Informatique des systèmes automatique critiques

Paul Caspi

Laboratoire **Verimag** *(CNRS-UJF-INPG)*

Une histoire technico-scientifique

pleine de bruit et de fureur

Informatique des systèmes automatique critiques

Paul Caspi

Laboratoire **Verimag** *(CNRS-UJF-INPG)*

- ⇒ **Les systèmes automatiques critiques et leurs exigences**
- ⇒ **Approche fiabiliste**
- ⇒ **Approche pragmatique**
- ⇒ **Approche logico-mathématique**
- ⇒ **Approche par modèles**
- ⇒ **Conclusion**

Exemples de systèmes automatiques critiques

commandes de vol

arrêt d'urgence

contrôle de vitesse

automatisme intégral

Exigences

Ne pas dégrader par rapport à l'existant

& **Ne pas produire de catastrophe visible**

& **Ne pas réduire l'espérance de vie de l'usager**

$$\Rightarrow$$

moins d'une défaillance dangereuse pour

$$\begin{cases} un\ milliard\ d'heures \\ cent\ mille\ ans \end{cases}$$

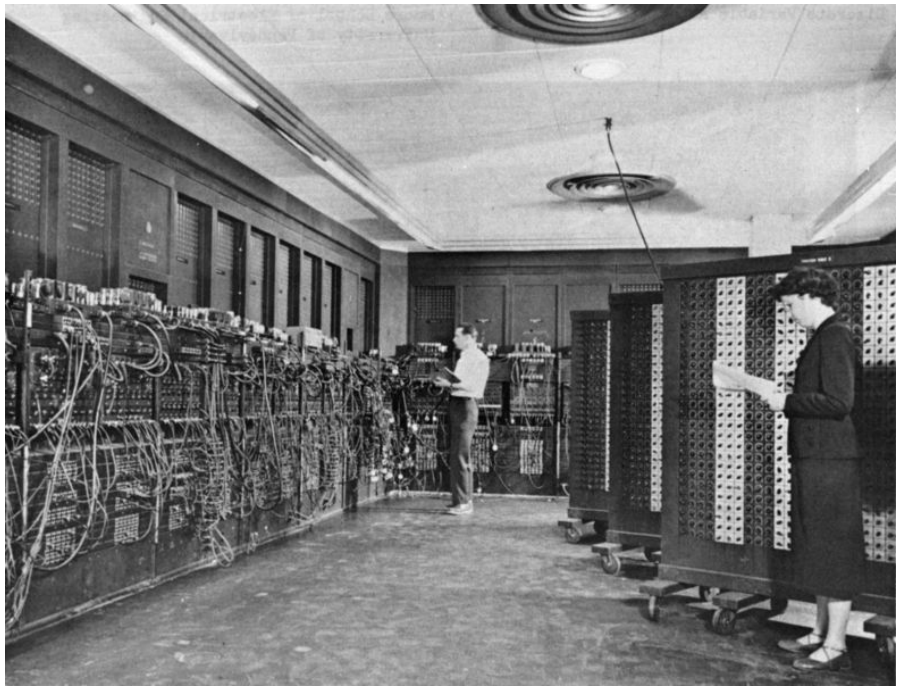
de fonctionnement !

Question : cela a-t-il un sens ?

Approche fiabiliste

- **ordinateurs et fiabilité**
- **le cas du logiciel**

Ordinateurs et fiabilité : une vieille histoire



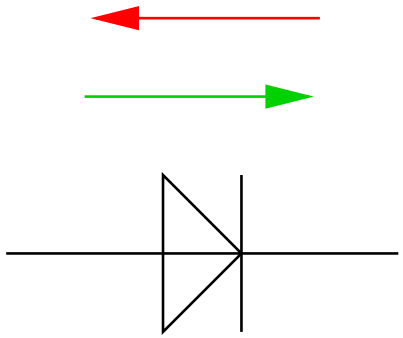
Eniac : très lent, très peu fiable : 17 468 lampes

la probabilité qu'une lampe casse pendant un calcul n'est pas négligeable

von Neumann **: Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components**

Solution : redondance

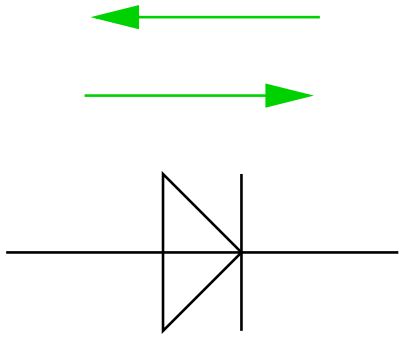
Exemple : une diode



Quels sont les types de pannes ?

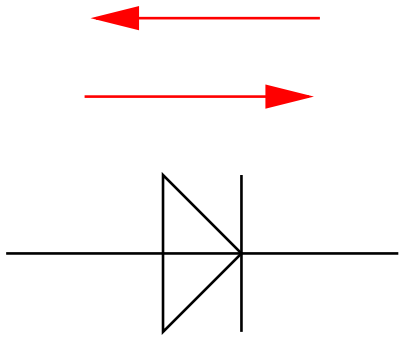
Solution : redondance

Panne de type court-circuit



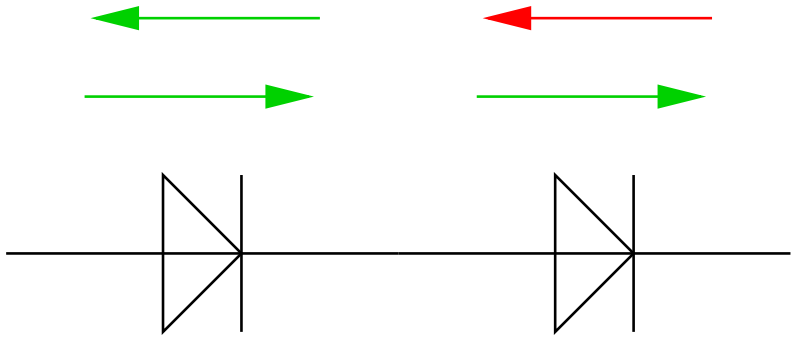
Solution : redondance

Panne de type coupure



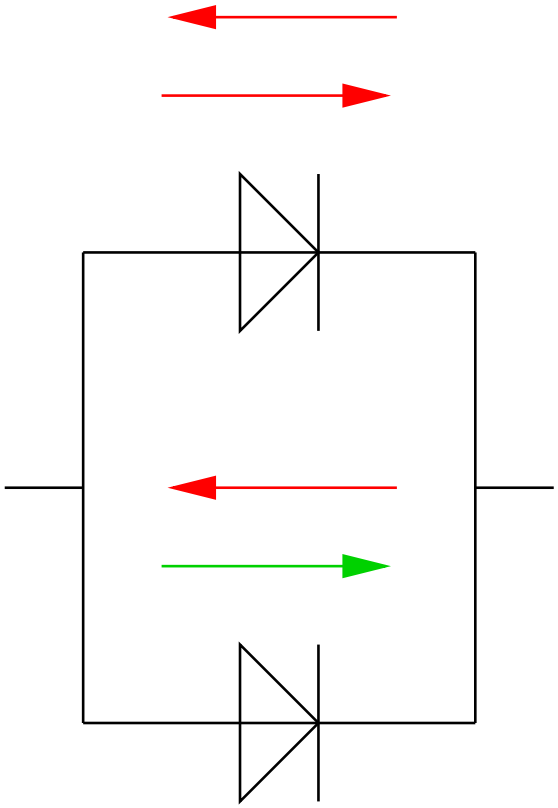
Solution : redondance

pour tolérer les court-circuits



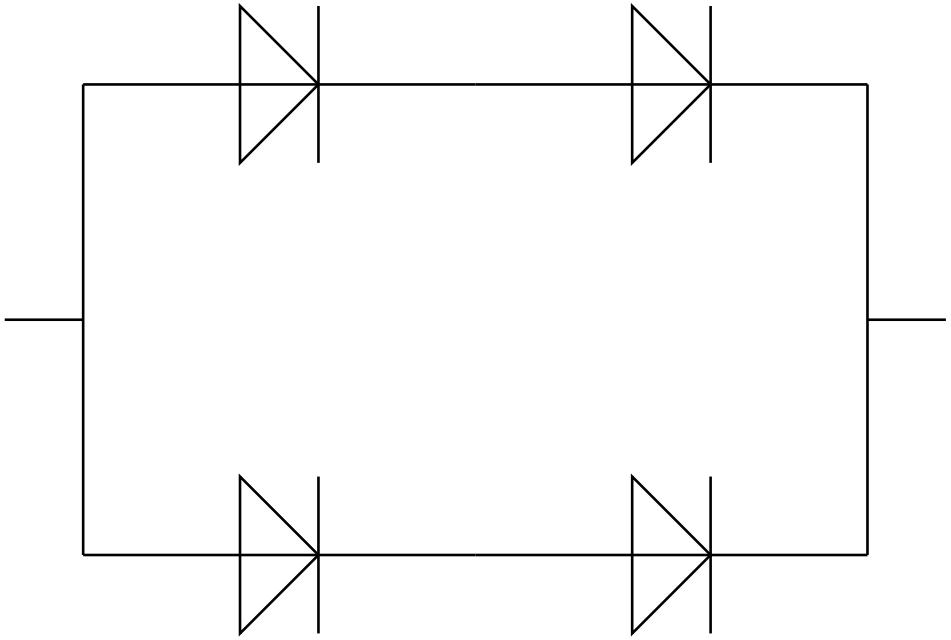
Solution : redondance

pour tolérer les coupures



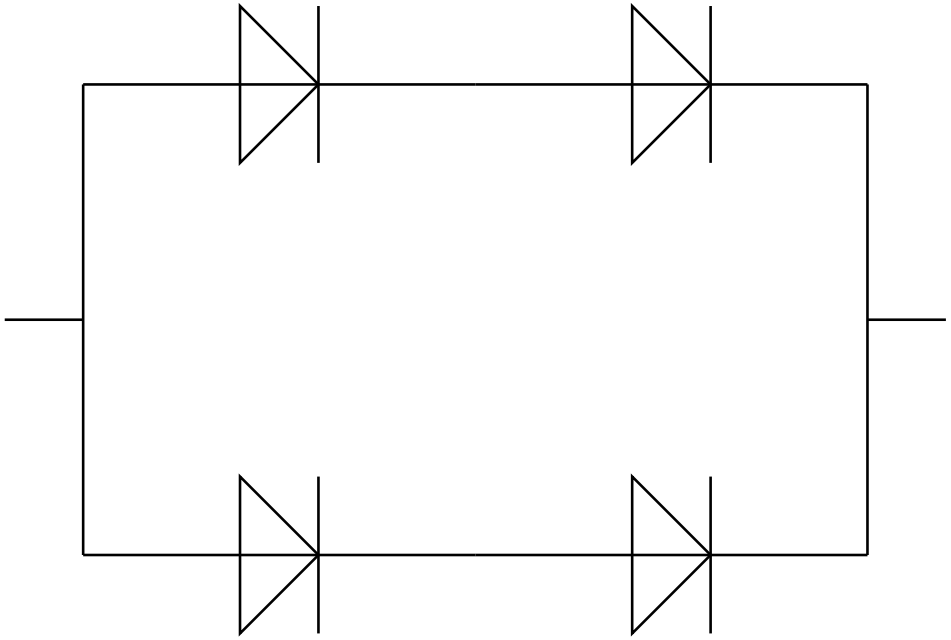
Solution : redondance

pour tolérer les deux



Solution : redondance

pour tolérer les deux

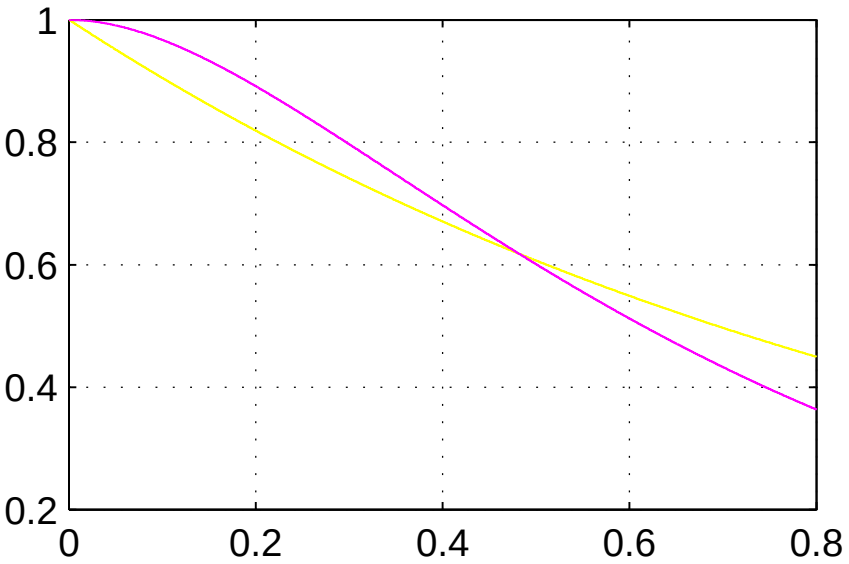


Est-ce que ça marche ? Pourquoi ?

Est-ce que ça marche ?

Pas toujours : plus de diodes $\Rightarrow$ plus de pannes ! !

Calculs de fiabilité :



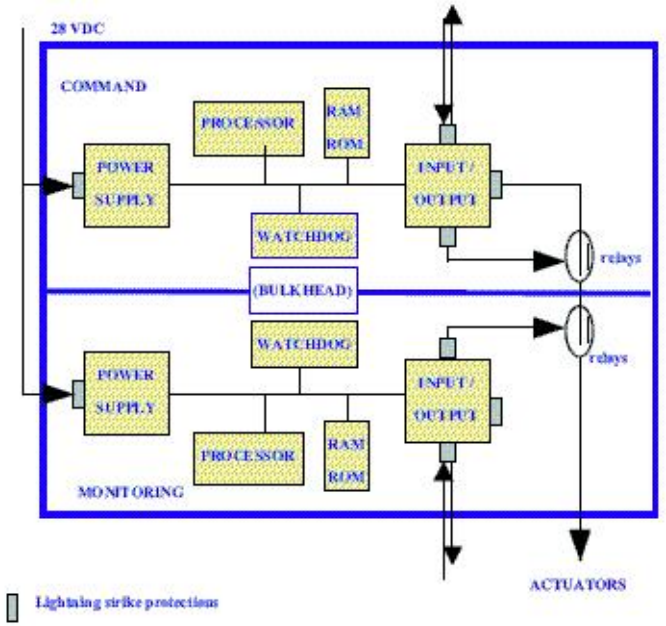
Time offset: 0

— $R_1 = e^{-\lambda t}$

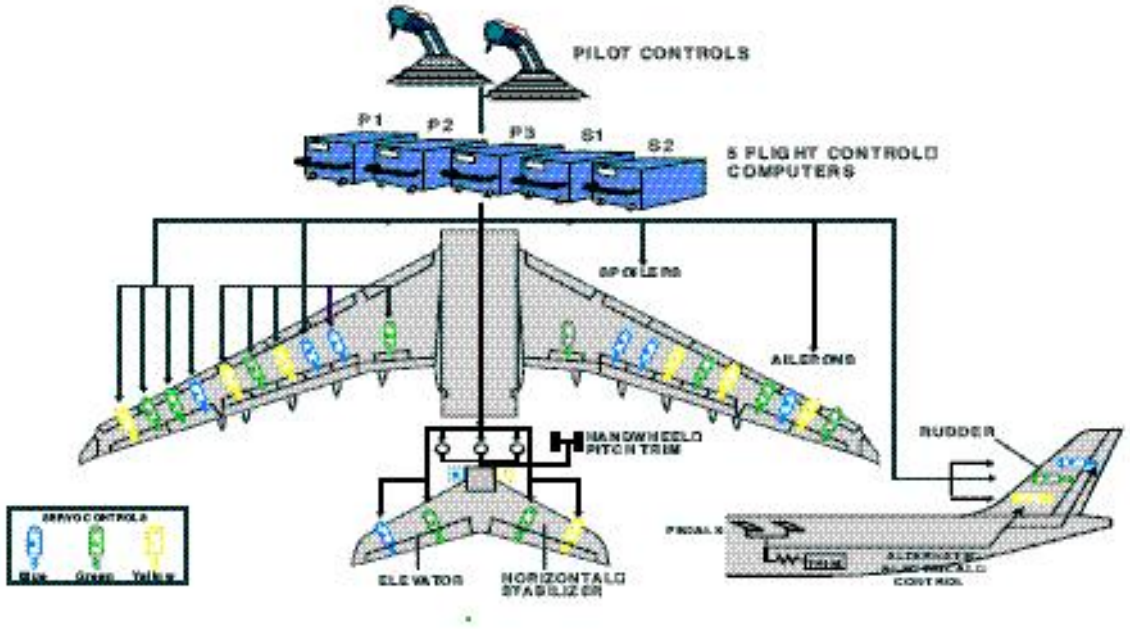
— $R_4 = 2R_1^2 - R_1^4$

Exemple : commandes de vol d'Airbus

redondance de détection



redondance de remplacement



On peut calculer des probabilités ? !

Et le logiciel. . . ?

- **Pourquoi le logiciel ?**
- **Le logiciel ne s'use ni ne grille**

Et pourtant. . .

- **Une mission Mariner vers Mars s'est perdue parce qu'il manquait une virgule dans un programme (en Fortran)**
- **Dès les années 70, IBM a lançé des études de fiabilité du logiciel**

De quoi s'agit-il ?

Et le logiciel. . . ?

- **Pourquoi le logiciel ?**
- **Le logiciel ne s'use ni ne grille**

Et pourtant. . .

- **Une mission Mariner vers Mars s'est perdue parce qu'il manquait une virgule dans un programme (en Fortran)**
- **Dès les années 70, IBM a lançé des études de fiabilité du logiciel**

De quoi s'agit-il ?

- **Il s'agit d'erreurs de conception qui se révèlent à l'usage**

Et le logiciel. . . ?

Les praticiens :

- **Mais nous faisons de la conception depuis des siècles sans que l'on ne nous dérange avec cette question**

Qu'est-ce qui a changé ?

Et le logiciel. . . ?

Les praticiens :

- **Mais nous faisons de la conception depuis des siècles sans que l'on ne nous dérange avec cette question**

Qu'est-ce qui a changé ?

- **le logiciel permet de faire des conceptions infiniment plus complexes que précédemment**
- **faire des logiciels corrects est un des plus grands défis que l'humanité ait jamais affronté**

  songez aux millions de lignes de programmes que vous activez lorsque vous cliquez sur Internet !

Comment fait-on ?

Et le logiciel. . . ?

Idée : faire comme pour le matériel, appliquer le programme de von Neumann

- **mesurer, évaluer**
- **accroître par redondance**

Et le logiciel. . . ?

Idée : faire comme pour le matériel, appliquer le programme de von Neumann

- **mesurer, évaluer**
- **accroître par redondance**

Obstacles

- **la fiabilité du logiciel se mesure-t-elle ? oui**
- **se mesure-t-elle précisément ? non**

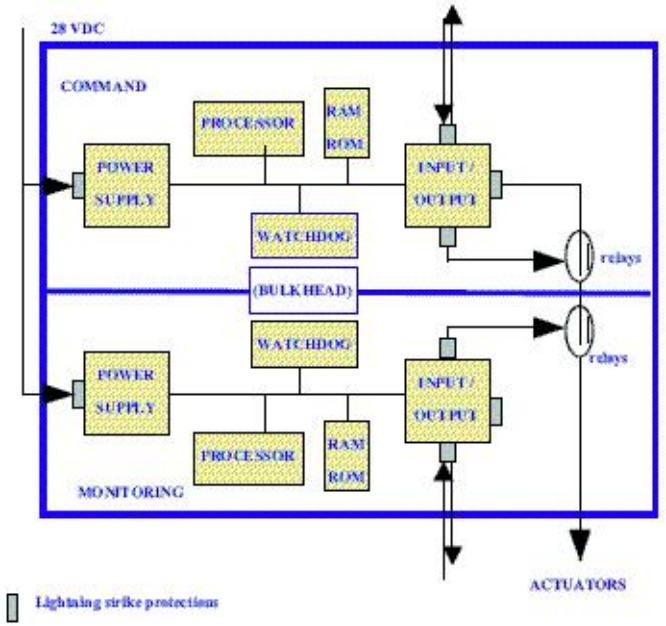
  d'autant plus que le système a été conçu de grande qualité. songez au temps qu'il faut pour évaluer un système à moins d' une défaillance par milliard d'heure de fonctionnement
- **s'accroît-elle facilement ? non**

  problèmes de redondance indépendantes

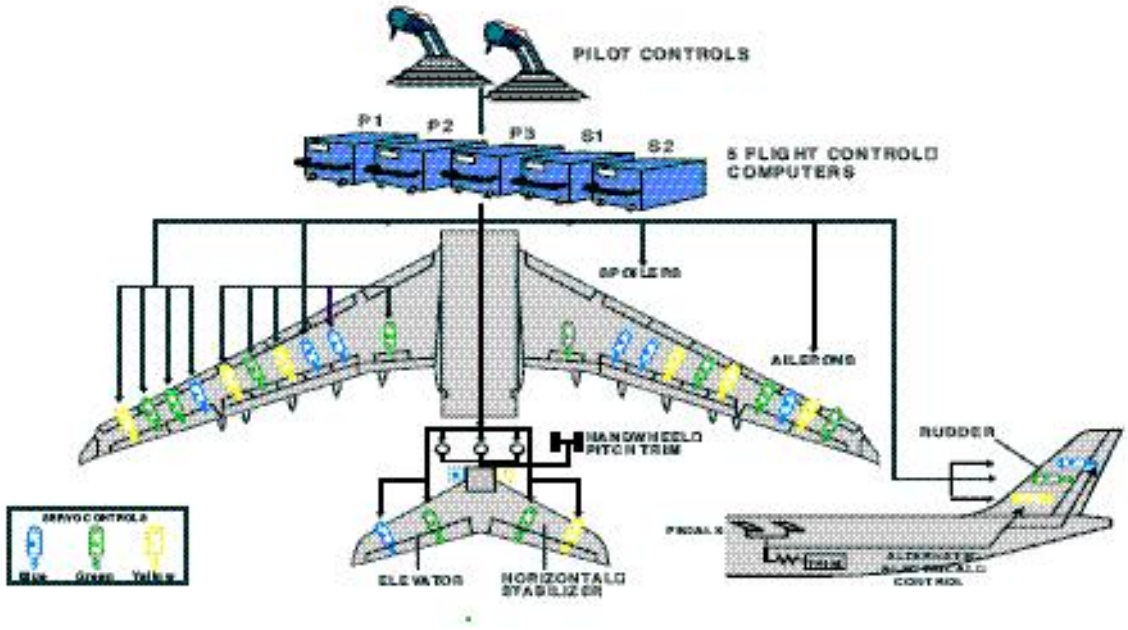
Cependant utilisées comme complément...

Commandes de vol d'Airbus

redondance de détection



redondance de remplacement



Les fiabilistes ont été néanmoins surpris...

lorsqu'apparûrent les premiers systèmes critiques programmés

Exemples :

- **arrêt d'urgence des centrales nucléaires**
- **Airbus A320**

Comment avaient été conçus ces systèmes ?

Méthodes traditionnelles de développement rigoureux

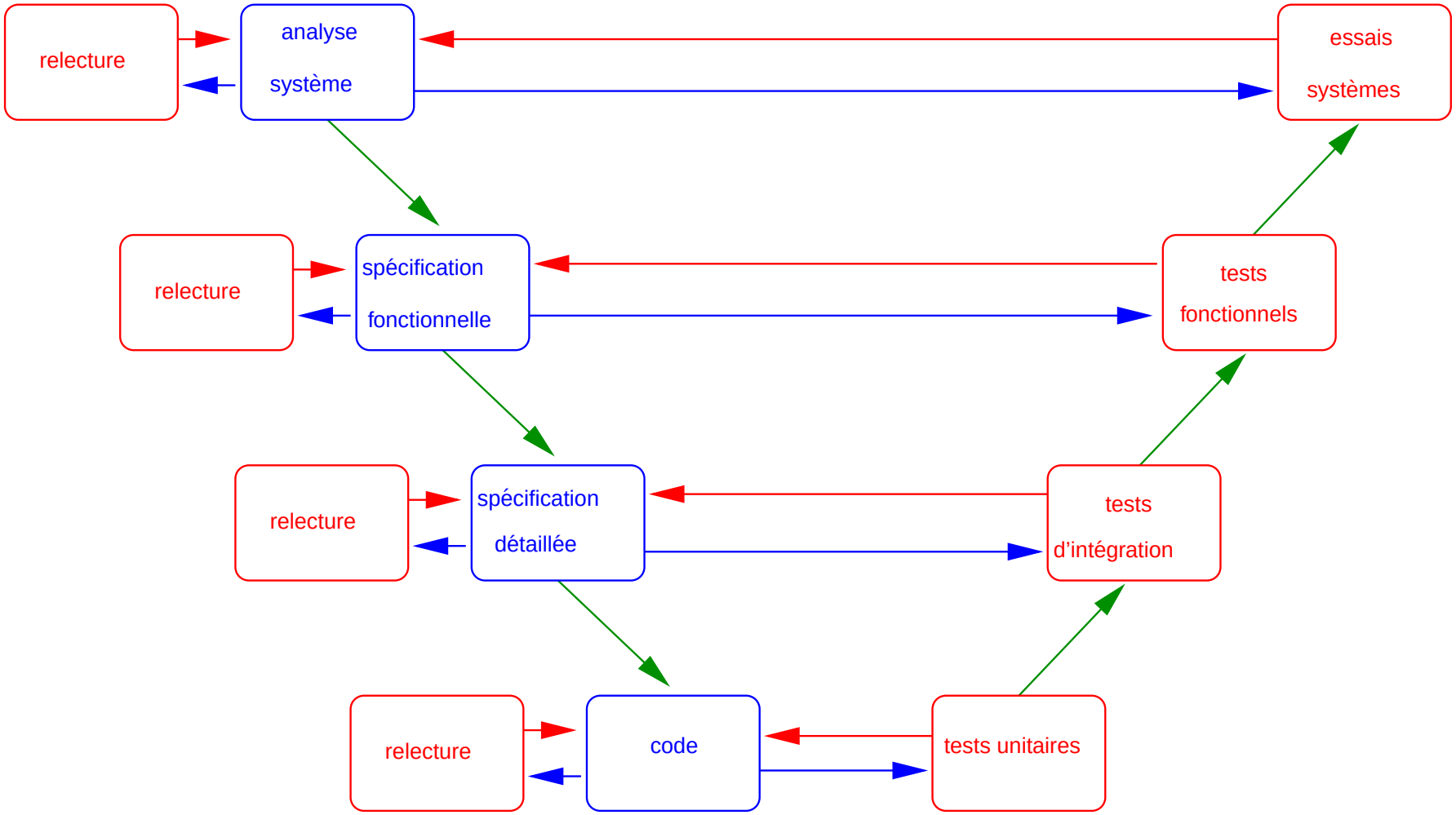


Diagramme en V

Méthodes traditionnelles de développement rigoureux

- Processus coûteux (plus de 3 fois le prix normal ?)
- Processus long

  peut ne pas converger

  peut générer des retards (SACEM, MAGGALY)
- Nécessite des acteurs éprouvés
- Est à la base de la plupart des systèmes critiques existants

Les universitaires pendant ce temps...

Est-ce ainsi que l'on construit des ponts ?

Par essais et correction d'erreurs ?

Ne sommes-nous pas au Moyen-Age ?

Sémantique axiomatique des programmes

Floyd, Hoare

Axiomes :

$$\overline{F[e/x]\{\texttt{x} := \texttt{e}\}F}$$

Règles d'inférence :

$$\frac{F \wedge C\{\texttt{P}\}F}{F\{\texttt{tant que C faire P}\}\neg C \wedge F}$$

Un programme est traduit en formules logiques sur lesquelles on peut raisonner et faire des preuve !

Mais...

Mais...

La logique est difficile, plus difficile que les calculs de ponts...

Mais...

La logique est difficile, plus difficile que les calculs de ponts...

Indécidabilité de la logique de premier ordre (Turing, Kleene)

Nécessité de l'intervention d'une intelligence humaine

Mais...

La logique est difficile, plus difficile que les calculs de ponts...

Indécidabilité de la logique de premier ordre (Turing, Kleene)

Nécessité de l'intervention d'une intelligence humaine

Qu'est-ce que l'intelligence humaine ?

Construction prouvée de programmes et de systèmes _

D'où l'idée de construction progressive prouvée à la B (J.R. Abrial)

Reste tout de même difficile :

- **longue (peut ne pas converger)**
- **coûteuse (nécessité de personnel qualifié)**
- **dans les applications RATP, il faut traduire en logique non seulement les programmes mais aussi les environnements (aiguilleurs, conducteurs, voies, aiguilles...)**

Méthode B Système en cours de développement

Peu de succès probants à part l'emblématique METEOR

Méthodes formelles partielles

D’autres universitaires ont cherché à contourner ces difficultés

- **Vérification par modèles**
- **Interprétation abstraite**
- **Développement par modèles**

Vérification par modèle

Idée (J. Sifakis - Verimag, E. Clarke) :

se restreindre à des parties décidables de la logique. (espaces d'états finis)

- **moindre besoin de personnel spécialisé**
- **génération de contre-exemples en cas d'échec**
- **nombreux cas d'application :**
  - **systèmes logiques (signalisation,....)**
  - **circuits**

La vérification par modèles est courante en conception de circuits et équipe les environnements de conception assistée

Mais risques d'explosion combinatoire

Interprétation abstraite

Idée (P. Cousot - Ecole normale supérieure) :

Approximer supérieurement de façon décidable la logique non décidable

C'est une semi-décision :

- **oui : preuve concluante**
- **non : on ne sait pas (approximation trop grossière ?)**

Problème : trouver l'approximation est aussi difficile que faire la preuve

D'où l'idée de se restreindre à des approximations prédéfinies pour des propriétés spécifiques :

- **erreurs à l'exécution : débordements (Ariane V)...**
- **propriétés temps réel**

Succès industriel

Développement par modèles

Dans certains domaines comme l'automatique, on peut produire automatiquement le code à partir des modèles métier

- **Petite histoire de la réalisation des automatismes**
- **Arrivée de l'informatique**
- **Réaction des praticiens**
- **Quelques leçons de cette histoire**

De la première mécanique …

De la première mécanique …

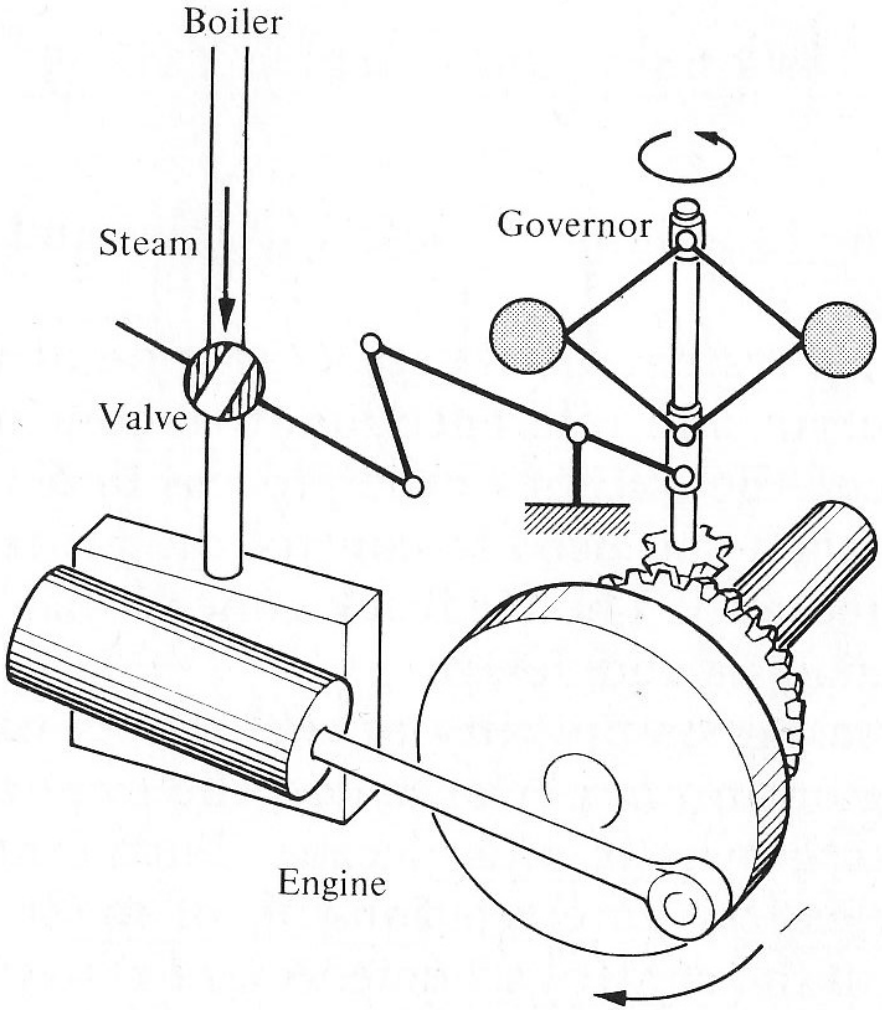


Figure 1.6. Watt flyball governor.

. . .à l'électronique . . .

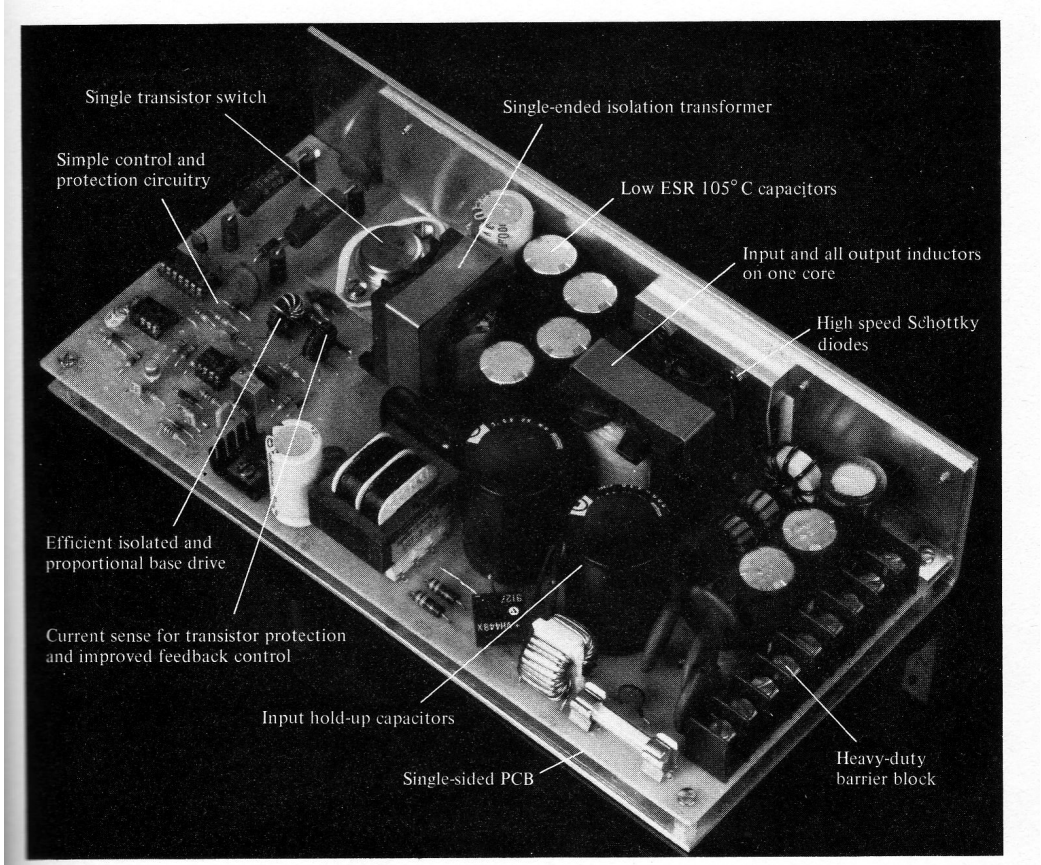


Figure E2.5. (a) Prototype circuit for switcher. (b) Filter circuit for 150 w switcher (idealized). (Photo courtesy of Ron Burian and TESLAco.)

…à la schématique …

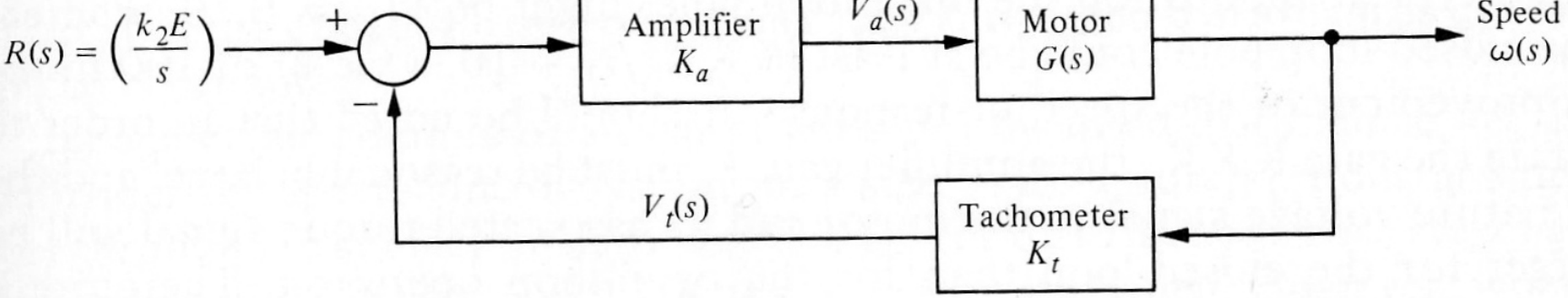


Figure 3.8. Closed-loop speed control system.

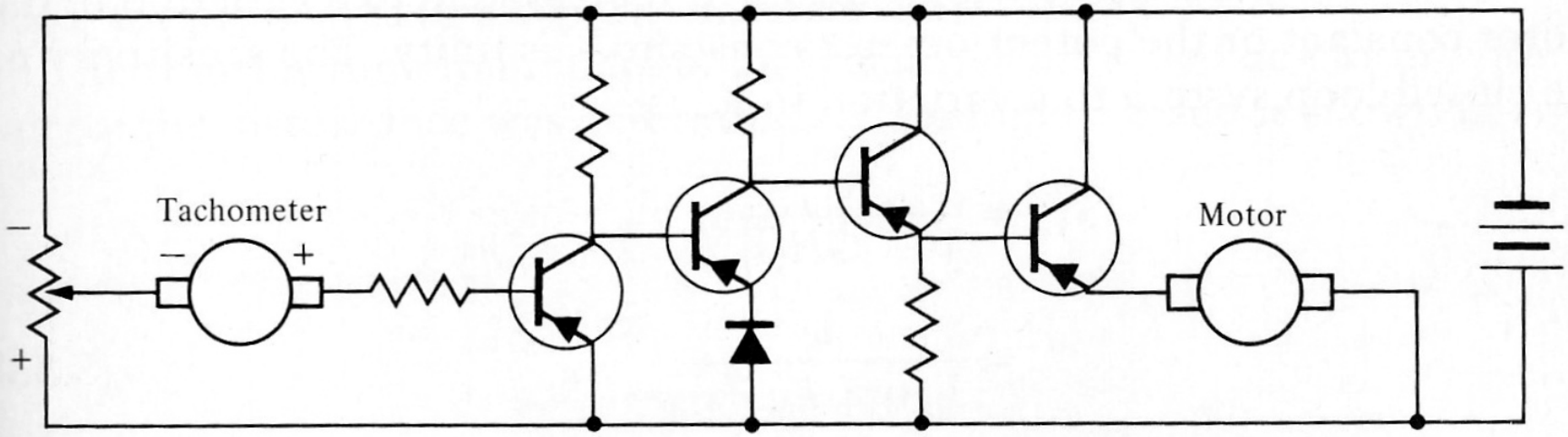
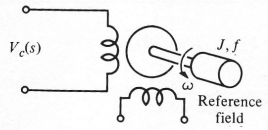


Figure 3.9. Transistorized speed control system.

…et aux mathématiques

Element or System | **G(s)**

7. ac-motor, two-phase control field

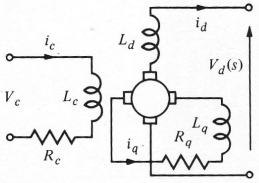


$$\frac{\theta(s)}{V_c(s)} = \frac{K_m}{s(\tau s + 1)}$$

$\tau = J/(f - m)$

m = slope of linearized torque-speed (normally negative)

8. Amplidyne



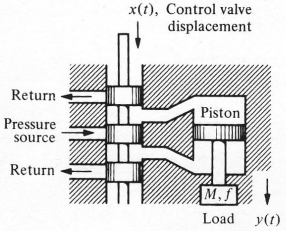
$$\frac{V_d(s)}{V_c(s)} = \frac{(K/R_cR_q)}{(s\tau_c + 1)(s\tau_q + 1)}$$

$\tau_c = L_c/R_c$, $\tau_q = L_q/R_q$

For the unloaded case, $i_d \simeq 0$, $\tau_c \simeq \tau_q$

$0.05 \text{ sec} < \tau_c < 0.5 \text{ sec}$

9. Hydraulic actuator



$$\frac{Y(s)}{X(s)} = \frac{K}{s(Ms + B)}$$

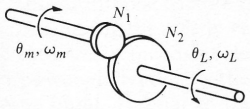
$K = \frac{Ak_x}{k_p}$, $B = \left(f + \frac{A^2}{k_p}\right)$

$k_x = \left.\frac{\partial g}{\partial x}\right|_{x_0}$, $k_p = \left.\frac{\partial g}{\partial P}\right|_{P_0}$,

$g = g(x, P)$ = flow

A = area of piston

10. Gear train

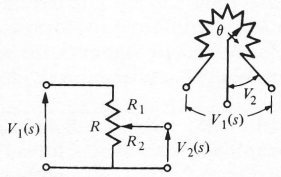


$$\text{Gear ratio} = n = \frac{N_1}{N_2}$$

$N_2\theta_L = N_1\theta_m$, $\theta_L = n\theta_m$

$\omega_L = n\omega_m$

11. Potentiometer

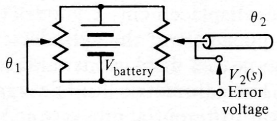


$$\frac{V_2(s)}{V_1(s)} = \frac{R_2}{R} = \frac{R_2}{R_1 + R_2}$$

$$\frac{R_2}{R} = \frac{\theta}{\theta_{max}}$$

Element or System | **G(s)**

12. Potentiometer error detector bridge

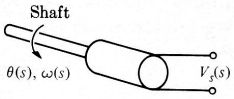


$V_2(s) = k_s(\theta_1(s) - \theta_2(s))$

$V_2(s) = k_s\theta_{error}(s)$

$k_s = \frac{V_{battery}}{\theta_{max}}$

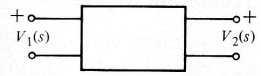
13. Tachometer



$V_2(s) = K_t\omega(s) = K_ts\theta(s)$;

K_t = constant

14. dc-amplifier



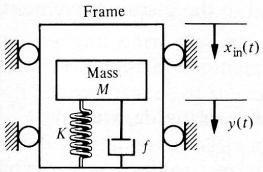
$$\frac{V_2(s)}{V_1(s)} = \frac{k_a}{s\tau + 1}$$

R_o = output resistance

C_o = output capacitance

$\tau = R_oC_o$, $\tau \ll 1$ and is often negligible for servomechanism amplifier

15. Accelerometer



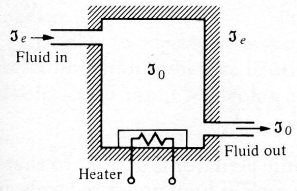
$x_o(t) = y(t) - x_{in}(t)$,

$$\frac{X_o(s)}{X_{in}(s)} = \frac{-s^2}{s^2 + (f/M)s + K/M}$$

For low-frequency oscillations, where $\omega < \omega_n$,

$$\frac{X_o(j\omega)}{X_{in}(j\omega)} \simeq \frac{\omega^2}{K/M}$$

16. Thermal heating system



$$\frac{\tau(s)}{q(s)} = \frac{1}{C_ts + (QS + 1/R)}, \text{ where}$$

$\tau = \tau_o - \tau_e$ = temperature difference due to thermal process

C_t = thermal capacitance

Q = fluid flow rate = constant

S = specific heat of water

R_t = thermal resistance of insulation

$q(s)$ = rate of heat flow of heating element

Arrivée de l'informatique

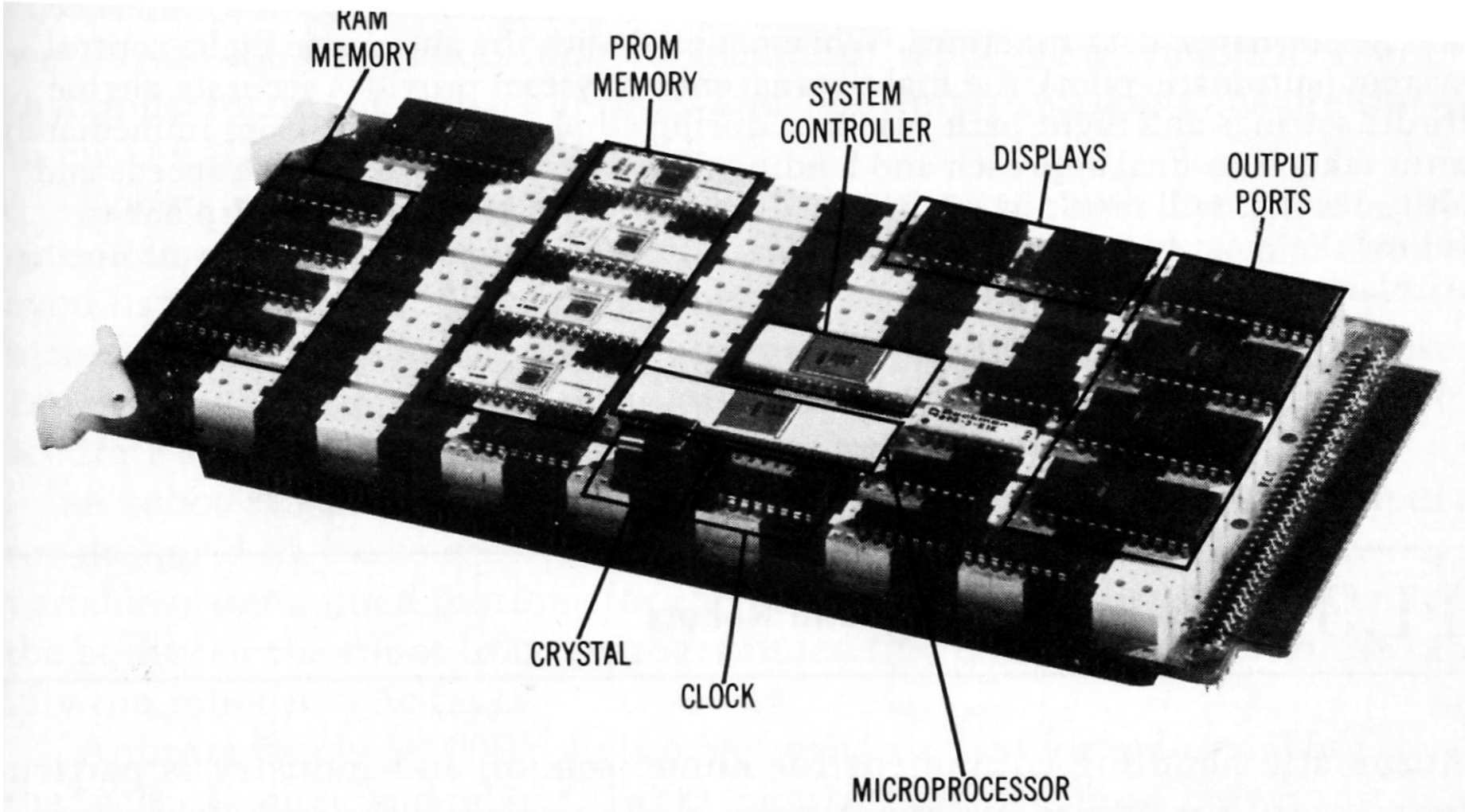
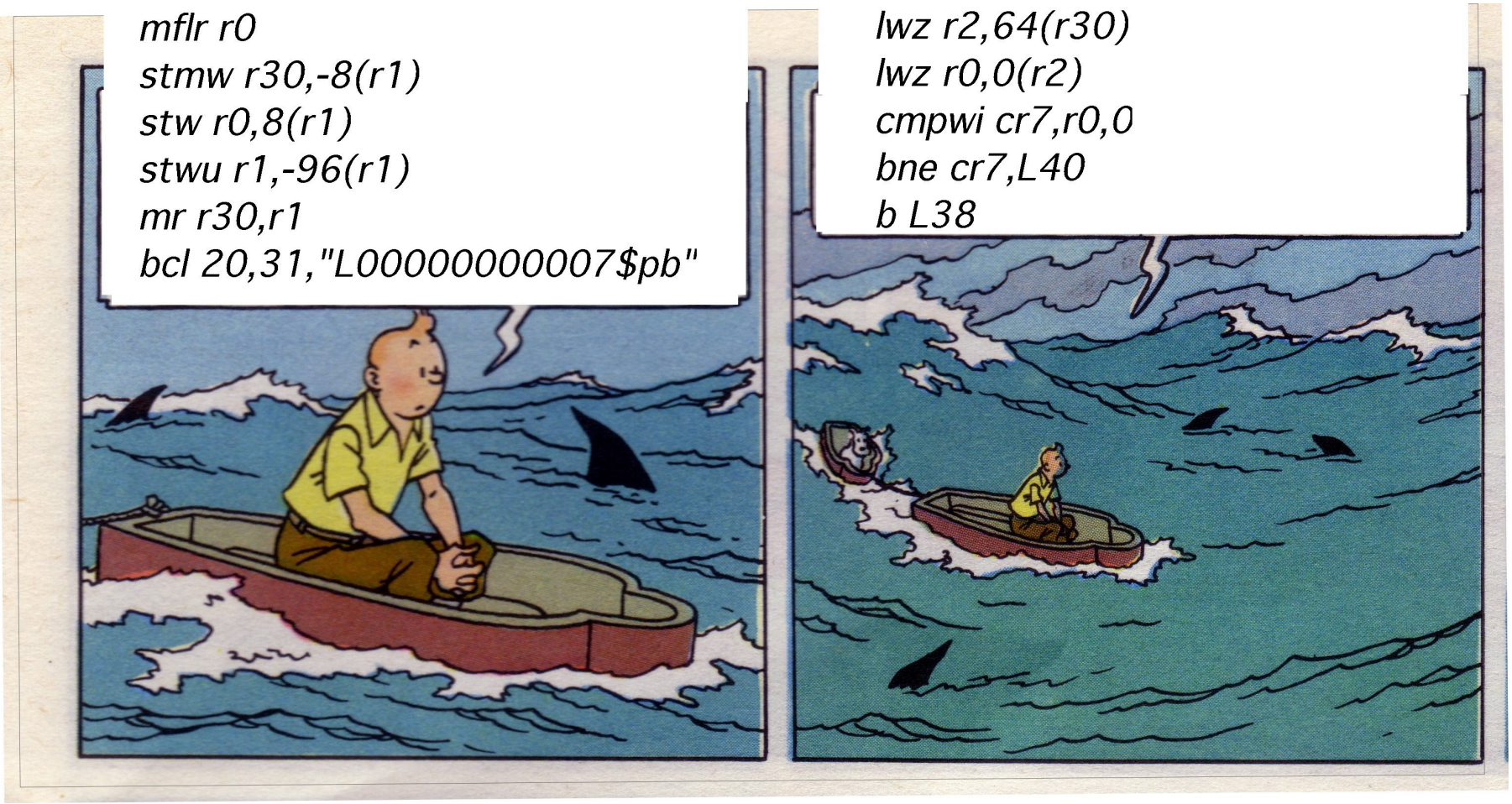
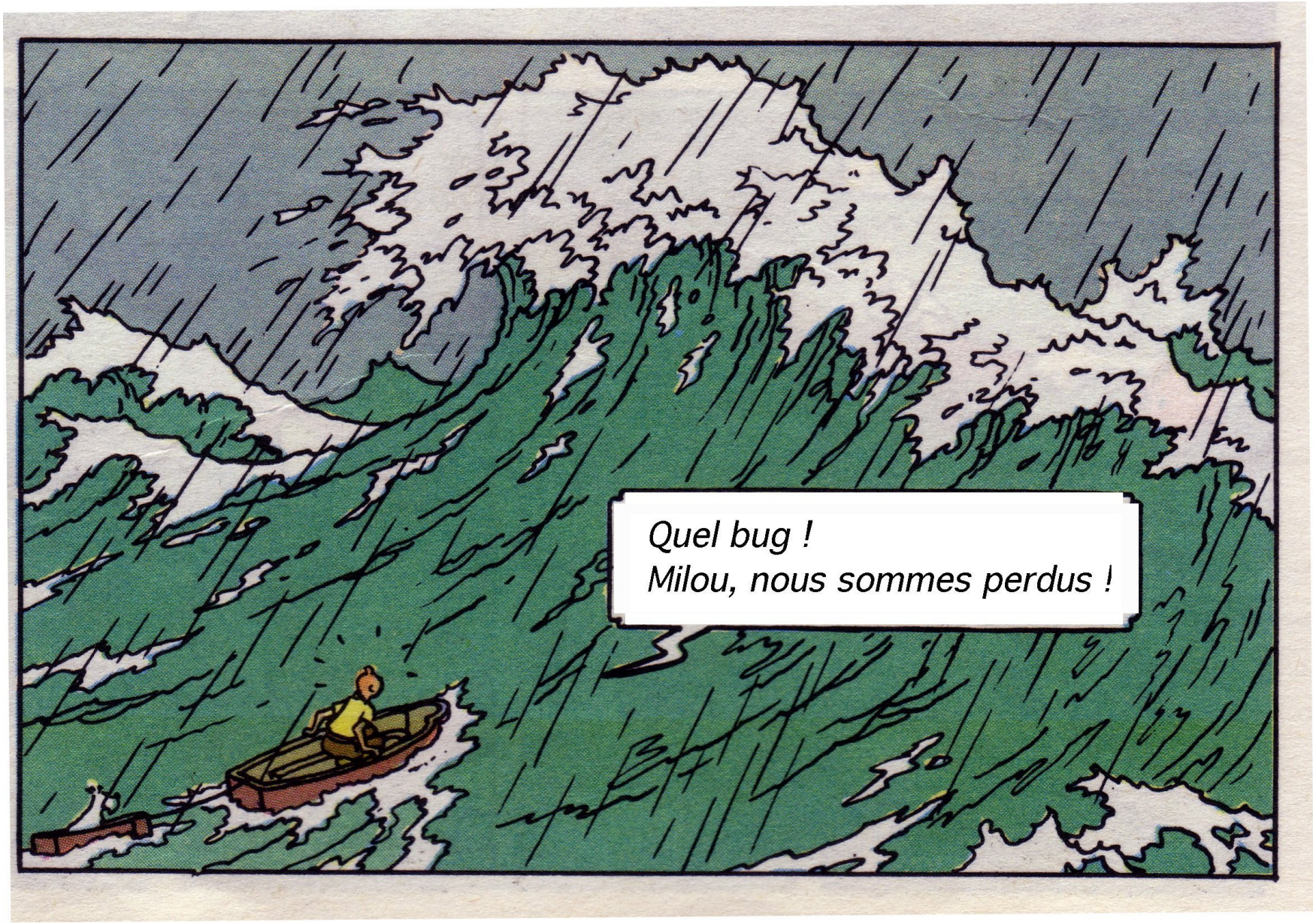


Figure 11.7. A small microcomputer built on a breadboard. (Courtesy of Western Electric.)

Désarroi de l'automaticien

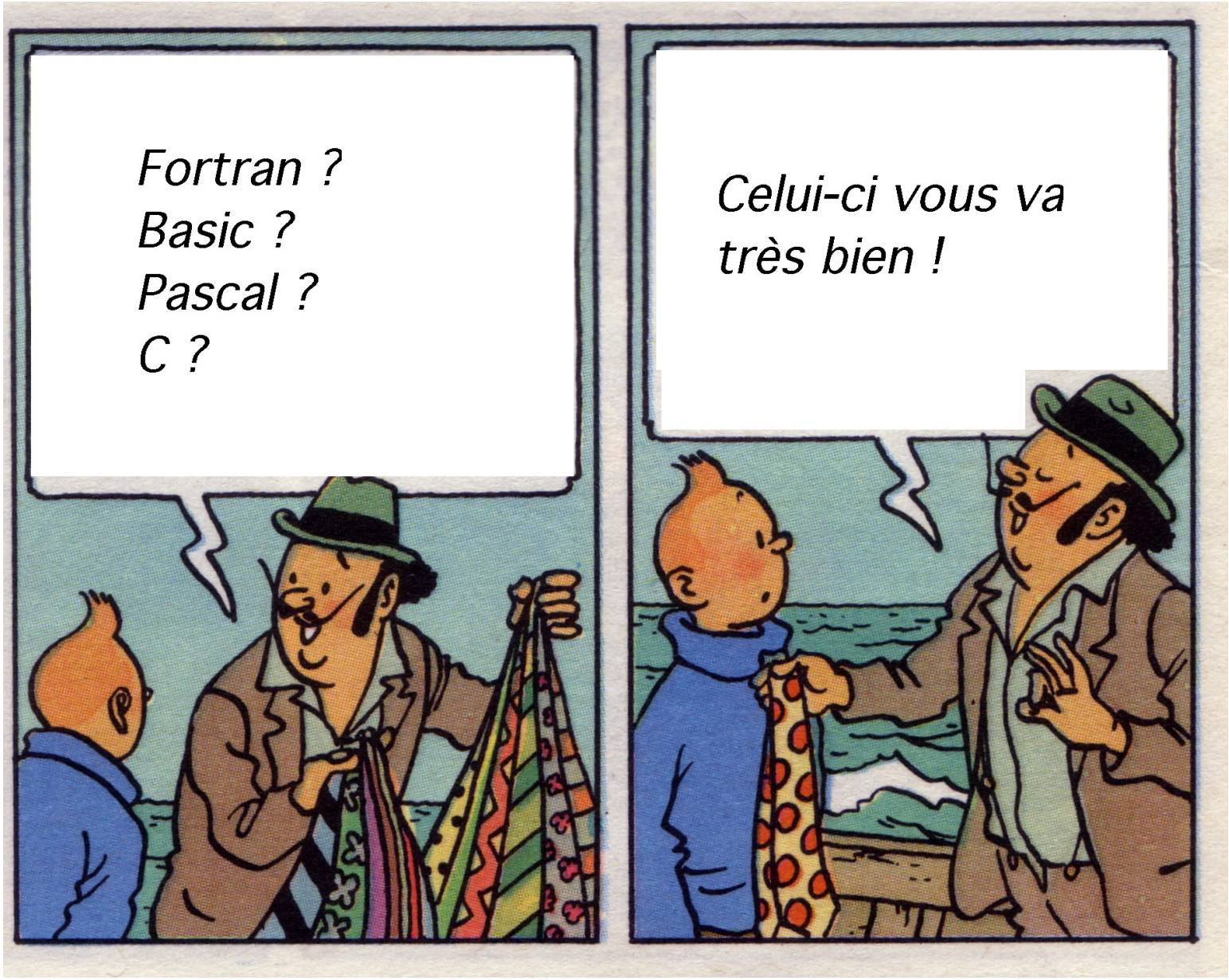


Désarroi de l'automaticien

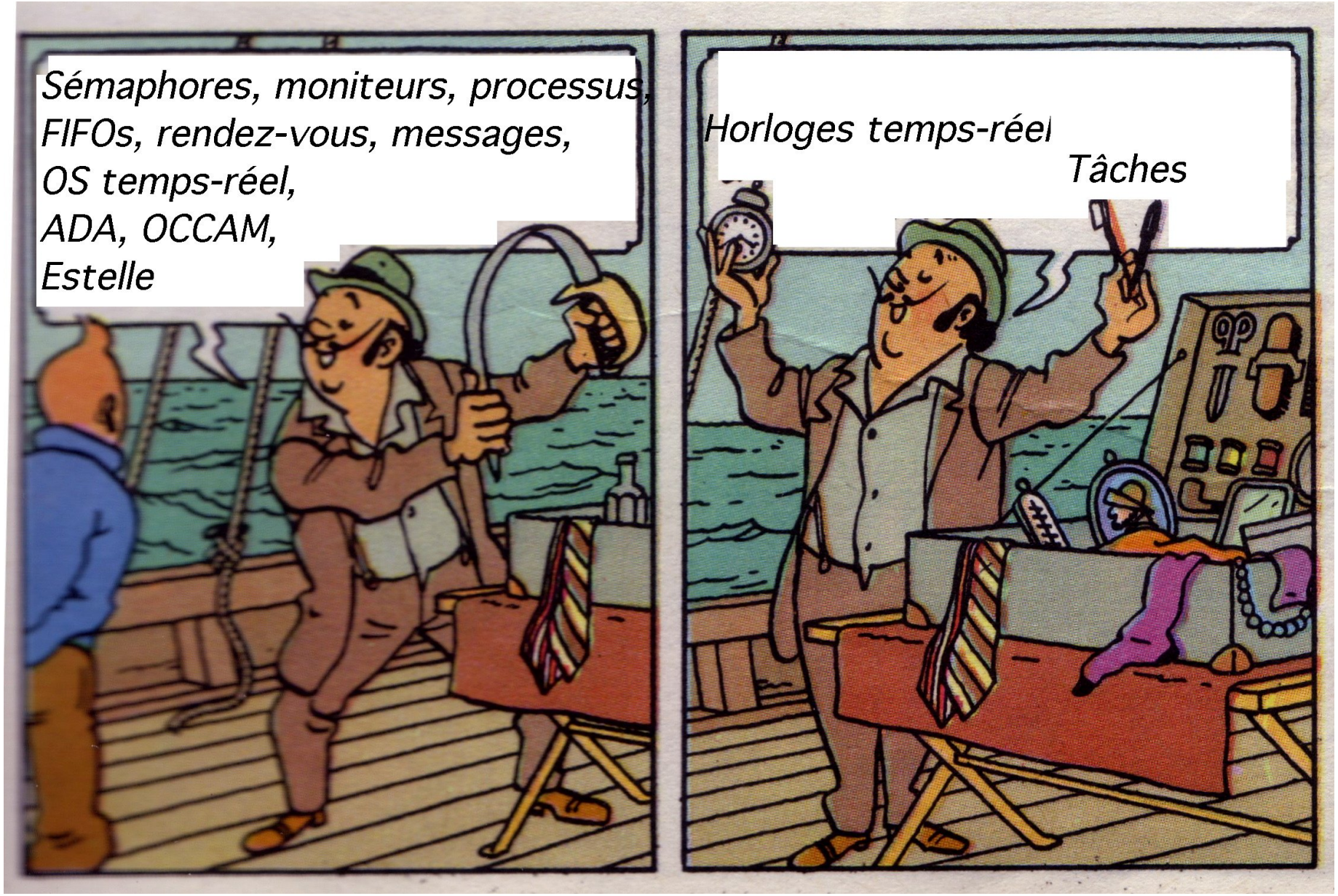


Rencontre avec l'informatique

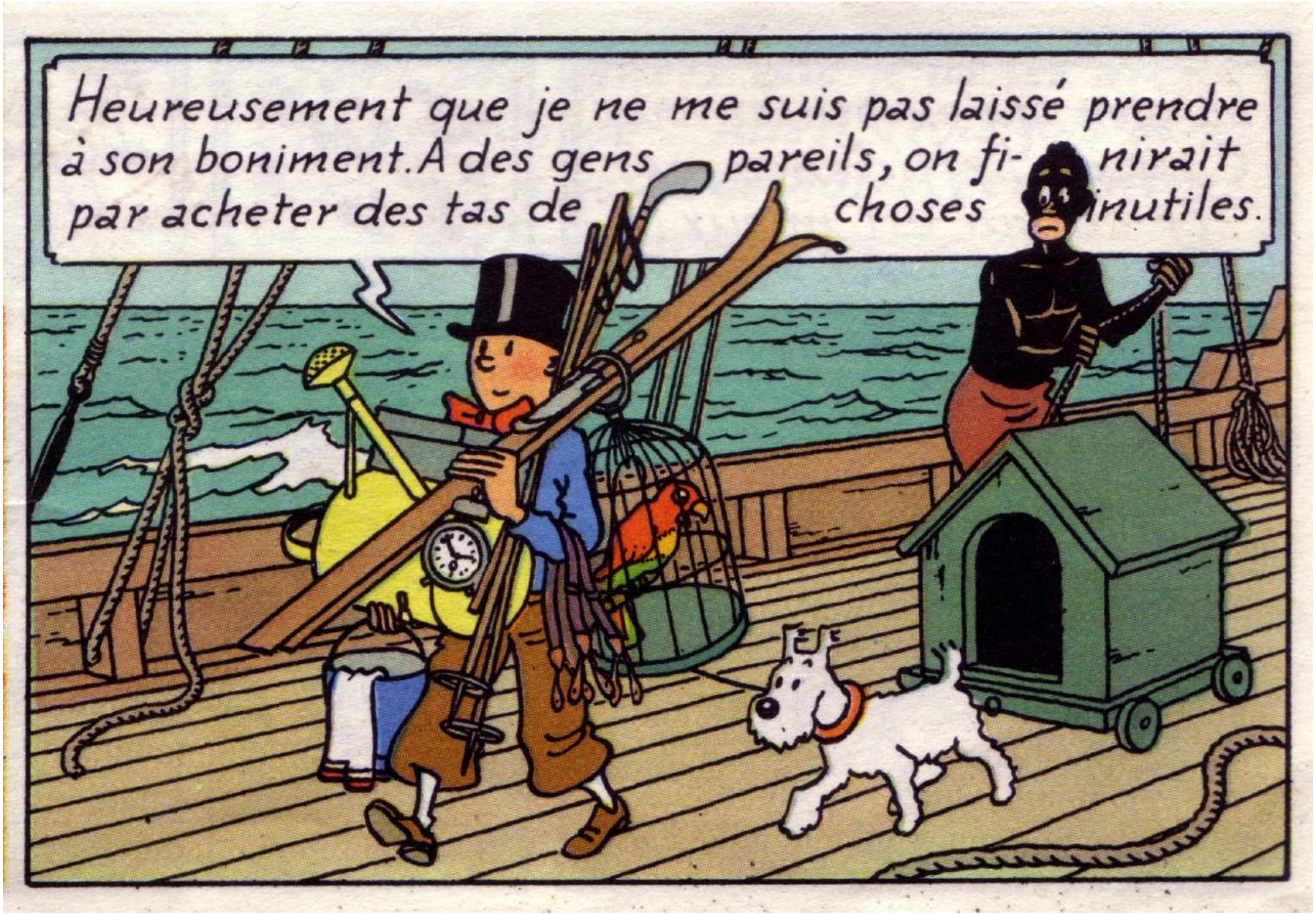
Rencontre avec l'informatique



Rencontre avec l'informatique



Rencontre avec l'informatique

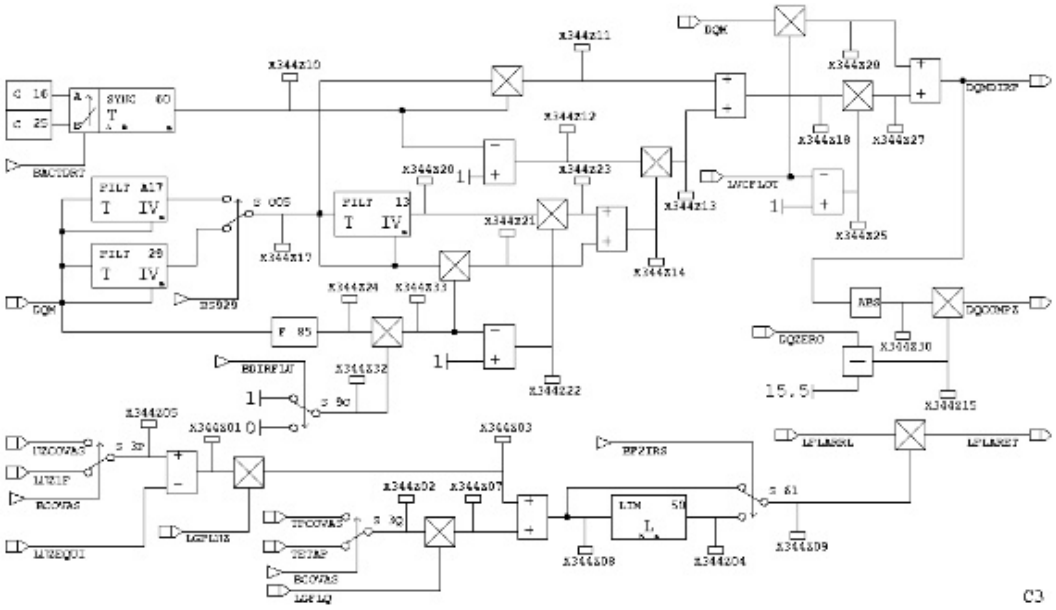


Réaction des praticiens

Réaction des praticiens

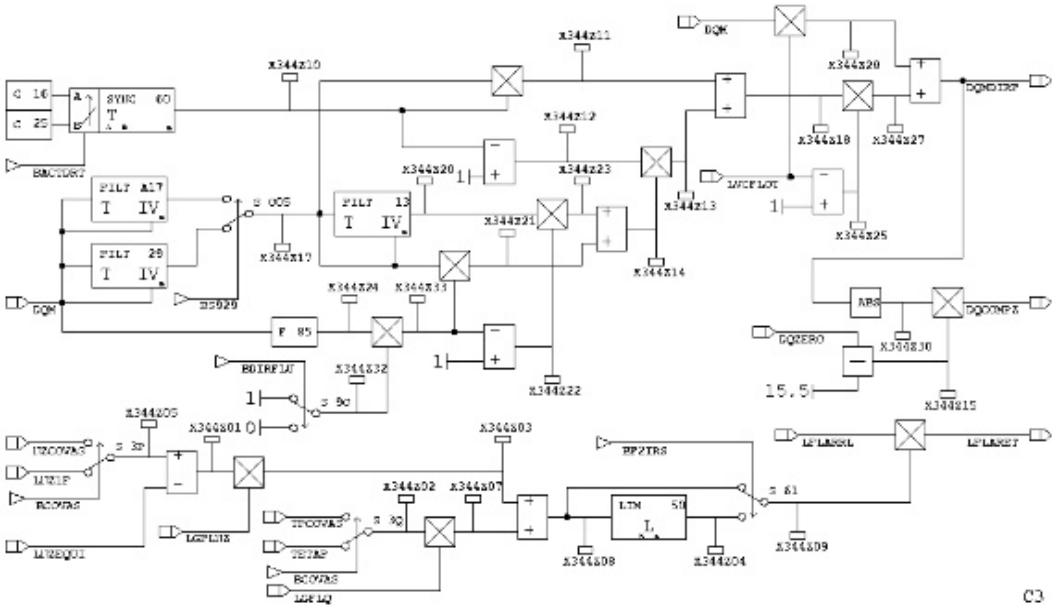
A l'Aérospatiale dans les années 80

control models (block-diagrams)



A l'Aérospatiale dans les années 80

control models (block-diagrams)

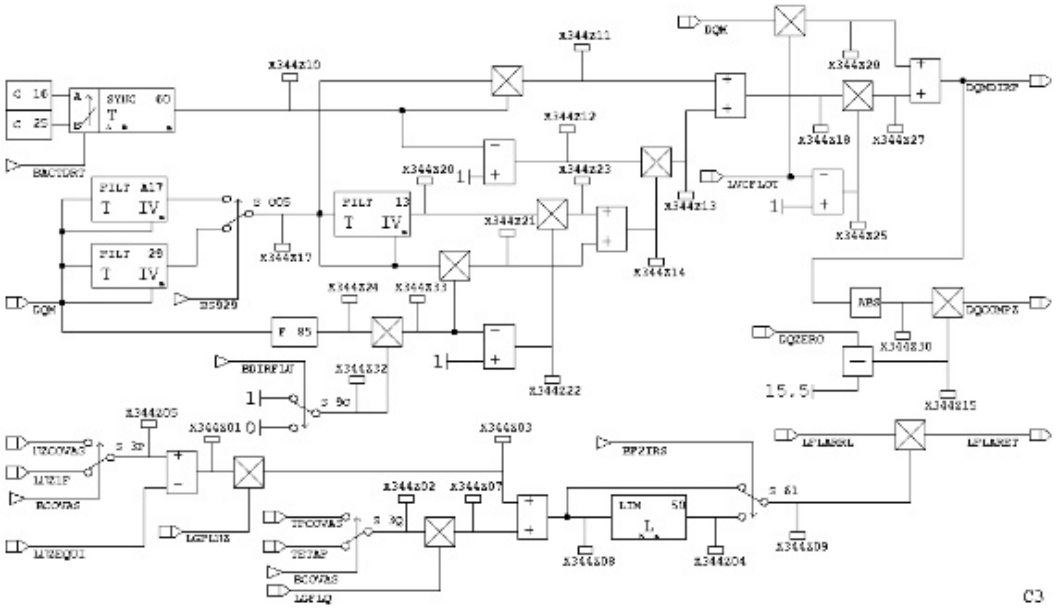


=

formal software specification

A l'Aérospatiale dans les années 80

control models (block-diagrams)



=

formal software specification

↓

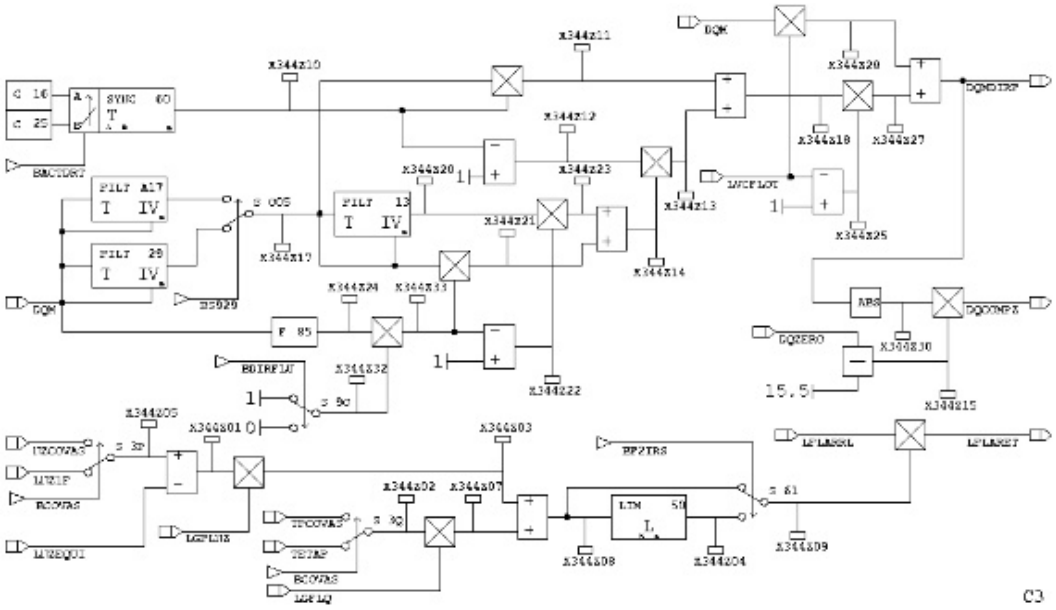
automatic code generation

↓

Software

A l'Aérospatiale dans les années 80

control models (block-diagrams)



= **formal software specification**

↓

automatic code generation

↓

Software

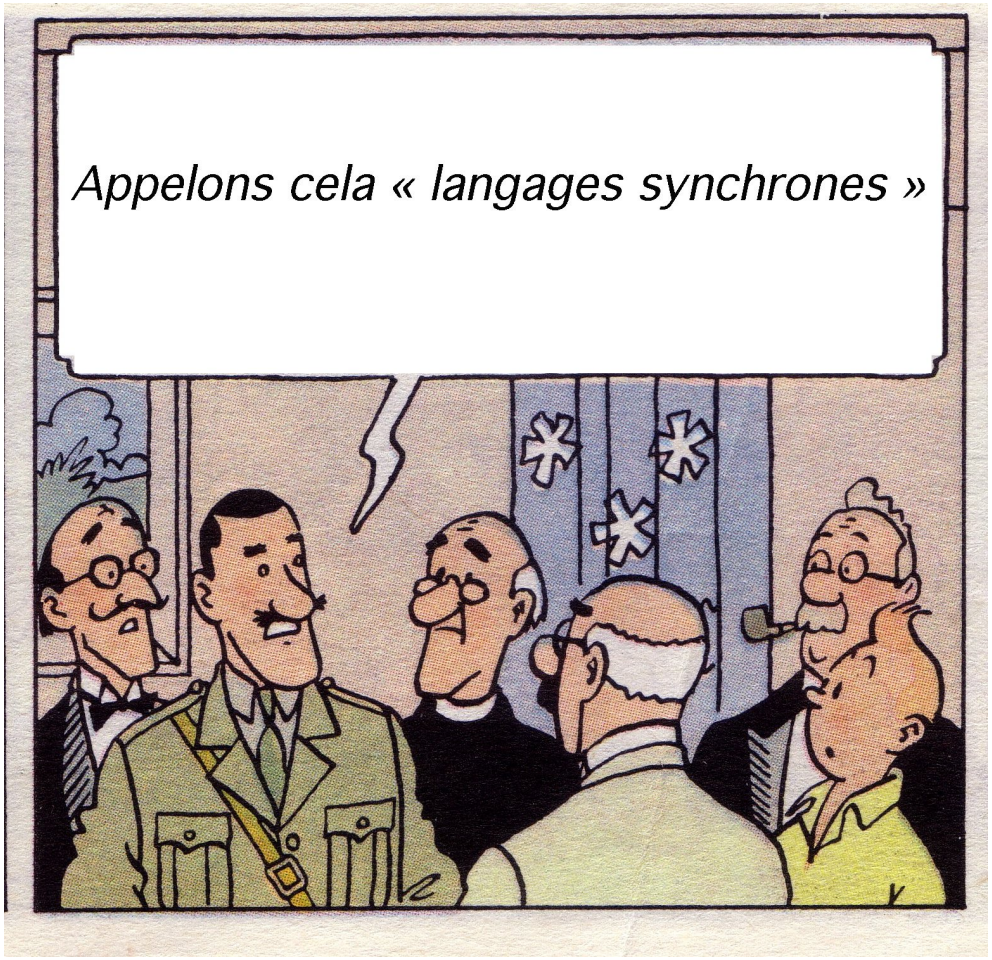
"Spécification Assistée par Ordinateur"(SAO)

"Computer Aided Specification"

Intérêt de l'approche

- Formalisme proche des habitudes des gens (ingénieurs, clients, fournisseurs, certifieurs, pilotes d'essai)
- Fortes propriétés mathématiques
- Parallélisme, communications résolus par compilation
- code objet séquentiel, simple, robuste

Le rôle des théoriciens



Conforter, formaliser, généraliser

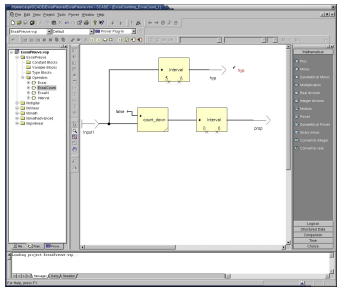
Optimiser

Outiller

Depuis SAO. . .

De puissants outils de développement par modèles :

- **SAO remplacé parSCADE**

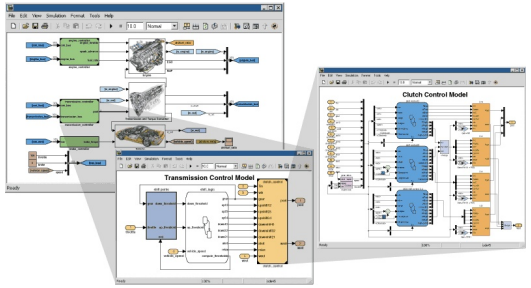


produit commercial partiellement fondé sur la technologie

synchrone de

génération de code qualifiée Do178B level A

- **Simulink/Stateflow**



boite à outil à temps continu/discret

standard de fait en modélisation automatique

- **outils de méthodes formelles**

PROVER TECHNOLOGY

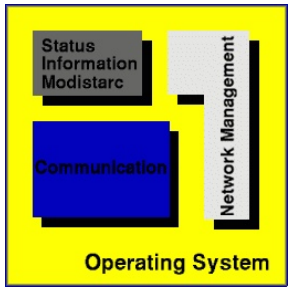
PolySpace TECHNOLOGIES
Delivering safe code · faster

. . .

Depuis SAO. . .

Plateformes d'exécution plus diverses

- multi-tâche



WIND RIVER

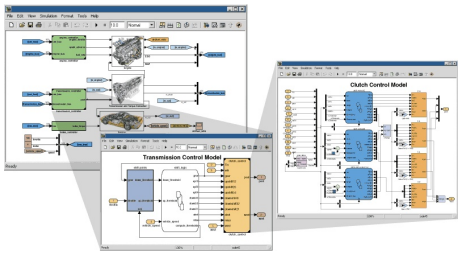
- multi-processeur et réparti

TTTech

FlexRay™

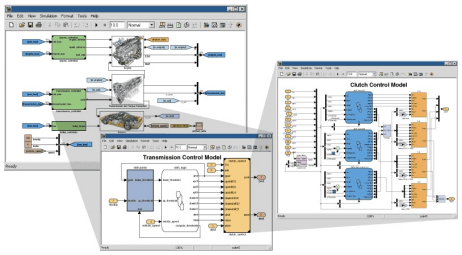
Etat de l’art

modelling

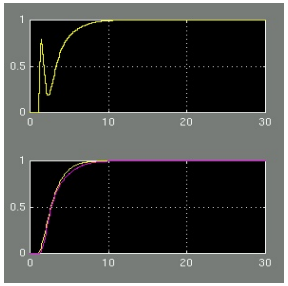


Etat de l’art

modelling

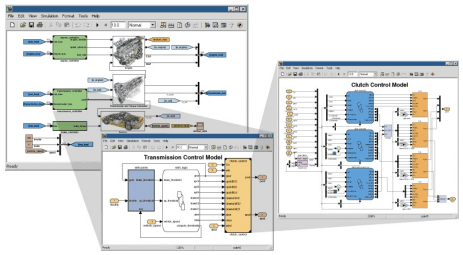


simulation debugging

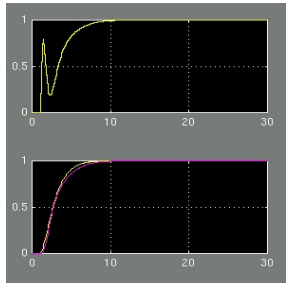


Etat de l’art

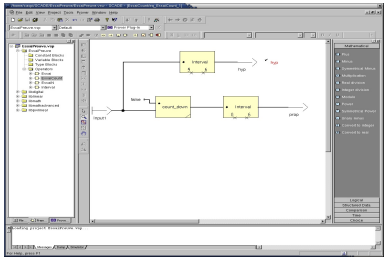
modelling



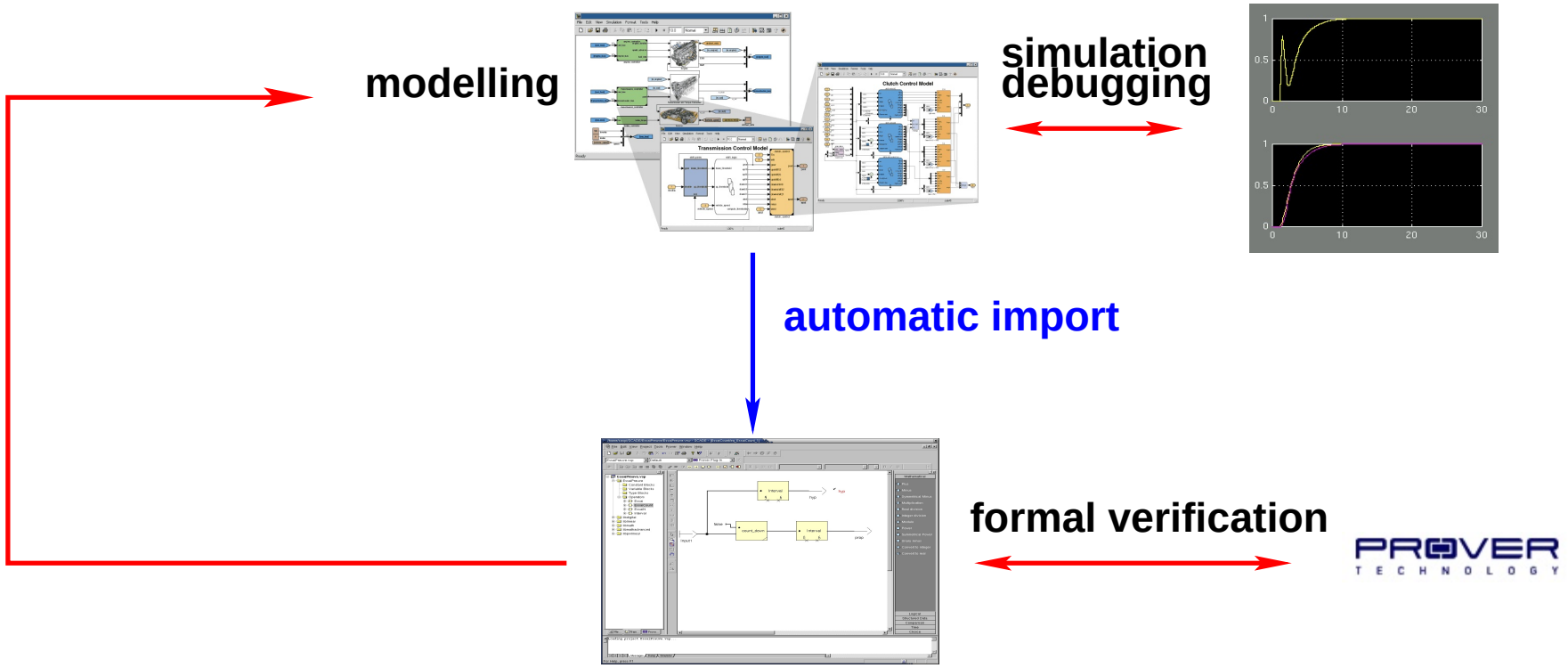
simulation
debugging



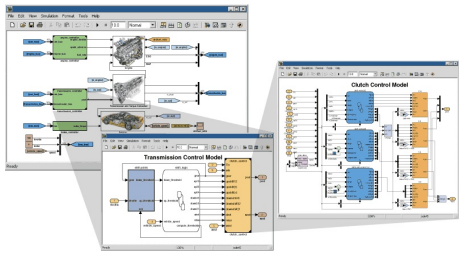
automatic import



Etat de l’art

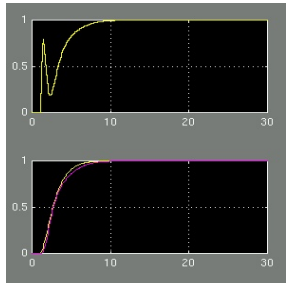

modelling
simulation
debugging
automatic import
formal verification
PROVER
TECHNOLOGY

Etat de l'art

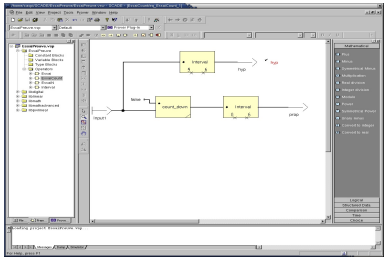


modelling

simulation
debugging



automatic import



formal verification

PROVER TECHNOLOGY

architecture choice
automatic code generation

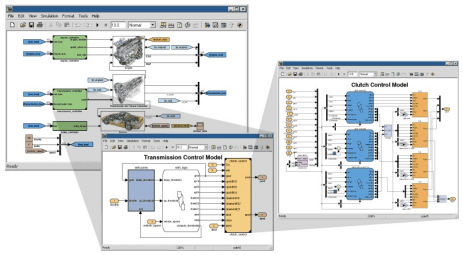
WIND RIVER

Status Information Modistarc
Communication
Network Management
Operating System

TTTech

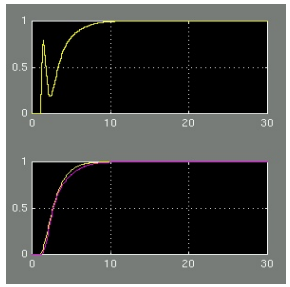
FlexRay™

Etat de l'art

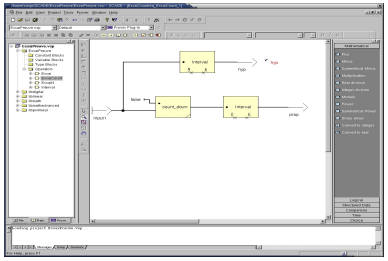


modelling

simulation
debugging



automatic import



formal verification

PROVER TECHNOLOGY

architecture choice
automatic code generation

tests

WIND RIVER

Status Information Modistarc
Communication
Network Management
Operating System

TTTech

FlexRay™

Un point fondamental : la fidélité

What you { *model*, *simulate*, *prove* } **is what you** { *implement*, *execute* }

(Gérard Berry 1984)

En 20 ans, on est passé de ça. . .

à ça. . .

Les méthodes ne sont pas exclusives

Exemples :

- **METEOR**
  - **Analyse système classique (expérience MAGGALY)**
  - **puis développement en B**
- **Commandes de vol Airbus**
  - **Redondances logicielles**
  - **Développement classique**
  - **Génération automatique de code (SCADE)**
  - **Interprétation abstraite (temps d'exécution, erreurs)**
  - **Vérification par modèles**

    ProverPlugin intégré à SCADE

De l’artisanat à l’industrie

En vingt ans, l’informatique des automatismes critiques est passée

- **de l’artisanat :**

  conception papier, codage manuel, validation sur cible
- **à l’industrie :**

  conception logique et architecturale, validation sur modèles formels vérifiables et simulables,

  génération de code automatique assurant l’identité des comportements entre les modèles et leur déploiements

C’est un progrès considérable qu’il faut encore poursuivre, renforcer et étendre

- **plus d’architectures, vérification formelle plus automatique, applications plus hétérogènes (automatismes et télécoms,...)**

Quelques leçons

- **Le monde de la recherche**
- **Comment rencontrer la pratique ?**
- **Quelques pistes de réflexion**

Le monde de la recherche...

est souvent un monde clos,

cloisonné en communautés qui ne se parlent pas

avec ses rites

- **articles,**
- **conférences,...**

qui dessinent un monde de connaissance idéal...

mais parfois illusoire

Il est bon, de temps en temps, d'ouvrir les yeux et de regarder le monde tel qu'il est

Comment rencontrer la pratique ?

C'est plus difficile qu'on ne le croit

La recherche industrielle fait souvent barrage à cette rencontre

Système de justifications croisées :

- **le chercheur académique montre son implication pratique par sa collaboration avec le chercheur industriel...**
- **le chercheur industriel montre sa compétence scientifique par sa collaboration avec le chercheur académique...**

$\Rightarrow$ contrats communs, régionaux, nationaux, européens

Recommandations

Il faut donc être atypique et curieux,

croyant et sceptique,

il faut à la fois

- **penser le monde tel qu'il devrait être...**
- **et regarder le monde tel qu'il est.**