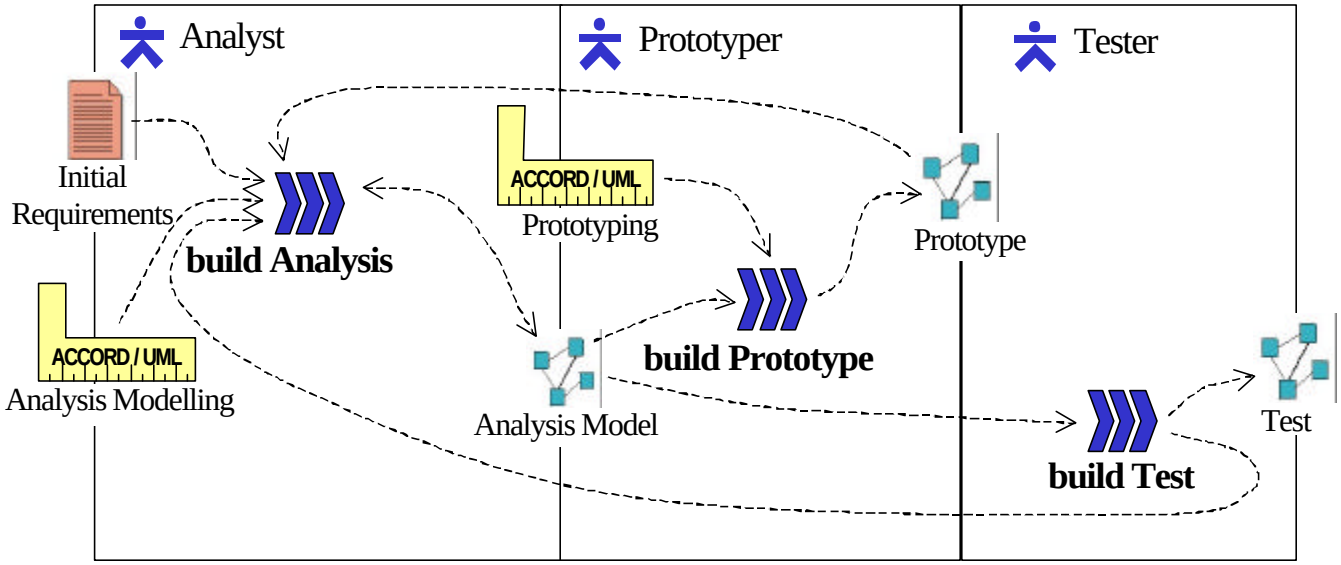
Modélisation UML Temps Réel pour une implantation synchrone

1. Base méthodologique UML
2. Évolution pour implantation synchrone
3. Étapes à venir

Base méthodologique

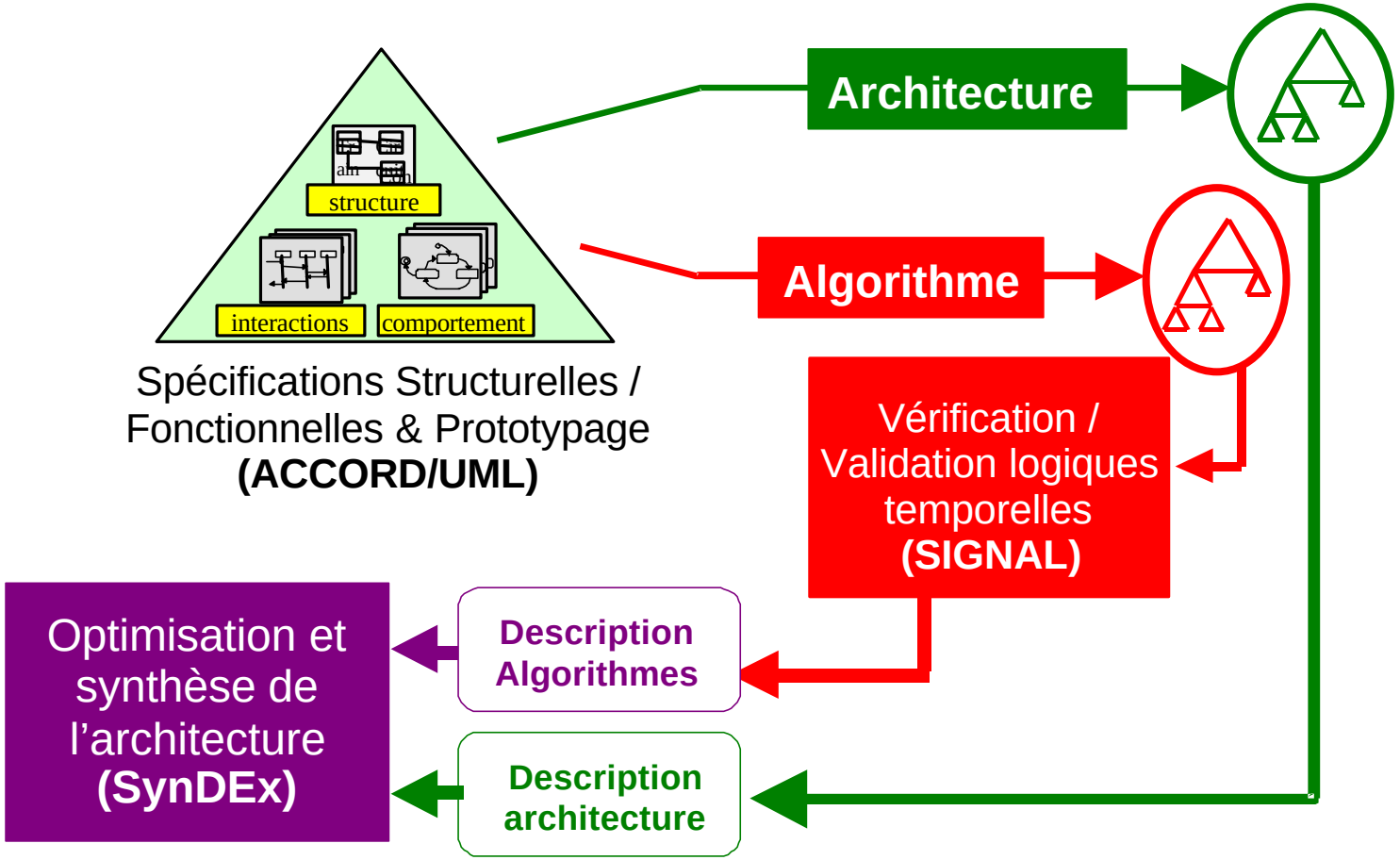
- Méthode ACCORD/UML
  - Validation des concepts avec partenaires extérieurs
  - Création d'un exemple de référence (régulateur vit.)
  - Création d'un guide utilisateur



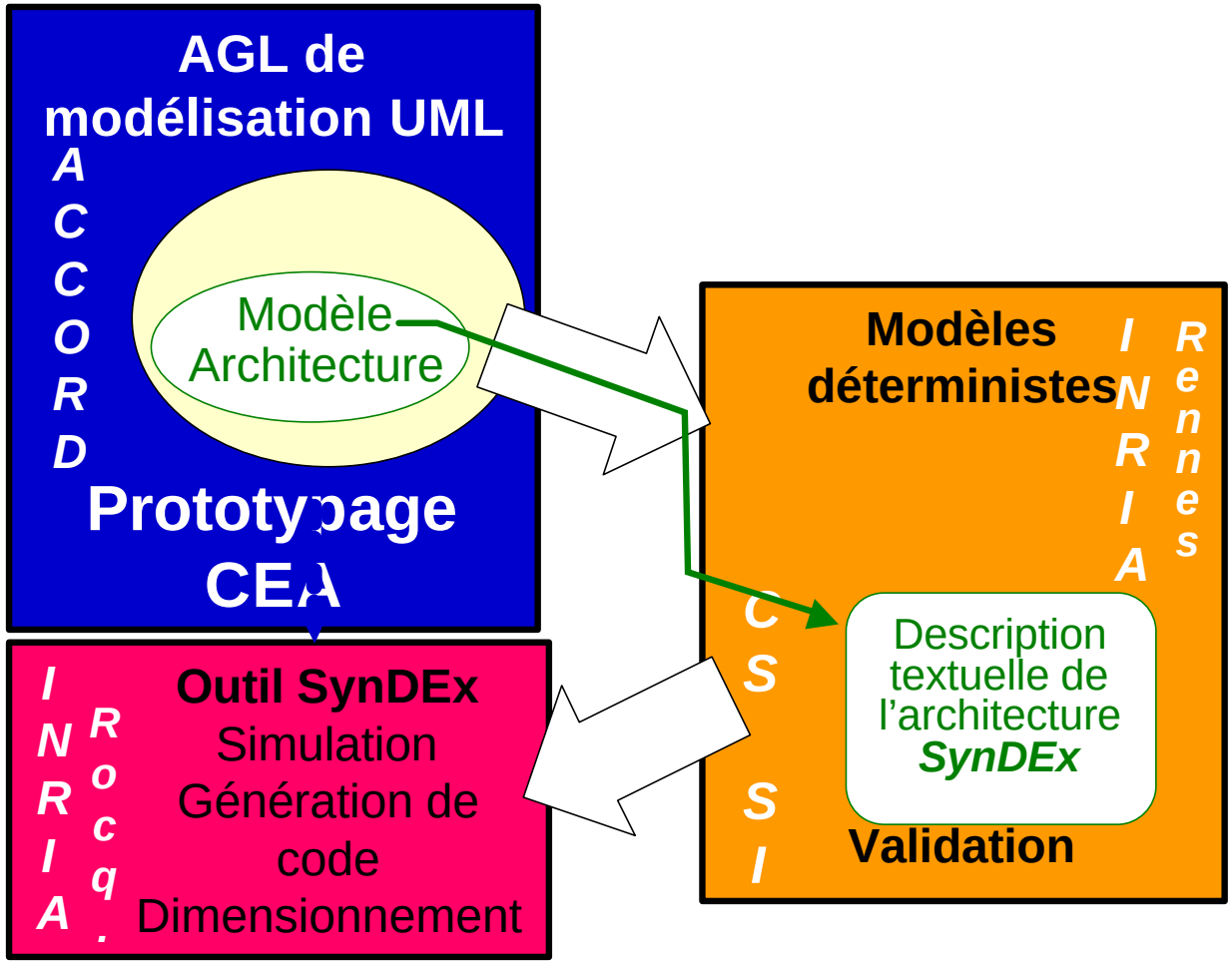
Supports méthodologiques

- Analyse Préliminaire & Détaillée
  - → Module d'analyse préliminaire & détaillée : PAM & DAM
    - → *Réalisé en « J » sous Objecteering, Livré début mars*
  - → Modèle de l'exemple de référence, niveau PAM & DAM
    - → *Réalisé en UML sous Objecteering, livré début mars*
  - → Manuel utilisateur, modélisation niveau PAM
    - → *Document word avec clips intégrés, livré début mars*
- Prototypage
  - → Module de prototypage : PrM
    - →*Réalisé en « J » sous Objecteering + bibliothèque classes C++ Livré début avril*
  - → Modèle de l'exemple de référence, niveau PrM
    - → *Réalisé en UML sous Objecteering, livré début avril*

Évolution pour implantation synchrone



Volet architecture



Architecture matérielle dans UML

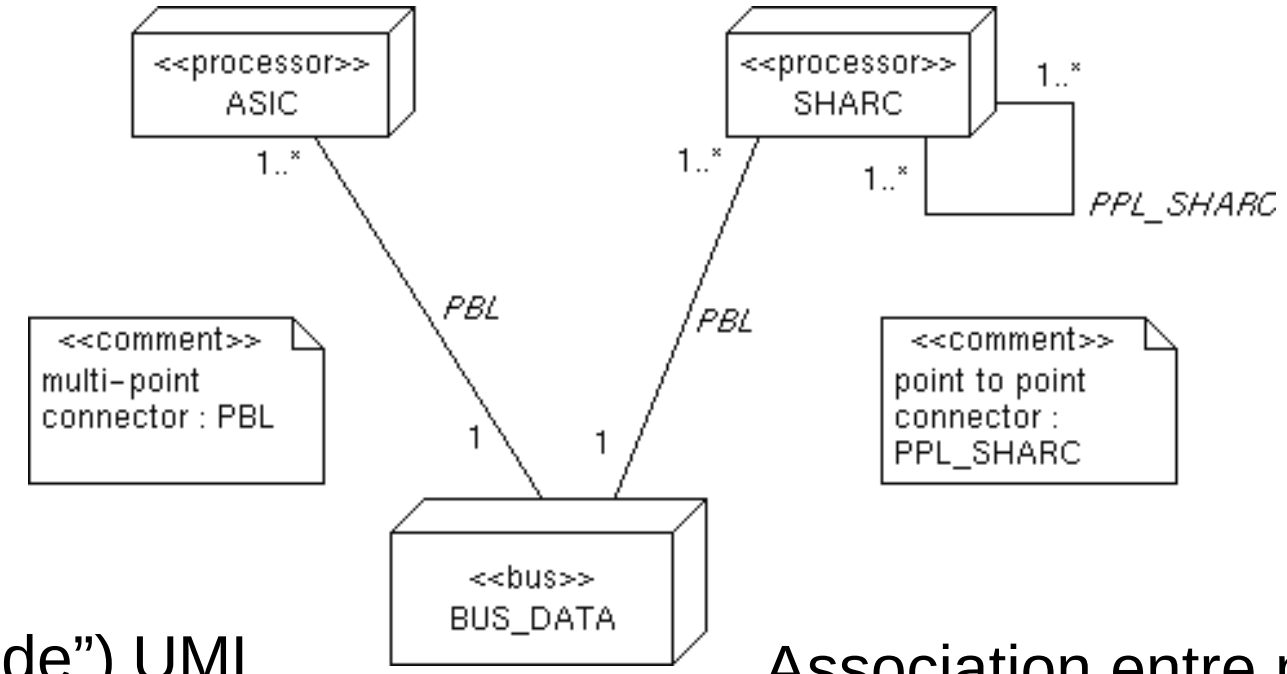
- 2 points de vues…
  - *Eléments de référence du déploiement*
    → Diagramme de déploiement :
    - Description des éléments physiques (nœuds) qui composent le système

Types (classes) des nœuds

  - *Emploi des éléments de références*
    → Diagramme d'instance de déploiement :
    - Topologie exacte retenue
    - Instances des nœuds décrits dans le diagramme de déploiement

Instances des nœuds

Exemple : diagramme de déploiement



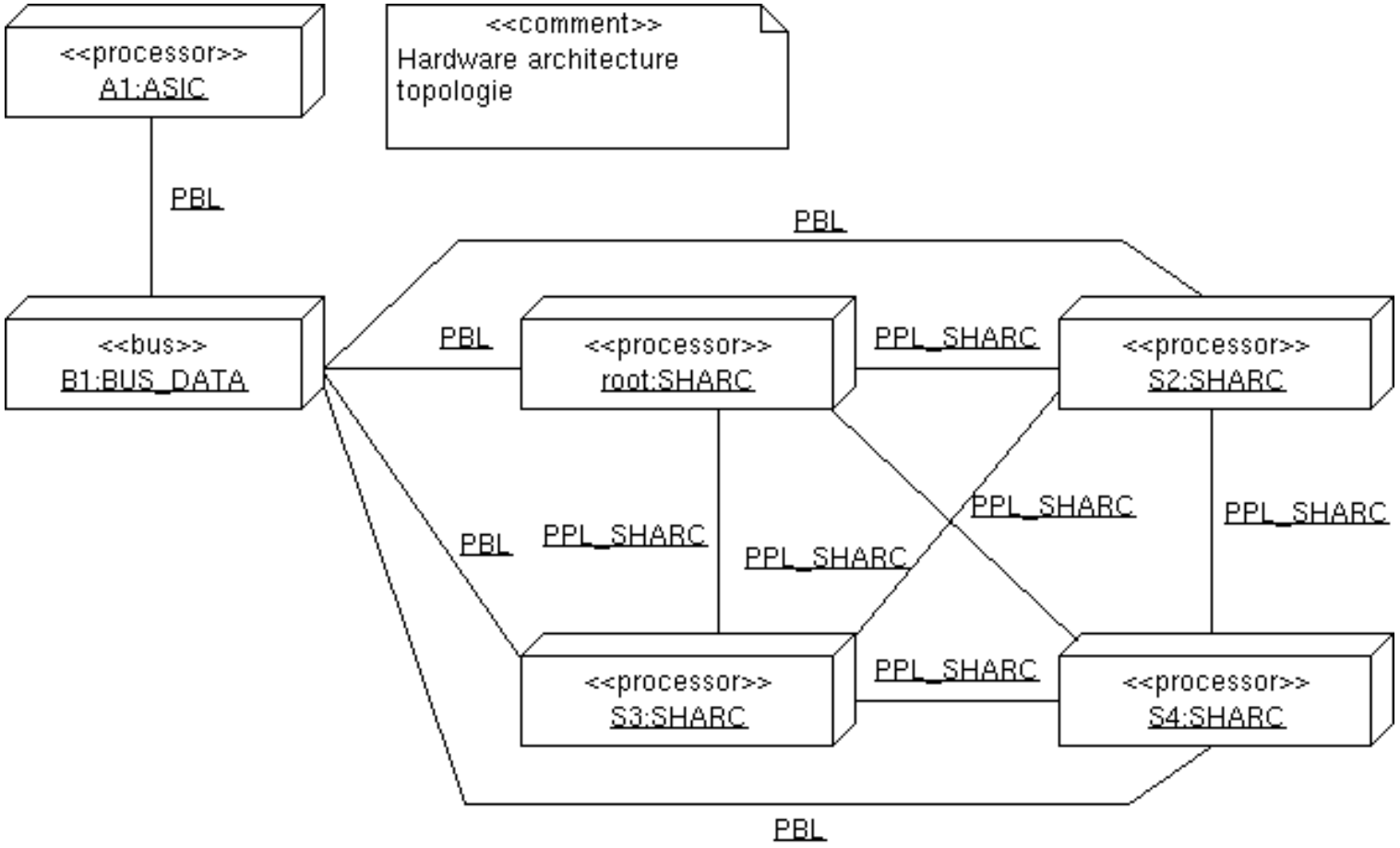
Nœud (”node”) UML

- Ressource matérielle physique
  → peut-être précisée par stéréotype
  (ex : « Dispositif », « Mémoire »)

Association entre nœuds

- Support de communication
  → peut être précisée stéréotype
  (ex : « TCP/IP », « RNIS »)

Exemple : diagramme d'instance de déploiement



Choix retenus

➔ Norme UML 1.4 :

- ✓ *Utilisation des diagrammes de déploiement floue dans UML*
- ✓ *Nécessite de caractériser des types de nœuds et leur utilisation*
- ✓ *Choix retenus inspirés de la description matérielle en SynDEx*

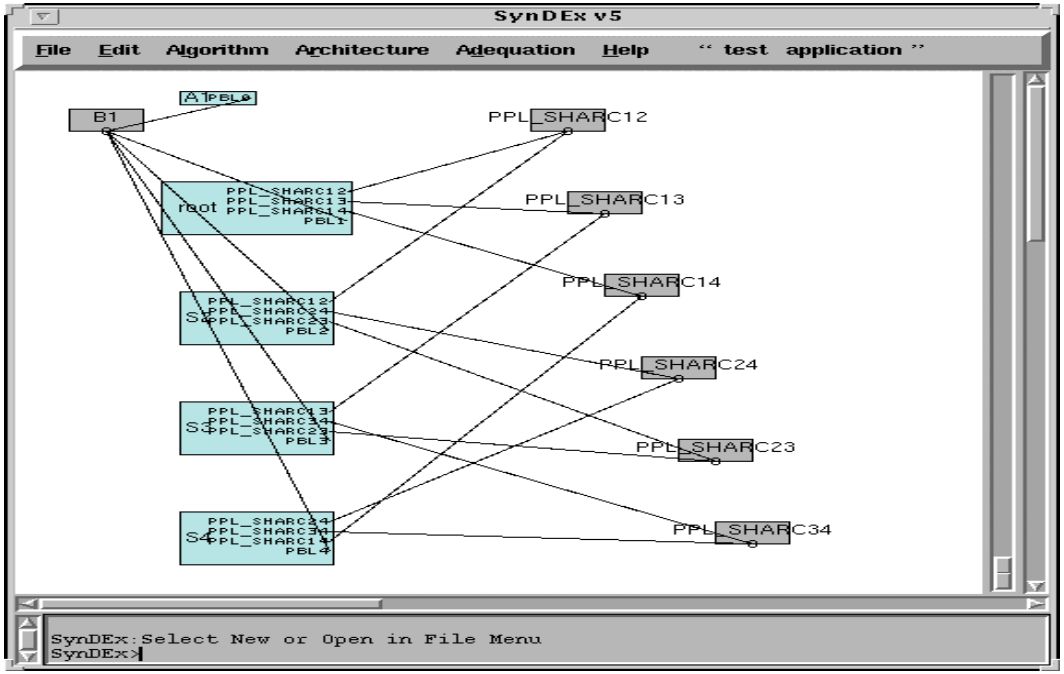
➔ Spécialisation pour ACOTRIS :

- ✓ Stéréotype «*processor*» : *Ressource de calcul*
  - ➢ Possède en général une mémoire interne
  - ➢ Connectée à d'autres «processor» (*pt à pt*) ou à un «bus»
- ✓ Stéréotype «*bus*» : *Support de communication*
  - ✓ Connecteur multipoints
  - ✓ Ne peut pas être relié a un autre nœud «bus»
  - ✓ Diagrammes de déploiement plus lisibles
- ✓ *Enrichir le métamodèle avec des stéréotypes*

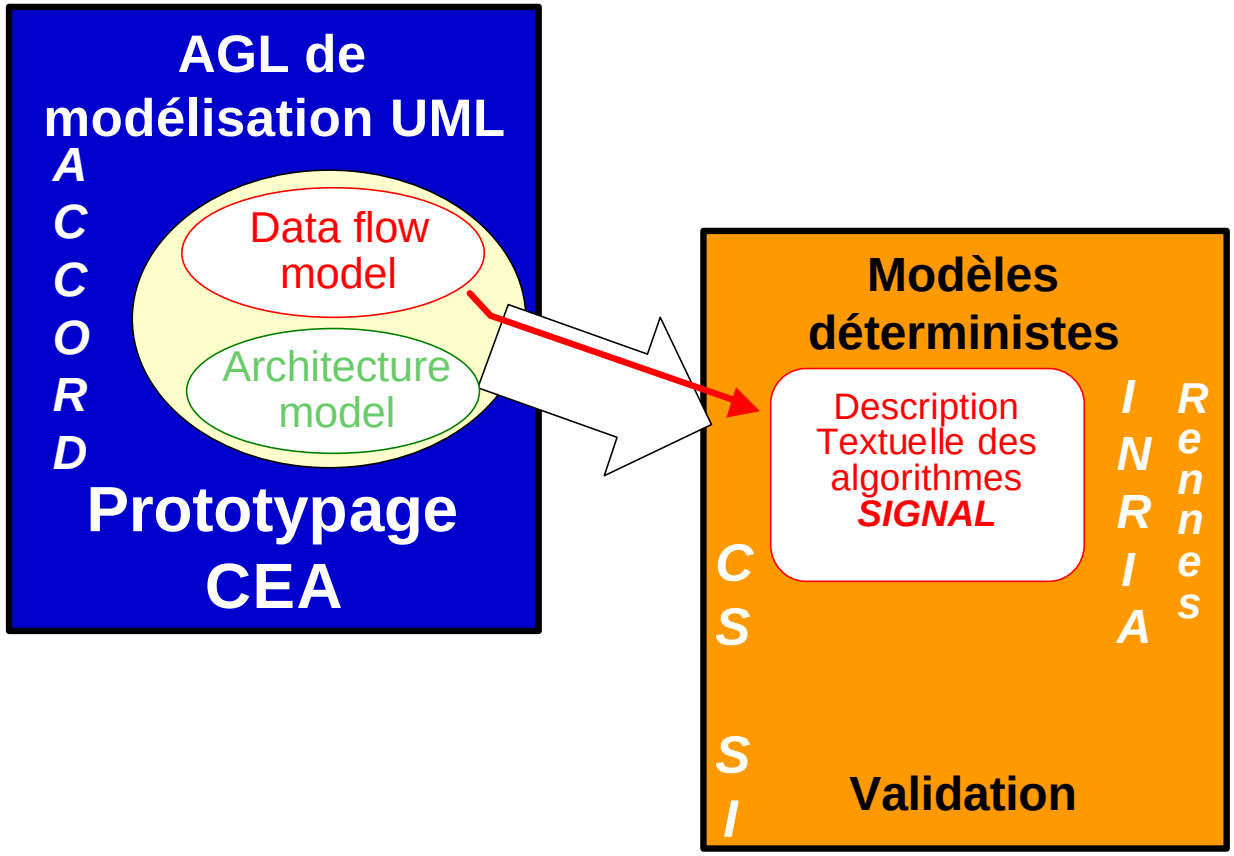
Production du modèle SynDEx

➔ Module de transformation UML dédié

- ✓ *Établir les contrôles cohérence modèle d'arch. Matérielle*
- ✓ *Créer un fichier format SynDEx avec infos matériel*

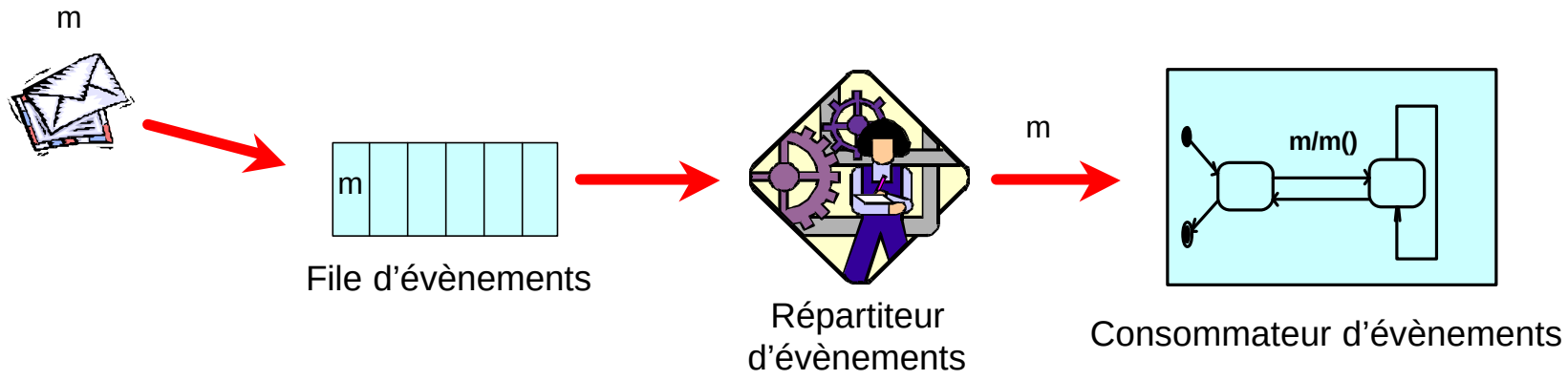


Volet fonctionnel



Sémantique des machines à états

UML Standard



- Pas RTC (Run-To-Completion) / 1 seul évènement à la fois

=> Une sémantique à « paramétrer » !!!

Sémantique des machines à états

- Une sémantique à paramétrer
  - Points ouverts de variation sémantique (POVS) => paramétrer !!!
    (ex : politique d'extraction des évènements de la file d'attente, …)
  - Ambiguïtés sémantiques => clarifier !!!
    (ex : *terminaison des états orthogonaux, …*)
- ACCORD/UML
  - POVS
    - File d 'attente = Boîte aux lettres
    - Extraction des messages = Earliest Deadline First
  - Ambiguïtés sémantiques
    - Règles supplémentaires décrites en OCL

    (ex : pas d'états orthogonaux = CompositeState - self.subvertex->select(v|v.oclIsKindOf(CompositeState))->select(p|p.isConcurrent = true)->size = 0)

Modélisation SIGNAL de la file d ’attente

- Module BAL SIGNAL :
  - Interface SIGNAL

```
Sig_BAL = struct        { [size] (msg, dl);
                          int nb_msg;
                        }
```

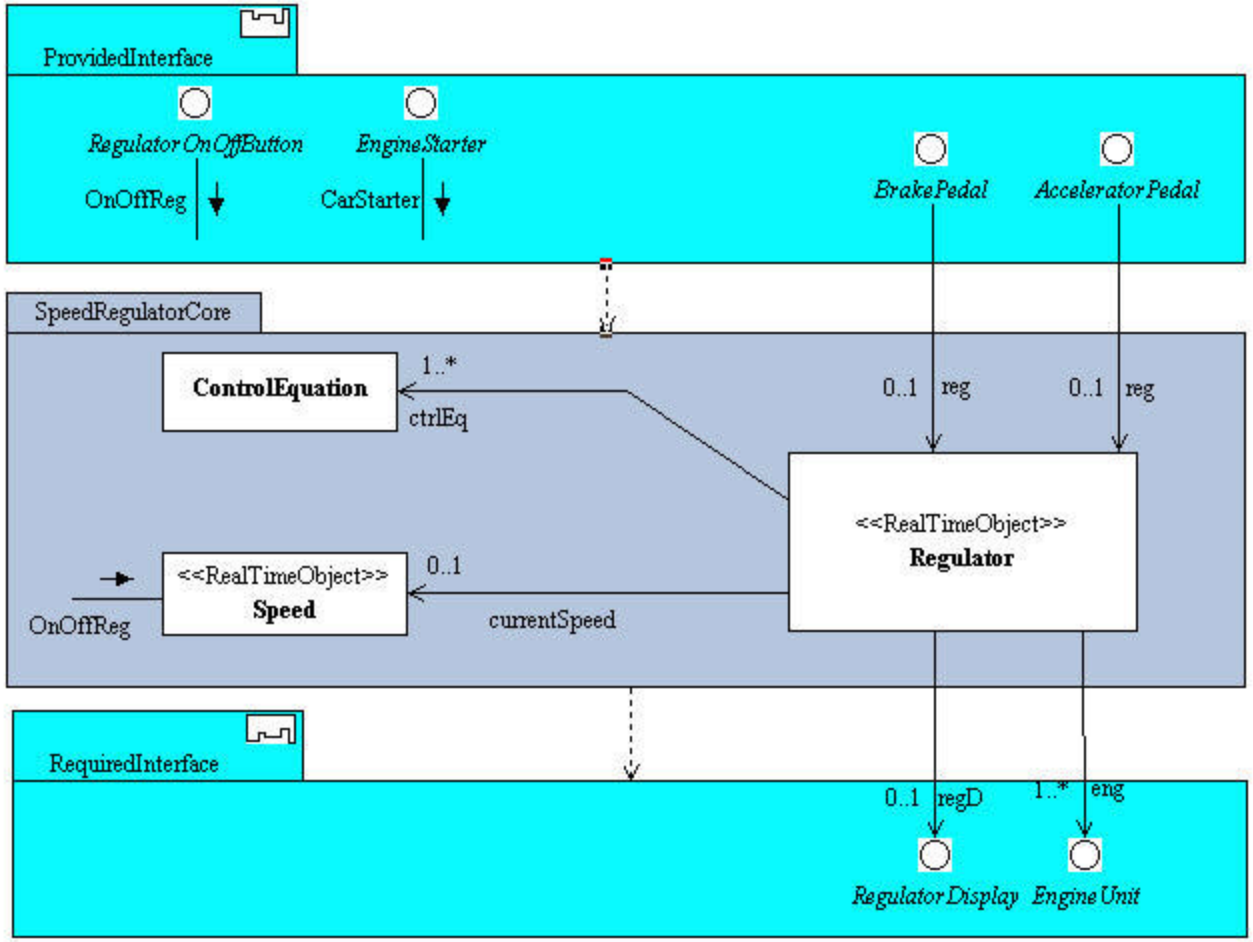
File d ’attente bornée !!!

```
Process Queue           { }
  ( ? msg ;
       dl ;
    ! diagnostique ;
  )
  pragma_c « …obj.queue() »
```

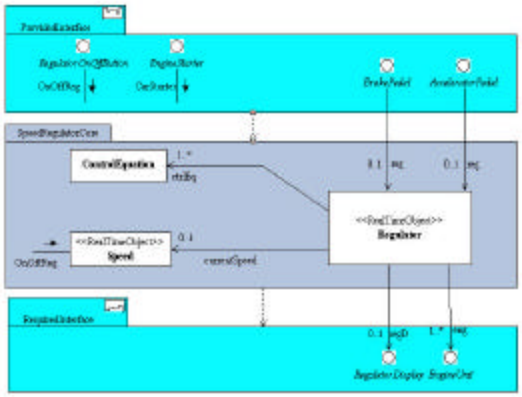
```
Process GetMsg          { }
      ( ?
        ! msg ;
          dl ;
          diagnostique ;
      )
      pragma_c « …obj.getMsg() »
```

- Implémentation externe de ces 2 processus en C

Architecture générale ACCORD/UML



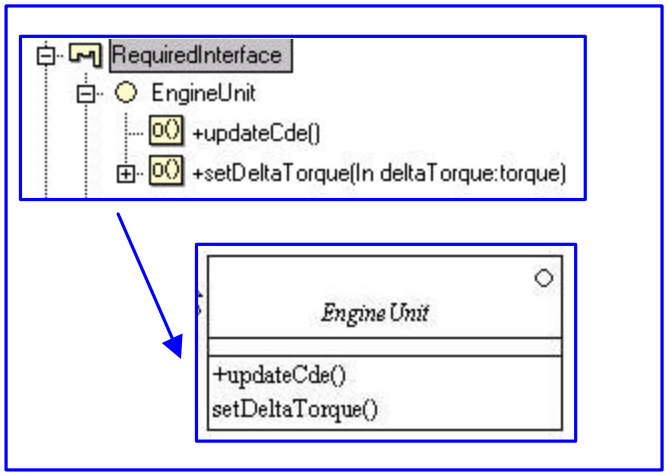
Principes généraux de traduction



```
process main_SpeedRegulator =
 ( ? event Tick;
       event OnOffReg, …;
   !
 )
 (| ProvidedInterface_exec(OnOffReg, …)
  | ReadyToExec_Speed := (when first_msg_Speed) ^+ end_Step_Speed
  | clk_getMsg_Speed := shift_event(ReadyToExec_Speed, Tick)
  | (EvtToExec_Speed, OK_get_Speed) := getMsg_Speed(clk_getMsg_Speed)
  | (end_Step_Speed, Speed_nextState) := chart_precond_Speed(EvtToExec_Speed,…)
  …
  idem pour la classe Regulator
  |)
 where
event ReadyToExec_Speed, end_Step_Speed, clk_getMsg_Speed;
boolean first_msg_Speed, OK_get_Speed;
string EvtToExec_Speed;
…
 use ProvidedInterface;
 use BAL_Speed;
 …
 use RequiredInterface;
 use Speed;
end;
```

Modélisation des interfaces

- Interfaces requises
  - Communications du système vers l'extérieur
  - 1 opération → 1 process SIGNAL
  - Ces process sont rassemblés dans un module SIGNAL



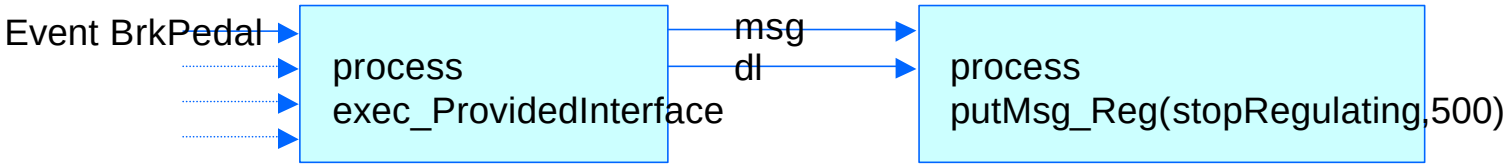
Interface requise du modèle UML

```
module RequiredInterface =
process setDeltaTorque =
 ( ? event clk_setDeltaTorque;
      dreal newDeltaTorque;
   !  )
 spec
 (|   clk_setDeltaTorque ^= newDeltaTorque
  |)
…
end;
```

Traduction SIGNAL

Modélisation des interfaces

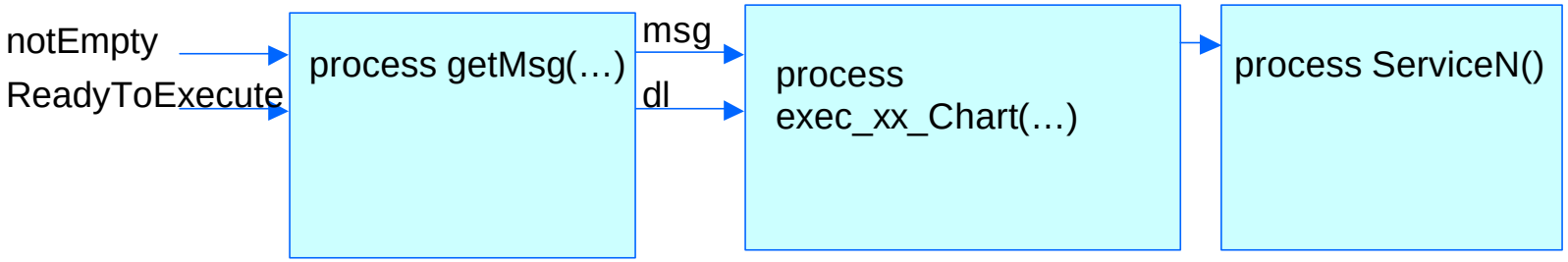
- Interfaces fournies
  - « Récolte » les évènements extérieurs
  - Ajoute des contraintes temporelles à ces évènements
  - Ecrit le message dans la file d 'attente de l 'objet



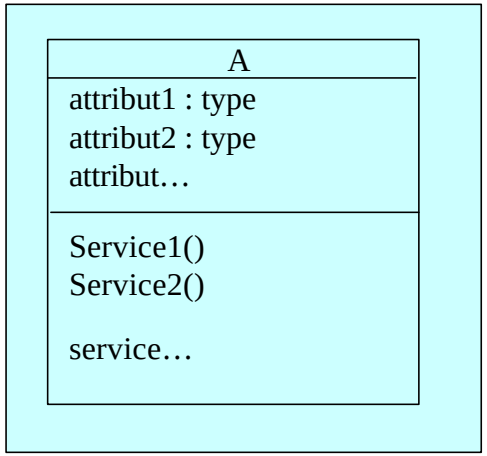
- SIGNAL prend en compte les évènements simultanés
  - Suréchantillonnage de l'horloge d'arrivée des évènements
  - Écriture séquentielle des messages dans les files d'attente

Extraction des messages

- Vider la boite aux lettres : getMsg_obj
  - Boite non vide
  - ReadyToExecute = True <=> Le pas RTC précédent est terminé



Modélisation d'un objet temps réel

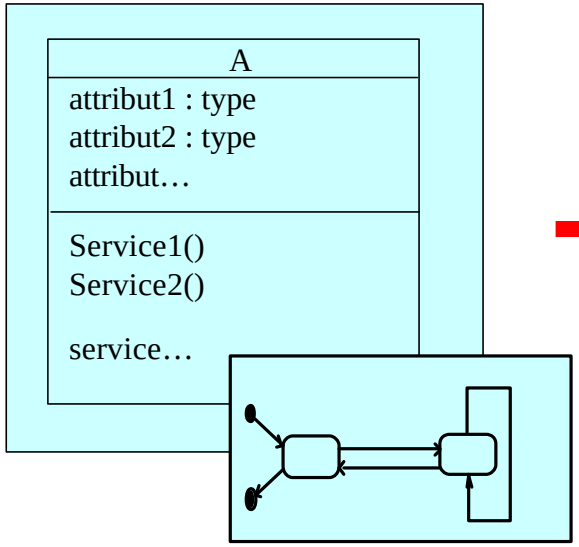


Exemple de classe

```
Module A =
(?…
  !…
)
process service1 =
(|…
 |)
process service2 =
(|…
 |)
……..
```

```
process main =
(|…
|)
where
  statevar type attribut1;
  ….
```

Modélisation d'un objet temps réel



Exemple de classe

Module A =

...

process service1 = ...

process service2 = …

process exec_A_Chart =…

Vérifie que le msg fait référence à une transition tirable, exécute le processus associé.

Met à jour l 'état courant et le booléen ReadyToExec (Pas RTC).

process main =

…

where

type AStates = enum(state1,…) ;

AStates ACurrentState ;

boolean AReadyToExec ;

….

Étapes à venir

- Transformation SIGNAL

  - Formaliser la transformation ACCORD/UML → SIGNAL
  - Définir la transformation ASL → SIGNAL
  - Analyser des capacités de preuve sur le modèle SIGNAL
  - Évaluer de l'approche sur les applications MBDA et SITIA
  - Définir les actions de standardisation à l'OMG
    - Ex : intégration modélisation et sémantique synchrone à UML 2

Étapes à venir (suite)

- Transformation SynDEx
  - Maj avec SynDEx V6
  - Enrichir le modèle d'architecture matérielle
    - infos sur les temps de communication
    - infos sur les temps d'execution
  - Formaliser la transformation UML → SynDEx
  - Analyser les capacités de placement sur le modèle SynDEx
  - Définir les actions de standardisation à l'OMG
    - Ex : actions sur les langages de description d'architecture en UML