

MODEL-DRIVEN (*SOFTWARE*) ENGINEERING

HACK YOUR OWN LANGUAGE!

MASTER 1 ICE, 2020-2021

BENOIT COMBEMALE
PROFESSOR, UNIV. RENNES 1 & INRIA, FRANCE

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



WHO WE ARE?



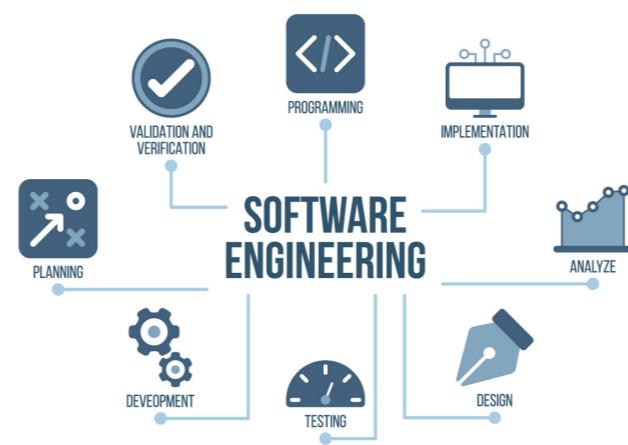
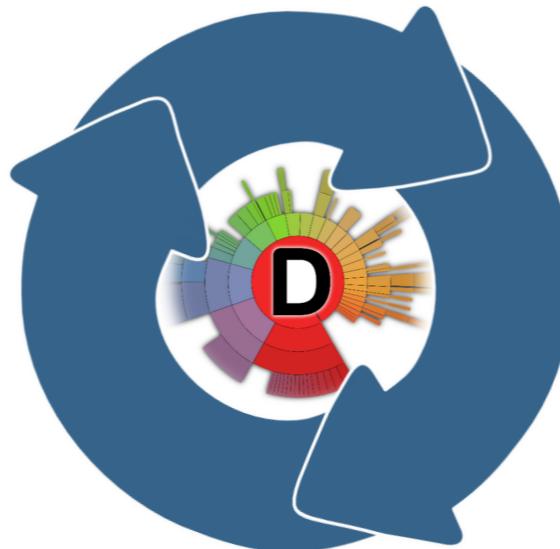
The DiverSE team

- Inria/IRISA project-team in **Software Engineering (SE)**
- Strong background in **Model-Driven software/systems Engineering (MDE)**
- Software architecture, software variability, software testing, software languages, simulation, resilience eng.
- Applied to smart, heterogeneous, and distributed CPS (e.g., IoT, transport, Industry 4.0)
- 9 Prof. and Inria/CNRS researchers, 1 Inria RSE, ~20 PhD, 3 Post-doc, 3 SE
- Deductive and empirical scientific approaches
- Open source software development
- Strong contractual activity (esp. EU and industry projects)



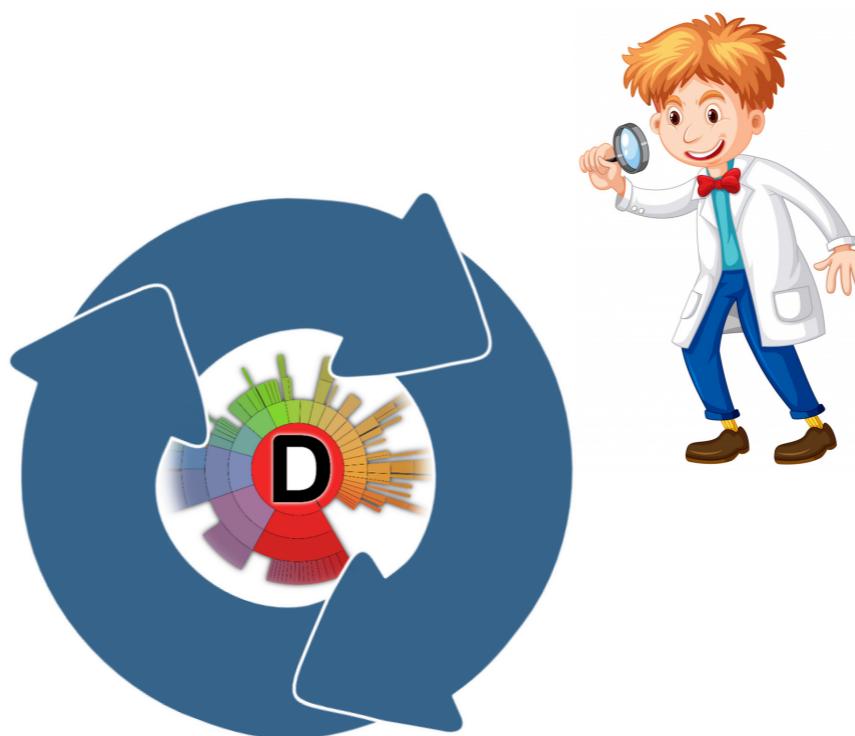
The DiverSE team

A Software Engineering Group

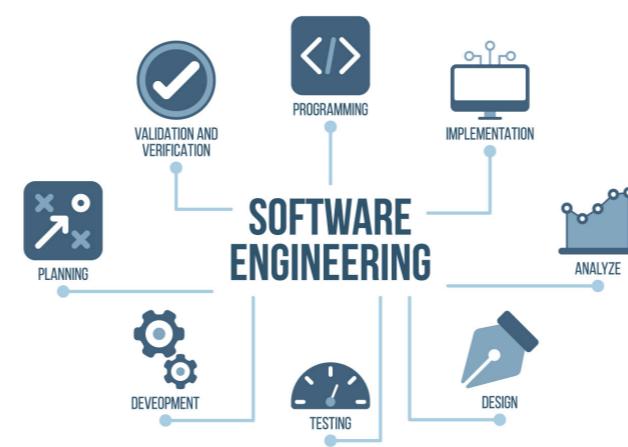


The DiverSE team

A Software Engineering Group



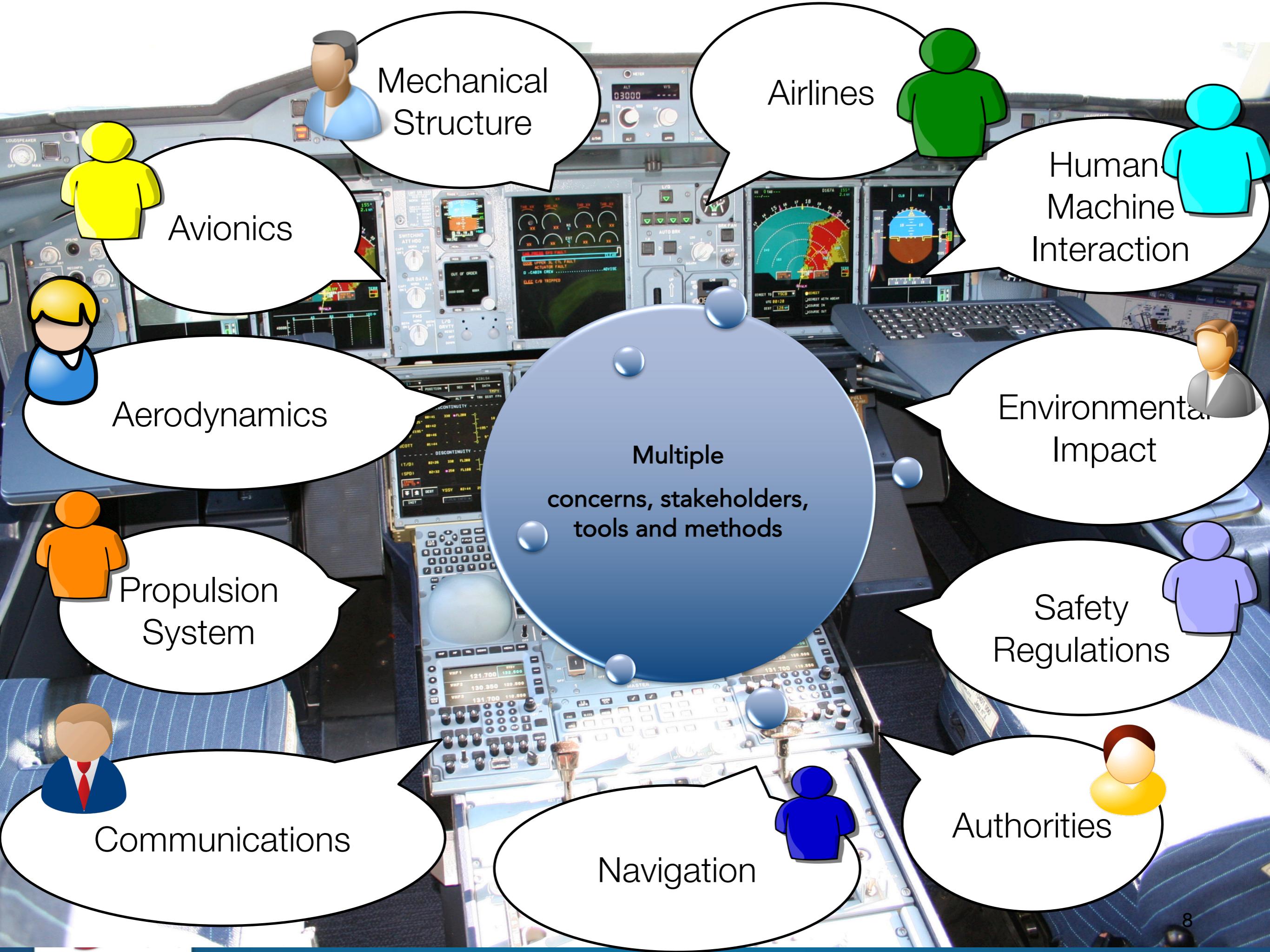
- Diversity of...
- stakeholders
 - concerns
 - configurations
 - platforms
 - environments
 - requirements...



- Multi-engineering approach
- Domain-specific modeling
- High variability and customization
- Platform heterogeneity
- Openness and dynamicity

Complex Software-Intensive Systems





Mechanical
Structure

Avionics

Aerodynamics

Propulsion
System

Communications

Navigation

Airlines

Human
Machine
Interaction

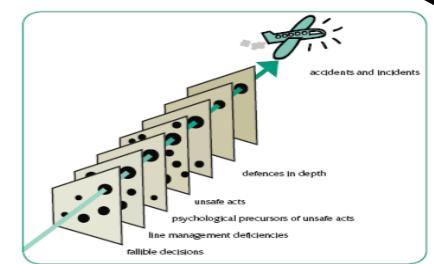
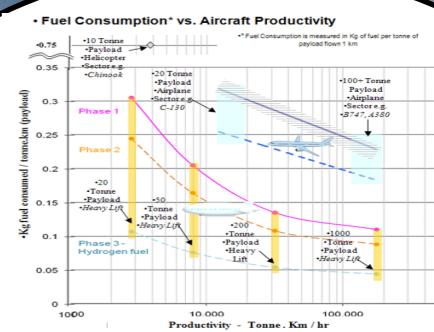
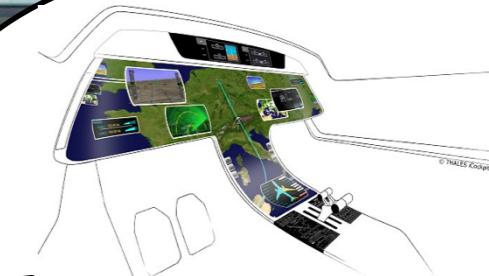
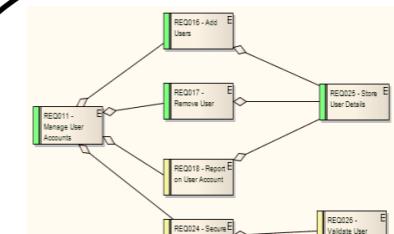
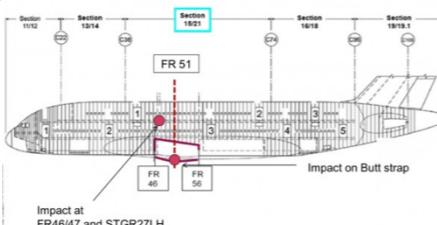
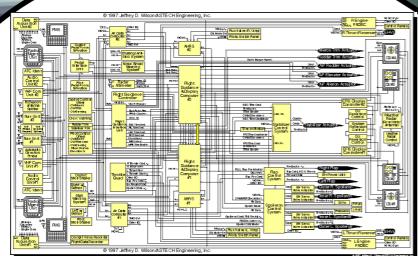
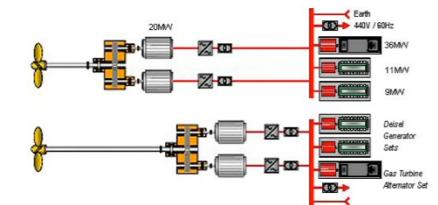
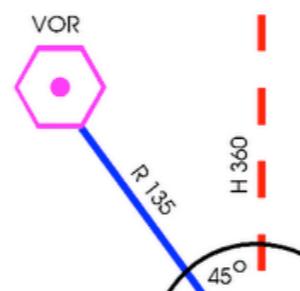
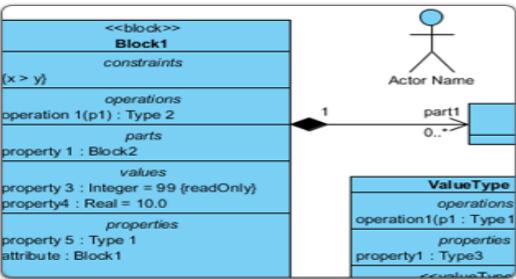
Environmental
Impact

Safety
Regulations

Authorities

Multiple
concerns, stakeholders,
tools and methods

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

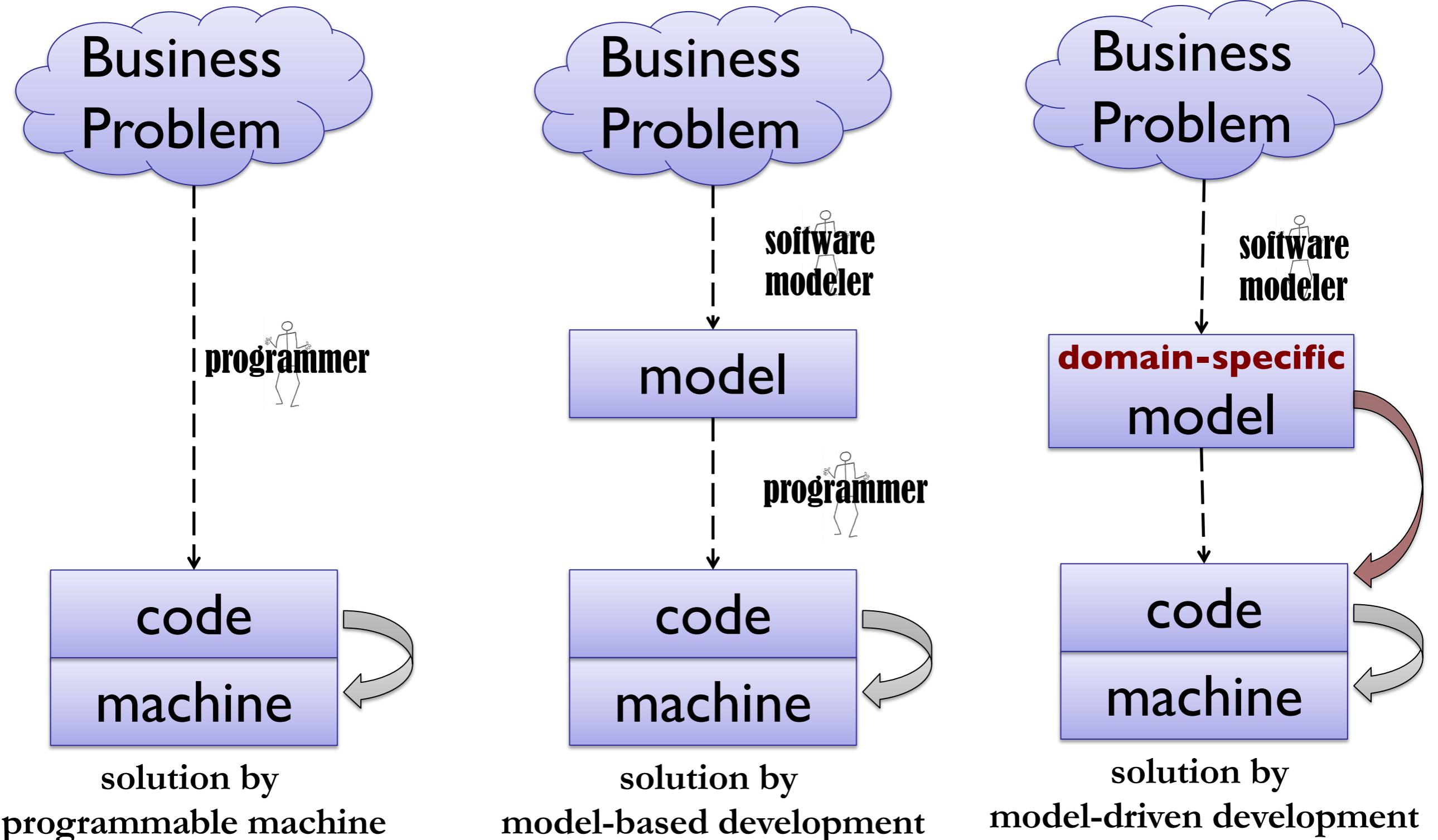
Model and Reality in Software

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



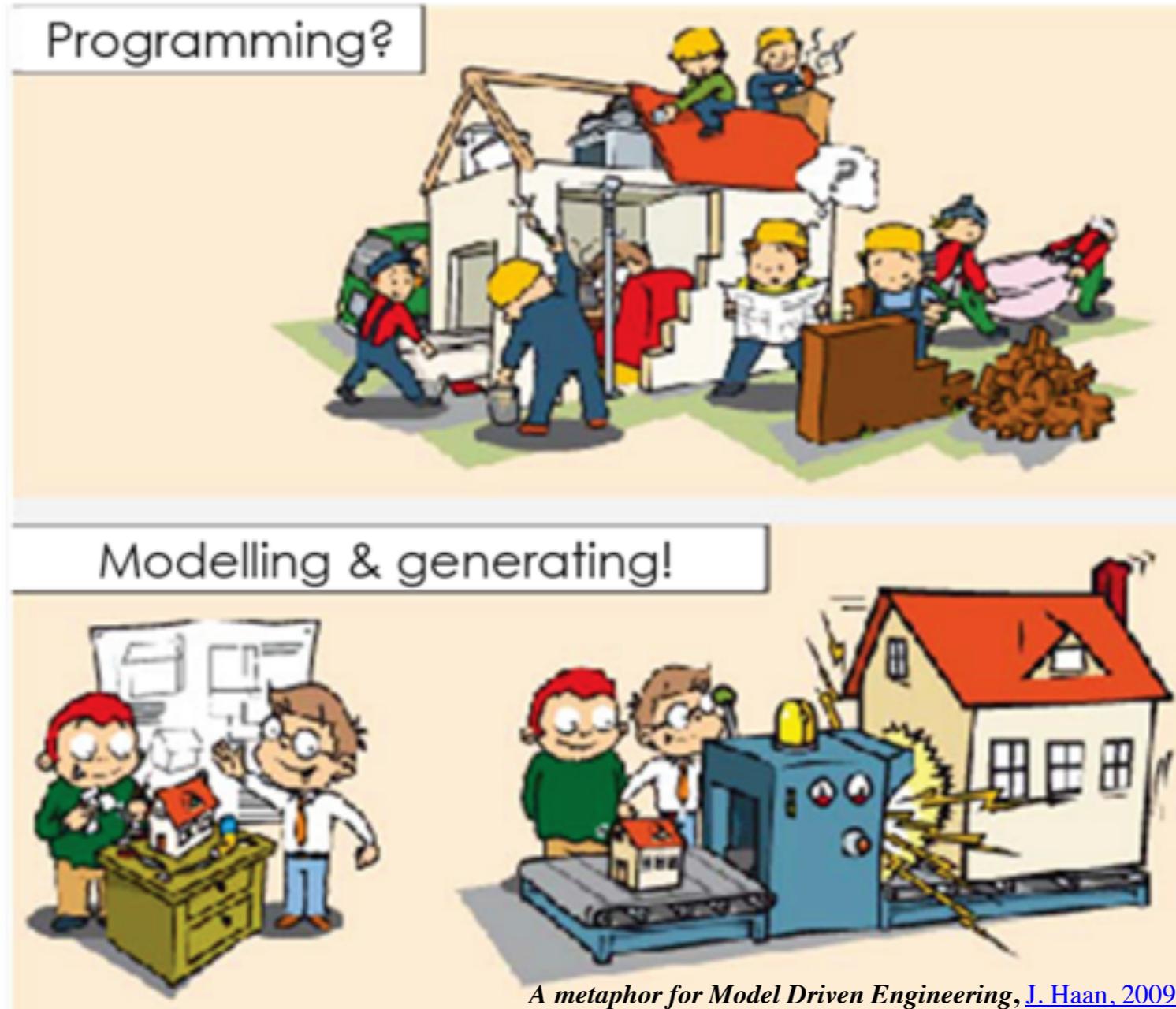
- Software Models: from contemplative to productive

Evolution in Software Modeling

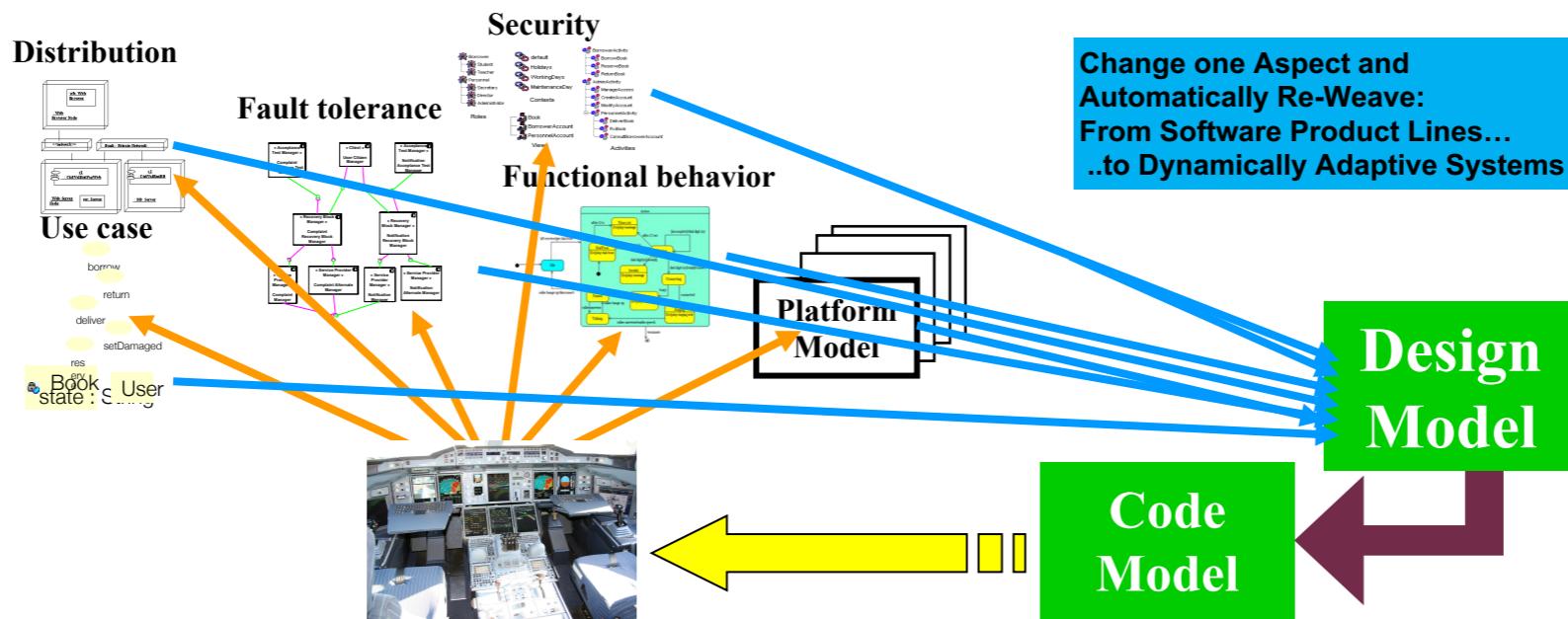


From Gregor Engels (Universität Paderborn, Germany)
Panel "When will Code become Irrelevant?", MoDELS 2011.

Towards Model-Driven Engineering (MDE)



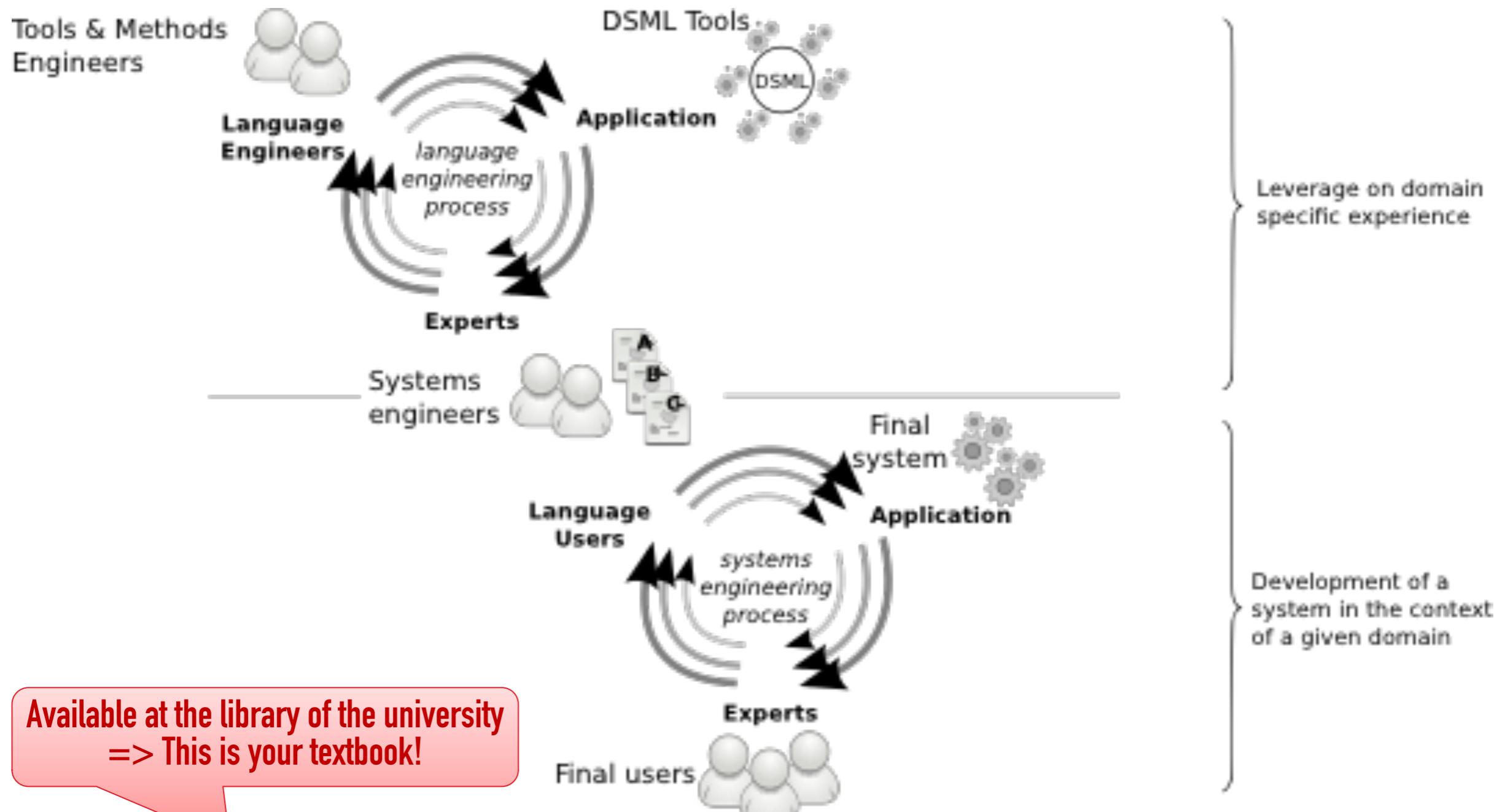
Model-Driven Engineering (MDE)



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

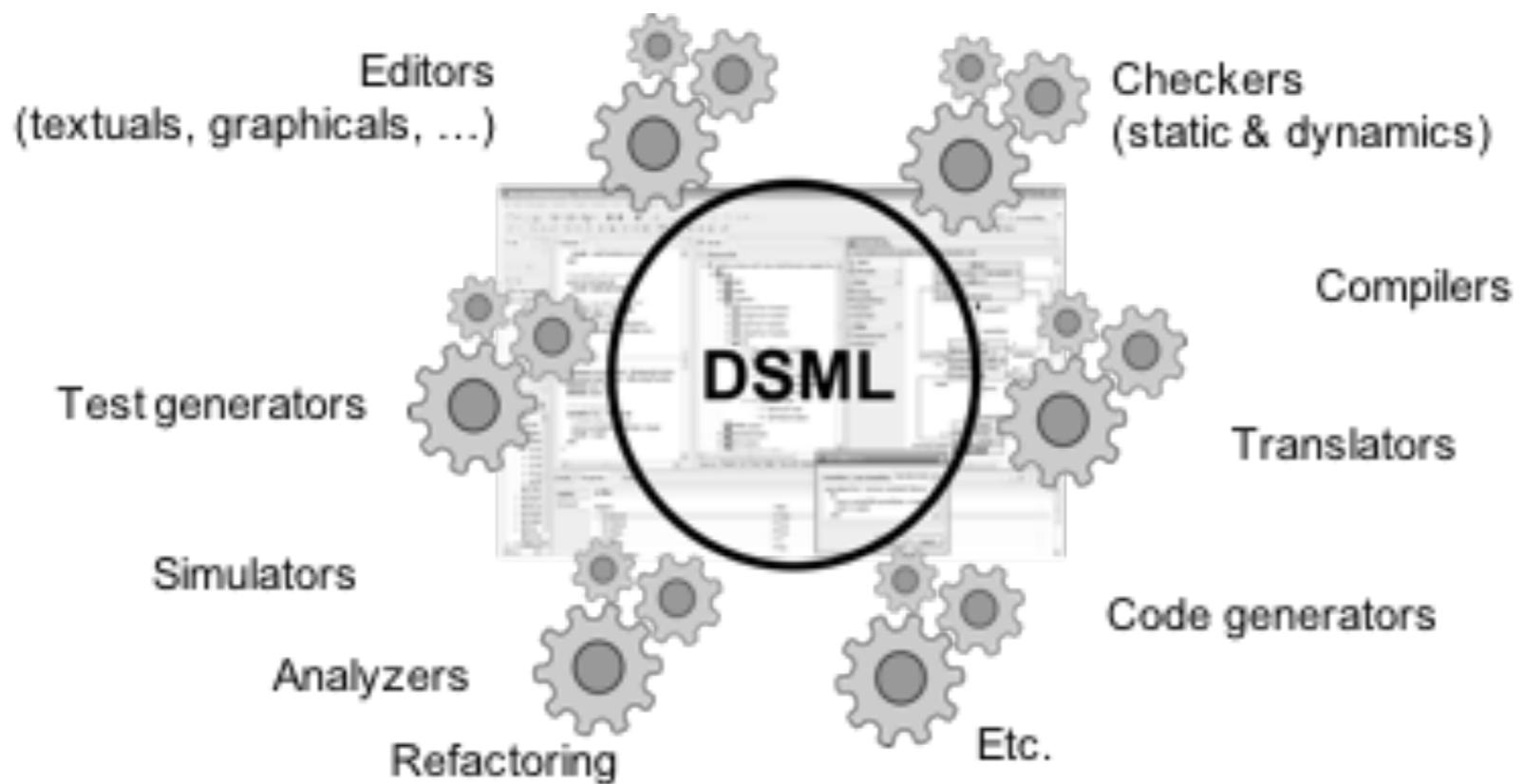
J. Whittle, J. Hutchinson, and M. Rouncefield, "The State of Practice in Model-Driven Engineering," IEEE Software, vol. 31, no. 3, 2014, pp. 79–85.

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

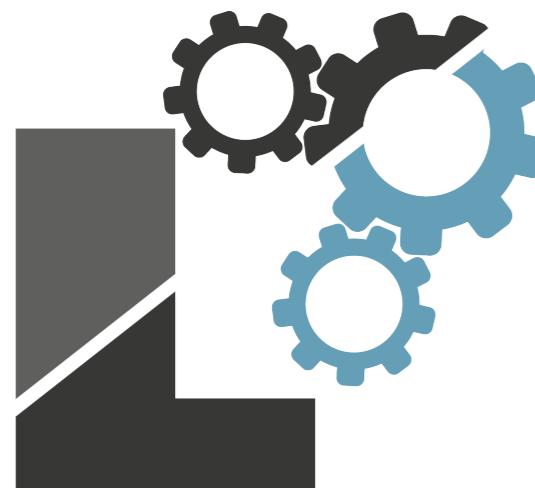
"Software Languages are Software Too"

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
 - 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>)...

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 Arduino Designer interface within the Eclipse GEMOC Studio environment. The interface is divided into several panes:

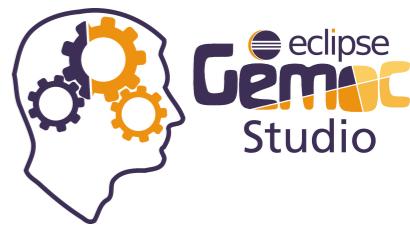
- Top Left:** A "Debug Configurations" view showing a project named "model.arduino".
- Top Center:** A "runtime-arduinoDebug - platform:/resource/org.gemoc.sample.arduino.sequential.blinker/model.aird/Sketch - Gemoc Studio" window showing the menu bar and toolbar.
- Left Column:** A "Debug" view showing a tree structure for "blinker [ALE Launcher]". The "Model debugging" node is expanded, showing "(VariableDeclaration) org.gemoc.sequential.model.arduino.impl.VariableDeclarationImpl@9b02cf4 -> execute()".
- Middle Left:** A "Hardware" view showing a schematic of an Arduino Board with three LEDs labeled "RED LED", "BLUE LED", and "WHITE LED" connected to pins 0, 1, and 2 respectively.
- Middle Right:** A "Sketch" view showing a statechart-like diagram for a "newSketch" state. It features a "Repeat 5" loop with three parallel regions: "BLUE LED : (i%2)", "RED LED : ((i/2)%2)", and "WHITE LED : ((i/4)%2)". Each region contains an LED component and a guard condition involving variable "i". Below the repeat loop is a "Set i = (i+1)" transition with guards "i < 5" and "i >= 0".
- Bottom Left:** A "Console" view showing the output: "Modeling workbench console", "About to initialize and run the GEMOC Execution Engine...", and "Initialization done, starting engine...".
- Bottom Right:** A "Variables" view showing a table of variables and their 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>

MDE Introduction (M1ICE)
Benoit Combemale, Feb. 2021

Transformation Lg Debugger



The screenshot shows the Eclipse Gemoc Studio interface with the title "Debug - minitl-models/sample.minitl - Gemoc Studio". The left pane displays the "Debug" view with a tree structure of debug targets, including "run_AtoB [Gemoc Sequential eXecutable Model]" and "Gemoc debug target". The "Model debugging" node is expanded, showing steps like "construct()", "apply()", and "execute()". A blue bar highlights "Global context : Transformation". The center pane shows the "Variables" view with a table of variables and their values. The right pane shows the "Outline" and "Gemoc Engines" views. The bottom pane contains a "Console" tab showing log output related to the GEMOC Execution Engine initialization.

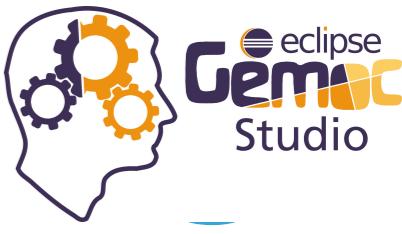
Name	Value
currentObject (simpleAtoB.AToB.a :ObjectTemplate)	org.eclipse.emf.ecore.impl.DynamicEObjectImpl@7
currentObject (simpleAtoB.AToB.b :ObjectTemplate)	null
inputModel (simpleAtoB :Transformation)	[org.eclipse.emf.ecore.impl.DynamicEObjectImpl@2]
inputModelURI (simpleAtoB :Transformation)	platform:/resource/minitl-models/modelA.xmi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

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

Wimmer, Bousse et al.

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

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

Outline

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**

	Extra Water	Rain	Hyd	Biomass	LAI
John's Exp...					
Surface...					
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	0.05144648...	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

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 Foddered

solver search limit :8 secs

Extra Water needed : 161600m³

Tractor, People, John, Henry, Massey Fergusson 1

***corntasks dependencies**

LABOUR → SEMIS → RECOLTE → IRRIGATION

Climate Data

	Rain (mm)	Temperature (°C)	Ray (Joules/cm ²)
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 (M1ICE)
Benoit Combemale, Feb. 2021

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

The diagram shows an activity named 'test2' with an initial node, a join node, a fork node, and a final node. Edges connect these nodes with token offer counts: edge3 (initialNode2 to forkNode1, tokenOfferCount=1), edge4 (forkNode1 to action2, tokenOfferCount=0), edge5 (forkNode1 to action3, tokenOfferCount=0), edge6 (action2 to joinNode1, tokenOfferCount=0), edge7 (joinNode1 to finalNode2, tokenOfferCount=0), and edge8 (finalNode2 to initialNode2, tokenOfferCount=0). Nodes 'action2' and 'action3' have a 'heldTokens = 0' constraint.

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)

The timeline shows 11 execution states numbered 0 to 10. State 4 is highlighted in orange, indicating the current execution state. States 0, 1, 2, 3, 5, 6, 7, 8, 9, and 10 are shown in purple.

Timeline for dynamic information

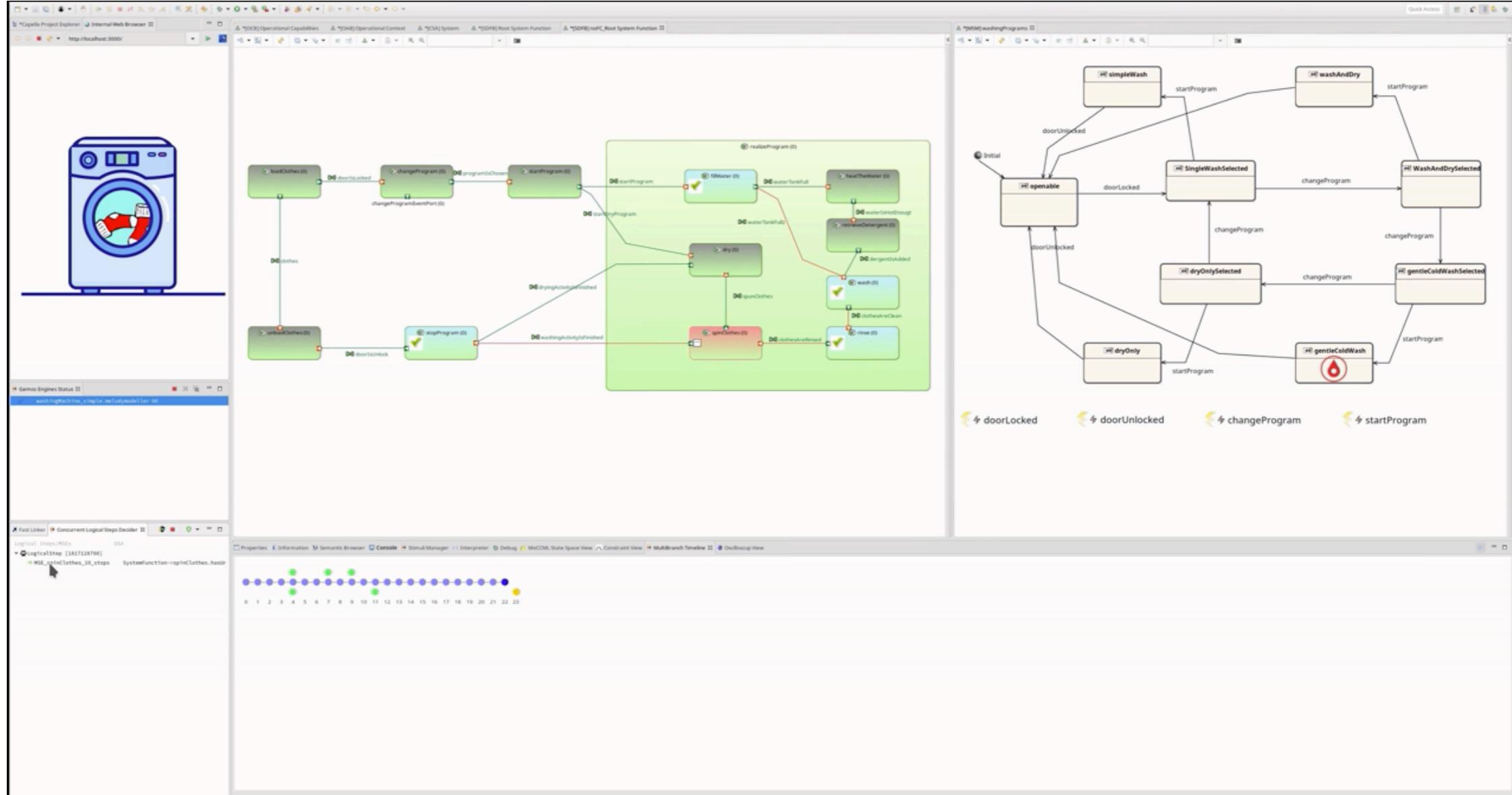
- trace (test2 :Activity)
- heldTokens (test2.initialNode2 :InitialNode)
- heldTokens (test2.forkNode1 :ForkNode)
- heldTokens (test2.action2 :OpaqueAction)
- heldTokens (test2.action3 :OpaqueAction)
- heldTokens (test2.joinNode1 :JoinNode)
- heldTokens (test2.finalNode2 :ActivityFinalNode)

Gemoc Engines Status

test2.ac 8

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

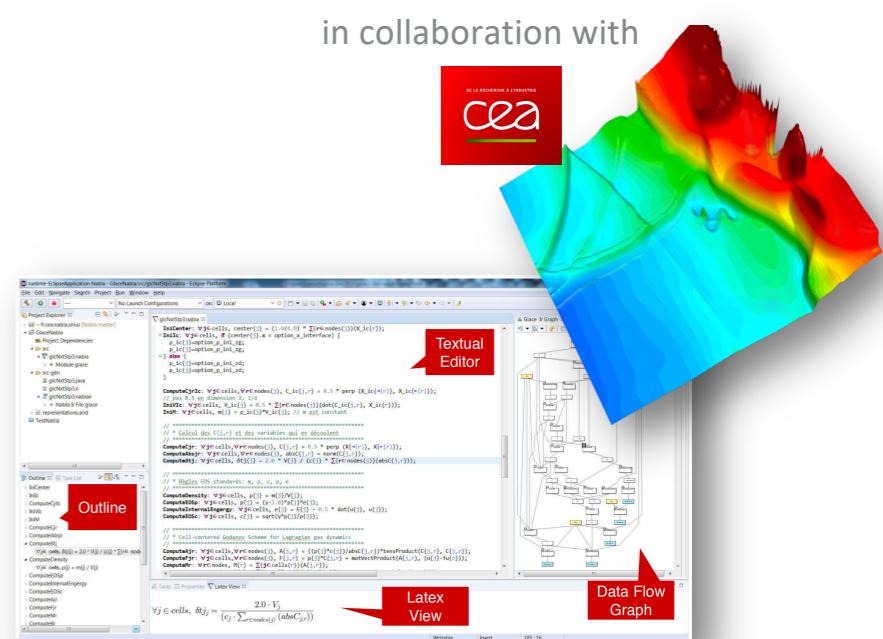
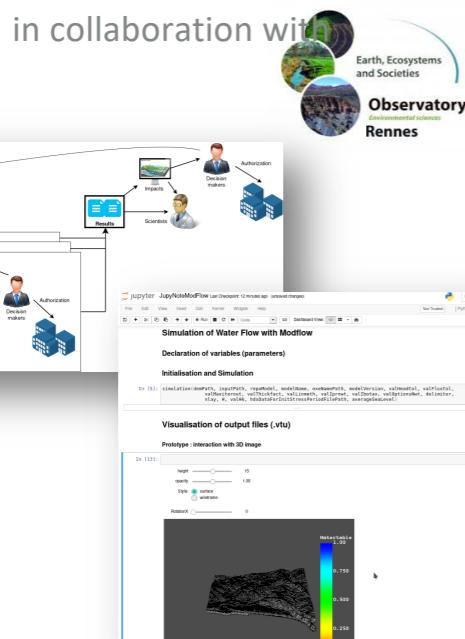
xCapella



Model Execution (M1ICE)
Benoit Combemale, Feb. 2021

Crosscutting Challenges for Various Domains

Towards Virtual Labs From Modeling Environment to Digital Twin



Tradeoff Analysis for
Water Flood Prediction

Design Space Exploration
for Cyber-Physical Systems

High-Performance Computing
for Numerical Analysis

Content of the course

- **Metamodeling: Ecore, OCL and ATL**
- **Hack your own Domain-Specific Languages**
 - **Domain model (EMF Ecore)**
 - **Textual *and* graphical syntaxes (Xtext *and* Sirius)**
 - **Compiler and interpreter (Xtend)**
- **Companion webpage:**

<http://combemale.fr/course/ice1/>

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