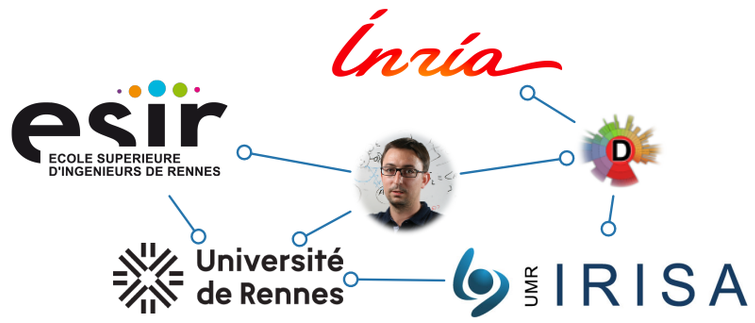
MODEL-DRIVEN *(SOFTWARE)* ENGINEERING

HACK YOUR OWN LANGUAGE!

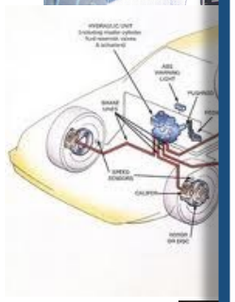
ESIR3 ASE, 2023-2024

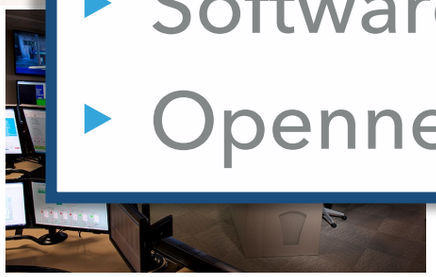
BENOIT COMBEMALE
FULL PROFESSOR, UNIVERSITY OF RENNES, FRANCE

HTTP://COMBEMALE.FR
BENOIT.COMBEMALE@IRISA.FR
@BCOMBEMALE

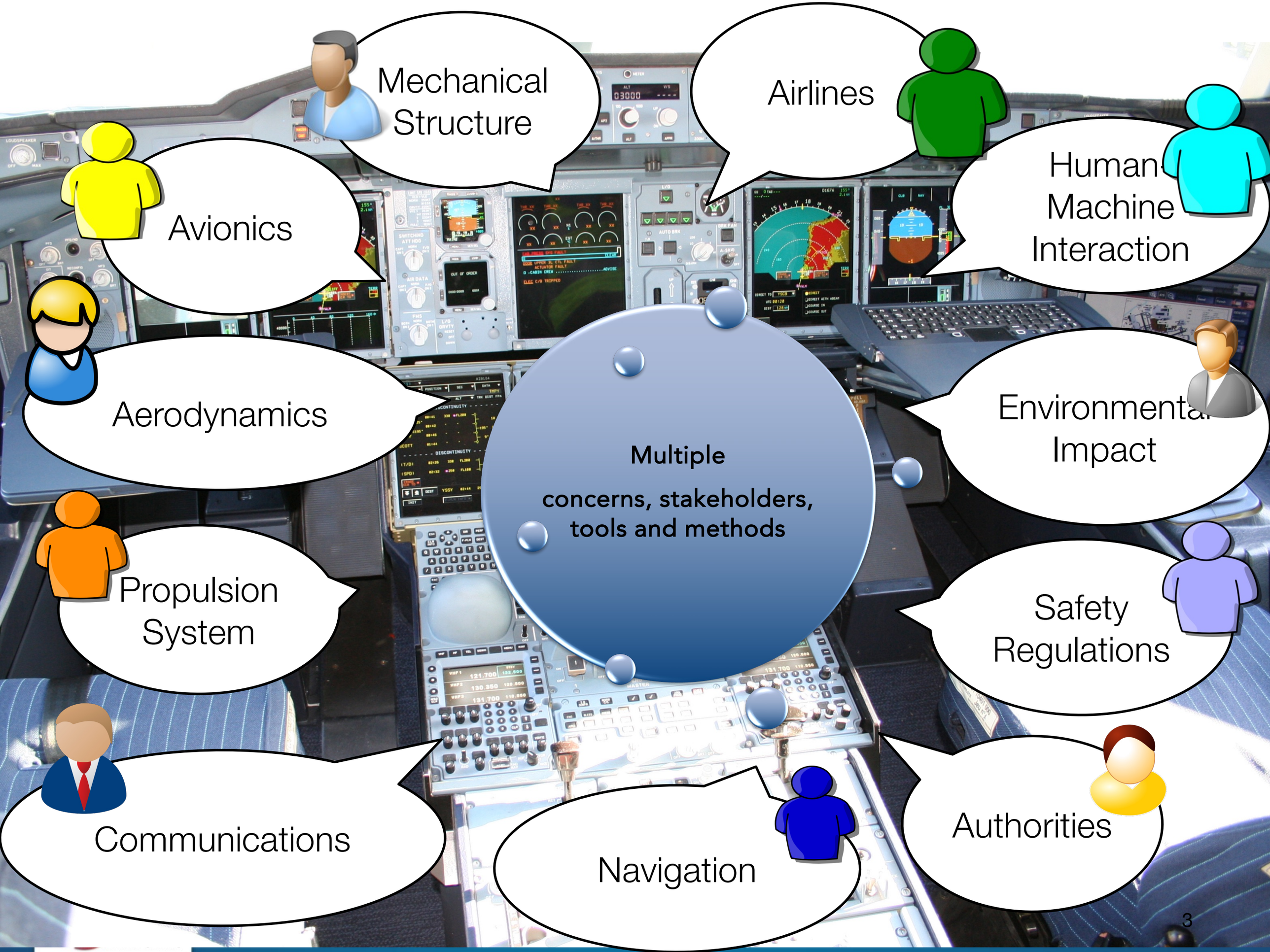


Complex Software-Intensive Systems

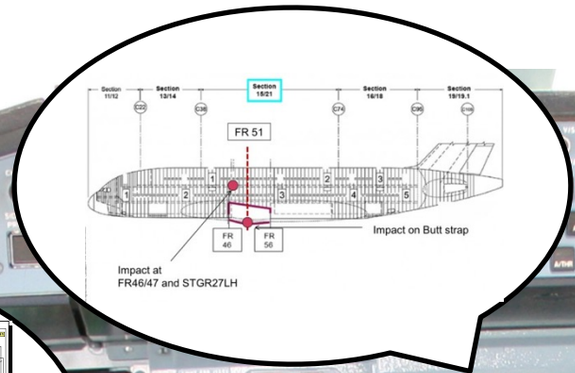


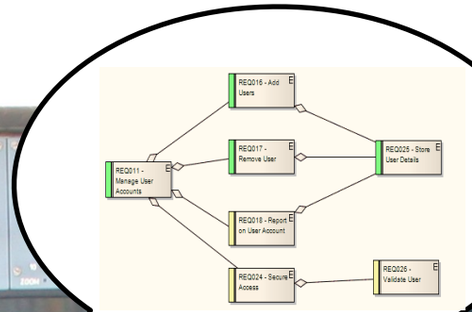


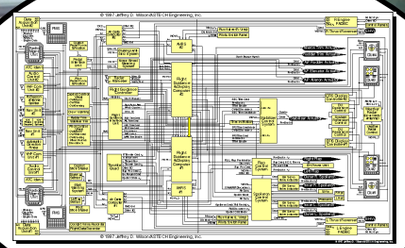
- Multi-engineering approach
- Domain-specific modeling
- High variability and customization
- Software as integration layer
- Openness and dynamicity

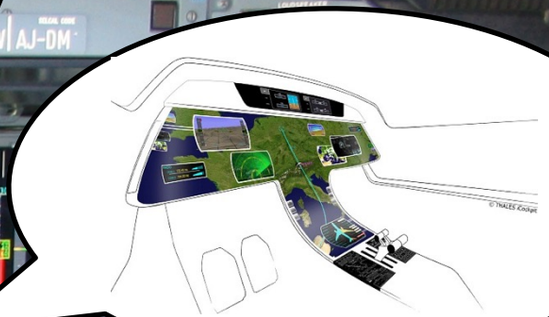

Mechanical Structure
Airlines
Avionics
Human-Machine Interaction
Aerodynamics
Multiple
concerns, stakeholders, tools and methods
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorities

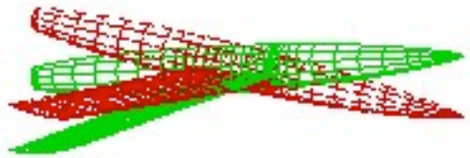
Heterogeneous Modeling

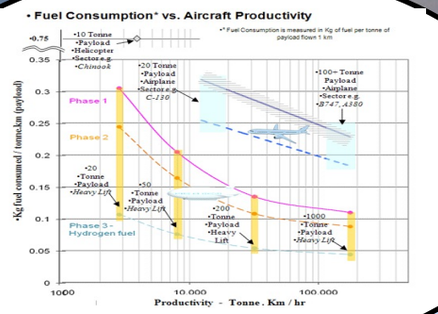


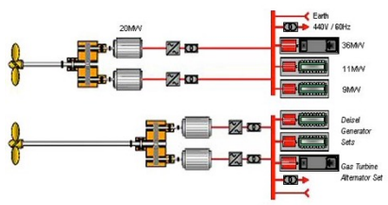


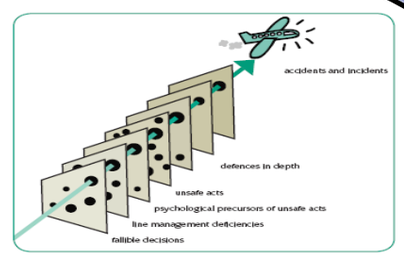


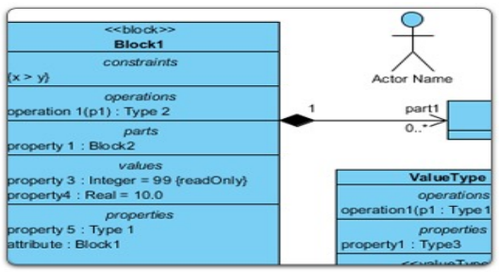


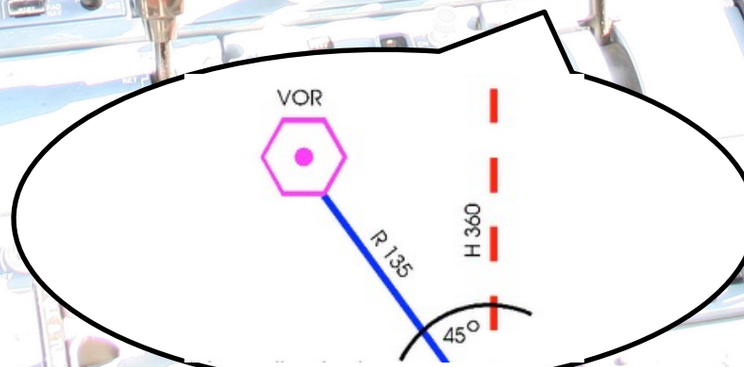


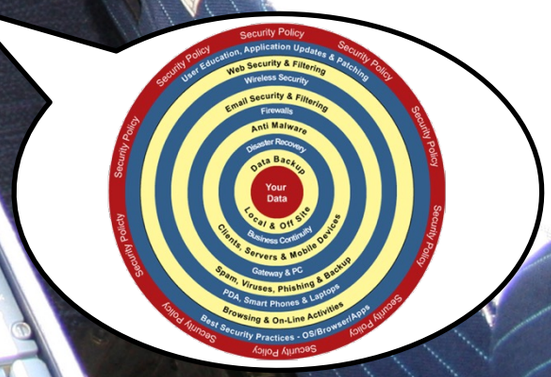












Software Modeling: Why Should I Care?

- 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?

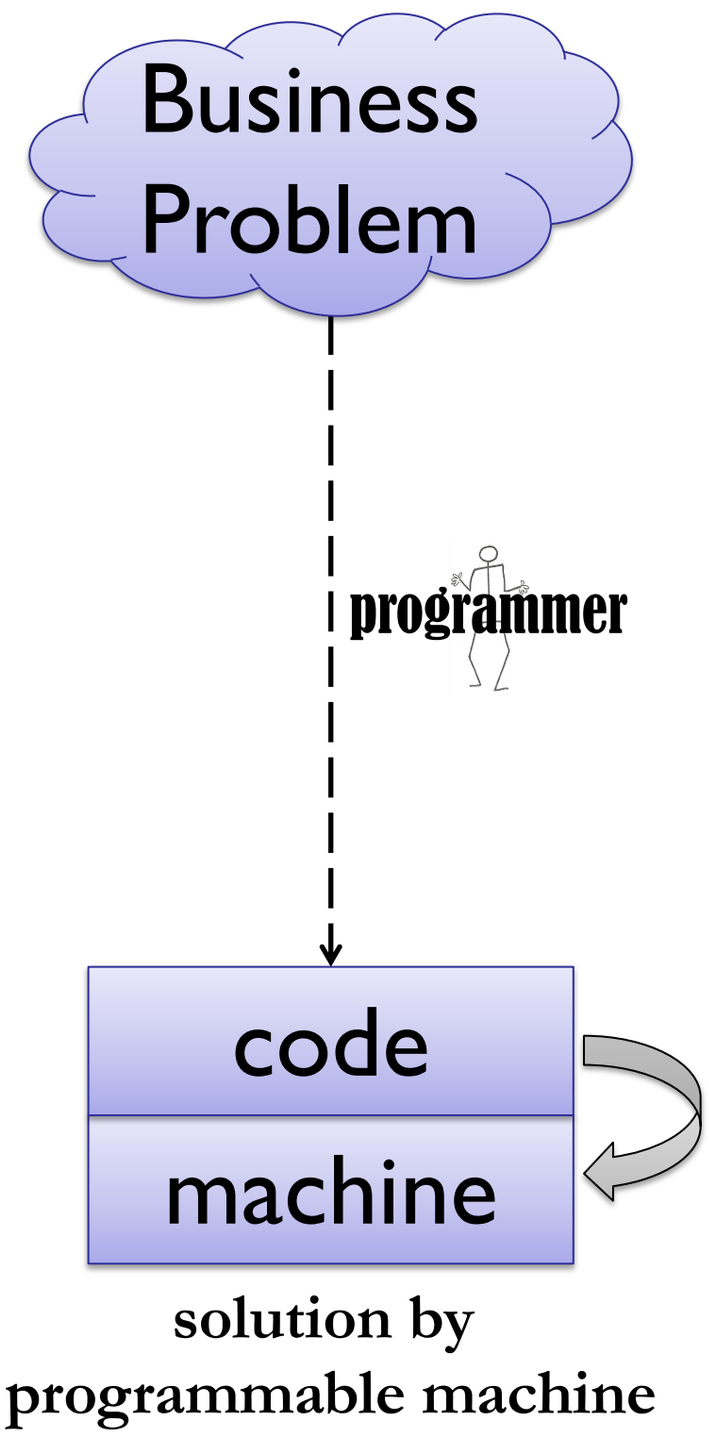
- Pour réfléchir :
  - représentation abstraite
  - séparation des préoccupations
- Pour communiquer :
  - représentation graphique
  - génération de documentation
- Pour automatiser le développement :
  - génération de code
  - application de patrons
  - migration
- Pour vérifier :
  - validation et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

Model and Reality in Software

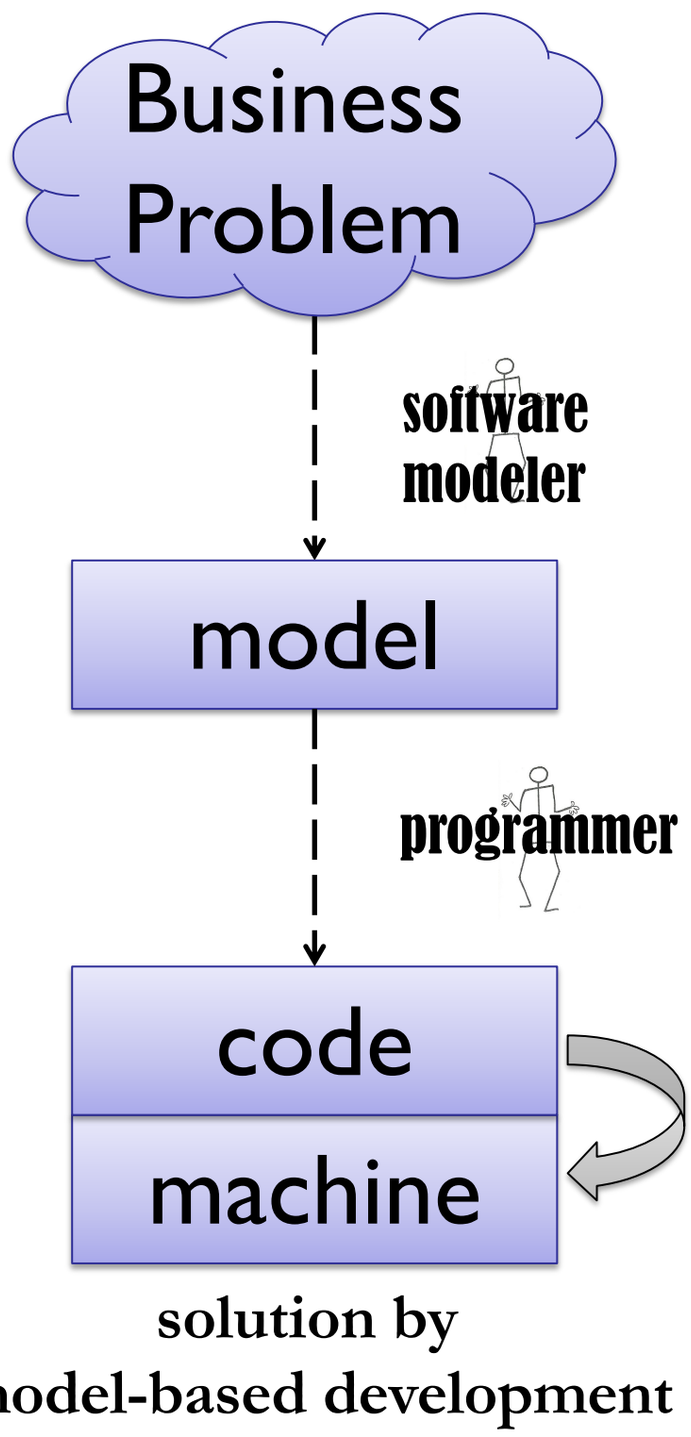
- **Sun Tse:** *“Do not take the map for the reality”*
- **William James:** *“The concept 'dog' does not bite”*
- **Magritte:**

- Software Models: from contemplative to productive

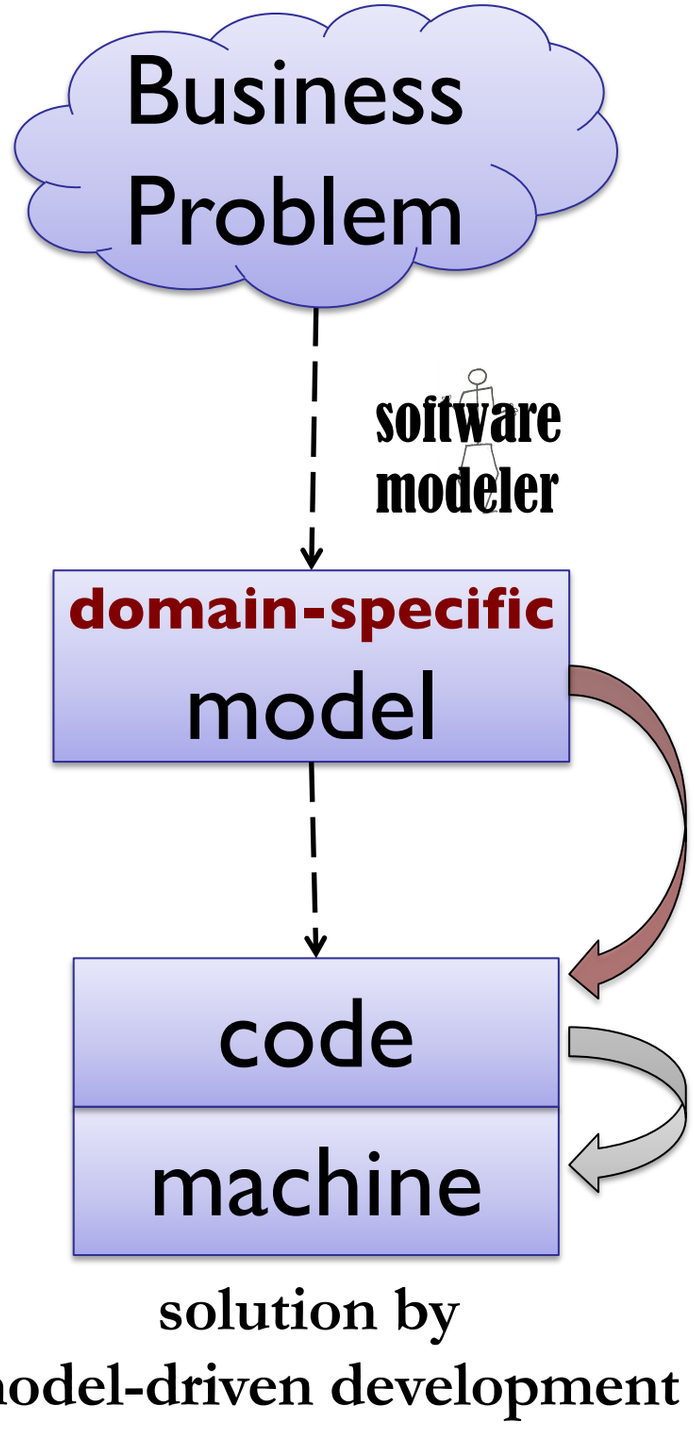
Evolution in Software Modeling



solution by programmable machine



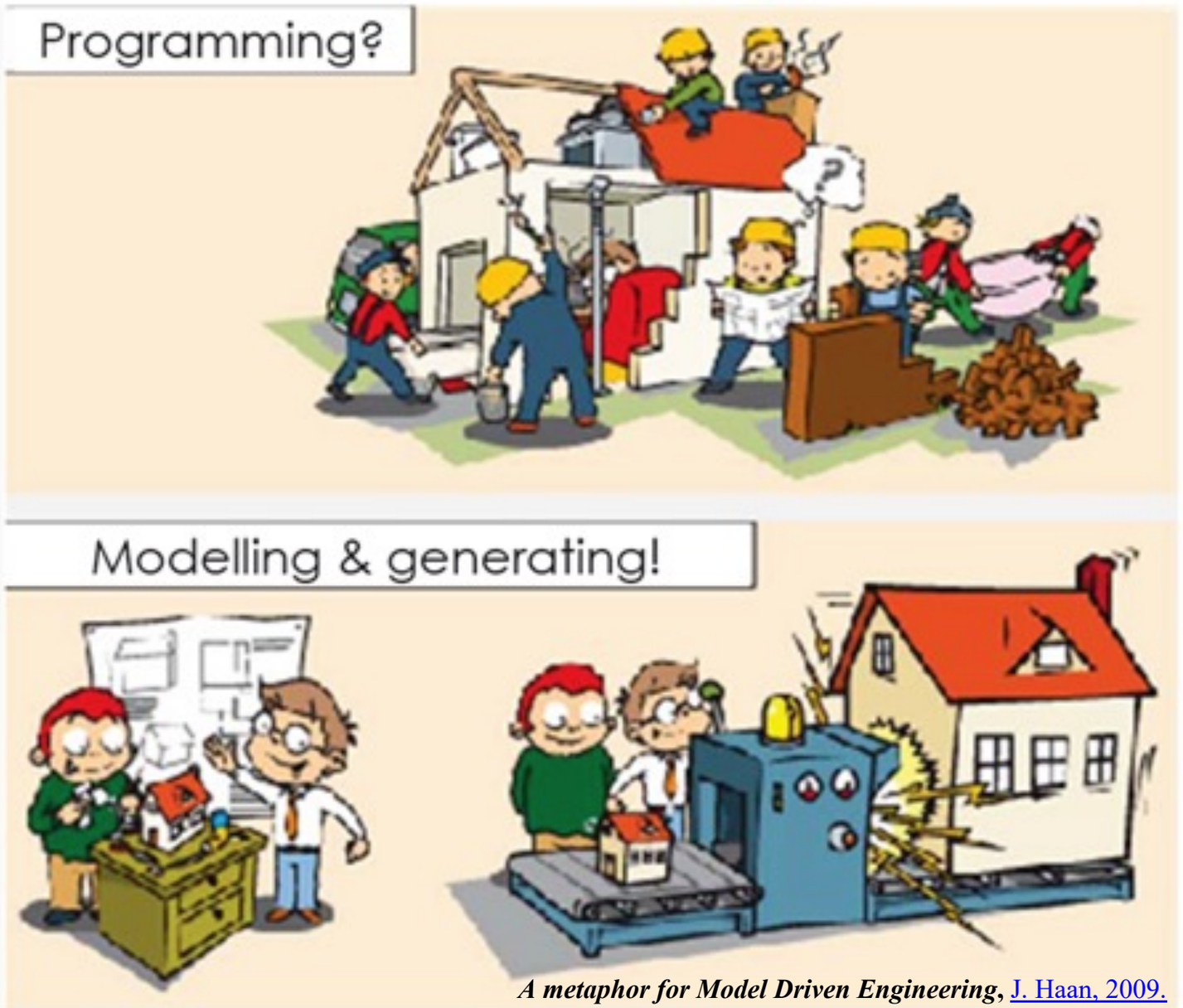
solution by model-based development



solution by model-driven development

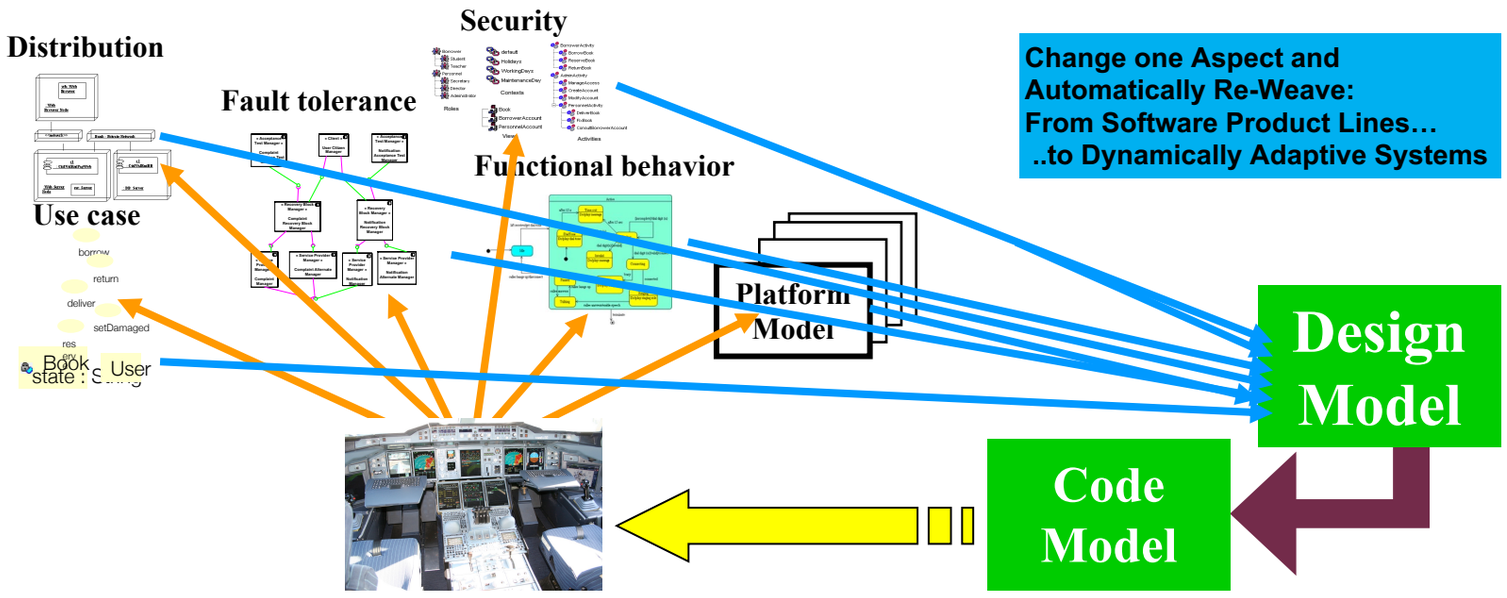
From Gregor Engels (Universität Paderborn, Germany)
Panel "When will Code become Irrelevant?", MoDELS 2011.

Towards Model-Driven Engineering (MDE)



***A metaphor for Model Driven Engineering*, J. Haan, 2009.**

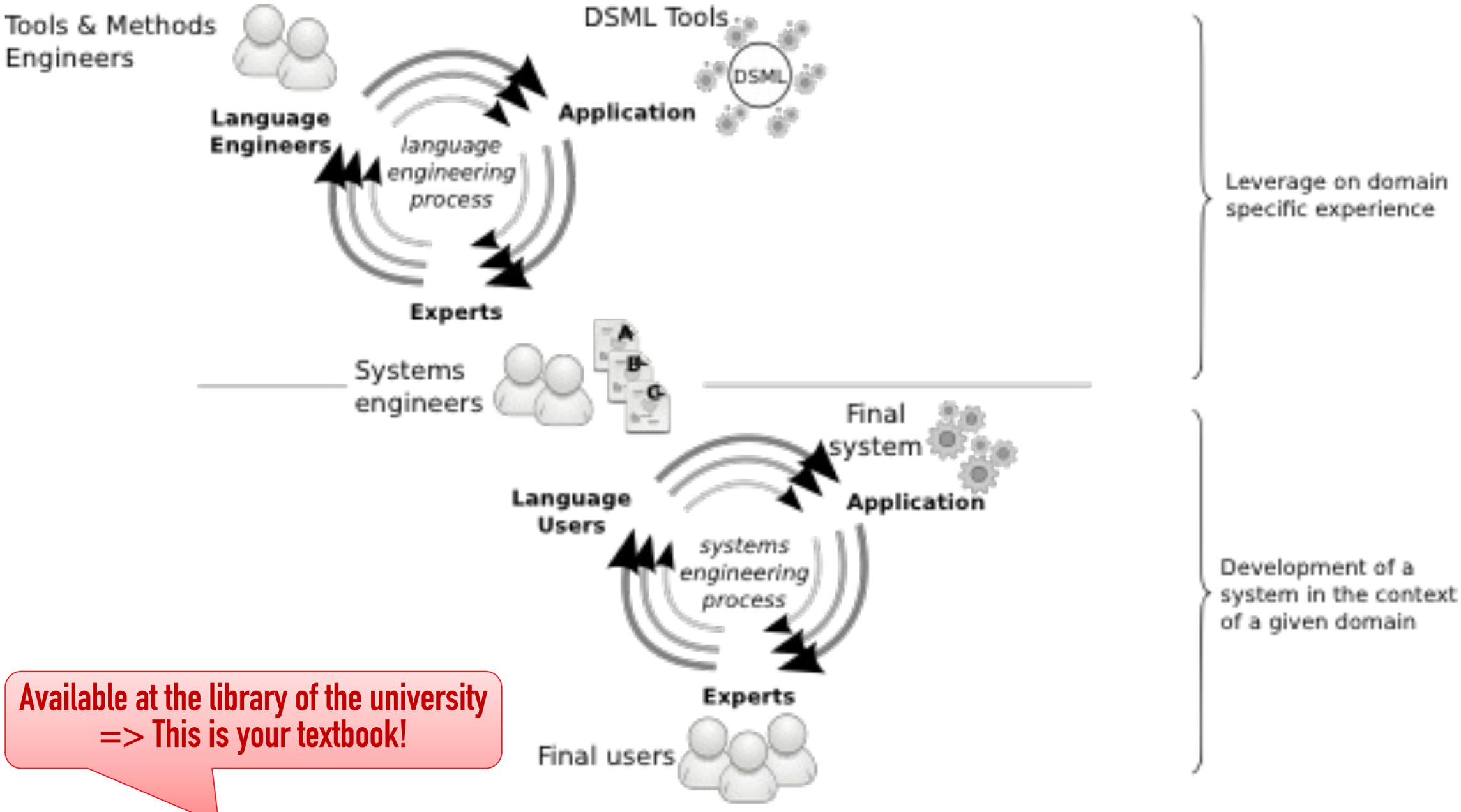
Model-Driven Engineering (MDE)



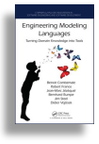
"Perhaps surprisingly, the majority of MDE examples in our study followed domain-specific modeling paradigms"

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)

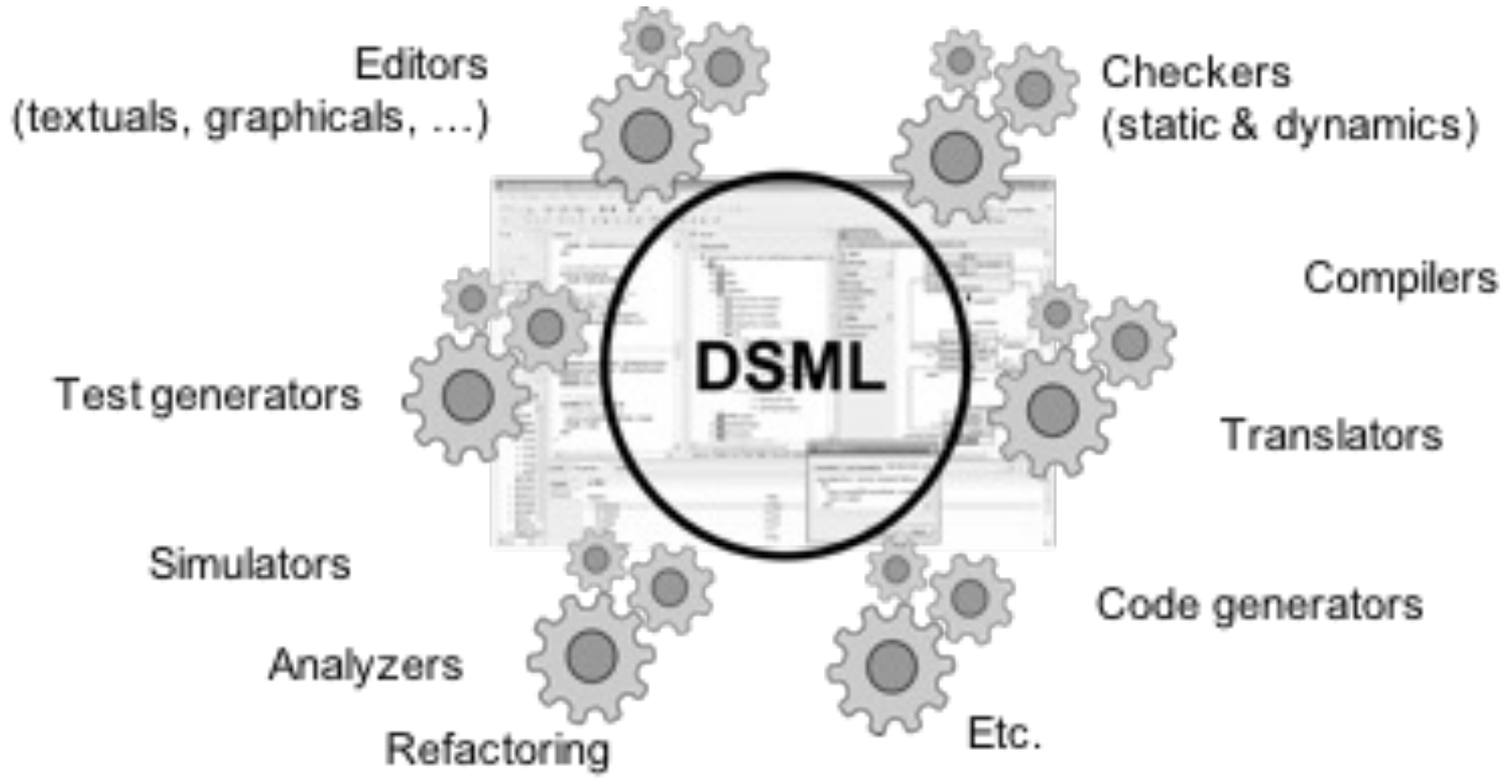


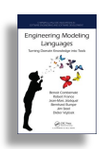
Available at the library of the university
=> This is your textbook!



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
  - 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

http://gemoc.org/studio

also

http://eclipse.org/gemoc

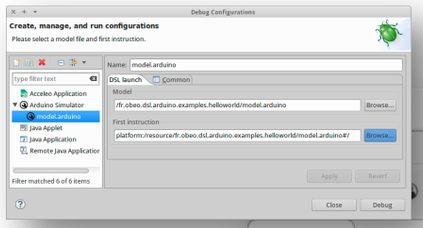
Language Workbench

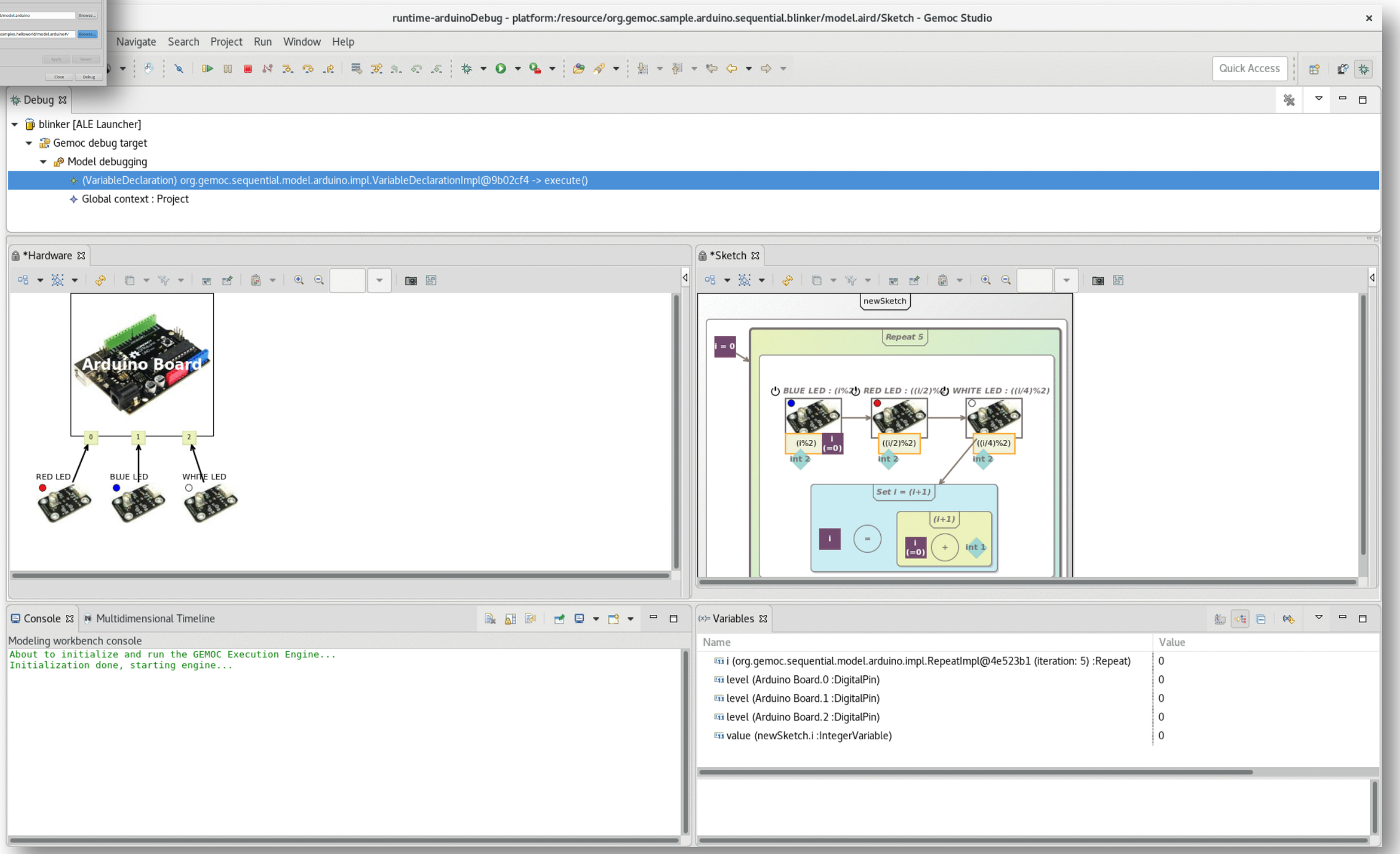
Design and integrate your executable DSMLs

Modeling Workbench

Edit, simulate and animate your heterogeneous models

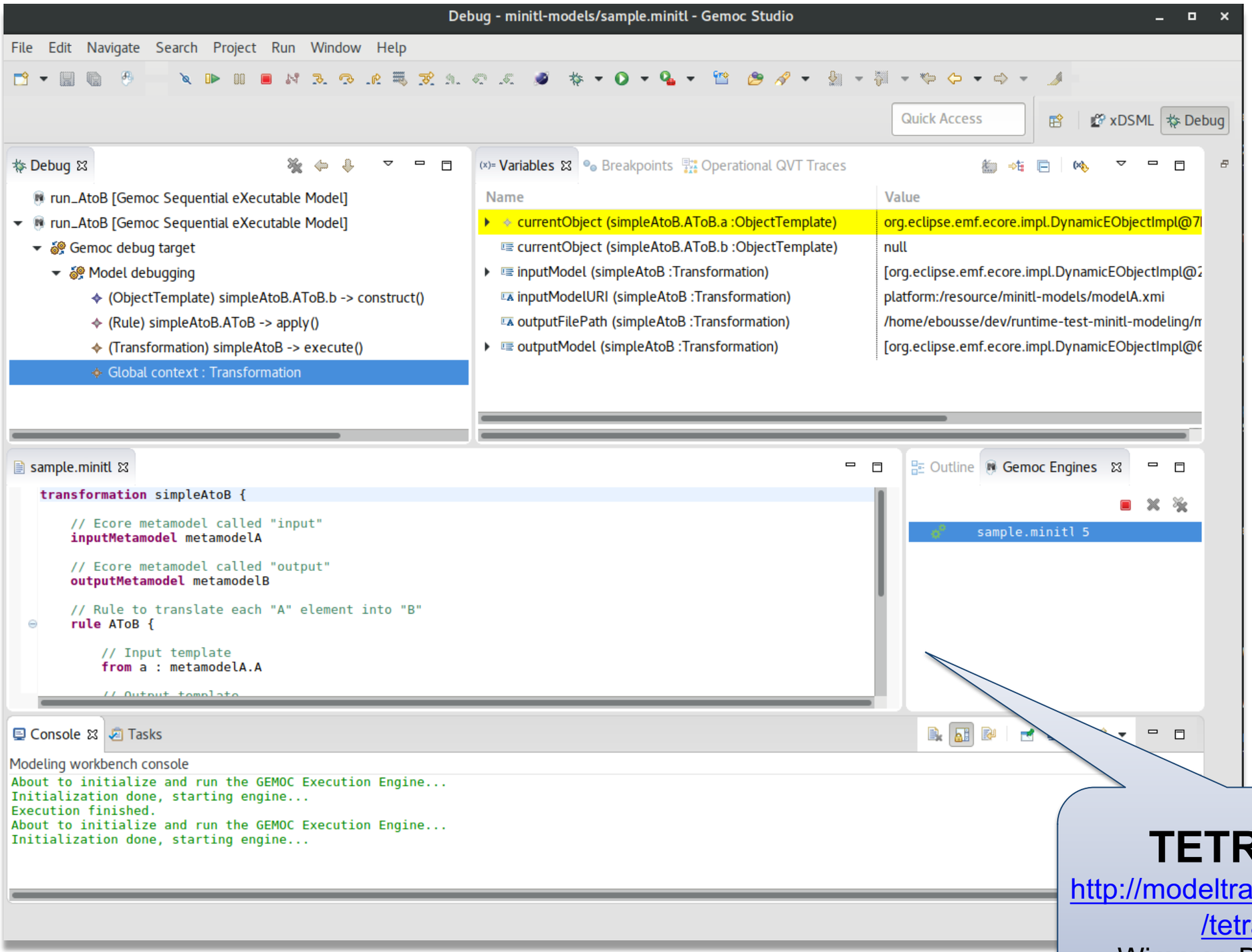
Arduino Designer





https://github.com/gemoc/arduinomodeling

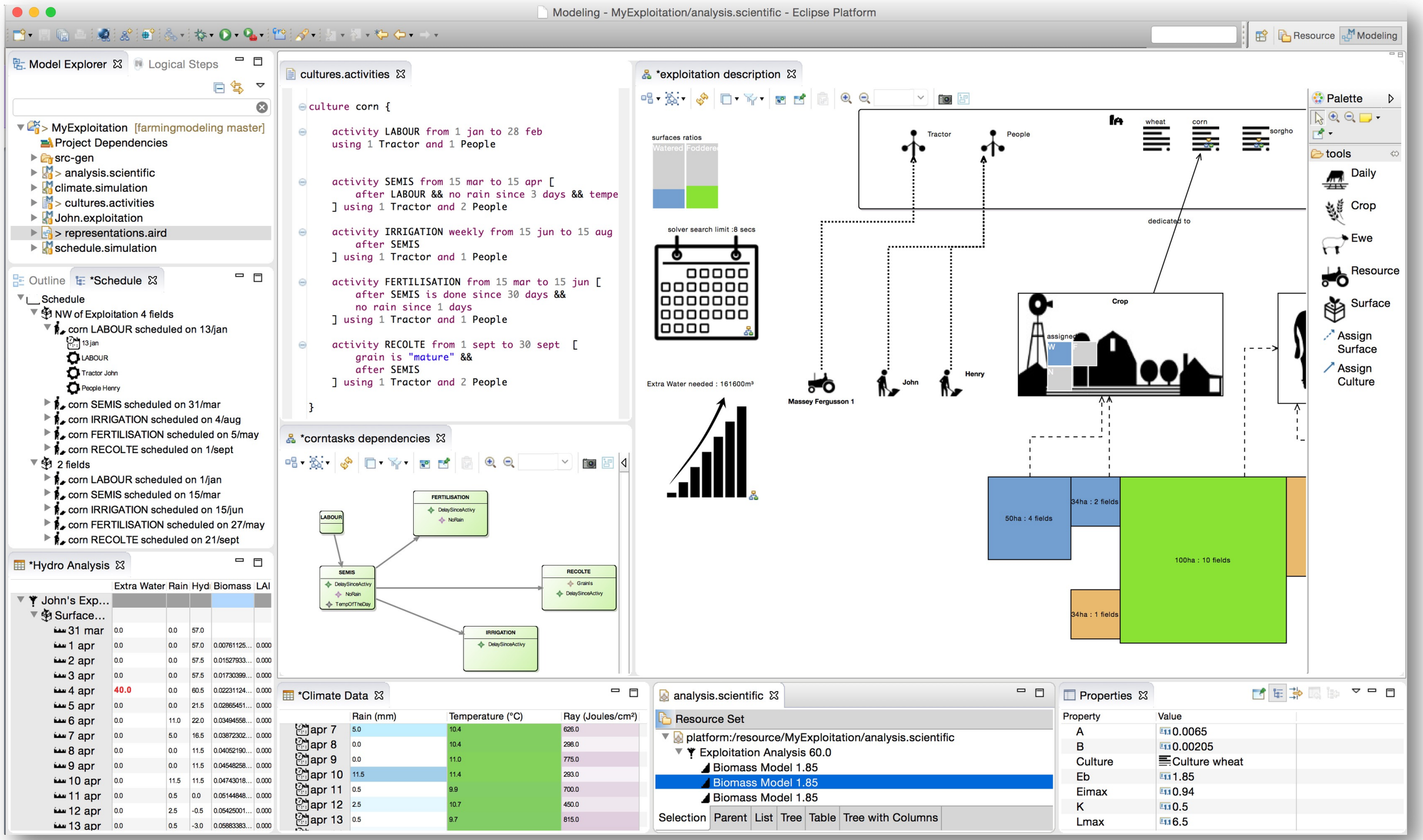
Transformation Lg Debugger



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

https://github.com/tetrabox/minitl

Farming System Model



https://github.com/gemoc/farmingmodeling

Université de Rennes

MDE Introduction (ESIR3 SI, ASE)

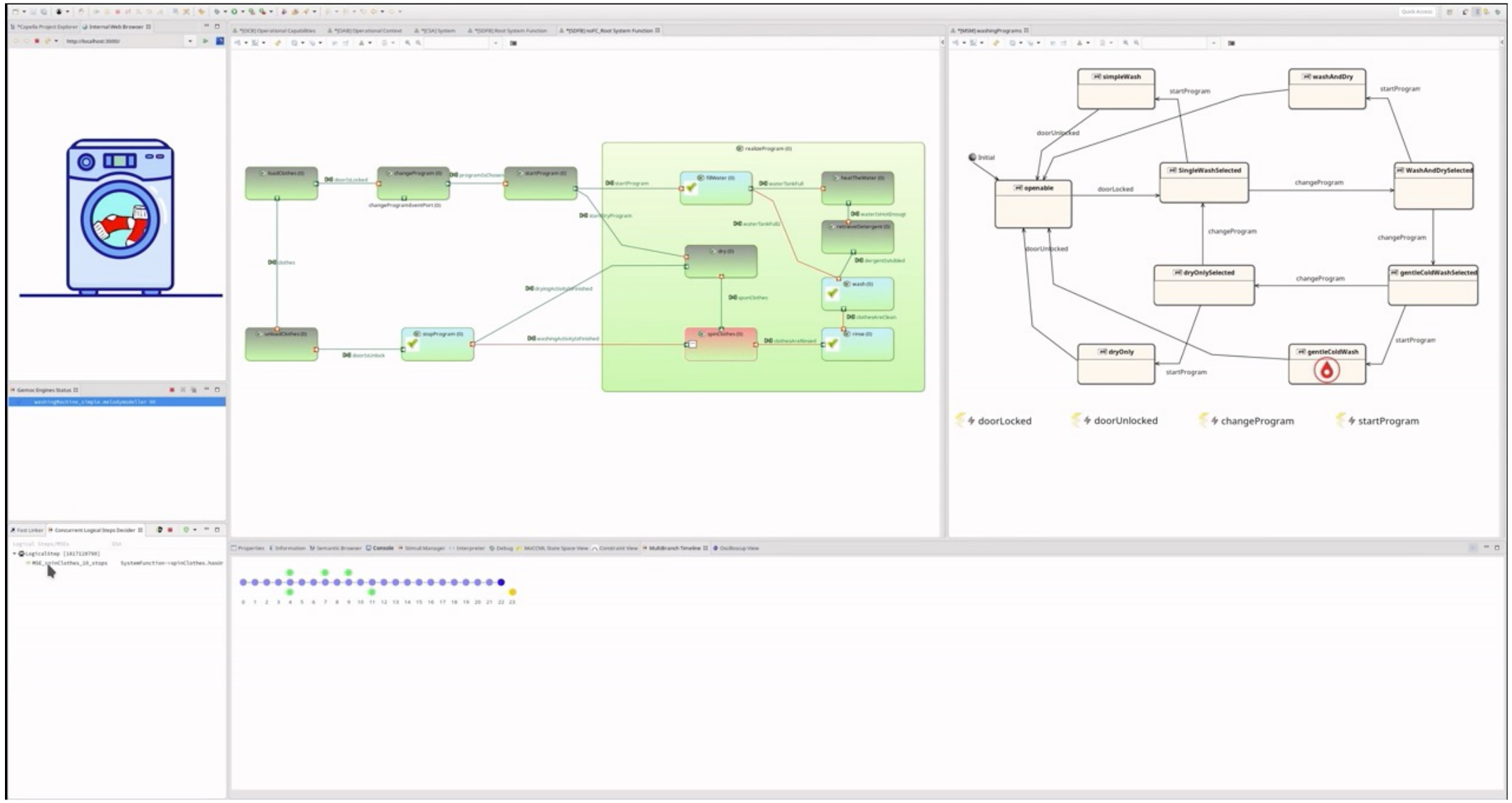
Benoit Combemale, Oct. 2023

UMR IRISA

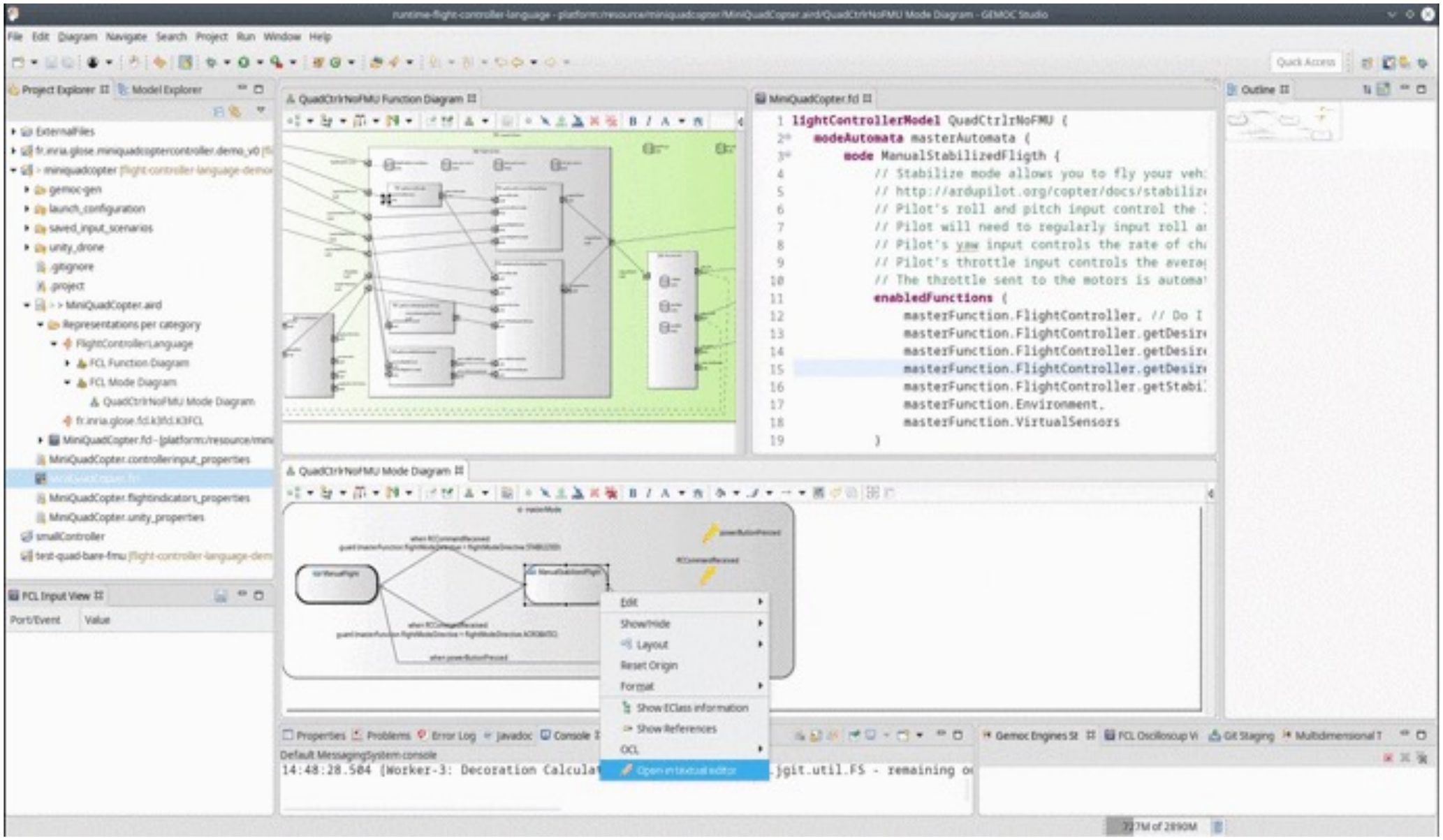
UML Activity Diagram

https://github.com/gemoc/activitydiagram

xCapella


simpleWash
washAndDry
startProgram
startProgram
doorUnlocked
Initial
openable
doorLocked
SingleWashSelected
changeProgram
WashAndDrySelected
changeProgram
changeProgram
doorUnlocked
dryOnlySelected
changeProgram
gentleColdWashSelected
dryOnly
gentleColdWash
startProgram
startProgram
doorLocked
doorUnlocked
changeProgram
startProgram

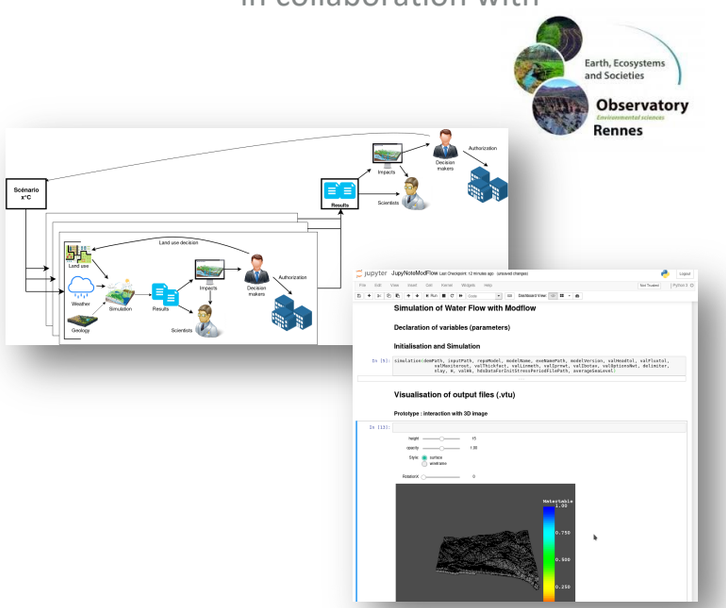
FCL


File Edit Diagram Navigate Search Project Run Window Help
Project Explorer
Model Explorer
QuadCtrlrNoFMU Function Diagram
MiniQuadCopter.fcl
1 lightControllerModel QuadCtrlrNoFMU {
2 modeAutomata masterAutomata {
3 mode ManualStabilizedFligth {
4 // Stabilize mode allows you to fly your veh
5 // http://ardupilot.org/copter/docs/stabiliz
6 // Pilot's roll and pitch input control the
7 // Pilot will need to regularly input roll a
8 // Pilot's yaw input controls the rate of ch
9 // Pilot's throttle input controls the avera
10 // The throttle sent to the motors is automa
11 enabledFunctions {
12 masterFunction.FlightController, // Do I
13 masterFunction.FlightController.getDesir
14 masterFunction.FlightController.getDesir
15 masterFunction.FlightController.getDesir
16 masterFunction.FlightController.getStabi
17 masterFunction.Environment,
18 masterFunction.VirtualSensors
19 }
Outline
QuadCtrlrNoFMU Mode Diagram
Edit
Show/Hide
Layout
Reset Origin
Format
Show EClass information
Show References
OCL
Open in textual editor
PCL Input View
Port/Event
Value
Properties
Problems
Error Log
Javadoc
Console
Default MessagingSystem-console
Gemoc Engines St
FCL Oscilloscop Vi
Git Staging
Multidimensional T

Crosscutting Challenges for Various Domains

Towards Virtual Labs
From Modeling Environment to Digital Twin

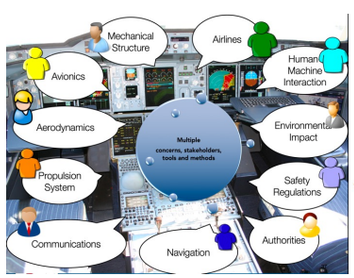
in collaboration with

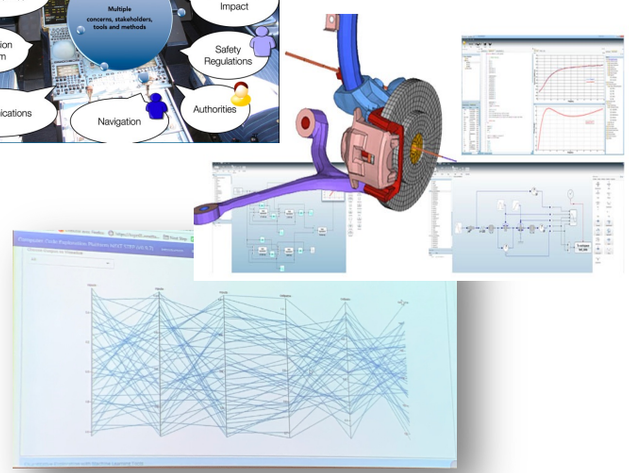


Tradeoff Analysis for Water Flood Prediction

in collaboration with

SAFRAN

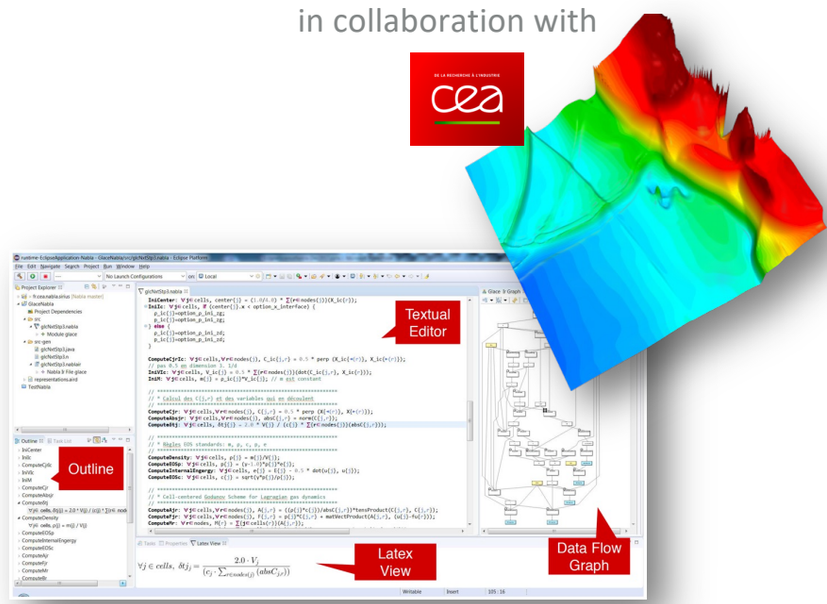




Design Space Exploration for Cyber-Physical Systems

in collaboration with

cea



High-Performance Computing for Numerical Analysis

Université de Rennes

UMR IRISA

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