Calcul d'indicateurs de sûreté de fonctionnement de modèles AltaRica 3.0 par simulation stochastique

Benjamin Aupetit

université
PARIS-SACLAY

Calcul d'indicateurs de sûreté de fonctionnement de modèles AltaRica 3.0 par simulation stochastique

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 573,
interfaces : approches interdisciplinaires, fondements, applications et innovation (Interfaces)
Spécialité de doctorat: Informatique
Unité de recherche : Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, 91190, Gif-sur-Yvette, France
Référent : CentraleSupélec

Thèse présentée et soutenue en visioconférence totale, le 23 juin 2020, par

Benjamin AUPETIT

Composition du jury:

Marija Jankovic Professeure des Universités, CentraleSupélec (LGI)	Présidente
Jean-François Pétin Professeur des Universités, ENSEM (CRAN)	Rapporteur
Laurent Piétrac Maître de Conférences HDR, INSA de Lyon (Ampère)	Rapporteur
Michel Batteux Docteur, IRT SystemX	Examinateur
Antoine Rauzy Professeur des Universités, NTNU (IPK)	Directeur de thèse
Jean-Marc Roussel Maître de Conférences HDR, ENS Paris-Saclay (LURPA)	Co-directeur de thèse

Thèse de doctorat

NNT: 2020UPASC004

Remerciements

Je tiens à remercier les membres du jury de ma soutenance de thèse : Marija Jankovic pour avoir accepté de présider ce jury, ainsi que Jean-François Pétin et Laurent Piétrac pour avoir accepté de rapporter mon manuscrit. J'ai apprécié les échanges que nous avons pu avoir avant et pendant la soutenance. Je remercie Antoine Rauzy d'avoir accepté de diriger ma thèse, depuis Châtenay-Malabry puis depuis Trondheim. Malgré la distance, les échanges scientifiques que nous avons pu avoir ont été formateurs. Merci aussi à Michel Batteux pour son encadrement au quotidien et pour sa patience. J'ai beaucoup appris grâce à lui, et ce aussi bien scientifiquement qu'humainement.

Jean-Marc Roussel a eu un rôle particulier durant ma thèse : outre m'avoir proposé pour ce sujet de recherche à la fin de mon Master 2 à Cachan, et outre avoir accepté d'en être codirecteur, il a été très présent durant la seconde partie de ma thèse. Il ne fait pas de doute que sans son aide, je n'aurais pas poursuivi jusqu'à la soutenance. Je lui adresse toute mon amitié et mes remerciements.

J'ai eu l'occasion de rencontrer de nombreuses personnes pendant la première partie de ma thèse, avec lesquelles j'ai eu plaisir à travailler ou juste discuter. Il y a en particulier "l'équipe AltaRica" : Tatiana, Melissa, Pierre-Antoine, Benoit, Anthony, Abraham... Mais aussi les collègues, cobureaux ou comachine à café : Anaïs, Marine, Emna, Flavien, Makhlouf, Ali, Stephen, Sullivan, Ming, Laurent, James ... Sans oublier NoL@g, ainsi que les membres du LGI et du LURPA.

Durant la seconde partie de ma thèse, j'ai aussi pu travailler et discuter avec plusieurs personnes sur d'autres sujets que de la recherche. Elles ont été accueillantes pour me permettre de prendre pied dans mon nouvel emploi, et compréhensives pour me laisser du temps pour finir ma thèse. Merci donc à Bernard, Isabelle, Romain, Rachid, Mathieu, Bertrand, Valérie, Ariane, Françoise, Laurent, Leïla, Thomas...

Je tiens aussi à remercier mes amis, avec et sans apostrophes, qui m'ont supporté durant cette période : Antoine, Laura, Laureen, Marissa, Michel, Pierre-Antoine, Steven, Valentine, Vanessa ...

Je souhaite remercier en particulier Johanna et Maxime d'avoir été aussi présents, et ce dans tous les moments.

Benjamin

Sommaire

Introduction générale

La conception d'un système critique nécessite la réalisation de différentes études de sûreté permettant de statuer si le système qui est en cours de mise au point fonctionnera avec le taux de disponibilité escompté et que les conséquences de ses défaillances resteront dans les limites attendues de l'analyse de risques. Pour les systèmes les plus critiques, ces études de sûreté démarrent dès la phase d'avant-projet. Elles ont pour objectif d'écarter au plus tôt les solutions techniques inacceptables d'un point de vue de la criticité des risques encourus, ainsi que celles pour lesquelles la disponibilité opérationnelle est insuffisante. Grâce à son expérience, un ingénieur fiabiliste est en mesure d'écarter rapidement une très mauvaise solution qui pourrait lui être présentée. Cependant, il lui est difficile d'interclasser plusieurs solutions acceptables sans réaliser une analyse comparée précise s'appuyant sur des critères qualitatifs et/ou quantitatifs.

Classer des solutions potentielles en s'appuyant sur le taux de disponibilité est déjà un exercice très difficile à réaliser en raison du nombre de paramètres à prendre en compte et de leur interaction. Ce taux dépend naturellement de la structure physique du système. Il faut donc tenir compte des différents taux de défaillance de chacun des composants qui le composent, des taux de réparation des éléments réparables et de la politique de redondance mise en place. Il est parfois nécessaire de prendre en compte également la politique de maintenance envisagée pour l'installation voire de la disponibilité des équipes et de leurs équipements.

Pour estimer la simple disponibilité d'un système, il est donc nécessaire d'étudier le comportement dynamique de celui-ci, de suivre ses évolutions aux cours du temps en analysant ses réactions (changement d'états) lors de l'occurrence des événements de panne ou de réparation de chacun des composants élémentaires, et donc de les évaluer quantitativement.

D'un point de vue technique, le chiffrage de la disponibilité d'un système pour une histoire donnée ne présente pas de difficulté majeure : il suffit de rejouer cette histoire en mesurant le temps de séjour dans les états où le système est disponible, de sommer ces mesures et de diviser le tout par la durée totale de l'histoire. Ce chiffre est donc simple à obtenir lorsque l'histoire est connue.

Ce qui rend difficile l'estimation de la disponibilité d'un système est que les occurrences des événements de panne et de réparation ne se produisent pas à des dates fixes, mais doivent suivre des modèles de distributions probabilistes variés pour être représentatifs de la réalité. Il n'y a donc pas une seule histoire possible pour un système, mais une infinité d'histoires possibles. Pour être à même de conclure sur la disponibilité du système, ce n'est pas donc une histoire particulière qu'il est nécessaire de simuler, mais un grand nombre de ces histoires pour ensuite conduire une étude probabiliste sur les valeurs mesurées (espérance, bornes extrêmes, écart-type). La qualité du résultat dépendra du nombre d'histoires simulées, mais également de leur représentativité.

Lorsque le comportement du système à étudier est décrit par un modèle probabiliste, la simulation stochastique de ce modèle permet de construire une histoire possible pour ce système afin de l'évaluer ensuite. Cette simulation stochastique doit être faite en respectant, d'une part, le comportement dynamique du modèle probabiliste support (choix des événements qui doivent occurrer) et, d'autre part, les modèles de distribution retenus pour les occurrences de ces événements (date de leur occurrence). C'est au simulateur stochastique que revient la charge de choisir les événements qui doivent survenir et la date de leur occurrence à partir du modèle proposé. La qualité des résultats obtenus repose en grande partie sur la capacité du simulateur stochastique à prendre en compte toutes les spécificités du comportement dynamique porté par le modèle.

D'une manière générale, établir des indicateurs de sûreté de fonctionnement dans lesquels on peut avoir confiance par simulation stochastique nécessite d'être en mesure :

- **de pouvoir jouer un grand nombre d'histoires afin d'être dans la capacité d'intercepter des événements rares** : en raison de la différence d'ordre de grandeur entre les probabilités d'occurrence des événements de panne et ceux de réparation, la simulation stochastique a tendance à ne parcourir que la partie du modèle où seuls quelques composants sont en panne. La défaillance complète du système peut ne jamais être atteinte si le nombre d'histoires simulées est faible.
- **de reproduire avec justesse le comportement du modèle support de l'étude** : en raison de la complexité des systèmes étudiés, le modèle support de l'étude mise à disposition de l'analyste n'est plus l'expression directe de l'espace d'état du système étudié, mais un moyen de le générer. Le simulateur stochastique doit être en mesure de reproduire sans erreur chaque évolution du modèle afin de ne pas introduire de biais de comportement. Plus le pouvoir d'expression du langage de modélisation est élevé, plus il est difficile de reproduire son comportement via un autre langage.
- **de dérouler les études statistiques sur les indicateurs recherchés au plus tôt** : lorsque le calcul des indicateurs et les études statistiques qui s'y rattachent sont faits en dehors du simulateur stochastique, il est alors nécessaire de conserver la trace complète de l'histoire générée afin de pouvoir l'exploiter avec précision. Lorsque les modèles sont très fins, c'est-à-dire, très détaillés, la trace complète de l'histoire générée peut rapidement donner lieu à un volume de données important.

Dérouler une étude de sûreté de fonctionnement en estimant ses indicateurs par simulation stochastique a forcément un coût en raison du nombre d'histoires qu'il sera nécessaire de construire et d'évaluer. Cependant, les performances actuelles des ordinateurs rendent cette approche envisageable. L'ingénieur fiabiliste n'est plus obligé de se contraindre à travailler avec des arbres de défaillance ou chaines de Markov car il ne disposait que de ces outils mathématiques pour lui permettre d'établir des chiffres.

Lorsqu'il souhaite établir ses indicateurs de sûreté de fonctionnement en phase d'avant projet, l'ingénieur fiabiliste est confronté à des contraintes supplémentaires. Il doit être en mesure de dérouler ses études dans un délai relativement court avec des moyens limités tant sur le plan des ressources humaines que calculatoires.

D'une manière générale, pour pouvoir être opérationnel, un ingénieur fiabiliste doit disposer :

- **d'un langage de haut niveau disposant d'un fort pouvoir d'expression** afin de modéliser simplement les éléments qu'il doit prendre en compte : plus le langage sera expressif, plus le temps de création des modèles sera court.
- **d'un environnement logiciel d'édition** lui permettant de pourvoir **saisir et tester** les éléments de **son modèle** : en raison de la complexité du comportement à décrire, le modélisateur ne peut plus proposer un modèle sans pouvoir observer ses évolutions. Le modélisateur a besoin de pouvoir simuler son modèle en mode pas-à-pas afin de vérifier que les synchronisations qu'il envisage sont correctement exprimées.
- **d'un environnement logiciel ouvert pour le calcul des indicateurs de sûreté** : d'un point de vue strictement calculatoire, l'ingénieur fiabiliste a à sa disposition plusieurs stratégies pour établir ces indicateurs de sûreté. Il peut travailler à partir de simulations stochastiques, de chaines de Markov ou d'arbres de défaillance. Il est évident que selon l'outil mathématique choisi, seuls certains aspects de sa modélisation pourront être pris en compte, mais cela peut être suffisant pour dégrossir des solutions. Pour l'ingénieur fiabiliste, l'idéal est de pouvoir choisir librement sa technique de calcul en fonction de son besoin tout en exploitant au mieux les informations contenues dans son modèle.
- **d'un environnement logiciel de calcul qualifiable** : en raison des impacts économiques pouvant être en jeu, tout logiciel de calcul d'indicateurs devrait faire l'objet de campagnes d'évaluation afin de permettre à ses utilisateurs de pouvoir juger de ses performances tant sur le plan de la rapidité d'obtention des chiffres, que sur leur représentativité.

C'est dans ce cadre que se place le projet AltaRica 3.0. Celui-ci vise à fournir un environnement logiciel permettant l'édition et l'analyse de modèles de sûreté de fonctionnement, autour du formalisme de modélisation AltaRica 3.0. Le simulateur stochastique de modèle AltaRica 3.0, qui est l'un des outils de cet environnement logiciel, a été l'objet d'étude de ce manuscrit.

Chapitre I

État de l'art

1 Introduction

Afin d'aborder le thème principal de cette thèse, qui est l'établissement d'indicateurs de sûreté de fonctionnement par simulation stochastique à partir d'un modèle AltaRica 3.0, il nous a été nécessaire de revenir sur les méthodes actuelles dont disposent les fiabilistes pour obtenir ces chiffres.

Lors de cette analyse, nous avons fait le point sur les avantages et les inconvénients de ces techniques et les possibilités qu'elles offrent. Nous avons également analysé comment les indicateurs de sûreté de fonctionnement sont calculés lors de leur utilisation. Dans le cadre de ce travail de thèse, nous avons d'ailleurs expérimenté ces méthodes afin de disposer de chiffres pour évaluer notre simulateur stochastique. Pour que cette évaluation repose sur des données vérifiables, il nous a été nécessaire de construire des études de cas adaptées à ces approches afin de pouvoir évaluer les écarts introduits par le paramétrage du simulateur stochastique.

Nous avons ensuite fait le point sur la simulation stochastique, les difficultés que cette méthode pose, et les techniques pouvant être mises en œuvre pour accélérer les calculs.

Enfin, l'étude d'AltaRica 3.0 a montré que ce langage présente des particularités qui rendent son comportement difficile voire impossible à reproduire dans les langages d'entrée de simulateurs stochastiques existants.

2 Approches centrées sur l'outil

Les approches de sûreté de fonctionnement centrées sur l'outil partent du principe que c'est l'outil de simulation qui est le plus contraignant dans la démarche de simulation : le modèle que fournit l'analyste doit donc s'adapter à l'outil utilisé. Il s'agit de l'approche historique : elle a commencé avec des formalismes de modélisation qui étaient déjà utilisés pour les études de sûreté de fonctionnement avant l'arrivée de l'informatique.

Ce sont bien les moyens de calculs limités des logiciels informatiques (années 1980–1990) qui sont à l'origine de la contrainte principale et forte portant sur les formalismes de modélisation : pour pouvoir être facilement simulables, les modèles doivent présenter des propriétés mathématiques exploitables par les outils de simulation pour faciliter les calculs.

Deux familles de modèles sont principalement exploitées avec cette approche : les modèles booléens (arbres de défaillances, de fautes, blocs-diagrammes de fiabilité), décrits en I.2.1 et les modèles états-transitions (chaînes de Markov, réseaux de Petri stochastiques) décrits en I.2.2.

2.1 Modèles booléens

Les modèles booléens de sûreté de fonctionnement représentent les combinaisons de conditions de bon fonctionnement (ou de mauvais fonctionnement) du système étudié. Ces conditions s'expriment à partir de variables booléennes (à deux états, par exemple un composant est défaillant ou non). Ils reposent sur des propriétés facilitant leur simulation, comme l'indépendance des variables, au prix d'une représentativité des systèmes étudiés faible.

Les formalismes booléens principalement utilisés pour des études de sûreté de fonctionnement sont les arbres de défaillances et les blocs-diagrammes de fiabilité.

2.1.a Arbres de défaillances

Un arbre de défaillances permet de représenter, pour un système, les combinaisons de fautes qui peuvent mener à l'occurrence d'un évènement redouté, comme la défaillance du système. L'objectif d'une étude quantitative par arbre de défaillances est d'obtenir la probabilité d'occurrence de l'évènement redouté, à une date donnée. Il est aussi possible d'obtenir des grandeurs telles que des facteurs d'importance d'un composant dans la fiabilité d'un système.

i Modèle

Le modèle, représentable graphiquement sous forme d'arbre, est composé :

- de l'évènement redouté, qui est le sommet de l'arbre ;
- de connecteurs logiques, similaires à des portes de systèmes combinatoires : ET, OU (ainsi que des connecteurs dérivés de ces portes : k parmi n...).
- d'évènements intermédiaires, dont des combinaisons suivant des connecteurs logiques permettent d'obtenir l'évènement redouté ou d'autres évènements intermédiaires ;
- d'évènements de base, formant les feuilles de l'arbre et dont des combinaisons via des connecteurs logiques permettent d'obtenir des évènements intermédiaires. Ils représentent des évènements élémentaires, comme des pannes de composants, pour lesquels les probabilités d'occurrence sont connues et qu'il n'est pas possible ou utile de décomposer.

Les évènements de base sont des variables booléennes, dont la valeur (*true* ou *false*) correspond à l'occurrence (ou non) de l'évènement correspondant, à une date donnée. Les évènements intermédiaires et l'évènement redouté sont des expressions booléennes, combinaisons des évènements de base par les connecteurs logiques.

ii Démarche de modélisation

La modélisation d'une solution technique par arbre de défaillances suit une méthodologie normalisée [69], qui est déductive ("top-down") : à partir de l'évènement redouté, chaque évènement est raffiné en une combinaison de causes qui sont des évènements existants ou nouveaux. Le raffinement continue jusqu'à obtenir des évènements de base, auxquels il est possible d'associer des distributions de probabilités d'occurrence.

iii Méthode de calcul

Le calcul de la probabilité d'occurrence de l'évènement redouté est réalisé par la détermination des coupes minimales, c'est-à-dire les combinaisons d'évènements les plus courtes qui mènent à l'évènement redouté.

Coupes minimales Une coupe est un ensemble d'évènements dont les occurrences mènent à l'évènement redouté. En d'autres termes, une coupe est un ensemble de variables booléennes tel que si elles valent *true*, l'expression booléenne de l'évènement redouté aura pour valeur *true*. Une coupe est minimale si tous les évènements qui la composent sont nécessaires, c'est-à-dire s'il n'est pas possible de retirer une ou plusieurs variables de l'ensemble pour obtenir ainsi une autre coupe.

Obtention des coupes minimales Historiquement, les coupes minimales d'un arbre de défaillances étaient calculées par l'analyste, par réarrangement de l'arbre ou par factorisation booléenne [75]. Ces méthodes ont été reprises pour les premiers algorithmes de calcul des coupes minimales.

L'arbre de défaillances est équivalent à un arbre dont l'évènement redouté est obtenu par une porte OU, regroupant chacune des coupes minimales. Par des règles de réarrangement de l'arbre (semblables aux règles de simplification ou d'expansion d'équations booléennes), il est possible de passer du modèle à un tel arbre équivalent. Les coupes minimales peuvent alors être directement lues sur l'arbre équivalent.

Des méthodes similaires dans le principe peuvent être réalisées sur l'expression booléenne de l'évènement redouté.

Calcul de la probabilité La probabilité qu'une combinaison d'évènement survienne est calculée en faisant le produit des probabilités des évènements qui la composent. La probabilité d'occurrence de l'évènement redouté peut être approximée comme étant la somme des probabilités des coupes minimales. Cette approximation néglige la probabilité que les évènements composant deux coupes minimales distinctes occurrent tous.

L'approximation du calcul mène à surestimer la probabilité d'occurrence de l'évènement redouté. Le système en cours de conception peut alors être surdimensionné par rapport aux exigences de sûreté de fonctionnement.

Toutefois, à cause de l'explosion combinatoire, le nombre de coupes minimales peut être très important : celles comportant le plus d'évènements peuvent alors être négligées, car ce sont les moins probables. Ce calcul peut mener à une sous-estimation de la probabilité d'occurrence de l'évènement redouté.

2.1.b Blocs-diagrammes de fiabilité

À l'inverse d'un modèle par arbre de défaillances, un modèle par bloc-diagramme de fiabilité va représenter les combinaisons de conditions permettant au système étudié de fonctionner correctement. Ils permettent de calculer la probabilité de bon fonctionnement du système, en fonction des probabilités de bon fonctionnement de ses composants à une date donnée.

i Modèle

Un modèle bloc-diagramme de fiabilité est représenté graphiquement par des blocs reliés

entre eux par des arcs [70]. Chaque composant du système est modélisé par un bloc. Un groupe de bloc peut représenter un sous-système. La façon dont sont reliés les blocs (ou groupes de blocs) par des arcs modélise la condition à laquelle le système est en état de bon fonctionnement :

- Si les blocs sont reliés en série, alors le bon fonctionnement de tous les composants ou sous-systèmes correspondants est nécessaire au bon fonctionnement du système ;
- Si les blocs sont reliés en parallèle, alors le bon fonctionnement d'un seul des composants ou sous-systèmes correspondants est nécessaire au bon fonctionnement du système ;
- Une notation particulière, appelée "vote majoritaire" permet d'indiquer qu'au moins k composants ou sous-systèmes sur n sont nécessaires au bon fonctionnement du système.

ii Démarche de modélisation

De la même façon qu'avec les arbres de défaillances, la démarche de modélisation est déductive : à partir du système, un raffinement permet de lister les combinaisons de sous-systèmes nécessaires à son bon fonctionnement. Ces sous-systèmes sont ensuite à leur tour raffinés, jusqu'à obtenir des composants élémentaires pour lesquels les probabilités de défaillance sont connues.

iii Calculs et équivalence aux arbres de défaillances

L'estimation de la probabilité de bon fonctionnement du système modélisé par un bloc-diagramme de fiabilité se fait par traduction du bloc-diagramme en équation booléenne. Les mêmes méthodes de calculs que pour les arbres de défaillances peuvent alors être utilisées, pour obtenir ici des probabilités de bon fonctionnement (c'est-à-dire la fiabilité) à partir des coupes minimales.

2.1.c Limitations des modèles booléens

Les formalismes booléens présentent des limitations communes, portant sur trois aspects : les calculs, l'expressivité des modèles et le principe de modélisation.

i Calculs

L'obtention des coupes minimales est l'étape la plus coûteuse des calculs, et rend difficile l'utilisation de modèles de grande taille en raison de l'explosion combinatoire.

ii Expressivité

Une limitation importante des études par modèles booléens est l'hypothèse d'indépendance des évènements, nécessaire pour le calcul simple de la probabilité d'occurrence d'une coupe. Il n'est donc pas possible de représenter deux modes de défaillance d'un même composant mutuellement exclusifs.

De même, les connecteurs logiques sont combinatoires : ils ne prennent pas en compte l'ordre d'occurrence des évènements, ou le délai entre eux. Le cas de la défaillance d'un composant après celle de son mécanisme de redondance (provoquant une faute) ne peut pas être distingué du cas inverse (défaillance du composant avant le mécanisme de redondance, ne provoquant pas nécessairement de faute).

De façon plus générale, il est difficile de modéliser les réactions d'un système à un évènement tel qu'une défaillance, comme une reconfiguration. En effet, le modèle est statique : il ne peut pas être modifié en fonction des évènements survenus. Des mécanismes tels que des redondances froides ou tièdes, répandues dans les systèmes critiques car améliorant la disponibilité, ne sont pas modélisables de façon suffisamment représentative pour pouvoir les différencier d'une redondance chaude.

La représentativité des modèles booléens des comportements des systèmes modélisés est donc limitée, ce qui peut conduire l'analyste de sûreté de fonctionnement à surestimer les risques, et donc à surdimensionner le système réel étudié.

iii Principe de modélisation

Un arbre de défaillances est spécifique à un système et à un évènement redouté, tout comme un bloc-diagramme de fiabilité est spécifique à un système et à une fonction du système. Dans le cadre d'une étude de sûreté de fonctionnement, il est donc nécessaire de produire un modèle pour chaque évènement redouté, chaque fonction, et chaque solution technique étudiée : le nombre de modèle à créer peut donc être très important.

La démarche de conception par raffinement successif à partir de l'évènement redouté (ou de la fonction), si elle permet de limiter les erreurs de modélisation en segmentant le travail, rend difficile la réutilisation d'un modèle en cas de modification de la solution technique étudiée.

La modélisation d'un système par modèle booléen est donc très coûteuse en temps de travail en phase de conception.

2.1.d Travaux conduits pour l'amélioration des possibilités de calcul

L'amélioration des méthodes de calcul de modèles booléens a fait l'objet de plusieurs travaux, essentiellement en ce qui concerne la recherche des coupes minimales.

Les coupes minimales peuvent être efficacement déterminées grâce à des méthodes utilisant des arbres de décisions binaires (BDD) [59, 61]. Les arbres de décisions binaires sont des graphes acycliques : chaque nœud correspond à une variable booléenne (ici, un évènement de base de l'arbre de défaillances ou l'état d'un composant du bloc-diagramme de fiabilité), et un arc entre deux nœuds correspond à un "ET" logique entre les variables. L'arbre de décisions binaires est donc une représentation de l'expression booléenne de l'évènement redouté (ou de l'état du système), qui est d'autant plus compacte que les variables sont correctement ordonnées : des heuristiques permettent d'améliorer cette compacité [27].

Couplés à des implémentations réfléchies [63], les arbres de décisions binaires repoussent les limites en taille et améliorent la rapidité du calcul de l'estimation de grandeurs probabilistes à partir de modèles booléens. Il n'est plus nécessaire d'établir les coupes minimales et le calcul des grandeurs probabilistes est exact.

2.1.e Travaux sur l'expressivité

L'expressivité limitée des formalismes booléens ne permet qu'une faible représentativité des modèles et oblige pour compenser à surdimensionner les systèmes étudiés. Cette limitation vient principalement du caractère combinatoire (statique) du modèle.

i Modèles dynamiques

Plusieurs travaux ont proposé des portes logiques dynamiques pour les arbres de défaillances [30], ce qui a mené aux arbres de défaillances dynamiques. Des propositions comparables ont mené aux blocs-diagrammes de fiabilité dynamiques.

La porte ET-Prioritaire [21] est l'une de ces portes dynamiques pour arbres de défaillances, sa sortie n'étant le booléen vrai que si les occurrences des évènements en entrées se sont produites dans un ordre spécifique : elle n'est donc pas combinatoire, mais séquentielle. On peut ainsi représenter la différence de conséquences entre la défaillance d'un composant avant, ou après, celle d'un mécanisme de redondance. Les portes dynamiques permettent ainsi une meilleure expressivité des arbres de défaillances, donc une meilleure représentativité des modèles.

L'estimation d'indicateurs par calcul sur des modèles booléens dynamiques est en revanche plus complexe [18, 45, 37] car les propriétés mathématiques des modèles statiques ne sont plus utilisables lors des calculs. Une solution est la conversion du modèle en une chaîne de Markov (voir I.2.2.a.iii).

ii Multi-états

L'utilisation de variables booléennes, associée à l'hypothèse d'indépendance des variables, ne permet pas une modélisation représentative des modes de défaillances différents d'un composant : celui-ci ne peut être modélisé que comme en bon état ou défaillant.

Une solution proposée est d'étendre les formalismes booléens à des variables multi-états, ce qui permet d'obtenir des arbres de défaillances multi-états (Multi-state Fault Trees, MFT) [16], ainsi que des blocs-diagrammes de fiabilité multi-états [77]. Dans ces modèles, les variables peuvent donc avoir une valeur dans un ensemble autre que binaire, permettant de représenter notamment des modes de défaillance exclusifs.

Les calculs sur ces modèles multi-états sont toujours possibles par la recherche des coupes minimales, en adaptant les algorithmes utilisés [78]. Cependant, le coût de calcul augmente.

iii Modèles non booléens

Les modèles booléens (statiques ou non) peuvent aussi être utilisés pour piloter des processus markoviens ou des réseaux de Petri. Les modèles obtenus ne sont alors plus calculables analytiquement : la simulation stochastique doit alors être utilisée [67].

2.2 Modèles états-transitions

Les modèles états-transitions permettent de représenter les évolutions d'un système en fonction d'évènements (tels que des défaillances ou des réparations de composants) mais aussi en fonction de l'état du système au moment de ces évènements. Ce sont donc des modèles dynamiques.

Deux formalismes états-transitions sont principalement utilisés pour les études de sûreté de fonctionnement : les chaînes de Markov, et les réseaux de Petri stochastiques.

2.2.a Chaînes de Markov

Un modèle par chaîne de Markov [42] est une représentation exhaustive des états atteignables par un système, et des transitions possibles entre chacun de ces états. Un tel modèle

permet de calculer la probabilité que le système soit dans un état particulier (par exemple, en panne) à une date donnée.

i Modèle

Une chaîne de Markov est une machine à états : elle est composée d'un ensemble d'états, reliés par des transitions orientées. Les états représentent chacun un état du système, auquel est associé après calcul la probabilité que le système soit dans cet état à l'instant donné.

Des probabilités de franchissement sont associées à chaque transition, sous la forme d'une fréquence constante (chaîne de Markov à temps continu). Pour chaque transition partant d'un état, il y a la probabilité correspondante de changer d'état pour celui indiqué par la transition.

Le nombre d'états d'une chaîne de Markov est fini : dans le cas contraire, le modèle est un processus de Markov. Comme l'espace d'état est discret, il n'est pas possible de représenter une usure continue d'un composant par exemple.

Les états intéressants (par exemple, les états dans lesquels le système est en panne) sont marqués : la simulation d'une chaîne de Markov consistera à déterminer la probabilité, à une date donnée, que le système soit dans un tel état marqué. On peut ainsi obtenir la fiabilité et la disponibilité d'un système, ainsi que des temps moyens (MTTF, MTTR).

ii Modélisation

La construction d'un modèle d'un système par chaîne de Markov peut être très laborieuse : le nombre d'états est le produit des nombres d'états de chacun des modèles des composants du système étudié, ce qui peut rapidement conduire à un nombre très élevé. Cette tâche est alors coûteuse, et est source d'erreur.

Une chaîne de Markov est équivalente au graphe d'états d'un automate à états : il est donc possible de construire une chaîne de Markov (qui est un modèle bas-niveau) à l'aide d'un formalisme état-transition de plus haut niveau, pour obtenir son espace d'état par calcul.

iii Calculs

Pour être évaluable, une chaîne de Markov doit respecter la propriété fondamentale suivante : son évolution ne doit dépendre que de son état courant. Cette propriété est appelée hypothèse markovienne, et ne permet de décrire que des phénomènes "sans mémoire" (c'est-à-dire, sans mémoire autre que celle représentée par l'état courant du modèle). Elle implique en particulier que les probabilités de transition sont constantes. Les chaînes de Markov ne peuvent donc être utilisées que pour modéliser des phénomènes par distributions de probabilité exponentielles : une modélisation classique lors d'études de sûreté de fonctionnement approxime par une telle distribution le taux de défaillance (λ) et de réparation (μ) d'un composant.

Cette propriété permet un calcul simple de la probabilité, par résolution d'équations différentielles de façon analytique ou approchée [60].

iv Limitations et améliorations

L'explosion combinatoire qui survient lors de la modélisation de systèmes avec un nombre important de composants et d'états rend les calculs, voire même la construction même de la chaîne de Markov, coûteux. Dans [15], l'auteur utilise une propriété des modèles de sûreté

de fonctionnement pour rendre possible la construction de la chaîne de Markov d'un tel système : les taux de défaillance de composants étant plusieurs ordres de grandeur plus faibles que les taux de réparation correspondant, les états correspondant à un nombre important de composants défaillants sont peu probables, et leur différenciation n'est pas utile. Il est donc possible de fusionner ces états en un seul état absorbant ("the sink"), dans lequel le système est considéré défaillant, ce qui mène à surestimer la probabilité de défaillance.

2.2.b Réseaux de Petri stochastiques

Les réseaux de Petri stochastiques généralisés [1] sont une classe des réseaux de Petri [52] :

- généralisé : les transitions sont pondérées. Une transition peut consommer un nombre quelconque de jetons (dans aucune, une ou plusieurs places), et en produire un nombre quelconque (dans aucune, une ou plusieurs places).
- stochastiques : à chaque transition temporisée est associé un délai stochastique, défini par une distribution de probabilités exponentielle. De plus, plusieurs transitions immédiates peuvent être en concurrence : la transition tirée est choisie en fonction des probabilités associées.

i Modélisation

La modélisation peut se faire par raffinement successif, ou par assemblage de sous-systèmes modélisés indépendamment. De nombreux comportements sont généralement représentables par des réseaux de Petri [12]. Les propriétés stochastiques de cette classe permettent de modéliser les défaillances par une distribution de probabilité exponentielle (hypothèse markovienne). Elle est fortement expressive et permet donc une bonne représentativité des systèmes pour une étude de sûreté de fonctionnement [41].

Toutefois, le modèle, bien que représentable graphiquement, est difficilement lisible pour des tailles importantes. Des logiciels spécialisés permettent une modélisation à plus haut niveau, par assemblage de composants définis dans des bibliothèques spécifiques à un domaine ou une industrie.

ii Simulation

Grâce à l'hypothèse markovienne respectée par ses transitions (distributions de probabilités exponentielles), le graphe d'état d'un réseau de Petri stochastique généralisé correspond à une chaîne de Markov à temps continu, ce qui permet la réalisation de calculs suivant les mêmes méthodes (voir I.2.2.a.iii).

Cependant, la taille de la chaîne de Markov obtenue peut être trop importante pour son calcul, en raison de l'explosion combinatoire. Il est alors possible de limiter cette taille, par exemple en supprimant les états fugaces (qui sont quittés immédiatement à cause d'une transition immédiate) [17].

2.2.c Avantages et limitations des formalismes états-transitions

De façon générale, l'expressivité des formalismes états-transitions permet une bonne représentativité des systèmes modélisés. Ils sont toutefois limités par leur difficulté de modélisation, car ils restent des formalismes de bas-niveau. De plus, en raison de l'explosion com-

binatoire rencontrée soit lors de leur modélisation (pour les chaînes de Markov), soit lors du calcul de leur espace d'état (pour les réseaux de Petri), leur calcul analytique est très coûteux.

2.3 Limitations des approches centrées sur l'outil

Les approches centrées sur l'outil ont facilité le passage à des méthodes de sûreté de fonctionnement quantitatives avec expériences numériques, en réutilisant les formalismes "historiques" (arbres de défaillances, blocs-diagrammes de fiabilité...). Ces formalismes ont été choisis car ils présentent des propriétés facilitant la partie calculatoire de ces expériences.

Mais ces approches ne permettent d'obtenir des grandeurs de sûreté de fonctionnement que peu représentatives, car :

- Les modèles sont peu représentatifs des systèmes étudiés, l'expressivité des formalismes étant limitée : la modélisation est contrainte par le formalisme et ses limites ;
- Dans le cas de systèmes complexes, la taille des modèles est trop importante pour des calculs exacts, malgré les propriétés des modèles : le postulat de départ n'est pas suffisant pour permettre une simulation simple. Les calculs doivent alors être approximés, en faisant des hypothèses simplificatrices ou en ayant recours à des calculs partiels (simulation stochastique...).

La confiance dans les résultats obtenus par une approche centrée sur l'outil est donc limitée, car le résultat de l'ensemble des approximations (de modélisation et de calcul) n'est pas forcément maitrisé.

3 Approches centrées sur le modèle

À l'opposé des approches centrées sur l'outil, les approches centrées sur le modèle (MBSA : "Model-Based Safety Assessment") considèrent qu'il est d'abord important d'avoir un modèle représentatif du système étudié. Elles cherchent donc à donner à l'analyste le langage dont il a besoin pour concevoir son modèle, sans contraindre la modélisation afin d'obtenir des propriétés particulières. Les difficultés de calculs ne sont pas anticipées : ils peuvent être réalisés en convertissant le modèle vers un formalisme de l'approche centrée sur l'outil, ou par d'autres méthodes de calcul.

En partant de ce même postulat, ces approches diffèrent dans leurs choix de solutions pour la modélisation. Certaines utilisent des modèles fonctionnels (non dédiés à la sûreté de fonctionnement) en ajoutant des informations dysfonctionnelles, d'autres composent des modèles dysfonctionnels exploitant des formalismes existants, et enfin certaines proposent un formalisme de modélisation dédié à la sûreté de fonctionnement.

3.1 Depuis des modèles fonctionnels

Dans la démarche de conception d'un système critique, les études fonctionnelles et dysfonctionnelles sont réalisées en parallèle, les secondes validant les solutions techniques proposées par les premières. Les modèles utilisés par ces deux ingénieries sont historiquement distincts, et ne contiennent pas les mêmes informations.

Comme pour les études de sûreté de fonctionnement, il existe une approche d'ingénierie fonctionnelle centrée sur le modèle (MBSE : "Model Based System Engineering"), pour laquelle l'étude fonctionnelle passe par un modèle représentatif du système à partir duquel

sont réalisées différentes expériences numériques : le modèle n'est pas spécifique à une étude fonctionnelle particulière.

Ces modèles fonctionnels contiennent beaucoup d'informations, dont certaines peuvent être utiles pour réaliser des études dysfonctionnelles.

Les auteurs de Hip-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies) [51] ont constaté que lors de la conception d'un système, plusieurs modèles de sûreté de fonctionnement sont successivement produits (FFA, HAZOP, FMEA...), et que la cohérence de ces modèles n'est pas certaine, ce qui peut mener à des erreurs. Ils proposent une démarche outillée pour assurer cette cohérence entre modèles, grâce à un modèle de la structure du système. Dans ce modèle, les différents composants sont connectés entre eux, et chacun peut être décrit par un modèle d'ingénierie externe.

Le comportement dysfonctionnel peut être précisé pour permettre la génération automatique des différents modèles utilisés par l'étude de sûreté de fonctionnement, qu'ils soient qualitatifs (FMEA [73]...) ou quantitatifs (arbres de défaillances [31], réseaux de Petri [32]...). Le calcul d'indicateurs est alors réalisé en utilisant les méthodes spécifiques aux formalismes des modèles générés.

Cette démarche apporte donc des solutions pour faciliter la création et la gestion des modèles, mais elle se heurte aux mêmes limites de modélisation et de calcul que les formalismes historiques utilisés.

3.2 Avec plusieurs modèles dysfonctionnels

Les formalismes classiques de sûreté de fonctionnement présentent chacun des limitations de leur expressivité. Pour accroitre les possibilités d'expression et d'analyse, des approches industrielles proposent d'utiliser conjointement plusieurs formalismes, par exemple pour décrire des parties différentes d'un même système. Cette solution a pour avantage de réutiliser les compétences en modélisation des analystes de sûreté de fonctionnement, formés à ces outils (arbres de défaillance, blocs-diagrammes...).

L'atelier GRIF (GRaphiques Interactifs pour la Fiabilité) [25] propose plusieurs "packages", dont l'un permet une modélisation d'un système en utilisant des formalismes booléens, et un autre des réseaux de Petri (réseaux de Petri stochastiques à prédicats).

Le "package" booléen permet d'assembler des descriptions par arbre de défaillances, bloc-diagramme de fiabilité, arbre d'évènements ... en un même modèle. Le modèle obtenu par cette démarche "multi approches" [19] est traduit en un équivalent booléen, qui peut être étudié par un moteur de calcul booléen (logiciel Albizia) ou par simulation de Monte-Carlo (logiciel Moca).

Le "package" simulation permet quant à lui de modéliser un système par des réseaux de Petri stochastiques à prédicats, sous forme directement de réseaux de Petri, ou indirectement par des blocs-diagramme stochastiques ou des formalismes spécifiques à l'industrie pétrolière (diagramme pétrolier PFD). Ces derniers sont traduits au moment de la simulation en réseaux de Petri. Les modèles obtenus peuvent être utilisés pour obtenir, par simulation de Monte-Carlo (par le logiciel Moca-RP), différentes mesures de sûreté de fonctionnement ou de performance (productivité) : ils ne sont pas spécifiques à l'étude d'une fonction du système, et peuvent être réutilisés tout au long du cycle de vie du système.

3.3 Avec un formalisme dédié : AltaRica 3.0

L'approche centrée sur le modèle proposée avec AltaRica 3.0 part d'un second postulat [65] : il n'existe pas de formalisme permettant de tout modéliser, chaque point de vue doit utiliser un formalisme et une méthodologie spécifique.

Cette approche propose donc un formalisme dédié aux études de sûreté de fonctionnement probabilistes (AltaRica 3.0), qui doit être suffisamment expressif pour que l'analyste puisse décrire avec précision le système étudié. Pour l'exploiter, l'analyste a à sa disposition un ensemble d'outils d'analyse, suivant l'étude à réaliser et les caractéristiques du modèle : le projet AltaRica 3.0 a pour objectif de fournir un tel ensemble d'outils.

4 AltaRica 3.0

AltaRica 3.0 est un langage de modélisation destiné aux études de sûreté de fonctionnement. La version précédente du langage, AltaRica Data-Flow [13], est une généralisation à la fois des réseaux de Petri et des blocs-diagrammes. Elle est inspirée des réseaux de Petri pour les notions d'état, d'évènements et de transitions gardées, et des blocs-diagrammes pour les notions de synchronisation d'évènements, de description hiérarchique, et de flux. Cette notion de flux permet de représenter simplement des interactions entre différentes parties du modèle. Toutefois, les flux bidirectionnels ne peuvent pas être modélisés simplement, ce qui rend difficile la modélisation de systèmes bouclés.

C'est pourquoi une nouvelle version du langage, AltaRica 3.0, a été spécifiée. Elle présente des améliorations d'AltaRica Data-Flow aussi bien pour la structuration du modèle que pour la modélisation de comportements.

Pour permettre l'analyse des modèles AltaRica 3.0, des outils logiciels sont nécessaires. L'un des objectifs du projet AltaRica 3.0 et du projet OpenAltaRica est de fournir un ensemble d'outils de modélisation et d'analyse des modèles AltaRica 3.0.

4.1 Le langage de modélisation AltaRica 3.0

Le langage de modélisation AltaRica 3.0 est un formalisme de modélisation haut-niveau dédié aux études de sûreté de fonctionnement [53]. La spécification complète de ce langage est disponible [8].

AltaRica 3.0 est un langage orienté prototype : l'architecture des modèles est formalisée par S2ML [9], ce qui permet de construire efficacement la structure du modèle [57]. Le formalisme mathématique, permettant la modélisation des comportements, est celui des systèmes à transitions gardées (GTS), qui sont des automates à états.

4.1.a Automate à états

Un modèle AltaRica 3.0 est un automate à états : l'état courant est porté par les variables d'état, tandis que des transitions permettent de passer d'un état à l'autre. Les transitions sont tirées lorsque l'évènement correspondant survient et si la garde de la transition est vérifiée : leur action est alors exécutée.

La figure I.1 est un modèle AltaRica 3.0 d'un système qui se comporte comme une mémoire booléenne. Sa variable d'état booléenne *vsState* contient son état courant. Lorsque l'évè-

nement *eSwitchOff* occurre, la transition correspondante est tirée : si *vsState* a pour valeur *true* (garde de la transition), alors sa valeur passe à *false* (action de la transition). La figure I.2 représente le graphe des états accessibles (restreint à la variable d'état) de ce modèle.

```
1  block System
2    // le système une variable d'état booléenne
3    Boolean vsState (init = true);
4
5    // deux évènements permettent de basculer la valeur de la variable d'état
6    event eSwitchOff (delay = 1);
7    event eSwitchOn (delay = 1);
8    transition
9      // Syntaxe des transitions :
10     // évènement : garde -> actions ;
11     eSwitchOff: vsState -> vsState := false;
12     eSwitchOn: not vsState -> vsState := true;
13 end
```

FIGURE I.1 – Modèle AltaRica 3.0 d'un système à deux états

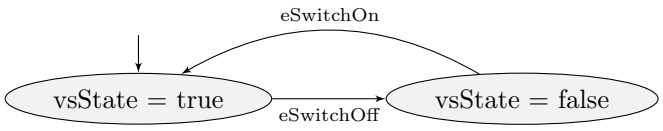

FIGURE I.2 – Graphe des états accessibles du modèle AltaRica 3.0 figure I.1

Les variables d'état peuvent être de différents types : booléens, nombres entiers et réels, ou encore appartenir à un domaine énuméré défini par l'utilisateur. Il est aussi possible de définir des n-uplets de variables de différents types (appelés *record*), semblables aux structures proposées en C/C++.

4.1.b Automate à états stochastique et temporisé

Les évènements des modèles AltaRica 3.0 sont temporisés : ils occurrent à des dates, déterminées par leurs délais. Les délais courent à partir de l'instant auquel la garde de la transition associée est vérifiée. Ils peuvent être constants (comme ceux du modèle figure I.1, qui occurrent alternativement toutes les 1 unité de temps), mais peuvent aussi être déterminés de façon stochastique.

i Délais stochastiques

Le modèle AltaRica 3.0 de la figure I.3 décrit un système qui peut défaillir et être réparé. Ce type de système est connu dans le domaine de la sûreté de fonctionnement comme étant un "système Lambda-Mu" : ses évènements occurrent après des délais qui suivent deux distributions de probabilités exponentielles (c'est-à-dire à taux constants) de paramètres *lambda* et *mu*, exprimés en occurrences par unités de temps.

Plusieurs distributions de probabilités sont disponibles dans le langage AltaRica 3.0 : exponentielle, Weibull... Le modélisateur peut aussi définir lui-même des lois de probabilité spécifiques [11].

```
block System
  Boolean vsState (init = true);

  parameter Real lambda = 0.0001; // taux de défaillance
  parameter Real mu = 0.01; // taux de réparation
  // évènement de défaillance
  event eFailure (delay = exponential(lambda));
  // évènement de réparation
  event eRepair (delay = exponential(mu));

  transition
    eFailure: vsState -> vsState := false;
    eRepair: not vsState -> vsState := true;
end
```

FIGURE I.3 – Modèle AltaRica 3.0 d'un système à défaillance et réparation ("lambda-mu")

```
block System
  Boolean vsState (init = true);
  Boolean vsStop (init = false);

  // deux évènements permettent de basculer la valeur de la variable d'état
  event eSwitchOff (delay = 1);
  event eSwitchOn (delay = 1);
  // un évènement avec un délai trop long pour survenir
  event eStop (delay = 2);
  transition
    eSwitchOff: vsState and not vsStop -> vsState := false;
    eSwitchOn: not vsState and not vsStop -> vsState := true;
    eStop: not vsState and not vsStop -> vsStop := true;
end
```

FIGURE I.4 – Modèle AltaRica 3.0 d'un système avec une transition qui ne sera jamais tirée

ii Décompte du délai

Lorsque la garde d'une transition devient vérifiée le délai de l'évènement correspondant est déterminé, au besoin à l'aide d'un nombre aléatoire s'il est stochastique. Ce délai décroit alors avec l'avancement du temps de la simulation : l'évènement correspondant survient, et la transition est tirée, lorsque le délai arrive à zéro. Si entre-temps la garde n'est plus vérifiée (suite au tir d'autres transitions), le délai ne décroit plus. Lorsque la garde redevient vérifiée, le délai est de nouveau déterminé : il est soit remis à sa valeur (dans le cas d'un délai non stochastique), soit de nouveau choisi à l'aide d'un nombre aléatoire (s'il est stochastique).

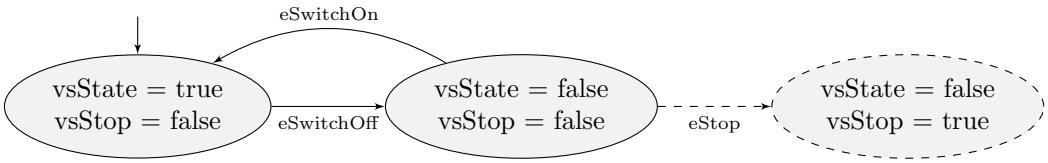

FIGURE I.5 – Graphe des états accessibles du modèle AltaRica 3.0 figure I.4

Ainsi, dans le cas du modèle de la figure I.4, l'évènement *eStop* qui a un délai de 2 unités de temps ne peut survenir que depuis le même état que l'évènement *eSwitchOn* (les gardes des transitions correspondantes sont identiques). Seulement, le second évènement a un délai plus court que le premier : l'évènement *eSwitchOn* occurrera donc toujours avant que *eStop* ai pu survenir. *eStop* ne surviendra donc jamais, et l'état dans lequel *vsStop* est vraie ne sera jamais atteint (figure I.5).

Cette "remise à zéro" du délai lorsque la garde n'est plus vérifiée est le comportement par défaut, mais peut être empêchée en paramétrant l'évènement (*policy* = *memory*) : le délai décroit toujours lorsque la garde est vérifiée, mais il n'est alors déterminé à nouveau que lorsque l'évènement est effectivement survenu. Cette primitive du langage permet de modéliser des phénomènes d'usure ou de consommation : par exemple, une plaquette de frein peut être utilisée pendant un certain temps avant de défaillir, ne plus freiner ne remet pas la plaquette à neuf.

iii Évènements simultanés

Les délais définis par des distributions de probabilités rendent les modèles AltaRica 3.0 stochastiques. L'autre aspect stochastique des modèles AltaRica 3.0 est rencontré lorsque deux évènements occurrent à la même date : l'ordre du tir des transitions correspondantes est alors déterminé aléatoirement, en prenant en compte un poids dans le choix.

Ce type de cas est par exemple rencontré lors de la modélisation de panne au démarrage d'un système. Au moment du démarrage, deux comportements sont possibles : soit le système démarre, soit il défaille. Ceci peut être modélisé en AltaRica 3.0 par deux évènements de même date d'occurrence : la première transition à être tirée va être déterminée stochastiquement.

Dans l'exemple figure I.6, qui est un modèle d'un interrupteur, la transition de démarrage sans défaillance met la variable d'état représentant la position de l'interrupteur à *true*, rendant impossible le tir de l'autre transition (à cause de sa garde). Inversement, le tir de la défaillance au démarrage rend le tir de la transition de démarrage impossible. Une seule des deux transitions sera donc tirée. Ce choix se fera en prenant en compte le paramètre *expectation* de l'évènement : la probabilité de tirer la défaillance au démarrage est de *gamma*.

Dans le cas ou deux évènements simultanés ne seraient pas associés à des transitions exclusives, la première à être tirée serait choisie de la même manière. La seconde, toujours tirable et associée à un évènement dont le délai serait alors nul, serait tirée immédiatement après.

iv Notion d'état

Les modèles AltaRica 3.0 sont des descriptions d'automates à états temporisés. Formellement, l'état de l'automate comprend les valeurs des variables d'état, ainsi que les valeurs des différents délais (que nous appellerons ensuite l'échéancier).

```
block System
  Boolean vsState (init = true);
  Boolean vsSwitch (init = false);

  // taux de défaillance au démarrage
  parameter Real gamma = 0.01;
  // évènement de démarrage sans défaillance
  event eStart (delay = Dirac(0), expectation = 1 - gamma);
  // évènement de défaillance au démarrage
  event eFailureOnDemand (delay = Dirac(0), expectation = gamma);

  transition
    eStart: vsState and not vsSwitch -> vsSwitch := true;
    eFailureOnDemand: vsState and not vsSwitch -> vsState := false;
end
```

FIGURE I.6 – Modèle AltaRica 3.0 d'un système à démarrage et panne au démarrage

Par simplicité, les graphes des états accessibles représentés dans ce document ne différencient les états que par les valeurs des variables d'états, sans tenir compte des valeurs des différents délais.

4.1.c Architecture d'un système

Un système peut être décrit comme étant un ensemble de composants assemblés suivant une architecture. Dans un modèle AltaRica 3.0, le comportement de chaque composant est décrit par un automate à états temporisé et stochastique, tandis que l'architecture du système est décrite à l'aide de constructions issues de S2ML.

Une fois l'architecture décrite, les comportements des différents composants doivent être synchronisés. Deux mécanismes sont disponibles dans AltaRica 3.0 : la synchronisation des évènements, et les variables de flux avec les assertions.

i S2ML et orientation prototype

AltaRica 3.0 utilise S2ML pour décrire l'architecture d'un système, ce qui en fait un langage orienté prototype. Il est ainsi possible de définir des modèles de composants, qui pourront ensuite être instanciés plusieurs fois dans un même modèle de système ou de sous-système (mécanisme de composition).

Le système modélisé figure I.7 possède deux composants réparables ("Lambda-Mu"). Au lieu de décrire deux fois le comportement d'un composant réparable, un modèle est créé (mot-clé *class*). Il est ensuite instancié une fois pour chaque composant dans la description du système (qui n'est pas une classe, mais un bloc, comme l'indique le mot-clé *block*) : le système est donc composé de deux composants réparables. Il est possible de modifier à l'instanciation des valeurs de paramètres : ici, le second composant aura un taux de défaillance différent du premier.

D'autres constructions de l'architecture du modèle AltaRica 3.0 sont possibles :

- une classe ou un bloc peut être défini en étendant une autre classe (mécanisme d'héritage) ;

```
class RepairableComponent // description d'un composant réparable
  Boolean vsState (init = true);

  parameter Real lambda = 0.0001; // taux de défaillance
  parameter Real mu = 0.01; // taux de réparation
  event eFailure (delay = exponential(lambda)); // évènement de défaillance
  event eRepair (delay = exponential(mu)); // évènement de réparation

  transition
    eFailure: vsState -> vsState := false;
    eRepair: not vsState -> vsState := true;
end

block System // description du système
  // le système a deux composants réparables :
  // Component1 et Component2
  RepairableComponent Component1;
  RepairableComponent Component2 (lambda = 0.04);
  // Component2 sera instancié avec un taux de défaillance lambda différent
end
```

FIGURE I.7 – Modèle AltaRica 3.0 d'un système avec deux composants réparables

- un bloc ou une instance de classe peut faire partie de plusieurs blocs simultanément (mécanisme d'agrégation) ;
- un bloc peut être défini en clonant un autre bloc (mécanisme de clonage).

Ces différents mécanismes de construction permettent de décrire de façon efficace l'architecture d'un modèle AltaRica 3.0 d'un système complexe. Ils rendent aussi possible la construction de bibliothèques de modèles de composants et systèmes génériques (via les classes et les paramètres), à instancier ou étendre lors de la modélisation d'une solution technique : le gain de temps lors de la modélisation de plusieurs modèles proches peut alors être important.

ii Synchronisation des évènements

Pour synchroniser le comportement de plusieurs composants, il est possible de synchroniser leurs évènements.

Un cas d'utilisation classique est la survenance d'une défaillance de cause commune : un évènement (un incendie, une surtension électrique...) cause la défaillance de plusieurs composants simultanément. Ainsi, dans le modèle de la figure I.8, l'évènement *eCCF* est synchronisé avec les défaillances des deux composants.

La synchronisation avec chaque défaillance est optionnelle (indiqué par le " *?*") : un composant qui est déjà défaillant n'empêche pas la défaillance de cause commune de survenir. Seules les transitions synchronisées tirables (dont les gardes sont vérifiées) seront tirées. Il est possible de rendre les synchronisations obligatoires (en remplaçant les " *?*" par des " *!*") : les transitions synchronisées devront obligatoirement être tirables pour que l'évènement synchronisé survienne.

Les évènements correspondant aux transitions synchronisées (ici, les défaillances des composants) peuvent toujours survenir, indépendamment de la transition. Il est possible de les empêcher de survenir seuls à l'aide du mot-clé *hidden*.

```
class RepairableComponent
  Boolean vsState (init = true);

  parameter Real lambda = 0.0001;
  parameter Real mu = 0.01;
  event eFailure (delay = exponential(lambda));
  event eRepair (delay = exponential(mu));

  transition
    eFailure: vsState -> vsState := false;
    eRepair: not vsState -> vsState := true;
end

block System
  RepairableComponent Component1;
  RepairableComponent Component2;
  // évènement de la défaillance de cause commune
  event eCCF (delay = exponential(0.002));
  transition
    // synchronisation des défaillances des composants
    // avec la défaillance de cause commune
    eCCF: ?Component1.eFailure & ?Component1.eFailure
end
```

FIGURE I.8 – Modèle AltaRica 3.0 d'un système avec deux composants réparables et une défaillance de cause commune

iii Variables de flux et assertions

Un bloc peut être composé de variables de flux, qui peuvent avoir les mêmes types que les variables d'état (booléen, entiers, réels, énumérés). Celles-ci peuvent être utilisées dans les gardes des transitions, et permettent donc de synchroniser les comportements de plusieurs composants.

Les transitions ne peuvent en revanche pas, dans leur action, leur affecter de valeur. La valeur des variables de flux est calculée après chaque tir de transition, à partir des assertions. Les variables de flux sont des variables "sans mémoire", contrairement aux variables d'état, et leurs valeurs ne dépendent que des valeurs actuelles des autres variables de flux et des variables d'état.

Le modèle AltaRica 3.0 de la figure I.9 est celui d'un système avec deux composants réparables ("Lambda-Mu") qui possèdent chacun deux variables de flux : une d'entrée (*vfIn*) et une de sortie (*vfOut*). Leurs assertions indiquent que la valeur de la variable de sortie est égale à la valeur de la variable d'entrée, à la condition que le composant soit en état de bon fonctionnement. La garde de la transition de défaillance a été modifiée par rapport aux exemples précédents, pour que le composant ne puisse défaillir que lorsqu'il est effectivement utilisé, c'est-à-dire lorsqu'il est en bon état et que la variable d'entrée est à la valeur *true*.

Le système possède deux composants, reliés en série : la sortie du premier composant est envoyée à l'entrée du second composant, comme décrit par les assertions du bloc qui décrit le système. L'entrée du système est envoyée au premier composant, et la sortie du second composant est reliée à la sortie du système.

Le graphe des états accessibles de ce modèle, figure I.10, indique en plus des valeurs des variables d'états les valeurs de deux variables de flux. On peut ainsi remarquer que dans l'état où le premier composant est défaillant et le second en bon fonctionnement, ce dernier voit sa variable de flux d'entrée à la valeur *false* : il ne peut donc pas défaillir.

```
class RepairableComponent
  Boolean vsState (init = true);

  parameter Real lambda = 0.0001;
  parameter Real mu = 0.01;
  event eFailure (delay = exponential(lambda));
  event eRepair (delay = exponential(mu));

  Boolean vfIn (reset = false); // déclaration de deux variables de flux
  Boolean vfOut (reset = false); // à l'aide du mot-clé "reset"
  transition
    // Pas de défaillance si le composant n'est pas utilisé
    eFailure: vsState and vfIn -> vsState := false;
    eRepair: not vsState -> vsState := true;

  assertion
    // Si le composant n'est pas défaillant, alors le flux en sortie est é
    gal au flux en entrée
    if vsState then vfOut := vfIn;
end

block System
  RepairableComponent Component1;
  RepairableComponent Component2;

  Boolean vfStart (reset = false);
  Boolean vfEnd (reset = false);
  assertion
    vfStart := true; // valeur de départ
    // Association en série des composants :
    // vfStart --> Component1 --> Component2 --> vfEnd
    Component1.vfIn := vfStart;
    Component2.vfIn := Component1.vfOut;
    vfEnd := Component2.vfOut;
end
```

FIGURE I.9 – Modèle AltaRica 3.0 d'un système avec deux composants réparables et synchronisation par variables de flux

Le calcul des valeurs des variables de flux après chaque tir de transition est réalisé en propageant les assertions, par le mécanisme dit du point fixe (voir A.2.2).

Le principe de ce mécanisme est une application itérative des assertions jusqu'à un point fixe dans lequel aucune assertion ne peut plus être appliquée. Les variables de flux non encore affectées prennent alors leur valeur par défaut (configurée par le mot-clé *reset*), et une vérification de la cohérence de l'ensemble des valeurs est menée.

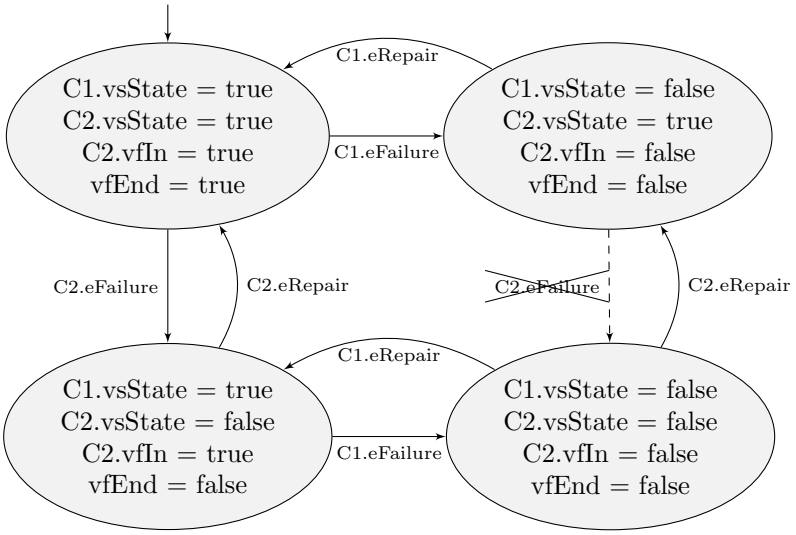

FIGURE I.10 – Graphe des états accessibles du modèle AltaRica 3.0 figure I.9

Il est possible de définir des relations bidirectionnelles entre variables de flux : A dépend de B, et B dépend de A. Les directions des flux sont alors déterminées après chaque tir de transition par le mécanisme du point fixe, et dépendent des valeurs des variables d'état : elles peuvent donc changer au cours d'une même simulation.

Cette possibilité, qui est une évolution majeure du langage par rapport à sa version précédente, permet de modéliser des composants de façon acausale (sans entrée ou sortie définie). Un exemple d'utilisation est le cas de tuyaux formant un réseau de distribution de fluide ou d'énergie : un câble électrique peut être utilisé pour transmettre de l'énergie dans un sens ou dans l'autre. Sa modélisation est donc grandement facilitée par cette possibilité, au prix d'un calcul plus coûteux lors de la simulation du modèle.

La mise à plat de ce mécanisme, c'est-à-dire le calcul exhaustif des valeurs des variables de flux en fonction des valeurs des variables d'état, est dans le cas général prohibitif. Il est donc difficile de compiler, sans perte d'information, un modèle AltaRica 3.0 dans un formalisme d'automates à états dépourvu de mécanisme du point fixe.

4.1.d Observateurs

Les observateurs AltaRica 3.0 permettent au modélisateur d'indiquer quelles grandeurs ou quantités du modèle sont possiblement intéressantes pour l'étude de sûreté de fonctionnement du modèle. Ils ne sont pas utilisables dans les transitions ou les assertions : ils ne sont destinés qu'aux outils d'analyse de modèles, et donc à l'analyste.

Techniquement, les observateurs AltaRica 3.0 sont des variables (booléennes, entières, réelles ou énumérées), dont la valeur est définie par une expression calculée après chaque tir de transition et après calcul des valeurs des variables de flux : ils n'ont pas de mémoire (comme les variables de flux). Cette expression dépend des variables d'état et de flux du modèle.

Dans le modèle de la figure I.9, une grandeur intéressante pour une étude de sûreté de fonctionnement est la valeur en sortie du système, c'est-à-dire la valeur de la variable de flux

de sortie du bloc modélisant le système : elle correspond au bon fonctionnement ou non du système complet. Un observateur peut alors être défini en ajoutant, dans la description du bloc :

```
24  // définition de l'observateur
25  observer Boolean oOut = vfEnd;
```

4.1.e Mise à plat d'un modèle AltaRica 3.0

L'architecture est une information utile pour la modélisation, mais est superflue pour l'analyse du comportement d'un modèle : avant d'être étudié, un modèle AltaRica 3.0 est mis à plat.

Un modèle AltaRica 3.0 est structuré en plusieurs composants. Chaque composant est un GTS qui échange des informations ou est synchronisé avec d'autres composants. Le résultat de la mise à plat d'un modèle AltaRica 3.0 est un unique GTS.

i Systèmes de transitions gardées (GTS)

Les systèmes de transitions gardées [62] sont conçus comme un formalisme pivot [56] pour la modélisation et les analyses de sûreté de fonctionnement. Il généralise les formalismes classiques, comme les blocs-diagramme, les chaînes de Markov et les réseaux de Petri. Une définition synthétique est disponible dans l'annexe A.

Ce sont des modèles de machines à état temporisés et stochastiques : les transitions sont associées à des évènements, dont les délais d'occurrence sont déterminés par des distributions de probabilité (sans restriction sur les distributions de probabilités). Ils permettent la description de modèles de systèmes avec des boucles instantanées, et la définition de composants acausaux. Les systèmes avec des flux bidirectionnels (systèmes électriques, réseaux...) sont ainsi facilement modélisables.

ii Mise à plat de l'architecture

Certains mécanismes présents dans le langage de modélisation AltaRica 3.0 n'existent pas dans les GTS, et doivent donc être remplacés par des mécanismes équivalents pour obtenir un unique GTS : c'est le cas des mécanismes apportés par S2ML, comme la composition. La mise à plat de l'architecture consiste à synthétiser tous les automates à états décrits dans le modèle AltaRica 3.0, en supprimant l'architecture mais en conservant les comportements décrits par les synchronisations.

```
block System
  Boolean Component1.vsState (init = true);
  Boolean Component2.vsState (init = true);

  event Component1.eFailure (delay = exponential(0.0001));
  event Component1.eRepair (delay = exponential(0.01));
  event Component2.eFailure (delay = exponential(0.0001));
  event Component2.eRepair (delay = exponential(0.01));

  Boolean vfStart (reset = false);
  Boolean vfEnd (reset = false);
  Boolean Component1.vfIn (reset = false);
  Boolean Component1.vfOut (reset = false);
  Boolean Component2.vfIn (reset = false);
  Boolean Component2.vfOut (reset = false);

  observer Boolean oOut = vfEnd;
  transition
    Component1.eFailure: Component1.vsState and Component1.vfIn ->
    Component1.vsState := false;
    Component1.eRepair: not Component1.vsState -> Component1.vsState :=
    true;
    Component2.eFailure: Component2.vsState and Component2.vfIn ->
    Component2.vsState := false;
    Component2.eRepair: not Component2.vsState -> Component2.vsState :=
    true;
  assertion
    vfStart := true;
    Component1.vfIn := vfStart;
    if Component1.vsState then Component1.vfOut := Component1.vfIn;
    Component2.vfIn := Component1.vfOut;
    if Component2.vsState then Component2.vfOut := Component2.vfIn;
    vfEnd := Component2.vfOut;
end
```

FIGURE I.11 – GTS obtenu par mise à plat du modèle AltaRica 3.0 figure I.9

Par exemple, la figure I.11 présente le GTS obtenu par mise à plat du modèle AltaRica 3.0 présenté figure I.9. On peut remarquer que les instanciations des composants réparables ont été substituées par les GTS correspondants, et que toutes les variables, assertions, transitions et tous les évènements et observateurs sont au niveau du bloc (et non plus dans un composant du bloc). Le comportement du modèle AltaRica 3.0 et de ce GTS sont strictement identiques : seule l'information sur l'architecture (c'est-à-dire à quel composant "appartient" chaque évènement ou variable) est supprimée.

iii Simplifications lors de la mise à plat

La mise à plat d'un modèle AltaRica 3.0 en un unique GTS peut être l'occasion de simplifier le modèle, par exemple en supprimant des variables de flux qui ne font que "recopier" des valeurs de variables d'état ou en réorganisant les assertions.

La spécification de ces opérations de simplification garantit uniquement que le comportement vu au travers des observateurs sera conservé : les études du comportement d'un modèle

AltaRica 3.0 simplifié et non simplifié donneront les mêmes résultats, car les deux se feront au travers des observateurs.

4.2 Le projet AltaRica 3.0

Le projet AltaRica 3.0 [54] vise à proposer un ensemble cohérent d'outils destinés aux études de sûreté de fonctionnement. Ces outils sont regroupés en quatre ateliers :

- un atelier de gestion des modèles de sûreté de fonctionnement, permettant la comparaison et la synchronisation de divers modèles d'un même système ou de différentes solutions techniques ;
- un atelier d'interfaçage avec des modèles Open-PSA. Open-PSA [68] est un format d'échange de modèles de sûreté de fonctionnement : son objectif est de permettre l'étude d'un même modèle par différents outils de calcul acceptant des formats d'entrée différents.
- un atelier d'édition et d'animation graphique de modèles, GraphXica [55]. L'utilisation d'outils graphiques de représentation d'un modèle facilite leur compréhension et la communication des résultats des études de sûreté de fonctionnement.
- un atelier d'édition et d'étude de modèles AltaRica 3.0, dans lequel les travaux présentés dans la suite de ce document s'inscrivent.

L'atelier AltaRica 3.0 propose un ensemble d'outils de calcul pour réaliser des études de sûreté de fonctionnement. Tous ces outils travaillent sur des modèles AltaRica 3.0 : l'analyste a ainsi à sa disposition plusieurs outils lui permettant de choisir une stratégie de traitement adaptée à son besoin et ses contraintes.
Ces outils permettent :

- la simulation du modèle AltaRica 3.0 (simulateur pas-à-pas, simulateur stochastique)
- la compilation vers un autre formalisme, afin de l'étudier avec un outil dédié à ce formalisme (compilateurs vers les arbres de défaillances et les chaînes de Markov), sous certaines conditions sur le modèle.

4.2.a Simulateur pas-à-pas

Le simulateur pas à pas permet de simuler un modèle AltaRica 3.0, sans tenir compte des distributions de probabilité sur les délais. Les informations de temporisation du GTS ne sont pas prises en compte. L'analyste a le choix de la transition à tirer parmi celles qui sont tirables.

Cet outil ne permet pas de calculer des indicateurs de sûreté de fonctionnement, mais permet au modélisateur de tester ses modèles. Cette étape est nécessaire car elle permet à l'analyste de corriger son modèle avant de l'exploiter avec un autre outil de calcul pour évaluer des grandeurs de sûreté de fonctionnement.

4.2.b Compilateur vers les arbres de défaillances

Le compilateur vers les arbres de défaillances [53] permet de réaliser des études par arbres de défaillances d'un modèle AltaRica 3.0. Le modèle est compilé en un arbre de défaillances au format Open-PSA, et est ensuite simulable à l'aide d'un outil de calcul spécialisé tel que XFTA [64].

L'évènement redouté est un observateur AltaRica 3.0 : il est donc possible de réaliser des études par arbres de défaillances portant sur des évènements redoutés différents, à partir d'un même modèle AltaRica 3.0.

Pour que les résultats obtenus soient significatifs, les modèles AltaRica 3.0 utilisés doivent présenter des propriétés proches de celles des arbres de défaillances : par exemple, les évènements doivent être indépendants, sinon les grandeurs obtenues par ces études peuvent ne pas être représentatives. C'est également le cas des modèles AltaRica 3.0 avec des cycles (par exemple des composants pouvant être réparés), car les études par arbres de défaillances ne prennent pas en compte cet aspect des modèles.

4.2.c Compilateur vers les chaînes de Markov

De la même manière que pour les arbres de défaillances, le compilateur vers les chaînes de Markov [15] permet de réaliser des études par chaînes de Markov d'un modèle AltaRica 3.0.

Pour que l'étude soit représentative, le modèle AltaRica 3.0 doit présenter les mêmes propriétés mathématiques que celles attendues d'une chaîne de Markov : les transitions doivent être soit immédiates, soit être associées à des distributions de probabilités exponentielles, afin de respecter l'hypothèse markovienne.

4.2.d Simulateur stochastique

Les compilateurs vers les arbres de défaillances et vers les chaînes de Markov permettent d'évaluer des grandeurs de sûreté de fonctionnement, à la condition que le modèle étudié présente des propriétés particulières. Une solution pour l'étude de modèles sans propriétés particulières est l'utilisation de la simulation stochastique.

Les travaux présentés dans la suite de ce document concernent la spécification et le développement d'un outil de simulation stochastique de modèles AltaRica 3.0.

4.3 Le projet OpenAltaRica

Les travaux présentés dans ce document ont été réalisés au sein du projet OpenAltaRica [71], qui est un projet de recherche piloté par l'IRT SystemX, en partenariat industriel avec Apsys, Thales et Safran. Ses objectifs sont :

- améliorer l'efficacité des processus de modélisation dans le domaine de la sûreté de fonctionnement, et faciliter l'intégration de la sûreté de fonctionnement avec les autres activités d'ingénierie des systèmes complexes. Le chapitre IV est un cas d'étude de la modélisation d'un système mécatronique, prenant en compte aussi bien la partie matérielle que la partie logicielle pour mener l'étude de sûreté de fonctionnement.
- travailler à la mise en œuvre de l'approche dirigée par les modèles (MBSA) pour la certification des systèmes critiques. Ainsi, le chapitre V propose une méthodologie pour évaluer le bon fonctionnement d'un outil de simulation stochastique dans un contexte de certification d'un système critique.

Pour atteindre ses objectifs, le projet OpenAltaRica propose une plateforme logicielle qui est une implémentation de référence des outils et méthodes d'études de sûreté de fonctionnement basés sur AltaRica 3.0. Le simulateur stochastique de modèles AltaRica 3.0 dont la

spécification (chapitre II) et l'implémentation (chapitre III) sont présentées dans ce document est l'un des composants de la plateforme OpenAltaRica (téléchargeable depuis [71]).

5 Simulation stochastique

Lorsque le modèle ne présente pas de propriétés mathématiques facilitant les calculs, ou lorsqu'il est de taille trop importante, un calcul exhaustif n'est pas possible. La simulation stochastique permet d'obtenir des estimations de grandeurs sur des modèles, quels que soient les modèles et leur taille, tant qu'ils sont simulables (au sens interprétables). Sa limitation majeure est son coût en temps de calcul, qui peut être prohibitif.

Cette méthode a déjà été expérimentée pour réaliser des études de sûreté de fonctionnement. On s'intéresse particulièrement à la simulation stochastique de modèles de sûreté de fonctionnement états-transitions (réseaux de Petri stochastiques généralisés, chaînes de Markov, GTS...), donc à la simulation de machines à états stochastiques et temporisés.

Après avoir décrit le principe de la simulation stochastique (I.5.1), son utilisation pour des études de sûreté de fonctionnement sera décrite (I.5.2), ainsi que différents travaux cherchant à améliorer cette méthode (I.5.3).

5.1 Principe de la simulation stochastique

Le principe de la simulation stochastique est de réaliser un grand nombre d'expériences identiques. Le caractère stochastique du modèle va produire, pour chaque expérience, des résultats différents [1]. Une étude probabiliste sur cet ensemble de résultats va permettre d'obtenir une estimation du comportement moyen du modèle : cette estimation devrait être proche de la valeur réelle, d'après la loi des grands nombres.

Loi des grands nombres (théorème de Markov)

Pour des observations indépendantes et non identiquement distribuées [76] :

Soit $\{Z_t\}$ une suite de variables aléatoires indépendantes d'espérance finie $\mu_t \equiv \mathbb{E}(Z_t)$. Si :

$$\exists \delta > 0 \text{ tel que } \sum_{t=1}^{\infty} \frac{\mathbb{E}|Z_t - \mu_t|^{1+\delta}}{t^{1+\delta}} < \infty$$

Alors :

$$\bar{Z}_t - \bar{\mu_t} \overset{\text{presque sûrement}}{\longrightarrow} 0$$

Ainsi, la moyenne (dite empirique) d'un échantillon de variables aléatoires (les résultats de chaque expérience) tend vers l'espérance de leur loi de probabilité (la valeur réelle) lorsque la taille de l'échantillon augmente. L'écart entre l'estimation et la valeur réelle sera ainsi probabilistiquement plus faible avec de plus grands échantillons.

1. On confondra dans le cadre de ce travail la simulation stochastique avec les méthodes de Monte-Carlo : les modèles étudiés sont stochastiques, l'aléatoire est présent à la fois dans le modèle simulé et dans la méthode de simulation.

5.1.a Simulation stochastique d'un automate à états

La simulation stochastique d'un automate à états stochastique et temporisé consiste à générer des *histoires* possibles pour cet automate. Une histoire est une suite d'états, depuis un état initial jusqu'à un état final, séparés par des transitions datées. En raison de l'aspect stochastique du modèle, chaque histoire simulée va être différente. C'est grâce à cette variation des histoires qu'il sera possible d'estimer les indicateurs recherchés.

Algorithme 1 : Simulation stochastique d'un automate à états

```
1 pour n simulations faire
2 |   Générer une histoire en simulant l'automate à état;
3 fin
4 Étudier les histoires générées pour estimer les indicateurs recherchés;
```

i Simulation d'un automate à états temporisé

Un automate à états temporisé est un système à évènements discrets : l'état du modèle évolue lorsqu'un évènement se produit. Ces évènements sont donc à l'origine de transitions qui permettent de passer d'un état à un autre.

L'algorithme de simulation d'un tel automate peut alors se résumer à :

Algorithme 2 : Simulation d'un automate à états temporisé

```
1 Affecter les valeurs initiales aux variables;
2 tant que la condition d'arrêt n'est pas vérifiée faire
3 |   Mettre à jour l'échéancier des évènements tirables;
4 |   Trouver le prochain évènement programmé;
5 |   Avancer le temps jusqu'au prochain évènement programmé;
6 |   Modifier les valeurs des variables suivant la transition;
7 fin
```

L'exécution de cet algorithme permet d'obtenir une histoire, c'est-à-dire une suite d'états séparés par des évènements (avec leur date d'occurrence) représentant l'évolution de l'état de l'automate au cours de la simulation.

ii Simulation d'un automate à états stochastique et temporisé

Pour les automates à états stochastiques et temporisés, les évènements surviennent à des dates aléatoires en suivant des distributions de probabilité. L'algorithme de simulation de ces automates est donc identique, à l'étape de mise à jour de l'échéancier près : les évènements seront ajoutés à l'échéancier en choisissant au hasard leur date d'occurrence. Deux simulations du même automate donneront donc deux histoires différentes. La simulation stochastique de cet automate va permettre de générer un ensemble d'histoires qui permettront alors d'estimer les indicateurs recherchés.

5.1.b Génération de nombres aléatoires

La simulation stochastique repose sur la génération d'une quantité importante de nombres aléatoires, car les choix des transitions de l'automate à tirer et leurs dates de tirs sont déterminés à l'aide de nombres aléatoires.

La génération de nombres aléatoires par logiciel (déterministe) est un problème connu : il ne peut exister que des générateurs de nombres pseudo-aléatoires. Une littérature abondante existe sur ce sujet [49], et de nombreux générateurs de nombres pseudo-aléatoires de qualité sont disponibles : ce thème ne sera pas développé dans ce document.

5.1.c Exploration partielle de l'espace d'état

La simulation stochastique ne réalise pas une exploration exhaustive de l'espace d'état de l'automate à états. Chaque histoire va correspondre à seulement un *chemin* dans l'espace d'état. L'ensemble des histoires ne va donc couvrir qu'une partie de l'espace d'état accessible.

Cette exploration partielle est un avantage pour sa performance et son utilisabilité : l'espace d'état n'est pas construit de façon complète (comme pour la simulation par chaîne de Markov), ni même exploré entièrement de façon symbolique. Des modèles complexes peuvent ainsi être simulés stochastiquement sans moyens de calcul importants, puisqu'une partie seulement de l'espace d'état est parcourue.

En contrepartie, les comportements représentés dans la partie non explorée ne seront pas visibles. Comme le choix des chemins parcourus repose sur leurs probabilités d'occurrence, il devient difficile d'observer des comportements "rares", c'est-à-dire avec une très faible probabilité d'occurrence.

5.1.d Estimation des valeurs et erreurs

Les comportements rencontrés lors d'une simulation stochastique ne correspondant qu'à une partie de l'espace d'état accessible, les mesures ne sont pas complètes. Ainsi, il n'est pas possible de connaître précisément une grandeur : on ne peut que l'estimer.

Dans le cas d'une grandeur probabiliste (comme le sont les grandeurs de sûreté de fonctionnement), on cherchera à estimer les paramètres de sa distribution de probabilité, par exemple ceux de la loi de Bernoulli (II.3.3.a) pour les distributions discrètes de probabilité (par exemple pour des valeurs booléennes), ou de la loi normale pour les distributions continues (II.3.3.b).

La grandeur étudiée est inconnue, on souhaite en obtenir une estimation : la simulation stochastique donne une valeur estimée de cette grandeur, ainsi qu'un intervalle de confiance concernant cette estimation. La valeur estimée correspond à la valeur qu'aurait la grandeur si le tirage était parfait et complet, c'est-à-dire que toutes les possibilités avaient été mesurées en respectant leurs probabilités. Comme le tirage n'est ni parfait ni complet, la véritable valeur de la grandeur est différente de cette valeur estimée. Elle en est toutefois probablement à une faible distance : l'intervalle de confiance permet de borner cette distance, à un degré de confiance près.

Un résultat typique de simulation stochastique sera donc composé de deux valeurs :

- La valeur estimée : X
- L'intervalle de confiance à $n\%$: $[X^n_{min}; X^n_{max}]$

Ce résultat pourra être lu : "Il y a $n\%$ de probabilité que la valeur soit comprise entre X^n_{min} et X^n_{max}".

En accord avec la loi des grands nombres, l'intervalle de confiance décroit avec la taille de l'échantillon. Le degré de confiance peut être choisi arbitrairement : il est classiquement de 95% ou de 99%. Un degré de confiance de 99% implique qu'une estimation sur 100 (le 1% restant) sera erronée : la valeur réelle de la grandeur sera en dehors de l'intervalle de confiance.

5.2 Utilisation de la simulation stochastique pour la sûreté de fonctionnement

La simulation stochastique présente de nombreux avantages pour des analyses de sûreté de fonctionnement [79]. Le plus important est l'absence de contrainte sur le modèle (dans le cas de modèles de type automate à état) : tous les comportements peuvent être simulés, ce qui n'est pas le cas des simulations utilisant les chaînes de Markov ou les arbres de défaillances. Ainsi, les conséquences de redondances ou de politiques de maintenance sur des indicateurs de sûreté de fonctionnement peuvent être calculées par cette méthode.

Pour la même raison, il est possible de calculer des indicateurs de sûreté de fonctionnement complexes, du moment qu'ils peuvent être calculés expérience par expérience. L'estimation de leur valeur ne dépendra alors pas de leur complexité.

Les caractéristiques propres à la simulation stochastique génèrent toutefois des inconvénients pour son utilisation pour une étude de sûreté de fonctionnement. On peut citer le risque d'erreur et le traitement des évènements rares.

5.2.a Risque d'erreur

L'estimation des valeurs, avec son intervalle de confiance, comporte un risque d'erreur. Ce risque est connu, et peut être maîtrisé : l'intervalle de confiance est calculé pour un risque d'erreur donné. Il est donc possible de décider du niveau de risque pris. De plus, en raison de la distribution normale de la valeur estimée, une valeur estimée fausse (c'est-à-dire dont l'intervalle de confiance ne contient pas la valeur de la grandeur mesurée) sera, avec une forte probabilité, quand même proche de la grandeur mesurée : plus simplement, si le degré de risque porte sur la valeur, le risque que l'ordre de grandeur de la valeur mesurée soit faux est bien plus faible. Dans le cadre d'une étude de sûreté de fonctionnement d'avant-projet, une estimation de l'ordre de grandeur est généralement suffisante pour comparer des solutions techniques. Ce risque n'empêche donc pas l'utilisation de la simulation stochastique pour les études préliminaires de sûreté de fonctionnement.

5.2.b Évènements rares

L'exploration partielle de l'espace d'état rend l'observation d'évènements "rares" difficile. Ainsi, un évènement survenant avec une probabilité de 10^{-6} demandera environ 10^{6} expériences par simulation stochastique pour être observé, et quelques ordres de grandeur supplémentaires pour pouvoir estimer de façon satisfaisante ses conséquences.

Les phénomènes étudiés dans le cadre d'une étude de sûreté de fonctionnement sont souvent provoqués par un dysfonctionnement de plusieurs composants. Or, la défaillance concomitante de plusieurs composants est un évènement rare. Il est donc nécessaire de générer un grand nombre d'histoires pour les observer et mesurer les conséquences des évènements rares.

5.3 Accélération de la simulation stochastique

Pour permettre la prise en compte des évènements rares en un temps raisonnable, la simulation stochastique peut être accélérée. Cette accélération peut être obtenue soit en accélérant la simulation d'une histoire du modèle étudié, soit en améliorant l'algorithme de simulation stochastique pour diminuer le nombre d'histoires complètes à générer pour mesurer l'influence de ces évènements rares.

Un indicateur de la performance d'une simulation stochastique utilisé historiquement est la "figure of merit" (FOM) [66] :

$$\text{FOM} = \frac{1}{R^2 \cdot T}$$

Avec R l'erreur relative du résultat de la simulation, et T le temps utilisé pour obtenir un nombre fixé de mesures. La valeur de la FOM dépend évidemment du modèle simulé et du nombre de mesures retenues pour T. La FOM est donc utile pour comparer la simulation d'un même modèle par des méthodes de Monte-Carlo différentes : plus la FOM est grande, plus la performance est bonne. Plus la performance est bonne, plus un nombre important d'histoires pourront être générées en un temps donné, et donc plus les évènements rares pourront être étudiés.

L'erreur relative peut être estimée par la variance : un ensemble de méthodes d'accélération de la simulation de Monte-Carlo visent à l'amélioration de la FOM par la réduction de la variance. Un autre ensemble de méthodes visent à réduire le temps de calcul nécessaire pour obtenir un grand nombre de mesures.

5.3.a Méthodes d'amélioration de la FOM par réduction de la variance

i Importance Sampling

La méthode nommée "Importance Sampling" (IS) [44] consiste à modifier des distributions de probabilités, associées à des évènements du modèle, afin de privilégier l'occurrence de l'évènement rare étudié.

Les modifications doivent être raisonnées, et les résultats obtenus à l'aide du modèle modifié doivent être corrigés pour estimer ceux du modèle non modifié. Le calcul de cette correction et l'utilisation de cette technique sont plus difficiles et le gain est plus faible lorsque le temps de mission simulé devient grand [47].

Cette technique ne permet d'estimer la probabilité d'occurrence que d'un seul évènement rare à la fois : l'étude de plusieurs évènements rares doit faire l'objet d'études distinctes. De plus, il est nécessaire de connaître les combinaisons de transitions qui mènent à cet évènement : cette information est facile à obtenir dans le cas de la simulation d'une chaîne de Markov (il s'agit d'une recherche de chemin dans le graphe d'état, qui est construit). La construction du graphe d'états d'un modèle AltaRica 3.0 a généralement un coût prohibitif. Il devient donc difficile de trouver ces combinaisons de transitions. Cette méthode n'est pas utilisable pour accélérer la simulation stochastique de modèles AltaRica 3.0 qui ne présentent pas de particularité permettant une construction facile de leur graphe d'état.

ii Splitting et Multilevel Splitting

Un évènement est d'autant plus rare qu'il est le résultat d'une suite de tirs de transitions de probabilités faibles. Pour pouvoir l'observer, il faut donc avoir la "chance" que la simulation stochastique tire la première transition, puis avoir de nouveau la "chance" qu'elle tire dans la même histoire la seconde transition, et ainsi de suite jusqu'à ce que l'évènement rare soit observé. Si dans une histoire générée, toutes les transitions nécessaires sauf la dernière sont tirées, alors l'évènement rare n'est pas observé, et l'algorithme de simulation de Monte-Carlo recommence la simulation de l'histoire suivante à l'état initial.

Pour pouvoir observer l'évènement rare en un nombre limité de simulations, il est tentant de ne pas recommencer la simulation de chaque histoire à l'état initial, et de réutiliser le début

d'une histoire qui s'est rapprochée de l'évènement rare. C'est le principe de la méthode du "Splitting" [39] : lorsque l'histoire générée atteint un état proche de l'évènement rare, elle est divisée en plusieurs sous-histoires qui vont chacune être générées à partir de cet état. Les résultats de ces différentes sous-histoires seront comptabilisés dans le résultat final de façon pondérée (la somme des poids des sous-histoires vaudra le poids d'une seule histoire), afin de ne pas corrompre le résultat de la simulation de Monte-Carlo.

Cette méthode peut être étendue pour être à plusieurs niveaux : le "Multilevel Splitting" [24, 46] est efficace lorsque l'évènement redouté est le résultat d'une longue suite d'évènements, en subdivisant les histoires (et les sous-histoires) en différents états, chacun plus proche de l'évènement redouté.

Le choix des états dans lesquels subdiviser est important : ils doivent être suffisants pour rendre la méthode efficace, mais ils ne doivent pas être trop nombreux pour ne pas rendre les calculs plus coûteux qu'une simulation de Monte-Carlo classique. Ce choix peut être fait prenant en compte la probabilité des évènements survenus (Fixed Effort Method) [28]. Le paramétrage est difficile : la méthode est moins efficace lorsque les états intermédiaires ne sont pas assez rares. Une solution pour déterminer un bon paramétrage est une pré simulation du modèle [2].

De même que pour l'"Importance Sampling", ces méthodes de réduction de la variance nécessitent de déterminer une distance de l'état du modèle à l'évènement rare, afin de déterminer le ou les états justifiant une séparation ("splitting") de l'histoire. De plus, la notion d'état d'un modèle AltaRica 3.0 inclut l'état de l'échéancier, et donc les dates de tir des prochaines transitions : en utilisant cette méthode pour la simulation stochastique de modèles AltaRica 3.0, les sous-histoires d'une même histoire auraient le même début de suite de transitions. Cette méthode est donc adaptée à la simulation stochastique de chaînes de Markov, mais n'est pas facilement adaptable pour accélérer la simulation stochastique de modèles AltaRica 3.0 pour lesquels l'échéancier contient une partie de l'état.

iii Méthode de conditionnement temporel

Le conditionnement temporel [23] part de la même constatation que les méthodes de "Splitting" : lorsque la simulation est dans un état proche de l'évènement rare, il est regrettable de ne pas en profiter pour observer ses conséquences. Les méthodes de "Splitting" divisent l'histoire générée en plusieurs sous-histoires à partir d'un tel état, en espérant qu'une ou plusieurs de ces sous-histoires permettent d'observer l'évènement rare et ses conséquences. Le principe des méthodes de conditionnement temporel est de forcer la ou les transitions qui mènent à cet évènement rare : elle est ainsi parfois appelée "Failure Forcing" [40].
Lorsque le modèle simulé est dans un état proche de l'évènement rare :

- soit la prochaine transition tirée rapproche de l'évènement rare ;
- soit la prochaine transition tirée ne rapproche pas de l'évènement rare ;

La date d'occurrence des transitions est choisie stochastiquement à partir de la distribution de probabilité. En forçant à ce que la date d'occurrence des transitions rapprochant de l'évènement rare soit plus proche qu'un certain délai déterminé, en "tronquant à droite" leurs distributions de probabilité (les délais d'occurrence supérieurs au délai déterminé ne sont alors plus tirables), il devient bien plus probable que la suite de la simulation se rapproche de l'évènement

rare. Les conséquences de cette "troncature" sur la probabilité de l'histoire sont calculables, et permettent de pondérer les résultats obtenus par une histoire ainsi modifiée.

À l'origine limitée aux modèles dans lesquels l'évènement rare est absorbant, une extension de cette méthode (EMCT) aux modèles pour lesquels des cycles d'évènements rares sont possibles est proposée dans [20].

Ces méthodes demandent, comme pour celles de "Splitting", le calcul d'une distance de l'état du modèle à l'évènement rare : pour la même raison (coût prohibitif de construction du graphe d'états), elles ne sont généralement pas adaptées pour accélérer la simulation d'un modèle AltaRica 3.0.

iv Méthodes de quasi-Monte-Carlo randomisé

La méthode de Monte-Carlo a été historiquement utilisée pour estimer des valeurs numériques d'intégrales. Un exemple de cours est l'estimation de l'aire d'un cercle par le tirage aléatoire des coordonnées d'un grand nombre de points dans un plan borné : en mesurant la proportion de points à l'intérieur du cercle, on estime le rapport entre l'aire du cercle et celle du plan borné (et on peut ainsi estimer la valeur de π). Les coordonnées des points sont choisies aléatoirement et de façon uniforme : il est donc possible de choisir plusieurs points proches, tandis qu'une autre région du plan borné ne sera presque pas choisie. Plus le nombre de points sera important, plus l'estimation sera précise (et plus la variance du résultat sera faible).

Les méthodes de quasi Monte-Carlo randomisé (RQMC) proposent de diminuer plus rapidement la variance du résultat en ne choisissant pas aléatoirement mais quasi aléatoirement les coordonnées des points. L'utilisation de séquences de nombres à faible discrépance (c'est-à-dire composée de nombres quasi uniformément répartis) à la place d'un générateur de nombres (pseudo-)aléatoires permet de répartir les valeurs choisies dans l'espace des variables du modèle. Dans l'exemple de l'estimation de l'aire du cercle, cette méthode est très proche d'une discrétisation du plan. Les points étant uniformément répartis, l'estimation va converger plus rapidement, et la variance du résultat sera plus faible. Cette méthode est particulièrement efficace pour améliorer la convergence dans le cas de modèles avec un grand nombre de variables [50, 38].

Cette méthode est toutefois difficilement applicable pour les problèmes de sûreté de fonctionnement. Dans ces modèles, les évènements correspondant à une défaillance ont une probabilité d'occurrence faible. Ils sont donc associés à des distributions de probabilité telles que seuls des nombres aléatoires appartenant à un petit ensemble vont provoquer leur occurrence durant l'histoire. Choisir un ensemble de nombres aléatoires uniformément répartis dans leur ensemble de départ ne présente donc pas d'intérêt : ces évènements ne surviendront pas.

5.3.b Méthodes d'amélioration de la FOM par réduction du temps de calcul par histoire

Le temps de calcul par histoire est l'autre paramètre à réduire pour améliorer la "figure of merit". Deux familles de méthodes permettent une réduction du temps de calcul par histoire : par modifications de l'algorithme de la simulation stochastique (méthodes de dissociation), et méthodes issues du génie informatique. En effet, le temps de calcul par histoire est fortement lié à la performance de la simulation du modèle : son amélioration dépend donc grandement de l'efficacité algorithmique et de l'implémentation logicielle de cette simulation.

i Méthodes de dissociation

Les méthodes de dissociation requièrent un modèle pour lequel des composants ou des sous-systèmes indépendants peuvent être identifiés. Le principe est alors de simuler indépendamment leur comportement, pour obtenir de chacun un ensemble d'histoires. On obtient alors les histoires et les mesures du système complet en faisant le produit des ensembles d'histoires de chaque sous-système indépendant.

Si cette méthode permet d'obtenir un très grand nombre d'histoires en limitant le nombre de simulations, elle ne peut être utilisée qu'à la condition d'identifier les parties indépendantes. Elle n'est efficace que si le calcul du produit des ensembles d'histoires, ainsi que leur mise en mémoire et le calcul de la mesure sur le modèle complet, sont moins coûteux que la simulation stochastique simple du système complet.

Dans le cas d'un modèle AltaRica 3.0 bouclé, la propagation des variables par point fixe (et donc l'absence de Data-Flow par rapport à la version précédente du langage) rend les différentes parties de l'automate à états interdépendantes. Il n'est donc pas possible, dans le cas général, d'isoler des sous-systèmes indépendants. L'utilisation des méthodes de dissociation pour accélérer la simulation stochastique ne concernerait donc qu'un sous-ensemble des modèles AltaRica 3.0, et le gain en performances est incertain.

ii Calcul parallèle

Une démarche classique en génie informatique pour accélérer des calculs est le recours au parallélisme : le temps total de calcul peut être grandement réduit en permettant à des calculs d'être réalisés simultanément sur des unités de calculs plus ou moins indépendants. Les méthodes vont du calcul distribué (sur plusieurs ordinateurs, plusieurs processeurs) à la programmation multithread (sur plusieurs cœurs d'un même processeur).

Ces répartitions des calculs sont faciles sur des simulations qui ne requièrent que peu de synchronisation (éléments finis...) : c'est le cas des simulations de Monte-Carlo. En effet, la génération des histoires, les mesures et une partie du traitement des résultats peuvent être réalisées de façon totalement indépendante (en calculs et en mémoire) sur des ordinateurs différents, voire même à des temps et des lieux géographiques différents.

Cette démarche ne présente aucune difficulté de mise en œuvre pour la simulation de Monte-Carlo, et est une fonctionnalité classique des logiciels de simulation stochastique (par exemple Moca-RP, simulateur de Monte-Carlo de GRIF [25], propose du calcul multi-CPU). Elle peut donc être utilisée pour obtenir plus rapidement des résultats par simulation stochastique de modèles AltaRica 3.0, au prix d'une plus grande puissance de calcul.

iii Interprétation ou compilation

La génération d'un grand nombre d'histoires implique de nombreux calculs similaires pour simuler le modèle. Pour accélérer cette génération, il est utile de réaliser des calculs préliminaires permettant de simplifier la simulation.

Ainsi, il est plus efficace de générer un simulateur stochastique spécifique à un modèle qui sera compilé, que d'utiliser un logiciel qui va interpréter un modèle [34]. Ce simulateur stochastique spécifique permet de réaliser de façon préliminaire des simplifications, ainsi que des vérifications. Le gain en performance peut alors être beaucoup plus important que celui, classique, entre langages interprétés et compilés.

6 Simulation stochastique de modèle AltaRica 3.0

Les modèles de sûreté de fonctionnement par AltaRica 3.0 ne présentent pas, de façon générale, de propriétés mathématiques permettent leur simulation de façon simple :

- les évènements ne sont pas indépendants : la compilation vers des arbres de défaillances est exclue si l'on souhaite tenir compte des spécificités décrites dans le modèle ;
- les distributions de probabilité ne sont pas exponentielles : la compilation vers des chaînes de Markov ne peut pas être systématique.

L'exploitation d'un simulateur stochastique est la seule solution possible pour certains modèles. Une solution simple est alors d'exploiter un simulateur stochastique déjà existant.

6.1 Compilation vers un autre formalisme

Plusieurs outils de simulation stochastique existent, mais aucun ne travaille sur des modèles AltaRica 3.0 ou des GTS. La conversion, ou la compilation, d'un modèle AltaRica 3.0 vers un autre formalisme pour être utilisé par l'un de ces logiciels est un obstacle :

- si le formalisme est moins expressif qu'AltaRica 3.0, des informations du modèle AltaRica 3.0 ne vont pas pouvoir être exprimées dans le formalisme d'arrivée. Les résultats de la simulation stochastique ne seront alors pas représentatifs.
- des formalismes aussi expressifs qu'AltaRica 3.0 existent, par exemple les réseaux de Petri stochastiques et temporisés pour lesquels des simulateurs stochastiques existent. Cependant ils ne permettent pas la propagation des assertions par un mécanisme du point fixe. La conversion d'un modèle AltaRica 3.0 vers ces formalismes demande alors une mise à plat du point fixe, ce qui provoque, dans le cas général, une explosion combinatoire du modèle. Le modèle obtenu ne serait alors pas simulable de façon efficace.

Pour pouvoir simuler stochastiquement un modèle AltaRica 3.0, un outil dédié à ce langage doit donc être utilisé.

6.2 Simulateur stochastique AltaRica 3.0

Une première version d'un outil de simulation stochastique de modèle AltaRica 3.0 a été développée [11].

La simulation stochastique, pour être utilisable, doit être efficace en ressources de calculs. Des travaux ont été menés sur l'accélération de la simulation stochastique de modèles AltaRica Dataflow [35], et les résultats ont été utilisés pour concevoir le simulateur stochastique AltaRica 3.0. Toutefois, ses performances ne sont pas suffisantes : leur amélioration est l'un des sujets des travaux rapportés ici (chapitre III).

7 Apports et organisation de ce document

Afin de pouvoir utiliser la simulation stochastique de modèles AltaRica 3.0 pour calculer des indicateurs de sûreté de fonctionnement, plusieurs problèmes doivent être résolus.

Une fois les systèmes à étudier modélisés en AltaRica 3.0, l'analyste de sûreté de fonctionnement doit pouvoir exprimer ce qu'il souhaite voir calculé par simulation stochastique. Le chapitre II propose des indicateurs, permettant de relier les grandeurs probabilistes des exigences de sûreté de fonctionnement aux éléments du modèle AltaRica 3.0.

La mise en oeuvre d'une étude par simulation stochastique demande des moyens de calcul important. Le chapitre III présente la démarche et les solutions apportées au simulateur stochastique de modèles AltaRica 3.0 pour améliorer ses performances, et ainsi diminuer ce coût.

Les systèmes complexes et critiques sont une combinaison de parties mécaniques et de logiciels de commande et de contrôle. Le chapitre IV est un cas d'étude qui montre que les modèles AltaRica 3.0 et la simulation stochastique permettent l'étude de sûreté de fonctionnement de ces systèmes mécatroniques.

Les outils utilisés lors d'étude de sûreté de fonctionnement doivent être testés et évalués, afin d'avoir confiance dans les résultats fournis. Les outils de simulation stochastique présentent des caractéristiques qui rendent ces évaluations complexes. Le chapitre V propose une méthodologie pour s'assurer du bon fonctionnement de ces outils.

Chapitre II

Besoins de la simulation stochastique

1 Introduction

L'objectif d'une analyse de sûreté de fonctionnement en phase d'avant-projet est de comparer différentes solutions techniques par rapport à des exigences de sûreté, afin de ne retenir que celles qui sont les plus intéressantes pour la suite du projet. La réalisation d'expériences numériques sur des systèmes virtuels permet d'obtenir d'obtenir des grandeurs numériques comparables aux valeurs de ces exigences.

Le principe des expériences numériques (figure II.1) est l'utilisation d'un modèle, c'est-à-dire une représentation mathématique du système sous forme d'équations, de machines à états... Des indicateurs numériques sont obtenus par la réalisation d'une expérience numérique : il peut s'agir de calculs réalisés sur le modèle (résolution des équations différentielles, construction de l'espace d'état...) ou de mesures réalisées sur les résultats d'une simulation (pas-à-pas, par éléments finis, ou encore stochastique). Les valeurs des indicateurs sont ensuite comparées aux niveaux des exigences, afin de conclure sur leur satisfaction par le système étudié.

Une expérience numérique est réalisée à l'aide d'un outil de calcul (un logiciel). Pour qu'une étude de sûreté de fonctionnement par une expérience numérique soit réalisable, il faut que cet outil :

- soit capable de comprendre le modèle, donc son langage d'expression ;
- soit capable de réaliser des calculs sur ce modèle, d'après la technique de simulation ou de calcul choisie ;
- fournisse des résultats utilisables, c'est-à-dire que l'on peut comparer aux exigences.

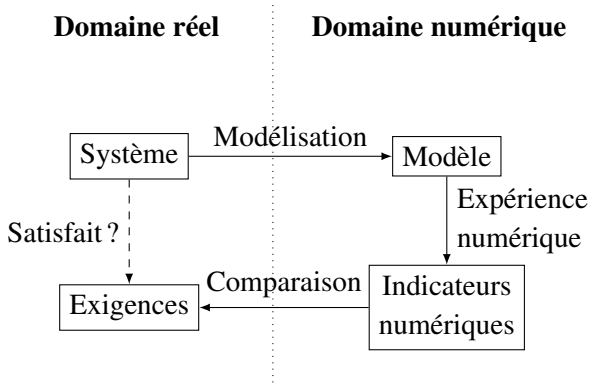

FIGURE II.1 – Principe de l'expérience numérique

L'utilisation d'expériences numériques pour établir des indicateurs est un problème complexe dont la mise en œuvre est fortement dépendante du contexte d'utilisation.

On s'intéresse dans ce chapitre au calcul d'indicateurs :

- de sûreté de fonctionnement : les exigences étudiées sont donc des exigences de sûreté de fonctionnement ;
- pour des modèles AltaRica 3.0 : le langage d'expression des modèles ;
- par simulation stochastique, qui est la technique de simulation retenue.

L'objectif des travaux de ce chapitre est la spécification des indicateurs qu'un outil de simulation stochastique de modèles AltaRica 3.0 doit fournir lors d'une étude de sûreté de fonctionnement.

1.1 Illustration des problématiques

Cette partie a pour objectif de faciliter la compréhension des problématiques soulevées par l'utilisation de la simulation stochastique, en l'illustrant de façon sommaire grâce à un cas d'étude fictif. Nous allons considérer qu'un analyste de sûreté de fonctionnement souhaite obtenir la valeur du temps moyen entre pannes (MTBF) d'une solution technique, par simulation stochastique.

1.1.a Désignation des états d'intérêt

La MTBF est une grandeur probabiliste qui quantifie la durée moyenne entre deux défaillances d'un système. Afin de calculer la MTBF de la solution technique, il est donc d'abord nécessaire d'indiquer quels états de l'automate représentent le système étudié comme étant en panne.

La solution technique à étudier est représentée par un modèle AltaRica 3.0, qui est un automate à états temporisé stochastique. À partir de ce modèle, la simulation stochastique va générer un ensemble d'histoires différentes. Une histoire est une évolution possible de l'automate, et représente ainsi une évolution possible de la solution technique étudiée. Cette évolution correspond à une succession d'états de l'automate, séparés par des transitions datées. Un état contient l'ensemble des informations identifiant de manière unique la configuration de l'automate. Un état contient donc notamment les valeurs de chacune des variables de l'automate.

Les seules informations utiles au calcul d'une MTBF contenues dans une histoire sont alors l'état de panne, ou non, de l'automate, et les dates de passage d'un état à l'autre. On peut ainsi ne retenir de chaque histoire que la seule information, synthétique, de l'état de panne en fonction de la date simulée (représentable par un chronogramme, figure II.2).

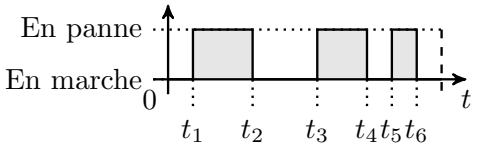

FIGURE II.2 – Histoire résumée à l'état de défaillance

Dans un modèle AltaRica 3.0, un observateur permet au modélisateur de définir une expression sur les variables de l'automate. Un observateur permet donc de synthétiser les informations contenues dans un état pour ne retenir que l'information utile à une étude particulière.

1.1.b Mesure du temps entre pannes

Pour obtenir la MTBF, l'analyste de sûreté de fonctionnement doit décrire à son outil de simulation stochastique comment mesurer le temps moyen entre pannes sur chacune de ces histoires :

- Relever les dates de pannes :
  - C'est-à-dire les fronts montants de la fonction temporelle indiquant si le système est en panne ;
  - Sur l'exemple (figure II.2) : t_1, t_3 et t_5 ;
- Calculer la moyenne des écarts entre panne :
  - Calculer $t_5 - t1$;
  - Diviser cette valeur par le nombre de pannes moins un.

Cet algorithme permet de calculer le temps moyen entre pannes pour une histoire.

De façon plus générale, les mesures à retenir de chaque histoire doivent pouvoir être paramétrées dans l'outil de simulation stochastique. Dans le cadre de ces travaux, le moyen technique retenu pour décrire les mesures à réaliser sur chaque histoire est l'utilisation d'indicateurs.

1.1.c Estimation du temps moyen entre pannes

L'application de la mesure décrite précédemment sur chacune des histoires générées par simulation stochastique permet d'obtenir une liste de temps moyen entre pannes.

Seulement, cet ensemble d'histoires générées n'est qu'une sous-partie des histoires possibles de la solution technique étudiée. Le temps moyen entre pannes recherché est une grandeur probabiliste correspondant à l'ensemble des histoires possibles, pas seulement de celles générées.

Il n'est pas possible de calculer de façon exacte la grandeur probabiliste à partir d'une sous-partie des histoires. Il est en revanche possible de l'estimer, c'est-à-dire d'obtenir une valeur associée à un intervalle de confiance, à la manière d'un "sondage d'opinion" (il n'est

pas possible de connaître les résultats d'un vote avant sa clôture, mais à partir d'un sondage, portant sur une partie restreinte des votants, il est possible d'indiquer avec un certain degré d'incertitude une estimation du résultat, à un intervalle donné près. Plus la part de la population interrogée sera grande, plus l'intervalle sera restreint pour le même degré d'incertitude).

La simulation stochastique va ainsi permettre d'obtenir une estimation de la grandeur probabiliste, associée à un intervalle de confiance. C'est cette estimation qui pourra être comparée aux niveaux des exigences, ou utilisée pour comparer des solutions techniques entre elles.

1.2 Organisation du chapitre

Dans un premier temps, le contexte d'une telle étude et les besoins qu'elle présente sont détaillés (II.2). Les choix techniques retenus pour permettre de telles analyses sont présentés (II.3). Le déroulement d'une étude de sûreté de fonctionnement par simulation stochastique, utilisant les choix techniques proposés, est décrite (II.4). Enfin, les possibilités de calcul sont illustrées (II.5).

2 Contexte d'une analyse de sûreté de fonctionnement par simulation stochastique

La conception d'un système se déroule classiquement par raffinements successifs, en partant de l'analyse des besoins qui permettent de définir un ensemble de systèmes à assembler, qui sont ensuite détaillés jusqu'à la conception ou le choix de chaque composant. Les solutions techniques qui seront utilisées lors des étapes suivantes sont choisies parmi elles, ou à partir d'elles, en prenant en compte notamment des critères de sûreté de fonctionnement.

C'est par exemple le cas lors de la conception d'un avion suivant la norme ARP 4754 [3] : la branche descendante du cycle en V (figure II.3) concerne la conception par raffinements successifs du système complexe et critique qu'est un avion. Chaque étape de conception est validée par une étude de sûreté de fonctionnement (Aircraft FHA, PSSA, System FHA, PASA...).

Le rôle de l'analyste de sûreté de fonctionnement est d'indiquer quelles solutions sont à retenir : une des difficultés est que ces solutions ne sont pas complètement spécifiées, car nous sommes seulement dans les étapes intermédiaires de la conception. Par son expertise, l'analyste peut rapidement écarter des solutions qui ne sont pas viables, par exemple parce qu'elles présentent un point unique de défaillance ("single point of failure") évident. Pour le classement des solutions techniques restantes, il est nécessaire de disposer d'indicateurs numériques, représentant le niveau de respect des exigences de sûreté de fonctionnement de chaque solution (ou variante de solution) étudiée. Pour permettre l'estimation des valeurs numériques de ces indicateurs, les solutions techniques à étudier doivent être modélisées puis étudiées grâce à des expériences numériques.

2.1 Caractéristiques des exigences à traiter

Les exigences de sûreté de fonctionnement (RAMS) font partie des exigences non fonctionnelles : elles décrivent des propriétés que le système doit présenter. Elles sont soit liées à ses fonctions (le système doit assurer ses fonctions principales correctement au moins 99% du

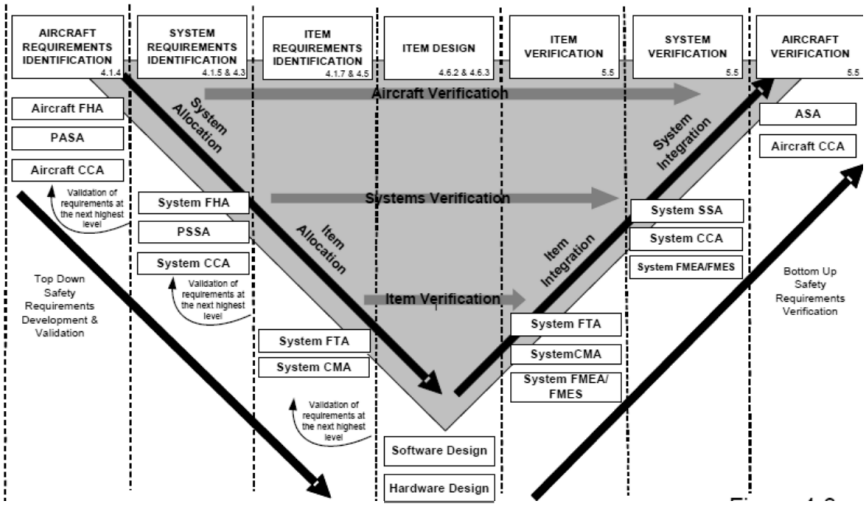

FIGURE II.3 – Extrait de l'ARP 4754 : Cycle en V

temps...), soit liées à son environnement (le système ne doit pas polluer, il ne doit pas blesser un opérateur).

Alors que les exigences fonctionnelles portent sur les propriétés du système dans les phases de bon fonctionnement, les exigences de sûreté de fonctionnement portent sur ses propriétés en dehors de ces phases : quand un composant est défaillant, ou quand un élément extérieur au système le perturbe (incendie, perte d'alimentation électrique...). Comme de tels évènements ne peuvent être évités, un système physique totalement sûr ne peut pas être construit : il est toujours possible que toutes les mesures mises en place soient défaillantes. On ne peut donc construire que des systèmes avec une probabilité de défaillance la plus faible possible mais non nulle : les exigences correspondantes sont donc probabilistes, c'est-à-dire que leurs niveaux sont exprimés en fréquences moyennes (par exemple en nombre de défaillances par heure de fonctionnement) ou en temps moyens (MTTF, MTBO...), mais pas de façon absolue ("aucune défaillance").

C'est une différence importante entre la vérification de programmes informatiques (*model-checking*) et le calcul d'indicateurs de sûreté de fonctionnement de systèmes : on peut vérifier qu'un programme ne sera jamais dans un état non désiré, tandis qu'un évènement catastrophique peut toujours survenir sur un système physique.

Contexte 1 : Exigences

Les exigences à évaluer sont probabilistes.

L'étude de sûreté de fonctionnement vise à évaluer ces fréquences et temps moyens, afin de le comparer aux niveaux des exigences de sûreté de fonctionnement ou de classer différentes solutions techniques.

2.2 Modèles

Les exigences de sûreté de fonctionnement sont liées au comportement dysfonctionnel du système. Les modèles classiques de systèmes sont fonctionnels : ils décrivent comment le système se comporte dans des circonstances normales.

Par exemple, un modèle simple d'une diode semi-électrique décrit son comportement suivant le signe du potentiel électrique : le courant la parcourant est soit nul, soit positif (la diode est bloquante ou passante). Les conditions de bon fonctionnement sont des limites explicites de ce modèle : le courant ne doit pas dépasser un certain maximum, de même que le potentiel électrique, ou encore la température de l'environnement doit être comprise dans une plage de fonctionnement. Si les limites de fonctionnement de la diode sont dépassées, le composant peut adopter un comportement éloigné de son modèle fonctionnel : la diode peut devenir passante ou bloquante en toutes circonstances, ou encore agir comme une résistance, de façon temporaire ou définitive.

Ces comportements dysfonctionnels ne sont pas décrits par le modèle fonctionnel, et ont des conséquences sur le bon fonctionnement du système. Le modèle utilisé pour étudier les performances de sûreté de fonctionnement du système doit donc contenir des informations sur le comportement dysfonctionnel des composants.

D'autres limites au modèle fonctionnel existent : le processus de modélisation oblige à négliger certains aspects, comme le vieillissement, l'usure ou les défauts de fabrication. Ces limites ne sont pas forcément explicites ni maîtrisées, mais provoquent aussi des dysfonctionnements des systèmes.

Si l'occurrence et les conséquences d'une surtension sont modélisables par une équation systématique (si la tension dépasse le maximum admissible, alors le composant défaille), ce n'est pas possible pour les causes non maitrisées. Elles sont alors modélisées de façon probabiliste : il existe une probabilité que le composant défaille. Le chiffrage précis de cette probabilité demande une connaissance du composant, voire une étude statistique sur son fonctionnement au cours du temps, ce qui ne peut être réalisé qu'a posteriori.

Dans le cas d'une étude de sûreté de fonctionnement en phase de conception, ce chiffrage précis n'est pas disponible. Dans les premières phases de la conception du système, les composants ne sont pas encore choisis, seules des descriptions des sous-systèmes peuvent être disponibles. Pour avoir un modèle le plus représentatif possible, l'analyste de sûreté de fonctionnement peut proposer des probabilités (taux de défaillance...) pour modéliser ces causes, à partir de ses connaissances, de bases de données de composants ou sous-systèmes similaires, et des spécifications disponibles.

Contexte 2 : Modèles

Les modèles sont :

- stochastiques, c'est-à-dire non déterministes : il existe plusieurs évolutions au cours du temps possibles ;
- faiblement représentatifs quantitativement : toutes les grandeurs ne sont pas forcément connues précisément.

En plus du comportement dysfonctionnel des sous-systèmes et composants, le modèle peut comporter des informations représentant des stratégies de maintenance. Celles-ci ont des conséquences sur le respect des exigences de sûreté de fonctionnement, l'étude réalisée doit donc en tenir compte.

2.3 Besoins fonctionnels d'une étude de sûreté de fonctionnement

Une étude de sûreté de fonctionnement en phase de conception répond à deux besoins fonctionnels : évaluer des grandeurs probabilistes, correspondant aux performances des solutions techniques aux exigences de sûreté de fonctionnement, et comparer des solutions techniques suivant ces mêmes exigences.

2.3.a Évaluer des grandeurs probabilistes de façon fiable

Les résultats d'une étude de sûreté de fonctionnement sont utilisés par l'analyste de sûreté de fonctionnement pour décider des solutions techniques à retenir pour la conception d'un système critique. L'analyste doit donc avoir confiance dans les résultats des expériences numériques réalisées.

Ce point est d'autant plus important que la méthode de calcul utilisée, la simulation stochastique, est basée sur une exploration partielle et aléatoire de l'espace d'histoires du modèle : le résultat ne peut donc pas être une valeur calculée de façon exacte, mais seulement une estimation. De plus, une partie des comportements intéressants pour une étude de sûreté de fonctionnement peuvent être qualifiés de "rares", c'est-à-dire avec une très faible fréquence d'occurrence. Ils peuvent ne pas être visibles par une exploration trop partielle.

L'outil de calcul doit donc fournir à l'analyste un moyen de quantifier la confiance qu'il peut accorder au résultat.

Besoin fonctionnel 1 : Quantifier la fiabilité des résultats

Les résultats obtenus doivent être associés à une mesure de leur représentativité.

2.3.b Comparer des solutions techniques

Pour classer les différentes solutions techniques à étudier, il est nécessaire d'obtenir des valeurs comparables. Les résultats obtenus par l'étude de chaque solution technique doivent correspondre aux mêmes calculs, permettant l'évaluation des mêmes critères de sûreté de fonctionnement. Le paramétrage des résultats attendus d'une étude par simulation stochastique doit donc être indépendant du modèle, pour être applicable à la simulation des modèles des différentes solutions techniques.

Besoin fonctionnel 2 : Comparaison des résultats

Les résultats obtenus doivent permettre une comparaison des modèles étudiés.

2.4 Contexte opérationnel d'une étude de sûreté de fonctionnement

Dans le cadre des travaux rapportés dans ce document, les études de sûreté de fonctionnement sont réalisées dans un contexte opérationnel spécifique : les modèles utilisés sont exprimés à l'aide d'AltaRica 3.0, et la méthode de calcul retenue est la simulation stochastique.

2.4.a Caractéristiques des modèles AltaRica 3.0 à prendre en compte

AltaRica 3.0 est un formalisme de modélisation haut-niveau, dédié aux études de sûreté de fonctionnement. Une spécification complète d'AltaRica 3.0 est disponible [8]. Il s'agit d'un langage orienté prototype, permettant la réalisation de bibliothèque de composants réutilisables dans plusieurs modèles. La modélisation de plusieurs variantes de solutions techniques proches est ainsi facilitée.

Le formalisme mathématique utilisé est celui des systèmes de transitions gardées (GTS) (annexe A) : le comportement dynamique est ainsi décrit à l'aide d'une machine à états temporisée et stochastique.

i Transitions et évènements AltaRica 3.0

Comme dans les formalismes de machines à états classiques, les transitions permettent de passer d'un état à un autre. Dans un modèle AltaRica 3.0, ces transitions sont associées à un évènement qui est nommé. Cet évènement peut être stochastique et temporisé. Pour le système modélisé, il représente l'occurrence d'une défaillance, d'une réparation, ou d'une synchronisation, c'est-à-dire des évènements utiles à une étude de sûreté de fonctionnement.

Contexte opérationnel 1 : Evènements AltaRica 3.0

Les évènements AltaRica 3.0 portent des informations utiles à une étude de sûreté de fonctionnement.

ii Variables AltaRica 3.0

Comme dans les formalismes de machines à états classiques, une partie de l'information sur l'état courant du modèle est comprise dans les valeurs des variables. Les grandeurs probabilistes à estimer lors d'une étude de sûreté de fonctionnement pouvant porter sur l'état courant du modèle, il faut pouvoir accéder à ces valeurs.

Contexte opérationnel 2 : Variables AltaRica 3.0

Les variables d'état et de flux AltaRica 3.0 portent des informations utiles à une étude de sûreté de fonctionnement.

iii Observateurs AltaRica 3.0

Pour faciliter les expériences numériques sur les modèles AltaRica 3.0, la notion d'observateur a été introduite dans le langage. Sur le plan mathématique, il s'agit d'expressions nommées, portant sur les variables du modèle. La valeur de chaque expression est calculée après chaque tir de transition, en utilisant les valeurs des variables concernées. En utilisant uniquement les valeurs courantes des variables, les observateurs ne prennent pas en compte leurs valeurs précédentes, et ne peuvent donc pas être utilisés comme des mémoires des états passés.

Contexte opérationnel 3 : Observateurs AltaRica 3.0

Les observateurs AltaRica 3.0 portent sur l'état courant du modèle, et ne prennent pas en compte leurs valeurs précédentes.

Un observateur permet au concepteur du modèle d'indiquer un "point de mesure" potentiellement intéressant pour une analyse. Le nom qui est donné à un observateur permet à l'analyste le moyen d'y faire référence.

Par exemple, un observateur peut être défini dans le modèle pour mesurer le nombre de composants en panne dans l'état courant : lors d'une analyse par simulation, il ne sera alors pas nécessaire de lister les composants et de compter ceux en panne, puisque cette quantité est déjà calculée.

```
1  class Pipe
2    Boolean working (init = true); // variable d'etat : bon fonctionnement du
       composant
3    parameter Real lambda = 0.0001; // taux de defaillance
4    Boolean in (reset = false); // flux d'entree
5    Boolean out (reset = false); // flux de sortie
6    // évènement de défaillance
7    event failure (delay = exponential(lambda));
8    transition
9      // transition de défaillance
10     failure: working -> working := false;
11   assertion
12     if working then out := in;
13 end
14
15 block PipeNetwork
16   // 3 tuyaux
17   Pipe P1, P2, P3;
18   observer Boolean oOut = P3.out;
19   assertion
20     // entree --> P1 --> P2 --> P3 --> sortie (observateur)
21     P1.in := true;
22     P2.in := P1.out;
23     P3.in := P2.out;
24 end
```

FIGURE II.4 – Modèle AltaRica 3.0 d'un réseau de trois tuyaux

iv Gestion des transitions, variables et observateurs d'un modèle AltaRica 3.0

Une analyse d'un modèle AltaRica 3.0 commence par la mise à plat de son architecture, c'est-à-dire la compilation de l'arborescence de GTS en un unique GTS. Ensuite, des simplifications peuvent être apportées au modèle par l'outil, par exemple une simplification des assertions (voir III.3), ce qui a pour effet de supprimer des variables de flux du modèle.

Par exemple, le modèle AltaRica 3.0 présenté figure II.4 représente trois tuyaux connectés en série. Chaque modèle de tuyau a deux variables de flux, une pour son entrée et une pour sa

sortie. Si le tuyau est en bon état, le flux en entrée est reporté à la variable de flux de sortie. Le flux en sortie d'un tuyau est directement relié au flux en entrée du tuyau suivant. Pour des raisons de performances, le GTS obtenu lors de la mise à plat du modèle AltaRica 3.0 peut supprimer des variables. Dans ce cas, la variable de flux représentant l'entrée d'un tuyau sera toujours de valeur égale à la variable de flux représentant la sortie du tuyau précédent. Il n'est donc pas utile de conserver deux variables de flux : une seule peut être conservée. Le GTS obtenu (figure II.5) ne possède donc plus qu'une seule de ces deux variables : l'autre a été substituée dans toutes les expressions par la première.

Il n'est alors plus possible d'utiliser ces variables lors de l'expérience : les comportements des deux modèles (originel et simplifié) sont identiques, mais les informations ne sont plus portées par les mêmes variables.

D'autres simplifications pourraient être apportées, au choix du compilateur AltaRica 3.0 : par exemple, un groupe de variables booléennes pourraient être remplacées par une seule variable entière, ou deux transitions immédiates remplacées par une seule, si cela peut entrainer une simplification du modèle sans en modifier le comportement. De telles modifications ne sont pas encore implémentées dans le compilateur AltaRica 3.0, mais ne sont pas interdites : seuls les observateurs sont garantis sans modification par l'outil d'analyse.

Contexte opérationnel 4 : Modèles AltaRica 3.0

Un modèle AltaRica 3.0 peut subir des transformations et des simplifications avant sa simulation.

2.4.b Simulation stochastique de modèle AltaRica 3.0

Un modèle AltaRica 3.0 est un GTS temporisé stochastique (voir A.5.2). Un tel automate est déterministe : la partie aléatoire, l'oracle, est extérieure au GTS. Durant une simulation, cet oracle est utilisé, en consommant son premier élément, pour deux occasions : déterminer le délai stochastique des évènements, et choisir lequel des évènements prévus à la même date doit survenir en premier. Pour simuler un modèle AltaRica 3.0, il faut donc lui fournir un accès à un oracle. C'est la lecture de cet oracle (la consommation de son premier élément) qui va faire varier l'évolution de l'automate.

Ces différentes évolutions de l'automate, générées par simulation stochastique, seront appelées des **histoires**.

Contexte opérationnel 5 : Histoires

La simulation stochastique d'un modèle AltaRica 3.0 génère des histoires, c'est-à-dire des suites d'états du modèle, séparés par des tirs de transitions.

2.5 Besoins opérationnels

Une simulation stochastique d'un modèle AltaRica 3.0 génère un ensemble d'histoires, c'est-à-dire un ensemble d'évolutions de l'automate. Cet ensemble n'est pas directement exploitable par un analyste, dont le rôle est d'évaluer des grandeurs probabilistes correspondant à des critères d'exigences de sûreté de fonctionnement pour comparer des solutions techniques.

```
block PipeNetwork
  Boolean P1.working (init = true);
  Boolean P2.working (init = true);
  Boolean P3.working (init = true);

  Boolean P1.in (reset = false);
  Boolean P1.out (reset = false);
  // P2.in substituée par P1.out, donc P2.in supprimée
  // Boolean P2.in (reset = false);
  Boolean P2.out (reset = false);
  // P3.in substituée par P2.out, donc P3.in supprimée
  // Boolean P3.in (reset = false);
  Boolean P3.out (reset = false);

  event P1.failure (delay = exponential(0.0001));
  event P2.failure (delay = exponential(0.0001));
  event P3.failure (delay = exponential(0.0001));

  observer Boolean oOut = P3.out;
  transition
    P1.failure: P1.working -> P1.working := false;
    P2.failure: P2.working -> P2.working := false;
    P3.failure: P3.working -> P3.working := false;
  assertion
    P1.in := true;
    if P1.working then P1.out := P1.in;
    // P2.in substituée par P1.out
    // if P2.working then P2.out := P2.in; devient :
    if P2.working then P2.out := P1.out;
    // P3.in substituée par P2.out
    // if P3.working then P3.out := P3.in; devient :
    if P3.working then P3.out := P2.out;

    // P2.in substituée par P1.out
    // P2.in := P1.out;
    // est devenue P1.out := P1.out;
    // donc est supprimée.
    // P3.in substituée par P2.out
    // P3.in := P2.out;
    // est devenue P2.out := P2.out;
    // donc est supprimée.
end
```

FIGURE II.5 – Exemple de simplification du GTS obtenu lors de la mise à plat du modèle AltaRica 3.0 figure II.4

Pour relier ces grandeurs probabilistes aux ensembles d'histoires qui ont été générés, il est nécessaire de lui fournir un moyen d'exprimer les calculs à réaliser sur les ensembles d'histoires pour lui permettre d'obtenir des estimations des grandeurs probabilistes voulues.

2.5.a Correspondance avec le modèle

Les grandeurs probabilistes à estimer portent sur le comportement du modèle, donc sur les états atteints (et les dates correspondantes) lors de l'évolution du modèle.

L'analyste doit pouvoir indiquer quels sont les états utiles pour l'étude qui l'intéresse. Cette personnalisation est nécessaire car un même modèle est classiquement l'objet de plusieurs études de sûreté de fonctionnement. Les états utiles pour l'estimation des grandeurs probabilistes peuvent différer d'une étude à l'autre. L'analyste de sûreté de fonctionnement doit donc pouvoir indiquer quels sont ces états utiles.

Besoin opérationnel 1 : Correspondance avec le modèle

Les états utiles du modèle à évaluer doivent être décrits à l'outil de simulation stochastique.

2.5.b Comparabilité

Chaque solution technique à évaluer est modélisée par un modèle AltaRica 3.0 différent. Les descriptions des états utiles vont donc être différentes. Des parties importantes des modèles peuvent être communes, surtout dans le cas de l'étude de variantes d'une solution technique, mais ce sont sur ces différences que vont porter les grandeurs probabilistes à évaluer.

Pour pouvoir comparer différentes solutions techniques (besoin fonctionnel 2), les résultats doivent être des mesures comparables portant sur des modèles différents. La description des mesures à réaliser doit donc être indépendante de la description des états utiles, afin de pouvoir être identique sur les différents modèles étudiés.

Besoin opérationnel 2 : Indépendance de la description des états et de l'estimation

Les descriptions des états utiles doivent être indépendantes de la description des mesures à réaliser.

2.5.c Expressivité

Les grandeurs probabilistes à étudier sont variées : elles peuvent porter sur des quantités (calculs de productivité), des durées (MTTF, MTBF...), ou encore sur l'atteinte ou non d'états particuliers (disponibilité...). Elles peuvent de plus être le résultat d'un calcul portant sur l'histoire complète du modèle (temps entre deux pannes, nombre moyen de pannes...).

Pour pouvoir estimer une grandeur probabiliste, il faut pouvoir exprimer le calcul à réaliser.

Besoin opérationnel 3 : Description des calculs

Il doit être possible de réaliser les calculs des indicateurs classiques de sûreté de fonctionnement.

2.5.d Non ambiguïté

Les résultats, c'est-à-dire les estimations des grandeurs probabilistes, permettent de prendre des décisions quant à des questions de sûreté de fonctionnement. Pour qu'ils soient exploitables, il ne doit pas y avoir de différence entre ce que l'analyste de sûreté de fonctionnement exprime et les calculs qui sont effectivement réalisés. L'absence d'ambiguïté dans l'expression des estimations est donc un besoin opérationnel.

Besoin opérationnel 4 : Formalisme des descriptions

Les calculs à réaliser doivent pouvoir être décrits sans ambiguïté.

3 Choix techniques

Pour estimer des grandeurs probabilistes par simulation stochastique de modèles AltaRica 3.0, dans le cadre d'une étude de sûreté de fonctionnement, il est nécessaire d'avoir un moyen d'exprimer ce que l'on souhaite estimer : les états utiles, et les calculs à réaliser. Dans cette partie, les choix techniques retenus pour le simulateur stochastique de modèles AltaRica 3.0 du projet OpenAltaRica, répondant aux besoins fonctionnels et opérationnels détaillés précédemment, sont présentés.

La solution technique retenue repose sur les observateurs AltaRica 3.0 étendus aux évènements pour décrire les états utiles, et un ensemble d'indicateurs décrivant les calculs à réaliser sur les histoires (résultats des simulations stochastiques), et qui couvrent les besoins classiques de sûreté de fonctionnement.

3.1 Description des états utiles

Dans un modèle AltaRica 3.0, ou plus précisément dans un GTS, l'information sur l'état courant est portée par les valeurs des variables. Il faut donc pouvoir accéder à ces valeurs pour réaliser des mesures portant sur l'état du modèle (besoin opérationnel 1). Toutefois, les variables sont susceptibles de modifications (simplifications, substitutions…, voir II.2.4.a.iv) lors de la phase de compilation du modèle AltaRica 3.0 en un modèle GTS, c'est-à-dire entre la modélisation et la simulation d'un modèle (contexte opérationnel 4). Le langage AltaRica 3.0 propose comme "points de vue" sur l'état du modèle des observateurs (voir A.3), qui permettent d'éviter ce problème : un indicateur est donc configuré avec un observateur du modèle AltaRica 3.0, et va réaliser ses mesures de comportement par rapport à l'évolution de la valeur de cet observateur au cours d'une histoire.

Choix technique 1 : Description des états utiles

Les états utiles au calcul sont décrits à l'aide d'un observateur AltaRica 3.0.

Les observateurs AltaRica 3.0 sont des expressions nommées qui sont évaluées à chaque nouvel état du modèle, et qui ne dépendent que de ses variables. Si les valeurs des variables peuvent suffire pour connaître l'état courant du modèle, des informations pertinentes sur son comportement peuvent être données par les évènements survenant lors des simulations (contexte

opérationnel 1). Les observateurs AltaRica 3.0 ne dépendant que des variables, il n'y a pas d'entité AltaRica 3.0 permettant de définir les évènements qui peuvent être utiles de connaître du modèle.

Pour y remédier, les observateurs sont étendus aux évènements.

Choix technique 2 : Description des états utiles : évènements

Pour pouvoir utiliser les informations portées par les occurrences d'évènements, les observateurs AltaRica 3.0 sont étendus aux évènements.

3.1.a Observateurs AltaRica 3.0 d'évènement

Un observateur d'évènement est une valeur booléenne dépendant de la date, et correspondant à un ou plusieurs évènements du modèle AltaRica 3.0 (ou, de façon équivalente, GTS). Sa valeur à la date t est :

- *true* si l'évènement ou l'un des évènements correspondants est survenu à la date t ;
- *false* sinon.

Une représentation équivalente à cette fonction booléenne est une liste de dates pour laquelle elle vaut *true* : cette liste correspond à la liste des dates auxquelles l'évènement est survenu.

La définition, dans le modèle AltaRica 3.0, d'observateurs d'évènement permet de réaliser des mesures sur les évènements qui surviennent : leurs fréquences, leurs dates de première occurrence... (voir B.3)

3.1.b Comportement temporel

La simulation de modèles AltaRica 3.0 étant évènementielle, elle est réalisée pas de simulation par pas de simulation : chaque pas correspond au tir d'une transition suite à l'occurrence de l'évènement qui l'étiquette (contexte opérationnel 5).

Plusieurs évènements peuvent survenir à la même date, menant au tir de plusieurs transitions, et donc à plusieurs pas de simulation survenant à la même date simulée. Un observateur peut donc alterner entre plusieurs valeurs lors de pas successif survenant à la même date. Il prend alors une valeur lors d'un pas et la quitte lors d'un autre pas, les deux étant à la même date simulée. Une valeur est prise de façon furtive, sur une durée simulée nulle. Elle est donc mesurable (il existe un état de l'histoire dans lequel l'observateur prend cette valeur) en observant les pas de simulation, mais il n'existe pas de date à laquelle le système est dans cet état.

Le tir de plusieurs transitions à la même date survient par exemple avec une modélisation de "redondance froide" : un composant défaille, un composant de secours démarre immédiatement pour le remplacer. Le modèle est simulé tir de transition par tir de transition, mais les exigences portent sur le comportement temporel du modèle : une interruption de fonctionnement d'une durée nulle n'est pas à prendre en compte, le système modélisé rempli sa fonction de façon continue.

Les comportements que les indicateurs mesurent doivent donc porter sur le comportement simulé temporel, et ignorer les états furtifs. Ces états furtifs doivent donc être filtrés entre les histoires générées par simulation stochastique et les histoires utilisées pour les mesures des indicateurs.

Choix technique 3 : Comportement temporel

Pour réaliser des calculs sur le comportement temporel et non sur le comportement pas-à-pas, les histoires sont filtrées pour supprimer les états furtifs.

Filtrage des histoires

L'exécution d'un GTS temporisé peut être représentée par une séquence (voir A.5.1). Une histoire générée par simulation stochastique étant une exécution d'un GTS temporisé, elle peut donc être représentée par une telle séquence :

$$(\sigma_0, d_0, t_0) \overset{e_0}{\rightarrow} (\sigma_1, d_1, t_1) \dots (\sigma_{n-1}, d_{n-1}, t_{n-1}) \overset{e_{n-1}}{\rightarrow} (\sigma_n, d_n, t_n)$$

Avec :

- σ_i l'ensemble des valeurs des variables à l'état i ;
- d_i l'échéancier c'est-à-dire pour chaque évènement le délai avant qu'il ne survienne ;
- t_i la date à laquelle l'évènement e_i survient.

Un pas correspondant au tir d'une transition suite à l'occurrence de l'évènement qui l'étiquette, il est noté :

$$(\sigma_i, d_i, t_i) \overset{e_i}{\rightarrow} (\sigma_{i+1}, d_{i+1}, t_{i+1})$$

Pour réaliser des mesures sur le comportement simulé temporel des histoires générées, un filtrage de la séquence doit être effectué.

$$\forall i \in [\![0, n[\![, \text{si } t_i = t_{i+1}$$

Alors l'état $i+1$ est furtif et doit être filtré : les pas i et $i+1$ sont fusionnés. Ainsi :

$$(\sigma_i, d_i, t_i) \overset{e_i}{\rightarrow} (\sigma_{i+1}, d_{i+1}, t_{i+1}) \overset{e_{i+1}}{\rightarrow} (\sigma_{i+2}, d_{i+2}, t_{i+2})$$

Devient :

$$(\sigma_i, d_i, t_i) \overset{E_i=(e_i,e_{i+1})}{\rightarrow} (\sigma_{i+2}, d_{i+2}, t_{i+2})$$

On peut remarquer que les observateurs d'évènements (définis en II.3.1) ne sont pas affectés par ce filtrage, puisqu'ils ne portent pas sur les valeurs des variables et que les occurrences des évènements sont conservées.

3.2 Expression des calculs à réaliser

La solution technique retenue pour répondre au besoin opérationnel 3 est un ensemble d'indicateurs. Un indicateur est une description d'une mesure à réaliser sur l'évolution de la valeur d'un observateur AltaRica 3.0 lors d'une histoire générée. L'indicateur est à choisir en fonction des calculs à réaliser, parmi un ensemble. Les indicateurs de cet ensemble sont présentés dans l'annexe B.

3.2.a Lien avec le modèle AltaRica 3.0

L'indicateur est relié à un observateur du modèle AltaRica 3.0 : pour chaque histoire, l'indicateur va mesurer un comportement, c'est-à-dire une grandeur (booléenne, nombre ou date) dépendant de l'évolution de la valeur prise par l'observateur durant l'histoire (sa valeur moyenne, la première date à laquelle une valeur a été prise...). Les mesures sur chaque histoire générée par simulation stochastique vont créer une série de mesures : des grandeurs probabilistes sont estimées à partir de ces séries, ce qui constitue le résultat de la simulation (figure II.6).

Suivant l'indicateur retenu, la grandeur mesurée peut être une quantité, une date, ou une valeur booléenne, ce qui a une incidence sur l'estimation de la grandeur probabiliste (voir II.3.3).

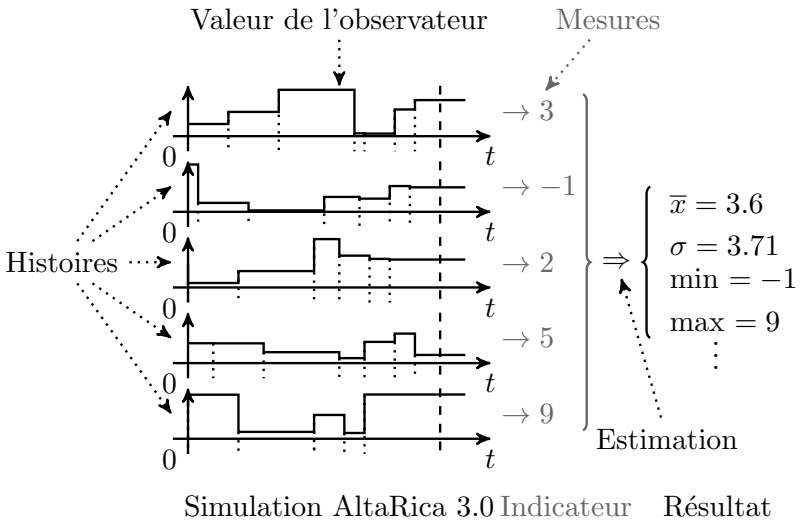

FIGURE II.6 – Relation entre les histoires, les observateurs, les indicateurs et les résultats

Par exemple, pour estimer une MTTF, un indicateur "First Occurrence Date" (voir B.1.2) peut être utilisé, en configurant un observateur du modèle à étudier dont la valeur booléenne indique si le système est en panne. Pour chaque histoire générée par simulation stochastique, l'indicateur va mesurer la première date à laquelle l'observateur prend la valeur *true*. L'ensemble des histoires va ainsi être utilisé pour créer une série de mesures, qui serviront ensuite à l'estimation de la MTTF.

Cette séparation entre la description des calculs à réaliser, et la description des états utiles, permet de répondre au besoin opérationnel 2 : un même indicateur va pouvoir être utilisé pour l'étude de plusieurs modèles (voir II.4.1.b)

3.2.b Ensemble d'indicateurs

Le besoin opérationnel 3 demande que les grandeurs probabilistes classiques de sûreté de fonctionnement soient mesurables. Une étude de ces grandeurs classiques (voir II.5) permet

de lister un ensemble restreint de mesures qui doivent pouvoir être réalisées sur ces évolutions de valeurs afin de couvrir la majorité de ces grandeurs :

- la valeur de l'observateur à une date spécifiée ;
- si l'observateur a pris une valeur spécifique au cours de son histoire, ou si un évènement est survenu ;
- la valeur moyenne de l'observateur pendant l'histoire ;
- le nombre d'occurrence d'un évènement ou de changement de valeur ou de l'observateur pendant l'histoire ;
- le temps moyen entre occurrences ou changement de valeurs de l'observateur pendant l'histoire ;
- la date de première occurrence d'un évènement ou de première prise d'une valeur spécifique de l'observateur pendant l'histoire.

Chacune de ces mesures doit être adaptée au type d'observateur lié : le nombre de changement de valeur est adapté à un observateur numérique, tandis que l'on comptera le nombre d'occurrences d'un évènement, alors que ces deux mesures correspondent au décompte des fronts du chronogramme correspondant.

L'information portée par une histoire filtrée peut être visualisée à l'aide d'un chronogramme, qui représente l'évolution de la valeur de l'observateur au cours du temps simulé. Suivant le type de l'observateur AltaRica 3.0 (booléen, numérique ou d'évènement), la valeur est bien sûr différente (figures II.7,II.8 et II.9).

Certaines mesures ne sont pas possibles pour certains observateurs : par exemple, mesurer la valeur moyenne d'un observateur d'évènement pendant l'histoire n'a pas de sens.

Il est ainsi possible de définir 14 indicateurs, résumés dans le tableau II.1. Pour répondre au besoin opérationnel 4, chaque indicateur est défini et documenté, réduisant le risque d'ambiguïté (ces définitions sont disponibles en annexe B).

Choix technique 4 : Ensemble d'indicateurs

L'ensemble des mesures réalisables est défini.

Limites de l'expressivité

Limiter les possibilités de calculs à un ensemble d'indicateurs, non modifiable lors de l'étude de sûreté de fonctionnement, ne permet pas une expressivité complète et libre des calculs. Une solution serait de permettre la description du calcul à l'aide d'un langage, qui serait alors proche d'un langage de programmation ou de calcul scientifique (générique comme Matlab[43], ou spécialisé comme R[72]). La difficulté de l'utilisation d'un tel langage pour spécifier les calculs viendrait toutefois s'ajouter à celle de la modélisation des solutions techniques en AltaRica 3.0, et plus largement à celle de l'analyse de sûreté de fonctionnement menée.

L'utilisation d'un indicateur parmi un ensemble déjà spécifié présente une expressivité suffisante pour les indicateurs classiques (voir II.5). Toutefois, certaines industries peuvent utiliser des indicateurs probabilistes de sûreté de fonctionnement spécifiques qui ne sont pas calculables par les indicateurs contenus dans cet ensemble. Ces besoins particuliers peuvent

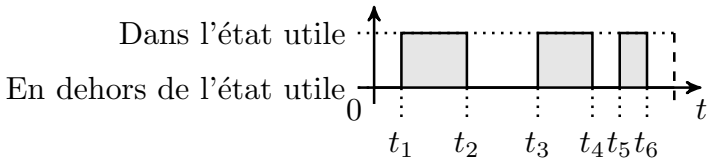

FIGURE II.7 – Chronogramme représentant l'évolution de la valeur d'un observateur booléen lors d'une histoire filtrée

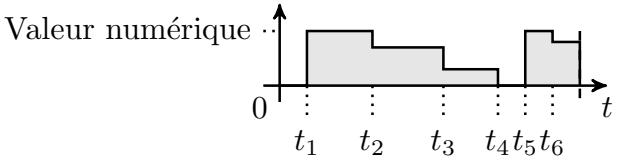

FIGURE II.8 – Chronogramme représentant l'évolution de la valeur d'un observateur numérique lors d'une histoire filtrée

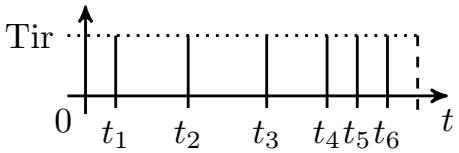

FIGURE II.9 – Chronogramme représentant l'évolution de la valeur d'un observateur d'évènement lors d'une histoire filtrée

justifier d'étendre cet ensemble d'indicateurs, et d'implémenter ces calculs dans une version spécifique du logiciel de simulation stochastique.

3.2.c Choix de l'indicateur à utiliser

Le choix de l'indicateur à utiliser pour l'estimation de grandeurs probabilistes de sûreté de fonctionnement peut faire l'objet de règles ou de motifs, semblables aux "patterns" de modélisations que l'analyste de sûreté de fonctionnement pourra suivre pour obtenir ses modèles AltaRica 3.0 [10].

3.3 Estimation des valeurs probabilistes

Les mesures obtenues d'un indicateur forment une série de valeurs qui ne sont pas une mesure complète de l'information souhaitée du modèle, mais une information partielle : sa valeur n'a été mesurée que pour un nombre fini d'histoires du modèle, alors qu'il existe une infinité d'histoires possibles du modèle. Cette infinité d'histoires constitue la population inconnue, et la série de mesures un échantillon de variables indépendantes.

Observateur \ Mesure	Valeur à la date de mesure	Valeur moyenne pendant l'histoire	Nombre d'occurrences pendant l'histoire
Booléen	Has Value	Sojourn Time	Occurrences
Numérique	Value	Mean Value	Changes
Évènement	*sans objet*	*sans objet*	Firings
Type de la mesure :	dépend de l'observateur	Quantité	

Observateur \ Mesure	Est survenu pendant l'histoire	Temps moyen entre chaque occurrence	Temps avant première occurrence
Booléen	Had Value	Mean Time Between Occurrences	First Occurrence Date
Numérique	*sans objet*	Mean Time Between Changes	*sans objet*
Évènement	Has Been Fired	Mean Time Between Firings	First Firing Date
Type de la mesure :	Booléen	Date	

TABLE II.1 – Synthèse des indicateurs

L'estimation de la grandeur probabiliste va donc correspondre à l'estimation des paramètres d'une loi de probabilité à partir de l'échantillon, et à la détermination de l'intervalle de confiance qui mesure la validité de l'estimation.

Choix technique 5 : Estimation des valeurs

Deux lois de probabilité sont utilisées pour l'estimation :
- La loi de Bernoulli pour les grandeurs booléennes ;
- La loi normale (ou loi de Laplace-Gauss) pour les grandeurs numériques et les dates.

Les résultats de la simulation stochastique sont les estimations du ou des paramètres de la loi de probabilité correspondante, ainsi que la détermination de l'intervalle de confiance qui mesure la validité de l'estimation.

Notations

On utilisera dans la suite les notations suivantes :

n	Nombre d'éléments dans la série
x_i	Valeur du i-ème élément de la série
$\overline{x}$	Moyenne arithmétique de la série
σ	Écart-type de la série
α	Risque de première espèce (par défaut 5%, complément à 1 du taux de confiance)

3.3.a Estimation du paramètre d'une loi de Bernoulli

La loi de Bernoulli est une distribution discrète de probabilités : elle est utilisée pour estimer une information booléenne. Elle ne dépend que d'un paramètre $p \in [0, 1]$, qui est la

probabilité de prendre la valeur 1 (ou *true*). La probabilité de prendre l'autre valeur (0 ou *false*) est son complément : $q = 1 - p$.

L'estimation du paramètre p à partir de la série de mesures booléenne est réalisée en calculant directement la proportion de valeurs *true* dans la série.

L'intervalle de confiance est déterminé par :

$$\left[p - \sqrt{p(1-p)}\frac{t_\alpha}{\sqrt{n}}, p + \sqrt{p(1-p)}\frac{t_\alpha}{\sqrt{n}}\right]$$

avec t_α le quantile d'ordre $1 - \dfrac{\alpha}{2}$ d'une loi normale centrée réduite.

Les résultats de la simulation stochastique pour un indicateur à valeurs booléennes contiennent donc l'estimation du paramètre p et la détermination de l'intervalle de confiance.

3.3.b Estimation des paramètres d'une loi normale

Une loi normale est une distribution continue qui est utilisée pour estimer une information à valeur réelle. Elle dépend de deux paramètres : son espérance $\mu \in \mathbb{R}$ et son écart-type $\sigma \in \mathbb{R}^+$. Sa densité de probabilité est donnée par :

$$f_{\text{normale}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

i Espérance

L'estimation de l'espérance est égale à la moyenne arithmétique de la série. La définition formelle de la moyenne arithmétique d'une série est :

$$\overline{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$

L'intervalle de confiance de l'estimation de l'espérance est déterminé par :

$$\left[\overline{x}\left(1 - \sigma\frac{t_\alpha}{\sqrt{n}}\right), \overline{x}\left(1 + \sigma\frac{t_\alpha}{\sqrt{n}}\right)\right]$$

avec t_α le quantile d'ordre $1 - \dfrac{\alpha}{2}$ d'une loi normale centrée réduite. Ce calcul nécessite celui de l'estimation de l'écart-type, mais est réalisé à la fin de l'estimation.

ii Écart-type

L'estimation de l'écart-type se fait en calculant, sur la série de mesures :

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}}$$

Cette équation peut être écrite :

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n} - \overline{x}^2}$$

La valeur estimée de l'écart-type peut faire l'objet d'un encadrement par un intervalle de confiance, mais l'écart-type n'étant pas utilisé classiquement pour estimer des grandeurs probabilistes de sûreté de fonctionnement, le calcul de cette intervalle de confiance ne présente pas ici d'intérêt particulier.

3.3.c Estimation des dates

L'estimation de la loi de probabilité des informations représentant des dates est réalisée en supposant que la loi correspondante est une loi normale. Ce choix est fait par expérience : la loi normale a été trouvée adaptée pour modéliser des phénomènes issus de plusieurs évènements aléatoires, ce qui est le cas pour les études de sûreté de fonctionnement.

Un problème apparaît toutefois pour l'estimation de la loi de probabilité lorsque le phénomène étudié par simulation stochastique est mesuré par une date. Les histoires ne sont simulées que jusqu'à la date correspondant au temps de mission. Les mesures des indicateurs ne sont possibles que si le phénomène à mesurer se produit avant le temps de mission : si le phénomène ne s'est pas produit, il n'a pas de date d'occurrence.

L'estimation de la date moyenne est alors réalisée avec un échantillonnage tronqué. La figure II.10 illustre les conséquences d'une telle troncature : les estimations de l'espérance et de l'écart-type (μ et σ estimés) sont éloignées des paramètres de la loi normale réelle (μ et σ réels). Les estimations des paramètres sont correctes : c'est le choix d'estimer par une loi normale non tronquée qui n'est alors pas adapté.

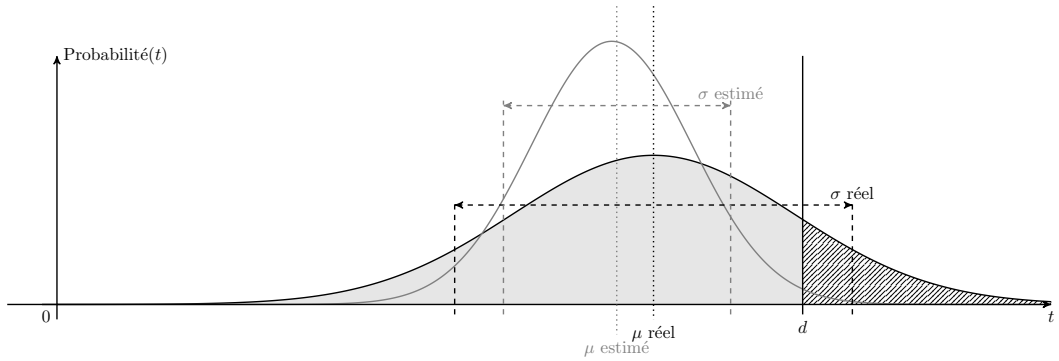

FIGURE II.10 – Écarts entre une loi normale et son estimation à partir d'un échantillonnage tronqué

Plusieurs solutions peuvent pallier ce problème :

- Un choix plus judicieux serait d'estimer les paramètres d'une loi normale tronquée. Mais la loi est tronquée à cause des choix de l'analyste de sûreté de fonctionnement, pas par propriété du modèle : les paramètres estimés ne seraient pas pertinents pour la suite de l'étude de sûreté de fonctionnement ;
- La différence de résultats entre l'estimation d'une loi normale tronquée ou non est négligeable si l'écart-type est négligeable devant l'écart entre la troncature et l'espérance. En configurant un temps de mission suffisamment grand par rapport à l'espérance du phénomène qu'il souhaite estimer, l'analyse de sûreté de fonctionnement limitera les conséquences de cette troncature, mais au prix d'un coût de simulation plus important. Si les résultats de la simulation stochastique comportent la proportion d'histoires pour lesquelles l'évènement ne s'est pas produit (et donc pour lesquelles la mesure n'est pas disponible), l'analyste peut évaluer si les estimations des paramètres d'une loi normale sont représentatives du modèle étudié, ou s'il doit configurer un temps de mission plus grand.

3.3.d Informations sur la série de mesures

En plus des estimations probabilistes, des informations supplémentaires sur la distribution de la série de mesures peuvent être fournies comme résultats de la simulation stochastique : valeurs minimum et maximum, quantiles... Dans le cas de séries de mesure contenant des valeurs "non disponible", la proportion de telles valeurs est aussi un résultat de la simulation stochastique. Ces informations ne concernent que les mesures à valeurs réelles, puisque la distribution de la série de mesure à valeurs booléenne est entièrement connue par l'estimation du paramètre de la loi de Bernoulli.

4 Démarche d'utilisation des indicateurs

Dans le contexte d'une étude de sûreté de fonctionnement en phase d'avant-projet, l'analyste de sûreté de fonctionnement doit évaluer des grandeurs probabilistes de façon à comparer des solutions techniques proposées par des analystes fonctionnels, et choisir celles qui seront utilisées pour la suite de la conception, par rapport à des exigences dysfonctionnelles. Pour cela, il a à sa disposition des outils de modélisation AltaRica 3.0, et un outil de simulation stochastique de modèle AltaRica 3.0.

Dans cette partie, une démarche permettant de mener une étude de sûreté de fonctionnement par simulation stochastique de modèles AltaRica 3.0, montrant l'utilisation d'indicateurs, est proposée. Elle est présentée graphiquement par la figure II.11. Cette démarche utilise de façon importante l'orientation prototype du langage AltaRica 3.0 [57].

4.1 Déroulement d'une étude de sûreté de fonctionnement

Les informations d'entrée fournies à l'analyste de sûreté de fonctionnement sont :

- des descriptions des solutions techniques proposées. Ces descriptions comprennent les niveaux architecturaux et fonctionnels, et peuvent donc être sous diverses formes (modèles architecturaux, documents constructeurs, ...), et provenir de différentes sources. Des informations sur les comportements dysfonctionnels des composants peuvent faire partie de ces informations.
- des exigences de sûreté de fonctionnement que ces solutions techniques doivent satisfaire, et possiblement optimiser.

À partir de ces informations, l'analyste doit construire les modèles AltaRica 3.0, configurer la simulation stochastique (indicateurs) et en interpréter les résultats.

4.1.a Construction d'une bibliothèque de composants AltaRica 3.0

Des solutions techniques d'un même projet peuvent partager des sous-systèmes communs, d'autant plus si elles sont des variantes d'une même solution technique. Il est donc pertinent, pour diminuer le coût de l'étude de sûreté de fonctionnement, de mutualiser la modélisation de ces différents composants et sous-systèmes, à partir des descriptions des solutions techniques proposées. La bibliothèque de composants peut aussi être issue d'études de sûreté de fonctionnement précédentes, permettant la réutilisation des efforts de modélisation.

AltaRica 3.0 est un langage orienté prototype : il permet la construction de bibliothèques de modèles de composants et de sous-systèmes, appelés classes. Il possède de plus des primitives permettant d'ajouter, modifier, ou copier ces classes, comme un code de programmation

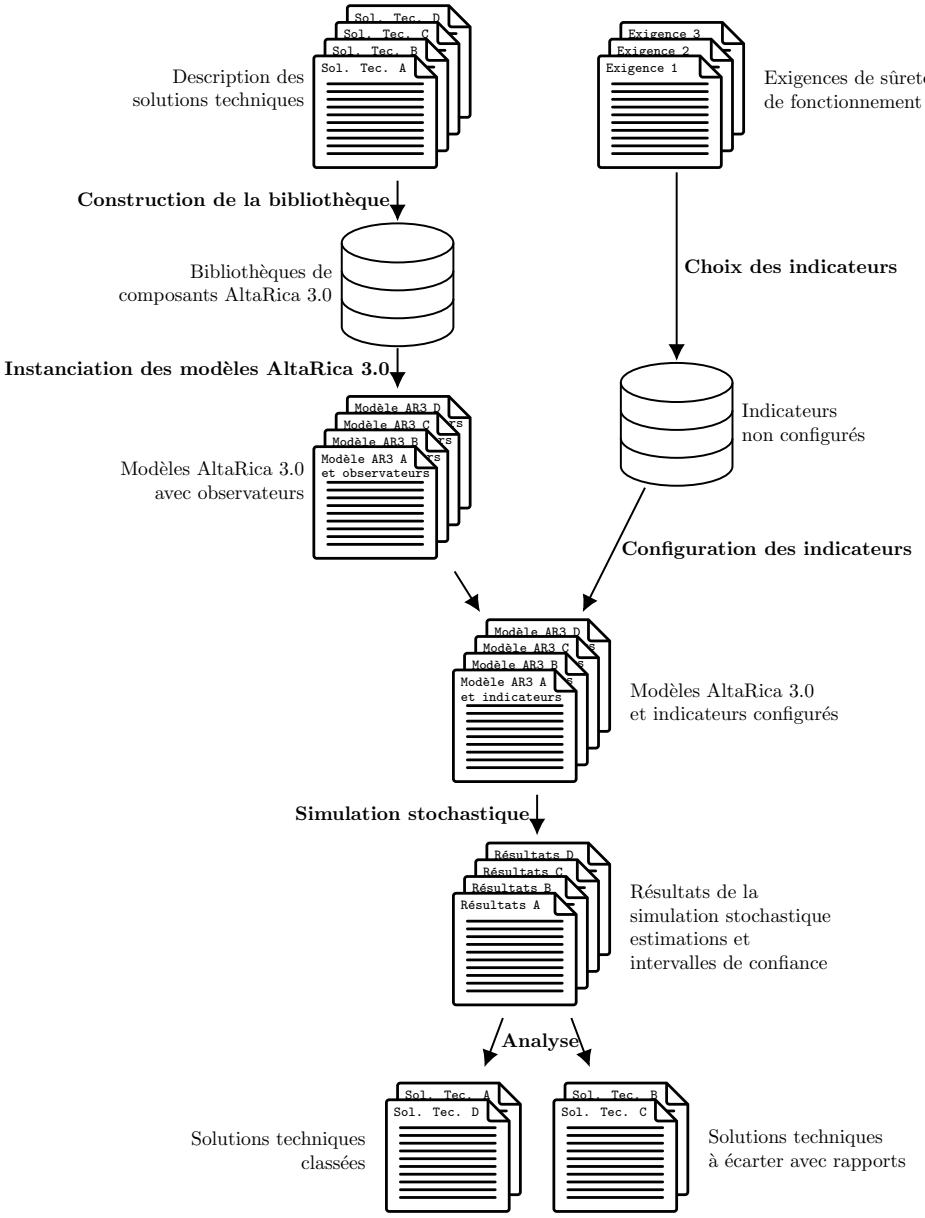

FIGURE II.11 – Déroulement d'une étude de sûreté de fonctionnement

pourrait être factorisé (héritage, surcharge…) : les sous-systèmes et composants n'ont donc pas à être strictement identiques pour pouvoir utiliser le même modèle dans une bibliothèque.

Informations partielles

Lors d'une étude de conception d'avant-projet, des informations peuvent ne pas être disponibles. C'est par exemple souvent le cas des taux de défaillance de composants, dont les valeurs sont nécessaires pour les études de sûreté de fonctionnement. L'analyste doit alors évaluer ces valeurs, de façon approchée, à partir de composants comparables, de base de données rassemblant ce type d'informations, et de son expérience. Comme l'un des besoins fonctionnels est

de comparer des solutions techniques, il est utile de définir les valeurs des caractéristiques dysfonctionnelles lors de la modélisation des composants en bibliothèque. Ainsi, tous les modèles de solution technique pourront utiliser les mêmes valeurs lorsqu'ils utiliseront les mêmes composants. La représentativité des valeurs choisies pour la modélisation aura alors une importance moindre, puisque les grandeurs probabilistes estimées avec les différents modèles seront comparées entre elles et non retenues comme grandeurs absolues. La comparaison des conséquences de différentes architectures de solutions techniques sera possible, même avec des informations partielles ou imprécises.

La construction de cette bibliothèque de composants permet à l'analyste de sûreté de fonctionnement de prévoir comment les comportements intéressants à mesurer pour son étude seront représentés (par des états, des évènements AltaRica 3.0...), ce qui sera utile pour le choix des indicateurs.

4.1.b Choix des indicateurs

L'analyste de sûreté de fonctionnement extrait des exigences de sûreté de fonctionnement les critères qui correspondent à des grandeurs probabilistes. À partir de ces grandeurs probabilistes, et de ses choix de modélisation fait précédemment, il choisit les indicateurs qu'il va utiliser pour estimer ces grandeurs probabilistes.

Ces indicateurs seront utilisés pour l'analyse des différentes solutions techniques, ils ne sont donc pas encore configurés, c'est-à-dire reliés à des observateurs AltaRica 3.0.

4.1.c Instanciation des modèles AltaRica 3.0

Pour chaque solution technique, et variante de solution technique, à étudier, l'analyste de sûreté de fonctionnement va instancier un modèle. Il va pour cela utiliser la bibliothèque créée, en instanciant les composants et sous-systèmes concernés, et en les associant suivant l'architecture de la solution technique modélisée. Les composants et sous-systèmes qui ne sont pas communs avec d'autres solutions techniques, et qui ne font donc pas partie de la bibliothèque, peuvent être directement instanciés lors de leur déclaration (prototypes).

L'analyste doit créer, pour chaque modèle de solution technique, les observateurs AltaRica 3.0 qui seront utilisés pour son étude. Les architectures et les composants des différentes solutions techniques étant possiblement différents, les expressions des observateurs (c'est-à-dire les variables et évènements utilisés dans les expressions des observateurs) peuvent être différents : l'analyste doit donc les définir de façon individuelle, dans chaque modèle, en s'assurant toutefois qu'ils correspondent à des grandeurs comparables entre elles.

4.1.d Configuration des indicateurs

Une fois les modèles des différentes solutions techniques instanciés, l'analyste de sûreté de fonctionnement peut configurer les indicateurs en les reliant aux observateurs correspondant pour chaque modèle. Cette étape est normalement rapide puisque sauf exception les modèles, et donc les observateurs utilisés, sont similaires.

4.1.e Simulation stochastique

Une fois les modèles AltaRica 3.0 conçus et les indicateurs configurés, l'analyste de sûreté de fonctionnement paramètre la simulation stochastique en indiquant les temps de mission (qui seront les dates auxquelles les indicateurs effectueront leurs mesures), et le nombre d'histoires à générer.

Le simulateur stochastique AltaRica 3.0 génère alors les histoires. Les indicateurs réalisent leurs mesures sur les histoires générées et filtrées, puis calculent les résultats demandés.

4.1.f Analyse des résultats

L'analyste de sûreté de fonctionnement peut alors relier les résultats de la simulation stochastique, c'est-à-dire les estimations des paramètres des lois de probabilités, aux grandeurs probabilistes des exigences de sûreté de fonctionnement.

Il a alors la possibilité d'écarter les solutions techniques non satisfaisantes, et de classer celles restantes entre elles par rapport aux critères de sûreté de fonctionnement.

Les résultats de la simulation stochastique comportent des estimations de l'intervalle de confiance. Si celles-ci ne sont pas suffisantes pour prendre des décisions, l'analyste peut décider de lancer une nouvelle étude. Les intervalles de confiance sont de façon presque sûre décroissants avec le nombre de mesures, donc avec le nombre d'histoires générées : l'analyste devra donc configurer un nombre d'histoires à générer plus important pour réduire l'intervalle de confiance.

4.2 Suite de la conception

Les résultats de l'étude de sûreté de fonctionnement peuvent être utilisés pour la suite de la conception du système. Lors des prochaines itérations du cycle de conception, des études de sûreté de fonctionnement seront à mener sur des solutions techniques qui seront des raffinements de celles retenues. L'analyste de sûreté de fonctionnement pourra alors réutiliser sa bibliothèque de composants, ses modèles AR3 et ses choix d'indicateurs, qu'il pourra lui aussi raffiner. Cette capitalisation des modèles permettra un gain de temps lors des prochaines étapes de conception.

5 Expériences

La définition des indicateurs permet de spécifier les résultats souhaités d'une analyse de sûreté de fonctionnement par simulation stochastique. Ces résultats sont à comparer aux niveaux des exigences de sûreté de fonctionnement, portant sur des grandeurs probabilistes. Des définitions des grandeurs probabilistes classiquement utilisées dans les exigences de sûreté de fonctionnement peuvent être trouvées dans la littérature, notamment [58].

La pertinence des indicateurs retenus pour la simulation stochastique dépend de leur adéquation avec ces grandeurs probabilistes. Dans cette partie, des exemples de choix d'indicateurs pour estimer ces grandeurs sont proposés.

5.1 Disponibilité, fiabilité

La disponibilité (availability) $A(t)$ est la probabilité que le système soit en état de bon fonctionnement à la date t. L'indisponibilité (unavailability) $U(t)$ est son complément :

$$U(t) = 1 - A(t)$$

La fiabilité (reliability) $R(t)$ est la probabilité que le système ai fonctionné sans interruption de la date de démarrage $t = 0$ à la date t. La défiabilité (unreliability) $F(t)$ est son complément :

$$F(t) = 1 - R(t)$$

Ces grandeurs sont à mesurer au temps de mission, c'est-à-dire à la fin de la phase de fonctionnement étudiée. Suivant l'étude menée, il peut être utile de les mesurer à intervalles réguliers : les courbes obtenues peuvent par exemple permettre de dimensionner des fréquences de maintenance.

La modélisation AltaRica 3.0 doit proposer un moyen de savoir si le modèle simulé est dans un état de bon fonctionnement ou non, par le biais d'un observateur qui est alors booléen. L'indicateur "Has value" sur cet observateur va mesurer pour chaque histoire si le modèle est dans l'état de bon fonctionnement à la date de mesure. L'estimation probabiliste permet alors d'obtenir l'espérance d'avoir le modèle en état de bon fonctionnement à la date de mesure, ce qui correspond à la disponibilité.

De même, l'indicateur "Had value" sur cet observateur mesure si le modèle a été dans un autre état que celui de bon fonctionnement pendant un temps non nul, c'est-à-dire s'il y a eu une interruption du bon fonctionnement. L'espérance de cette mesure (estimée grâce à l'échantillonnage réalisé) est correspond à la défiabilité à la date de mesure, ce qui permet d'en déduire la fiabilité.

5.2 Temps moyens

Plusieurs temps moyens sont utilisés pour quantifier les performances de sûreté de fonctionnement d'un système.

5.2.a Mean Up Time et Mean Down Time

Le temps moyen de bon fonctionnement (Mean Up Time) $MUP(t)$ et le temps moyen de panne (Mean Down Time) $MDT(t)$ sont respectivement les moyennes de temps que le système a passé dans un état de bon fonctionnement et de panne pendant le temps de mission, c'est-à-dire entre la date de démarrage $t = 0$ et la date de mesure $t = d$. L'un est donc le complément à t de l'autre :

$$MUP(t) = t - MDT(t)$$

Pour les estimer, un observateur booléen va être défini dans le modèle AltaRica 3.0 pour obtenir l'information de bon fonctionnement du modèle simulé à chaque moment de l'histoire générée. L'indicateur "Sojourn Time" (temps de séjour) sur cet observateur va permettre d'estimer l'espérance du temps de séjour du modèle dans l'état de bon fonctionnement. Cette estimation correspondra au temps moyen de bon fonctionnement exigé.

5.2.b Mean Time To Failure et Mean Time Between Failure

Le temps moyen avant panne (Mean Time To Failure) $MTTF(t)$ et le temps moyen entre pannes (Mean Time Between Failure) $MTBF(t)$ sont respectivement le temps moyen entre la date de démarrage $t = 0$ et la date de première panne, et le temps entre chaque panne survenue entre la date de démarrage $t = 0$ et la date de mesure $t = d$.

Le même observateur booléen que celui utilisé pour l'estimation du temps moyen de bon fonctionnement ou du temps moyen de panne peut être utilisé pour obtenir l'information de bon fonctionnement du modèle simulé : l'information pertinente ici étant la survenance d'une panne, on va s'intéresser aux occurrences (aux "fronts montants" de la valeur de cet observateur) qui indiquent le passage de l'état de bon fonctionnement à celui de panne, et donc la survenance d'une panne. Les mesures peuvent être réalisées par les indicateurs "First occurrence date" et "Mean time between occurrences", et les estimations des espérances réalisées correspondront au temps moyen avant panne et du temps moyen entre pannes.

5.2.c Mean Time To Repair

Le temps moyen de réparation (Mean Time To Repair) $MTTR(t)$ est le temps moyen que le système passe dans l'état de panne, entre la survenance d'une panne et sa réparation. En utilisant le même observateur booléen que ci-dessus, il est possible d'obtenir l'estimation de deux informations :

- le temps passé dans l'état de panne en moyenne dans chaque histoire, avec un indicateur "Sojourn time" ;
- le nombre d'occurrences de pannes moyen par histoire, avec un indicateur "Occurrences".

La division du premier par le second donne une quantité qui peut s'approcher du temps moyen de réparation. Toutefois, le résultat obtenu peut mener à discussion.

Ce phénomène peut être illustré par un modèle pour lequel deux histoires sont générées par simulation stochastique :

- la première histoire ne comporte qu'une seule panne et donc une seule réparation, espacées de 100 heures. La mesure de l'indicateur "Sojourn time" est 100, la mesure de l'indicateur "Occurrences" est 1 (car une seule panne). Le temps moyen de réparation sur cette unique histoire est de 100 heures.
- la seconde histoire comporte 99 pannes et autant de réparations, espacées chacune d'un temps négligeable. La mesure de l'indicateur "Sojourn time" est proche de 0, la mesure de l'indicateur "Occurrences" est 99. Le temps moyen de réparation sur cette unique histoire est proche de 0 heure.

L'estimation du temps moyen de réparation peut alors être réalisée par deux calculs :

- par la division du temps de séjour par le nombre d'occurrences : la moyenne estimée du temps passé dans l'état de panne est de 50 heures, tandis que la moyenne estimée du nombre d'occurrences de pannes est de 50. Le temps moyen de réparation calculé vaut alors 1 heure.
- par moyenne des temps moyens de réparation sur chaque histoire : le temps moyen de réparation vaut alors 50 heures.

Cette différence est celle entre une moyenne générale, et la moyenne des moyennes, appelée aussi paradoxe de Simpson [29]. Cet exemple, caricatural, montre l'importance de la

définition de la quantité que l'on souhaite estimer, à laquelle l'analyste de sûreté de fonctionnement doit prêter attention.

Dans notre cas, la différence va converger vers 0 avec l'augmentation du nombre d'histoires générées, hormis pour des modèles particuliers (automates à comportements exclusifs). Il n'y a pas d'indicateur permettant de mesurer le temps moyen de réparation en utilisant la seconde définition. Si une analyse de sûreté de fonctionnement nécessite une telle mesure, un indicateur spécifique devra être défini et implémenté dans l'outil de simulation stochastique.

5.3 Densité de défaillance, taux de défaillance

La densité de défaillance $f(t)$ est la dérivée temporelle de la défiabilité.

$$f(t) = \frac{dF(t)}{dt}$$

Le taux de défaillance $r(t)$ est la probabilité que le système défaille entre les instants t et $t + dt$, c'est-à-dire que le système soit en état de bon fonctionnement à la date t et soit en panne à la date $t + dt$, pour une durée dt suffisamment petite.

$$r(t) = \lim_{dt \to 0} \frac{dp(\text{le système fonctionne à t et est en panne à t+dt})}{dt}$$

La densité et le taux de défaillance jouent un rôle important dans les pratiques industrielles. Elles sont facilement calculables par résolution de modèles basés sur des chaînes de Markov, qui sont des résolutions d'équations différentielles et permettent donc d'obtenir la dérivée temporelle d'une valeur. La simulation stochastique portant sur des estimations de lois de probabilité à partir d'échantillons, il n'est pas possible d'obtenir par ce moyen des fonctions continues et dérivables pour les calculer.

Une solution est d'approximer ces dérivées par des calculs sur des intervalles plus grands, que l'on mesure en un nombre limité de dates par simulation stochastique :

$$f(t) = \frac{dF(t)}{dt} \approx \frac{F(t + \Delta t) - F(t)}{\Delta t}$$

L'indicateur utilisé pour la défiabilité peut alors être utilisé pour estimer la densité de défaillance. Par la même méthode, l'indicateur utilisé pour l'indisponibilité peut être utilisé pour estimer le taux de défaillance.

5.4 Facteur d'importance de diagnostic

Le facteur d'importance de diagnostic $DIF(t)$ d'un composant est la probabilité que le composant soit défaillant à la date t, étant donné que le système est en panne à cette même date t. Une estimation de cette probabilité peut être facilement obtenue en définissant deux couples d'observateurs-indicateurs :

- Un observateur booléen donnant l'information "le système est en panne", et un indicateur "Has value" sur cet observateur : on aura alors l'indisponibilité du système ;
- Un observateur booléen donnant l'information "le système est en panne ET le composant est défaillant", et un indicateur "Has value" sur cet observateur.

Les deux observateurs ne sont pas indépendants : le second observateur ne peut avoir pour valeur *true* que si le premier a aussi pour valeur *true*. On peut donc obtenir une estimation de

$DIF(t)$ en fonction des espérances estimées de la disponibilité du système $U(t)$ et du second couple $UC(t)$:

$$DIF(t) = \frac{UC(t) - (1 - U(t))}{U(t)}$$

5.5 Performance de production

La performance de production $PA(t)$ est le ratio entre un niveau de production d'un système observé $OPL(t)$ au temps t (donc la production totale entre la date de démarrage $t = 0$ et la date de mesure), et un niveau de production de référence $RPL(t)$ (la production attendue ou prévisionnelle sur la même période de temps). Habituellement, le calcul de $OPL(t)$ est suffisant pour déterminer $PA(t)$, le niveau de production de référence étant déterminé par un autre modèle.

Le niveau de production est classiquement calculé comme une fonction de récompense $r(t)$: en AltaRica 3.0, un observateur à valeur entière ou réelle est donc adapté. On a donc :

$$OPL(t) = \int_0^t r(t')dt'$$

Un indicateur "Mean value" sur cet observateur va permettre d'estimer l'espérance de production moyenne au cours de la période. En multipliant par la durée de la période, on obtiendra une estimation de $OPL(t)$.

5.6 Optimisation de la maintenance

L'optimisation de la maintenance vise à définir des stratégies de maintenance pour minimiser à la fois le temps de panne, la probabilité d'un accident catastrophique, et le coût total des opérations, tout en maximisant la production et donc le profit tiré d'un système. Deux grandeurs sont importantes pour étudier ce type d'optimisation : le temps moyen de panne du système (voir II.5.2.a) et le nombre moyen d'interventions de maintenance.

Une intervention de maintenance peut correspondre, dans le modèle AltaRica 3.0, à plusieurs éléments :

- Le tir d'une transition "réparation". Un observateur d'évènement sur l'évènement correspondant à cette transition permettra de mesurer la survenance d'une intervention dans l'histoire générée. Un indicateur "Firings" permettra d'estimer le nombre moyen de tirs de la transition par histoire générée, et donc le nombre moyen d'interventions de maintenances.
- Un changement de valeurs de variables d'état, conséquence du tir de diverses actions. Un observateur booléen permettra de mesurer ces changements de valeurs, une occurrence (passage à l'état *true*) signifiant le début d'une intervention. Un indicateur "Occurrences" permettra alors d'estimer le nombre moyen d'interventions par histoires.

5.7 Durées

Les systèmes mécatroniques ont souvent des comportements dépendant du temps : il peut être acceptable de perdre une fonction, à la condition que cette perte ne soit pas trop d'une durée trop longue. Par exemple, il est acceptable de perdre un signal GPS pour une courte durée, mais il une perte trop longue peut avoir des conséquences importantes sur la qualité du

positionnement d'un système. De même, une alarme traitée rapidement est acceptable, mais pas si ce temps dépasse une certaine limite.

Les observateurs et les indicateurs proposés ne permettent pas d'obtenir directement ce type d'information si le comportement (alarme non traitée rapidement) n'est pas implémenté dans le modèle AltaRica 3.0. Une extension possible de l'ensemble des indicateurs serait donc de pouvoir mesurer des comportements dépendant du temps passé dans un état.

6 Conclusion

La spécification des résultats d'une simulation stochastique doit faire le lien entre les grandeurs probabilistes à mesurer, et les éléments du modèle simulé. La solution technique proposée pour la simulation stochastique de modèles AltaRica 3.0 est un ensemble d'indicateurs et l'utilisation des observateurs AltaRica 3.0 étendus aux évènements. Un indicateur mesure un comportement, vu à travers un observateur AltaRica 3.0, du modèle simulé au cours des histoires générées par simulation stochastique, pour en obtenir une série de mesures. Ces mesures forment un échantillonnage du comportement que l'on souhaite quantifier, et permettent de l'estimer. Le résultat obtenu est ainsi formé d'une estimation des paramètres de la loi de probabilité correspondante, associée à un intervalle de confiance permettant d'apprécier la qualité du résultat obtenu.

Une grande partie des grandeurs classiques de sûreté de fonctionnement peuvent être obtenus par un ou plusieurs indicateurs de l'ensemble proposé. Toutefois, ce dernier ne couvre pas tous les cas d'utilisation, soit par limitation de la simulation stochastique, soit par absence d'un indicateur correspondant. Des extensions de l'ensemble d'indicateurs sont possibles pour permettre l'estimation de ces grandeurs.

Chapitre III

Implémentation d'un simulateur stochastique

1 Introduction

La simulation stochastique est un outil dont les résultats sont des estimations de grandeurs probabilistes sur le comportement du modèle, représentant des performances du système réel. Ces résultats sont constitués de valeurs associées à des intervalles de confiance. Par principe, la simulation stochastique permet d'obtenir rapidement des résultats avec un intervalle

de confiance important, ou des résultats avec un petit intervalle de confiance au prix d'importants moyens de calcul. L'obtention d'un intervalle de confiance restreint avec des moyens de calcul faibles n'étant pas réaliste (le problème étant #P-difficile [74]), le compromis entre ces deux grandeurs est un choix réalisé par l'analyste, en fonction de ses contraintes.

Une implémentation logicielle réfléchie peut toutefois permettre de rapprocher les capacités de l'outil de simulation stochastique des demandes industrielles, en améliorant ses performances de calcul.

Une implémentation dite naïve, au plus près des principes de la simulation stochastique et de la simulation de modèles AltaRica 3.0 telle que décrite dans la spécification, est présentée en III.2. Cette implémentation sert de point de départ à une démarche d'amélioration de la performance, utilisée pour le simulateur stochastique du projet OpenAltaRica, proposée en III.3.

2 Implémentation naïve

Un simulateur stochastique de modèle AltaRica 3.0 peut être implémenté de façon naïve, c'est-à-dire en suivant la spécification du langage AltaRica 3.0 [8] et en décomposant la simulation stochastique en plusieurs fonctions élémentaires.

Pour simuler stochastiquement un modèle AltaRica 3.0 il faut tout d'abord être en mesure de le simuler, c'est-à-dire de créer une histoire d'un modèle, ce qui est présenté en III.2.1. Cette fonction de simulation peut ensuite être utilisée dans une fonction de simulation stochastique, dont le rôle est de générer un grand nombre d'histoires du modèle. L'implémentation naïve d'une telle fonction est décrite en III.2.2. Enfin, des mesures sont réalisées sur les histoires générées, pour ensuite estimer les grandeurs probabilistes de sûreté de fonctionnement souhaitées : une discussion sur l'implémentation de cette fonction est l'objet de III.2.3.

2.1 Simulation d'un modèle AltaRica 3.0

Un modèle AltaRica 3.0 après mise à plat est une description d'une machine à états temporisée et stochastique GTS (voir A). La simulation d'un GTS est décrite par l'algorithme 3.

2.1.a État initial

Les trois premières étapes de l'algorithme de simulation initialisent la machine à état. Les valeurs initiales des variables d'état sont définies dans le modèle. Celles des variables de flux et des observateurs sont calculées en utilisant les assertions et les expressions des observateurs.

2.1.b Échéancier

Le rôle de l'échéancier est de gérer la survenance de chaque évènement du modèle. Il doit être capable :

- d'indiquer quel est le prochain évènement devant survenir (celui qui a le plus petit délai) ;
- de supprimer de son catalogue d'évènements ceux qui sont liés à des transitions qui ne sont plus tirables (dont la garde n'est plus vérifiée) ;

Algorithme 3 : Simulation d'un GTS

```
1  Assigner aux variables d'état leurs valeurs initiales;
2  Propager les valeurs aux variables de flux en utilisant les assertions;
3  Calculer les valeurs des observateurs;
4  tant que la condition d'arrêt n'est pas vérifiée faire
5  |  Mettre à jour l'échéancier;
6  |  Trouver le prochain évènement programmé dans l'échéancier;
7  |  si plusieurs évènements sont programmés à la même date alors
8  |  |  L'un d'entre eux est choisi aléatoirement;
9  |  fin
10 |  Tirer la transition associée à cet évènement;
11 |  Appliquer l'action de la transition sur les variables d'état;
12 |  Propager les valeurs aux variables de flux en utilisant les assertions;
13 |  Calculer les valeurs des observateurs;
14 fin
```

- d'ajouter à son catalogue les évènements associés à des transitions qui deviennent tirables, avec leur date de tir déterminée aléatoirement en utilisant la loi probabiliste de l'évènement concerné.

Ce mécanisme doit gérer chaque évènement du modèle, et vérifier à chaque pas (chaque itération de la boucle *tant que*) la valeur de chaque garde. Une implémentation naïve peut donc conduire à une complexité très importante, et donc à une consommation de ressources de calcul conséquente.

2.1.c Propagation des assertions

Les valeurs des variables de flux sont mises à jour à chaque pas de simulation, par propagation des assertions.

Les assertions sont des assignations, qui peuvent être conditionnelles, des variables de flux. Les expressions des assignations dépendent des valeurs des variables d'état et des valeurs d'autres variables de flux. Sous certaines conditions (modèles acausaux) des "boucles" d'assignations peuvent ainsi être formées : A dépend de B, qui dépend de C, qui dépend de A. La propagation des assertions (voir A.2.2) vise à déterminer, pour chaque pas de simulation, la valeur de chaque variable de flux. Cette valeur ne dépend pas de sa valeur précédente : les variables de flux ne sont pas des mémoires comme les variables d'état.

Au début de la propagation des assertions, toutes les variables de flux sont considérées indéfinies, seules les valeurs des variables d'état sont connues. Ensuite, l'ensemble des assertions est parcouru en boucle. Pour chaque assertion :

- Si l'assertion ne dépend que de variables définies :
  - si la variable à assigner n'est pas encore définie, alors sa valeur est calculée en utilisant cette assertion ;
  - si la variable à assigner est déjà définie, alors sa valeur est comparée à celle obtenue en utilisant cette assertion : en cas d'écart, une erreur est alors signalée.
- Si l'assertion dépend d'au moins une variable indéfinie, elle est ignorée pour cette itération.

Le parcours en boucle de l'ensemble des assertions est arrêté si toutes les assertions ont été utilisées, si toutes les variables de flux ont été définies, ou si après une itération sur l'ensemble des assertions aucune autre variable de flux ne devient définie. Les variables de flux non encore définies se voient alors affectées de leur valeur par défaut (définie dans le modèle). Les assertions sont alors toutes évaluées pour vérifier qu'il n'y a pas d'incohérence dans les valeurs des variables (dans le cas contraire, cela indique que le modèle est défectueux car il contient deux assertions incohérentes, et une erreur est signalée).

La vitesse de propagation des assertions dépend du modèle : si toutes les assertions n'utilisent que des variables d'état, elles seront toutes immédiatement dépendantes de variables définies, et toutes les valeurs des variables de flux seront calculées en une seule itération. Dans le cas d'un modèle data-flow, c'est-à-dire dont l'ensemble des assertions ne présente pas de boucle, un ordre de propagation des assertions en une seule passe peut être déterminé : cet ordre sera identique pour chaque pas de simulation, quel que soit l'état du modèle, et permettra aussi un calcul en une seule itération [53]. Mais l'une des avancées d'AltaRica 3.0 est la possibilité de modéliser des systèmes par des modèles non data-flow : il n'est donc pas possible, dans le cas général, de déterminer un tel ordre unique de propagation des assertions.

Sous l'hypothèse que le nombre d'assertions n est du même ordre de grandeur que le nombre de variables de flux, et que le nombre de variables utilisées par chaque assertion est négligeable devant n, l'algorithme de résolution mène à une complexité dans le meilleur des cas de l'ordre de $O(n)$ opérations élémentaires, et dans le pire des cas de $O(n^2)$ opérations élémentaires.

Ce mécanisme de résolution itérative, en testant systématiquement chaque assertion jusqu'à la résolution complète, peut donc consommer une part importante des ressources de calcul.

2.2 Simulation stochastique

Le principe de la simulation stochastique est de générer un nombre important d'histoires (résultat d'une simulation du modèle), et d'en obtenir des estimations probabilistes sur son comportement. Le mécanisme de simulation doit donc garder en mémoire les résultats de chaque histoire (traces d'une simulation, sous forme de séquence GTS, voir II.2.4.b). Ces traces doivent contenir toutes les informations sur les histoires qui pourraient être nécessaires aux mesures des indicateurs (partie III.2.3).

AltaRica 3.0 permet de définir des observateurs dans le modèle : ils spécifient les grandeurs remarquables, c'est-à-dire celles qui peuvent être utiles à connaitre du modèle, et sont donc potentiellement nécessaires aux calculs des indicateurs. Les traces peuvent donc être limitées aux évolutions des valeurs de ces observateurs. Toutefois, un modèle d'un système pouvant être créé pour différentes études, tous les observateurs ne sont pas systématiquement utiles : le mécanisme de simulation n'étant pas en mesure de définir lesquels le sont, ni quelles valeurs sont utiles, les traces contiennent des informations superflues, ce qui est une cause de consommation de ressources de calcul et de mémoire supérieure au nécessaire.

2.3 Mesures des indicateurs, et estimation des grandeurs probabilistes

Les mesures des indicateurs sont réalisées à partir des traces, a posteriori de leur génération (post-traitement). Ce mécanisme doit lire ces traces et filtrer les histoires (voir II.3.1.b), pour

effectuer les mesures. Ensuite, l'estimation des lois probabilistes sur les ensembles de mesures permet d'obtenir les indicateurs de sûreté de fonctionnement demandés avec leurs intervalles de confiance.

La quantité de données étant importante, car proportionnelle au nombre d'histoires générées par la simulation stochastique, cette étape peut être coûteuse en mémoire et en ressources de calcul.

3 Amélioration des performances

L'implémentation naïve décrite précédemment permet d'obtenir un simulateur stochastique de modèles AltaRica 3.0 fonctionnel. Toutefois, les performances en temps de calcul obtenues sont extrêmement basses. Afin d'obtenir un outil utilisable dans un contexte industriel avec ses contraintes économiques, il est nécessaire de les améliorer.

L'architecture retenue pour le simulateur stochastique du projet AltaRica 3.0 est présentée en III.3.1, et la démarche d'ingénierie logicielle utilisée pour améliorer ses performances est décrite en III.3.2.

3.1 Architecture

L'outil de simulation stochastique utilise comme données d'entrée un modèle AltaRica 3.0, un ensemble d'indicateurs, ainsi qu'un paramétrage de la simulation stochastique (critère d'arrêt en temps de mission et nombre d'histoires à générer). Les données de sortie attendues sont les estimations probabilistes des mesures des indicateurs.

La performance de l'outil de simulation stochastique pertinente pour une utilisation industrielle est le temps écoulé entre le lancement de l'outil avec les données d'entrée fournies, et l'obtention des données de sortie. L'implémentation naïve découpe la simulation stochastique d'un modèle AltaRica 3.0 en différentes fonctions. La performance concerne l'ensemble de l'outil : c'est une amélioration du temps de calcul global qui est recherchée, et non uniquement des améliorations de fonctions spécifiques. L'amélioration de la performance peut donc passer par une réorganisation des fonctions à implémenter.

Le simulateur stochastique du projet OpenAltaRica est une chaîne d'outils, représentée figure III.1 : le modèle AltaRica 3.0 est analysé pour produire des classes C++ dont la compilation produit un simulateur stochastique spécifique au modèle. Cette technique, utilisée pour améliorer les performances de la simulation, a été étudiée par [35] et a été présentée pour cet outil dans [11].

Le simulateur stochastique spécifique au modèle AltaRica 3.0 est obtenu en plusieurs étapes :

1. Le modèle AltaRica 3.0 est transformé en un modèle GTS par le compilateur AltaRica 3.0, qui est commun à plusieurs outils du projet OpenAltaRica ;
2. Le compilateur du simulateur stochastique analyse le modèle GTS et les indicateurs configurés pour générer des classes C++. Cette étape permet de réaliser des calculs préliminaires qui ne seront donc plus à faire lors de l'étape de simulation. Elle permet aussi de simplifier le modèle simulé. Par exemple, les observateurs non utilisés dans des indicateurs ne seront pas calculés : le modélisateur peut donc fournir un modèle avec

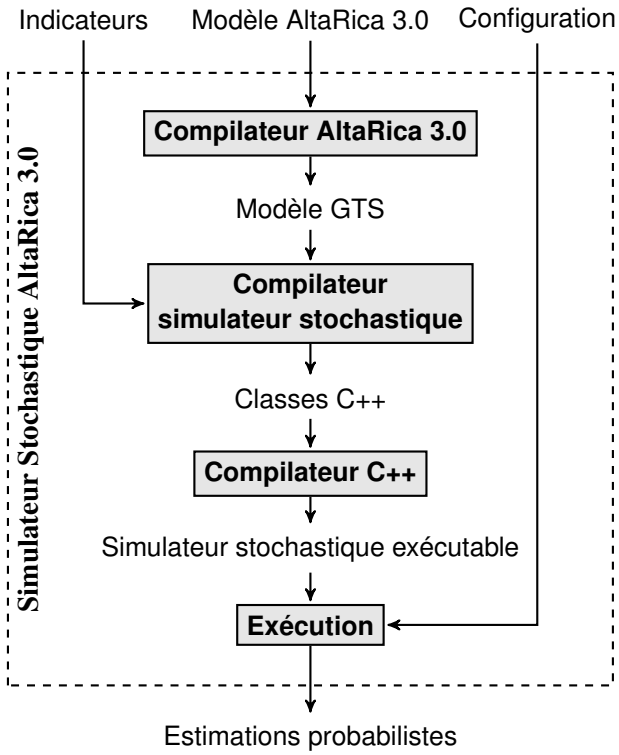

FIGURE III.1 – Chaîne d'outils du simulateur stochastique AltaRica 3.0

un grand nombre d'observateurs pour les différentes études, sans que cela ne pénalise la simulation stochastique.

3. Un compilateur C++ (gcc par exemple) est utilisé pour obtenir, à partir des classes C++, le simulateur stochastique exécutable spécifique au modèle ;

4. L'exécution produit les estimations probabilistes configurées. La simulation stochastique est paramétrée à cette étape : nombre d'histoires à générer et temps de mission. Le même exécutable peut donc être utilisé pour obtenir rapidement des résultats avec un intervalle de confiance important, et des résultats avec un intervalle de confiance faible au prix d'un plus grand temps de calcul.

3.2 Amélioration itérative

Une fois l'architecture globale d'un logiciel définie, l'optimisation de ses performances peut être réalisée de façon itérative : le logiciel est exécuté avec des modèles couvrant plusieurs cas d'utilisation (voir chapitre V), ces exécutions permettant à un outil de profilage de réaliser des mesures sur les fonctions (au sens C++) exécutées : durée d'exécution et nombre d'appels de chaque fonction. Ces mesures permettent d'identifier les "goulets d'étranglement" du logiciel, c'est-à-dire les fonctions consommant le plus de puissance de calcul. Ces fonctions sont donc celles dont l'optimisation est à privilégier. Une fois l'optimisation de la partie lente réalisée, le processus peut être réitéré pour identifier et optimiser une nouvelle partie de l'outil.

Plusieurs itérations d'amélioration des performances ont été réalisées sur le simulateur stochastique AltaRica 3.0 : elles ont concerné la majorité des parties du logiciel. Les améliorations ont consisté à :

- modifier les structures de données utilisées ;
- changer les algorithmes de calcul pour en baisser la complexité ;
- déporter des calculs de l'étape de génération des histoires et de leur analyse à l'étape de génération des classes C++ : tous les calculs pouvant être réalisés de façon préliminaire ne sont alors plus à faire à chaque histoire générée, ce qui permet une économie de puissance de calcul.

Deux améliorations sont présentées dans ce chapitre : le passage de l'analyse des histoires par les indicateurs (filtrage, mesure, et estimation) a posteriori au traitement en cours de simulation, et l'amélioration du calcul de la mise à jour des gardes.

3.3 Mesure et estimation probabiliste

L'implémentation naïve de la simulation stochastique réalise un post-traitement des résultats : chaque histoire simulée est stockée. Une fois toutes les histoires générées, elles sont filtrées, et les mesures des indicateurs sont réalisées sur les séquences stockées. Les estimations probabilistes sont ensuite calculées.

Cette organisation du calcul implique le stockage d'informations inutiles, mais aussi l'utilisation d'une grande quantité de mémoire. Cette mémoire devant être parcourue et analysée durant le post-traitement, le coût en puissance de calcul est important.

Pour éviter ces coûts, la suite d'opérations peut être modifiée : au lieu de réaliser les simulations, le filtrage, les mesures, puis l'estimation, ces quatre opérations peuvent être réalisées "au fil de l'eau". Le principe est de réaliser les calculs dès que les informations utiles sont disponibles, évitant ainsi une mise en mémoire.

3.3.a Filtrage des histoires

La simulation du modèle est réalisée tir de transition après tir de transition, alors que les mesures portent sur le comportement temporel : il est nécessaire de filtrer les états de durée nulle (voir II.3.1.b).

Les seules informations à filtrer sont celles sur les valeurs des observateurs : il n'est pas nécessaire de filtrer les états complets (par exemple les valeurs des variables, qui ne sont pas utiles aux mesures des indicateurs). Ce filtrage n'a pas besoin d'être réalisé a posteriori, c'est-à-dire après avoir généré toute l'histoire. Il peut être réalisé "au fil de l'eau", c'est-à-dire au fur et à mesure de la génération de l'histoire, pour chaque observateur en ne retenant que ses deux derniers états et la date du dernier changement de valeur. Lors de l'ajout d'un nouvel état, il suffit de comparer la date avec celle du changement de valeur, pour déterminer si le temps a avancé, ou si l'état précédent doit être filtré.

Il est ainsi possible de ne pas garder en mémoire l'intégralité d'une histoire générée avant de la filtrer.

3.3.b Mesures des indicateurs

De la même façon que pour le filtrage des histoires, les mesures des indicateurs peuvent être réalisées "au fil de l'eau".

Des calculs préliminaires, réalisés par le compilateur du simulateur stochastique, permettent de savoir quels évènements (changement de valeur d'un observateur numérique, front montant d'un observateur booléen, occurrence d'un évènement...) sont utiles pour qu'un indicateur réalise une mesure. Cette analyse permet d'inclure des appels aux fonctions de mesure de l'indicateur et d'estimation probabilistes directement dans les fonctions de simulation du modèle, puisque ces fonctions sont générées spécifiquement pour le modèle simulé.

Par exemple, si l'indicateur n'a besoin que des occurrences d'un observateur (c'est-à-dire les dates auxquelles un observateur booléen prend la valeur *true*), alors il peut réaliser des mesures immédiatement après chaque occurrence.

Ainsi, l'histoire n'est pas gardée en mémoire, et n'est pas relue après la génération. De plus, il n'est pas nécessaire de vérifier à chaque pas de simulation, c'est-à-dire après chaque tir de transition, si l'observateur à considérer a évolué puisque c'est la fonction de simulation C++ elle-même qui va appeler les fonctions de mesure de l'indicateur.

Le gain en mémoire et en temps de calcul par rapport à l'implémentation naïve (analyse des histoires a posteriori) est alors important.

3.3.c Estimations probabilistes

Les estimations probabilistes sont réalisées à partir des mesures des indicateurs. Les formules, présentées en II.3.3, demandent pour plusieurs d'entre elles d'avoir accès à l'ensemble des valeurs de la série pour pouvoir être calculées.

Elle peuvent être remplacées par des algorithmes qui calculent les estimations au fur et à mesure de la génération des histoires, ce qui évite de devoir garder en mémoire l'intégralité des différentes mesures.

i Estimation du paramètre de la loi de Bernoulli

Le calcul de la proportion de valeurs *true* dans une série de valeurs booléennes, nécessaire pour estimer le paramètre p d'une loi de Bernoulli (voir II.3.3.a), peut être réalisé en ne gardant en mémoire que le nombre de valeurs *true* rencontrées, et le nombre de valeurs totale, sous la forme de deux compteurs.

L'estimation de la proportion sera réalisée, à la fin de la simulation, en divisant le premier compteur par le second.

La mise en mémoire et le parcours final de la liste des mesures sont ainsi économisés par rapport à l'implémentation naïve.

ii Estimation des paramètres de la loi normale

L'estimation des paramètres de la loi normale (espérance et écart-type) par les formules présentées en II.3.3.b présente deux inconvénients dans le cas d'une série avec un grand nombre de valeurs, ce qui est le cas pour la simulation stochastique de modèles de sûreté de fonctionnement :

- il est nécessaire d'avoir toutes les valeurs de la série en mémoire pour la calculer ;
- on se heurte aux défauts des opérations en virgule flottante (débordements, absorption catastrophique).

Pour éviter ces problèmes, le calcul de la moyenne arithmétique (pour estimer l'espérance) est réalisé mesure par mesure [36], en utilisant deux valeurs en mémoire :

- Une variable *mean* qui mémorise la moyenne actuelle ;

- Une seconde variable *counter* qui mémorise le nombre de mesures ajoutées.

A chaque nouvelle mesure (d'indice i et de valeur x_i), ces deux variables sont mises à jour à l'aide des instructions suivantes :

$$mean = mean + \frac{1}{counter}(x_i - mean)$$

$$counter = counter + 1$$

La moyenne arithmétique est donnée, à la fin de la simulation stochastique, par la valeur de la variable *mean*.

De même, pour l'estimation de l'écart-type, trois variables sont utilisées :

- Une variable *squaredmean* mémorise la moyenne des carrés des mesures ajoutées ;
- La variable *mean* utilisée pour l'estimation de l'espérance (cf II.3.3.b.i) ;
- La variable *counter* utilisée pour l'estimation de l'espérance ;

A chaque nouvelle mesure (d'indice i et de valeur x_i), la variable *squaredmean* est mise à jour en suivant l'instruction suivante :

$$squaredmean = squaredmean + \frac{1}{counter}(x_i^2 - squaredmean)$$

L'écart-type est calculé à la fin de la simulation en utilisant l'équation :

$$\sigma = \sqrt{squaredmean - mean^2}$$

3.3.d Conclusion

Le traitement des histoires pour obtenir les estimations probabilistes est possible car les indicateurs configurés sont connus avant la simulation. L'inconvénient corolaire est qu'en cas de modification des indicateurs, un nouveau simulateur stochastique spécifique doit être généré, compilé et exécuté : il n'est pas possible de réaliser un post-traitement différent sur les histoires générées et enregistrées dans le cas de l'implémentation naïve.

3.4 Gardes affectées

Une amélioration importante apportée au simulateur stochastique AltaRica 3.0 concerne la mise à jour des gardes [1].

3.4.a Relations entre transitions

Les transitions servent à modéliser le passage d'un état du système à un autre. Les gardes sont des fonctions booléennes des variables d'état et de flux du modèle. Si la valeur d'une garde est *true*, alors la transition associée peut être tirée. Si cette valeur est *false*, alors la transition ne peut pas être tirée.

Le paradigme orienté prototype du langage AltaRica 3.0 mène à des modèles qui sont des assemblages, suivant une architecture, de composants. Les comportements de ces composants sont décrits par des GTS, composés de variables d'état, d'évènements et de transitions. Les actions des transitions affectent des variables d'état du composant, et les gardes des transitions dépendent de variables d'états et de flux de ce même composant. Les GTS des différents

1. Cette amélioration a fait l'objet d'une publication [5].

composants sont reliés en utilisant les assertions. Les comportements des composants peuvent aussi être reliés en synchronisant des transitions : ce mécanisme est lourd et est réservé à certains motifs de modélisation.

Le modèle AltaRica 3.0 obtenu garde donc généralement une relative indépendance entre les transitions de composants différents : il n'y a que si des assertions relient une variable d'état d'un composant à une variable de flux d'un autre composant utilisée dans une garde qu'une transition peut avoir un effet sur la garde d'un autre composant.

3.4.b Illustration

```
class Composant
  parameter Real pLambda = 10e-6;
  // Taux de défaillance en défaillances/heure
  parameter Real pMu = 10e-3;
  // Taux de réparation en réparations/heure
  parameter Real pDef = 10e-4;
  // Taux de défaillance au démarrage en défaillances/démarrages

  Boolean veDemarre (init = false);
  // Le composant est-il démarré ?
  Boolean veFonctionne (init = true);
  // Le composant est-il en état de fonctionner ?

  Boolean vfEntree (reset = false);
  Boolean vfSortie (reset = false);
  Boolean vfDemarrer (reset = false);
  // Le composant doit-il démarrer (ordre extérieur au composant)

  event eDemarrer (delay = 1 - pDef);
  event eDefaillanceDemarrage (delay = 0, expectation = pDef);
  event eDefaillance (delay = exponential(pLambda));
  event eReparation (delay = exponential(pMu));
  transition
    eDemarrer: not veDemarre and veFonctionne and vfDemarrer
               -> veDemarre := true;
    eDefaillanceDemarrage: not veDemarre and veFonctionne and vfDemarrer
               -> veFonctionne := false;
    eDefaillance: veDemarre
               -> {veFonctionne := false; veDemarre := false}
    eReparation: not veFonctionne
               -> veFonctionne := true;
  assertion
    if veDemarre then vfSortie := vfEntree;
end
```

FIGURE III.2 – Modèle AltaRica 3.0 d'un composant avec défaillance au démarrage

Un exemple de modèle de sûreté de fonctionnement classique est le modèle série-parallèle avec défaillance au démarrage (sans réparateurs limités), décrit dans l'annexe C.1. Une modélisation AltaRica 3.0 de la configuration avec 2 ensembles en série de 2 composants en

```
class Ensemble
  Composant C1;
  Composant C2;
  // Instanciation des composants

  Boolean vfEntree (reset = false);
  Boolean vfSortie (reset = false);

  assertion
    C1.vfEntree := vfEntree;
    C2.vfEntree := vfEntree;
    // L'entrée de chaque composant est celle de l'ensemble
    vfSortie = C1.vfSortie or C2.vfSortie;
    // La sortie de l'ensemble est vraie si au moins une sortie de
    composant est vraie

    C1.vfDemarrer := not (C1.veDemarre or C2.veDemarre);
    C2.vfDemarrer := not (C1.veDemarre or C2.veDemarre);
    // Si aucun composant n'est démarré, alors il faut en démarrer un
end

block Systeme
  Ensemble E1;
  Ensemble E2;
  // Instanciation des ensembles

  observer Boolean oProduction = E2.vfSortie;
  // Le système fonctionne-t-il ?

  assertion
    E1.vfEntree = true;
    E2.vfEntree = E1.vfSortie;
    // Liaison entre les ensembles
end
```

FIGURE III.3 – Modèle AltaRica 3.0 d'un système série-parallèle

parallèle, avec un composant de rechange dans chaque ensemble, est proposée figures III.2 et III.3.

Ce modèle comporte 16 transitions (4 par composants). En déterminant les liens entre :

- les actions des transitions et les variables d'état ;
- les variables d'état et les variables de flux ;
- et les variables (d'état et de flux) et les gardes ;

on peut obtenir la matrice des relations entre les actions et les gardes des transitions, présentée table III.1 [2]. Cette matrice est très creuse : le tir d'une transition n'est donc susceptible de modifier la valeur que de quelques gardes. Par exemple, le tir d'une transition d'un composant de l'ensemble 1 ne peut en aucun cas avoir un effet sur les gardes des transitions des composants de l'ensemble 2, car les deux ensembles sont indépendants.

2. Exemple d'interprétation : le tir de la transition E1.C1.eDemarrer va modifier des valeurs de variables qui vont peut-être changer la valeur de la garde de E1.C2.eDemarrer. À l'inverse, le tir de E1.C1.eDemarrer ne va modifier la valeur d'aucune des variables utilisées par les transitions de E2 : aucune garde d'une transition de E2 ne va donc changer de valeur après le tir de E1.C1.eDemarrer.

Actions \ Gardes	E1.C1.eDemarrer	E1.C1.eDefaillanceDemarrage	E1.C1.eDefaillance	E1.C1.eReparation	E1.C2.eDemarrer	E1.C2.eDefaillanceDemarrage	E1.C2.eDefaillance	E1.C2.eReparation	E2.C1.eDemarrer	E2.C1.eDefaillanceDemarrage	E2.C1.eDefaillance	E2.C1.eReparation	E2.C2.eDemarrer	E2.C2.eDefaillanceDemarrage	E2.C2.eDefaillance	E2.C2.eReparation
E1.C1.eDemarrer	**1**	**1**	**1**	*0*	**1**	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C1.eDefaillanceDemarrage	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C1.eDefaillance	**1**	**1**	**1**	**1**	**1**	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C1.eReparation	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C2.eDemarrer	**1**	**1**	*0*	*0*	**1**	**1**	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C2.eDefaillanceDemarrage	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C2.eDefaillance	**1**	**1**	*0*	*0*	**1**	**1**	**1**	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E1.C2.eReparation	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
E2.C1.eDemarrer	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	**1**	*0*	**1**	**1**	*0*	*0*
E2.C1.eDefaillanceDemarrage	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*
E2.C1.eDefaillance	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	**1**	**1**	**1**	**1**	*0*	*0*
E2.C1.eReparation	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**	*0*	*0*	*0*	*0*
E2.C2.eDemarrer	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	*0*	**1**	**1**	**1**	*0*
E2.C2.eDefaillanceDemarrage	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**
E2.C2.eDefaillance	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	*0*	**1**	**1**	**1**	**1**
E2.C2.eReparation	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**1**	**1**	*0*	**1**

TABLE III.1 – Matrice de relation des transitions, entre les variables affectées dans les actions et les variables utilisées dans les gardes

3.4.c Amélioration

L'implémentation naïve propose (en III.2.1), après chaque tir de transition, de calculer la nouvelle valeur de la garde de chaque transition pour obtenir la liste de celles dont la valeur à changé, afin d'ajouter ou de supprimer de l'échéancier les évènements correspondants. Cette évaluation est de complexité de l'ordre de $O(n)$ opérations élémentaires, avec n le nombre de transitions dans le modèle : comme elle est exécutée systématiquement après chaque tir de transition de chaque histoire générée, son coût en puissance de calcul est très important, d'autant plus sur les modèles possédant un nombre important de transitions.

Le calcul préliminaire par le compilateur du simulateur stochastique de la matrice de relation entre les actions et les gardes des transitions permet d'indiquer, dans les classes C++, quelles gardes doivent être évaluées après chaque tir de transition. Lors de la génération des histoires par le simulateur stochastique spécifique au modèle, cette étape n'est alors plus que de complexité de l'ordre de $O(1)$ opérations élémentaires : l'économie de puissance de calcul est donc particulièrement importante pour les modèles avec un grand nombre de transitions.

3.4.d Expérimentations et conclusion

Cette modification de l'implémentation, en réalisant des calculs préliminaires pour éviter des évaluations systématiques dans la boucle de la génération des histoires, permet une amélioration de la performance de l'outil de simulation stochastique. Le gain en complexité étant d'un ordre de grandeur par rapport au nombre de transitions, le gain en performance sur cette partie de la simulation peut tendre vers 100%. Toutefois, les mesures montrent des diminutions entre 30% et 80% du temps de calcul suivant le modèle testé (parmi l'ensemble de modèles présenté dans l'annexe C) : cette modification ne porte que sur une étape de l'algorithme de génération d'histoires. La modification d'autres étapes est à réaliser lors d'une autre itération d'amélioration du simulateur stochastique AltaRica 3.0.

4 Conclusion

L'implémentation d'un simulateur stochastique de modèles AltaRica 3.0 peut être réalisée de façon immédiate et naïve, les algorithmes de simulation stochastique et de simulation de modèle AltaRica 3.0 étant connus et définis. Le besoin de performance, afin d'obtenir des résultats avec un faible intervalle de confiance en un temps de calcul réduit, peut être satisfait avec une réflexion sur l'architecture de l'outil logiciel et la mise en place d'une démarche d'amélioration itérative.

Pour répondre à ce besoin, le simulateur stochastique du projet OpenAltaRica est décomposé en une chaîne d'outils, permettant la réalisation de calculs préliminaires et la production d'un simulateur stochastique spécifique au modèle. Cet ensemble d'outils a été amélioré de façon itérative, en concentrant à chaque fois les efforts sur les parties les plus coûteuses en calcul. Les améliorations ont porté sur les algorithmes de génération d'histoires, les structures de données, et sur l'analyse des histoires générées (filtrage, mesure et estimation probabiliste) simultanément à leur génération.

Le résultat est une chaîne d'outils dont les performances sont présentées dans une étude d'un système dans le chapitre IV, et dont le bon fonctionnement peut être étudié en utilisant le chapitre V.

Chapitre IV

Analyse de sûreté de fonctionnement d'un système mécatronique

1 Introduction

Un cas d'étude a été proposé par Boniol et Wiels [14] comme banc d'essai pour des techniques et outils de vérification de propriétés comportementales de systèmes. Ce cas d'étude, un train d'atterrissage d'avion, est composé à la fois d'une partie mécanique et d'une partie logicielle, dont la combinaison forme le système. Pour cette raison, il est représentatif d'un grand ensemble de systèmes mécatroniques [33].

Plusieurs approches de modélisation et de vérification de ce cas d'étude ont été proposées lors d'un atelier spécifique, dont [14] est le résultat. Ces approches se concentrent sur la partie logicielle du système. Ce chapitre propose une étude du point de vue de l'analyste de sûreté de fonctionnement [1].

L'objectif principal d'une analyse de sûreté de fonctionnement probabiliste est d'évaluer le risque que "quelque chose se passe mal". Les normes et standards de sûreté, tels que [3] et [4], préconisent de réaliser une analyse de sûreté de fonctionnement probabiliste pour chaque système critique, dont font partie les systèmes aéronautiques tels qu'un train d'atterrissage. Actuellement, les analyses de sûreté de fonctionnement se concentrent sur les parties matérielles des systèmes, tandis que les parties logicielles sont vérifiées par d'autres méthodes (model-checking, preuves...). Les méthodes de sûreté de fonctionnement telles que les arbres de défaillances (Fault Trees) ou les arbres d'évènements (Event Trees [58]) sont largement utilisées et bien maîtrisées. Elles ne sont toutefois pas suffisamment expressives pour analyser les systèmes mécatroniques, ces systèmes ayant des comportements dépendant du temps et de l'ordre des évènements. Pour cet ensemble de systèmes, il est nécessaire d'embarquer dans le modèle analysé au minimum une abstraction du logiciel de contrôle.

Le langage AltaRica 3.0 est suffisamment expressif pour permettre de décrire les parties mécaniques, mais aussi les parties logicielles d'un tel système. Le modèle AltaRica 3.0 obtenu ne peut alors pas être étudié par tous les outils de l'atelier AltaRica 3.0, tels que le compilateur vers les arbres de défaillances, pour les mêmes raisons que les outils classiques de sûreté de fonctionnement ne permettent pas l'étude d'un système dynamique. La simulation stochastique permet cette étude, et fourni des estimations des grandeurs probabilistes concernées par les exigences de sûreté de fonctionnement.

La contribution principale de ce chapitre est la démonstration des capacités d'AltaRica 3.0 et de la simulation stochastique pour réaliser des analyses de sûreté de fonctionnement pertinentes de systèmes mécatroniques, avec un effort d'ingénierie raisonnable.

Le cas d'étude est présenté synthétiquement avec sa modélisation en AltaRica 3.0 en IV.2. La modélisation de la partie logicielle du système est détaillée en IV.3. La démarche de vérification des exigences de sûreté de fonctionnement est présentée en IV.4, et des résultats expérimentaux sont rapportés en IV.5.

2 Cas d'étude

Cette partie contient une description partielle du cas d'étude du train d'atterrissage ("landing gear"), ainsi que des éléments de modélisation AltaRica 3.0. Une description complète du cas d'étude est présente dans l'article de Boniol et Wiels [14].

1. Cette étude a été présentée dans [7].

2.1 Architecture du système

Le système du train d'atterrissage est composé de trois parties : une partie mécanique et hydraulique, un contrôleur, et une interface pour le pilote (figure IV.1). En raffinant la décomposition de chaque partie, le système contient au total 72 éléments interagissant entre eux.

La structure du modèle AltaRica 3.0 reflète l'architecture physique du système. AltaRica 3.0 propose des primitives de construction issues à la fois du paradigme orienté objet, et du paradigme orienté prototype (voir [9]). Les prototypes sont utilisés pour représenter un composant avec une unique occurrence dans le système, tandis que les classes et leurs instances sont utilisées pour décrire des composants soit avec de multiples occurrences dans un système, soit qui peuvent être réutilisés d'un modèle à un autre par l'utilisation de bibliothèques de composants. Dans ce cas d'étude, comme le montre la figure IV.2, l'interface avec le pilote et le switch analogiques (qui sont uniques dans le système) sont modélisés à l'aide d'un prototype, tandis que les différentes électrovalves sont toutes des instances d'une même classe ElectroValve.

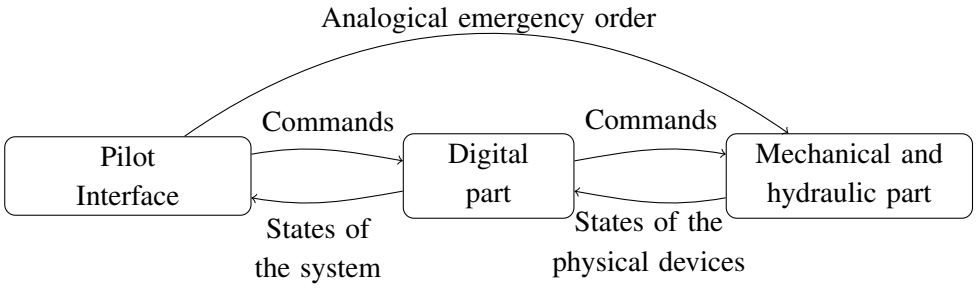

FIGURE IV.1 – Architecture du système de train d'atterrissage

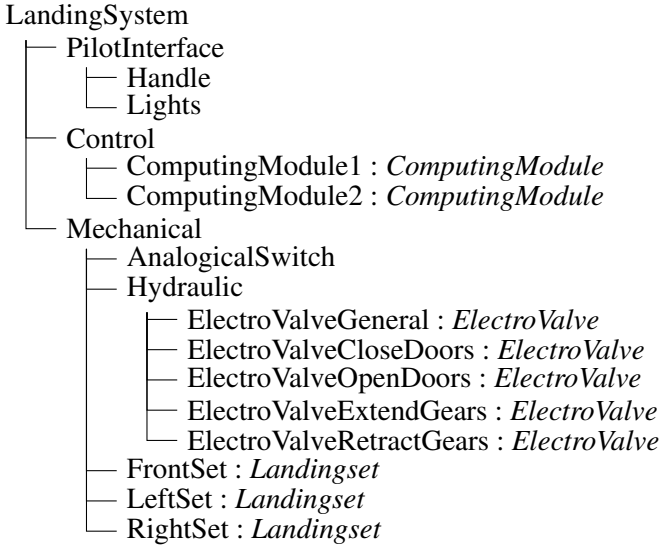

FIGURE IV.2 – Modèle AltaRica 3.0 partiel de la structure : les noms en italique sont des classes, les composants sans classe sont des prototypes

2.2 Comportement physique

Le comportement physique fonctionnel des parties mécaniques du système est décrit par des automates temporisés. Leur modélisation AltaRica 3.0 est donc immédiate.

Afin de réaliser une analyse de sûreté de fonctionnement, des hypothèses ont été faites sur les modes de défaillances des composants, et sur les probabilités de ces défaillances, ces données n'étant pas décrites dans [14].

La première hypothèse est que les défaillances sont immédiates et sans usure. La seconde hypothèse est la répartition des composants en deux catégories :

- les composants avec une défaillance au démarrage, dont le taux de défaillance est exprimé en défaillances par utilisation ;
- les composants avec une défaillance stochastique, dont le taux de défaillance (λ) est exprimé en défaillances par heure. Les défaillances de ces composants sont modélisées par une loi exponentielle (taux de défaillance constant), par cohérence avec la première hypothèse.

Dans les deux cas, les valeurs de taux de défaillance retenues (voir table IV.1) sont réalistes, mais pas réelles.

TABLE IV.1 – Modes et taux de défaillance des composants

Composant	Mode	Taux
Analogical Switch	à l'utilisation	10^{-6}
Electrovalve	à l'utilisation	10^{-4}
Cylinder	taux constant	$10^{-5}h^{-1}$
Sensor	taux constant	$10^{-4}h^{-1}$
Computing Module	taux constant	$10^{-6}h^{-1}$

2.3 Interface avec le pilote

L'interface avec le pilote est composée d'une poignée haut/bas, qui est utilisée pour lancer les séquences d'extension et de rétractation des trains d'atterrissage, et d'un ensemble de voyants lumineux qui indiquent les positions des portes et des roues, ainsi que l'état du système.

Le comportement de la poignée est similaire à une mémoire : la poignée reste dans l'état dans laquelle elle a été mise. Elle peut donc être modélisée en AltaRica 3.0 par une variable d'état booléenne, avec deux transitions modélisant l'abaissement et la levée de la poignée.

Les voyants lumineux ne font qu'afficher une information reçue, sans la garder en mémoire : ils peuvent donc être modélisés par des variables de flux.

2.4 Partie contrôle

La partie contrôle est composée de deux calculateurs identiques, exécutant en parallèle le même logiciel. Ce logiciel est chargé de contrôler les roues et les portes, de détecter les anomalies, et d'informer le pilote via l'interface. Dans [14] le logiciel n'est pas décrit, car sa définition fait partie des propositions d'études à mener. Son comportement est partiellement

spécifié par des séquences d'opérations et des contraintes de temps. Pour mener cette étude, un logiciel répondant à ces spécifications a été défini et est présenté en IV.3.1.

Chaque calculateur reçoit 54 valeurs d'entrée booléennes, et émet 8 valeurs booléennes (3 vers l'interface avec le pilote, et 5 vers la partie mécanique). Pour le modèle AltaRica 3.0, les deux calculateurs font partie du bloc de contrôle (figure IV.3). Ce bloc distribue les entrées vers chacun des calculateurs, et agrège leurs sorties.

2.4.a Entrées

Les entrées de la partie contrôle sont des valeurs booléennes provenant de la poignée de l'interface avec le pilote, et des capteurs de la partie mécanique. Elles sont modélisées en AltaRica 3.0 par des variables de flux, et sont directement reliées aux modèles des calculateurs par des assertions.

Dans le système, les capteurs sont triplés : pour chaque information utile de position, il y a 3 valeurs à prendre en compte. Le langage AltaRica 3.0 permet de gérer ces groupes de valeurs à transmettre à travers le modèle à l'aide de "record" (une structure de variables), facilitant ainsi le travail de modélisation : dans la figure IV.3, l'instanciation d'un de ces groupes de valeurs est montrée ligne 15, tandis que les liaisons vers les deux modules sont en ligne 18 et 19.

2.4.b Sorties

Les sorties de la partie contrôle sont des valeurs électriques vers l'interface avec le pilote (voyants lumineux) et des commandes électriques pour les électrovalves hydrauliques. Ils sont modélisés par des variables de flux booléennes. Les ordres provenant des deux calculateurs peuvent être contradictoires, si ces derniers ont des sorties différentes. Les sorties de la partie contrôle sont la composition des sorties des calculateurs par un OU électrique : si l'un des calculateurs donne l'ordre avec la valeur vrai, alors la partie contrôle transmet cet ordre même si l'autre calculateur ne donne pas l'ordre.

La modélisation de cette composition est illustrée dans la figure IV.3 à la ligne 21.

2.5 Partie mécanique

La partie mécanique contient les trains d'atterrissage (roues et portes) et le circuit hydraulique.

2.5.a Trains d'atterrissage

La partie mécanique est composée de trois trains d'atterrissage identiques : avant (front), gauche (left) et droit (right). Chaque train comporte une porte et une roue amovible. La porte est ouverte et fermée par un vérin (cylinder), et est maintenue en position fermée par un verrou (latching box). La roue est étendue et rétractée par un autre vérin, et deux verrous la maintiennent en position haut ou en position fermée.

La séquence d'extension est composée de plusieurs étapes : déverrouillage et ouverture de la porte, déverrouillage de la roue, extension, verrouillage de la roue, et enfin fermeture et verrouillage de la porte. La séquence de rétractation est similaire. Chaque étape a une durée nominale, qui dépend du train d'atterrissage (avant, gauche ou droite). Comme il s'agit

```
block Control
  ComputingModule Module1;
  ComputingModule Module2;
  /* Orders to PilotInterface */
  Boolean vfOutLockedDown( reset = true );
  Boolean vfOutManeuvering ( reset = false );
  Boolean vfOutPhysicalFailure ( reset = false );
  /* Orders to hydraulic */
  Boolean vfOutEVGeneral (reset = false);
  Boolean vfOutEVCloseDoors (reset = false);
  Boolean vfOutEVOpenDoors (reset = false);
  Boolean vfOutEVRetractGears (reset = false);
  Boolean vfOutEVExtendGears (reset = false);
  /* From Left Gear sensors */
  ThreeValues vfInLeftDoorOpen (reset = false);
  [...]
  assertion
    Module1.vfInLeftDoorOpen := vfInLeftDoorOpen;
    Module2.vfInLeftDoorOpen := vfInLeftDoorOpen;
    [...]
    vfOutLockedDown := Module1.vfOutLockedDown or Module2.vfOutLockedDown;
  [...]
end
```

FIGURE IV.3 – Modèle AltaRica 3.0 partiel de la partie contrôle

de processus physiques, les durées réelles varient de plus ou moins 20% autour de la durée nominale.

Pour modéliser ce comportement physique, les vérins sont décrits dans le modèle AltaRica 3.0 par des automates avec des lois de temporisation définies par l'utilisateur (figure IV.4). Les lois de temporisation définies par l'utilisateur sont des distributions cumulatives de probabilités décrites par un ensemble de points [11] (figure IV.5). Grâce à cette primitive du langage AltaRica 3.0, le délai des différentes actions est déterminé stochastiquement lors des simulations, en respectant l'écart de 20% autour de la valeur nominale.

2.5.b Circuit hydraulique

Les électrovalves mettent sous pression différentes portions du circuit hydraulique. 5 électrovalves sont présentes dans le circuit hydraulique : une générale, et une pour chacun des 4 mouvements des trains d'atterrissage (extension/rétractation des roues, et ouverture/fermeture des portes). Les mouvements des trois trains d'atterrissage dépendent donc des mêmes électrovalves. Elles sont commandées par des ordres électriques provenant de la partie contrôle. Un interrupteur analogique empêche tout ordre électrique de la partie contrôle vers l'électrovalve si il n'y a pas eu d'ordre récent de la part du pilote. Un triplet de capteurs discrets informe la partie contrôle de l'état de chaque vérin.

Les électrovalves étant des composants physiques, leur ouverture et leur fermeture demandent du temps. Chaque mouvement a une probabilité de défaillance, comme défini en IV.2.2. Un automate décrivant le comportement dysfonctionnel de l'électrovalve est présenté figure IV.6. La classe AltaRica 3.0 modélisant cet automate est en figure IV.7. Cette classe est ensuite instanciée dans le modèle AltaRica 3.0 pour chacune des 5 électrovalves.

```
LandingSet Front (
 GearCylinder.eUnlockDown.delay = [ 0.0:0.0, 0.32:0.0, 0.48:1.0],
 GearCylinder.eFinishGoingHigh.delay =
                                  [ 0.0:0.0, 1.28:0.0, 1.92:1.0],
 GearCylinder.eLockHigh.delay =   [ 0.0:0.0, 0.32:0.0, 0.48:1.0],
 GearCylinder.eUnlockHigh.delay = [ 0.0:0.0, 0.64:0.0, 0.96:1.0],
 GearCylinder.eFinishGoingDown.delay =
                                  [ 0.0:0.0, 1.6:0.0, 2.4:1.0],
 GearCylinder.eLockDown.delay =   [ 0.0:0.0, 0.64:0.0, 0.96:1.0],

 DoorCylinder.eFinishGoingHigh.delay =
                                  [ 0.0:0.0, 0.96:0.0, 1.44:1.0],
 DoorCylinder.eLockHigh.delay =   [ 0.0:0.0, 0.64:0.0, 0.96:1.0],
 DoorCylinder.eUnlockHigh.delay = [ 0.0:0.0, 0.96:0.0, 1.44:1.0],
 DoorCylinder.eFinishGoingDown.delay =
                                  [ 0.0:0.0, 0.96:0.0, 1.44:1.0]
);
```

FIGURE IV.4 – Instanciation du train d'atterrissage avant dans le modèle AltaRica 3.0 : définition des lois de temporisation pour chaque mouvement physique des vérins de porte et de roue

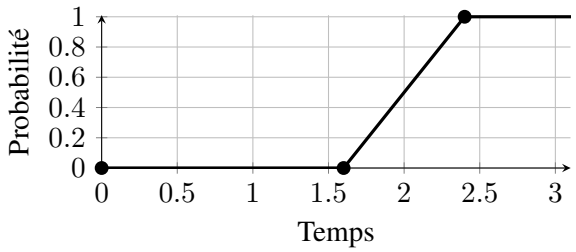

FIGURE IV.5 – Distribution cumulative de probabilités pour la durée du mouvement de rétractation de la roue avant

Au total, la partie mécanique comporte 3 portes, 9 verrous, 6 vérins, 5 électrovalves, 18 triplets de capteurs, et un interrupteur analogique.

3 Contrôleur

Le système présenté dans [14] spécifie des contraintes que le logiciel exécuté par les deux calculateurs doit satisfaire, la définition de celui-ci faisant partie des propositions d'études à mener.

Les exigences à vérifier portent en partie sur le comportement du système. Celui-ci étant mécatronique, son comportement est le résultat de ses parties mécaniques et de son logiciel de contrôle. Pour réaliser une analyse de sûreté de fonctionnement prenant en compte son comportement, il est alors nécessaire d'avoir un modèle de son logiciel de contrôle qui accompagne le modèle de ses parties mécaniques.

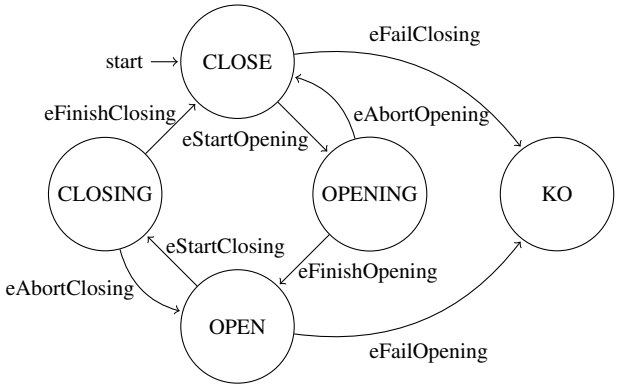

FIGURE IV.6 – Automate du comportement fonctionnel et dysfonctionnel d'une électrovalve

```
1  domain ElectroValveState { CLOSE, OPEN, CLOSING, OPENING }
2  class ElectroValve
3    Boolean vsKO ( init = false );
4    ElectroValveState vsState (init = CLOSE);
5    //Input, Output and Command
6    Boolean vfIn, vfOut, vfE (reset = false);
7    //Time to Open and close
8    parameter Real pOpening = 1.0;
9    parameter Real pClosing = 3.6;
10   event eAbortOpening (delay = Dirac(0));
11   event eAbortClosing (delay = Dirac(0));
12   event eStartClosing (delay = Dirac(0), expectation = 0.9999 );
13   event eStartOpening (delay = Dirac(0), expectation = 0.9999 );
14   event eFailClosing (delay = Dirac(0), expectation = 1.0e-4 );
15   event eFailOpening (delay = Dirac(0), expectation = 1.0e-4 );
16   event eFinishClosing (delay = Dirac(pClosing));
17   event eFinishOpening(delay = Dirac(pOpening));
18   transition
19     // Functionnal transitions
20     eStartClosing: not vsKO and vsState == OPEN and vfE == false -> vsState
       := CLOSING;
21     eFinishClosing: not vsKO and vsState == CLOSING -> vsState := CLOSE;
22     eStartOpening: not vsKO and vsState == CLOSE and vfE -> vsState :=
       OPENING;
23     eFinishOpening: not vsKO and vsState == OPENING -> vsState := OPEN;
24     eAbortOpening: not vsKO and vsState == OPENING and vfE == false ->
       vsState := CLOSE;
25     eAbortClosing: not vsKO and vsState == CLOSING and vfE -> vsState :=
       OPEN;
26     // Dysfunctionnal transitions
27     eFailClosing: not vsKO and vsState == OPEN and vfE == false -> vsKO :=
       true;
28     eFailOpening: not vsKO and vsState == CLOSE and vfE -> vsKO := true;
29   assertion
30     if vsState != CLOSE then vfOut :=: vfIn;
31 end
```

FIGURE IV.7 – Classe AltaRica 3.0 modélisant les électrovalves

Pour réaliser cette étude, un modèle d'un logiciel de contrôle a été réalisé. Ce modèle est une abstraction du logiciel qui serait réellement embarqué dans les deux calculateurs, mais il respecte toutes les contraintes imposées. Il est divisé en deux parties, une par fonction à assurer : le contrôle, et la surveillance.

3.1 Contrôle

Le logiciel contrôle les actionneurs (électrovalves) d'après les valeurs obtenues par les capteurs et les commandes reçues du pilote au travers de l'interface. La définition retenue est décrite par le GTS représenté graphiquement [2] figure IV.8.

3.2 Surveillance

La fonction de surveillance consiste à informer le pilote de l'état du système, via les trois voyants lumineux de l'interface avec le pilote.

3.2.a Anomalies

Le voyant rouge doit être allumé en cas d'anomalie détectée, qu'elle provienne des capteurs ou des parties mécaniques.

i Anomalie des capteurs

Grâce au triplement des capteurs, les anomalies de ceux-ci sont facilement détectables : si à un instant l'un des capteurs envoie une valeur différente des deux autres, il est ignoré définitivement. Ensuite, si les deux derniers capteurs envoient des valeurs différentes, alors il y a une anomalie.

ii Anomalies des parties mécaniques

Des limites temporelles permettent de détecter des anomalies de la partie mécanique. Par exemple, le circuit hydraulique doit être sous pression 2 secondes après l'ordre d'ouverture de l'électrovalve générale, et ne doit plus être sous pression 10 secondes après l'ordre de fermeture de cette électrovalve. Si ces limites ne sont pas respectées, alors il y a une anomalie qui doit être signalée par le calculateur au pilote. Un exemple de détection d'une telle anomalie est proposé figure IV.9.

iii Surveillance mutuelle

Une fonction de surveillance classique dans le domaine aéronautique, où les sous-systèmes redondants sont nombreux, est celle de surveillance mutuelle. Cette fonction, non demandée dans [14], a été ajoutée suite à l'hypothèse de possible défaillance des calculateurs : si un calculateur ne répond plus, l'autre signale une anomalie au pilote en utilisant le voyant rouge.

2. Pour une meilleure lisibilité, les transitions correspondant à un contre-ordre (le pilote demandant une extension lors d'une séquence de rétractation ou inversement) ne sont pas entièrement décrites, et sont représentées en pointillés.

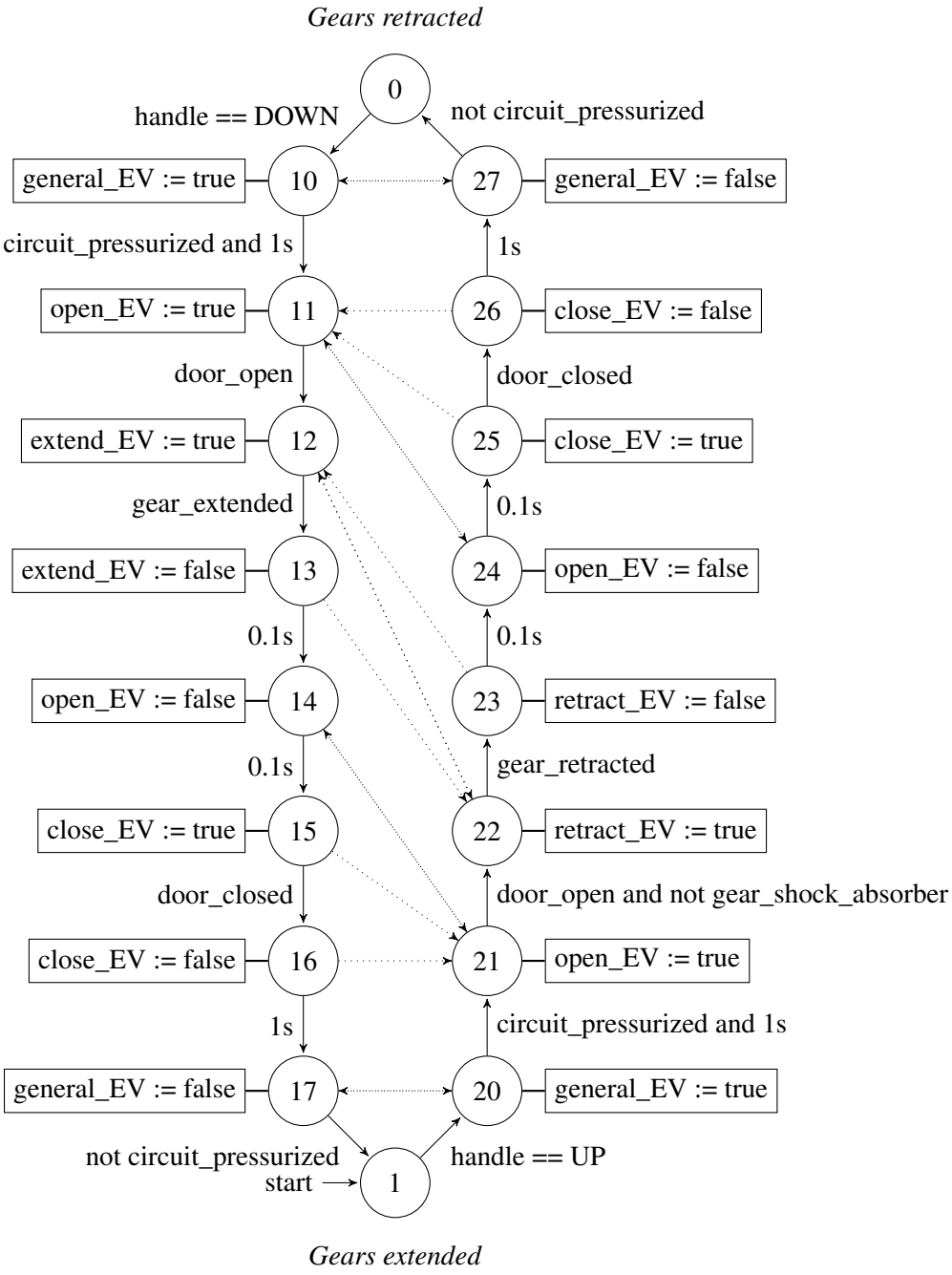

FIGURE IV.8 – Représentation graphique du GTS de la fonction contrôle du modèle du logiciel

```
1  Boolean vsHydraulicMustBePressurized (init = false);
2  event eHydraulicMustBePressurized (delay = Dirac(0));
3  event eHydraulicMustNotBePressurized (delay = Dirac(0));
4  event eHydraulicTooLongToBePressurized (delay = Dirac(2));
5  event eHydraulicTooLongToNotBePressurized (delay = Dirac(10));
6  transition
7    eHydraulicMustBePressurized: vfOutEVGeneral and not
     vsHydraulicMustBePressurized
8                                -> vsHydraulicMustBePressurized := true;
9    eHydraulicMustNotBePressurized: not vfOutEVGeneral and
     vsHydraulicMustBePressurized
10                               -> vsHydraulicMustBePressurized := false;
11   eHydraulicTooLongToBePressurized: vsHydraulicMustBePressurized and not
     vfInPressure.vfOut
12                                   and not vsAnomaly
13                               -> vsAnomaly := true;
14   eHydraulicTooLongToNotBePressurized: not vsHydraulicMustBePressurized
15                                      and vfInPressure.vfOut and not
     vsAnomaly
16                               -> vsAnomaly := true;
```

FIGURE IV.9 – Surveillance des délais de pressurisation hydraulique en AltaRica 3.0

```
1  assertion
2    // Green light : gears are locked down
3    vfOutLockedDown := vsState == 0;
4    // Orange light : system is in motion
5    vfOutManeuvering := vsState != 0 and vsState != 1;
```

FIGURE IV.10 – Modèle AltaRica 3.0 des commandes des voyants vert et orange

3.2.b État du système

La commande du voyant vert (les roues sont verrouillées en position basse) et celle du voyant orange (le système est en mouvement) peuvent être déduites directement de la partie contrôle (figure IV.8) : le premier correspond uniquement à l'état 0 de l'automate, tandis que le second correspond à tous les états des séquences d'extension et de rétractation (donc tous les états sauf le 0 et le 1) . Elles sont donc facilement décrites dans le modèle AltaRica 3.0 en utilisant des variables de flux et des assertions par rapport à l'état de l'automate modélisant le logiciel de contrôle (figure IV.10).

4 Exigences

L'article de Boniol et Wiels [14] fournit un ensemble d'exigences que le système doit vérifier. Elles peuvent être réparties en deux catégories :

- les exigences d'accessibilité qui ne dépendent que de l'état courant du système. Par exemple :

(R_{31}) *When the command line is working (normal mode), the stimulation of the gears outgoing or the retraction electro-valves can only happen when the three doors are locked down.*

- les exigences temporelles, dans lesquelles une action doit être terminée en un temps limité. Par exemple :
  (R_{11}) *When the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then the gears will be locked down and the doors will be seen closed less than 15 seconds after the handle has been pushed.*

4.1 Exigences d'accessibilité

Les exigences d'accessibilité, comme (R_{31}), demandent qu'un état particulier (état redouté) ne soit jamais rencontré par le système.

Pour exprimer la propriété correspondante, on définit :

- un observateur booléen, dont l'expression sur les variables du modèle prend la valeur *true* si et seulement si l'état du modèle correspond à l'état redouté (par exemple, figure IV.11) ;
- un indicateur "Had Value", qui permet de savoir si cet observateur a pris la valeur *true* au cours de l'histoire générée, et donc si l'état redouté a été rencontré au cours de l'histoire.

```
1 observer Boolean R31 =
2      Digital.vfNormalMode
3      and (Hydraulic.EVRetractGears.vfE
4           or Hydraulic.EVExtendGears.vfE)
5      and not (Physical.Front.vfOutDoorOpen
6               and Physical.Left.vfOutDoorOpen
7               and Physical.Right.vfOutDoorOpen);
```

FIGURE IV.11 – Modélisation AltaRica 3.0 d'une exigence d'accessibilité

4.2 Exigences temporelles

Les exigences temporelles, en demandant à ce qu'une action soit effectuée en un temps limité, impliquent de mesurer le temps écoulé entre deux évènements différents : le début et la fin d'une action. Il est donc nécessaire de garder en mémoire une information (le premier évènement a eu lieu) pour agir en conséquence lorsque le second évènement est rencontré. Les évènements concernés par ces exigences correspondent dans le modèle AltaRica 3.0 à des fronts montant ou descendant d'expressions booléennes, mais pourraient correspondre avec une autre modélisation à des tirs de transition.

Pour exprimer la propriété correspondante, on définit :

- un chronomètre sous forme de GTS (figure IV.12) : le premier évènement déclenche le chronomètre, le second l'arrête et le remet à zéro, et si le temps limite configuré est dépassé une variable d'état vient garder en mémoire cette information de dépassement ;
- des assertions pour configurer les évènements de déclenchement et d'arrêt (figure IV.13) ;
- un observateur booléen, dont l'expression prend la valeur *true* en cas de dépassement par le chronomètre du temps limite (variable vsTimeOut) ;

- un indicateur "Has Value", qui permet de savoir si cet observateur a la valeur *true* à la fin de l'histoire générée, et donc si l'action a duré plus longtemps que le temps limite lors de l'histoire.

4.3 Exigence de sûreté

Pour cette étude, une exigence correspondant à un évènement catastrophique (le voyant vert est allumé sans qu'une anomalie ou un mouvement ne soit signalé par les voyants rouges et orange, mais toutes les roues ne sont pas verrouillées sorties, et toutes les portes ne sont pas verrouillées fermées) a été ajoutée : (R_a) *When the pilot interface indicates that the aircraft is ready to land (green light on) without any problem (red light off) or movement (orange light off), then every gear is locked down and every door is locked.*

L'état redouté correspondant est "le voyant vert est allumé *et* les voyants rouge et orange sont éteints, *et* (une roue n'est pas verrouillée baissée *ou* une porte n'est pas verrouillée)". Il s'agit donc d'une exigence d'accessibilité : la propriété a été exprimée comme décrit en IV.4.1.

```
1  class RequirementTimeOut
2   Boolean vfStart (reset = false);
3   Boolean vfStop (reset = false);
4   // Flow variables for configuration of start/stop
5   parameter Real pTimeOut = 1;
6   // Configuration of the time out, in seconds
7   Boolean vsRunning (init = false);
8   Boolean vsTimeOut (init = false);
9   event eStart (delay = 0);
10  event eStop (delay = 0);
11  event eTimeOut (delay = pTimeOut);
12  observer event oeTimeOut = eTimeOut;
13  transition
14    eStart: vfStart and not vfStop and not vsRunning -> vsRunning := true;
15    eStop: vfStop and vsRunning -> vsRunning := false;
16    eTimeOut: vsRunning and not vsTimeOut -> vsTimeOut := true;
17 end
```

FIGURE IV.12 – Modèle AltaRica 3.0 d'un chronomètre pour les exigences temporelles

5 Résultats expérimentaux

Une fois le modèle construit et les indicateurs configurés, il est nécessaire de paramétrer la simulation stochastique (partie IV.5.1). Une fois les calculs terminés, les résultats ont été étudiés (partie IV.5.4).

5.1 Mise en place de la simulation stochastique

Pour permettre le calcul de la simulation stochastique, il est nécessaire d'ajouter des informations concernant le contexte dans lequel le système modélisé doit être testé.

```
1  RequirementTimeOut R_11(pTimeOut = 15);
2  assertion
3   R_11.vfStart := PilotInterface.vsHandle == DOWN;
4   R_11.vfStop := not Digital.vfNormalMode
5                  or PilotInterface.vsHandle != DOWN
6                  or (Physical.Front.vfOutGearLockDown
7                      and Physical.Front.vfOutDoorLockedHigh
8                      and Physical.Left.vfOutGearLockDown
9                      and Physical.Left.vfOutDoorLockedHigh
10                     and Physical.Right.vfOutGearLockDown
11                     and Physical.Right.vfOutDoorLockedHigh );
```

FIGURE IV.13 – Instanciation et configuration d'un chronomètre pour l'exigence (R_{11}) dans le modèle AltaRica 3.0

5.1.a Pilote virtuel et cycles de vols

La majorité des exigences à évaluer portent sur des comportements lors des phases de décollage et d'atterrissage. Il est donc nécessaire de "faire décoller et atterrir" le modèle pour pouvoir étudier ces phases par simulation stochastique. Une représentation du pilote a donc été ajoutée.

Le pilote virtuel commande la poignée de l'interface pour réaliser des cycles de décollage et d'atterrissage.

Par hypothèse, l'avion subit une maintenance toutes les 10 heures de fonctionnement, un vol dure une heure, et il y a une heure au sol entre chaque vol. Ces valeurs ont été choisies pour être représentatives d'un avion effectuant des navettes entre deux villes distantes de quelques centaines de kilomètres, plusieurs fois par jours, avec une maintenance la nuit.

Un cycle (un décollage et un atterrissage) dure donc deux heures, et cinq cycles sont prévus par histoire, soit un temps de mission de 10 heures. En cas d'anomalie détectée, les cycles sont arrêtés pour cette histoire : il n'y a pas de réparation.

5.1.b Paramétrage de la simulation stochastique

La simulation stochastique est configurée pour produire 2×10^8 histoires de 10 heures, soit environ 2×10^9 décollages et atterrissages.

5.2 Dimensions du modèle

Le modèle AltaRica 3.0 complet contient 129 composants (instances de classes et prototypes). Le GTS obtenu par compilation du modèle AltaRica 3.0 est formé de 997 variables (d'état et de flux) et de 504 transitions. Le modèle mis à plat comporte environ 14 000 lignes.

L'espace d'état de l'automate (sans prise en compte de la complexité temporelle) a une taille d'environ 2.6×10^{105} états possibles. Bien que cet espace d'état présente de nombreuses symétries, son exploration exhaustive n'est pas envisageable.

L'utilisation de la simulation stochastique pour en estimer des grandeurs probabilistes est donc justifiée.

5.3 Calcul et performances

Le simulateur stochastique spécifique au modèle a été exécuté parallèlement sur 4 processeurs, pour diviser le temps d'exécution : les résultats ont ensuite été fusionnés. Ces exécutions ont demandé environ 17 heures sur un Intel Xeon E312xx (Sandy Bridge) à 4 coeurs à 2.6 GHz avec 8 GB RAM, soit un temps de calcul non parallélisé de 68 heures.

5.4 Résultats

Chaque indicateur a permis d'estimer l'espérance pour l'observateur correspondant. Ces estimations ont permis de calculer les valeurs des grandeurs probabilistes correspondant aux exigences. Toutes les exigences définies dans [14] ont ainsi pu être analysées : les résultats sont présentés dans la table IV.2.

TABLE IV.2 – Résultats de l'analyse des propriétés par simulation stochastique

Req.	Type	Valeur	Req.	Type	Valeur
(R_a)	Accessibilité	100%	(R_{61})	Temporelle	100%
(R_{11})	Temporelle	99.9999945%	(R_{62})	Temporelle	100%
(R_{12})	Temporelle	99.9999940%	(R_{63})	Temporelle	100%
(R_{21})	Accessibilité	100%	(R_{64})	Temporelle	100%
(R_{22})	Accessibilité	100%	(R_{71})	Temporelle	100%
(R_{31})	Accessibilité	100%	(R_{72})	Temporelle	100%
(R_{32})	Accessibilité	**0%**	(R_{73})	Temporelle	100%
(R_{41})	Accessibilité	100%	(R_{74})	Temporelle	100%
(R_{42})	Accessibilité	100%	(R_{81})	Temporelle	100%
(R_{51})	Accessibilité	100%	(R_{82})	Temporelle	100%

5.4.a Exigences non satisfaites

La simulation stochastique a généré des histoires dans lesquelles des états redoutés ont été rencontrés, ou des temps limités ont été dépassés.

i Exigence (R_{32})

L'exigence (R_{32}) est : *When the command line is working (normal mode), the stimulation of the doors opening or closure electro-valves can only happen when the three gears are locked down or up.*

Cette exigence ne peut pas être respectée, car pour étendre ou rétracter les roues, les portes doivent être ouvertes. Comme il n'y a pas de verrouillage des portes en position ouverte, l'électrovalve d'ouverture des portes doit rester commandée pour pouvoir bouger les roues.

Une nouvelle analyse a été réalisée avec une version partielle de cette exigence, ne portant que sur la commande de l'électrovalve de fermeture des portes qui ne peut survenir que lorsque les trois roues sont verrouillées. Cette version partielle de l'exigence a été satisfaite dans toutes les histoires générées.

ii Exigences (R_{11}) et (R_{12})

Ces deux exigences sont :

- (R_{11}) *When the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then the gears will be locked down and the doors will be seen closed less than 15 seconds after the handle has been pushed.*
- (R_{12}) est identique, mais pour la rétractation des roues.

Les propriétés correspondant à ces deux exigences ont presque toujours été satisfaites dans les histoires générées lors de la simulation stochastique : seulement 11 et 12 histoires concernent des comportements ne les respectant pas. La proportion d'histoires concernées est de 5.5×10^{-9}, avec comme intervalle de confiance à 95% $[5.49993 \times 10^{-9}; 5.50007 \times 10^{-9}]$ (soit une largeur d'intervalle de 7.4×10^{-14}).

Ordre de grandeur de la probabilité

Dans le domaine aéronautique 10^{-9} est la limite haute de probabilité de défaillance pour les systèmes critiques, c'est-à-dire ceux pour lesquels les conséquences sont catastrophiques et dont l'occurrence doit être très rare.

Une estimation d'un indicateur de sûreté de fonctionnement de cet ordre de grandeur peut être réalisée par simulation stochastique, avec un intervalle de confiance à 95% très restreint et un coût de calcul très raisonnable.

5.4.b Exigences satisfaites

Toutes les autres exigences (marquées "100%" dans la table de résultats) ont été respectées dans strictement toutes les histoires générées : aucun des 2×10^9 décollages et atterrissages simulés n'a enfreint ces exigences. Comme il ne s'agit que d'une exploration partielle de l'espace d'état, ce résultat ne constitue pas une preuve du respect de ces propriétés par le modèle dans tous les cas de figure, et encore moins une preuve du respect strict des exigences par le système.

Toutefois, la probabilité pour chaque propriété qu'il existe au moins une histoire possible (qui n'a donc pas été générée ici) du modèle qui ne la respecte pas est incluse dans l'intervalle de confiance à 95% : $[0; 1.5 \times 10^{-9}]$ (d'après [26]) [3].

5.4.c Disponibilité

Les résultats de la simulation stochastique indiquent que 0.78% des histoires générées ont rencontré une défaillance problématique (défaillance d'un composant physique, ou défaillance de deux capteurs d'un même groupe). Bien que cette grandeur probabiliste ait une valeur inacceptable d'après les normes aéronautiques, chacune de ces défaillances a été signalée au pilote comme anomalie, car (R_a) n'a jamais été enfreinte.

La fonction de surveillance du modèle de logiciel de contrôle semble donc remplir son rôle correctement : s'agissant d'une partie logicielle, des outils de vérification destinés aux

3. La probabilité que le système respecte strictement l'exigence est quant à elle nulle : il s'agit d'un système physique. Son modèle ne prend pas tout en compte, et le risque nul n'existe pas, même si sa probabilité peut être très faible.

programmes informatiques peuvent être utilisés pour s'assurer de son bon fonctionnement (model-checking, vérification...).

6 Conclusion

Cette analyse du cas d'étude proposé par [14] montre l'utilisation possible d'AltaRica 3.0 pour la modélisation d'un système mécatronique avec son comportement logiciel (y compris la détection des défaillances matérielles par le logiciel) et de la simulation stochastique pour la vérification de propriétés probabilistes d'un tel modèle.

La simulation stochastique du modèle obtenu a permis d'évaluer le respect d'un ensemble d'exigences par le système modélisé. Bien que cette technique ne puisse pas à elle seule prouver formellement des propriétés, elle permet d'obtenir des résultats à un coût réduit : à l'inverse du model-checking conventionnel, elle est indépendante de la taille de l'espace d'état, et permet donc l'analyse de systèmes complexes.

Chapitre V

Évaluation d'un simulateur stochastique

1 Introduction

La qualité d'une analyse de sûreté de fonctionnement est dépendante de la qualité de l'outil utilisé. C'est pourquoi, avant de conduire une telle analyse, il est important de pouvoir évaluer cette qualité.

Une analyse de sûreté de fonctionnement prend place dans un contexte industriel de conception, avec ses contraintes économiques (temps et capacités de calcul disponibles). Le choix d'un outil d'analyse (et du langage de modélisation associé) ne peut donc pas être basé uniquement sur son bon fonctionnement, mais doit aussi prendre en compte ses capacités (permet-il de mesurer ce dont j'ai besoin ?) et ses performances (peut-il être utilisé en un temps raisonnable ?). Les performances d'un outil doivent donc aussi être évaluées avant son utilisation.

L'évaluation d'un outil de simulation stochastique présente plusieurs difficultés. Le premier est inhérent à son fonctionnement : en utilisant l'aléatoire et en ne produisant que des estimations probabilistes comme résultats, il est difficile de différencier un résultat faux d'un résultat peu probable. Le second est lié à son utilisation : ses performances dépendent du modèle simulé.

L'objectif de ce chapitre est de présenter une méthode pour évaluer les outils de simulation stochastique pour la sûreté de fonctionnement, et son application au simulateur stochastique du projet OpenAltaRica. Cette méthode permet d'évaluer des exigences fonctionnelles, telles que la performance et l'exactitude. Elle prend en compte les fonctions classiques des simulateurs stochastiques évènementiels (échéancier, estimations probabilistes...) et les propriétés de la simulation stochastique (résultats estimés et non exacts, coût du calcul ...). Elle peut être appliquée sur n'importe quel simulateur stochastique évènementiel. Cette application mène à la création d'un kit d'évaluation spécifique à un simulateur stochastique.

Les critères d'évaluation pertinents pour un outil de simulation stochastique pour une étude de sûreté de fonctionnement sont listés en V.2. La définition d'un simulateur stochastique évènementiel et les conséquences de cette définition sur l'évaluation de ces outils sont présentées en V.3. La méthode de construction d'un kit d'évaluation est proposée en V.4, en l'appliquant au simulateur stochastique de modèles AltaRica 3.0. Cette application a mené à la construction du kit présent dans l'annexe C, et des résultats expérimentaux de son utilisation sont présentés en V.5.

2 Critères d'évaluation d'un simulateur stochastique

Un outil destiné à une étude de sûreté de fonctionnement doit répondre à plusieurs besoins : il doit être utile, donner des résultats corrects, et avoir un coût d'utilisation limité. Une méthode d'évaluation d'un tel outil doit donc viser à quantifier ou qualifier ces caractéristiques.

2.1 Utilité

L'objectif d'un simulateur stochastique pour la sûreté de fonctionnement est d'obtenir des résultats probabilistes d'un modèle pour prédire les performances RAMS d'un système. Il est donc important que l'outil utilisé soit capable de calculer les résultats dont l'analyste de sûreté de fonctionnement à besoin.

En fonction du contexte ou de l'industrie dans lesquels l'étude prend place, les résultats attendus peuvent être des mesures de sûreté de fonctionnement classiques (MTBF, MTTR, disponibilité...), ou des mesures plus spécifiques. De même, les systèmes étudiés peuvent présenter des particularités (systèmes bouclés, fortes synchronisations...). L'évaluation d'un outil de sûreté de fonctionnement doit donc permettre de s'assurer que les capacités de l'outil sont suffisantes et adaptées pour les études envisagées.

2.2 Qualité des résultats

Les analyses de sûreté de fonctionnement sont réalisées pour le dimensionnement de systèmes critiques au regard d'exigences de sûreté et de fiabilité. Une sous-évaluation peut conduire à une trop grande probabilité d'évènement problématique, ce qui peut avoir des conséquences catastrophiques. À l'inverse, une surévaluation peut conduire à un surdimensionnement d'un système, et donc à un coût plus important que nécessaire.
De tels écarts peuvent provenir :

- d'une mauvaise modélisation du système ;
- d'une mauvaise configuration de l'analyse ;
- d'erreurs d'interprétation du modèle ;
- d'erreurs de simulation du modèle ;
- d'erreurs dans les calculs probabilistes.

La mauvaise modélisation du système et la mauvaise configuration de l'analyse sont de la responsabilité de l'analyste : des outils peuvent l'aider à détecter certaines erreurs, et des méthodologies peuvent réduire le risque d'erreurs.

En revanche, les trois autres erreurs proviennent de l'outil : un outil de sûreté de fonctionnement qui fonctionne correctement ne devrait pas, si le modèle et la configuration sont corrects, donner de résultat erroné. La qualité des résultats obtenus par un outil doit donc être évaluée : il est nécessaire d'avoir confiance dans le bon fonctionnement de l'outil.

2.3 Coût d'utilisation de l'outil

Le contexte industriel présente un enjeu économique : la réalisation d'une étude à un coût que l'on cherche à minimiser. Le coût de l'utilisation d'un outil logiciel a plusieurs composantes, dont :

- La licence du logiciel ;
- L'installation du logiciel par le service informatique de l'entreprise ;

- Le temps d'apprentissage de l'utilisation du logiciel par l'analyste ;
- Le temps de travail de l'analyste pour une étude ;
- Le matériel nécessaire à l'utilisation du logiciel ;

Les coûts fixes (licence, installation, formation) de l'utilisation d'un outil logiciel sont facilement évaluables, car ils sont approximativement identiques quelque soit le contexte industriel, et du même ordre de grandeur que pour des outils similaires. Le temps de travail de l'analyste, c'est-à-dire le degré d'automatisation des tâches de la démarche utilisant le logiciel, est un coût dont l'évaluation passe par l'expérimentation.

Le matériel nécessaire à l'utilisation du logiciel est quant à lui un coût qui dépend de l'utilisation prévue de l'outil. Dans le cas de la simulation stochastique, l'estimation de ce coût prend une importance particulière : cette méthode d'analyse consiste à calculer des statistiques d'histoires générées aléatoirement du modèle. Pour que les résultats statistiques soient fiables, le nombre d'histoires générées doit être important : plus on génère d'histoires, meilleurs sont les résultats. Mais la génération d'histoires demande du temps de calcul. Pour obtenir des résultats de qualité, il est nécessaire d'investir un temps de calcul correspondant.

Il est donc nécessaire, pour évaluer un outil de simulation stochastique, d'évaluer la quantité de temps de calcul, et donc le coût en matériel, à investir pour obtenir des résultats de la qualité souhaitée.

3 Machine de simulation stochastique

Les différents formalismes état-transition dédiés aux études de sûreté de fonctionnement permettent de décrire des automates probabilistes : chaînes de Markov (I.2.2.a), réseaux de Petri stochastiques (I.2.2.b), AltaRica 3.0 (I.3.3)... Ces automates probabilistes ont tous la possibilité d'être étudiés par simulation stochastique. Il est possible de définir une machine, abstraite, de simulation stochastique, c'est-à-dire de représenter ce qu'est un simulateur stochastique de ces automates probabilistes. Cette définition va permettre de tirer des conclusions, indépendamment des particularités de chaque formalisme, sur la façon d'évaluer ces outils.

3.1 Définition d'une machine de simulation stochastique

Une machine de simulation stochastique est un n-uplet $< \Sigma, E, T, C, \sigma_0, \sigma, S, d, I, p >$ où :

- $(\Sigma, E, T, C, \sigma_0)$ représente le modèle ;
- (σ, S, d, I) représente l'état actuel de la simulation ;
- p représente la configuration de la simulation.

3.1.a Modèle

Le modèle est représenté par le quintuplet $(\Sigma, E, T, C, \sigma_0)$ où :

- Σ est un ensemble des variables. Les valeurs des variables de Σ est noté σ.
- E est un ensemble d'évènements e, associés à des distributions temporelles stochastiques pour les délais de tir.
- T est un ensemble de transitions, c'est-à-dire des triplets $t =< e, g, u >$ tels que :
  - $e \in E$;

  - g est une fonction de Σ vers $False, True$: c'est la garde de la transition, qui ne peut être tirée que si $g(\sigma) =$ True ;
  - u est une fonction de Σ dans Σ : c'est l'action de la transition, c'est-à-dire la conséquence du tir de la transition sur les valeurs des variables du modèle.
- C est un ensemble de contraintes que Σ doit respecter.
- σ_0 est un ensemble des valeurs initiales des variables de Σ.

Par définition, le modèle n'évolue pas pendant la simulation : ces éléments sont des constantes.

3.1.b État de la simulation

L'état actuel de la simulation est représenté par le quadruplet (σ, S, d, I), où :
- σ est l'ensemble des valeurs, dans cet état, des variables de Σ.
- S est l'échéancier, qui enregistre les délais choisis auxquels les évènements de E doivent survenir.
- d est la date actuelle de la simulation.
- I est un ensemble d'indicateurs, avec leur état actuel : notamment, le nombre d'histoires déjà générées par la simulation fait partie de I.

3.1.c Configuration de la simulation

Pour réaliser une simulation stochastique, les paramètres suivants doivent être définis :
- un nombre d'histoires à générer ;
- un critère d'arrêt pour que la simulation ne soit pas infinie : ce peut être un nombre de transitions, ou un temps de mission (c.-à-d. une date après laquelle les histoires ne seront plus générées).

Ces paramètres font partie de p.

3.1.d Déroulement d'une simulation

Une simulation stochastique se déroule de la façon suivante :
- Initialisation de la machine de simulation stochastique ;
- Pour chaque histoire à générer (d'après p) :
  - Initialisation de l'état de la simulation :

$$\sigma = \sigma_0$$

$$d = 0$$

$$S = \emptyset$$

  - Tant que le critère d'arrêt n'est pas satisfait :
    - Mise à jour de l'échéancier S :
      - Calcul des gardes :

$$\forall t =< e, g, u > \in T, \text{ calcul de } g(\sigma)$$

      - Insertion des transitions activées avec choix des délais :

$$\forall t \in T \text{et} \notin S, g(\sigma) = \textit{true} \rightarrow S = S \cup \{t, e.\text{delai}(\sigma)\}$$

- Suppression des transitions inactivées :

$$\forall t \in S, g(\sigma) = \mathit{false} \rightarrow S = S/t$$

- Choix de la ou des transitions $T' \subset S$ à tirer : plus petite date de tir, concurrence, synchronisations ;
- Avance de la date actuelle à la date de tir : $d = d + S.\text{date}(T')$
- Retrait de la ou des transitions de l'échéancier : $S = S \setminus T'$;
- Application des actions de la ou des transitions à l'état de la simulation :
  - Mise à jour directe des variables (applications des actions des transitions tirées) :

$$\tau = \sigma$$

$$\forall t \in T', < e, g, u >= t, \tau = u(\tau)$$

  - Mise à jour indirecte des variables (résolution des contraintes) :

$$\gamma = A(\tau)$$

  - Mise à jour des indicateurs : $I = I.\text{update}(\gamma, d)$
  - Remplacement de l'état : $\sigma = \gamma$
- Calcul et sortie des résultats de la simulation stochastique (I)

3.2 Exemple : Système de Transitions Gardées stochastiques

Les Systèmes de Transitions Gardées (GTS) sont la sémantique sous-jacente des automates AltaRica 3.0. Une description succincte est proposée en A.

Une machine de simulation stochastique d'un GTS est composée d'un GTS stochastique, de l'état courant de la simulation, et de la configuration de la simulation stochastique.

3.2.a Modèle

Le modèle de la machine de simulation stochastique correspond au GTS stochastique :

$$(\Sigma, E, T, C, \sigma_0) \equiv \langle V_{GTS} = S_{GTS} \cup F_{GTS}, E_{GTS}, T_{GTS}, A_{GTS}, \iota_{GTS}, delay, expectation \rangle$$

- L'ensemble des variables Σ correspond à V_{GTS} ;
- L'ensemble des évènements E correspond à l'ensemble des évènements e de E_{GTS}, auxquels sont associés les fonctions $delay$ et $expectation$;
- L'ensemble des transitions T correspond à T_{GTS} : les triplets $t =< e, g, u >$ de T correspondant aux transitions de T_{GTS} notées $e : G \rightarrow P$:
  - Les étiquettes d'évènements e sont les éléments de E correspondant aux évènements de E_{GTS} avec leurs distributions temporelles stochastiques et leurs espérances ;
  - Les gardes g correspondent aux gardes G ;
  - Les actions de transitions u correspondent aux instructions P.
- L'ensemble des contraintes C correspond à l'assertion A_{GTS} ;
- L'ensemble des valeurs initiales σ_0 correspond à ι_{GTS} ;

3.2.b Configuration de la simulation

La configuration de la simulation stochastique d'un GTS comporte :

- Un temps de mission, qui est le critère d'arrêt ;
- Le nombre d'histoires à générer.

Ces deux paramètres font partie de p.

La configuration de la simulation contient aussi la configuration des indicateurs, qui font partie de I qui compose l'état de la simulation.

3.2.c État de la simulation

L'état actuel de la simulation est représenté par le quadruplet (σ, S, d, I), où :

- σ est l'ensemble des valeurs, dans cet état, des variables de Σ.
- La définition des GTS temporisés ne mentionne pas explicitement l'existence d'un échéancier. Toutefois, un mécanisme doit être prévu pour garder en mémoire les transitions tirables et leurs dates de tir (dépendant de la fonction $delay$ et de l'oracle et calculée au moment où la transition devient tirable) : c'est le rôle de l'échéancier S ;
- La date actuelle de simulation est d ;
- L'ensemble d'indicateurs I, avec leur état actuel ;

3.2.d Simulation

Les différentes étapes de la simulation d'un GTS stochastique peuvent être décrites de façon naturelle par celles d'une machine de simulation stochastique. Quelques points particuliers sont à noter :

- La mise à jour directe des variables correspond à l'application de l'action de la transition $Update(P, \sigma)$, tandis que la mise à jour indirecte des variables correspond à la propagation par l'assertion $Propagate(A, \tau)$.
- À chaque pas de simulation d'un GTS, une seule transition est tirée :
  - Si plusieurs évènements sont prévus à la même date, le mécanisme d'espérance (expectation) permet, avec l'oracle, de choisir celui qui surviendra.
  - La synchronisation de transitions n'existe pas pour les GTS : elle existe pour les modèles AltaRica 3.0, mais lors de la mise à plat du modèle en un seul GTS, les transitions synchronisées sont fusionnées en des transitions simples.

3.3 Conséquences sur l'évaluation

L'évaluation d'un outil logiciel de simulation stochastique évènementiel est sujette à des contraintes techniques ou des contraintes dues au contexte industriel. Pour les respecter, une méthode d'évaluation doit avoir certaines caractéristiques.

3.3.a Évaluation par morceaux

Les modèles à étudier à l'aide de l'outil de simulation stochastique peuvent être divers. L'outil, en étant capable de gérer cette diversité, a un fonctionnement complexe. L'évaluation de son bon fonctionnement est donc complexe. Comme pour tout problème complexe, le diviser en plusieurs parties moins importantes peut faciliter la tâche.

Un outil de simulation stochastique évènementiel est la combinaison de trois fonctions :

- un mécanisme de mise à jour du modèle ;
- un échéancier ;
- le traitement statistique.

Son bon fonctionnement repose sur le bon fonctionnement de ces trois fonctions, de façon individuelle, et entre elles.

i Mécanisme de mise à jour du modèle et échéancier

Les sous-fonctions du mécanisme de mise à jour les plus classiquement sollicitées (issues de V.3.1.d) dépendent des caractéristiques du modèle :

- Mise à jour des variables : une transition tirée peut nécessiter le changement de valeur d'une seule variable, ou avoir pour conséquences des instructions complexes. La mise à jour des variables est alors sollicitée.
- Résolution des contraintes : le tir d'une transition peut avoir des conséquences limitées à un sous-ensemble limité du modèle (par exemple sur un seul composant), ou avoir des répercussions sur presque tout le modèle. Le modèle peut pouvoir être décomposé en plusieurs parties indépendantes, ou bien des variables sont utilisées dans tout le modèle, empêchant une telle décomposition. La résolution des contraintes est alors sollicitée.

De même, les sous-fonctions de l'échéancier les plus classiquement sollicitées (issues de V.3.1.d) dépendent des caractéristiques du modèle :

- Le nombre d'évènements dans l'échéancier : cette quantité peut évoluer fortement durant une simulation. Plus l'échéancier contient de transitions, plus sa gestion peut être coûteuse, notamment lors du choix de la ou des transitions à tirer.
- Le nombre d'activations et de désactivations de transitions : à chaque pas de simulation, l'activation d'une transition implique le calcul d'un délai stochastique et l'inclusion de l'évènement correspondant dans l'échéancier. De même, la désactivation d'une transition implique la suppression de l'évènement correspondant de l'échéancier. C'est pourquoi un grand nombre d'activations et de désactivations à chaque pas de simulation peut dégrader les performances de l'échéancier.
- La synchronisation par transitions : certains modèles peuvent contenir des synchronisations de transitions, qui sont gérées par l'échéancier. Un grand nombre de telles synchronisations peut être coûteux.

L'approche proposée doit tester des parties délimitées de l'outil, de façon aussi indépendante que possible. Une fois qu'une partie a été testée avec succès, elle est supposée fonctionner correctement et peut être utilisée lors de l'évaluation d'autres parties de l'outil.

3.3.b Évaluation basée sur l'expérimentation

La simulation stochastique utilise de façon importante un générateur de nombre pseudo-aléatoire. De plus, la diversité des modèles qui peuvent être étudiés est considérable. C'est pourquoi il est très difficile de "prouver" ou de "vérifier" ce type d'outils.

De façon pragmatique, l'évaluation d'un simulateur stochastique doit donc utiliser l'expérimentation, afin d'améliorer suffisamment la confiance de l'utilisateur dans son bon fonctionnement.

3.3.c Comparaison des résultats

Les résultats obtenus ne peuvent être considérés comme corrects sans références auxquelles les comparer. Il faut donc comparer les résultats obtenus par simulation stochastique à des résultats obtenus par d'autres outils ou techniques. Les autres outils traditionnellement utilisés en sûreté de fonctionnement peuvent être de bons choix : arbres de défaillances, chaînes de Markov... Certains modèles peuvent aussi avoir des solutions analytiques utilisables pour une telle comparaison.

3.3.d Répétition des expériences

La simulation stochastique consiste à générer aléatoirement des histoires du modèle, pour en estimer des grandeurs probabilistes. Les résultats obtenus ne sont donc pas des valeurs exactes, mais des estimations probabilistes avec des intervalles de confiance. Par exemple, une estimation d'un temps moyen entre défaillances (MTBF) peut être donnée par une simulation stochastique avec un intervalle de confiance à 95% : il y a alors une probabilité de 5% que la vraie valeur de la MTBF de ce modèle soit en dehors de cet intervalle de confiance. Une expérience peut donc donner un intervalle de confiance qui ne contient pas la vraie valeur : avec ce caractère aléatoire, le tirage aléatoire des mesures peut être "malchanceux", c'est-à-dire que certains comportements peuvent être sur-représentés dans l'échantillon. La qualité des résultats d'un simulateur stochastique ne peut donc pas être évaluée par une unique expérience.

Il est donc important de répéter les expériences, afin de mesurer la proportion de "résultats malchanceux" (dont l'intervalle de confiance obtenu ne contient pas la vraie valeur). Par exemple, si 95% des expériences donnent un intervalle de confiance à 95% contenant la vraie valeur (et donc 5% sont des "résultats malchanceux"), alors l'outil donne des résultats corrects, avec une bonne évaluation de l'intervalle de confiance. Si cette proportion n'est pas respectée, alors il peut y avoir un problème : avec l'outil, avec le modèle, ou avec la référence utilisée pour connaître la valeur vraie.

3.3.e Coût de l'évaluation

L'évaluation de l'outil étant réalisée par expérimentation, il est nécessaire d'y consacrer du temps et de la puissance de calcul. Dans un contexte industriel, le temps et la puissance de calcul sont contraints. La méthodologie d'évaluation doit permettre de prévoir ces coûts de temps et de puissance de calcul, et de dimensionner les expériences au budget alloué (en temps et en puissance de calcul).

La méthodologie d'évaluation doit aussi privilégier des expériences les plus utiles possible, afin d'utiliser le temps de calcul de façon efficace.

4 Méthodologie d'évaluation et construction du kit

Pour évaluer les caractéristiques attendues d'un outil de simulation stochastique évènementiel (V.2), nous proposons une méthodologie qui respecte les contraintes énumérées en section V.3.3.

Cette méthodologie est en deux parties : le calcul des statistiques est d'abord testé, puis la qualité des résultats et les performances sont évaluées.

4.1 Calcul des statistiques

Le résultat attendu d'un outil de simulation stochastique étant sous forme de statistiques, l'évaluation du bon fonctionnement de leur calcul est importante. De plus, elle peut être réalisée de façon quasi indépendante du bon fonctionnement de l'échéancier et du mécanisme de mise à jour du modèle.

4.1.a Modèles déterministes

Pour tester le calcul des statistiques sans impliquer les parties stochastiques du logiciel, un premier ensemble de modèles déterministes (non stochastiques) est à construire. Le but est d'évaluer le bon fonctionnement de tous les calculs statistiques en comparant les résultats obtenus à des valeurs connues. En utilisant des modèles simples, les fonctions de mise à jour du modèle et l'échéancier ne sont pas utilisés de façon importante. Ces modèles déterministes à construire dépendent des capacités de l'outil de simulation stochastique, et de ce qui en est attendu dans les études envisagées.

Par exemple, si les études envisagées demandent le calcul de temps moyens avant défaillance (MTTF), l'un des modèles déterministes peut être aussi simple qu'un unique composant avec une défaillance déterministe. La simulation stochastique de ce modèle devrait donner une MTTF précisément égale au délai déterministe configuré pour la défaillance.

Une fois ces modèles construits, les résultats de leurs simulations doivent être exactement ceux attendus. Comme ces modèles sont simples, les temps de calcul devraient être petits, voire négligeables.

4.1.b Modèles stochastiques

Une fois les essais avec des modèles déterministes effectués, des modèles stochastiques peuvent être utilisés afin de tester le calcul des statistiques sur ce type de modèles, et les fonctions stochastiques de la simulation.

En fonction des estimations probabilistes que l'on souhaite obtenir, plusieurs comportements stochastiques doivent être modélisés. La construction des modèles peut réutiliser les modèles déterministes, en étendant leurs comportements pour qu'ils soient stochastiques.

Par exemple, le modèle déterministe pour calculer une MTTF peut être étendu en changeant le délai déterministe pour un délai stochastique, par exemple par une loi de Weibull afin d'avoir confiance dans le bon calcul des délais associés à ce type de loi. Des difficultés de calcul d'estimations probabilistes peuvent aussi être testées par ces modèles stochastiques : par exemple, on pourra tester le bon fonctionnement du calcul de l'écart-type de distribution dans le cas d'évènements avec deux valeurs possibles de délais.

Le temps de calcul de la simulation de ces modèles peut être facilement estimé par rapport au temps de calcul de la simulation des modèles déterministes. Les résultats doivent être comparés à ceux attendus : par exemple, les résultats de la simulation d'une loi de Weibull (ou similaire) sont connus analytiquement.

	Failure	Repairable Failure	Limited repair crew	Failure on Demand	Markov Chain components	Markov Chain Hidden failure	Fault Tree	Round Robin	Mesh	Game Of Life
Mise à jour du modèle										
Mise à jour des variables										+
Résolution des contraintes				+			+	+	+	
Échéancier										
Évènements dans l'échéancier	+	+	+	+	+	+	+			
Activations & désactivations					+	+		+		
Synchronisation par transitions			+	+						+

FIGURE V.1 – Caractéristiques présentées par les modèles du kit d'évaluation AltaRica 3.0

4.1.c Conclusion

Cette première étape permet de s'assurer du bon fonctionnement du calcul statistique de l'outil. Les étapes suivantes peuvent donc l'utiliser pour tester le bon fonctionnement des autres fonctions de l'outil.

4.2 Mécanisme de mise à jour du modèle et échéancier

Le mécanisme de mise à jour du modèle et l'échéancier travaillent de façon intriquée. C'est pourquoi il est difficile de tester l'un sans impliquer l'autre. Toutefois, des modèles judicieusement choisis ne vont utiliser qu'une partie de chacun, et permettent alors de tester de façon semi-indépendante différentes parties. On utilise pour cela la décomposition en différentes caractéristiques proposée en V.3.3.a.

4.2.a Construction de modèles

Un ensemble de modèles d'étude, qui utilisent chacun un sous-ensemble de ces caractéristiques, doit être construit. Afin de pouvoir solliciter l'outil dans plusieurs dimensions, l'idéal est qu'ils soient définis de façon paramétrique : un modèle arbitrairement "grand" peut ainsi être testé. Leurs résultats doivent pouvoir être vérifiés à l'aide d'autres outils, comme des outils classiques de sûreté de fonctionnement (chaînes de Markov, arbres de défaillances ...), des outils de vérification informatiques classiques (model-checking), ou bien pouvoir être obtenus de façon analytique.

Pour l'évaluation du simulateur stochastique AltaRica 3.0, un tel ensemble de modèles a été défini. Ils sont essentiellement issus du domaine de la sûreté de fonctionnement. Une description complète de ces modèles [1], avec les méthodes de vérification des résultats obtenus, est disponible dans l'annexe C. Chacun utilise des combinaisons de caractéristiques différentes, comme le montre la figure V.1.

1. Ces modèles ont été présentés dans [6].

4.2.b Temps de calcul

Une étude préliminaire peut être réalisée en utilisant ces modèles, afin d'estimer le temps de calcul nécessaire pour leur simulation et dimensionner les expériences. Plusieurs expériences sont à réaliser, en modifiant un paramètre à la fois :

- le nombre d'histoires générées dans une simulation stochastique ;
- le temps de mission d'une simulation ;
- la taille du modèle.

Les résultats obtenus permettent le tracé d'abaques concernant le temps de simulation nécessaire pour chaque modèle, en fonction des différents paramètres. L'analyse de ces abaques permet de déterminer pour quelles caractéristiques l'outil de simulation est efficace, ou peu efficace.

Pour réaliser un ensemble d'expériences avec un temps et une puissance de calcul contraints, les abaques permettent de choisir les ensembles de paramètres pour respecter ces contraintes.

4.2.c Vérification des résultats

Comme le résultat est une estimation probabiliste à partir de valeurs "tirées au hasard", la qualité des résultats n'est pas directement qualifiable de bonne ou mauvaise : même avec un outil de simulation stochastique parfait, la possibilité d'avoir des "résultats malchanceux" existe. Pour un intervalle de confiance à 95%, la probabilité d'un "résultat malchanceux" d'un tel outil parfait est de 5%. Sur une campagne d'expériences, il est donc normal et attendu d'obtenir de tels résultats.

La campagne d'expériences doit donc être construite de façon à répéter les expériences et à mesurer la proportion de tels "résultats malchanceux".

4.2.d Conclusion

La réalisation de ces expériences donne des informations sur les performances, en temps de calcul, de l'outil de simulation, en fonction des caractéristiques du modèle que l'utilisateur veut simuler. De plus, le bon fonctionnement de l'outil est ainsi testé, en expérimentant le bon fonctionnement de chaque fonction et de leurs interactions.

Il est alors possible de conclure sur le bon fonctionnement de l'outil, et de son adéquation avec le contexte industriel (contraintes) et les études (modèles) envisagées.

4.3 Limitations de cette méthode

Une des limitations de cette méthode concerne le générateur de nombres pseudo-aléatoires. Bien que les expériences proposées l'utilisent et que les résultats obtenus sont comparés à des références, elles ne testent pas les propriétés classiques que l'on peut attendre d'un tel générateur, comme l'uniformité et la périodicité.

Un générateur de nombres pseudo-aléatoires non uniforme peut falsifier les statistiques calculées de la probabilité d'un évènement redouté. Une période trop faible limite le nombre d'histoires différentes qu'un simulateur stochastique est capable de générer, et annule le bénéfice de la génération d'un grand nombre d'histoires puisque plusieurs d'entre elles seront anormalement identiques.

Le test d'un générateur de nombres pseudo-aléatoires est difficile, et fait l'objet d'intenses recherches notamment de la part du domaine de la cryptographie, qui propose des kits d'évaluation dédiés à ce problème spécifique [49].

5 Résultats expérimentaux

Un kit d'évaluation pour le simulateur stochastique AltaRica 3.0 a été créé en utilisant la méthode décrite dans la section précédente. Quelques résultats d'expériences sont présentés ci-dessous.

Le simulateur stochastique AltaRica 3.0 étant une chaîne d'outil, la performance en temps de calcul doit prendre en compte le délai entre le début de la chaîne, et sa fin (l'obtention de résultats). Toutefois, les temps de calcul pour les étapes préliminaires sont constants pour un modèle donné, et deviennent rapidement négligeables devant le temps de calcul utilisé par l'exécution du simulateur stochastique spécifique au modèle à mesure que le nombre d'histoires générées augmente. Les résultats d'expériences présentés ne contiendront donc que le temps de calcul de cette exécution.

Toutes les expériences ont été réalisées sur un ordinateur avec Microsoft Windows 7, un processeur à 2 coeurs de 2.30GHz avec SMT, et 8 Go de mémoire vive.

5.1 Estimation des grandeurs probabilistes

Le simulateur stochastique AltaRica 3.0 estime des grandeurs probabilistes à partir de mesures d'indicateurs. Ces indicateurs sont liés à des observateurs AltaRica 3.0, et donnent des informations sur l'évolution de la valeur de l'observateur durant chaque histoire générée. Les observateurs AltaRica 3.0 désignent des valeurs d'intérêt du modèle, et peuvent être de type booléen, entier, réel, ou énuméré. Plusieurs indicateurs sont disponibles : la valeur de l'observateur à une date spécifique, la valeur moyenne durant une histoire, la date de la première occurrence d'une valeur spécifique, le temps moyen entre deux occurrences d'une valeur spécifique...

5.1.a Modèles déterministes

Un ensemble de modèles déterministes ont été conçus pour tester chaque indicateur que le simulateur stochastique AltaRica 3.0 propose. Ces modèles sont simples, et plusieurs modèles permettent de tester un même indicateur : un sans particularité, et d'autres pour tester des limites ou des cas particuliers des indicateurs. Par exemple, pour l'indicateur permettant de mesure la première date à laquelle un observateur prend une valeur spécifique, le modèle sans particularité fait prendre cette valeur à une date déterminée, tandis que pour d'autres modèles la valeur est prise dès l'état initial de l'histoire, à la date $t = 0$, à une date supérieure au temps de mission simulé, ou encore qui la font prendre à plusieurs occasions, mais de manière furtive.

Les différents types (booléen, entier, réel, énuméré) des observateurs utilisés par les indicateurs sont pris en compte par des modèles de test similaires, mais distincts.

Comme ces modèles sont déterministes, il n'est pas utile de générer un grand nombre d'histoires : une quantité de seulement 10^4 histoires à générer a été retenue. Le temps de calcul a été de moins de 10 secondes.

Ces modèles ont originellement été conçus pour des tests fonctionnels durant les phases de conception et d'amélioration du simulateur stochastique AltaRica 3.0. Les expériences menées donnent donc les résultats attendus.

5.1.b Modèles stochastiques

Les comportements déterministes des modèles évoqués à la section précédente ont été étendus pour avoir un comportement stochastique : des délais déterministes d'évènements ont été remplacés par des lois temporelles stochastiques. AltaRica 3.0 permet l'utilisation des lois exponentielles (à taux constant) et de Weibull, mais aussi de définir des lois spécifiques de façon discrète (cf. IV.2.5.a). Plusieurs versions des modèles ont été générées, avec différentes combinaisons de lois de probabilité et de paramètres, pour chaque indicateur.

Des modèles supplémentaires ont été spécifiquement créés pour tester le résultat de séries stochastiques avec des propriétés particulières. Par exemple, pour l'indicateur permettant de mesurer la première date à laquelle un observateur prend une valeur spécifique, un modèle avec seulement deux dates possibles, choisies avec de grandes valeurs, mais un faible écart, génère une série de mesures par l'indicateur qui à une grande valeur moyenne, mais un écart-type faible : les calculs sur ce type de séries peuvent mettre en lumière des erreurs liées à la précision du calcul en virgule flottante. Sur le même indicateur, un modèle avec une probabilité de ne jamais rencontrer la valeur spécifique génère une série de mesures avec des valeurs absentes. Ces modèles permettent ainsi de tester le bon fonctionnement du calcul de l'estimation probabiliste dans de tels cas.

Comme les expériences sur les modèles déterministes, très proches des modèles stochastiques, ont été rapides, il n'a pas été réalisé d'étude préliminaire pour dimensionner cette expérience. Le nombre d'histoires générées par modèle a été conservé (10^4) : l'ensemble des simulations est réalisé en moins d'une minute.

5.1.c Résultats

Les résultats de ces deux étapes n'ont pas montré d'anomalies. L'estimation des grandeurs probabilistes est donc considérée comme fonctionnant correctement dans la suite de l'étude.

5.2 Mécanisme de mise à jour du modèle et échéancier

Pour tester les différentes caractéristiques du mécanisme de mise à jour du modèle et de l'échéancier, les modèles présentés dans l'annexe C sont utilisés. Des scripts permettent de générer les modèles AltaRica 3.0 des dimensions souhaitées, suivant les paramètres de chaque cas d'étude.

Dans cette section sont présentés quelques résultats de ces expériences.

5.2.a Étude des performances en temps de calcul

i Influence du nombre d'histoires générées

Le nombre d'histoires générées à une influence directe sur le temps de calcul : le temps de calcul utilisé pour simuler les histoires est, normalement, proportionnel au nombre d'histoires générées. Mais cette quantité a aussi une influence sur les performances du calcul des

estimations probabilistes. Des algorithmes de calcul peuvent être peu efficaces avec un grand nombre de valeurs.

Une étude rapide sur plusieurs modèles (figure V.2) montre que quelque soit le modèle le temps de calcul à une asymptote linéaire. Les non-linéarités qui apparaissent pour les simulations avec un faible nombre d'histoires (en dessous de quelques milliers d'histoires, c'est-à-dire en dessous de quelques dixièmes de seconde de temps de calcul) sont facilement expliquées par des "coûts fixes" de l'exécution du simulateur stochastique spécifique au modèle : chargement de l'exécutable en mémoire, lecture de la configuration, écriture du fichier de résultats.

Les autres expériences sur les performances en temps de calcul peuvent donc être réalisées avec un nombre raisonnable d'histoires à générer, tant que le temps de calcul de ces "coûts fixes" est négligeable devant le temps de calcul de la simulation.

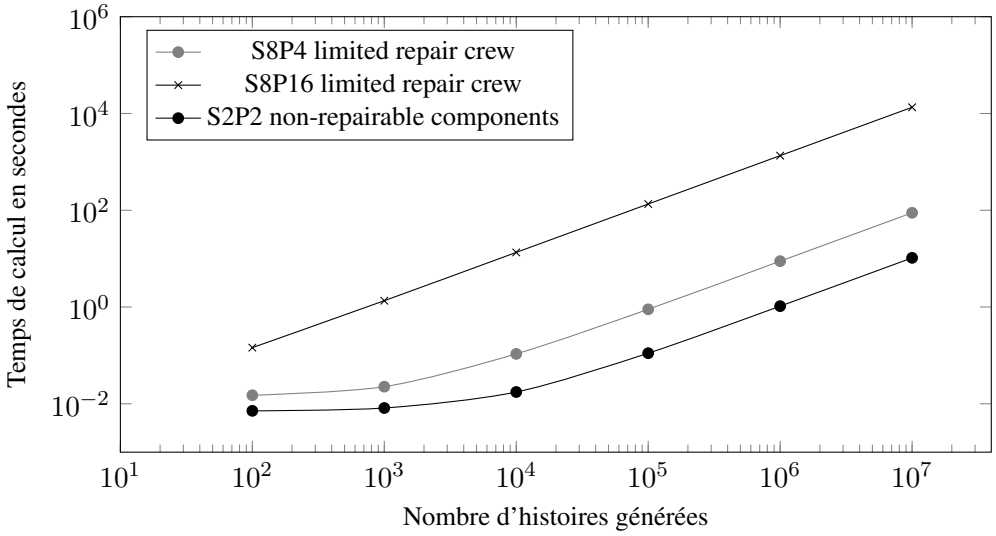

FIGURE V.2 – Temps de calcul en fonction du nombre d'histoires générées, pour différents modèles

ii Influence du temps de mission

Une étude de l'influence du temps de mission sur les performances en temps de calcul a été menée. Les résultats montrent que les performances sont affectées par le nombre moyen de transitions tirées par histoire, mais pas directement par le temps de mission (bien qu'un temps de mission plus élevé peut impliquer plus de transitions tirées par histoires générées).

Comme les modèles AltaRica 3.0 sont des automates évènementiels, et que le simulateur stochastique n'a rien à calculer pour les périodes entre deux évènements, aussi grandes soient-elles, ce résultat est cohérent. Le résultat obtenu serait différent pour un simulateur stochastique non évènementiel, par exemple par les simulateurs multiphysiques (Matlab/Simulink...).

iii Influence du nombre de composants dans les modèles série-parallèle

Les modèles série-parallèle sont des assemblages en séries d'ensembles de composants

assemblés en parallèle. Plusieurs comportements sont proposés dans les modèles du kit d'évaluation (voir C.1) Le nombre de composants de ces modèles a une conséquence directe sur le nombre de variables d'état et de flux, et le nombre d'évènements et de transitions du modèle. Certains comportements proposés demandent en plus des synchronisations par transitions ou par propagation des variables de flux. En faisant varier le nombre de composants, on fait donc varier la sollicitation de différentes fonctions du mécanisme de mise à jour du modèle et de l'échéancier. Il est ainsi possible de déterminer l'influence de l'usage de ces fonctions sur les performances de l'outil.

Les résultats des expériences (figure V.3) montrent que le comportement des réparateurs limités, qui est modélisé en utilisant des synchronisations de transitions, n'affecte pas les performances de l'outil : les modèles de tailles similaires avec et sans ce comportement sont simulés à la même vitesse. De plus, le temps de calcul augmente linéairement avec le nombre de composants. Ce résultat n'est pas surprenant, car la mise à plat du modèle AltaRica 3.0 en GTS dans la chaîne d'outils du simulateur stochastique AltaRica 3.0 substitue des transitions simples aux transitions synchronisées, en combinant les gardes et les actions. Il n'y a donc pas de différence, dans l'exécutable de simulation stochastique spécifique au modèle, entre une transition synchronisée et une transition simple.

En revanche, les résultats (figure V.3) montrent que les performances sont dégradées lorsqu'un comportement avec démarrage à froid est utilisé. Ce comportement est modélisé en utilisant des synchronisations par propagations (des assertions AltaRica 3.0). Pour des modèles de dimensions similaires, ceux avec ce comportement demandent des temps de calcul supérieurs. De plus, le temps de calcul n'augmente plus linéairement avec le nombre de composants, mais quadratiquement. D'un point de vue d'utilisateur de l'outil, il paraît judicieux de restreindre les modélisations utilisant ces synchronisations par propagation, d'utiliser un autre outil si elles sont nombreuses et inévitables, ou d'en payer le coût en puissance de calcul. Du point de vue du concepteur de l'outil, la performance de cette fonction serait à améliorer.

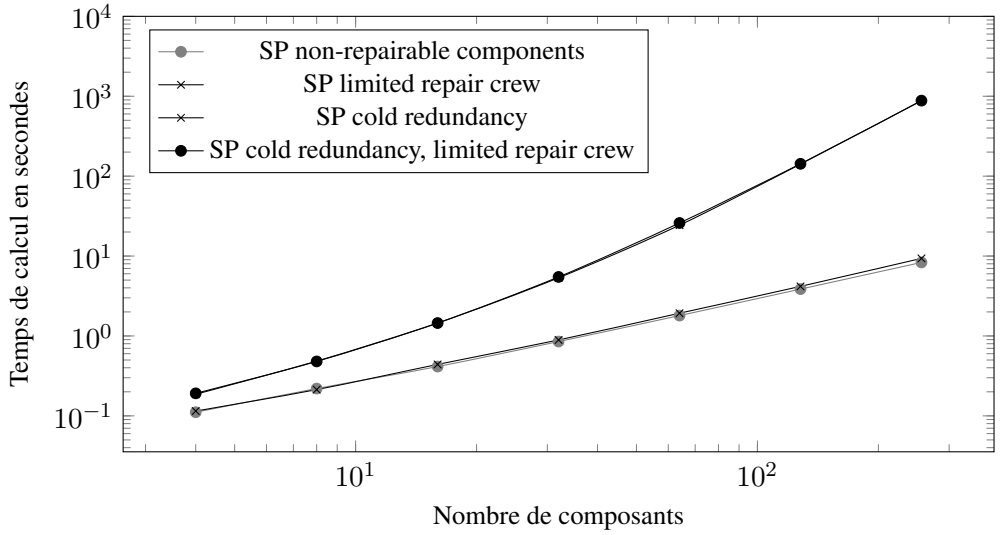

FIGURE V.3 – Temps de calcul par rapport au nombre de composants dans un modèle série-parallèle, pour divers comportements

iv Conclusion

Les résultats de ces expériences permettent d'obtenir des abaques du temps de calcul nécessaire à la simulation des différents modèles du kit d'évaluation, en fonction de leurs dimensions et de la configuration de la simulation (temps de mission, nombre d'histoires). En plus de donner des indications sur les performances des différentes fonctions de la simulation stochastique de l'outil, ils préparent à l'étape suivante de l'évaluation en permettant de dimensionner les modèles et choisir les paramètres, afin de réaliser les expériences d'évaluation des résultats en respectant des contraintes de temps de calcul disponible.

5.2.b Évaluation des résultats

Pour comparer les résultats obtenus par l'outil de simulation stochastique, les modèles sont étudiés en utilisant d'autres méthodes (principalement des chaînes de Markov et des arbres de défaillance, tels que décrits dans l'annexe C) pour obtenir des valeurs de référence. Une contrainte d'expérimentation de 60 heures de calcul a été fixée [2]. Un plan d'expérimentation a été établi :

- en respectant la contrainte de temps de calcul disponible, grâce aux abaques calculés lors de l'expérience précédente ;
- en réitérant plusieurs fois chaque simulation ;
- en variant les modèles, leurs dimensions, et les paramètres de simulation.

Ce plan d'expérimentation est composé de 1920 simulations stochastiques.

Le résultat pertinent, pour cette étude, de chaque simulation est la position de la valeur de référence par rapport à l'intervalle de confiance calculé. La distribution de cette position relativement à l'intervalle de confiance est montrée figure V.4.

La plupart des résultats contiennent la valeur de référence dans l'intervalle de confiance, mais certains sont des "résultats malchanceux" : l'intervalle de confiance calculé ne contient pas la valeur de référence. Avec plusieurs taux d'intervalle de confiance (95%, 99% et 99,9%), la proportion de résultats corrects est toujours proche du taux de confiance, comme attendu.

Un résultat différent, en donnant une proportion d'intervalles de confiance calculés contenant la valeur de référence éloignée du taux de confiance, signalerait une anomalie [3]. Si cette proportion est anormale uniquement pour un modèle ou un groupe de modèles déterminé, alors l'anomalie peut provenir du calcul de la valeur de référence. Sinon, elle peut provenir d'une différence de modélisation entre le modèle simulé stochastiquement et celui utilisé pour l'obtention de la valeur de référence, ou d'un problème dans l'outil de simulation stochastique.

5.3 Résultats de l'évaluation

Les résultats des différentes expériences de cette évaluation du simulateur stochastique AltaRica 3.0 ont permis de tester le bon fonctionnement des différentes fonctions de la simulation stochastique : les résultats sont conformes à ceux attendus d'une telle simulation.

Comme attendu, des résultats obtenus sont, avec une faible occurrence, éloignés des valeurs de référence : ce type de résultats est inhérent à la simulation stochastique, et peut être

2. Cette contrainte est arbitraire : elle correspond au nombre d'heures entre le vendredi 18h et le lundi 6h.

3. Ce cas de figure est survenu dans une version précédente ce cette expérience, et a permis de corriger une erreur dans le calcul de la valeur de référence.

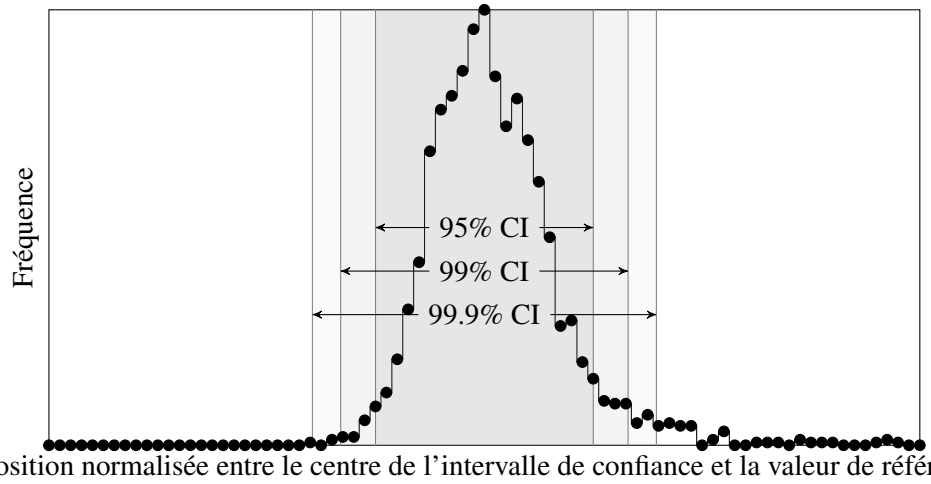

FIGURE V.4 – Distribution de l'erreur (position relative entre le centre de l'intervalle de confiance et la valeur de référence, normalisée par la largeur de l'intervalle de confiance), sur une expérience de 1920 simulations stochastiques

minimisé en augmentant le nombre d'histoires générées. Les résultats des simulations stochastiques obtenus sont comparables à ceux d'autres méthodes d'analyse, en prenant en compte l'intervalle de confiance. Mais il ne s'agit pas d'un test exhaustif, et encore moins d'une preuve du bon fonctionnement : celui-ci n'est donc pas certain. Ces résultats ne peuvent donc qu'améliorer la confiance de l'utilisateur dans l'outil.

Cette évaluation a aussi permis de mesurer les performances en temps de calcul de l'outil, en fonction des paramètres de simulation et des caractéristiques des modèles simulés. Les abaques obtenus par les expériences donnent une vue d'ensemble du comportement, en ce qui concerne le temps de calcul, de l'outil en fonction des caractéristiques des modèles (taille, nombre de composants, particularités…) et de la simulation (nombre d'histoire, temps de mission, fréquence des évènements). Par exemple, la simulation de comportements modélisés par des synchronisations par propagation apparait coûteuse.

6 Conclusion

La méthode proposée dans ce chapitre permet l'évaluation, pour un simulateur stochastique pour la sûreté de fonctionnement, de ses performances (en termes de temps de calcul nécessaire) et de qualité de ses résultats, en prenant en compte les besoins de l'utilisateur. L'application de cette méthode produit un kit d'évaluation, comprenant des modèles de test à implémenter et à simuler avec l'outil. Les résultats des simulations stochastiques sont comparés à ceux d'autres méthodes, afin d'améliorer la confiance dans le bon fonctionnement de l'outil, et pour détecter d'éventuels problèmes qu'il pourrait comporter. Ces mêmes simulations sont utilisées pour mesurer ses performances et évaluer le coût de son utilisation.

Cette méthode a été appliquée sur le simulateur stochastique AltaRica 3.0, et a permis d'observer la bonne qualité de ses résultats, et d'obtenir des relations entre ses performances et les caractéristiques des modèles (comportements modélisés, tailles des modèles). Elle est applicable à d'autres outils de simulation stochastique, et peut être adaptée pour évaluer d'autres outils d'analyse de sûreté de fonctionnement.

Conclusion

Les travaux de cette thèse portent sur le calcul d'indicateurs de sûreté de fonctionnement par simulation stochastique de modèles AltaRica 3.0. Plusieurs aspects de l'utilisation de cette méthode ont été abordés.

L'utilisation d'expériences numériques pour mesurer des performances d'une solution technique nécessite la création d'un modèle. Les exigences de sûreté de fonctionnement portant sur les performances du système, il est nécessaire de faire correspondre les critères de ces exigences aux éléments du modèle. C'est le rôle des observateurs AltaRica 3.0, mais ils doivent être étendus aux évènements pour permettre la mesure de l'ensemble des comportements pertinents.

Les exigences de sûreté de fonctionnement portent sur des grandeurs probabilistes. La simulation stochastique permet de générer des histoires possibles de l'évolution du modèle. Un ensemble d'indicateurs permet à l'analyste de sûreté de fonctionnement d'exprimer les calculs permettant d'obtenir, à partir de ces histoires, les grandeurs probabilistes qui l'intéressent.

La qualité des résultats d'une simulation stochastique est proportionnelle au nombre d'histoires qui ont été générées. Le coût, en temps de calcul, peut alors devenir important. Pour permettre l'utilisation de la simulation stochastique dans un contexte industriel, il est nécessaire de disposer d'un outil performant. Une démarche d'amélioration des performances de l'outil de simulation stochastique de modèles AltaRica 3.0 a été utilisée, et a conduit à plusieurs modifications des algorithmes de simulation.

Les systèmes qui nécessitent des études de sûreté de fonctionnement possèdent à la fois une partie mécanique, et une partie logicielle les contrôlant. Le cas d'étude sur un système mécatronique critique (train d'atterrissage d'avion) a permis de montrer que :

- le langage AltaRica 3.0 permet la modélisation et l'étude de tels systèmes complexes ;
- la simulation stochastique permet d'obtenir des résultats intéressants en des temps raisonnables : cette méthode peut donc être envisagée pour mener des études de sûreté de fonctionnement de ce type de système.

Le choix des outils logiciels, utilisés dans une étude de sûreté de fonctionnement, doit faire l'objet d'un choix argumenté par l'analyste de sûreté de fonctionnement : il doit avoir confiance dans les résultats obtenus par ces outils pour prendre ses décisions. Les outils de simulation stochastique présentent des particularités rendant leur évaluation complexe. Une méthodologie d'évaluation de ces outils a été proposée, et appliquée à l'outil de simulation stochastique de modèles AltaRica 3.0.

Les différents travaux présentés dans ce manuscrit ont été menés avec pour objectif commun de permettre l'utilisation de la simulation stochastique comme méthode de calcul. La complexité grandissante des systèmes critiques rend le calcul d'indicateurs de sûreté de fonctionnement par les méthodes classiques difficile. La simulation stochastique permet l'analyse de modèles plus représentatifs des solutions techniques étudiées. Les contraintes et difficultés d'utilisation de cette méthode (exploitation des résultats, coût des calculs, évaluation des outils...) peuvent être surmontées pour permettre son utilisation dans un contexte industriel.

Annexes A

Systèmes de transitions gardées (GTS)

1 Définition

Un système de transitions gardées (GTS) est un quintuplet $\langle V, E, T, A, \iota \rangle$:

- $V = S \uplus F$ est un ensemble de variables, divisées en deux ensembles disjoints :
  - S est l'ensemble des variables d'état, décrivant l'état du système ;
  - F est l'ensemble des variables de flux, permettant l'échange d'informations entre composants (classiquement des liens entrée-sortie).
- E est un ensemble de symboles, appelés évènements ;
- T est un ensemble de transitions, notées $e : G \rightarrow P$;
- A est une assertion, c'est-à-dire une instruction sur V ;
- ι est une affectation de variables de V, appelé valeurs initiales ou par défaut.

Une transition $e : G \rightarrow P$ est un triplet $\langle e, G, P \rangle$ où $e \in E$ est un évènement, G est une garde (une formule booléenne construite sur V), et P est une instruction construite sur V et appelée action ou post-condition. Une transition $e : G \rightarrow P$ est dite *tirable* dans un état donné σ (c'est-à-dire pour un ensemble de valeurs de variables σ) si sa garde G est satisfaite dans cet état (la formule booléenne a pour valeur *true*).

2 Instructions

Les assertions et les actions des transitions sont des instructions. Une instruction peut être :

- L'instruction vide, notée $skip$;

- Une affectation notée $v := E$, où v est une variable et E une expression construite sur les variables de V ;
- Une affectation conditionnelle notée "if C then I", où C est une expression booléenne et I une instruction ;
- Un bloc d'instructions, noté $\{I_1, \ldots, I_n\}$.

Le membre gauche d'une assignation dépend de l'instruction : s'il s'agit de l'action d'une transition, alors seule une variable d'état peut être assignée. S'il s'agit d'une assertion, alors seule une variable de flux peut être assignée.

2.1 Application des instructions

Soit σ les valeurs des variables avant le tir de la transition $e : G \rightarrow P$. L'application de l'instruction P sur σ consiste à calculer les nouvelles valeurs des variables τ comme suit :
Au départ, $\tau = \sigma$, puis :

- Si P est une instruction vide, alors τ est inchangé ;
- Si P est une assignation $v := E$, alors $\tau(v)$ est affecté de la valeur de $\sigma(E)$. Si la valeur de v a déjà été modifiée et est différente de celle de ce calcul, alors une erreur est signalée ;
- Si P est une assignation conditionnelle "if C then I" et $\sigma(C)$ vaut *true*, alors l'instruction I est appliquée sur τ ;
- Si P est un bloc d'instructions $\{I_1, \ldots, I_n\}$, alors les instructions $I_1, \ldots, I_n$ sont appliquées successivement sur τ.

Il est important de noter que dans le mécanisme ci-dessus, les membres droits des assignations et les expressions conditionnelles sont évaluées dans le contexte de σ. Le résultat ne dépend donc pas de l'ordre dans lequel les instructions d'un bloc sont appliquées : elles peuvent donc l'être de façon parallèle.

On note Update(P, σ) les valeurs des variables τ, résultant de l'application de l'instruction P sur σ.

2.2 Propagation des assertions

Soit A l'assertion, et τ les valeurs des variables obtenues après l'application de l'action d'une assertion. L'application de A consiste à calculer de nouvelles valeurs de variables (qui ne va différer que pour les variables de flux) noté π. Soit D un ensemble de variables de flux non évaluées, valant au départ $D = F$. On commence par assigner à chaque variable d'état de π sa valeur dans τ :

$$\forall v \in S, \pi(v) = \tau(v)$$

Puis :

- Si A est une instruction vide, alors π est inchangé ;
- Si A est une assignation $v := E$, alors si $\pi(E)$ peut être évalué dans le contexte de π, c'est-à-dire si toutes les variables présentes dans E ne sont pas dans D, alors $\pi(v)$ prend la valeur de $\pi(E)$ et v est retirée de D. Si la valeur de v a déjà été calculée et est différente de celle calculée, alors une erreur est signalée ;
- Si A est une assignation conditionnelle "if C then I" et $\pi(C)$ peut être évaluée dans le contexte de π et $\pi(C)$ vaut *true*, alors l'instruction I est appliquée sur π. Sinon, π est inchangée.

- Si A est un bloc d'instructions $\{I_1, \ldots, I_n\}$, alors les instructions $I_1, \ldots, I_n$ sont appliquées sur π **de façon répétée** jusqu'à ce qu'il n'y ai plus de possibilité d'assigner une variable de flux.

Si après avoir appliqué A sur π il reste des variables non évaluées dans D, alors ces variables prennent pour valeur leurs valeurs de défaut :

$$\forall v \in D, \pi(v) = \text{reset}(v)$$

et A est appliquée sur π pour vérifier que toutes les assignations sont vérifiées. Si ce n'est pas le cas, une erreur est signalée.

On note Propagate(A, τ) les valeurs de variables obtenues par l'application de l'instruction A sur τ.

3 Observateurs

Les observateurs sont des expressions nommées qui dépendent de variables de V et sont évalués à chaque nouvel état du système. Ils ne peuvent pas être utilisés dans des transitions ou des assertions, et ne peuvent donc pas être utilisés pour décrire le comportement du système.

Comme leur nom l'indique, ce sont des quantités observables : ils sont utilisés pour définir ce qui peut être utile de connaître du modèle, et constituent un point de vue sur son état pour l'utilisateur.

4 Graphe d'accessibilité

Soit σ les valeurs des variables juste avant le tir d'une transition. Le tir de la transition transforme σ en Fire$(e : G \to P, A, \sigma)$, défini par :

$$\text{Fire}(e : G \to P, A, \sigma) = \text{Propagate}(A, \text{Update}(P, \sigma))$$

Les GTS sont des représentations implicites de structures de Kripke, c'est-à-dire de graphes dont les nœuds sont étiquetés par des assignations de variables et dont les arcs sont étiquetés par des évènements. Plus exactement, la sémantique d'un GTS $\langle V = S \uplus F, E, T, A, \iota \rangle$ est une structure de Kripke, c'est-à-dire un graphe $\Gamma = (\Sigma, \Theta)$, avec Σ un ensemble de valeurs de variables (aussi appelés états et représentant les noeuds du graphe), et Θ est un ensemble de triplets $\langle s, e, q \rangle$, avec $s, q \in \Sigma$ et $e \in E$, représentant les transitions du graphe. Γ est la plus petite structure de Kripke, telle que :

1. σ_0 est l'état initial de la structure de Kripke : $\sigma_0 = \text{Propagate}(A, \iota) \in \Sigma$;
2. Si $\sigma \in \Sigma$ et $\exists t = \langle e, G, P \rangle \in T$ tel que $\sigma(G) = TRUE$,alors l'état $\tau = \text{Fire}(P, A, \sigma)$ est dans Σ et la transition (σ, e, τ) est dans Θ.

Le calcul de $\Gamma = (\Sigma, \Theta)$ peut rencontrer des erreurs : un GTS correctement conçu évite ce problème. La structure de Kripke Γ est aussi appelée graphe d'accessibilité.

5 Système de transitions gardées temporisé et stochastique

Une structure de temps probabiliste peut être créée au-dessus des GTS pour obtenir des modèles temporisés stochastiques. L'idée est d'associer à chaque évènement les informations suivantes :

- Un délai qui peut être déterministe ou stochastique, et peut dépendre de l'état courant. Lorsqu'une transition étiquetée par l'évènement devient tirable à la date t, un délai d est calculé et la transition est tirée à la date $t + d$ si elle reste tirable de t à $t + d$;
- Un poids, aussi appelé espérance (expectation), qui est utilisé pour déterminer la probabilité que la transition soit tirée dans le cas où plusieurs transitions sont à tirer à la même date.

5.1 Système de transitions gardées temporisé

Un GTS temporisé est un tuple $\langle V, E, T, A, \iota, delay\rangle$, où $\langle V = S \uplus F, E, T, A, \iota\rangle$ est un GTS et $delay : E \to \mathbb{R}_+$ est une fonction qui associe à chaque évènement un nombre réel positif.

La sémantique d'un GTS temporisé peut être définie en termes d'exécutions. Une exécution est une séquence

$$(\sigma_0, d_0, t_0) \overset{e_0}{\to} (\sigma_1, d_1, t_1) \dots (\sigma_{n-1}, d_{n-1}, t_{n-1}) \overset{e_{n-1}}{\to} (\sigma_n, d_n, t_n)$$

où $t_i \in \mathbb{R}_+$ représente la date à laquelle l'évènement e_i survient et où la transition étiquetée par cet évènement est tirée. Le pas $(\sigma_i, d_i, t_i) \overset{e_i}{\to} (\sigma_{i+1}, d_{i+1}, t_{i+1})$ correspond à l'avancement du temps et au tir de la transition étiquetée par l'évènement e.

5.2 Système de transitions gardées stochastique

L'interprétation des GTS temporisés ne spécifie pas comment les délais sont calculés, et inclue donc le cas où les délais sont stochastiques. Pour définir la notion de stochastique, il est nécessaire d'introduire la notion d'oracle. L'idée est de déléguer toute la partie "aléatoire" d'une simulation à un oracle. De cette façon, le GTS est strictement déterministe, mais son comportement dépend du résultat d'un oracle. Toutes les parties non déterministes des exécutions sont donc concentrées dans l'oracle.

Formellement, un *oracle* o est une séquence infinie de nombres réels compris entre 0 et 1 (inclus) : $o : \mathbb{N} \to [0; 1] \subset \mathbb{R}$. La seule opération disponible d'un oracle est de consommer son premier élément. Cette opération retourne le premier élément et le reste de la séquence, qui est elle-même un oracle.

Enfin, un GTS stochastique est un tuple $\langle V = S \cup F, E, T, A, \iota, delay, expectation\rangle$, où :

- $\langle V = S \uplus F, E, T, A, \iota\rangle$ est un GTS ;
- $delay$ est une fonction des évènements et de l'oracle vers les nombres réels positifs ;
- $expectation$ est une fonction des évènements vers les nombres réels positifs.

Lorsque les tirs de plusieurs transitions sont prévus à la même date, une est choisie aléatoirement en utilisant l'oracle, par rapport à leurs espérances (expectation). La probabilité $p(e_k : G_k \to P_k)$ de tirer la transition $e_k : G_k \to P_k$ est définie par :

$$p(e_k : G_k \to P_k) = \frac{expectation(e_k)}{\sum\limits_{e_i : d_n(e_i)=0} expectation(e_i)}$$

Annexes B

Ensemble d'indicateurs

Notations

Les définitions des indicateurs utilisent des notations mathématiques : une synthèse des symboles utilisés est présentée dans la table B.2.

Occurrence

Les définitions des indicateurs utilisent la notion d'occurrence d'un observateur. Il y a une occurrence d'un observateur obs à la valeur P à la date t si :

$$\begin{cases} \exists \epsilon > 0, \forall t' \in [t-\epsilon, t[, \text{obs}(t') & \neq P \\ \exists \epsilon > 0, \forall t' \in [t, t+\epsilon], \text{obs}(t') & = P \end{cases}$$

On définit la fonction booléenne Occurrence(obs, P, t) qui vaut *true* si il y a une occurrence de obs à la valeur P à la date t, et *false* sinon. De façon plus intuitive : l'occurrence d'un observateur obs à la valeur P correspond à un front montant de la fonction $t \mapsto \text{obs}(t) = P$.

Indicateur	Type de l'observateur	Type de la mesure
Has Value	fini	booléen
First Occurrence Date	fini	date
Had Value	fini	booléen
Mean Time Between Occurrences	fini	date
Occurrences	fini	quantité
Sojourn Time	fini	quantité
Changes	infini	quantité
Mean Time Between Changes	infini	date
Mean Value	infini	quantité
Value	infini	quantité
Firings	évènement	quantité
First Firing Date	évènement	date
Has Been Fired	évènement	booléen
Mean Time Between Firings	évènement	date

TABLE B.1 – Ensemble des indicateurs retenus

X	Valeur de la mesure de l'indicateur
obs(t)	Valeur de l'observateur à la date t
d	Date de la mesure
P	Paramètre configuré
true, *false*	Valeurs du domaine booléen
Im(obs)	Domaine image de l'observateur

TABLE B.2 – Symboles utilisés dans la définition des indicateurs

1 Indicateurs d'observateurs booléens ou énumérés

1.1 Has Value

L'indicateur "Has Value" ("a la valeur") mesure la valeur de l'observateur à valeurs booléennes ou énumérées à la date de mesure, et la compare au paramètre P avec lequel l'indicateur est configuré. La mesure est un booléen : elle vaut *true* si la valeur de l'observateur est égale au paramètre, et *false* sinon (figure B.1).

Définition de l'indicateur **Has Value** :

$$X = \textit{true} \text{ si } (\text{obs}(d) = P), \textit{false} \text{ sinon}$$

Cet indicateur trouve son utilité lorsque l'information recherchée ne dépend que de l'état du modèle simulé à la date de mesure.

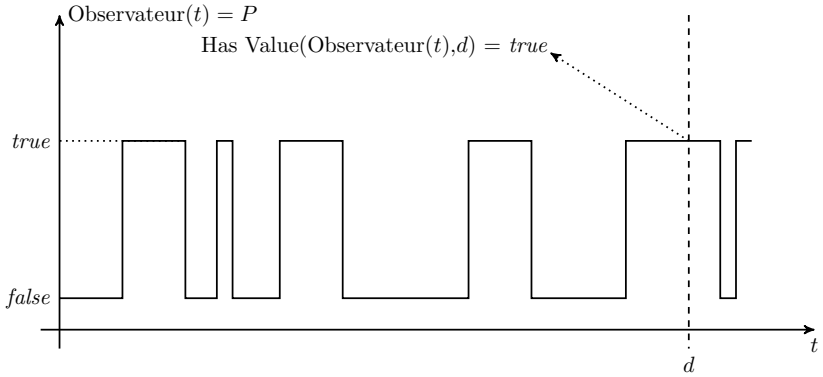

FIGURE B.1 – Exemple de mesure d'un indicateur Has Value

1.2 First Occurrence Date

L'indicateur "First Occurrence Date" ("date de première occurrence") mesure la date à laquelle l'observateur à valeurs booléennes ou énumérées prend pour la première fois de l'histoire, de façon non furtive, la valeur correspondante au paramètre P avec lequel l'indicateur est configuré (figure B.2).

Si, à la date d'observation, il n'y a eu aucune occurrence, alors la mesure de l'indicateur est "non disponible".

Définition de l'indicateur **First Occurrence Date** :

Si $\exists t \in [0, d]$, tel que : Occurrence(obs, P, t) = *true*
et $\forall t' \in [0, t[$, Occurrence(obs, P, t') = *false*
Alors $X = t$
Sinon $X =$ non disponible

Cet indicateur peut être utilisé pour évaluer un temps moyen de première panne (MTTF) d'un système.

1.3 Had Value

L'indicateur "Had Value" ("a eu la valeur") mesure si l'observateur à valeurs booléennes ou énumérées a pris, au cours de l'histoire, avant la date d'observation, et de façon non furtive, la valeur correspondante au paramètre P avec lequel l'indicateur est configuré : autrement dit, si il y a eu une occurrence de $obs(t) = P$ (figure B.3).

Définition de l'indicateur **Had Value** :

Si $\exists t \in [0, d]$, Occurrence(obs, P, t) = *true*
Alors $X =$ *true*, sinon $X =$ *false*

Cet indicateur peut être utilisé pour obtenir la probabilité qu'un sous-système réparable soit au moins une fois en panne durant un temps de mission.

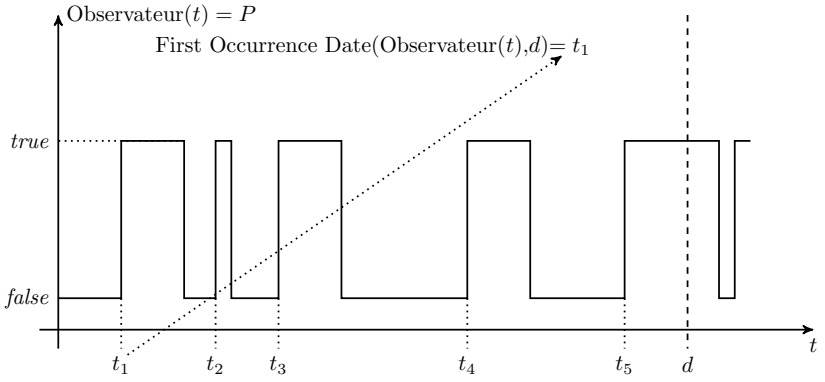

FIGURE B.2 – Exemple de mesure d'un indicateur First Occurrence Date

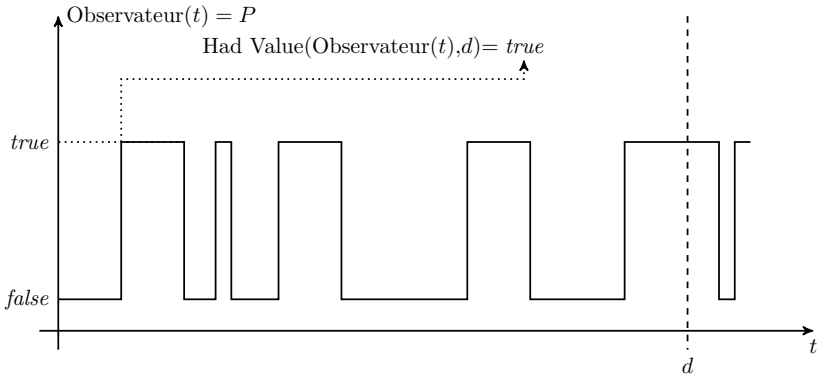

FIGURE B.3 – Exemple de mesure d'un indicateur Had Value

1.4 Mean Time Between Occurrences

L'indicateur "Mean Time Between Occurrences" ("temps moyen entre occurrences") mesure le temps moyen entre deux occurrences de l'observateur à valeurs booléennes ou énumérées à la valeur configurée P, tout au long de l'histoire (figure B.4).

Définition de l'indicateur **Mean Time Between Occurrences** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que [1] :

$$\begin{cases} \forall i \in [\![1, n-1]\!], & t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], & \text{Occurrence}(\text{obs}, P, t_i) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, & \text{Occurrence}(\text{obs}, P, t) = \textit{false} \end{cases}$$

$$\text{Si } n > 1, \quad \text{alors } X = \tfrac{1}{n-1}.\sum_{i=1}^{n-1}(t_{i+1} - t_i) = \frac{1}{n-1}.(t_n - t_1)$$
$$\text{sinon } X = \text{non disponible}$$

1. Il s'agit de la liste ordonnée et complète des dates d'occurrence de l'observateur à la valeur P.

Cet indicateur peut être utilisé pour mesurer une MTBF.

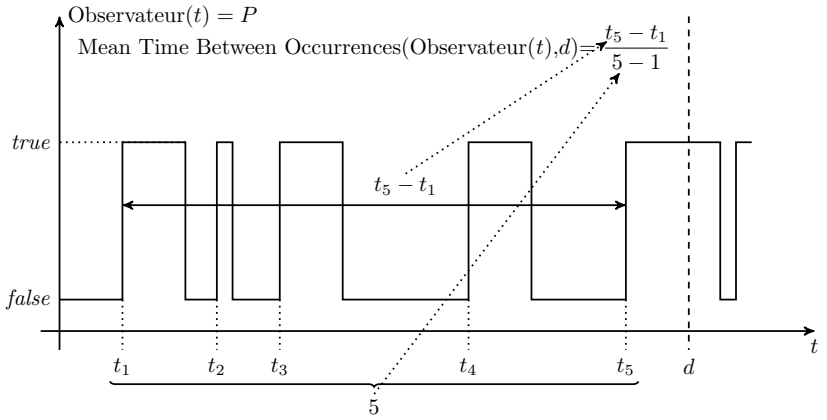

FIGURE B.4 – Exemple de mesure d'un indicateur Mean Time Between Occurrences

1.5 Occurrences

L'indicateur "Occurrences" mesure le nombre d'occurrences de l'observateur à valeurs booléennes ou énumérées à la valeur configurée P entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.5).

Définition de l'indicateur **Occurrences** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que :

$$\begin{cases} \forall i \in [\![1, n-1]\!], & t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], & \text{Occurrence}(\text{obs}, P, t_i) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, & \text{Occurrence}(\text{obs}, P, t) = \textit{false} \end{cases}$$

Alors $X = n$

Cet indicateur peut être utilisé pour mesurer le nombre d'arrêts de production d'une installation.

1.6 Sojourn Time

L'indicateur "Sojourn Time" ("temps de séjour") mesure le temps simulé de l'histoire (entre le début de l'histoire $t = 0$ et la date de mesure $t = d$) pendant lequel l'observateur à valeurs booléennes ou énumérées vaut la valeur configurée P (figure B.6).

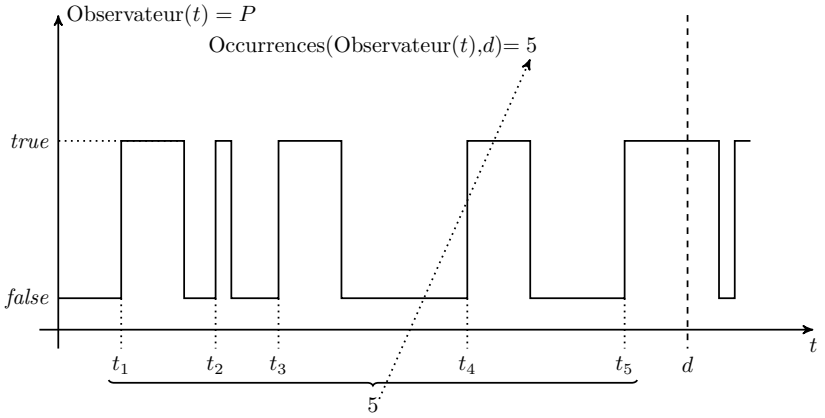

FIGURE B.5 – Exemple de mesure d'un indicateur Occurrences

Définition de l'indicateur **Sojourn Time** :
La définition de la mesure de l'indicateur utilise la fonction :

$$f_{\text{obs},P} : \begin{cases} \mathbb{R} \longrightarrow \mathbb{R} \\ t \longmapsto \begin{cases} 1 & \text{si } (\text{obs}(t) = P), \\ 0 & \text{sinon.} \end{cases} \end{cases}$$

La mesure de l'indicateur est alors définie par :

$$X = \int_0^d f_{\text{obs},P}(t)dt$$

Cet indicateur peut être utilisé pour mesurer un taux de disponibilité.

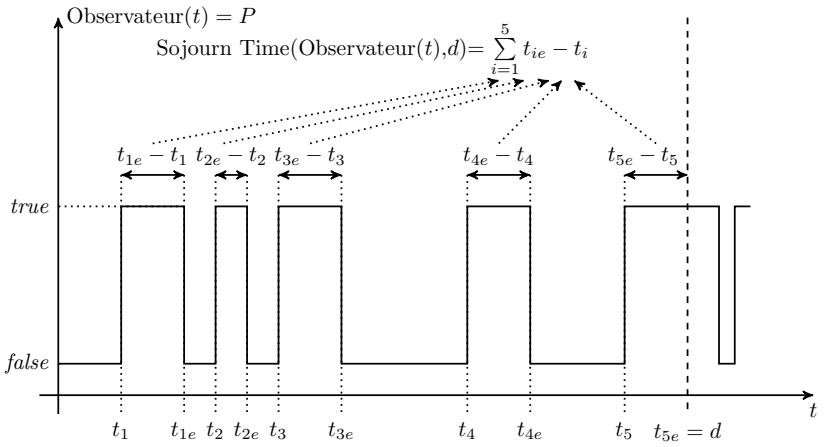

FIGURE B.6 – Exemple de mesure d'un indicateur Sojourn Time

2 Indicateurs d'observateurs réels ou entiers

2.1 Changes

L'indicateur "Changes" ("changements") mesure le nombre de changements de valeur d'un observateur à valeurs réelles ou entières, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.7).

Définition de l'indicateur **Changes** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que :

$$\begin{cases} \forall i \in [\![1, n[\![, t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], \exists v \in \text{Im(obs)} \text{ tel que Occurrence(obs}, v, t) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, \forall v \in \text{Im(obs)}, \text{Occurrence(obs}, v, t) = \textit{false} \end{cases}$$

Alors $X = n$

Cet indicateur peut être utilisé pour mesurer la stabilité d'une production.

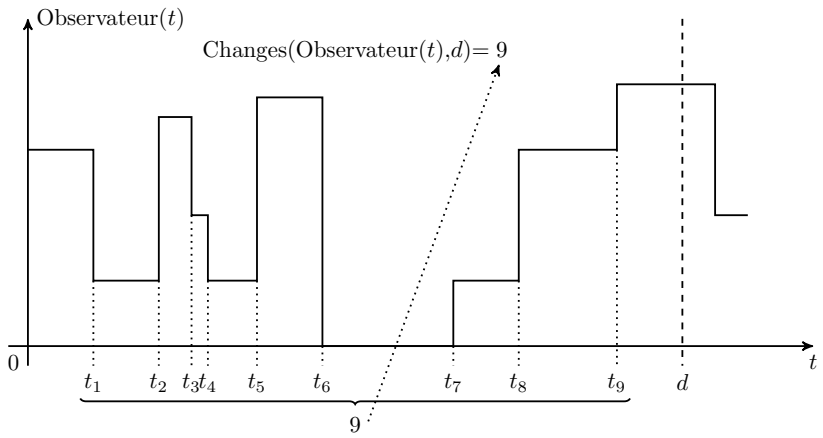

FIGURE B.7 – Exemple de mesure d'un indicateur Changes

2.2 Mean Time Between Changes

L'indicateur "Mean Time Between Changes" ("temps moyen entre changements") mesure le temps moyen entre chaque changement de valeur d'un observateur à valeurs réelles ou entières, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.8).

Définition de l'indicateur **Mean Time Between Changes** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que :

$$\begin{cases} \forall i \in [\![1, n[\![, t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], \exists v \in \text{Im(obs)} \text{ tel que Occurrence(obs}, v, t) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, \forall v \in \text{Im(obs)}, \text{Occurrence(obs}, v, t) = \textit{false} \end{cases}$$

Si $n > 1$, alors $X = \dfrac{1}{n-1}.\displaystyle\sum_{i=1}^{n-1}(t_{i+1} - t_i) = \dfrac{1}{n-1}.(t_n - t_1)$
sinon $X =$ non disponible

Cet indicateur peut être utilisé pour mesurer la stabilité d'une production.

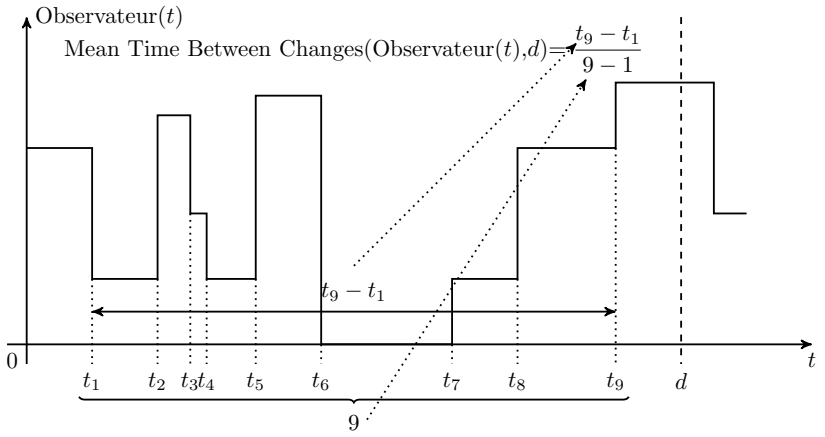

FIGURE B.8 – Exemple de mesure d'un indicateur Mean Time Between Changes

2.3 Mean Value

L'indicateur "Mean Value" ("valeur moyenne") mesure la valeur moyenne d'un observateur à valeurs réelles ou entières, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.9).

Définition de l'indicateur **Mean Value** :

$$X = \int_0^d \text{obs}(t)dt$$

Cet indicateur peut être utilisé pour mesurer la production moyenne d'une installation, ou le nombre moyen de composants défaillants.

2.4 Value

L'indicateur "Value" ("valeur") mesure la valeur d'un observateur à valeurs réelles ou entières à la date de mesure ($t = d$) (figure B.10).

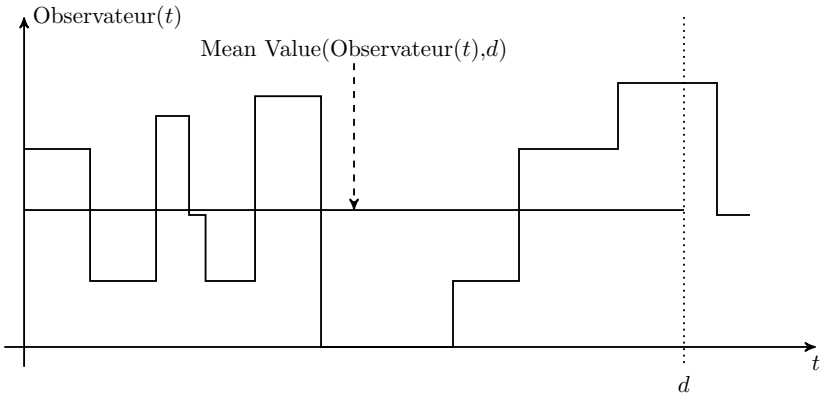

FIGURE B.9 – Exemple de mesure d'un indicateur Mean Value

Définition de l'indicateur **Value** :

$$X = obs(d)$$

Cet indicateur peut être utilisé pour mesurer la distribution des états d'un système modélisé, au temps de mission.

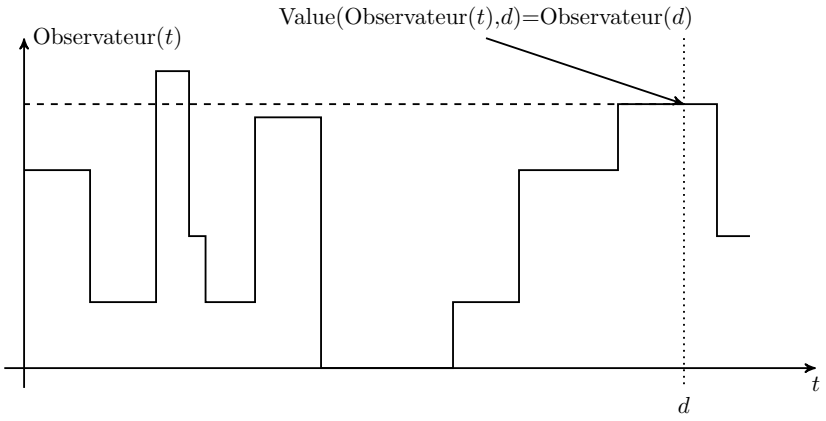

FIGURE B.10 – Exemple de mesure d'un indicateur Value

3 Indicateurs d'observateurs d'évènements

3.1 Firings

L'indicateur "Firings" ("tirs") mesure le nombre de survenances de l'évènement observé, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.11).

Définition de l'indicateur **Firings** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que :

$$\begin{cases} \forall i \in [\![1, n[\![, t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], \text{obs}(t_i) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, \forall v \in \text{Im}(\text{obs}), \text{obs}(t_i) = \textit{false} \end{cases}$$

Alors $X = n$

Cet indicateur peut être utilisé pour mesurer la fréquence d'occurrence d'un comportement modélisé.

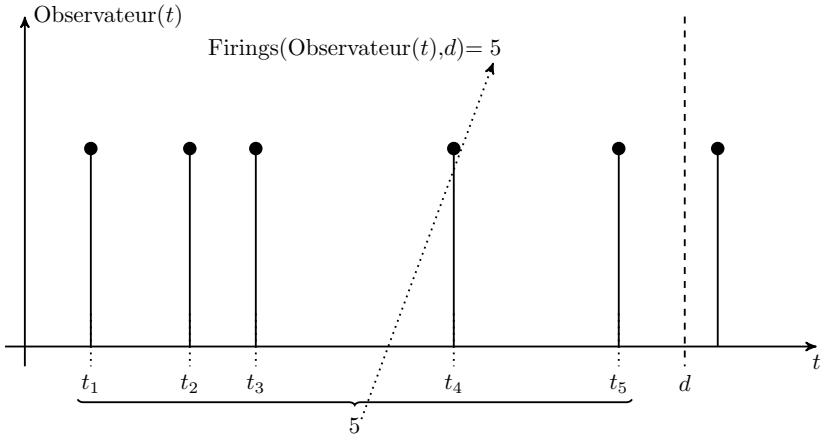

FIGURE B.11 – Exemple de mesure d'un indicateur Firings

3.2 First Firing Date

L'indicateur "First Firing Date" ("date de premier tir") mesure la date de première survenance de l'évènement observé, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.12).

Définition de l'indicateur **First Firing Date** :
Si $\exists t \in [\![0, d]\!]$, tel que :

$$\begin{cases} \text{obs}(t) = \textit{true} \\ \forall t' \in [\![0, t[\![, \text{obs}(t') = \textit{false} \end{cases}$$

Alors $X = t$, sinon $X =$ non disponible

Cet indicateur peut être utilisé pour mesurer la MTTF d'un composant.

3.3 Has Been Fired

L'indicateur "Has Been Fired" ("a été tiré") mesure si l'évènement observé est survenu au cours de l'histoire, avant la date d'observation (figure B.13).

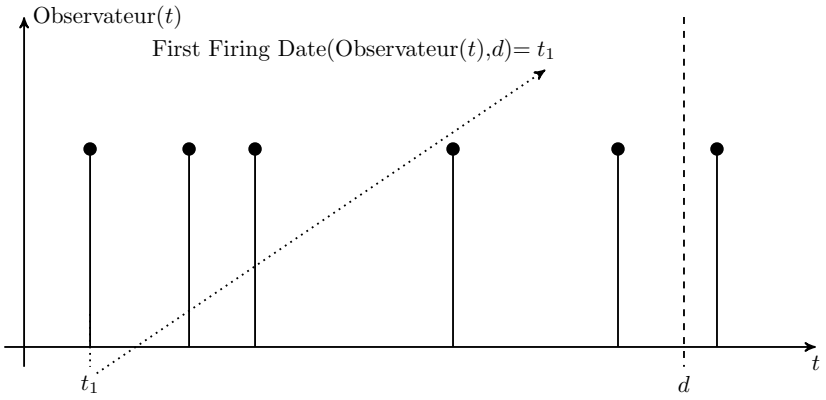

FIGURE B.12 – Exemple de mesure d'un indicateur First Firing Date

Définition de l'indicateur **Has Been Fired** :

Si $\exists t \in [0, d]$ tel que obs$(t) =$ *true*, alors $X =$ *true*, sinon $X =$ *false*

Cet indicateur peut être utilisé pour mesurer la probabilité d'occurrence d'un comportement pendant le temps de mission.

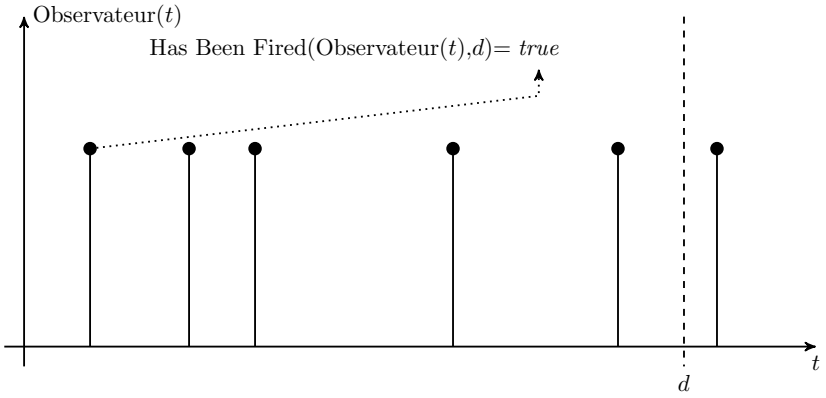

FIGURE B.13 – Exemple de mesure d'un indicateur Has Been Fired

3.4 Mean Time Between Firings

L'indicateur "Mean Time Between Firings" ("temps moyen entre tirs") mesure le temps moyen entre deux survenances de l'évènement observé, entre le début de l'histoire ($t = 0$) et la date de mesure ($t = d$) (figure B.14).

Définition de l'indicateur **Mean Time Between Firings** :
Soient $n \in \mathbb{N}$ et $(t_1, t_2, \ldots, t_n) \in [0, d]^n$ la liste des dates telles que [2] :

$$\begin{cases} \forall i \in [\![1, n-1]\!], & t_i < t_{i+1} \\ \forall i \in [\![1, n]\!], & \text{obs}(t) = \textit{true} \\ \forall t \in [0, d] \setminus \{t_1, t_2, \ldots, t_n\}, & \text{obs}(t) = \textit{false} \end{cases}$$

$$\text{Si } n > 1, \quad \text{alors } X = \tfrac{1}{n-1}.\sum_{i=1}^{n-1}(t_{i+1} - t_i) = \frac{1}{n-1}.(t_n - t_1)$$
$$\text{sinon } X = \text{non disponible}$$

Cet indicateur peut être utilisé pour mesurer une MTBF d'un composant.

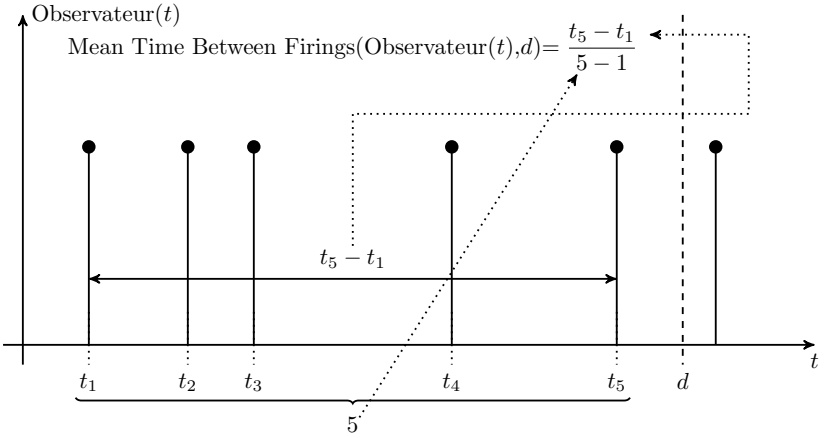

FIGURE B.14 – Exemple de mesure d'un indicateur Mean Time Between Firings

2. Il s'agit de la liste ordonnée et complète des dates de survenance de l'évènement observé.

Annexes C

Cas d'étude pour kit d'évaluation

L'évaluation du bon fonctionnement et de la performance d'un simulateur stochastique d'automates à états stochastiques et temporisés nécessite des cas d'étude. Ceux-ci doivent solliciter chacun différents mécanismes de l'outil de simulation stochastique, et leurs résultats doivent être vérifiables à l'aide d'autres outils.

1 Cas d'étude d'architectures série-parallèle

Les cas d'étude suivants partagent une même architecture : un assemblage de composants série-parallèle (figure C.1). Ils sont inspirés des modèles bloc-diagramme, répandus dans les analyses de sûreté de fonctionnement.

Un flux booléen part d'un côté de l'assemblage, et traverse les ensembles de composants jusqu'à la fin de l'assemblage si au moins un chemin existe. Un chemin existe s'il existe au moins un composant en état de bon fonctionnement dans chaque ensemble. Si aucun chemin n'existe, alors le flux ne parvient pas à la fin de l'assemblage.

Les dimensions des cas d'étude sont paramétriques, et déterminées par deux valeurs : le nombre de composants parallèles dans un ensemble, et le nombre d'ensembles (assemblés en série) dans le cas d'étude.

L'indicateur probabiliste pertinent sur ces cas d'étude est la probabilité que le flux ne parvienne pas à la fin de l'assemblage à une date spécifique.

Les cas d'étude utilisant cette architecture diffèrent sur le comportement des composants : ceux-ci sont détaillés dans la suite de cette partie.

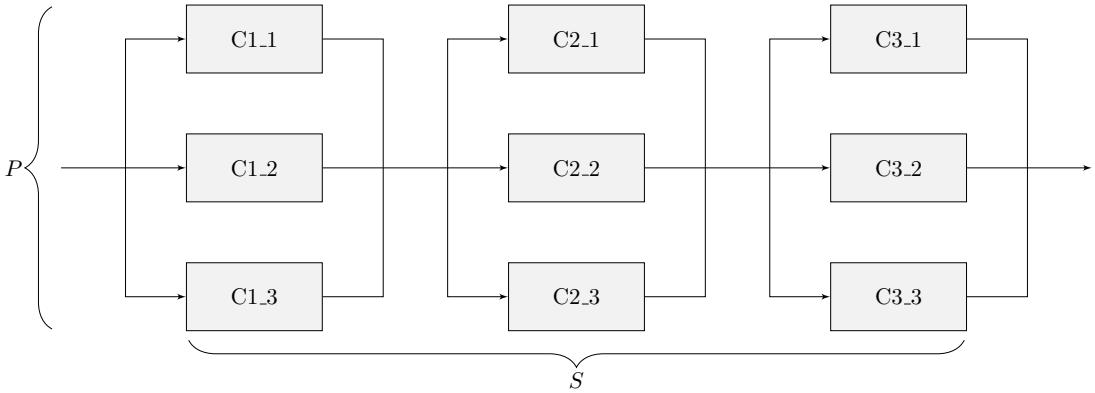

FIGURE C.1 – Architecture et dimensions paramétriques d'un cas d'étude série-parallèle

1.1 Cas d'étude série-parallèle avec défaillance

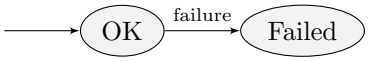

FIGURE C.2 – Graphe d'états d'un composant avec défaillance

Les composants de ce cas d'étude série-parallèle peuvent être dans deux états : bon fonctionnement, et défaillant (figure C.2). L'état initial est celui de bon fonctionnement, et ils ne peuvent que défaillir suivant une loi probabiliste librement choisie. Ce comportement est celui utilisé dans les méthodes d'analyse basées sur les arbres de défaillances.

Ce cas d'étude est simple, et ne sollicite pas outre mesure les différentes fonctions d'un simulateur stochastique évènementiel sauf avec un grand nombre de composants : un grand nombre d'évènements sont alors dans l'échéancier.

Valeur de référence par arbres de défaillances

La valeur de référence peut être obtenue par une méthode d'analyse basée sur les arbres de défaillances. Un arbre de défaillances correspondant à ce cas d'étude est une combinaison par portes OU de combinaisons par porte ET des défaillances de chaque composant d'un ensemble, pour chaque ensemble (figure C.3).

Valeur de référence par formule analytique

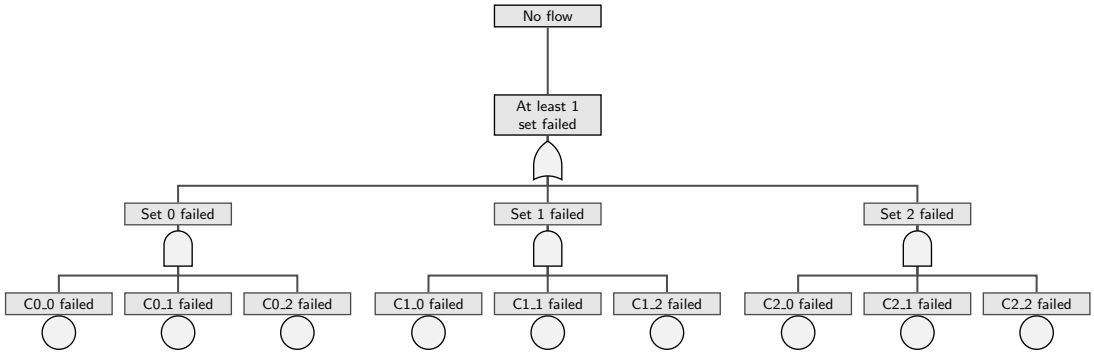

FIGURE C.3 – Arbre de défaillance modélisant un cas d'étude série-parallèle avec défaillance, de dimension série 3 et parallèle 3

Si tous les composants ont la même probabilité d'être dans l'état défaillant à la date de fin de mission, alors la valeur de référence de l'indicateur peut être calculée analytiquement.

Soit $F_c(t)$ la probabilité qu'un composant soit dans l'état défaillant à la date t. Alors la probabilité, pour un ensemble, que tous ses composants soient dans l'état défaillant est :

$$F_s(t) = (F_c(t))^P$$

La probabilité qu'un chemin existe (si aucun des ensembles n'est dans cet état), et donc que le flux parvienne au bout de l'assemblage, est :

$$F_f(t) = 1 - (1 - F_s(t))^S = 1 - (1 - (F_c(t))^P)^S$$

Pour le cas particulier de composants avec une loi de défaillance à taux λ constant (loi exponentielle) :

$$F_c(t) = 1 - e^{-\lambda.t}$$

$$F_f(t) = 1 - (1 - F_s(t))^S = 1 - (1 - (1 - e^{-\lambda.t})^P)^S$$

1.2 Cas d'étude série-parallèle avec défaillance réparable

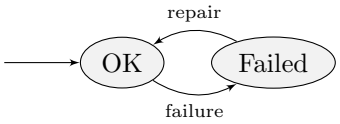

FIGURE C.4 – Graphe d'états d'un composant avec défaillance réparable

Les composants de ce cas d'étude peuvent être dans les deux mêmes états que ceux du cas d'étude précédent, mais ils peuvent aussi être réparés (figure C.4). Les composants avec ce comportement qui utilisent des lois de probabilité à taux constant (lois exponentielles) sont communément surnommés "lambda-mu" dans le contexte de la sûreté de fonctionnement.

La simulation stochastique de modèles de ce cas d'étude devrait, comme pour le cas d'étude précédent, solliciter l'échéancier par le nombre d'évènements à gérer. Ce cas d'étude va toutefois demander l'insertion d'un évènement dans l'échéancier à chaque pas de simulation, contrairement au précédent.

Valeur de référence par formule analytique

Si tous les composants ont la même probabilité d'être dans l'état défaillant ($F_c(t)$) à la date de fin de mission (t), alors la valeur de référence de l'indicateur peut être calculée analytiquement.

Dans le cas particulier de composants avec des taux constants λ et μ pour les évènements de défaillance et de réparation :

$$F_c(t) = \frac{\lambda}{\lambda + \mu}(1 - e^{-(\lambda+\mu).t})$$

Par la même réflexion que pour le cas d'étude précédent, la probabilité que le flux puisse atteindre le bout de l'assemblage à la date donnée est :

$$F_f(t) = 1 - (1 - F_s(t))^S = 1 - (1 - (F_c(t))^P)^S$$

Valeur de référence par chaînes de Markov

Si les lois de probabilités des défaillances et de réparation sont à taux constant (lois exponentielles), l'hypothèse markovienne est respectée. Une chaîne de Markov à temps continu peut alors être utilisée pour modéliser ce cas d'étude.

Un exemple d'un tel modèle (de dimension série 2 parallèle 2, tous les composants étant identiques) est donné figure C.5. Le calcul de la valeur de référence se fait en sommant les probabilités des places marquées (grisées sur la figure). La taille de la chaîne de Markov, dans le cas de composants identiques, est de $(P + 1)^S$: elle augmente très rapidement avec le nombre d'ensembles, ce qui peut être un problème pour obtenir une valeur de référence avec des cas test de grandes dimensions.

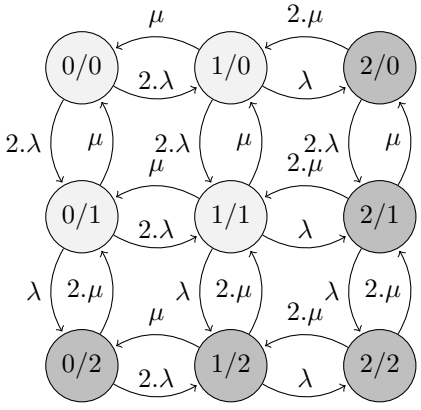

FIGURE C.5 – Chaîne de Markov à temps continu modélisant un cas d'étude série-parallèle avec défaillance réparable, de dimension série 2 et parallèle 2

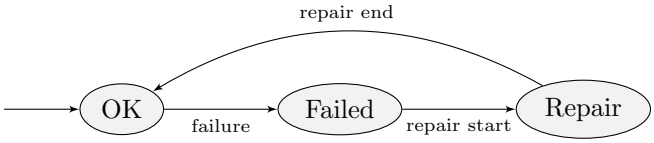

FIGURE C.6 – Graphe d'états d'un composant avec réparateurs limités

1.3 Cas d'étude série-parallèle avec réparateurs limités

Les composants de ce cas d'étude ont un comportement ressemblant à ceux du cas d'étude précédent : la différence est la limitation du nombre de composants pouvant être simultanément en phase de réparation dans tout l'assemblage. Le comportement d'un composant est représenté figure C.6 : les deux nouvelles transitions (début et fin de réparation) sont à synchroniser avec un automate comptant le nombre d'équipes de réparations actives, c'est-à-dire le nombre de composants dans l'état de réparation.

Les composants ne sont donc plus indépendants, et la simulation stochastique d'un modèle de ce cas d'étude va solliciter l'échéancier : en plus du nombre d'évènements présents dans l'échéancier, les composants devront être synchronisés par leurs transitions.

Valeur de référence par chaînes de Markov

Dans le cas où l'hypothèse markovienne est satisfaite (lois de probabilité des défaillances et de réparations sont à taux constants), une chaîne de Markov à temps continu peut être utilisée pour modéliser ce cas d'étude. En raison des trois états de chaque composant, la chaîne de Markov obtenue comporte rapidement un grand nombre de places [1].

1.4 Cas d'étude série-parallèle avec défaillance au démarrage

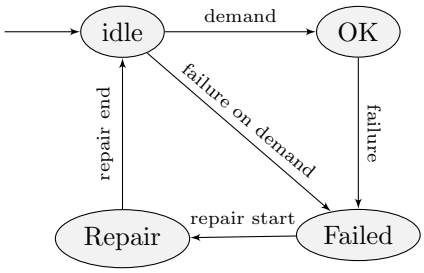

FIGURE C.7 – Graphe d'états d'un composant avec défaillance à la demande

Le comportement des composants de ce cas d'étude étend celui du cas d'étude précédent (équipes de réparation limitées) (figure C.7). Sur un ensemble de composants assemblés parallèlement, seul un nombre défini sont en fonctionnement : les autres composants en état de bon fonctionnement sont à l'arrêt. Lorsqu'un composant défaille, un composant à l'arrêt doit démarrer : ce démarrage peut soit bien se passer (le composant est alors en fonctionnement), soit causer sa défaillance immédiate (défaillance au démarrage).

1. Pour cette raison, elle ne peut pas être facilement représentée graphiquement, même pour de petites dimensions.

Ce comportement est appelé défaillance au démarrage, ou redondance froide. La modélisation de ce comportement utilise, en plus de la synchronisation par transitions pour les équipes de réparation limitées, l'utilisation de synchronisation par propagation pour démarrer les composants.

Valeur de référence par chaînes de Markov

Dans le cas où l'hypothèse markovienne est satisfaite (lois de probabilité des défaillances et de réparations sont à taux constants), une chaîne de Markov à temps continu peut être utilisée pour modéliser ce cas d'étude.

2 Cas d'étude basés sur des chaînes de Markov

Les chaînes de Markov sont un outil important dans les analyses de sûreté de fonctionnement quantitatives. C'est un formalisme de modélisation bas-niveau qui, malgré une limitation importante (l'hypothèse markovienne), permet la modélisation d'une grande diversité de systèmes, et le calcul de plusieurs indicateurs de sûreté de fonctionnement, de productivité ou de disponibilité.

Le calcul des chaînes de Markov est théoriquement réalisé par produit de matrices, représentant les équations différentielles reliant les états du modèle. Des méthodes permettent d'approcher de façon efficace les valeurs (voir [15]) sans avoir à réaliser ce calcul qui peut être très lourd.

Mais une chaîne de Markov peut être convertie en un automate de façon systématique, et peut ainsi être simulée stochastiquement. Les chaînes de Markov peuvent donc être utilisées comme cas d'étude pour évaluer des simulateurs stochastiques évènementiels, tout en permettant d'obtenir des valeurs de référence grâce aux outils dédiés au calcul de ces chaînes.

Deux cas d'études basés sur les chaînes de Markov sont proposés pour l'évaluation des simulateurs stochastiques évènementiels. Ils sont inspirés de comportements courants de sûreté de fonctionnement : bien qu'ils puissent être modélisés par des formalismes de haut niveau, on s'intéresse aux modèles obtenus par conversion d'un modèle chaîne de Markov en un automate. Ces modèles possèdent des caractéristiques permettant de solliciter des ensembles de fonctions particulières des simulateurs stochastiques évènementiels. Un inconvénient important des chaînes de Markov est l'explosion combinatoire en nombre d'états et donc en nombre de transitions. Une fois transformé en automate, l'état courant est encodé par une unique variable (en numérotant les états), tandis que les transitions dépendent toutes de cette unique variable, tant pour leurs conditions de tir que pour leurs conséquences. Les modèles obtenus vont alors impliquer, lors de leurs simulations stochastiques, un grand nombre d'évènements dans l'échéancier, et un grand nombre d'activations et de désactivations d'évènements à chaque pas par l'échéancier, ce qui va permettre l'évaluation des performances de la simulation des modèles avec ces caractéristiques.

2.1 Cas d'étude de chaîne de Markov série-parallèle

Ce cas d'étude est la version chaîne de Markov à temps continu du cas d'étude série-parallèle avec défaillance au démarrage, décrit en C.1.4. Bien que la description soit identique,

le modèle obtenu sera différent, puisqu'il sera la conversion d'une chaîne de Markov au lieu d'être une modélisation en utilisant un formalisme de haut niveau.

L'indicateur statistique utile de ce cas d'étude est la proportion de temps passé dans un état où le flux traverse l'assemblage (c'est-à-dire un chemin existe).

Valeur de référence par chaînes de Markov

Comme ce cas d'étude est une chaîne de Markov, le modèle dans ce formalisme peut être utilisé pour obtenir une valeur de référence.

2.2 Cas d'étude de chaîne de Markov série-parallèle avec panne cachée

Ce cas d'étude est similaire au précédent, en ajoutant un mode de défaillance à chaque composant : la défaillance peut être cachée, c'est à dire non détectée. Le composant ne sera pas en état de bon fonctionnement, mais ne sera pas en réparation et un composant en veille ne démarrera pas pour le remplacer. Sa réparation et son remplacement ne démarreront qu'une fois la défaillance détectée. Un test de tous les composants est réalisé à intervalle régulier : ce test est instantané, et les composants défaillants sont tous détectés.

La modélisation de ce comportement peut être réalisée par une chaîne de Markov à temps continu multiphase.

Valeur de référence par chaînes de Markov

Comme ce cas d'étude est une chaîne de Markov, le modèle dans ce formalisme peut être utilisé pour obtenir une valeur de référence.

3 Cas d'étude basés sur des arbres de défaillance

Les arbres de défaillance sont un autre outil d'analyse de sûreté de fonctionnement important. Ils sont utilisés pour obtenir des coupes minimales, qui sont les listes d'évènements de taille minimale menant à un évènement redouté. Ils peuvent être utilisés pour obtenir la probabilité d'occurrence de l'évènement redouté. Toutefois, ils souffrent de deux limitations : les évènements élémentaires doivent être indépendants, et le calcul de modèles de taille importante est coûteux.

Les arbres de défaillance combinent des évènements de base, avec des probabilités d'occurrence, en utilisant des portes logiques (OU, ET, ...) pour définir les conditions d'occurrence de l'évènement redouté. Ce comportement peut facilement être converti en un automate, et peut donc être simulé stochastiquement. L'automate obtenu possède alors une transition pour chaque évènement de base, et l'occurrence d'un évènement de base est propagée dans une grande partie du modèle, par les portes logiques auxquelles il est relié. En plus d'avoir un grand nombre d'évènements dans l'échéancier, la simulation stochastique de ce type d'automate implique une mise à jour du modèle importante, permettant d'évaluer les performances d'un outil lors de la simulation de modèles avec ces caractéristiques.

Un cas d'étude basé sur les arbres de défaillance est proposé pour l'évaluation des outils de simulation stochastique évènementielle.

3.1 Cas d'étude arbre de défaillance à trois niveaux

Les arbres de défaillance à trois niveaux ont une structure particulière : ils ont été conçus pour générer, lors de leur calcul par un outil d'analyse d'arbres de défaillances, un diagramme de décision binaire de taille exponentielle [48].

Ces arbres de défaillance sont composés :

- d'un premier niveau d'évènements de base ;
- d'un second niveau qui combine les évènements du premier niveau deux à deux à l'aide de portes logiques ET ;
- d'un troisième niveau qui combine les portes du second niveau par des portes logiques OU.

La particularité est que l'un des évènements combinés par chaque porte du second niveau est identique pour toute une branche de l'arbre (figure C.8).

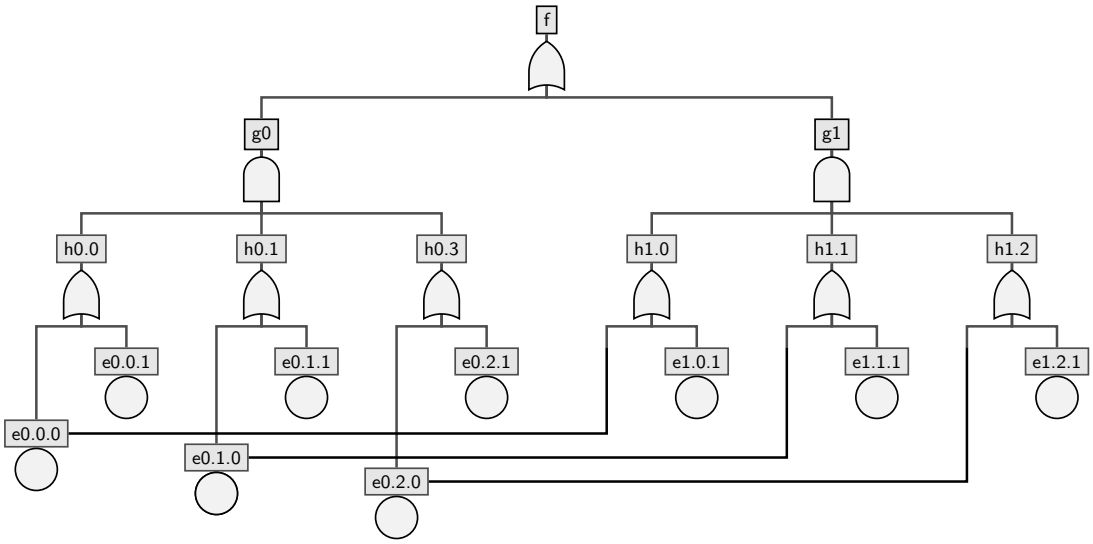

FIGURE C.8 – Structure d'un arbre de défaillance à trois niveaux

Valeur de référence par arbre de défaillance

Comme ce cas d'étude est un arbre de défaillance, le modèle dans ce formalisme peut être utilisé pour obtenir une valeur de référence.

4 Autres cas d'étude

4.1 Cas d'étude à activations séquentielles

Dans certains systèmes réels, des sous-systèmes ne sont actifs que lors d'un nombre limité de phases d'utilisation du système. Dans une modélisation de tels systèmes, le comportement des composants de ces sous-systèmes doit être synchronisé avec la phase actuelle.

Le cas d'étude à activations séquentielles est basé sur ce comportement. Son architecture est un assemblage parallèle série de composants, regroupés en lignes (figure C.9). Le principe

du flux parcourant l'assemblage est identique à celui des cas d'étude série parallèle. Les composants sont soit en fonctionnement soit en veille, et soit en état de bon fonctionnement soit défaillants. Les défaillances se font suivant une loi à choisir uniquement lorsque le composant est en fonctionnement, les réparations se font de façon immédiate lors de la mise en veille du composant. Chaque ligne est mise en fonctionnement et mise en veille régulièrement, de façon séquentielle et périodique : les lignes 1 et 2 sont en fonctionnement, puis les lignes 2 et 3... puis les lignes n et 1, en boucle.

La simulation de modèles de ce cas d'étude implique un grand nombre d'activations et de désactivations de transitions lors de la mise en fonctionnement et de la mise en veille des lignes, ce qui permet d'évaluer la performance de la simulation de modèles présentant cette caractéristique.

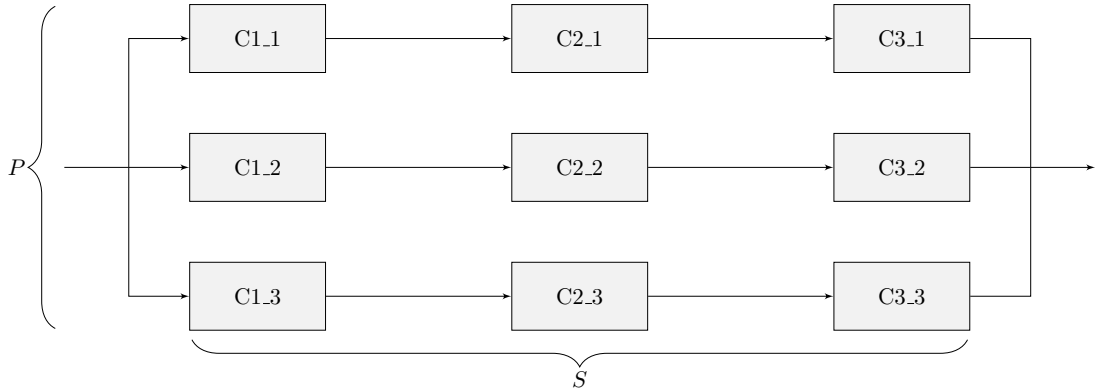

FIGURE C.9 – Architecture du cas d'étude à activations séquentielles

4.2 Cas d'étude réseau maillé

Les réseaux sont une classe importante de systèmes industriels : distribution d'énergie, réseaux d'information, ou encore systèmes fortement redondants et interconnectés. La simulation de modèles de réseaux demande la propagation d'informations à de nombreux endroits du modèle, ce qui peut être coûteux à calculer, particulièrement lorsqu'il existe plusieurs chemins différents d'un point à l'autre du réseau.

Le cas d'étude réseau maillé est constitué de plusieurs composants fortement interconnectés (figure C.10), positionnés sur une grille. Chaque composant est connecté à ses quatre voisins directs : les connexions sont bidirectionnelles. Lorsqu'un composant fonctionne correctement, l'information circule entre ses quatre connexions. Dans le cas contraire, il n'y a pas de circulation de l'information entre ses connexions. Les composants défaillent et sont réparés suivant des lois à choisir.

Le composant dans un coin est choisi comme origine de l'information : l'indicateur utile est la probabilité qu'au temps de mission, le composant situé dans le coin opposé reçoive l'information émise. Ce problème est celui de l'existence d'un chemin de composants en état de bon fonctionnement entre les deux coins de la grille.

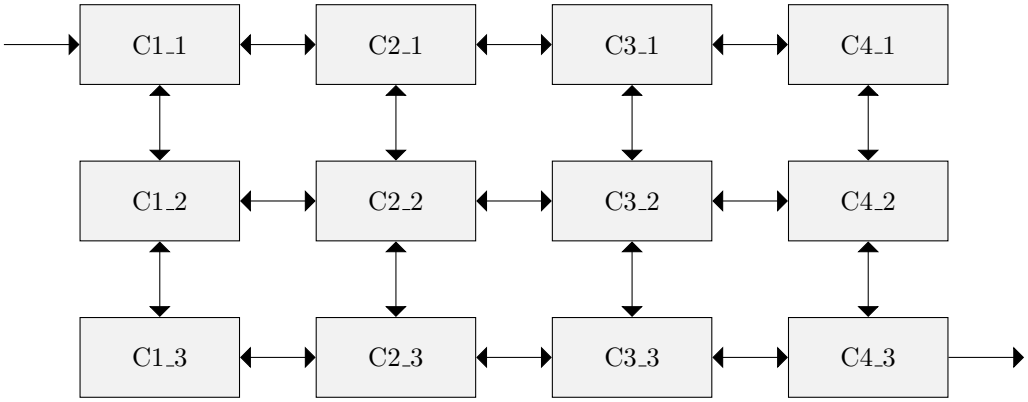

FIGURE C.10 – Architecture du cas d'étude réseau maillé

4.3 Cas d'étude jeu de la vie

La simulation stochastique d'automates cellulaires peut être utilisée pour étudier les phénomènes de propagation : pandémies, incendies ... Ces automates sont composés d'un nombre important de cellules, qui évoluent à chaque pas suivant leur propre état, et l'état des cellules voisines.

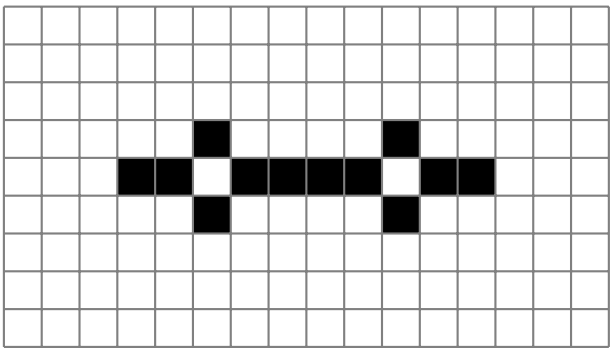

FIGURE C.11 – Schéma de départ "pentadécathlon" pour le cas d'étude jeu de la vie

Les automates du jeu de la vie [22] sont des automates cellulaires composés d'une grille de cellules carrées. Ces cellules sont soit vivantes, soit mortes. À chaque pas, le nouvel état de chaque cellule est calculé en fonction de l'état de ses 8 cellules voisines au pas précédent, et de son propre état :

- Si exactement 3 cellules voisines sont vivantes, alors la cellule devient (ou reste) vivante ;
- Si exactement 2 cellules voisines sont vivantes, alors la cellule reste dans son état au pas précédent ;
- Sinon, la cellule meurt (ou reste morte).

Plusieurs variantes de règles existent et peuvent être utilisées à la place de celles-ci : le point intéressant est leur déterminisme. Le résultat d'une simulation stochastique est connu d'avance, et sera toujours le même pour un schéma de départ donné. Plusieurs schémas de départ remarquables existent, certains étant périodiques : c'est le cas pour le schéma de départ présenté

figure C.11 qui forme la figure nommée "pentadécathlon". Cette figure a une période de 15 pas.

L'indicateur probabiliste utile peut être le temps moyen entre deux occurrences d'un schéma donné, cet indicateur impliquant de surveiller l'état de chaque cellule pour reconnaitre le schéma.

Liste des figures

Liste des tableaux

Bibliographie

[1] Marco AJMONE MARSAN, Gianni CONTE et Gianfranco BALBO. « A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems ». In : *ACM Trans. Comput. Syst.* 2.2 (mai 1984), pages 93-122. ISSN : 0734-2071. DOI : 10.1145/190.191 (cité p. 19).

[2] Steffen Nebel ANDREA DIETER et Bernd BERTSCHE. « Rare Event Simulation Speed-Up of Extended Colored Stochastic Petri Nets ». In : *Proceedings of the European Safety and Reliability Conference, ESREL 2010*. 2010, pages 1314-1319. ISBN : 978-0-415-60427-7 (cité p. 40).

[3] ARP4754A. *Guidelines for Development of Civil Aircraft and Systems*. Tome 4. 2. Warrendale, Pennsylvania, USA : Society of Automotive Engineers, oct. 2010, pages 871-879. DOI : 10.4271/2011-01-2564 (cité pp. 49, 92).

[4] ARP4761A. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*. Warrendale, Pennsylvania, USA, juil. 2004. DOI : 10.4271/arp4761 (cité p. 92).

[5] Benjamin AUPETIT, Michel BATTEUX, Antoine RAUZY et Jean-Marc ROUSSEL. « Improving performances of the AltaRica 3.0 stochastic simulator ». In : *Safety and Reliability of Complex Engineered Systems*. Zürich, Switzerland : Informa UK Limited, sept. 2015, pages 1815-1823. DOI : 10.1201/b19094-236 (cité p. 85).

[6] Benjamin AUPETIT, Michel BATTEUX, Antoine RAUZY et Jean-Marc ROUSSEL. « Vers la définition d'un kit d'évaluation pour les simulateurs stochastiques ». In : *Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement*. INIST, déc. 2016. DOI : 10.4267/2042/61811 (cité p. 119).

[7] Benjamin AUPETIT, Michel BATTEUX, Antoine RAUZY et Jean-Marc ROUSSEL. « Safety Analyzes of Mechatronics Systems : a Case Study ». In : *IFAC-PapersOnLine* 50.1 (juil. 2017), pages 11150-11155. DOI : 10.1016/j.ifacol.2017.08.1234 (cité p. 92).

[8] Michel BATTEUX, Tatiana PROSVIRNOVA et Antoine RAUZY. *AltaRica 3.0 Language Specification*. AltaRica Association. 2015 (cité pp. 22, 53, 78).

[9] Michel BATTEUX, Tatiana PROSVIRNOVA et Antoine RAUZY. *System Structure Modeling Language (S2ML)*. AltaRica Association. Déc. 2015 (cité pp. 22, 93).

[10] Michel BATTEUX, Tatiana PROSVIRNOVA et Antoine RAUZY. « AltaRica 3.0 in 10 Patterns ». In : *International Journal of Critical Computer-Based Systems* (2018) (cité p. 63).

[11] Michel BATTEUX et Antoine RAUZY. « Stochastic simulation of AltaRica 3.0 models ». In : *Proceedings of the European Safety and Reliability Conference, ESREL 2013*. Amsterdam (The Netherlands) : Informa UK Limited, sept. 2013, pages 1093-1100. DOI : 10.1201/b15938-165 (cité pp. 24, 43, 81, 96).

[12] Andrea BOBBIO. « System Modelling with Petri Nets ». In : *Systems Reliability Assessment*. Sous la direction d'A. G. COLOMBO et A. Saiz de BUSTAMANTE. Dordrecht : Springer Netherlands, 1990, pages 103-143. ISBN : 978-94-009-0649-5 (cité p. 19).

[13] Marie BOITEAU, Yves DUTUIT, Antoine RAUZY et Jean-Pierre SIGNORET. « The AltaRica Data-Flow Language in Use : Assessment of Production Availability of a MultiStates System ». In : *Reliability Engineering and System Safety* 91.7 (juil. 2006), pages 747-755 (cité p. 22).

[14] Frédéric BONIOL, Virginie WIELS, Yamine Ait AMEUR et Klaus-Dieter SCHEWE, éditeurs. *ABZ 2014 : The Landing Gear Case Study*. Tome 433. Communications in Computer and Information Science. Springer International Publishing, 2014. ISBN : 978-3-319-07511-2. DOI : 10.1007/978-3-319-07512-9 (cité pp. 92, 94, 97, 99, 101, 105, 107).

[15] Pierre-Antoine BRAMERET. « Assessment of reliability indicators from automatically generated partial Markov chains ». Thèse de doctorat. École Normale Supérieure de Cachan – ENS Cachan, juil. 2015 (cité pp. 18, 34, 150).

[16] Leonardo CALDAROLA. « Fault Tree Analysis with Multistate Components ». In : *Synthesis and Analysis Methods for Safety and Reliability Studies*. Sous la direction de G. APOSTOLAKIS, S. GARRIBBA et G. VOLTA. Boston, MA : Springer US, 1980, pages 199-248. ISBN : 978-1-4613-3036-3. DOI : 10.1007/978-1-4613-3036-3_11 (cité p. 17).

[17] Gianfranco CIARDO, Jogesh MUPPALA et Kishor TRIVEDI. « SPNP : stochastic Petri net package ». In : *Proceedings of the Third International Workshop on Petri Nets and Performance Models, PNPM89*. Déc. 1989, pages 142-151. DOI : 10.1109/PNPM.1989.68548 (cité p. 19).

[18] Carlo A. CLAROTTI. « Limitations of Minimal Cut-Set Approach in Evaluating Reliability of Systems with Repairable Components ». In : *IEEE Transactions on Reliability* R-30.4 (oct. 1981), pages 335-338. ISSN : 0018-9529. DOI : 10.1109/TR.1981.5221106 (cité p. 17).

[19] Nicolas CLAVÉ, Pierre-Joseph CACHEUX, Antoine DUTERTRE et Céline VINUESA. « GRIF-Bool : évaluation des risques à l'aide d'un outil d'analyse Booléenne multi-approches ». In : *Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement* (déc. 2016). DOI : 10.4267/2042/61803 (cité p. 21).

[20] Maïder ESTECAHANDY. « Méthodes accélérées de Monte-Carlo pour la simulation d'événements rares. Applications aux Réseaux de Petri ». 2016PAUU3008. Thèse de doctorat. 2016 (cité p. 41).

[21] J. B. FUSSELL, E. F. ABER et R. G. RAHL. « On the Quantitative Analysis of Priority-AND Failure Logic ». In : *IEEE Transactions on Reliability* R-25.5 (déc. 1976), pages 324-326. ISSN : 0018-9529. DOI : 10.1109/TR.1976.5220025 (cité p. 17).

[22] Martin GARDNER. « Mathematical Games ». In : *Scientific American* 222.5 (mai 1970), pages 124-127. ISSN : 00368733, 19467087. DOI : 10.1038/scientificamerican0570-124 (cité p. 154).

[23] Robert GARNIER. « Une méthode efficace d'accélération de la simulation des réseaux de Petri stochastiques ». 1998BOR10593. Thèse de doctorat. 1998, 155 p (cité p. 40).

[24] Paul GLASSERMAN, Philip HEIDELBERGER, Perwez SHAHABUDDIN et Tim ZAJIC. « Multilevel Splitting for Estimating Rare Event Probabilities ». In : *Oper. Res.* 47.4 (avr. 1999), pages 585-600. ISSN : 0030-364X. DOI : 10.1287/opre.47.4.585 (cité p. 40).

[25] *GRIF*. URL : http://grif-workshop.fr (cité pp. 21, 42).

[26] James A. HANLEY. « If Nothing Goes Wrong, Is Everything All Right ? » In : *JAMA* 249.13 (avr. 1983), page 1743. DOI : 10.1001/jama.1983.03330370053031 (cité p. 106).

[27] Christina IBAÑEZ-LLANO et Antoine RAUZY. « Variable Ordering Heuristics for BDD based on Minimal Cutsets ». In : *Proceeding of the PSAM'9 Conference*. Hong Kong, 2008 (cité p. 16).

[28] Marnix JOSEPH et Johann GARVELS. « The splitting method in rare event simulation ». Thèse de doctorat. Oct. 2000. ISBN : 90-365-14320 (cité p. 40).

[29] Steven JULIOUS et Mark MULLEE. « Confounding and Simpson's paradox ». In : *BMJ* 309.6967 (1994), pages 1480-1481. ISSN : 0959-8138. DOI : 10.1136/bmj.309.6967.1480 (cité p. 72).

[30] Sebastian JUNGES, Dennis GUCK, Joost-Pieter KATOEN et Mariëlle STOELINGA. « Uncovering Dynamic Fault Trees ». In : *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Juin 2016, pages 299-310. DOI : 10.1109/DSN.2016.35 (cité p. 17).

[31] Sohag KABIR, Yiannis PAPADOPOULOS, Martin WALKER, David PARKER, Jose AIZPURUA UNANUE, Jörg LAMPE et Erich RÜDE. « A Model-Based Extension to HiP-HOPS for Dynamic Fault Propagation Studies ». In : sept. 2017, pages 163-178. ISBN : 978-3-319-64118-8. DOI : 10.1007/978-3-319-64119-5_11 (cité p. 21).

[32] Sohag KABIR, Martin WALKER et Yiannis PAPADOPOULOS. « Dynamic system safety analysis in HiP-HOPS with Petri Nets and Bayesian Networks ». In : *Safety Science* 105 (fév. 2018). DOI : 10.1016/j.ssci.2018.02.001 (cité p. 21).

[33] Siddhartha Kumar KHAITAN et James D. MCCALLEY. « Design Techniques and Applications of Cyberphysical Systems : A Survey ». In : *IEEE Systems Journal* 9.2 (juin 2015), pages 350-365. DOI : 10.1109/jsyst.2014.2322503 (cité p. 92).

[34] Minh-Thang KHUU. « Génération d'un Simulateur Stochastique Guidée par la Description du Système ». In : *Proceedings of Conference on Research, Innovation and Vision for the Future in Computing and Communication Technologies*. Hanoï (Vietnam), 2004, pages 103-106 (cité p. 42).

[35] Minh-Thang KHUU. « Contribution à l'accélération de la simulation stochastique sur des modèles AltaRica Data Flow ». Dirigée par Rauzy, Antoine. Thèse de doctorat. Université de la Méditerranée (Aix-Marseille II), 2008, 1 vol. (133 p.) (cité pp. 43, 81).

[36] Donald KNUTH. *The Art of Computer Programming*. Tome 1-3. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1998 (cité p. 84).

[37] Jan KRCÁL et Pavel KRCÁL. « Scalable Analysis of Fault Trees with Dynamic Features ». In : *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Juin 2015, pages 89-100. DOI : 10.1109/DSN.2015.29 (cité p. 17).

[38] P. L'ECUYER, Christian LÉCOT et Bruno TUFFIN. *A Randomized Quasi-Monte Carlo Simulation Method for Markov Chains*. Research Report RR-5545. INRIA, 2005, page 32 (cité p. 41).

[39] Pierre L'ECUYER, Valerie DEMERS et Bruno TUFFIN. « Splitting for Rare-Event Simulation ». In : *Proceedings of the 2006 Winter Simulation Conference*. Déc. 2006, pages 137-148. DOI : 10.1109/WSC.2006.323046 (cité p. 40).

[40] Elmer E. LEWIS et Franz BÖHM. « Monte Carlo simulation of Markov unreliability models ». In : *Nuclear Engineering and Design* 77.1 (1984), pages 49-62. ISSN : 0029-5493. DOI : 10.1016/0029-5493(84)90060-8 (cité p. 40).

[41] Manish MALHOTRA et Kishor TRIVEDI. « Dependability Modeling Using Petri-Nets ». In : *Reliability, IEEE Transactions on* 44 (oct. 1995), pages 428-440. DOI : 10.1109/24.406578 (cité p. 19).

[42] Andreï A. MARKOV. « Extension of the Law of Large Numbers to Dependent Quantities (in Russian) ». In : *Izvestiya Fiziko-Matematicheskikh Obschestva Kazan University* 15 (1906), pages 135-156 (cité p. 17).

[43] *Matlab, par Mathworks*. URL : https://www.mathworks.com (cité p. 62).

[44] Robert MELCHERS. « Search-based importance sampling ». In : *Structural Safety* 9 (déc. 1990), pages 117-128. DOI : 10.1016/0167-4730(90)90003-8 (cité p. 39).

[45] Guillaume MERLE. « Modélisation algébrique des arbres de défaillance dynamiques, contribution aux analyses qualitative et quantitative ». 2010DENS0020. Thèse de doctorat. 2010, 1 vol. (208 p.) (cité p. 17).

[46] Khazen MICHAEL et Dubi ARIE. « Monte Carlo variance reduction in applications to Systems Reliability using Phase Space Splitting ». In : *Monte Carlo Methods and Applications* 10.2 (2004), pages 117-128 (cité p. 40).

[47] Marvin K. NAKAYAMA et Perwez SHAHABUDDIN. « Quick Simulation Methods For Estimating The Unreliability Of Regenerative Models Of Large, Highly Reliable Systems ». In : *Probab. Eng. Inf. Sci.* 18.3 (juil. 2004), pages 339-368. ISSN : 0269-9648. DOI : 10.1017/S0269964804183058 (cité p. 39).

[48] Macha NIKOLSKAIA et Antoine RAUZY. « Heuristics for BDD handling of sum-of-products formulae ». In : *Proceedings of the European Safety and Reliability Association Conference, ESREL'98*. 1998, pages 1459-1465 (cité p. 152).

[49] *NIST test suite for random numbers*. URL : https://csrc.nist.gov/projects/random-bit-generation/ (cité pp. 37, 121).

[50] Art B. OWEN. « Latin Supercube Sampling for Very High-dimensional Simulations ». In : *ACM Trans. Model. Comput. Simul.* 8.1 (jan. 1998), pages 71-102. ISSN : 1049-3301. DOI : 10.1145/272991.273010 (cité p. 41).

[51] Yiannis PAPADOPOULOS et John A. MCDERMID. « Hierarchically Performed Hazard Origin and Propagation Studies ». In : *Computer Safety, Reliability and Security*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1999, pages 139-152. ISBN : 978-3-540-48249-9 (cité p. 21).

[52] Carl Adam PETRI. *Kommunikation mit Automaten*. Dissertation, Schriften des IIM 2. Bonn : Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, 1962 (cité p. 19).

[53] Tatiana PROSVIRNOVA. « AltaRica 3.0 : a Model-Based approach for Safety Analyses ». Thèses. Palaiseau, France : École Polytechnique, nov. 2014 (cité pp. 22, 33, 80).

[54] Tatiana PROSVIRNOVA, Michel BATTEUX, Pierre-Antoine BRAMERET, Abraham CHERFI, Thomas FRIEDLHUBER, Jean-Marc ROUSSEL et Antoine RAUZY. « The AltaRica 3.0 Project for Model-Based Safety Assessment ». In : *Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS'2013*. Tome 46. 22. York, Great Britain : International Federation of Automatic Control, sept. 2013, pages 127-132. ISBN : 978-3-902823-49-6. DOI : 10.3182/20130904-3-uk-4041.00028 (cité p. 33).

[55] Tatiana PROSVIRNOVA, Michel BATTEUX, Ahmed MAAROUF et Antoine RAUZY. « GraphXica : a Language for Graphical Animation of models ». In : *Proceedings of the European Safety and Reliability conference, ESREL 2013*. Amsterdam, The Netherlands : CRC Press, sept. 2013. ISBN : 9781138001237 (cité p. 33).

[56] Tatiana PROSVIRNOVA et Antoine RAUZY. « Système de Transitions Gardées : formalisme pivot de modélisation pour la Sûreté de Fonctionnement ». In : *Actes du congrès LambdaMu'18 (actes électroniques)*. Tours (France) : IMdR, oct. 2012 (cité p. 31).

[57] Tatiana PROSVIRNOVA et Antoine RAUZY. « The structural constructions of AltaRica 3.0 ». In : *Actes du congrès LambdaMu'19 (actes électroniques)*. Dijon (France) : IMdR, oct. 2014 (cité pp. 22, 67).

[58] Marvin RAUSAND et Arnljot HØYLAND. *System Reliability Theory : Models, Statistical Methods, and Applications, 2nd Edition*. Hoboken, New Jersey : John Wiley et Sons Inc, 11 déc. 2003. 664 pages. ISBN : 978-0471471332 (cité pp. 70, 92).

[59] Antoine RAUZY. « New algorithms for fault trees analysis ». In : *Reliability Engineering & System Safety* 40.3 (1993), pages 203-211. ISSN : 0951-8320. DOI : 10.1016/0951-8320(93)90060-C (cité p. 16).

[60] Antoine RAUZY. « An Experimental Study on Six Algorithms to Compute Transient Solutions of Large Markov Systems ». In : *Reliability Engineering and System Safety* 86.1 (oct. 2004), pages 105-115. DOI : 10.1016/j.ress.2004.01.007 (cité p. 18).

[61] Antoine RAUZY. « BDD for Reliability Studies ». In : *Handbook of Performability Engineering*. Amsterdam, the Netherlands : Elsevier, 2008, pages 381-396. ISBN : 978-1-84800-130-5 (cité p. 16).

[62] Antoine RAUZY. « Guarded Transition Systems : a new States/Events Formalism for Reliability Studies ». In : *Proceedings of the Institution of Mechanical Engineers, Part O : Journal of Risk and Reliability* 222.4 (déc. 2008), pages 495-505. DOI : 10.1243/1748006xjrr177 (cité p. 31).

[63] Antoine RAUZY. « Anatomy of an Efficient Fault Tree Assessment Engine ». In : *Proceedings of International Joint Conference PSAM'11/ESREL'12*. Helsinki, Finland, juin 2012. ISBN : 978-162276436-5 (cité p. 16).

[64] Antoine RAUZY. « XFTA : Pour que cent arbres de défaillance fleurissent au printemps ». In : *Actes du Congrès Lambda-Mu 18*. Sous la direction de Jean-François BARBET. Oct. 2012 (cité p. 33).

[65] Antoine RAUZY et Chaire BLÉRIOT-FABRE. « Model-Based Safety Assessment : Rational and trends ». In : *2014 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics (MECATRONICS2014- Tokyo)*. Nov. 2014, pages 1-10. DOI : 10.1109/MECATRONICS.2014.7018626 (cité p. 22).

[66] Reuven Y. RUBINSTEIN. *Simulation and the Monte Carlo Method*. 1st. New York, NY, USA : John Wiley & Sons, Inc., 1981. ISBN : 0471089176 (cité p. 39).

[67] Jean-Pierre SIGNORET, Yves DUTUIT, Pierre-Joseph CACHEUX, Cyrille FOLLEAU, Stéphane COLLAS et Philippe THOMAS. « Make your Petri nets understandable : Reliability block diagrams driven Petri nets ». In : *Reliability Engineering & System Safety* 113 (2013), pages 61-75. ISSN : 0951-8320. DOI : 10.1016/j.ress.2012.12.008 (cité p. 17).

[68] Epstein STEVEN et Rauzy ANTOINE. *Open-PSA Model Exchange Format*. Rapport technique. The Open-PSA Initiative, 2007 (cité p. 33).

[69] IEC TC 56. *Fault tree analysis (FTA)*. Rapport technique IEC 61025. The International Electrotechnical Commission, 2006 (cité p. 13).

[70] IEC TC 56. *Reliability block diagrams*. Rapport technique IEC 61078. The International Electrotechnical Commission, 2016 (cité p. 15).

[71] *The OpenAltaRica Project*. URL : https://www.openaltarica.fr (cité pp. 34, 35).

[72] *The R Project for Statistical Computing*. URL : https://www.r-project.org/ (cité p. 62).

[73] Andreas UHLIG, Gerd KURZBACH, Rainer HAMANN, Yiannis PAPADOPOULOS, Martin WALKER et Bernd LÜHMANN. « Simulation model based risk and reliability analysis using extended simulationX system models to generate FMEA tables ». In : (jan. 2007), pages 225-239 (cité p. 21).

[74] Leslie G. VALIANT. « The Complexity of Enumeration and Reliability Problems ». In : *SIAM Journal on Computing* 8.3 (août 1979), pages 410-421. DOI : 10.1137/0208032 (cité p. 78).

[75] William E. VESELY, Francine F. GOLDBERG, Norman H. ROBERTS et David F. HAASL. *Fault tree handbook*. Rapport technique. Nuclear Regulatory Commission Washington dc, 1981 (cité p. 14).

[76] Halbert WHITE. « CHAPTER III - Laws of Large Numbers ». In : *Asymptotic Theory for Econometricians*. Sous la direction d'Halbert WHITE. Economic Theory, Econometrics, and Mathematical Economics. San Diego : Academic Press, 1984, pages 29-60. ISBN : 978-0-12-746650-7. DOI : 10.1016/B978-0-12-746650-7.50007-4 (cité p. 35).

[77] Alan P. WOOD. « Multistate Block Diagrams and Fault Trees ». In : *IEEE Transactions on Reliability* R-34.3 (août 1985), pages 236-240. ISSN : 0018-9529. DOI : 10.1109/TR.1985.5222131 (cité p. 17).

[78] Xinyu ZANG, Dazhi WANG, Hairong SUN et Kishor S. TRIVEDI. « A BDD-based algorithm for analysis of multistate systems with multistate components ». In : *IEEE Transactions on Computers* 52.12 (déc. 2003), pages 1608-1618. ISSN : 0018-9340. DOI : 10.1109/TC.2003.1252856 (cité p. 17).

[79] Enrico ZIO. *The Monte Carlo Simulation Method for System Reliability and Risk Analysis*. Springer Series in Reliability Engineering. Springer London, jan. 2013, 198p. ISBN : 978-1-4471-4587-5. DOI : 10.1007/978-1-4471-4588-2 (cité p. 38).

université PARIS-SACLAY

ÉCOLE DOCTORALE
Interfaces:
matériaux, systèmes, usages

Titre: Calcul d'indicateurs de sûreté de fonctionnement de modèles AltaRica 3.0 par simulation stochastique

Mots clés: AltaRica, MBSA, simulation stochastique

Résumé: Dans un contexte de conception d'un système critique complexe, les études de sûreté de fonctionnement permettent de faire des choix de solutions techniques. Le langage de modélisation choisi doit avoir un pouvoir d'expression suffisant pour représenter les différents comportements envisagés : dans ces travaux, AltaRica 3.0 est utilisé. Mais le calcul d'indicateurs de sûreté de fonctionnement sur des modèles complexes est alors difficile : une solution est l'utilisation de la simulation stochastique pour les estimer. Il est alors nécessaire pour l'analyste de décrire quels sont les indicateurs qu'il souhaite estimer, et quelles sont leurs relations avec le modèle : un ensemble de mesures, couvrant les besoins classiques en sûreté de fonctionnement, est proposé. La qualité des estimations est liée au nombre de mesures, et donc à la performance de l'outil logiciel de simulateur stochastique : des améliorations de la simulation stochastique de modèles AltaRica 3.0 ont été implémentées dans l'outil de la plateforme OpenAltaRica. L'utilisation d'outils de calcul logiciel dans un contexte de certification doit faire l'objet d'une évaluation sur leur qualité : une méthodologie d'évaluation de simulateurs stochastiques de sûreté de fonctionnement, non limitée à AltaRica 3.0, est présentée. Enfin, une étude de cas d'un système mécatronique complexe permet de présenter les possibilités de la simulation stochastique et du langage AltaRica 3.0 pour une étude de sûreté de fonctionnement de cette classe de systèmes.

Title: Assessment of reliability indicators of AltaRica 3.0 models by stochastic simulation

Keywords: AltaRica, MBSA, stochastic simulation

Abstract: Safety assessment of a critical and complex system allows choices of technical solutions. The chosen modelling language for those assessments must have sufficient power of expression to represent the different behaviours envisaged: AltaRica 3.0 is here used. But computation of dependability indicators on complex models is then difficult: stochastic simulation is a solution, but only allow to estimate values. It is then necessary for the analyst to describe which indicators he wishes to estimate, and what their relations with the model are: a set of measures, covering the conventional needs in dependability, is proposed. The estimation quality is related to the number of measurements, and therefore to the performances of the stochastic simulator software tool: improvements of stochastic simulation of AltaRica 3.0 models have been implemented in the tool of the OpenAltaRica platform. The use of software computation tools in a certification context must be evaluated on their quality: a stochastic simulator reliability evaluation methodology, not limited to AltaRica 3.0, is presented. Finally, a case study of a complex mechatronic system presents the possibilities of stochastic simulation and AltaRica 3.0 for a safety study of this class of systems.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France