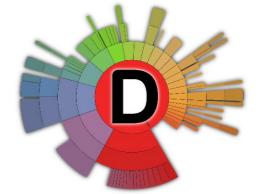


MODEL-DRIVEN (SOFTWARE) ENGINEERING

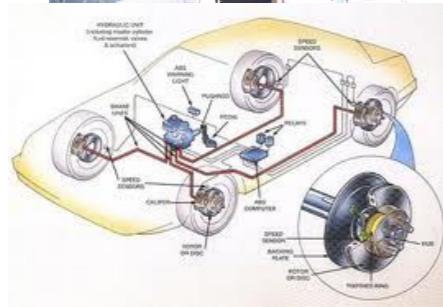
HACK YOUR OWN LANGUAGE!

UNIVERSITY OF RENNES, ESIR, 2025-2026

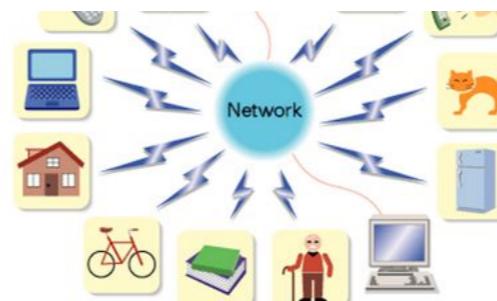
*Slides adapted from materials originally created by Prof. Benoit Combemale.
Used with permission.*



Complex Software-Intensive Systems

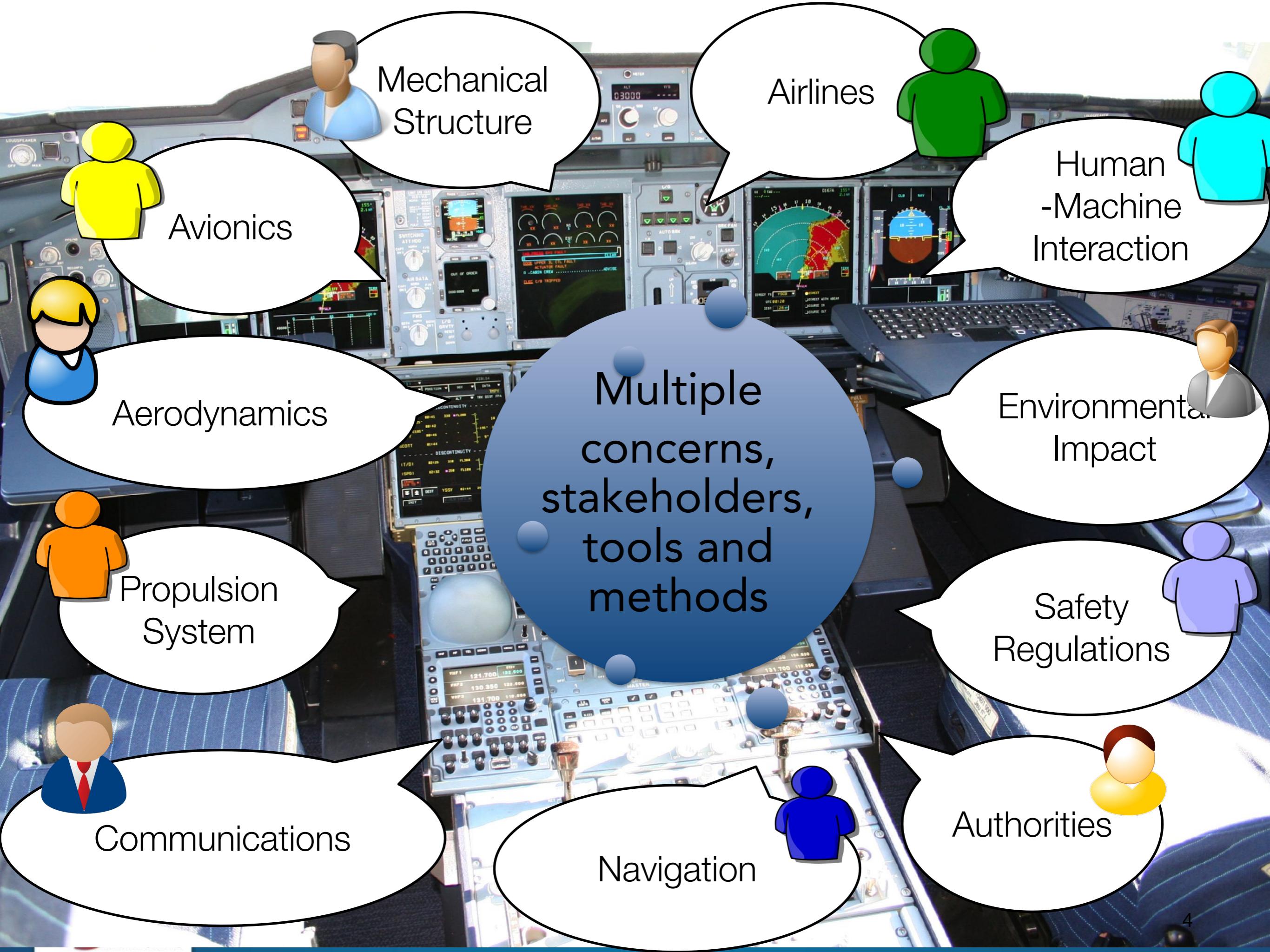


Software intensive systems



Complex Software-Intensive Systems

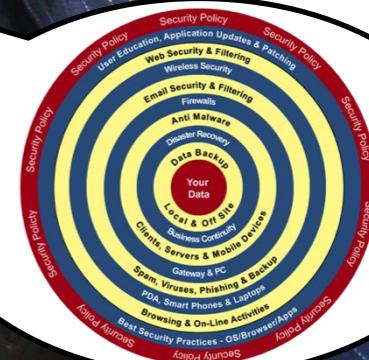
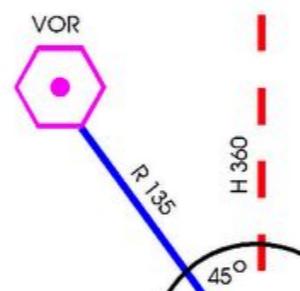
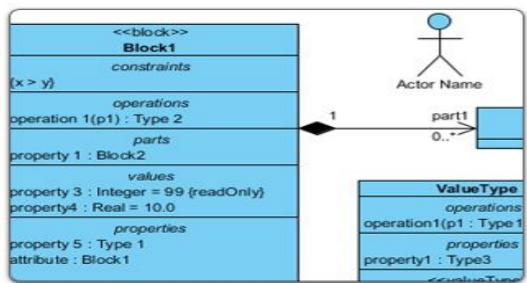
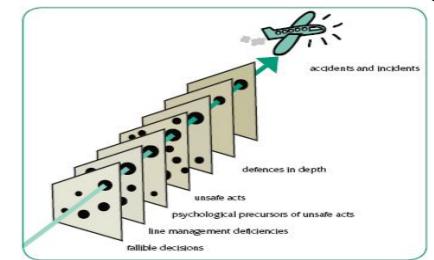
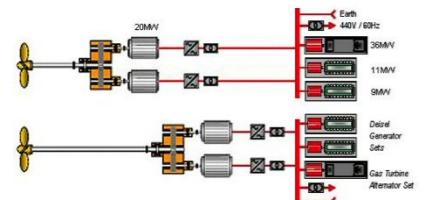
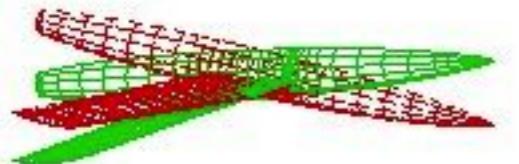
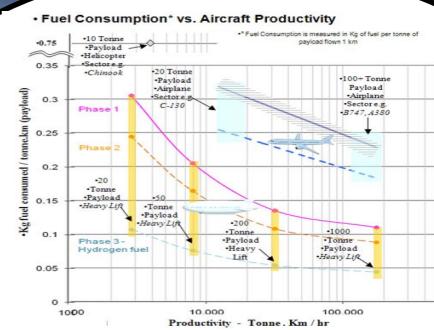
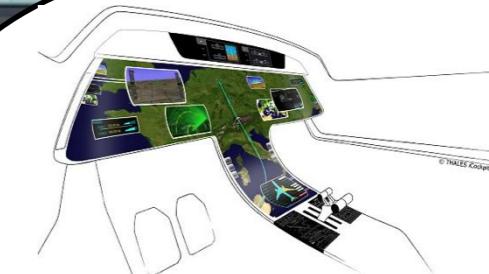
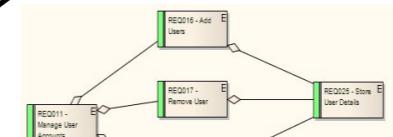
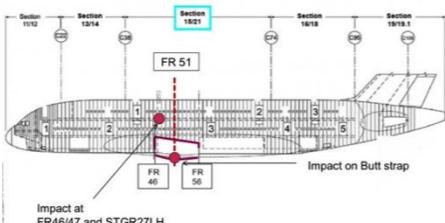
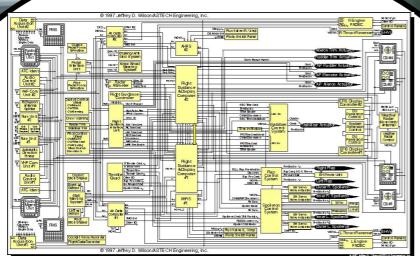




A central blue callout bubble contains the text: "Multiple concerns, stakeholders, tools and methods". Surrounding this central bubble are several white callout bubbles, each containing a different concern or stakeholder, accompanied by a small colored icon of a person:

- Mechanical Structure (blue shirt)
- Airlines (green shirt)
- Human -Machine Interaction (cyan shirt)
- Environmental Impact (grey suit)
- Safety Regulations (purple shirt)
- Authorities (yellow shirt)
- Navigation (blue shirt)
- Communications (blue suit)
- Propulsion System (orange shirt)
- Aerodynamics (blue shirt)
- Avionics (yellow shirt)
- Mechanical Structure (blue shirt)

Heterogeneous Modeling



Deep Variability in Software Systems

- Variability occurs on all concerns
- Variability showcases interdependencies
- Variability impacts soft/sys properties

=> Combinatorial explosion of the epistemic and ontological variability

Deep Variability refer to the interaction of all external “factors” modifying the behavior (including both functional and nonfunctional properties) of a software system

Parameters, Input Data Programming Style	e.g., random seed selection e.g., $x+(y+z)$ vs. $(x+y)+z$
Language	 python  Java  C++  F
Compiler & VM	 GCC  Erlang  JVM  Lua
Library	 NumPy  blas  jblas  PETSc
Platform	 Ubuntu  Windows  Mac OS  Android
Processor	 intel  AMD  RISC-V  ARM
Micro- architecture	Inner state of 

Software Modeling: Why Should I Care?

- To reflect:
 - abstract representation
 - separation of concerns
- To communicate:
 - graphical representation
 - documentation generation
- To automate development:
 - code generation
 - pattern application
 - migration
- To verify:
 - model validation and verification (e.g., simulation, model checking...)
 - model-based testing

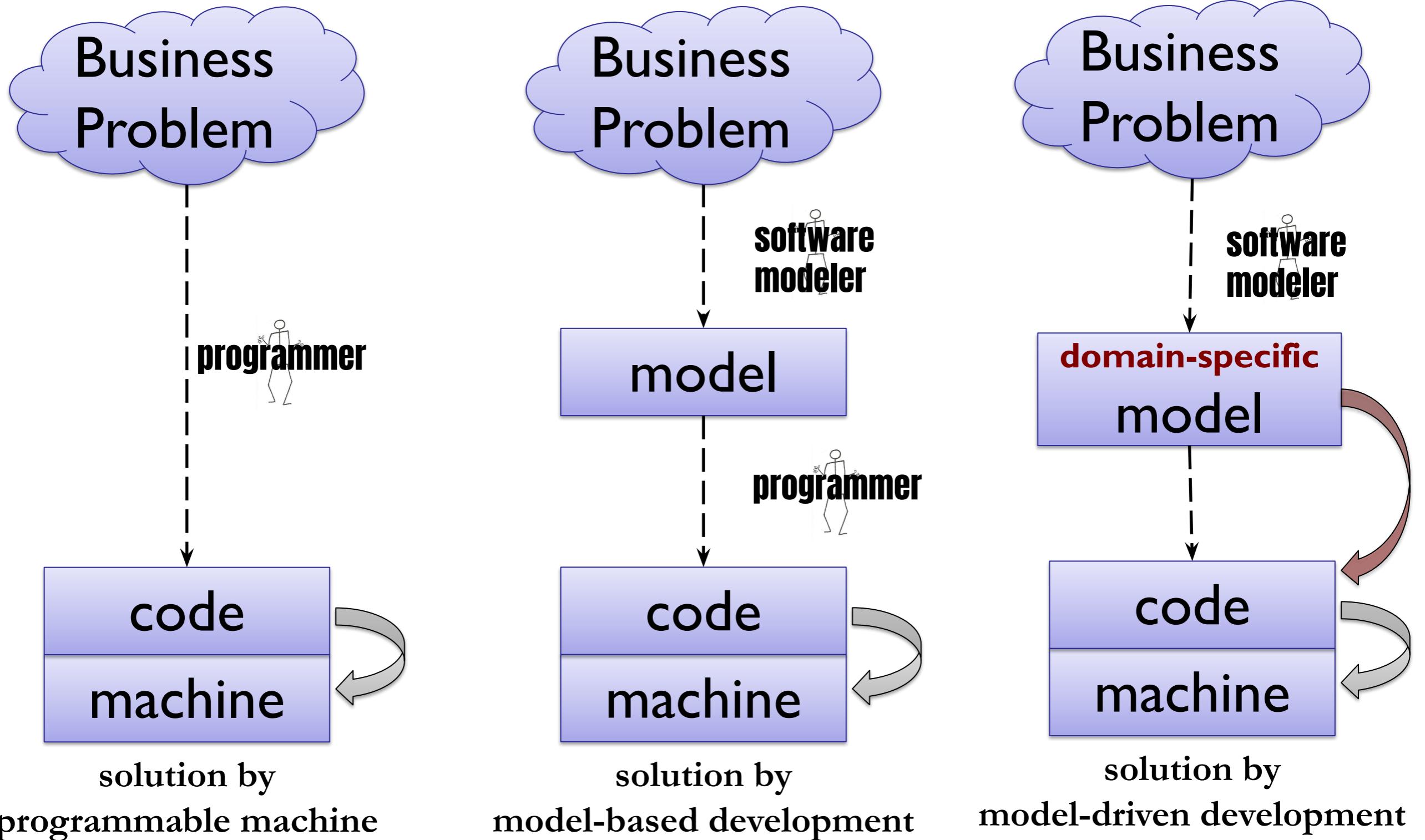
Model and Reality in Software

- **Sun Tse:** “*Do not take the map for the reality*”
- **William James:** “*The concept 'dog' does not bite*”
- **Magritte:**

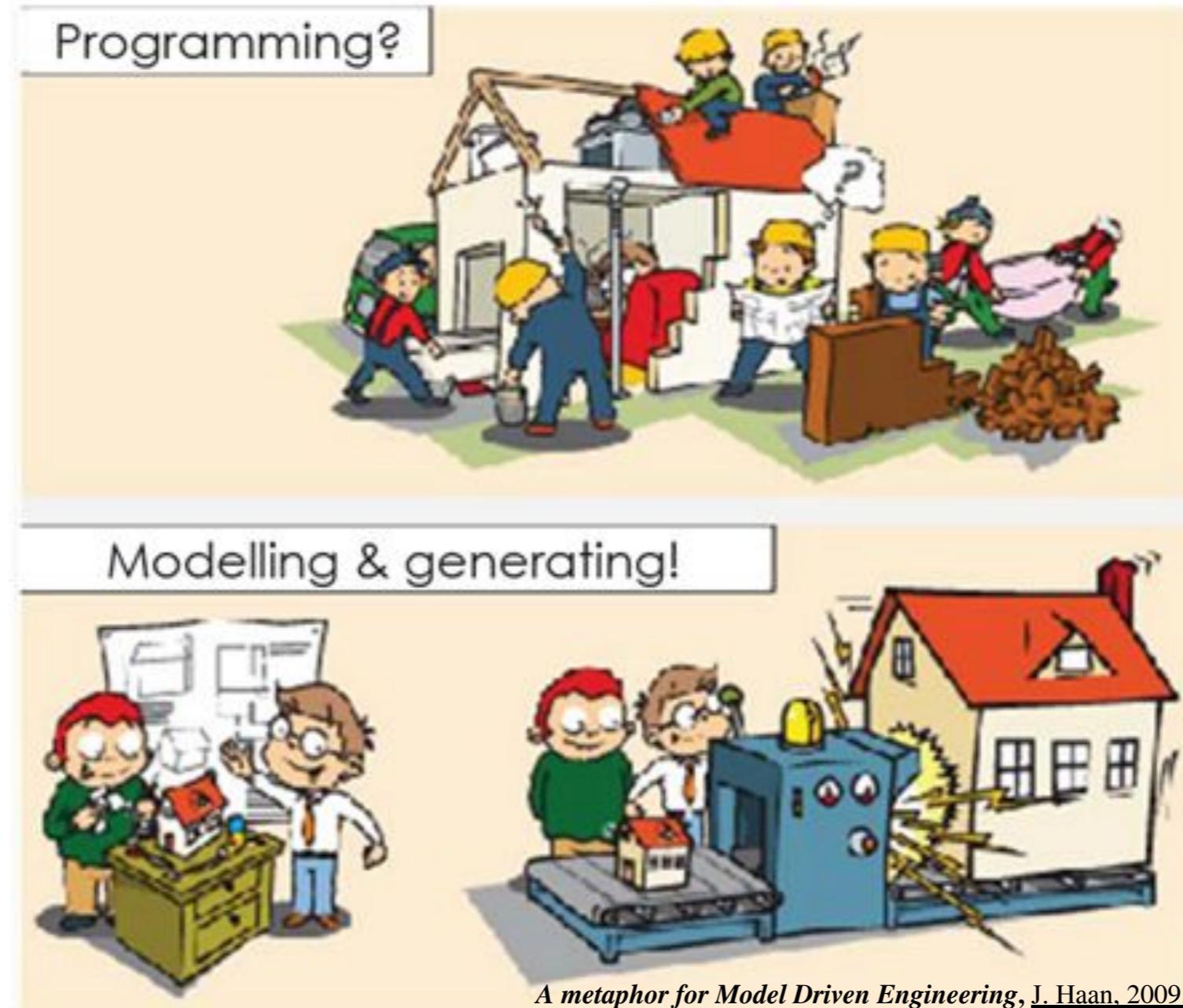


- Software Models: from contemplative to productive

Evolution in Software Modeling

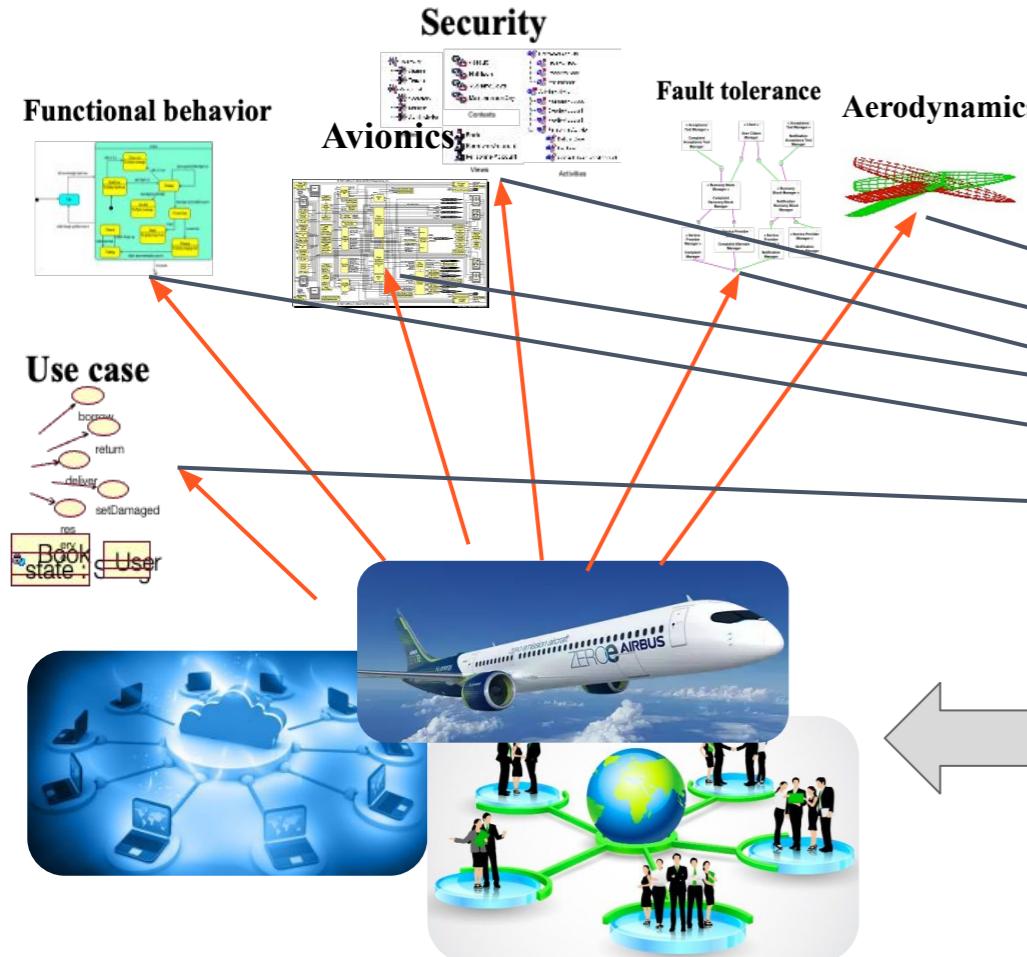


Towards Model-Driven Engineering (MDE)



Model-Driven Engineering (MDE)

*Separation of concerns,
abstraction*

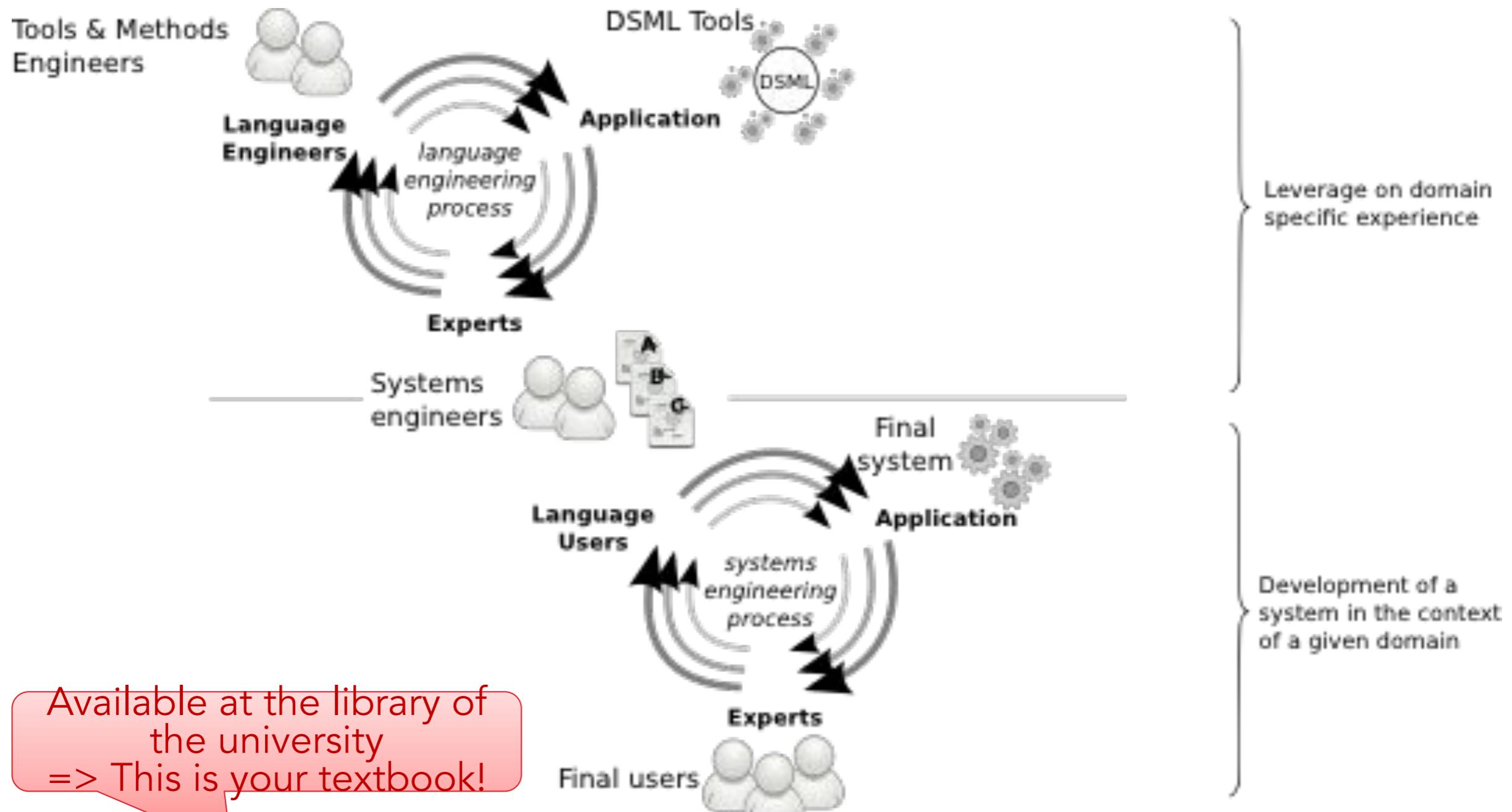


Unification power
of models, from
design to runtime

*Automated analysis,
simulation,
development...*

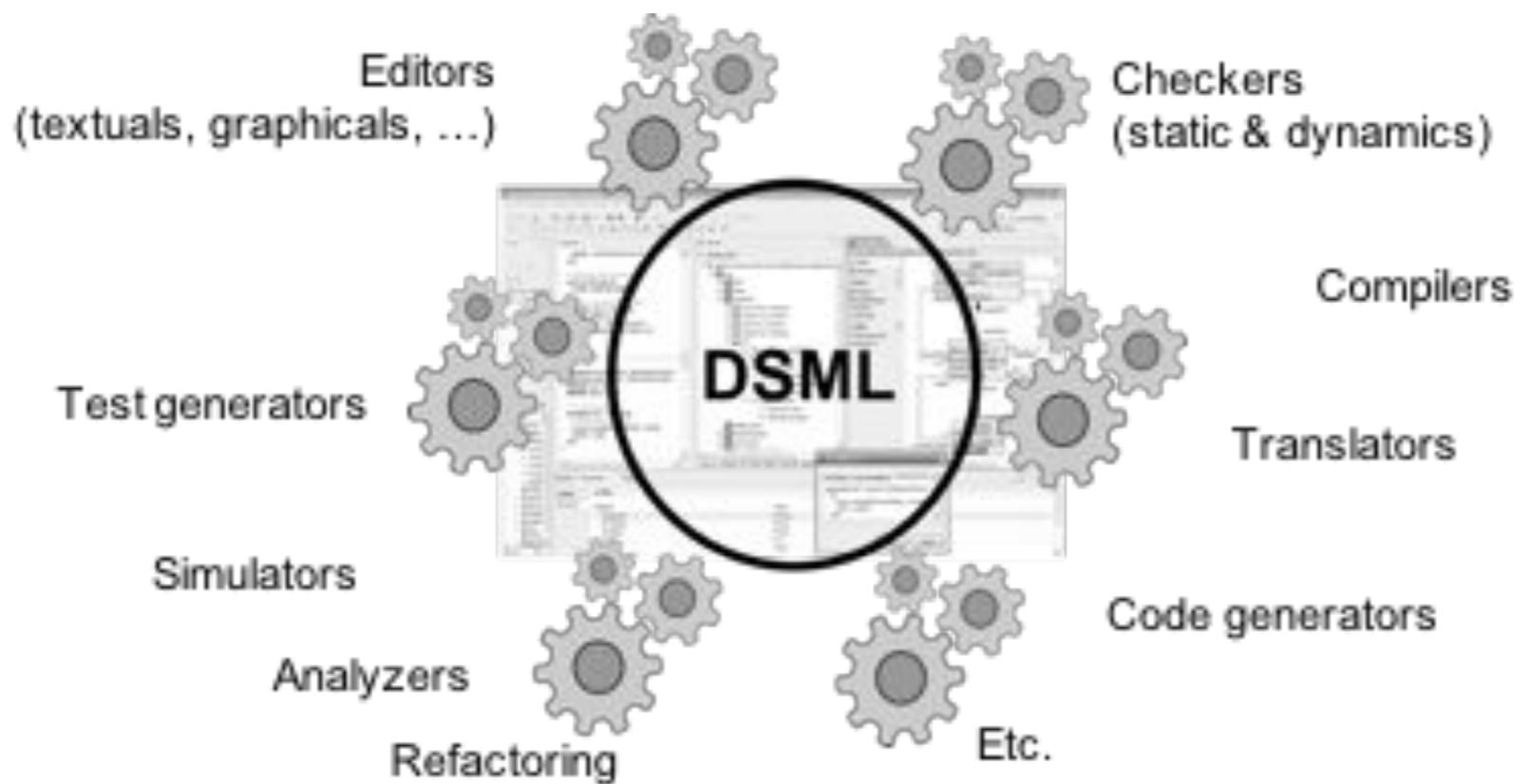
*complex software, systems
and organizations*

Language-Oriented Programming (LOP)



Engineering Modeling Languages: Turning Domain Knowledge into Tools, by Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek. Chapman and Hall/CRC, pp.398, 2016. Companion website: <http://mdebook.irisa.fr>

Language-Oriented Programming (LOP)



Engineering Modeling Languages: Turning Domain Knowledge into Tools, by Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek. Chapman and Hall/CRC, pp.398, 2016. Companion website: <http://mdebook.irisa.fr>

Domain-Specific Languages (DSLs)



- Targeted to a particular kind of problem, with dedicated notations (textual or graphical), support (editor, checkers, etc.)
- Promises: more « efficient » languages for resolving a set of specific problems in a domain

Domain-Specific Languages (DSLs)

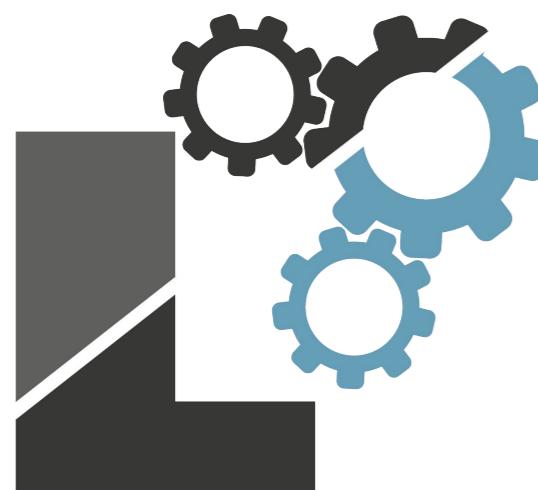
"Software Languages are Software Too"

J-M. Favre, D. Gasevic, R. Lämmel, and E. Pek. "Empirical language analysis in software linguistics," In Software Language Engineering, volume 6563 of LNCS, pages 316–326. Springer, 2011.

Software Language Engineering (SLE)

- Application of systematic, disciplined, and measurable approaches to the development, deployment, use, and maintenance of software (domain-specific) languages
- Supported by various kind of "language workbench"
 - Eclipse EMF, xText, Sirius, Melange, GEMOC, Papyrus
 - Eclipse Langium
 - Jetbrain's MPS
 - MS DSL Tools
 - Etc.
- Various shapes and ways to implement software languages
 - External, internal or embedded DSLs, Profile, etc.
 - Grammar, metamodel, ontology, etc.
- More and more literature, a dedicated Intl. conference (ACM SLE, cf. <http://www.sleconf.org>)...

GEMOC Studio



*Language
Workbench*

*Design and
integrate your
executable
DSMLs*



<http://gemoc.org/studio>

also

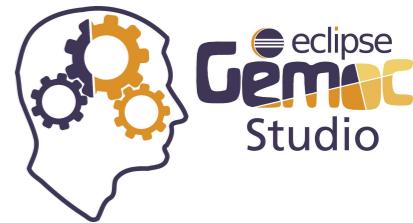
<http://eclipse.org/gemoc>



*Modeling
Workbench*

*Edit, simulate and
animate your
heterogeneous
models*

Arduino Designer



The screenshot displays the Eclipse GEMOC Studio interface for Arduino modeling. The top navigation bar shows the title "runtime-arduinoDebug - platform:/resource/org.gemoc.sample.arduino.sequential.blinker/model.aird/Sketch - Gemoc Studio". The left sidebar contains a "Debug Configurations" section with a list of items including "model.arduino", "Arduino Simulator", "Java Applet", and "Remote Java Application". Below it is a "Debug" view showing a tree structure for a "blinker [ALE Launcher]" project, with "Gemoc debug target" and "Model debugging" expanded, and a specific step in the execution stack highlighted: "(VariableDeclaration) org.gemoc.sequential.model.arduino.impl.VariableDeclarationImpl@9b02cf4 -> execute()".

The central workspace is divided into two main areas: "Hardware" on the left and "Sketch" on the right. The "Hardware" view shows a virtual representation of an Arduino Board with three LEDs labeled "RED LED", "BLUE LED", and "WHITE LED" connected to digital pins 0, 1, and 2 respectively. The "Sketch" view shows a statechart-like diagram titled "newSketch" with a "Repeat 5" loop. Inside the loop, three parallel regions represent the LEDs: "BLUE LED : (i%2)", "RED LED : ((i/2)%2)", and "WHITE LED : ((i/4)%2)". Each region has its own logic, including assignments for "i" and "int 2". A "Set i = (i+1)" action is shown at the bottom of the loop. The "Console" view at the bottom left shows initialization logs, and the "Variables" view on the bottom right lists variables with their current values:

Name	Value
i (org.gemoc.sequential.model.arduino.impl.RepeatImpl@4e523b1 (iteration: 5) :Repeat)	0
level (Arduino Board.0 :DigitalPin)	0
level (Arduino Board.1 :DigitalPin)	0
level (Arduino Board.2 :DigitalPin)	0
value (newSketch.i :IntegerVariable)	0

<https://github.com/gemoc/arduinomodeling>

Transformation Lg Debugger



Screenshot of the Eclipse Gemoc Studio interface showing the Debug perspective for a Minitl model named "sample.minitl".

The Debug view shows a tree of execution contexts:

- run_AtoB [Gemoc Sequential eXecutable Model]
- run_AtoB [Gemoc Sequential eXecutable Model]
- Gemoc debug target
 - Model debugging
 - (ObjectTemplate) simpleAtoB.AToB.b -> construct()
 - (Rule) simpleAtoB.AToB -> apply()
 - (Transformation) simpleAtoB -> execute()
- Global context : Transformation

The Variables view displays the current state of variables:

Name	Value
currentObject (simpleAtoB.AToB.a :ObjectTemplate)	org.eclipse.emf.ecore.impl.DynamicEObjectImpl@7
currentObject (simpleAtoB.AToB.b :ObjectTemplate)	null
inputModel (simpleAtoB :Transformation)	[org.eclipse.emf.ecore.impl.DynamicEObjectImpl@2
inputModelURI (simpleAtoB :Transformation)	platform:/resource/minitl-models/modelA.xmi
outputFilePath (simpleAtoB :Transformation)	/home/ebousse/dev/runtime-test-minitl-modeling/r
outputModel (simpleAtoB :Transformation)	[org.eclipse.emf.ecore.impl.DynamicEObjectImpl@6

The code editor shows the Minitl transformation code:

```
transformation simpleAtoB {
    // Ecore metamodel called "input"
    inputMetamodel metamodelA

    // Ecore metamodel called "output"
    outputMetamodel metamodelB

    // Rule to translate each "A" element into "B"
    rule AToB {
        // Input template
        from a : metamodelA.A
        // Output template
    }
}
```

The Console view shows the execution logs:

```
Modeling workbench console
About to initialize and run the GEMOC Execution Engine...
Initialization done, starting engine...
Execution finished.
About to initialize and run the GEMOC Execution Engine...
Initialization done, starting engine...
```

A callout bubble in the bottom right corner contains the text:

TETRABox
<http://modeltransformation.net/tetrabox/>
Wimmer, Bousse et al.

<https://github.com/tetrabox/minitl>

Farming System Modeling



Model Explorer Logical Steps

cultures.activities

```

culture corn f
activity LABOUR from 1 jan to 28 feb
using 1 Tractor and 1 People

activity SEMIS from 15 mar to 15 apr [
    after LABOUR & no rain since 3 days && tempe
] using 1 Tractor and 2 People

activity IRRIGATION weekly from 15 jun to 15 aug
after SEMIS
] using 1 Tractor and 1 People

activity FERTILISATION from 15 mar to 15 jun [
    after SEMIS is done since 30 days &&
    no rain since 1 days
] using 1 Tractor and 1 People

activity RECOLTE from 1 sept to 30 sept [
    grain is "mature" &&
    after SEMIS
] using 1 Tractor and 2 People
}

```

*exploitation description

surfaces ratios
Watered Foddered

solver search limit :8 secs

Extra Water needed : 161800m³

Tractor People wheat corn sorgho

dedicated to

Palettes

Daily Crop Ewe Resource Surface Assign Surface Assign Culture

*Schedule

NW of Exploitation 4 fields

corn LABOUR scheduled on 13/jan

corn SEMIS scheduled on 31/mar

corn IRRIGATION scheduled on 4/aug

corn FERTILISATION scheduled on 5/may

corn RECOLTE scheduled on 1/sept

2 fields

corn LABOUR scheduled on 1/jan

corn SEMIS scheduled on 15/mar

corn IRRIGATION scheduled on 15/jun

corn FERTILISATION scheduled on 27/may

corn RECOLTE scheduled on 21/sept

*Hydro Analysis

John's Exp... Surface...

	Extra Water	Rain	Hyd	Biomass	LAI
31 mar	0.0	0.0	57.0		
1 apr	0.0	0.0	57.0	0.00761125...	0.000
2 apr	0.0	0.0	57.5	0.01527833...	0.000
3 apr	0.0	0.0	57.5	0.01730399...	0.000
4 apr	40.0	0.0	60.5	0.02231124...	0.000
5 apr	0.0	0.0	21.5	0.02885451...	0.000
6 apr	0.0	11.0	22.0	0.03494558...	0.000
7 apr	0.0	5.0	16.5	0.03872302...	0.000
8 apr	0.0	0.0	11.5	0.04052190...	0.000
9 apr	0.0	0.0	11.5	0.04548258...	0.000
10 apr	0.0	11.5	11.5	0.04743018...	0.000
11 apr	0.0	0.5	0.0	0.05144848...	0.000
12 apr	0.0	2.5	-0.5	0.05425001...	0.000
13 apr	0.0	0.5	-3.0	0.05683383...	0.000

*Climate Data

	Rain (mm)	Temperature (°C)	Ray (Joules/cm²)
apr 7	5.0	10.4	626.0
apr 8	0.0	10.4	290.0
apr 9	0.0	11.0	775.0
apr 10	11.5	11.4	263.0
apr 11	0.5	9.9	700.0
apr 12	2.5	10.7	450.0
apr 13	0.5	9.7	815.0

analysis.scientific

Resource Set

- platform:/resource/MyExploitation/analysis.scientific
 - Exploitation Analysis 60.0
 - Biomass Model 1.85
 - Biomass Model 1.85**
 - Biomass Model 1.85

Properties

Property	Value
A	0.0065
B	0.00205
Culture	Culture wheat
Eb	1.85
Eimax	0.94
K	0.5
Lmax	6.5

<https://github.com/gemoc/farmingmodeling>

UML Activity Diagram



Debug - platform:/resource/org.modelexecution.operationalsemantics.ad.samplemodels/model/test2.aird/test2 Activity Diagram - Gemoc Studio

File Edit Diagram Navigate Search Project Run Window Help

Quick Access Debug xDSML

Variables

Name	Value
heldTokens (ActivityFinalNode_finalNode2 :ActivityFinalNode)	
heldTokens (ForkNode_forkNode1 :ForkNode)	
heldTokens (InitialNode_initialNode2 :InitialNode)	[activitydiagram.impl.ControlTokenImpl]
heldTokens (JoinNode_joinNode1 :JoinNode)	

Breakpoints

Opaque Action action2

No details to display for the current selection.

Console *test2 Activity Diagram

Properties

Opaque Action action2

Property	Value
Activity	Activity test2
Incoming	Control Flow edge4
Name	action2
Outgoing	Control Flow edge6
Running	true

Multidimensional Timeline

All execution states (11)

Timeline for dynamic information

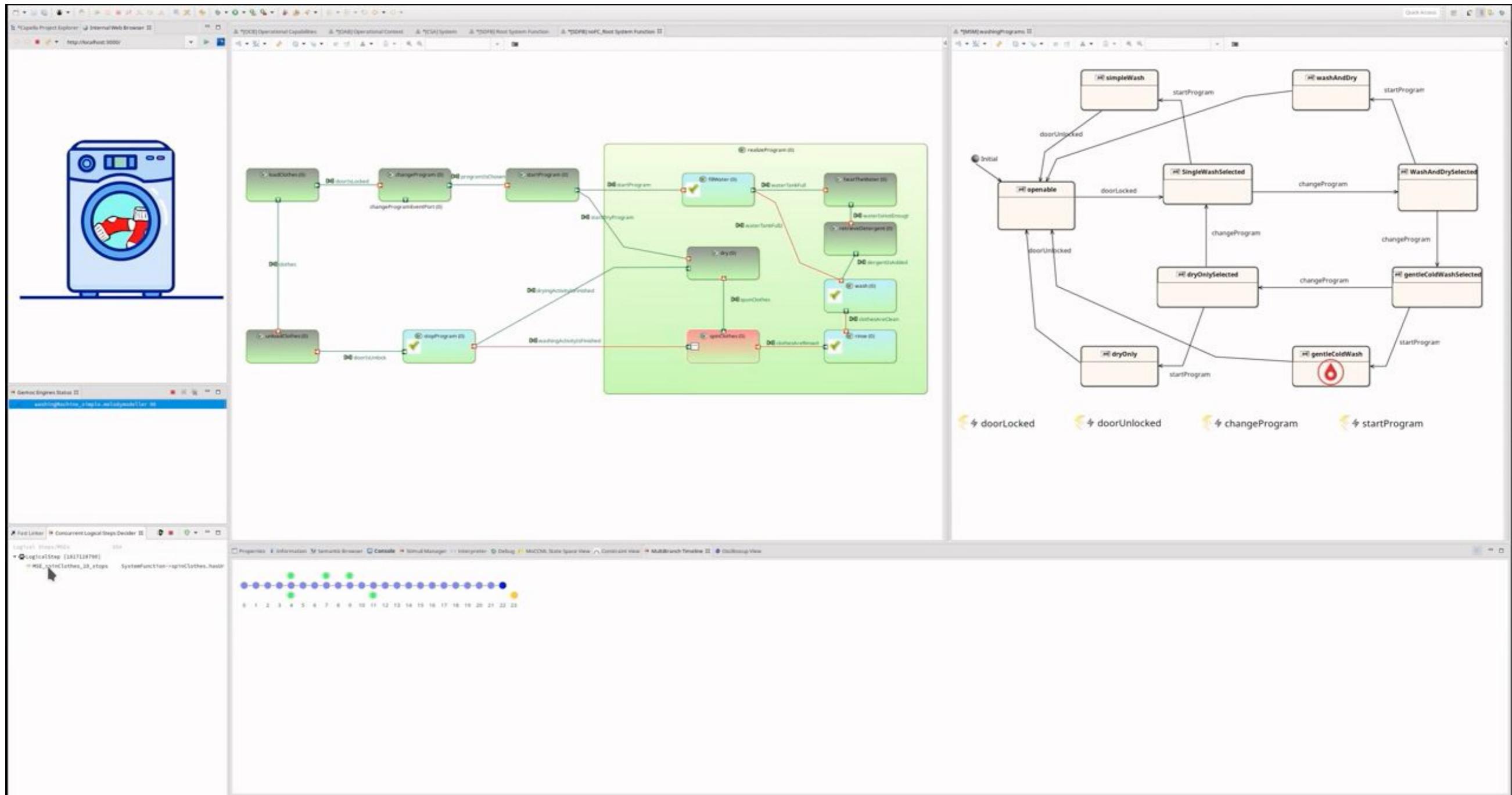
- trace (test2 :Activity)
- heldTokens (test2.initialNode2 :InitialNode)
- heldTokens (test2.forkNode1 :ForkNode)
- heldTokens (test2.action2 :OpaqueAction)
- heldTokens (test2.action3 :OpaqueAction)
- heldTokens (test2.joinNode1 :JoinNode)
- heldTokens (test2.finalNode2 :ActivityFinalNode)

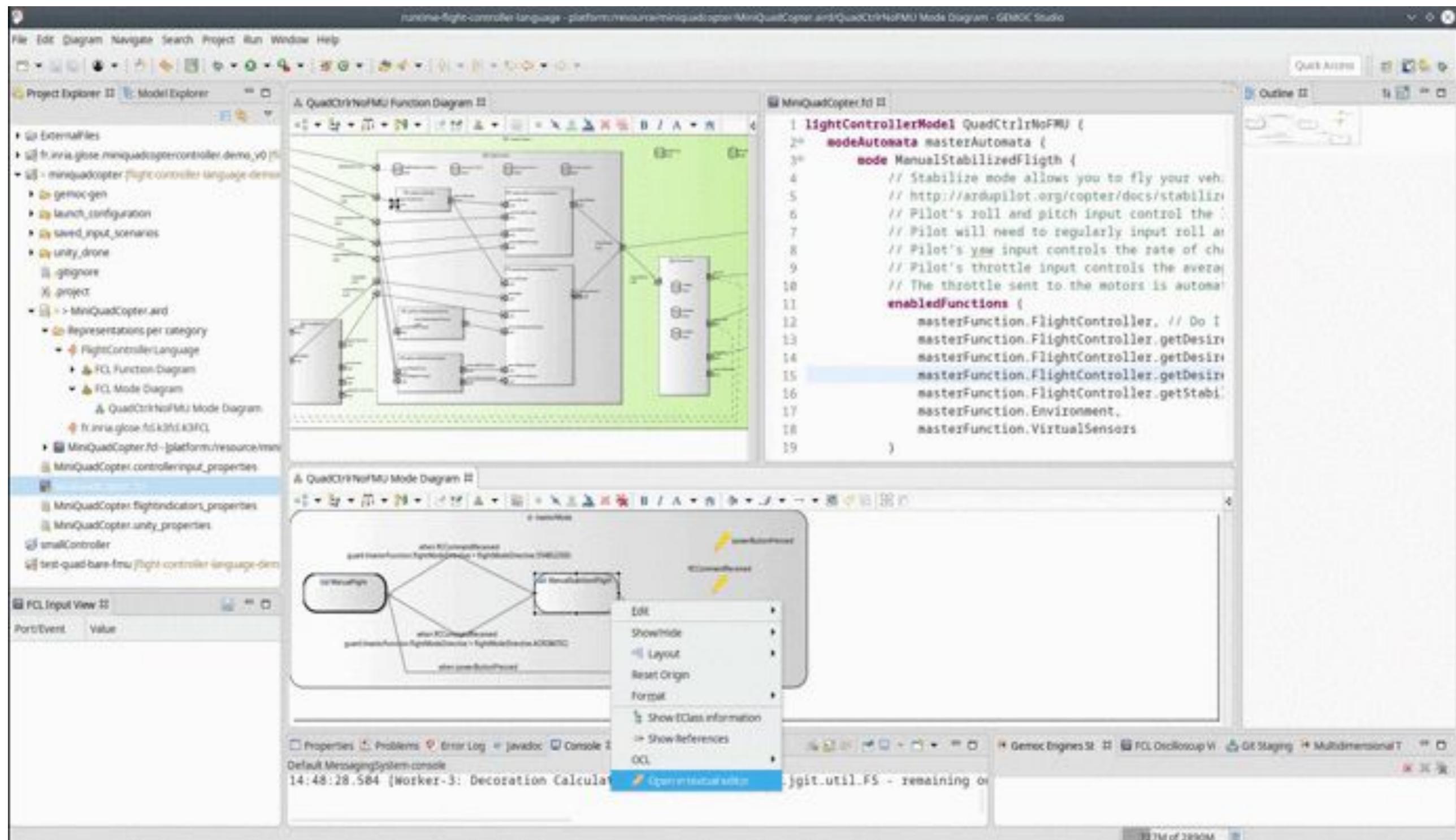
Gemoc Engines Status

test2.ac 8

<https://github.com/gemoc/activitydiagram>

xCapella

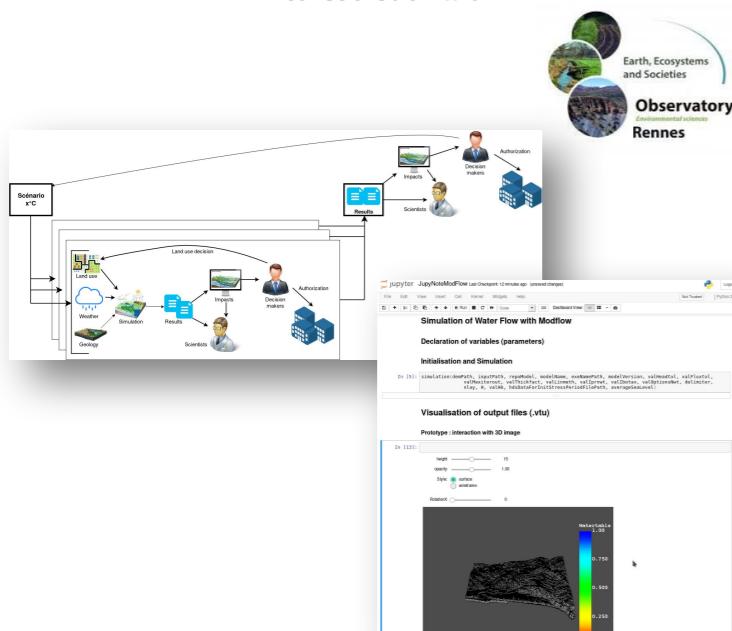




Crosscutting Challenges for Various Domains

Towards Virtual Labs From Modeling Environment to Digital Twin

in collaboration with



Tradeoff Analysis for
Water Flood Prediction

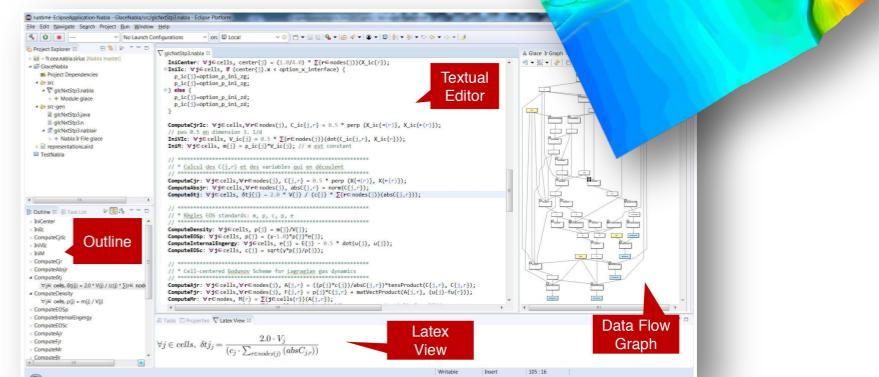


Design Space Exploration
for Cyber-Physical Systems

in collaboration with

SAFRAN

in collaboration with



High-Performance Computing
for Numerical Analysis

Content of the course

- **Metamodeling**
- **Hack your own Domain-Specific Languages**
 - **Domain model (EMF Ecore)**
 - **Textual syntaxes: Xtext/Langium**
 - **Compiler and interpreter (Xtend/TypeScript)**
- **Companion webpage:**
<https://stephaniechallita.github.io/ase/>

Flipped classroom

- **The group prepares and presents parts of the lecture**
 - **Based on slides and textual content**
 - **During next session, students present and discuss the lecture content**
- **Finite State Machine (FSM) companion project:**
 - **Apply together concepts seen during lectures**
 - **Basic project to built-up the evaluated project**

Project: RobotML

- **Over 12 labs sessions**
- **Subject: a DSL to program robot's movements**
- **In pair**
- **Evaluation**
 - **Metamodel / 5**
 - **Concret syntaxe / 5**
 - **Semantics (compiler & interpreter) / 10**

You will learn how to

- **Automatically translate** abstract design models to executable code, test cases and documentation
- **Automatically manipulate** your model/code to analyze, transform, refactor..
- **Build or customize your own abstractions languages and development environments**, or even **software**, to build complex software-intensive systems
- Eventually **limit the accidental complexity** of industrial developments

Additionally, you will also

- **Demystify** language formalisms, paradigms and principles
- **Have a deeper insight** on some of them
- **Manage the industrial complexity** of developments and associated toolchains