Reifying Concurrency in Language Design and Implementation

Benoit Combemale (Inria & Univ. Rennes 1)
http://people.irisa.fr/Benoit.Combemale
benoit.combemale@irisa.fr
@bcombemale

Gemoc

DiverSE Diversity-Centric Software Engineering

UNIVERSITÉ DE RENNES 1

UMR IRISA

esir ECOLE SUPERIEURE D'INGENIEUR DE RENNES

Inria

Outline

1. Explicit and Formal Concurrency in Executable DSML
2. Behavioral Coordination of Heterogeneous Models

EXPLICIT AND FORMAL CONCURRENCY IN EXECUTABLE DSML

Motivations

- Concurrency is at the heart of modern software-intensive systems and platforms
  - Complex systems (e.g., IoT, CPS) are highly concurrent systems per se.
  - Modern platforms (e.g., many-core, GPGPU, distributed platforms) are providing more and more parallelism.

⇒ Require correct specification of concurrency for leveraging the unique characteristics of these systems and their deployment on these platforms.

Motivations

- 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

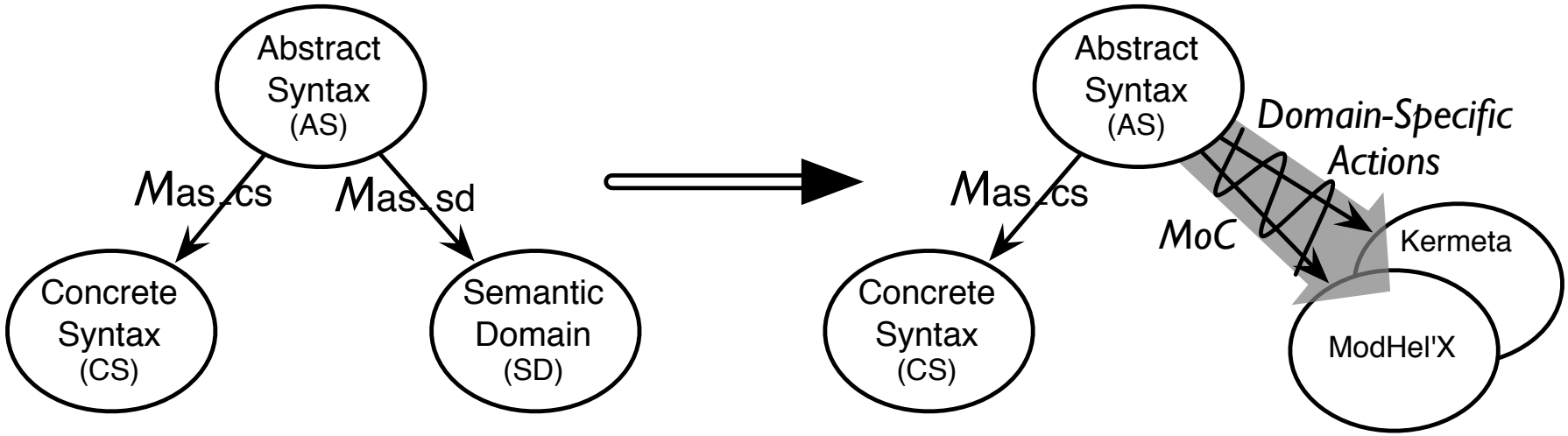
Grand Challenge

Cross fertilization of the **language** theory and the **concurrency** theory

"Concurrency models were generally event-based, and avoided the use of state. They did not easily describe algorithms or the usual way of thinking about them based on the standard model."

Leslie Lamport, “Turing Lecture: *The Computer Science of Concurrency: The Early Years*,” Com. ACM, vol. 58, no. 6, 2015, pp. 71–76.

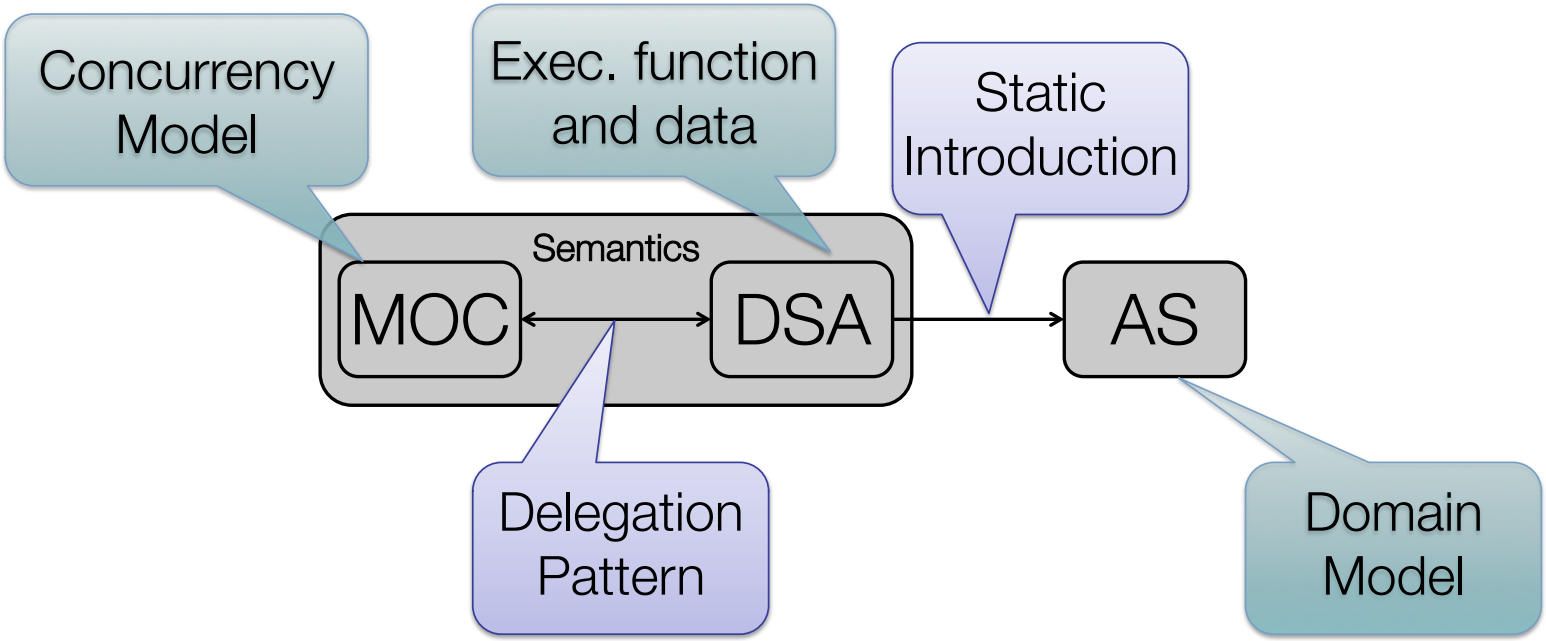
Approach



Benoit Combemale, Cécile Hardebolle, Christophe Jacquet, Frédéric Boulanger, Benoit Baudry, "Bridging the Chasm between Executable Metamodeling and Models of Computation," In SLE 2012.

Approach

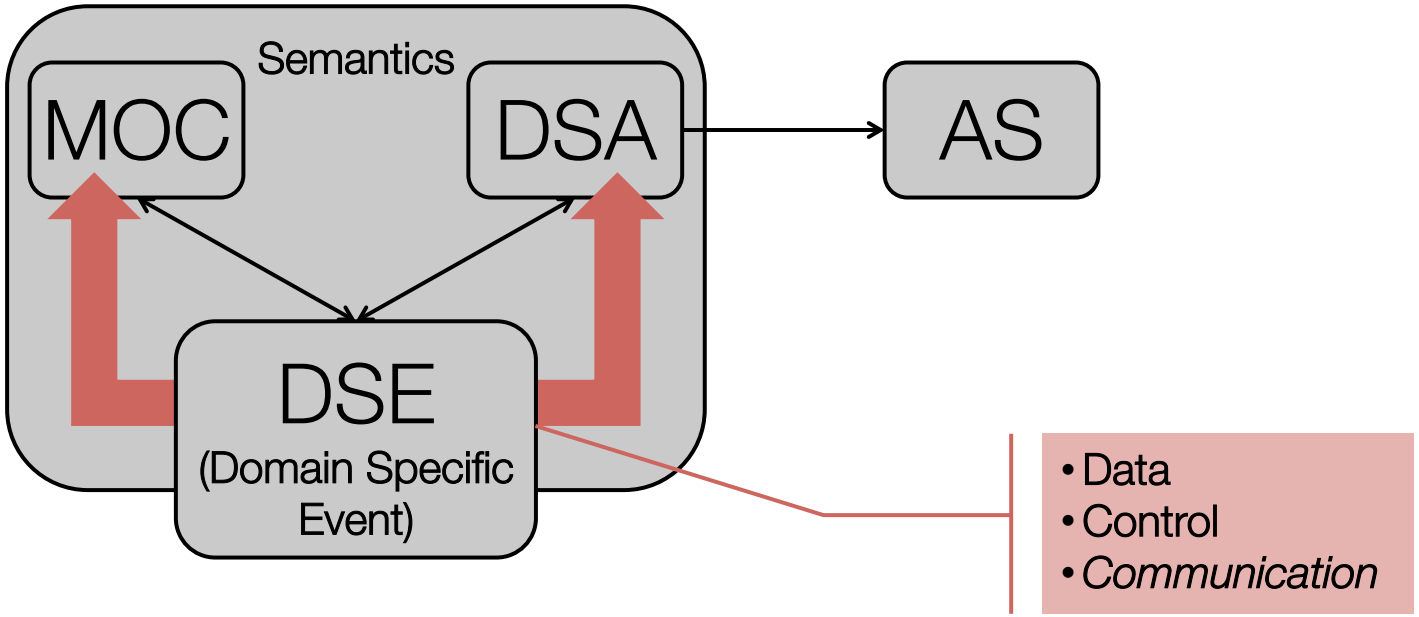
The MoCC serves as a (family of) scheduler(s) of the execution functions that manipulate the execution data (i.e. program state)



Benoit Combemale, Cécile Hardebolle, Christophe Jacquet, Frédéric Boulanger, Benoit Baudry, "Bridging the Chasm between Executable Metamodeling and Models of Computation," In SLE 2012.

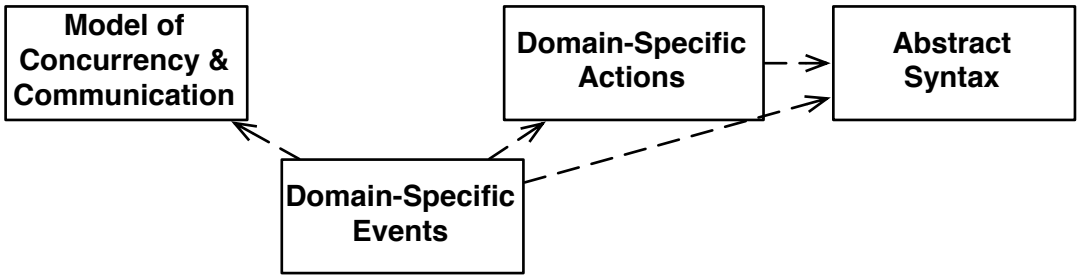
Approach

The DSE serve as a mapping between the MOC and the DSA

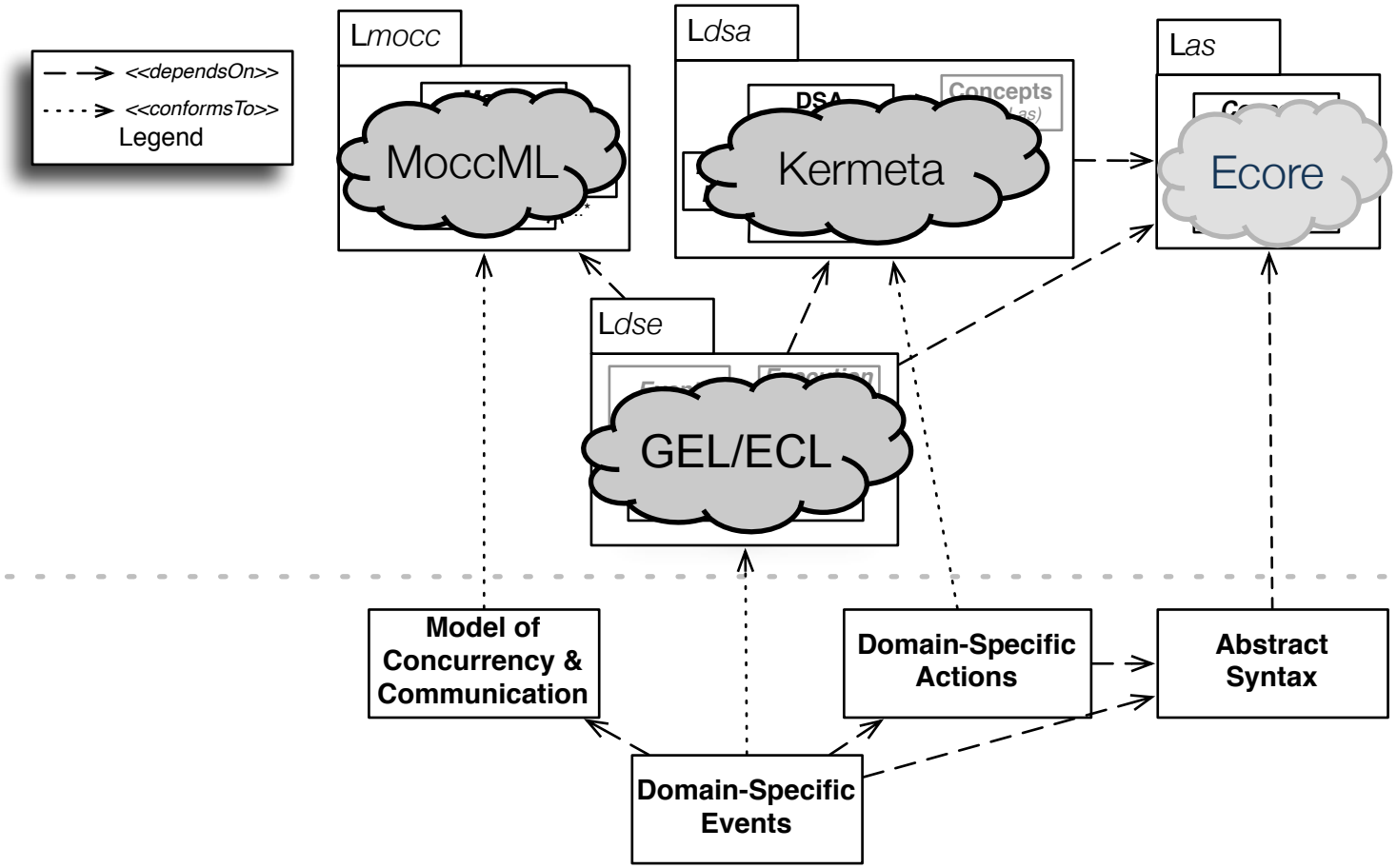


Benoit Combemale, Julien Deantoni, Matias Vara Larsen, Frédéric Mallet, Olivier Barais, Benoit Baudry, Robert France, "Reifying Concurrency for Executable Metamodeling," In SLE 2013

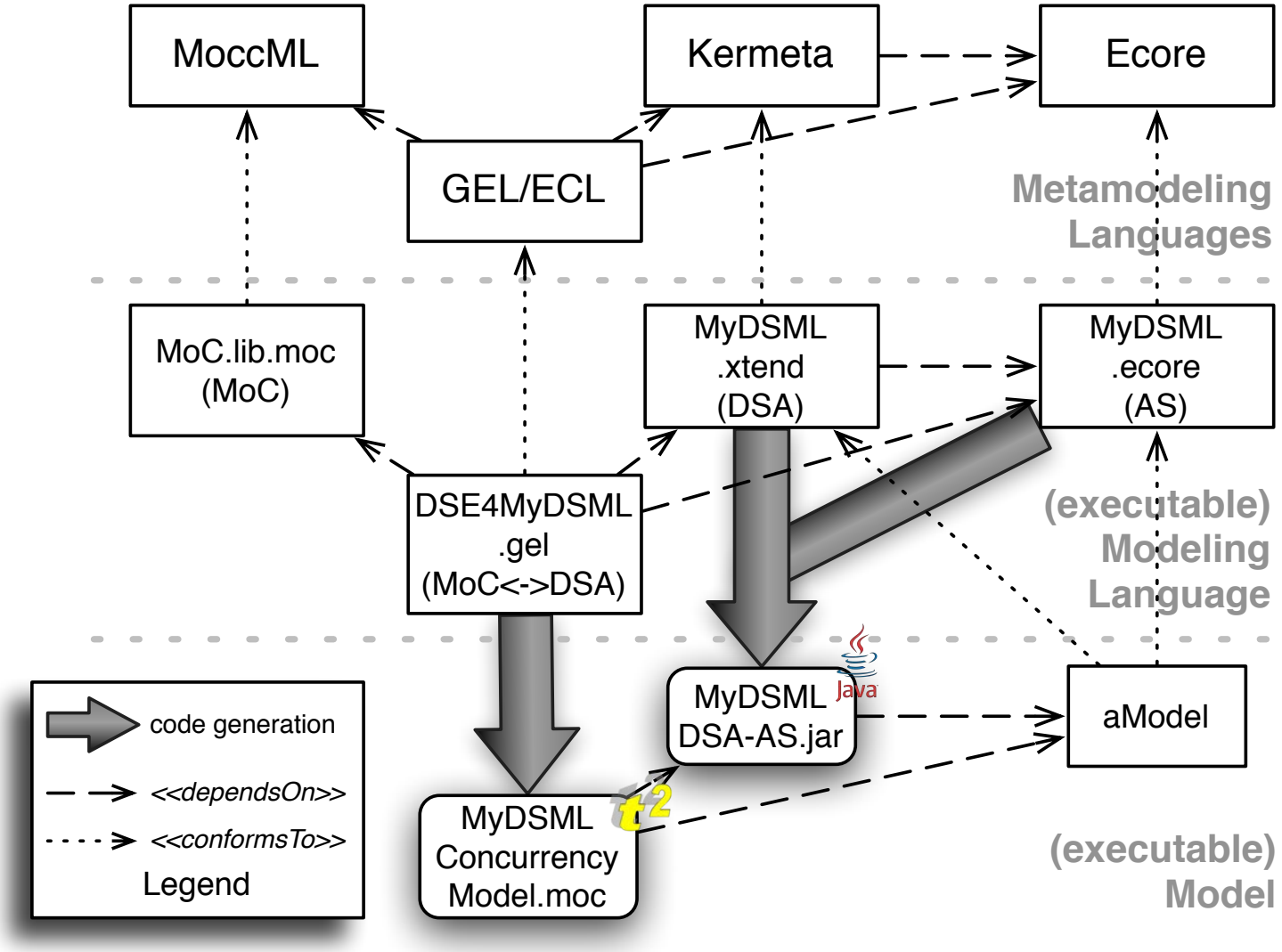
Contribution



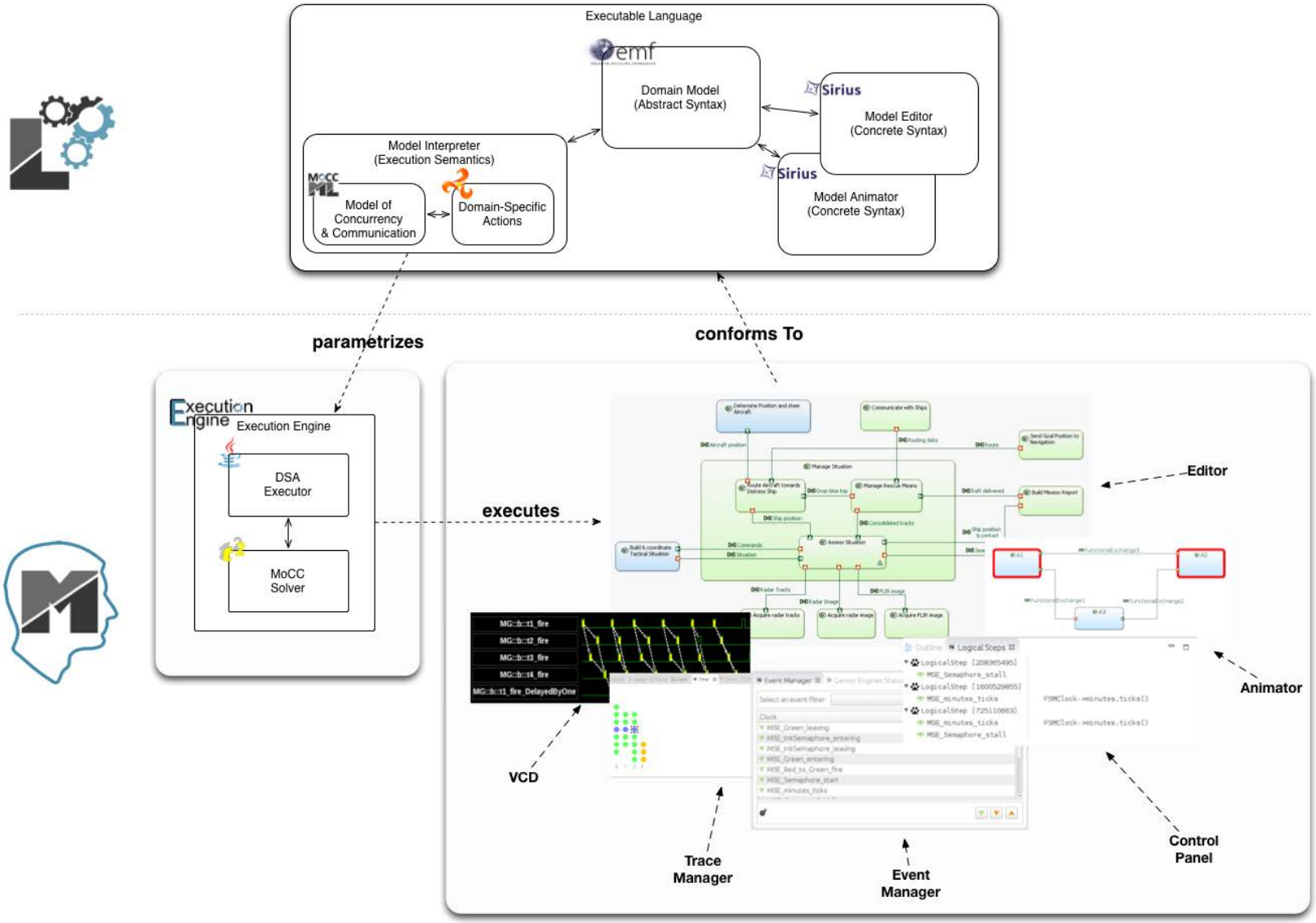
Contribution



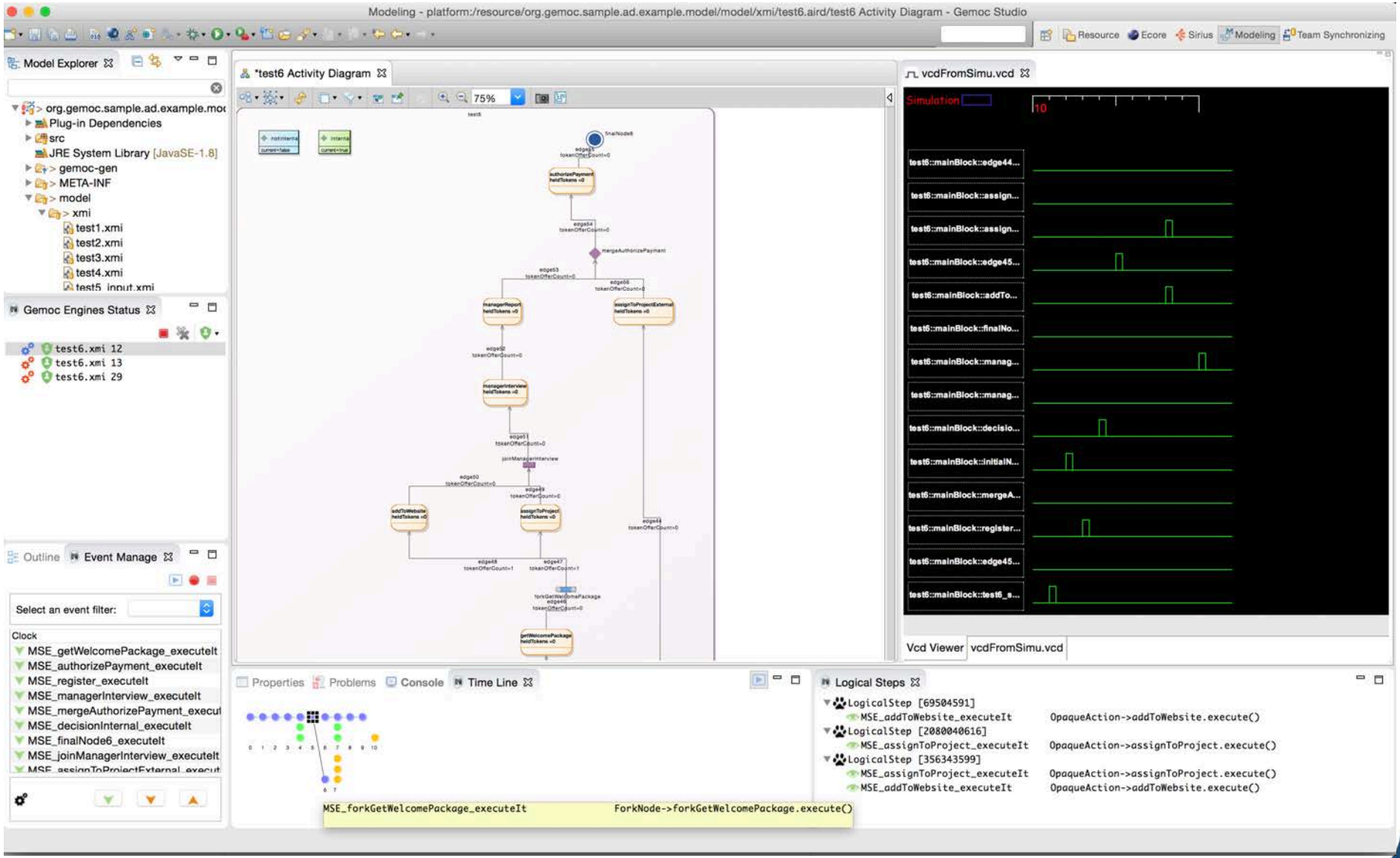
Contribution



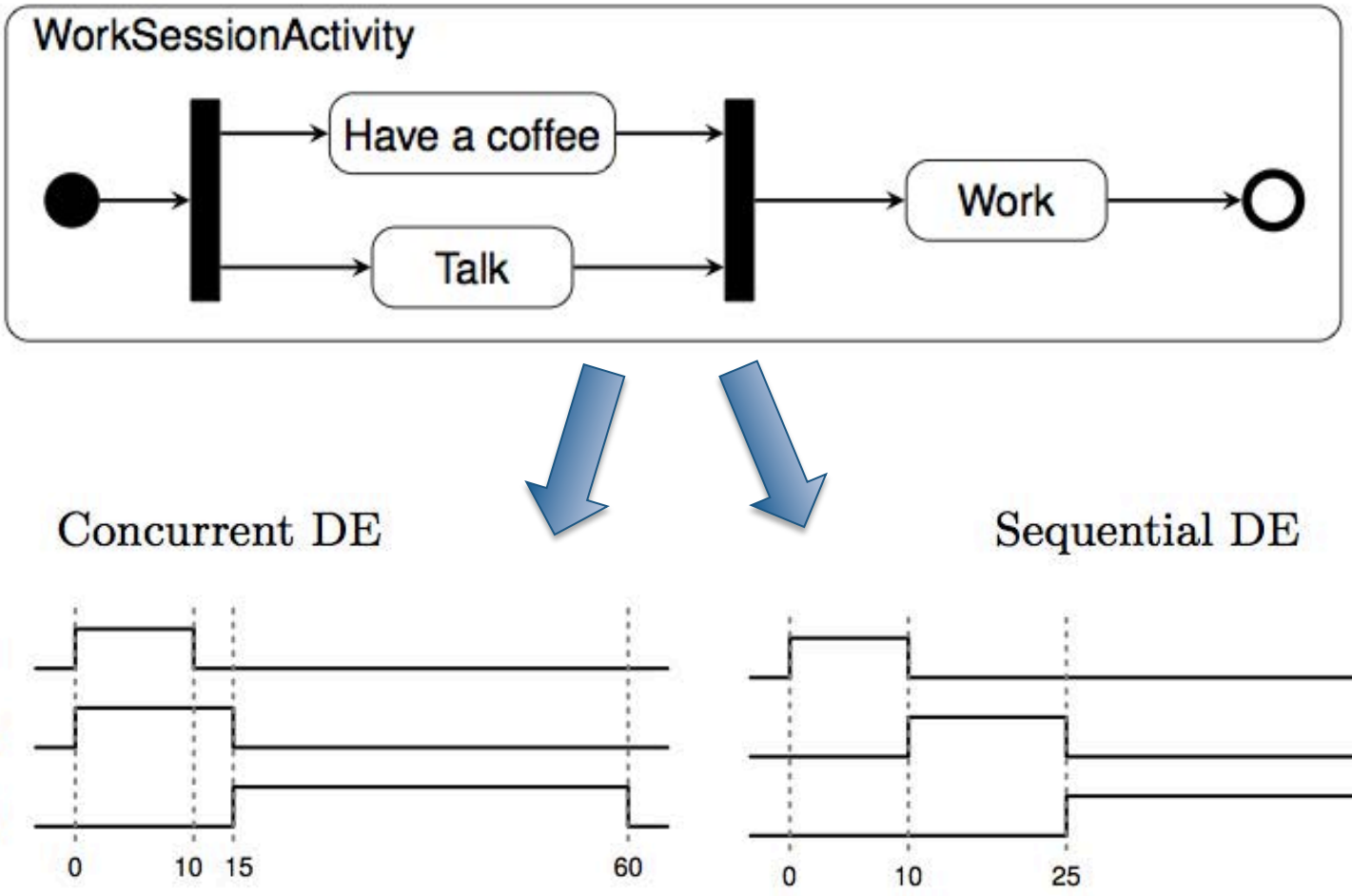
Result: Model Execution, Simulation and Animation



Result: Model Execution, Simulation and Animation



Result: Semantic Variation Points



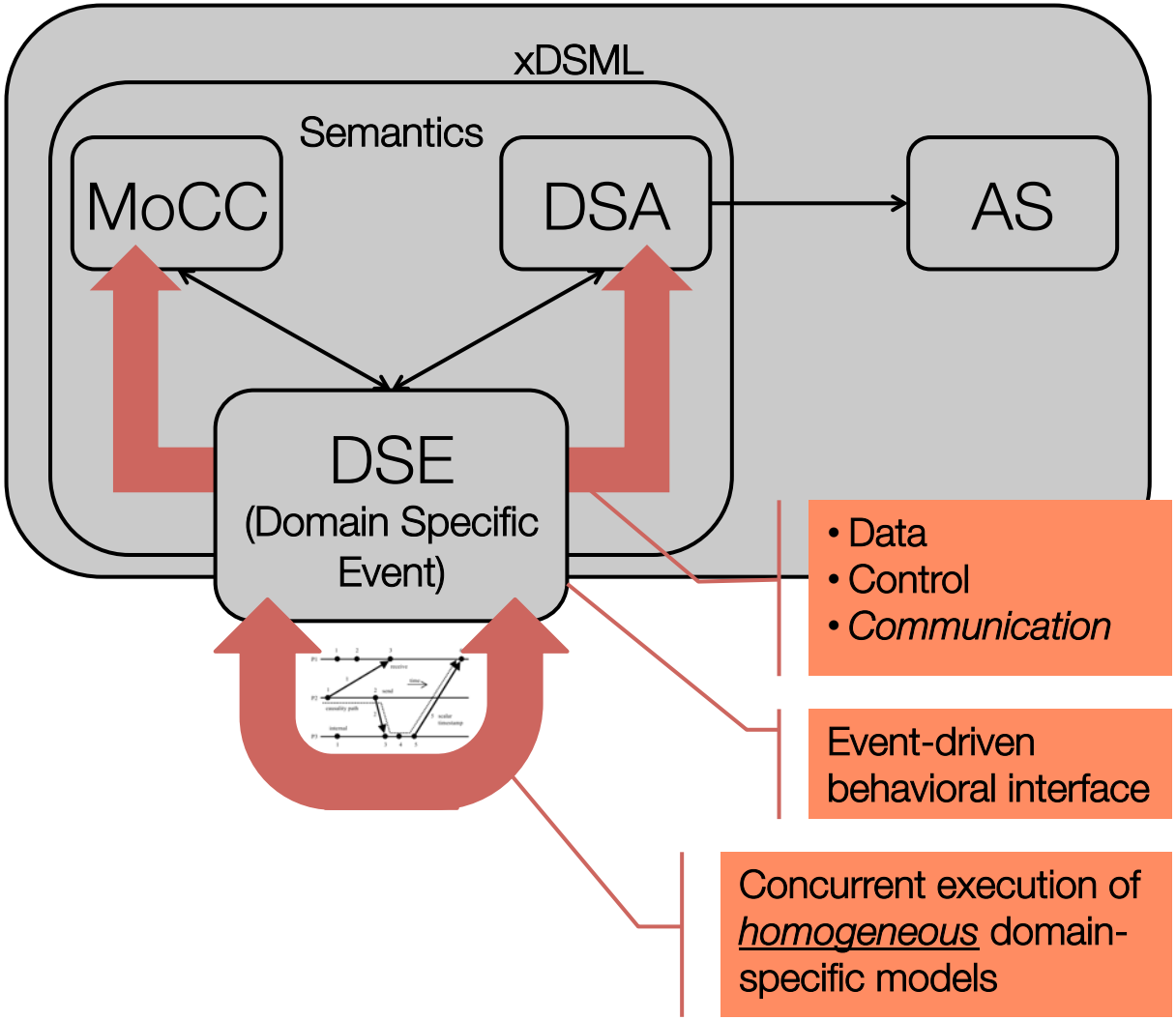
BEHAVIORAL COORDINATION OF HETEROGENEOUS MODELS

Towards a behavioral language interface

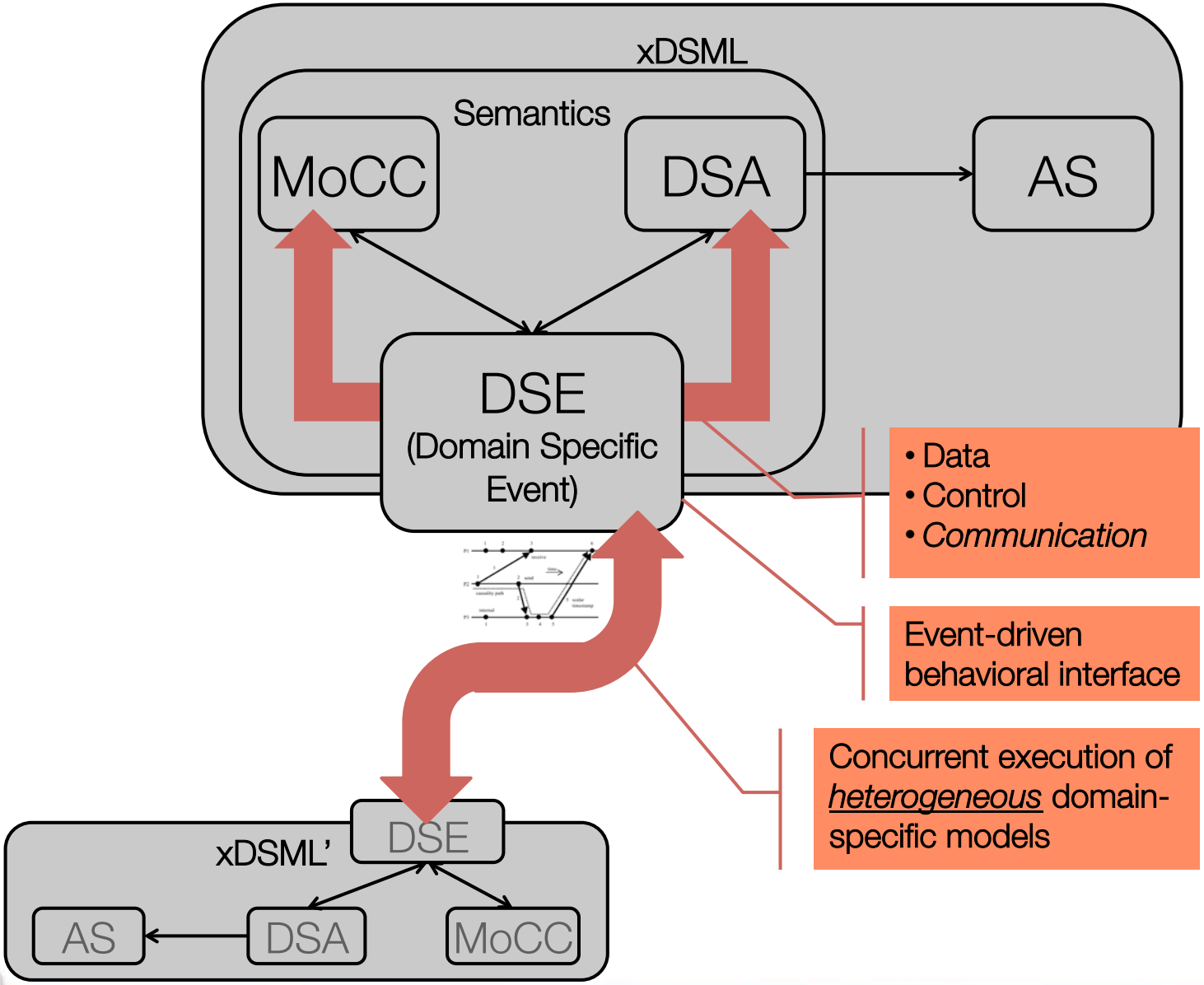
- The application of the MOC to a given model results in an event structure
- Consequently, the MOC define a symbolic event structure, whose the events mapped to the DSA correspond to the visible model state changes

⇒ This mapping can serve as a behavioral language interface used to define patterns that will coordinate conforming models

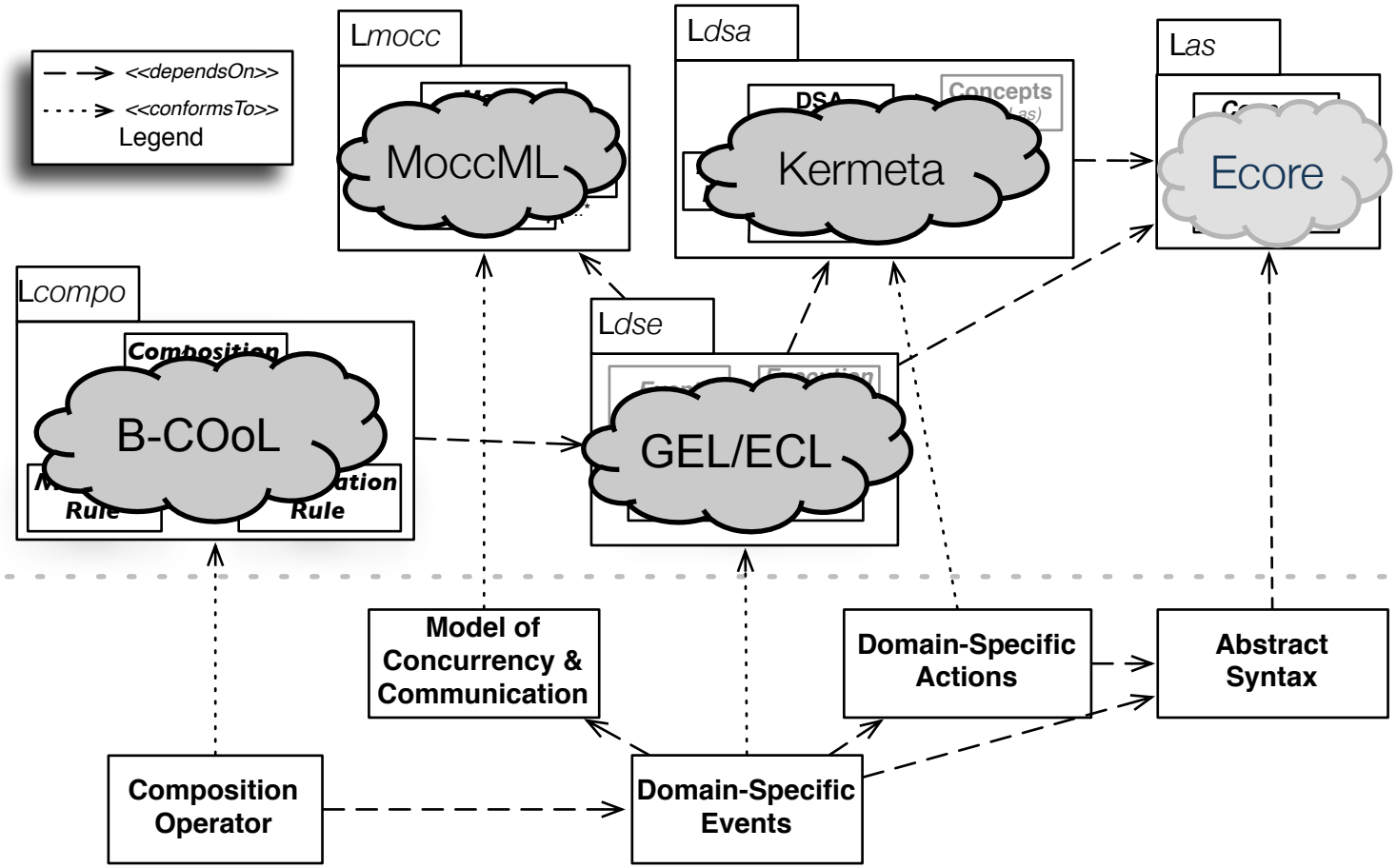
Approach


xDSML
Semantics
MoCC
DSA
AS
DSE
(Domain Specific
Event)
• Data
• Control
• *Communication*
Event-driven
behavioral interface
Concurrent execution of
homogeneous domain-
specific models

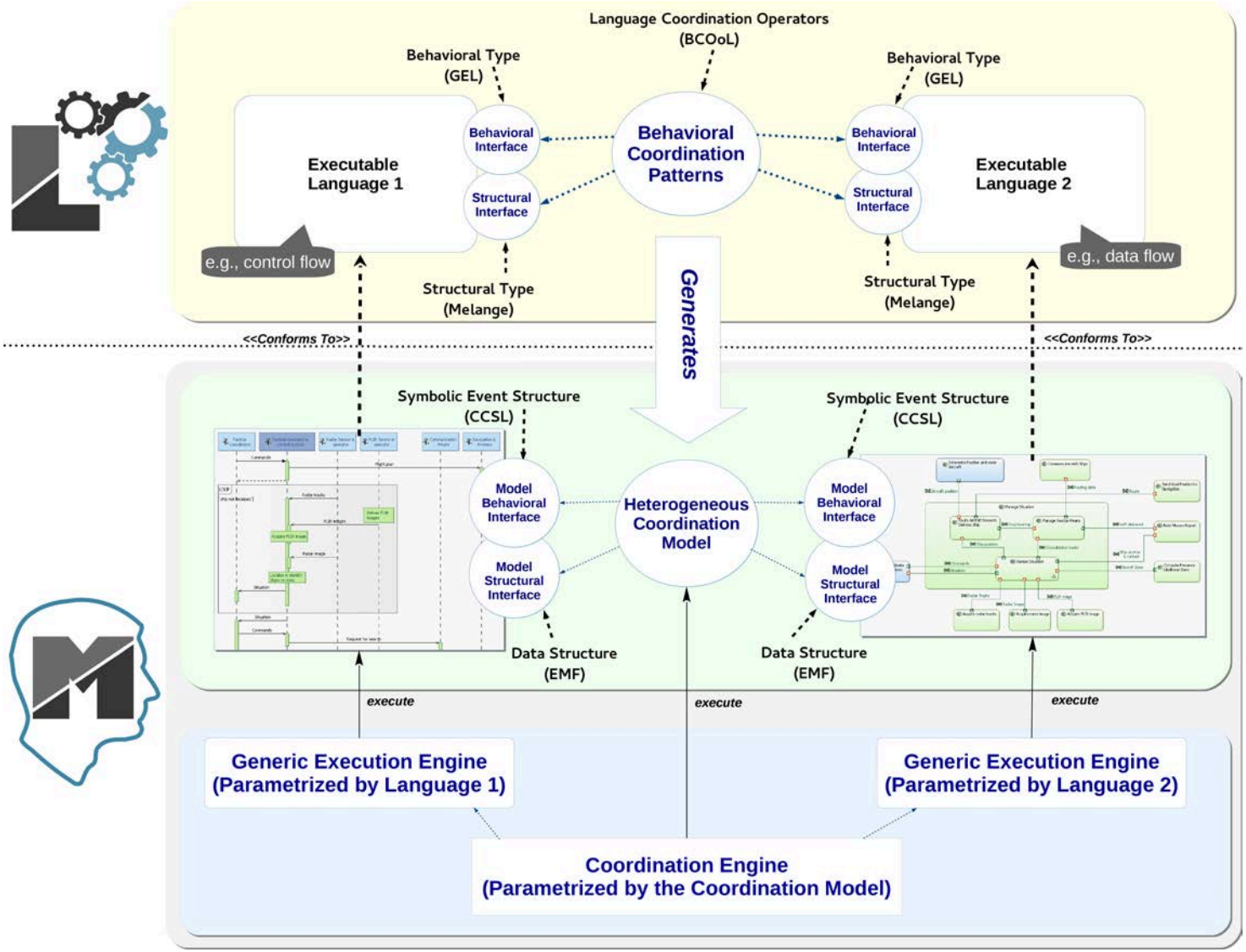
Approach


xDSML
Semantics
MoCC
DSA
AS
DSE
(Domain Specific
Event)
• Data
• Control
• *Communication*
Event-driven
behavioral interface
Concurrent execution of
heterogeneous domain-
specific models
xDSML'
DSE
AS
DSA
MoCC

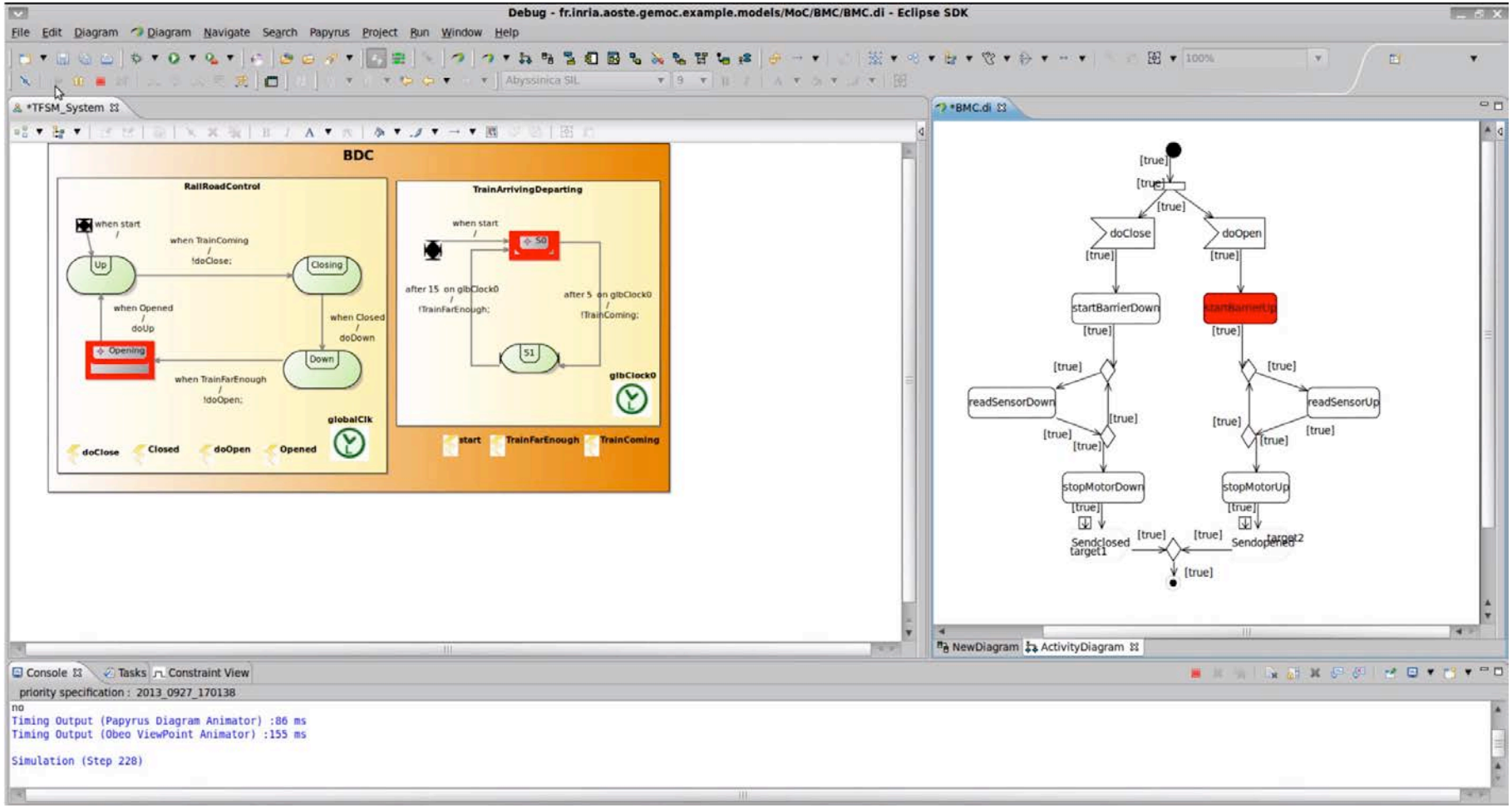
Contribution



Towards a behavioral language interface



Model Coordination



Conclusion / Open issues

- Conclusion
  - A concurrent and modular executable metamodeling approach
  - An explicit behavioral language interface
  - A behavioral coordination operator language

- Open issues
  - Formalization of the mapping between the MOC and the DSA
  - Formalization of coordination patterns