langages, syntaxe et sémantique

pourquoi & comment ?

Julien Deantoni

objectifs

- Comprendre la différence entre syntaxe, sémantique statique et sémantique comportementale
- Comprendre l'origine des principaux formalismes existants , liés à la notion de Modèle de Calcul (MoC).
- Être en mesure de simuler un modèle en spécifiant sa sémantique comportementale
- Être en mesure de faire des activitées de vérification et validation; et surtout en comprendre les enjeux et problèmes actuels.

→ et surtout en comprendre les enjeux et problèmes actuels.

Agenda

1. D'où les modèles/langages existants peuvent-ils venir ? (et présentation du projet fil rouge)
   - Présentation d'un premier langage exécutable et de l'environnement technologique pour le projet
2. Un langage, sa syntaxe et surtout sa sémantique comportementale
   - Réalisation d'un langage jouet pour comprendre le rôle de chacun des constituants du langage
3. Variantes des sémantiques comportementales
   - Description de différentes manières de donner de la sémantique comportementale
4. Mener une activitée de vérification et de validation
   - Pouvez-vous valider vos modèles ?
5. Au délà de l'utilisation d'un langage unique
   - Notion d'exécution hétérogène, et de co-simulation.
6. Finalisation du projet fil rouge
7. Présentation du projet + Examen

D'OÙ LES MODÈLES PEUVENT'ILS VENIR ?

Les propos tenus lors de cette UE sont teintés par une expérience personnelle et ne se veulent volontairement pas neutres...

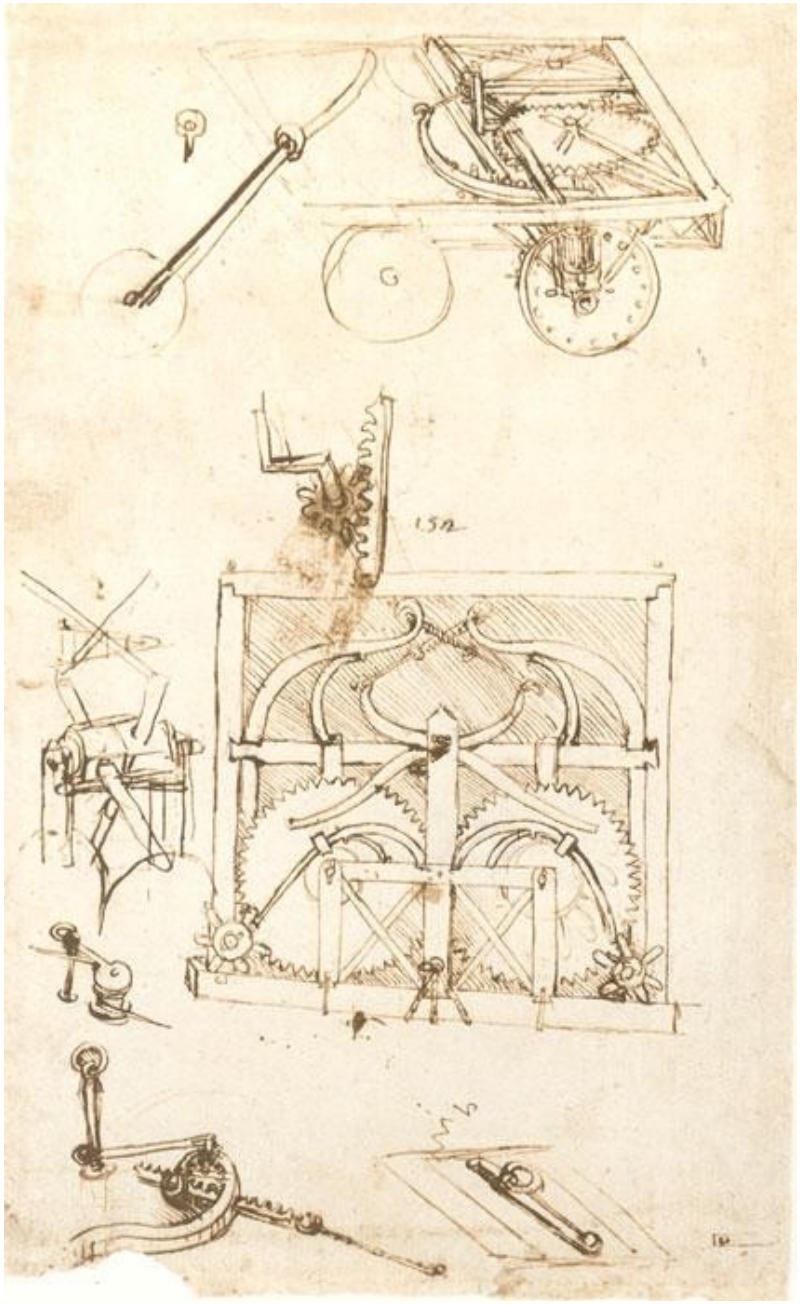
Languages dédiés et modèles

- ...un principe récent et en pleine expansion

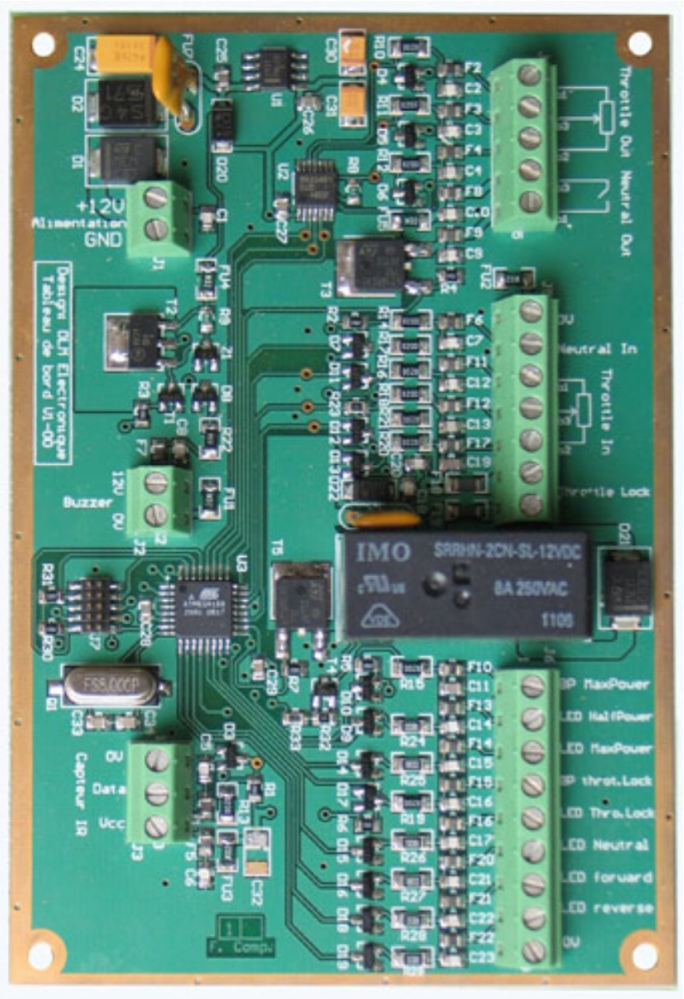
Languages dédiés et modèles

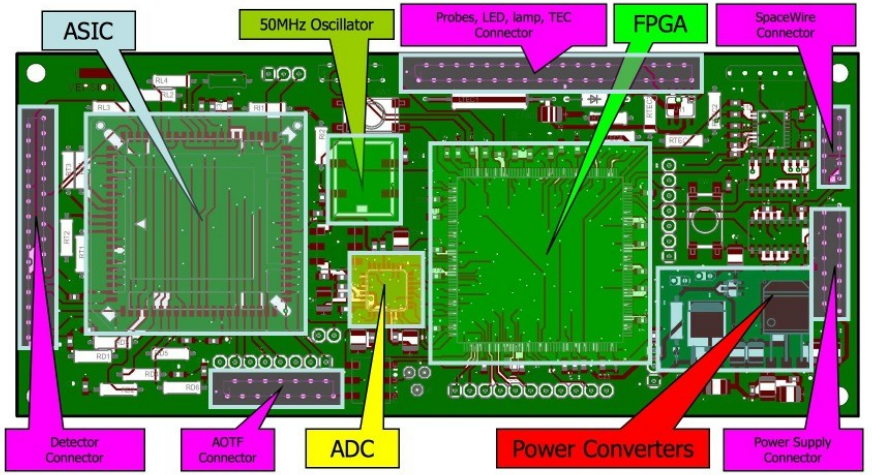
- ...un principe récent !?!?

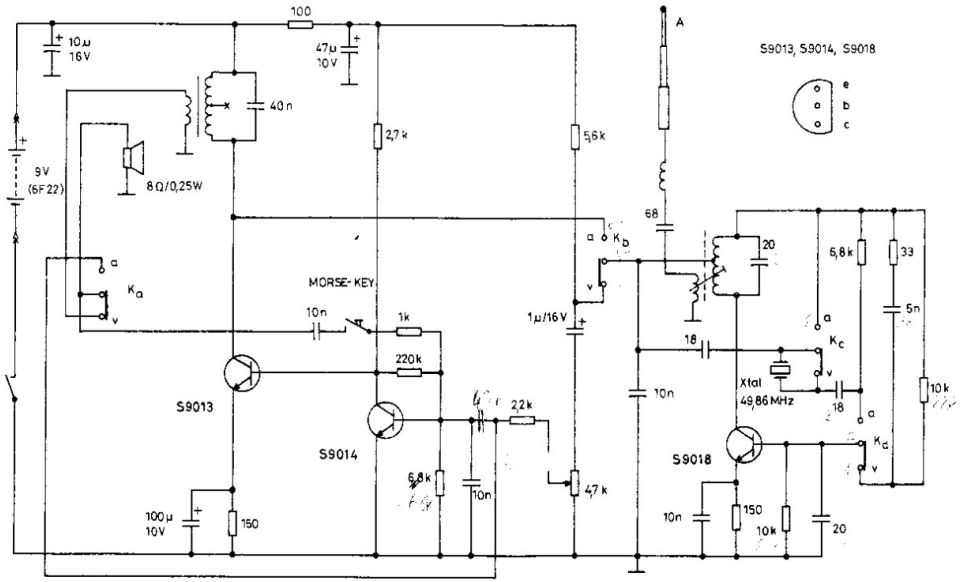
Autour de 1493, Léonard de Vinci



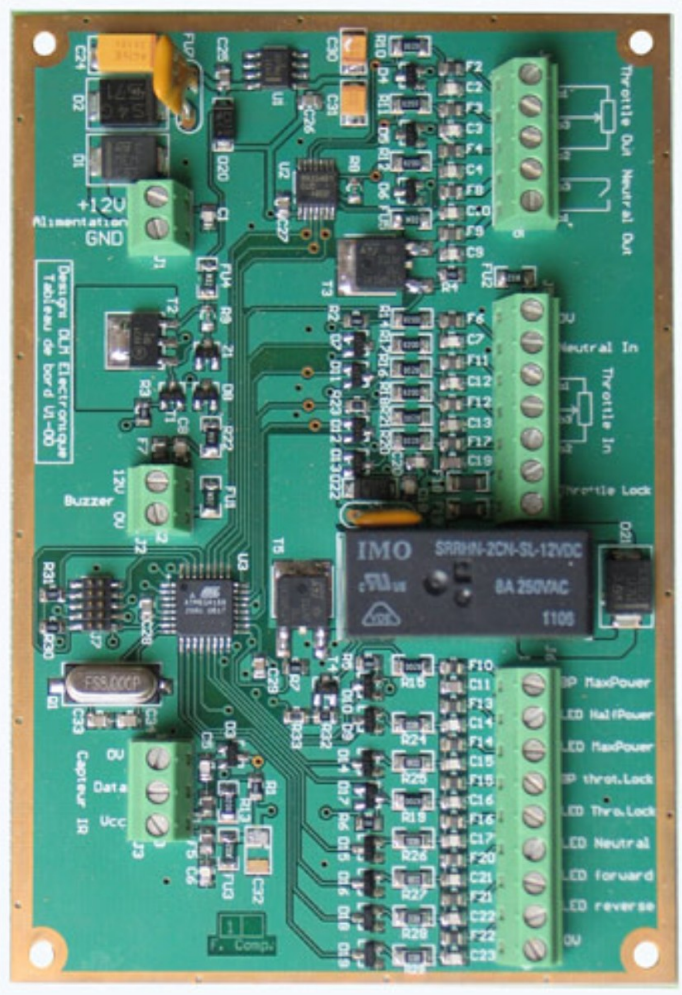
Languages dédiés et modèles

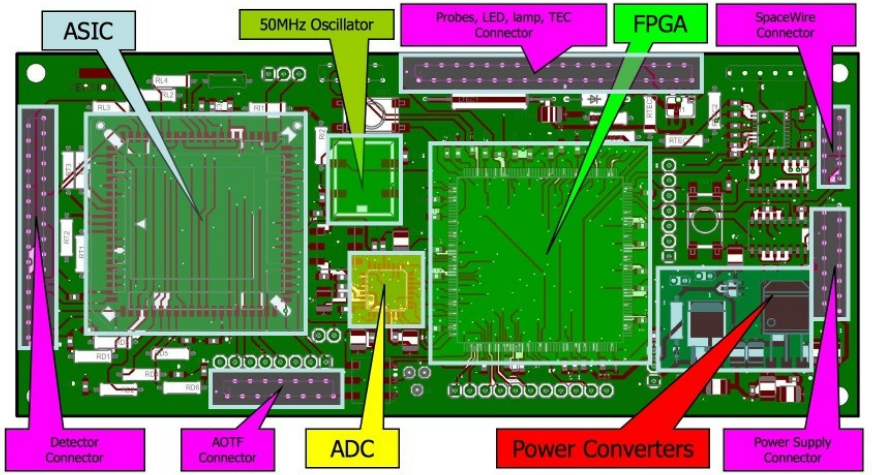


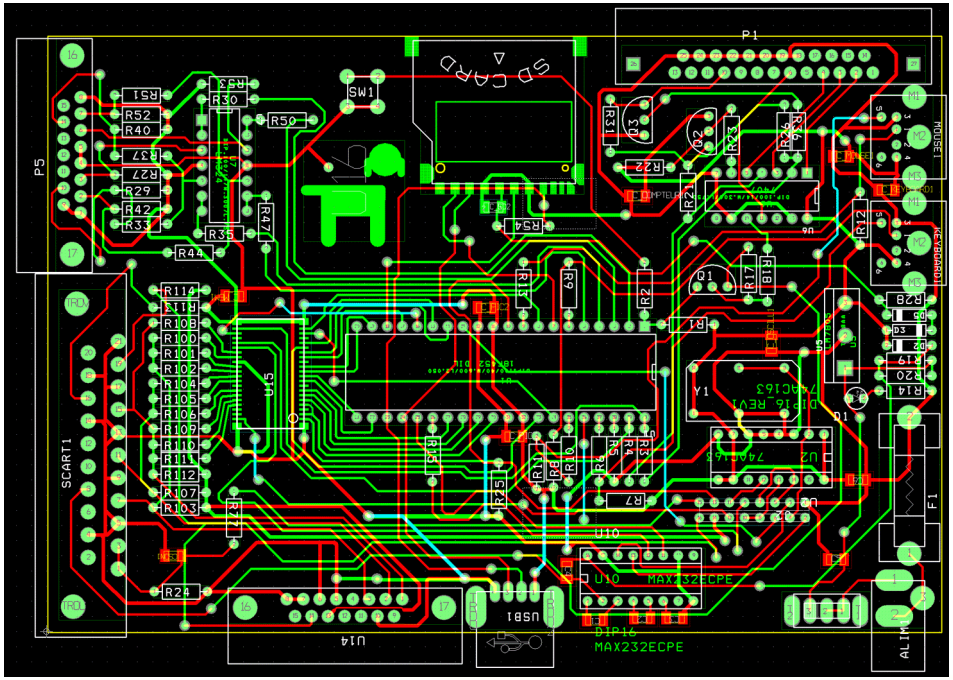




Languages dédiés et modèles







Languages dédiés et modèles

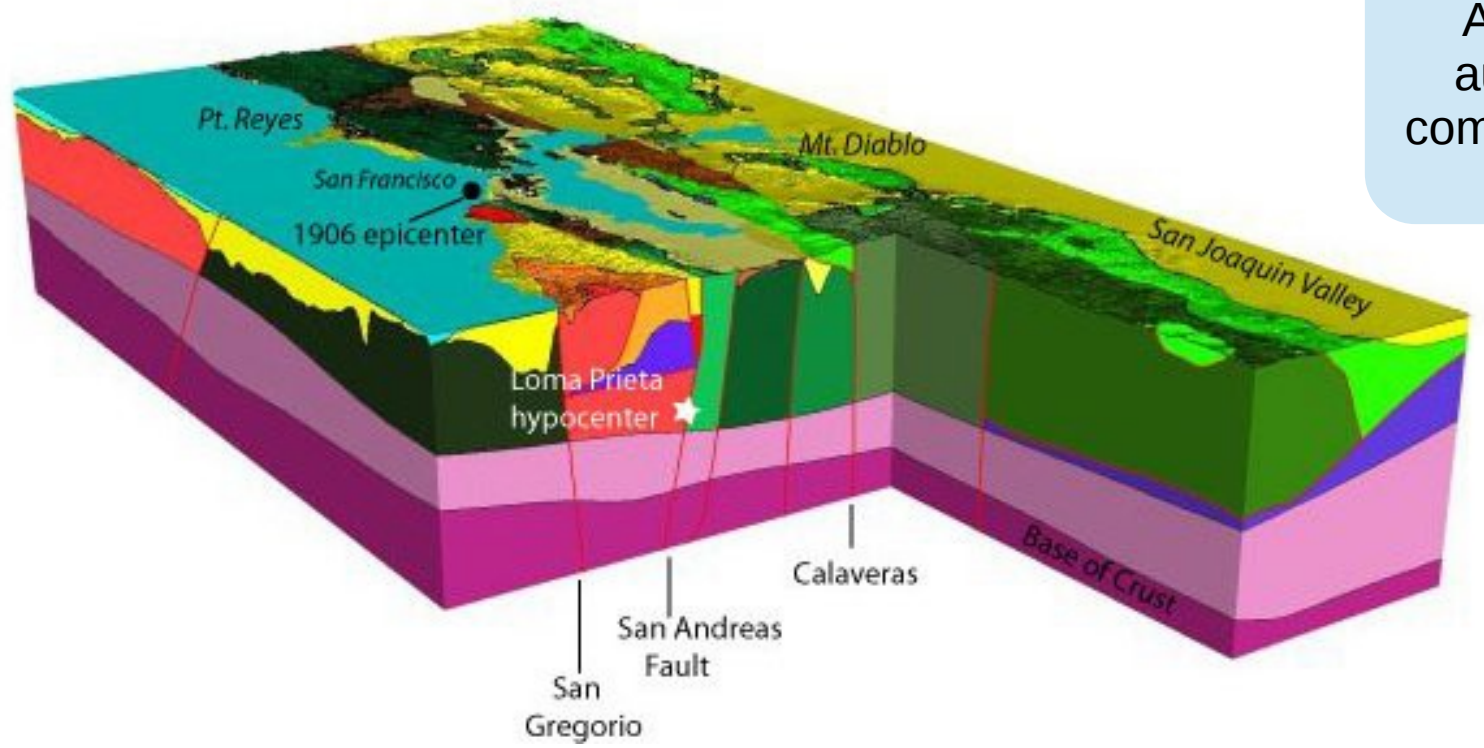
- ...un principe ~~récent et~~ en pleine expansion

Languages dédiés et modèles

- ...un principe ~~récent et~~ avec des technologies en pleines expansions → notion de *software/system language engineering* et de *langage workbench*

Two possible usages of models

- *Descriptive :* Abstraction of an existing reality

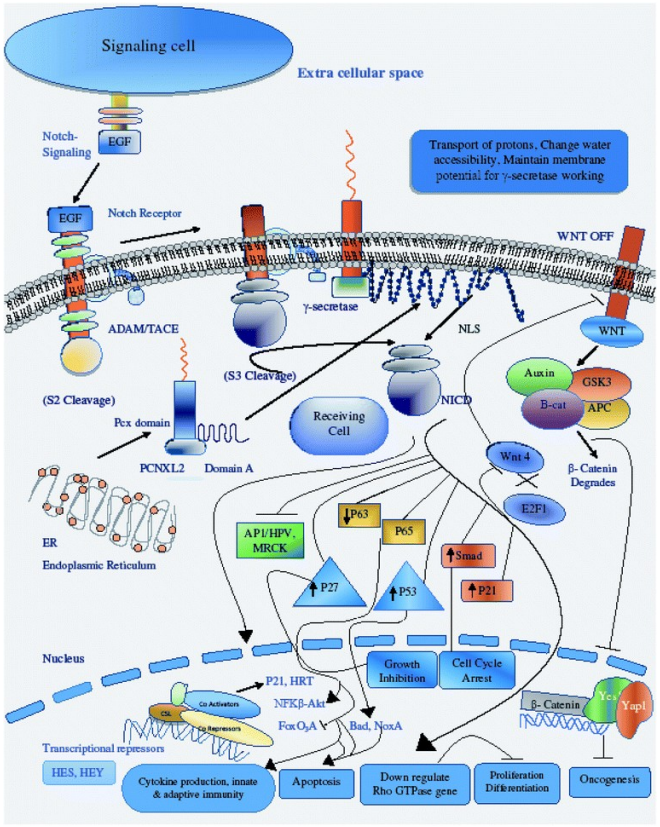


To understand
Analyse
automate
communicate
etc

- *Prescriptive :* Specification of something to be realized

Two possible usages of models

- *Descriptive :* Abstraction of an existing reality

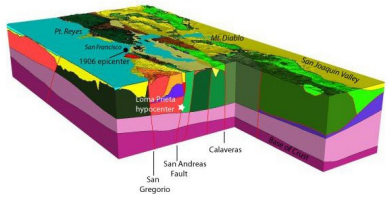


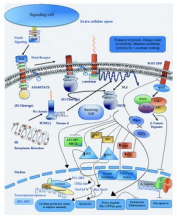
To understand
Analyse
automate
communicate
etc

- *Prescriptive :* Specification of something to be realized

Two usages of models

- *Descriptive :* Abstraction of an existing reality



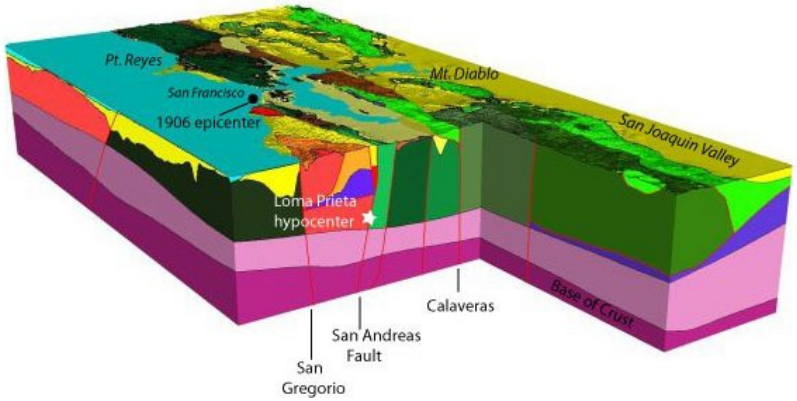


- *Prescriptive :* Specification of something to be realized

To understand
Analyse
automate
communicate
etc

Two usages of models

- *Descriptive :* Abstraction of an existing reality



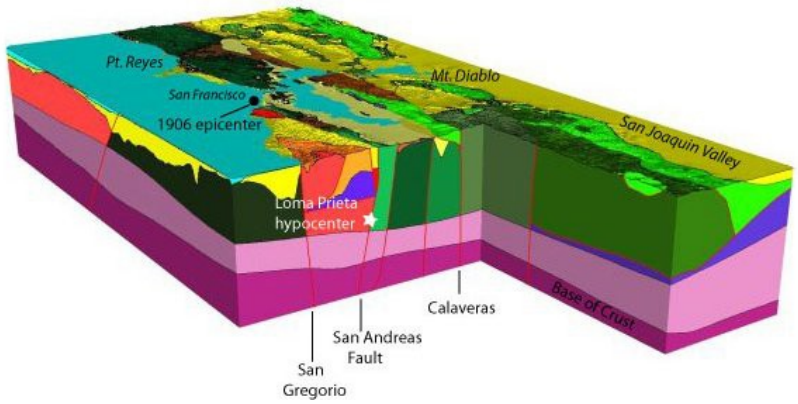
Attention, un même modèle peut jouer différents rôles

- *Prescriptive :* Specification of something to be r ..ized

Il est primordial de définir le rôle et le but de vos modèles

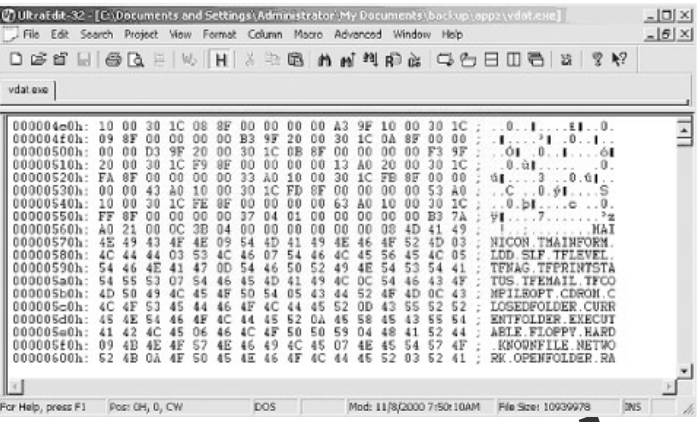
Two usages of models

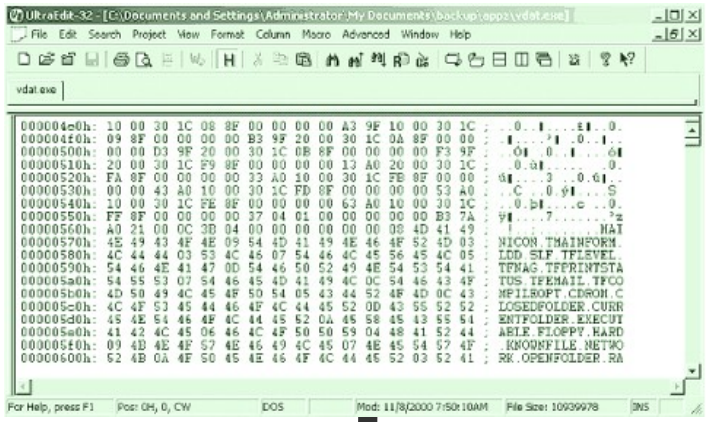
- **Abstraction of an existing reality**

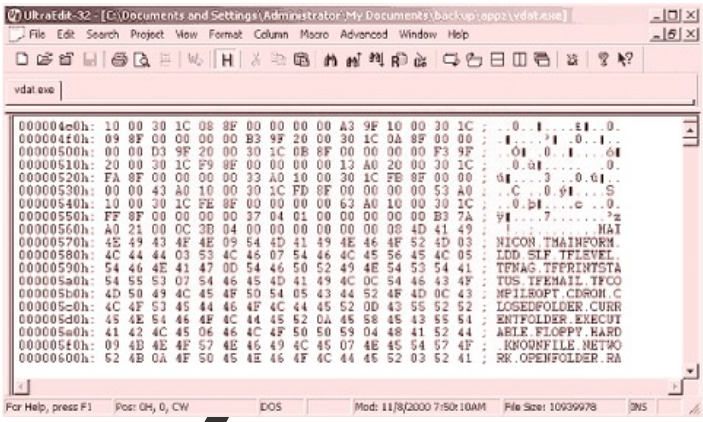


- Specification of something to be realized

Abstraction of an existing reality

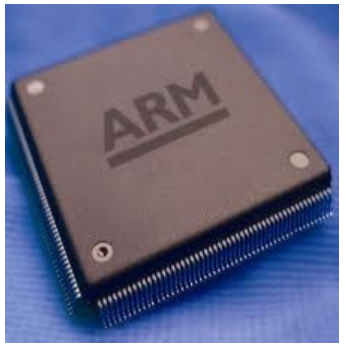






Avant avant ....

Executable par



Abstraction of an existing reality

- langage représentant un certain code exécutable en abstrayant certains aspects

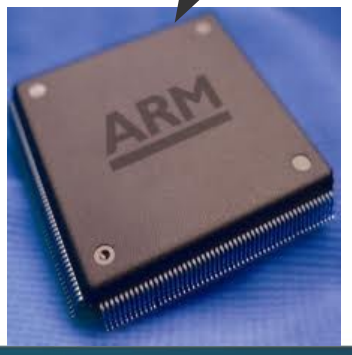
```
;
;  enable the timer for tics
;
ticson:
   ld      a,0
   out0    tmdr0l  ; set timer count lo=0
   out0    rldr0l  ; set reload
   ld      a,tmrtic
   out0    tmdr0h  ; set reload and count=1e00=60 tics/sec
   out0    rldr0h
   ld      a,11h
   out0    tcr ; enable timer 0 + interrupts
   ret
```

```
{
203FA354: 7C0802A6  mflr    r0
203FA358: 90010004  stw     r0,4(SP)
203FA35C: 9421FFE0  stwu    SP,-32(SP)
203FA360: 39610020  addi    r11,SP,32
203FA364: 4800022D  bl      __save_gpr+0x34 (0x203fa590)
    char    *   pDIPSwitch1;
    char    *   pDIPSwitch2;
    char    *   p7Seg;
    char        byVal;
    int         j;
```

```
LDR r0,[p_a]        ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]        ; load b into r1
ADD r3,r0,r1        ; compute a + b
STR r3,[p_w]        ; w = a + b
LDR r2,[p_c]        ; load c into r2
ADD r0,r2,r3        ; compute c + w, reusing r0 for x
STR r0,[p_x]        ; x = c + w
LDR r0,[p_d]        ; load d into r0
ADD r3,r2,r0        ; compute c + d, reusing r3 for y
STR r3,[p_y]        ; y = c + d
```

Avant....

Automatic generation / compilation



Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

sub add load store jump

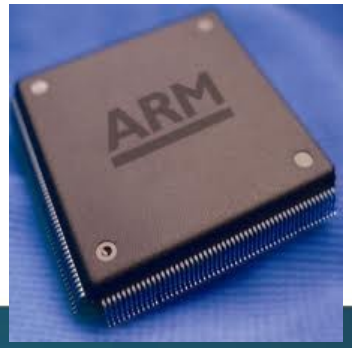
```
;
;  enable the timer for tics
;
ticson:
   ld      a,0
   out0    tmdr01  ; set timer count lo=0
   out0    rldr01  ; set reload
   ld      a,tmrtic
   out0    tmdr0h  ; set reload and count=1e00=60 tics/sec
   out0    rldr0h
   ld      a,11h
   out0    tcr ; enable timer 0 + interrupts
   ret
```

```
{
203FA354: 7C0802A6  mflr    r0
203FA358: 90010004  stw     r0,4(SP)
203FA35C: 9421FFE0  stwu    SP,-32(SP)
203FA360: 39610020  addi    r11,SP,32
203FA364: 4800022D  bl      __save_gpr+0x34 (0x203fa590)
    char    *   pDIPSwitch1;
    char    *   pDIPSwitch2;
    char    *   p7Seg;
    char        byVal;
    int         j;
```

```
LDR r0,[p_a]        ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]        ; load b into r1
ADD r3,r0,r1        ; compute a + b
STR r3,[p_w]        ; w = a + b
LDR r2,[p_c]        ; load c into r2
ADD r0,r2,r3        ; compute c + w, reusing r0 for x
STR r0,[p_x]        ; x = c + w
LDR r0,[p_d]        ; load d into r0
ADD r3,r2,r0        ; compute c + d, reusing r3 for y
STR r3,[p_y]        ; y = c + d
```

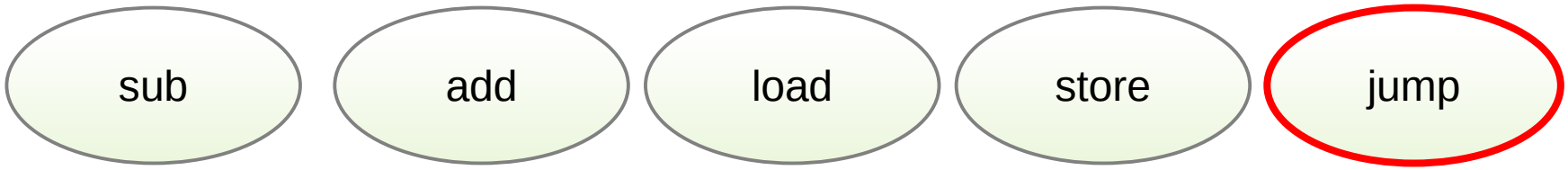
Avant....

Automatic generation / compilation



Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects



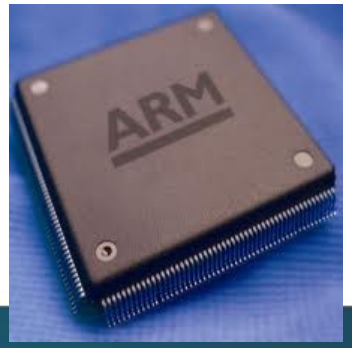

```
;
;  enable the timer for tics
;
ticson:
   ld      a,0
   out0    tmdr0l  ; set timer count lo=0
   out0    rldr0l  ; set reload
   ld      a,tmrtic
   out0    tmdr0h  ; set reload and count=1e00=60 tics/sec
   out0    rldr0h
   ld      a,11h
   out0    tcr ; enable timer 0 + interrupts
   ret
```

```
{
203FA354: 7C0802A6  mflr    r0
203FA358: 90010004  stw     r0,4(SP)
203FA35C: 9421FFE0  stwu    SP,-32(SP)
203FA360: 39610020  addi    r11,SP,32
203FA364: 4800022D  bl      __save_gpr+0x34 (0x203fa590)
    char    *   pDIPSwitch1;
    char    *   pDIPSwitch2;
    char    *   p7Seg;
    char        byVal;
    int         j;
```

```
LDR r0,[p_a]      ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]      ; load b into r1
ADD r3,r0,r1      ; compute a + b
STR r3,[p_w]      ; w = a + b
LDR r2,[p_c]      ; load c into r2
ADD r0,r2,r3      ; compute c + w, reusing r0 for x
STR r0,[p_x]      ; x = c + w
LDR r0,[p_d]      ; load d into r0
ADD r3,r2,r0      ; compute c + d, reusing r3 for y
STR r3,[p_y]      ; y = c + d
```

Avant....

Automatic generation / compilation



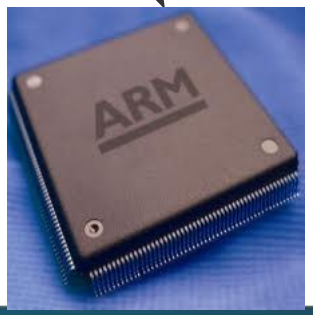
Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

```
nodename = getNodename()
label=symbol.sym_name.get(int(ast[0]),ast[0])
print '    %s [label="%s' % (nodename, label),
if isinstance(ast[1], str):
   if ast[1].strip():
      print '= %s"];' % ast[1]
   else:
      print '"]'
else:
   print '"];'
   children = []
   for n, child in enumerate(ast[1:]):
      children.append(dotwrite(child))
   print '    %s -> {' % nodename,
   for name in children:
      print '%s' % name,
```

Avant hier....

Automatic generation / compilation



Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

Conditions
(if then else, switch case, etc)

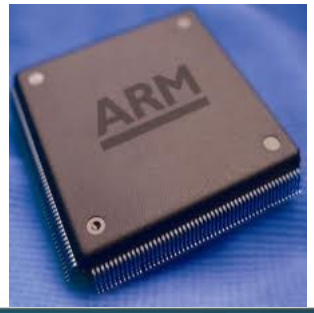
Loop
(while, for, ...)

...

Avant hier....

```
nodename = getNodename()
label=symbol.sym_name.get(int(ast[0]),ast[0])
print '    %s [label="%s' % (nodename, label),
if isinstance(ast[1], str):
   if ast[1].strip():
      print '= %s"];' % ast[1]
   else:
      print '"]'
else:
   print '"];'
   children = []
   for n, child in enumerate(ast[1:]):
      children.append(dotwrite(child))
   print '    %s -> {' % nodename,
   for name in children:
      print '%s' % name,
```

Automatic generation / compilation



Reifying **good** concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

GOTO ?

At the machine code level, a goto is a form of branch or jump statement.

Although at the pre-ALGOL meeting held in 1959, Heinz Zemanek explicitly threw doubts on the necessity of GOTO statements, at the time no one paid attention to his remark, including Edsger Dijkstra, who would later become the iconic opponent of GOTO. [3] The 1970s and 1980s saw a decline in the use of GOTO statements in favor of the "structured programming" paradigm, with goto criticized as leading to "unmaintainable spaghetti code" (see below).

Source: wikipedia

Reifying **good ?** concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

[...]In that letter Dijkstra argued that unrestricted GOTO statements should be abolished from higher-level languages because **they complicated the task of analyzing and verifying the correctness of programs** (particularly those involving loops)

Some programmers, such as Linux Kernel designer and coder Linus Torvalds or software engineer and book author Steve McConnell, also object to Dijkstra's point of view, stating that **GOTOs can be a useful language feature, improving program speed, size and code clearness**, but only when used in a sensible way by a comparably sensible programmer.

Source: wikipedia

Reifying **good ?** concepts

Good is a point of view

- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

[...]In that letter Dijkstra argued that unrestricted GOTO statements should be abolished from higher-level languages because **they complicated the task of analyzing and verifying the correctness of programs** (particularly those involving loops)

Some programmers, such as Linux Kernel designer and coder Linus Torvalds or software engineer and book author Steve McConnell, also object to Dijkstra's point of view, stating that **GOTOs can be a useful language feature, improving program speed, size and code clearness**, but only when used in a sensible way by a comparably sensible programmer.

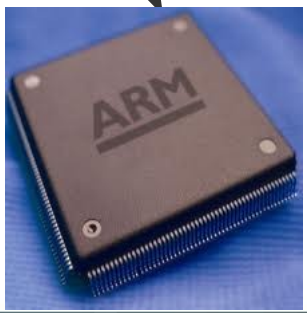
REIFYING **GOOD ?** CONCEPTS

Good is a point of view

- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

```
nodename = getNodename()
label=symbol.sym_name.get(int(ast[0]),ast[0])
print '    %s [label="%s' % (nodename, label),
if isinstance(ast[1], str):
   if ast[1].strip():
      print '= %s"];' % ast[1]
   else:
      print '"]'
else:
   print '"];'
   children = []
   for n, child in enumerate(ast[1:]):
      children.append(dotwrite(child))
   print '    %s -> {' % nodename,
   for name in children:
      print '%s' % name,
```

Hier....



Automatic generation / compilation

REIFYING GOOD ? CONCEPTS

Good is a point of view

- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

Procedures (and functions)
Stress on structuring control flow
Less concern for structuring data

Modules
Group together data and procedures manipulating the data
Access control, information hiding, encapsulation
Separate the specification (interface) from the implementation (body)

Abstract data types
The module is turned into a type
One may declare instances (i.e. objects) of this type

Object-orientation
Hierarchy among abstract data types (inheritance)
Dynamic (run-time) resolution of the exact type
of an object (dynamic typing, late binding, polymorphism)

REIFYING **GOOD ?** CONCEPTS

Good is a point of view

- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

Il est primordial de définir le rôle et le but de vos modèles et donc de vos langages

REIFYING **GOOD ?** CONCEPTS

Good is a point of view

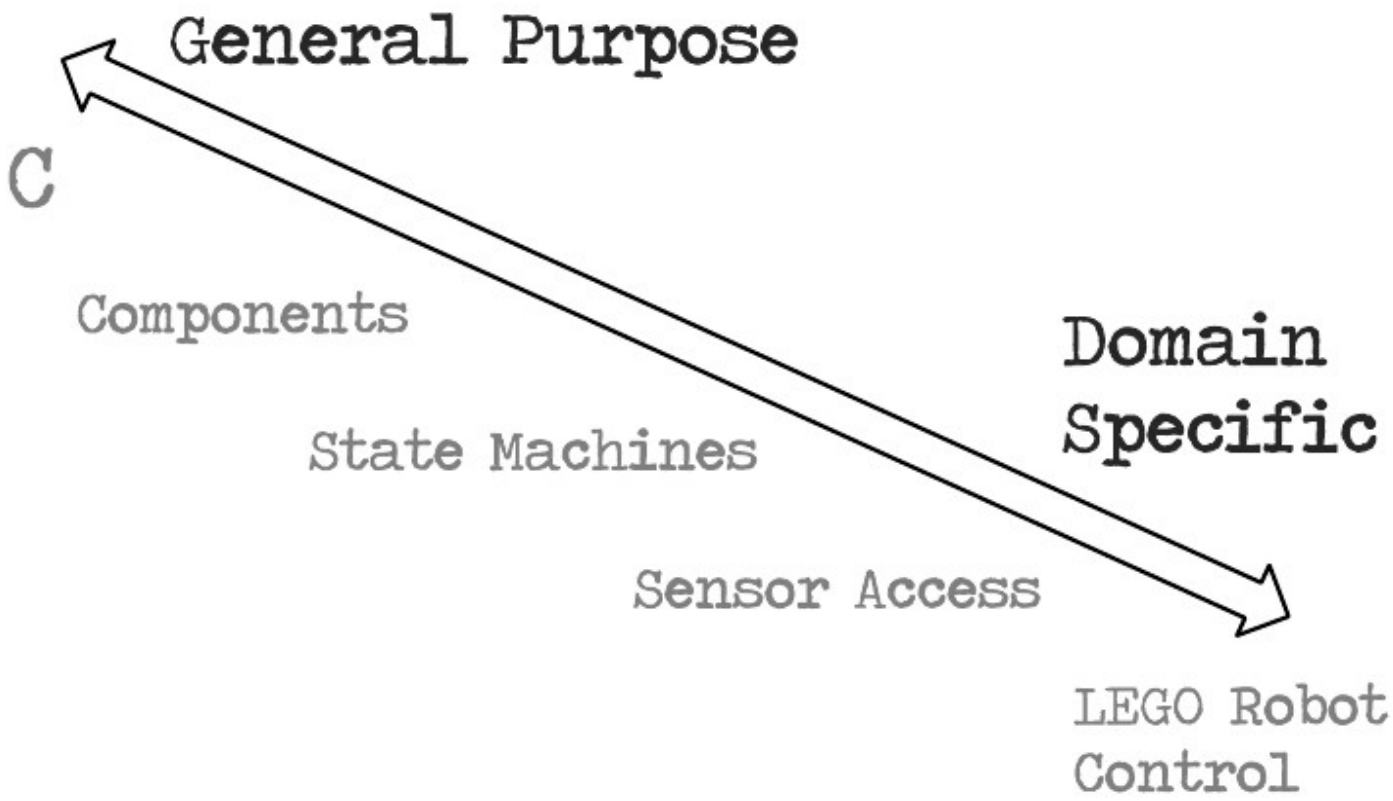
- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

Il est primordial de définir le rôle et le but de vos modèles et donc de vos langages

Use the most appropriate language(s) at the right level(s) of abstraction

Hans Vangheluwe

General purpose VS Domain specific

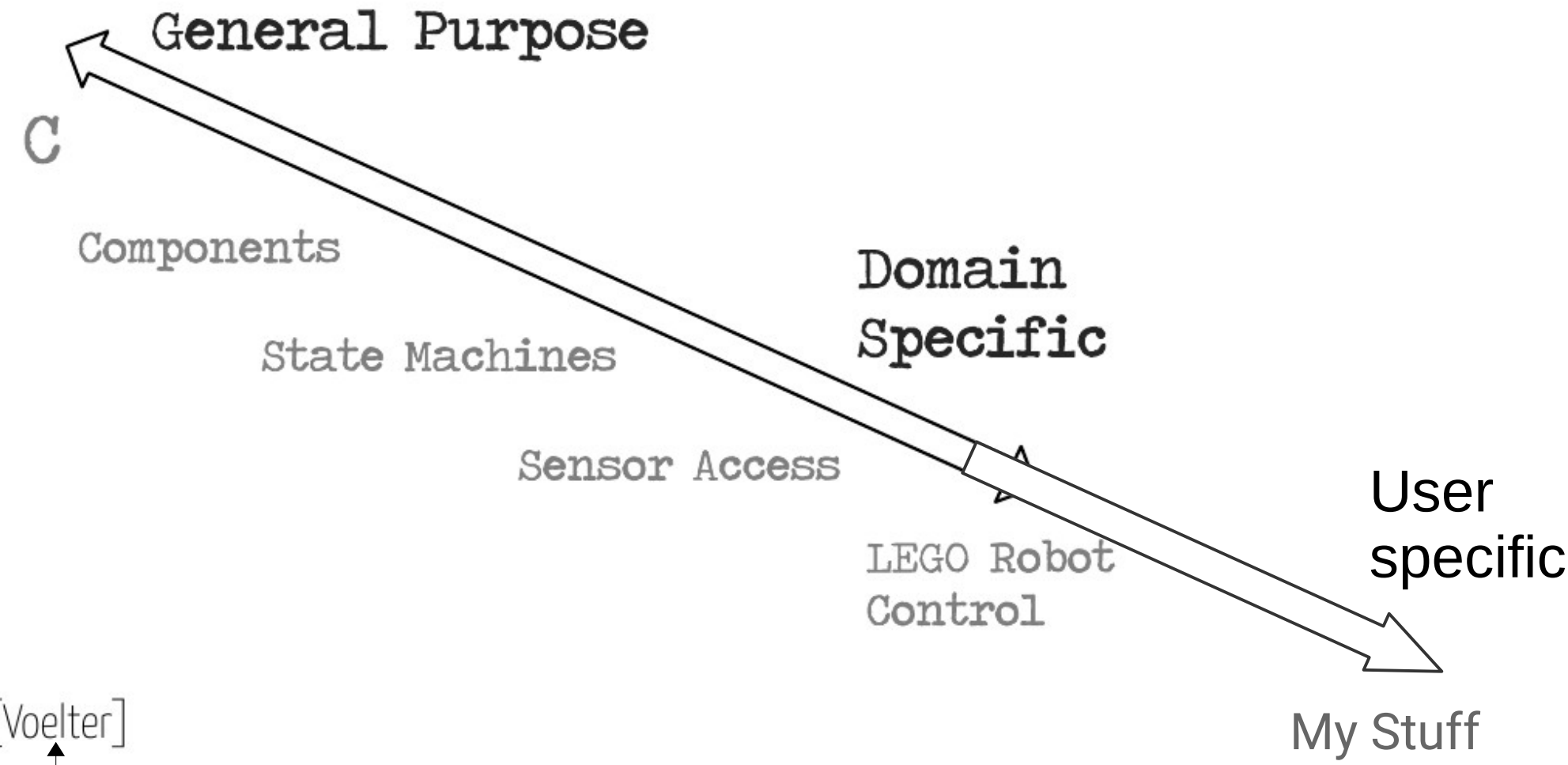


[Voelter]

[Mosser]

C'est **une** des manières de classer les modèles

General purpose VS Domain specific



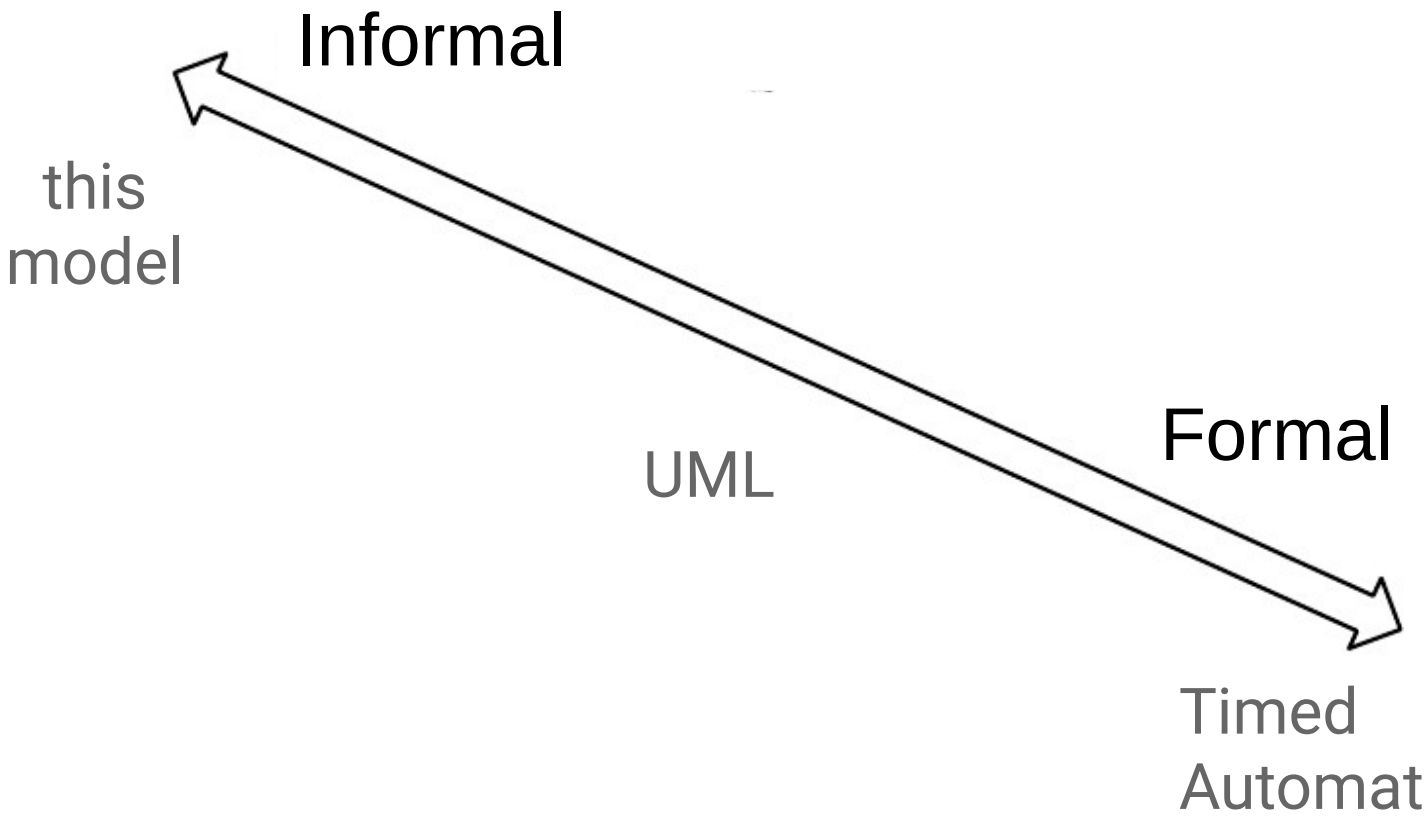
[Voelter]

[Mosser]

[Deantoni]

C'est **une** des manières de classer les modèles

Informal VS Formal



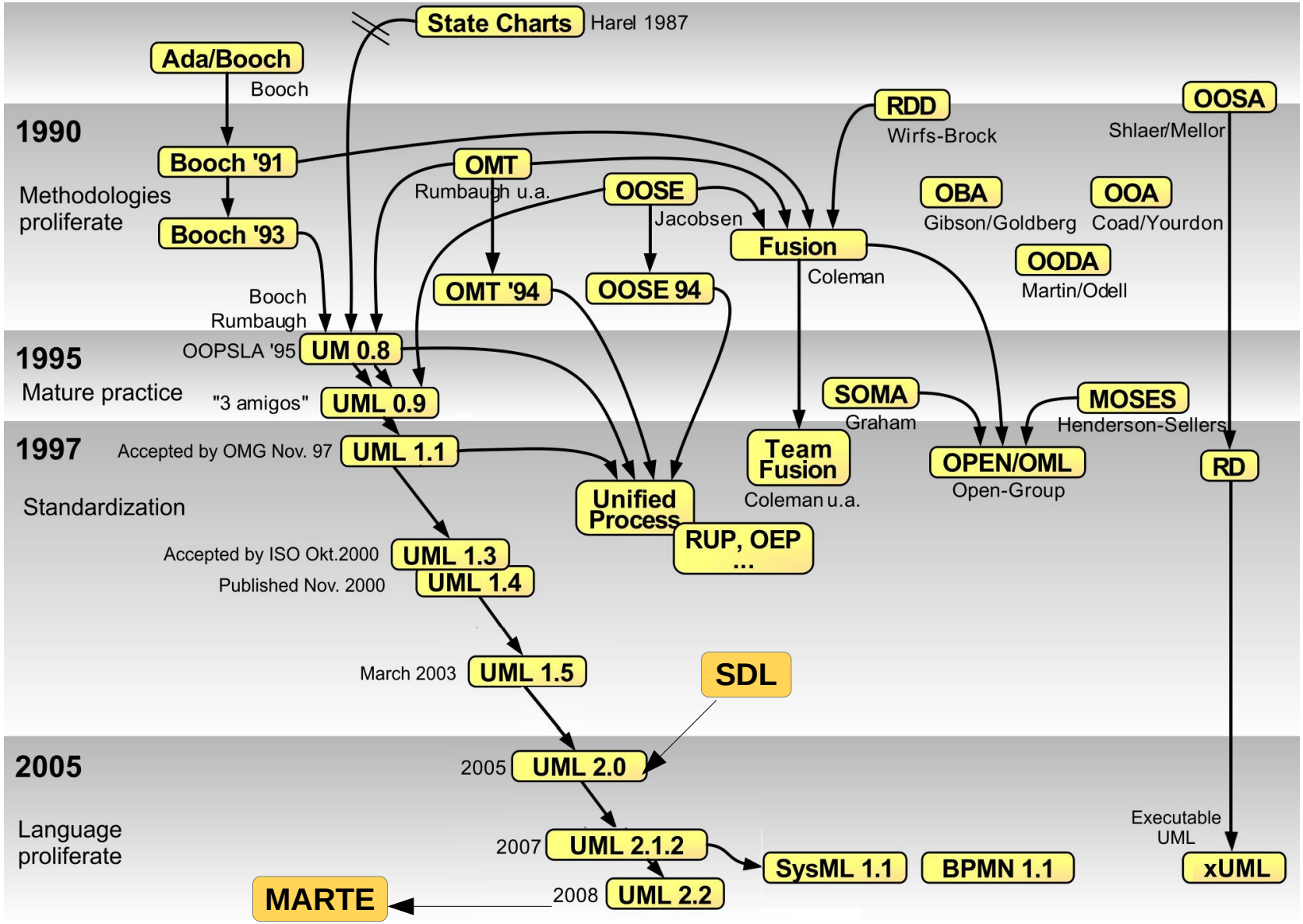
C'est **une** des manières de classer les modèles

L'EXPLOSION ORIENTÉE OBJET

- Une cinquantaine de méthodes objet au début des années 90
  - Confusion, attentisme
- Consensus autour d'idées communes
  - Objets, classes, associations, sous-systèmes, cas d'utilisation
- Recherche d'un langage commun unique
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation

Nouvel objectif → nouvelles reifications

CONSENSUS SUR LES BONS CONCEPTS



LES TROIS *AMIGOS*

Notation unifiée UML

- Basée sur les méthodes de BOOCH, OMT et OOSE

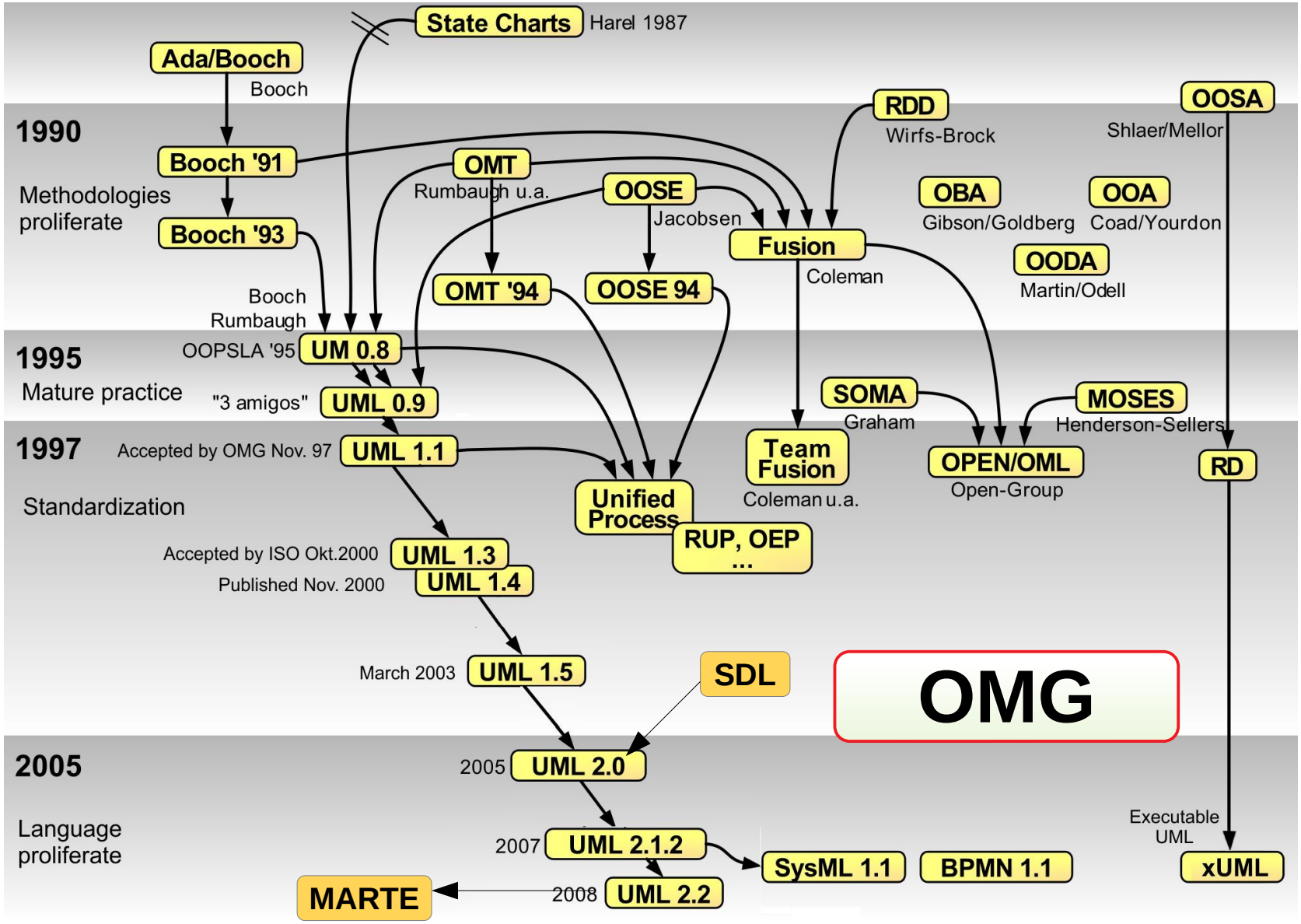


Grady Booch James Rumbaugh Ivar Jacobson

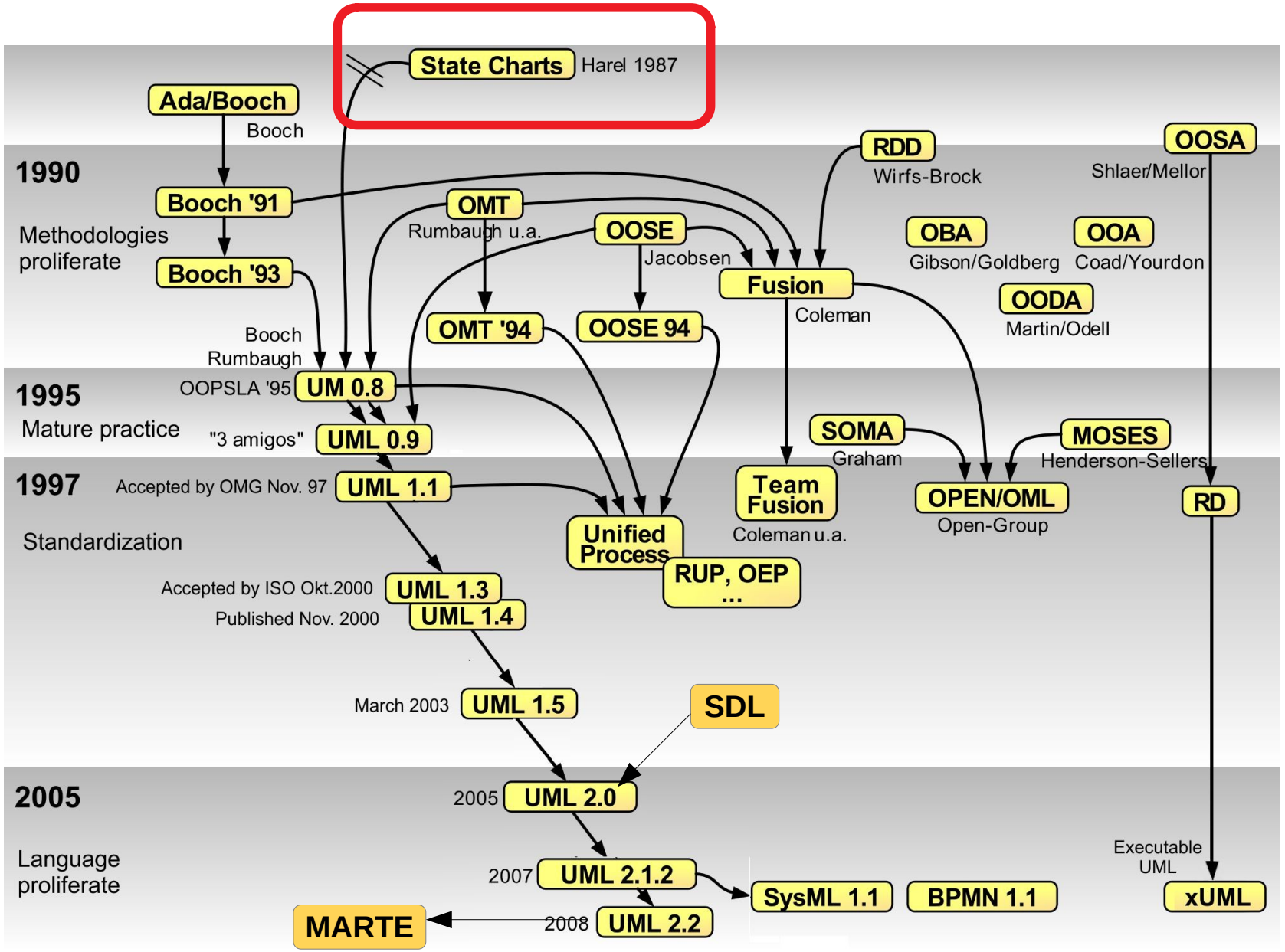
- Influencée par les bonnes idées des autres méthodes
- Mûrie par le travail en commun

Petite histoire d'UML Pierre-Alain Muller 9

CONSENSUS SUR LES BONS CONCEPTS



CONSENSUS SUR LES BONS CONCEPTS



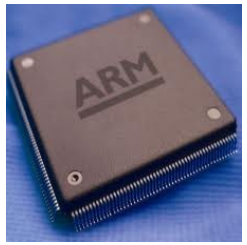
CONSENSUS SUR LES BONS CONCEPTS

Avec des objectifs pas nécessairement consensuels

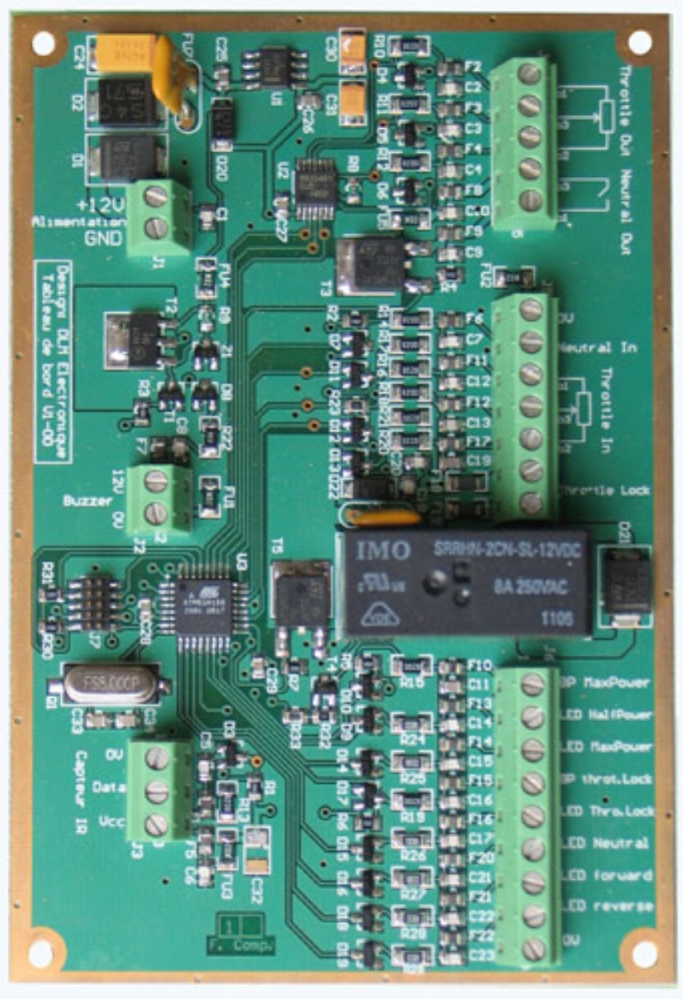
UML s'est éloigné de la nécessité de pouvoir compiler le modèle vers du code exécutable

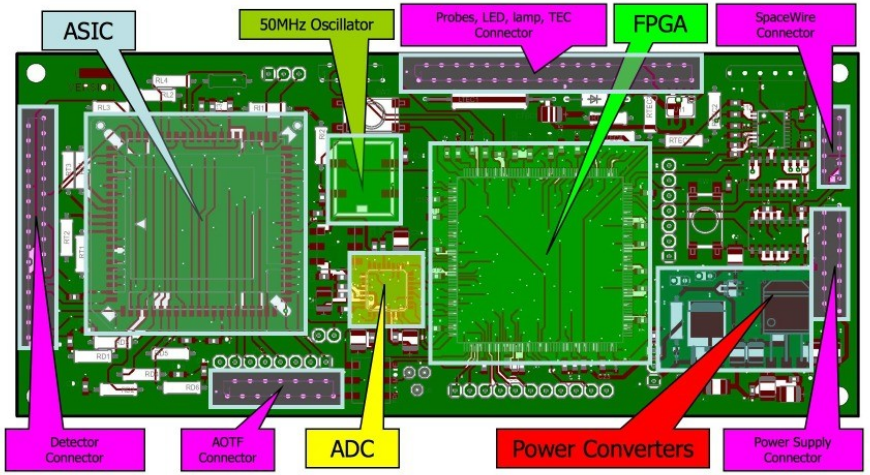
```
print '    %s [label="%s' % (nodename, label),
if isinstance(ast[1], str):
   if ast[1].strip():
      print '= %s"];' % ast[1]
   else:
      print '"]'
else:
   print '"];'
   children = []
   for n, child in enumerate(ast[1:]):
      children.append(dotwrite(child))
   print '    %s -> {' % nodename,
   for name in children:
      print '%s' % name,
```

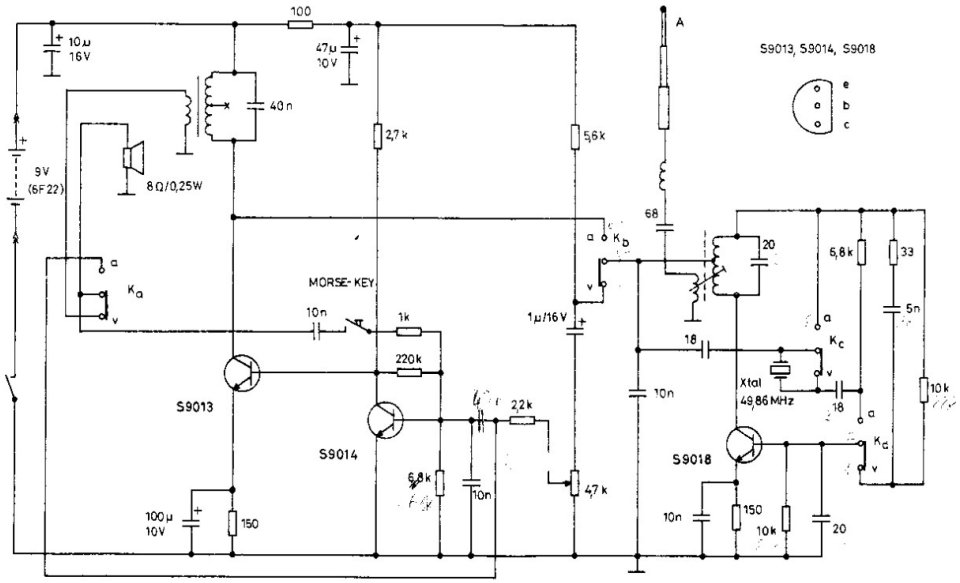
Automatic generation / compilation



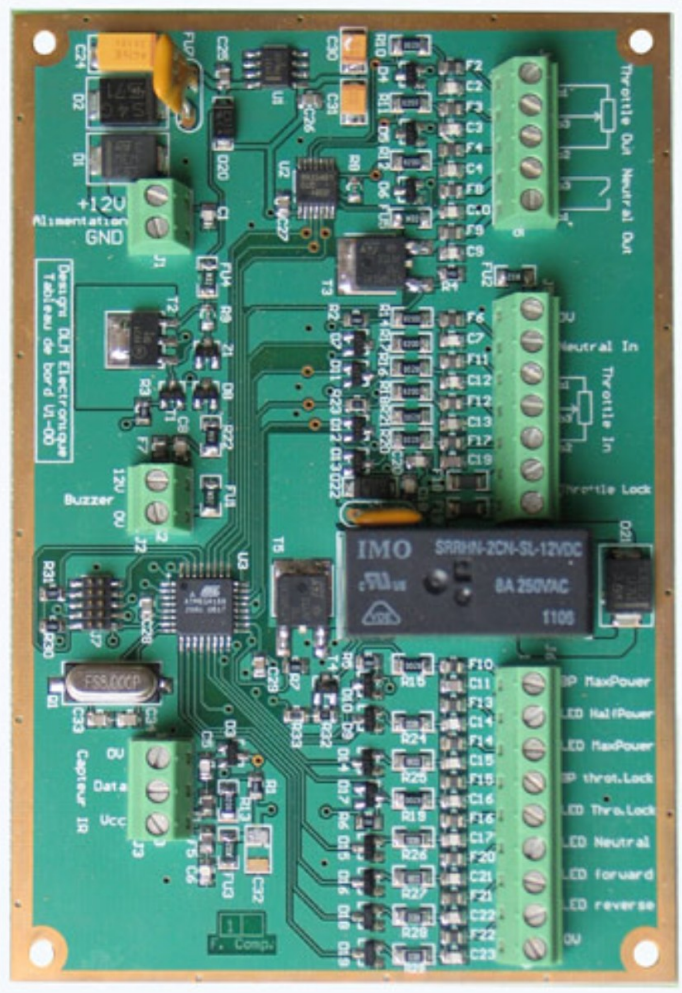
L'ingénierie dirigée par les modèles...

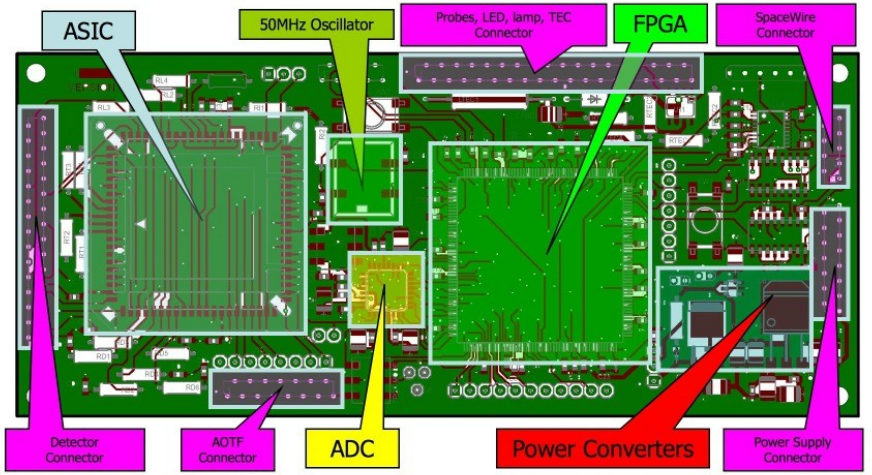


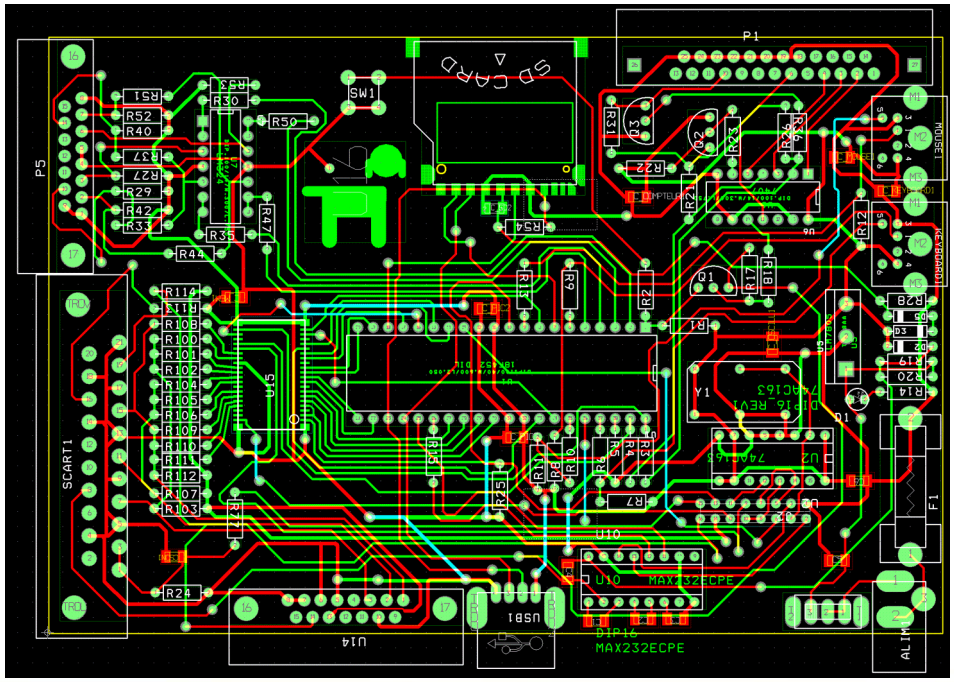




L'ingénierie dirigée par les modèles...







Languages dédiés et modèles

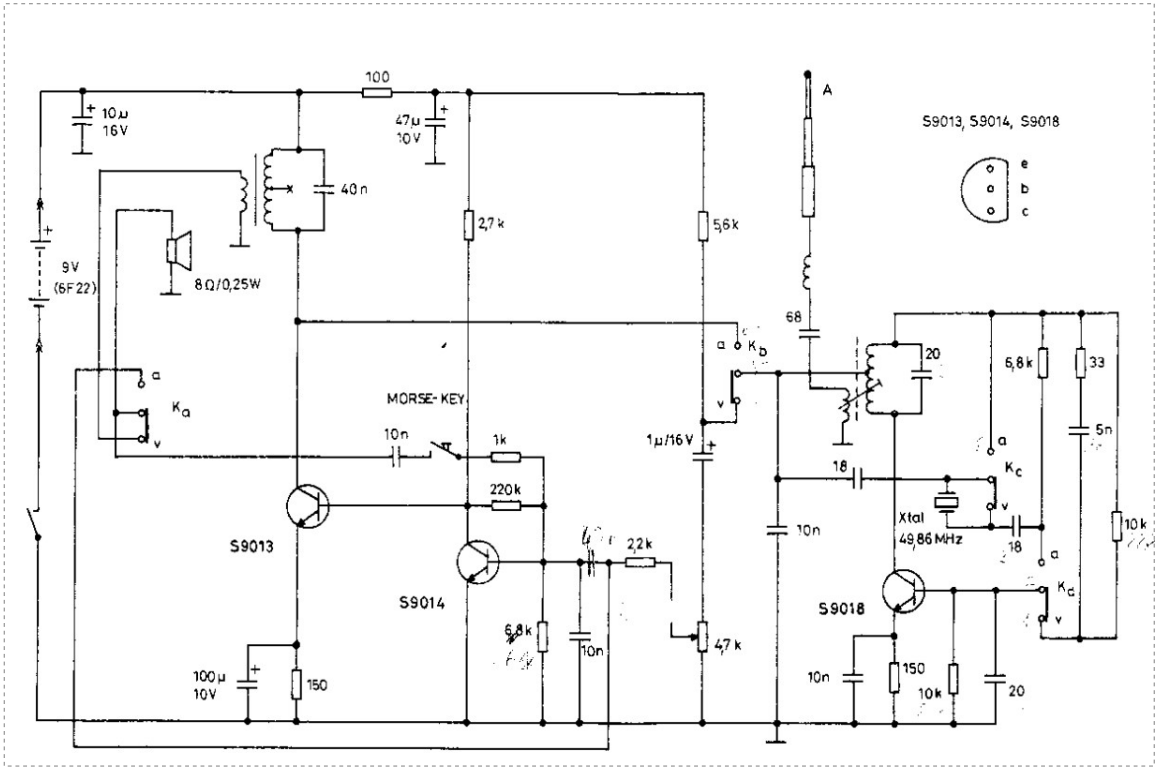
- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer...

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer...entre personnes comprenant le modèle.

Languages dédiés et modèles

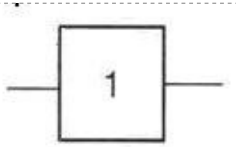
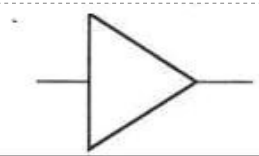
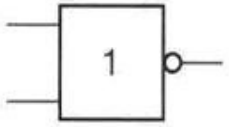
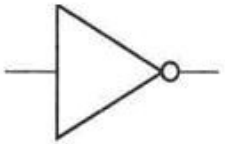
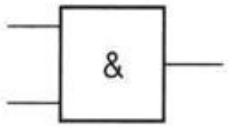
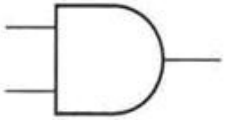
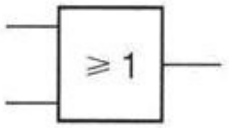
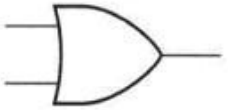
- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.



Tous les électroniciens !?

Languages dédiés et modèles

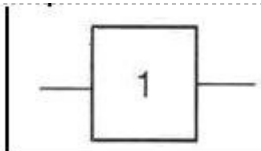
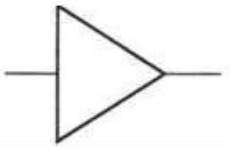
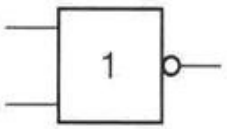
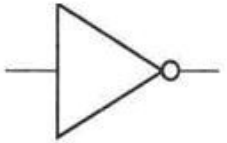
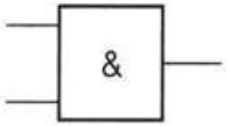
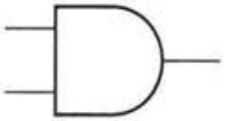
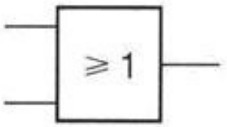
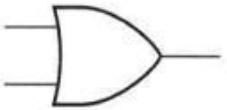
- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.

Porte OUI (YES)			entrée sortie 0 0 1 1
Porte NON (NO)			entrée sortie 0 1 1 0
Porte ET (AND)			entrées sortie 0 0 0 0 1 0 1 0 0 1 1 1
Porte OU (OR)			entrées sortie 0 0 0 0 1 1 1 0 1 1 1 1

Deux notations pour chaque porte ?

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.

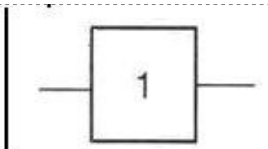
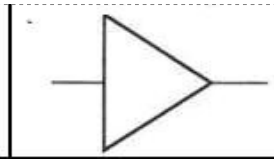
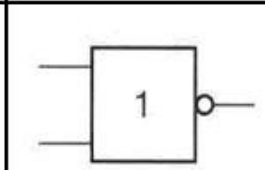
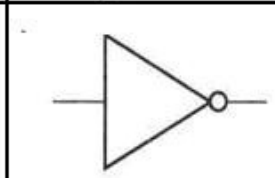
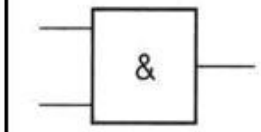
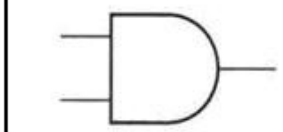
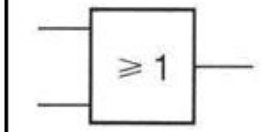
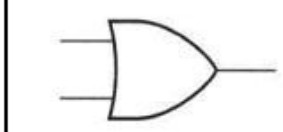
Porte OUI (YES)			entrée sortie 0 0 1 1
Porte NON (NO)			entrée sortie 0 1 1 0
Porte ET (AND)			entrées sortie 0 0 0 0 1 0 1 0 0 1 1 1
Porte OU (OR)			entrées sortie 0 0 0 0 1 1 1 0 1 1 1 1

Deux notations pour chaque porte ?!

Mais la sémantique est donnée...

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.

Porte OUI (YES)			entrée sortie 0 0 1 1
Porte NON (NO)			entrée sortie 0 1 1 0
Porte ET (AND)			entrées sortie 0 0 0 0 1 0 1 0 0 1 1 1
Porte OU (OR)			entrées sortie 0 0 0 0 1 1 1 0 1 1 1 1

Deux notations pour chaque porte ?!

Mais la sémantique est donnée...

→ toutes les personnes comprenant la logique booléenne ?!

Rapide notion de sémantique

> *La sémantique est tout ce qui concerne l'étude scientifique du sens des unités linguistiques et de leurs combinaisons.*
>
> *Dictionnaire Larousse 1994*

- sémantique statique
  - Contraintes s'assurant que la syntaxe est utilisée correctement
- sémantique comportementale
  - Dans quel ordre les éléments syntaxiques sont interprétés et quel est leur «*effet*» (seul et combinés)

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.
  - Les personnes connaissant / ayant accès à la sémantique des éléments du modèle

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.
  - Les personnes connaissant / ayant accès à la sémantique des éléments du modèle
- ...permet d'encapsuler du savoir / des bonnes pratiques et de les placer dans l'outillage

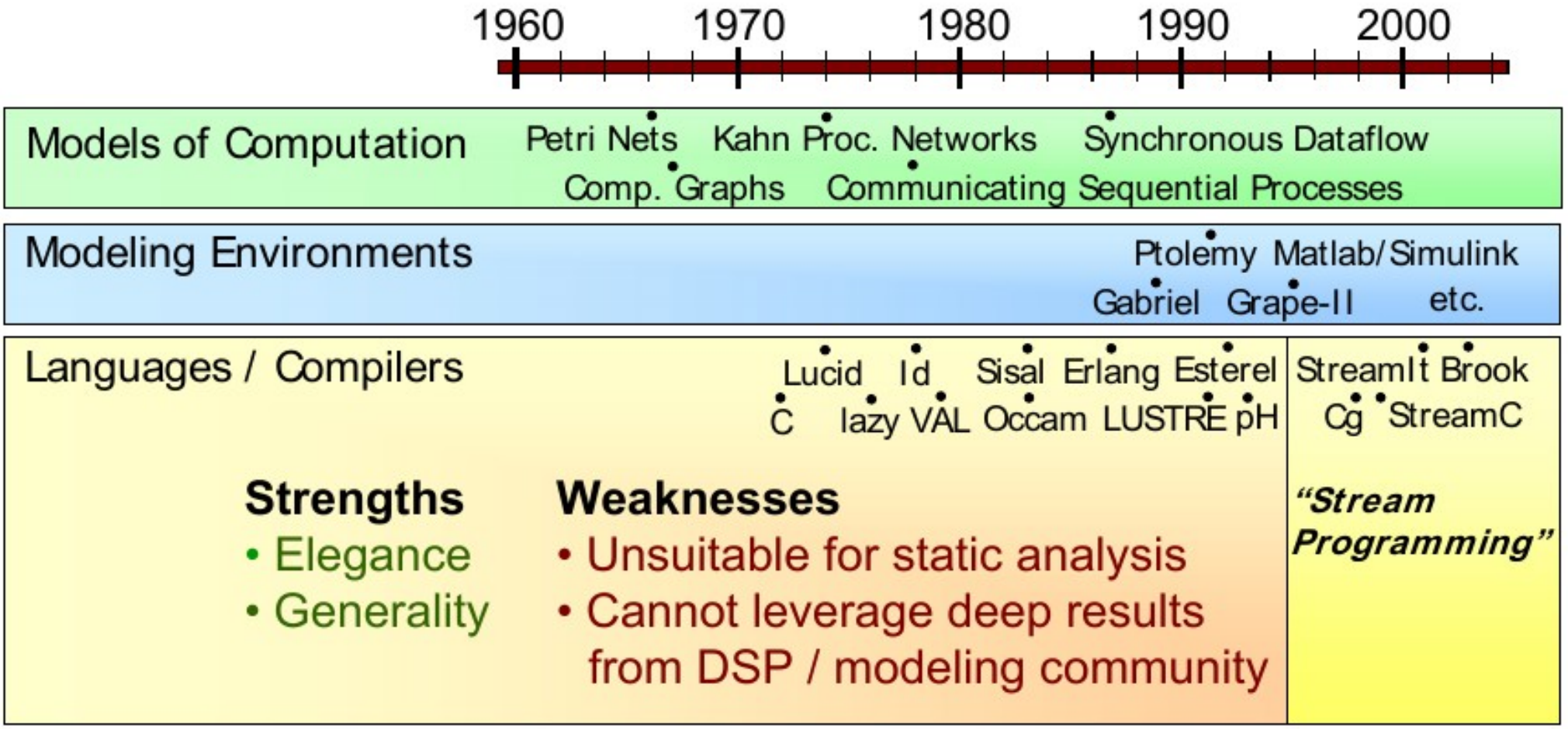
Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.
  - Les personnes connaissant / ayant accès à la sémantique des éléments du modèle
- ...permet d'encapsuler du savoir / des bonnes pratiques et de les placer dans l'outillage
  - ...permet d'analyser / d'automatiser les actions récurrentes d'un domaine
    - Si l'abstraction est correcte vis-à-vis de l'analyse / automatisation !

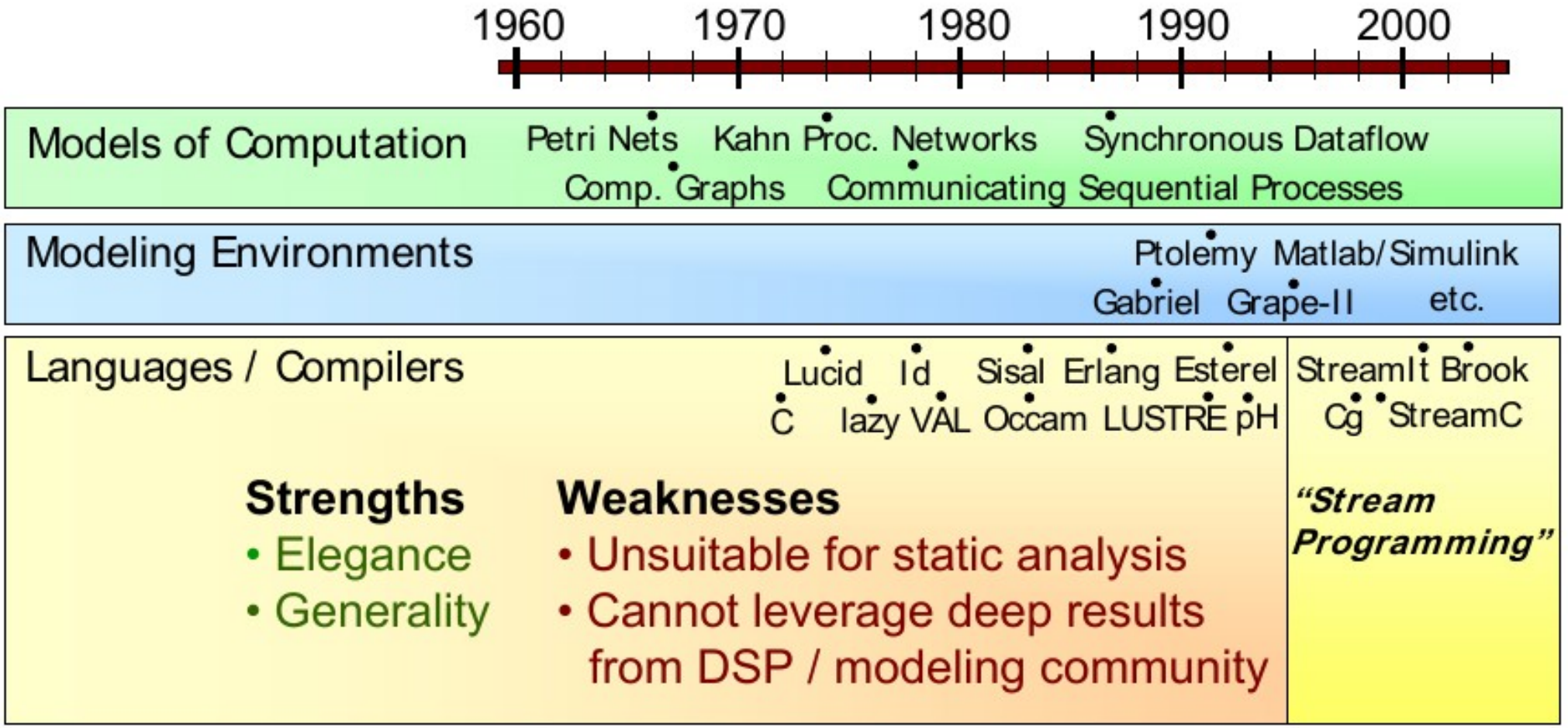
Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.
  - Les personnes connaissant / ayant accès à la sémantique des éléments du modèle
- ...permet d'encapsuler du savoir / des bonnes pratiques et de les placer dans l'outillage
  - ...permet d'analyser / d'automatiser les actions récurrentes d'un domaine
    - Si l'abstraction est correcte vis-à-vis de l'analyse / automatisation !

Other kinds of models

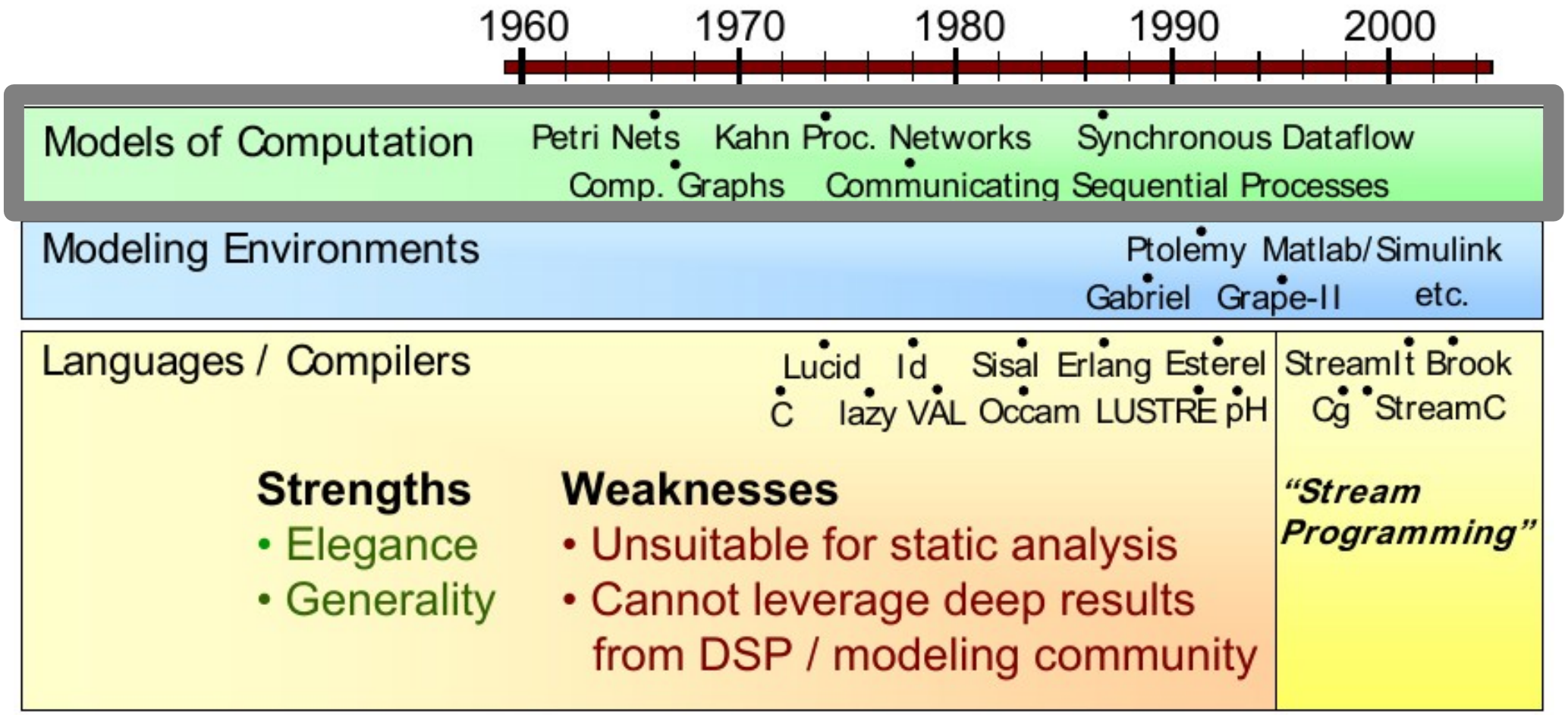


Other kinds of models



Modèles d'abstraction forte, plus proche de l'analyse du problème que de la description de la solution
→ ***réel impact sur l'informatique... par le côté théorique***

Other kinds of models



Modèles d'abstraction forte, plus proche de l'analyse du problème que de la description de la solution
→ ***réel impact sur l'informatique... par le côté théorique***

Et par où commencer ?

→ le choix d'un *langage workbench*

Language Workbench

- C'est un outil permettant de faciliter la définition de nouveaux langages
- Cache une partie de la compléxitée inhérante, par exemple, au parsing ou à la représentation graphique
- N'est pas utilisé par le end user.
- Fourni plus ou moins de services

Language Workbench

- Exemples
  - MPS : https://www.jetbrains.com/mps/concepts/domain-specific-languages/
  - spoofax, rascal, racket,
  - Meta-edit+ http://www.metacase.com/webcasts/DSM_Definition.html
  - EMF ad eclipse modeling : https://www.eclipse.org/modeling/
    - Gemoc Studio :
      - http://gemoc.org/studio.html
      - http://gemoc.org/gallery.html

Language Workbench

- Exemples
  - MPS : https://www.jetbrains.com/mps/concepts/domain-specific-languages/
  - spoofax, rascal, racket,
  - Meta-edit+ http://www.metacase.com/webcasts/DSM_Definition.html
  - EMF ad eclipse modeling : https://www.eclipse.org/modeling/
    - **Gemoc Studio :**
      - http://gemoc.org/studio.html
      - http://gemoc.org/gallery.html

GEMOC Language Workbench

- How to start
  - First, learning metamodelisation et syntax stuff with tutorials
    - https://www.eclipse.org/modeling/emf/ && https://eclipsesource.com/blogs/tutorials/emf-tutorial/ && https://mbaron.developpez.com/tutoriels/eclipse/emf/creation-instanciation-modeles/
    - https://www.eclipse.org/Xtext/index.html
    - https://www.eclipse.org/sirius/ && https://www.eclipse.org/sirius/getstarted.html
  - Second, play with **a** behavioural semantics:)