

ONERA

Centre d'Études et de Recherches de
Toulouse

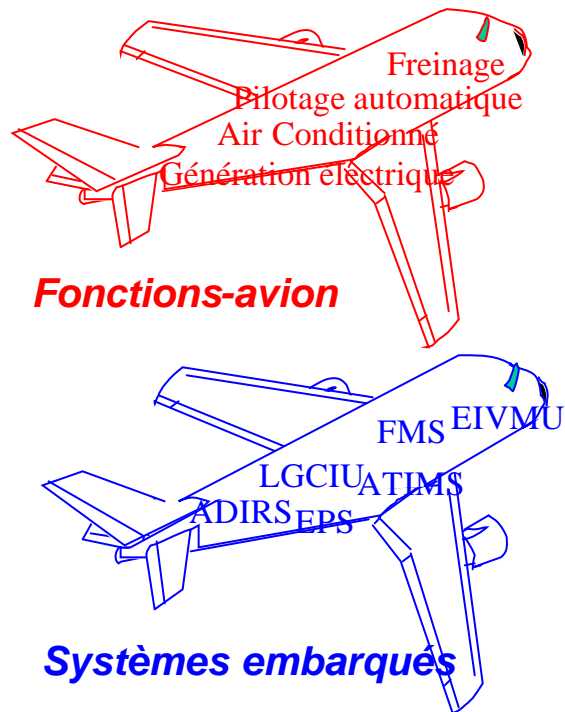
Modèles conceptuels pour le développement et l'analyse de systèmes avioniques modulaires

J. Foisseau, G. Bel, F. Boniol, V. Wiels

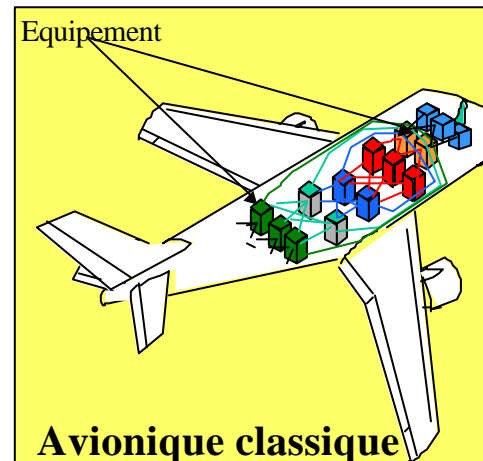
ONERA - CERT

Workshop UML & TR - Ecole Centrale de Nantes - 22 novembre 2001

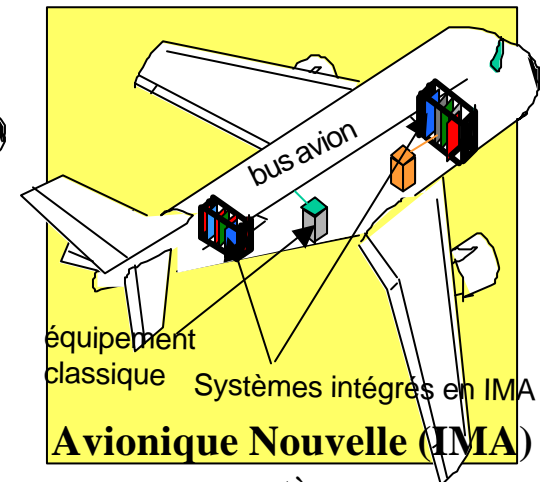
1. Contexte - le domaine avionique -



Ensembles de logiciels, calculateurs, bus, capteurs, actionneurs (*équipements*), définissant des systèmes réalisant les **fonctions-avion**, et respectant des contraintes temps-réel et des contraintes de criticité (certification)




• vue équipement



• vue système

1. Contexte - le domaine avionique -

Constats

- **interdépendance** grandissante entre les systèmes de l'avion
 - **complexité** intrinsèque des systèmes
 - de plus en plus de fonctions avion à installer à bord
 - coexistence de **plusieurs points de vue** (et donc plusieurs métiers)
 - fonctionnel
 - sûreté de fonctionnement
 - performances temps réel...
- 
- Problème de la cohérence des informations définies, manipulées, stockées par ces différents métiers**
- coût de l'avionique : > 1/3 coût d'un avion

Evolutions

- **évolution technologique** (IMA)
- évolution de la **certification**

2. Le projet PRISME : objectifs

Etudier une approche globale (avion) pour

- la **définition**,
 - la **validation**,
 - **l'intégration**
 - et la **qualification**
- de systèmes avioniques IMA
(parties traitement de l'information)

Prise en compte de 3 points de vue

- **fonctionnel**,
- **performance temps réel**
- **sûreté de fonctionnement**,

Fiche Projet

- **4 ans, 1997-2000**
- **10 personnes (~5 eqtp par an)**
- **3 Départements ONERA impliqués :**
 - DTIM (informatique)**
 - DCSD (automatique)**
 - DOTA (optique)**
- **collaborations industrielles :**
 - Airbus, Dassault Aviation**
- **collaborations universitaires :**
 - LaBRI, IRCyN**

2. Le projet PRISME : approche, hypothèses

Approche

développement d'une avionique sur la base d'une suite de modèles pour différents niveaux d'abstraction

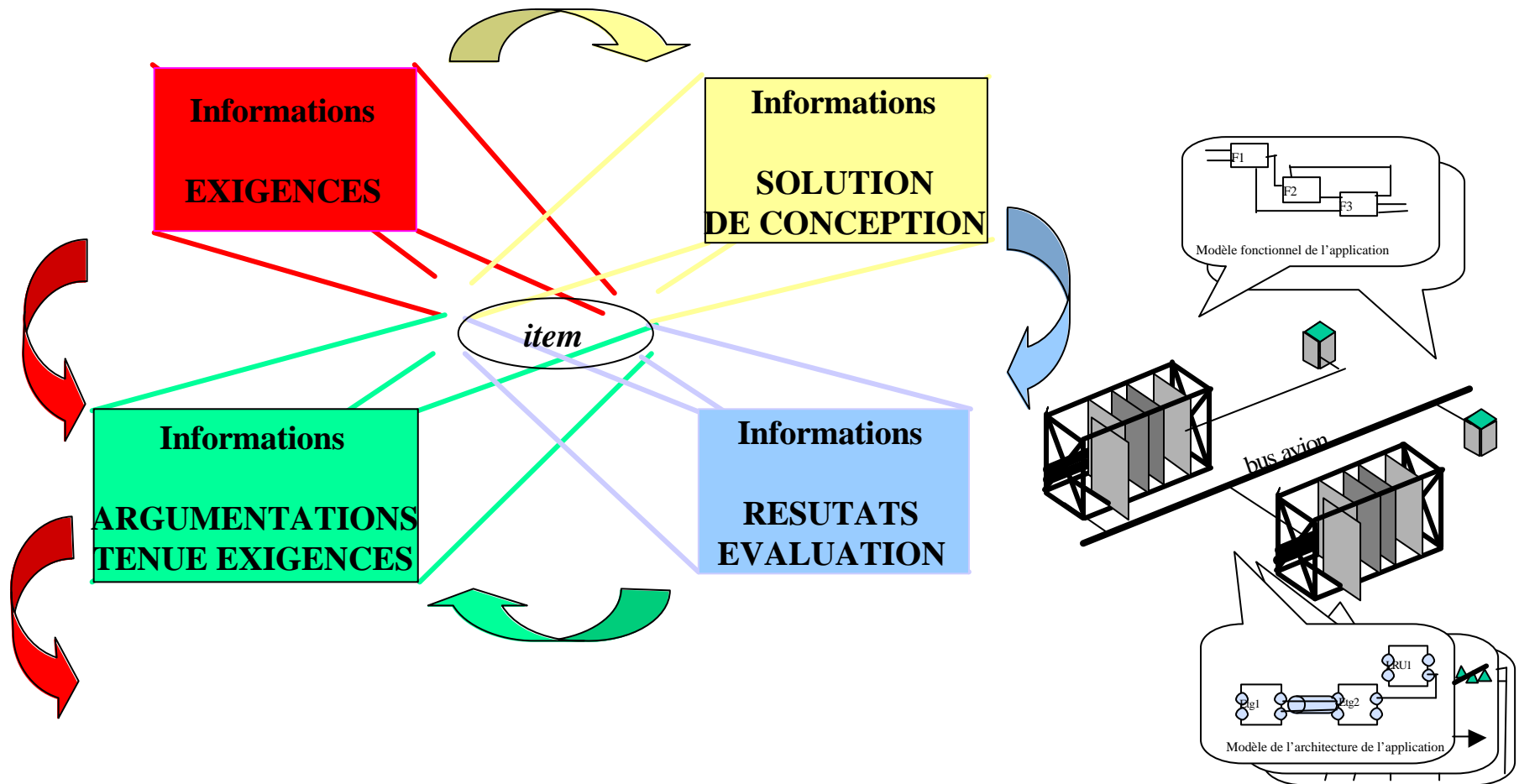
Hypothèses

- il existe une description pivot - à trouver! - (notion de squelette) d'une avionique, définition de référence pour les descriptions métier (fonctionnel, performances TR et Sûreté de Fonctionnement)
- pas de développement de nouveaux langages de spécification (réutilisation des langages du domaine : Lustre, Esterel, SDL, etc.)
- pas de développement d'outils nouveaux d'évaluation d'une avionique (utilisation d'outils existants : SCADE, SES/Workbench, Modline, ALTARICA, etc.)

2. Le projet PRISME : résultats

- **Les définitions des modèles conceptuels**
(diagrammes UML, spécifications formelles en TELOS-like)
- **Une méthodologie de modélisation**
- **Une plate-forme logicielle**
 - *une BD mémorisant les informations selon les différents modèles*
 - *couplage avec des langages de spécification (Esterel, Lustre, Altarica, SdL) et avec des outils d'évaluation, d'analyse comportementale, ou de vérification de propriétés (model-checking)*
- **Une plate-forme matérielle**
 - *des stations UNIX TR connectées sur un bus FDDI permettant d'émuler des architectures avioniques et des protocoles de communication différents (Arinc 629, AFDX, etc.) pour recalage des modèles*
- **Une chaîne de compilation vers la plate forme matérielle**
- **Une démonstration**
 - *une application pilote fournie par EADS Airbus SA*

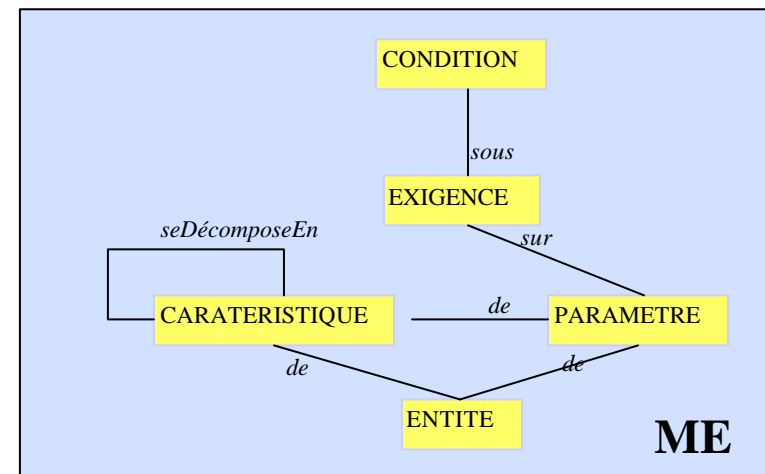
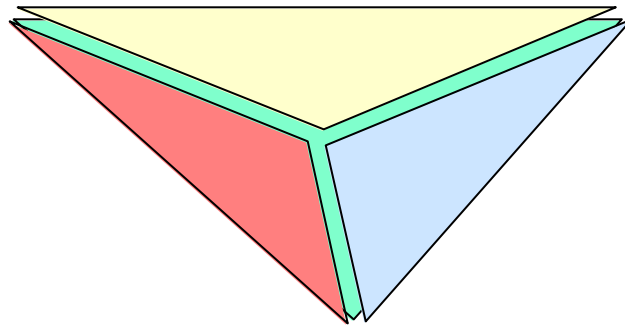
3. PRISME : les besoins en modèles conceptuels



3. PRISME : les besoins en modèles conceptuels

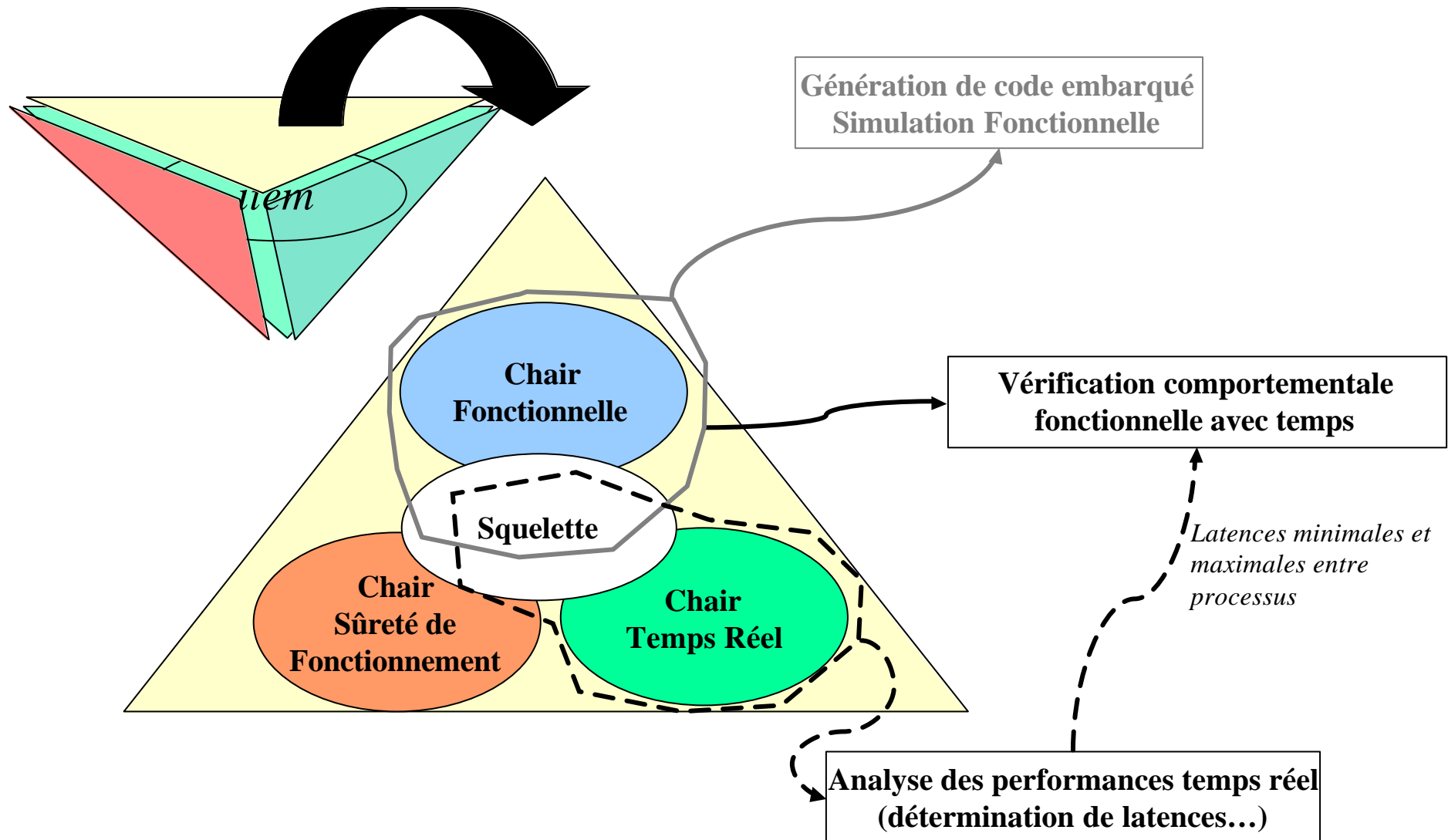
Modèles conceptuels nécessaires dans le PRF AVN:

- description des exigences définissant une avionique **ME**
 - description d'une solution de conception d'une avionique **MSol**
 - description d'évaluations d'une avionique **MR**
 - description d'une argumentation de tenue des exigences **MArg**
- (NB : modèles interdépendants)



Exemple : extrait du modèle conceptuel des exigences

4. Les sous modèles conceptuels de MSol



4. Les sous modèles conceptuels de MSol

1. Un squelette :

Pourquoi : pour assurer la cohérence structurelle entre les différentes vues du système

=> squelette = toutes les informations communes à au moins deux vues

=> organisation du squelette en 3 parties :

→ **une partie « logique » (ML)** : description de l'architecture fonctionnelle du système en termes de « computations » et de « données »

→ **une partie « matérielle » (MA)** : description de l'architecture formée par les bus numériques et les ressources de traitement ou de transfert

→ **une partie « intégration » logique sur matériel (MI)** : description des choix de regroupement des « computations » en partitions, des choix d'allocation des partitions sur des ressources de traitements, des choix d'allocation des données sur des « chemins physiques »...

=> représentation structurelle d'un système avionique

Mais : pas de sémantique (comportement défini dans les chairs...)

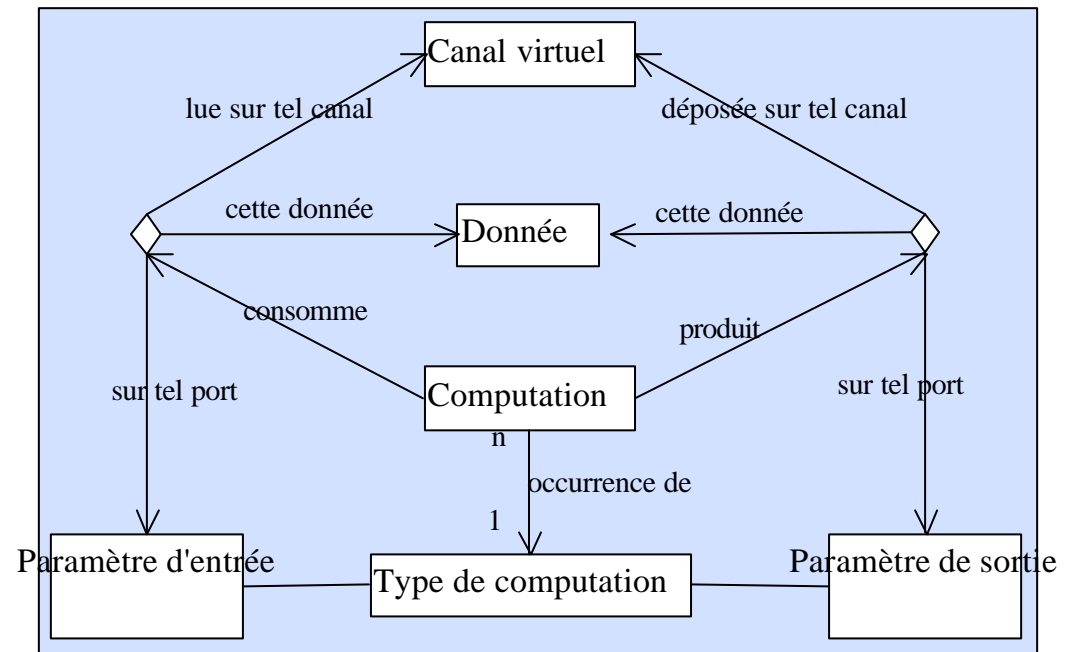
4. Les sous modèles conceptuels de MSol

1.1. Partie "logique" du squelette (ML)

Objectif : permettre la description de l'architecture fonctionnelle d'un système avionique en termes de :

- **computations** et de **types de computation**
- **données logiques** (échangées entre les computations)
- **canaux virtuels** entre les computations

=> indépendamment de l'architecture matérielle



Exemple : extrait du sous-modèle conceptuel ML

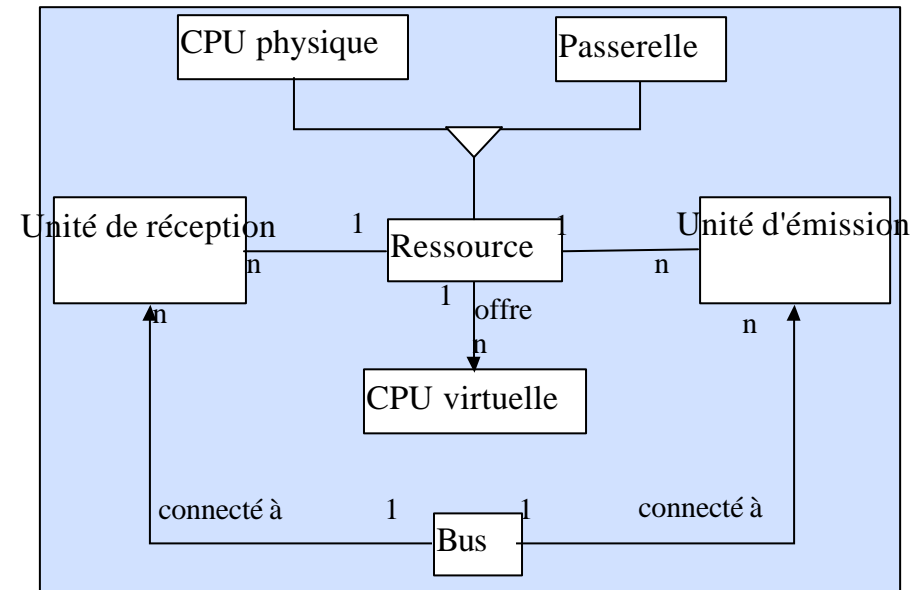
4. Les sous modèles conceptuels de MSol

1.2. Partie "matérielle" du squelette (MA)

Objectif : permettre la description de l'architecture matérielle d'un système avionique en termes de :

- **CPU physiques** et **unités d'émission** et **unités de réception** bus
- **CPU virtuelles** (offertes par les CPU physiques)
- **bus** et **ressources de communication** (passerelles, switchs...)

=> indépendamment de l'architecture fonctionnelle



Exemple : extrait du sous-modèle conceptuel MA

4. Les sous modèles conceptuels de MSol

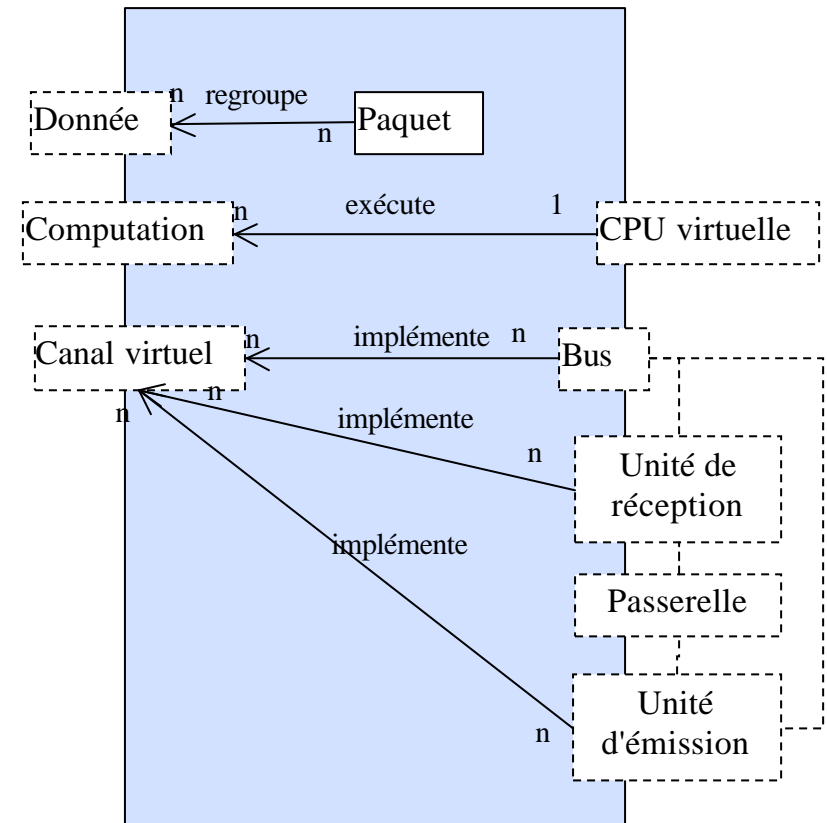
1.3. Partie "intégration" du squelette (MI)

Objectif : permettre la description des choix de placement de computations et des canaux virtuels de ML sur les objets de MA en termes de :

- **paquets** (regroupement de données)
- **relations d'allocation** (des computations sur les CPU virtuelles, et des canaux virtuels sur les bus, passerelles...)

=> couplage fonctionnel matériel

=> offre une vue globale du système pour la vérification (par exemple) de propriétés fonctionnelles et temps réel



Exemple : extrait du sous-modèle conceptuel MI

4. Les sous modèles conceptuels de MSol

2. Chair de la vue "Fonctionnelle" : modèle MFct

MFct = objets et attributs fonctionnels associés aux éléments de « ML » :

- **types** de données abstraits
 - => typage des « données logiques »
 - => typage de la signature des « computations-types »
- **codes** des « computations-types » (Lustre, Esterel)
- **habillage** des « computations » (cycles, conditions d'activation...)
- définition des **hypothèses** de fonctionnement

=> sémantique locale synchrone (Lustre et Esterel)

=> mais, sémantique globale asynchrone

(prise en compte de délais de communication éventuellement variables mais bornés entre les computations)

facette fonctionnelle = ML + MFct

4. Les sous modèles conceptuels de MSol

3. Chair de la vue "Performance Temps Réel" : modèle MPTR

MPTR = objets et attributs de performances associés aux éléments de « MA », et règles d'occupation des ressources

- **taille** des paquets transitant sur les bus
- **taille** des computations
- **performances** des calculateurs et des bus
- **durée** des créneaux de temps de chaque CPU virtuelle...

=> **évaluation et dimensionnement des performances temps-réel du système par des techniques de simulation ou des techniques analytiques**

(évaluation des latences, de leur variations, identification de goulots d'étranglement...)

facette performance temps réel = MA + MI + MPTR

5. A partir de MSOL...

A partir de la vue MPTR et du squelette...

analyse et détermination des latences à travers le réseau avionique

=> par simulation :

- modélisation en **SDL** du comportement des objets architecturaux (CPU, OS, protocoles...)
- modélisation en **SDL** d'une abstraction orientée temps du comportement des computations et des communications

=> simulation et mesure des latences observées

=> par vérification paramétrée :

- modélisation en **Automates Temporisés** du comportement des objets architecturaux (CPU, OS, protocoles...)
- modélisation en **Automates Temporisés** d'une abstraction orientée temps du comportement des computations et des communications

=> vérification de propriétés de latences par "**model checking**"

=> détermination des latences par "**model checking paramétré**"

5. A partir de MSOL...

A partir de la vue MFct, du squelette, et de la détermination des latences...

analyse et vérification du comportement fonctionnel global du système avionique
=> preuve des exigences fonctionnelles

=> par "model checking" :

- modélisation des computations et des canaux virtuels entre computations dans un **langage synchrone** (y compris les retards observés sur les canaux virtuels)
- => vérification par "**model checking**" (LESAR, NP-tools, SMV, ...)

=> par résolution de contrainte :

- modélisation (d'une abstraction) du système global sous la forme de **variables, d'équations et inéquations** sur ces variables

=> **un ensemble de contraintes**

=> exigences à démontrer = **une autre contrainte P**

=> démonstration **par résolution** qu'il est impossible de satisfaire les contraintes modélisant le système et la contrainte "non P"

=> **technique qui peut marcher à l'envers** (détermination des "bonnes" latences de telle sorte qu'une exigence soit satisfaite)

6. Exemple...

Fonction de l'application :

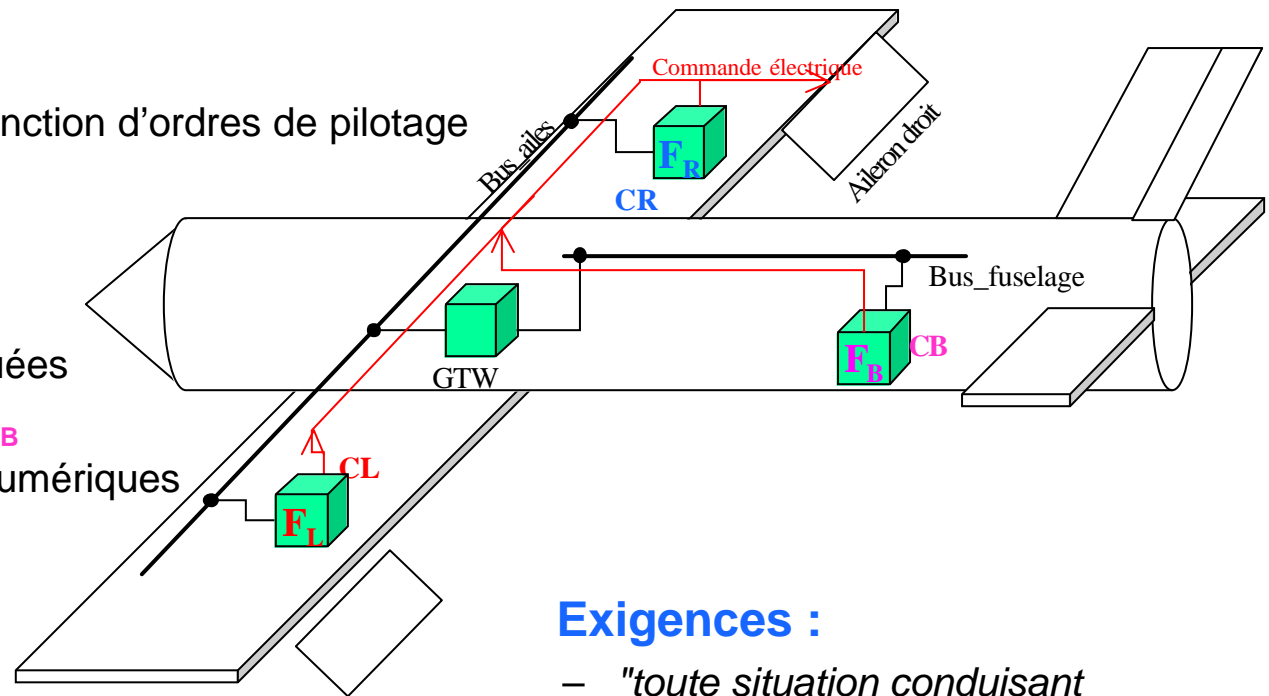
- asservir l'aileron droit en fonction d'ordres de pilotage reçus du pilote

Composition l'application :

- 3 fonctions F_R , F_L , F_B , allouées sur 3 calculateurs C_R , C_L , C_B interconnectés par 2 bus numériques

Principes fonctionnels :

- F_R produit un ordre de gouverne vers l'aileron droit tant que C_R n'est pas défaillant
- F_L produit un ordre de gouverne si celle-ci n'a rien reçu de F_R depuis 2 cycles (40 ms)
- F_B produit un ordre de gouverne si celle-ci n'a rien reçu de F_R et F_L depuis 5 cycles (100 ms)



Exigences :

- "toute situation conduisant l'aileron droit à être commandé par plus de deux fonctions au même instant doit être extrêmement improbable"
- "toute situation conduisant l'aileron droit à ne plus être commandé pendant plus de 160 ms doit être extrêmement improbable"

6. Exemple...

L'exemple dans les termes (simplifiés) des modèles conceptuels...

- ML {
- FR (computation), instance de F (type de computation)
 - produit ordre (donnée) sur canal0 (canal virtuel)
 - produit ordre (donnée) sur canal1 (canal virtuel)
 - produit ordre (donnée) sur canal2 (canal virtuel)
 - FL (computation), instance de F (type de computation)
 - consomme ordre (donnée) sur canal1 (canal virtuel)
 - produit ordre (donnée) sur canal0 (canal virtuel)
 - produit ordre (donnée) sur canal1 (canal virtuel)
 - FB (computation), instance de F (type de computation)
 - consomme ordre (donnée) sur canal1 (canal virtuel)
 - consomme ordre (donnée) sur canal2 (canal virtuel)
 - produit ordre (donnée) sur canal0 (canal virtuel)
 - gouverne (actionneur) consomme ordre (donnée) sur canal0 (canal virtuel)

- MA {
- CR (CPU physique) connecté à bus_aile (bus)
 - ...

- MI {
- FR alloué à CR
 - ...

**+ chairs (infos comportementales) fonctionnelle
et temps réel...**

6. Exemple...

A partir de la description de l'exemple...

=> formalisation en automates temporisés => latences sur les canaux

latence canal0 \hat{I} [0, 0]

latence canal1 \hat{I} [0, 20ms]

latence canal0 \hat{I} [0, 40ms]

=> formalisation de la vue fonctionnelle + valeur des latences + comportement asynchrone en LUSTRE => vérification de la propriété :

"sous l'hypothèse que seuls CR, CL et CB peuvent tomber en panne, et sous l'hypothèse que seule deux pannes sont possibles au cours d'un vol, alors :

**l'aileron ne sera jamais commandé par deux fonctions simultanément,
et l'aileron ne restera pas sans être commandé plus de 160 ms"**

=> construction d'un **argumentaire** montrant que les deux exigences initiales sont tenues...

4. Conclusion

Ce qui a été fait :

- des modèles conceptuels qui permettent de capturer :
 - les informations manipulées dans un développement
 - les relations de dépendances entre ces informations
- des techniques d'analyse et de vérification sur des instances de ces modèles conceptuels
- l'ensemble des modèles conceptuels a été formalisé, puis exploités, en TELOS - PROLISP

Mais, ce qui n'a pas été fait

- seuls les modèles conceptuels du squelette, des exigences, des argumentations, des résultats... ont été décrits dans un formalisme à la UML
- seule la notion de modèle de classes de UML a été utilisée
- => les informations dynamiques sont reportées et manipulées hors de UML

- les modèles conceptuels n'ont pas été intégrés dans un outil commercial
- et le lien avec les outils de vérification (UPPAAL, NP-tools, LESAR, SMV...) n'a pas été automatisé (mais la faisabilité de l'automatisation a été démontrée)