Motifs formels d'architectures de systèmes pour la sûreté de fonctionnement

Christophe KEHREN

Encadrants

André ARNOLD (LaBRI)
Antoine RAUZY (IML)
Christel SEGUIN (ONERA)

20 décembre 2005

ONERA

But

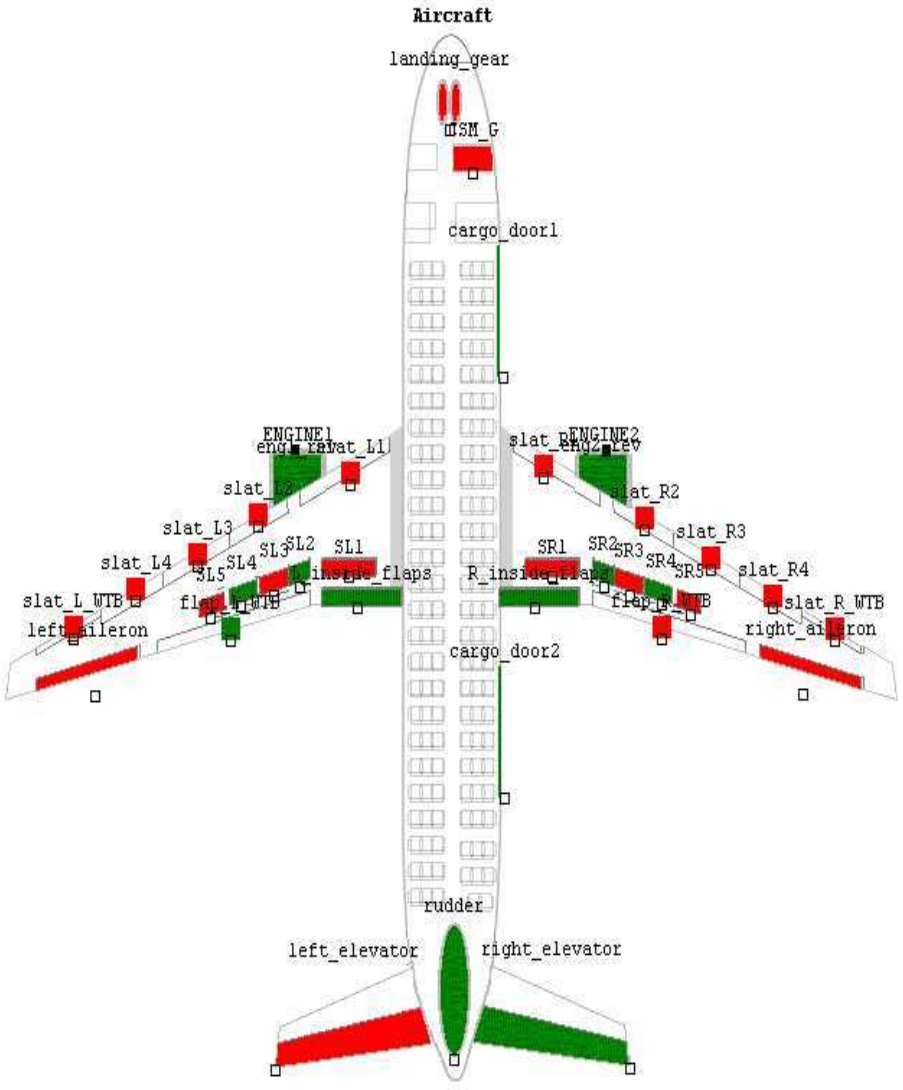
Objectif principal de cette thèse

Assiter l'évaluation de la sûreté de fonctionnement de systèmes embarqués ...

- en phase amont de conception d'architectures
- vis à vis d'exigences qualitatives

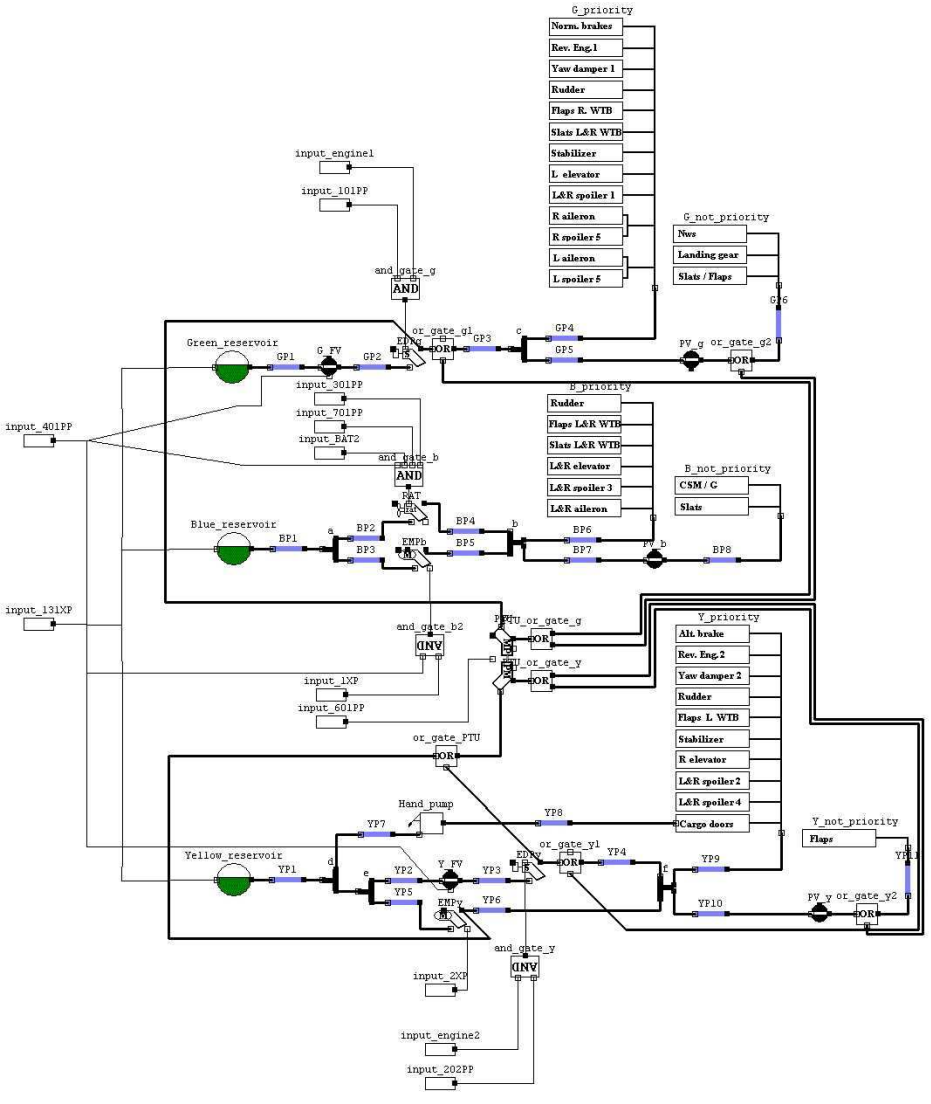
Des systèmes ...

1. Hétérogènes



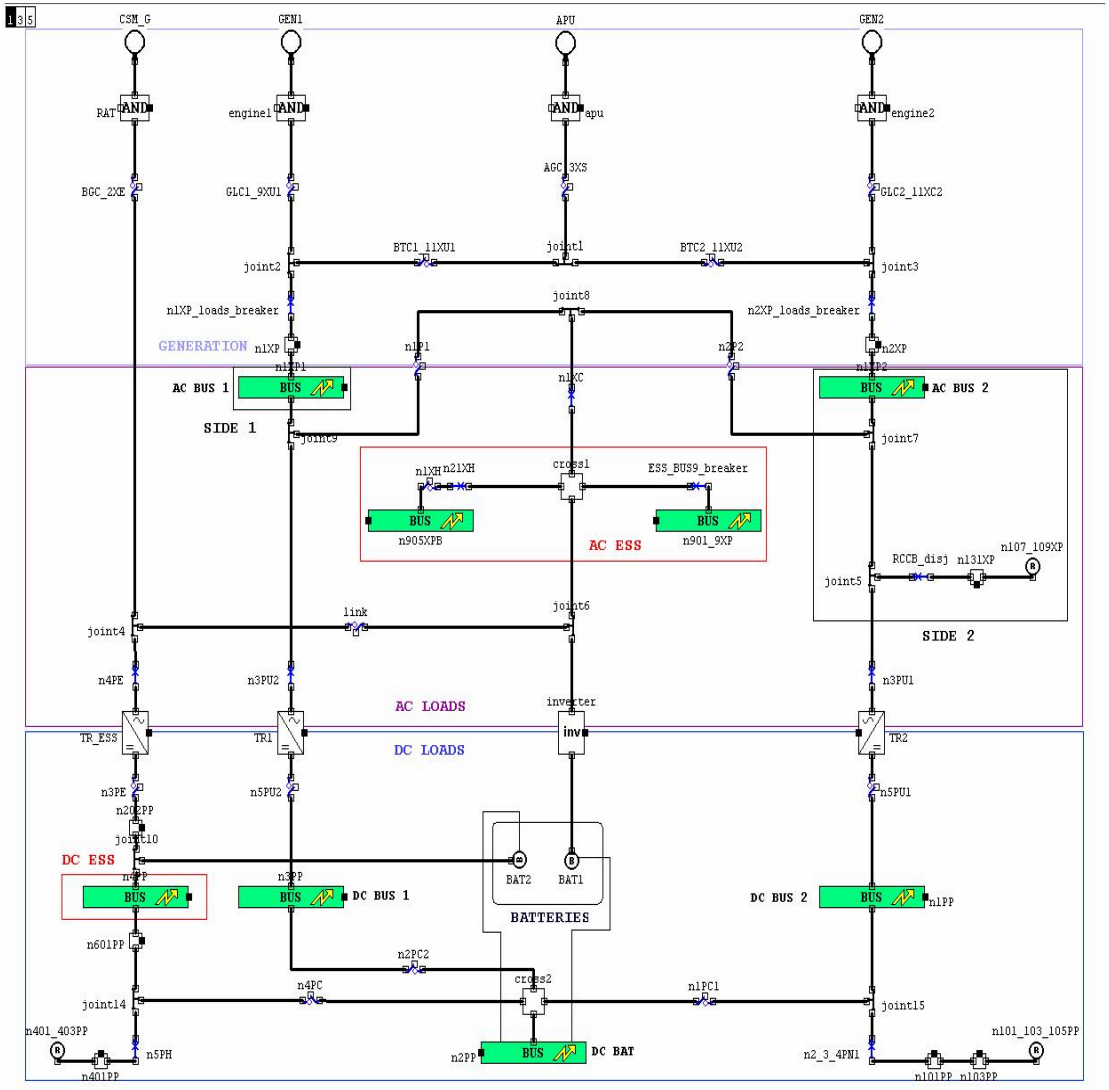
Des systèmes ...

2. Critiques (= fortement redondés)



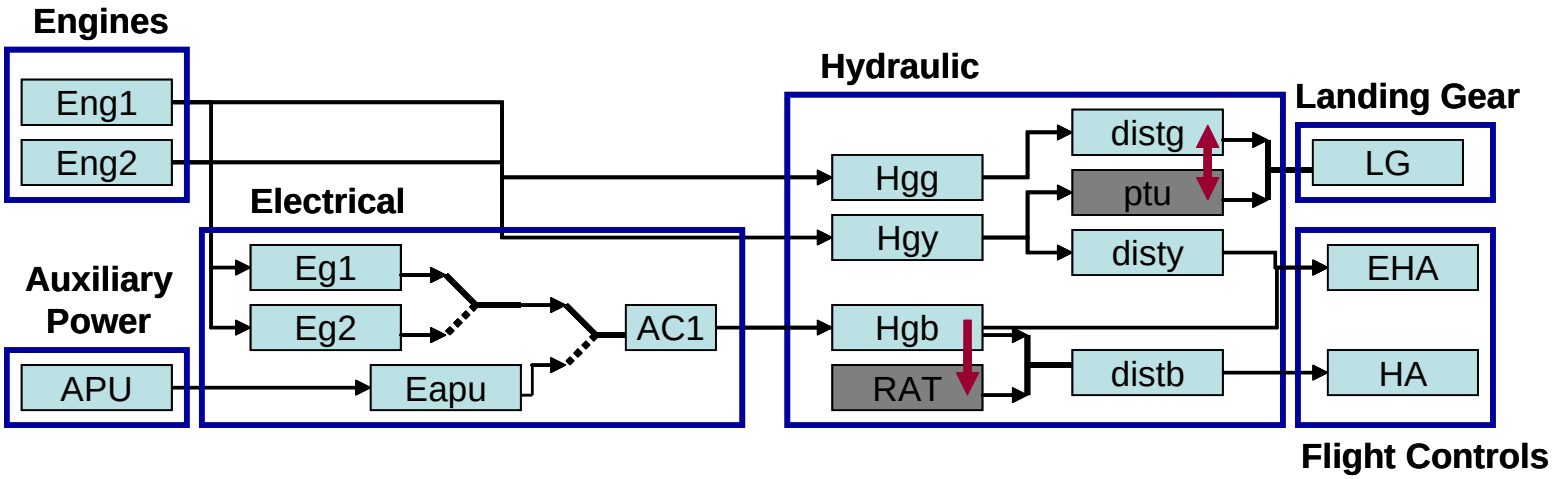
Des systèmes ...

3. Dynamiques (= nombreuses reconfigurations)



Exemple de systèmes

Vue de l'analyste : schéma simplifié multi-systèmes



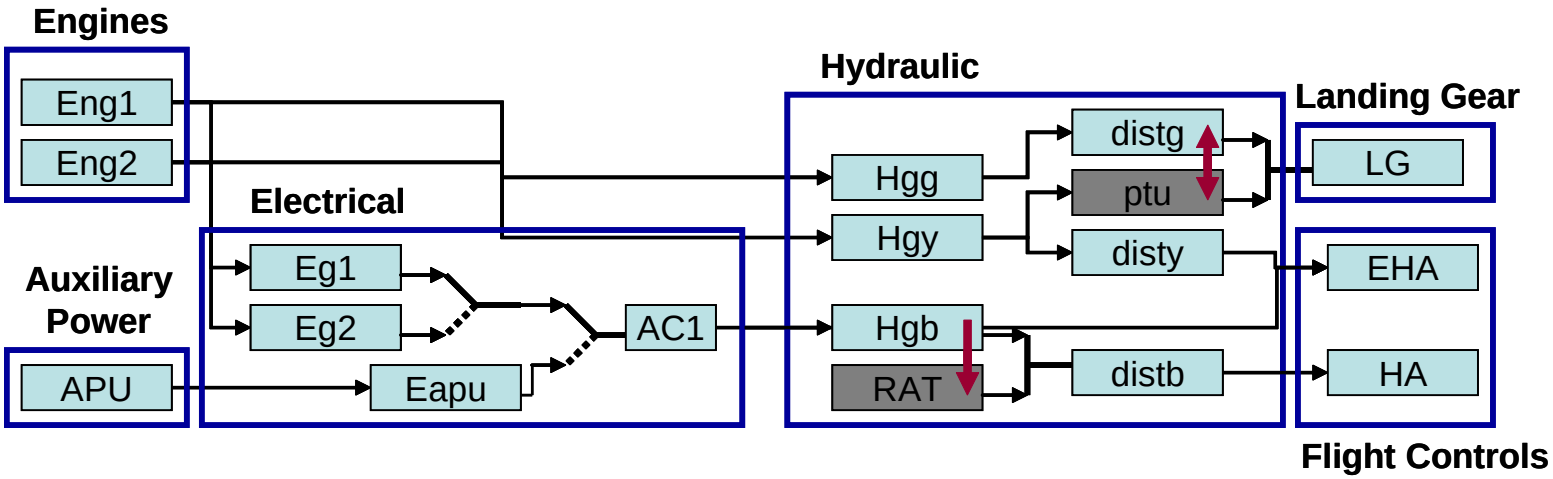
Des morceaux d'architectures ...

- ayant des caractéristiques identiques
  - structures récurrentes
  - services fournis récurrents

= motifs d'architectures de sûreté

Exemple de systèmes

Vue de l'analyste : schéma simplifié multi-systèmes



Des morceaux d'architectures ...

- ayant des caractéristiques identiques
  - structures récurrentes
  - services fournis récurrents

= motifs d'architectures de sûreté

Objectifs

Fournir les moyens de capitaliser

- les solutions architecturales de sûreté récurrentes et matures (gain de temps en conception)

Formaliser ces solutions

- identifier les concepts (architectures et propriétés types)
- proposer une notation
- proposer des stratégies d'évaluation d'architectures

Formaliser l'utilisation de ces solutions

- réutilisation de ces solutions dans une architecture
- reconnaissances de ces solutions
- etc.

Objectifs

Fournir les moyens de capitaliser

- les solutions architecturales de sûreté récurrentes et matures (gain de temps en conception)

Formaliser ces solutions

- identifier les concepts (architectures et propriétés types)
- proposer une notation
- proposer des stratégies d'évaluation d'architectures

Formaliser l'utilisation de ces solutions

- réutilisation de ces solutions dans une architecture
- reconnaissances de ces solutions
- etc.

Objectifs

Fournir les moyens de capitaliser

- les solutions architecturales de sûreté récurrentes et matures (gain de temps en conception)

Formaliser ces solutions

- identifier les concepts (architectures et propriétés types)
- proposer une notation
- proposer des stratégies d'évaluation d'architectures

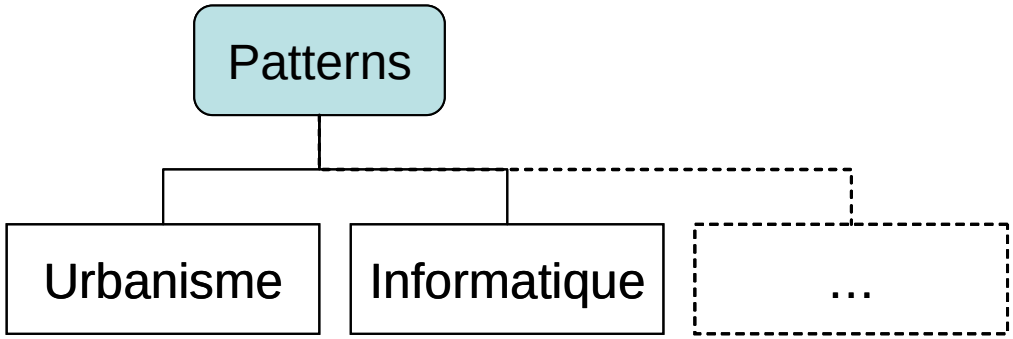
Formaliser l'utilisation de ces solutions

- réutilisation de ces solutions dans une architecture
- reconnaissances de ces solutions
- etc.

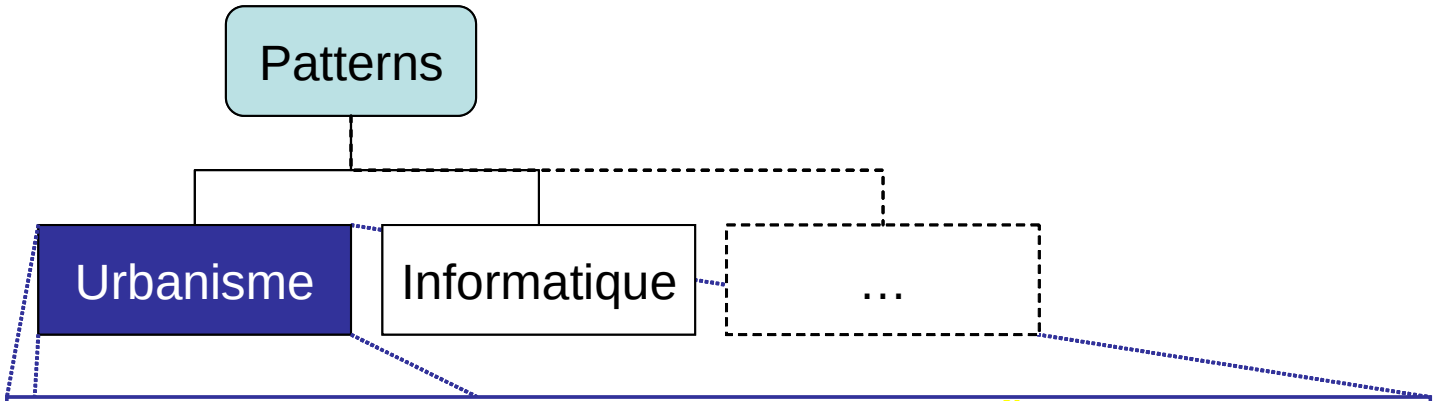
Plan

Plan

Des motifs ailleurs ...



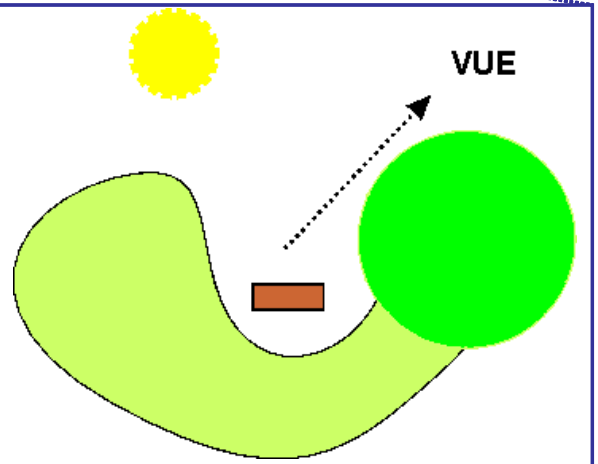
Des motifs ailleurs ...



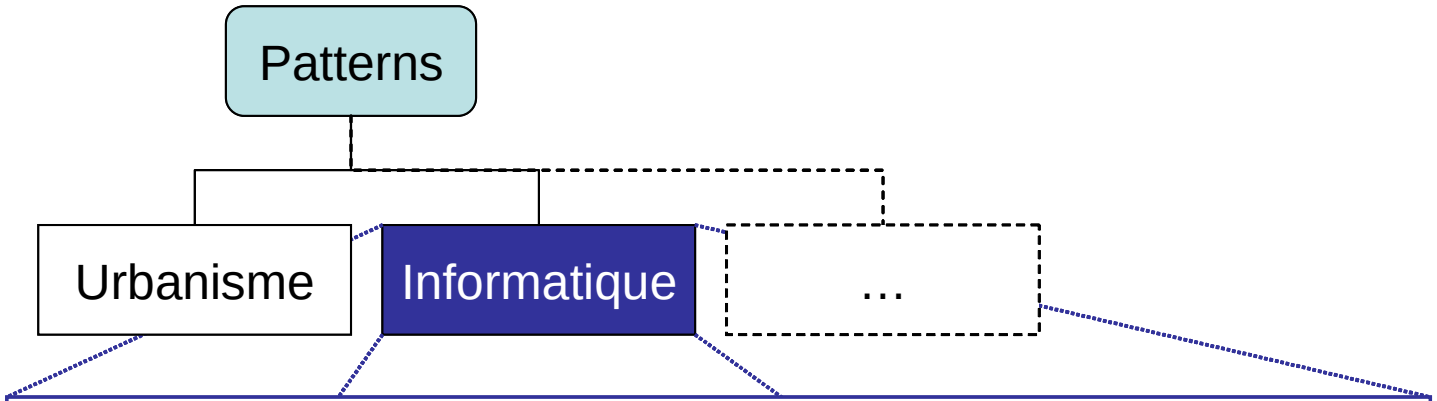
Problème : Trouver l'emplacement d'un banc dans un parc.

Solution : Protéger le banc des intempéries (près d'un arbre) tout en permettant un bon ensoleillement et une belle vue.

etc.



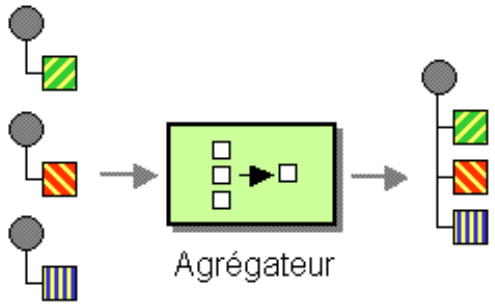
Des motifs ailleurs ...



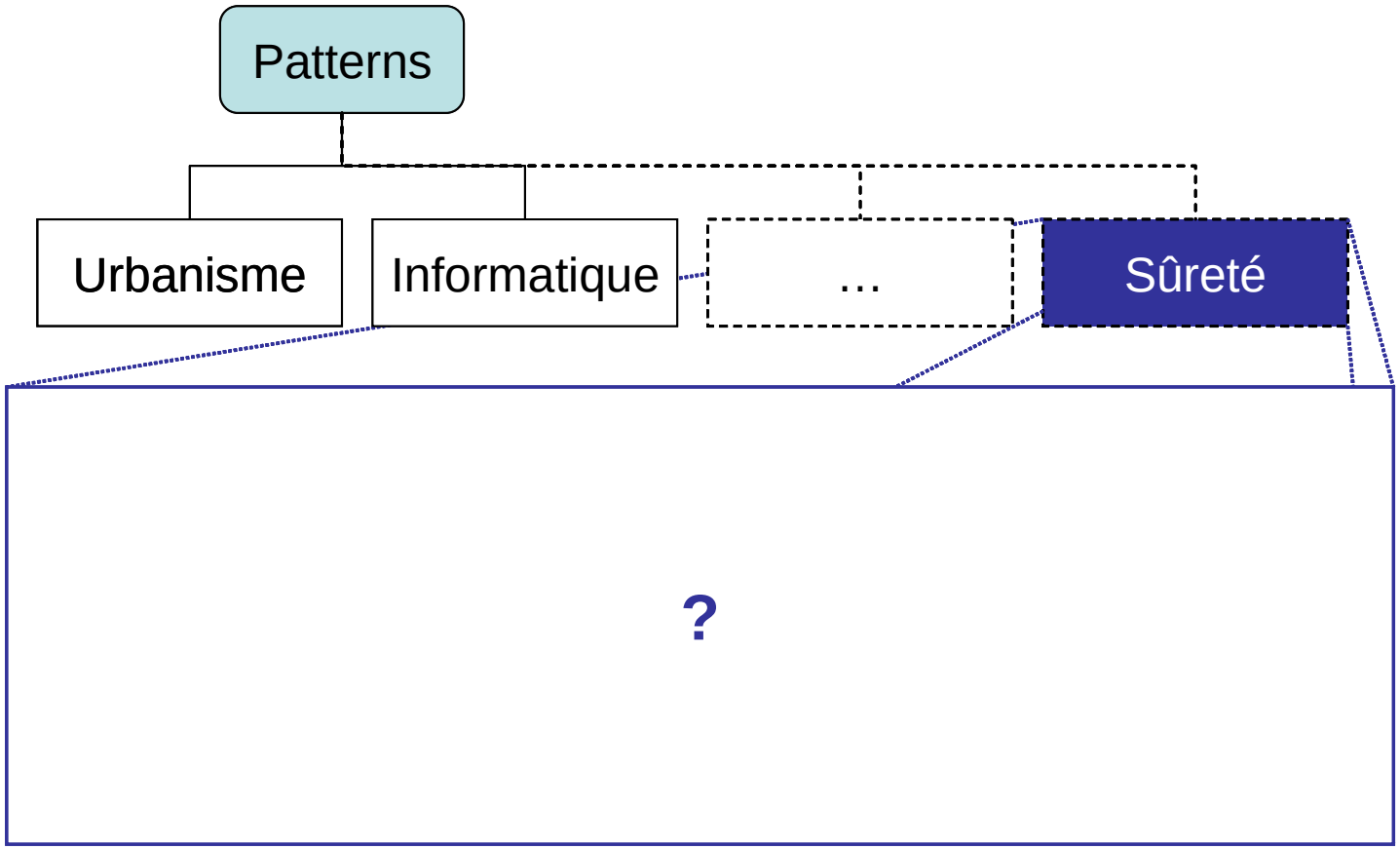
Problème : Comment combiner des messages individuels mais corrélés afin qu'ils soient traités de manière globale.

Solution : Utiliser un filtre pour agréger et stocker différents messages corrélés jusqu'à ce qu'un message complet ait été reçu.

etc.

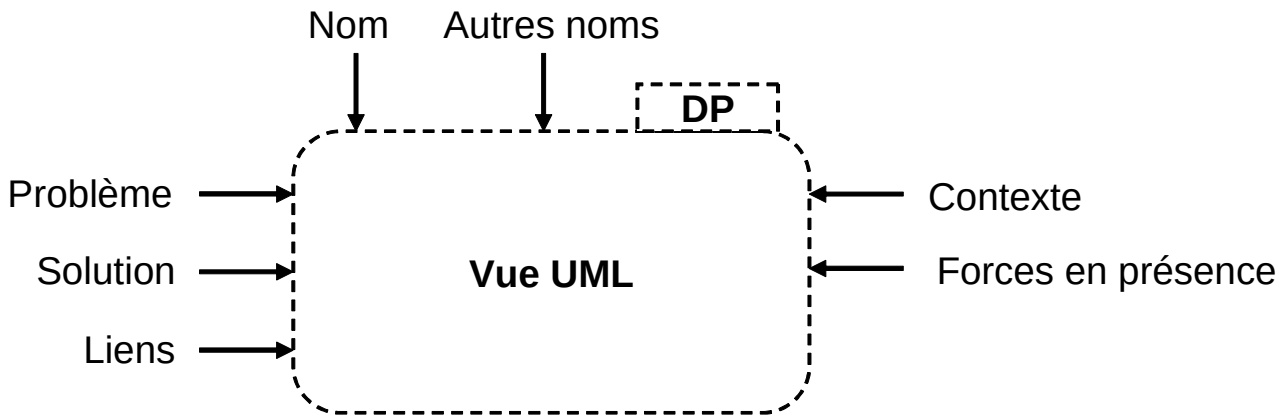


Des motifs ailleurs ...



Un *design pattern*

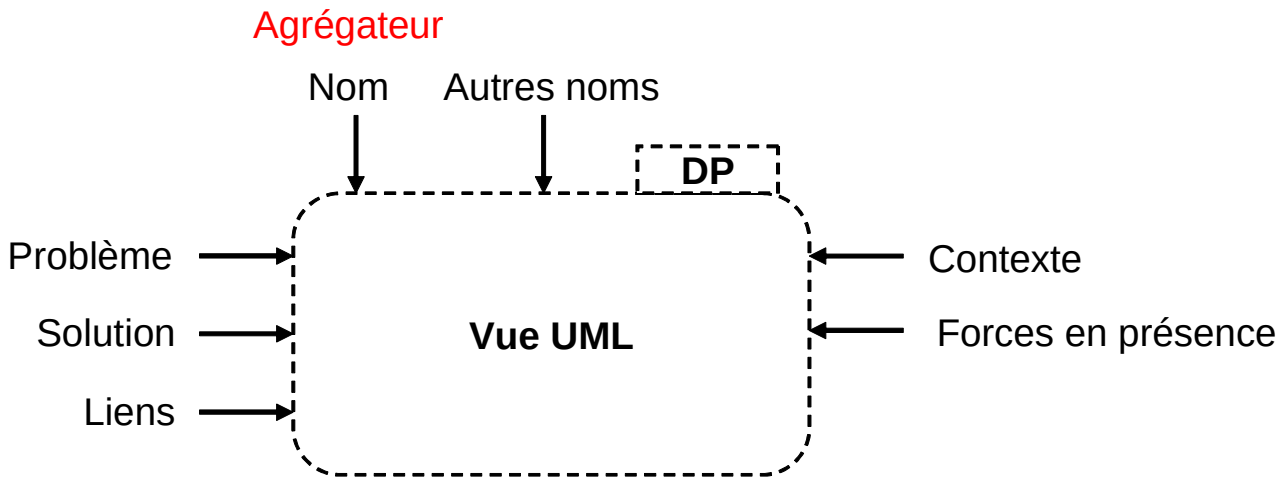
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

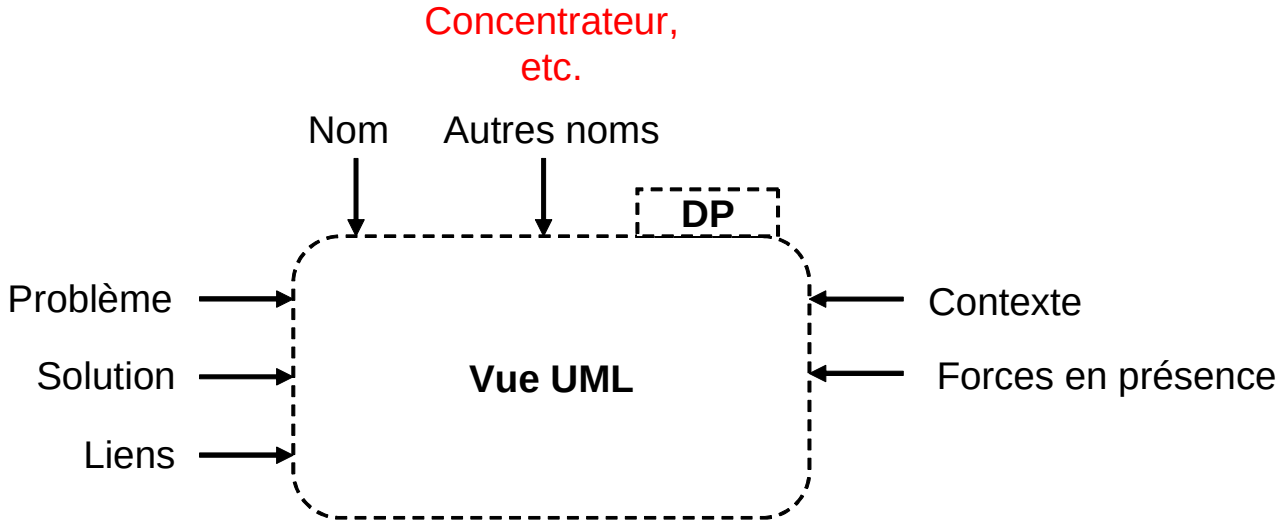
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

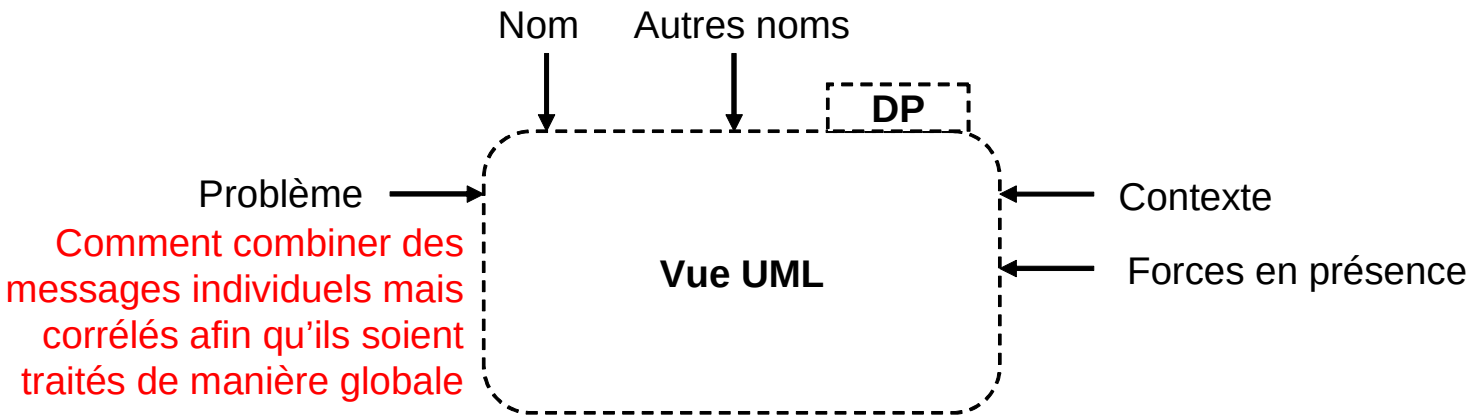
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

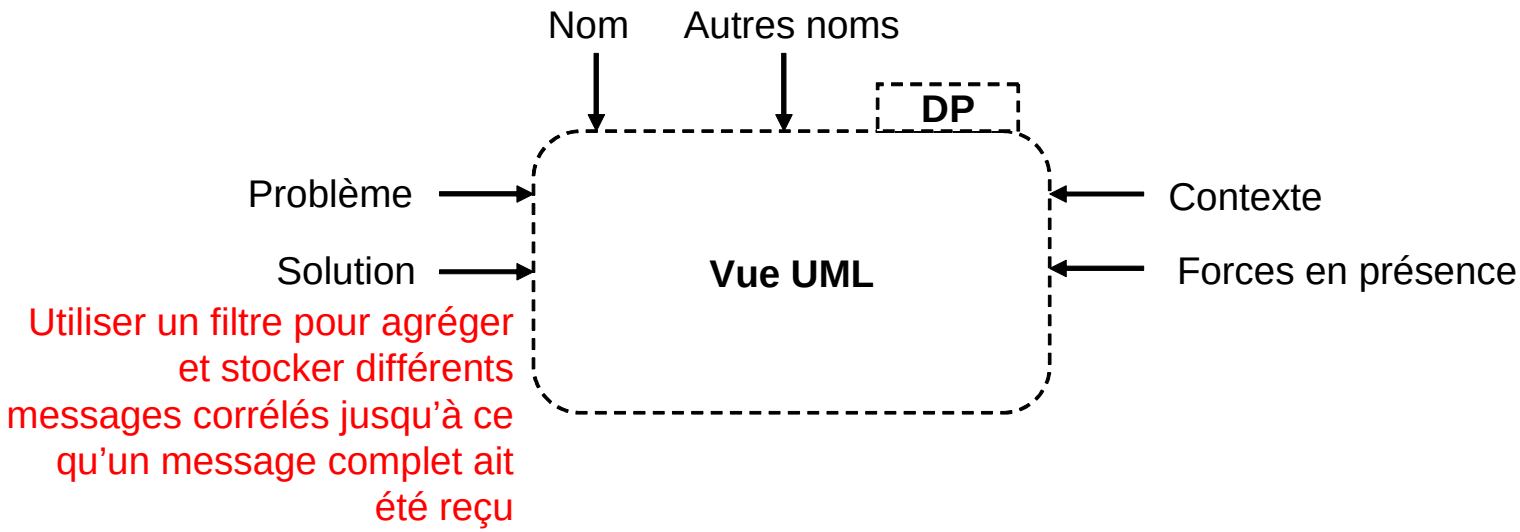
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

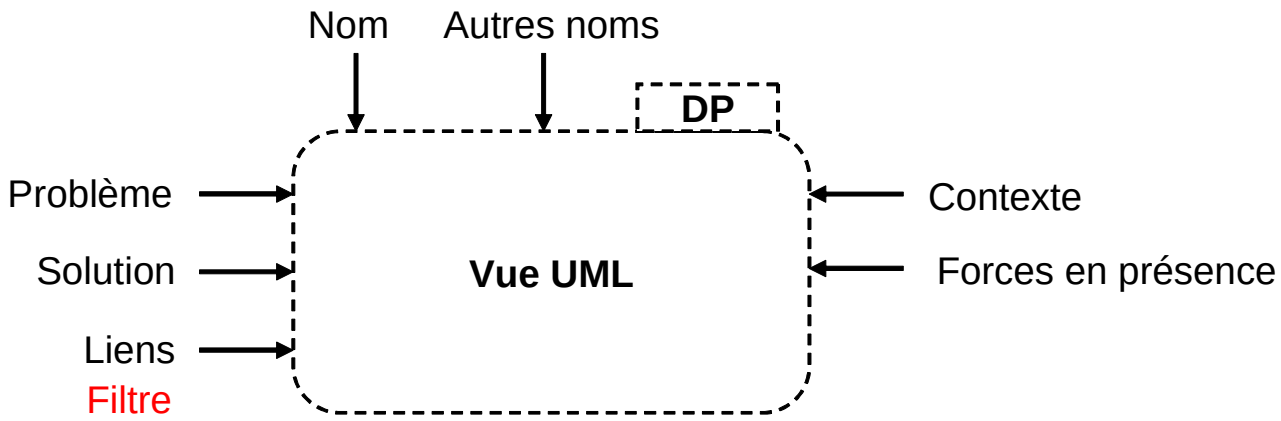
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

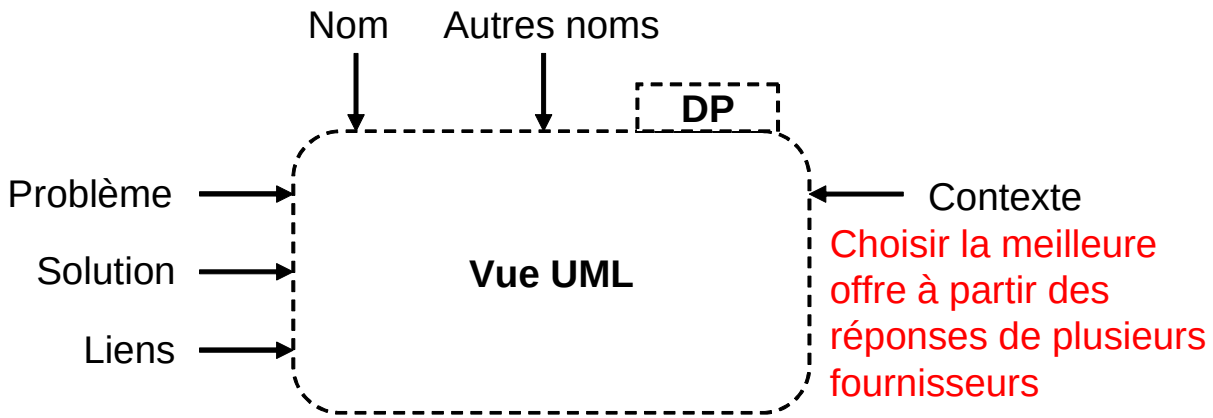
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un *design pattern*

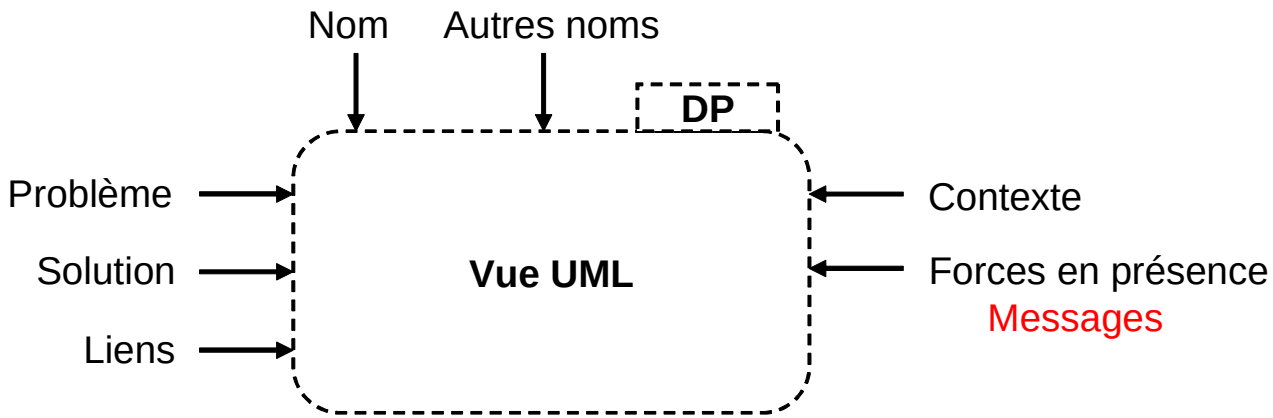
Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

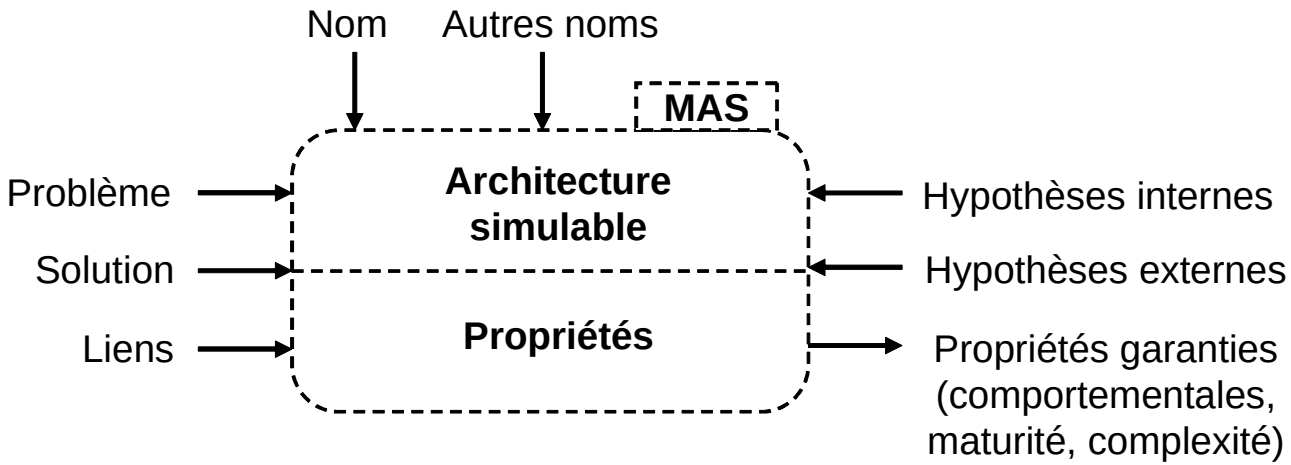
Un *design pattern*

Formalisation par Gamma, Helm, Johnson et Vlissides (1995)



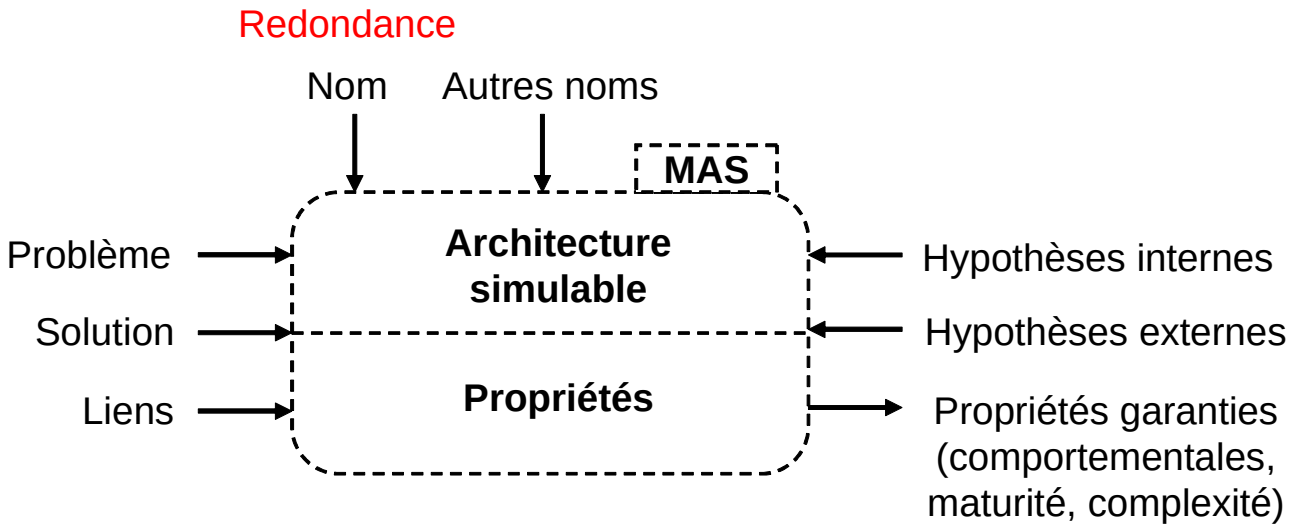
- Aide à la conception de logiciels
- Réutilisation des connaissances et du savoir-faire

Un motif d'architecture de sûreté



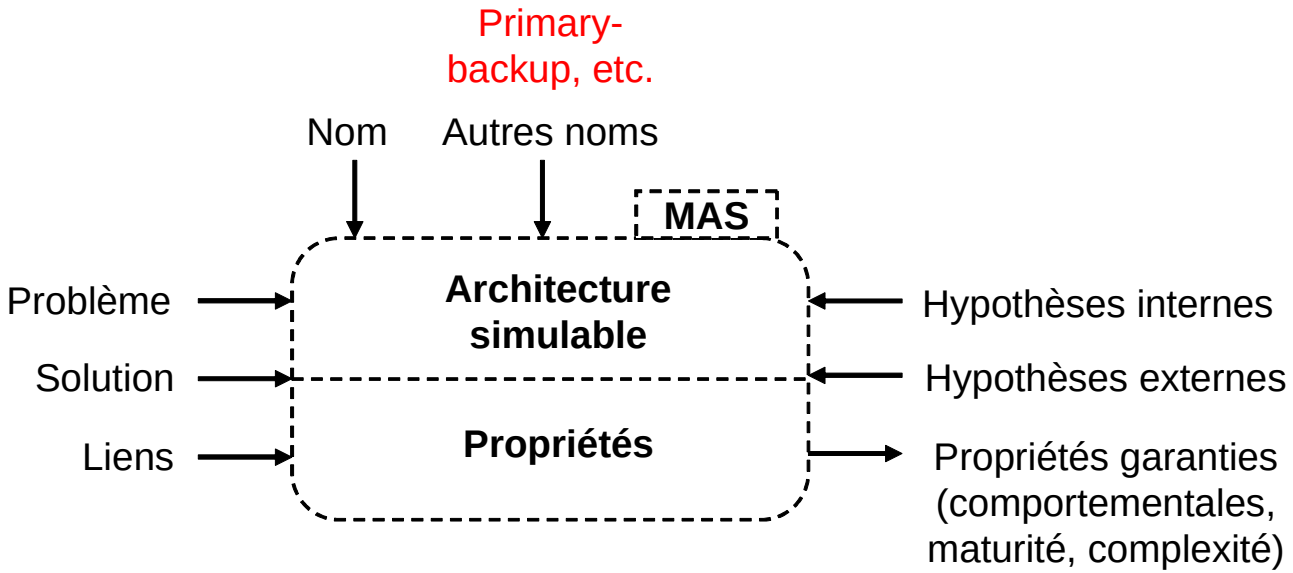
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



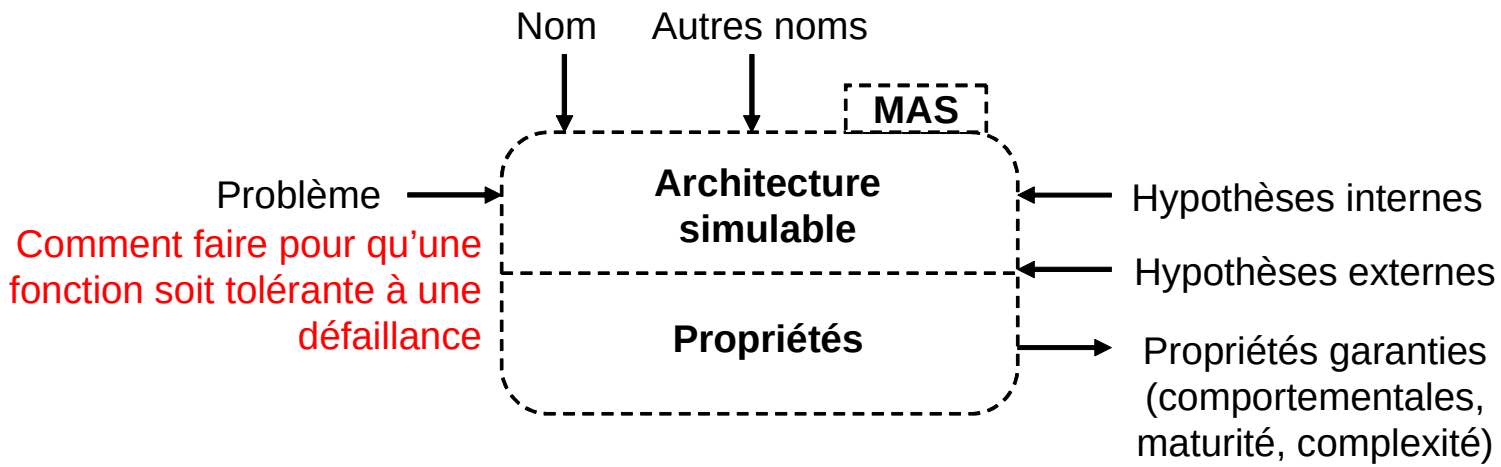
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



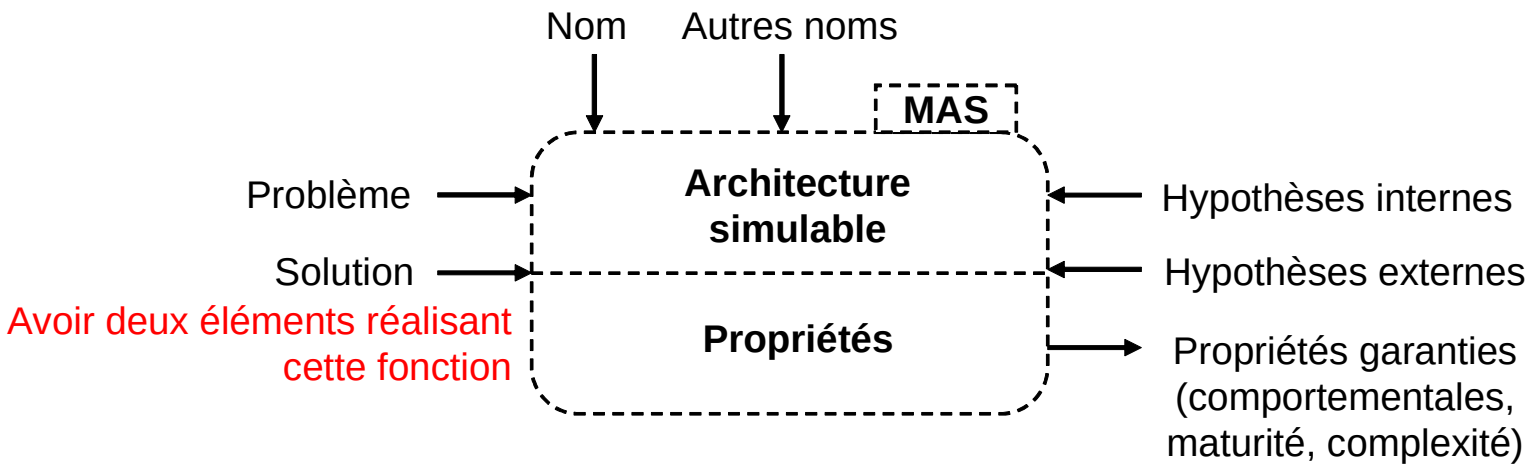
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



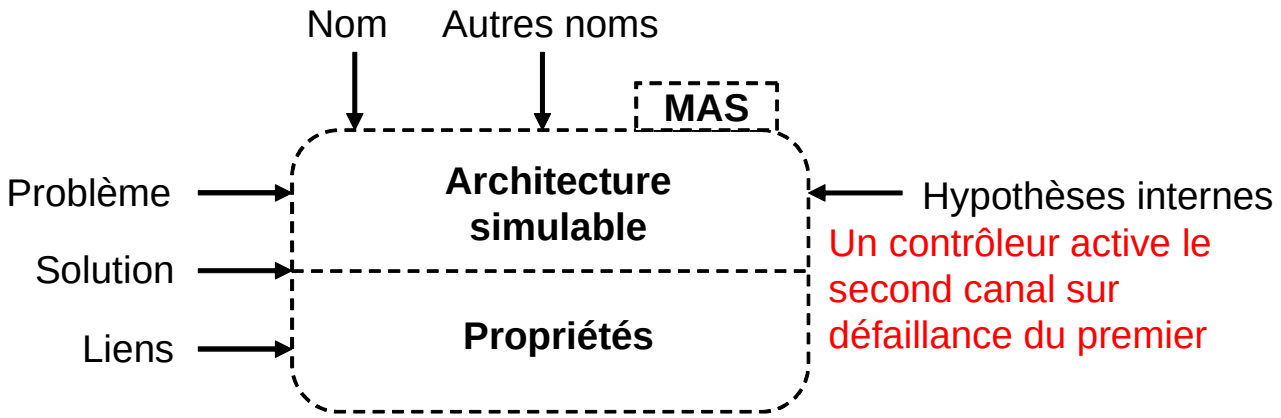
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



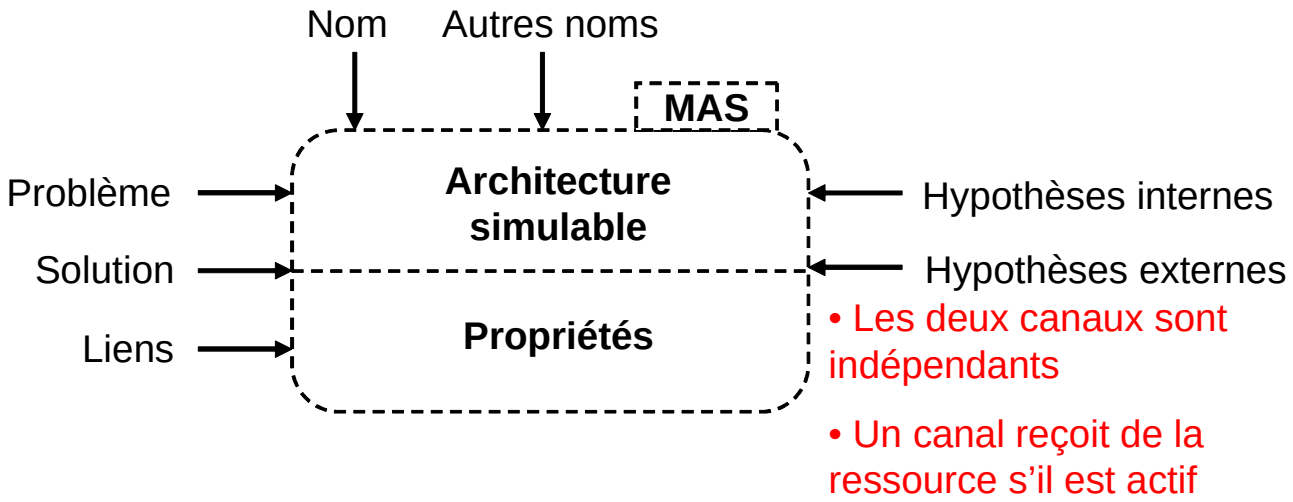
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



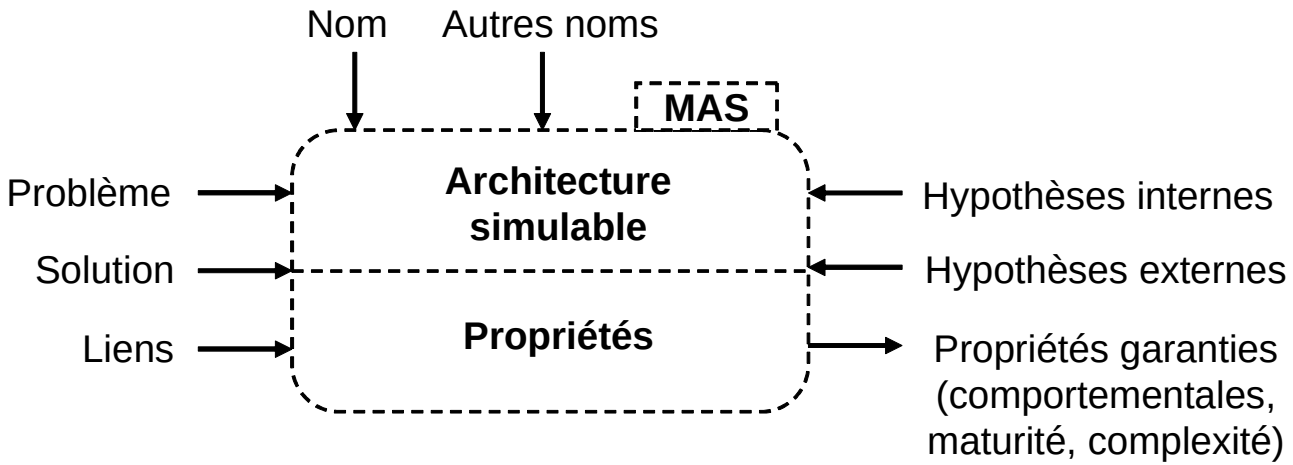
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

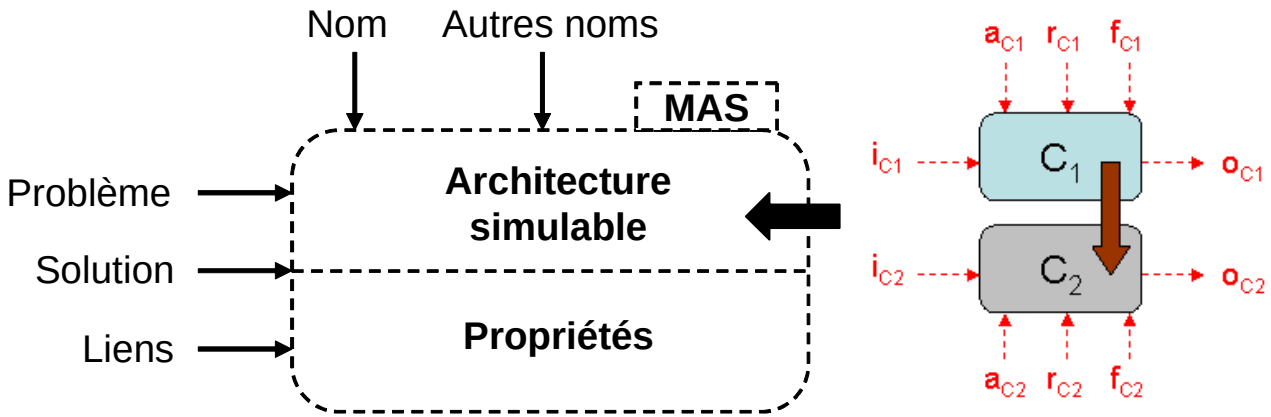
Un motif d'architecture de sûreté



Ce motif garantit la réalisation d'une fonction après occurrence d'une défaillance

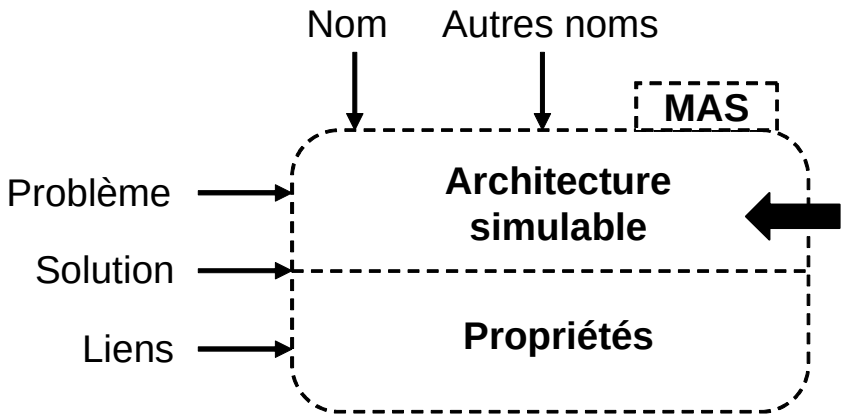
- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

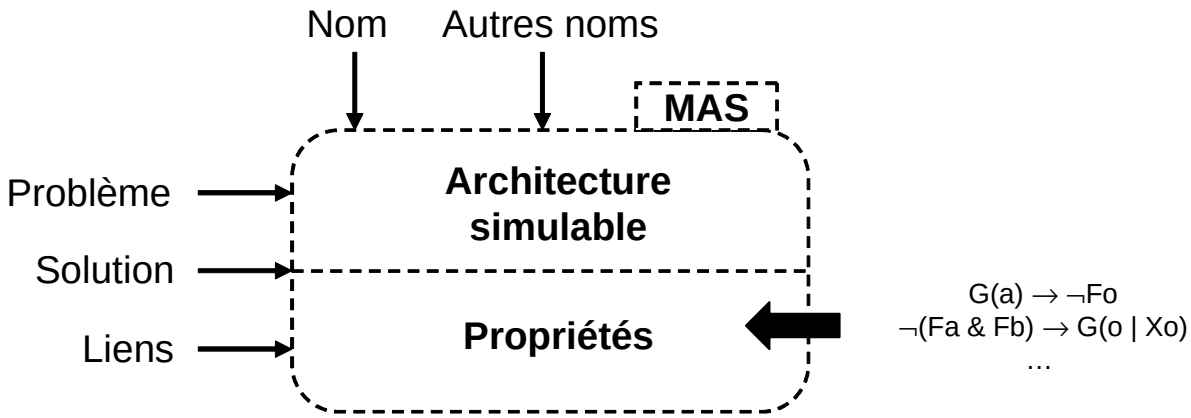
Un motif d'architecture de sûreté




```
node example
 …
 event b;
 trans a |- b -> a:=false;
 assert o=a;
 init a:=true;
edon
```

- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Un motif d'architecture de sûreté



- Architecture : description structure + comportement simulable
- Propriétés : formalisation hypothèses/propriétés

Partie architecture

Langage (*LaBRI*)

- formel hiérarchique et compositionnel
  - → systèmes complexes
- adapté à la SdF
  - → modélisation de la propagation des pannes (modes + evts)

Outils

- outils dédiés (édition/analyse : OCAS, Simfia, Arboost ; validation : MecV)
- passerelles vers autres langages (Lustre, SMV)

Partie architecture

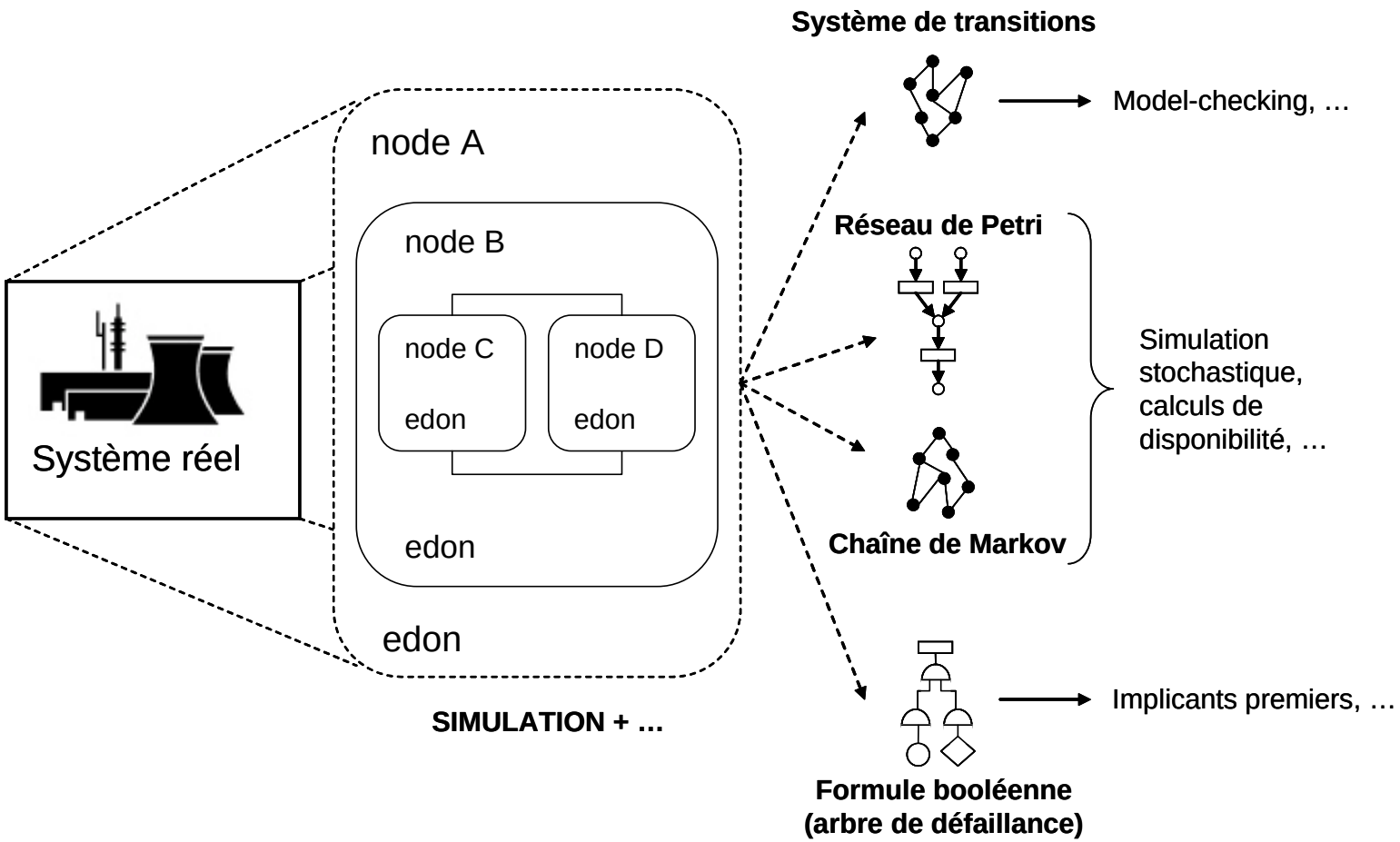
Langage (*LaBRI*)

- formel hiérarchique et compositionnel
  - $\longrightarrow$ systèmes complexes
- adapté à la SdF
  - $\longrightarrow$ modélisation de la propagation des pannes (modes + evts)

Outils

- outils dédiés (édition/analyse : OCAS, Simfia, Arboost ; validation : MecV)
- passerelles vers autres langages (Lustre, SMV)

Que fait-on avec ALTARICA ?

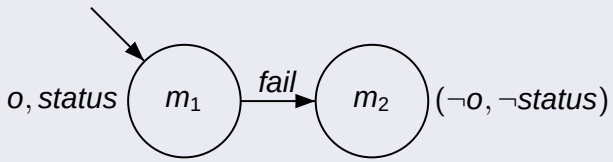


ALTARICA : concepts

Un composant quelconque

```
node component
 state status :bool;
 flow o :out :bool;
 event fail;
 trans status |- fail -> status :=false;
 assert o=status;
 init status :=true;
edon
```

Comportement : structure de Kripke étiquetée



Graphe dirigé, étiqueté sur ses états et transitions

Partie propriétés

- Evaluation d'exigences du type ...

  "La perte totale de puissance hydraulique est considérée catastrophique."

Exigence quantitative

"Sa probabilité d'occurrence doit être inférieure à 10^{-9}/FH."

Exigence qualitative

"Aucune panne simple ne doit mener à cette condition de défaillance (*FC*)."

Partie propriétés

- Evaluation d'exigences du type ...

  "La perte totale de puissance hydraulique est considérée catastrophique."

Passage quantitatif $\rightarrow$ qualitatif (généralisation)

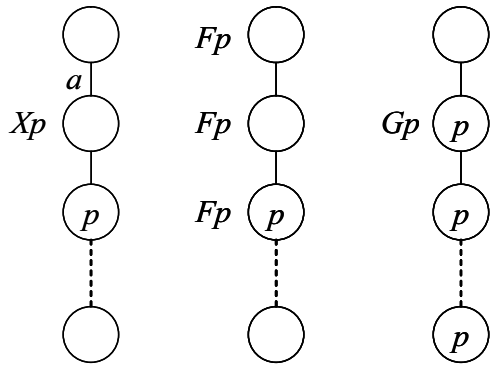
"S'il y a moins de N défaillances individuelles, alors cette *FC* ne doit pas se produire" avec :

- $N = 0$ pour $FC = Minor$
- $N = 1$ pour $FC = Major$ ou $Hazardous$
- $N = 2$ pour $FC = Catastrophic$

State-Event LTL : concepts

SE-LTL = (logique propositionnelle + opérateurs temporels)

- *p* prop. vraie dans un état si état étiqueté par *p*
- *a* événement vrai dans un état si *a* étiquette la transition issue de cet état
- *Xp* : *p* est vraie dans le prochain état
- *Fp* : il existe un état futur dans lequel *p* sera vraie
- *Gp* : globalement la propriété *p* est vraie dans tous les états



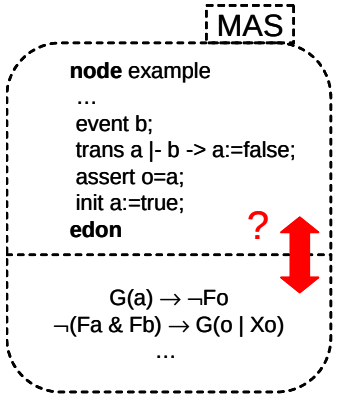
State-Event LTL (suite)

Intérêts

- Spécifier simplement des propriétés de traces d'exécutions
- Pouvoir parler des états et des événements

> "Une défaillance simple ne doit pas entraîner la perte globale du système."
>
> Cette spécification traite à la fois d'états et d'événements

Conclusion



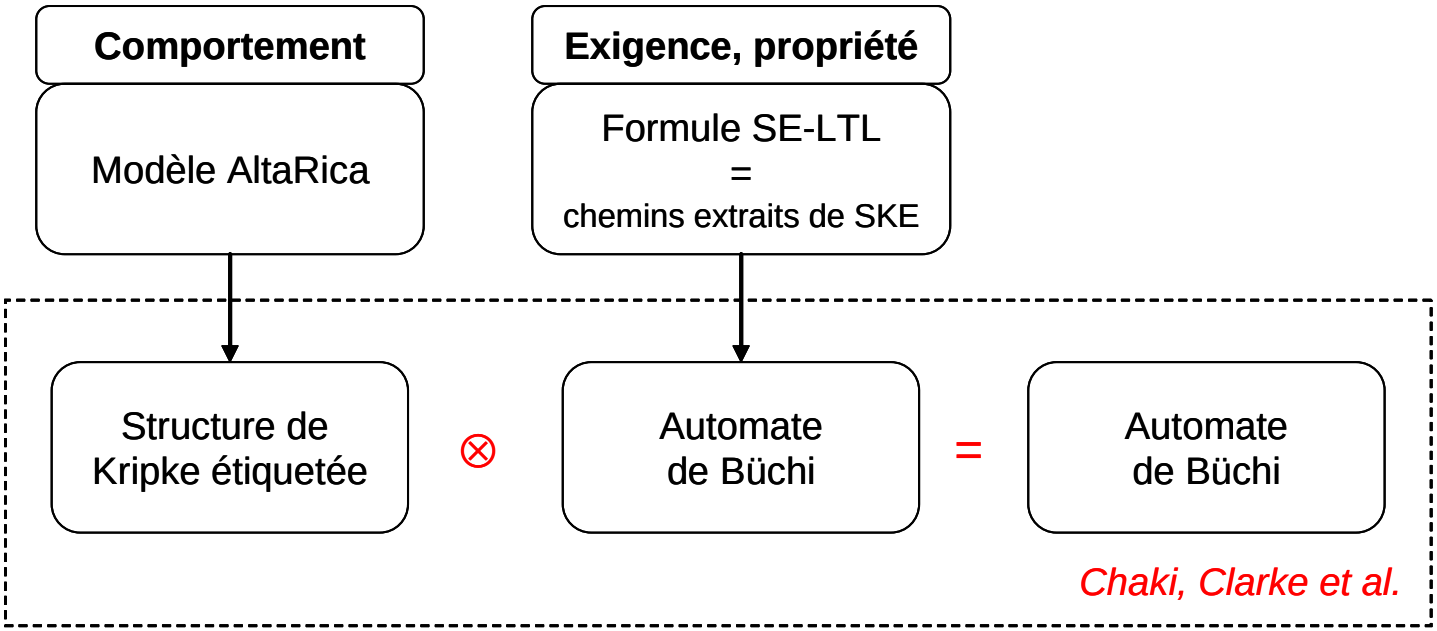
Problème

A	*B*	$A \otimes B$
ALTARICA	ALTARICA	OK
Propriété	Propriété	OK
ALTARICA	Propriété	?

→ Définir le comportement attendu de cette notation mixte

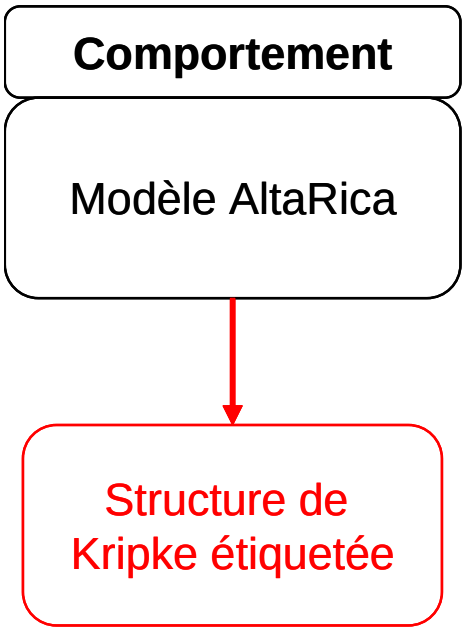
Plan

Comportement de la notation mixte



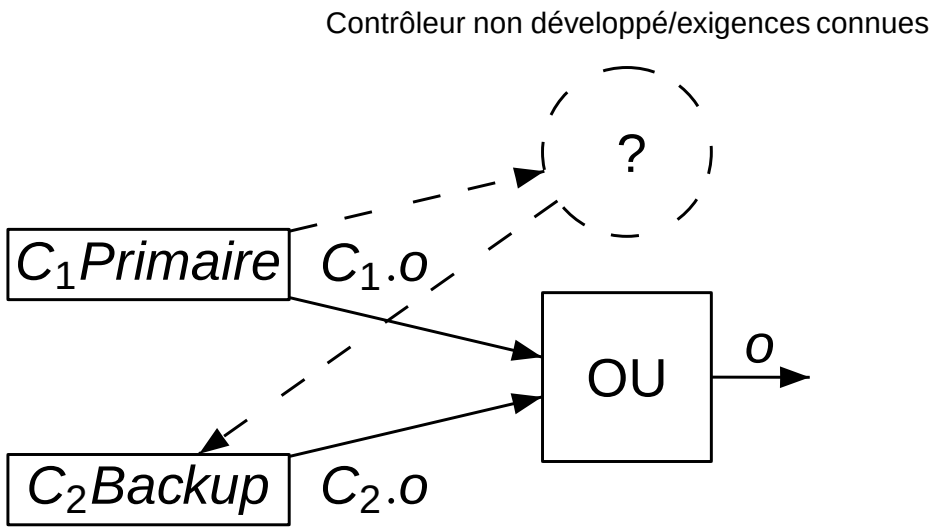
- Caractérisation des exécutions appartenant à l'intersection des exécutions des deux automates

Comment ... 1ère étape

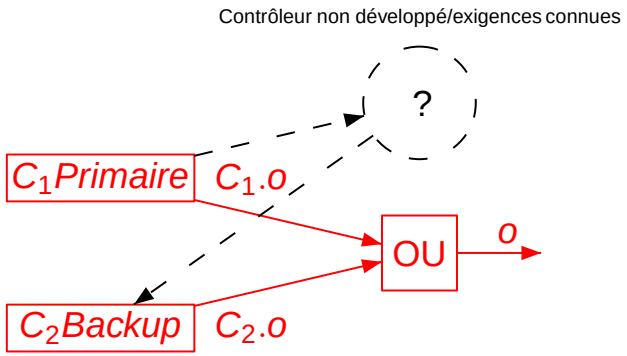


Exemple d'une redondance

- Redondance passive (contrôlée) de deux composants



Exemple d'une redondance (suite)

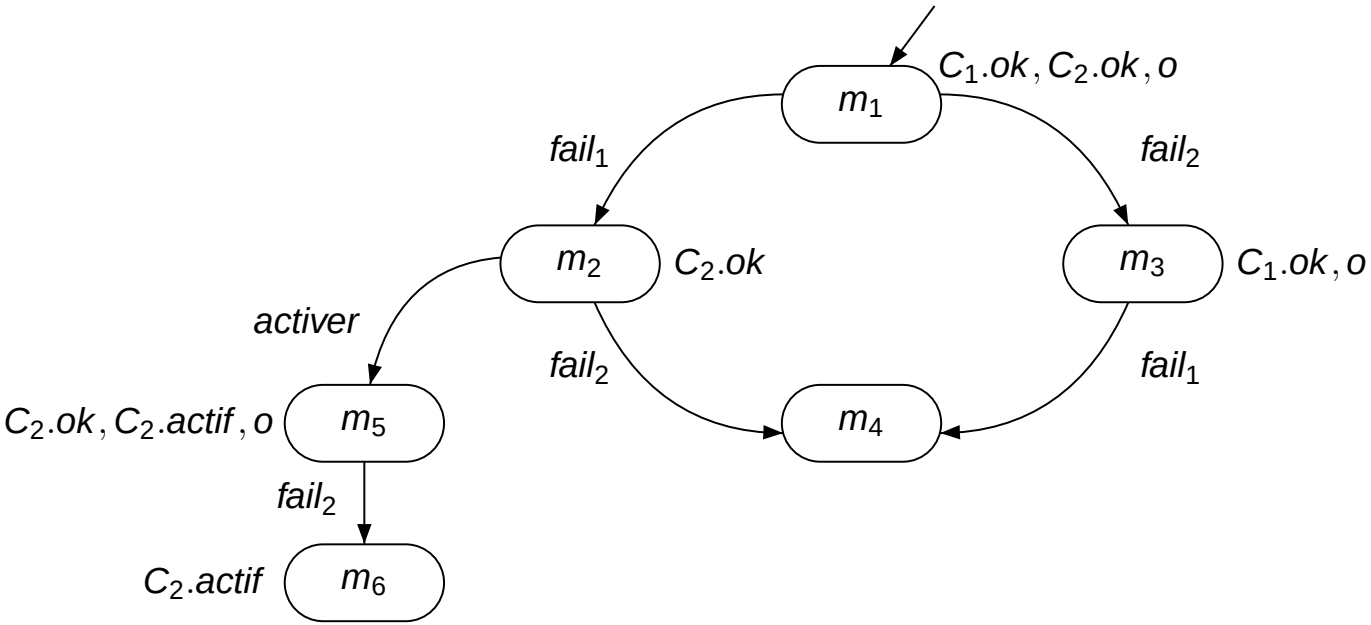


Code AltaRica du système contrôlé

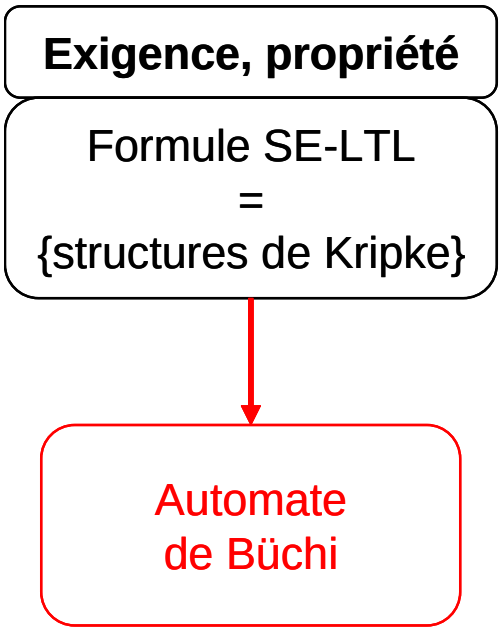
```
node redundancy_behavior
 ...
 trans C1.ok |- fail1 -> C1.ok :=false;
 C2.ok |- fail2 -> C2.ok :=false;
 assert o = (C1.i and C1.r and C1.ok) or (C2.i and C2.r and
C2.ok);
 init C1.ok :=true, C2.ok :=true;

edon
```

Exemple d'une redondance (suite)

Structure de Kripke étiquetée de la redondance



Comment ... 2ème étape



Rappel sur les automates de Büchi

Résultat de Wolper, Vardi et Sisla

"Pour tout φ de LTL on peut construire A_φ tel que $L_\omega(A_\varphi)$ correspond à l'ensemble des executions satisfaisant φ"

- φ_{LTL} sur proposition atomique d'états $P \longrightarrow A_{\varphi_{LTL}}$
- φ_{SE-LTL} sur proposition atomique d'états P et événements Σ s'interprète comme φ_{LTL} sur $P \cup \Sigma$

Définition des automates de Büchi

- Automate acceptant des traces infinies
- 6-uplet $\langle S, Init, P, \mathcal{L}, T, Acc \rangle$
- Une séquence infinie d'états est acceptée ssi elle contient une infinité de fois des états acceptants

Retour sur l'exemple d'une redondance

AltaRica

Code AltaRica du système contrôlé

$+$

Hypothèse interne : propriété d'activation

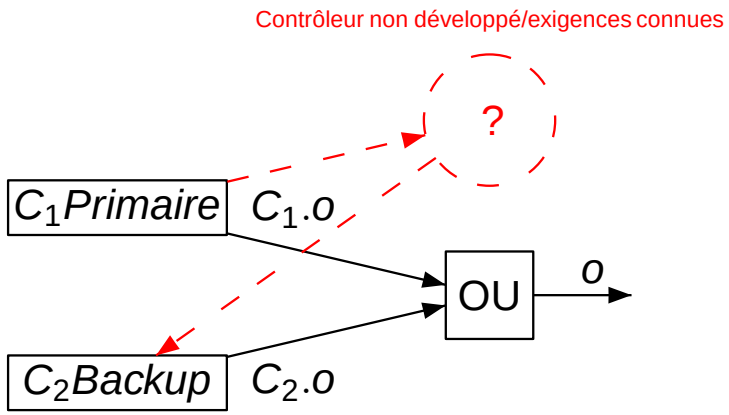
$A_b : G(\neg C_1.ok \Rightarrow XC_2.actif)$

$=$

Propriété garantie : tolérance à la panne simple

$P_g : \neg(Ffail_1 \wedge Ffail_2) \Rightarrow G(o \vee Xo)$

Retour sur l'exemple d'une redondance (suite)



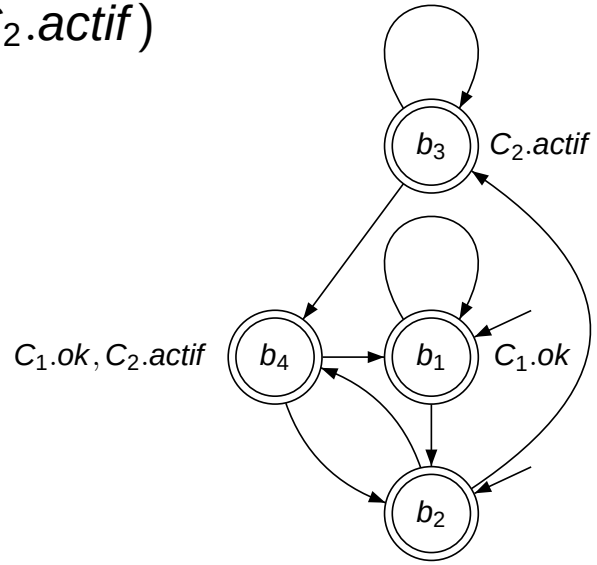
Code ALTARICA étendu du contrôleur

```
node redundancy_controller
 state C1.ok,C2.actif :bool;
 assert G(not(C1.ok) -> XC2.actif);
 init C1.ok :=true, C2.actif :=false;
edon
```

Retour sur l'exemple d'une redondance (suite)

Partie contrôleur de la redondance passive

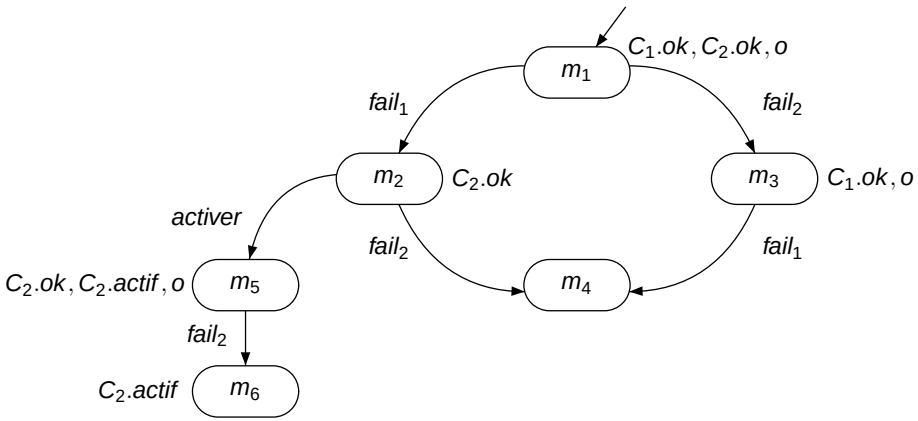
$A_b : G(\neg C_1.ok \Rightarrow XC_2.actif)$



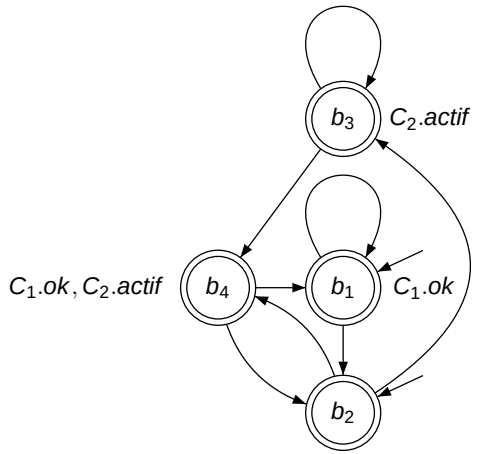
SKE vs Büchi

- $K_M = \langle S, I, P, \mathcal{L}, T, \Sigma, \varepsilon \rangle$ une structure de Kripke étiquetée
- $B = \langle S_B, Init_B, P \cup \Sigma, \mathcal{L}_B, T_B, Acc \rangle$ un automate de Büchi

SKE : K_M



Büchi : B



Produit SKE/Büchi [Chaki, Clarke et al.]

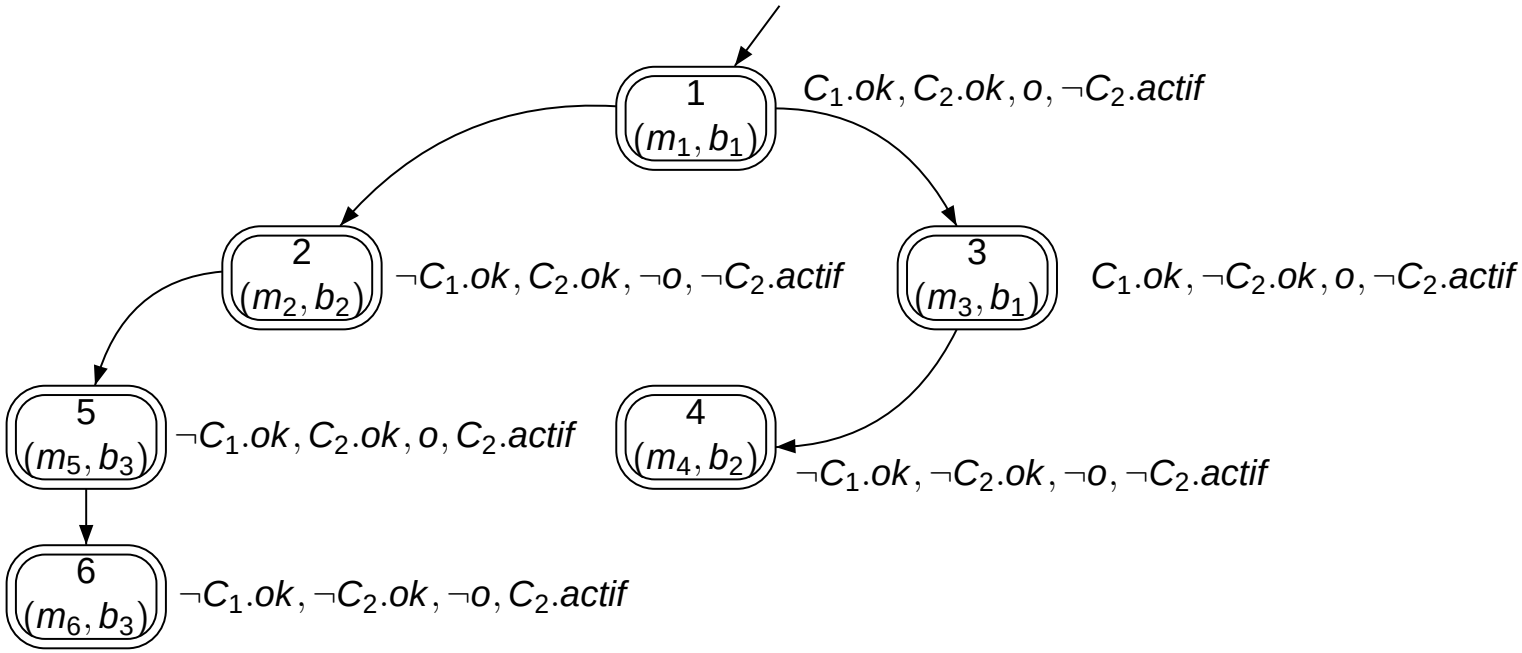
$K_M \otimes B = (S', Init', P \cup \Sigma, \widetilde{\mathcal{L}}, T', Acc')$ est un automate de Büchi tel que :

Construction de $K_M \otimes B$

1. $S' = \{(s, b) \in S \times S_B$ avec s et b des états compatibles $\}$
2. $(s, b) \longrightarrow (s', b')$ ssi $\exists x \in \Sigma \mid s \xrightarrow{x} s'$, $b \longrightarrow b'$ et s et b sont compatibles
3. $(s, b) \in Init'$ ssi $s \in Init$ et $b \in Init_B$
4. $(s, b) \in Acc'$ ssi $b \in Acc$

Retour sur l'exemple d'une redondance

Automate reconnaissant les exécutions du produit *système contrôlé/contrôleur*



Définition formelle d'un motif

Définition d'un motif C

Soient K_M une SKE, P un ensemble de formules de SE-LTL alors $C = (K_M, P)$ avec :

- $P = \{A_e, A_b, P_g\}$

Comportement d'un motif ...

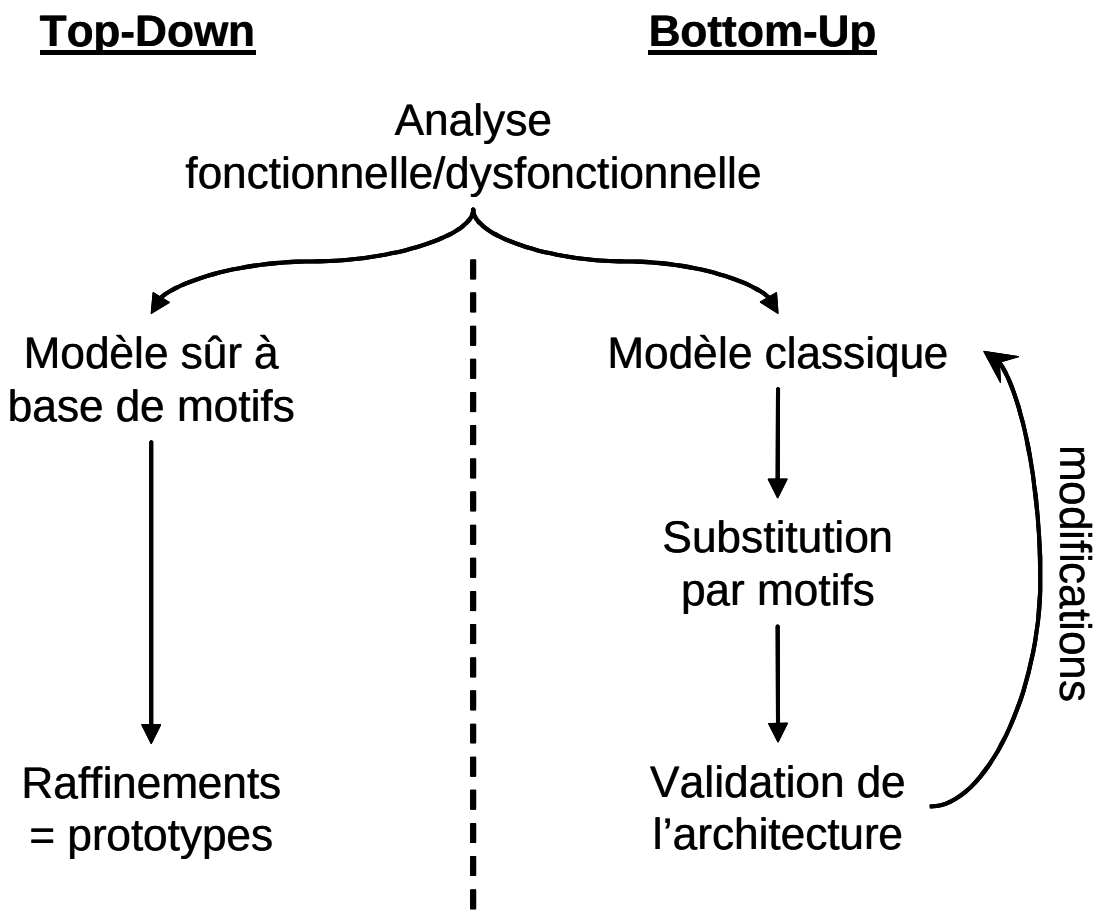
- ensemble des exécutions reconnues par $K_M \otimes A_b$

C bien formé ssi ...

1. $\forall \phi \in P$, ϕ n'est pas une tautologie et est consistante
2. K_M est un modèle plausible (au moins 1 état initial, 1 transition)
3. $A_b \cup A_e$ et K_M sont compatibles
4. $\forall p \in P_g$, p est conséquence logique de K_M, A_b et A_e

Plan

Deux approches



Scénarii d'utilisation

1. Construction d'une architecture à base de motifs
2. Substitution de motifs dans une architecture
3. Preuve de propriétés et allocation d'exigences

Scénarii d'utilisation

1. Construction d'une architecture à base de motifs
2. Substitution de motifs dans une architecture
3. Preuve de propriétés et allocation d'exigences

Scénarii d'utilisation

1. Construction d'une architecture à base de motifs
2. Substitution de motifs dans une architecture
3. Preuve de propriétés et allocation d'exigences

Scénarii d'utilisation

1. Construction d'une architecture à base de motifs
2. Substitution de motifs dans une architecture
3. Preuve de propriétés et allocation d'exigences

Scénarii d'utilisation

1. Construction d'une architecture à base de motifs
2. Substitution de motifs dans une architecture
3. Preuve de propriétés et allocation d'exigences

1. Intégration de motifs

Problème

Comment intégrer correctement un motif au sein d'une architecture ?

Solution

- Vérifier la compatibilité types I/O
- Vérifier les services fournis par l'architecture hôte
  - $\rightarrow$ hypothèses d'environnement instanciées
  - $\rightarrow$ allocation d'exigences dérivées aux composants

$\Rightarrow$ héritage des propriétés garanties

- Moyens : Symbolic Model Verifier (SMV) [MacMillan]

2. Substitution de motifs

Problème

Comment substituer correctement un morceau d'architecture par un motif au sein d'une architecture ?

Solution

- Montrer l'équivalence de simulation des deux modèles
  - puissance d'abstraction intéressante
  - préservation des propriétés de *ACTL**
- Montrer la tenue des hypothèses internes par l'archi.

- Moyens : SMV et MecV [Vincent]

3. Preuve de propriétés et allocation d'exigences

Problème

Comment prouver des propriétés/exigences sur un modèle à base de motifs ?

Solution

- Utiliser les hypothèses internes et d'environnement

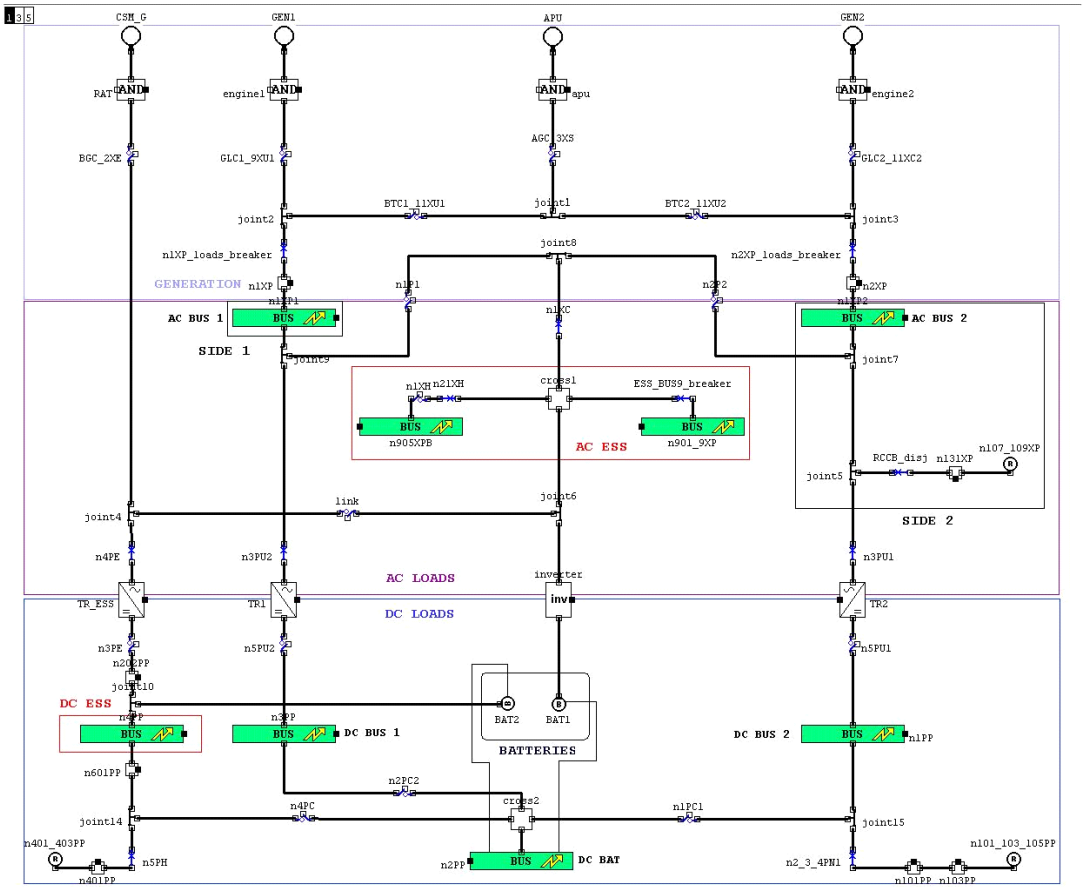
  $\models_{K_M} (\bigwedge_{s \in S} A_{b_s} \wedge \bigwedge_{s \in S} A_{e_s}) \rightarrow R$
- Utiliser les propriétés garanties

  $\models_{K_M} \bigwedge_{s \in S} P_{g_s} \rightarrow R$

- Moyens : SMV

Plan

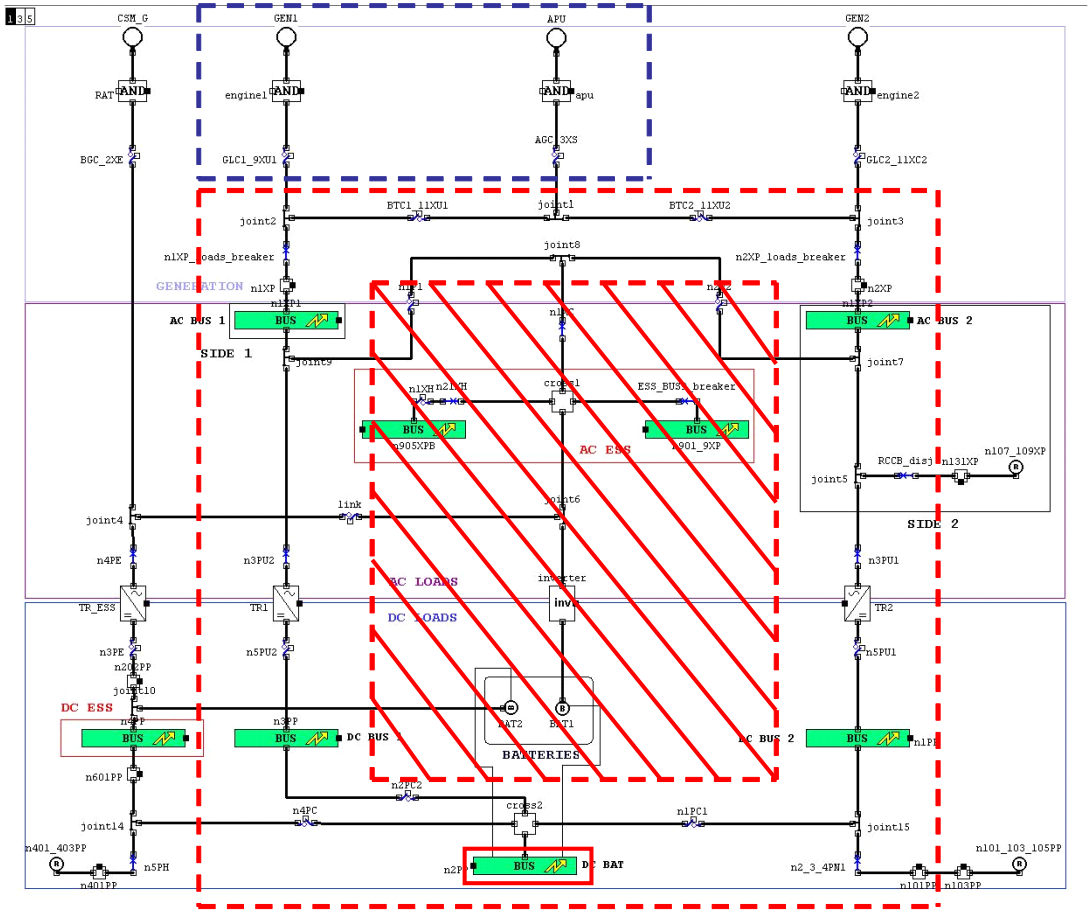
Modèle du système électrique type A320



Modélisation avec OCAS
Quelques chiffres ...

- 2 lignes nominales
- 1 ligne de secours
- ≈ 50 composants
- ≈ 15 éléments commandés
- ≈ 100 défaillances

Modèle du système électrique type A320



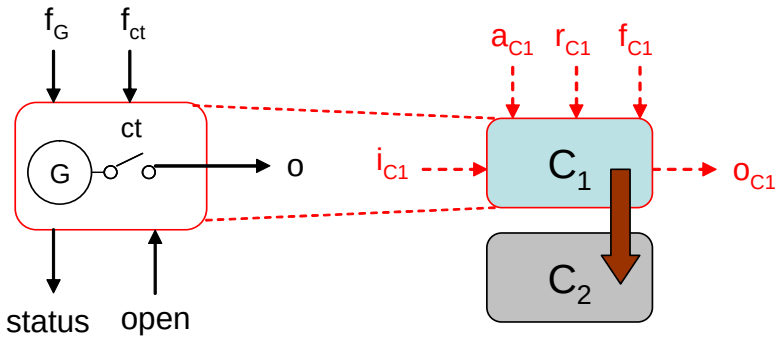
Modélisation avec OCAS
Quelques chiffres ...

- 2 lignes nominales
- 1 ligne de secours
- ≈ 50 composants
- ≈ 15 éléments commandés
- ≈ 100 défaillances

Modèle abstrait simple

Substitution

- Redondance de génération = motif
  - assemblage : générateur + contacteur = canal ?



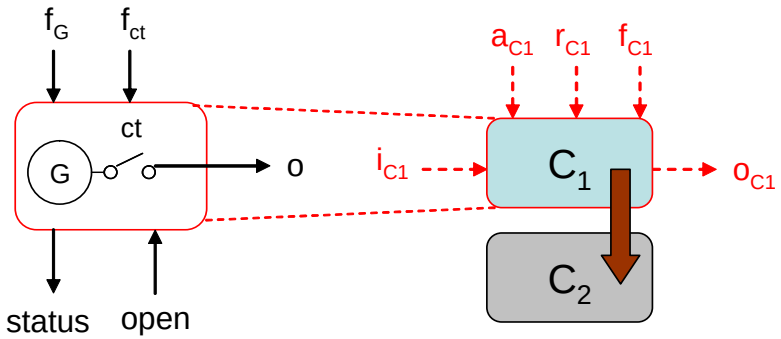
Classes d'équivalence :

o_{C1} = (status & ¬open)
i_{C1} = 1
a_{C1} = ¬open
r_{C1} = 1
f_{C1} = (f_G | f_{ct})

Modèle abstrait simple

Substitution

- Redondance de génération = motif
  - assemblage : générateur + contacteur = canal ?



Classes d'équivalence :

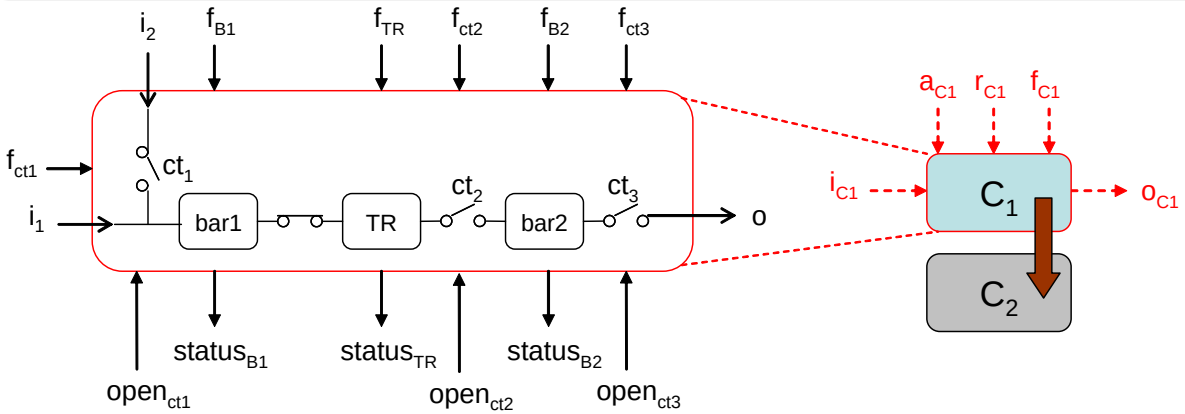
o_{C1} = (status & ¬open)
i_{C1} = 1
a_{C1} = ¬open
r_{C1} = 1
f_{C1} = (f_G | f_{ct})

SMV, MecV

Modèle abstrait simple

Substitution

- Redondance de distribution = motif
  - assemblage : disjoncteur + bars + contacteur + transformateur = canal ?



Classes d'équivalence :

$o_{C1} = (status_{B1} \,\&\, status_{TR} \,\&\, status_{B2}) \,\&\, \neg(open_{ct2} \mid open_{ct3}) \,\&\, (i_1 \mid (i_2 \,\&\, \neg open_{ct1}))$

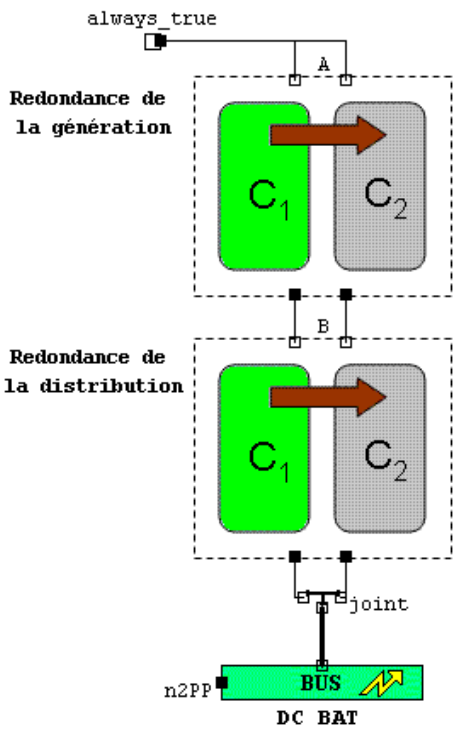
$i_{C1} = (i_1 \mid i_2)$

$a_{C1} = \neg(open_{ct1} \mid open_{ct2} \mid open_{ct3})$

$r_{C1} = 1$

$f_{C1} = (f_{B1} \mid f_{TR} \mid f_{ct2} \mid f_{B2} \mid f_{ct3})$

Modèle abstrait simple



Hypothèses internes instanciées

Génération

$$G((f_G \vee f_{ct}) \rightarrow X(\neg open_{C_2}))$$

Distribution

$$G((f_{B_1} \vee f_{TR} \vee f_{ct_2} \vee f_{B_2} \vee f_{ct_3}) \rightarrow Xa_{C_2})$$

Contrôleur

Validation/construction du contrôleur du système

Résultats

Principalement ...

- prototypage rapide d'architectures sûres
- assistance/vérification de l'allocation d'exigences de sûreté
- obtention d'une vue synthétique et fonctionnelle de l'architecture

mais aussi ...

- réduction significative du nombre de nœuds BDD alloués pour preuve SMV
- réduction importante du cône d'influence (SMV)

Plan

Bilan

1. Définition de motifs d'architectures de sûreté
   - contenu utile du point de vue SdF
2. Définition d'un cadre formel unique
   - SE-LTL comme langage/sémantique pivot entre formalisme déclaratif et opérationnel
   - caractérisation des exécutions acceptables d'un système spécifié en langage mixte
3. Elaboration d'un catalogue de motifs (restreint)
4. Définition de scénarii d'utilisation des motifs
5. Application de l'approche à un système concret

Perspectives

1. Extension du catalogue de motifs
   - architectures plus complexes (COM-MON, etc.)
   - éviter les antipatterns (attribut)
2. Outillage de l'approche (implémentation dans ISAAC)
   - consultation de base de données
   - aide à la substitution de motifs
3. Etude de l'impact d'une conception basée motifs sur les AdD
   - simplification ou non ?
   - 1 motif $\stackrel{?}{=}$ sous-arbre d'un AdD dynamique