Preuves formelles appliquées à la modélisation Système

SAMOKISH Andrei, Laboratoire Methodes Formelles, Knowledge Inside

CONTEJEAN Evelyne, Laboratoire Methodes Formelles

Motivation: Analyse des risques dans le domaine nucléaire

AMDE (Failure Mode and Effects Analysis) méthodologie et outil pour:

- identifier les risques
- déterminer leurs causes
- déterminer un plan d'action pour réduire les risques et
- déterminer les risques résiduels éventuels

Conformément aux lois et normes

Basé sur une spécification

Objectif: modèle formel du système + AMDE
[Analogie: vérification de programme]

Enjeux et difficultés

- Traduire les normes existantes en description formelle est un enjeu
  - nouvelle approche
  - devrait venir avec l'automatisation
  - ne doit pas être affiché à l'utilisateur final

- Spécification de system
  - Correction
  - Complétude (aucun oubli)

- AMDE = des milliers de lignes. Complexe pour un être humain.

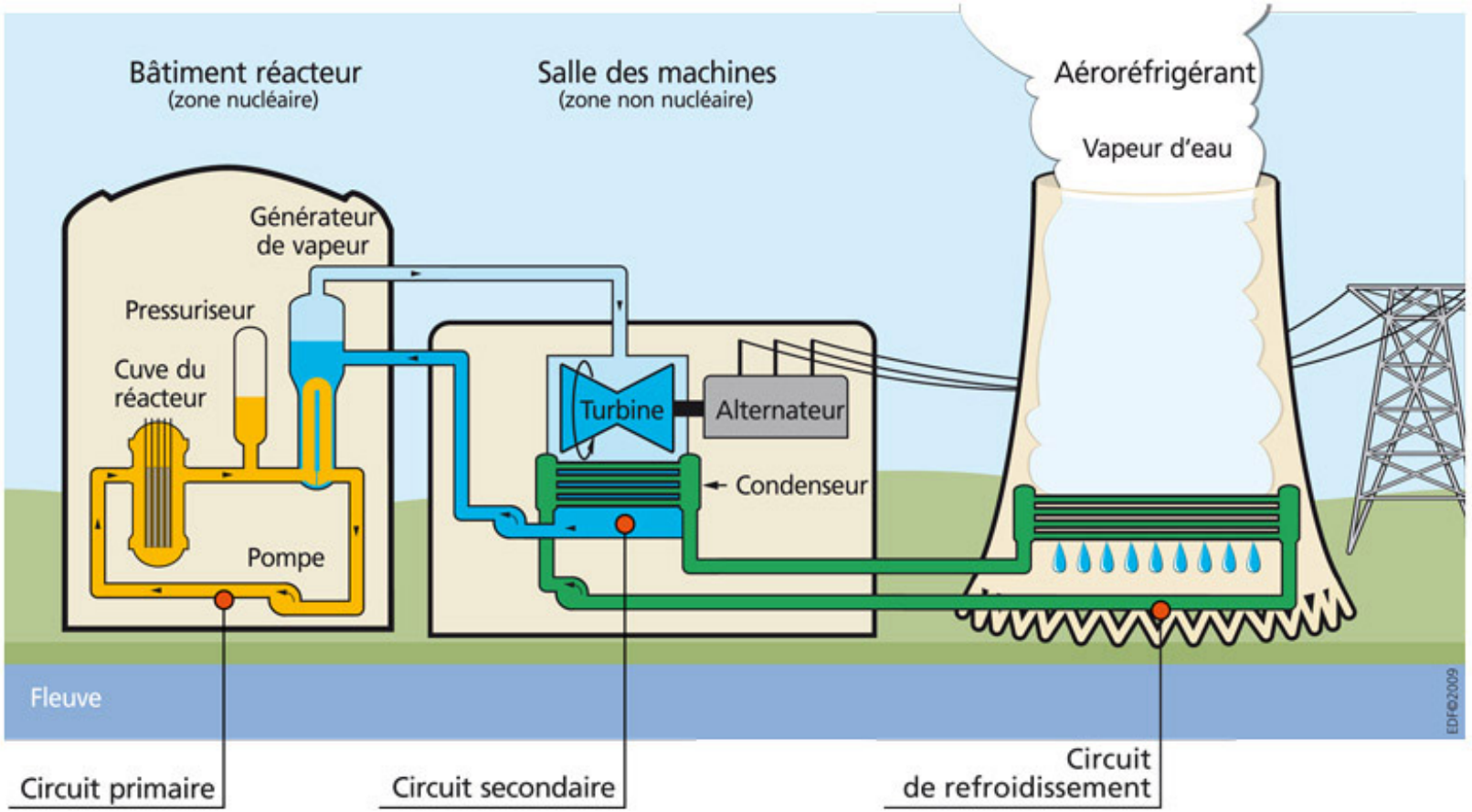
- Méthode systématique pour chaque ligne
  - Correction
  - Complétude (aucun oubli)

Feuille de route

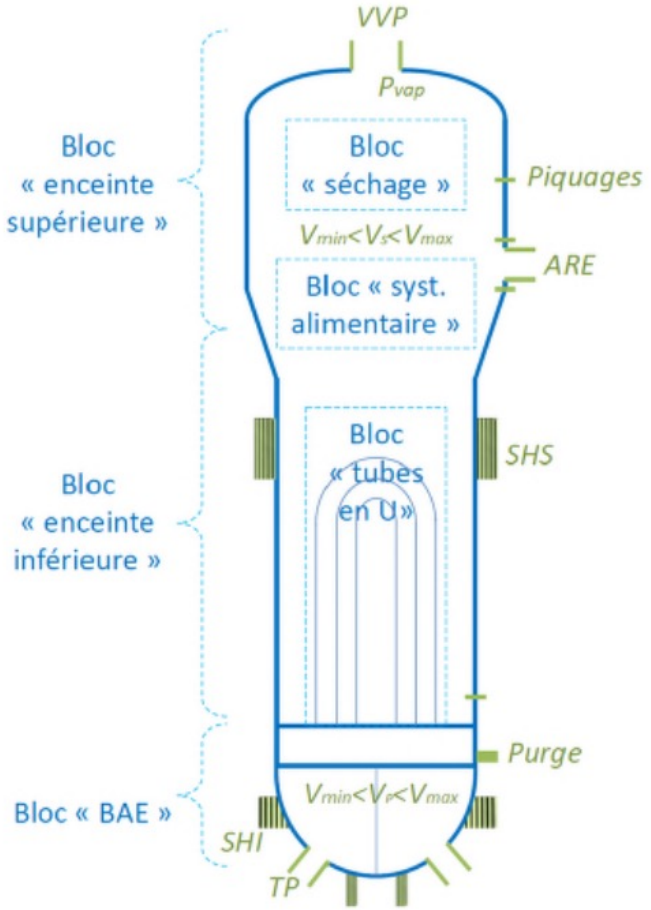
- Formaliser le système par une analyse fonctionnelle externe et interne

- Formaliser la AMDE
  - Traduire les exigences légales
  - Robustesse
  - Automatisation

La modélisation système d'un générateur de vapeur

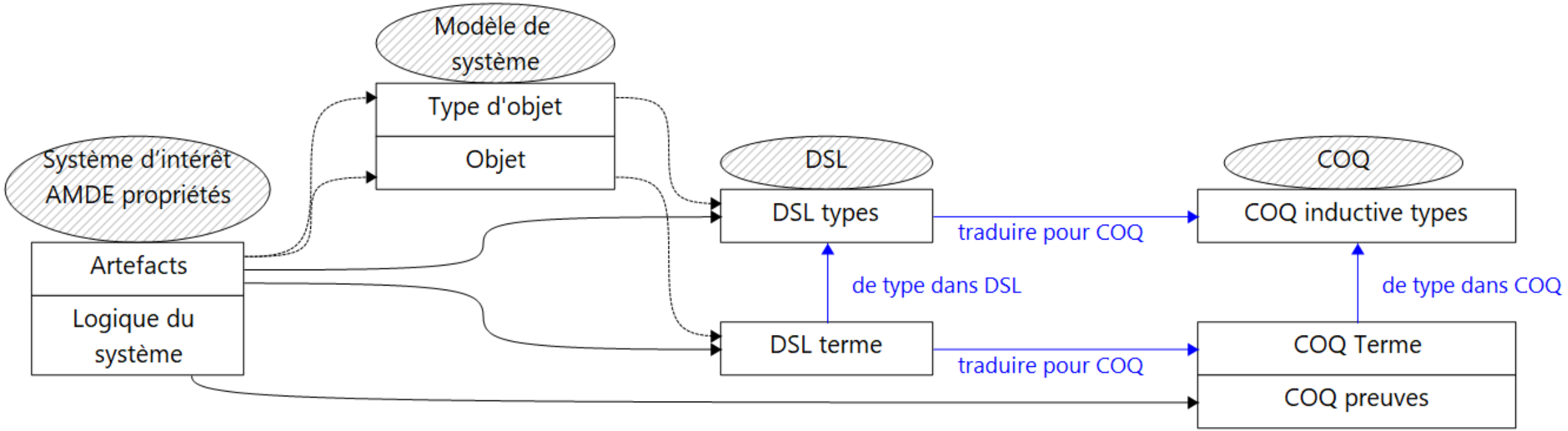


Principe d'un réacteur nucléaire (IRSN)

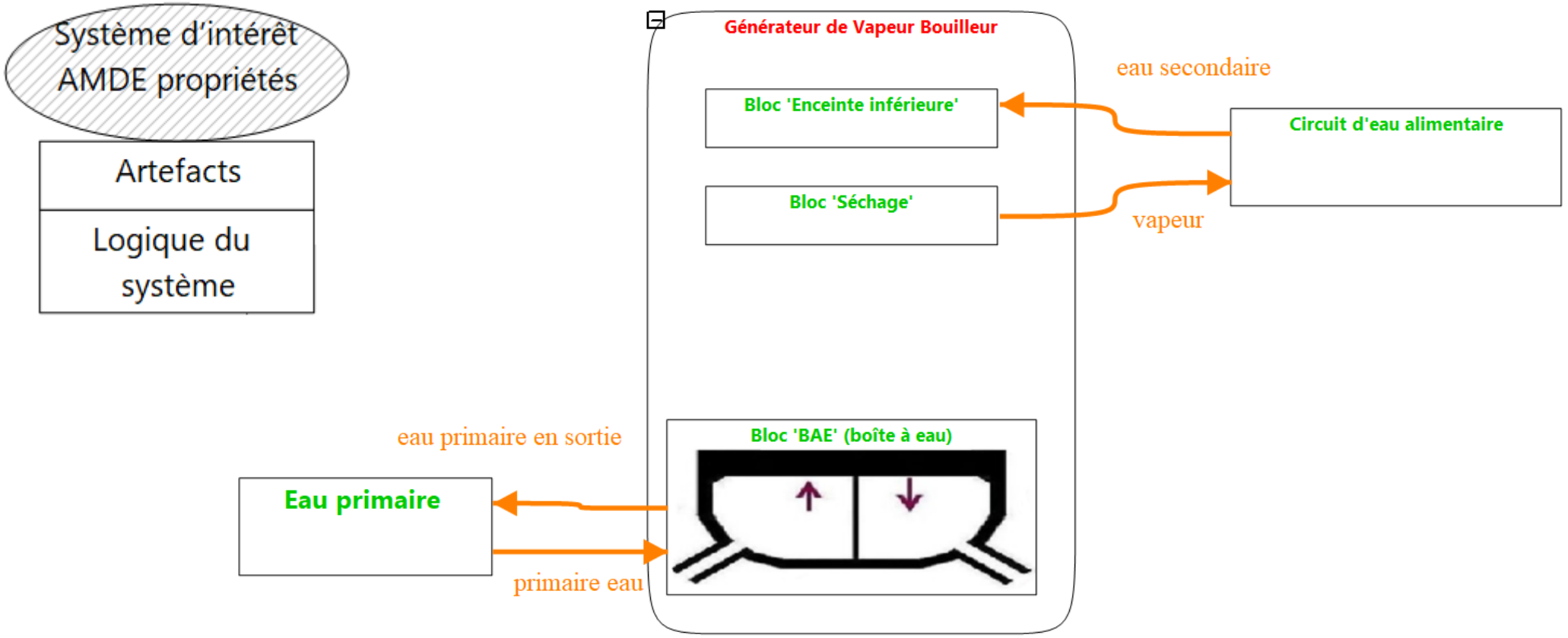


Principe d'un générateur de vapeur (AFCEN guide)

Processus de formalisation



Un modèle simplifié d'un générateur de vapeur

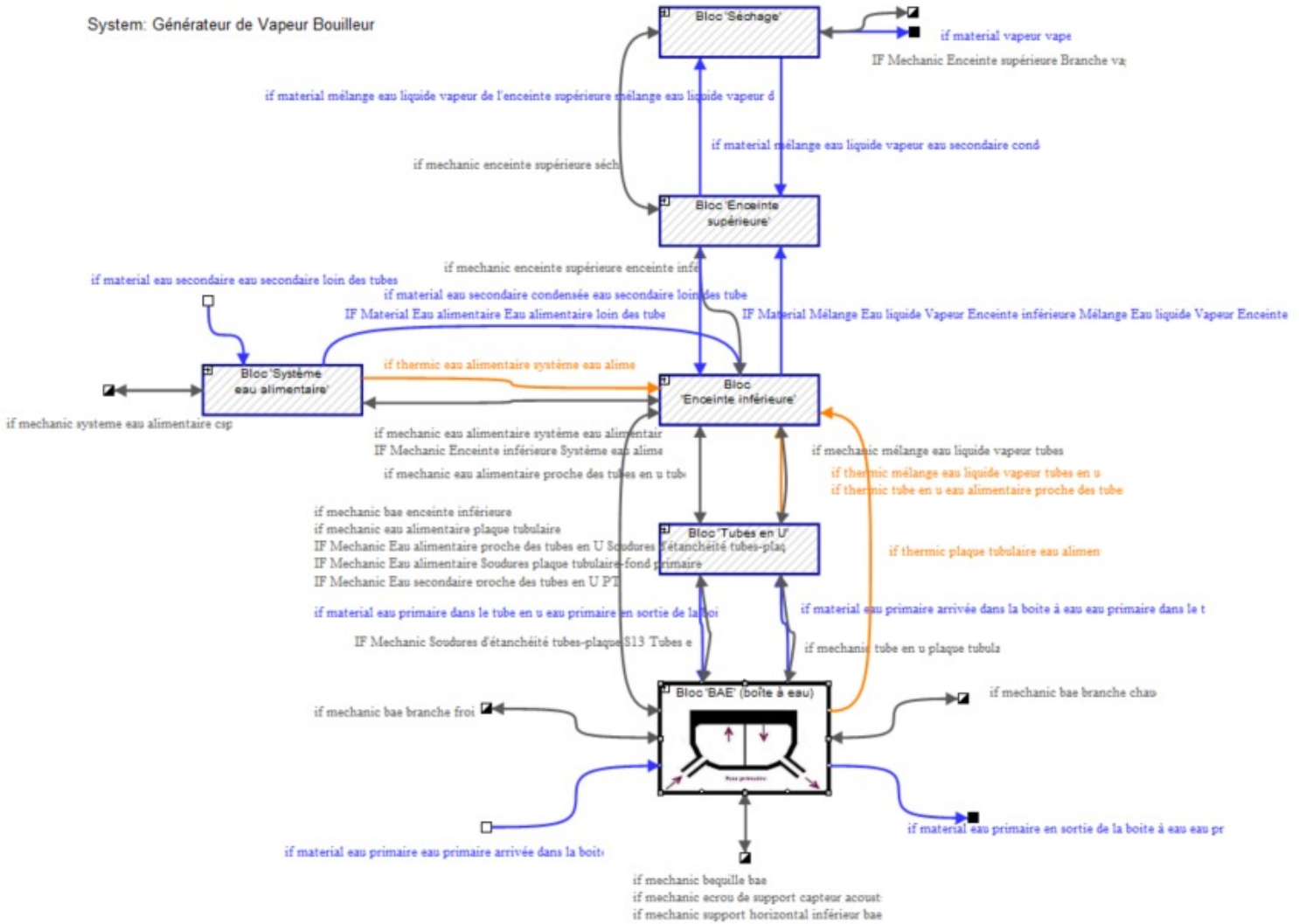


Exemple de modèle réel

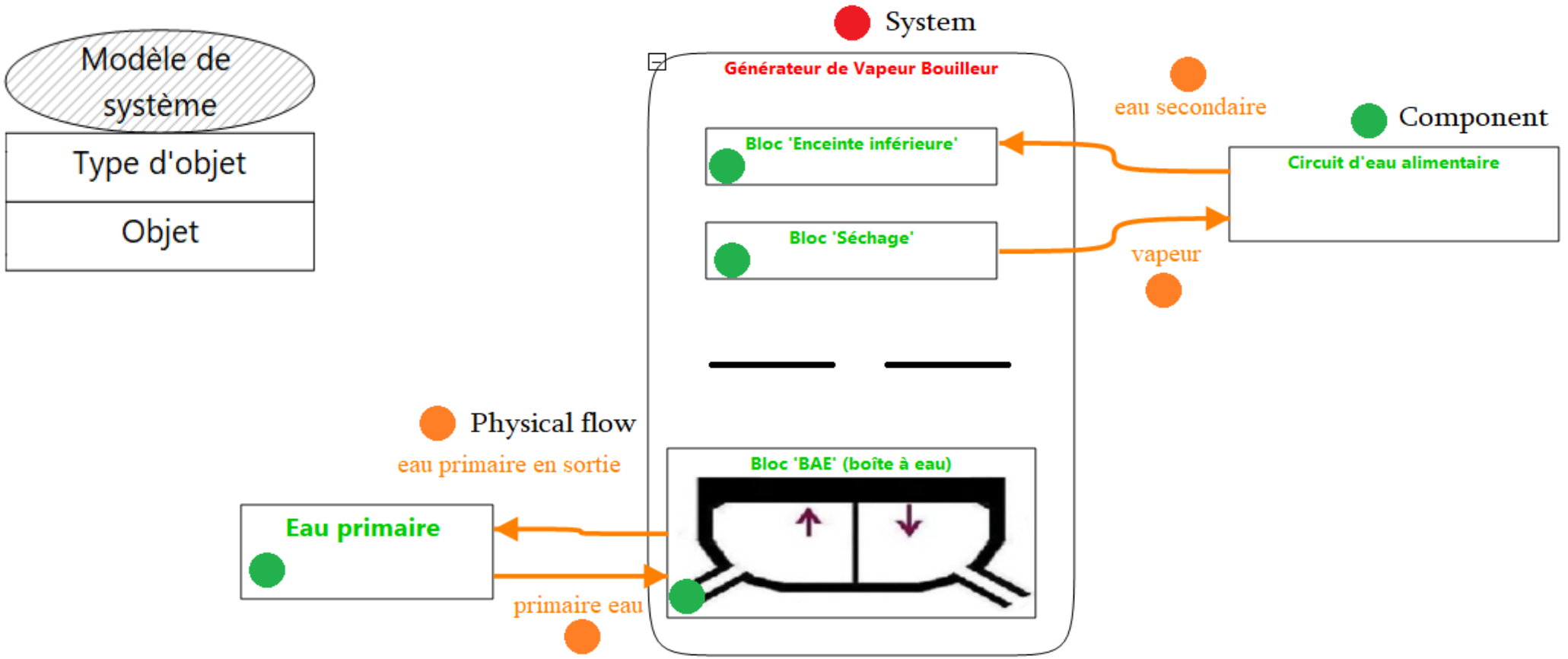
Système d'intérêt
AMDE propriétés

Artefacts

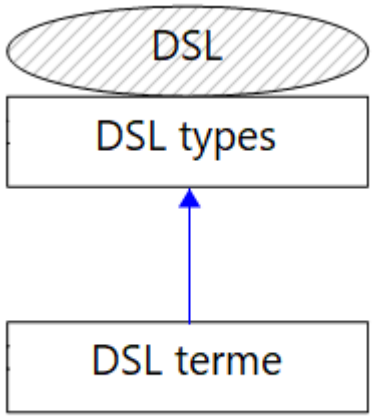
Logique du système

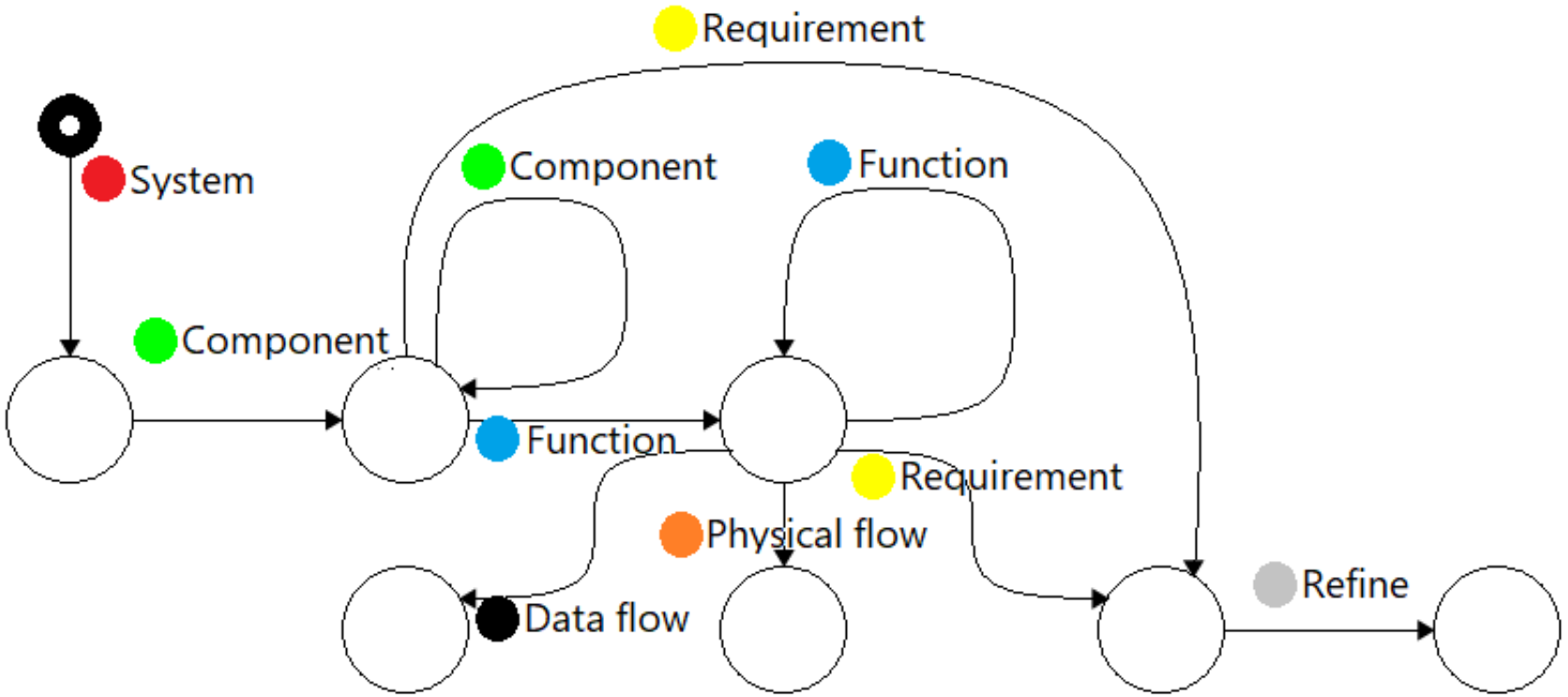


Un modèle simplifié d'un générateur de vapeur avec le typage de ses objets

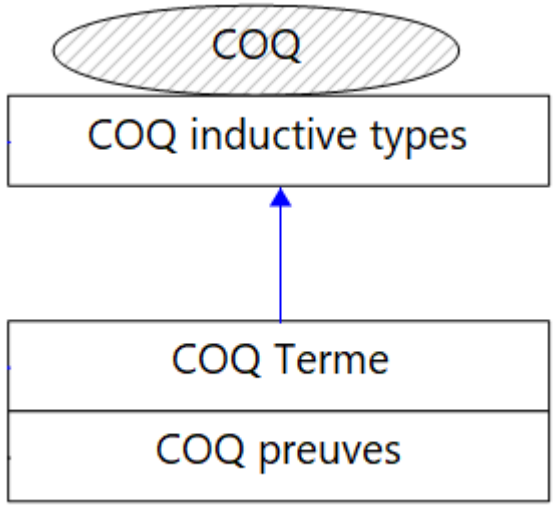


DSL


DSL
DSL types
DSL terme


Requirement
System
Component
Function
Component
Function
Requirement
Physical flow
Refine
Data flow

Traduction du DSL vers des types inductifs dans COQ (shallow embedding)




```
Inductive System : Type :=
| CSystem : Identifier -> System
| CComponent_System : list Component -> System -> System
with Component : Type :=
| CComponent :  Identifier -> Component
| CComponent_Component : list Component -> Component -> Component
| CFunction_Component : list Function -> Component -> Component
with Function : Type :=
| CFunction :  Identifier -> Function
| CFunction_Function : list Function -> Function -> Function
| CRequirement_Function : list Requirement -> Function -> Function
| CData_flow_Function : list Data_flow -> direction -> Function -> Function
| CPhysicla_flow_Function : list Physical_flow -> direction -> Function -> Function
with Requirement : Type :=
| CRequirement : Identifier -> Requirement
| CRefine_Requirement : list Refine -> direction -> Requirement -> Requirement
with Data_flow : Type :=
| CData_flow :  Identifier -> Data_flow
with Physical_flow : Type :=
| CPhysical_flow : Identifier -> Physical flow
with Refine : Type :=
| CRefine : Identifier -> Refine.
```

System

Component

Function

Requirement

Data flow

Physicla flow

Refine

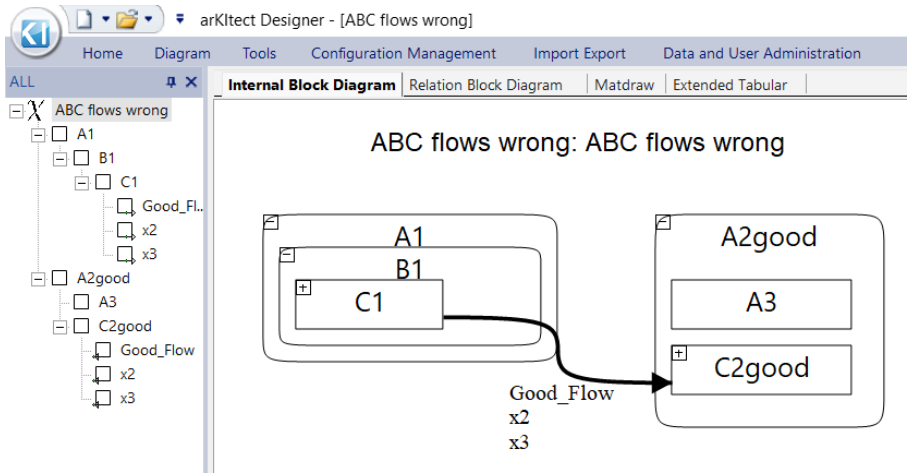
Traduction des termes du DSL en termes COQ

```
...
Definition Feedwater_inlet_27 := CFunction_Component [ ... ]
  (CPhysical_flow_Component [ Secondary_water; ... ] input
  (CComponent 2194728288258))).
...
Definition Drying_block_11 := CFunction_Component [... ]
  (CComponent_Component [... ]
  (CComponent 1103806595096)).
...
Definition Steam_Generator_Boiler_1 := CComponent_System
  [ Drying_block_11;
    Feedwater_inlet_27;
    Water_tank_32;
    ... ]
  (CSystem 1013612281857).
```

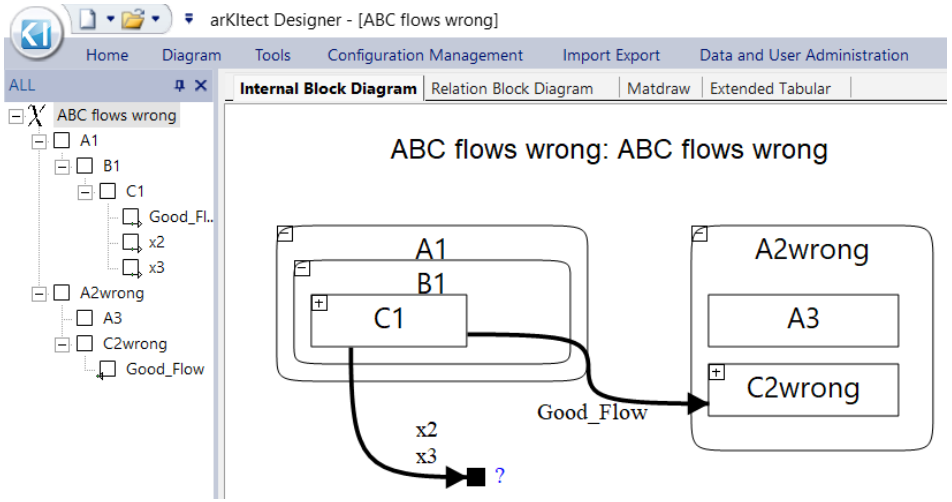
Une preuve simple: tous les flux produits sont consommés.

```
Fixpoint  msgfunction (d:Physical_flow)(p:direction)(g:Function): Prop :=
match g with
|CFunctionMM n => False
|CPhysical_flow_Function listData prod obj => (In d listData)/\(prod = p) \/ (msgfunction d p obj)
...
end.
Fixpoint msgComponent (d:Physical_flow)(p:direction)(g:Component){struct g}: Prop :=
match g with
| CComponent n => False
| CFunction_Component lf s => ( fold_left or (map (msgFunction d p) lf ) False ) \/ (msgComponent p s)
...
end.
Fixpoint msgsystem (d:Physical_flow)(p:direction)(g:System){struct g}: Prop :=
match g with
| CSystemMM n => False
| CComponent_SystemMM lf s => ( fold_left or (map (msgComponent d p) lf ) False ) \/ (msgsystem d p s)
end.

Definition EveryConsumedDataIsProduced(s: System) :=
forall (d:Physical_flow), (msgsystem d input s)-> (msgsystem d output s).

Goal (EveryConsumedDataIsProduced Steam_Generator_Boiler_1).
intro. simpl. tauto.
Qed.

Definition EveryProducedDataIsConsumed(s: System) :=
forall (d:Physical_flow), (msgsystem d output s)-> (msgsystem d input s).
```

Exemple



Exemple de système complet



Exemple de système incomplet
Flux x2, x3 n'ont pas de consommateurs

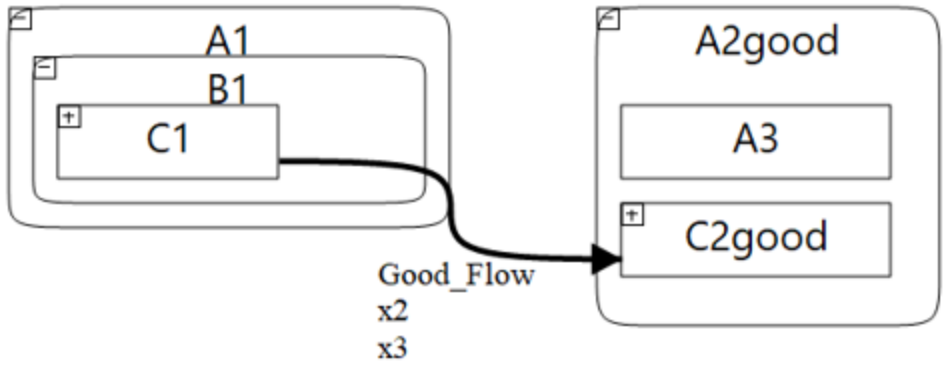
```
Definition Good_Flow := Cflow 85899345922.
Definition x2 := Cflow 94489280513.
Definition x3 := Cflow 94489280514.
Definition C1 := Cflow_C [ x2; Good_Flow; x3 ] output (CC 85899345921).
Definition B1 := CC_B [ C1 ] (CB 8589934607).
Definition A1 := CB_A [ B1 ] (CA 8589934605).
Definition A3 := CA 25769803779.
```

```
Definition C2good := Cflow_C [ x2; Good_Flow; x3 ] input (CC 60129542149).
Definition A2good := CA_A [ A3 ] (CC_A [ C2good ] (CA 8589934606)).
Definition ABC_flows_good :=
           CA_ABC_flows [ A1; A2good ] (CABC_flows 0).
```

```
Definition C2wrong := Cflow_C [ Good_Flow ] input (CC 60129542149).
Definition A2wrong := CA_A [ A3 ] (CC_A [ C2wrong ] (CA 8589934606)).
Definition ABC_flows_wrong :=
           CA_ABC_flows [ A1; A2wrong ] (CABC_flows 0).
```

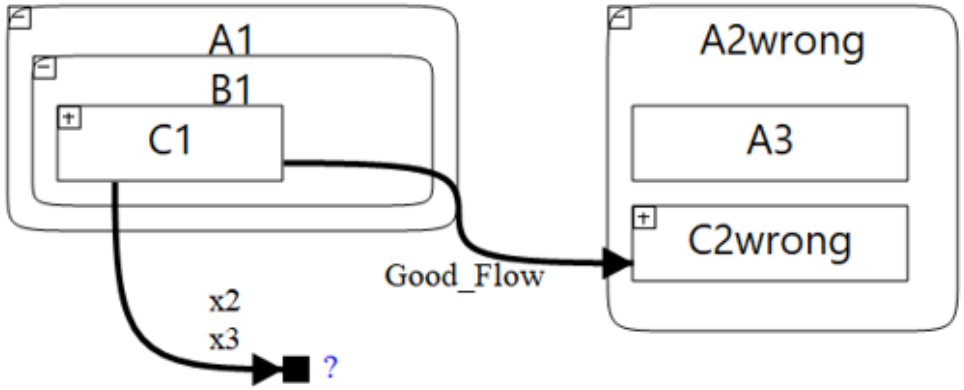
Exemple de système complet

```
Theorem try1 : (EveryProducedDataIsConsumed_flow_b ABC_flows_good) = true.
compute.
 trivial.
Qed.

1 goal
                                                                    (1/1)
_____________________________________________________________________
true = true
```

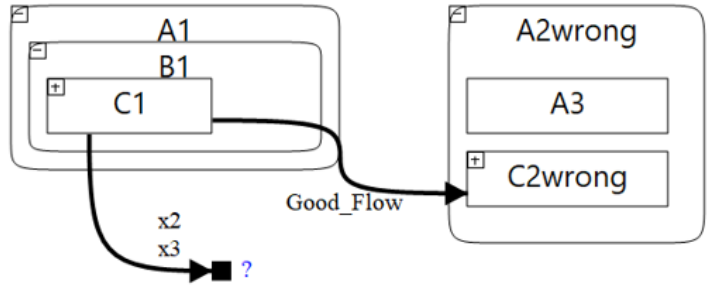


Exemple de système incomplet

```
Theorem try2 : (EveryProducedDataIsConsumed_flow_b ABC_flows_wrong) = true.
compute.
 trivial.
Abort All.

1 goal
                                                        (1/1)
false = true
```



Exemple de système incomplet




```
Theorem try2study : (EveryProducedDataIsConsumed_flow_b ABC_flows_wrong) = true.
unfold EveryProducedDataIsConsumed_flow_b.
simpl allFlowsABC_flows.
```

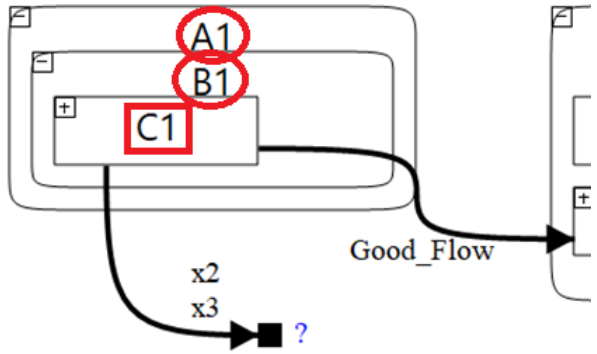
```
1 goal
_____________________________________(1/1)
forallb
  (fun d : flow =>
   if msgABC_flows_flow_b d output ABC_flows_wrong
   then msgABC_flows_flow_b d input ABC_flows_wrong
   else true) [x2; Good_Flow; x3; Good_Flow] = true
```

```
unfold forallb.
repeat rewrite andb_true_iff.
repeat split.
```

```
2 goals
_____________________________________(1/2)
(if msgABC_flows_flow_b x2 output ABC_flows_wrong
 then msgABC_flows_flow_b x2 input ABC_flows_wrong
 else true) = true
_____________________________________(2/2)
(if msgABC_flows_flow_b x3 output ABC_flows_wrong
 then msgABC_flows_flow_b x3 input ABC_flows_wrong
 else true) = true
```

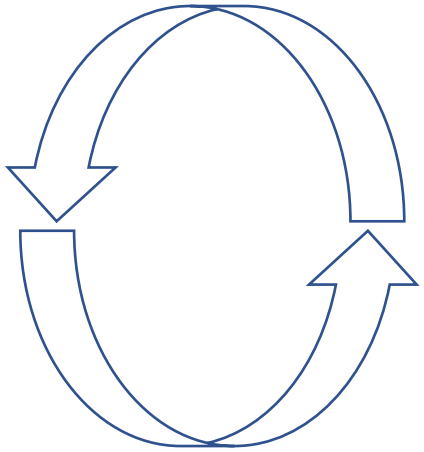
Exemple de système incomplet

```
set (problem_produced1 := msgABC_flows_flow_b x2 input ABC_flows_wrong).
set (problem_consumed1 := msgABC_flows_flow_b x2 output ABC_flows_wrong).
 …
assert (locate problem out : problem consumed1 = false).
rewrite hproblem.
repeat rewrite orb_false_iff.
repeat split.
unfold A1.
```



(1/3)

msgA_flow_b x2 output A1 = false



```
rewrite msgA_flow_b_unfold.
unfold existsb.
repeat rewrite orb_false_iff.
repeat split.
unfold B1.
rewrite msgB_flow_b_unfold.
unfold existsb.
repeat rewrite orb_false_iff.
repeat split.
unfold C1.
```

msgB_flow_b x2 output B1 = false

msgC_flow_b x2 output C1 = false

AMDE

Component › Failure mode — Requirement › Risk reduction means

Function › Failure effect — Cause

Composant / Groupement de composants	Fonction Technique	Mode de défaillance	Effets de la défaillance		Causes		Parade (Moyen de réduction du risque)					
			Description	Risque pression (P) / Radiopro (R)	Phase de vie	Description cause	Prévention des causes	Mode de preuve documentaire	Solution retenue	Détection des causes	Mode de preuve documentaire	Solution retenue
Virole	Résister à la pression et autres charges avec des facteurs de sécurité suffisant	Instabilité plastique	E1, E3, E5	P / R	Conception	Méthode inadéquate pour le choix de la géométrie	Utilisation d'une méthode de calcul éprouvée intégrant des facteurs de sécurité	Note de dimensionnement (méthode retenue)	Utilisation du RCC-M B3300, B3200	- Vérification du dimensionnement - Etude du comportement mécanique - Epreuve hydraulique	- Note de dimensionnement - DAC - PV d'épreuve	Utilisation du RCC-M 3200 Utilisation du RCC-M B5000 + Annexe ZZ
composants ou groupes de composants	*Fonction technique*	*cf § 5.5.1.2*	*Identification des événements redoutés*	*Risque pression (oui/non)* *Risque radiopro (oui/non)*		*Typologie de scénario (cf §5.5.1.4)*	*Objectif permettant de réduire l'apparition de la cause*	*Preuve documentaire pour la prévention*	*Moyen utilisé le cas échéant*	*Objectif permettant de détecter l'apparition de la cause ou des conséquences de la cause avant la phase d'exploitation*	*Preuve documentaire pour la détection*	*Moyen utilisé le cas échéant*

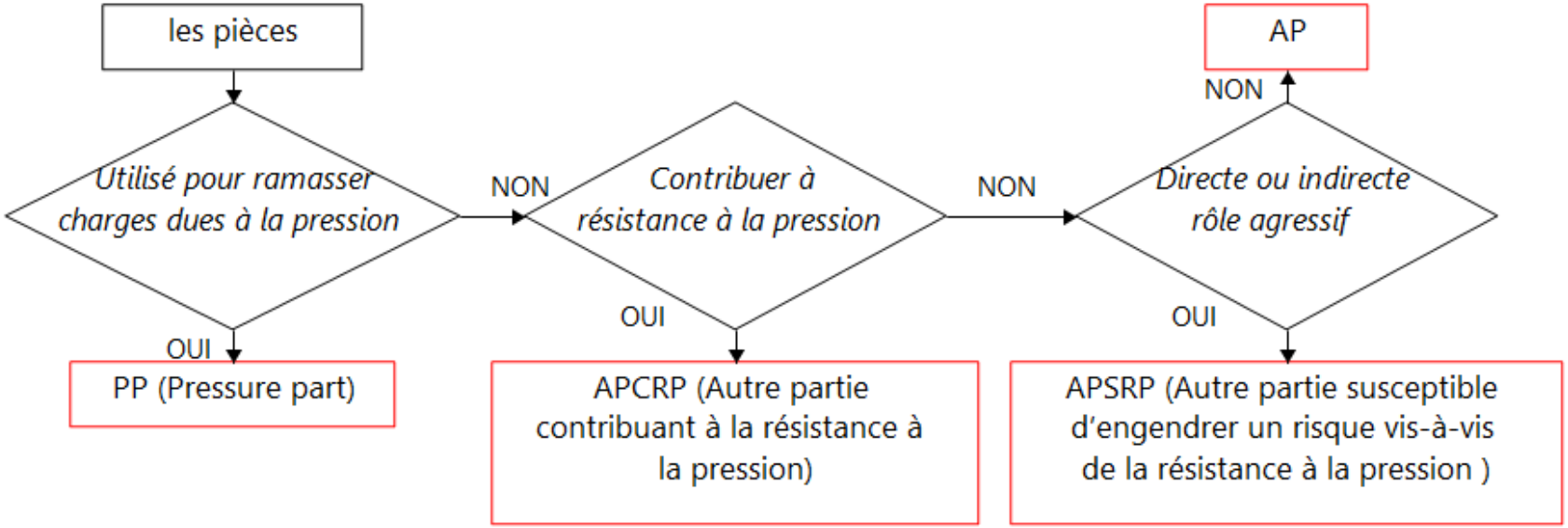
Propriétés de l'AMDE

- Revue exhaustive de tous les couples fonction/composant
  - Toutes les interfaces ont un producteur et au moins un consommateur
  - Tous les composants ont au moins une fonction allouée.
  - Chaque couple fonction/composant a au moins une ligne AMDE

- Allocation des exigences
  - Assurer la cohérence des liens de traçabilité

Propriétés des AMDE: Conformément aux lois et normes

Exemple:
Classification des composants de l'équipement en function de leur rôle et leur resistance à la pression et/ou à des milieux activés (French Nuclear Safety Authority Guidelines).



Conclusion

Modélisation système et preuves formelles:

- modèle DSL construit
- traduisant des modèles système dans un DSL
- traduction du DSL vers COQ.
- création automatique de fonctions d'assistance
- simplifié le travail des ingénieurs système

Perspective:

- vérifier les propriétés d'exhaustivité et d'exactitude et signaler les anomalies à l'utilisateur
- étudier des objectifs de haut niveau
- création de vues de données (filtres)