MODEL EXECUTION

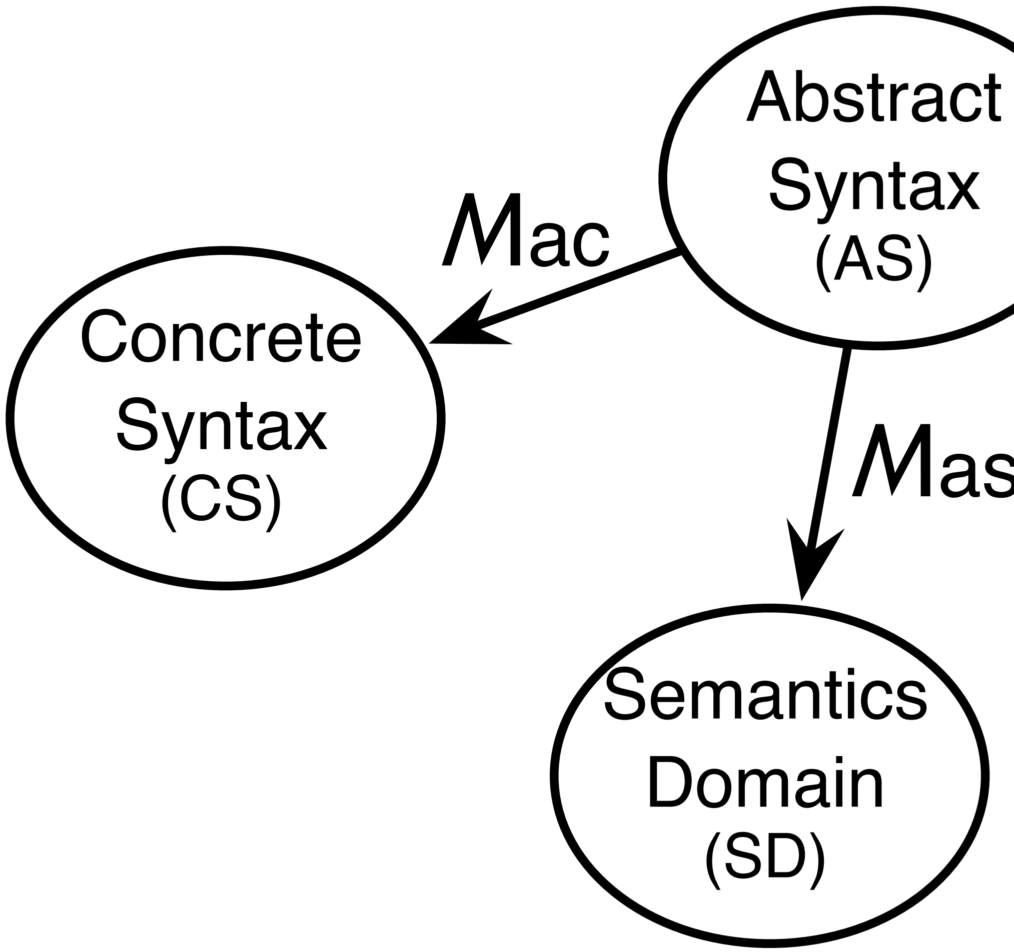
BUILD YOUR OWN COMPILER AND VIRTUAL MACHINE

ESIR3 ASE, 2022-2023

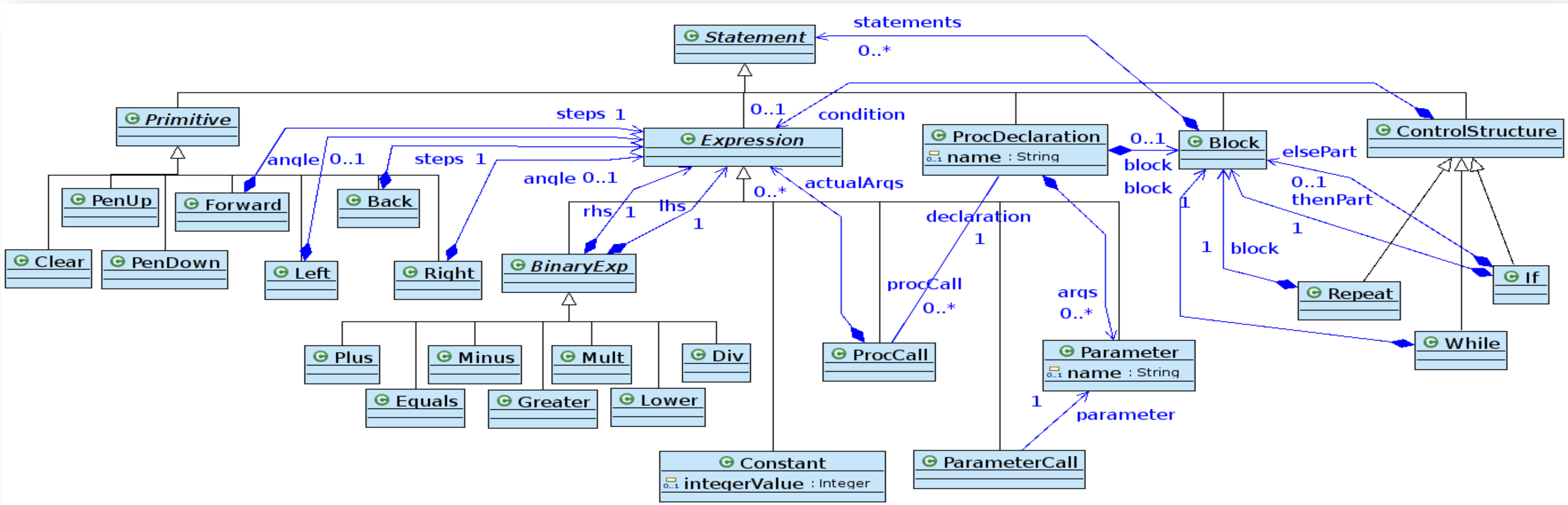
BENOIT COMBEMALE
PROFESSOR, UNIV. RENNES 1 & INRIA, FRANCE

HTTP://COMBEMALE.FR
BENOIT.COMBEMALE@IRISA.FR
@BCOMBEMALE

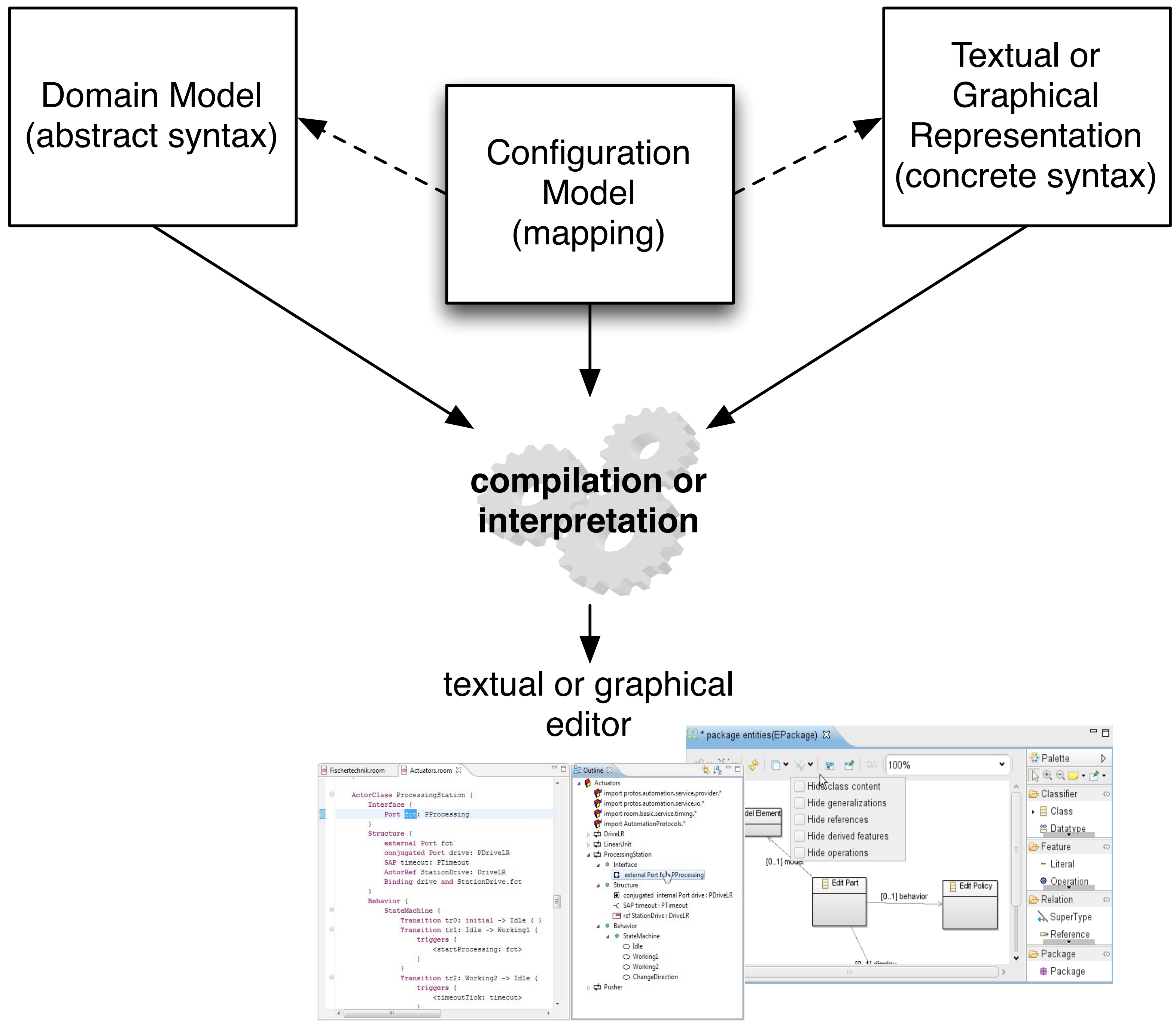
Reminder about what is a language



Reminder about what is an abstract syntax



Reminder about what is a concrete syntax



Reminder about what is a semantics

- Any "meaning" given to the domain model
  → compiler, interpreter, analysis tool, refactoring tool, etc.
- Thanks to model transformations

  program = data + algorithms ☺

- In practices?
  - It requires to "traverse" the domain model, and… do something!
  - Various languages, and underlying paradigms:
    - *Declarative* (rule-based): mostly for pattern matching (*e.g.*, analysis, refactoring)
    - *Imperative* (visitor-based):
      - *interpreter pattern*: mostly for model interpretation (*e.g.*, execution, simulation)
      - *template*: mostly for text generation (*e.g.*, code/test/doc generators)

Reminder of the previous lectures / labs

Build your own (Domain-Specific) Language

1. Build your abstract syntax as a domain model with Ecore (possibly additional constraints with OCL, aka. context conditions)
2. Build your concrete syntax (textual with Xtext, *graphical with Sirius*)
3. Build your generators
   - Documentation generator
   - Code generator (/compiler)

Objectives of the coming lecture/labs

4. **Build your interpreter (/ VM)**
5. Build your animator

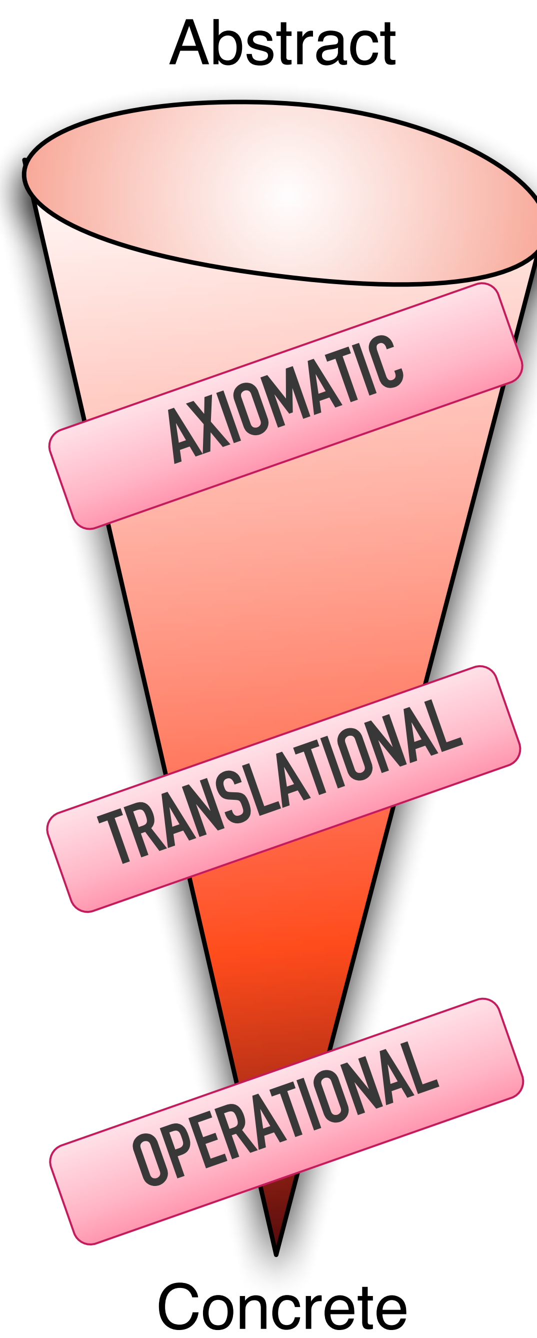
Get your own modeling workbench with

model edition, compilation, execution, simulation, (graphical) animation and debugging

Definition of the Behavioral Semantics of DSL

```
int x;
void decr () {
        if ( x>0 )
        x = x-1;
}
```

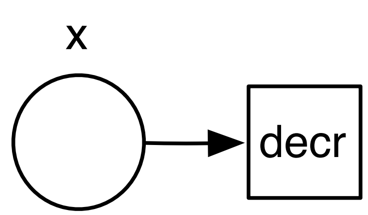
System
x : Int
decr()



- Axiomatic

```
context System::decr() post :
    self .x =   if ( self .x@pre>0 )
                    then self.x@pre - 1
                    else self.x@pre
                endif
```

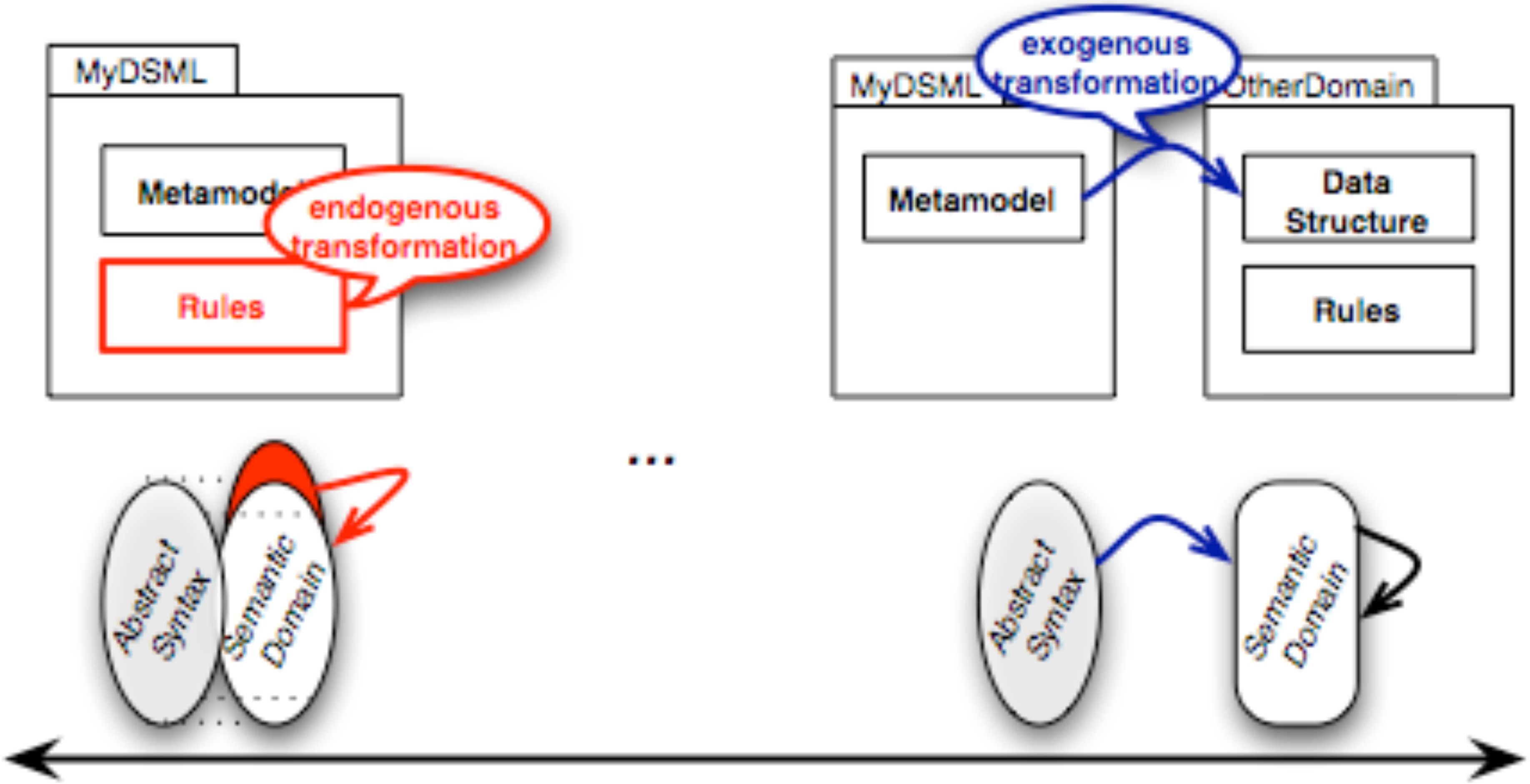
- Denotational/translational



- Operational

```
operation decr () is do
    if x>0 then x = x - 1 end
```

Definition of the Behavioral Semantics of DSL

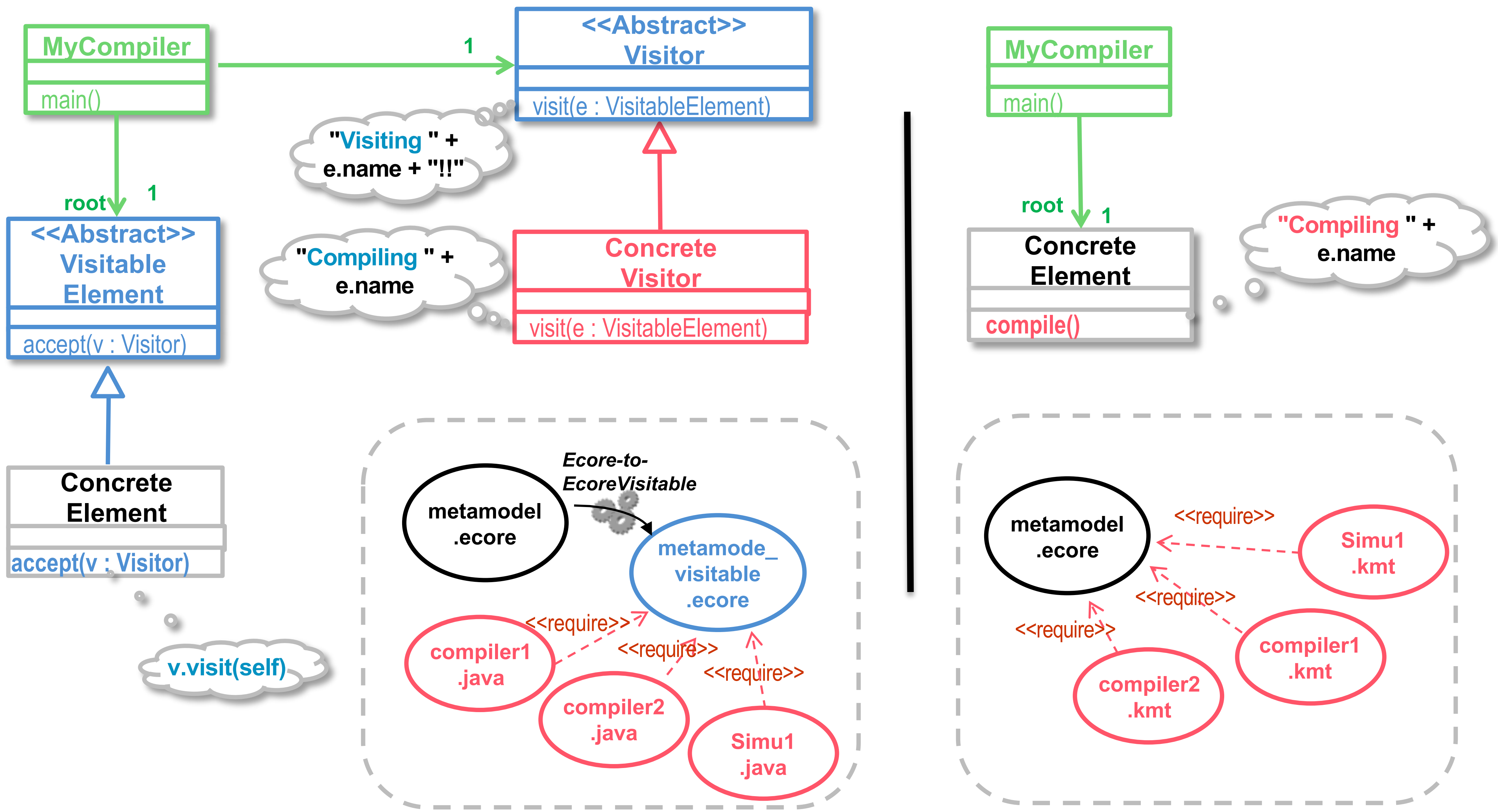


Definition of the model interpretation thanks to an action language (i.e., operational semantics)

Definition of the model compilation thanks to a transformation language (i.e., denotational semantics)

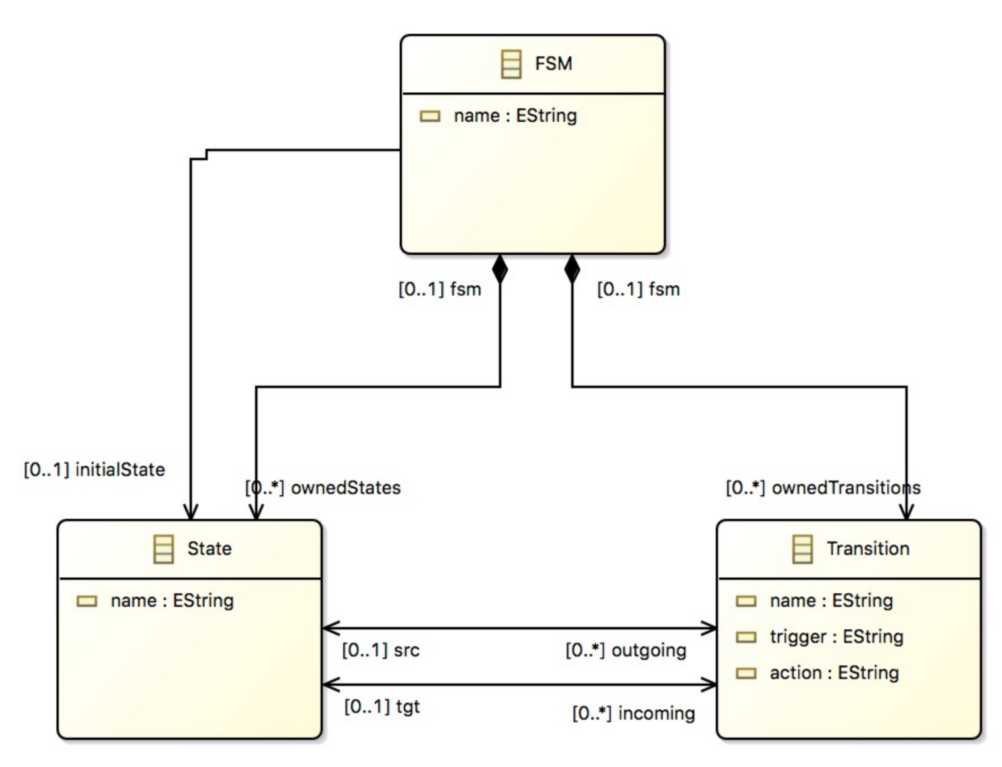
Implement your own compiler / interpreter

- Visitor-based?
  - Interpreter/visitor patterns, static introduction (aka. open class)



Implement your own interpreter with Xtend/K3

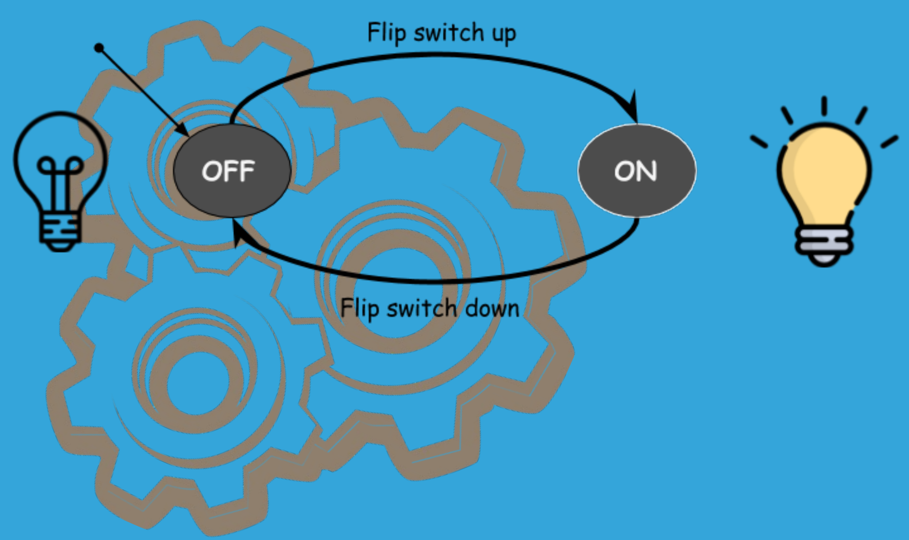
```
@Aspect(className=State)
class StateAspect {
    @Step
    def public void step(String inputString) {
            // Get the valid transitions
            val validTransitions =  _self.outgoing.filter[t | inputString.compareTo(t.trigger) == 0]

            if(validTransitions.empty) {
                    //just copy the token to the output buffer
                    _self.fsm.outputBuffer.enqueue(inputString)
            }

            if(validTransitions.size > 1) {
                    throw new Exception("Non Determinism")
            }

            // Fire transition first transition (could be random%VT.size)
            if(validTransitions.size > 0){
                    validTransitions.get(0).fire
                    return
            }
            return

    }
}
```



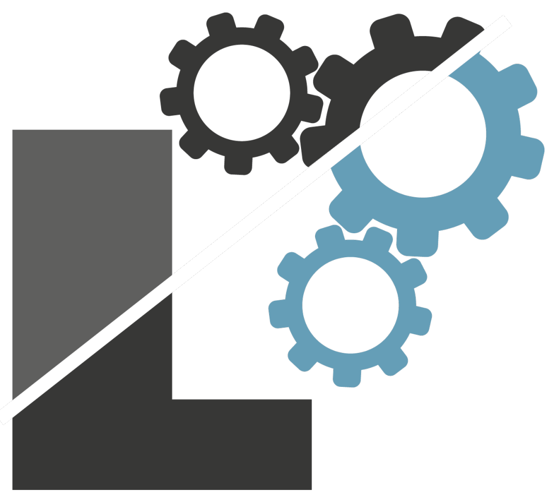

```
@Aspect(className=Transition)
class TransitionAspect {
    @Step
    def public void fire() {
            println("Firing " + _self.name + " and entering " + _self.tgt.name)
            val fsm = _self.src.fsm
            fsm.currentState = _self.tgt
            fsm.outputBuffer.enqueue(_self.action)
            fsm.consummedString = fsm.consummedString + fsm.underProcessTrigger
    }
}
```

```
@Aspect(className=FSM)
class FSMAspect {

    public State currentState

}
```

Part 4: define an interpreter for your language



The GEMOC Studio



Language Workbench

Design and integrate your executable DSMLs

http://gemoc.org/studio

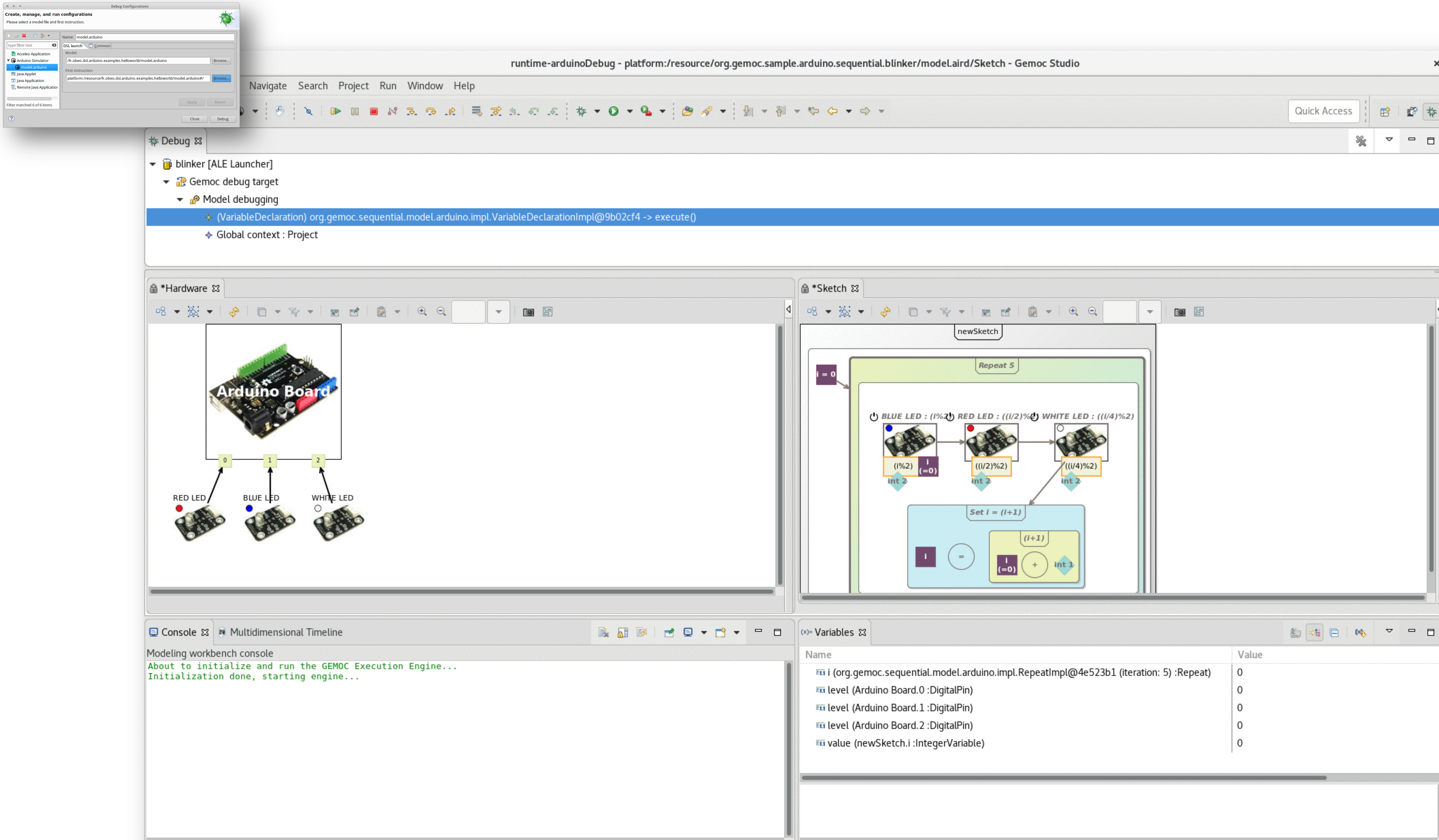
and

http://eclipse.org/gemoc

Modeling Workbench

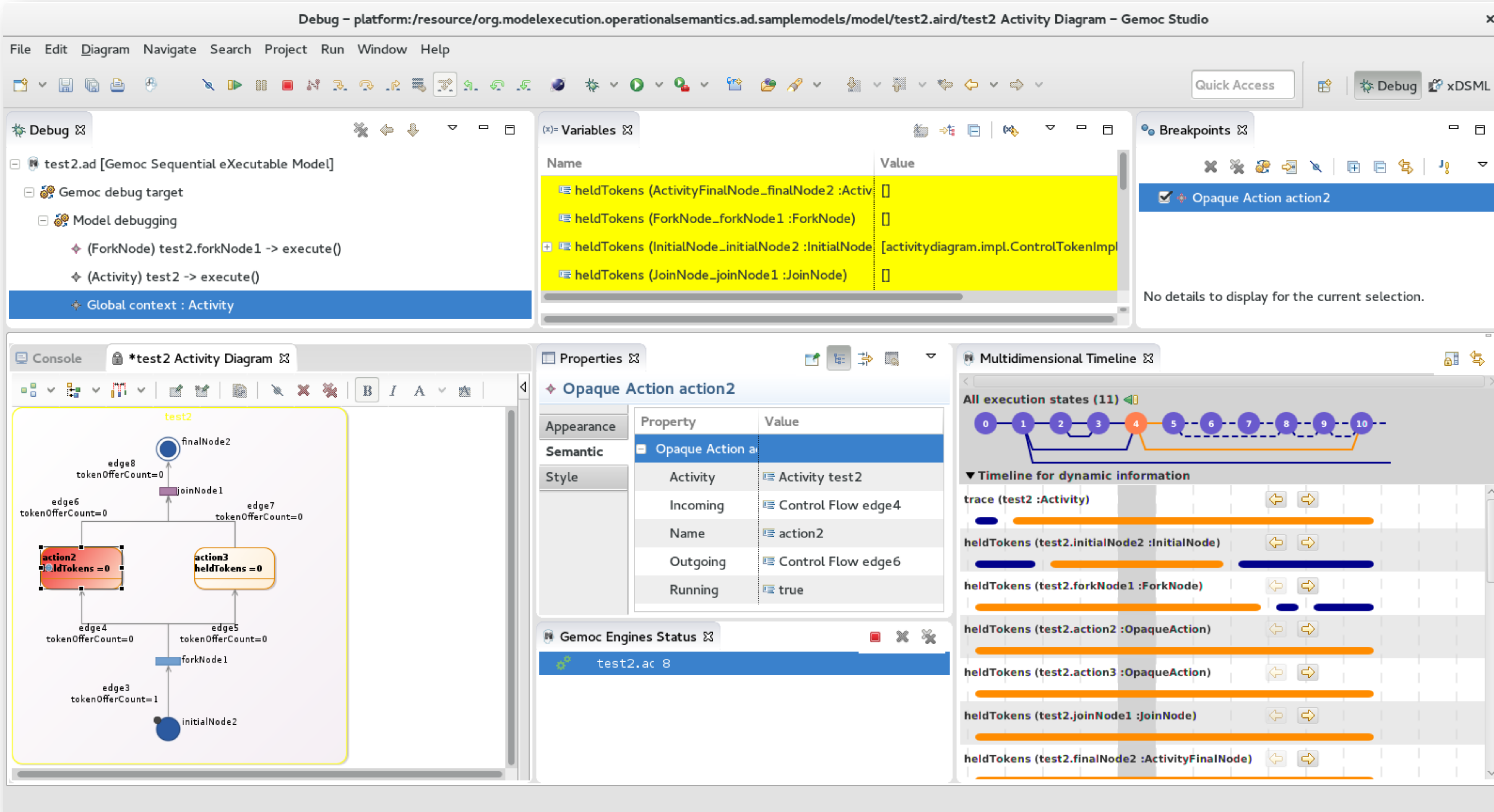
Edit, simulate and animate your heterogeneous models

Arduino Designer



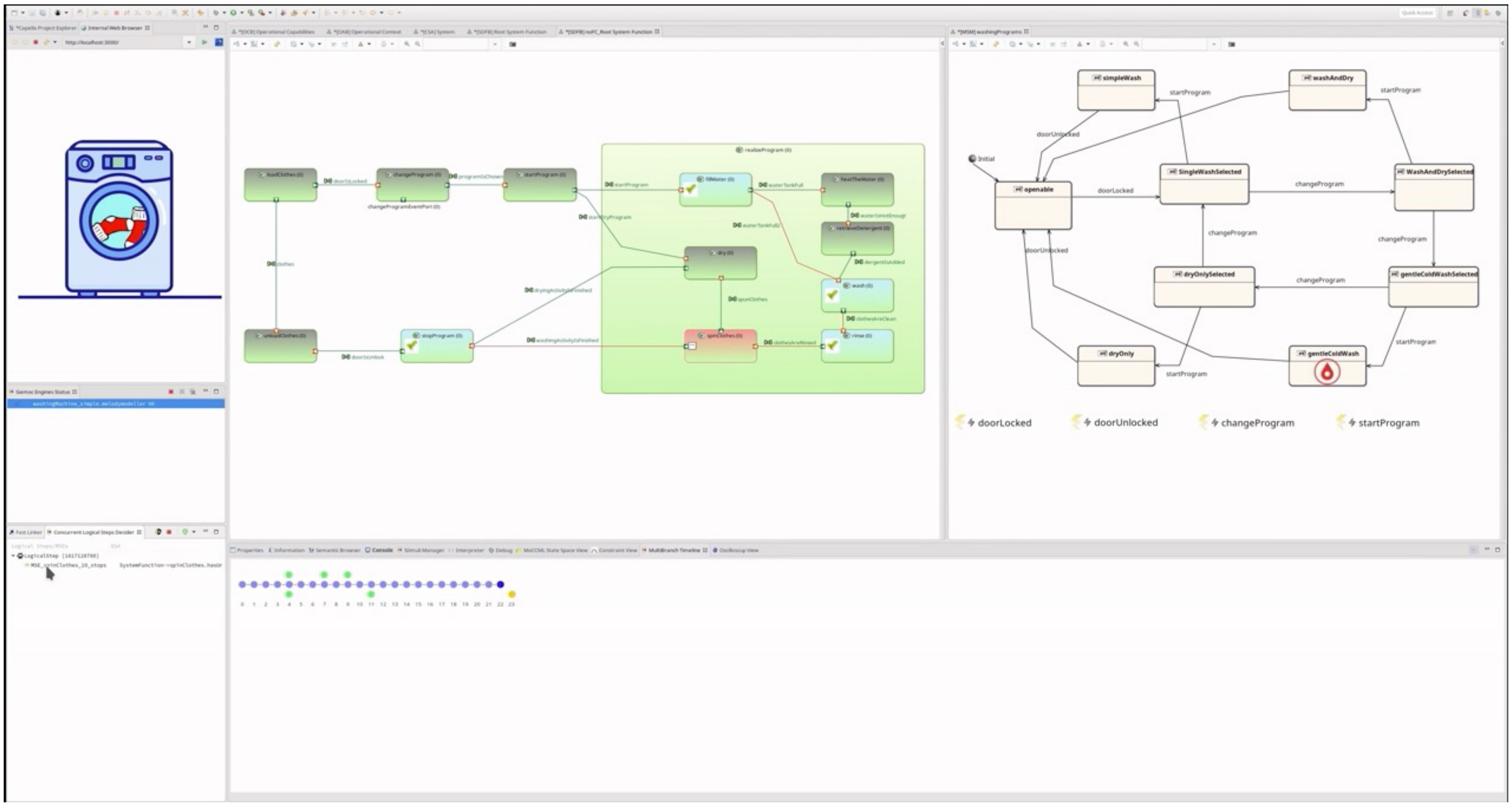
https://github.com/gemoc/arduinomodeling

UML Activity Diagram

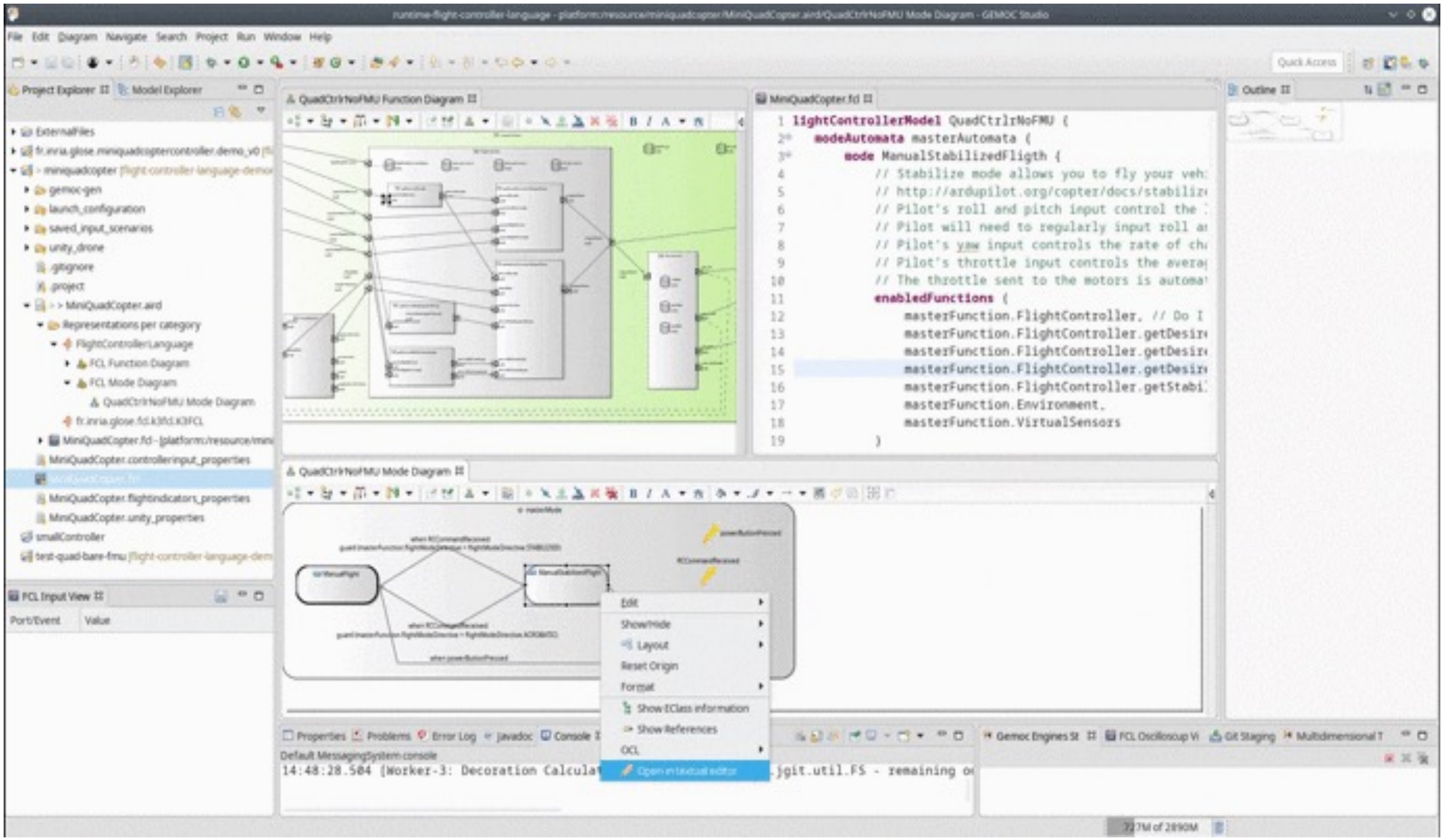


https://github.com/gemoc/activitydiagram

xCapella



FCL


QuadCtrlrNoFMU Function Diagram
MiniQuadCopter.fcl
lightControllerModel QuadCtrlrNoFMU {
modeAutomata masterAutomata {
mode ManualStabilizedFligth {
// Stabilize mode allows you to fly your veh
// http://ardupilot.org/copter/docs/stabiliz
// Pilot's roll and pitch input control the
// Pilot will need to regularly input roll a
// Pilot's yaw input controls the rate of ch
// Pilot's throttle input controls the avera
// The throttle sent to the motors is automa
enabledFunctions {
masterFunction.FlightController, // Do I
masterFunction.FlightController.getDesir
masterFunction.FlightController.getDesir
masterFunction.FlightController.getDesir
masterFunction.FlightController.getStabi
masterFunction.Environment,
masterFunction.VirtualSensors
}
QuadCtrlrNoFMU Mode Diagram
Project Explorer
Model Explorer
Outline
FCL Input View
Port/Event
Value
Edit
Show/Hide
Layout
Reset Origin
Format
Show EClass information
Show References
OCL
Open in textual editor
Properties
Problems
Error Log
Javadoc
Console
Default MessagingSystem console
14:48:28.504 [Worker-3: Decoration Calcula
jgit.util.FS - remaining o
Gemoc Engines St
FCL Oscilloscup Vi
Git Staging
Multidimensional T