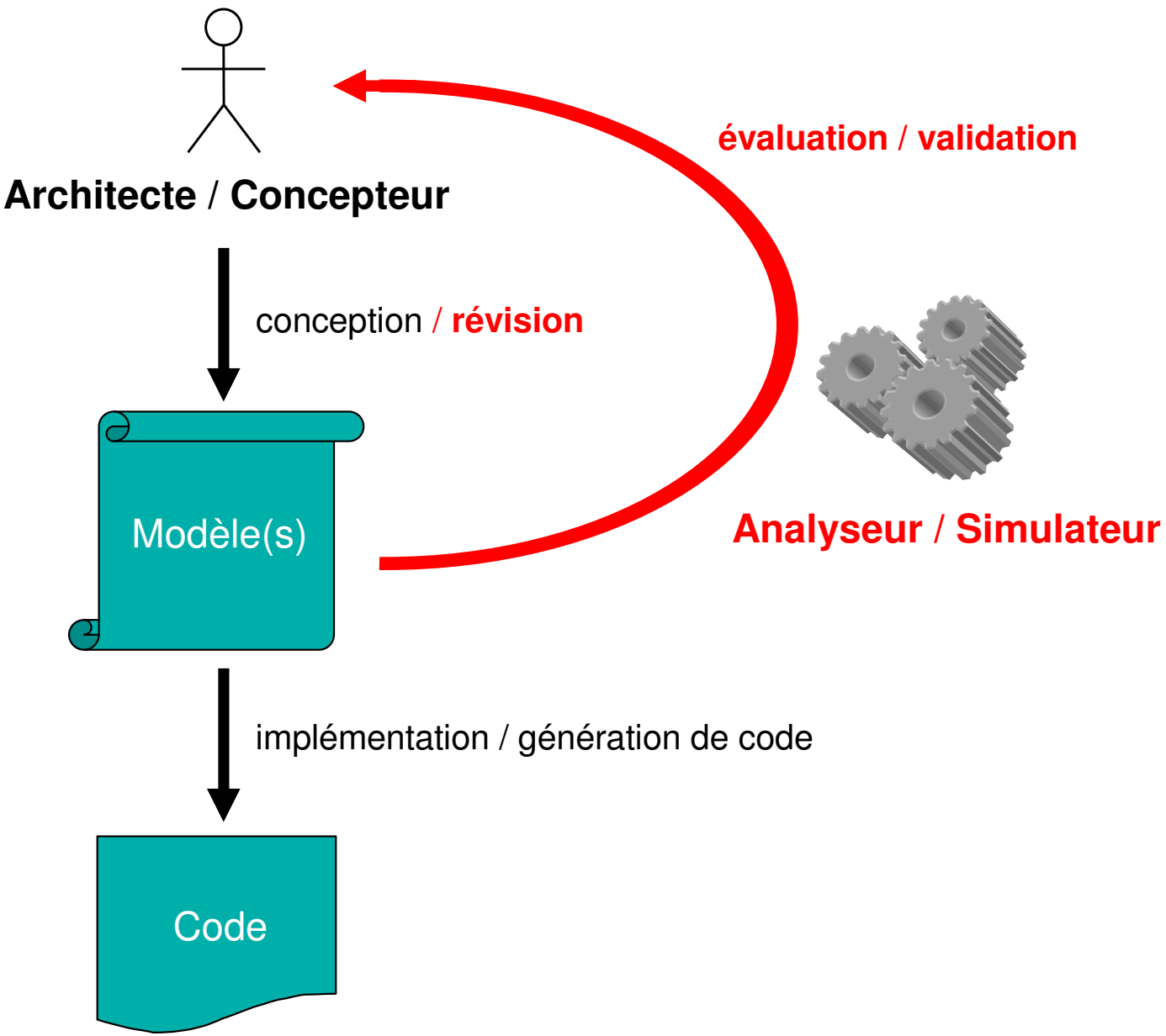
Mise en œuvre d'ATL pour la validation de systèmes

O. Constant, Verimag et Obeo

F. Jouault, AtlanMOD et Ecole des Mines de Nantes

Journées Neptune 2009, 27/05/2009

http://www.obeo.fr


Architecte / Concepteur
conception / révision
Modèle(s)
implémentation / génération de code
Code
évaluation / validation
Analyseur / Simulateur

Validation des modèles

- Validation des modèles
  - Structure : « well-formedness rules »
  - **Sémantique** : propriétés attendues
- Usage d'outils tiers spécialisés
  - Concepts d'analyse <> concepts métiers (+formats)
    - Ex: Diagramme d'activités UML / réseau de files d'attente, système de réécriture

- D'où traducteurs de modèles de conception en modèles d'analyse
  - Peuvent être complexes et difficiles à valider
  - Mais doivent être corrects (sémantiquement)

  ➔ Comment les concevoir ?

Aperçu d'ATL

- AtlanMOD Transformation Language
  - Créé par l'INRIA, industrialisé par Obeo
    - www.eclipse.org/m2m/atl/, www.atl-pro.com

- Un langage ***dédié*** à la transformation de modèles
  - Hybride impératif / **déclaratif**
  - Partie déclarative: requêtes OCL et règles
  - Haut niveau d'abstraction: n'est visible dans un modèle que ce qui est défini dans son métamodèle

  ➔ Code lisible et concis

  ➔ Risques d'erreur réduits par rapport à un langage impératif générique (ex: Java)

Une transformation ATL déclarative

- Règle standard (« matched »)

```
rule AtomicRichComponent2AtomType {
    from
        c1 : HRC!RichComponent (c1.isAtomic())
    to
        c2 : BIP!AtomType (
            name <- c1.name,
            ports <- c1.ports->collect(p | p.type.signals->collect(s |
                        thisModule.SignalOnPort2Port(
                            Tuple{ port = p, signal = s }
                        ))->flatten()
        )
}
```

Motif source

Motif cible

- Requête (« helper »)

```
helper context HRC!RichComponent def: isAtomic() : Boolean =
    self.part->isEmpty();
```

- Règle paresseuse unique (« unique lazy »)

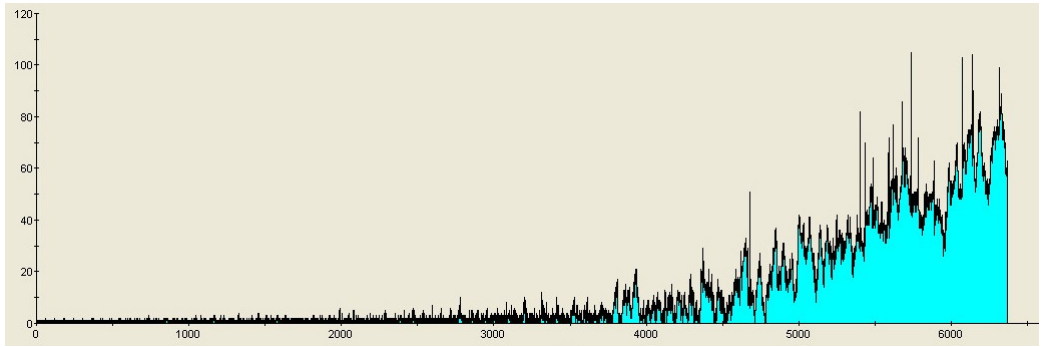
```
unique lazy rule SignalOnPort2Port {
    from
        signalOnPort : TupleType(
            port : HRC!Port,
            signal : HRC!InteractionPoint)
    to
        p : BIP!Port (
            name <- signalOnPort.signal.name,
            type <- thisModule.resolveTemp(
                        signalOnPort.port.component.package,
                        'defaultPortType')
        )
}
```

Référence à un élément cible

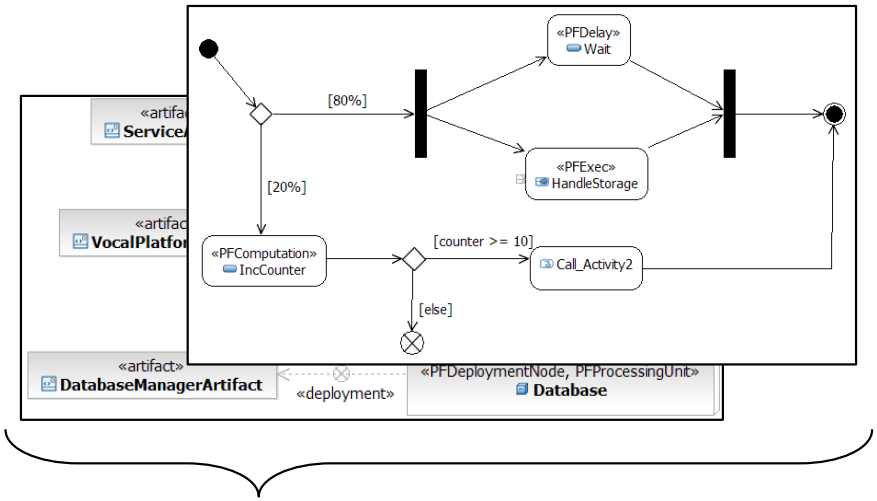
- Cas d'étude 1 : projet RNRT PerSiForm
  - INRIA, INT, Orange, Orpheus, Verimag
- Objet : évaluation des performances de SI
  - Entrée : modèle UML sous RSX avec profil dédié
    - Diagrammes de CU, d'activités, de déploiement
    - Consommations de CPU, de mémoire, de bande passante
  - Sortie : modèle de performance pour un simulateur commercial

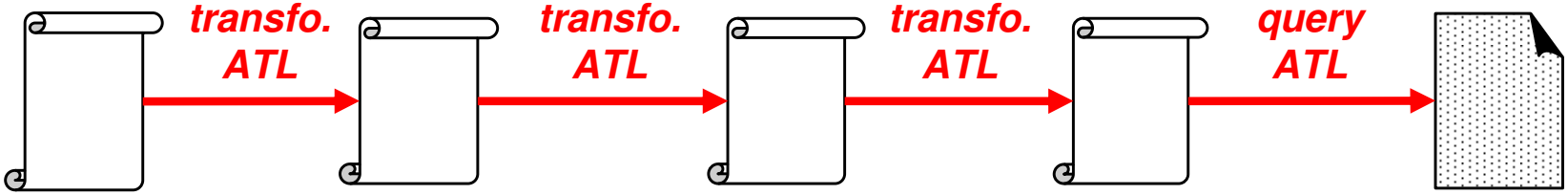
➔ Détecter les goulets d'étranglement pour valider/reconcevoir l'architecture du système

Ex: Evolution de la demande sur un CPU



Chaîne PerSiForm





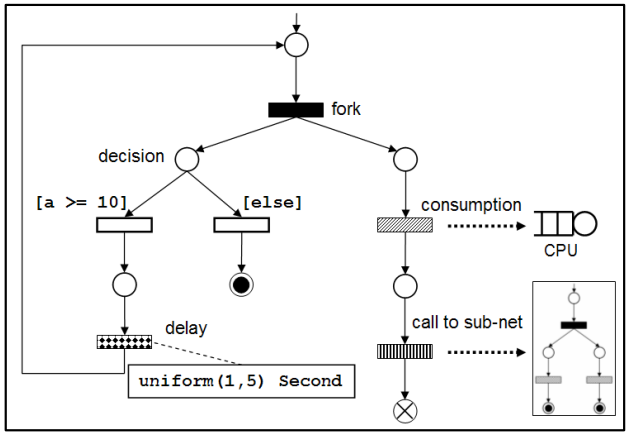
Modèle UML2
+ profil dédié

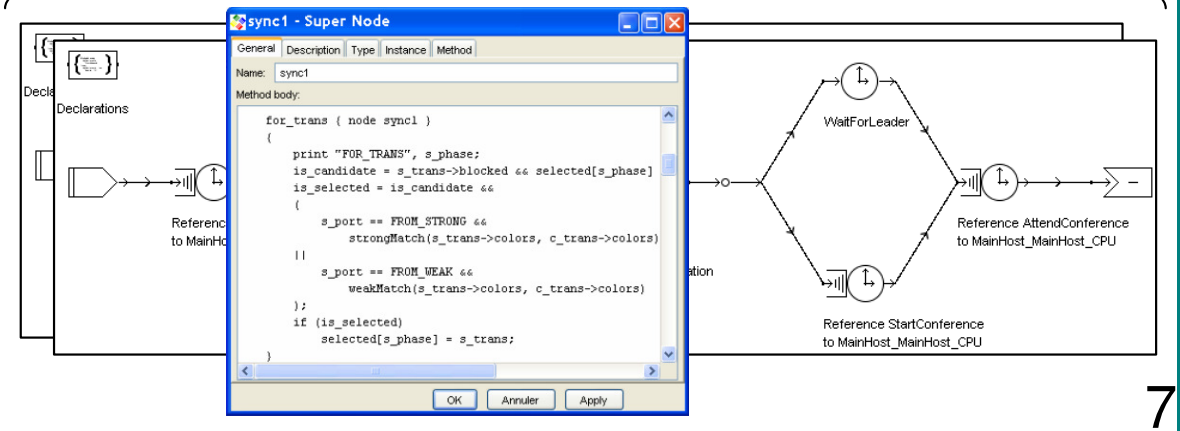
Modèle
rés. de Petri

Modèle de
performance

Modèle
XML

Fichier XML
pour
simulateur





Cas d'étude - OpenEmbeDD

- Cas d'étude 2 : projet ANR OpenEmbeDD
  - Tâche Thales / Verimag
- Objet : valider l'intégration de systèmes avioniques ARINC 653
  - Entrée : modèle UML/MARTE avec librairie Thales
  - Sortie : idem PerSiForm → réutilisation de la chaîne

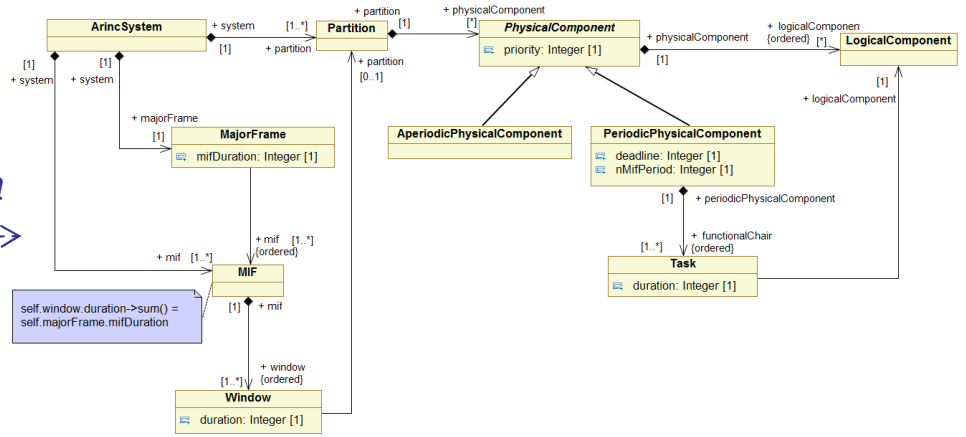
➔ Détecter d'éventuels dépassements d'échéance pour redéfinir le déploiement des composants logiques et l'ordonnancement ARINC

Extension OpenEmbeDD de PerSiForm

Modèle UML2
+MARTE
+librairie

conforme à

Modèle ARINC



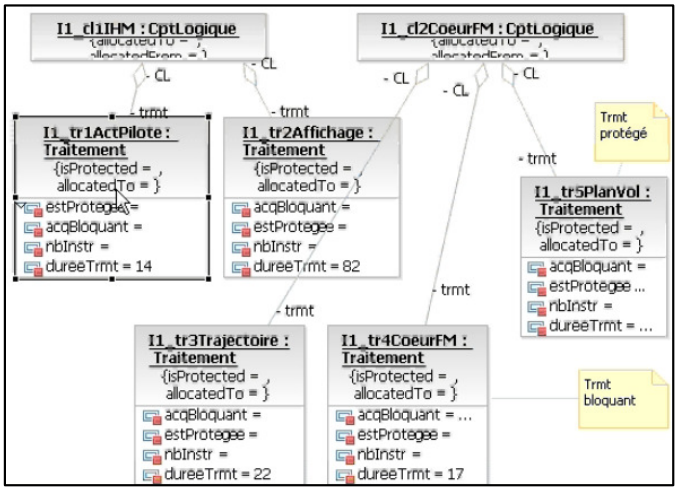
transfo. ATL

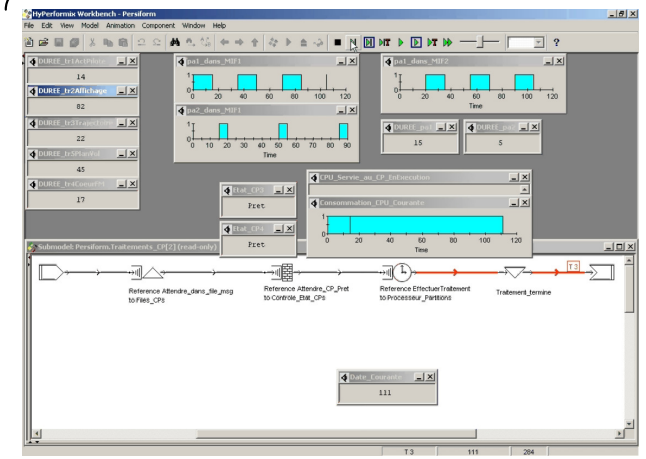
transfo. ATL

Modèle de performance

Modèle XML

Fichier XML pour simulateur



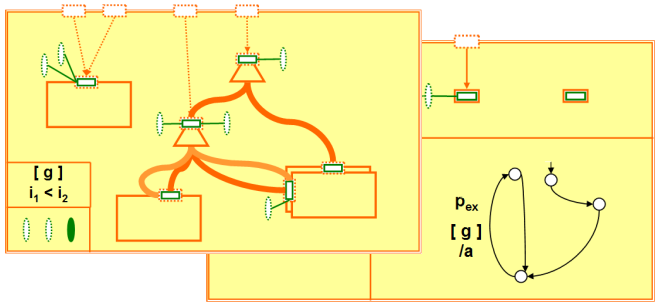


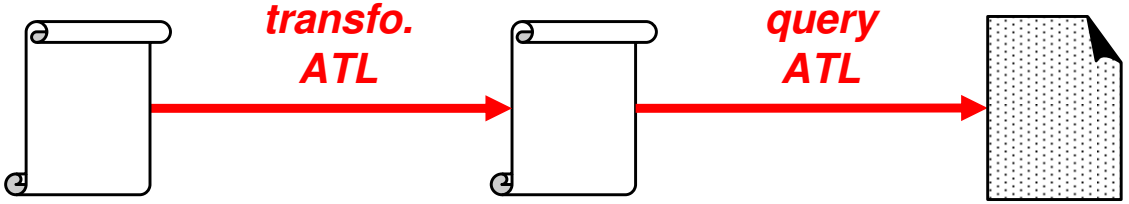
Cas d'étude - Speeds

- Cas d'étude 3 : projet européen Speeds
  - Verimag + 16 partenaires
- Objet : vérifier la cohérence fonctionnelle d'architectures à composants
  - Entrée : modèle HRC (DSL Speeds)
    - Composants munis de contrats (Assume/Promise)
    - Ex: « Si le composant reçoit un signal DemandeTransport au plus toutes les minutes, il envoie chaque fois un signal NotifieVoiture dans le quart d'heure qui suit. »
  - Sortie : spécification textuelle pour un moteur d'analyse par réécriture (Maude)

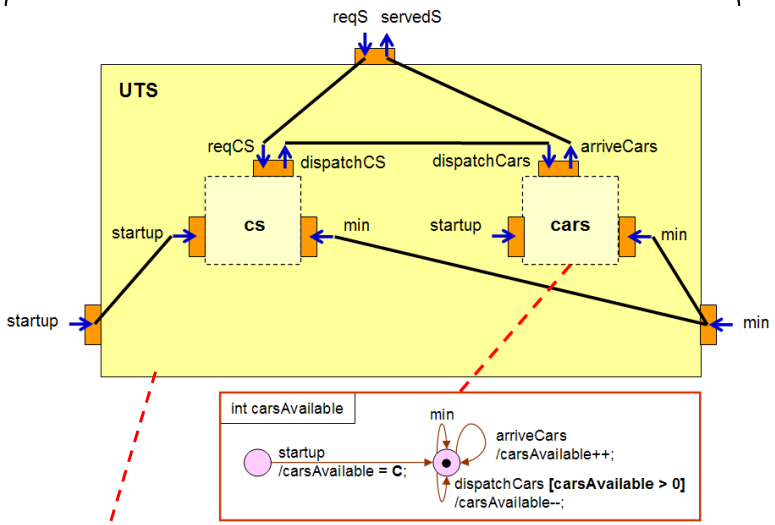
  ➔ Détecter les incohérences pour affaiblir/renforcer des contrats, remplacer un composant par un autre, modifier des assemblages

Chaîne SPEEDS





Modèle HRC | **Modèle BIP** | **Fichier Maude**




```
        encapsulate
        ||
        (
        K("CS", {'p-fc-cs_reqCS,'p-fc-cs_dispatchCS,'p-time-cs_min,'p-startup-cs_startup}, NoBeh),
        K("Cars", {'p-fc-cars_dispatchCars,'p-fc-cars_arriveCars,'p-time-cars_min,'p-startup-cars_startup}, NoBeh)
        )
        ^
        (
        < {'p-fc-cs_dispatchCS, 'p-fc-cars_dispatchCars} ; {{'p-fc-cs_dispatchCS, 'p-fc-cars_dispatchCars}} >,
        < {'p-time-cars_min, 'p-time-cs_min} ; {{'p-time-cars_min, 'p-time-cs_min}} >,
        < {'p-startup-cars_startup, 'p-startup-cs_startup} ; {{'p-startup-cars_startup, 'p-startup-cs_startup}} >
        )
        with {'p-fc-uts_reqS,'p-fc-uts_servedS,'p-time-uts_min,'p-startup-uts_startup}
        as "UTS"
        )
        ^
        (
        < {'IntervalNEvents_4_e_env, < {'p-time-cars_min, 'p-time-cs_min} ; {{'p-time-cars_min, 'p-time-cs_min}} >} ; {{'IntervalNEvents
_4_e_env, < {'p-time-cars_min, 'p-time-cs_min} ; {{'p-time-cars_min, 'p-time-cs_min}} >}} >,
        < {'p-fc-cs_reqCS, < {'EventNEvents_3_e_env, 'Pattern3_2_e1_env} ; {{'EventNEvents_3_e_env, 'Pattern3_2_e1_env}} >} ; {{'p-fc-cs
_reqCS, < {'EventNEvents_3_e_env, 'Pattern3_2_e1_env} ; {{'EventNEvents_3_e_env, 'Pattern3_2_e1_env}} >}} >,
        < {< {'p-startup-cars_startup, 'p-startup-cs_startup} ; {{'p-startup-cars_startup, 'p-startup-cs_startup}} >, < {'Pattern3_2_sta
rtup_env, 'EventNEvents_3_startup_env, 'IntervalNEvents_4_startup_env} ; {{'Pattern3_2_startup_env, 'EventNEvents_3_startup_env, 'Interv
alNEvents_4_startup_env}} >} ; {{< {'p-startup-cars_startup, 'p-startup-cs_startup} ; {{'p-startup-cars_startup, 'p-startup-cs_startup}}
 >, < {'Pattern3_2_startup_env, 'EventNEvents_3_startup_env, 'IntervalNEvents_4_startup_env} ; {{'Pattern3_2_startup_env, 'EventNEvents_
3_startup_env, 'IntervalNEvents_4_startup_env}} >}} >
        )
    ) .
endm

search pbs =>* DL(st) .
```

whenever [reqS] occurs [servedS] occurs within [(15, min)]

Conclusion

- ATL déclaratif: abstraction et flexibilité
  - Niveau d'abstraction tend à favoriser la concision, la maintenabilité et l'extensibilité
  - Traitement de modèles complexes
    - Motifs sophistiqués grâce aux différents types de règles
    - Manipulation de fragments de code enfouis
    - Mais ATL n'est pas un générateur de code (/Acceleo)
- Transformations complexes divisées en chaînes
  - Métamodèles intermédiaires pour
    - Séparer les préoccupations « sémantiques » et techniques
    - Manipuler explicitement des concepts du plus haut niveau d'abstraction possible
    - Disposer de sous-transformations réutilisables

Merci