

MODEL-DRIVEN *(SOFTWARE)* ENGINEERING

COURSE INTRODUCTION

MASTER 1 ICE, 2017-2018

BENOIT COMBEMALE
PROFESSOR, UNIV. TOULOUSE, FRANCE

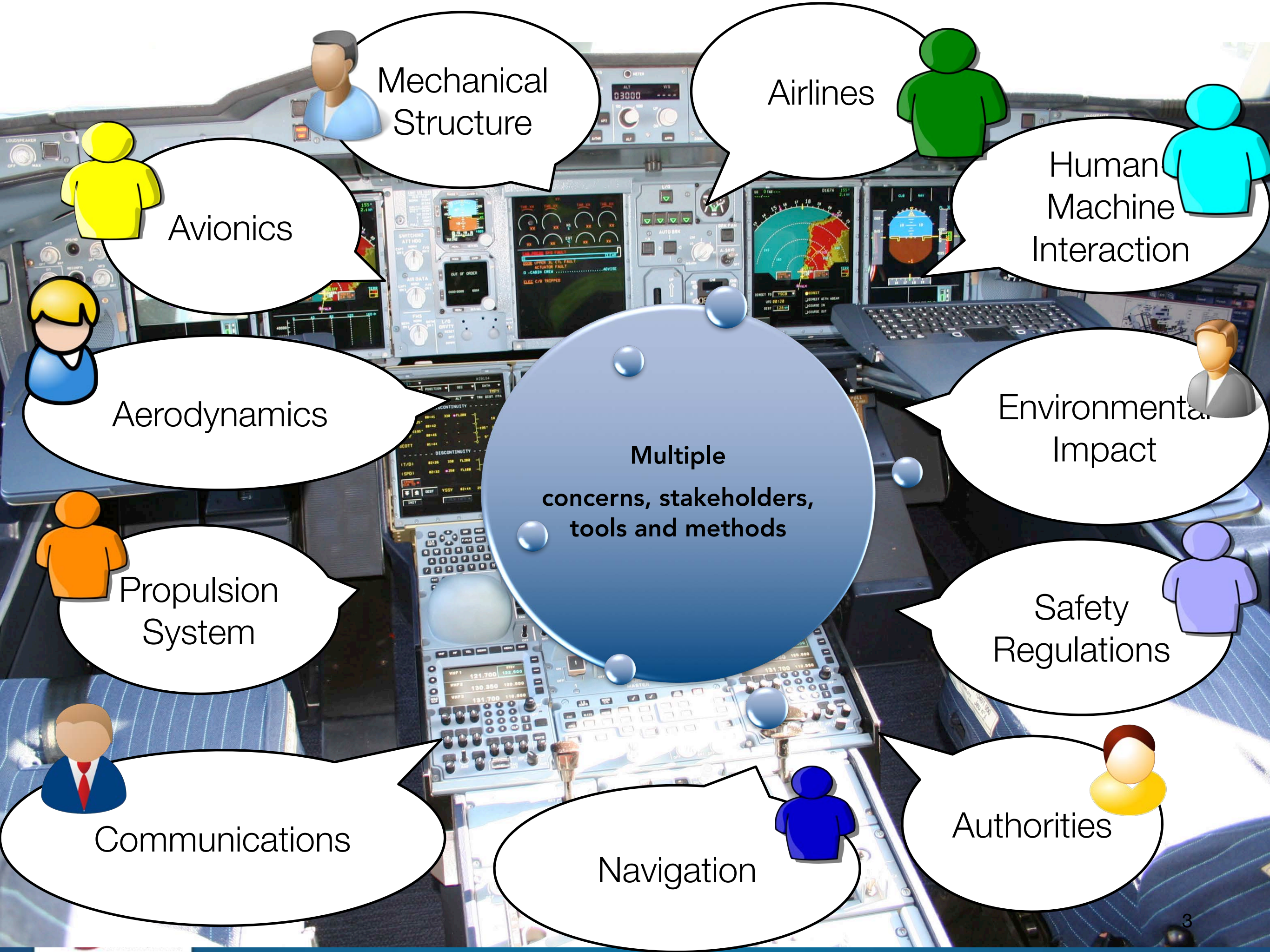
[HTTP://COMBEMALE.FR](http://combemale.fr)
[BENOIT.COMBEMALE@IRIT.FR](mailto:benoit.combemale@irit.fr)
[@BCOMBEMALE](#)

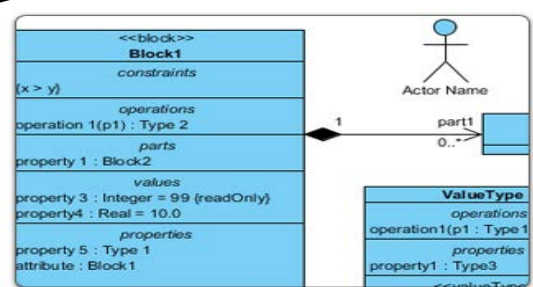
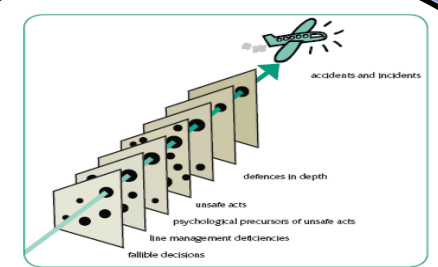
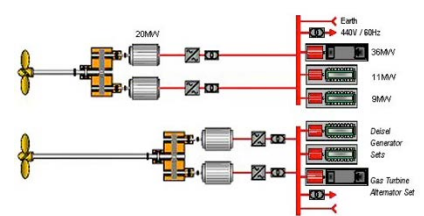
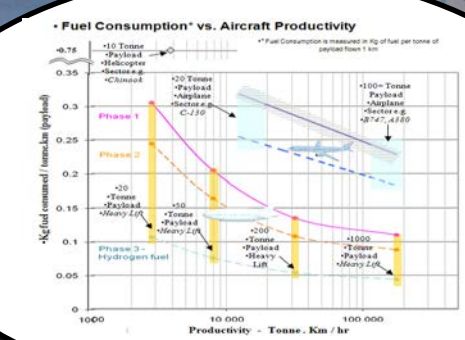
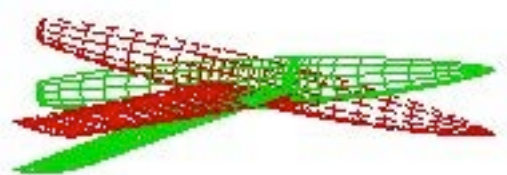
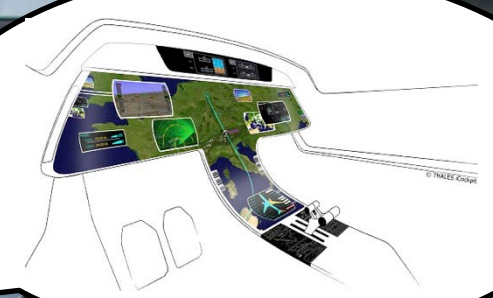
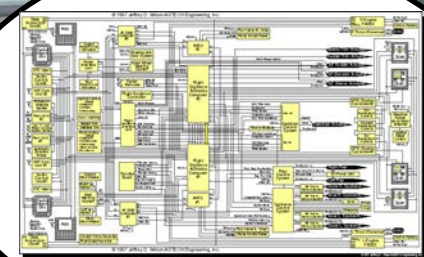
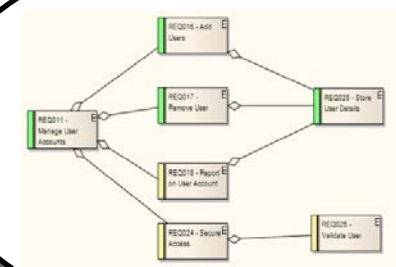
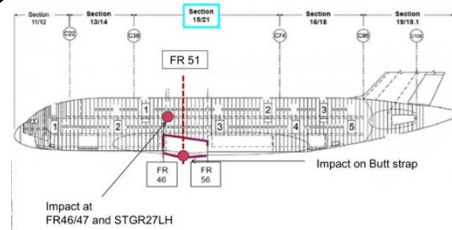


Complex Software-Intensive Systems

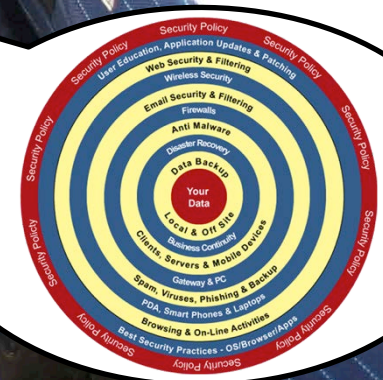
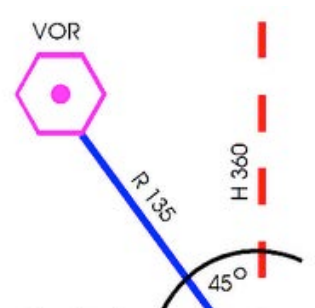


- ▶ Multi-engineering approach
- ▶ Domain-specific modeling
- ▶ High variability and customization
- ▶ Software as integration layer
- ▶ Openness and dynamicity





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

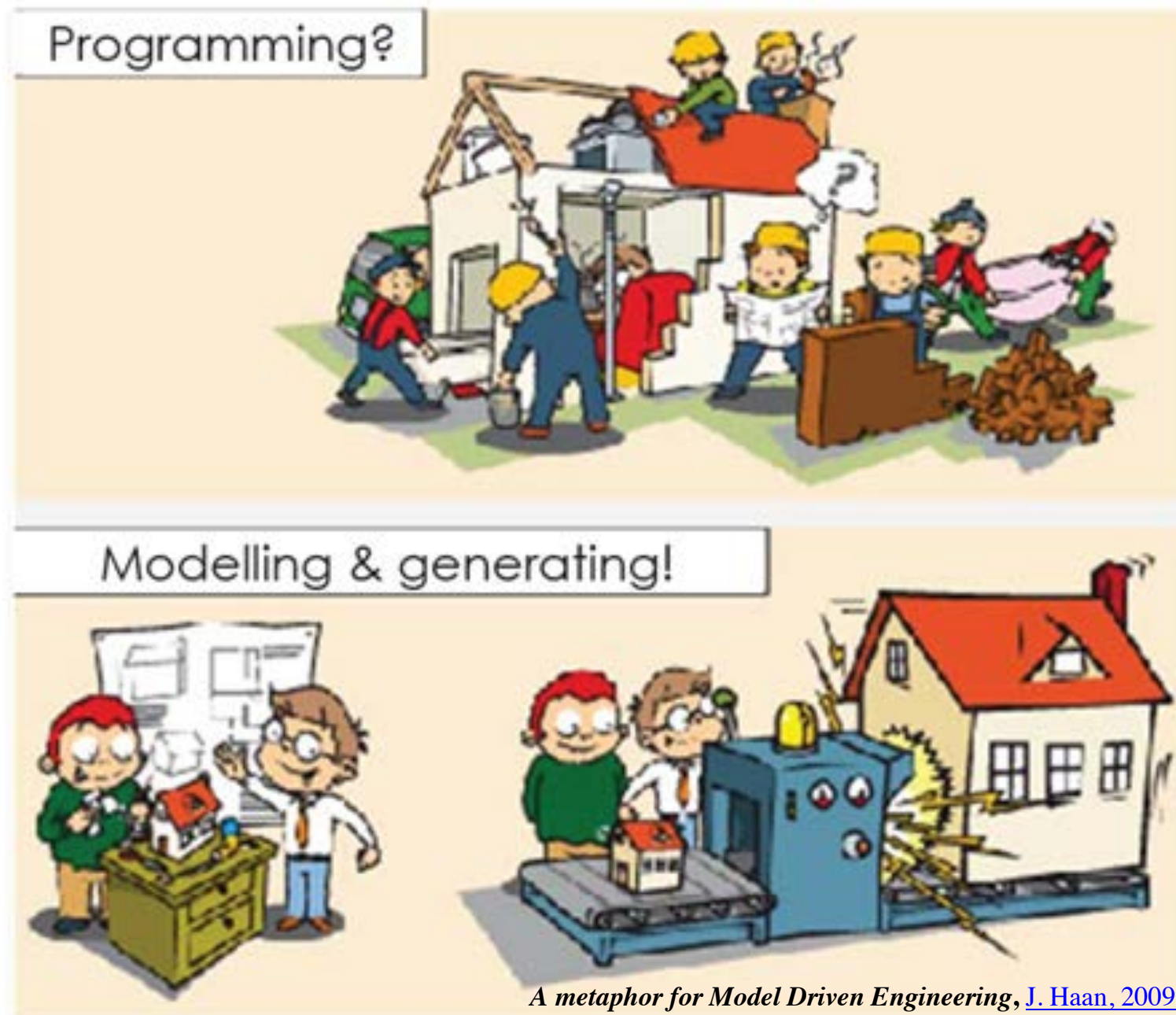
Model and Reality in Software

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

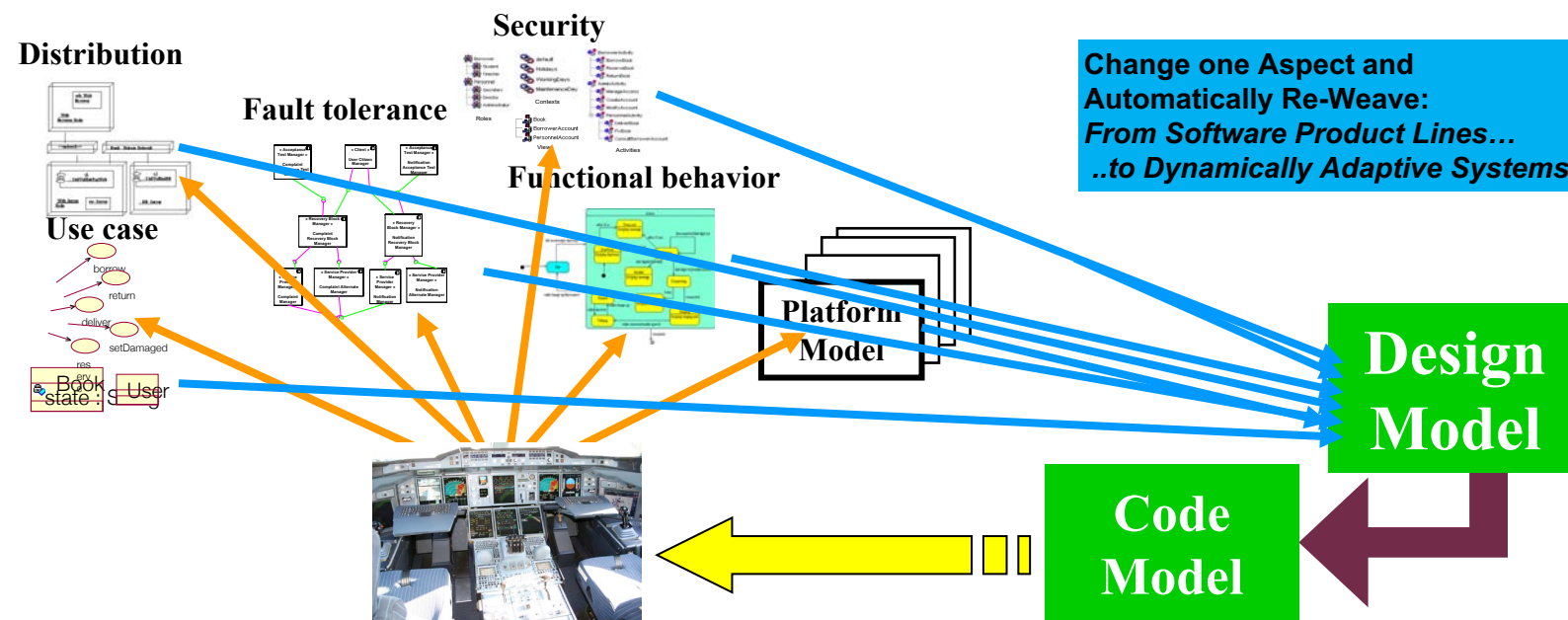


- Software Models: from contemplative to productive

Towards Model-Driven Engineering (MDE)



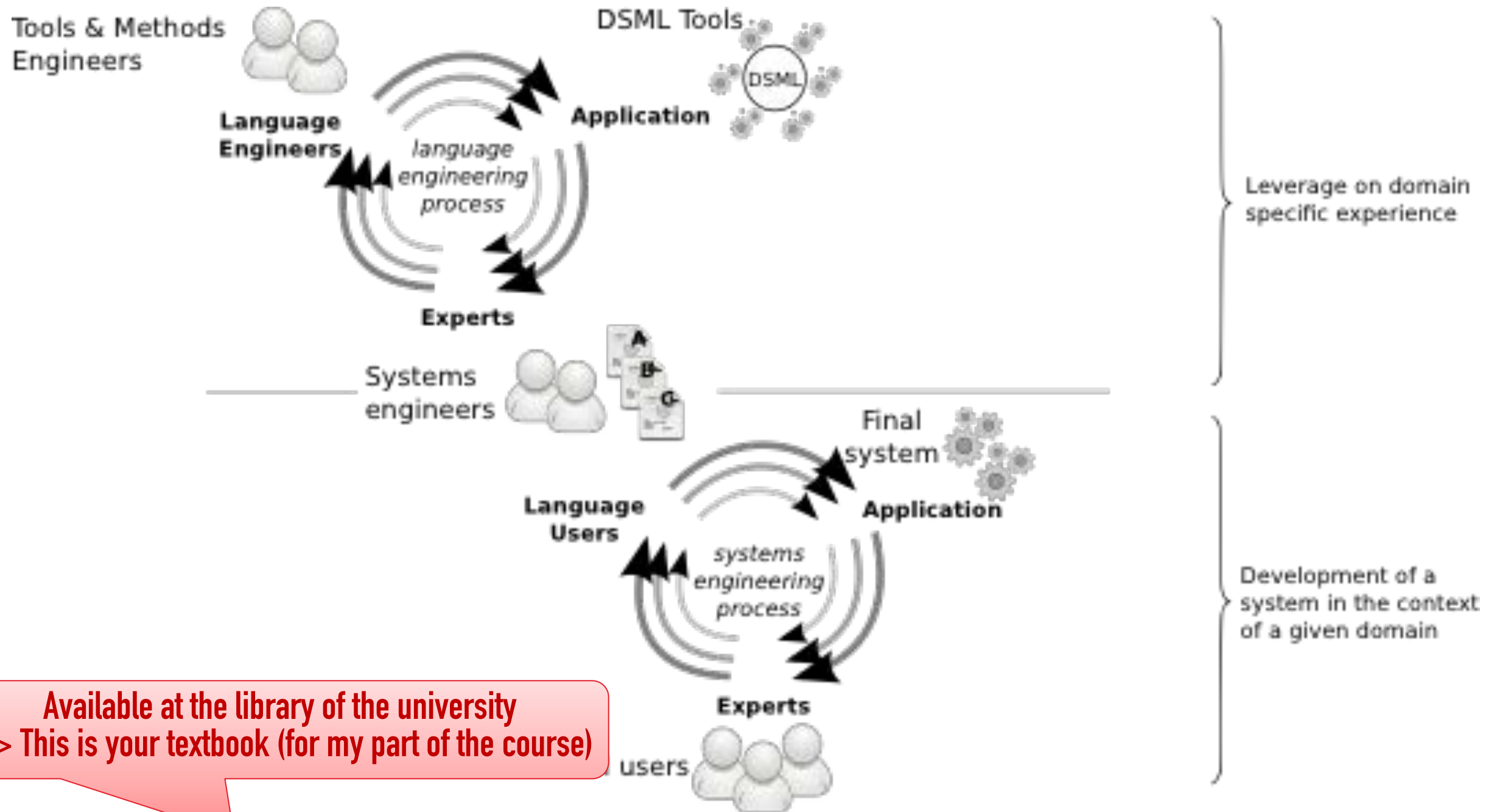
Model-Driven Engineering (MDE)



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

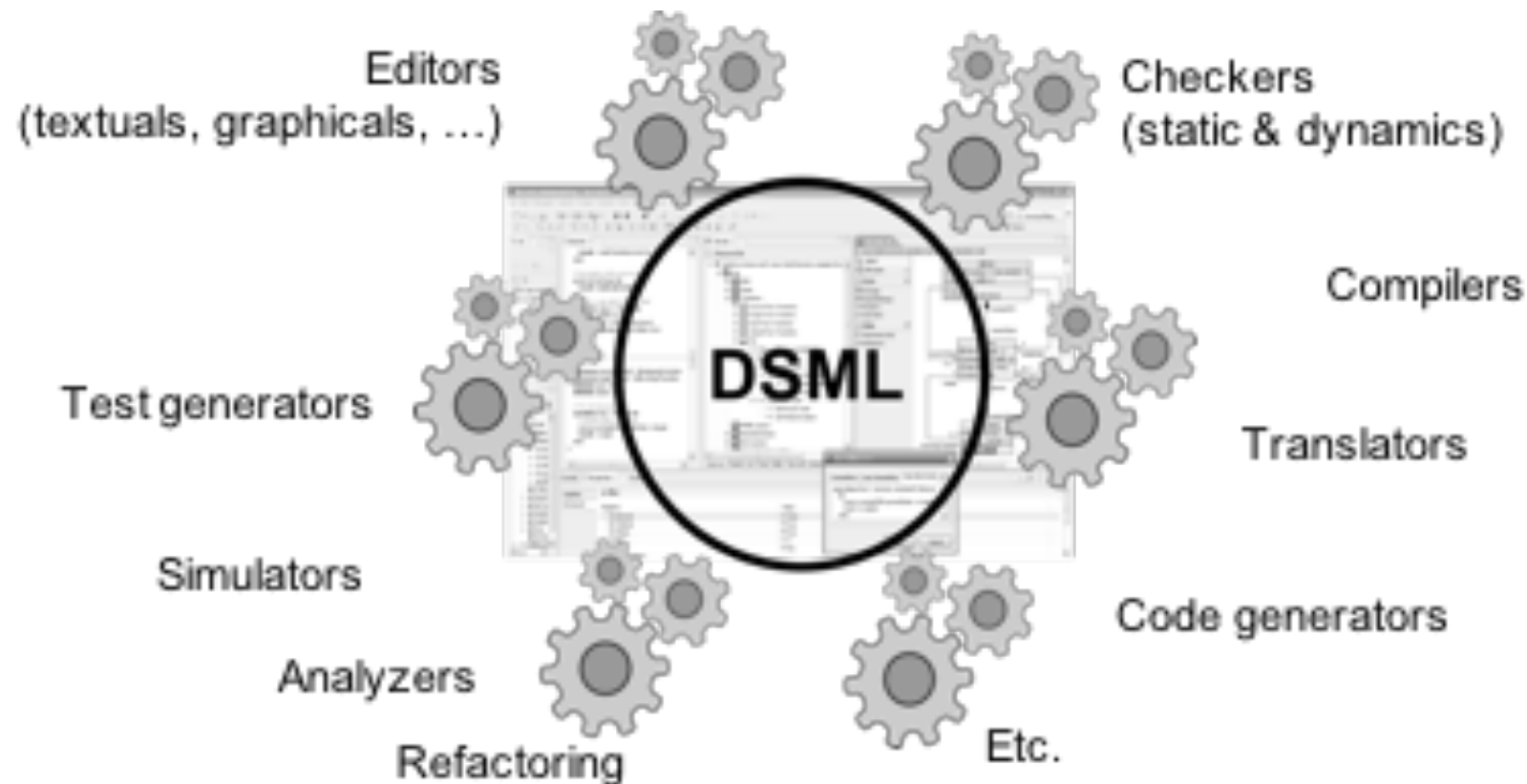
J. Whittle, J. Hutchinson, and M. Rouncefield, "The State of Practice in Model-Driven Engineering," IEEE Software, vol. 31, no. 3, 2014, pp. 79–85.

Model-Driven Engineering (MDE)



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>

Model-Driven Engineering (MDE)



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

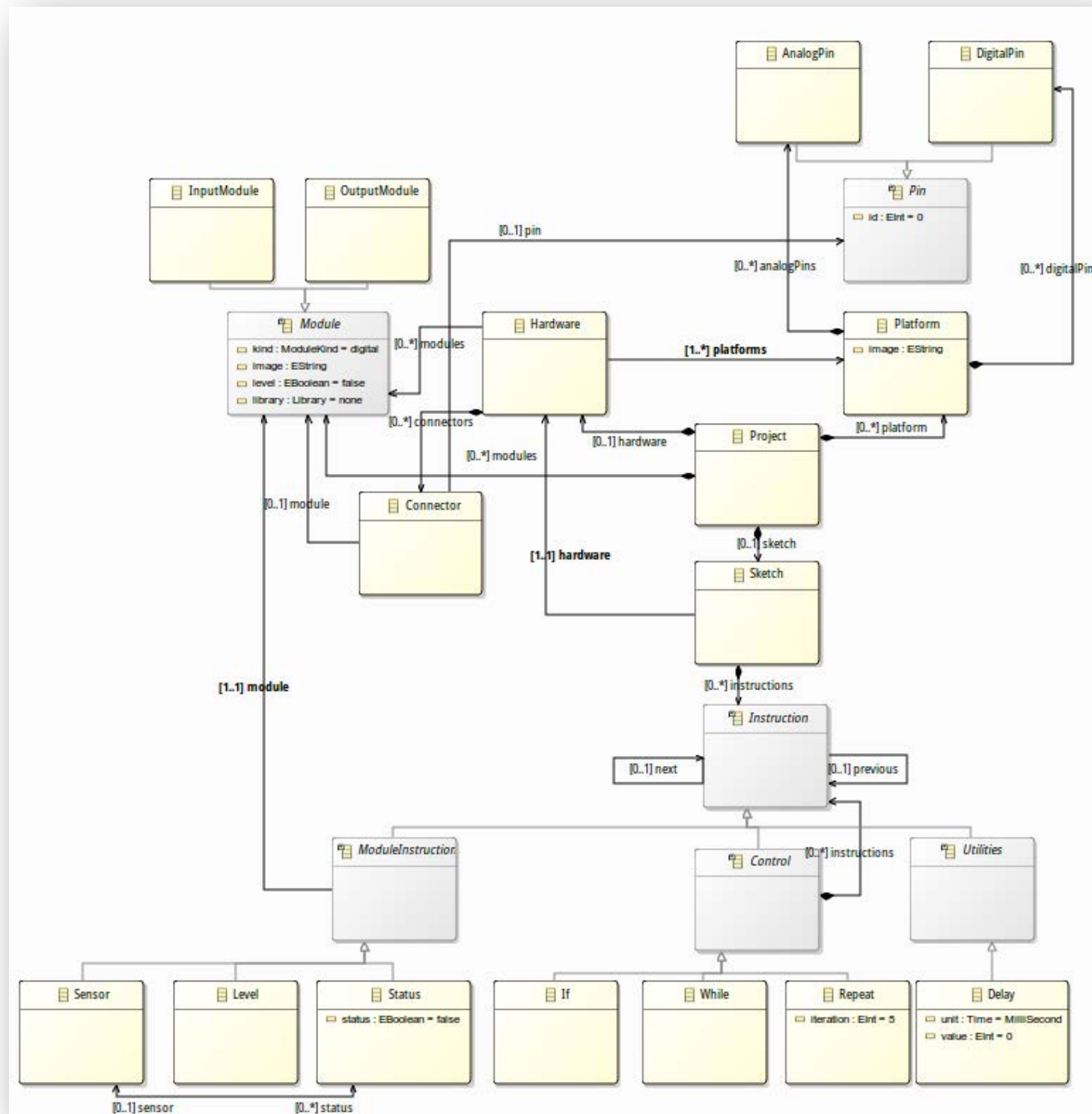
"Software Languages are Software Too"

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.

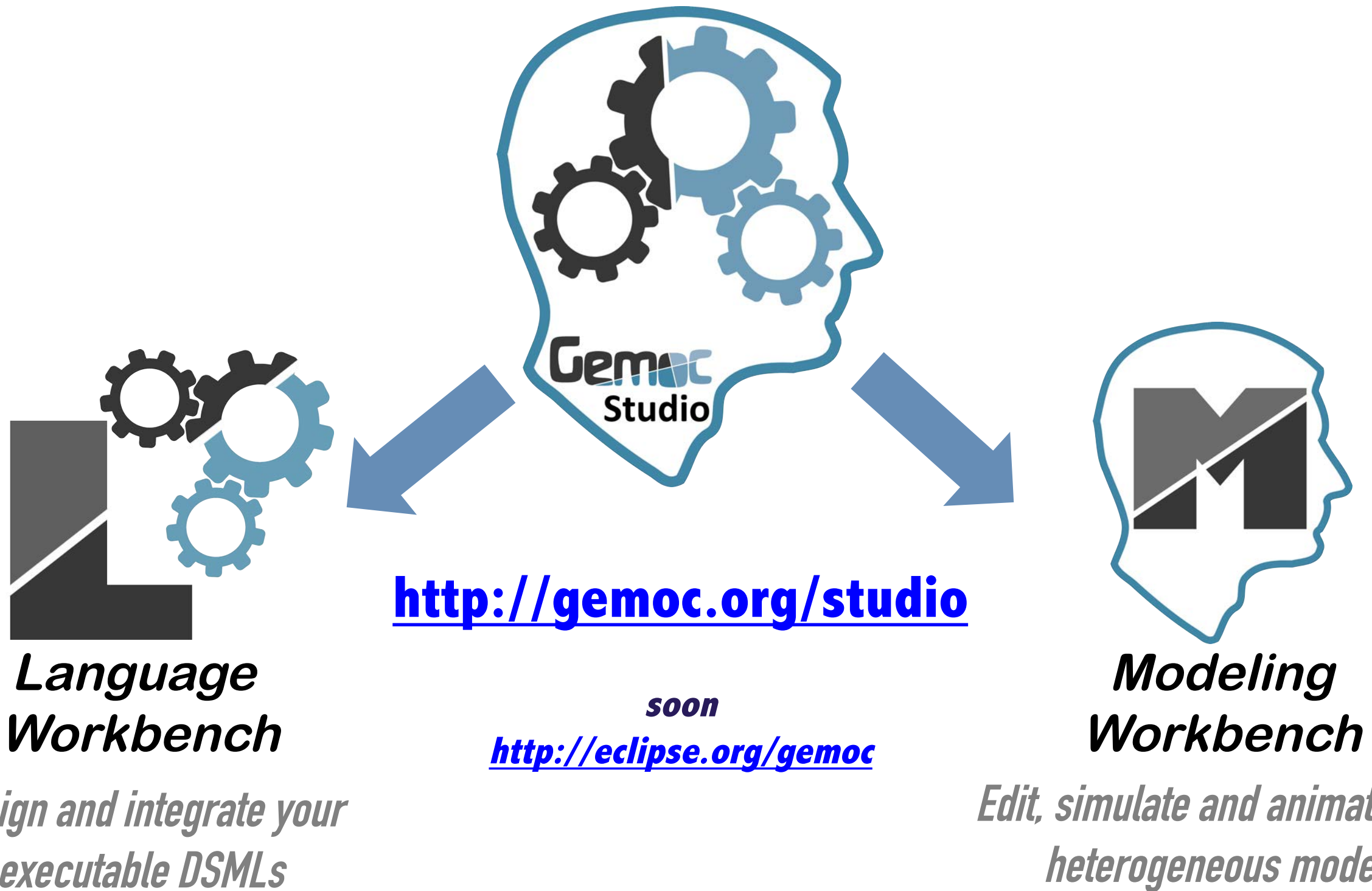
Software Language Engineering (SLE)

- 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
 - JetBrains'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>)...

Ecore Tools: Graphical Edition of Ecore Models



The GEMOC Studio



Arduino Designer



The screenshot shows the Eclipse IDE with the Gemoc Studio extension. The main window is titled "runtime-arduinoDebug - platform:/resource/org.gemoc.sample.arduino.sequential.blinker/model Laird/Sketch - Gemoc Studio".

Debug Console: Shows the execution of a model debugging target. The output includes: (VariableDeclaration) org.gemoc.sequential.model.arduino.impl.VariableDeclarationImpl@9b02cf4 -> execute() and Global context : Project.

Hardware View: Displays an "Arduino Board" with three digital pins (0, 1, 2) connected to three LEDs: RED LED, BLUE LED, and WHITE LED.

Sketch Editor: Shows a "newSketch" with a "Repeat 5" loop. The loop contains three LED control blocks: "BLUE LED : ((i/2)%2)", "RED LED : ((i/2)%2)", and "WHITE LED : ((i/4)%2)". Each block has an "int 2" output. Below the loop is a "Set i = (i+1)" block, which is part of a larger block containing a variable "i" and a loop condition "i (=0)".

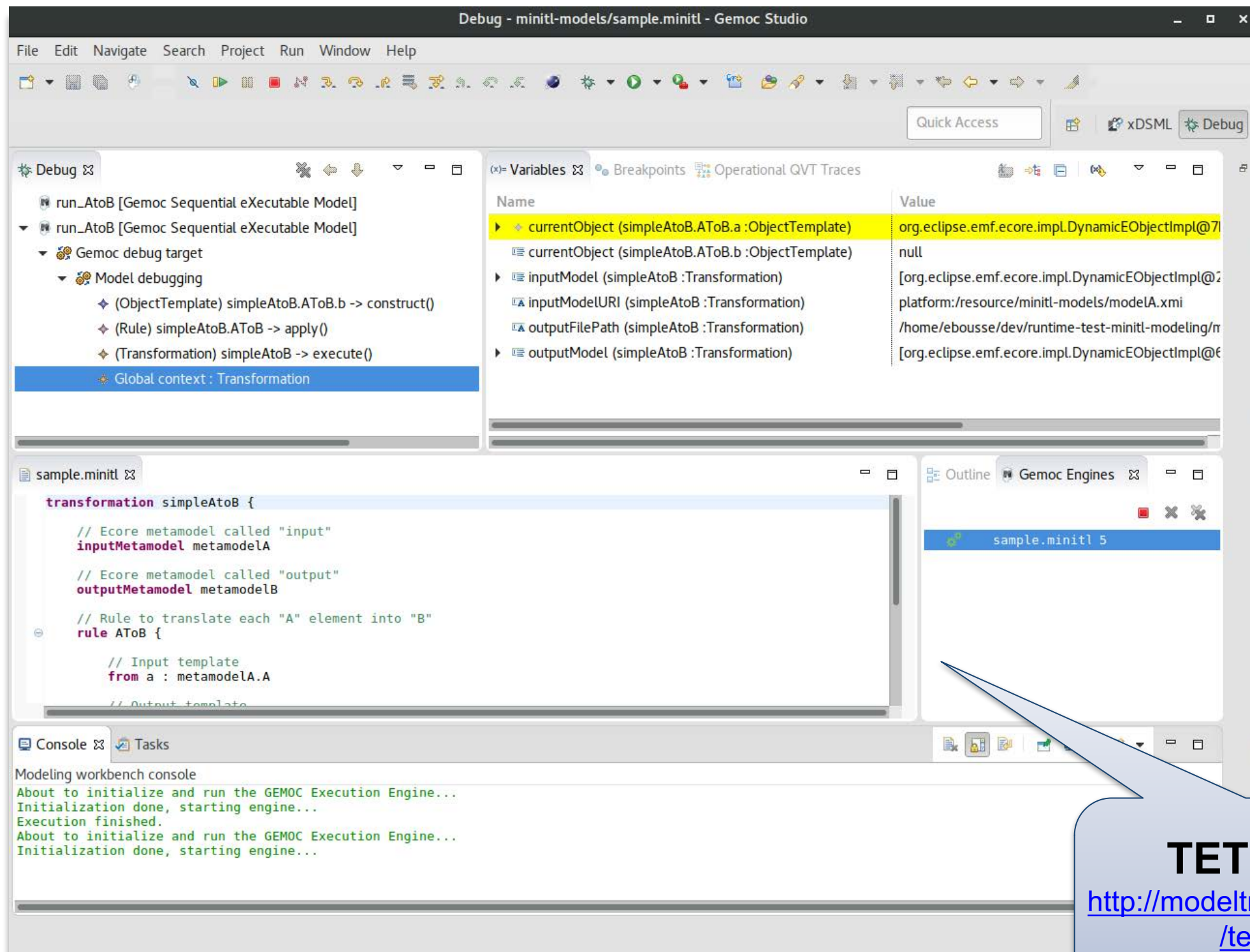
Console: Shows the "Modeling workbench console" output: "About to initialize and run the GEMOC Execution Engine..." and "Initialization done, starting engine..."

Variables View: Lists the current state of variables:

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



TETRABox

<http://modeltransformation.net/tetrabox/>

Wimmer, Bousse *et al.*

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

MDE Introduction (M1ICE)
Benoit Combemale, Dec. 2017

Farming System Modeling



Modeling - MyExploitation/analysis.scientific - Eclipse Platform

Model Explorer [farmingmodeling master]

- Project Dependencies
 - src-gen
 - analysis.scientific
 - climate.simulation
 - cultures.activities
 - John.exploitation
 - representations.aird
 - schedule.simulation

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

***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	0.05863383...	0.000

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**

surfaces ratios: Watered, Fodder

solver search limit : 8 secs

Extra Water needed : 161600m³

***corntasks dependencies**

```

graph TD
    LABOUR --> SEMIS
    SEMIS --> FERTILISATION
    SEMIS --> IRRIGATION
    FERTILISATION --> RECOLTE
    IRRIGATION --> RECOLTE
    
```

***Climate Data**

	Rain (mm)	Temperature (°C)	Ray (Joules/cm²)
7 apr	5.0	10.4	626.0
8 apr	0.0	10.4	298.0
9 apr	0.0	11.0	775.0
10 apr	11.5	11.4	293.0
11 apr	0.5	9.9	700.0
12 apr	2.5	10.7	450.0
13 apr	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

Palette

- Daily
- Crop
- Ewe
- Resource
- Surface
- Assign Surface
- Assign Culture

Diagram

The diagram illustrates the farming system model. It shows a central 'Crop' entity (wheat, corn, sorgho) dedicated to a 'Tractor' and 'People' (John, Henry). The crop is assigned to a 'Surface' (50ha : 4 fields, 34ha : 2 fields, 100ha : 10 fields, 34ha : 1 field). The crop is also assigned to a 'Resource' (Massey Ferguson 1). The crop is also assigned to a 'Surface' (50ha : 4 fields, 34ha : 2 fields, 100ha : 10 fields, 34ha : 1 field). The crop is also assigned to a 'Resource' (Massey Ferguson 1). The crop is also assigned to a 'Surface' (50ha : 4 fields, 34ha : 2 fields, 100ha : 10 fields, 34ha : 1 field). The crop is also assigned to a 'Resource' (Massey Ferguson 1).

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

MDE Introduction (M1ICE)
Benoit Combemale, Dec. 2017

Activity Diagram Debugger



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 eXecutable Model]

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

Global context : Activity

(x) 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

test2

```
graph TD
    initialNode2((initialNode2)) -- edge3 tokenOfferCount=1 --> forkNode1[ForkNode1]
    forkNode1 -- edge4 tokenOfferCount=0 --> action2[action2]
    forkNode1 -- edge5 tokenOfferCount=0 --> action3[action3]
    action2 -- edge6 tokenOfferCount=0 --> joinNode1[JoinNode1]
    action3 -- edge7 tokenOfferCount=0 --> joinNode1
    joinNode1 -- edge8 tokenOfferCount=0 --> finalNode2((finalNode2))
```

Properties Opaque Action action2

Property	Value
Activity	Activity test2
Incoming	Control Flow edge4
Name	action2
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>

Content of the course

- **Domain-Specific Languages**
- **Model management (static analysis, generators)**
- **Experimentations with the Eclipse Modeling Framework (incl. Ecore and OCL), Xtext, Xtend, and Sirius**

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, domain-specific, software-intensive systems
- Eventually **limit the accidental complexity** of industrial processes 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