

Intégration des analyses de sûreté de fonctionnement dans le processus de conception système.

ROBIN CRESSENT, VINCENT IDASIAK, FRÉDÉRIC KRATZ

Laboratoire PRISME / ENSI Bourges
88 boulevard Lahitolle, 18000, Bourges, FRANCE

{robin.cressent,vincent.idasiak,frederic.kratz}@ensi-bourges.fr

Résumé — Cet article s'inscrit dans des travaux visant à développer une méthode permettant d'améliorer la mise en place d'analyse de sûreté de fonctionnement au cours des phases d'ingénierie système. L'ISBM devient un concept fondamental pour la conception des systèmes et notre méthode tire pleinement avantage de cette approche et tente d'apporter les outils qui permettront l'intégration des activités de sûreté de fonctionnement aux processus de conception système. Cette méthode appelée MéDISIS propose une solution basée sur l'utilisation de SysML.

Actuellement, nous expérimentons notre méthode dans un contexte industriel : la conception du contrôleur de vol d'un véhicule hypersonique. Ce projet nous a permis d'adapter notre méthode aux besoins propres de notre partenaire industriel, pour ainsi répondre aux besoins de conceptions spécifiques des systèmes temps réels embarqués. Dans cet article, nous présentons donc l'architecture de MéDISIS et l'ensemble des outils qu'elle propose.

Mots clés — Sûreté de fonctionnement, Ingénierie Système, SysML, AADL, Simulink.

I. INTRODUCTION

Actuellement, l'Ingénierie Système dirigée par les Modèles (notée : ISBM) devient un concept prédominant pour la conception des systèmes et est principalement utilisée en Ingénierie Système (IS) [1]. L'utilisation de modèles permet d'obtenir des vues plus consistantes, cohérentes, réutilisables et expressives du système à développer, ce qui permet une réalisation et une gestion plus aisée de son processus de conception. Cependant, les activités de sûreté de fonctionnement sont généralement oubliées dans cette phase et souvent réalisées a posteriori.

Notre contribution à l'ISBM consiste à définir une méthode facilitant la mise en place d'analyse de sûreté de fonctionnement au cours d'un processus d'IS dès les premières phases de conception. Cette méthode, présentée dans plusieurs articles [2], [3], [4], s'appelle MéDISIS et s'appuie sur l'utilisation de SysML [5].

La méthode MéDISIS (figure 1) propose une approche déductive et itérative qui vise à faciliter les analyses de fiabilité et l'utilisation de divers outils et langages permettant une validation du comportement dysfonctionnel du système. MéDISIS se compose de plusieurs processus :

- Déduction du comportement dysfonctionnel et identification des exigences impactées, grâce à la génération d'AMDEC.
- Construction d'un modèle formel intégrant les comportements fonctionnels et dysfonctionnels en Altarica DataFlow.
- Analyse et quantification de l'impact des

comportements dysfonctionnels sur les contraintes temporelles en AADL.

- Passerelle vers la conception détaillée et la simulation du système avec Simulink.

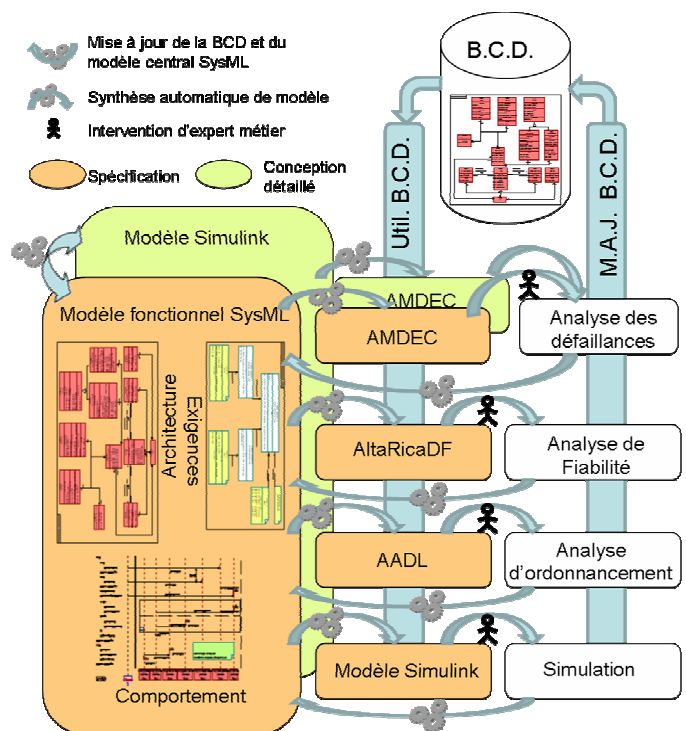


Figure 1. Résumé des différents processus qui composent MéDISIS

Actuellement, nous expérimentons notre méthode dans un contexte industriel : la conception du contrôleur de vol d'un véhicule hypersonique (projet L.E.A. avec MBDA comme partenaire industriel). Ce projet nous permet d'adapter notre méthode aux contraintes de conception industrielles et a révélé par exemple la nécessité de mettre au point un outil spécifique pour les études d'ordonnancement, afin de souligner les impacts possibles, sur la disponibilité, du non-respect des contraintes temporelles. De plus, nous avons mis en évidence plusieurs bonnes pratiques à intégrer aux activités de conceptions. Ces pratiques, appliquées à quelques éléments de modélisation SysML (Diagramme paramétrique, diagramme de bloc...), permettent de collecter et de classifier les paramètres qui influent sur la fiabilité, tels que le cycle de vie du système, le profil de mission réel du système, les conditions d'utilisation et les modes de défaillances. Pour chaque type de paramètres identifié, nous définissons l'ensemble des flux qui seront utilisés pour créer des vecteurs de tests et réaliser des

études de sûreté de fonctionnement.

Dans cet article, nous présentons le processus utilisé lors des premières phases de conception du projet LEA. Plusieurs outils et routines d'analyses ont été définis pour aider et pour optimiser la vitesse et la qualité des analyses de sûreté de fonctionnement, lors de chaque phase de conception. Ces différents processus permettent de construire un Environnement de Développement Système (EDS) complet intégrant MéDISIS et un support à l'ingénierie système. Les résultats de ces études permettent de déduire le comportement dysfonctionnel de chaque composant, qui sera thésaurisé dans la BCD (Base de données des Comportements Dysfonctionnels) pour être utilisé ultérieurement, notamment lors de la synthèse des modèles Altarica DF [6]. Un processus MéDISIS a aussi été conçu récemment pour supporter les analyses propres aux systèmes embarqués. Ce processus permet la création d'un modèle AADL [7] exploitable pour des études de contraintes temps réelles réalisées à l'aide de l'outil d'ordonnancement nommé Cheddar [8]. Le premier processus outillé, développé au sein de MéDISIS est la synthèse automatique d'AMDEC qui a été largement expliquée dans plusieurs publications telles que [3] et [4], aussi nous ne décrivons pas ce processus. Cependant, nous pouvons rappeler que lors de cette génération d'AMDEC, le mode de défaillance de chaque composant est répertorié dans la BCD constituant ainsi une base d'artefacts utilisés pour l'établissement des représentations formelle ou semi-formelles du système et de son comportement dysfonctionnel. Cela permet à terme l'introduction des modes de défaillances dans le modèle simulable du système grâce aux Diagrammes Paramétriques et aux IBD de SysML, traduits en blocks Simulink. Après avoir introduit le processus de traduction entre Simulink et SysML, nous évoquerons l'intérêt d'utiliser les blocks Simulink, générés à partir des diagrammes paramétriques et des IBD des composants identifiés dans l'AMDEC, dans le but de réaliser des études de propagation de faute.

II. UNE PASSERELLE VERS L'ANALYSE FORMELLE AVEC ALTARICA DF

Le second outil que nous avons souhaité intégrer à MéDISIS est la liaison vers des moyens formels de validation et de qualification des comportements dysfonctionnels. Il existe de nombreux langages formels, nous nous sommes concentrés sur la mise en place de passerelles vers les outils utilisant l'analyse fonctionnel d'un système et ceux dédiés aux analyses de sûreté de fonctionnement. Nous avons donc décidé de nous concentrer sur l'utilisation d'Altarica DF, largement utilisé lors des études de Sûreté de fonctionnement aidés en cela par des logiciels tels que BPA-DAS (Dassault Systems, www.3ds.com).

Cette passerelle vers Altarica DF est réalisée en deux grandes étapes : la traduction du modèle SysML vers une description fonctionnelle du système en Altarica DF, et la modélisation du point de vue dysfonctionnel grâce à la Base de données des Comportements Dysfonctionnels élaborée lors de précédentes études telle que l'AMDEC [2], [3], [4]. La première étape de traduction est primordiale pour établir un modèle consistant à partir de la description fonctionnelle du système. Comme SysML et Altarica DF partagent une approche orientée objet, un certain nombre d'éléments sont aisément traduits. Cependant, les deux langages divergent sur plusieurs aspects, par exemple, la gestion des états et des

variables de flux. Ces divergences imposent d'avoir recours à des règles de traduction plus élaborées qu'une simple équivalence. L'automatisation complète de ce processus de traduction n'est envisageable que si l'on se contraint à respecter, sous SysML, certaines règles de modélisation, telles que l'utilisation d'allocation entre les différents éléments du modèle [2].

Compléter la vue fonctionnelle grâce à la description des comportements dysfonctionnels des composants est un des apports majeurs de l'EDS MéDISIS et de la BCD qui centralise les informations pertinentes pour les études de SdF. En effet, les données récoltées par l'AMDEC peuvent être incorporées au modèle Altarica DF grâce à leur thésaurisation dans la BCD. Le modèle complet permettant les analyses de SdF est alors obtenu et peut être utilisé avec des outils logiciels présents sur le marché. Le méta-modèle de la BCD a été conçu afin d'être cohérent avec la modélisation système en SysML et pour stocker les éléments nécessaires à la construction du modèle Altarica DF final. Par conséquent, la BCD est construite en SysML et intègre des diagrammes d'états prévus pour préparer la modélisation des comportements dysfonctionnels de chaque composant.

MéDISIS a été conçue pour être un EDS évolutif visant à connecter tous les outils d'analyses nécessaires à la validation du comportement d'un système. Ainsi, elle s'est récemment développée pour intégrer la gestion de contraintes temps réels dans la conception de systèmes embarqués à fortes contraintes temporelles. Ces développements sont décrits dans les paragraphes suivants.

III. UNE PASSERELLE VERS LA CONCEPTION DE SYSTEMES EMBARQUES AVEC AADL

AADL est un langage formel et textuel apparu pour la première fois en 2004. La forme graphique ainsi que d'autres extensions du langage ont été ajoutées en 2006. La révision récente du standard [7] montre l'intérêt de la communauté à maintenir le langage à jour. L'utilisation d'AADL offre la possibilité d'analyser plus formellement les systèmes embarqués temps réels. De plus, plusieurs outils associés à AADL existent, tel que Cheddar [8] ; ils permettent d'étudier l'ordonnancement des tâches, la charge processeur et le respect des contraintes temporelles. Pour permettre ces analyses, nous avons recours à une transformation du modèle SysML vers un modèle AADL.

Le but de cette traduction est de permettre la réutilisation des connaissances contenues dans le modèle SysML, pour réaliser une analyse du comportement temps réel du système. Cependant, certaines informations, telles que les propriétés temporelles du système, sont la plupart du temps absentes d'un modèle SysML. En effet, SysML est en général utilisé à un haut niveau de conception qui ne se préoccupe pas ou peu des détails des caractéristiques temporelles des composants. Néanmoins, nous pouvons aider la réutilisation des informations contenues par le modèle SysML et faciliter la complétion des informations manquantes de façon à accélérer l'analyse et assurer la cohérence des données métiers devant être intégrées. Dans cette perspective, nous avons identifié les liens envisageables entre les deux langages.

Une fois encore, l'approche orientée objet des deux langages permet une traduction efficace des concepts architecturaux. Cependant, puisque AADL est un langage de modélisation de plus bas niveau que SysML, il utilise des types

de composants plus spécialisés. Pour classer les composants selon les 10 catégories (Memory, Processor, Bus, Device, Process, Thread, Data, Thread Group, Subprogram, System) proposées en AADL, nous devons envisager une autre source d'information, car la seule lecture du modèle SysML ne permet pas de réaliser ce classement. Plusieurs techniques de classifications sont possibles :

- Imposer une méthodologie de modélisation en SysML afin d'attribuer aux blocks SysML des stéréotypes AADL.
- Faire appel à un expert pour classer chaque composant. Utiliser un questionnaire pour faciliter l'intervention de l'expert peut être souhaitable.
- Utiliser une base de données qui indexera les correspondances entre les blocks existant en SysML et leur catégorie en AADL. Ces informations peuvent alors être basées sur le retour d'expérience de précédentes analyses.

Un problème similaire se présente pour définir toutes les propriétés qui dimensionnent le système, notamment les propriétés de temps d'exécution des tâches logicielles, les priorités de préemption, la taille des données échangées sur les bus,... Ces propriétés quantifiées sont indispensables à l'utilisation d'outils tels que Cheddar [8] ou des études de type RMA [11].

Pour résoudre ces problèmes, nous utilisons une méthode similaire à celle utilisée pour créer des AMDEC et des modèles Altarica DF : faire appel à un expert pour compléter notre modèle et maintenir une base de données de REX pour limiter le travail de l'expert pour les futurs projets. Finalement les différentes étapes suivies pour construire un modèle AADL [12] se résume ainsi :

Etape n°1. Identifier tous les blocks et parts SysML et établir la hiérarchie entre toutes ces entités, en prenant en compte les différents niveaux de conception.

Etape n°2. Cartographier les relations entre composants en utilisant les ports et les connections.

Etape n°3. Classifier par les catégories adaptées chaque composant du système (e.g. block "Mémoire Partagée" appartient à la catégorie *mémoire*).

Etape n°4. Créer le modèle structurel en AADL (l'architecture du système peut être construite textuellement et graphiquement).

Etape n°5. Renseigner l'ensemble des propriétés qui ne peuvent être déduites du modèle SysML.

Etape n°6. Créer le modèle AADL final complet.

Les étapes 1, 2, 4, 6 peuvent être automatisées, les étapes 3 et 5 nécessitent de faire appel à un expert métier aidé pour cette activité de renseignement par la base de donnée orientée composant, comme cela avait été suggéré auparavant.

Le tableau 1 présente les correspondances entre les concepts principaux des deux langages : SysML et AADL. Ce tableau est le support des premières étapes de traduction : 1, 2 et 4.

Si on applique ces étapes de traduction sommaire à notre BCD (construite et maintenu en SysML), nous obtenons une version AADL de la BCD, permettant ainsi une première représentation des comportements dysfonctionnels des

composants en AADL. Le besoin de représenter les comportements dysfonctionnels en AADL n'est pas nouveau, d'ailleurs, une extension de ce langage a été publiée par la SAE en 2006 [13] sous le nom de "error model annex". Cette extension apporte des éléments de modélisation des comportements dysfonctionnels en AADL et propose de l'aide pour générer des études de fiabilité. L'utilisation de l'"error model annex" et les bénéfices que l'on peut retirer de son utilisation sont présentés dans de nombreux travaux tels que [14].

Concepts	AADL	SysML
Composant logiciel /Implémentation	Composant logiciel /Implémentation	Block Part
Composant matériel /Implémentation	Composant matériel /Implémentation	Block Part
Relation	Binding	Block Bindings
Sous-composants	Sous-composants	Part
Connecteurs Flux	Port Connections Event, Data, Data-Event In, Out, Inout	Flow ports Value type / Block Flow Port Direction / Interface
Etats	Modes	State Diagram/state
Propriétés	Properties	Requirement Diagram, Parametric Diagram

Tableau 1. Table de correspondance des concepts entre AADL et SysML

La différence principale entre notre représentation dysfonctionnelle (issue de la traduction de la BCD) qui utilise des éléments de modélisation AADL classique et l'utilisation de l'"error model annex" est la modélisation de la propagation de défaillance. De part sa construction, notre BCD modélise la propagation de défaillance par l'intermédiaire de flux fonctionnels. Cette représentation, plus naturelle de la propagation des erreurs est un point clef de l'établissement de modèle simulable comportant des mécanismes de contrôle et de recouvrement d'erreur réaliste. Un composant doit réaliser un diagnostic de ses entrées pour détecter une défaillance ou implémenter une commande robuste afin de l'absorber. (i.e. "l'error model annex" utilise un signal dédié à la propagation de défaillance). Ceci a l'avantage d'être plus réaliste lors de la simulation un système complet aussi bien fonctionnellement que dysfonctionnellement et ainsi analyser l'impact réel, corrigé ou non, d'une défaillance sur les données de sorties. Cependant cette méthode, adaptée à la conception, est trop lourde pour permettre la génération d'arbres de défaillance ou de modèle de Markov, souvent utilisés pour les études de fiabilité et de disponibilité. De plus, les modèles d'erreurs de composants bas niveaux nécessitent des informations spécifiques aux types d'études auxquels on veut les soumettre. Les études de sûreté de fonctionnement nécessitent des informations propres à la fiabilité des systèmes : probabilité de défaillance, temps de maintenance, mécanisme de tolérance aux fautes, paramètres stochastiques du système (i.e. Occurrence des événements redoutés et leur propagation).

Finalement, l'utilisation de l'"error model annex" permettrait d'améliorer la modélisation dysfonctionnelle de nos composants dans la BCD AADL. Les outils associés à l'"error model annex" sont très utiles pour effectuer des analyses de sûreté du système modélisé en SysML. Cependant,

nous gardons également notre modélisation des erreurs, pour prendre en compte les comportements dysfonctionnels présents lors de la conception. Ceci est présenté plus en détail dans le paragraphe suivant.

IV. UNE PASSERELLE VERS LA CONCEPTION DÉTAILLÉE AVEC MATLAB/SIMULINK

Nous avons voulu, avec ce dernier processus, transposer notre méthode aux outils de conception modernes. Dans le cadre d'un partenariat, nous utilisons MéDISIS comme support à la conception d'un véhicule hypersonique, et plusieurs problèmes se sont posés à nous. Le premier est le déploiement de nos outils au sein du processus industriel de notre partenaire, et le second est le déploiement de notre méthodologie sur les outils utilisés communément par notre partenaire. Simulink, outil de conception de système largement utilisé, offre des éléments de modélisation compatibles avec SysML s'est imposé comme un choix naturel.

L'apport majeur attendu de la modélisation de notre système en Simulink est la possibilité de simuler le système pour collecter les informations concernant la propagation d'erreur, et cela très tôt dans le processus de conception. Nous allons donc maintenant décrire comment traduire, le plus efficacement possible, les éléments SysML vers Simulink. Puis nous décrirons les possibilités qu'offre un modèle Simulink intégrant des éléments de BCD en matière d'étude des comportements dysfonctionnels d'un système.

Concept	Simulink	SysML
Composants	Block	Block / Part
Relation	Line	Block Association
Sous-composants	Subsystems	Part
Connecteurs Flux	Inport / Outport Line	Flow ports Flow specification
Etats	Stateflow Diagram/states	State Diagram /States
Contraintes	Block	Parametric diagram /Constraint block
Relation entre Contraintes	Line	Parametric diagram /Connections
Exigences	Block	Requirement Diagram
Relations entre Exigences	Line	Requirement Diagram /Connections

Tableau 2. Table de correspondance des éléments de modélisation entre Simulink et SysML

Dans un premier temps, on peut aisément remarquer des correspondances entre les éléments de modélisation SysML et ceux de Simulink : les **Blocks** et les **Line** sont les entités de base d'un modèle Simulink. Un block représente un système qui peut contenir d'autres systèmes (**subsystem**). Les systèmes utilisent des ports d'entrée (**Inport**) et des ports de sorties (**Outport**). Une **line** relie deux **blocks** ensemble. On retrouve une organisation similaire au sein de multiples diagrammes de SysML.

Un **block** Simulink correspond à un *block* SysML et un **subsystem** correspondra à un des éléments d'un *internal block diagram*. Les **lines** entre les **blocks** Simulink correspondent aux *ports* interconnectés en SysML. Les flux de contrôle et les flux de données qui transitent par un **connector** Simulink seront directement associés aux *control* et *data flow* de

SysML. L'association des *standard ports* et des *flow ports* nous permet de définir la structure des **inport/outport** et les **lines** représentant les flux/interactions entre les différents **blocks** en Simulink. Au niveau de la représentation du comportement des composants, les éléments **Stateflow** en Simulink sont un sur-ensemble des *Statecharts* de SysML. Enfin, les contraintes imposées au système, modélisées à l'aide de *parametric diagram* en SysML seront représentés à l'aide de **blocks** et **lines** eux aussi.

Comme nous pouvons le voir dans le tableau 2, différents type d'éléments de modélisation SysML peuvent être transformés en un même type d'éléments en Simulink, par exemple : les **lines** en Simulink représenteront aussi bien les connections entre blocks que les flux de données modélisées différemment en SysML. En effet, la traduction de SysML vers Simulink est surjective, ce qui signifie qu'il y aura une perte d'information lors du processus de traduction, ou tout du moins une perte de précision dans la représentation du système. Dans l'autre sens, une traduction de Simulink vers SysML donnerait un modèle incomplet, puisque Simulink par nature n'offre qu'un seul point de vue, du système, associant, les vues statique, fonctionnelle et dynamique de SysML.

L'ajout de ce processus de traduction à MéDISIS (figure 1) permet d'obtenir un modèle fonctionnel en Simulink en parallèle du modèle SysML. Pour rendre la simulation de la propagation de défaillance possible, nous allons utiliser une librairie dysfonctionnelle associée à notre BCD.

À partir du modèle fonctionnel de notre système en SysML, nous avons vu comment MéDISIS participe à la rédaction d'AMDEC, et comment les résultats obtenus pendant ce processus peuvent être réutilisés pour modéliser le comportement dysfonctionnel du système en Simulink. À ce moment, nous obtenons le même niveau de modélisation que lors de la traduction vers AADL en ayant recours à l'"error model annex", mais nous sommes en plus capables d'étudier les effets physiques des modes de défaillances au niveau du système entier par simulation. Il est donc possible de commencer à envisager la spécification et la conception de mécanismes ou lois de contrôle permettant de limiter la propagation d'erreur.

En offrant la possibilité de modéliser fonctionnellement et dysfonctionnellement le système, Simulink permet d'aborder sa conception détaillée tout en évaluant sa performance du point de vue de la sûreté de fonctionnement. Le modèle permet également un meilleur dimensionnement des effets des modes de défaillance significatifs retenus à partir de l'AMDEC. Le modèle SysML du système devient le modèle pivot entre le modèle de conception du système et son analyse par les modes de défaillance et de leurs effets.

A. Mise à jour de la BCD pour la conception en Simulink

Après avoir construit la BCD à partir du modèle fonctionnel, nous obtenons une liste des modes de défaillance et de leur gravité, nous pouvons donc introduire dans le modèle de conception le comportement dysfonctionnel des composants sélectionnés en fonction de leur gravité potentielle. Pour les composants sélectionnés, l'objectif est de mieux connaître l'effet d'une défaillance ou d'évaluer la performance d'une loi de contrôle ou d'un mécanisme de protection devant être appliqué.

Premièrement, nous devons mettre à jour la BCD du projet. Comme cela a été dit, le processus de génération

d'AMDEC apporte les informations afférentes à la sûreté de fonctionnement qu'il est important de stocker dans la BCD. De même, les connaissances des comportements dysfonctionnels sont obtenues après exploitation du modèle Altarica ou AADL et seront thésaurisées dans la BCD sous la forme de diagramme SysML. Pour chaque composant sélectionné, nous pourrions donc tirer parti de ces informations pour adapter notre modèle de défaillance générique (Figure 2) au composant traité.

Ce modèle dysfonctionnel générique introduit plusieurs paramètres de configuration :

- Type de défaillance (Failure Type)
- Mode d'activation (Activation Mode)
- Loi de défaillance et de réparation (Failure and repair laws).

Concernant le type de défaillance, plusieurs choix s'offrent à nous : pas de service, service dégradé, service intermittent, et service normal.

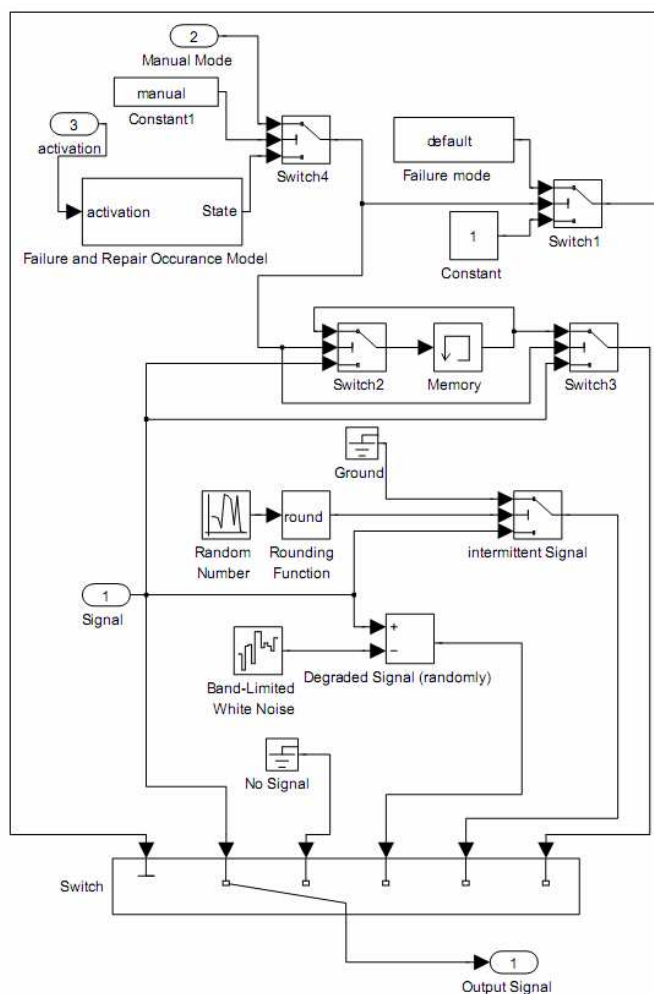


Figure 2 Modèle Simulink de simulation d'un mode de défaillance.

Le mode d'activation propose deux choix : déclenchement manuel ou déclenchement automatique en suivant des lois de défaillance et de réparation prédéterminées. Dans notre étude présentée ci-dessous, nous avons utilisé le mode manuel. Nous activons un ou plusieurs modes de défaillance selon les scénarios de tests élaborés par les études précédentes.

Pour utiliser le mode d'activation automatique, il est nécessaire de présenter d'autres paramètres de configuration :

- La loi de défaillance (loi exponentielle ou loi

uniforme),

- Le taux de défaillance (dans le cas d'une loi exponentielle),
- La possibilité ou non de réparer le composant,
- Le taux de réparation dans le cas d'une loi exponentielle. Ces paramètres sont référencés dans la BCD pour chaque composant et chaque mode d'activation d'un défaut, ils sont établis pour l'occasion ou repris d'une précédente étude.

Ce bloc de simulation de défaillance représente un mode de défaillance générique pour un composant électronique [15] (i.e. adapté aux systèmes embarqués). Pour une meilleure compréhension, nous illustrons ensuite l'intégration d'un tel bloc (Figure 3).

B. Mode de défaillance générique et modèle fonctionnel

Nous pouvons placer le **Block** Mode de Défaillance (BMdD) pour chaque signal étudié. Deux points de vue sont possibles pour le placement exact de ce **block**. Dans le cas d'une étude par injection de faute, on choisit d'insérer le BMdD en tant qu'entrée d'un autre **block** (**block** fonctionnel ou **block** composant) pour étudier l'effet de la défaillance sur ce **block**. D'un autre côté, quand on doit étudier l'impact de la propagation d'erreur à partir des résultats de l'AMDEC, on insère le BMdD à la sortie du **block** défaillant.

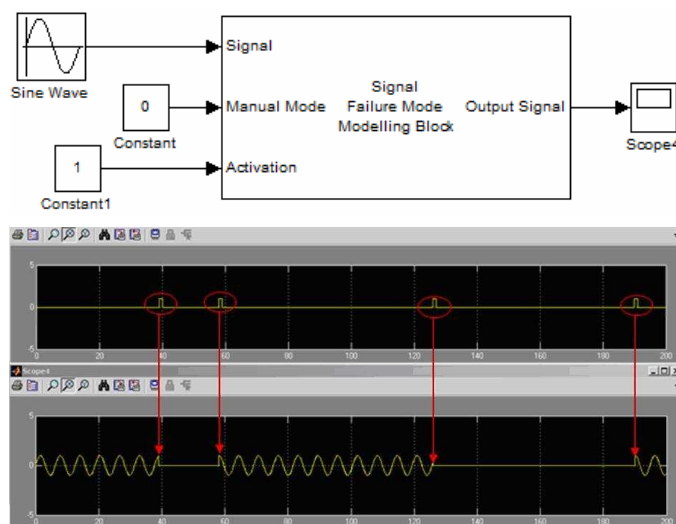


Figure 3 Exemple de simulation de défaillance.

La Figure 3 montre comment le BMdD influe sur un signal. Dans cet exemple, nous avons décidé d'utiliser le mode automatique d'activation en configurant les lois de défaillance et de réparation dans les paramètres de configuration du **block**. L'entrée « Manual Mode » est connecté à la constante 0 afin d'invalider le mode manuel. Le signal constant à 1 pour l'entrée « Activation » indique que la génération de défaillance (suivant la loi de défaillance) est possible tout au long de la simulation. En effet, l'entrée binaire « Manual Mode » peut recevoir un signal qui commandera l'occurrence des défaillances (lors du passage à 1), et l'entrée binaire « Activation » peut inhiber la possibilité d'occurrence d'une défaillance en mode automatique (pas de défaillance possible lorsque l'entrée vaut 0). Le signal sur lequel on applique le BMdD est un signal sinusoïdal. La représentation graphique des signaux (Figure 3) représente le signal d'occurrence des défaillances et réparation interne au BMdD (en haut) et le signal de sortie du BMdD (en bas). On notera que le BMdD est configuré ici pour simuler une défaillance de type : pas de

service.

V. CONCLUSION

La complexité de la conception multi-domaine et la difficulté d'intégrer la sûreté de fonctionnement tôt dans le processus de conception sont des obstacles à un meilleur processus de conception. En adoptant une approche basée sur les modèles pour concevoir des systèmes complexes, nous espérons appréhender plus efficacement les problèmes propres : à certains niveaux de modélisation imposés par les outils d'analyse, à l'échange d'informations entre les différents domaines, à la communication entre un outil spécialisé et un outil système et inclure ces processus dans la méthode MéDISIS.

La prise en compte des systèmes temps réels embarqués est facilitée par la traduction vers AADL, et grâce à la passerelle vers Simulink, nous finalisons la phase de spécification du système et nous entamons le processus de conception détaillé de notre système. Finalement, l'ensemble des processus intégré à MéDISIS permettent depuis les premières phases de spécification du système jusqu'au début de la conception détaillée, de faciliter les études de sûreté de fonctionnement en mettant à profit les connaissances mise à disposition par la modélisation fonctionnelle. Toutefois, comme nous avons pu le voir avec les systèmes embarqués qui ont nécessités l'apport de la modélisation AADL, certains domaines d'ingénierie dispose et repose sur des outils et études spécifiquement adaptés. Ces outils représentent les limites actuelles de MéDISIS et donc aussi les perspectives d'évolution de la méthode. La place de la BCD dans l'ensemble des processus est primordiale, et donc l'amélioration des performances de MéDISIS est conditionnée par l'évolution de sa BCD et notamment par un rapprochement de celle-ci aux bases de données existantes issue de normes.

VI. REFERENCES

- [1] Friedenthal, S., Moore, A. & Steiner, R., "A Practical Guide to SysML : The Systems Modeling Language", *The MK/OMG press, Elsevier*, 2008.
- [2] David, P., Idasiak, V. & Kratz, F., "Automating the synthesis of AltaRica Data-Flow models from SysML". *Proceedings of ESREL 2009*, Prague, Czech Republic, 7-10 September 2009.
- [3] David, P., Idasiak, V. & Kratz, F., "Use and improvements os SysML in reliability study". *Proceedings of the 55th Annual Reliability and Maintainability Symposium, RAMS 2009*, Fort Worth, Texas, USA, 26-29 janauary 2009.
- [4] David, P, Idasiak, V. & Kratz, F., "Reliability study of complex physical systems using SysML", *Journal of Reliability Engineering and System Safety (2009)*, Volume 95, Issue 4, April 2010, Pages 431-450..
- [5] OMG 2008. "OMG Systems Modeling Language (*OMG SysML*) V1.1., 1st November 2008.
- [6] Rauzy, A., "Mode Automata and their compilation into Fault tree". *Reliability Engineering and System Safety* 78: 1-12, 2002.
- [7] SAE, Society of Automotive Engineers, "*Architecture Analysis & Design Language. Specification V2*", January 2009.
- [8] Singhoff, F., "The Cheddar AADL Property sets (Release 2.x). *LISyC technical report*, February 2007
- [9] Price, C. & Taylor, N., "Automated multiple failure FMEA". *Reliability Engineering and System Safety* Vol. 76, pp. 1-10, 2002.
- [10] Teoh, P. & Case, K., "Failure modes and effects analysis through knowledge modelling. *Journal of Materials Processing Technology* 153-154, pp. 253-260, 2004.
- [11] Klein, M., Ralya, T., Pollak, B., Obenza, R., Harbour, M. & Harbour G. "A practitioner's Handbook for Real-Time Analysis". *Kluwer Academic Publishers*, 1993.
- [12] Cressent, R., David, P. & Idasiak, V. & Kratz, F., "Increasing Reliability of Embedded Systems in a SysML Centered MBSE Process: Application to the LEA Project", *1st M-BED workshop, during DATE 2010*, Dresden, Germany, 12 March 2010.
- [13] SAE, Society of Automotive Engineers, June 2006. "*SAE Standards: AS5506/1, Architecture Analysis & Design*."
- [14] Feiler, P. & Rugina A. "Dependability Modeling with the Architecture Analysis & Design Language (AADL)", *Carnegie Mellon Institute, CMU/SEI-2007-TN-043*, July 2007.
- [15] Niemann, H. & Stoustrup, J., "An Architecture for Fault Tolerant Controllers", *International Journal of Control*, 78(14):1091-1110, 2005.