L'informatique embarquée

Cours : Gérard Berry

Chaire d'innovation technologique Liliane Bettencourt

Gerard.Berry@college-de-france.fr

Séminaire : Gérard Ladier

Airbus Industries

Gerard.Ladier@airbus.com

Collège de France, 15 février 2008

Alerte aux pucerons!

- Infestation massive par pucerons enfouis partout
- Qui apprennent à se coordonner par radio
- De plus en plus d'applications critiques

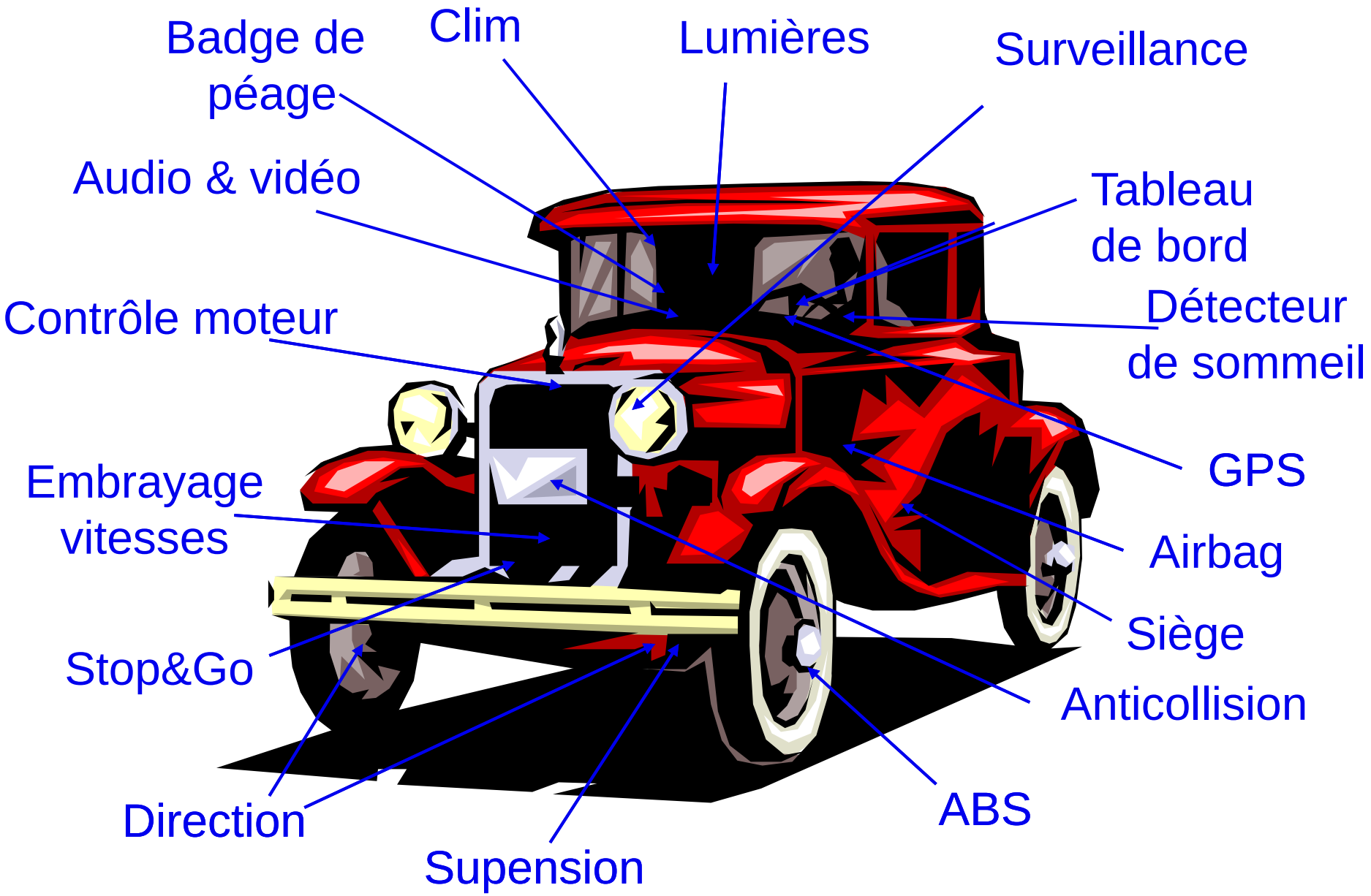
Applications et contraintes variées

pilotage, frein, distribution électrique, carburant, etc.
safety-critical => certification

trajectoire et attitude, imagerie, transmission
mission-critical => qualité supérieure

téléphone, audio, TV, DVD, jeux, ...
business critical => time-to-market

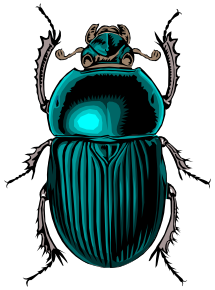
pacemakers, contrôle du glucose, robots chirurgiens,...
life-critical => TBD (!)



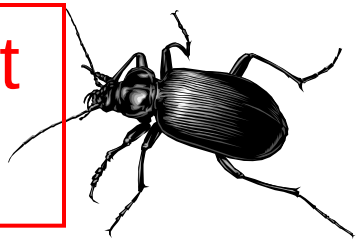
Coordination globale - Coordination avec la route et le trafic

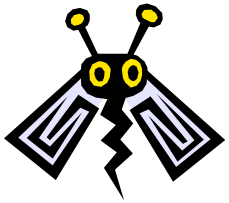
Ennemi public numéro 1 : le bug

- Therac 25 : irradiations léthales
- Les Patriot de Dharan
- Ariane 501
- Pathfinder, Mars Polar Lander
- Les ennuis des autos haut-de-gamme
- Pentium, Athlon
- Bugs des téléphones et appareils photos



Si on les laisse faire les bugs poussent plus vite que la loi de Moore!





Comment éviter ou contrôler les bugs?

1. De la vérification, encore et toujours
   - mais peut-on vraiment en faire plus?
2. De meilleures techniques de design
   - modèles de calcul plus simples
   - spécifications formelles lisibles par tous les acteurs
3. De meilleurs outils
   - exécution immédiate des spécification, maquettes virtuelles
   - génération automatique des circuits et codes embarqués
   - vérification formelle des propriétés critiques

Approche à la fois scientifique et industrielle

Déterminisme ou non-déterminisme

- Déterminisme : réactions toujours identiques aux mêmes sollicitations

  freinage, contrôle de vol, comptage de roues de train

- Non-déterminisme : réactions non constantes

  accès Web, tirage au sort

Non-déterminisme externe: celui de l'environnement (toujours présent)

≠

Non-déterminisme interne : celui du système (rarement souhaitable)

Temps-réel

- Fort : temps de réponse partie de la fonctionnalité
  - pilotage, freinage, contrôle moteur, airbag
  - son, vidéo, suivi GPS
  - gestion de trains de voitures

Avant l'heure, c'est pas l'heure,
après l'heure, c'est plus l'heure !

- Faible : temps de réponse raisonnable
  - initialisation GPS, climatisation, lumières
  - surveillance de bon fonctionnement
  - gestion des limites de vitesse

Parallélisme

- Les systèmes embarqués comportent toujours des activités parallèles

  compactes, temps-réel fort : pilotage 3 axes
  freinage 4 roues
  réception / émission radio

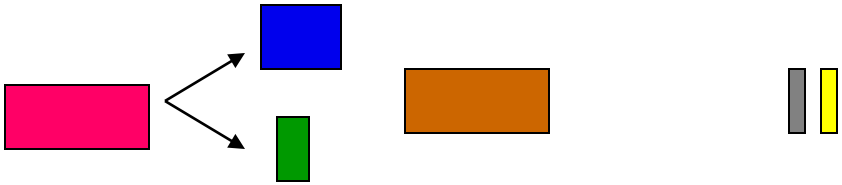
  distribuées, temps réel faible : contrôle aérien
  surveillance des fonctions

- Comment programmer et vérifier le parallélisme ?

  tâches + OS, ordonnancement dynamique ou statique
  "threads" et "callbacks" de C ou Java
  programmation synchrone

Approche classique : tâches et OS

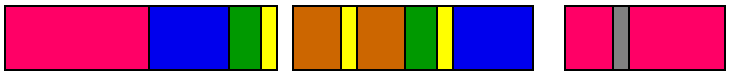
- Découpe en tâches périodiques / sporadiques



- Ordonnancement dynamique par événements / timers

  paramètres: masquages, interruptions, priorités, etc.

  souple, mais très difficile à maîtriser et valider (PathFinder)



- Ordonnancement statique

  statique pour périodiques, trous pour sporadiques

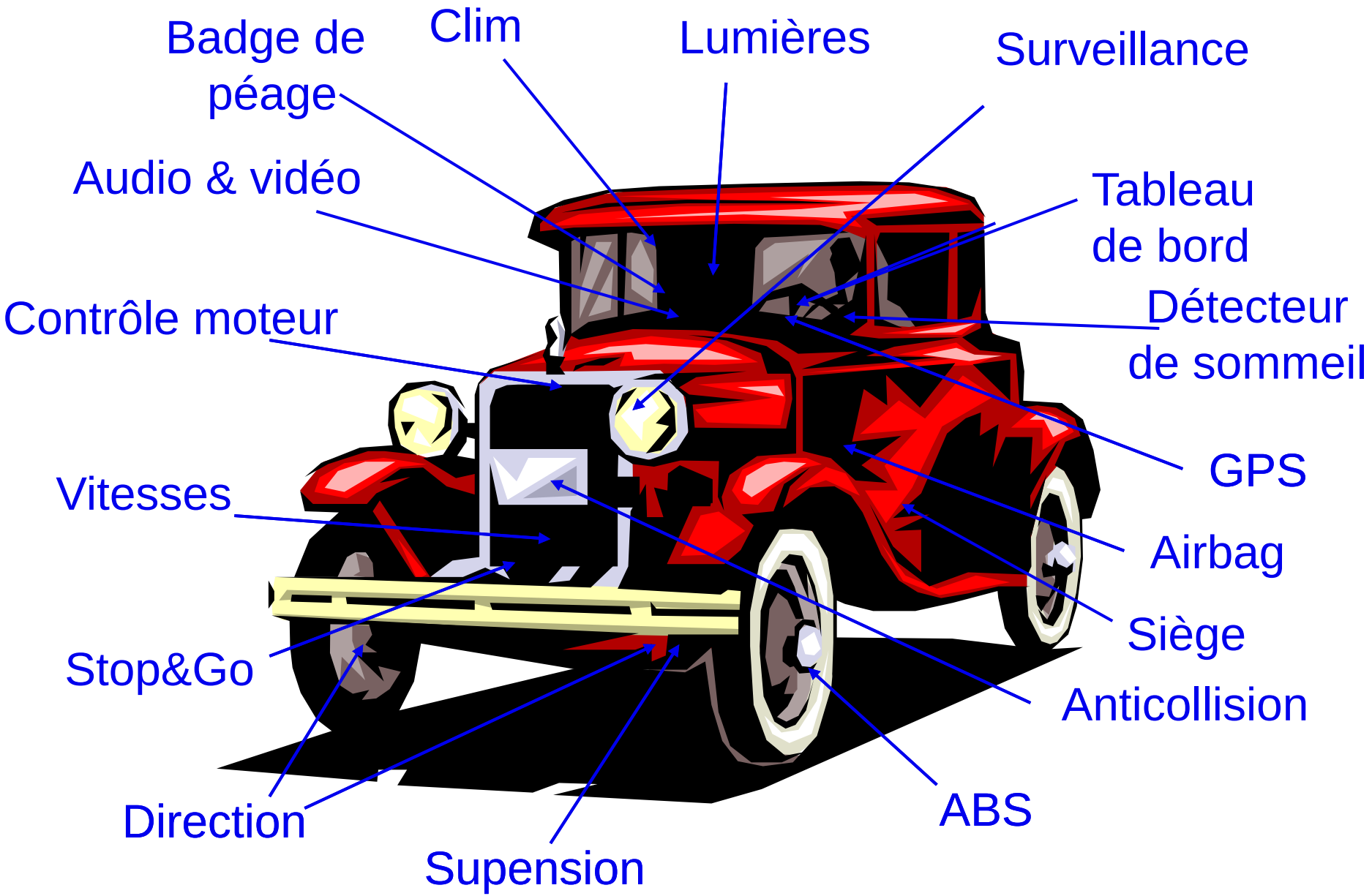
  maîtrisable mais difficile à calculer, instable aux changements

Goulots d'étranglement

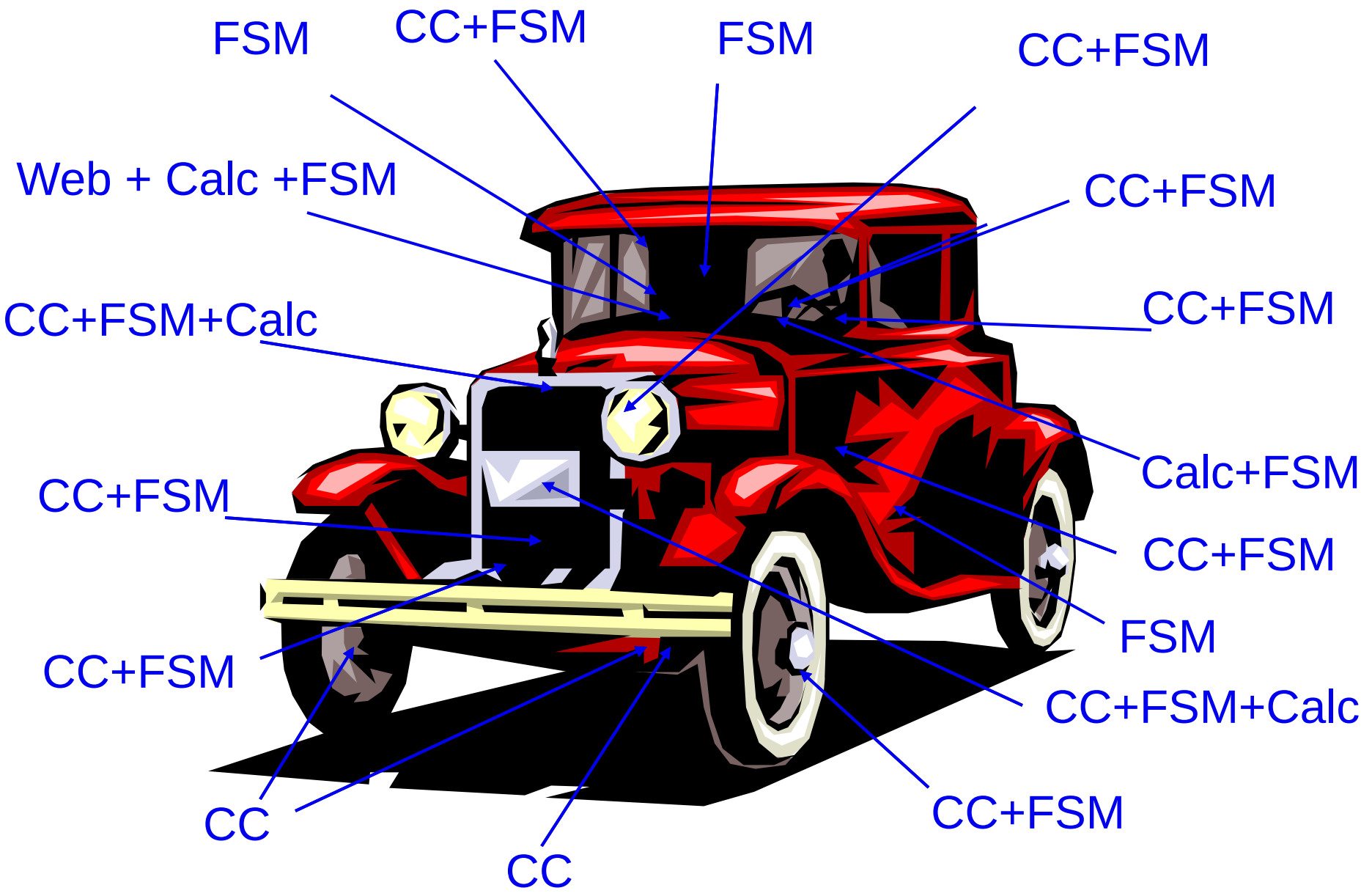
- La complexité des applications va en croissant
  - de plus en plus de fonctions, d'algorithmes, d'interactions
  - spécifications globales et locales très difficiles à écrire
- La recherche de performance accroît la complexité
  - répartir les fonctions accroît le non-déterminisme
  - minimiser la puissance dissipée complique la logique
- La vérification est le vrai goulot d'étranglement
  - le non-déterminisme fait exploser le nombre de tests
  - l'interconnection des parties est très difficile à tester
  - idem pour l'interaction matériel / logiciel
  - la répartition simplifie le matériel, pas la vérification!..

Anatomie des applications embarquées

- CC : contrôle continu, traitement du signal
  - résolution d'équations différentielles, filtrage numérique
  - spécification et simulation en Matlab / Scilab
- FSM : automates finis, graphes de transition
  - contrôle discret, protocoles, sécurité, etc.
  - automates plats ou hiérarchiques
- Calc : calcul intensif
  - navigation, cryptage, etc.
  - C + librairies
- Web : navigation, audio / video
  - interaction graphique / audio / vidéo
  - réseaux flots de données, Java

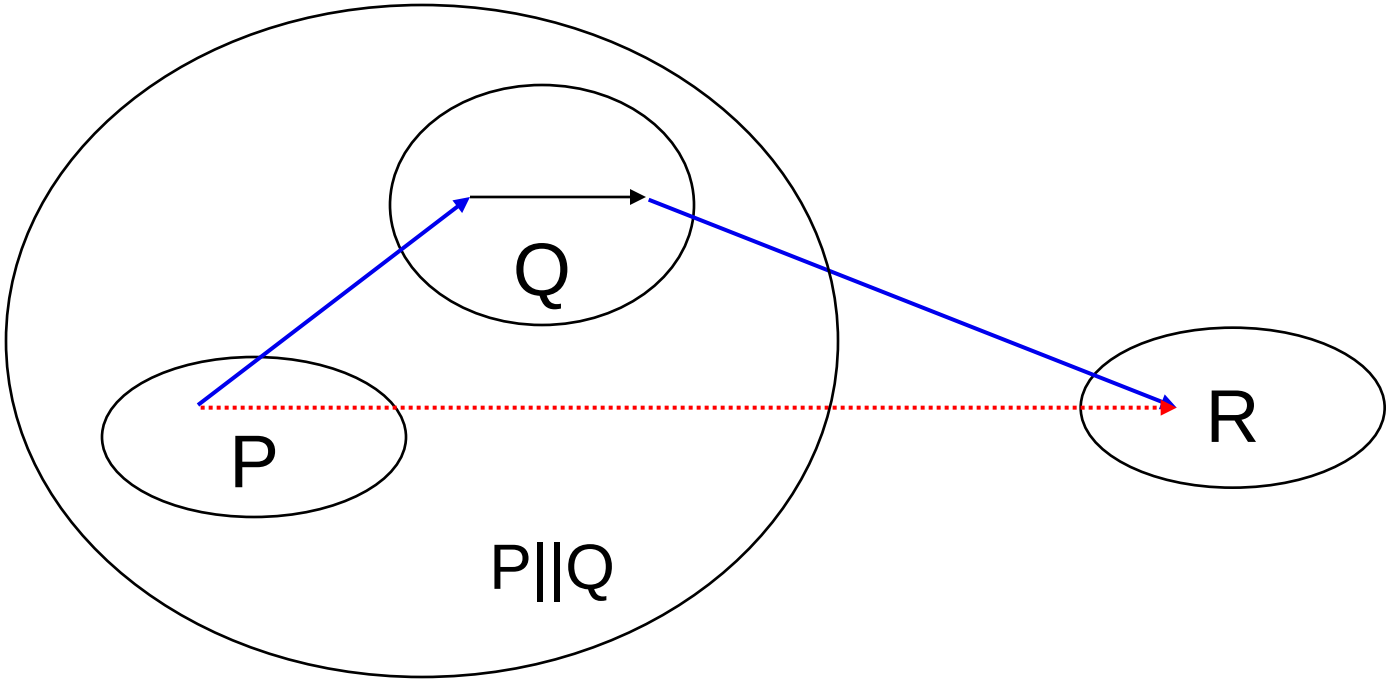


Coordination globale CC+FSM+Calc

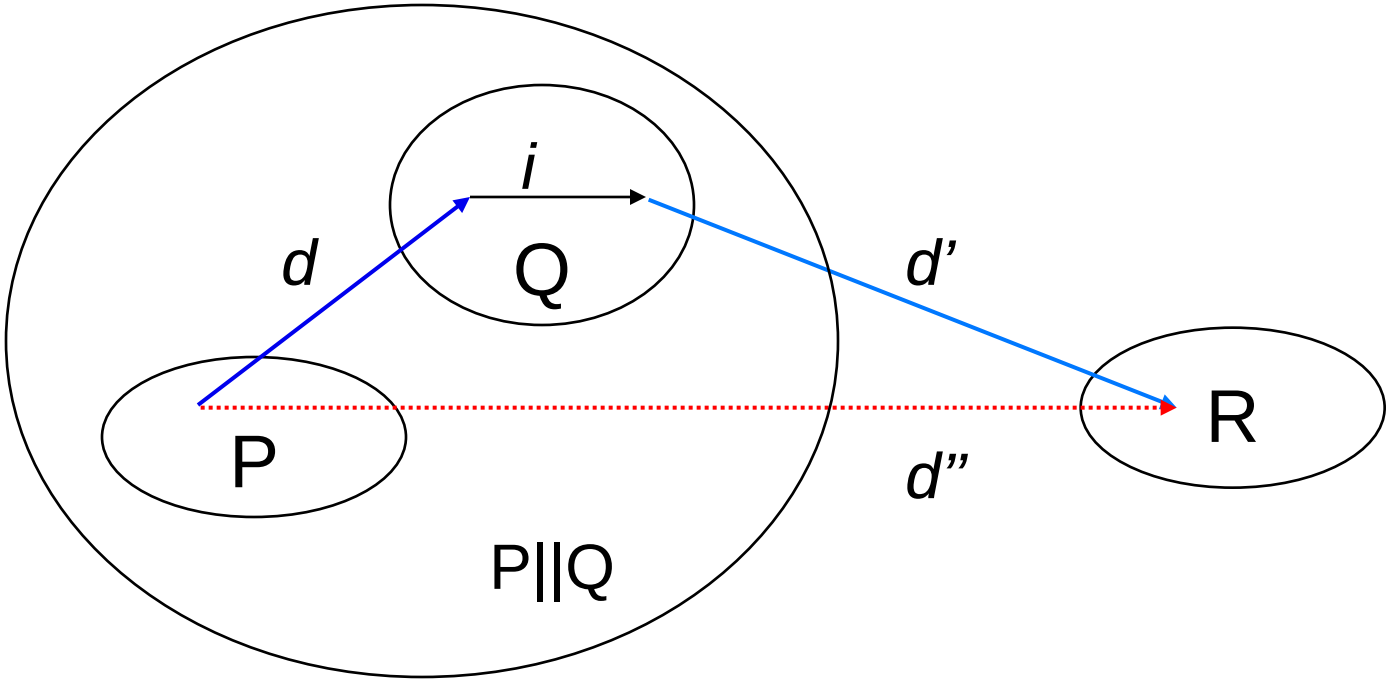


Coordination globale CC+FSM+Calc


PARALLILISM
PIRELLI
EQUILIBRAGE



Parallélisme : le principe de compositionnalité



$$d'' = d + i + d'$$

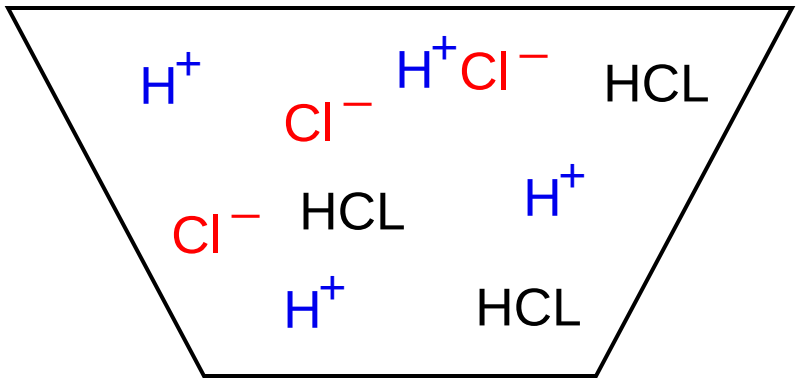
$$d'' \sim d \sim i \sim d'$$

$$d \sim d + d$$

Exactement 3 solutions (en informatique et en physique)

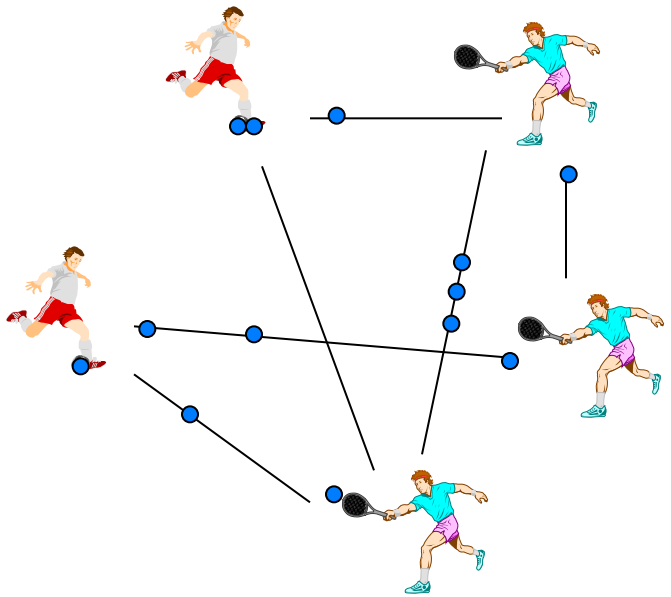
1. *d* arbitraire — asynchronisme
2. *d* nul — synchronisme
3. *d* prévisible — vibration

Délai arbitraire : le mouvement Brownien



$H^+ + Cl^- \longleftrightarrow HCL$

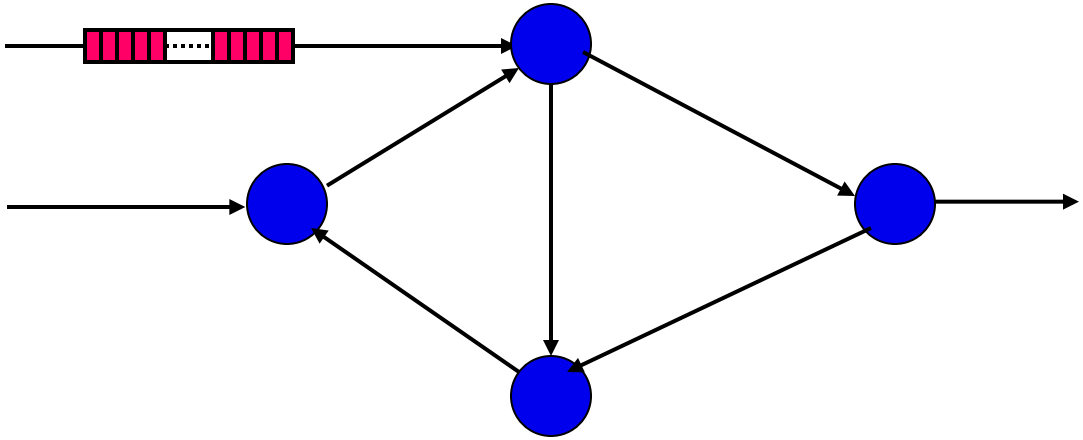
réaction chimique



routage internet

Modèles : réseaux de Kahn, CSP / ADA,...
π-calcul, CHAM, Join-Calculus, Ambients,...

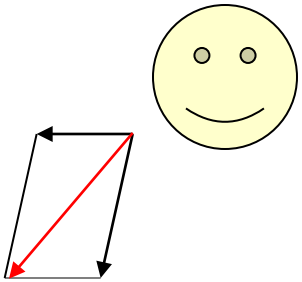
Réseaux de Kahn



- noeuds = programmes déterministes
- flèches = files fifo non bornées
- exécution asynchrone des noeuds
- mais avec lecture bloquante

Parallélisme globalement déterministe
Sémantique mathématique élégante

Délai nul : la mécanique Newtonienne



Parallélisme + Déterminisme
Calculs faisables

Délai prévisible : la vibration

La meilleure illustration connue de la vibration se trouve dans "Le sceptre d'Ottokar", Hergé, page 29, dernière vignette.

La puissance de la voix de Bianca Castafiore chantant l'air des bijoux du Faust de Gounod suffit à faire trembler le micro et les musiciens !

propagation de la lumière, du son, de l'électricité, des vagues ou du compteur ordinal

La vibration implémente le délai nul !

Dans "Le sceptre d'Ottokar", page 38, première vignette, Bianca Castafiore chante pour le roi Muskar XII de Syldavie et ses invités dans la grande salle du château de Klow.

Bien que la vitesse du son soit finie, elle est suffisamment grande par rapport à la taille de la salle pour paraître infinie aux spectateurs.

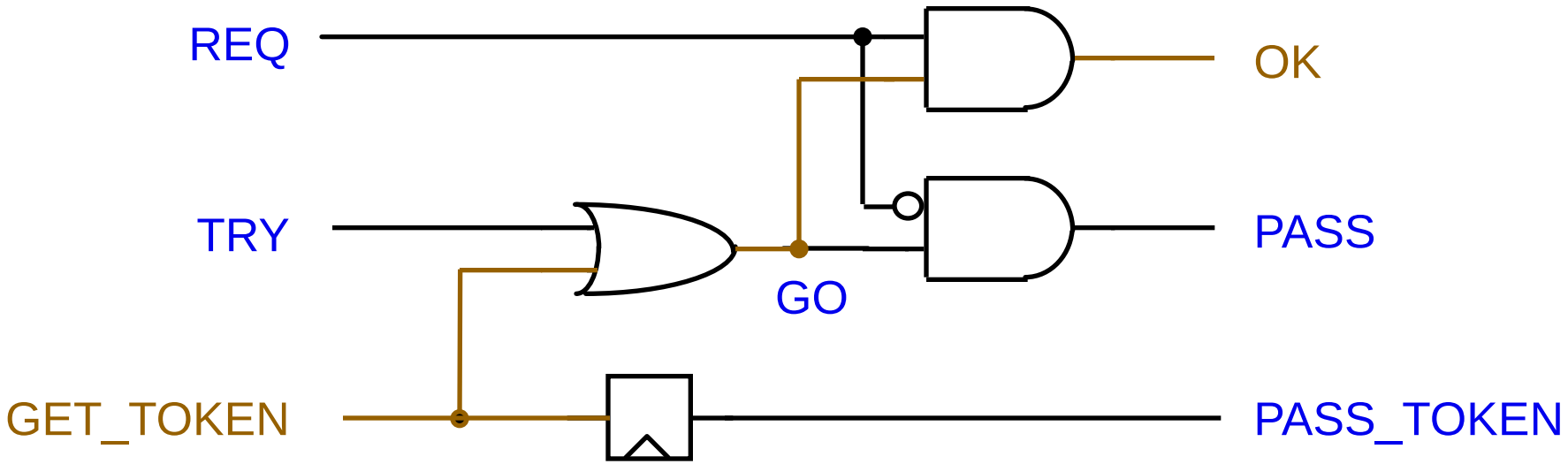
Si la pièce est assez petite,
on peut négliger la vitesse du son

Spécifier et programmer en délai nul

Implémenter en délai prévisible

Contrôler la taille de la pièce

Circuits : vision logique zéro-délai

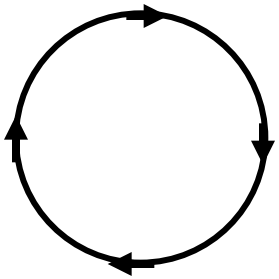


OK = REQ and GO
PASS = not REQ and GO
GO = TRY or GET_TOKEN
PASS_TOKEN = reg(GET_TOKEN)

attendre le temps critique = résoudre les équations!

Logiciel synchrone

Exécution cyclique



lire les entrées

calculer la réaction

produire les sorties

Synchrone = délai 0 = dans le même cycle

propagation parallèle du contrôle

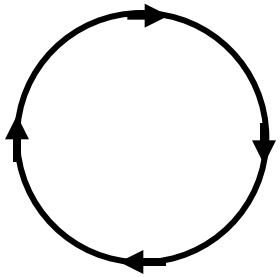
propagation parallèle des signaux

Mathématiques très élégantes

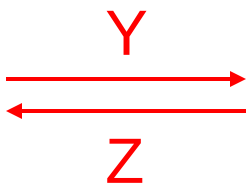
Pas d'interférences => déterminisme par construction

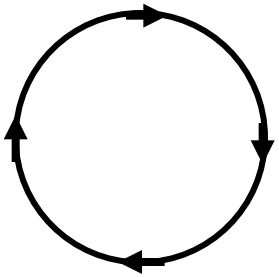
Taille de la pièce = Worst Case Execution Time (AbsInt)

Parallélisme = fusion de cycles

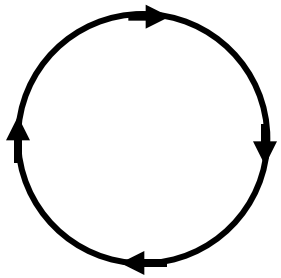


```
input X, Z;
output Y, T;
Y = X+1;
T = Z/2
```





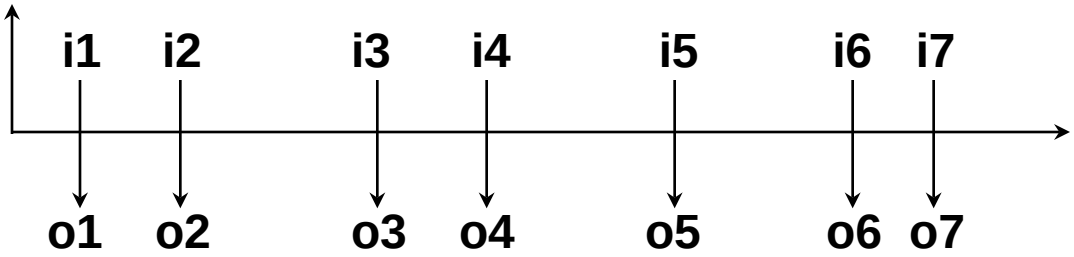
```
input Y;
output Z;
Z = Y*3;
```



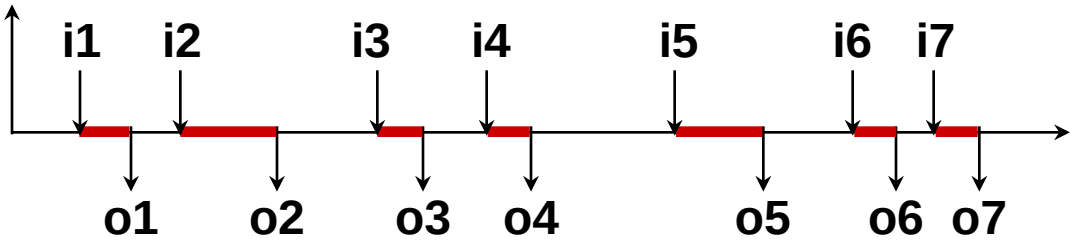
```
input X;
output T;
local Y, Z;
Y = X+1;
Z = Y*3;
T=Z/2;
```

Pas de synchronisation, pas de blocages,
pas de changements de contextes

Temps logique et temps physique



Temps logique



Temps physique

WCET OK = absence de recouvrement

Le coureur Esterel

```
trap HeartAttack in
    every Morning do
        abort
            loop
                abort  run Slowly  when  100 Meter ;
                abort
                    every Step do
                        run Jump || run Breathe || run CheckHeart
                    end every
                when 15 Second ;                    exit HeartAttack
                run FullSpeed
            each  Lap
        when  4 Lap
    end every
handle HeartAttack fo
   run RushToHospital
end trap
```

Lustre = réseaux de Kahn synchrones

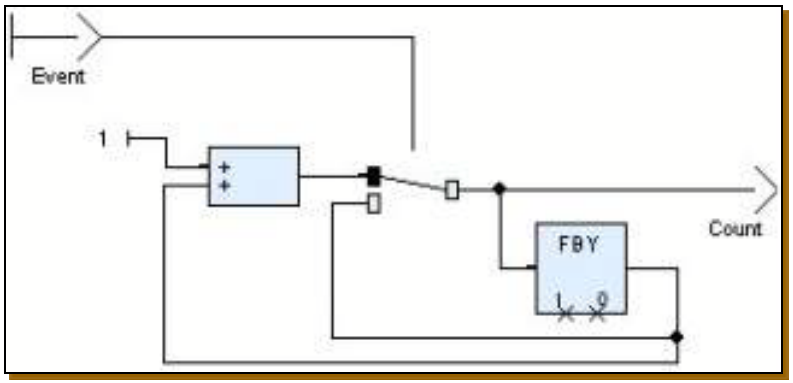
Compteur d'événements

Event = false true false true true false false false true true false

Count = 0 1 1 2 3 3 3 3 4 5 5

$$\begin{cases} Count(0) = 0 \\ \forall t > 0,\ Count(t) = \begin{cases} Count(t-1)+1, & if\ \ Event(t) = true \\ Count(t-1), & otherwise \end{cases} \end{cases}$$

```
Count = 0 ->
    (if Event
     then pre(Count)+1
     else pre(Count))
```

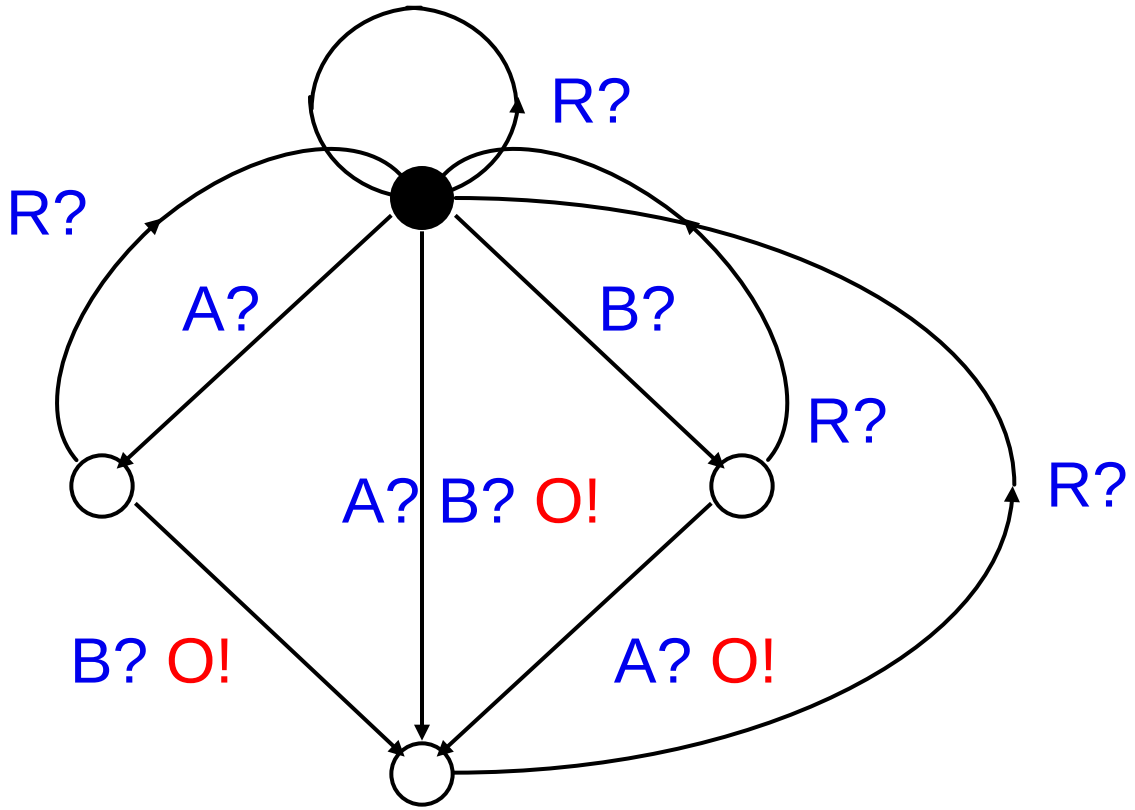


Lustre textuel

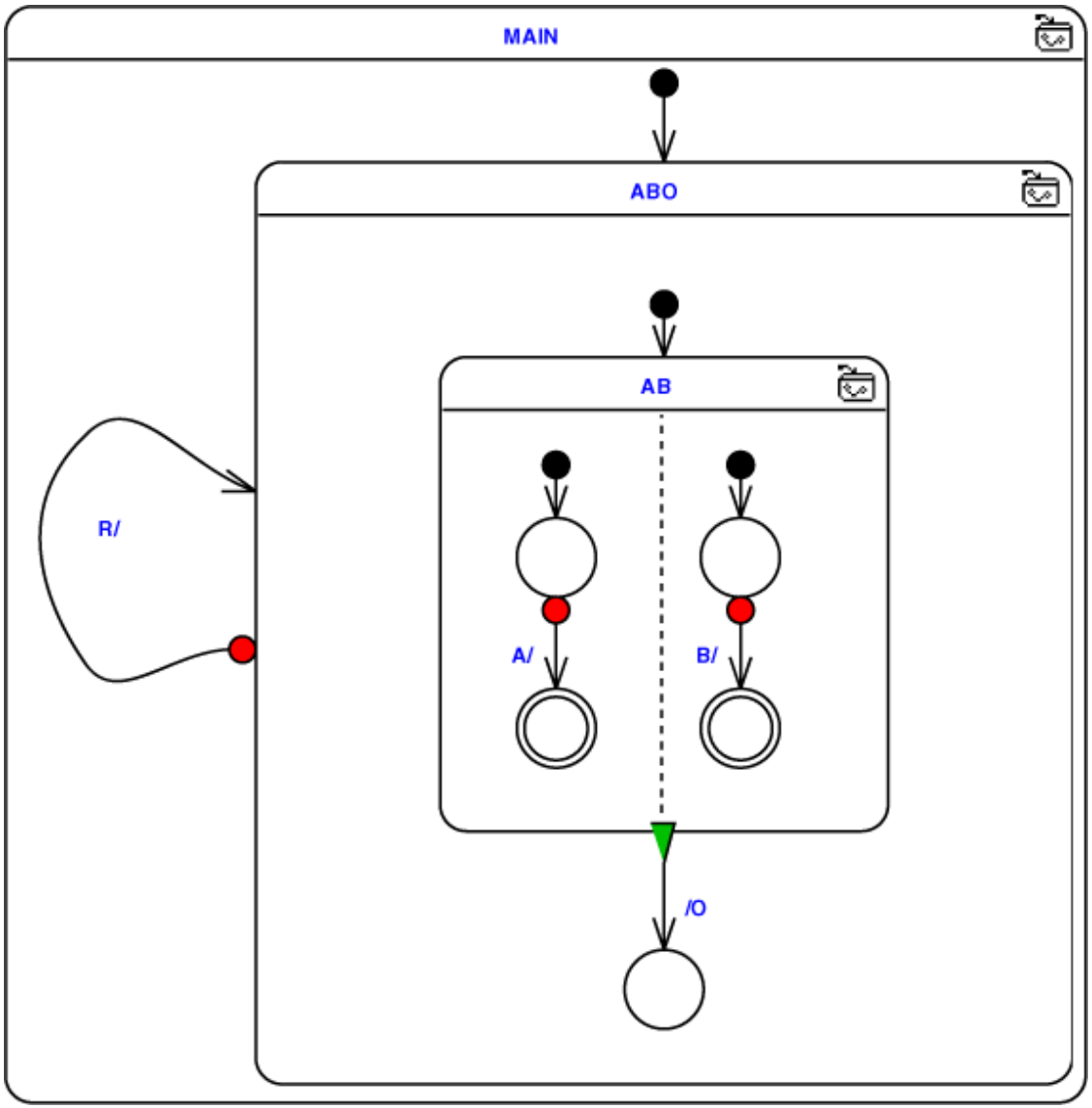
Diagramme de blocs SCADE

Automate plat : ABRO

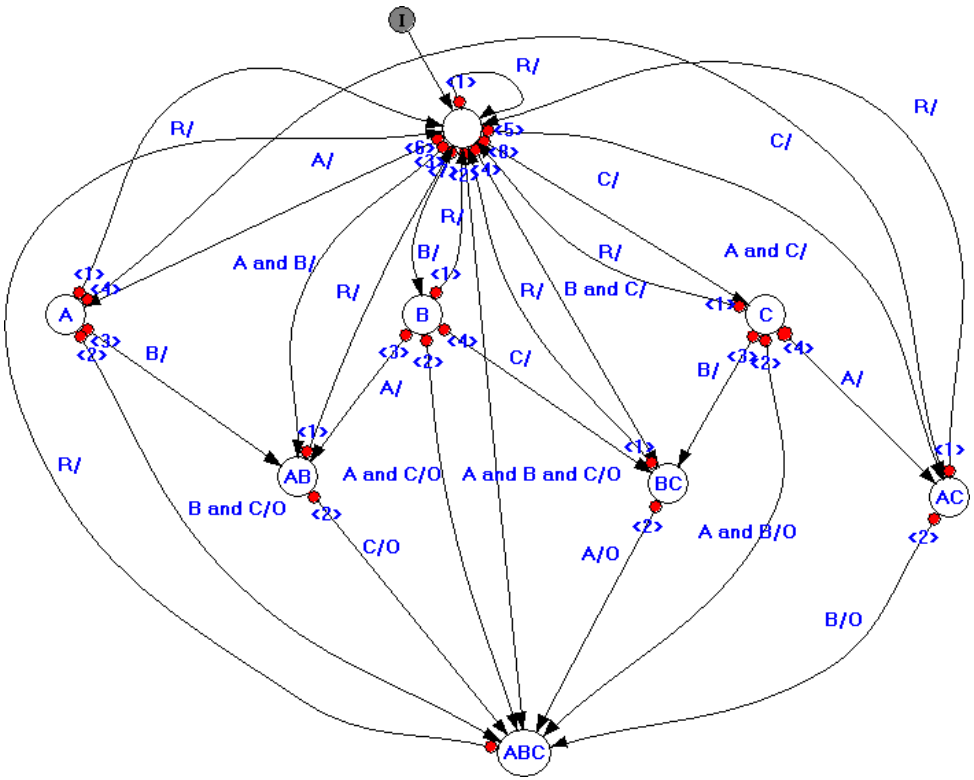
Emettre O dès que A and B sont arrivés
Recommencer à zéro à chaque R

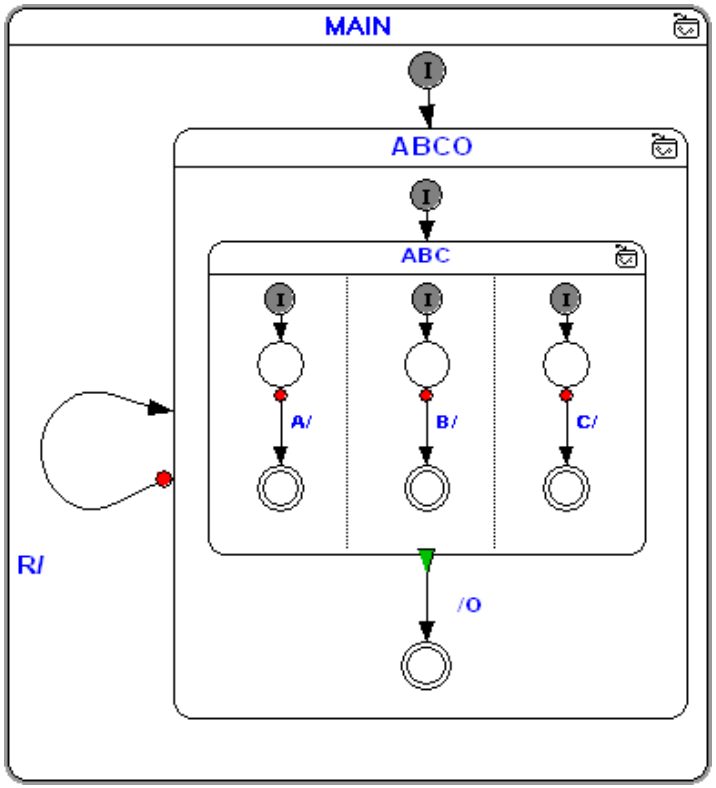


ABRO hiérarchique



ABCRO : vive la hiérarchie!





Automate plat

exponentiel

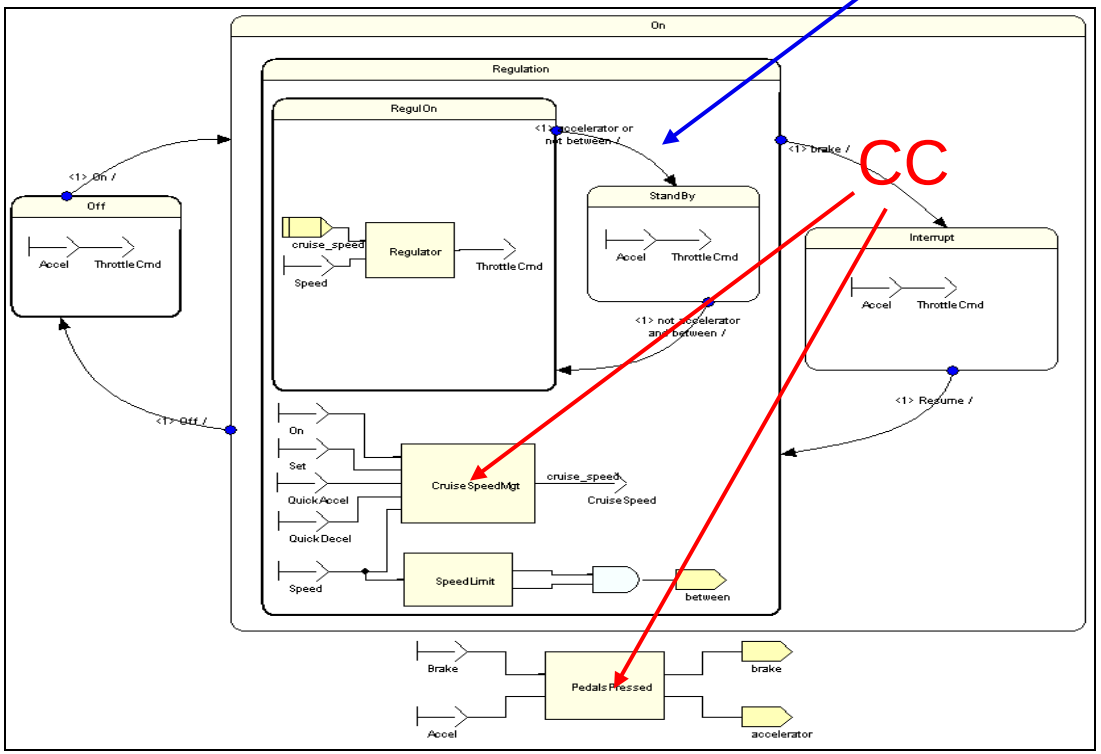
Automate hiérarchique

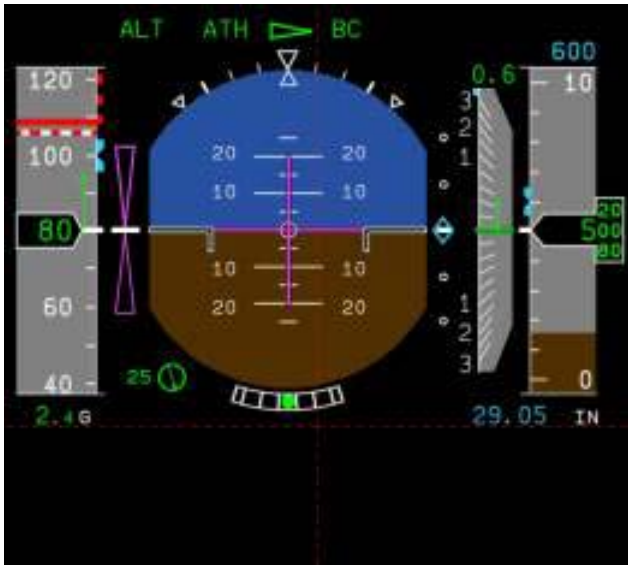
linéaire

SCADE 6 = CC + FSM + Calc

FSM

CC





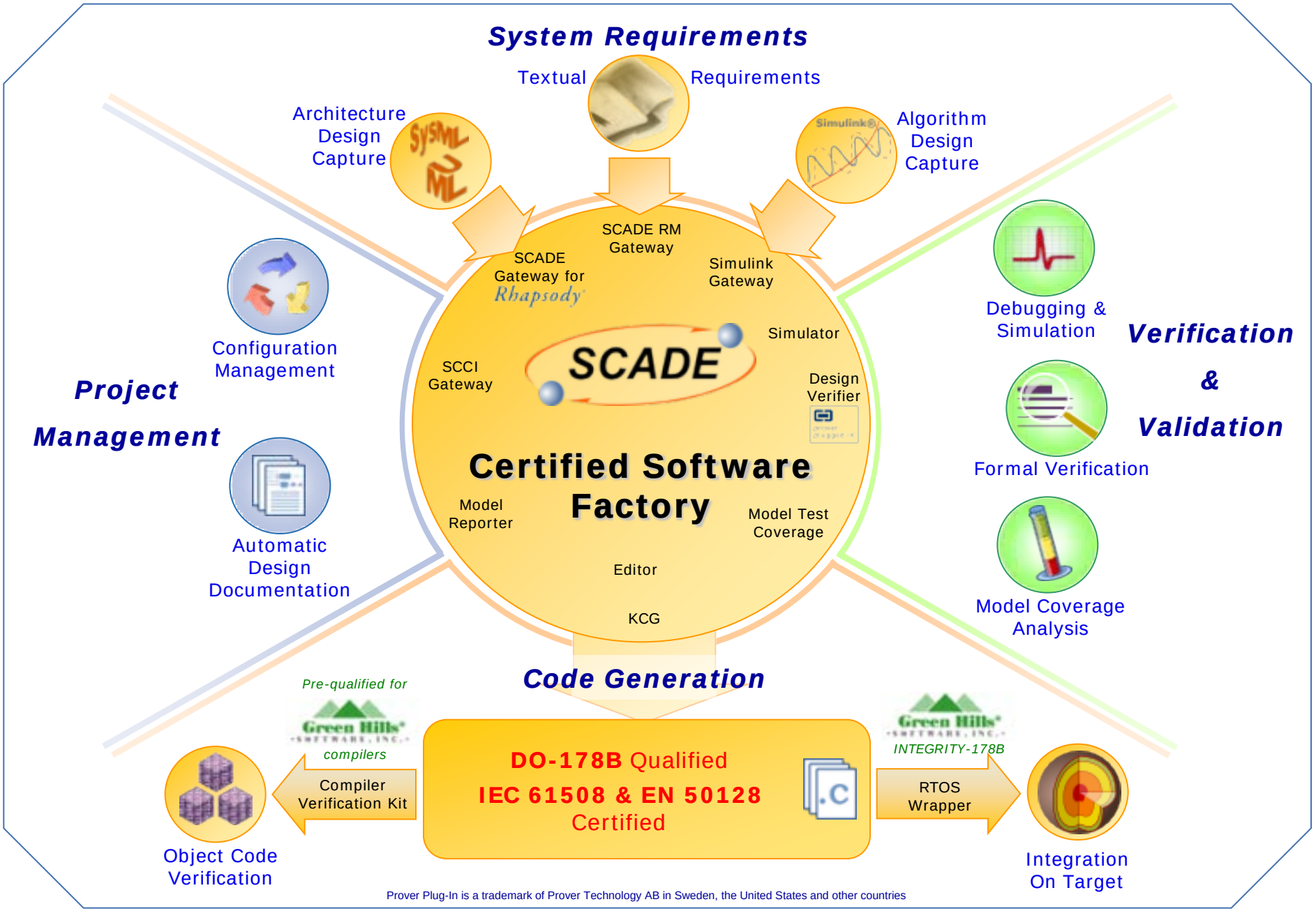
SCADE Display

Intégration de diagrammes de blocs, d'automates et de visualiseurs

Sémantique mathématique

- Définit le sens des programmes
  - Lustre: équations aux flots infinis
  - Esterel : règles d'inférences logique
- Apporte la sûreté de programmation
- Définit ce que compiler veut dire

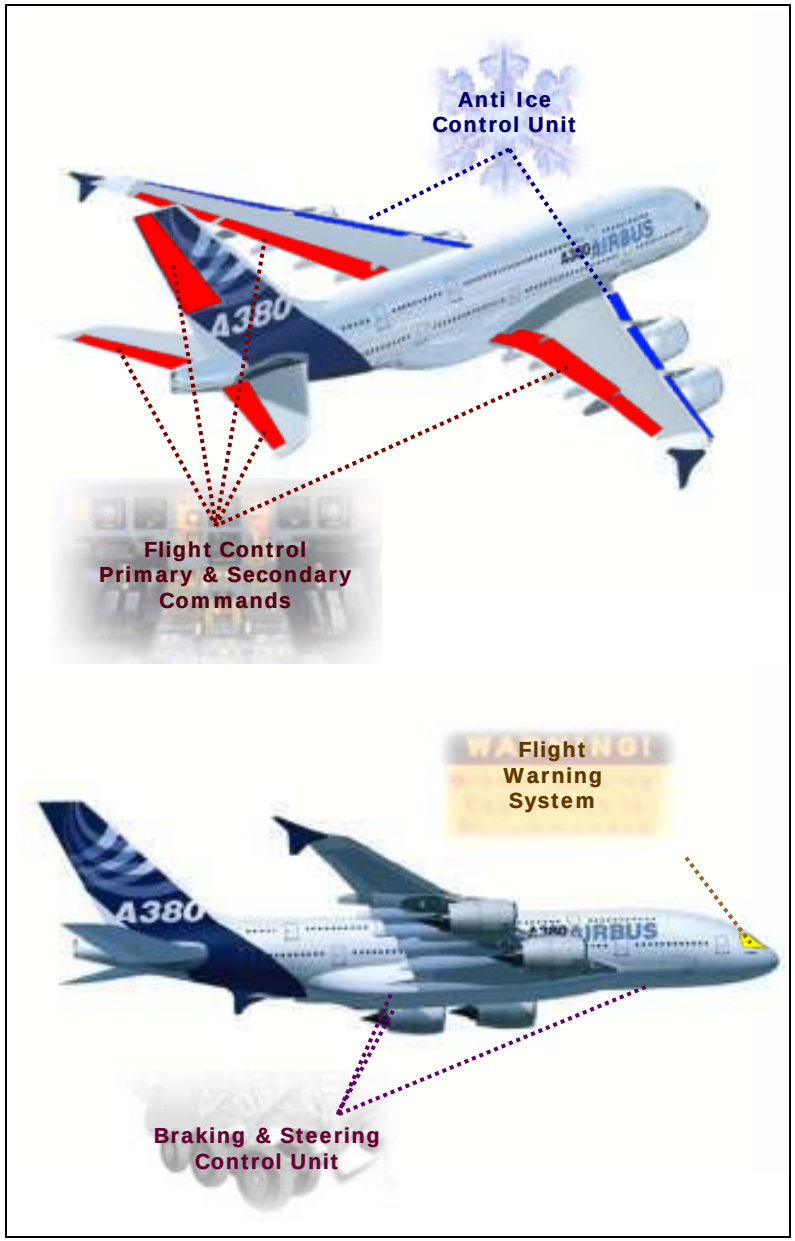
Se méfier des contrefaçons !


System Requirements
Textual Requirements
Architecture Design Capture
Algorithm Design Capture
SCADE RM Gateway
SCADE Gateway for Rhapsody
Simulink Gateway
Simulator
SCCI Gateway
SCADE
Design Verifier
Certified Software Factory
Model Reporter
Model Test Coverage
Editor
KCG
Configuration Management
Project Management
Automatic Design Documentation
Debugging & Simulation
Verification & Validation
Formal Verification
Model Coverage Analysis
Code Generation
Pre-qualified for
Green Hills SOFTWARE, INC.
compilers
Compiler Verification Kit
DO-178B Qualified
IEC 61508 & EN 50128 Certified
.C
INTEGRITY-178B
RTOS Wrapper
Object Code Verification
Integration On Target
Prover Plug-In is a trademark of Prover Technology AB in Sweden, the United States and other countries

Code généré par KCG

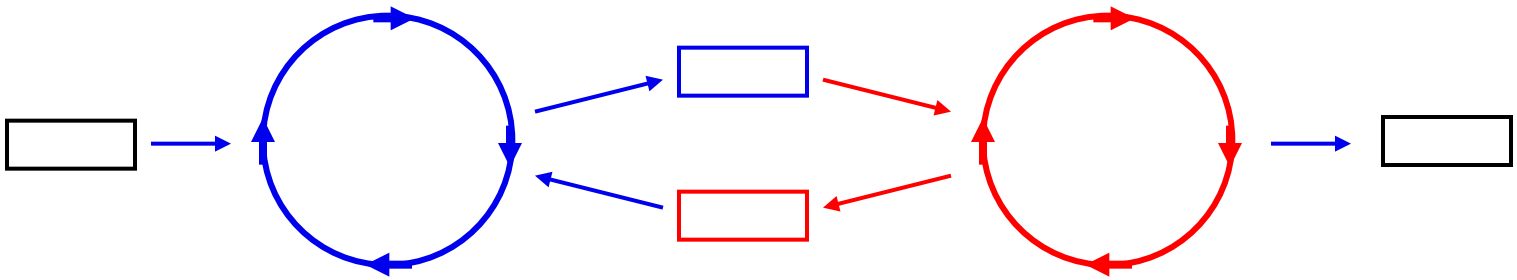
- Petit sous-ensemble de C
- Portable (indépendant du processeur, du compilateur, et de l'OS)
- Structuré, lisible, traçable
- Avec allocation mémoire statique
- Sans arithmétique de pointeurs
- Sans récursion
- A temps d'exécution borné et prévisible
- Optimisé en temps et mémoire

SCADE dans l'Airbus A380



- Contrôle de vol
- FADEC (contrôle moteur)
- Freinage et direction
- Gestion électrique
- Anti-givrage
- Système d'alarmes
- Cockpit:
  - PFD : Primary Flight Display
  - ND : Navigation Display
  - EWD : Engine Warning Display
  - SD : System Display
- ...

Extension du modèle: des pièces aux châteaux



Echantillonage mutuel (Caspi, et. al.)

- Fonctionne à causes de raisons d'automatique, pas d'informatique (Théorème de Nyquist distribué)
- Semblable aux circuits multi-horloges, mais plus simple (pas de métastabilité des mémoires)
- Alternative : Time Triggered Networks (TTP, FlexRay)

Conclusion

- L'embarqué: un domaine en explosion
  - avec de grandes exigences de qualité
- Qui demande une conception spécifique
  - contrôle du parallélisme et du temps réel
  - respect du déterminisme
  - génération de code prédictible
- Et des techniques de vérification spécifiques
  - cf. séminaire et prochain cours!