ANALYSE DES EXIGENCES DE SURETE D'UN SYSTÈME ELECTRIQUE PAR MODEL-CHECKING*

ANALYSIS OF AN ELECTRICAL SYSTEM SAFETY REQUIREMENTS BY MODEL-CHECKING*

P. Bieber, C. Castel, C. Kehren, C.Seguin
ONERA-CERT
2, av. Edouard-Belin, BP4025
31055 Toulouse cedex
{nom}@cert.fr

Résumé

Dans le cadre du projet européen ESACS (*Enhanced Safety Assessment for Complex Systems)*, nous avons modélisé en Altarica un système de génération et de distribution électrique proche de celui de l'AIRBUS A320. Nous avons formalisé les exigences de sûreté de ce système en logique temporelle et nous avons utilisé le model-checking pour démontrer que le modèle de système électrique satisfait effectivement ses exigences de sûreté.

Summary

Within the European project ESACS (Enhanced Safety Assessment for Complex Systems), we modelled with Altarica an electrical generation and distribution system based on the AIRBUS A320 system. We used temporal logic to formalize safety requirements of this system and we used model-checking to prove that the electrical system model enforces its safety requirements.

Introduction

Les analyses de sûreté de systèmes complexes peuvent être réalisées à l'aide de nombreux outils efficaces. Pour l'analyse de modèles statiques dans un contexte industriel l'approche par arbres de défaillances est la méthode de prédilection qui permet d'obtenir des résultats à la fois qualitatifs et quantitatifs. L'aspect qualitatif est particulièrement utile pour l'analyse préliminaire lorsque l'on s'intéresse à savoir si une combinaison de pannes simple, double ou triple cause un événement redouté. Cette analyse doit être confirmée par des résultats quantitatifs comme la probabilité d'occurrence d'un tel événement.

Lorsque l'on souhaite étudier des modèles dynamiques, l'approche fondée sur les arbres de défaillances n'est plus forcément adaptée car dans de tels modèles la notion de séquence d'événements apparaît. Ainsi, l'occurrence de l'événement redouté ne dépend pas uniquement de combinaisons d'événements mais également de l'ordre dans lequel ils se réalisent. On atteint alors les limites de la logique propositionnelle employée pour décrire l'arbre de défaillances classique car elle ne se prête pas à la formalisation de cette notion de séquence. On peut néanmoins s'intéresser aux aspects quantitatifs avec l'utilisation des méthodes stochastiques (e.g. chaînes de Markov, simulation de Monte-Carlo).

Notre objectif est de réconcilier les modèles dynamiques avec les analyses qualitatives en utilisant un outil de model-cheking pour vérifier la tenue d'exigences de sûreté.

Cette étude a été réalisée dans le cadre du projet européen ESACS (*Enhanced Safety Assessment for Complex Systems)* et appliquée à un système de génération et de distribution électrique Airbus qui est décrit dans le chapitre suivant. L'étude s'est déroulée en trois étapes principales que nous décrivons dans la suite de l'article: production d'un modèle du système électrique, production d'un modèle des exigences à satisfaire, vérification des exigences. Nous terminons l'article en comparant notre approche avec des travaux se fondant sur d'autres outils de vérifications que le model-checking.

Système de génération et distribution électrique

Le rôle du système électrique est fondamental dans un aéronef puisqu'il alimente de nombreux éléments critiques (e.g. servocommandes de gouvernes, calculateurs embarqués, etc.). Comme la perte d'énergie électrique peut aboutir à la perte du contrôle de l'avion, ce système doit respecter l'exigence de sûreté suivante : la perte totale d'énergie électrique est considérée catastrophique. Le taux d'occurrence de cette condition de défaillance doit être inférieur à 10^{-9} par heure de vol et une seule défaillance ne doit pas aboutir à cette condition de défaillance.

Pour respecter cette exigence, un système électrique d'un avion de type A320 est organisé en deux sous-systèmes: le système normal et le système de secours. Chacun des sous-systèmes comprend des générateurs, des barres d'alimentation, des contacteurs, des disjoncteurs, des transformateurs et des jonctions. Le système normal est composé de 2 générateurs principaux GEN1 entraîné par le réacteur 1, GEN 2 entraîné par le réacteur 2 et une unité de puissance auxiliaire APU. Le système de secours est composé d'un générateur de secours CSM_G alimenté par le système hydraulique et la RAT (Ram Air Turbine) qui est déployée automatiquement en cas de perte des générateurs principaux. La puissance électrique est fournie aux charges électriques par le biais des quatre barres de distribution pour le système nominal: ACside1, DCside1, ACside2, DCside2 et par le biais de deux barres de distribution essentielles dans le système de secours : ACess et DCess. La conversion du courant alternatif vers le courant continu est réalisée par les transformateurs TR1, TR2 et TRess. Le système comprend également des disjoncteurs de façon à limiter la propagation de courts-circuits.

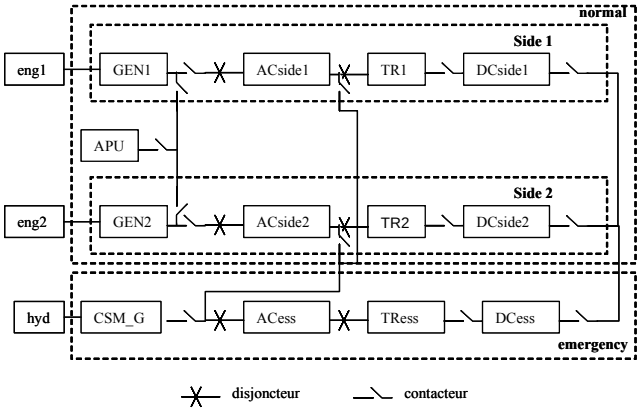

Figure 1 − Architecture du système électrique

Les contacteurs sont contrôlés de façon à implanter plusieurs types de reconfigurations. Par exemple, tous les générateurs du système normal (GEN1, GEN2 or APU) peuvent être utilisées pour fournir de l'électricité à toutes les barres de distribution des

* Publié dans les actes du congrès Lambda-Mu 14, Colloque de Maîtrise des Risques et Sûreté de Fonctionnement 2004

systèmes normal ou de secours lorsque un ou deux générateurs ne sont pas utilisables. D'autres règles de reconfiguration s'appliquent lorsque un des transformateurs est perdu. Finalement, lorsque tous les générateurs du système normal sont perdus, le générateur de secours fournir de l'électricité exclusivement aux barres de distribution essentielles (ACess et DCess).

Modélisation du système électrique

Le langage AltaRica

Le modèle formel du circuit électrique a été réalisé à l'aide du langage AltaRica [1] créé par le Laboratoire Bordelais de Recherche en Informatique (LaBRI) qui permet de décrire à la fois le comportement d'un système dans le cas nominal et présence de défaillances. Nous l'avons retenu car c'est un langage formel simple, à la fois hiérarchique et compositionnel. Sa sémantique et sa syntaxe clairement définies lui permettent d'être couplé à différents outils de fiabilité et de validation comme Aralia [9], Moca-RP ou encore de model-checking comme Mec V [8] ou SMV.
Le développement de modèles AltaRica est supporté par Cecilia OCAS workshop de Dassault Aviation qui fournit un éditeur graphique de modèles et un gestionnaire de composants. De plus, cet outil intègre des fonctions de simulation interactive et de génération d'arbre de défaillance. Nous présentons maintenant les constituants du langage AltaRica qui nous ont été utiles pour modéliser le système électrique.

Chaque composant est décrit par un nœud (`node`) qui comporte trois parties : la déclaration des variables et des événements, la définition des transitions et la définition des assertions.

Chaque composant possède une variable d'état (`state`), dont la valeur dénote le mode de fonctionnement ou de dysfonctionnement du composant. Les composants possèdent également un nombre fini de variables de flux (`flow`) qui font le lien entre le composant et son environnement. Dans les modèles que nous avons réalisés, ces variables sont soit booléennes soit elles prennent leurs valeurs dans un petit ensemble énuméré de constantes. Les composants possèdent également des événements (`event`) qui permettent de faire évoluer la valeur des variables d'état. Les événements sont utiles pour modéliser l'occurrence de défaillances ou la réaction à des conditions sur les variables de flux (nous appelons ce dernier événement `update`). Cet événement particulier joue un rôle très important dans la modélisation de l'impact sur le système des défaillances en cascade.

La rubrique `trans` regroupe les définitions de transitions sous la forme: `g |- evt -> e`. La garde `g` est une formule booléenne qui peut contenir la variable d'état ainsi que des variables de flux. La garde définit la configuration dans laquelle la transition peut être réalisée si l'événement `evt` se produit. L'effet `e` modifie la valeur de la variable d'états si la garde est vérifiée et si l'événement se produit.

Les relations de dépendance entre l'état du composant et la valeur de ses flux sont déclarés dans les assertions (`assert`). Dans les modèles que nous avons réalisés, les assertions décrivent les règles de calcul de la valeur des flux de sortie en fonction de l'état du composant et de la valeur des flux d'entrée.

Ces concepts sont illustrés par l'exemple suivant. Le composant `basic_wire` a une variable d'entrée nommée `a_V_e` qui indique la présence de tension en entrée sur la borne a du composant et une variable de sortie nommée `b_V_s` qui indique la présence de tension en sortie sur la borne b. La variable `status` indique l'état du composant, sa valeur est égale à `ok` lorsque aucune défaillance ne s'est produite et à `lost` lorsque le composant ne peut plus transmettre de tension. L'événement `fail_loss` décrit une défaillance qui fait passer le composant dans l'état `lost`. La transition associée à cet événement ne peut se déclencher que si le composant est dans l'état `ok`. L'assertion signifie que la valeur de la tension en sortie est égale à la valeur de la tension en entrée si le composant est dans l'état `ok` et sinon (l'état du composant est `lost`) alors il n'y a pas de tension en sortie.

```
node basic_wire
 state
  status : {ok,lost};
 flow
  a_V_e : bool : in;
  b_V_s : bool : out;
 event
  fail_loss;
 trans
  (status=ok) |- fail_loss -> status:= lost;
 assert
    b_V_s = case {(status=ok) : a_V_e,
                  else : false};
 init
    status := ok ;
edon
```

Dans un modèle de système, les instances de nœuds AltaRica sont interconnectées par des assertions qui relient les flux d'entrée d'un composant avec les flux de sortie d'autres composants.

Modélisation AltaRica des composants électriques

Nous avons tout d'abord développé avec l'aide d'AIRBUS une bibliothèque de composants électriques : générateurs, disjoncteurs, transformateurs, contacteurs, barre de distribution et charge électrique. Pour chaque composant, nous avons modélisé l'effet des modes de défaillance sur son comportement interne ainsi que la propagation des défaillances vers son environnement.

Nous nous sommes intéressés à des modes de défaillance pouvant aboutir à la perte totale ou partielle de puissance électrique. Nous supposons que tous les composants peuvent ne plus être capable de générer, transmettre ou fournir de l'électricité. Nous avons aussi supposé que des courts-circuits peuvent survenir dans les barres d'alimentation. Finalement, les contacteurs et les disjoncteurs peuvent rester bloqués en position ouverte ou fermée. La table 1, résume les modes de défaillance associés aux composants de notre bibliothèque.

Composant	***Modes de défaillance***
contacteur	*perte, bloqué*
disjoncteur	*perte, bloqué*
transformateur	*perte, court-circuit*
charge électrique	*perte, court-circuit*
barre de distribution	*perte, court-circuit*
générateur	*perte, court-circuit*

Table 1 − Modes de défaillance considérés

Nous utilisons les flux des composants du système électrique pour modéliser la propagation des défaillances au sein du système électrique. Il faut donc que les flux rendent compte de la présence ou de l'absence de tension, il faut également qu'ils puissent formaliser la propagation des courts-circuits.

Etant donné l'architecture du système électrique étudié, il n'est pas possible de supposer que la propagation de la tension entre les bornes d'un composant aura toujours le même sens. En effet, en fonction des configurations des contacteurs, une borne ou l'autre imposera la tension au composant. Nous avons donc considéré que chaque borne des composants était susceptible d'avoir une tension en entrée et une tension en sortie. En revanche, la présence simultanée de tension en entrée sur plusieurs bornes d'un même composant est assimilée à un court-circuit du composant.

En ce qui concerne, les courts circuits nous avons ajouté des flux qui leurs sont dédiés à tous les composants de la bibliothèque, même à ceux auxquels ce mode de défaillance n'a pas été associé (comme les contacteurs et les disjoncteurs). Comme pour la tension, chaque borne du composant peut avoir un signal de court-circuit en entrée et en sortie. Ceci nous permet de propager le court-circuit dans l'ensemble du système électrique.

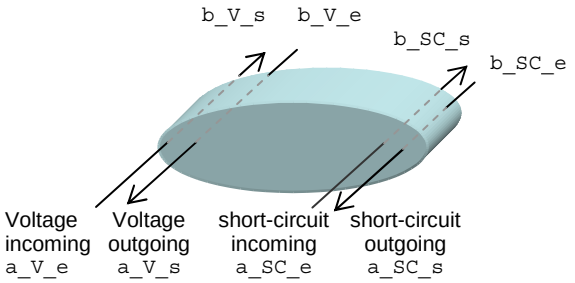

Figure 2 − Flux associés à une barre de distribution

L'exemple suivant montre en détail comment nous modélisons la propagation des défaillances pour une barre de distribution. La barre a deux bornes : a et b, à chaque borne les quatre flux sont définis.

```
node bus_bar
 flow
  a_SC_e : bool : in; // port a incoming SC
  a_SC_s : bool : out; // port a outgoing SC
  a_V_e : bool : in; // port a incoming V
  a_V_s : bool : out; // port a outgoing V
  b_SC_e : bool : in; // port b incoming SC
  b_SC_s : bool : out; // port b outgoing SC
  b_V_e : bool : in; // port b incoming V
  b_V_s : bool : out; // port b outgoing V
state
  status : {ok,sc,lost};
 event
   fail_loss,fail_SC,update;
 trans
    status = ok) |- fail_loss -> status := lost;
   (status = ok) |- fail_SC -> status := sc;
   (status = ok) and ((a_V_e and b_SC_e) or
   (b_V_e and a_SC_e) or (a_V_e and b_V_e))
                 |- update -> status := sc;
 assert
   a_V_s = case((status = ok): b_V_e,
                else : false);
   b_V_s = case((status = ok): a_V_e
                else : false);
   a_SC_s = case((status = sc):  a_V_e,
                 (status= lost) : false,
                (status= ok): b_SC_e and a_V_e);
   b_SC_s = case((status = sc):  b_V_e,
                 (status= lost) : false,
                 (status= ok): a_SC_e and b_V_e);
 init
   status := ok;
edon
```

La barre de distribution peut être dans l'état nominal (`status=ok`) dans l'état perdu (`status=lost`) ou dans l'état court-circuit (`status=sc`). Les deux premières transitions décrivent le fait que la barre passe de l'état nominal vers les états perdus ou court-circuit lors de l'occurrence d'un événement de défaillance (`fail_loss` ou `fail_sc`). La dernière transition décrit trois cas pour lesquels le composant passe aussi dans l'état court-circuit. Ces cas correspondent à la propagation d'une défaillance dans l'environnement de la barre de distribution. Si la barre est dans l'état nominal et si sur une borne une tension en entrée est présente et sur la borne opposée un court-circuit en entrée est présent alors après occurrence de l'événement de mise à jour `update` le composant passe dans l'état court-circuit. De même, si la barre est dans l'état nominal et si une tension en entrée est simultanément présente sur les deux bornes de la barre alors le composant passe dans l'état court-circuit.

Les assertions associées aux signaux de présence de tension en sortie sur la borne a (`a_V_s`) et sur la borne b (`b_V_s`) modélisent la règle suivante : quand la barre est dans un état nominal elle propage la tension d'une borne vers l'autre, dans tous les autres modes nous supposons que la tension n'est pas propagée. Les assertions associées aux signaux de présence de court-circuit en sortie sur la borne a (`a_SC_s`) et sur la borne b (`b_SC_s`) modélisent la règle suivante : Quand la barre est dans l'état court-circuit, elle envoie un signal de court-circuit vers une borne si une tension en entrée est présente sur cette borne, quand la barre est perdue elle ne propage pas les courts-circuits et dans le mode nominal elle propage le signal de court-circuit reçu sur une borne vers l'autre borne si une tension en entrée est présente sur cette dernière.

Modélisation AltaRica du système électrique

Grâce aux capacités de l'éditeur graphique de modèles de l'environnement Cecilia OCAS, le modèle de l'architecture a été obtenu rapidement en interconnectant les composants puisés dans la bibliothèque. Le système électrique que nous avons modélisé utilise 20 classes de composants, il contient environ 70 instances de composants.

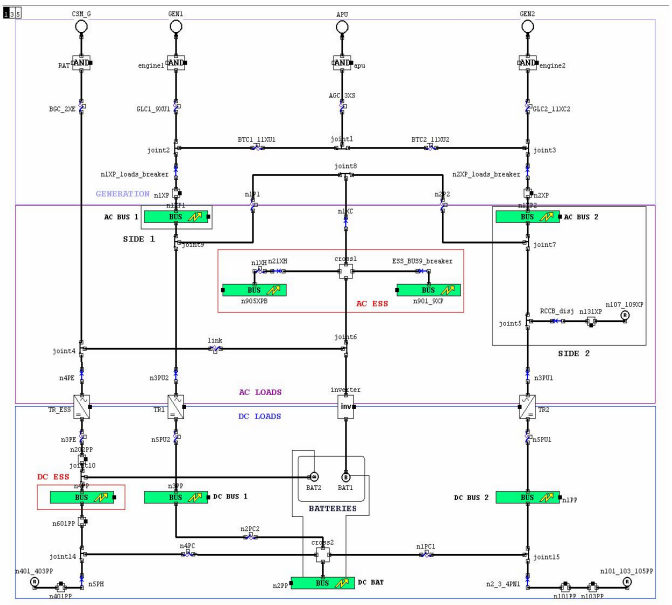

Figure 3 - Vue graphique du modèle de système électrique

Le système comporte 15 contacteurs qui permettent un grand nombre de reconfigurations possibles. L'architecture du système électrique nous a été proposée par AIRBUS sans la description du contrôle des contacteurs. Nous avons donc développé une composant contrôleur qui délivre un ordre d'ouverture ou de fermeture à chacun des contacteurs du modèle en fonction de l'observation de l'état courant des générateurs et des transformateurs.

Nous avons validé le modèle avec AIRBUS en utilisant le simulateur graphique de Cecilia OCAS. Nous avons déclenché des événements de défaillances dans un ou plusieurs composants, puis nous avons observé la propagation des défaillances dans l'ensemble du système électrique.

Le modèle ainsi créé comporte des circularités de calcul sur les flux de données dues à des boucles physiques dans l'architecture du système électrique (i.e. des chemins tels que l'entrée d'un composant dépend de sa sortie). Nous avons cherché à limiter ces circularités car bien que le simulateur OCAS soit capable de traiter des modèles avec circularité, les techniques de vérification d'exigences (model-checker SMV ou SCADE, générateur automatique d'arbre de défaillance associés à AltaRica) ne sont pas capables de traiter ce type de modèles.

Pour parvenir à casser ces boucles, nous avons introduit des délais de propagation pour les signaux de tension et de court-circuit. Par exemple, reprenons le modèle `bus_bar` présenté plus haut. En mode nominal le court-circuit se propage instantanément entre les bornes.

```
   a_SC_s = case((status = sc):  a_V_e,
                 else : false);
   b_SC_s = case((status = sc):  b_V_e,
                 else : false);
```

Si l'on supprime le cas du mode `ok` dans les assertions associées aux flux `a_SC_s` et `b_SC_s` alors le court-circuit ne se propage plus instantanément. En effet, si un court-circuit est présent sur une des bornes, ce n'est qu'après un événement `update` que la barre passe dans l'état `sc`, et que le court-circuit peut se propager sur l'autre borne.

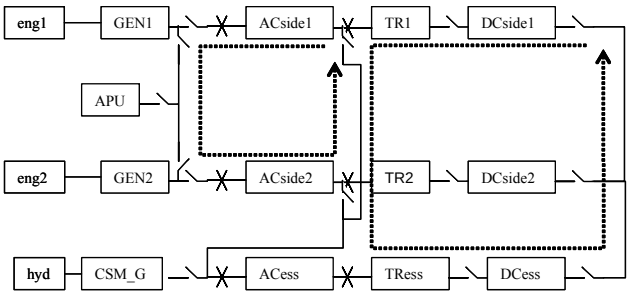

Figure 4 − 2 cycles dans le système électrique

Pour éliminer toutes les définitions circulaires il n'est pas nécessaire de modifier tous les composants. Nous nous sommes contentés d'introduire des délais de propagation pour les signaux de tension et de court-circuit au niveau des composants de jonction présents dans toutes les boucles du modèle. Le modèle modifié est de type flot de données donc il peut être traité par les outils de model-checking pour SMV ou SCADE mais la propagation des signaux électriques n'est pas instantanée.

Modélisation des exigences

Exigences qualitatives

Comme nous l'avons déjà mentionné dans la présentation du système électrique, la principale exigence de sûreté est : la perte totale d'énergie électrique (c'est-à-dire la perte de trois barres de distribution) est considérée catastrophique. Nous nous sommes intéressés à deux autres exigences du même type: la perte de deux barres de distribution est considérée majeure et la perte d'une barre de distribution est considérée mineure.

Nous associons à chacune de ces exigences une exigence qualitative qui est de la forme "*s'il y a moins de N défaillances individuelles alors la perte de N+1 barres de distributions ne doit pas arriver*" avec N = 0,1,2. Ce type d'exigences qualitatives permet de détecter de grosses anomalies (comme une panne simple qui causerait la perte de deux ou trois barres de distribution) en phase préliminaire de conception d'une architecture. Puis, lorsque les taux de défaillances des constituants du modèle sont connus, l'analyse de l'architecture peut être affinée en évaluant la tenue des exigences quantitatives.

Pour modéliser aisément ces exigences nous avons introduit dans le modèle des nœuds AltaRica baptisés observateurs dont l'objet est uniquement d'observer la tension en sortie des différentes barres de distribution et de détecter si une, deux ou trois barres ne fournissent pas de tension. L'autre composant utile pour décrire les exigences est le compteur de défaillances qui incrémente le compteur `FailC` dès lors qu'un événement de défaillance a lieu.

Exigences temporelles

Pour modéliser les exigences qualitatives, l'utilisation des observateurs qui détectent la perte des barres dans un seul état n'est pas suffisante. Considérons la perte des barres ACside1, ACside2, et ACess. L'observateur associé à cette condition de défaillance possède un flux de sortie `ACside1_side2_ess` dont la valeur est égale à faux dès qu'il n'y a plus de tension en sortie sur ces trois barres. Il est tout à fait possible que, lorsque plusieurs des générateurs sont perdus, ce flux soit égal à faux durant une courte période puis redevienne vrai lorsqu'une reconfiguration comme la mise en marche du générateur de secours s'est terminée. Une description correcte de la perte de puissance électrique devrait modéliser le fait qu'il n'est pas possible de récupérer de la puissance électrique.

Aussi avons nous utilisé les opérateurs de Logique Temporelle Linéaire [2] pour modéliser des conditions de défaillance dynamiques comme la perte permanente ou la perte bornée dans le temps d'électricité. La formule temporelle `F G ACside1_side2_ess=false` décrit la perte permanente d'électricité. `F` est l'opérateur Futur tel que la formule `F p` est vraie s'il existe un instant dans le futur où la formule `p` est vraie et `G` est l'opérateur Toujours (ou Globalement) tel que la formule `G p` est vraie si la formule `p` est vraie dans tous les instants futurs. Le formule précédente peut être lue "*Il existe un instant dans le futur à partir duquel la puissance électrique sera totalement perdue pour toujours*".

Finalement, pour vérifier l'exigence "*s'il y a moins de 2 défaillances individuelles alors la perte de 3 barres de distributions ne doit pas arriver*" nous chercherons à montrer que le modèle du système électrique satisfait les deux formules suivantes :

```
G (FailC ≤ 2) -> ~ F G ACside1_side2_ess=false
G (FailC ≤ 2) -> ~ F G DCside1_side2_ess=false
```

Vérification des exigences

La dernière étape de notre étude concerne la vérification des exigences.

Génération d'arbre de défaillance

Une première approche pour vérifier les exigences de sûreté de fonctionnement consiste à générer un arbre de défaillance à partir du modèle AltaRica puis à utiliser un outil comme ARALIA [?] pour calculer les coupes ou évaluer les probabilités d'occurrence des conditions de défaillance. Cette approche est suivie par Dassault Aviation.

L'arbre généré est une formule booléenne dont les atomes sont les noms d'événements du modèle AltaRica et qui décrit des chemins qui mènent le système de l'état initial à un état qui satisfait la condition de défaillance. Comme un chemin est décrit par une conjonction d'événements et que l'opérateur conjonction est commutatif, il est important que l'ordre d'apparition des événements dans un chemin n'ait pas d'importance. Or ceci n'est pas vérifié dans le modèle électrique que nous avons réalisé. En effet, l'état résultant de la séquence composée d'un court-circuit dans une barre d'alimentation puis de plusieurs `update` n'est pas le même que celui qui résulte de la séquence composée de plusieurs `update` puis du court-circuit. Dans la première séquence, le court-circuit se propage et peut détruire tous les composants qui ne sont pas protégés par un disjoncteur alors que dans la seconde séquence le court-circuit ne se propage pas et certains composants non protégés peuvent encore fonctionner.

Nous n'avons donc pas pu utiliser les outils de génération d'arbre de défaillance pour étudier notre modèle de système électrique.

Model-checking

Pour vérifier les exigences nous avons utilisé le model-checker *Symbolic Modèle Verifier* (SMV) [4]. Nous avons développé un traducteur qui produit un modèle SMV à partir d'un modèle AltaRica.

Le model-checking permet de vérifier qu'une formule de la logique temporelle linéaire est vraie pour un modèle donné. Un model-checker tel que SMV construit une formule qui décrit l'ensemble des états initiaux, puis il construit la formule qui décrit l'ensemble des états qui peuvent être atteints après occurrence d'un seul événement à partir d'un des états initiaux et SMV continue en construisant les formules qui décrivent les états atteignables en jouant plus d'événements et ceci jusqu'à rencontrer une formule équivalente à la formule précédente ce qui signifie que l'on n'atteindra plus de nouveaux états en jouant plus d'événements. Pour vérifier qu'une exigence de la forme `G p` est vraie, il suffit de tester que `p` est une conséquence logique de toutes les formules d'états atteignables. La construction des formules d'états atteignables ainsi que les tests de conséquence logique peuvent

être réalisés de façon très efficace en utilisant la structure de données des BDD (Binary Decision Diagrams).

Si le model-checker construit une formule d'état atteignable qui invalide la formule à satisfaire alors un contre-exemple est généré. Ce contre-exemple peut être interprété comme une séquence d'événements qui viole l'exigence. On peut alors jouer la séquence d'événements avec le simulateur OCAS afin de localiser les corrections éventuelles à apporter à l'architecture.

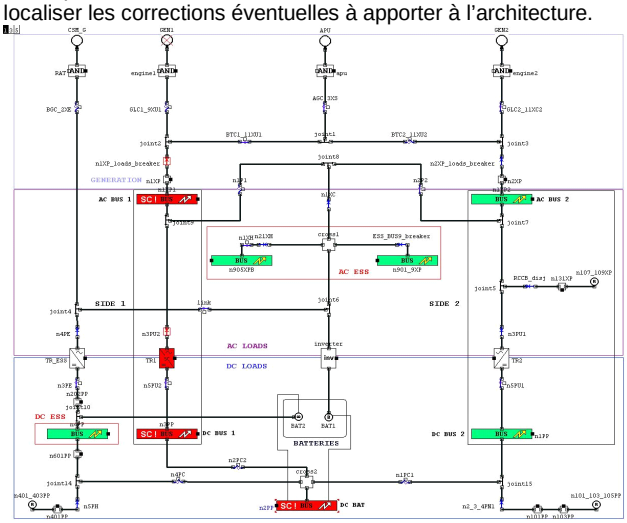

Figure : Analyse d'un contre-exemple avec le simulateur OCAS

Dans un premier temps, la vérification des exigences a échoué, ceci était lié aux déficiences du composant contrôleur des contacteurs que nous avions ajouté à l'architecture de système électrique. Dans ces cas, SMV a fourni des séquences d'événements comprenant plusieurs défaillances et des événements `update` et qui mènent dans des états ne satisfaisant pas les exigences. Ces séquences ont ensuite été simulées dans l'environnement OCAS et elles ont permis de corriger les points faibles de notre contrôleur. Finalement, toutes les exigences ont été vérifiées en moins de 10 secondes et ce bien que les formules possèdent jusqu'à une centaine de variables booléennes dans leurs cônes d'influence (i.e. la valeur de vérité de la formule dans un état dépend de la valeur de vérité d'une centaine de variables).

Comparaison avec d'autres approches

A travers l'expérience réalisée, nous présentons des moyens alternatifs et prometteurs de modélisation et d'analyse de la sûreté de fonctionnement des systèmes très dynamiques.

Modélisation

Tout d'abord, nous avons modélisé qualitativement la dynamique de la propagation des pannes et des moyens de recouvrement au sein d'un unique modèle formel AltaRica.

Classiquement, les notions qualitatives sont décrites par un ensemble de modèles statiques (arbres de défaillances ou blocs diagrammes fonctionnels par exemple). Pour prendre en compte les aspects dynamiques le concept d'arbre dynamique a été introduit [6], [7].

Cependant, l'approche suivie vise à compenser une autre limitation des modèles de type « arbres ». En effet, chaque arbre peut être de taille conséquente tout en étant dédié à un unique événement. Si ce type de modèle est bien maîtrisé par les ingénieurs de sûreté de fonctionnement, il est difficile d'en extraire une vue globale du système partageable avec les concepteurs de système. De plus, les modifications d'un ensemble d'arbres (pour prendre en compte une modification d'architecture ou la découverte de nouveaux modes de défaillance) peut s'avérer très lourdes.

Plusieurs propositions de langages ont été introduites pour réaliser des modèles plus globaux de propagations des pannes. Par exemple, la notation FPTN a été définie pour réaliser des modèles globaux dont on peut extraire de multiples arbres de défaillances [11]. Cependant, comme l'objectif est de générer des arbres classiques, les modèles FPTN offrent une vue hiérarchisée statique de l'architecture du système et ne peuvent pas gérer aisément les aspects dynamiques. Un autre langage, Figaro permet de décrire un système de manière hiérarchique à partir de bases de connaissance [10]. Selon la richesse de la base de connaissance utilisée, il sera possible de réaliser un modèle statique du système analysé et de générer des arbres de défaillances à partir de cette vue. Ou bien on effectuera des analyses plus quantitatives sur la vue dynamique d'un système. AltaRica est un langage moins riche mais qui a été formellement défini pour modéliser des architectures à la fois statiques et dynamiques et s'interfacer facilement avec de nombreux outils existants. L'expérience réalisée montre que l'expressivité semble suffisante pour modéliser les architectures dans les phases amont de conception et que le couplage avec un outil de type model-checking est effectif. Enfin dans le projet ESACS [12], des notations formelles comme StateMate ou Scade traditionnellement utilisées pour spécifier les comportements attendus des systèmes informatiques ont aussi permis de réaliser des modèles formels globaux et détaillés des dysfonctionnements d'un système. On notera que les concepts d'AltaRica peuvent être encodés dans ces langages (le traducteur vers SMV en témoigne). Cependant AltaRica offre une surcouche très utile pour modéliser les modes de défaillances, leur évolution au sein d'un automate de modes et l'effet des modes sur la propagation des erreurs.

Evaluation

La manière de conduire l'analyse qualitative diffère aussi de la pratique classique. Usuellement, pour s'assurer qu'un fonctionnement du système tolère n pannes, on calcule les coupes minimales de l'arbre de défaillance qui décrit les causes du dysfonctionnement et l'on teste qu'elles sont au moins d'ordre n+1. Le model-checking offre la même garantie puisque les comportements sont exhaustivement analysés sans chercher à exhiber pour autant les coupes. Et si l'exigence n'est pas satisfaite, un contre-exemple explicatif est produit. De plus, il peut s'agir d'une configuration très rare, combinant dans le temps événements normaux et anormaux, difficile à détecter par simple simulation. Enfin, les exemples traités montrent que la technique est performante sur des cas significatifs.

Cependant, à la différence de l'approche par arbre de défaillances, l'utilisation du model-checking ne nous permet pas d'avoir une vision globale du nombre et de la longueur des séquences qui invalident une exigence. En effet, lorsqu'une propriété n'est pas vérifiée un unique contre exemple (i.e. une séquence d'événements) est fourni par SMV. Pour aller plus loin et valider la démarche, nous souhaiterions comparer les scénarios imaginés par l'analyste de sûreté de fonctionnement avec toutes les séquences d'événements extractibles d'un modèle par les outils de type model-checking. Plusieurs pistes sont aujourd'hui en cours d'étude.

Dans le cadre du projet ESACS, d'autres partenaires (voir [5]) ont utilisé de façon interactive un model-checker pour générer plusieurs contre-exemples. La première étape est similaire à notre approche, on cherche à vérifier une formule `Loss` correspondant à la perte du système électrique. Le model-checker propose un contre-exemple `C1` qui peut être interprété comme une séquence d'événements. Pour trouver de nouveaux contre-exemples, on soumet au model-checker la formule `Loss and not C1`, c'est-à-dire l'on recherche une situation où le système est perdu mais le contre-exemple `C1` n'est pas satisfait. Un nouveau contre-exemple `C2` est calculé par le model-checker et le procédé décrit précédemment peut être répété pour trouver de nouveaux contre-exemples. Un traducteur d'Altarica vers le langage SCADE qui est pris en entrée de cet outil a été réalisé par le LaBri. Cet outil a été utilisé sur un modèle de système hydraulique de plus petite taille que le système électrique décrit dans cet article.

Cecilia OCAS contient un outil de génération de séquences pour AltaRica qui calcule pour une taille fixée l'ensemble des séquences. Ces séquences sont calculées en particuliers pour quantifier la disponibilité des systèmes. Aujourd'hui, des problèmes de performance, conduisent à limiter la taille des

séquences à 2 ou 3 événements. Ceci doit être améliorer car une séquence typique de contre-exemple comprend une ou deux défaillances et jusqu'à une dizaine d'événements `update`. De plus, les séquences générées comprennent beaucoup de séquences superflues (des séquences qui contiennent toutes les permutations possibles d'événements). Finalement, comme ces séquences vont servir pour calculer des disponibilités, elles ne sont pas forcément minimales, elles contiennent des événements qui sont inutiles pour atteindre la situation redoutée.

La plupart des techniques décrites dans cette dernière section n'ont pas atteint le même stade de maturité que les outils de model-checking ou de calcul sur les arbres de défaillance. De plus, sur le plan qualitatif ces approches semblent buter sur la définition de « scénario minimal » conduisant à un état redouté. Le concept mériterait d'être plus finement étudié.

Conclusion

Cette étude a montré comment utiliser des techniques formelles pour décrire un système avion et ses exigences qualitatives et temporelles. Ces descriptions et les outils (model-checking, simulation interactive) sont utiles pour assister l'analyse de la sûreté de fonctionnement. Nous n'avons pas rencontré de problèmes de performances de ces outils.

Nous comptons poursuivre nos recherches sur ces thèmes dans le cadre du projet ISAAC (Improvement of Safety Analysis of Aircraft Complex systèmes) qui a débuté en 2004.

Remerciements

Les travaux décrits dans cet article ont été réalisés dans le cadre du projet ESACS (G4RD-CT-2000-00361) financé par l'Union Européenne.

Références

[1] A. Griffault, S. Lajeunesse, G. Point, A. Rauzy, J.-P. Signoret, P. Thomas - The AltaRica language - International Conference on Safety and Reliability, ESREL'98, Juin 1998

[2] Z. Manna, A. Pnueli – Temporal verification of reactive systems – Springer, 1995

[3] A. Rauzy - Modes automata et their compilation into fault trees - Reliability Engineering and System Safety, 78:1-12, 2002

[4] K.L. McMillan - Symbolic Model Checking - Kluwer Academic Publishers, 1993

[5] O. Akerlund, S. Nadjm-Tehrani, G. Staalmarck - Integration of Formal Methods into System Safety and Reliability Analysis - 17th International System Safety Conference, Août 1999

[6] J. B. Dugan, Z. Tang - Minimal Cut Set/Sequence Generation for Dynamic fault Trees - Annual Reliability and Maintainability Symposium 2004 Proceedings, Los Angeles, USA, Janvier 2004

[7] M. Bouissou - Boolean Logic Driven Markov Processes : a powerful new formalism for specifying and solving very large Markov models - PSAM6, Puerto Rico, 2002

[8] A. Vincent, A. Griffault, G. Point - Vérification formelle pour Altarica - Maitrise des risques et sûreté de fonctionnement, lambda-mu 14, 2004.

[9] Y. Dutuit & A. Rauzy. Exact and Truncated Computations of Prime Implicants of Coherent and non-Coherent Fault Trees within Aralia. Reliability Engineering and System Safety, 58:127-144, 1997

[10] M. BOUISSOU, H. BOUHADANA, M. BANNELIER, N. VILLATTE - Knowledge modelling and reliability processing : presentation of the FIGARO language and associated tools Safecomp'91, Trondheim (Norvège), novembre 1991.

[11] P. Fenelon, J.A. McDermid, M. Nicholson, and D.J. Pumfrey, Towards Integrated Safety Analysis and Design, ACM Computing Reviews, Vol. 2, No. 1, p. 21-32, 1994.

[12] M. Bozzano, A. Villafiorita et al. ESACS: an integrated methodology for design and safety analysis of complex systems. ESREL 2003 European Safety and Reliability Conference, 2003.