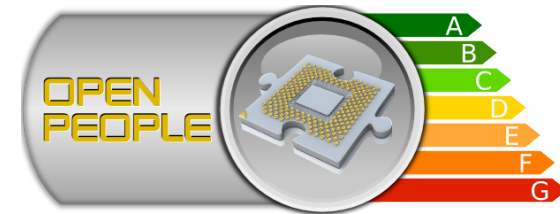


Open-PEOPLE
Open Power and Energy Optimization
Platform and Estimator



Utilitaires pour transformations ATL

Dominique BLOUIN

Lab-STICC, Université de Bretagne Sud, Lorient, FRANCE

Journée transformations de modèles et outillage de l'IDM

7 novembre 2011

- Introduction.
- Model Handler ATL pour AADL.
- Composition de transformations et héritage.
- Conclusion et perspectives.

Les transformations de modèles au Lab-STICC - UBS:

- AADL → SystemC (Open-PEOPLE).
- AADL → PADL (Power Analysis Description Language, outil CAT).
- MARTE vers AADL (SPICES).
- DSL de conception de code automates programmables → analyses et génération de code.
- DSL de conception de contrôle-commande domotique → génération de code de configuration.
- MARTE → FPGA (projet MOPCOM et FAMOUS).

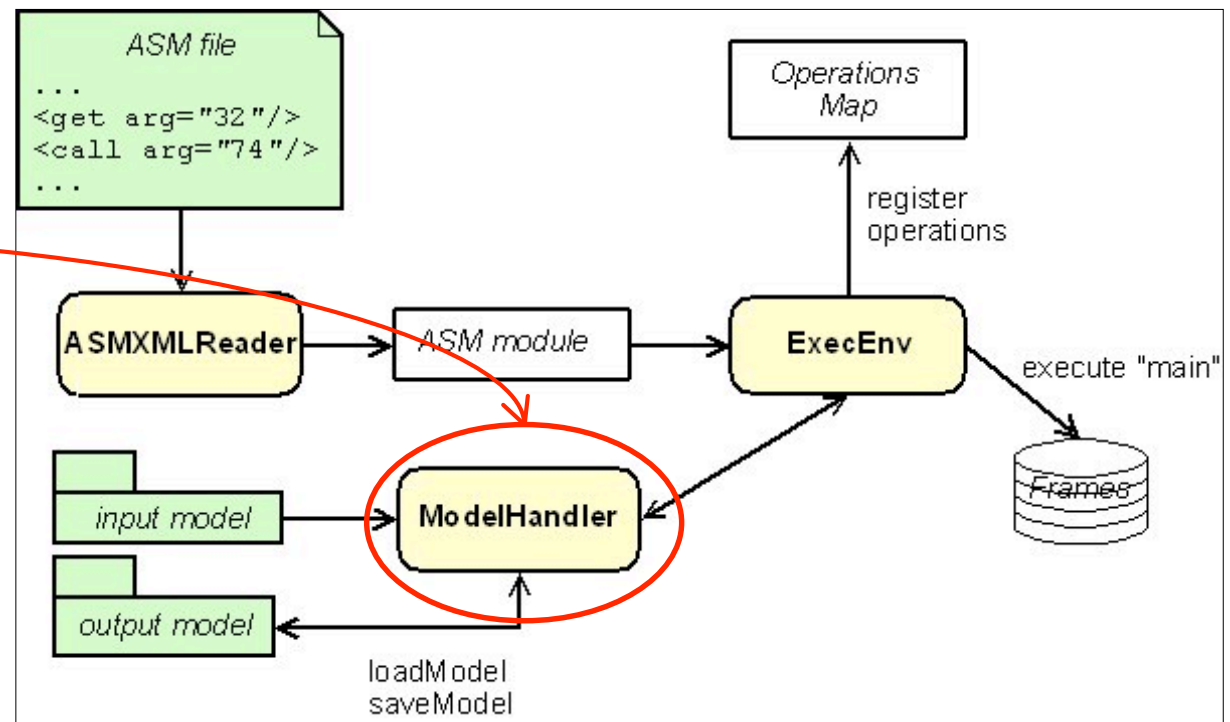
- ATL → Atlas Transformation Language.
- Bonne maturité (version 3), passé du labo à l'entreprise.
- Open source.
- Architecture proche de QVT (règles déclaratives).
- Code compilé et interprété par une machine virtuelle (VM).
- Adaptable (2 VM fournies).

■ Regular VM

- Première VM fournie.
- Manipule une abstraction de modèle.
- Modèle handlers pour EMF, UML, MDR

■ EMF-specific VM

- Manipule les modèles EMF directement.
- Meilleures performances.
- Pas adaptable.



MODEL HANDLER POUR AADL

- AADL → Architecture Analysis and Design Language.
- Description d'architecture dédié aux systèmes embarqués (critiques, temps réel).
- Issu de l'avionique, standardisé par la SAE (Society of Automotive Engineers).
- Composants (10 catégories prédéfinies) avec connecteurs.
- Deux types de modèles:
 - Déclaratif (définition de types de composants).
 - Instanciés (utilisation des types prédéfinis).
- Implémenté dans un outil open source intégré à Eclipse (OSATE).
 - Développé au SEI.

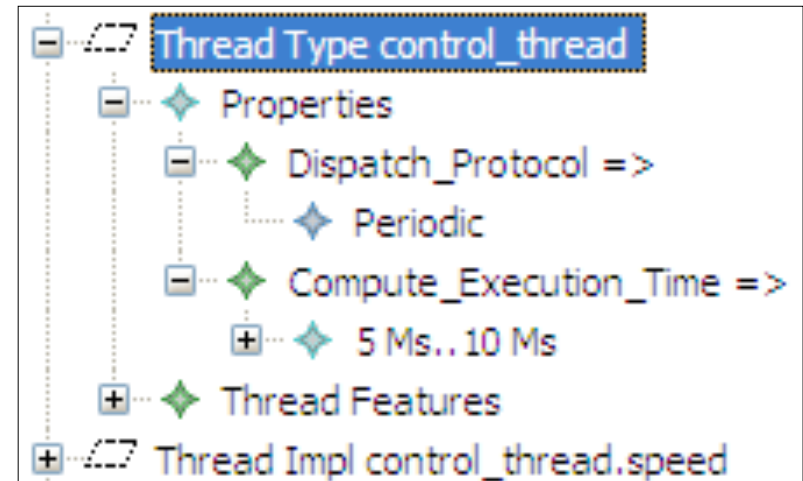
MODEL HANDLER POUR AADL

Recherche des propriétés:

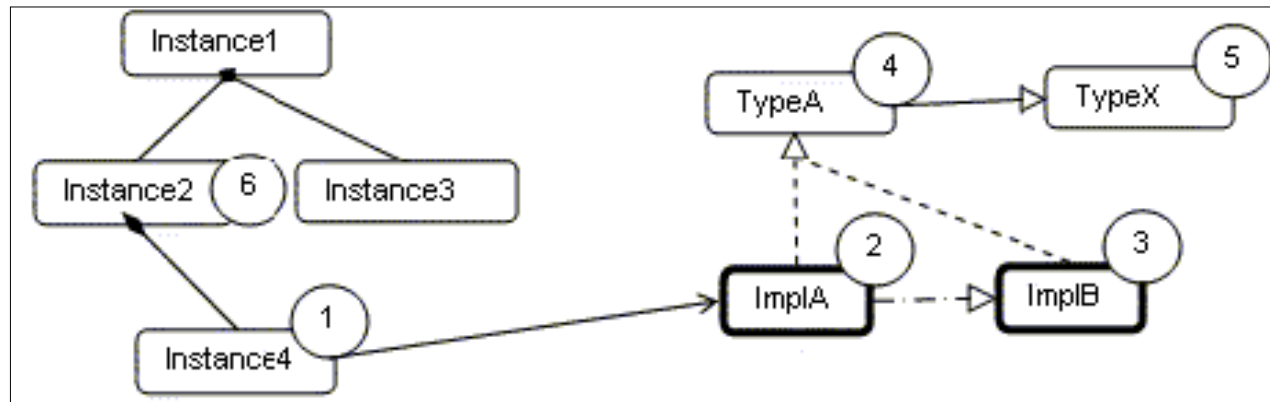
- Structure des propriétés complexe (M1).
- Algorithme de recherche complexe.
- Fournir une bibliothèque de helper réutilisables pour les propriétés AADL prédéfinies.
- Utiliser la méthode prédéfinie du méta-modèle.

- Problème: *PropertyNotFoundException*

Structures des propriétés AADL

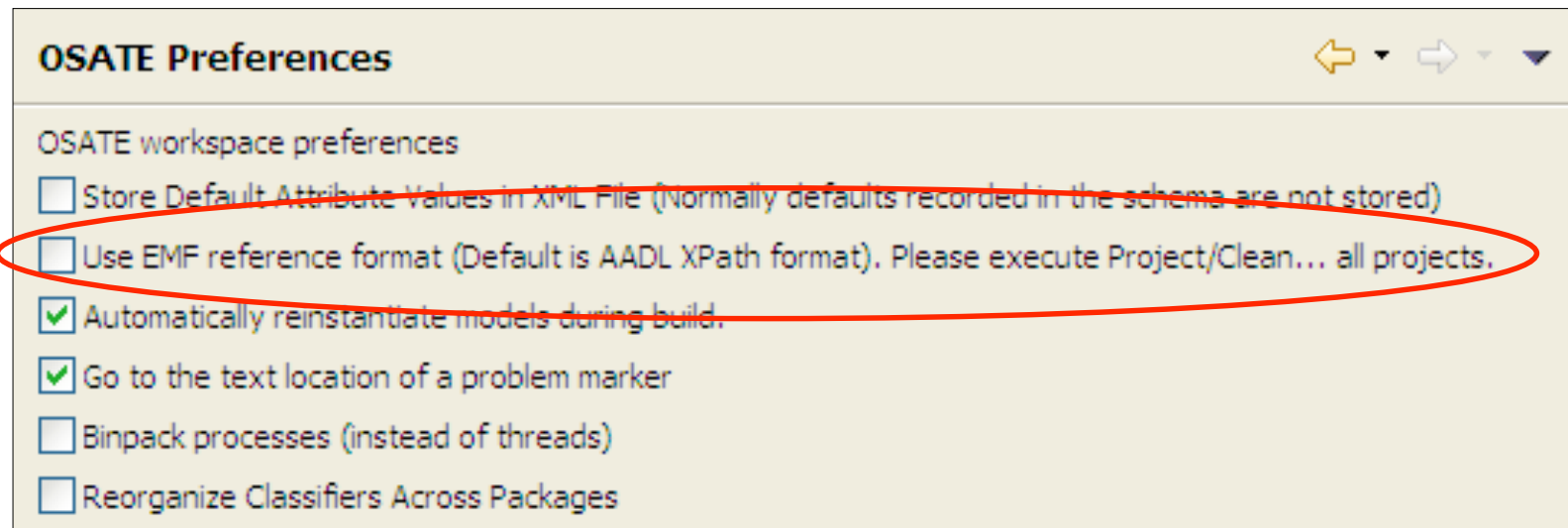


Algorithme de recherche des propriétés AADL



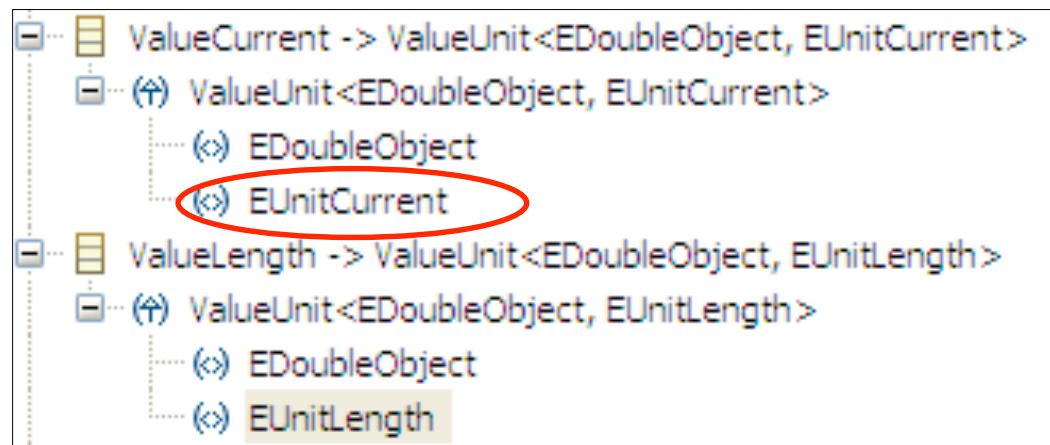
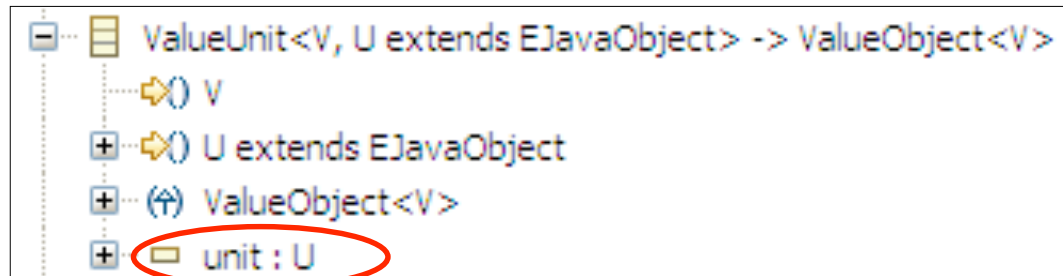
Gestion des ressources dans OSATE:

- Toutes les ressources AADL sont gérées par le OsateResourceManager.
- Visibilité dans tout l'espace de travail (workspace) Eclipse.
 - Pour les propriétés, « allInstances » doit ramener toutes les définitions de propriétés visibles.
- Par défaut, les références stockées sous un format particulier (xpath).



Gestion des génériques Ecore:

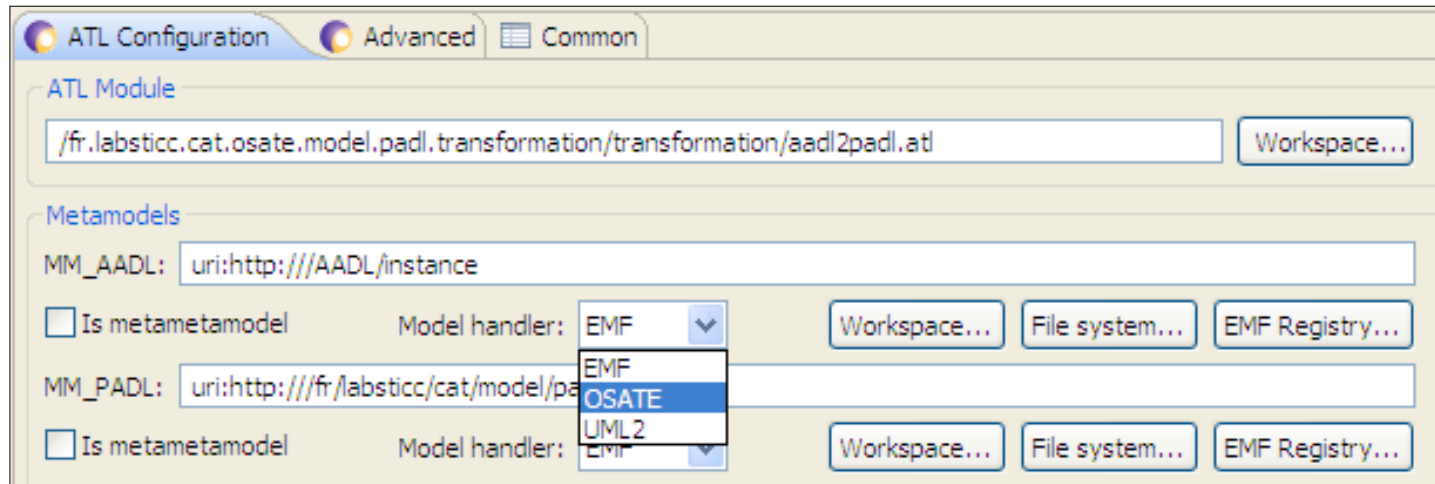
- Problème de détermination des types génériques du modèle de sortie:



- *ClassCastException...*

MODEL HANDLER POUR AADL

Solution: développer un model handler dédié à AADL/OSATE.



■ Redéfinition de classes pour:

- Gérer le *PropertyNotFoundException*.
- Déléguer la gestion des ressources AADL à OSATE.
- Corriger le problème des génériques.
- Visibilité des éléments de méta-modél répartis sur plusieurs fichiers.
 - Déjà corrigé pour la EMF-specific VM.

The screenshot shows the Eclipse IDE console with the following error message:

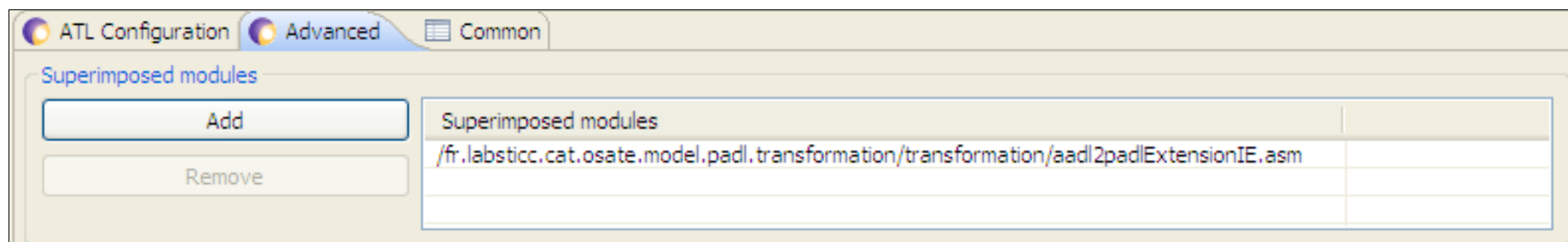
```
ATL
Found AADL properties:: OrderedSet {}
org.eclipse.m2m.atl.engine.vm.VMException: Cannot find metamodel element DeviceImpl in model MM_AADL
    at A.__matchDeviceImpl() : ??#2(testAllInstances.atl)
        local variables = {self=testAllInstances : ASModule}
        local stack = []
```

COMPOSITION DE TRANSFORMATIONS ET HERITAGE

Factorisation / composition des transformations:

- CAT: outil d'estimation de la consommation **extensible**.
 - Utilise son propre méta-modèle de description d'architecture orientée conso.
 - Intégré à OSATE; utilise une transformation ATL de base.
 - Une extension de modèle de conso pour CAT contient (entre autres):
 - Extension de méta-modèle.
 - **Extension de la transformation de modèle de base pour les nouveaux éléments.**

- ATL fournit un mécanisme de composition appelé superposition de modules:



COMPOSITION DE TRANSFORMATIONS ET HERITAGE

- Malheureusement, l'héritage des règles n'est pas géré...

```
rule ComponentBus extends ComponentHardware {  
  from  
    s : MM_AADL!ComponentInstance( s.isBus() )  
  to  
    t : MM_PADL!Bus(  
      dataProfile <- s.dataProfile(),  
      dataSource <- s.dataSource(),  
      length <- s.busLength(),  
      width <- s.busWidth()  
    )  
}
```

- Solution: Transformation de second ordre (HOT).
 - Un module de transformation ATL est un modèle EMF conforme au méta-modèle définissant le langage ATL.
 - Peut être transformé avec ATL.

Annotation définissant la règle héritée

```
-- Transforms an AADL asic bus component into a PADL asic bus.  
-- @extends ComponentBus  
rule ComponentBusASIC { -- extends ComponentBus {  
  from  
    s : MM_AADL!ComponentInstance( s.isBusASIC() )  
  to  
    t : MM_PADL!BusASIC(  
      nbMetalLayers <- s.nbMetalLayers(),  
      buffered <- s.isBufferized()  
    )  
}
```

COMPOSITION DE TRANSFORMATIONS ET HERITAGE

- Création d'une transformation de copie ATL pour le méta-modèle ATL

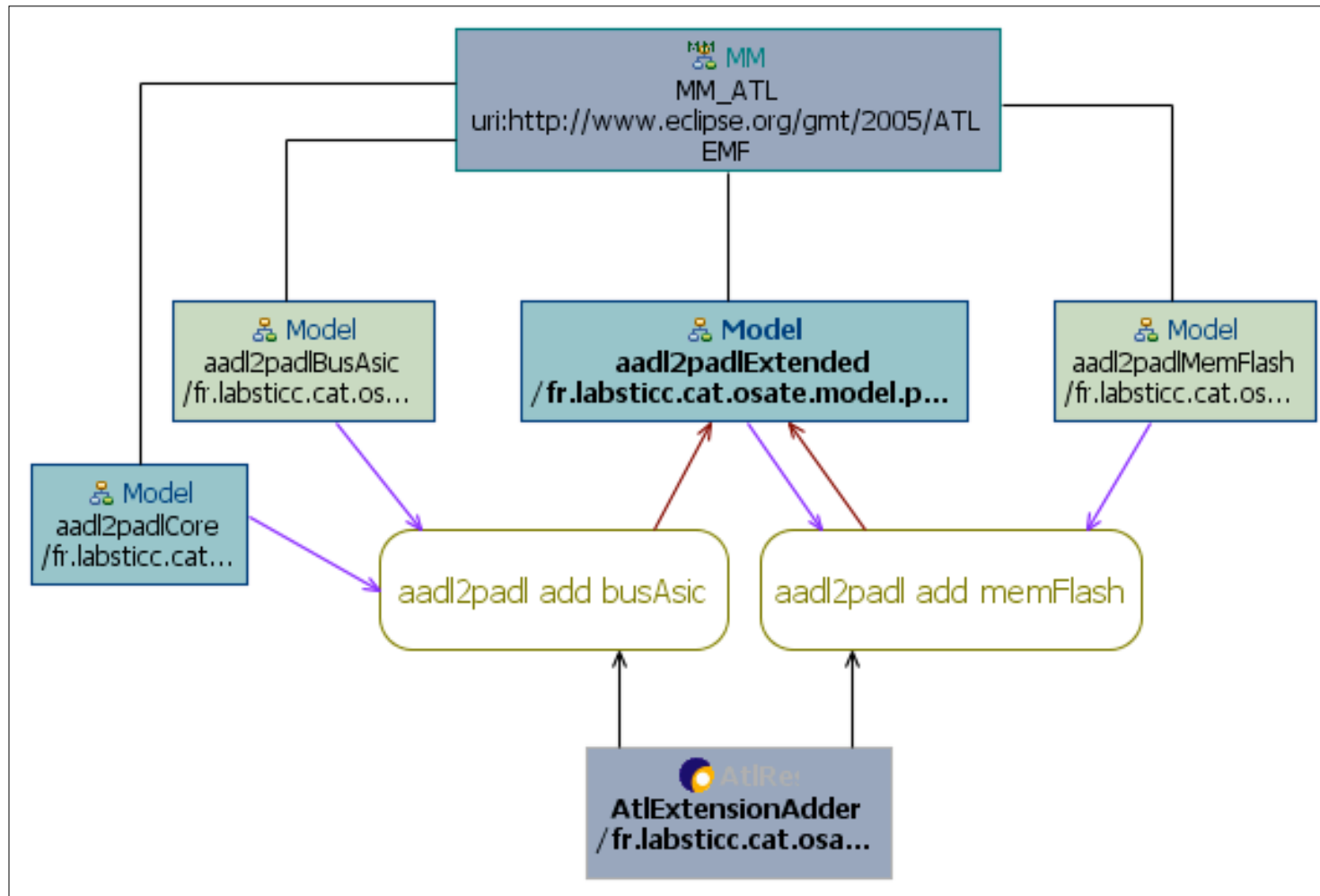
```
rule MatchedRule {
  from s : MM_ATL!"ATL::MatchedRule" (
    -- if thisModule.inElements->includes(s) then
      s.oclIsTypeOf(MM_ATL!"ATL::MatchedRule"))
    --else false endif)
  to t : MM_ATL!"ATL::MatchedRule" (
    "module" <- thisModule.MASTER_MODULE, -- Changed
    superRule <- s.superRule,
    superRule <- s.superRule(), -- Changed
    location <- s.location(),
    commentsBefore <- s.commentsBefore,
```

```
-- Transforms an AADL asi bus component into a PADL asic bus.
-- @extends ComponentBus
rule ComponentBusASIC { -- extends ComponentBus {
  from
    s : MM_AADL!ComponentInstance( s.isBusASIC() )
  to
    t : MM_PADL!BusASIC(
      nbMetalLayers <- s.nbMetalLayers(),
      buffered <- s.isBufferized()
    )
}
```

COMPOSITION DE TRANSFORMATIONS ET HERITAGE

- Représentation en diagramme ATL Flow.

- <http://opensource.urszeidler.de/ATLflow/>



- Implémentation d'une classe de pré-compilation d'extensions ATL.

Conclusion:

- ATL est un bon outil pour les transformations.
- L'aspect déclaratif du langage permet d'écrire des transformations de manière productive.
- Extensible pour l'adapter aux besoins spécifiques.
 - Par contre il faut savoir bricoler...
- Mode raffinement amélioré avec le compilateur 2010 (pas de copie).
 - Certains concepts tels que les lazy rules, les éléments d'entrée multiples, ...etc. ne sont pas gérés.
- Reste des soucis de performances (passage à l'échelle) pour les très gros modèles.
 - Une solution pourrait être l'implémentation d'un compilateur ciblant du bytecode java.

Perspectives:

■ Model handler pour AADL:

- AADL V2 publié en janvier 2009.
- OSATE 2 en cours de développement.
 - Snapshot disponible sur <http://gforge.enseeiht.fr/projects/osate2/>
- Complètement reconstruit avec le framework Xtext.
- Modèle handler à mettre à jour pour prendre en compte le nouveau mode de gestion des ressources, visibilité (clause *with*), etc.

■ Composition des transformations:

- Nouvelle VM en développement (EMFTVF).
 - <http://soft.vub.ac.be/soft/research/mdd/emftvm>
- Composition implémentée par la VM devient indépendante du langage de transformation.
- Suffit de développer un compilateur pour chaque langage.

■ AADL:

- www.aadl.info
- <https://wiki.sei.cmu.edu/aadl/>

■ ATL:

- <http://eclipse.org/atl/>
- http://wiki.eclipse.org/ATL/Developer_Guide