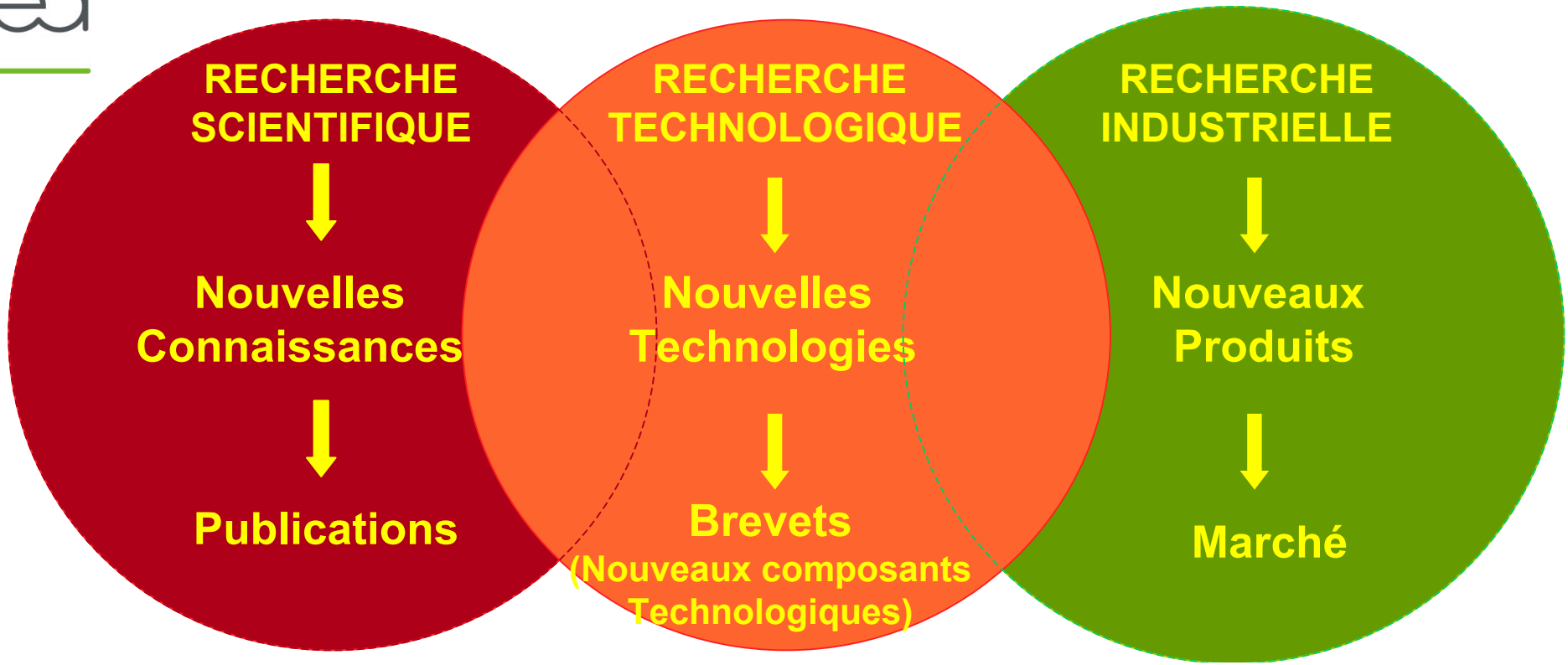
# Méthodes et Outils pour maîtriser la complexité et la sûreté des systèmes

**Contact : Didier JUVIN ➔ didier.juvin@cea.fr**

# Le CEA/LIST : un pôle de Recherche Technologique

## centré sur les systèmes à logiciels prépondérants




350 salariés en île de France,
40 sur le thème présenté :
(Méthodes et Outils pour les systèmes temps réel critiques)

## Des compétences issues du Nucléaire …

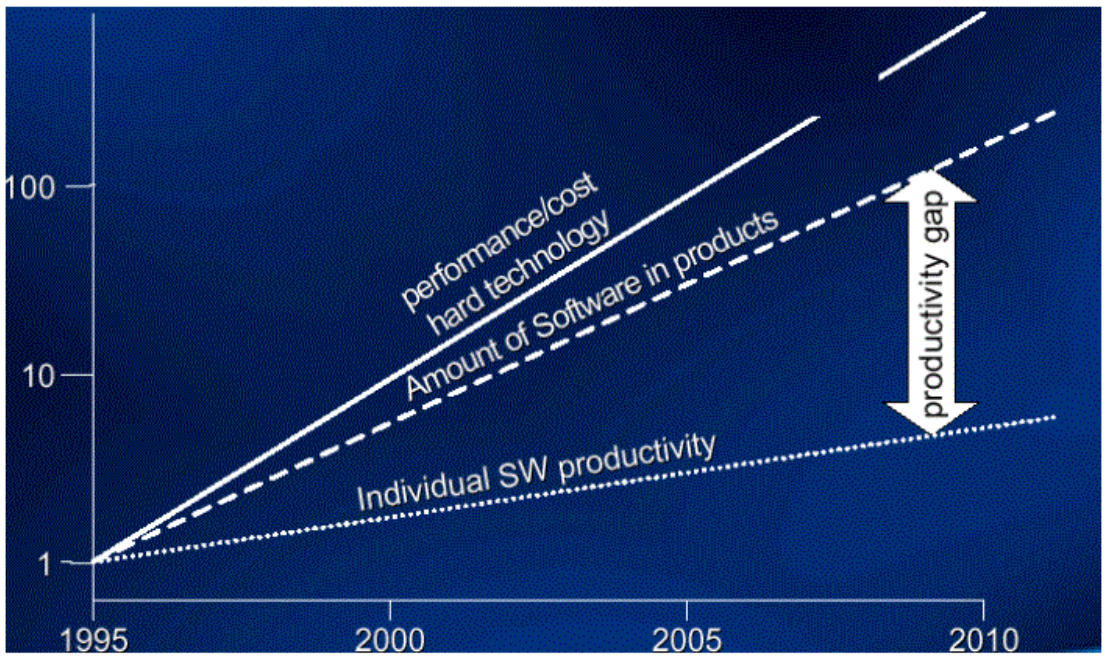
**le SPIN**
**(Système de Protection Intégré Numérique)**
**est un système critique opérationnel**
**depuis 1982**

***Le CEA / LIST a développé CLAIRE, un outil de test système transféré à Duons-Systèmes***

# Méthodes et outils pour maîtriser la complexité et la sûreté des systèmes

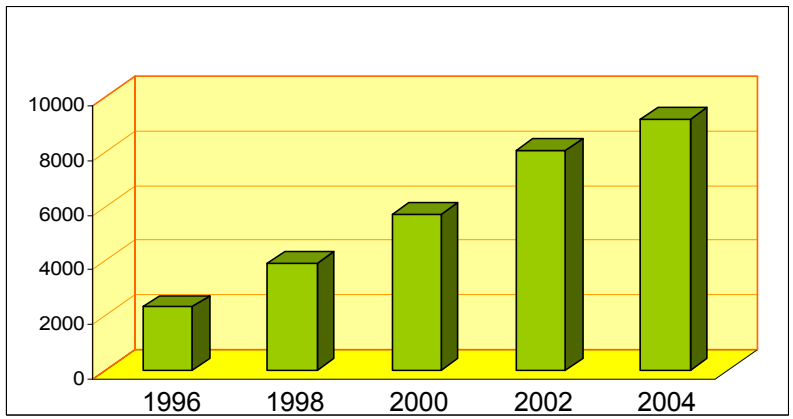
Objectif : offrir des gains de productivité dans le développement du logiciel et dans la maîtrise du cycle de vie :

➔ passer de l'état artisanal actuel à des modes de production industriels, sans rupture de culture




ITEA roadmap March 2001

Coût des logiciels embarqués M$ (source BCC)

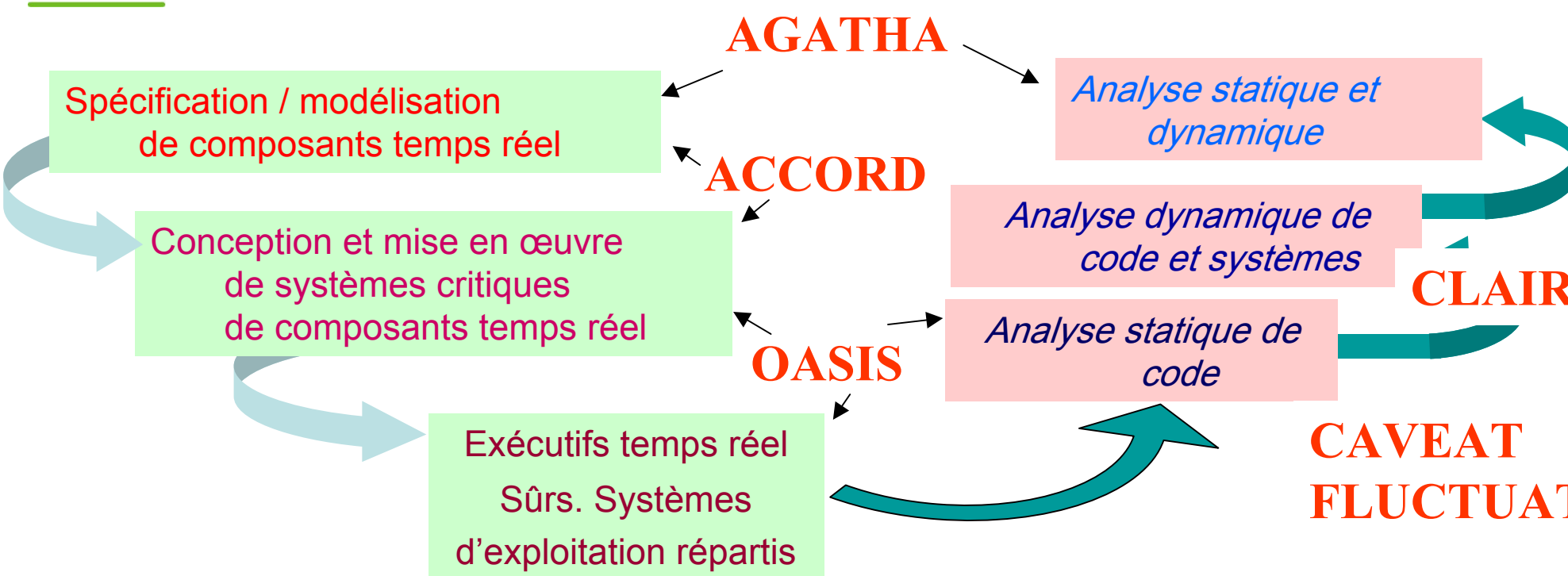


# Thèmes de recherche :

- **Génération de scénarii de test à partir de langages d'automates communicants,**
- **Modélisation de systèmes de composants logiciels,**
- **Génération de code exécutable,**
- **Analyse statique de code,**
- **Analyse dynamique de code exécutable…**

**➔ Sûreté de fonctionnement des Systèmes Complexes**

# Offrir des gains de productivité et maîtriser la sûreté à toutes les étapes du cycle de vie du logiciel…



# 1/ Analyse et vérification de spécifications de systèmes industriels complexes

**Mise en œuvre à l'échelle industrielle de techniques avancées d'analyse de spécifications et génération de stratégies de tests**

- **Exploitation des principaux formats industriels (SDL, UML, statecharts...)**

  ➔ ***Outils supports :*** **AGL ObjectGeode, Statemate, Objecteering**

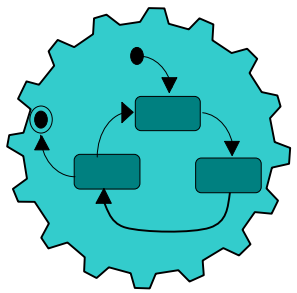
- **Identification de chaque comportement du système par analyse de son modèle,**

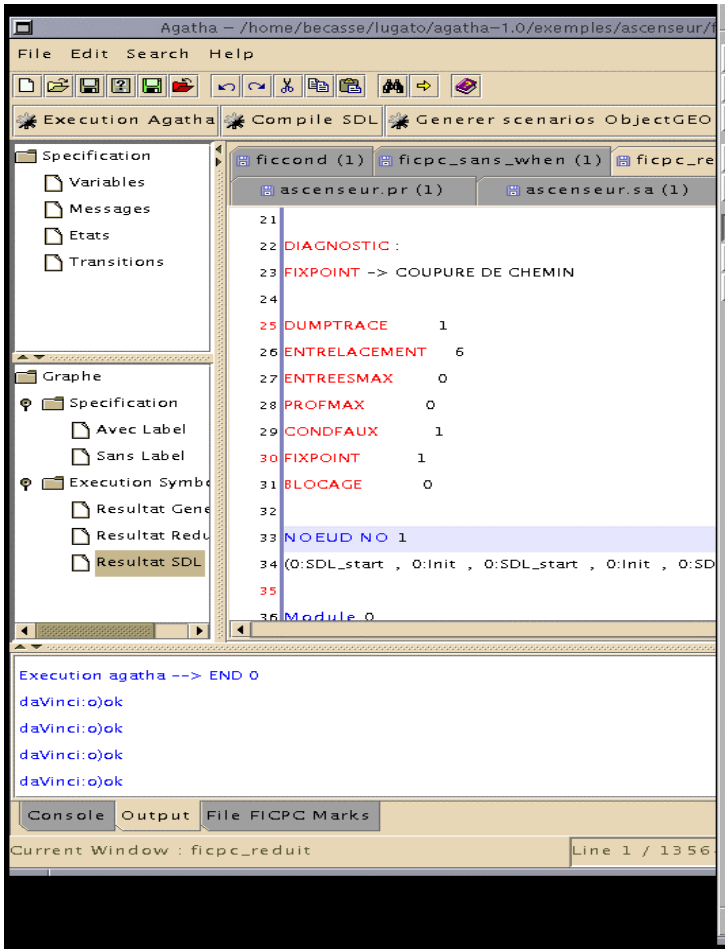
- **Production des vecteurs de test associés**

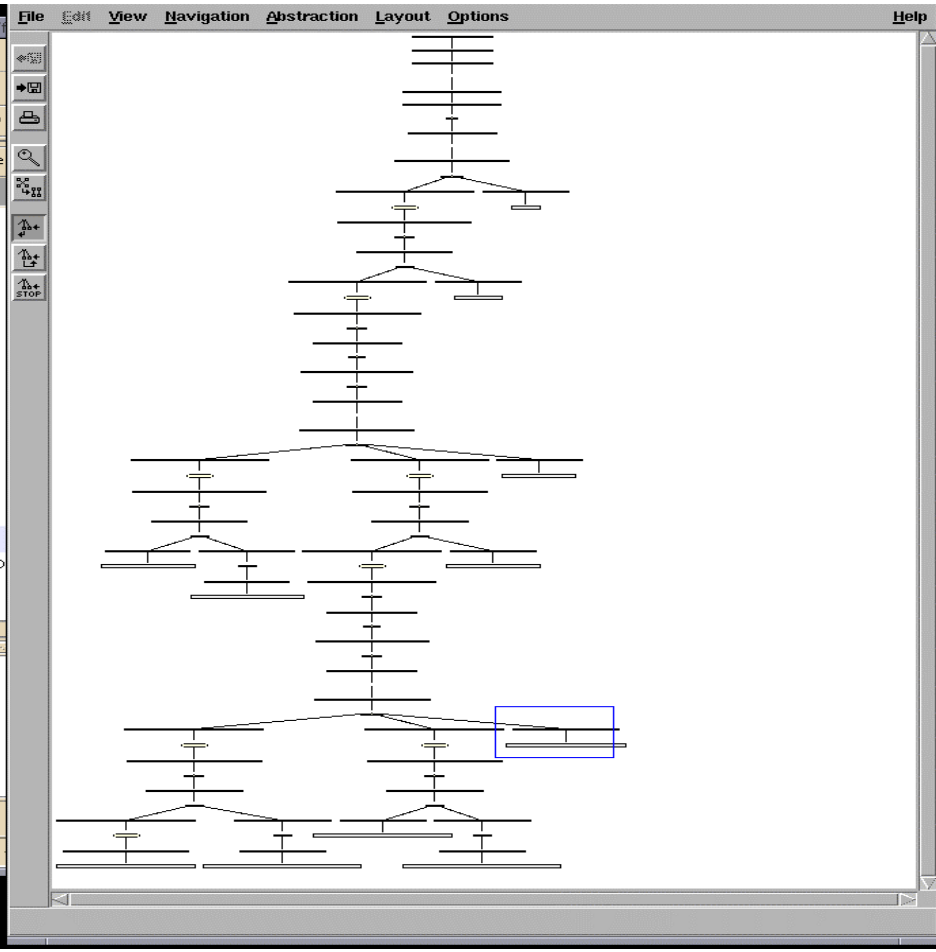
***Outil AGATHA***

**EDF**

**PSA (fonctions habitacle,…),**
**EADS (Ariane5), Trusted Logic**





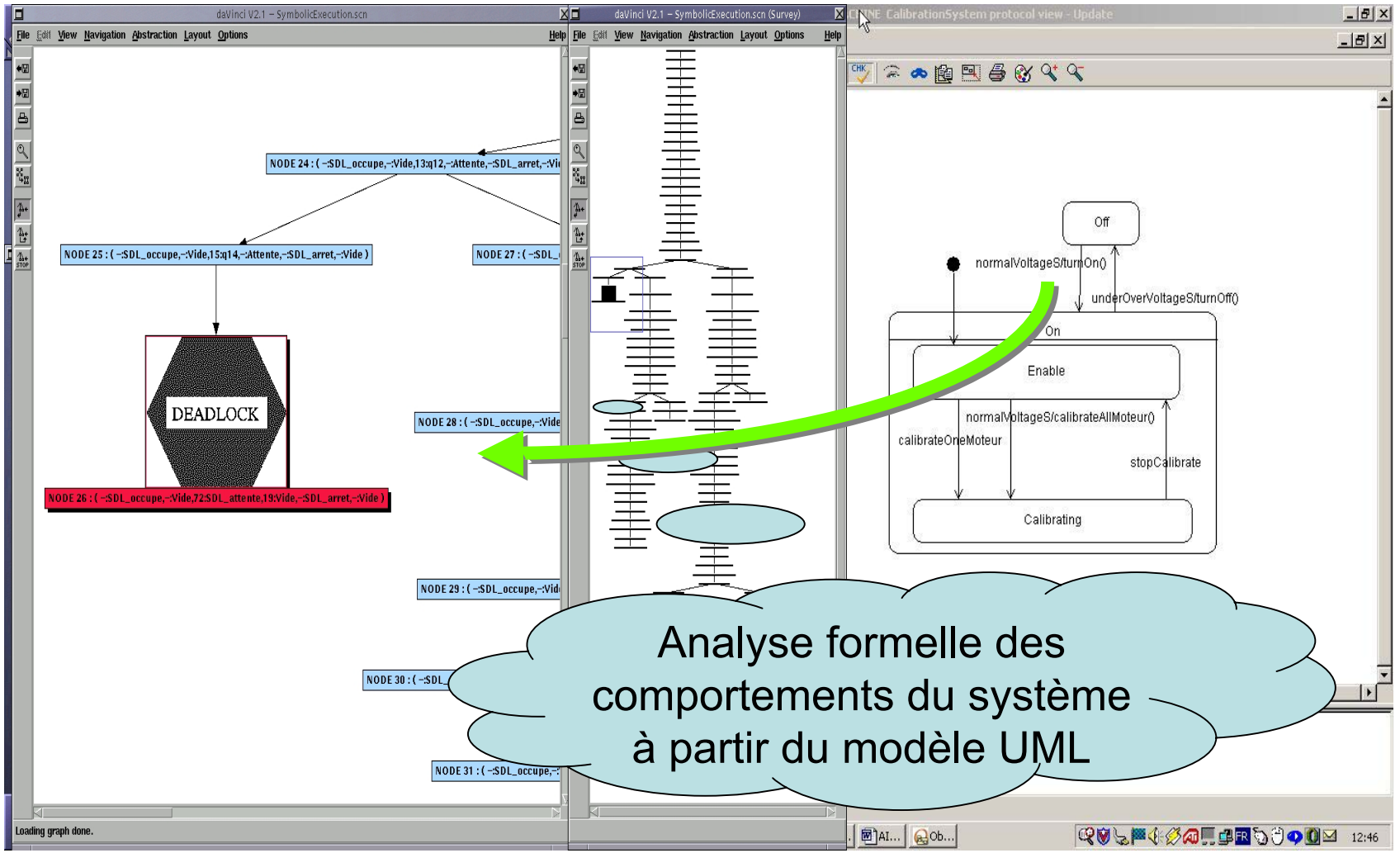




# Outil AGATHA

## Analyse et validation de spécifications

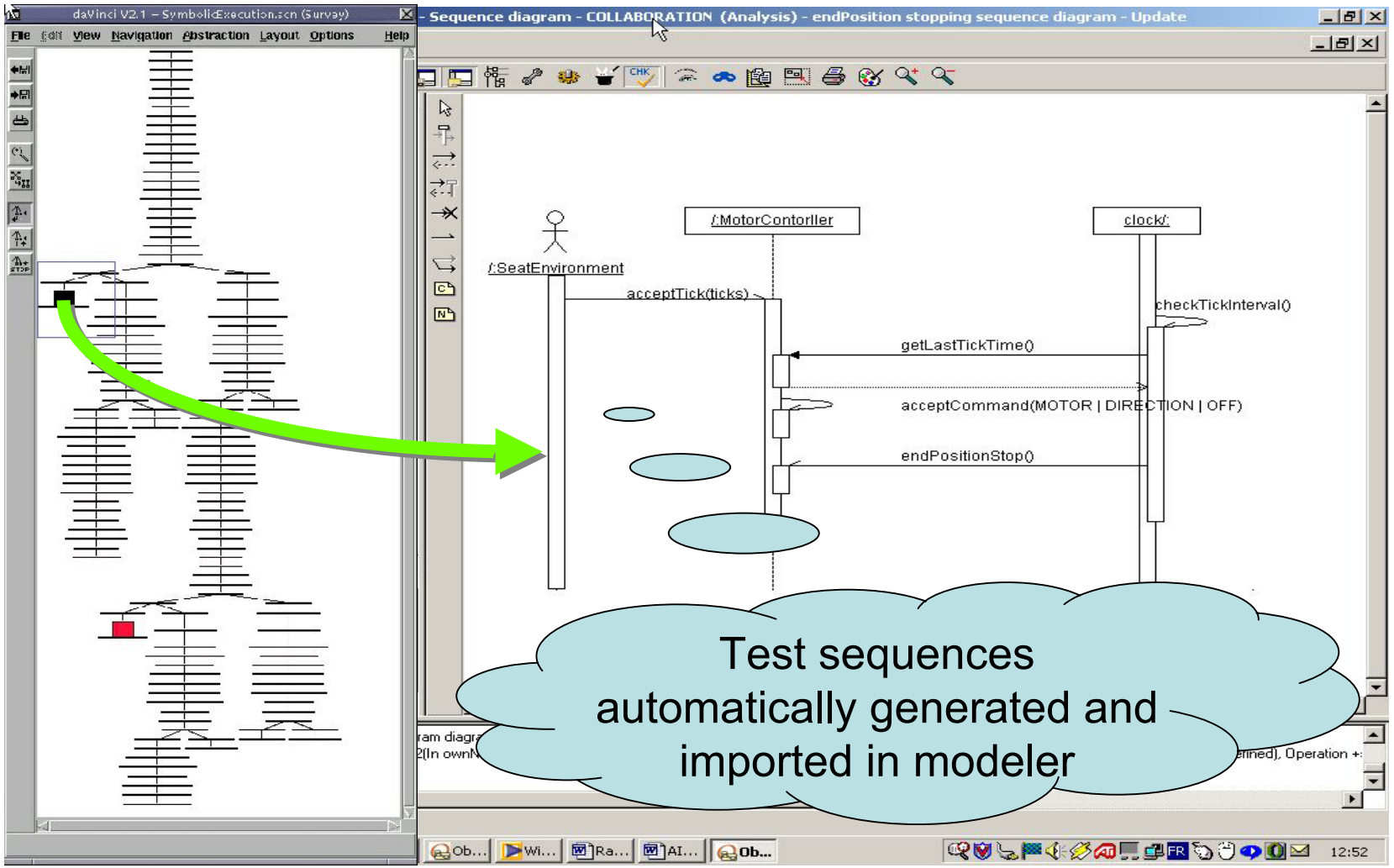
# Analyse du modèle à partir de la spécification

## Détection d'incohérences, dead-locks, famines,…



# Génération des séquences de tests

## Model structuration, test representation



# 2/ Conception orientée sûreté

## Conception de Systèmes Temps Réel Multitâches déterministes pour des applications de sûreté

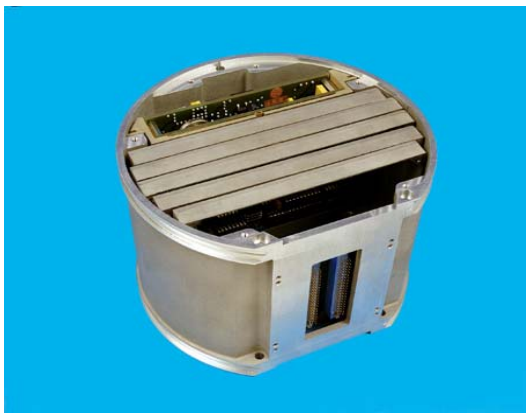
- **Support d'exécution tolérant aux fautes**
- **Respect de toutes les contraintes temps réel (CEI-880)**
- **Preuve du dimensionnement matériel (CPU, mémoire...)**
- **Réduction des coûts de conception et de validation**
- **Meilleure pérennité et maintenabilité.**

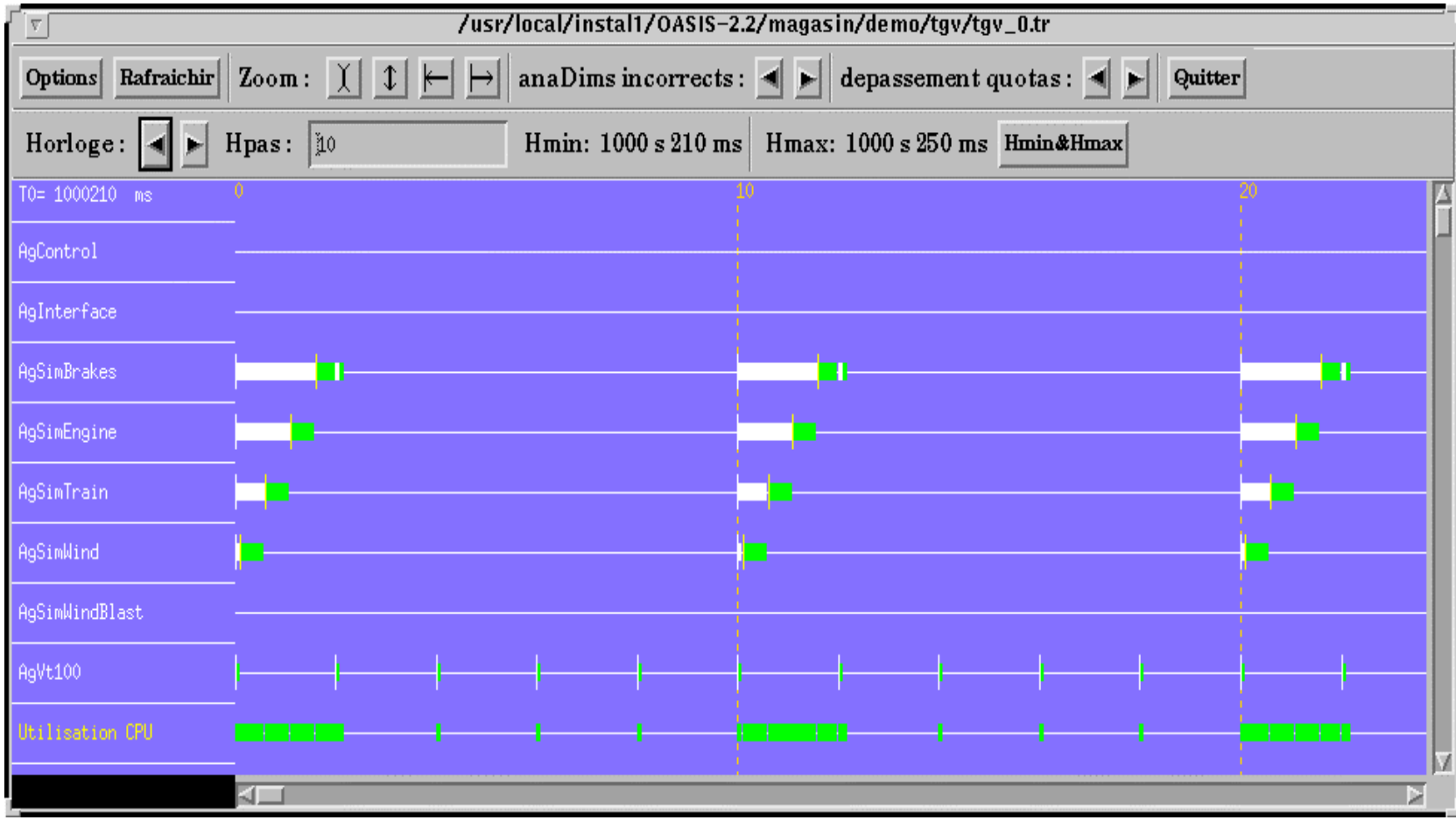
**FRAMATOME , EDF**

**SMIE,**

**EAST-AEE**

***Projet OASIS***






**Projet OASIS**

Conception de systèmes temps réel
Multitâches Déterministes pour les applications de sûreté

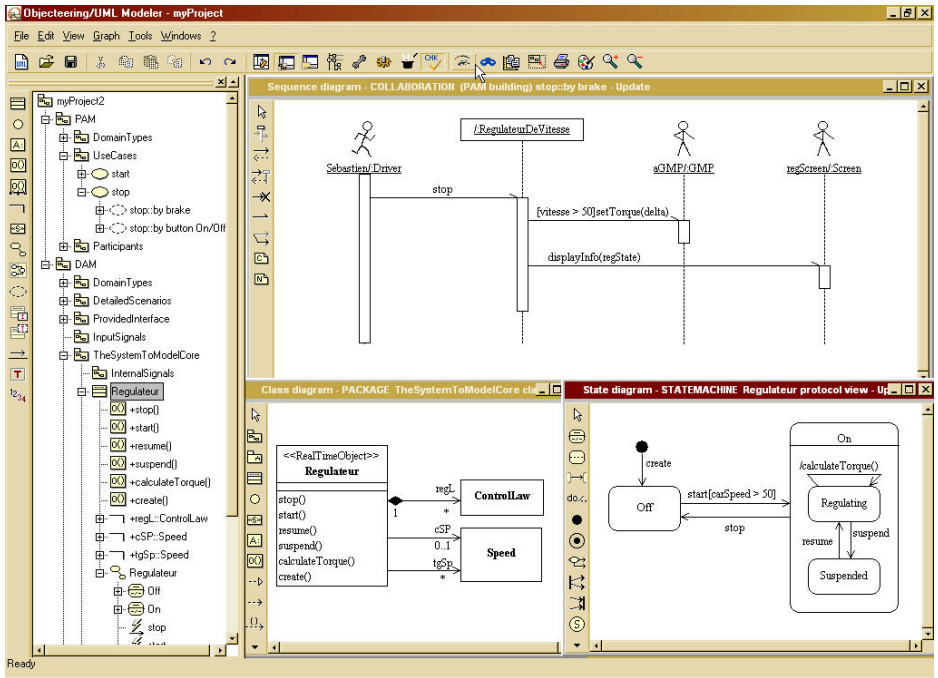
# 3/ Modélisation de systèmes de composants logiciels :

*Génération automatique d'implémentations de code temps réel*

## Méthodologie de conception d'applications temps réel UML à l'aide de composants génériques

A partir de spécifications déclaratives UML des contraintes de parallélisme, de temps réel et de distribution,

l'outil ACCORD génère automatiquement le code temps réel de l'application, en toute transparence des choix d'implantation

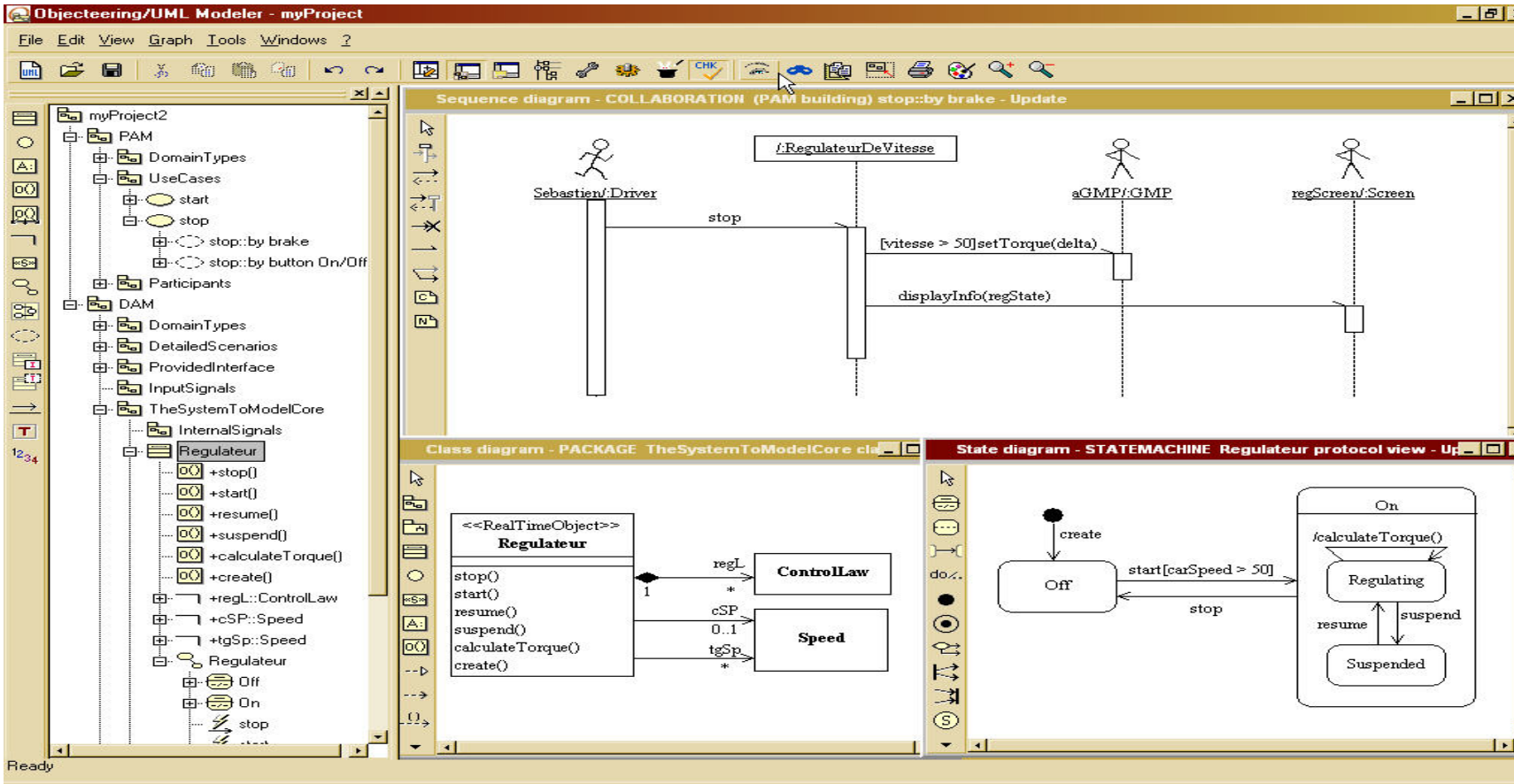



**PSA (X by wire, contrôle moteur,…), MECEL,..**

**Softeam**

**THALES**

***Projet ACCORD***




# Projet ACCORD

Conception et mise en oeuvre modulaire de composants temps réel réutilisables

# 4/ Analyse statique de code :

*Preuve interactive de propriétés du logiciel*

**Mise en œuvre à l'échelle industrielle de techniques avancées de compréhension des programmes**

Identification de menaces (division par zéro, blocage,…)

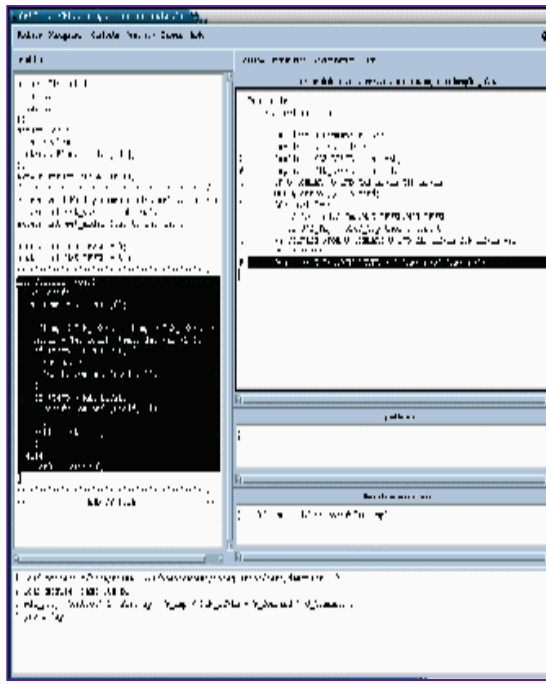
Preuve formelle

Précision de calculs utilisant des nombres flottants

Génération de cas de test (pilotes et oracles)

Au niveau du :

- du source (C ou SystemC)
- du binaire exécutable (PowerPC755)
- de spécifications (Lustre)
- de modèles (Promela/SPIN, géométriques) extraits du source code




**EDF, FRAMATOME**

**AIRBUS : logiciels critiques de ses futurs avions (A380, A400M)**

**DASSAULT**

**TNI Valiosys**

***Projet CAVEAT***

# Analyse statique de code :

## *Preuve interactive de propriétés du logiciel*

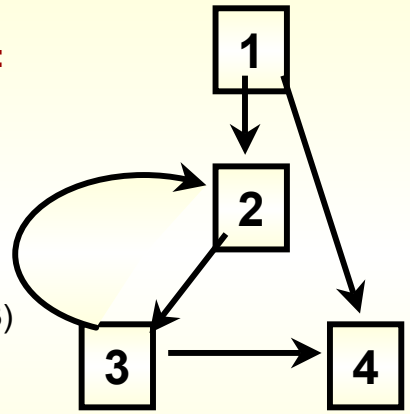
### Techniques utilisées :

- Preuve à la Hoare
- Interprétation abstraite
- Programmation logique contrainte
- Interprétation géométrique et topologie algébrique

**Analyse du programme :**

```
1 : while (x>=1000) {
2 :   x= x-a;
3 :   }
4 :
```

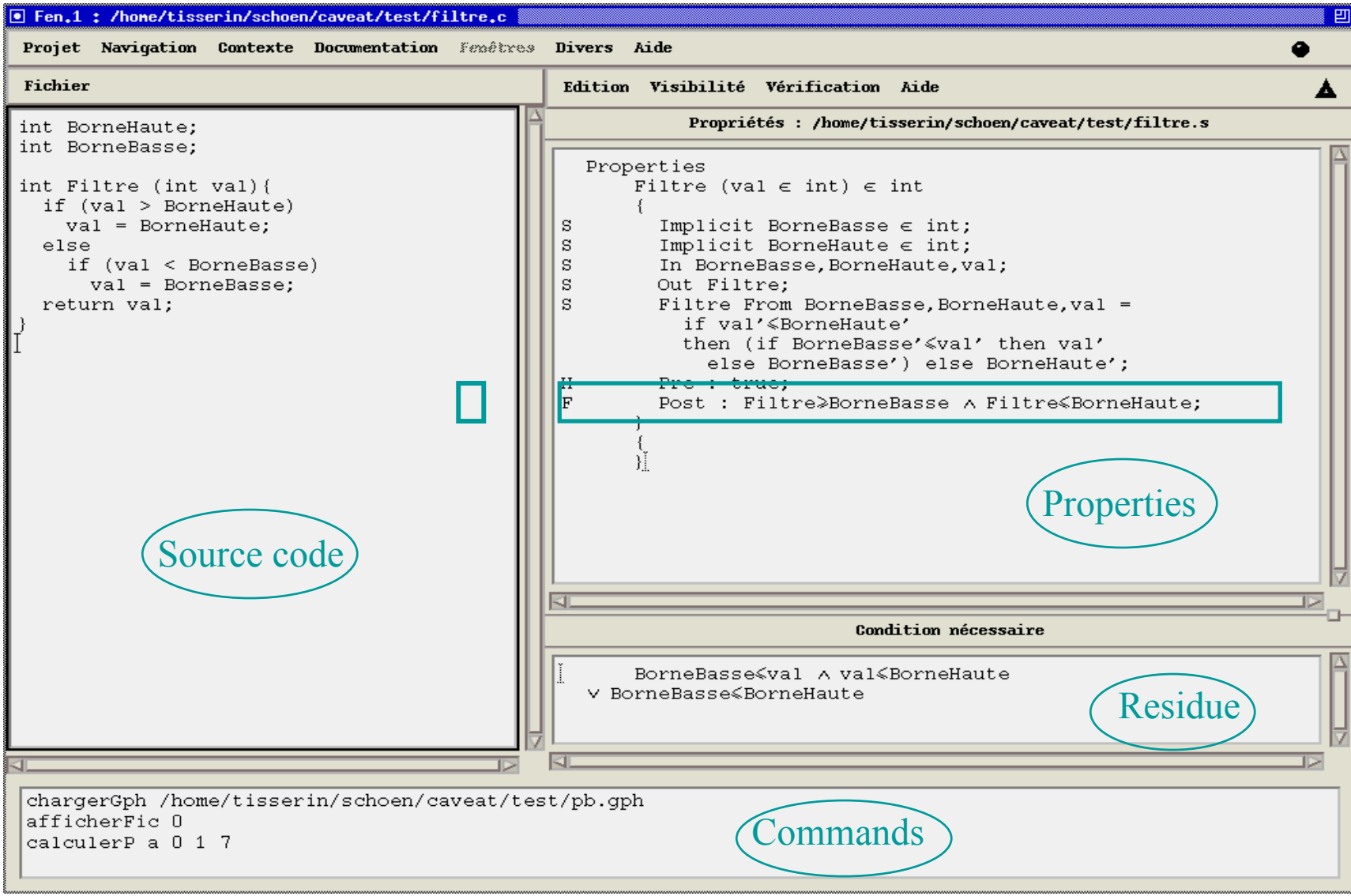
P1 : init(x)
P2 : (1000 ≤ x) ∧ (P1 ∨ P3)
P3 : P2[x+a]
P4 : (x<1000) ∧ (P1 ∨ P3)



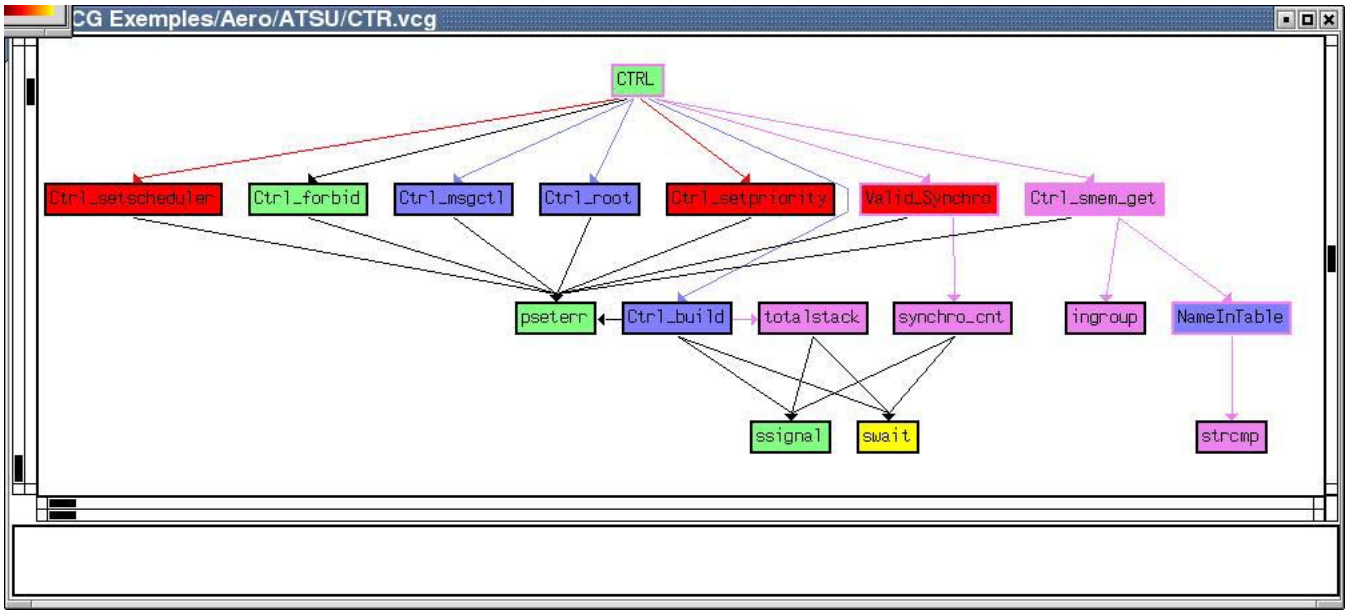

Recherche de cas de non-terminaison
(exemple de menace) ➔ **x ≥ 1000 ∧ a ≤ 0**

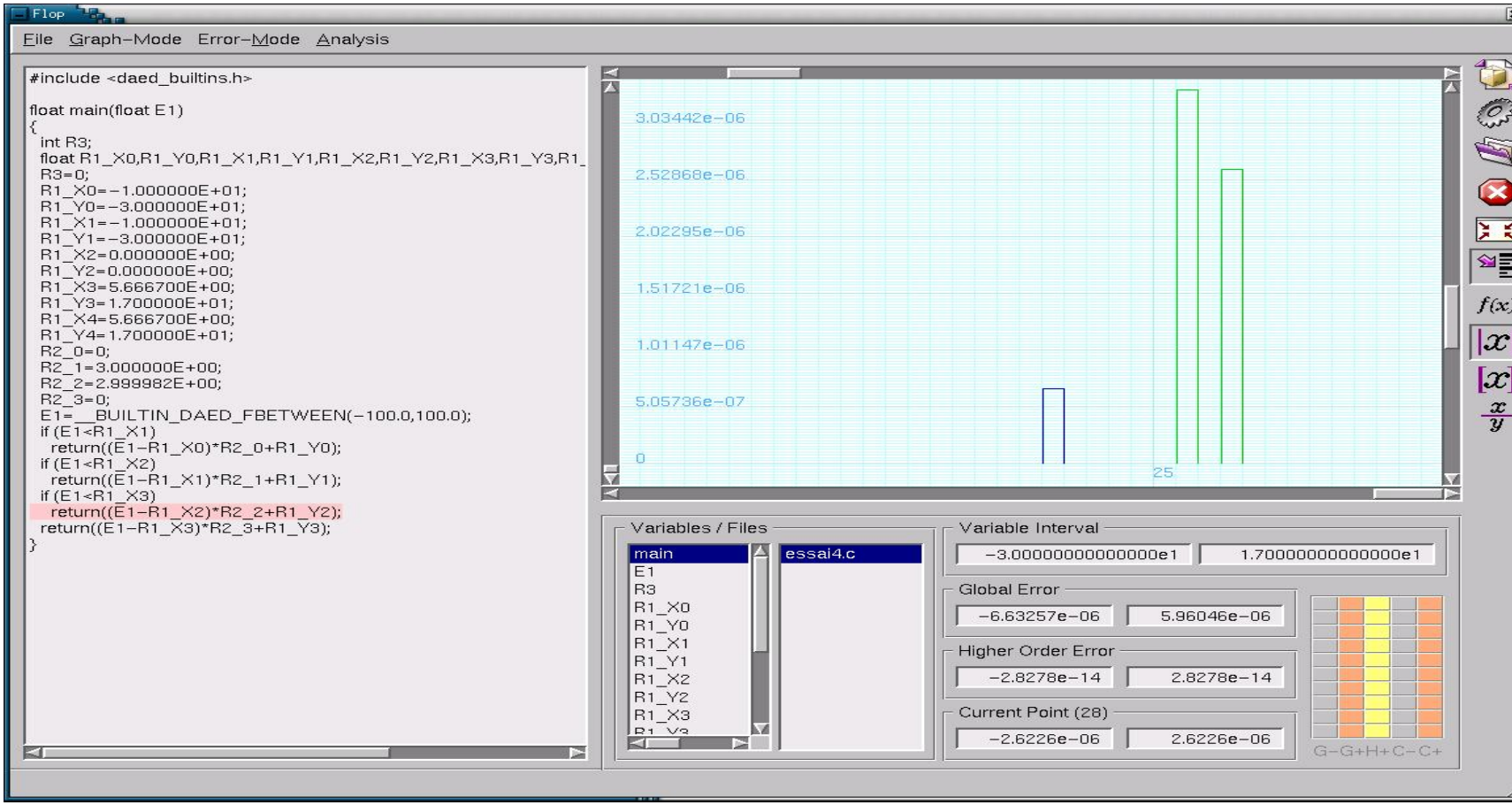
***Ex :***
***Interprétation abstraite***

***Projet CAVEAT***


Fen.1 : /home/tisserin/schoen/caveat/test/filtre.c
Projet Navigation Contexte Documentation Fenêtres Divers Aide
Fichier
Edition Visibilité Vérification Aide
Propriétés : /home/tisserin/schoen/caveat/test/filtre.s
int BorneHaute;
int BorneBasse;
int Filtre (int val){
if (val > BorneHaute)
val = BorneHaute;
else
if (val < BorneBasse)
val = BorneBasse;
return val;
}
Properties
Filtre (val ∈ int) ∈ int
{
S Implicit BorneBasse ∈ int;
S Implicit BorneHaute ∈ int;
S In BorneBasse,BorneHaute,val;
S Out Filtre;
S Filtre From BorneBasse,BorneHaute,val =
if val'⩽BorneHaute'
then (if BorneBasse'⩽val' then val'
else BorneBasse') else BorneHaute';
H Pre : true;
F Post : Filtre⩾BorneBasse ∧ Filtre⩽BorneHaute;
}
{
}
Source code
Properties
Condition nécessaire
BorneBasse⩽val ∧ val⩽BorneHaute
∨ BorneBasse⩽BorneHaute
Residue
chargerGph /home/tisserin/schoen/caveat/test/pb.gph
afficherFic 0
calculerP a 0 1 7
Commands

# Caveat : graphe d'appels avec menaces






# *Projet Fluctuat*

Etude de la précision numérique des nombres flottants

# 5/ Analyse dynamique du système

## Environnement de test de systèmes distribués temps réel utilisant des simulateurs de microprocesseurs

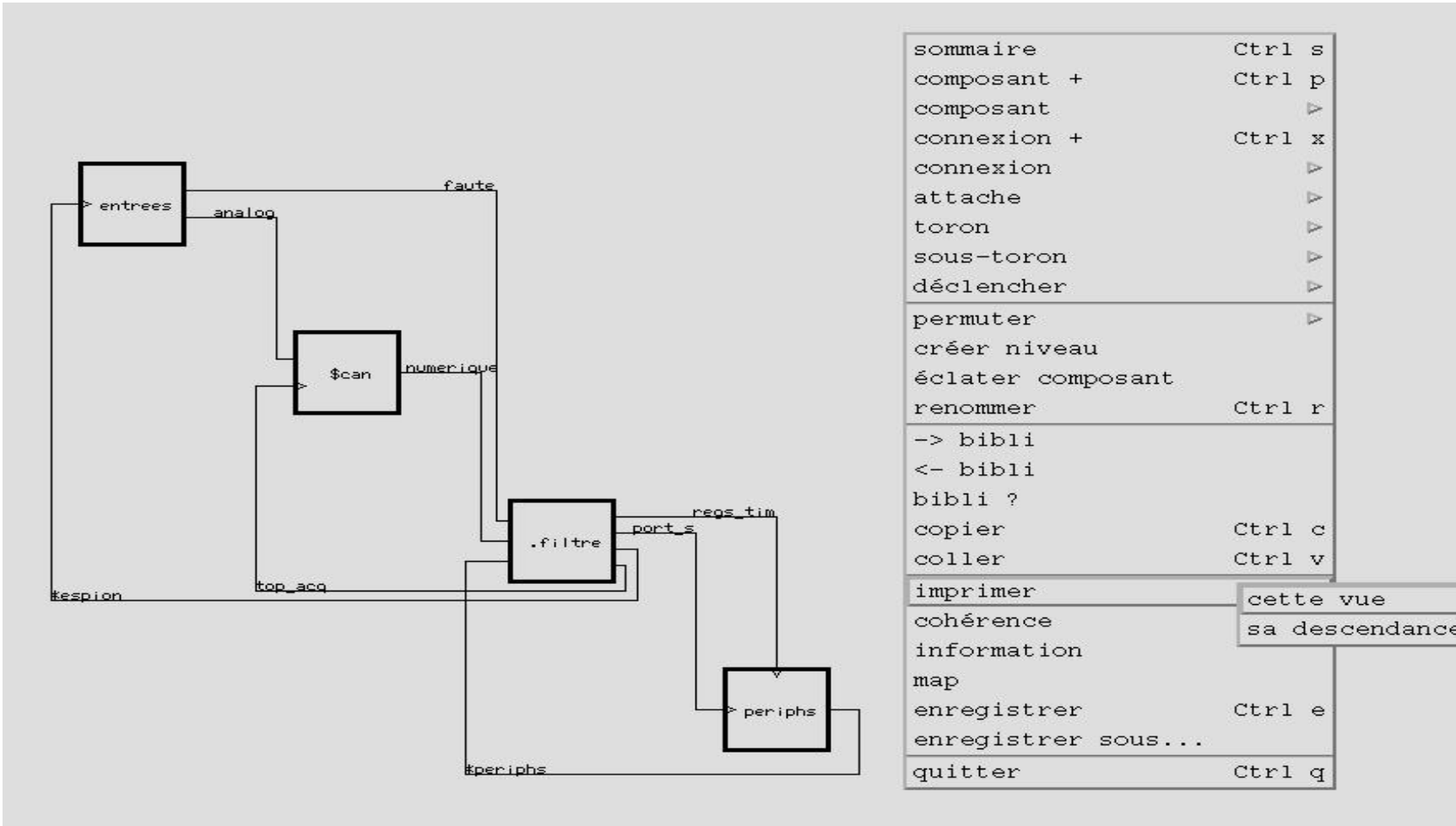
- **Test fonctionnel de systèmes par modélisation de l'environnement matériel et logiciel du code,**
- **Exécution du code final brut (non instrumenté) sans perturbation du temps réel (prise en compte des caches, pipe-lines,…)**
- **Gestion de formalismes multiples (binaire, C, SystemC,…)**
- **Performance : 1MIPS simulé sur PC Linux**

***Projet CLAIRE***

**IRSN**

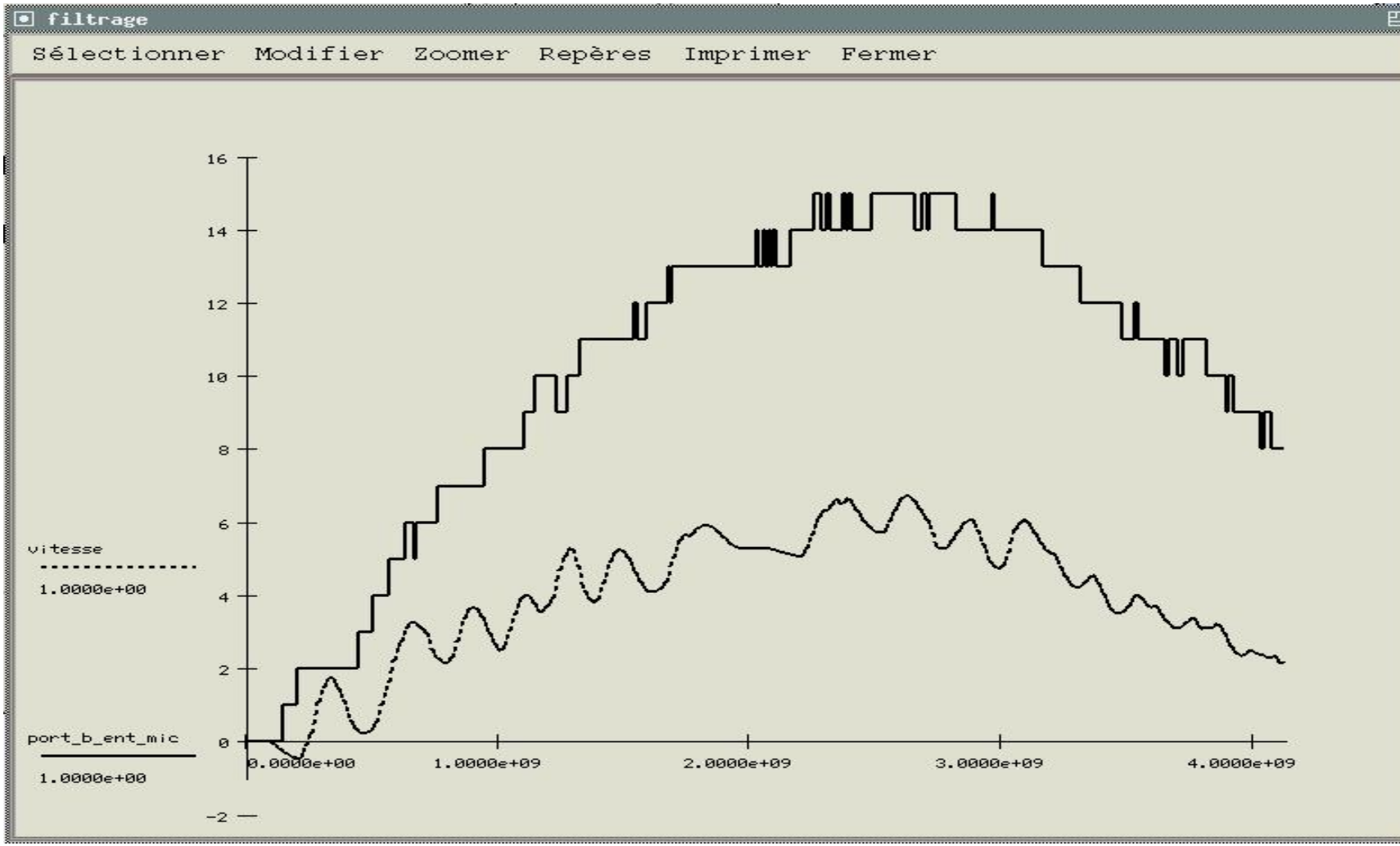
**AIRBUS**

**Duons Systèmes, TNI Valiosys**




**Projet CLAIRE**

# Analyse Dynamique




## *Projet CLAIRE*

# Analyse Dynamique