# EMI : Une approche pour unifier l'analyse et l'exécution embarquée à l'aide d'un interpréteur de modèles pilotable

Application aux modèles UML des systèmes embarqués

**Valentin BESNARD**

École doctorale MathSTIC

Mardi 15 juin 2021 — GDR-GPL

**Rapporteurs**

Frédéric BONIOL, ONERA/DTIS

Benoît COMBEMALE, Université de Rennes 1, IRISA

**Examinateurs**

Isabelle BORNE, Université Bretagne Sud, IRISA

Julien DEANTONI, Université Côte d'Azur, I3S

Frédéric JOUAULT, ESEO

**Directeur de thèse**

Philippe DHAUSSY, ENSTA Bretagne, Lab-STICC

**Encadrants**

Matthias BRUN, ESEO

Ciprian TEODOROV, ENSTA Bretagne, Lab-STICC

**Partenaire industriel**

David OLIVIER, Davidson Consulting

# Qui suis-je ?

## Mon parcours

- Ingénieur ESEO à Angers
  - Spécialité logiciels embarqués
  - 3 fois Vice-champion de France de robotique
- Doctorat en génie logiciel

## Aujourd'hui

- Ingénieur R&D chez Ateme, l'un des leaders mondiaux dans l'édition de logiciels pour la diffusion et la compression de contenu vidéo
- Initiation d'un projet pour vérifier formellement (en live) si des flux vidéos respectent des propriétés

# Sommaire

# Sommaire

# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société

# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués

# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

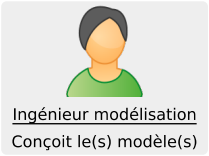
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs



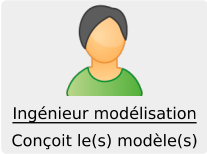
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs



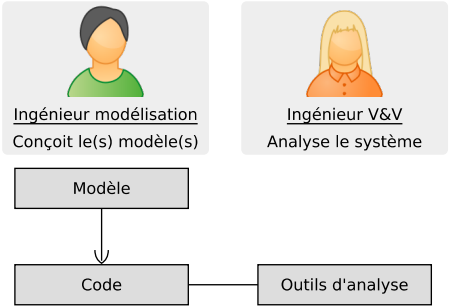
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs
- Faciliter la vérification et la validation (V&V) durant les phases de conception



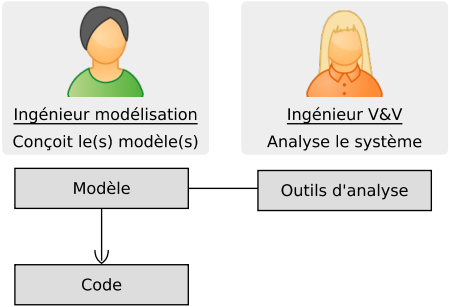
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs
- Faciliter la vérification et la validation (V&V) durant les phases de conception



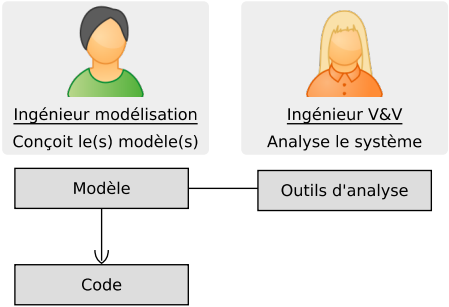
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs
- Faciliter la vérification et la validation (V&V) durant les phases de conception
- Garantir que les résultats des activités de V&V obtenus restent applicables aux systèmes opérationnels qui s'exécutent en production



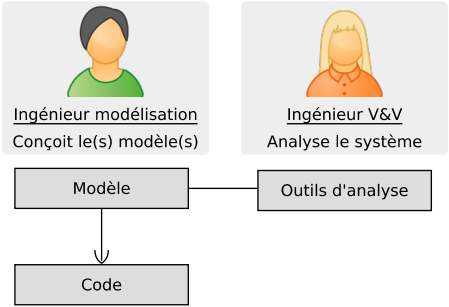
# Contexte

## Observations

- Omniprésence des systèmes embarqués dans notre société
- Complexité croissante des logiciels embarqués
- Augmentation des risques d'introduction de fautes et de la complexité d'analyse

## Besoins

- Assister la conception des logiciels embarqués par les ingénieurs
- Faciliter la vérification et la validation (V&V) durant les phases de conception
- Garantir que les résultats des activités de V&V obtenus restent applicables aux systèmes opérationnels qui s'exécutent en production
- ... pour différents langages de modélisation



# Sommaire

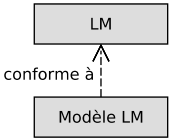
# Identification des approches d'analyse et d'exécution

Présentation des approches existantes dans l'état de l'art :

AC#1 Traduction vers modèle vérifiable et code

- Mbeddr [Voe+12]

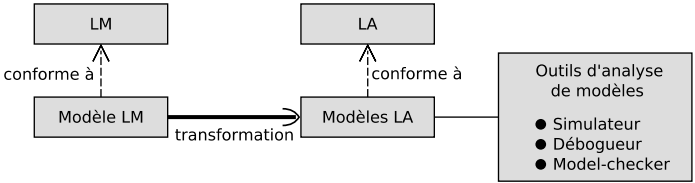
# AC#1 : Traduction vers modèle vérifiable et code




Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

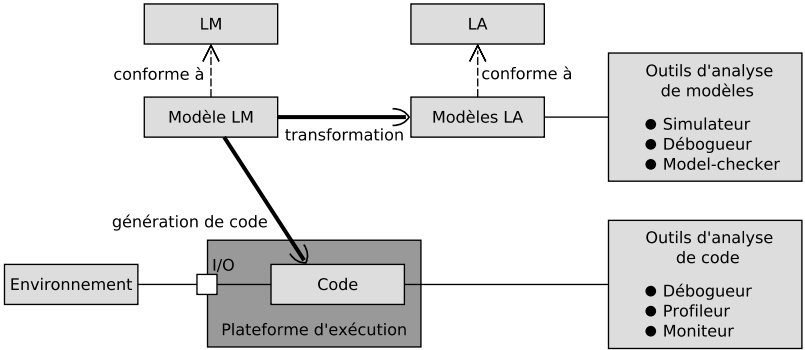
# AC#1 : Traduction vers modèle vérifiable et code




Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

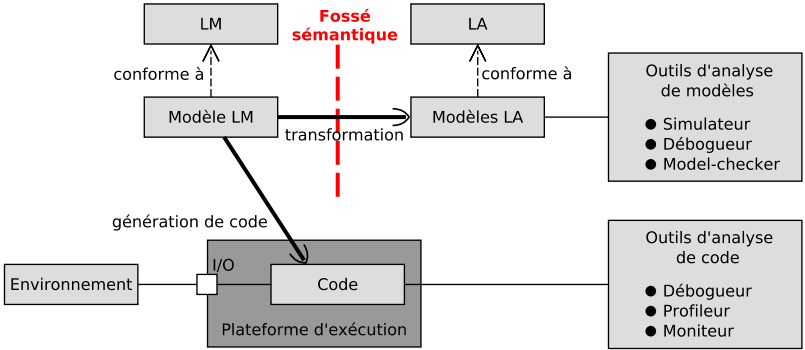
# AC#1 : Traduction vers modèle vérifiable et code




Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

# AC#1 : Traduction vers modèle vérifiable et code

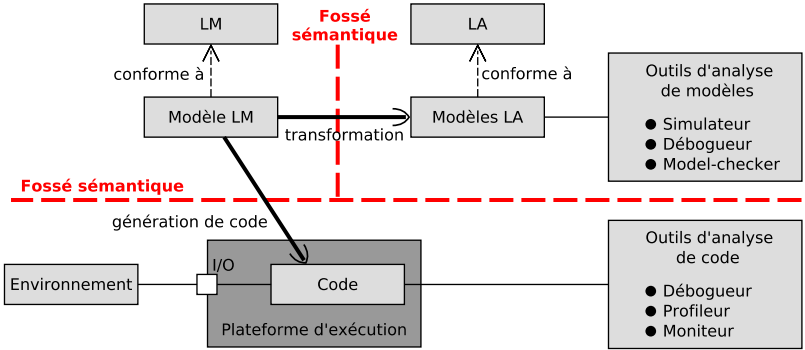



P#1 Fossé sémantique entre le modèle de conception et les modèles d'analyse

Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

# AC#1 : Traduction vers modèle vérifiable et code

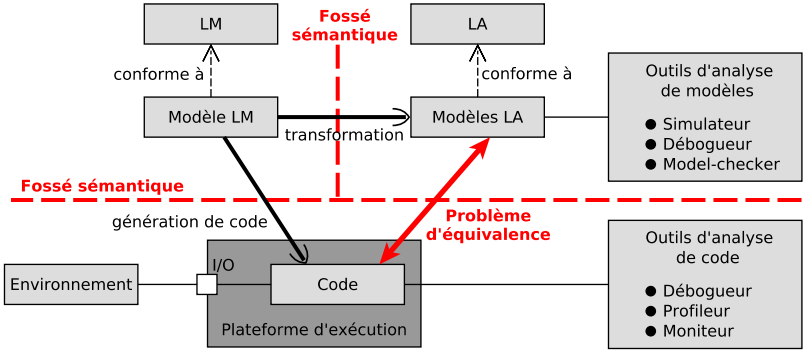



P#2 Fossé sémantique entre le modèle de conception et le code exécutable

Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

# AC#1 : Traduction vers modèle vérifiable et code




P#3 Problème d'équivalence entre les modèles d'analyse et le code exécutable

Exemple d'outil : Mbeddr [Voe+12]

[Voe+12] Voelter et al., « Mbeddr : An Extensible C-based Programming Language and IDE for Embedded Systems », 2012

# Identification des approches d'analyse et d'exécution

Présentation des approches existantes dans l'état de l'art :

AC#1 Traduction vers modèle vérifiable et code
- Mbeddr [Voe+12]

AC#2 Traduction vers modèle vérifiable puis code
- RobotChart [Miy+19]

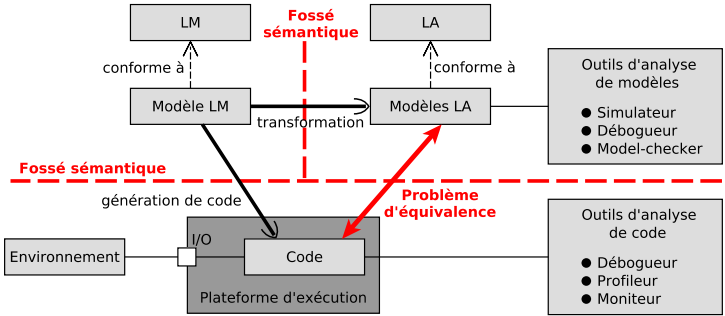
AC#3 Traduction vers code vérifiable
- Divine [Bar+17]

AC#4 Traduction modèle vérifiable vers code
- SPOT [DP04] avec générateur de code

AC#5 Raffinement de modèles jusqu'au code
- Atelier B (`https://www.atelierb.eu/`)

AC#6 Interprétations spécifiques vérification et exécution réelle
- Java PathFinder [Bra+00]

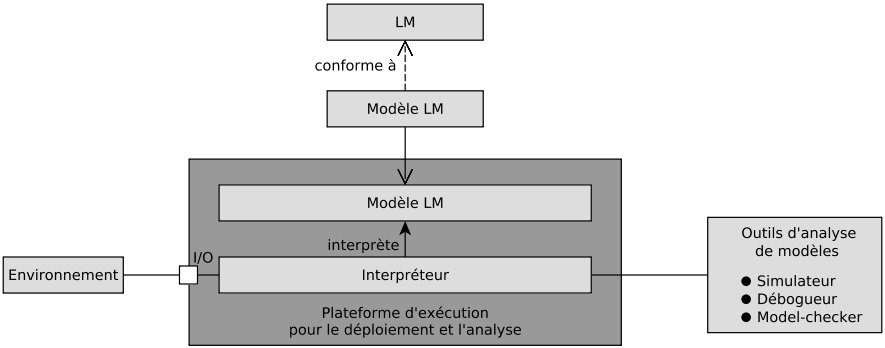
# Question de recherche




**Question de recherche : Est-il possible d'unifier l'analyse et l'exécution embarquée de modèles ?**

# Sommaire

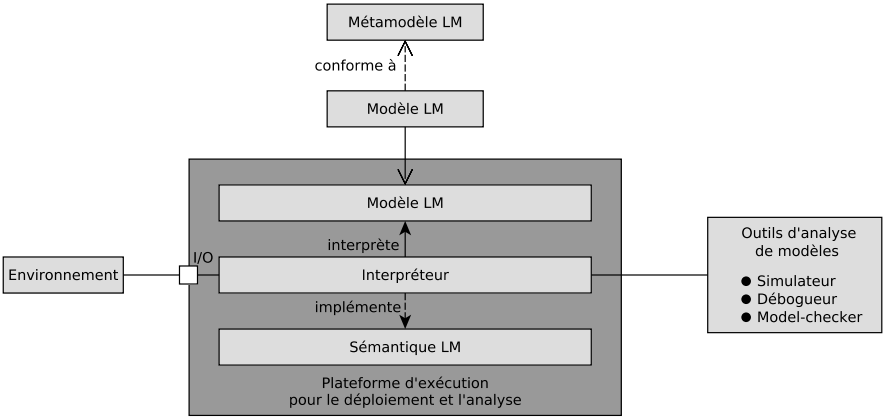
# Aperçu de l'approche EMI




- Un seul couple (modèle + sémantique) pour les activités d'analyse et l'exécution embarquée
- Un seul et même environnement pour appliquer toutes ces activités

$\Rightarrow$ **Unification des activités d'analyse et de l'exécution embarquée**

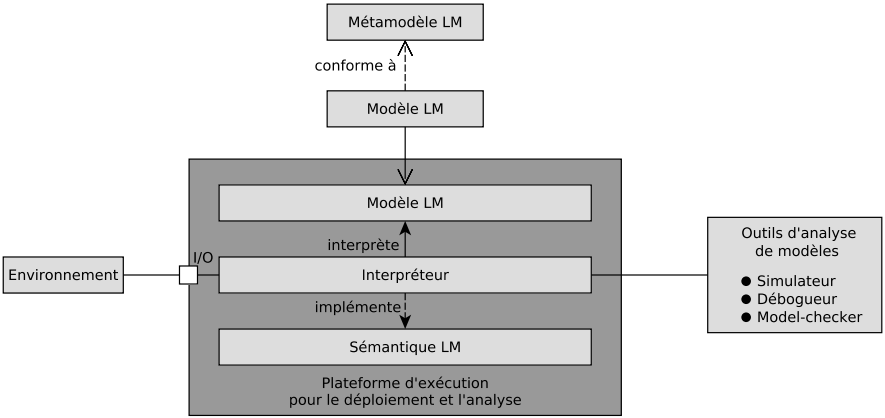
# Aperçu de l'approche EMI




- Un seul couple (modèle + sémantique) pour les activités d'analyse et l'exécution embarquée
- Un seul et même environnement pour appliquer toutes ces activités

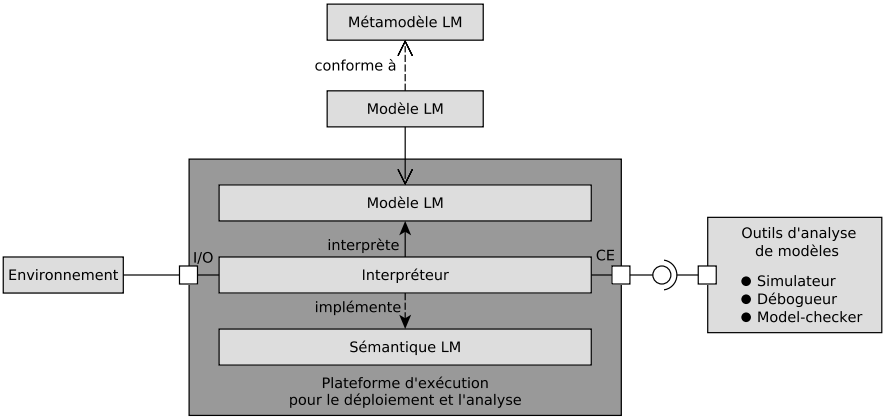
$\Rightarrow$ **Unification des activités d'analyse et de l'exécution embarquée**

# Aperçu de l'approche EMI



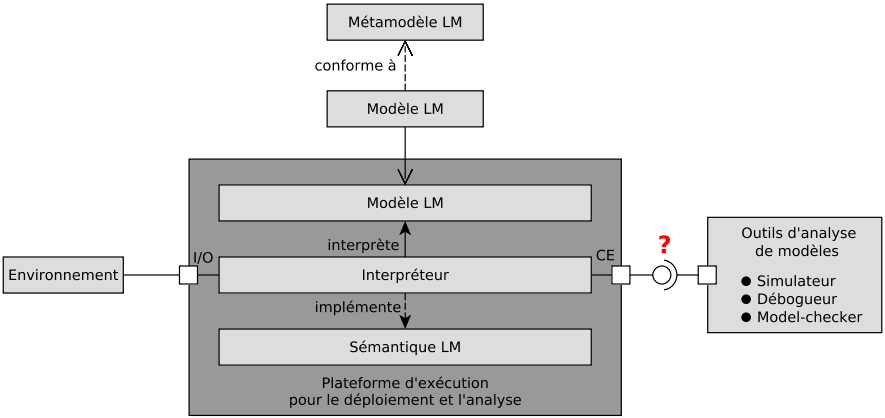

- Besoin de rendre l'interpréteur pilotable pour appliquer toutes ces activités de développement

# Aperçu de l'approche EMI



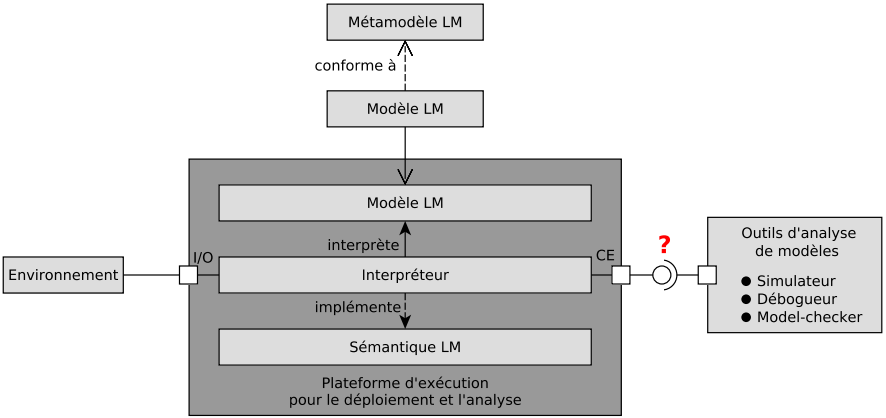

- Besoin de rendre l'interpréteur pilotable pour appliquer toutes ces activités de développement
- Besoin d'interfacer les outils d'analyse avec l'interpréteur

# Aperçu de l'approche EMI




- Besoin de rendre l'interpréteur pilotable pour appliquer toutes ces activités de développement
- Besoin d'interfacer les outils d'analyse avec l'interpréteur

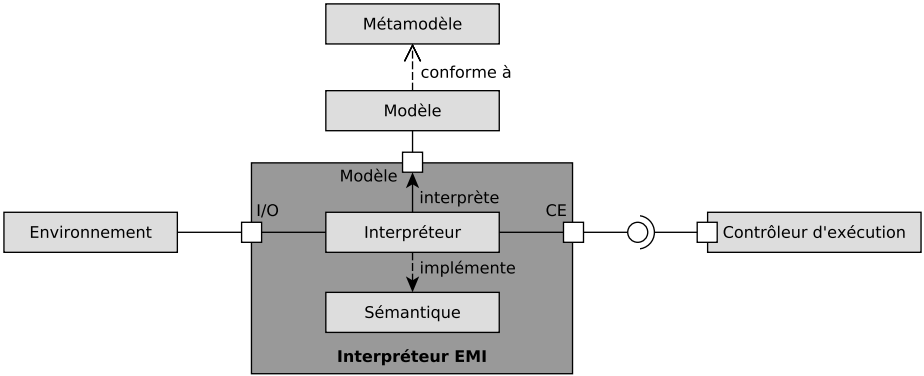
# Aperçu de l'approche EMI



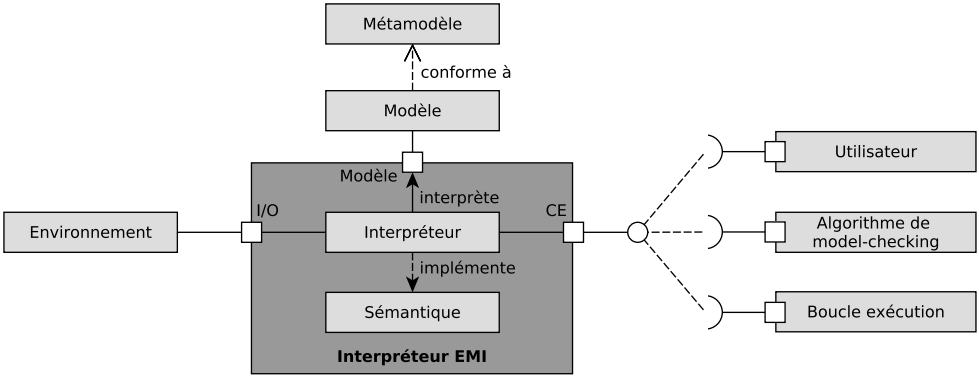

- Besoin de rendre l'interpréteur pilotable pour appliquer toutes ces activités de développement
- Besoin d'interfacer les outils d'analyse avec l'interpréteur

  $\Rightarrow$ **Définir et formaliser les interfaces minimales pour répondre à ces besoins**

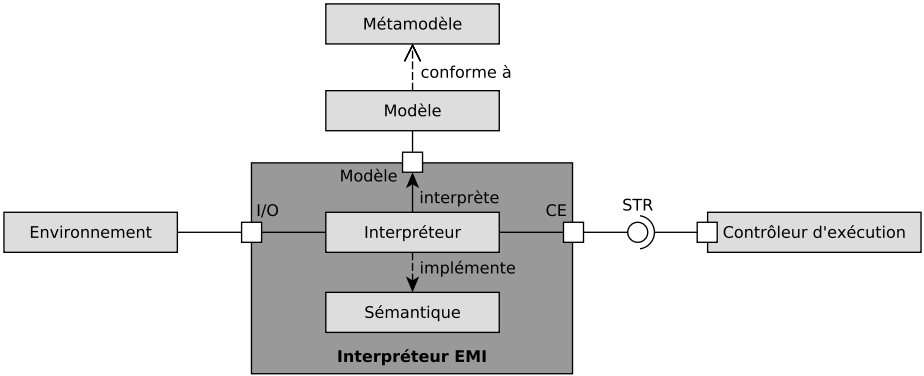
# (1) Piloter l'exécution du modèle



# (1) Piloter l'exécution du modèle



# (1) Piloter l'exécution du modèle



# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

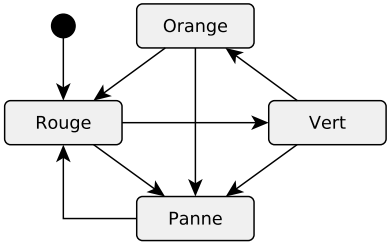
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution




[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

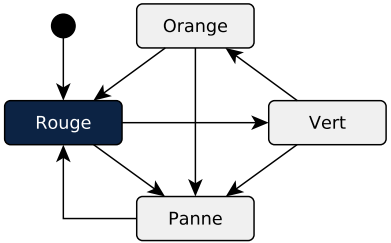
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution




1. `str.initial` $= \{\{\texttt{Rouge}\}\}$

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

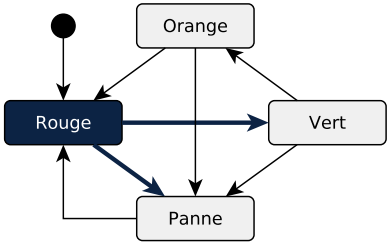
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution




1. `str.initial` $= \{\{\texttt{Rouge}\}\}$
2. `str.actions {Rouge}` $= \{\rightarrow_{\texttt{Vert}}, \rightarrow_{\texttt{Panne}}\}$

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

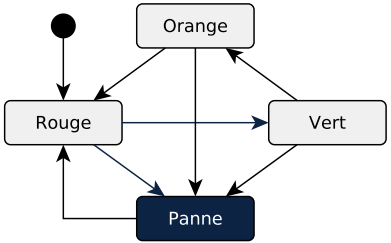
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution




1. $\texttt{str.initial} = \{\{\texttt{Rouge}\}\}$
2. $\texttt{str.actions}\ \{\texttt{Rouge}\} = \{\rightarrow_{\texttt{Vert}}, \rightarrow_{\texttt{Panne}}\}$
3. $\texttt{str.execute}\ \{\texttt{Rouge}\}\ \rightarrow_{\texttt{Panne}} = \{\{\texttt{Panne}\}\}$

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

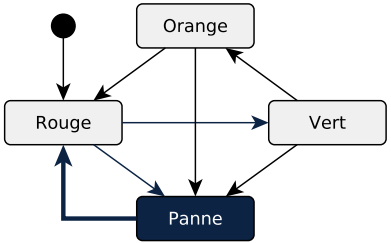
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

Représentation symbolique des pas d'exécution




1. $\texttt{str.initial} = \{\{\texttt{Rouge}\}\}$
2. $\texttt{str.actions}\ \{\texttt{Rouge}\} = \{\rightarrow_{\texttt{Vert}}, \rightarrow_{\texttt{Panne}}\}$
3. $\texttt{str.execute}\ \{\texttt{Rouge}\}\ \rightarrow_{\texttt{Panne}} = \{\{\texttt{Panne}\}\}$
4. $\texttt{str.actions}\ \{\texttt{Panne}\} = \{\rightarrow_{\texttt{Rouge}}\}$

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

# (1) Piloter l'exécution du modèle

## Interface de pilotage (STR) en Lean [Mou+15]

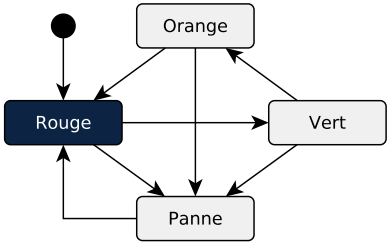
```
structure STR (C A : Type) :=
  (initial : set C)
  (actions : C → set A)
  (execute : C → A → set C)
```

## Configuration (C)

Ensemble de données d'exécution à un instant donné

## Action (A)

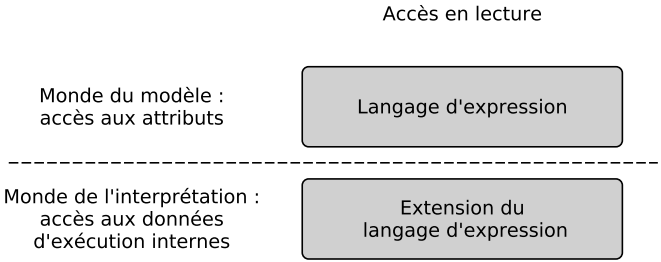
Représentation symbolique des pas d'exécution



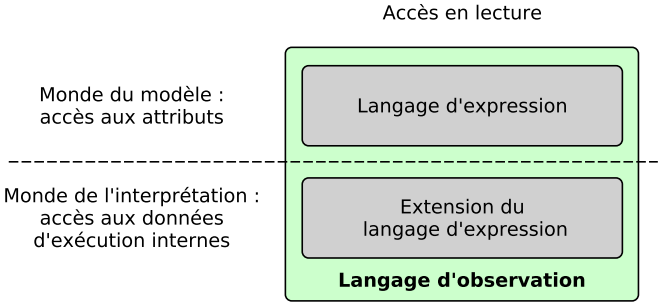

1. `str.initial` $= \{\{\texttt{Rouge}\}\}$
2. `str.actions {Rouge}` $= \{\rightarrow_{\texttt{Vert}}, \rightarrow_{\texttt{Panne}}\}$
3. `str.execute {Rouge}` $\rightarrow_{\texttt{Panne}} = \{\{\texttt{Panne}\}\}$
4. `str.actions {Panne}` $= \{\rightarrow_{\texttt{Rouge}}\}$
5. `str.execute {Panne}` $\rightarrow_{\texttt{Rouge}} = \{\{\texttt{Rouge}\}\}$

[Mou+15] Moura et al., « The Lean Theorem Prover (System Description) », 2015

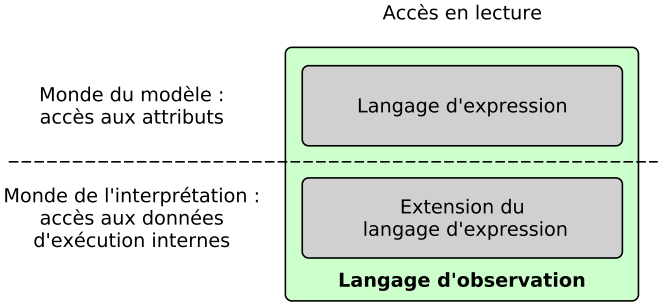
# (2) Questionner l'exécution du système



# (2) Questionner l'exécution du système



# (2) Questionner l'exécution du système



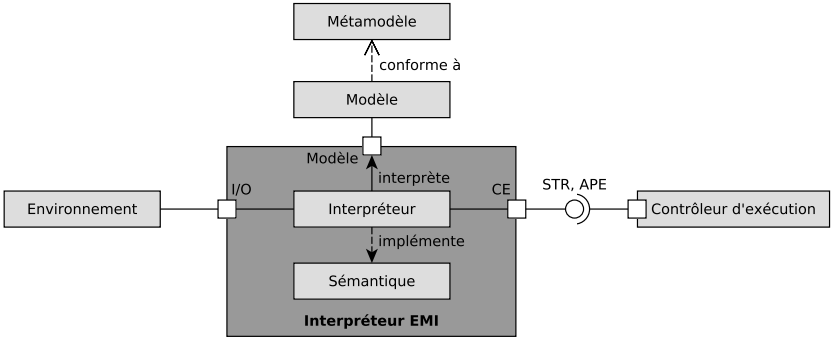

## Langage d'observation

Langage sans effet de bord **utilisant les concepts du langage de modélisation**

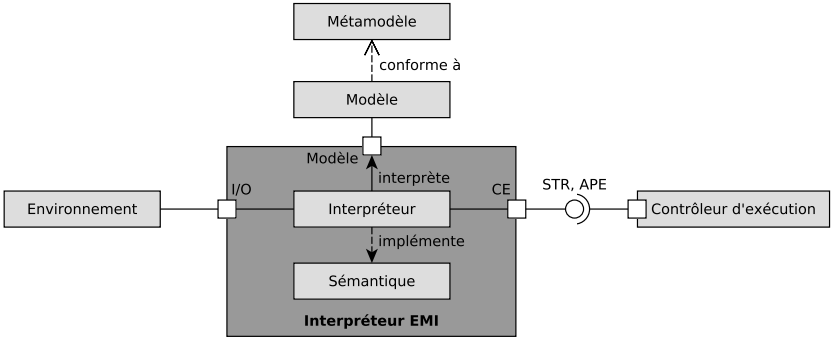
## Propositions atomiques

Prédicats permettant de questionner l'exécution du système

# (2) Questionner l'exécution du système



# (2) Questionner l'exécution du système

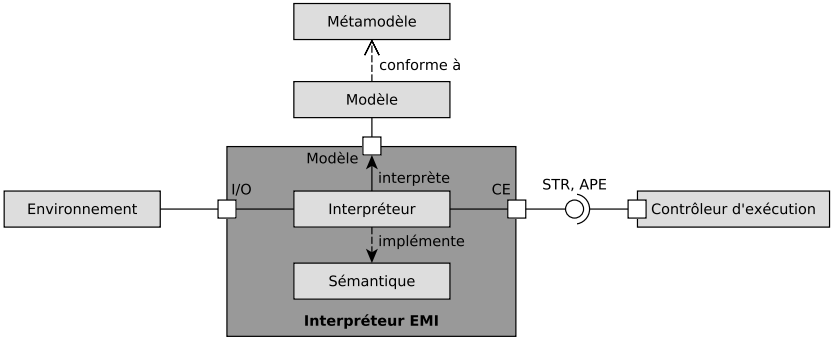



## Interface d'évaluation des propositions atomiques (APE)

```
structure APE (C A L : Type) :=
  (eval : L → C → A → C → bool)
```

Avec `L` le type des propositions atomiques

# (2) Questionner l'exécution du système




## Interface d'évaluation des propositions atomiques (APE)

```
structure APE (C A L : Type) :=
  (eval : L → C → A → C → bool)
```

Avec L le type des propositions atomiques

# Sommaire

# Expérimentations : Concevoir des interpréteurs

## EMI-UML : interpréteur embarqué de modèles UML

- Basé sur un sous-ensemble d'UML :
  - Classes
  - Structures composites
  - Machines à états

# Expérimentations : Concevoir des interpréteurs

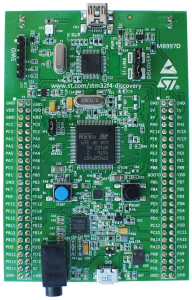
## EMI-UML : interpréteur embarqué de modèles UML

- Basé sur un sous-ensemble d'UML :
  - Classes
  - Structures composites
  - Machines à états
- Utilisation du langage C comme langage hôte

# Expérimentations : Concevoir des interpréteurs

## EMI-UML : interpréteur embarqué de modèles UML

- Basé sur un sous-ensemble d'UML :
  - Classes
  - Structures composites
  - Machines à états
- Utilisation du langage C comme langage hôte
- Déploiement sur :
  - PC de développement (Linux)
  - Cible embarquée en *bare-metal* (sans OS)



Cible embarquée STM32

[Bes+19] Besnard et al., « EMI : Un Interpréteur de Modèles Embarqué pour l'Exécution et la Vérification de Modèles UML », 2019
[Bes+17] Besnard et al., « Towards one Model Interpreter for Both Design and Deployment », 2017

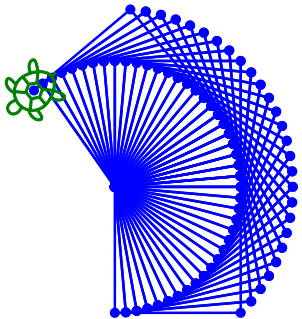
# Expérimentations : Concevoir des interpréteurs

## EMI-UML : interpréteur embarqué de modèles UML

- Basé sur un sous-ensemble d'UML :
  - Classes
  - Structures composites
  - Machines à états
- Utilisation du langage C comme langage hôte
- Déploiement sur :
  - PC de développement (Linux)
  - Cible embarquée en *bare-metal* (sans OS)

## Et d'autres interpréteurs...

- EMI-LOGO : interpréteur de modèles LOGO



[Jou+20] Jouault et al., « Designing, Animating, and Verifying Partial UML Models », 2020
[Bes+19] Besnard et al., « EMI : Un Interpréteur de Modèles Embarqué pour l'Exécution et la Vérification de Modèles UML », 2019
[Bes+17] Besnard et al., « Towards one Model Interpreter for Both Design and Deployment », 2017

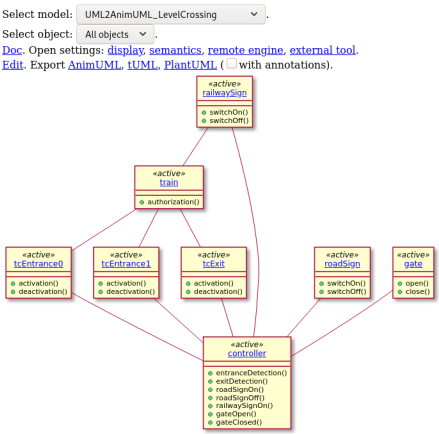
# Expérimentations : Concevoir des interpréteurs

## EMI-UML : interpréteur embarqué de modèles UML

- Basé sur un sous-ensemble d'UML :
  - Classes
  - Structures composites
  - Machines à états
- Utilisation du langage C comme langage hôte
- Déploiement sur :
  - PC de développement (Linux)
  - Cible embarquée en *bare-metal* (sans OS)

## Et d'autres interpréteurs...

- EMI-LOGO : interpréteur de modèles LOGO
- AnimUML : interpréteur Web de modèles UML [Jou+20]

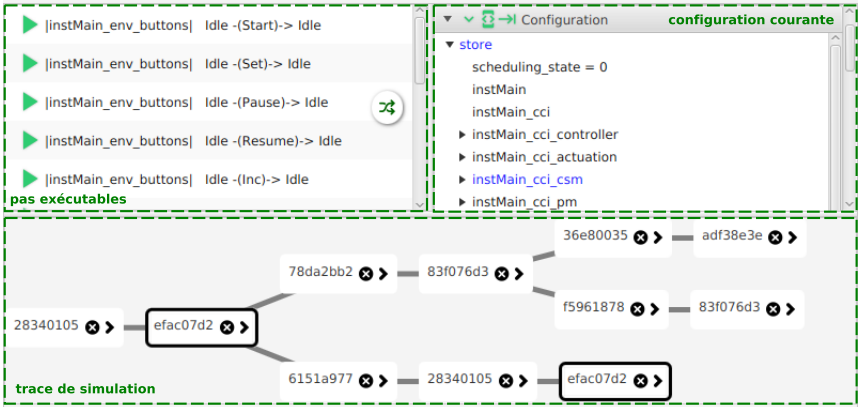



[Jou+20] Jouault et al., « Designing, Animating, and Verifying Partial UML Models », 2020
[Bes+19] Besnard et al., « EMI : Un Interpréteur de Modèles Embarqué pour l'Exécution et la Vérification de Modèles UML », 2019
[Bes+17] Besnard et al., « Towards one Model Interpreter for Both Design and Deployment », 2017

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation

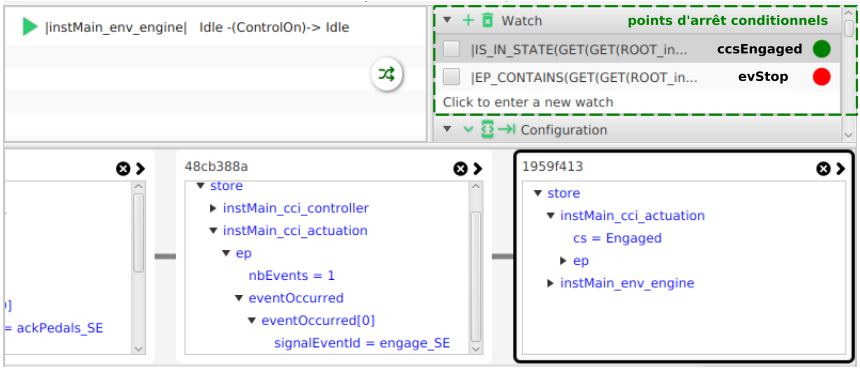



[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers

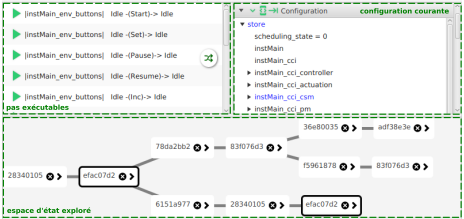



[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*

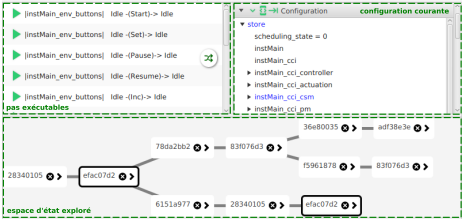



[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL

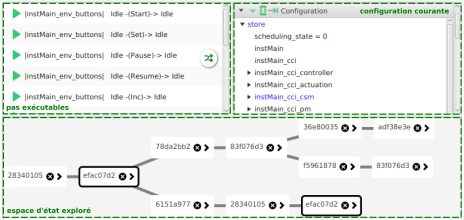



[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML



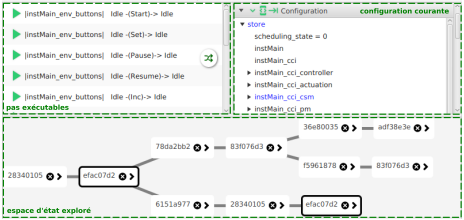

[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML

Bisimulation avec AnimUML



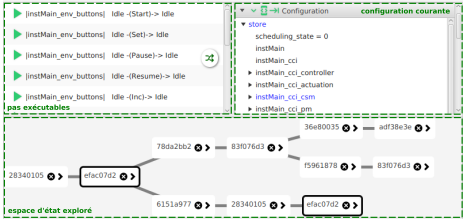

[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

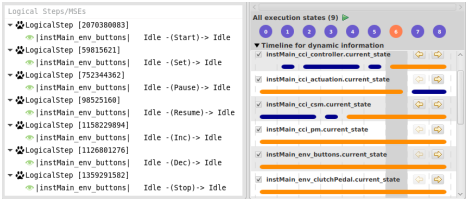
# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML

Bisimulation avec AnimUML
Débogage avec Gemoc Studio [Bou+16]



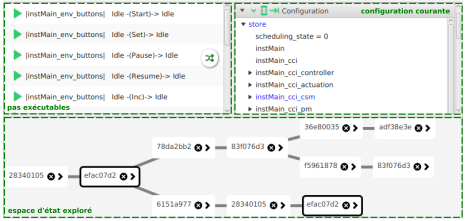

[Bou+16] Bousse et al., « Execution Framework of the GEMOC Studio (Tool Demo) », 2016
[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

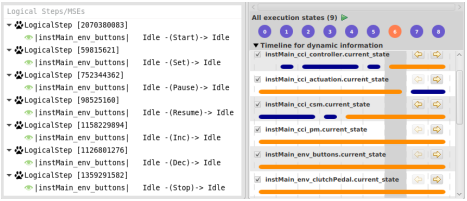
# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML

Bisimulation avec AnimUML
Débogage avec Gemoc Studio [Bou+16]
Model-checking avec l'outil Menhir [FTL20]




[FTL20] Fournier et al., « Menhir : Generic High-Speed FPGA Model-Checker », 2020
[Bou+16] Bousse et al., « Execution Framework of the GEMOC Studio (Tool Demo) », 2016
[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

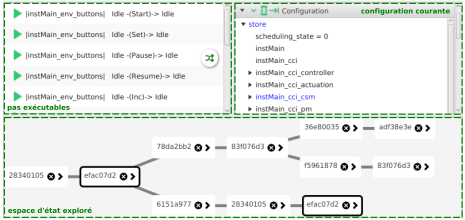
Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

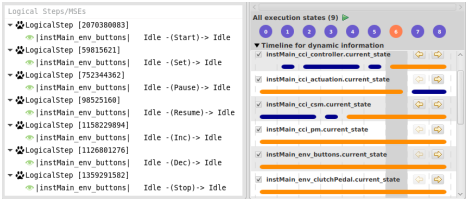
- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML

Bisimulation avec AnimUML
Débogage avec Gemoc Studio [Bou+16]
Model-checking avec l'outil Menhir [FTL20]

Activités sur la plateforme d'exécution :

- Exécution embarquée sur une cible STM32



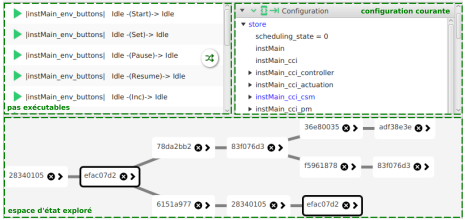




[FTL20] Fournier et al., « Menhir : Generic High-Speed FPGA Model-Checker », 2020
[Bou+16] Bousse et al., « Execution Framework of the GEMOC Studio (Tool Demo) », 2016
[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Expérimentations : Mener diverses activités de développement

Activités d'analyse avec le *model-checker* OBP2 [BGT20] :

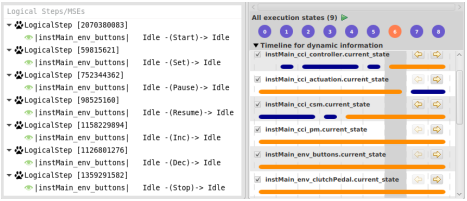
- Simulation interactive ou animation
- Débogage omniscient et multivers
- Détection de *deadlocks*
- *Model-checking* LTL
- *Model-checking* avec des automates UML




Bisimulation avec AnimUML
Débogage avec Gemoc Studio [Bou+16]
Model-checking avec l'outil Menhir [FTL20]

Activités sur la plateforme d'exécution :

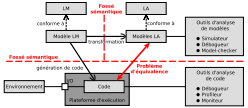
- Exécution embarquée sur une cible STM32
- *Monitoring* avec des automates observateurs

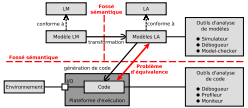


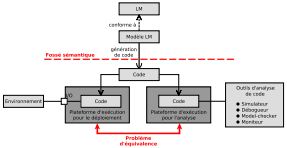

[FTL20] Fournier et al., « Menhir : Generic High-Speed FPGA Model-Checker », 2020
[Bou+16] Bousse et al., « Execution Framework of the GEMOC Studio (Tool Demo) », 2016
[BGT20] Brumbulli et al., « Automatic Verification of BPMN Models », 2020

# Sommaire

# Conclusion



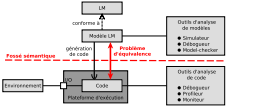



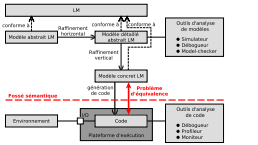



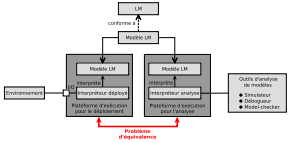

P#1 Fossé sémantique entre le modèle de conception et les modèles d'analyse

P#2 Fossé sémantique entre le modèle de conception et le code exécutable

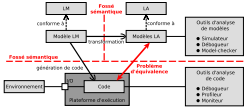
P#3 Problème d'équivalence entre les modèles d'analyse et le code exécutable

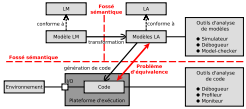


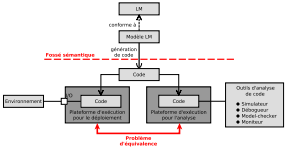



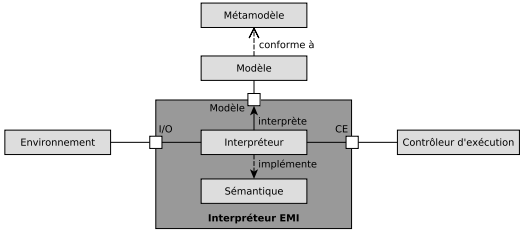



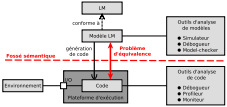
# Conclusion

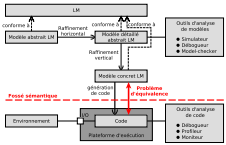

LM
LA
conforme à
Modèle LM
transformation
Modèles LA
Fossé sémantique
génération de code
Problème d'équivalence
Environnement
I/O
Code
Plateforme d'exécution
Outils d'analyse de modèles
Simulateur
Débogueur
Model-checker
Outils d'analyse de code
Débogueur
Profileur
Moniteur

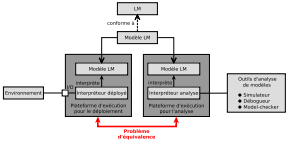

LM
LA
Fossé sémantique
conforme à
Modèle LM
transformation
Modèles LA
génération de code
Problème d'équivalence
Environnement
I/O
CE
Code
Plateforme d'exécution
Outils d'analyse de modèles
Simulateur
Débogueur
Model-checker
Outils d'analyse de code
Débogueur
Profileur
Moniteur


LM
conforme à
Modèle LM
génération de code
Fossé sémantique
Code
Environnement
I/O
Code
Plateforme d'exécution pour le déploiement
Code
Plateforme d'exécution pour l'analyse
Outils d'analyse de code
Simulateur
Débogueur
Model-checker
Moniteur
Problème d'équivalence

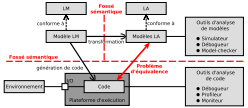

Métamodèle
conforme à
Modèle
Modèle
interprète
I/O
CE
Environnement
Interpréteur
Contrôleur d'exécution
implémente
Sémantique
Interpréteur EMI

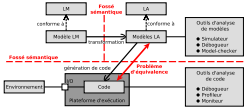

LM
conforme à
Modèle LM
génération de code
Problème d'équivalence
Fossé sémantique
Environnement
I/O
Code
Plateforme d'exécution
Outils d'analyse de modèles
Simulateur
Débogueur
Model-checker
Outils d'analyse de code
Débogueur
Profileur
Moniteur

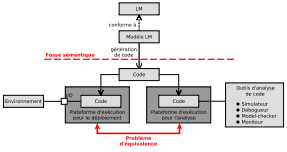

LM
conforme à
Modèle abstrait LM
Raffinement horizontal
Modèle abstrait LM
conforme à
Modèle concret LM
Raffinement vertical
Modèle concret LM
génération de code
Problème d'équivalence
Fossé sémantique
Environnement
I/O
Code
Plateforme d'exécution
Outils d'analyse de modèles
Simulateur
Débogueur
Model-checker
Outils d'analyse de code
Débogueur
Profileur
Moniteur

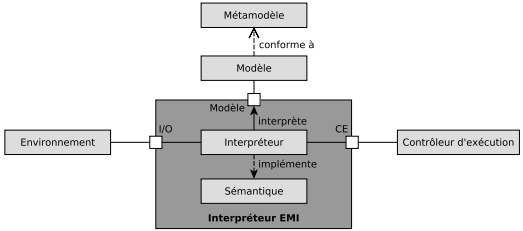

LM
conforme à
Modèle LM
Modèle LM
interprète
Interpréteur déploy.
Modèle LM
interprète
Interpréteur analyse
Environnement
I/O
Plateforme d'exécution pour le déploiement
Plateforme d'exécution pour l'analyse
Outils d'analyse de modèles
Simulateur
Débogueur
Model-checker
Problème d'équivalence

# Conclusion



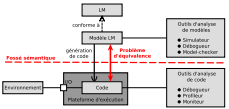



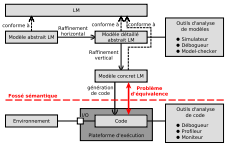



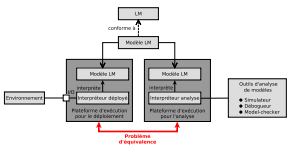

## Contributions

1. Unifier l'analyse et l'exécution embarquée de modèles
2. Faciliter l'adoption des techniques de vérification formelle par les ingénieurs

... pour différents langages de modélisation logicielle ($\rightarrow$ ingénieur langage)








# EMI : Une approche pour unifier l'analyse et l'exécution embarquée à l'aide d'un interpréteur de modèles pilotable

Application aux modèles UML des systèmes embarqués

**Valentin BESNARD**

École doctorale MathSTIC

Mardi 15 juin 2021 — GDR-GPL

**Rapporteurs**
Frédéric BONIOL, ONERA/DTIS
Benoît COMBEMALE, Université de Rennes 1, IRISA
**Examinateurs**
Isabelle BORNE, Université Bretagne Sud, IRISA
Julien DEANTONI, Université Côte d'Azur, I3S
Frédéric JOUAULT, ESEO

**Directeur de thèse**
Philippe DHAUSSY, ENSTA Bretagne, Lab-STICC
**Encadrants**
Matthias BRUN, ESEO
Ciprian TEODOROV, ENSTA Bretagne, Lab-STICC
**Partenaire industriel**
David OLIVIER, Davidson Consulting