ANALYSE DES PERFORMANCES DE DIAGNOSTIC D'UN SYSTEME AU MOYEN D'UNE MODELISATION ALTARICA 3.0

Emmanuel Clement, Dominique Riera, Michel Batteux

ANALYSE DES PERFORMANCES DE DIAGNOSTIC D'UN SYSTEME AU MOYEN D'UNE MODELISATION ALTARICA 3.0

DIAGNOSIS PERFORMANCE ASSESSMENT OF A SYSTEM BY THE USE OF ALTARICA 3.0 MODELS

Emmanuel CLEMENT, Dominique RIERA
Thales Defense Mission Systems
10 Avenue 1ère Dfl
29200 Brest

Michel BATTEUX
IRT SystemX
8, avenue de la Vauve
CS 90070 Palaiseau

Résumé

Nous proposons une méthode permettant d'analyser les performances d'une stratégie de diagnostic d'un système complexe par l'intermédiaire d'une modélisation avec AltaRica 3.0 (Prosvirnova et al.,2013). Cette analyse permettra de démontrer les niveaux de taux de couverture de tests intégrés et de taux de localisation de pannes.

Summary

We are presenting an approach which aims to assess diagnosis performance of complex systems by the use of AltaRica 3.0 models. This analysis aims to demonstrate fault cover rates and fault localization rates

Introduction

1 Contexte

La complexité croissante des systèmes entraîne, notamment en Sûreté de fonctionnement, la nécessité de progresser dans les méthodes d'analyses en développant l'ingénierie dirigée par les modèles (MBSA) (Riera et al.,2012). Le MBSA (Model-Based Safety Assessment) permet, par la réalisation de modèles de plus haut niveau, c'est-à-dire proches des architectures fonctionnelles et physiques des systèmes, d'assurer le partage et la cohérence des hypothèses, des données d'entrée et des résultats des analyses entre les différents acteurs de la Sûreté de fonctionnement. Cela conduit à une cohérence, à une validation et à une capitalisation des modèles et des analyses. C'est dans ce cadre qu'intervient cette communication.

Cette démarche s'inscrit dans une démarche plus globale d'organisation des analyses FMDTS (Fiabilité, Maintenabilité, Disponibilité, Testabilité et Sécurité) autour d'un modèle MBSA unique, validé et partagé.

2 Objectif

L'objectif principal de cette publication est de proposer une méthode permettant d'analyser les performances d'une stratégie de diagnostic d'un système complexe par l'intermédiaire d'une modélisation avec AltaRica 3.0 (Prosvirnova et al.,2013). Cette analyse permettra de démontrer les niveaux de taux de couverture de tests intégrés, de taux de localisation de pannes ainsi que les probabilités de pannes dangereuses non détectées (CEI 61508).

3 Approche

A partir d'un modèle MBSA habituel conçu pour analyser les évènements redoutés, nous proposons une méthode pour le compléter avec les tests intégrés (Built-in tests) du système. Ceci dans le but de démontrer l'atteinte des niveaux de taux de couverture et de taux de localisation de pannes. Cette modélisation permet de plus de quantifier la probabilité de pannes dangereuses non détectées (CEI 61508).

Deux aspects distincts seront développés :

- L'intégration de la stratégie de tests internes (Built-in tests) du système dans le modèle MBSA par l'intermédiaire d'« observer » (Prosvirnova et al.,2013);
- La quantification des performances de diagnostic par post-traitement des résultats obtenus après analyse du modèle MBSA (ARNAUD.,2014)

Les « observer », représentant les tests internes, permettent d'obtenir une équation booléenne de la capacité de détection de ce test. A partir de cette équation, la liste des coupes minimales réduite à l'ordre 1 est obtenue. Les coupes minimales d'ordre 1 représentent la liste des défaillances couvertes par le test.

Afin de bien illustrer le sujet, un cas concret sera étudié tout au long de la communication. Les patterns de modélisation des tests seront explicités. Ensuite, l'obtention des coupes minimales réduites à l'ordre 1 et les résultats de l'analyse seront décrits.

La communication est construite de la façon suivante :

- Présentation du cas industriel ;
- Présentation d'AltaRica 3.0 et d'OpenAltaRica ;
- Description du cas concret ;
- Description de la méthode et des patterns de modélisation ;
- Conclusion.

Cas industriel

1. Contexte

La testabilité est la discipline de la Sûreté de fonctionnement qui s'intéresse aux capacités de diagnostic d'un système. Deux performances principales sont estimées :

1. Le taux de couverture de pannes ;
2. Le taux de localisation.

1.1 Taux de couverture de pannes

Le taux de couverture de pannes représente le rapport entre la somme des taux de défaillances détectés par au moins un test et la somme de l'ensemble des taux de défaillances du système. Ne sont retenus dans le taux de couverture, que les modes de défaillances ayant un impact sur la capacité opérationnelle ou bien sur la sécurité du système.

$$TC = \frac{\sum \lambda_{détecté}}{\sum \lambda} \qquad \{1\}$$

1.2 Taux de localisation

Le taux de localisation à 1, 2 ou 3 éléments remplaçables représente la capacité à localiser l'élément défectueux directement par la stratégie de tests.
Pour chaque signature de tests, les modes de défaillances sont classés de la probabilité la plus forte à la probabilité la plus faible. Le taux de localisation à X item(s) est calculé en sommant les taux de défaillances des X premier(s) mode(s) de défaillances, ordonnés du plus probable au moins probable, de chaque signature de test divisé par la somme des taux de défaillances détectés du système.

$$TL_{1\ item} = \frac{\sum_{signature} \lambda_{premier\ choix}}{\sum \lambda_{détecté}} \qquad \{2\}$$

$$TL_{2\ items} = \frac{\sum_{signature} \lambda_{premier\ choix} + \sum_{signature} \lambda_{deuxième\ choix}}{\sum \lambda_{détecté}} \qquad \{3\}$$

$$TL_{3\ items} = \frac{\sum_{signature} \lambda_{premier\ choix} + \sum_{signature} \lambda_{deuxième\ choix} + \sum_{signature} \lambda_{troisème\ choix}}{\sum \lambda_{détecté}} \qquad \{4\}$$

2. Présentation du cas industriel

La méthode que nous proposons a été appliquée sur une carte électronique complexe et portait sur la capacité d'autodiagnostic du système.

La performance à analyser est le taux de couverture de pannes.

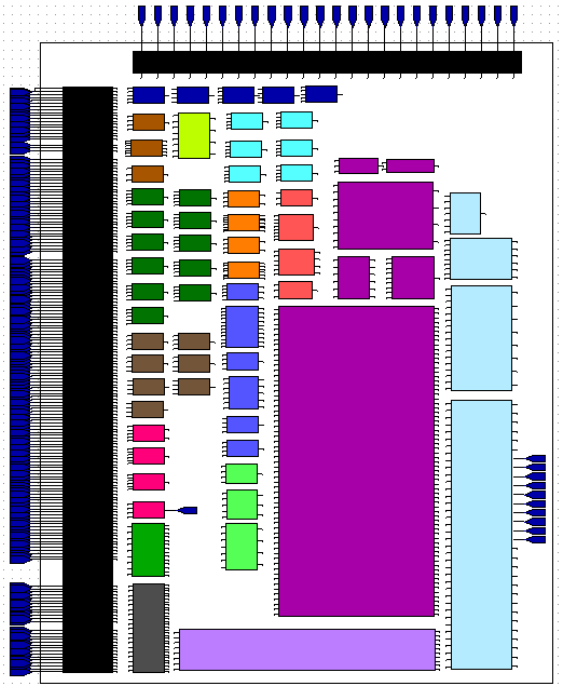

Figure 1 - Modèle de la carte sous eXpress

Les résultats de la méthode que nous présentons ont été comparés aux résultats d'une analyse réalisée avec un logiciel du marché spécialisé dans les analyses de testabilité « eXpress » de la société « DSI International ».

OpenAltaRica et AltaRica 3.0

1. AltaRica 3.0

AltaRica 3.0 (Prosvirnova *et al.*, 2013) est, comme son nom l'indique, la troisième version du langage de modélisation AltaRica . Il améliore la version précédente, AltaRica Data-Flow (Boiteau *et al.*, 2006) suivant deux points fondamentaux. La sémantique d'exécution basée les Systèmes de Transitions Gradées (GTS) et le paradigme de structuration basé sur S2ML.

En premier lieu sa sémantique d'exécution est basée sur les Systèmes de Transitions Gardées, GTS pour Guarded Transitions Systems (Rauzy, 2008). Les GTS généralisent les formalismes classiques d'analyse du risque (arbres de défaillance, chaînes de Markov, réseaux de Pétri stochastiques, etc.), sans augmenter la complexité des calculs d'indicateurs du risque. L'exécution des modèles AltaRica 3.0 est similaire aux autres formalismes à événements discrets dans le sens où chaque fois qu'une transition est tirable, elle est alors planifiée et sera alors potentiellement tirée après un certain délai (Cassandras et al., 2008), (Zimmerman, 1976). Ces délais peuvent être déterministes ou stochastiques ; et dans ce dernier cas, AltaRica 3.0 fournit les délais usuels (du type exponentiel, Weibull, etc.) et des délais empiriques.

De plus, le paradigme de structuration du langage AltaRica 3.0 est basé sur S2ML, pour System Structure Modeling Language (Batteux et *al.* 2015). S2ML unifie les deux paradigmes (de structuration) dominants des langages de modélisation, i.e. l'orienté objet et l'orienté prototype. Il permet de modéliser suivant deux approches combinables. L'approche dite 'top-down', avec une vision au niveau système permettant la réutilisation de schémas de modélisation, et donc utilisant le paradigme orienté prototype. L'approche dite 'bottom-up', avec une vision au niveau composants permettant la réutilisation de composants, définis en bibliothèques, et donc utilisant le paradigme orienté objet.

Le pouvoir d'expression du langage AltaRica 3.0 est seulement une partie de la solution d'analyse du risque de systèmes complexes. L'autre partie concerne les outils d'évaluation associés, notamment l'efficacité pratique des algorithmes d'évaluation. En effet, l'évaluation pour la sûreté ou la fiabilité est un problème dit de complexité #P-hard (Valiant 1979). Cela signifie que cette barrière de complexité n'est pas un problème de technologie, mais un problème théorique. Il est donc nécessaire de concevoir des outils d'évaluation implémentant efficacement les algorithmes de traitement. Le langage AltaRica 3.0 vient de ce fait avec un ensemble d'outils permettant de calculer différents indicateurs. La Figure 2 représente l'ensemble des outils d'évaluation associés au langage AltaRica 3.0. Cette figure indique aussi les outils graphiques permettant de concevoir et éditer des modèles AltaRica 3.0, mais aussi de les simuler graphiquement. Différents outils permettent d'évaluer les modèles AltaRica 3.0. Le simulateur pas-à-pas est l'outil permettant de réaliser des expériences virtuelles sur les modèles (vérification manuelle de modèles par exemple). Les deux outils de compilation : vers les arbres de défaillance et vers les chaînes de Markov (pas encore disponible) permettent de générer, à partir de modèles AltaRica 3.0, des modèles vers ces formalismes cibles. L'utilisation d'outils d'évaluation associés à ces formalismes est ensuite possible. Le générateur de séquences permet d'obtenir l'ensemble des séquences d'évènements amenant à un événement redouté considéré. Enfin, le simulateur stochastique permet d'obtenir des statistiques sur des exécutions aléatoires de modèles AltaRica 3.0. L'ensemble de ces outils est distribué au sein de la plateforme OpenAltaRica issu du projet du même nom.

2. OpenAltaRica

Le projet OpenAltaRica est un projet réalisé au sein de l'Institut de Recherche Technologique (IRT) SystemX et en partenariat avec Thales, Apsys, Safran et AltaRica Association. D'une durée de 5 ans et ayant démarré fin 2014, ce projet vise l'implémentation de la plateforme de référence pour le langage AltaRica 3.0. Cette plateforme d'expérimentation, disponible sur le site du projet (www.openAltaRica.fr), intègre déjà l'ensemble des outils développés. Elle sera complétée par les outils en cours de développement.

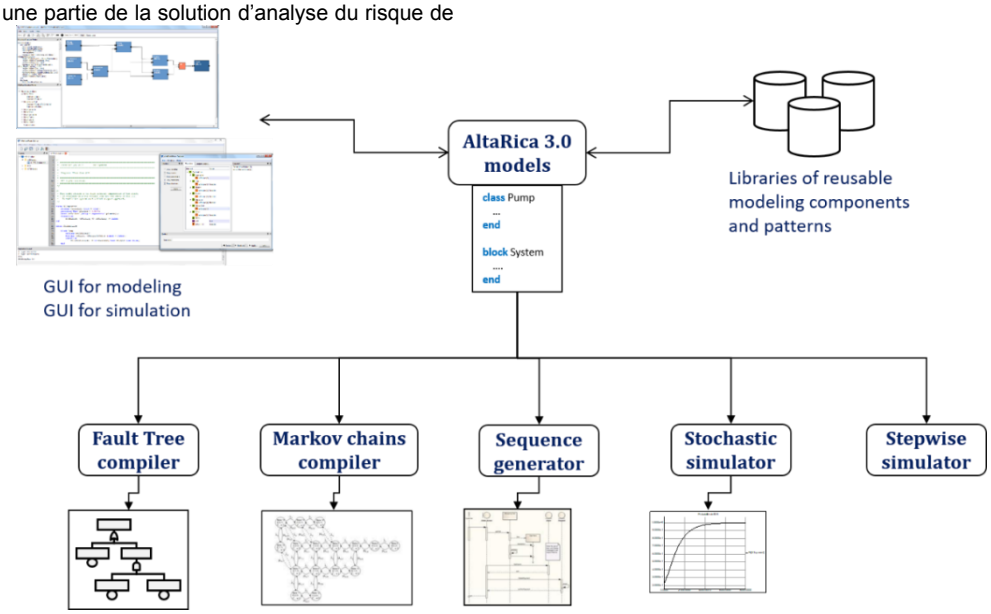

Figure 2 -Description des outils d'évaluation associés au langage AltaRica 3.0

Description de la méthode

1. Principe général

La méthode que nous proposons est de créer un modèle de dépendances, fonctionnelles ou bien organiques en fonction du choix, et d'y ajouter les tests. La simulation de ce modèle de dépendance permet, par propagation des effets des défaillances, d'identifier les modes de défaillances vus par chacun des tests.

La création de ce modèle est organisée en 3 phases principales. Premièrement, nous réalisons le modèle AltaRica 3.0 du système et des tests à analyser. Deuxièmement, nous simulons le modèle afin d'obtenir les différentes signatures de tests. Troisièmement, nous réalisons un post-traitement des signatures de tests afin de quantifier les performances de couverture et de localisations des tests.

2. Cas applicatif

Afin d'illustrer la méthode, un cas applicatif simple sera traité. Ce cas applicatif est constitué de 4 LRU (**L**ine-**R**eplaceable **U**nit), de 7 modes de défaillances et de 2 tests. La figure 3 représente ce modèle :

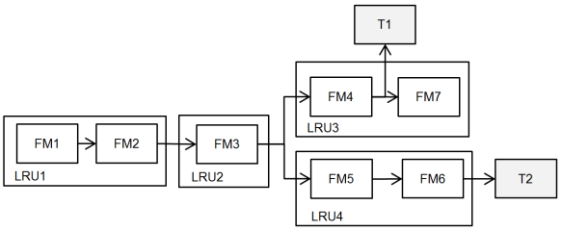

Figure 3 - Synoptique du cas applicatif

Les éléments remplaçables sont : LRU1, LRU2, LRU3 et LRU4. Les modes de défaillances sont : FM1, FM2 ; FM3, FM4, FM5, FM6, FM7. Finalement, les 2 tests sont T1 et T2.

3. Modélisation AltaRica 3.0

3.1 Modélisation des modes de défaillances

Le niveau le plus bas du modèle retenu est le mode de défaillances des LRU. Il y a donc 7 modes de défaillances à modéliser. Pour cela, nous avons réalisé une classe AltaRica « FailureMode » qu'il sera nécessaire d'instancier autant de fois que nécessaire.

Le modèle des modes de défaillance est constitué d'une entrée « IN », d'une sortie « OUT », de deux états « KO=TRUE » et « KO=FALSE ». Son comportement est décrit de la façon suivante : « OUT » est égal à « IN » si « KO=FALSE » sinon « OUT » vaut « False ». Une représentation graphique possible serait le bloc ci-après.

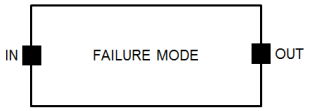

Figure 4 - Représentation graphique d'un mode de défaillance

La figure 5 représente le graphe d'états retenu :

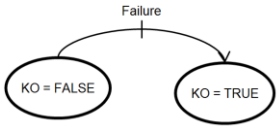

Figure 5 - Graphe d'états d'un mode de défaillance

Il n'est pas nécessaire de renseigner une loi de probabilité pour la transition « Failure » car nous souhaitons obtenir de la simulation du modèle AltaRica les signatures des tests. Seule une analyse qualitative est menée directement sur le modèle.

La figure 6 représente le code AltaRica correspondant à la classe « FailureMode » :

```
class FailureMode
  // Flows
  Boolean in, out ( reset = false );
  // States
  Boolean KO (init = false);
  // Events
  event Failure;
  // Transitions :
  transition
    Failure : KO == false -> KO := true ;
  // Assertions
  assertion
    out := in and (KO == false );
end
```

Figure 6 - Code AltaRica 3.0 d'un mode de défaillance

3.2 Modélisation des unités remplaçables

Il n'est pas nécessaire de représenter l'architecture de notre cas test directement dans le code AltaRica puisque la relation « mode » de défaillance » et « unités remplaçables » est réalisée par le post-traitement. En revanche, la représentation de l'architecture, dans le modèle, apporte un niveau d'information nécessaire au partage du modèle par les différents métiers. Par souci de compréhension, nous vous proposons les deux versions. Le premier avec la représentation des « unités remplaçables » et le deuxième comprenant seulement les modes de défaillances.

Pour représenter le niveau « unité remplaçable », nous créons des « block(s) » AltaRica contenant les modes de défaillances spécifiques à chacun des « LRU » ainsi que les « flux » internes au « LRU » et qui relient les modes de défaillances entre eux.

Ce niveau de modélisation ne comprend aucun comportement spécifique. Seule, la classe « FailureMode » est instanciée autant que nécessaire.

Le code AltaRica 3.0 du LRU1 est le suivant.

```
block LRU1
  // Failure modes instanciation
  FailureMode FM1, FM2;
  // Flux instanciation
  assertion
    FM2.in := FM1.out;
end
```

Figure 7 - Code AltaRica 3.0 du LRU1

3.3 Modélisation des tests

L'intégration des tests dans le modèle est réalisée par l'ajout d'« observer ». Un « observer » est un objet AltaRica qui permet d'observer des propriétés spécifiques du modèle et défini par l'utilisateur.

Le code AltaRica 3.0 correspondant aux tests T1 et T2 est présenté ci-dessous.

```
// Tests initialisation
observer Boolean T1 = LRU3.FM4.out == false ;
observer Boolean T2 = LRU4.FM6.out == false ;
```

Figure 8 - Code AltaRica 3.0 des tests

3.4 Modélisation du niveau système

La dernière étape de la création du modèle est la réalisation du niveau « Système ». Ce niveau système comprend la déclaration des « LRU », la création des « flux » entre les modes de défaillance ainsi que la déclaration des « observer ».

Le code AltaRica 3.0 du niveau « Système » comprenant la représentation des « LRU » est le suivant.

```
block system
  block LRU1
    // Failure modes instanciation
    FailureMode FM1, FM2;
    // Flux instanciation
    assertion
      FM2.in := FM1.out;
  end
  block LRU2
    // Failure modes instanciation
    FailureMode FM3;
  end
  block LRU3
    // Failure modes instanciation
    FailureMode FM4, FM7;
    // Flux instanciation
    assertion
      FM7.in := FM4.out;
  end
  block LRU4
    // Failure modes instanciation
    FailureMode F51, FM6;
    // Flux instanciation
    assertion
      FM6.in := FM5.out;
  end
  assertion
    // Flux instanciation
    LRU1.FM1.in := true ;
    LRU2.FM3.in := LRU1.FM2.out ;
    LRU3.FM4.in := LRU1.FM3.out ;
    LRU4.FM5.in := LRU1.FM3.out ;
  // Tests initialisation
  observer Boolean T1 = LRU3.FM4.out == false ;
  observer Boolean T2 = LRU4.FM6.out == false ;
end
```

Figure 9 - Code AltaRica 3.0 du système avec architecture

Le code AltaRica 3.0 synthétique est le suivant :

```
block system
  // Failure mode instanciation
  FailureMode FM1, FM2, FM3, FM4,
              FM5, FM6, FM7;
  // flux instanciation
  assertion
    FM1.in := true ;
    FM2.in := FM1.out;
    FM3.in := FM2.out;
    FM4.in := FM3.out;
    FM5.in := FM3.out;
    FM6.in := FM5.out;
    FM7.in := FM4.out;
  // Tests initialisation
  observer Boolean T1 = FM4.out == false ;
  observer Boolean T2 = FM6.out == false ;
end
```

Figure 10 - Code AltaRica 3.0 du système sans architecture

3. Simulation du modèle

3.1 Généralité

Nous souhaitons obtenir du modèle, la signature des tests. Dit autrement, nous souhaitons obtenir la liste des coupes minimales d'ordre 1 de chacun des « observer ».

Deux solutions sont envisageables. Premièrement, nous pouvons, avec la plateforme OpenAltaRica, générer l'arbre de défaillance correspondant à chaque « observateur », puis, avec l'aide d'outils comme Arbre-Analyste (CLEMENT et al.,2014), en obtenir les coupes minimales à l'ordre 1. Deuxièmement, nous pouvons utiliser le simulateur pas-à-pas de la plateforme OpenAltaRica. Celui-ci permet de tirer un ensemble d'instructions sur le modèle puis d'en obtenir les valeurs des « observer ». La deuxième solution est apparue beaucoup plus performante, en termes de temps de simulation, sur les modèles de grandes tailles. Pour cette raison, nous présentons, dans la suite de la publication, la deuxième solution.

3.1 Algorithme de simulation

Le simulateur pas-à-pas permet de charger un modèle puis de jouer des instructions prévues à l'avance afin d'obtenir les valeurs des « observer ». Ces instructions sont générées automatiquement puis transmises directement au simulateur pas-à-pas.
L'algorithme employé est décrit dans le logigramme suivant.

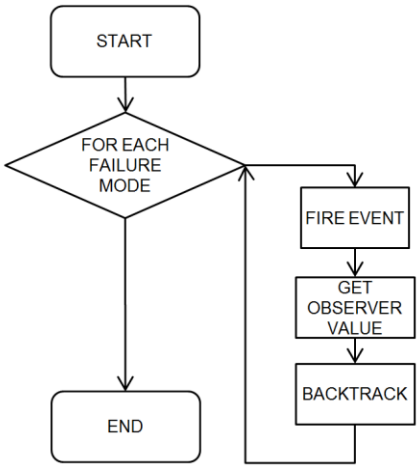

Figure 11 - Algorithme de la simulation

Le fichier contenant les instructions à jouer dans le simulateur pas-à-pas pour mener l'analyse sur notre cas applicatif est le suivant.

```
1  print tr
2  fire 0
3  print o
4  back
5  fire 1
6  print o
7  back
8  fire 2
9  print o
10 back
11 fire 3
12 print o
13 back
14 fire 4
15 print o
16 back
17 fire 5
18 print o
19 back
20 fire 6
21 print o
22 back
23 quit
```

Figure 12 - Instruction du simulateur

Les instructions sont les suivantes :

- « print tr » : affiche la liste des transitions ;
- « fire x » : tire la transition n°x ;
- « print o »: affiche l'état des « observateurs » ;
- « back » : annule le tirage précédent ;
- « quit » : quitte la simulation.

Les résultats de la simulation sont de la forme suivante :

```
1  Fireable transition(s):
2    [0]   LRU1.FM1.Failure (delay = constant( 1.0 ))
3    [1]   LRU1.FM2.Failure (delay = constant( 1.0 ))
4    [2]   LRU2.FM3.Failure (delay = constant( 1.0 ))
5    [3]   LRU3.FM4.Failure (delay = constant( 1.0 ))
6    [4]   LRU3.FM7.Failure (delay = constant( 1.0 ))
7    [5]   LRU4.FM5.Failure (delay = constant( 1.0 ))
8    [6]   LRU4.FM6.Failure (delay = constant( 1.0 ))
9  Observer(s):
10   T1 = true
11   T2 = true
12 Observer(s):
13   T1 = true
14   T2 = true
15 Observer(s):
16   T1 = true
17   T2 = true
18 Observer(s):
19   T1 = true
20   T2 = false
21 Observer(s):
22   T1 = false
23   T2 = false
24 Observer(s):
25   T1 = false
26   T2 = true
27 Observer(s):
28   T1 = false
29   T2 = true
```

Figure 13 - Résultat de la simulation

Nous obtenons la liste des transitions qui peuvent être tirées. Cette liste nous permet ensuite de réaliser des tirages (fire event) et retours en arrière (backtrack). À chaque tirage, nous sauvegardons les valeurs des observateurs.

4. Post-traitement des résultats de la simulation

Le post-traitement se déroule en trois grandes étapes.

1. Construisons des signatures des tests à partir des résultats de la simulation du modèle ;
2. Quantification du taux de couverture ;
3. Quantifions des performances de localisation.

4.1 Construction des signatures des tests

A partir des résultats de la simulation, nous pouvons reconstruire les signatures des tests.

La première partie des résultats nous informe de l'ordre des tirages des évènements. La suite indique l'état des « observer » pour chaque tirage. Ainsi, les lignes 10 et 11 de la figure 13 nous informent que le mode de défaillance « FM1 » de l'élément « LRU1 » est vu par les tests « T1 » et « T2 ».

En analysant tous les résultats de cette façon, nous reconstruisons les signatures comme présenté dans le tableau ci-dessous.

SIGNATURE	DEFAILLANCES
T1	FM4
T2	FM5 FM6
T1+T2	FM1 FM2 FM3

Figure 14 - Signatures des tests par mode de défaillance

4.2 Quantification du taux de couverture

Le tableau ci-après présente la valeur des taux de défaillance pour chaque mode de défaillances.

MODE	LAMBDA
FM1	1,0E-06
FM2	1,0E-06
FM3	1,0E-06
FM4	1,0E-06
FM5	1,0E-06
FM6	1,0E-06
FM7	1,0E-06

Figure 15 - Taux de défaillance

Le taux de couverture est obtenu de la façon suivante :

$$TC = \frac{FM1+FM2+FM3+FM4+FM5+FM6}{FM1+FM2+FM3+FM4+FM5+FM6+FM7}$$

$$TC = \frac{6.10^{-6}}{7.10^{-6}} = 86\% \qquad \{5\}$$

4.3 Quantification du taux de localisation

Les taux de localisation sont quantifiés à partir de la table des signatures. La première étape est de concaténer les modes de défaillances de chaque signature par LRU. Puis, on ordonne, pour chaque signature, les LRU par leur probabilité de pannes localisées, de la plus importante à la plus faible. Ainsi, la table des signatures de tests devient :

SIGNATURE	DEFAILLANCES
T1	LRU3[FM4]
T2	LRU4[FM5, FM6]
T1+T2	LRU1[FM1, FM2], LRU2[FM3]

Figure 16 - Signatures des tests par LRU

Le calcul du taux de localisation à une LRU est réalisé de la façon suivante :

$$TL_{1\ item} = \frac{1.10^{-6} + 2.10^{-6} + 2.10^{-6}}{1.10^{-6} + 2.10^{-6} + 2.10^{-6} + 1.10^{-6}}$$

$$TL_{1\ item} = \frac{5.10^{-6}}{6.10^{-6}} = 83\% \qquad \{6\}$$

Le calcul du taux de localisation à 2 LRU est réalisé de la façon suivante :

$$TL_{2\ items} = \frac{1.10^{-6} + 2.10^{-6} + 2.10^{-6} + 1.10^{-6}}{1.10^{-6} + 2.10^{-6} + 2.10^{-6} + 1.10^{-6}}$$

$$TL_{2\ items} = \frac{6.10^{-6}}{6.10^{-6}} = 100\% \qquad \{7\}$$

Cas industriel

La méthode présentée dans cette publication a été utilisée pour l'analyse du taux de couverture de tests d'une carte électronique. Les résultats ont été comparés à un outil du commerce spécialiste des analyses de testabilité : le logiciel « eXpress ».

Le modèle est constitué de 65 blocs fonctionnels et de 621 modes de défaillances. Le modèle contient un nombre important de flux bouclés ainsi que des flux bidirectionnels. Le code AltaRica 3.0 comprend en tout 6.300 lignes.

D'un point de vue temps de calcul, la durée de la simulation est à peine perceptible : moins de 3 secondes.

Le tableau ci-après présente les résultats des calculs du taux de couverture.

	FAULTS DETECTED	
	QUANTITY	PROBABILITY
eXpress	47%	79,4%
AltaRica	47%	79,4%

Figure 17 - Comparaison des résultats

Conclusion

Cette publication a pour but de présenter une nouvelle méthode de quantification des performances de testabilité par l'utilisation d'un modèle AltaRica 3.0.

Cette méthode s'articule autour de trois étapes principales. Premièrement, un modèle AltaRica 3.0 de notre système est réalisé afin de représenter les relations de dépendance des constituants physiques ou fonctionnels du système. Deuxièmement, le simulateur « pas-à-pas » est exécuté sur le modèle afin d'obtenir les états des « observer » correspondants aux tests intégrés. Puis, troisièmement, un post-traitement automatisé est mené sur les résultats de la simulation « pas-à-pas » afin de quantifier les performances de détection et de localisation.

Nous avons présenté une application de la méthode à un cas industriel complexe. Les résultats ainsi obtenus sont cohérents des résultats issus d'un logiciel dédié du marché.

Une réflexion est actuellement menée pour faire évoluer la méthode afin de prendre en compte les pannes multiples dans l'estimation des performances de localisation. Par ailleurs, il est aussi envisageable de réaliser des analyses de conception et d'optimisation de la stratégie de test dans une optique de spécification.

Par ailleurs, l'utilisation d'une méthode telle que celle présentée ici, permet de construire des analyses de testabilité à partir de modèles issues de logiciels d'ingénierie système basée sur les modèles (Capella).

Cette démarche s'inscrit dans une démarche plus globale d'organisation des analyses FMDTS (Fiabilité, Maintenabilité, Disponibilité, Testabilité et Sécurité) autour d'un modèle MBSA unique, validé et partagé.

8 Références

CEI 61508 - Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité

T. PROSVIRNOVA, M. BATTEUX, P-A BRAMERET, A. CHERFI, T. FRIEDLHUBER, J-M ROUSSEL, A. RAUZY, 2013, The AltaRica 3.0 Project for Model-Based Safety Assessment, in Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013, York (Great Britain).

D. RIERA, F. MILCENT, J. PARISOT, E. CLEMENT, 2012, Modélisation dynamique en Sûreté de Fonctionnement : une avancée pour l'analyse des systèmes complexes, Actes du Congrès Lambda-Mu 18.

L. ARNAUD – Contribution à l'amélioration de la testabilité et du diagnostic de systèmes complexes : Application aux systèmes avioniques. 2014.

E. CLEMENT.; A. RAUZY.; T. THOMAS - Arbre Analyste : un outil d'arbres de défaillances respectant le standard Open-PSA et utilisant le moteur XFTA.

Capella - www.polarsys.org/capella/

RAUZY A. (2008). Guarded transition systems: a new states/events formalism for reliability studies

BATTEUX, M.; PROSVIRNOVA,T.; RAUZY,A. (2015). System Structure Modeling Language (S2ML).

VALIANT, L. (1979). The Complexity of Computing the Permanent

CASSANDRAS, C.; LAFORTUNE, S. (2008), Introduction to Discrete Event Systems, 2nd Edition, Springer - Verlag.

ZIMMERMANN, K. (1976). Extrem´aln´ı Algebra. Ekonomick´y `ustav ˇCSAV