Vers la définition d'un kit d'évaluation pour les simulateurs stochastiques

Benjamin Aupetit, Michel Batteux, Antoine Rauzy, Jean-Marc Roussel

VERS LA DEFINITION D'UN KIT D'EVALUATION POUR LES SIMULATEURS STOCHASTIQUES

TOWARDS A DEFINITION OF AN EVALUATION KIT FOR STOCHASTIC SIMULATORS

AUPETIT B.	BATTEUX M.	RAUZY A.	ROUSSEL J.-M.
IRT SystemX & LGI, CentraleSupélec	IRT SystemX	NTNU	LURPA, ENS Cachan
Palaiseau & Châtenay-Malabry	Palaiseau	Trondheim	Cachan

Résumé

Un simulateur stochastique évènementiel est principalement composé de trois mécanismes : le mécanisme de mise à jour du modèle, de l'échéancier, et de calcul des indicateurs. Les modèles simulés font appel à des degrés divers à ces trois mécanismes. Lorsqu'un utilisateur souhaite sélectionner un tel outil afin de l'inclure dans un processus de conception, il doit établir des critères pouvant prendre en compte, en plus de la qualité des résultats obtenus, la performance de l'outil pour des modèles de la taille et du type de ceux qu'il étudiera.

Pour faciliter ce choix, il est proposé un kit d'évaluation, composé de cas-tests de taille paramétrable et sollicitant différentes composantes du simulateur stochastique évènementiel. Ces cas-tests sont issus de divers domaines de la sûreté de fonctionnement ou de la simulation d'automates. Les résultats de leurs simulations sont comparables à l'aide d'autres outils (chaînes de Markov, arbres de défaillances, formules analytiques, …). L'utilisateur peut ainsi évaluer ou estimer, suivant son utilisation prévue, la performance et la qualité des résultats de l'outil de simulation stochastique évènementielle correspondant à la taille et aux caractéristiques de ses modèles propres.

Summary

An event-based stochastic simulator is mainly composed of three mechanisms: the mechanism to update the model, to schedule the transitions, and to compute the indicators. Simulated models use these three mechanisms at various degrees. When an user wants to select a stochastic simulation tool for inclusion in its process, criteria should be established. They can take into account, in addition to the quality of results, the performance of the tool for models of the size and type of those that will be studied.

To facilitate the choice of the tool according to the intended use, an evaluation kit is proposed, composed of test-cases of parameterizable sizes and corresponding to various mechanisms of the stochastic simulator tool. These test-cases are from various areas of safety assessment or automata simulation. The results of their simulations are comparable with other tools (Markov chains, fault trees, analytical formulas ...). The user can evaluate or estimate, according to its intended use, the performance and the quality of the results of the event-based stochastic simulation tool, according to the size and characteristics of its own models.

Introduction

Les études de sûreté de fonctionnement de systèmes complexes se heurtent à plusieurs problèmes. Le premier est la difficulté de la modélisation de ces systèmes. L'utilisation d'outils de bas niveau, comme des arbres de défaillance ou des chaînes de Markov, implique des modèles imposants et difficiles à maintenir. Le développement de langages de haut niveau permet de résoudre ce problème. Ainsi, AltaRica 3.0 est un langage de haut niveau pour la modélisation de systèmes évènementiels, dédié à l'analyse des risques et de la performance de systèmes complexes (Prosvirnova et al., 2013).

Le second problème est l'exploitation des modèles pour en obtenir des indicateurs numériques. Pour les études probabilistes de risques, plusieurs outils existent : par exemple, les arbres de défaillances, ou encore les chaînes de Markov. Parmi eux, la simulation stochastique présente l'avantage d'accepter les modèles qui ne sont pas étudiables par les autres outils (modèles dynamiques, modélisation de comportement fonctionnel, …) (Zio, 2013).

L'évolution des outils permet d'automatiser certaines tâches présentes dans les processus de conception de systèmes. Dans le cadre de la conception de systèmes critiques, il convient de s'assurer que les outils utilisés sont adaptés pour les tâches qu'ils remplacent. Cette démarche est la qualification des outils. Historiquement, elle a été initiée par le domaine aéronautique, notamment pour les outils de génération de code embarqué, et a été standardisée au sein de la norme DO-178/ED-12.

La révision C de la norme DO-178/ED-12 (RTCA, 2012) a requis la création d'un document intitulé « Software Tools Qualification Considerations » et nommé DO-330/ED-215 (RTCA, 2011). Ce document est un guide pour la qualification d'outils logiciels, issu de l'expérience aéronautique, mais indépendant du domaine[1].

Les outils de traitement des modèles de sûreté de fonctionnement basés sur des langages de haut niveau tels qu'AltaRica 3.0 participent aux processus de conception de systèmes critiques, en remplacement d'autres tâches (par exemple, l'écriture d'un arbre de défaillance). Afin de faciliter l'acceptation, par l'industrie et par les autorités de sûreté, de ces outils et de ces langages, il est important d'apporter des garanties, tant du point de vue de l'exactitude des résultats qu'ils fournissent que de leur capacité à « passer à l'échelle industrielle ».

La section suivante montre le besoin et l'utilité d'un kit d'évaluation. La section 3 présente la méthodologie mise en place pour la construction d'un tel kit. La section 4 décrit les différents modèles composants ce kit, et la section 5 discute de l'application de ce kit pour un simulateur stochastique évènementiel.

Utilité d'un kit d'évaluation

Le processus de qualification d'un outil est décrit dans le document DO-330/ED-215. Ce processus décrit les actions de chaque partie prenante, et les exigences suivant le niveau de qualification requis. Parmi ces exigences figure la performance de l'outil.

[1]Si le domaine est indépendant du domaine (aéronautique, ferroviaire, …), il doit toutefois être accompagné d'un document spécifique au domaine et décrivant son applicabilité, les critères de qualification d'outils, et les niveaux de qualification adéquats suivants les critères utiles.

1 Parties prenantes

La norme DO-330/ED-215 indique la présence d'au moins deux parties prenantes lors du processus de qualification d'un outil : l'utilisateur de l'outil, et le développeur de l'outil. L'utilisateur identifie l'outil à utiliser, son impact sur ses processus, son usage dans le cadre de ses processus, et réalise la qualification de l'outil. Il est important de noter que la qualification d'un outil est spécifique à un utilisateur et à un processus. Un outil qualifié ne doit pas être réutilisé dans un processus différent sans être requalifié. La qualification doit être réalisée par l'utilisateur. Le rôle du développeur de l'outil est limité à la fourniture d'informations à l'utilisateur, sur l'outil et son processus de développement.

2 Niveau de qualification

Les outils d'étude de sûreté de fonctionnement sont des outils de vérification : ils permettent de s'assurer de la bonne conception du système. Les données produites par ces outils ne sont pas présentes dans le système final (à l'inverse, par exemple, d'un générateur de code embarqué). La norme DO-178C/ED-12C est (entre autres) le document spécifique au domaine aéronautique précisant les critères de qualification d'outils et les niveaux de qualifications adéquats des outils. D'après ce document, les outils de vérification sont à qualifier suivant le niveau TQL-5 (Tool Qualification Level), ce qui est le niveau le plus bas. Ce niveau est adapté aux outils dont une éventuelle défaillance présente des conséquences faibles.
La principale tâche de qualification à ce niveau est la rédaction d'un TOR (Tool Operational Requirement, un « cahier des charges » opérationnel), suivie de la vérification de l'adéquation entre l'outil et ce TOR. A ce niveau de qualification, rien n'est exigé de la part du développeur de l'outil. Le TQL-5 est donc adapté à des outils commerciaux, qui ont été développés sans interaction directe avec l'utilisateur de l'outil (COTS). Le développeur peut cependant assister l'utilisateur dans cette démarche, en proposant à l'utilisateur une aide à la qualification, via un kit de qualification. Un tel kit peut comprendre par exemple un Developer-TOR, c'est-à-dire un exemple de TOR destiné à faciliter la rédaction du TOR par l'utilisateur, qui pourra y choisir ses exigences, et y ajouter des exigences spécifiques à son utilisation. Ou encore, contenir des données d'entrée, qui permettront de vérifier le bon fonctionnement de l'outil installé dans son environnement opérationnel.

3 Efficacité de l'outil de simulation stochastique

La simulation stochastique consiste à simuler un grand nombre de fois et de façon pseudo-aléatoire l'évolution du système modélisé, à observer pour chacune de ces évolutions la valeur de différents indicateurs, et à faire des statistiques sur les valeurs observées. On obtient ainsi des estimations avec un intervalle de confiance. Pour réduire cet intervalle de confiance, il faut augmenter le nombre de simulations, ce qui augmente d'autant le temps nécessaire pour obtenir des résultats. Un simulateur stochastique doit donc non seulement fournir des résultats corrects, mais aussi être efficace en temps de calcul.
L'efficacité d'un simulateur stochastique n'est pas une mesure absolue, elle dépend des caractéristiques des modèles simulés. C'est donc à l'utilisateur de déterminer ses exigences d'efficacité suivant les caractéristiques qui l'intéressent, en termes d'exactitude des résultats obtenus, de temps de calcul, et de ressources de calcul utilisées.
Afin d'aider l'utilisateur à exprimer et évaluer l'efficacité d'un simulateur stochastique suivant ses exigences, la construction d'un kit d'évaluation est présentée dans la section suivante. Ce kit peut être vu comme un moyen de mesure de la compatibilité de l'outil avec les besoins de l'utilisateur (exprimés dans le TOR), comme un support d'échange d'exigence entre l'utilisateur et le développeur (lors du développement d'un outil), ou bien encore comme un benchmark pour simulateur stochastique de modèle évènementiel.

Méthodologie

Afin d'évaluer les résultats et les performances d'un simulateur stochastique, il est nécessaire de disposer d'un ensemble de modèles faisant intervenir les différents mécanismes du simulateur.

1 Caractéristiques de la simulation stochastique de modèles évènementiels

La simulation stochastique de modèles évènementiels repose essentiellement sur deux mécanismes. Le premier de ces mécanismes consiste à gérer, au sens large, la mise à jour du modèle. Il est appelé après le tir d'une transition, afin de calculer les conséquences de ce tir, et de les appliquer sur le modèle de données interne à l'outil. Le second mécanisme est la gestion de l'échéancier. Son rôle est d'ordonnancer les tirs de transitions, en prenant en compte la légalité des tirs, leurs délais, ou encore leur concurrence. L'échéancier indique ensuite au premier mécanisme quelle transition doit être tirée. Ce dernier lui indique en retour quelles transitions sont tirables ou non, et donc quelles transitions sont à ajouter ou à retirer de l'échéancier.
La performance d'un simulateur stochastique dépend donc directement des performances individuelles de ces deux mécanismes, ainsi que de leur interaction.
La performance du mécanisme de mise à jour du modèle, dépend de plusieurs caractéristiques du modèle simulé :

- Une transition peut avoir un impact très restreint (mise à jour d'une unique variable) ou au contraire très étendu (mise à jour de toutes les variables du modèle) : la différence de temps de calcul entre ces deux extrêmes peut être très significative ;
- Le tir d'une transition peut avoir des répercussions sur un sous-ensemble réduit du modèle (par exemple, uniquement sur le composant qui vient de défaillir), ou bien avoir une propagation importante (reconfiguration d'une majeure partie du modèle) ;
- Les différentes parties du modèle peuvent être indépendantes, ou bien être synchronisées via des variables.

La performance de l'échéancier dépend elle aussi de plusieurs caractéristiques du modèle simulé :

- Le nombre de transitions présentes dans l'échéancier (ce dernier peut varier significativement au cours de la simulation et d'une simulation à l'autre) ;
- Le nombre d'activations et de désactivations de transitions à chaque pas, qui nécessitent des calculs de date de tir, ainsi qu'un ordonnancement des transitions à tirer ;
- Le nombre de transitions synchronisées, dont la légalité des tirs et les actions peuvent dépendre et agir sur différentes parties du modèle.

L'utilité d'un outil de simulation stochastique est d'obtenir des indicateurs statistiques sur les histoires simulées. En plus de ces deux mécanismes, il faut donc considérer le calcul de ces indicateurs dans la performance de l'outil :

- Les indicateurs que l'on souhaite obtenir du modèle peuvent être simple à observer (par exemple, dépendant de la valeur d'une unique variable du modèle), ou bien plus coûteux en calculs (dépendant de la valeur de nombreuses variables et/ou de l'historique de la simulation) ;
- Les modèles peuvent contenir un unique indicateur, ou bien un grand nombre.

2 Choix des modèles

Les observations ci-dessus conduisent à la définition d'un ensemble de cas-tests permettant d'évaluer chacune de ces caractéristiques des simulateurs stochastiques. Ces caractéristiques étant fortement liées, la plupart des modèles recouvrent plusieurs caractéristiques.

Pour vérifier la capacité de l'outil à passer à l'échelle, les modèles sont définis de façon paramétrique, permettant ainsi de faire varier leur taille à la demande. Il est ainsi possible de faire correspondre la sollicitation des différentes caractéristiques à l'utilisation prévue de l'outil, c'est-à-dire à l'ordre de grandeur de taille des modèles que l'outil aura à traiter. Cette démarche permet d'évaluer le temps de calcul nécessaire pour obtenir des résultats de la qualité attendue.

De plus, afin de s'assurer de leur qualité, les résultats obtenus peuvent être comparés aux résultats donnés par d'autres types d'outils d'analyse. Certains modèles peuvent ainsi faire l'objet de calculs analytiques, et d'autres peuvent être étudiés après traduction en chaînes de Markov ou en arbres de défaillance.

Nous présentons donc ici la construction raisonnée d'un ensemble de cas-tests paramétriques permettant de d'évaluer les simulateurs stochastiques. Afin de rendre notre travail utile à la communauté au-delà de l'évaluation du simulateur stochastique AltaRica 3.0, nous avons essayé de rendre ces cas tests indépendants du langage de modélisation et du simulateur stochastique utilisés.

Compte-tenu de l'objectif visé, ces cas tests ne sont donc ni des modèles directement exploitables (nous donnons toutefois leur traduction en AltaRica 3.0), ni non plus des systèmes réels : ce sont des artefacts destinés à évaluer telles ou telles caractéristiques de l'outil testé. Ils sont toutefois inspirés de cas d'études classiques en sûreté de fonctionnement ou en théorie des automates, permettant leur compréhension rapide, et facilitant leur comparaison avec les modèles à étudier.

Modèles

Le kit d'évaluation est composé de différents cas-tests, de provenance diverses : modèles série-parallèle, chaînes de Markov, arbres de défaillances, mécanismes séquentiels, réseaux, ou encore automates cellulaires.

1 Modèles série-parallèles

De nombreuses modélisations dysfonctionnelles de systèmes complexes sont de type série-parallèle. C'est pourquoi un premier ensemble de modèles est basé sur une architecture de ce type. On considère alors des composants, avec une entrée et une sortie : si le composant est en état de fonctionnement, sa sortie est égale à son entrée. Ces composants sont assemblés en parallèle (dont la sortie est égale à l'entrée si il y a au moins un composant en état de fonctionnement). Ces assemblages sont ensuite disposés en série. La sortie de cet ensemble est égale à l'entrée si chaque assemblage a au moins un composant en état de fonctionnement.

Il est possible de faire varier différentes caractéristiques :

- Le nombre de composants en parallèle (P), et le nombre d'assemblages en série (S), permettent de faire varier la taille du modèle ;
- Le typage des données circulant : il est possible de faire circuler des variables de n'importe quel type autorisé par l'outil de simulation stochastique étudié (booléen, entier, réel, texte, …), pour correspondre au plus près à l'utilisation prévue de l'outil ;
- Le comportement des composants.

Ce dernier point permet de décliner, sur une même architecture générique, plusieurs modèles qui vont faire intervenir plus ou moins fortement plusieurs caractéristiques de l'outil de simulation stochastique.

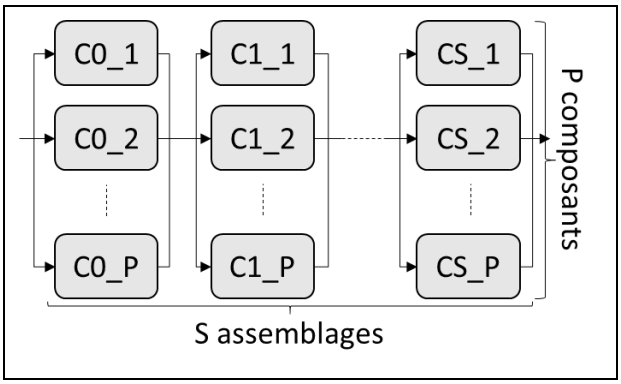

Figure 1. Architecture série-parallèle

1.a Composant avec défaillance simple ou réparable

Les composants avec défaillance simple sont utilisés dans les études de sûreté de fonctionnement pour représenter un composant qui peut défaillir. Leur comportement est descriptible par un automate à deux états et une seule transition : ils sont en état de bon fonctionnement à l'instant 0, et peuvent défaillir suivant un taux de défaillance constant : leur date de défaillance suit donc une loi exponentielle de paramètre λ. En rajoutant une transition de l'état défaillant à l'état de bon fonctionnement, suivant un taux constant μ, on obtient un composant réparable.

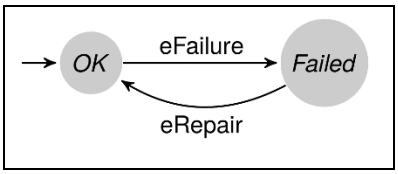

Figure 2. Comportement d'un composant réparable

Un modèle série-parallèle avec des composants de ce type à un taux de fonctionnement asymptotique calculable analytiquement :

$$\text{Taux de fonctionnement}(t \to \infty) = \left(1 - \left(\frac{\lambda}{\lambda+\mu}\right)^{P}\right)^{S} \quad \{1\}$$

Le comportement des composants étant limité, leur simulation ne va pas être coûteuse. L'intérêt de ce type de modèle réside donc dans la simulation de l'architecture série-parallèle, surtout si elle est de taille importante. Le mécanisme de propagation des valeurs des variables sera sollicité : l'entrée de chaque composant dépend de l'état de tous les composants en amont. De plus, l'échéancier doit gérer autant de transitions qu'il y a de composants.

```
class Component
  Boolean vsCondition (init = true);
  parameter Real lambda = 1.0e-7;
  parameter Real mu = 1.0e-4;
  event failure (delay = exponential(lambda));
  event repair (delay = exponential(mu));
  Boolean vfIn (reset = false);
  Boolean vfOut (reset = false);
  transition
    failure: vsCondition == true -> vsCondition := false;
    repair: vsCondition == false -> vsCondition := true;
  assertion
    vfOut := if vsCondition then vfIn else false;
end
```

Figure 3. Modélisation AltaRica 3.0 de composant réparable

1.b Composant avec défaillance et réparateurs limités

La limitation du nombre d'équipe de réparation est utile lorsque l'on s'intéresse à la fiabilité d'un système avec sa politique de maintenance. Les composants avec défaillance réparable et réparateurs limités ont un comportement proche des composants avec défaillance réparable : un nombre prédéfini d'équipe de réparation est disponible, le nombre de composants en cours de réparation est donc limité à ce nombre.

```
class Component
  Boolean vsCondition (init = true);
  parameter Real pLambda = 1.0e-7;
  event eFailure (delay = exponential(pLambda));
  parameter Real pMu = 1.0e-4;
  event eRepair (delay = exponential(pMu));
  Boolean vsRepair (init = false);
  event eStartRepair;
  Boolean vfIn (reset = false);
  Boolean vfOut (reset = false);
  transition
    eFailure: vsCondition == true -> vsCondition := false;
    eStartRepair: vsCondition == false and vsRepair == true
                          -> vsRepair := true;
    eRepair: vsCondition == false and vsRepair == false
                          -> {vsCondition := true;
                              vsRepair := false;}
  assertion
    vfOut := if vsCondition then vfIn else false;
end
class RepairCrew
  Integer vsAR (init = 2);
  event startJob, endJob;
  transition
    startJob: vsAR > 0 -> vsAR := vsAR - 1;
    endJob: true -> vsAR := vsAR + 1;
end
block System
        RepairCrew RC;
        Component C1;
```

```
        event C1_eStartRepair (delay=Dirac(0.0));
        event C1_eEndRepair (delay=exponential(1.0e-4));
        transition
                C1_eStartRepair: !C1.eStartRepair & !RC.startJob;
                C1_eEndRepair: !C1.eRepair & !RC.endJob;
end
```

Figure 6. Modélisation AltaRica 3.0 de composant réparable avec réparateurs limités

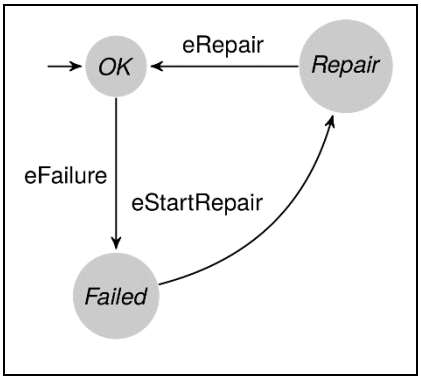

Figure 5. Comportement d'un composant réparable pour synchronisation avec des réparateurs limités

Le taux de fonctionnement de ce modèle est calculable par chaîne de Markov. Une modélisation AltaRica 3.0 possible de ce type de modèle fait appel à la synchronisation des transitions de réparation des composants à des transitions d'utilisation des équipes de réparateur.
En plus de solliciter la propagation des valeurs des variables et d'insérer un grand nombre de transitions dans l'échéancier, à cause de la structure série-parallèle, ce type de modèle utilise fortement la synchronisation gérée par l'échéancier.

1.c Composant avec défaillance réparable et redondance froide

La redondance froide correspond à des composants inutilisés, pour lesquels l'hypothèse qu'ils ne peuvent pas tomber en panne dans cet état est faite. Lorsqu'un composant actif est défaillant, il est remplacé par un composant inutilisé qui devient actif. Son activation peut provoquer une panne (panne à l'allumage). Les composants réparés sont remis dans le stock des composants inutilisés.
Les composants avec défaillance réparable et redondance froide ont donc, en plus du comportement des composants avec défaillance réparable, un état de non fonctionnement, dans lequel ils ne peuvent pas défaillir. Ils suivent la séquence d'état suivante : bon fonctionnement, défaillance, non fonctionnement. La transition de non fonctionnement à bon fonctionnement se fait à la demande. Dans un assemblage en parallèle, un nombre fixé de composants sont en état de bon fonctionnement, les autres sont en non fonctionnement ou défaillants. Lors de la défaillance d'un composant, un autre composant, en état de non fonctionnement, est mis en marche pour le remplacer.
Le taux de fonctionnement de ce modèle est calculable par chaîne de Markov. Une modélisation AltaRica 3.0 possible de ce type de modèle fait appel à la synchronisation des composants par variables de flux : le calcul du nombre de composants en marche permet de commander le démarrage de composants en veille, via une variable de flux.
Le remplacement des composants défaillants par ceux inutilisés implique une synchronisation des différents composants, sollicitant la mise à jour du modèle. La propagation des valeurs des variables et le grand nombre de transitions sont aussi des caractéristiques de ce type de modèle, liées à l'architecture série-parallèle.

2 Modèles chaînes de Markov

De nombreuses modélisations de sûreté de fonctionnement utilisent les chaînes de Markov. Les simulateurs stochastiques évènementiels ont la possibilité de simuler des modèles correspondant à des chaînes de Markov.
Les résultats obtenus par simulation stochastique sur la fiabilité de ces systèmes est à comparer avec une étude par un outil de chaînes de Markov. La traduction d'une chaîne de Markov en AltaRica 3.0 peut être réalisée à l'aide d'une variable d'état entière, contenant le numéro de l'état courant, et de transitions gardées par cet état courant et le modifiant vers le nouvel état.
La simulation stochastique de ce type de modèle sollicite l'échéancier, qui doit gérer un grand nombre de transitions, mais aussi un grand nombre d'activations et de désactivations de transitions, puisqu'à chaque évènement, toutes les transitions actives sont désactivées. L'observation du fonctionnement est également complexe, puisqu'il faut lister l'ensemble des états de la chaîne de Markov dans lesquels le système est en marche (ou en panne).

```
block MarkovChain
  Integer vsID (init = 0);
  observer Boolean oWorking = vsID == 0 or vsID == 1 or [...]
  event e0to1 (delay = exponential(1.0e-4));
  event e1to2 (delay = exponential(1.0e-4));
  event e1to0 (delay = exponential(1.0e-2));
  [...]
  transition
        e0to1: vsID == 0 -> vsID := 1;
        e1to2: vsID == 1 -> vsID := 2;
        e1to0: vsID == 1 -> vsID := 0;
        [...]
end
```

Figure 6. Modélisation AltaRica 3.0 partielle d'une chaîne de Markov

2.a Réparateurs limités et redondance froide

Un ensemble de composants avec redondance froide (avec panne au démarrage) et réparateurs limités est facilement modélisable avec un langage de haut niveau, mais aussi à l'aide d'une chaîne de Markov. On s'intéresse ici à la modélisation via une chaîne de Markov d'un tel système.

Ce cas-test est constitué de n composants avec défaillance réparable et état de non fonctionnement. On souhaite qu'au moins k composants soient en état de bon fonctionnement. Il y a un nombre R d'équipes de réparation de composants. Il y a donc au plus k composants en marche et R composants en réparation, les autres étant en état de non fonctionnement ou défaillants. Dans chaque état, il y a donc une transition de défaillance d'un des composants en marche, ce qui engendre son remplacement par un composant en veille si il y en a un de disponible (et donc son éventuelle défaillance), et le début de sa réparation si un réparateur est disponible.

On s'intéresse au taux de fonctionnement, c'est-à-dire à la proportion de temps où au moins k composants sont en état de bon fonctionnement. La valeur de ce taux de fonctionnement est calculable à l'aide d'un outil de calcul de chaînes de Markov.

2.b Composants avec panne cachée

La détection des pannes n'étant pas forcément immédiate, il peut être utile de modéliser des tests périodiques de composants. La modélisation de ces tests périodiques est possible à l'aide de chaînes de Markov multi-phases.

On considère des composants qui peuvent défaillir selon deux modes exclusifs : soit de façon visible, soit de façon cachée. Le composant en défaillance visible est réparé suivant un taux constant. Le test périodique fait passer les composants de l'état de défaillance cachée à l'état de défaillance visible.

3 Modèles arbre de défaillance

Les arbres de défaillance sont utilisés notamment pour l'étude de sûreté de fonctionnement de systèmes critiques. Ils sont utilisés pour obtenir leurs coupes minimales, c'est-à-dire les listes d'évènements les plus courtes menant à l'évènement redouté, mais aussi pour obtenir la probabilité d'occurrence de l'évènement redouté. Avec des modèles importants, ce calcul peut être coûteux, et la simulation stochastique peut permettre d'obtenir des résultats.

Les arbres de défaillance sont composés d'évènements de base, associés par des portes logiques pour composer l'évènement redouté. Le tir d'un évènement de base fait donc se propager l'information jusqu'à l'évènement redouté.

Afin de traduire un arbre de défaillance en AltaRica 3.0, les évènements de base sont modélisés par des composants avec une variable d'état booléenne initialement fausse. Une transition modifie cette variable d'état à vrai suivant un évènement correspondant à la loi de l'évènement de base correspondant. Les différentes variables d'état sont ensuite assemblées, via des assertions correspondant aux portes logiques, pour composer l'évènement redouté. L'observation porte sur cet évènement redouté.

Lors de la simulation stochastique d'un arbre de défaillance, l'échéancier doit gérer un nombre important de transitions, chaque évènement de base en générant une. L'information d'occurrence des évènements de base parcourt l'arbre et les portes logiques, le mécanisme de mise à jour du modèle doit donc propager cette information dans une partie importante du modèle.

3.a Arbre de défaillance à 3 niveaux

Les arbres de défaillance à 3 niveaux sont typiques de ce que l'on peut s'attendre à obtenir comme structure d'arbre de défaillance, et ont pour propriété de générer un diagramme de décision binaire de taille exponentielle (Nikolskaia et al., 1998). Ces arbres sont formés d'un premier niveau d'évènements, assemblés par un ET pour former l'évènement redouté. Les évènements du second niveau sont assemblés par un OU pour former les évènements du premier niveau. Enfin, les évènements de base sont rassemblés, deux par deux, pour former les évènements du second niveau à l'aide de ET. Dans chaque paire d'évènement de base, le premier évènement est identique à celui dans la même position dans la première branche, tandis que le second est indépendant.

```
class BasicEvent
  Boolean vsFired (init = false);
  parameter Real pLambda = 1.0e-4;
  event eFire (delay = exponential(pLambda));
  Boolean vfOut (reset = false);
  transition
    eFire: not vsFired -> vsFired := true;
end
block FaultTree
        BasicEvent e_1_1_1, e_1_2_1, e_1_3_1;
        BasicEvent e_1_1_2, e_1_2_2, e_1_3_2;
        BasicEvent e_2_1_2, e_2_2_2, e_2_3_2;
        Boolean h_1_1, h_1_2, h_1_3 (reset = false);
```

```
        Boolean h_2_1, h_2_2, h_2_3 (reset = false);
        Boolean g_1, g_2 (reset = false);
        Boolean f (reset = false);
        observer Boolean TopEvent = f;
        assertion
          h_1_1 := e1_1_1.vsFired and e_1_1_2.vsFired;
          h_1_2 := e1_2_1.vsFired and e_1_2_2.vsFired;
          h_1_3 := e1_3_1.vsFired and e_1_3_2.vsFired;
          h_2_1 := e1_1_1.vsFired and e_2_1_2.vsFired;
          h_2_2 := e1_2_1.vsFired and e_2_2_2.vsFired;
          h_2_3 := e1_3_1.vsFired and e_2_3_2.vsFired;
          g_1 := h_1_1 or h_1_2 or h_1_3;
          g_2 := h_2_1 or h_2_2 or h_2_3;
          f := g_1 and g_2;
end
```

Figure 7. Modélisation AltaRica 3.0 d'un arbre de défaillance à 3 niveaux

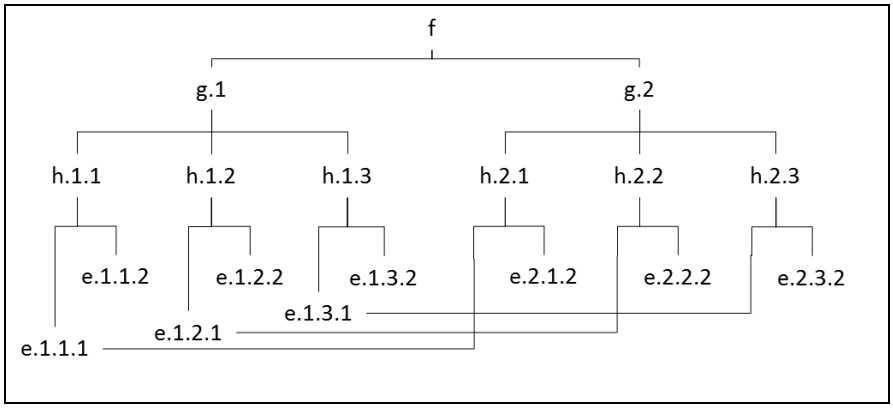

Figure 8. Arbre de défaillance à 3 niveaux

4 Round-Robin

Certains sous-systèmes sont utilisés au cours d'un nombre limité de phases du système. Leurs composants doivent donc être synchronisés avec la phase actuelle.

4.a Round-Robin

Le cas-test Round-Robin représente un système utilisé par parties, de façon séquentielle. Les différentes lignes de composants sont activées et désactivées de façon tournante et régulière, de telle sorte qu'il y ait toujours une ou deux lignes actives. Les autres lignes sont alors inactives. Les composants actifs peuvent défaillir suivant un taux constant, et sont réparés de façon systématique lorsqu'ils sont inactifs.

Ce type de modèle sollicite l'échéancier, en raison du grand nombre d'activations et désactivations de transitions (les défaillances ne sont possibles que lorsque le composant est actif), et la mise à jour du modèle en raison de la synchronisation des composants des différentes lignes lors des activations et désactivations.

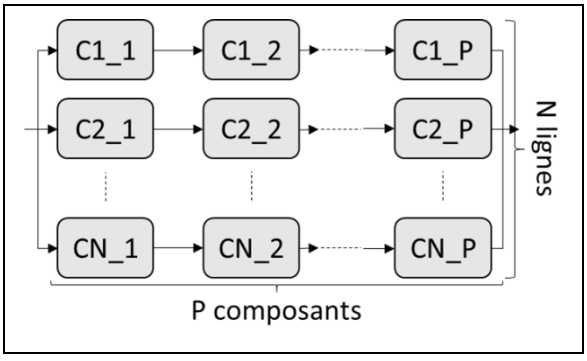

Figure 9. Architecture Round Robin

5 Réseaux

Les structures sous forme de réseaux sont présentes dans de nombreux systèmes : par exemple, distribution d'énergie ou d'information, ou encore systèmes fortement redondants et interconnectés. La simulation stochastique de réseaux implique de propager l'information qui y circule, ce qui sollicite fortement le mécanisme de mise à jour de l'état du modèle, notamment lorsque plusieurs chemins sont possibles pour aller d'un point à un autre.

5.a Mesh

Ce cas-test représente un ensemble de composants fortement interconnectés, disposés sous forme de grille, et échangeant une information (par exemple, un potentiel électrique, ou une pression hydraulique). Chaque composant (ou nœud du réseau) échange bi-directionnellement avec ses quatre voisins. Un nœud peut défaillir, suivant un taux constant, et être réparé, suivant un taux constant. Si un nœud est en état de fonctionnement, alors ses quatre ports d'entrée-sortie sont à la même valeur. Sinon, leurs valeurs ne sont pas contraintes. Les ports extérieurs d'un des nœuds situés dans un coin sont mis à un potentiel (ou une pression). On observe la valeur des ports extérieurs du composant situé dans le coin opposé.
Ce modèle est bouclé : la valeur peut se propager dans chaque lien suivant les deux directions.
Le problème peut se résumer à l'existence d'un chemin de composants en état de fonctionnement entre deux coins de la grille.
La taille du modèle est paramétrable en faisant varier la dimension de la grille.

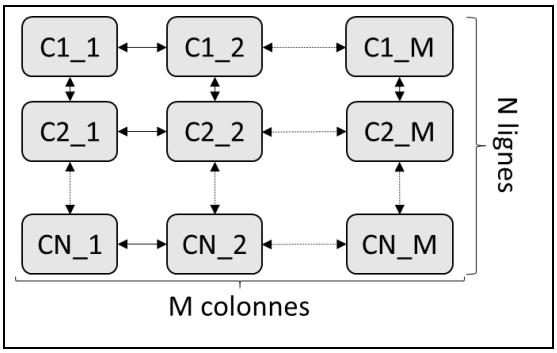

Figure 10. Architecture d'un Mesh

6 Jeu de la vie

La simulation stochastique d'automates cellulaires est utilisée notamment pour l'étude de propagations (par exemple : épidémies, incendies). Ces automates sont constitués de nombreuses cellules, organisées suivant une structure définie. L'évolution se fait par pas : à chaque pas, l'état de chaque cellule dépend de l'état précédent des cellules voisines, suivant des lois qui peuvent être stochastiques ou déterministes, identiques ou différentes suivant les cellules, constantes ou variantes suivant le pas.

6.a Jeu de la vie

On s'intéresse ici à une classe particulière d'automates cellulaires: le jeu de la vie de Conway. Le jeu est constitué d'une grille de cellules carrées. Les cellules sont soit mortes, soit vivantes. A chaque pas, l'état de chaque cellule dépend de l'état des 8 cellules voisines à l'état précédent. Si exactement 3 cellules voisines étaient vivantes, alors la cellule devient vivante. Si exactement 2 cellules voisines étaient vivantes, alors la cellule reste dans son état précédent. Sinon, la cellule meurt.
Ces automates étant déterministes, les résultats de leur simulation stochastique peuvent être connus préalablement, et ne dépendent que de la taille de la grille et de la configuration initiale. De nombreuses configurations initiales possèdent des propriétés particulières, comme des périodes, ou des représentations graphiques remarquables. Par exemple, la figure nommée pentadécathlon (découverte par J. Conway en 1970) a une période de 15 pas. On peut donc chercher à observer cette période.
L'évolution se faisant dans toutes les cellules simultanément, la modélisation de cet automate demande une synchronisation de la mise à jour de chaque cellule. Sa simulation stochastique sollicite donc l'échéancier pour cette synchronisation, ainsi que le mécanisme de mise à jour du modèle car toutes les variables sont affectées à chaque transition. L'observation d'une période oblige à surveiller la valeur de chaque cellule : le calcul de l'indicateur est donc complexe.

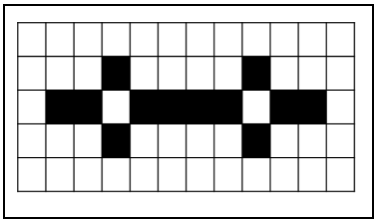

Figure 11. Configuration initiale d'un pentadécathlon (les cellules noires sont vivantes)

Application

La table 1 résume les caractéristiques sollicitées par chacun des cas-tests. Pratiquement toutes sont sollicitées par un ou plusieurs cas-tests, dont les résultats sont vérifiables par des méthodes différentes.
Concernant la quantité d'indicateurs, aucun cas-test ne dépend de la performance de l'outil de simulation stochastique dans ce domaine. Pour évaluer cette performance, il suffit de prendre un cas-test existant et de rajouter un grand nombre d'indicateurs : par exemple, en plaçant un indicateur sur la fiabilité de chaque composant d'un Série-Parallèle avec composants réparables.

L'utilisation de ce kit d'évaluation est à adapter aux exigences de l'utilisateur, et au processus dans lequel il souhaite étudier son utilisation. Une première étape est de déterminer, à partir des modèles qui seront à étudier, les caractéristiques de l'outil de simulation stochastique évènementiel qui seront les plus sollicitées. Il est ensuite possible de sélectionner les cas-tests qui permettent d'évaluer ces caractéristiques parmi ceux présents dans le kit d'évaluation. L'étape suivante dépend des exigences de l'utilisateur.

Afin d'estimer la puissance de calcul, ou le temps de calcul, qui seront nécessaires, pour simuler les modèles envisagés, il suffit de paramétrer les cas-tests choisis pour que leur taille corresponde, et les simuler. Si le temps de simulation estimé est trop important, il est possible de l'extrapoler à partir d'instances plus petites des cas-tests.
Afin d'estimer le nombre de simulations nécessaires pour obtenir un intervalle de confiance suffisamment réduit sur les indicateurs, il est possible de simuler une instance du cas-test de la taille souhaitée avec plus ou moins de simulations, et d'interpoler ainsi un nombre minimal de simulations pour que cet intervalle de confiance réponde à l'exigence.

Dans le cadre du développement du simulateur stochastique de modèles AltaRica 3.0 (Batteux et al., 2013), l'ensemble de ces cas-tests est utilisé afin de dresser un bilan des points forts et des faiblesses de l'outil dans une version donnée, et d'obtenir des indicateurs sur la progression entre versions. Par exemple, la simulation du cas-test Mesh a permis de détecter un problème de propagation des valeurs de variables, qui devenait extrêmement coûteux lorsque la taille du cas-test augmentait. Ce défaut a été résolu dans la version suivante.

	Série-Parallèle				Chaîne de Markov					
	Défaillance simple	Défaillance réparable	Réparateurs limités	Redondance froide	Réparateurs limités redondance froide	Panne cachée	Arbre de défaillances	Round Robin	Mesh	Jeu de la vie
Mise à jour du modèle :										
• MàJ des variables										+
• MàJ du modèle	+	+	+	+			+		+	
• Synchronisation par propagation				+				+		
Echéancier :										
• Nombre de transitions	+	+	+	+	+	+	+			
• (Des)activations de transitions					+	+		+		
• Synchronisation par transitions			+							+
Indicateurs :										
• Quantité d'indicateurs										
• Complexité des indicateurs					+	+				+

Table 1. Caractéristiques testées en fonction du cas-test

Conclusion

Le kit d'évaluation proposé ici permet l'évaluation d'outils de simulation stochastique évènementielle, tant d'un point de vue de la qualité des résultats que de celui de la performance en temps des simulations. Il couvre les principales caractéristiques des mécanismes de simulation stochastique évènementielle, et les résultats sont comparables à l'aide de différentes méthodes (formules analytiques, chaînes de Markov, arbres de défaillances, …). Il est utilisé pour le simulateur stochastique AltaRica 3.0, tout en étant utilisable pour d'autres simulateurs stochastiques de modèles évènementiels. Il contribue ainsi à favoriser l'acceptation, par l'industrie et par les autorités de sûreté, des analyses probabilistes du risque, en renforçant la confiance que l'on peut avoir dans ces outils.

Références

Nikolskaia et Rauzy, 1998. Heuristics for BDD handling of sum-of-products formulae. In Proceedings of the European Safety and Reliability Association Conference, ESREL'98, Trondheim, June 1998.
Batteux et Rauzy, 2013, Stochastic Simulation of AltaRica 3.0 models, in Proceedings of the European Safety and Reliability Conference, ESREL 2013
Prosvirnova, Batteux, Brameret, Cherfi, Friedlhuber, Roussel et 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
RTCA, 2011, DO-330 Software Tool Qualification Considerations.
RTCA, 2012, DO-178C Software Considerations in Airborne Systems and Equipment Certification.
Zio 2013, The Monte Carlo Simulation Method for System Reliability and Risk Analysis. Springer.