Model-Based Safety Assessment : Comment renforcer la confiance dans les modèles ?

Milcent Frédéric, Batteux Michel, Tatiana Prosvirnova, Xavier de Bossoreille

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Model-Based Safety Assessment : Comment renforcer la confiance dans les modèles ?

Model-Based Safety Assessment : How to increase trust in models ?

MILCENT Frédéric
Naval Group / Equipements Navals
Ruelle Sur Touvre
frederic.milcent@naval-group.com

BATTEUX Michel
Institut de Recherche Technologique SystemX
Palaiseau
michel.batteux@irt-systemx.fr

DE BOSSOREILLE Xavier
Airbus Protect
Blagnac
xavier.debossoreille@airbus.com

PROSVIRNOVA Tatiana
DTIS / Université de Toulouse
Toulouse
tatiana.prosvirnova@onera.fr

Résumé — Les méthodes d'analyse de Sûreté de Fonctionnement de systèmes complexes de type Model-Based Safety Assessment (MBSA) existent depuis de nombreuses années et leurs avantages sont maintenant bien connus. Toutefois, les modélisateurs sont souvent confrontés à des questions concernant la représentativité du modèle et la pertinence des simulations réalisées.

Le but de cet article est d'apporter quelques éléments de réponse à ces questions en proposant un cadre et des arguments aux modélisateurs pour s'assurer de la bonne mise en œuvre de la démarche, garantir un niveau de confiance satisfaisant et ainsi prouver la validité de leurs modèles.

Ces premiers éléments essentiellement basés sur les expériences respectives des auteurs, notamment sur le langage AltaRica, pourraient être enrichis par la mise en place d'échanges rassemblant utilisateurs et experts du MBSA, par exemple, à travers un projet d'intérêt général sous l'égide de l'IMdR.

Mots-clefs — MBSA, modèle, processus, Sécurité, Sûreté de Fonctionnement

Abstract— Model-Based Safety Assessment (MBSA) methods for analyzing the safety and the dependability of complex systems existed for many years and their advantages are now well known. However, modelers are often confronted with questions concerning the representativeness of their models and the relevance of the simulations carried out.

The purpose of this article is to provide some answers to these questions by proposing a framework and some arguments for modelers to ensure the proper implementation of the approach, to guarantee a satisfactory level of trust and thus to prove the validity of their models.

These first elements, essentially based on the respective experiences of the authors, in particular on the AltaRica language, could be enriched by the establishment of a working group bringing together more MBSA users and experts, for example, through a project of general interest under the aegis of the IMdR.

Keywords — MBSA, Model, Process, Safety, Dependability

I. OBJECTIF / CONTEXTE[1]

La complexité grandissante des systèmes nécessite de plus en plus de recourir à des approches Model-Based (MBSE, MBD…). Dans le domaine de la Sûreté de Fonctionnement, des solutions de type Model-Based Safety Assessment (MBSA) ont été développées pour permettre d'évaluer au mieux la sûreté et la disponibilité des systèmes complexes et d'optimiser leur maintenance. La Figure 1 présente quelques exemples d'approches MBSA. Cette publication s'appuiera sur l'expérience de ses auteurs qui utilisent majoritairement le langage AltaRica. Néanmoins, la problématique soulevée reste commune quel que soit le langage utilisé.

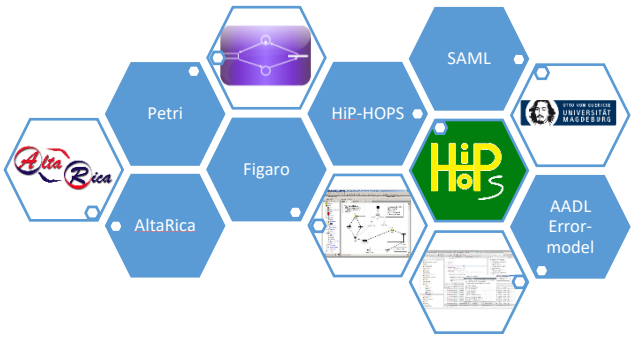


Figure 1: Exemples de langages MBSA

La rupture méthodologique liée à ces solutions et la forte expressivité des langages associés peuvent parfois entraîner des questions concernant la représentativité du modèle et l'exactitude des simulations réalisées. Ces doutes prennent souvent leur source dans une méconnaissance de la méthode ou des langages. Toutefois, cela révèle un enjeu majeur : comment apporter la preuve de la validité des résultats obtenus avec une approche MBSA ?

II. METHODOLOGIE

Le but de cet article est d'apporter quelques éléments de réponses aux questions soulevées en proposant un cadre et des arguments aux modélisateurs pour s'assurer de la bonne mise en œuvre de la démarche, garantir un niveau de confiance satisfaisant et ainsi prouver la validité de leurs modèles.

Pour cela, un processus simple est proposé dans cet article. Celui-ci est inspiré de la norme ISO / IEC / IEEE DIS 24641[7] (en cours de développement) qui propose un équivalent pour le Model-Based System and Software Engineering (MBSSE). Il repose sur 3 sous-processus (voir Figure 2) :

- Plan MBSA (Préparer),
- Perform MBSA (Réaliser)
- Support MBSA (Soutenir)

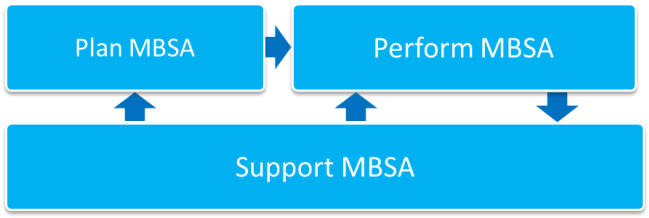


Figure 2 : Processus MBSA global

Plusieurs solutions sont apportées et certaines règles sont préconisées pour aider le modélisateur dans l'application de cette démarche.

III. PLAN MBSA

Avant de réaliser une étude MBSA, il est fortement recommandé de préparer cette étude et en premier lieu de bien vérifier que ce type d'approche est nécessaire. En effet, réaliser un modèle est trop souvent perçu comme un objectif plutôt qu'un moyen permettant de répondre à un besoin.

L'approche MBSA est particulièrement adaptée aux systèmes complexes, de grande taille ou avec un fort aspect dynamique (reconfigurations, dépendances multiples…).

Si le système est simple, de taille limitée, avec peu d'interactions et sans reconfiguration, il vaudra mieux s'orienter vers des méthodes plus classiques (Arbres de Défaillances, Diagrammes de Fiabilité…).

A. *Définir les objectifs*

La première étape consiste à définir les objectifs de modélisation comme :

- les exigences du système,
- les événements redoutés,
- les cibles de fiabilité ou de sécurité,
- les performances à observer,
- …

Ces objectifs doivent être clairement définis avant de débuter la modélisation. En cas de modification significative des objectifs en cours de modélisation, le travail déjà effectué pourrait être complètement remis en cause.

B. *Planifier le développement*

La deuxième étape consiste à planifier le développement du modèle en définissant les règles, la syntaxe et la sémantique, en établissant le périmètre, en identifiant les bases de données ou modèles existants ou les comportements déjà modélisés et validés (pour optimiser le travail de modélisation).

C. *Données d'entrée*

La troisième étape consiste à collecter toutes les données d'entrées nécessaires telles que la nomenclature, les plans, les données de fiabilité, le système de soutien logistique, l'ingénierie système.

Le couplage du modèle MBSA à un modèle MBSE (Model Based System Engineering) existant peut constituer une source de données d'entrée très profitable. Malheureusement, la réalisation d'un modèle MBSE n'est pas systématique et celui-ci fait souvent défaut.

D. *Couplage avec d'autres approches Model-Based*

Le couplage MBSE / MBSA est un moyen efficace pour s'assurer que le modèle est représentatif du système conçu.

La perspective d'un modèle unique traitant de toutes les spécialités (Ingénierie Système, Sûreté de Fonctionnement…) semble utopique :

- Il y a trop de points de vue et d'objectifs différents
- Les modélisateurs doivent disposer de toutes les compétences pour gérer toutes les spécialités pour être en mesure de réaliser un tel modèle
- Certaines normes ou pratiques imposent également une indépendance entre conception et sécurité, dans ce cas, un modèle unique ne peut pas être utilisé

Le couplage recherché est donc plutôt basé sur l'ingénierie concourante et la coexistence des 2 modèles MBSE / MBSA avec une comparaison et une synchronisation des informations partagées (structure, noms, fonctions, comportements, flux…) à intervalles réguliers.

La synchronisation des 2 modèles peut s'avérer complexe car très dépendante des types de modèles MBSE et MBSA (langages, logiciels…) et car chaque modèle génère un

nombre conséquent de données et d'informations qui sont inutiles pour l'autre modèle. Dans le modèle MBSA, des informations indispensables seront intégrées mais ne seront pas reportées dans le modèle MBSE telles que les données de fiabilité, les comportements dysfonctionnels, les données relatives au système de soutien (intervalles de maintenance, temps de réparation…), les causes communes / modes communs, …

La synchronisation est facilitée par l'utilisation de méthodes structurées permettant de trouver les informations recherchées à un emplacement défini et présentées de façon standard comme le propose, par exemple, la méthode ARCADIA supportée par le logiciel Capella (voir Figure 3).

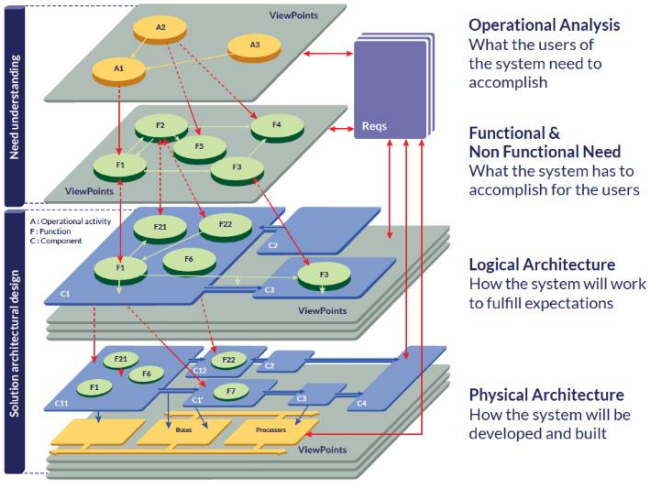


Figure 3 : Structure des modèles MBSE (méthode ARCADIA)

De nombreux projets visant à lier les approches Ingénierie Système et Sûreté de Fonctionnement ont été menés depuis les années 2000. Pour le moment, il semble qu'aucune solution idéale ne se dégage pour réaliser le couplage des modèles.

Plusieurs initiatives sont en cours comme le projet S2C (System & Safety Continuity) ou la plateforme SmartSync présentés succinctement dans les paragraphes suivants à titre d'exemples.

E. Initiative S2C

S2C (System & Safety Continuity) est un projet de recherche mené conjointement par l'IRT Saint Exupéry et l'IRT SystemX. Celui-ci a pour but de définir un cadre méthodologique, adapté à l'industrie aéronautique, pour faciliter les échanges entre les activités de conception de l'architecture système (SE) et les activités d'analyse safety (SA). Le projet s'intéresse plus particulièrement à la mise en cohérence du modèle d'architecture système (MBSE) avec le modèle d'analyse safety (MBSA).

Les travaux réalisés concernent la définition d'une méthodologie et d'outils associés de mise en cohérence des modèles MBSE et MBSA, et de maintien de cette cohérence tout au long du cycle de vie de ces modèles et des itérations[3]. Un guide méthodologique sur le MBSA est également produit afin de promouvoir son déploiement dans des contextes de certification aéronautique.

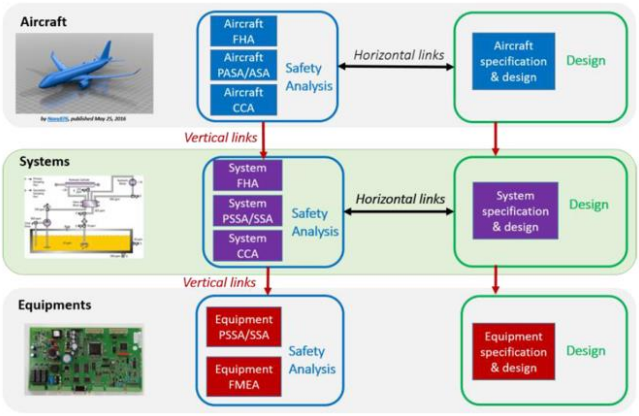


Figure 4 : Projet S2C

F. La plateforme SmartSync[11]

La synchronisation de modèles est une méthode qui permet de vérifier la cohérence entre deux modèles issus de différents domaines d'ingénierie. Elle fonctionne en 3 étapes (voir Figure 5) :

- Abstraction de modèles ;
- Comparaison de modèles ;
- Concrétisation de modèles.

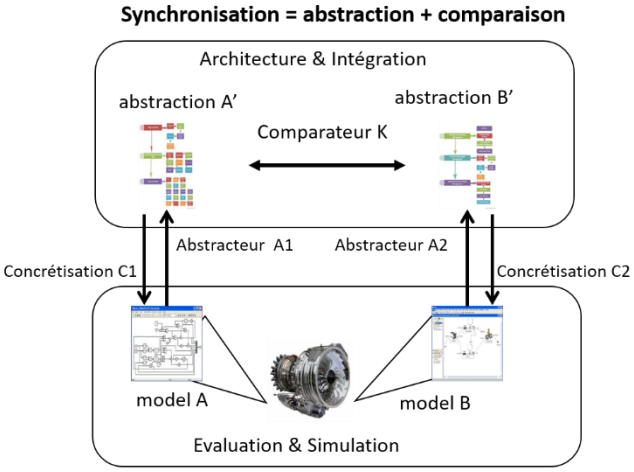


Figure 5: Synchronisation de modèles

L'abstraction de modèles consiste à extraire des informations communes, à partir de deux modèles écrits dans deux langages de modélisation différents. Cela signifie en pratique qu'il faut les transformer dans un formalisme commun aussi appelé 'pivot'.

Ensuite les deux abstractions sont comparées selon certains critères. Les critères de comparaison peuvent être multiples, par exemple ils peuvent prendre en compte la structure hiérarchique des modèles ou la topologie. Cette comparaison permet soit de détecter certains types d'incohérences entre les deux modèles de départ, soit de confirmer leur cohérence.

L'étape de concrétisation consiste à mettre à jour les modèles de départ pour corriger les incohérences trouvées lors de l'étape précédente.

La méthode de synchronisation a été implémentée dans la plateforme SmartSync, où le langage S2ML (System Structure Modelling Language) a été utilisé comme langage

pivot [5]. En effet, les langages de modélisation issus de différentes disciplines d'ingénierie sont très hétérogènes et utilisent des formalismes mathématiques très différents (exemple, équations différentielles, automates d'états finis, algèbre de Bool probabiliste, etc.). Mais ces langages partagent les constructions qui permettent de structurer les modèles, c'est-à-dire de les représenter sous forme de hiérarchies de composants connectés entre eux.

S2ML est un langage de modélisation qui regroupe les constructions structurelles issues des langages de programmation orienté-objet et orienté-prototype. Ces constructions structurelles sont utilisées par les langages de modélisation issus de différents domaines d'ingénierie. C'est pour cela que le langage S2ML est un bon candidat pour être le langage pivot de la synchronisation des modèles.

S2ML propose trois constructions de base pour structurer les modèles : des ports pour représenter, par exemple, les variables ou les évènements des modèles, des connections pour représenter les relations entre les ports (elles peuvent représenter les équations ou les transitions des modèles) et les conteneurs qui peuvent représenter les composants, les systèmes et sont composés d'autres conteneurs, de ports et de connections.

S2ML implémente également trois relations entre les conteneurs: composition, héritage et agrégation, et deux moyens de réutiliser les modèles : classe/instance et prototype/clonage.

Comme montré en Figure 6, le processus de synchronisation implémenté dans SmartSync est un processus itératif [11].

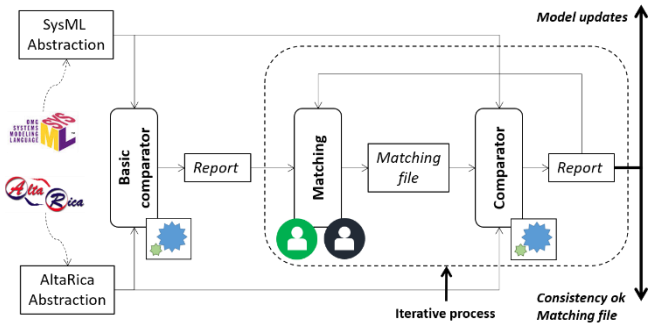


Figure 6: Principes de fonctionnement de la plateforme SmartSync

Après l'étape de transformation de modèles en S2ML, les deux modèles S2ML sont comparés (étape "Basic comparator" sur le schéma Figure 6) et un rapport est produit. Ce rapport est analysé par les ingénieurs qui ont créé les modèles de départ (étape "Matching"). Il leur permet de se mettre d'accord sur les différences entre les deux modèles et de produire un fichier de correspondance qui est ensuite utilisé pour comparer les modèles abstraits à nouveau (étape "Comparator"). A la fin, soit le fichier de correspondance liste toutes les incohérences validées ensemble par les ingénieurs, soit il existe des incohérences et les modèles de départ doivent être mis à jour pour les corriger.

La plateforme SmartSync a été utilisée pour réaliser plusieurs expérimentations de synchronisation de modèles. Sur un cas d'étude d'actionneur électromécanique pour un petit avion, les modèles d'architecture en SysML ont été mis en cohérence avec les modèles de sûreté de fonctionnement en AltaRica 3.0 [9]. Pour le même cas d'étude, les modèles d'architecture système en SysML ont été mis en cohérence avec les modèles de simulation multi-physique en Modelica [10]. Sur un cas d'étude d'un système électrique, les modèles d'architecture en Capella ont été synchronisés avec les modèles de sûreté de fonctionnement en AltaRica 3.0 [11].

IV. SUPPORT MBSA

Le sous-processus "Support MBSA" rassemble tous les moyens qui peuvent être mis à la disposition des modélisateurs pour les aider à planifier et à réaliser des modèles.

Trois éléments sont identifiés dans ce sous-processus :

- La gestion des compétences
- Base(s) de données / modèles / comportements existants
- Les outils et les moyens

A. Gestion des compétences

Les approches MBSA existent depuis de nombreuses années mais elles sont encore peu ancrées dans les pratiques des industriels. C'est pourquoi la formation est indispensable pour acquérir, développer et entretenir les compétences des modélisateurs.

La mise en place d'un processus interne de qualification des modélisateurs sanctionné par une habilitation MBSA peut également participer à la démonstration du niveau de qualité du modèle réalisé.

Les actions de sensibilisation à la démarche sont également de bons vecteurs de communication pour améliorer la connaissance des interlocuteurs des modélisateurs (concepteurs, clients…), démocratiser la pratique et faciliter les échanges.

B. Base(s) de données / modèles / comportements existants

Constituer une base de données collectant les modèles ou les comportements modélisés existants peut fournir une source d'information précieuse pour créer de futurs modèles.

Collaborer avec d'autres modélisateurs et échanger des modèles sont également des pratiques qui permettent d'améliorer les connaissances des modélisateurs et d'optimiser les modèles.

C. Outils / Moyens

Le choix des logiciels et des moteurs de calcul est également primordial dans l'obtention de résultats valides. Il s'agit de sélectionner des logiciels qui sont appropriés au besoin et qui sont adaptés aux compétences des modélisateurs et aux interfaces éventuelles avec les logiciels de MBSE.

Quelques logiciels permettent de supporter l'approche MBSA : Simfianeo, OpenAltaRica Platform, RS Model Builder, CECILIA, GRIF…

Pour juger des performances d'un logiciel, des cas test peuvent être menés pour mettre à l'épreuve les moteurs de calcul et garantir que les résultats obtenus sont pertinents.

Les moteurs de calculs demandent parfois des capacités importantes aux matériels informatiques sur lesquels ils sont utilisés. Sur des postes de travail de type bureautique cela occasionne souvent des temps de calcul importants du fait de la saturation des processeurs ou de la mémoire notamment quand les probabilités recherchées sont faibles. Dans ce cas, l'accès à de moyens de calcul plus performants peut être nécessaire.

D. Simfianeo[2]

SimfiaNeo est un logiciel développé par Airbus Protect pour outiller les analyses de Sûreté de Fonctionnement, en s'appuyant sur le langage AltaRica, plus particulièrement sa version dite "DataFlow". L'application se présente comme un modeleur graphique, permettant une vision proche des modèles de type MBSE tels que présentés dans Capella ou Cameo.

Les briques de plus bas niveau portent l'essentiel du comportement via AltaRica. Ceci favorise la modélisation locale élément par élément (en général équipement par équipement et/ou fonction par fonction), puis la reconstruction du système en connectant ces éléments entre eux, comme un jeu de briques.

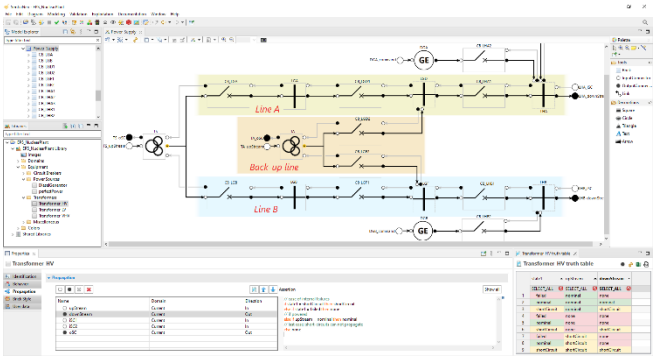


Figure 7 : Vue de l'interface de SimfiaNeo

Le logiciel inclut différents outils pour aider à la vérification du modèle. Ceux-ci incluent les mécanismes décrits au V.B, mais également d'autres outils tels que la génération de tables de vérité pour faciliter la relecture par des acteurs non-initiés au langage AltaRica.

Enfin, SimfiaNeo permet la réalisation de calculs sur le modèle : génération d'AMDE, génération de coupes ou séquences menant aux situations redoutées, et simulation de Monte-Carlo. Des mécanismes supplémentaires sont présents pour aider le modélisateur dans l'analyse des résultats produits, et la génération de rapports correspondants.

E. La plateforme OpenAltaRica

OpenAltaRica est la plateforme associée au langage AltaRica 3.0. Elle est constituée d'un ensemble d'outils logiciels permettant de concevoir et évaluer des modèles AltaRica 3.0.

AltaRica 3.0 est la troisième version du langage AltaRica. Il est basé est la combinaison du cadre mathématique GTS, pour Guarded Transition Systems [3], et du paradigme de structuration S2ML [5]. Le cadre mathématique GTS permet de rendre compte des aspects comportementaux que l'on souhaite modéliser. Ce cadre reste similaire aux autres formalismes à événements discrets (voir [12] pour une présentation générale), dans le sens où chaque fois qu'une transition est tirable, elle est planifiée et sera potentiellement tirée après un certain délai qui peut être déterministe ou stochastique.

La plateforme OpenAltaRica est constituée d'un environnement de développement intégré, nommé AltaRica Wizard [6] qui permet de concevoir et gérer les modèles AltaRica 3.0 en mode projet, ainsi que de quatre chaînes d'outils d'évaluation des modèles : un simulateur interactif, un compilateur vers les systèmes d'équations booléennes, un générateur de séquences critiques, ainsi qu'un simulateur stochastique. La Figure 8 donne une vision schématique de cette plateforme.

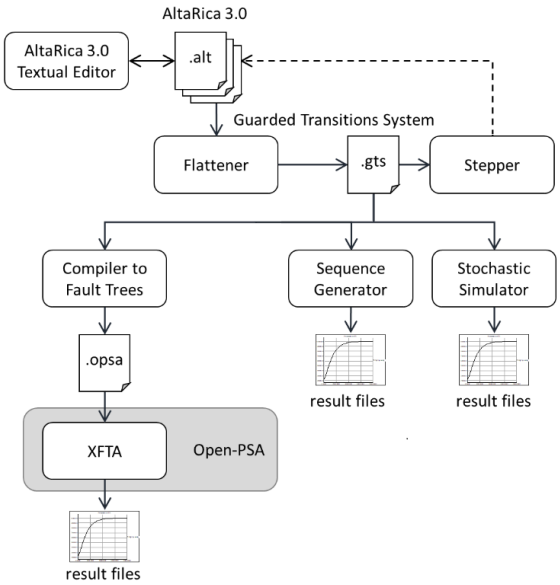


Figure 8 : Vision schématique de la plateforme OpenAltaRica

La simulation interactive rend possible l'animation des modèles en tirant des transitions, ou annulant les tirages, en affichant les valeurs des variables et des observateurs. Il s'agit d'un outil utile, et pratique, pour valider les modèles, comme peuvent le faire les outils de débogage des langages de programmation. L'algorithme du simulateur interactif AltaRica 3.0 implémente la sémantique abstraite d'exécution des GTS, c'est-à-dire qu'il tient compte des temporalités sous-jacentes des différentes transitions déterministes et stochastiques. Ainsi il y a une correspondance, au sens de la bisimulation entre les exécutions abstraites issues du simulateur interactif, et les exécutions concrètes issues du simulateur stochastique. Plus d'explications se trouvent dans [14].

Les systèmes d'équations booléennes (stochastiques) sont le cadre mathématique sous-jacents des modèles arbre de défaillance et schémas-blocs de fiabilité. La compilation vers ce type de modèles a été le succès initial de la technologie AltaRica. En effet, AltaRica permet de réduire l'écart entre les spécifications des systèmes et les modèles de sûreté de fonctionnement. Ainsi avec un même modèle AltaRica, il est possible de générer plusieurs arbres de défaillances correspondant à différents événements redoutés à évaluer. Plus d'explication se trouvent dans [13].

La génération de séquences critiques permet d'explorer partiellement les chemins du graphe d'accessibilité d'un modèle AltaRica 3.0, afin de générer les séquences critiques d'événements menant de l'état initial à un état redouté (donnant l'événement redouté évalué). Le générateur de séquences critiques AltaRica 3.0 utilise l'algorithme de simulation du simulateur interactif, permettant d'obtenir l'ensemble correct et complet des séquences critiques.

Enfin la simulation stochastique est un outil puissant pour obtenir des indicateurs de performance de modèles. Il s'agit

de générer aléatoirement plusieurs exécutions du modèle, d'observer certaines quantités durant ces exécutions au moyen d'indicateurs, puis d'en calculer des statistiques. Le simulateur stochastique AltaRica 3.0 est un outil polyvalent et efficace. Il permet en effet d'exprimer un large choix d'indicateurs (temps de séjour, nombre d'occurrences à une valeur, temps moyen entre deux valeurs, etc.). Le modèle AltaRica 3.0 n'est pas interprété mais transformé en un programme exécutable afin de rendre la génération des exécutions beaucoup plus rapide. Plus d'informations se trouvent dans [15].

V. PERFORM MBSA

Le sous-processus "Perform MBSA" (voir Figure 9) constitue le cœur du processus global car il s'agit ici de réaliser le modèle. Ce sous-processus est en partie itératif avec des phases successives de :

- Construction
- Vérification
- Simulation
- Validation

Il est également constitué :

- D'une phase de documentation réalisée en parallèle de la partie itérative durant toute la réalisation du modèle
- D'une phase de capitalisation lorsque le modèle est validé

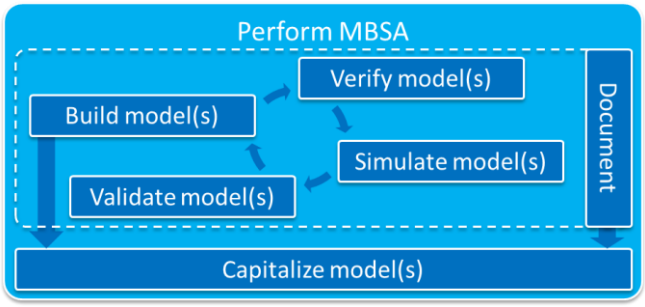
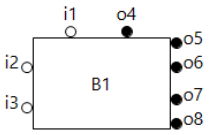


Figure 9 : Sous-processus "Perform MBSA"

A. Construction du modèle

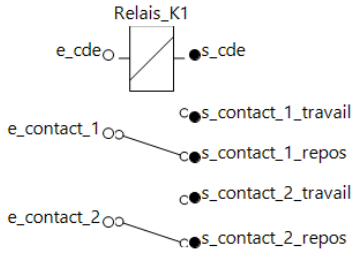
La construction des modèles est essentiellement basée sur la compréhension du système et sa traduction par le modélisateur. Afin de limiter les erreurs et de faciliter la reprise d'un modèle existant par un autre modélisateur, il convient de fixer certaines règles :

- Utilisez des noms explicites pour les blocs, les liens, les entrées/sorties, les transitions, les états, les paramètres, les domaines… La Figure 10 présente un exemple de composant modélisé (relais) sans précaution sur les nommages et la Figure 11 présente une version de ce même composant avec des noms explicites ; la seconde version permet de mieux comprendre le comportement et d'éviter les erreurs lors de la définition des transitions et assertions.
- Utiliser des classes / librairies. Cela facilitera la vérification et la validation du modèle car si la classe est validée, toutes les instances le seront aussi.
- Modéliser étape par étape (des composants au système en passant par tous les sous-ensembles).
- Utiliser, si possible, des représentations graphiques identiques ou similaires à celles de la conception. Cela permet une meilleure compréhension du modèle par les interlocuteurs du modélisateur et participe à l'amélioration de la représentativité de celui-ci par rapport au système.




```
trans
~state1 and i1 and state2 |- Event1 -> state1 := true;
state1 and ~i1 and state2 |- Event2 -> state1 := false;
state2 |- Event3 -> state2 := false, state1 := true;
state2 |- Event4 -> state2 := false, state1 := false;
~state2 |- Event5 -> state2:=true;
assert
o4 = i1;
o5 = if state1 then i2 else false;
o6 = if ~state1 then i2 else false;
o7 = if state1 then i3 else false;
o8 = if ~state1 then i3 else false;
```

Figure 10 : Exemple d'un relais modélisé sans utilisation de noms explicites




```
trans
~etat_ferme and e_cde and etat_fonct |- fermeture -> etat_ferme := true;
etat_ferme and ~e_cde and etat_fonct |- ouverture -> etat_ferme := false;
etat_fonct |- def_ferm -> etat_fonct := false, etat_ferme := true;
etat_fonct |- def_ouv -> etat_fonct := false, etat_ferme := false;
~etat_fonct |- reparation -> etat_fonct := true;

assert
s_cde = e_cde;
s_contact_1_travail = if etat_ferme then e_contact_1 else false;
s_contact_1_repos = if ~etat_ferme then e_contact_1 else false;
s_contact_2_travail = if etat_ferme then e_contact_2 else false;
s_contact_2_repos = if ~etat_ferme then e_contact_2 else false;
```

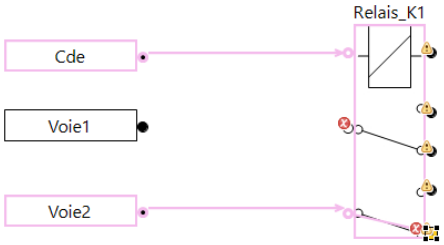
Figure 11 : Exemple d'un relais modélisé avec utilisation de noms explicites

B. Vérification du modèle

La plupart des logiciels inclue une fonctionnalité permettant de vérifier que le modèle respecte la syntaxe et la sémantique propres au langage dans lequel il est rédigé. Cette vérification (souvent paramétrable) informe le modélisateur des erreurs présentes dans le modèle ou l'avertit d'éventuels problèmes (absence de lien, présence de boucles…).

Certains logiciels indiquent visuellement l'emplacement de l'erreur ou de l'avertissement et permettent de mettre en

évidence graphiquement les dépendances des entrées / sorties pour aider le modélisateur à corriger son modèle. La Figure 12 montre le résultat d'une vérification du modèle par le logiciel et met en évidence l'absence de connexion au niveau de l'entrée *e_contact_1* ainsi que la dépendance de la sortie sélectionnée aux blocs *Cde* et *Voie2*.



	Level	Message	Rule name	Project
1	WARNING	In brick LM23_MBSA.Relais_K1: the outp...	Output connector has effects	LM23_MBSA
2	WARNING	In brick LM23_MBSA.Relais_K1: the outp...	Output connector has effects	LM23_MBSA
3	WARNING	In brick LM23_MBSA.Relais_K1: the outp...	Output connector has effects	LM23_MBSA
4	WARNING	In brick LM23_MBSA.Relais_K1: the outp...	Output connector has effects	LM23_MBSA
5	WARNING	In brick LM23_MBSA.Relais_K1: the outp...	Output connector has effects	LM23_MBSA
6	ERROR	In brick LM23_MBSA.Relais_K1: the inpu...	Input connector has a cause	LM23_MBSA

Figure 12 : Exemple de résultat de vérification par le logiciel - Simfianeo

En effet, un composant peut être conforme du point de vue de la syntaxe et de la sémantique sans que son comportement soit celui attendu par le modélisateur (inversion de valeurs, copier/coller non modifié…). La vérification par le logiciel n'est donc pas suffisante. La simulation pas-à-pas peut être utilisée pour contrôler le comportement du modèle. Il s'agit de vérifier que les sorties présentent les valeurs attendues en faisant varier les entrées [8].

La Figure 13 montre que les sorties travail du relais sont bien égales à leurs entrées respectives si la fermeture du relais est commandée.

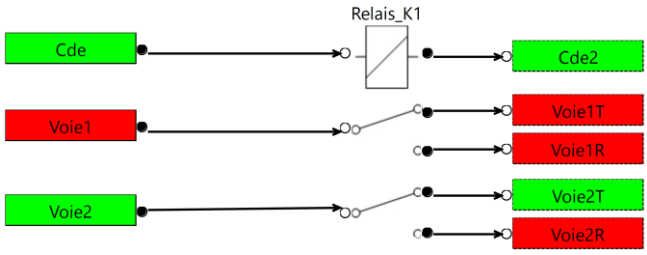


Figure 13 : Exemple de vérification par simulation pas-à-pas

Il convient de compléter ces vérifications par un échange avec les collaborateurs en charge de la conception du système afin de vérifier que la compréhension par le modélisateur est cohérente avec le comportement prévu en conception. Le couplage entre MBSE et MBSA permet de partager des concepts ce qui réduit le risque d'incohérence.

C. • Simulation

La simulation du modèle permet d'obtenir les résultats attendus mais ces résultats sont souvent sensibles aux paramètres retenus. Il convient donc d'avoir sélectionné ou fixé les paramètres appropriés :

- Type de moteur(s) de calcul ; Par exemple, certains prennent en compte les comportements dynamiques d'autres non, c'est pourquoi il faut s'assurer que le moteur choisi permet d'interpréter correctement le modèle.
- Limitation de l'ordre ou de la probabilité des séquences / coupes ; De trop fortes restrictions sur ce point peut entraîner la troncature d'éléments ayant une contribution importante dans le résultat. A l'inverse peu ou pas de restriction peut conduire à des temps de calcul importants.
- Type de résultat attendu : nombre, date, probabilité (finale ou moyenne) …
- Choix du nombre d'histoires (simulations de Monte Carlo) ; Certains moteurs sont mieux optimisés mais, en général, il est nécessaire de fixer un nombre d'histoires 100 fois supérieur à l'inverse de la probabilité recherchée (pour une probabilité de 10^{-4} , il faudrait 1 000 000 histoires).

D. Validation du modèle

De façon générale, l'obtention de résultats probabilistes égaux à 1 ou à 0 ou l'absence de séquences / coupes doivent être considéré comme des alertes. De tels résultats peuvent être dus à une erreur dans le modèle (événement redouté effectif avec les conditions initiales par exemple) ou à un mauvais paramétrage.

Analyser les résultats de simulation et les contributions peut également permettre de détecter des erreurs passées inaperçues lors de la vérification. La présence dans les séquences d'un élément a priori non concerné par la simulation peut, par exemple, être due à une inversion de connexion (élément situé à bâbord dans une chaîne tribord).

Placer des cibles intermédiaires peut également permettre de faciliter la localisation des erreurs.

Le contrôle du modèle par un autre modélisateur permet d'apporter un point de vue critique bienvenu pour la validation du modèle.

E. Documenter le modèle

Cette étape de documentation du modèle est importante pour la reprise ou la réutilisation de tout ou partie du modèle.

A l'instar des développements informatiques, il est recommandé d'ajouter des commentaires dans le "code" du modèle à chaque fois que le modélisateur le juge utile.

Les hypothèses prises doivent également être tracées et explicitées : traduction des objectif(s), abstractions, éléments modélisés / non modélisés, entrées du modèle…

Les comportements particuliers peuvent être décrits avec des éléments graphiques comme des graphes d'états présentant les différentes variables et état de l'élément et les transitions qui interviennent dans le changement d'état / variable. La Figure 14 présente le graphe d'état du relais pris en exemple dans les paragraphes précédents (voir Figure 11).

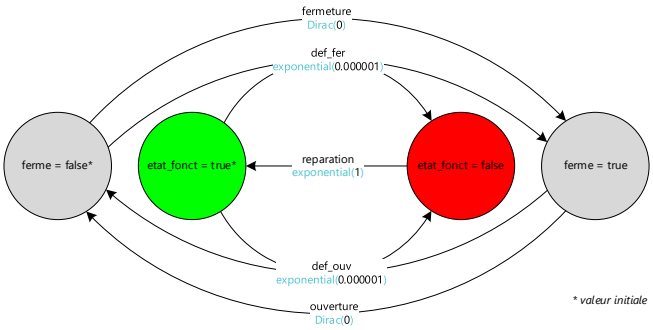


Figure 14 : Exemple de graphe d'état - Relais

Il faut bien évidemment sauvegarder les résultats des simulations mais également les paramètres de calcul retenus.

Toutes ces informations peuvent être synthétisées dans un rapport de modélisation accompagnant le modèle. Certains logiciels permettent de générer un rapport contenant les informations relatives au modèle qui peut s'avérer être une base solide pour constituer ce rapport de modélisation.

F. Capitaliser le modèle

Stockez le modèle réalisé et la documentation associée dans une base de données (ou tout autres espace équivalent) permet de pouvoir le retrouver aisément, de le partager avec d'autres modélisateurs et de constituer une base de connaissance utile pour l'analyse de nouveaux systèmes présentant des similitudes avec celui étudié.

VI. CONCLUSION

Les éléments développés dans cette publication permettent d'apporter aux modélisateurs quelques solutions pour renforcer la confiance dans leurs modèles et dans les résultats associés.

La Figure 15 reprend de façon synthétique l'ensembles des étapes du processus proposé pour la réalisation d'une étude MBSA.

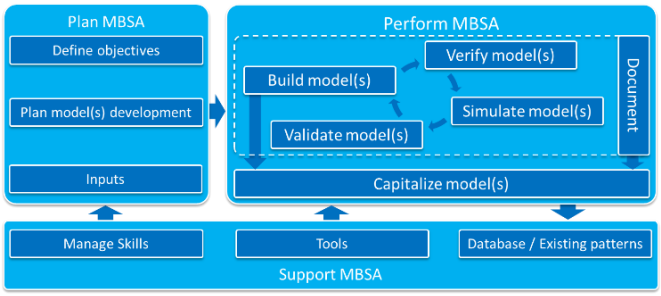


Figure 15 : Processus global de réalisation d'une étude MBSA

Ces premiers éléments essentiellement basés sur les expériences respectives des auteurs, notamment sur le langage AltaRica, gagneraient à être enrichis par la mise en place d'échanges rassemblant utilisateurs et experts du MBSA comme par exemple à travers un projet IMdR.

REFERENCES

[1] F. Milcent, M. Batteux, X. de Bossoreille, T. Prosvirnova. "MBSA: Increase trust in models", Conférence eSRel (2021), panel session, Angers, France.

[2] X. de Bossoreille, M. Machin, L. Sagaspe. "Un nouvel outil de Safety pour maîtriser la complexité des systèmes", Congrès Lambda-Mu 21 (2018), Reims, France.

[3] M. Batteux, T. Prosvirnova, A. Rauzy. "AltaRica 3.0 in 10 Modeling Patterns", International Journal of Critical Computer-Based Systems. Inderscience Publishers. Vol. 9, Num. 1–2 (2019), pp 133–165.

[4] R. Demachy, S. Guilmeau. "Structural consistency of MBSE and MBSA models using consistency links". Congrès ERTS (2022), Toulouse, France.

[5] M. Batteux, T. Prosvirnova, A. Rauzy. "From Models of Structures to Structures of Models". IEEE International Symposium on Systems Engineering (2018). Roma, Italy.

[6] M. Batteux, T. Prosvirnova, and A. Rauzy. "AltaRica Wizard : un environnement intégré de modélisation et simulation pour AltaRica 3.0". Congrès Lambda-Mu 21 (2018), Reims, France.

[7] "Systems and software engineering – Methods and tools for Model based systems and software engineering", ISO / IEC / IEEE DIS 24641

[8] D. Riera, F. Milcent, J. Parisot, E. Clement, "Modélisation dynamique en Sûreté de Fonctionnement : une avancée pour l'analyse des systèmes complexes". Congrès Lambda-Mu 18 (2012), Tours, France.

[9] M. Batteux, J.-Y. Choley, F. Mhenni, T. Prosvirnova, A. Rauzy, "Synchronization of System Architecture and Safety Models: a Proof of Concept", International Symposium on Systems Engineering (2019), Edinburgh (United Kingdom).

[10] M. Batteux, J.-Y. Choley, F. Mhenni, L. Palladino, T. Prosvirnova, al., "Synchronization of system architecture, multi-physics and safety models", CSD&M 2019, Paris, France.

[11] M. Batteux, T. Prosvirnova, A. Rauzy, "Model synchronization: a formal framework for the management of heterogeneous models". International Symposium on Model Based Safety Assessment, *IMBSA 2019*, Thessaloníki (Greece).

[12] C. G. Cassandras, S. Lafortune, "Introduction to Discrete Event Systems", Springer, New-York (USA), 2018.

[13] M. Batteux, T. Prosvirnova, A. Rauzy, "Advances in the simplification of Fault Trees automatically generated from AltaRica 3.0 models", Conférence eSRel (2018), Trondheim, (Norway), pp 907–914.

[14] M. Batteux, T. Prosvirnova, A. Rauzy. "Abstract executions of stochastic discrete event systems", International Journal of Critical Computer-Based Systems, Inderscience Publishers, Vol. 10, Num. 3 (2022), pp 202–226.

[15] B. Aupetit, M. Batteux, A. Rauzy & J-M. Roussel. "Vers la définition d'un kit d'évaluation pour les simulateurs stochastiques", Congrès Lambda-Mu 20 (2016), Saint-Malo, France.