

# MODEL-DRIVEN *(SOFTWARE)* ENGINEERING

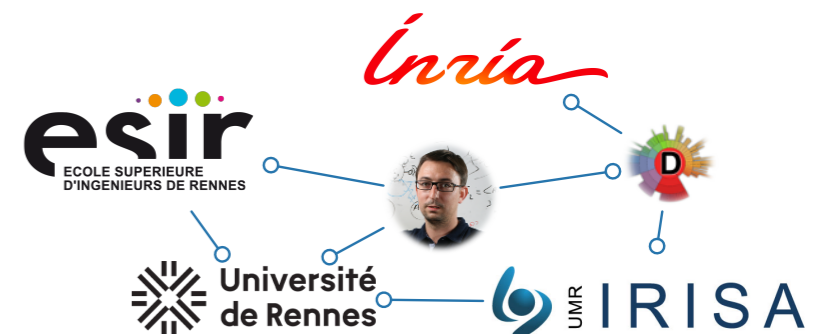
*HACK YOUR OWN LANGUAGE!*

---

ESIR3 ASE, 2023-2024

**BENOIT COMBEMALE**  
FULL PROFESSOR, UNIVERSITY OF RENNES, FRANCE

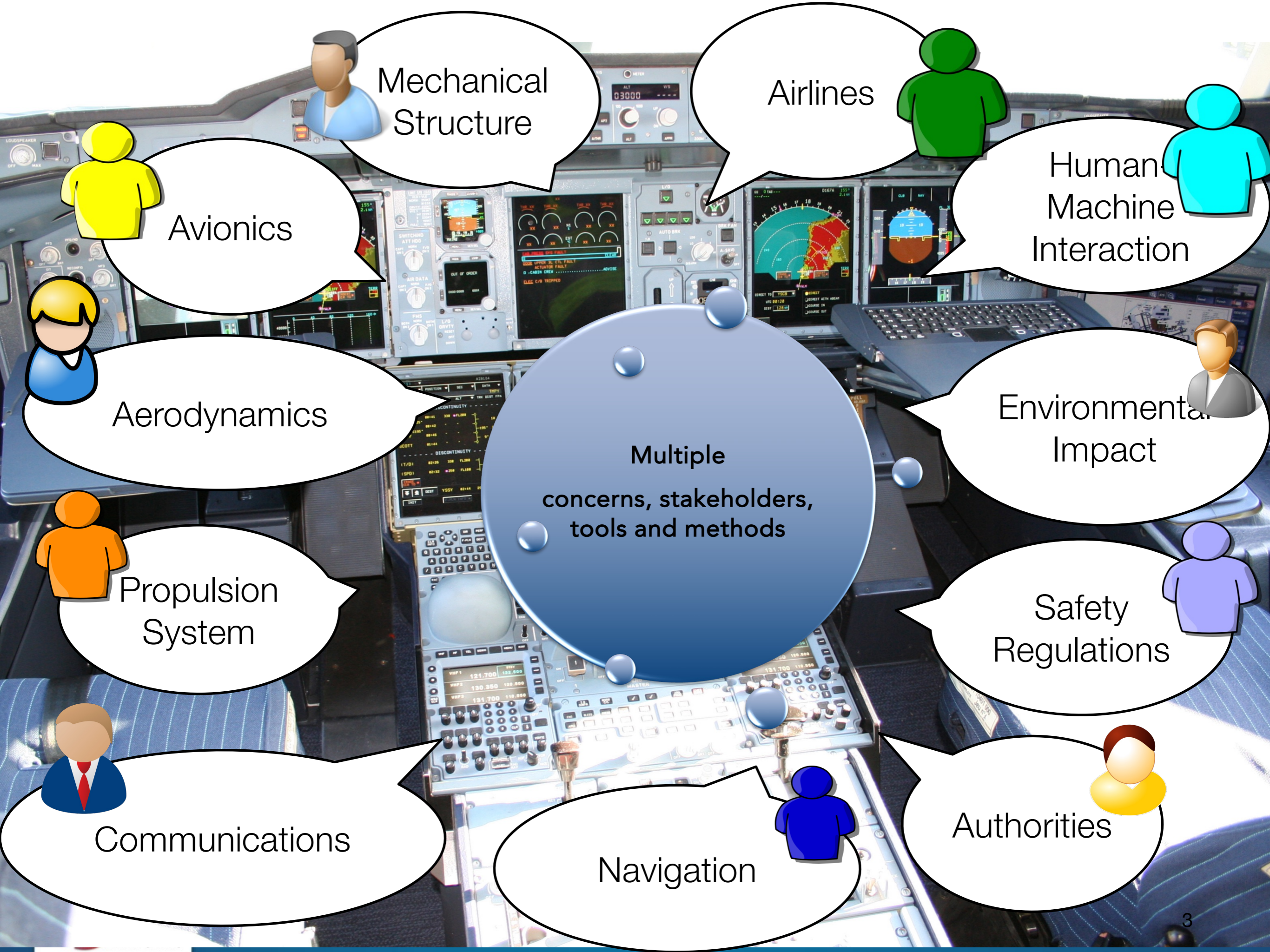
[HTTP://COMBEMALE.FR](http://combemale.fr)  
[BENOIT.COMBEMALE@IRISA.FR](mailto:benoit.combemale@irisa.fr)  
[@BCOMBEMALE](https://twitter.com/bcombemale)



# Complex Software-Intensive Systems



- ▶ Multi-engineering approach
- ▶ Domain-specific modeling
- ▶ High variability and customization
- ▶ Software as integration layer
- ▶ Openness and dynamicity



Mechanical Structure

Airlines

Human Machine Interaction

Avionics

Environmental Impact

Aerodynamics

Multiple concerns, stakeholders, tools and methods

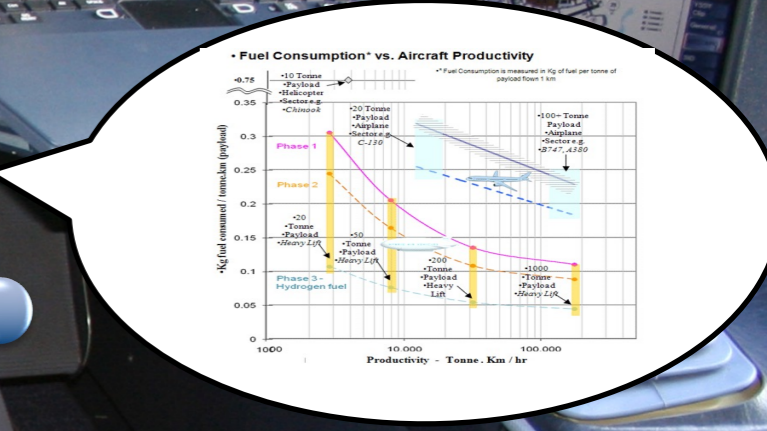
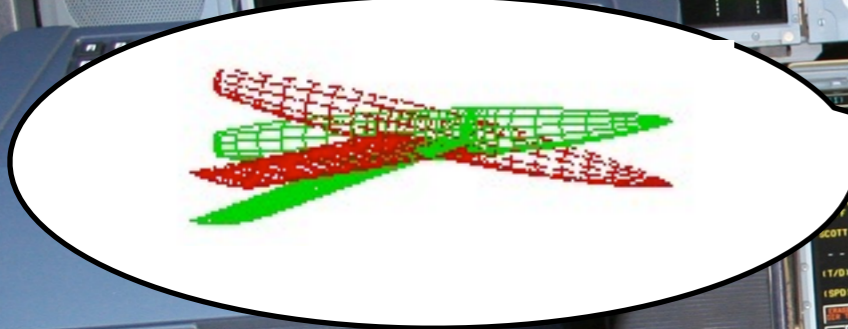
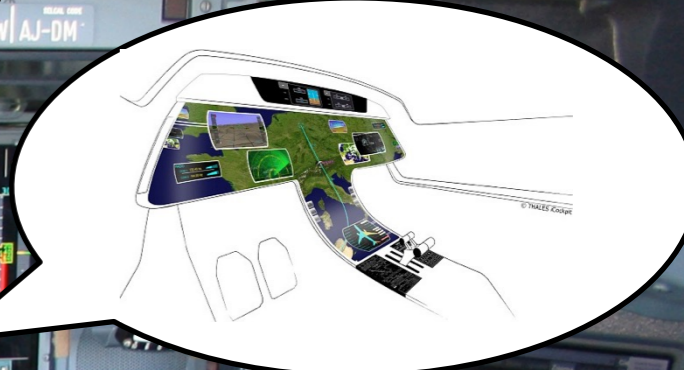
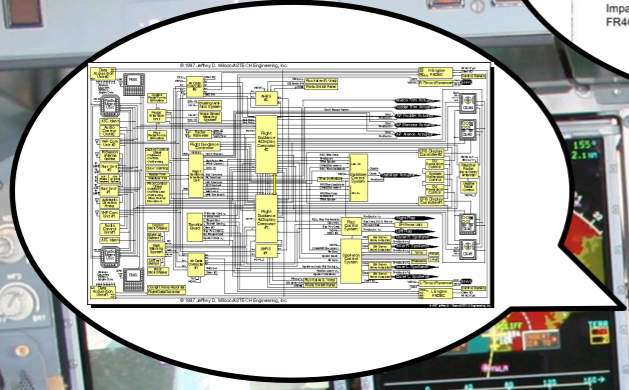
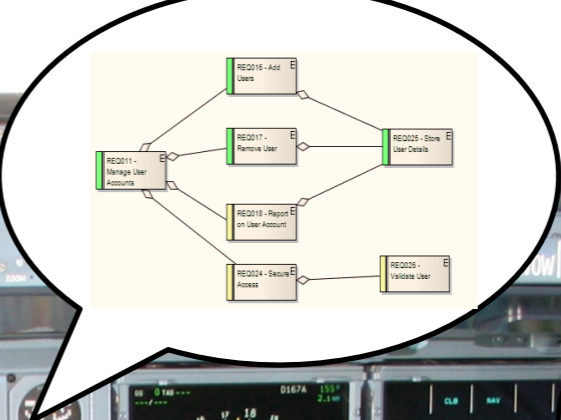
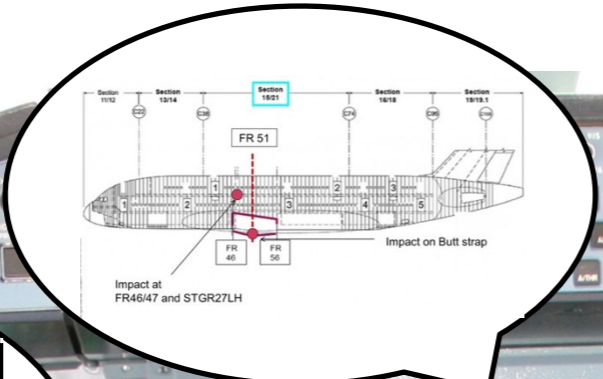
Safety Regulations

Propulsion System

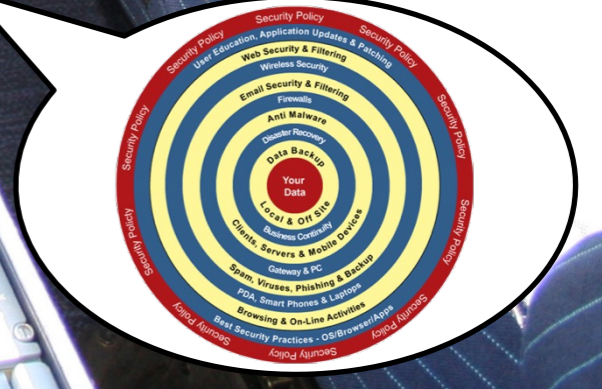
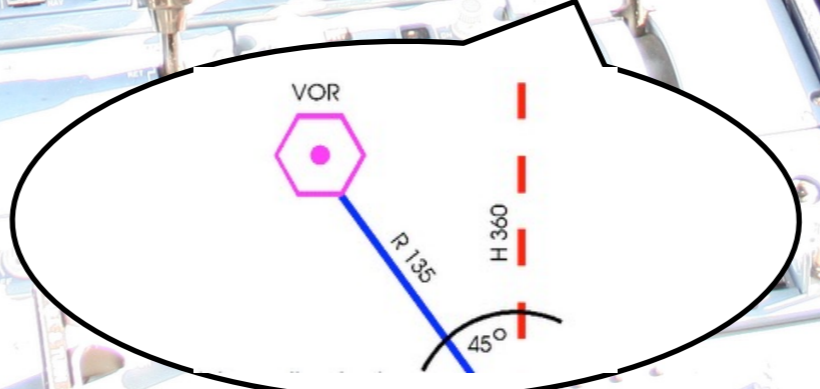
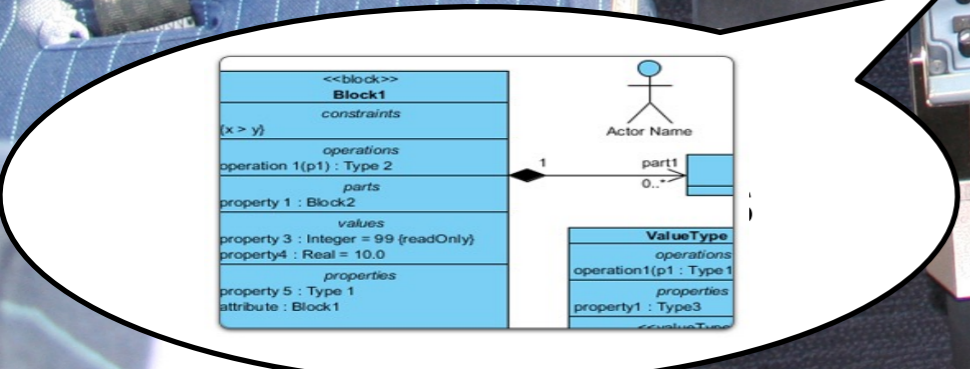
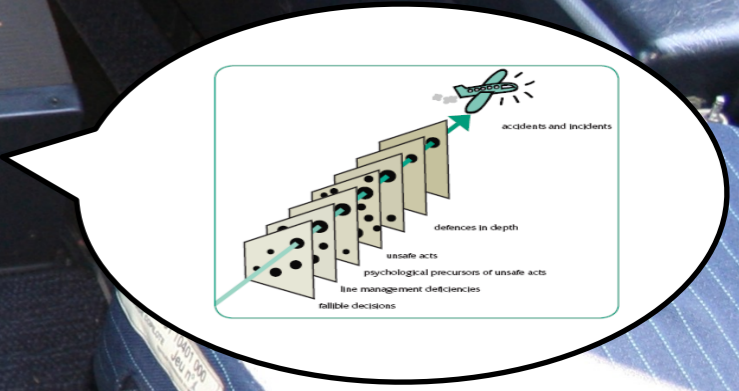
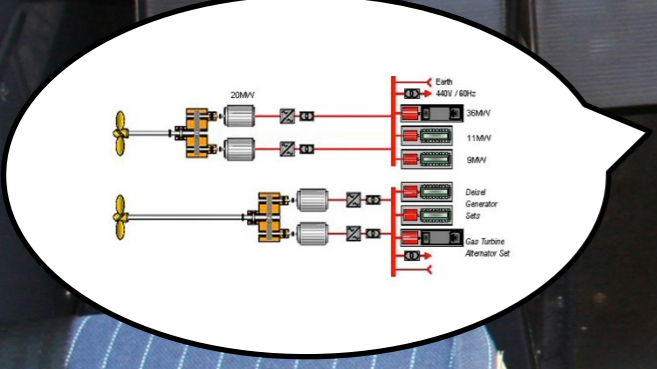
Authorities

Communications

Navigation



# Heterogeneous Modeling



# Software Modeling: Why Should I Care?

---

- ▶ Pour réfléchir :
  - ▶ représentation abstraite
  - ▶ séparation des préoccupations
- ▶ Pour communiquer :
  - ▶ représentation graphique
  - ▶ génération de documentation
- ▶ Pour automatiser le développement :
  - ▶ génération de code
  - ▶ application de patrons
  - ▶ migration
- ▶ Pour vérifier :
  - ▶ validation et vérification de modèles (e.g., simulation, model-checking...)
  - ▶ model-based testing

# Software Modeling: Why Should I Care?

---

- ▶ Pour réfléchir :
  - ▶ représentation abstraite
  - ▶ séparation des préoccupations
- ▶ Pour communiquer :
  - ▶ représentation graphique
  - ▶ génération de documentation
- ▶ Pour automatiser le développement :
  - ▶ génération de code
  - ▶ application de patrons
  - ▶ migration
- ▶ Pour vérifier :
  - ▶ validation et vérification de modèles (e.g., simulation, model-checking...)
  - ▶ model-based testing

# Model and Reality in Software

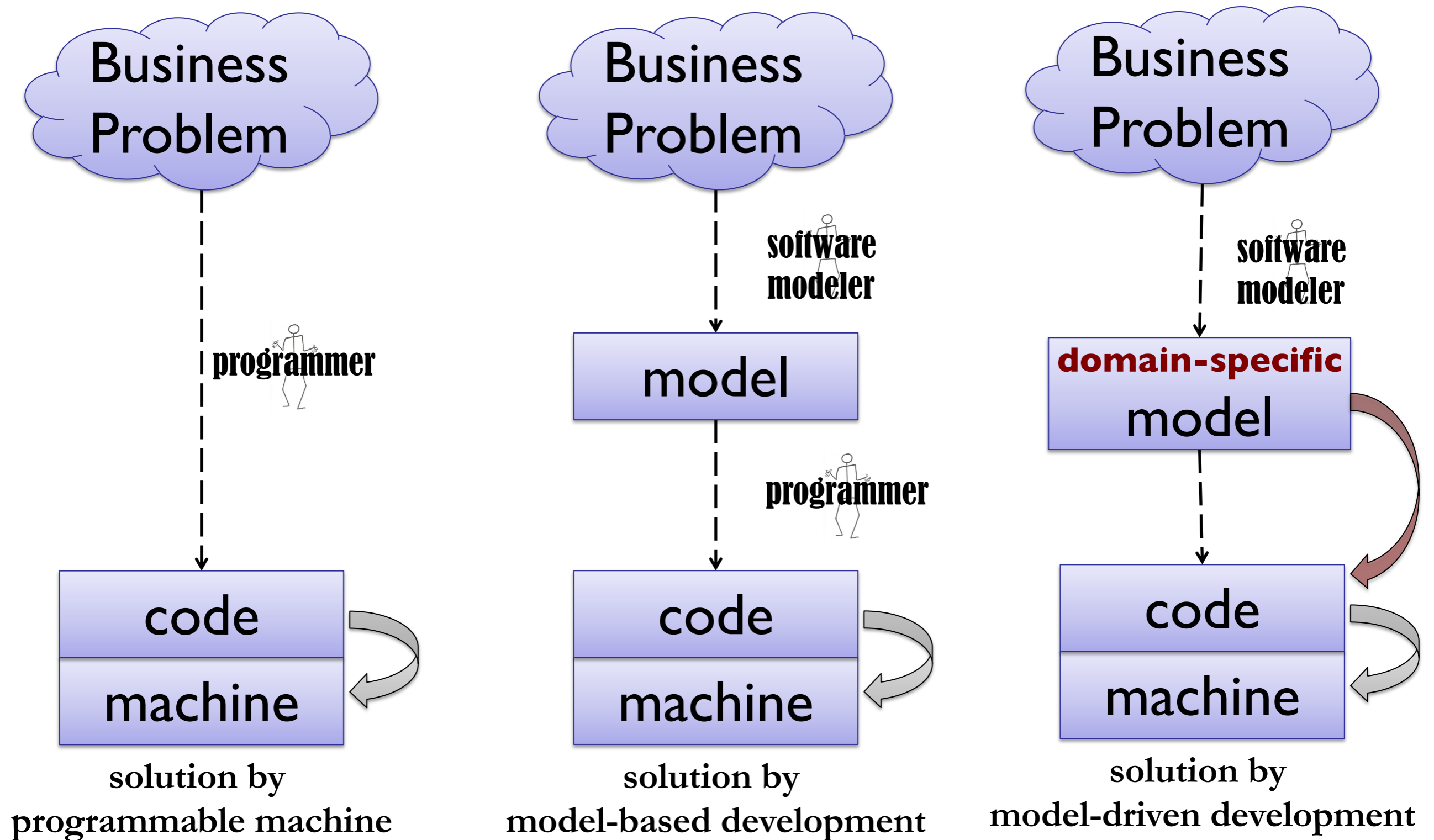
---

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



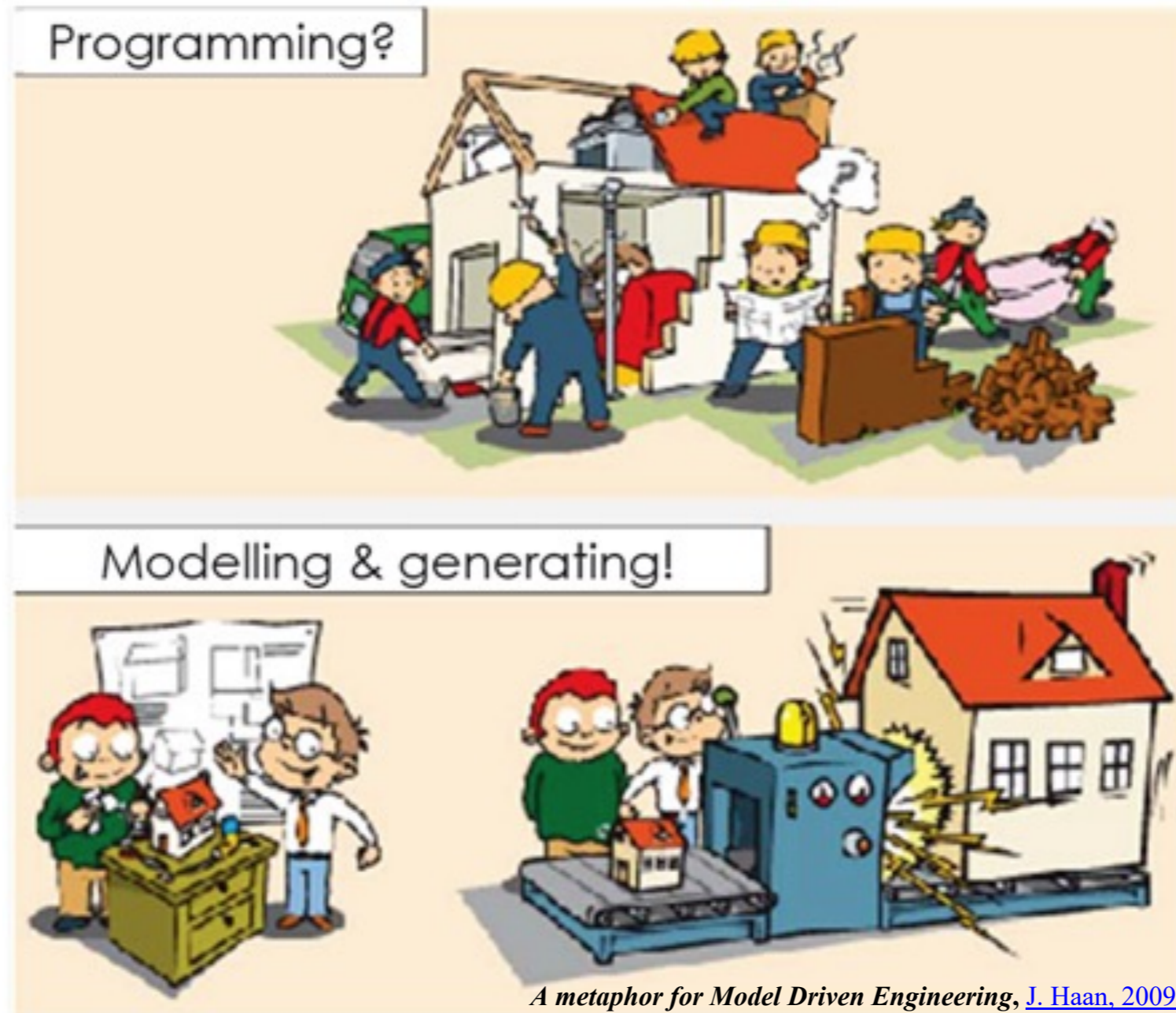
- Software Models: from contemplative to productive

# Evolution in Software Modeling

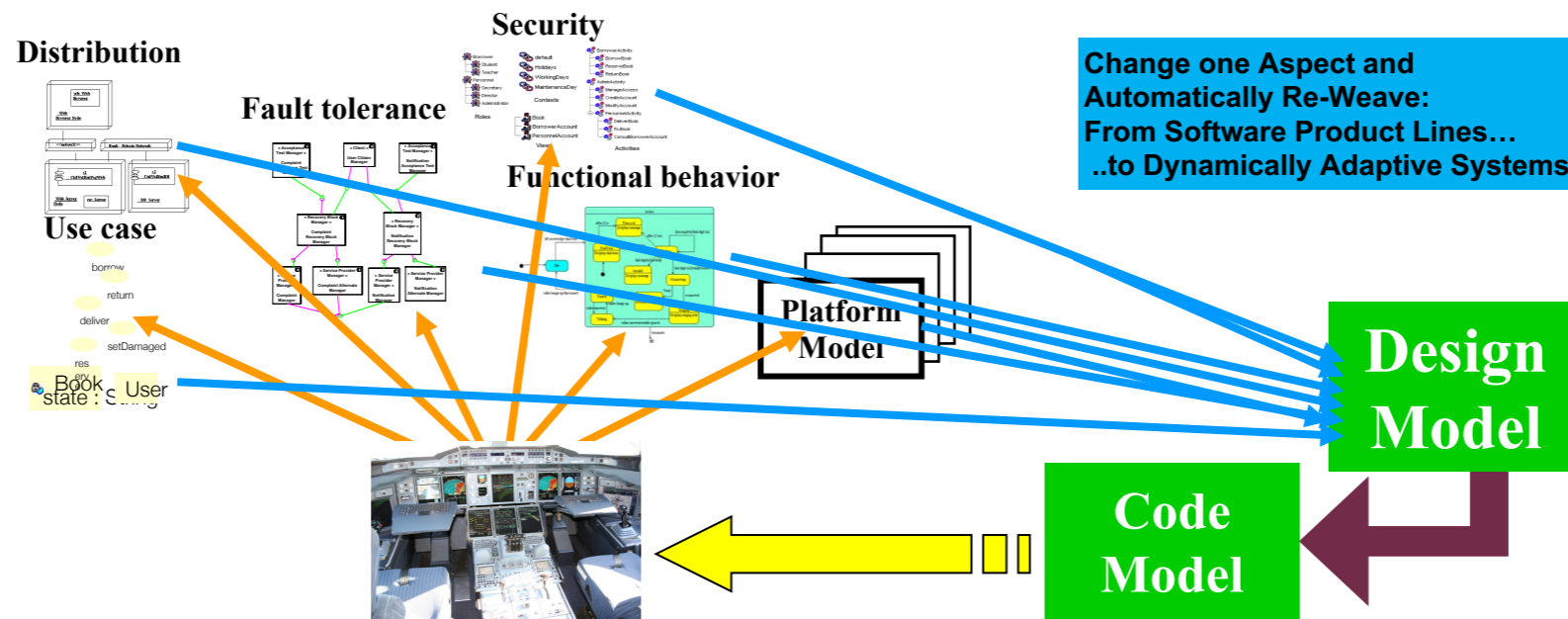




# Towards Model-Driven Engineering (MDE)



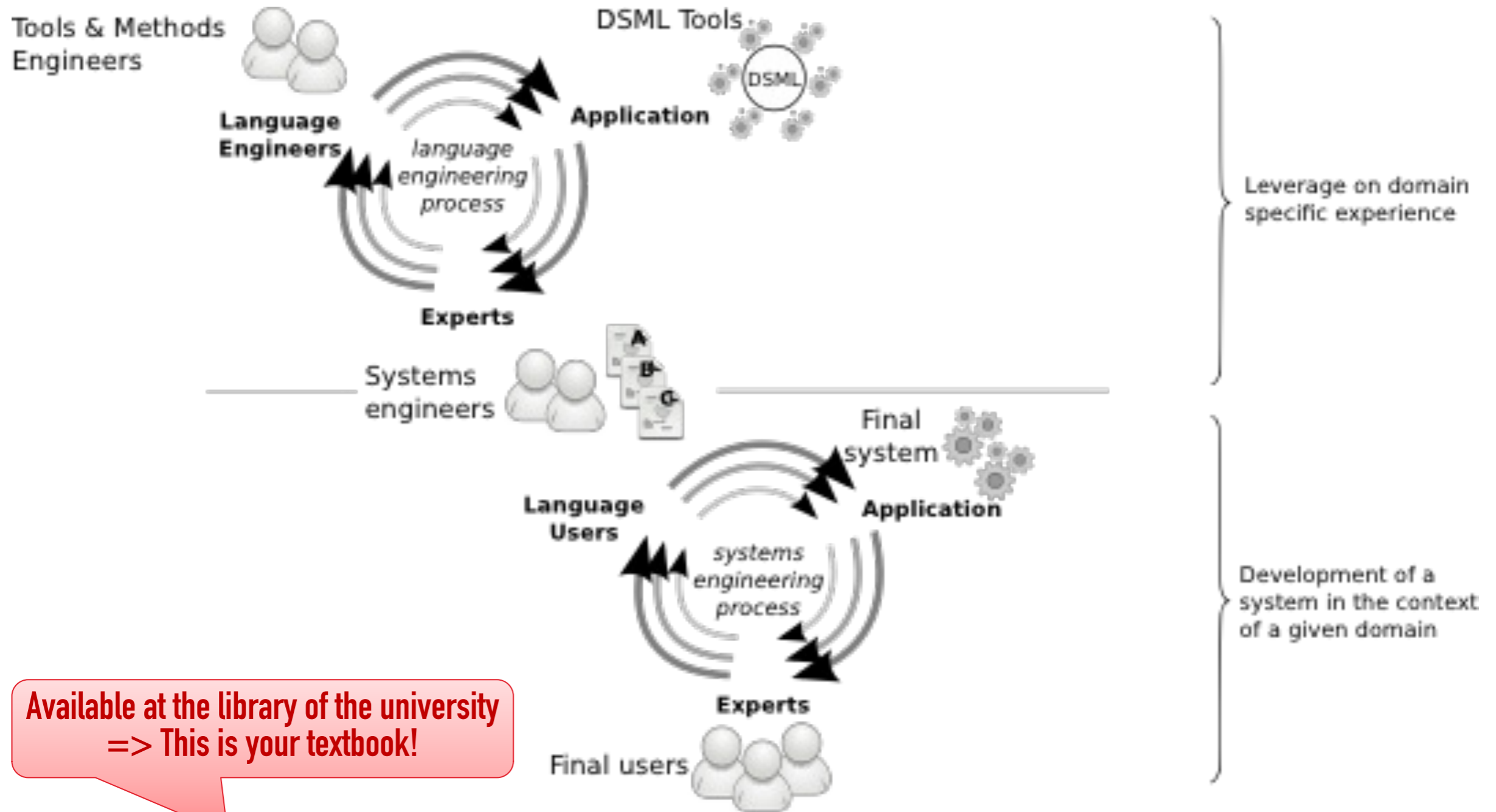
# Model-Driven Engineering (MDE)



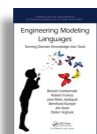
*"Perhaps surprisingly, the majority of MDE examples in our study followed domain-specific modeling paradigms"*

J. Whittle, J. Hutchinson, and M. Rouncefield, "The State of Practice in Model-Driven Engineering," IEEE Software, vol. 31, no. 3, 2014, pp. 79–85.

# Language-Oriented Programming (LOP)

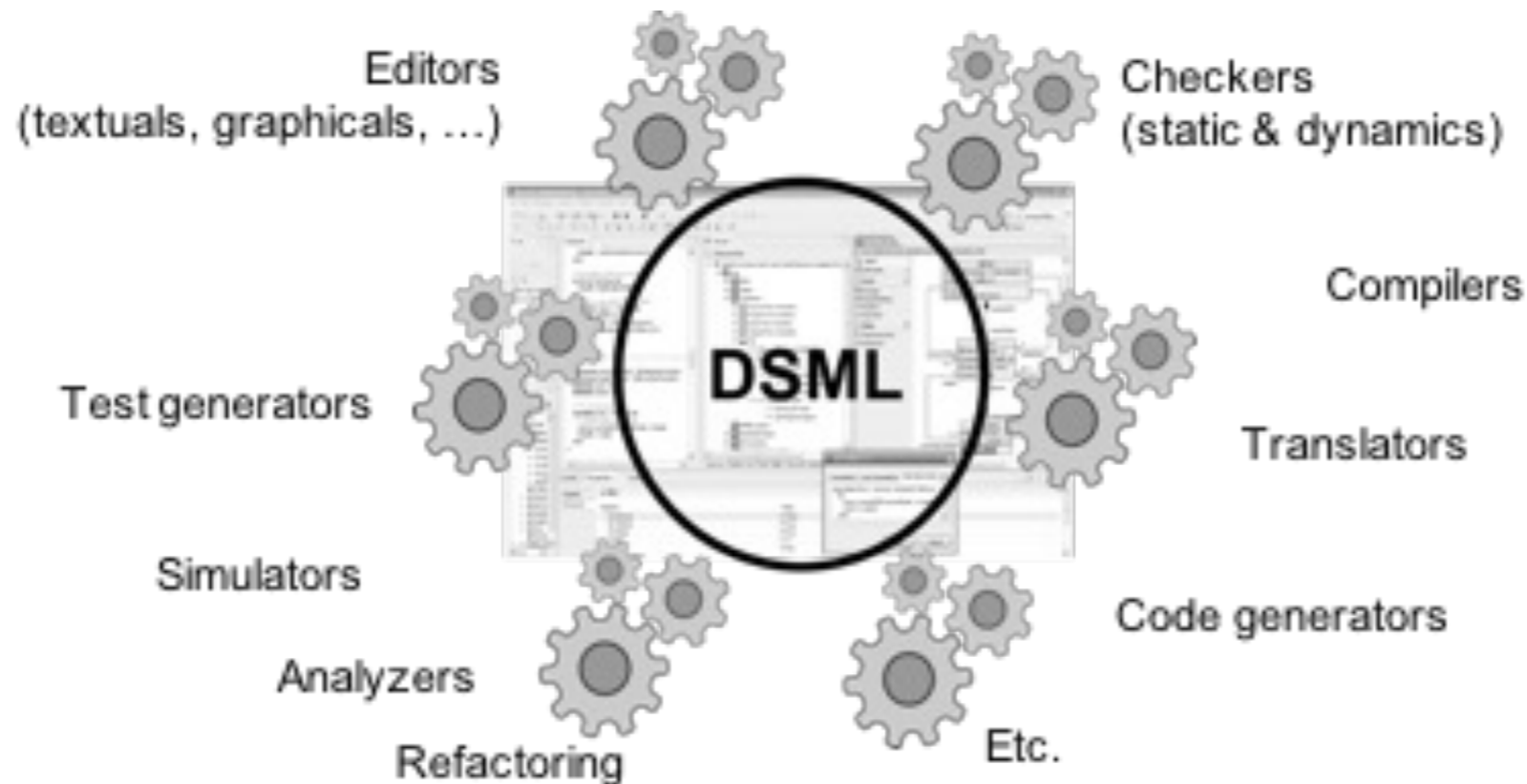


Available at the library of the university  
=> This is your textbook!



**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"***

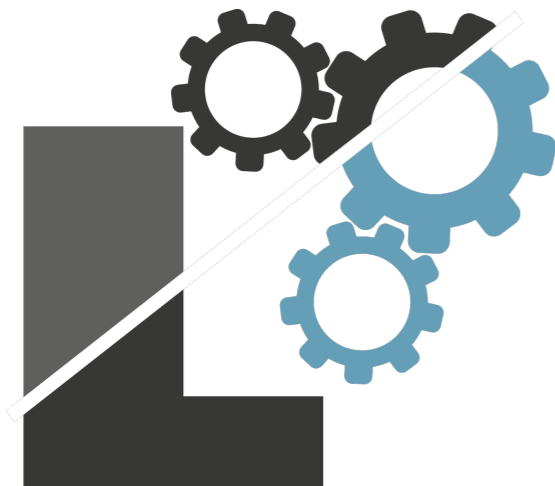
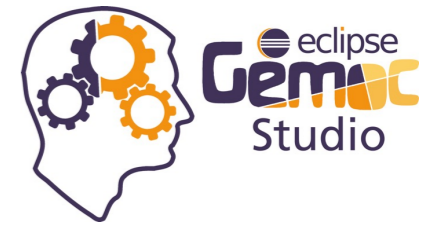
J-M. Favre, D. Gasevic, R. Lämmel, and E. Pek. "Empirical language analysis in software linguistics," In Software Language Engineering, volume 6563 of LNCS, pages 316-326. Springer, 2011.

# 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
  - JetBrains' 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>)...

# The 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 IDE interface for Arduino Designer. The top toolbar includes navigation and execution icons. The main workspace is divided into several panels:

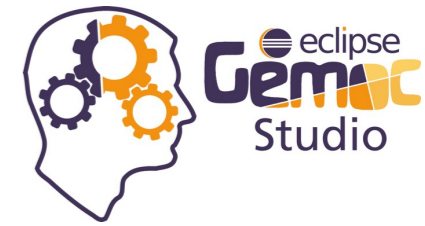
- Debug Console:** Shows the execution of a variable declaration: `(VariableDeclaration) org.gemoc.sequential.model.arduino.impl.VariableDeclarationImpl@9b02cf4 -> execute()`.
- Hardware View:** Shows a 3D model of an Arduino Board with three LEDs connected to pins 0, 1, and 2. The LEDs are labeled RED LED, BLUE LED, and WHITE LED.
- Sketch View:** Shows a flowchart for a sketch named 'newSketch'. It starts with `i = 0`, followed by a 'Repeat 5' loop. Inside the loop, there are three LED control blocks: `BLUE LED : ((i/2)%2)`, `RED LED : ((i/2)%2)`, and `WHITE LED : ((i/4)%2)`. Each block has a corresponding variable `i` and a value of 0. Below the loop is a 'Set i = (i+1)' block, which is implemented as `i = i + 1`.
- Console View:** Shows the output of the modeling workbench console: `About to initialize and run the GEMOC Execution Engine... Initialization done, starting engine...`
- Variables View:** Shows a table of variables and their values:

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

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

MDE Introduction (ESIR3 SI, ASE)  
Benoit Combemale, Oct. 2023

# Transformation Lg Debugger



The screenshot shows the Gemoc Studio debugger interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for debugging and development. The main workspace is divided into several panes:

- Debug Console:** Shows the execution stack for the transformation rule. The current context is "Global context: Transformation".
- Variables:** A table showing the current state of variables during execution.
- Code Editor:** Displays the transformation rule code for "simpleAtoB".
- Outline:** Shows the structure of the project, including "sample.minitl 5".
- Console:** Displays the output of the GEMOC Execution Engine, showing initialization and execution steps.

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...

```
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
```

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...

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

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

MDE Introduction (ESIR3 SI, ASE)  
Benoit Combemale, Oct. 2023

# Farming System Model



Modeling - MyExploitation/analysis.scientific - Eclipse Platform

**Model Explorer**

- MyExploitation [farmingmodeling master]
  - Project Dependencies
  - src-gen
  - analysis.scientific
  - climate.simulation
  - cultures.activities
  - John.exploitation
  - representations.aird
  - schedule.simulation

**Logical Steps**

```

cultures.activities {
  culture corn {
    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**

**Outline**

- Schedule
  - NW of Exploitation 4 fields
    - corn LABOUR scheduled on 13/jan
    - LABOUR
    - Tractor John
    - People Henry
    - 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

**\*corntasks dependencies**

**Hydro Analysis**

	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	0.01527933...	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	0.02865451...	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	0.05883383...	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	298.0
apr 9	0.0	11.0	775.0
apr 10	11.5	11.4	293.0
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>

MDE Introduction (ESIR3 SI, ASE)  
Benoit Combemale, Oct. 2023

# 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

Debug test2.ad [Gemoc Sequential eExecutable Model]

- Gemoc debug target
- Model debugging
  - (ForkNode) test2.forkNode1 -> execute()
  - (Activity) test2 -> execute()
- Global context : Activity

(x)= Variables

Name	Value
heldTokens (ActivityFinalNode_finalNode2 :ActivityFinalNode)	0
heldTokens (ForkNode_forkNode1 :ForkNode)	0
heldTokens (InitialNode_initialNode2 :InitialNode)	[activitydiagram.impl.ControlTokenImpl]
heldTokens (JoinNode_joinNode1 :JoinNode)	0

Breakpoints

- Opaque Action action2

No details to display for the current selection.

Console \*test2 Activity Diagram

Properties

Opaque Action action2

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

Gemoc Engines Status

- test2.ac 8

Multidimensional Timeline

All execution states (11)

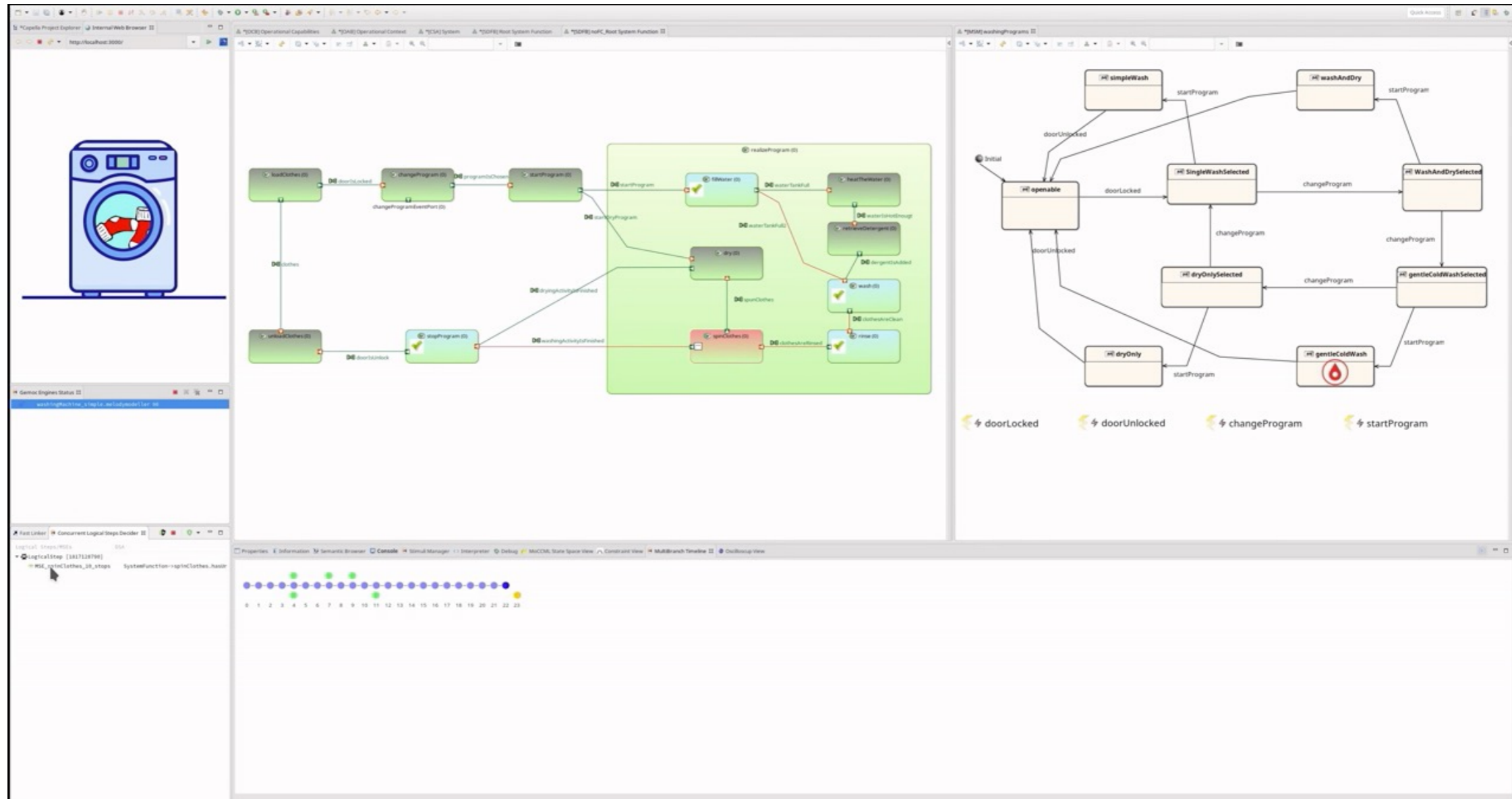
Timeline for dynamic information

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)

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

# xCapella



# FCL



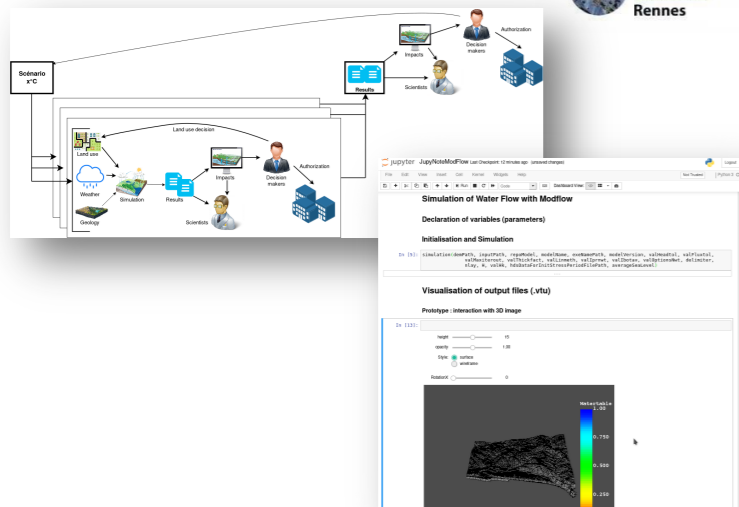
The screenshot displays the Eclipse Gemoc Studio interface with several panels:

- Project Explorer:** Shows the project structure for 'MiniQuadCopter.fcl'. The 'FCL Mode Diagram' is selected.
- Model Explorer:** Shows the 'QuadCtrlNoFMU Function Diagram' and 'QuadCtrlNoFMU Mode Diagram'.
- Code Editor:** Displays the 'MiniQuadCopter.fcl' code, showing a state machine with modes like 'modeAutomata masterAutomata' and 'mode ManualStabilizedFlight'. The 'enabledFunctions' section lists various controller functions.
- Diagram View:** Shows a state transition diagram with nodes for 'ManualStabilizedFlight' and 'AutomatedFlight', connected by transitions labeled with events like 'when RCCommandReceived'.
- Console:** Shows the output of the model execution, including a timestamp and a message: '14:48:28.584 [Worker-3: Decoration Calculat...].git.util.FS - remaining o...'

# Crosscutting Challenges for Various Domains

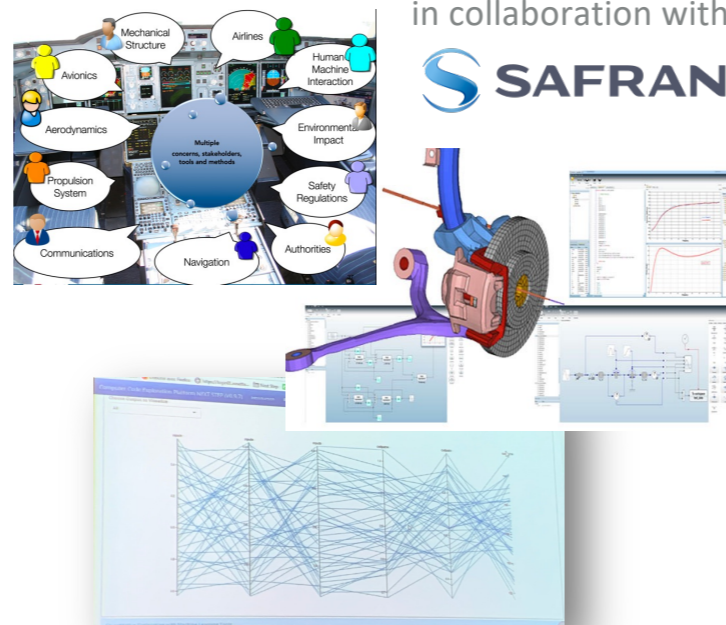
## Towards Virtual Labs From Modeling Environment to Digital Twin

in collaboration with



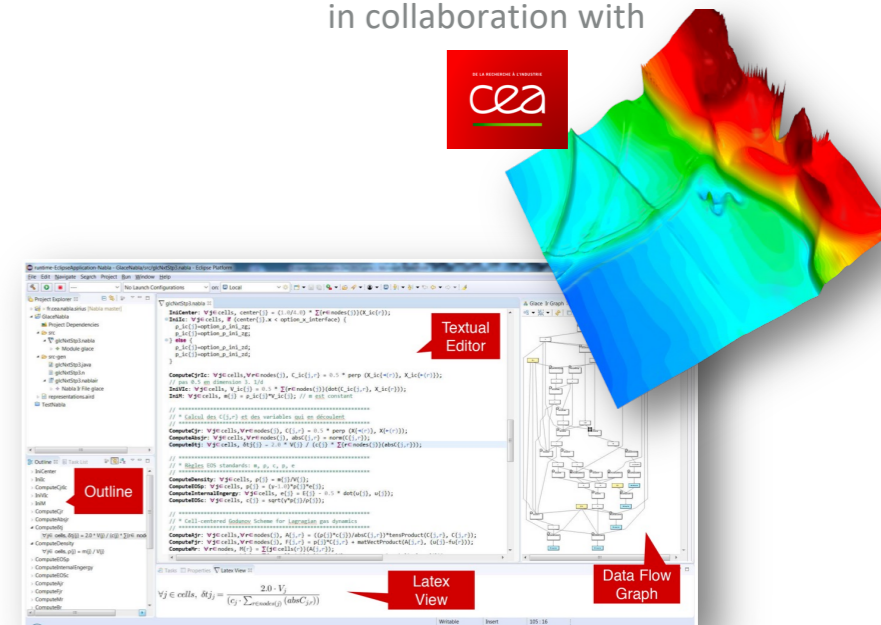
Tradeoff Analysis for Water Flood Prediction

in collaboration with



Design Space Exploration for Cyber-Physical Systems

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 *and graphical* syntaxes (Xtext/Langium & Sirius)**
  - **Compiler and interpreter (Xtend/Typescript)**
- **Companion webpage:**

**<http://combemale.fr/course/esir/esir3/>**



# You will learn how to

---

- **Automatically translate** abstract design models to executable code, test cases and documentation
- **Automatically manipulate** your model/code to analyze and refactor it
- **Build or customize your own abstractions**, or even **software languages and development environments**, 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