Les langages de description d'architecture pour le temps réel

Anne-Marie Déplanche, Sébastien Faucou

IRCCyN (UMR n°6597)

Équipe “Systèmes Temps Réel”

prenom.nom@irccyn.ec-nantes.fr

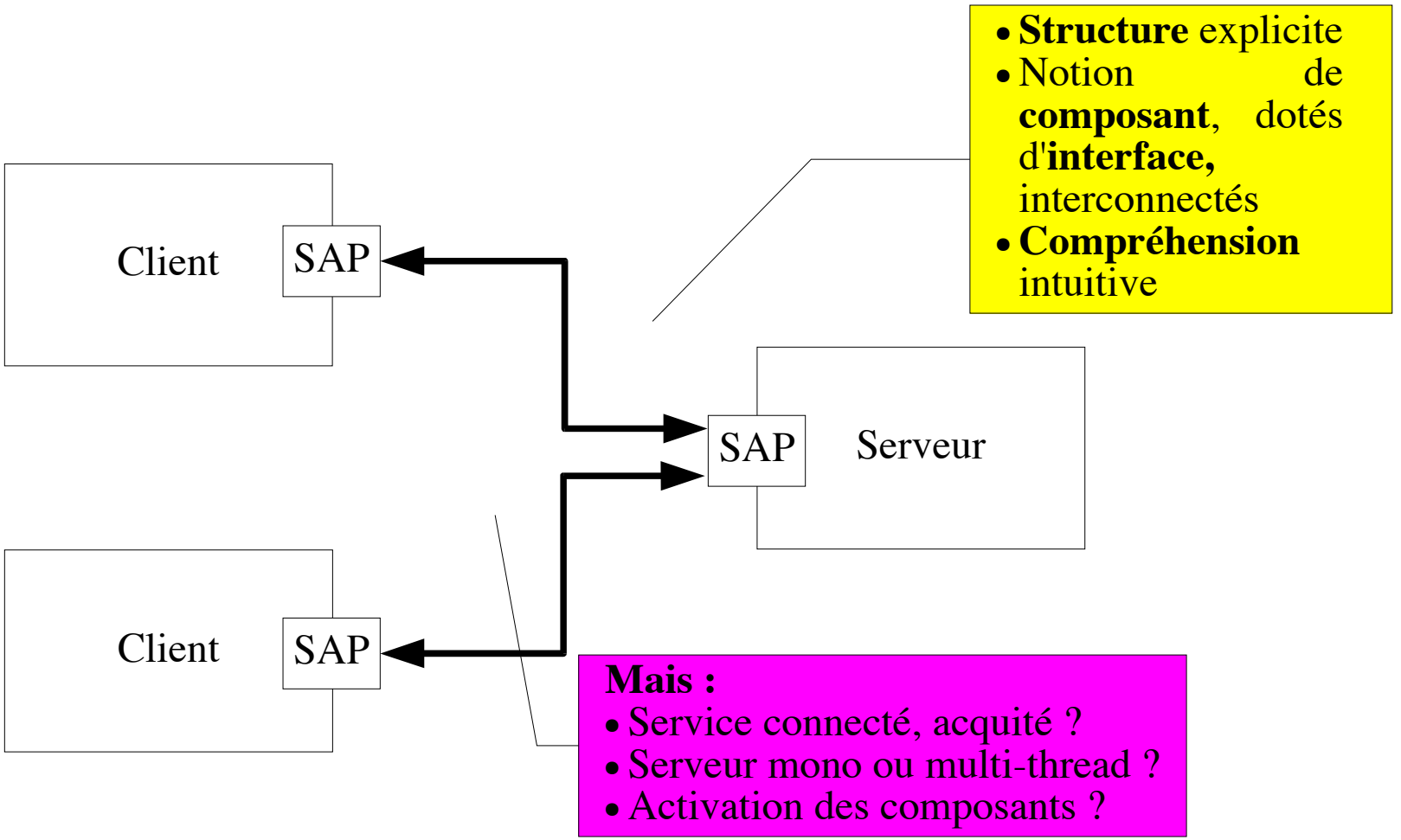
http://www.irccyn.ec-nantes.fr

Architecture logicielle ?

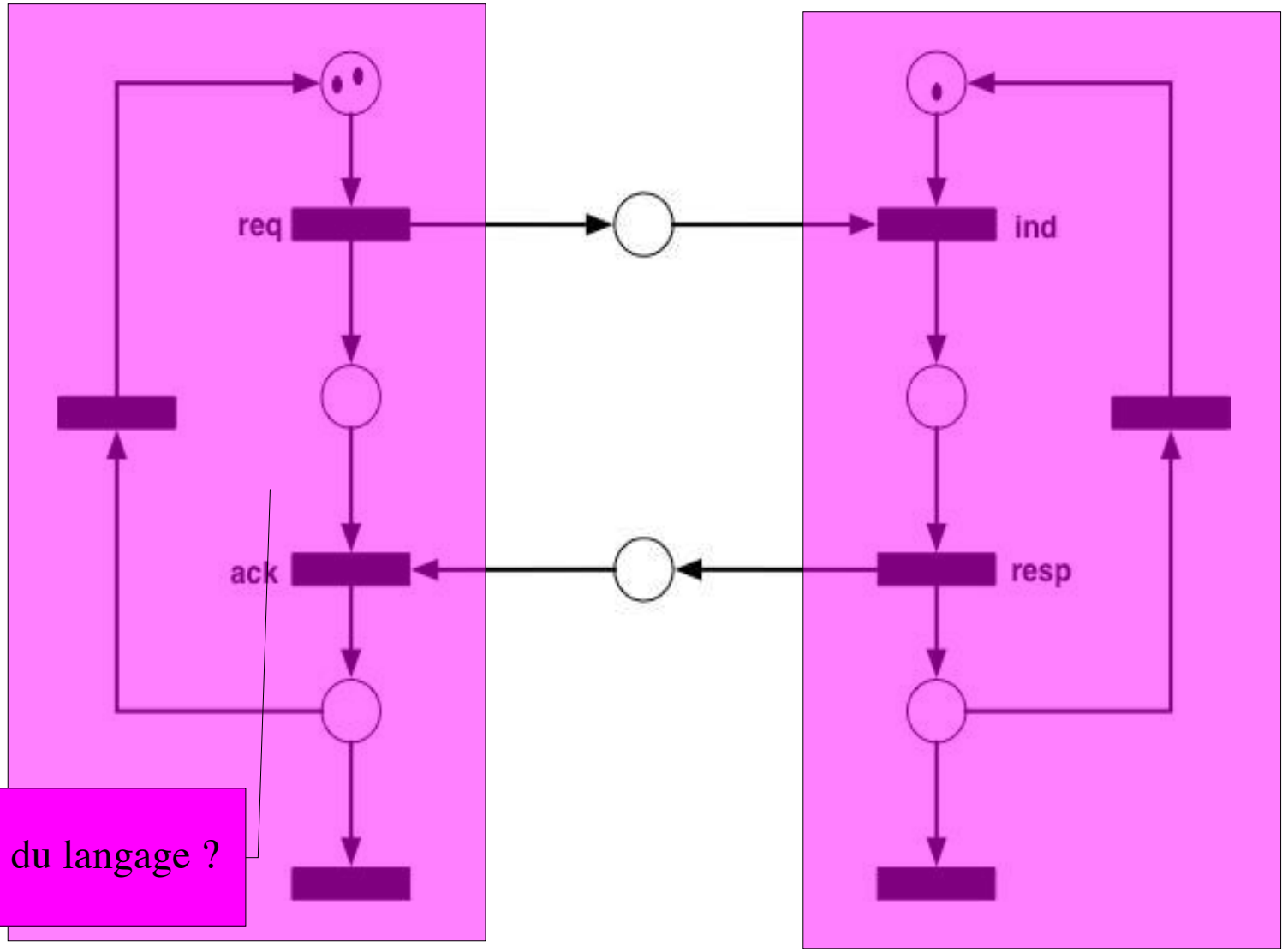
"... a view of the system that includes the system's major ***components****, the* ***behavior*** *of those components as visible to the rest of the system, and the ways in which the components* ***interact and coordinate*** *to achieve the system's mission" [7]*

Exemple (simpliste) : décrire l'architecture logicielle d'un système constitué de deux clients et un serveur. Le service est acquitté, sans connexion.

Avec des boîtes et des flèches



Avec un langage formel



- **Structure** ?
- **Modularité** du langage ?
- **Lisibilité** ?

Langage de description d'architecture logicielle ?

- Idée : langage (i.e. possédant une syntaxe et une sémantique) qui offre des abstractions et mécanismes adaptés à la modélisation de **l'architecture logicielle** d'un système
- Nom : ADL pour « *Architecture Description Language* »
- Objectifs :
  - lisible, analysable, manipulable, implémentable
  - modélisation de la structure et du comportement
  - adapté aux projets de grande taille (abstraction)
  - permettre la réutilisation des composants, des architectures, etc.

L'exemple en Wright [2]

- Notion de composants, de connecteurs **natives**
- Modèle **lisible et structuré**
- Réutilisation : type et instance
- Spécification (formelle) des **comportements**

```
Configuration Clients-Serveur

    Component Serveur
        Port SAP = req -> ack' Π §
        Computation = SAP.req -> SAP.ack' -> Computation Π §

    Component Client
        Port SAP = ind -> resp Π §
        Computation = SAP.ind' -> SAP.resp -> Computation Π §

    Connector Con_C_S
        Role Client = req' -> ack -> Client Π §
        Role Serveur = ind -> resp' -> Serveur Π §
        Glue = Client.req' -> Serveur.ind -> Serveur.resp' ->
            Client.ack -> Glue Π §
    Instances
        s : Serveur
        c1, c2 : Client
        c1_to_s, c2_to_s : Con_C_S
    Attachements
        c1.SAP as c1_to_s.Client
        c2.SAP as c2_to_s.Client
        s.SAP as c1_to_s.Serveur
        s.SAP as c2_to_s.Serveur

End Clients-Serveur
```

Plan de l'exposé

1. ADLs

- historique, caractéristiques, panorama

2. ADLs pour le Temps Réel

- intérêt, besoins spécifiques

3. Quelques exemples

- CLARA
- MetaH
- COTRE

4. Conclusions et perspectives

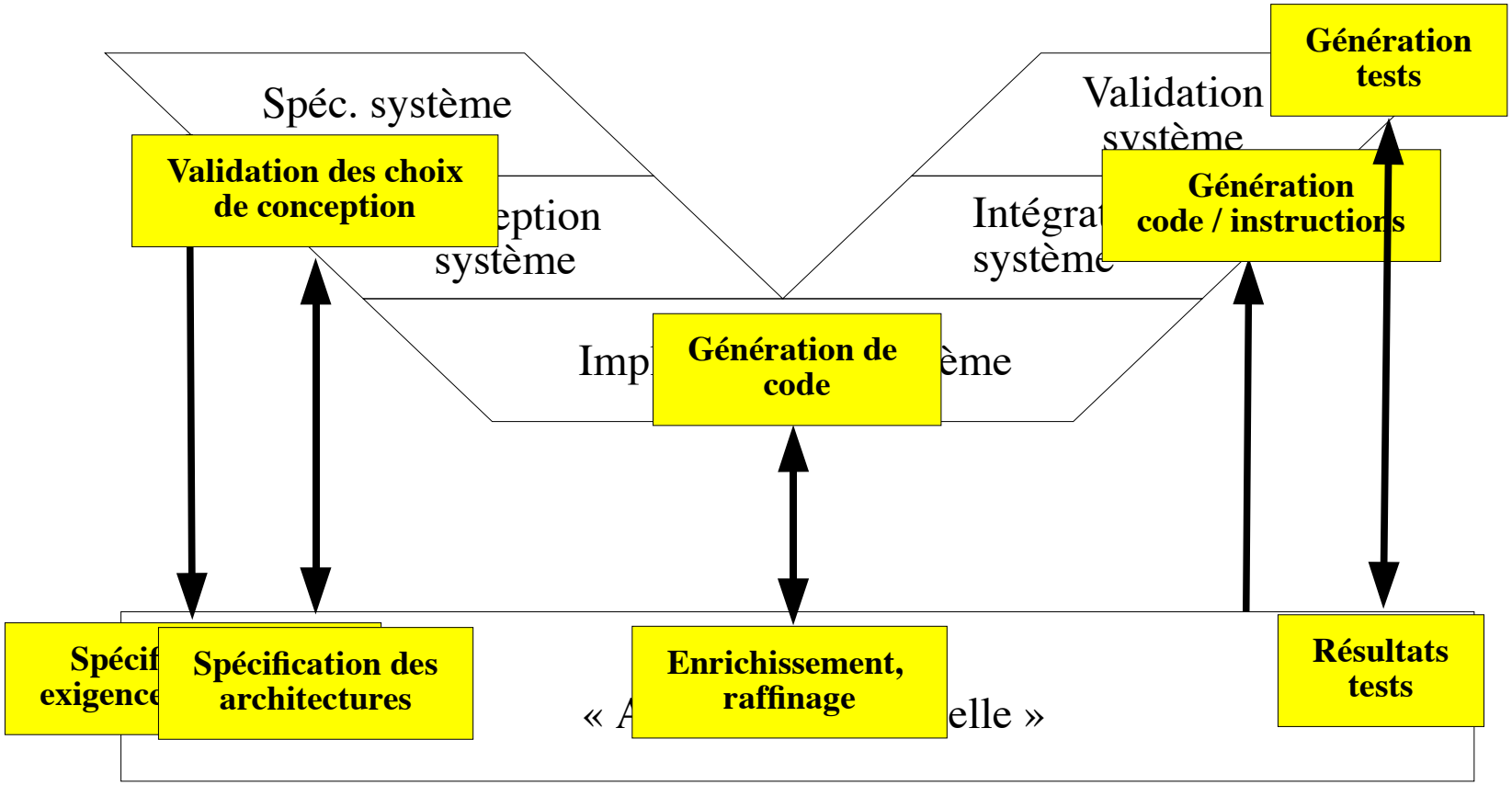
Un peu d'histoire

- 1976 : *« Programming-in-the-large vs. programming-in-the-small »*, de Remer et Kron
  - ➔ Mise en évidence de l'intérêt de la **conception de haut-niveau**
- 1993 : *« An introduction to software architecture »*, Garlan et Shaw
  - ➔L'étude des **architectures logicielles** est présentée comme une discipline à part entière
- 1998 : *« A classification and comparison framework for software architecture description language »*, Medvidovic et Taylor
  - ➔Identification des caractéristiques des **ADLs**

Architecture logicielle : quelles motivations ?

- Maîtriser la **complexité** du système
  - ➔ Vision globale, modèle structuré, hiérarchique
- Servir de **racine commune** pour :
  - ➔ Conception détaillée et implantation
  - ➔ Estimation des coûts et gestion de projet
  - ➔ Analyses et validation
  - ➔ Évolutions
- Support pour la **réutilisabilité** des systèmes logiciels
  - ➔ Des composants, des sous-systèmes, des « principes » de conception, etc.

Place dans le cycle de vie



Comment reconnaître un ADL ?

Langage de modélisation, basé sur le tryptique (et demi)

- **Composant**
- **Connecteur**
- **Configuration**
- ... et **Interface**

C'est le « minimum vital » ...

Composant ?

- Représente une entité logiciel de traitement ou de stockage répondant à un **besoin fonctionnel / applicatif**
  - ➔Atomique ou Composé
  - ➔Manipulable via son (ses ?) **Interface(s)**
  - ➔Dont l'état **observé** évolue en réaction aux sollicitations émises par son environnement
  - ➔Pour laquelle on spécifie des **propriétés** fonctionnelles et extra-fonctionnelles (offertes et requises)

Abstraction / Encapsulation

Vue externe

Analyse, documentation

Interface ?

Ensemble de points d'interaction d'un composant ou d'un connecteur avec son environnement, généralement orientés

- S.A.P., port (en entrée, en sortie), service (offert, requis), etc.

+

Contraintes et propriétés définissant les conditions d'utilisation de l'entité via son interface (= **contrat**)

Connecteur ?

- Composant spécialisé dans l'**interaction**, ne répondant pas directement à un besoin fonctionnel / applicatif
  - ➔ Un connecteur propose des **rôles** aux composants (ex : producteur, consommateur, souscripteur, etc.)
  - ➔ Le connecteur est le « metteur en scène » (**glue**)
  - ➔ Entité de même rang que les composants applicatifs (idée issue de la « communauté ADL » ... cf. UML2)

Notion de compatibilité port / rôle

Pas nécessairement une unité identifiable au niveau implémentation

Configuration ?

- Graphe de composants et de connecteurs spécifiant tout ou partie d'une architecture selon un certain point de vue
  - ➔ Doit respecter des contraintes syntaxiques, sémantiques, spécifiques
  - ➔ Unité de V&V
  - ➔ Encapsulable dans un composant (non atomique) et réutilisable

Exemples de « point de vue » :
- **flot de données**
- **flot de contrôle**
- **répartition et messagerie**
- **etc.**

Différenciation entre ADLs

- Focus sur
  - ➔ analyse, génération de code, conception, etc.
- « *implementation constraining* » ou « *implementation independant* »
  - ➔ selon que l'ADL pré-suppose ou non l'existence de certains services au sein du support d'exécution
- Modélisation des connecteurs
  - ➔ Connecteurs explicites ou implicites, liste figée ou extensible par l'architecte, etc.
- etc.

Fiche d'identité détaillée [32]

Modélisation des architectures :

- **Composants : Interface**, Typage, Sémantique, Contraintes, Évolution, Propriétés extra-fonctionnelles
- **Connecteurs : Interface**, Typage, Sémantique, Contraintes, Évolution, Propriétés extra-fonctionnelles
- **Configurations :** Lisibilité, Compositionnalité, Raffinage et traçabilité, Hétérogénéité, « Scalabilité », Évolutivité, Dynamisme, Contraintes, Propriétés extra-fonctionnelles

Outils support :

- Spécification active, Vues multiples, Analyse, Raffinement, Génération d'implémentation, Dynamisme

Quelques ADLs académiques

Nom	Focus	Réf	Origine
Wright	Modélisation et analyse du comportement dynamique des systèmes concurrents	[2]	CMU
Darwin	Système massivement distribué	[28]	ICL
Rapide	Modélisation et simulation du comportement dynamique des architectures		
UniCon	Génération de code de liaison pour l'interconnexion de composants pré-existants	[42]	CMU
ACME	Langage pivot		CMU
MetaH	Conception, validation et génération d'applications temps réel embarquées (avionique)	[39]	Honeywell

Plan de l'exposé

1. ADLs

- historique, caractéristiques, panorama

2. ADLs pour le Temps Réel

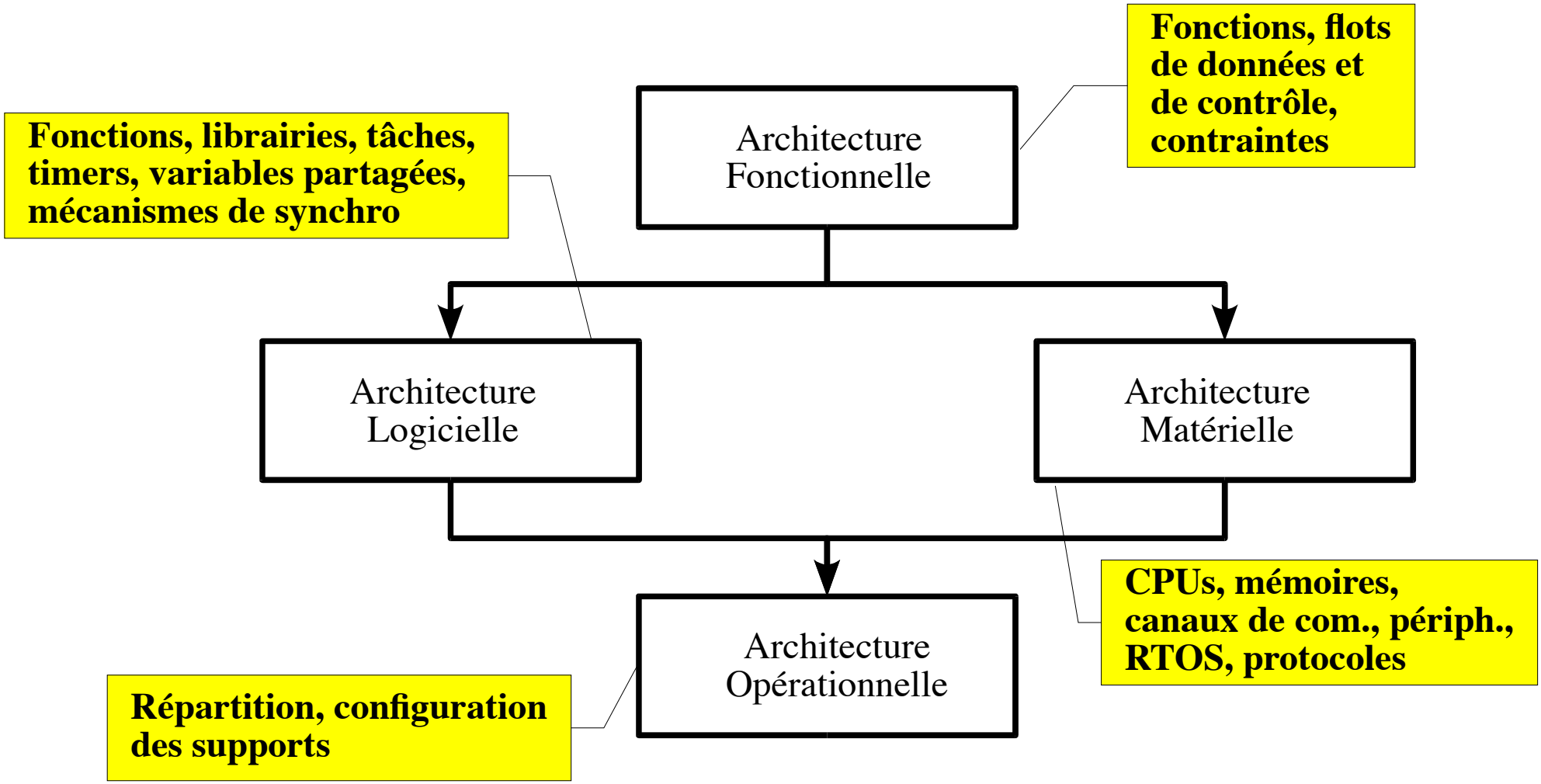
- intérêt, besoins spécifiques

3. Quelques exemples

- CLARA
- MetaH
- COTRE

4. Conclusions et perspectives

Conception des STR



Intérêt des ADLs pour la conception des STR

- **Maîtrise de la complexité** fonctionnelle et extra-fonctionnelle
- **Socle commun** pour la prise en compte de différents aspects : réactivité, ordonnancement et distribution, sûreté de fonctionnement, etc.
- Et de différentes activités : conception, V&V, codage, déploiement, etc.
- Offre une **vision globale** tôt dans le processus de développement : nécessaire pour la **prise en compte des aspects opérationnels**

Prise en compte des spécificités du domaine ?

ADL Temps Réel = ADL conçu pour permettre la prise en compte des spécificités des STR

Quelles spécificités ?

- Réactivité et environnement physique
- Temps
- Multiples architectures (AF, AL, AM, AO)
- Modes de fonctionnement
- Liens avec les outils de V&V

Réactivité et environnement physique

- Plusieurs fonctionnalités à assurer simultanément
  - ➔ activées pour répondre aux besoins du procédé
  - ➔ dont l'exécution affecte l'état du procédé
  - ➔ exécutées sur le même support d'exécution (pb de partage de ressource sous contraintes)
  - ➔ parfois interdépendantes (communication et synchronisation)

Un ADLTR doit permettre la **description non ambigüe des flots de contrôle** qui traversent le système (= système de pilotage + procédé)

Temps

- Spécification de flots de contrôle périodiques, de chiens de garde, etc.
- Quantification des **délais** d'exécution des traitements, d'acheminement des messages, etc.
  - ➔ estimations a priori puis exigences puis mesures puis ...
- Expression des **contraintes temporelles**
  - ➔ aide au raffinement / traçabilité au cours du développement du système

Les aspects liés au temps doivent également être pris en compte par les **outils** et le **processus de développement** associé à l'ADLTR

AF, AL, AM, AO ? (1)

- Les ADLs non temps réel sont de niveau AF (majorité) ou AL :
  - Conception de l'AM = problème réglé (hors champ)
  - AO prise en compte via attributs des composant (ex : noeud hôte)
- **Insuffisant** pour la description d'un STR :
  - Conception de l'AM = **problème à résoudre** : lien avec procédé, dimensionnement sous contrainte (de coût, d'embarquabilité, de perf., de SdF, etc.)
  - Conception de l'AO = **problème à résoudre :** plusieurs AO pour 1 couple (AL,AM), recherche d'une solution optimale

AF, AL, AM, AO ? (2)

- Description de l'**AM** :
  - Composants = CPU + RTOS, mémoire, périphériques
  - Connecteurs = canaux de communication + protocoles
  - Configuration = topologie du système
  - Il faut **quantifier / décrire** : puissance de calcul, politique d'ordo., bande passante, protocole MAC, taux d'erreurs, etc.
- Description de l'**AO** :
  - Projection de l'AL sur l'AM : distribution des composants de l'AL, configuration des composants systèmes de l'AO, liens entre év. matériels et logiciels, etc.

Modes de fonctionnement

- Les ADLs non temps réel traitent parfois la reconfiguration dynamique des architectures
  - ➔ Le + souvent au niveau du composant
- La notion de « mode de fonctionnement » nécessite elle une approche **au niveau des configurations** :
  - ➔ un mode englobe une configuration
  - ➔ spécification des conditions / stratégies de commutation entre modes concurrents au niveau du composant père

Outillage

1)Systèmes temps réel **= systèmes critiques** : fortes exigences en terme de V&V

2)Les performances « temps réel » sont fortement **conditionnées par les choix de conception architecturale**

3)Niveau architectural adapté pour l'exploration de l'**espace des solution**s (solutions peu coûteuses à construire)

1) + 2) + 3) => besoin d'outils pour guider et valider l'étape de conception des architectures

Comment fabriquer son propre ADLTR ?

- Conserver les **bonnes propriétés des ADLs** « classiques » : approche non ambigüe, adaptée aux projets de grande taille, etc.
- Se laisser guider par **les outils et techniques** de V&V de la communauté TR pour identifier les extensions et enrichissements nécessaires
- Proposer une **méthode de développement** centrée sur l'architecture, associant les différents outils et techniques en vue de produire une architecture opérationnelle validée et optimisée

Plan de l'exposé

1. ADLs

- historique, caractéristiques, panorama

2. ADLs pour le Temps Réel

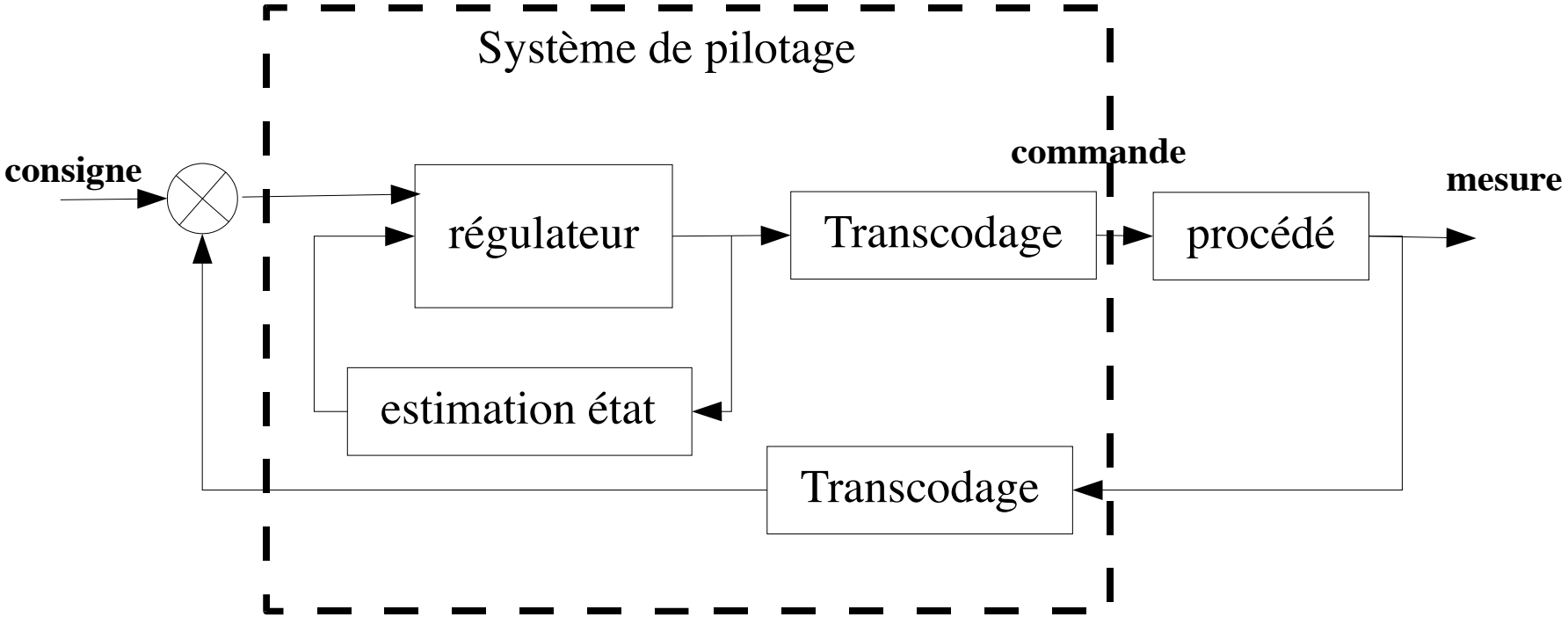
- intérêt, besoins spécifiques

3. Quelques exemples

- CLARA
- MetaH
- COTRE

4. Conclusions et perspectives

Cahier des charges (1)

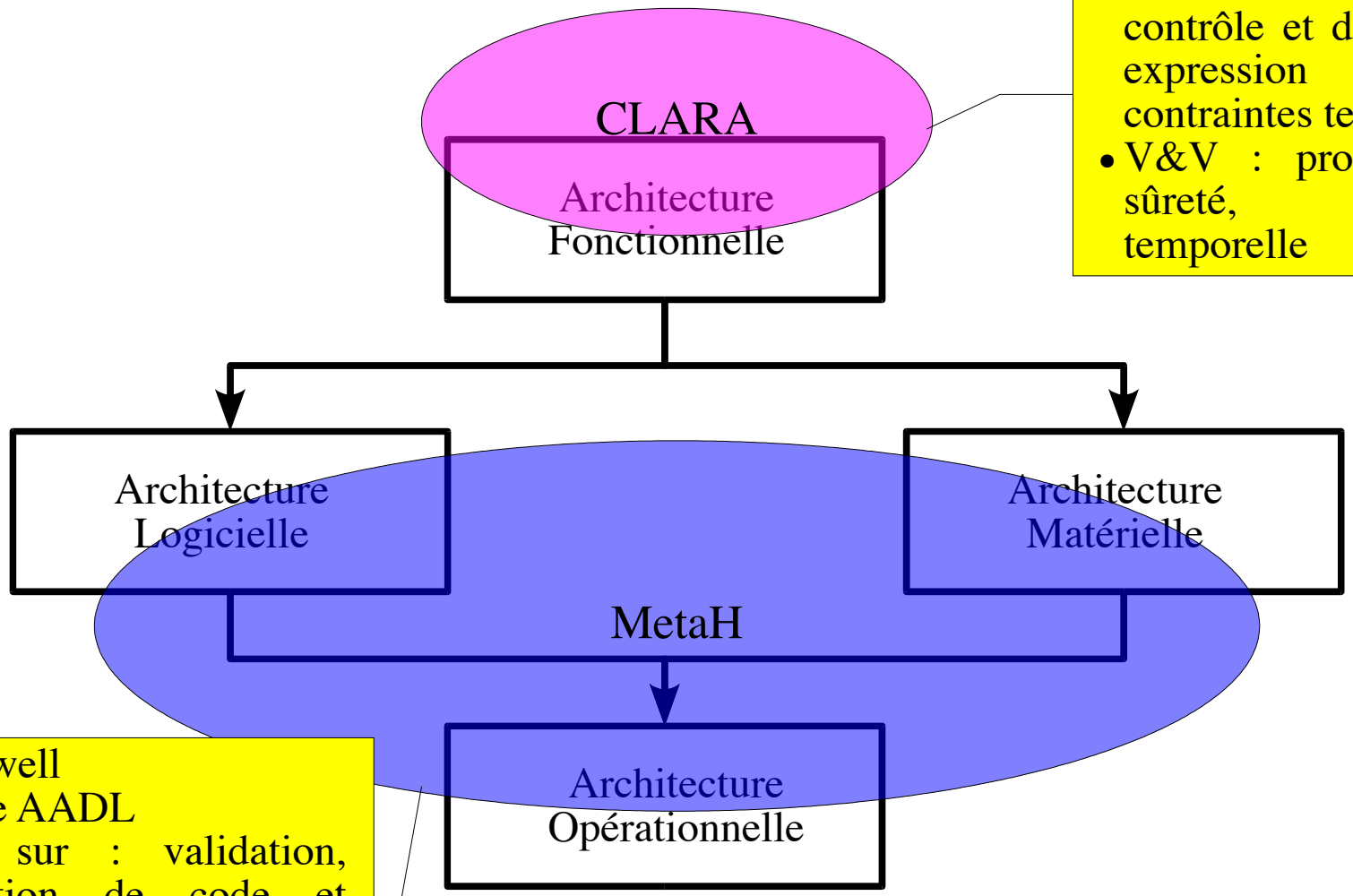


Cahier des charges (2)

Commande échantillonnée :

- Période = 500 ms ± 20 ms
- Échéance de bout-en-bout « mesure - commande » = 250 ms
- Architecture matérielle monoprocesseur

CLARA et MetaH



- IRCCyN
- Focus sur : flots de contrôle et de données, expression des contraintes temporelles
- V&V : propriétés de sûreté, cohérence temporelle

- Honeywell
- Base de AADL
- Focus sur : validation, intégration de code et génération des images binaires
- V&V : ordonnançabilité, SdF

Description en CLARA : définition des types composants (

- Activité **atomique**
- **Comportement commun** aux différentes instances du type
- **Séquence** d'appel à des primitives « CLARA » (send, receive, signal, wait) et des fonctions utilisateurs
- Remplacé par une **sous-config. dans le cas d'une activité composée**

- Déclaration de l'**interface d'échange**
- **Ports orientés et typés** (types « utilisateurs »)
- **interface de contrôle implicite** (ports START et END)

```
TYPE_ACTIVITY T_CalculCommande;

  VAR_IN   etat, consigne : t_etat;
           mesure : t_cmd_in;
  VAR_OUT  commande : t_cmd_out;

  BEHAVIOR
     receive(mesure) ->
     receive(etat) ->
     receive(consigne) ->
     call(calculCmd, 10ms, 20ms) ->
     send(commande);

END_ACTIVITY;
```

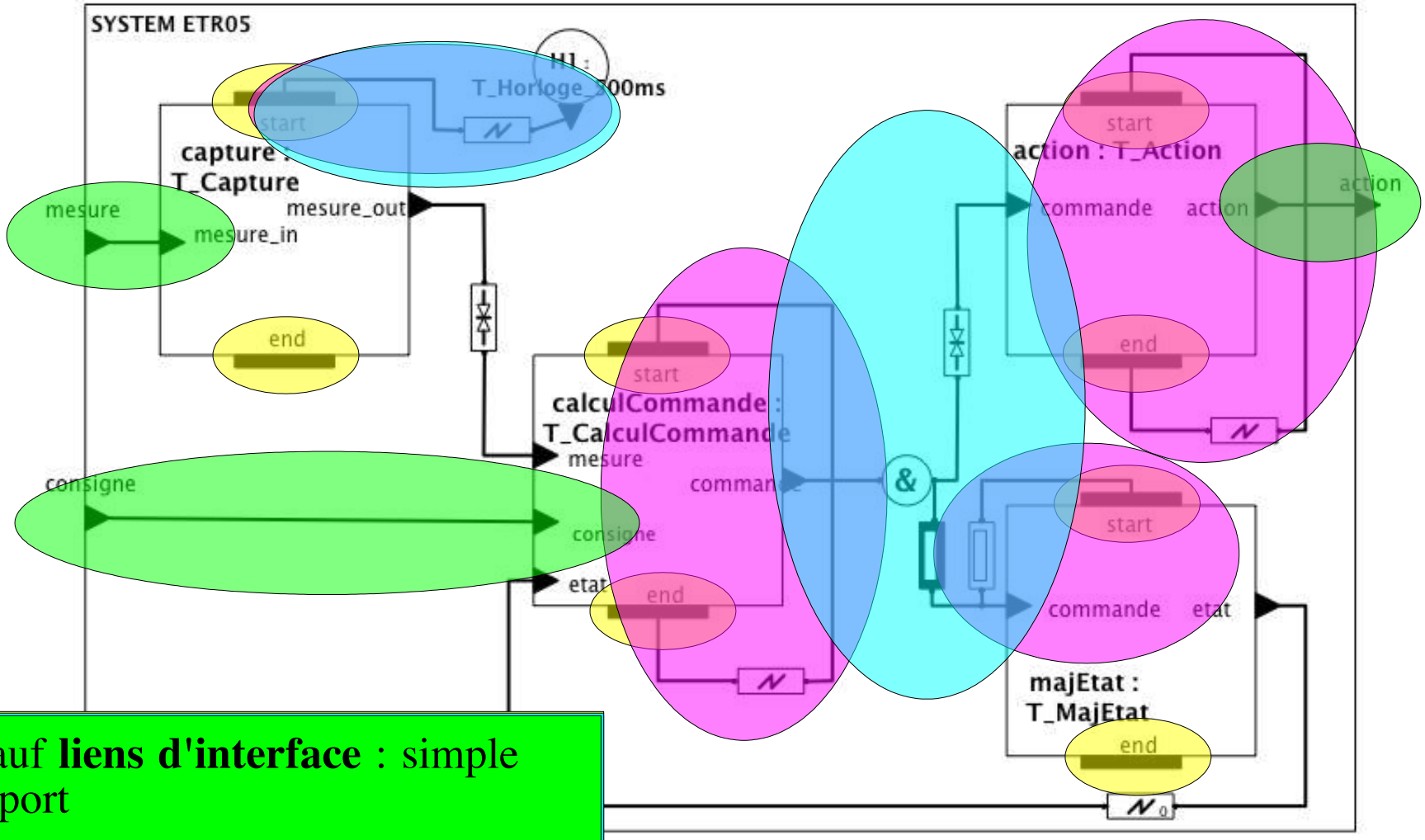
Description en CLARA : déclaration des types de composant (2)

```
TYPE_GENERATOR T_Horloge_500ms;
    SIGNAL_OUT out;
    BEHAVIOR clock;
    PERIODIC 500ms;
END_GENERATOR;
```

- Interface du générateur : port en sortie pour émettre un signal

- Générateur périodique (**horloge**)
- Autre comportement : sporadique, apériodique, etc.

Description en CLARA : la configuration (graphique)



Sauf **liens d'interface** : simple report

Description en CLARA : la configuration (textuel)

```
SYSTEM Etr05;
VAR_IN mesure : t_mesure; consigne : t_etat;
VAR_OUT action : t_action;
GENERATOR H1 : T_Horloge_500ms;
ACTIVITY
    capture : T_Capture;
    calculCommande : T_CalculCommande;
    majEtat : T_MajEtat;
    action : T_Action;
LINKS
   // liens d'interface
   mesure TO capture.mes
   ...
   // liens inter-activités
   capture.mesure_out TO calculCommande.mesure: RDVd;
   ...
   // liens d'activation
   H1.out TO capture.start : MEM;
   ...
CONSTRAINTS
    Abs(capture.mesure_in.cnf) < 25ms;
   480ms < capture.mesure_in.cnf < 520ms;
   0ms < capture.mesure_in.cnf : action.action.cnf < 250ms;
END_SYSTEM.
```

Spécification des contraintes de temps : absolue ou relative, portent sur des instances consécutives

Description en MetaH : déclaration des types de composant

```
process ICalculCommande is
   etat: in port ETR05.T_ETAT;
   consigne: in port ETR05.T_ETAT;
   mesure: in port ETR05.T_CMD_IN;
   commande: out port ETR05.T_CMD_OUT;
end IcalculCommande;
```

Définition d'un type d'**interface** : ports orientés et **typés**, événements en sortie

```
process implementation ICalculCommande.Default is
   calculCmd: subprogram;
paths
   <<Normal>>:=calculCmd;
attributes
   calculCmd'SourceTime:=20ms;
   self'ComputePath:=Normal;
end ICalculCommande.Default;
```

Définition d'une **implémentation** possible pour cette interface : sous-composant (ici **subprogram**) et flots de contrôle

Description en MetaH : AL

```
macro Etr05 is
   mesure: in port ETR05.T_MESURE;
   ...
end Etr05;

macro implementation Etr05.Default is
   capture: periodic process ICapture.Default;
   calculCommande: periodic process ICalculCommande.Default
   ...
connections
   capture.mesure_in <- mesure;
   action <<- action.action;
   ...
   calculCommande.mesure <<- capture.mesure_out;
   calculCommande.etat <- majEtat.etat;
   ...
attributes
   majEtat'Period:= 500ms;
   ...
end Etr05.Default;
```

- Macro : composant « **structurant** »
- Implémentation donnée par une configuration

Deux « **connecteurs** » :
- Transfert **immédiat** (terminaison processs producteur)
- Transfert **retardé** (fin période process producteur)

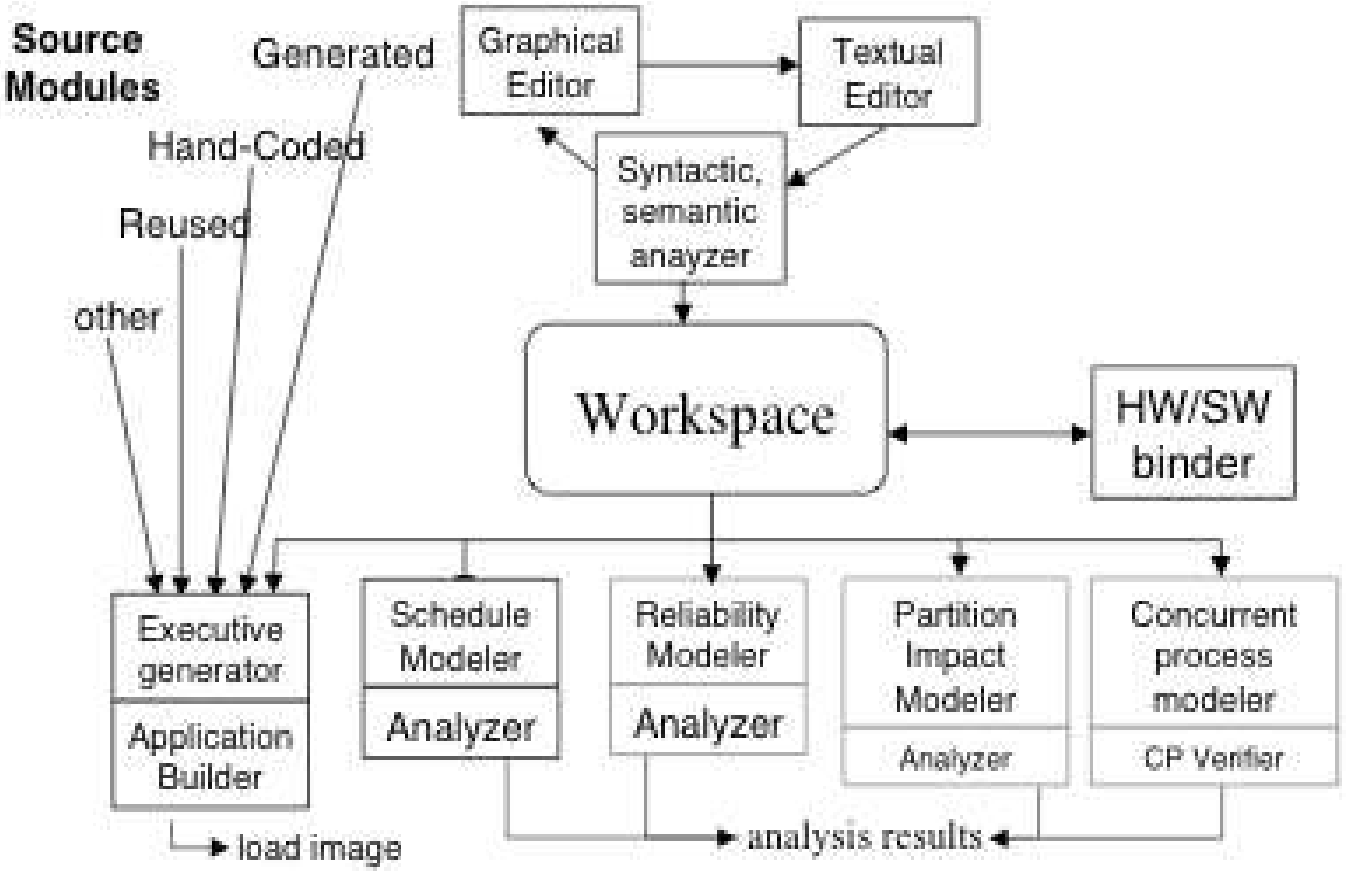
Description en MetaH : AO

Mapping des composants de l'AL sur ceux de l'AM

```
application Etr05 is

   macro Etr05.Default on processor MPC565.Default

connections
   Etr05.mesure <- MPC565.port1;
   Etr05.consigne <- MPC565.port2;
   MPC565.port3 <<- Etr05.action;

attributes
   MPC565'ClockPeriodMax:= 64us;
end Etr05;
```

Mise en correspondance des interfaces : sources et puits appartiennent à l'AM

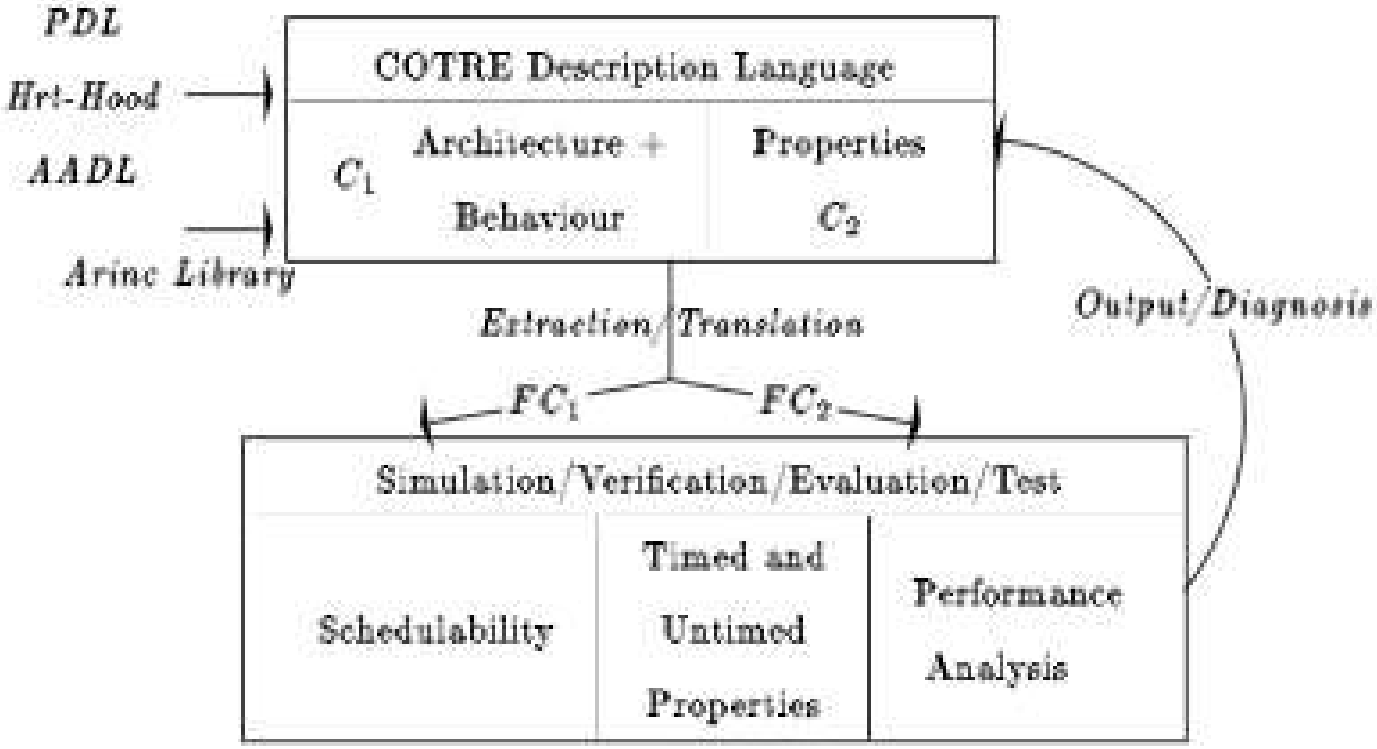
Aperçu de l'atelier autour de MetaH



COTRE (1)

- Composants Temps RÉel : projet RNTL (Féria, ENSTBr, Airbus, TNI)
- Objectifs : démarche et outils de modélisation et **validation** d'architecture de logiciels temps réel ds domaine avionique
- Point de départ : langages et notations utilisés dans l'industrie tels que HRT-HOOD ou AADL
- Apport : **ponts vers les outils de vérifications formelles**
- Évolution : annexe à AADL (description comportementale)

Cotre (2)


PDL
Hrt-Hood
AADL
Arinc Library
COTRE Description Language
Architecture +
Behaviour
C_1
Properties
C_2
Extraction/Translation
FC_1
FC_2
Output/Diagnosis
Simulation/Verification/Evaluation/Test
Schedulability
Timed and
Untimed
Properties
Performance
Analysis

Plan de l'exposé

1. ADLs

- historique, caractéristiques, panorama

2. ADLs pour le Temps Réel

- intérêt, besoins spécifiques

3. Quelques exemples

- CLARA
- MetaH
- COTRE

4. Conclusions et perspectives

Conclusions ?

- Embarqué et Temps Réel = **domaine d'application privilégié** pour l'approche « orientée architecture »
- Historiquement :
  - 90's : naissance, développement académique
  - 00's : industrialisation : EAST-EEA, **AADL**
- Aujourd'hui ?
  - Intégration d'outils, processus de développement (MDA ?)
  - Styles, pattern pour le temps réel et l'embarqué ?

Questions ?