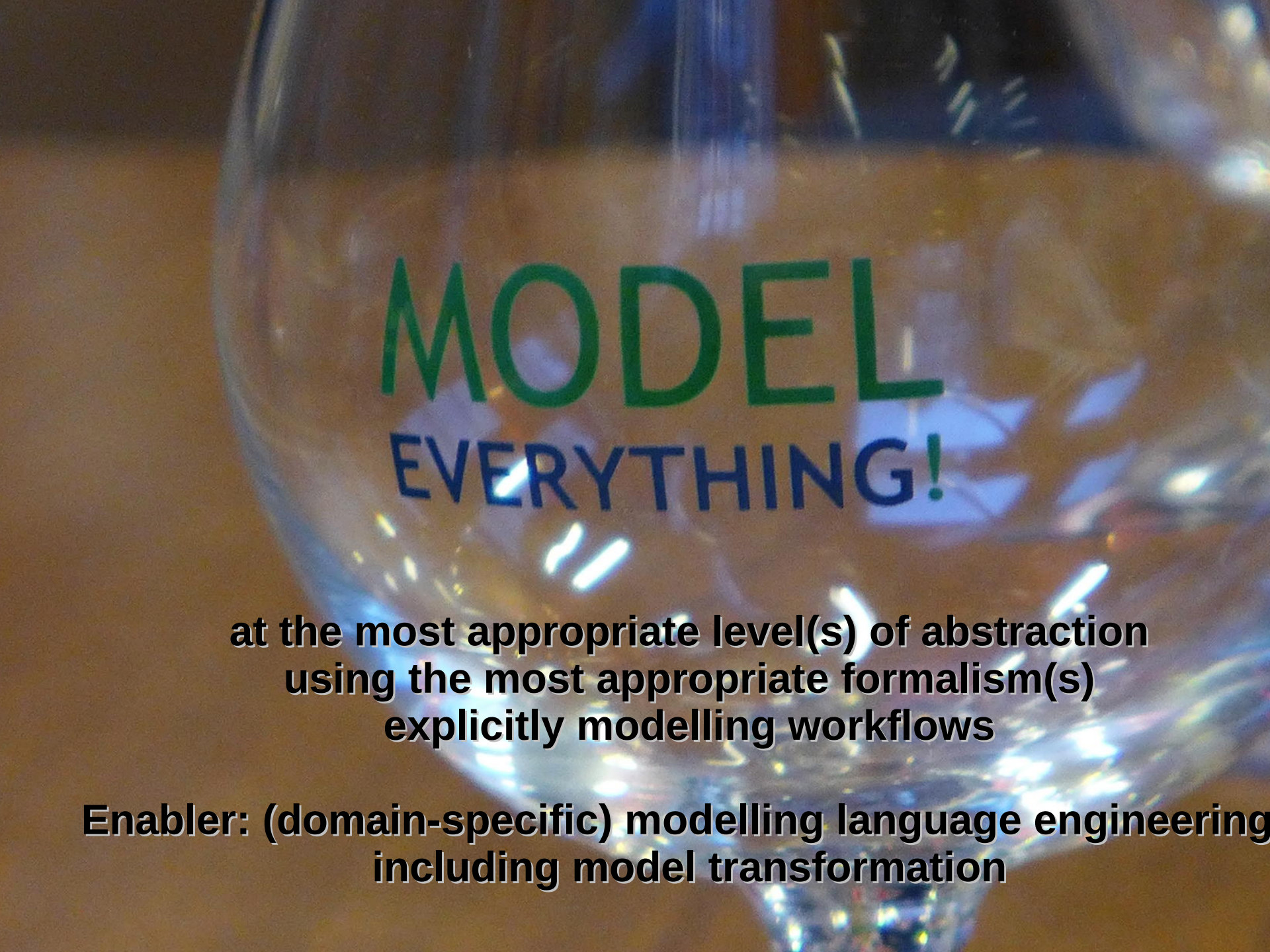
(Domain-Specific) Modelling Language Engineering

Hans Vangheluwe

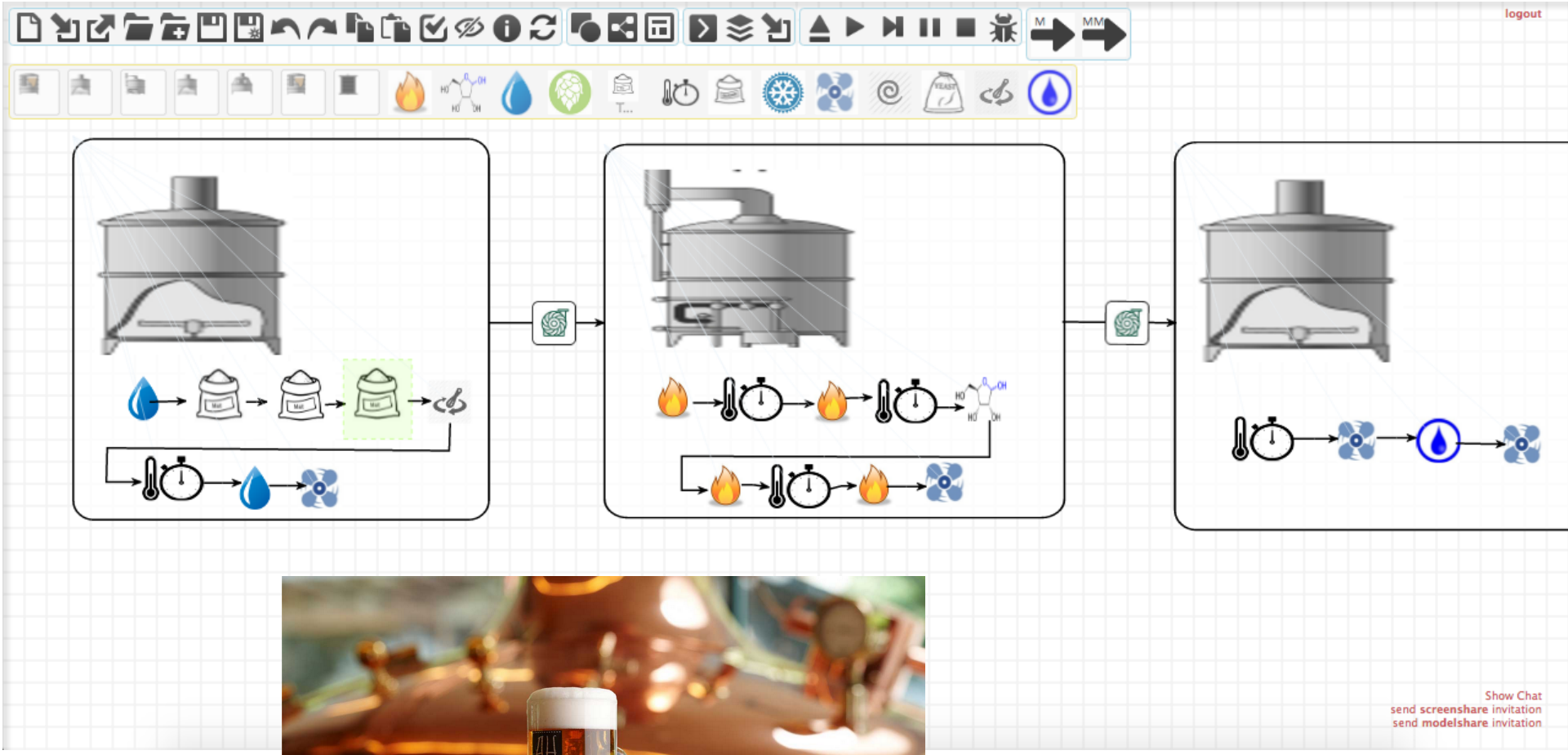
http://msdl.cs.mcgill.ca/

Universiteit Antwerpen
Ansymo
Antwerp Systems & Software Modelling
University of Antwerp
Nexor
Cyber-Physical Systems
University of Antwerp
FLANDERS MAKE
MANUFACTURING INNOVATION NETWORK
McGill

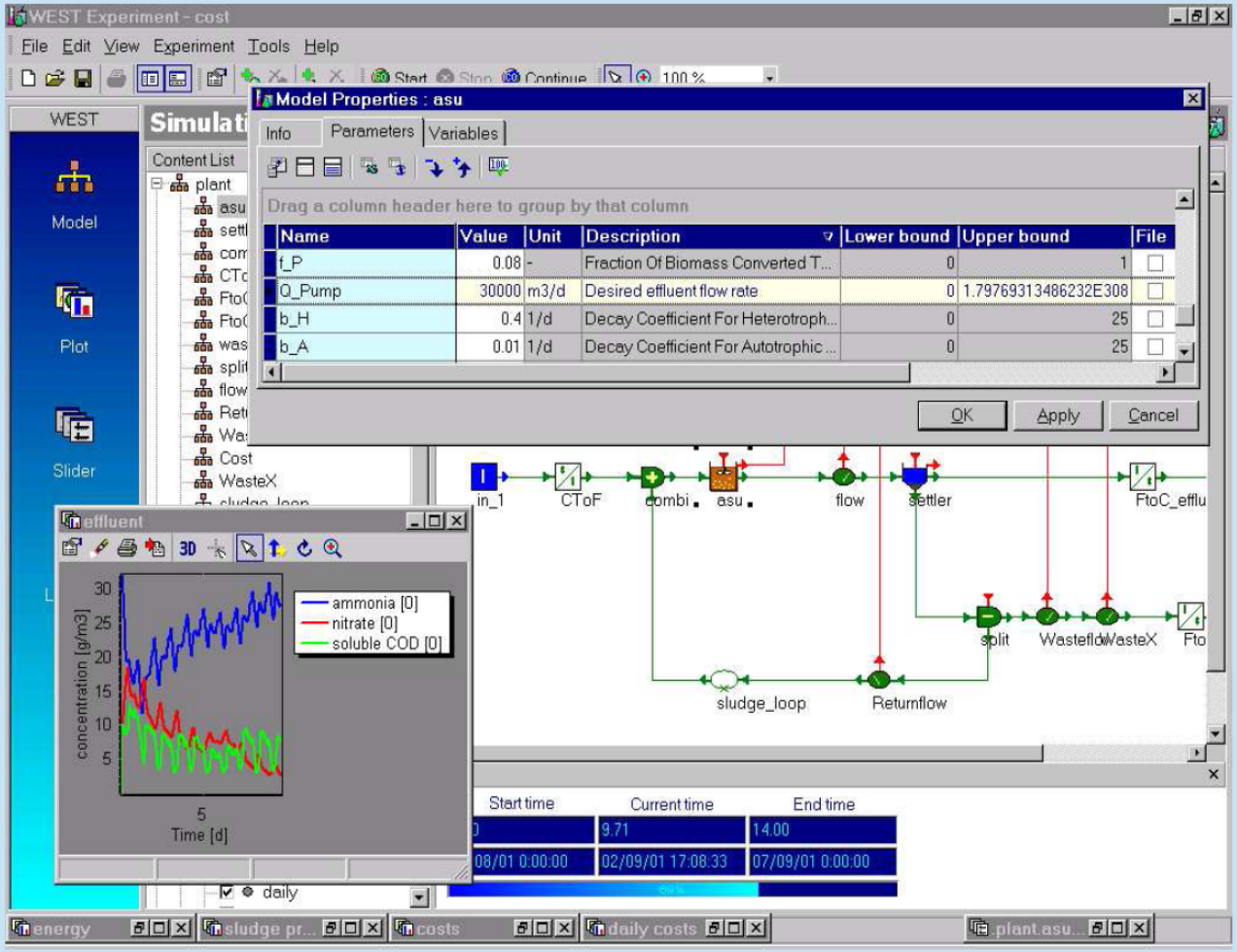


at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling workflows

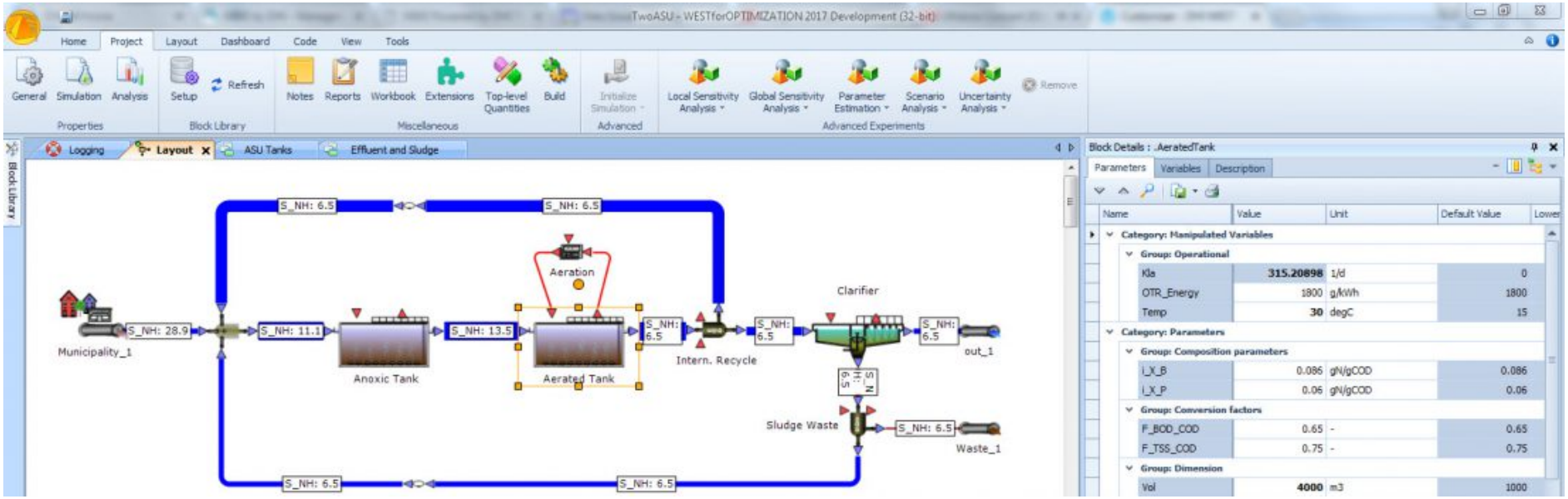
Enabler: (domain-specific) modelling language engineering
including model transformation


logout
Show Chat
send screenshare invitation
send modelshare invitation

DS(V)M Environment



WEST: modelling biological wastewater treatment.
Henk Vanhooren, Jurgen Meirlaen, Youri Amerlinck, Filip Claeys, Hans Vangheluwe and Peter A.Vanrolleghem.
Journal of Hydroinformatics 5 (2003) 27-50



http://www.mikebydhi.com/products/west

Why DS(V)M ?
(as opposed to General Purpose modelling)

- **match the user's mental model** of the problem domain
- **maximally constrain** the user (to the problem at hand)
  $\Rightarrow$ easier to learn
  $\Rightarrow$ avoid errors
- **separate** domain-expert's work
  from analysis/transformation expert's work

Anecdotal evidence of 5 to 10 times speedup

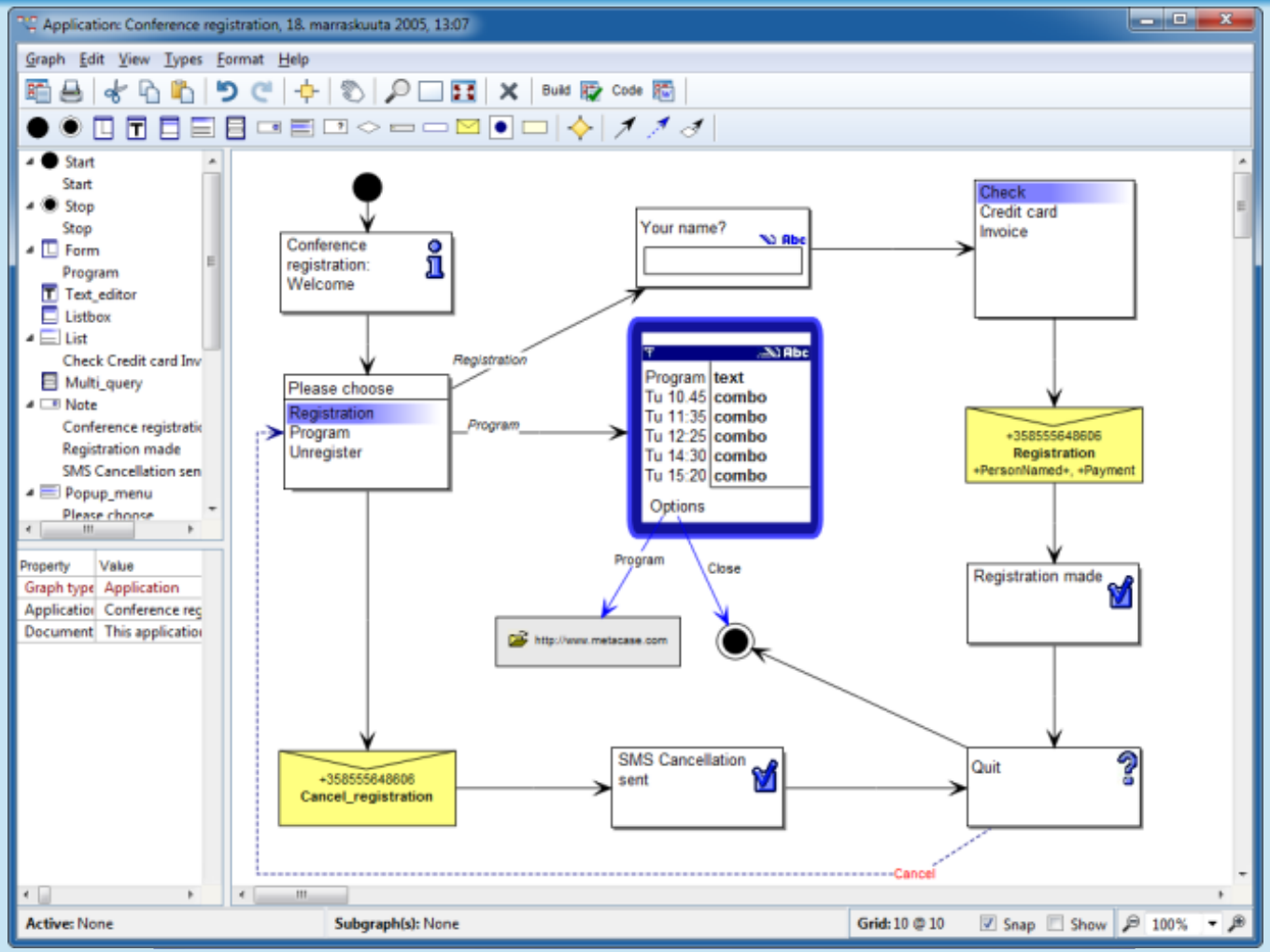
Steven Kelly and Juha-Pekka Tolvanen. Domain-Specific Modeling: Enabling Full Code Generation. Wiley, 2008.

Laurent Safa. The practice of deploying DSM, report from a Japanese appliance maker trenches. In Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM'06), pp. 185-196, 2006.

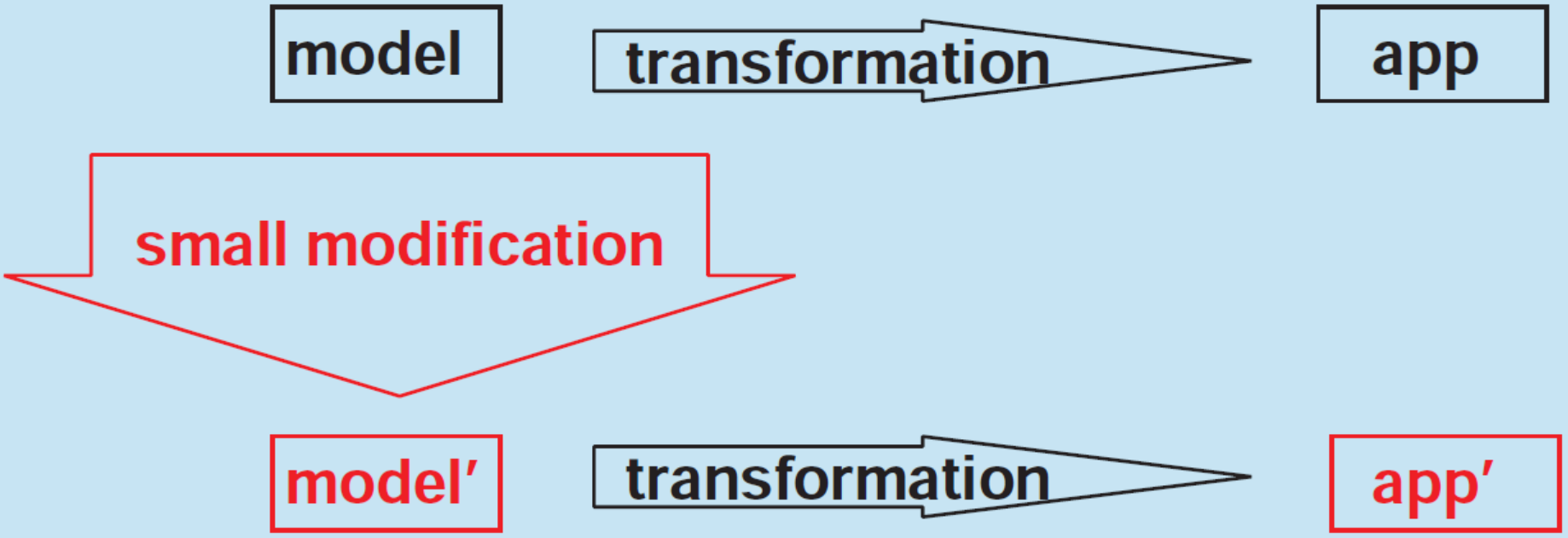
DS(V)M Example in Software Domain
smart phones, the application

MetaEdit+ (www.metacase.com)

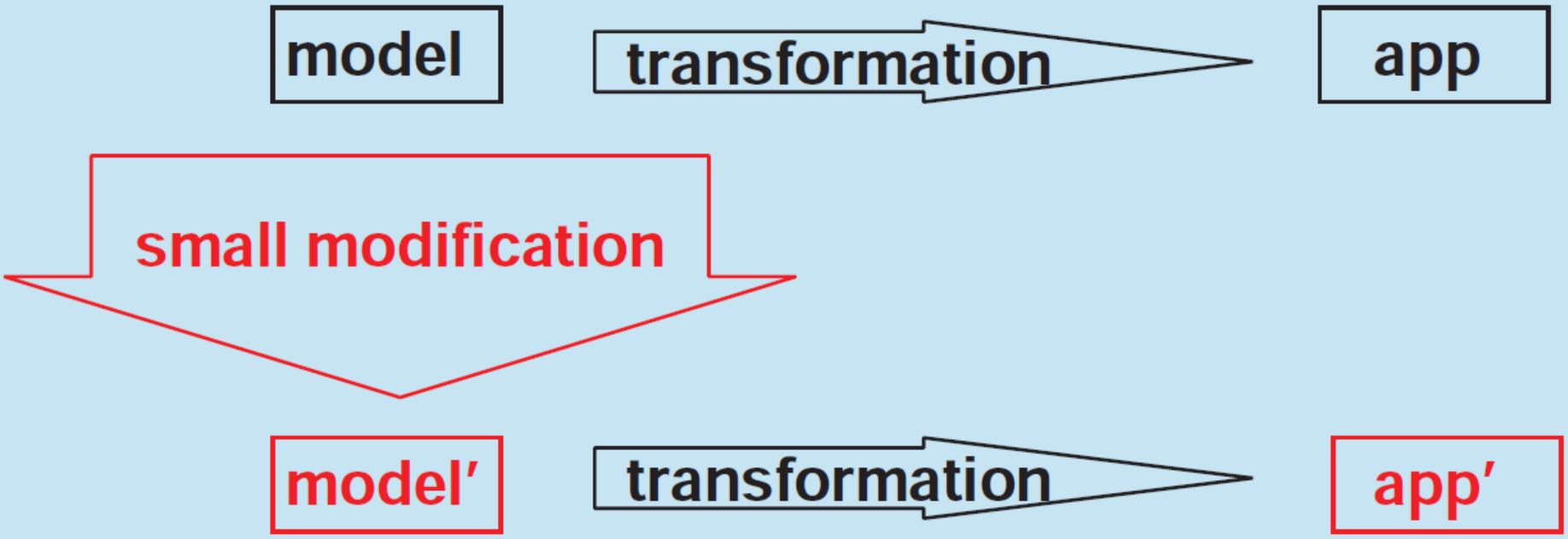
DS(V)M Example: smart phones, the Domain-Specific model



Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

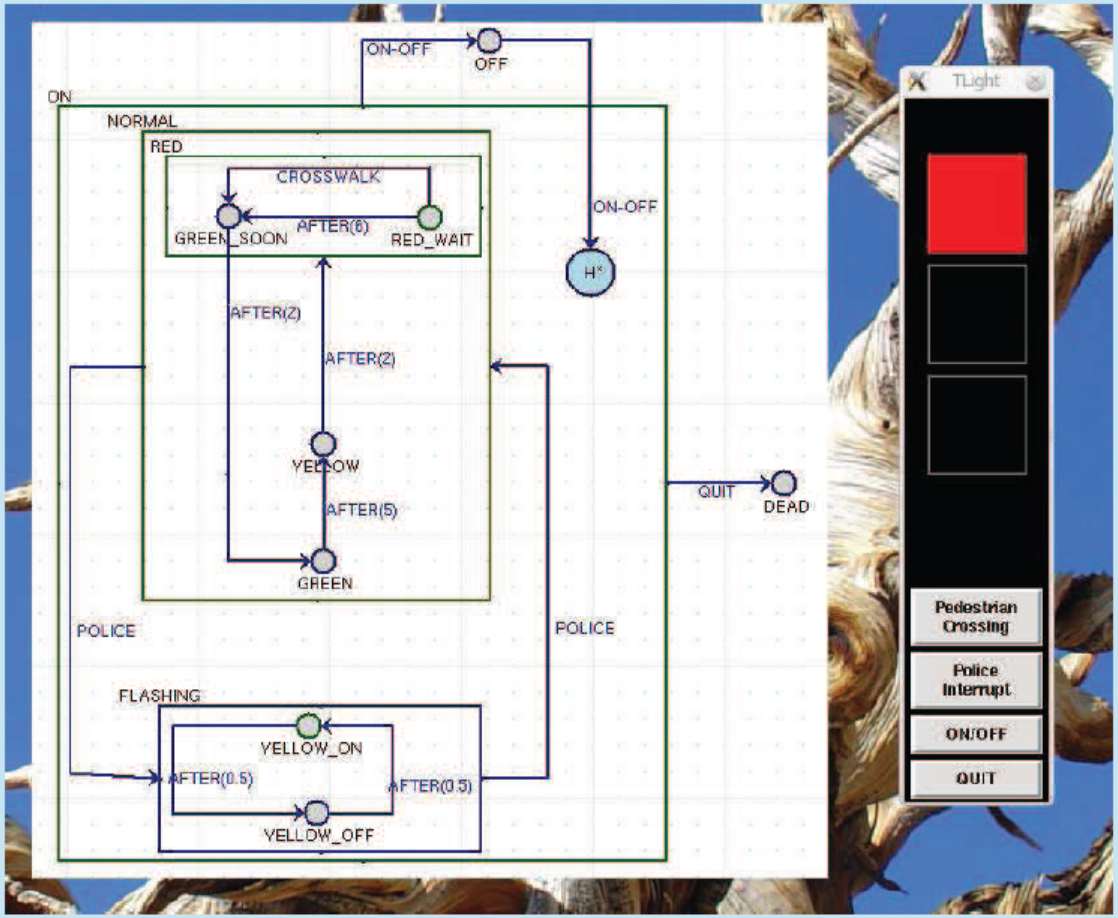


Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

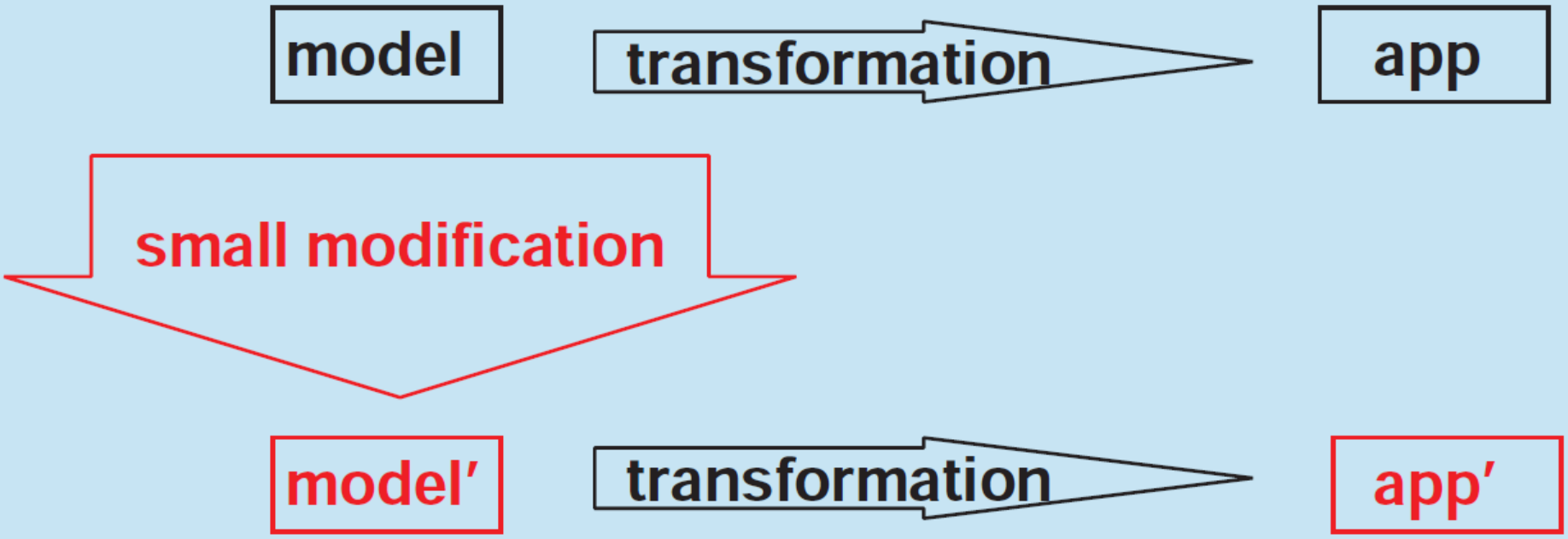


small **modification** in model may lead to large change in **app**
$\sim$ choice of formalism (e.g., Statecharts)

Statecharts



Model-Based Development:
Modify the Model
(e.g., based on feature model of product family)

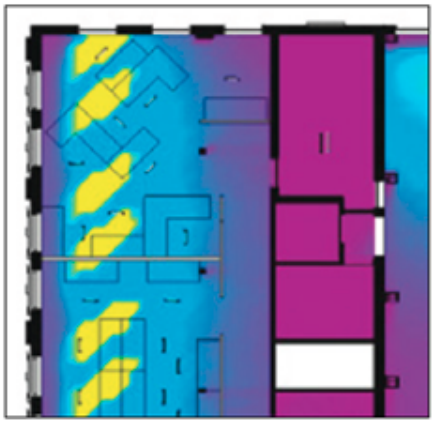


small **modification** in model may lead to large change in **app**
$\sim$ choice of formalism (e.g., Statecharts)


AUTODESK.
AUTODESK.

Nagy, Danil, Damon Lau, John Locke, Jim Stoddart, Lorenzo Villaggi, Ray Wang, Dale Zhao and David Benjamin.
“Project Discover : An Application of Generative Design for Architectural Space Planning.” (2017).

“fitness”

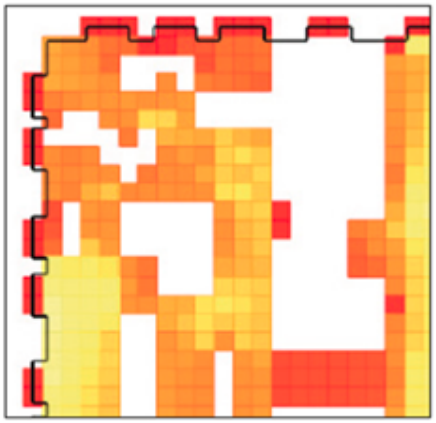


1. Daylight

2. Low Visual Distraction

3. Views to Outside

4. Adjacency Preference



5. Circulation

6. Work Styles

7. Low Acoustic Distraction

8. Low Density

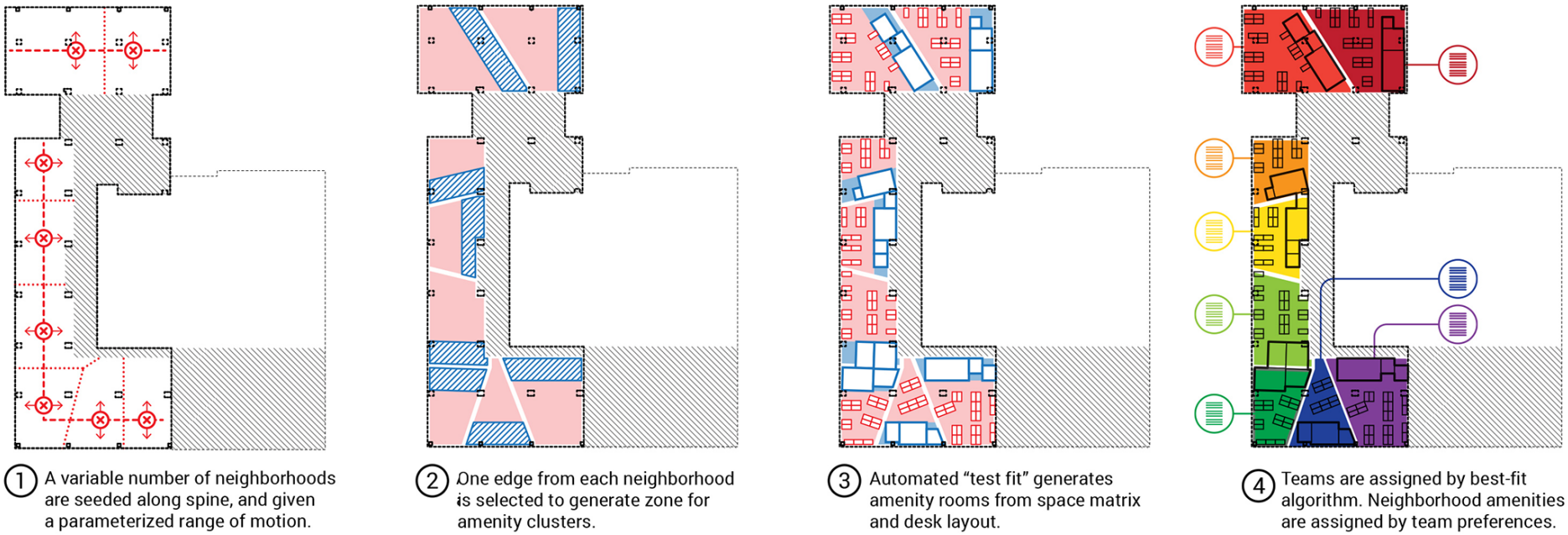
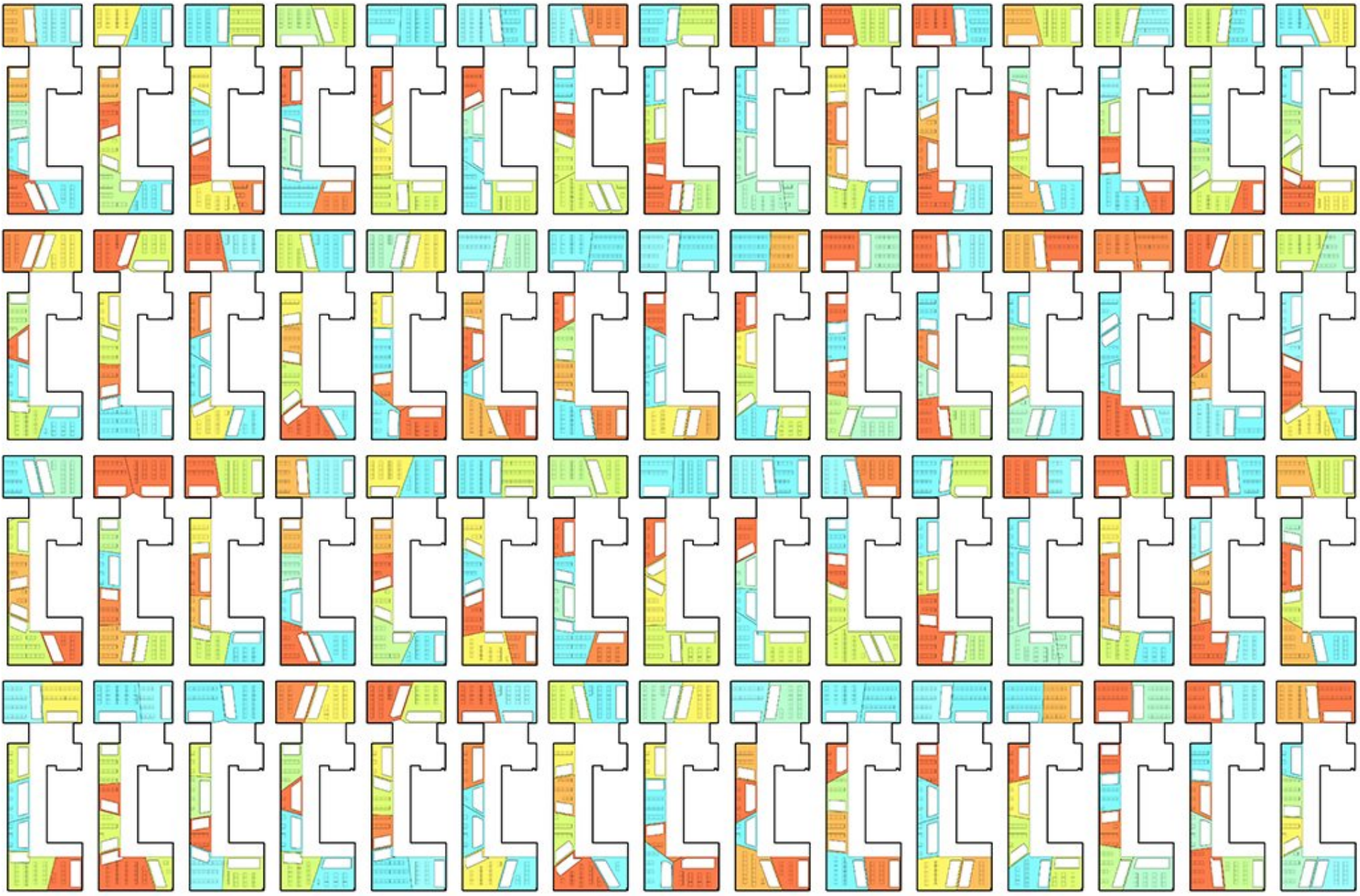


Figure 1. Description of specification of geometric model

generated (evolutionary – e.g., Genetic Algorithms)



“fitness” evaluation
for each of the generated designs

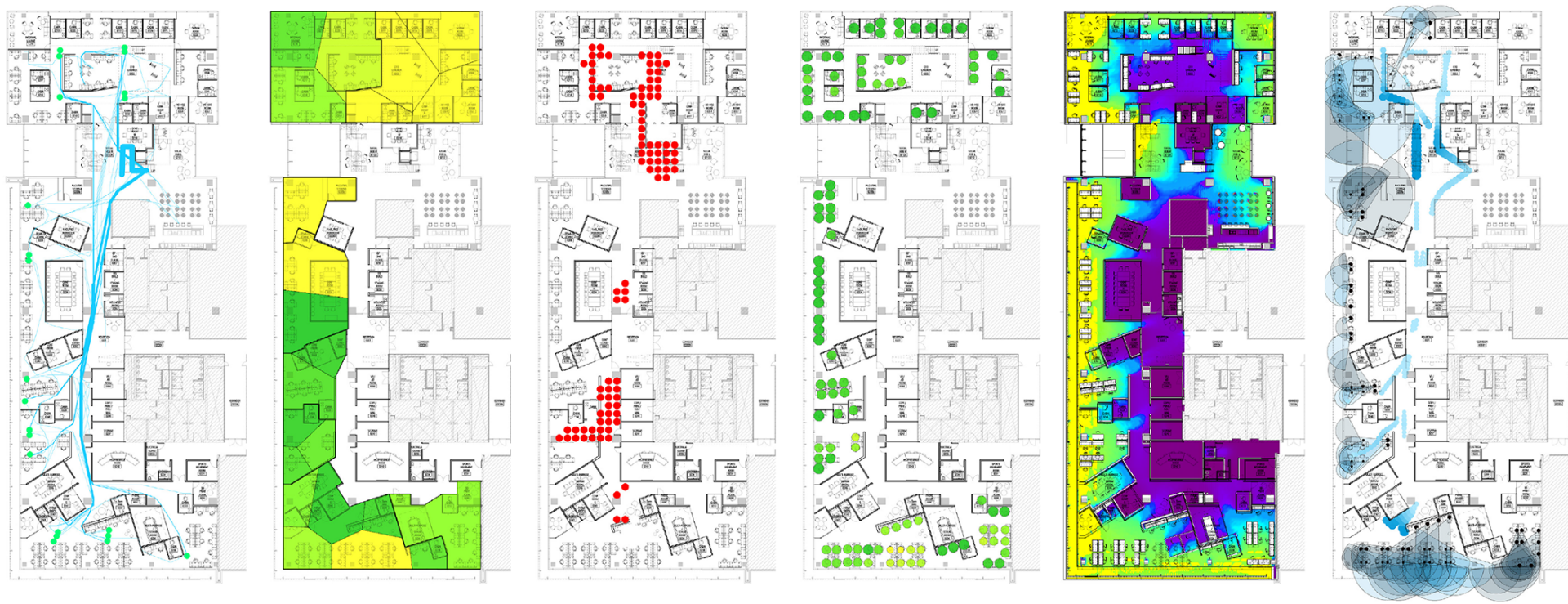
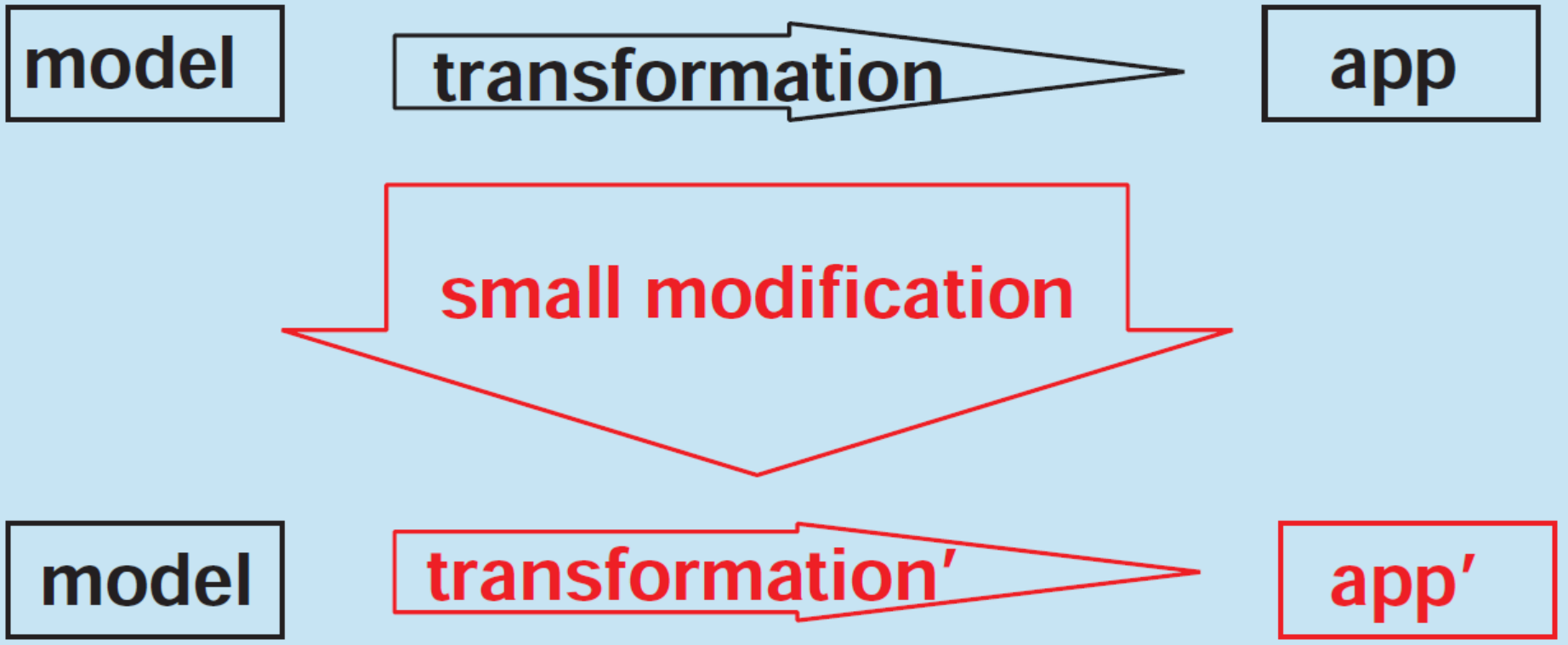
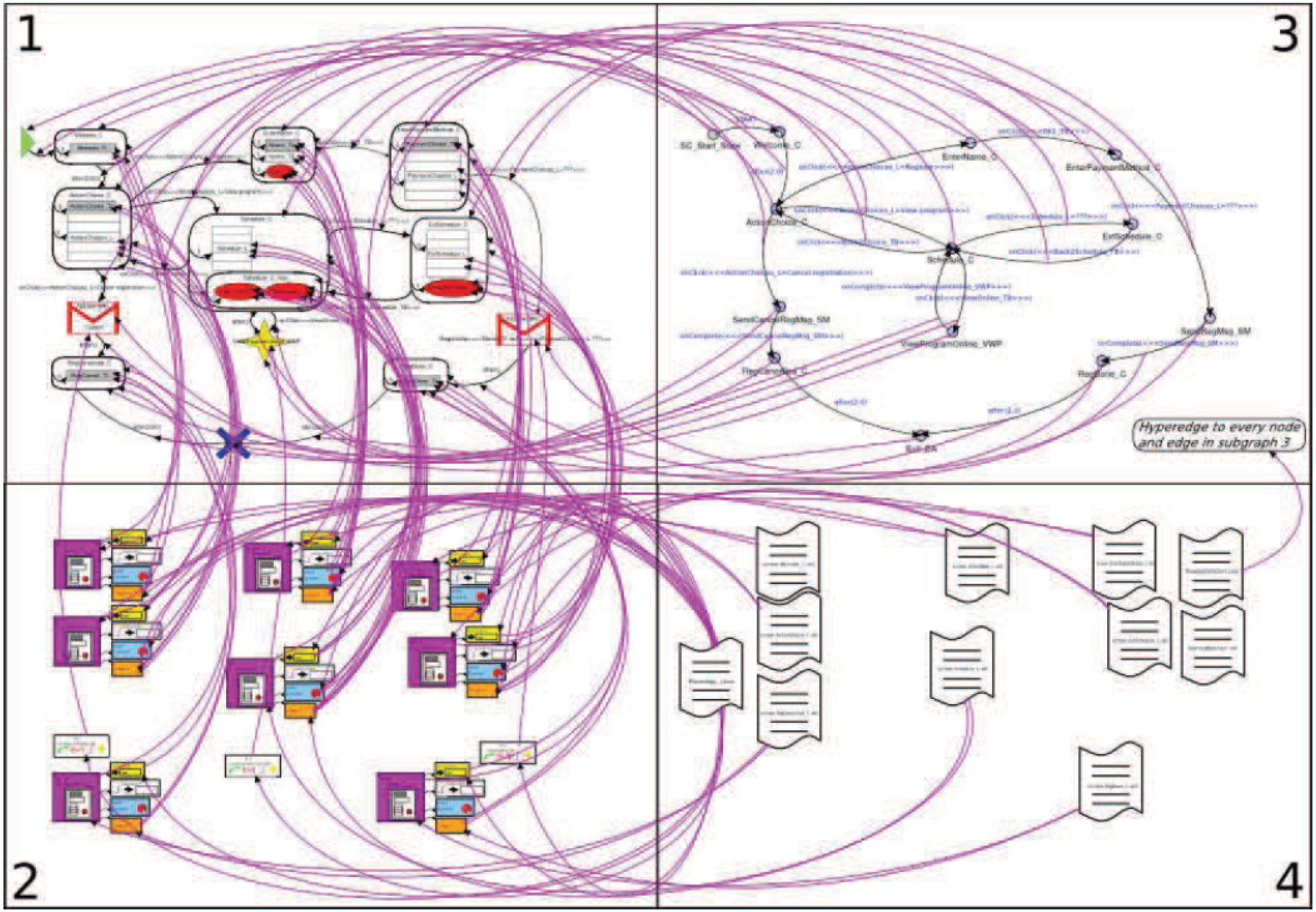


Figure 2. Design metrics (from left to right: adjacency preference, work style preference, buzz, productivity, daylight, and views to outside)

Model-Based Development:
Modify the Transformation
(e.g., target platform changes, or optimization)

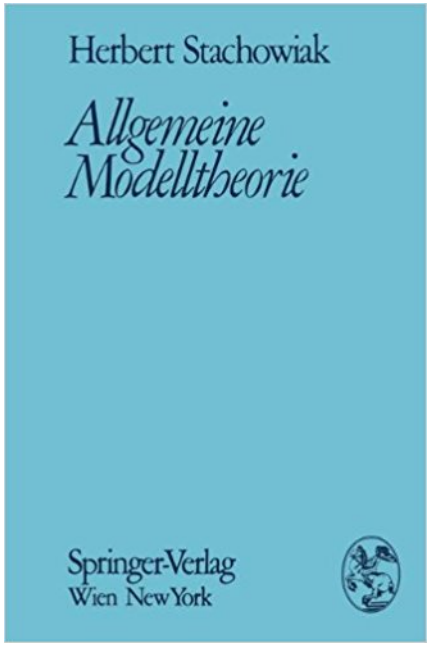


Can be Multi-Step/Multi-Formalism



Building DS(V)M Tools Effectively . . .

- **development cost** of DS(V)M Tools may be prohibitive!
- $\Rightarrow$ need **Modelling Language Engineering**

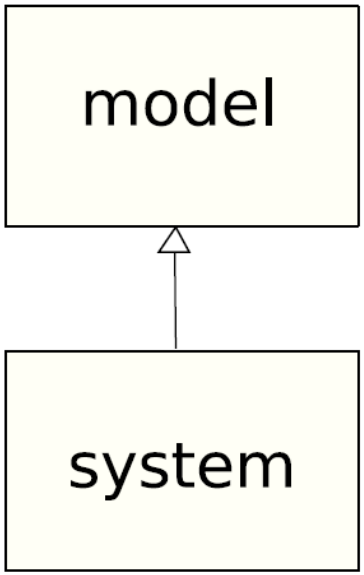


Model Features

mapping feature	A model is based on an original.[4]
reduction feature	A model only reflects a (relevant) selection of an original's properties.
pragmatic feature	A model needs to be usable in place of an original with respect to some purpose.

Jean Bézivin

Everything is a model !

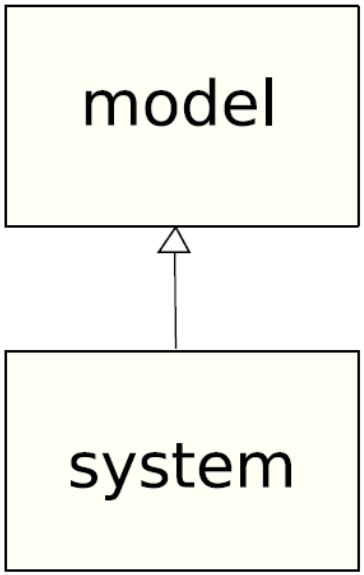


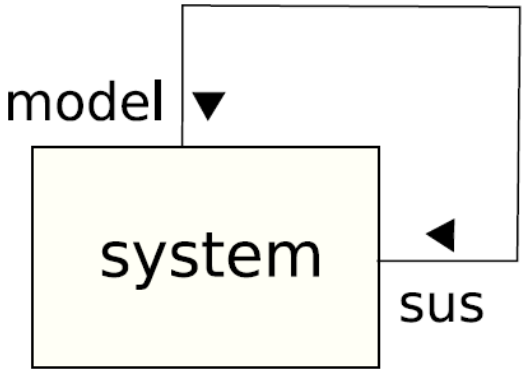
Jean Bézivin

Everything is a model !

Jean-Marie Favre

Nothing is a model !





Jean Bézivin

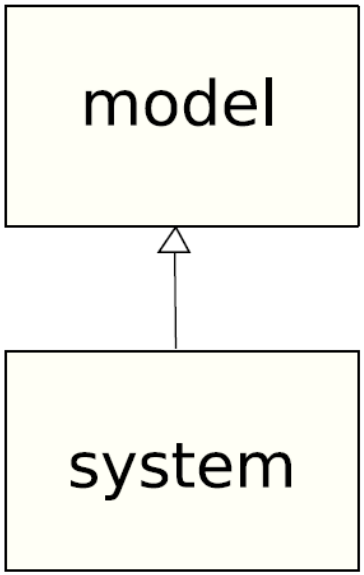
Everything is a model !

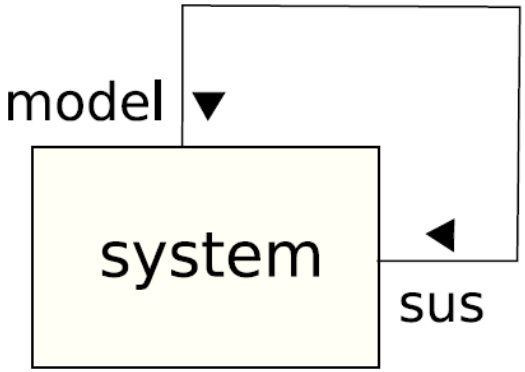
Jean-Marie Favre

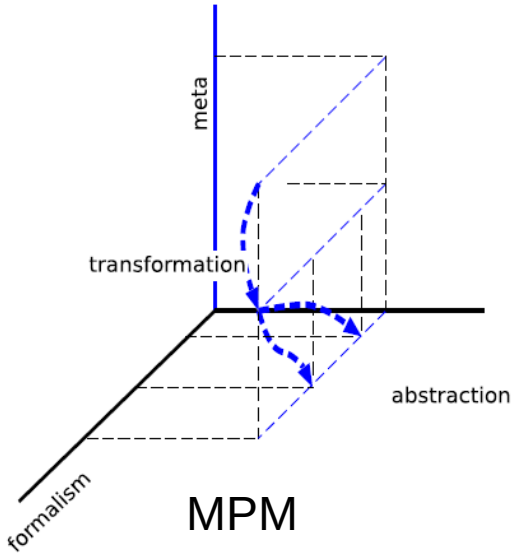
Nothing is a model !

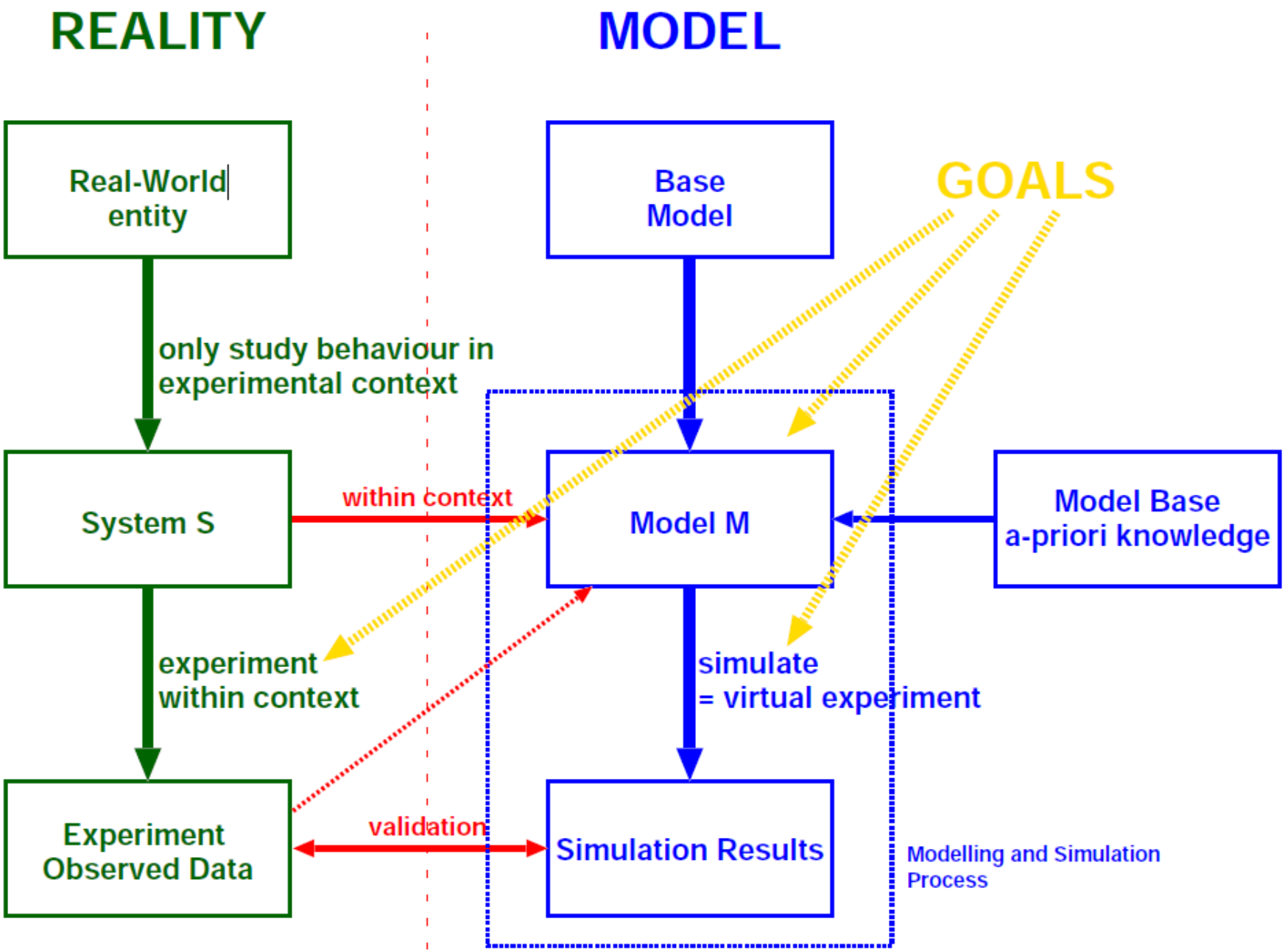
Hans Vangheluwe

Model everything !



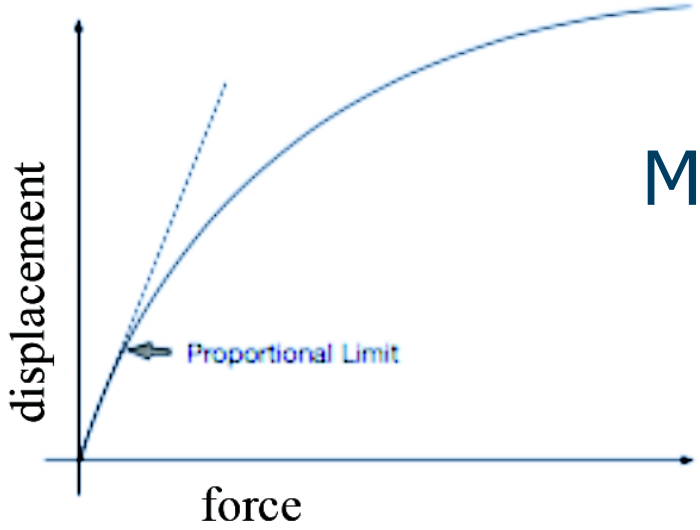






Bernard P. Zeigler. *Multi-faceted Modelling and Discrete-Event Simulation*. Academic Press, 1984.

Model Validity Range

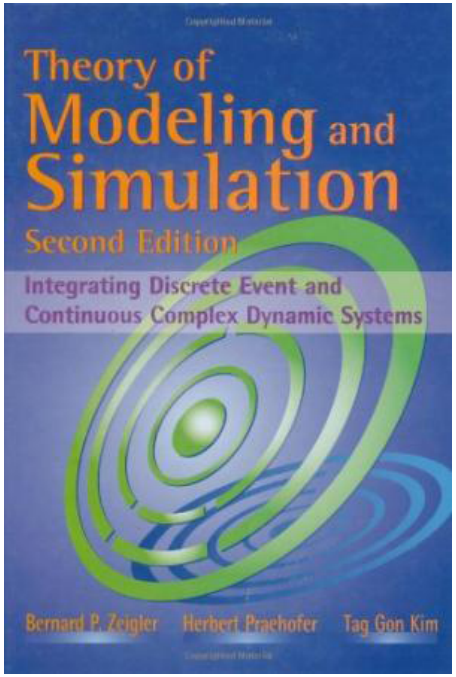


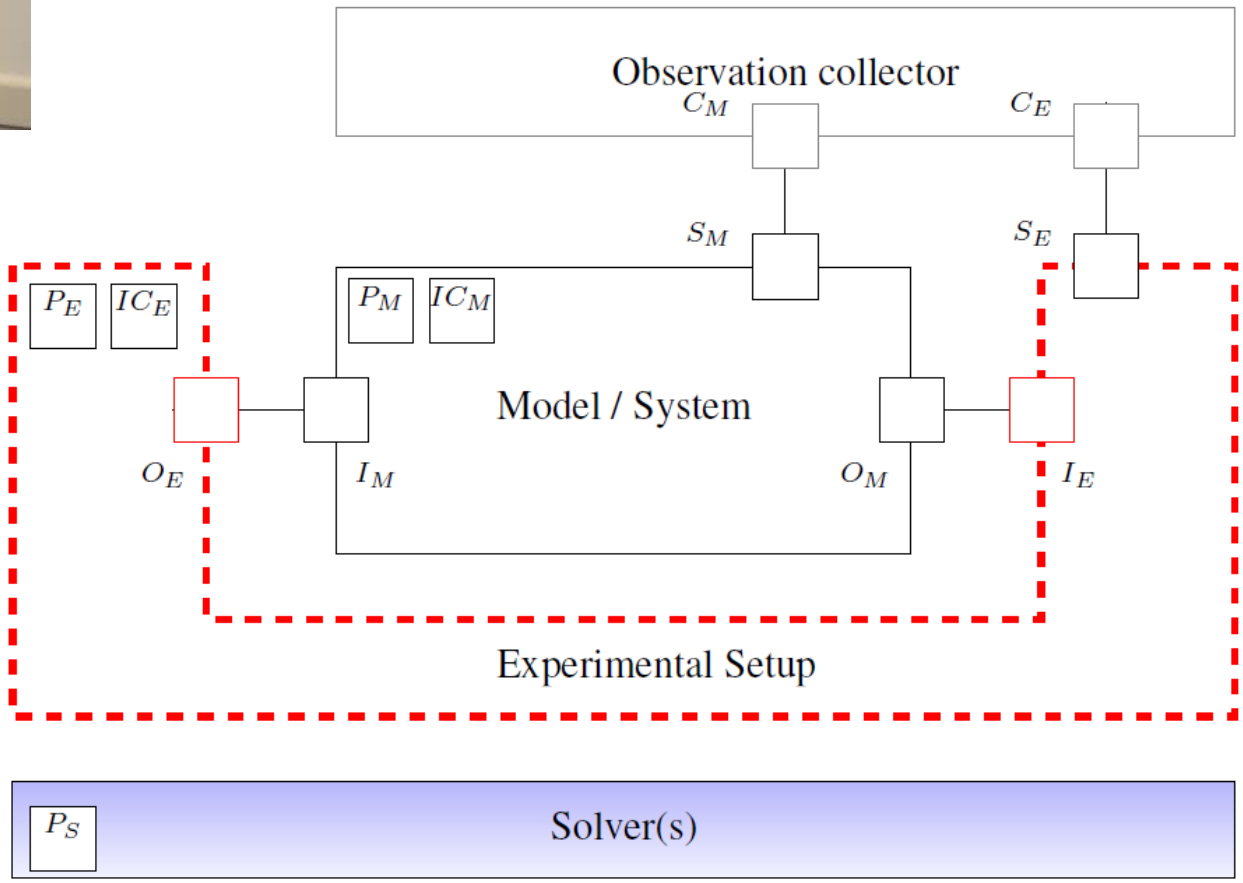
$$F = -kx$$

O.D. Inches	O.D. mm	CENTURY STOCK NUMBER	FREE LENGTH Inches	FREE LENGTH mm	I.D. Inches	I.D. mm	RATE Lbs./In.	RATE N/mm	SUGG. MAX. DEFL. Inches	SUGG. MAX. DEFL. mm	SUGG. MAX. LOAD Lbs.	SUGG. MAX. LOAD N	SOLID LENGTH Inches	SOLID LENGTH mm	WIRE DIA. Inches	WIRE DIA. mm	TOTAL COILS	MAT'L	ENDS	FINISH
0.036	.91	10075	.59	15.1	.022	.6	2.6	.46	.15	3.8	.39	1.7	.35	8.9	0.007	0.2	49.0	SST	C	N
0.036	.91	JJ-7	.63	15.9	.024	.6	1.6	.28	.16	4.1	.25	1.1	.25	6.2	0.006	0.2	40.0	SST	C	N
0.040	1.02	2924	.66	16.8	.020	.5	11	2.0	.13	3.2	1.4	6.4	.50	12.6	0.010	0.3	48.5	MW	C	N
0.040	1.02	10778	.69	17.5	.028	.7	1.0	.17	.35	8.9	.35	1.6	.30	7.7	0.006	0.2	49.5	MW	C	N
0.054	1.37	RR-6	.25	6.4	.036	.9	6.2	1.1	.09	2.2	.56	2.5	.16	4.1	0.009	0.2	16.5	SST	C	N
0.054	1.37	10619	.72	18.3	.038	1.0	1.6	.29	.37	9.3	.60	2.7	.32	8.1	0.008	0.2	39.0	MW	C	N
0.057	1.45	70000	.13	3.3	.045	1.1	3.7	.66	.07	1.7	.25	1.1	.04	1.0	0.006	0.2	5.75	MW	C	N
0.057	1.45	70000S	.13	3.3	.045	1.1	3.3	.57	.05	1.3	.17	.74	.04	1.0	0.006	0.2	5.75	SST	C	N
0.057	1.45	70009	.13	3.3	.043	1.1	6.9	1.2	.06	1.5	.40	1.8	.05	1.2	0.007	0.2	6.00	MW	C	N
0.057	1.45	70009S	.13	3.3	.043	1.1	6.0	1.1	.04	1.1	.26	1.2	.05	1.2	0.007	0.2	6.00	SST	C	N
0.057	1.45	70018	.13	3.3	.041	1.0	12	2.1	.05	1.2	.57	2.5	.06	1.4	0.008	0.2	6.13	MW	C	N
0.057	1.45	70018S	.13	3.3	.041	1.0	11	1.8	.03	.88	.37	1.6	.06	1.4	0.008	0.2	6.13	SST	C	N
0.057	1.45	70001	.19	4.8	.045	1.1	2.3	.40	.11	2.8	.25	1.1	.06	1.4	0.006	0.2	8.13	MW	C	N
0.057	1.45	70001S	.19	4.8	.045	1.1	2.0	.35	.08	2.1	.17	.74	.06	1.4	0.006	0.2	8.13	SST	C	N
0.057	1.45	70010	.19	4.8	.043	1.1	4.0	.70	.10	2.5	.40	1.8	.07	1.8	0.007	0.2	8.88	MW	C	N
0.057	1.45	70010S	.19	4.8	.043	1.1	3.5	.61	.07	1.9	.26	1.2	.07	1.8	0.007	0.2	8.88	SST	C	N
0.057	1.45	70019	.19	4.8	.041	1.0	7.4	1.3	.08	2.0	.57	2.5	.08	2.0	0.008	0.2	8.75	MW	C	N
0.057	1.45	70019S	.19	4.8	.041	1.0	6.4	1.1	.06	1.4	.37	1.6	.08	2.0	0.008	0.2	8.75	SST	C	N
0.057	1.45	70002	.25	6.4	.045	1.1	1.7	.30	.15	3.8	.25	1.1	.07	1.7	0.006	0.2	10.3	MW	C	N
0.057	1.45	70002S	.25	6.4	.045	1.1	1.5	.26	.11	2.8	.17	.74	.07	1.7	0.006	0.2	10.3	SST	C	N
0.057	1.45	70011	.25	6.4	.043	1.1	3.1	.54	.13	3.3	.40	1.8	.08	2.1	0.007	0.2	11.0	MW	C	N
0.057	1.45	70011S	.25	6.4	.043	1.1	2.7	.47	.10	2.5	.26	1.2	.08	2.1	0.007	0.2	11.0	SST	C	N
0.057	1.45	70020	.25	6.4	.041	1.0	5.3	.92	.11	2.8	.57	2.5	.10	2.5	0.008	0.2	11.5	MW	C	N
0.057	1.45	70020S	.25	6.4	.041	1.0	4.6	.80	.08	2.0	.37	1.6	.10	2.5	0.008	0.2	11.5	SST	C	N
0.057	1.45	70003	.31	7.9	.045	1.1	1.4	.24	.19	4.7	.25	1.1	.08	2.0	0.006	0.2	12.4	MW	C	N
0.057	1.45	70003S	.31	7.9	.045	1.1	1.2	.21	.14	3.6	.17	.74	.08	2.0	0.006	0.2	12.4	SST	C	N
0.057	1.45	70012	.31	7.9	.043	1.1	2.4	.42	.17	4.2	.40	1.8	.10	2.6	0.007	0.2	13.5	MW	C	N
0.057	1.45	70012S	.31	7.9	.043	1.1	2.1	.37	.12	3.2	.26	1.2	.10	2.6	0.007	0.2	13.5	SST	C	N
0.057	1.45	70021	.31	7.9	.041	1.0	4.1	.72	.14	3.6	.57	2.5	.12	3.1	0.008	0.2	14.3	MW	C	N
0.057	1.45	70021S	.31	7.9	.041	1.0	3.6	.62	.10	2.6	.27	1.6	.12	3.1	0.008	0.2	14.3	SST	C	N

www.centuryspring.com

Experimental (Validity) Frame





Joachim Denil, Stefan Klikovits, Pieter J. Mosterman, Antonio Vallecillo, and Hans Vangheluwe. The experiment model and validity frame in M&S. In Proceedings of the 2017 Symposium on Theory of Modeling and Simulation - DEVS , TMS/DEVS '17, part of the Spring Simulation Multi-Conference, pages 1085 – 1096. Society for Computer Simulation International, April 2017.

Abstraction Relationship

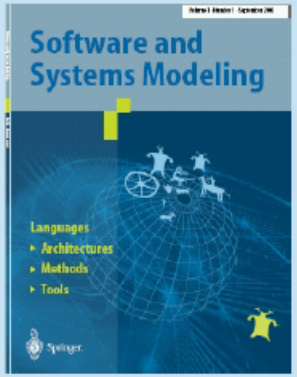
foundation: the *information* contained in a model M.
Different *questions* (properties) $P = I(M)$ which can be asked concerning the model.
These questions either result in true or false.

Abstraction and its opposite, *refinement* are
relative to a non-empty set of questions (properties) P.

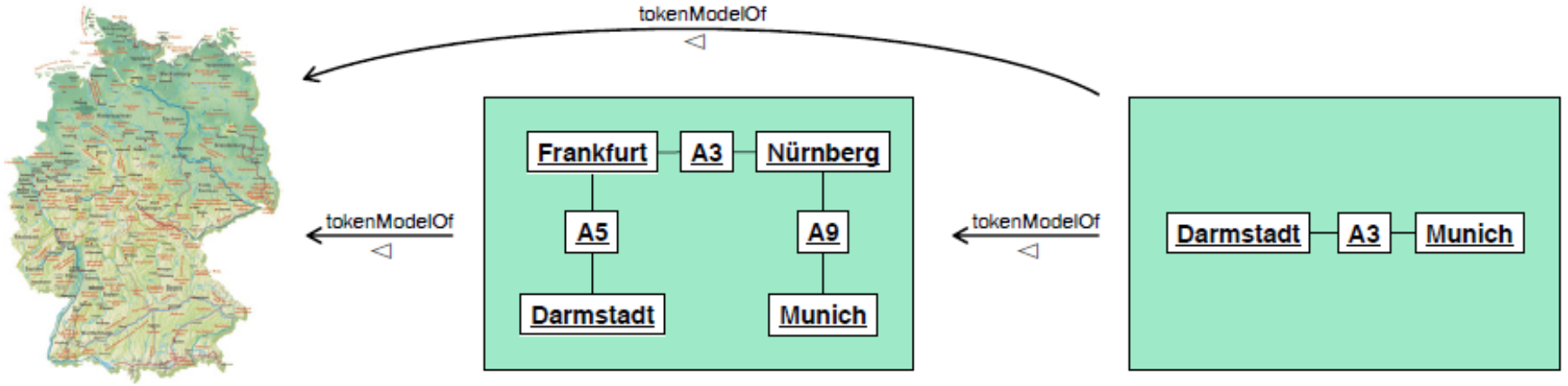
- If M_1 is an *abstraction* of M_2 with respect to P, for all $p \in P$: $M_1 \models p \Rightarrow M_2 \models p$. This is written $M_1 \sqsupseteq_P M_2$.
- M_1 is said to be a *refinement* of M_2 iff M_2 is an *abstraction* of M_1. This is written $M_1 \sqsubseteq_P M_2$.

Dissecting Modelling

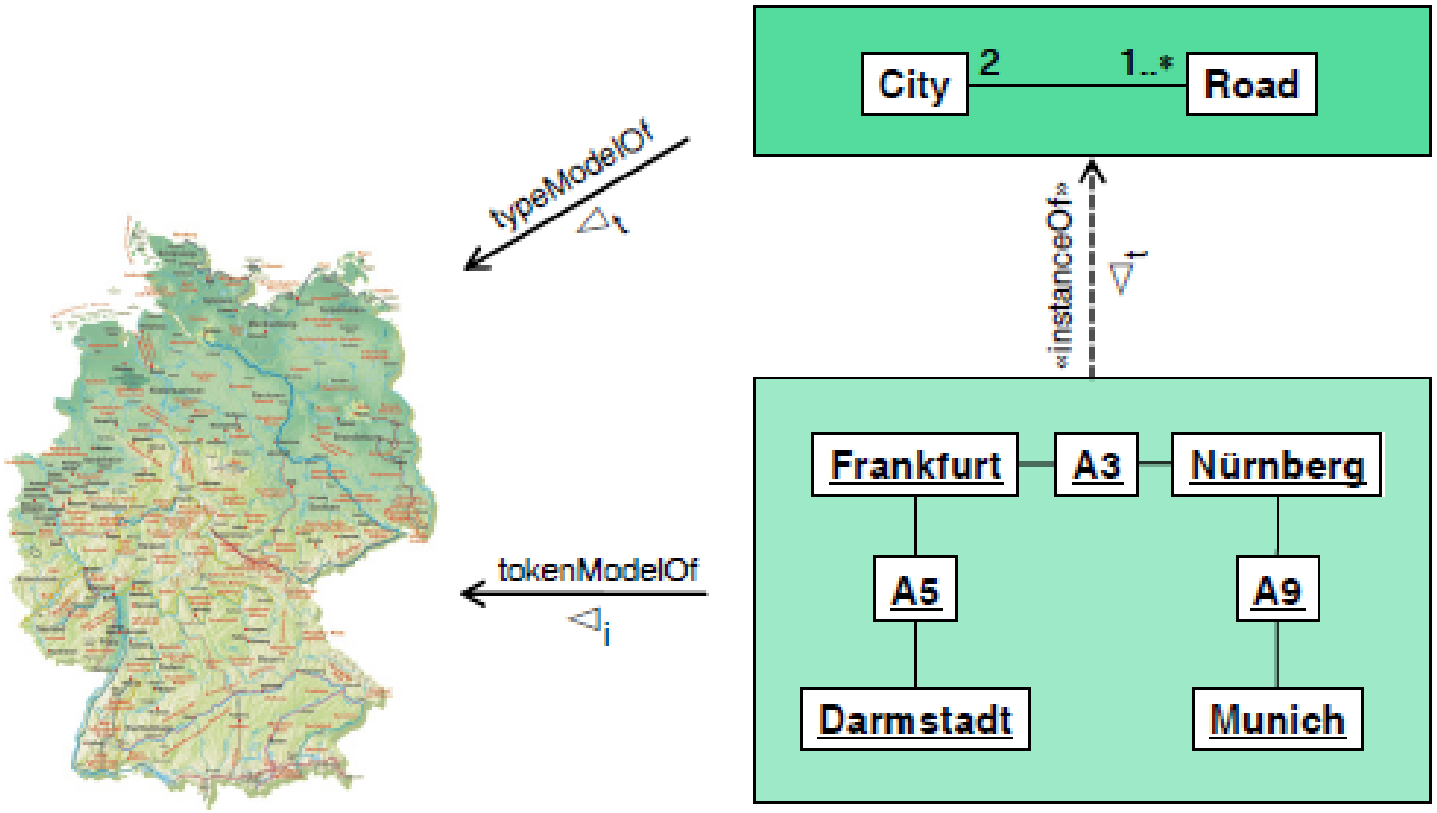
Thomas Kühne. *Matters of (Meta-)Modeling*.
Software and System Modeling 5(4): 369-385 (2006)



Token Models



Rôles a Model may Play



$$\mathcal{S} \lhd \mathcal{M}$$

$$\rho(\mathcal{S}, \mathcal{M}) \rightarrow \mathcal{S} \lhd \mathcal{M}$$

$$\alpha = \tau \circ \alpha' \circ \pi$$

$$h(a \oplus b) = h(a) \otimes h(b)$$

$$\pi(e_1) = \pi(e_2) \nrightarrow e_1 = e_2$$

$$e_1 \sim_\pi e_2 \rightarrow \pi(e_1) = \pi(e_2)$$

$$\varepsilon(C) = \{x \mid P(x)\}, \quad \text{where } P = \iota(C)$$

$$\forall x : \iota(C_{special})(x) \rightarrow \iota(C_{general_1})(x)$$

$$\varepsilon(C_{special}) \subseteq \varepsilon(C_{general})$$

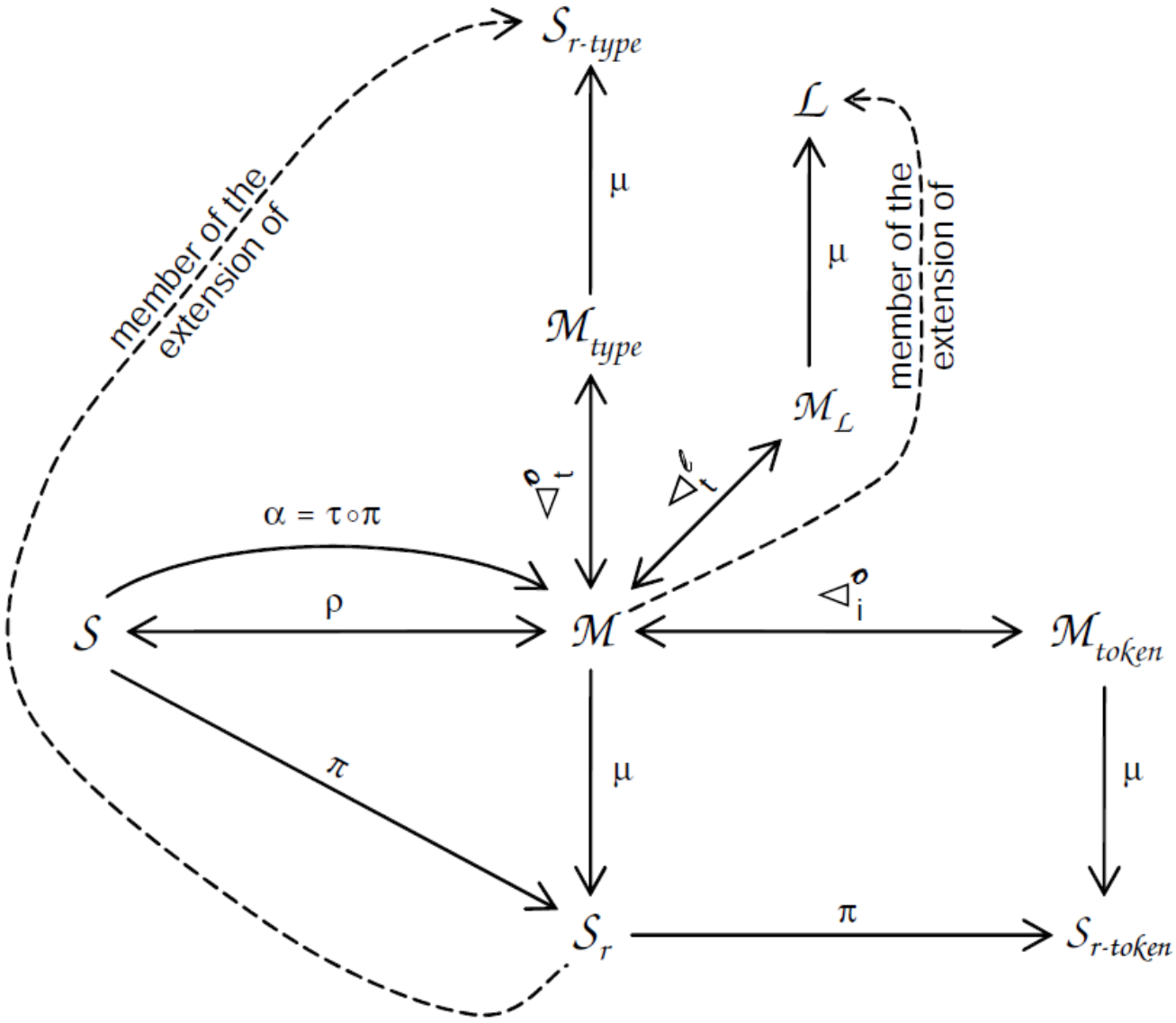
$$e_1 R^n e_2 = \begin{cases} \exists e : e_1 R^{n-1} e \wedge\ e R\, e_2 & , n > 1 \\ e_1 R\, e_2 & , n = 1 \end{cases}$$

acyclic $\forall\, e_1, e_2, n : e_1 R^n\, e_2 \rightarrow \neg\, e_2 R\, e_1$, and

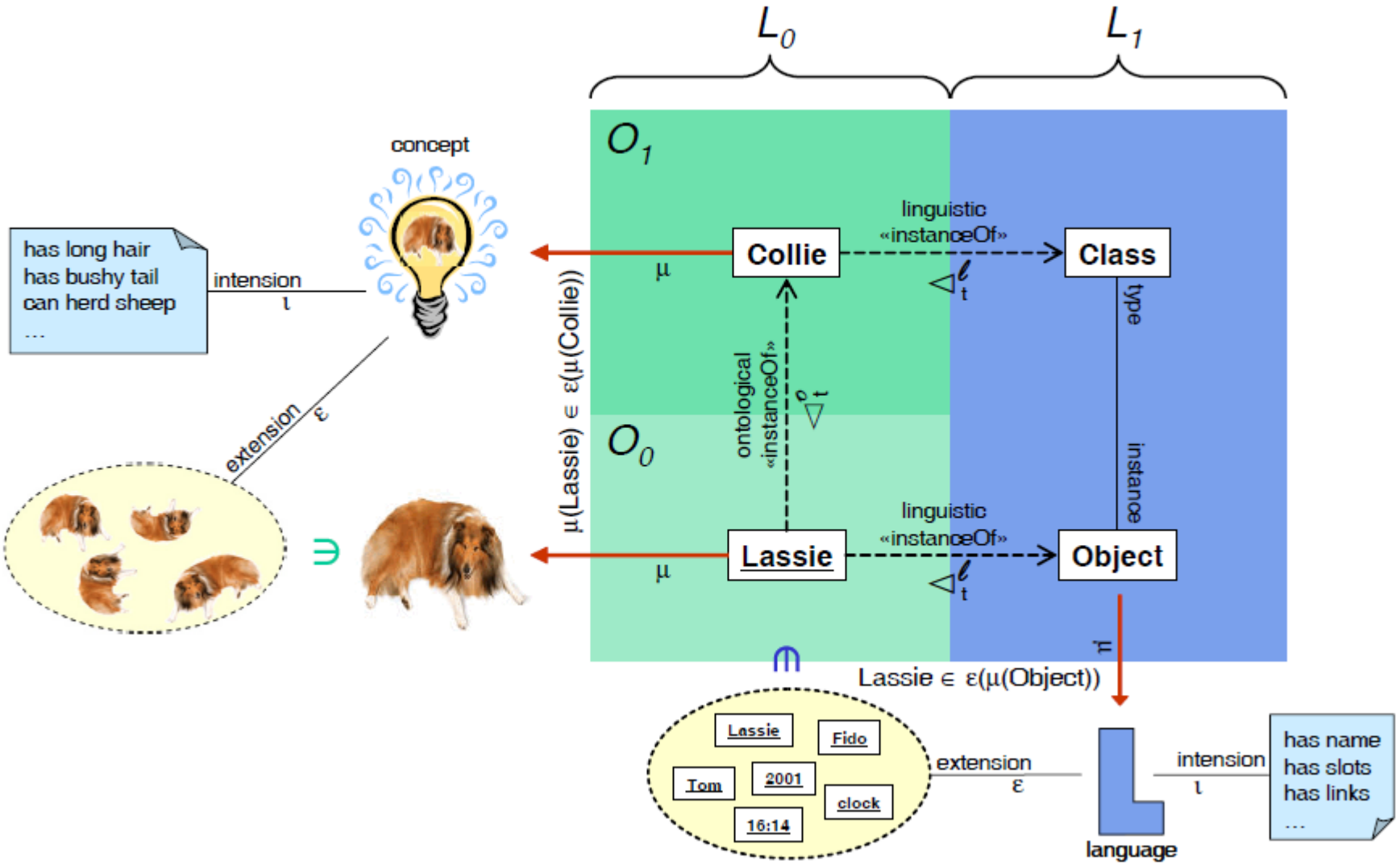
anti-transitive $\forall\, n \geq 2 : R^n \cap R = \emptyset$.

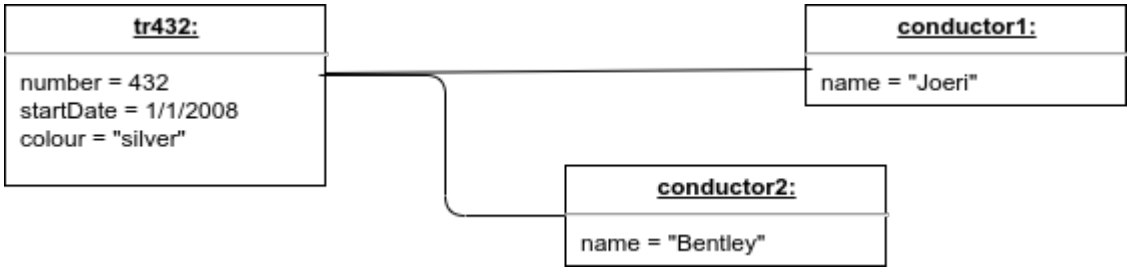
level respecting $\forall\, n, m :$
$(\exists\, e_1, e_2 : e_1 R^n e_2 \wedge\ e_1 R^m e_2) \rightarrow$
$n = m$

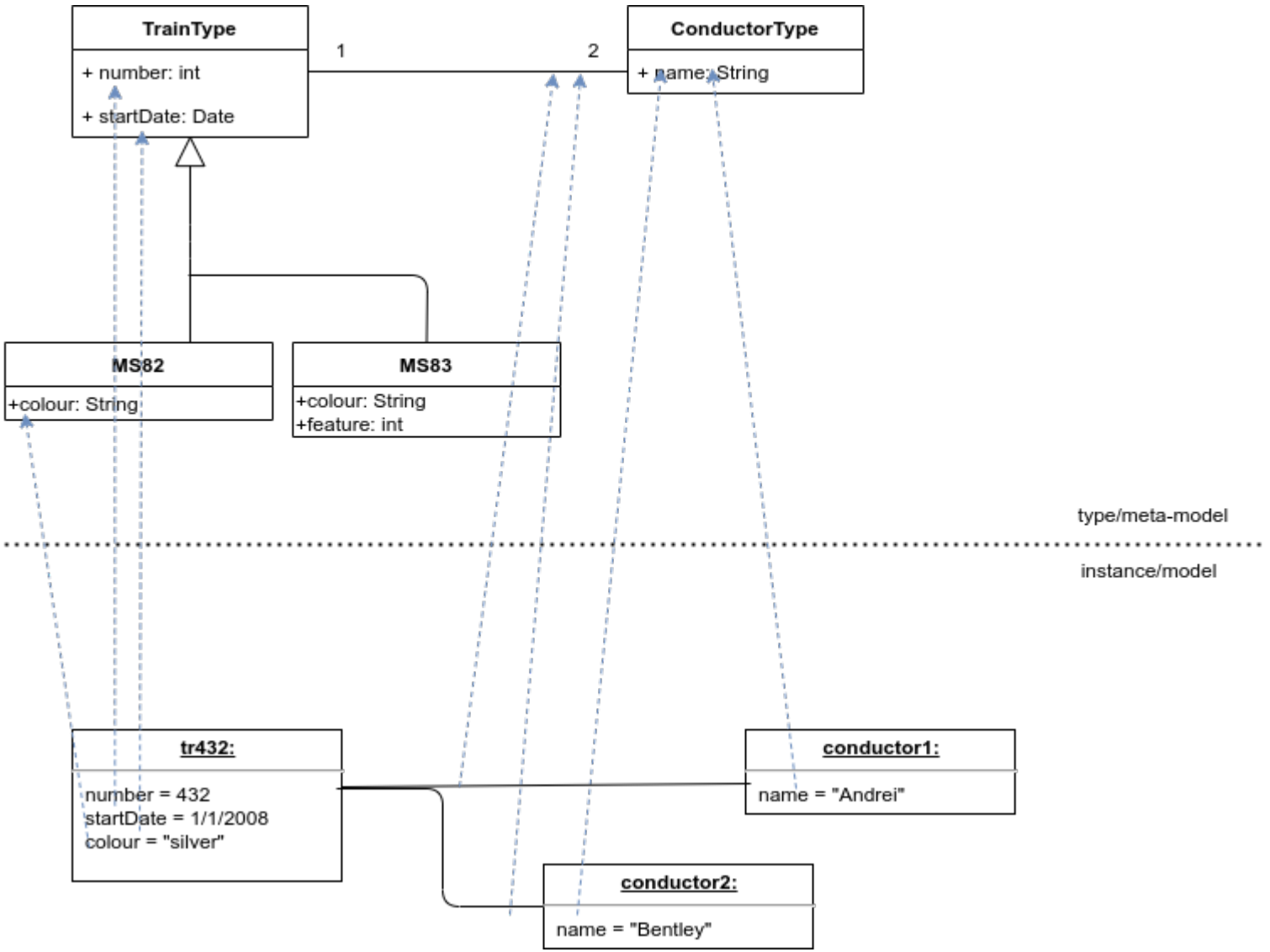
notation	*name*	*description*
α	abstraction	creates a model from a system using projection (π) and possibly classification (Λ) and generalization (Γ), hence $\mathbb{S} \lhd \alpha(\mathbb{S})$.
Λ	classification	creates a type model, hence $\mathcal{M} \lhd_t \Lambda(\mathcal{M})$
Γ	generalization	creates a supermodel, hence $\mathbb{S} \lhd_t \mathcal{M} \rightarrow \mathbb{S} \lhd_t \Gamma(\mathcal{M})$
π	projection	a homomorphic mapping creating a reduced system from a given system, using selection and reduction of information.
ρ	represents	records the intention of a model to represent a system.
μ	meaning	assigns meaning to a model (element). If $\rho(\mathbb{S}, \mathcal{M})$ then one may define $\mu(\mathcal{M}) = \pi(\mathbb{S})$.
$\lhd$	model-of	holds between a system and a model describing the former.
$\lhd_i$	token model-of	holds between a system and a model representing the former in a one-to-one fashion. Model elements may be regarded as designators for system elements.
$\lhd_t$	type model-of	holds between a system and a model classifying the former in a many-to-one fashion. Model elements are regarded as classifiers for system elements.
$\lhd^o$	ontological model-of	indicates that the model controls the *content* of its elements, hence $\mathbb{S} \lhd_t^o \mathcal{M} \rightleftharpoons \mu(\mathbb{S}) \in \varepsilon(\mu(\mathcal{M}))$ and $\mathbb{S} \lhd_i^o \mathcal{M} \rightleftharpoons \mu(\mathcal{M}) = \pi(\mu(\mathbb{S}))$. Assuming $\mu(\mathbb{S}) = \mathbb{S}$, for systems which do not model anything, we have $\mathbb{S} \lhd_i^o \mathcal{M} \rightleftharpoons \mu(\mathcal{M}) = \pi(\mathbb{S})$ and thus $\rho(\mathbb{S}, \mathcal{M}) \rightarrow \mathbb{S} \lhd_i^o \mathcal{M}$ (see definition of μ above).
$\lhd^l$	linguistic model-of	indicates that the model controls the *form* of its elements. This automatically implies $\lhd_t^l$ and, hence $\mathbb{S} \lhd_t^l \mathcal{M} \rightleftharpoons \mathbb{S} \in \varepsilon(\mu(\mathcal{M}))$

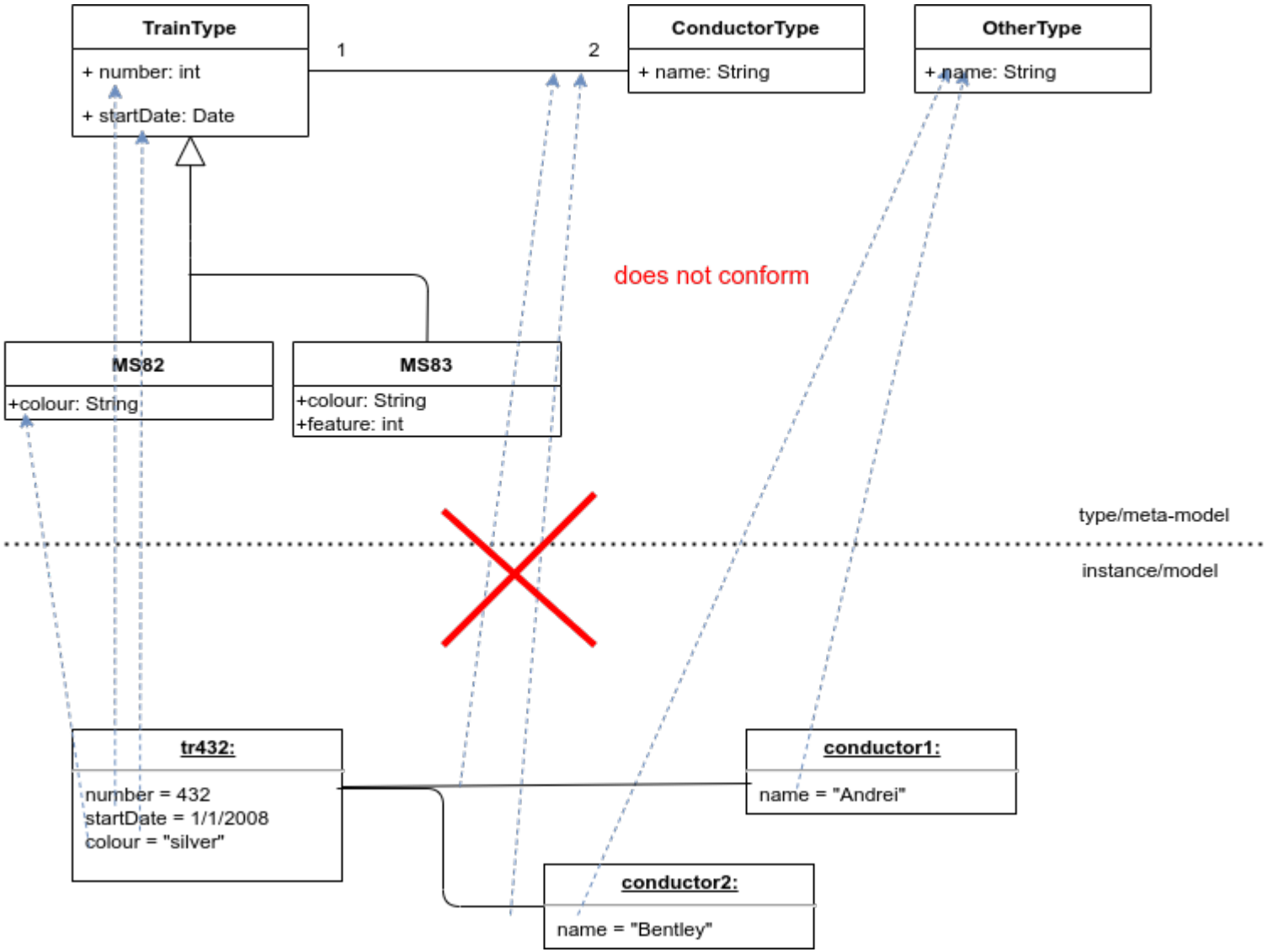

$\mathcal{S}_{r\text{-}type}$
$\mathcal{L}$
μ
μ
member of the extension of
member of the extension of
$\mathcal{M}_{type}$
$\mathcal{M}_{\mathcal{L}}$
$\triangle_t^{\circ}$
$\triangle_l^{\circ}$
$\alpha = \tau \circ \pi$
ρ
$\mathcal{S}$
$\mathcal{M}$
$\triangle_i^{\circ}$
$\mathcal{M}_{token}$
π
μ
μ
$\mathcal{S}_r$
π
$\mathcal{S}_{r\text{-}token}$

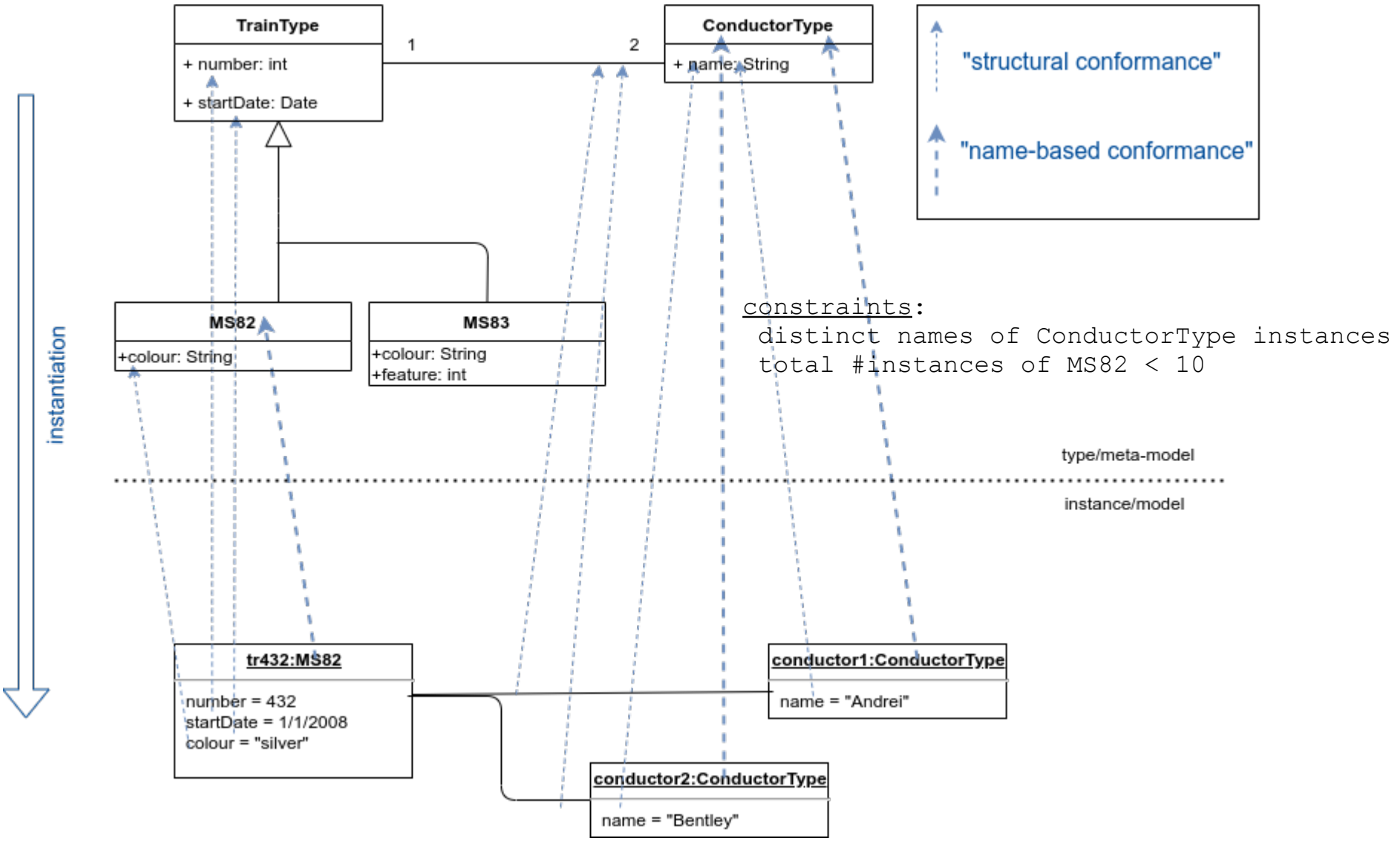
Ontological vs. Linguistic Conformance/Instatiation

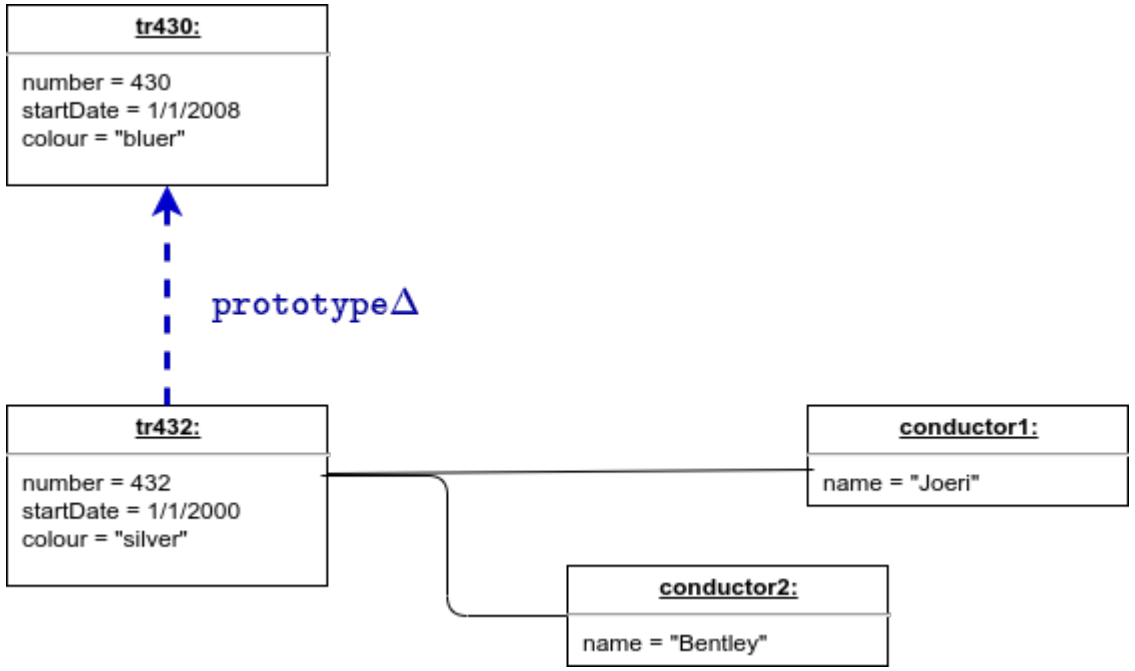



tr432:
number = 432
startDate = 1/1/2008
colour = "silver"
conductor1:
name = "Joeri"
conductor2:
name = "Bentley"


TrainType
+ number: int
+ startDate: Date
1
2
ConductorType
+ name: String
MS82
+colour: String
MS83
+colour: String
+feature: int
type/meta-model
instance/model
tr432:
number = 432
startDate = 1/1/2008
colour = "silver"
conductor1:
name = "Andrei"
conductor2:
name = "Bentley"


TrainType
+ number: int
+ startDate: Date
1
2
ConductorType
+ name: String
OtherType
+ name: String
does not conform
MS82
+colour: String
MS83
+colour: String
+feature: int
type/meta-model
instance/model
tr432:
number = 432
startDate = 1/1/2008
colour = "silver"
conductor1:
name = "Andrei"
conductor2:
name = "Bentley"


TrainType
+ number: int
+ startDate: Date
1
2
ConductorType
+ name: String
"structural conformance"
"name-based conformance"
MS82
+colour: String
MS83
+colour: String
+feature: int
constraints:
distinct names of ConductorType instances
total #instances of MS82 < 10
instantiation
type/meta-model
instance/model
tr432:MS82
number = 432
startDate = 1/1/2008
colour = "silver"
conductor1:ConductorType
name = "Andrei"
conductor2:ConductorType
name = "Bentley"


tr430:
number = 430
startDate = 1/1/2008
colour = "bluer"
prototypeΔ
tr432:
number = 432
startDate = 1/1/2000
colour = "silver"
conductor1:
name = "Joeri"
conductor2:
name = "Bentley"

metaDepth http://metadepth.org/

Textual language for (multi-level) (meta-)modelling

Constraint and operations in the Epsilon Object Language (EOL)
`https://www.eclipse.org/epsilon/`

EOL is similar the UML's Object Constraint Language (OCL)

EOL can be used in metaDepth for:

- Queries within the model (limited to navigation)
- Computation/assignment, for example to give operational semantics

metaDepth: meta-model

```
Model TrainNet {

     Node Train{
          train_name : String {id};
          maxVelocity : int;
          maxVConstraint: $self.maxVelocity > 0 and self.maxVelocity <= 300$
          currSegment : Segment[1];
     }

     Node Segment{
               currTrain : Train[0..1];
               nextSegment : Segment[0..1];
     }


     Node FastTrain : Train {}
     Node IntercityExpress: Train {}
}
```

metaDepth: model

```
TrainNet tn {
    Train train1 {train_name = "T3456"; maxVelocity = 300; currSegment = seg1;}

    IntercityExpress train2 {train_name = "ICE1234"; maxVelocity = 250;}

    Segment seg1 {currTrain = train1;}
    Segment seg2 {}
}`      `
```

Verifying conformance between `tn` and `TrainNet`:

Constraint fails: `cardinality 0 nota allowed for currSegment in train2`

Reason: `train2` is not on a `Segment`

metaDepth - Epsilon Object Language (EOL)

Example query: What are the trains in each segment?

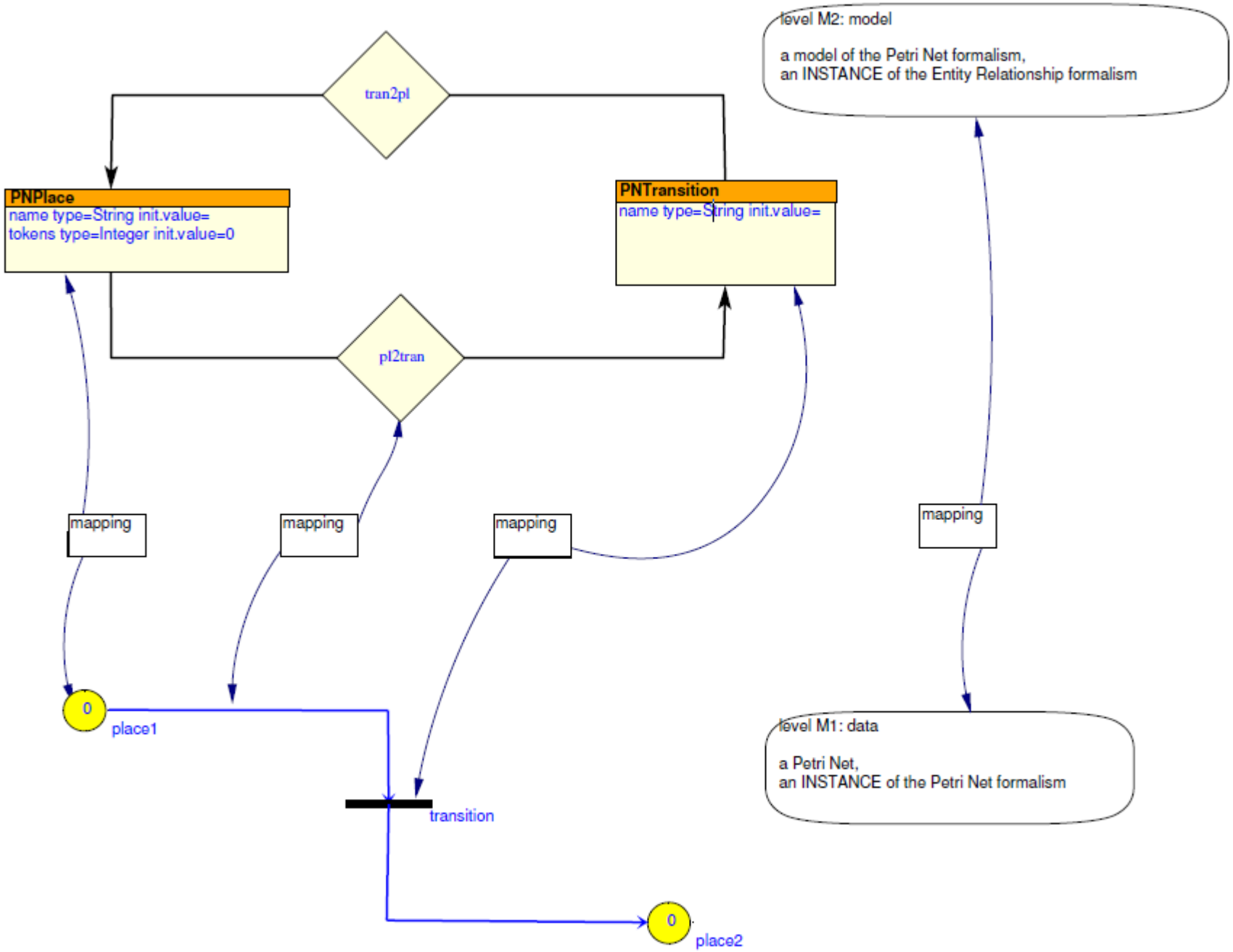
```
operation main(){
    for (s in Segment.allInstances()){
        if (Train.allInstances.excludes(s.currTrain)){
          ('Segment ' + s + "has no train.").println();
        }
        else{
          ("Segment " + s + " has train: '" + s.currTrain.toString + "'").println();
        }
}
```

Result:

```
Segment seg2 has no train.
Segment seg1 has train 'train1'
```

Caveat: explicit navigation (via . . .), need to know types
(or be able to query: Types.allInstances())

Linguistic Conformance: morphism



Name-based conformance checking Algorithm 1

```
Input: model, type_model
 1: type_to_elements ← populate_types(model, type_model)
 2: for el in model do
 3:     if not type(el) in type_to_elements then
 4:         return False
 5:     end if
 6:     append el to type_to_elements[type(el)]
 7: end for
 8: for type in type_to_elements do
 9:     if len(type_to_elements[type]) < get_minimum(type) then
10:         return False
11:     end if
12:     if len(type_to_elements[type]) > get_maximum(type) then
13:         return False
14:     end if
15:     for el in type_to_elements[type] do
16:         for attr in get_attributes(el) do
17:             if not type(attr) in get_attributes(type) then
18:                 return False
19:             end if
20:             if not type(get_value(attr)) = get_type(type(attr)) then
21:                 return False
22:             end if
23:         end for
24:     end for
25:     out_associations ← classify_by_type(get_out_associations(el))
26:     for assoc_type in out_associations do
27:         if not(get_in_type(assoc_type) in get_all_types(el)) then
28:             return False
29:         end if
30:         if len(out_associations[assoc_type]) < get_minimum_out(assoc_type) then
31:             return False
32:         end if
33:         if len(out_associations[assoc_type]) > get_maximum_out(assoc_type) then
34:             return False
35:         end if
36:     end for
37:     in_associations ← classify_by_type(get_in_associations(el))
38:     for assoc_type in in_associations do
39:         if not(get_out_type(assoc_type) in get_all_types(el)) then
40:             return False
41:         end if
42:         if len(in_associations[assoc_type]) < get_minimum_in(assoc_type) then
43:             return False
44:         end if
45:         if len(in_associations[assoc_type]) > get_maximum_in(assoc_type) then
46:             return False
47:         end if
48:     end for
49: end for
50: return True
```


In Algorithm 1, the linguistic conformance check of the MvK is shown. It checks whether a given model conforms to a given type model. It consists of four checks:

1. (Lines 1-7) Checks whether all elements in the model are typed by an element in the type model. Functions used are:
   (a) *populate_types*: returns a mapping between types of the type model, and instances of those type in the model.
   (b) *type*: returns the type of an element.
2. (Lines 8-14) Checks whether the minimum and maximum cardinality for each type in the type model is satisfied. Functions used are:
   (a) *get_mimimum*: returns the minimum cardinality for an given type.
   (b) *get_maximum*: returns the maximum cardinality for an given type.
3. (Lines 15-24) Checks, for all attributes of all elements in the model, whether a type definition for the attribute can be found in the type model (lines 17-19), and whether the type of the attribute value corresponds to the type defined in the attribute type (lines 20-22). Functions used are:
   (a) *get_attributes*: returns all attributes of an element.
   (b) *get_value*: returns the value of an attribute.
   (c) *get_type*: returns the type of the values which can be assigned to an attribute.
4. (Lines 25-49) Checks, for each incoming and outgoing association of each element of the model, whether, respectively, the incoming and outgoing cardinalities are satisfied. Also checks whether the types of the connected elements correspond to those defined in the association. Functions used are:
   (a) *classify_by_type*: classifies the given elements by their type, and returns a mapping between types and instances.
   (b) *get_out_associations*: returns all outgoing associations of an element.
   (c) *get_in_associations*: returns all incoming associations of an element.
   (d) *get_minimum_out*: returns the minimum number of outgoing associations of a particular type.

Name-based conformance checking Algorithm 1

```
Input: model, type_model
type_to_elements ← populate_types(model, type_model)
for el in model do
    if not type(el) in type_to_elements then
        return False
    end if
    append el to type_to_elements[type(el)]
end for
for type in type_to_elements do
    if len(type_to_elements[type]) < get_minimum(type) then
        return False
    end if
    if len(type_to_elements[type]) > get_maximum(type) then
        return False
    end if
    for el in type_to_elements[type] do
        for attr in get_attributes(el) do
            if not type(attr) in get_attributes(type) then
                return False
            end if
            if not type(get_value(attr)) = get_type(type(attr)) then
                return False
            end if
        end for
    end for
    out_associations ← classify_by_type(get_out_associations(el))
    for assoc_type in out_associations do
        if not(get_in_type(assoc_type) in get_all_types(el)) then
            return False
        end if
        if len(out_associations[assoc_type]) < get_minimum_out(assoc_type) then
            return False
        end if
        if len(out_associations[assoc_type]) > get_maximum_out(assoc_type) then
            return False
        end if
    end for
    in_associations ← classify_by_type(get_in_associations(el))
    for assoc_type in in_associations do
        if not(get_out_type(assoc_type) in get_all_types(el)) then
            return False
        end if
        if len(in_associations[assoc_type]) < get_minimum_in(assoc_type) then
            return False
        end if
        if len(in_associations[assoc_type]) > get_maximum_in(assoc_type) then
            return False
        end if
    end for
end for
return True
```

(e) *get_maximum_out*: returns the maximum number of outgoing associations of a particular type.
(f) *get_minimum_in*: returns the minimum number of incoming associations of a particular type.
(g) *get_maximum_in*: returns the maximum number of incoming associations of a particular type.
(h) *get_out_type*: returns the type defined for the outgoing multiplicity of an association.
(i) *get_in_type*: returns the type defined for the incoming multiplicity of an association.
(j) *get_all_types*: returns the type of the element, as well as all its subtypes.

MetaDepth http://metadepth.org/

Meta-model ↔ model

```
strict Model PetriNets@1 {
  abstract Node NamedElement{
    name  : String {id};
  }
  Node Place : NamedElement {
    tokens   : int = 0;
    outTrans : Transition[*] {ordered,unique};
    inTrans  : Transition[*] {ordered,unique};
    minTokens: $self.tokens>=0$
  }
  Node Transition : NamedElement {
    inPlaces : Place[*] {ordered,unique};
    outPlaces: Place[*] {ordered,unique};
  }
  Edge ArcPT (Place.outTrans, Transition.inPlaces) {
    weight: int = 1;
  }
  Edge ArcTP (Transition.outPlaces, Place.inTrans) {
    weight: int = 1;
  }
  minWeight(ArcTP, ArcPT): $self.weight>0$
  minPlaces:$Place.allInstances().size()>0$
}
```

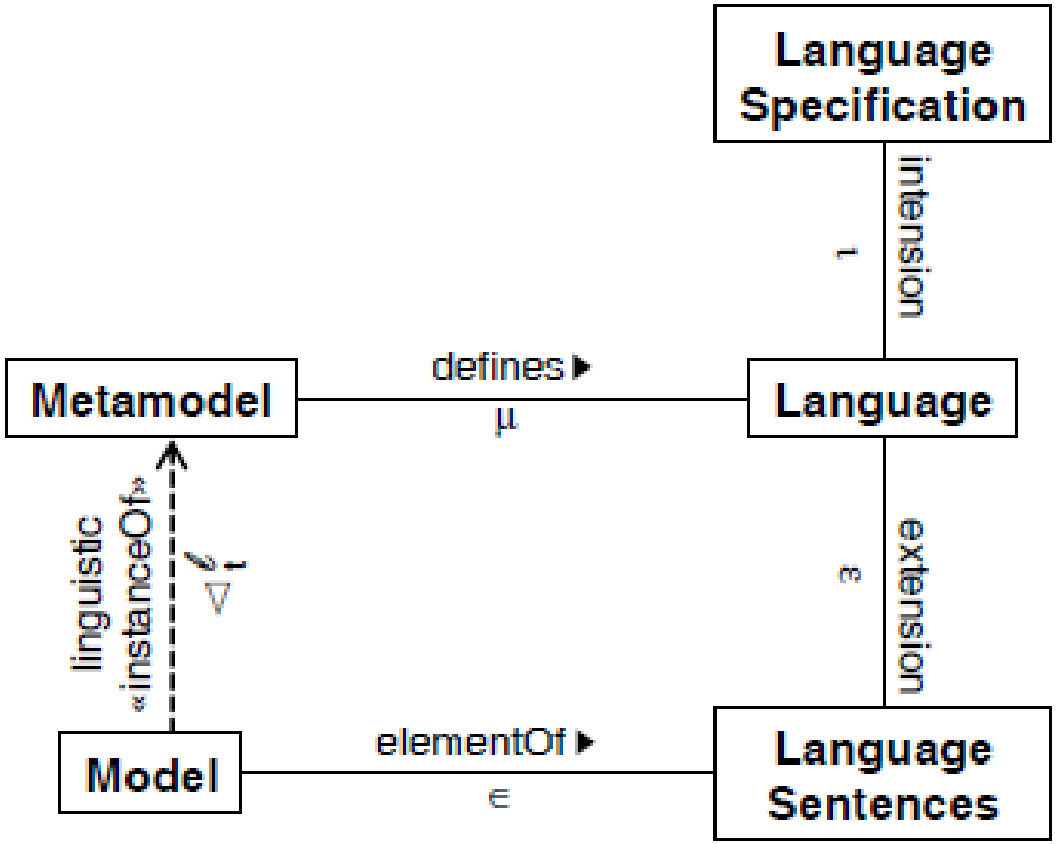
```
load "PetriNets"
PetriNets Test{
  Place p0{name="p0"; tokens=2;}
  Place p1{name="p1"; tokens=0;}
  Place p2{name="p2"; tokens=2;}
  Transition t1{name="t1";}
  ArcPT pt0(p0,t1){weight=1;}
  ArcPT pt1(p2,t1){weight=1;}
  ArcTP tp(t1,p1){weight=2;}
}
```

MetaDepth

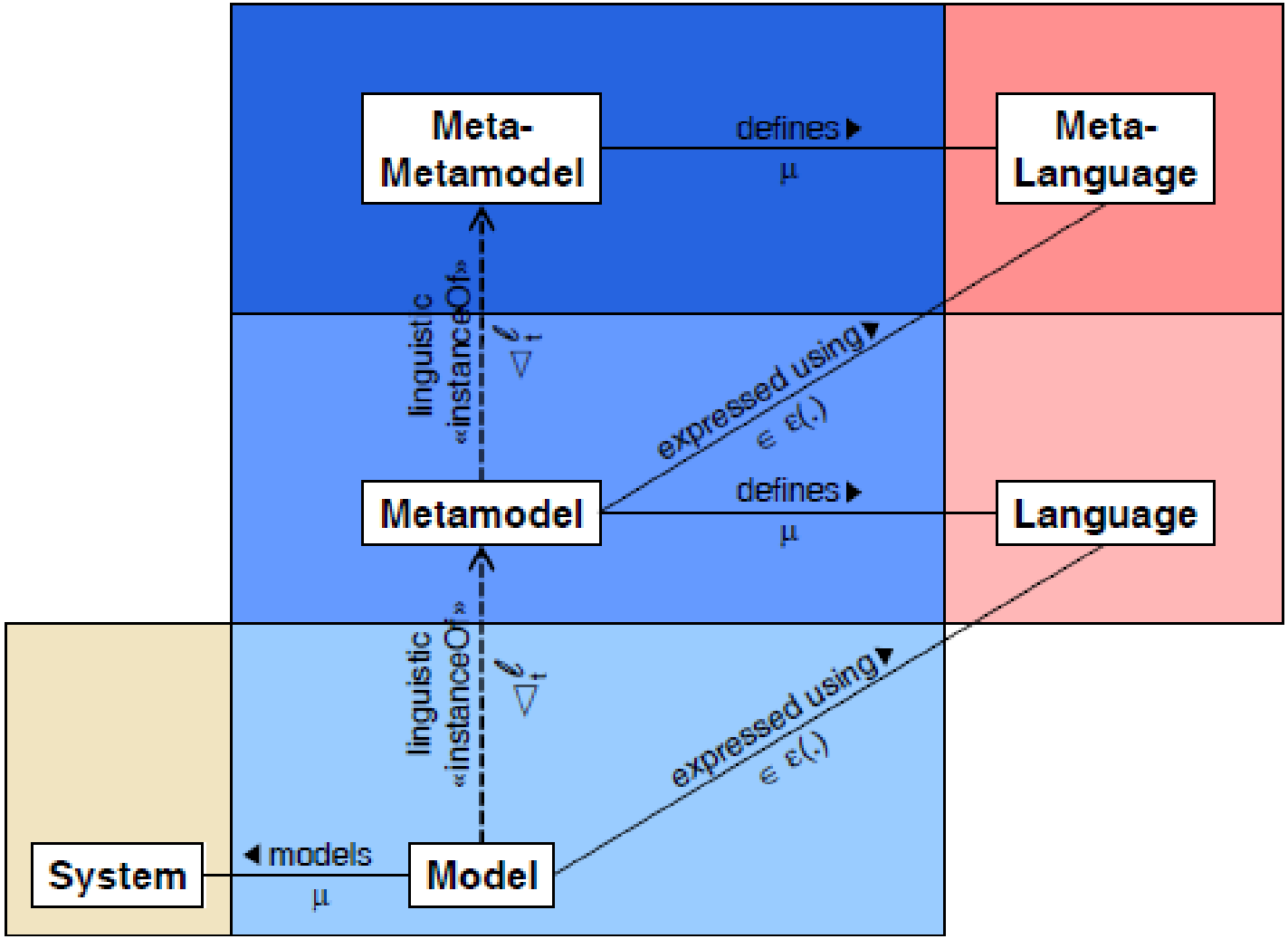
operational semantics

```
operation main() {
  'Simulating the Petri Net'.println();
  while (Transition.allInstances()->exists(t|t.enabled() and t.fire()) ) {}
}
operation Transition enabled() : Boolean {
  ('checking enabledness of '+self.name).println();
  return self.ArcPT->forAll(arc| arc.inPlaces.tokens>=arc.weight);
}
operation Transition fire() : Boolean {
  ('Firing '+self.name).println();
  for (arc in self.ArcPT)
    arc.inPlaces.tokens := arc.inPlaces.tokens-arc.weight;
  for (arc in self.ArcTP)
    arc.outPlaces.tokens := arc.outPlaces.tokens+arc.weight;
  return true;
}
```

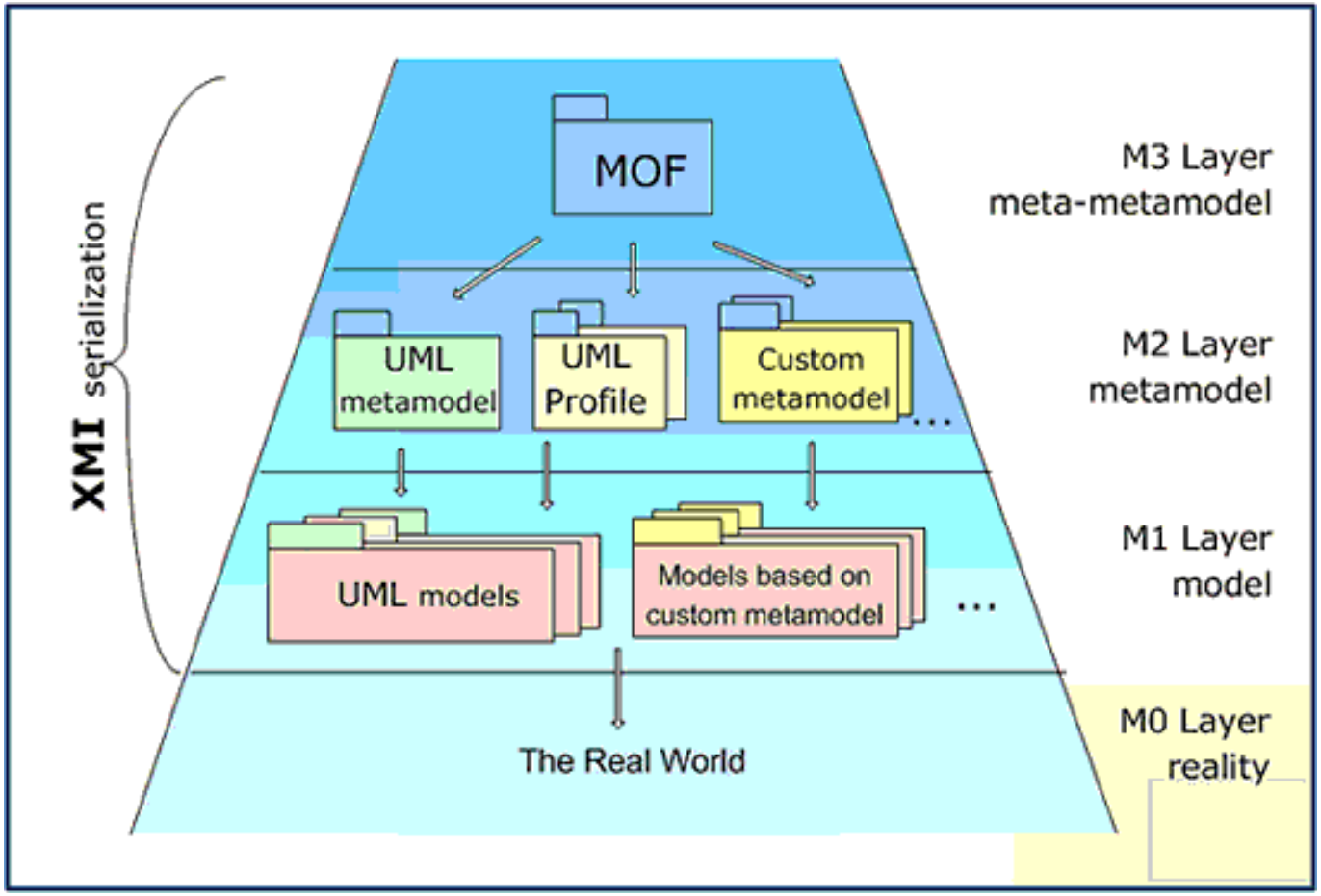
Meta-models as Language Definitions

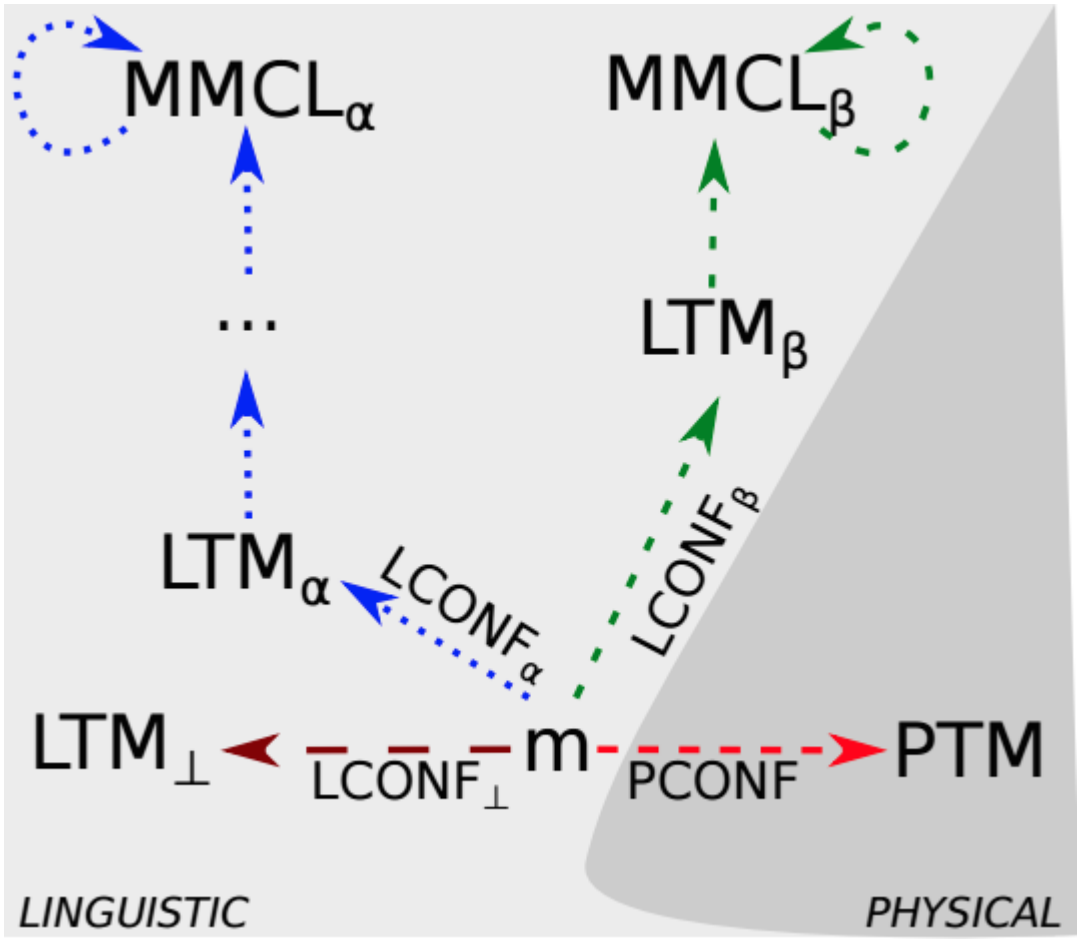


Language Definition Stack



Meta-hierarchy – OMG's 4 Layer Architecture



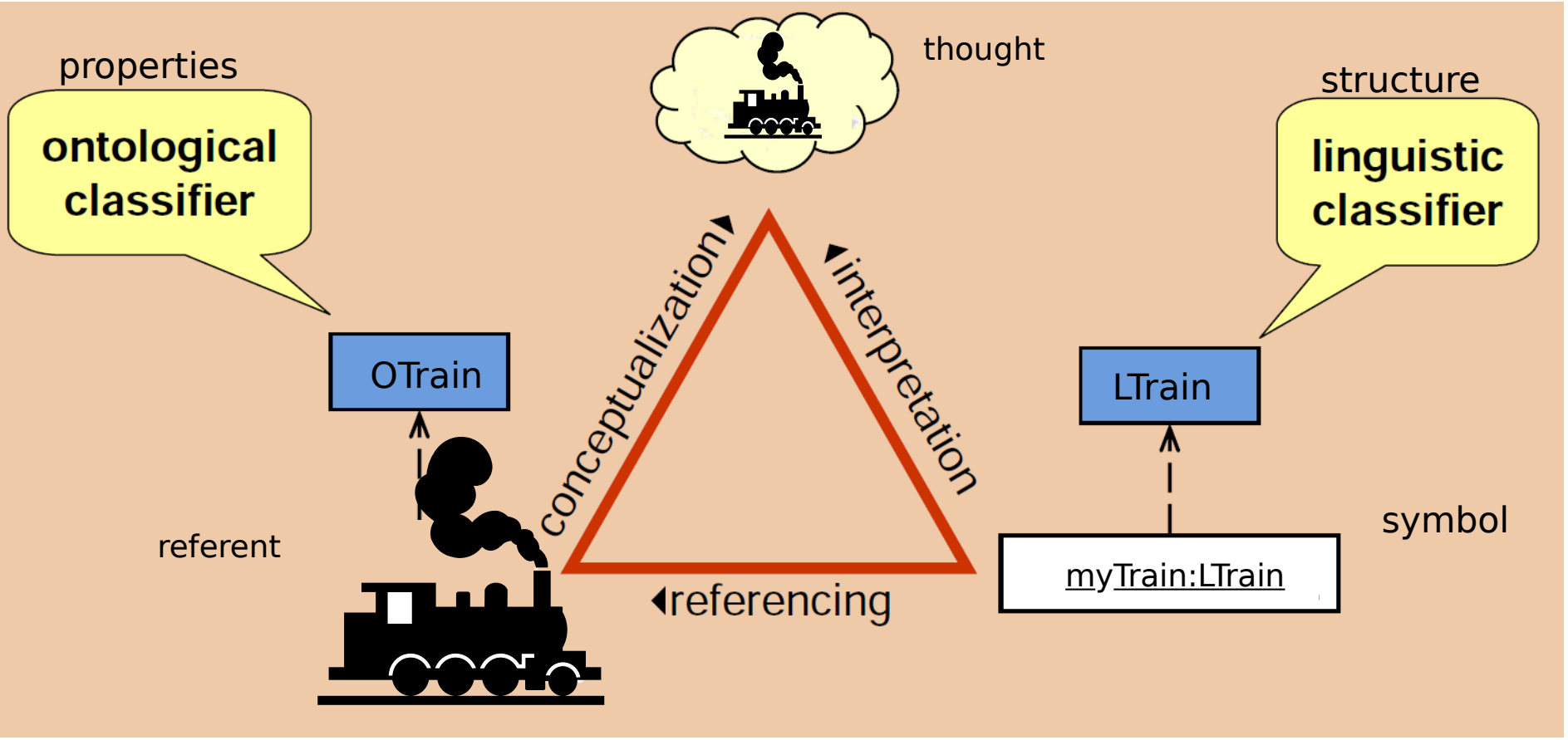


Simon Van Mierlo, Bruno Barroca, Hans Vangheluwe, Eugene Syriani, and Thomas Kuehne. Multi-level modelling in the Modelverse. In The first Workshop on Multi-Level Modelling (MULTI), co-located with the 17th International Conference on Model Driven Engineering Languages and Systems (MoDELS), Valencia, Spain, volume 1286 of CEUR Workshop Proceedings , pages 83 - 92, 2014.

Yentl Van Tendeloo, Bruno Barroca, Simon Van Mierlo, Hans Vangheluwe. Modelverse specification. 2016. http://msdl.cs.mcgill.ca/people/yentl/files/Modelverse.pdf

Triangle of Meaning
aka triangle of reference or semiotic triangle
Ogden and Richards in “the meaning of meaning” (1923)

caveat: collaborative modelling only works if collaborators have a shared interpretation



Syntax, **Semantics**, and all that Stuff

David Harel, Bernhard Rumpe.
Meaningful Modeling: What's the Semantics of "Semantics"?
IEEE Computer, vol. 37, no. 10, pp. 64-72, October, 2004.

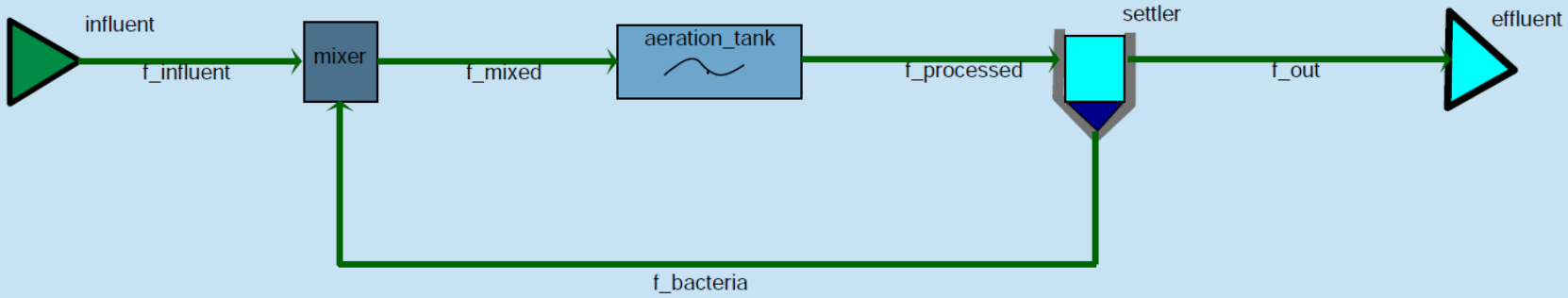
- "operational" semantics
- "denotational" (transformational) semantics

Operational vs. Denotational (Translational) semantics

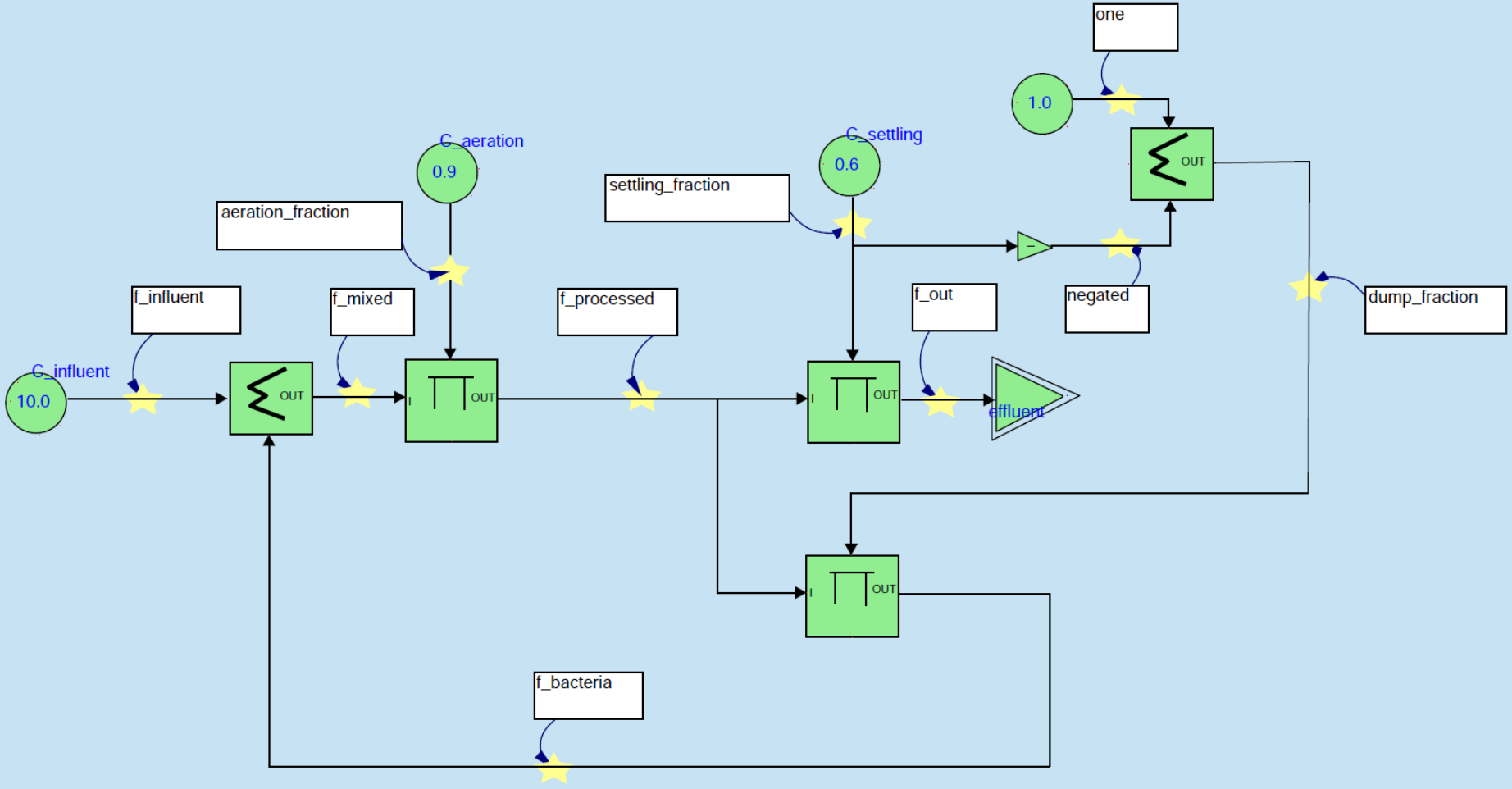
NATO's Sarajevo Waste Water Treatment Plant

www.nato.int/sfor/cimic/env-pro/waterpla.htm

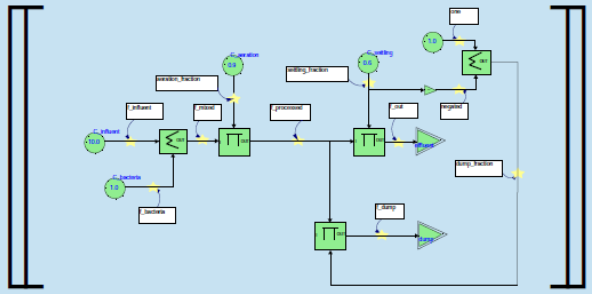
What does this WWTP model mean?



. . . its meaning (steady-state abstraction): Causal Block Diagram (CBD)

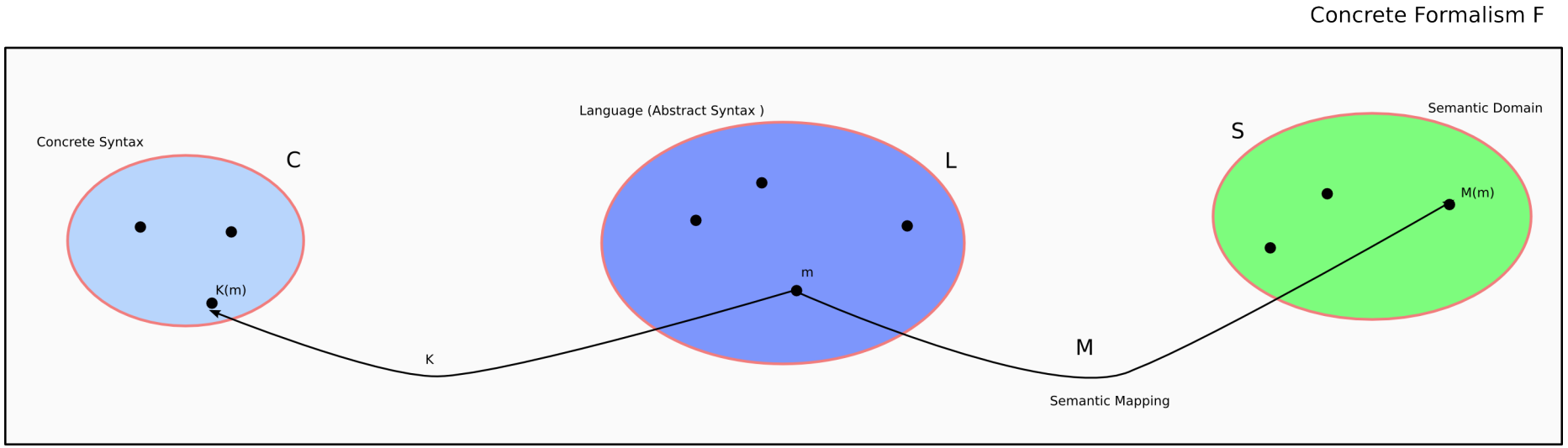


Meaning of the CBD . . . semantic mapping onto algEqns

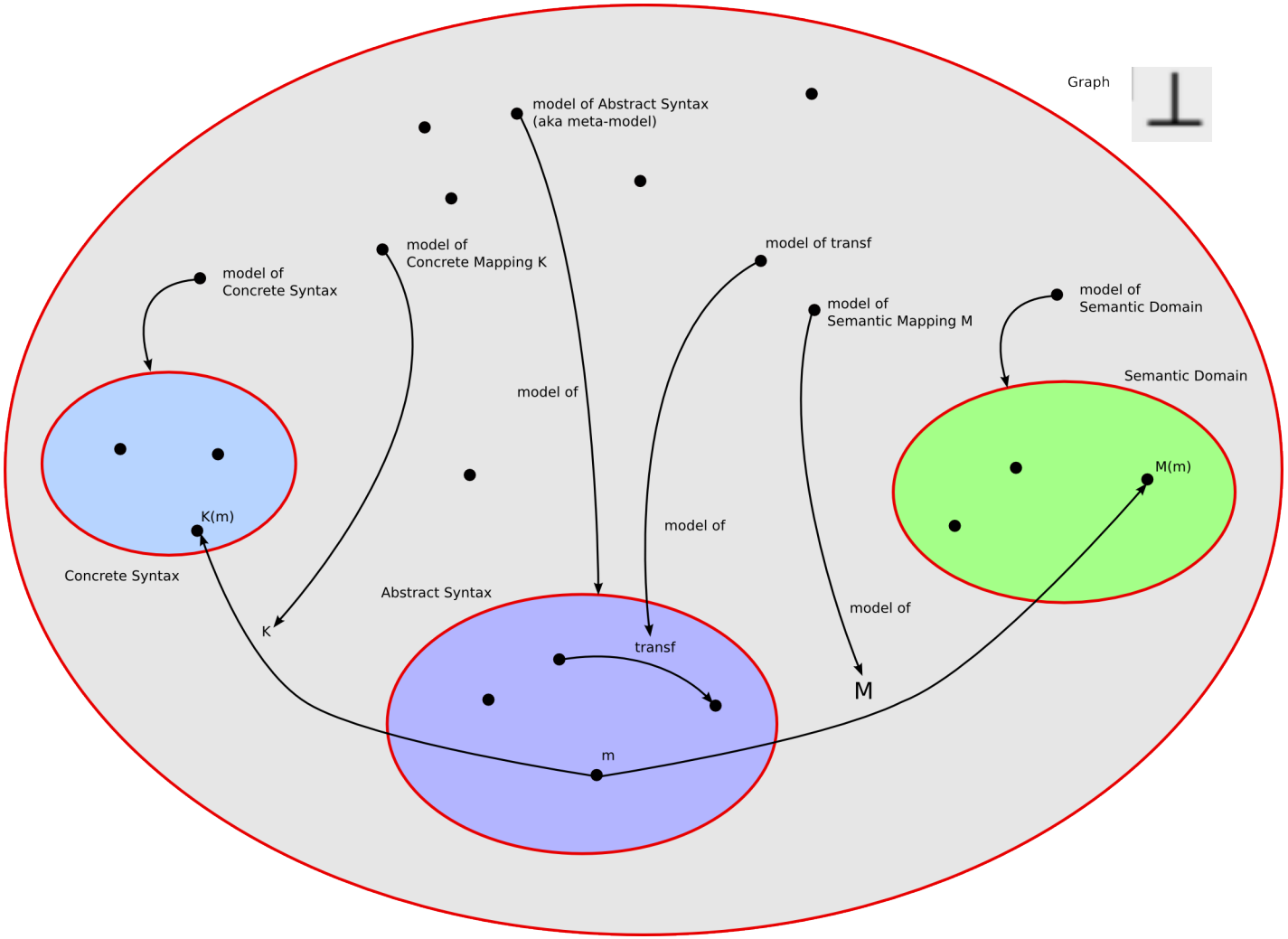


$$
\left\{
\begin{array}{rcl}
f\_influent & = & C\_influent \\
f\_bacteria & = & C\_bacteria \\
f\_mixed & = & f\_influent + f\_bacteria \\
aeration\_fraction & = & C\_aeration \\
f\_processed & = & aeration\_fraction * f\_mixed \\
settling\_fraction & = & C\_settling \\
negated & = & -settling\_fraction \\
one & = & 1 \\
dump\_fraction & = & one + negated \\
f\_dump & = & f\_processed * dump\_fraction \\
f\_out & = & settling\_fraction * f\_processed
\end{array}
\right.
$$

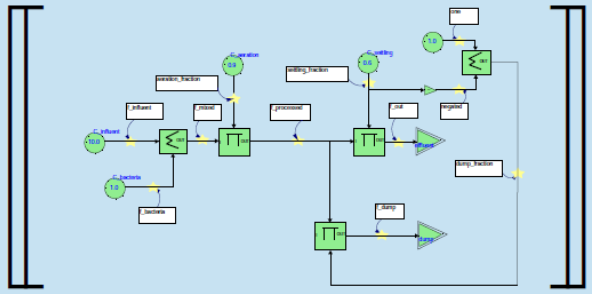
“linguistic” view on Modelling Languages/Formalisms: Syntax and Semantics



Explicit "linguistic" Modelling of Modelling Languages/Formalisms

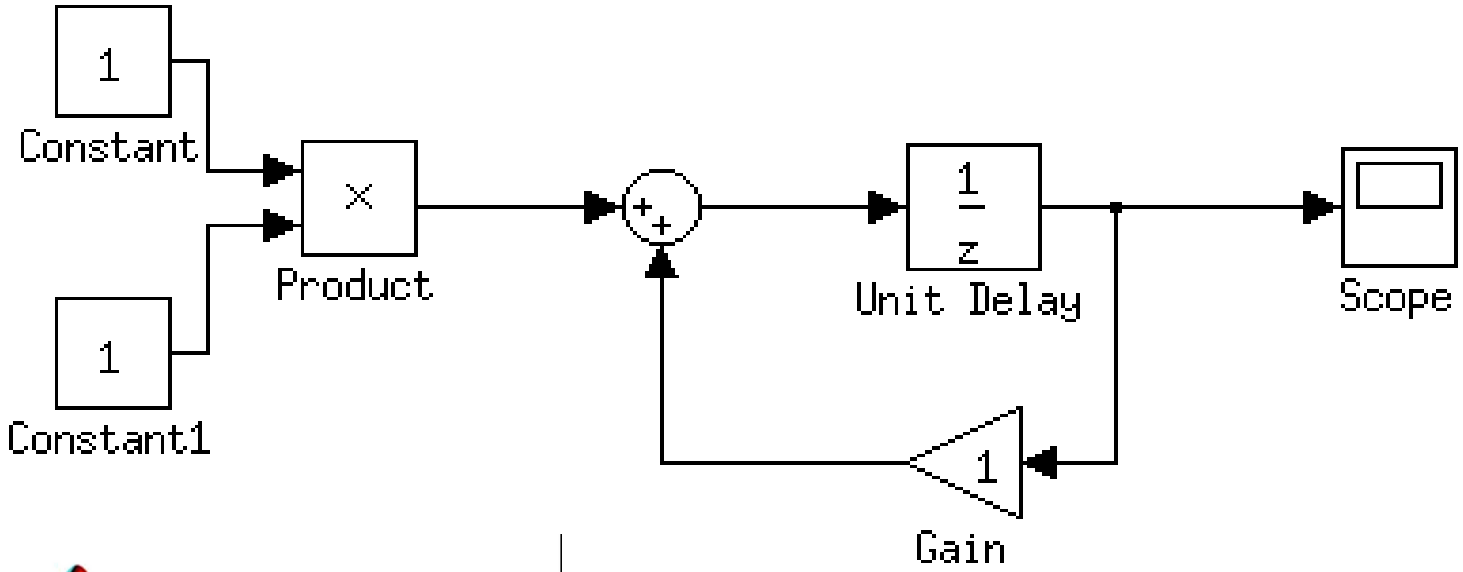


Meaning of the CBD . . . semantic mapping onto algEqns

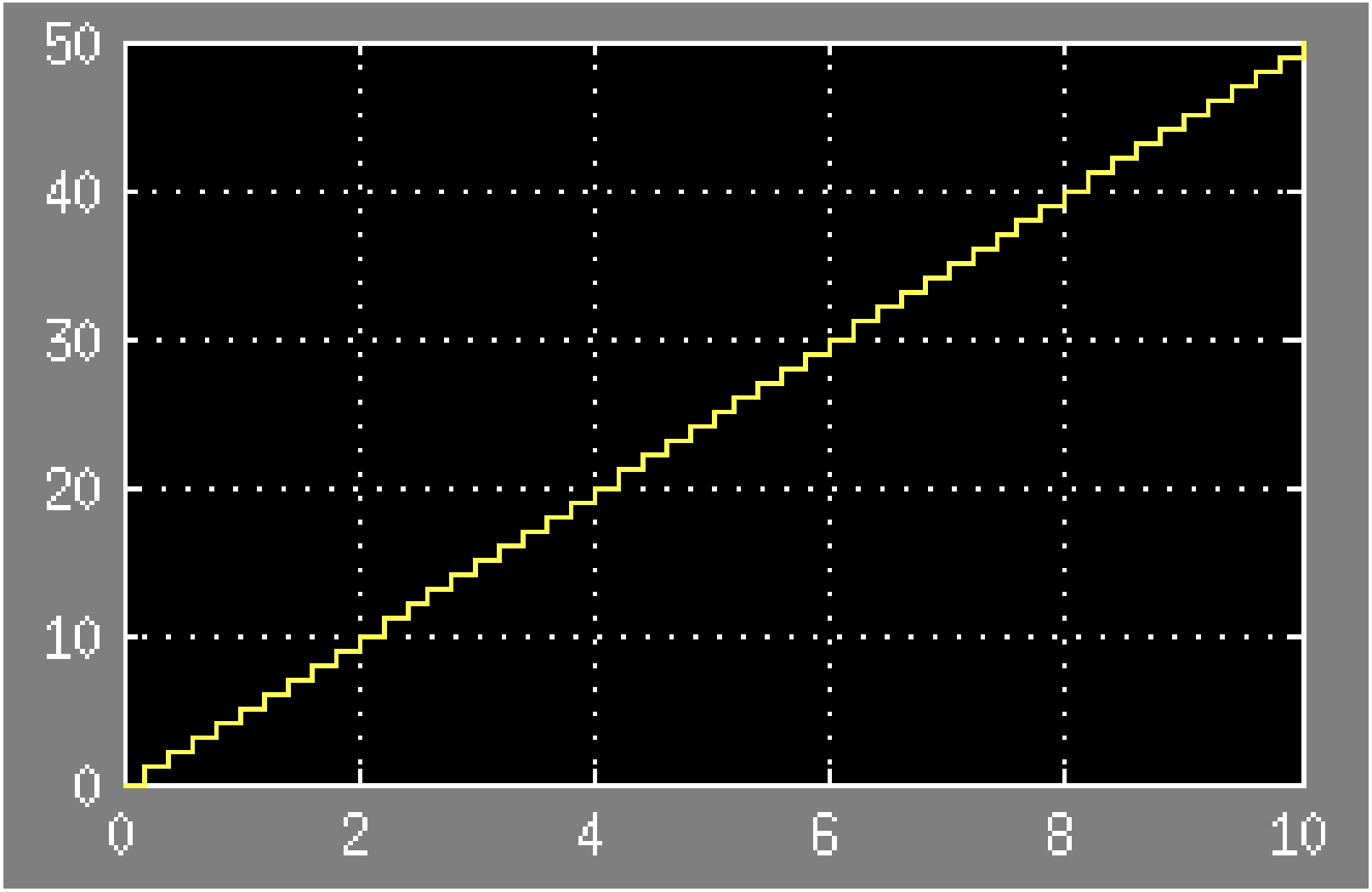


$$
\left\{
\begin{array}{rcl}
f\_influent & = & C\_influent \\
f\_bacteria & = & C\_bacteria \\
f\_mixed & = & f\_influent + f\_bacteria \\
aeration\_fraction & = & C\_aeration \\
f\_processed & = & aeration\_fraction * f\_mixed \\
settling\_fraction & = & C\_settling \\
negated & = & -settling\_fraction \\
one & = & 1 \\
dump\_fraction & = & one + negated \\
f\_dump & = & f\_processed * dump\_fraction \\
f\_out & = & settling\_fraction * f\_processed
\end{array}
\right.
$$

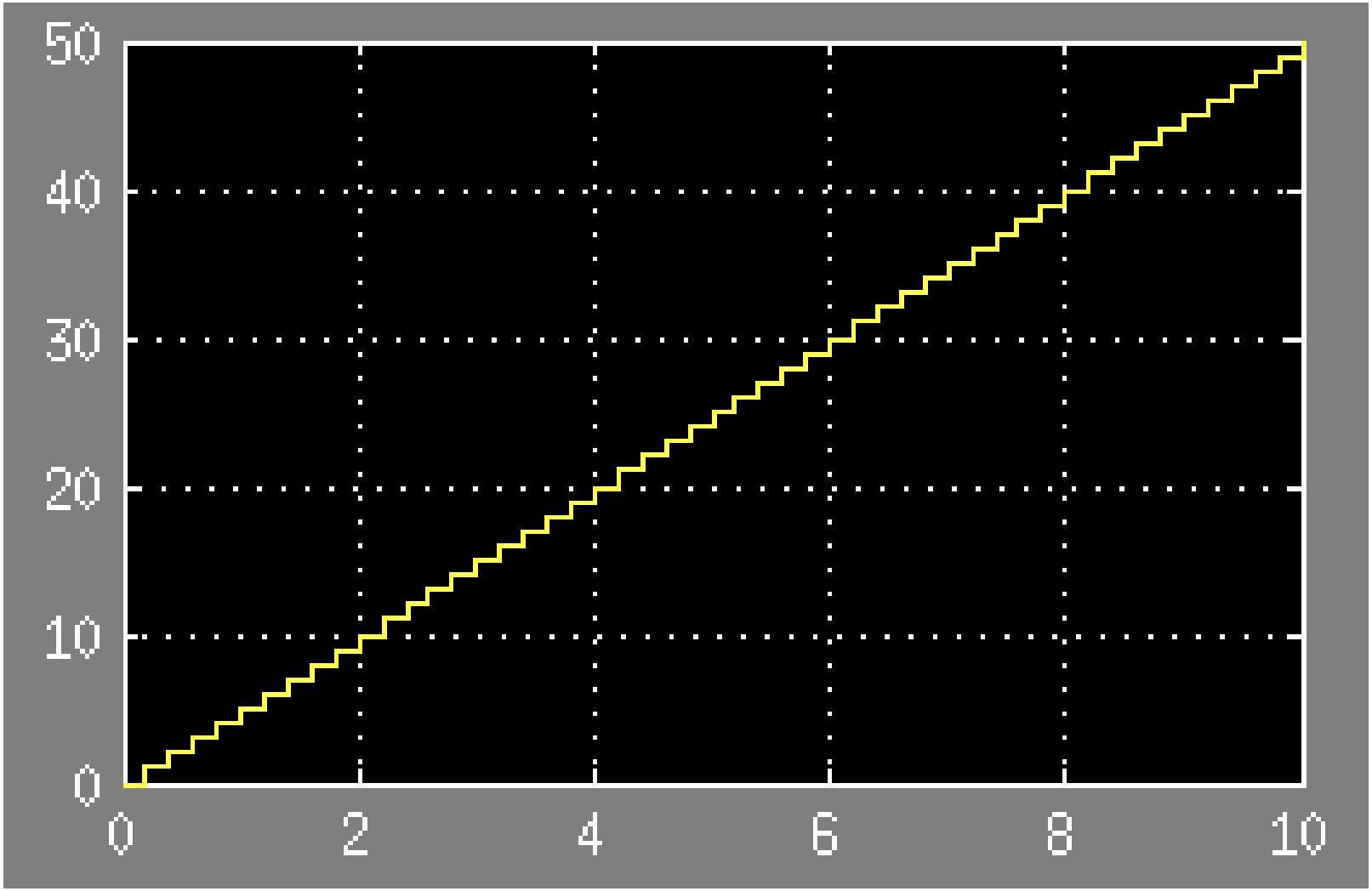
Causal Block Diagrams (syntax)

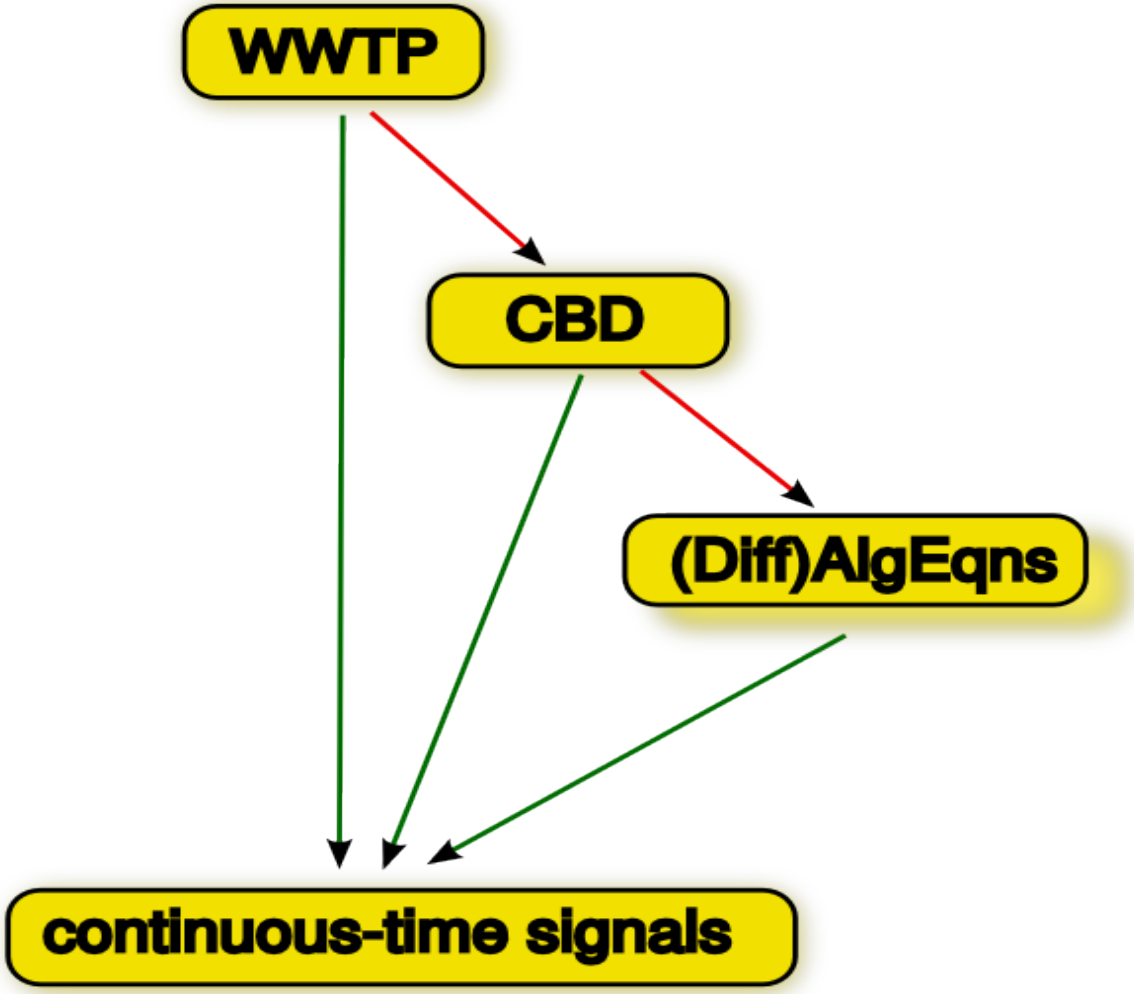


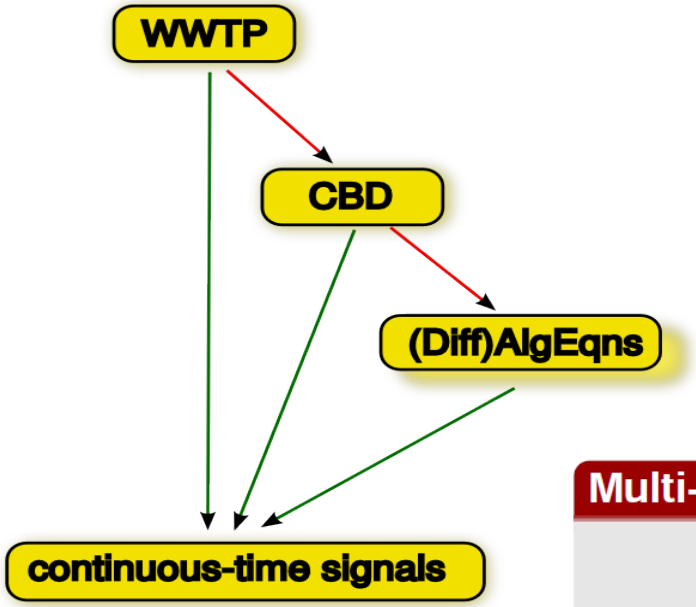
MATLAB®
SIMULINK®


50
40
30
20
10
0
0
2
4
6
8
10

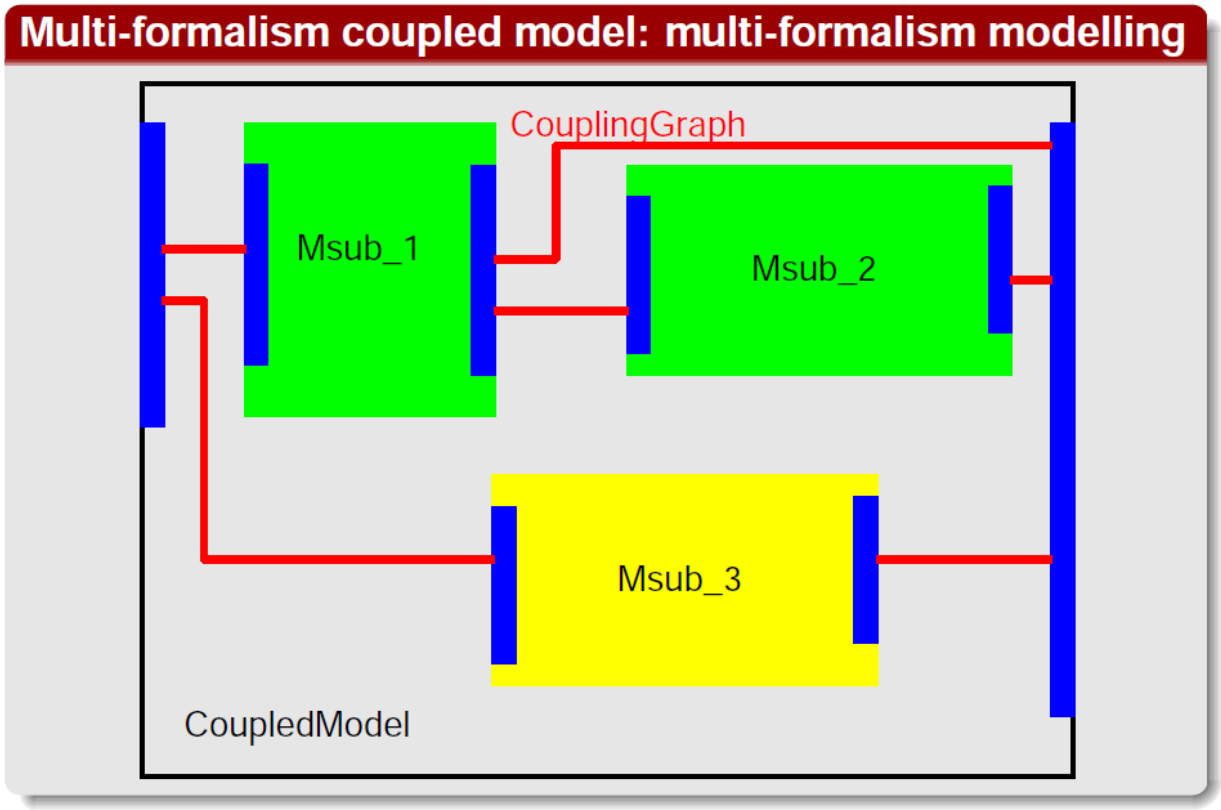
$logicalTime \leftarrow 0$
while not end_condition **do**
    $schedule \leftarrow LOOPDETECT(DEPGRAPH(cbd))$
    for gblock **in** schedule **do**
        $COMPUTE(gblock)$
    end for
    $logicalTime \leftarrow logicalTime + \Delta t$
end while


50
40
30
20
10
0
0
2
4
6
8
10

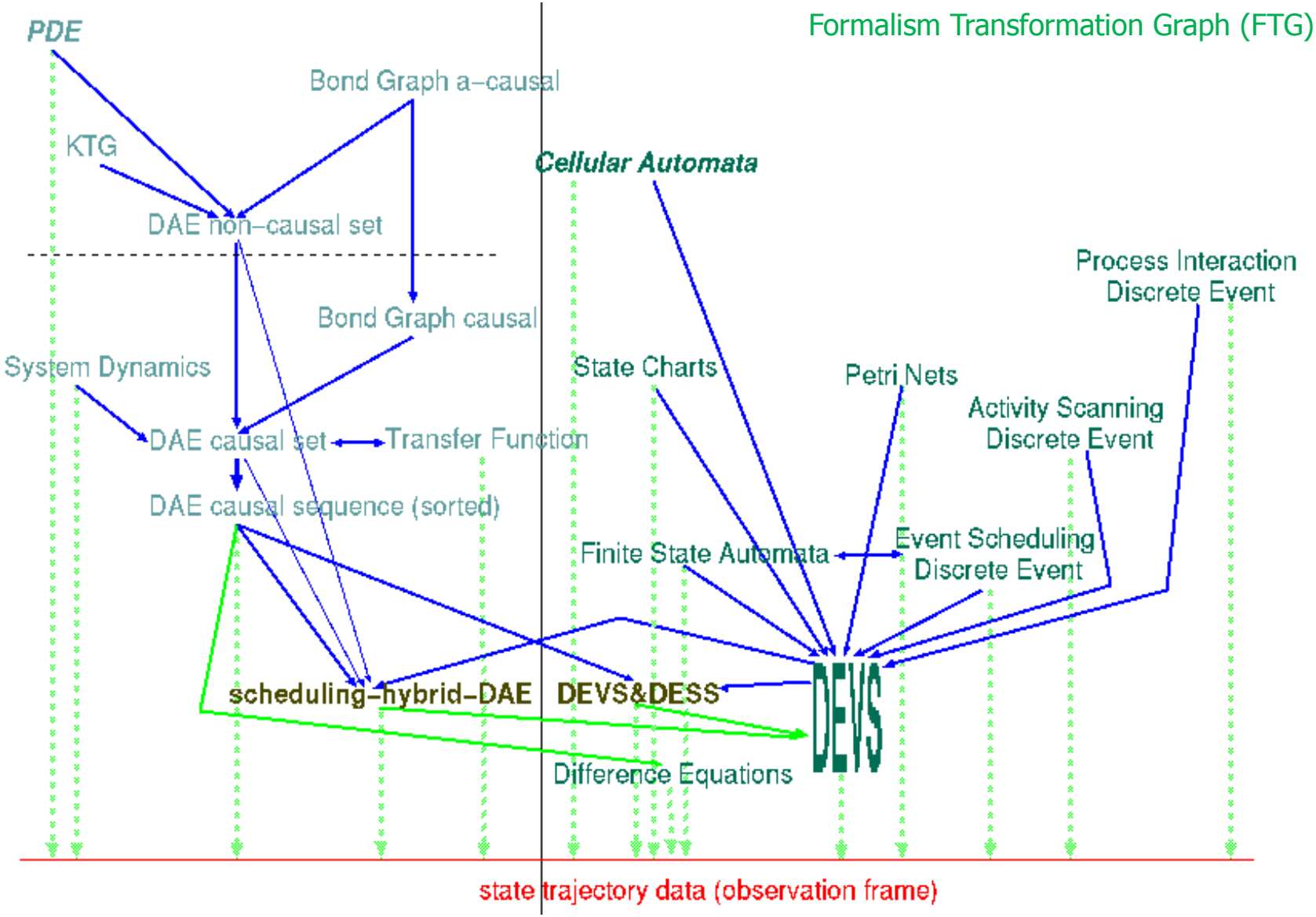

WWTP
CBD
(Diff)AlgEqns
continuous-time signals



Bran Selic: “fragmentation problem”

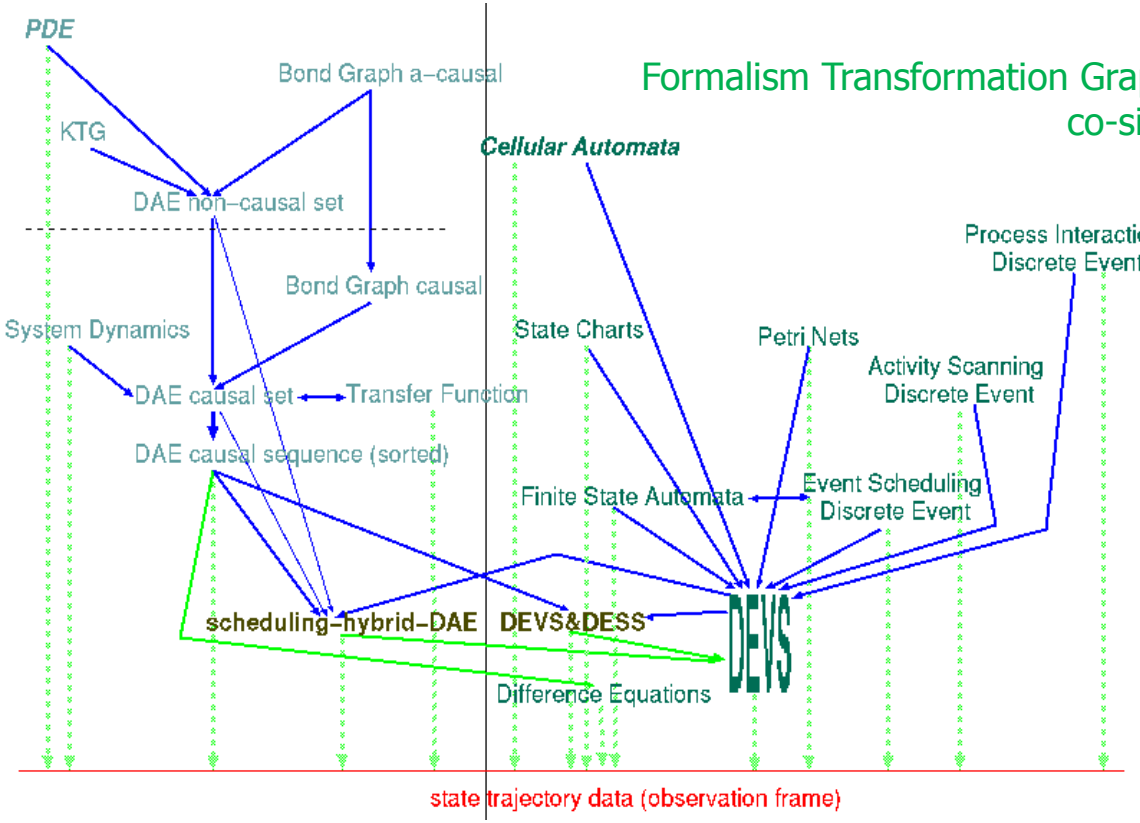


Formalism Transformation Graph (FTG)

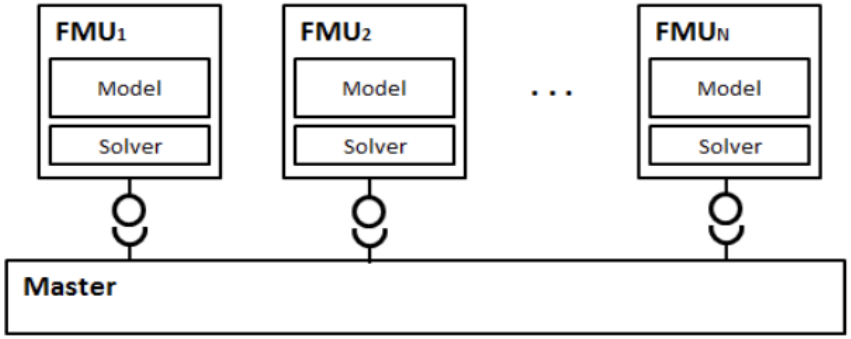


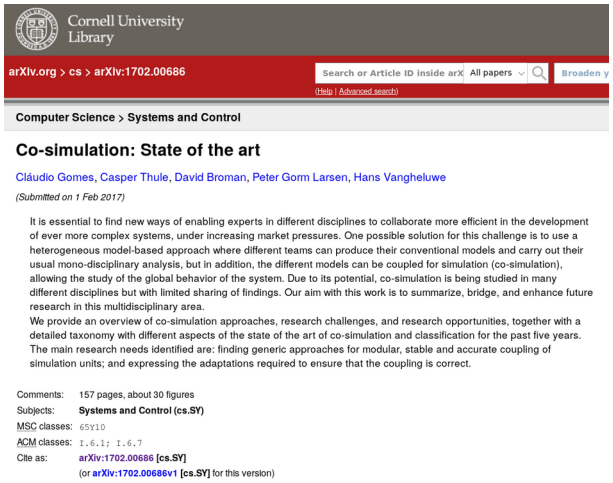
Hans Vangheluwe and Ghislain C. Vansteenkiste. A multi-paradigm modeling and simulation methodology: Formalisms and languages. In European Simulation Symposium (ESS) , pages 168 – 172. Society for Computer Simulation International (SCS), October 1996. Genoa, Italy.

Formalism Transformation Graph (FTG) co-simulation



FUNCTIONAL MOCK-UP INTERFACE




Cornell University Library

arXiv.org > cs > arXiv:1702.00686

Search or Article ID inside arX All papers Broaden y

(Help | Advanced search)

Computer Science > Systems and Control

Co-simulation: State of the art

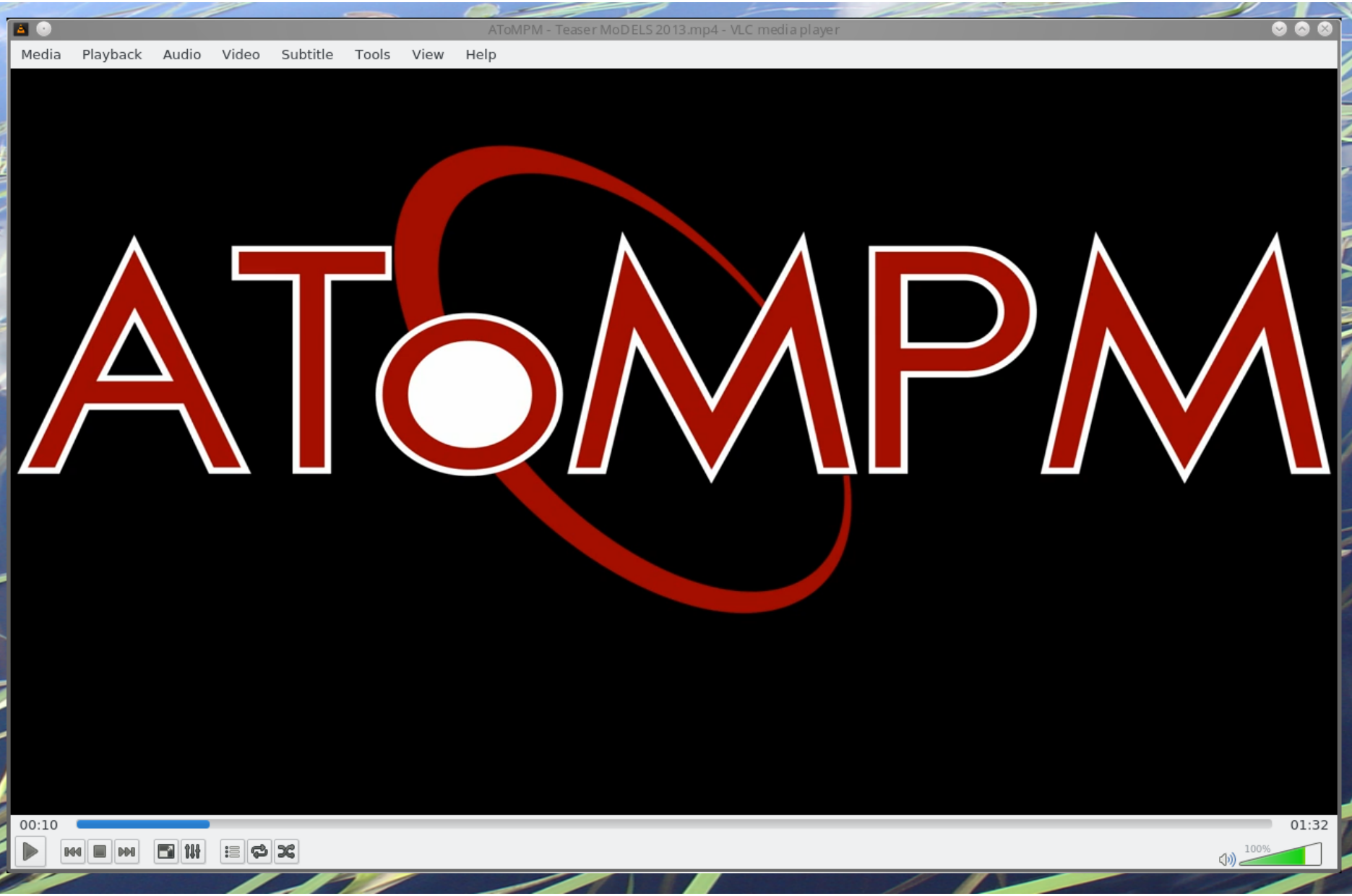
Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe

(Submitted on 1 Feb 2017)

It is essential to find new ways of enabling experts in different disciplines to collaborate more efficient in the development of ever more complex systems, under increasing market pressures. One possible solution for this challenge is to use a heterogeneous model-based approach where different teams can produce their conventional models and carry out their usual mono-disciplinary analysis, but in addition, the different models can be coupled for simulation (co-simulation), allowing the study of the global behavior of the system. Due to its potential, co-simulation is being studied in many different disciplines but with limited sharing of findings. Our aim with this work is to summarize, bridge, and enhance future research in this multidisciplinary area.
We provide an overview of co-simulation approaches, research challenges, and research opportunities, together with a detailed taxonomy with different aspects of the state of the art of co-simulation and classification for the past five years. The main research needs identified are: finding generic approaches for modular, stable and accurate coupling of simulation units; and expressing the adaptations required to ensure that the coupling is correct.

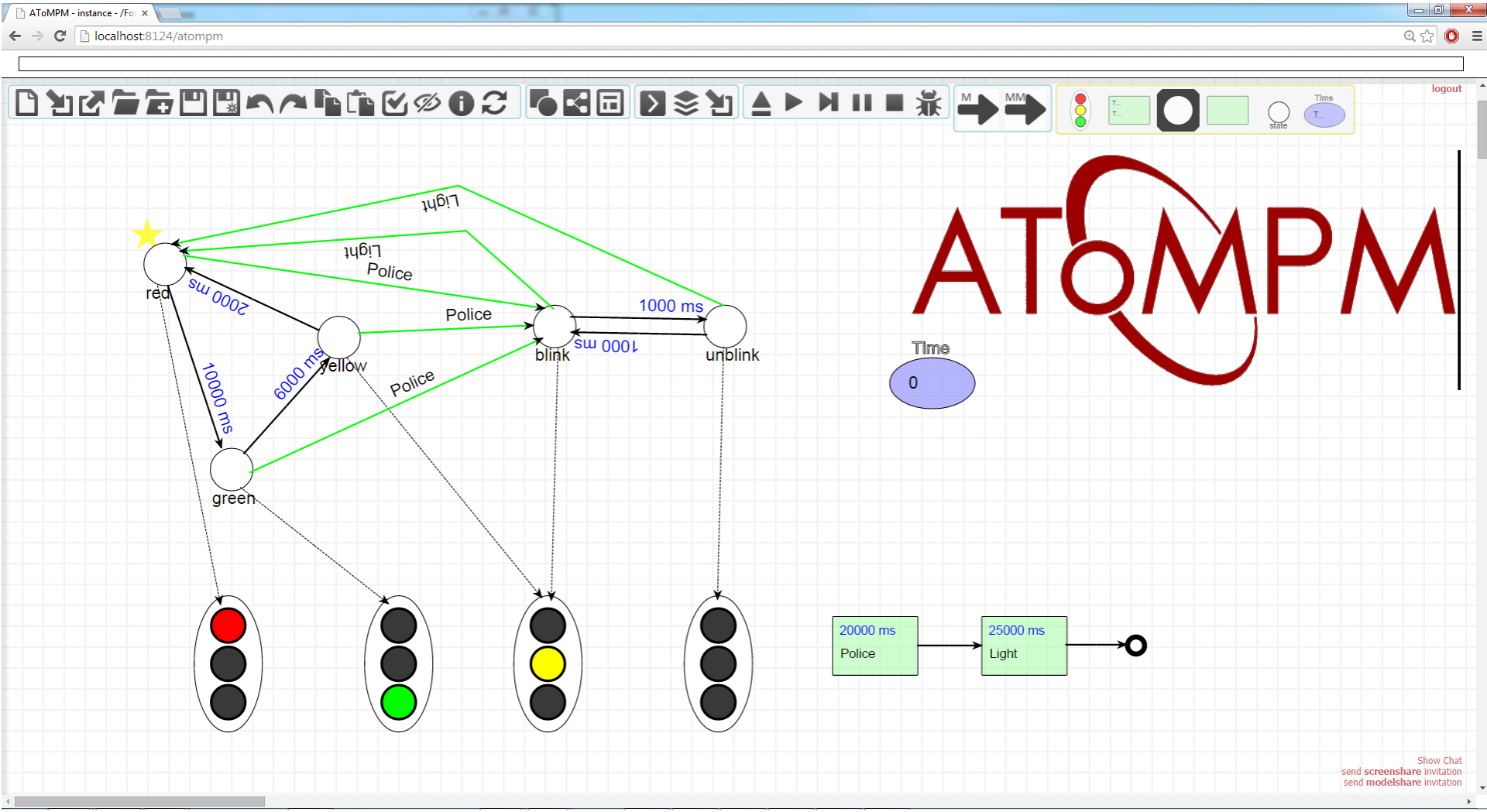
Comments: 157 pages, about 30 figures
Subjects: **Systems and Control (cs.SY)**
MSC classes: 65Y10
ACM classes: I.6.1; I.6.7
Cite as: **arXiv:1702.00686 [cs.SY]**
(or **arXiv:1702.00686v1 [cs.SY]** for this version)

Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe. Co-simulation: State of the art. ArXiv 1702.00686. 2017.



https://www.youtube.com/watch?v=RYtea2BiQ98

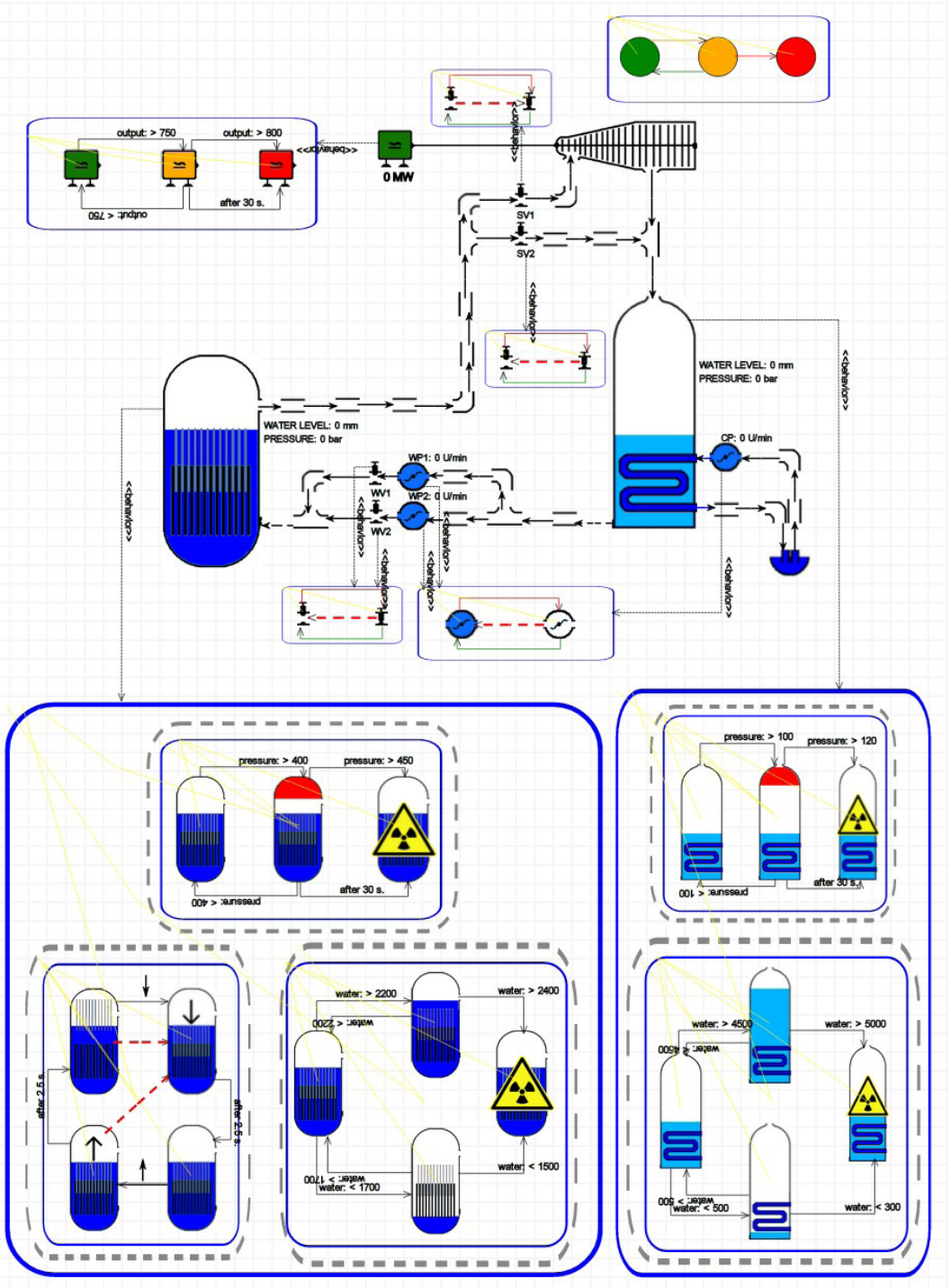
(domain-specific) “linguistic” Modelling Language Engineering



Raphael Mannadiar. A Multi-Paradigm Modelling Approach to the Foundations of Domain-Specific Modelling. PhD thesis, McGill Univ., 2012.

Eugene Syriani, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Huseyin Ergin. AToMPM: A web-based modeling environment. In Proceedings of MODELS'13 Demonstration Session co-located with the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, USA, pages 21- 25, 2013.

https://www.youtube.com/watch?feature=player_detailpage&v=RYtea2BiQ98


output: > 750
output: > 800
output: < 750
after 30 s.
<<behavior>>
0 MW
SV1
SV2
WATER LEVEL: 0 mm
PRESSURE: 0 bar
WP1: 0 l/min
WP2: 0 l/min
WV1
WV2
CP: 0 l/min
pressure: > 400
pressure: > 450
pressure: < 400
after 30 s.
pressure: > 100
pressure: > 120
pressure: < 100
after 30 s.
after 2.5 s.
after 2.5 s.
water: > 2200
water: > 2400
water: < 2200
water: < 1700
water: < 1500
water: > 4500
water: > 5000
water: < 4500
water: < 500
water: < 300
water: < 500

at the most appropriate level(s) of abstraction
using the most appropriate formalism(s)
explicitly modelling workflows

Enabler: (domain-specific) modelling language engineering, including model transformation

Pieter J. Mosterman and Hans Vangheluwe. Computer Automated Multi-Paradigm Modeling: An Introduction. Simulation: Transactions of the Society for Modeling and Simulation International , 80(9):433- 450, September 2004. Special Issue: Grand Challenges for Modeling and Simulation.