MODEL EXECUTION

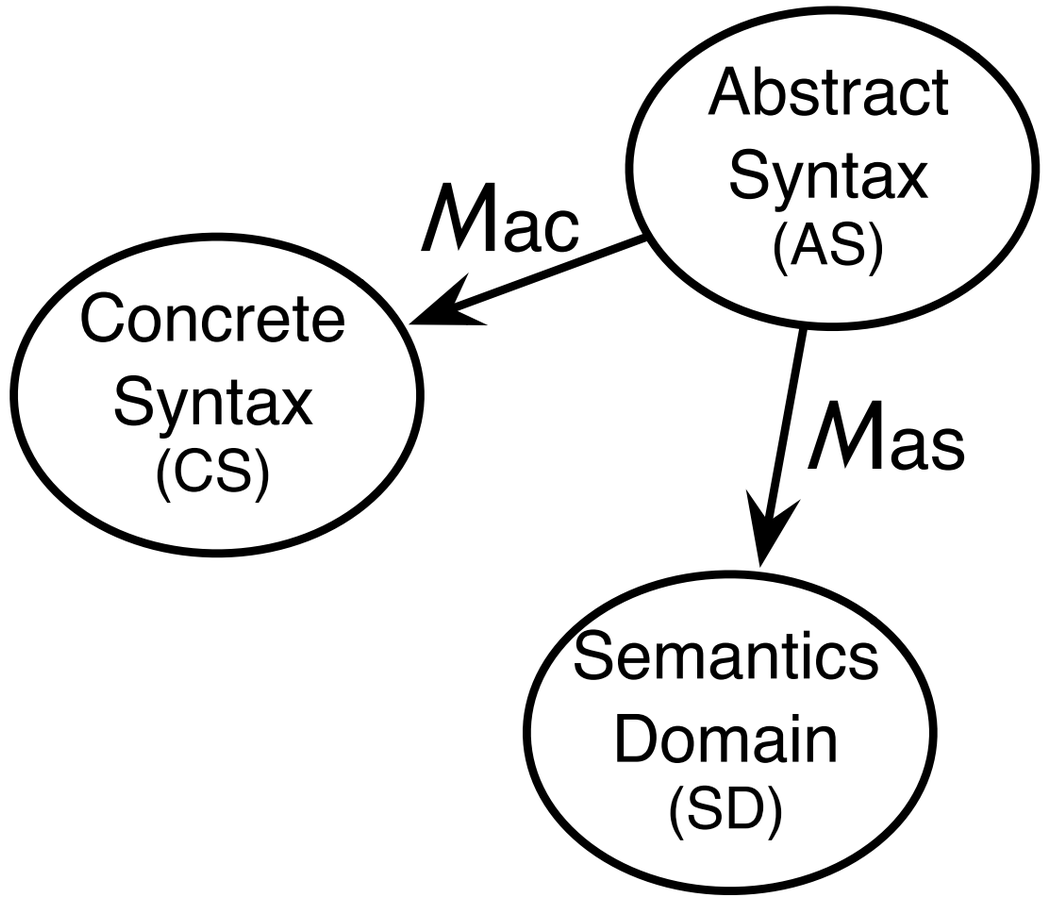
BUILD YOUR OWN VM FOR YOUR LANGUAGE

MASTER 1 ICE, 2017-2018

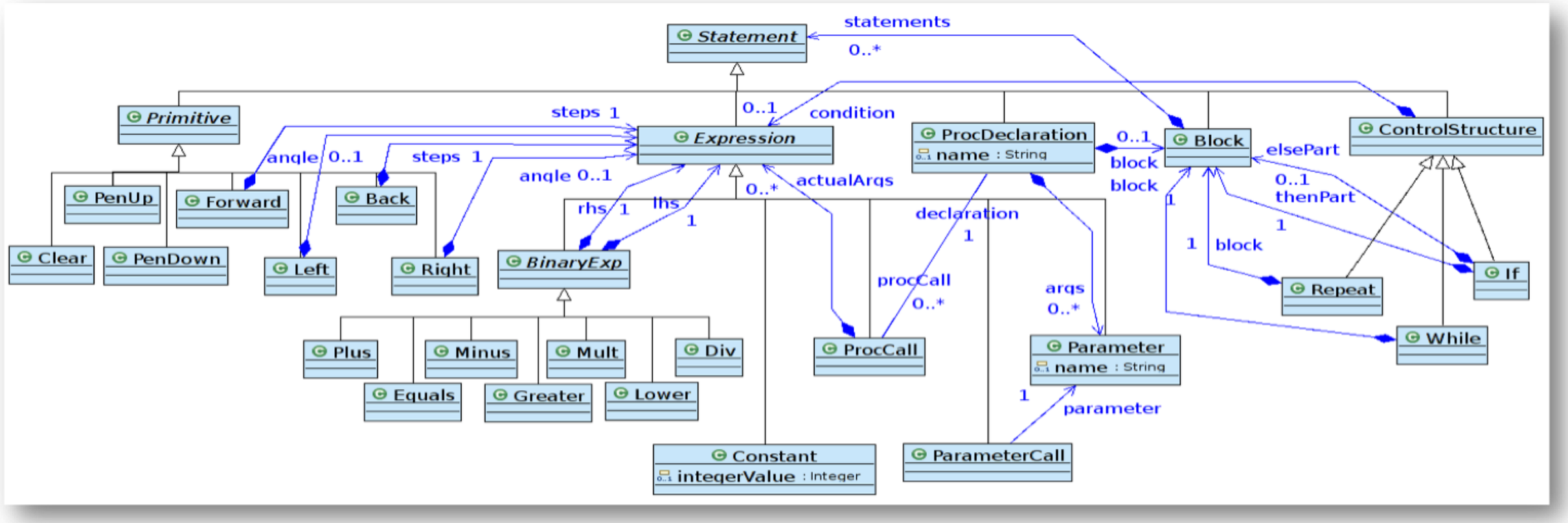
BENOIT COMBEMALE
PROFESSOR, UNIV. TOULOUSE, FRANCE

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

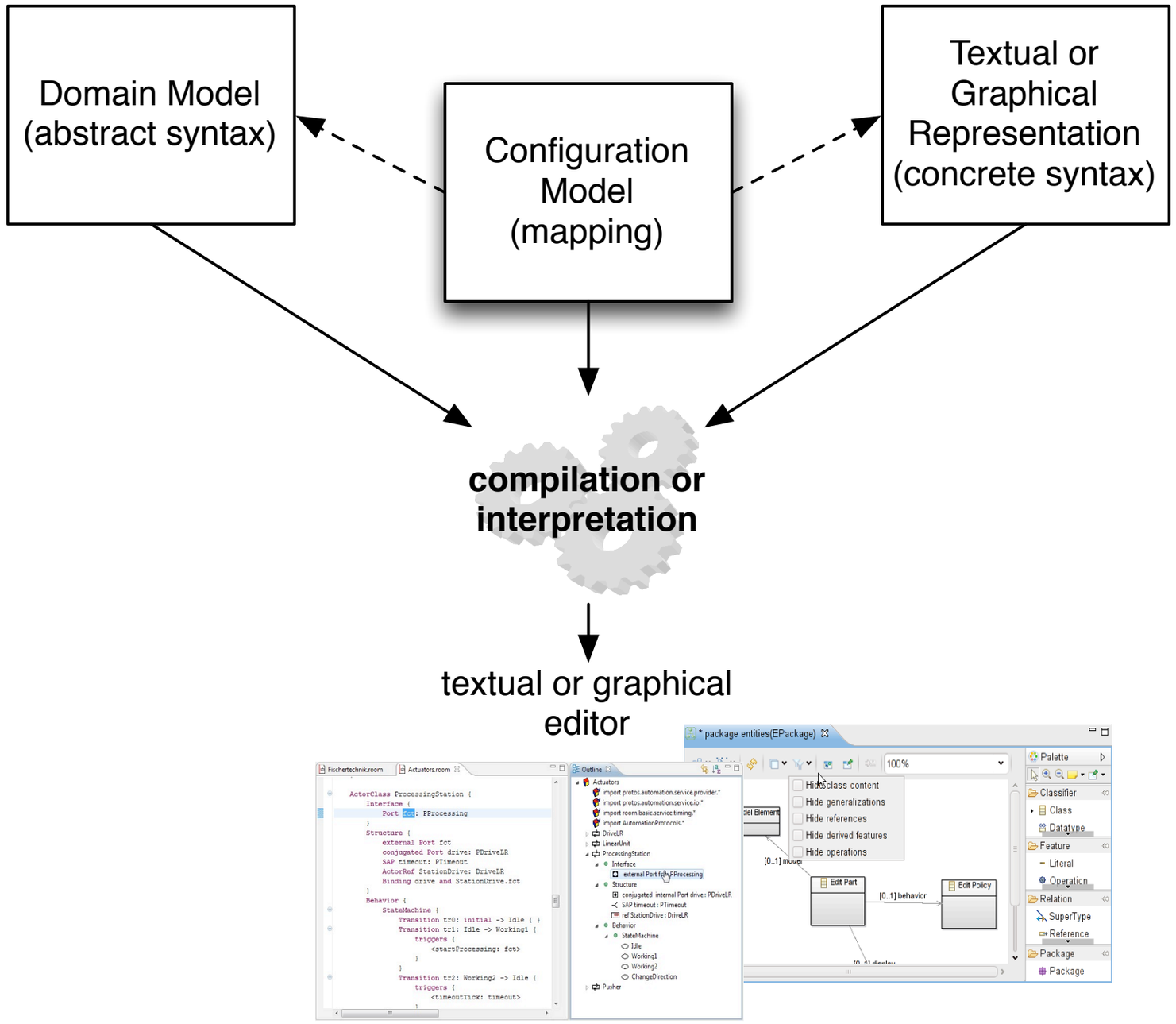
Reminder about what is a language



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

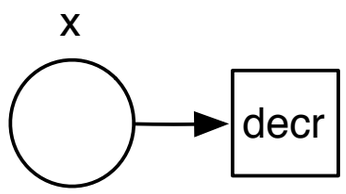
```
int x;
void decr () {
        if ( x>0 )
        x = x−1;
}
```

System
x : Int
decr()

- Axiomatic

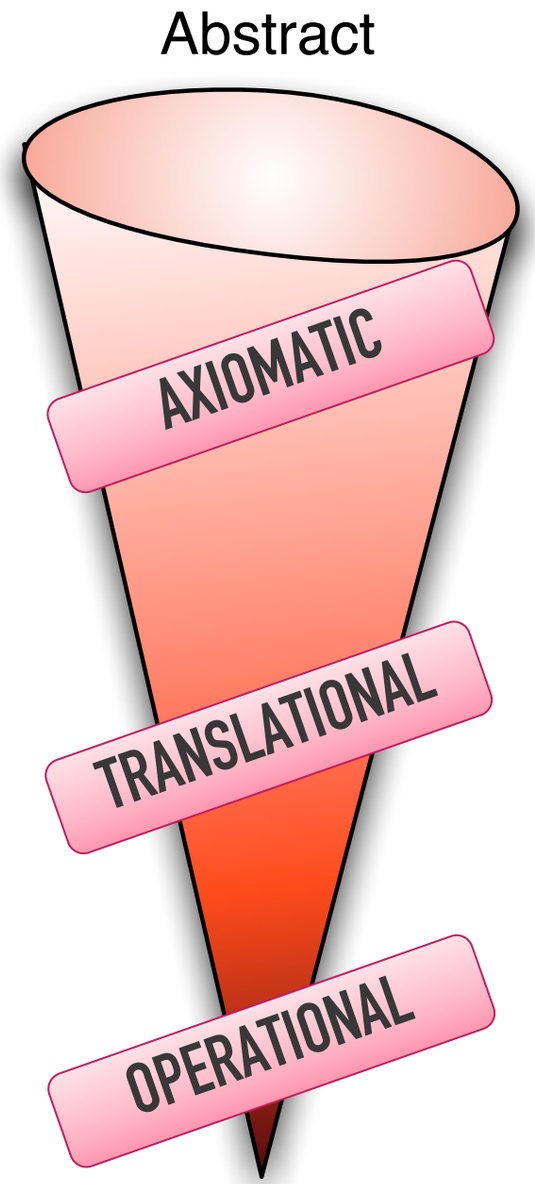
```
context System::decr() post :
        self .x =    if ( self .x@pre>0 )
                        then self.x@pre - 1
                        else self.x@pre
                endif
```

- Denotational/translational

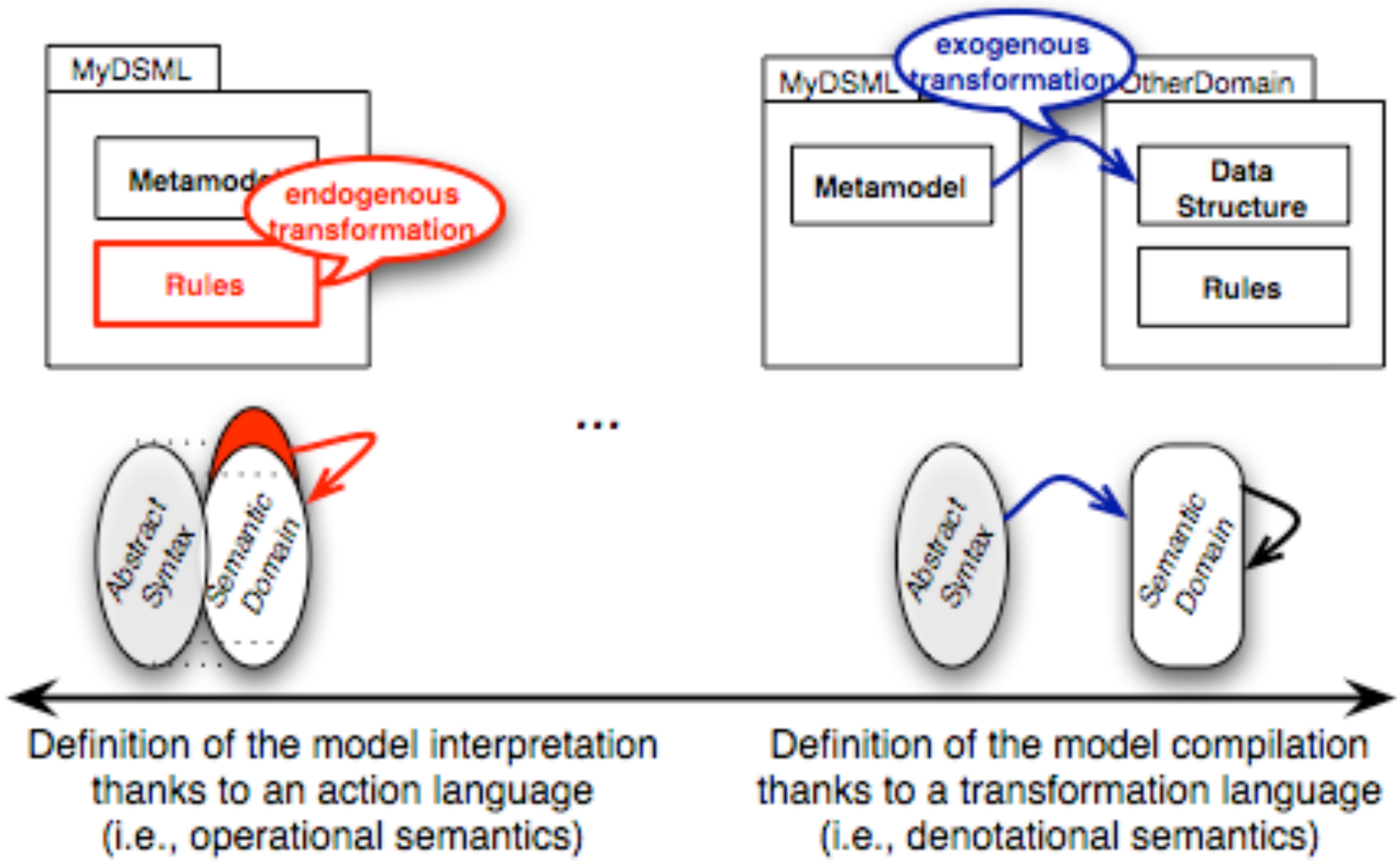


- Operational

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

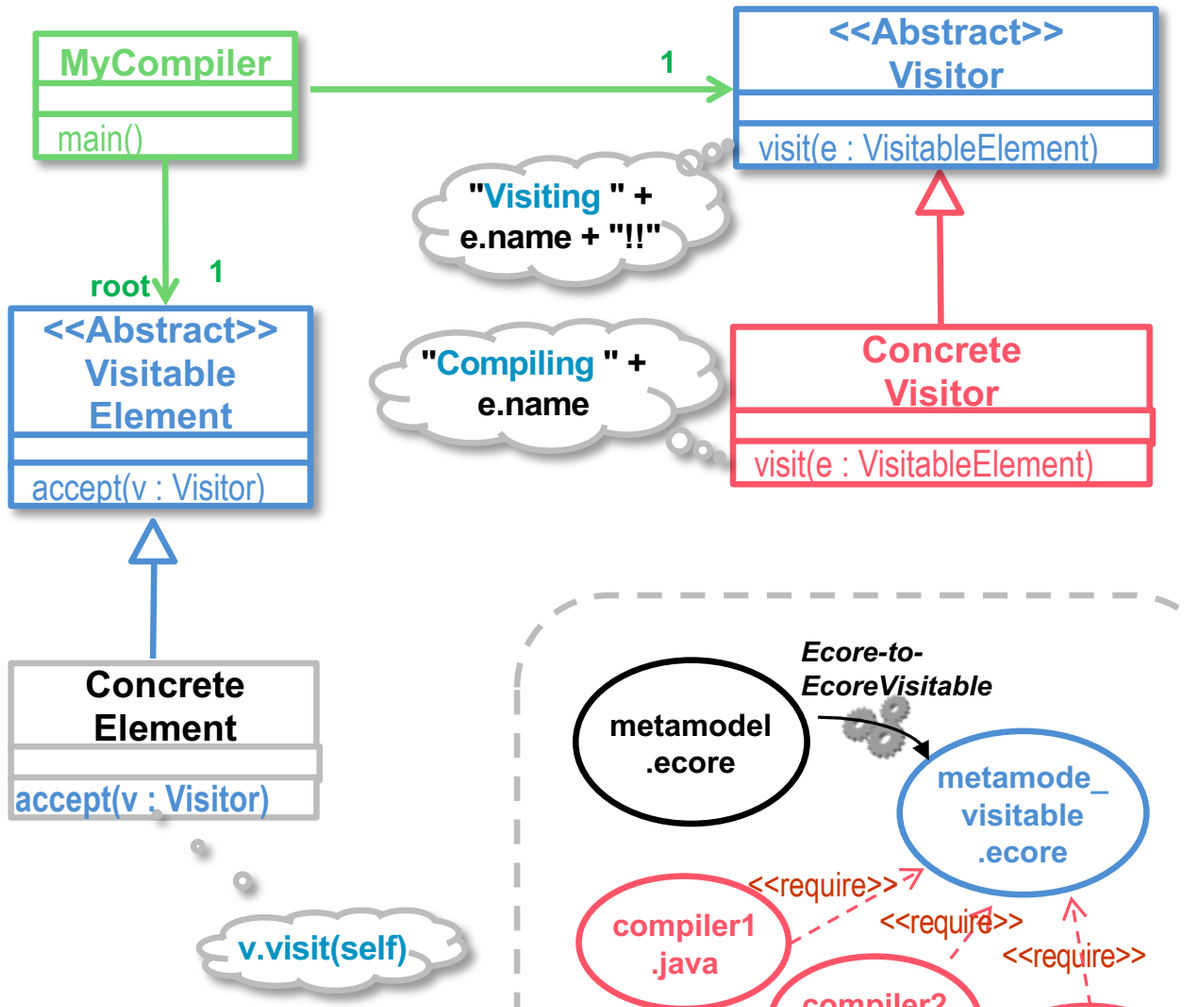


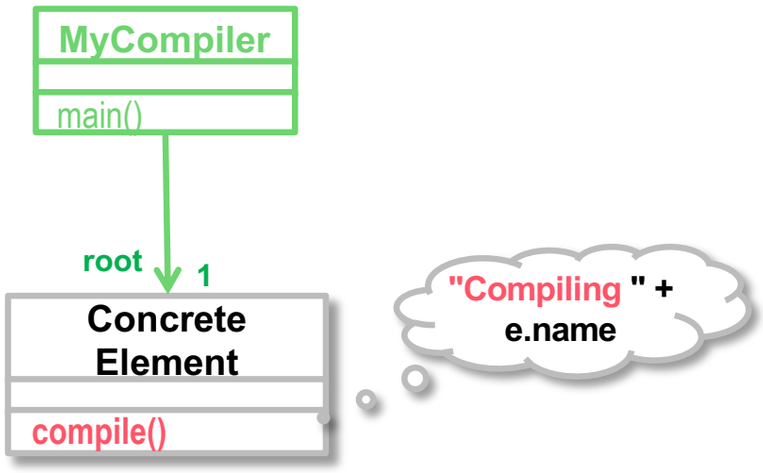
Definition of the Behavioral Semantics of DSL

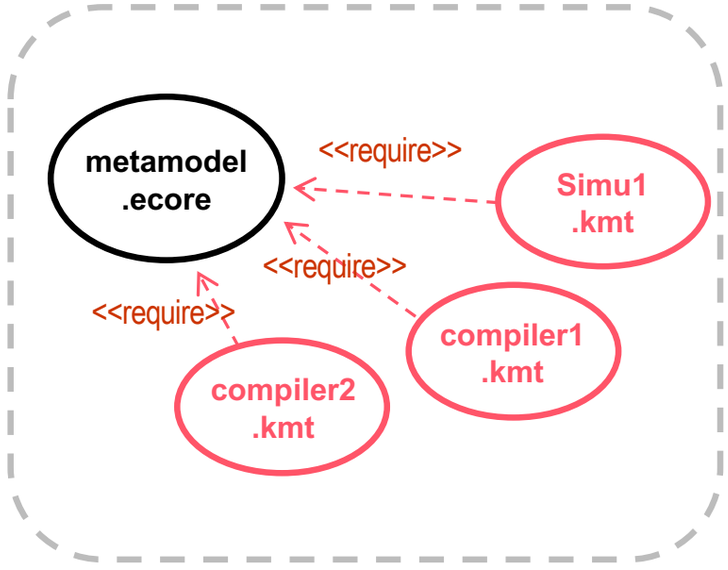


Implement your own interpreter

- Visitor-based?
  - Interpreter pattern, static introduction (aka. open class)







Implement your own interpreter with Xtend/K3

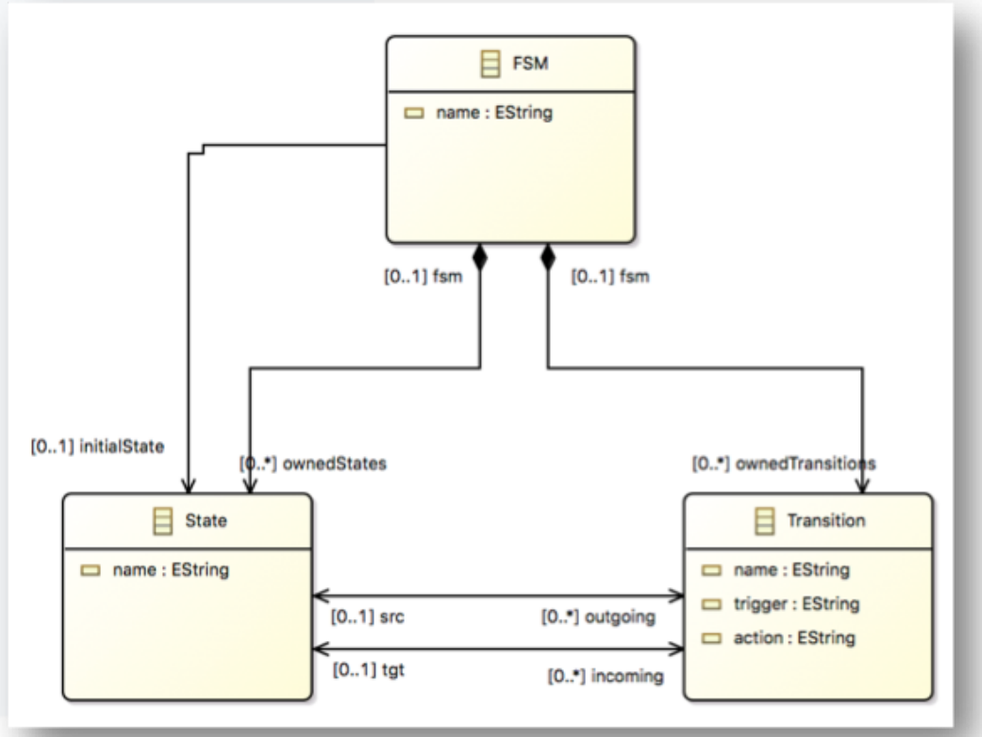
```
@Aspect(className=State)
class StateAspect {
    @Step
    def public void step(String inputString) {
            // Get the valid transitions
            val validTransitions =  _self.outgoing.filter[t | inputString.compareTo(t.trigger) == 0]

            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

}
```

Labs: Build a VM for your StateMachine DSL

- Implement the semantics with Xtend (/Kermeta)
- [optionel] Implement (or adapt from the MODELS tutorial) the graphical representation (Editor/Animator)

- Use GEMOC to get for free your modeling environment with model execution, animation and debugging

- https://github.com/gemoc/MODELS2017Tutorial

The GEMOC Studio

Language Workbench

Design and integrate your executable DSMLs

http://gemoc.org/studio

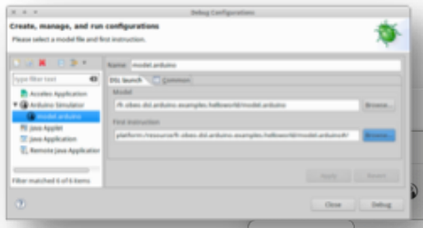
soon

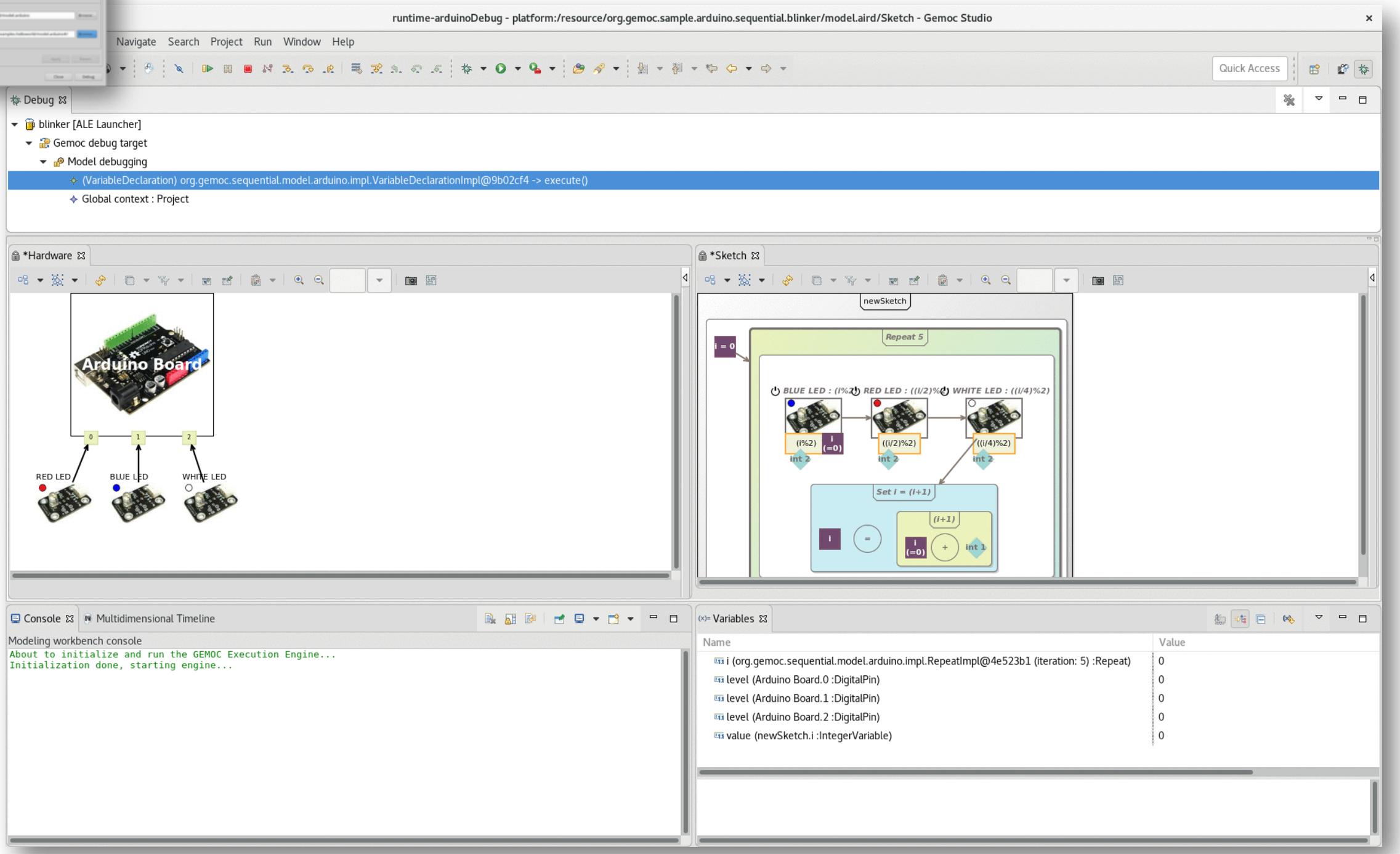
http://eclipse.org/gemoc

Modeling Workbench

Edit, simulate and animate your heterogeneous models

Arduino Designer





https://github.com/gemoc/arduinomodeling

Activity Diagram Debugger

https://github.com/gemoc/activitydiagram