Engineering Executable DSMLs (xDSMLs) for model executability, animation and debugging

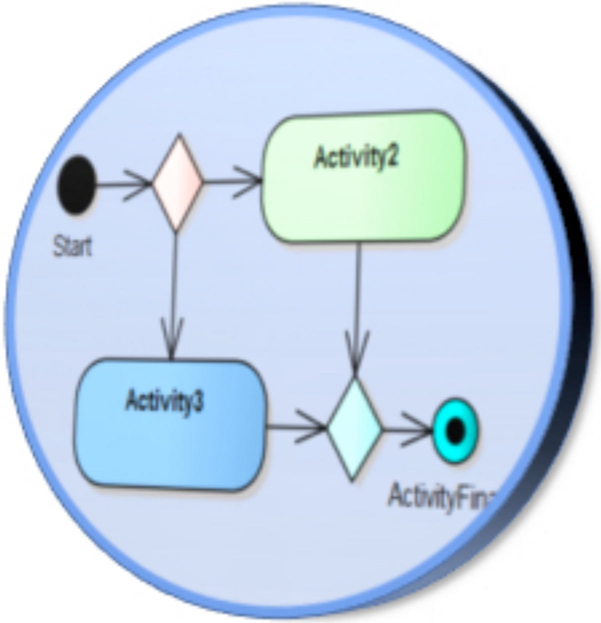
Final workshop of the ANR project GEMOC
March 17th, 2016

Benoit Combemale (Inria & Univ. Rennes 1)
http://people.irisa.fr/Benoit.Combemale
benoit.combemale@irisa.fr
@bcombemale

ANR-12-INSE-0011

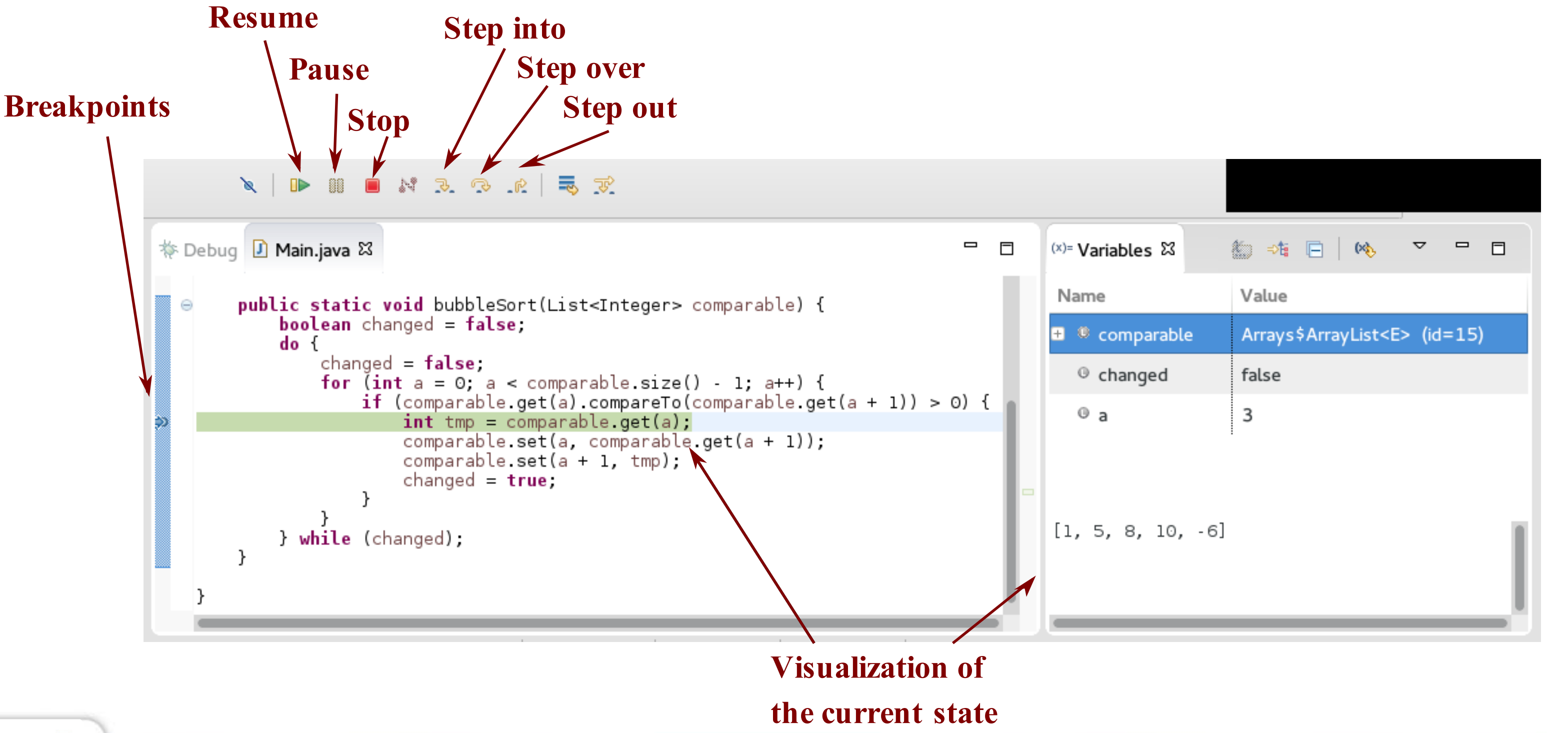
(Domain-Specific) Behavioral models

- Various engineering: software engineering, systems engineering, enterprise architecture, scientific modeling...
- Various domains: Business Processes, Orchestrations, Functional chains, Activities, Protocoles, Scenarios...
- Various analysis techniques for checking behavioral properties (early V&V)



Stepwise debugging

- Stepwise Debugging: find the cause of a defect by manually observing and controlling execution
- Central dynamic V&V activity



Resume

Pause

Stop

Step into

Step over

Step out

Breakpoints

Visualization of the current state

[redacted]

```
public static void bubbleSort(List<Integer> comparable) {
    boolean changed = false;
    do {
        changed = false;
        for (int a = 0; a < comparable.size() - 1; a++) {
            if (comparable.get(a).compareTo(comparable.get(a + 1)) > 0) {
                int tmp = comparable.get(a);
                comparable.set(a, comparable.get(a + 1));
                comparable.set(a + 1, tmp);
                changed = true;
            }
        }
    } while (changed);
}

}
```

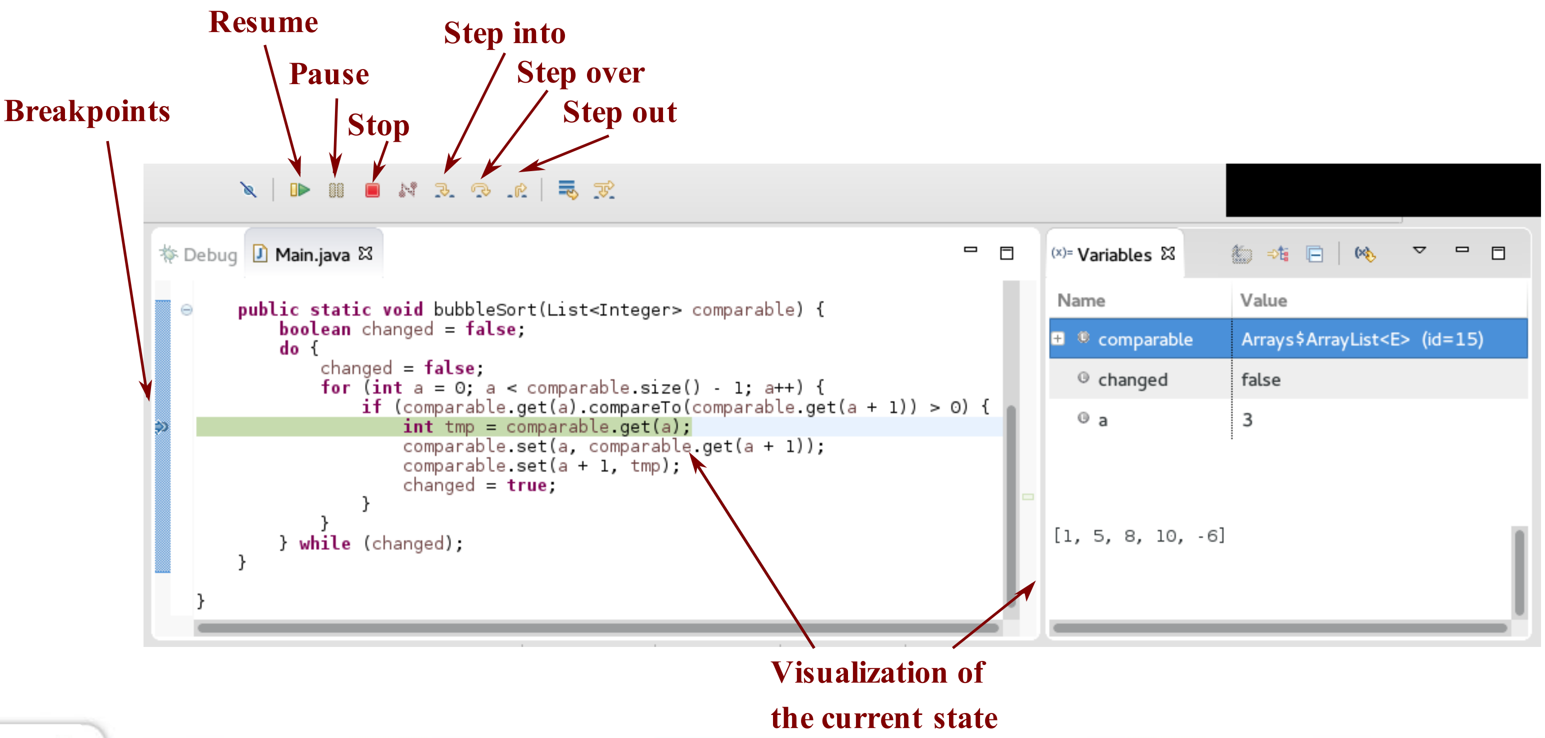
Name	Value
comparable	Arrays$ArrayList<E> (id=15)
changed	false
a	3

```
[1, 5, 8, 10, -6]
```

Stepwise debugging

- Intuitive model comprehension technique
  - No abstraction gap
  - Better turn-arounds

⇒ Fast convergence towards an acceptable design



[redacted]

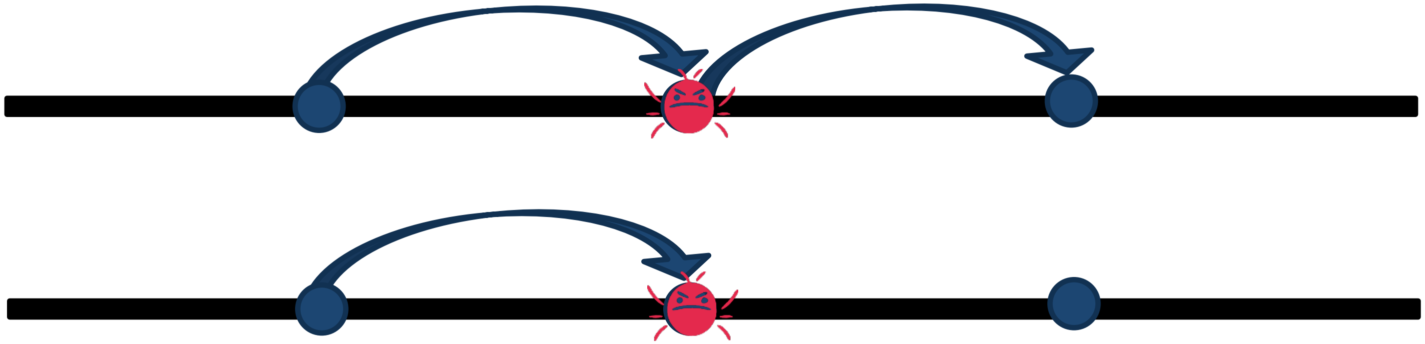
```
public static void bubbleSort(List<Integer> comparable) {
    boolean changed = false;
    do {
        changed = false;
        for (int a = 0; a < comparable.size() - 1; a++) {
            if (comparable.get(a).compareTo(comparable.get(a + 1)) > 0) {
                int tmp = comparable.get(a);
                comparable.set(a, comparable.get(a + 1));
                comparable.set(a + 1, tmp);
                changed = true;
            }
        }
    } while (changed);
}

}
```

Name	Value
comparable	Arrays$ArrayList<E> (id=15)
changed	false
a	3

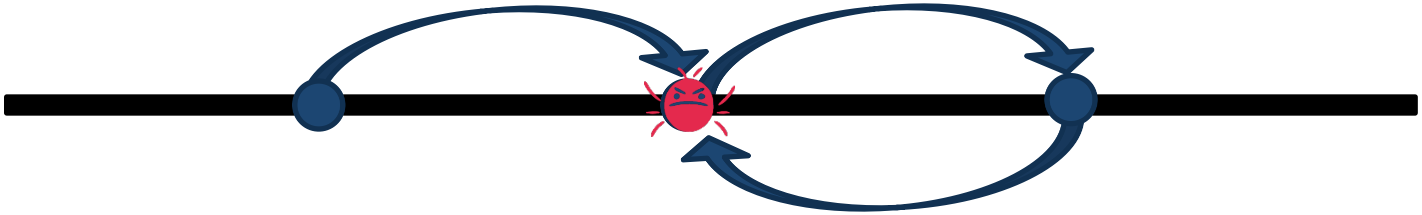
```
[1, 5, 8, 10, -6]
```

Omniscient debugging

- Stepwise debuggers only go forward

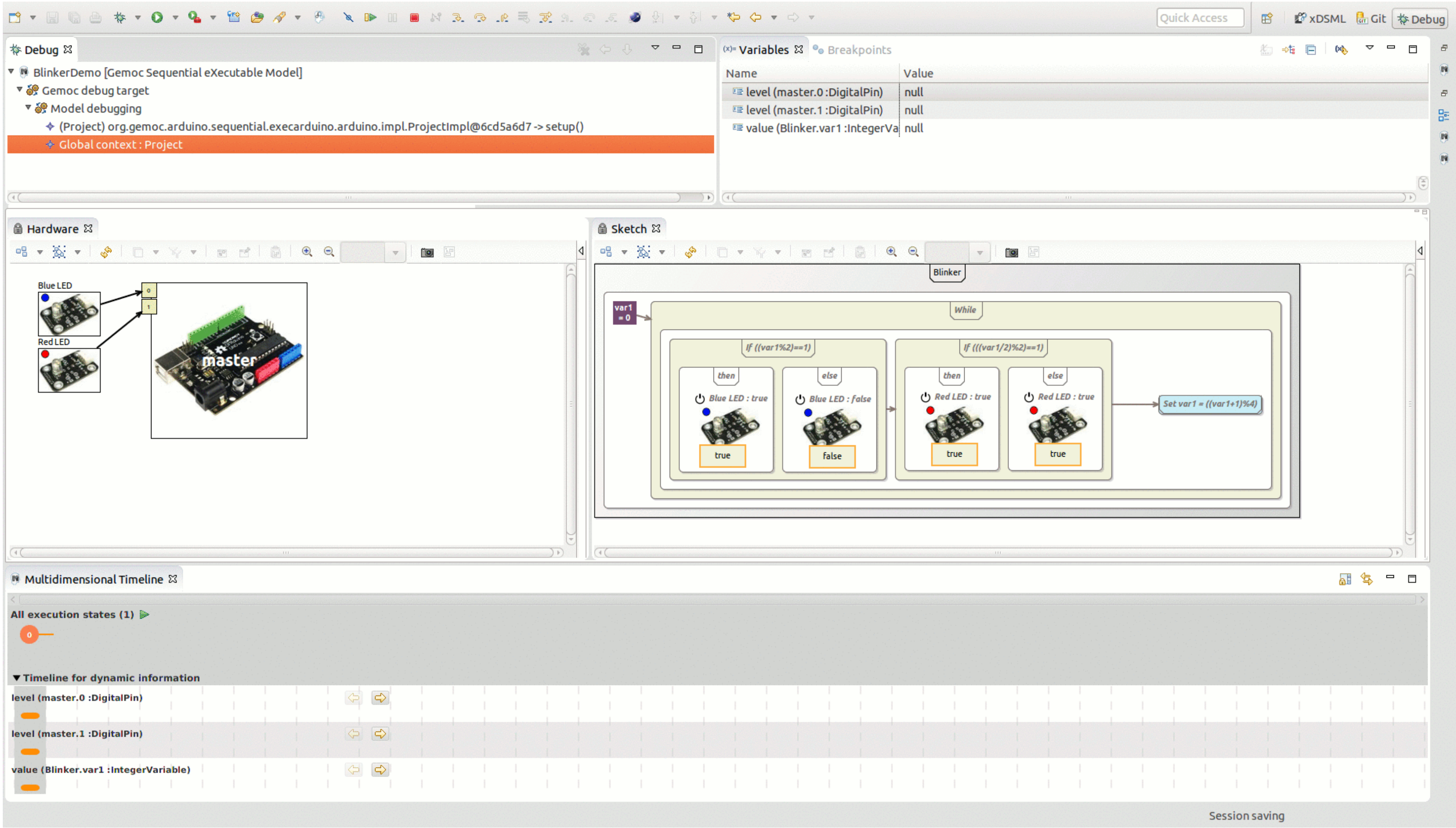


- Omniscient debuggers go forward and backward



- Omniscient debuggers typically rely on an **execution trace** storing previous states.

Expected Result for Activity Diagram



```
▼ BlinkerDemo [Gemoc Sequential eXecutable Model]
  ▼ Gemoc debug target
    ▼ Model debugging
        (Project) org.gemoc.arduino.sequential.execarduino.arduino.impl.ProjectImpl@6cd5a6d7 -> setup()
        Global context : Project
```

Name	Value
level (master.0 :DigitalPin)	null
level (master.1 :DigitalPin)	null
value (Blinker.var1 :IntegerVa	null

All execution states (1)

▼ Timeline for dynamic information

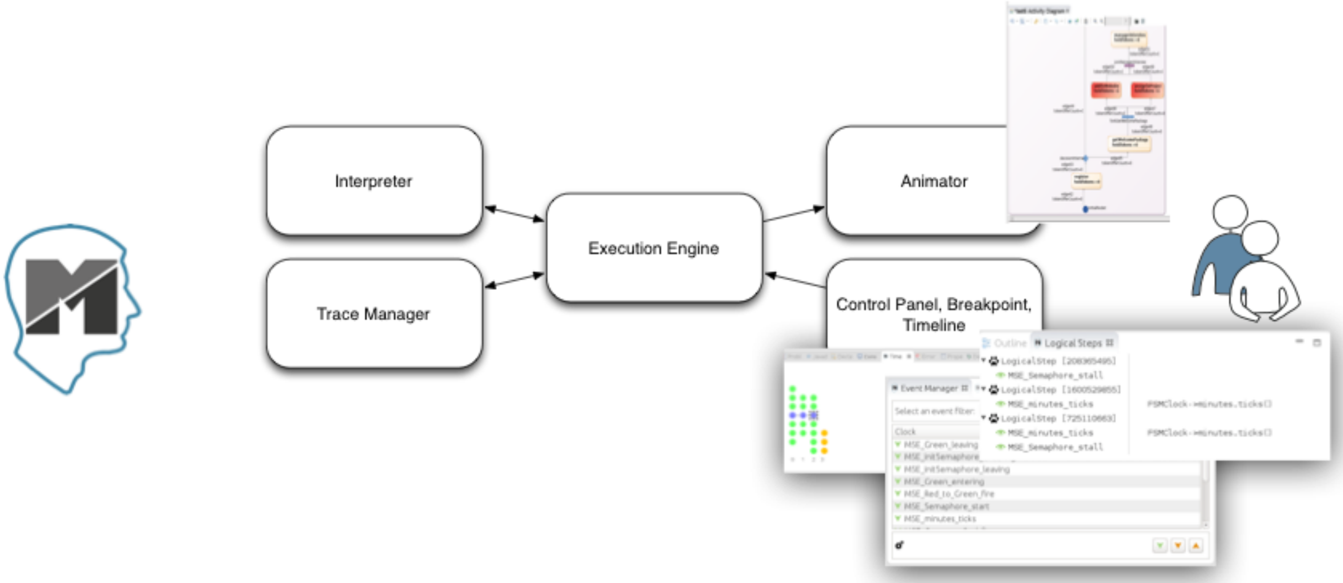
level (master.0 :DigitalPin)

level (master.1 :DigitalPin)

value (Blinker.var1 :IntegerVariable)

HOW TO PROVIDE EXECUTION, ANIMATION AND DEBUGGING FACILITIES FOR ANY EXECUTABLE DOMAIN-SPECIFIC MODELING LANGUAGE (xDSML)?

Required Tools



Interpreter

Trace Manager

Execution Engine

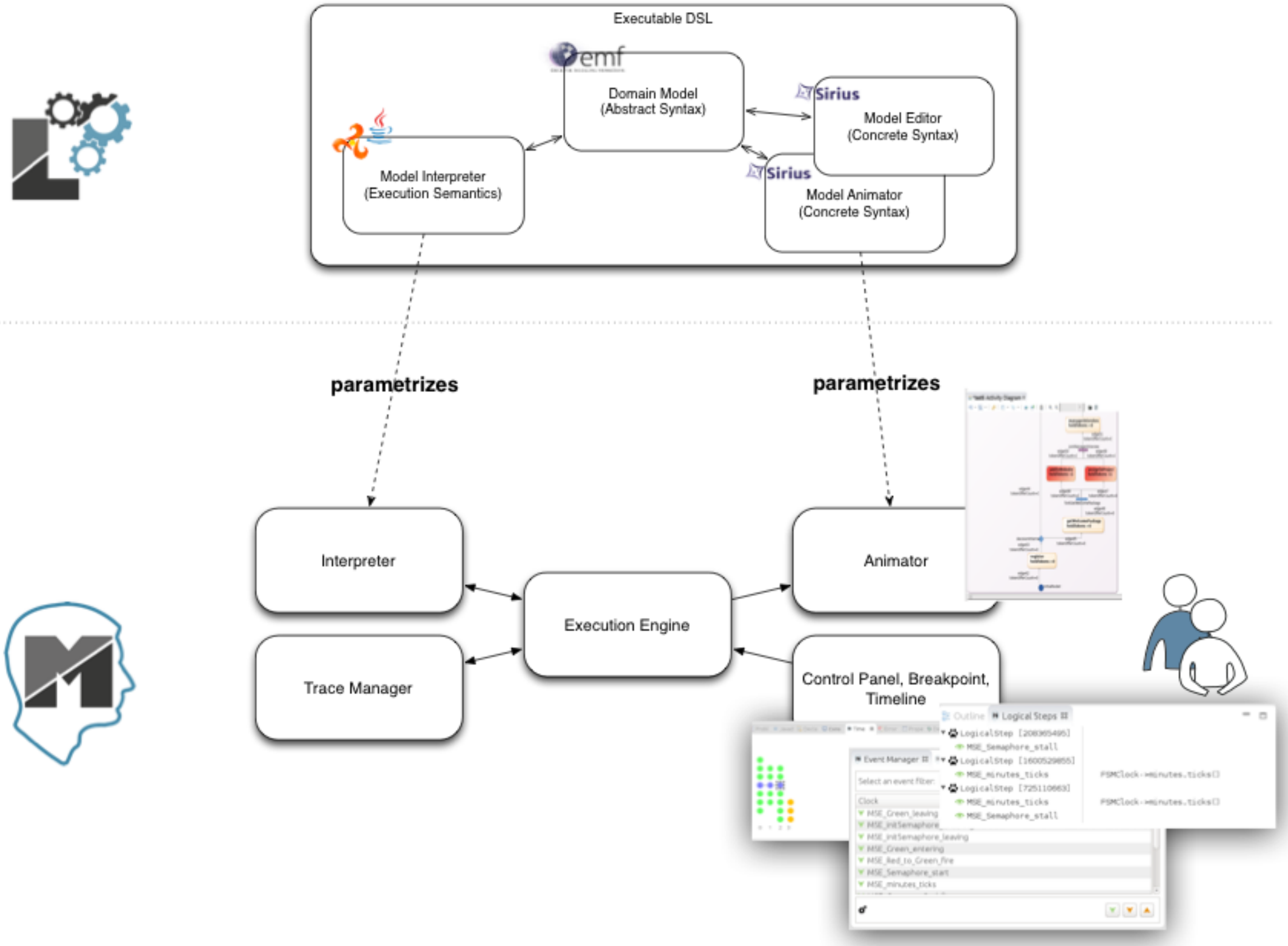
Animator

Control Panel, Breakpoint, Timeline

- LogicalStep [208365495]
  - MSE_Semaphore_stall
- LogicalStep [1600529855]
  - MSE_minutes_ticks FSMClock->minutes.ticks()
- LogicalStep [725110663]
  - MSE_minutes_ticks FSMClock->minutes.ticks()
  - MSE_Semaphore_stall

MSE_Green_leaving
MSE_initSemaphore
MSE_initSemaphore_leaving
MSE_Green_entering
MSE_Red_to_Green_fire
MSE_Semaphore_start
MSE_minutes_ticks

Required Tools



Proposed Approach

- Xtend/Kermeta to define the interpreter
- Sirius to define the animator by extension of the tooling description

- A generative approach for the trace manager
- A generic execution engine
- A generic control panel and timeline

Proposed Approach

- Leverage the GEMOC Execution Framework
  - Start the execution (@main), and initialize the model (@initialize)
  - Encapsulate stepwise execution in transactions, and control the execution (step-by-step, pause, resume)
  - Integration with the trace manager
- Leverage the Sirius Animation Runtime
  - Bridge the Eclipse Debug APIs and the EMF APIs, incl. the control panel with step (back) over/into/return
  - Transmit events and requests
  - Initialize the tooling extension
  - Provide off-the-shelf ecore model for runtime data

Animating Arduino Designer

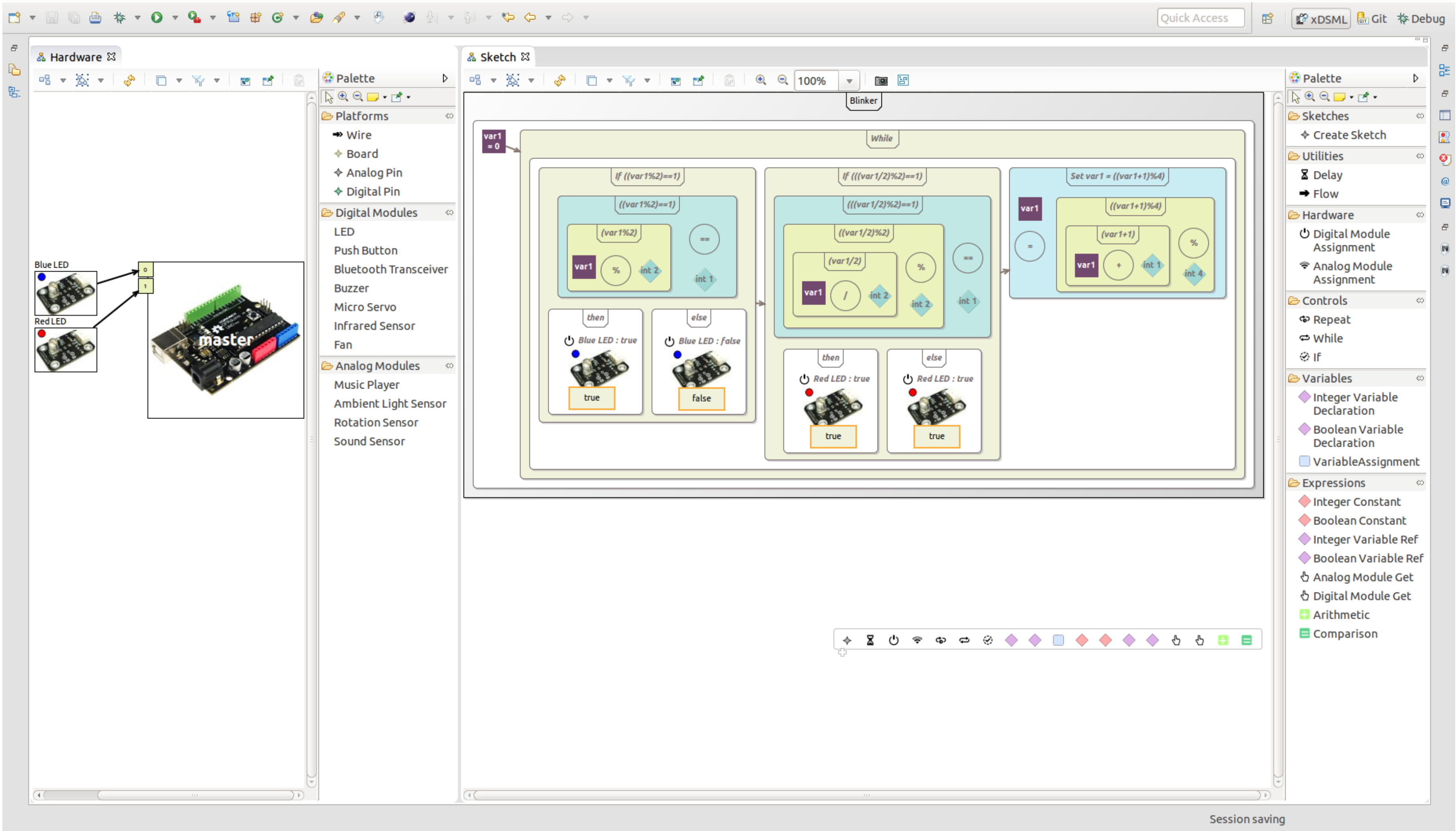
Arduino Designer

Powered by Sirius

1.0

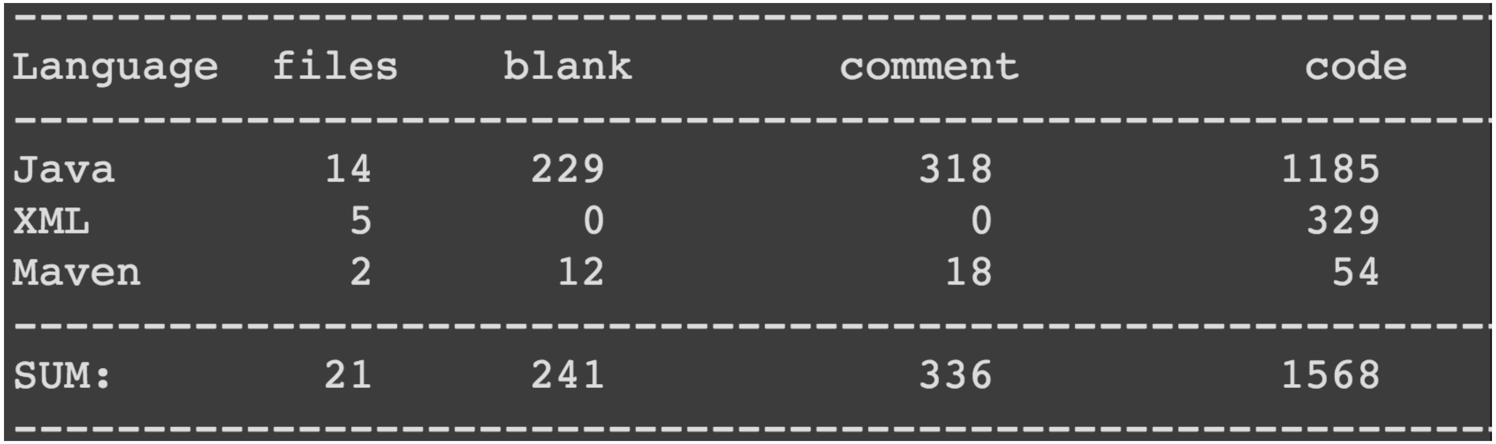
(c) 2010 - 2013 Obeo - All rights reserved

Animating Arduino Designer



Animating Arduino Designer

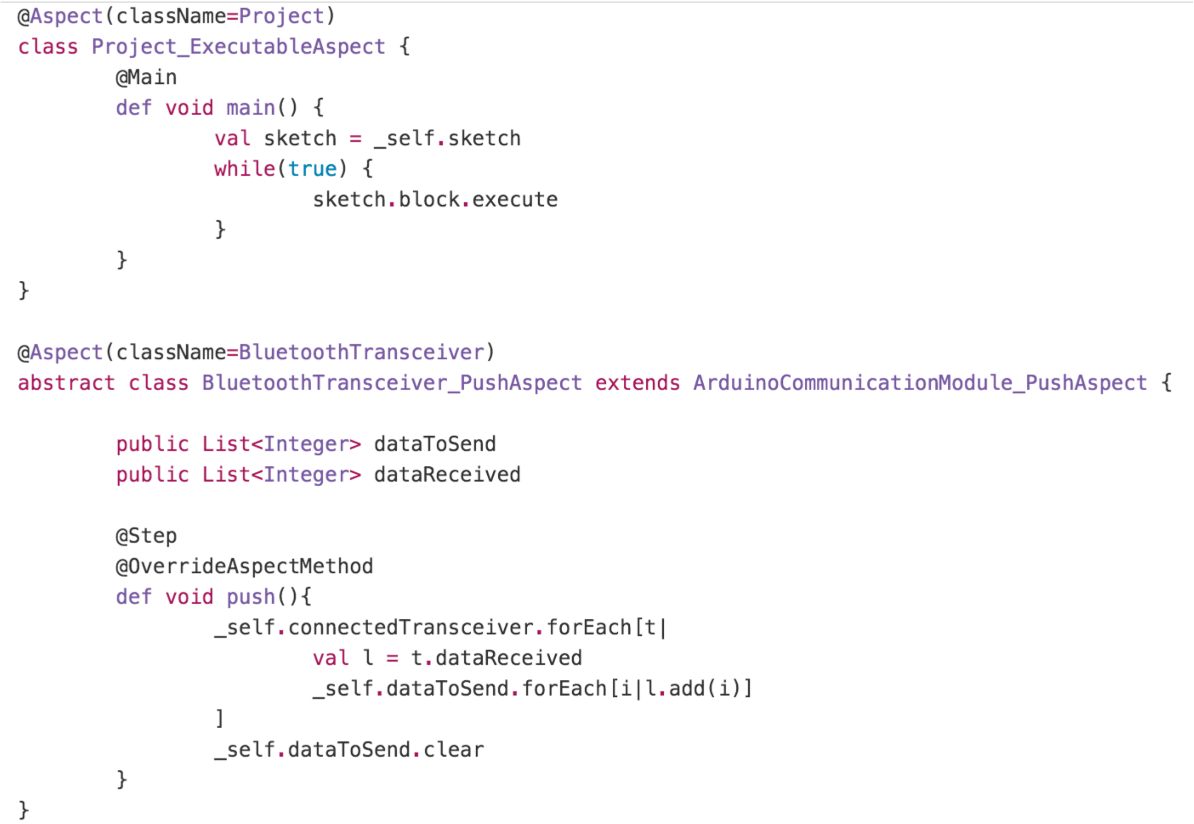
- fr.obeo.dsl.arduino.simulator
  - Interpreter, incl. execution functions and data
- fr.obeo.dsl.arduino.simulator.design
  - Animator (representation of the execution data)



```
---------------------------------------------------------
Language   files     blank          comment          code
---------------------------------------------------------
Java          14       229              318          1185
XML            5         0                0           329
Maven          2        12               18            54
---------------------------------------------------------
SUM:          21       241              336          1568
---------------------------------------------------------
```

Animating Arduino Designer

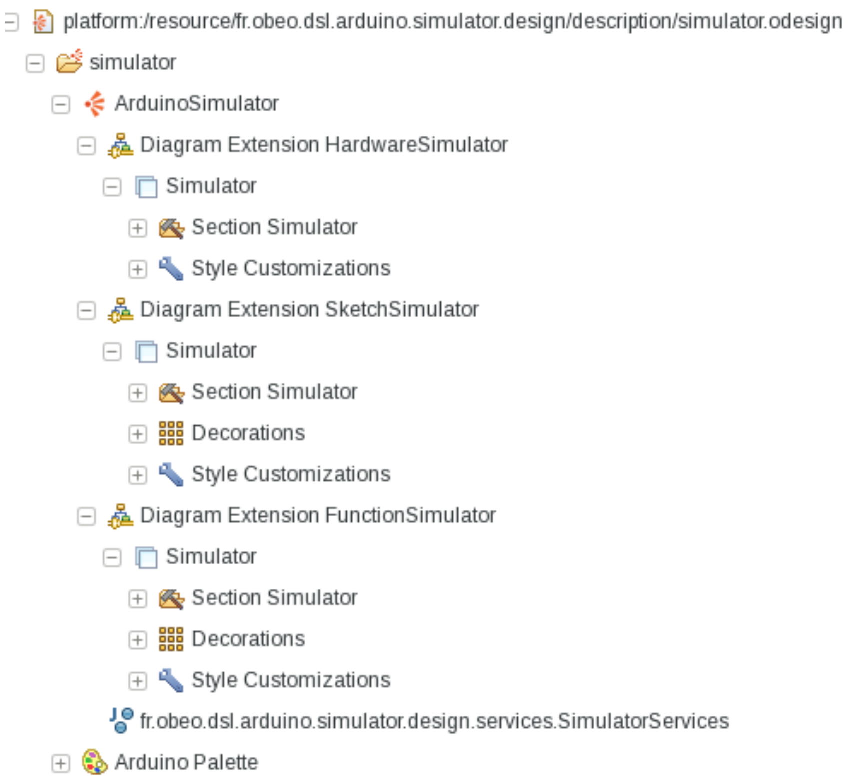
- fr.obeo.dsl.arduino.simulator
  - Interpreter, incl. execution functions and data



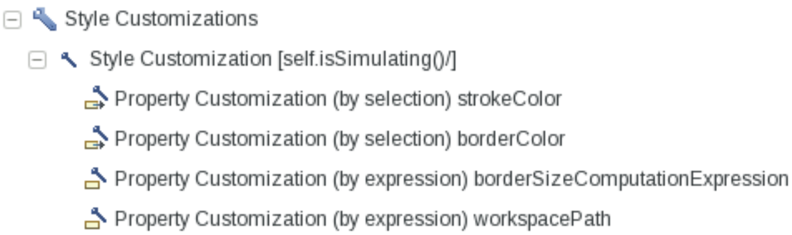
```
@Aspect(className=Project)
class Project_ExecutableAspect {
        @Main
        def void main() {
                val sketch = _self.sketch
                while(true) {
                        sketch.block.execute
                }
        }
}

@Aspect(className=BluetoothTransceiver)
abstract class BluetoothTransceiver_PushAspect extends ArduinoCommunicationModule_PushAspect {

        public List<Integer> dataToSend
        public List<Integer> dataReceived

        @Step
        @OverrideAspectMethod
        def void push(){
                _self.connectedTransceiver.forEach[t|
                        val l = t.dataReceived
                        _self.dataToSend.forEach[i|l.add(i)]
                ]
                _self.dataToSend.clear
        }
}
```

Animating Arduino Designer

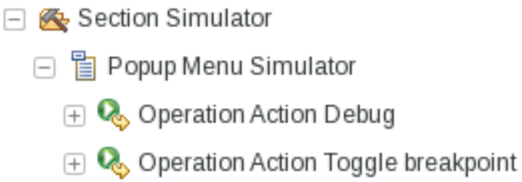
- fr.obeo.dsl.arduino.simulator.design
  - Animator (representation of the execution data)



```
⊟ platform:/resource/fr.obeo.dsl.arduino.simulator.design/description/simulator.odesign
  ⊟ simulator
    ⊟ ArduinoSimulator
      ⊟ Diagram Extension HardwareSimulator
        ⊟ Simulator
          ⊞ Section Simulator
          ⊞ Style Customizations
      ⊟ Diagram Extension SketchSimulator
        ⊟ Simulator
          ⊞ Section Simulator
          ⊞ Decorations
          ⊞ Style Customizations
      ⊟ Diagram Extension FunctionSimulator
        ⊟ Simulator
          ⊞ Section Simulator
          ⊞ Decorations
          ⊞ Style Customizations
      fr.obeo.dsl.arduino.simulator.design.services.SimulatorServices
    ⊞ Arduino Palette
```

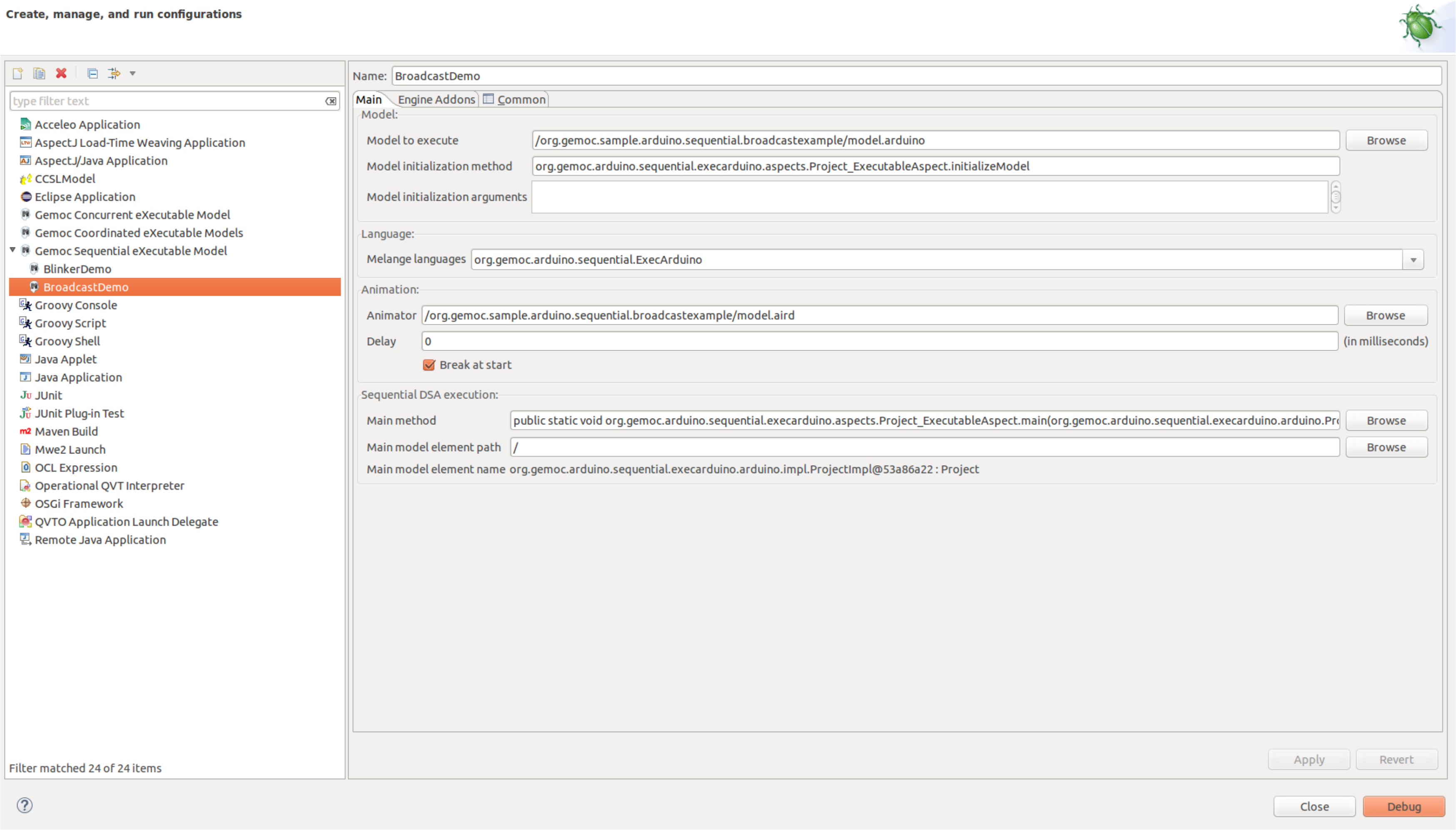


```
⊟ Style Customizations
  ⊟ Style Customization [self.isSimulating()/]
      Property Customization (by selection) strokeColor
      Property Customization (by selection) borderColor
      Property Customization (by expression) borderSizeComputationExpression
      Property Customization (by expression) workspacePath
```



```
⊟ Section Simulator
  ⊟ Popup Menu Simulator
    ⊞ Operation Action Debug
    ⊞ Operation Action Toggle breakpoint
```

Animating Arduino Designer



Create, manage, and run configurations

type filter text

- Acceleo Application
- AspectJ Load-Time Weaving Application
- AspectJ/Java Application
- CCSLModel
- Eclipse Application
- Gemoc Concurrent eXecutable Model
- Gemoc Coordinated eXecutable Models
- Gemoc Sequential eXecutable Model
  - BlinkerDemo
  - BroadcastDemo
- Groovy Console
- Groovy Script
- Groovy Shell
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Maven Build
- Mwe2 Launch
- OCL Expression
- Operational QVT Interpreter
- OSGi Framework
- QVTO Application Launch Delegate
- Remote Java Application

Filter matched 24 of 24 items

Name: BroadcastDemo

Model:

Model to execute /org.gemoc.sample.arduino.sequential.broadcastexample/model.arduino

Model initialization method org.gemoc.arduino.sequential.execarduino.aspects.Project_ExecutableAspect.initializeModel

Model initialization arguments

Language:

Melange languages org.gemoc.arduino.sequential.ExecArduino

Animation:

Animator /org.gemoc.sample.arduino.sequential.broadcastexample/model.aird

Delay 0 (in milliseconds)

☑ Break at start

Sequential DSA execution:

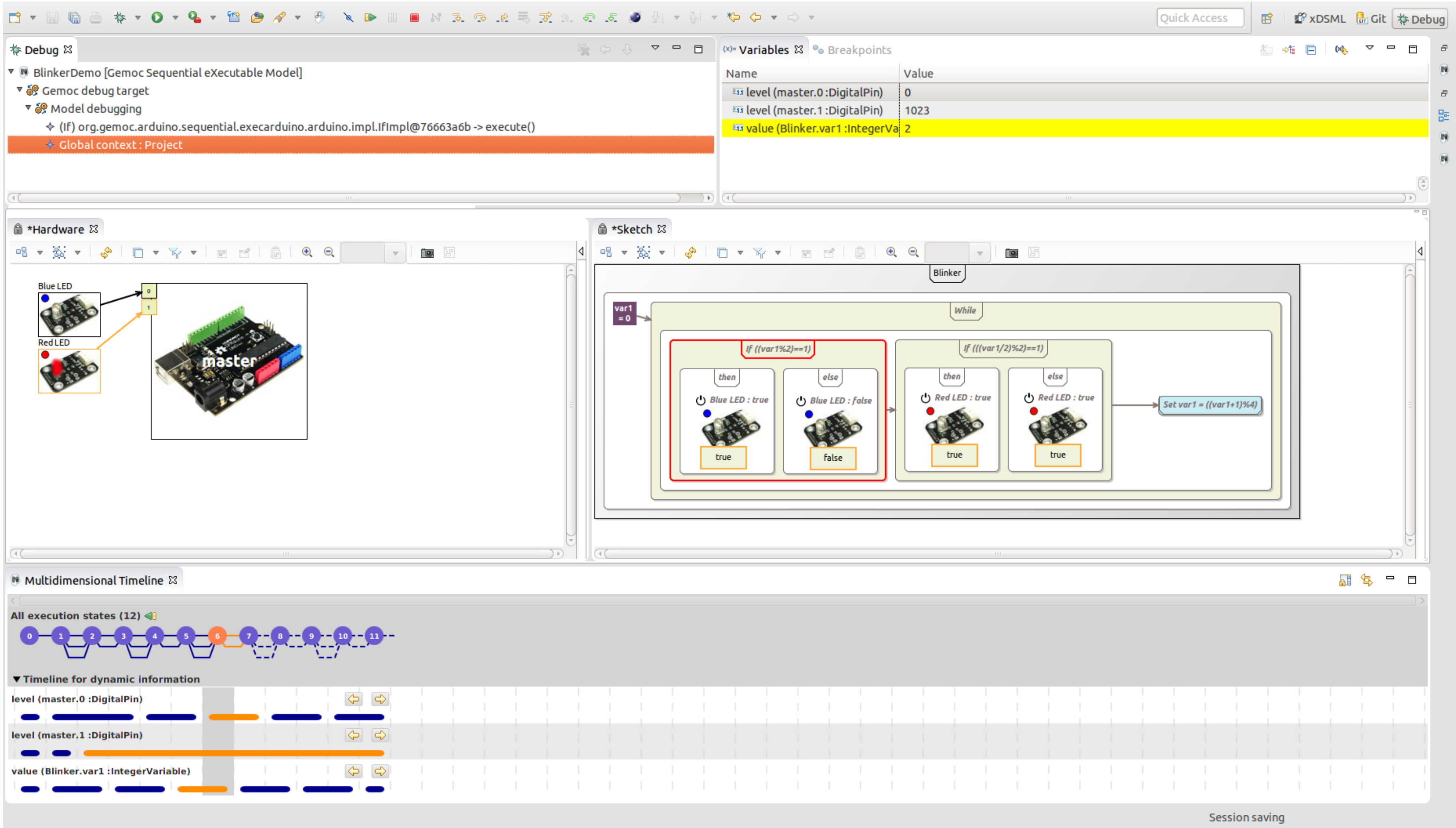
Main method public static void org.gemoc.arduino.sequential.execarduino.aspects.Project_ExecutableAspect.main(org.gemoc.arduino.sequential.execarduino.arduino.Pr

Main model element path /

Main model element name org.gemoc.arduino.sequential.execarduino.arduino.impl.ProjectImpl@53a86a22 : Project

https://github.com/gemoc/arduinomodeling

Animating Arduino Designer



Debug

- BlinkerDemo [Gemoc Sequential eXecutable Model]
  - Gemoc debug target
    - Model debugging
      - (If) org.gemoc.arduino.sequential.execarduino.arduino.impl.IfImpl@76663a6b -> execute()
      - Global context : Project

Variables

Name	Value
level (master.0 :DigitalPin)	0
level (master.1 :DigitalPin)	1023
value (Blinker.var1 :IntegerVa	2

Multidimensional Timeline

All execution states (12)

Timeline for dynamic information

level (master.0 :DigitalPin)

level (master.1 :DigitalPin)

value (Blinker.var1 :IntegerVariable)

https://github.com/gemoc/arduinomodeling

Wrap-up

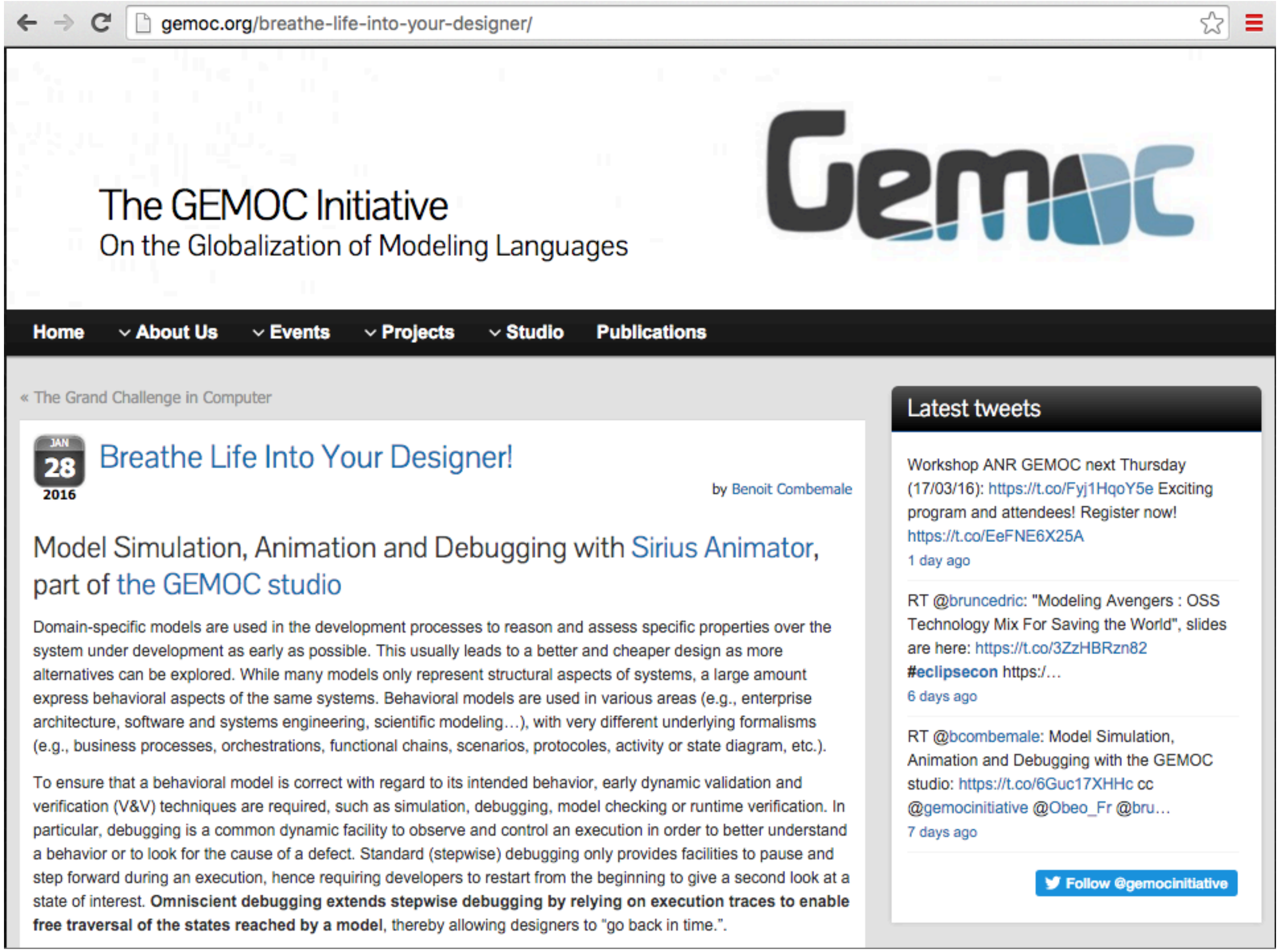
- Execution functions and data (execution semantics) weaved into Ecore model (using Xtend/Kermeta)
- Representation of the execution data as extension of the editor (using Sirius)

> Graphical animator, omnicient debugger, trace manager and timeline

Design only the features related to a given domain (execution functions and data + representation), and get for free an advanced execution, animation and debugging environment

=> Sirius Animator

DIY!



The GEMOC Initiative

On the Globalization of Modeling Languages

« The Grand Challenge in Computer

JAN 28 2016

Breathe Life Into Your Designer!

by Benoit Combemale

Model Simulation, Animation and Debugging with Sirius Animator, part of the GEMOC studio

Domain-specific models are used in the development processes to reason and assess specific properties over the system under development as early as possible. This usually leads to a better and cheaper design as more alternatives can be explored. While many models only represent structural aspects of systems, a large amount express behavioral aspects of the same systems. Behavioral models are used in various areas (e.g., enterprise architecture, software and systems engineering, scientific modeling…), with very different underlying formalisms (e.g., business processes, orchestrations, functional chains, scenarios, protocols, activity or state diagram, etc.).

To ensure that a behavioral model is correct with regard to its intended behavior, early dynamic validation and verification (V&V) techniques are required, such as simulation, debugging, model checking or runtime verification. In particular, debugging is a common dynamic facility to observe and control an execution in order to better understand a behavior or to look for the cause of a defect. Standard (stepwise) debugging only provides facilities to pause and step forward during an execution, hence requiring developers to restart from the beginning to give a second look at a state of interest. **Omniscient debugging extends stepwise debugging by relying on execution traces to enable free traversal of the states reached by a model**, thereby allowing designers to "go back in time.".

Latest tweets

Workshop ANR GEMOC next Thursday (17/03/16): https://t.co/Fyj1HqoY5e Exciting program and attendees! Register now! https://t.co/EeFNE6X25A
1 day ago

RT @bruncedric: "Modeling Avengers : OSS Technology Mix For Saving the World", slides are here: https://t.co/3ZzHBRzn82 **#eclipsecon** https:/…
6 days ago

RT @bcombemale: Model Simulation, Animation and Debugging with the GEMOC studio: https://t.co/6Guc17XHHc cc @gemocinitiative @Obeo_Fr @bru…
7 days ago

Follow @gemocinitiative

http://gemoc.org/breathe-life-into-your-designer