Spécification, vérification et optimisation des systèmes temps réel distribués embarqués

Méthodologie
Adéquation Algorithme Architecture

Yves SOREL
Yves.Sorel@inria.fr

http://www-rocq.inria.fr/syndex

Applications

- **Transports : avionique, automobile (AEE, EAST)**
- **Robotique mobile (CyCab, SAFE)**
- **Traitement du signal : radar, sonar**
- **Télécommunications : UMTS, soc (PROMPT)**
- **Traitement d'images : contrôle qualité ...**
- **…**

Objectifs

- **Sécurité de conception**
- **Prototypage rapide**
- **Optimisation**
- **Génération de code de série**
- ***Réduction du cycle de développement***

Caractéristiques

- **Algorithmes** : contrôle-commande, TSI
- **Réactif** : *stimulus - opérations - réaction*
- **Temps réel** : temps de réaction borné
  contraintes : latence, cadence
- **Distribué** : puissance calcul, modularité, câblage réduit

⇨ ***Architecture multicomposant hétérogène***

  - Réseau processeurs + circuits intégrés spécifiques
  - Circuit intégré spécifique (ASIC, ASIP, FPGA, IP)

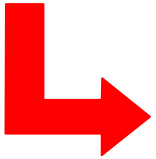
- **Embarqué** : minimisation des ressources

Méthodologie Adéquation Algorithme Architecture

- **Approche globale** fondée sur une extension de la sémantique des langages synchrones
- **Modèle unifié :** graphes factorisés
  - **A**lgorithme : opération / dépendance
  - **A**rchitecture : FSM / connexion
  - Implantation : transformations de graphes
- **Adéquation :** implantation optimisée
- **Macro-génération :**
  - Exécutifs pour multicomposant
  - VHDL structurel pour synthèse de circuit

Spécification Algorithme : hyper-graphe orienté

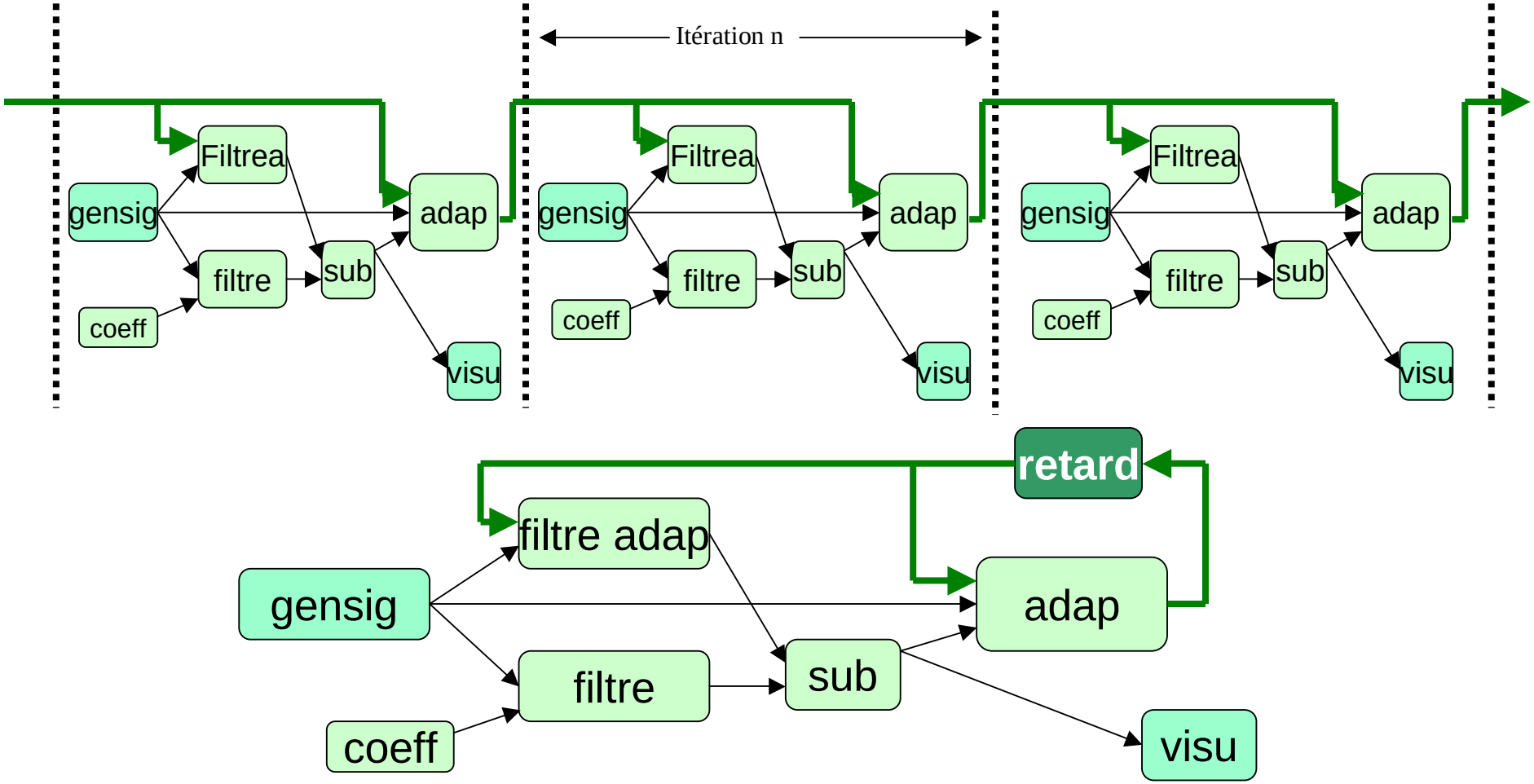
- **Sommet :** opération conditionnée ou non
- **Hyper-arc orienté :** dépendance de données (ordre partiel d'exécution) ou dépendance de conditionnement
- **Répétition infinie d'un sous-graphe :** réactif
- **Répétition finie d'un sous-graphe :** boucle



Factorisation = Graphe flot de données conditionné

- **Issu des Langages Synchrones (via DC), de Scicos, AVS, CamlFlow**

Exemple d'algorithme : égaliseur adaptatif



Vérification Algorithme

- **Langages Synchrones**
  - Esterel, Lustre, Signal, Statecharts ...
  - Spécification modulaire
  - Hors contraintes matérielles, indépendante des durées : temps logique
  - Réaction *simultanée avec* Stimulus
  - Vérifications :
    - Cycles de dépendance **avec** retard
    - Cohérence ordre sur événements
    - Vérification de propriétés temporelles logiques

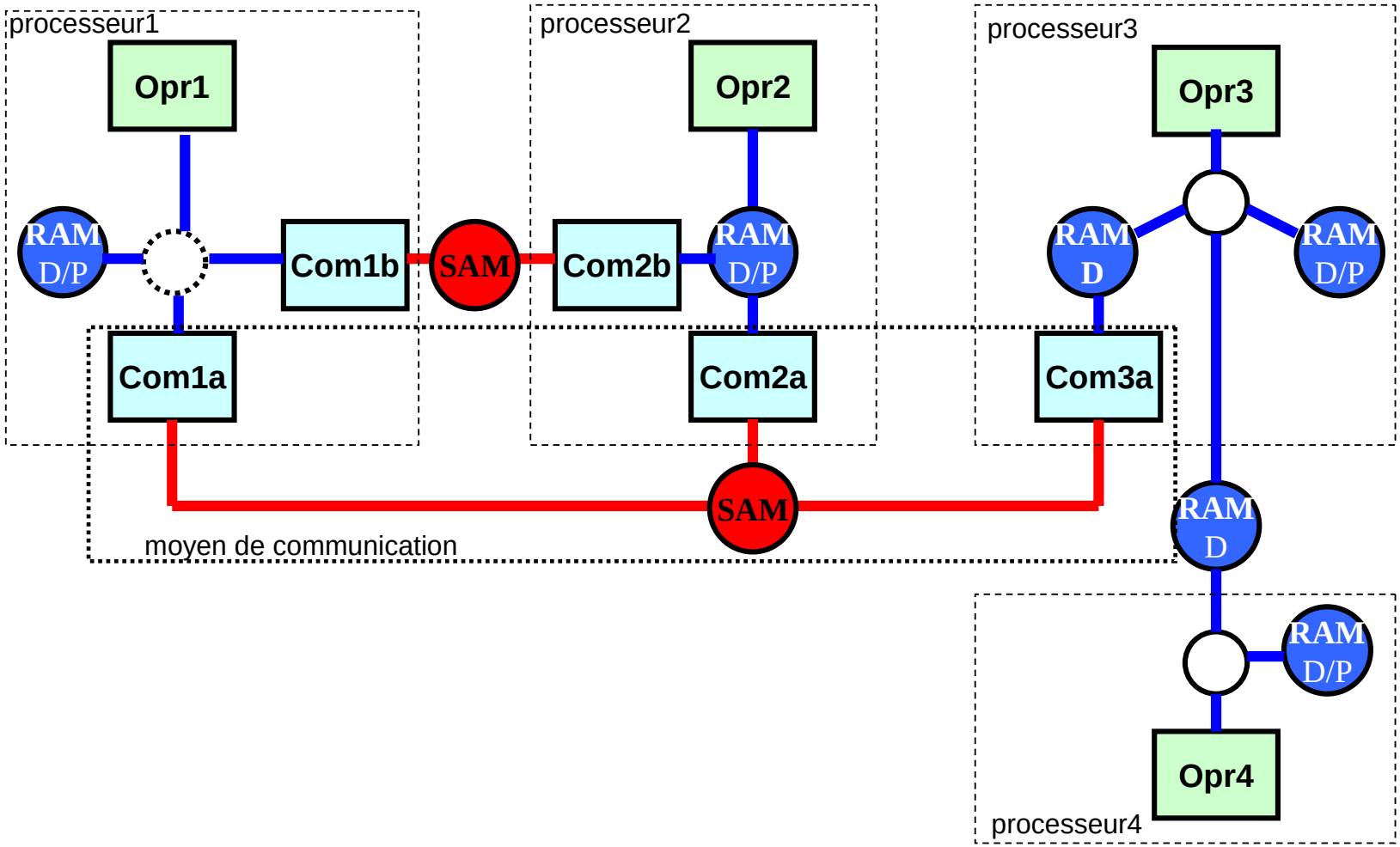
Spécification Architecture

- **Sommets (FSM)**
  - Opérateur : exécution séquentielle opérations de calcul
  - Communicateur : exécution séquentielle opérations comm.
  - Mémoire avec ou sans arbitre :
    - Accès aléatoire (RAM) : comm. non synchronisées
      - Programmes, données
    - Accès séquentiel (SAM) : comm. synchronisées, ordre R/W
      - Données
      - Point-à-point, multi-point avec ou sans diffusion matérielle
  - Bus/mux/démux avec ou sans arbitre :
    - Bus/mux/démux : sélection d'une mémoire parmi plusieurs
    - Bus/mux/démux/arbitre : arbitre l'accès aux mém. partagées

Spécification Architecture

- **Arcs orientés**
  - Connexions entre ces sommets pour modéliser les transferts de données
  - Connexions suivant un ensemble de règles :
    - Des opérateurs ne peuvent pas être connectés entre eux
    - Des communicateurs ne peuvent être pas connectés entre eux
    - Des mémoires ne peuvent pas être connectés entre elles
    - Des sommets bus/mux/demux avec ou sans arbitre peuvent être connectés entre eux
    - ...

Exemple d'Architecture



Implantation multicomposant

Toutes les implantations sont décrites en intention

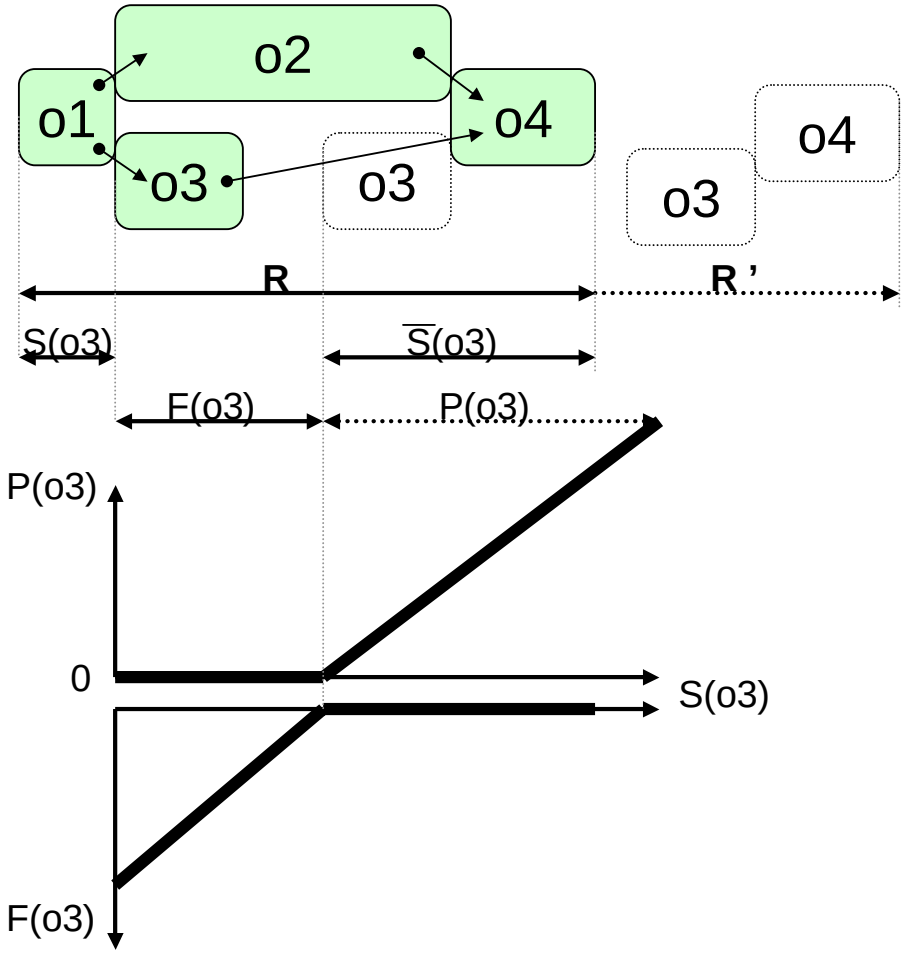
(Gal, Gar) —*rout o distrib o ordon*→ (Gal', Gar')

- **Routage** : création de toutes les routes inter-opérateur
- **Distribution** : *allocation spatiale* :
  - Partition et allocation des opérations sur les opérateurs
  - Partition des arcs inter-partition en fonction des routes
  - Création et allocation :
    - Sommets opérations de communication sur communicateurs
    - Sommets allocation sur mémoires
    - Sommets identité sur bus/mux/démux/ avec ou sans arbitres
- **Ordonnancement** : *allocation temporelle*
  - Ordre partiel → total : pour chaque partition opérations

Optimisation

- **Caractérisation opérations/opérateurs**
  - Mesures : durée, mémoire, interférences calculs/com.
- **Choix d'une implantation**
  - Qui respecte les contraintes temps réel et de distribution
  - Qui minimise les ressources
- **Distribution/ordonnancement :** hors ligne avec ou sans préemption
- **Problèmes NP-complet :** heuristiques
  - Rapides (prototypage) : gloutonne, ordonnancement liste
  - Lentes itératives : voisinage local, global

Optimisation latence=cadence, sans préemption



- Date de **début au plus tôt depuis le début** :

$$S(o_i) = \max_{\forall x_j \in pred(o)} E(x_j) \text{ (ou 0 si } pred(o_i) = \varnothing \text{)}$$

- Date de **fin au plus tôt depuis le début** :

$$E(o_i) = S(o_i) + \Delta(o_i)$$

- Date de **fin au plus tard depuis la fin** :

$$\overline{E}(o_i) = \max_{\forall x_j \in succ(o)} \overline{S}(x_j) \text{ (ou 0 si } succ(o_i) = \varnothing \text{)}$$

- Date de **début au plus tard depuis la fin** :

$$\overline{S}(o_i) = \overline{E}(o_i) + \Delta(o_i)$$

- **Flexibilité d'ordonnancement** :

$$F(o_i) = R - S(o_i) - \overline{S}(o_i)$$

- **Pénalité d'ordonnancement** :

$$P(o_i) = R - R'$$

- **Pression d'ordonnancement** :

$$\sigma(o_i) = P(o_i) - F(o_i)$$

Heuristique gloutonne ordonnancement liste

- Initialiser la liste de candidats avec les opérations sans prédécesseur :

$$Cand = \{o_i / pred(o_i) = \varnothing\}$$

- Tant que la liste n'est pas vide :

  ❶ pour chaque opération o_i de la liste rechercher le meilleur opérateur (avec prise en compte du coût des communications),

$$opr_{o_i} = \underset{\forall p_i \in Proc}{\mathbf{min}}\ \sigma(o_i, p_i)$$

  ❷ sélectionner l'opération la plus urgente à ordonnancer,

$$o_{urgente} = \underset{\forall o_i \in Cand}{\mathbf{max}}\ \sigma(o_i, opr_{o_i})$$

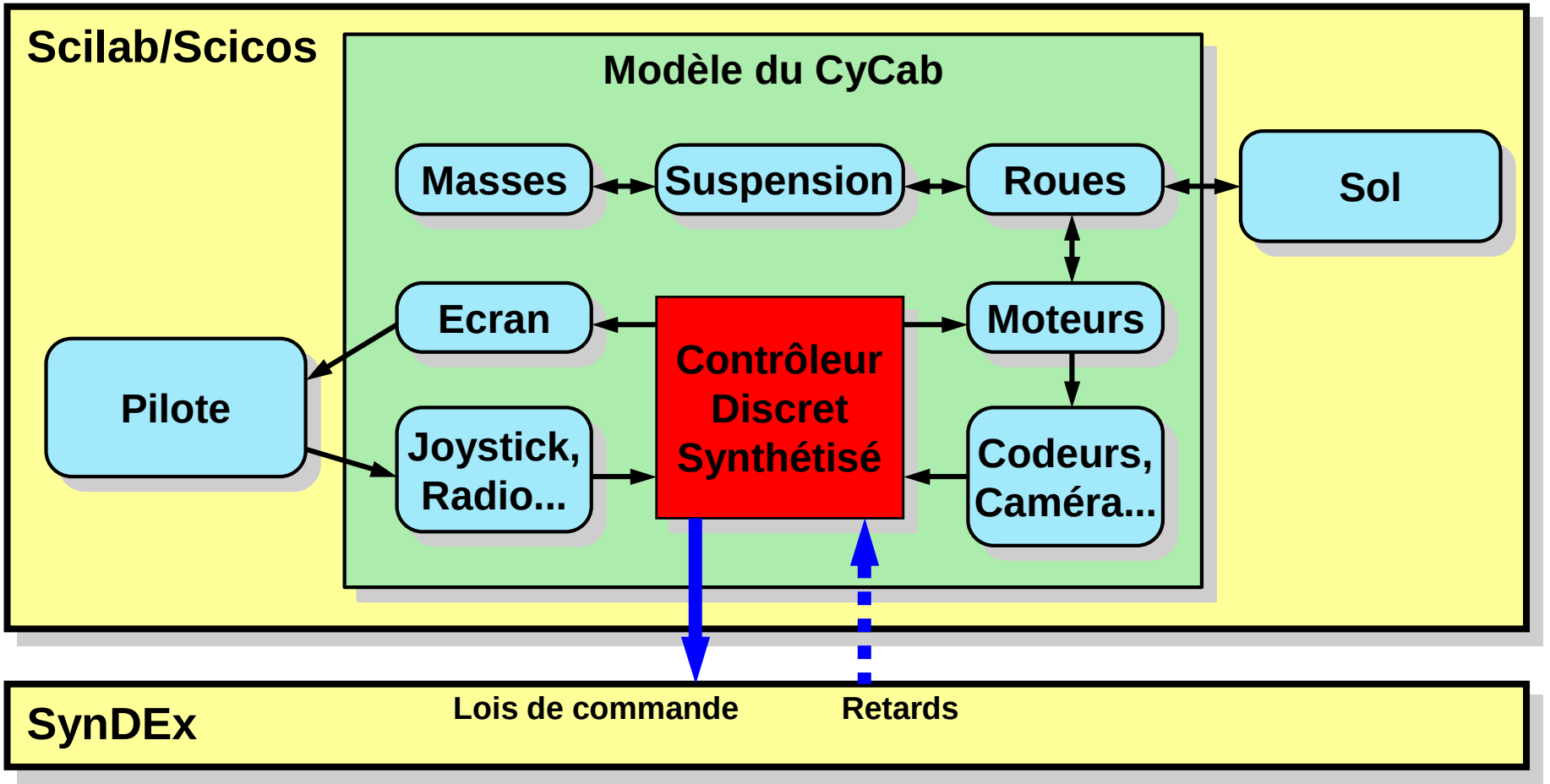
  ❸ retirer l'opération de la liste et ajouter tous ses successeurs devenus ordonnaçables,

$$Cand = Cand - o_{urgente} + Succ\_ordo(o_{urgente})$$

Logiciel CAO système : SynDEx

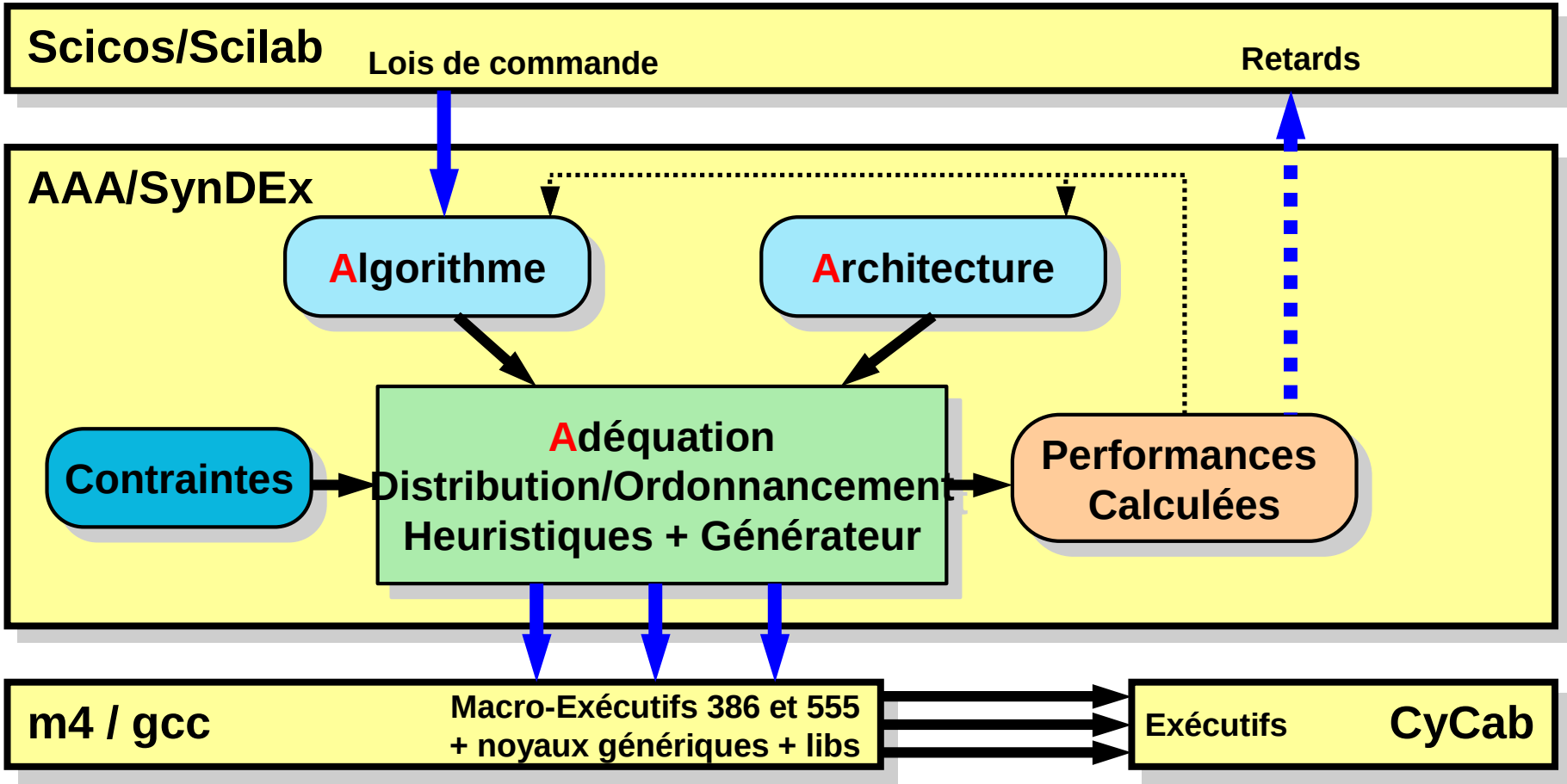
- **Supporte méthodologie AAA**
- **Interface avec Langages Synchrones**
- **Graphique interactif Unix X11 ou Windows**
- **Edition graphes algorithme et architecture**
- **Distribution et ordonnancement statiques**
- **Visualisation diagramme temps réel prédit**
- **Mesures performances temps réel**
- **Génération exécutifs temps réel distribués**

Modélisation / Simulation
Scilab / Scicos **Gratuit sur : www-rocq.inria.fr/scilab**



Implantation AAA / SynDEx

Gratuit sur : www-rocq.inria.fr/syndex



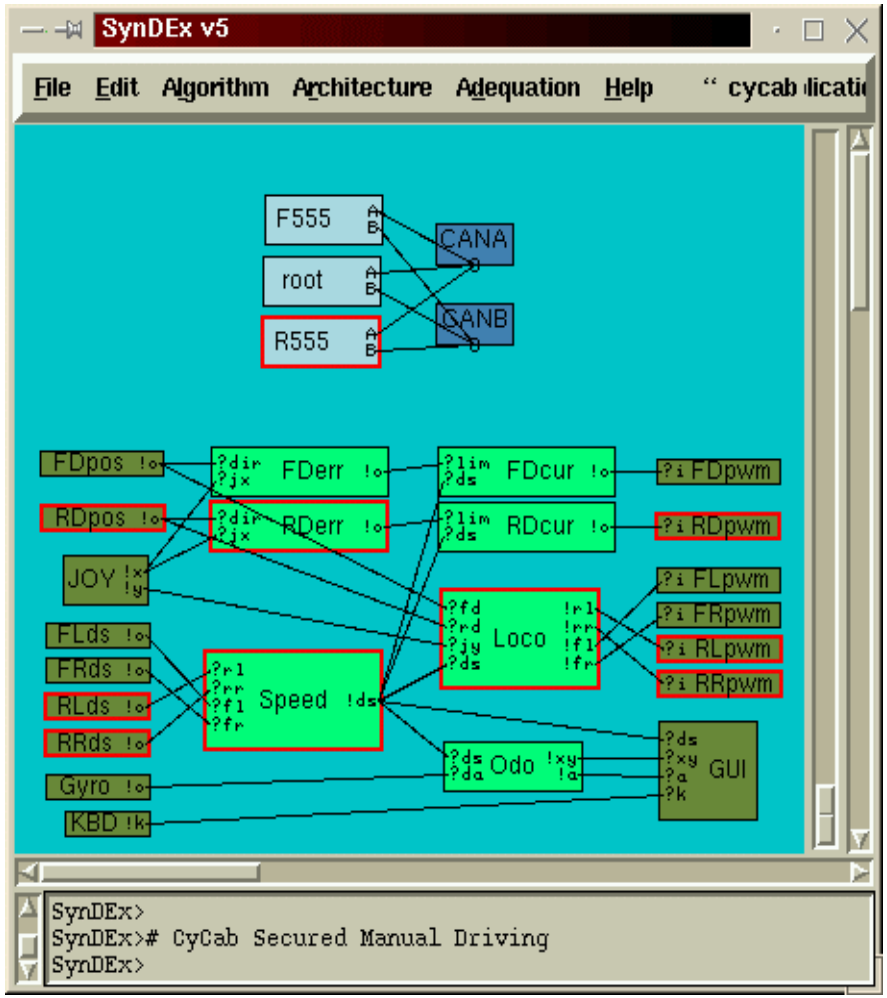
Exemple CyCab

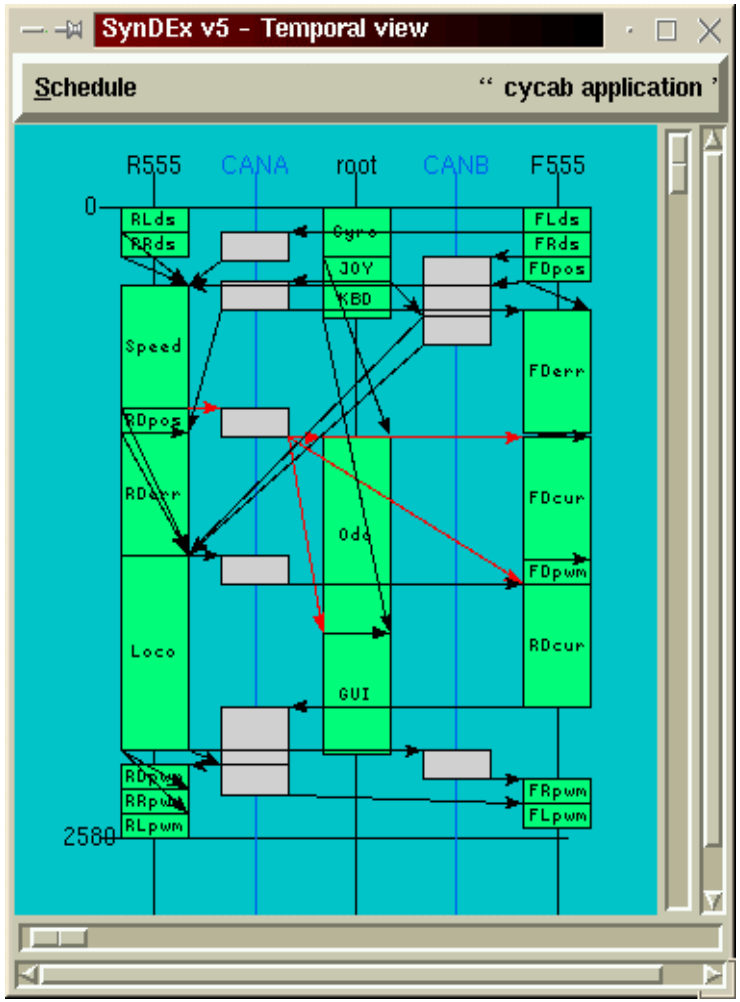
- Longueur 1,9m
- Largeur 1,2m
- Poids à vide 350kg
- Vitesse 30km/h
- Moteurs élec. 4x900w
- Autonomie 2h
- Recharge par induction

Industrialisé par Robosoft

http://www.robosoft.fr

SynDEx Conduite Manuelle CyCab





Synchronized Distributed Executives

- **Sans interblocage et à faible surcoût**
- **Macro-code processeur-indépendant (m4)**
  - Extensible, facilement portable (C ... asm)
- **Noyaux d'exécutif (macros) pour :**
  - Microcontrôleurs : MPC555, MC68332, 80C196
  - DSP : Sharc, TMS320C40
  - Microprocesseurs : i80x86, Transputer, C/Unix
  - Moyen de comm. : links, CAN, RS232, TCP/IP

Synthèse de circuit optimisée

- **Transformation de graphes**
  - Défactorisation (data parallélisme)
  - Défactorisation (pipe-line, retiming)
  - Jusqu'à contraintes temps réel satisfaites
- **Implantation de graphe factorisé**
  - Sommet factorisé = composant VHDL exécuté répétitivement sur diff. données
  - Hyperarc factorisé = signal VHDL
  - Composant/signaux contrôlés par compteurs/multiplexeurs/registres

Conclusion

- **Temps de développement très réduit**
  - Sécurité de conception
  - Prototypage rapide / implantation optimisée
  - Exécutifs générés sans debug
- **Perspectives :**
  - Multicontraintes temps réel
    - Hors ligne sans préemption
    - Hors ligne avec minimisation de la préemption
  - Co-design : unification processeurs circuits
  - Tolérance aux pannes : ARC TOLERE (BIP), IST SAFE