Model-based Development for Embedded Control Systems

- Which embedded control systems?
- Aérospatiale pioneering role
- State of the art
- Table of Contents
Which Embedded Control Systems?

- Safety critical systems
- Mission critical systems, time to market
Two Questions

Knowing the low reliability of computing technology

- thousands of car “recalled” for computing bugs
- Ariane V accident
- your personal computer …

1. Is it wise to use this poor technology in safety critical systems?

2. Why, nevertheless, things are not as bad as could be expected?
The safety-critical control industry has designed a very strong model-based development method

A short story of this method:

- Aérospatiale pioneering role
- How things evolved since then
- State of the Art and perspectives

Are academic people really aware of this story?
Aérospatiale pioneering steps in the early eighties — control models (block-diagrams)
Aérospatiale pioneering steps in the early eighties __

control models (block-diagrams)

\[
= \text{formal software specification}
\]
Aérospatiale pioneering steps in the early eighties

control models (block-diagrams)

formal software specification

automatic code generation

Software
Aérospatiale pioneering steps in the early eighties __

control models (block-diagrams)

\[ \text{formal software specification} = \text{Software} \]

↓

automatically code generation

↓

“Spécification Assistée par Ordinateur” (SAO)

“Computer Aided Specification”
Interest of SAO

Twofold:

- **Automatic code generation from high-level control models:**
  easier and earlier debugging

- **Graphic language close to the cultural background of avionic engineers, test pilots, suppliers, certification authorities, . . . :**
  allows easier communication within the enterprise
  preserves the know-how and makes easier the technology transfer

SAO participates to the success of A320
From then on...

Powerful model-based development tools:

- SAO replaced by SCADE
  - commercial product partially based on synchronous technology
  - Do178B level A qualified automatic code generator

- Simulink/Stateflow
  - continuous/discrete time simulation toolbox
    - the defacto standard in control modelling

- Formal methods: automatic mathematical proofs for dynamic systems
From then on…

More powerful execution platforms:

- multi-tasking

- distributed and multi-processor
State of the Art

modelling
State of the Art

modelling

simulation debugging

2009-2010 SLE, ENSIMAG
State of the Art

Automatic import
Simulation
Debugging
Modelling

Automatic import
Modelling
Simulation and debugging

2009-2010 SLE, ENSIMAG
State of the Art

- Modelling
- Simulation
- Debugging
- Automatic import
- Formal verification

2009-2010 SLE, ENSIMAG
State of the Art

- modelling
- automatic import
- formal verification
- architecture choice
- automatic code generation
- simulation
- debugging
State of the Art

- automatic code generation
- architecture choice
- formal verification
- automatic import
- simulation
- debugging
- modelling
- tests

2009-2010 SLE, ENSIMAG
Perspectives

Faithfulness tests
automatic code generation
architecture choice
formal verification
automatic import
simulation
debugging
modelling
more modelling frameworks
more formal tools
more architectures
more test methods

2009-2010 SLE, ENSIMAG
Perspectives

- more modelling frameworks:
  networks, telecommunications, ...

- more powerful formal methods

- more execution platforms
  CAN, Ethernet, Internet, ...

- more test methods
A Key Issue: Faithfulness

What you \{ model, simulate, prove \} \textbf{is what you} \{ implement, execute \}
Implantation sûre de systèmes contrôle/commande
Implantation sûre de systèmes contrôle/commande

- ”sûre” ?
  Étudier et expérimenter les méthodes qui permettent de garantir que l’implémentation respecte de bonnes propriétés :
  - temps-réel : notion relative, beaucoup de paramètres (matériel/logiciel)
  - déterminisme : essentiellement liée au logiciel, à l’exécutif (OS).

- N.B. nécessaire/requis pour les systèmes critiques (on parle de ”hard real-time”).
Implantation sûre de systèmes contrôle/commande
(suite)

But du cours

- Voir les méthodes classiques de conception/implantation sûres :
  - synchrone "pur" : systèmes échantillonnés, mono-tâche, sans exécutif (i.e. sans OS)
  - relachement du synchronisme, multi-tâches déterministe

- Expérimenter sur la brique Lego :
  - pas vraiment "critique", mais ...
  - suffisemment simple et représentatif pour illustrer les principes.