

A small image of the Earth as seen from space, showing the blue oceans and white clouds against the black background of space, located in the top left corner of the slide.

# Les ADL du point de vue de l'industrie

—  
**AADL**

**Jean-François Tilman**  
**Axlog Ingénierie**

# Introduction

- Augmentation des besoins lors du développement de systèmes embarqués.
- Recherche de solutions par l'industrie pour y répondre. Les ADL sont un élément de réponse.
- AADL, un ADL intéressant.
- Illustration par quelques projets de recherche industriels.

# Partie I

## Besoins et réponses par les ADL

# Tendances et besoins (1/3)

- Augmentation de la taille des systèmes embarqués

Airbus A300B  
(1974)  
25 ko



Airbus A380  
(2005)  
64 Mo

Spot 1  
(1980)  
48 ko



Mars Express  
(2003)  
1,2 Mo

Automobile haut de gamme  
(2003)  
plus de 30 ECU  
500 000 lignes de code  
(sécurité, environnement, confort, équipement)

# Tendances et besoins (2/3)

- Augmentation de la complexité
  - fonctions distribuées
  - interactions entre fonctions et avec l'environnement extérieur
- Maîtrise de la criticité
  - exigence de fiabilité toujours plus élevée
  - introduction de fonctions plus critiques (*X-by-wire* dans l'automobile)
  - quelques échecs ont mis en lumière les problèmes des pratiques actuelles
  - « *software crisis* » identifiée par l'ESA

# Tendances et besoins (3/3)

- Besoins d'interopérabilité
  - nombreux intervenants dans le développement
  - communication entre composants de fournisseurs concurrents
- Diminution des coûts et délais
  - Secteurs très concurrentiels
  - Marché réactif
  - => réutiliser les développements antérieurs
- **Conclusion** : de nombreux besoins contradictoires

# Éléments de solution (1/2)

- Utiliser un langage commun pour décrire le système
  - pour communiquer entre tous les intervenants
  - => utiliser des standards
  - pour éviter les incompréhensions
  - => disposer d'une sémantique forte et précise
- Prendre en compte le niveau *système*
  - pour maîtriser les contraintes entre logiciel et matériel
  - pour limiter les problèmes d'intégration
  - => décrire les interfaces entre matériel et logiciel

# Éléments de solution (2/2)

- Utiliser des méthodes formelles et des preuves
  - pour une meilleure qualité
  - pour limiter les efforts de tests
  - => décrire formellement le système
- Automatiser les phases de développement
  - génération automatique de code, de tests,
  - utilisation d'outils d'analyse
  - => descriptions traitables par ordinateur.
- Réutiliser les composants développés
  - => constituer des bibliothèques de composants

# Support par des ADL

- Intérêts
  - langage informatique => traitement automatique possible
  - description formelle => support possible de méthodes formelles et de preuves
  - peut décrire le niveau système
  - peut être standardisé => diffusion facilitée
- **Conclusion** : Les ADL sont une base possible pour construire un processus de développement amélioré.
- Ils ne suffiront pas, il faut aussi définir un processus.

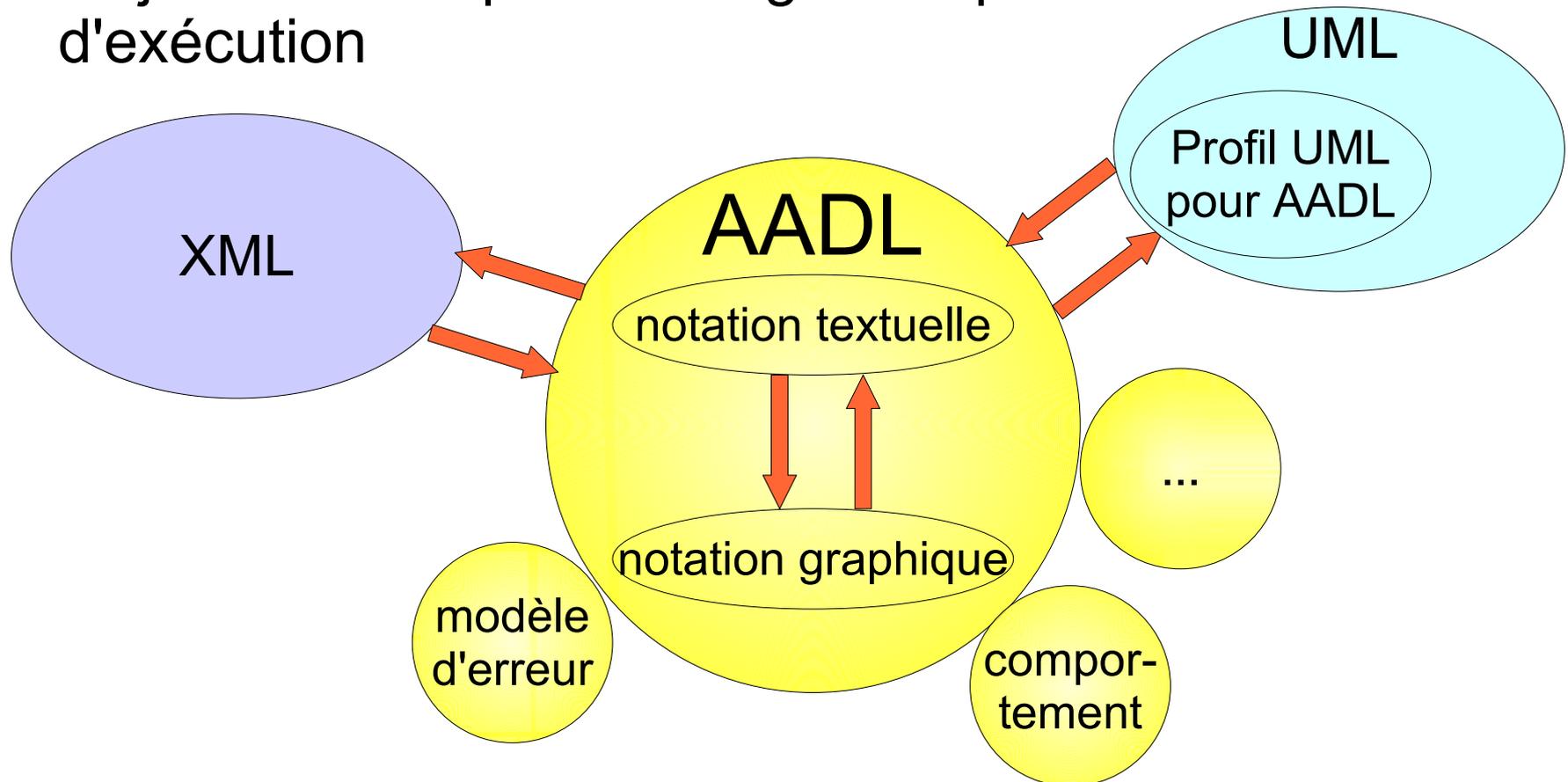
# Partie II

## AADL



# Description générale

- AADL = *Architecture analysis & design language*
- Objectif : description du logiciel + plate-forme d'exécution



# Historique

- 1991 MetaH, développé par Honeywell pour AMCOM
- 2000 lancement de la standardisation AADL (*Avionics Architecture Description Language*), basé sur MetaH
- 2003 changement de nom (*Architecture Analysis & Design Language*) pour ne pas le limiter au domaine avionique
- 2004 publication de la version 1.0 (ref. SAE AS5506)
- 2005 publication d'annexes au standard
- 2006 AADL version 2

# Standardisation

- Comité de standardisation international, sous l'autorité du SAE (Society of automotive engineers), division systèmes avioniques.
- Les membres du comité sont américains et européens :
  - SEI, Honeywell, Dassault Aviation, Lockheed Martin, EADS, Raytheon, Smith Industries, Axlog Ingénierie, Rockwell, Airbus, NAVAir, British MOD, US Army, Agence spatiale européenne, Boeing, TNI-Europe...

# Concept de composant

- Les composants sont les éléments de base d'une architecture AADL.
- Ils sont :
  - hiérarchisés
  - composés
  - interconnectés
- Ils appartiennent tous à l'une des 10 catégories prédéfinies.



# Catégories de composants

## catégories logicielles



process



subprogram



data



thread



thread group

## catégorie composite



system

## catégories plate-forme



processor



memory



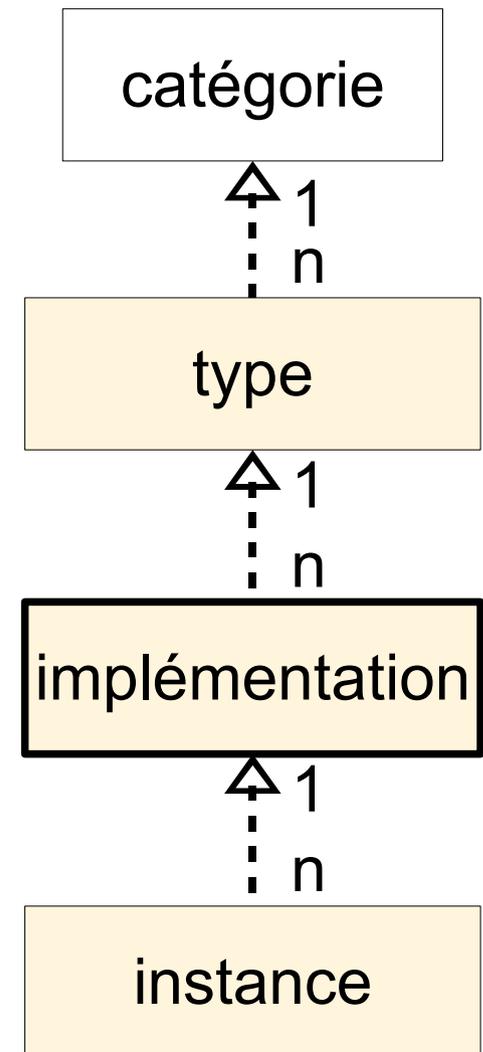
device



bus

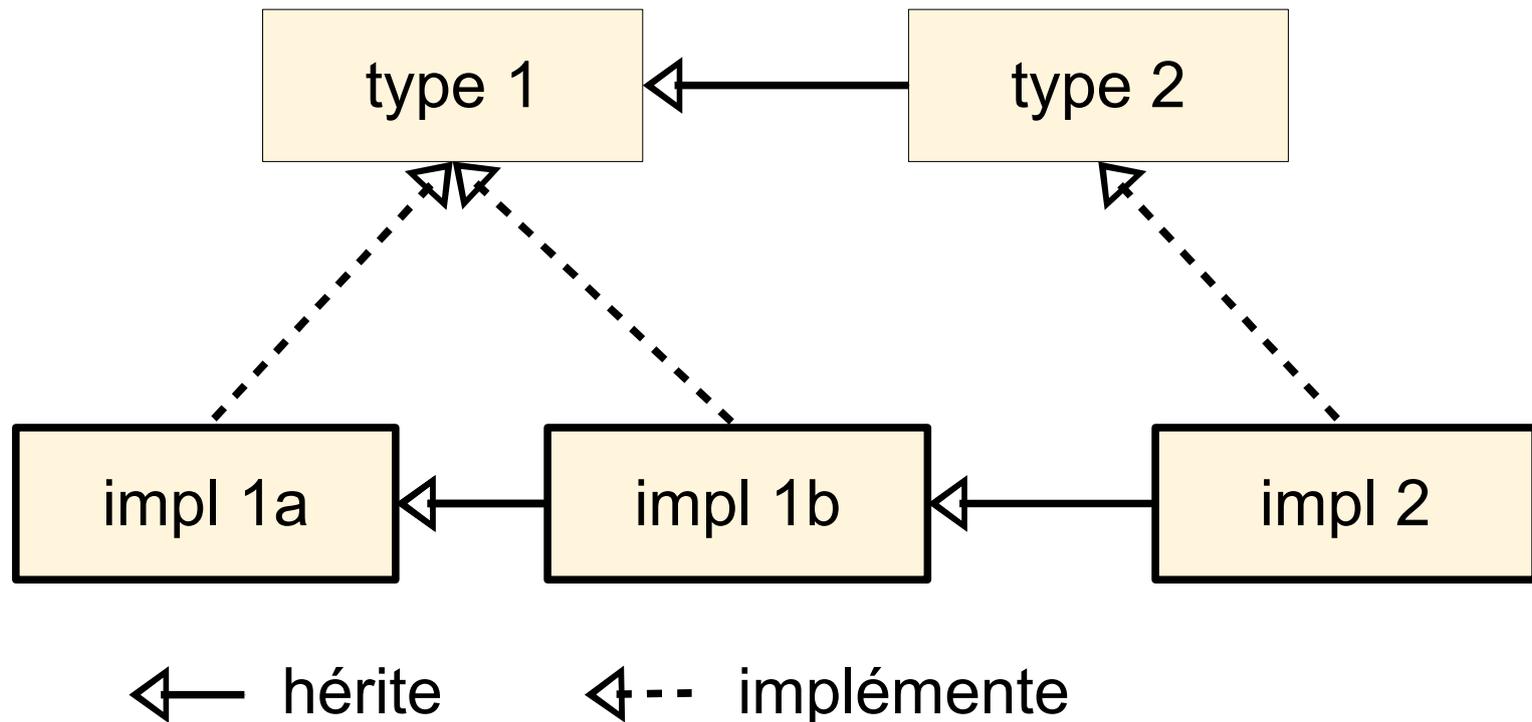
# Trois niveaux de description

- Catégorie (prédéfinie)
- Type
  - spécification de l'interface externe
- Implémentation
  - spécification du contenu
- Instance
  - instanciation d'un type ou d'une implémentation



# Héritage

- Un type peut hériter d'un autre type
- Une implémentation peut hériter d'une autre implémentation.

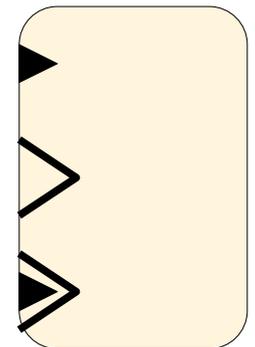
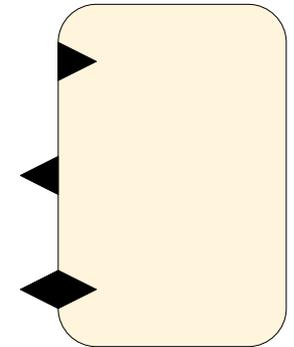


# Concept de caractéristique

- Une « caractéristique » (*feature*) est un élément de type  $_{AADL}$  de composant qui spécifie comment ce composant s'interface avec les autres composants du système.
- Quatre catégories de caractéristiques :
  - port ;
  - accès à un sous-composant ;
  - sous-programme ;
  - paramètre.

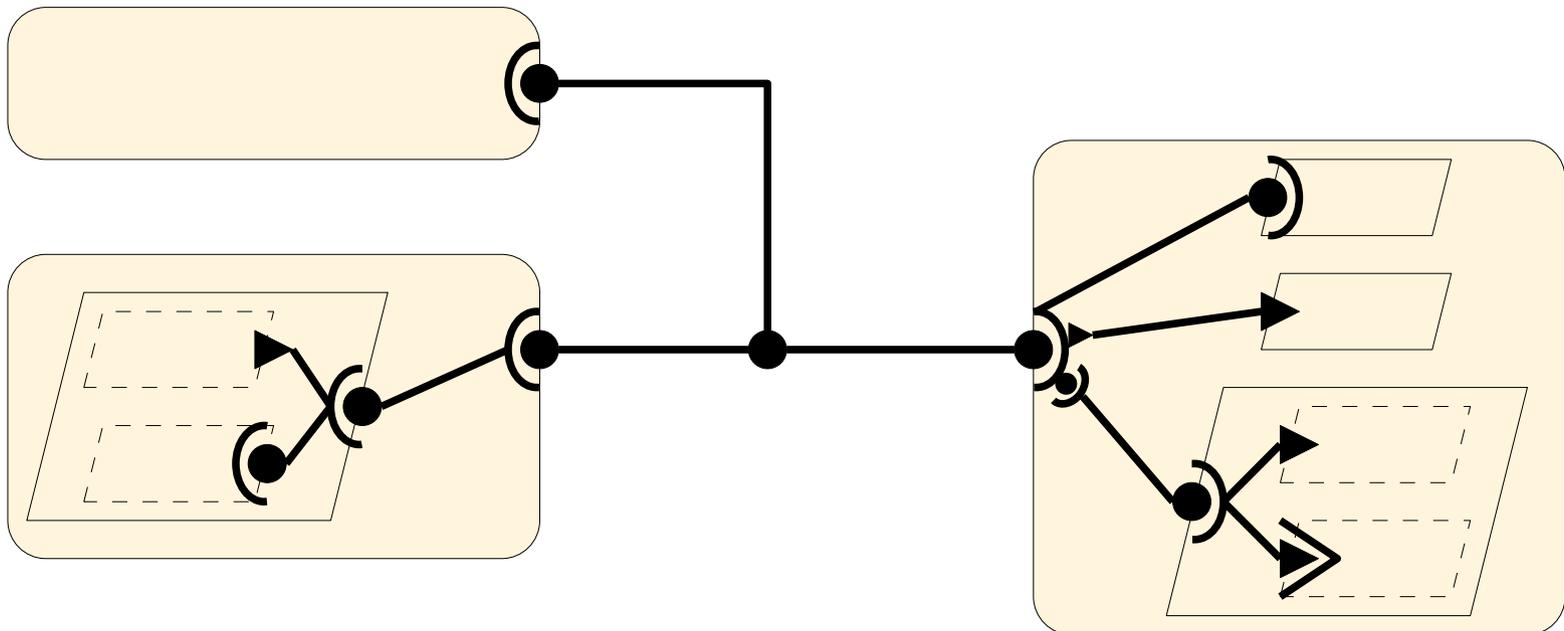
# Port

- Un « port » représente une interface de communication pour échanger des données ou des événements entre composants.
- Trois directions :
  - port entrant (*in*) ;
  - port sortant (*out*) ;
  - port bidirectionnel (*in out*).
- Trois types de ports :
  - port donnée (*data port*) ;
  - port événement (*event port*) ;
  - port événement-donnée (*event data port*).



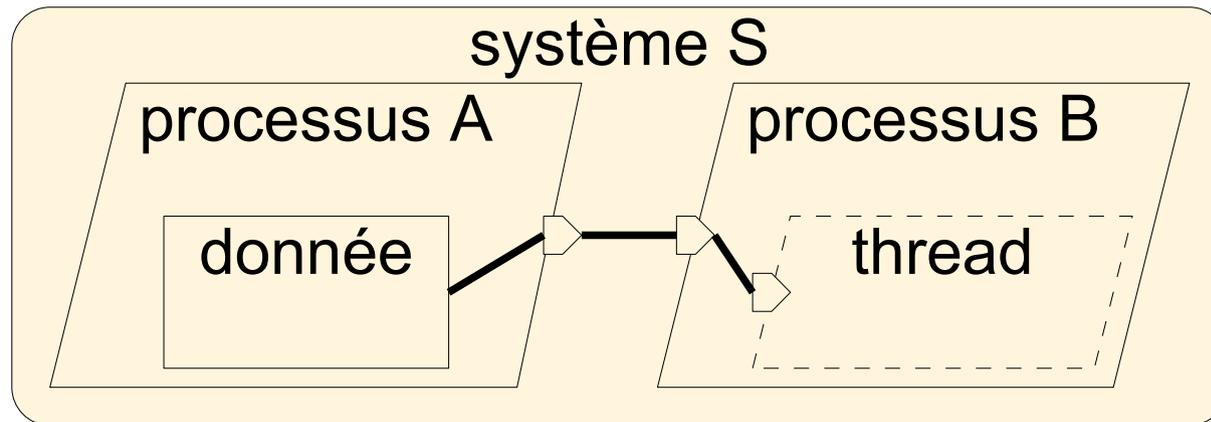
# Groupe de ports

- Les groupes de ports (*port group*) peuvent regrouper des ports et d'autres groupes de ports.
- Intérêt : ne pas multiplier les ports pour représenter, par exemple, un bus parallèle.

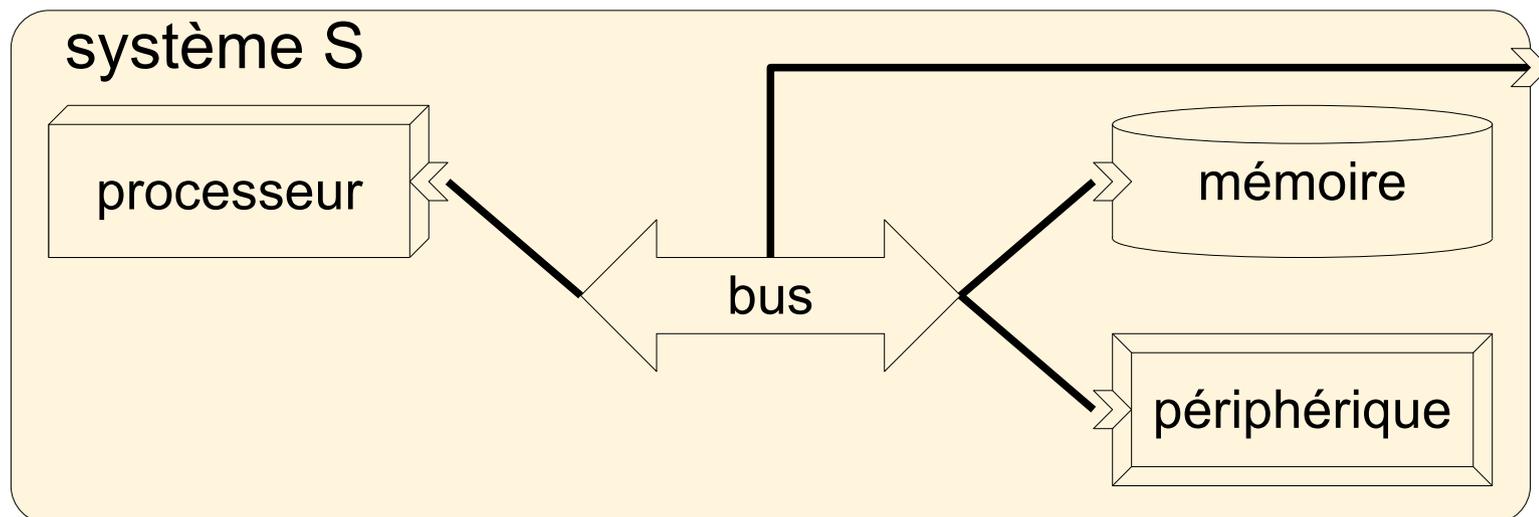


# Accès à un sous-composant

- Accès à une donnée



- Accès à un bus



# Sous-programme et paramètres

- Une caractéristique « sous-programme » (*subprogram*) représente :
  - soit un point d'entrée d'exécution du code source qui opère sur un composant donnée (*data subprogram*) ;
  - soit un point d'entrée pour un appel de procédure à distance (*server subprogram*).
- Une **caractéristique** sous-programme fait référence à un **composant** sous-programme.
- Un « paramètre » représente un argument de sous-programme.

# Concept de propriété

- Une « propriété » (*property*) fournit une information sur un composant, une caractéristique, une connexion, un mode ou un appel de sous programme.
- Une propriété est définie par :
  - un nom ;
  - un type ;
  - une valeur.
- Le type de propriété définit l'ensemble des valeurs acceptables.
- On peut définir de nouvelles propriétés dans un *property set*.



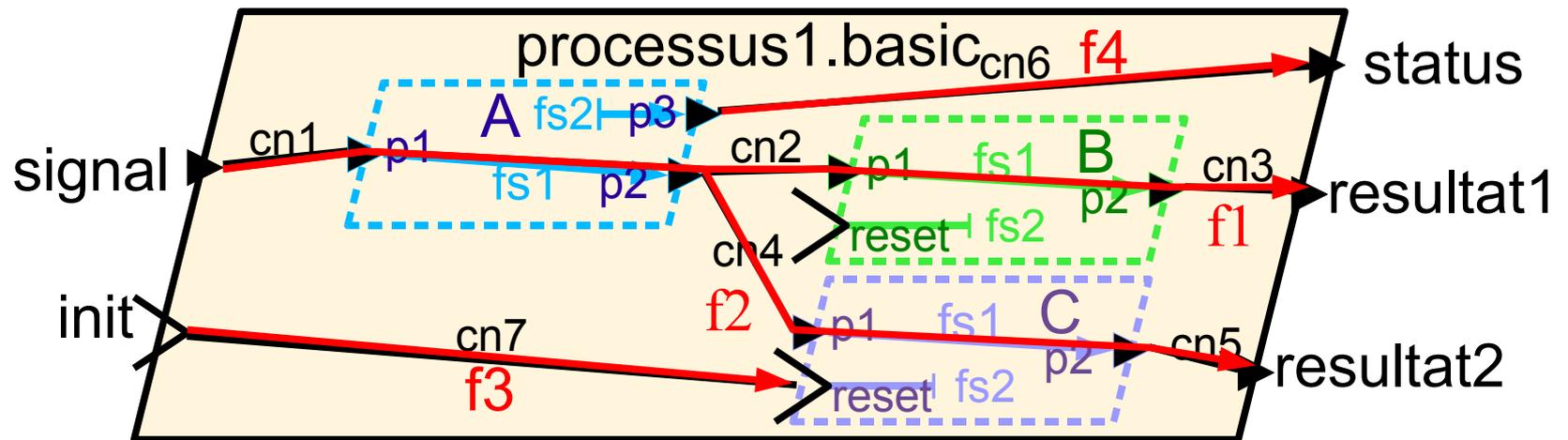
# Ensembles de propriétés prédéclarés

- Deux ensembles de propriétés sont prédéclarés :
  - **AADL\_Properties** définit des propriétés communes à toutes les spécifications AADL;
  - **AADL\_Project** définit des types et des constantes énumérés, dont les valeurs peuvent différer d'un projet à l'autre.
- Ces deux ensembles de propriétés font implicitement partie de toute spécification AADL.
- Les types et constantes de **AADL\_Project** peuvent être modifiées.

# Flux

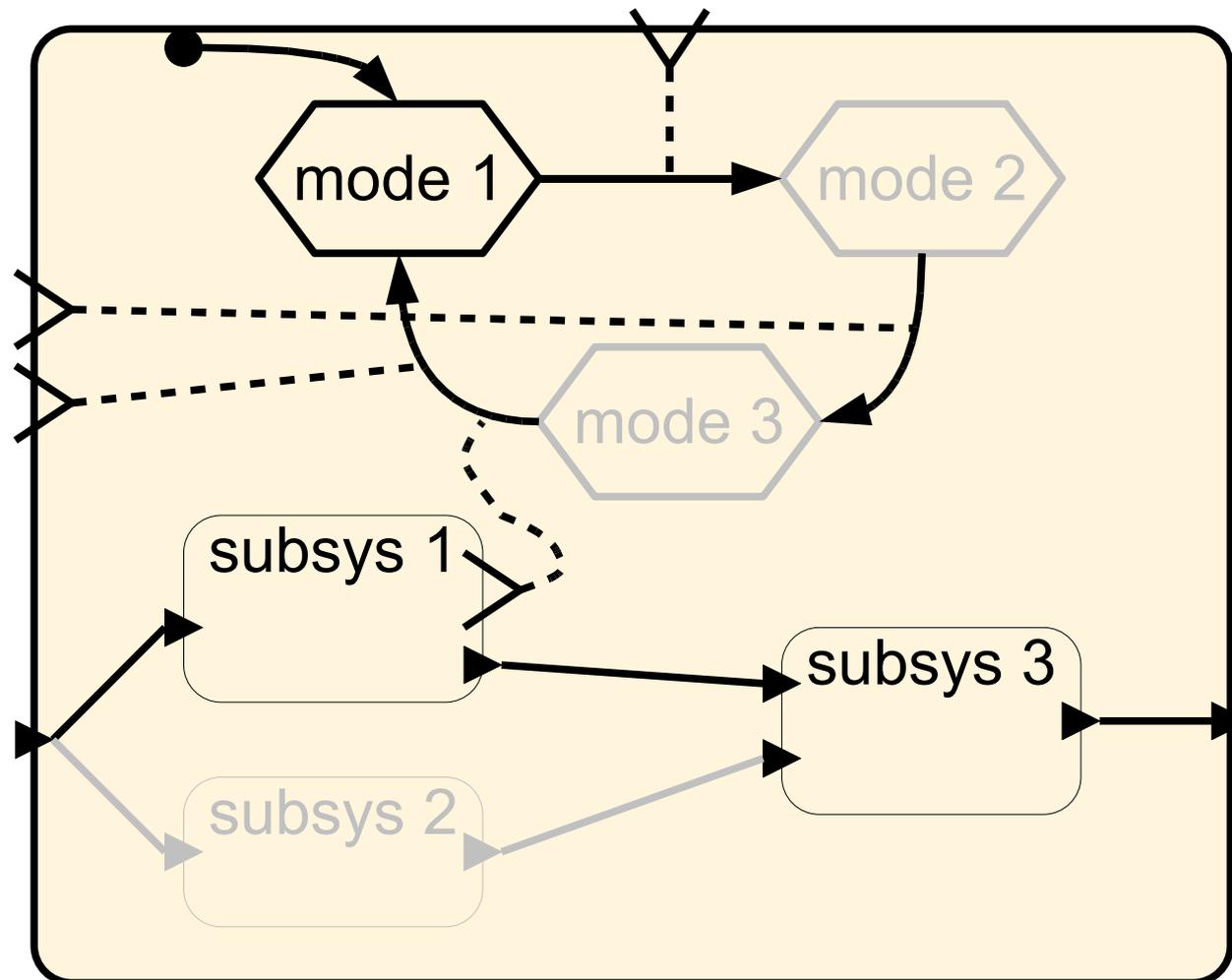
- Pouvoir spécifier des flux de bout en bout permet des analyses temporelles, de fiabilité, de propagation d'erreur, de qualité de service, etc.
- La description de flux complets est composé de plusieurs sortes de déclaration :
  - spécification de flux ;
  - implémentation de flux ;
  - déclaration de bout en bout.

# Exemple de flux



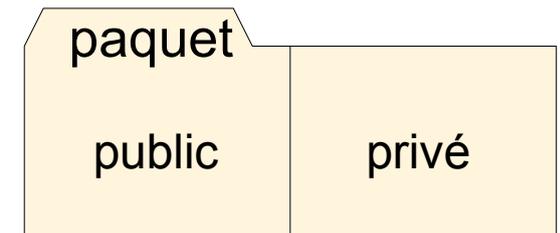
# Mode

- Un mode représente un état opérationnel d'un système.



# Paquets

- Un paquet (*package*) fournit le moyen d'organiser les descriptions en introduisant des espaces de noms.
- Un paquet peut contenir :
  - des types de composants ;
  - des implémentations de composants ;
  - des bibliothèques annexes.
- Le contenu de la section *public* est visible hors du paquet
- Le contenu de la section *private* n'est pas visible hors du paquet.



# Annexe

- Une annexe permet d'utiliser des déclarations exprimées dans un autre langage qu'AADL.
- Usage principal : support de nouvelles méthodes d'analyse ou de description de nouveaux aspects.
- Une clause annexe peut être utilisée dans un type ou une implémentation de composant.
- Une bibliothèques d'annexes peut être déclarée dans un paquet.
- Exemple : introduction de contraintes avec OCL.

```
{** nom annexe **}
```

# Partie III

## Exemples de projets

# Mises en œuvre

- Plusieurs projets de recherche français et européens impliquant des partenaires industriels développent ces approches :
  - définition de nouveaux processus de développement ;
  - introduction d'ADL au niveau du système embarqué ;
  - utilisation de méthodes formelles et de preuves.

# Assert

- Projet intégré européen du 6<sup>e</sup> PCRD, orienté aéronautique et spatial
- Objectif : définir un processus d'ingénierie système amélioré, basé sur l'utilisation de preuves et de méthodes formelles.
- Choix techniques :
  - PBSE (*Proof-based system engineering*) : continuité des preuves durant tout le cycle de développement ;
  - Utilisation d'AADL + extensions tout au long du cycle.
- <http://www.assert-online.net>

# EAST-EEA

- Projet européen ITEA reprenant le projet français AEE, dans le domaine de l'automobile
- Objectif : concevoir un processus pour la définition de l'architecture système et le développement des logiciels associés
- Un ADL a été défini, *EAST-ADL*, basé sur *AIL-transport* venant d'AEE.
- EAST-ADL et AADL ont des similarités. Le premier est dédié au domaine automobile alors que le second est plus généraliste.
- <http://www.east-eea.net>

# Cotre

- Projet français (RNTL) avec Airbus.
- Objectif: définir une démarche outillée pour modéliser et valider des architectures logicielles temps réel combinant des approches formelles dans un processus industriel.
- Utilise AADL et propose une extension pour supporter les informations comportementale.
- Ce travail sert de base au comité AADL pour définir une annexe comportementale standard.
- <http://www.laas.fr/COTRE/frindex.html>

# Topcased

- Projet avec Airbus qui prolonge et élargit le périmètre du projet Cotre.
- Objectif : développer un atelier de développement logiciel *open source*, pour répondre aux besoins de maintenance sur dix, vingt ou trente ans.
- La modélisation repose en particulier sur AADL et UML.
- Ce projet illustre la progression des concepts de la recherche vers l'industrie.
- <http://www.topcased.org/>

# Conclusion

- Les ADL sont un élément de solution pour répondre aux besoins de l'industrie.
- Les industriels sont engagés dans des projets de recherche pour les mettre en œuvre.
- Les ADL sont utilisés dans le cadre d'un processus défini.
- AADL est un des ADL les plus prometteurs pour ces domaines.

# Pour en savoir plus

- Comité SAE, « AADL standard », v1.0, ref. AS5506, [http://www.sae.org/servlets/productDetail?PROD\\_TYP=P=STD&PROD\\_CD=AS5506](http://www.sae.org/servlets/productDetail?PROD_TYP=P=STD&PROD_CD=AS5506)
- Site web officiel d'AADL : <http://www.aadl.info>
- Une présentation en français : [http://www.axlog.fr/R\\_d/aadl/](http://www.axlog.fr/R_d/aadl/)
- **AADL workshop 2005**, les 17-18 octobre à Massy : tutorial et présentation de projets autour d'AADL [http://www.axlog.fr/R\\_d/aadl/workshop2005\\_en.html](http://www.axlog.fr/R_d/aadl/workshop2005_en.html)