Toward understandable co-simulations in model driven engineering

Julien Deantoni

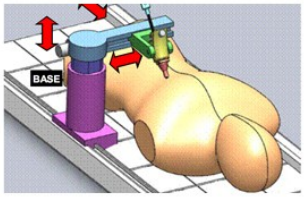
University of Nice, I3S CNRS

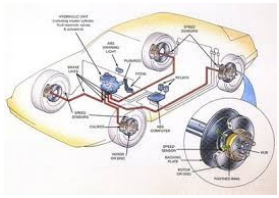
Julien.deantoni@polytech.unice.fr

INRIA Aoste

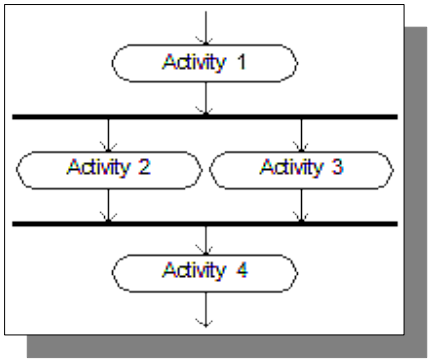
Embedded cyber-physical systems

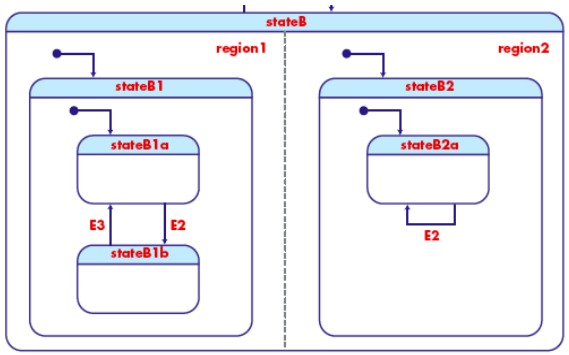






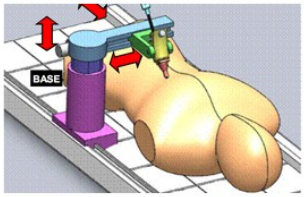

- Application
  - Concurrent application
    - State machine region
    - Data flow process network
    - ...
  - Heterogeneous
    - Control flow
    - Data flow
    - Time driven
    - Event driven
    - …

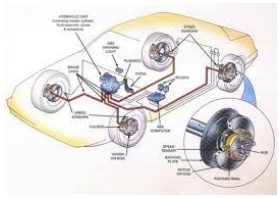




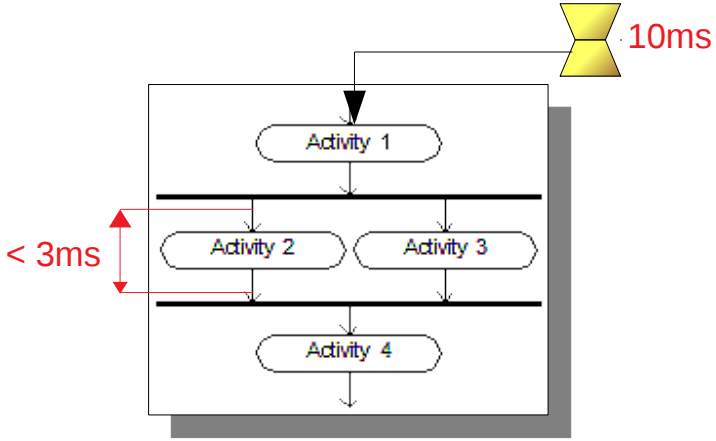
Embedded cyber-physical systems

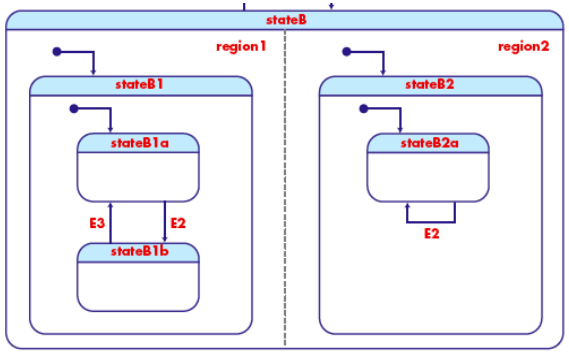






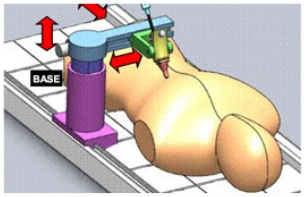

- Application
  - Concurrent
  - Heterogeneous
  - Constraints
    - safety-critical
    - hard real-time
    - Extra functional
      - low power
      - Cost
    - Cyber-Physical related

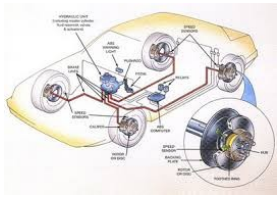




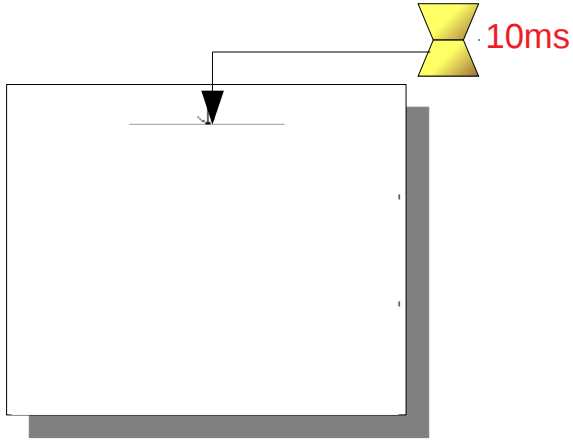
Embedded cyber-physical systems

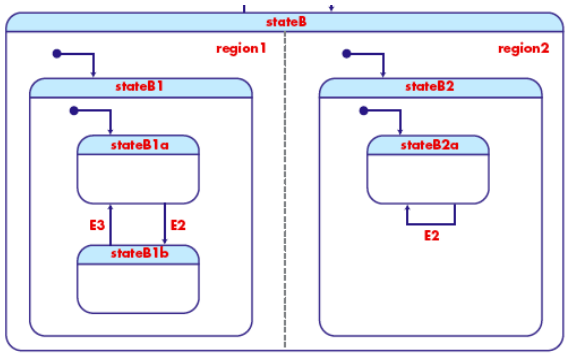






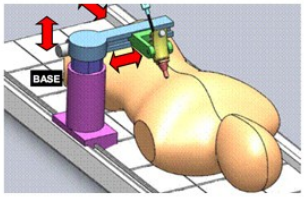

- Application
  - Concurrent
  - Heterogeneous
  - Constraints
    - safety-critical
    - hard real-time
    - Extra functional
      - low power
      - Cost
    - Cyber-Physical related

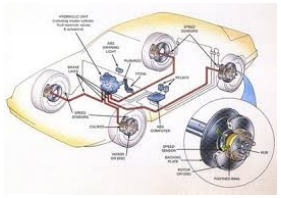




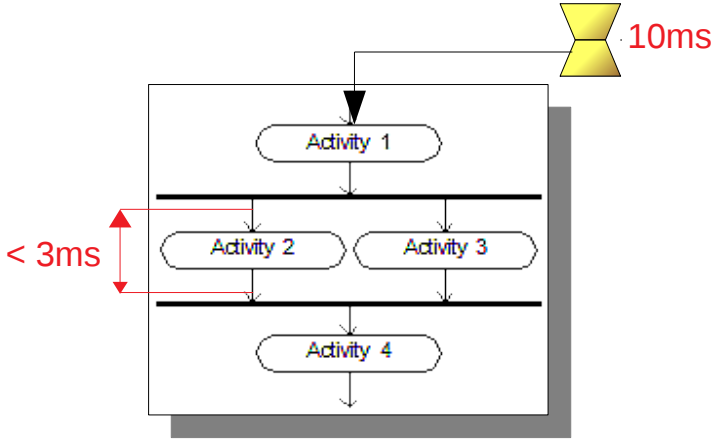
Embedded cyber-physical systems

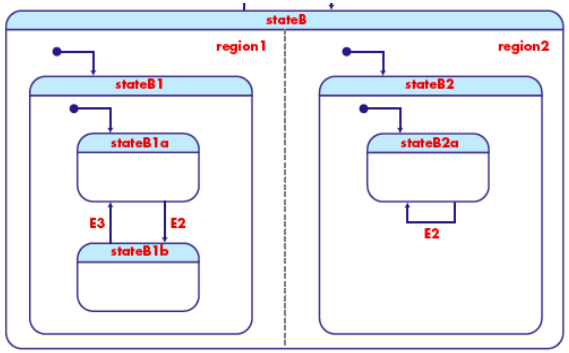







- Application
  - Concurrent
  - Heterogeneous
  - Constraints
    - safety-critical
    - hard real-time
    - Extra functional
      - low power
      - Cost
    - Cyber-Physical related

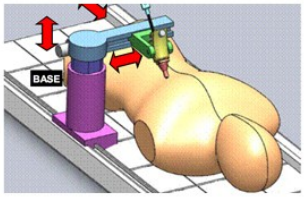


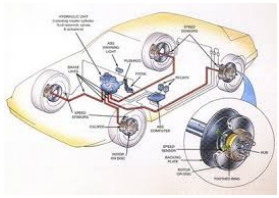


Need for V&V

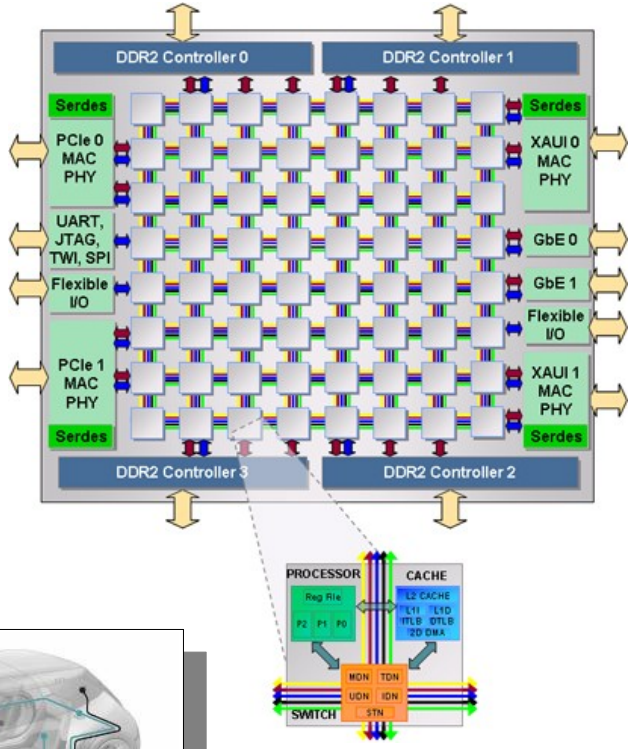
Embedded cyber-physical systems

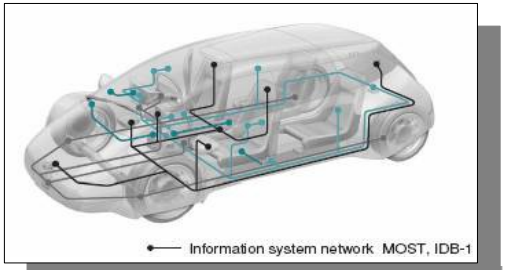






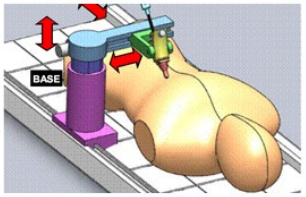

- Application
- Execution Platform
  - Parallel execution
  - Heterogeneous
  - Distributed
  - ...

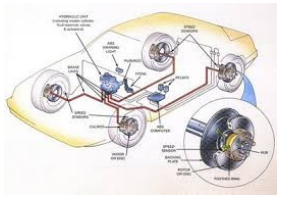




Embedded cyber-physical systems



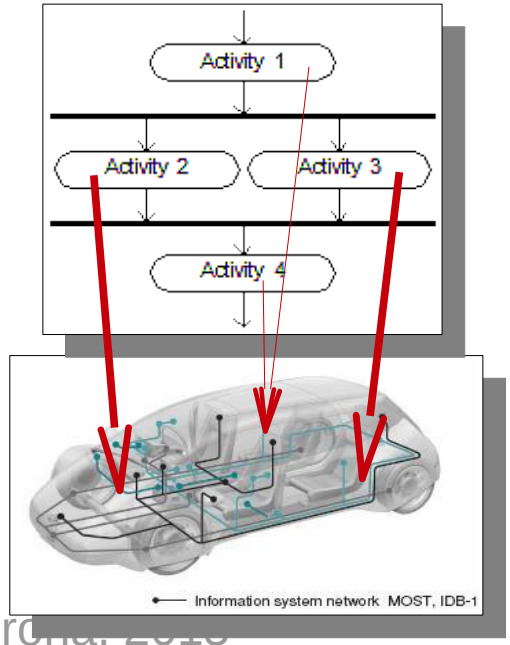





- Application

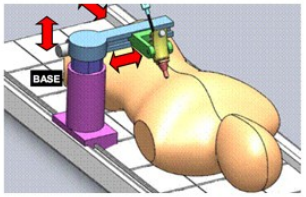
deployment

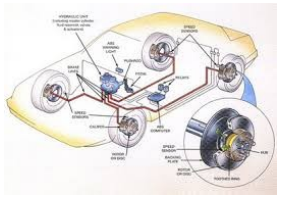
- Execution Platform



Embedded cyber-physical systems





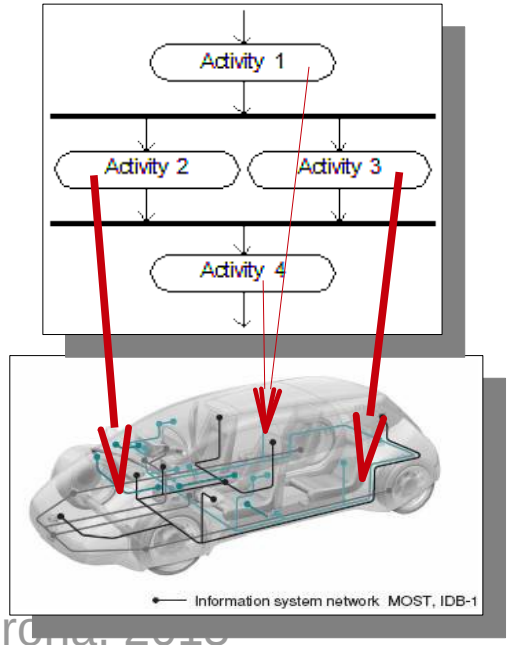



- Application

deployment

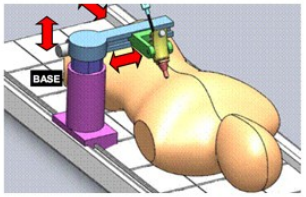
- Execution Platform

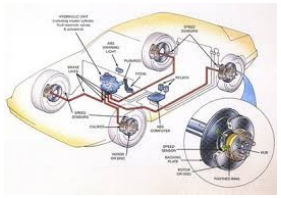
- Constraints
  - safety-critical
  - hard real-time
  - Extra functional
    - Cyber-Physical related



Embedded cyber-physical systems







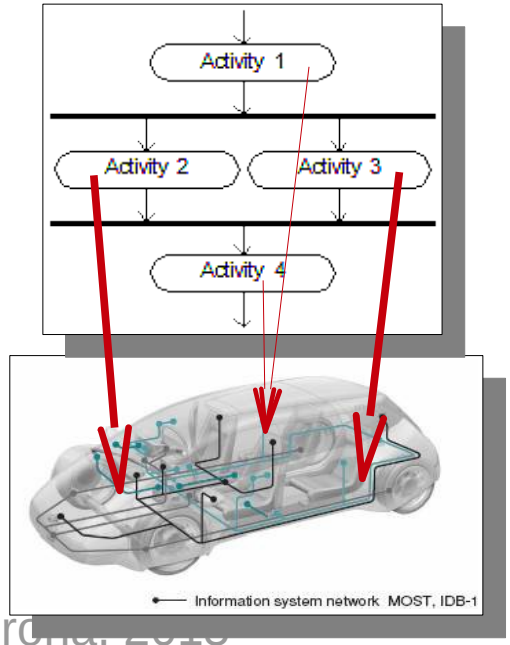

- Application

deployment

- Execution Platform

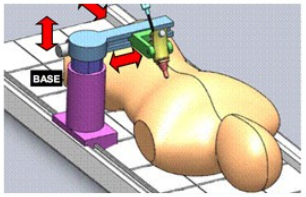
Different languages are used to address these different concerns

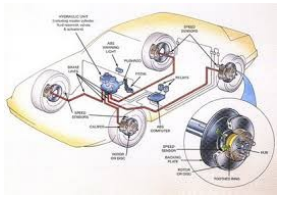
- Constraints
  - safety-critical
  - hard real-time
  - Extra functional
    - Cyber-Physical related



Embedded cyber-physical systems








- Application

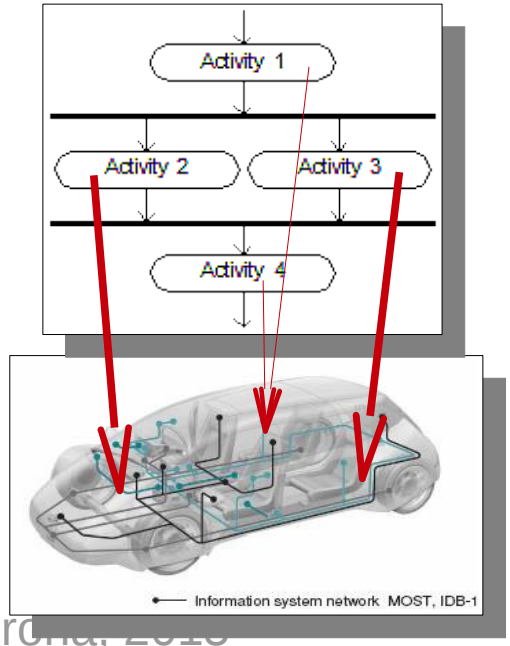
deployment

- Execution Platf
- Constraints
  - safety-critical
  - hard real-time
  - Extra functional
    - Cyber-Physical relat

Different languages are used to address these different concerns

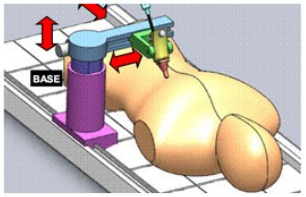
Need for V&V of the global system despite the use of different languages

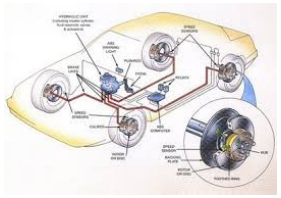
system ok ?



Embedded cyber-physical systems








- Application

deployment

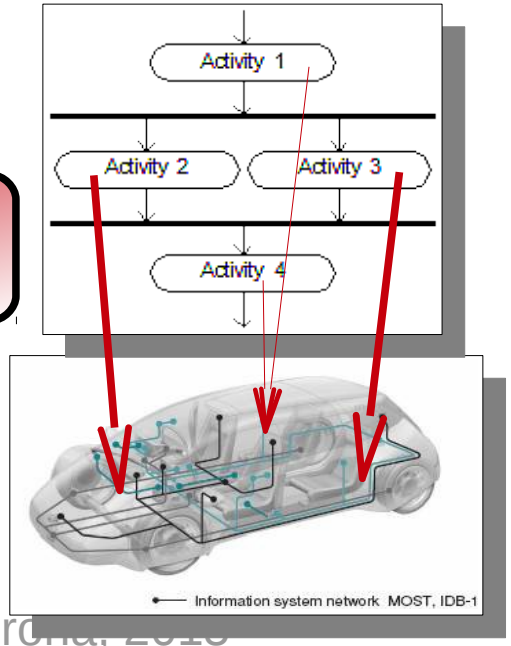
- Execution Platf
- Constraints
  - safety-critical
  - hard real-time
  - Extra functional
    - Cyber-Physical relat

Different languages are used to address these different concerns

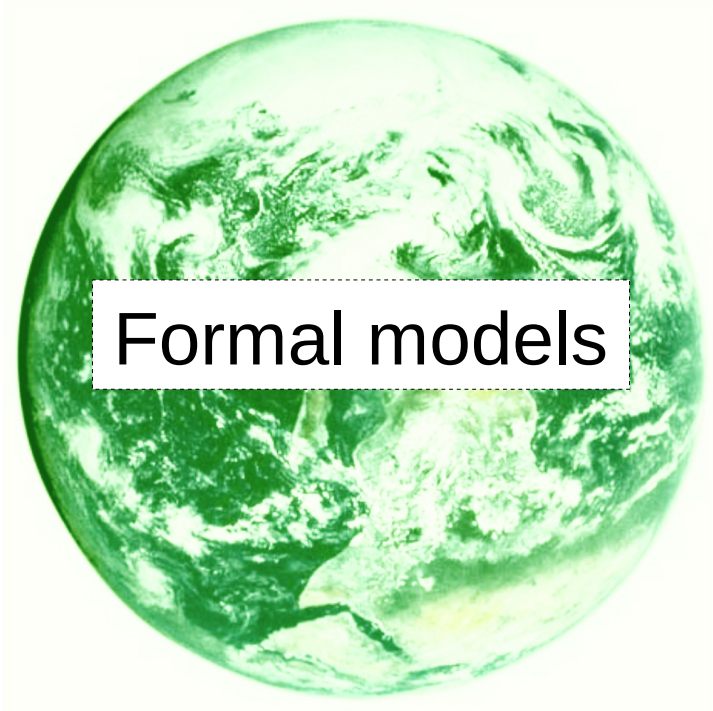
Need for V&V of the global system despite the use of different languages

...all along the development cycle

system ok ?



How to deal with that ?



Model-Driven Engineering

- **Pros**
  - Design in the large or in the small
  - Domain Specific Languages (For a specific goal, the adequate concepts are reified)
  - **Multi-view modeling**
  - Formal definition of concepts and relations

Model-Driven Engineering

- **Pros**
  - Design in the large or in the small
  - Domain Specific Languages (For a specific goal, the adequate concepts are reified)
  - Multi-view modeling
  - Formal definition of concepts and relations

⇒ **Tools take benefits of this:**

- **automatic generation of advanced editors, static checkers, … → automatic reasoning**

Model-Driven Engineering

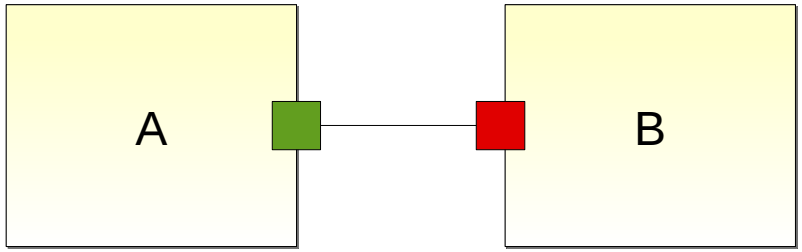
- **Pros**
  - **Formal definition of concepts and relations**

  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations

Model-Driven Engineering

- **Pros**
  - **Formal definition of concepts and relations**

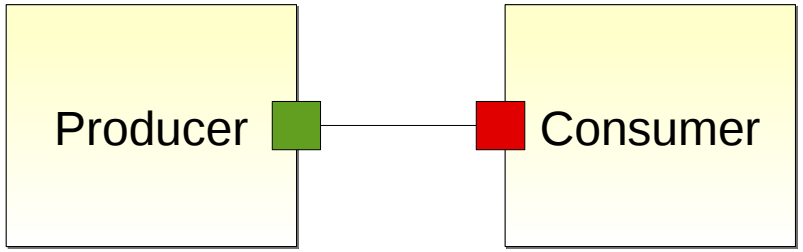
  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations



Model-Driven Engineering

- **Pros**
  - **Formal definition of concepts and relations**

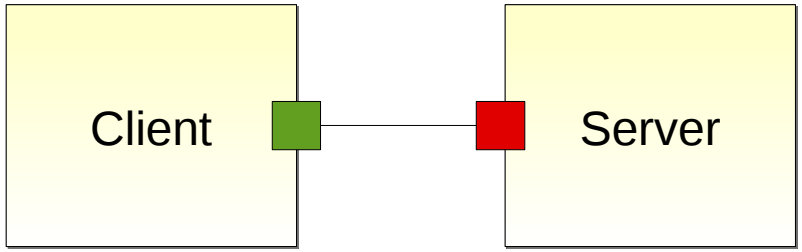
  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations



Model-Driven Engineering

- **Pros**
  - **Formal definition of concepts and relations**

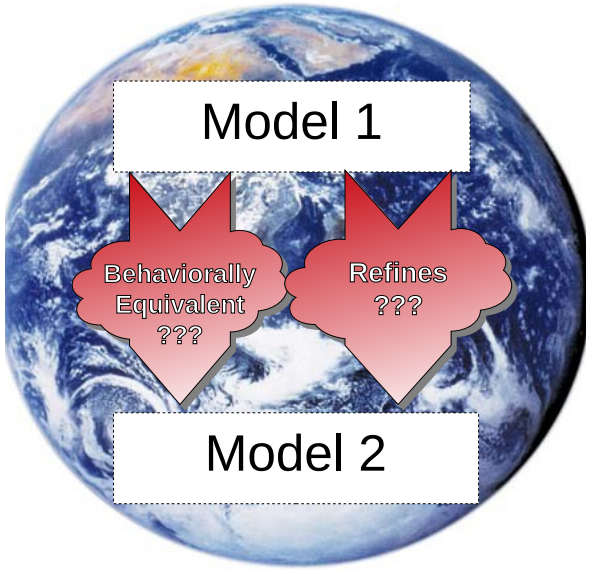
  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations



Model-Driven Engineering

- **Pros**
  - **Formal definition of concepts and relations**

  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations
  - Lack of precise explicit behavioral semantics

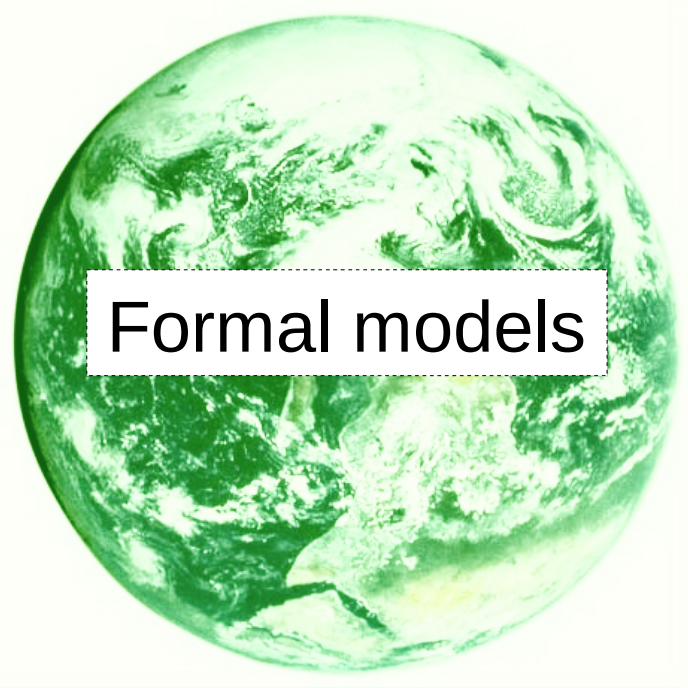


Model-Driven Engineering

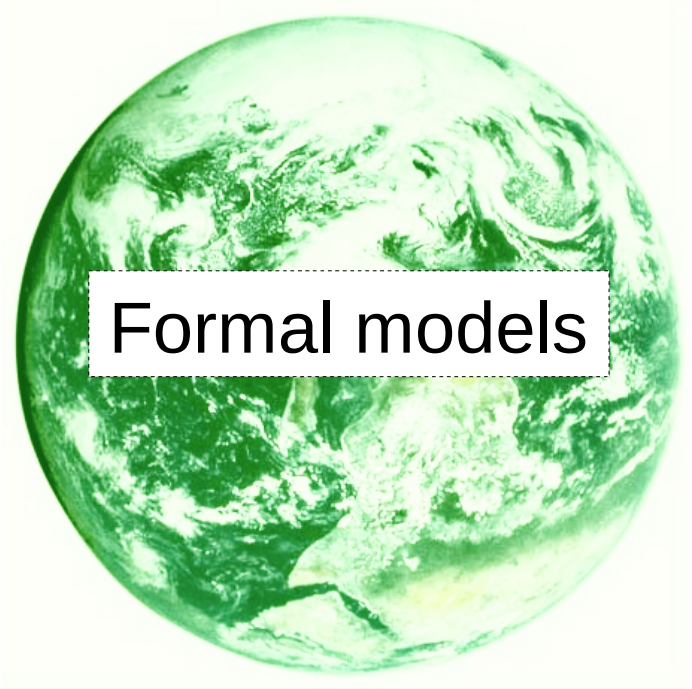
- **Pros**
  - **Formal definition of concepts and relations**

  ⇒ **Tools take benefits of this:**
    - **automatic generation of advanced editors, static checkers, … → automatic reasoning**
- **Cons**
  - Discrepancies in interpretations
  - Lack of precise behavioral semantics

  ⇒ **Tools suffer from this**
    - **No automatic generation of simulators, debuggers, animators, explorer, etc.**

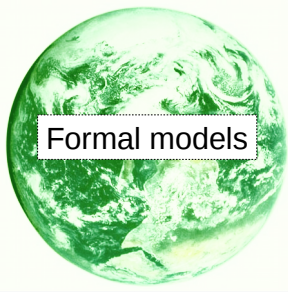


Models of Computation and Communications (MoCCs)



Models of Concurrency and Communications (MoCCs)

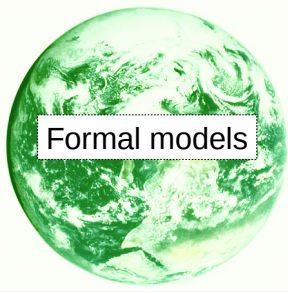
Formal Models (MoCC)



- **Pros**
  - Mathematical semantics (i.e. no ambiguity)

⇒ **Tools benefit from that !**

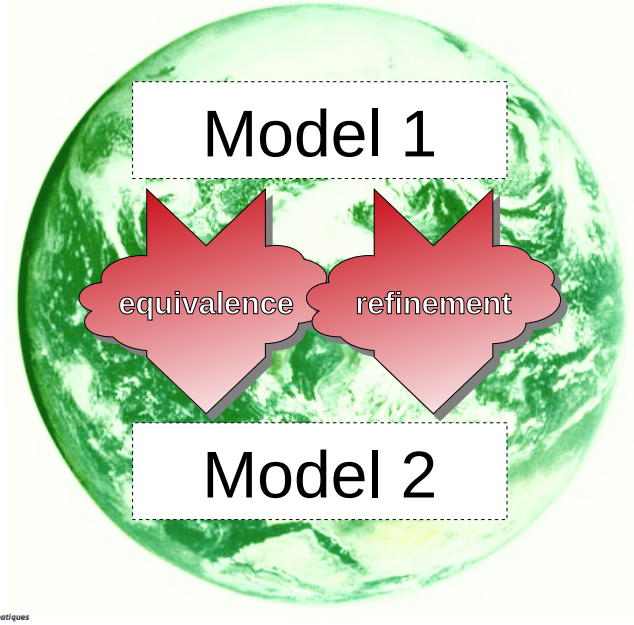
Formal Models (MoCC)



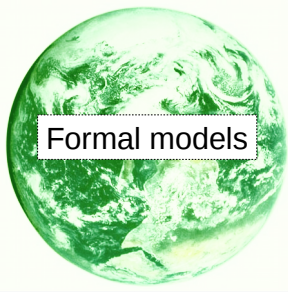
- **Pros**
  - Mathematical semantics (i.e. no ambiguity)

➡ **Tools benefit from that !**

- Powerful analysis and algorithmic methods
- Optimization / verification
- Guaranteed equivalence between code and model
- Basis for well-founded transformations



Formal Models (MoCC)

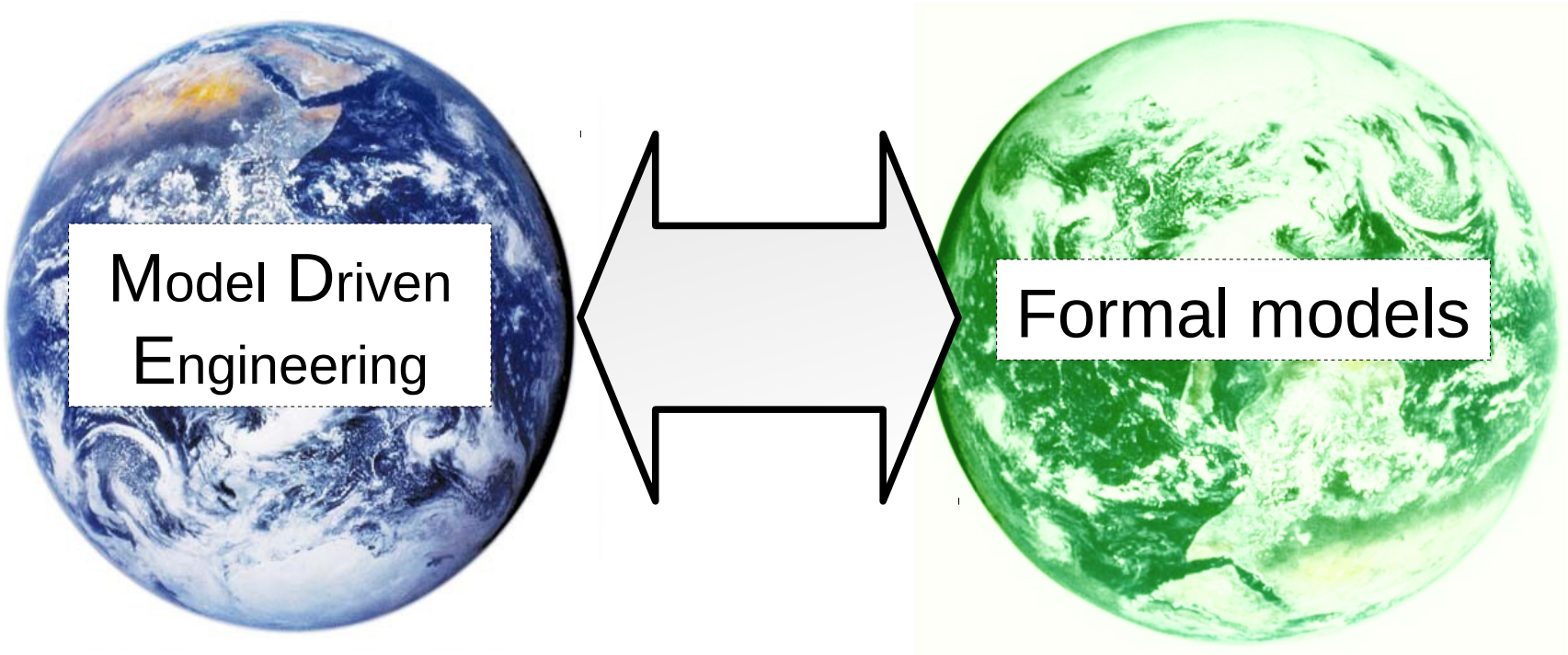


- **Pros**
  - Mathematical semantics (i.e. no ambiguity)

  ⇒ **Tools benefit from that !** Powerful analysis and algorithmic methods, Optimization / verification, Guaranteed equivalence between code and model, Basis for well-founded transformations

- **Cons**
  - **Distance from current mainstream engineering practice**
    - **Exotic formalisms for designers**
    - **Possibly far from a designer domain**
    - **Not necessarily adapted for all the models in a system**

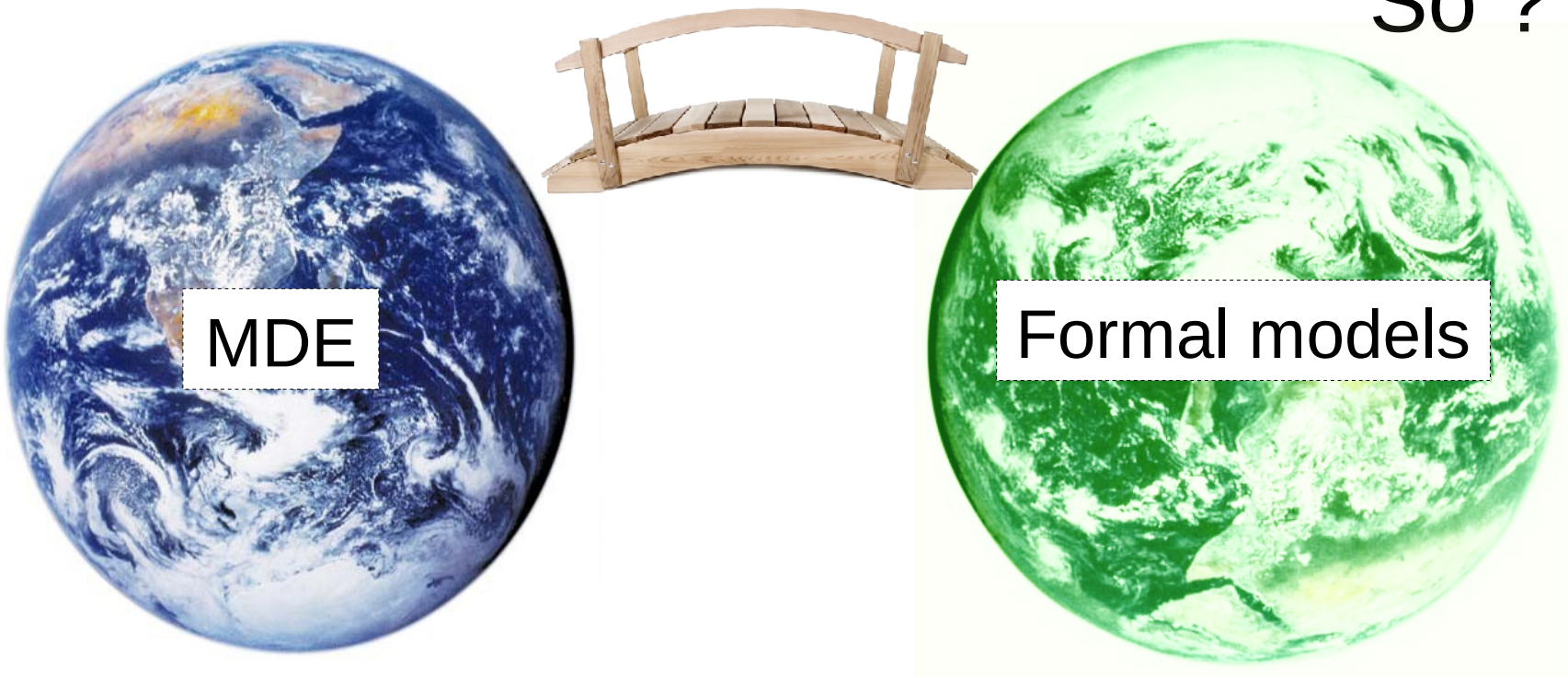
Where are we going ?



Tools, standards, engineering world
→ effective environment for executable DSMLs

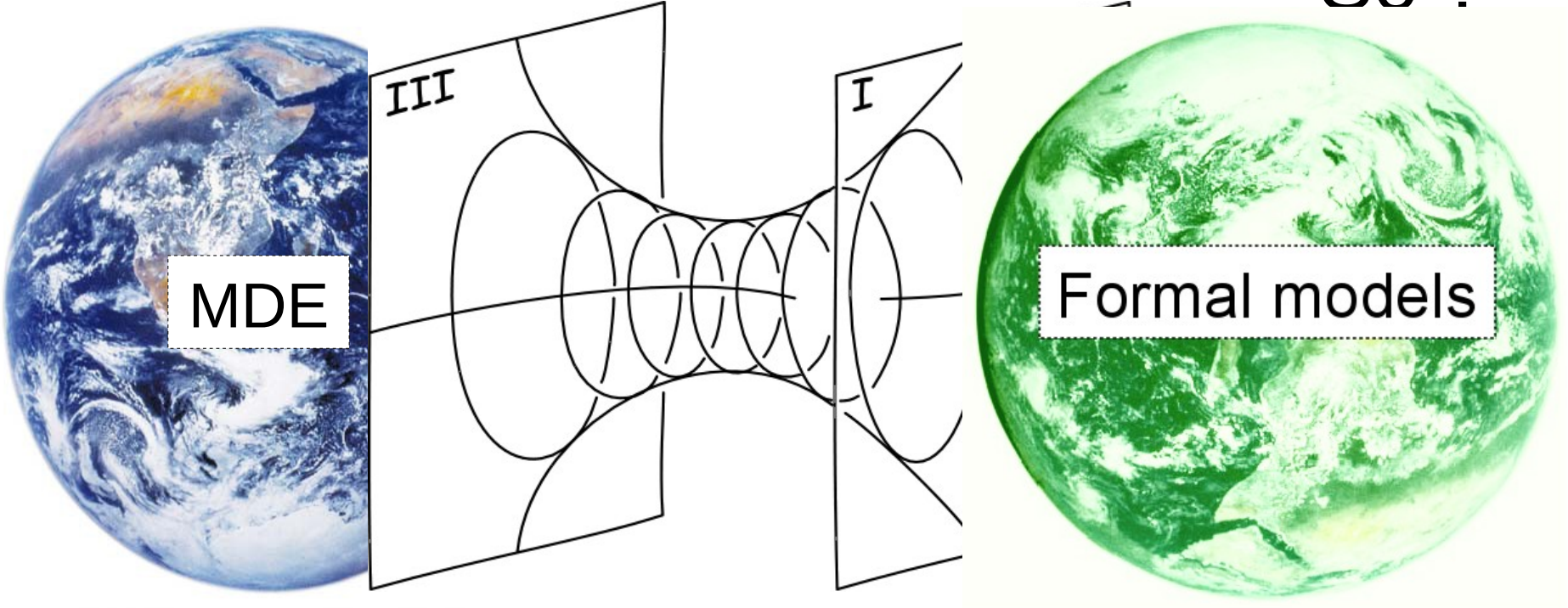
Verification / validation
→ effective mathematical mean to address a problem

So ?



Transformations are often used from a world to another one...

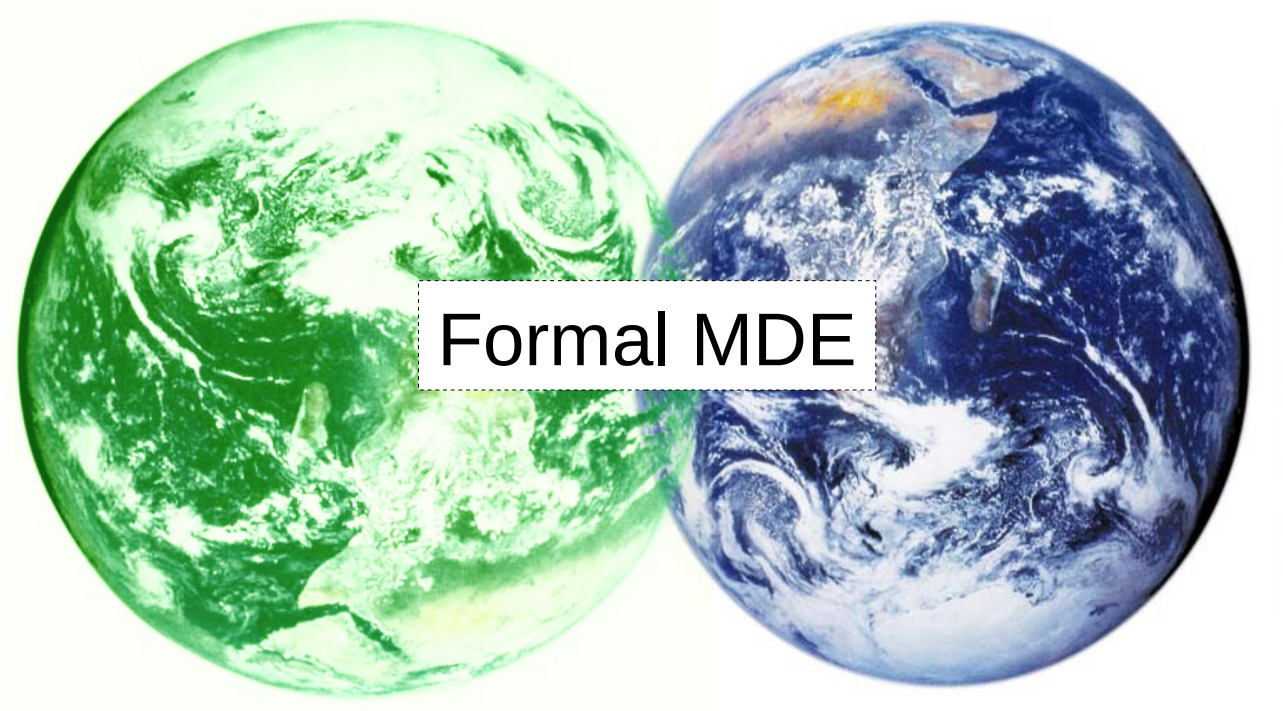
So ?



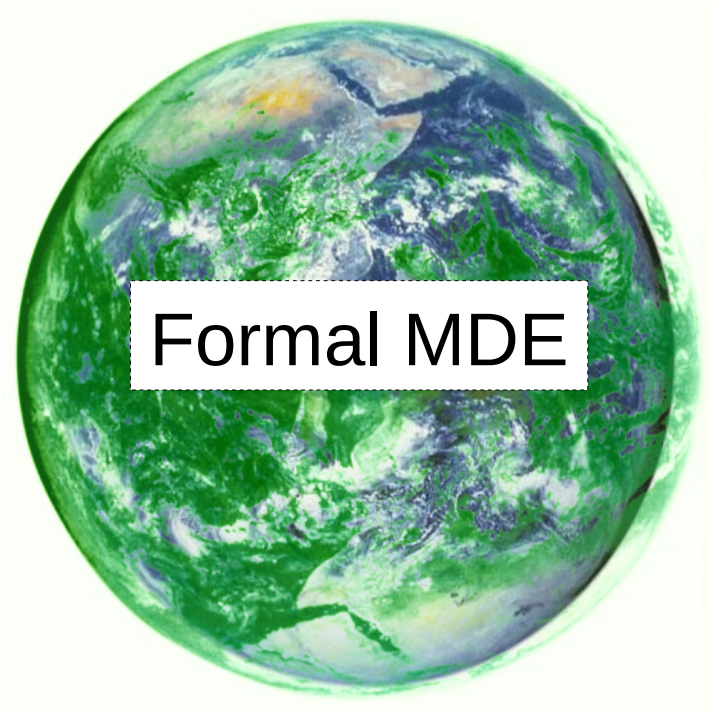
Transformations can distort models due to the abstraction gap between a DSL and a formal model

- Difficulty to understand the resulting behavioral semantics given by both the transformation and the semantics of the targeted language
- No direct way to play with the behavioral semantics

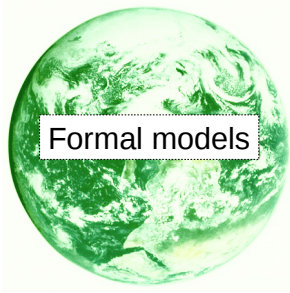
The finality ?



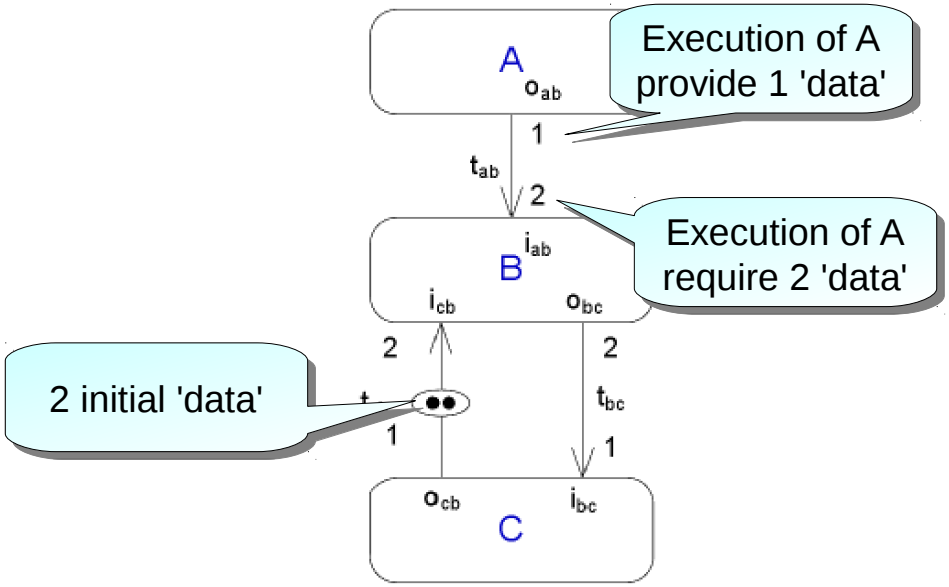
The finality !



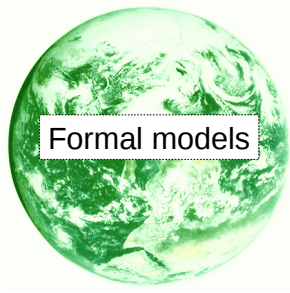
Formal Models (MoCC)



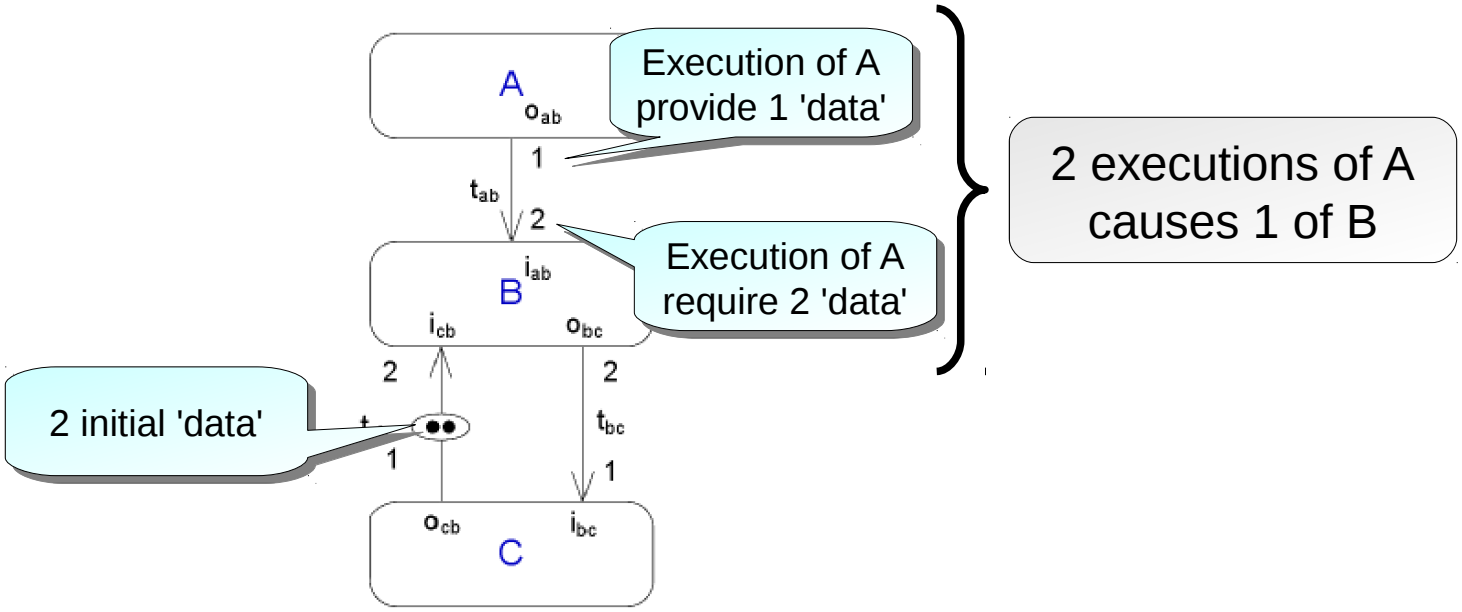
Synchronous Data Flow



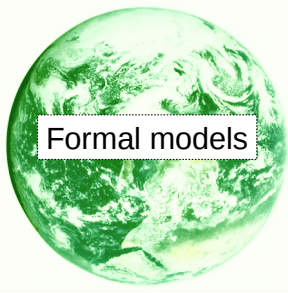
Formal Models (MoCC)



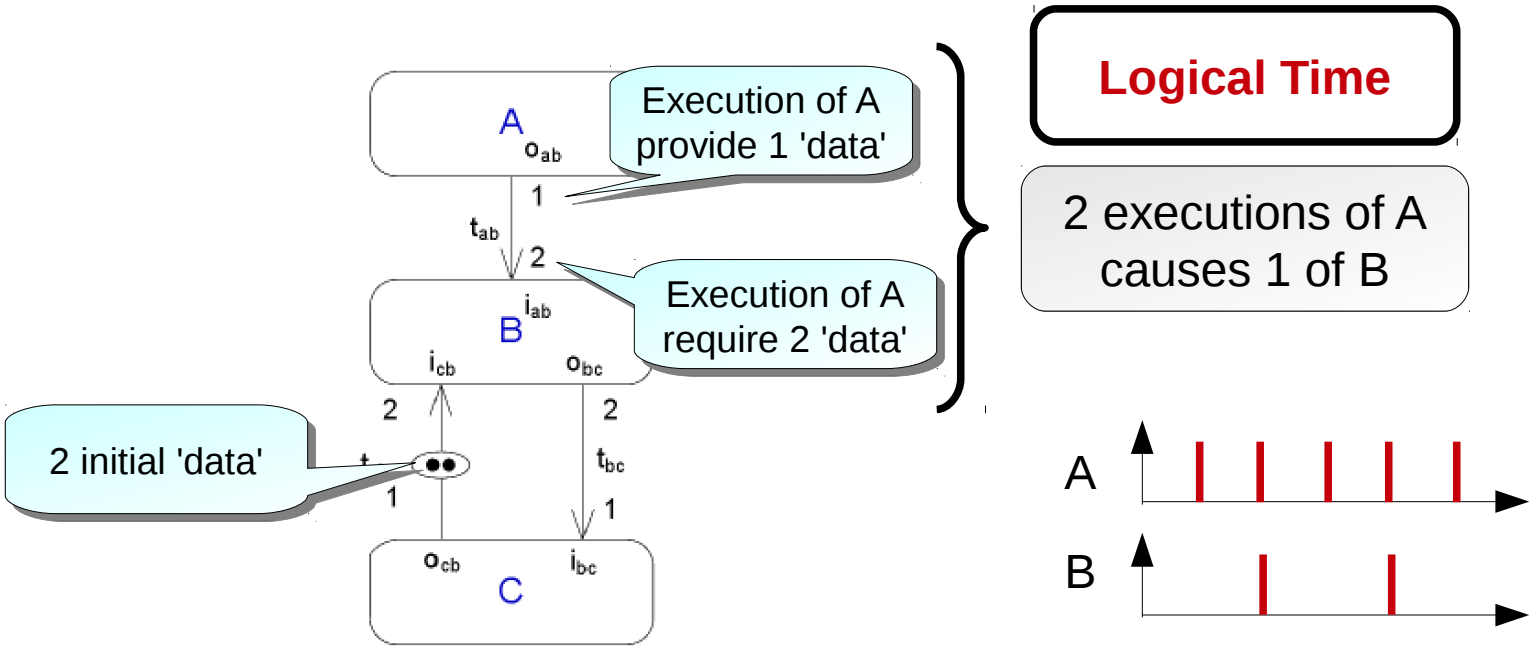
Synchronous Data Flow



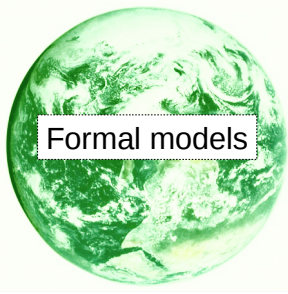
Formal Models (MoCC)



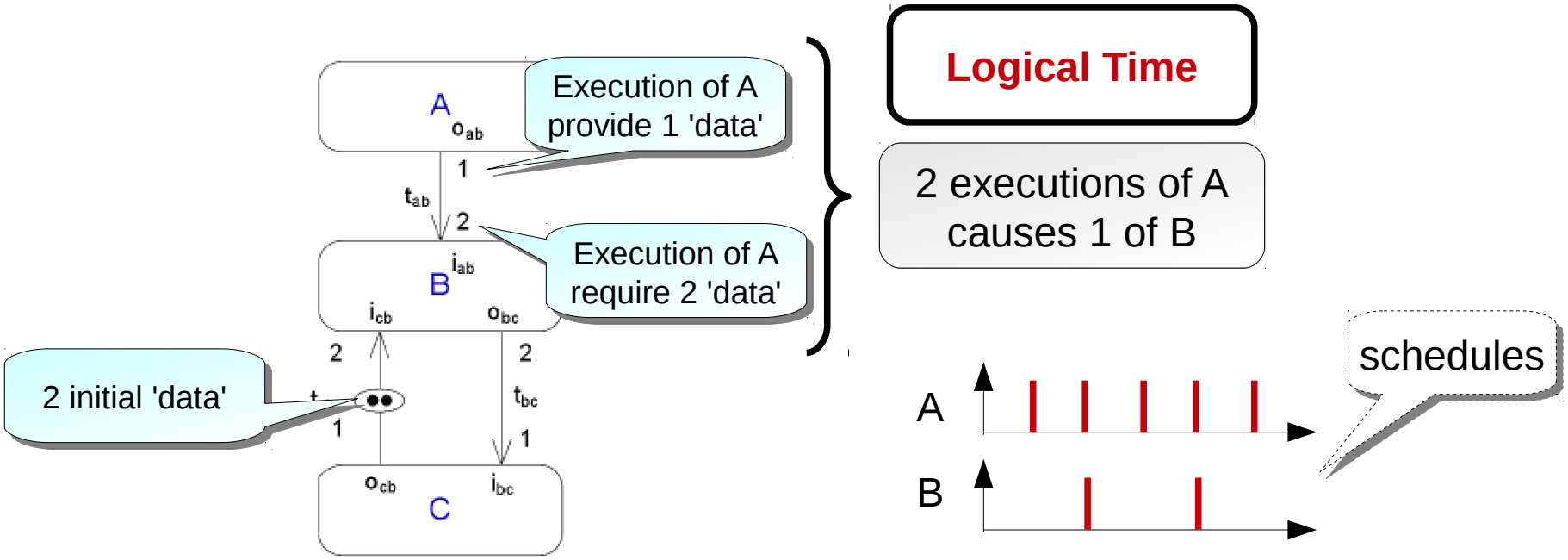
Synchronous Data Flow



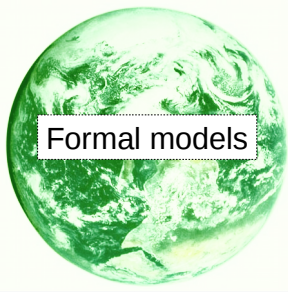
Formal Models (MoCC)



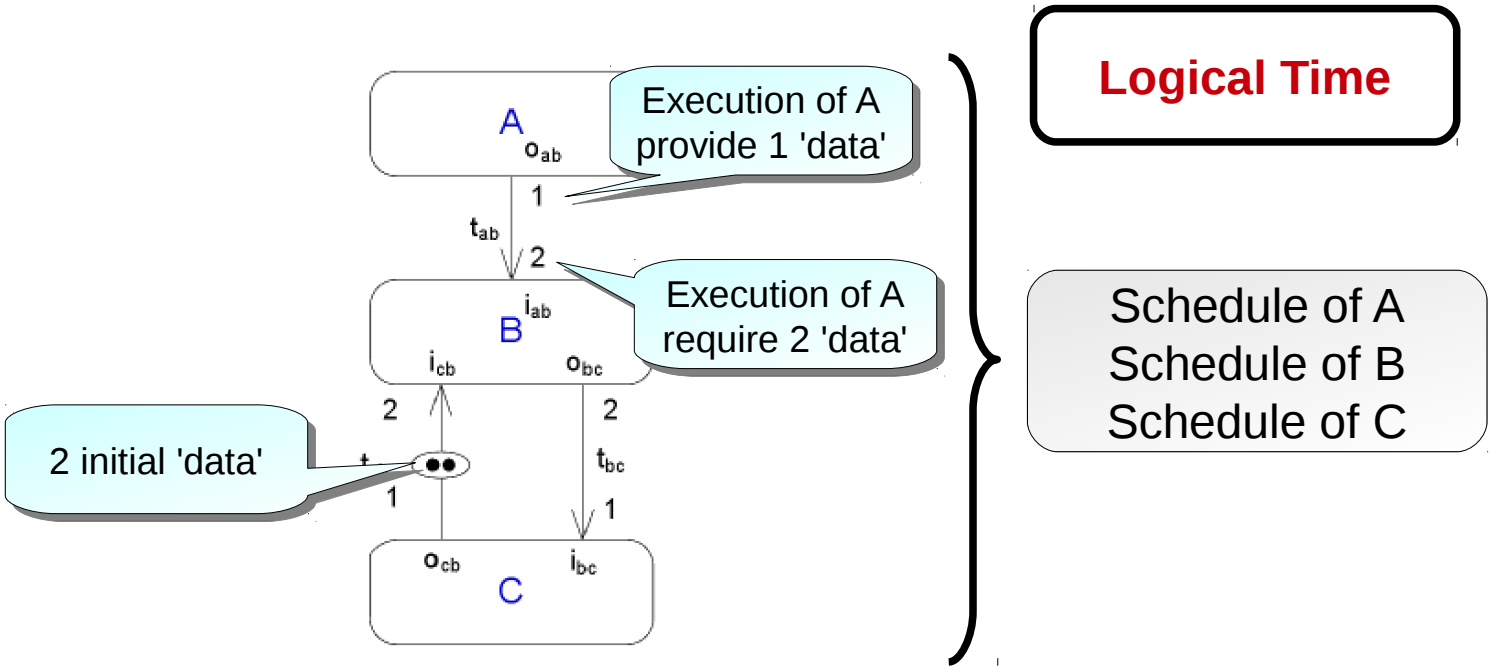
Synchronous Data Flow



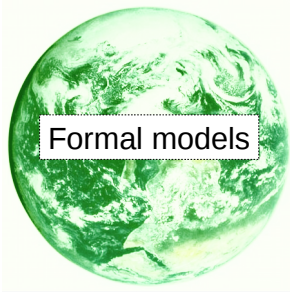
Formal Models (MoCC)



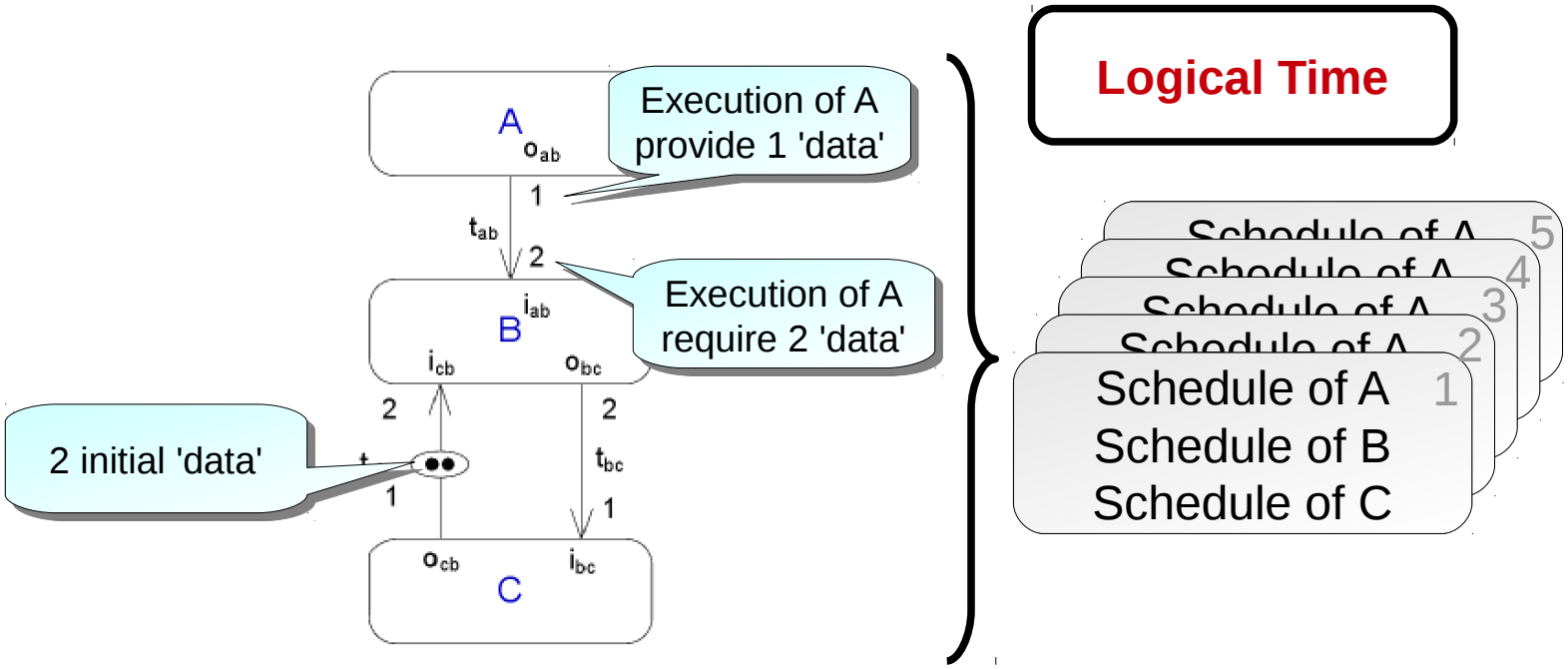
Synchronous Data Flow



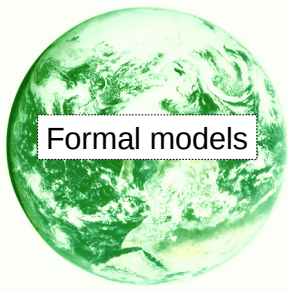
Formal Models (MoCC)



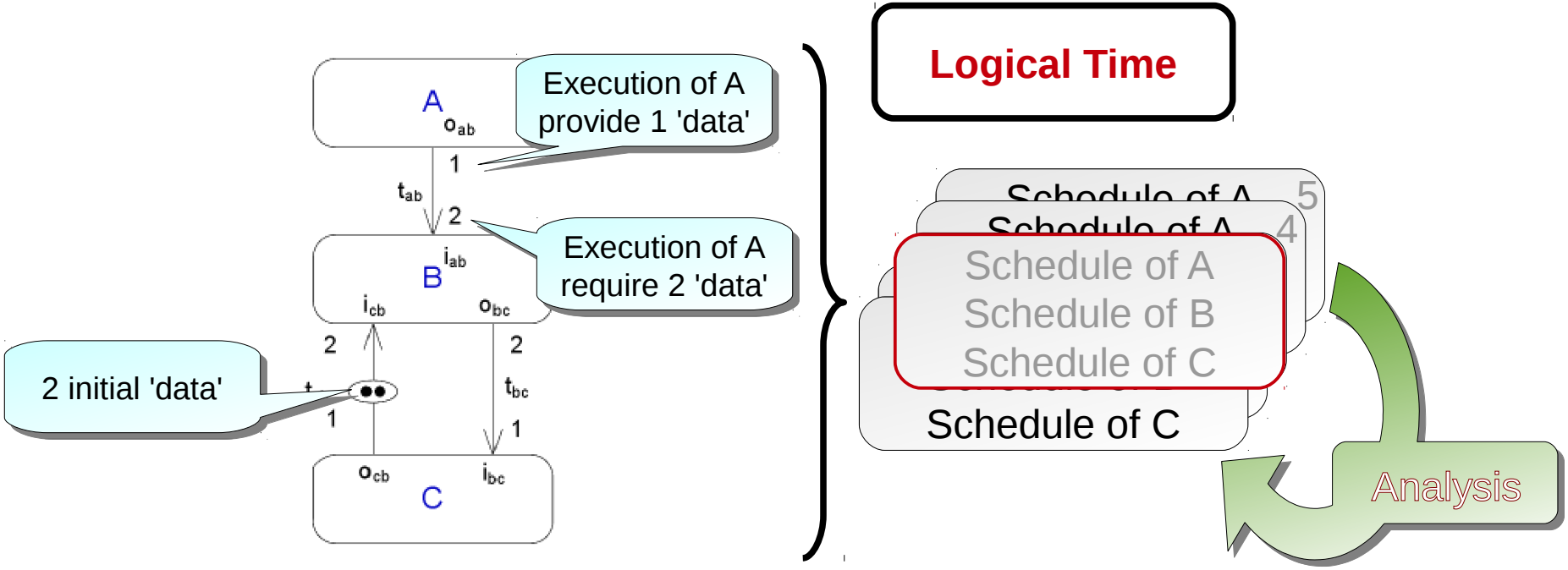
Synchronous Data Flow



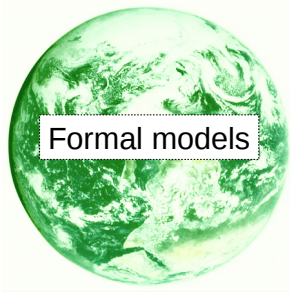
Formal Models (MoCC)



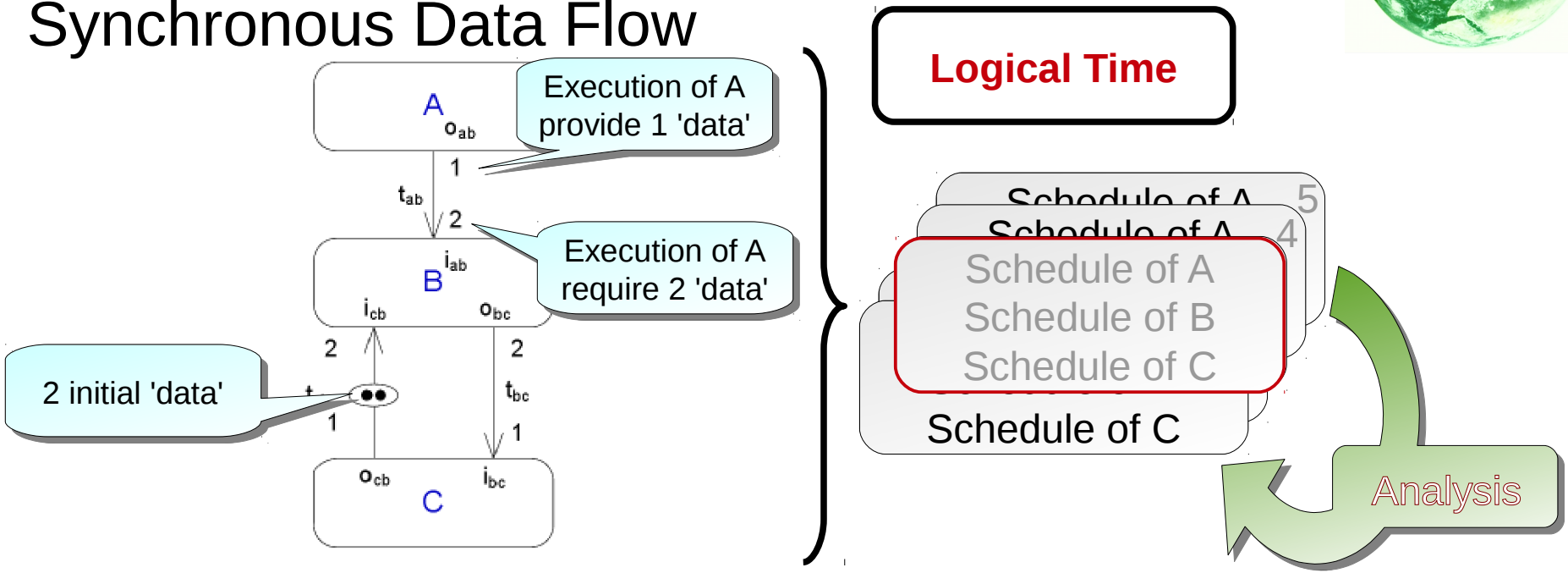
Synchronous Data Flow



Formal Models (MoCC)



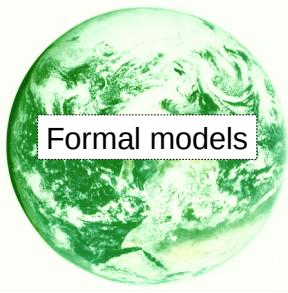
Synchronous Data Flow



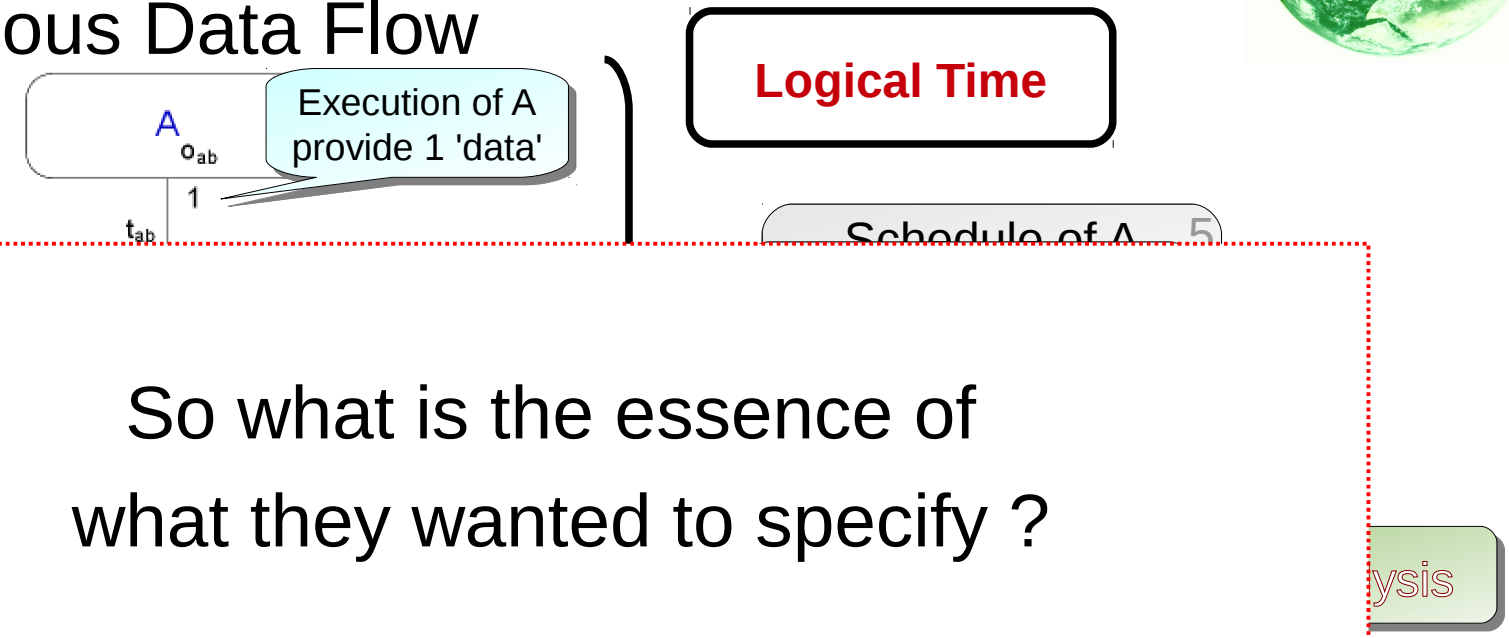
Still...

- (they are languages where) the semantics is either (mathematically) given on paper or encoded in tools and algorithms...

Formal Models (MoCC)



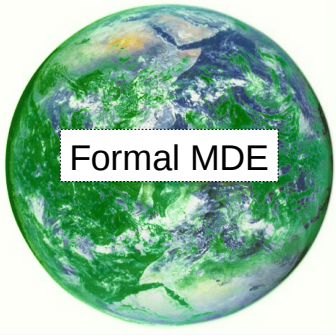
Synchronous Data Flow



Still...

- these are languages where the semantics is either (mathematically) given on paper or encoded in tools and algorithms...

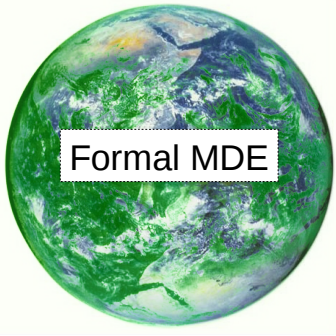
Time @ Design Time



- Expliciting the MoCC within an MDE environment
  - Adding explicit activation conditions
  - Adding relations between these activation conditions

→ formal semantics explicit within the model
(not hidden in the simulator / any transformation)

Time @ Design Time

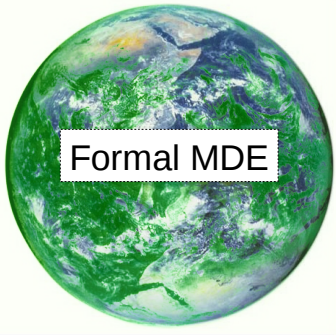


- Expliciting the MoCC within an MDE environment
  - Adding explicit activation conditions
  - Adding relations between these activation conditions

→ formal semantics explicit within the model
(not hidden in a simulator or a transformation)

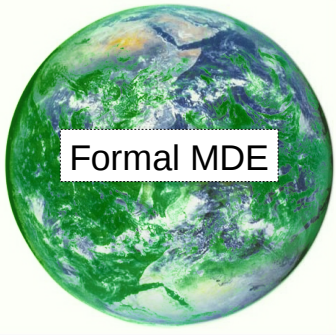
→ *The MARTE time model and CCSL were a first step toward that*

Time @ Design Time



- MDE is good for the structural concern
- Logical Time is good for the dynamic concern
  - Adding explicit activation conditions
  - Adding relations between these activation conditions

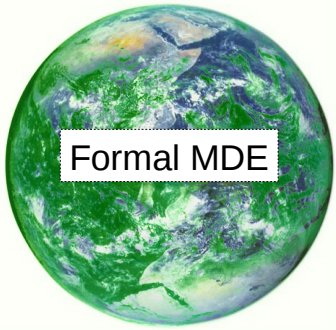
Time @ Design Time



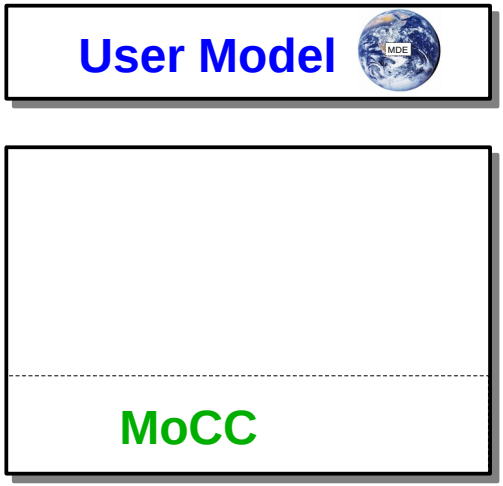
- MDE is good for the structural concern
- Logical Time is good for the dynamic concern
  - Adding explicit activation conditions
  - Adding relations between these activation conditions

User Model 

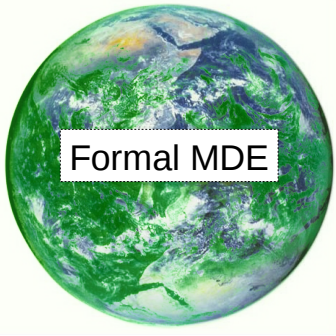
Time @ Design Time



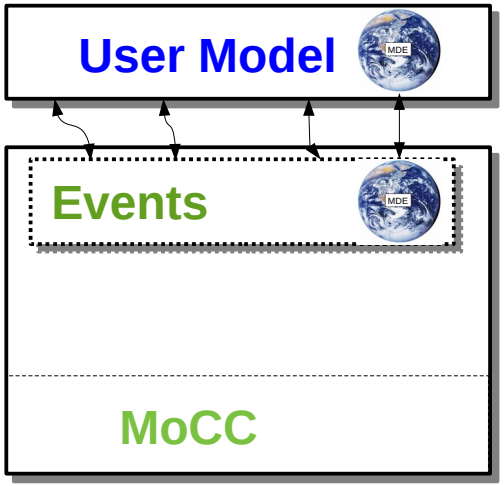
- MDE is good for the structural concern
- Logical Time MoCC is good for the dynamic concern
  - Adding explicit activation conditions
  - Adding relations between these activation conditions



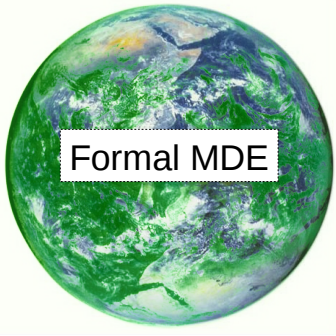
Time @ Design Time



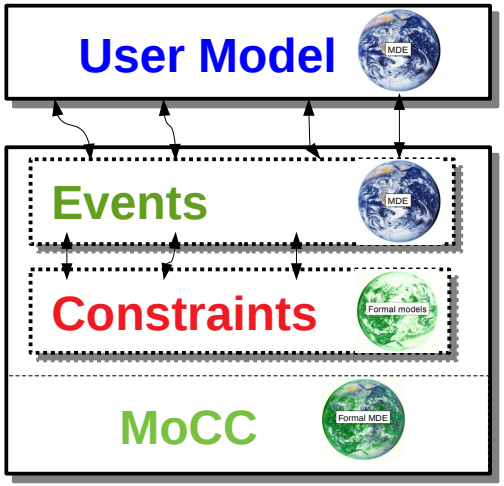
- MDE is good for the structural concern
- Logical Time is good for the dynamic concern
  - Adding explicit activation conditions
  - Adding relations between these activation conditions



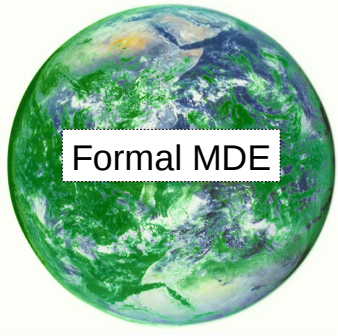
Time @ Design Time



- MDE is good for the structural concern
- Logical Time is good for the dynamic concern
  - Adding explicit activation conditions
  - Adding relations between these activation conditions

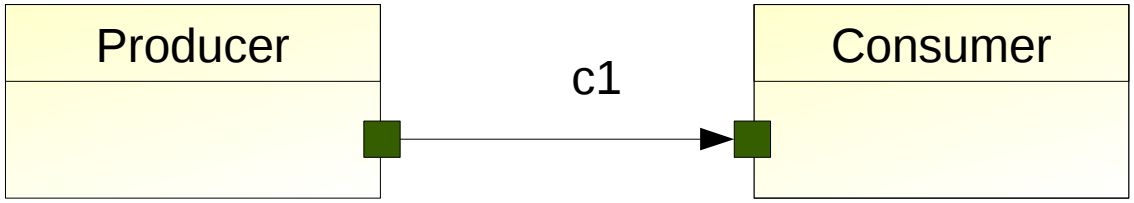


Time @ Design Time

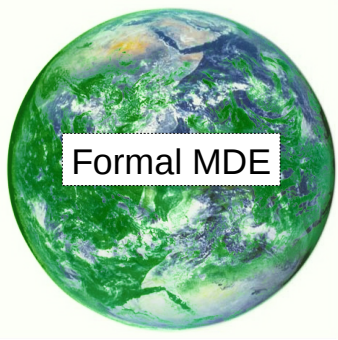


- Sketchy example of its use

User model

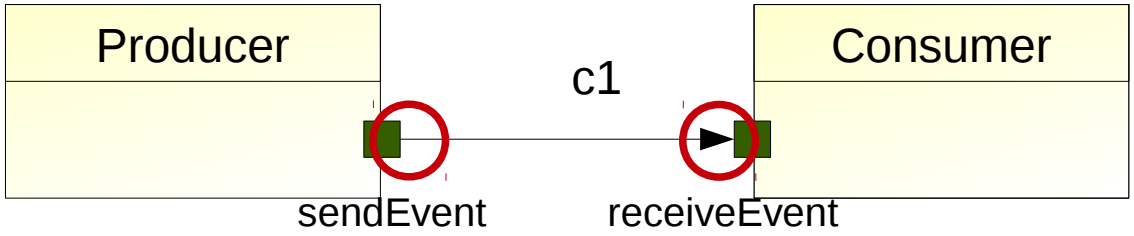


Time @ Design Time

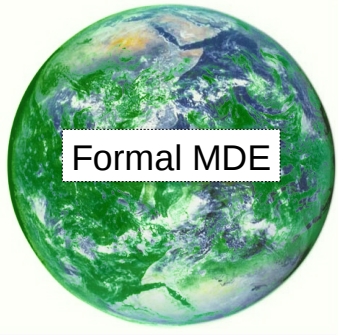


- Sketchy example of its use

User model

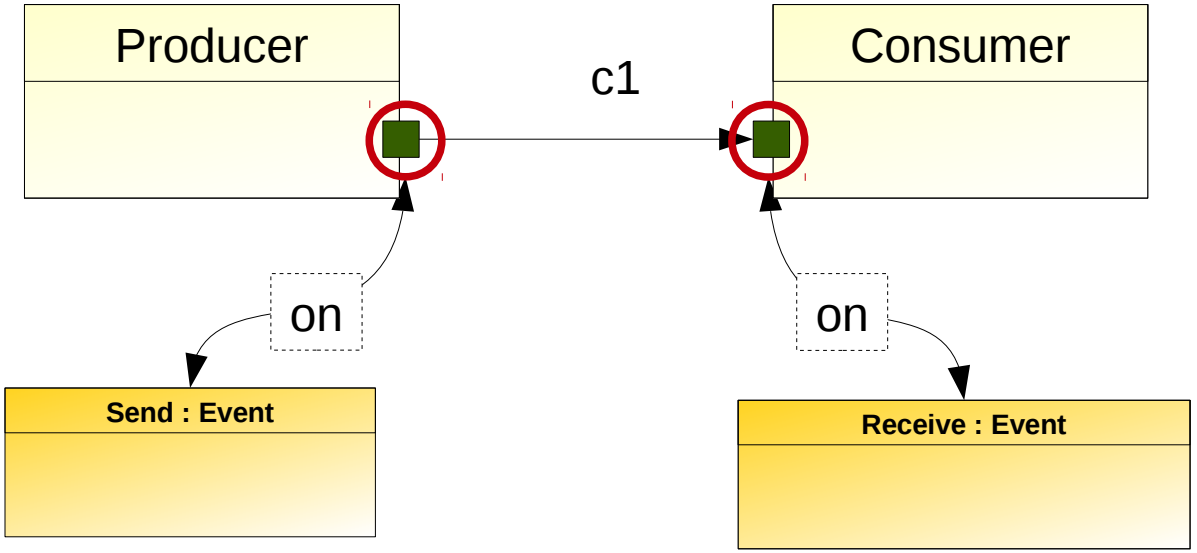


Time @ Design Time



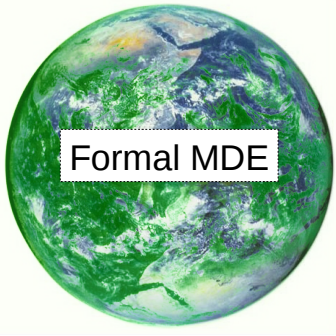
- Sketchy example of its use

User model



CCSL model

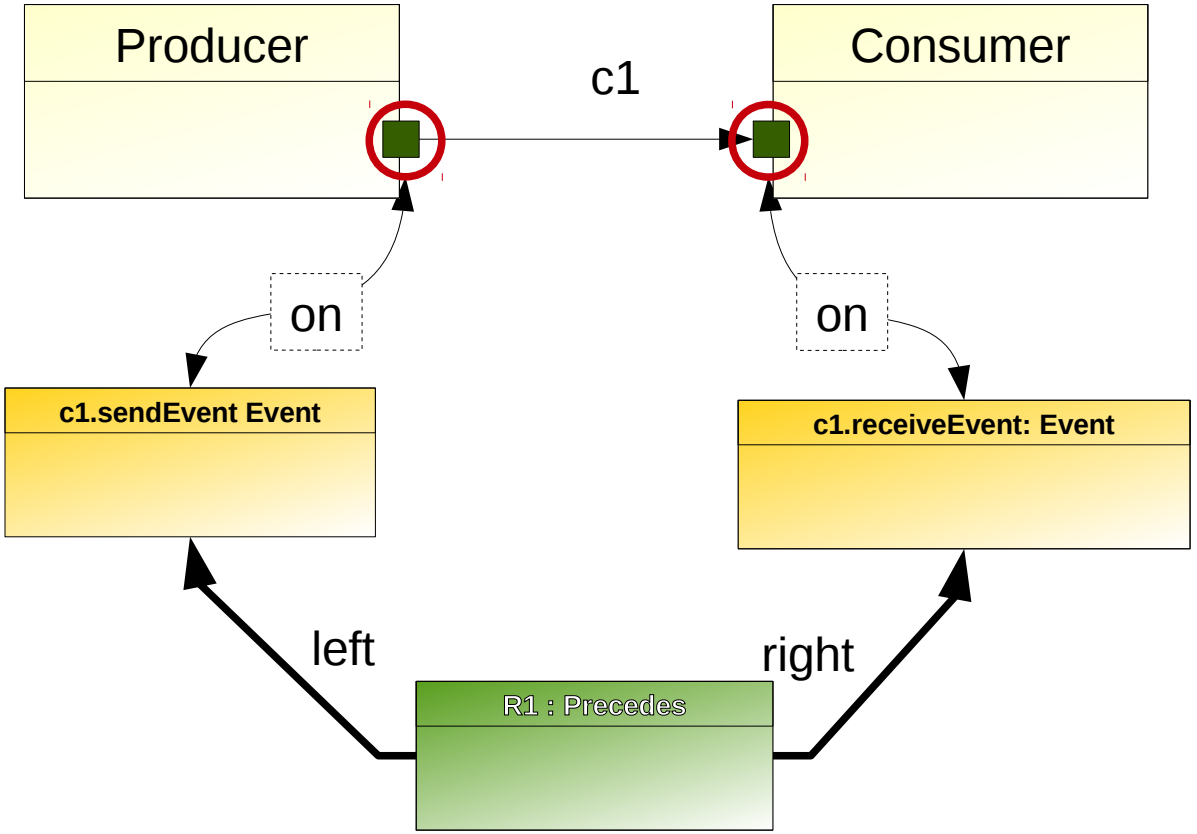
Time @ Design Time



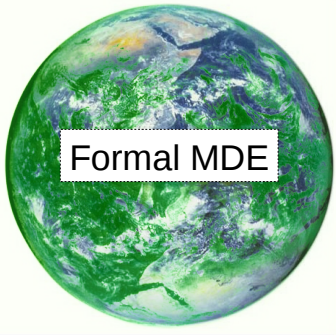
- Sketchy example of its use

User model

CCSL model

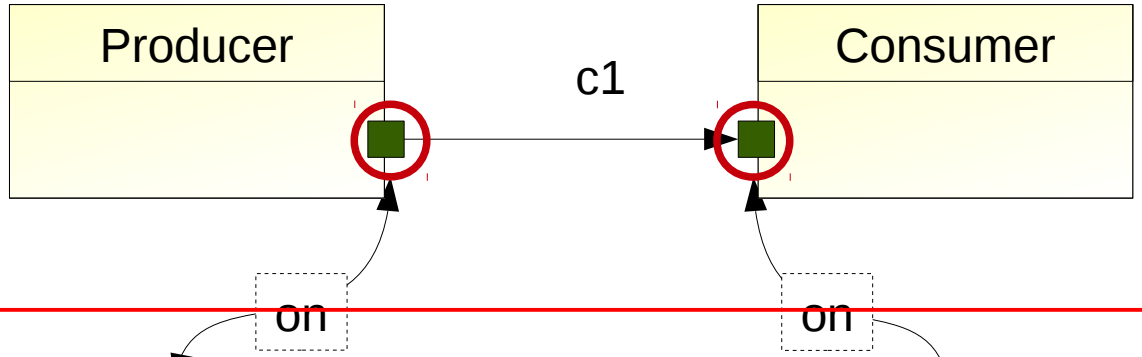


Time @ Design Time

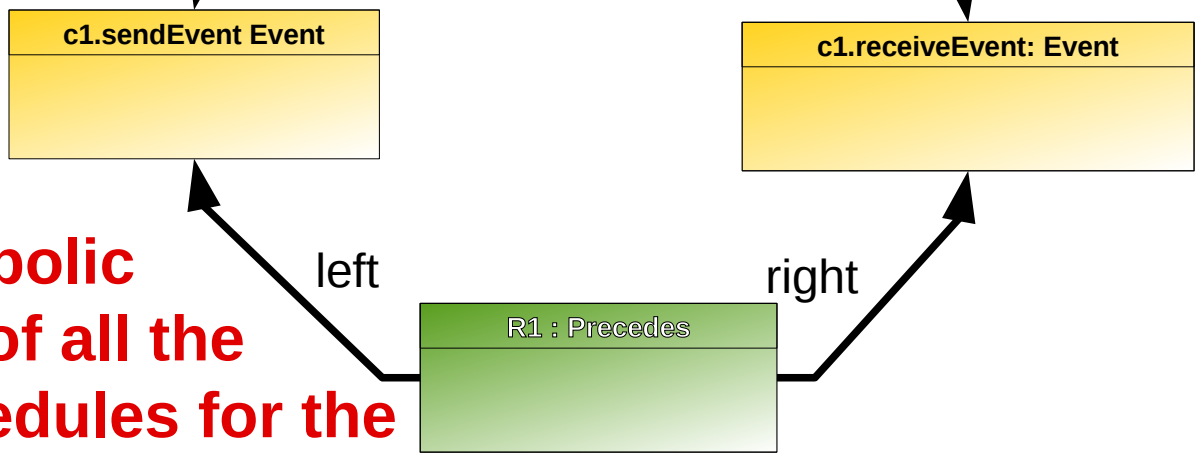


- Sketchy example of its use

User model



CCSL model



→ this is a symbolic representation of all the acceptable schedules for the user model

Implementation in TimeSquare

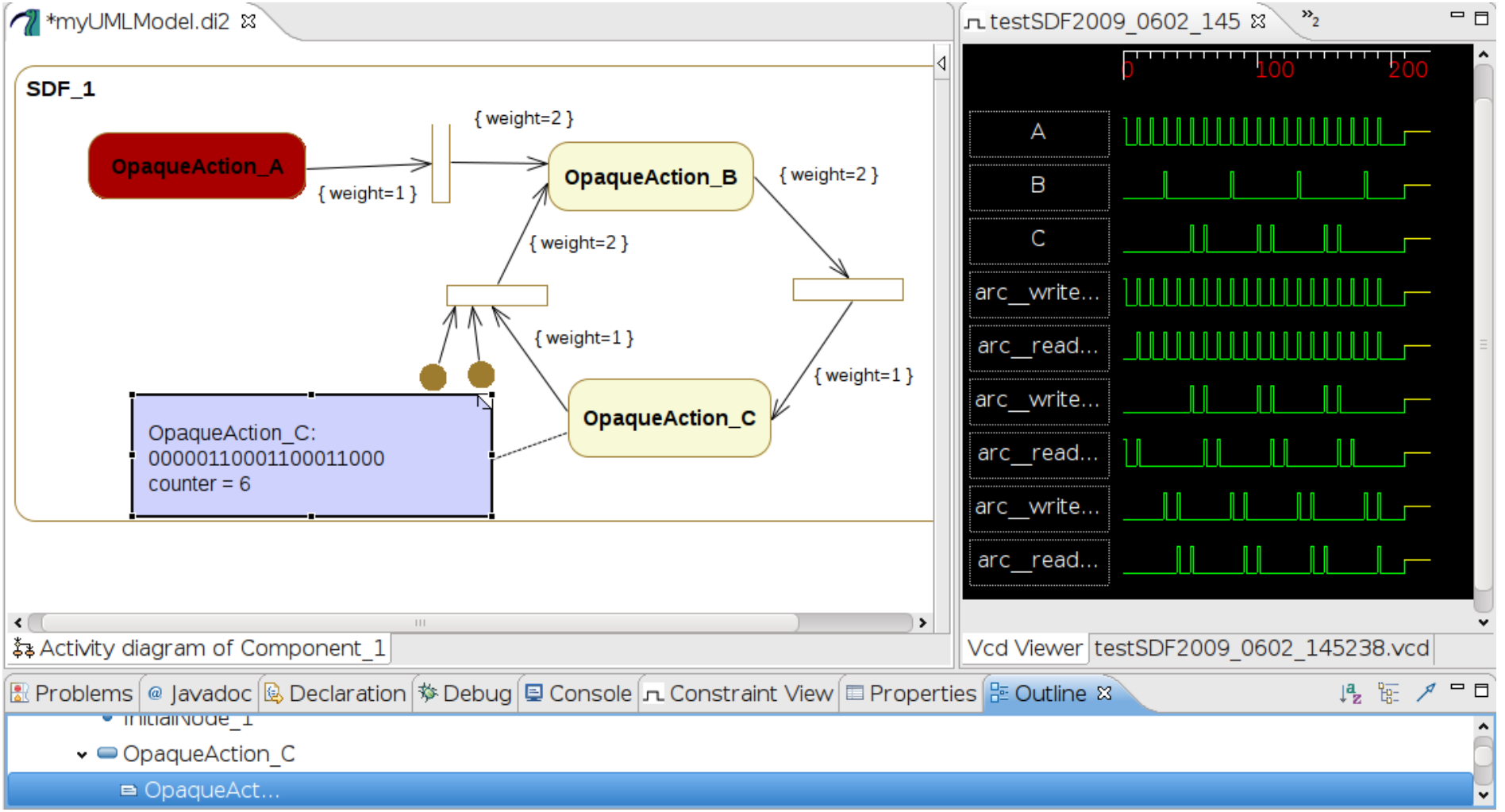
```
/*
 *  a simple SDF model, which use the SDF lib
 *
 *  @author:  Julien DeAntoni
 *  date :  Sun September 04 2010  15:04:26 CEST
 */
ClockConstraintSystem anSDFmodel {
    imports {
       //import
        import "platform:/plugin/fr.inria.aoste.timesquare.ccslkernel.model/ccsllibrary/kernel.ccslLib" as lib0;
        import "platform:/plugin/fr.inria.aoste.timesquare.ccslkernel.model/ccsllibrary/CCSL.ccslLib" as lib1;
        import "SDF.ccslLib" as lib2;
        import "model_SDF.uml" as model;

    }
    entryBlock main

        Block main {
        Clock A->activeA("ActorA")
        Clock B->activeB("ActorB")
        Clock C->activeC("ActorC")
        Integer zero=0
        Integer one=1
        Integer two=2
        Relation arcAtoB[Arc](   Arc_delay->    zero,
                                 Arc_source->   A,
                                 Arc_outWeight->one,
                                 Arc_target->   B,
                                 Arc_inWeight-> two
        )
        Relation arcBtoC[Arc](   Arc_delay->    zero,

        )
```

RelationDeclaration **Arc** : SDF::SDF_Relations::Arc

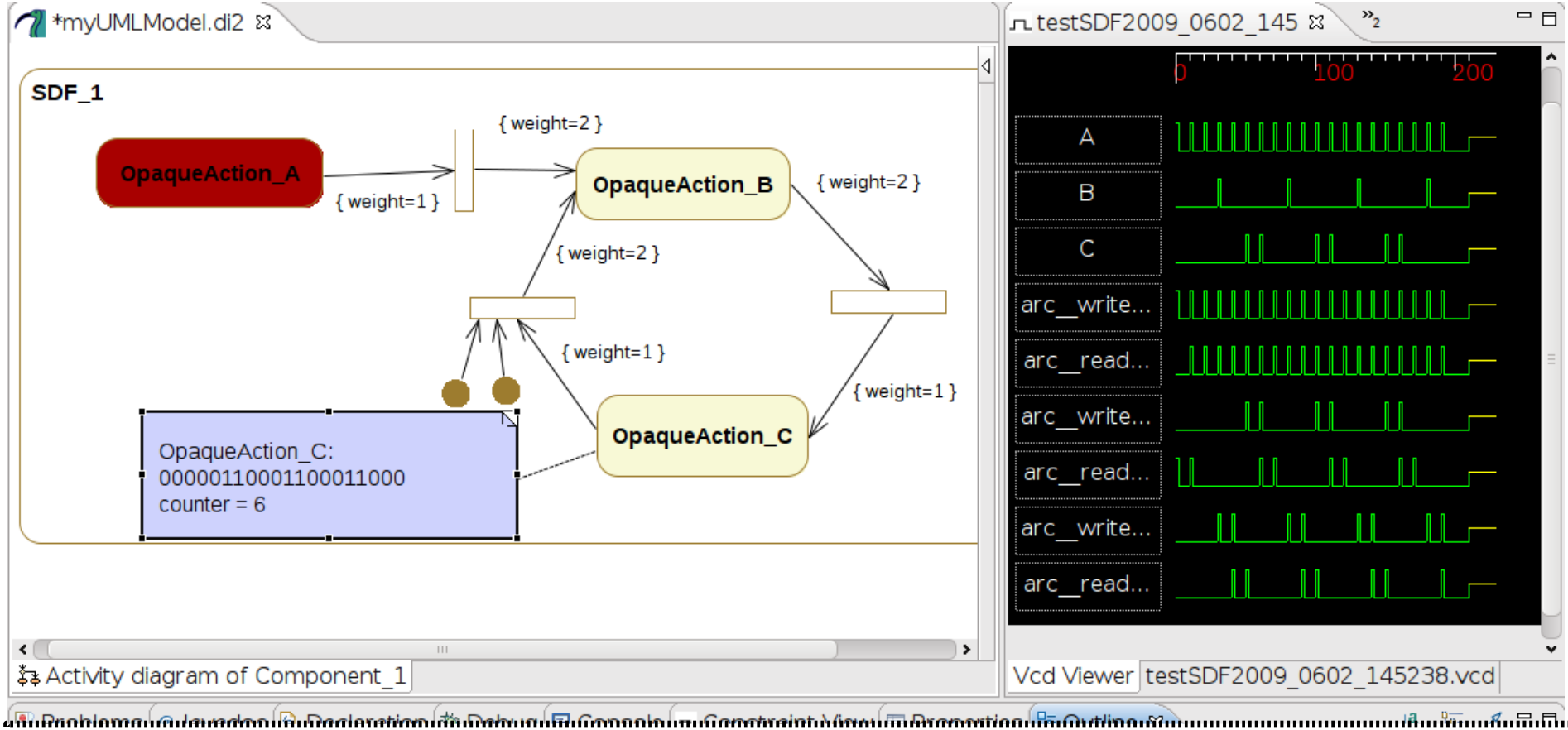
Containment hierarchy:
SDF [Library]
  SDF_Relations [RelationLibrary]
    Arc [RelationDeclaration]

Press 'F2' for focus

Implementation in TimeSquare



Implementation in TimeSquare

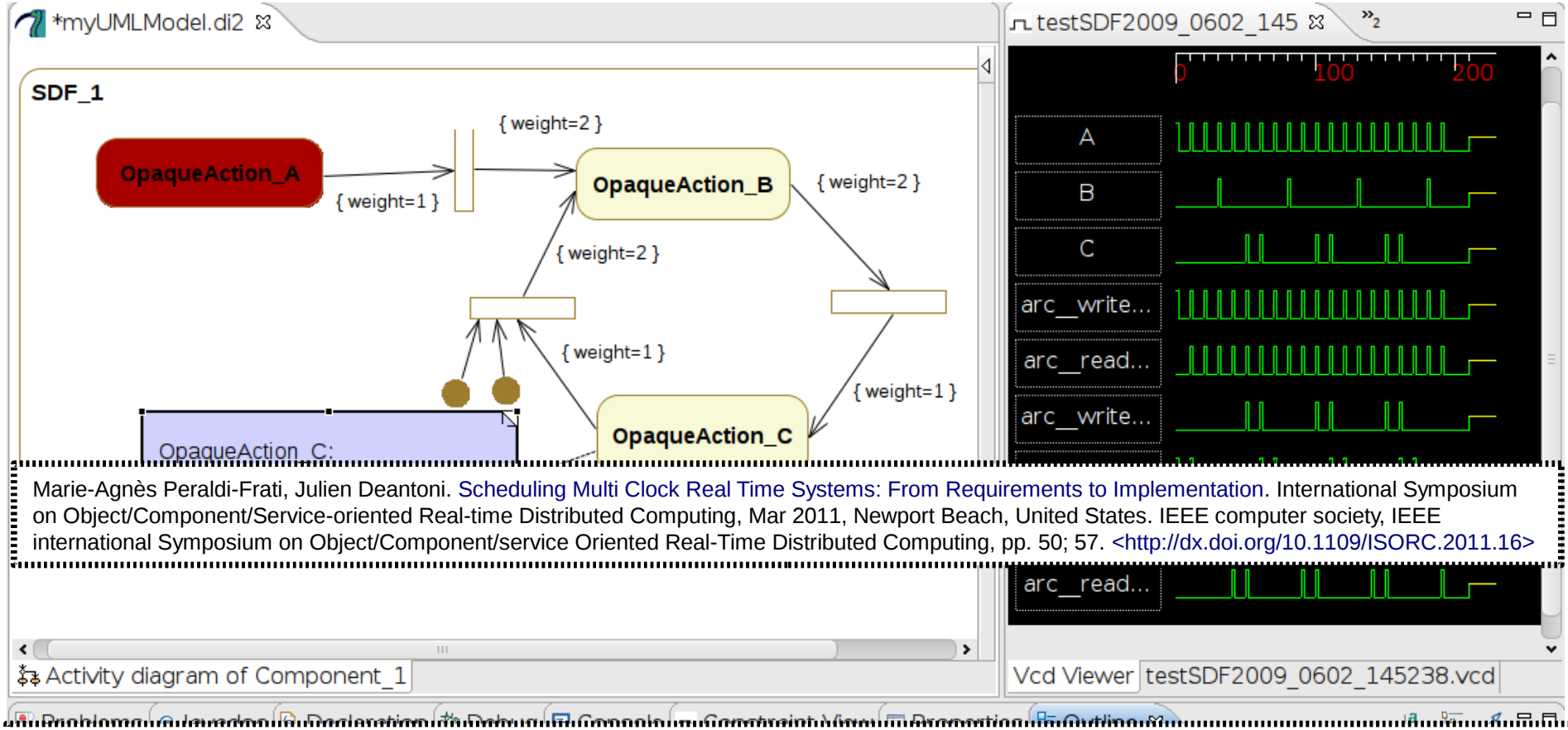


Julien Deantoni, Frédéric Mallet, Charles André. On the Formal Execution of UML and DSL Models. [Short paper]. WIP of the 4th International School on Model-Driven Development for Distributed, Realtime, Embedded Systems, Apr 2009, Aussois, France

Frédéric Mallet, Charles André, Julien Deantoni. Executing AADL models with UML/Marte. Int. Conf. Engineering of Complex Computer Systems - ICECCS'09, Jun 2009, Potsdam, Germany. pp. 371-376. <http://dx.doi.org/10.1109/ICECCS.2009.10>

Frédéric Mallet, Julien Deantoni, Charles André, Robert De Simone. The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering, Springer London, 2010, 6 (1-2), pp. 99-106. <http://dx.doi.org/10.1007/s11334-009-0109-0>

Implementation in TimeSquare



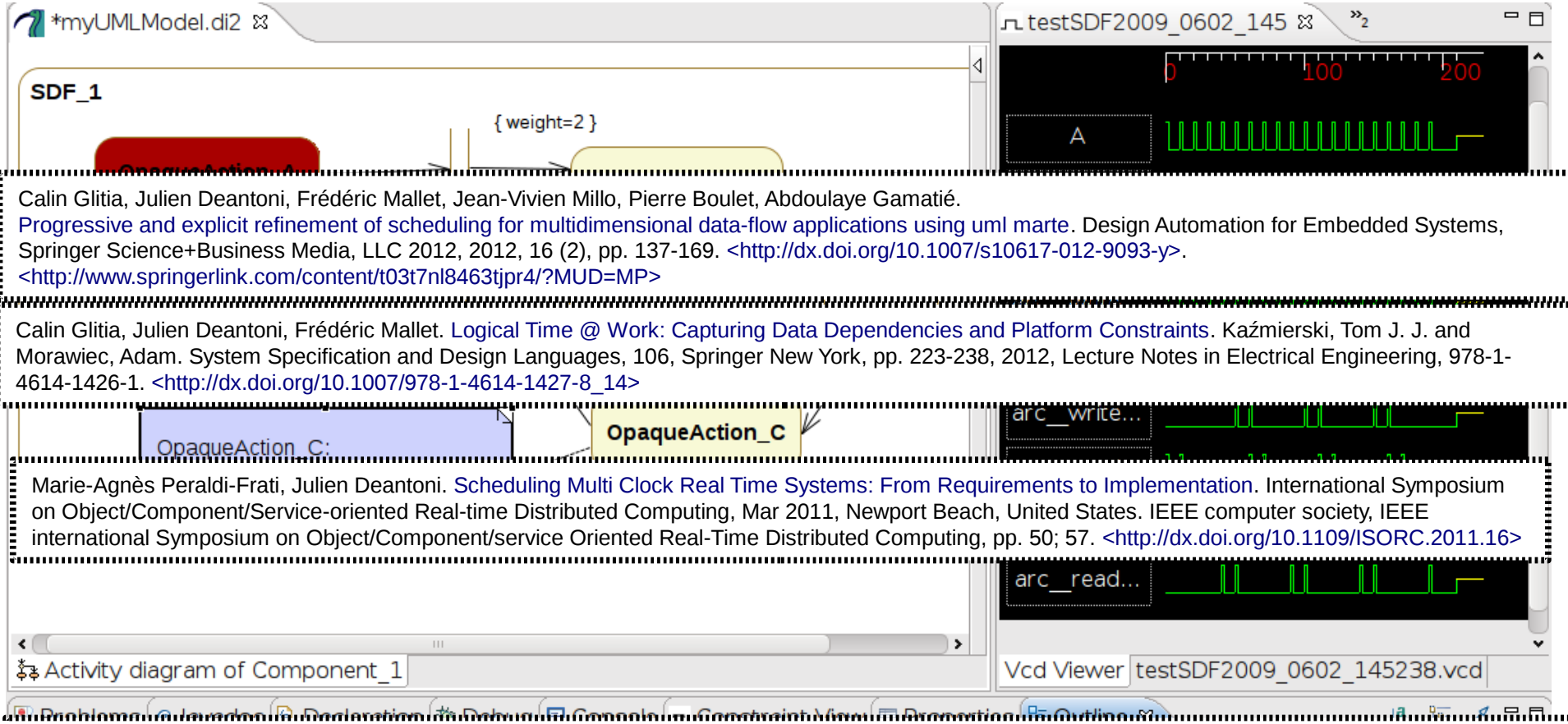
Marie-Agnès Peraldi-Frati, Julien Deantoni. Scheduling Multi Clock Real Time Systems: From Requirements to Implementation. International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, Mar 2011, Newport Beach, United States. IEEE computer society, IEEE international Symposium on Object/Component/service Oriented Real-Time Distributed Computing, pp. 50; 57. <http://dx.doi.org/10.1109/ISORC.2011.16>

Julien Deantoni, Frédéric Mallet, Charles André. On the Formal Execution of UML and DSL Models. [Short paper]. WIP of the 4th International School on Model-Driven Development for Distributed, Realtime, Embedded Systems, Apr 2009, Aussois, France

Frédéric Mallet, Charles André, Julien Deantoni. Executing AADL models with UML/Marte. Int. Conf. Engineering of Complex Computer Systems - ICECCS'09, Jun 2009, Potsdam, Germany. pp. 371-376. <http://dx.doi.org/10.1109/ICECCS.2009.10>

Frédéric Mallet, Julien Deantoni, Charles André, Robert De Simone. The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering, Springer London, 2010, 6 (1-2), pp. 99-106. <http://dx.doi.org/10.1007/s11334-009-0109-0>

Implementation in TimeSquare



Calin Glitia, Julien Deantoni, Frédéric Mallet, Jean-Vivien Millo, Pierre Boulet, Abdoulaye Gamatié. Progressive and explicit refinement of scheduling for multidimensional data-flow applications using uml marte. Design Automation for Embedded Systems, Springer Science+Business Media, LLC 2012, 2012, 16 (2), pp. 137-169. <http://dx.doi.org/10.1007/s10617-012-9093-y>. <http://www.springerlink.com/content/t03t7nl8463tjpr4/?MUD=MP>

Calin Glitia, Julien Deantoni, Frédéric Mallet. Logical Time @ Work: Capturing Data Dependencies and Platform Constraints. Kaźmierski, Tom J. J. and Morawiec, Adam. System Specification and Design Languages, 106, Springer New York, pp. 223-238, 2012, Lecture Notes in Electrical Engineering, 978-1-4614-1426-1. <http://dx.doi.org/10.1007/978-1-4614-1427-8_14>

Marie-Agnès Peraldi-Frati, Julien Deantoni. Scheduling Multi Clock Real Time Systems: From Requirements to Implementation. International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, Mar 2011, Newport Beach, United States. IEEE computer society, IEEE international Symposium on Object/Component/service Oriented Real-Time Distributed Computing, pp. 50; 57. <http://dx.doi.org/10.1109/ISORC.2011.16>

Julien Deantoni, Frédéric Mallet, Charles André. On the Formal Execution of UML and DSL Models. [Short paper]. WIP of the 4th International School on Model-Driven Development for Distributed, Realtime, Embedded Systems, Apr 2009, Aussois, France

Frédéric Mallet, Charles André, Julien Deantoni. Executing AADL models with UML/Marte. Int. Conf. Engineering of Complex Computer Systems - ICECCS'09, Jun 2009, Potsdam, Germany. pp. 371-376. <http://dx.doi.org/10.1109/ICECCS.2009.10>

Frédéric Mallet, Julien Deantoni, Charles André, Robert De Simone. The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering, Springer London, 2010, 6 (1-2), pp. 99-106. <http://dx.doi.org/10.1007/s11334-009-0109-0>

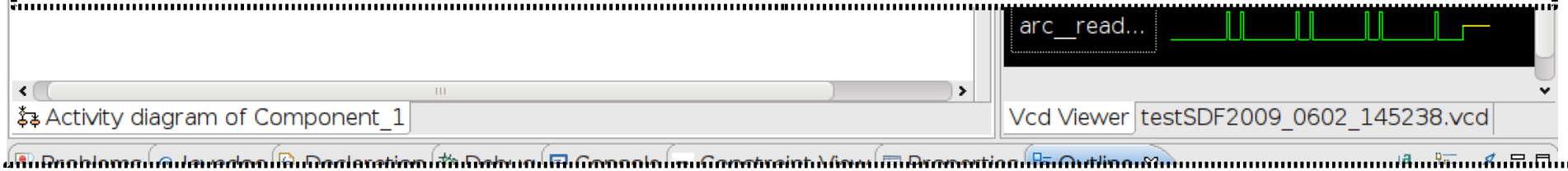
Implementation in TimeSquare

Kelly Garcés, Julien Deantoni, Frédéric Mallet. A Model-Based Approach for Reconciliation of Polychronous Execution Traces. SEAA 2011 - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Aug 2011, Oulu, Finland. IEEE. <http://www.computer.org/portal/web/csdl/doi/10.1109/SEAA.2011.47>

Calin Glitia, Julien Deantoni, Frédéric Mallet, Jean-Vivien Millo, Pierre Boulet, Abdoulaye Gamatié. Progressive and explicit refinement of scheduling for multidimensional data-flow applications using uml marte. Design Automation for Embedded Systems, Springer Science+Business Media, LLC 2012, 2012, 16 (2), pp. 137-169. <http://dx.doi.org/10.1007/s10617-012-9093-y>. <http://www.springerlink.com/content/t03t7nl8463tjpr4/?MUD=MP>

Calin Glitia, Julien Deantoni, Frédéric Mallet. Logical Time @ Work: Capturing Data Dependencies and Platform Constraints. Kaźmierski, Tom J. J. and Morawiec, Adam. System Specification and Design Languages, 106, Springer New York, pp. 223-238, 2012, Lecture Notes in Electrical Engineering, 978-1-4614-1426-1. <http://dx.doi.org/10.1007/978-1-4614-1427-8_14>

Marie-Agnès Peraldi-Frati, Julien Deantoni. Scheduling Multi Clock Real Time Systems: From Requirements to Implementation. International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, Mar 2011, Newport Beach, United States. IEEE computer society, IEEE international Symposium on Object/Component/service Oriented Real-Time Distributed Computing, pp. 50; 57. <http://dx.doi.org/10.1109/ISORC.2011.16>



Julien Deantoni, Frédéric Mallet, Charles André. On the Formal Execution of UML and DSL Models. [Short paper]. WIP of the 4th International School on Model-Driven Development for Distributed, Realtime, Embedded Systems, Apr 2009, Aussois, France

Frédéric Mallet, Charles André, Julien Deantoni. Executing AADL models with UML/Marte. Int. Conf. Engineering of Complex Computer Systems - ICECCS'09, Jun 2009, Potsdam, Germany. pp. 371-376. <http://dx.doi.org/10.1109/ICECCS.2009.10>

Frédéric Mallet, Julien Deantoni, Charles André, Robert De Simone. The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering, Springer London, 2010, 6 (1-2), pp. 99-106. <http://dx.doi.org/10.1007/s11334-009-0109-0>

Implementation in TimeSquare

Kelly Garcés, Julien Deantoni, Frédéric Mallet. A Model-Based Approach for Reconciliation of Polychronous Execution Traces. SEAA 2011 - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Aug 2011, Oulu, Finland. IEEE. <http://www.computer.org/portal/web/csdl/doi/10.1109/SEAA.2011.47>

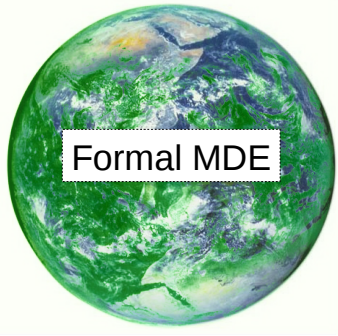
We were able to reason on
the scheduling state space of a model
at different places in the development cycle

Frédéric Mallet, Charles André, Julien Deantoni. Executing AADL models with UML/Marte. Int. Conf. Engineering of Complex Computer Systems - ICECCS'09, Jun 2009, Potsdam, Germany. pp. 371-376. <http://dx.doi.org/10.1109/ICECCS.2009.10>

Frédéric Mallet, Julien Deantoni, Charles André, Robert De Simone. The Clock Constraint Specification Language for building timed causality models. Innovations in Systems and Software Engineering, Springer London, 2010, 6 (1-2), pp. 99-106. <http://dx.doi.org/10.1007/s11334-009-0109-0>

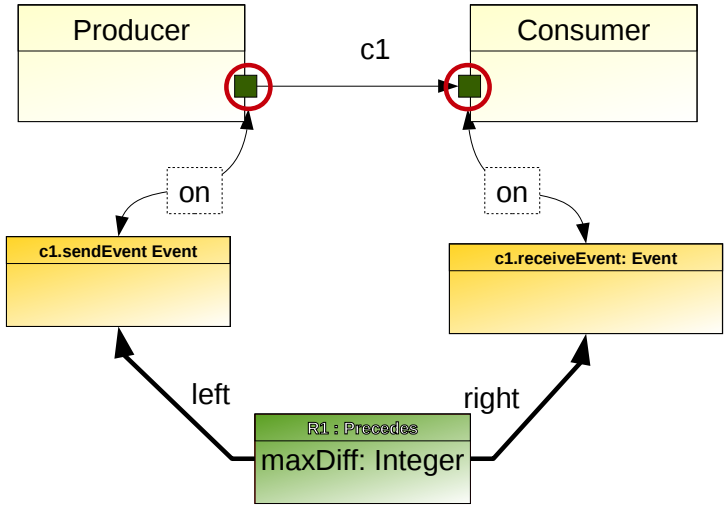
TimeSquare in one slide

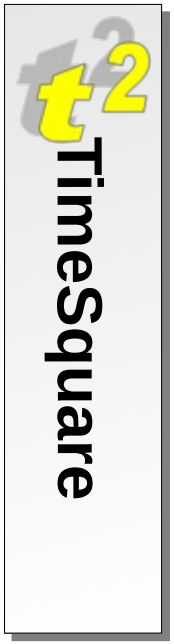
http://timesquare.inria.fr



User model

CCSL model





Simulation

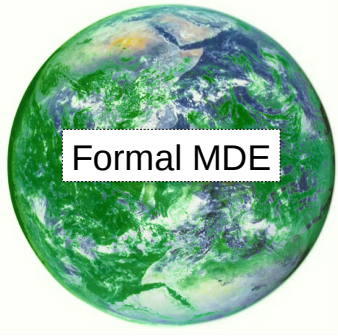
- Model animation
- Timing Diagram
- Code execution
- …

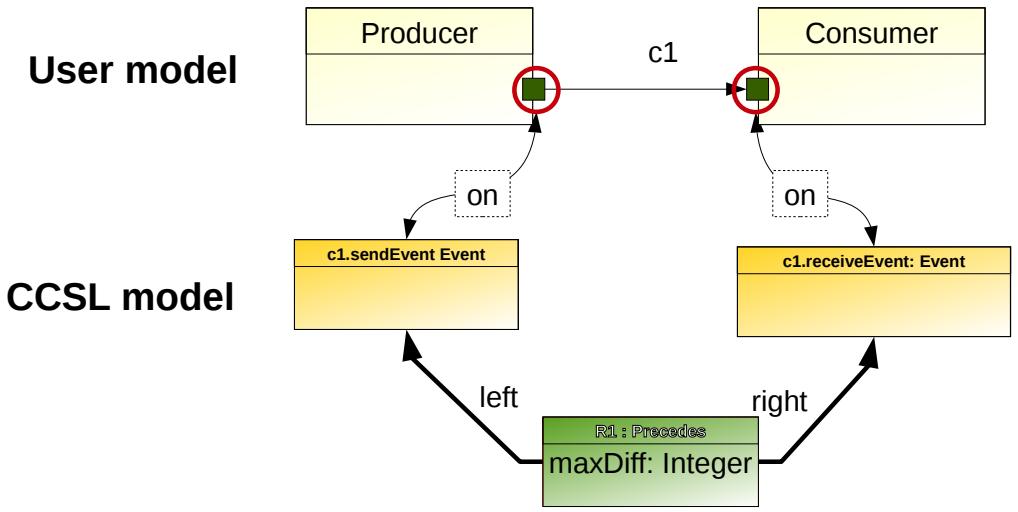
State space exploration

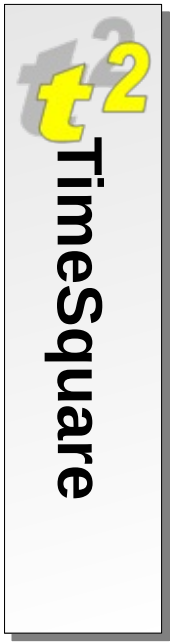
Julien Deantoni, Frédéric Mallet. TimeSquare: Treat your Models with Logical Time. Carlo A. Furia, Sebastian Nanz. TOOLS - 50th International Conference on Objects, Models, Components, Patterns - 2012, May 2012, Prague, Czech Republic. Springer, Objects, Models, Components, Patterns, 7304, pp. 34-41, Lecture Notes in Computer Science - LNCS. <http://dx.doi.org/10.1007/978-3-642-30561-0_4>

TimeSquare in one slide

http://timesquare.inria.fr







Simulation
- Model animation
- Timing Diagram
- Code execution
- …

State space exploration

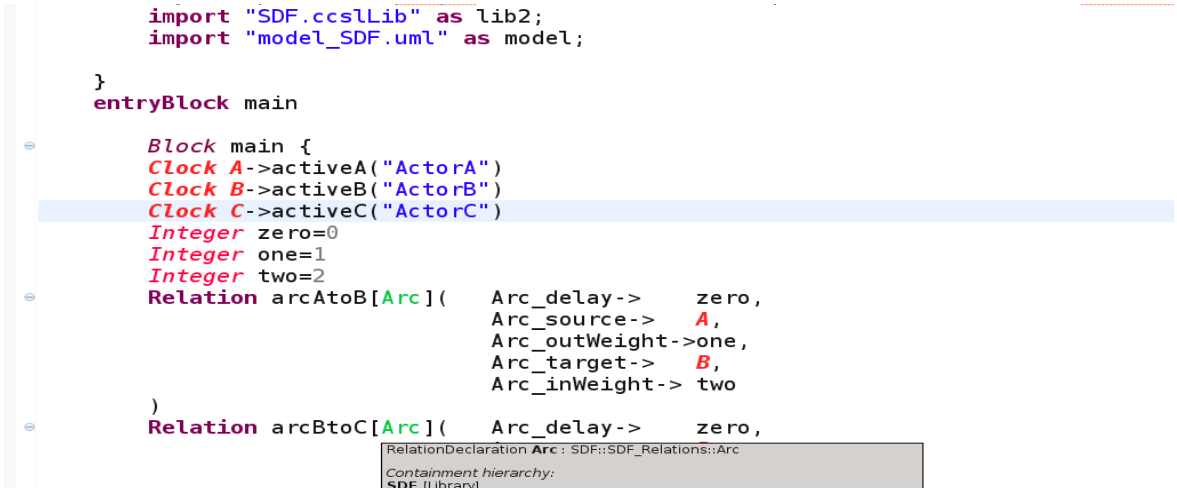
- Agnostic of the kind of user model
- Use the CCSL specification as a configuration of the model simulator
- Amenable to some V&V activities without knowledge of the user model syntax

Julien Deantoni, Frédéric Mallet. TimeSquare: Treat your Models with Logical Time. Carlo A. Furia, Sebastian Nanz. TOOLS - 50th International Conference on Objects, Models, Components, Patterns - 2012, May 2012, Prague, Czech Republic. Springer, Objects, Models, Components, Patterns, 7304, pp. 34-41, Lecture Notes in Computer Science - LNCS. <http://dx.doi.org/10.1007/978-3-642-30561-0_4>

Time @ Design Time

Good but:

– The CCSL specification is not a MoCC since it depends on a specific model

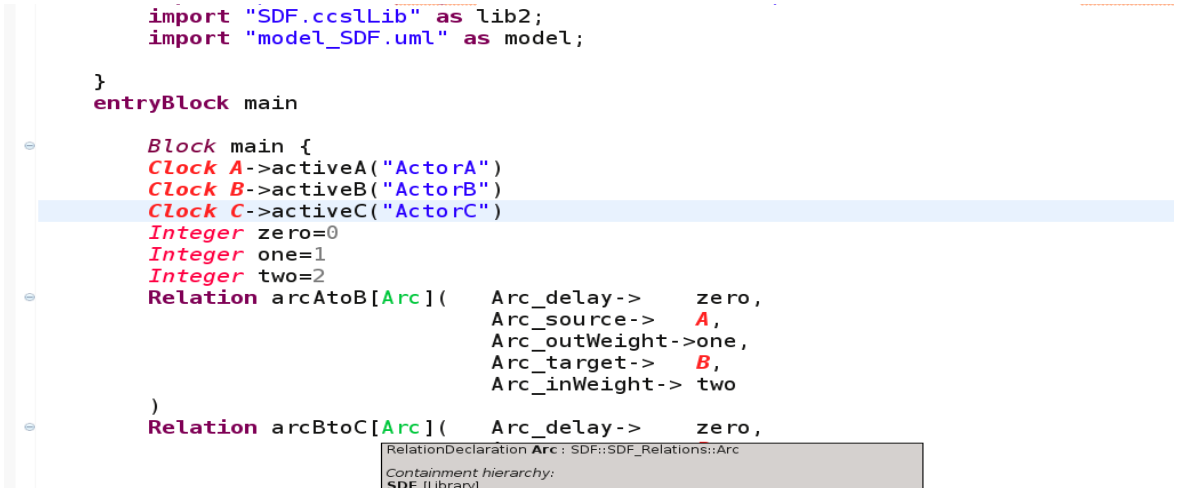


→ how can we provide a meta-language so that the MoCC can be defined on the syntax of a language.

Time @ Design Time

Good but:

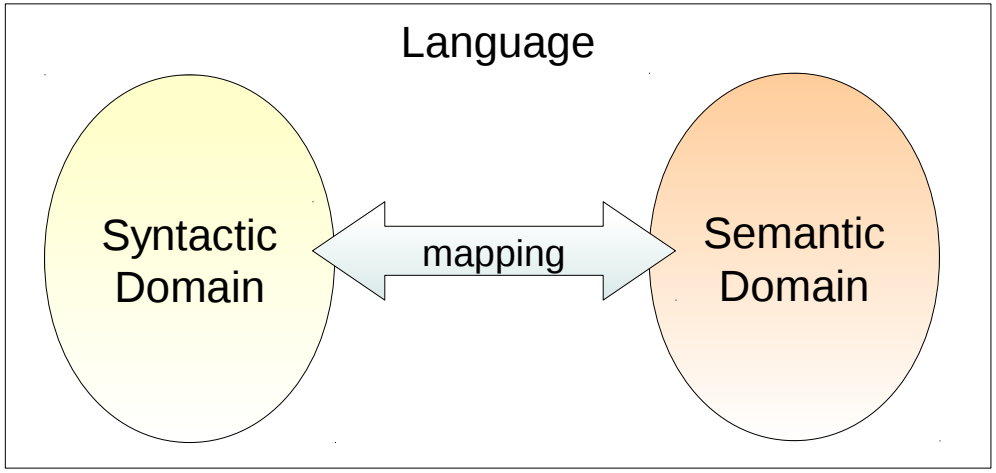
– The CCSL specification is not a MoCC since it depends on a specific model



→ how can we provide a meta-language so that the MoCC can be defined on the syntax of a language.

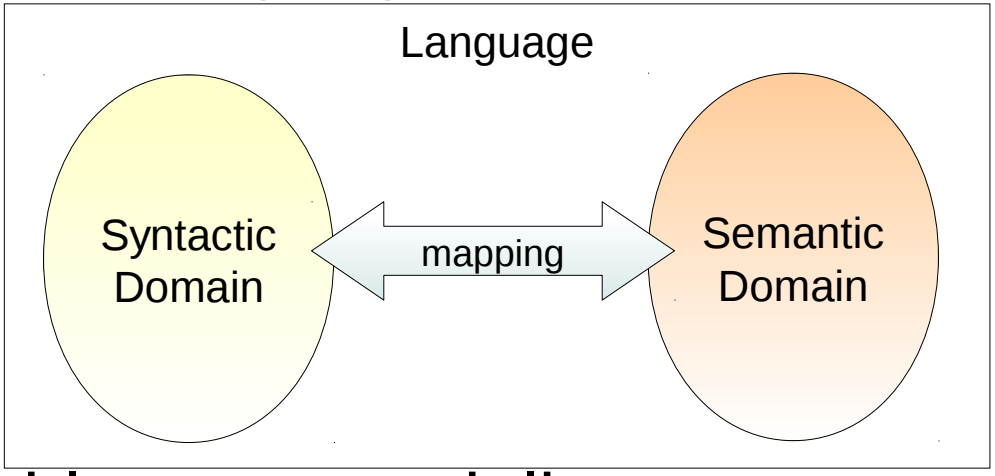
Reifying language behavioral semantics

- Executable language



Reifying language behavioral semantics

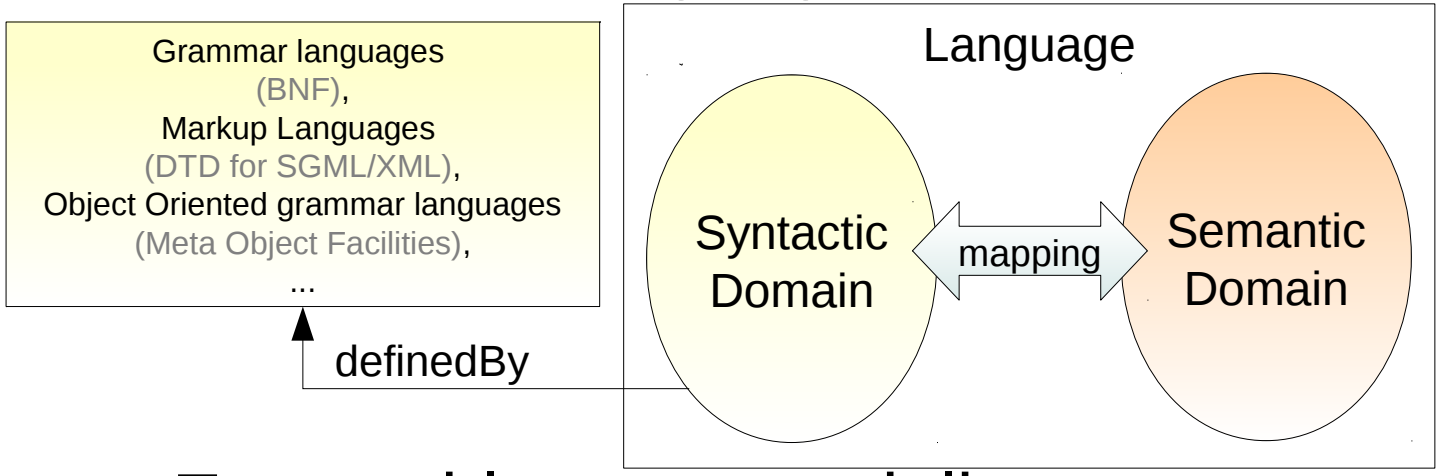
- Executable language



- Executable meta-modeling
  - Providing languages to define the language elements, i.e. providing some meta languages

Reifying language behavioral semantics

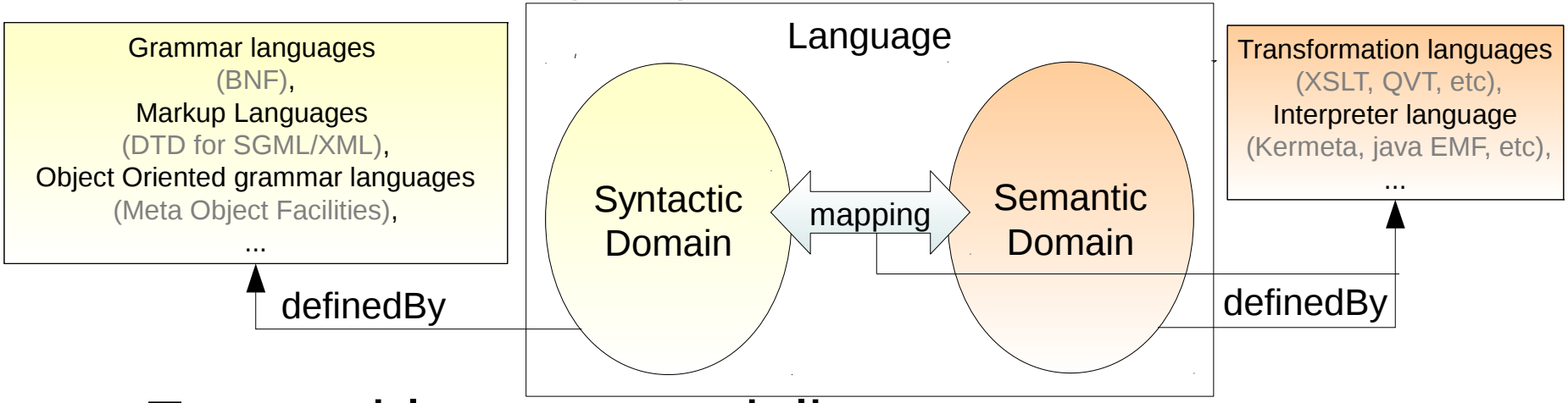
- Executable language



- Executable meta-modeling
  - Providing languages to define the language elements, i.e. providing some meta-languages

Reifying language behavioral semantics

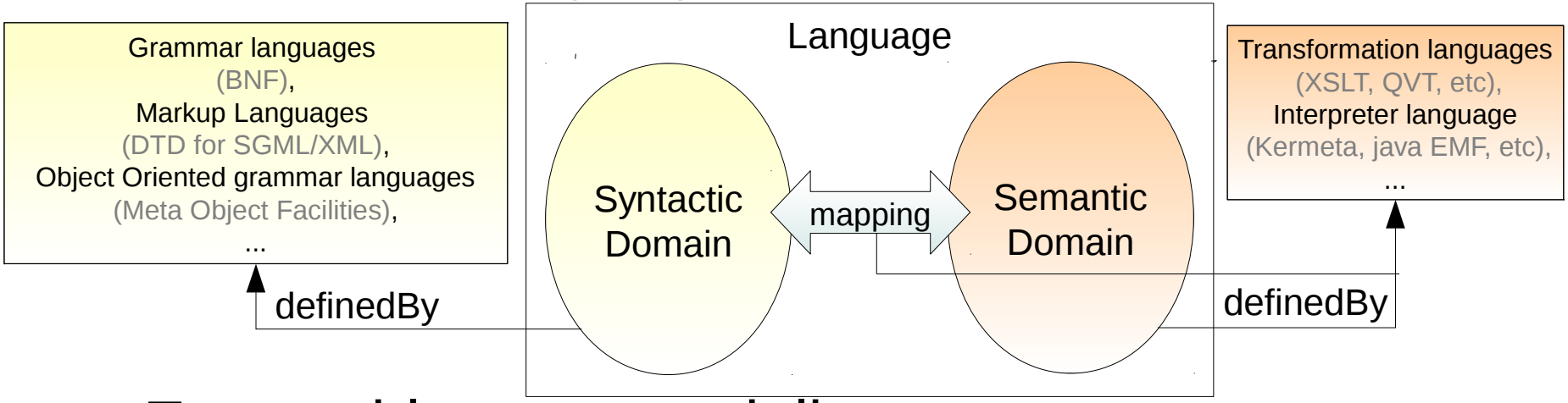
- Executable language



- Executable meta-modeling
  - Providing languages to define the language elements, i.e. providing some meta-languages

Reifying language behavioral semantics

- Executable language

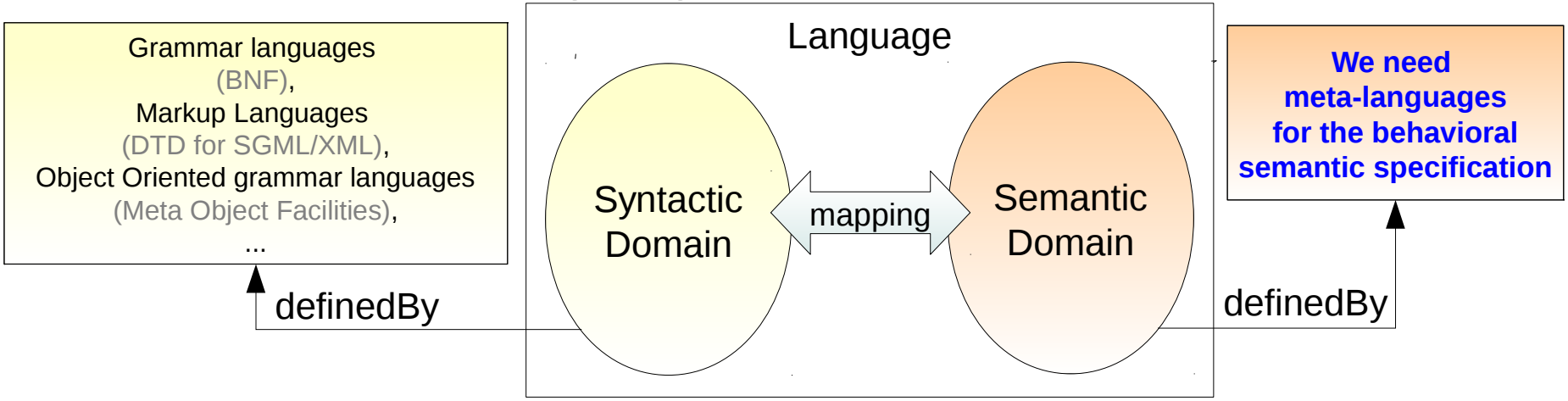


- Executable meta-modeling
  - Providing languages to define the language elements, i.e. providing some meta-languages

How can the behavioral semantics be manipulated ?
→ *we miss a concrete formal entity to represent it.*

Reifying language behavioral semantics

- Executable language

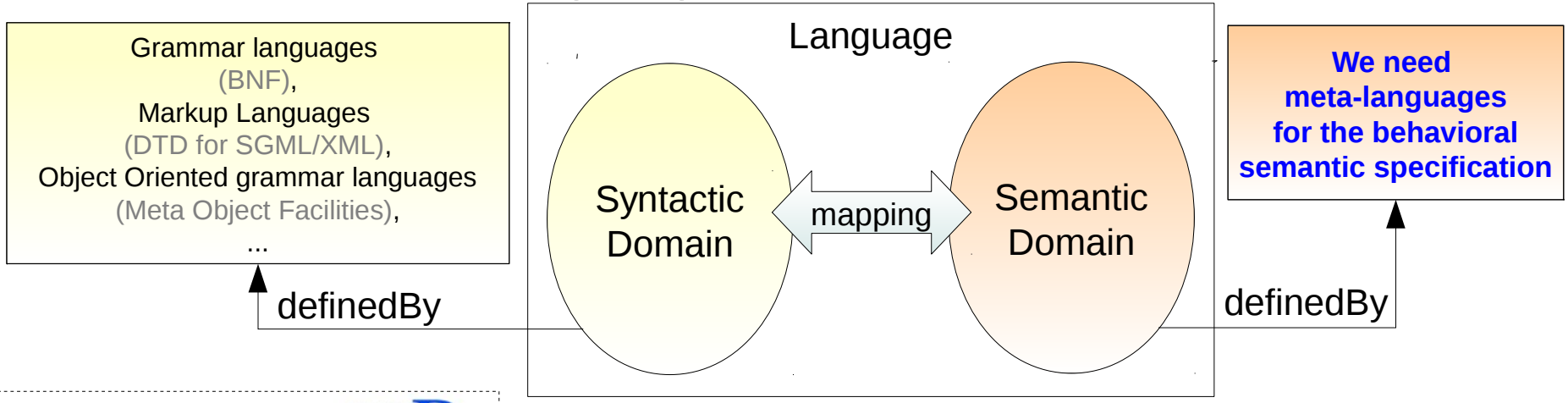


- Executable meta-modeling
  - Providing languages to define the language elements, i.e. providing some meta-languages

How can the behavioral semantics be manipulated ?
→ *we miss a concrete formal entity to represent it.*

Reifying language behavioral semantics

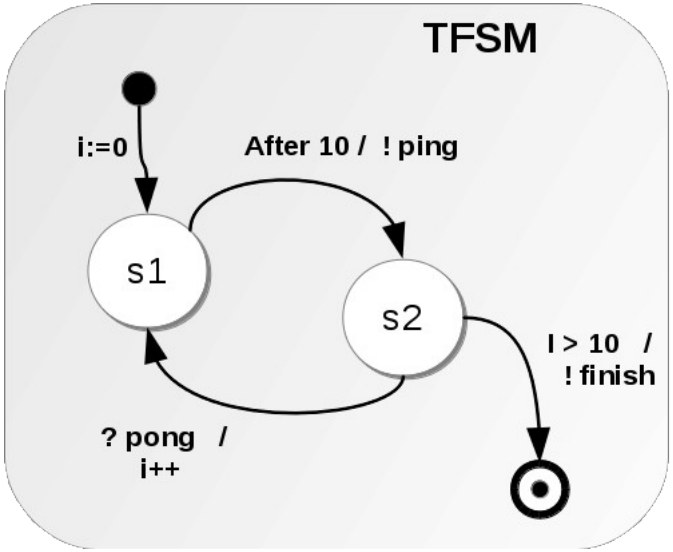
- Executable language



This is the first question we addressed in the GEMOC ANR project

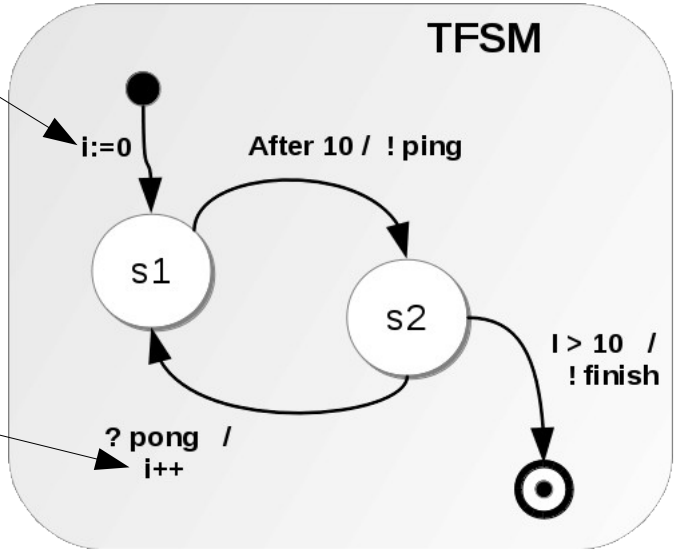
http://gemoc.org/ins

look at a simple example



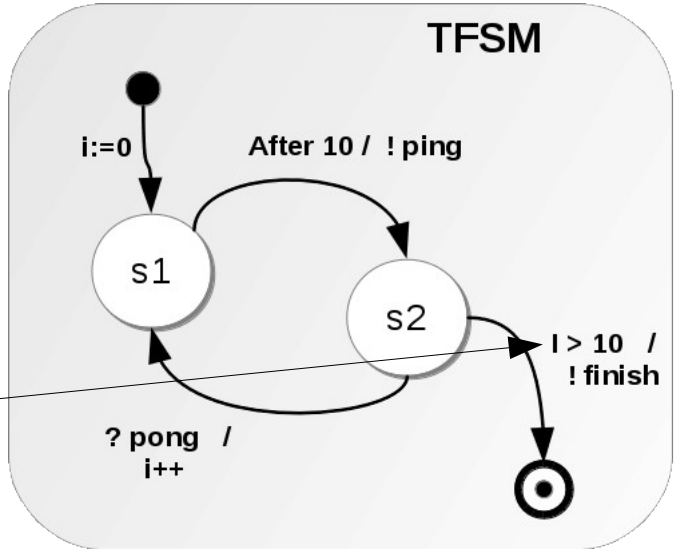
look at a simple example

- Data “management”
  - Structuring
  - Computing
  - …



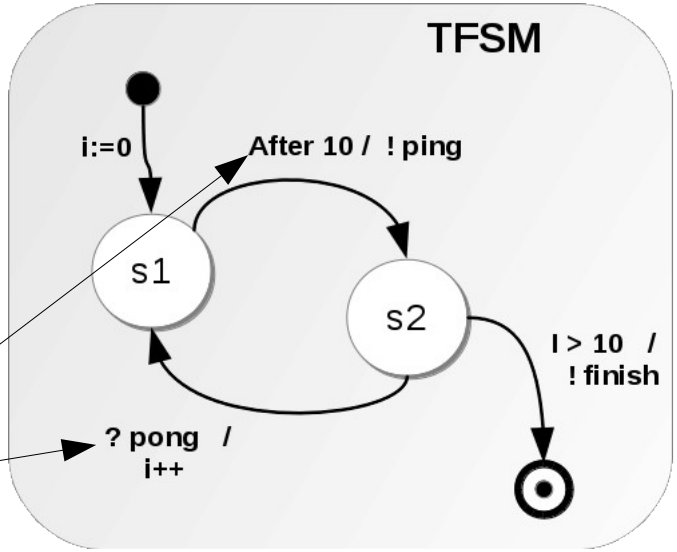
look at a simple example

- Data “management”
  - Structuring
  - Computing
  - …
- Conditional control
  - If (boolean condition)



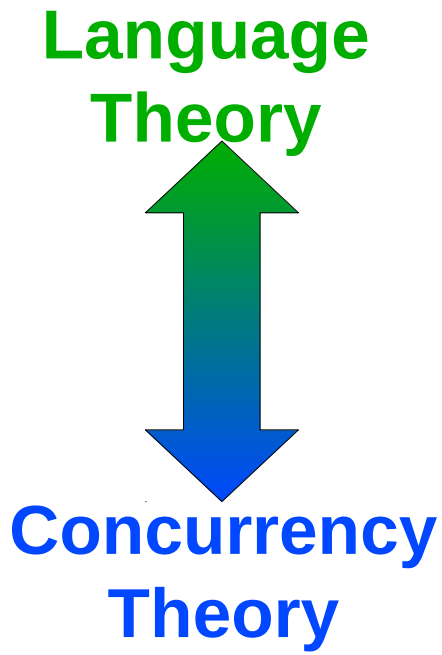
look at a simple example

- Data “management”
  - Structuring
  - Computing
  - …
- Conditional control
  - If (boolean condition)
  - When
    - Internal control (tokens)
    - External control (events)



look at a simple example

- Data “management”
  - Structuring
  - Computing
  - …
- Conditional control
  - If (boolean condition)
  - When
    - *Internal* control
    - *External* control

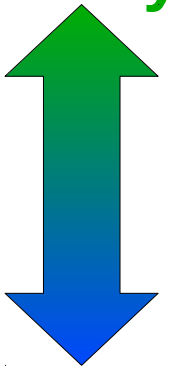


from what ingredients the semantic domain is made up with ?

- “sequential” semantics specification
  - Operational
  - Axiomatic
  - Translational
  - …
- Concurrency specification
  - Tagged signal
  - Event structure
  - Model of Computation
  - …

Specifies evolution of the model state

Language Theory

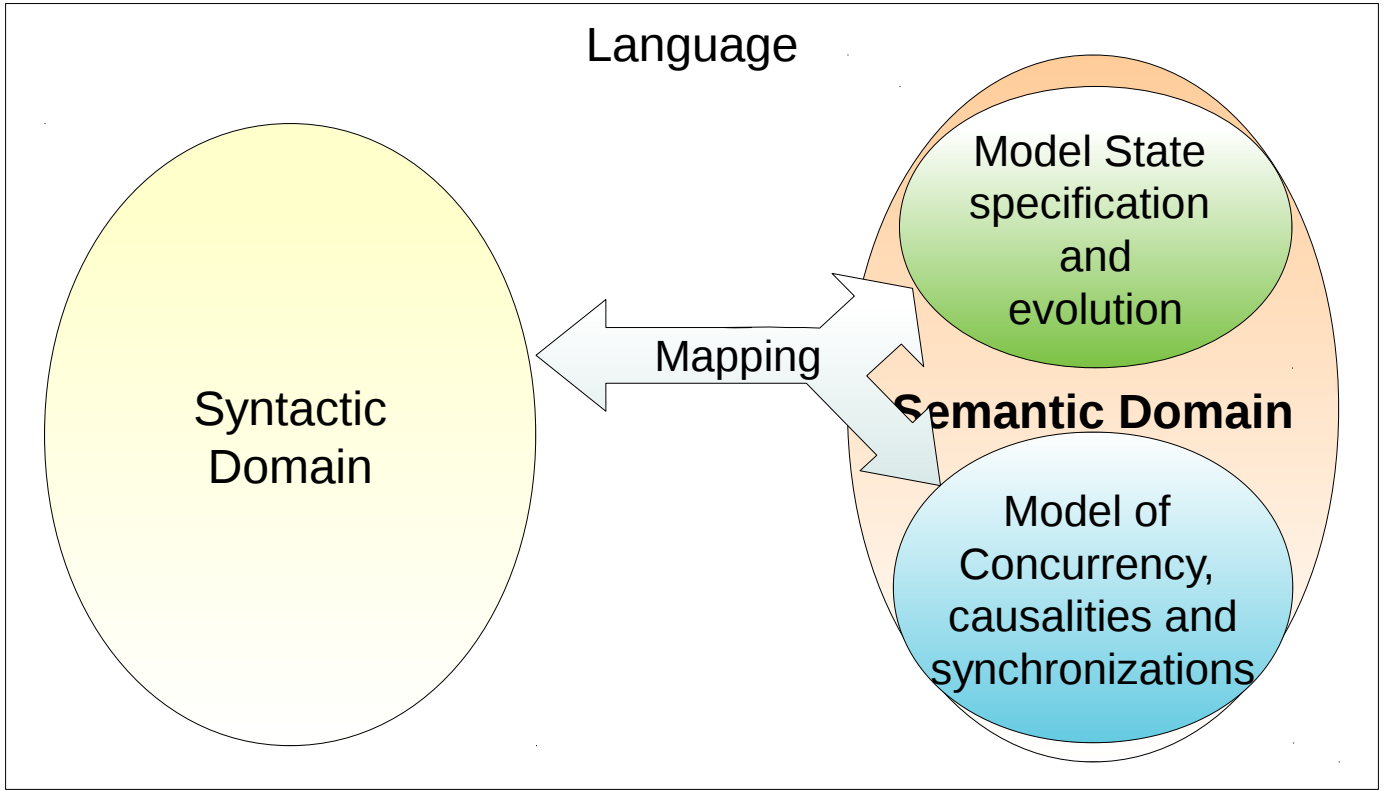


Concurrency Theory

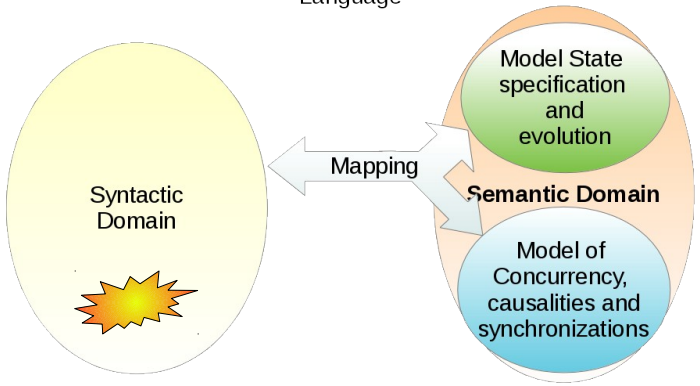
Specifies (possibly timed) causal relations and synchronizations

ANR-12-INSE-0011 GEMOC proposition

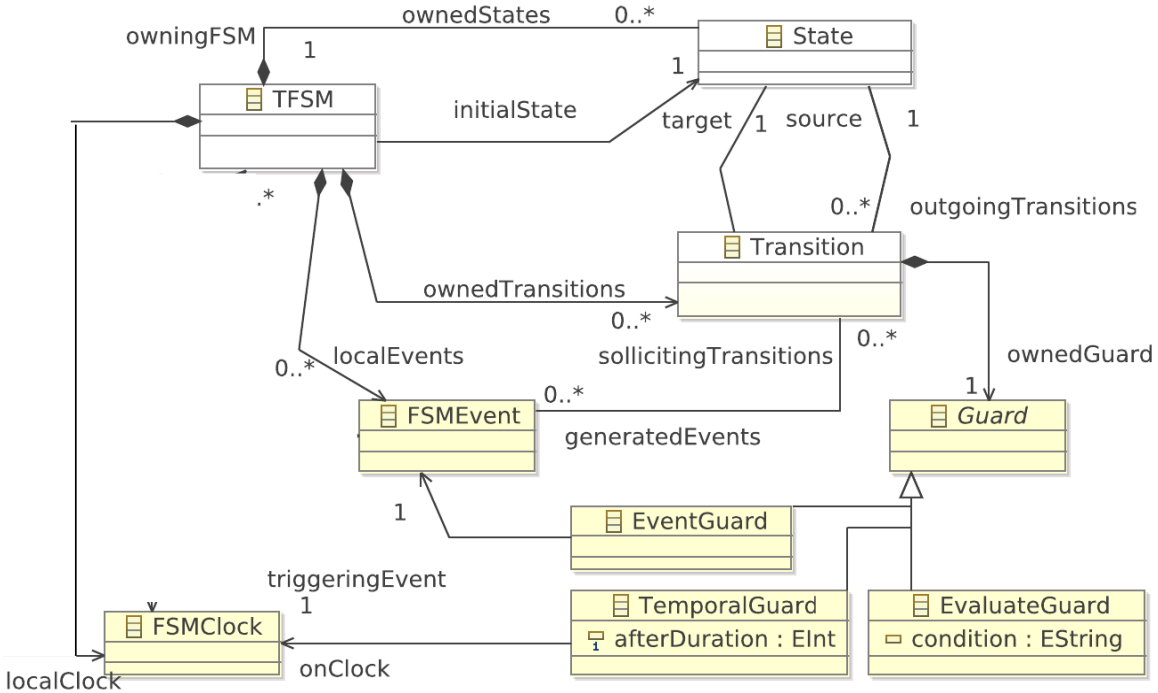
- Providing meta-languages for the semantic specification

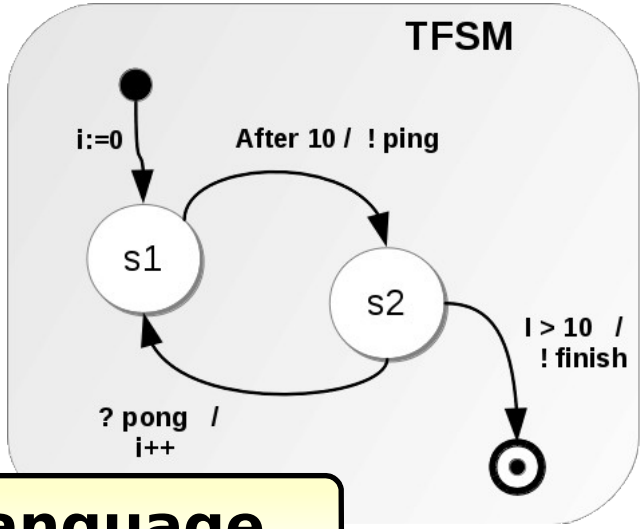


Meta-languages for executable modeling



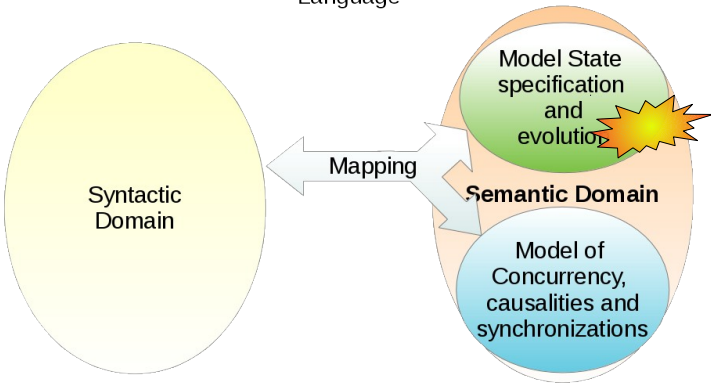
DSML $\stackrel{def}{=}$ < **Abstract Syntax**, semantics >



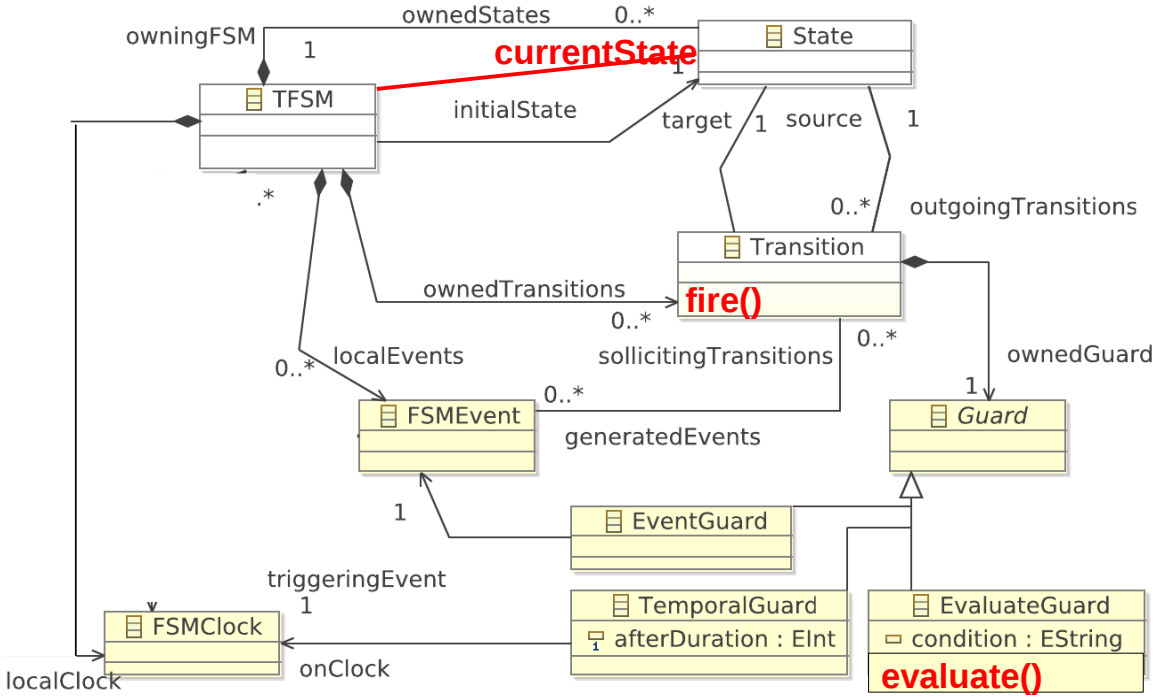


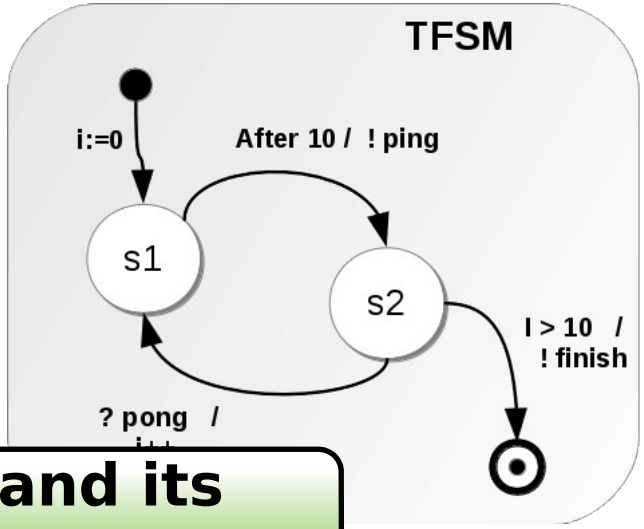
Specify the grammar of the language

Meta-languages for executable modeling



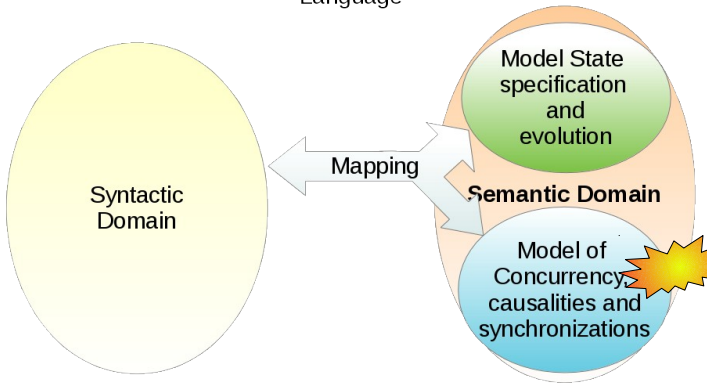
DSML $\stackrel{def}{=}$ < AS, Domain Specific Actions, …, … >



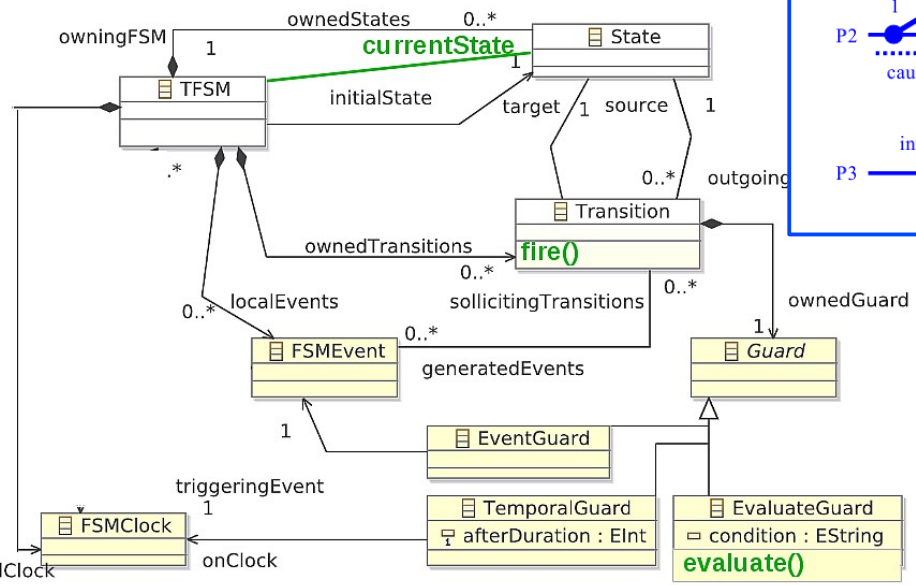


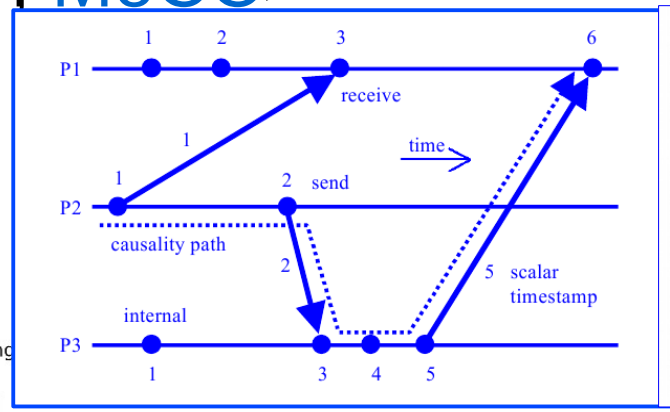
Specify the execution state and its evolution

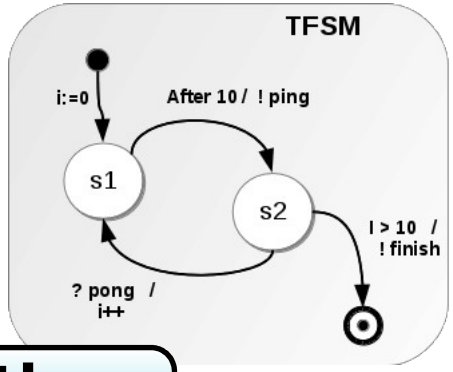
Meta-languages for executable modeling



DSML $\stackrel{def}{=}$ < AS, DSA, …, MoCC>

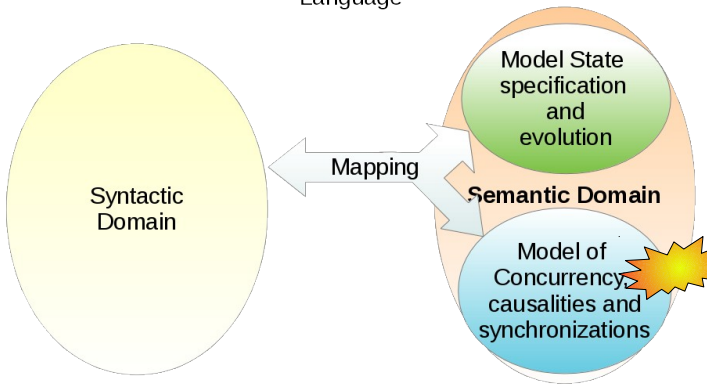




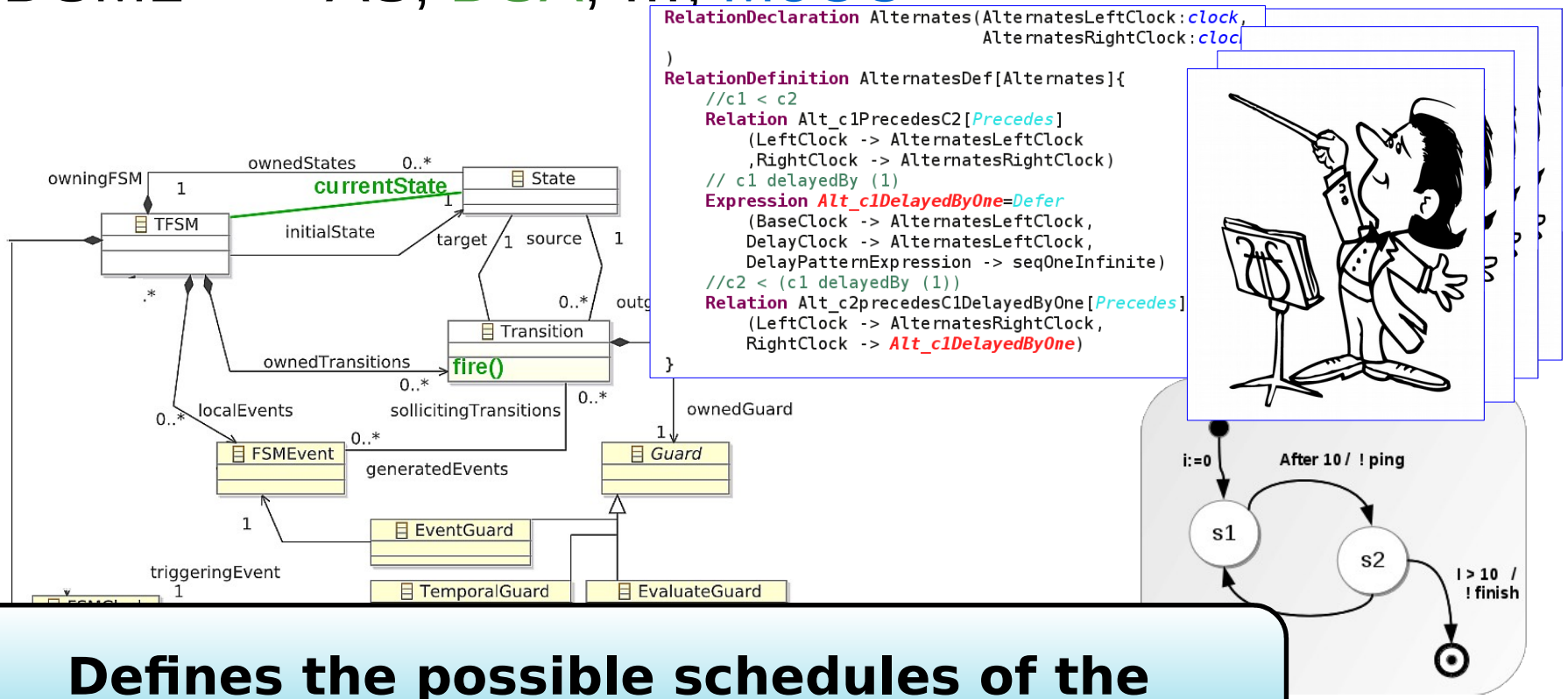


Defines the possible schedules of the previously defined actions

Meta-languages for executable modeling

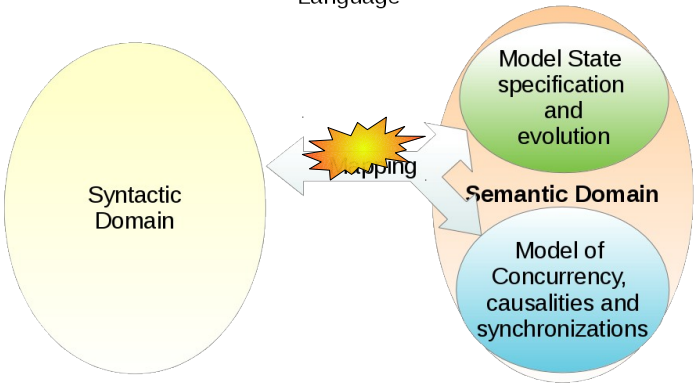


DSML $\stackrel{def}{=}$ < AS, DSA, …, MoCC>



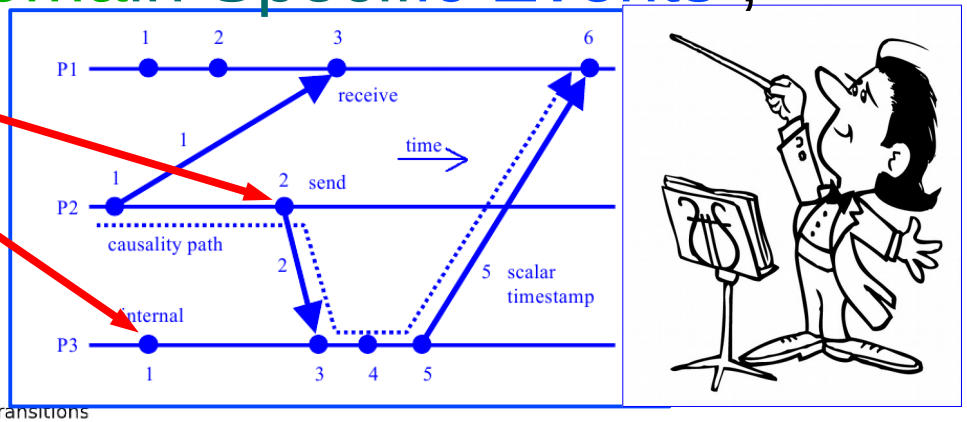
Defines the possible schedules of the previously defined actions in a symbolic way

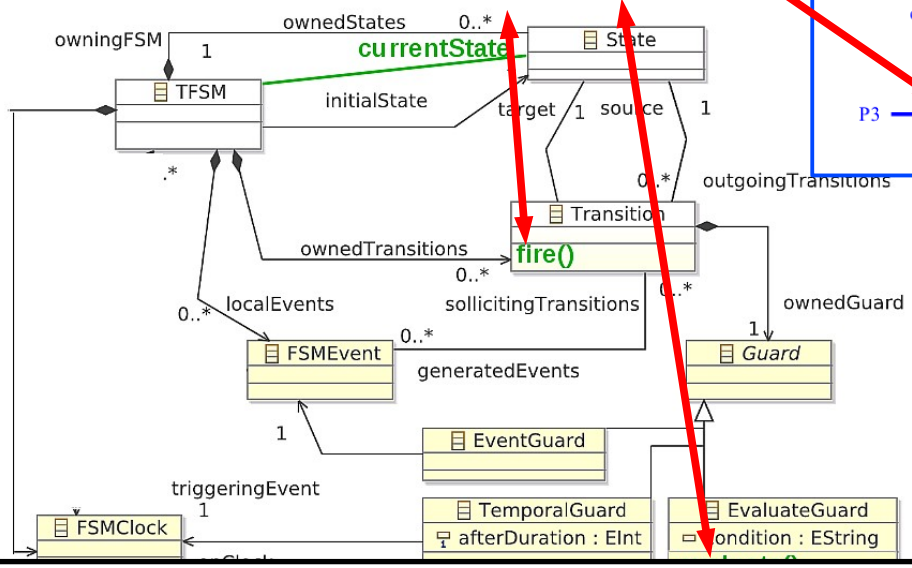
Meta-languages for executable modeling

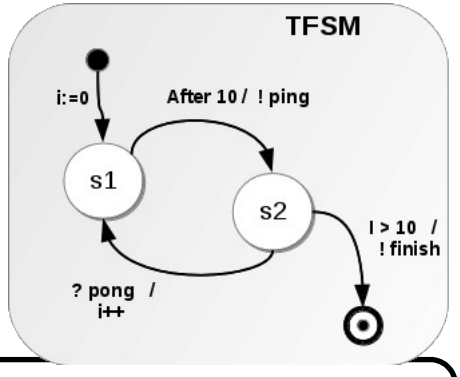


DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoCC >

Domain Specific Events

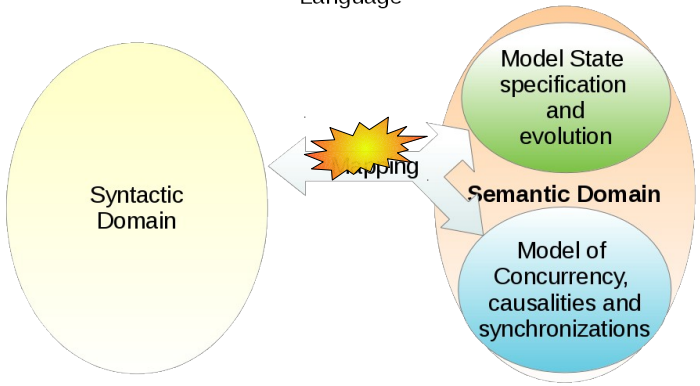






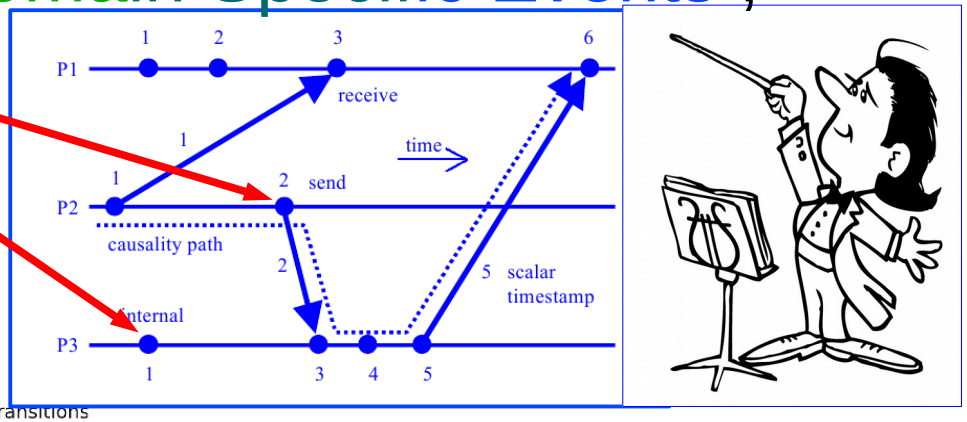
Realizes a mapping between the event from the MoCC and the action calls and returns

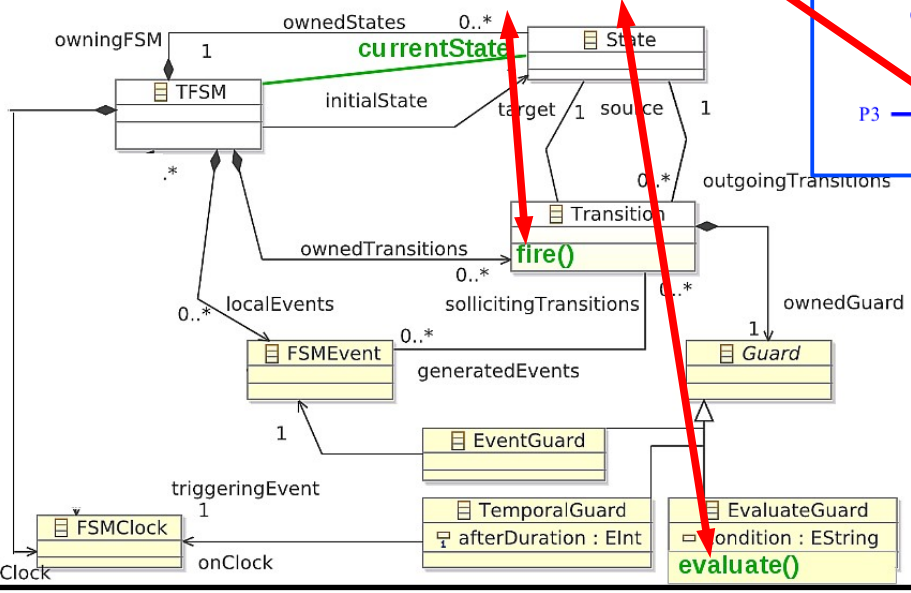
Meta-languages for executable modeling

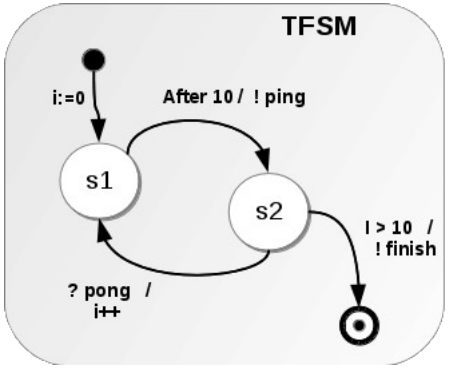


DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoCC >

Domain Specific Events

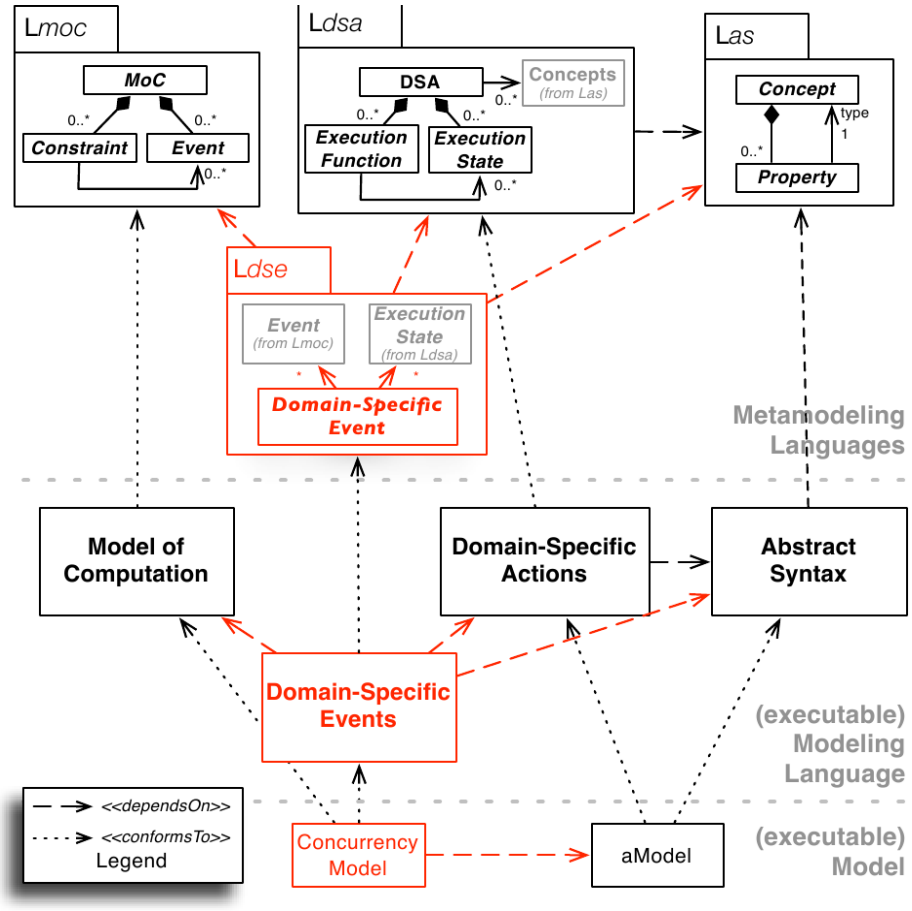






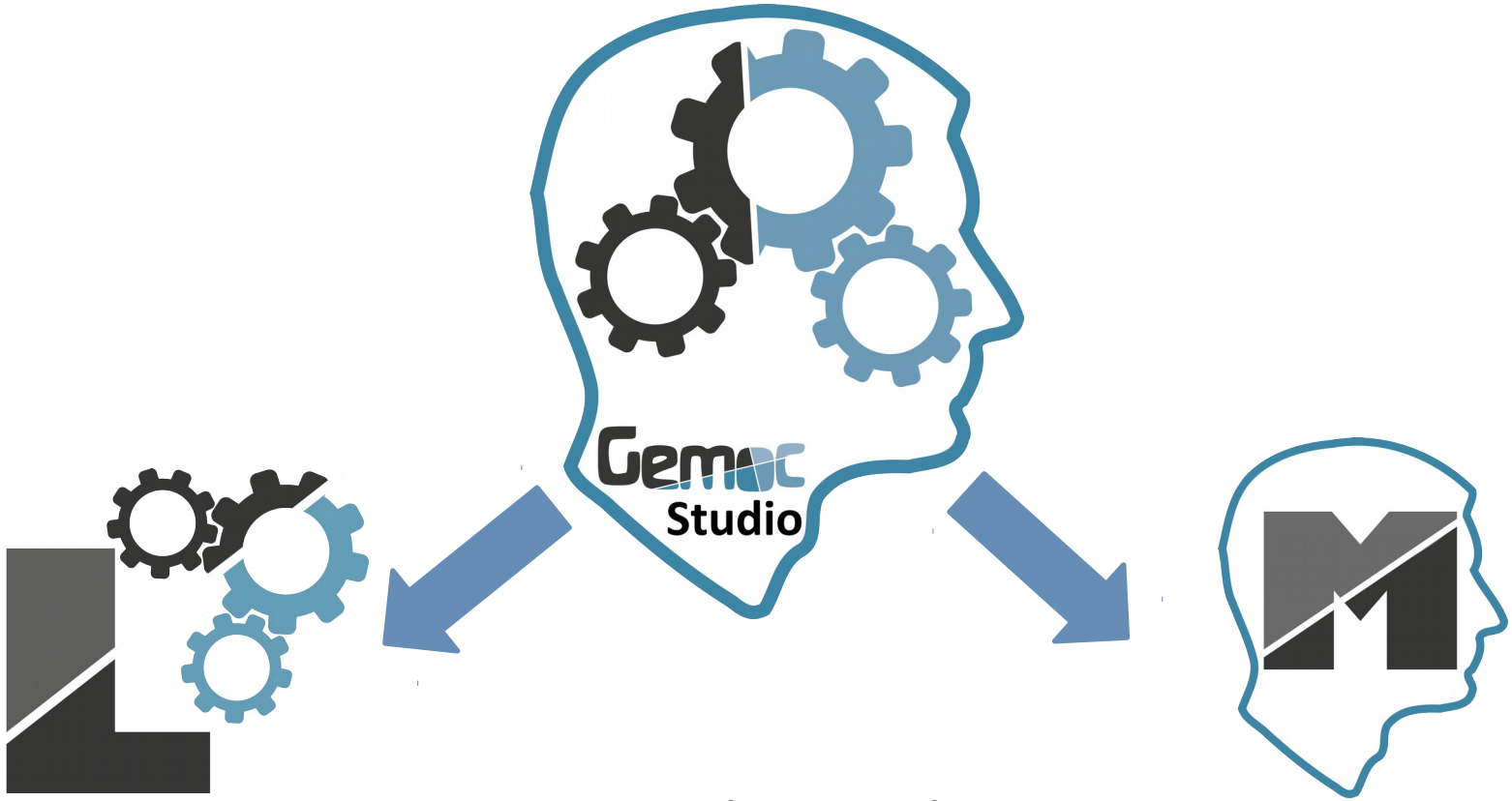
Event Constraint Language from inria Aoste team
(http://hal.inria.fr/hal-00721169)

Meta-languages for executable modeling

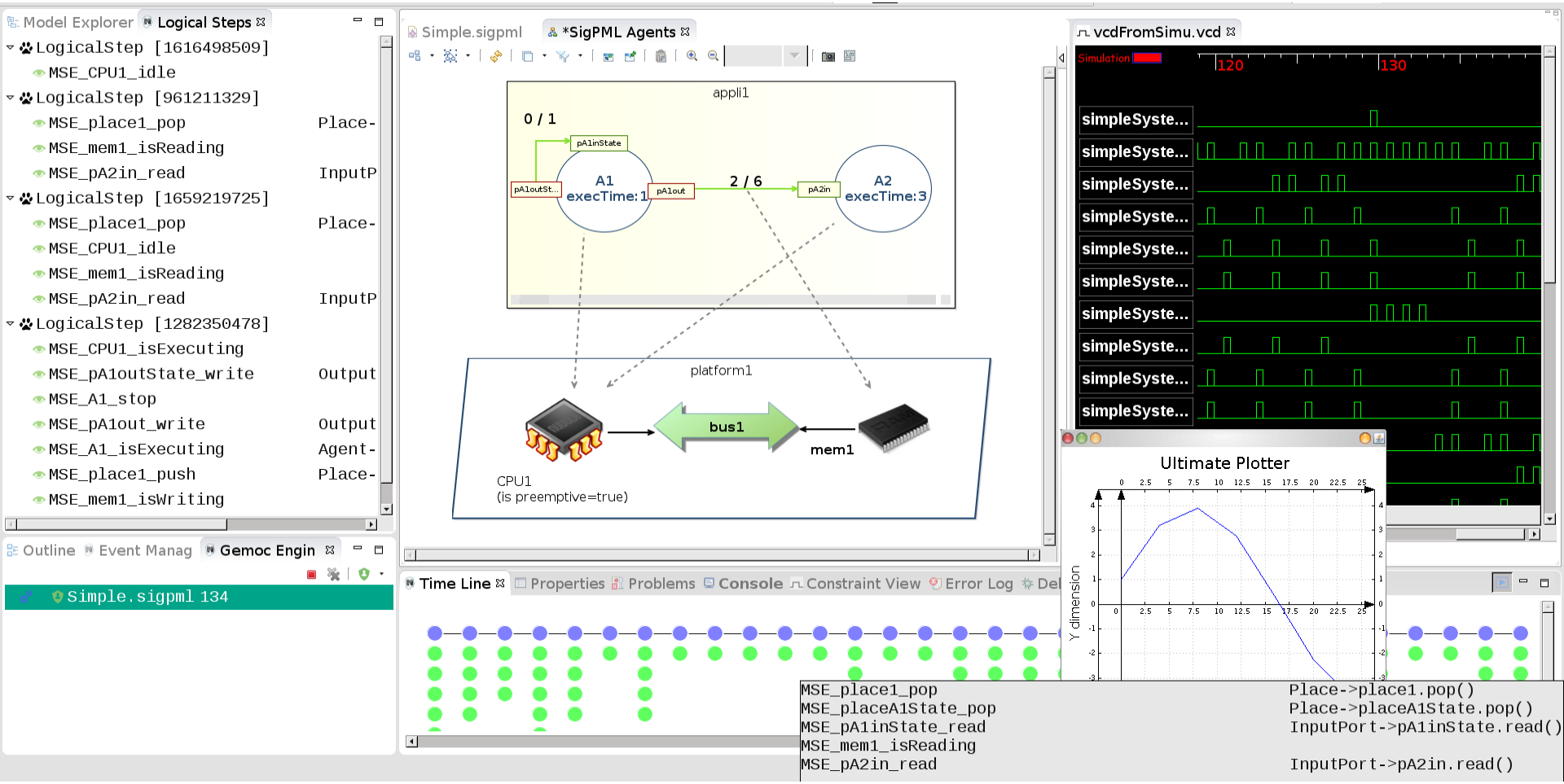


Reifying Concurrency for Executable Metamodeling (Benoit Combemale, Julien Deantoni, Matias Vara Larsen, Frédéric Mallet, Olivier Barais, Benoit Baudry, Robert France), In 6th International Conference on Software Language Engineering (SLE 2013) (Richard F. Paige Martin Erwig, Eric van Wyk, eds.), Springer-Verlag, 2013. [bibtex] [pdf]

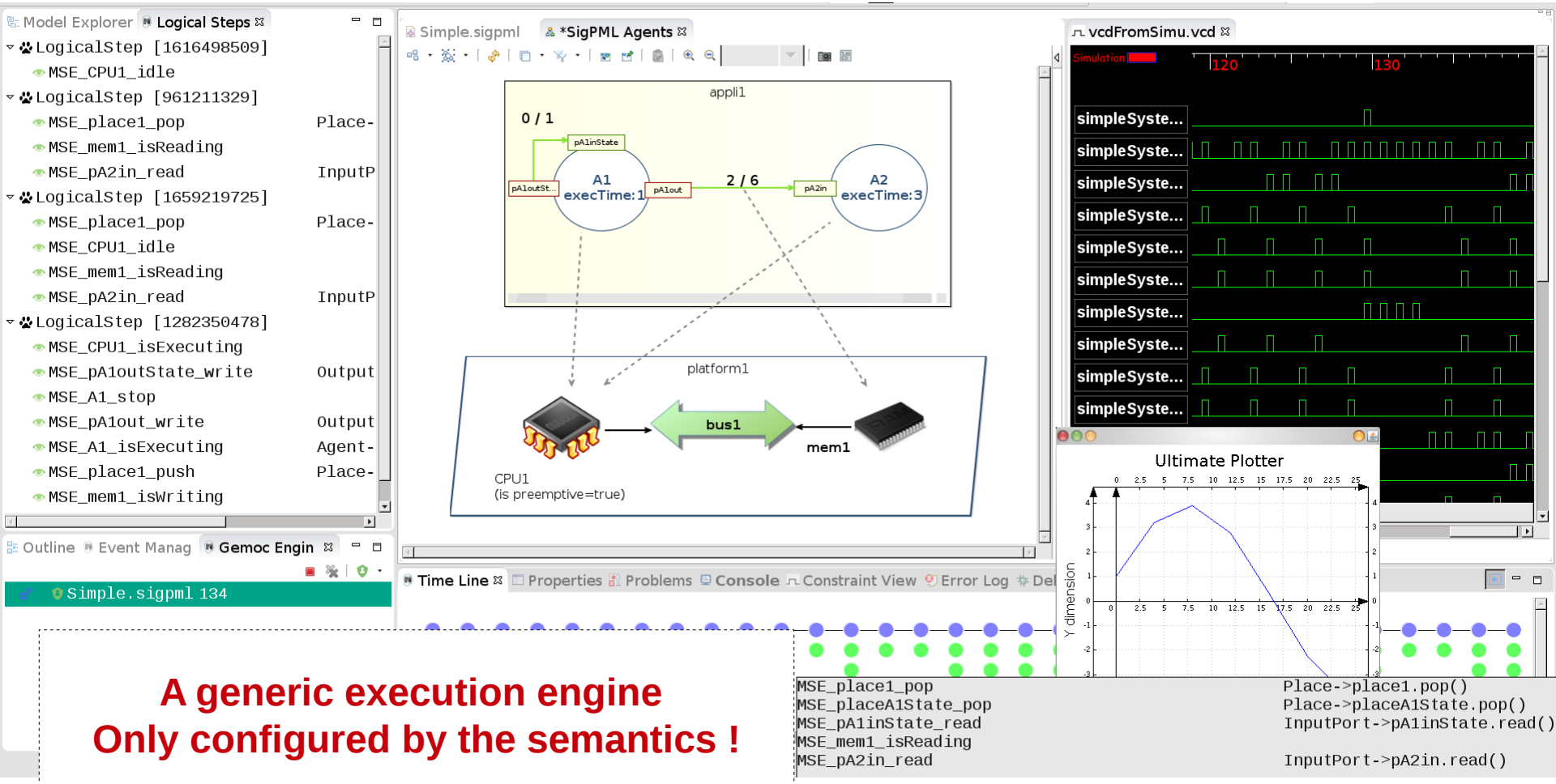
The GEMOC Studio



Modeling workbench: http://gemoc.org/studio/

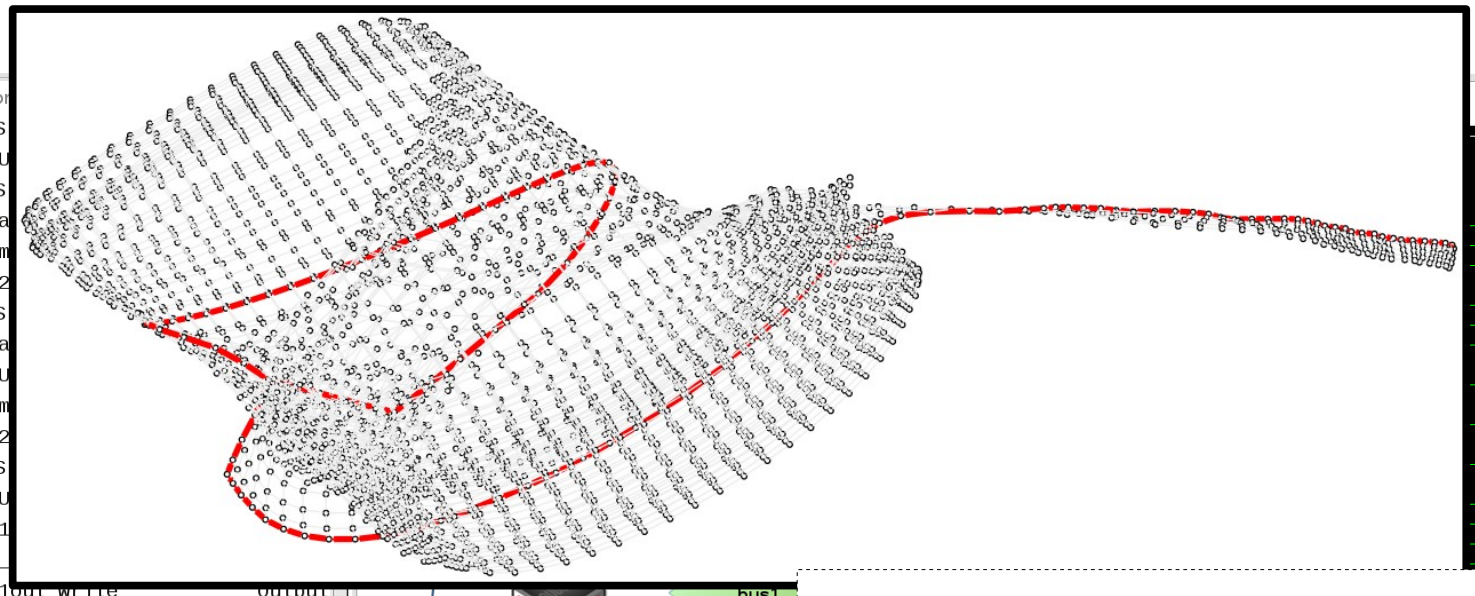


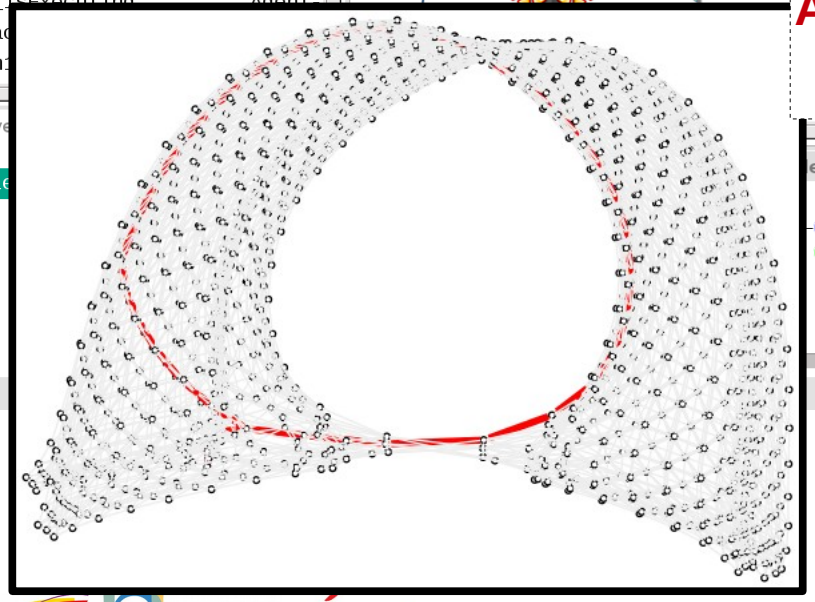
Modeling workbench: http://gemoc.org/studio/



A generic execution engine
Only configured by the semantics !

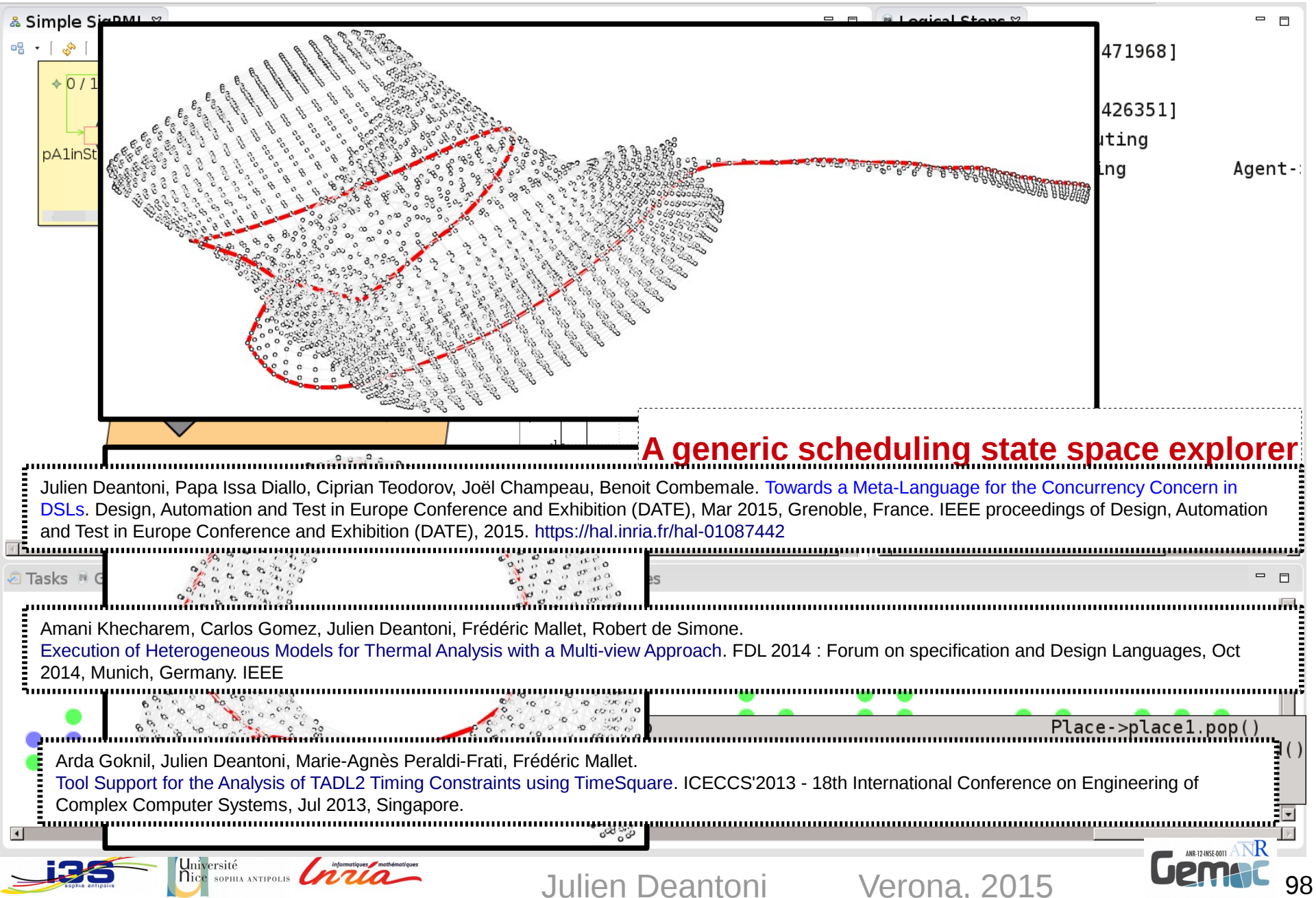
Modeling workbench: http://gemoc.org/studio/





A generic scheduling state space explorer
Only configured by the semantics !

Modeling workbench: http://gemoc.org/studio/



A generic scheduling state space explorer

Julien Deantoni, Papa Issa Diallo, Ciprian Teodorov, Joël Champeau, Benoit Combemale. Towards a Meta-Language for the Concurrency Concern in DSLs. Design, Automation and Test in Europe Conference and Exhibition (DATE), Mar 2015, Grenoble, France. IEEE proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE), 2015. https://hal.inria.fr/hal-01087442

Amani Khecharem, Carlos Gomez, Julien Deantoni, Frédéric Mallet, Robert de Simone.
Execution of Heterogeneous Models for Thermal Analysis with a Multi-view Approach. FDL 2014 : Forum on specification and Design Languages, Oct 2014, Munich, Germany. IEEE

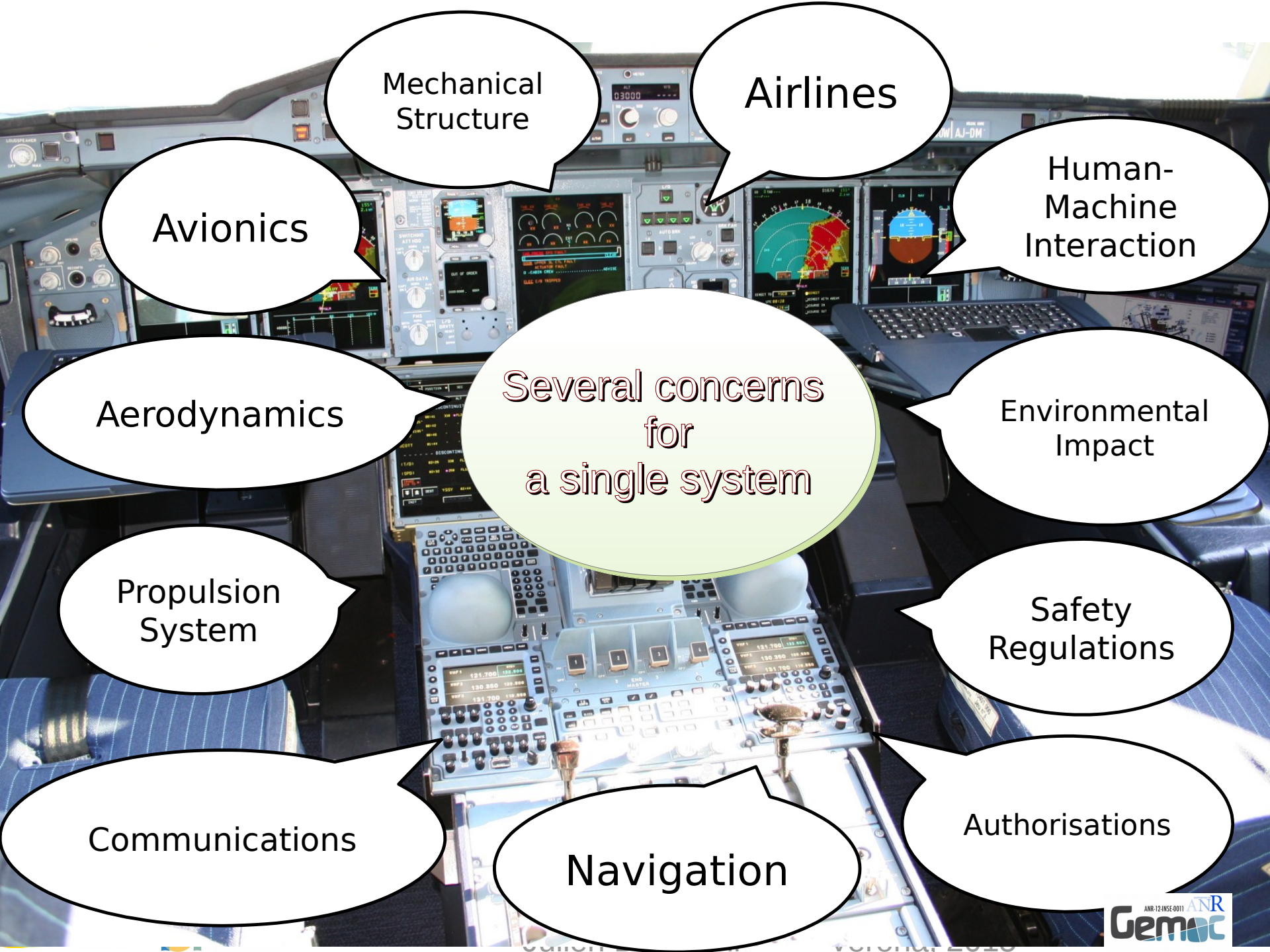
Arda Goknil, Julien Deantoni, Marie-Agnès Peraldi-Frati, Frédéric Mallet.
Tool Support for the Analysis of TADL2 Timing Constraints using TimeSquare. ICECCS'2013 - 18th International Conference on Engineering of Complex Computer Systems, Jul 2013, Singapore.

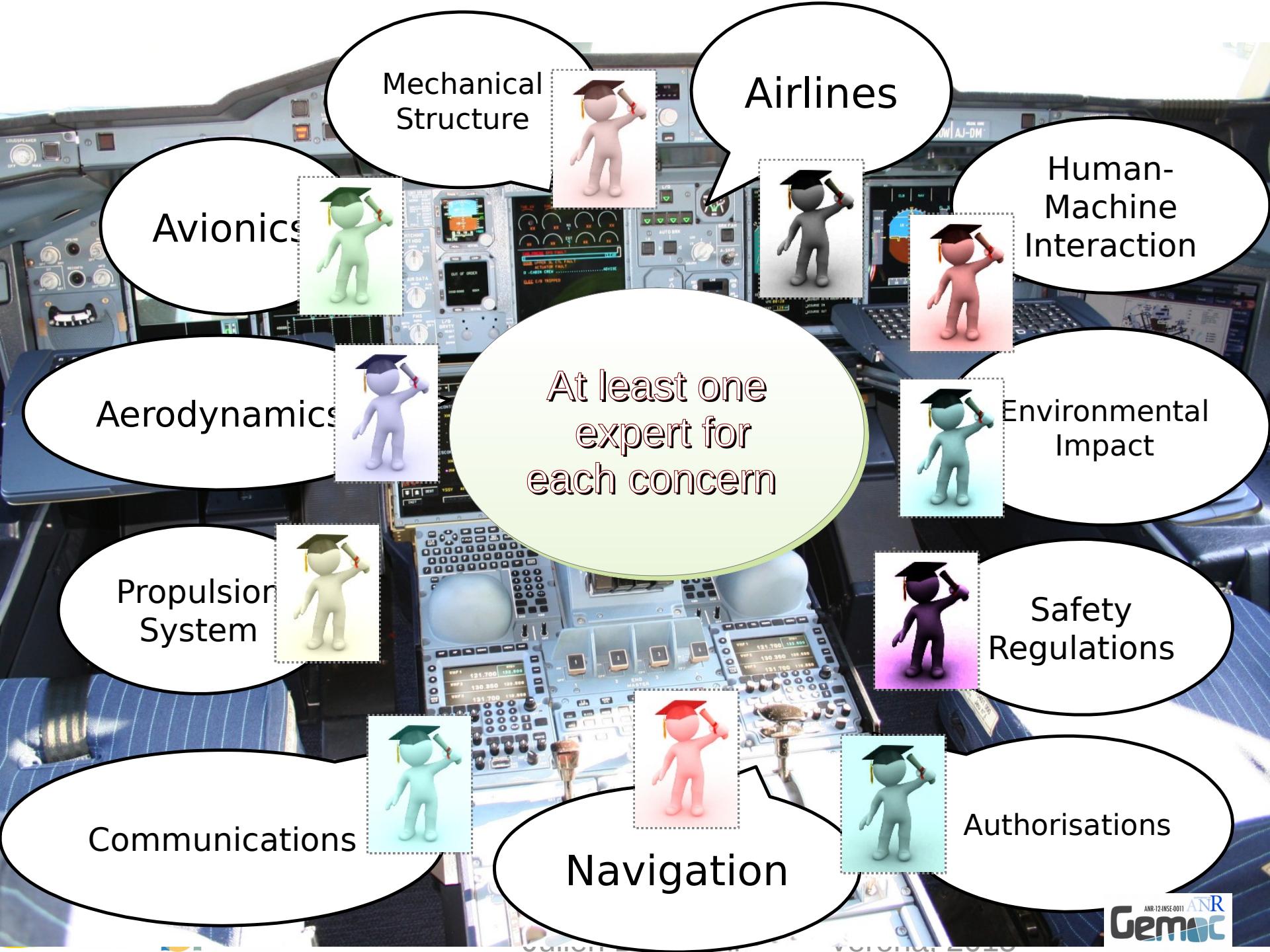
We were able to define the behavioral semantics of a single language

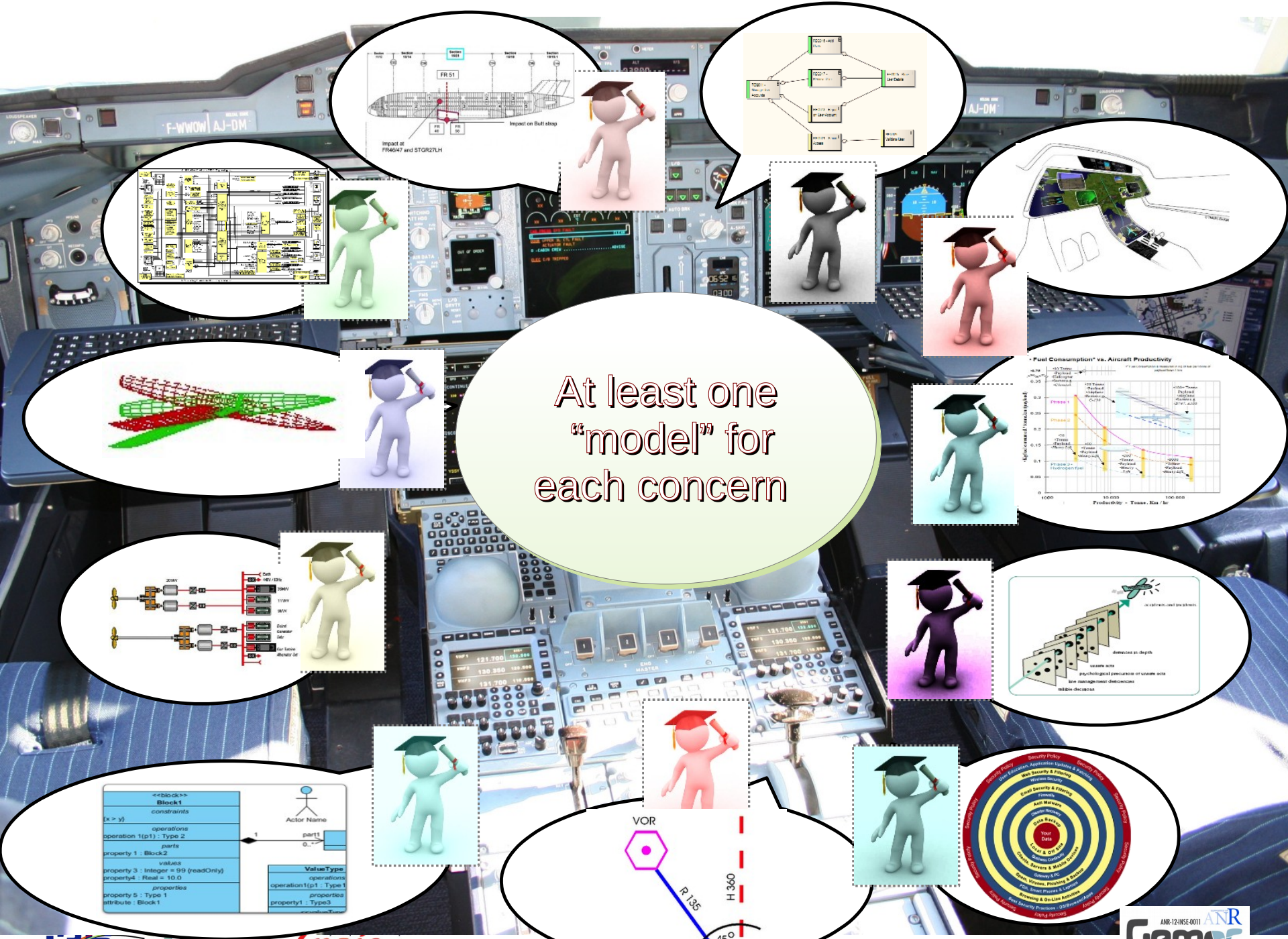
- Good but...
  - The previous definition only address a single language...
- What about system modeling and heterogeneous execution ?

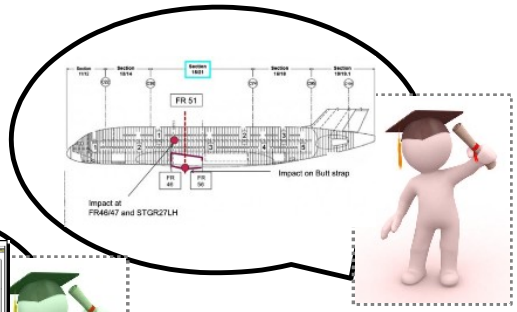
Globalizing Modeling Languages.
Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert France, Jean-Marc Jézéquel, Jeff Gray.
Computer, IEEE, 2014, pp. 10-13

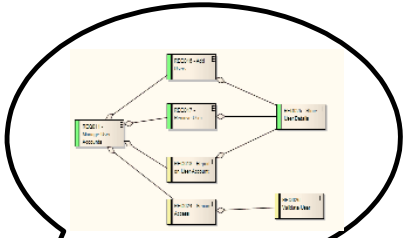

F-WWOW AJ-DM
F-WWOW AJ-DM

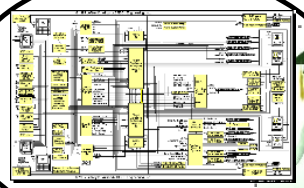

Mechanical Structure
Airlines
Avionics
Human-Machine Interaction
Several concerns for a single system
Aerodynamics
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorisations

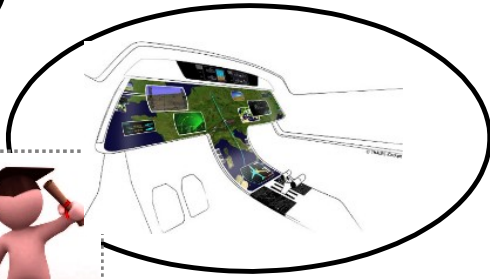

Mechanical Structure
Airlines
Avionics
Human-Machine Interaction
Aerodynamics
At least one expert for each concern
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorisations

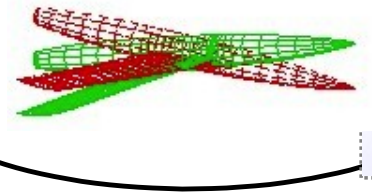

At least one "model" for each concern
Impact at FR46/47 and STGR27LH
Impact on Butt strap
Fuel Consumption* vs. Aircraft Productivity
Productivity - Tonne. Km / hr
<<block>>
Block1
constraints
{x > y}
operations
operation 1(p1) : Type 2
parts
property 1 : Block2
values
property 3 : Integer = 99 {readOnly}
property4 : Real = 10.0
properties
property 5 : Type 1
attribute : Block1
Actor Name
part1
ValueType
operations
operation1(p1 : Type1
properties
property1 : Type3
VOR
R 135
H 360
45°

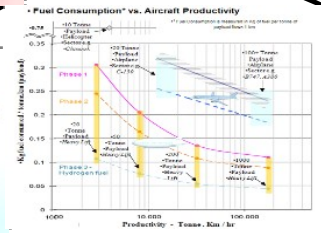


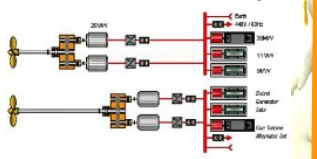


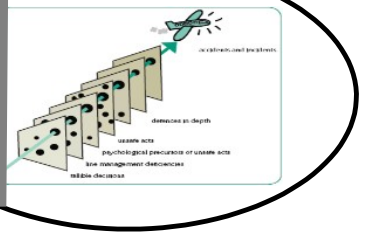


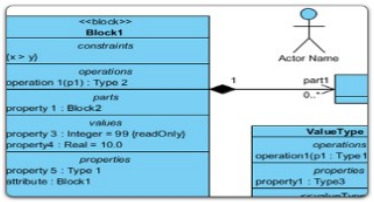


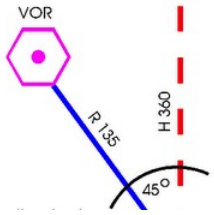


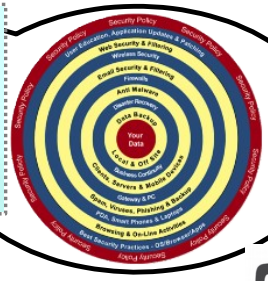






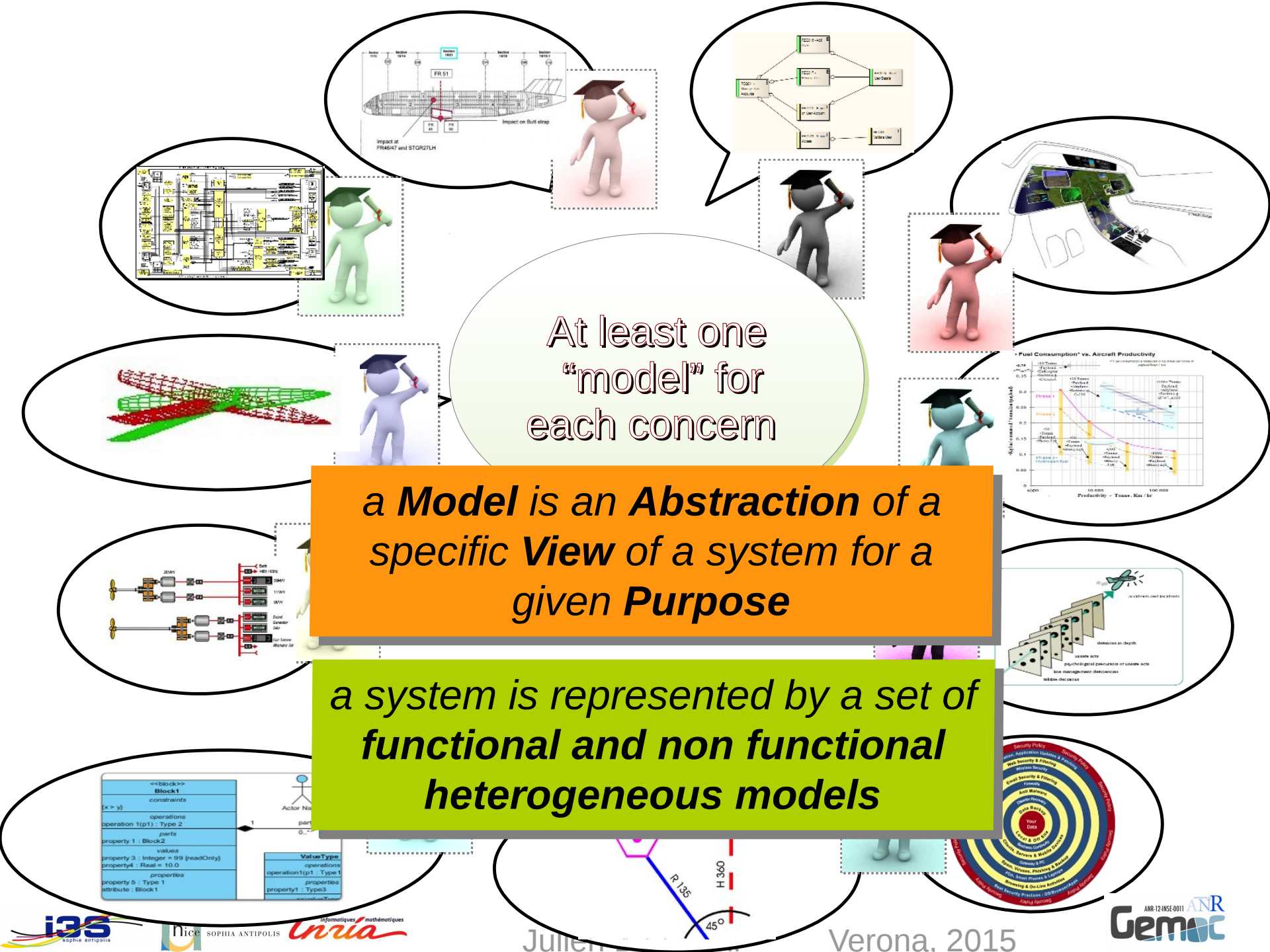


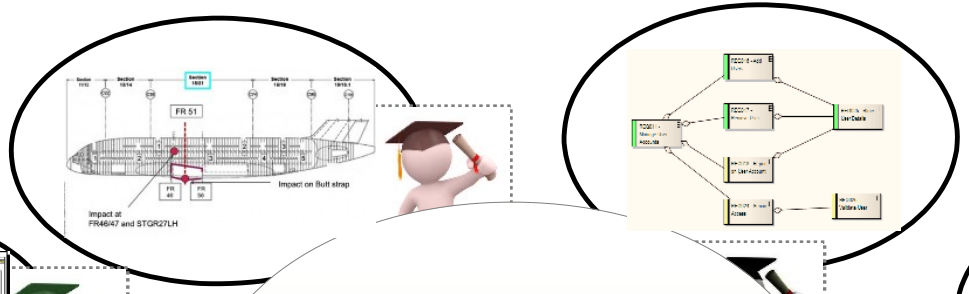
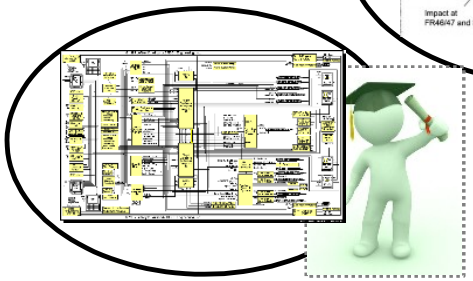


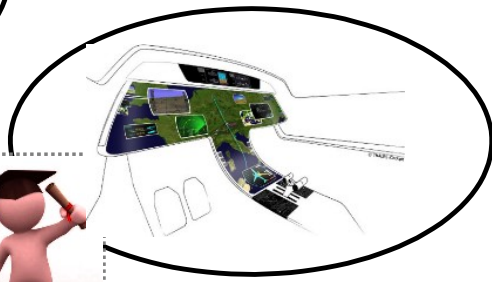


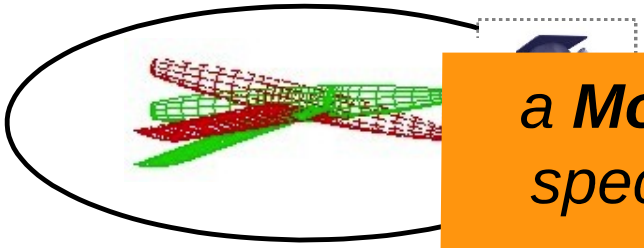
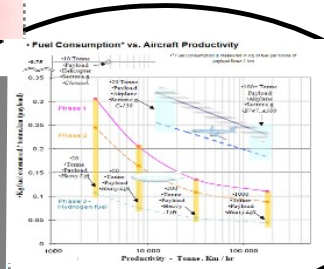
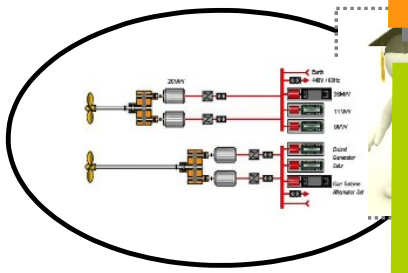
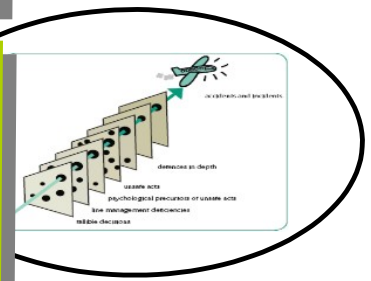
At least one "model" for each concern

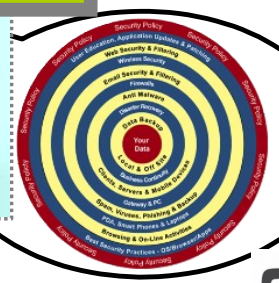
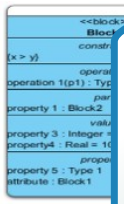
*a **Model** is an **Abstraction** of a specific **View** of a system for a given **Purpose***


At least one "model" for each concern
a **Model** is an **Abstraction** of a specific **View** of a system for a given **Purpose**
a system is represented by a set of **functional and non functional heterogeneous models**



At least one "model" for each concern

*a **Model** is an **Abstraction** of a specific **View** of a system for a given **Purpose***

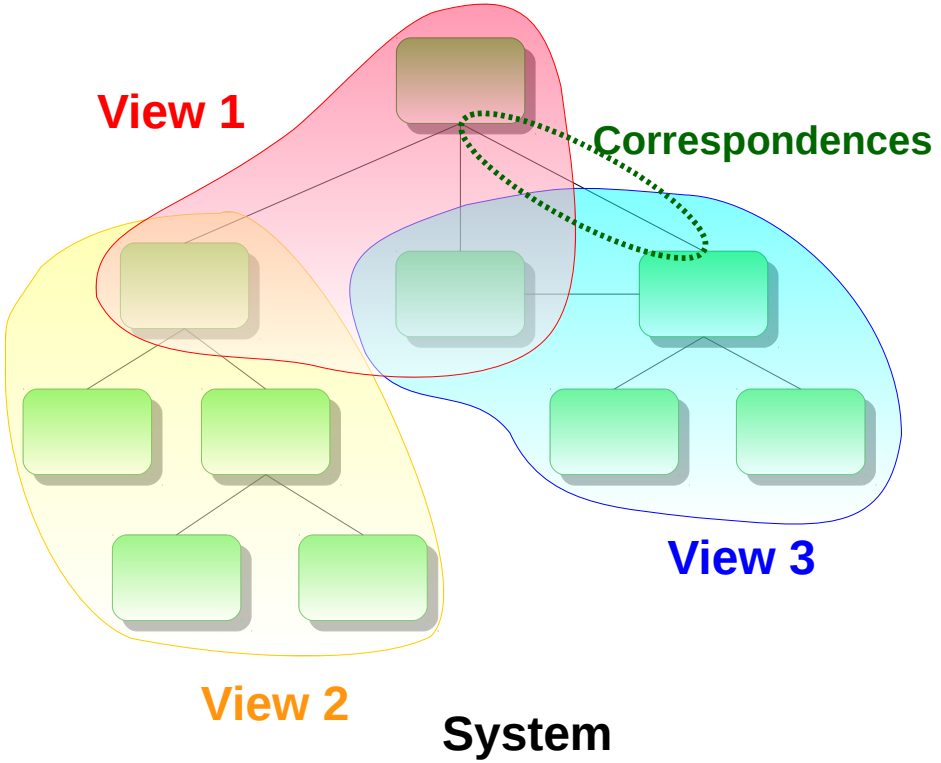
*a system is represented by a set of **functional and non functional heterogeneous models***

Heterogenous because they have different **Model of Computation**

Multi-View Approaches

- IEEE-42010 (updated version of IEEE-1401)

- **The System is composed of different views**
- **The view are expressed in domain specific languages**
- **Each domain specific language have its own behavioral semantics (and usually its own tooling)**

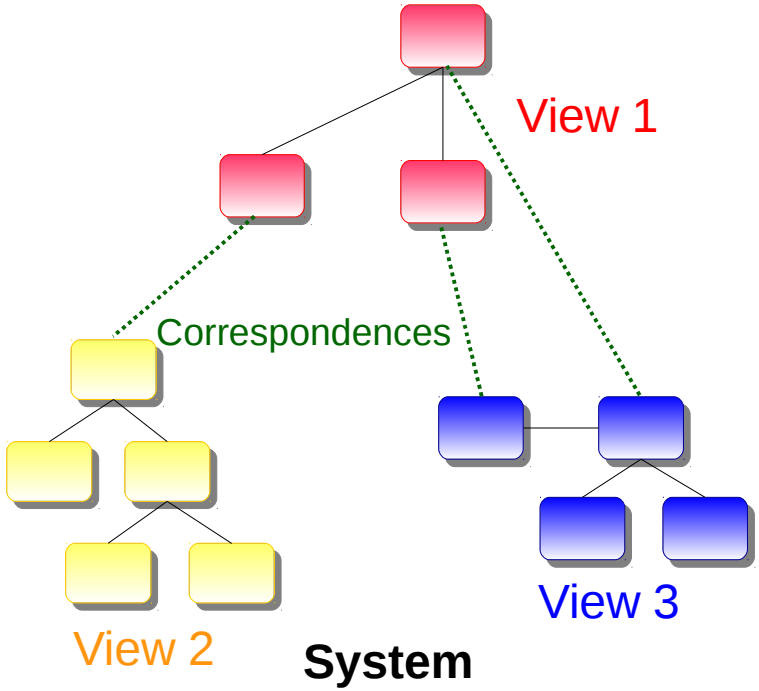


[M. Nassar, "*VUML: a viewpoint oriented UML extension*", 2003]

Multi-View Approaches

- IEEE-42010 (updated version of IEEE-1401)

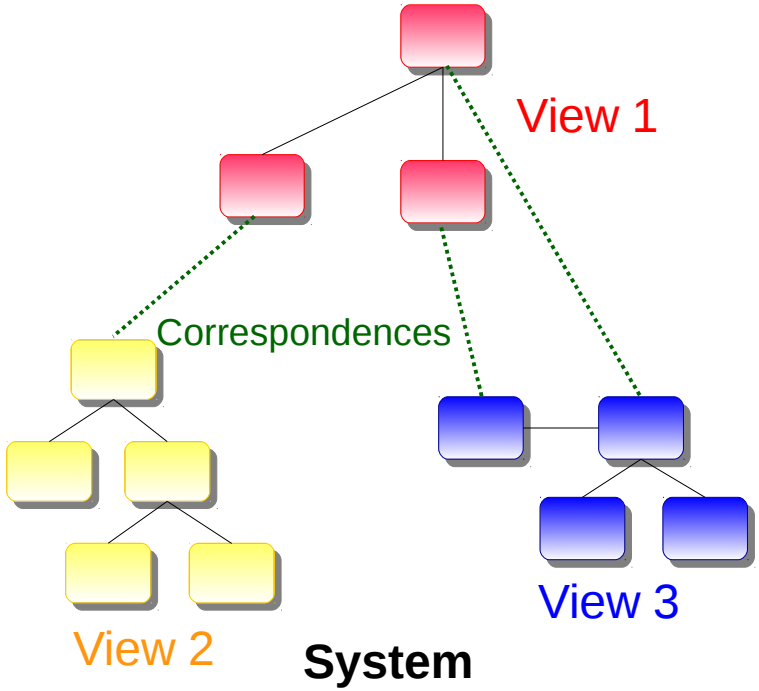
- **The System is composed of different views**
- **The view are expressed in domain specific languages**
- **Each domain specific language have its own behavioral semantics (and usually its own tooling)**



Multi-View Approaches

- IEEE-42010 (updated version of IEEE-1401)

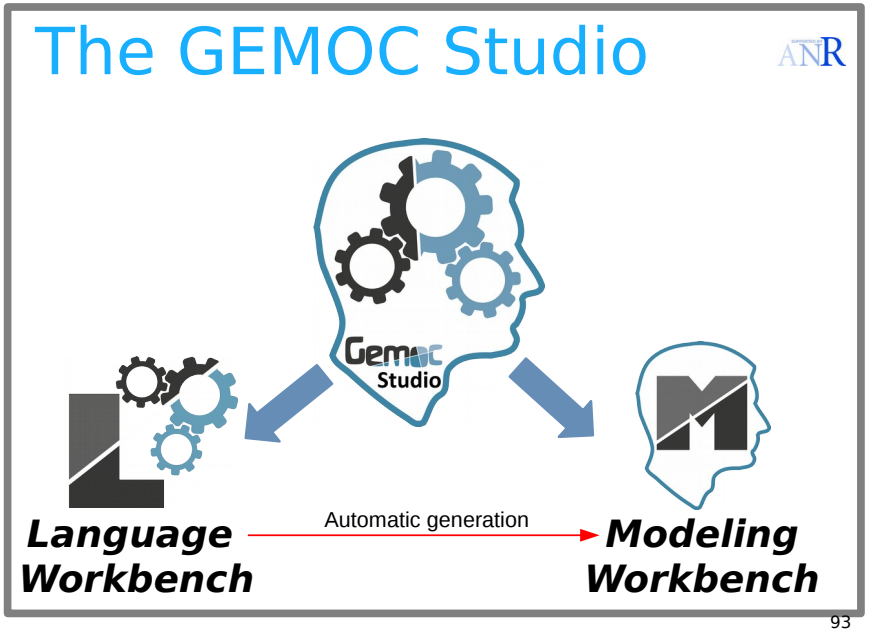
- **The System is composed of different views**
- **The view are expressed in domain specific languages**
- **Each domain specific language have its own behavioral semantics (and usually its own tooling)**

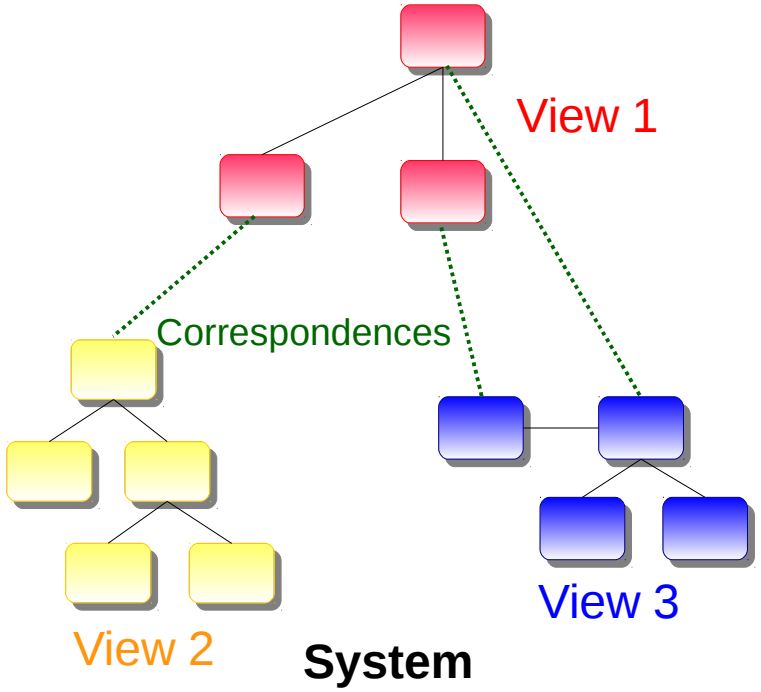


There is a need to coordinate/orchestrate all these model executions to understand the global behavior of the system

Multi-View Approaches

- IEEE-42010 (updated version of IEEE-1401)



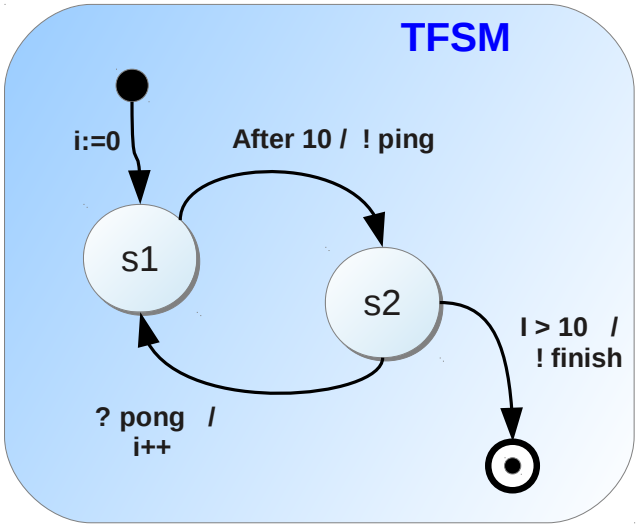


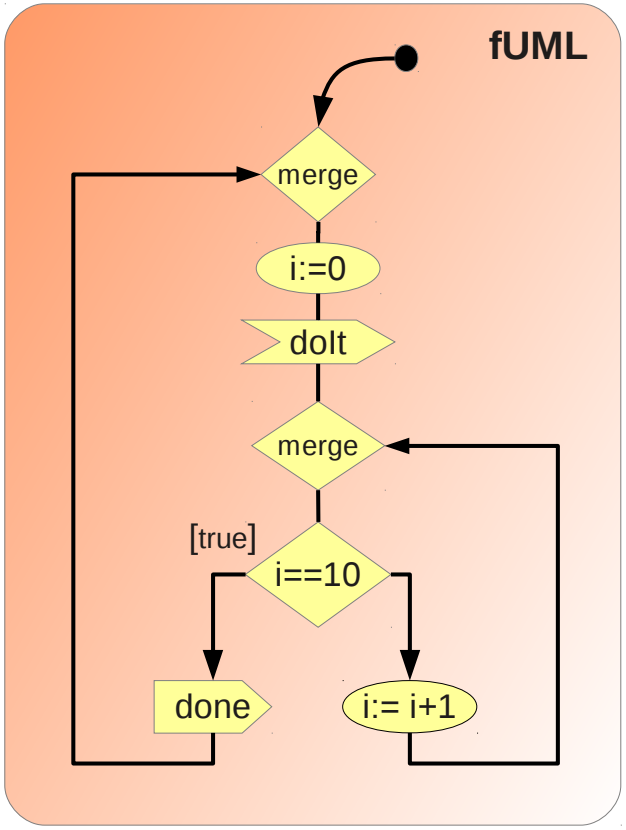
There is a need to coordinate/orchestrate all these model executions to understand the global behavior of the system

By taking advantage of explicit behavioral semantics

model behavioral coordination

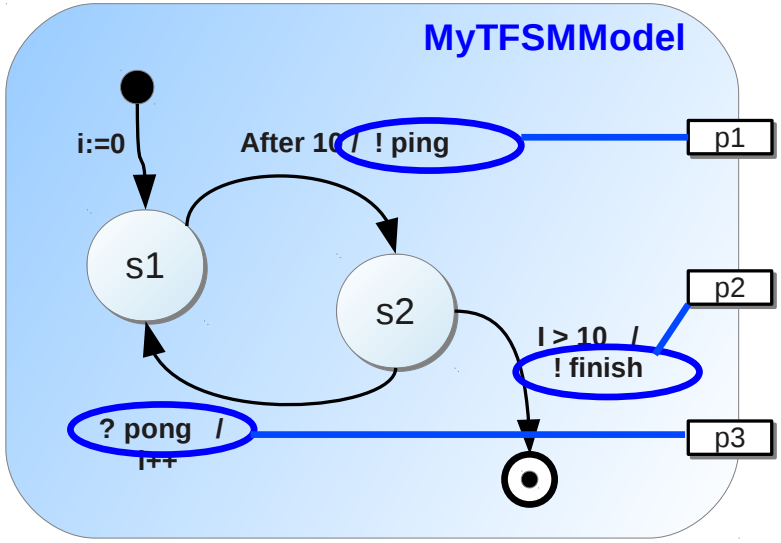
- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoCC >

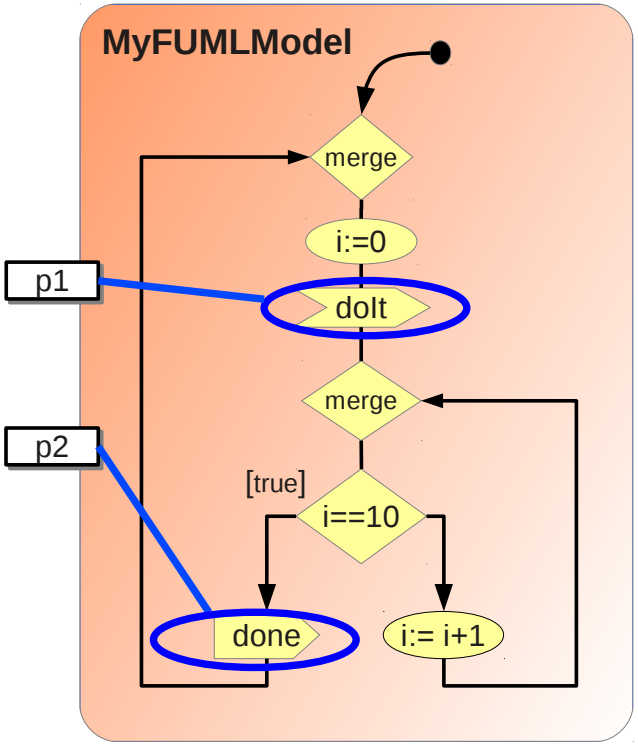




model behavioral coordination

- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >



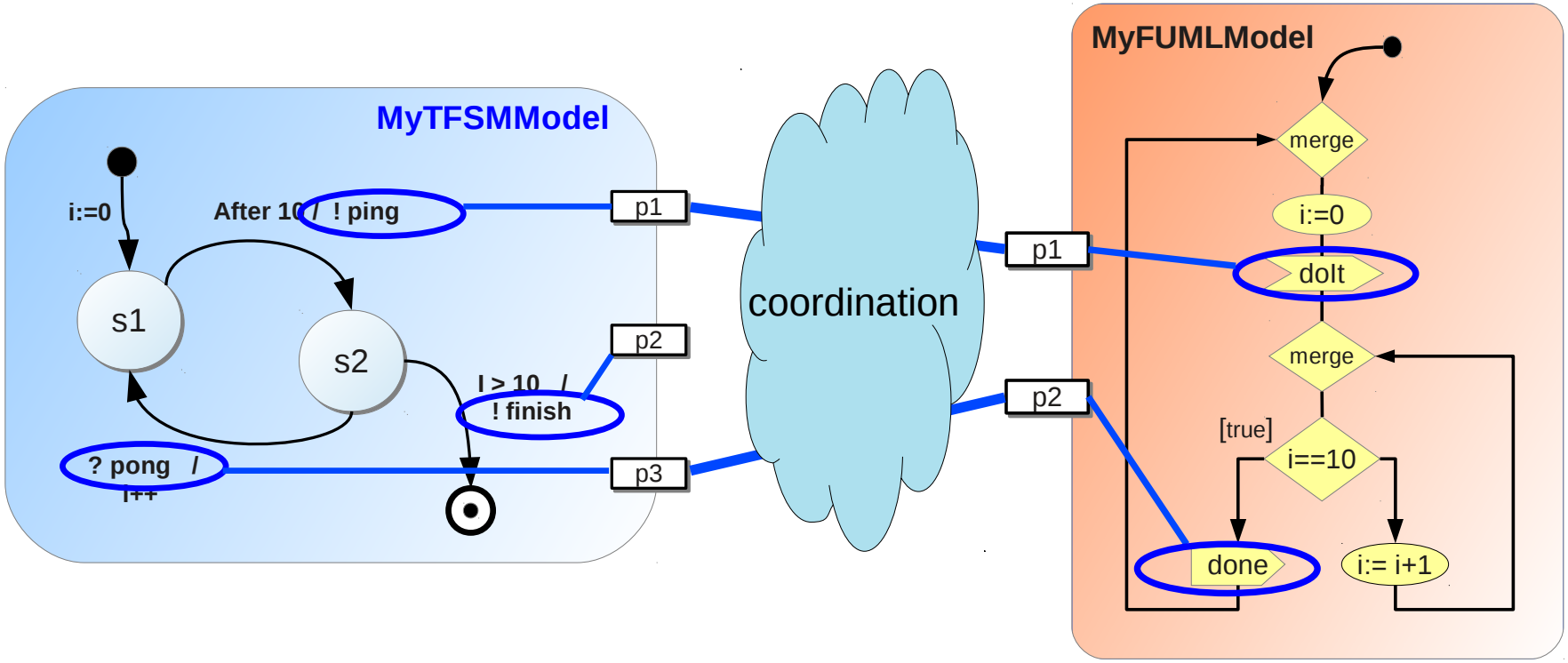


Automatically extracted model interface

(Instances of Domain Specific Events acts as coordination interfaces)

model behavioral coordination

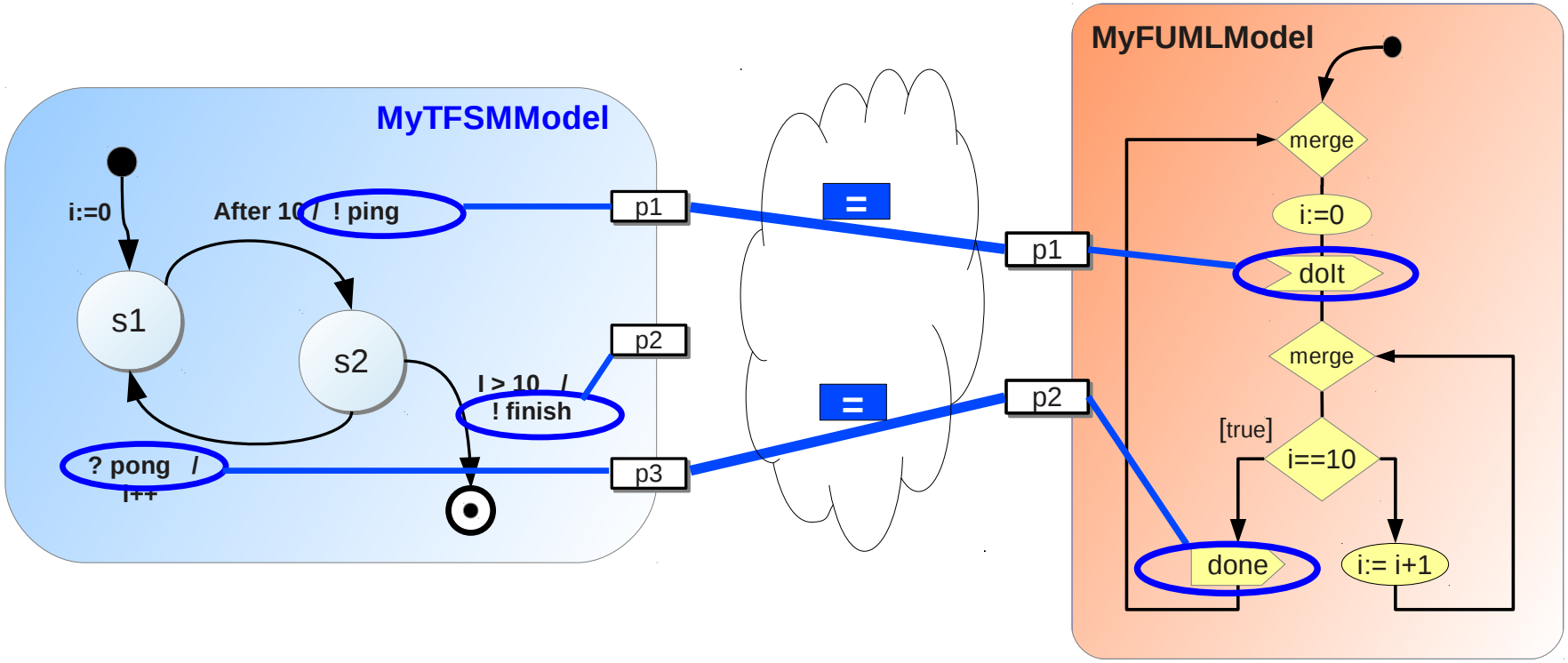
- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >



coordination interfaces synchronizes interface events

model behavioral coordination

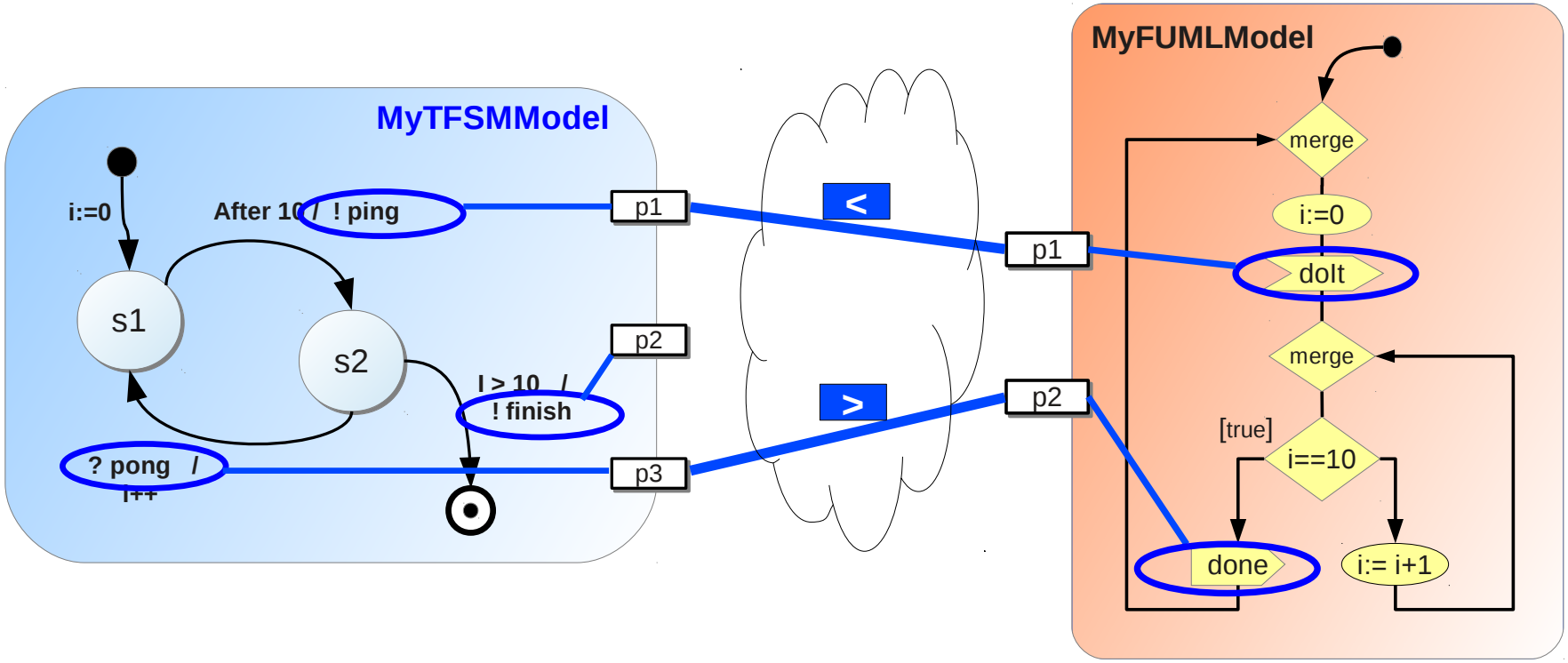
- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >



coordination interfaces synchronizes interface events

model behavioral coordination

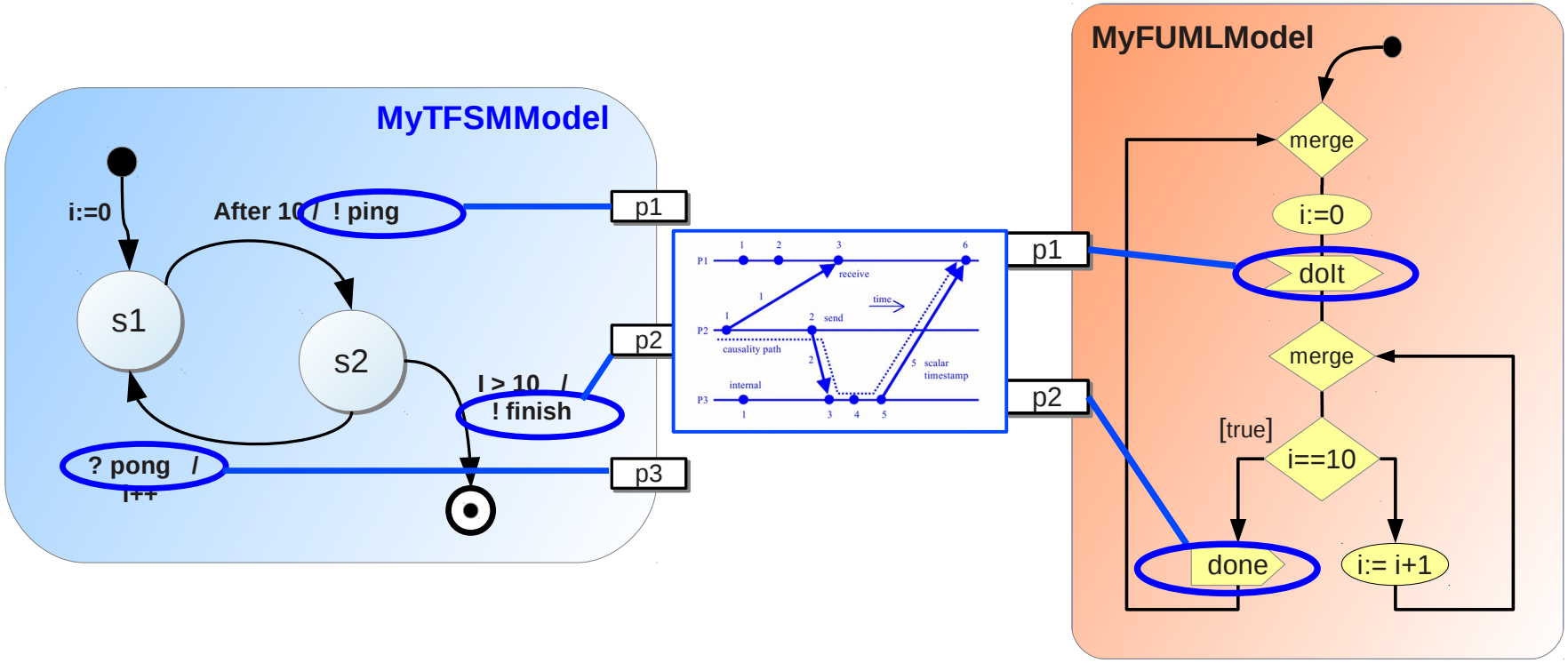
- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >



coordination interfaces synchronizes interface events

model behavioral coordination

- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >



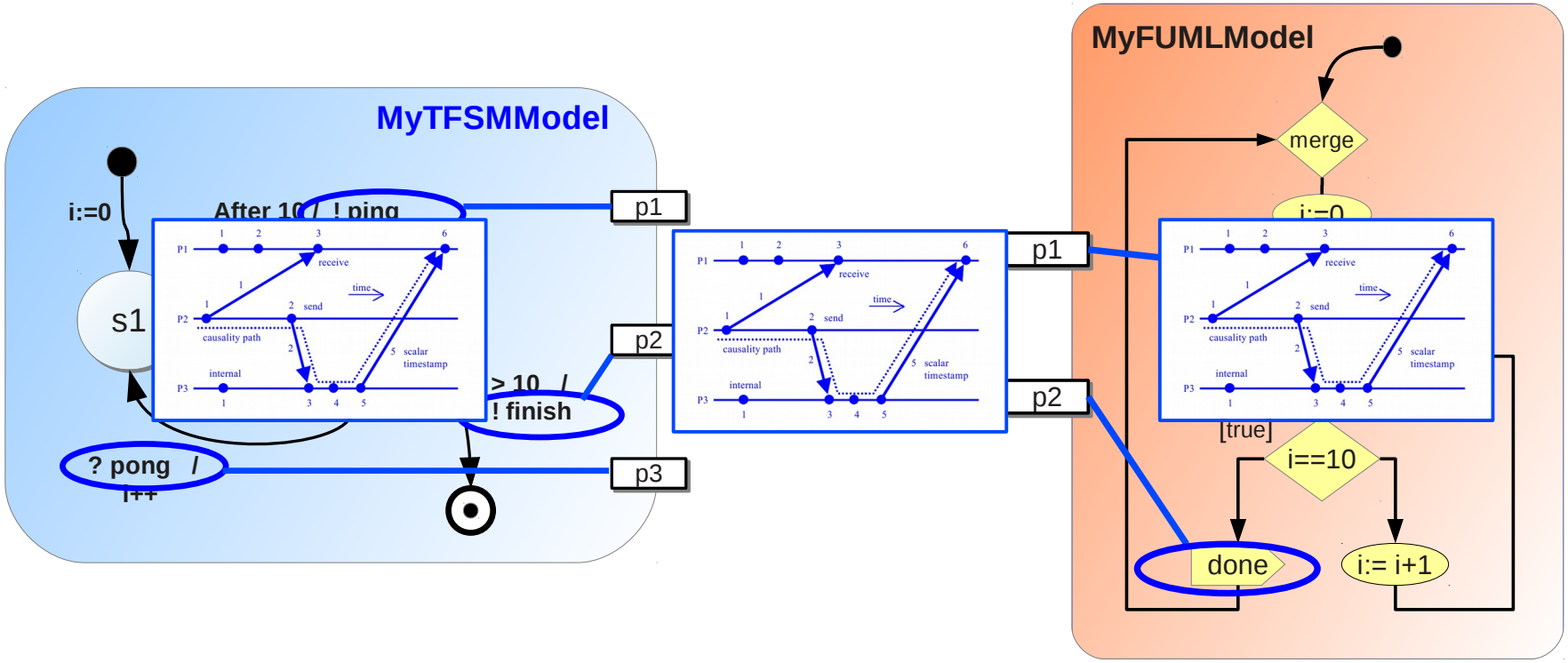
coordination interfaces synchronizes interface events

Railroad Crossing Heterogeneous Model (paper, slides)

Matias Ezequiel Vara Larsen, Arda Goknil,

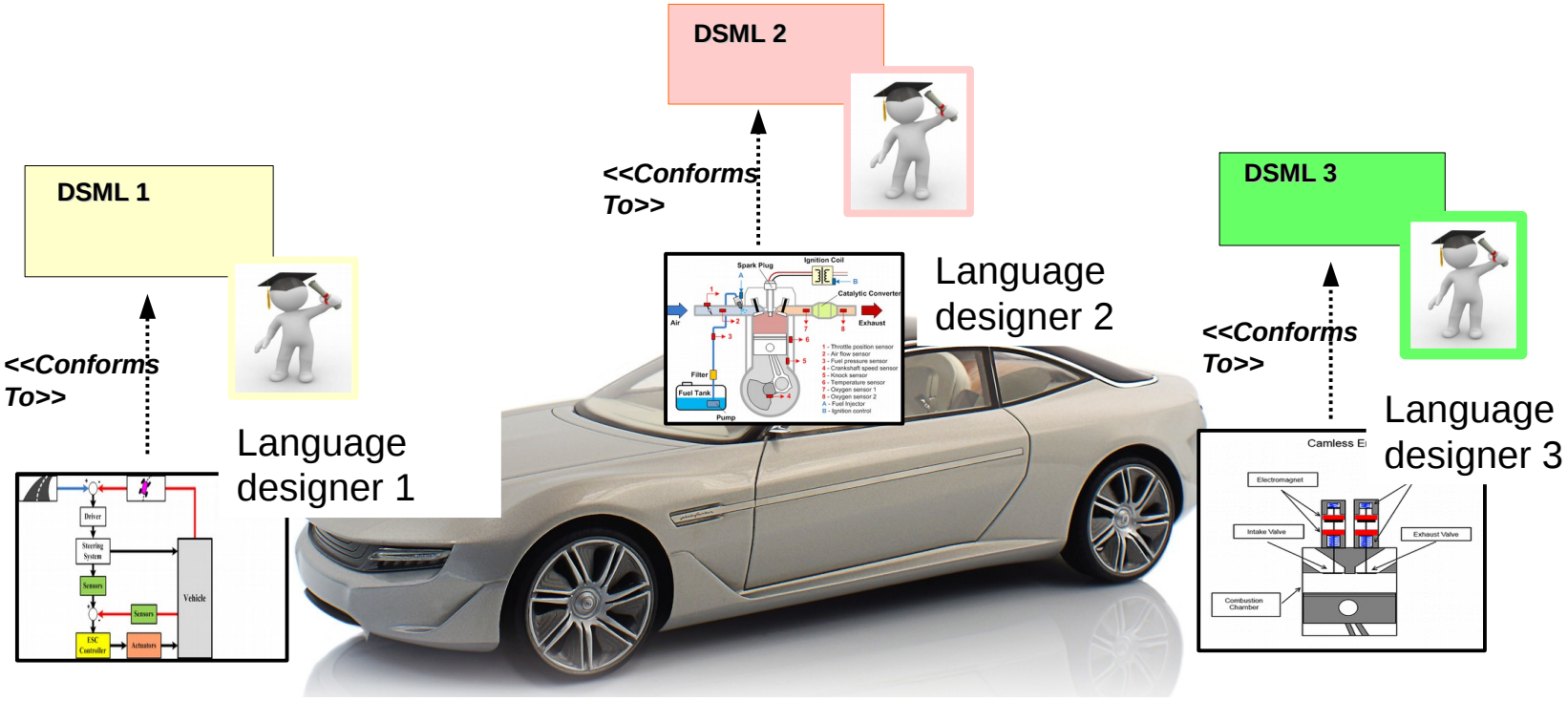
Gemoc@models 2013 Miami

- DSML $\stackrel{def}{=}$ < AS, DSA, Domain Specific Events , MoC >

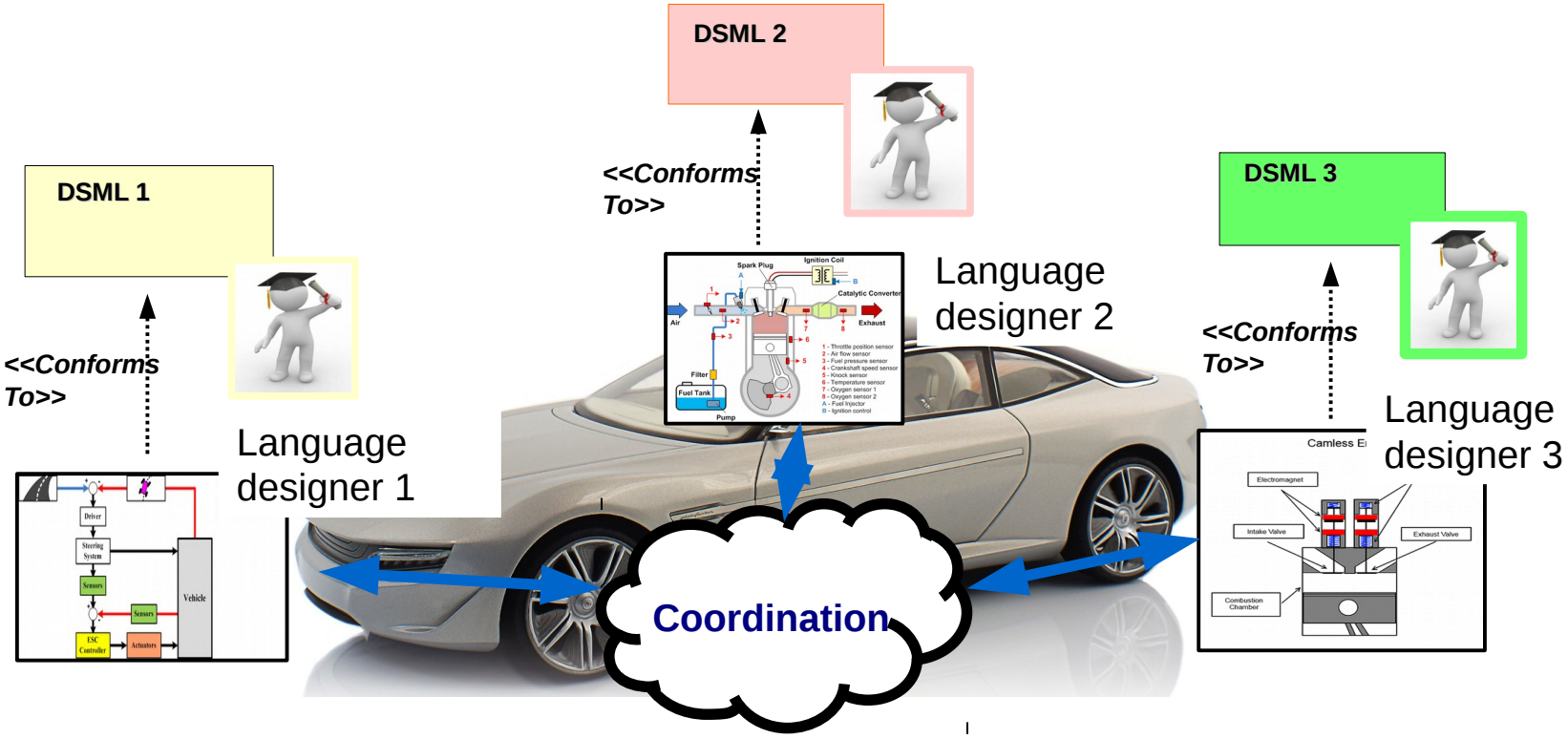


coordination interfaces synchronizes interface events

Heterogeneous development



Heterogeneous development



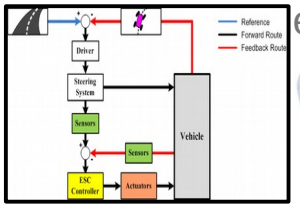
An integrator must specify ***how*** the models ***communicate/coordinate /interact*** one each other

Integrator

DEFINES

DSML 1

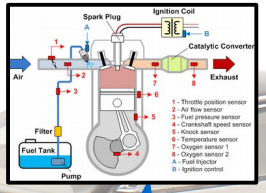
<<Conforms To>>



Language esigner 1

DSML 2

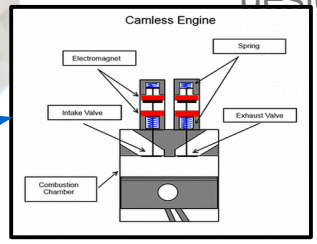
<<Conforms To>>



Language designer 2

DSML 3

<<Conforms To>>



Language designer 3

Coordination

BCOoL

Integrator

BCOol is a language for the integrator to express its coordination skills

DSML 1

DSML 2

DSML 3

<<Conforms To>>

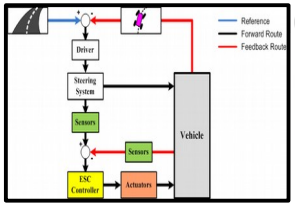
<<Conforms To>>

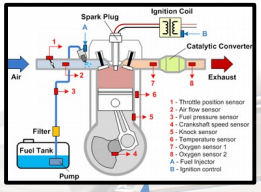
<<Conforms To>>

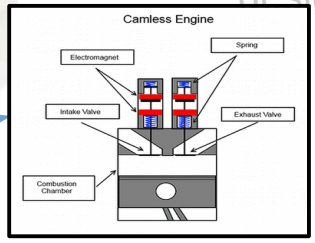
Language designer 1

Language designer 2

Language designer 3

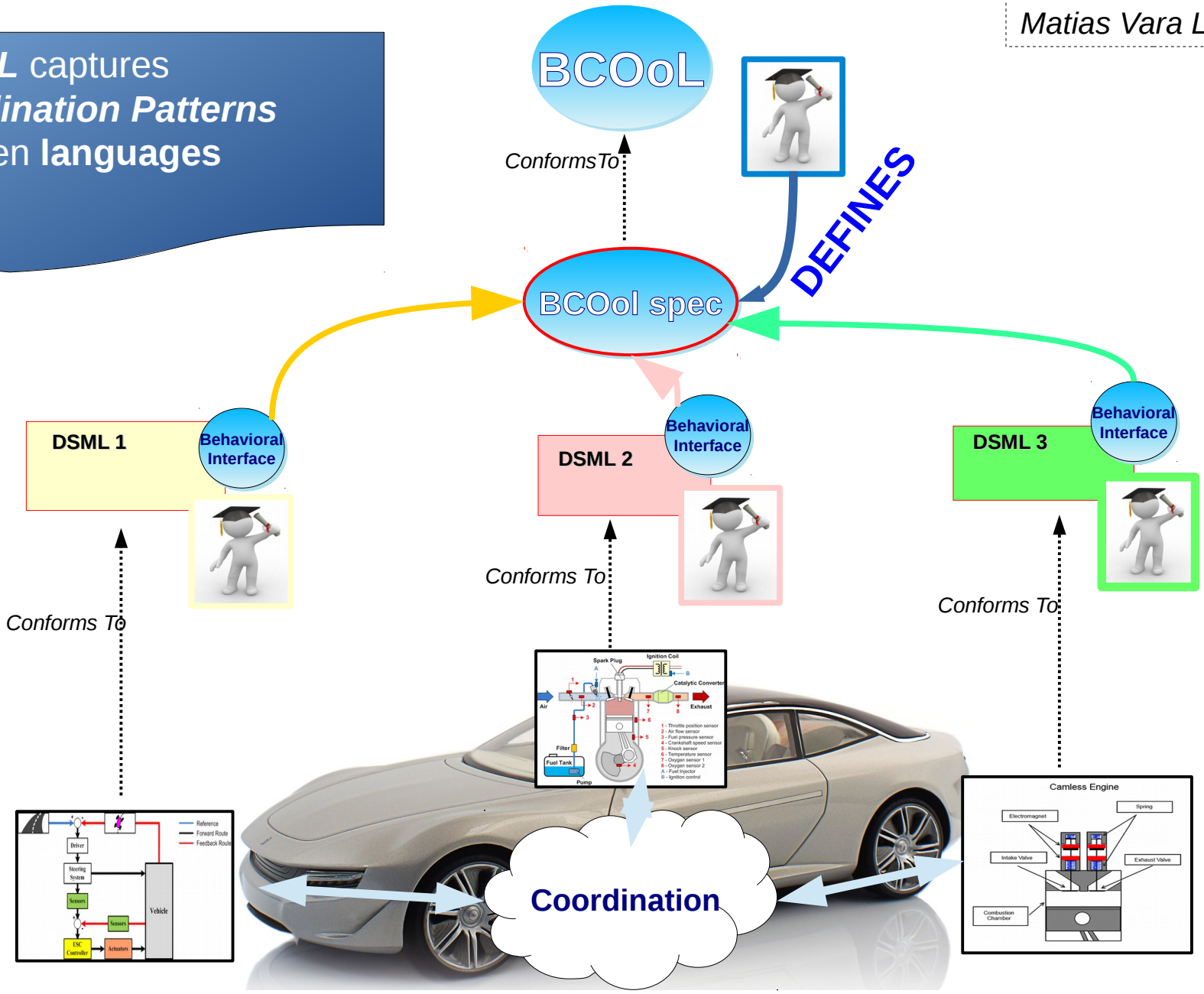




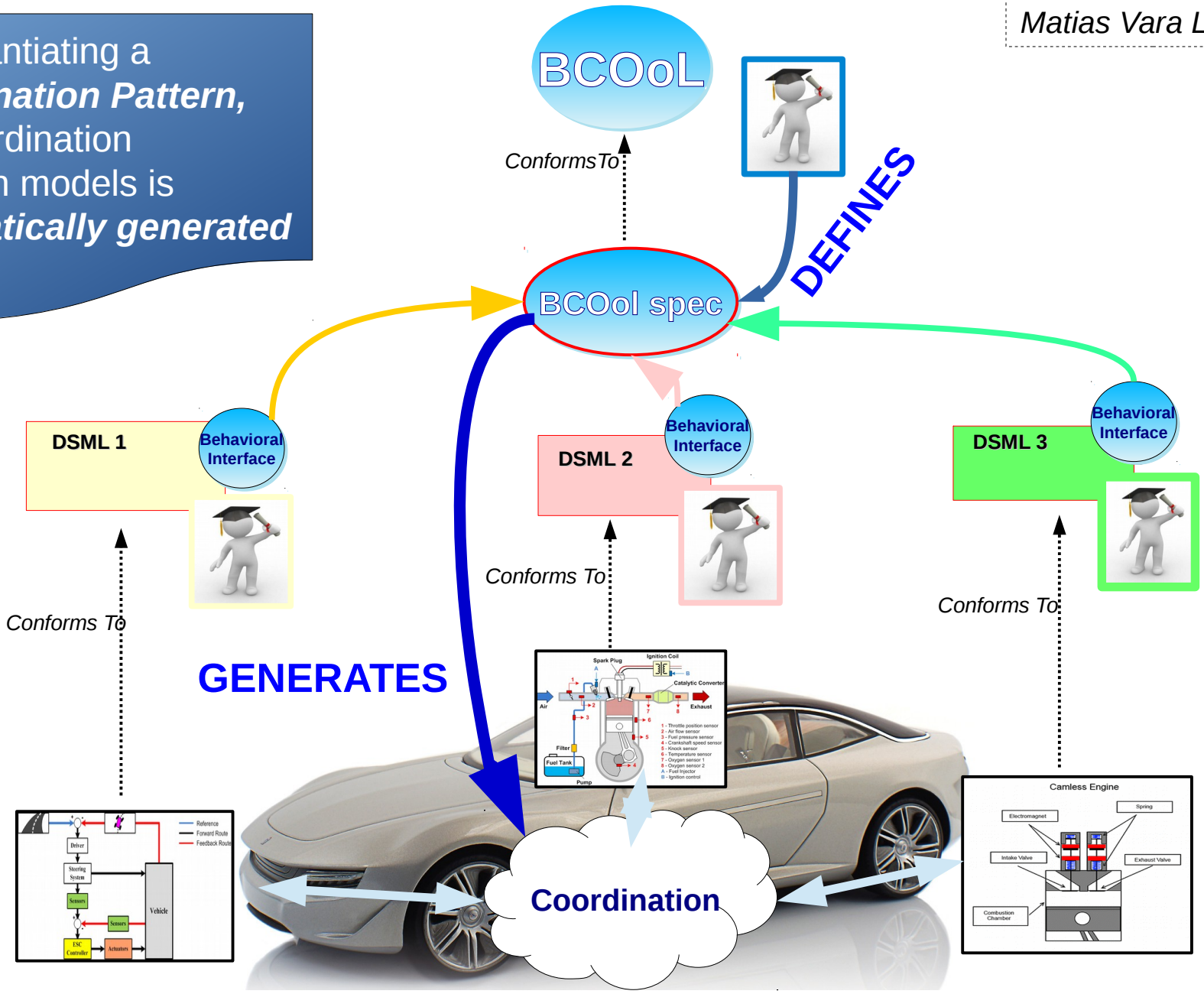


Coordination

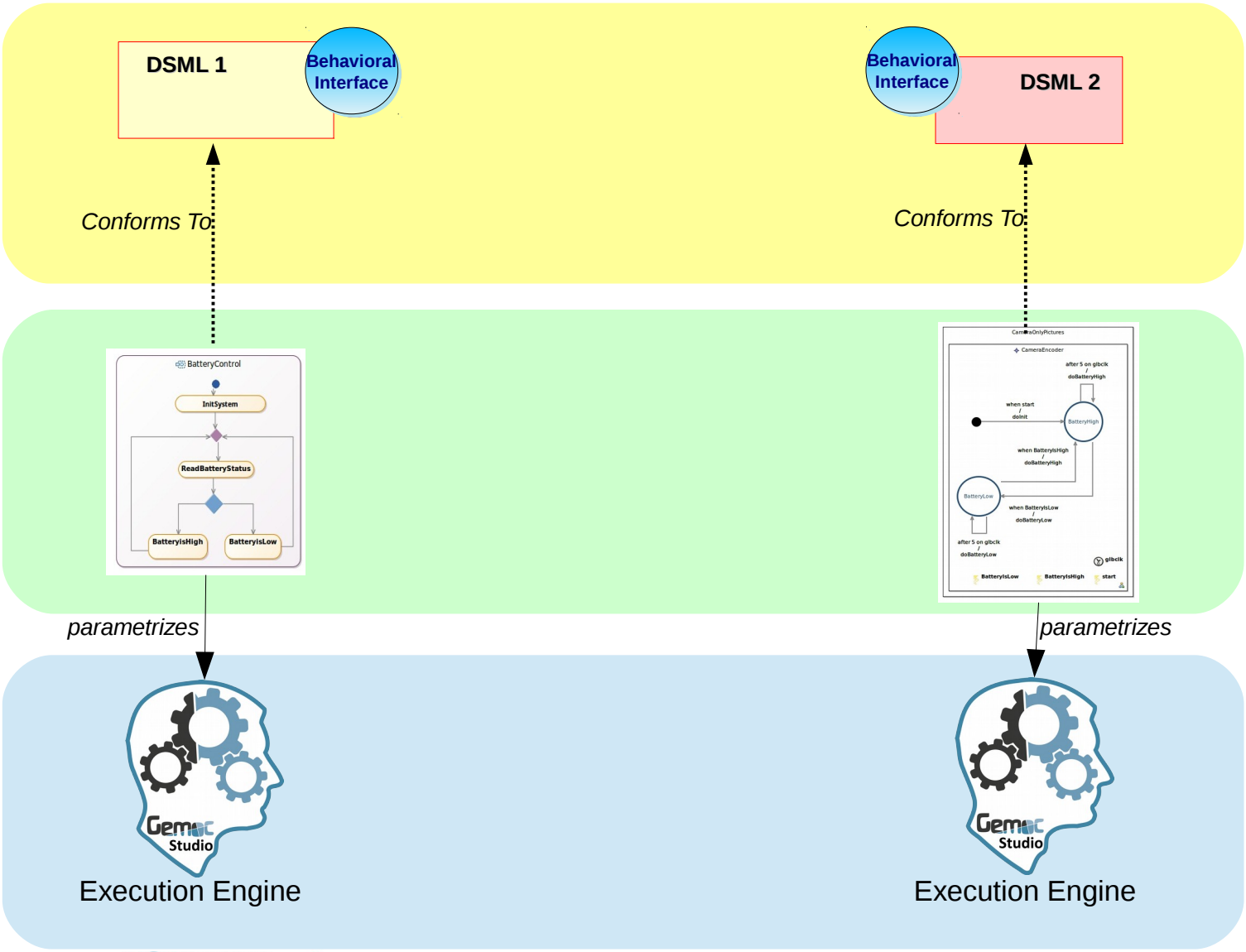
BCOoL captures **Coordination Patterns** between **languages**



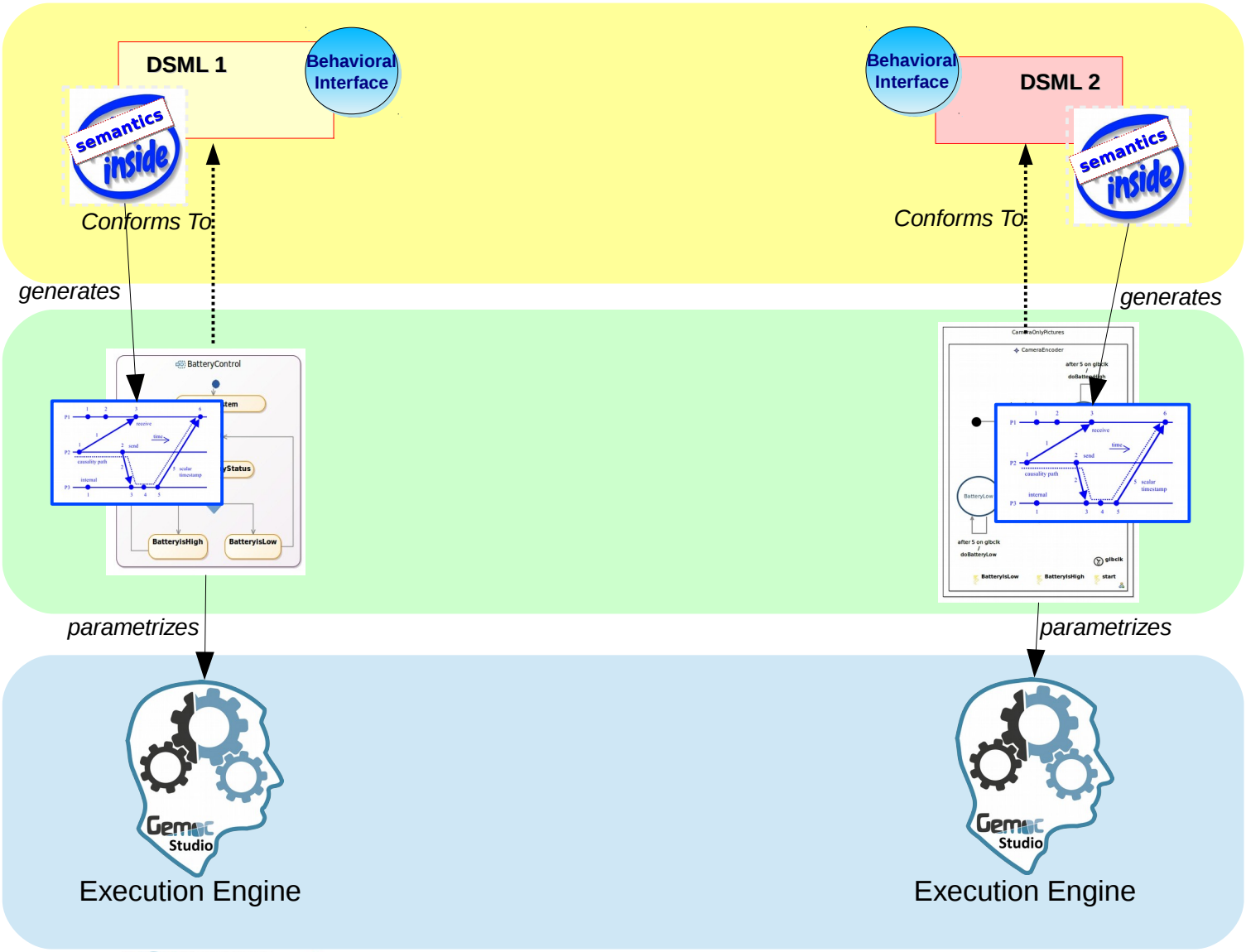
By instantiating a ***Coordination Pattern,*** the coordination between models is ***automatically generated***



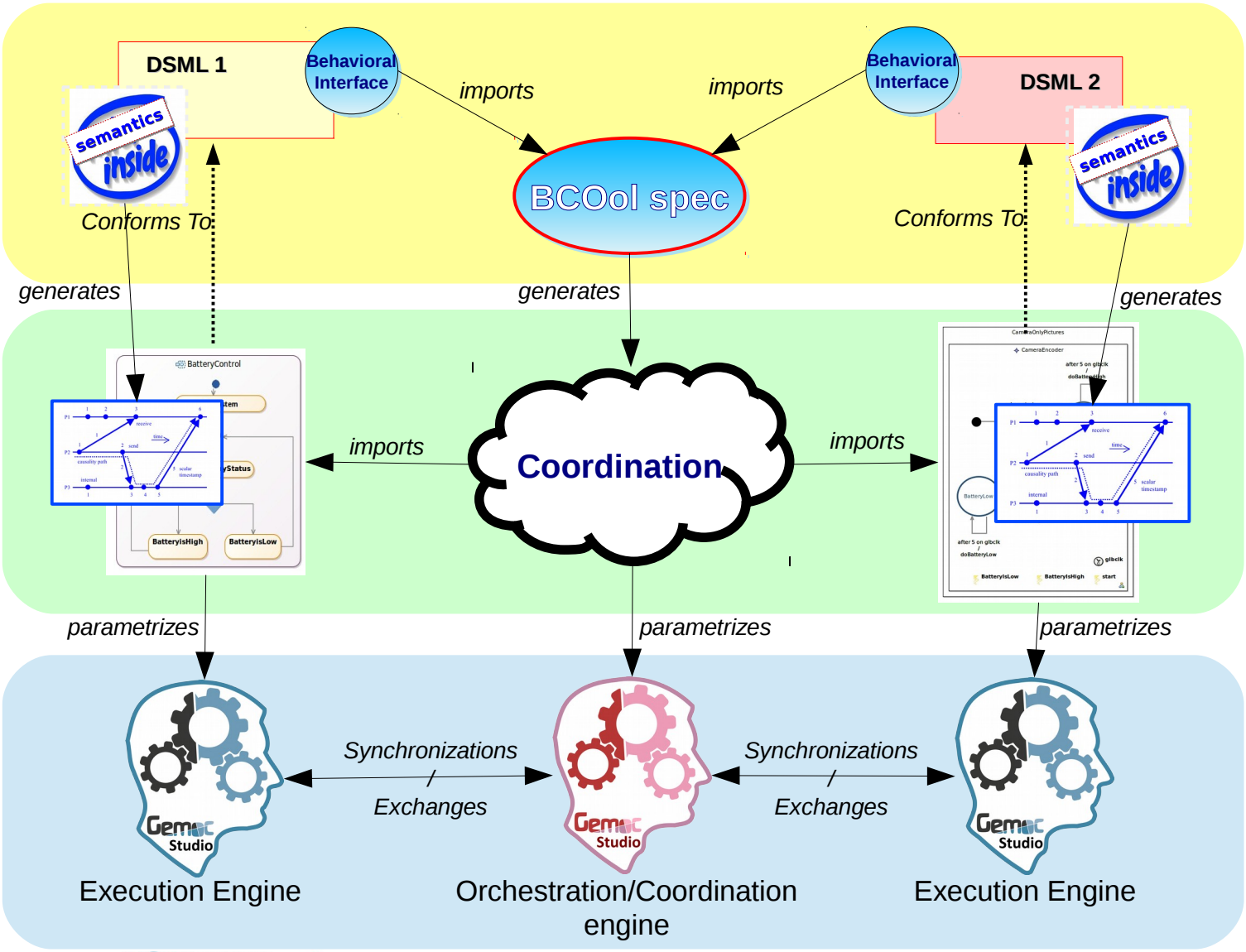
Sum up on our approach



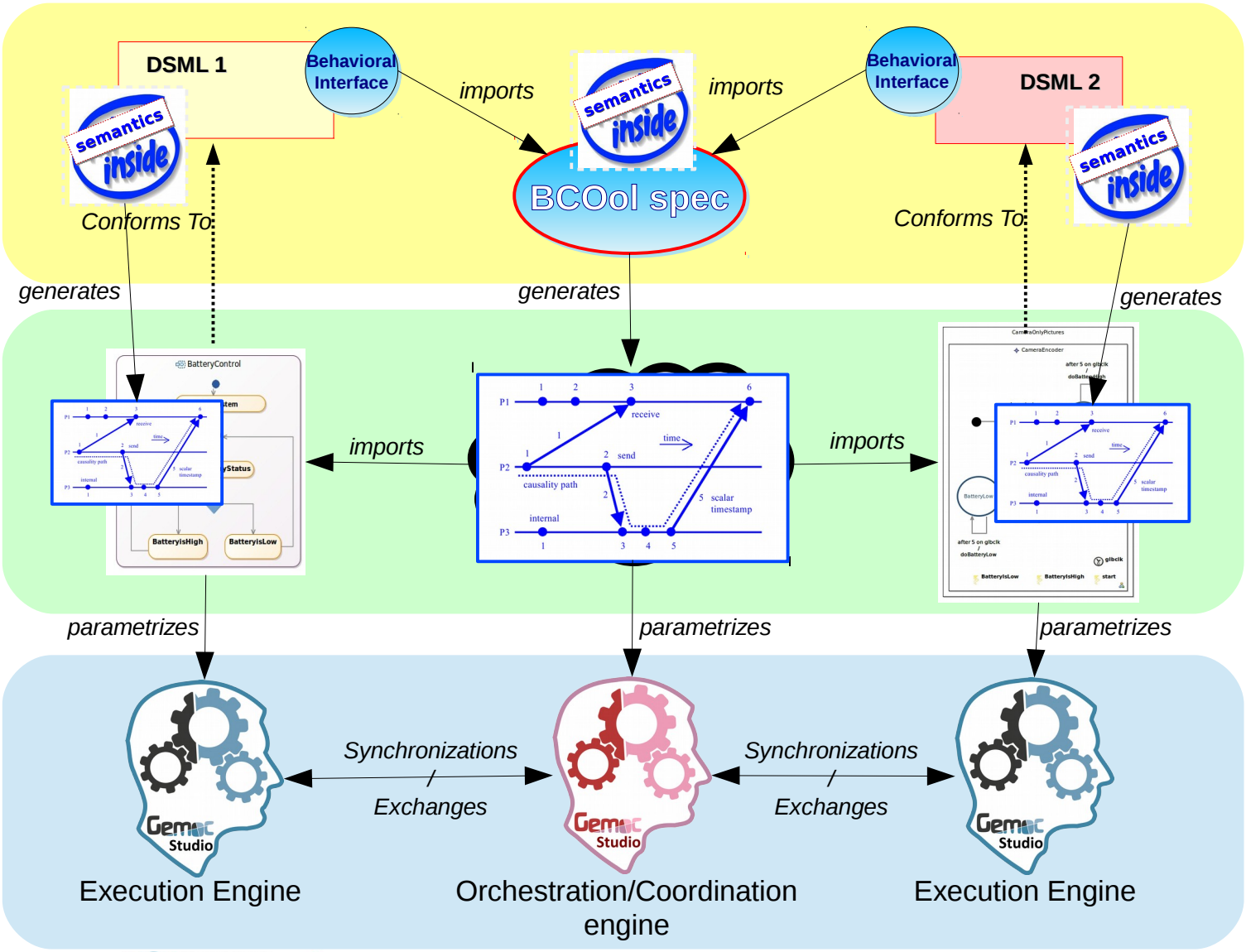
Sum up on our approach



Sum up on our approach

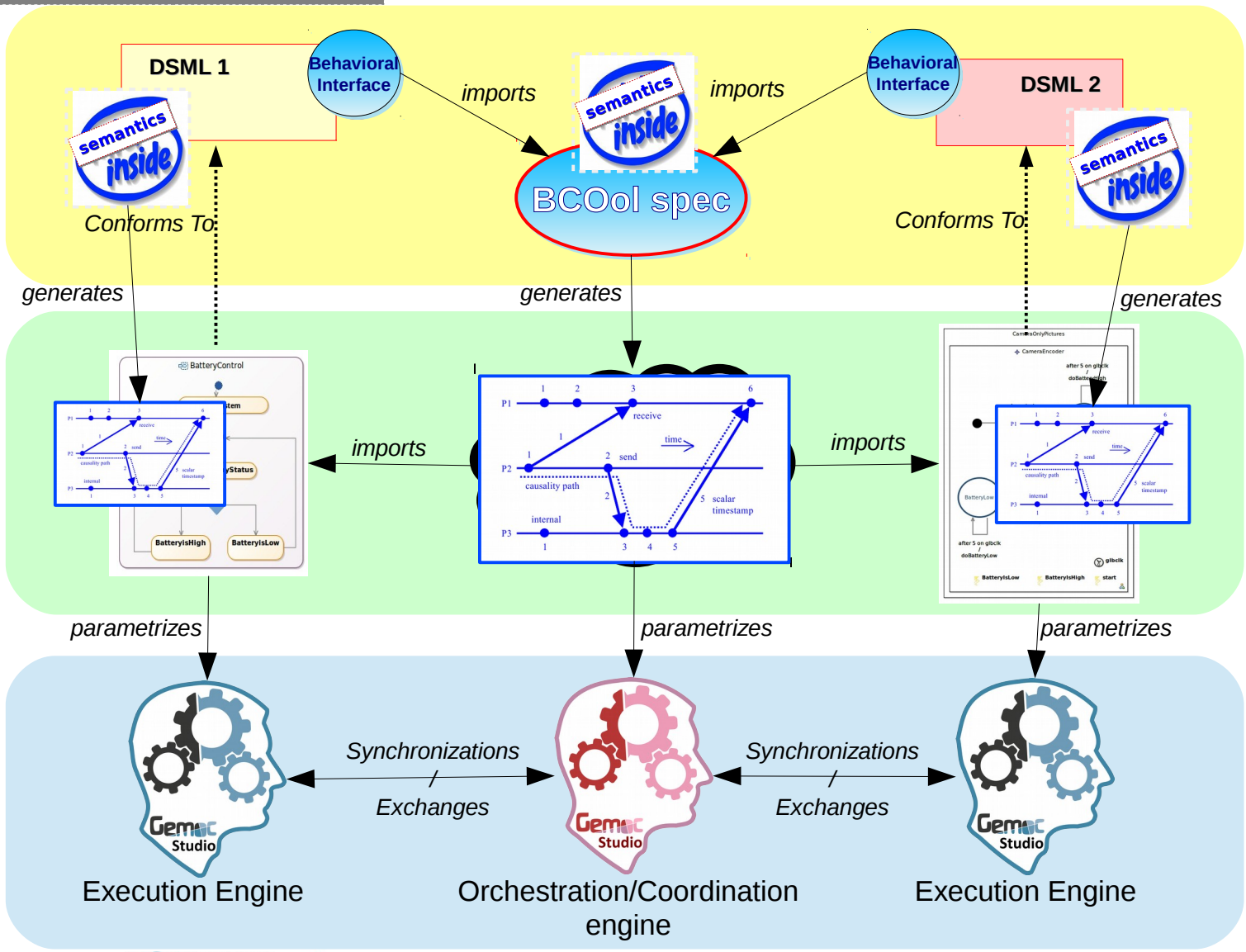


Sum up on our approach

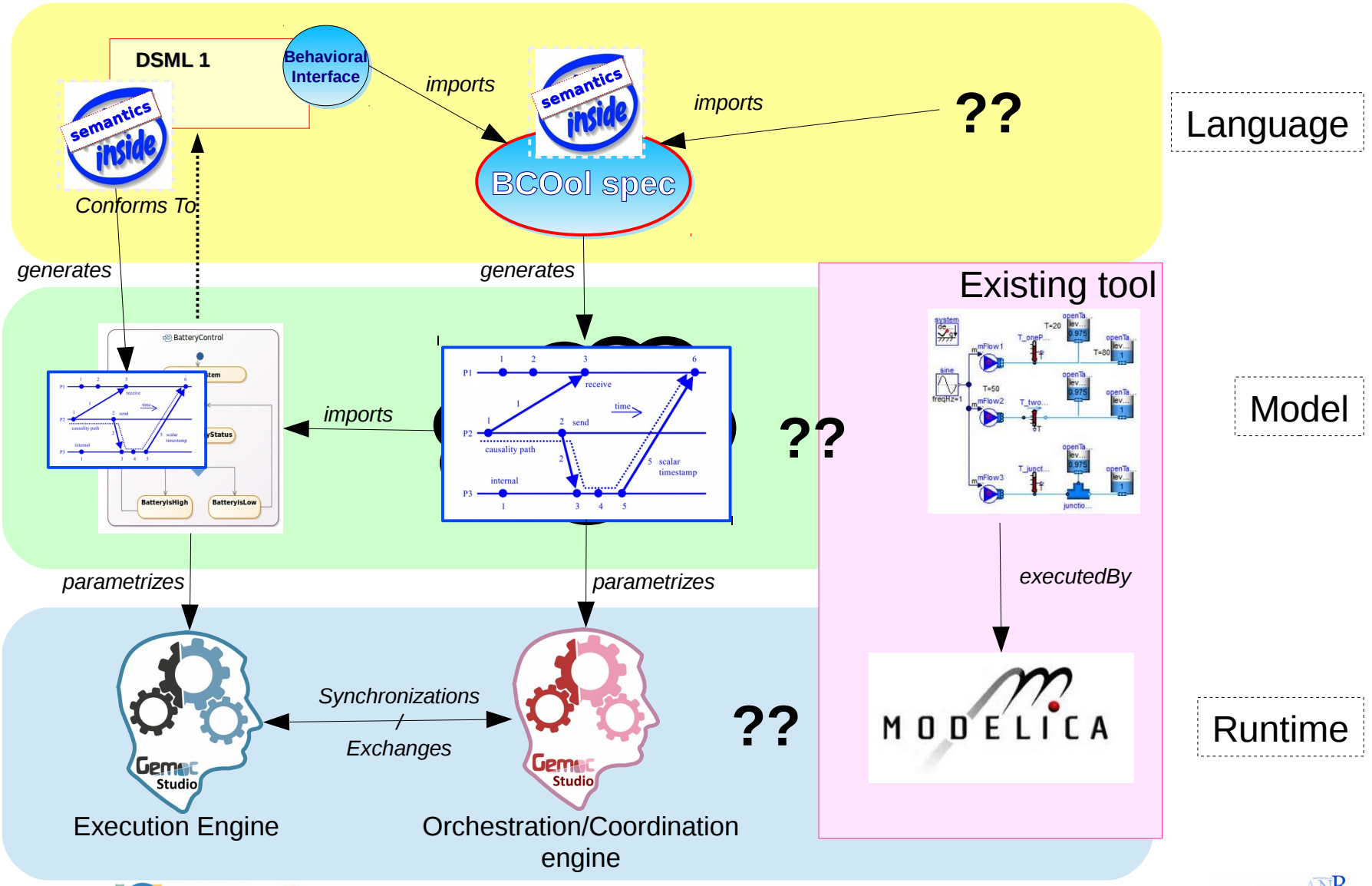


Only for discrete semantics

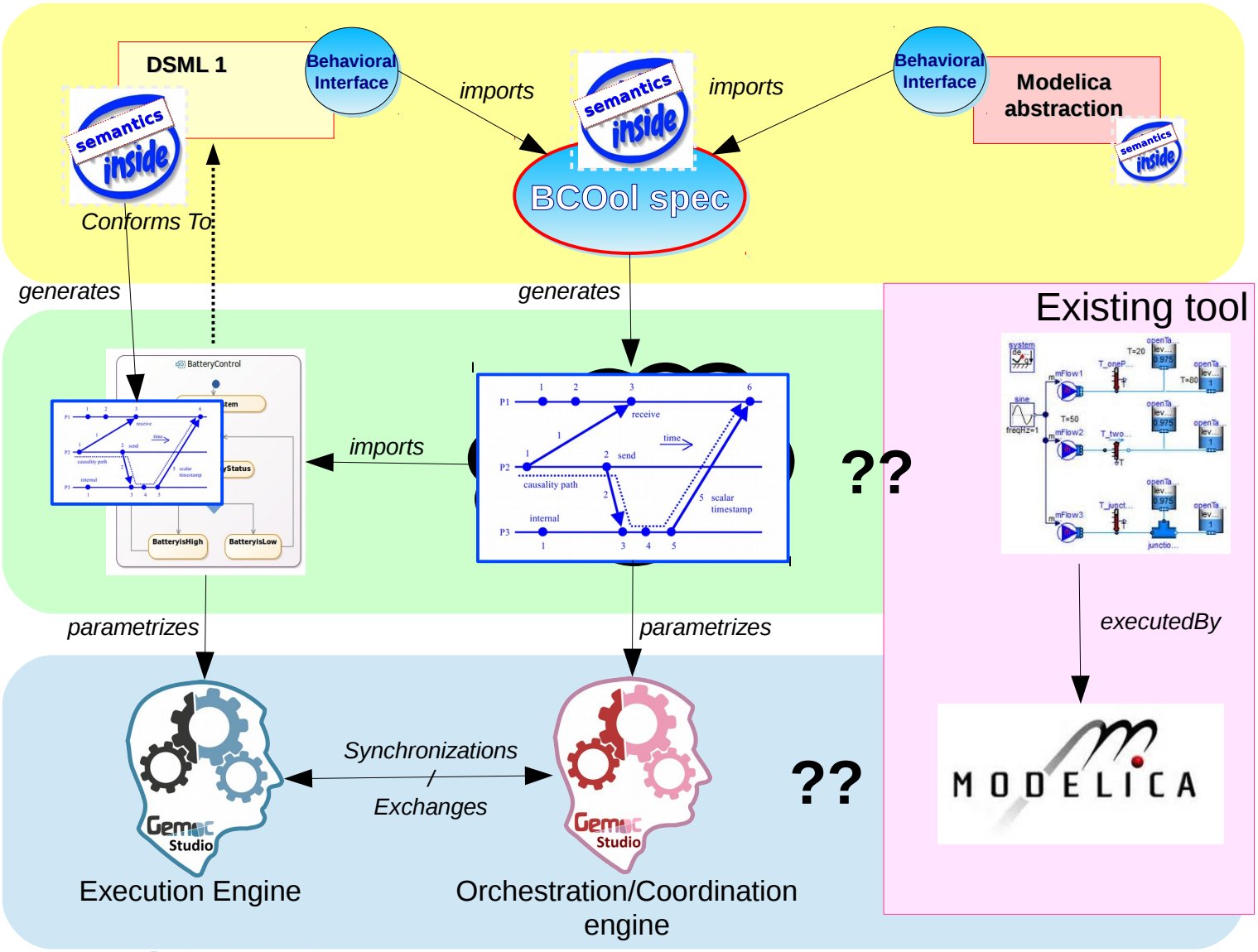
Sum up on our approach



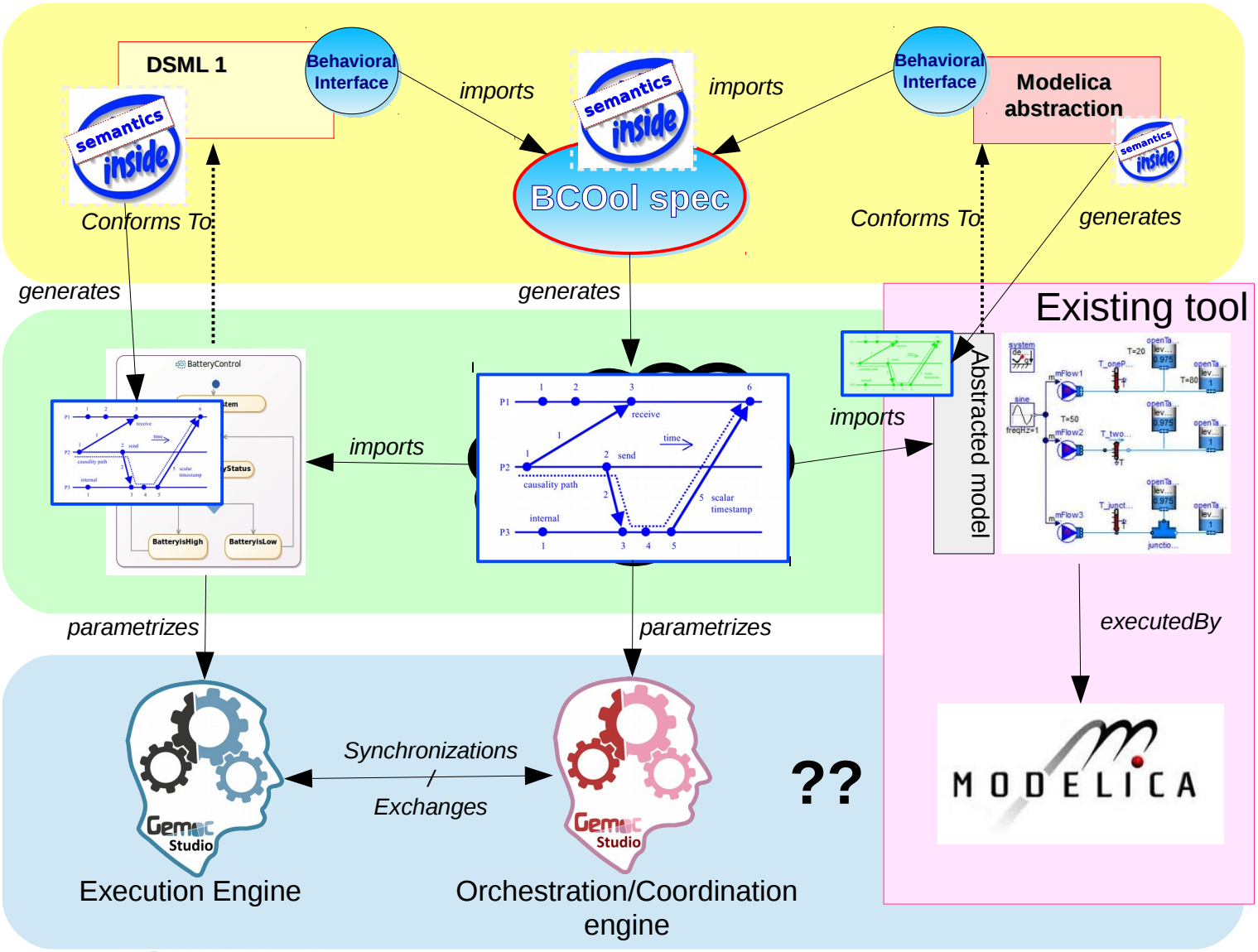
Current work



Current work



Current work



Current work

We did home made connectors with scilab...

DSML 1

Behavioral Interface

imports

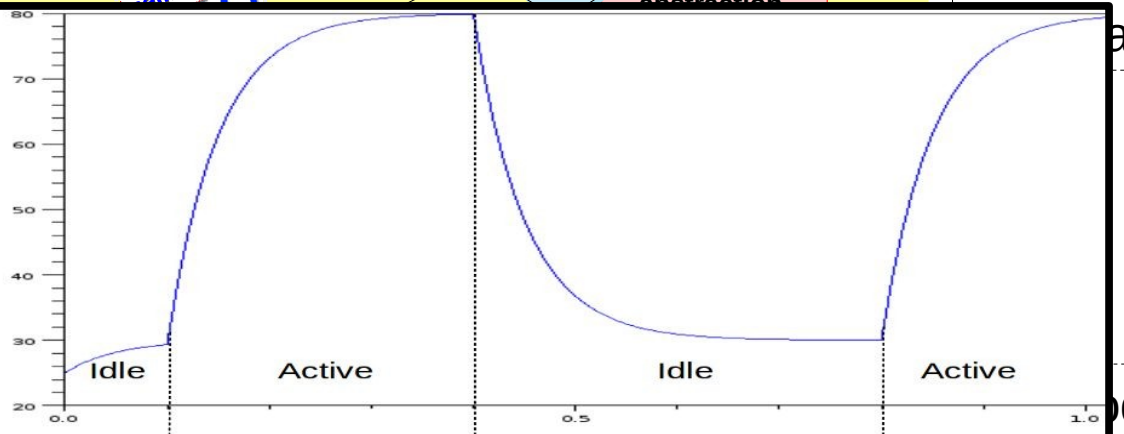
imports

Behavioral Interface

Modelica

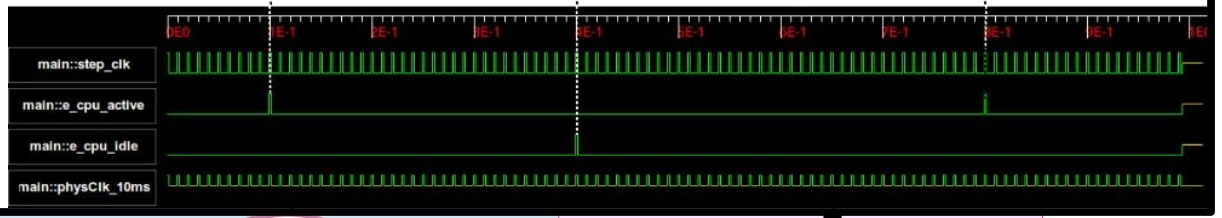
CPU Temperature Evolution

Scilab



Control evolution

TimeSquare



Amani Khecharem, Carlos Gomez, Julien Deantoni, Frédéric Mallet, Robert de Simone.
Execution of Heterogeneous Models for Thermal Analysis with a Multi-view Approach. FDL 2014 : Forum on specification and Design Languages, Oct 2014, Munich, Germany. IEEE

Exchanges

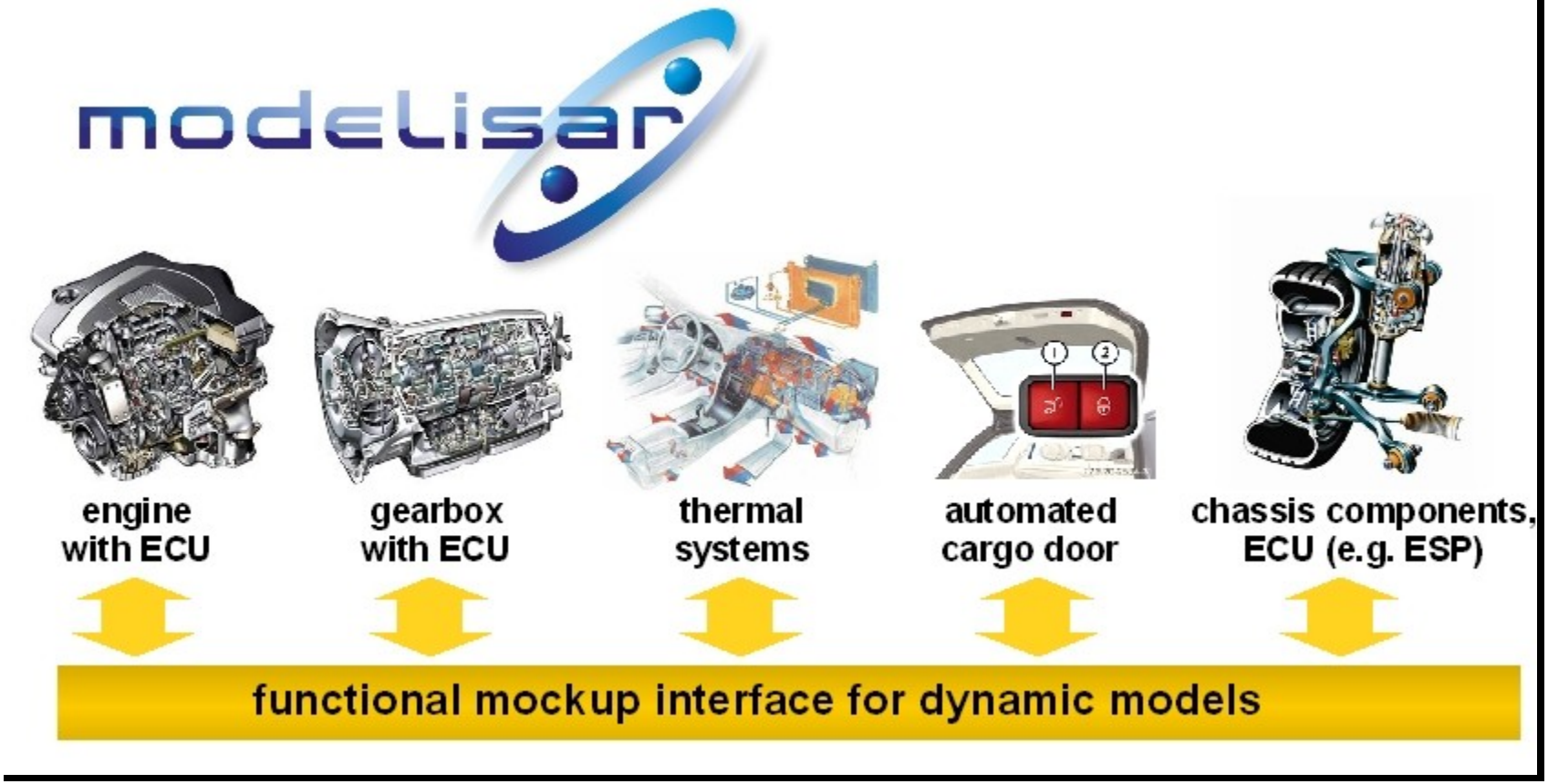
Gemoc Studio

Execution Engine

Gemoc Studio

Orchestration/Coordination engine

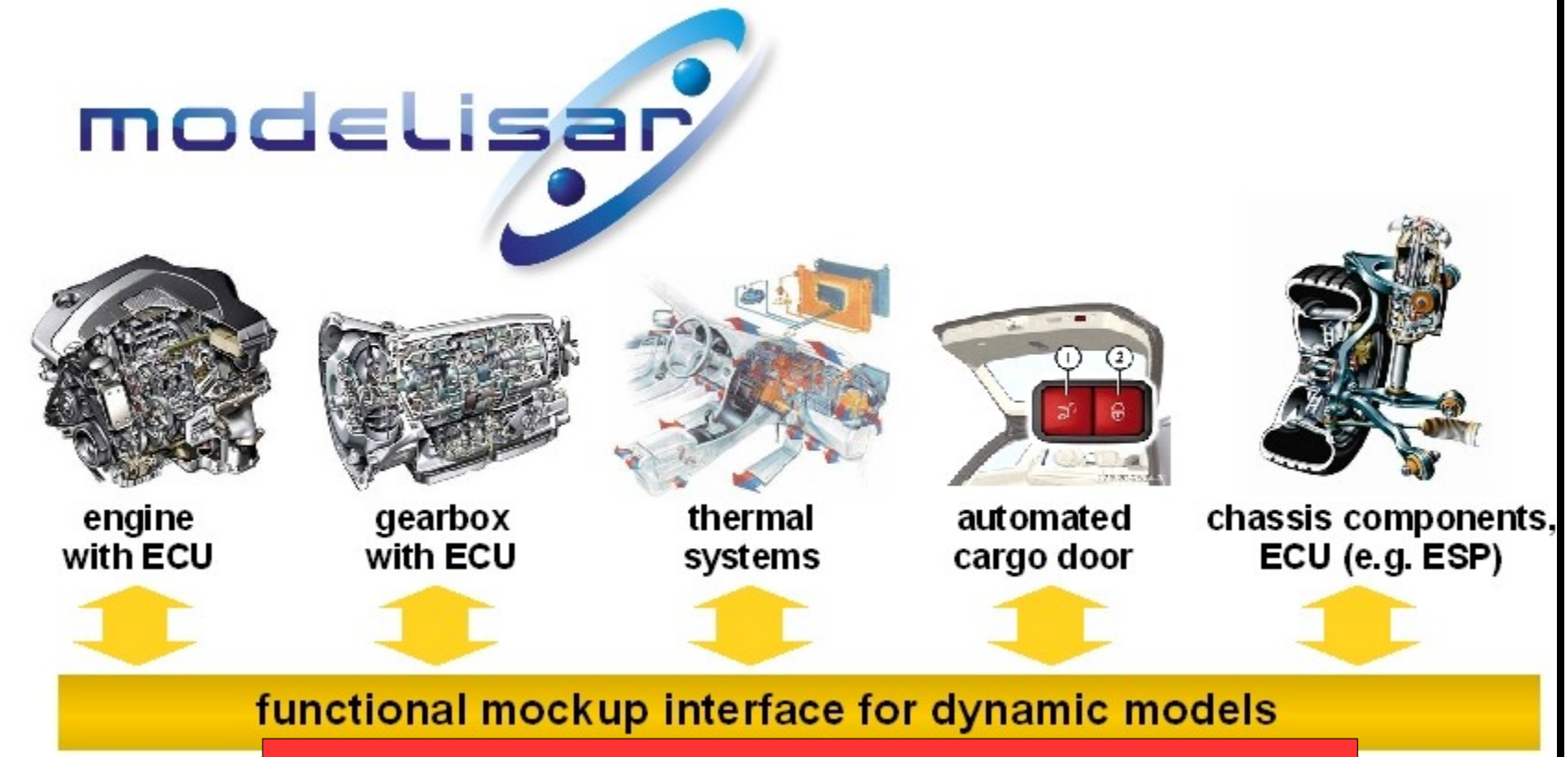
FMI / FMU



Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code

https://fmi-standard.org/

FMI / FMU

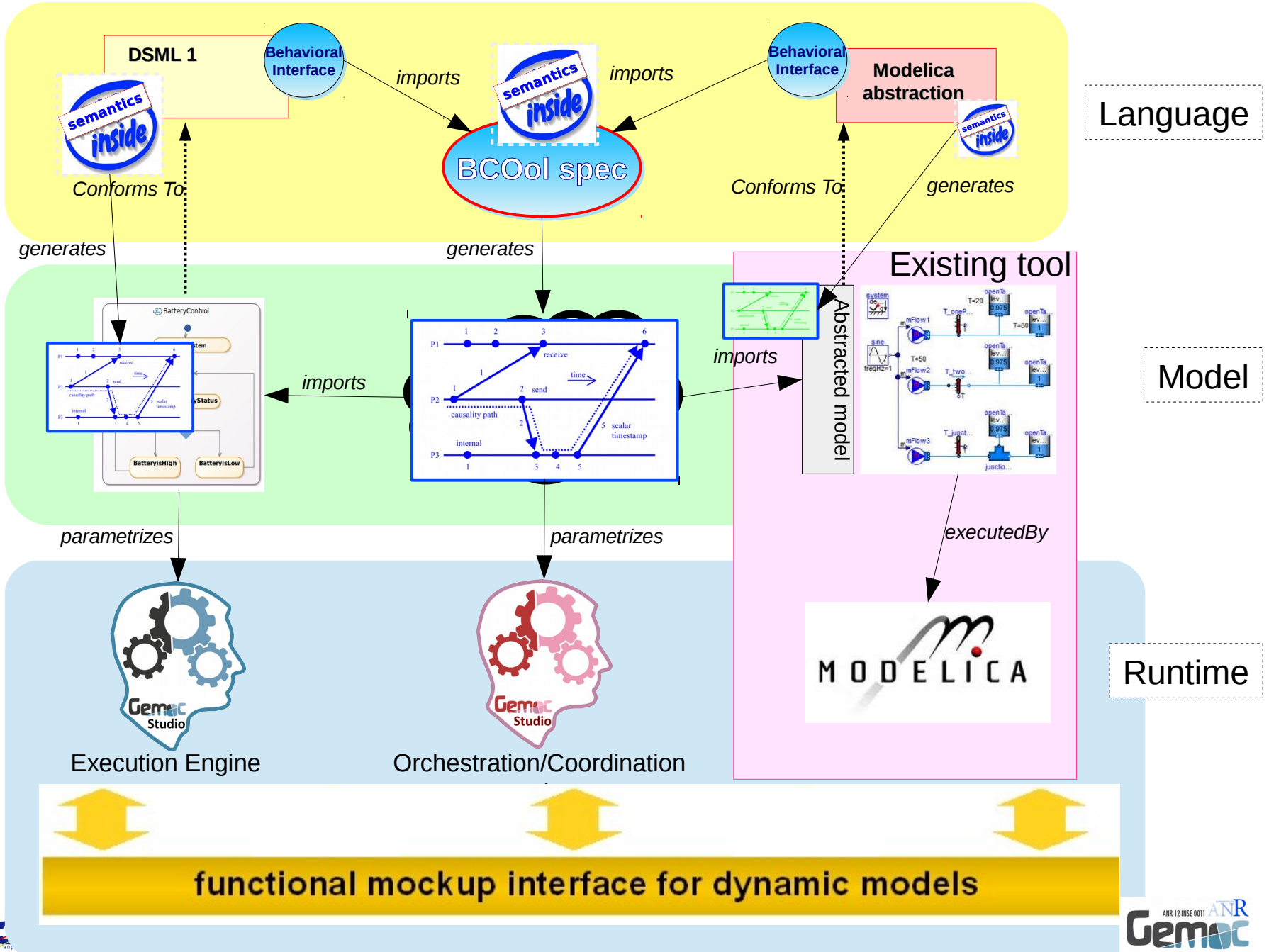


Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code

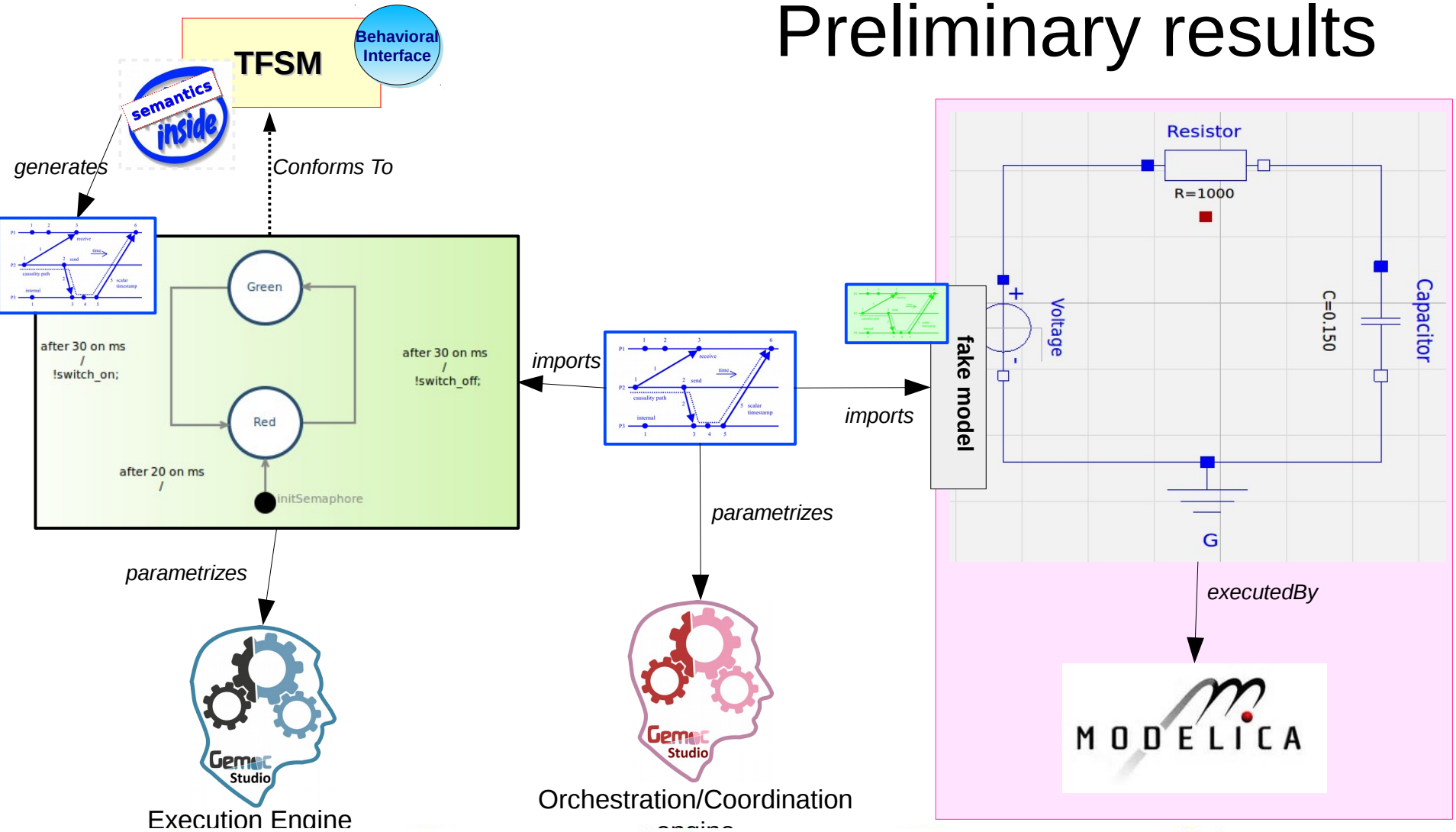
https://fmi-standard.org/

Michele Fabbri

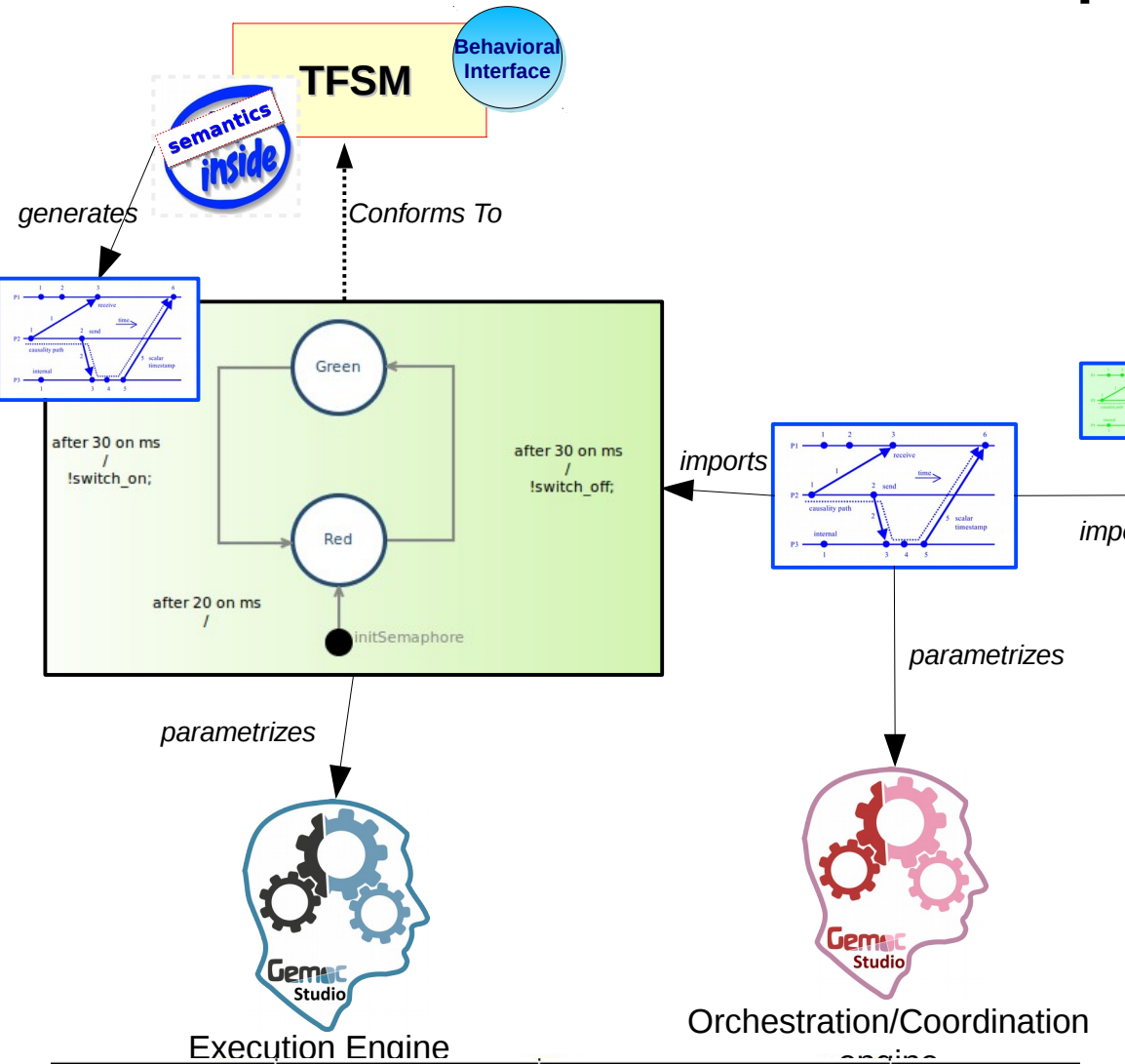
Current work

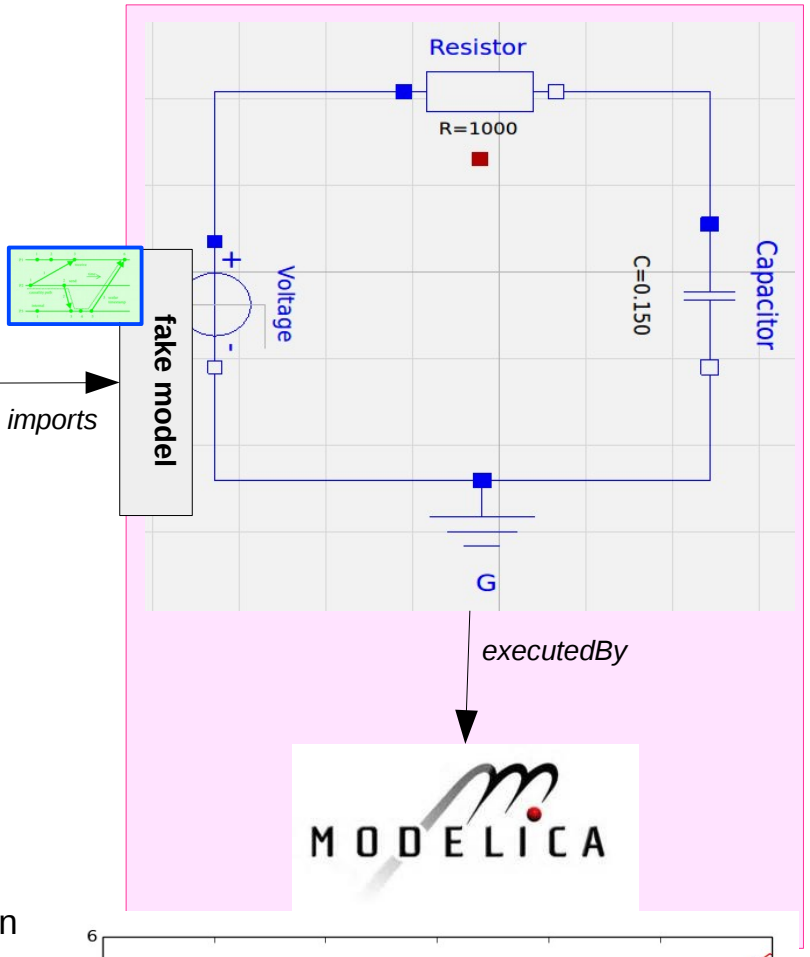


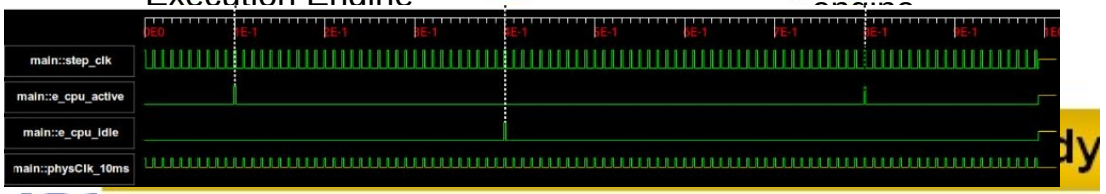
Preliminary results

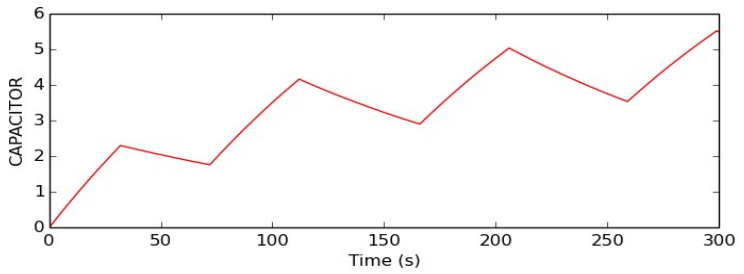


Preliminary results









Conclusion


model toys
model toys
giocattoli
NOTAIO
OXFORD
SCHOOL
of English

Conclusion

Conclusion

Languages should reify their behavioral semantics...
so that we can understand and automate reasoning...
on the execution and coordination of models...
at different level of abstraction...

This is the end...

...thanks...

Julien Deantoni

University of Nice, I3S CNRS

Julien.deantoni@polytech.unice.fr

INRIA Aoste