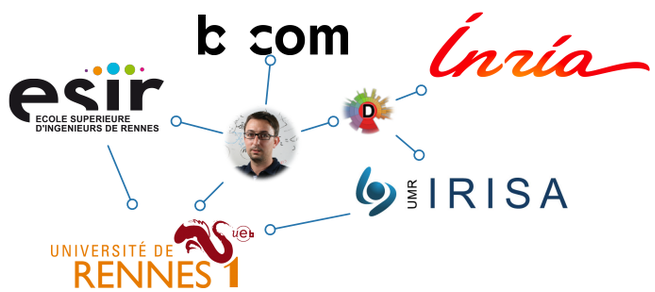
MODEL-DRIVEN *(SOFTWARE)* ENGINEERING

HACK YOUR OWN LANGUAGE!

MASTER 1 ICE, 2020-2021

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

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



WHO WE ARE?

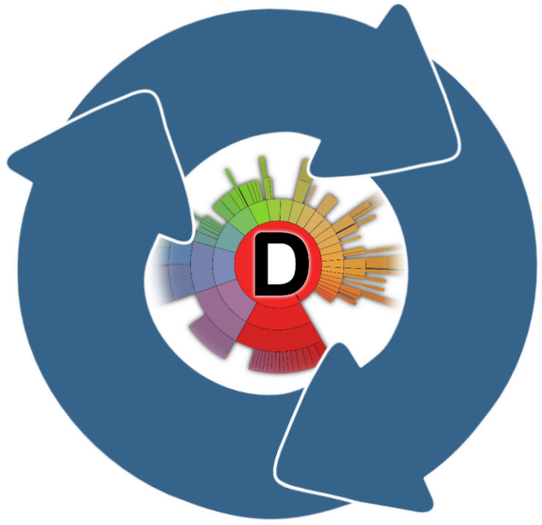
The DiverSE team

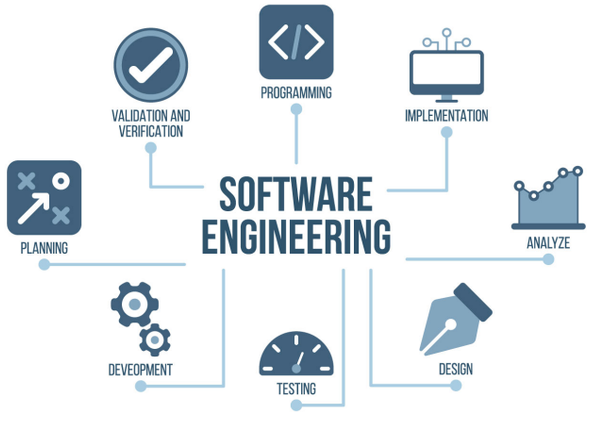
- Inria/IRISA project-team in **Software Engineering** (SE)
- Strong background in **Model-Driven** *software/systems* **Engineering** (MDE)
- Software architecture, software variability, software testing, software languages, simulation, resilience eng.
- Applied to smart, heterogeneous, and distributed 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

DiverSE
Diversity-Centric
Software Engineering

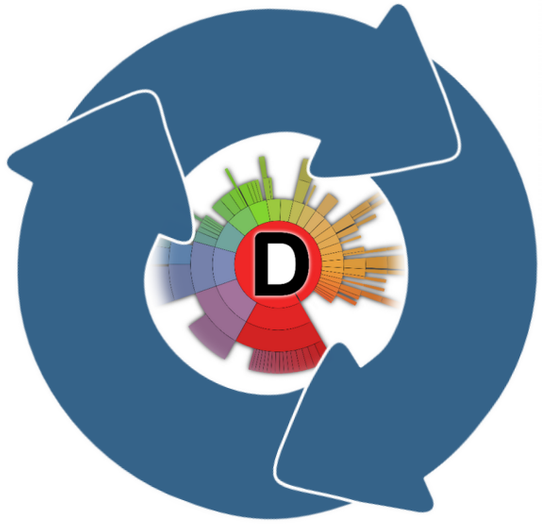
A Software Engineering Group





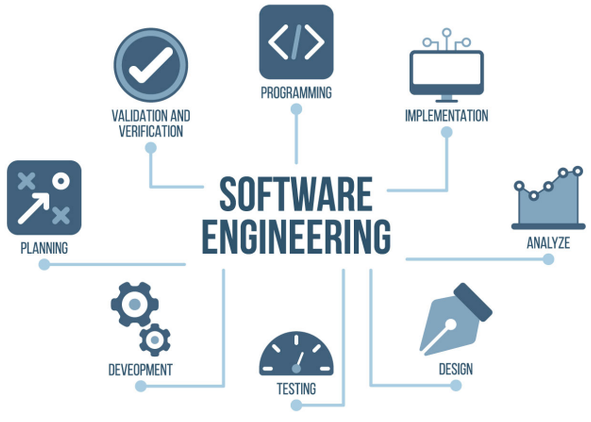
The DiverSE team

A Software Engineering Group



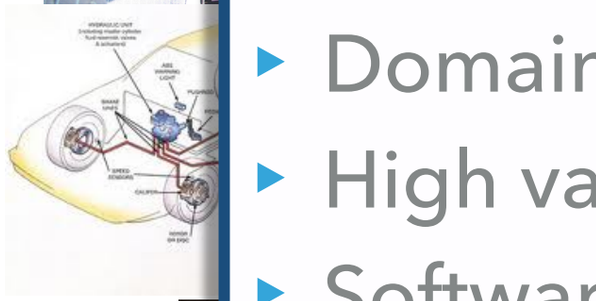
Diversity of…

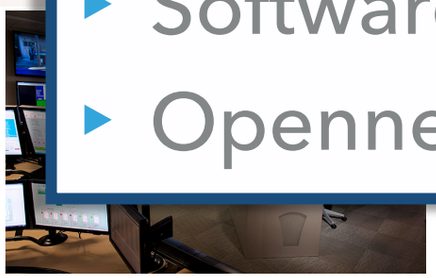
- stakeholders
- concerns
- configurations
- platforms
- environments
- requirements…



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

Complex Software-Intensive Systems





- 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

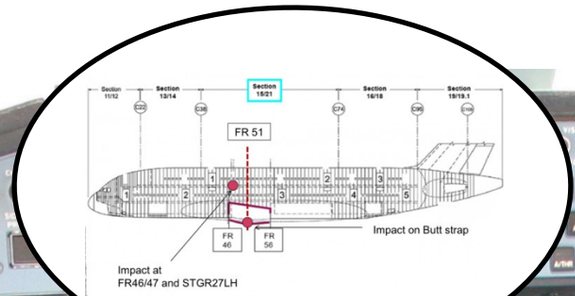
Safety Regulations

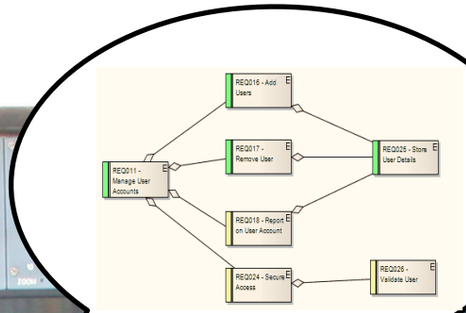
Communications

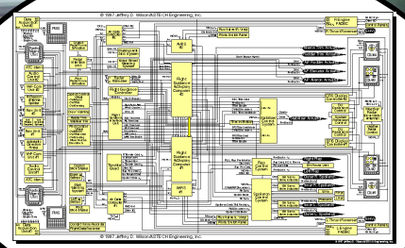
Navigation

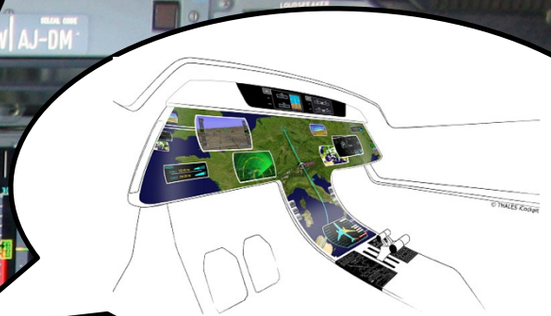
Authorities

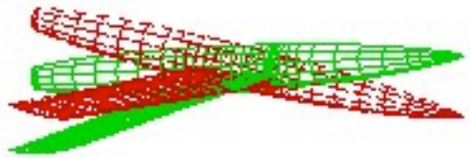
Heterogeneous Modeling

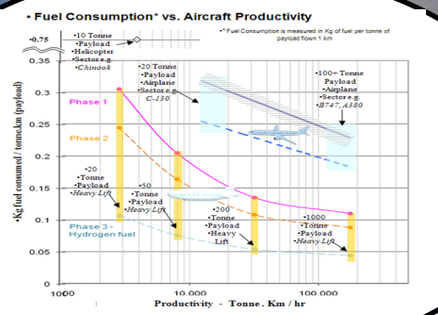


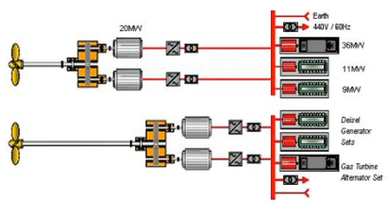


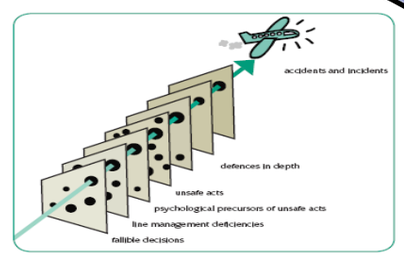


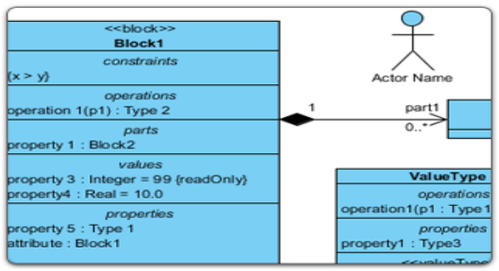


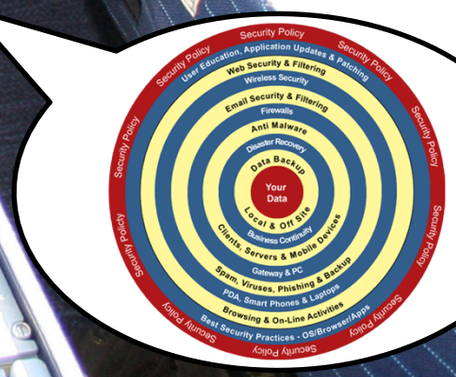












Software Modeling: Why Should I Care?

- 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?

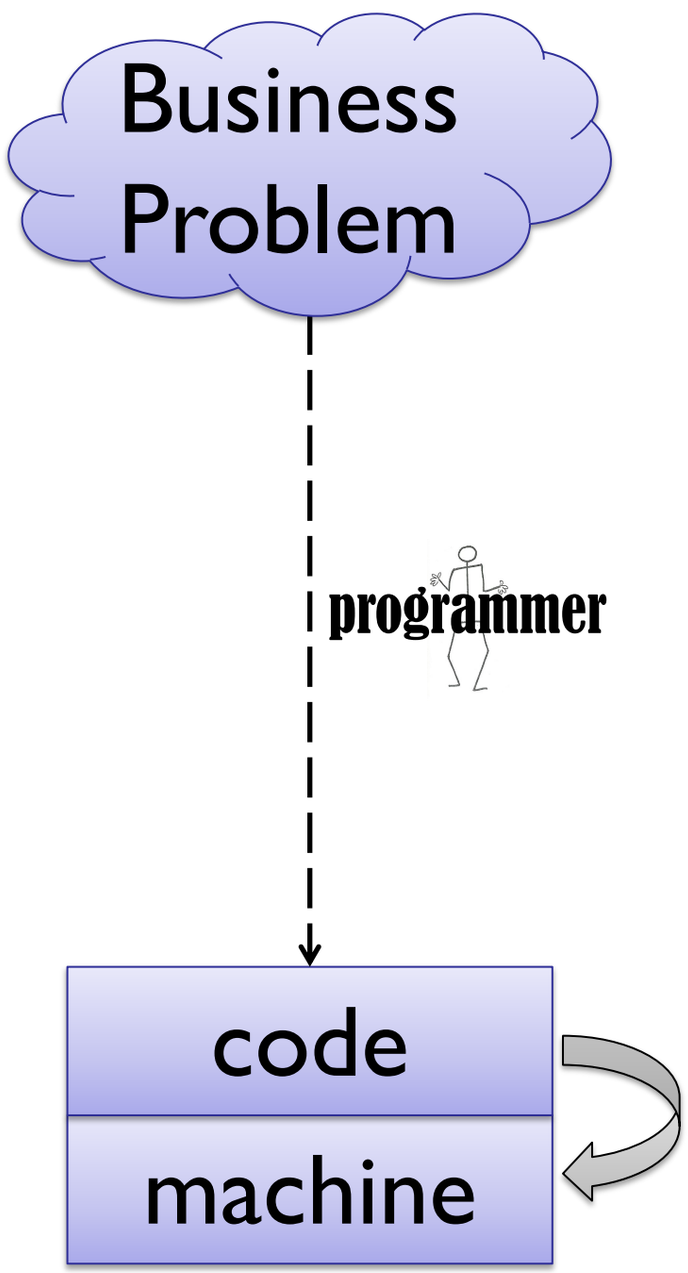
- Pour réfléchir :
  - représentation abstraite
  - séparation des préoccupations
- Pour communiquer :
  - représentation graphique
  - génération de documentation
- Pour automatiser le développement :
  - génération de code
  - application de patrons
  - migration
- Pour vérifier :
  - validation et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

Model and Reality in Software

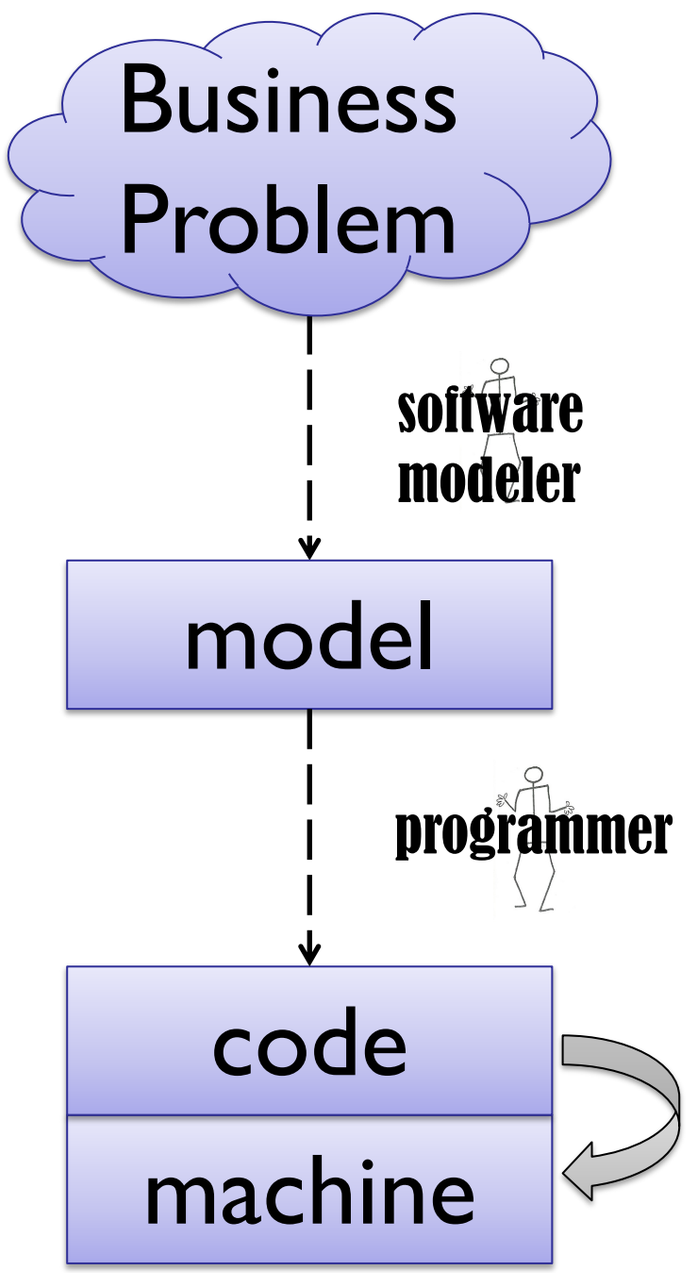
- **Sun Tse:** *"Do not take the map for the reality"*
- **William James:** *"The concept 'dog' does not bite"*
- **Magritte:**

- Software Models: from contemplative to productive

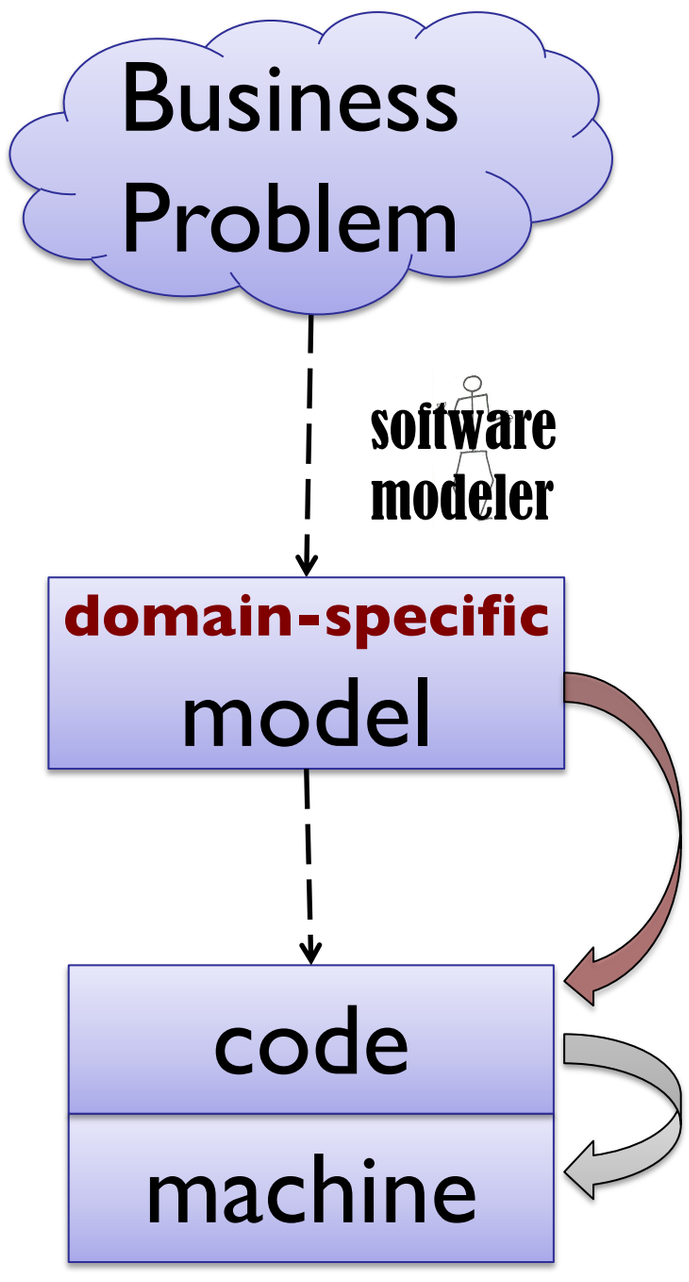
Evolution in Software Modeling



solution by programmable machine



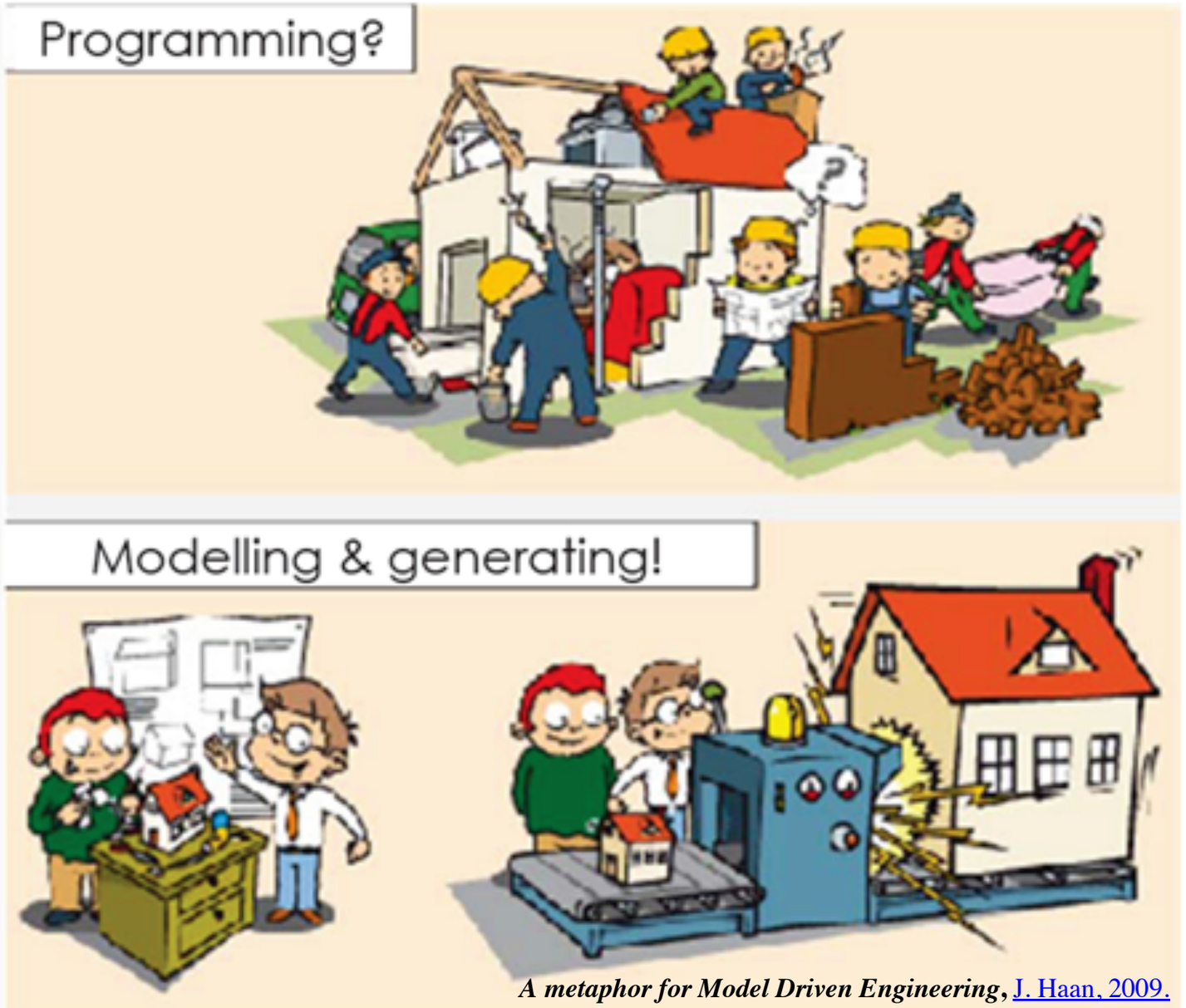
solution by model-based development



solution by model-driven development

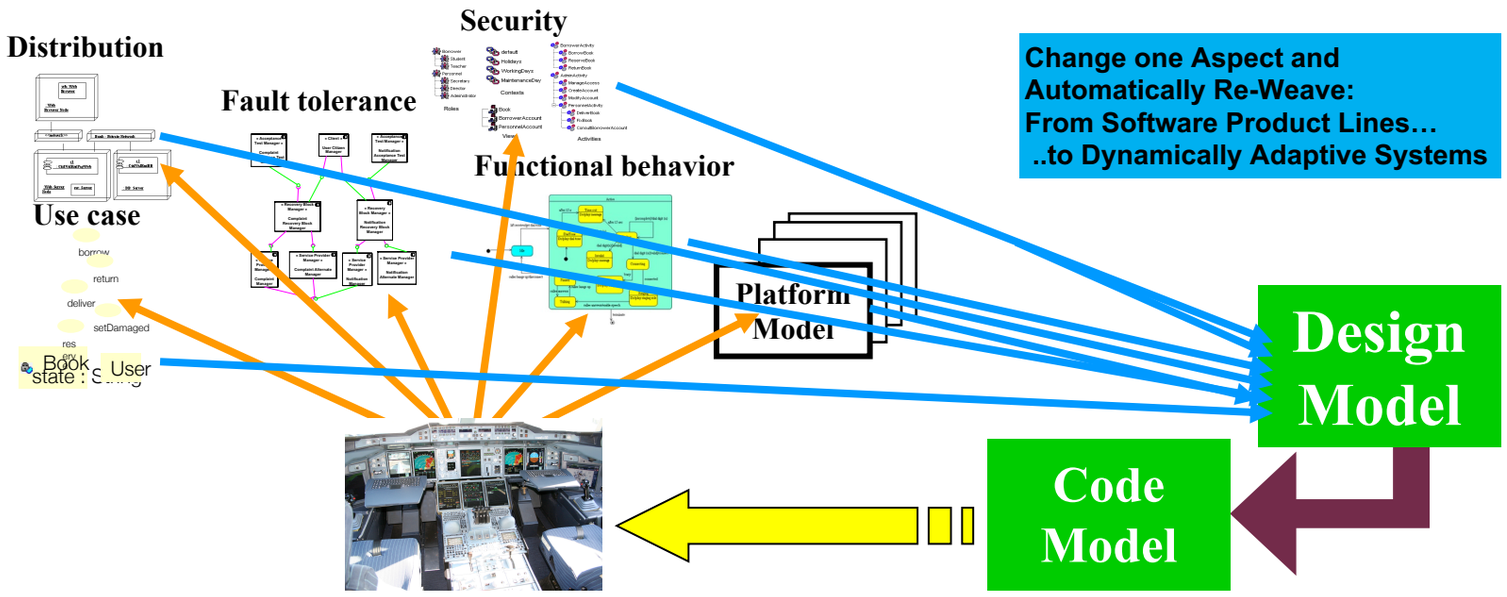
From Gregor Engels (Universität Paderborn, Germany)
Panel "When will Code become Irrelevant?", MoDELS 2011.

Towards Model-Driven Engineering (MDE)



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

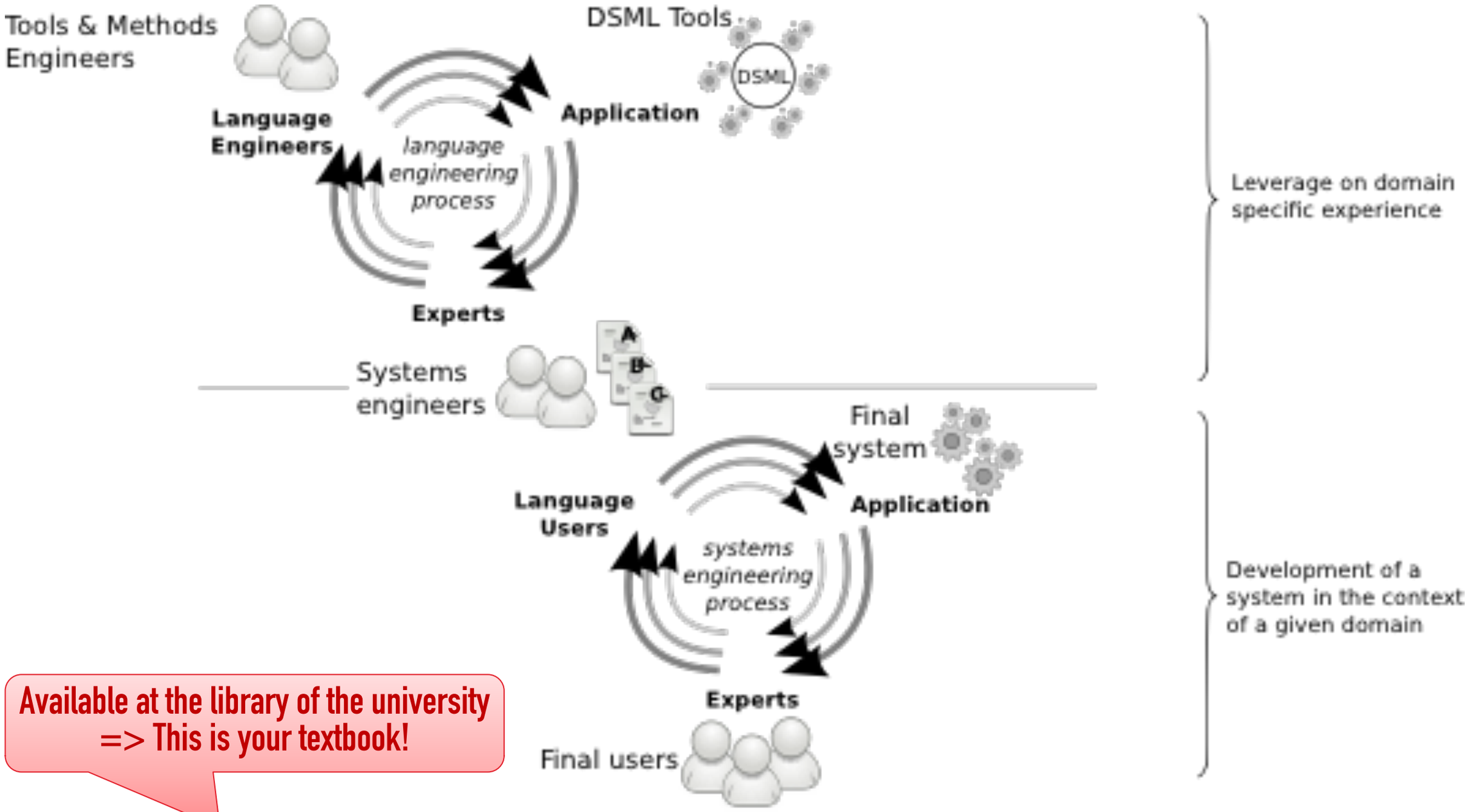
Model-Driven Engineering (MDE)



"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.

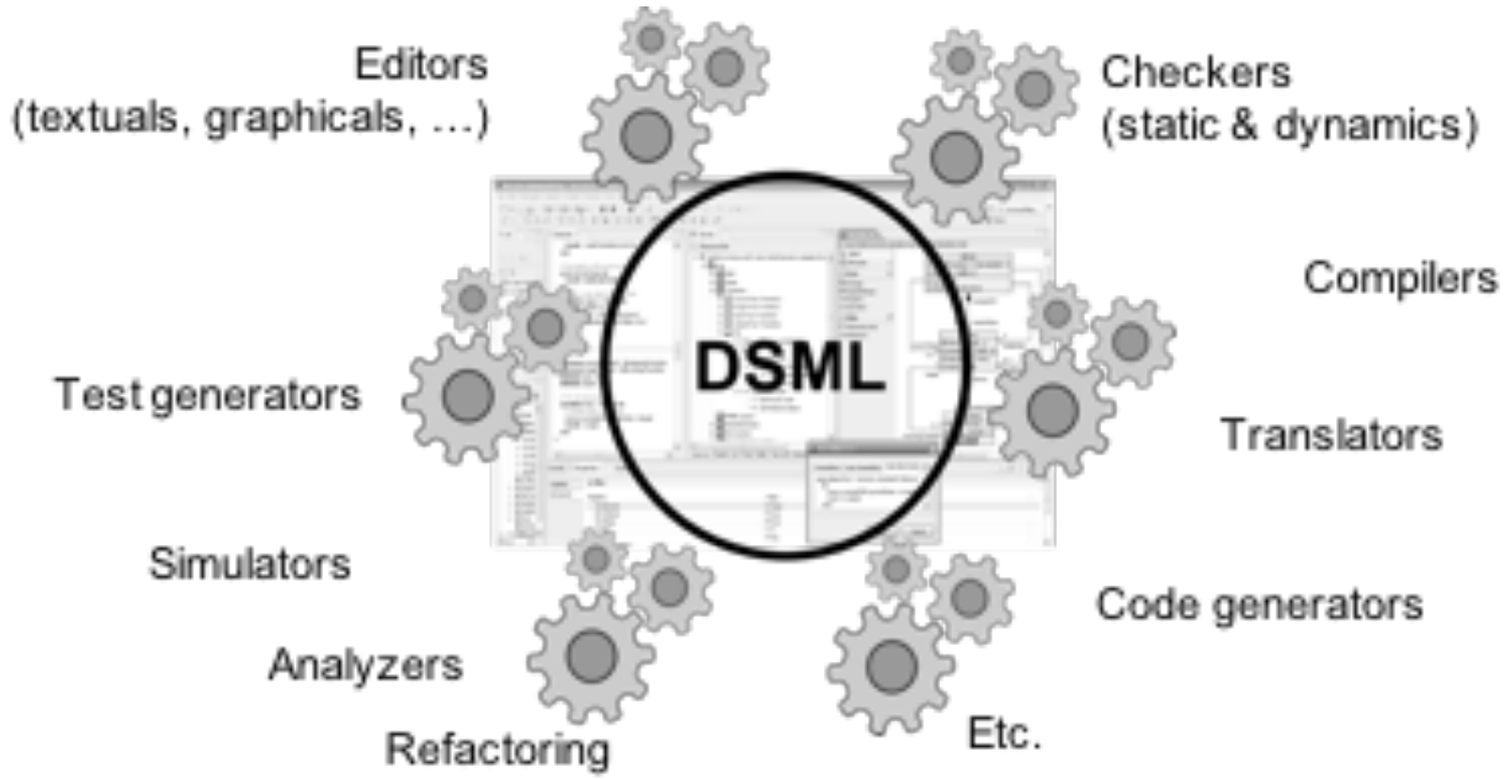
Model-Driven Engineering (MDE)



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

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

http://gemoc.org/studio

also

http://eclipse.org/gemoc

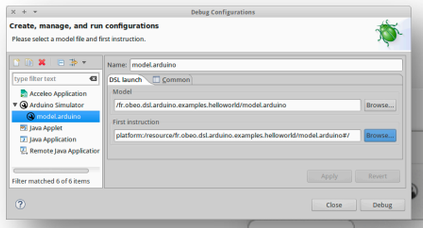
Language Workbench

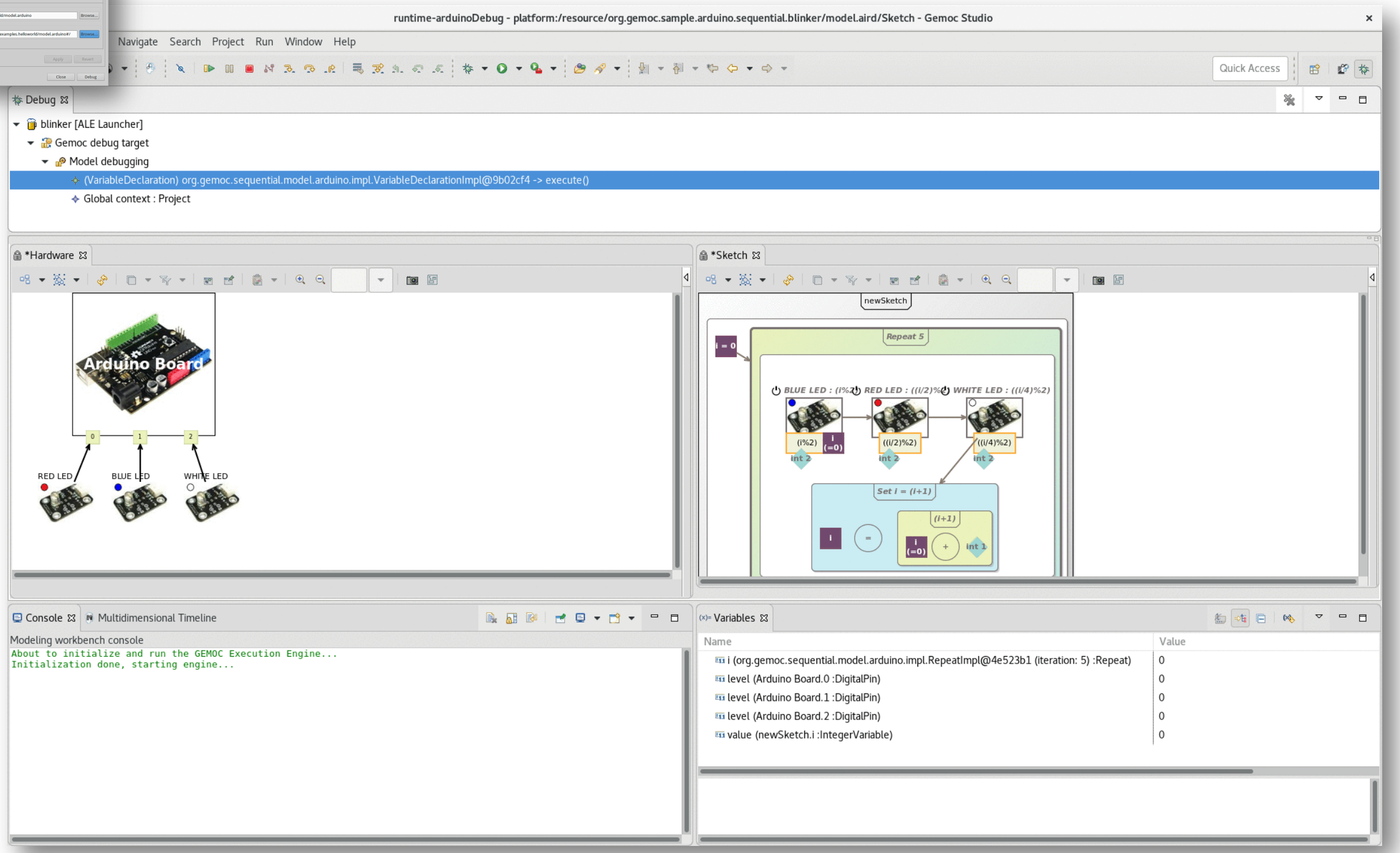
Design and integrate your executable DSMLs

Modeling Workbench

Edit, simulate and animate your heterogeneous models

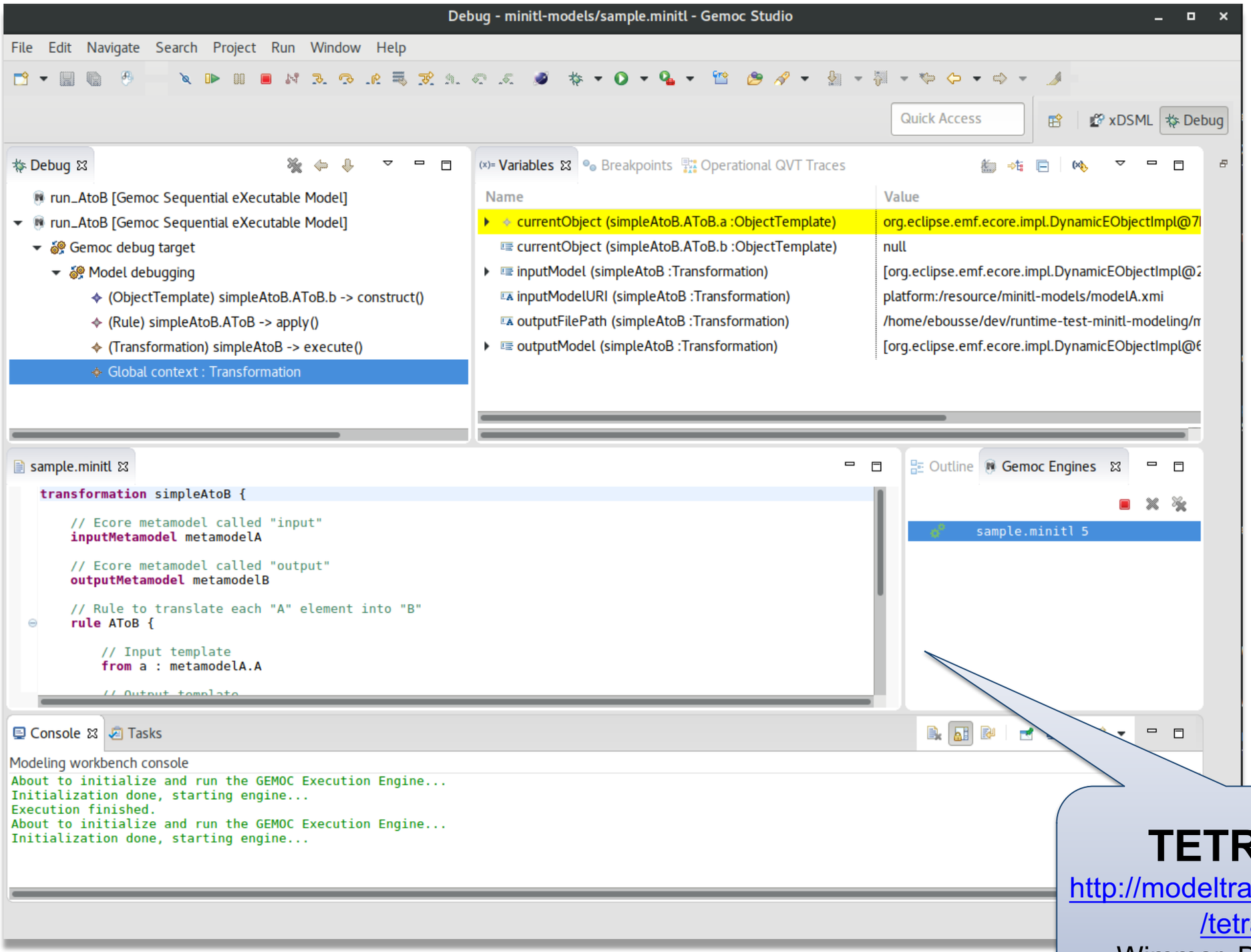
Arduino Designer





https://github.com/gemoc/arduinomodeling

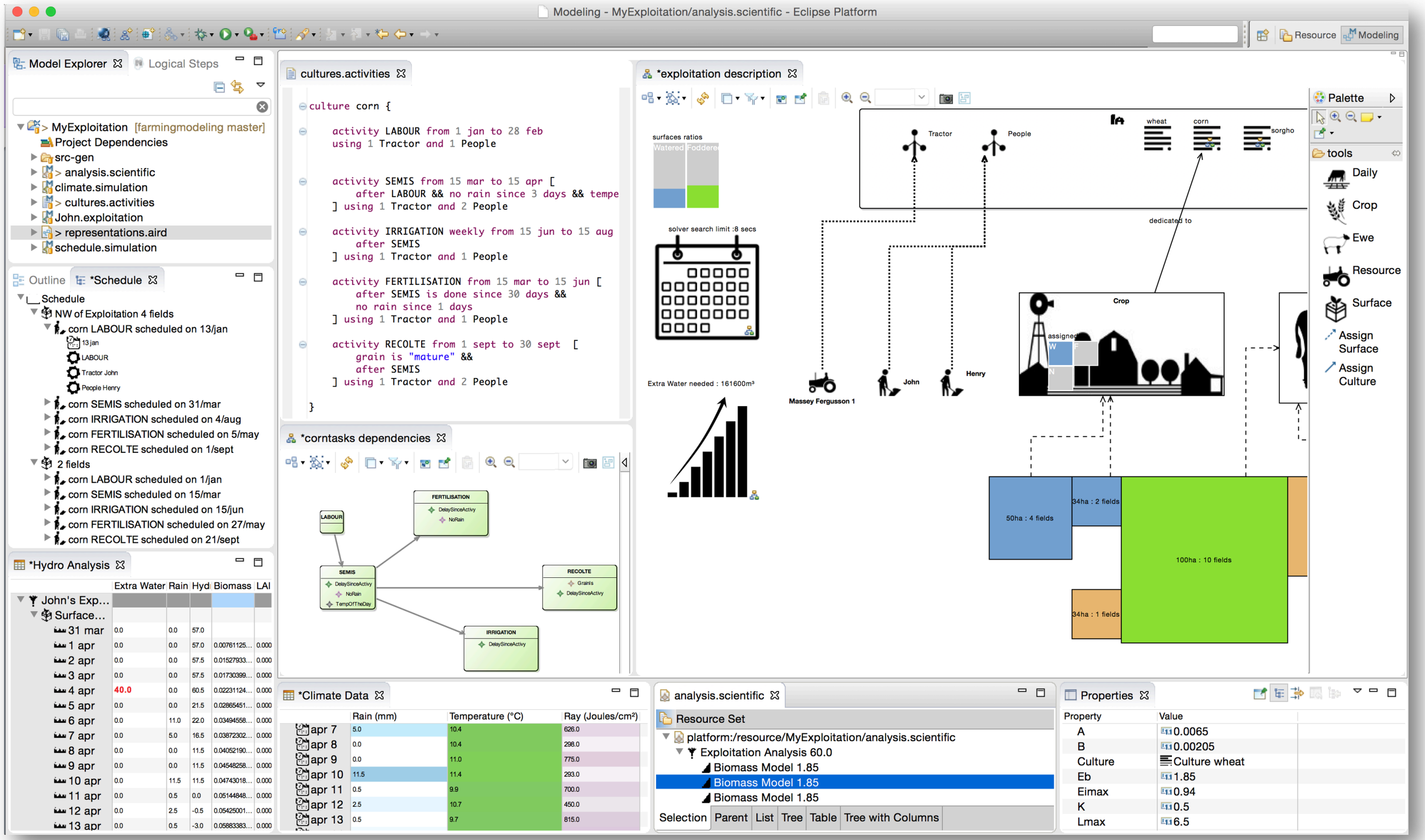
Transformation Lg Debugger



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

https://github.com/tetrabox/minitl

Farming System Model

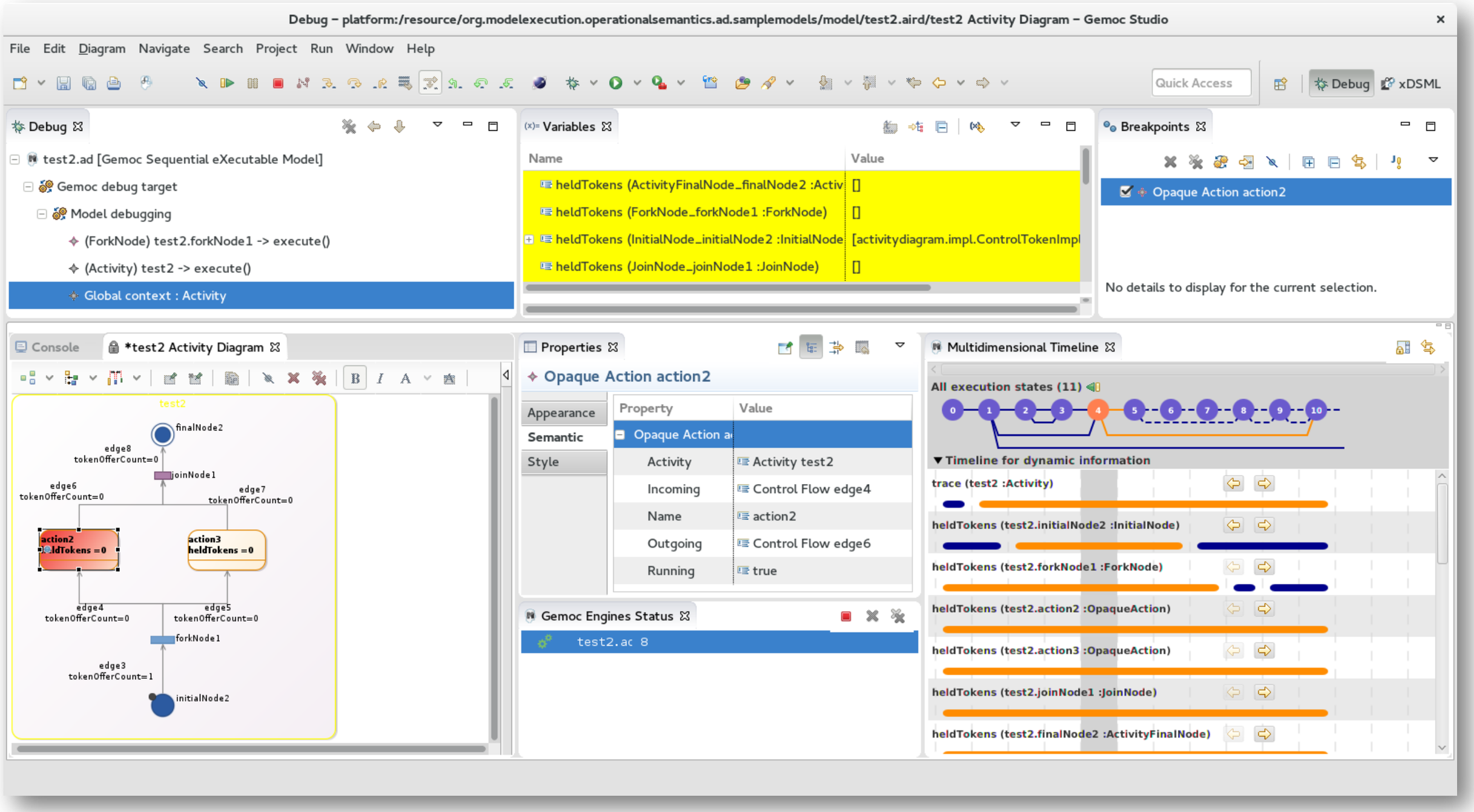


https://github.com/gemoc/farmingmodeling

MDE Introduction (M1ICE)

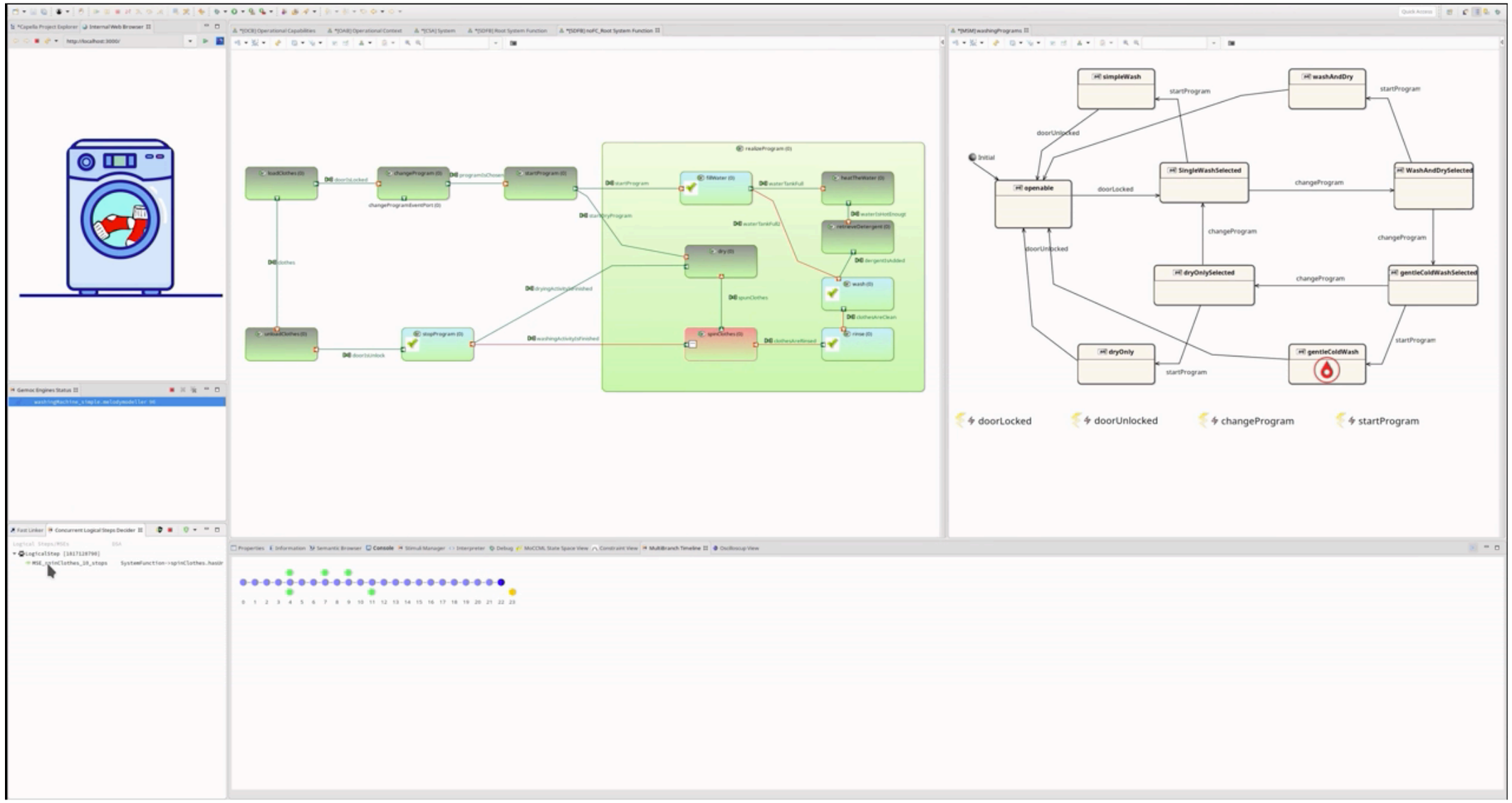
*Benoit Combemale, **Feb. 2021***

UML Activity Diagram



https://github.com/gemoc/activitydiagram

xCapella

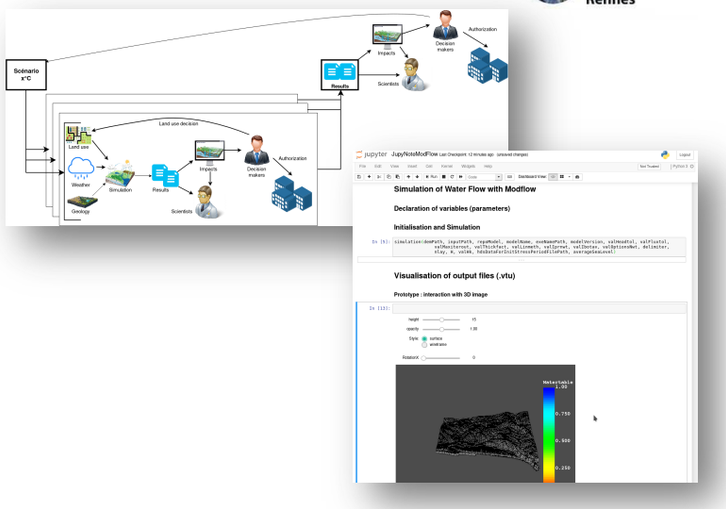

simpleWash
washAndDry
startProgram
doorUnlocked
Initial
openable
doorLocked
SingleWashSelected
changeProgram
WashAndDrySelected
dryOnlySelected
gentleColdWashSelected
dryOnly
gentleColdWash
doorLocked
doorUnlocked
changeProgram
startProgram

Crosscutting Challenges for Various Domains

Towards Virtual Labs

From Modeling Environment to Digital Twin

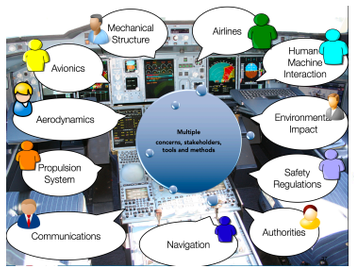
in collaboration with

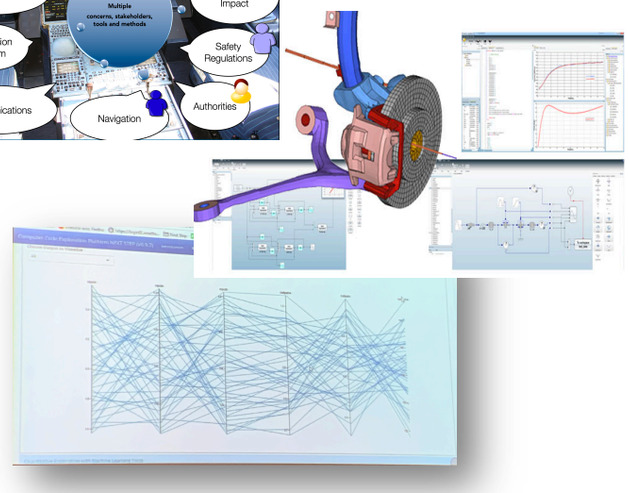


Tradeoff Analysis for
Water Flood Prediction

in collaboration with

SAFRAN

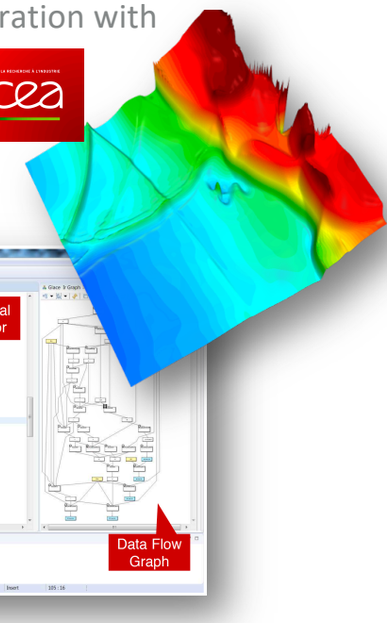




Design Space Exploration
for Cyber-Physical Systems

in collaboration with

cea



High-Performance Computing
for Numerical Analysis

UNIVERSITÉ DE RENNES 1

UMR IRISA

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