Étude de cas Scada

Client :	<->
Affaire/Projet :	COMON
Référence :	COMON/SPF 5 Annexe C
Révision :	0.1
État :	PRE
Date :	4 mai 2012
Classification :	Public
Nombre d'annexes :	0

Table des matières

1 Introduction

Un Scada, acronyme de l'anglais Supervisory Control And Data Acquisition (télésurveillance et acquisition de données), est un système de télégestion à grande échelle permettant de traiter en temps réel un grand nombre de télémesures et de contrôler à distance des installations techniques. C'est une technologie industrielle dans le domaine de l'instrumentation, dont les implémentations peuvent être considérées comme des frameworks d'instrumentation incluant une couche de type Middleware [1].

1. source : http ://fr.wikipedia.org/wiki/Supervisory_Control_and_Data_Acquisition

Ce document présente les tests effectués sur une étude de cas sur un Scada pour démontrer une utilisation possible des outils Verimag et de la démarche proposée dans COMON pour la validation d'objets génériques. Plus précisément, nous avons étudié l'objet «domaines algébriques» qui définit des domaines de fonctionnement à tester lors des campagnes de tests.

Après avoir décrit brièvement comment un tel objet était testé actuellement, nous montrons quelques exemples d'utilisation des méthodes et outils proposés par Verimag. Il est conseillé d'avoir lu le SPF5 (données formalisées) avant de lire ce document.

2 Description de l'objet « domaines algébriques »

Il s'agit d'une application qui nécessite de positionner un point dans un espace à 2 dimensions. Cet espace est divisé en 4 domaines de fonctionnements (qui peuvent se recouvrir). Chaque domaine est défini par deux intervalles de valeurs (un pour X, et un pour Y). L'opérateur peut sélectionner un seul domaine à la fois : le point testé sera alors choisi dans ce domaine. Les domaines peuvent aussi être forcés ; dans ce cas, le domaine forcé sera prioritaire par rapport au domaine sélectionné, et ce jusqu'à ce qu'il soit déforcé, ou qu'un autre domaine soit forcé.

Par ailleurs, l'espace des valeurs de X et Y est partitionné en 5 zones disjointes. Un domaine peut contenir plusieurs zones et les zones peuvent apparaissent dans plusieurs domaines. Ces zones se divisent en trois catégories :

1. Les zones autorisées (zone 1 et zone 4 dans l'exemple) : quand le point testé est dans une telle zone, aucune alarme ne doit être levée.
2. Les zones interdites (zone 2) : quand le point testé entre dans une telle zone, une alarme doit être levée.
3. Les zones d'accumulation (zone 3 et zone 5) : si le point reste un certain temps dans une telle zone, une alarme doit être levée. Ce compteur de temps s'incrémente dès que le point testé est dans la zone. L'opérateur peut aussi remettre ce compteur à zéro à tout instant.

3 Méthodes et outils de tests actuels

Les campagnes de test menées sont réalisées à l'aide de cahiers de test. Ces cahiers de test sont constitués d'un tableau dont les lignes représentent les différentes étapes du test, et dont les colonnes indiquent les actions à effectuer par l'opérateur. La colonne de gauche indique à l'opérateur un ensemble d'actions à réaliser, c'est-à-dire, un ensemble d'entrées du système sous test à positionner. Celle de droite indique ce que cet opérateur est censé observer à chaque étape.

Les ingénieurs qui ont développé cet objet se sont dotés d'un langage de script permettant de faciliter le travail de l'opérateur qui joue les cahiers de test, en lui permettant de positionner d'un coup un ensemble de variables. Ces scripts sont exécutés par un outil nommé « le stimulateur d'entrées/sorties du Scada ».

Nous présentons dans cette section un cahier de test permettant de tester certains aspects d'un objet Scada nommé « domaines algébriques », ainsi que le code d'un stimulateur permettant de faciliter le jeu de ce cahier.

3.1 Le stimulateur d'entrées/sorties

Nous commençons par présenter le script du stimulateur d'entrées/sorties, car il y est fait explicitement référence dans le cahier de test. Ce script permet à l'opérateur de jouer une séquence de 4 pas. À chaque pas, un certain nombre de variables sont positionnées à certaines valeurs. Les variables non mentionnées ne changent pas. Par exemple, lors du premier pas, 4 variables booléennes et 2 analogiques sont modifiées.

```
DEBUT_SCENARIO crt_019_c04_s04_presenceA;
CM ====================================================;
CC Scenario crt_019_c04_s04_presenceA  ;
CM ====================================================;
CM Lancement du pas 1 Initialisation;
CC 1 Tor d'election 1DMA001 a VRAI, les autres a FAUX;
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA001OIW  1 ( VALUE TOR VRAI 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA002OIW  1 ( VALUE TOR FAUX 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA003OIW  1 ( VALUE TOR FAUX 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA004OIW  1 ( VALUE TOR FAUX 0)
);
CM positionnement de x et y du 1DMA001 zone A autorisee;
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DOM003AIS  1 ( VALUE ANA 25.0 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DOM004AIS  1 ( VALUE ANA 40.0 0)
);
CM Fin pas 1;
CM ------------------------------------------------;
CC Cliquer sur ENTREE pour lancer le pas 2;
ATTENDRE;
CC Passage de X et Y dans la zone B alarme sur presence;
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DOM003AIS  1 ( VALUE ANA 40.0 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DOM004AIS  1 ( VALUE ANA 28.0 0)
);
CM Fin pas 2;
CM ------------------------------------------------;
CC Cliquer sur ENTREE pour lancer le pas 3;
ATTENDRE;
CC election du domaine 1DMA002 ;

GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA002OIW  1 ( VALUE TOR VRAI 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE  1 (
        1DMA001OIW  1 ( VALUE TOR FAUX 0)
);
```

```
CM Fin pas 3;
CM ----------------------------------------------;
CC Cliquer sur ENTREE pour lancer le pas 4;
ATTENDRE;
CC Positionner X et Y en zone autorisee de 1DMA003 ;

GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE 1 (
        1DOM003AIS 1 ( VALUE ANA -9.0 0)
);
GEN_MSG NEANT CCT INFORMATION_N1_NON_DATE INFO_NON_DATE 1 (
        1DOM004AIS 1 ( VALUE ANA 25.0 0)
);
CM Fin pas 4;
ATTENDRE;
FIN_SCENARIO;
```

3.2 Le cahier de test CRT_019_S04

Ce cahier de test est découpé en 7 étapes, qui font référence au 4 pas du stimulateur présenté juste avant. L'opérateur positionne à chaque étape un certain nombre de variables, soit par lui-même, soit via le stimulateur. Il vérifie à chaque étape « de visu » que le comportement du système sous test est conforme.

N°	Action	Résultat attendu / Commentaire
1	Lancer le pas 1 du scénario Activer Mimic depuis le finder. Afficher IMG_DOMA11	Vérifier l'affichage de l'image.
2	Lancer le pas 2 du scénario Positionner X,Y dans la zone interdite sur présence	Vérifier le point de fonctionnement (position, couleur, pictogramme) Vérifier que l'alarme est levée dans la fonction alarme Noter l'horodate
3	Lancer le pas 3 du scénario Elire DMAoo2 sans que le Pt change de Zone	Vérifier que l'alarme ci-dessus demeure en apparition à l'horodate du pas 2.
4	Forcer en DMAoo3 sans que le Pt change de Zone	Vérifier que l'alarme ci-dessus demeure en apparition à l'horodate du pas 2.
5	Forcer en DMAoo4 Le Point passe en Zone autorisée	Vérifier que l'alarme ci-dessus disparaît (le Pt n'appartient plus à une ZI).
6	DéForcer DMAoo4 Le domaine élémentaire élu redevient DMAoo2 sans que le Pt change de Zone	Vérifier que l'alarme à l'horodate courante
7	Lancer le pas 4 du scénario Positionner X,Y dans la zone autorisée	Vérifier le point de fonctionnement (position, couleur, pictogramme) L'alarme passe en disparition (il faut l'acquitter pour qu'elle disparaisse)

4 Utilisation de Lurette via une approche calquée sur le cahier de test

Nous illustrons dans cette section comment utiliser Lurette pour automatiser le déroulement du cahier de test `CRT_019_S04`.

4.1 Description des entrées/sorties du système sous test

Nous présentons d'abord la liste des entrées et sorties du système à tester qui sont utiles pour jouer le cahier de test en question. Les noms des variables suivent une convention de nommage basée sur le nom de l'objet Scada et sur le nom de la variable de l'objet : `<nom de l'objet>_<nom de la Variable dans Scada>`.

4.1.1 Entrées

Les entrées de l'objet « domaines algébriques » correspondent aux différentes actions accessibles depuis l'interface graphique pour un opérateur de tests.

– Choix des coordonnées du point à tester

```
A1DOM003AIS_VALUE: real;
A1DOM004AIS_VALUE: real;
```

– Sélection d'un domaine pour qu'il soit activé

```
A1DMA001OIW_VALUE, A1DMA002OIW_VALUE, A1DMA003OIW_VALUE, A1DMA004OIW_VALUE: bool;
```

– Forçage ou déforçage d'un domaine

```
A1DOM001DMA_V_FORCED1: bool;
A1DOM001DMA_V_FORCED2: bool;
A1DOM001DMA_V_FORCED3: bool;
A1DOM001DMA_V_FORCED4: bool;

A1DOM001DMA_V_DEFORCED1: bool;
A1DOM001DMA_V_DEFORCED2: bool;
A1DOM001DMA_V_DEFORCED3: bool;
A1DOM001DMA_V_DEFORCED4: bool;
```

– Remise à zéro des compteurs d'alarmes

```
A1DOM001DMA_V_RAZ_TIMER : bool;
```

4.1.2 Sorties

Les sorties de l'objet « domaines algébriques » sont équivalentes à des variables internes devant être exposées à l'opérateur depuis l'interface graphique.

– Les coordonnées du point affiché à l'écran

```
A1DOM001DMA_X_AVALUE: real;
A1DOM001DMA_Y_AVALUE: real;
```

– Le numéro du domaine élu (sélectionné par l'opérateur)

```
A1DOM001DMA_NUM_ED_ELEC: int;
```

– Le numéro du domaine forcé (0 si aucun)

```
A1DOM001DMA_NUM_ED_FORCED: int;
```

– Le domaine affiché par le Scada

```
A1DMA0010IW_LVALUE: bool;
A1DMA0020IW_LVALUE: bool;
A1DMA0030IW_LVALUE: bool;
A1DMA0040IW_LVALUE: bool;
```

– Les intervalles de définition de chacun des quatre domaines

```
A1DMA001_MIN_X, A1DMA001_MAX_X, A1DMA001_MIN_Y, A1DMA001_MAX_Y: real;
A1DMA002_MIN_X, A1DMA002_MAX_X, A1DMA002_MIN_Y, A1DMA002_MAX_Y: real;
A1DMA003_MIN_X, A1DMA003_MAX_X, A1DMA003_MIN_Y, A1DMA003_MAX_Y: real;
A1DMA004_MIN_X, A1DMA004_MAX_X, A1DMA004_MIN_Y, A1DMA004_MAX_Y: real;
```

– Le numéro de la zone courante

```
A1DOM001DMA_NUM_CUR_ZONE: int;
```

– La présence d'une alarme pour chacune des cinq zones

```
A1DOM001DMA_ALARME1: int;
A1DOM001DMA_ALARME2: int;
A1DOM001DMA_ALARME3: int;
A1DOM001DMA_ALARME4: int;
A1DOM001DMA_ALARME5: int;
```

– Les valeurs des timers calculés en interne

```
A1DOM001DMA_VAL_TIMER3: int;
A1DOM001DMA_VAL_TIMER5: int;
```

4.2 La colonne « action » du `CRT_019_S04` en Lutin

Nous présentons maintenant le nœud Lutin `crt019_s04`, qui exécute automatiquement les 7 pas du cahier de test, comme le ferait un opérateur. Nous définissons tout d'abord quelques macros booléennes triviales permettant de faciliter l'écriture et la lecture du programme.

```
let vfff(x,y,z,t:bool):bool =     x and not y and not z and not t
let fvff(x,y,z,t:bool):bool = not x and     y and not z and not t
let ffff(x,y,z,t:bool):bool = not x and not y and not z and not t
let vvvv(x,y,z,t:bool):bool =     x and     y and     z and     t
let tous_faux_7(x1,x2,x3,x4,x5,x6,x7:bool):bool =
      not x1 and not x2 and not x3 and not x4 and not x5 and not x6 and not x7
let tous_faux_8(x1,x2,x3,x4,x5,x6,x7,x8:bool):bool =
```

```
      not x1 and not x2 and not x3 and not x4 and not x5 and not x6 and not x7
      and not x8
```

L'entrée entière « pas » sert à contrôler depuis l'extérieur du nœud l'instant où l'on passe d'un pas à l'autre. Ce passage peut être réalisé par un opérateur, ou par un autre nœud Lutin. « pas » est censé prendre séquentiellement toutes les valeurs entre 1 et 7.

```
node crt019_s04(pas:int)
returns (
  A1DMA001OIW_VALUE, A1DMA002OIW_VALUE, A1DMA003OIW_VALUE, A1DMA004OIW_VALUE: bool;
  A1DOM003AIS_VALUE, A1DOM004AIS_VALUE: real;
  A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_FORCED2,
  A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_FORCED4: bool;
  A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
  A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4: bool;
) =
{
  loop {
    pas <= 1 and
    A1DOM003AIS_VALUE = 25.0 and
    A1DOM004AIS_VALUE = 40.0 and
    vfff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
    and
    ffff(A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_FORCED2,
         A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_FORCED4) and
    vvvv(A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
         A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
  }
```

Tant que l'entrée pas est inférieure ou égale à 1, les sorties de ce nœud sont positionnées en respectant la contrainte ci-dessus. Dès que pas vaut 2, on passe à la suite. Pour mettre en place une session complètement automatisée, on pourrait enlever l'entrée « pas » de ce nœud, et remplacer « loop » par « loop [minutes(4)] » par exemple.

```
fby loop {
  pas = 2 and
  A1DOM003AIS_VALUE = 40.0 and
  A1DOM004AIS_VALUE = 28.0 and
  vfff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  tous_faux_8(
    A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
    A1DOM001DMA_V_FORCED3,   A1DOM001DMA_V_FORCED4,
    A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
    A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
}
fby loop {
  pas = 3 and
  A1DOM003AIS_VALUE = 40.0 and
  A1DOM004AIS_VALUE = 28.0 and
  fvff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  tous_faux_8(
    A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
```

```
    A1DOM001DMA_V_FORCED3,   A1DOM001DMA_V_FORCED4,
    A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
    A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
}
fby loop {
  pas = 4 and
  A1DOM003AIS_VALUE = 40.0 and
  A1DOM004AIS_VALUE = 28.0 and
  fvff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  A1DOM001DMA_V_FORCED3 and
  tous_faux_7(
     A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
                              A1DOM001DMA_V_FORCED4,
     A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
     A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
}
fby loop {
  pas = 5 and
  A1DOM003AIS_VALUE = 40.0 and
  A1DOM004AIS_VALUE = 28.0 and
  fvff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  A1DOM001DMA_V_FORCED4 and
  tous_faux_7(
       A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
       A1DOM001DMA_V_FORCED3,
       A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
       A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
}
fby loop {
  pas = 6 and
  A1DOM003AIS_VALUE = 40.0 and
  A1DOM004AIS_VALUE = 28.0 and
  fvff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  A1DOM001DMA_V_DEFORCED4 and
  tous_faux_7(
    A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
    A1DOM001DMA_V_FORCED3,   A1DOM001DMA_V_FORCED4,
    A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
    A1DOM001DMA_V_DEFORCED3)
}
fby loop {
  pas >= 7 and
  A1DOM003AIS_VALUE = -9.0 and
  A1DOM004AIS_VALUE = 25.0 and
  fvff(A1DMA001OIW_VALUE,A1DMA002OIW_VALUE,A1DMA003OIW_VALUE,A1DMA004OIW_VALUE)
  and
  A1DOM001DMA_V_DEFORCED4 and
  tous_faux_7(
    A1DOM001DMA_V_FORCED1,   A1DOM001DMA_V_FORCED2,
    A1DOM001DMA_V_FORCED3,   A1DOM001DMA_V_FORCED4,
    A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
    A1DOM001DMA_V_DEFORCED3)
}
```

L'inconvénient de Lutin pour décrire ce type de scénario complètement déterministe est que l'on est obligé de mentionner les variables qui ne changent pas de valeur (nous réfléchissons à des évolutions du langage nous permettant de nous affranchir de cet inconvénient).

Néanmoins, un premier avantage de l'utilisation de Lutin est de pouvoir jouer, de façon automatisée, un scénario identique à celui réalisé par un opérateur qui jouerait le cahier de test `CRT_019_S04` avec le stimulateur d'entrée/sortie associé.

Mais le principal avantage de Lutin est ailleurs. En effet, ce scénario est inutilement déterministe. L'exigence du cahier de test « Positionner X,Y dans la zone interdite sur présence » a été traduite par le choix d'un point particulier dans la dite zone. Or en Lutin, il est facile de faire beaucoup mieux à moindre effort, en remplaçant la contrainte :

```
A1DOM003AIS_VALUE = 40.0 and A1DOM004AIS_VALUE = 28.0
```

par une contrainte comme celle-ci :

```
(Zone2_MIN_X < A1DOM003AIS_VALUE) and (A1DOM003AIS_VALUE < Zone2_MAX_X) and
(Zone2_MIN_Y < A1DOM004AIS_VALUE) and (A1DOM004AIS_VALUE < Zone2_MAX_Y)
```

Ainsi, à chaque fois que ce stimulateur Lutin est exécuté, une valeur différente sera choisie pour le point (X,Y) dans la zone interdite sur présence. On peut faire la même chose au pas 7, lors du choix d'un point dans la zone autorisée.

Dans le même esprit, on pourrait même modifier le cahier de test en remplacant l'exigence du `pas` 3 « Elire `DMAoo2` sans que le Pt change de Zone » par « Elire une des variables parmi { `DMAoo1`, `DMAoo2`, `DMAoo3`, `DMAoo4` } sans que le Pt change de Zone ». En effectuant une modification similaire aux `pas` 4, 5, et 6, nous obtiendrions, pour le même effort que celui réalisé actuellement, un nombre de tests bien plus grand.

4.3 La colonne « observation » du `CRT_019_S04` en Lustre

Afin de statuer automatiquement sur le bon déroulement de ce scénario de test, nous avons formalisé la colonne observation du cahier de test sous la forme d'un oracle Lustre.

```
node crt019_s04(
    pas                       : int;
    A1DOM001DMA_NUM_ED_ELEC   : int;
    A1DOM001DMA_NUM_ED_FORCED: int;
    A1DOM001DMA_ALARME1       : int;
    A1DOM001DMA_ALARME2       : int;
    A1DOM001DMA_ALARME3       : int;
    A1DOM001DMA_ALARME4       : int;
    A1DOM001DMA_ALARME5       : int;
    A1DOM001DMA_NUM_CUR_ZONE : int)
returns(
    ok : bool)
var
    ok_pas1, ok_pas2, ok_pas3, ok_pas4, ok_pas5, ok_pas6, ok_pas7: bool;
let
  ok_pas1 = (pas = 1 =>
      (A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME2=0 and
```

```
        A1DOM001DMA_ALARME3=0 and A1DOM001DMA_ALARME4=0 and
        A1DOM001DMA_ALARME5=0));
  ok_pas2 = (pas = 2 =>
      (A1DOM001DMA_NUM_CUR_ZONE=2 and A1DOM001DMA_ALARME2=1 and
        A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok_pas3 = (pas = 3 =>
      (A1DOM001DMA_NUM_CUR_ZONE=2 and A1DOM001DMA_ALARME2=1 and
        A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok_pas4 = (pas = 4 =>
      (A1DOM001DMA_NUM_CUR_ZONE=2 and A1DOM001DMA_ALARME2=1 and
        A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok_pas5 = (pas = 5 =>
      (A1DOM001DMA_NUM_CUR_ZONE=4 and A1DOM001DMA_ALARME1=0 and
        A1DOM001DMA_ALARME2=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok_pas6 = (pas = 6 =>
      (A1DOM001DMA_NUM_ED_ELEC=2 and A1DOM001DMA_NUM_ED_FORCED=0 and
        A1DOM001DMA_NUM_CUR_ZONE=2 and A1DOM001DMA_ALARME2=1 and
        A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok_pas7 = (pas = 7 =>
      (A1DOM001DMA_NUM_CUR_ZONE=4 and A1DOM001DMA_ALARME2=0 and
        A1DOM001DMA_ALARME1=0 and A1DOM001DMA_ALARME3=0 and
        A1DOM001DMA_ALARME4=0 and A1DOM001DMA_ALARME5=0));
  ok = ok_pas1 and ok_pas2 and ok_pas3 and ok_pas4 and
       ok_pas5 and ok_pas6 and ok_pas7;
tel
```

5 Utilisation de Lurette via une formalisation directe des attendus

Dans la section précédente, nous avons montré comment le processus de test actuellement pratiqué pouvait être automatisé (en terme de stimulation et de décision du test) et « aléatoirisé » pour obtenir une plus grande couverture.

Nous illustrons maintenant un exemple d'utilisation de Lurette un peu différente de la démarche basée sur l'écriture de cahier de test. L'idée est de ne pas écrire de scénarios précis dans un premier temps, mais de simplement écrire des contraintes formalisant le comportement d'un environnement réaliste, et de laisser la machinerie pseudo-aléatoire de Lutin couvrir toute la combinatoire. La décision du test est effectuée elle aussi à l'aide de propriétés plus abstraites, issues d'une formalisation des exigences fonctionnelles du système.

5.1 Environnements

Afin de faciliter la lecture, la modification et la réutilisation du code, nous avons décrit le comportement des variables à stimuler dans 4 nœuds Lutin différents :

- `gen_point_dans_domaine` qui génère les coordonnées X et Y du point de test en fonction du domaine actif ;
- `Elec` qui active un des domaines ;
- `RAZ` qui remet à zéro les compteurs d'alarmes.
- `Force` qui force et/ou déforce chacun des domaines ;

5.1.1 Génération du point dans un domaine

Grâce au nœud `gen_point_dans_domaine`, nous allons déplacer les coordonnées X et Y du point à tester dans les limites imposées par le domaine actif. Une version simplifiée de ce nœud a été présentée dans la section 3.2.3 du SPF5. Voici tout d'abord quelques macros définissant les 4 domaines.

```
let dom1(x,y:real):bool = between(x, A1DMA001_MIN_X, A1DMA001_MAX_X) and
                          between(y, A1DMA001_MIN_Y, A1DMA001_MAX_Y)
let dom2(x,y:real):bool = between(x, A1DMA002_MIN_X, A1DMA002_MAX_X) and
                          between(y, A1DMA002_MIN_Y, A1DMA002_MAX_Y)
let dom3(x,y:real):bool = between(x, A1DMA003_MIN_X, A1DMA003_MAX_X) and
                          between(y, A1DMA003_MIN_Y, A1DMA003_MAX_Y)
let dom4(x,y:real):bool = between(x, A1DMA004_MIN_X, A1DMA004_MAX_X) and
                          between(y, A1DMA004_MIN_Y, A1DMA004_MAX_Y)
```

Le choix du domaine est contrôlé par 2 entrées de l'environnement

`A1DOM001DMA_NUM_ED_ELEC`, qui provient d'une élection effectuée par le nœud `Elec` (voir plus bas), et `A1DOM001DMA_NUM_ED_FORCED` qui correspond au domaine forcé par le nœud `FORCE` (voir plus bas). Quand `A1DOM001DMA_NUM_ED_FORCED` vaut 0, cela signifie qu'aucun domaine n'est forcé, et c'est `A1DOM001DMA_NUM_ED_ELEC` qui est utilisé ; sinon on utilise le domaine forcé `A1DOM001DMA_NUM_ED_FORCED`.

```
node gen_point_dans_domaine(A1DOM001DMA_NUM_ED_ELEC, A1DOM001DMA_NUM_ED_FORCED:int)
returns(A1DOM004AIS_VALUE, A1DOM003AIS_VALUE: real) =
  exist dom_utilise: int in
  assert dom_utilise =
```

```
          if A1DOM001DMA_NUM_ED_FORCED = 0 then A1DOM001DMA_NUM_ED_ELEC
                                           else A1DOM001DMA_NUM_ED_FORCED
  in
  loop {
    if dom_utilise = 0 then dom1(A1DOM003AIS_VALUE, A1DOM004AIS_VALUE) else
    if dom_utilise = 1 then dom1(A1DOM003AIS_VALUE, A1DOM004AIS_VALUE) else
    if dom_utilise = 2 then dom2(A1DOM003AIS_VALUE, A1DOM004AIS_VALUE) else
    if dom_utilise = 3 then dom3(A1DOM003AIS_VALUE, A1DOM004AIS_VALUE) else
    if dom_utilise = 4 then dom4(A1DOM003AIS_VALUE, A1DOM004AIS_VALUE) else
    raise Error
  }
```

5.1.2 Élection d'un domaine

Le nœud `Elec` active au hasard un des quatre domaines et maintient ce choix pendant environ 5 minutes, puis recommence. Le nœud `exactement_1_parmi_4` au comportement éponyme, est défini en section 2.1 du SPF5.

```
node Elec() returns (
  A1DMA001OIW_VALUE, A1DMA002OIW_VALUE, A1DMA003OIW_VALUE, A1DMA004OIW_VALUE:bool
) =
  loop {
    exactement_1_parmi_4(A1DMA001OIW_VALUE, A1DMA002OIW_VALUE,
                         A1DMA003OIW_VALUE, A1DMA004OIW_VALUE)
    fby
    loop ~minutes(5) {
      A1DMA001OIW_VALUE = pre A1DMA001OIW_VALUE and
      A1DMA002OIW_VALUE = pre A1DMA002OIW_VALUE and
      A1DMA003OIW_VALUE = pre A1DMA003OIW_VALUE and
      A1DMA004OIW_VALUE = pre A1DMA004OIW_VALUE
    }
  }
```

5.1.3 Remise à zéro des compteurs d'alarmes

Le nœud `RAZ` s'occupe de remettre à zéro les compteurs d'alarmes dans 5% des cas et sinon il ne fait rien.

```
node RAZ() returns (A1DOM001DMA_V_RAZ_TIMER: bool) =
  loop
    {
      | 95 : not A1DOM001DMA_V_RAZ_TIMER
      | 5  :     A1DOM001DMA_V_RAZ_TIMER
    }
```

5.1.4 Forçage d'un domaine

Le nœud `Force` va jouer avec les boutons de forçage et de déforçage des domaines. Pour obtenir un comportement réaliste, on fait en sorte que les forçages/déforçages ne se produisent pas trop souvent. Par exemple, une fois sur 106 (car $\frac{20}{1000+100+20} = \frac{1}{106}$), un des quatre domaines sera forcé.

```
node Force()
returns (
    A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_DEFORCED1:bool;
    A1DOM001DMA_V_FORCED2, A1DOM001DMA_V_DEFORCED2:bool;
    A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_DEFORCED3:bool;
    A1DOM001DMA_V_FORCED4, A1DOM001DMA_V_DEFORCED4:bool;
) =
  loop {
    | 2000 : ffff(A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_FORCED2
                  A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_FORCED4)
         and vvvv(A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
                  A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
    | 100  : ffff(A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_FORCED2,
                  A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_FORCED4)
         and ffff(A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
                  A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
    | 20 : exactement_1_parmi_4(
                  A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_FORCED2,
                  A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_FORCED4)
         and ffff(A1DOM001DMA_V_DEFORCED1, A1DOM001DMA_V_DEFORCED2,
                  A1DOM001DMA_V_DEFORCED3, A1DOM001DMA_V_DEFORCED4)
  }
```

5.2 Oracles

La décision du bon déroulement d'un test est effectuée elle aussi à l'aide de propriétés issues d'une formalisation des exigences fonctionnelles du système. Nous illustrons comment on peut définir des critères de couverture qui permettent de décider si assez de tests ont été effectués.

5.2.1 Relation entre `VALUE` et `LVALUE`

Cet premier oracle ne vérifie pas que l'implémentation est conforme à sa spécification, mais il permet de s'assurer que la connexion entre le Scada et Lurette fonctionne correctement. Plus précisément, nous avons défini un oracle qui vérifie que certaines valeurs générées par des stimulateurs Lutin et leur équivalent en interne au Scada sont cohérentes.

La connexion entre la vision en temps discret du monde de Lurette, et celle en temps continu du Scada se fait en échantillonnant les valeurs des variables du Scada (comme décrit dans le CPR4). Si la fréquence d'échantillonage est trop élevée, il est normal d'observer que ces 2 variables n'ont pas les mêmes valeurs aux mêmes instants. En effet, il est normal que le Scada prenne un peu de temps pour propager les conséquences d'un changement de valeur. Nous vérifions donc l'égalité entre les 2 incarnations d'une variable avec le nœud `equal_b` (cf Section section 2.5 du SPF5), qui autorise une tolérance temporelle de `delta_t`. Les temps de cycle, et les niveaux de tolérance temporelle et numérique sont définies une fois pour toute.

```
const temps_de_cycle = 1.0;
const temps_de_cycle_ms = 1000;
const delta_t = 30;  -- tolerance temporelle en ms
const delta_v = 0.1; -- tolerance numerique
const d = 1 + (delta_t/temps_de_cycle_ms); -- vaut 1 si delta_t < temps_de_cycle
```

On peut remarquer que si la tolérance `delta_t` requise est inférieure au `temps_de_cycle`, la constante `d` vaut 1, et le nœud `equal_b` se comporte comme une égalité classique. Ensuite, l'oracle est tout simple.

```
node VALUE_et_LVALUE(
  A1DMA001OIW_LVALUE, A1DMA001OIW_VALUE : bool;
  A1DMA002OIW_LVALUE, A1DMA002OIW_VALUE : bool;
  A1DMA003OIW_LVALUE, A1DMA003OIW_VALUE : bool;
  A1DMA004OIW_LVALUE, A1DMA004OIW_VALUE : bool)
returns (
   ok: bool;
   c01, c02, c03, c04, c05, c06, c07, c08: bool);
var
  ok1, ok2, ok3, ok4: bool;
let
  ok1 = equal_b<<d>>(A1DMA001OIW_VALUE,A1DMA001OIW_LVALUE);
  ok2 = equal_b<<d>>(A1DMA002OIW_VALUE,A1DMA002OIW_LVALUE);
  ok3 = equal_b<<d>>(A1DMA003OIW_VALUE,A1DMA003OIW_LVALUE);
  ok4 = equal_b<<d>>(A1DMA004OIW_VALUE,A1DMA004OIW_LVALUE);
  ok = ok1 and ok2 and ok3 and ok4;

  c01 =     A1DMA001OIW_LVALUE;
  c02 = not A1DMA001OIW_LVALUE;
  c03 =     A1DMA002OIW_LVALUE;
  c04 = not A1DMA002OIW_LVALUE;
  c05 =     A1DMA003OIW_LVALUE;
  c06 = not A1DMA003OIW_LVALUE;
  c07 =     A1DMA004OIW_LVALUE;
  c08 = not A1DMA004OIW_LVALUE;
tel
```

Pour couvrir tous les cas possibles, il parait raisonnable de se contenter d'avoir vu passer tous les `DOMNUM_LVALUE` à faux et à vrai au moins une fois.

5.2.2 Position du point affiché

L'oracle de cette section relève de la même démarche que le précédent, en ce sens qu'il ne cherche pas à s'assurer de la conformité du système sous test à sa spécification. Le nœud `X_et_Y` vérifie que les coordonnées générées par le stimulateur Lutin correspondent bien aux coordonnées du point affiché dans le Scada. Pour vérifier cette égalité, on utilise le nœud `equal_r` qui vérifie l'égalité de 2 variables réelles avec une tolérance temporelle (`delta_t`) et numérique (`delta_v`).

```
node X_et_Y(
   A1DOM001DMA_X_AVALUE, A1DOM001DMA_Y_AVALUE: real;
   A1DOM003AIS_VALUE, A1DOM004AIS_VALUE: real)
returns (
   ok: bool)
var
  ok_X, ok_Y: bool;
let
  ok_X = equal_r<<d>>(delta_v, A1DOM001DMA_X_AVALUE, A1DOM003AIS_VALUE);
  ok_Y = equal_r<<d>>(delta_v, A1DOM001DMA_Y_AVALUE, A1DOM004AIS_VALUE);
  ok = ok_X and ok_Y;
tel
```

Le critère de couverture (pas défini ici) pourrait être de vérifier que l'on couvre toutes les zones de tous les domaines.

5.2.3 La zone courante

Le nœud `Cur_Zone` s'assure que les zones sont bien détectées par le Scada en vérifiant que le point testé se trouve bien dans la dite zone. Certaines zones apparaissant dans plusieurs domaines, on a encapsulé cette vérification dans les nœuds `in_zone_<Nom de la zone>`.

```
node Cur_Zone(
    A1DOM001DMA_NUM_CUR_ZONE : int;
    A1DOM001DMA_X_AVALUE, A1DOM001DMA_Y_AVALUE : real;
    A1DOM001DMA_NUM_ED_FORCED: int;
    A1DOM001DMA_NUM_ED_ELEC  : int)
returns (
    ok: bool;
    c21, c22, c23, c24, c25: bool);
var
  InB, InA2, InC : bool;
  okCur_ZoneB, okCur_ZoneA2, okCur_ZoneC : bool;
let
  InB = in_zone_B(A1DOM001DMA_X_AVALUE, A1DOM001DMA_Y_AVALUE,
                  A1DOM001DMA_NUM_ED_FORCED, A1DOM001DMA_NUM_ED_ELEC);
  InA2 = in_zone_A2(A1DOM001DMA_X_AVALUE, A1DOM001DMA_Y_AVALUE,
                    A1DOM001DMA_NUM_ED_FORCED, A1DOM001DMA_NUM_ED_ELEC);
  InC = in_zone_C(A1DOM001DMA_X_AVALUE, A1DOM001DMA_Y_AVALUE,
                  A1DOM001DMA_NUM_ED_FORCED, A1DOM001DMA_NUM_ED_ELEC);

  okCur_ZoneB  = equal_b<<d>>(InB,  A1DOM001DMA_NUM_CUR_ZONE = 4);
  okCur_ZoneA2 = equal_b<<d>>(InA2, A1DOM001DMA_NUM_CUR_ZONE = 2);
  okCur_ZoneC  = equal_b<<d>>(InC,  A1DOM001DMA_NUM_CUR_ZONE = 5);
  ok = okCur_ZoneB and okCur_ZoneA2 and okCur_ZoneC;

  c21 = (A1DOM001DMA_NUM_CUR_ZONE = 1);
  c22 = (A1DOM001DMA_NUM_CUR_ZONE = 2);
  c23 = (A1DOM001DMA_NUM_CUR_ZONE = 3);
  c24 = (A1DOM001DMA_NUM_CUR_ZONE = 4);
  c25 = (A1DOM001DMA_NUM_CUR_ZONE = 5);
tel
```

En terme de couverture, on requiert de passer au moins une fois par chaque zone.

5.2.4 Le forçage d'un domaine

Le nœud `Force` vérifie trois choses :

1. d'une part, que si un nœud précédemment forcé est déforcé alors l'ordre est bien pris en compte par le Scada ;
2. d'autre part, que lorsqu'aucun ordre de forçage ou de déforçage n'est reçu, la variable indiquant le numéro du domaine forcé ne change pas ;
3. enfin, que lorsqu'un ordre de forçage est passé, la variable indiquant le numéro du domaine forcé indique bien le bon domaine.

```
node Force(
    A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_DEFORCED1 : bool;
    A1DOM001DMA_V_FORCED2, A1DOM001DMA_V_DEFORCED2 : bool;
    A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_DEFORCED3 : bool;
    A1DOM001DMA_V_FORCED4, A1DOM001DMA_V_DEFORCED4 : bool;
    A1DOM001DMA_NUM_ED_FORCED: int)
returns (
    ok: bool;
    c31, c32, c33, c34, c35, c36, c37, c38, c39 : bool);
var
  ok_deforce1, ok_deforce2, ok_deforce3, ok_deforce4, ok_deforce: bool;
  ok_force1, ok_force2, ok_force3, ok_force4, ok_forcer: bool;
  ok_non_force: bool;
let
  -- 1 si un n\oe ud force est deforce alors l'ordre est bien pris en compte
  c31 = false -> (pre A1DOM001DMA_NUM_ED_FORCED = 1 and A1DOM001DMA_V_DEFORCED1);
  c32 = false -> (pre A1DOM001DMA_NUM_ED_FORCED = 2 and A1DOM001DMA_V_DEFORCED2);
  c33 = false -> (pre A1DOM001DMA_NUM_ED_FORCED = 3 and A1DOM001DMA_V_DEFORCED3);
  c34 = false -> (pre A1DOM001DMA_NUM_ED_FORCED = 4 and A1DOM001DMA_V_DEFORCED4);
  ok_deforce1 = (c31 => A1DOM001DMA_NUM_ED_FORCED = 0);
  ok_deforce2 = (c32 => A1DOM001DMA_NUM_ED_FORCED = 0);
  ok_deforce3 = (c33 => A1DOM001DMA_NUM_ED_FORCED = 0);
  ok_deforce4 = (c34 => A1DOM001DMA_NUM_ED_FORCED = 0);
  ok_deforce = (ok_deforce1 or ok_deforce2 or ok_deforce3 or ok_deforce4);

  -- 2 si aucun ordre n'est recu, le numero du domaine force ne change pas.
  c39 = tous_faux_8(A1DOM001DMA_V_FORCED1, A1DOM001DMA_V_DEFORCED1,
                    A1DOM001DMA_V_FORCED2, A1DOM001DMA_V_DEFORCED2,
                    A1DOM001DMA_V_FORCED3, A1DOM001DMA_V_DEFORCED3,
                    A1DOM001DMA_V_FORCED4, A1DOM001DMA_V_DEFORCED4);
   ok_non_force = true ->
      c39 => (A1DOM001DMA_NUM_ED_FORCED = pre A1DOM001DMA_NUM_ED_FORCED);

  -- 3 si un ordre de forcage est passe, le domaine force est le bon
  c35 = A1DOM001DMA_V_FORCED1;
  c36 = A1DOM001DMA_V_FORCED2;
  c37 = A1DOM001DMA_V_FORCED3;
  c38 = A1DOM001DMA_V_FORCED4;
  ok_force1 = (c35 => A1DOM001DMA_NUM_ED_FORCED = 1);
  ok_force2 = (c36 => A1DOM001DMA_NUM_ED_FORCED = 2);
  ok_force3 = (c37 => A1DOM001DMA_NUM_ED_FORCED = 3);
  ok_force4 = (c38 => A1DOM001DMA_NUM_ED_FORCED = 4);
  ok_forcer = ok_force1 and ok_force2 and ok_force3 and ok_force4;

  ok = ok_deforce and ok_forcer and ok_non_force;
tel
```

Les critères de couverture sont ici les conditions des trois différentes situations à vérifier :

- que chaque domaine précédemment forcé reçoive au moins une fois l'ordre de déforçage lui correspondant,
- que chaque domaine soit forcé au moins une fois,
- et enfin, que certaines fois, aucun ordre n'est passé (ni forçage, ni déforçage).

5.2.5 L'élection d'un domaine

Le nœud `Elec` vérifie que la variable indiquant le numéro du domaine élu correspond bien aux booléens indiquant si un domaine est activé. Là aussi, la vérification est effectuée avec une tolérance temporelle.

```
node Elec(
  A1DMA001OIW_LVALUE,A1DMA002OIW_LVALUE,A1DMA003OIW_LVALUE,A1DMA004OIW_LVALUE:bool;
  A1DOM001DMA_NUM_ED_ELEC: int)
returns (
    ok: bool;
    c41, c42, c43, c44: bool);
var
  ok_Elec1, ok_Elec2, ok_Elec3, ok_Elec4: bool;
let
  c41 = (A1DOM001DMA_NUM_ED_ELEC = 1);
  c42 = (A1DOM001DMA_NUM_ED_ELEC = 2);
  c43 = (A1DOM001DMA_NUM_ED_ELEC = 3);
  c44 = (A1DOM001DMA_NUM_ED_ELEC = 4);

  ok_Elec1 = equal_b<<d>>(c41, A1DMA001OIW_LVALUE);
  ok_Elec2 = equal_b<<d>>(c42, A1DMA002OIW_LVALUE);
  ok_Elec3 = equal_b<<d>>(c43, A1DMA003OIW_LVALUE);
  ok_Elec4 = equal_b<<d>>(c44, A1DMA004OIW_LVALUE);
  ok=ok_Elec1 and ok_Elec2 and ok_Elec3 and ok_Elec4;
tel
```

Afin de couvrir cet oracle, on s'assure de voir passer la variable indiquant le numéro du domaine au moins une fois par chaque numéro de domaine.

5.2.6 Les timers

Le nœud `Timers` vérifie les timers internes du Scada des deux zones d'accumulation en utilisant le nœud `Timer` qui est une modification du compteur `bistable` présenté en section 2.7 du SPF5.

```
node Timer(val_init, val_reset, val_incr : int; X, reset : bool)
returns (C : int);
var
  PC : int;
let
  PC = val_init -> pre C;
  C = if reset then val_reset else
      if X     then PC + val_incr else
                    PC;
tel
```

`Timer(val_init, val_reset, val_incr, X, reset)` renvoie un entier qui vaut :
- `val_init` à l'origine des temps ;
- `val_reset` quand `reset` est vrai ;
- sa valeur précédente sinon et si `X` est faux ;
- sa valeur précédente plus `val_incr` sinon (`X` est vrai).

Nous allons maintenant pouvoir utiliser ce nœud pour vérifier les compteurs internes du Scada.

```
node Timers(
```

```
    A1DOM001DMA_VAL_TIMER3  : int;
    A1DOM001DMA_VAL_TIMER5  : int;
    A1DOM001DMA_NUM_CUR_ZONE: int;
    A1DOM001DMA_V_RAZ_TIMER : bool)
returns (
    ok          : bool;
    c51, c52, c53, c54: bool;
  );
var
  VerifTimer3, VerifTimer5: bool;
  mon_timer3, mon_timer5: int;
let
  mon_timer3 = Timer(A1DOM001DMA_VAL_TIMER3, 0, temps_de_cycle_ms,
                     A1DOM001DMA_NUM_CUR_ZONE = 3,
                     A1DOM001DMA_V_RAZ_TIMER);
  mon_timer5 = Timer(A1DOM001DMA_VAL_TIMER5, 0, temps_de_cycle_ms,
                     A1DOM001DMA_NUM_CUR_ZONE = 5,
                     A1DOM001DMA_V_RAZ_TIMER);

  VerifTimer3 = abs_int(mon_timer3 - A1DOM001DMA_VAL_TIMER3) < 2000;
  VerifTimer5 = abs_int(mon_timer5 - A1DOM001DMA_VAL_TIMER5) < 2000;

  ok=VerifTimer3 and VerifTimer5;

  c51 = A1DOM001DMA_VAL_TIMER3 > 0 and A1DOM001DMA_V_RAZ_TIMER;
  c52 = A1DOM001DMA_VAL_TIMER5 > 0 and A1DOM001DMA_V_RAZ_TIMER;
  c53 = A1DOM001DMA_VAL_TIMER3 > 60000 and A1DOM001DMA_V_RAZ_TIMER;
  c54 = A1DOM001DMA_VAL_TIMER5 > 60000 and A1DOM001DMA_V_RAZ_TIMER;
tel
```

5.2.7 Les alarmes

Le nœud `Alarmes` vérifie que les alarmes se déclenchent bien et au bon moment dans le Scada, c'est-à-dire que :

1. l'alarme se déclenche immédiatement dans la zone interdite,
2. l'alarme se déclenche au bout de 60 secondes dans les zones d'accumulation,
3. aucune alarme ne se déclenche dans les zones autorisées,
4. toutes ces alarmes retombent une fois qu'elles n'ont plus de raison d'être présentes.

Nous vérifions ces propriétés comme dans la section 3.1 du SPF5 en utilisant le nœud `est_suivi_de` (section 2.4 du SPF5).

```
node Alarmes(
    A1DOM001DMA_ALARME1     : int;
    A1DOM001DMA_ALARME2     : int;
    A1DOM001DMA_ALARME3     : int;
    A1DOM001DMA_ALARME4     : int;
    A1DOM001DMA_ALARME5     : int;
    A1DOM001DMA_VAL_TIMER3  : int;
    A1DOM001DMA_VAL_TIMER5  : int;
    A1DOM001DMA_NUM_CUR_ZONE: int)
returns (
```

```
    ok: bool;
    c61, c62, c63, c64, c65, c66, c67: bool);
var
  Alarme_Zone2, Alarme_Zone3, Alarme_Zone5: bool;
  Fin_Alarme_Zone2, Fin_Alarme_Zone3, Fin_Alarme_Zone5: bool;
let
  Alarme_Zone2 = -- 1.  se declenche immediatement dans la zone interdite
    est_suivi_de(2.0,
      r_edge(A1DOM001DMA_NUM_CUR_ZONE = 2), r_edge(A1DOM001DMA_ALARME2 = 1));
  Alarme_Zone3 = -- 2. se declenche au bout de 60s dans les zones d'accumulation
    est_suivi_de(2.0,
      r_edge(A1DOM001DMA_VAL_TIMER3 > 60000), r_edge(A1DOM001DMA_ALARME3 = 1));
  Alarme_Zone5 = -- 2. idem
    est_suivi_de(2.0,
      r_edge(A1DOM001DMA_VAL_TIMER5 > 60000), r_edge(A1DOM001DMA_ALARME5 = 1));
  Fin_Alarme_Zone2 = -- 4. les alarmes retombent
    est_suivi_de(2.0,
      f_edge(A1DOM001DMA_NUM_CUR_ZONE = 2), f_edge(A1DOM001DMA_ALARME2 = 1));
  Fin_Alarme_Zone3 = -- 4. idem
    est_suivi_de(2.0,
      f_edge(A1DOM001DMA_VAL_TIMER3 > 60000), f_edge(A1DOM001DMA_ALARME3 = 1));
  Fin_Alarme_Zone5 = -- 4. idem
    est_suivi_de(2.0,
      f_edge(A1DOM001DMA_VAL_TIMER5 > 60000), f_edge(A1DOM001DMA_ALARME5 = 1));
  ok =
    Alarme_Zone2 and Alarme_Zone3 and Alarme_Zone5 and
    Fin_Alarme_Zone2 and Fin_Alarme_Zone3 and Fin_Alarme_Zone5 and
    A1DOM001DMA_ALARME1 = 0 and  -- 3. pas d'alarme dans les zones autorisees
    A1DOM001DMA_ALARME4 = 0;     -- 3. idem

  c61 = (A1DOM001DMA_NUM_CUR_ZONE = 1);
  c62 = (A1DOM001DMA_NUM_CUR_ZONE = 2);
  c63 = (A1DOM001DMA_NUM_CUR_ZONE = 3) and (A1DOM001DMA_VAL_TIMER3 > 60000);
  c64 = (A1DOM001DMA_NUM_CUR_ZONE = 3) and (A1DOM001DMA_VAL_TIMER3 <= 60000);
  c65 = (A1DOM001DMA_NUM_CUR_ZONE = 4);
  c66 = (A1DOM001DMA_NUM_CUR_ZONE = 5) and (A1DOM001DMA_VAL_TIMER5 > 60000);
  c67 = (A1DOM001DMA_NUM_CUR_ZONE = 5) and (A1DOM001DMA_VAL_TIMER5 <= 60000);
tel
```

On considère cet oracle couvert si on est passé par toutes les zones et si les timers ont dépassé la minute dans les zones d'accumulation.

6 Conclusion

Nous avons montré dans ce document comment nous pouvions avec Lurette (et les langages associés, Lustre et Lutin) automatiser entièrement le jeu d'un cahier de test Scada conçu pour être joué manuellement par un opérateur. Nous avons ensuite montré comment le stimulateur Lutin simulant le comportement de l'opérateur pouvait être légèrement modifié pour être rendu plus aléatoire, permettant de générer toute une famille de scénarios sur le modèle du cahier de test initial.

Dans un second temps, nous avons illustré l'utilisation de Lurette pour vérifier la conformité du système sous test en se basant directement sur une description informelle des attendus fonctionnels plutôt que sur des cahiers de test (eux-mêmes obtenus à partir d'attendus fonctionnels).

Parce que le système sous test était assez simple, mais néanmoins de façon inattendue, on peut remarquer que les séquences de test générées par la version rendue aléatoire du cahier de test et par la formalisation directe des attendus fonctionnels sont comparables en terme de couverture. Pourtant, le style de la description est différent. L'intérêt de la deuxième approche est d'être plus modulaire. En effet, la description de l'environnement est effectuée sous la forme de 4 nœuds mis en parallèle, chaque nœud s'occupant d'une partie des variables à générer. La version issue du cahier de test, quant à elle, est constituée d'un seul nœud gérant toutes les variables. Cela n'est sans doute pas important pour un petit environnement (ici, on gère une trentaine de variables). La modularité de la description ne favorise pas uniquement la réutilisabilité, mais aussi la concision. En effet, mettre en parallèle 2 programmes Lutin revient effectuer le produit de 2 automates ; et la taille du produit de 2 automates de taille `n1` et `n2` est $\texttt{n1} \times \texttt{n2}$. Cette discussion vaut d'ailleurs aussi bien pour le stimulateur que pour les oracles.

Néanmoins, il existe des cas où l'écriture de scénarios est indispensable. C'est notamment le cas pour les systèmes complexes, possédant de la mémoire et des modes de fonctionnement, car ce type de systèmes a besoin de scénarios pour pouvoir être testé exhaustivement dans chacun de ses modes. Cela n'est pas vraiment le cas dans ce cas d'étude ; si le cahier de test est présenté sous forme d'un scénario séquentiel, c'est surtout pour permettre à l'opérateur de procéder par étape, et lui éviter d'avoir à surveiller trop de variables à la fois.

Certes, l'effort mis en jeu pour obtenir un cahier de test automatisé est certainement un peu supérieur à celui nécessaire pour rédiger le cahier de test classique et de le jouer manuellement. Cependant, l'apport d'une version automatisée est énorme, en particulier en terme de non-regression. En effet, ces tests peuvent être joués à chaque modification, même mineure, de l'objet.

Par ailleurs, une bonne surprise de ce travail est d'avoir constaté après coup que la version « rendu aléatoire » du cahier de test couvrait en fait les 20 autres cahiers qui avaient été rédigés initialement pour tester l'objet. Ainsi, non seulement le test est automatisé, mais en plus la couverture obtenue à partir d'un scénario aléatoire est égale (voire supérieure) à celle obtenue en rédigeant 21 cahiers de test.