The DEPS Project

(DEsign Problem Specification)

GDR SEEDS

GT Systèmes complexes

January 17th, 2019

Laurent ZIMMER
Dassault Aviation

Pierre-Alain YVARS
SupMéca-Quartz

Outlines

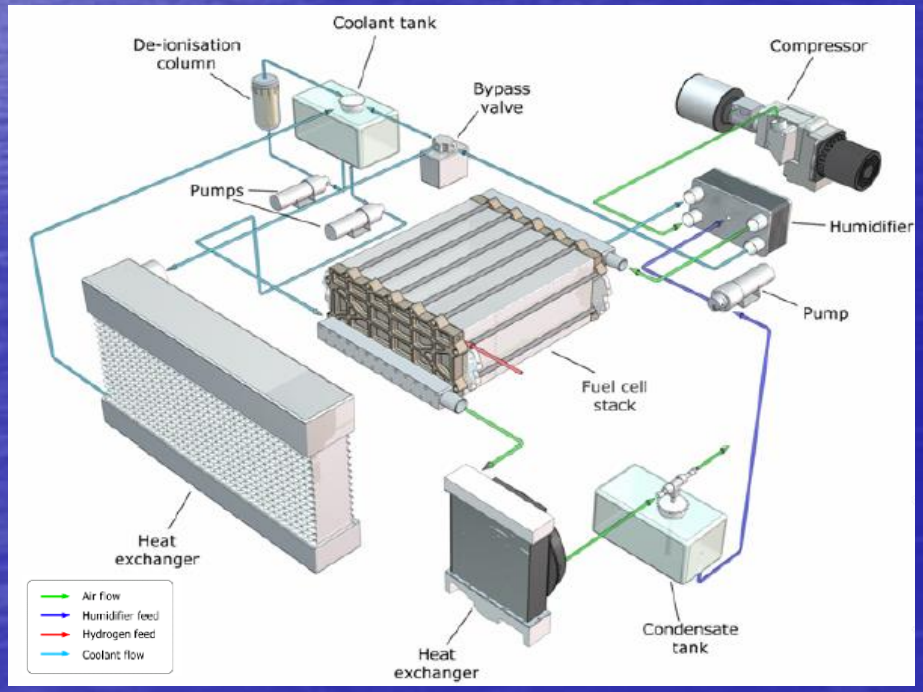
- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

Context

A system is a construct or collection of different elements that together produces results not obtainable by the elements alone (INCOSE Definition).

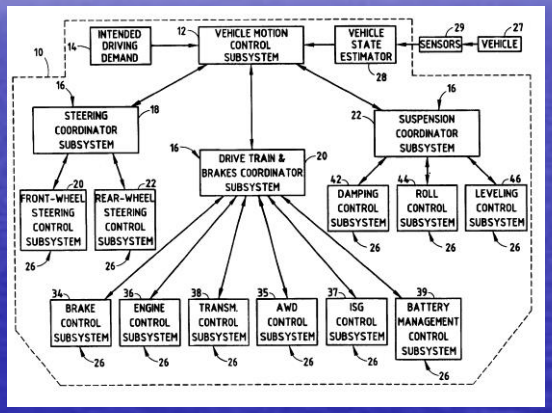
…

The value added by the system as a whole beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is how they are interconnected (Rechtin, 2000)



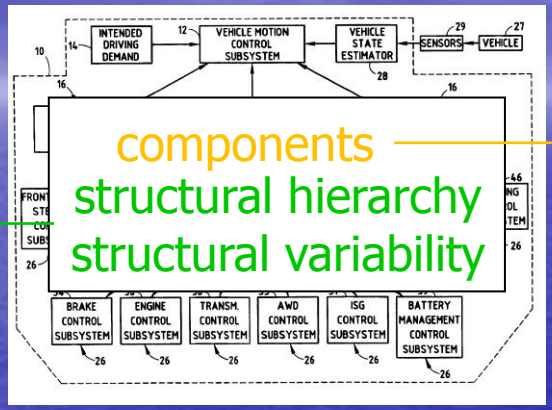
Some system design characteristics

functions
functional hierarchy
+
functional variability



components
structural hierarchy
+
structural variability

Some system design characteristics



World of Material architecture

- Graph representation
- Selection of values
- Selection of (on the Shelf) components
- Compatibility constraints

(One) Material/ (Many) Behavioral World

- Technical Requirements
- Physical laws
- Technological constraints
- Standards and regulation Rules

A Mixed Calculus world !

Some system design characteristics

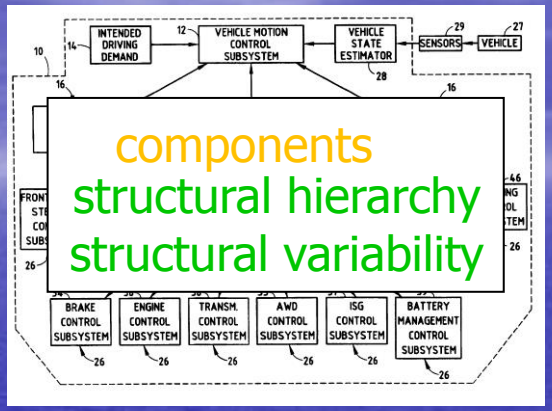
World of functional architecture

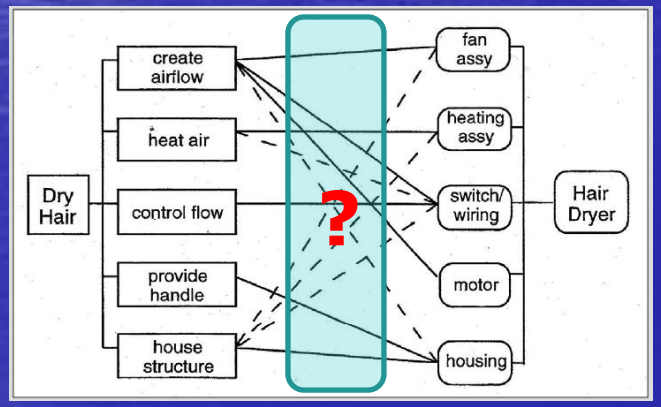
- Graph representation
- Duplication of functions
- Selection of functions
- Relations between functions

Functional World

- Functional requirements
  Time deadline, Power supply ..
- Non functional requirements
  Safety, Security, Sustainability ..

Moreover .. There are missing links

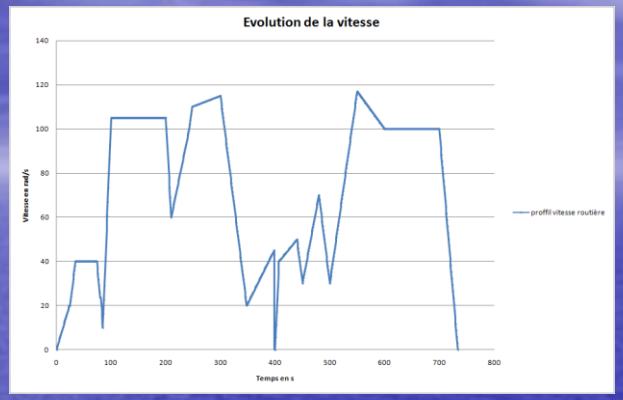


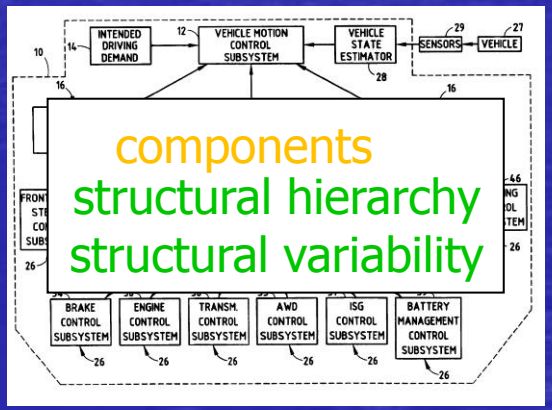


function – structure **allocations**

Moreover .. There are a missing links

use-case – functioning modes **associations**



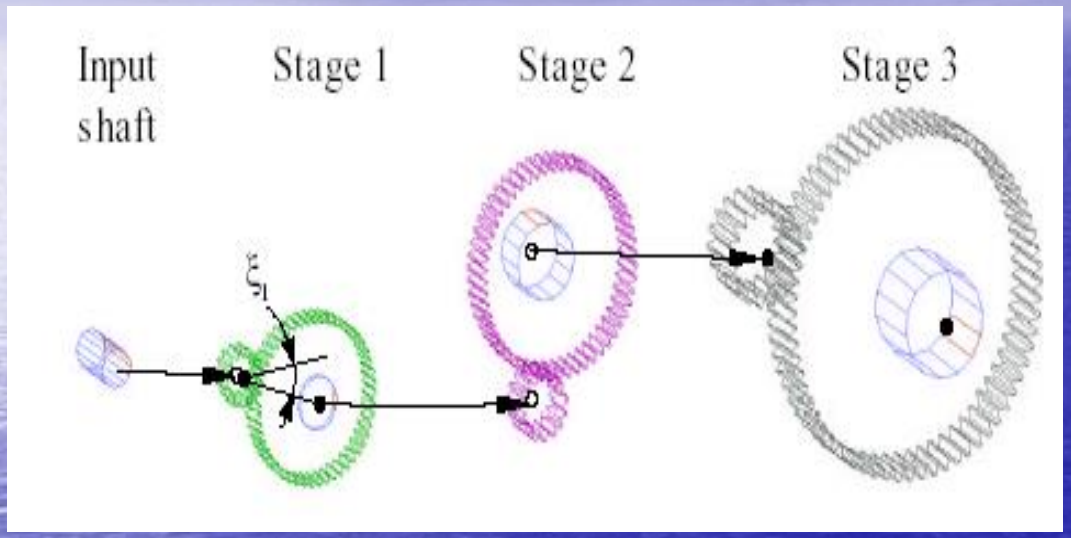


A taxonomy of Design Problems

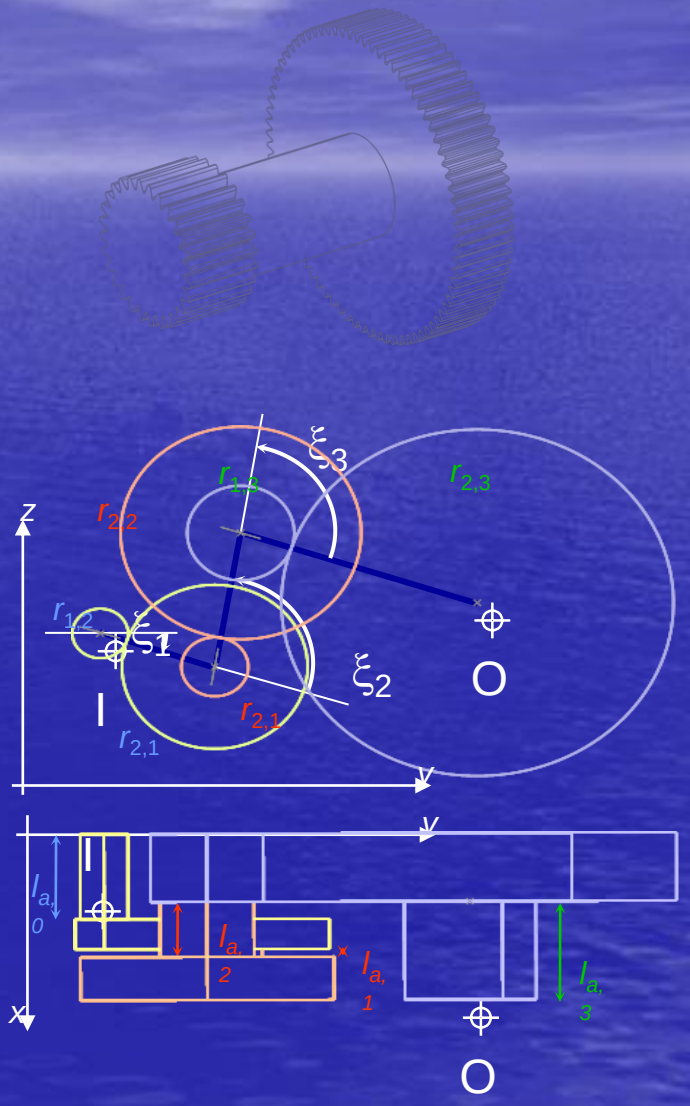
- Sizing
  - Setting all the design variables for all the given subsystems and components in order to satisfy the requirements
- Configuration
  - Constructing a system with the right number and types of predefined components. Compatibility constraints limit the number of possible combinations of the components.
- Allocation
  - allocating resources to different types of elements at different level (functions, tasks, software, …)
- Architectural design
  - Organising a system in subsystems and components in order to satisfy the requirements

Sizing Problem

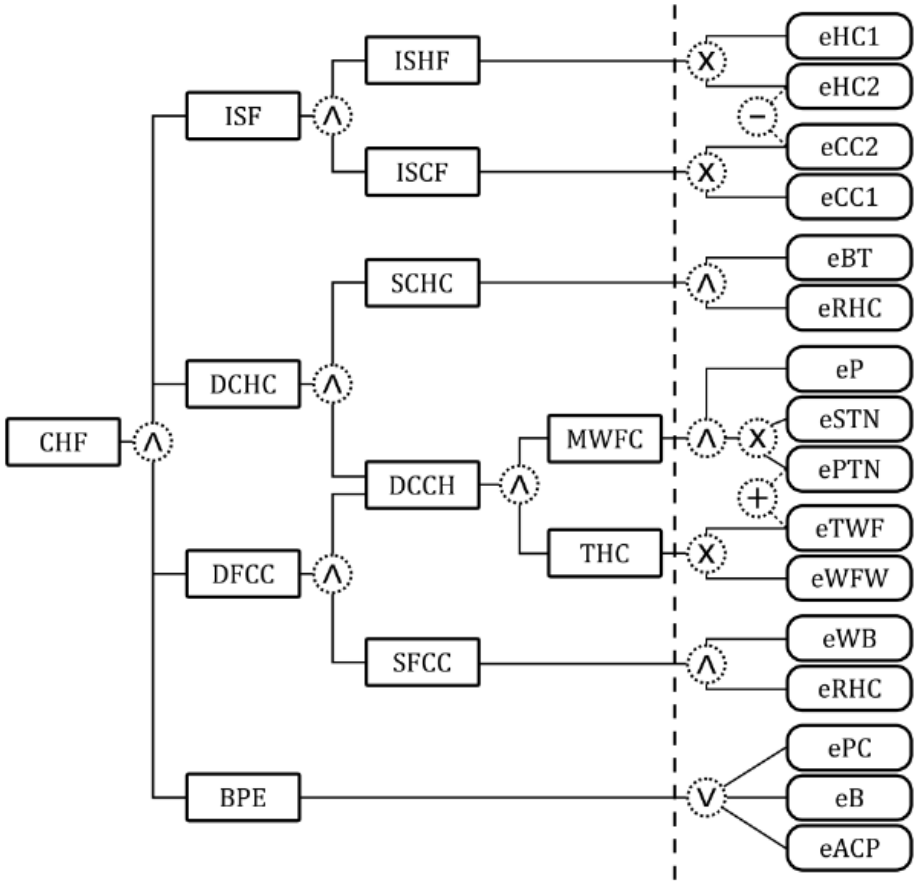
A three stages power transmission system



	Angle	Module	Tooth number		Face width	Shaft radius	Shaft length
			Pinion	Wheel			
Input Shaft						$r_{a,0}$	$l_{a,0}$
Stage 1	ξ_1	m_1	$Z_{1,1}$	$Z_{2,1}$	b_1	$r_{a,1}$	$l_{a,1}$
Stage 2	ξ_2	m_2	$Z_{1,2}$	$Z_{2,2}$	b_2	$r_{a,2}$	$l_{a,2}$
Stage 3	ξ_3	m_3	$Z_{1,3}$	$Z_{2,3}$	b_3	$r_{a,3}$	$l_{a,3}$



Configuration Problem

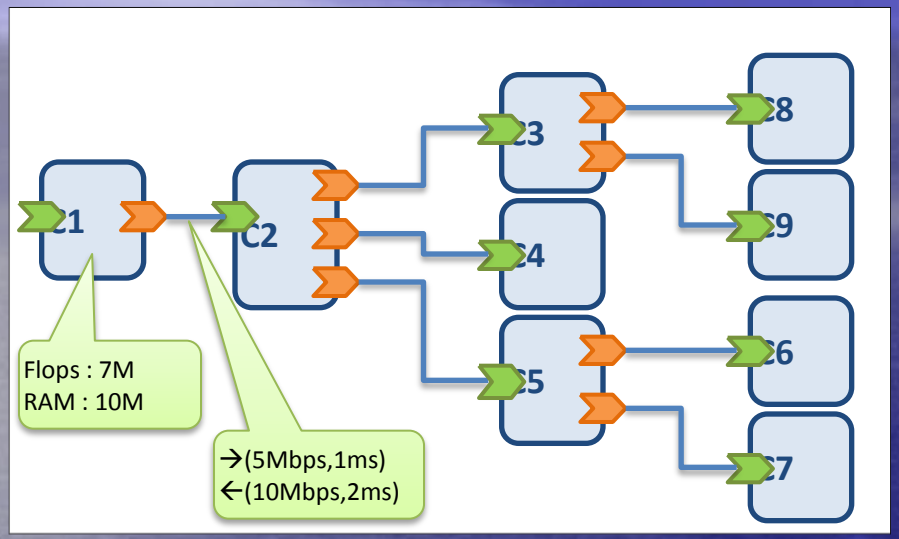


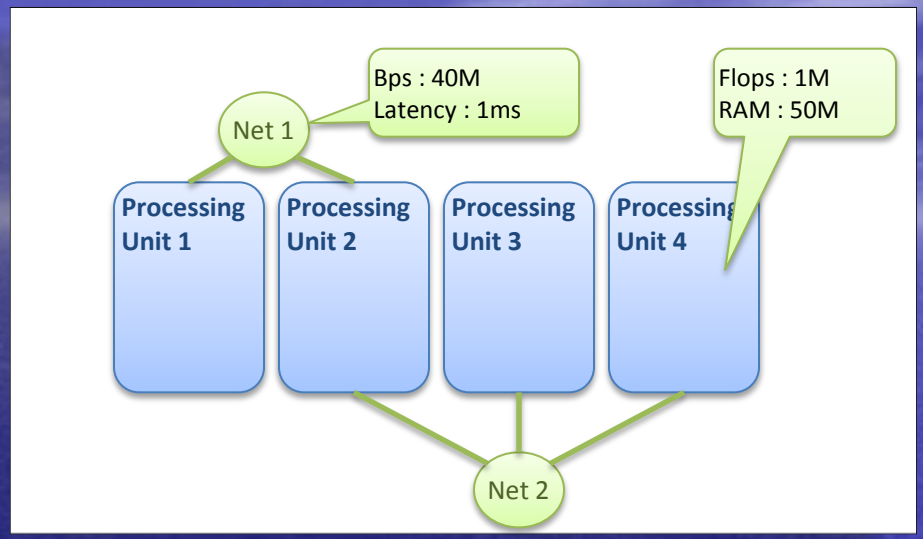
Cool/heater system configuration problem
(Yvars, Duhau, 2011)

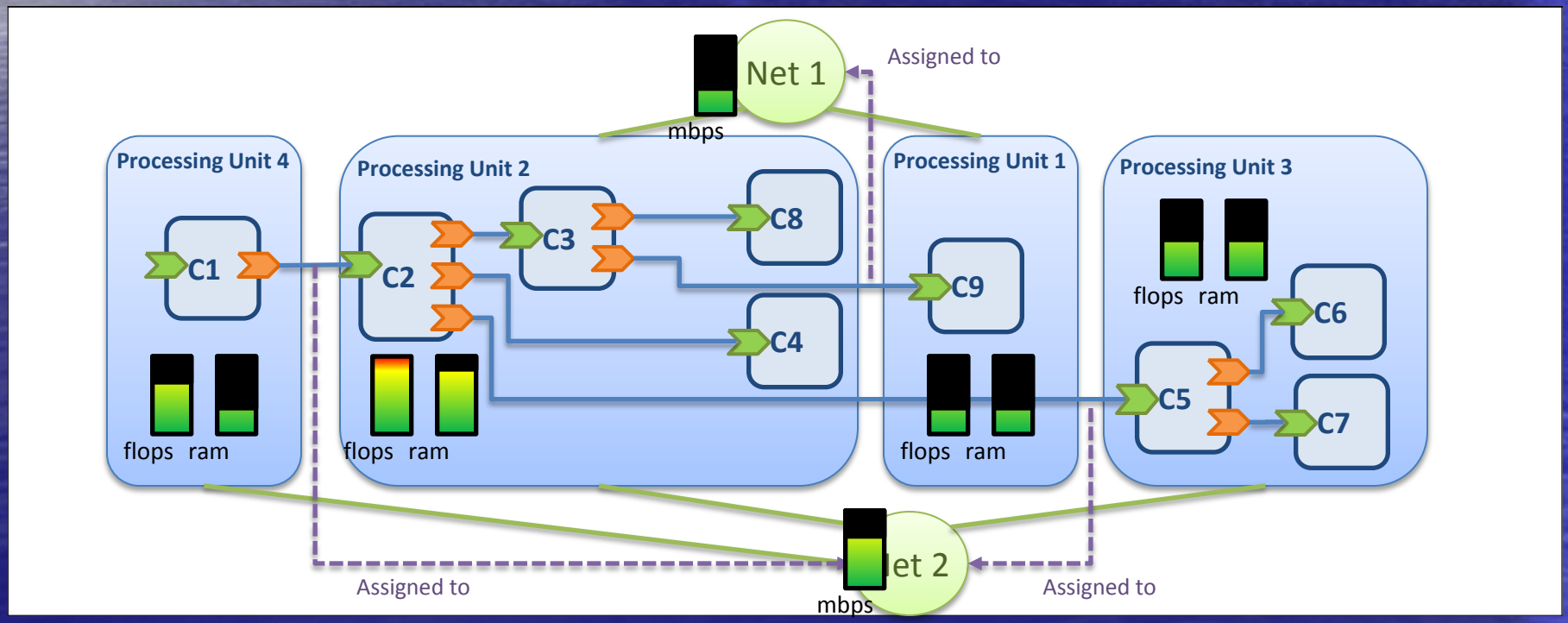
CHF:	{1}	*Cool and Heat Food*
ISF:	{1}	*Isolate and Stock Food*
DCHC:	{1}	*Diffuse Calories to the Heating Compartment*
DFCC:	{1}	*Diffuse Frigories to the Cooling Compartment*
BPE:	{1}	*Be Powered by Electricity*
ISHF:	{1}	*Isolate and Stock Hot Food*
ISCF:	{1}	*Isolate and Stock Cold Food*
SCHC:	{1}	*Stock Calories in the Heating Compartment*
DCCH:	{1}	*Diffuse Calories from the Cooling compartment to the Heating compartment*
SFCC:	{1}	*Stock Frigories in the Cooling Compartment*
MWFC:	{1}	*Make the Working Fluid Circulate*
THC:	{1}	*Transfer Heat through Convection*
eHC1:	{0, 1}	*Heating Compartment of a capacity of 1*
eHC2:	{0, 1}	*Heating Compartment of a capacity of 2*
eCC1:	{0, 1}	*Cooling Compartment of a capacity of 1*
eCC2:	{0, 1}	*Cooling Compartment of a capacity of 2*
eBT:	{1}	*Buffer Tank*
eRHC:	{1}	*Radiant Heating Component*
eWFW:	{0, 1}	*Working Fluid: Water*
eTWF:	{0, 1}	*Toxic Working Fluid*
eSTN:	{0, 1}	*Standard Tube Network*
ePTN:	{0, 1}	*Processed Tube Network*
eP:	{1}	*Pump*
eWB:	{1}	*Water Block*
ePC:	{0, 1}	*Photovoltaic Cells*
eB:	{0, 1}	*Battery*
eACP:	{0, 1}	*AC Power*

Cool/Heater Nomenclature

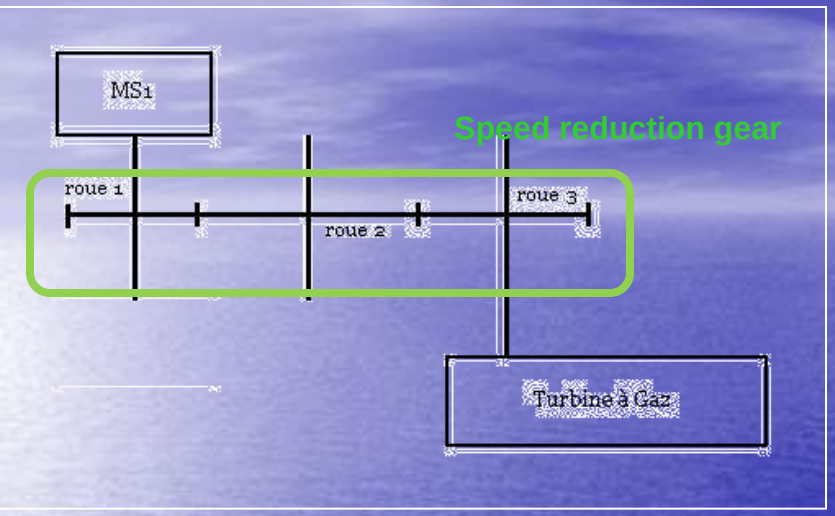
Allocation Problem

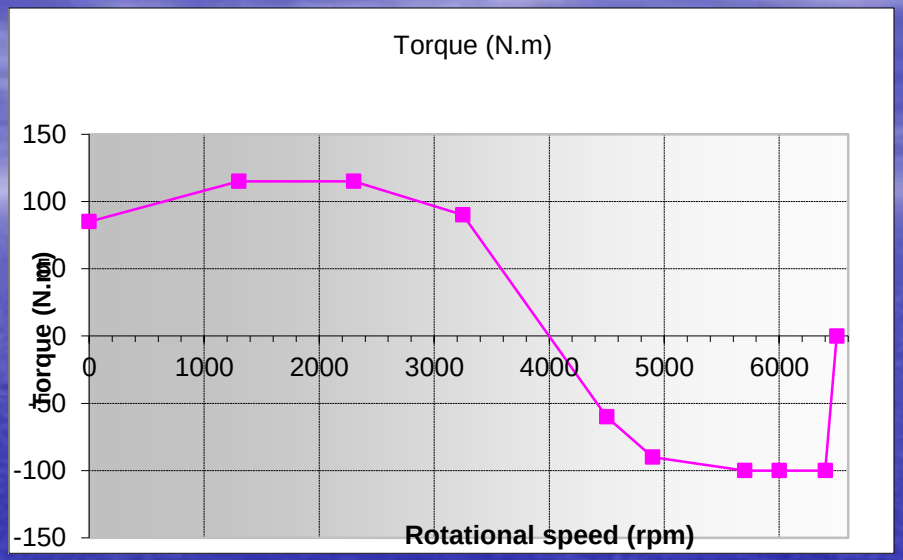


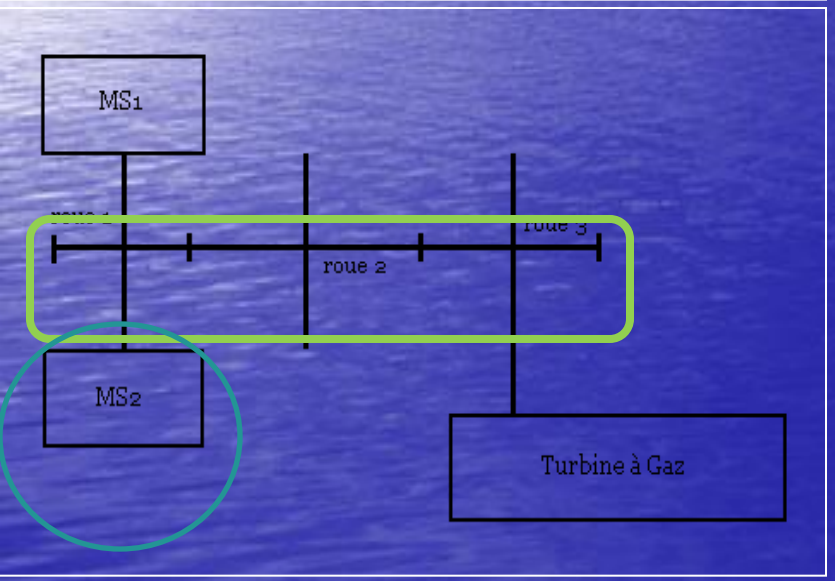


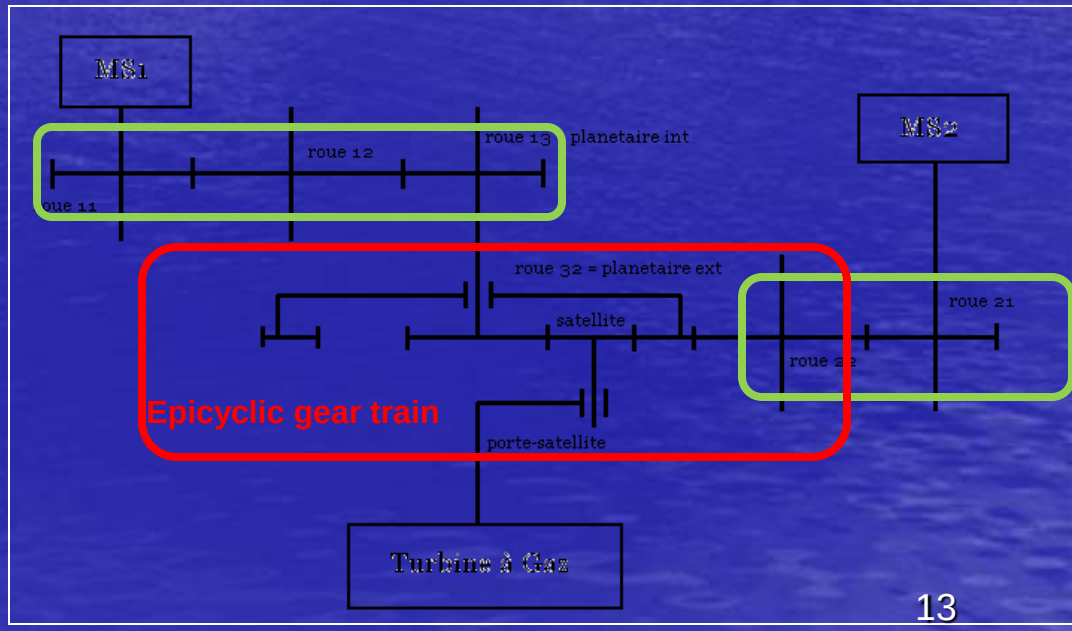


Architectural Design Problem





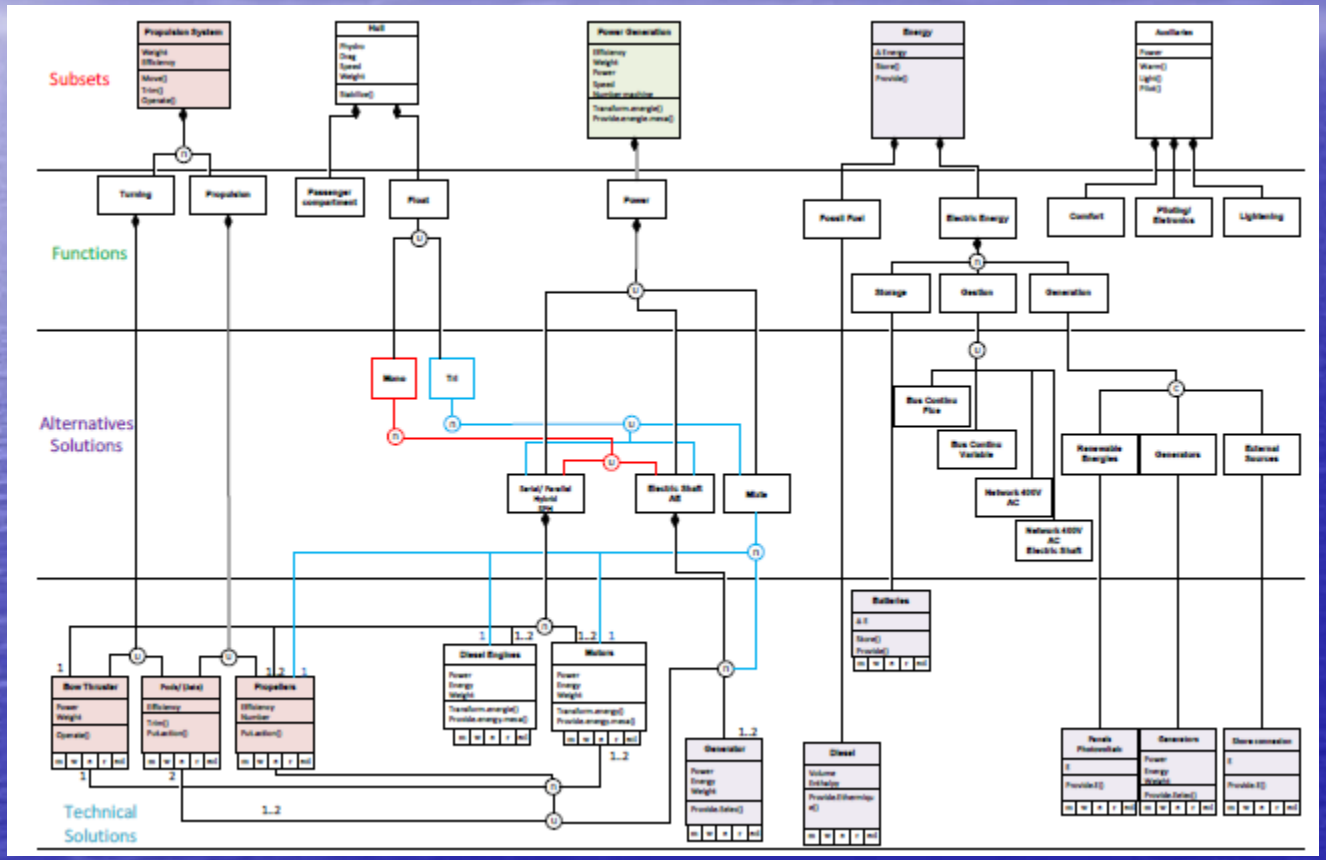




Gas turbine starter system
Dassault Aviation Engineering study
Zimmer, Yvars, 2011

Design problem +++

Hybrid passenger ferry



(Tchertchian, Yvars, Millet, 2012)

What is common to Design Problems ?

- Sizing
  - Some Design Parameters are not fixed
- Configuration
  - Number and type of some components are not fixed
- Allocation
  - Resources required by some elements are not fixed
- Architectural design
  - A mixing of everything above

What is common to Design Problems ?

A System to be designed is sub-defined

Designing a System means completing a sub-defined system which conforms to Requirements

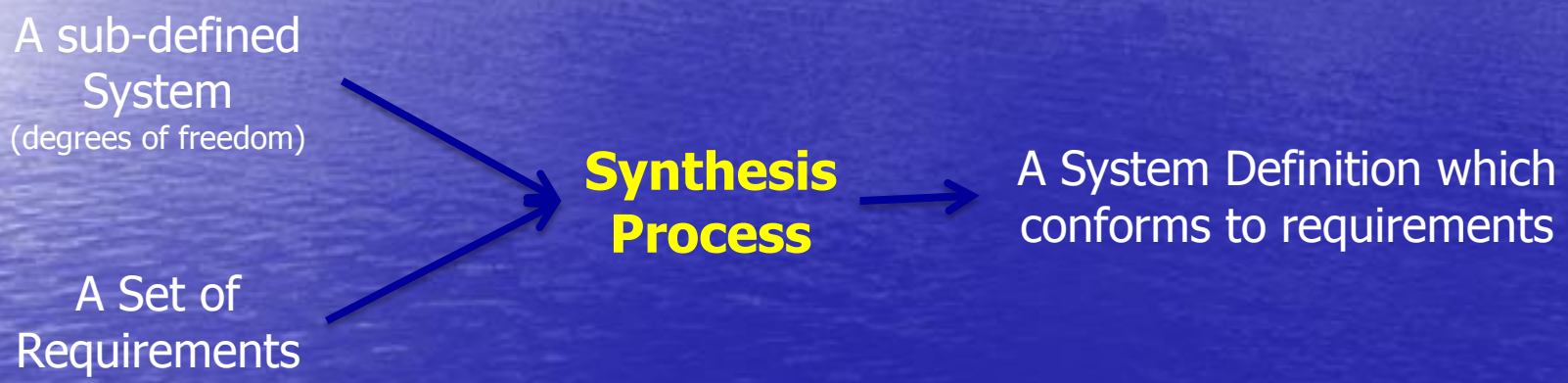
Our Manifesto

Solving a Design Problem

Is

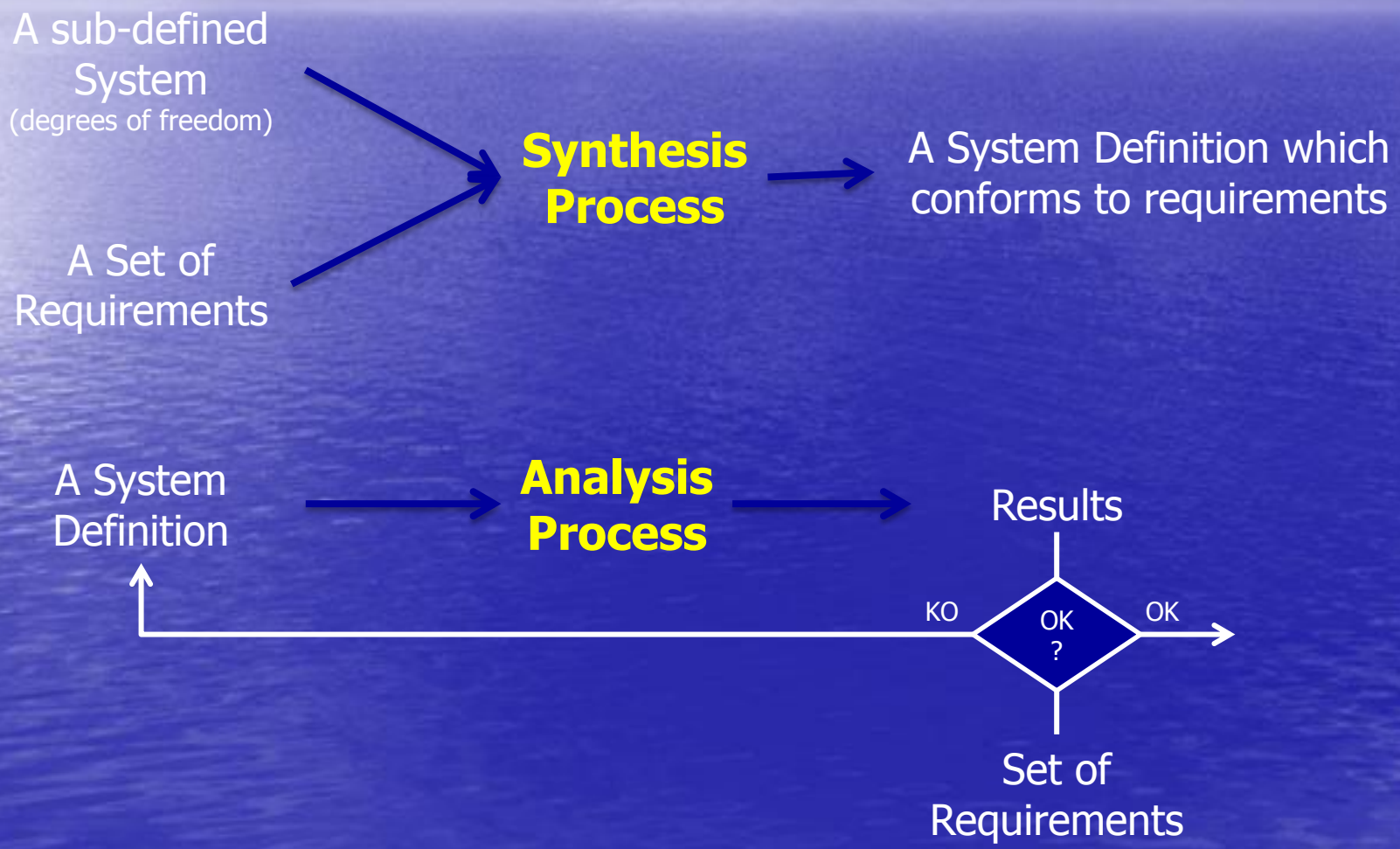
Completing a Sub-defined Model

A Design Problem is a matter of Synthesis

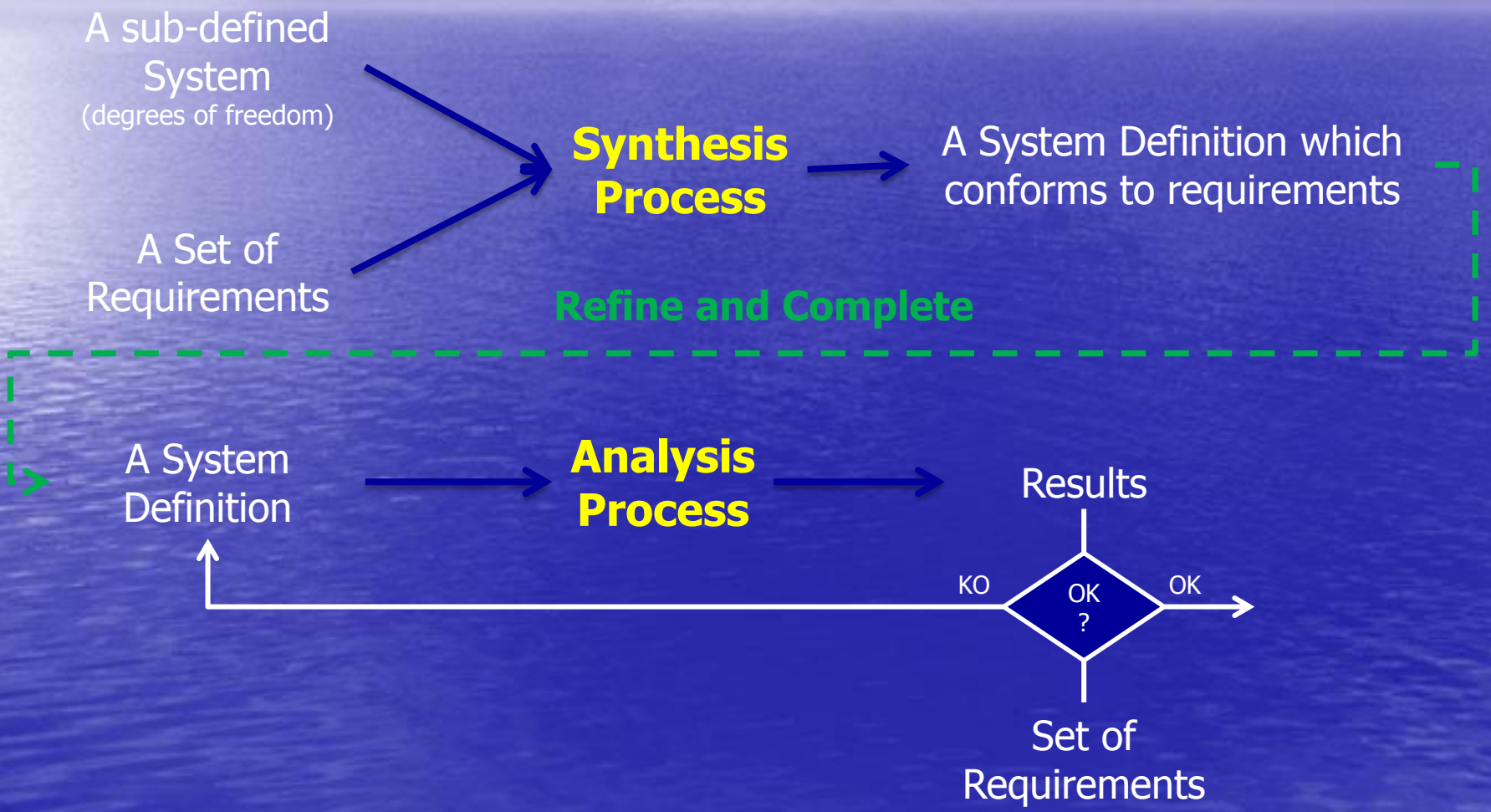


Synthesis isn't Analysis

Synthesis and Analysis are complementarity



Synthesis and Analysis are complementary



Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

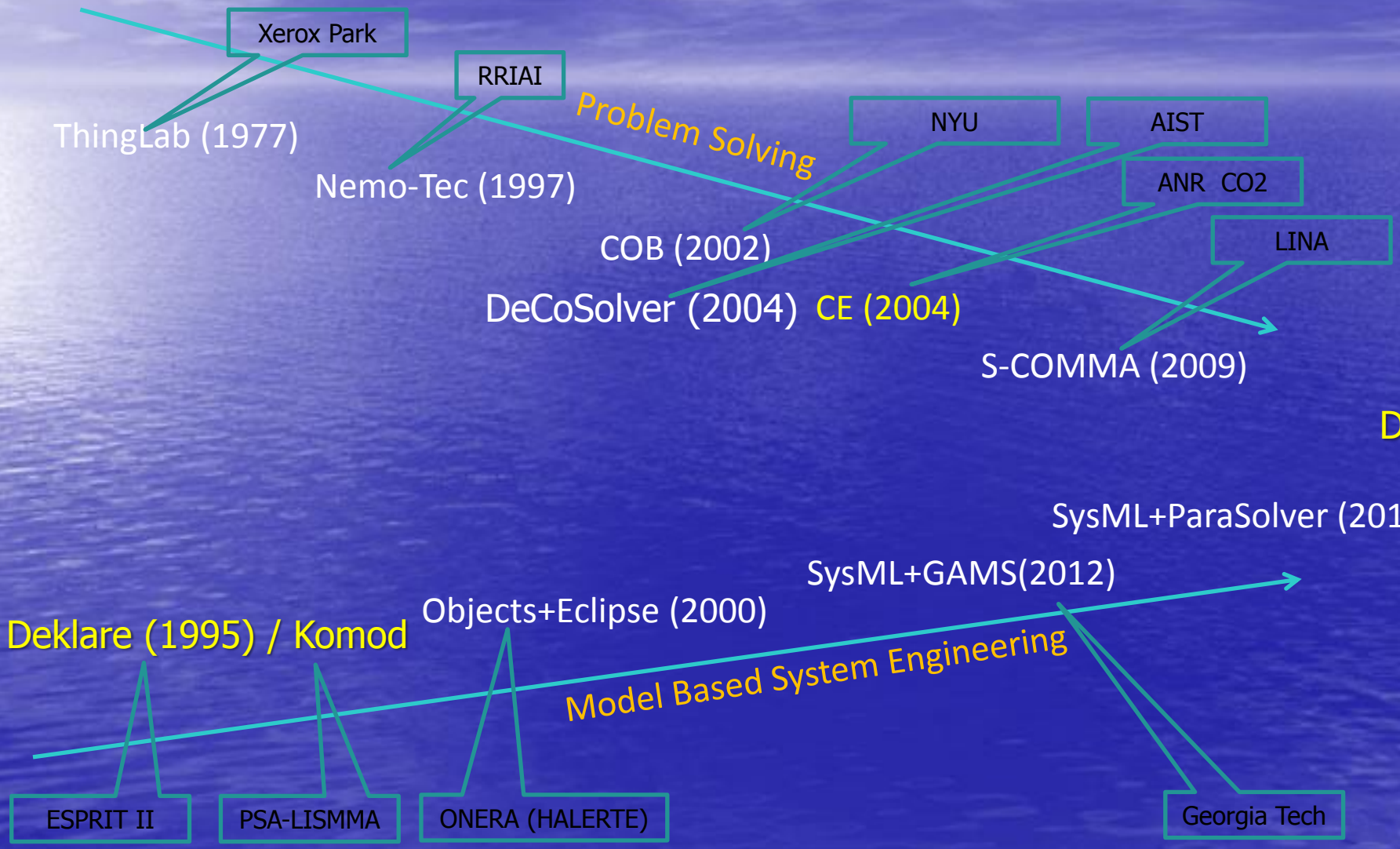
The DEPS Project

- Develop a formal modeling language for specifying System Design Problems
- Develop Problem Solving methods and tools

The DEPS Project

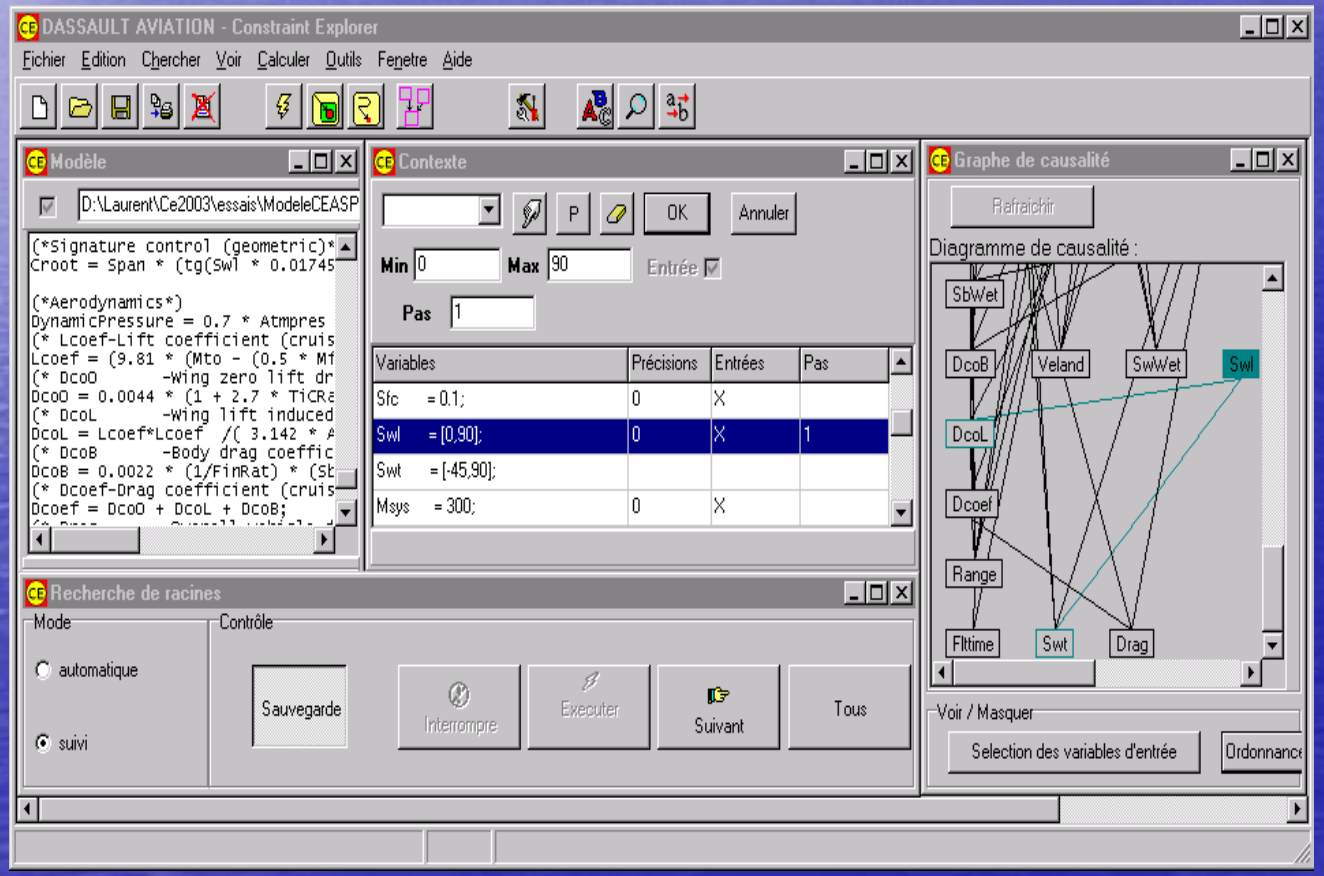
- Target: engineering, embedded, real-time, cyber-physical, software-intensive … Systems
- Design steps: preliminary design, architecture generation, system integration, system verification…

Background


Xerox Park
RRIAI
ThingLab (1977)
Nemo-Tec (1997)
Problem Solving
NYU
AIST
ANR CO2
LINA
COB (2002)
DeCoSolver (2004)
CE (2004)
S-COMMA (2009)
DEPS
SysML+ParaSolver (2013)
SysML+GAMS(2012)
Objects+Eclipse (2000)
Deklare (1995) / Komod
Model Based System Engineering
ESPRIT II
PSA-LISMMA
ONERA (HALERTE)
Georgia Tech

Constraint Explorer
(L. Zimmer et al. 2004)

- IDE
- Declarative language
- Interval Constraint Programming methods



- non linear equations and inequalities with discrete and continuous variables

Sizing of a 3 stages Power Transmission System (Yvars, Zimmer)

- 89 non linear equations
- 48 inequalities
- 1 optimization criterion

$$f_{Obj}(\mathbf{x}) = \pi.\sum_{s=0}^{s=3}\left(r_{a,s}{}^{2}.l_{a,s}\right) + \pi.\sum_{s=1}^{s=3}\left(b_s.m_s{}^{2}.\left(Z_{1,s}{}^{2} + Z_{2,s}{}^{2}\right)\right)$$

Variables	Initial Domains	First Propagation	Admissible solution	Optimal solution
Design Variables (DVs)				
Input Shaft				
$r_{a,0}$ (mm)	[10.0, 507.073]	[12.937, 503.942]	12.937	12.937
$l_{a,0}$ (mm)	[10.0, 1014.15]	[10.0, 1014.15]	10.155	10.0
Stage1				
m_1	[0.5, 50.0]	[0.5, 22.0]	1.5	1.5
$Z_{1,1}$	[11, 45]	[11, 45]	14	14
$Z_{2,1}$	[20, 150]	[20, 150]	20	20
b_1 (mm)	[10, 500]	[10, 330]	10.0	10.0
$r_{a,1}$ (mm)	[10.0, 507.073]	[10.0, 503.875]	10.0	10.0
$l_{a,1}$ (mm)	[10.0, 1014.15]	[10.0, 1014.15]	19.1168	84.9537729
ξ_1 (rad)	[-π, π]	[-π, π]	-0.1533	-0.058
Stage 2				
m_2	[0.5, 50.0]	[0.5, 22.0]	5.5	6.0
$Z_{1,2}$	[11, 45]	[11, 45]	16	17
$Z_{2,2}$	[20, 150]	[20, 150]	56	60
b_2 (mm)	[10, 500]	[10, 330]	14.0	10.0
$r_{a,2}$ (mm)	[10.0, 507.073]	[10.0, 503.875]	10.0	10.0
$l_{a,2}$ (mm)	[10.0, 1014.15]	[10.0, 1014.15]	10.0	10.0
ξ_2 (rad)	[-π, π]	[-π, π]	-0.0169	
Stage 3				
m_3	[0.5, 50.0]	[0.5, 22.0]	2.75	2.25
$Z_{1,3}$	[11, 45]	[11, 45]	17	17
$Z_{2,3}$	[20, 150]	[20, 150]	149	149
b_3 (mm)	[10, 500]	[10, 330]	16.0	10.0
$r_{a,3}$ (mm)	[10.0, 507.073]	[10.0, 503.875]	10.0	10.0
$l_{a,3}$ (mm)	[10.0, 1014.15]	[10.0, 1014.15]	189.22	123.5182346
ξ_3 (rad)	[-π, π]	[-π, π]	-0.4	-0.09
Performance Indicator				
f_{obj} (mm³)	[0, +	[24842.14,20821e7]	13194785.4	**8.1 e6**

$$Z_\beta^2 = \cos\beta_0$$

$$Z_H^2 = \frac{2\cos\beta_0\cos\alpha_{n0}\cos\alpha_t'}{\cos\alpha_{t0}{}^3\sin\alpha_t'}$$

$$Z_\varepsilon^2 = \begin{cases} \frac{1}{\varepsilon_\alpha} & si\ \varepsilon_\beta > 1 \\ \frac{4-\varepsilon_\alpha}{3}(1-\varepsilon_\beta) + \frac{\varepsilon_\beta}{\varepsilon_\alpha} & si\ \varepsilon_\beta < 1 \end{cases}$$

$$\varepsilon_\beta = \frac{b.\sin\beta_0}{\mu.m_{n0}}$$

- C_3 factor:

$$C_3 = \frac{Z_v^2}{K_v}$$

$$Z_v^2 = C_{zv} + \frac{2(1-C_{zv})}{\sqrt{0.8+\frac{32}{V}}}$$

$$C_{zv} = 0.85 - \frac{0.85(\sigma_{Hlim}-850)}{350}$$

$$\text{Si } \sigma_{Hlim} < 850MPa \text{ alors } \sigma_{Hlim} = 850\ MPa$$

$$\text{Si } \sigma_{Hlim} > 850MPa \text{ alors } \sigma_{Hlim} = 1200\ MPa$$

$$V = \frac{\pi.m_{n0}Z_1N_1}{60.10^3\cos\beta_0}$$

$$K_v = 1 + \left[\frac{K_1}{K_A\frac{F_t}{b}} + K_2\right].\frac{Z_1.V}{100}.\sqrt{\frac{u^2}{u^2+1}}$$

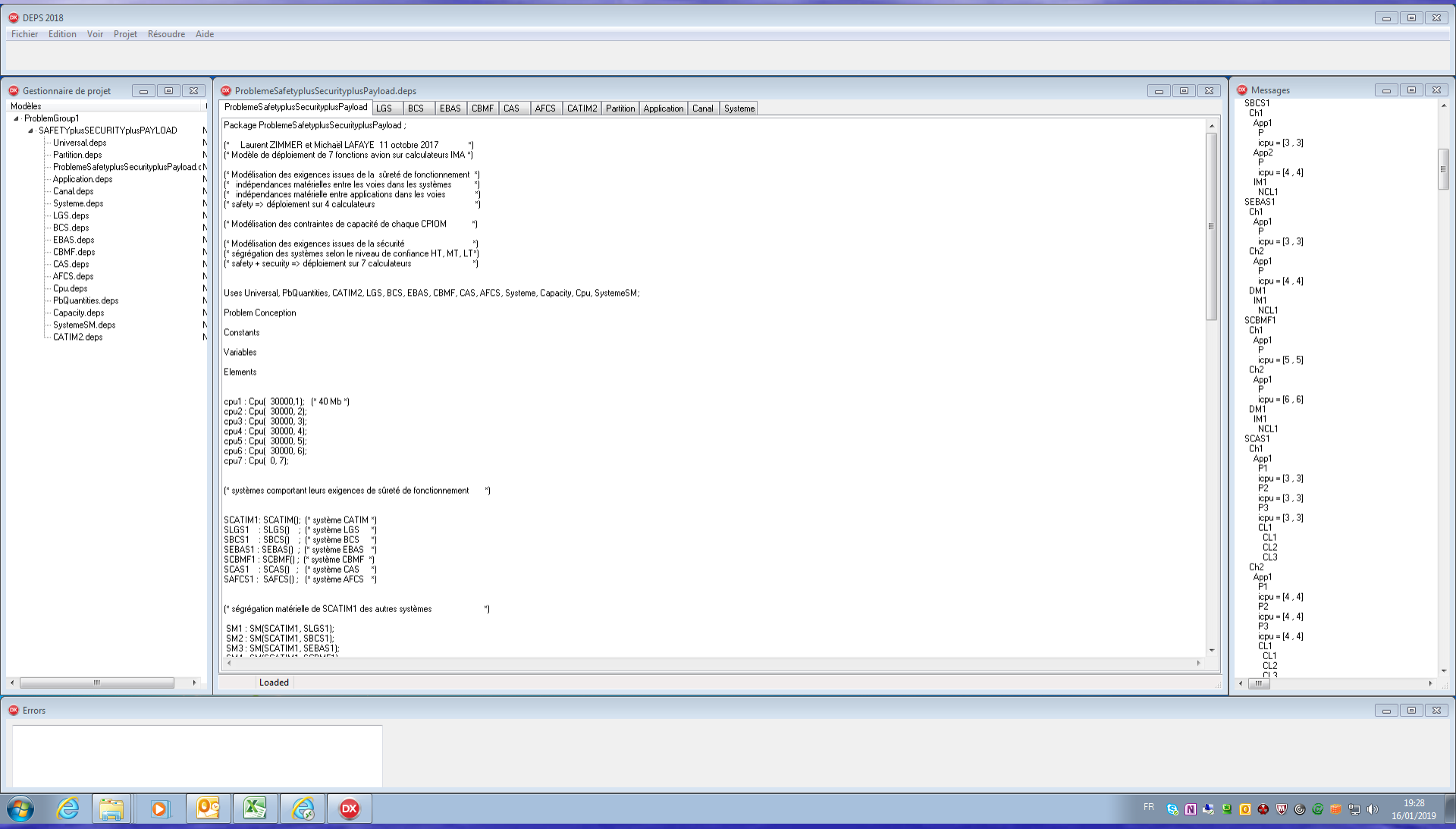
DEPS

(Yvars, Zimmer In Progress since 2014)

- **DEPS Studio** is the IDE
  - In fact an Integrated Modelling and Solving Environment
  - Project manager, Editors, debugger, **DEPS compiler**, **DEPS solver**
- **DEPS** (**DE**sign **P**roblem **S**pecification) **Language** is a DSML
- Constraint Solving Methods

DEPS ISN'T A TOOL !

DEPS IDE


DEPS 2018
Fichier Edition Voir Projet Résoudre Aide
Gestionnaire de projet
Modèles
ProblemGroup1
SAFETYplusSECURITYplusPAYLOAD
Universal.deps
Partition.deps
ProblemeSafetyplusSecurityplusPayload.d
Application.deps
Canal.deps
Systeme.deps
LGS.deps
BCS.deps
EBAS.deps
CBMF.deps
CAS.deps
AFCS.deps
Cpu.deps
PbQuantities.deps
Capacity.deps
SystemeSM.deps
CATIM2.deps
ProblemeSafetyplusSecurityplusPayload.deps
ProblemeSafetyplusSecurityplusPayload LGS BCS EBAS CBMF CAS AFCS CATIM2 Partition Application Canal Systeme
Package ProblemeSafetyplusSecurityplusPayload ;
(* Laurent ZIMMER et Michaël LAFAYE 11 octobre 2017 *)
(* Modèle de déploiement de 7 fonctions avion sur calculateurs IMA *)
(* Modélisation des exigences issues de la sûreté de fonctionnement *)
(* indépendances matérielles entre les voies dans les systèmes *)
(* indépendances matérielle entre applications dans les voies *)
(* safety => déploiement sur 4 calculateurs *)
(* Modélisation des contraintes de capacité de chaque CPIOM *)
(* Modélisation des exigences issues de la sécurité *)
(* ségrégation des systèmes selon le niveau de confiance HT, MT, LT *)
(* safety + security => déploiement sur 7 calculateurs *)
Uses Universal, PbQuantities, CATIM2, LGS, BCS, EBAS, CBMF, CAS, AFCS, Systeme, Capacity, Cpu, SystemeSM;
Problem Conception
Constants
Variables
Elements
cpu1 : Cpu(30000,1); (* 40 Mb *)
cpu2 : Cpu(30000, 2);
cpu3 : Cpu(30000, 3);
cpu4 : Cpu(30000, 4);
cpu5 : Cpu(30000, 5);
cpu6 : Cpu(30000, 6);
cpu7 : Cpu(0, 7);
(* systèmes comportant leurs exigences de sûreté de fonctionnement *)
SCATIM1: SCATIM(); (* système CATIM *)
SLGS1 : SLGS() ; (* système LGS *)
SBCS1 : SBCS() ; (* système BCS *)
SEBAS1 : SEBAS() ; (* système EBAS *)
SCBMF1 : SCBMF() ; (* système CBMF *)
SCAS1 : SCAS() ; (* système CAS *)
SAFCS1 : SAFCS() ; (* système AFCS *)
(* ségrégation matérielle de SCATIM1 des autres systèmes *)
SM1 : SM(SCATIM1, SLGS1);
SM2 : SM(SCATIM1, SBCS1);
SM3 : SM(SCATIM1, SEBAS1);
Loaded
Messages
SBCS1
Ch1
App1
P
icpu = [3 , 3]
App2
P
icpu = [4 , 4]
IM1
NCL1
SEBAS1
Ch1
App1
P
icpu = [3 , 3]
Ch2
App1
P
icpu = [4 , 4]
DM1
IM1
NCL1
SCBMF1
Ch1
App1
P
icpu = [5 , 5]
Ch2
App1
P
icpu = [6 , 6]
DM1
IM1
NCL1
SCAS1
Ch1
App1
P1
icpu = [3 , 3]
P2
icpu = [3 , 3]
P3
icpu = [3 , 3]
CL1
CL1
CL2
CL3
Ch2
App1
P1
icpu = [4 , 4]
P2
icpu = [4 , 4]
P3
icpu = [4 , 4]
CL1
CL1
CL2
Errors
FR
19:28
16/01/2019

Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

The DEPS language declarative features

- Model-based Knowledge Representation
  - inheritance, composition, aggregation, polymorphism
  - Attributes
    - constants, variables
    - integer or real values
  - Properties
    - algebraic equations or inequalities
- Ontology for engineers
  - quantities, dimensions, units

The DEPS language the compiler

- Ahead-of-time
- Package management
- Parsing/Error handling
- Generation of sub-defined model instances

The DEPS Language

Object-Oriented Language + Constraint Solving Language = DEPS Language

Sub-defined Models + Constraints = Models of Properties

```
Model Partition ()
Constants
Variables
  icpu : CpuIndex ;
Elements
Properties
End
```

```
Quantity
CpuIndex
 Kind : Integer ;
 Min  : 1 ;
 Max : 4 ;
 Unit : u ;
End
```

```
P1.icpu = P2.icpu;
```

```
Model colocalisation (P1, P2)
Constants
Variables
Elements
 P1 : Partition ;
 P2 : Partition ;
Properties
  P1.icpu = P2.icpu;
End
```

Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case: Synthesis of avionics embedded systems
- Ongoing studies and developments

DEPS Solver (1)

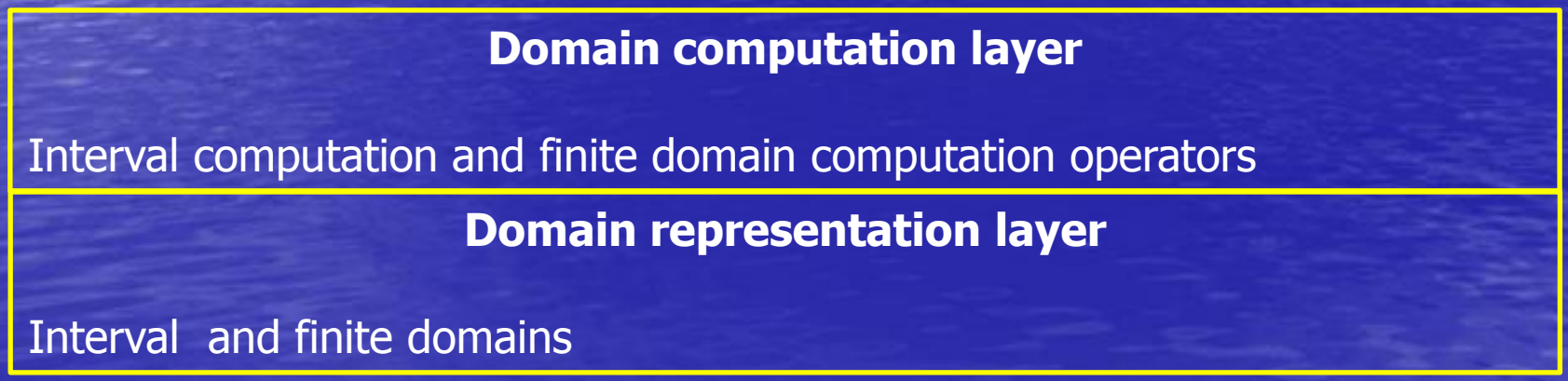
- A Need of solving capabilities addressing :
  - under constrained problems,
  - mixed non linear mathematical problems,
  - both equations and inequalities,
  - other relations.
- A Constraint based oriented solver
  - constraint programming on mix domains
- Built for dealing with DEPS models
- An object oriented architecture
  - extensible

DEPS Solver (2)

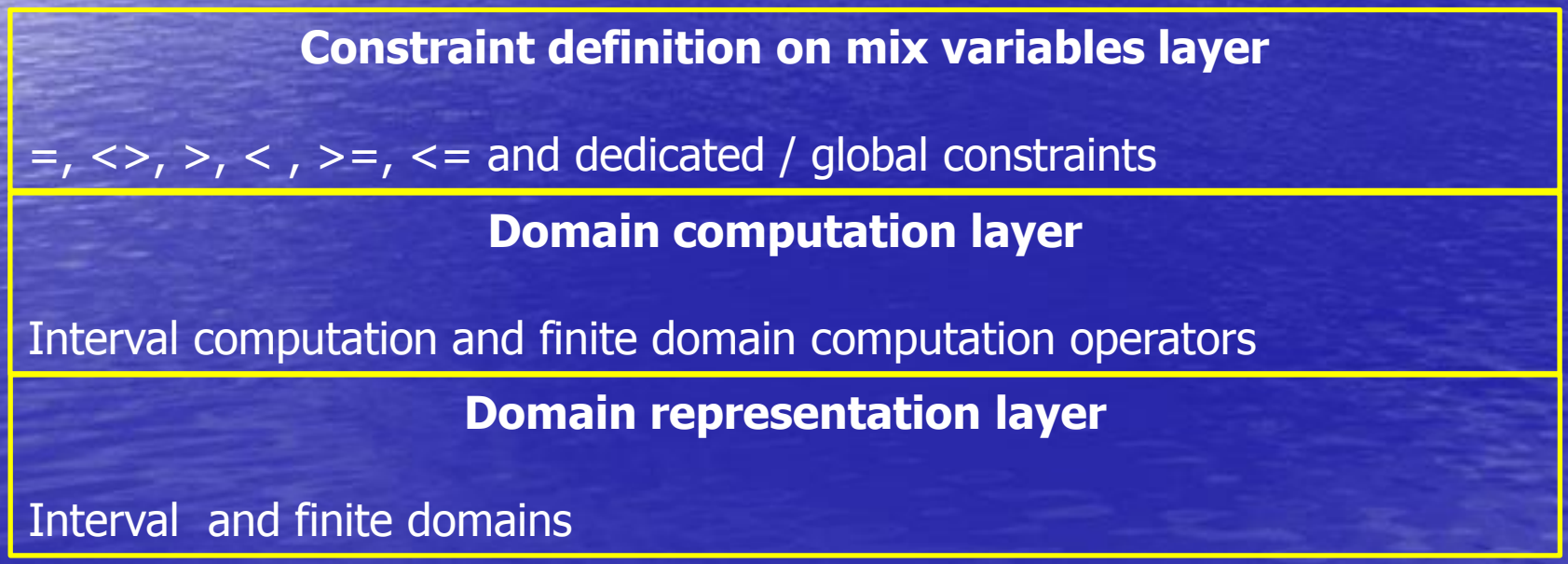
Domain representation layer

Interval and finite domains

DEPS Solver (3)



DEPS Solver (4)



DEPS Solver (5)

Constraint propagation layer Hull consistency (HC4Rev) extended to mix domains
Constraint definition on mix variables layer =, <>, >, < , >=, <= and dedicated / global constraints
Domain computation layer Interval computation and finite domain computation operators
Domain representation layer Interval and finite domains

DEPS Solver (6)

Solving mechanism layer

Branch and prune algorithm
Spliting strategies (first fail, round robin), variables scheduling

Constraint propagation layer

Hull consistency (HC4Rev) extended to mix domains

Constraint definition on mix variables layer

=, <>, >, < , >=, <= and dedicated / global constraints

Domain computation layer

Interval computation and finite domain computation operators

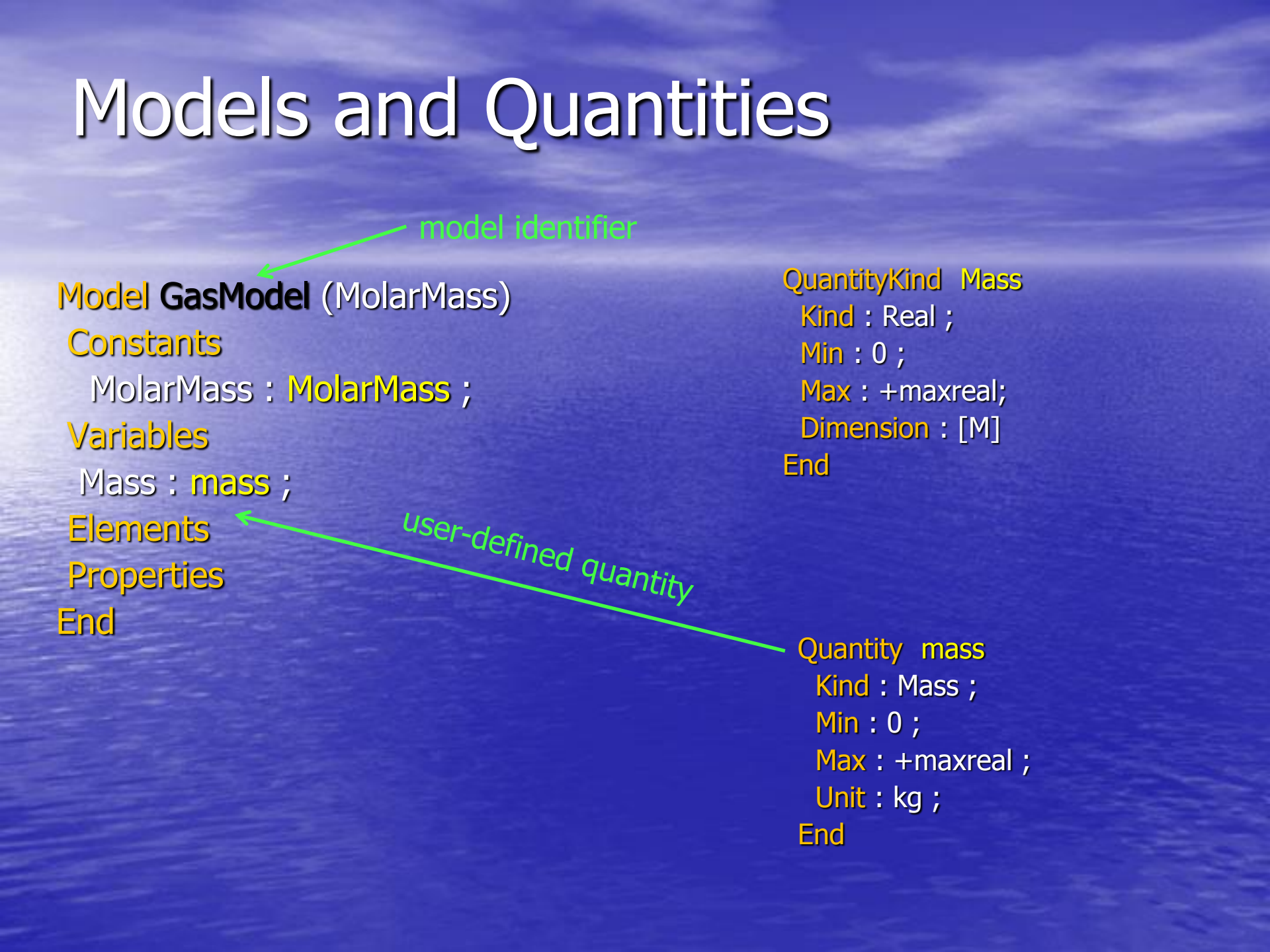
Domain representation layer

Interval and finite domains

Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

Models and Quantities



Part and shared models

reference binding

```
Model Tank(p, t, Gas)
 Constants
  R : Real = 8.314 ;
  p : Pressure ; instance declaration
  t :  Temperature ;
 Variables
  V : Volume ;
 Elements
 Gas : GasModel ;
 Properties
  p*V= (Gas.Mass/Gas.MolarMass)*R*t;
End
```

Instance construction
O2 part of Problem

reference binding
O2 shared by T1 and T2

```
Model Problem()
 Constants
 Variables
 Elements
  O2:GasModel(0.032);
  H2:GasModel(0.002);
  T1,T2:Tank(2.56e+7, 300, O2);
  T3:Tank(7.00e+7, 300, H2);
 Properties
  O2.Mass = 10 ;  (* or T1.Gas *)
  T1.V+T2.V< 500 ;
End
```

Aliases

```
Model Gas (ckilo, molarMass)
 Constants
  ckilo : DollarCostPerKilo;
  molarMass : MolarMass;
 Variables
  M : mass ;
  expr CoutStockage : DollarCost  ;
 Elements
 Properties
  CoutStockage := ckilo*M ;
End
```

Alias declaration

Alias definition

Universal constants

```
Model circle ()
 Constants
 Variables
  Diameter, Circumference: length ;
 Elements
 Properties
  Circumference = uPi*D ;
End
```

Derived models (inheritance)

```
Model Component (index)
 Constants
  index : ComponentIndex ;
 Variables
  I : intensity ;
 Elements
   P1, P2 : Port() ;
 Properties
   P1.I :=  I;
   P2.I := -I;
End
```

```
Model Port
 Constants
 Variables
  V : voltage ;
  expr I : intensity ;
 Elements
 Properties
End
```

```
Model Resistor(R) extends Component
 Constants
  R : Resistor;
 Variables
 Elements
 Properties
  P1.V-P2.V = R*I;
End
```

1 resistor model:

- 3 variables (unknowns)
- 1 equation
- 2 expressions

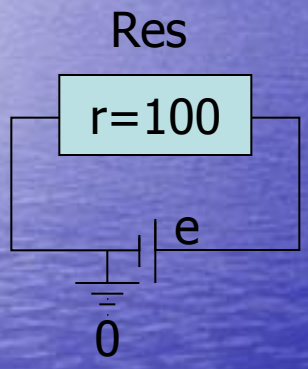
Model Signature

```
Model Component (index)
Constants
  index : ComponentIndex ;
 Variables
  I : intensity ;
 Elements
   P1, P2 : Port() ;
 Properties
   P1.I :=  I;
   P2.I := -I;
End
```

```
Model Resistor(R) extends Component [ComponentIndex]
 Constants
  R : Resistor;
 Variables
 Elements
 Properties
  P1.V-P2.V = R*I;
End
```

```
Model Resistor() extends Component [ComponentIndex]
 Constants
 Variables
 R : Resistor;
 Elements
 Properties
  P1.V-P2.V = R*I;
End
```

A very simple circuit

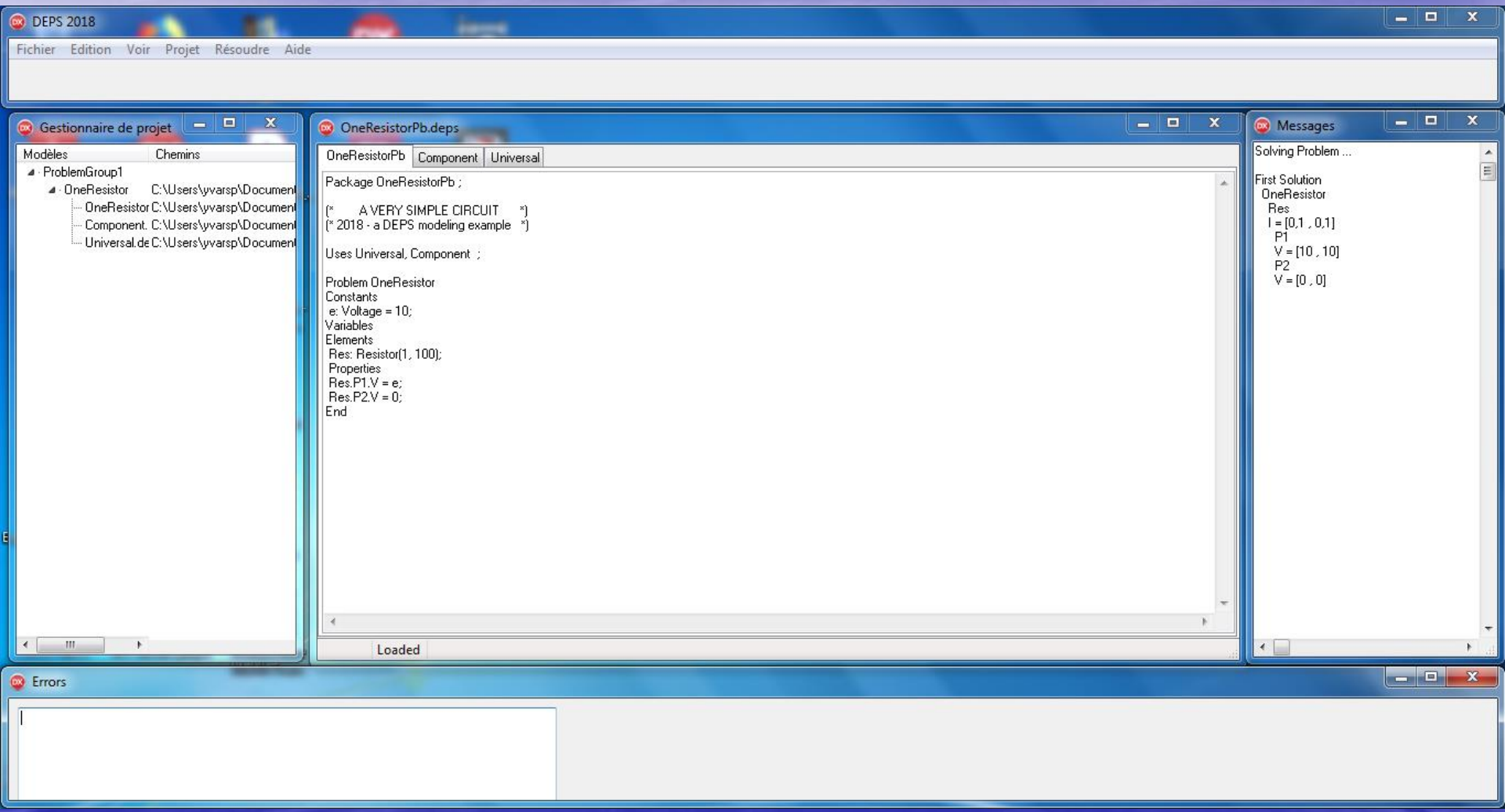



```
Problem One Resistor
 Constants
  e : Voltage = 10 ;
 Variables
 Elements
  Res : Resistor(1,100) ;
 Properties
  Res.P1.V  = e ;
  Res.P2.V  = 0;
End
```

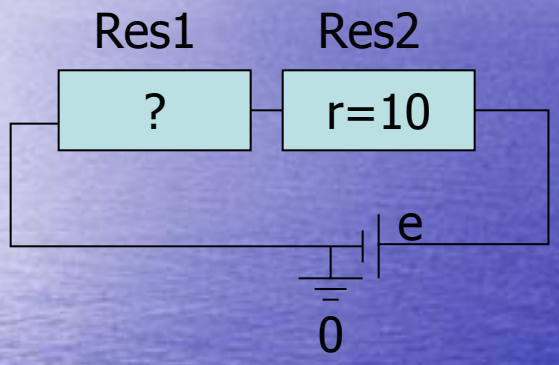
Order of bindings

1 problem :
- 3 variables (unknowns)
- 3 equations

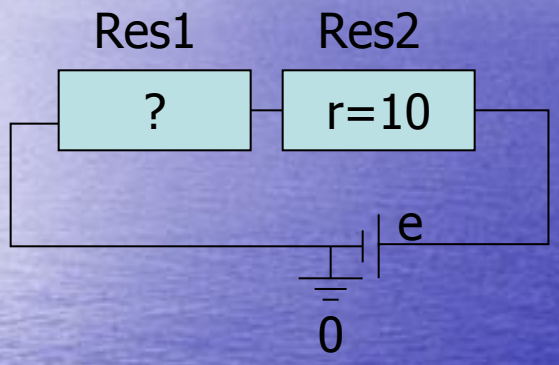
A very simple circuit



More complicated (1)



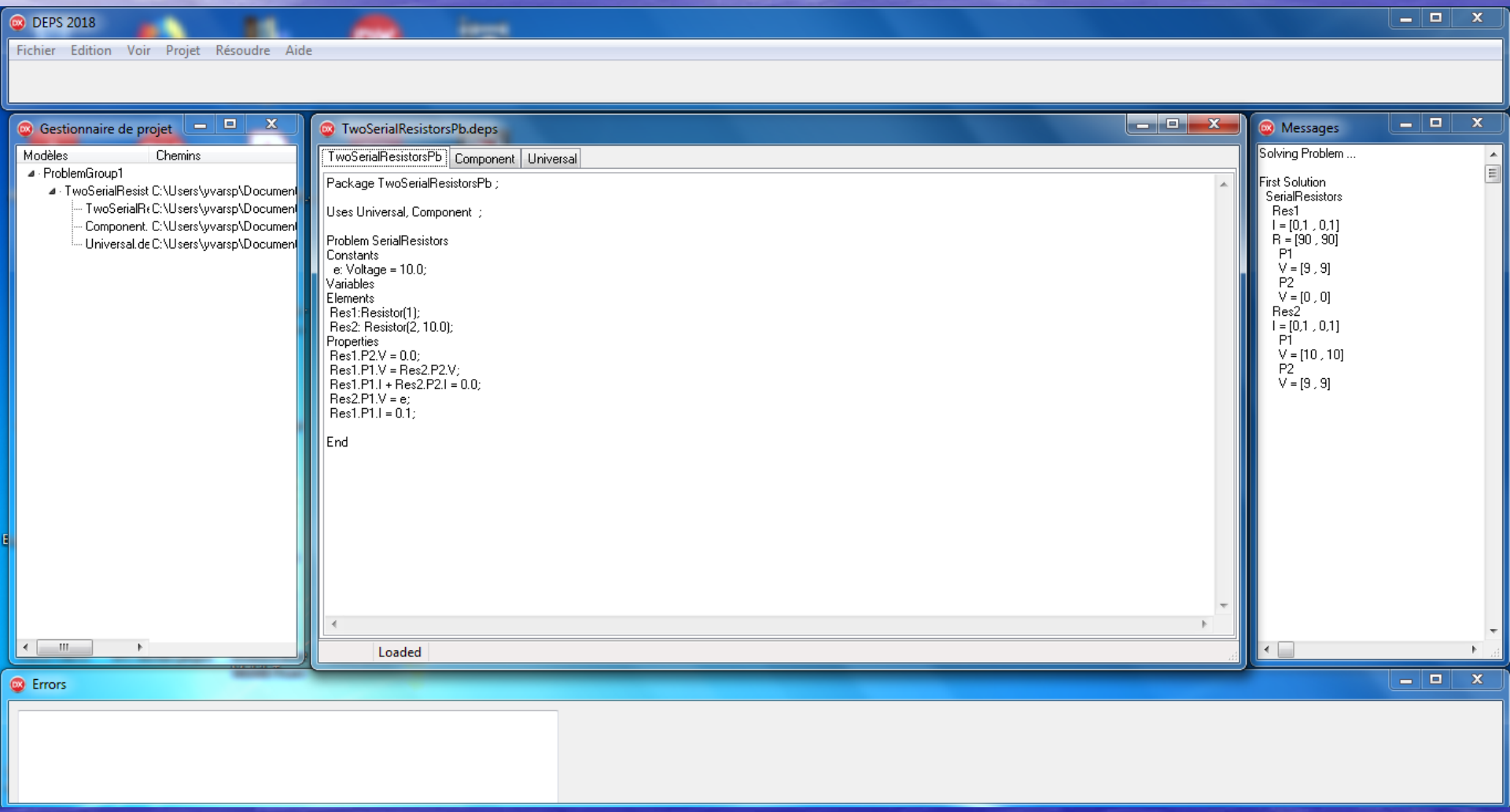
More complicated (2)



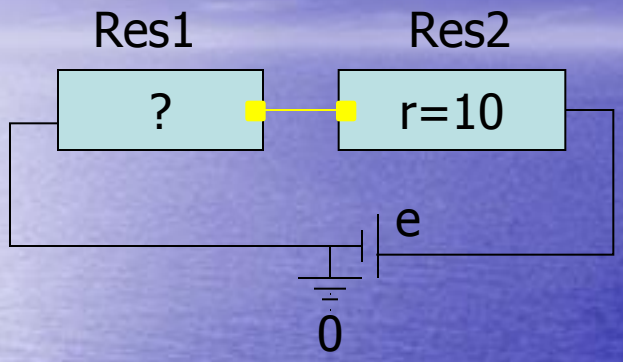
1 problem :
- 2x3 = 6 variables (unknowns)
- 2+4 =6 equations

```
Problem SerialResistors
 Constants
  e : Voltage = 10;
 Variables
 Elements
  Res1 : Resistor(1) ;
  Res2 : Resistor(2,10) ;
 Properties
  Res1.P2.V = 0 ;
  Res1.P1.V = Res2.P2.V ;
  Res1.P1.I + Res2.P2.I = 0 ;
  Res2.P1.V = e ;
  Res1.I = 0.1;
End
```

More complicated (3)

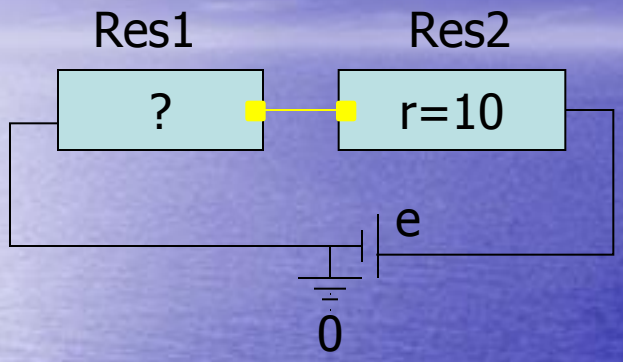


Improvement (1)




```
Model Node (P1,P2)
 Constants
 Variables
 Elements
  P1, P2 : Port ;
 Properties
  P1.V = P2.V ;
  P1.I+P2.I = 0 ;
End
```

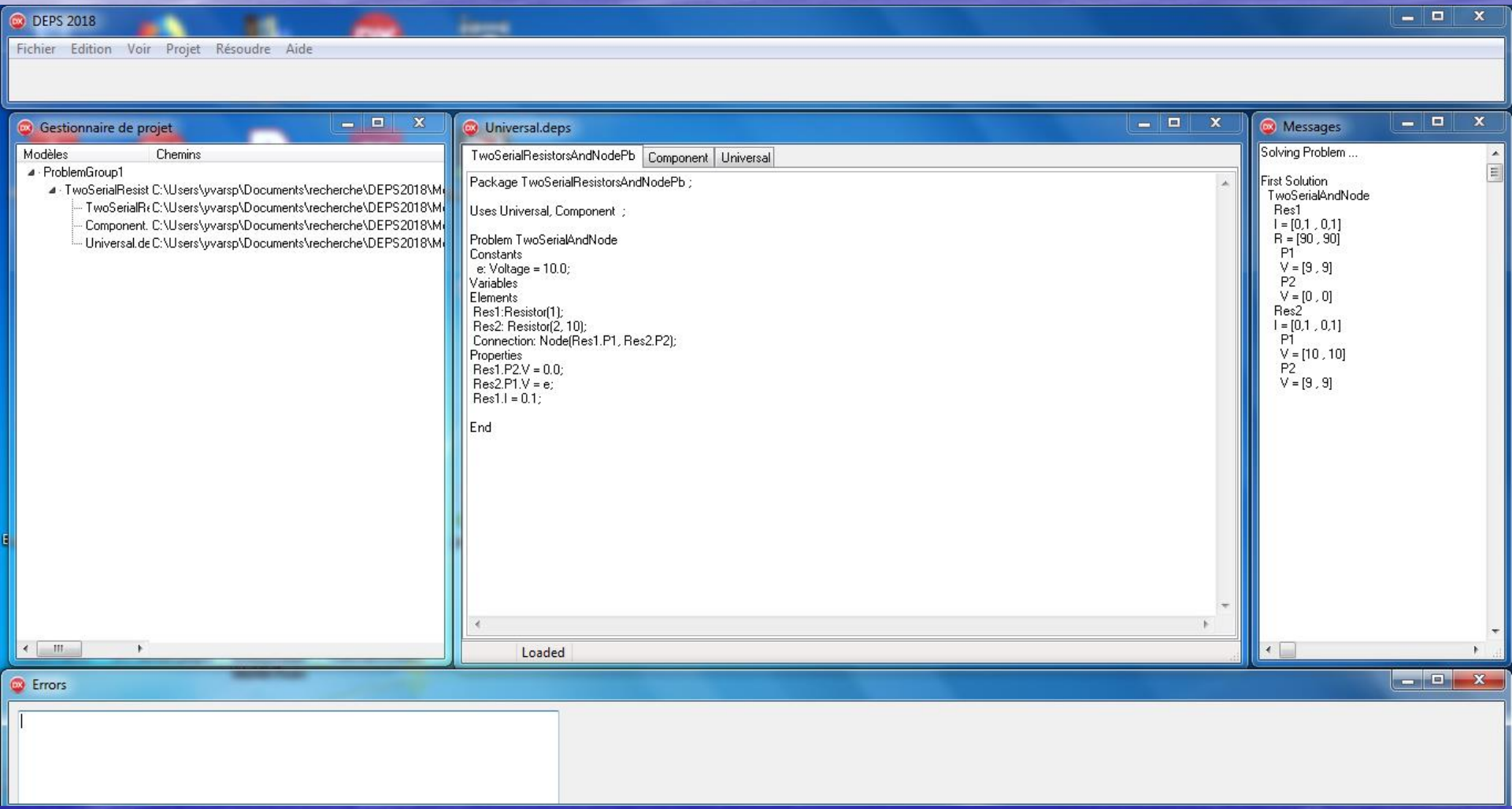
Improvement (2)



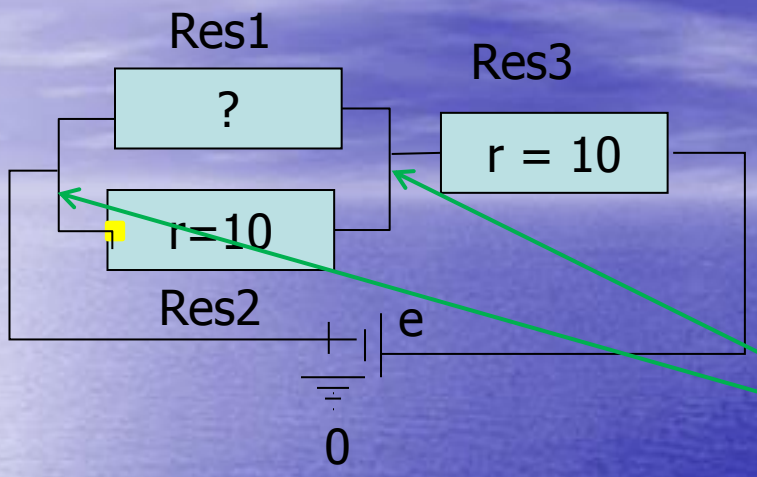

```
Model Node (P1,P2)
 Constants
 Variables
 Elements
  P1, P2 : Port ;
 Properties
  P1.V = P2.V ;
  P1.I+P2.I = 0 ;
End
```

```
Problem  TwoSerialAndNode
 Constants
  e : Voltage = 10;
 Variables
 Elements
  Res1 : Resistor(1) ;
  Res2 : Resistor(2, 10) ;
  Connection : Node(Res1.P1, Res2.P2) ;
 Properties
  Res1.P2.V = 0 ;
  Res2.P1.V = e ;
  Res1.I = 0.1;
End
```

Improvement (3)

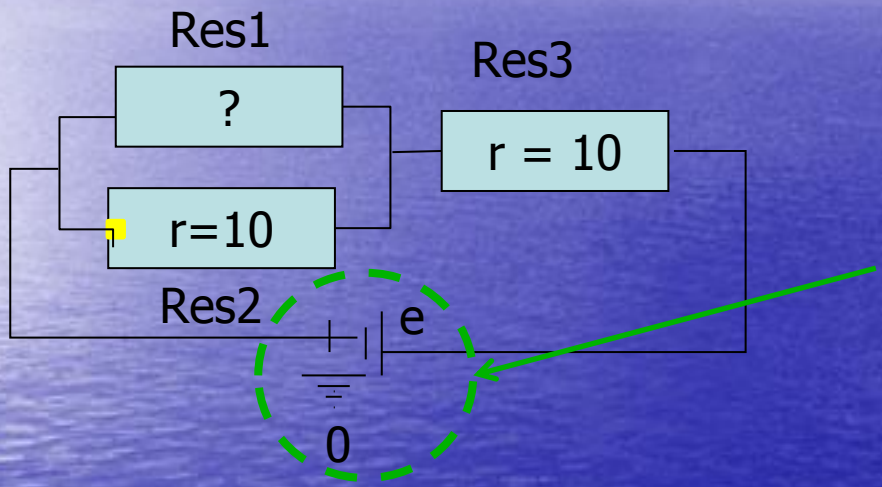


Serial and parallel (1)



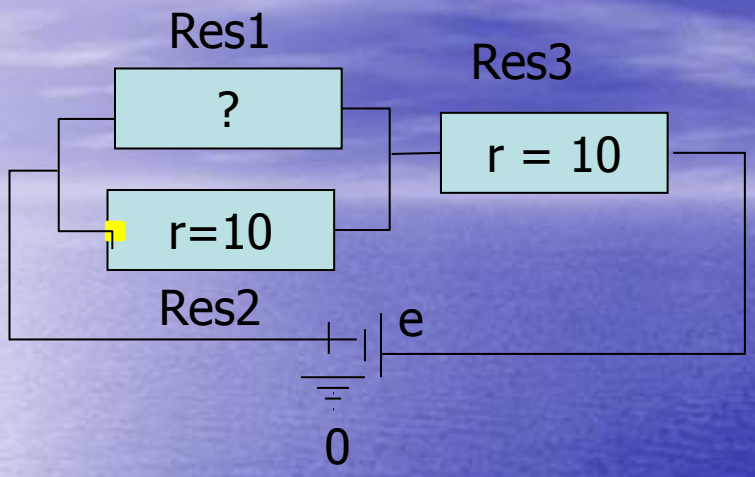

```
Model Node (P1,P2, P3)
 Constants
 Variables
 Elements
  P1, P2, P3 : Port ;
 Properties
  P1.V = P2.V ;
  P2.V = P3.V ;
  P3.V = P1.V ;
  P1.I+P2.I+P3.I = 0 ;
End
```

Serial and parallel (2)



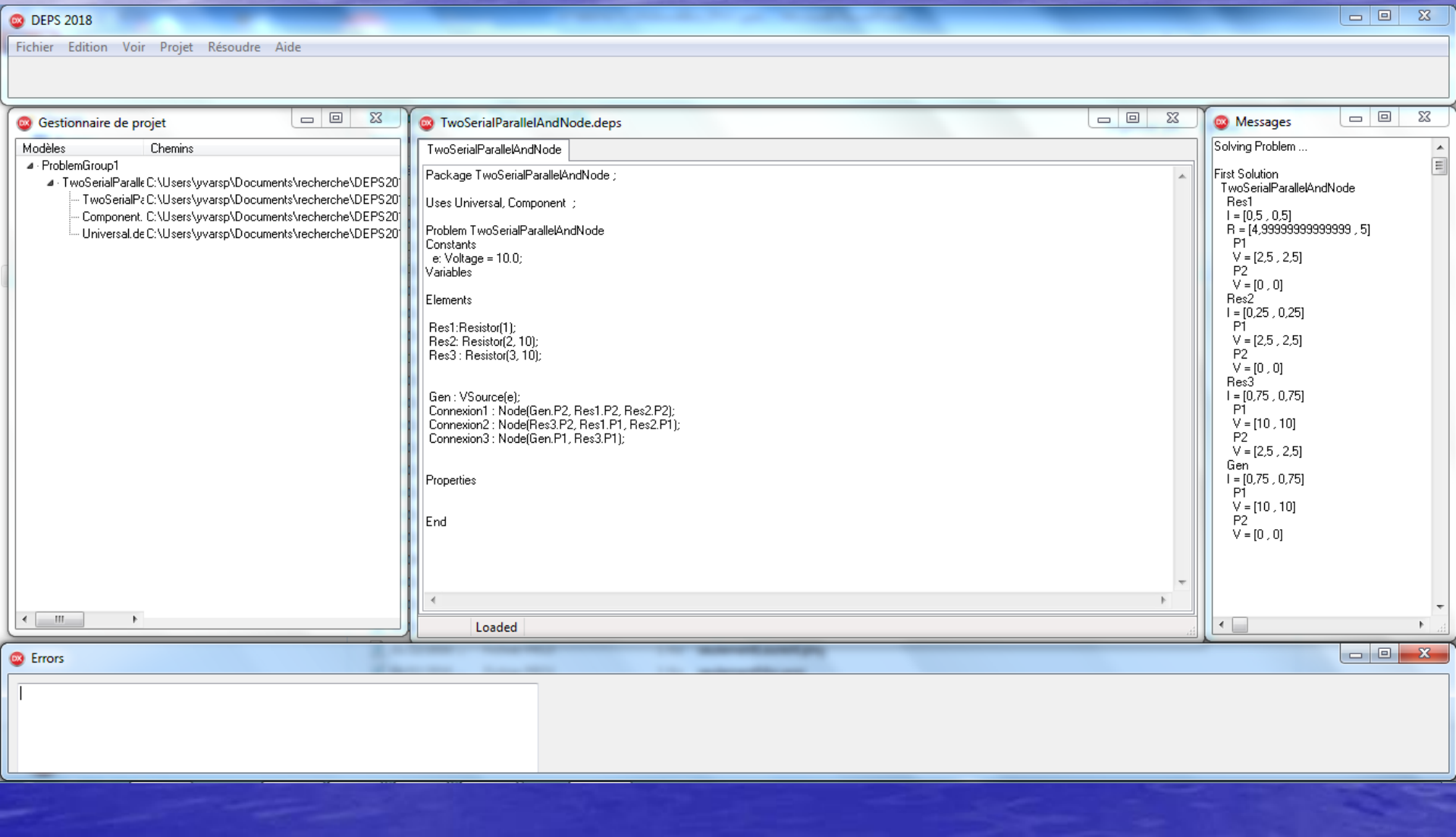

```
Model VSource(e)
 Constants
   e :Voltage;
Variables
   I : Intensity;
Elements
   P1 : Port();
   P2:  Port();
Properties
   P1.V = e;
   P2.V = 0;
   P2.I := I;
   P1.I := -I;
End
```

Serial and parallel (3)

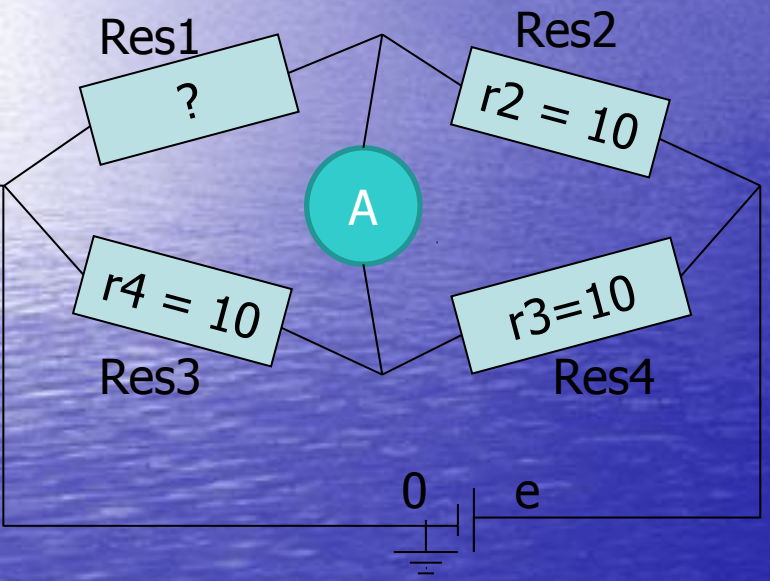



```
Problem TwoSerialParalellAndNode
 Constants
  e : Voltage = 10;
 Variables
 Elements
  Gen : Vsource(e);
  Res1 : Resistor(1) ;
  Res2 : Resistor(2, 10) ;
  Res3 : Resistor(2, 10) ;
  Connexion1 : Node(Gen.P2, Res1.P2,
    Res2.P2);
  Connexion2 : Node(Res3.P2, Res1.P1,
    Res2.P1);
  Connexion3 : Node(Gen.P1, Res3.P1);
Properties
End
```

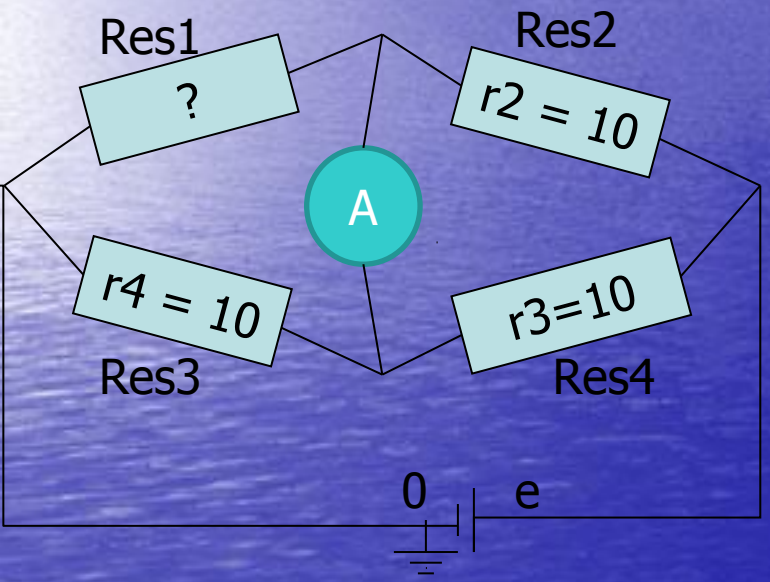
Serial and parallel (4)



Wheatstone Bridge (1)

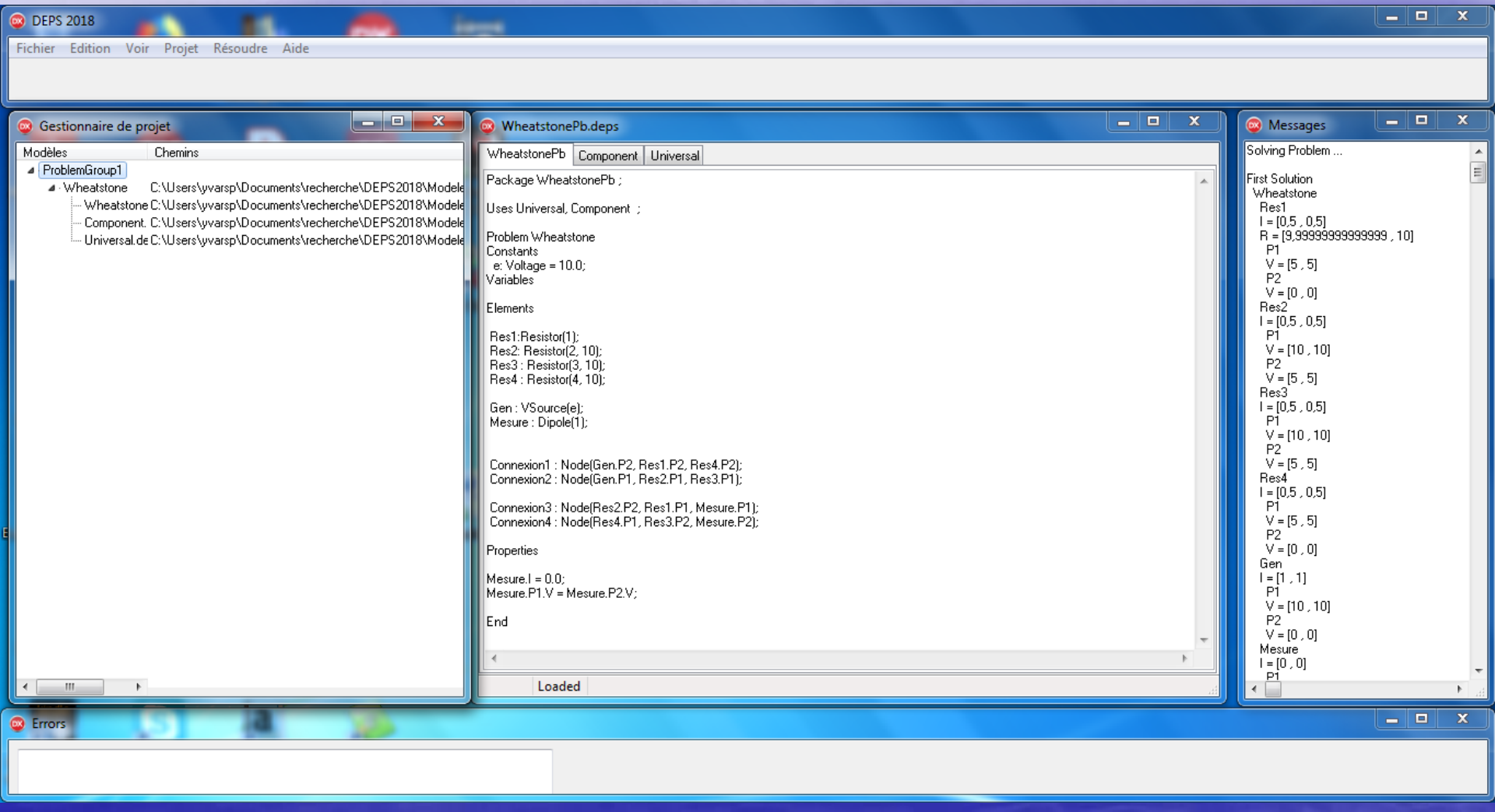


Wheatstone Bridge (2)




```
Problem Wheatstone
 Constants
  e : Voltage = 100;
 Variables
 Elements
 Res1:Resistor(1);
 Res2: Resistor(2, 10);
 Res3 : Resistor(3, 10);
 Res4 : Resistor(4, 10);
 Gen : VSource(e);
 Mesure : Component(1);
Connexion1 : Node(Gen.P2, Res1.P2, Res4.P2);
Connexion2 : Node(Gen.P1, Res2.P1, Res3.P1);
Connexion3 : Node(Res2.P2, Res1.P1, Mesure.P1);
Connexion4 : Node(Res4.P1, Res3.P2, Mesure.P2);
Properties
Mesure.I = 0.0;
Mesure.P1.V = Mesure.P2.V;
End
```

Wheatstone Bridge (3)



Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

Use-case:
Synthesis of avionics embedded systems

Modélisation d'exigences et synthèse d'architecture de plateforme informatique embarquée

Laurent Zimmer (Direction de la Prospective)

DASSAULT AVIATION

CORAC AME

Processus Générique de Définition d'Architecture

- Objectifs :
  - Définir un processus de conception d'architecture de plateforme informatique embarquée, pour un périmètre fonctionnel étendu à l'ensemble des domaines.
  - Via un ensemble d'étapes successives, le processus doit générer de façon assistée une architecture, répondant aux besoins opérationnels et aux contraintes de ses fonctions embarquées (de sûreté de fonctionnement, de sécurité des données, etc..).
  - Ce processus de génération doit, autant que possible, être prouvé correct par construction. Pour ce faire, il s'appui sur un ensemble de modèles formels (i.e., reposant sur des notations mathématiques), et sur des techniques d'optimisation et de recherche de solutions (de type résolution de contraintes).

CORAC AME

Processus Générique de Définition d'Architecture

- Objectifs :
  - Définir un processus de conception d'architecture de plateforme informatique embarquée, pour un périmètre fonctionnel étendu à l'ensemble des domaines.
  - Via un ensemble d'étapes successives, le processus doit générer de façon assistée une architecture, répondant aux besoins opérationnels et aux contraintes de ses fonctions embarquées (de sûreté de fonctionnement, de sécurité des données, etc..).
  - Ce processus de génération doit, autant que possible, être prouvé correct par construction. Pour ce faire, il s'appui sur un ensemble de modèles formels (i.e., reposant sur des notations mathématiques), et sur des techniques d'optimisation et de recherche de solutions (de type résolution de contraintes).

Notre Propos

- Modéliser formellement (en DEPS) le maximum d'exigences et de contraintes de conception qui portent sur les fonctions avion

- Générer (ou vérifier) un déploiement correct par construction sur la plateforme cible en dimensionnant (si besoin) cette dernière

Modélisation en DEPS

- Modélisation système :
  - Modèles des fonctions avion, des canaux, des voies, des applications, des partitions
- Modélisation des éléments de plateforme
  - Modèle des calculateurs
- Modélisation d'exigences :
  1) Modèles de patrons de sûreté de fonctionnement des systèmes
  2) Modèles de contraintes de capacité
  3) Modèles de sécurité inter-systèmes

Formalisation des exigences de sûreté de fonctionnement

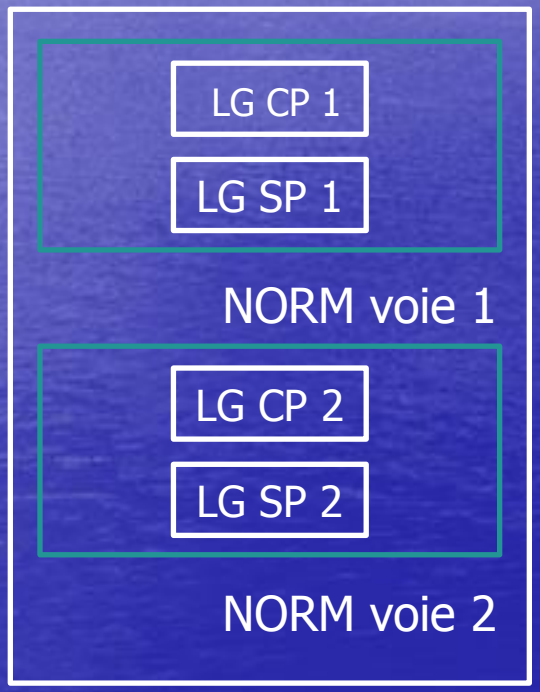
- Les experts en sûreté de fonctionnement étudient les cas de panne de chaque fonction avion et en fonction de la criticité de celles-ci préconisent :
  - des duplications, des triplications de chaînes de traitement, des redondances d'applications …
  - des ségrégations des ressources utilisées par les chaines ou les applications …

➢ Production de patrons d'architecture (schémas)

➢ Définition d'exigences de sûreté de fonctionnement associées (texte)

➢ *FORMALISATION des patrons et des exigences en DEPS*

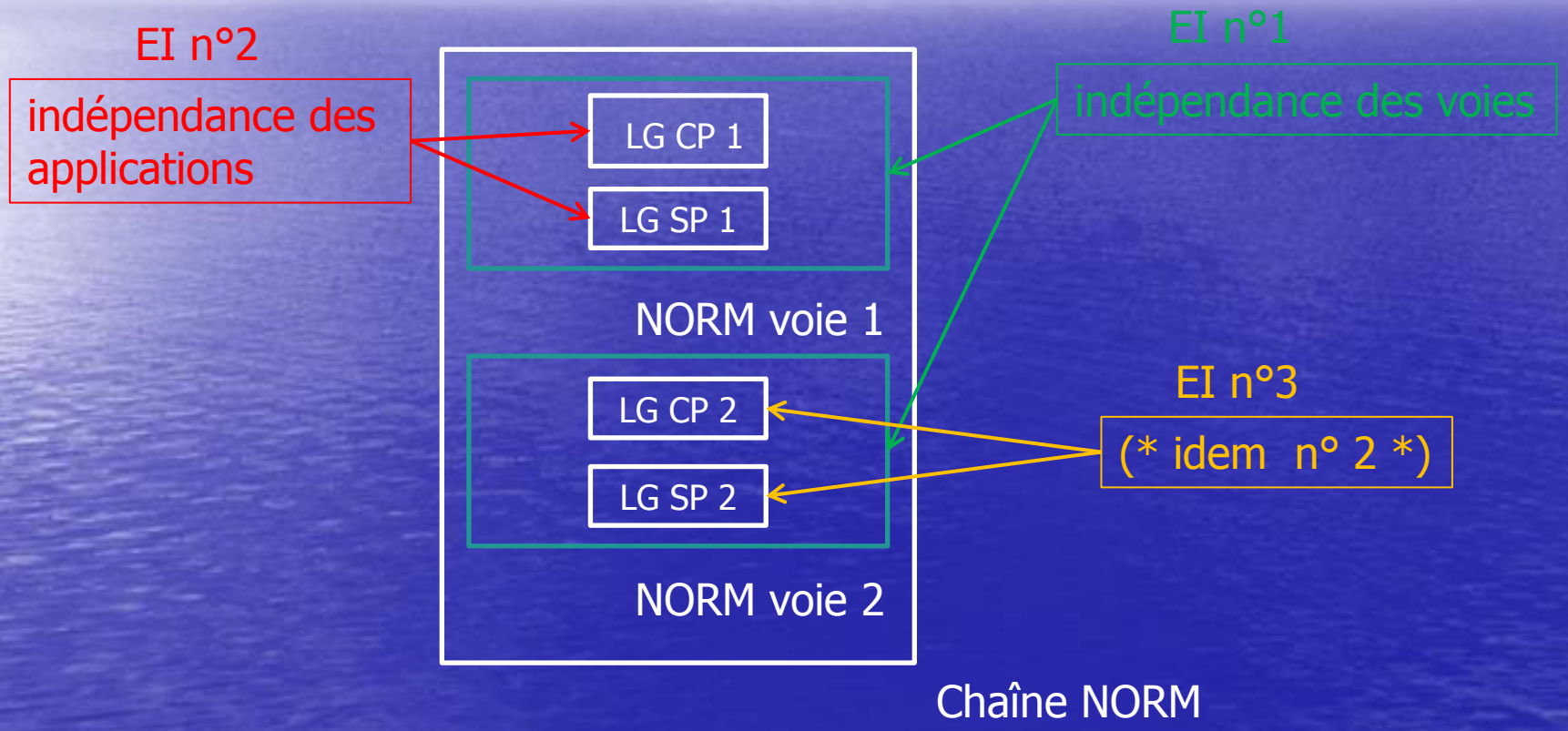
Patron d'architecture fonction LGS



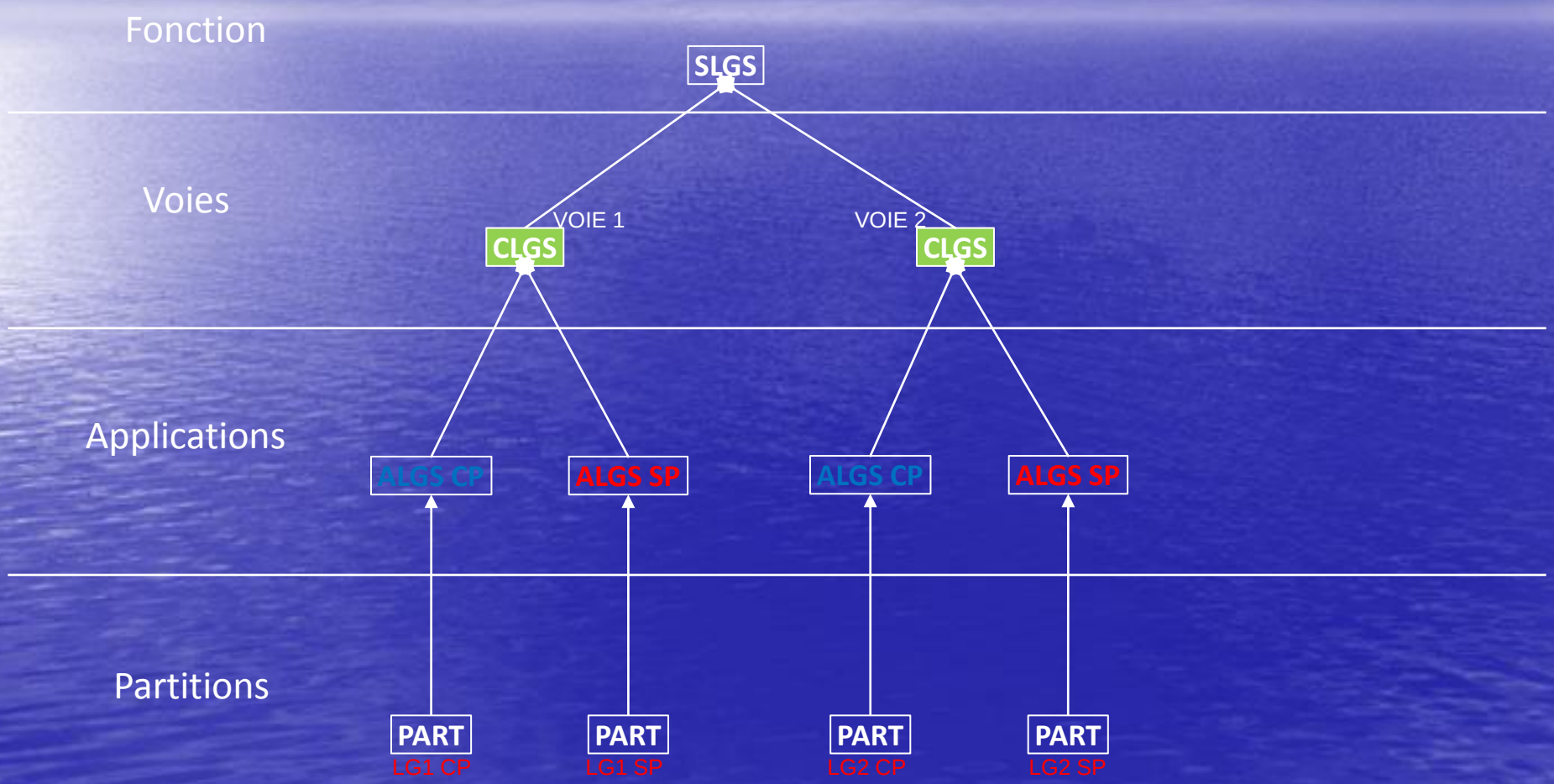
CP = control path
SP = safety path

Chaîne NORM

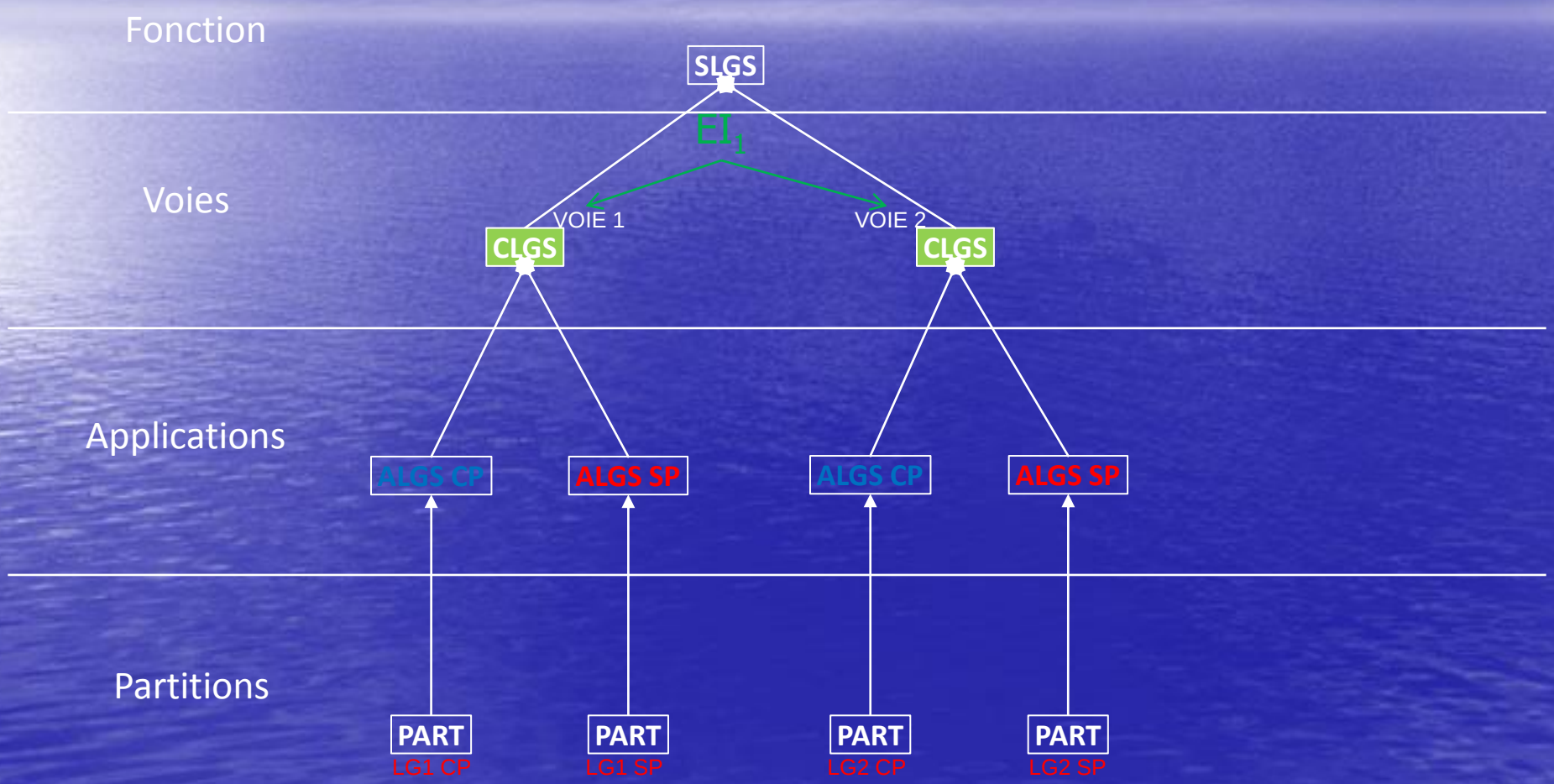
Exigences de Sûreté de Fonctionnement fonction LGS



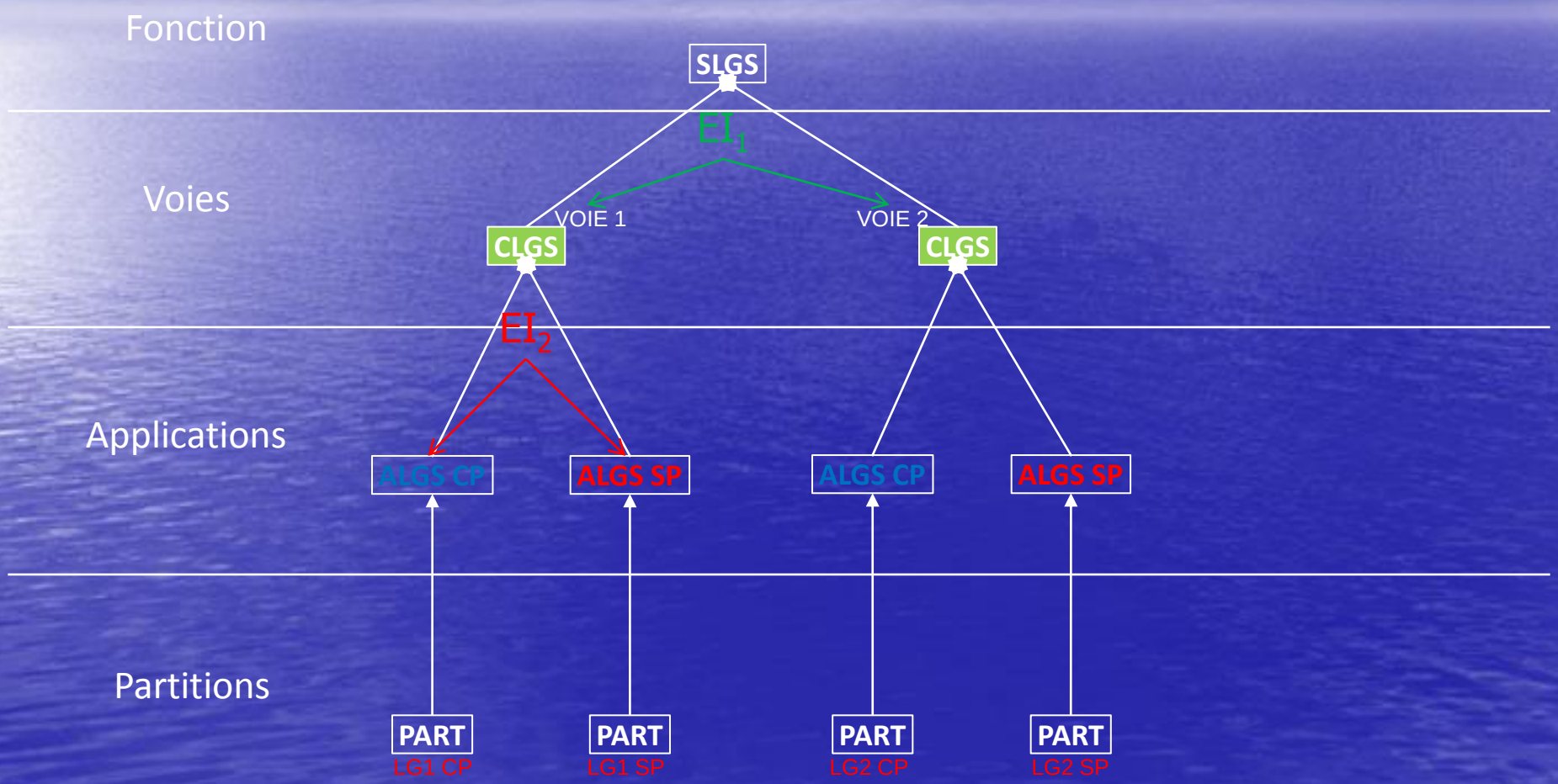
LGS : modélisation du patron



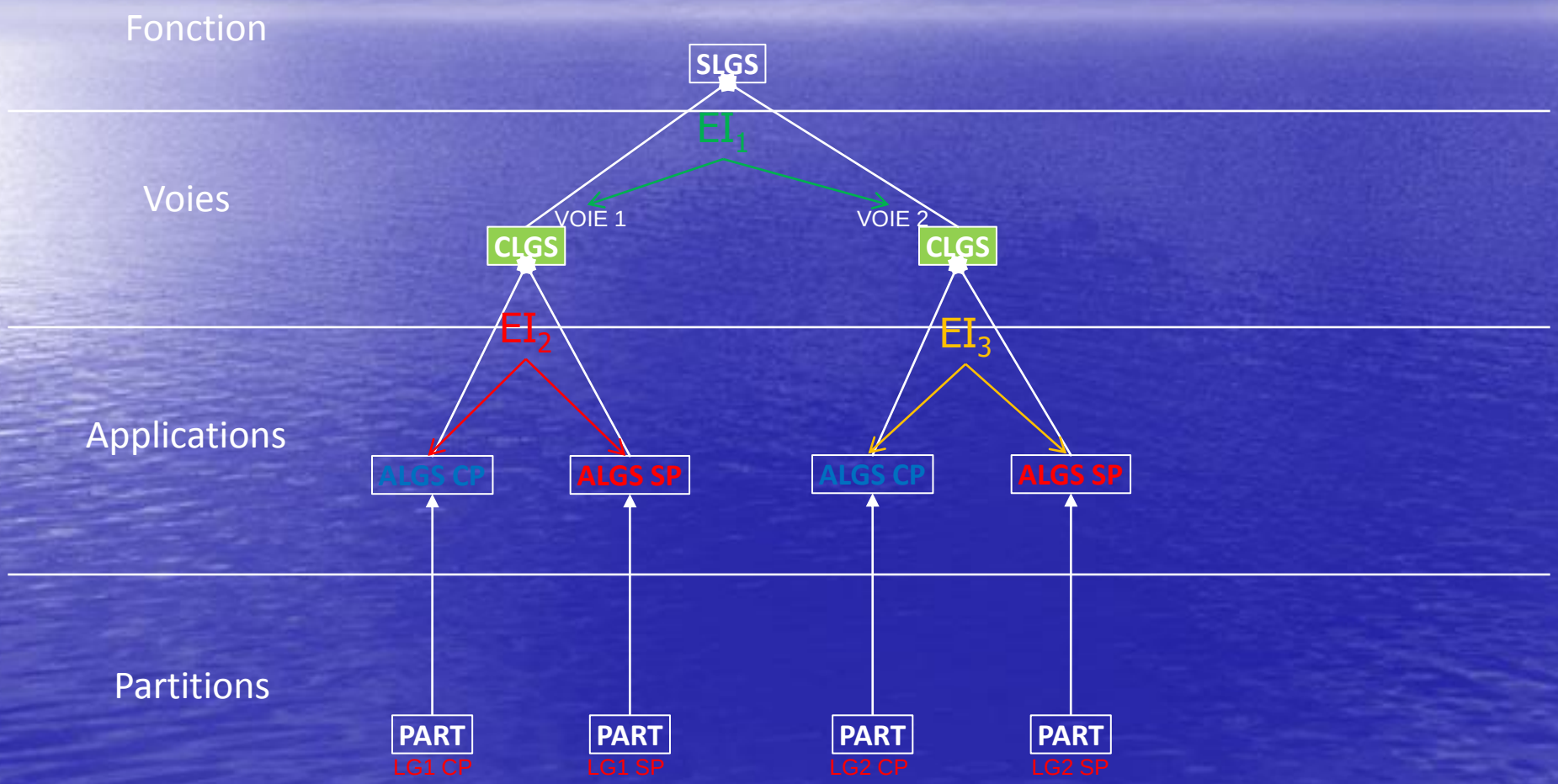
LGS : modélisation des exigences



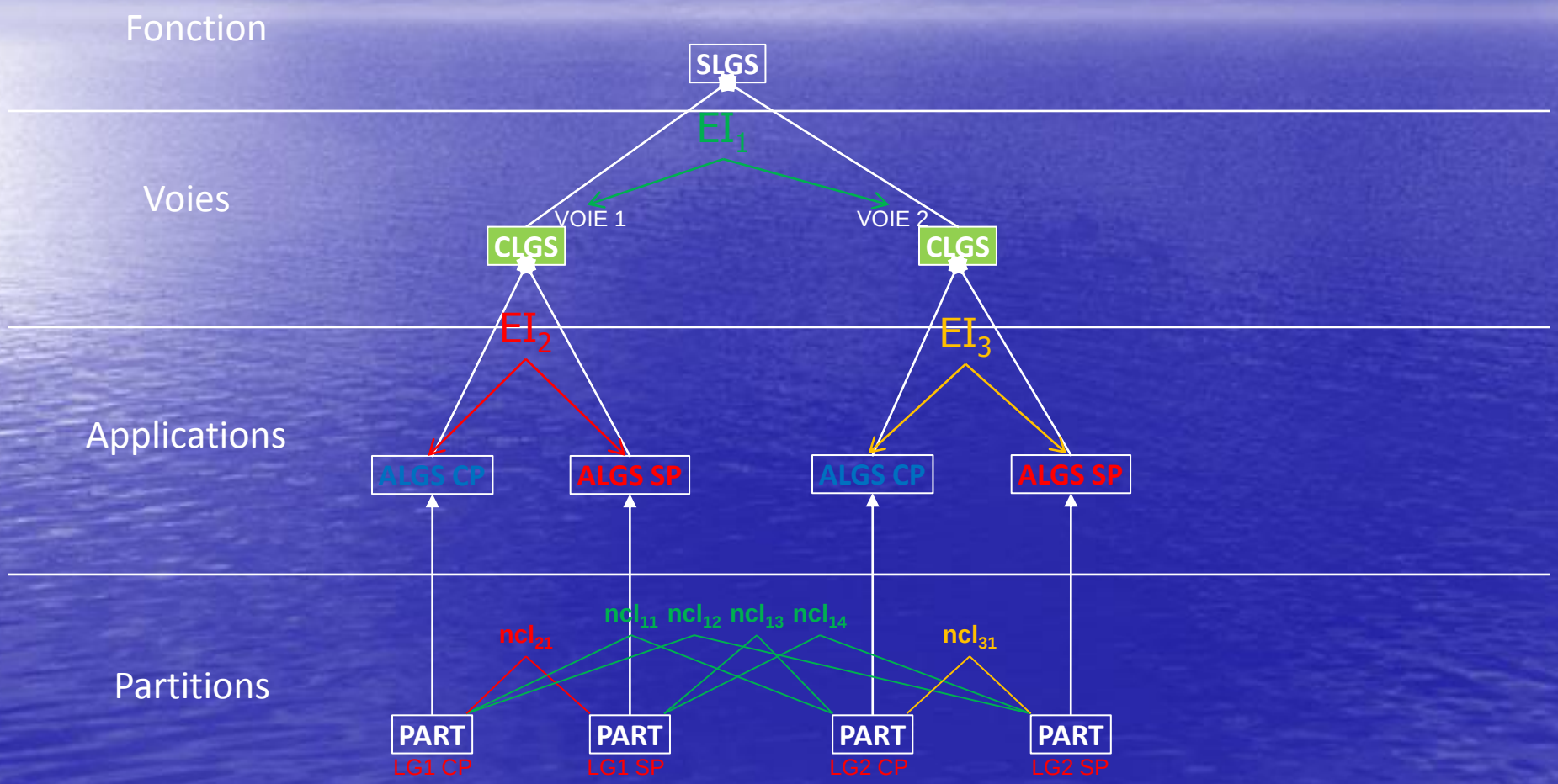
LGS : modélisation des exigences



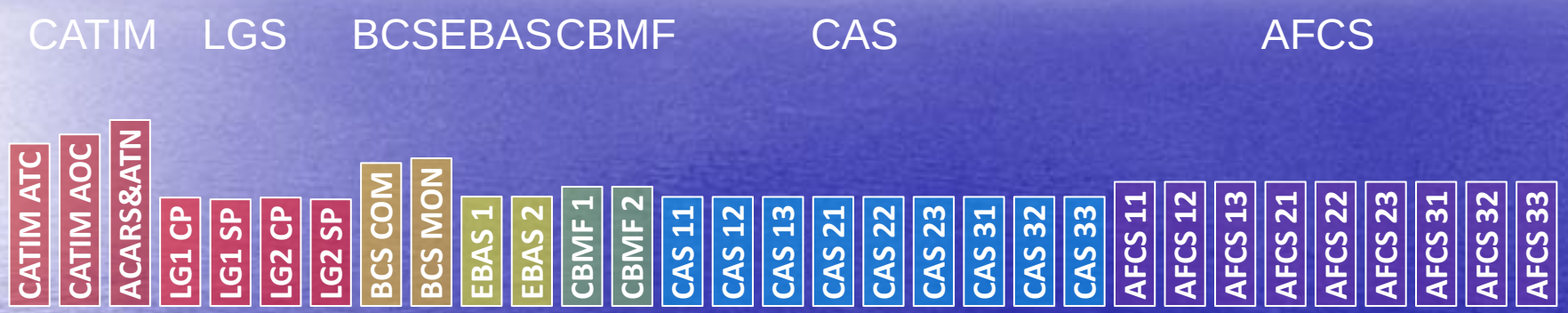
LGS : modélisation des exigences



LGS : contraintes de déploiement

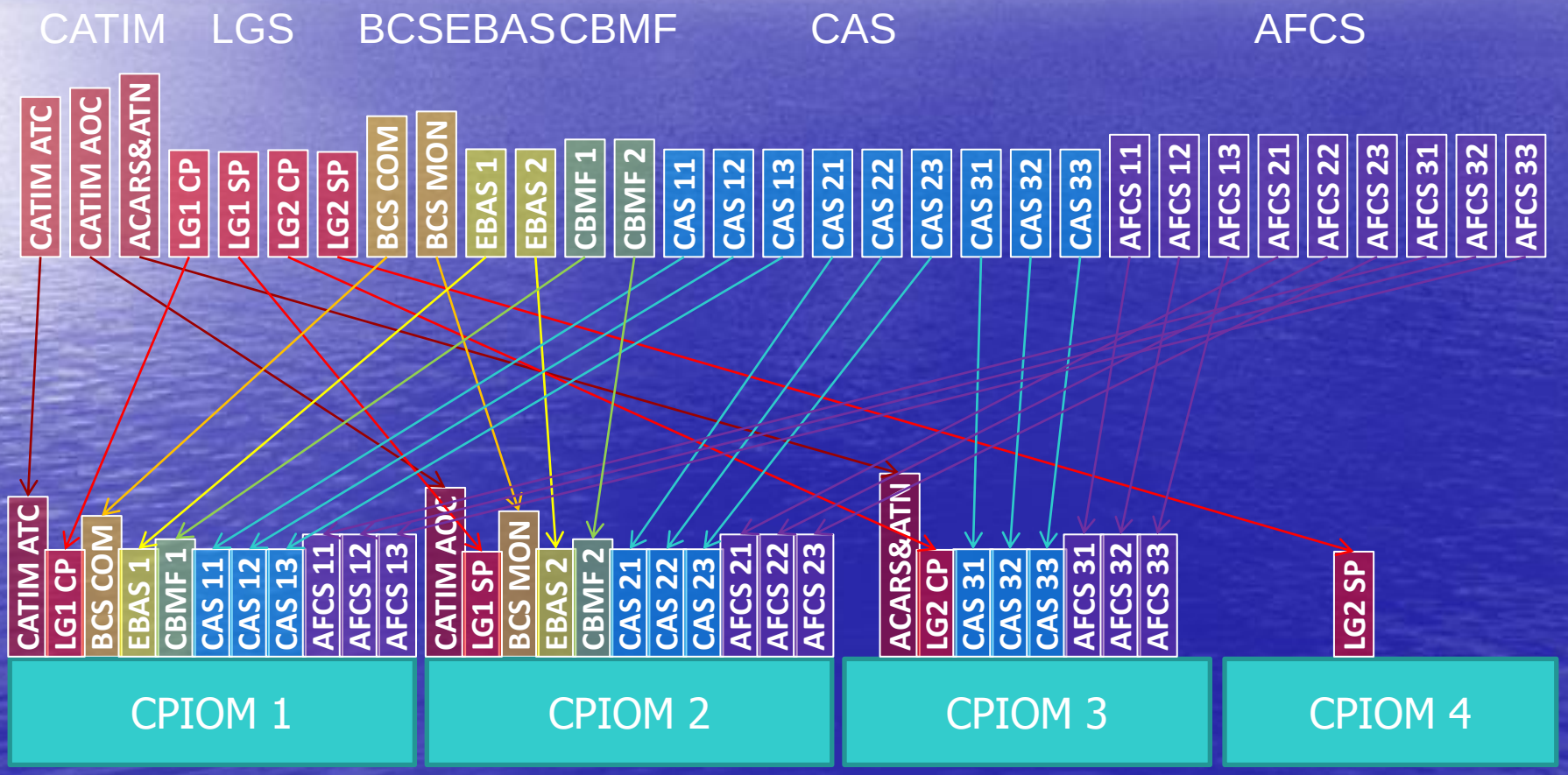


Démonstration

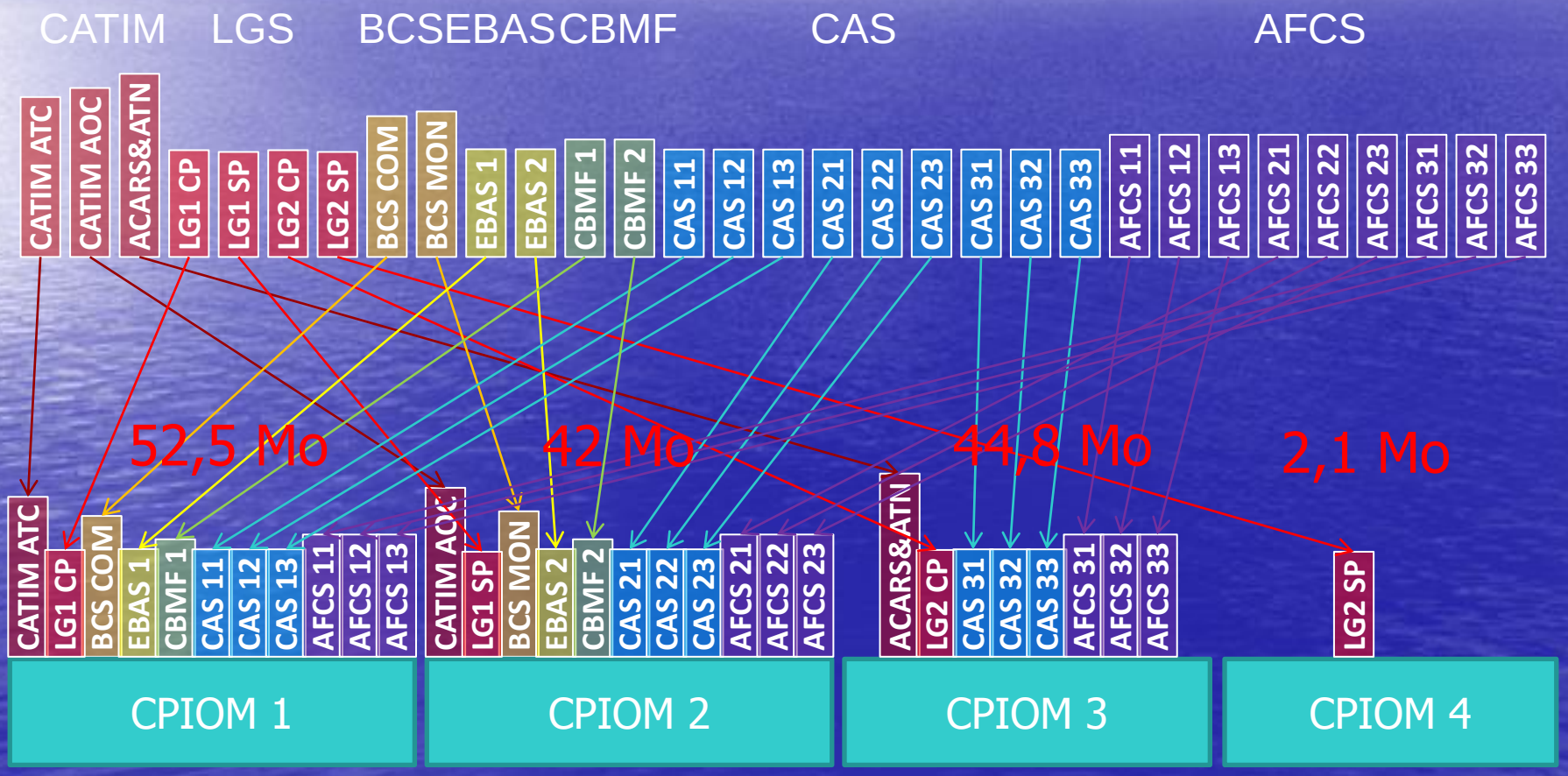


Déploiement de fonctions avion
respectant les contraintes
de Sûreté de fonctionnement

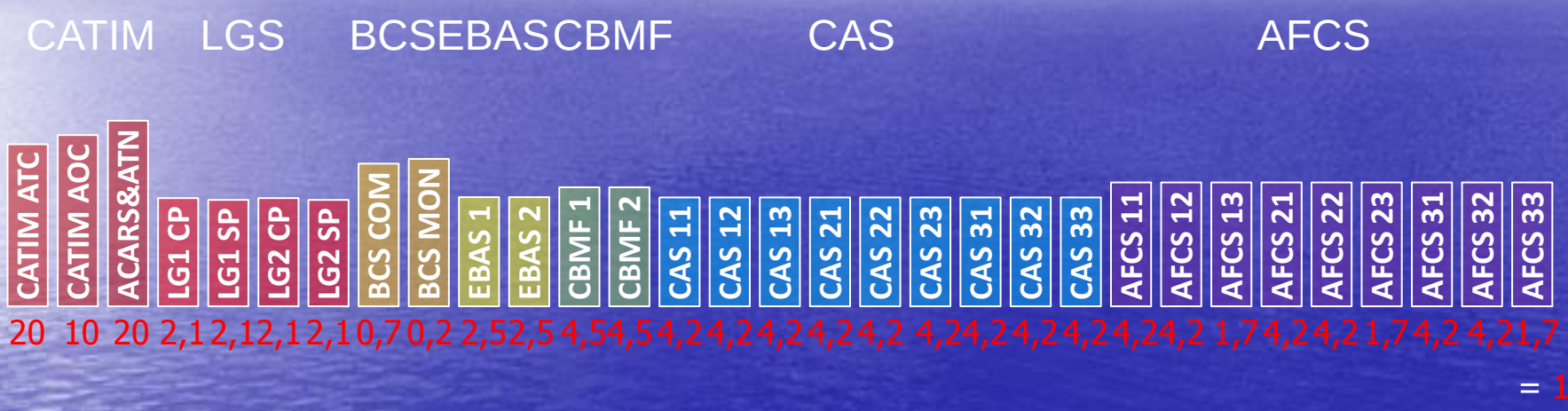
Résultat


CATIM
LGS
BCS
EBAS
CBMF
CAS
AFCS
CATIM ATC
CATIM AOC
ACARS&ATN
LG1 CP
LG1 SP
LG2 CP
LG2 SP
BCS COM
BCS MON
EBAS 1
EBAS 2
CBMF 1
CBMF 2
CAS 11
CAS 12
CAS 13
CAS 21
CAS 22
CAS 23
CAS 31
CAS 32
CAS 33
AFCS 11
AFCS 12
AFCS 13
AFCS 21
AFCS 22
AFCS 23
AFCS 31
AFCS 32
AFCS 33
CPIOM 1
CPIOM 2
CPIOM 3
CPIOM 4

Répartition des charges

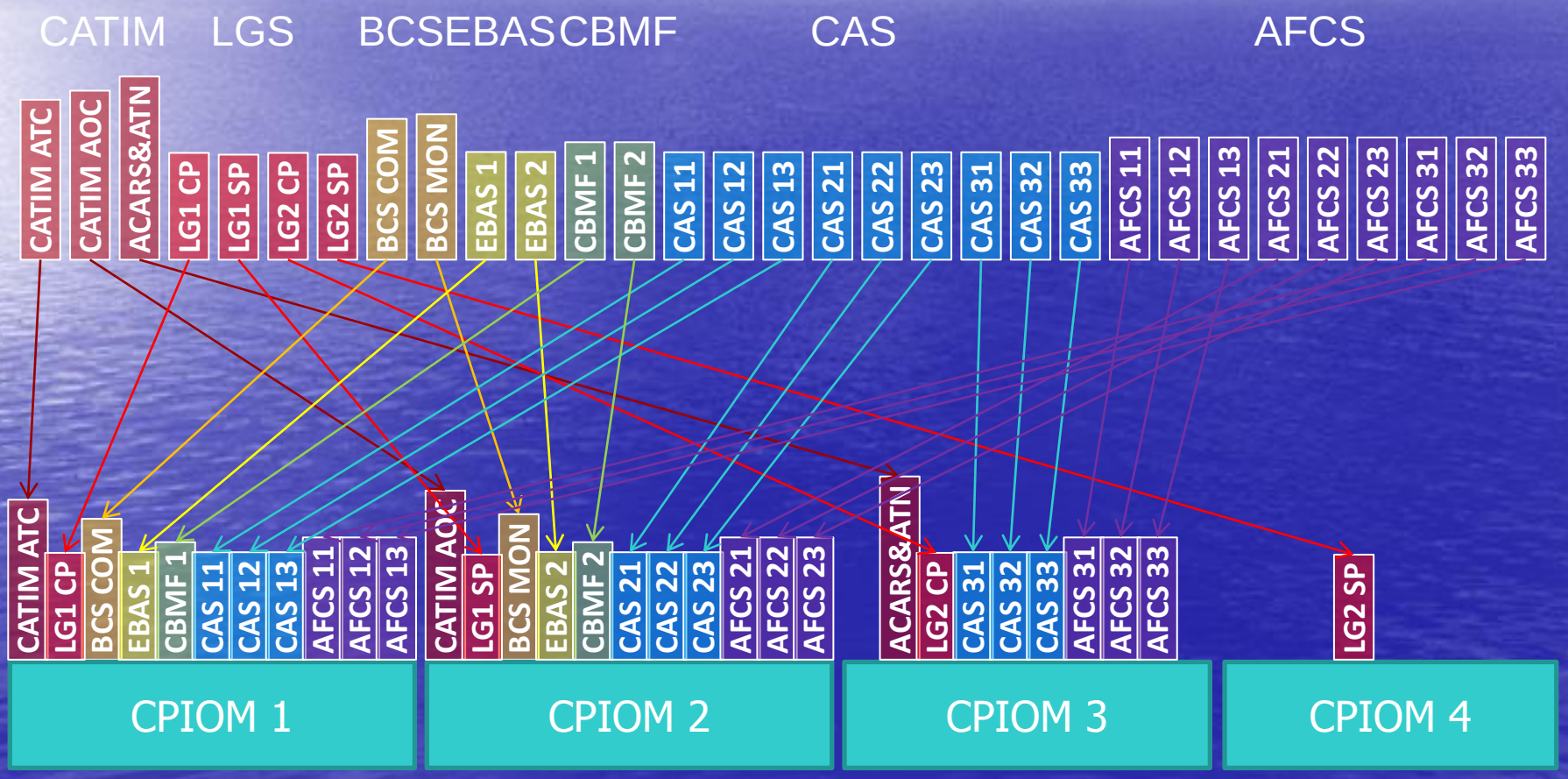


Démonstration



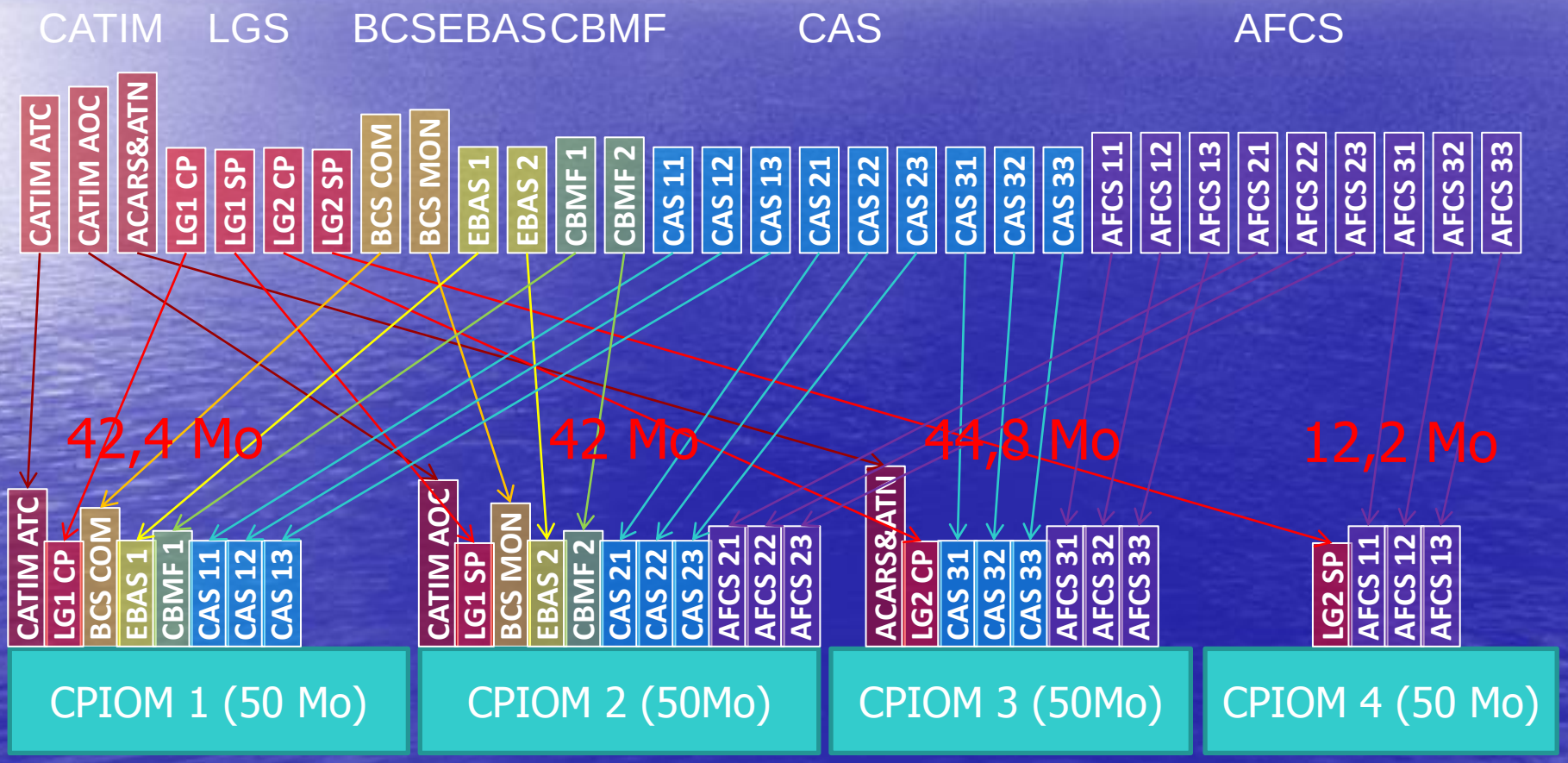
Déploiement des fonctions avion
Respectant les contraintes
- de Sûreté de fonctionnement
- de capacité mémoire (RAM) des CPIOMS

Sans contraintes de capacité



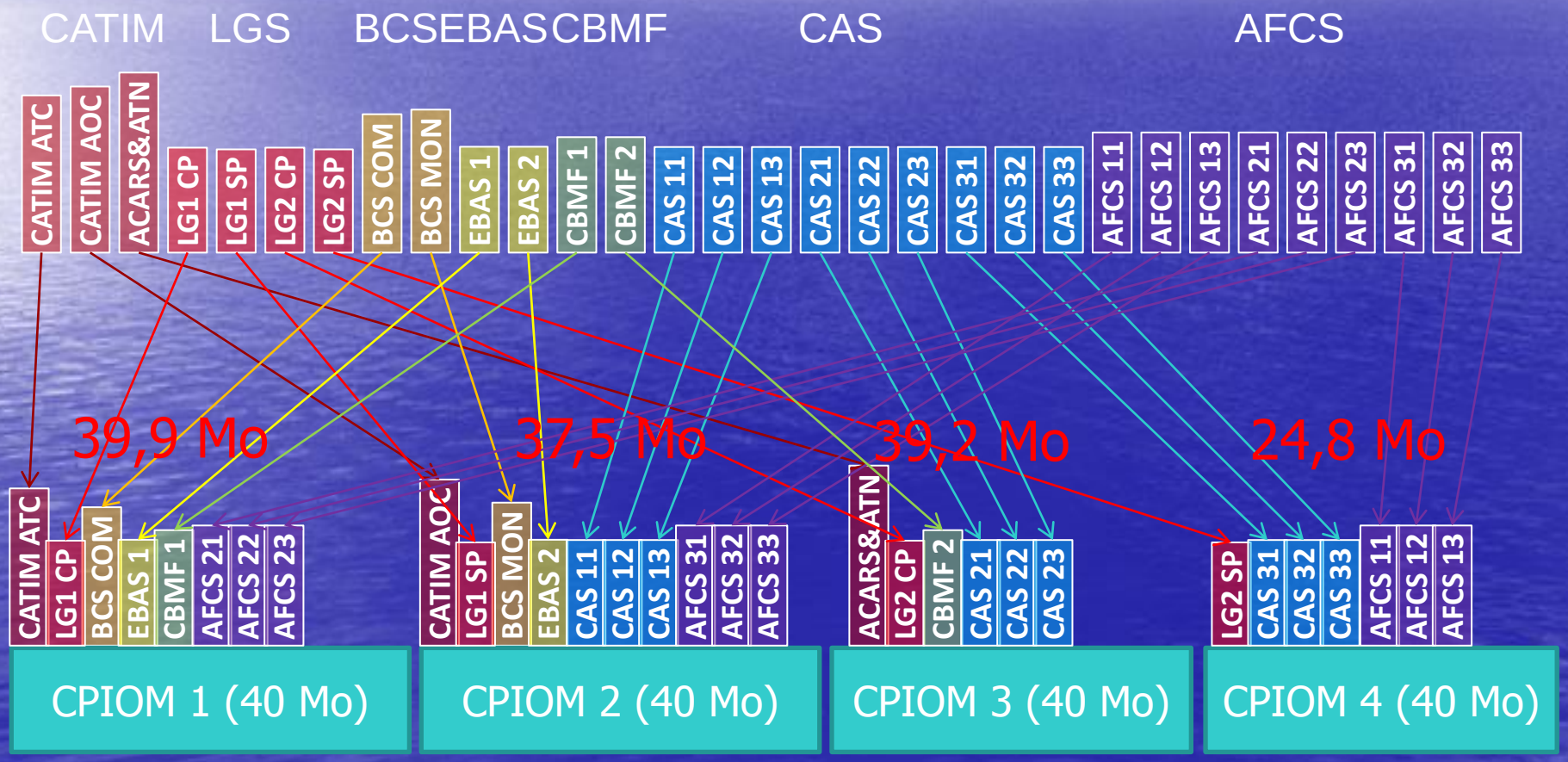
Avec contraintes de capacité

50 Mo

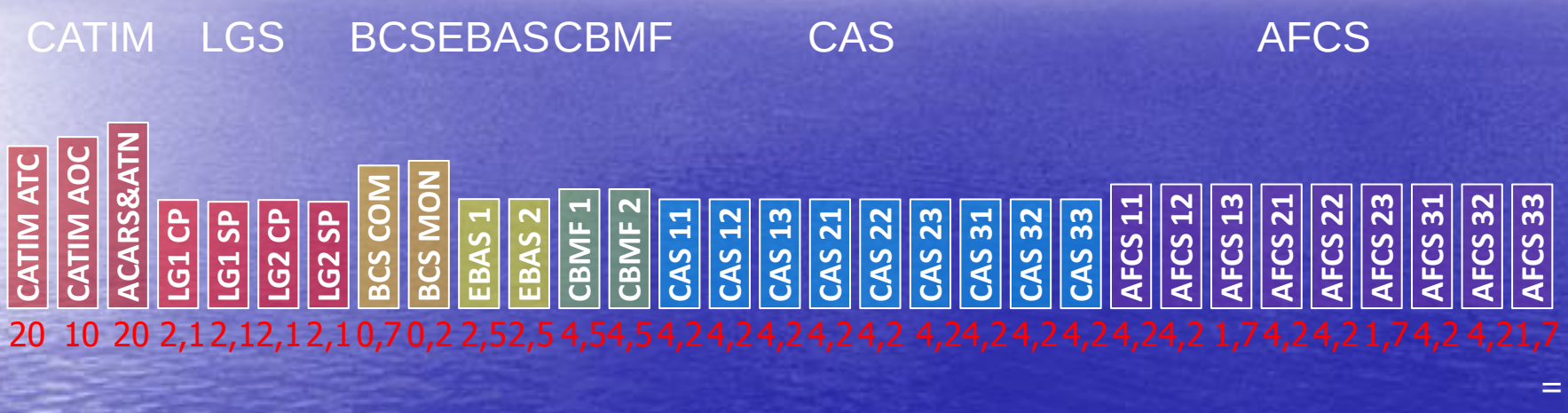


Avec contraintes de capacité

40 Mo



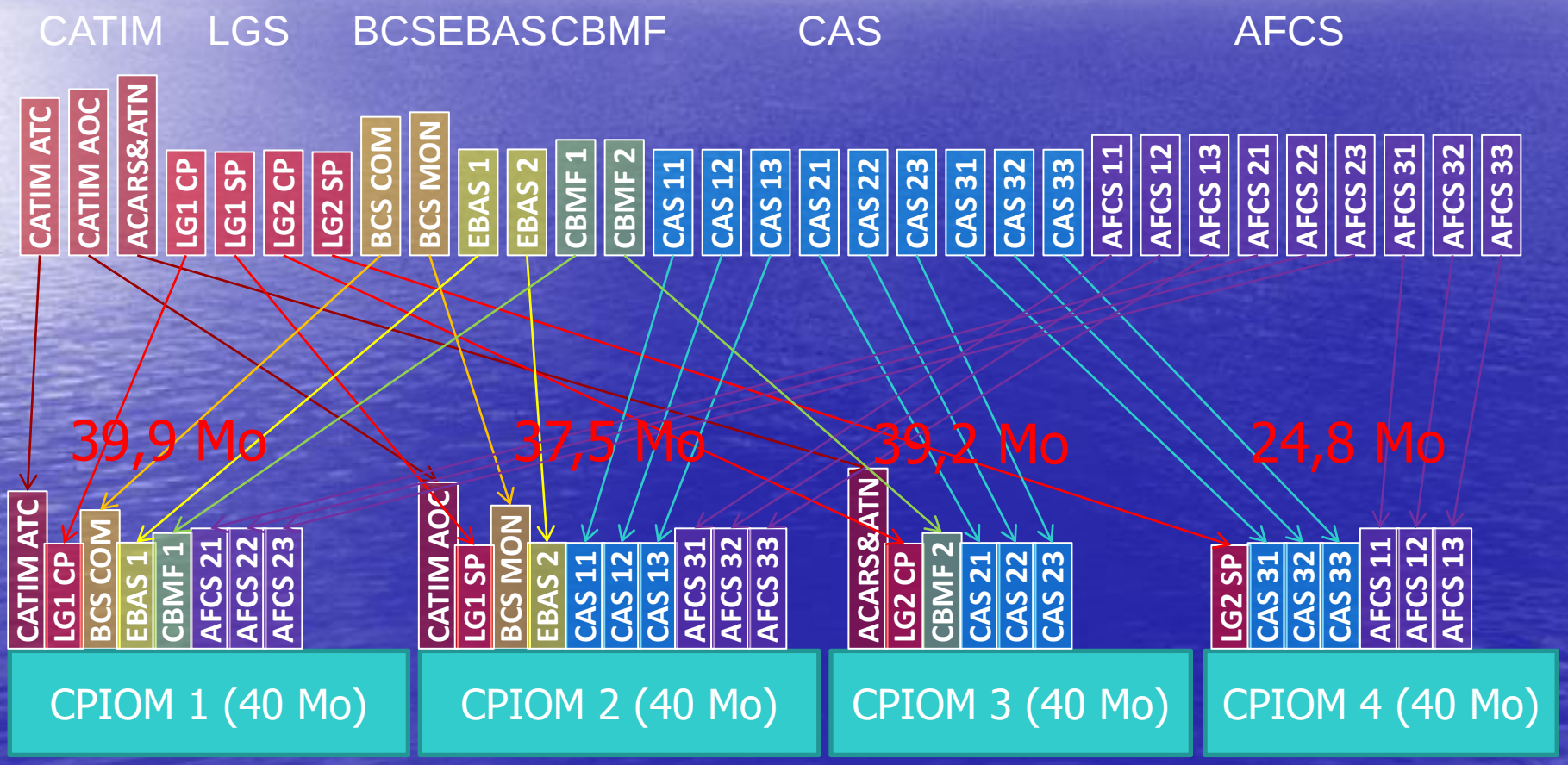
Démonstration



Déploiement des fonctions avion respectant les contraintes :

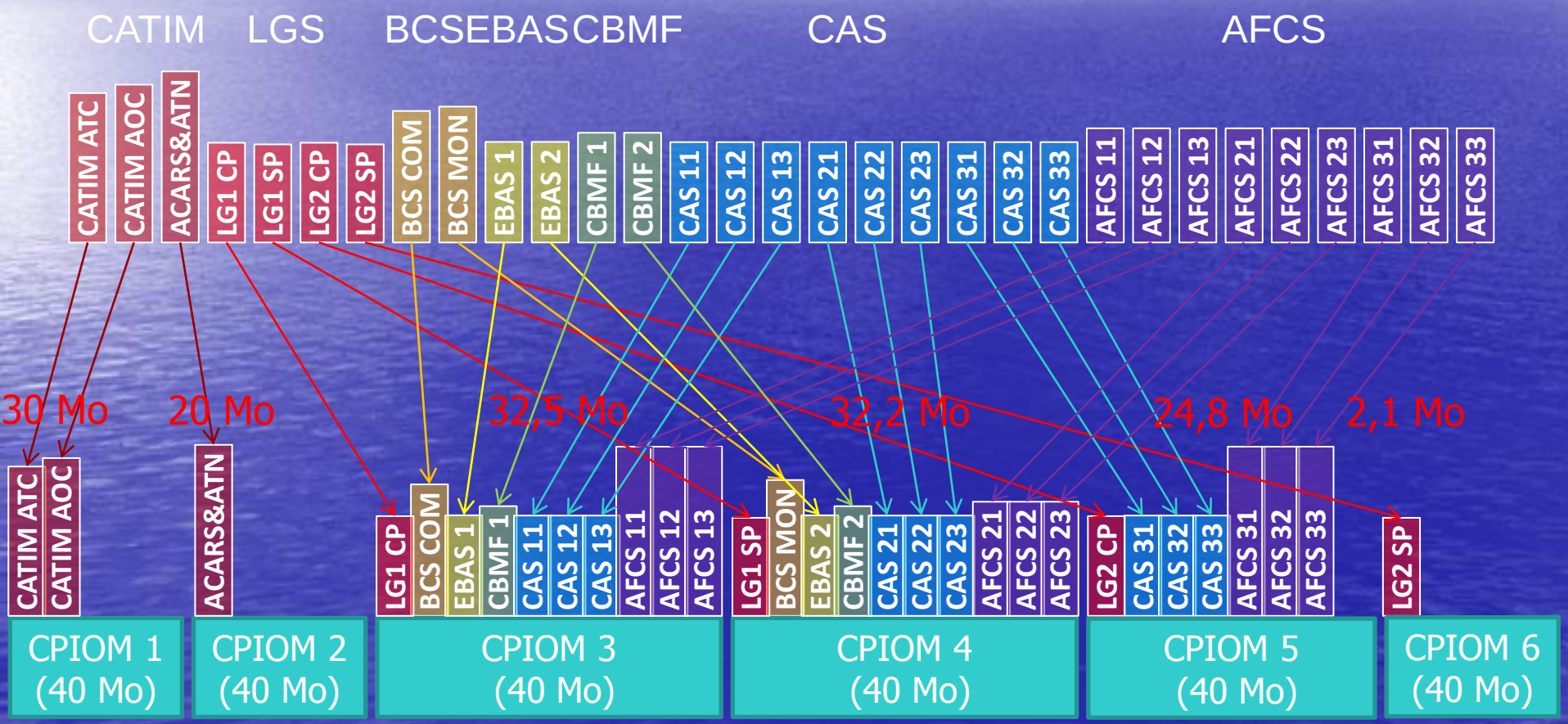
- de Sûreté de fonctionnement
- de Sécurité (Ségrégation matérielle de CATIM des autres systèmes)
- de capacité mémoire (RAM) des CPIOMS

Sans contraintes de sécurité



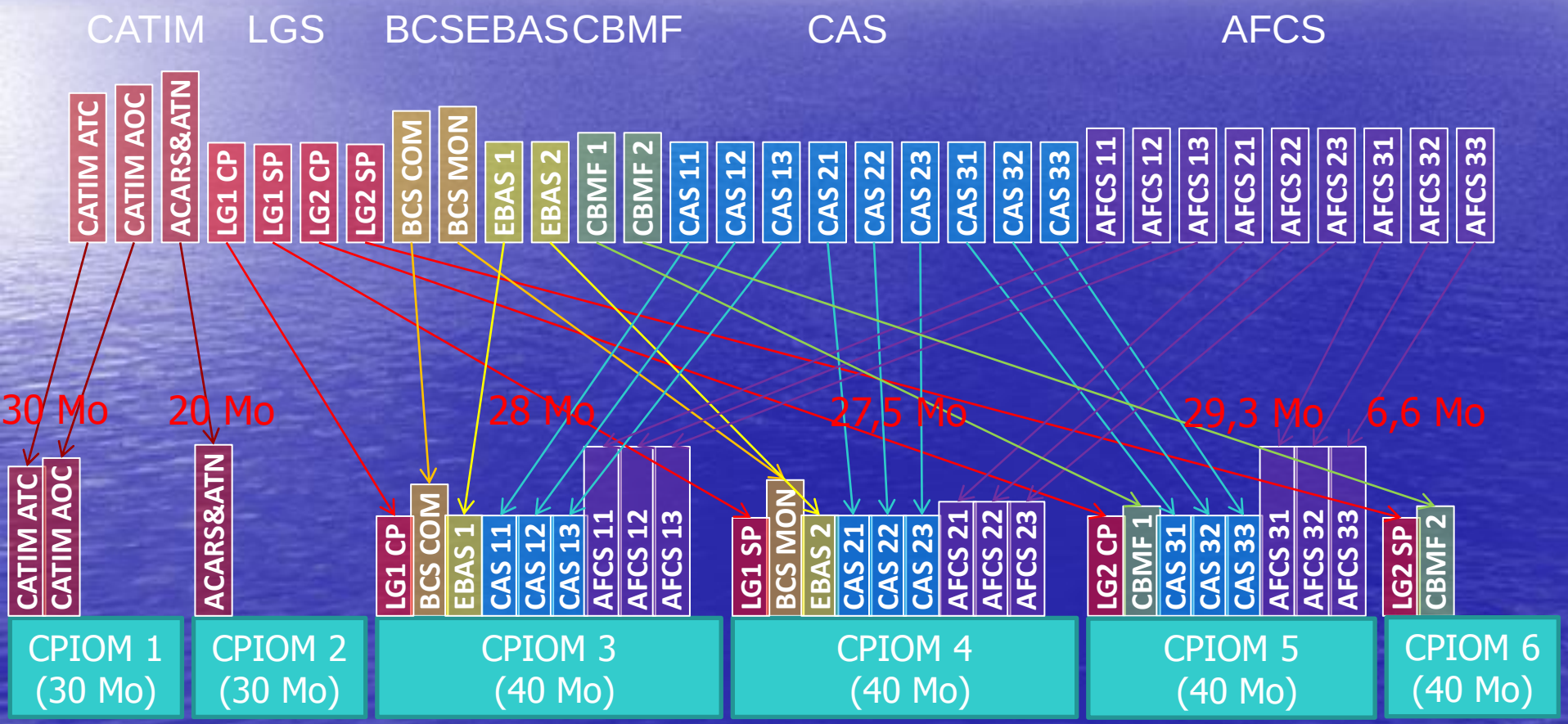
Avec contraintes de sécurité

et CPIOM de 40 Mo

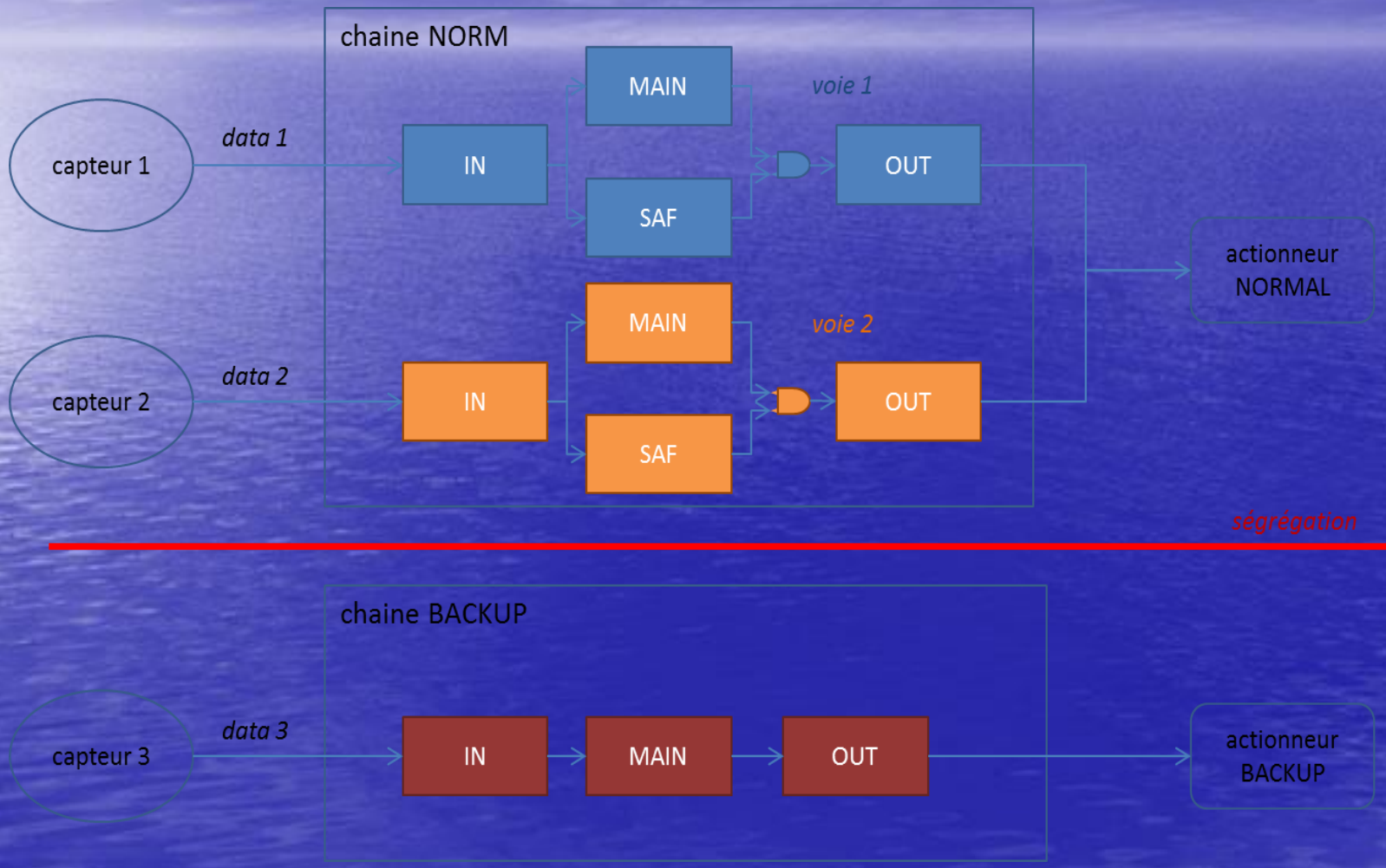


Avec contraintes de sécurité

et CPIOM de 30 Mo



Évolutions du modèle



Évolutions du modèle

- Chaînes de traitement complètes : capteurs, communications, actionneurs, alimentation électrique
- et encore :
  - Safety (FDAL, IDAL)
  - Security (SAL) …
- et aussi (pêle-mêle) : latences, compatibilité de fréquences, coûts, contraintes physiques (poids, encombrement), puissance électrique consommée, ségrégation matérielle, ségrégation des alimentations, fournisseurs …

Bilan DEPS/IMA

On dispose d'un formalisme de conception de haut niveau en rupture avec l'existant

- on peut envisager un modèle déclaratif d'architecture système, utilisable pendant tout son cycle de vie :
  - Génération d'architectures
  - Dimensionnement, Vérification d'une d'architecture
  - Modification incrémentale d' une architecture
    - Nouvelles fonctions avion, applications …
    - nouveaux composants
  - … Certification de l'architecture

Outlines

- Context
- The DEPS project
- The DEPS language
- The DEPS solver
- DEPS by example
- Use-case IMA
- Ongoing studies and developments

Etudes et développements en cours (1)

- Développement d'une nouvelle version de DEPS (L. Zimmer, P.A. Yvars)
  - enrichissement de l'ontologie et des types de base
  - collecteurs de modèles
  - sélecteurs de modèles
  - contraintes « catalogue »

Etudes et développements en cours (2)

- Coopération avec Dassault Aviation
  - Vérification de systèmes de génération et de distribution électrique
    - projet de synthèse SupMéca
- Portage de modèles CE
  - Pile à Combustible
  - Transmission de puissance à 3 étages
  - Robot
  - …

Etudes et développements en cours (3)

- Synthèse pour la conception en génie électrique

- Coopération :
  - Roberval (ex LEC) : A. Hubert (UTC), Y. Meyer (UTBM)
  - Quartz – SupMéca : P.A. Yvars
  - Dassault Aviation : L. Zimmer

- Thèse UTC de Séphora Diampovesa
  « modèles de synthèse pour la conception optimale en génie électrique – Application à l'électrification des véhicules »
  - débutée en novembre 2017
  - codirection A.Hubert (UTC), P.A. Yvars (SupMéca)

Informations

- Le langage DEPS est supporté par l'association *DEPSLink*
- www.depslink.com

- Contacts

- P.A. Yvars pierre-alain.yvars@supmeca.fr
- L. Zimmer Laurent.Zimmer@dassault-aviation.com

Some References

- DEPS

- P.A. Yvars, L. Zimmer, *"DEPS Un langage pour la spécification de problèmes de conception de Systèmes"*, proc of the 10th International Conference on Modeling, Optimization & SIMulation (MOSIM 2014), France

- L.Zimmer, M. Lafaye, P.A. Yvars, "Modélisation d'exigences pour la synthèse d'architecture avionique : Application à la sûreté de fonctionnement", 16eme journées AFADL, Approche Formelle dans l'Assistance au Développement de Logiciel, Montpellier, 2017.

- Constraint programming for design synthesis

- P.A. Yvars, L. Zimmer, *"System sizing with a model-based approach: application to the optimization of a power transmission system",* Mathematical Problem in Engineering, Vol 2018, july 2018.

- S. Diampovesa, A. Hubert, P.A. Yvars,Y. Meyer, L. Zimmer, "*Optimal Design for Electromagnetic Devices. A Synthesis Approach using Intervals and Constraint-based Methods*", 15th International Workshop on Optimization and Inverse Problems in Electromagnetism, September 2018, Hall in Tirol, Austria

- A.Hubert, P.A. Yvars, Y.Meyer, L. Zimmer, *"Conception préliminaire optimale des systèmes électriques. Une approche par synthèse"*, 2ème Conférence symposium de Génie Electrique, SGE2016, Grenoble, 2016.

- Y. Meyer, P.A. Yvars, T. Verdot, *"Systemic optimization of an active vibration micro-isolator: an interval computation and constraint propagation based approach",* IEEE AIM 2014 International Conference on Advanced Intelligent Mechatronics, Besançon, 20 janvier 2014.

- Y. Meyer, P.A. Yvars, "*Optimization of a passive structure for active vibration isolation: an interval computation and constraint propagation based approach*", Engineering Optimization, Volume 44, Issue 12, December 2012

- P.A. Yvars, P. Lafon, L. Zimmer, *"Optimization of Mechanical System: Contribution of Constraint Satisfaction Method"*, proc of IEEE CIE'39, International Conference on Computers and Industrial Engineering, Troyes, 6-8 juillet 2009.

Merci pour votre attention