Language Engineering with The GEMOC Studio

Tutorial @ MODELS 2017, September 18th, 2017

Benoit Combemale (Univ. Toulouse)
http://www.combemale.fr
benoit.combemale@irit.fr
@bcombemale

Julien Deantoni (Univ. Nice-Sophia-Antipolis)
http://www.i3s.unice.fr/~deantoni/
julien.deantoni@polytech.unice.fr

Tutorial website

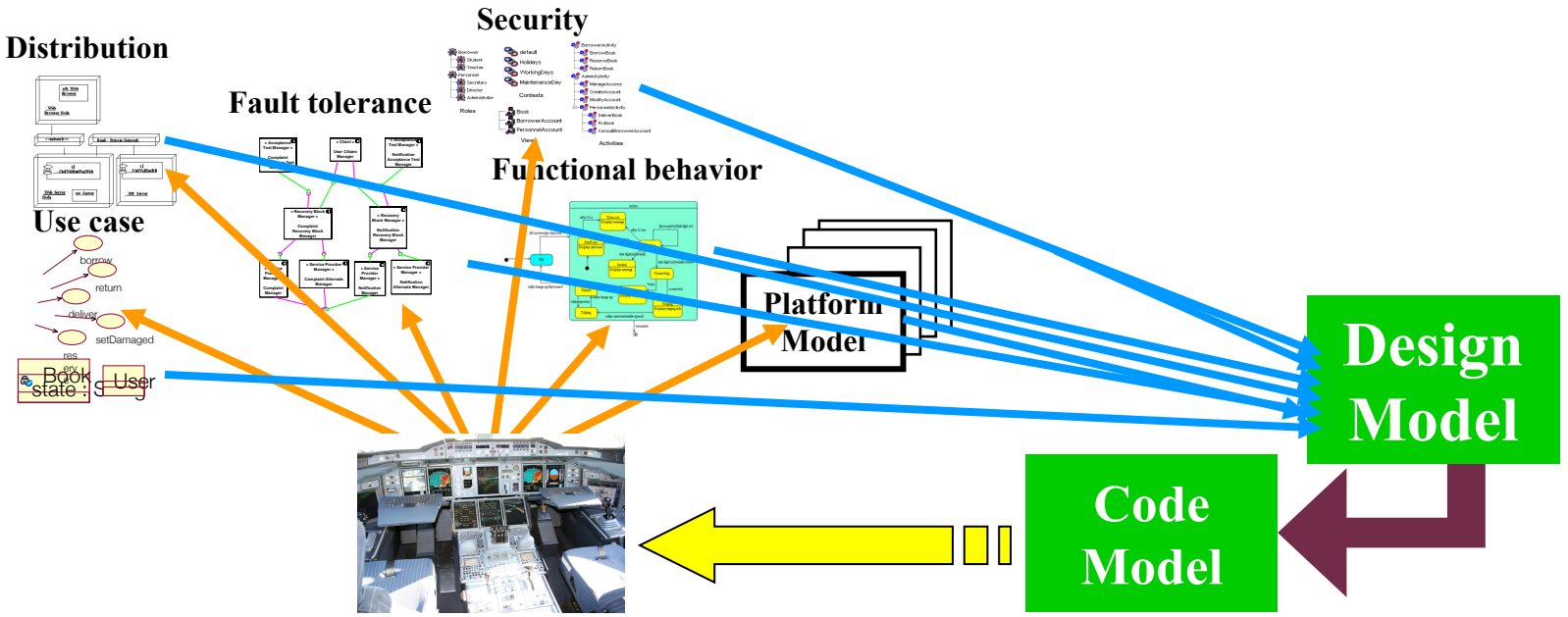
https://github.com/gemoc/MODELS2017Tutorial

See:

- README
- /slides/tutorial-slides.pptx
- /code/* ☺

LANGUAGE ENGINEERING

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.

Domain-Specific Languages (DSLs)

- Tailored for 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

- DSL ~ Modeling Language!

"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 languages
- Various shapes and ways to implement software languages
  - External, internal or embedded DSLs, Profile, etc.
  - Grammar, metamodel, ontology, etc.
- Supported by various kind of "**language workbench**"
  - Eclipse EMF, Xtext, Sirius, Melange, GEMOC, Papyrus
  - Jetbrain's MPS
  - Spoofax
  - MS DSL Tools
  - Etc.
- More and more literature, a dedicated Intl. conference (ACM SLE, cf. http://www.sleconf.org)…

GEMOC

The GEMOC initiative

An open and international initiative to

- foster **globalization of DSMLs** in open world
  - ⇒ impossible *a priori* unification
  - ⇒ requires *a posteriori* globalization
- coordinate (between members) and disseminate (for the members)

Organizes

- **workshop series**
- **Seminars at Dagstuhl, Bellairs**

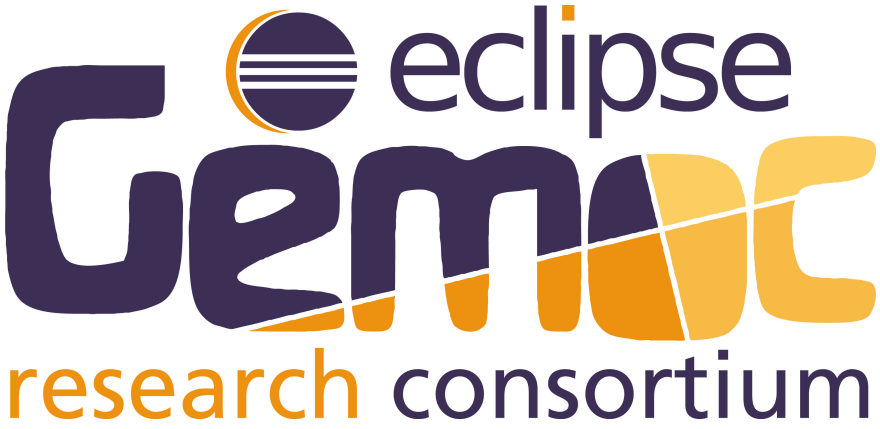
http://gemoc.org

@gemocinitiative

The GEMOC Research Challenges

- Modular language design and implementation
- Language interfaces
- Composition operators (for reuse/variability management, for coordination, …)

The GEMOC Eclipse Research Consortium



Sustain the GEMOC initiative as a research platform

- Project commitment
- Own budget for maintaining the studio and the dissemination

The GEMOC Studio

http://gemoc.org/studio

Language Workbench

Design and compose your executable DSMLs

Modeling Workbench

Edit and debug your heterogeneous models

The GEMOC Eclipse Project

http://eclipse.org/gemoc

Sustain the GEMOC studio as a research platform

- To support experimentations of new solutions by academics
- To support pilot projects by industry
- To foster relationships between academia and industry
- To help industrial transfer and innovation

The GEMOC Studio: Major Components

Language Workbench

Metaprogramming Approaches

Various ways to extend metamodels with interpreters, compilers, and model checkers (Xtend/Kermeta, XMOF, Java).

Sirius Animator

Define the execution semantics of your executable language along with an animation layer atop the graphical concrete syntax.

Modeling Workbench

Melange

Declare, extend and assemble the components of executable languages.

Generative Approaches

Automatically generate execution trace metamodels to support advanced features in the modeling workbench.

MoCCML

Specify a Model of Concurrency and Communication (MoCC) and its mapping to a specific abstract syntax and associated execution functions of a modeling language.

BCOoL

Specify the language coordination patterns to automatically coordinates the execution of heterogeneous models.

Language Workbench

Generic Execution Framework

Generic bridge in between various execution engines and shared runtime services.

Model Animator

Graphical and textual animation, breakpoint definition, and step-by-step model execution.

Modeling Workbench

Omniscient Debugger

Forward and backward navigation through the execution trace.

Concurrency Analysis

Analysis concurrency constraints on model execution traces.

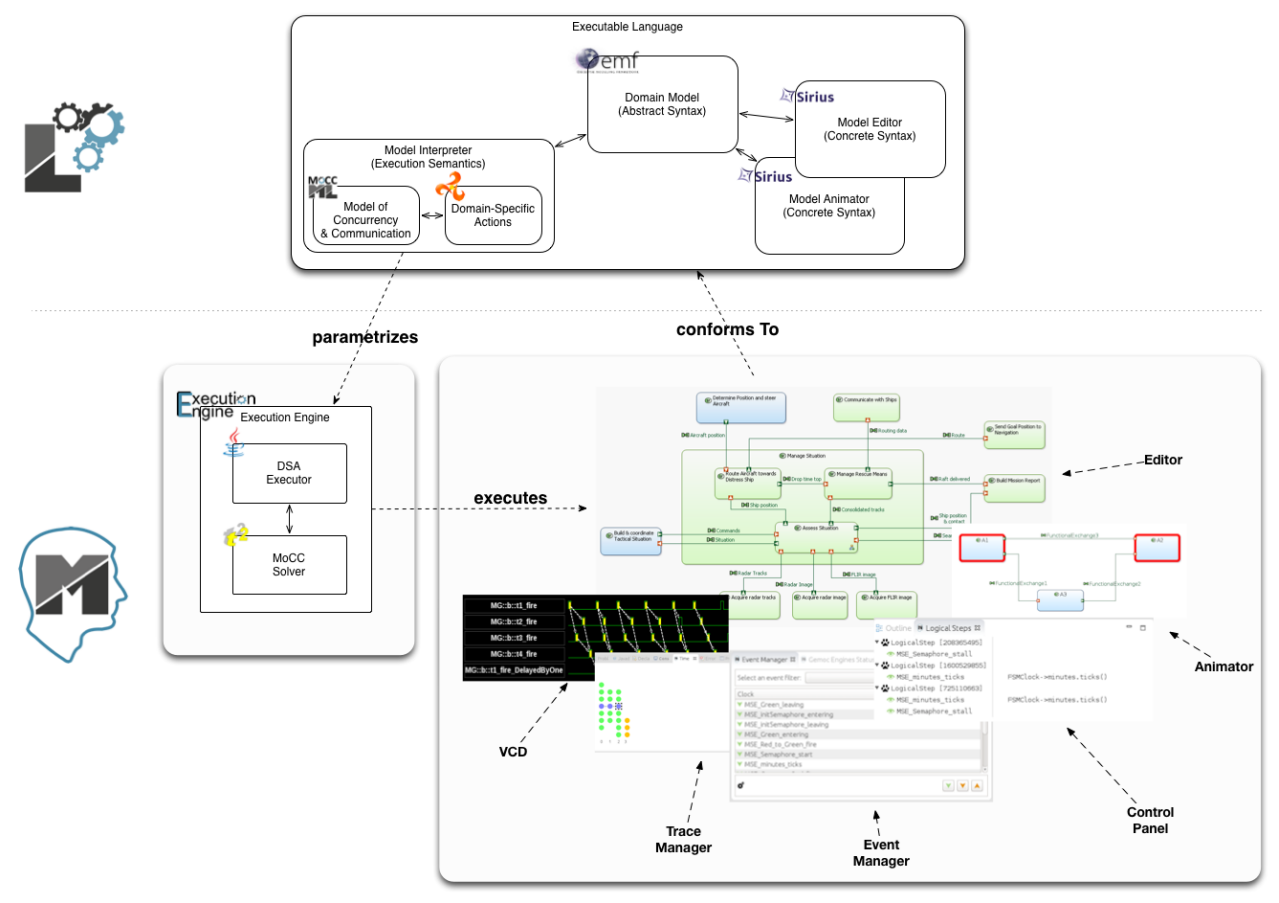
Coordination of Models

Concurrent and coordinated execution of heterogeneous models.

Domain-Specific Plugins

Add a plug-in to the execution framework, either for a specific DSL or to be shared for different DSLs.

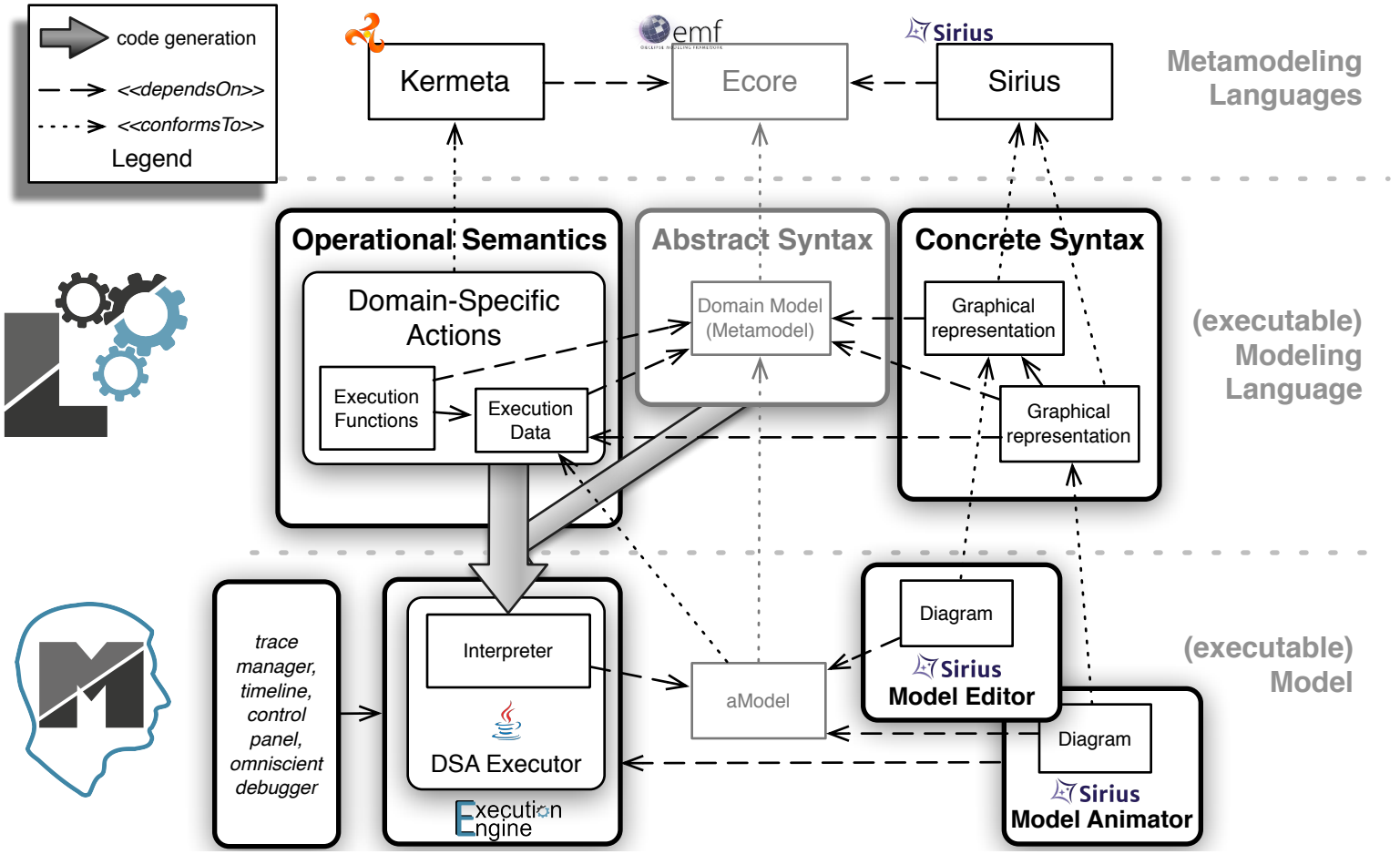
The GEMOC Studio



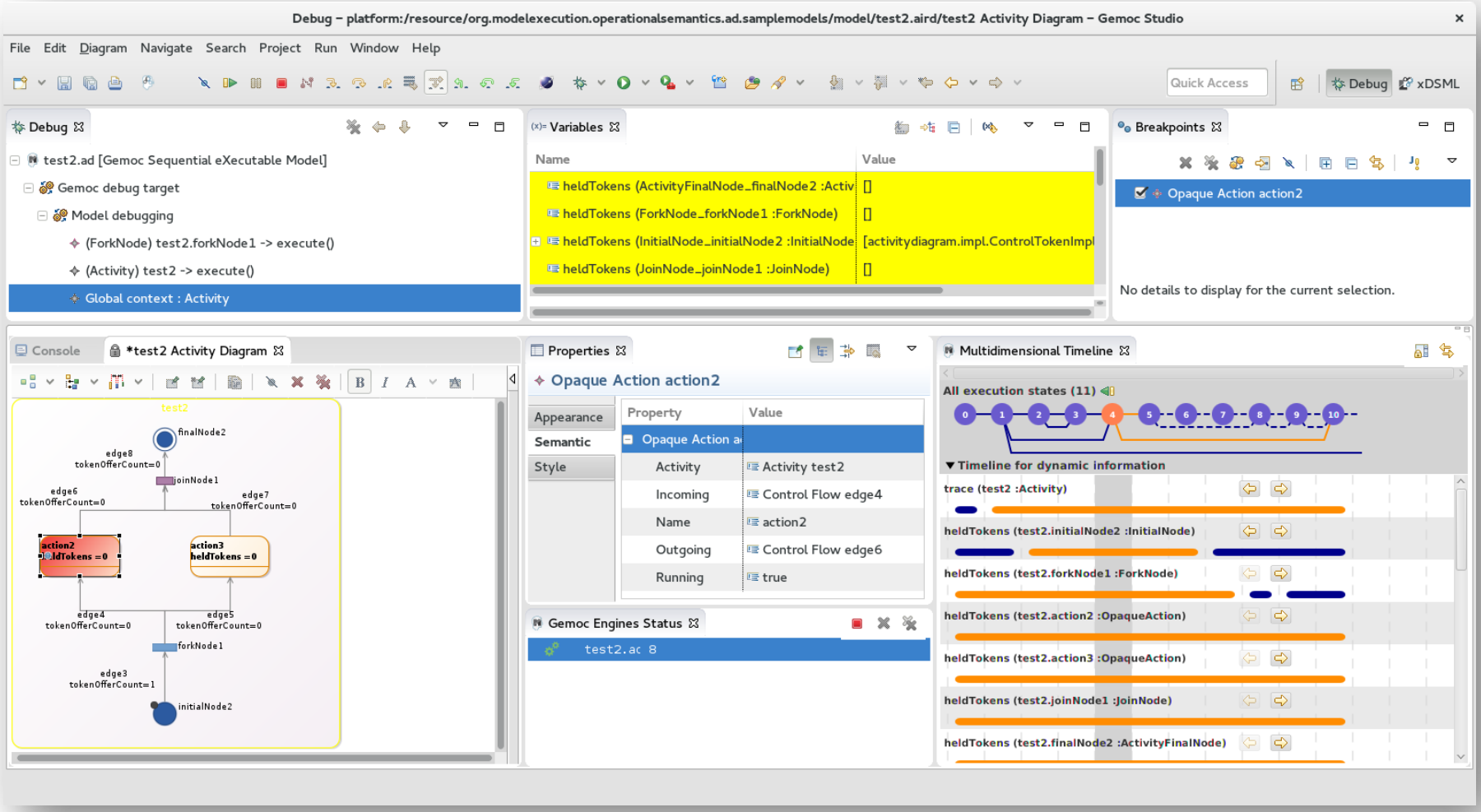
Benoit Combemale, Julien Deantoni, Olivier Barais, Arnaud Blouin, Erwan Bousse, Cédric Brun, Thomas Degueule and Didier Vojtisek, "A Solution to the TTC'15 Model Execution Case Using the GEMOC Studio," In 8th Transformation Tool Contest (TTC), 2015. **Overall Winner**

http://gemoc.org/studio/

Breathe Life Into Your Designer!



Activity Diagram Debugger



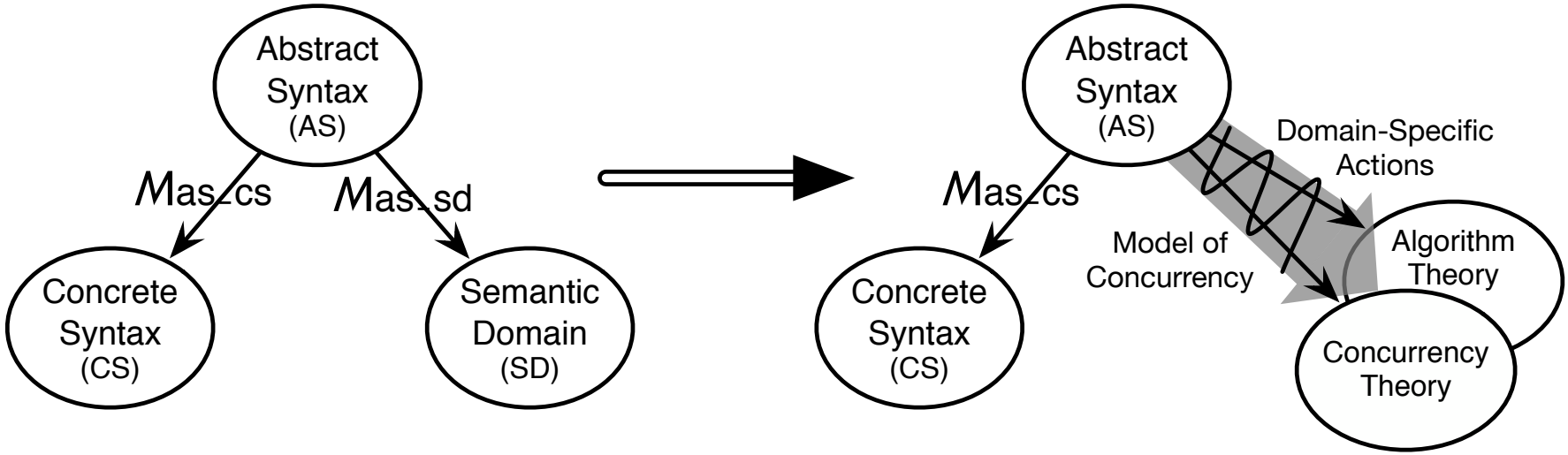
https://github.com/gemoc/activitydiagram

Reifying Concurrency in xDSML: Limitations

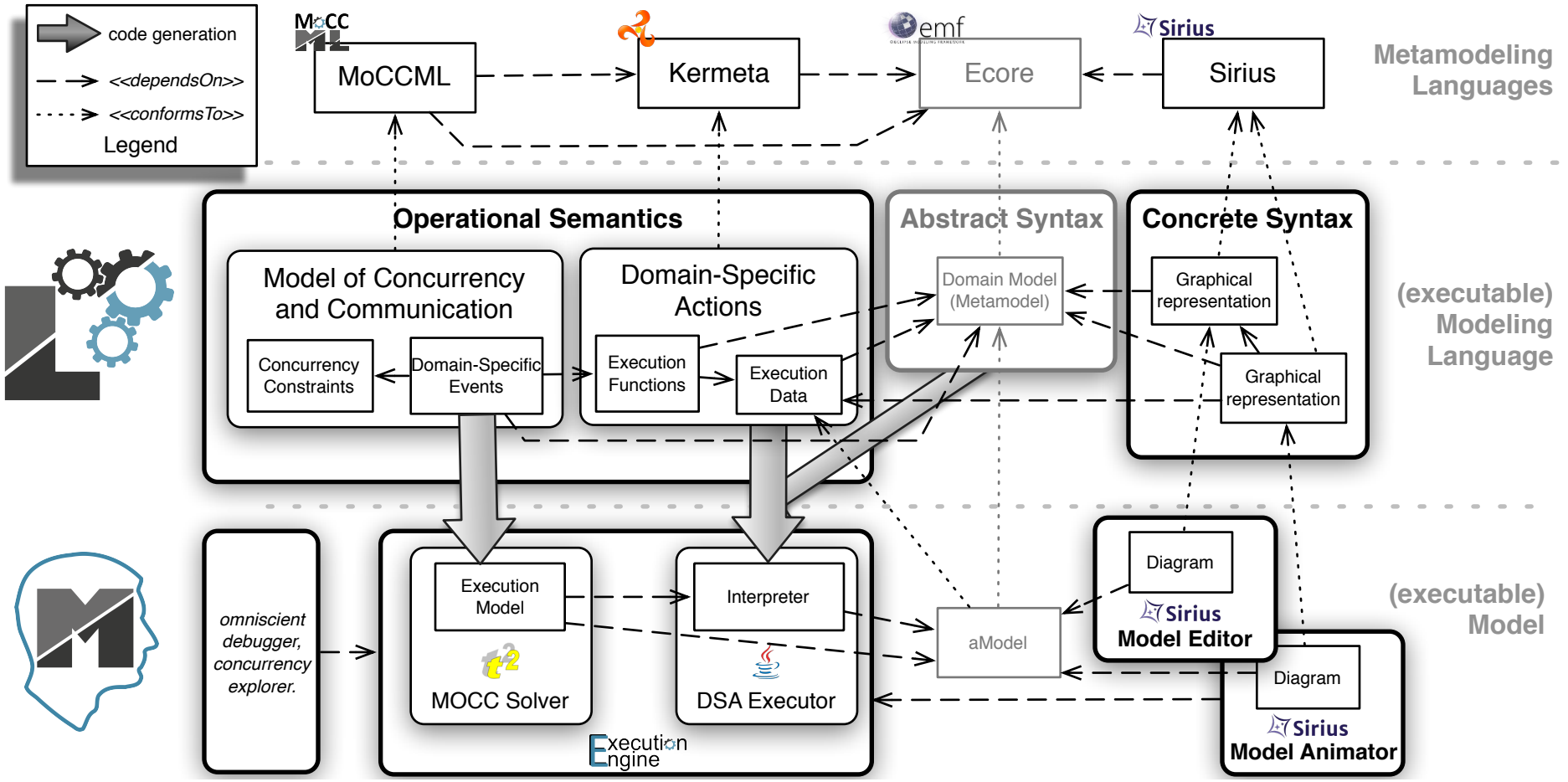
- Concurrency remains implicit and ad-hoc in language design and implementation:
  - Design: implicitly inherited from the meta-language used
  - Implementation: mostly embedded in the underlying execution environment

- The lack of an explicit concurrency specification in language design prevents:
  - leveraging the concurrency concern of a particular domain or platform
  - a complete understanding of the behavioral semantics
  - effective concurrency-aware analysis techniques
  - effective techniques for producing semantic variants
  - analysis of the deployment on parallel architectures

Reifying Concurrency in xDSML: Approach

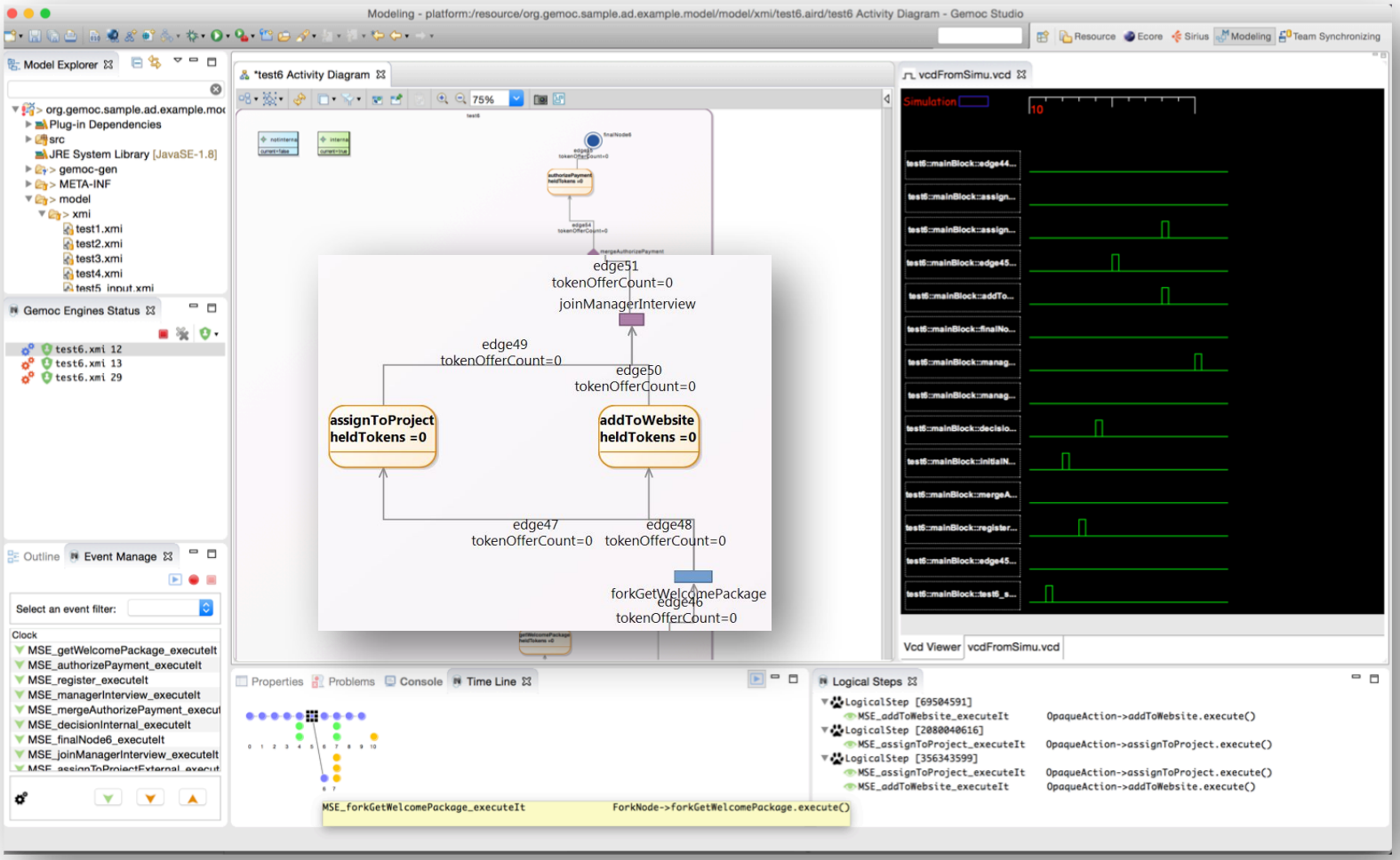


Weave Concurrency Constraints Into Your DSL!



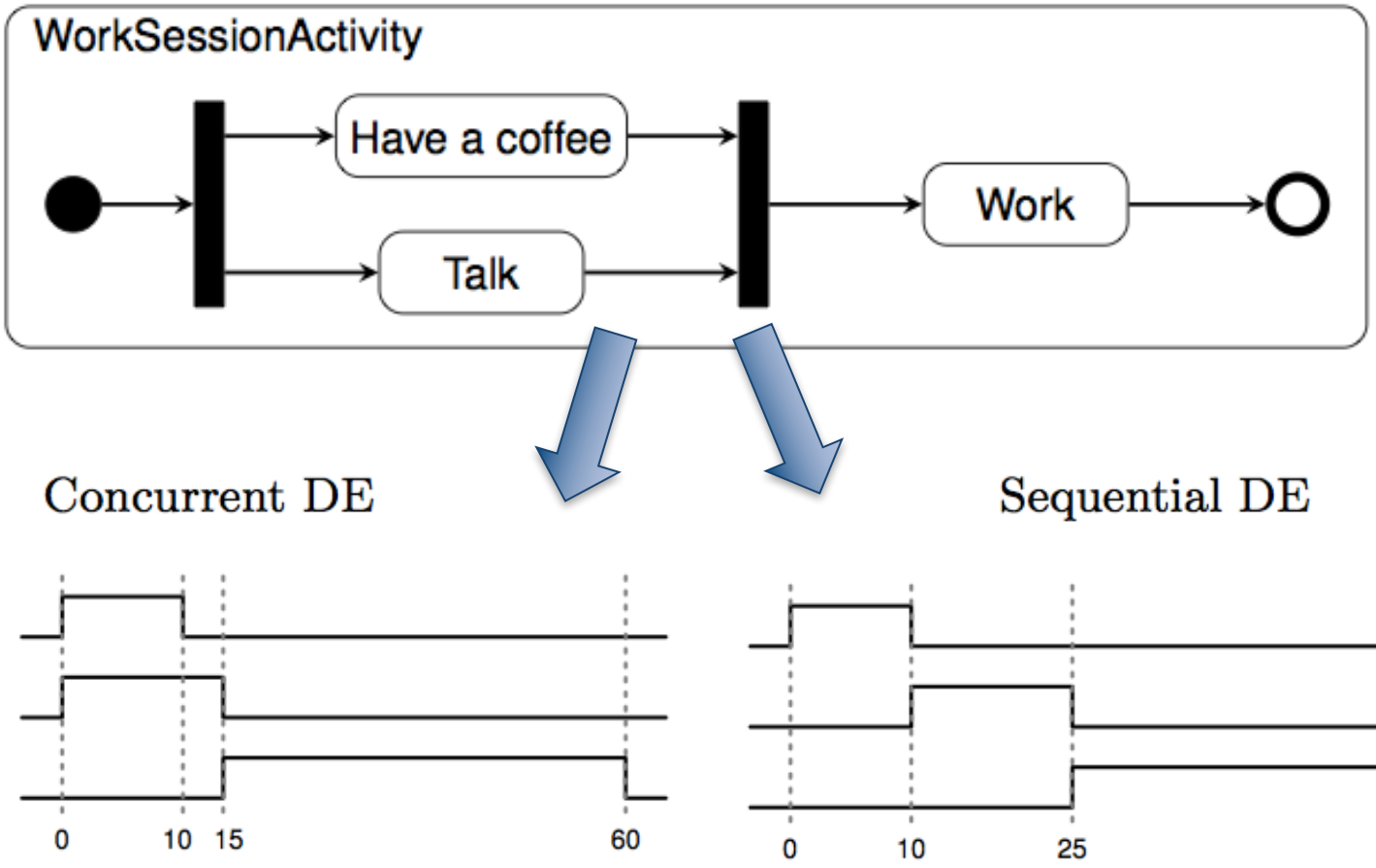
Benoit Combemale, Julien Deantoni, Matias Vara Larsen, Frédéric Mallet, Olivier Barais, Benoit Baudry, Robert France, "Reifying Concurrency for Executable Metamodeling," In Software Language Engineering (SLE), 2013

Activity Diagram Debugger



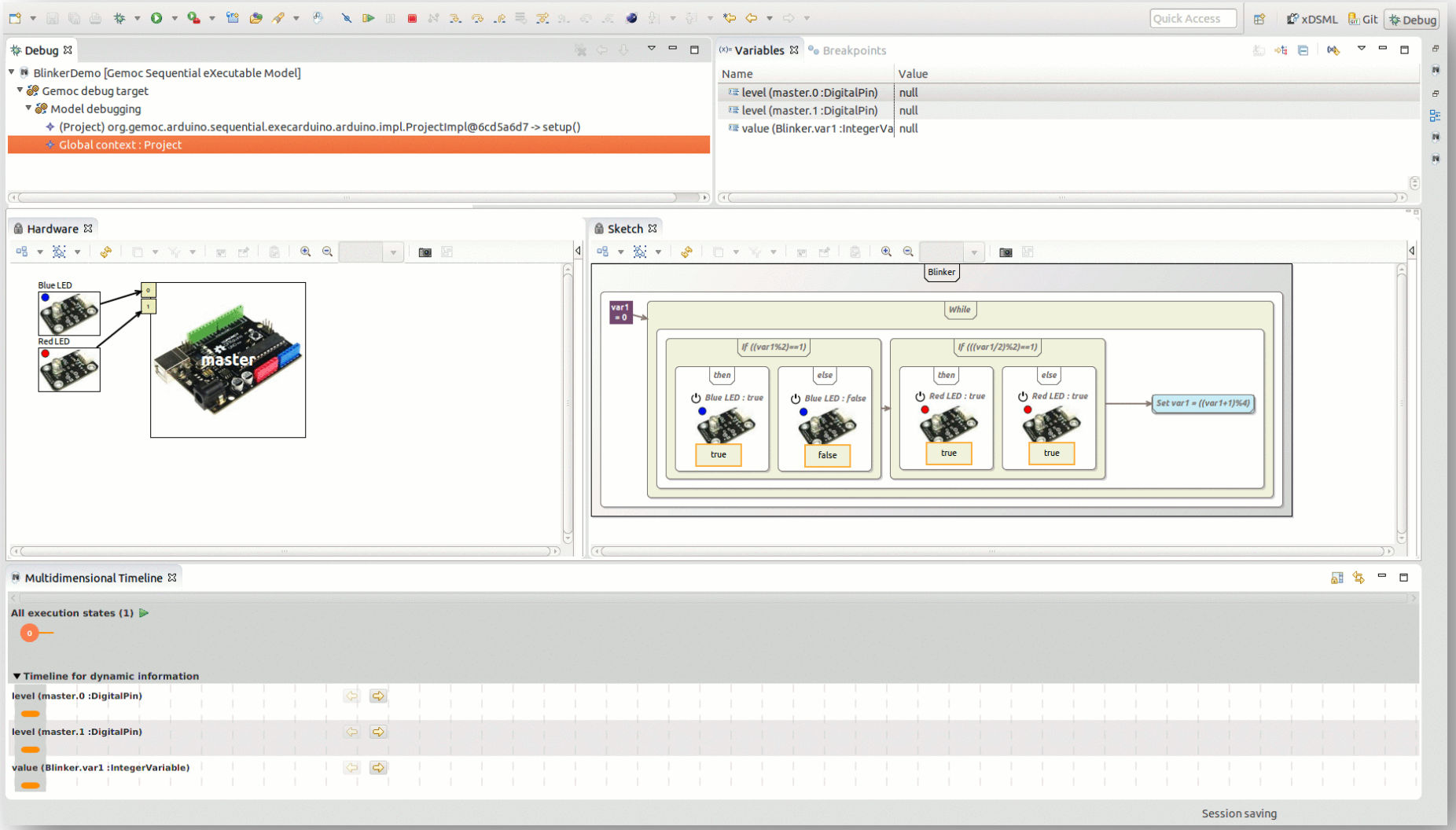
Benoit Combemale, Julien Deantoni, Matias Vara Larsen, Frédéric Mallet, Olivier Barais, Benoit Baudry, Robert France, "Reifying Concurrency for Executable Metamodeling," In Software Language Engineering (SLE), 2013

Coping with Semantic Variation Points



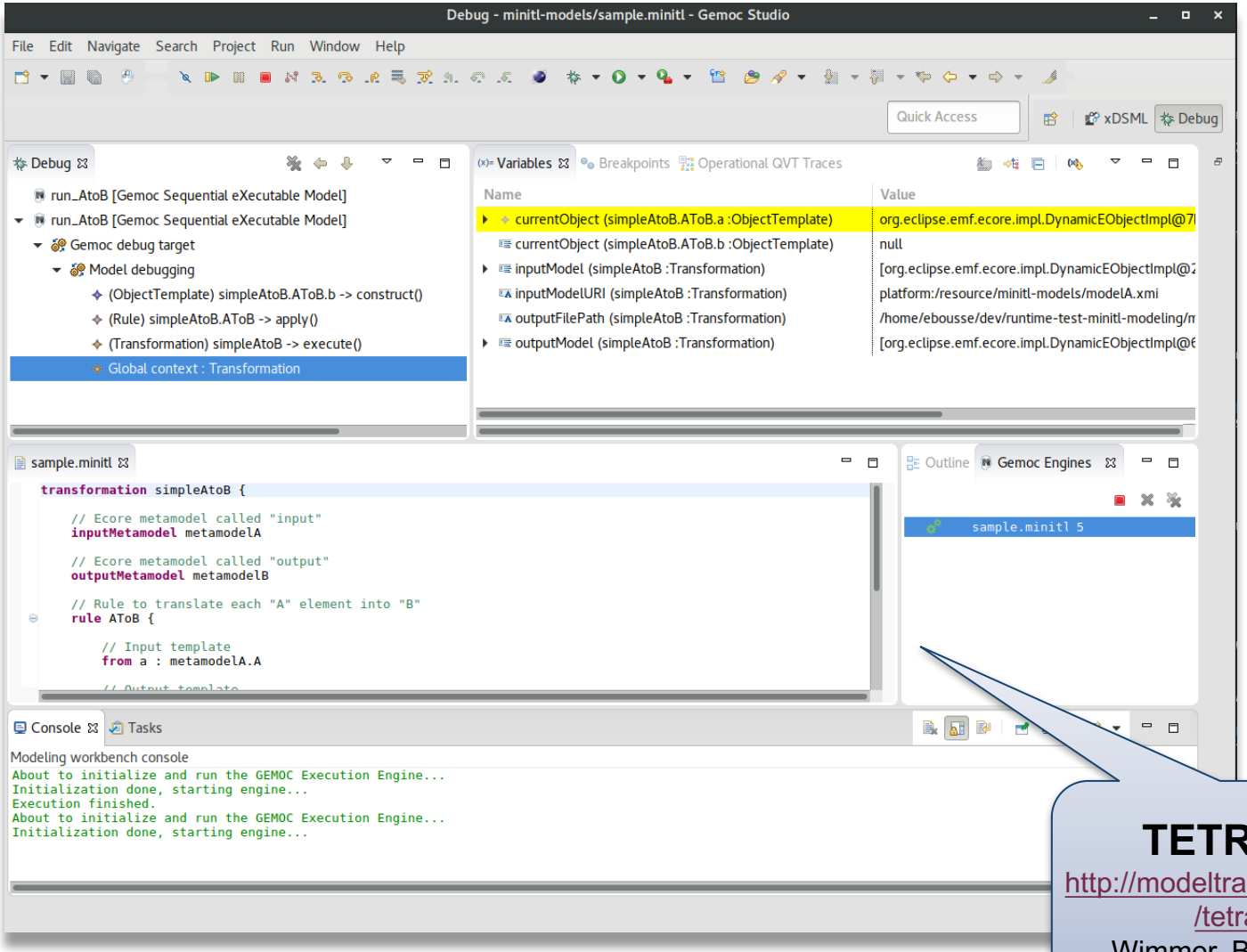
Florent Latombe, Xavier Crégut, Julien Deantoni, Marc Pantel, Benoit Combemale, "Coping with Semantic Variation Points in Domain-Specific Modeling Languages", In EXE@MoDELS 2015.

Arduino Designer (& Debugger)



https://github.com/gemoc/arduinomodeling

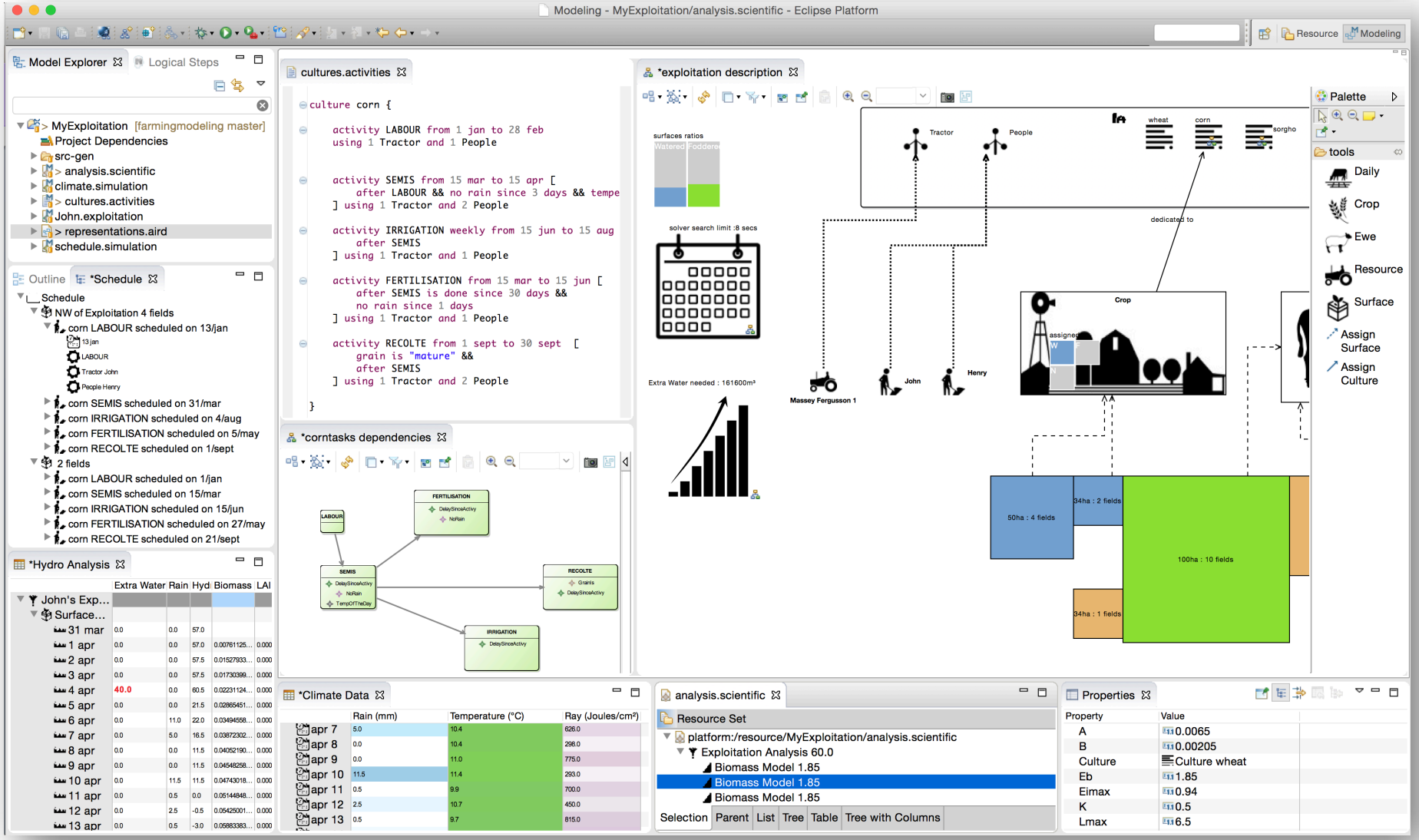
Transformation Lg Debugger



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

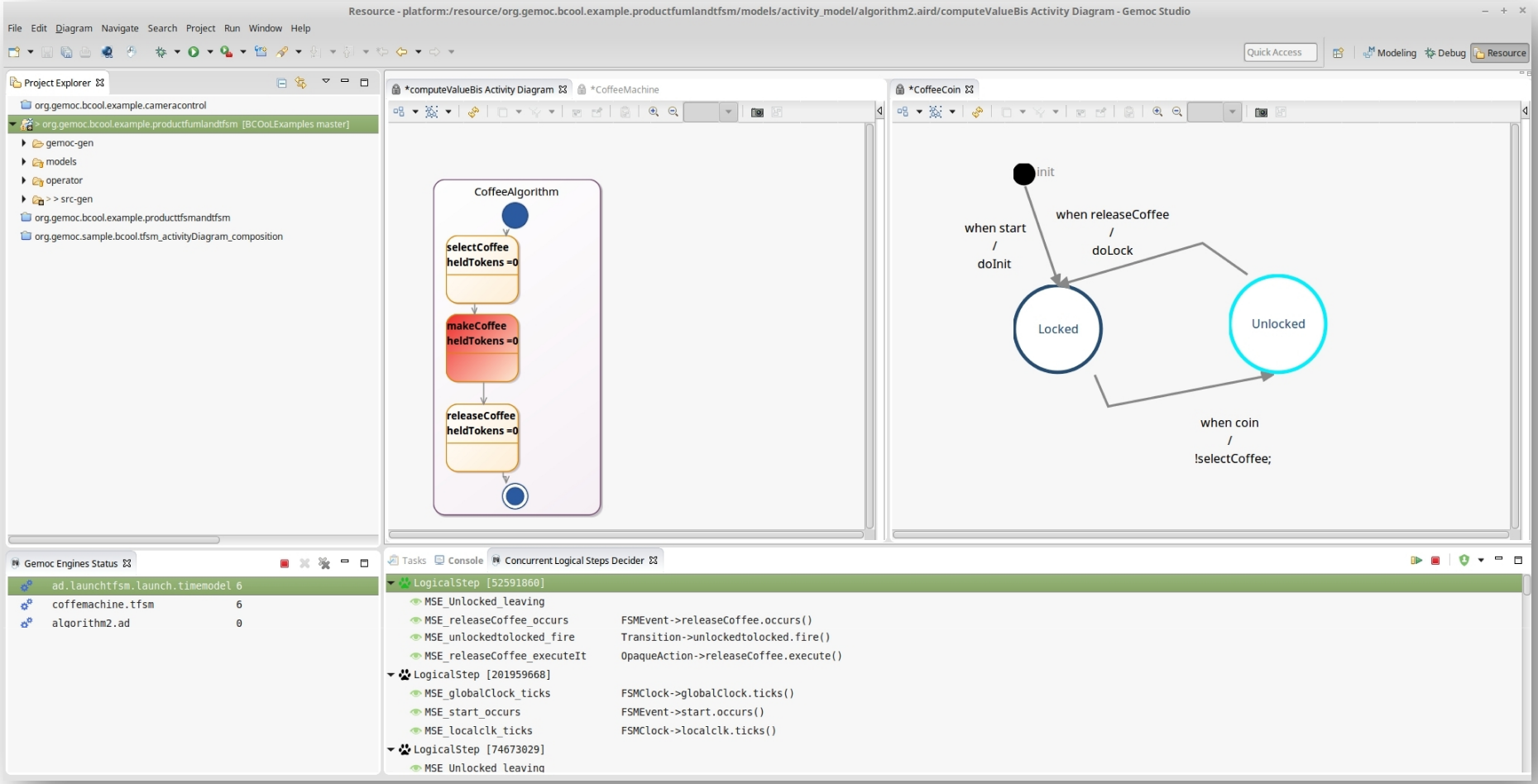
https://github.com/tetrabox/minitl

Farming System Modeling



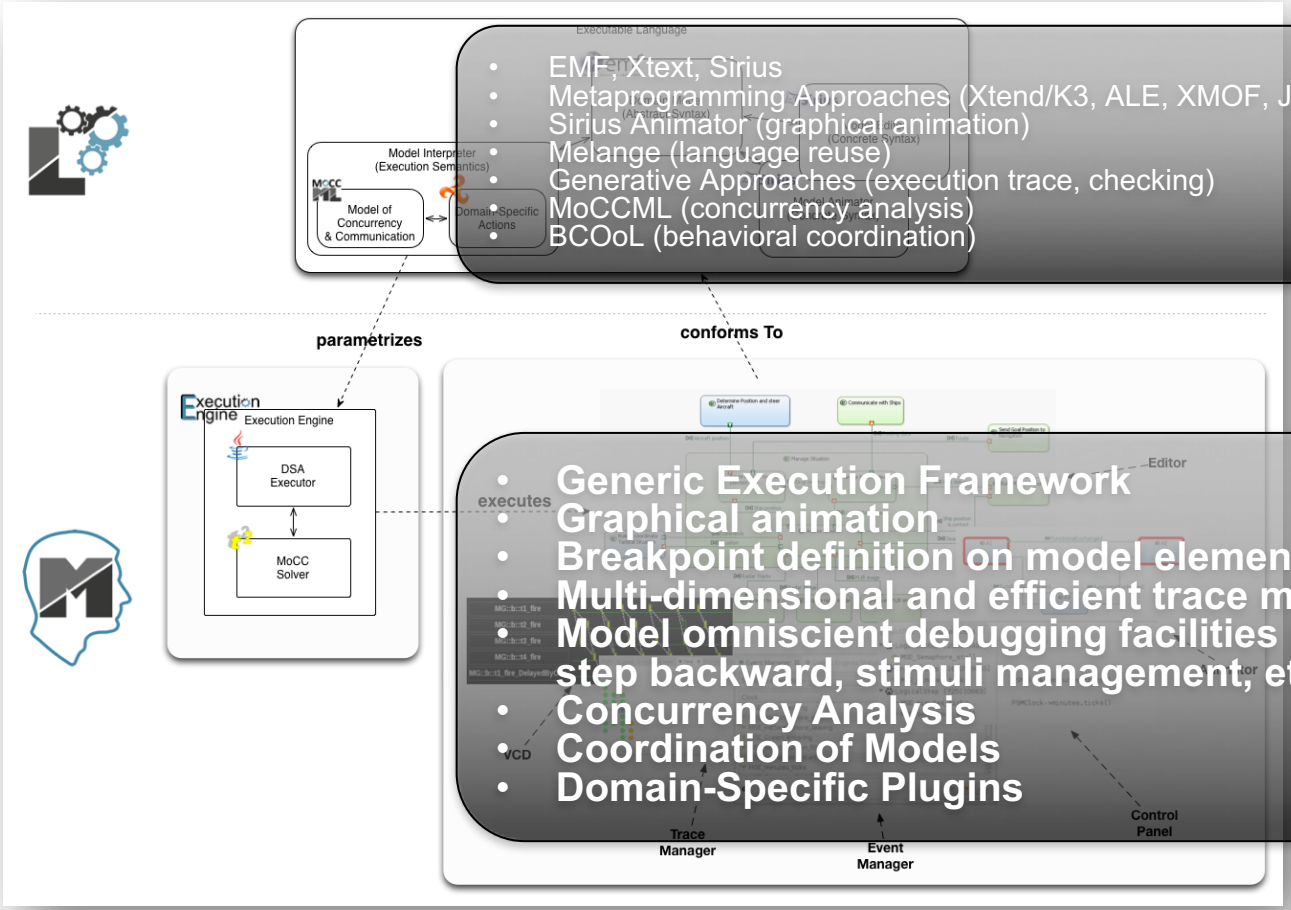
https://github.com/gemoc/farmingmodeling

Behavioral Model Coordination



Matias Ezequiel Vara Larsen, Julien Deantoni, Benoit Combemale, Frédéric Mallet, "A Behavioral Coordination Operator Language (BCOoL)," In MODELS 2015

The GEMOC Studio



- EMF, Xtext, Sirius
- Metaprogramming Approaches (Xtend/K3, ALE, XMOF, Java)
- Sirius Animator (graphical animation)
- Melange (language reuse)
- Generative Approaches (execution trace, checking)
- MoCCML (concurrency analysis)
- BCOoL (behavioral coordination)

- **Generic Execution Framework**
- **Graphical animation**
- **Breakpoint definition on model element**
- **Multi-dimensional and efficient trace management**
- **Model omniscient debugging facilities (incl., timeline, step backward, stimuli management, etc.)**
- **Concurrency Analysis**
- **Coordination of Models**
- **Domain-Specific Plugins**

Benoit Combemale, Julien Deantoni, Olivier Barais, Arnaud Blouin, Erwan Bousse, Cédric Brun, Thomas Degueule and Didier Vojtisek, "A Solution to the TTC'15 Model Execution Case Using the GEMOC Studio," In 8th Transformation Tool Contest (TTC), 2015. **Overall Winner**

http://gemoc.org/studio

Hands On

This tutorial

- See how EMF+GEMOC technologies help to (semi-automatically) build the tooling for a DSL IDE

- **Build an executable DSL**, and see how GEMOC provides
  - an execution engine,
  - A concurrency constraint solver,
  - a graphical model animator,
  - an omniscient debugger.

- Materials are available from

https://github.com/gemoc/MODELS2017Tutorial

You will learn how to

- **Build or customize your own abstractions**, or even **software languages and development environments**, to build complex, domain-specific, software-intensive systems
- **Apply language formalisms**, paradigms and principles
- **Extend a graphical, executable DSL** using state of the art techniques
- **Manage the industrial complexity** of developments and associated toolchains

A few notes on the tutorial

- Interweave a total of 2*30 min introduction / conclusion to SLE and GEMOC and 2*1h hands-on time

- Creating a sufficiently interesting, yet full-blown, modeling language including syntax, semantics, and editor/animator in 120 minutes is challenging to impossible

- To introduce the quintessential aspects of language engineering to you, your can directly access to clear workspaces at the different stages of the languages under development. Please look out for the following symbol: ⛔ on the tutorial website https://github.com/gemoc/MODELS2017Tutorial

Eclipse as a platform

- Integrated development platform with base workspace and an extensible plug-in system
- Contribute to existing IDE
  - Open structure with third party contributions
  - Update site, market place, …
- Or create your own application
  - Rich Client Platform (RCP) for general purpose applications
- **Don’t be afraid of the number of workspace projects**
  - Most of them are due to splitting the java packages for easier deployment and dependency management.

Modeling workbench vs. language workbench

- We use eclipse as a **language workbench** in which we define the constituents of and services for languages in form of **eclipse plugins**.
- From this we start another eclipse instance – the **modeling workbench**.
- Into this, we deploy the **tools and services to create models** conforming to the language(s) defined in the language workbench.

Hands on - content

- Teaser: xFSM Example
- Abstract syntax: Ecore
- Graphical concrete syntax: Sirius
- Execution semantics: Kermeta/MoCCML

2 parts:

- Breathe Life Into Your Designer!
- Weave Concurrency Constraints Into Your DSL!

BREATHE LIFE INTO YOUR DESIGNER!

Program outline (see README)

2.1. Overview the language example provided for the tutorial

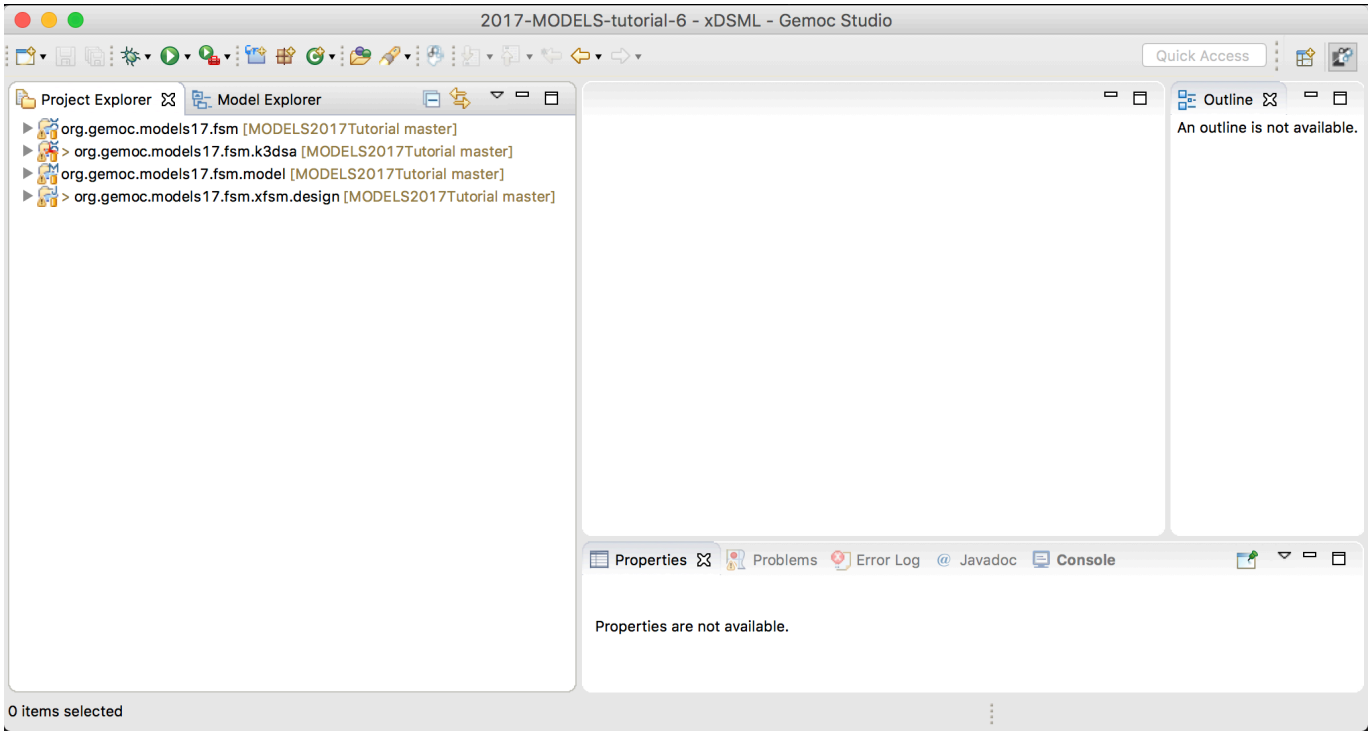
2.2. Playing with the modeling workbench

2.3. Complementing the execution semantics

2.4. Deploying the modeling workbench and playing with the example model

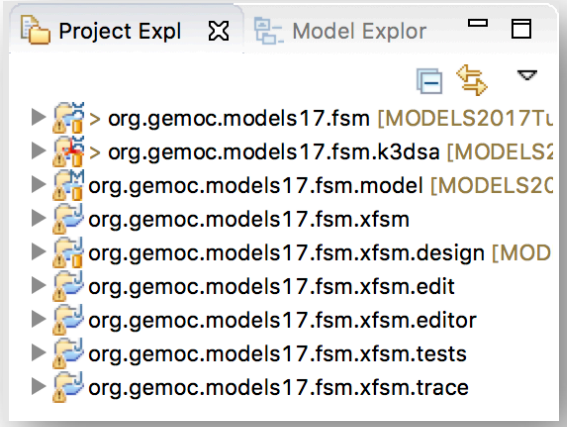
2.1. Overview the language example provided for the tutorial

1. Clone the Github repo of the tutorial
2. Start the GEMOC Studio on a fresh workspace
3. Import in your workspace the projects from /code/languagewb/*



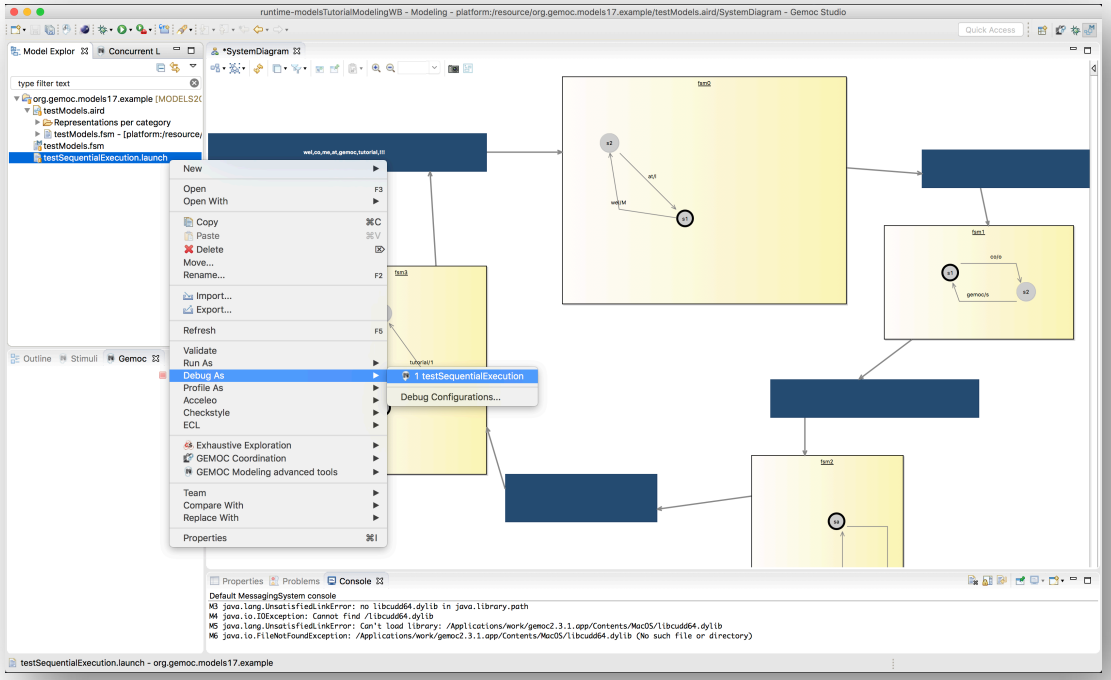
2.1. Overview the language example provided for the tutorial

4. Generate the structural interface of your language
   - Melange file -> Melange -> Generate All
5. Generate the domain-specific trace backend
   - Melange file -> GEMOC Language -> Generate Multidimensional Trace Addon project for language
6. Generate the EMF edition layer
   - Open org.gemoc.models17.fsm.xfsm/model/Model.genmodel
   - Right clic on the root -> Generate All



2.2. Playing with the modeling workbench

1. Launch a new Eclipse instance (use the launch config *modelsTutorialModelingWB*)
2. Import the model example (project code/modelingwb/ org.gemoc.models17.example)
3. Open the testModels.aird, and then the diagram (systemDiagram)
4. Start the launch configuration in debug mode



2.3. Complementing the execution semantics

Implement the methods:

- "step" on *State* which select a fireable transition according to the current state
- "fire" on *Transition* which makes the action to evolve the dynamic information related to the firing of the transition

2.3. Complementing the execution semantics

```
@Aspect(className=State)
class StateAspect {
    @Step
    def public void step(String inputString) {
            // Get the valid transitions
            val validTransitions =  _self.outgoing.filter[t | inputString.compareTo(t.trigger) == 0]

            if(validTransitions.empty) {
                    //just copy the token to the output buffer
                    _self.fsm.outputBuffer.enqueue(inputString)
            }

            if(validTransitions.size > 1) {
                    throw new Exception("Non Determinism")
            }

            // Fire transition first transition (could be random%VT.size)
            if(validTransitions.size > 0){
                    validTransitions.get(0).fire
                    return
            }
            return

    }
}
```

```
@Aspect(className=Transition)
class TransitionAspect {
    @Step
    def public void fire() {
            println("Firing " + _self.name + " and entering " + _self.tgt.name)
            val fsm = _self.src.fsm
            fsm.currentState = _self.tgt
            fsm.outputBuffer.enqueue(_self.action)
            fsm.consummedString = fsm.consummedString + fsm.underProcessTrigger
    }
}
```

2.4. Deploying the modeling workbench and playing with the example model

- Do 2.1 > step 4 (each time you modify the semantics)
- Do 2.2
- See the diff ☺

WEAVE CONCURRENCY CONSTRAINTS INTO YOUR DSL!

Julien ;)

(see https://github.com/gemoc/MODELS2017Tutorial)

CONCLUSION

Conclusion

- Scientific breakthroughs:
  - A concurrent and modular executable metamodeling approach
    - ➢ Cross-fertilization of the algorithm theory and the concurrency theory
  - An explicit behavioral language interface
  - The reification of the coordination concerns at the language level

- Technological breakthroughs:
  - Dedicated meta-languages integrated into the GEMOC Studio, atop Eclipse Modeling
  - Execution environment integrated within the Eclipse debug UI, incl. graphical animation, omniscient debugging, concurrency analysis and behavioral coordination

Conclusion

- Software components:
  - **Sirius Animator**: execution engine, animator designer/runtime, omniscient debugger, and trace/event managers
    - Host on ***`Sirius lab`*** for maturation as an Eclipse plugin (Obeo/INRIA)
    - ➢ *https://github.com/SiriusLab/ModelDebugging*
  - **MoccML**
    - will be diffused as an open source project (I3S/ENSTA Bretagne)
    - ➢ *https://github.com/gemoc/concurrency*
  - **BCOoL** and heterogeneous engine coordination
    - will be diffused as an open source project (I3S/INRIA)
    - ➢ *https://github.com/gemoc/coordination*
  - **GEMOC studio**: language and modeling workbench, wizard/dashboard, documentation and examples
    - ➢ *https://github.com/gemoc/gemoc-studio*

Ongoing Developments

- Deep investigation of the notion of language interface (viewpoint engineering, etc.)
- Domain-specific property language, incl. for breakpoint definition
- Simulation, model explorer, model checking
- Coordination of discrete and continuous models
- Co-simulation (incl., FMI)
- Live and collaborative modeling (e.g., for sustainability systems)

Perspectives

- Formal analysis of model coordination
- Adaptable MoC at the language level
  - @design/compile time: design space exploration, optimizing compilers
  - @runtime: code adaptation, code obfuscation
- SLE in Education

Hack your own languages?

Join us in the MDE/SLE group of the CNRS IRIT lab, in a freshly rebuilt campus of the warm city of Toulouse!

Open Positions for PhD and Postdoc

BENOIT COMBEMALE
PROFESSOR, UNIV. TOULOUSE, FRANCE

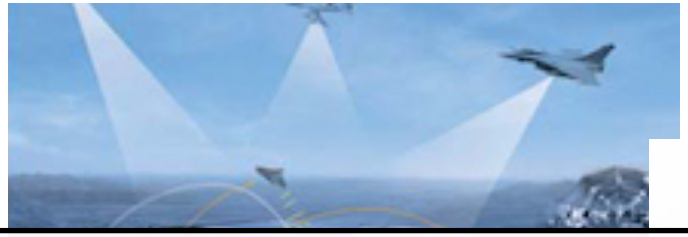
HTTP://COMBEMALE.FR
BENOIT.COMBEMALE@IRIT.FR
@BCOMBEMALE

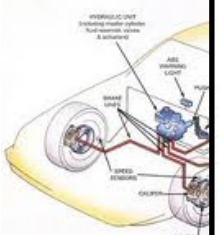
Université Fédérale Toulouse Midi-Pyrénées

UNIVERSITÉ TOULOUSE Jean Jaurès

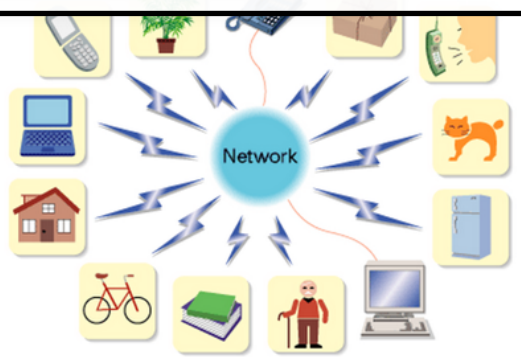
IRIT CNRS - INPT - UPS - UT1 - UT2J

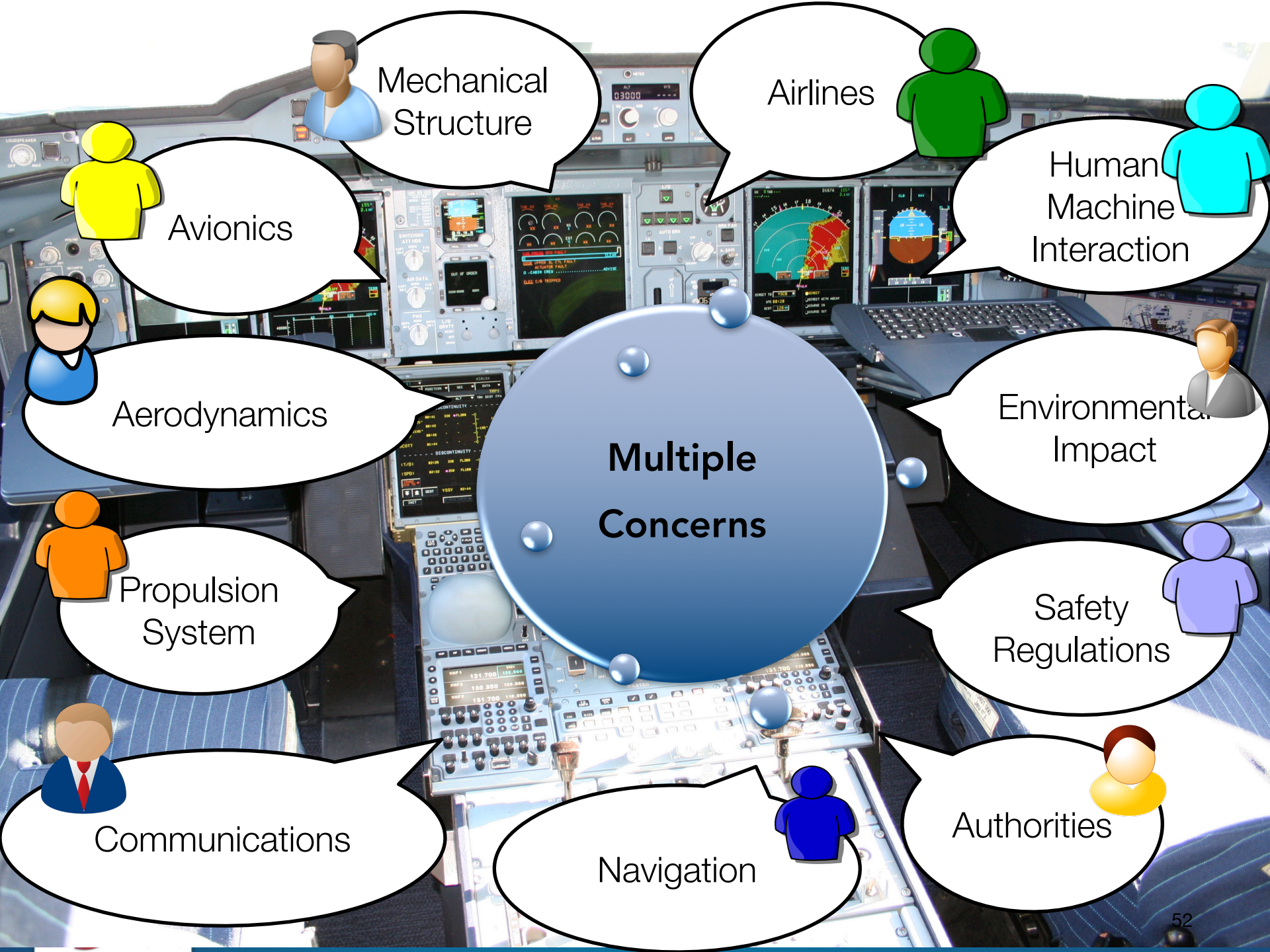
Complex Software-Intensive Systems



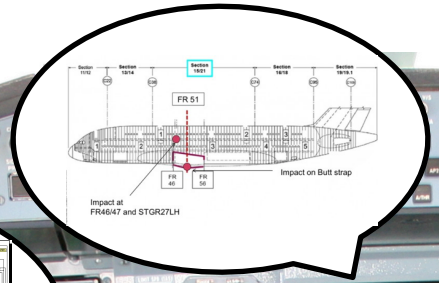


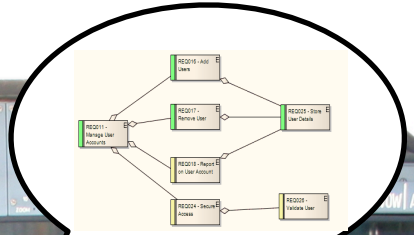
- Multi-engineering approach
- Domain-specific modeling
- High variability and customization
- Software as integration layer
- Openness and dynamicity

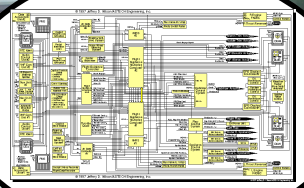


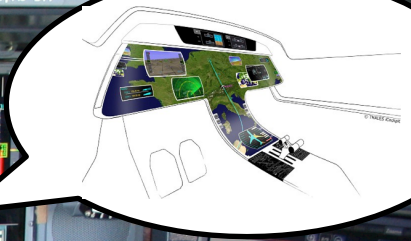

Mechanical Structure
Airlines
Avionics
Human Machine Interaction
Aerodynamics
Multiple Concerns
Environmental Impact
Propulsion System
Safety Regulations
Communications
Navigation
Authorities

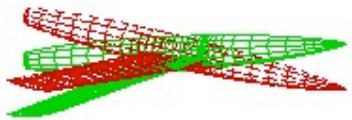
Heterogeneous Modeling

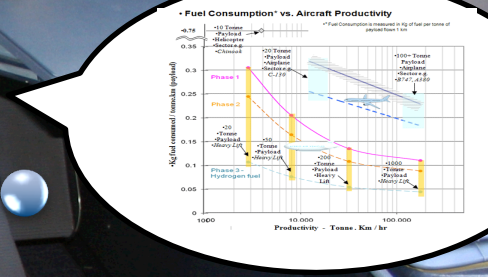


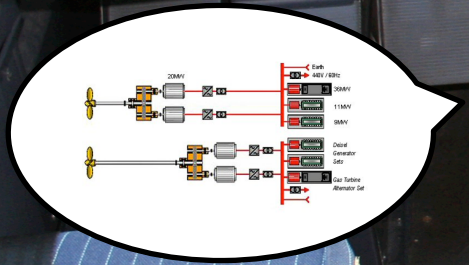


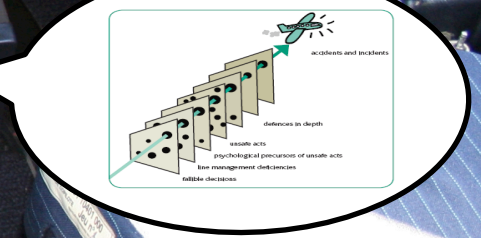


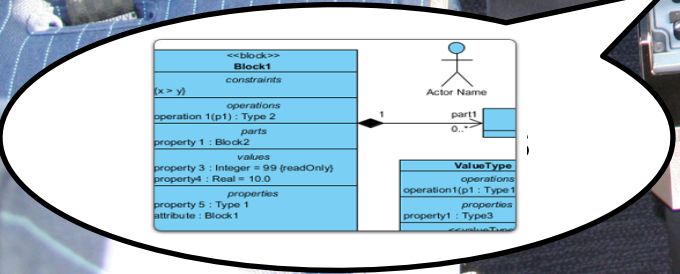


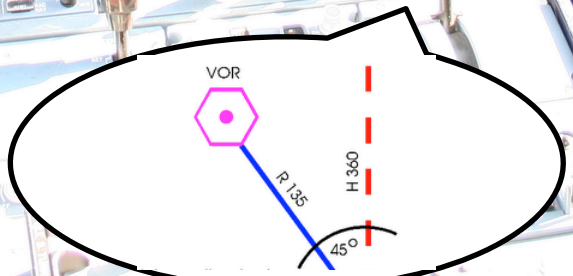


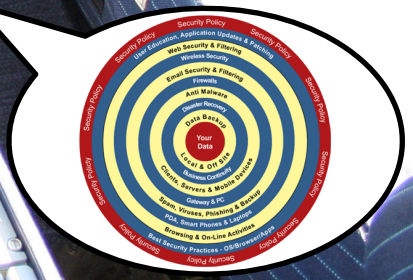




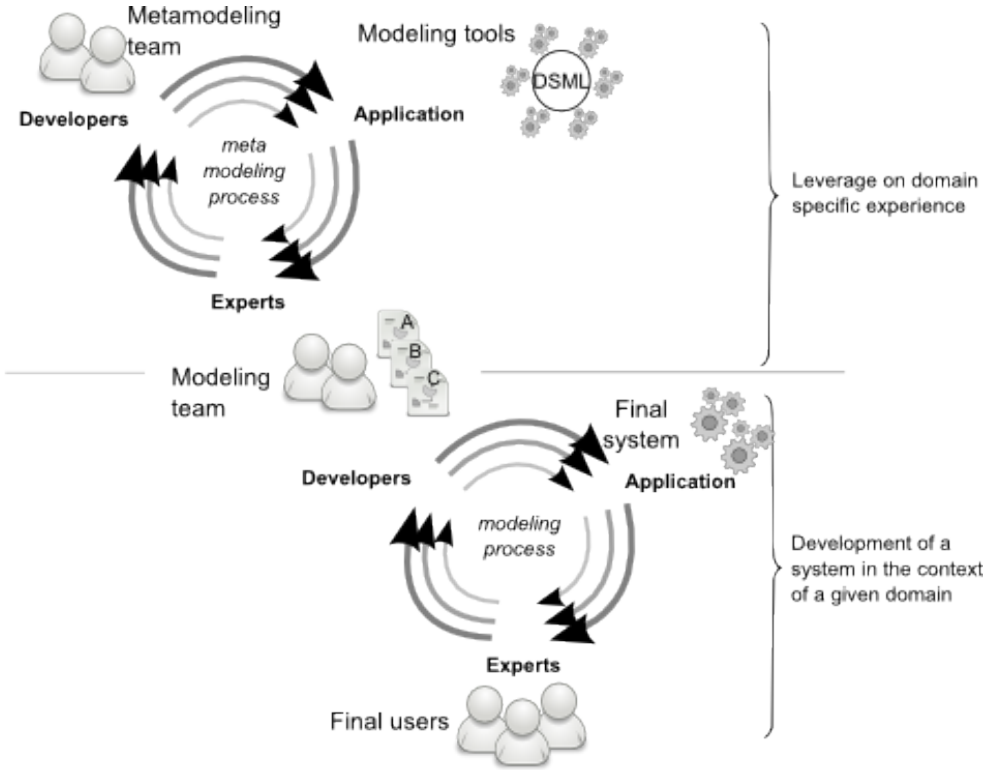




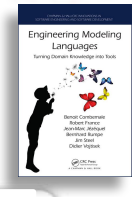




Language-Oriented Programming & Modeling

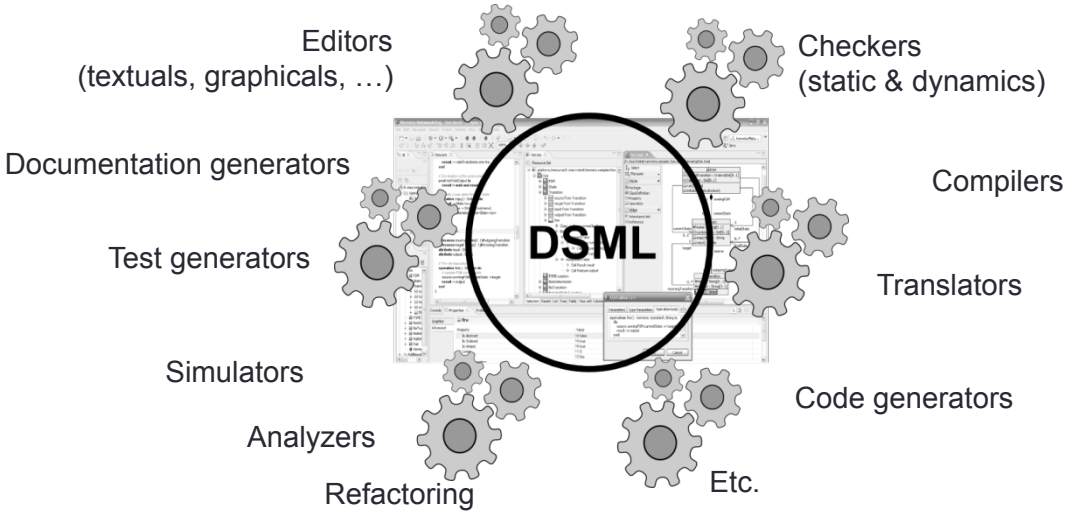


Ingénierie Dirigée par les Modèles : des concepts à la pratique, by Jézéquel, J.-M., Combemale, B., Vojtisek, D., Références sciences, ellipses (Eds.). 2012.

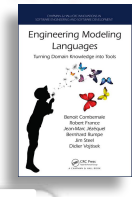


Engineering Modeling Languages: Turning Domain Knowledge into Tools, by Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim R.H. Steel, and Didier Vojtisek. Chapman and Hall/CRC, pp.398, 2016. Companion website: http://mdebook.irisa.fr

Language Design and Implementation



Ingénierie Dirigée par les Modèles : des concepts à la pratique by Jézéquel, J.-M., Combemale, B., Vojtisek, D., Références sciences, ellipses (Eds.). 2012.



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