SLE 2012
Dresden, Germany
5th International Conference on Software Language Engineering

Bridging the Chasm between Executable Metamodeling and Models of Computation

Benoit Combemale, Cécile Hardebolle, Christophe Jacquet,
Frédéric Boulanger, Benoit Baudry

E3S
Supelec
Systems
Science

Supélec

E3S
Supelec
Systems
Science

UMR IRISA

Inria
informatics mathematics

Outline

① Context: DSLs and their semantics

② Illustrating DSL example: fUML

③ Our "bridging" approach, illustrated on fUML

- Overview of the approach
- Executable metamodeling and Domain Specific Actions (DSAs)
- Models of Computation (MoCs)
- Bridge

④ Demo

⑤ Discussion and conclusion

Context: DSLs

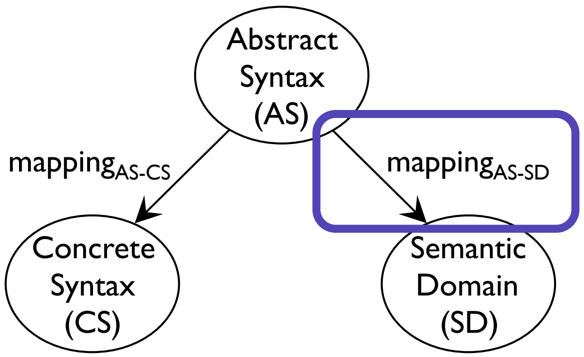
- Domain Specific Language (DSL) = language with a limited and dedicated set of concepts, designed for domain experts to express concerns about a system

- DSLs are successful
  - *[Karna et al.]* limited expressiveness + dedicated tools =
    ➡ productivity increase when building software-intensive systems
    ➡ reduction in the number of errors
  - *[Hutchinson et al.]* DSLs make the industrial adoption of model-driven engineering easier

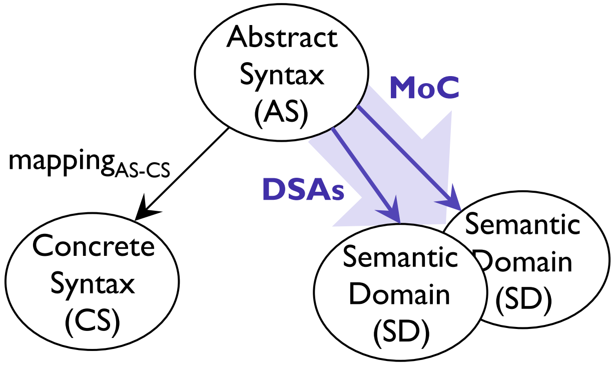
- The (formal) definition of the semantics of DSLs is necessary to benefit from tool generation, formal analysis, model execution, etc. but is a major difficulty *[Bryant et al.]*

How to define a DSL?

- *[Harel et al.]* DSL = abstract syntax
  + concrete syntax
  + semantic domain



- Our contribution = decomposition of the mapping$_{AS\text{-}SD}$ in two parts
  - Domain-Specific Actions (DSAs): semantics of domain specific concepts
  - Model of Computation (MoC): communication, concurrency and time semantics (≈ scheduling of DSAs)



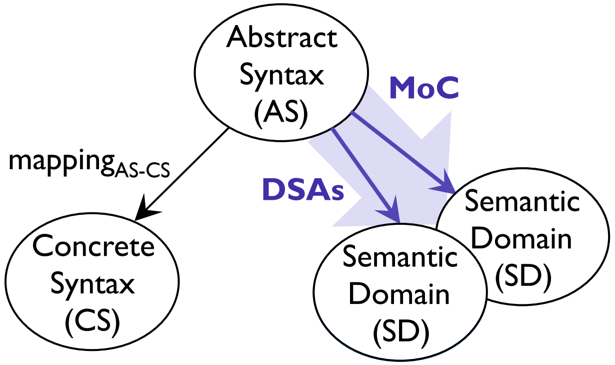
- Benefit = ➡ reuse of the MoC in different DSLs
  ➡ variations of a given DSL by varying the MoC

The "bridging" approach

- The "bridging" approach = decomposition of the mapping between abstract syntax and semantic domain in two parts
  - Domain-Specific Actions (DSAs): semantics of domain specific concepts
  - Model of Computation (MoC): communication, concurrency and time semantics (≈ scheduling of DSAs)

Weave executable semantics on metamodel elements (executable metamodeling)
➡ Kermeta

Define, reuse and compose the executable semantics of MoCs
➡ ModHel'X

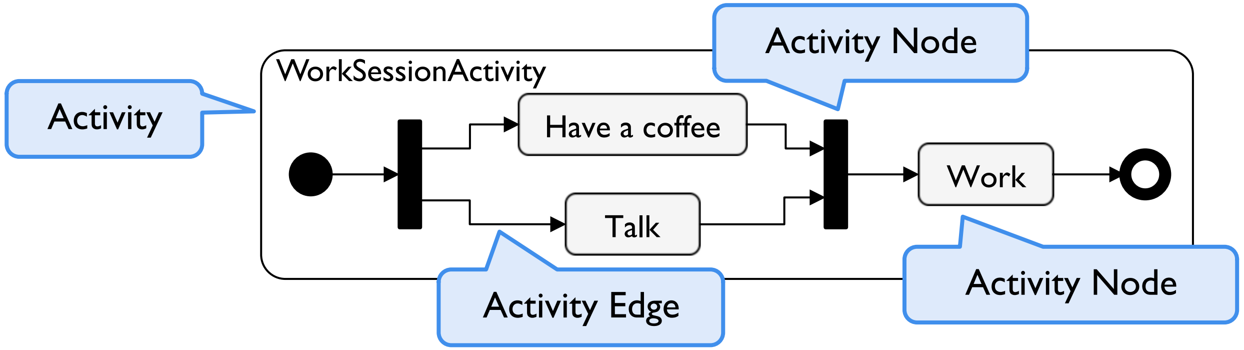


DSL example: fUML

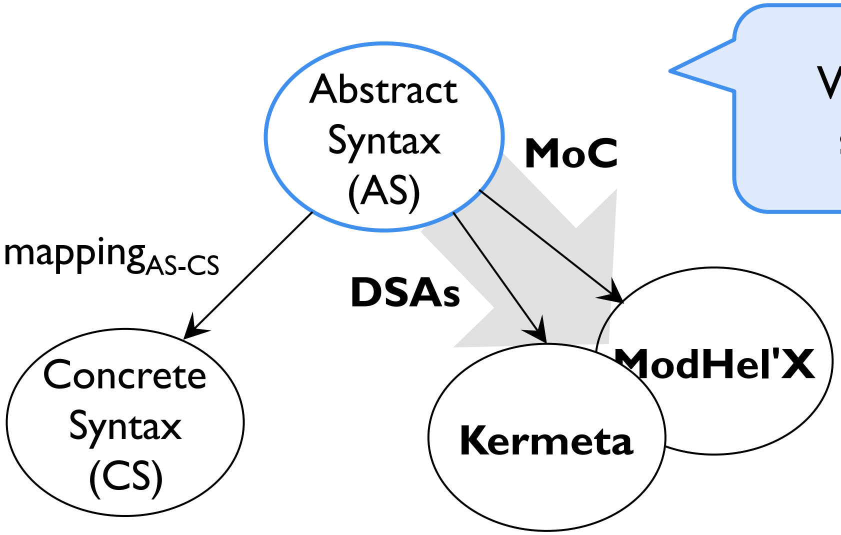
- Foundational UML (fUML) =
  semantics for an executable subset of UML

- fUML = DSL composed of:
  - A subset of the abstract syntax of UML, focused on Activity Diagrams
  - An execution model based on a system of tokens and offers

- Example fUML model:



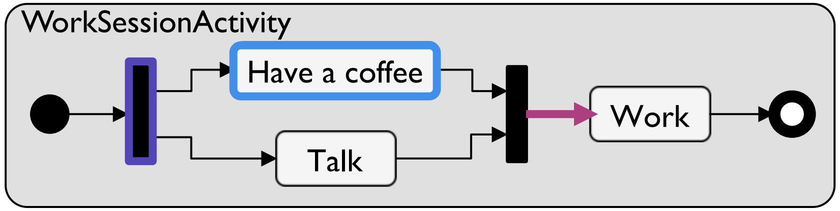
Steps of the "bridging" approach

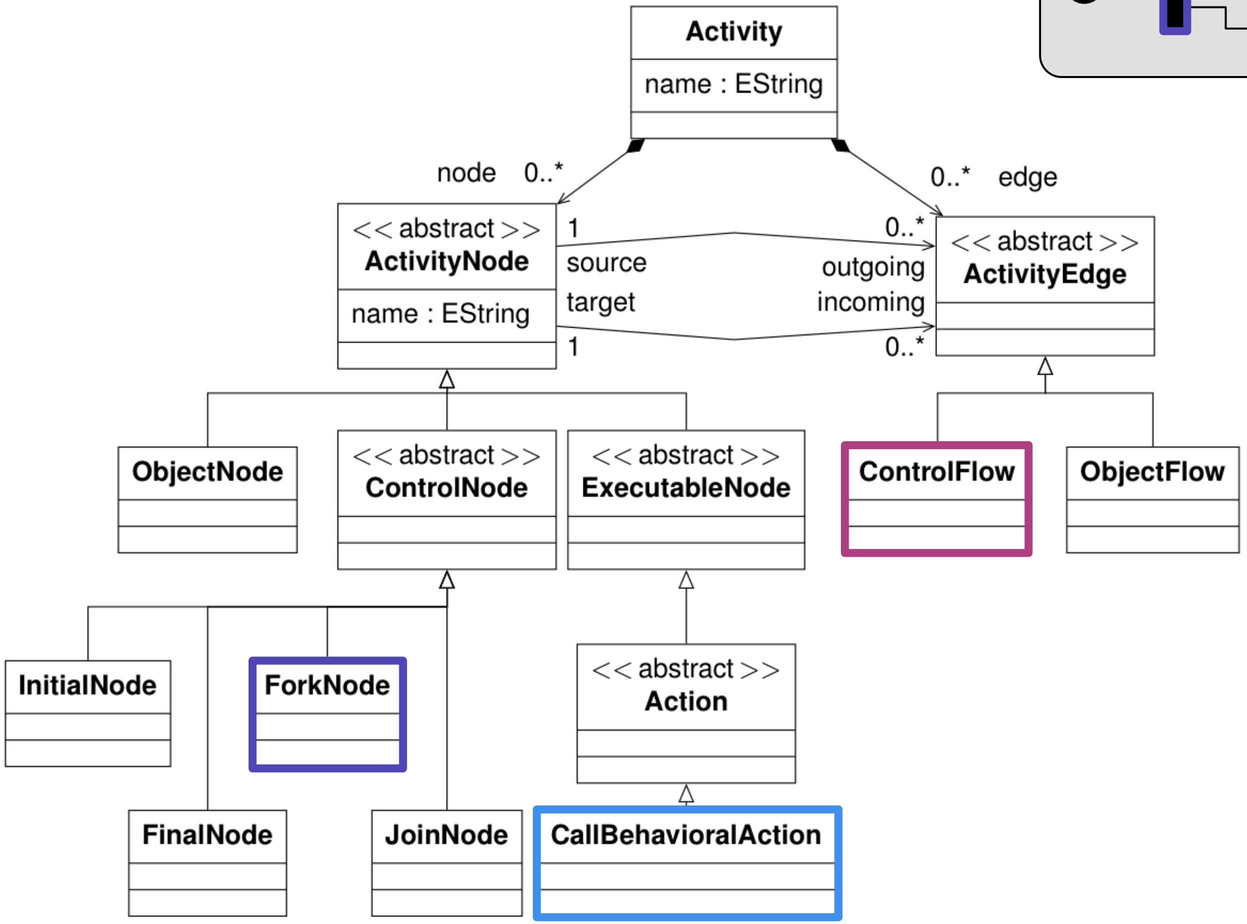


What is the **abstract syntax** of the DSL?

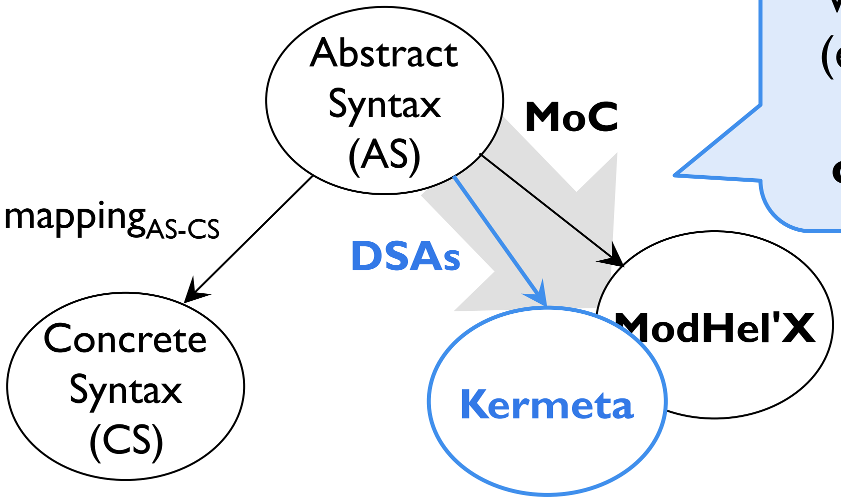
1. Define the metamodel of the DSL with Ecore (+ add static semantics with OCL)

The simplified-fUML metamodel





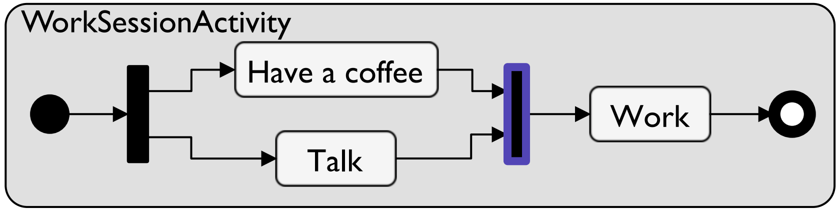
Steps of the "bridging" approach

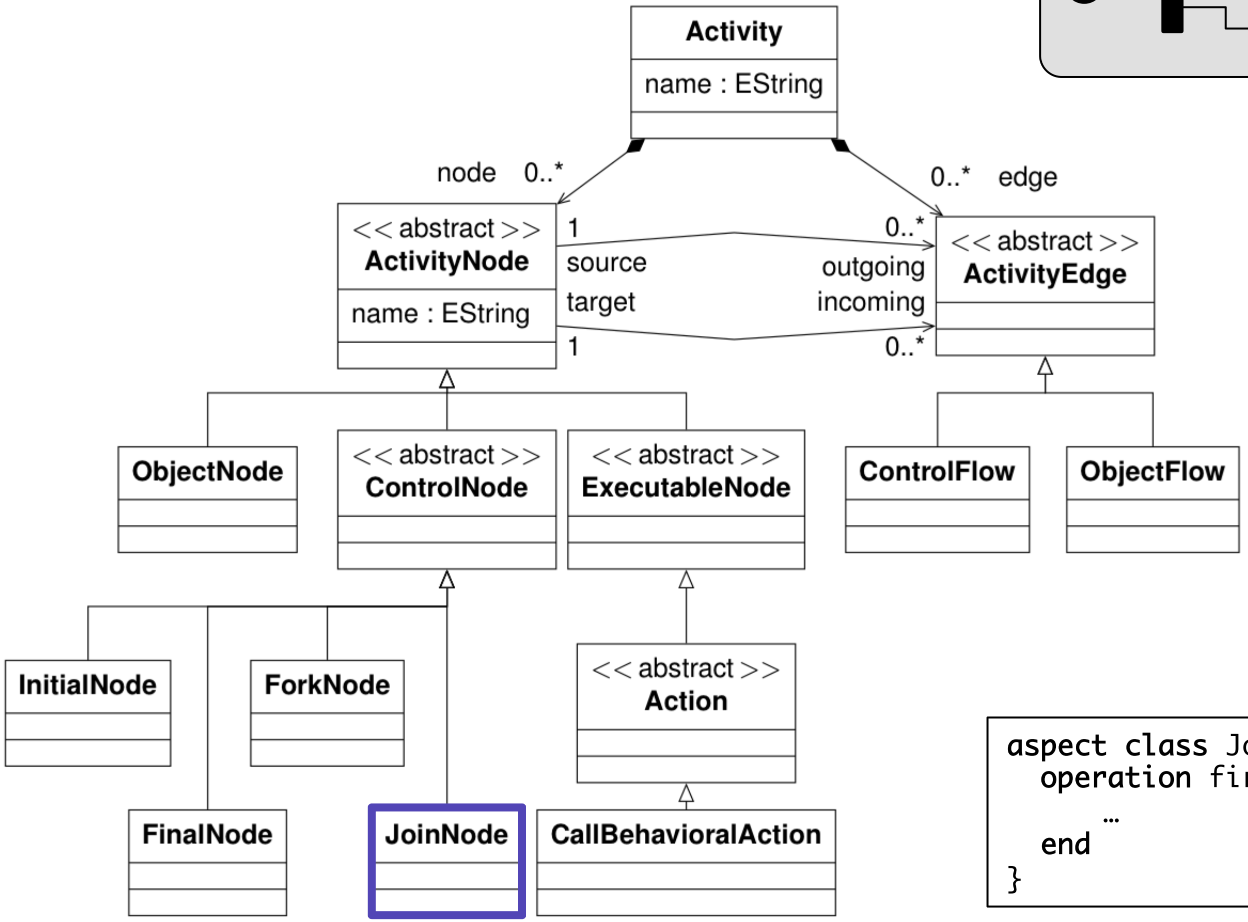


What is the **semantics** (executable behavior) **of the elementary concepts** of the DSL ?

1. Define the metamodel of the DSL with Ecore
   (+ add static semantics with OCL)
2. Weave executable semantics on basic concepts
   = define Domain Specific Actions (DSAs) with Kermeta

Domain Specific Actions (DSAs)

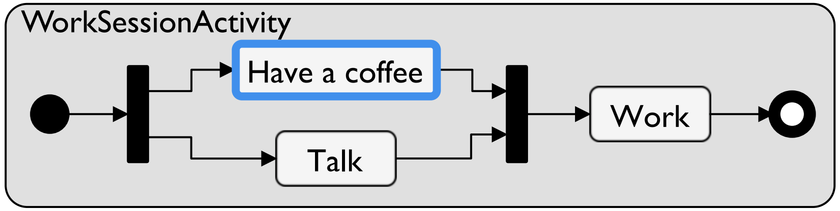


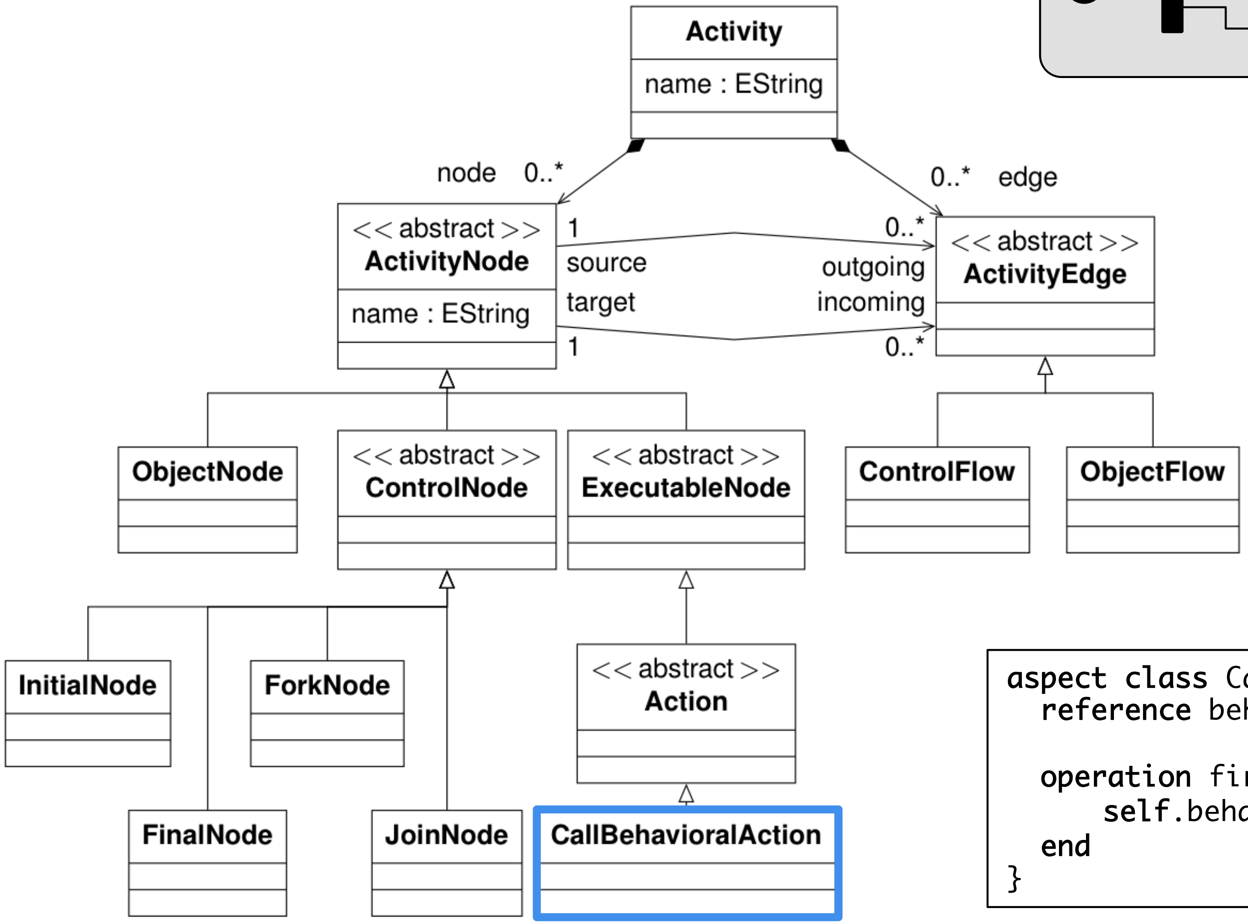


"A JoinNode waits for a token to be offered on all incoming flows and then offers tokens on its outgoing flow."

```
aspect class JoinNode {
  operation fire() is do
                    …
  end
}
```

Domain Specific Actions (DSAs)



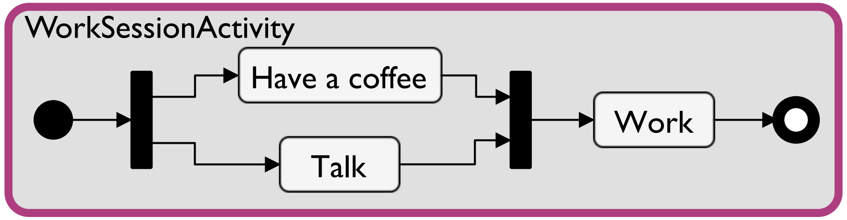


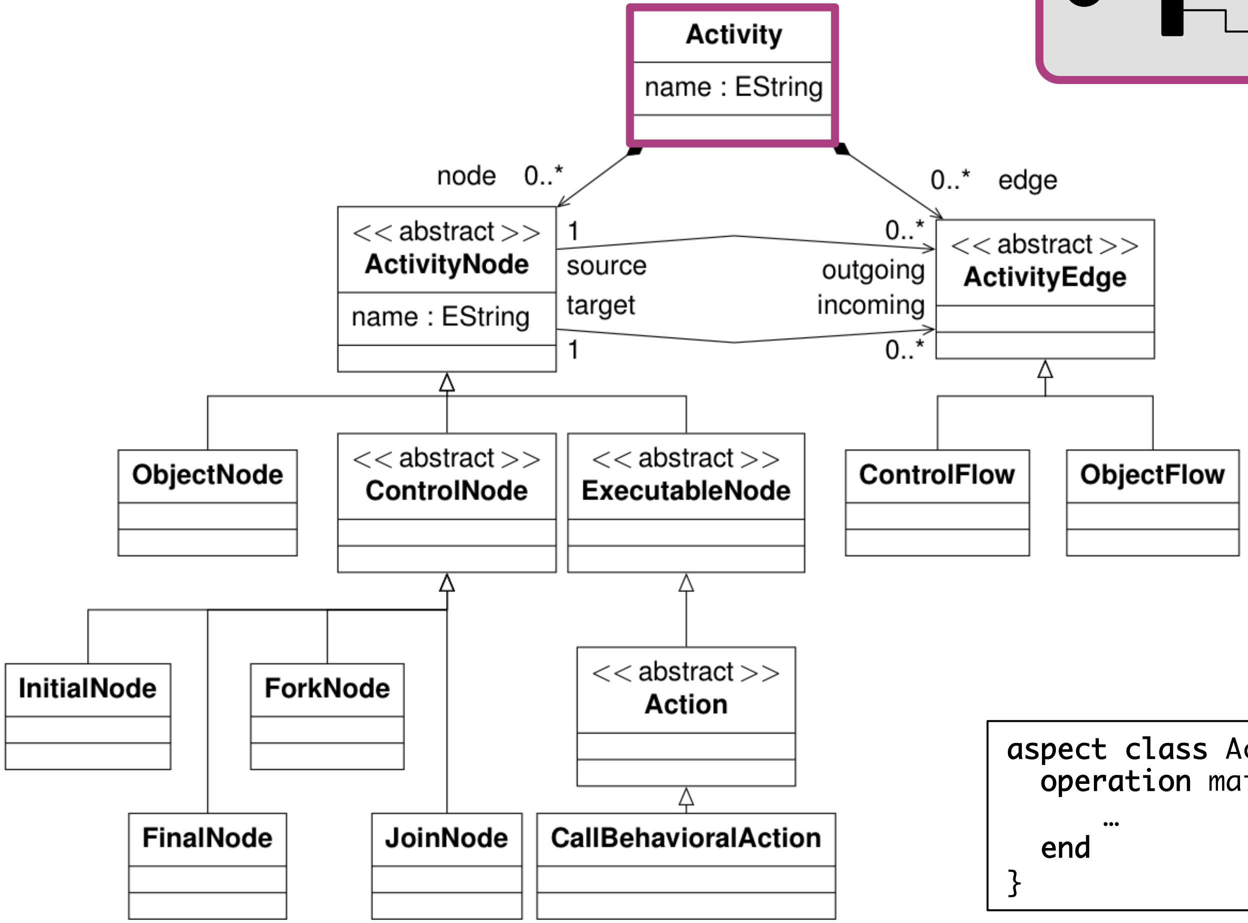
"A CallBehavioralAction calls its associated behavior."

```
aspect class CallBehavioralAction {
  reference behavior : BehavioralAction

  operation fire() is do
      self.behavior.call()
  end
}
```

Who orchestrates the execution?



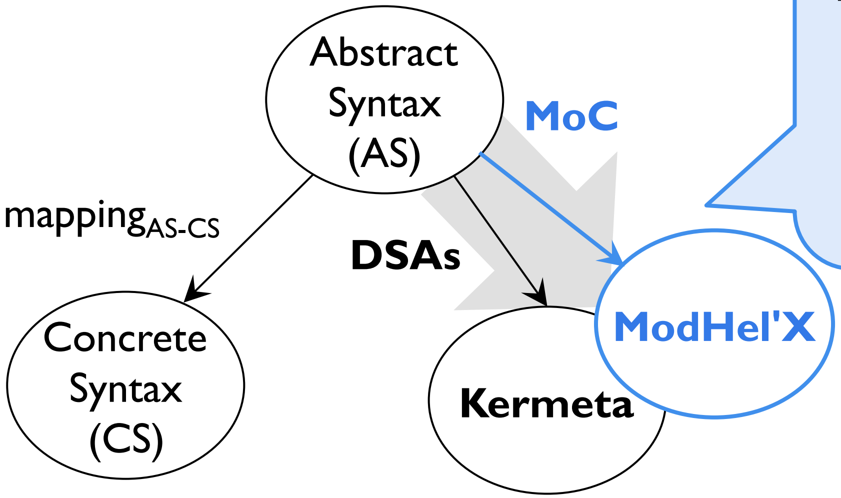


"Executing an Activity means **coordinating the execution of its subordinate actions using the control and data flow**."

```
aspect class Activity {
  operation main() is do
    …
  end
}
```

"Call fire on the elements of the model in the right order, transfer the tokens…"

Steps of the "bridging" approach

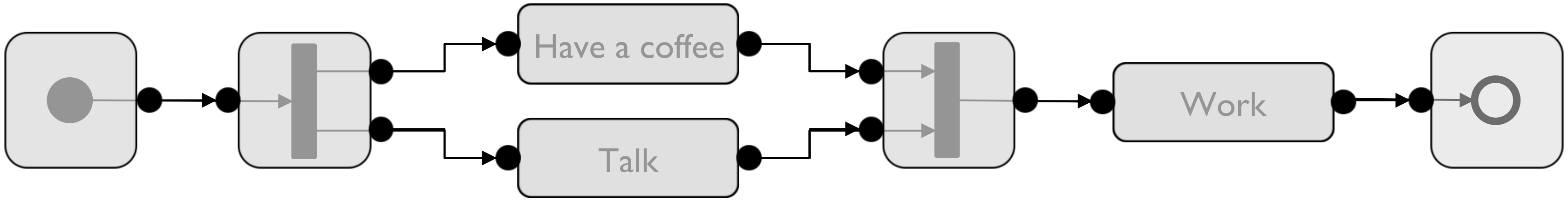


How to **schedule** the calls to **the executable behavior of the elementary concepts** of the DSL ?

1. Define the metamodel of the DSL with Ecore (+ add static semantics with OCL)
2. Weave executable semantics on basic concepts = define Domain Specific Actions (DSAs) with Kermeta
3. Choose a Model of Computation (MoC) with ModHel'X

Notion of Model of Computation

- A (graphical) model can often be abstracted as a block-diagram



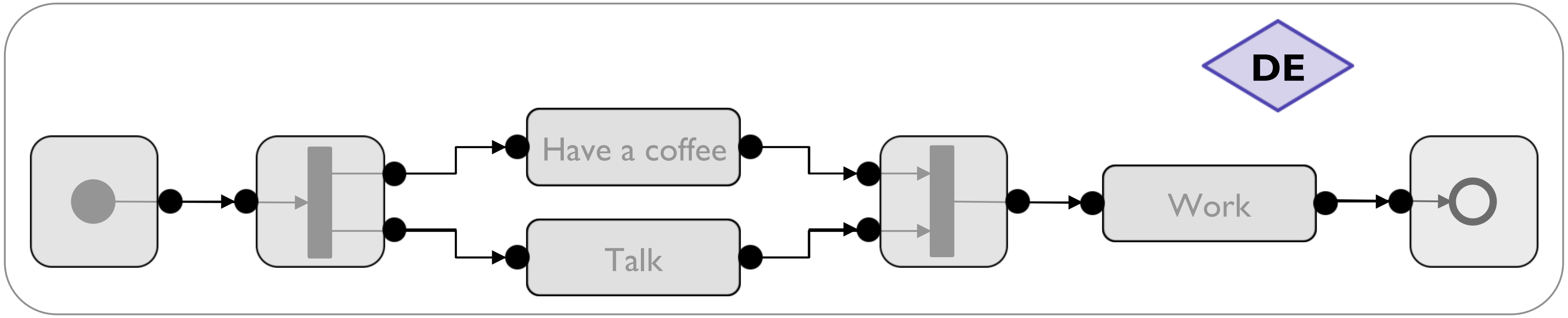
- Executing a block-diagram = executing its blocks…

…But in which order? It depends on:

  - The communication model (*how do these blocks communicate?*)
  - The concurrency model (*do these blocks execute in parallel?*)
  - The time model (*is there a notion of date or duration somewhere in this model?*)

➡ Rules given by the
Model of Computation (MoC)

A MoC for fUML



- Communication, concurrency, time?
  - ActivityNodes exchange tokens (control and objects)
  - ExecutableNodes may run concurrently
  - The execution of ExecutableNodes may take time

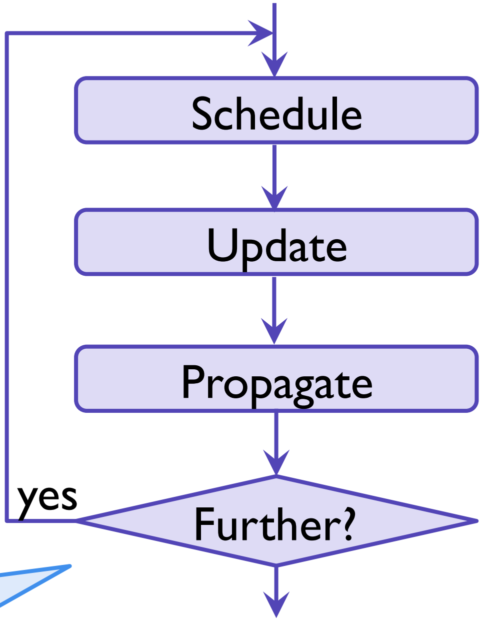
➡ Discrete Events (DE)

Description of a MoC

- MoC = abstract semantics + MoC specific semantics

API for a **generic execution engine** and for **heterogeneous model composition**

The DE specific semantics:



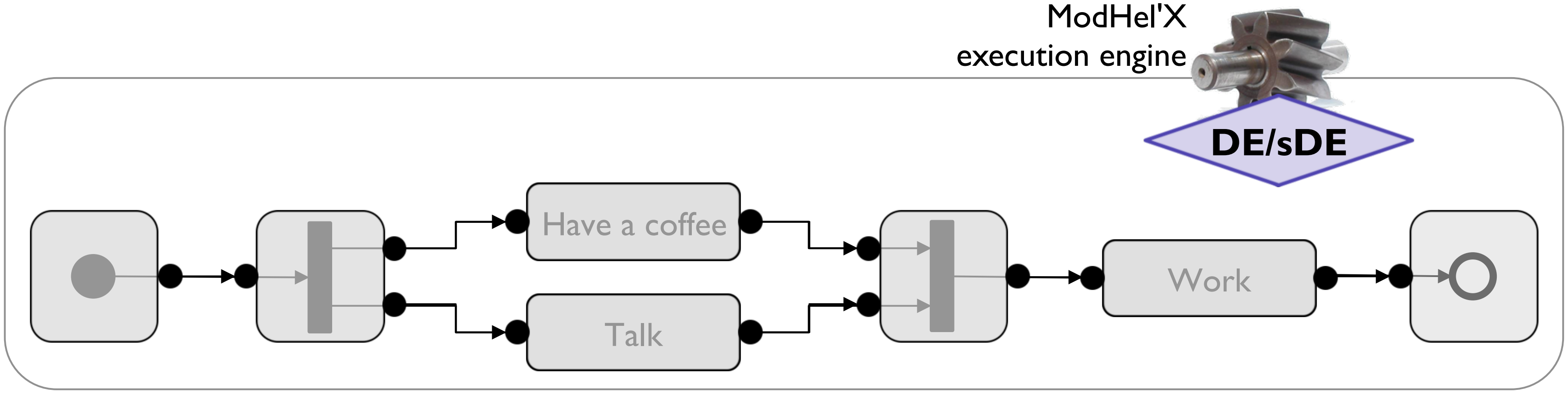
Schedule a block to fire according to the topological order in the graph of blocks and to a list of events to dispatch

Fire the scheduled block

Propagate events along edges

While there are events to process

Semantic variation points of fUML



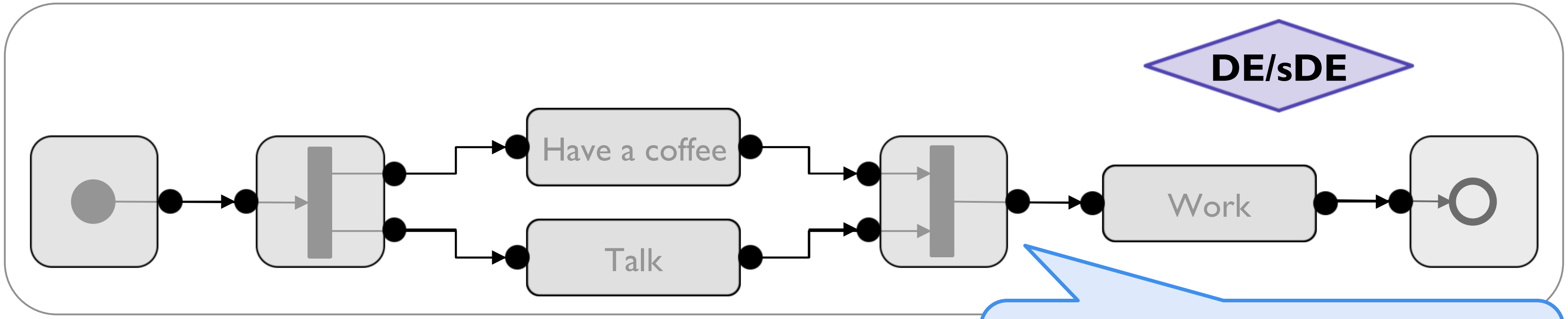
- Communication, concurrency, time?
  - ActivityNodes exchange tokens (control and objects)
  - ExecutableNodes may run concurrently
  - The execution of ExecutableNodes may take time

Unspecified in the fUML spec.

➡ Discrete Events (DE)

➡ Sequential Discrete Events (sDE)

Behavior of blocks?



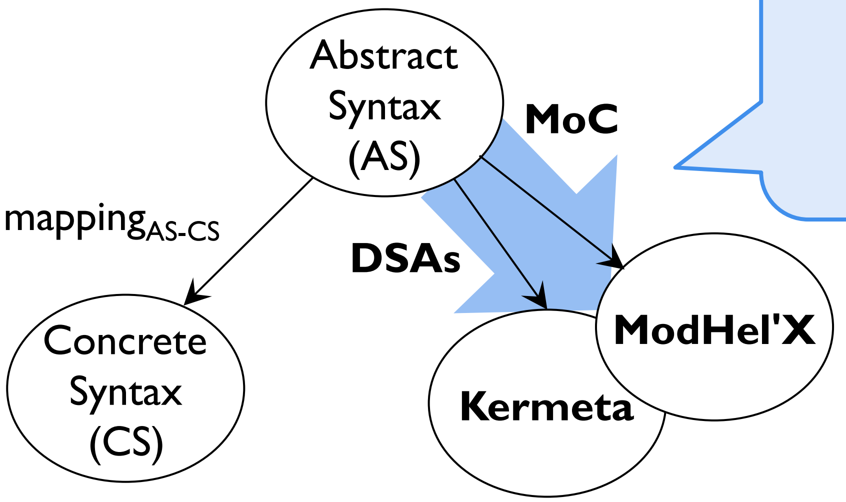
Domain Specific Actions (DSAs)

- Communication, concurrency, time?
  - ActivityNodes exchange tokens (control and objects)
  - ExecutableNodes may run concurrently
  - The execution of ExecutableNodes may take time

➡ Discrete Events (DE)

➡ Sequential Discrete Events (sDE)

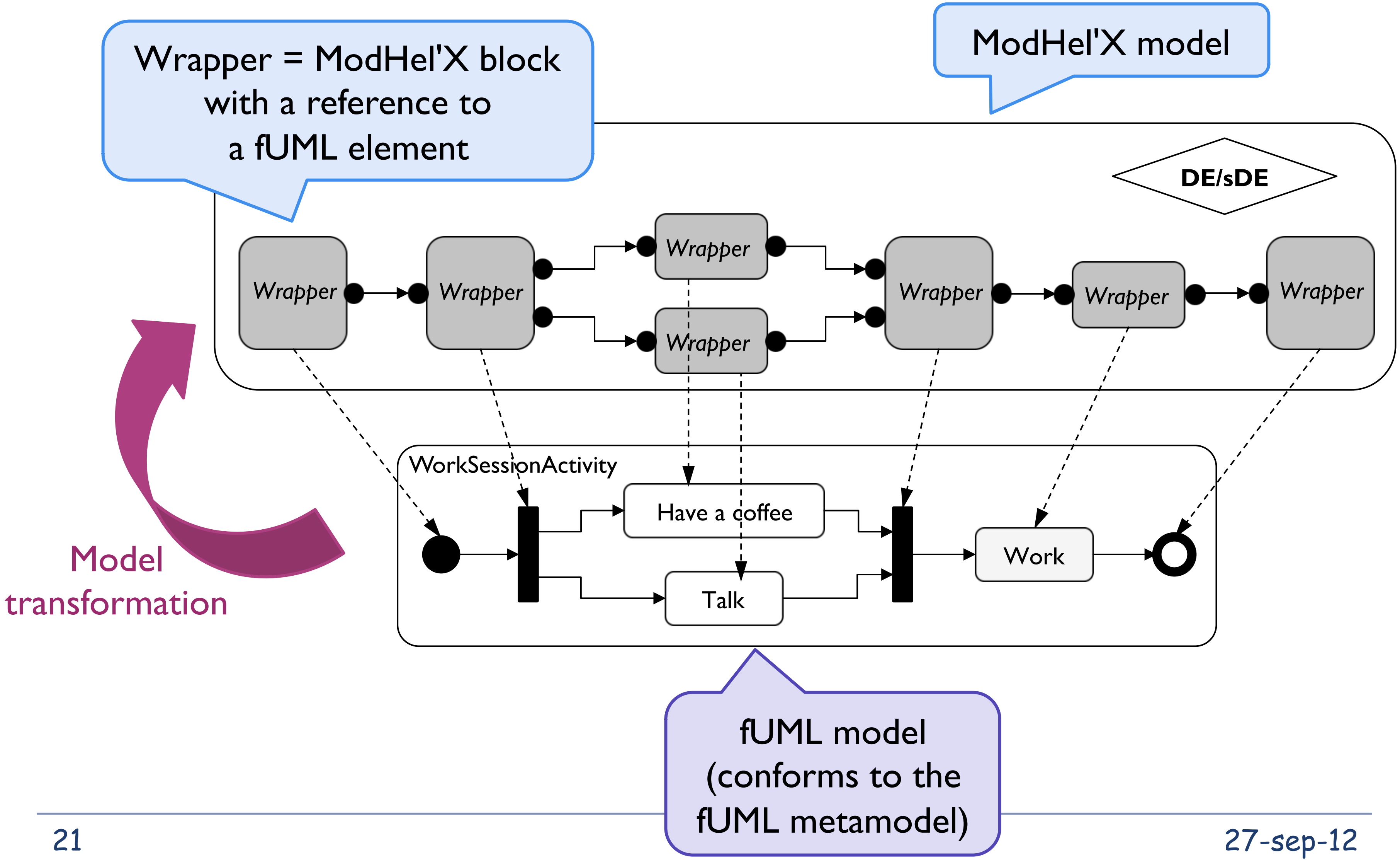
Steps of the "bridging" approach



How does **the MoC interact with the Domain Specific Actions?**

1. Define the metamodel of the DSL with Ecore
   (+ add static semantics with OCL)
2. Weave executable semantics on basic concepts
   = define Domain Specific Actions (DSAs) with Kermeta
3. Choose a Model of Computation (MoC) with ModHel'X
4. Bridge MoC and DSAs

The bridge: structure



Wrapper = ModHel'X block with a reference to a fUML element

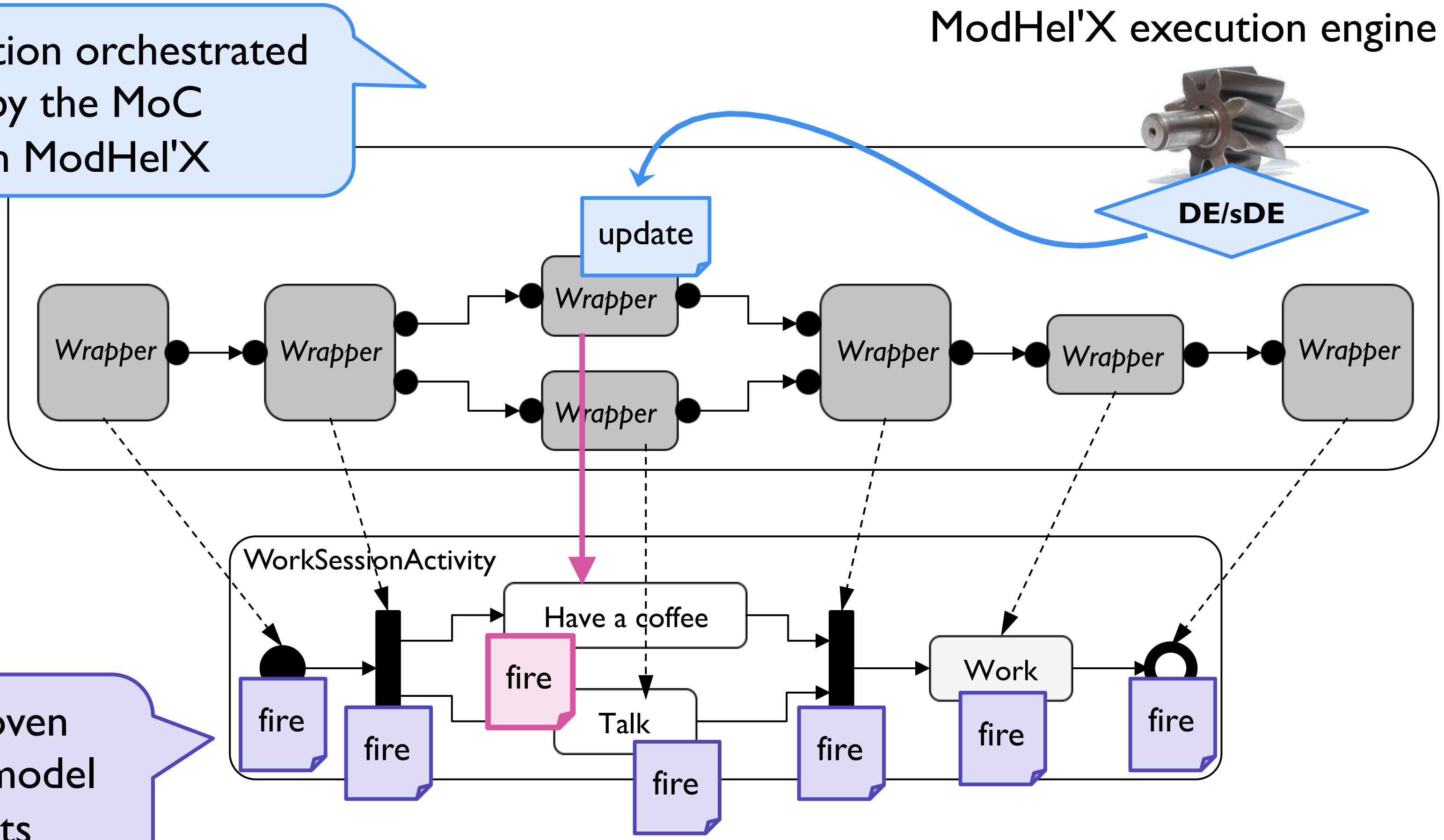
ModHel'X model

Model transformation

fUML model (conforms to the fUML metamodel)

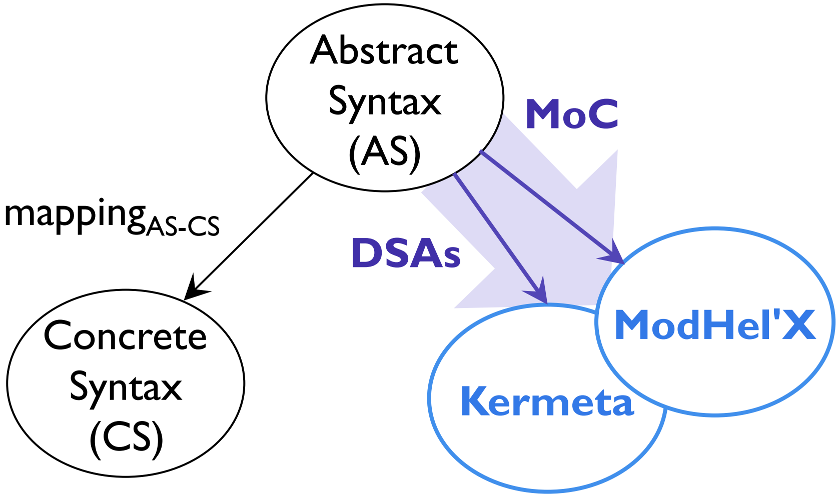
The bridge at runtime

Execution orchestrated by the MoC in ModHel'X



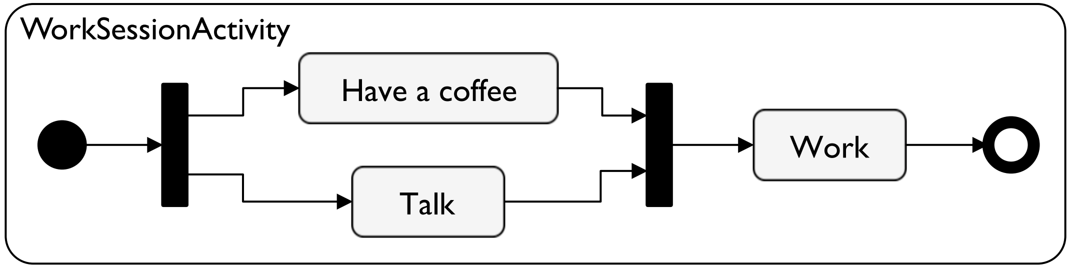
DSAs, woven on (meta)model elements with Kermeta

Steps of the "bridging" approach

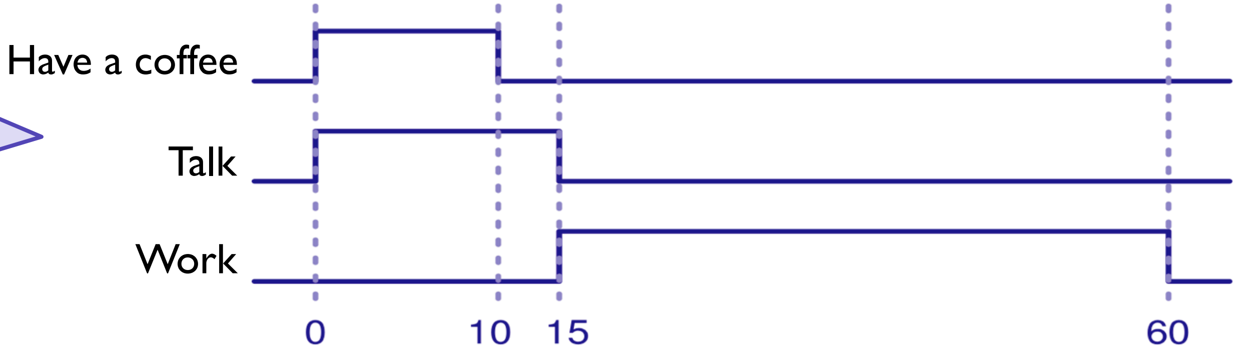


1. Define the metamodel of the DSL with Ecore
   (+ add static semantics with OCL)
2. Weave executable semantics on basic concepts
   = define Domain Specific Actions (DSAs) with Kermeta
3. Choose a Model of Computation (MoC) with ModHel'X
4. Bridge MoC and DSAs

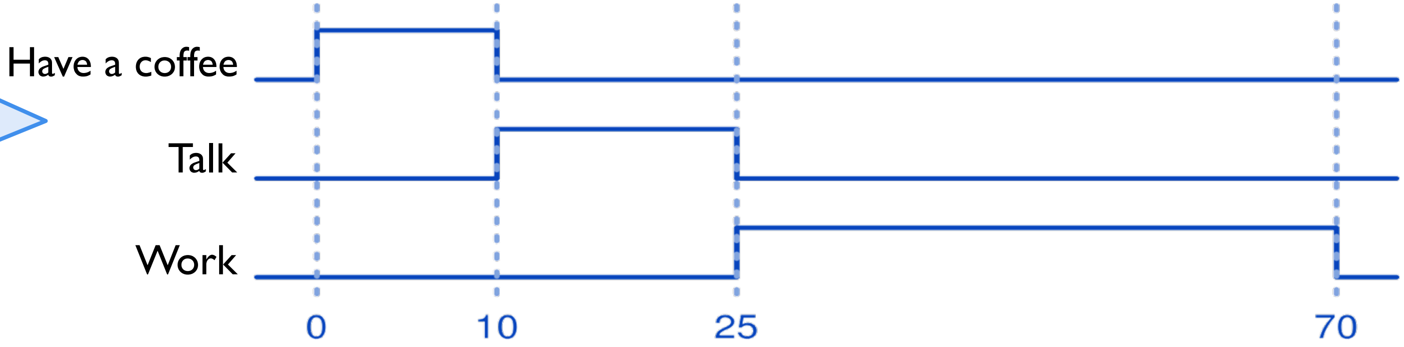
Demo: running the WorkSessionActivity



Discrete Events (DE)



Sequential Discrete Events (sDE)

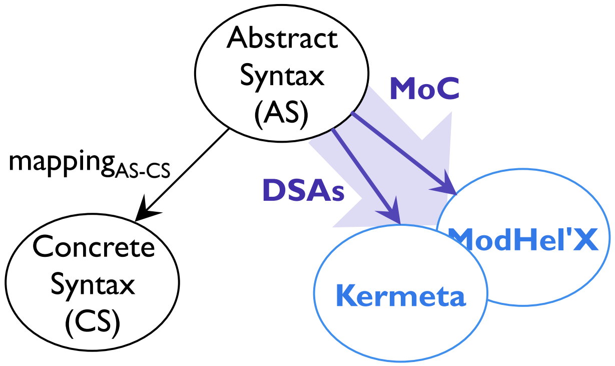


Discussion and future work

- Is this approach independent from Kermeta and ModHel'X?
  - Kermeta and ModHel'X = tools used for the proof-of-concept implementation, other tools could have been used (e.g. Ptolemy II)
- Are the MoC and the DSAs really independent from each other?
  - Well defined interface between MoC and DSAs ☛ ability to reuse the MoC and to obtain semantic variations of a DSL more easily
  - Further experiment is needed on different case studies to define best practices and bridging patterns for MoCs and DSAs
- What are the major perspectives of this work?
  - Take advantage of the heterogeneous composition capabilities of ModHel'X in order to build heterogeneous models using several DSLs

Conclusion

- The “bridging” approach = decomposition of the mapping between abstract syntax and semantic domain in two parts
  - Domain-Specific Actions (DSAs): semantics of domain specific concepts
  - Model of Computation (MoC): communication, concurrency and time semantics (≈ scheduling of DSAs)



- Benefit = ➡ reuse of the MoC in different DSLs
  ➡ variations of a given DSL by varying the MoC

- A proof-of-concept implementation has been made
  - State-of-the-art tools: Kermeta + ModHel'X
  - DSL case study: fUML

Supélec
E3S
Supelec
Systems
Science

UMR IRISA

Inria
informatics mathematics

Thank you!

Bibliography

- *[Karna et al.]* Karna, J., Tolvanen, J.P., Kelly, S.: Evaluating the use of Domain-Specific Modeling in Practice. In: 9th OOPSLA workshop on Domain-Specific Modeling. (2009)
- *[Hutchinson et al.]* Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: ICSE), ACM (2011) 471–480
- *[Bryant et al.]* Bryant, B.R., Gray, J., Mernik, M., Clarke, P.J., France, R.B., Karsai, G.: Challenges and directions in formalizing the semantics of modeling languages. Comput. Sci. Inf. Syst. 8(2) (2011) 225–253
- *[Harel et al.]* Harel, D., Rumpe, B.: Meaningful Modeling: What's the Semantics of "Semantics"? Computer 37(10) (2004) 64–72
- *[OMG]* Object Management Group, Inc.: Semantics of a Foundational Subset for Executable UML Models (fUML), v1.0. (2011)
- *[Kermeta]* Muller, P.A., Fleurey, F., Jézéquel, J.M.: Weaving Executability into Object-Oriented Meta-Languages. In: MoDELS. Volume 3713 of LNCS., Springer (2005) 264–278
- *[ModHel'X]* Boulanger, F., Hardebolle, C.: Simulation of Multi-Formalism Models with ModHel'X. In: Proceedings of ICST'08, IEEE Comp. Soc. (2008) 318–327
- *[PtolemyII]* Eker, J., Janneck, J.W., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity – the Ptolemy approach. Proc. of the IEEE 91(1) (2003) 127–144