Étude de cas COMON

Client :	<->
Affaire/Projet :	COMON
Référence :	COMON/SPF 5 Annexe D
Révision :	0.1
État :	PRE
Date :	25 avril 2012
Classification :	Public
Nombre d'annexes :	0

Table des matières

1 Introduction

Ce document présente les tests effectués avec les outils Verimag autour de l'étude de cas définie dans le [SPF1] et qui représente un sous ensemble fictif mais représentatif d'un circuit hydraulique de centrale nucléaire (voir. Figure 1).

Nous commençons par décrire le cas d'étude et expliquer comment nous interagissons avec quand il est entièrement simulé dans Alices. Puis, nous présentons comment nous avons formalisé quelques scénarios de fonctionnement du cas d'étude (cf. chapitre 2). Ensuite, nous montrons comment nous avons formalisé les attendus du système et nous discutons la couverture obtenue avec les différents scénarios (cf. chapitre 3). Enfin, nous présentons succinctement les différences induites par le remplacement du N2 informatisé Alices par Adacs.

Il est conseillé d'avoir lu le SPF5 (données formalisées) avant de lire ce document.

1.1 Description du cas d'étude

Ce cas d'étude est inspiré de la topologie d'un circuit typique de centrale nucléaire. Il se compose de deux boucles primaires contenant chacune une bâche, une consigne de chauffe ainsi que plusieurs vannes

(TOR et réglantes), pompes et capteurs (température, niveau, pression et débit). Ces deux boucles alimentent le circuit complet qui se contient une bâche, l'échangeur avec le circuit d'eau froide ainsi que divers autres vannes, pompes et capteurs.

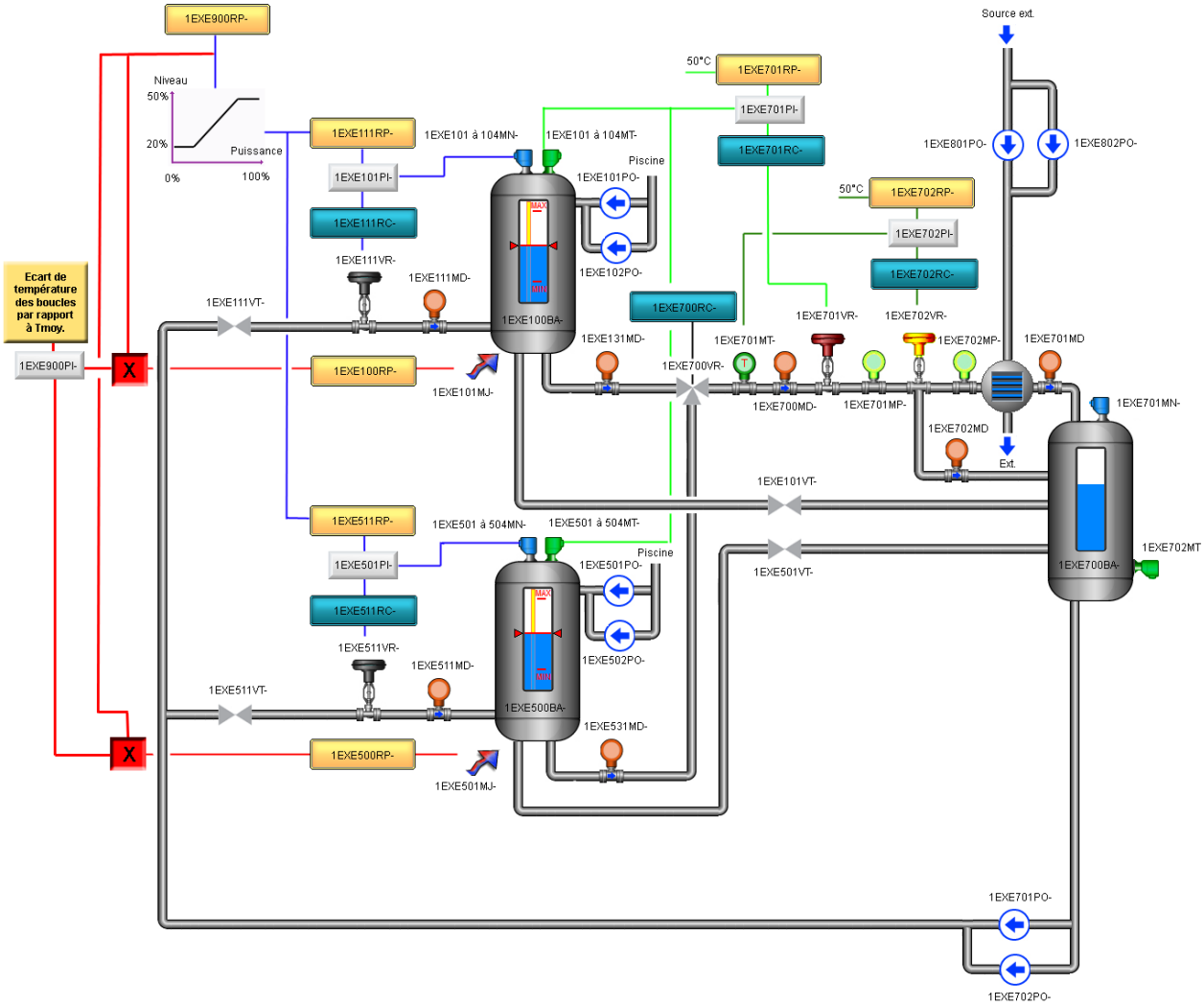

FIGURE 1 – Représentation du cas d'étude

Plus techniquement :

- Un relais de point de consigne 1EXE900RP pilote une puissance de chauffe appliquée en parallèle sur les deux bâches des boucles 1 et 2.
- Deux relais de point de consigne 1EXE100RP et 1EXE500RP permettent de surcharger en mode manuel la puissance appliquée sur les deux boucles sachant que la somme de ces puissances ne peut en aucun cas excéder la consigne fixée au 1EXE900RP.
- Une régulation de puissance permet, lorsque 1EXE100RP et 1EXE500RP sont en mode automatique, de répartir automatiquement la puissance de consigne sur les deux boucles en fonction de l'écart de température entre chaque boucle et la température moyenne du circuit.

- La mesure de la puissance appliquée détermine une consigne de niveau d'eau sur ces deux bâches si les relais de point de consignes correspondants 1EXE111RP et 1EXE511RP sont en mode automatique.
- La régulation des niveaux est assurée par comparaison à la mesure et par l'action des deux vannes réglantes 1EXE111VR et 1EXE511VR.
- Le circuit évacue ses calories par un échangeur dont l'efficacité dépend du débit des pompes 1EXE801 et 802PO et de la température de la source froide.
- La température est contrôlée par deux régulations distinctes et à fonctionnement exclusif.
- La première de ces régulations par la vanne réglante 1EXE701VR contrôle la température en modifiant le débit général du circuit sachant que la 1EXE702VR est forcée en manuel à 100% d'ouverture. Cette régulation modifie le débit global passant dans l'échangeur. Elle est précise et souple mais provoque la sollicitation des deux régulations de niveau.
- La deuxième régulation par la vanne réglante 1EXE702VR contrôle la température en contournant tout ou partie du débit d'eau passant par l'échangeur. Cette régulation sollicite très peu les régulations de niveau puisque le débit global du circuit reste quasi-constant.
- Chaque boucle peut être isolée ou vidangée. Elles sont également pourvues de systèmes d'injection d'eau de sécurité (1EXE101, 102, 501 et 502PO).
- Une bâche 1EXE700BA récupère en entrée les débits sortant de l'échangeur et de son contournement. Elle joue le rôle de tampon. Elle est assez volumineuse pour qu'elle ne constitue pas une contrainte de premier ordre sur notre étude de cas. Cependant, elle est dimensionnée pour qu'une fuite ou qu'un débordement important des bâches finisse par provoquer une vidange du circuit.
- À la sortie de ce réservoir, deux pompes 1EXE701 et 702PO assurent l'alimentation des bâches en eau.

Dans la suite, nous présentons les variables d'entrées/sorties du cas d'étude. Les noms de ces variables suivent une convention permettant de facilement faire le lien avec les noms utilisés dans Alices. Par exemple : `Regulator_N2_900RP_ORDER_VAL` équivaut à `Nom_VARIABLE`. `Nom` étant le nom dans Alices de cet objet à savoir un `Regulator` du niveau `N2` nommé `900RP`. `VARIABLE` étant une entrée/sortie de cet objet, à savoir ici l'entrée `ORDER_VAL` du `Regulator`.

1.2 Entrées

Nous pouvons agir sur le système par différents moyens. Il s'agit soit de commandes censées provenir de l'opérateur, soit d'injection de pannes :

- Modification de la consigne de puissance (entre 0 et 100%)

```
Regulator_N2_900RP_ORDER_VAL : real;
```

- Modification de la répartition de la puissance entre les deux boucles (modification de la consigne, changement de mode auto/manu)

```
Regulator_N2_100RP_CMD_AUTO_PO : bool;
Regulator_N2_100RP_CMD_MANU_PO : bool;
Regulator_N2_100RP_ORDER_VAL : real;
Regulator_N2_500RP_CMD_AUTO_PO : bool;
Regulator_N2_500RP_CMD_MANU_PO : bool;
Regulator_N2_500RP_ORDER_VAL : real;
```

- Modification des consignes de niveau et de température de RPC qui contrôlent les vannes réglantes 111 et 511VR

```
Regulator_N2_111RC_CMD_AUTO_PO: bool;
Regulator_N2_111RC_CMD_MANU_PO: bool;
Regulator_N2_111RC_ORDER_VAL: real;
Regulator_N2_111RP_CMD_AUTO_PO: bool;
Regulator_N2_111RP_CMD_MANU_PO: bool;
Regulator_N2_111RP_ORDER_VAL: real;
Regulator_N2_511RC_CMD_AUTO_PO: bool;
Regulator_N2_511RC_CMD_MANU_PO: bool;
Regulator_N2_511RC_ORDER_VAL : real;
Regulator_N2_511RP_CMD_AUTO_PO : bool;
Regulator_N2_511RP_CMD_MANU_PO : bool;
Regulator_N2_511RP_ORDER_VAL : real;
```

– Modification de la position des vannes réglantes 700, 701 et 702VR

```
Regulator_N2_700RC_ORDER_VAL : real;
Regulator_N2_701RC_CMD_AUTO_PO : bool;
Regulator_N2_701RC_CMD_MANU_PO : bool;
Regulator_N2_701RC_ORDER_VAL : real;
Regulator_N2_701RP_CMD_AUTO_PO : bool;
Regulator_N2_701RP_CMD_MANU_PO : bool;
Regulator_N2_701RP_ORDER_VAL : real;
Regulator_N2_702RC_CMD_AUTO_PO : bool;
Regulator_N2_702RC_CMD_MANU_PO : bool;
Regulator_N2_702RC_ORDER_VAL : real;
Regulator_N2_702RP_CMD_AUTO_PO : bool;
Regulator_N2_702RP_CMD_MANU_PO : bool;
Regulator_N2_702RP_ORDER_VAL : real;
```

– Ouverture et fermeture des vannes TOR 101, 111, 501 et 511VT

```
Actuator_N2_101VT_CMD_OFF_PO: bool;
Actuator_N2_101VT_CMD_ON_PO: bool;
Actuator_N2_111VT_CMD_OFF_PO: bool;
Actuator_N2_111VT_CMD_ON_PO: bool;
Actuator_N2_501VT_CMD_OFF_PO: bool;
Actuator_N2_501VT_CMD_ON_PO: bool;
Actuator_N2_511VT_CMD_OFF_PO: bool;
Actuator_N2_511VT_CMD_ON_PO: bool;
```

– Mise en marche et arrêt des pompes 101, 102, 501, 502, 701, 702, 801 et 802PO

```
Actuator_N2_101PO_CMD_OFF_PO: bool;
Actuator_N2_101PO_CMD_ON_PO: bool;
Actuator_N2_102PO_CMD_OFF_PO: bool;
Actuator_N2_102PO_CMD_ON_PO: bool;
Actuator_N2_501PO_CMD_OFF_PO: bool;
Actuator_N2_501PO_CMD_ON_PO: bool;
Actuator_N2_502PO_CMD_OFF_PO: bool;
Actuator_N2_502PO_CMD_ON_PO: bool;
Actuator_N2_701PO_CMD_OFF_PO: bool;
Actuator_N2_701PO_CMD_ON_PO: bool;
Actuator_N2_702PO_CMD_OFF_PO: bool;
Actuator_N2_702PO_CMD_ON_PO: bool;
```

```
Actuator_N2_801PO_CMD_OFF_PO: bool;
Actuator_N2_801PO_CMD_ON_PO: bool;
Actuator_N2_802PO_CMD_OFF_PO: bool;
Actuator_N2_802PO_CMD_ON_PO: bool;
```

– Mise en défaut de certains équipements (tous les objets de l'étude de cas peuvent êtrent mis en pannes, nous présentons uniquement les pannes utilisées dans le scénario de pannes, cf. section 2.4)

```
-- Fuite sur la bache 100
EXE100BA_Panne1_OnOff: bool;
EXE100BA_Panne1_Valeur: real [0.0; 100.0];
-- Blocage du capteur de niveau 101MN
EXE101MN_Panne2_OnOff: bool;
EXE101MN_Panne2_Valeur: real [50.0; 500.0];
-- Blocage du capteur de niveau 102MN
EXE102MN_Panne2_OnOff: bool;
EXE102MN_Panne2_Valeur: real [50.0; 500.0];
-- Blocage du capteur de niveau 103MN
EXE103MN_Panne2_OnOff: bool;
EXE103MN_Panne2_Valeur: real [50.0; 500.0];
-- Blocage du capteur de niveau 104MN
EXE104MN_Panne2_OnOff: bool;
EXE104MN_Panne2_Valeur: real [50.0; 500.0];
-- Blocage en position ferme des pompes 701 et 702PO
EXE701PO_Panne1_OnOff: bool;
EXE702PO_Panne1_OnOff: bool
```

1.3 Sorties

Comme décrit dans la section 5.3 du CPR4 nous avons accès à toutes les variables de la zone d'échange. Mais nous utilisons principalement les sorties du niveau N2 pour les oracles, soit environ 850 variables.

2 Environnements

En se basant sur les scénarios décrits dans le [SPF1] nous avons proposé plusieurs scénarios :
- changement de puissance de chauffe,
- changement des boucles utilisées,
- changement de systèmes de régulation de température,
- déclenchement de pannes.

Chaque scénario représente une généralisation d'un scénario imaginé pour l'étude de cas. Toutefois, ces scénarios peuvent être combinés ensemble. Soit par une mise en parallèle ce qui conduirait à faire plusieurs changements en même temps comme une baisse de puissance de chauffe et le déclenchement d'une ou de plusieurs pannes. Soit en les séquentialisant pour effectuer un scénario pendant un certain temps puis basculer sur un autre scénario.

2.1 Scénario de changement de puissance

Le circuit hydraulique de l'étude de cas doit évacuer la chaleur reçue quelle qu'elle soit. Ce scénario fait varier la puissance en entrée du circuit entre 0 et 100%, en s'assurant toutefois que le système soit dans un état stable avant chaque changement.

L'algorithme consiste à attendre la stabilisation du système, par rapport aux températures et niveaux des bâches des deux boucles, en utilisant le nœud `is_stable` qui a été présenté dans la section 2.3 du SPF5. Ensuite, une nouvelle valeur est choisie pour la puissance du système.

```
exist est_stable : bool in
exist est_stable_T1, est_stable_T2 : bool in
exist est_stable_N1, est_stable_N2 : bool in

let initialisation =
  Regulator_N2_900RP_ORDER_VAL = 100.0 and
  est_stable = false and
  est_stable_T1 = false and
  est_stable_T2 = false and
  est_stable_N1 = false and
  est_stable_N2 = false
in
initialisation
fby
loop {
  run est_stable_T1 := is_stable(Analog_Internal_N2_100MT_AVALUE_v) in
  run est_stable_T2 := is_stable(Analog_Internal_N2_500MT_AVALUE_v) in
  run est_stable_N1 := is_stable(Analog_Internal_N2_100MN_AVALUE_v) in
  run est_stable_N2 := is_stable(Analog_Internal_N2_500MN_AVALUE_v) in
    let est_stable =
      est_stable_T1 and est_stable_T2 and
      est_stable_N1 and est_stable_N2 in

  -- On attend la stabilite du systeme
  loop {maintient(Regulator_N2_900RP_ORDER_VAL) and est_stable = false}
  fby
  between(Regulator_N2_900RP_ORDER_VAL, 0.0, 100.0)
}
```

Toutefois, pour plus de réalisme le changement de la consigne de puissance ne doit pas s'effectuer instantanément mais plutôt par petits changements successifs jusqu'à arriver à la nouvelle consigne en utilisant le nœud `change_consigne_pas_a_pas` présenté dans la section 3.2.2 du SPF5.

L'algorithme principal devient alors :

```
exist ConsigneTarget : real in
exist est_stable : bool in
exist est_stable_T1, est_stable_T2 : bool in
exist est_stable_N1, est_stable_N2 : bool in

let initialisation =
  Regulator_N2_900RP_ORDER_VAL = 100.0 and
  est_stable = false and
  est_stable_T1 = false and
  est_stable_T2 = false and
  est_stable_N1 = false and
  est_stable_N2 = false
in
initialisation
fby
loop {
  run est_stable_T1 := is_stable(Analog_Internal_N2_100MT_AVALUE_v) in
  run est_stable_T2 := is_stable(Analog_Internal_N2_500MT_AVALUE_v) in
  run est_stable_N1 := is_stable(Analog_Internal_N2_100MN_AVALUE_v) in
  run est_stable_N2 := is_stable(Analog_Internal_N2_500MN_AVALUE_v) in
    let est_stable =
      est_stable_T1 and est_stable_T2 and
      est_stable_N1 and est_stable_N2
     in

  -- On attend la stabilite du systeme
  loop {maintient(Regulator_N2_900RP_ORDER_VAL) and est_stable = false}

  fby -- Choisir une nouvelle consigne
  { maintient(Regulator_N2_900RP_ORDER_VAL) and
  between(ConsigneTarget, 0.0, 100.0) }

  fby
  assert maintient(ConsigneTarget) in
  run Regulator_N2_900RP_ORDER_VAL :=
    change_consigne_pas_a_pas(est_stable,
                              pre Regulator_N2_900RP_ORDER_VAL,
                              pre ConsigneTarget,
                              PAS)
    in
      while (Regulator_N2_900RP_ORDER_VAL <> ConsigneTarget)
}
```

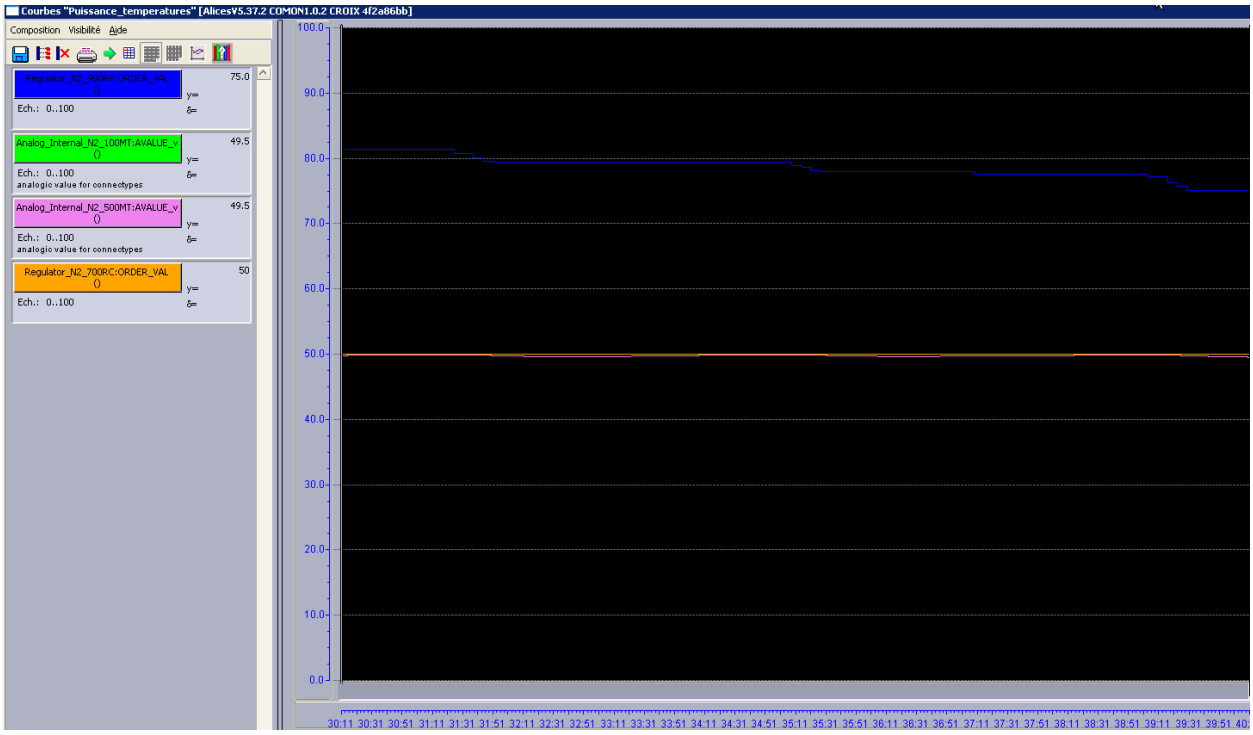

FIGURE 2 – Changements de puissance pas à pas. La courbe du haut (bleue) indique les changements de la consigne de puissance de chauffe. Ces changement sont conditionnés par la stabilité du système, d'où le fait que la courbe bleue n'évolue plus quand les capteurs de températures ne sont plus stables (courbes du bas, vertes et rose superposées).

2.2 Scénario de changement des boucles utilisées

Le circuit hydraulique de l'étude de cas fonctionne en mode nominal sur deux boucles : la première est associée à la bâche 1EXE100BA, et la seconde à la bâche 1EXE500BA. Il est toutefois possible de passer sur une seule des deux boucles, voir d'être en un tiers de puissance sur une boucle et deux tiers de puissance sur l'autre boucle.

L'algorithme consiste à attendre la stabilisation des températures et niveaux des bâches des deux boucles, en utilisant le nœud `is_stable`. Ensuite, une nouvelle valeur est choisie pour l'ouverture de la vanne réglante 700VR. Cette ouverture influe sur la contribution des deux boucles et elle possède un certain nombre de positions préférées définies dans le nœud `choix_boucles_pondere`.

```
let choix_boucles_pondere(ConsigneTarget, preConsigne : real) : trace =
 {
   |   preConsigne = 50.0  and (ConsigneTarget = 66.0 or ConsigneTarget = 33.0)
   |   preConsigne = 33.0  and ConsigneTarget = 0.0
   |2: preConsigne = 33.0  and ConsigneTarget = 50.0
   |   preConsigne = 66.0  and ConsigneTarget = 100.0
   |2: preConsigne = 66.0  and ConsigneTarget = 50.0
   |   preConsigne = 100.0 and ConsigneTarget = 66.0
   |   preConsigne = 0.0   and ConsigneTarget = 33.0
  }
```

Toutefois, pour plus de réalisme le changement de cette consigne d'ouverture ne doit pas s'effectuer instantanément mais plutôt par petits changements successifs jusqu'à arriver à la nouvelle consigne en utilisant le nœud `change_consigne_pas_a_pas`.

L'algorithme principal étant :

```
exist ConsigneTarget : real in
exist est_stable : bool in
exist est_stable_T1, est_stable_T2 : bool in
exist est_stable_N1, est_stable_N2 : bool in

let initialisation =
  Regulator_N2_700RC_ORDER_VAL = 50.0 and
  est_stable = false and
  est_stable_T1 = false and
  est_stable_T2 = false and
  est_stable_N1 = false and
  est_stable_N2 = false
in
initialisation
fby
loop {
  run est_stable_T1 := is_stable(Analog_Internal_N2_100MT_AVALUE_v) in
  run est_stable_T2 := is_stable(Analog_Internal_N2_500MT_AVALUE_v) in
  run est_stable_N1 := is_stable(Analog_Internal_N2_100MN_AVALUE_v) in
  run est_stable_N2 := is_stable(Analog_Internal_N2_500MN_AVALUE_v) in
    let est_stable =
      est_stable_T1 and est_stable_T2 and
      est_stable_N1 and est_stable_N2 in

  -- On attend la stabilite du systeme
  loop {maintient(Regulator_N2_700RC_ORDER_VAL) and est_stable = false}
  fby -- Choisir une nouvelle consigne
```

```
    { maintient(Regulator_N2_700RC_ORDER_VAL)
   &>
    choix_boucles_pondere(ConsigneTarget, pre Regulator_N2_700RC_ORDER_VAL) }

   fby
   assert maintient(ConsigneTarget) in
   run Regulator_N2_700RC_ORDER_VAL :=
     change_consigne_pas_a_pas(est_stable,
                               pre Regulator_N2_700RC_ORDER_VAL,
                               pre ConsigneTarget,
                               PAS)
   in
     while (Regulator_N2_700RC_ORDER_VAL <> ConsigneTarget)
}
```

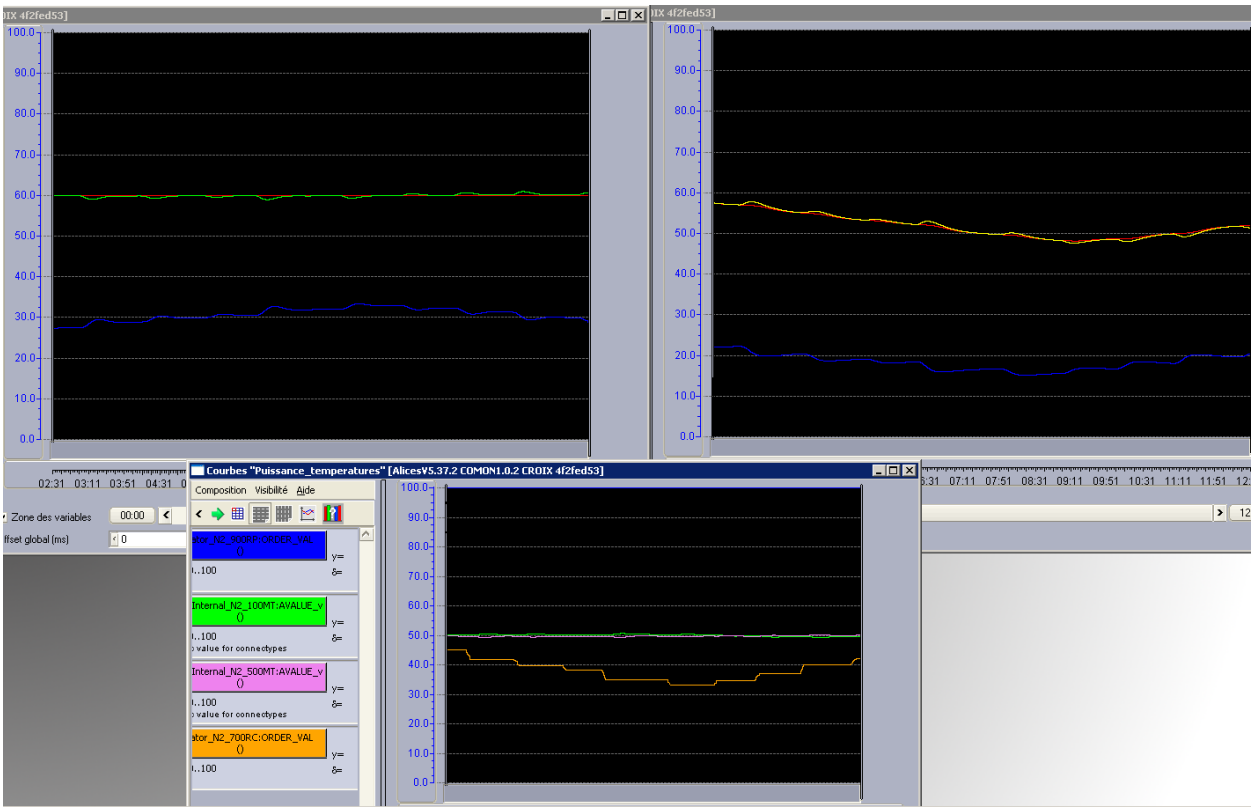

FIGURE 3 – Changements de boucles pas à pas. La courbe orange (en bas) montre que le circuit est passé en mode 2/3 - 1/3 en mettant une boucle à 50% puis le circuit revient vers un fonctionnement à deux boucles, le tout en pas à pas. Les courbes du haut montrent l'évolution des capteurs des deux boucles primaires. On voit sur les courbes de droite que toutes les valeurs baissent puis remontent.

2.3 Scénario de changement de système de régulation de température

Le circuit hydraulique de l'étude de cas peut être régulé soit par la vanne 702 soit par la 701. Ce scénario nous permet de jouer le rôle d'un opérateur qui basculerait indéfiniment d'une régulation à une autre, en s'assurant toutefois que le système soit dans un état stable avant chaque changement. Le fonctionnement des ces régulations devant être exclusif, il nous suffit de tester le mode de ces vannes pour connaître celle qui est utilisée.

```
if Regulator_N2_702RC_MODE
then
  [...] -- C'est la vanne 702 qui est utilisee
else
  [...] -- C'est la vanne 701 qui est utilisee
```

L'état initial de la simulation considère que la 702 régule le système. Passer à la 701 se fait alors en 3 étapes :

1- On passe la 701RP et la 701RC en automatique et 702RC en manuelle.

```
let passage_a_701_etape_1  = {
      Regulator_N2_701RC_CMD_AUTO_PO and
      Regulator_N2_701RP_CMD_AUTO_PO and
  not Regulator_N2_702RC_CMD_AUTO_PO and
      Regulator_N2_702RP_CMD_AUTO_PO and
  not Regulator_N2_701RC_CMD_MANU_PO and
      Regulator_N2_702RC_CMD_MANU_PO
} in
```

2- La 702RC étant passée en manuel, l'état de la 702RP passe lui aussi en manuel, on arrête alors de passer cette commande.

```
let passage_a_701_etape_2  = {
      Regulator_N2_701RC_CMD_AUTO_PO and
      Regulator_N2_701RP_CMD_AUTO_PO and
  not Regulator_N2_702RC_CMD_AUTO_PO and
  not Regulator_N2_702RP_CMD_AUTO_PO and
  not Regulator_N2_701RC_CMD_MANU_PO and
  not Regulator_N2_702RC_CMD_MANU_PO
} in
```

3- Enfin, toutes les commandes sont positionnées à faux.

```
let etape_3  = {
  not Regulator_N2_701RC_CMD_AUTO_PO and
  not Regulator_N2_701RP_CMD_AUTO_PO and
  not Regulator_N2_702RC_CMD_AUTO_PO and
  not Regulator_N2_702RP_CMD_AUTO_PO and
  not Regulator_N2_701RC_CMD_MANU_PO and
  not Regulator_N2_702RC_CMD_MANU_PO
} in
```

Des étapes similaires permettent de passer de la régulation 701 à la 702.

```
let passage_a_702_etape_1  = {
```

```
  not Regulator_N2_701RC_CMD_AUTO_PO and
      Regulator_N2_701RP_CMD_AUTO_PO and
      Regulator_N2_702RC_CMD_AUTO_PO and
      Regulator_N2_702RP_CMD_AUTO_PO and
      Regulator_N2_701RC_CMD_MANU_PO and
  not Regulator_N2_702RC_CMD_MANU_PO
} in

let passage_a_702_etape_2  = {
  not Regulator_N2_701RC_CMD_AUTO_PO and
  not Regulator_N2_701RP_CMD_AUTO_PO and
      Regulator_N2_702RC_CMD_AUTO_PO and
      Regulator_N2_702RP_CMD_AUTO_PO and
  not Regulator_N2_701RC_CMD_MANU_PO and
  not Regulator_N2_702RC_CMD_MANU_PO
} in
```

Ces quelques combinateurs permettent de rendre l'algorithme plus lisible, en donnant un nom parlant à ces différentes étapes.

```
exist vanne_active : bool in
exist est_stable : bool in
exist est_stable_T1, est_stable_T2 : bool in
exist est_stable_N1, est_stable_N2 : bool in

let initialisation = {
  Regulator_N2_701RC_CMD_AUTO_PO = false and
  Regulator_N2_701RP_CMD_AUTO_PO = false and
  Regulator_N2_702RC_CMD_AUTO_PO = false and
  Regulator_N2_702RP_CMD_AUTO_PO = false and
  Regulator_N2_701RC_CMD_MANU_PO = false and
  Regulator_N2_702RC_CMD_MANU_PO = false and
  est_stable = false and
  est_stable_T1 = false and
  est_stable_T2 = false and
  est_stable_N1 = false and
  est_stable_N2 = false and
  vanne_active = Regulator_N2_702RC_MODE
} in

let pas_de_changements = {
  maintient_b(Regulator_N2_701RC_CMD_AUTO_PO) and
  maintient_b(Regulator_N2_701RC_CMD_MANU_PO) and
  maintient_b(Regulator_N2_701RP_CMD_AUTO_PO) and
  maintient_b(Regulator_N2_702RC_CMD_AUTO_PO) and
  maintient_b(Regulator_N2_702RC_CMD_MANU_PO) and
  maintient_b(Regulator_N2_702RP_CMD_AUTO_PO) and
  maintient_b(vanne_active)
} in
```

Le nœud principal initialise les différentes commandes puis se met en attente de la stabilité du système en utilisant le nœud `is_stable`. La stabilité est testée par rapport aux températures et niveaux des bâches des deux boucles. Ensuite, la régulation active est changée entre la régulation 701 et la régulation 702. Enfin, on laisse au système le temps d'effectuer le changement de régulation avant de refaire le test de stabilité.

```
initialisation
fby
loop {
  run est_stable_T1 := is_stable(Analog_Internal_N2_100MT_AVALUE_v) in
  run est_stable_T2 := is_stable(Analog_Internal_N2_500MT_AVALUE_v) in
  run est_stable_N1 := is_stable(Analog_Internal_N2_100MN_AVALUE_v) in
  run est_stable_N2 := is_stable(Analog_Internal_N2_500MN_AVALUE_v) in
    let est_stable =
      est_stable_T1 and est_stable_T2 and
      est_stable_N1 and est_stable_N2 in

  -- On attend la stabilite du systeme
  loop {pas_de_changements and est_stable = false}

  fby
  if vanne_active
  then  -- Si c'est la vanne 702 qui regule on bascule sur la 701
    passage_a_701_etape_1 and maintient_b(vanne_active)
  else  -- Si c'est la vanne 701 qui regule on bascule sur la 702
    passage_a_702_etape_1 and maintient_b(vanne_active)

  fby
  loop [2] pas_de_changements

  fby
  if vanne_active
  then  -- Si c'est la vanne 702 qui regule on bascule sur la 701
    passage_a_701_etape_2 and maintient_b(vanne_active)
  else  -- Si c'est la vanne 701 qui regule on bascule sur la 702
    passage_a_702_etape_2 and maintient_b(vanne_active)

  fby
  etape_3 and vanne_active = not pre vanne_active
  fby  -- On attend que le changement de regulation s'effectue
  loop [SMALL_DELAY] pas_de_changements
}
```

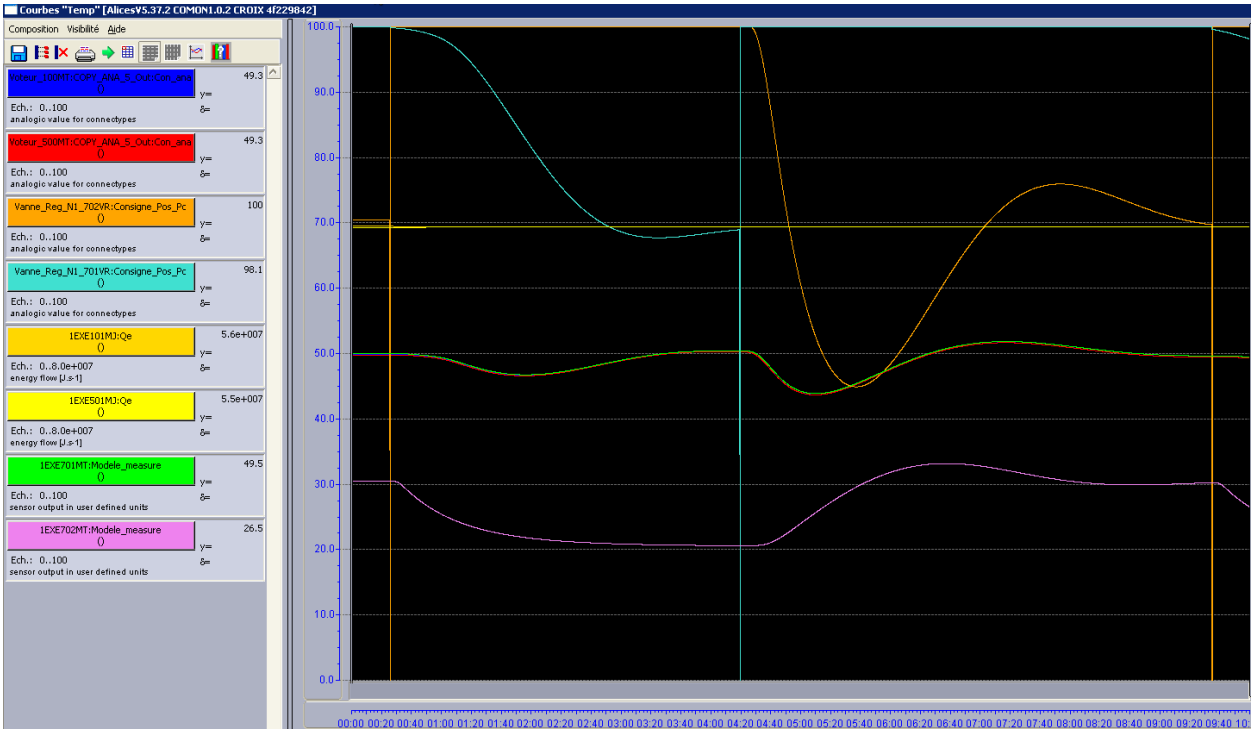

FIGURE 4 – Changements de régulation après stabilisation du système. La courbe bleue clair indique un passage sur la régulation 701. La courbe orange indique le passage sur la régulation 702. Et ainsi de suite, à chaque fois que le système se stabilise.

2.4 Scénario de déclenchement de pannes

Comme expliqué dans la section 1.2, nous avons la possibilité de mettre en panne les différents objets de l'étude de cas. Certains objets disposent même de plusieurs pannes (blocage, dérive ... etc.). L'idée de ce scénario est de générer des pannes au hasard afin de tester le fonctionnement de l'étude de cas lors de situations dégradées voir lors de situations d'urgence. Nous savons par ailleurs qu'il existe un certain nombre d'hypothèses sur le fonctionnement des objets du cas d'étude et le mode de fonctionnement du circuit hydraulique (cf. la section 3). À savoir que lorsque deux objets redondés sont en pannes (où bien au moins 3 sur 4 si quadri-redondés) cela va mener à une situation d'urgence avec déclenchement d'actions classées. C'est pour cela que notre algorithme va essayer de générer un maximum de pannes en évitant tant que possible une situation d'urgence.

Le scénario de déclenchement de pannes est une généralisation du nœud `gen_pannes` de la section 3.2.1 du SPF5. Dans sa version interactive, le scénario de pannes prends deux entrées : « n » et « reset » qui correspondent respectivement aux nombre de pannes souhaitées et à un relancement du choix des pannes.

```
let b2i(b:bool):int = if b then 1 else 0 in
let nb_pannes = b2i(EXE100BA_Panne1_OnOff) +
                b2i(EXE101MN_Panne2_OnOff) +
                b2i(EXE102MN_Panne2_OnOff) +
                b2i(EXE103MN_Panne2_OnOff) +
                b2i(EXE104MN_Panne2_OnOff) +
                b2i(EXE701PO_Panne1_OnOff) +
                b2i(EXE702PO_Panne1_OnOff)
in
let bouge_pas = maintient_b(EXE100BA_Panne1_OnOff)  and
                maintient  (EXE100BA_Panne1_Valeur) and
                maintient_b(EXE101MN_Panne2_OnOff)  and
                maintient  (EXE101MN_Panne2_Valeur) and
                maintient_b(EXE102MN_Panne2_OnOff)  and
                maintient  (EXE102MN_Panne2_Valeur) and
                maintient_b(EXE103MN_Panne2_OnOff)  and
                maintient  (EXE103MN_Panne2_Valeur) and
                maintient_b(EXE104MN_Panne2_OnOff)  and
                maintient  (EXE104MN_Panne2_Valeur) and
                maintient_b(EXE701PO_Panne1_OnOff)  and
                maintient_b(EXE702PO_Panne1_OnOff)
in
let Urgence = exactement_2_parmi_2(EXE701PO_Panne1_OnOff, EXE702PO_Panne1_OnOff)
              or
              au_moins_3_parmi_4(EXE101MN_Panne2_OnOff, EXE102MN_Panne2_OnOff,
                                 EXE103MN_Panne2_OnOff, EXE104MN_Panne2_OnOff)
in
loop {
  { (nb_pannes = n and not Urgence) |> nb_pannes = min_i(n, 7) }
  fby
  loop [minutes(0),minutes(5)] {
    bouge_pas and not reset
  }
}
```

En effet, soit le nombre de pannes est égal à « n », une entrée du nœud et on s'assure de ne pas générer une situation d'urgence, soit le nombre de pannes demandées est trop grand et nous allons générer une situation d'urgence. Ensuite, nous maintenons ces pannes entre 0 et 5 minutes avant d'en choisir d'autres. On sort

aussi de la boucle à la réception d'un « reset » (l'autre entrée du nœud).

Au lieu de laisser « n » et « reset » au choix de l'opérateur on peut aussi utiliser le nœud suivant pour les tirer au hasard et maintenir ce choix pendant un certain temps.

```
node n_reset() returns (n:int = 0; reset: bool = false) = {
  loop {
    loop [minutes(1),minutes(3)] {
      not reset and maintient_i(n)
    }
    fby
    between_i(n, 0, 8) and reset
  }
}
```

Le scénario de déclenchement de pannes fait appel à des entrées du cas d'étude qui n'entrent pas en conflit avec les autres scénarios. D'où le fait que nous pouvons exécuter ce scénario parallèlement à n'importe lequel des autres scénarios décrits plus haut.

3 Oracles

Les propriétés fonctionnelles du cas d'étude ont été décrites de deux manières équivalentes dans le [SPF1] : hiérarchiquement, et par rapport à des niveaux de tolérance. Cette dernière description se décline comme suit :

- garantir la sauvegarde de la centrale (sécuritaire),
- garantir le fonctionnement de la centrale,
- garantir le fonctionnement en mode dégradé,
- garantir le fonctionnement en mode nominal.

En plus de ces propriétés, nous avons ajouté la propriété de stabilisation du système et une propriété concernant les pannes et le déclenchement du mode classé.

3.1 Garantir la sauvegarde

Le système reste dans la limite de température et de niveau garantissant la sécurité des installations.

1. Rester entre 0 et 100°C dans chacune des bâches.
2. Avoir un niveau de bâche entre 0 et 30 m pour les 3 bâches.

On définit le nœud `plage_ok` qui vérifie qu'un capteur est correct, ainsi que les constantes définissant les limites de fonctionnement des différents types de capteurs (débits, niveaux, pressions et températures).

```
const MIN             : real =    0.0;
const MAX             : real =  100.0;
const MAX_DEBIT       : real = 1500.0;
const MAX_NIVEAU      : real =   30.0;
const MAX_PRESSION    : real =   50.0;
const MAX_TEMPERATURE : real =  100.0;

node plage_ok(min, max, x:real; inval:bool) returns (s:bool);
let
  s = (not inval) and (min <= x and x <= max);
tel
```

Ensuite, on instancie ce nœud avec les capteurs à surveiller : voteurs de température 100 et 500MT, voteur de niveau 100 et 500MN et le niveau de la troisième bâche 701MN.

```
capteurs_temp_baches =
  plage_ok(MIN, MAX_TEMPERATURE,
           Analog_Internal_N2_100MT_AVALUE_v, Analog_Internal_N2_100MT_INVAL_v)
  and
  plage_ok(MIN, MAX_TEMPERATURE,
           Analog_Internal_N2_500MT_AVALUE_v, Analog_Internal_N2_500MT_INVAL_v);

capteurs_niv_baches =
  plage_ok(MIN, MAX_NIVEAU,
           Analog_Internal_N2_100MN_AVALUE_v, Analog_Internal_N2_100MN_INVAL_v) and
  plage_ok(MIN, MAX_NIVEAU,
           Analog_Internal_N2_500MN_AVALUE_v, Analog_Internal_N2_500MN_INVAL_v) and
  plage_ok(MIN, MAX_NIVEAU,
           Analog_input_N2_701MN_AVALUE_v, Analog_input_N2_701MN_INVAL_v);

Prop_Sauvegarde = capteurs_temp_baches and capteurs_niv_baches;
```

La propriété de sauvegarde doit toujours être vérifiée quelque soit le mode de fonctionnement de l'étude de cas.

3.2 Garantir le fonctionnement

En plus de la garantie de sauvegarde, de la chaleur est apportée sur au moins une des deux boucles, et de la chaleur est évacuée par l'échangeur. On sépare cette propriété en plusieurs parties :

- de la chaleur est apportée sur une des deux boucles, au moins,
- les régulations des niveaux des boucles non isolées doivent être en fonctionnement,
- au moins l'une des deux régulations de température doit être en fonctionnement.

3.2.1 Chaleur apportée sur au moins une des deux boucles

```
chaleur_apportee =
  (Regulator_N2_100RP_AVALUE > 0.0 and not Regulator_N2_100RP_INVAL) or
  (Regulator_N2_500RP_AVALUE > 0.0 and not Regulator_N2_500RP_INVAL);
```

3.2.2 Les régulations des niveaux fonctionnent

```
regulation_niv =
  (not boucle2_isolee =>
      regulation_fonctionnelle(Regulator_N2_111RP_AVALUE,
                               Regulator_N2_111RP_MODE,
                               Regulator_N2_111RP_INVAL) )
  and
  (not boucle1_isolee =>
      regulation_fonctionnelle(Regulator_N2_511RP_AVALUE,
                               Regulator_N2_511RP_MODE,
                               Regulator_N2_511RP_INVAL) );
```

En ayant précédemment définie la notion de boucles isolées comme suit :

```
boucle1_isolee = EXE030SIT_LVALUE and
                 ((Analog_input_N2_531MD_AVALUE_v / 15.0) <= 5.0);
boucle2_isolee = EXE030SIT_LVALUE and
                 ((Analog_input_N2_131MD_AVALUE_v / 15.0) <= 5.0);
```

Et le nœud `regulation_fonctionnelle` comme tel :

```
node regulation_fonctionnelle(value:real; mode, inval:bool)
  returns (s:bool);
let
  s = mode => plage_ok(MIN, MAX, value, inval);
tel
```

3.2.3 Une des régulations de température fonctionne

```
regulation_temp =
  regulation_fonctionnelle(Regulator_N2_701RP_AVALUE,
                           Regulator_N2_701RP_MODE,
                           Regulator_N2_701RP_INVAL) or
  regulation_fonctionnelle(Regulator_N2_702RP_AVALUE,
                           Regulator_N2_702RP_MODE,
                           Regulator_N2_702RP_INVAL);
```

Ce qui nous donne :

```
Prop_Fonctionnement =
  chaleur_apportee and regulation_niv and regulation_temp;
```

La propriété de fonctionnement doit aussi toujours être vérifiée quelque soit le mode de fonctionnement de l'étude de cas.

3.3 Garantir le fonctionnement en mode dégradé

En plus de la garantie de sauvegarde, et de la garantie de fonctionnement, aucun des équipements non redondés n'est en défaut et au moins un des équipements redondés est fonctionnel. Pour les capteurs redondés, au moins 2 des 4 capteurs sont fonctionnels. On sépare cette propriété en plusieurs parties :

- les équipements non redondés doivent être fonctionnels si la boucle correspondante est en fonctionnement,
- la moitié, au moins, des équipements redondés doivent être fonctionnels si la boucle correspondante est en fonctionnement.

3.3.1 Les équipements non redondés fonctionnent

```
equipements_non_redondes =
  consignes and regulations and vannes and capteurs_non_redondes;
```

Nous avons groupés les vérifications par type d'objets : consignes, régulations, vannes et capteurs non redondés.

```
consignes =
  -- 2 boucles
  plage_ok(MIN, MAX, Regulator_N2_900RP_AVALUE, Regulator_N2_900RP_INVAL) and
  plage_ok(MIN, MAX, Regulator_N2_700RC_AVALUE, Regulator_N2_700RC_INVAL) and
  -- boucle 1
  ( not boucle2_isolee =>
      plage_ok(MIN, MAX_TEMPERATURE,
               Regulator_N2_100RP_AVALUE, Regulator_N2_100RP_INVAL)
  ) and
  -- boucle 2
  ( not boucle1_isolee =>
      plage_ok(MIN, MAX_TEMPERATURE,
               Regulator_N2_500RP_AVALUE, Regulator_N2_500RP_INVAL)
  );
```

```
regulations =
  -- 2 boucles
  regulation_fonctionnelle(Regulator_N2_701RP_AVALUE,
                           Regulator_N2_701RP_MODE,
                           Regulator_N2_701RP_INVAL) and
  regulation_fonctionnelle(Regulator_N2_702RP_AVALUE,
                           Regulator_N2_702RP_MODE,
                           Regulator_N2_702RP_INVAL) and
  regulation_fonctionnelle(Regulator_N2_701RC_AVALUE,
                           Regulator_N2_701RC_MODE,
                           Regulator_N2_701RC_INVAL) and
  regulation_fonctionnelle(Regulator_N2_702RC_AVALUE,
                           Regulator_N2_702RC_MODE,
                           Regulator_N2_702RC_INVAL) and
  -- boucle 1
  ( not boucle2_isolee =>
    (
      regulation_fonctionnelle(Regulator_N2_111RP_AVALUE,
                               Regulator_N2_111RP_MODE,
                               Regulator_N2_111RP_INVAL) and
      regulation_fonctionnelle(Regulator_N2_111RC_AVALUE,
                               Regulator_N2_111RC_MODE,
                               Regulator_N2_111RC_INVAL)
    )
  ) and
  -- boucle 2
  ( not boucle1_isolee =>
    (
      regulation_fonctionnelle(Regulator_N2_511RP_AVALUE,
                               Regulator_N2_511RP_MODE,
                               Regulator_N2_511RP_INVAL) and
      regulation_fonctionnelle(Regulator_N2_511RC_AVALUE,
                               Regulator_N2_511RC_MODE,
                               Regulator_N2_511RC_INVAL)
    )
  );

vannes =
  ( not boucle2_isolee =>
    (
      vanne_tor(Actuator_N2_111VT_LVALUE, Actuator_N2_111VT_INVAL) and
      vanne_tor(not Actuator_N2_101VT_LVALUE, Actuator_N2_101VT_INVAL)
    )
  ) and
  ( not boucle1_isolee =>
    (
      vanne_tor(Actuator_N2_511VT_LVALUE,  Actuator_N2_511VT_INVAL) and
      vanne_tor(not Actuator_N2_501VT_LVALUE, Actuator_N2_501VT_INVAL)
    )
  );

capteurs_non_redondes =
  plage_ok(MIN, MAX_PRESSION,
           Analog_input_N2_701MP_AVALUE_v, Analog_input_N2_701MP_INVAL_v) and
  plage_ok(MIN, MAX_PRESSION,
           Analog_input_N2_702MP_AVALUE_v, Analog_input_N2_702MP_INVAL_v) and
```

```
  plage_ok(MIN, MAX_TEMPERATURE,
           Analog_input_N2_701MT_AVALUE_v, Analog_input_N2_701MT_INVAL_v) and
  plage_ok(MIN, MAX_TEMPERATURE,
           Analog_input_N2_702MT_AVALUE_v, Analog_input_N2_702MT_INVAL_v) and
  plage_ok(MIN, MAX_DEBIT,
           Analog_input_N2_700MD_AVALUE_v, Analog_input_N2_700MD_INVAL_v) and
  plage_ok(MIN, MAX_DEBIT,
           Analog_input_N2_701MD_AVALUE_v, Analog_input_N2_701MD_INVAL_v) and
  plage_ok(MIN, MAX_DEBIT,
           Analog_input_N2_702MD_AVALUE_v, Analog_input_N2_702MD_INVAL_v) and
  ( not boucle2_isolee =>
    (
      plage_ok(MIN, MAX_DEBIT,
               Analog_input_N2_111MD_AVALUE_v, Analog_input_N2_111MD_INVAL_v) and
      plage_ok(MIN, MAX_DEBIT,
               Analog_input_N2_131MD_AVALUE_v, Analog_input_N2_131MD_INVAL_v)
    )
  ) and
  ( not boucle1_isolee =>
    (
      plage_ok(MIN, MAX_DEBIT,
               Analog_input_N2_511MD_AVALUE_v, Analog_input_N2_511MD_INVAL_v) and
      plage_ok(MIN, MAX_DEBIT,
               Analog_input_N2_531MD_AVALUE_v, Analog_input_N2_531MD_INVAL_v)
    )
  );
```

Pour cela, nous utilisons des nœud précédemment définis ainsi que le nœud `vanne_tor`.

```
node vanne_tor(etat_vanne, inval: bool) returns (s: bool);
let
  s = (not inval) and etat_vanne;
tel
```

3.3.2 La moitié, au moins, des équipements redondés fonctionnent

```
equipements_redondes = pompes and capteurs_quadri;
```

Nous avons groupé les vérifications par type d'objets : pompes et capteurs redondés.

```
pompes =
  aumoins_1_pompe(not Actuator_N2_101PO_LVALUE, Actuator_N2_101PO_INVAL,
                  not Actuator_N2_102PO_LVALUE, Actuator_N2_102PO_INVAL)
and
  aumoins_1_pompe(not Actuator_N2_501PO_LVALUE, Actuator_N2_501PO_INVAL,
                  not Actuator_N2_502PO_LVALUE, Actuator_N2_502PO_INVAL)
and
  aumoins_1_pompe(Actuator_N2_701PO_LVALUE, Actuator_N2_701PO_INVAL,
                  Actuator_N2_702PO_LVALUE, Actuator_N2_702PO_INVAL)
and
  aumoins_1_pompe(Actuator_N2_801PO_LVALUE, Actuator_N2_801PO_INVAL,
                  Actuator_N2_802PO_LVALUE, Actuator_N2_802PO_INVAL);
```

```
capteurs_quadri =
  aumoins_2_capteurs_quadri(MIN, MAX_TEMPERATURE,
           Analog_input_N2_101MT_AVALUE_v, Analog_input_N2_102MT_AVALUE_v,
           Analog_input_N2_103MT_AVALUE_v, Analog_input_N2_104MT_AVALUE_v,
           Analog_input_N2_101MT_INVAL_v,  Analog_input_N2_102MT_INVAL_v,
           Analog_input_N2_103MT_INVAL_v,  Analog_input_N2_104MT_INVAL_v)
and
  aumoins_2_capteurs_quadri(MIN, MAX_TEMPERATURE,
           Analog_input_N2_501MT_AVALUE_v, Analog_input_N2_502MT_AVALUE_v,
           Analog_input_N2_503MT_AVALUE_v, Analog_input_N2_504MT_AVALUE_v,
           Analog_input_N2_501MT_INVAL_v,  Analog_input_N2_502MT_INVAL_v,
           Analog_input_N2_503MT_INVAL_v,  Analog_input_N2_504MT_INVAL_v)
and
  aumoins_2_capteurs_quadri(MIN, MAX_NIVEAU,
           Analog_input_N2_101MN_AVALUE_v, Analog_input_N2_102MN_AVALUE_v,
           Analog_input_N2_103MN_AVALUE_v, Analog_input_N2_104MN_AVALUE_v,
           Analog_input_N2_101MN_INVAL_v,  Analog_input_N2_102MN_INVAL_v,
           Analog_input_N2_103MN_INVAL_v,  Analog_input_N2_104MN_INVAL_v)
and
  aumoins_2_capteurs_quadri(MIN, MAX_NIVEAU,
           Analog_input_N2_501MN_AVALUE_v, Analog_input_N2_502MN_AVALUE_v,
           Analog_input_N2_503MN_AVALUE_v, Analog_input_N2_504MN_AVALUE_v,
           Analog_input_N2_501MN_INVAL_v,  Analog_input_N2_502MN_INVAL_v,
           Analog_input_N2_503MN_INVAL_v,  Analog_input_N2_504MN_INVAL_v);
```

Pour cet oracle, nous utilisons les nœuds `aumoins_1_pompe` et `aumoins_2_capteurs_quadri`. On définit ce dernier en utilisant le nœud `au_moins_2_parmi_4` (cf. section 2.1 du SPF5).

```
node aumoins_1_pompe(etat1, inval1, etat2, inval2:bool)
  returns (s:bool);
let
  s = (etat1 and (not inval1)) or (etat2 and (not inval2));
tel

node aumoins_2_capteurs_quadri(min, max,
                               value1, value2, value3, value4:real;
                               inval1, inval2, inval3, inval4:bool)
  returns (s:bool);
let
  s = au_moins_2_parmi_4([plage_ok(min, max, value1, inval1),
                          plage_ok(min, max, value2, inval2),
                          plage_ok(min, max, value3, inval3),
                          plage_ok(min, max, value4, inval4)]);
tel
```

La propriété de fonctionnement en mode dégradé s'écrit alors comme suit :

```
etat_degrade = EXE040SIT_LVALUE;

Prop_Fonctionnement_Degrade =
  etat_degrade => (equipements_non_redondes and equipements_redondes);
```

Cette propriété vérifie qu'il y a assez d'équipements fonctionnels pour assurer le fonctionnement global du cas d'étude bien que le circuit hydraulique soit en mode dégradé. Cette propriété est conditionnée par le fait d'être en situation dégradée.

3.4 Garantir le fonctionnement en mode nominal

En plus de la garantie de sauvegarde et de fonctionnement, aucun équipement n'est en défaut. Cette propriété est plus stricte que celle du fonctionnement en mode dégradé vu qu'elle ne tolère aucune défaillance parmi les éléments redondés.

```
etat_nominal = EXE050SIT_LVALUE;

Prop_Fonctionnement_Nominal =
  etat_nominal => (equipements_non_redondes and
                   pompes_toutes_ok and
                   capteurs_quadri_tous);
```

On factorise la vérification des objets par type.

```
pompes_toutes_ok =
  pompes_ok(not Actuator_N2_101PO_LVALUE, Actuator_N2_101PO_INVAL,
            not Actuator_N2_102PO_LVALUE, Actuator_N2_102PO_INVAL)
and
  pompes_ok(not Actuator_N2_501PO_LVALUE, Actuator_N2_501PO_INVAL,
            not Actuator_N2_502PO_LVALUE, Actuator_N2_502PO_INVAL)
and
  pompes_ok(Actuator_N2_701PO_LVALUE, Actuator_N2_701PO_INVAL,
            Actuator_N2_702PO_LVALUE, Actuator_N2_702PO_INVAL)
and
  pompes_ok(Actuator_N2_801PO_LVALUE, Actuator_N2_801PO_INVAL,
            Actuator_N2_802PO_LVALUE, Actuator_N2_802PO_INVAL);

capteurs_quadri_tous =
  capteurs_quadri_ok(MIN, MAX_TEMPERATURE,
           Analog_input_N2_101MT_AVALUE_v, Analog_input_N2_102MT_AVALUE_v,
           Analog_input_N2_103MT_AVALUE_v, Analog_input_N2_104MT_AVALUE_v,
           Analog_input_N2_101MT_INVAL_v,  Analog_input_N2_102MT_INVAL_v,
           Analog_input_N2_103MT_INVAL_v,  Analog_input_N2_104MT_INVAL_v)
and
  capteurs_quadri_ok(MIN, MAX_TEMPERATURE,
           Analog_input_N2_501MT_AVALUE_v, Analog_input_N2_502MT_AVALUE_v,
           Analog_input_N2_503MT_AVALUE_v, Analog_input_N2_504MT_AVALUE_v,
           Analog_input_N2_501MT_INVAL_v,  Analog_input_N2_502MT_INVAL_v,
           Analog_input_N2_503MT_INVAL_v,  Analog_input_N2_504MT_INVAL_v)
and
  capteurs_quadri_ok(MIN, MAX_NIVEAU,
           Analog_input_N2_101MN_AVALUE_v, Analog_input_N2_102MN_AVALUE_v,
           Analog_input_N2_103MN_AVALUE_v, Analog_input_N2_104MN_AVALUE_v,
           Analog_input_N2_101MN_INVAL_v,  Analog_input_N2_102MN_INVAL_v,
           Analog_input_N2_103MN_INVAL_v,  Analog_input_N2_104MN_INVAL_v)
and
  capteurs_quadri_ok(MIN, MAX_NIVEAU,
           Analog_input_N2_501MN_AVALUE_v, Analog_input_N2_502MN_AVALUE_v,
           Analog_input_N2_503MN_AVALUE_v, Analog_input_N2_504MN_AVALUE_v,
           Analog_input_N2_501MN_INVAL_v,  Analog_input_N2_502MN_INVAL_v,
           Analog_input_N2_503MN_INVAL_v,  Analog_input_N2_504MN_INVAL_v);
```

Les nœuds `pompes_ok` et `capteurs_quadri_ok` utilisent le nœud `exactement_4_parmi_4` (cf. section 2.1 du SPF5).

```
node capteurs_quadri_ok(min, max, value1, value2, value3, value4: real;
                                  inval1, inval2, inval3, inval4: bool)
  returns (s:bool);
let
  s = exactement_4_parmi_4([plage_ok(min, max, value1, inval1),
                            plage_ok(min, max, value2, inval2),
                            plage_ok(min, max, value3, inval3),
                            plage_ok(min, max, value4, inval4)]);
tel

node pompes_ok(etat1, inval1, etat2, inval2:bool) returns (s:bool);
let
  s = exactement_4_parmi_4([etat1, not inval1, etat2, not inval2]);
tel
```

La propriété de fonctionnement nominal doit être vérifiée uniquement lorsque le circuit hydraulique est en mode nominal. Elle nous assure alors le fonctionnement correct de tous les objets du cas d'étude.

3.5 Propriété de stabilité du système

Une caractéristique du cas d'étude indiquée dans le [SPF1] est que la régulation du système est fonctionnelle si au bout de 5 minutes après un changement de consigne (commande opérateur), le niveau et/ou la température atteint la nouvelle consigne de manière stable.

Cet oracle a été discuté dans la section 3.1.3 du SPF5 et il se traduit comme suit :

```
pas_changement = vrai_depuis_n_secondes(minutes(5),
                                        aucun_chgt_consigne and etat_nominal);

Prop_Stabilisation = ( pas_changement => tout_est_stable );
```

Le nœud `vrai_depuis_n_secondes` ayant été présenté dans la section 2.2 du SPF5, nous définissons les variables `aucun_chgt_consigne` et `tout_est_stable` en utilisant le nœud `is_stable` défini dans la section 2.3 du SPF5.

```
aucun_chgt_consigne = false -> (
  Regulator_N2_900RP_ORDER_VAL   = pre Regulator_N2_900RP_ORDER_VAL   and
  Regulator_N2_700RC_ORDER_VAL   = pre Regulator_N2_700RC_ORDER_VAL   and
  Regulator_N2_701RP_CMD_AUTO_PO = pre Regulator_N2_701RP_CMD_AUTO_PO and
  Regulator_N2_702RP_CMD_AUTO_PO = pre Regulator_N2_702RP_CMD_AUTO_PO );

tout_est_stable = is_stable(Analog_Internal_N2_100MT_AVALUE_v) and
                  is_stable(Analog_Internal_N2_500MT_AVALUE_v) and
                  is_stable(Analog_Internal_N2_100MN_AVALUE_v) and
                  is_stable(Analog_Internal_N2_500MN_AVALUE_v);
```

Cette propriété vérifie que le système se stabilise au maximum au bout de cinq minutes si aucune commande opérateur n'est passée entre temps. Ce temps est assuré uniquement lorsque l'étude de cas est dans un état nominal de fonctionnement.

3.6 Propriété liée aux actions classées

Il est intéressant de tester le fonctionnement de l'étude de cas lors du déclenchement d'actions classées. Par manque de temps, dans le cadre du projet COMON, il n'a pas été mis en place de scénario de reprise après déclenchement d'actions classées. C'est pourquoi cet oracle se contente de vérifier qu'un passage en situation d'urgence est uniquement possible après déclenchement d'un certain nombre de pannes et que ça implique forcément le déclenchement d'actions classées.

```
etat_urgence = EXE010SIT_LVALUE;

action_classee =     On_off_input_N2_101CPO_VALUE or
                     On_off_input_N2_102CPO_VALUE or
                     On_off_input_N2_501CPO_VALUE or
                     On_off_input_N2_502CPO_VALUE or
                 not On_off_input_N2_701CPO_VALUE or
                 not On_off_input_N2_702CPO_VALUE or
                     On_off_input_N2_101CVT_VALUE or
                 not On_off_input_N2_111CVT_VALUE or
                     On_off_input_N2_501CVT_VALUE or
                 not On_off_input_N2_511CVT_VALUE or
                 not On_off_input_N2_111CVR_VALUE or
                 not On_off_input_N2_511CVR_VALUE or
                 not On_off_input_N2_700CVR_VALUE or
                     On_off_input_N2_701CVR_VALUE or
                 not On_off_input_N2_101MJ_VALUE  or
                 not On_off_input_N2_501MJ_VALUE;

urgence = (EXE701PO_Panne1_OnOff and EXE702PO_Panne1_OnOff)
          or
          (au_moins_3_parmi_4([EXE101MN_Panne2_OnOff, EXE102MN_Panne2_OnOff,
                               EXE103MN_Panne2_OnOff, EXE104MN_Panne2_OnOff]));

Prop_Mode_Classe = etat_urgence => ( action_classee and urgence );
```

Cette propriété sera testée uniquement lorsque le cas d'étude sera dans une situation d'urgence.

3.7 Propriétés non codées

Il y a certaines propriétés qui ont été décrites dans le [SPF1] mais qui n'ont pu être formalisées. Cela est du au flou de la description en langage naturel de ces propriétés. En effet, il est difficile d'évaluer des termes tels que « relativement proportionnel », « tolérance temporelle » et « de façon cohérente » qui ne donnent pas d'indications précises sur les données physiques sous-jacentes à vérifier. C'est aussi du à la nature même de cette étude de cas qui est un sous ensemble fictif dont le dimentionnement doit être affiné.

```
-------------------------------------------------------------------------------
-- TODO Garantie de fonctionnement :
-------------------------------------
-- d) Le debit passant par l'echangeur doit etre relativement
--    proportionnel a la puissance de chauffe avec une tolerance temporelle
-- e) Le debit evacue par chaque bache doit etre relativement
--    proportionnel a la puissance de chauffe avec une tolerance temporelle
-------------------------------------------------------------------------------
-- TODO  Proprietes hierarchiques autres :
```

```
-----------------------------------------
-- a) Les regulations de niveau en fonctionnement doivent reagir de facon
--    coherente avec une variation de niveau
-- b) Les regulations de temperatures en fonctionnement doivent reagir de
--    facon coherente avec une variation de temperatures
-------------------------------------------------------------------------
```

3.8 Couverture des oracles

Les propriétés de sauvegarde, de fonctionnement et de stabilisation doivent toujours être vérifiées sans aucun prérequis. Les propriétés de fonctionnement nominal, de fonctionnement dégradé et d'action classées seront uniquement vérifiées lorsque le cas d'étude sera dans un mode de fonctionnement particulier : mode nominal, mode dégradé, mode urgence.

En effet, le circuit hydraulique peut se trouver dans différentes situations dont certaines ont été identifiées comme étant de bonnes conditions pour évaluer la couverture de nos scénarios.

- état nominal,
- état dégradé,
- état 2/3 - 1/3 sur boucle 1,
- état 2/3 - 1/3 sur boucle 2,
- boucle 1 isolée,
- boucle 2 isolée,
- état d'urgence,
- état nominal et aucune action opérateur depuis 5 minutes,
- régulation de température sur 701VR,
- régulation de température sur 702VR,
- pannes mais sans état d'urgence.

Nos scénarios vont essayer de couvrir toutes ces conditions. Toutefois, il est évident qu'un scénario simple ne peut pas couvrir toutes ces conditions. Par exemple, le mode urgence n'est censé se déclencher qu'à la suite d'un nombre important de pannes. Donc il est doit être impossible de couvrir cette condition avec des scénarios n'impliquant pas de déclenchement de pannes.

4 Utilisation depuis Adacs et Alices en même temps

Dans le cadre de l'avancement du projet COMON, nous avons été amené à remplacer le N2 informatisé Alices par Adacs. Nous avons réutilisé nos trois premiers scénarios tels quels en se servant du dictionnaire de noms pour faire le lien entre les noms de variables à la Alices et les noms des mêmes objets dans Adacs. Il n'y a que le scénario de déclenchement de pannes que l'on n'a pas transposé coté Adacs, ce qui est normal vu que ce dernier agit au niveau N1.

Les oracles aussi ont été réutilisés sans changements. Nous les avons même enrichis en y ajoutant une propriété concernant la gestion des relais à point de consigne (RPC) et une autre concernant les situations.

La version du N2 simulé Alices étant bien moins complète qu'Adacs, nous pourrions affiner nos oracles en tenant compte des informations liées au matériel et qui n'étaient que partiellement câblées dans Alices (à savoir : mise en test, invalidité, discordance … etc.). Nous pourions aussi enrichir nos scénarios en tenant compte entre autres des retours de prise en compte des commandes, et des déclenchements et fin des timers avant d'effectuer de nouveaux changements.

Dans ce mode de fonctionnement, Lurette se connecte à la fois à Alices pour déclencher des pannes, et à Adacs pour les scénarios de fonctionnement. Les oracles vérifient alors les données à la sortie des deux ateliers.

5 Conclusion

Nous avons démontré l'utilisation de Lutin pour représenter des scénarios de stimulations du cas d'étude. Toutes les possibilités de perturbation du cas d'étude n'ont pas été utilisées. Notamment le changement manuel des consignes de chauffe de chaque boucle primaire, ou le passage de certaines régulations en mode manuel.

Nous avons démontré l'utilisation de Lustre pour écrire des oracles représentant une formalisation des propriétés fonctionnelles de l'étude de cas. Cependant, certaines propriétés n'ont pu être traduites en oracles au vu du flou concernant leur interprétation. Notamment des propriétés vérifiant que deux grandeurs évoluent proportionnellement l'une par rapport à l'autre avec un décalage temporel. Le fait de ne pas pouvoir quantifier le facteur de proportionnalité ou le décalage temporel dans la spécification nous a empêché de les formaliser.

Nous avons définis des critères de couverture par rapport aux propriétés que nous avons vérifié. Mais il faudrait approfondir cette question des critères pour être sûrs d'avoir couverts tous des cas de tests.