Journée de veille technologique
« UML : au delà de la documentation »

Vendredi 10 novembre 2000, IRISA, Rennes

Méta-modèle de génération de code pour le temps-réel embarqué

François Terrier, Sébastien Gérard
Tél. : 01 69 08 62 59 ; Francois.Terrier@cea.fr

Laboratoire *Logiciel pour la Sûreté des Procédés*

Système embarqués bientôt > 50% du marché

✓ *Une part de + en + importante pour le logiciel*

☞ Comment maîtriser…

Besoins des systèmes temps réel

- ✓ Importance de la dynamique dans ces systèmes
  - → ***langages de modélisation spécialisés***
  - ➔ *mais nécessité de standard largement diffusé*

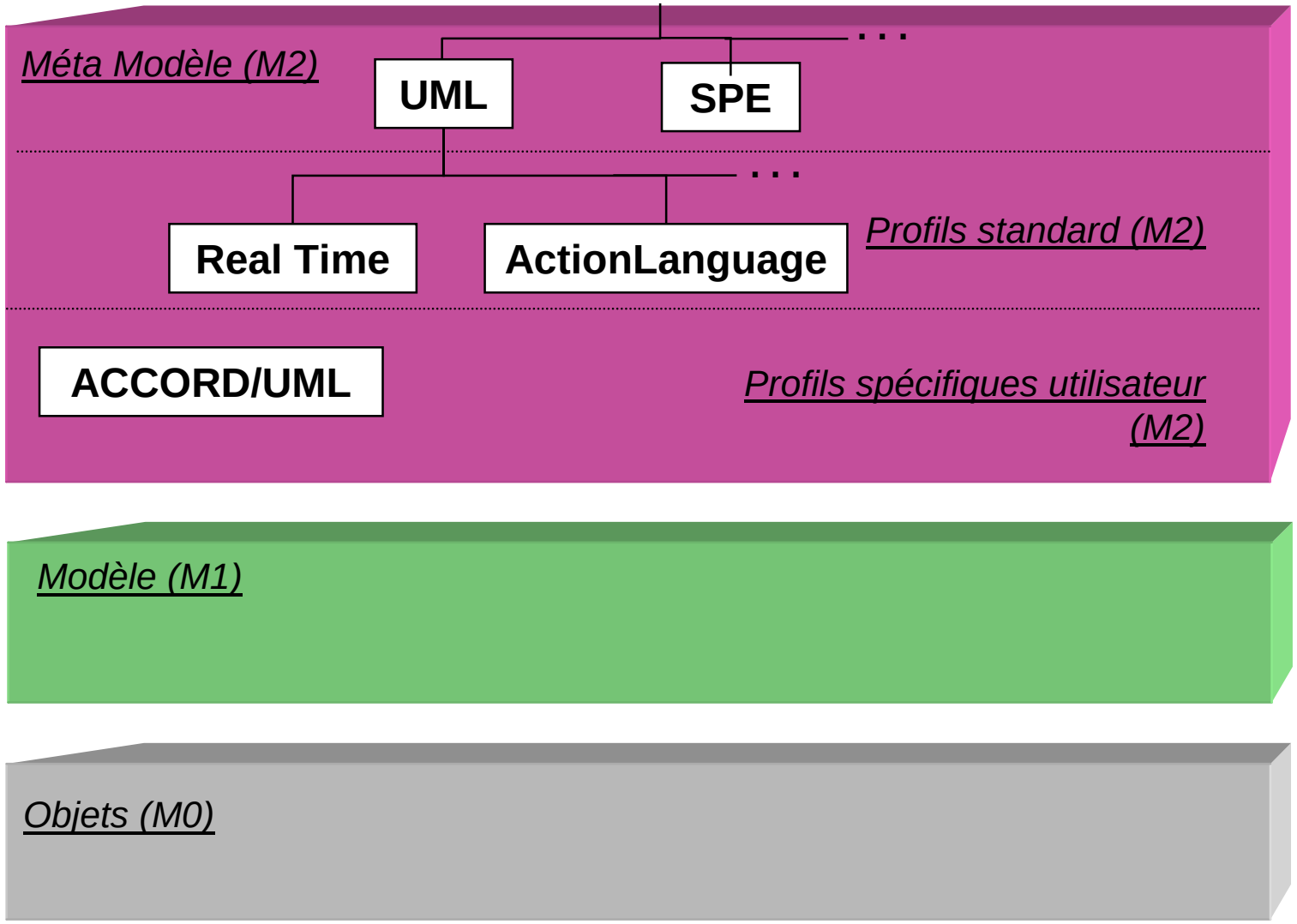
- ✓ Variabilité des pratiques en fonction des contextes

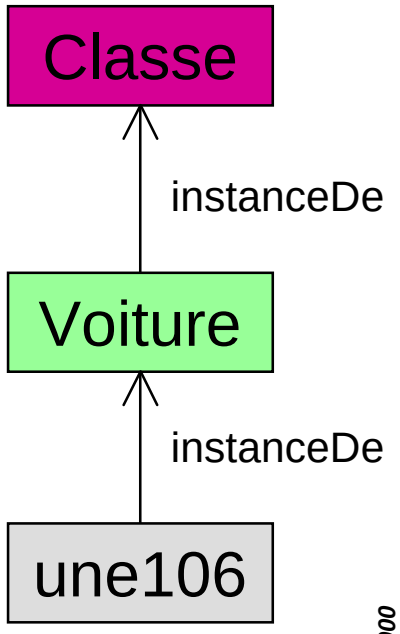
  petits systèmes enfouis ou installations de control-commande, automates, systèmes distribués, systèmes critiques pour la sûreté, diffusion de masse ou au cas par cas, grand public ou « niches », calcul parralèle…

  → *Faible degré d'intégration et d'automatisation des « bonnes pratiques »*

  ➔ ***Passage de l'artisanal à la production industrielle***

Organisation de UML





Définition d'un profil UML

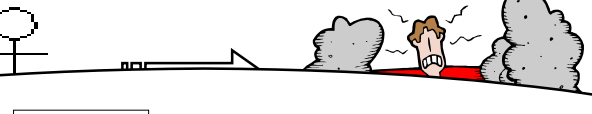
✓ Objectif

Spécialiser un [illegible] servant de référence [illegible] d'application partic[illegible]

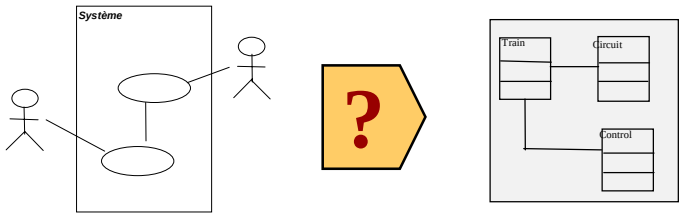
Méta-cl[illegible]

Stéréotypes, valeurs marquées et « Points de variation sémantique » et

e.g.:



e.g.:



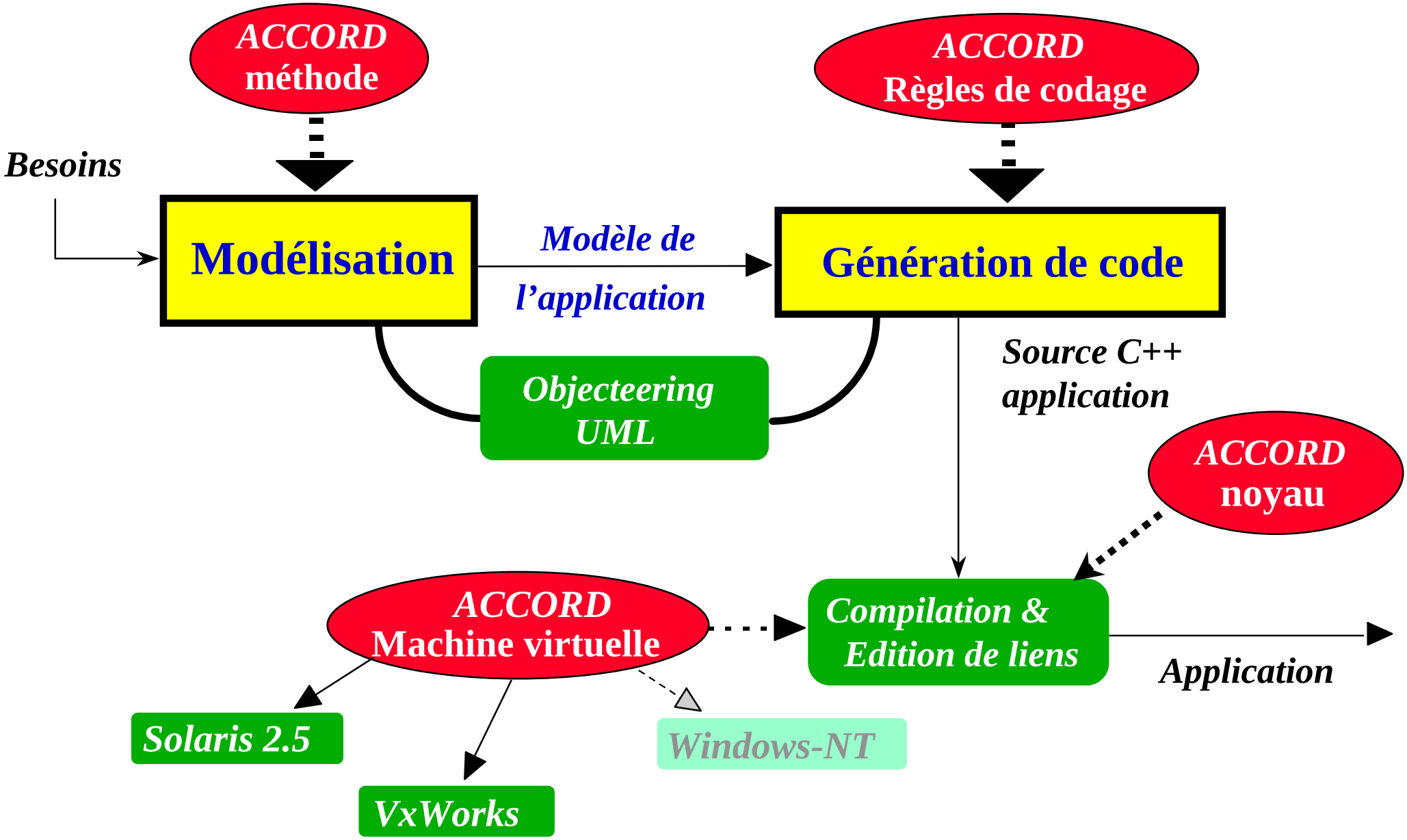
✓ Un profil peut [illegible]

- les éléments sélectio[illegible]
- des mécanismes d'extens[illegible]
- des descriptions sémantiq[illegible] profil,
- des notations supplémentaires,
- des règles de transformation, de validation ou de présentation.

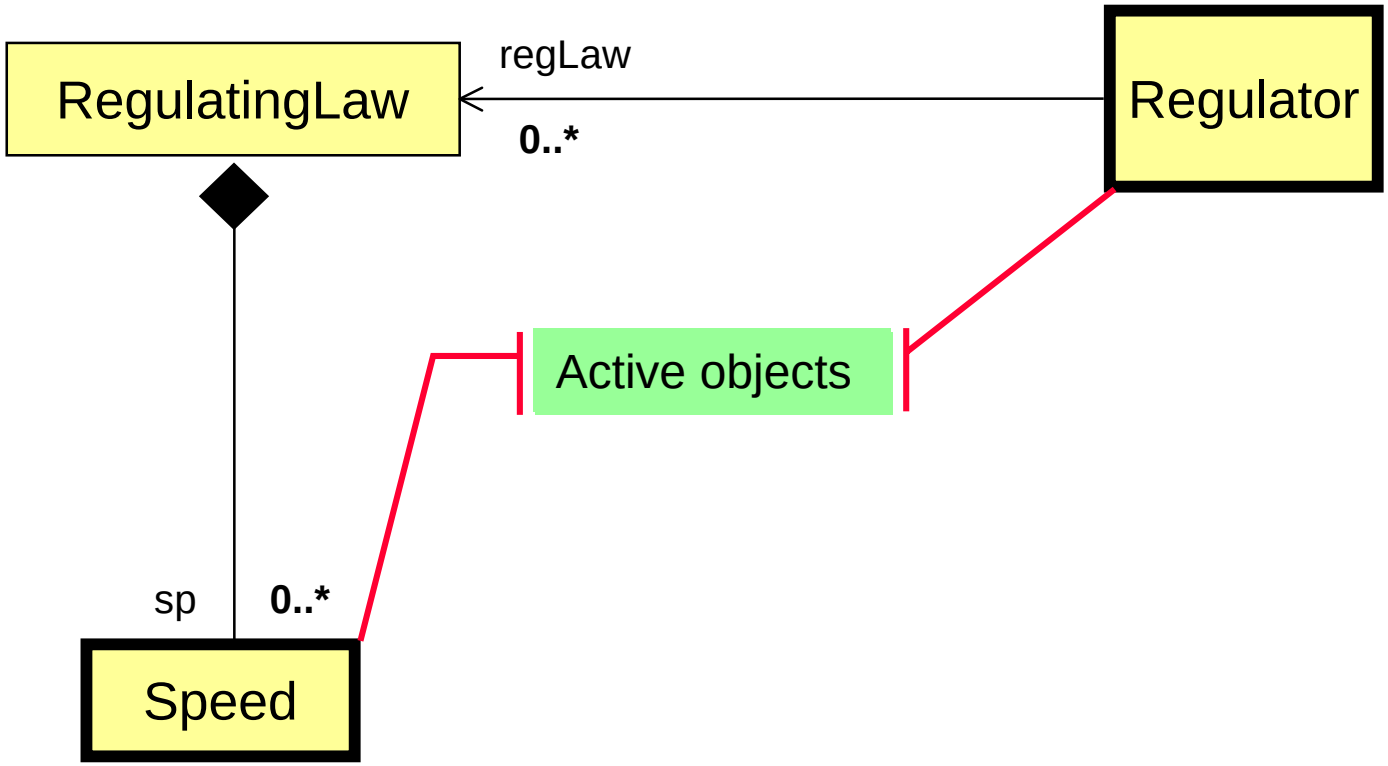
« Méta-modélisation » dans la plate-forme ACCORD

- **Introduction de concepts de modélisation « haut niveau »**
  - ✓ Objets temps réel
  - ✓ Signaux…
- **Introduction de règles de modélisation**
  - ✓ Structuration des modèles
  - ✓ Structuration des automates
  - ✓ Utilisation des signaux
- **Définition de mécanismes de mise en œuvre**
  - ✓ Pattern des signaux
  - ✓ Pattern des objets temps réel…
  - ✓ Génération automatique de code
- **Analyse de modèles pour la validation**
  - ✓ Génération automatique de séquences de tests

ACCORD : un « framework » de dévelopemment

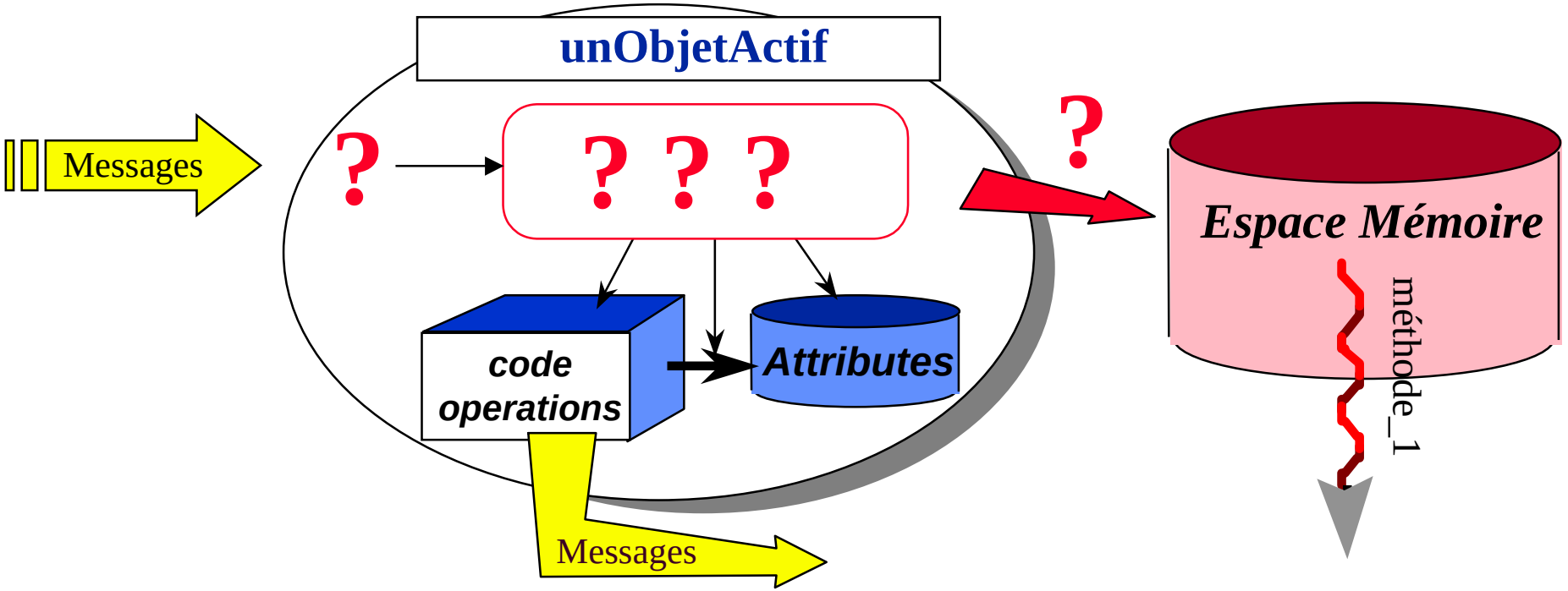


Diagrammes de classe : spécification des sources de parallélisme



→ *Déclaration des objets actifs*

Le concept UML d'objet actif

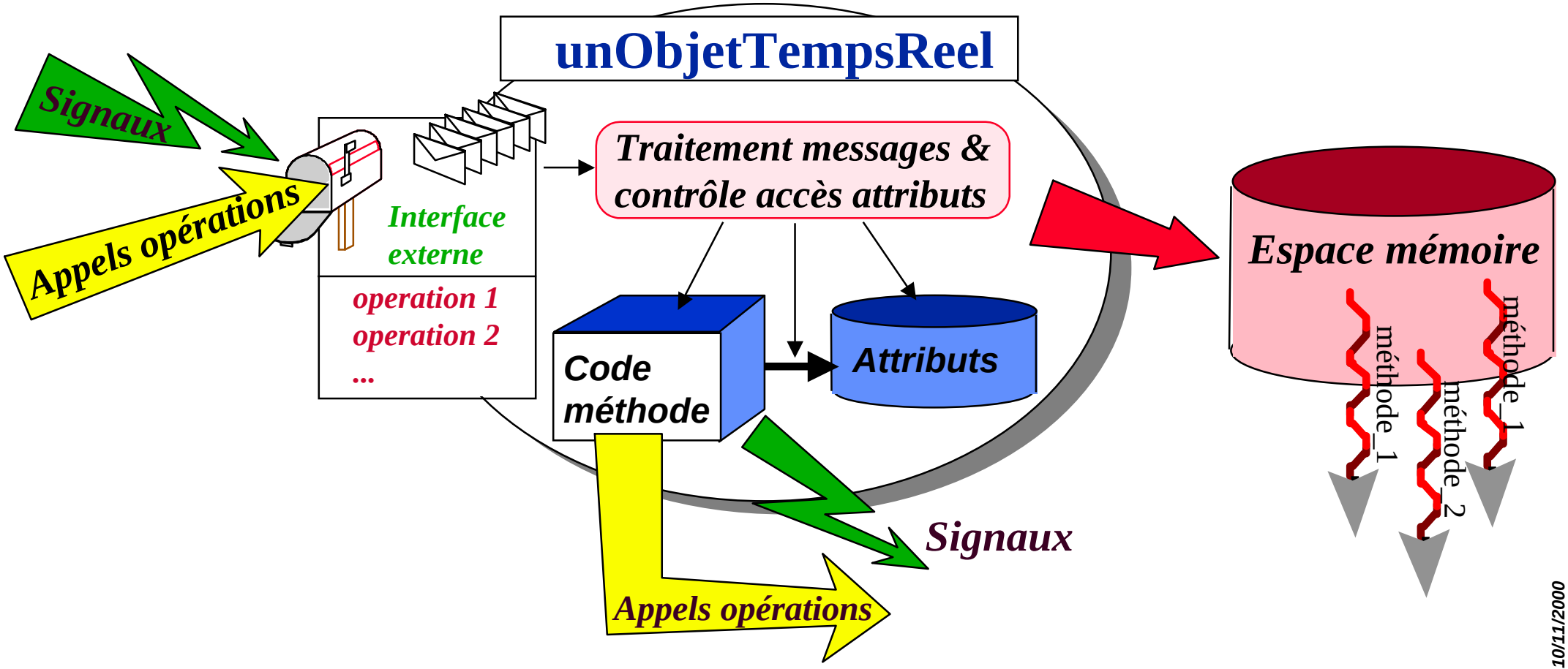


✓Caractéristiques des objets actifs de UML

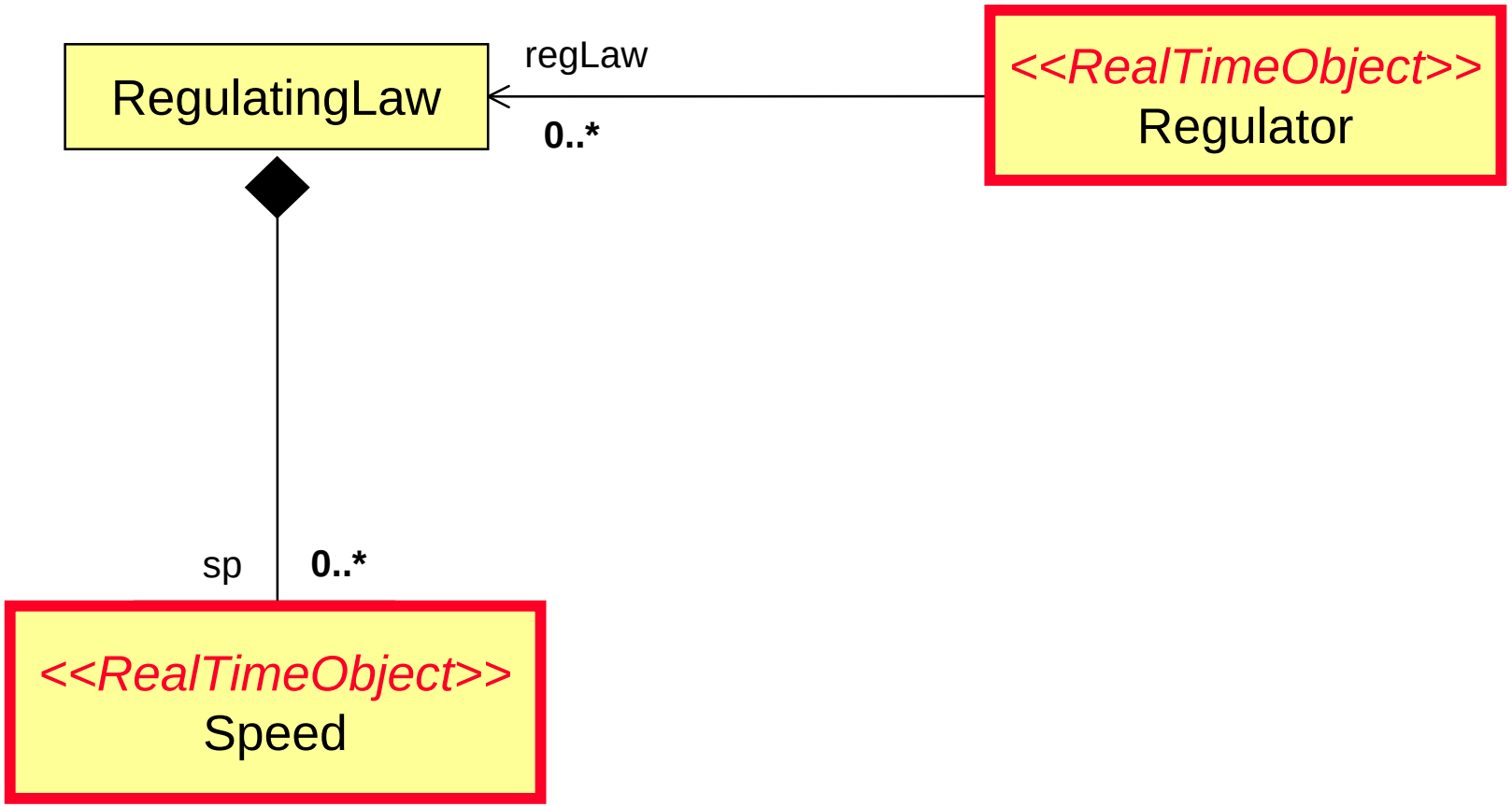
- Comportement mal défini
- Lien avec les ressources… « mono thread » ?

Objets temps réel ACCORD

Point de vue : ***une entité de calcul autonome avec une interface orientée objets UML standard***



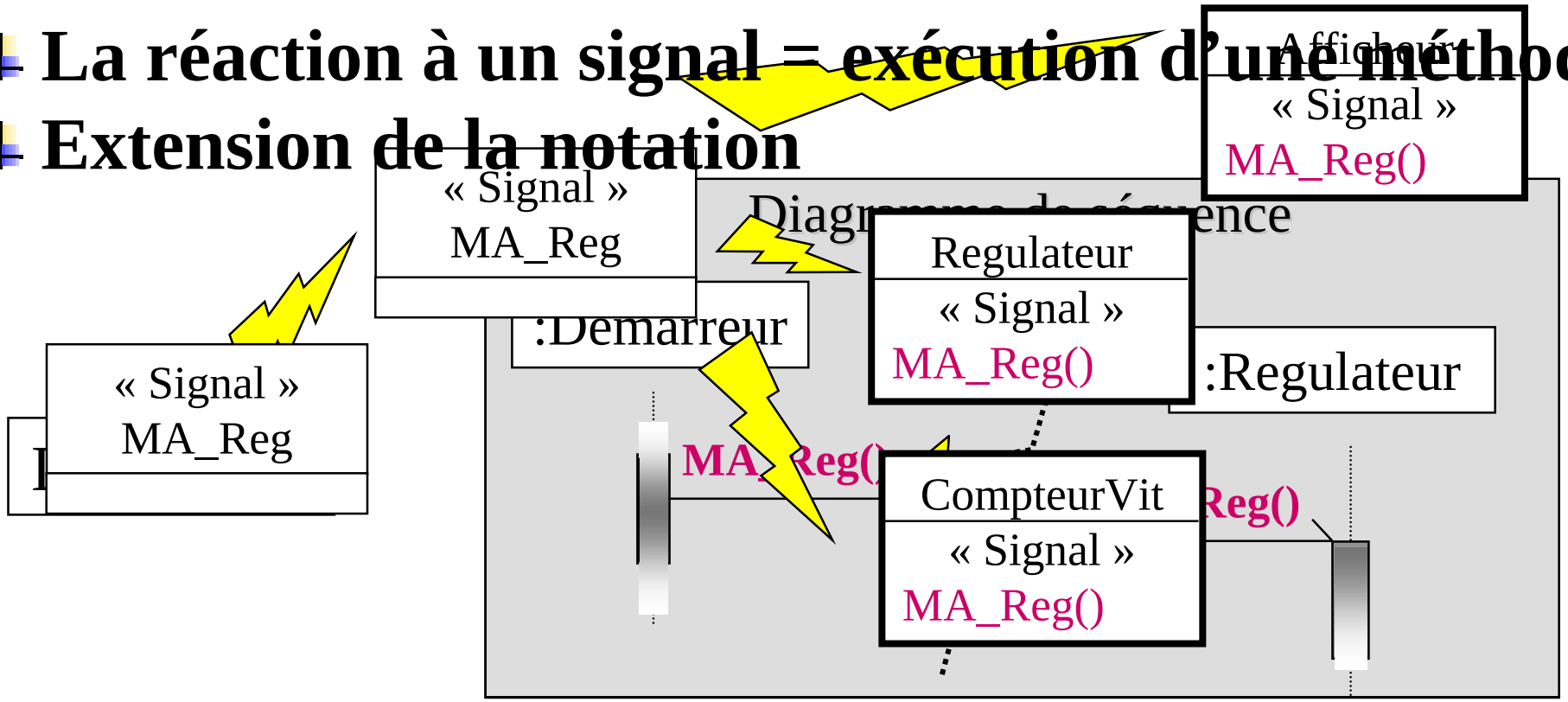
Un nouveau « stéréotype » dans les diagrammes de classe en ACCORD/UML



→ *Déclaration des objets temps réel*

Les signaux dans ACCORD/UML

- **Communication de type diffusion (target = *all*)**
  - diffusé à tous les objets déclarés sensibles
  - envoi ➔ destinataire inconnu
  - réception ➔ émetteur inconnu
- **La réaction à un signal = exécution d'une méthode**
- **Extension de la notation**



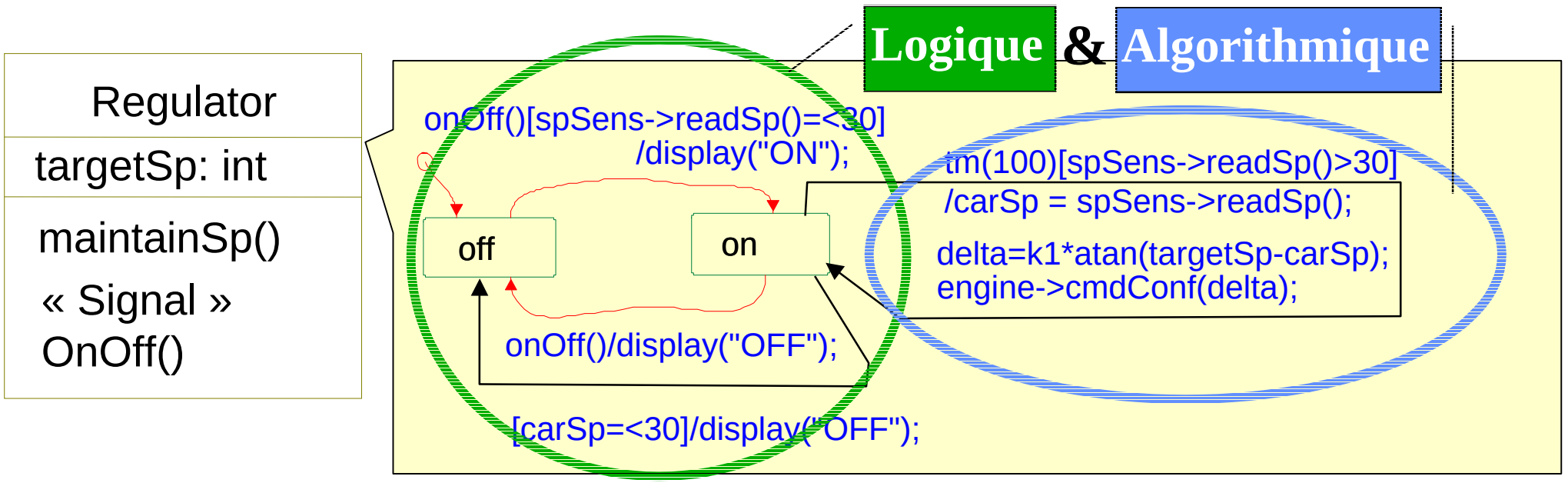
« Méta-modélisation » dans la plate-forme ACCORD

- *Introduction de concepts de modélisation « haut niveau »*
  - ✓ *Objets temps réel*
  - ✓ *Signaux…*
- **Introduction de règles de modélisation**
  - ✓ **Structuration des modèles**
  - ✓ ***Structuration des automates***
  - ✓ **Utilisation des signaux**

- *Définition de mécanismes de mise en œuvre*
  - ✓ *Pattern des signaux*
  - ✓ *Pattern des objets temps réel…*
  - ✓ *Génération automatique de code*
- *Analyse de modèles pour la validation*
  - ✓ *Génération automatique de séquences de tests*

Modélisation du comportement

Défauts: ➔ mélange logique de contrôle & algorithmique
➔ perte d'une relation claire avec l'interface de l'objet

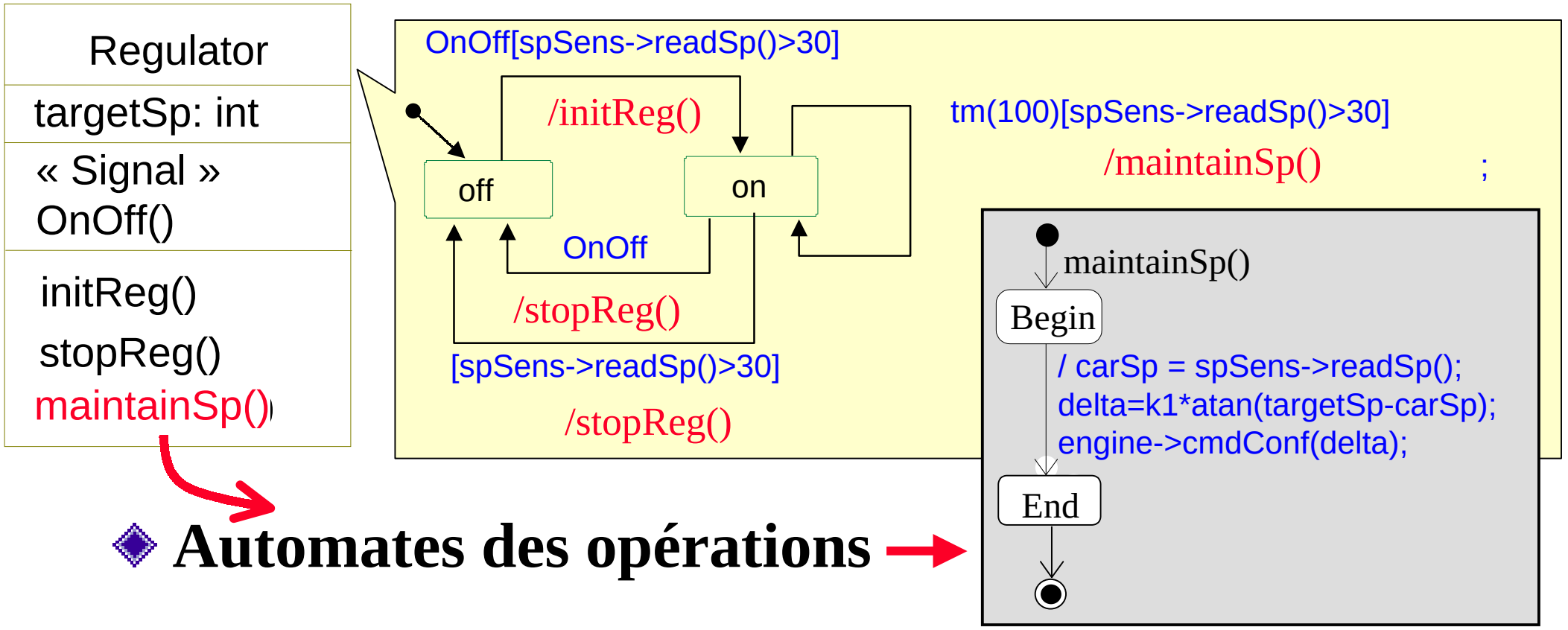


➔ Lisibilité ↘

➔ Réutilisation ↘

Approche ACCORD/UML :
séparer les problèmes

- **Les automates de contrôle**
  - ➢ ***affectation des séquences d'actions à des opérations***



- **Automates des opérations** →

Exemple de règles de modélisation ACCORD/UML en OCL

SendAction

[3] Les cibles d'une action de type *SendAction* est l'ensemble des instances constituant le système et possédant une réception attachée au type du signal émis.

self.target.body = "all"

SignalEvent

[1] La direction des paramètres d'un événement signal doit être de type « in ».

self.parameter → forAll(p | p.kind = # in)

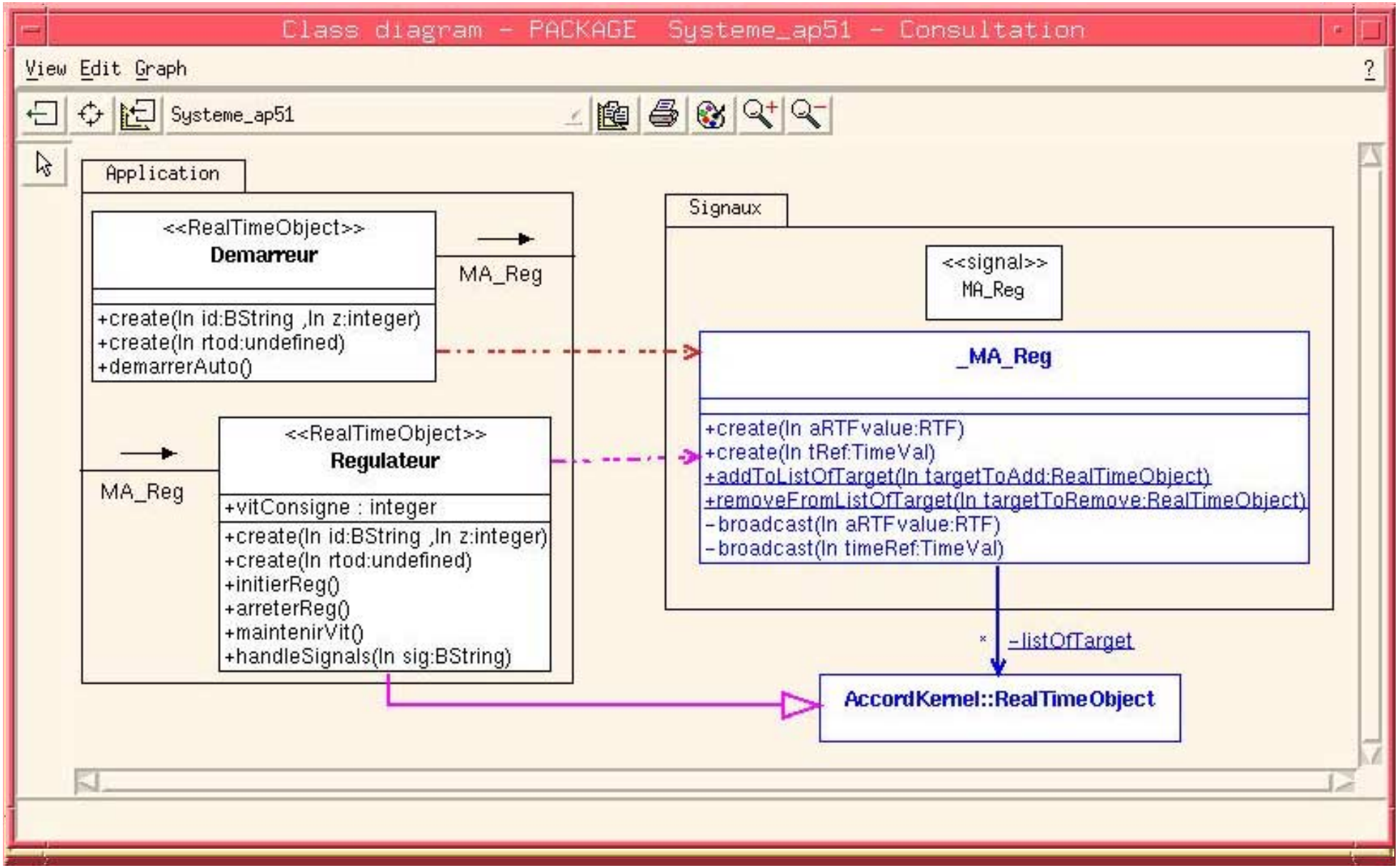
[2] Un événement de type SignalEvent possède autant de paramètres que son signal associé possède d'attributs

self.parameter → size = self.signal.allAttributes → size

« Méta-modélisation » dans la plate-forme ACCORD

- *Introduction de concepts de modélisation « haut niveau »*
  - ✓ ***Objets temps réel***
  - ✓ ***Signaux…***
- *Introduction de règles de modélisation*
  - ✓ ***Structuration des modèles***
  - ✓ ***Structuration des automates***
  - ✓ ***Utilisation des signaux***
- Définition de mécanismes de mise en œuvre
  - ✓ **Pattern des signaux**
  - ✓ **Pattern des objets temps réel…**
  - ✓ **Génération automatique de code**
- *Analyse de modèles pour la validation*
  - ✓ ***Génération automatique de séquences de tests***

Mise en œuvre de la communication par signal



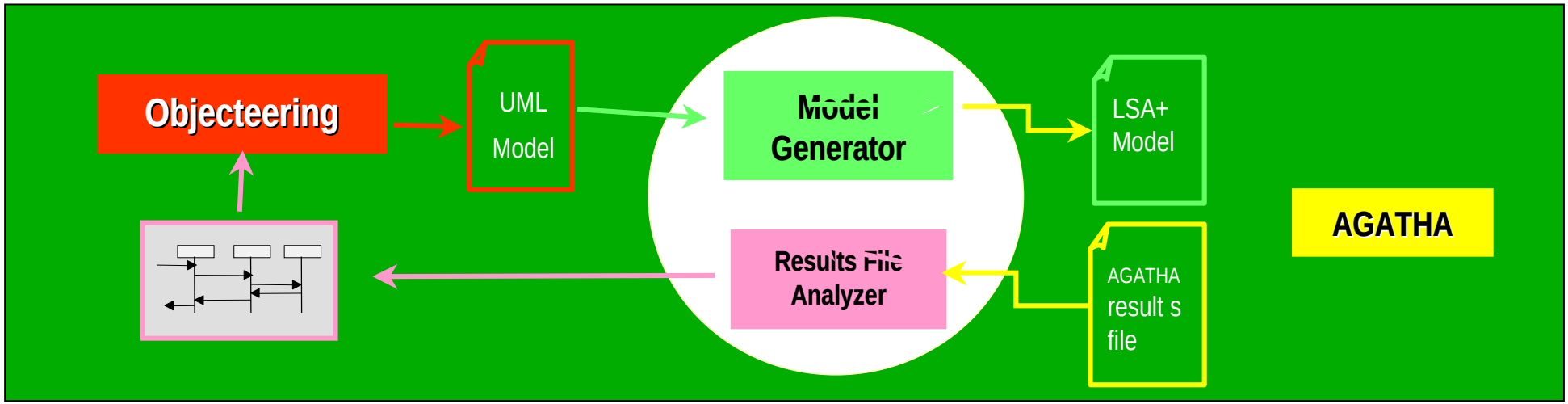
« Méta-modélisation » dans la plate-forme ACCORD

- *Introduction de concepts de modélisation « haut niveau »*
  - ✓ ***Objets temps réel, signaux…***
- *Introduction de règles de modélisation*
  - ✓ ***Structuration des automates***
  - ✓ ***Structuration des automates***
  - ✓ ***Utilisation des signaux***
- *Définition de mécanismes de mise en œuvre*
  - ✓ ***Pattern des signaux***
  - ✓ ***Pattern des objets temps réel…***
  - ✓ ***Génération automatique de code***

Analyse de modèles pour la validation

- ✓ **Génération automatique de séquences de tests**

Test cases generation process



- Designer invokes the tool which translates UML models into elementary automaton specifications
- AGATHA generates an exhaustive functional tree with associated Path Conditions which is converted into UML Sequence Diagrams
- The designer can study these diagrams and fix the errors in the model
- This process can be repeated until the model is correct

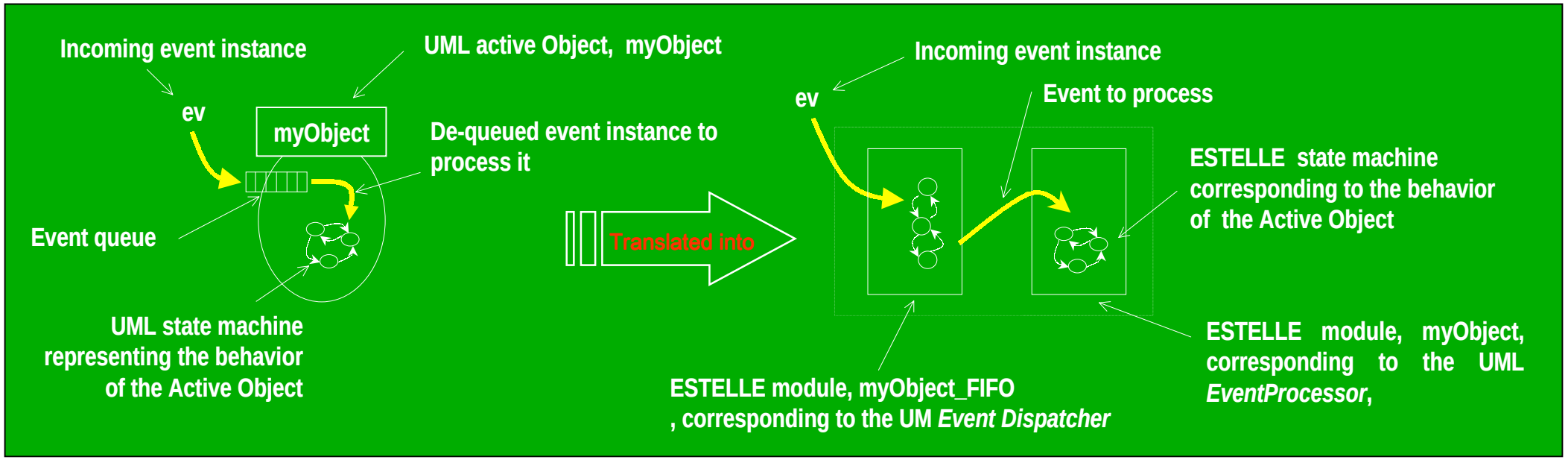
⇒ Fully **transparent** processus : the user does not have to know anything about AGATHA or its low level language

UML active objects conversion

Communication between UML active objects

- ✓ **Asynchronous with message queues**
- ✓ **Queuing and precessing rules of UML state machines**
- ✓ **RTC assumption to respect**

➔ ***1 Active Object = 2 ESTELLE modules***



Conclusions

- **Profil ACCORD/UML => expression haut niveau de:**
  - Contrainte TR, Concurrence, Communication, Comportement
- **AGL industriel : Objecteering + modules d'extension**
  - Support des notations dédiées, Règles de modélisation
  - Génération de code TR mutli-tâches (C++, Solaris et VxWorks)
  - Framework multi-tâches temps-réel
- **Évaluation sur un cas d'étude PSA: *Régulateur***
- **Contrat avec PSA pour le transfert**
- **Évaluation en cours sur application Télécom**

Travaux en cours et perspectives...

Poursuite des travaux

- Liens outils existant (Rhapsody, TAU UML) ⇒ *AIT-WOODDES*
  ⇒ ***proposition OMG***
- Relation modèles flots de données/synchrones ⇒ *RNTL ACOTRIS*
- Déploiement/distribution et lien Soft/Hard ⇒ *CEA-DAPNIA*
  → *Intégration des standard pour les applications de simulations*
- Techniques de validation par le test d'une application UML
  ⇒*Thèses « Propriétés Temps Réel » et « composants »*

Perspectives ouvertes

- Formalisation et génération de modèles d'implantation, définition d'une bibliothèque de patrons dédiés TR
- Intégration des contraintes liées aux systèmes critiques
- Liaison avec le Co-Design et les modèles continus