The ***B***ehavioral ***C***oordination ***O***perat***o***r ***L***anguage

MPM4CPS ***27th of April 2018***

Julien Deantoni
Universite Cote d'Azur,
CNRS I3S, INRIA KAIROS
Julien.deantoni@polytech.unice.fr

Modeling the behavioral semantics of languages & their coordination

MPM4CPS ***27th of April 2018***

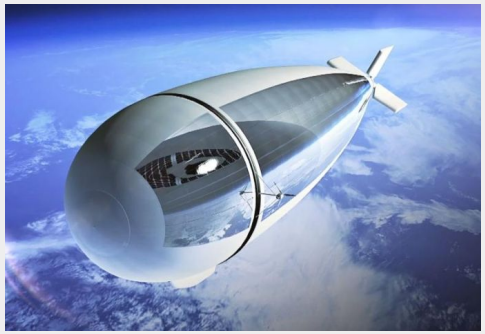
Julien Deantoni
Universite Cote d'Azur,
CNRS I3S, INRIA KAIROS
Julien.deantoni@polytech.unice.fr

System

*The term "system" comes from the Latin word systēma, in turn from Greek σύστημα : "**a whole made of several parts or members**".*

wikipedia

Airbus 390



Thales Alenia Space: stratobus and satellites

Many stakeholders are needed to develop such systems

Renault Autonomous Car

Safran Nacelle

System Engineering

Mechanical Structure

Airlines

Avionics

Human-Machine Interaction

Aerodynamics

Airbus 390

Environmental Impact

Propulsion System

Safety Regulations

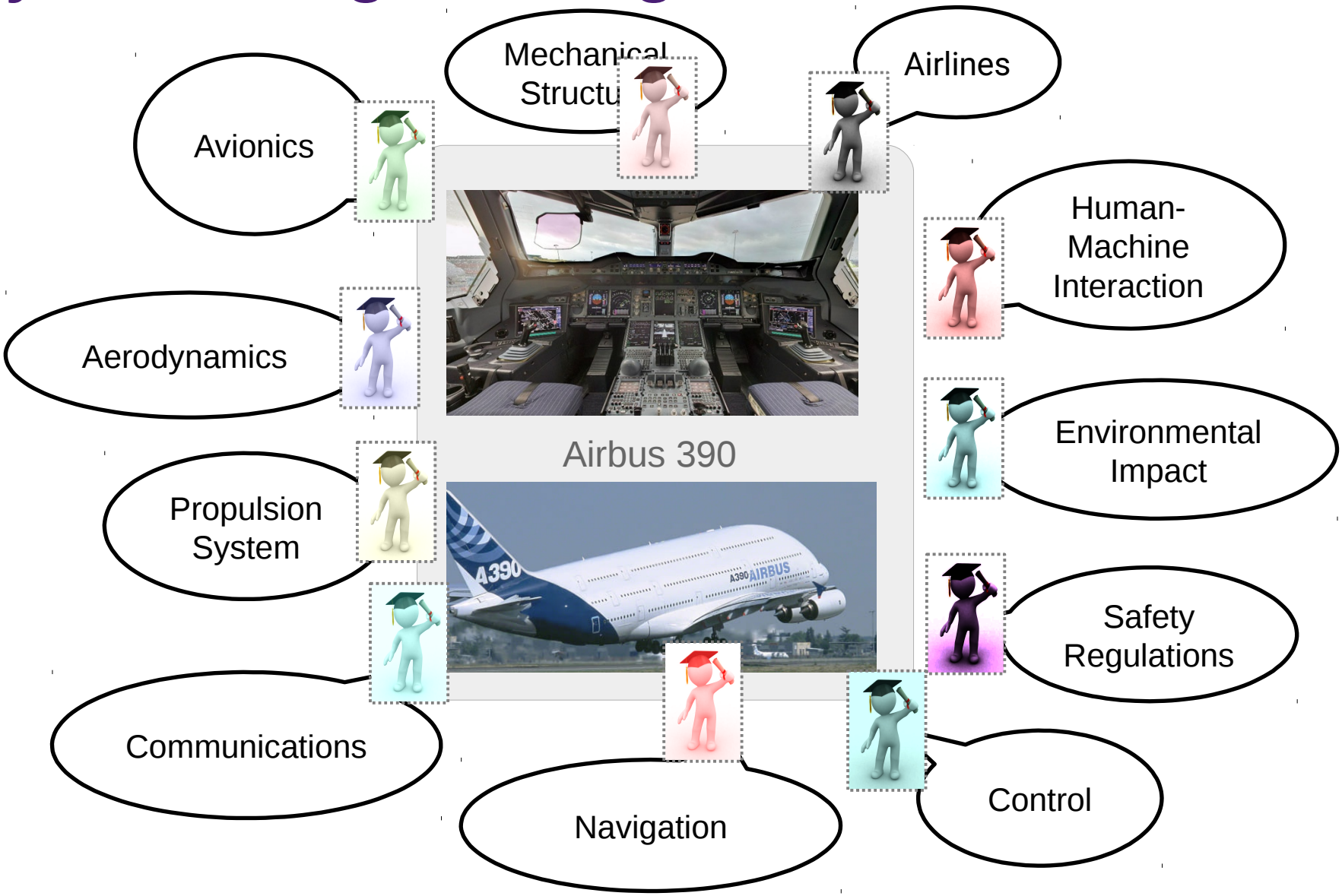
Communications

Navigation

Control

Several concerns for a single system

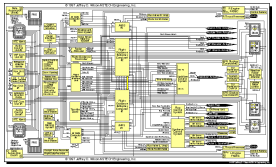
System Engineering

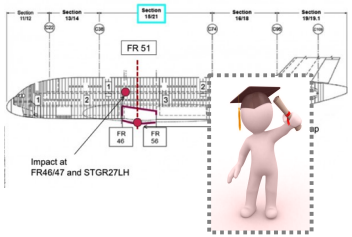


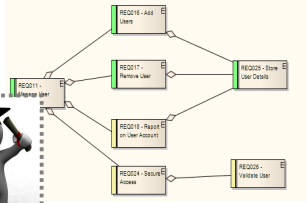
Several concerns for a single system

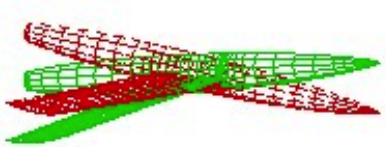
At least one expert by concern

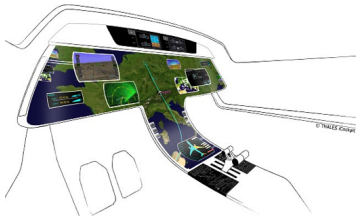
System Engineering

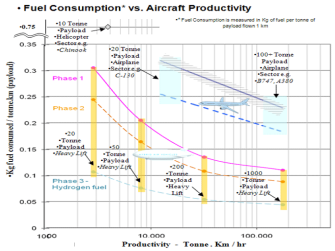


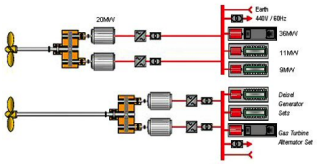




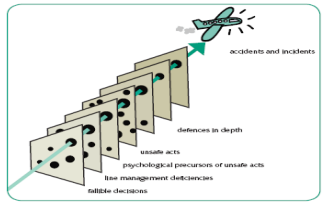


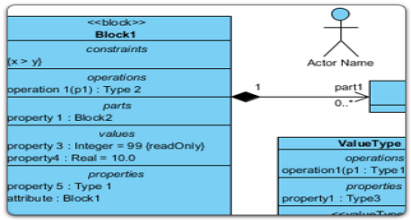


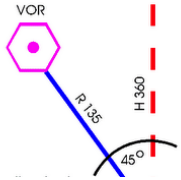


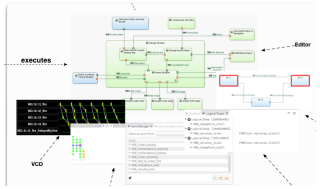


Airbus 390







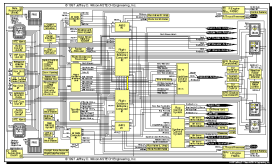


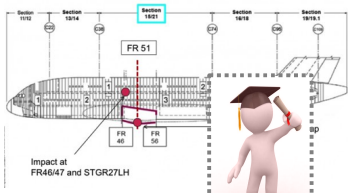
Several concerns for a single system

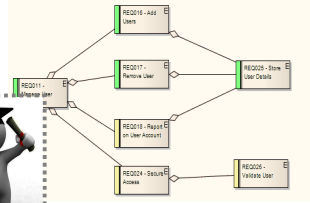
At least one expert by concern

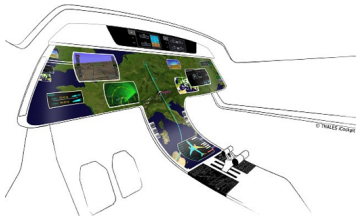
At least one “model” by concern (expressed in a DSML)

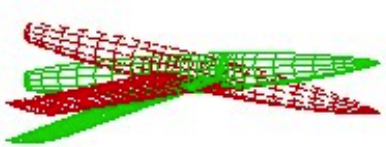
System Engineering

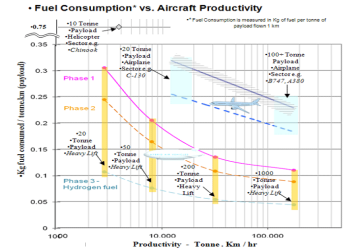


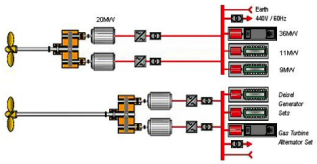




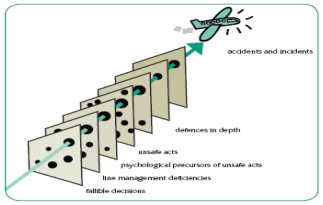


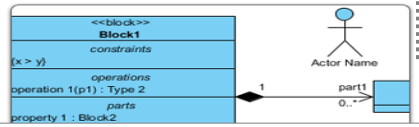


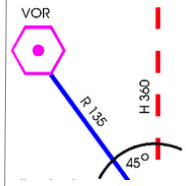




Airbus 390







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

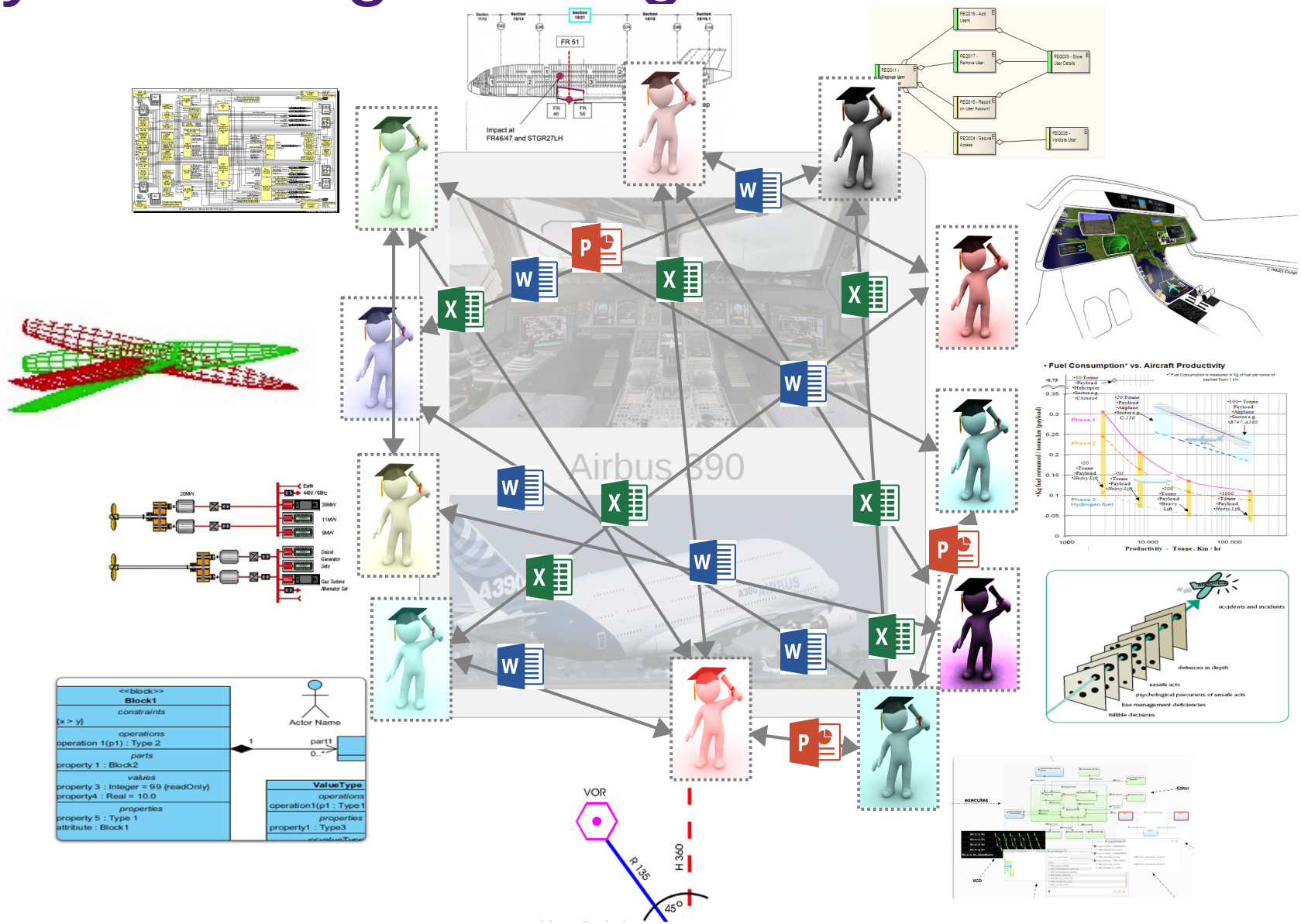
*a **Model** is an **Abstraction** of a specific **Concern/View** of a system for a given **Purpose***

Several concerns for a single system

At least one expert by concern

At least one "model" by concern (expressed in a DSML)

System Engineering

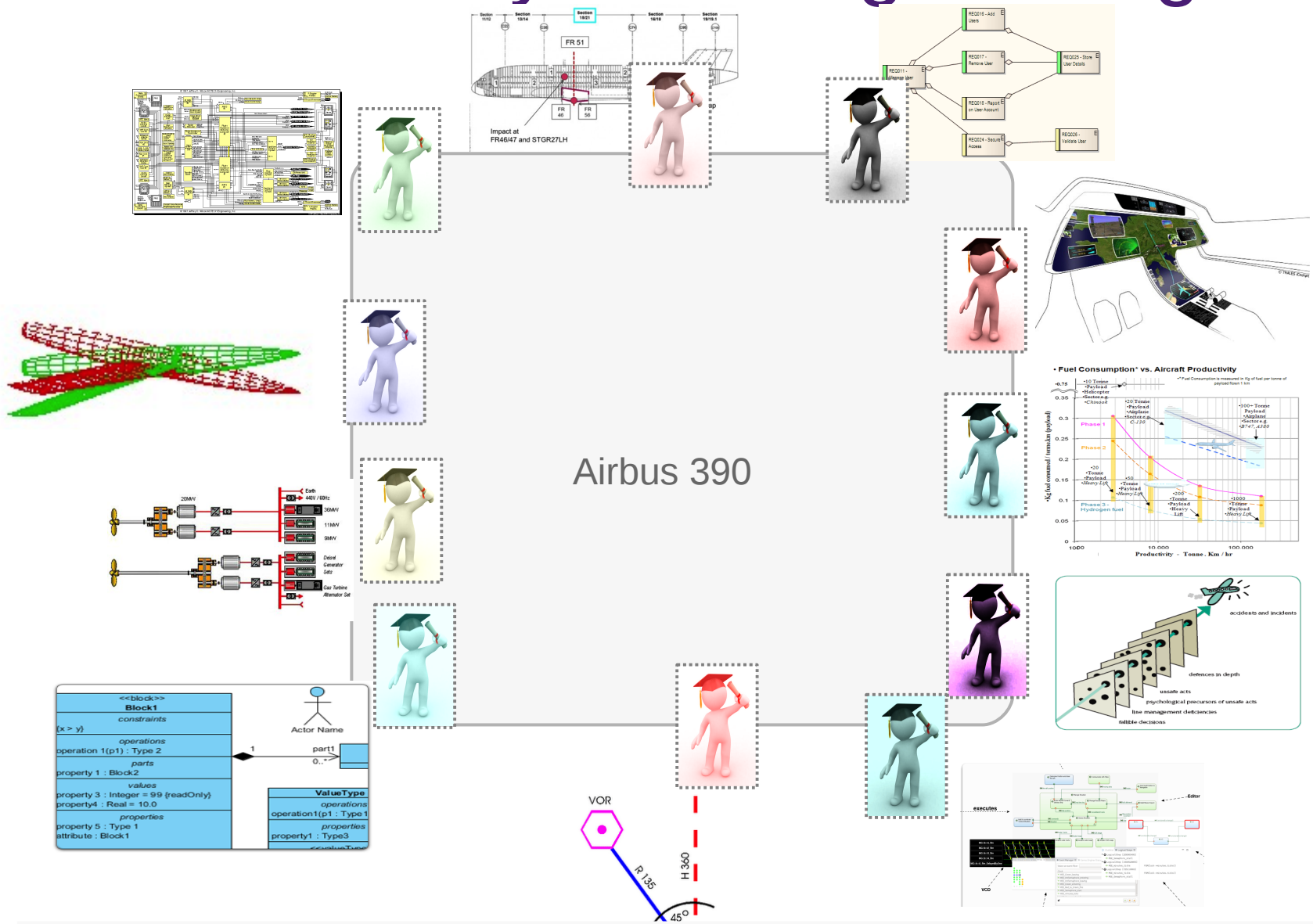


Several concerns for a single system

At least one expert by concern

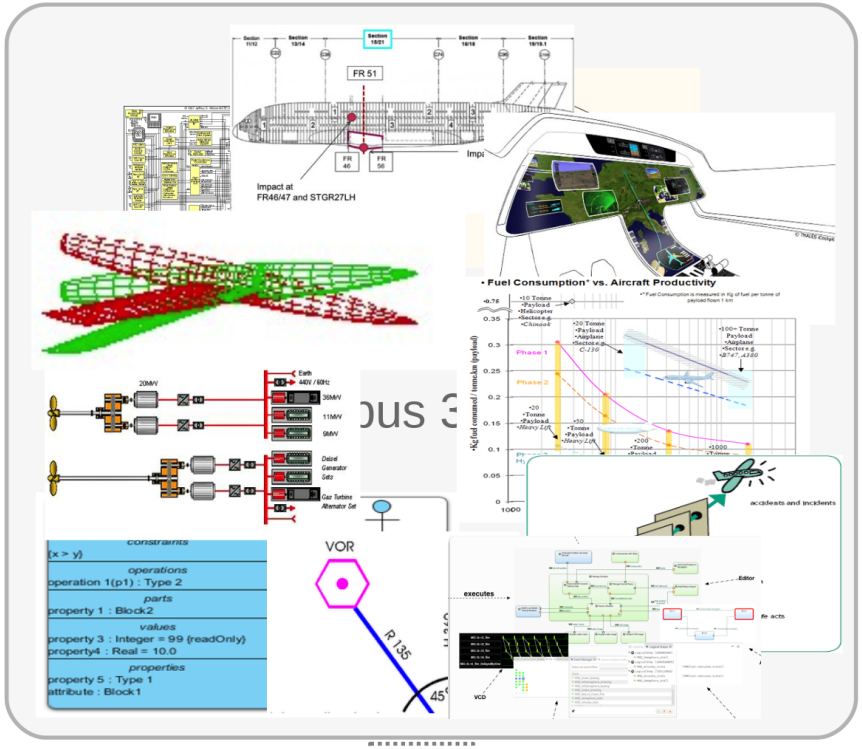
At least one “model” by concern (expressed in a DSML)

Model Based System Engineering



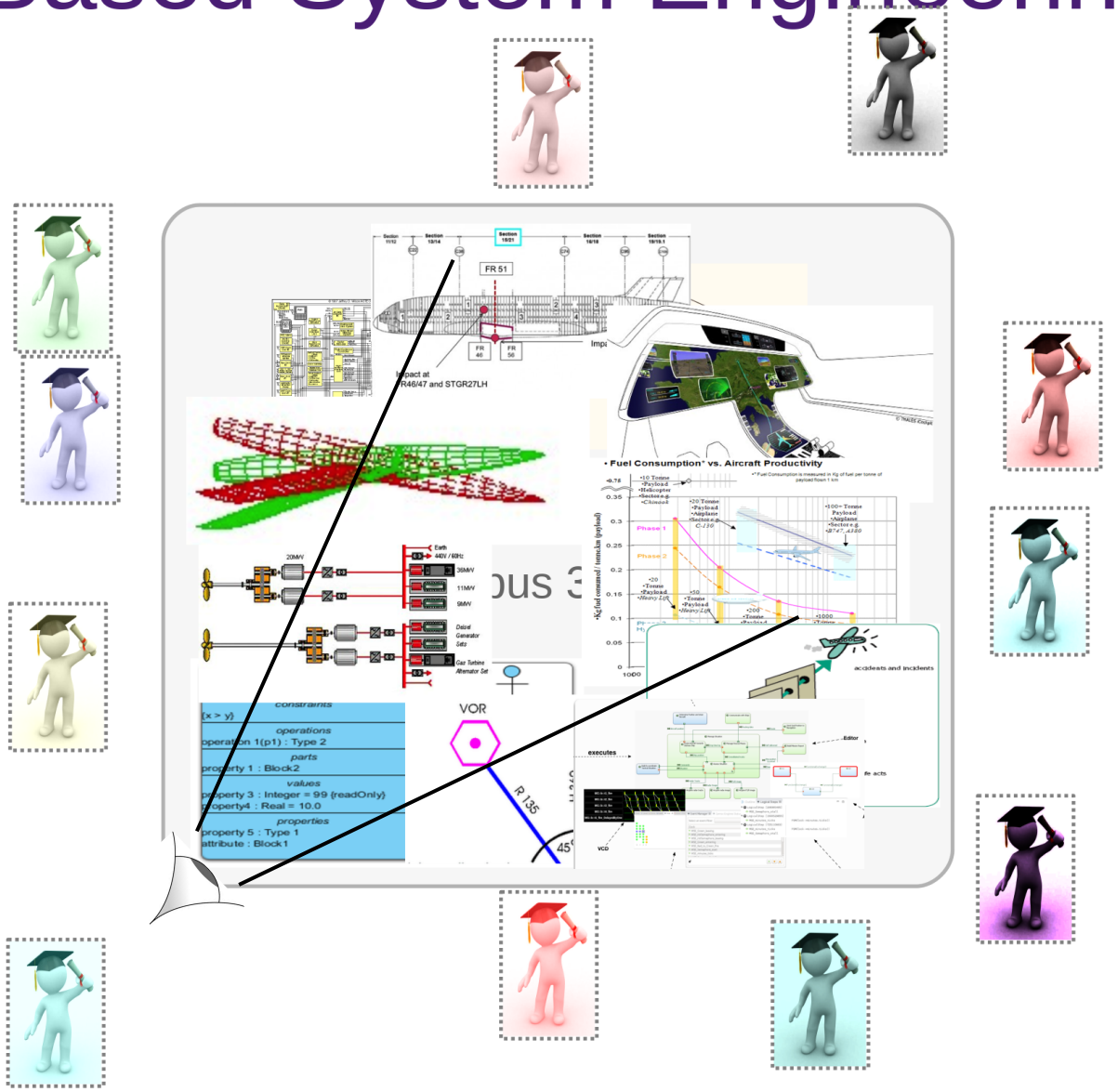
Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.

Model Based System Engineering



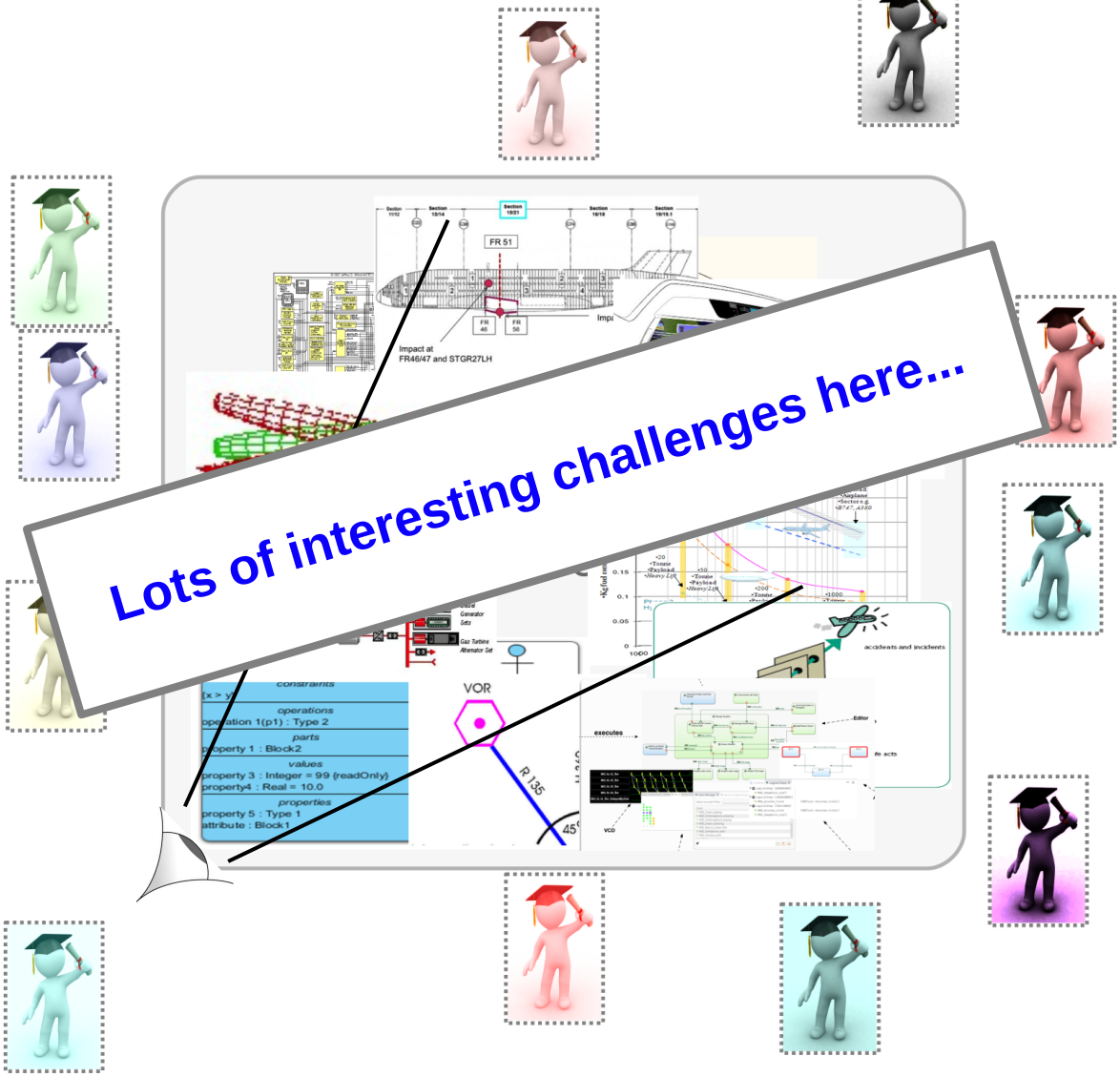
Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.

Model Based System Engineering



Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.

Model Based System Engineering



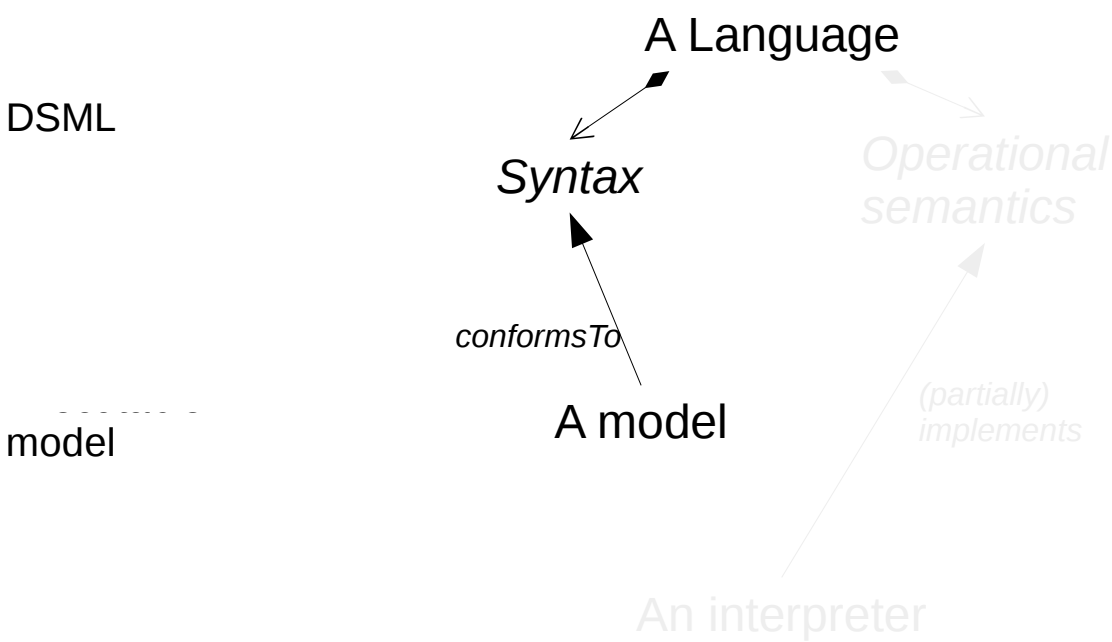
Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.

Model Based System Engineering

Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.

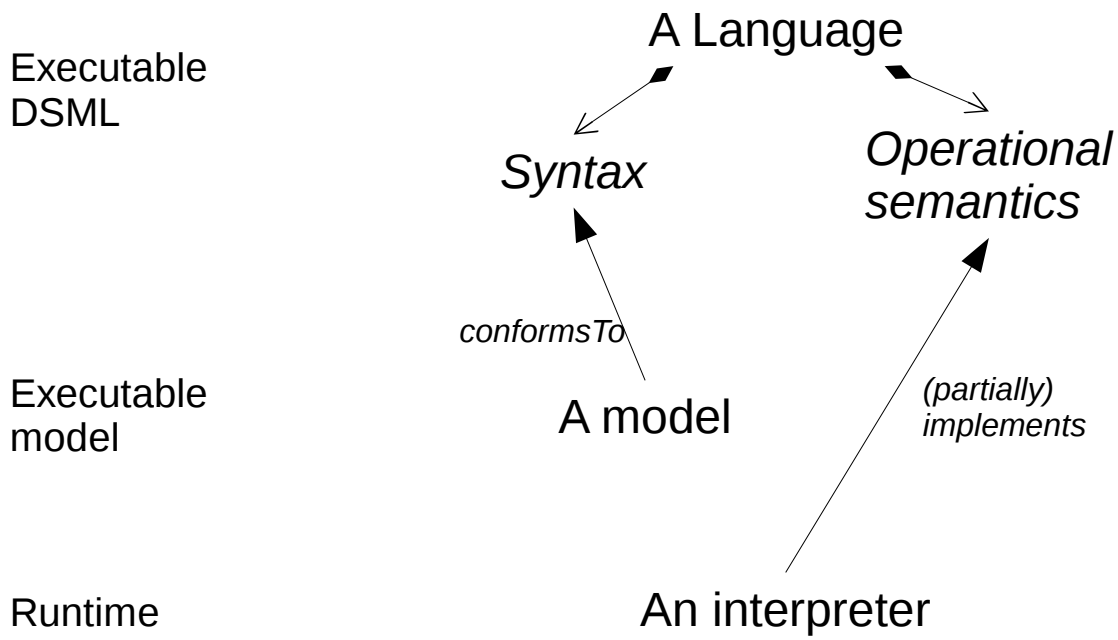
Context

- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



Context

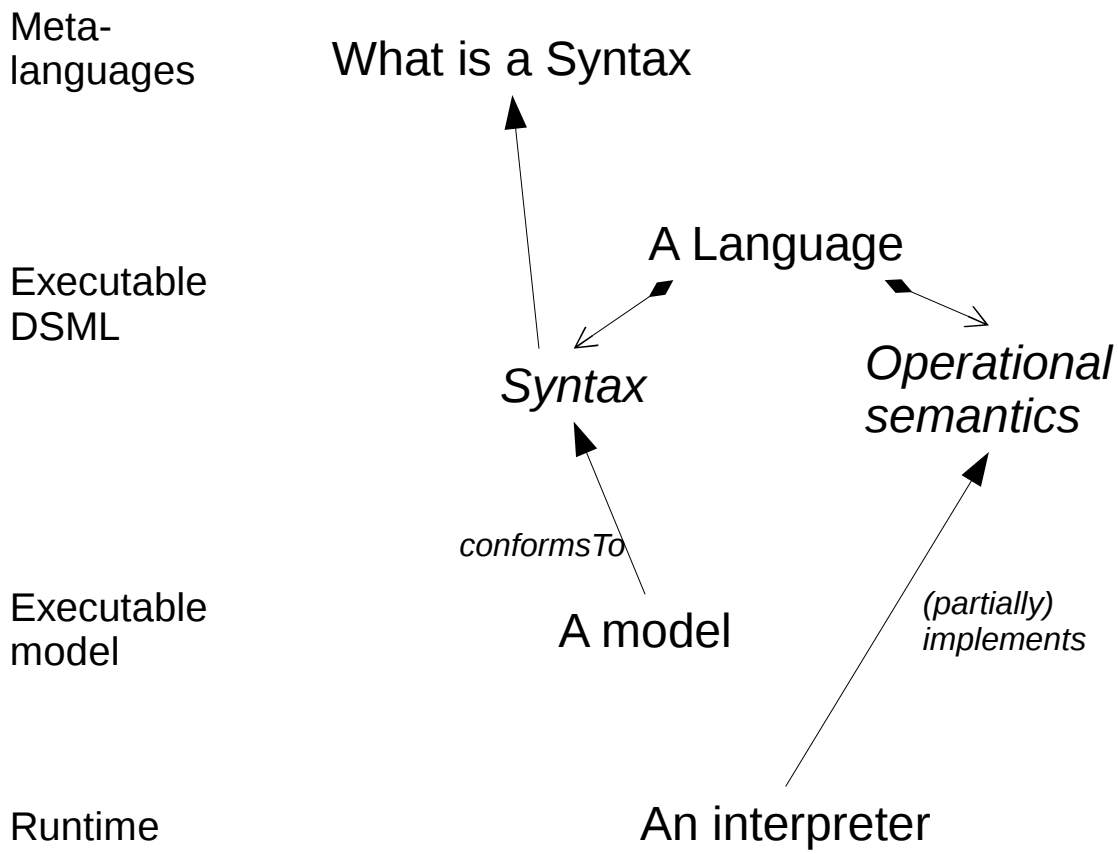
- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



We need to make the operational semantics explicit… and as formal as possible

Context

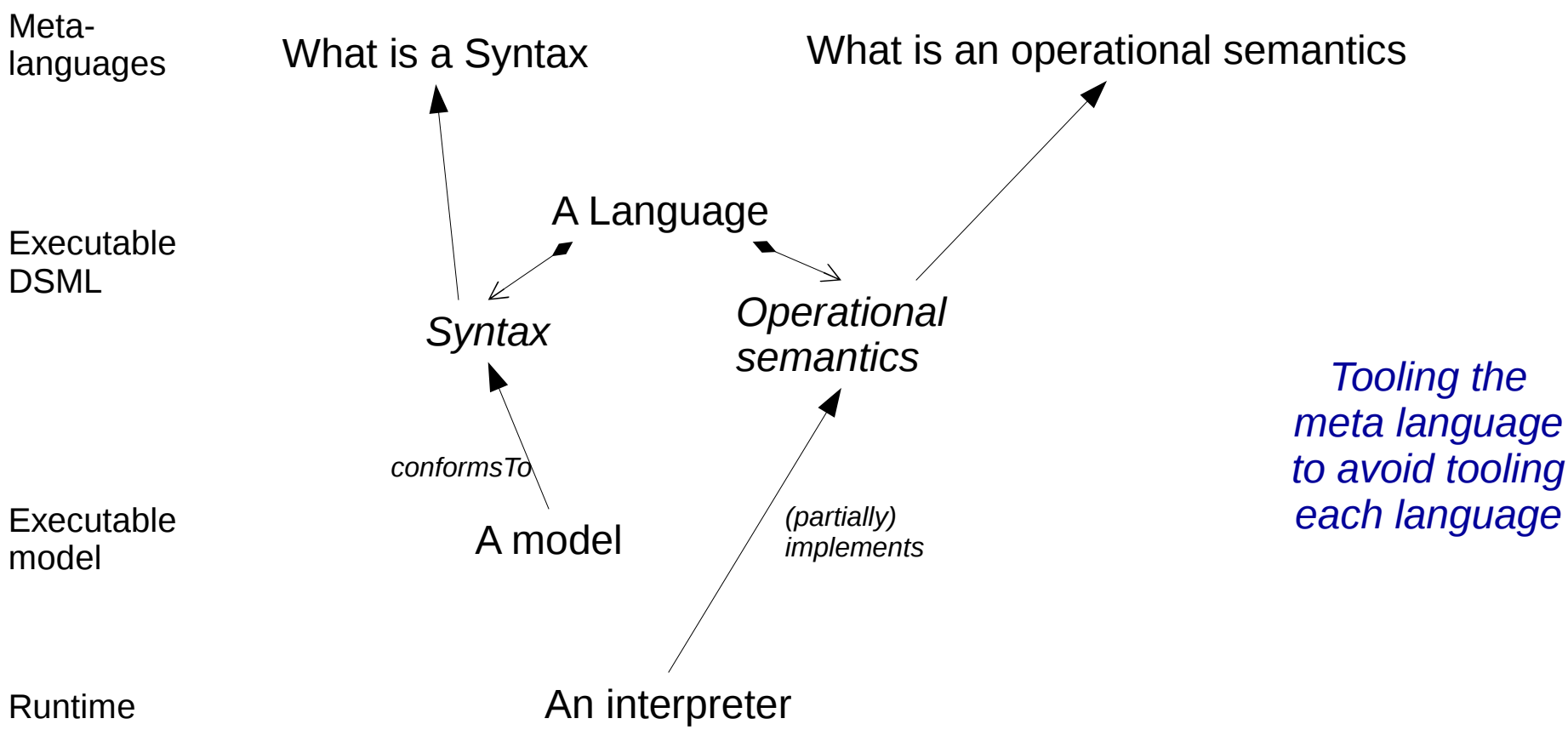
- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



Tooling the meta language to avoid tooling each language

Context

- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



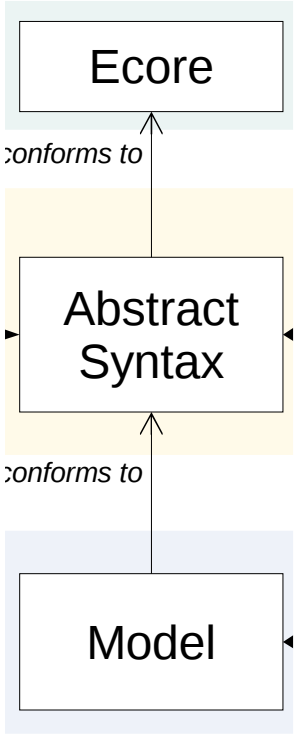
The GEMOC approach

Meta-languages

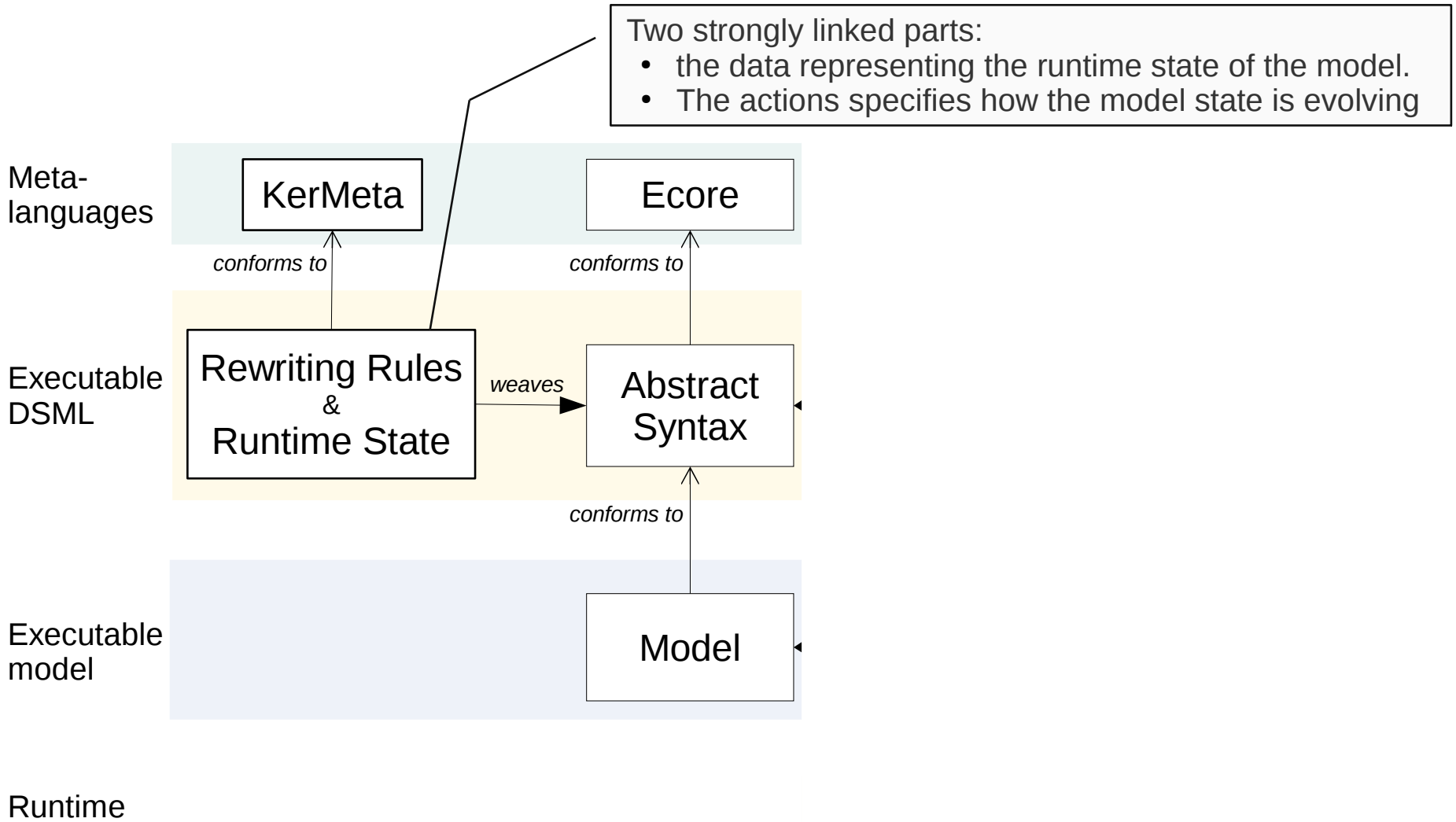
Executable DSML

Executable model

Runtime

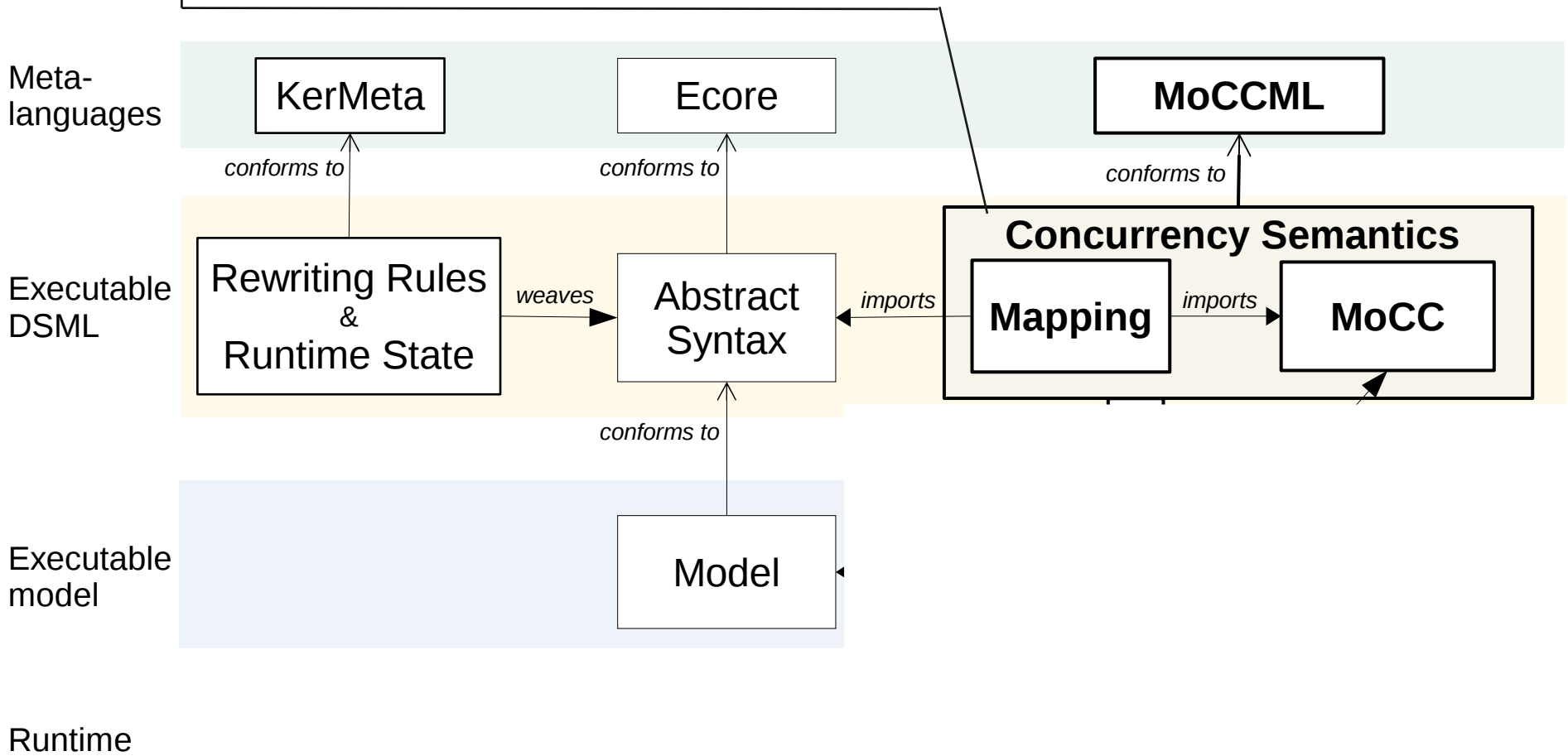


The GEMOC approach

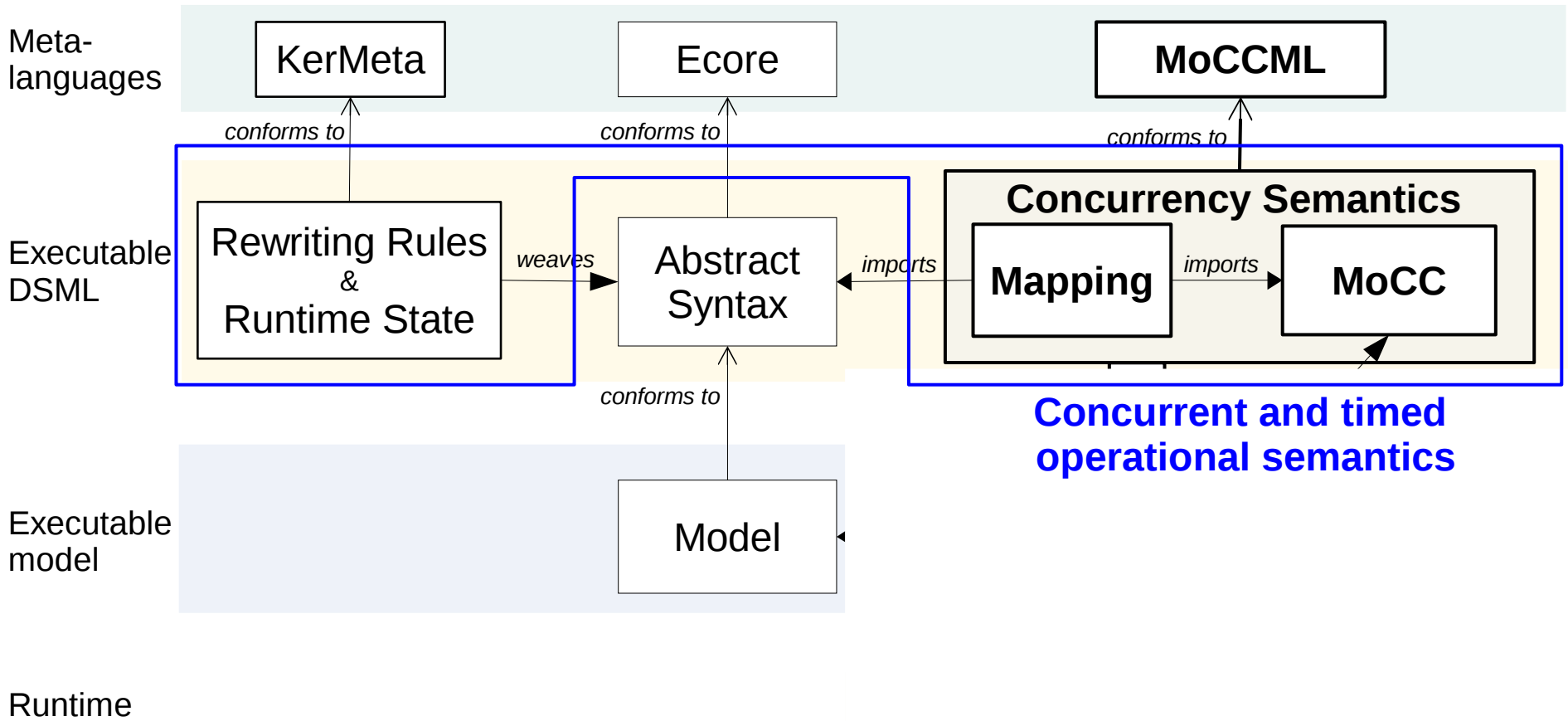


The GEMOC approach

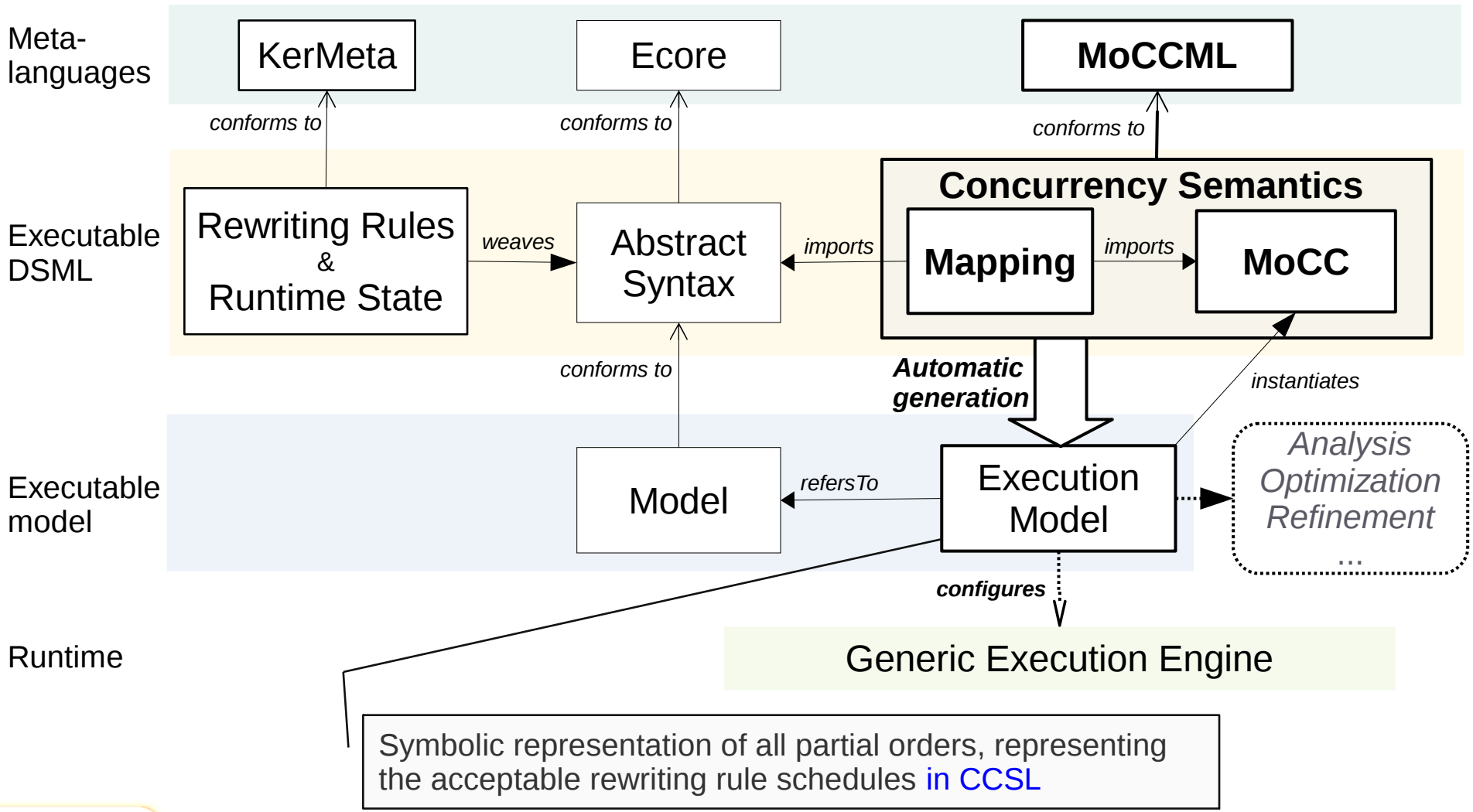
It Specifies **when** the rewriting rules that make the model evolving are called.
It models the (possibly timed) causalities and synchronizations between the rewriting rules



The GEMOC approach

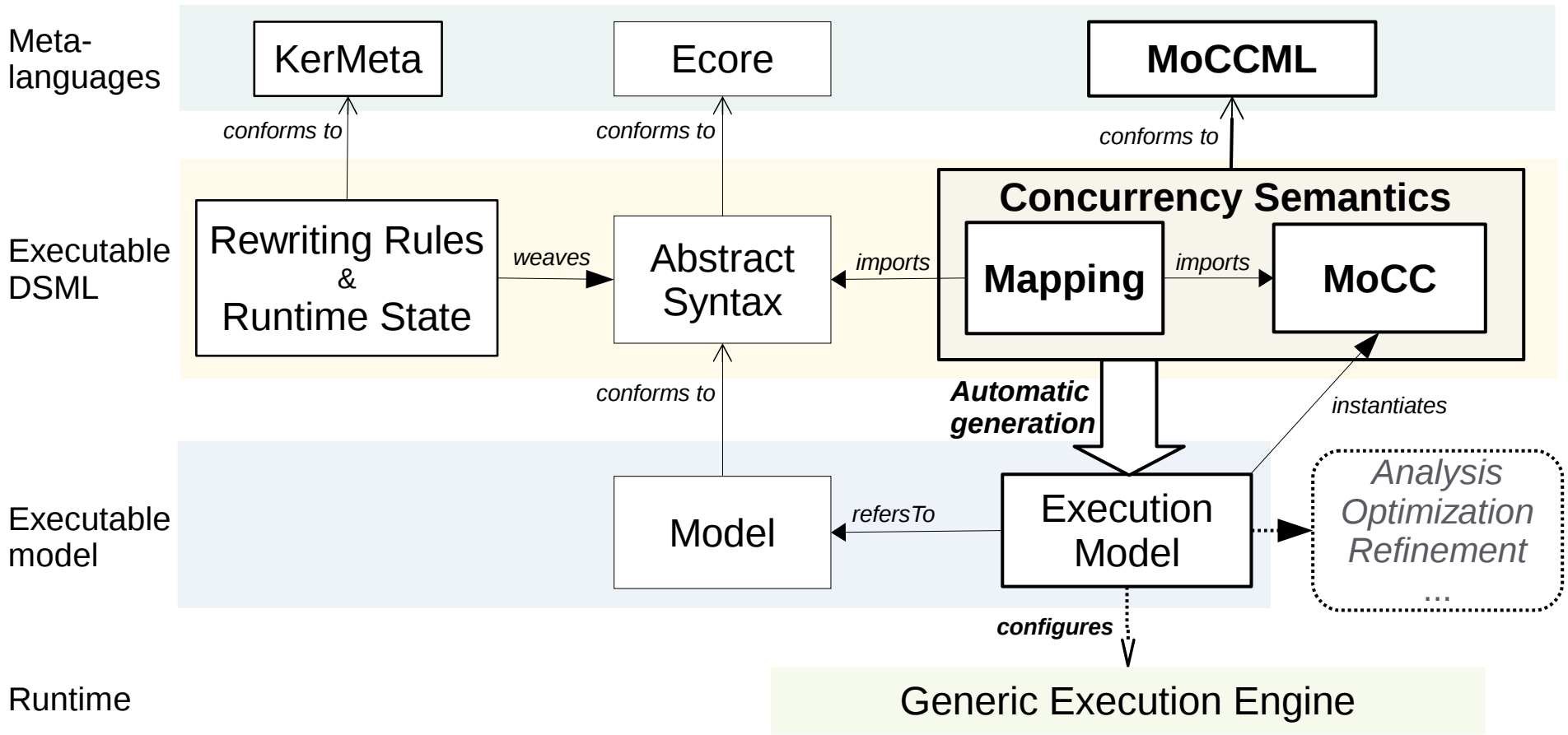


The GEMOC approach



The GEMOC approach

+ graphical concrete syntax in Sirius, which uses a meta-language as well



The GEMOC Studio

eclipse

Research Consortium

http://eclipse.org/gemoc

http://gemoc.org/studio

Language Workbench

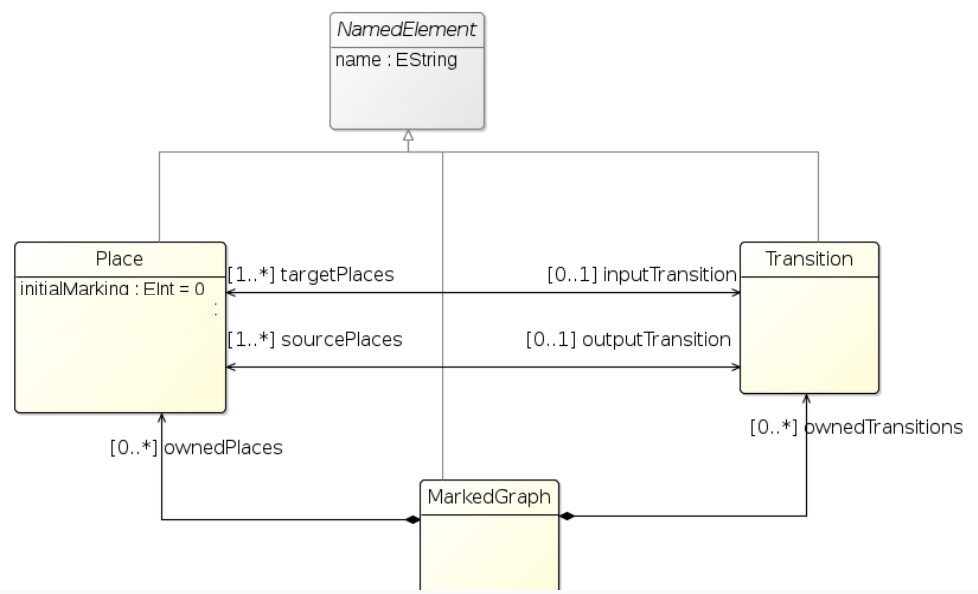
Modeling Workbench

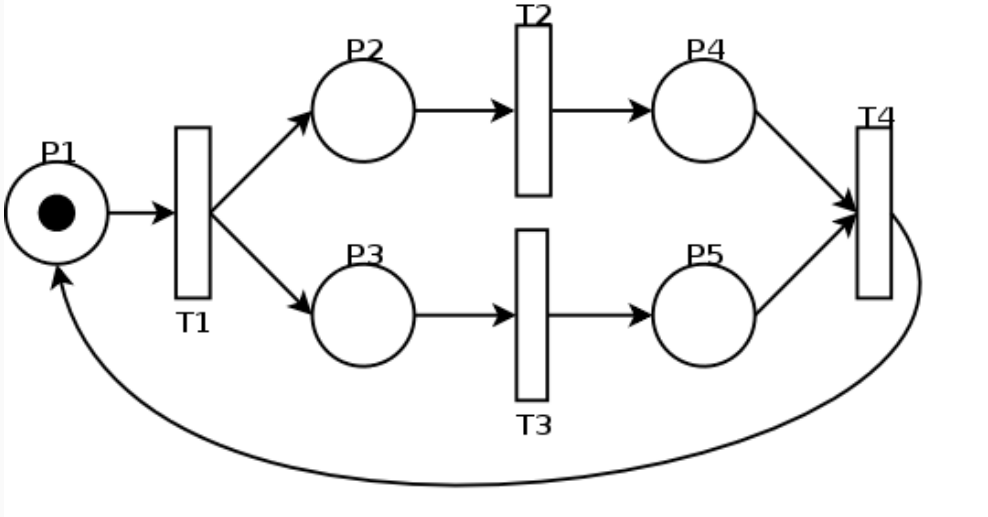
Design and compose your executable DSMLs

Edit and debug your heterogeneous models

Running example: AS

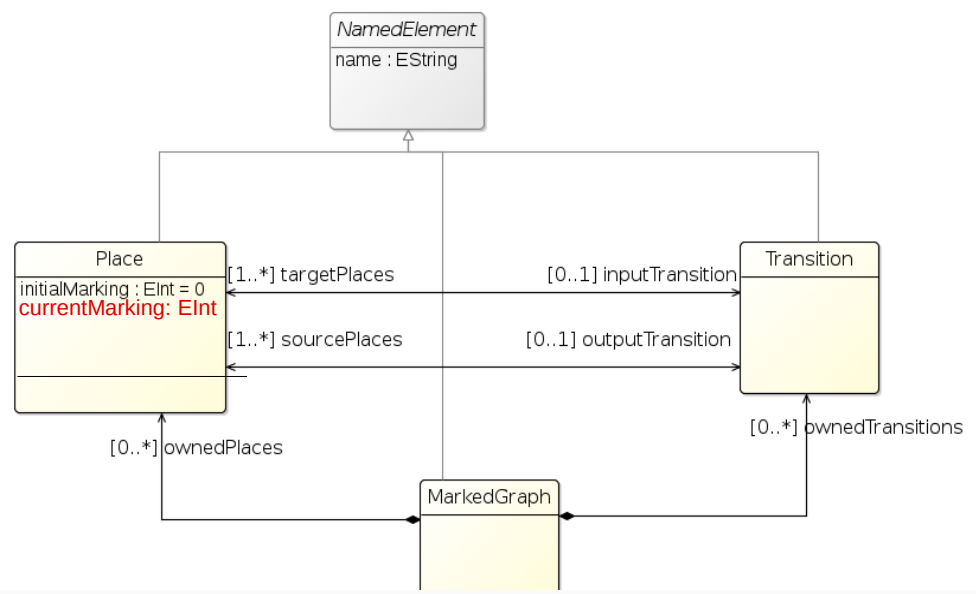
Ecore+Sirius

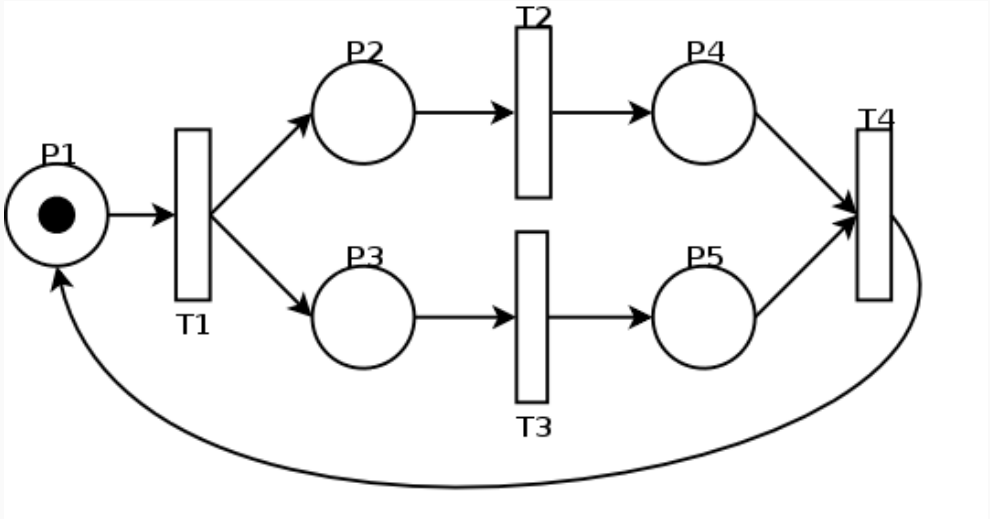




Running example: AS+DSA

Kermeta3





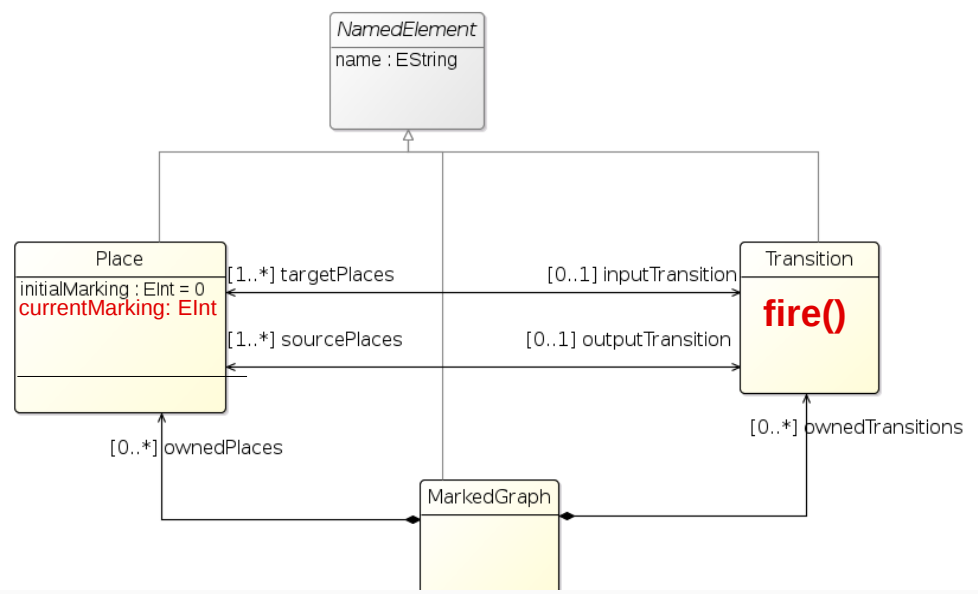
Domain Specific Action

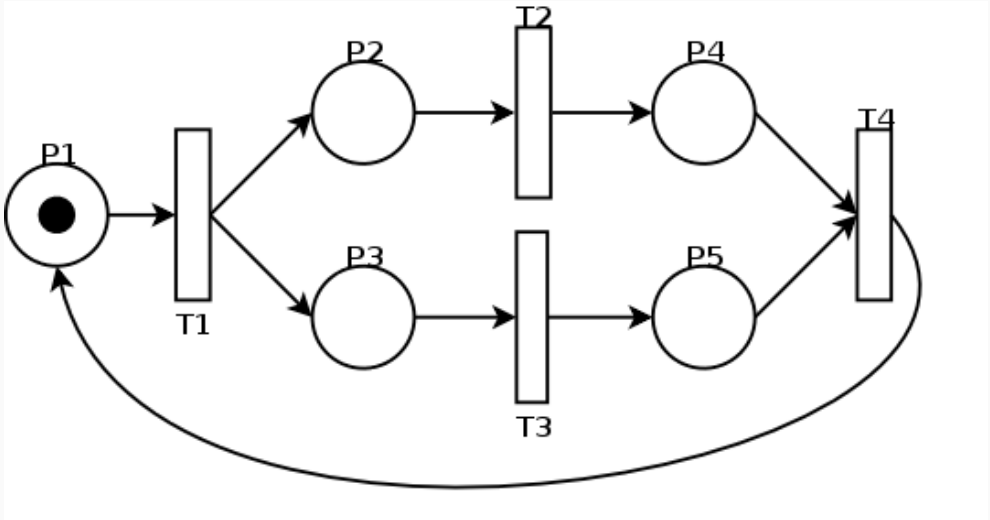
(model state)

- The ***current marking*** represents the runtime state of this simple language

Running example: AS+DSA

Kermeta3





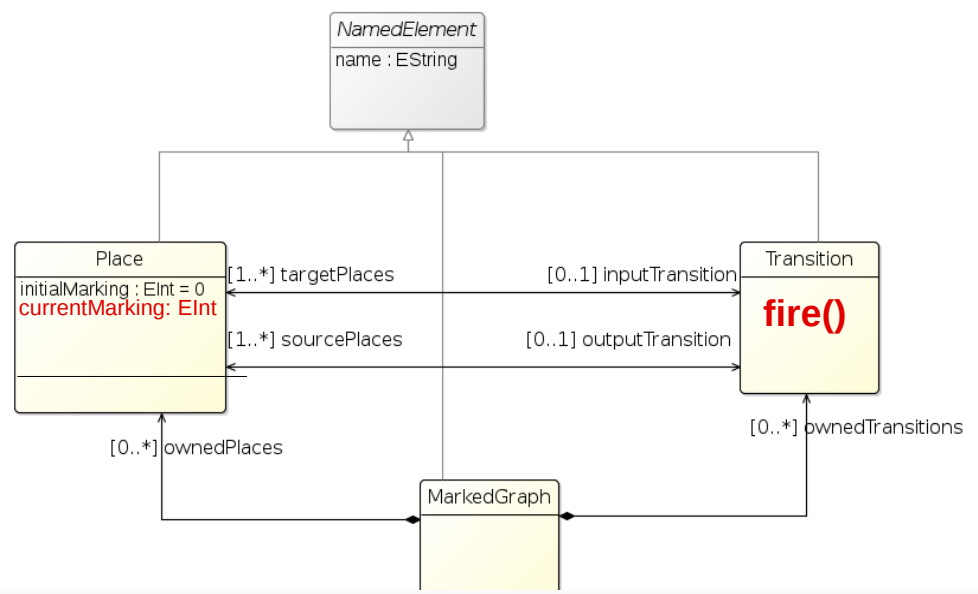
Domain Specific Action

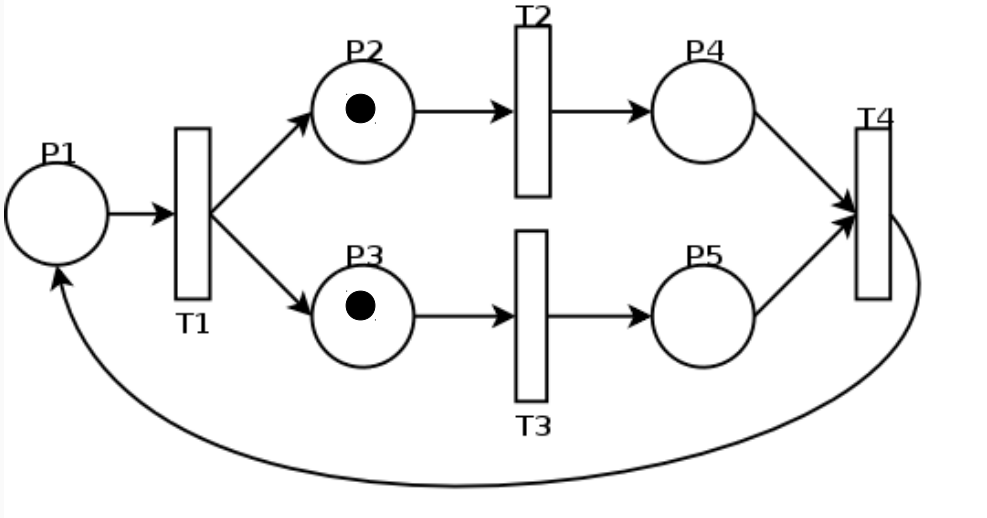
(execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
            currentMarking – –
        ]
    _self.targetPlaces.forEach [
            currentMarking ++
        ]
}
```

Running example: AS+DSA

Kermeta3





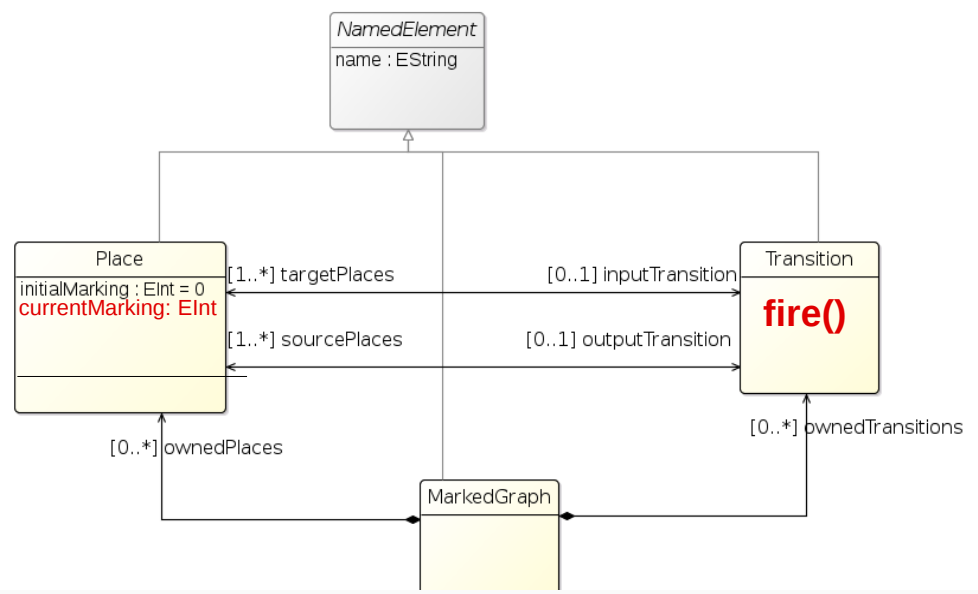
Domain Specific Action

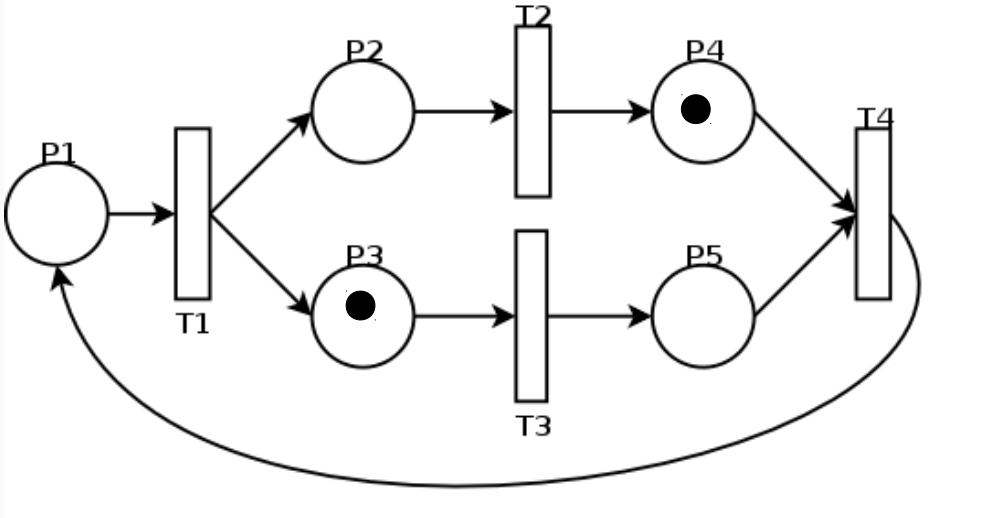
(execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
            currentMarking – –
        ]
    _self.targetPlaces.forEach [
            currentMarking ++
        ]
}
```

Running example: AS+DSA

Kermeta3



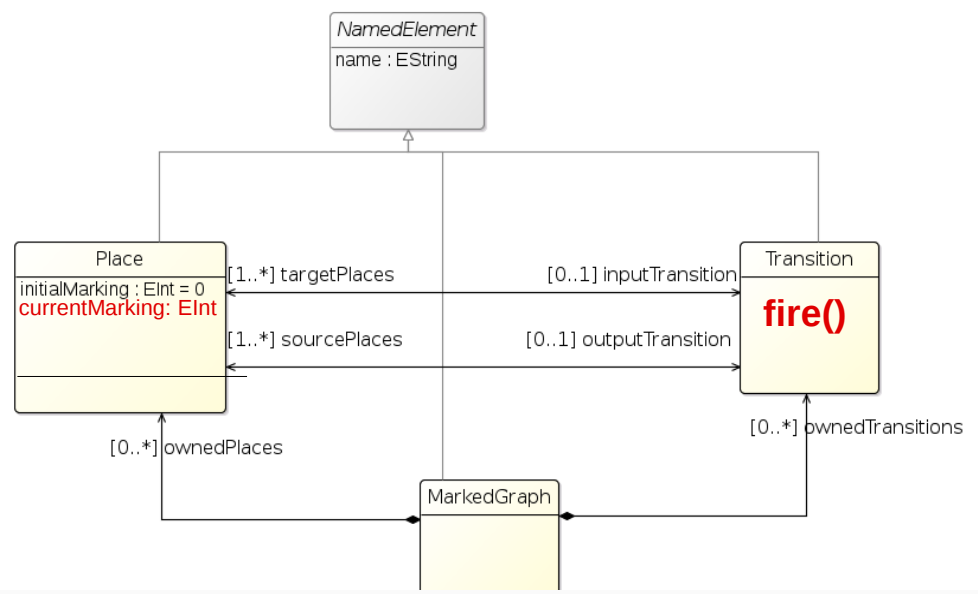


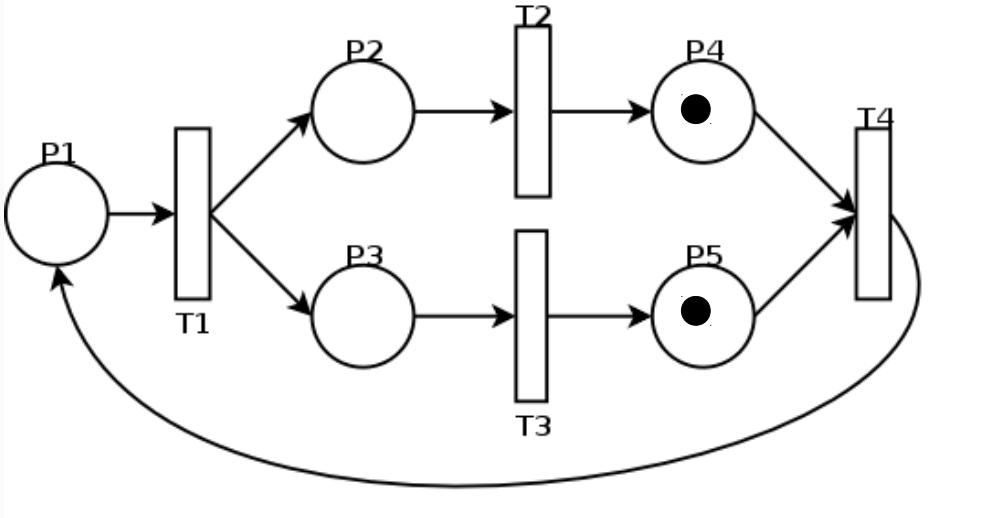
Domain Specific Action (execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
                currentMarking – –
            ]
    _self.targetPlaces.forEach [
                currentMarking ++
            ]
  }
```

Running example: AS+DSA

Kermeta3



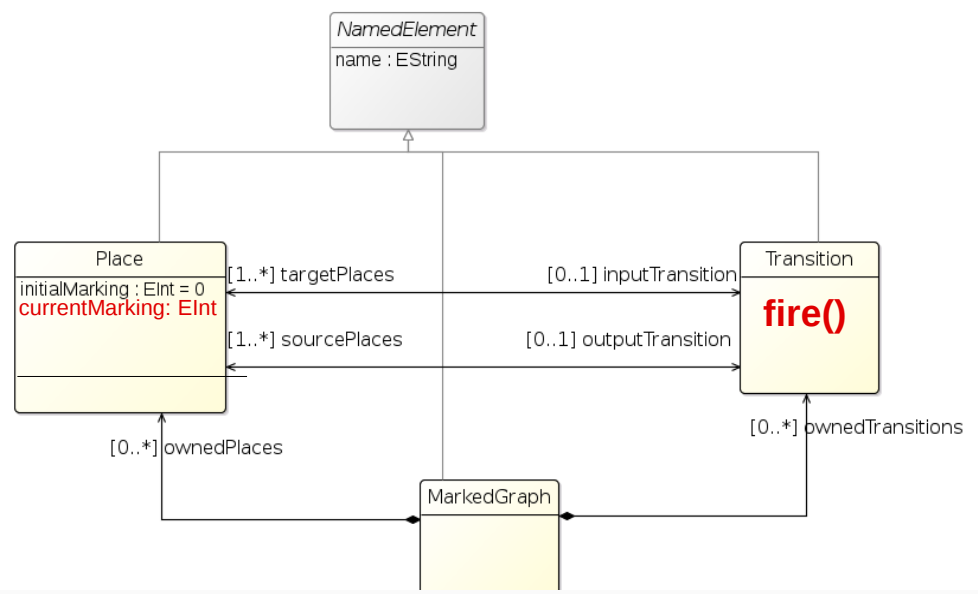


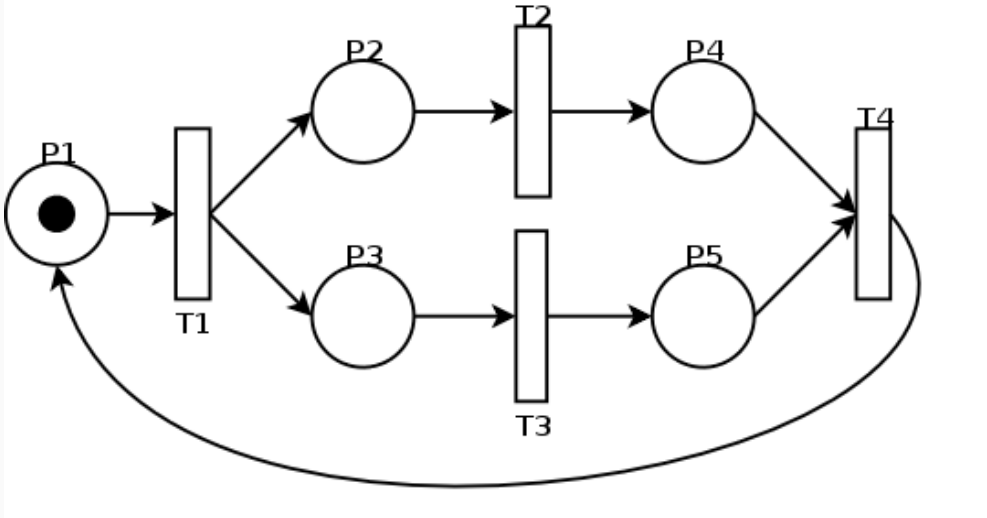
Domain Specific Action (execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
                currentMarking – –
            ]
    _self.targetPlaces.forEach [
                currentMarking ++
            ]
  }
```

Running example: AS+DSA

Kermeta3





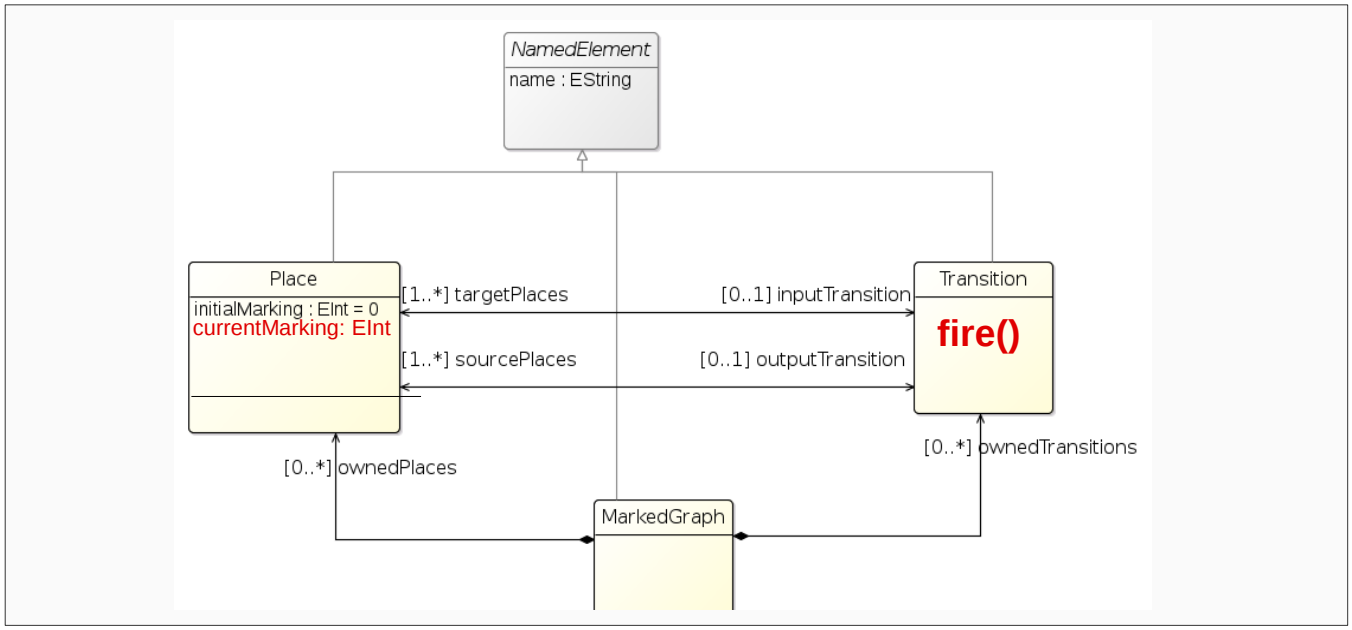
Domain Specific Action

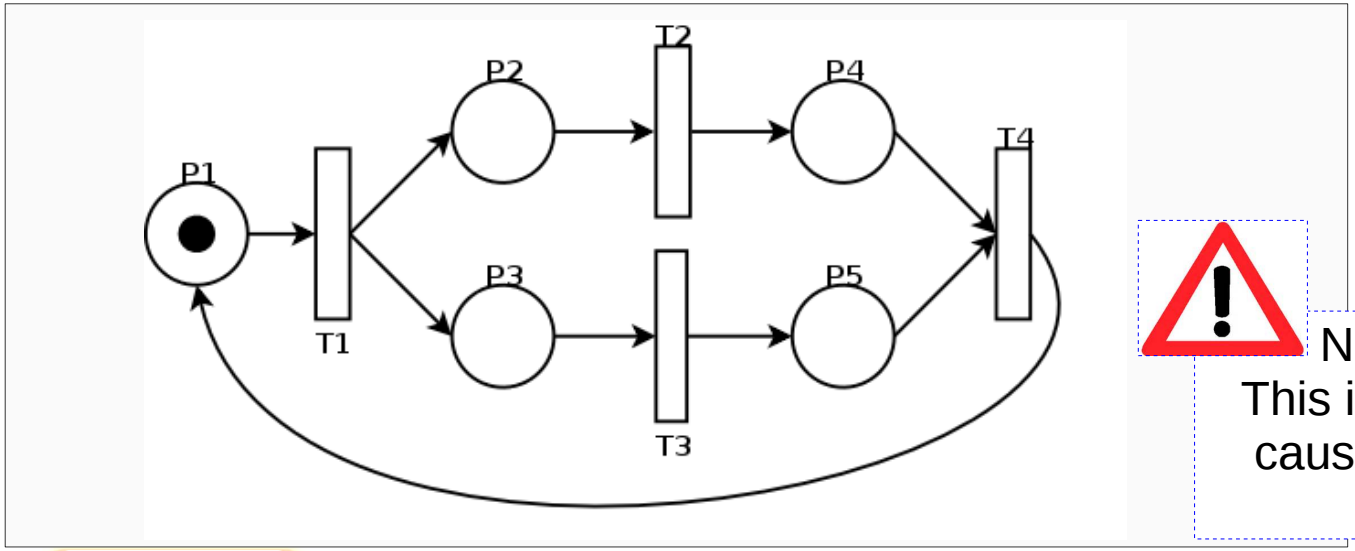
(execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
                currentMarking – –
            ]
    _self.targetPlaces.forEach [
                currentMarking ++
            ]
  }
```

Running example: AS+DSA

Kermeta3





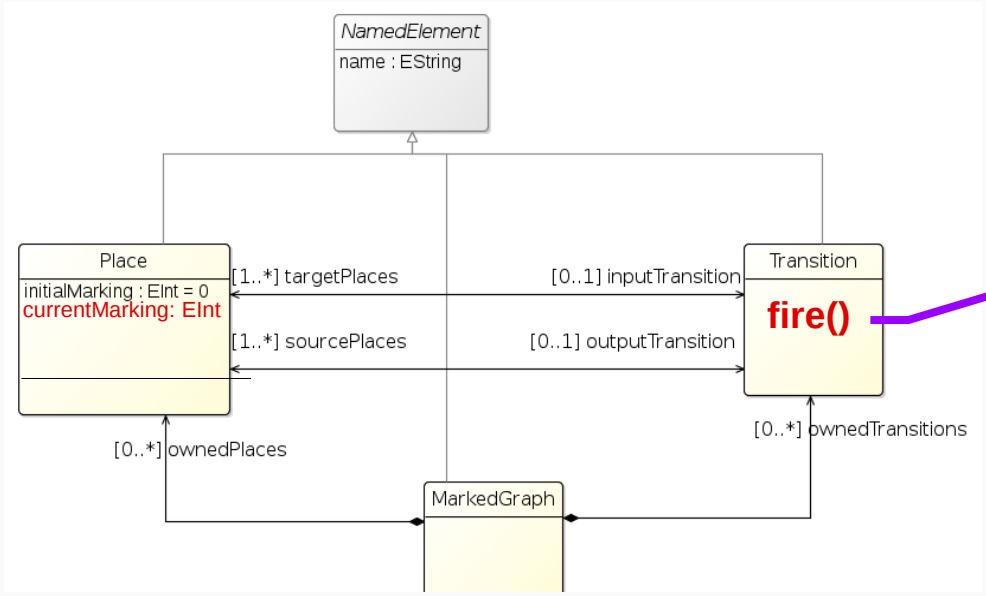
Domain Specific Action (execution functions)

```
def fire(){
    _self.sourcePlaces.forEach [
                currentMarking – –
            ]
    _self.targetPlaces.forEach [
                currentMarking ++
            ]
  }
```

Nobody calls the *fire()* operation. This is the model of concurrency and causality that specifies **when** things happen

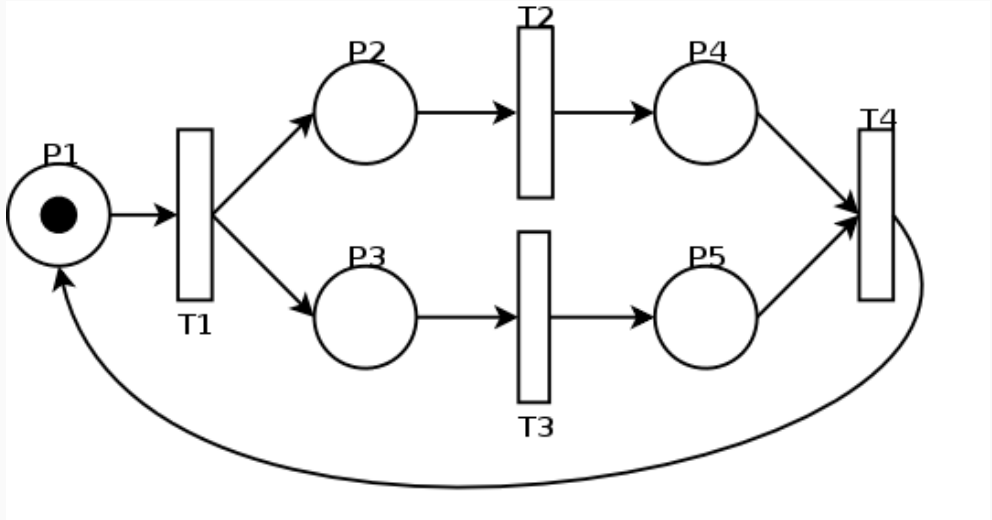
Running example: AS+DSA+DSE

ECL



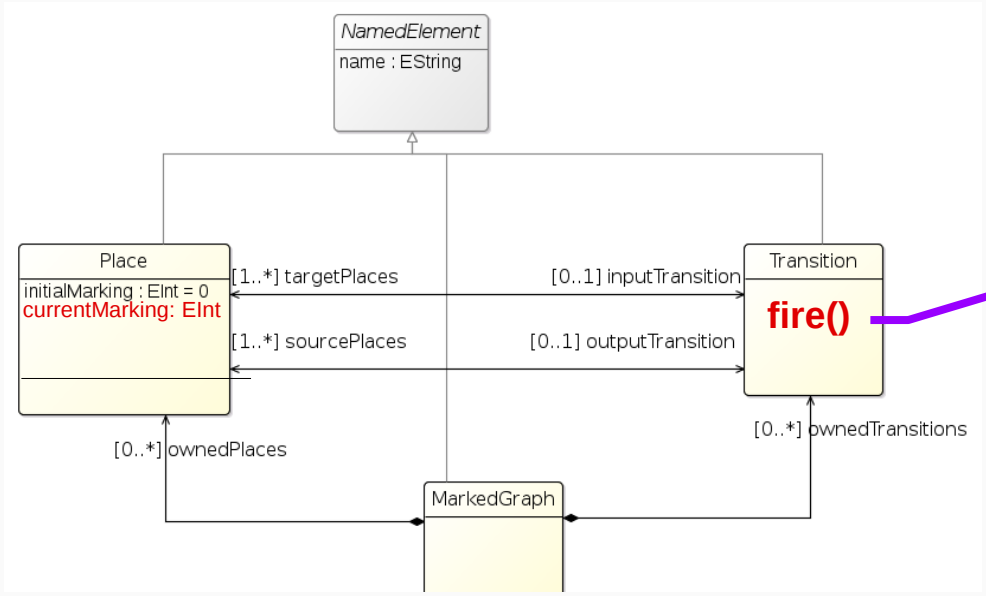
fireIt: DSE

Domain Specific Events act as “handles” to the DSA



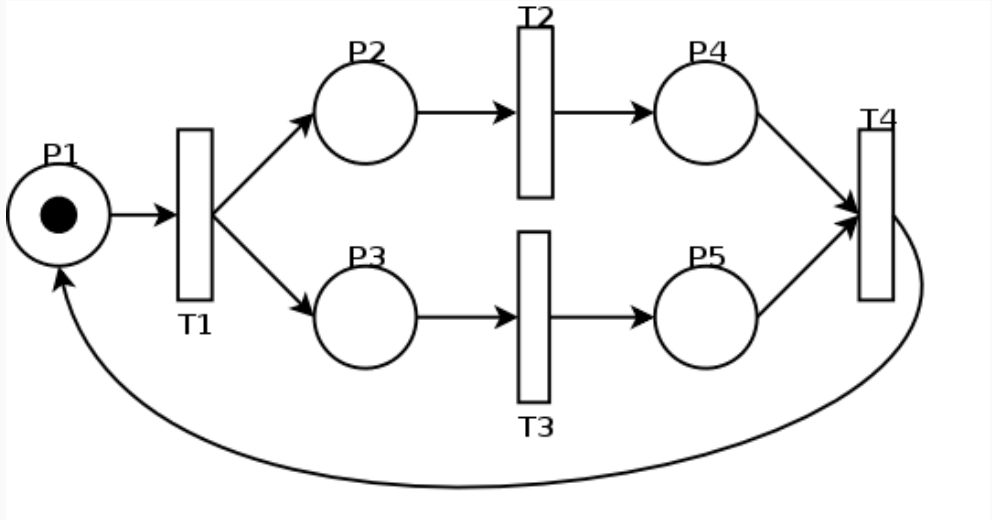
Running example: AS+DSA+DSE

MoCCML mapping (ex ECL)



- fireIt: DSE

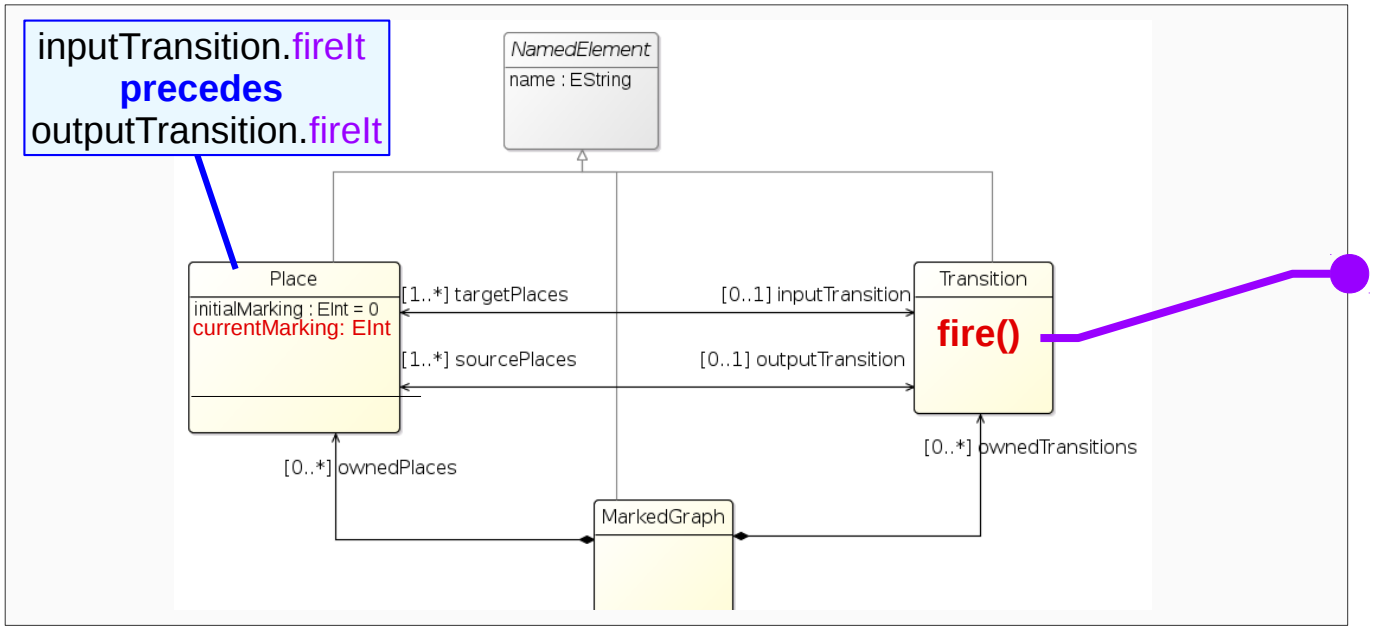
Domain Specific Events act as “handles” to the DSA



- fire_T1: fireIt
- fire_T2: fireIt
- fire_T3: fireIt
- fire_T4: fireIt

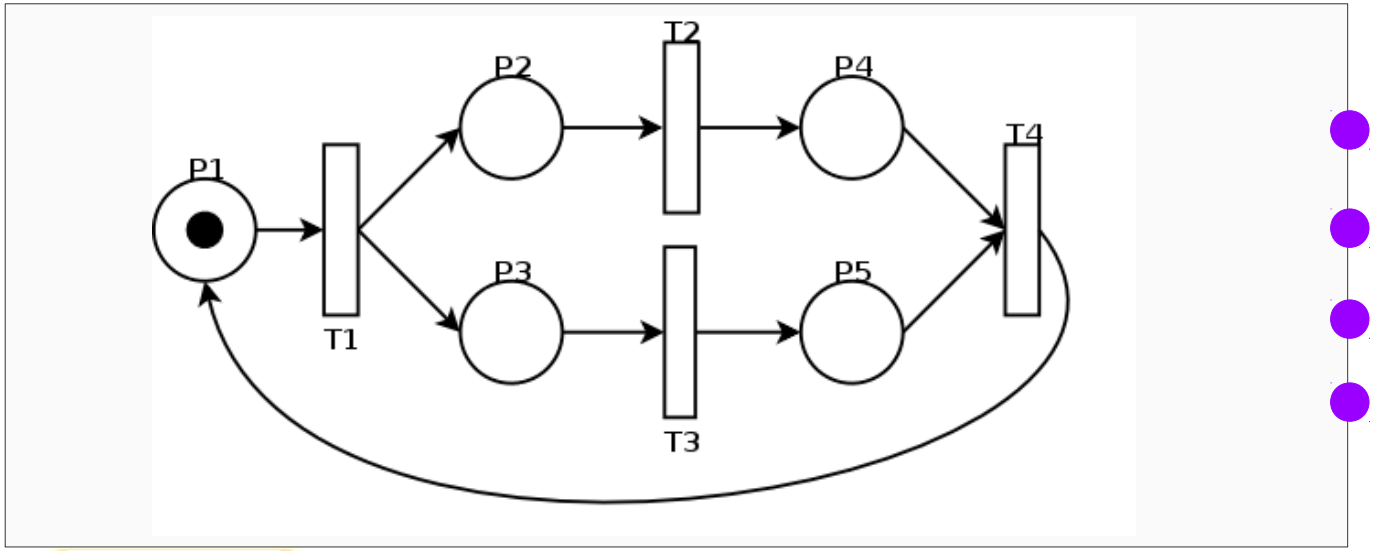
Running example: AS+DSA+DSE+MoCC

MoCCML



- fireIt: DSE

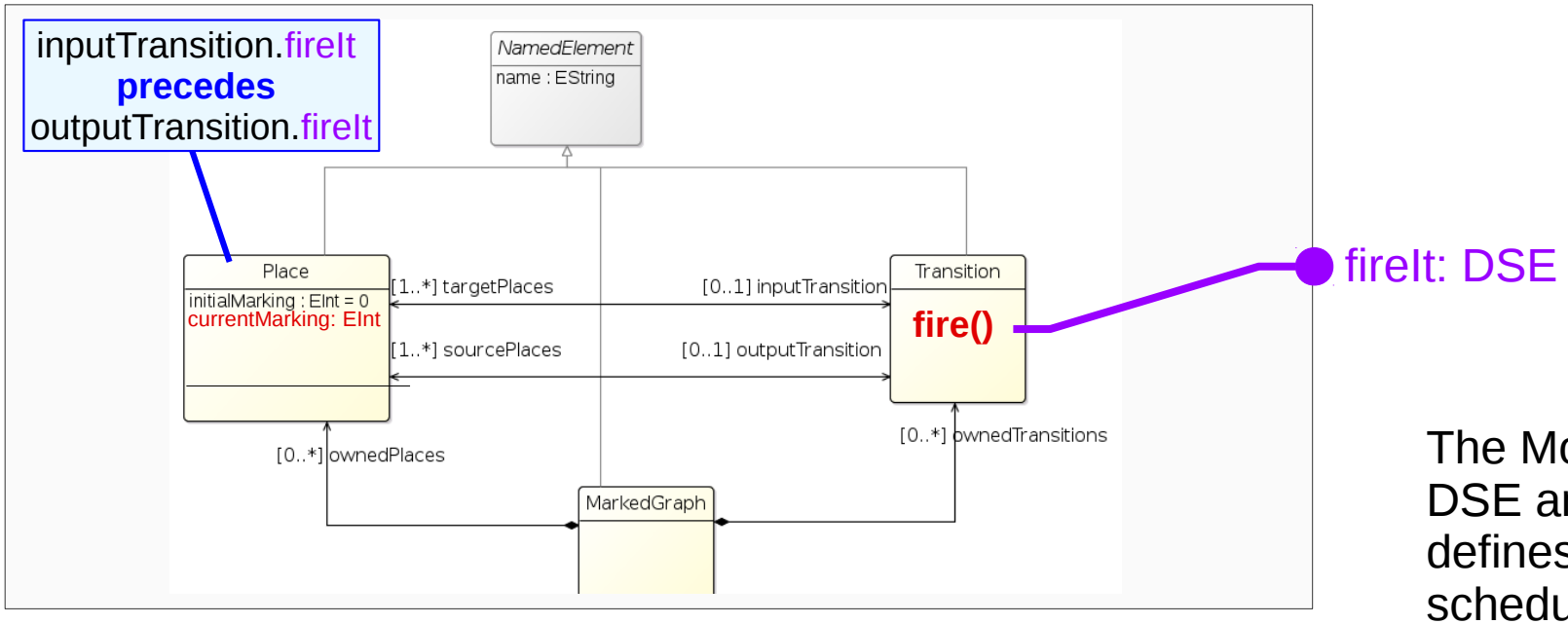
The MoCC constrains the DSE and consequently defines the acceptable schedules of the actions



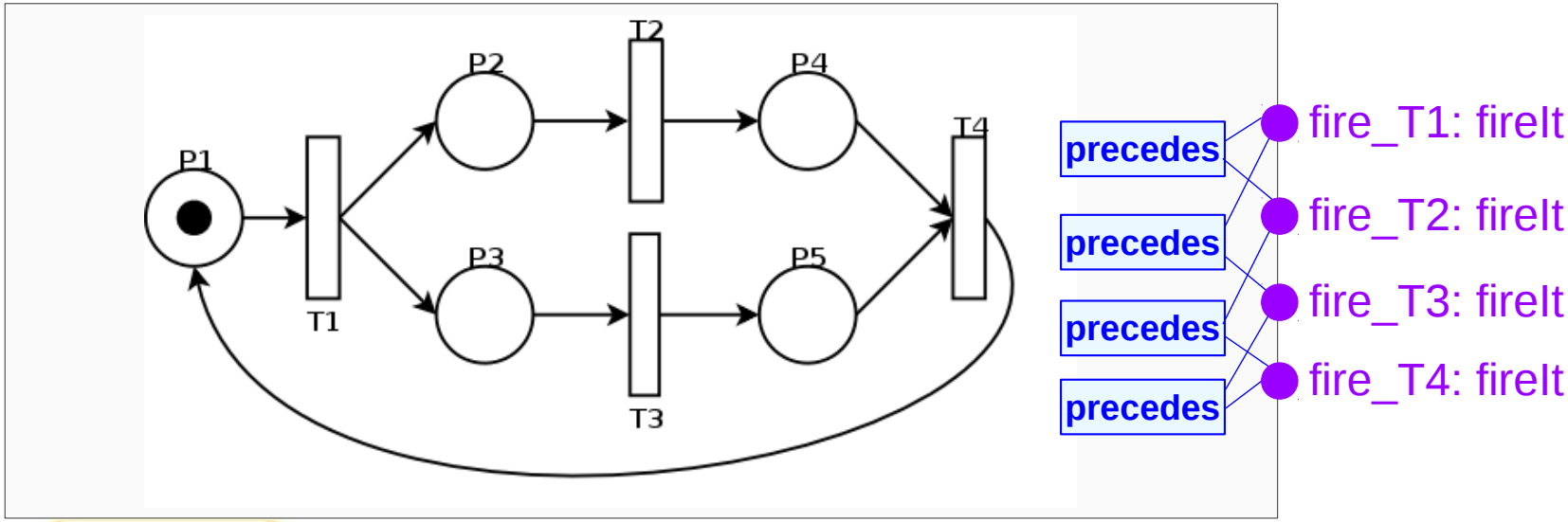
- fire_T1: fireIt
- fire_T2: fireIt
- fire_T3: fireIt
- fire_T4: fireIt

Running example: AS+DSA+DSE+MoCC

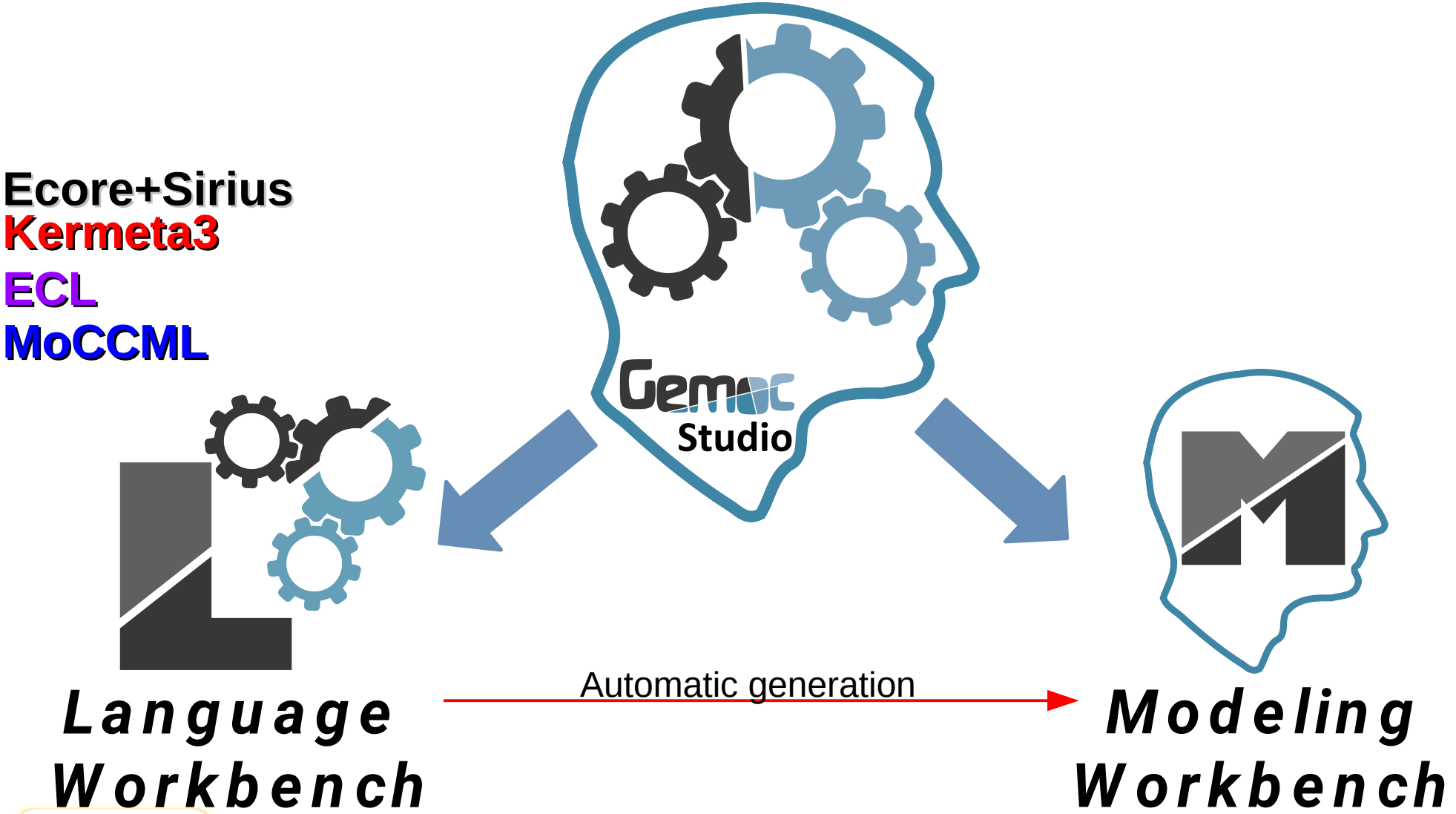
MoCCML



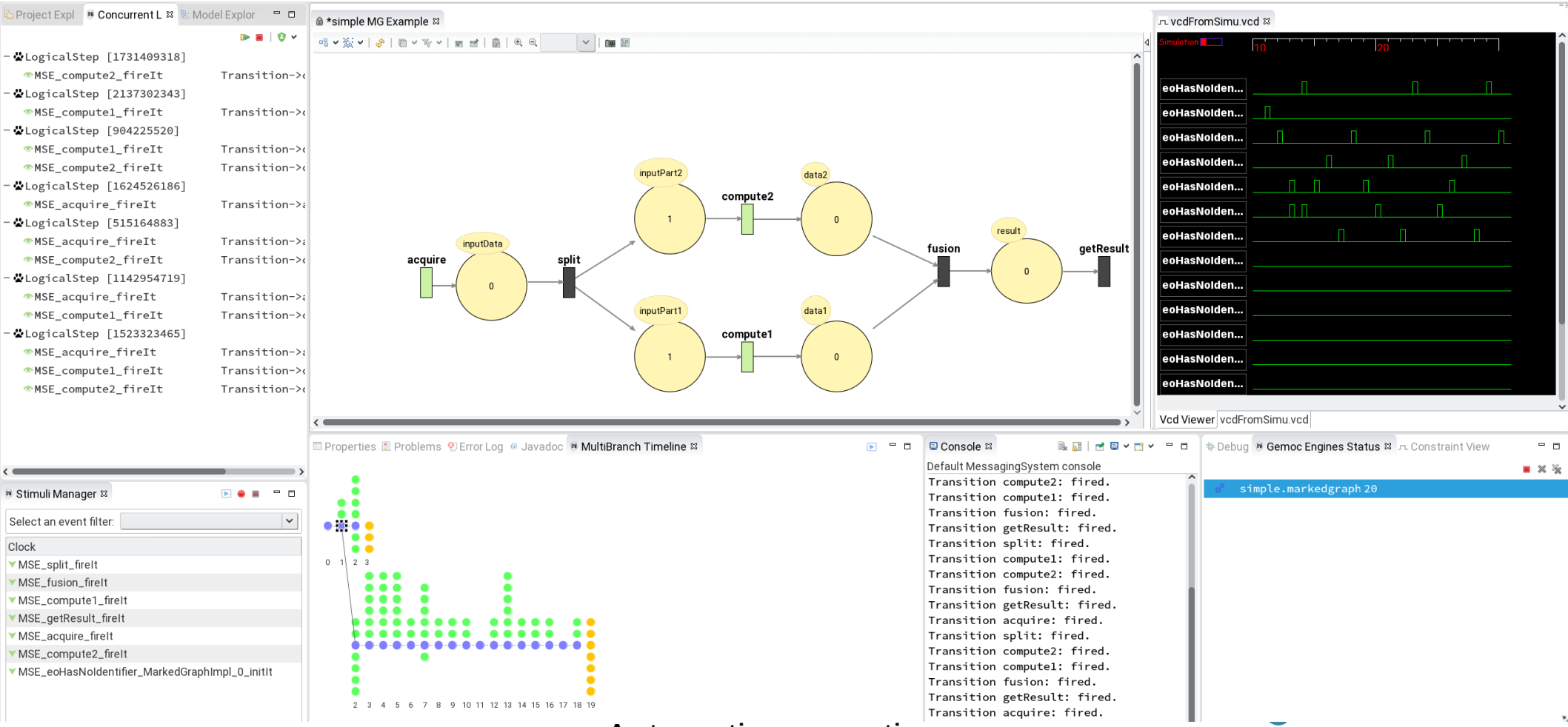
The MoCC constrains the DSE and consequently defines the acceptable schedules of the actions



The GEMOC Studio

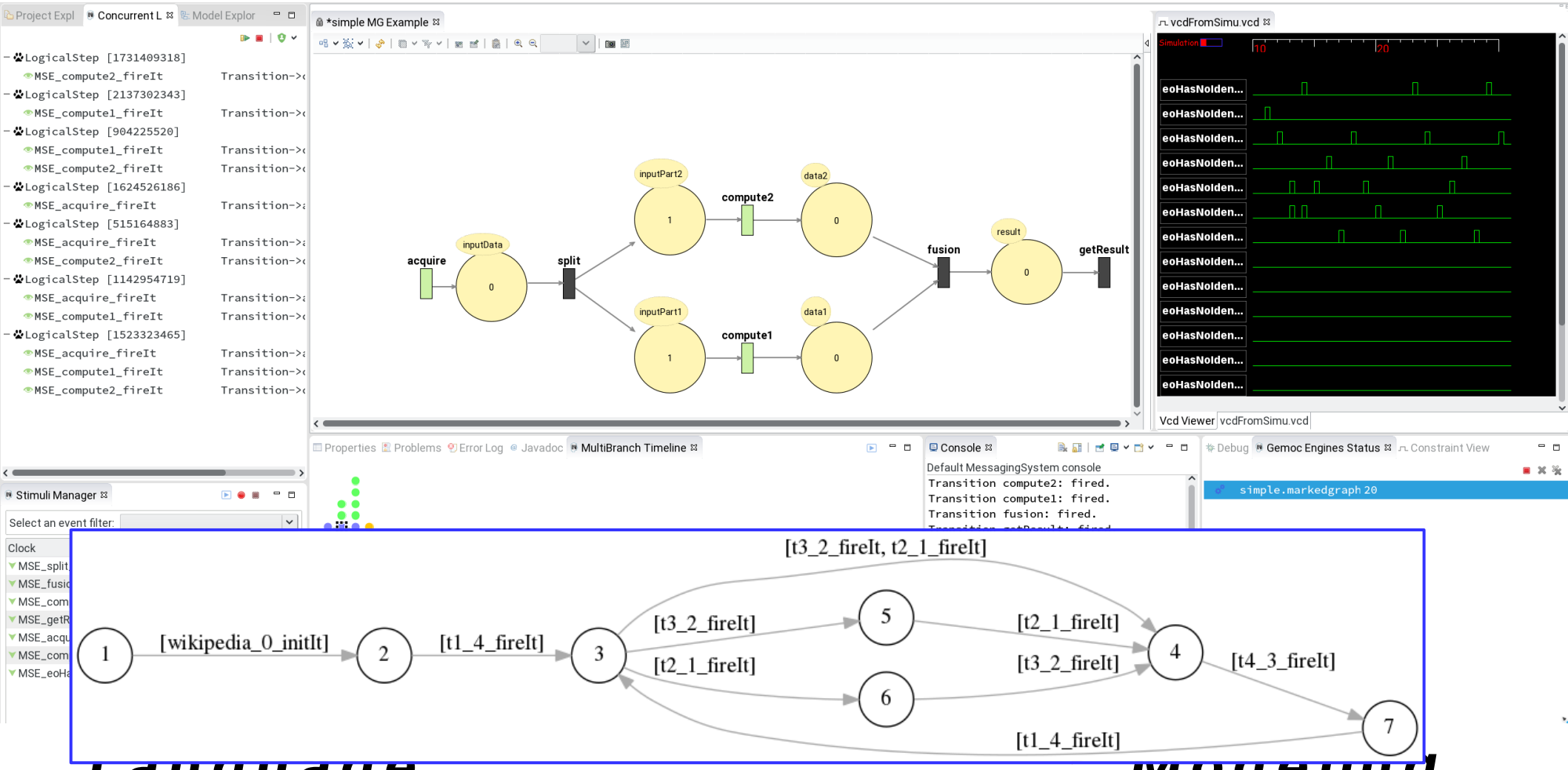


The GEMOC Studio



Language Workbench → Automatic generation → **Modeling Workbench**

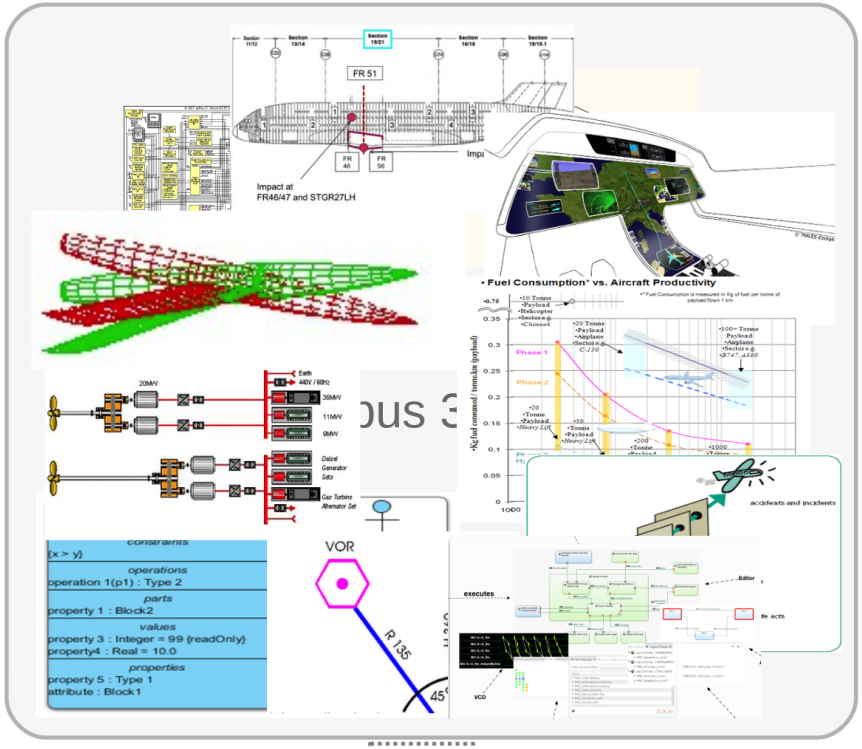
The GEMOC Studio



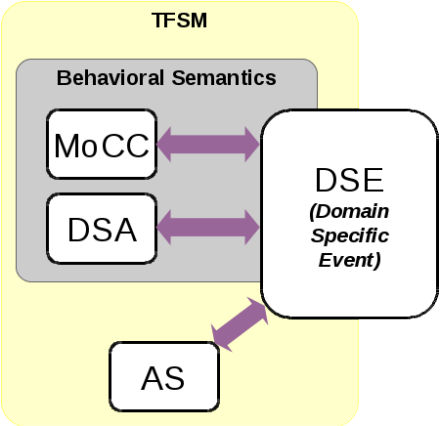
Language Workbench

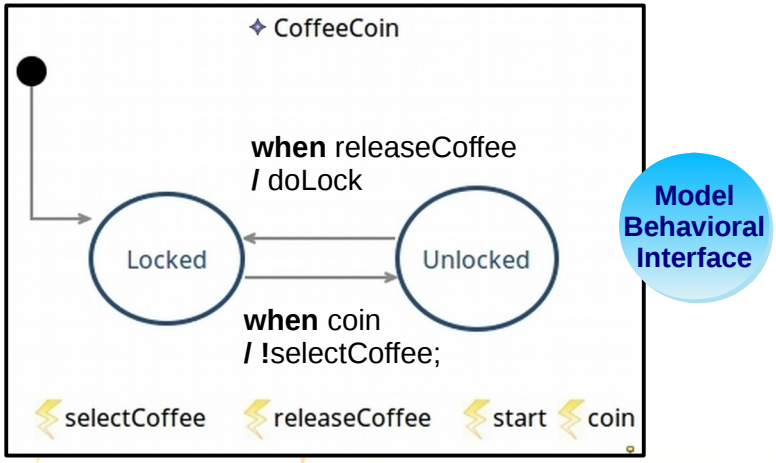
Modeling Workbench

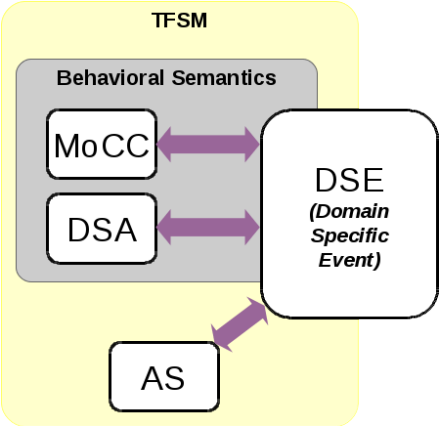
Model Based System Engineering

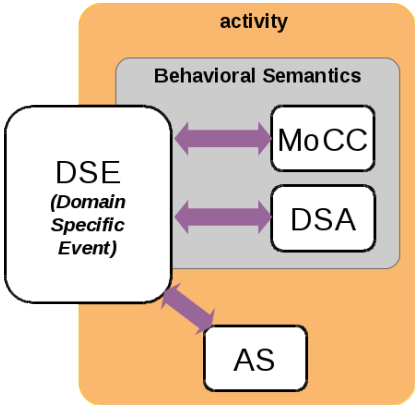


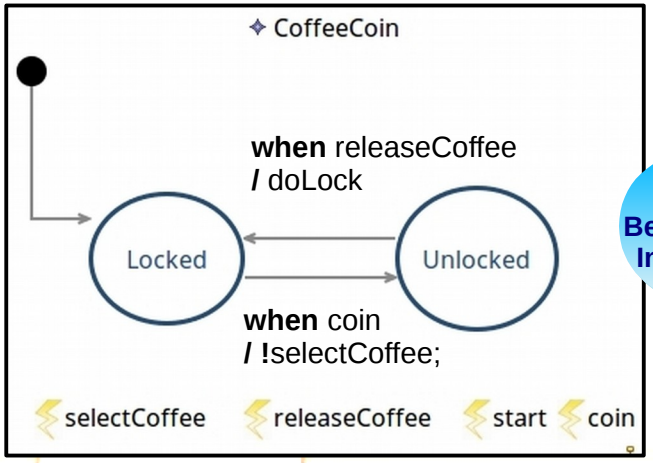
Model Based System Engineering specifies in a model the correspondences between models from the different concerns, all along the product life cycle.


TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS


CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Model Behavioral Interface

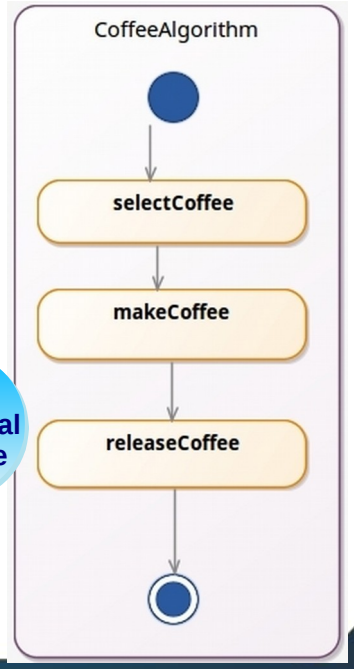


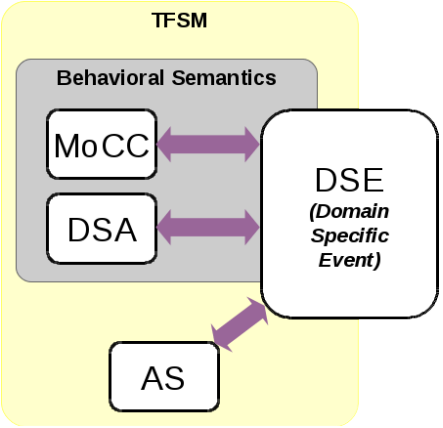


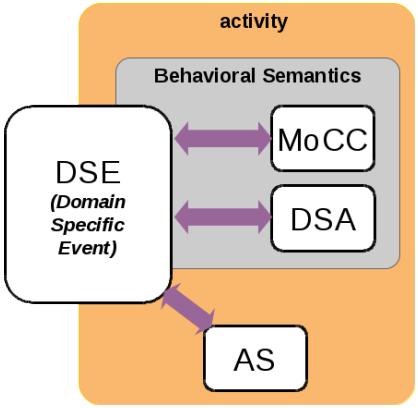


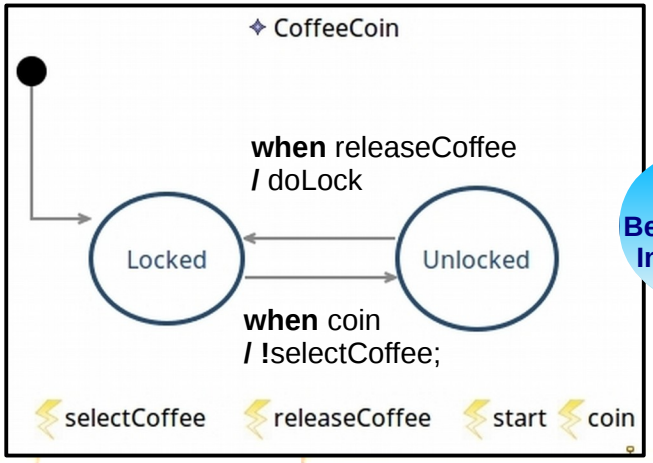
Model Behavioral Interface

Model Behavioral Interface

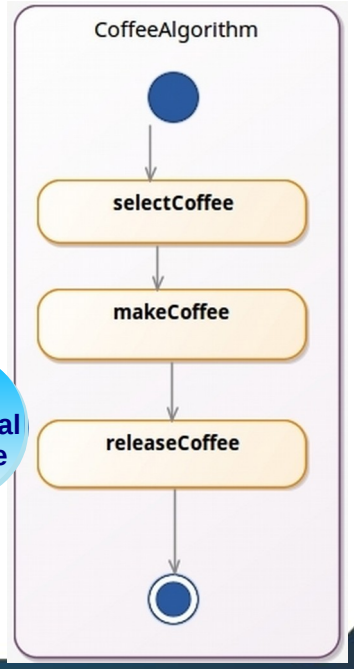


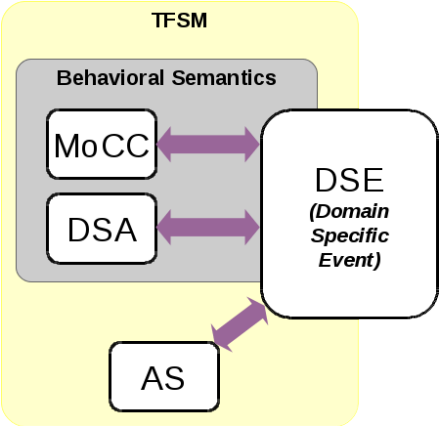

TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS

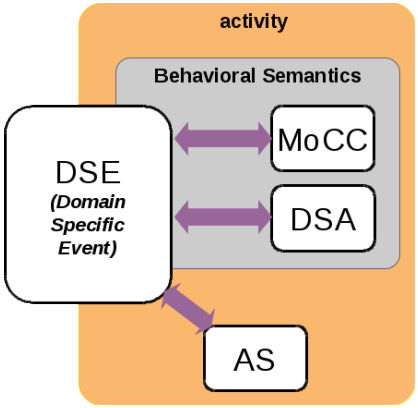

activity
Behavioral Semantics
DSE
(Domain Specific Event)
MoCC
DSA
AS

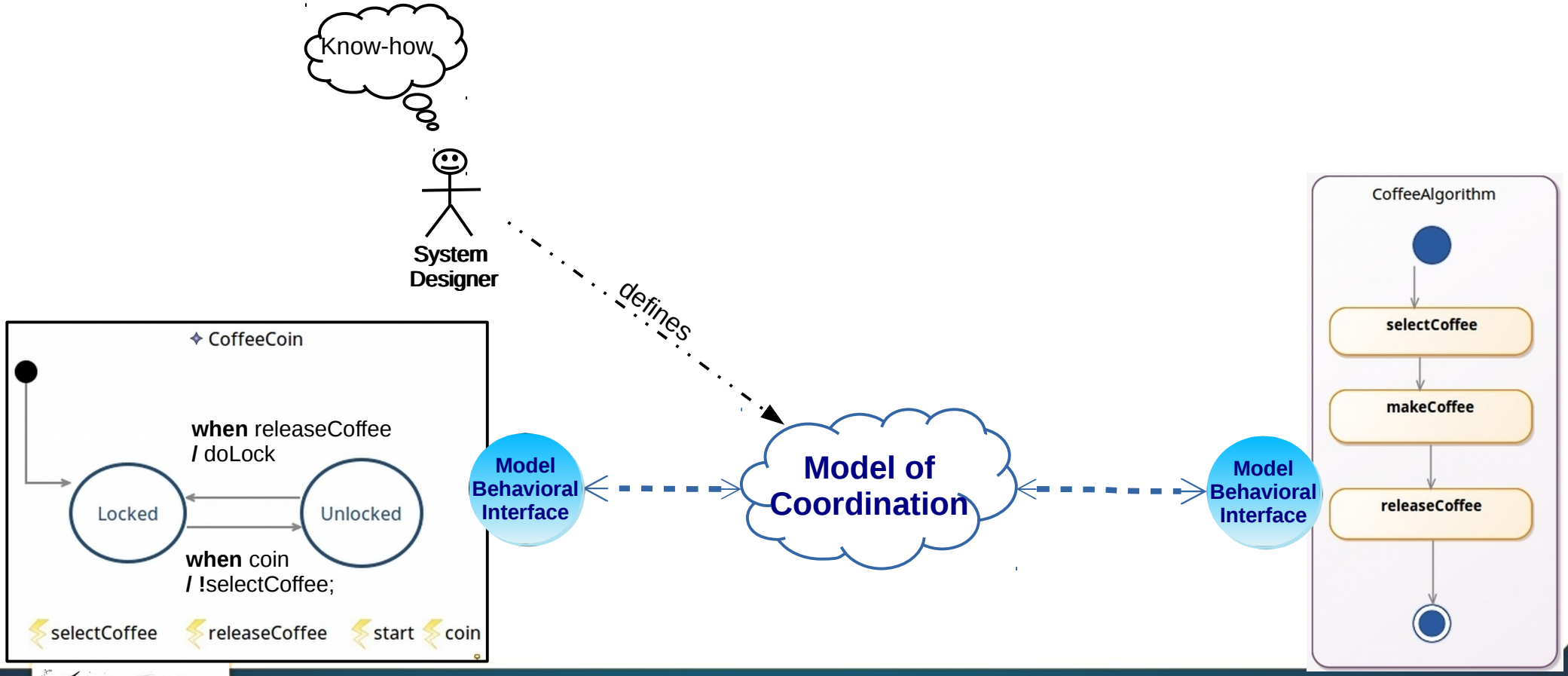

CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin

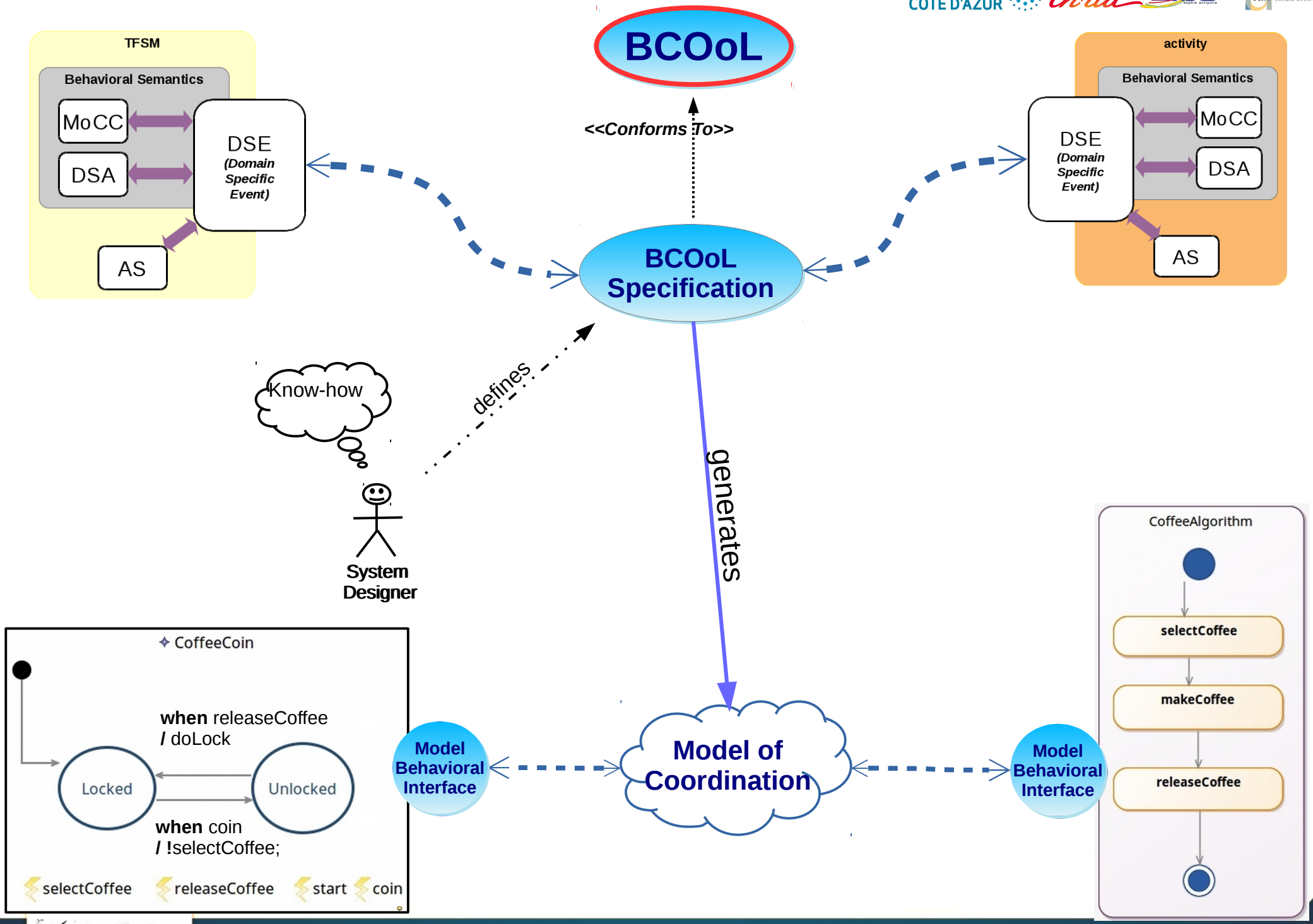

Model Behavioral Interface
Model of Coordination
Model Behavioral Interface

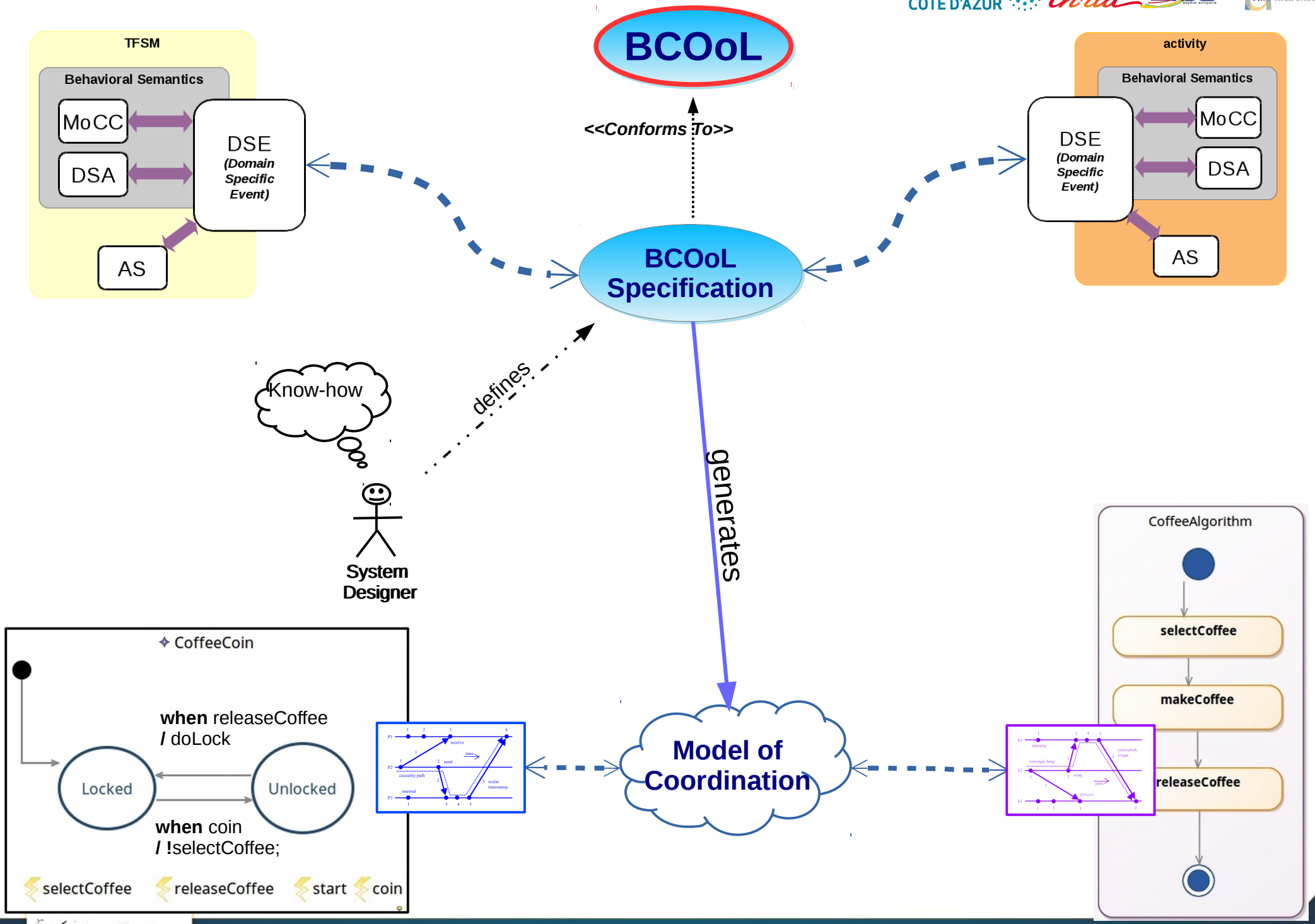

CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee

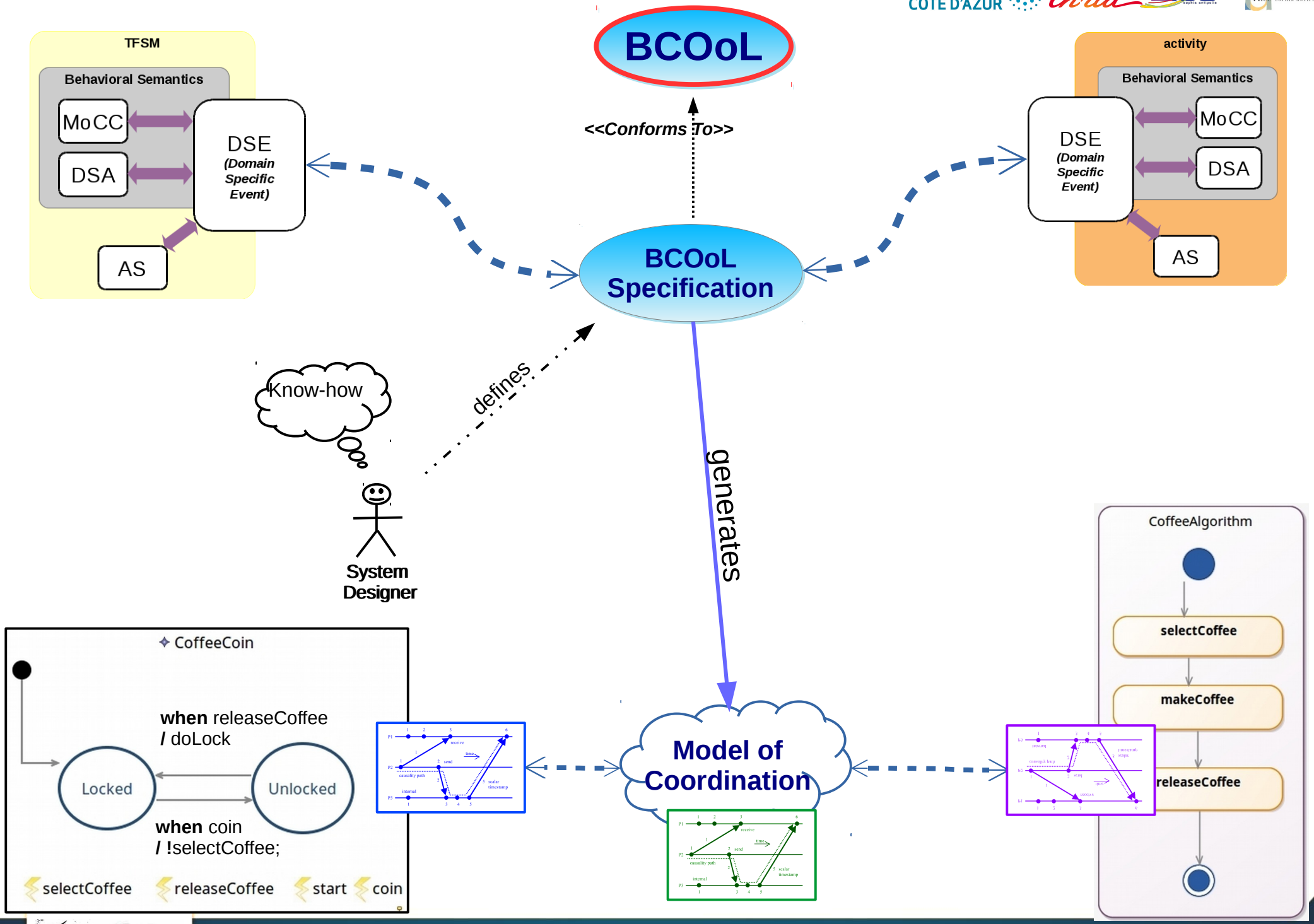

TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS

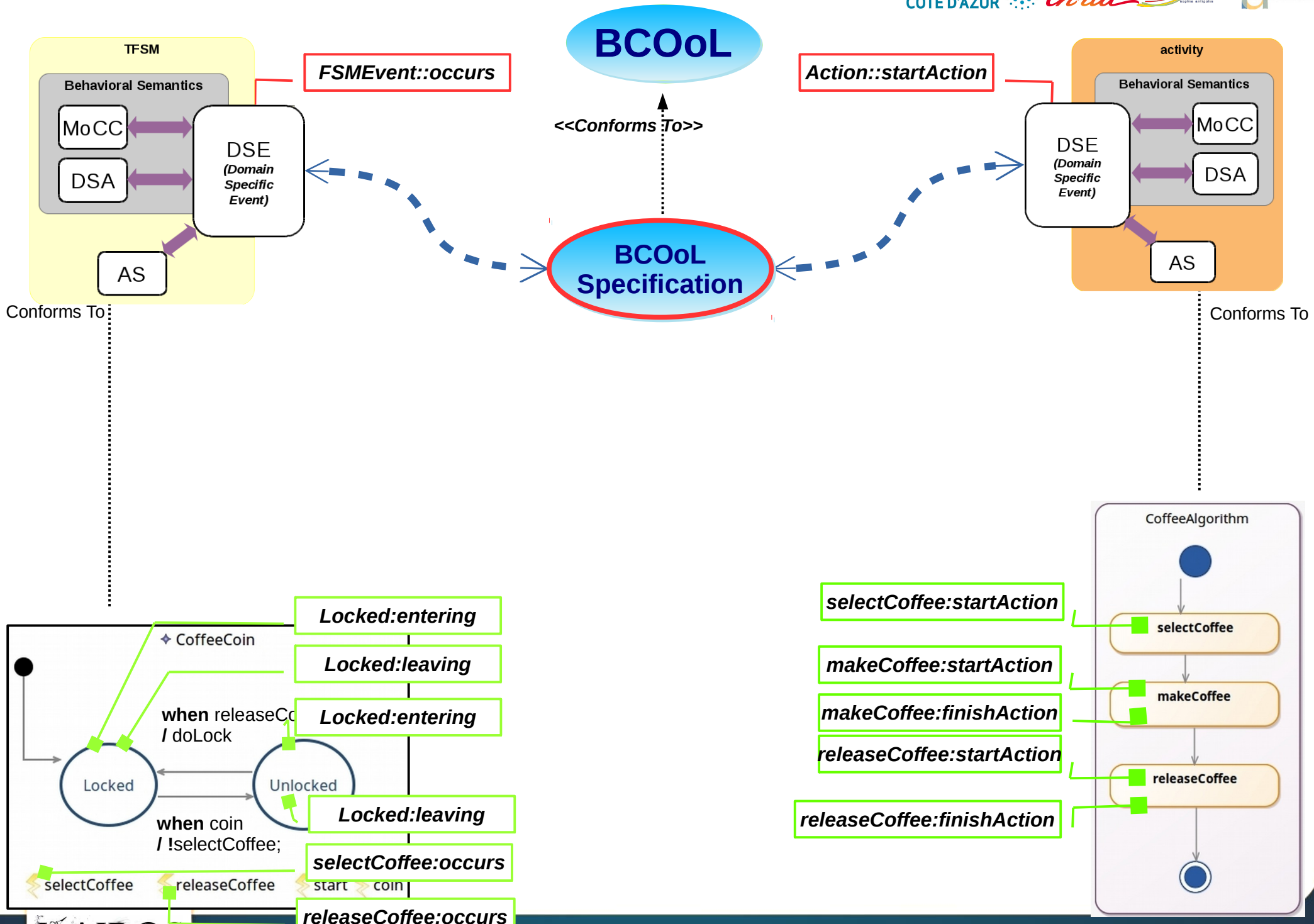

activity
Behavioral Semantics
DSE
(Domain Specific Event)
MoCC
DSA
AS


Know-how
System Designer
defines
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Model Behavioral Interface
Model of Coordination
Model Behavioral Interface
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee

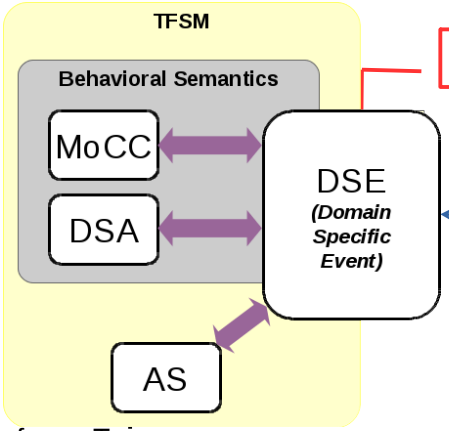

BCOoL
<<Conforms To>>
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
BCOoL Specification
Know-how
defines
System Designer
generates
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Model Behavioral Interface
Model of Coordination
Model Behavioral Interface
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee


BCOoL
<<Conforms To>>
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
BCOoL Specification
Know-how
defines
System Designer
generates
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Model of Coordination
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee


BCOoL
<<Conforms To>>
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
BCOoL Specification
Know-how
defines
System Designer
generates
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Model of Coordination
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee


BCOoL
<<Conforms To>>
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
BCOoL Specification
Conforms To
Conforms To
CoffeeCoin
when releaseCo
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
Locked:entering
Locked:leaving
Locked:entering
Locked:leaving
selectCoffee:occurs
releaseCoffee:occurs
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee
selectCoffee:startAction
makeCoffee:startAction
makeCoffee:finishAction
releaseCoffee:startAction
releaseCoffee:finishAction

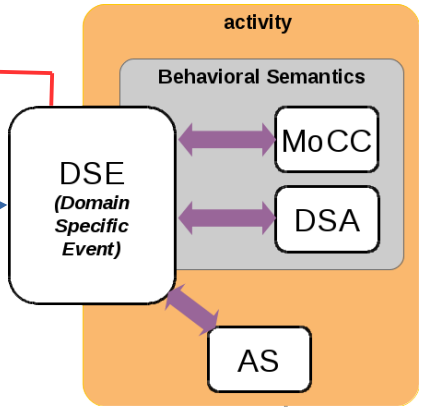
BCOoL



FSMEvent::occurs

<<Conforms To>>

Action::startAction

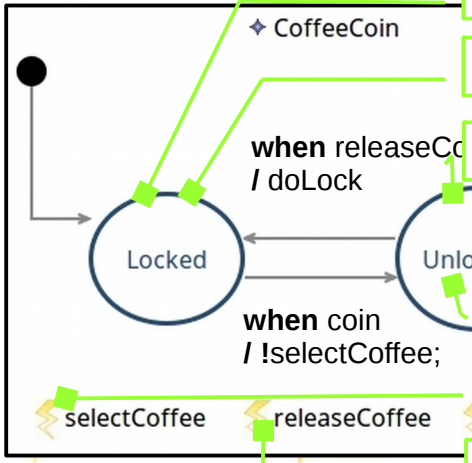


Conforms To

Conforms To

SyncFSMEventsAndActions.bcool

```
ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
  When(occurs.name = startAction.name);
CoordinationRule:
    Rendezvous (occurs, startAction)
End Operator;
```



Locked:entering

Locked:leaving

Locked:entering

Locked:leaving

selectCoffee:occurs

releaseCoffee:occurs

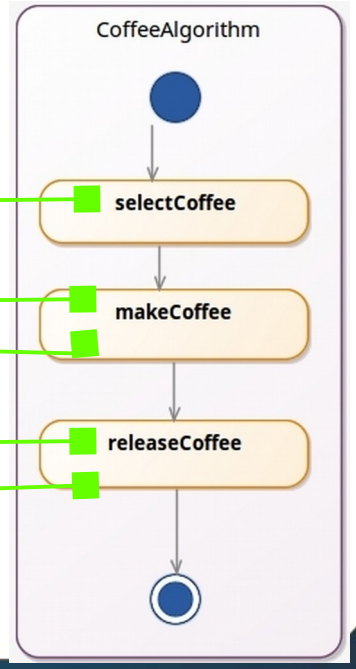
selectCoffee:startAction

makeCoffee:startAction

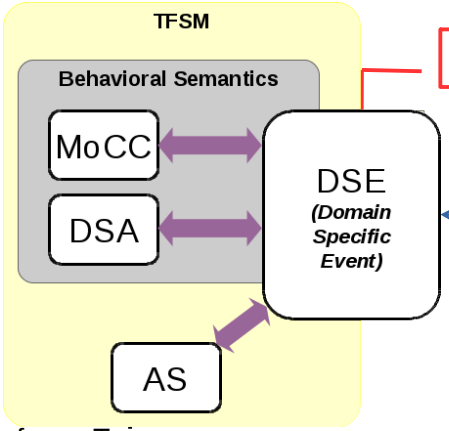
makeCoffee:finishAction

releaseCoffee:startAction

releaseCoffee:finishAction



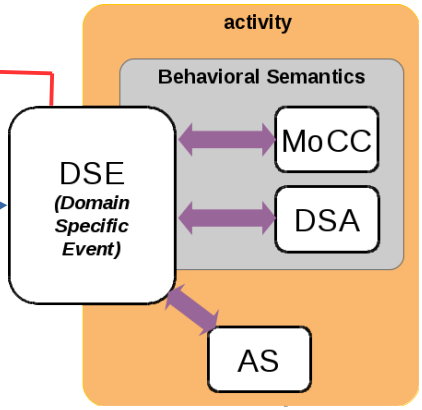
BCOoL



FSMEvent::occurs

<<Conforms To>>

Action::startAction

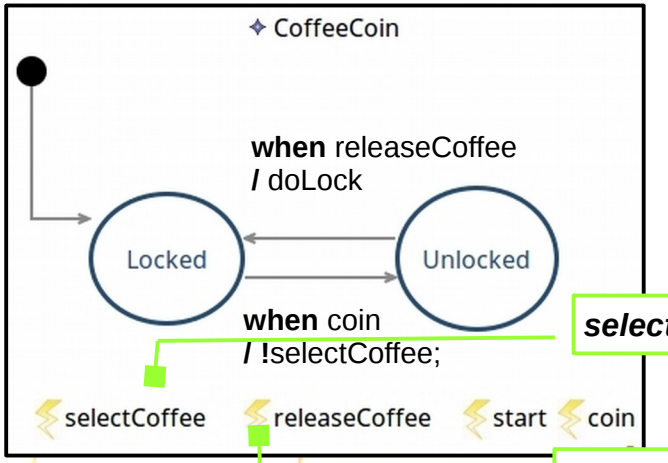


Conforms To

Conforms To

SyncFSMEventsAndActions.bcool

```
ImportInterface tfsm;
ImportInterface Activity;

Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
  When(occurs.name = startAction.name);
CoordinationRule:
    Rendezvous (occurs, startAction)
End Operator;
```



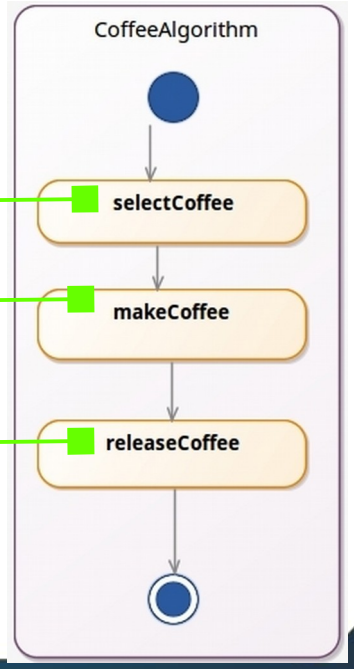
selectCoffee:occurs

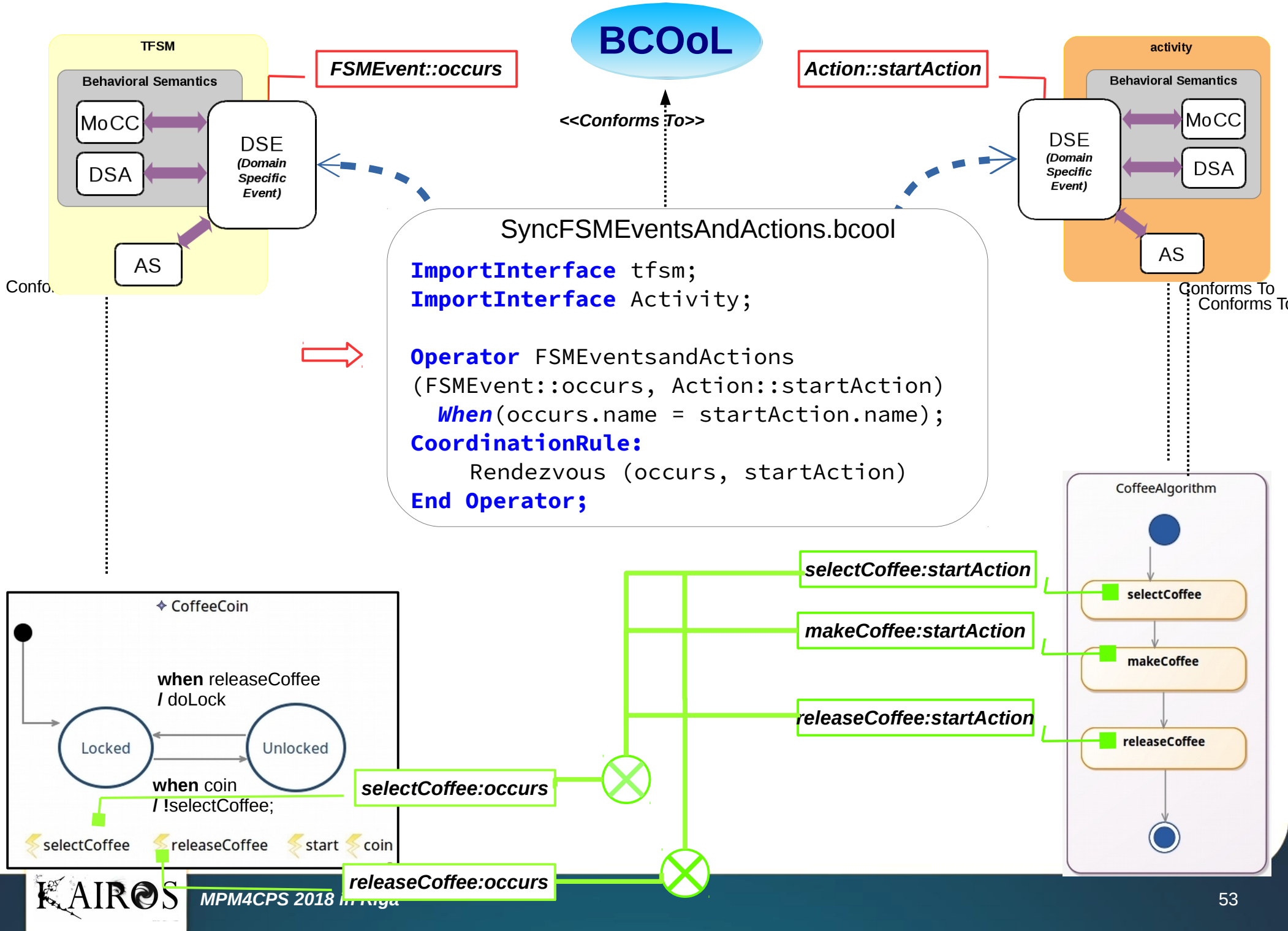
releaseCoffee:occurs

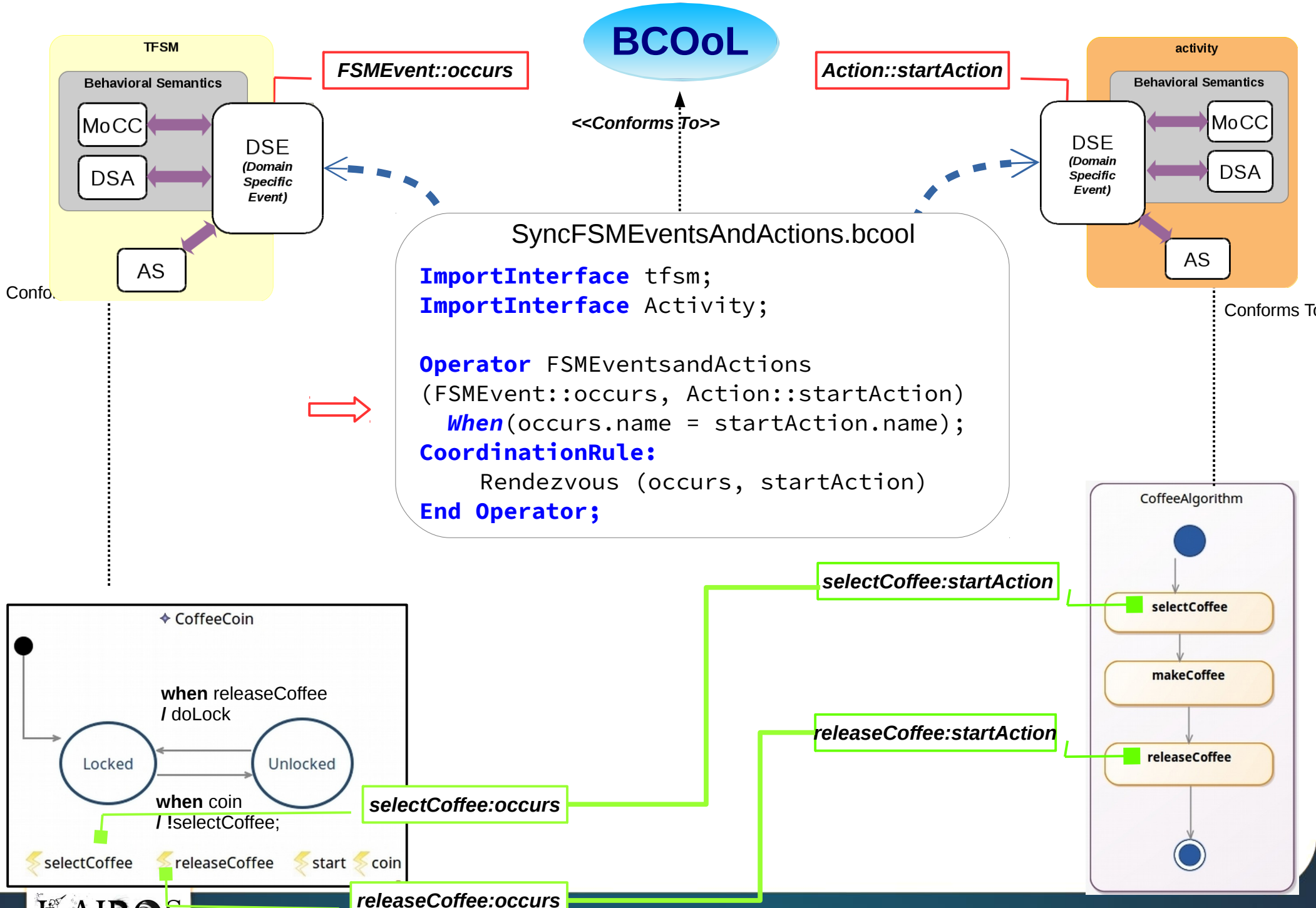
selectCoffee:startAction

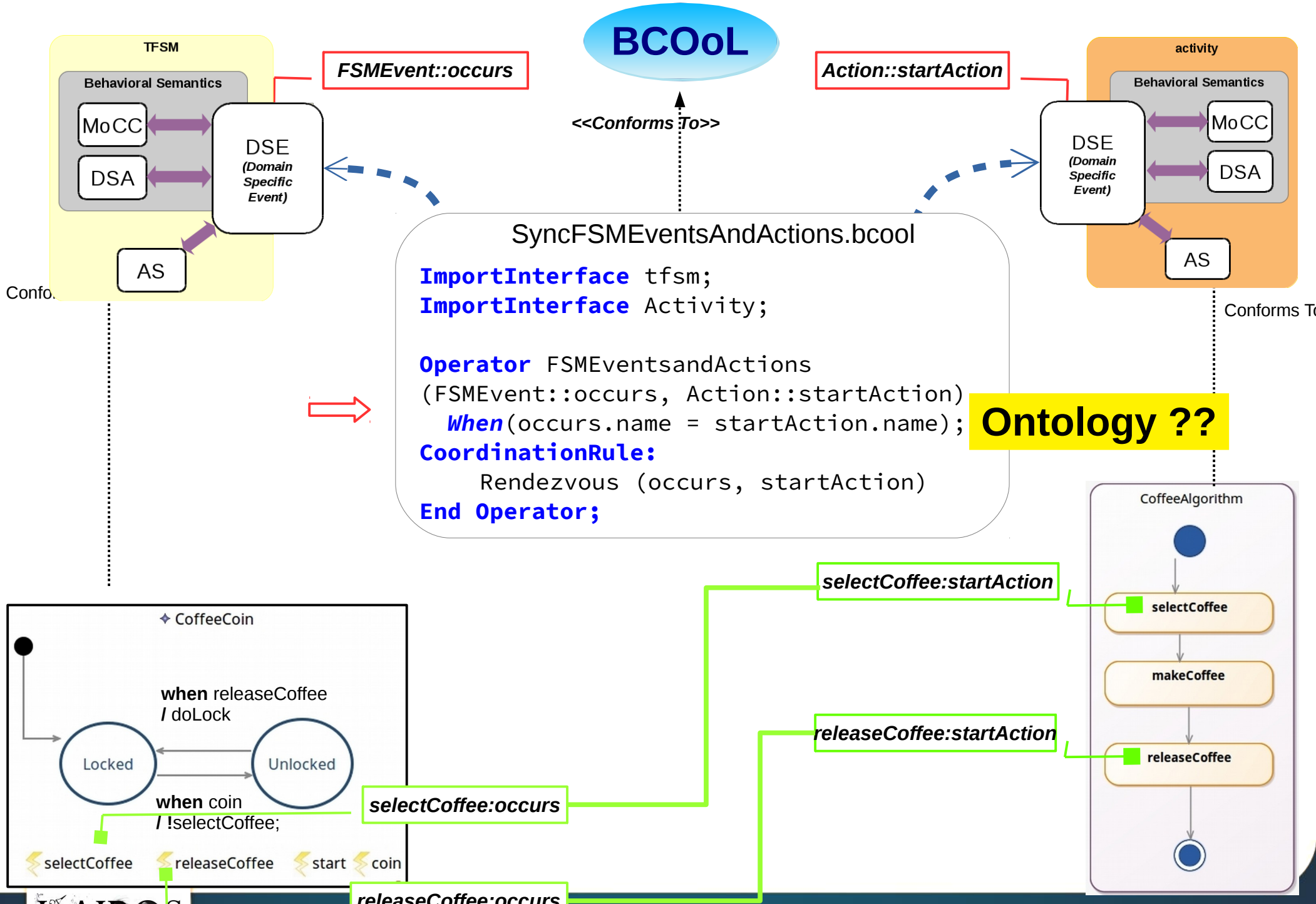
makeCoffee:startAction

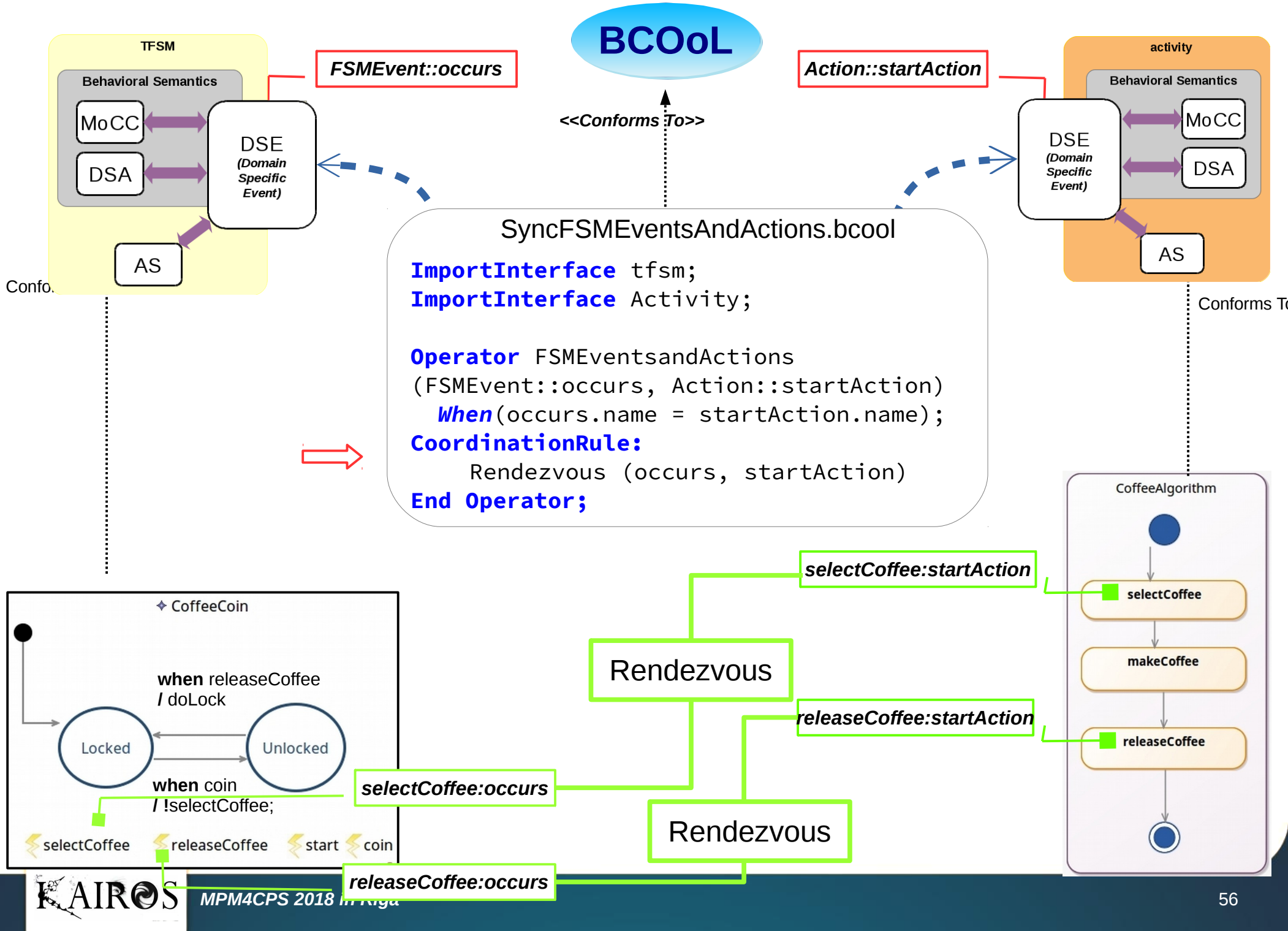
releaseCoffee:startAction

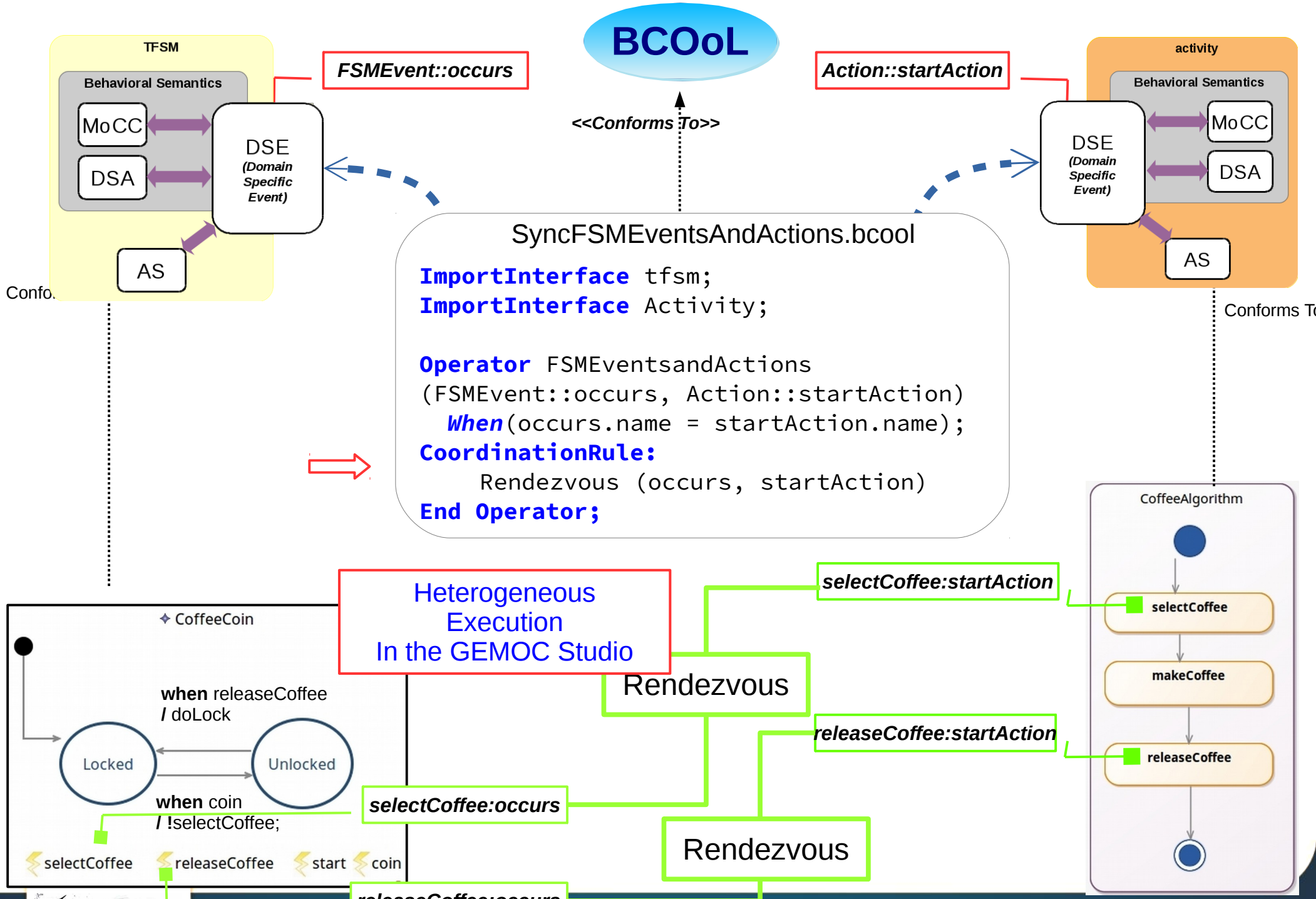



BCOoL
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
<<Conforms To>>
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
Conforms To
Conforms To
SyncFSMEventsAndActions.bcool
ImportInterface tfsm;
ImportInterface Activity;
Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When(occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee
selectCoffee:startAction
makeCoffee:startAction
releaseCoffee:startAction
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
selectCoffee:occurs
releaseCoffee:occurs


BCOoL
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
<<Conforms To>>
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
Conforms To
SyncFSMEventsAndActions.bcool
ImportInterface tfsm;
ImportInterface Activity;
Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When(occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee
selectCoffee:startAction
releaseCoffee:startAction
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
selectCoffee:occurs
releaseCoffee:occurs


BCOoL
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
<<Conforms To>>
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
Conforms To
SyncFSMEventsAndActions.bcool
ImportInterface tfsm;
ImportInterface Activity;
Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When(occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;
Ontology ??
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee
selectCoffee:startAction
releaseCoffee:startAction
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
selectCoffee:occurs
releaseCoffee:occurs


BCOoL
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
<<Conforms To>>
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
Conforms To
SyncFSMEventsAndActions.bcool
ImportInterface tfsm;
ImportInterface Activity;
Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When(occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee
selectCoffee:startAction
Rendezvous
releaseCoffee:startAction
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
selectCoffee:occurs
Rendezvous
releaseCoffee:occurs


BCOoL
TFSM
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
FSMEvent::occurs
<<Conforms To>>
Action::startAction
activity
Behavioral Semantics
MoCC
DSA
DSE
(Domain Specific Event)
AS
Conforms To
SyncFSMEventsAndActions.bcool
ImportInterface tfsm;
ImportInterface Activity;
Operator FSMEventsandActions
(FSMEvent::occurs, Action::startAction)
When(occurs.name = startAction.name);
CoordinationRule:
Rendezvous (occurs, startAction)
End Operator;
Heterogeneous Execution In the GEMOC Studio
CoffeeCoin
when releaseCoffee
/ doLock
Locked
Unlocked
when coin
/ !selectCoffee;
selectCoffee
releaseCoffee
start
coin
selectCoffee:startAction
Rendezvous
releaseCoffee:startAction
selectCoffee:occurs
Rendezvous
releaseCoffee:occurs
CoffeeAlgorithm
selectCoffee
makeCoffee
releaseCoffee

Implemented into the GEMOC studio

syncproducttfsmwithfuml.bcool

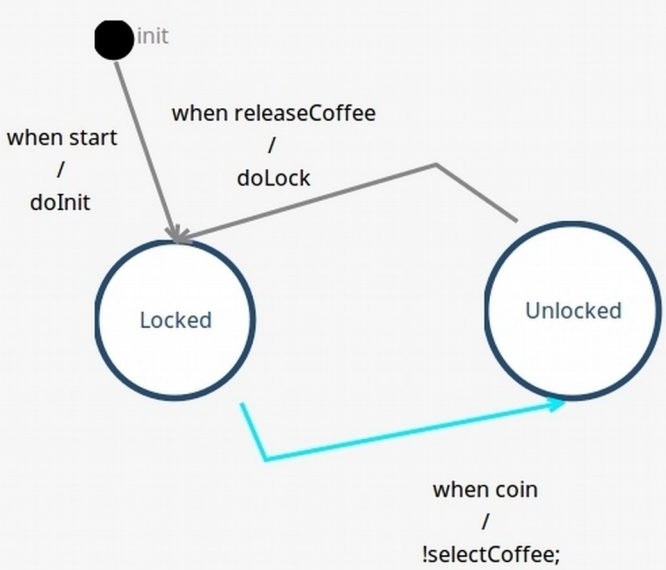
```
SyncProductTfsmwithfUML

ImportLib "platform:/resource/org.gemoc.bcool.example.productfumlandtfsm/operator/facilities.bcoollib"
ImportLib "platform:/resource/org.gemoc.bcool.example.productfumlandtfsm/operator/bcoollib.ccslLib"

ImportInterface "platform:/plugin/org.modelexecution.operationalsemantics.gemoc.ecl/model/ActivitiyDiagramV2.ecl" as ad
ImportInterface "platform:/plugin/org.gemoc.sample.tfsm.eclmoc2as/ecl/TFSM.ecl" as tfsm

Spec test

Operator  MatchingandCoordinationSharedEvents (dse1 : tfsm::occurs, dse2 : ad::executeIt)
MatchingCorrespondance: when  dse1.name = dse2.name ;
       CoordinationRule:
          facilities.RendezVous(dse1, dse2)
end operator;
```

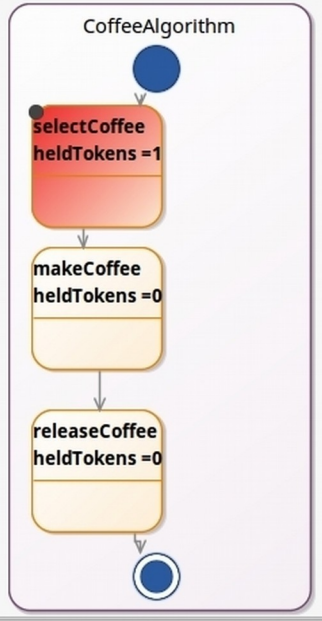
Editing facilities by using Xtext

Concurrent Logical Steps Decider

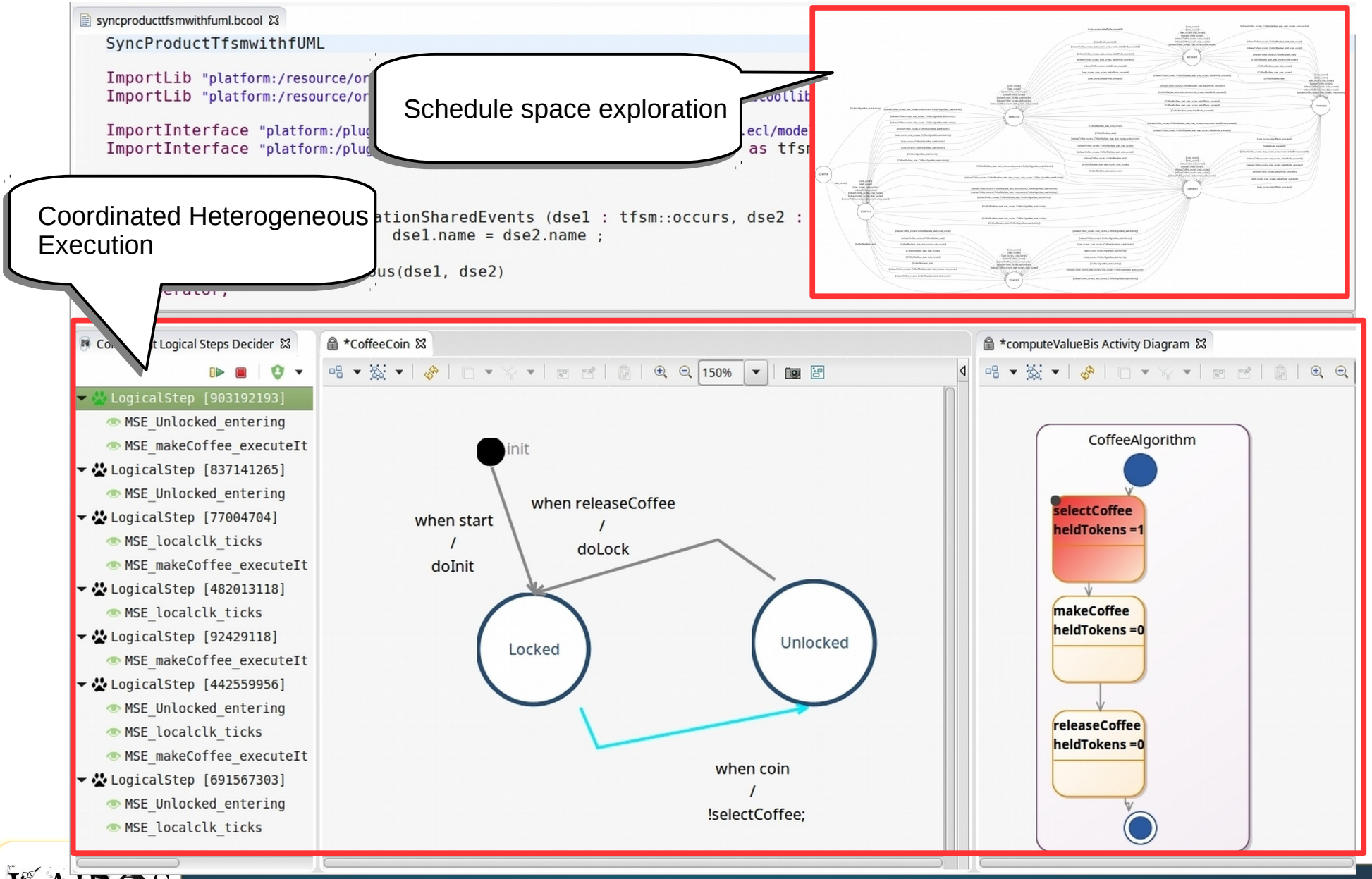
- LogicalStep [903192193]
  - MSE_Unlocked_entering
  - MSE_makeCoffee_executeIt
- LogicalStep [837141265]
  - MSE_Unlocked_entering
- LogicalStep [77004704]
  - MSE_localclk_ticks
  - MSE_makeCoffee_executeIt
- LogicalStep [482013118]
  - MSE_localclk_ticks
- LogicalStep [92429118]
  - MSE_makeCoffee_executeIt
- LogicalStep [442559956]
  - MSE_Unlocked_entering
  - MSE_localclk_ticks
  - MSE_makeCoffee_executeIt
- LogicalStep [691567303]
  - MSE_Unlocked_entering
  - MSE_localclk_ticks

*CoffeeCoin

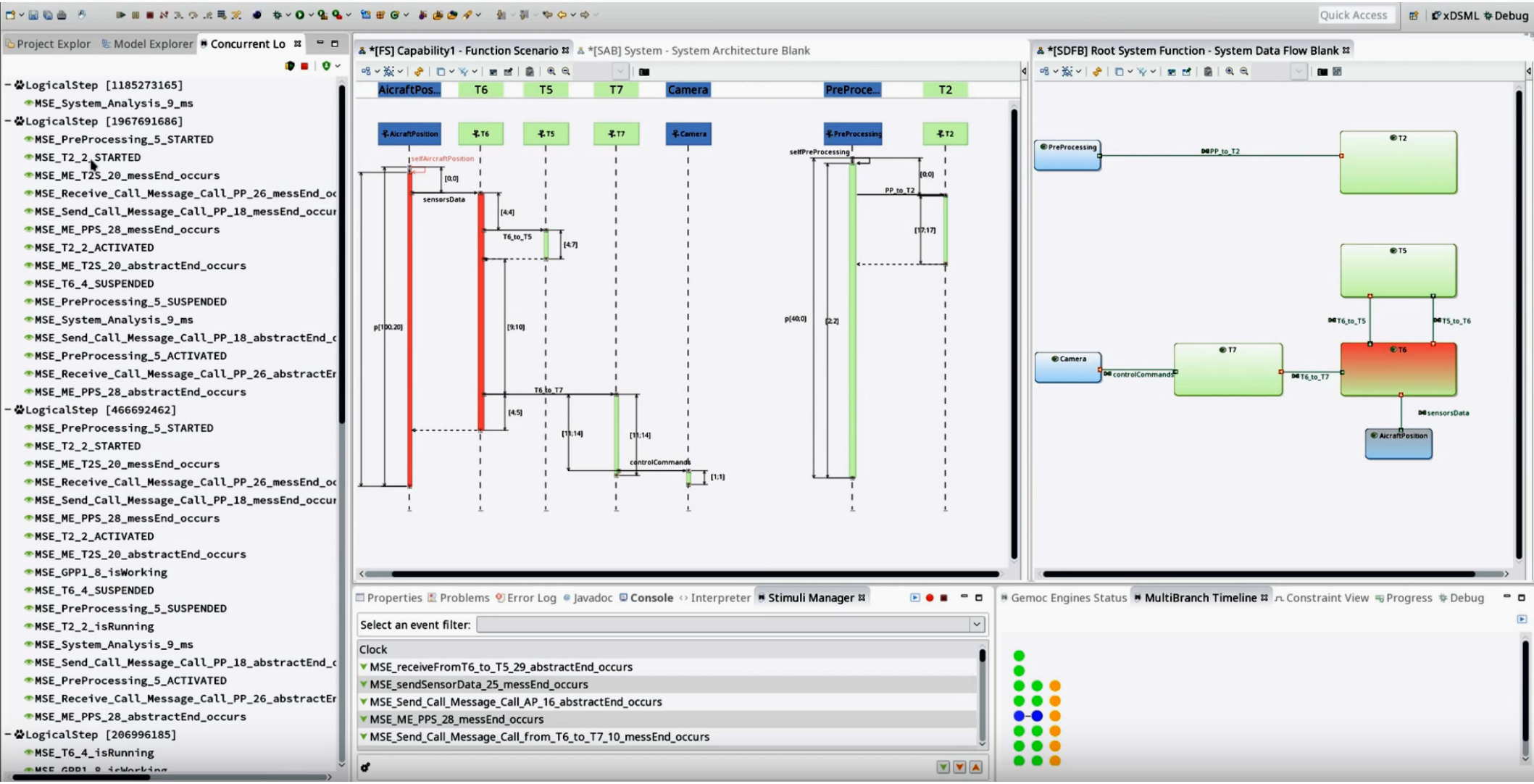




Implemented into the GEMOC studio

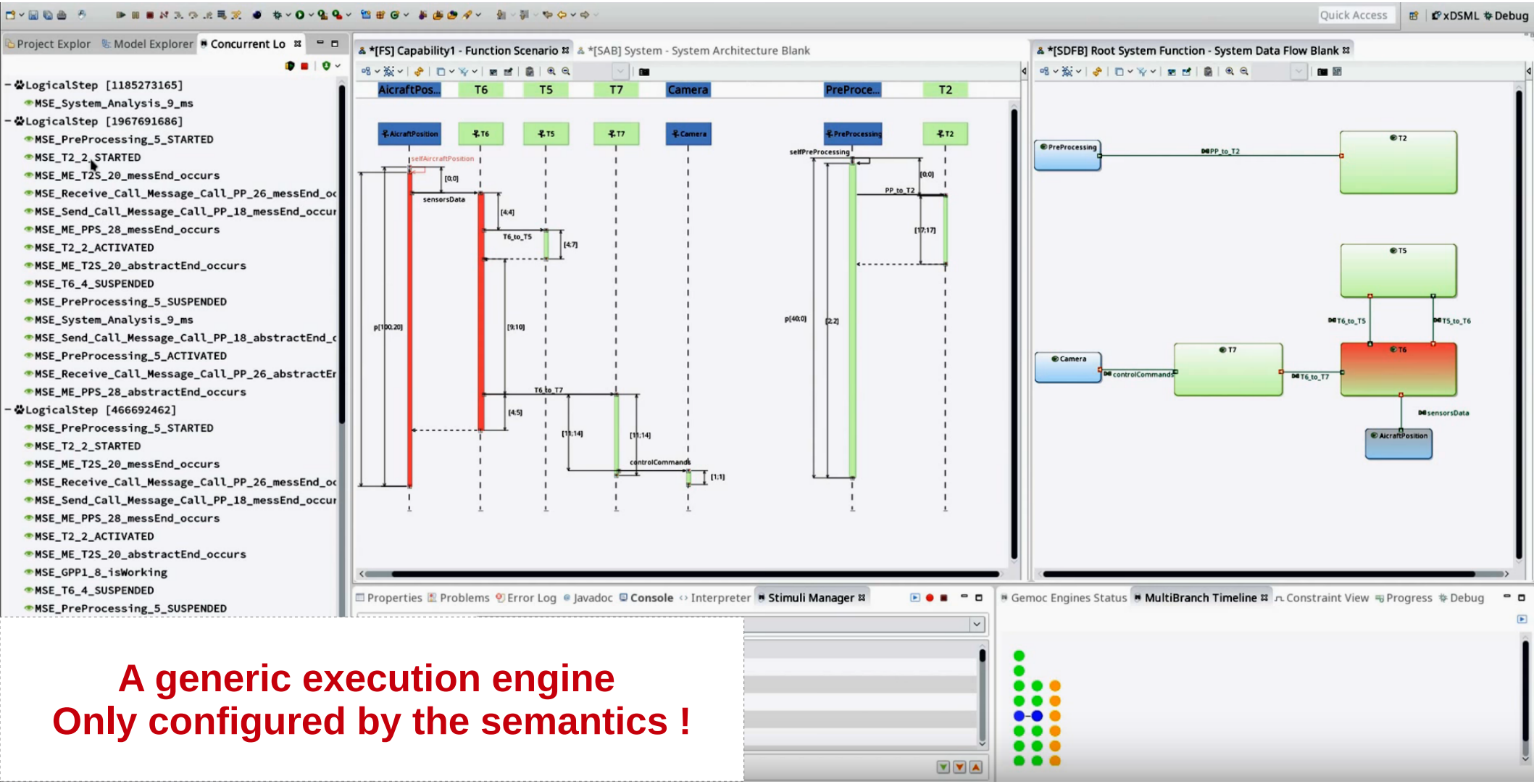


Modeling workbench: http://gemoc.org/studio/



Making the Capella language executable….

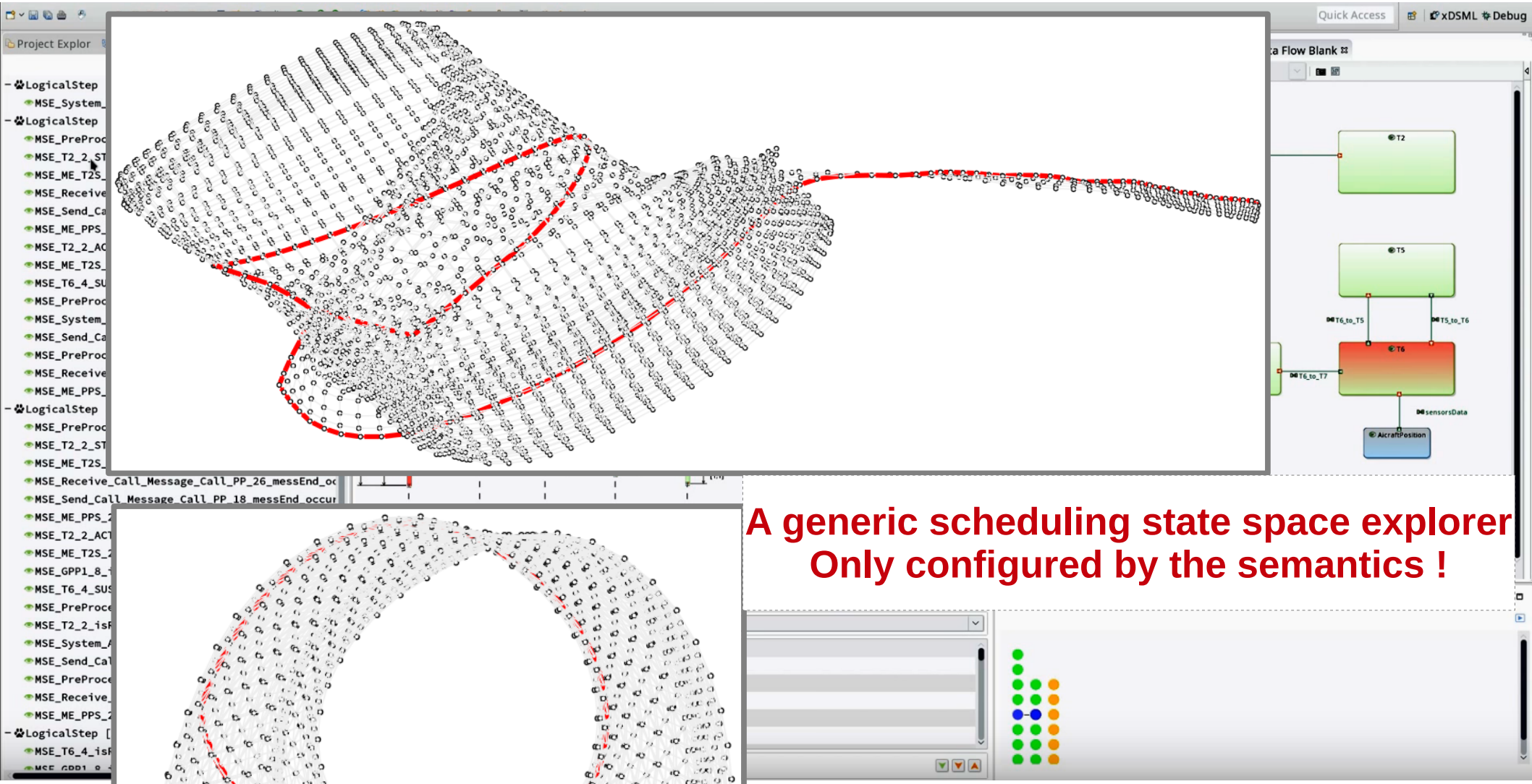
Modeling workbench: http://gemoc.org/studio/



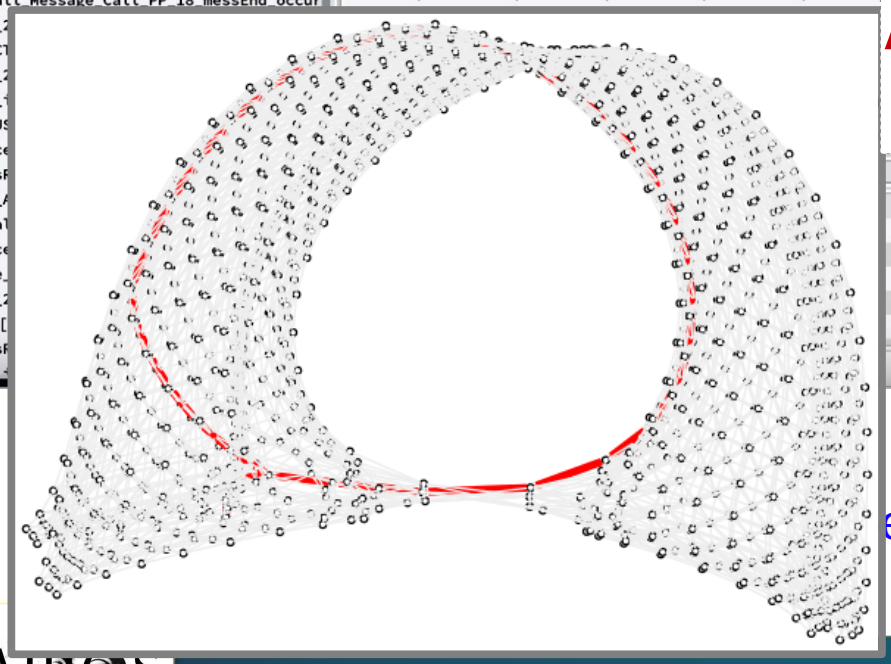
A generic execution engine
Only configured by the semantics !

Making the Capella language executable….

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



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



ella language executable….

Tested on several languages

- ArduinoML
- UML
  - Sequence Diagram
  - State Charts
  - Activity diagram (challenge winner)
- Capella
  - Temporal scenario, functional architecture and deployement
  - Mode automata and functional architecture

- Deployed temporal and modal MSC
- MuVArch
- SMcube
- Communicating FSM
- TimedFSM
- Inra DSL for cow management
- ComponentModel and TFSM
- HW language for bus analysis
- MarkedGraph
- SDF (+deployment)
- AADL
- Communicating real time tasks
- ArduinoML and Sequence Diagram
- EAST-ADL

Video:
- Simulation and trace checking on the Capella system engineering language: https://www.youtube.com/watch?v=ESIX2PFGiDU
- Simulation of Multi communication Arduino boards: https://youtu.be/dtJZyK1RM2A
- BCOoL: https://youtu.be/-o6DAzlglMw

Conclusion

http://gemoc.org

- We provided meta languages for:
  - The modeling of executable languages
  - The modeling of coordination patterns
- We tooled the meta languages so that;
  - A modeling workbench with debugging facilities is generated from the language specification
  - An heterogeneous execution engine is generated from a coordination pattern and specific models
  - Generic analysis is possible
- We are currently:
  - Applying such techniques to study interaction in model-based system engineering solutions (with Thales and Safran)
  - Extending the approach to support continuous time models
  - Studying how this co modeling can be used to generate co simulation masters (based on FMI standard)
  - Transferring the technology in a company that uses Connected Objects and IoT standard to monitor senior daily live activities environment in a non intrusive way.

Conclusion

http://gemoc.org

- We provided meta languages for:
  - The modeling of executable languages
  - The modeling of coordination patterns
- We tooled the meta languages so that;
  - A modeling workbench with debugging facilities is generated from the language specification
  - An heterogeneous execution engine is generated from a coordination pattern and specific models
  - Generic analysis is possible
- We are currently:
  - Applying such techniques to study interaction in model-based system engineering solutions (with Thales and Safran)
  - Extending the approach to support continuous time models
  - Studying how this co modeling can be used to generate co simulation masters (based on FMI standard)
  - Transferring the technology in a company that uses Connected Objects and IoT standard to monitor senior daily live activities environment in a non intrusive way.

Conclusion

http://gemoc.org

- We provided meta languages for :
  - The modeling of executable languages
  - The modeling of coordination patterns
- We tooled the meta languages so that :
  - A modeling workbench with debugging facilities is generated from the language specification
  - An heterogeneous execution engine is generated from a coordination pattern and specific models
  - Generic analysis is possible
- We are currently:
  - Applying such techniques to study interaction in model-based system engineering solutions (with Thales and Safran)
  - Extending the approach to support continuous time models
  - Studying how this co modeling can be used to generate co simulation masters (based on FMI standard)
  - Transferring the technology in a company that uses Connected Objects and IoT standard to monitor senior daily live activities in a non intrusive way.