ÉCOLE SUPÉRIEURE D'ÉLECTRICITÉ

Supélec

ISSSE 2007

ModHel'X: an approach to multi-formalism modeling

Cécile Hardebolle, Frédéric Boulanger, Dominique Marcadet, Guy Vidal-Naquet

26 September 2007

SYSTEM@TIC PARIS-REGION Pôle de compétitivité

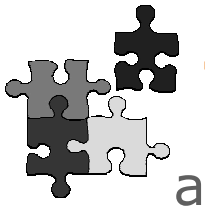
Cécile HARDEBOLLE
Supélec – Departement d'Informatique
✉ cecile.hardebolle@supelec.fr

Introduction

- Context
  - Modeling the behavior of heterogeneous systems
    - Software/hardware, digital/analog, internal/external IPs…
  - Using different models to represent one system
    - Level of refinement, focus aspects (NFP…), Domain Specific Modeling Languages (DSMLs)

Introduction

- Context
  - Modeling the behavior of heterogeneous systems
    - Software/hardware, digital/analog, internal/external IPs…
  - Using different models to represent one system
    - Level of refinement, focus aspects (NFP…), Domain Specific Modeling Languages (DSMLs)

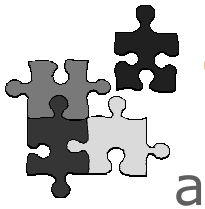


Русский 精忠

Multi-formalism modeling =
allow the use of several modeling languages in a model

Introduction

- Context
  - Modeling the behavior of heterogeneous systems
    - Software/hardware, digital/analog, internal/external IPs…
  - Using different models to represent one system
    - Level of refinement, focus aspects (NFP…), Domain Specific Modeling Languages (DSMLs)



Русский 精忠

Multi-formalism modeling =
allow the use of several modeling languages in a model

- What for ?
  - Simulation, code generation, verification, validation, tests…
  - Maximize model reuse, facilitate and optimize designers collaboration
- Main issues
  - Describe the semantics of a modeling language precisely
  - Define the semantics of a combination of modeling languages in a model

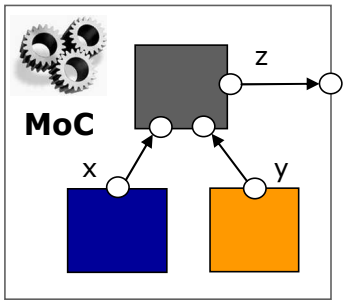
Related work

- Kermeta [Muller05]: meta-programming language allowing to define the semantics of a modeling language using UML meta-models
- ATOM^3 [deLara02]: meta-model transformation tool for multi-paradigm modeling
- Ptolemy II [Lee03]: heterogeneous modeling framework based on the model of computation concept, with a component-oriented abstract syntax
- BIP [Sifakis06]: language for defining heterogeneous interactions, with formal properties for verification
- Tagged Signal Model [Lee98]: mathematical framework for comparing models of computation, no execution purpose
- Rosetta [Kong03]: heterogeneous modeling framework with a denotational semantics, implements the facets concept

The ModHel'X approach

- Vocabulary
  - Modeling language = structure + semantics
    - UML meta-model + execution operations (imperative semantics)

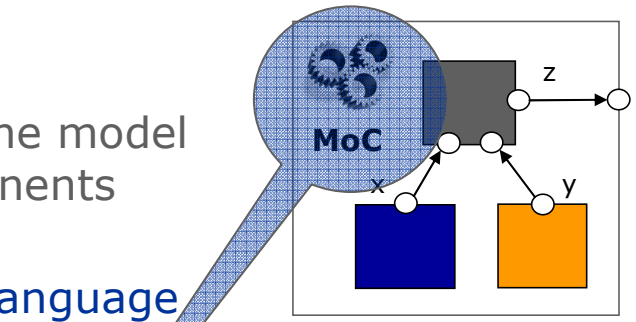
    or
    - Fixed (component-oriented) abstract syntax + Model of Computation
  - Model of computation (MoC)
    - Set of rules that define the behavior of the model by combining the behaviors of its components
    - = description of the "computation and communication features" of a modeling language



The ModHel'X approach

- Vocabulary
  - Modeling language = structure + semantics
    - UML meta-model + execution operations (imperative semantics)

    or
    - Fixed (component-oriented) abstract syntax + Model of Computation
  - Model of computation (MoC)
    - Set of rules that define the behavior of the model by combining the behaviors of its components
    - = description of the "computation and communication features" of a modeling language

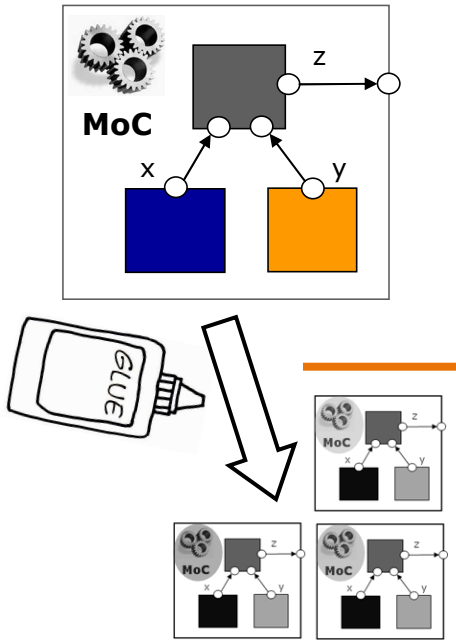


- Objectives
  - Allow a precise description of
    - the semantics of models of computation

The ModHel'X approach

- Vocabulary
  - Modeling language = structure + semantics
    - UML meta-model + execution operations (imperative semantics)

    or
    - Fixed (component-oriented) abstract syntax + Model of Computation
  - Model of computation (MoC)
    - Set of rules that define the behavior of the model by combining the behaviors of its components
    - = description of the "computation and communication features" of a modeling language

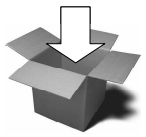


- Objectives
  - Allow a precise description of
    - the semantics of models of computation
    - the semantics of their interactions (the "glue" !)
  - Provide support for the interpretation of compositions of MoCs in order to allow the execution of heterogeneous models

ÉCOLE SUPÉRIEURE D'ÉLECTRICITÉ

Main principles

- Encapsulation



  - Components of a model = black-boxes with well defined interfaces
    - Goal: decouple the internal mechanism of a component from the model in which it is used

Main principles

- Encapsulation

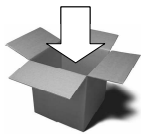
  - Components of a model = black-boxes with well defined interfaces
    - Goal: decouple the internal mechanism of a component from the model in which it is used

- Observation

  - Execution of a model = observations of its behavior = snapshots
    - Triggered by time, environment changes and by components of the model
  - Snapshot = combination of observations of the components of the model according to the MoC
  - Observation of a component (black-box) = update of its interface

Main principles

Encapsulation



- Components of a model = black-boxes with well defined interfaces
  - Goal: decouple the internal mechanism of a component from the model in which it is used

Observation

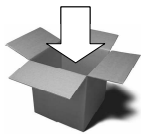
- Execution of a model = observations of its behavior = snapshots
  - Triggered by time, environment changes and by components of the model
- Snapshot = combination of observations of the components of the model according to the MoC
- Observation of a component (black-box) = update of its interface

Hierarchy & delegation

- Behavior of a component = internal model + internal MoC
- Update of a component = update of its internal model
- Semantic adaptation at the border of the component

Main principles

Encapsulation



- Components of a model = black-boxes with well defined interfaces
  - Goal: decouple the internal mechanism of a component from the model in which it is used

Observation

- Execution of a model = observations of its behavior = snapshots
  - Triggered by time, environment changes and by components of the model
- Snapshot = combination of observations of the components of the model according to the MoC
- Observation of a component (black-box) = update of its interface

Hierarchy & delegation

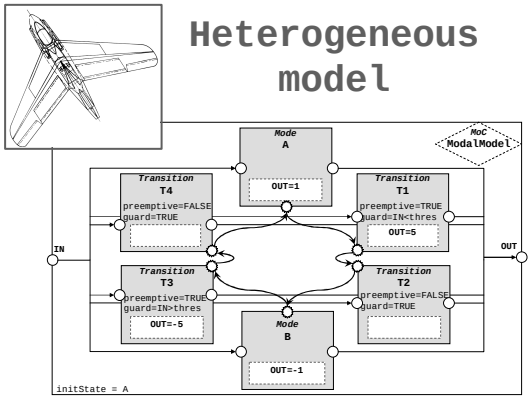
- Behavior of a component = internal model + internal MoC
- Update of a component = update of its internal model
- Semantic adaptation at the border of the component

one MoC by layer + local heterogeneity = reduced complexity

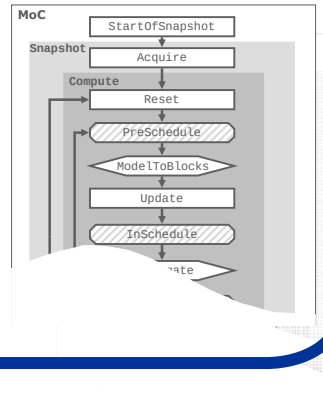
General architecture of ModHel'X

Executable descriptions of each MoC =

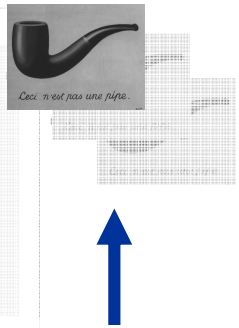
Heterogeneous model



Generic model of execution



Semantics of the MoC



Generic execution engine

Generic meta-model for the representation of the structure of heterogeneous models

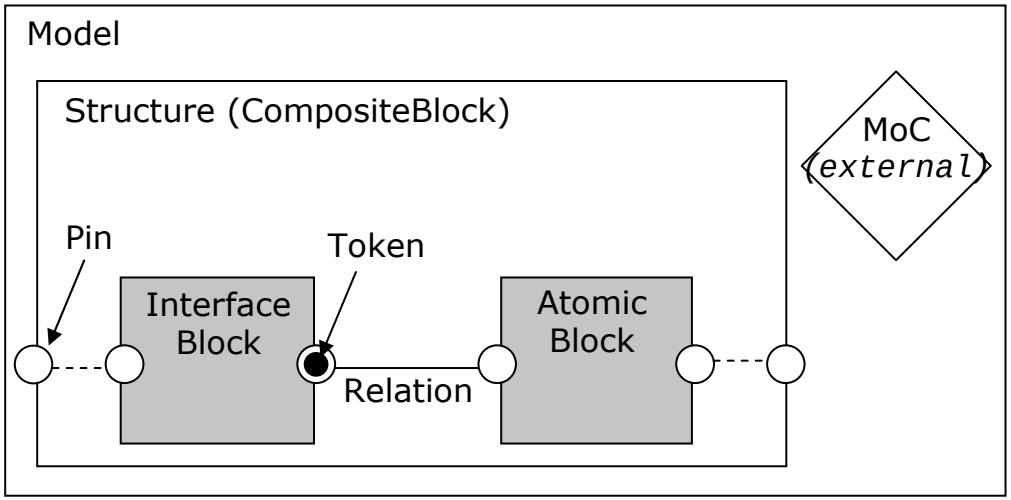
Generic algorithm for executing heterogeneous models

Language for describing

- The operational semantics of a MoC
- The interactions between MoCs

Representing an heterogeneous model

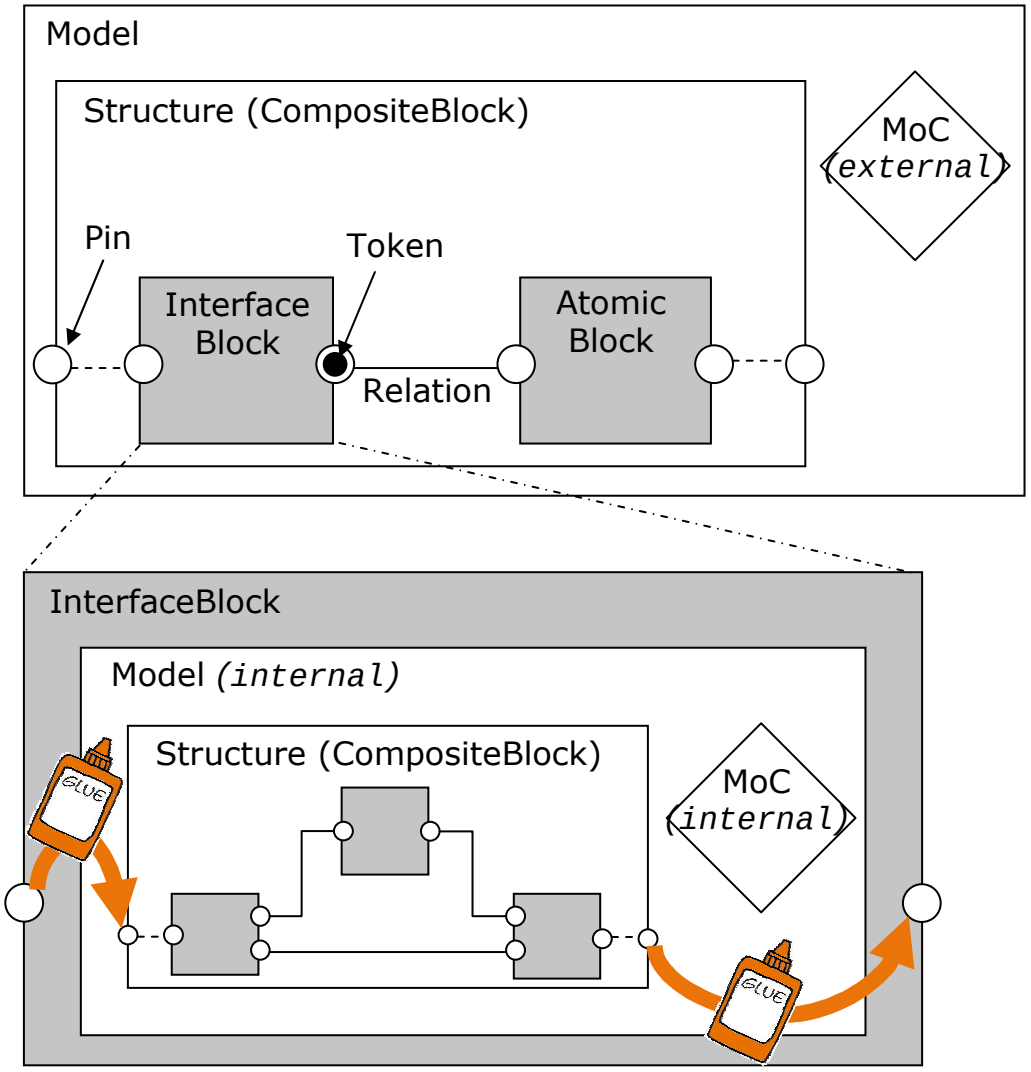
- A set of few basic and generic objects



- Two separate aspects
  - Structural: Blocks, Pins, Relations, Tokens
  - Behavioral: Model of computation
- Specialization of these objects for each MoC

Representing an heterogeneous model

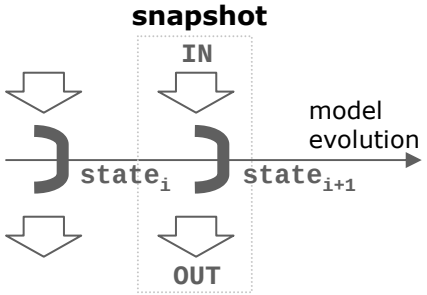
- A set of few basic and generic objects



- Two separate aspects
  - Structural: Blocks, Pins, Relations, Tokens
  - Behavioral: Model of computation
- Specialization of these objects for each MoC
- Hierarchical heterogeneity
  - An InterfaceBlock has an internal model
  - The internal & the external MoCs can be different
  - The InterfaceBlock realizes the semantic adaptation

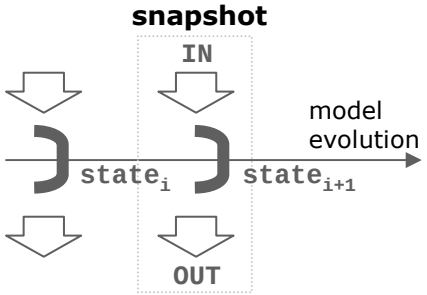
Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model

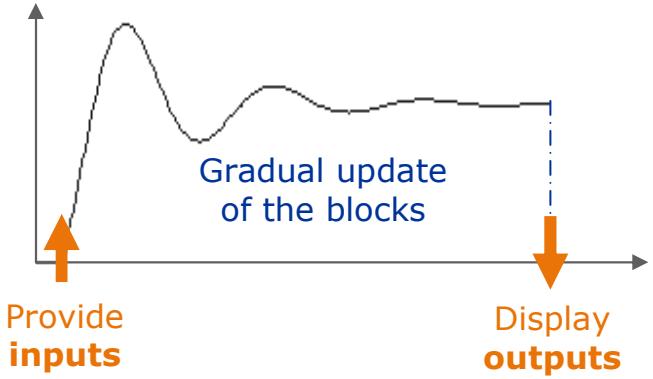


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



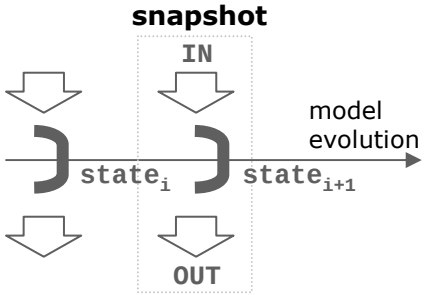
- One snapshot =



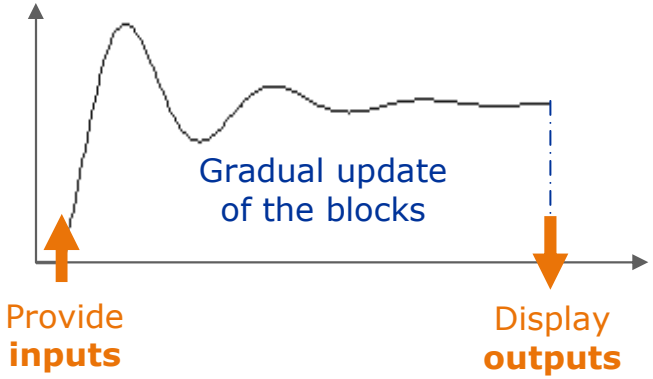
- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model

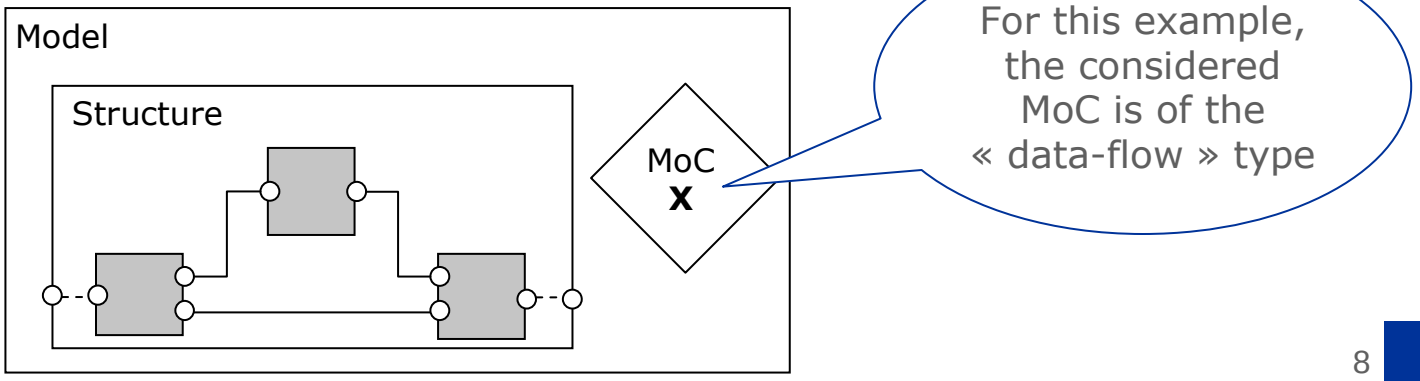


- One snapshot =



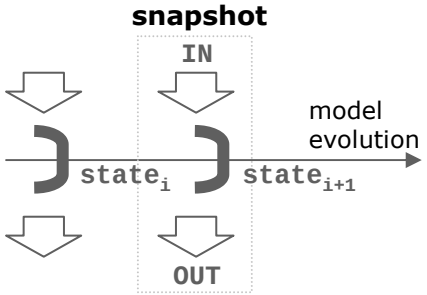
  - Causal execution
    - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
  - Gradual update of the model blocks
    - Schedule of the block to update
    - Update of the block
    - Propagation of the produced data

- Example

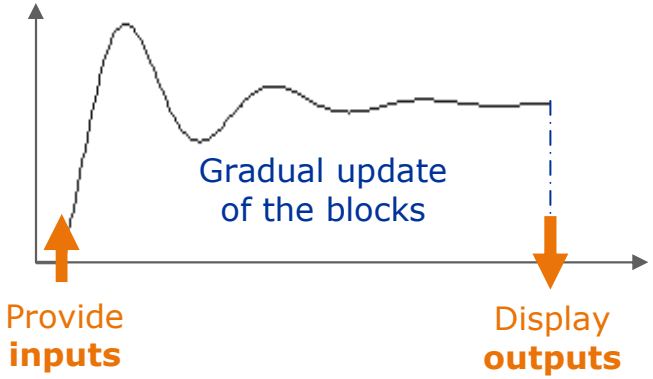


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



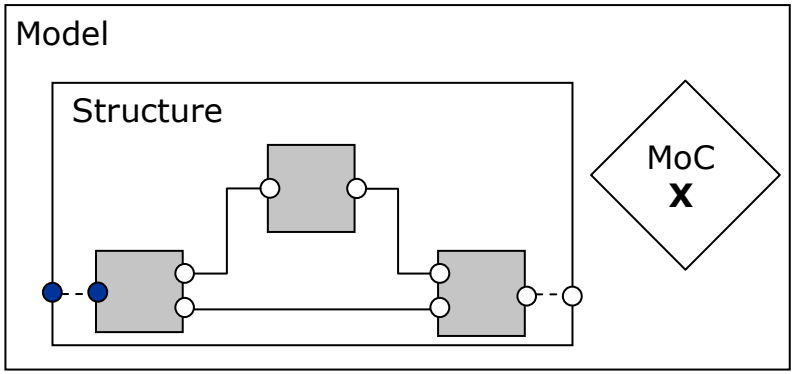
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

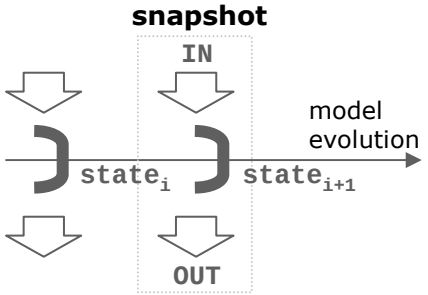
- Example

Provide inputs

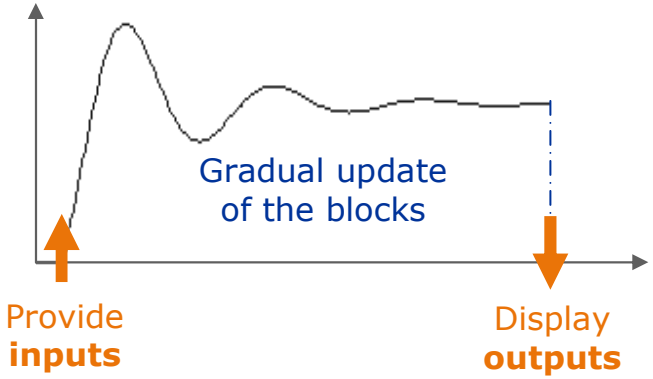


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



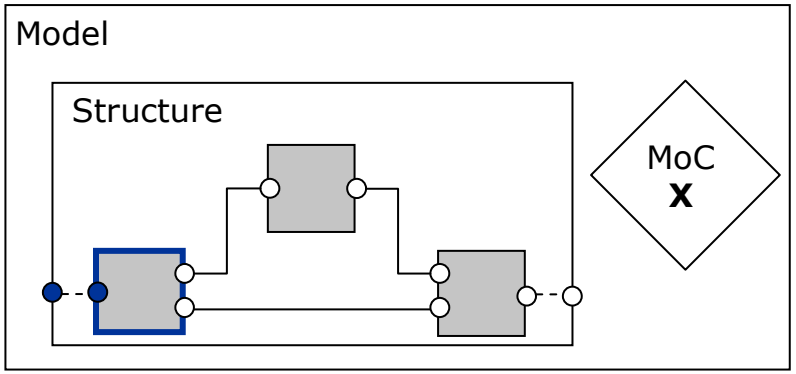
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

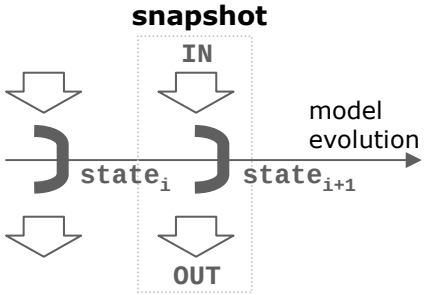
- Example

Schedule block

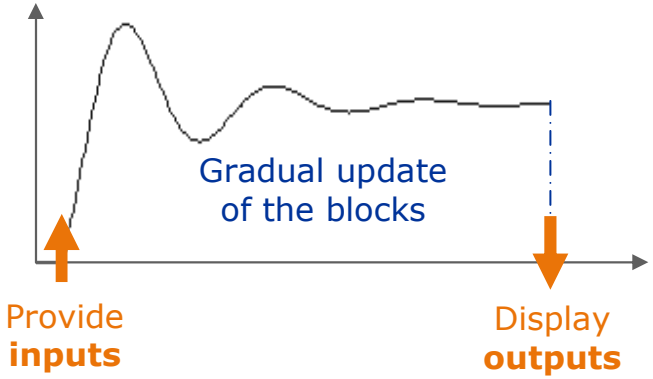


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



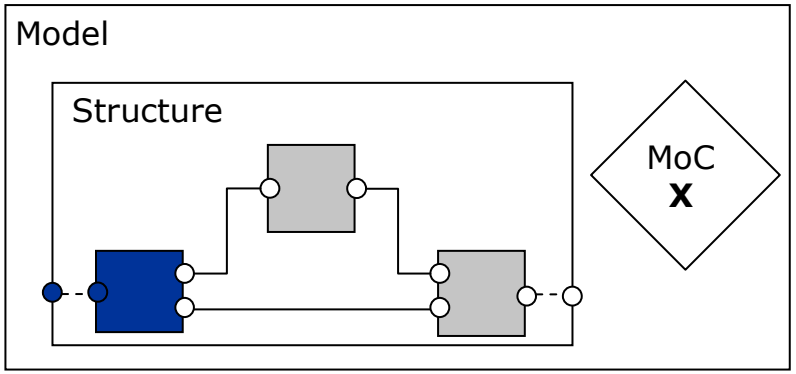
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

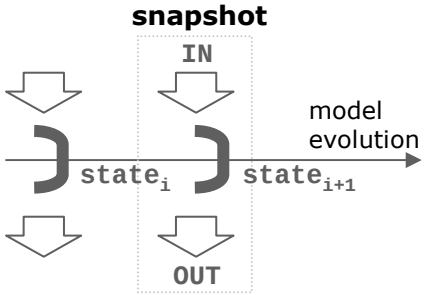
- Example

Update block

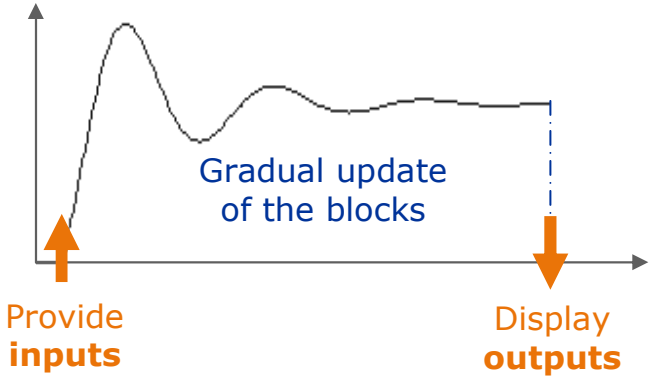


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



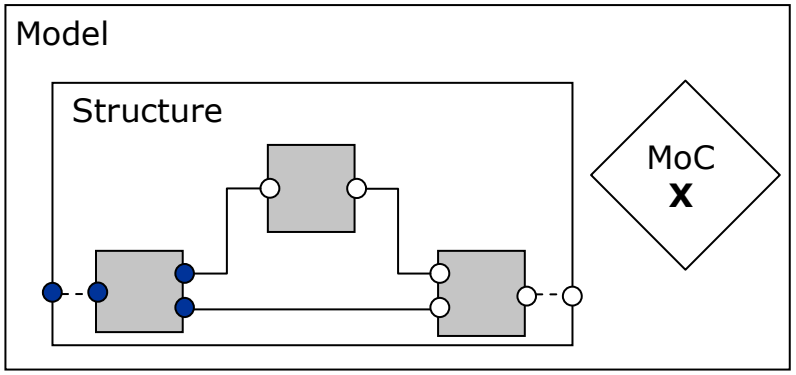
- One snapshot =



  - Causal execution
    - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
  - Gradual update of the model blocks
    - Schedule of the block to update
    - Update of the block
    - Propagation of the produced data

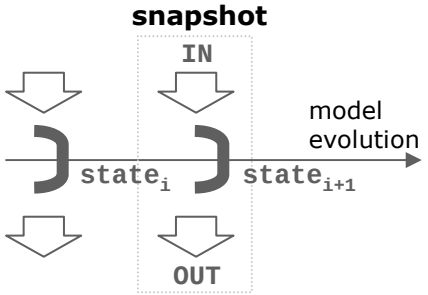
- Example

After update

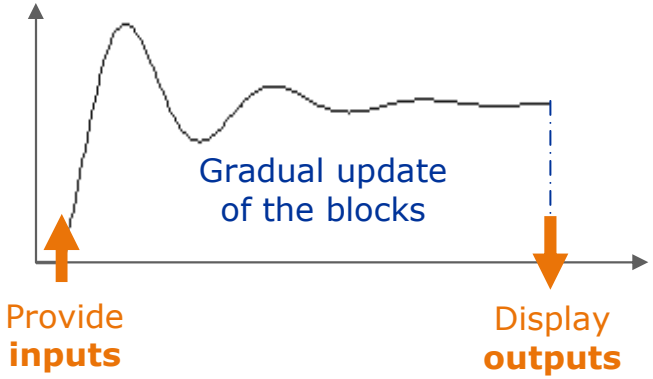


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



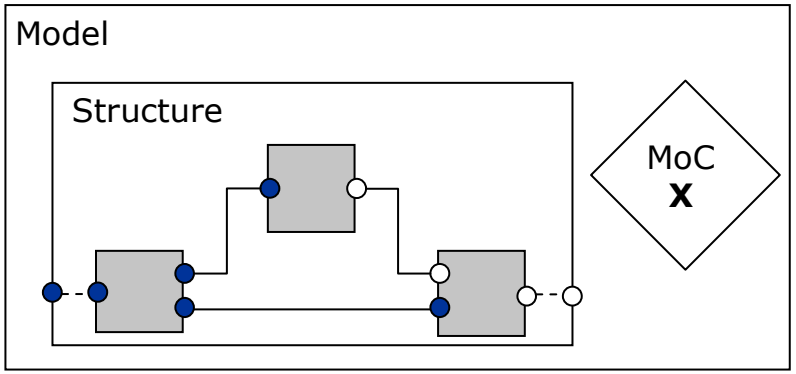
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

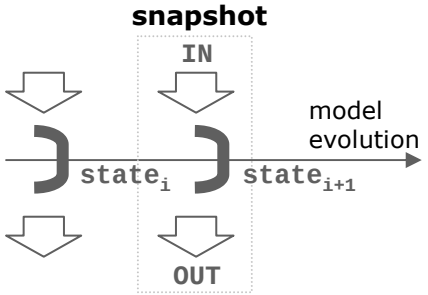
- Example

Propagate data

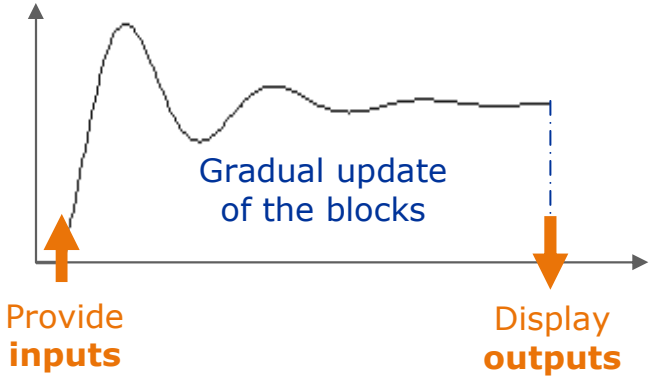


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



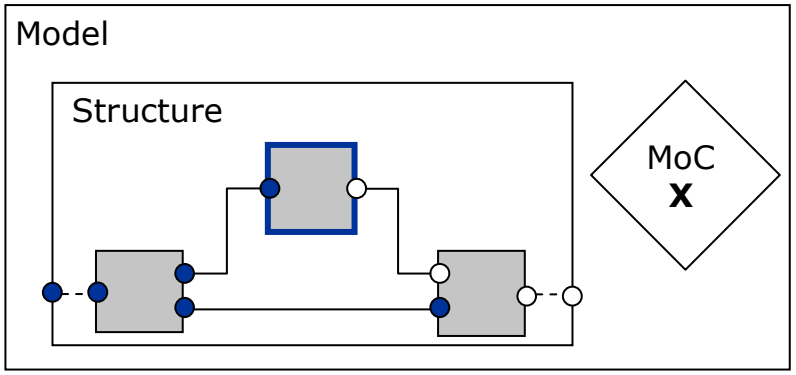
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

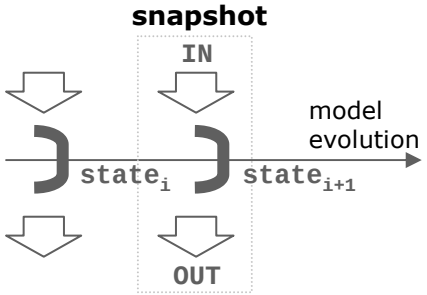
- Example

Schedule block

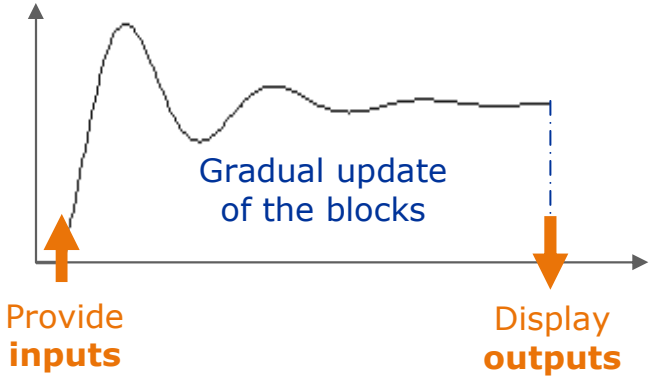


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



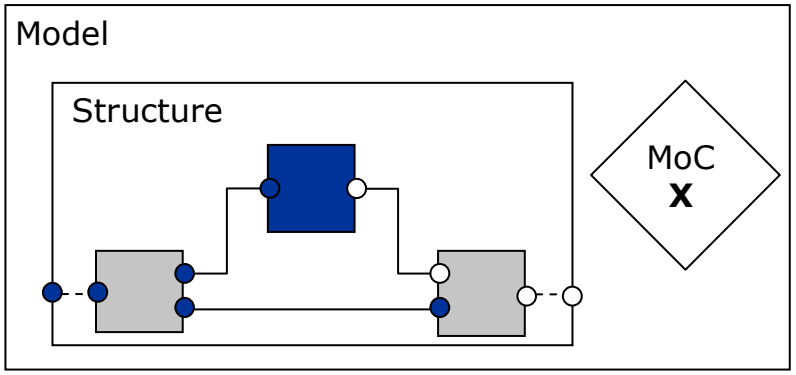
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

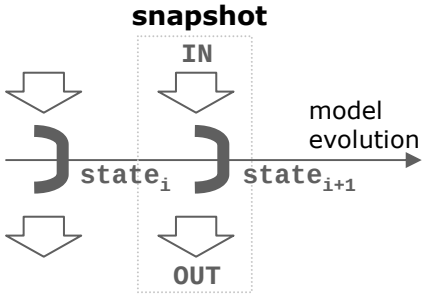
- Example

Update block

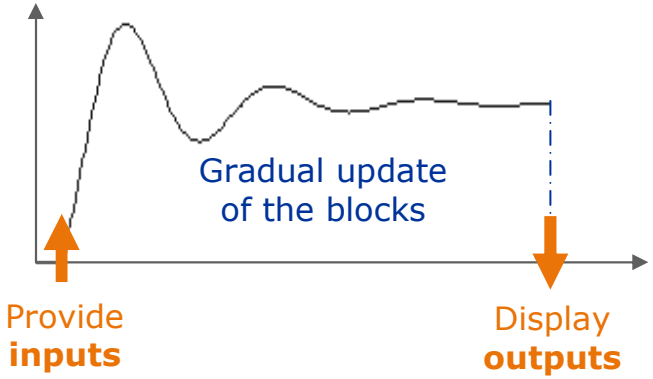


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



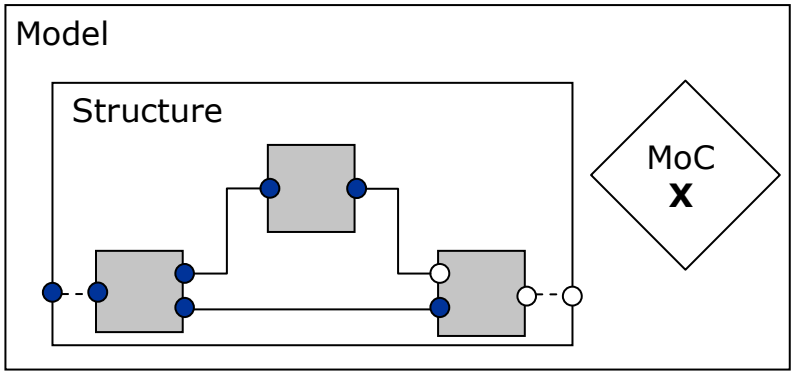
- One snapshot =



  - Causal execution
    - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
  - Gradual update of the model blocks
    - Schedule of the block to update
    - Update of the block
    - Propagation of the produced data

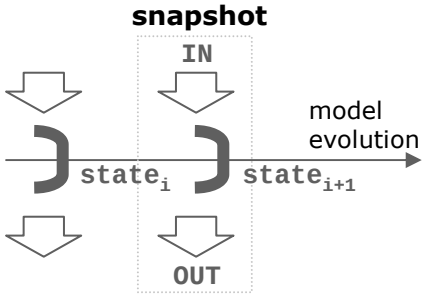
- Example

After update

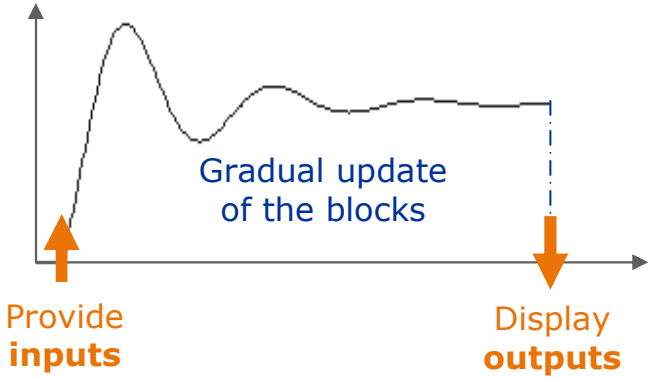


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



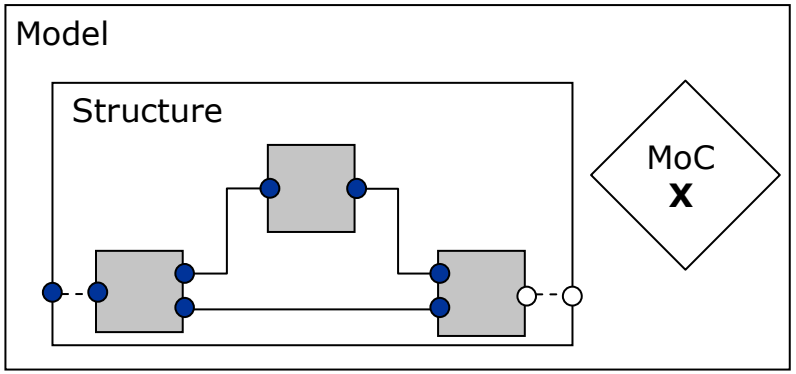
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

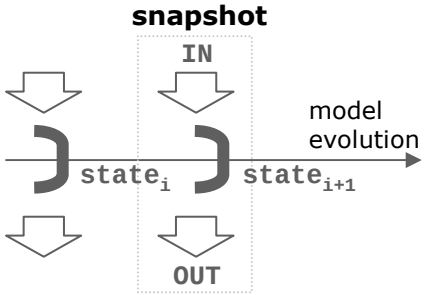
- Example

Propagate data

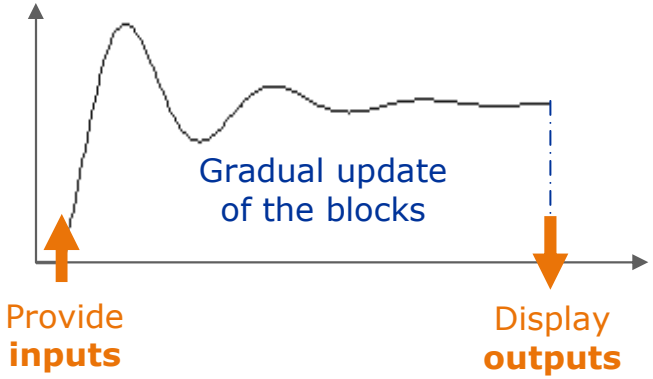


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



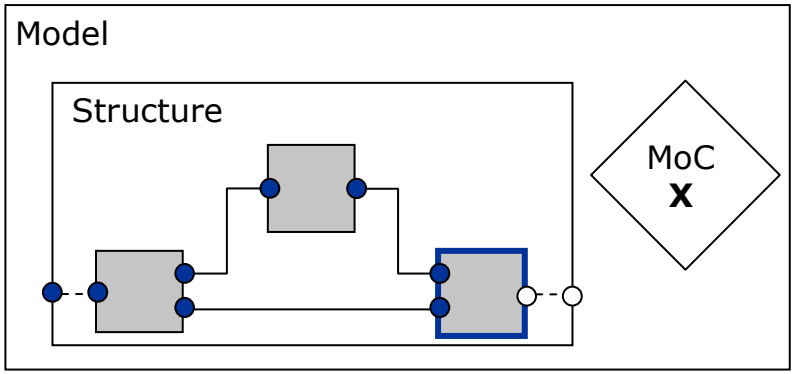
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

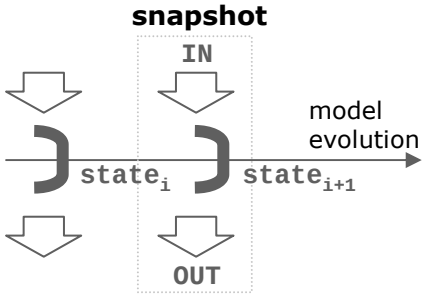
- Example

Schedule block

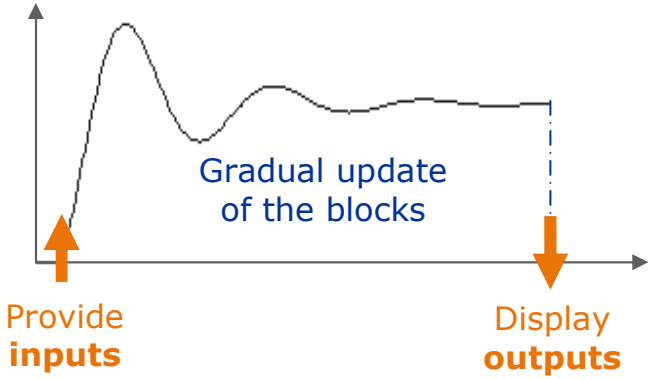


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



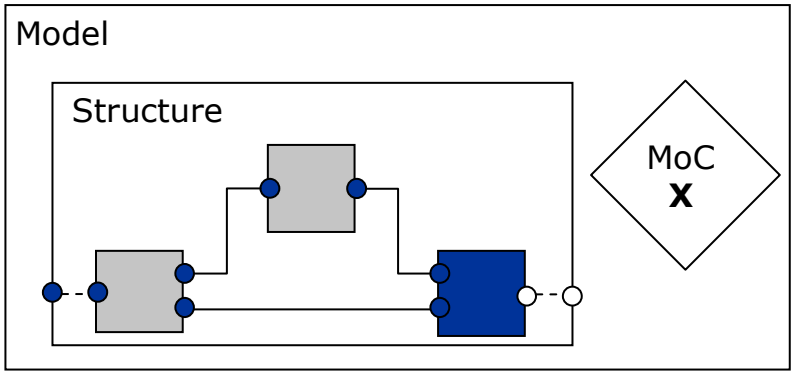
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

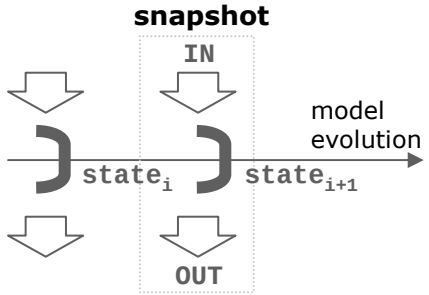
- Example

Update block

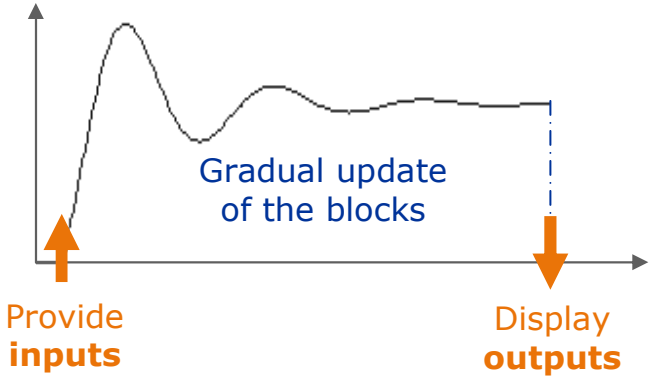


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



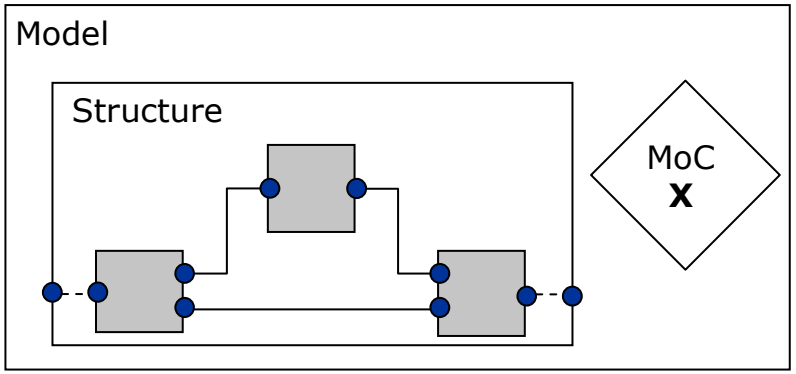
- One snapshot =



- Causal execution
  - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
- Gradual update of the model blocks
  - Schedule of the block to update
  - Update of the block
  - Propagation of the produced data

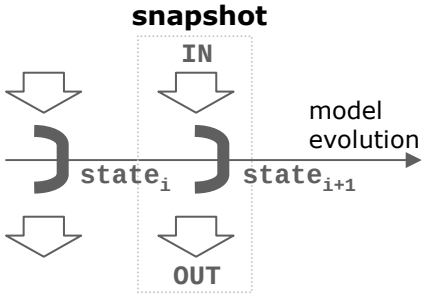
- Example

After update

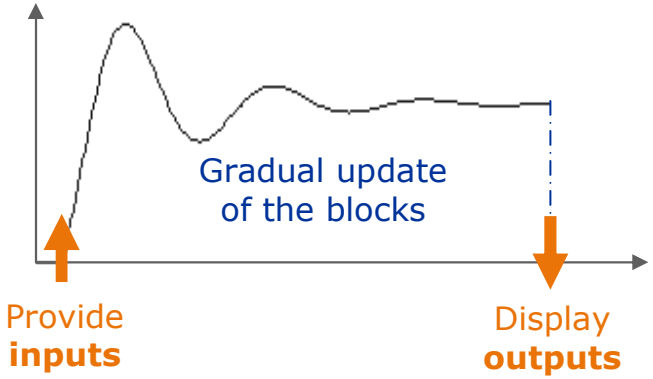


Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



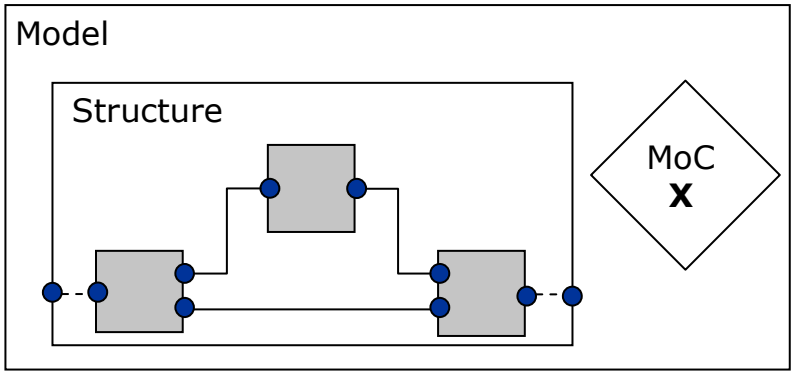
- One snapshot =



  - Causal execution
    - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
  - Gradual update of the model blocks
    - Schedule of the block to update
    - Update of the block
    - Propagation of the produced data

- Example

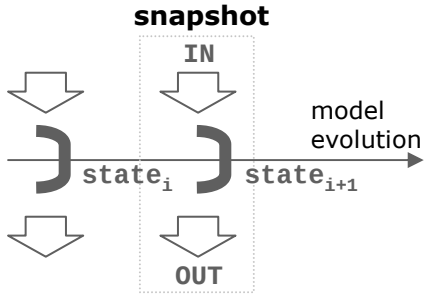
Display outputs



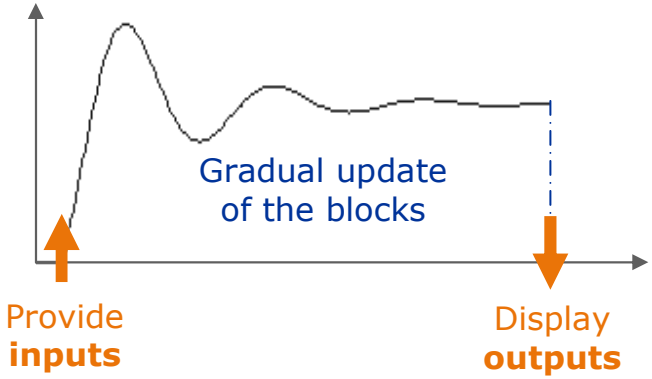
All the outputs are known
→ the snapshot is
determined

Executing an heterogeneous model

- One execution =
  - A sequence of successive snapshots of the model



- One snapshot =



  - Causal execution
    - State$_i$ + **inputs$_i$** → **outputs$_i$** + State$_{i+1}$
  - Gradual update of the model blocks
    - Schedule of the block to update
    - Update of the block
    - Propagation of the produced data

- Generic execution algorithm
  - A set of generic operations
  - Semantics specified using our language
  - Hierarchical execution
    - InterfaceBlocks have special operations in order to adapt the semantics between MoCs

Intended workflow & needed effort

ONCE !

1. An expert of a modeling language describes
   - The structural and semantic elements of the language
     - Specialized meta-model
     - Imperative semantics of the MoC
   - Transformations from the original meta-model of the language to the ModHel'X meta-model for maximizing the reuse of existing models
2. Experts define interaction patterns for each pair of model of computation that may interact
   - Interaction pattern = "classical glue"
   - Parameters allow the designers to fine tune the adaptation
3. Designers use ModHel'X

Conclusion & perspectives

- Conclusion: ModHel'X is a generic and modular framework for executing heterogeneous models with
  - A generic meta-model for representing heterogeneous models
  - A generic algorithm for executing heterogeneous models
  - A language for specifying the semantics of models of computations and of their interactions
- Applications: (joint) simulation, code generation, etc.

- Work in progress
  - Prototype based on the Eclipse Modeling Framework (EMF)
    - Several implemented MoCs
  - Concrete syntax of our language (OMG ImperativeOCL – QVT)
  - Formal semantics of our algorithm and language

- Perspectives
  - Facets (non-functional properties for embedded systems)
  - Model refinement & symbolic execution

Bibliography

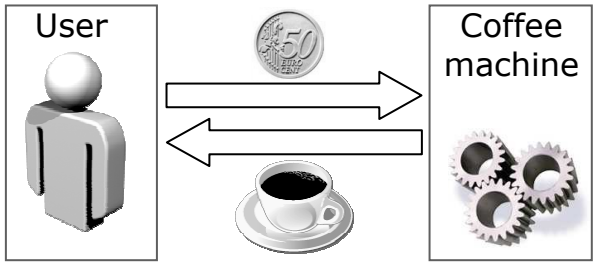
- [Muller05] Muller, P.-A., F. Fleurey and J.-M. J´ez´equel, Weaving executability into object-oriented meta-languages, in: Proceedings of the 8th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS/UML 2005), 2005, pp. 264–278.
- [deLara02] de Lara, J. and H. Vangheluwe, ATOM3: A tool for multi-formalism modelling and meta-modelling, in: 5th Fundamental Approaches to Software Engineering International Conference (FASE 2002), 2002, pp. 595–603.
- [Lee03] Eker, J., J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs and Y. Xiong, Taming heterogeneity – the Ptolemy approach, Proceedings of the IEEE, Special Issue on Modeling and Design of Embedded Software 91 (2003), pp. 127–144.
- [Sifakis06] Basu, A., M. Bozga and J. Sifakis, Modeling heterogeneous real-time systems in BIP, in: 4th IEEE International Conference on Software Engineering and Formal Methods (SEFM06), 2006, pp. 3–12.
- [Lee98] Lee, E. A. and A. L. Sangiovanni-Vincentelli, A framework for comparing models of computation, IEEE Trans. on CAD of Integrated Circuits and Systems 17 (1998), pp. 1217–1229.
- [Kong03] C. Kong and P. Alexander. The Rosetta meta-model framework. In Proceedings of the IEEE Engineering of Computer-Based Systems Symposium and Workshop (ECBS'03), april 2003

ÉCOLE SUPÉRIEURE D'ÉLECTRICITÉ

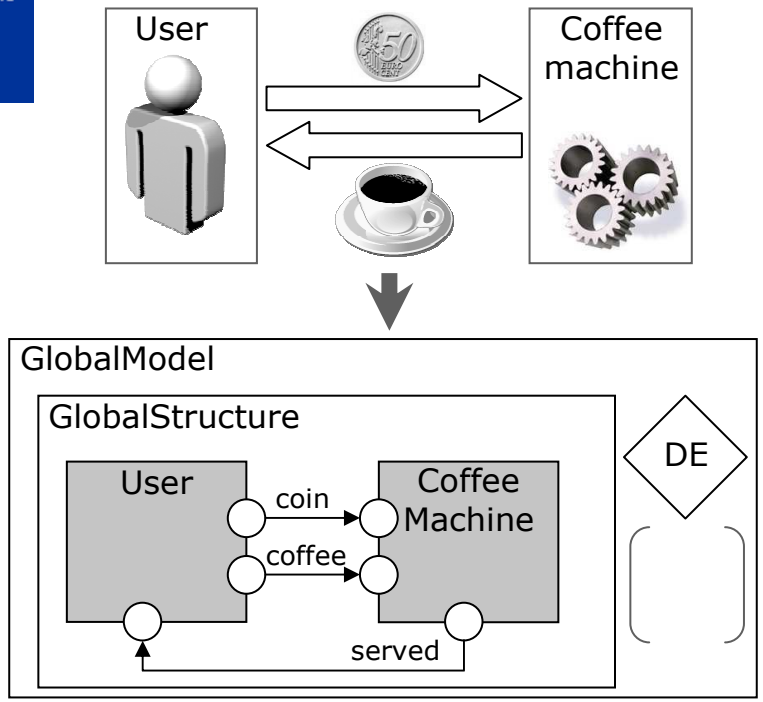
Supélec

Appendix

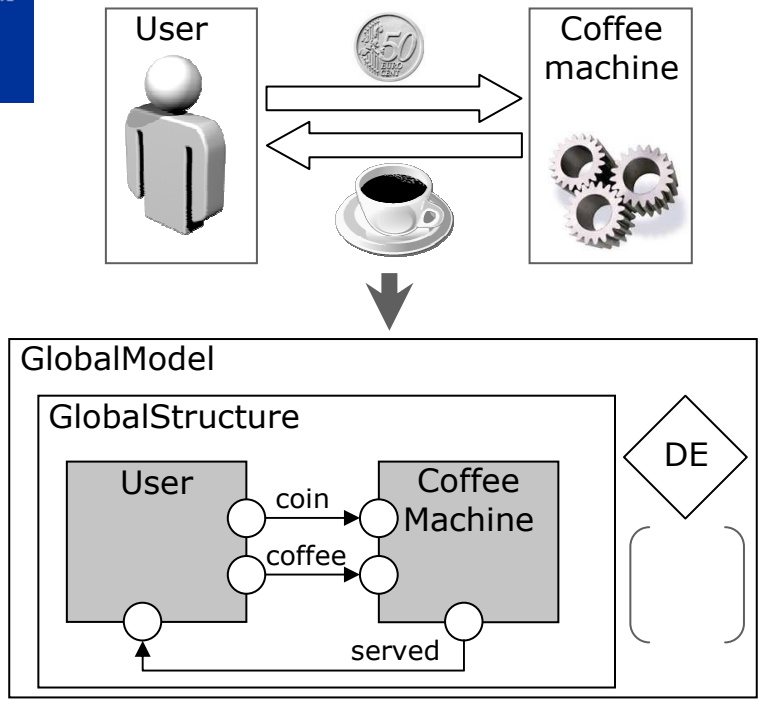
The coffee machine example



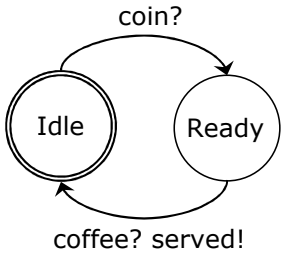
The coffee machine example



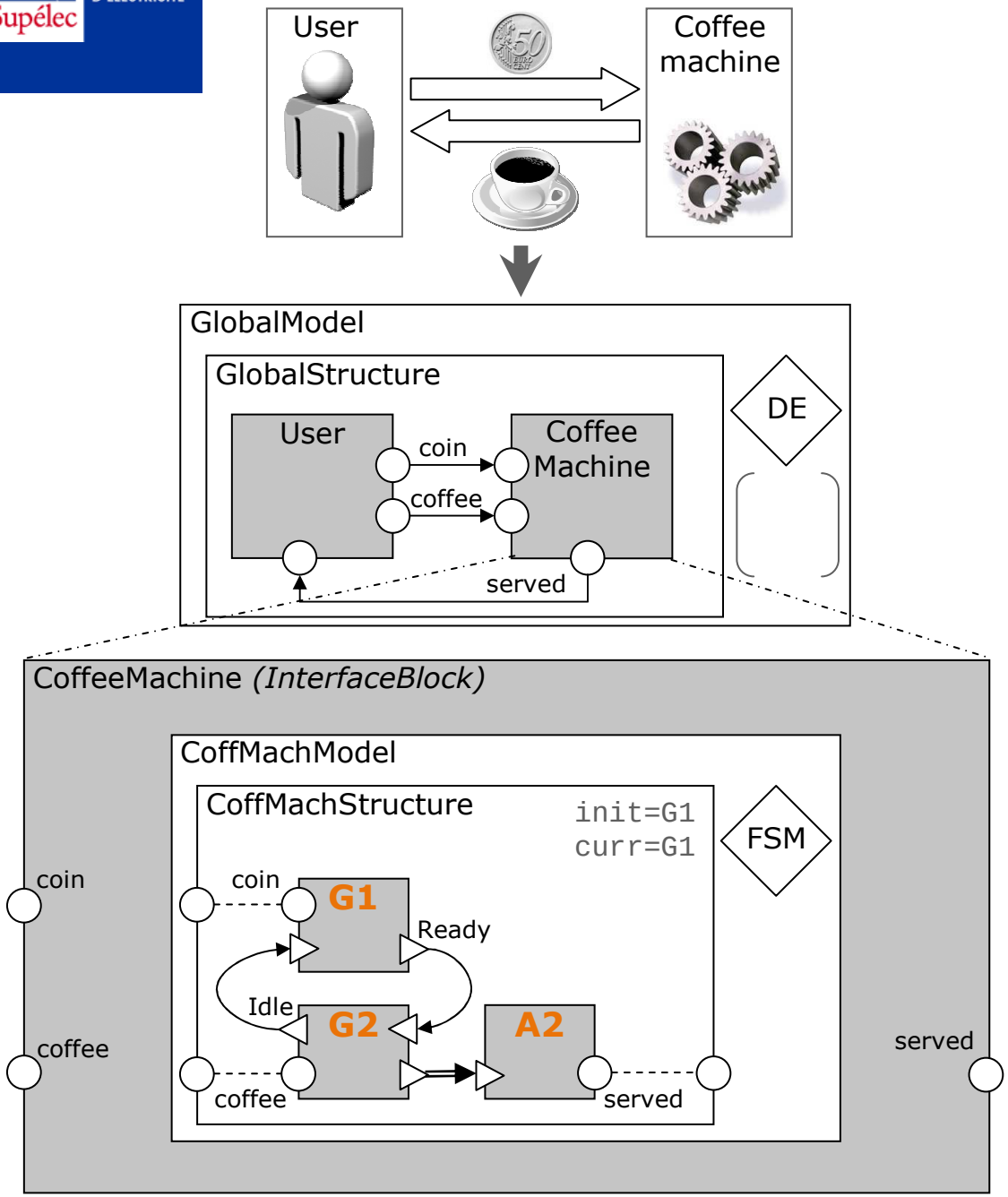
The coffee machine example



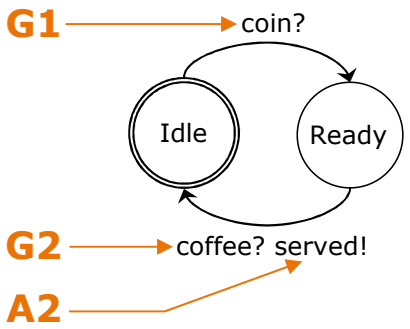
- Coffee machine automaton



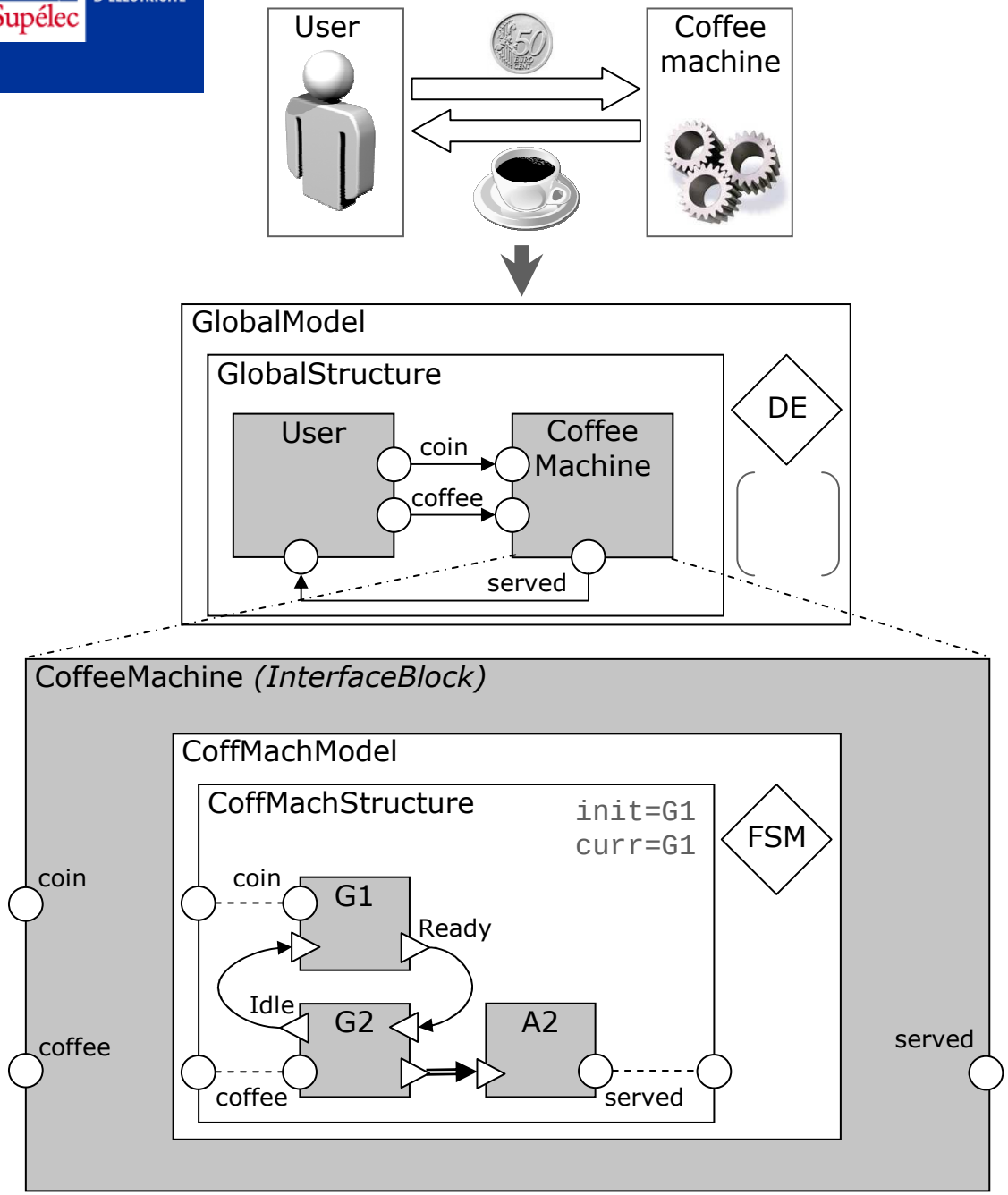
The coffee machine example



- Coffee machine automaton

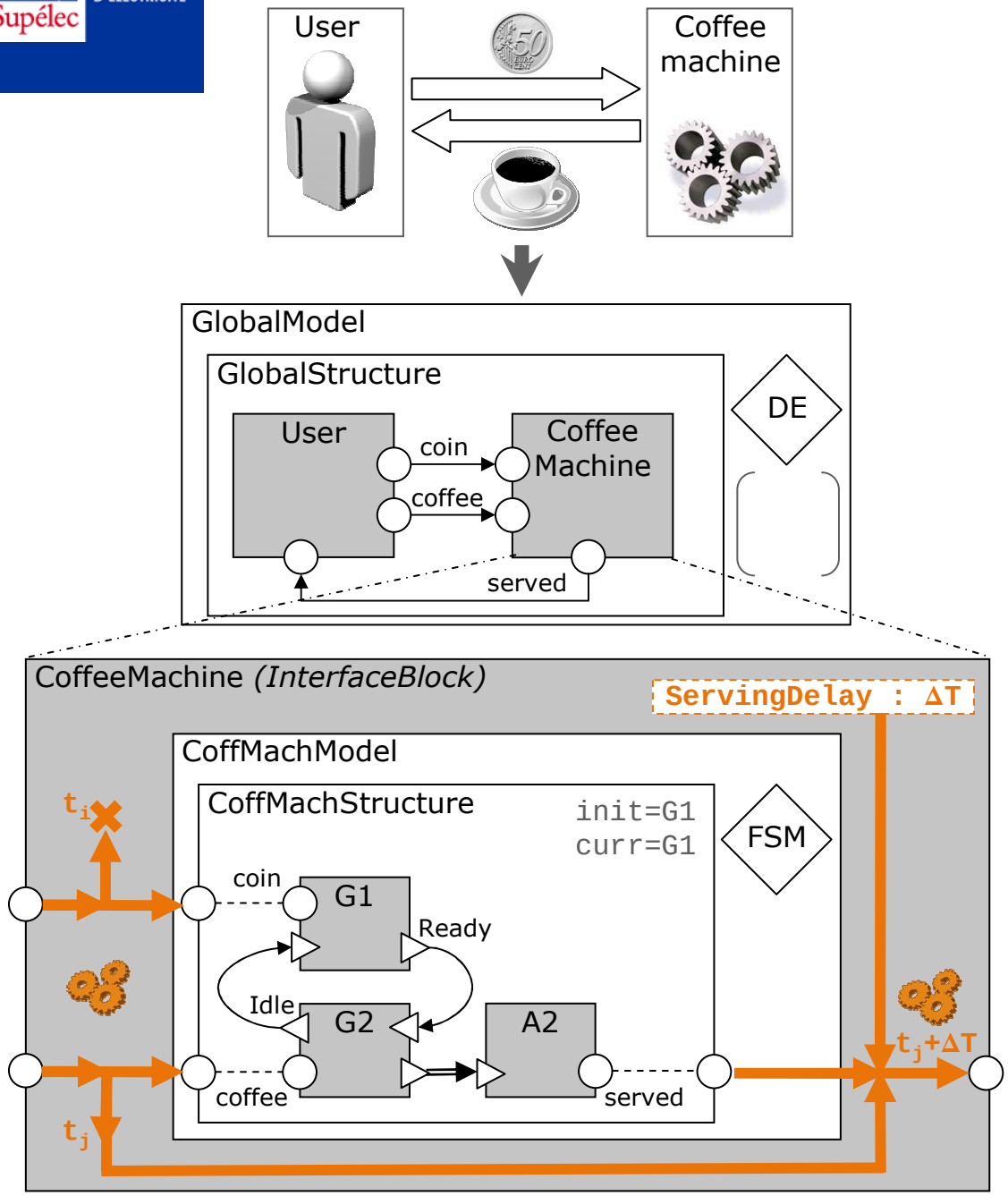


The coffee machine example



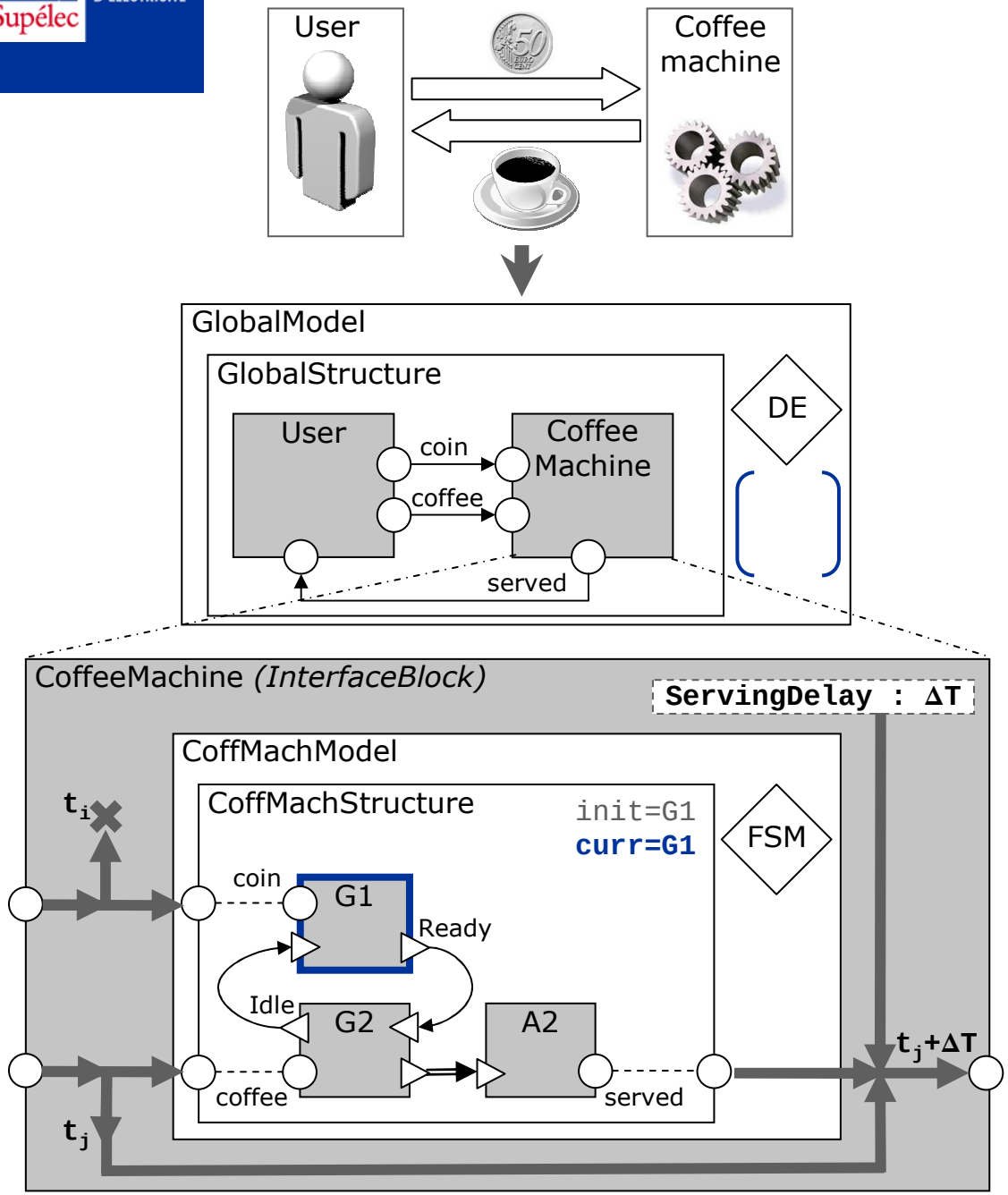
- Time “gluing” ?
  Serving delay ?

The coffee machine example



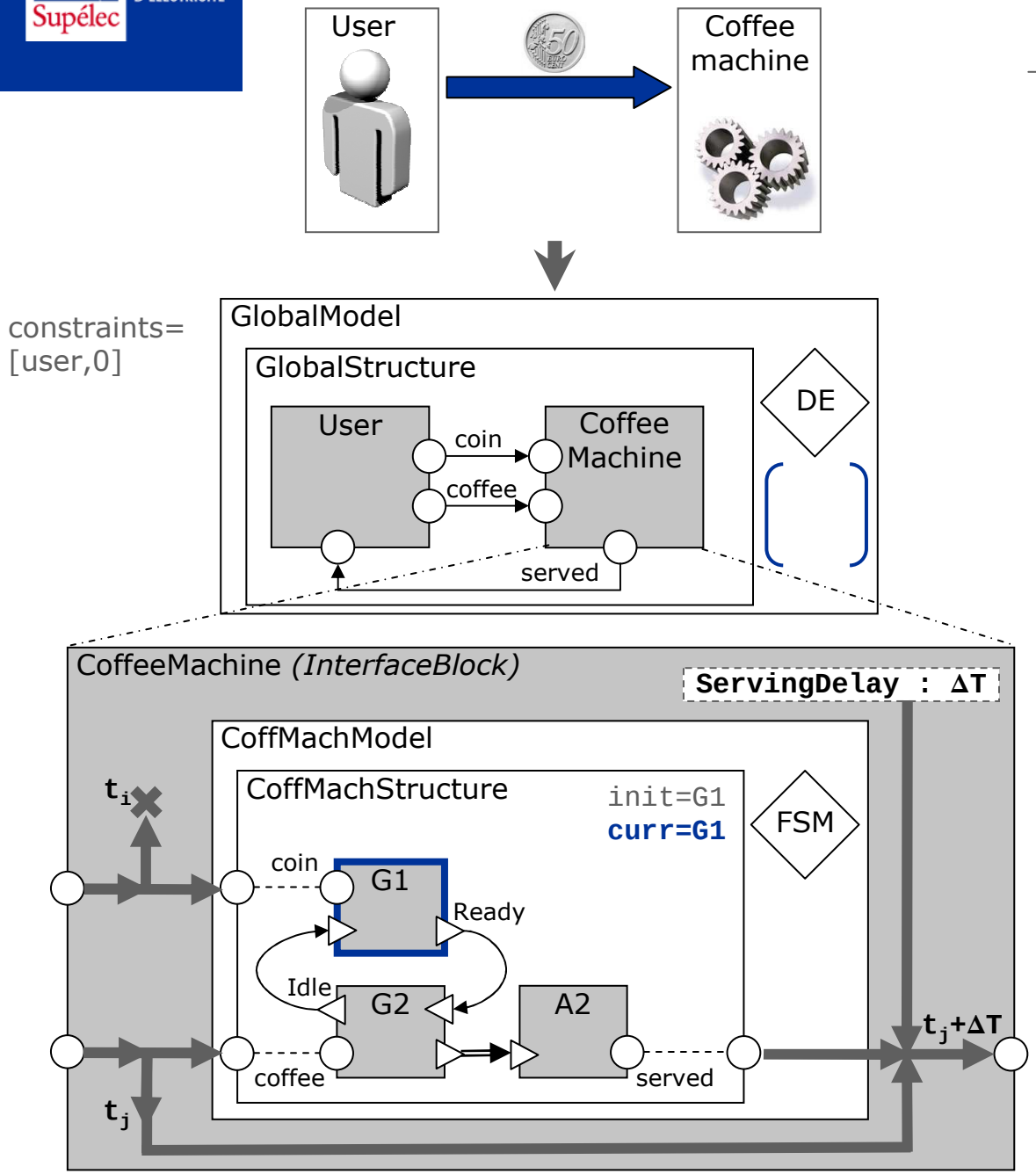
- Time "gluing" ? Serving delay ?
  - adaptIn
  - adaptOut

Running the model

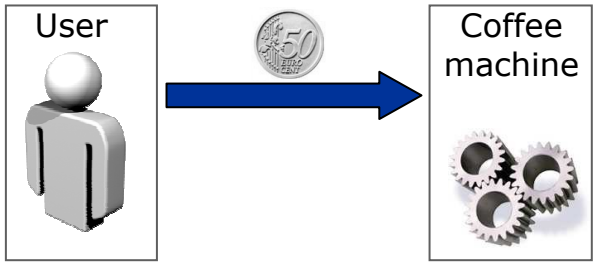


Running the model

- Constraint on the user

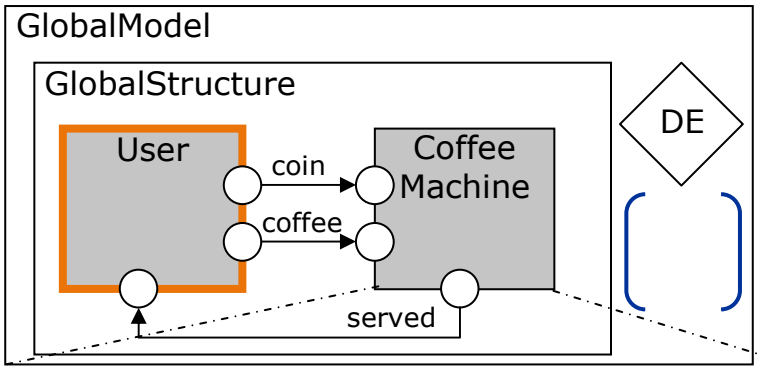


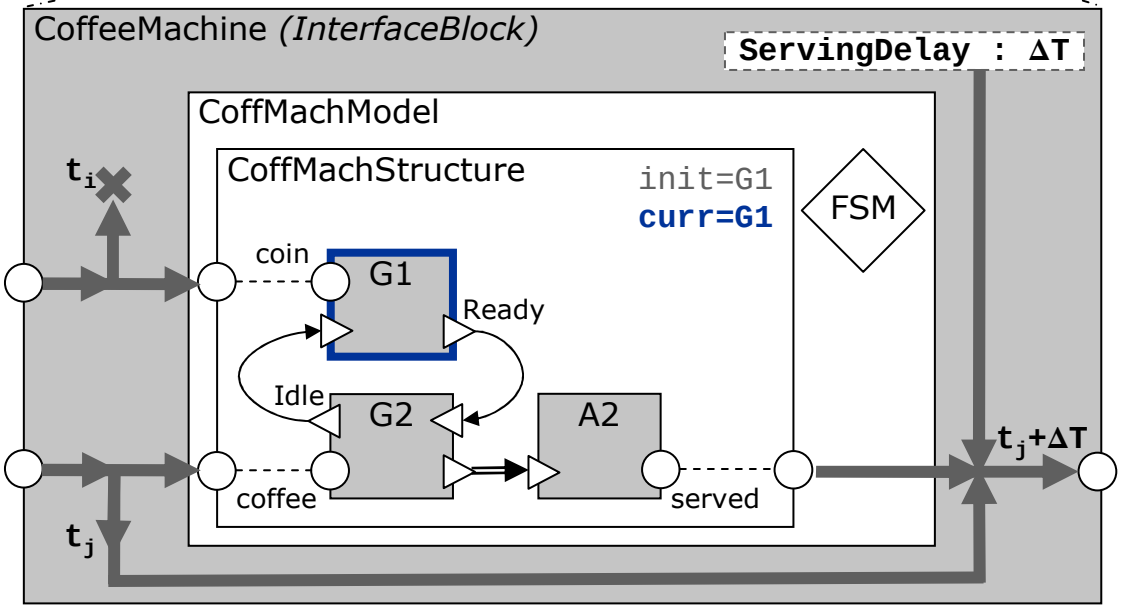
Running the model



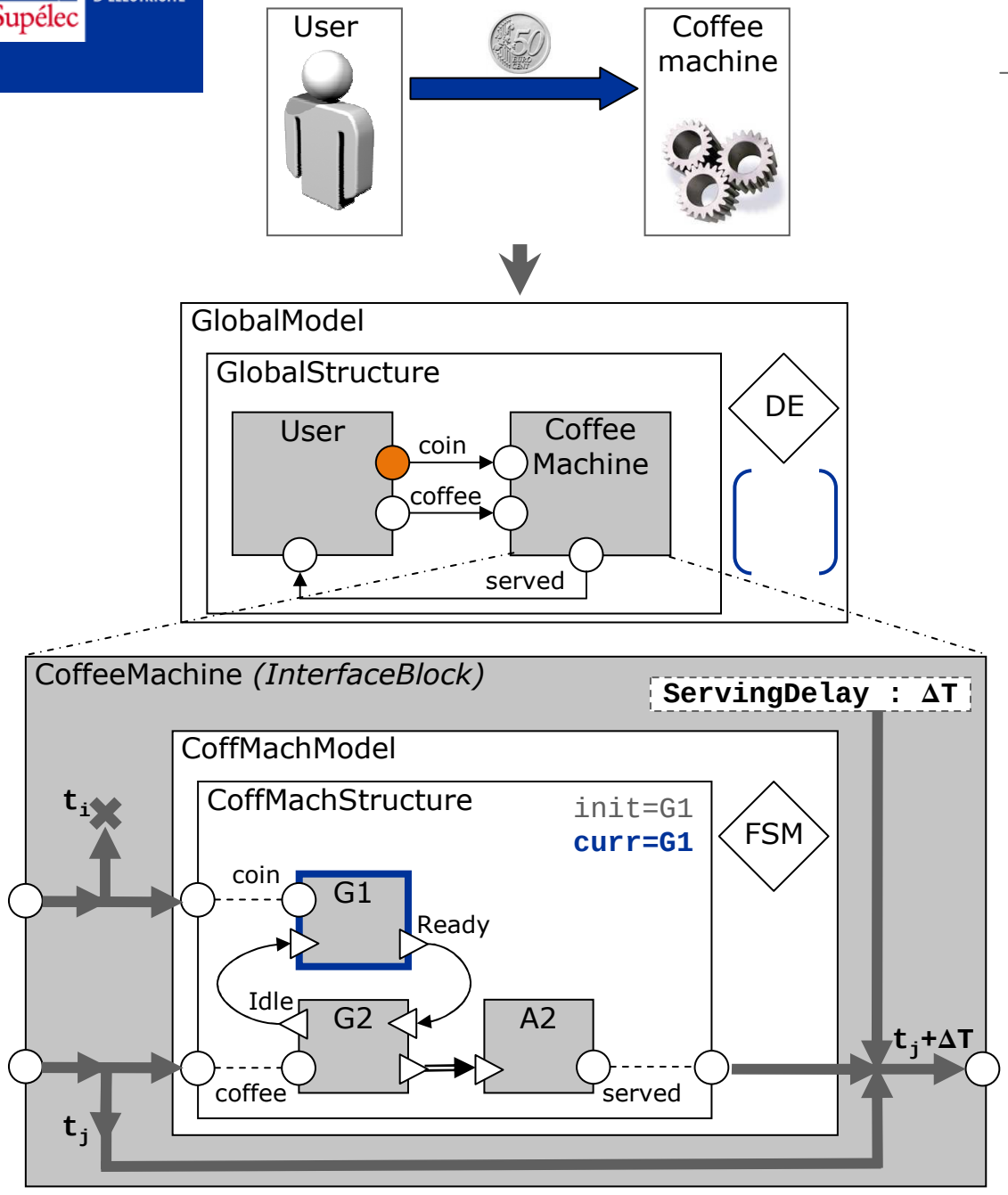
- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

constraints=
[user,0]



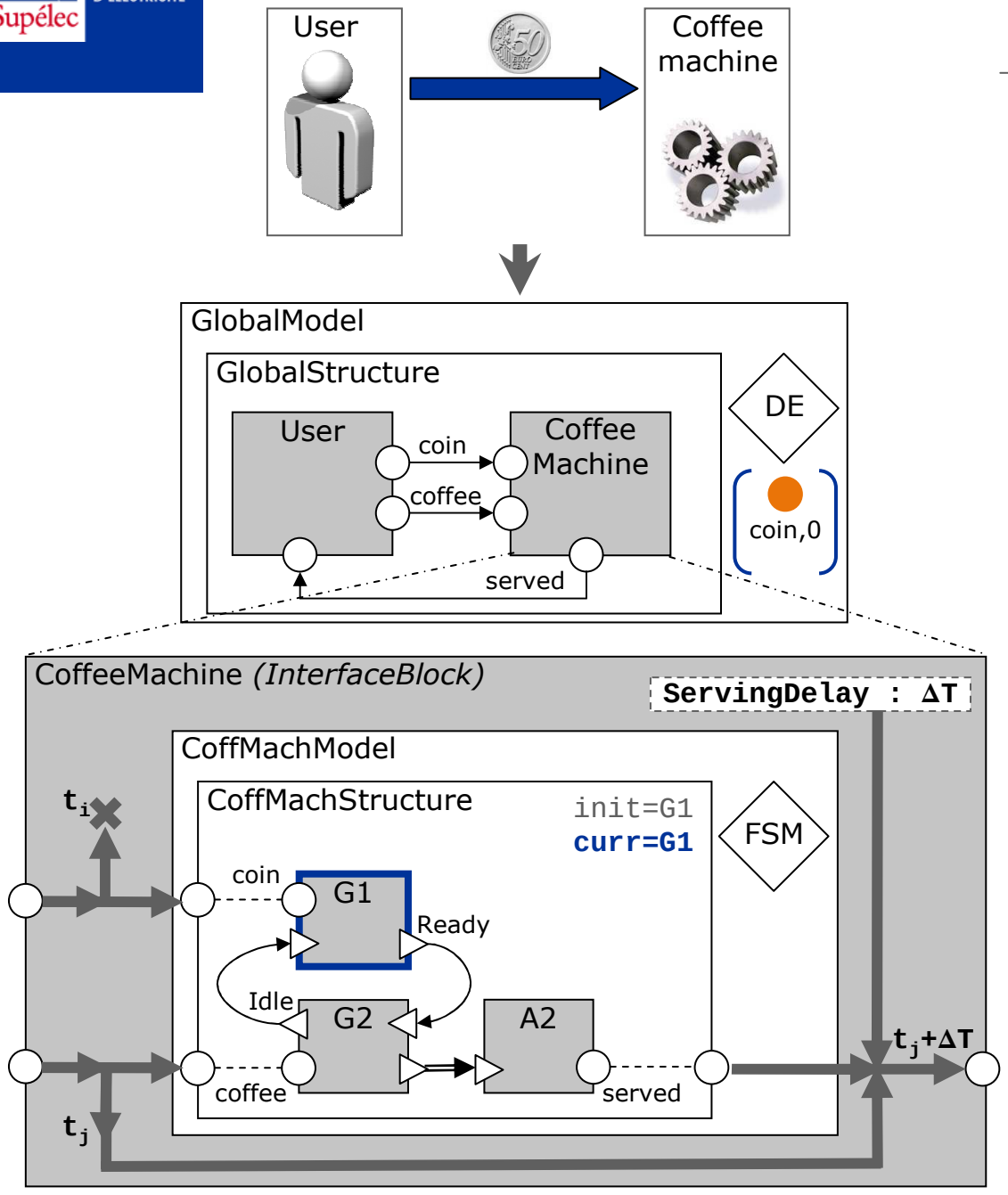


Running the model



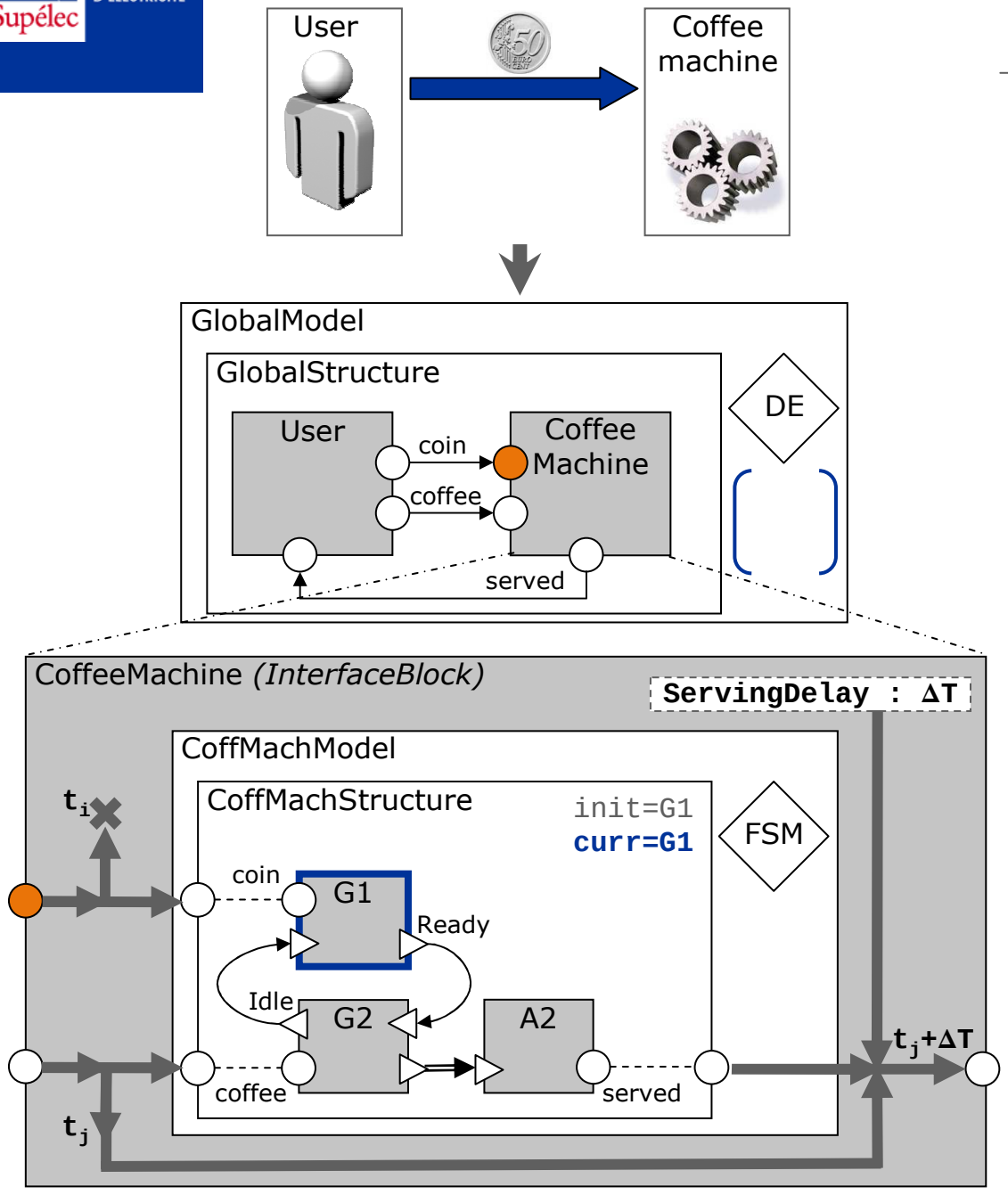
- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

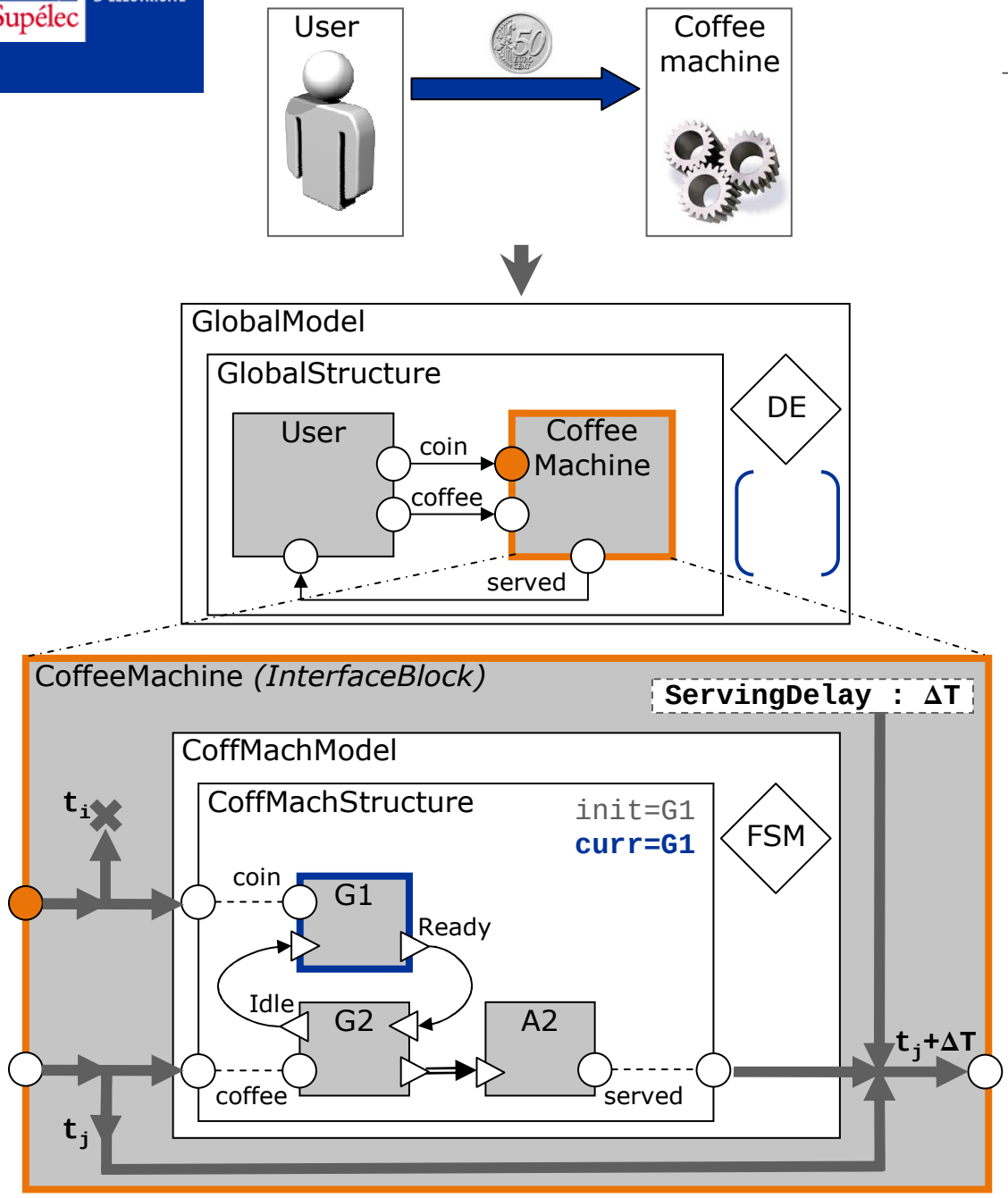
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

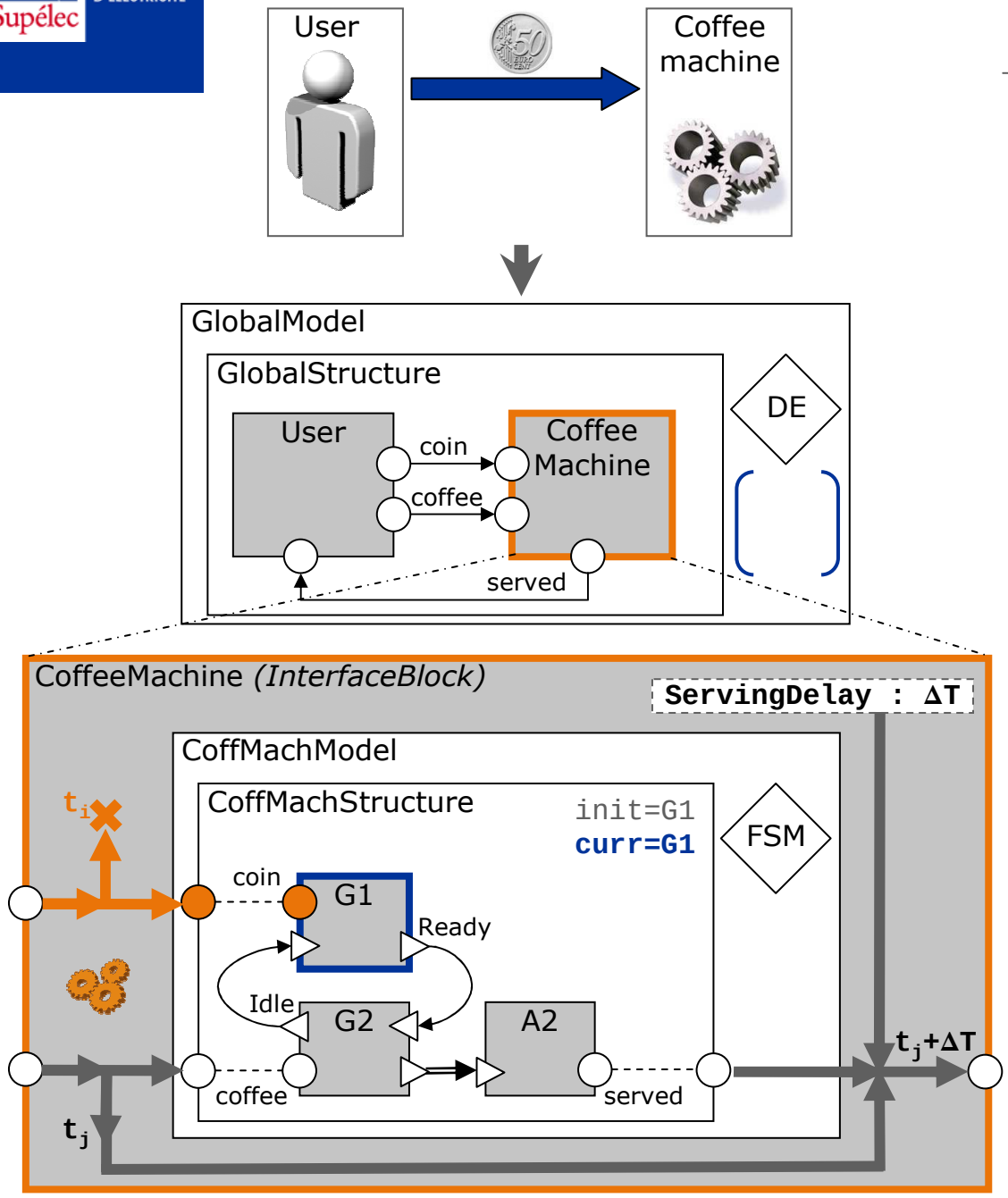
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

Running the model

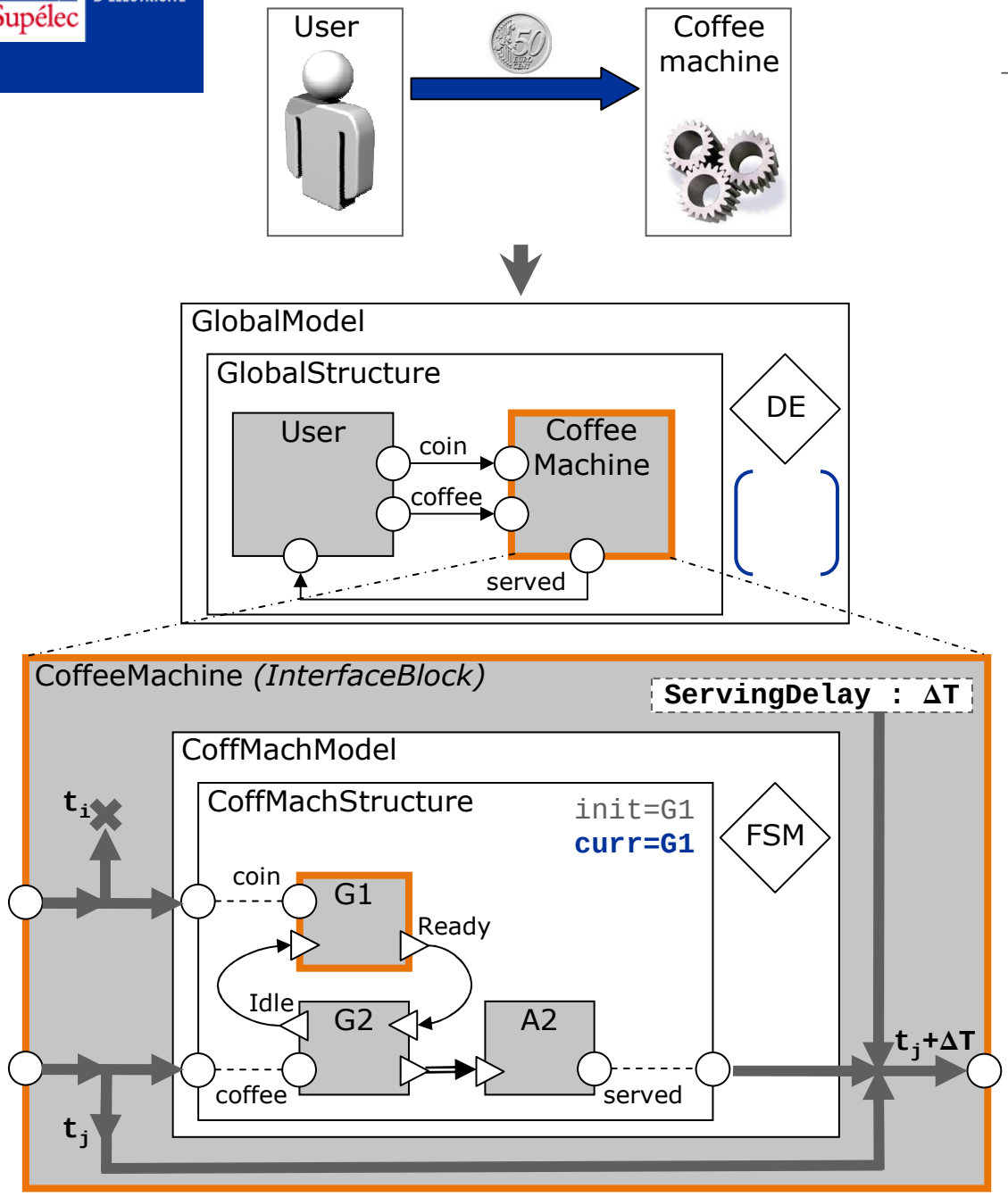


- Constraint on the user

- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

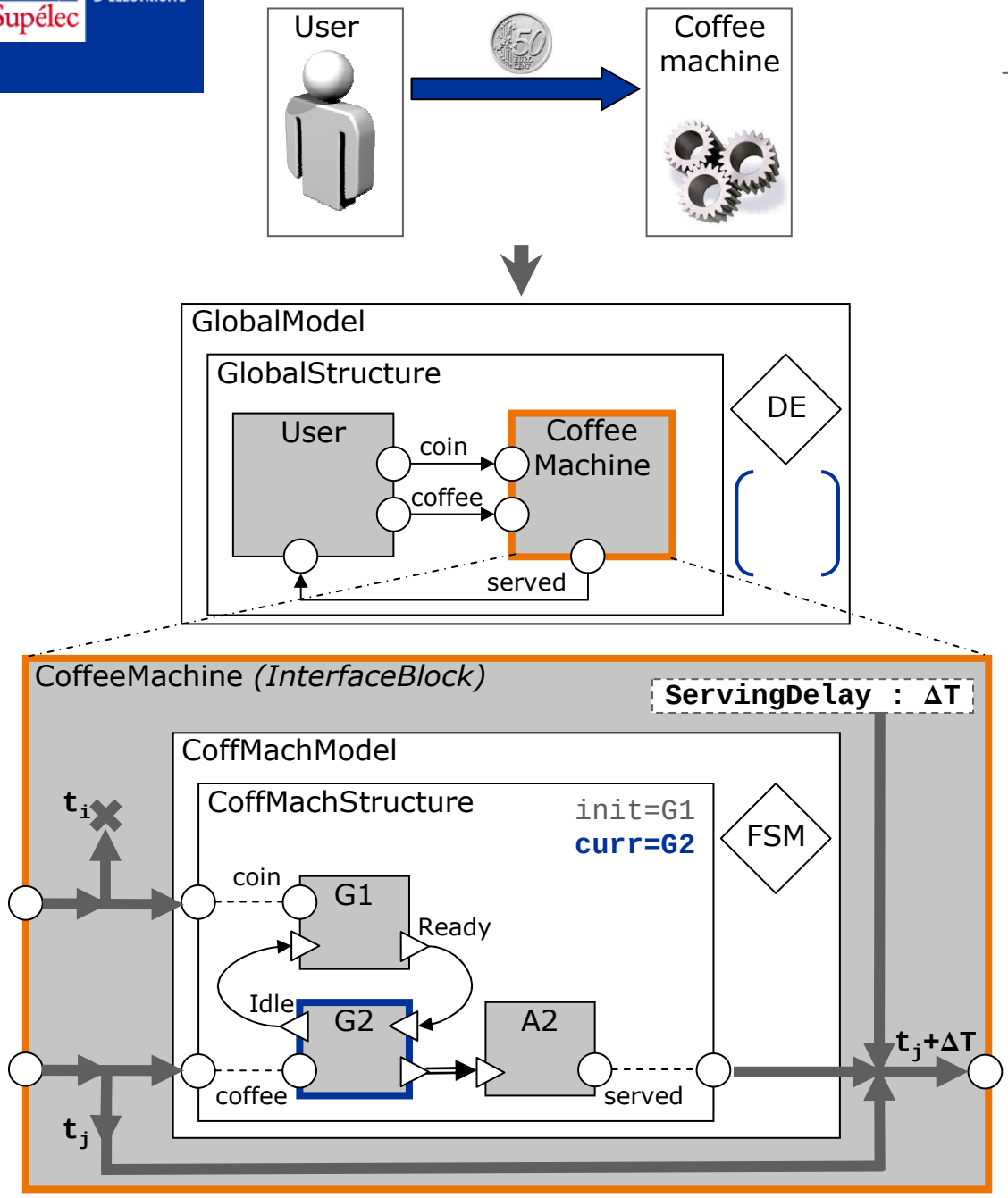
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

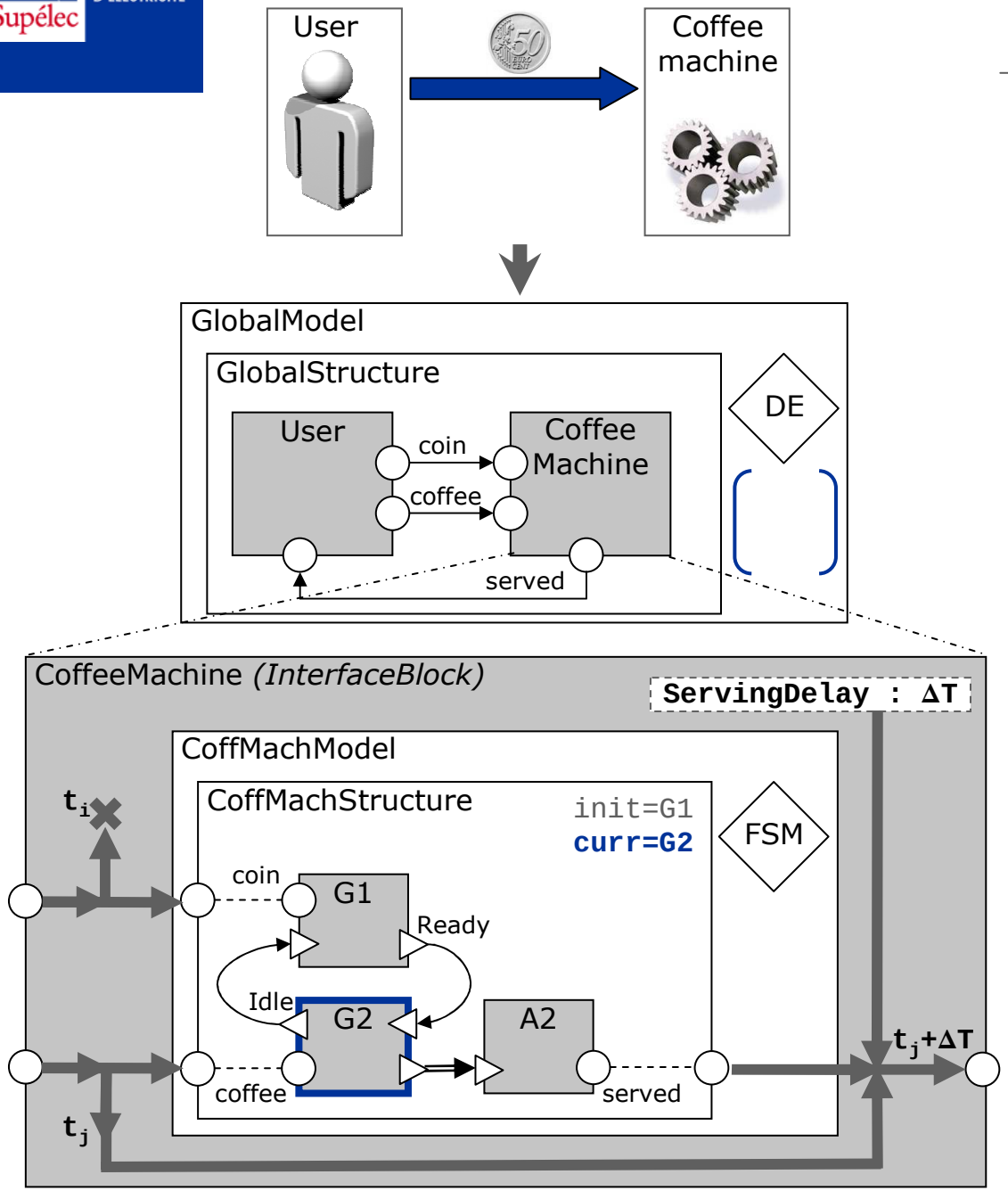
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

Running the model

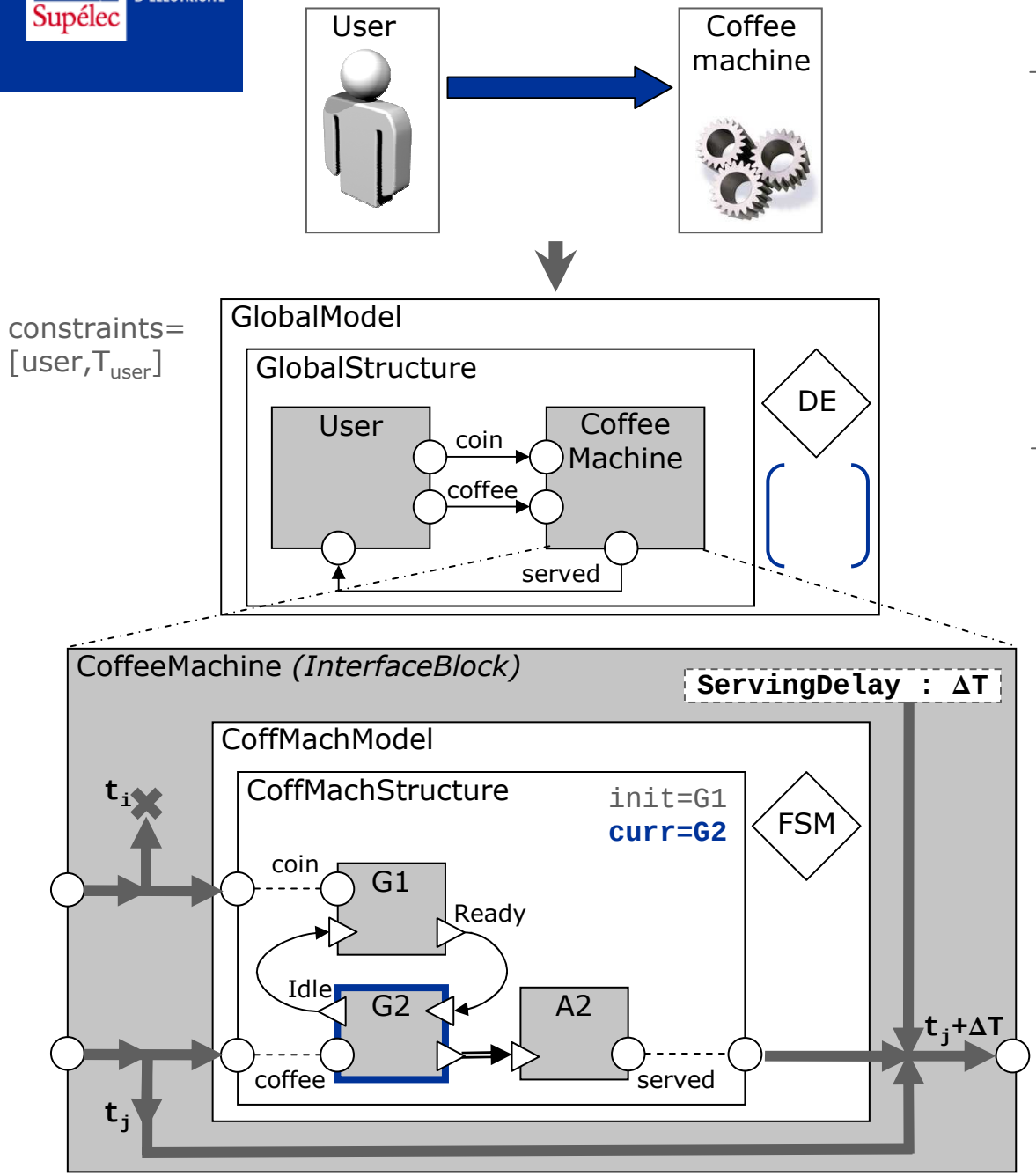


- Constraint on the user

- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"

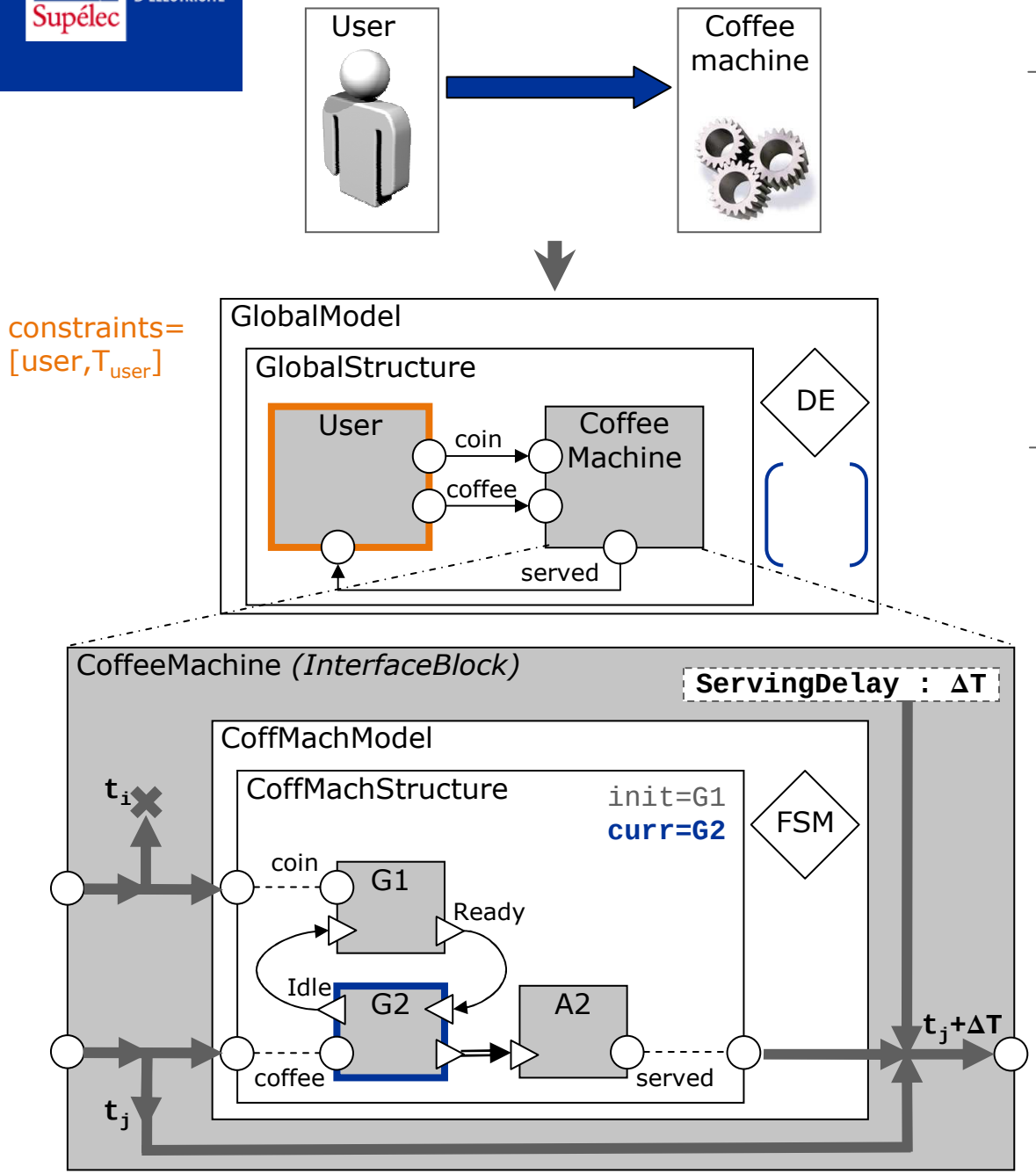
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

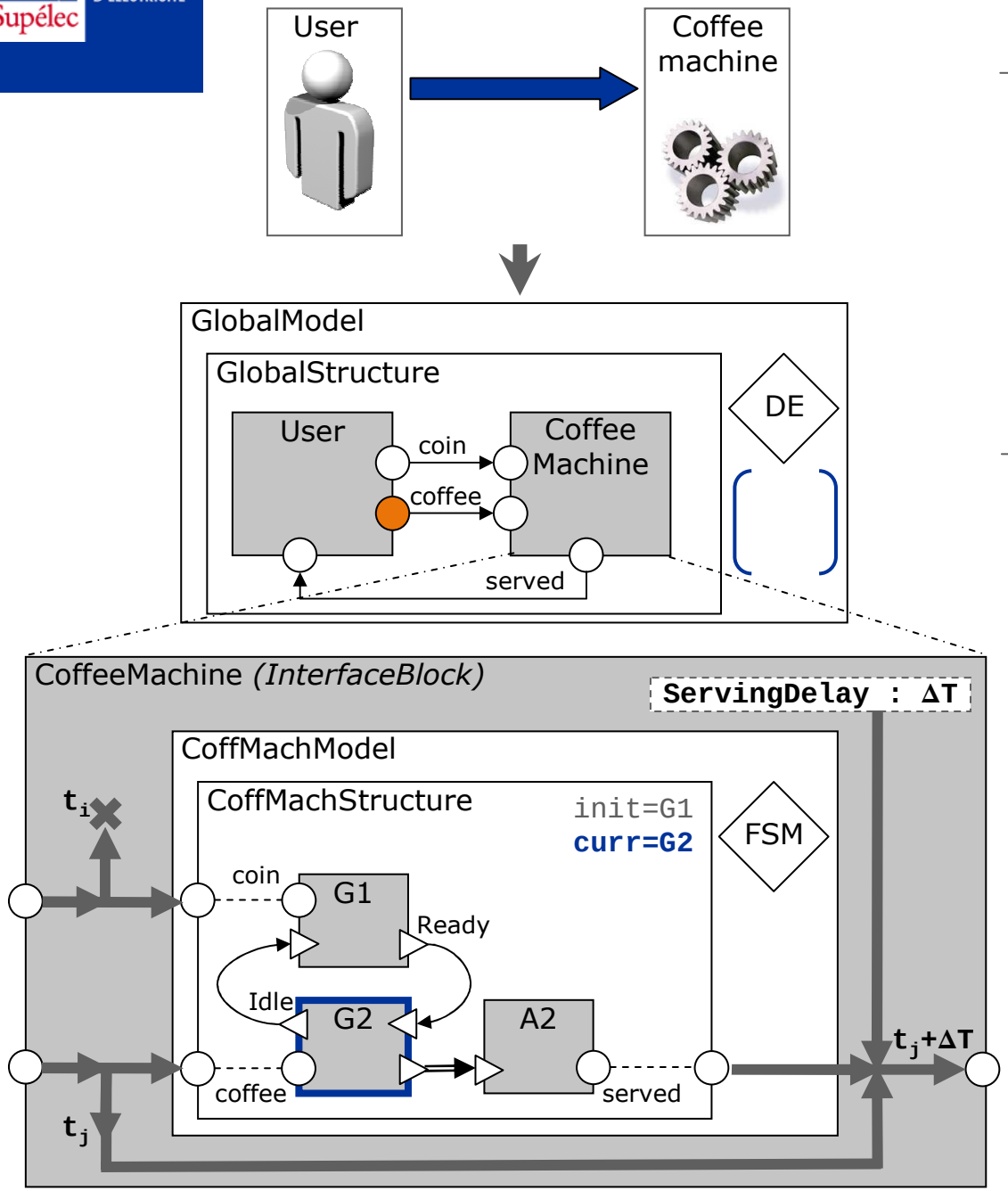
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

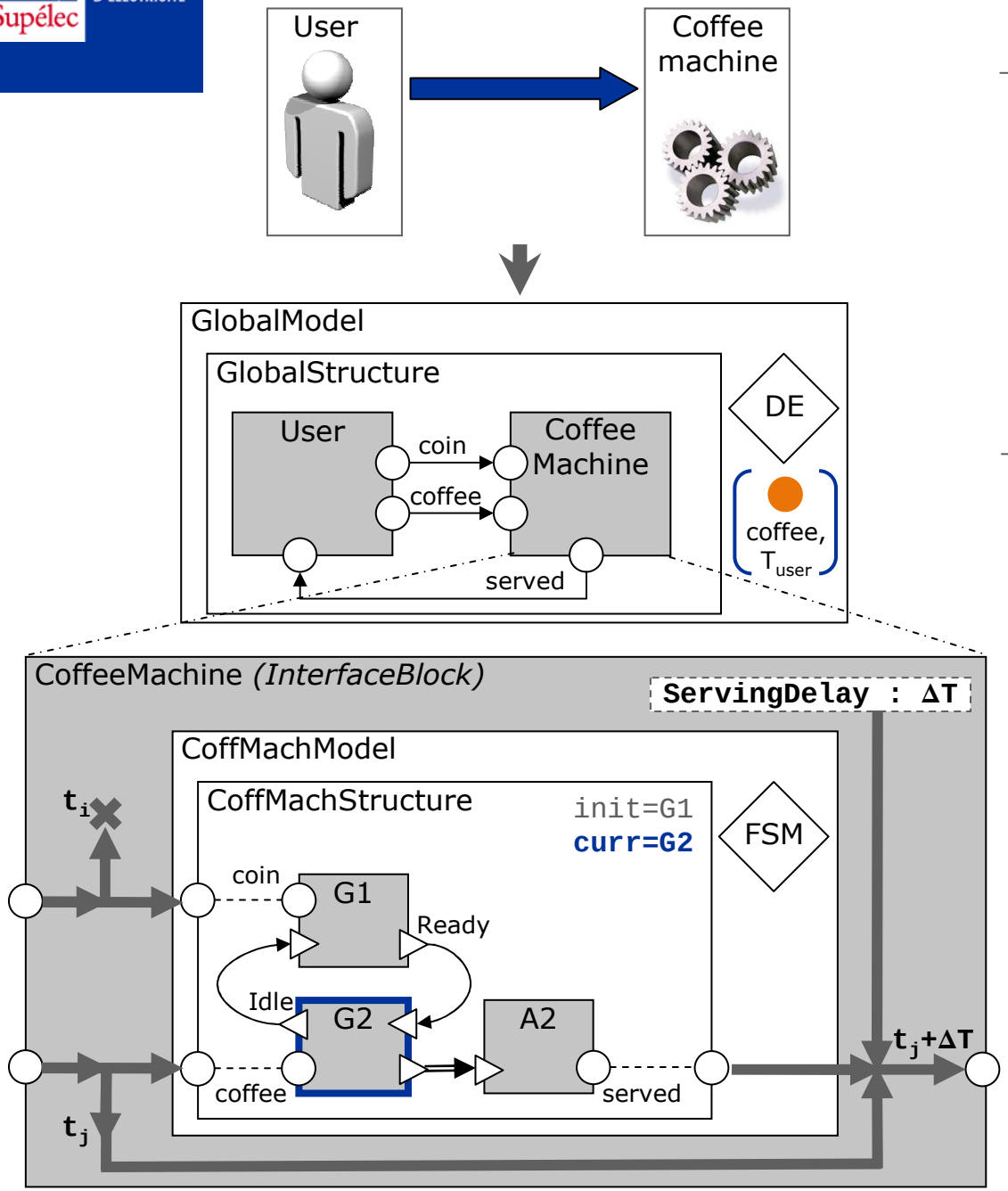
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

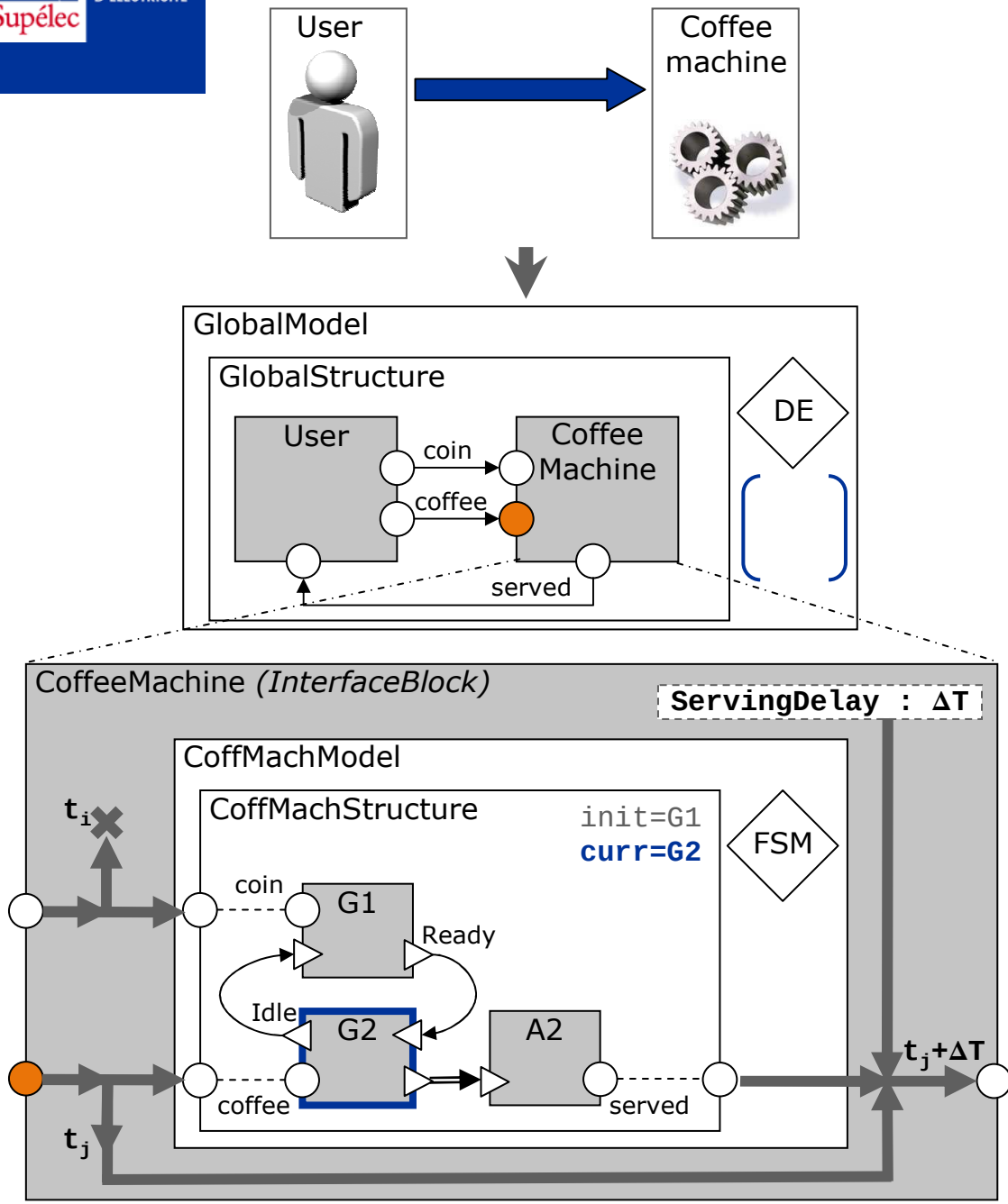
Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

Running the model



- Constraint on the user

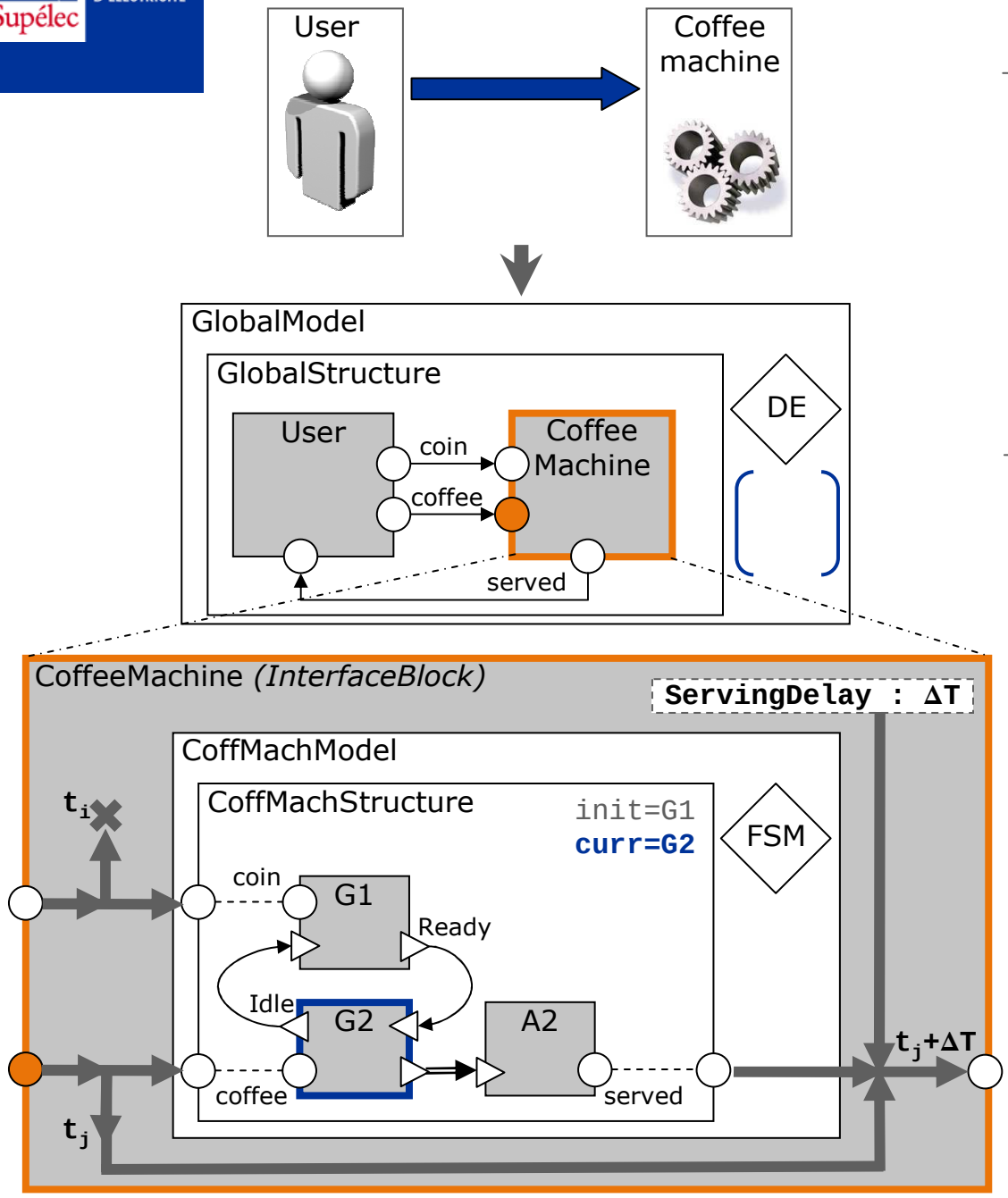
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model



- Constraint on the user

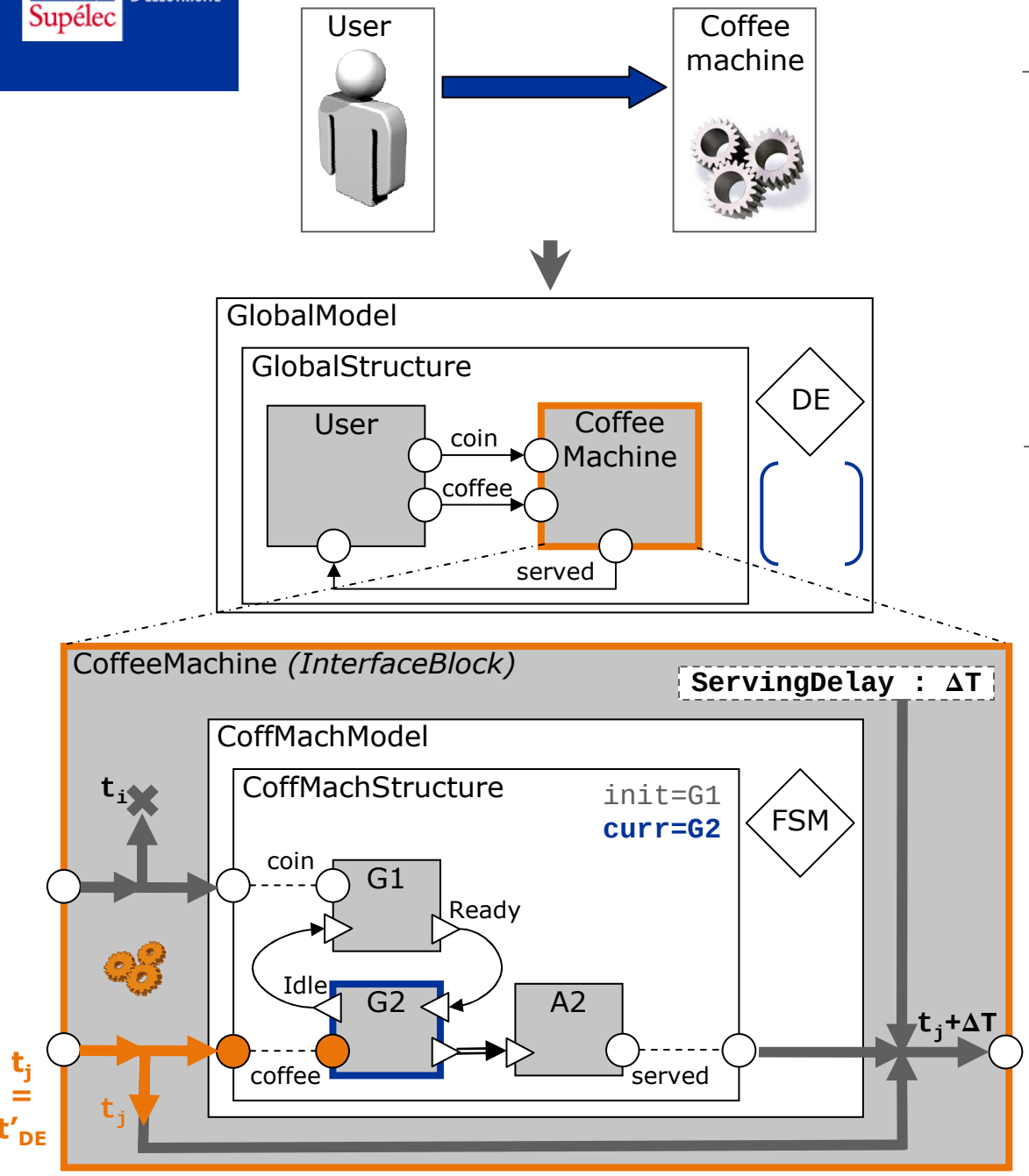
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model

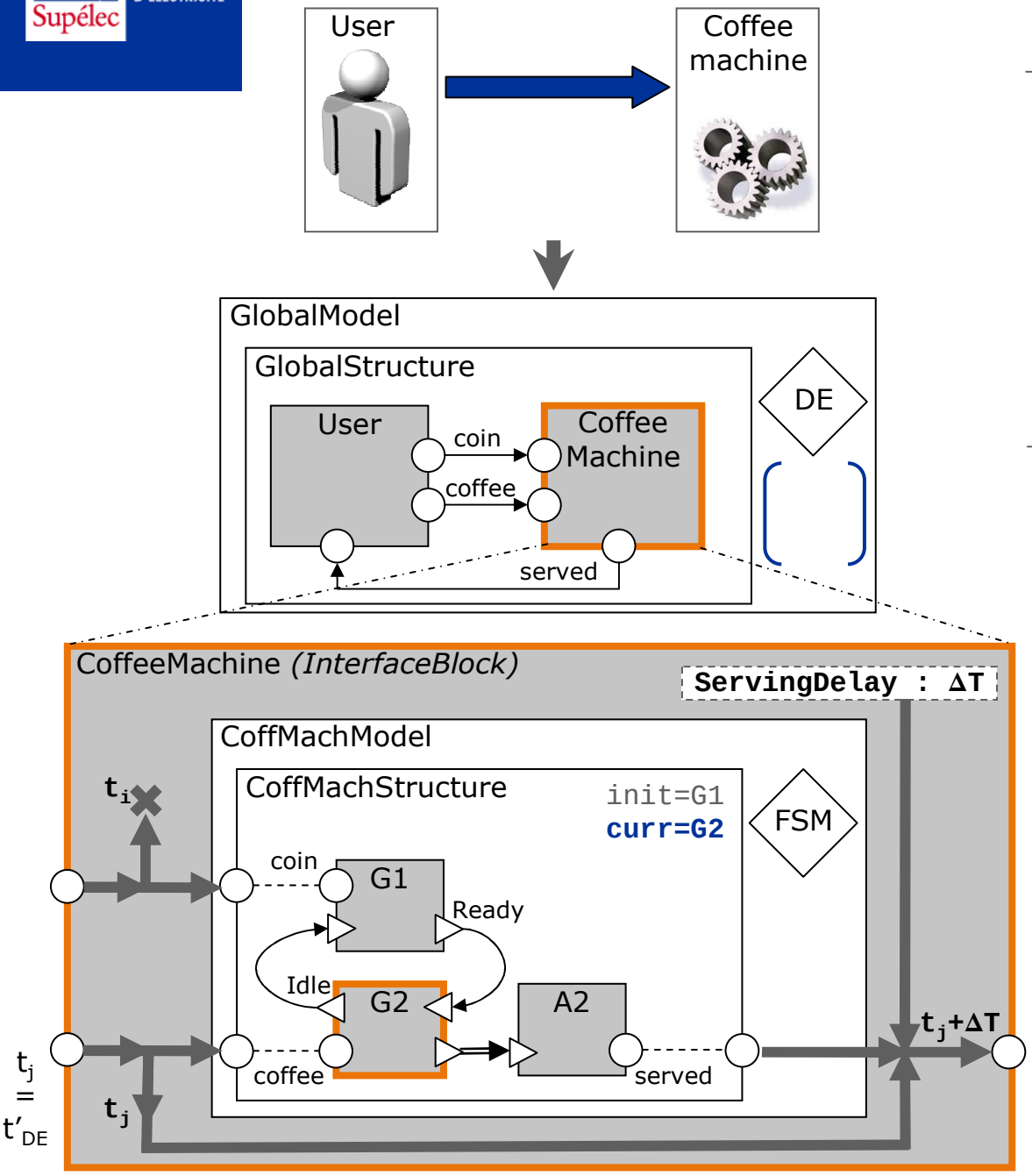


- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model



- Constraint on the user

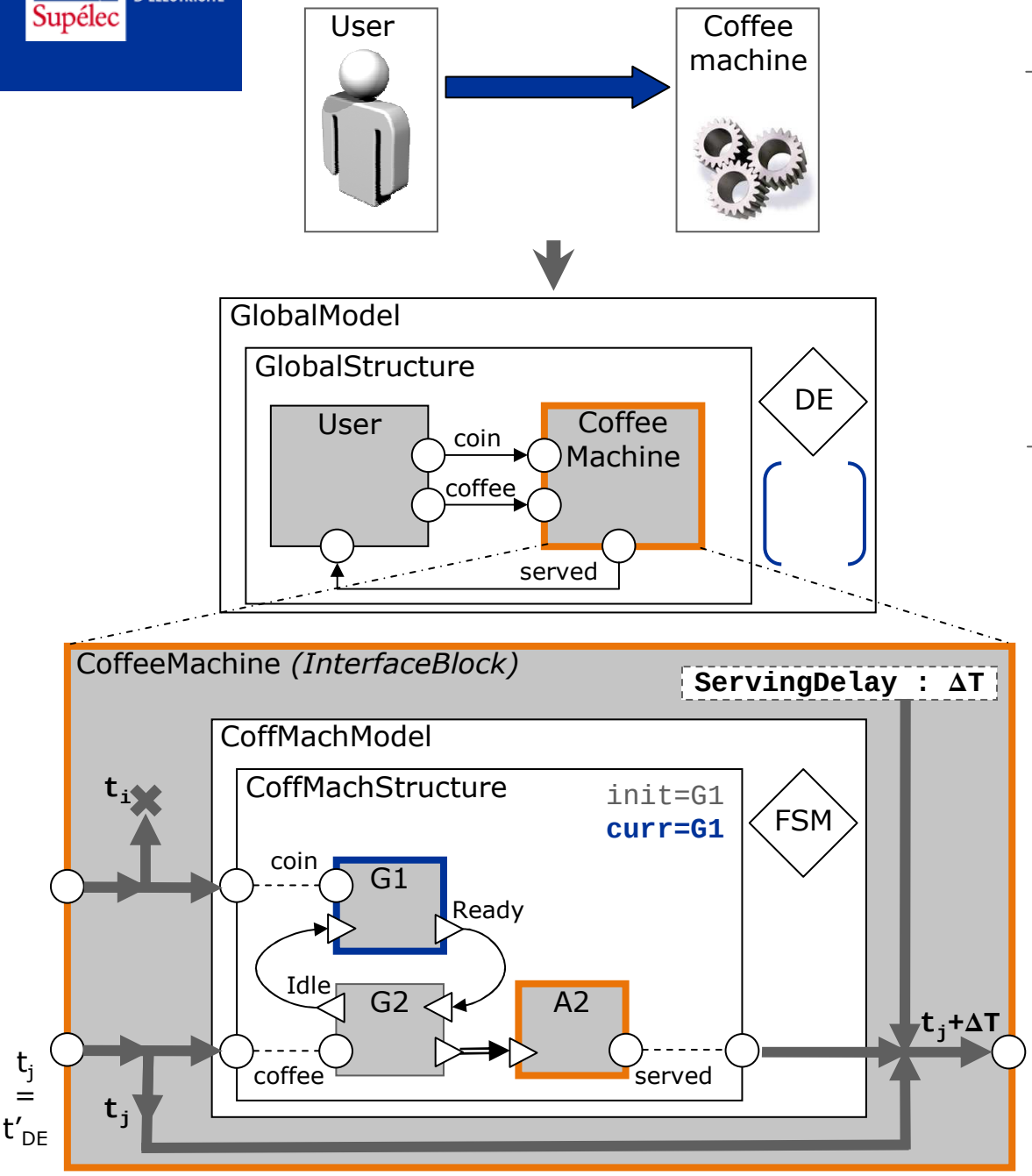
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model



- Constraint on the user

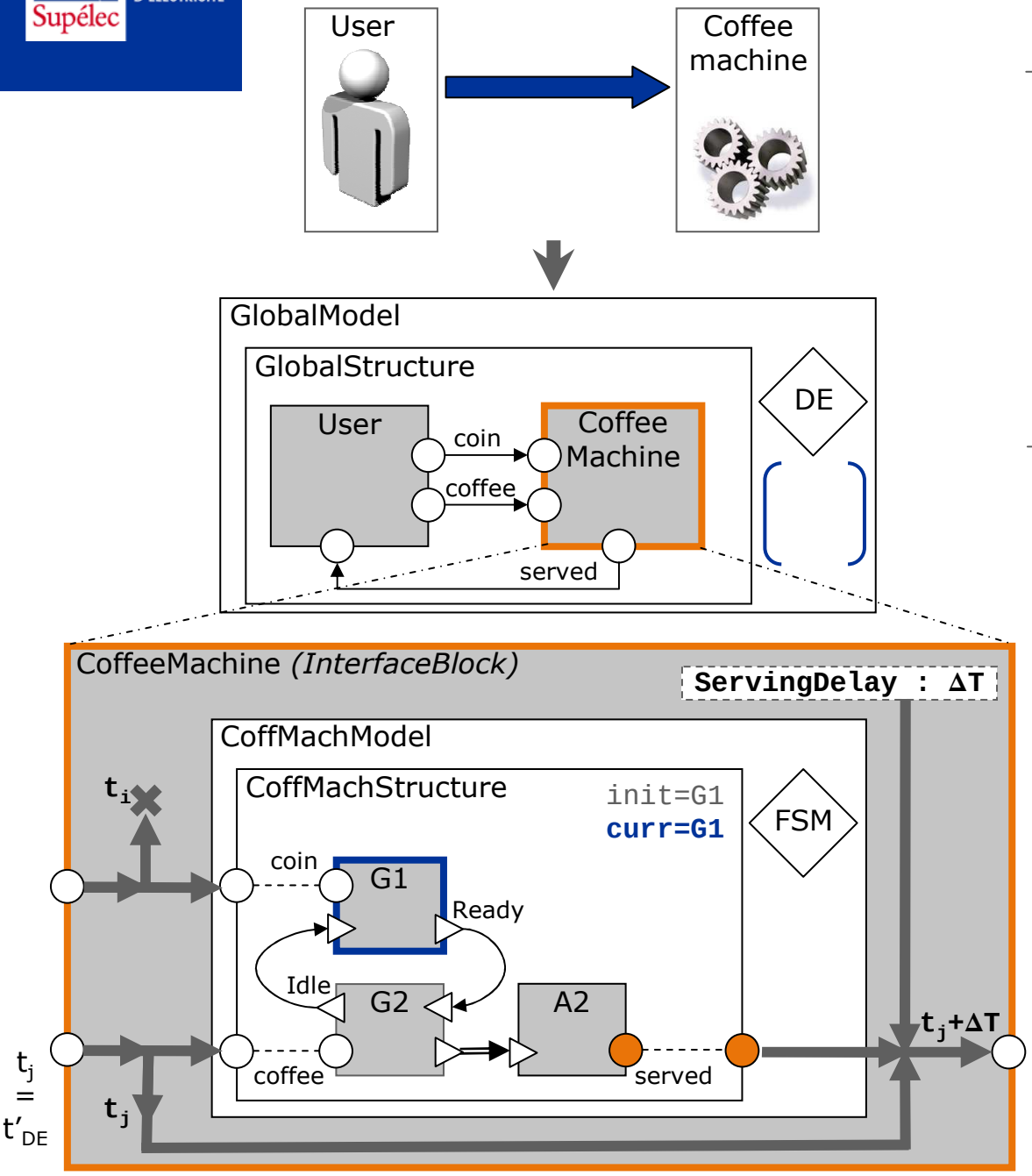
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model



- Constraint on the user

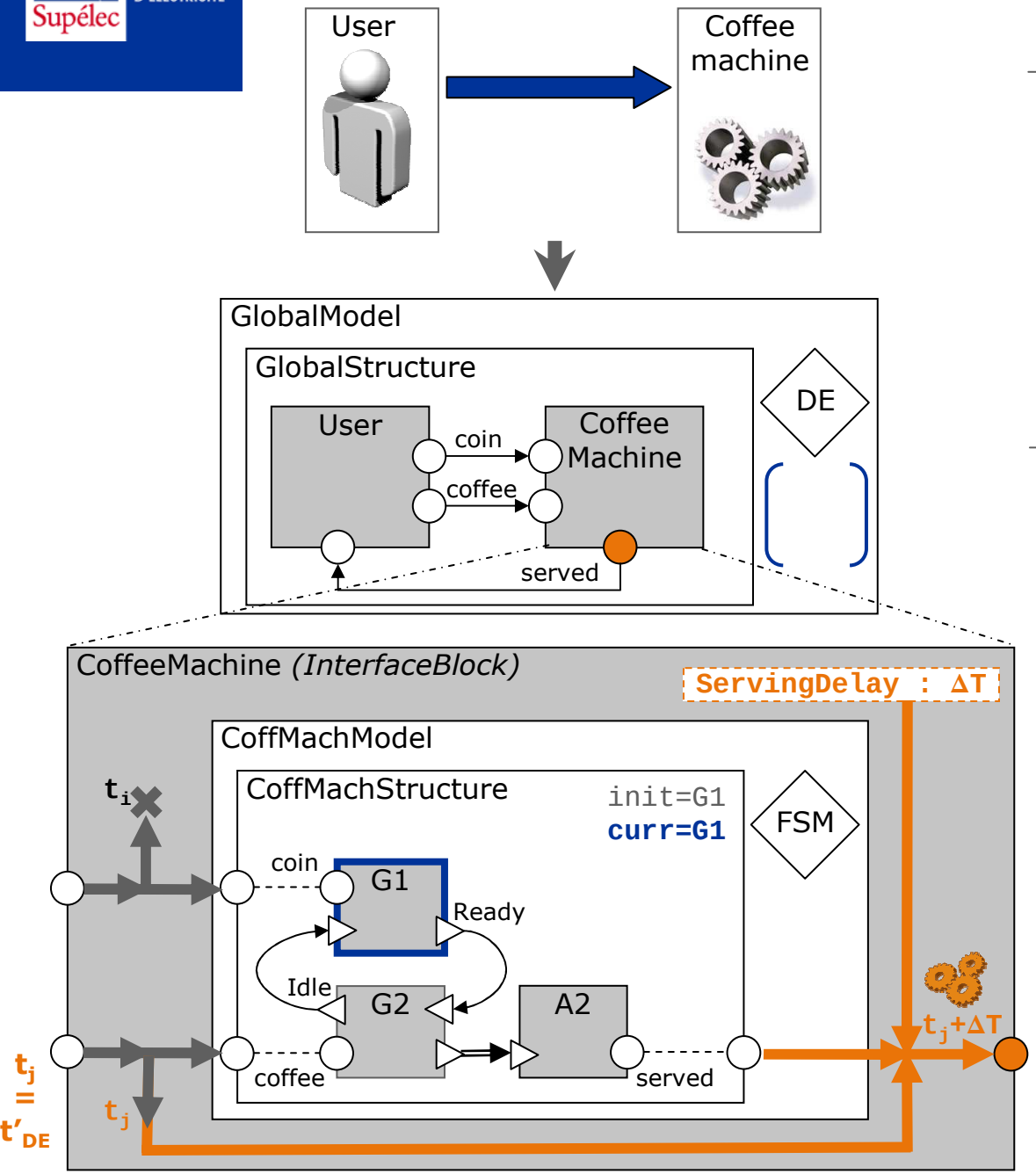
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model



- Constraint on the user

- First snapshot
  - $t_{DE} = 0$

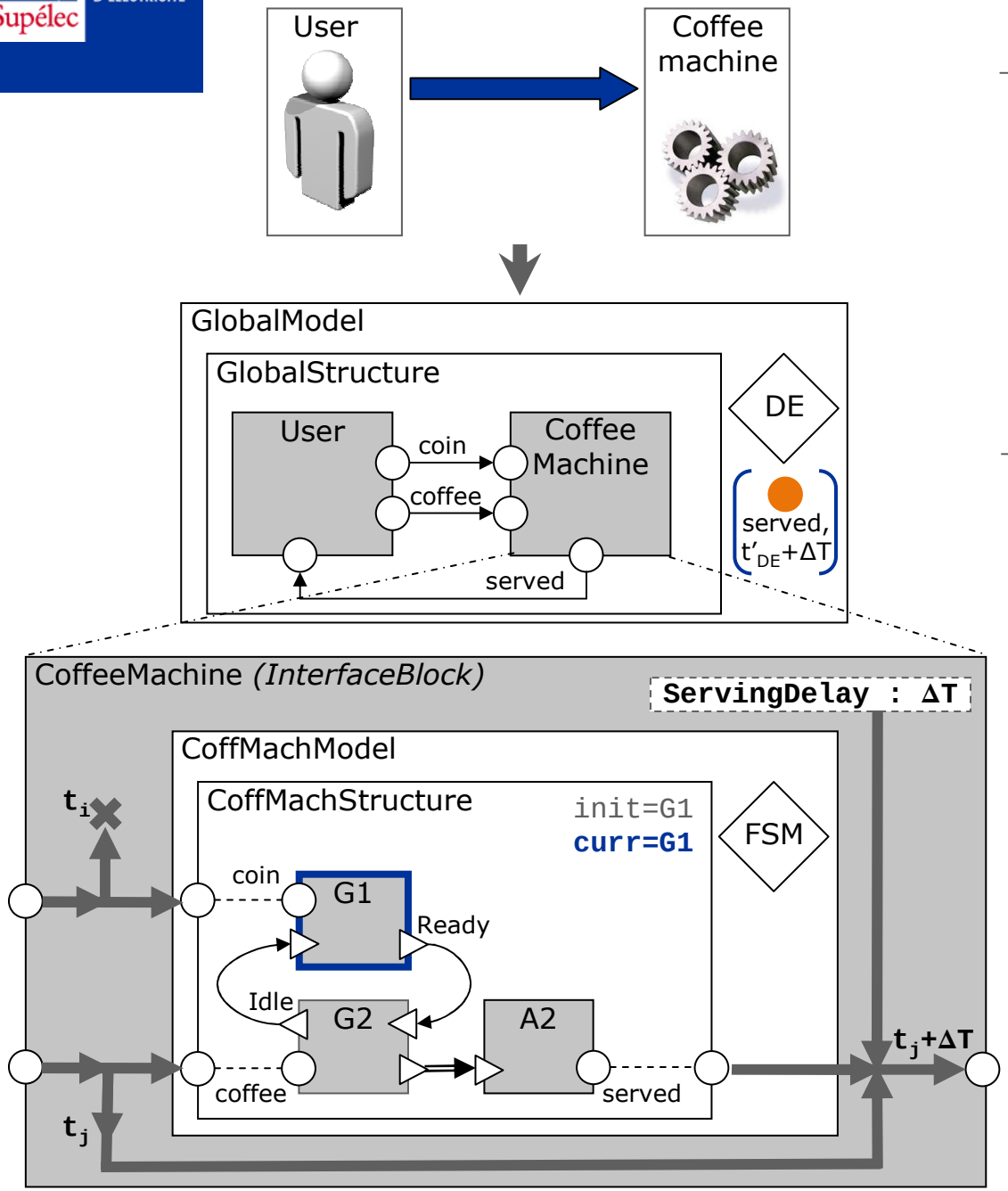
  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

$t_j + \Delta T = t'_{DE} + \Delta T$

Running the model



- Constraint on the user

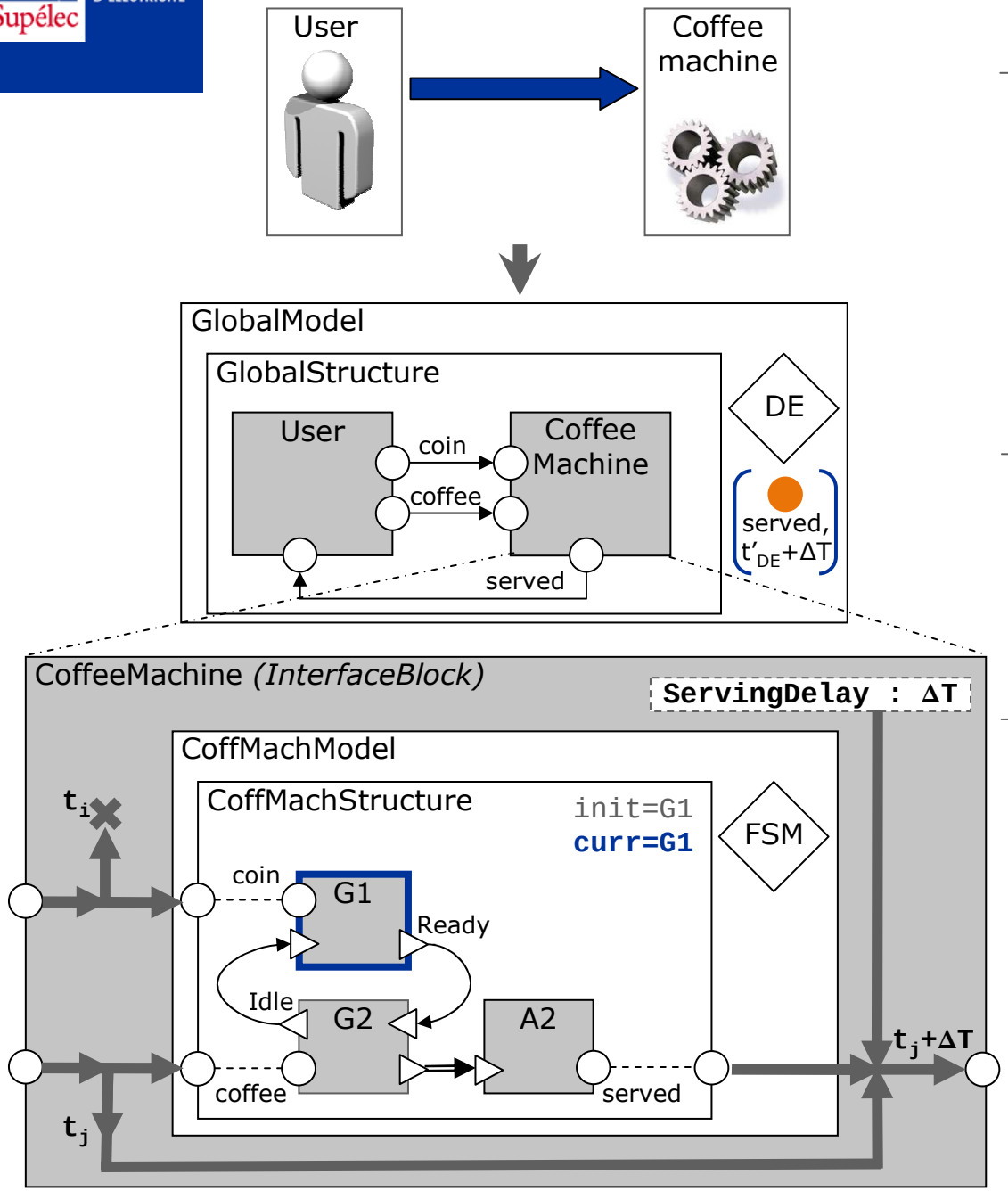
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model

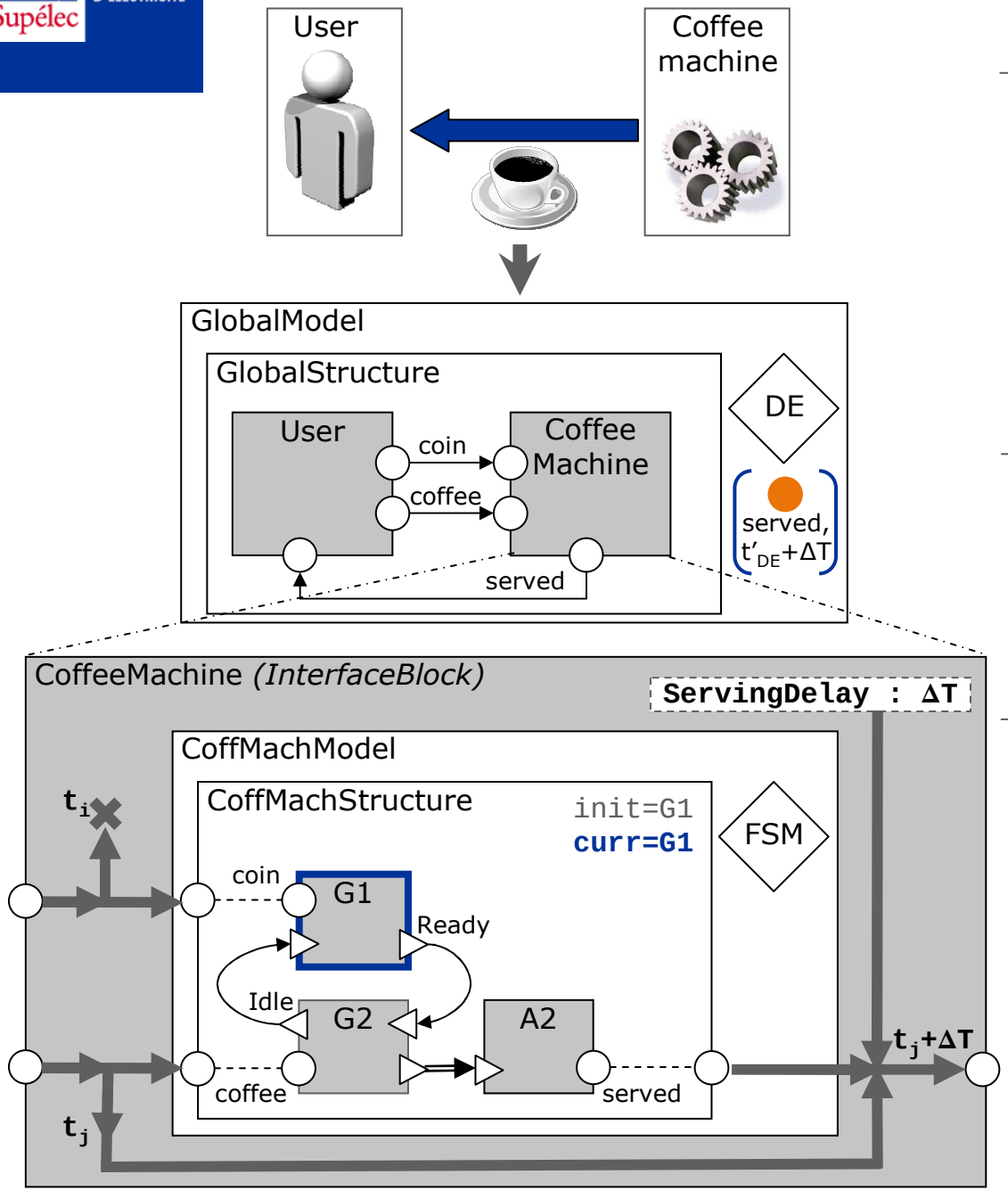


- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

Running the model

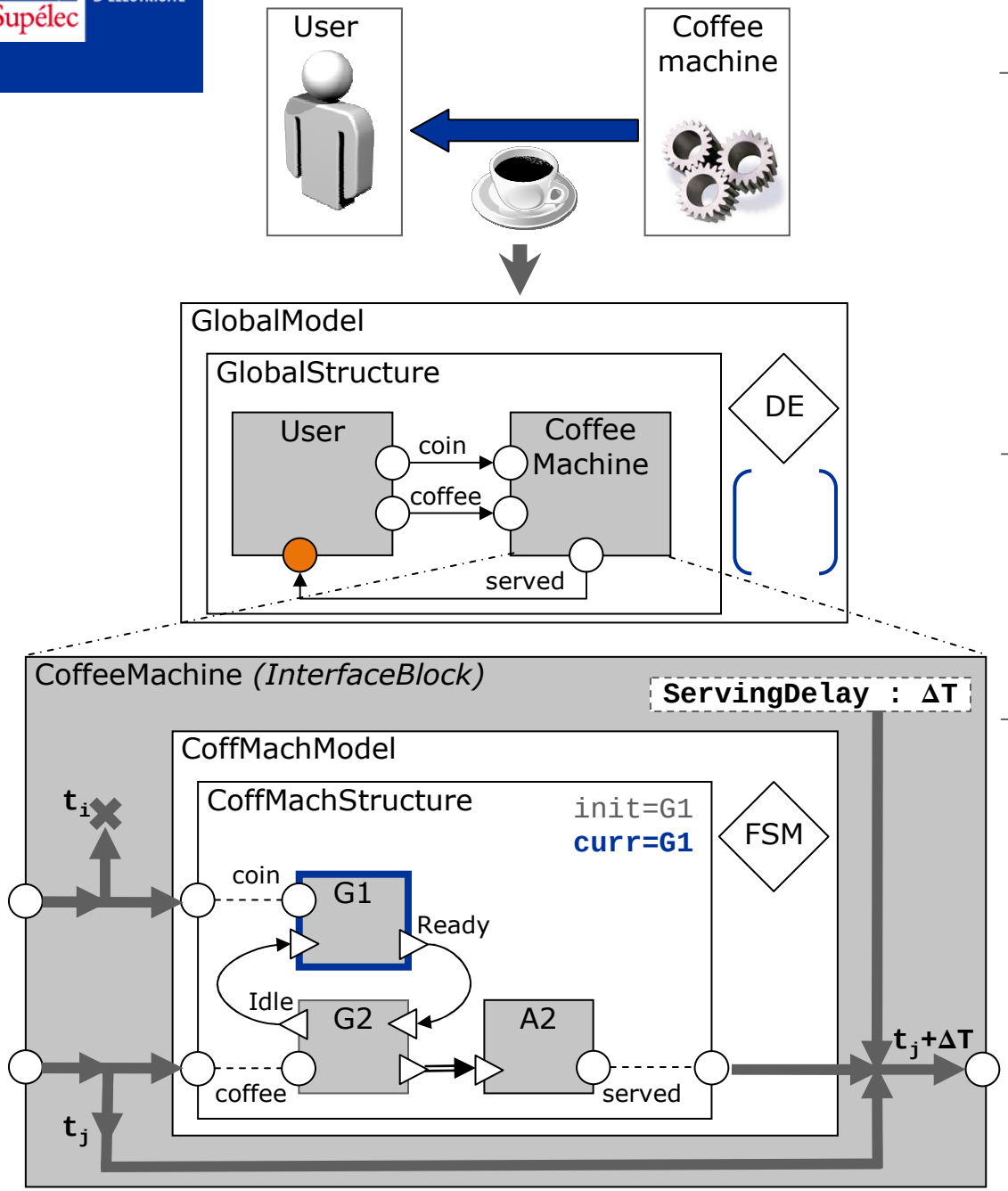


- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"
- Third snapshot
  - $t''_{DE} = t'_{DE} + \Delta T$

Running the model



- Constraint on the user

- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

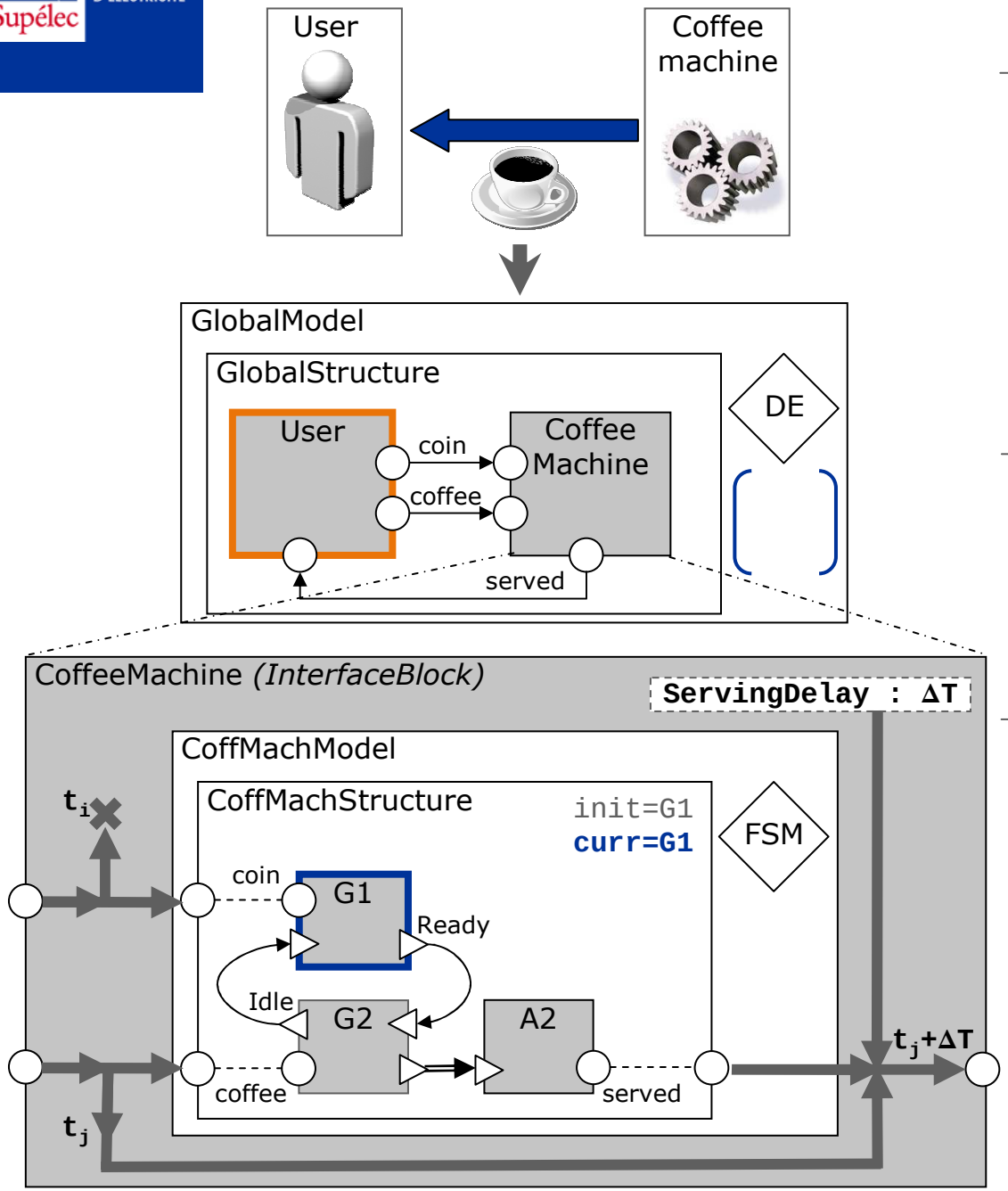
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

- Third snapshot
  - $t''_{DE} = t'_{DE} + \Delta T$

  "the machine delivers the coffee"

Running the model



- Constraint on the user

- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user

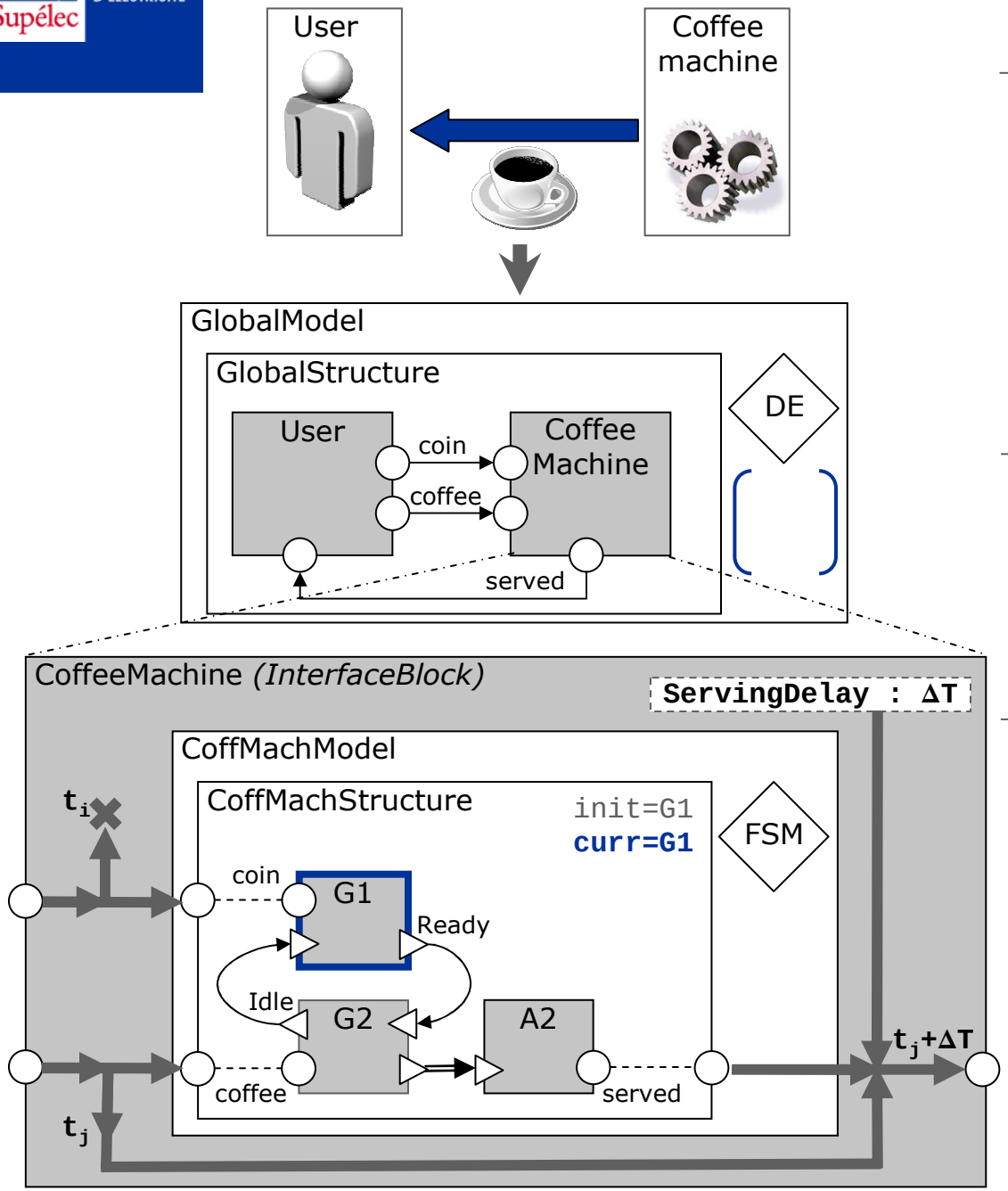
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"

- Third snapshot
  - $t''_{DE} = t'_{DE} + \Delta T$

  "the machine delivers the coffee"

Running the model



- Constraint on the user
- First snapshot
  - $t_{DE} = 0$

  "the user inserts the coin"
  - Constraint on the user
- Second snapshot
  - $t'_{DE} = T_{user}$

  "the user pushes the button"
- Third snapshot
  - $t''_{DE} = t'_{DE} + \Delta T$

  "the machine delivers the coffee"