MODEL-DRIVEN *(SOFTWARE)* ENGINEERING

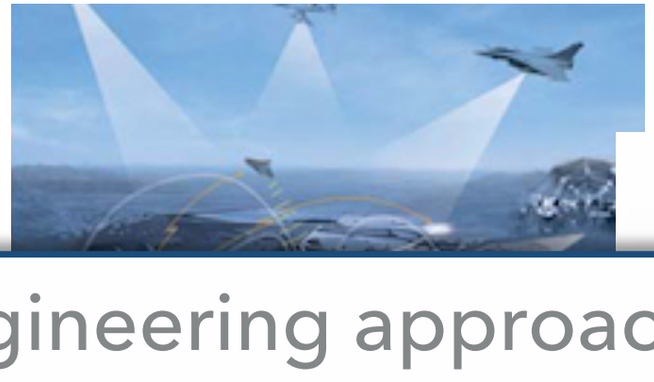
COURSE INTRODUCTION

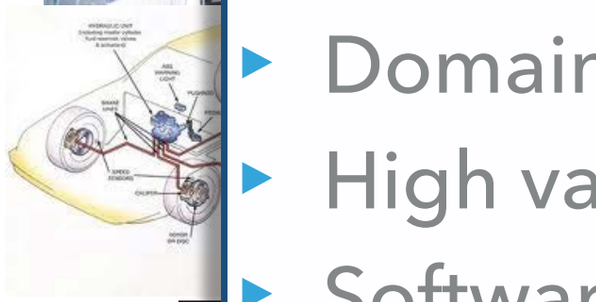
MASTER 1 ICE, 2017-2018

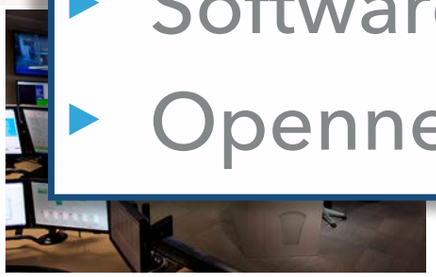
BENOIT COMBEMALE
PROFESSOR, UNIV. TOULOUSE, FRANCE

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

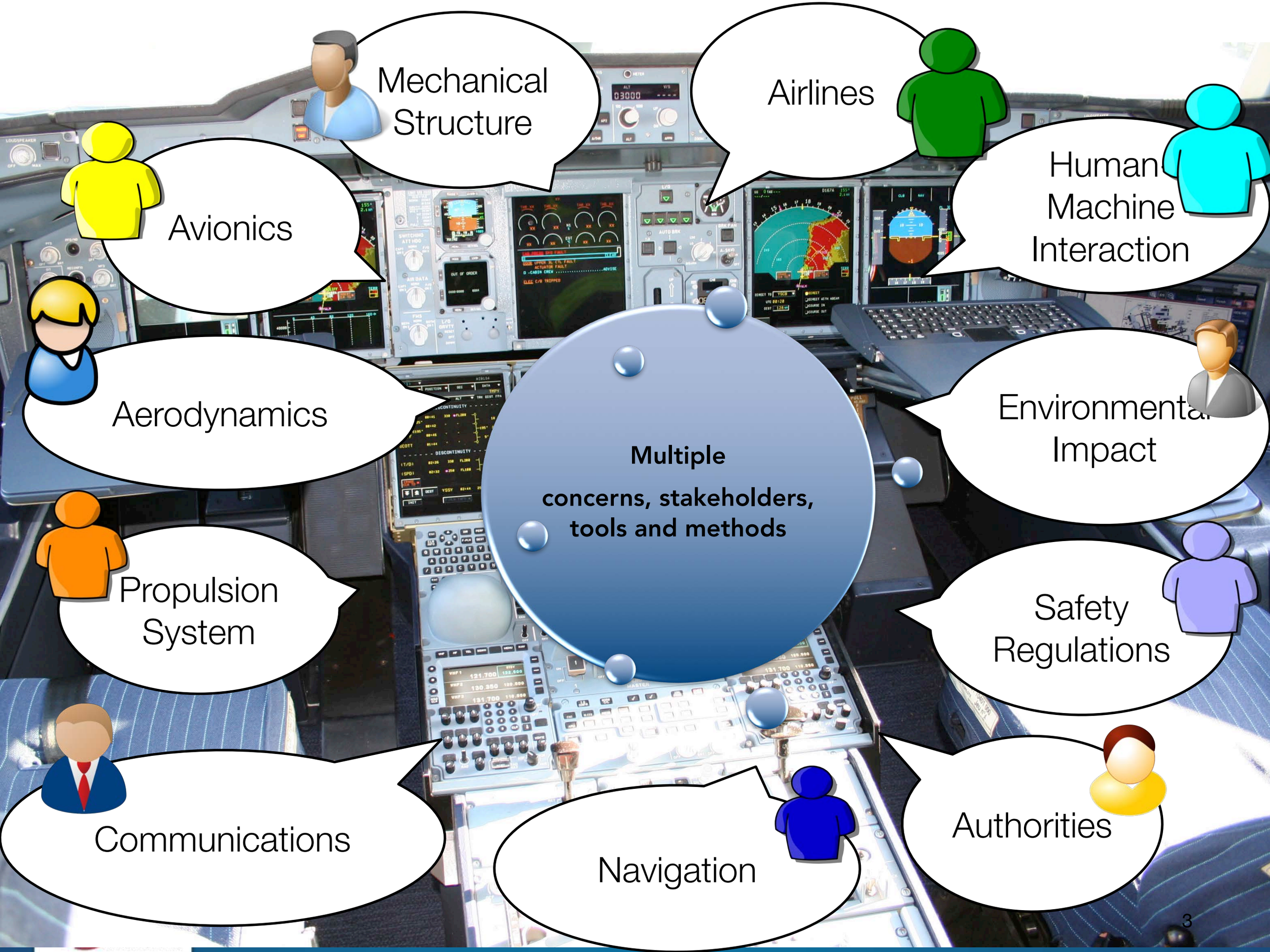
Complex Software-Intensive Systems



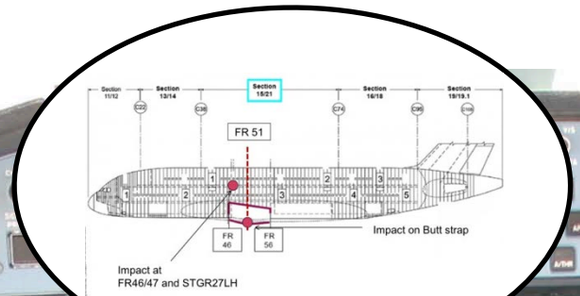


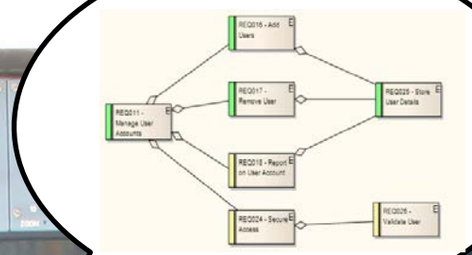


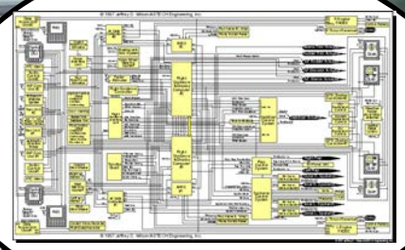
- Multi-engineering approach
- Domain-specific modeling
- High variability and customization
- Software as integration layer
- Openness and dynamicity

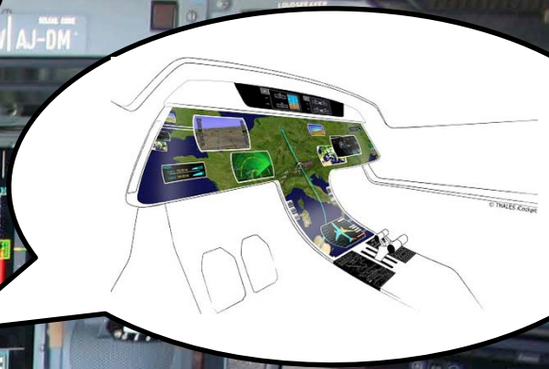

Mechanical Structure
Airlines
Avionics
Human Machine Interaction
Aerodynamics
Multiple
concerns, stakeholders, tools and methods
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorities

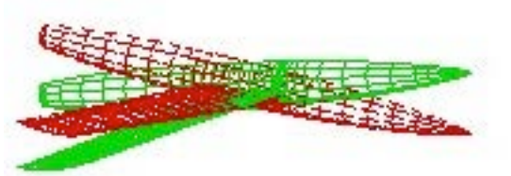
Heterogeneous Modeling

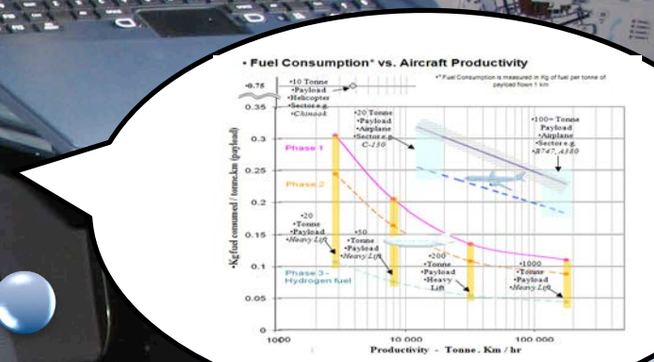


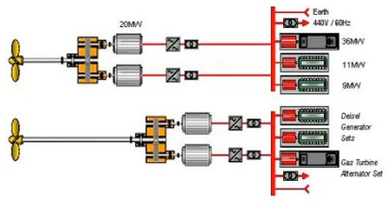


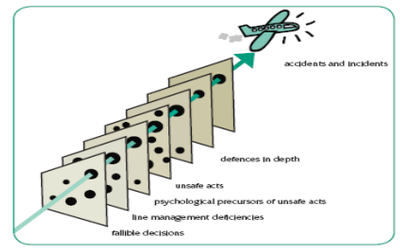


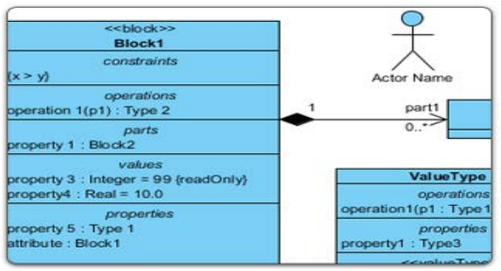


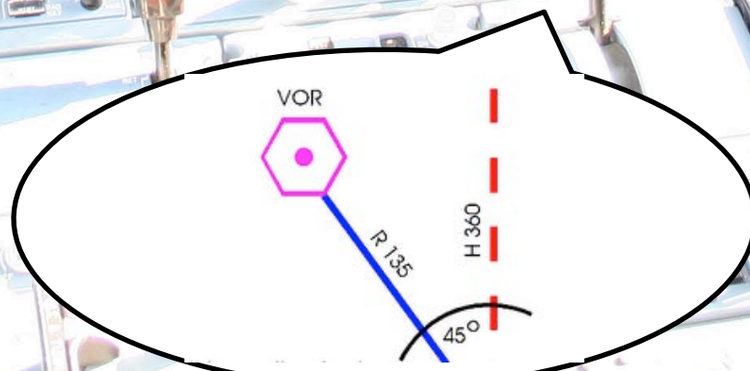


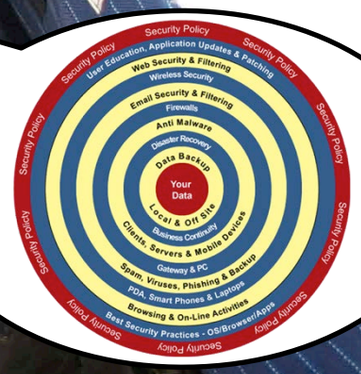












Software Modeling: Why Should I Care?

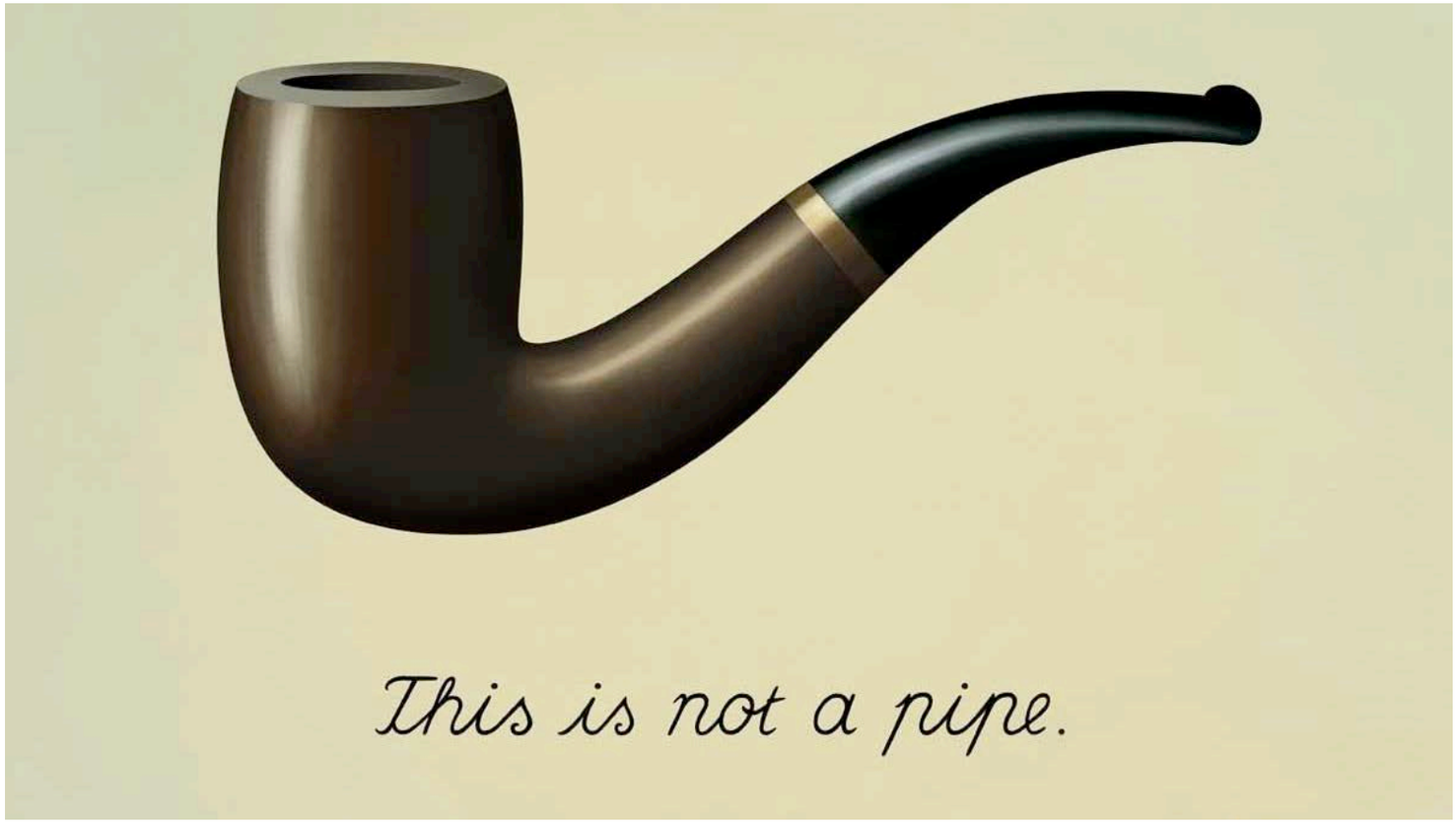
- Pour réfléchir :
  - représentation abstraite
  - séparation des préoccupations
- Pour communiquer :
  - représentation graphique
  - génération de documentation
- Pour automatiser le développement :
  - génération de code
  - application de patrons
  - migration
- Pour vérifier :
  - validation et 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

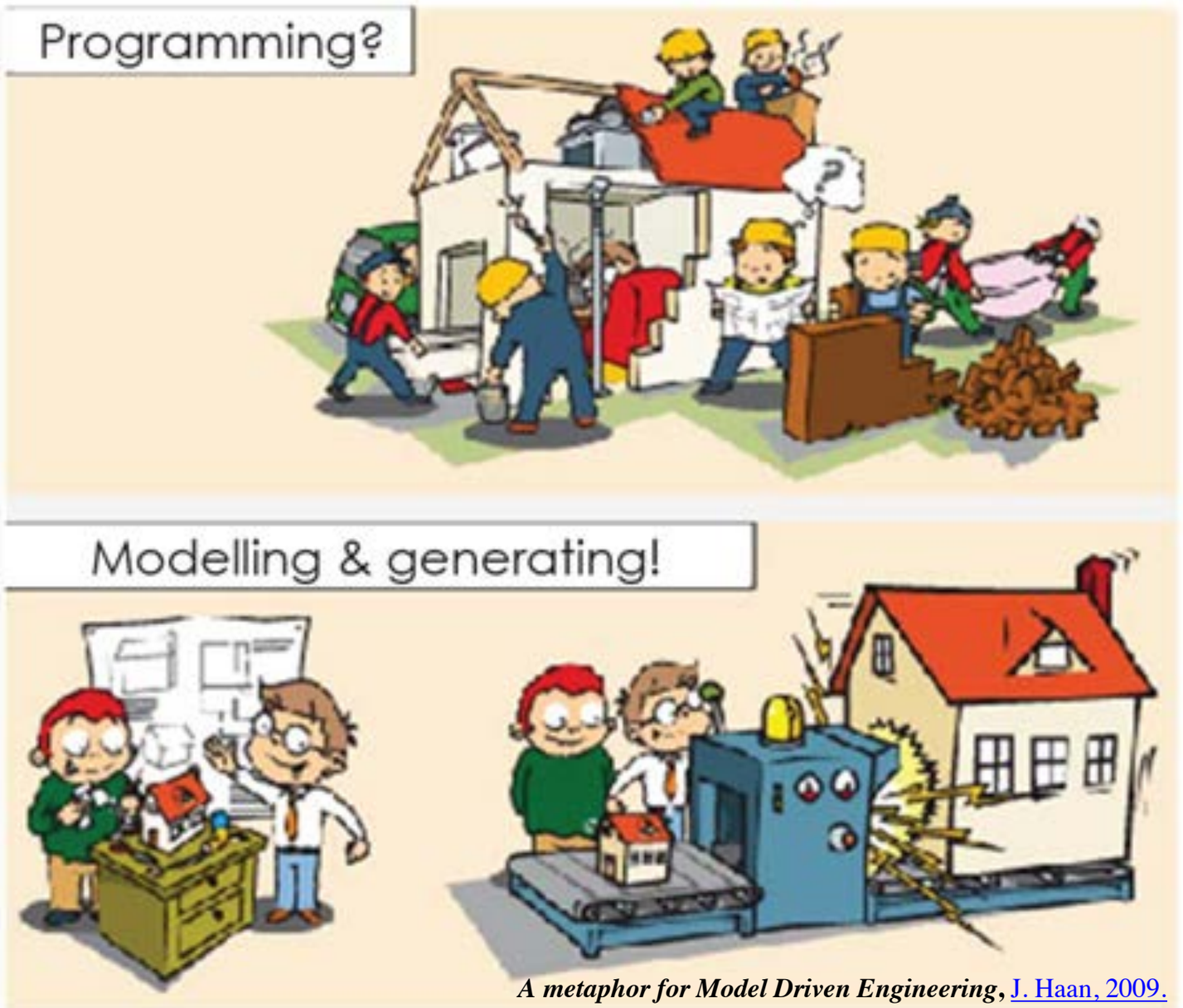
Model and Reality in Software

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



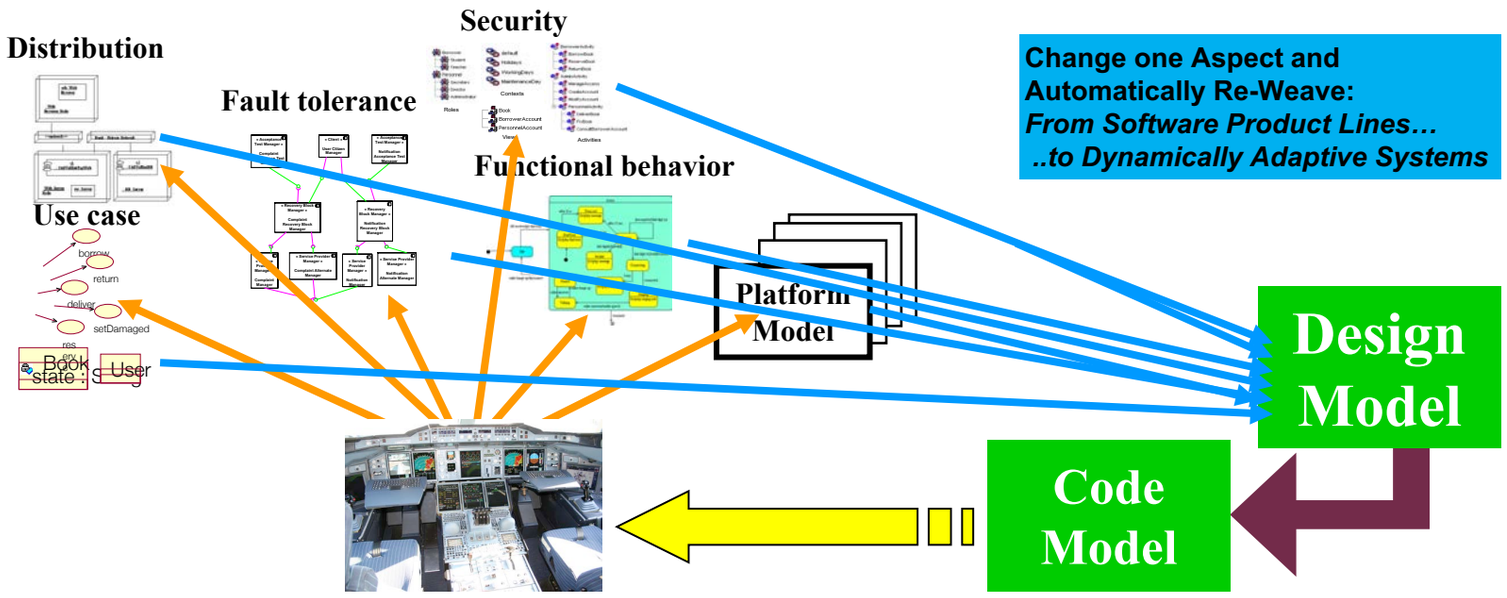
- Software Models: from contemplative to productive

Towards Model-Driven Engineering (MDE)



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

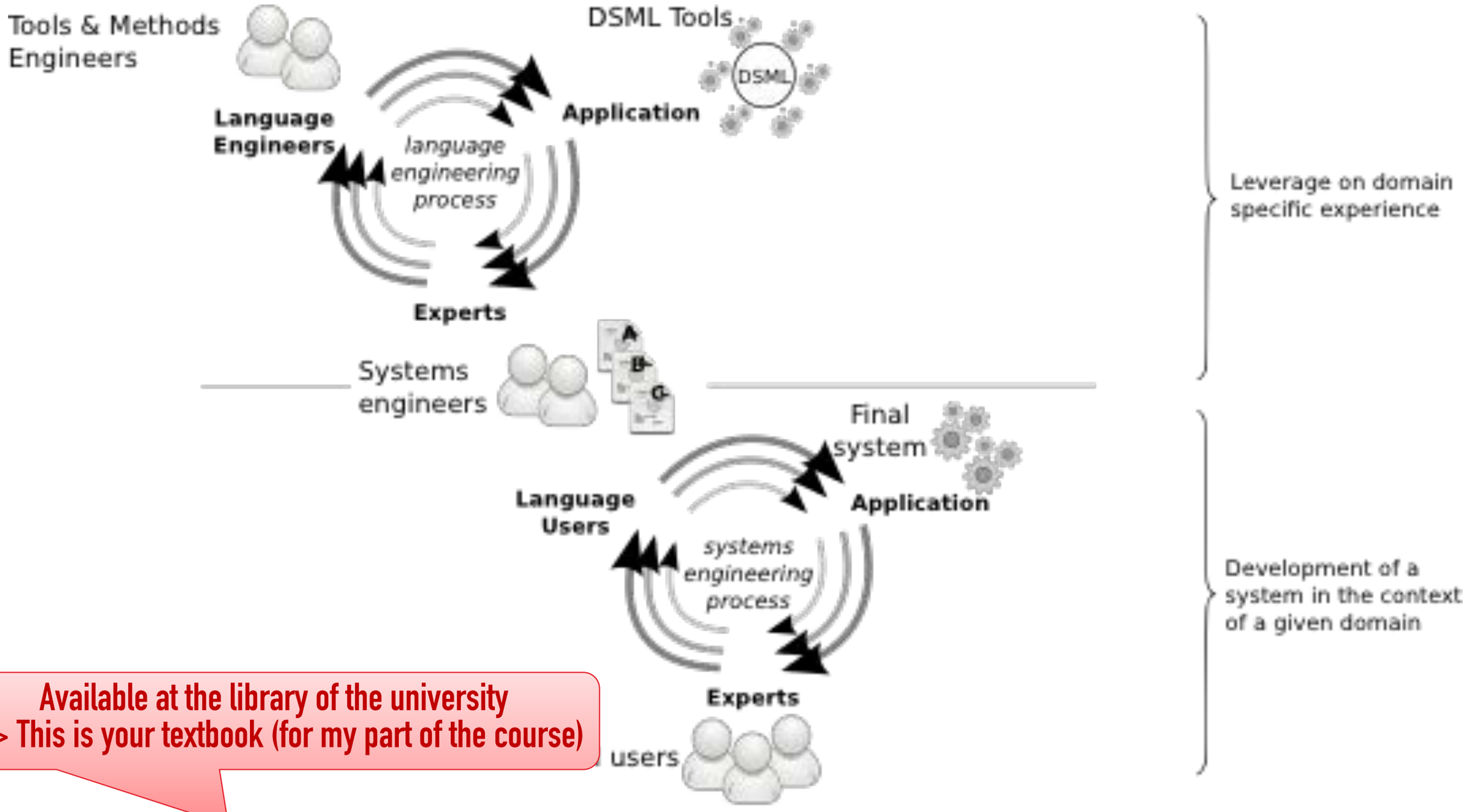
Model-Driven Engineering (MDE)



"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.

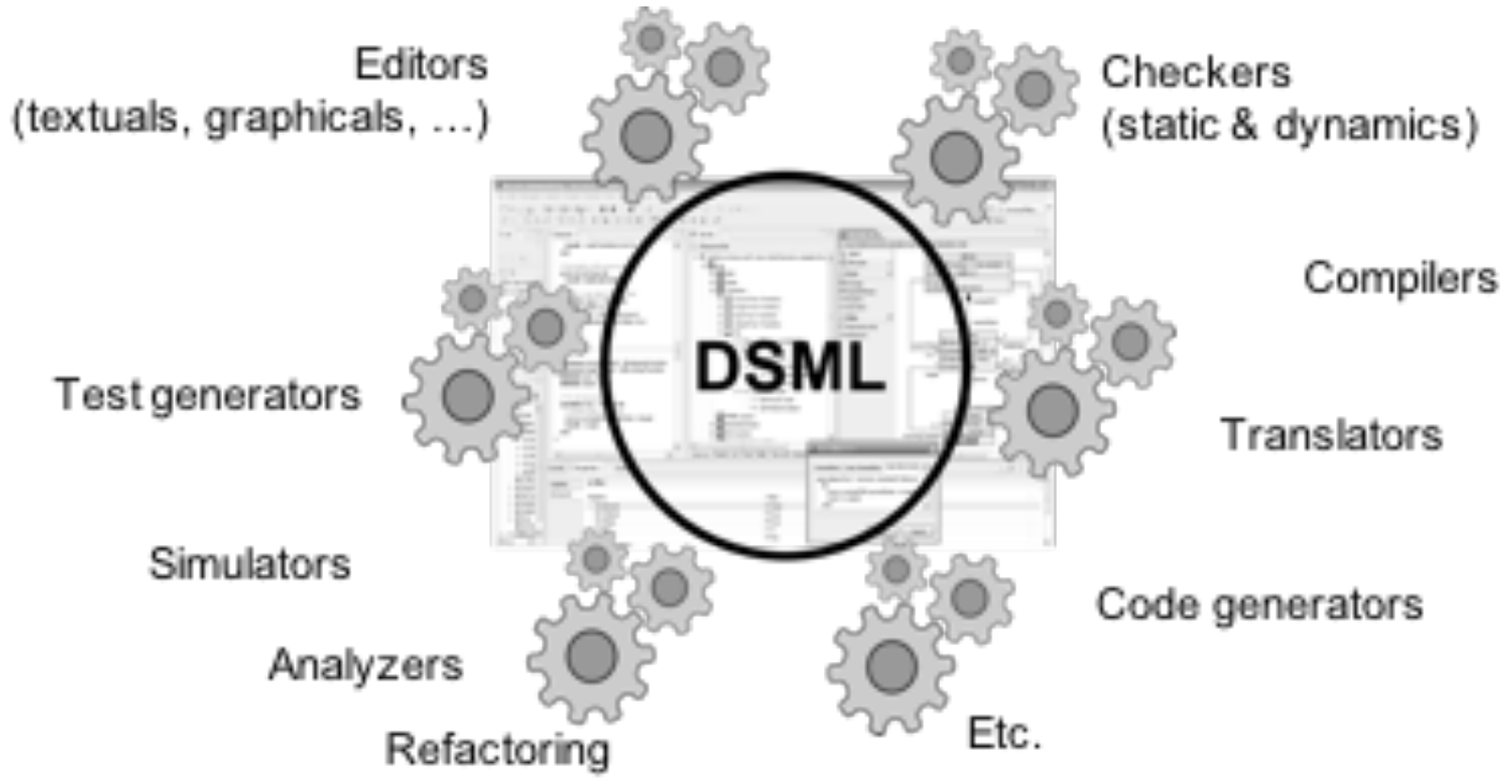
Model-Driven Engineering (MDE)



Available at the library of the university
=> This is your textbook (for my part of the course)

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

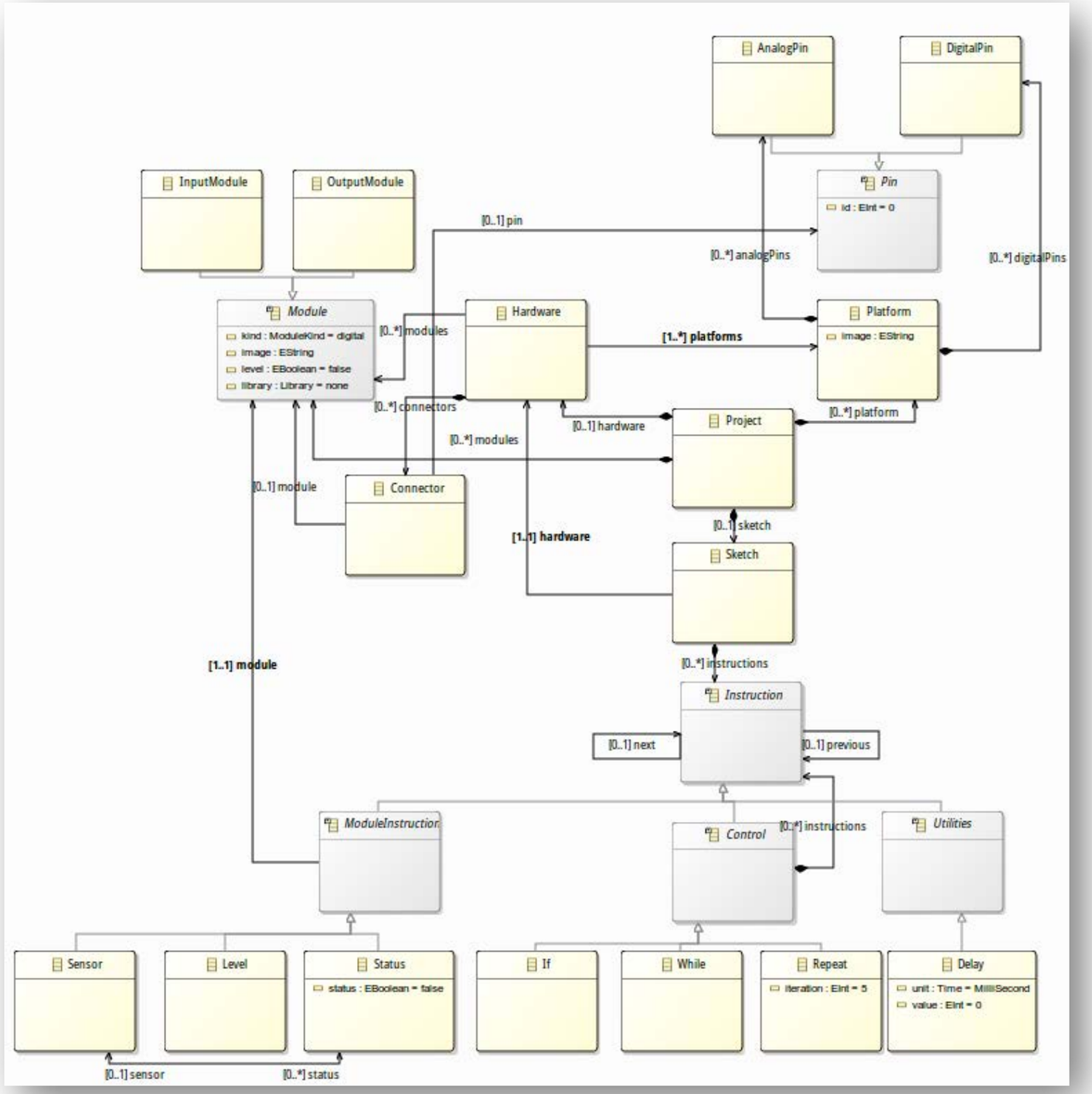
"Software Languages are Software Too"

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)…

Ecore Tools: Graphical Edition of Ecore Models



The GEMOC Studio

http://gemoc.org/studio

soon

http://eclipse.org/gemoc

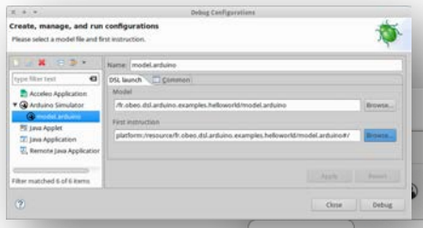
Language Workbench

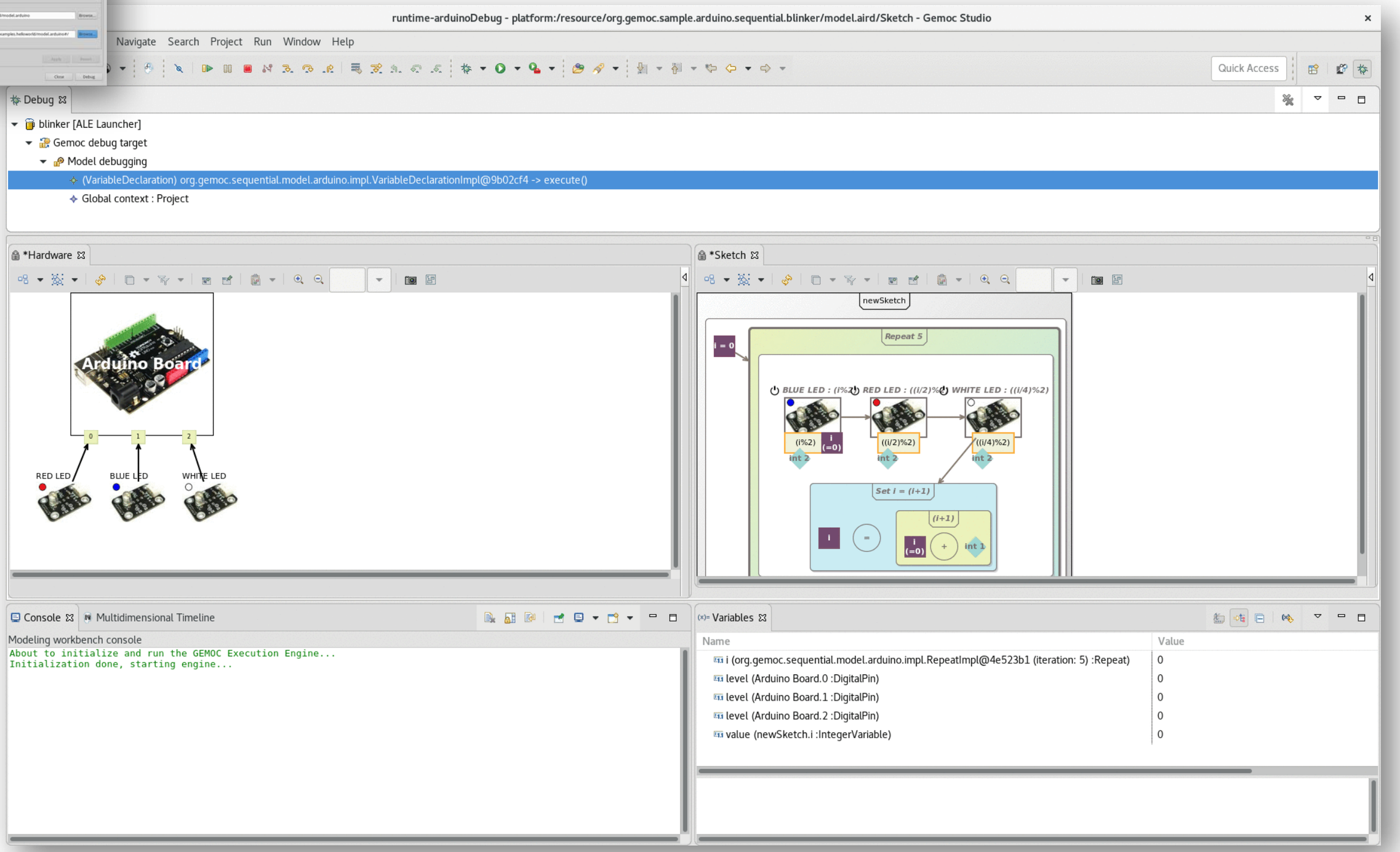
Design and integrate your executable DSMLs

Modeling Workbench

Edit, simulate and animate your heterogeneous models

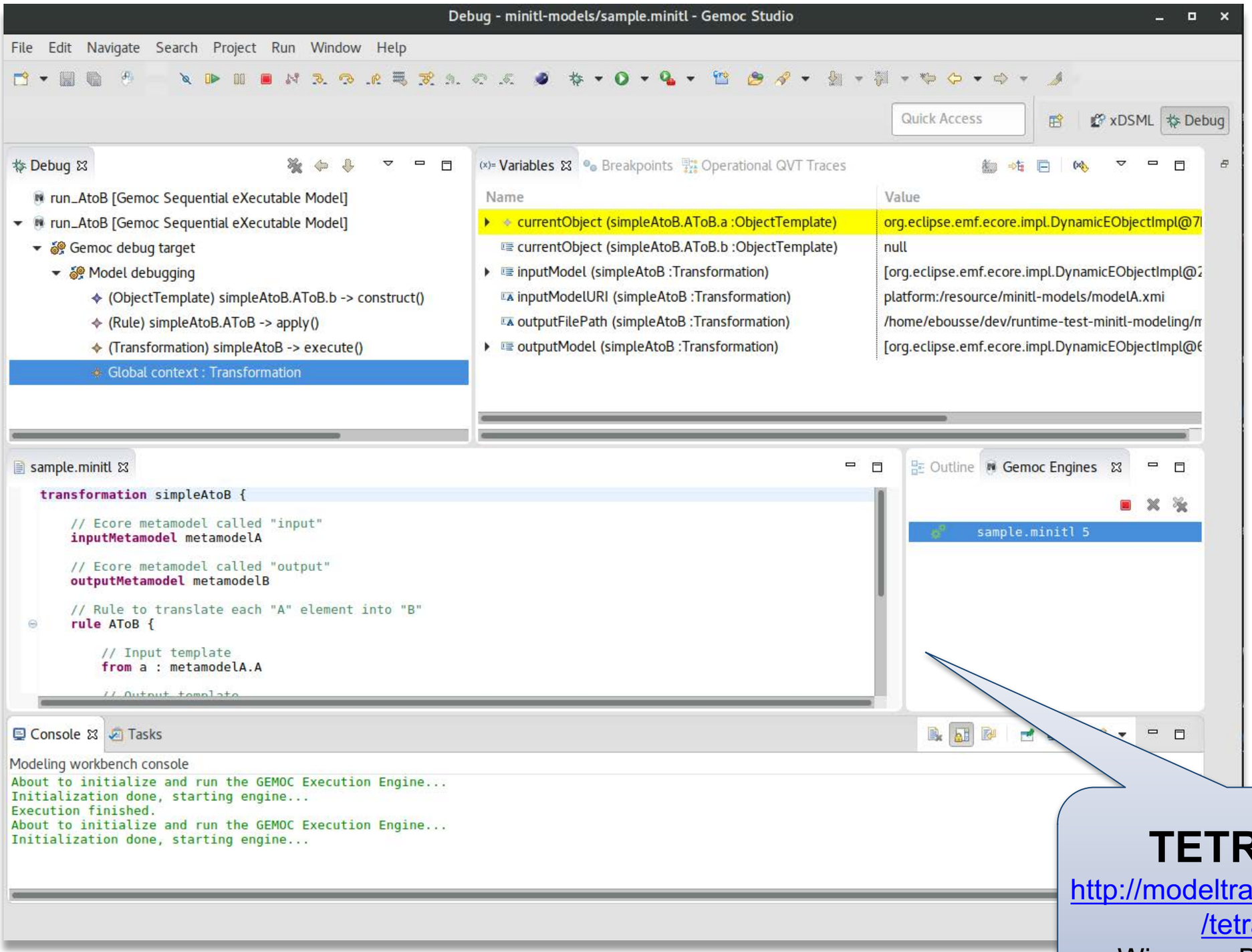
Arduino Designer





https://github.com/gemoc/arduinomodeling

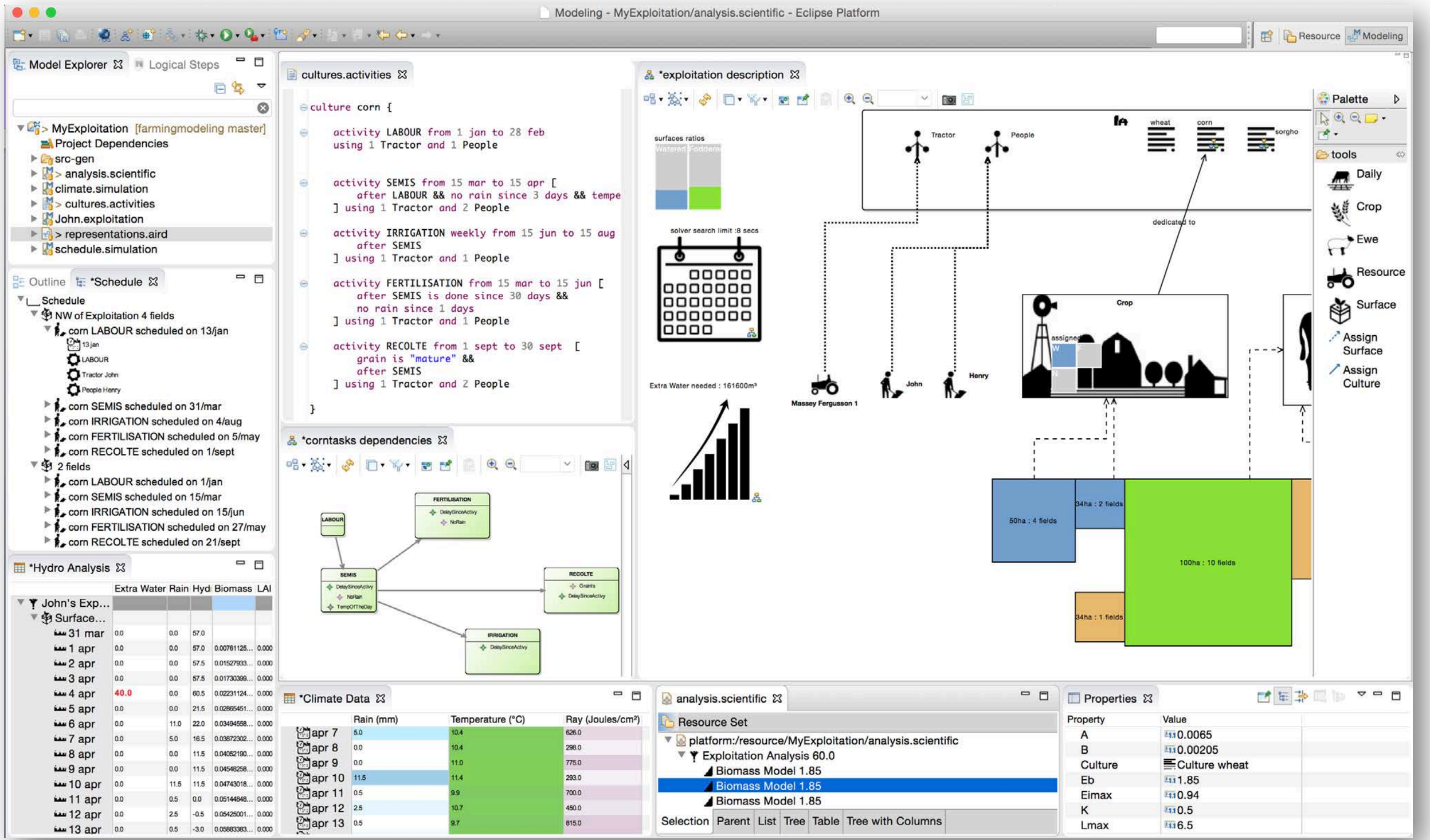
Transformation Lg Debugger



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

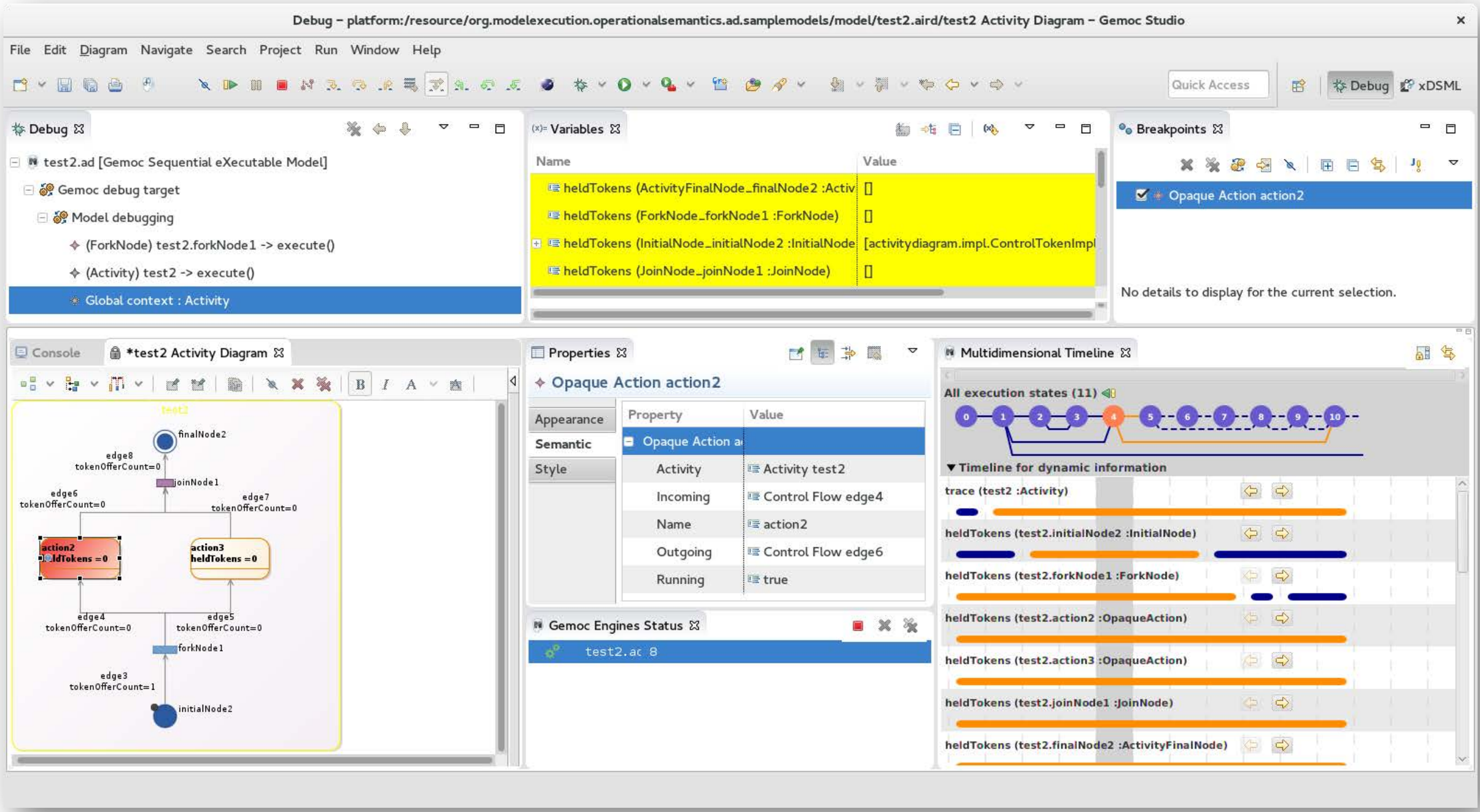
https://github.com/tetrabox/minitl

Farming System Modeling



https://github.com/gemoc/farmingmodeling

Activity Diagram Debugger



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