Apports de SysML à la modélisation des systèmes complexes à fortes contraintes de sûreté de fonctionnement

Robin Cressent, Pierre David, Vincent Idasiak, Frédéric Kratz

Apports de SysML à la modélisation des systèmes complexes à fortes contraintes de sûreté de fonctionnement.

Robin CRESSENT, Pierre DAVID, Vincent IDASIAK, Frédéric KRATZ (Institut Prisme/ENSI de Bourges)

RESUME

La complexité des nouveaux systèmes ne cesse de s'accroître, en termes d'intégration de multiples technologies, de nombre de composants, ainsi qu'en termes de performances attendues et en contraintes sécuritaires. De plus, les phases de conception de ces systèmes doivent également garantir la tenue de délais stricts pour un coût maîtrisé. Nous avons développé une méthode d'analyse de la sûreté de fonctionnement des systèmes complexes, intégrée aux méthodes d'ingénierie système dirigées par les modèles nommée MéDISIS. Nous montrons l'apport du langage SysML [14] pour l'établissement d'un modèle commun aux différents intervenants du projet, la traçabilité des exigences, et l'automatisation des liens vers les études de sûreté de fonctionnement [5,6,7,8].

Nous présentons dans un premier temps, comment extraire des diagrammes SysML fonctionnels, les informations nécessaires aux études de risques et constituer une AMDEC préliminaire servant de base aux études de sûreté de fonctionnement. Pour cela, des règles d'analyse des modèles SysML sont formulées à partir de l'étude de méta-modèles et corroborées par le biais d'études de cas industriels. À partir de ces règles, nous proposons une procédure de traitement des modèles SysML pour la génération d'AMDEC préliminaire. Cette procédure fait appel au retour d'expérience géré à travers une base de données des comportements dysfonctionnels.

Dans un second temps, nous expliquons la mise en place d'une traduction des modèles SysML vers une description utilisant AltaRica Data Flow, destinée à l'analyse quantitative de la SDF [15]. Organisée en deux étapes, la création des modèles en AltaRica Data Flow, comporte la traduction directe de la partie fonctionnelle depuis des modèles SysML. Puis la partie concernant le comportement dysfonctionnel est ajoutée en exploitant les données recueillies lors de la phase d'AMDEC et gérées à travers notre base de données des comportements dysfonctionnels.

Le troisième axe développé est celui des contraintes temporelles, où nous soulignons les points communs entre SysML et des langages de description d'architecture (ADL) comme AADL (Architecture Analysis & Design Language)[3]. Ici, nous nous attachons à focaliser les modèles sur les contraintes temporelles tous en gardant le lien vers les exigences et permettre ainsi d'introduire très en amont du cycle de développement les spécifications temporelles notamment lors des choix d'architectures [4].

En termes de conclusion, nous indiquons comment ces activités s'inscrivent dans le cadre de l'étude et la conception d'un véhicule expérimental hypersonique, fruit d'une collaboration en cours avec la société MBDA. Nous indiquerons les difficultés et gains obtenus lors du déploiement de notre méthode dans un processus déjà existant. Ces réflexions portent sur les phases amont du cycle de développement.

1 INTRODUCTION

Si l'on étudie les systèmes à fortes contraintes de Sûreté de Fonctionnement, on constate qu'ils ne cessent de se complexifier. Cette tendance se vérifie aussi bien pour la réalisation de systèmes de sécurité, que pour celle de transport de personnes ou de systèmes d'exploration spatiale. Cela transparaît notamment à travers la complexification des interfaces et composants remplissant les missions confiées aux systèmes..

La démarche d'Ingénierie Système adoptée pour leur conception doit permettre de gérer toutes les tâches nécessaires au développement du système. Les analyses de Sûreté de Fonctionnement prennent une part importante de ce cycle de développement, consomment un temps non négligeable et requièrent une forte expertise du domaine tant les techniques et formalismes à employer sont complexes. Le défi qui se dégage est donc de favoriser l'interconnexion des activités de conception pures et des étapes d'étude de la Sûreté de Fonctionnement. Pour cela, il faut comprendre les techniques employées pour ces deux facettes de la création des systèmes et en particulier, analyser comment unifier et faire coopérer les outils employés par chaque communauté.

En effet, de tels développements ne se font sans l'utilisation de méthodes rigoureuses supportées par les formalismes et outils adéquats. L'utilisation de l'Ingénierie Système basée sur les Modèles est primordiale pour des projets de cette envergure. Nous avons donc cherché à dresser une méthodologie d'intégration des analyses de Sûreté de Fonctionnement au processus d'Ingénierie Système basée sur les modèles, que nous avons intitulée MéDISIS. Le but recherché est celui appelé de leurs vœux par les auteurs de [16] : « [dissoudre] les frontières entre les environnements de développement virtuel au service des Concepteurs, et les référentiels de simulation accompagnant le travail des Fiabilistes, Logisticiens et Risk Managers, […] pour laisser place à une discipline commune que l'on pourrait qualifier d'Assurance Performentielle… ».

2 L'ISBM ET LA GESTION DES EXIGENCES DE SDF

2.1 Présentation générale de l'ISBM et ses défis

L'Ingénierie Système Basée sur les documents a longtemps été l'approche usuelle pour l'ingénierie des systèmes physiques. Puis certains domaines l'ont abandonnée au profit d'approche d'ISBM.

L'ISBM est l'utilisation formalisée de modèles comme supports aux activités de spécification, conception, analyse, vérification et validation des systèmes, débutant de la phase de conception préliminaire et se poursuivant à travers le développement jusqu'aux dernières phases de vie du système [12]. Les techniques d'ISBM mettent l'accent sur l'évolution et la progression dans le niveau de détail lors de l'étude du système. L'ISBM est censée faire progresser l'Ingénierie Système dans de nombreux domaines comme la communication entre les différents acteurs d'un projet, la précision de l'analyse, l'intégration des résultats ou la réutilisation des connaissances produites, tous ces aspects participent à une réduction substantielle des risques liés au développement. Pour être mise en place, l'ISBM doit être déployée par une méthodologie. [9] indique qu'une méthodologie d'ISBM peut être décrite comme une collection de processus, méthodes et supports utilisés pour assister la réalisation de l'IS dans un contexte « basé modèle » ou « dirigé par les modèles ». Cette définition fait apparaître la notion centrale de modèle, en prise directe avec la ou les méthodes membres de la méthodologie. L'ISBM possède deux types d'entités fondamentales : le modèle du système et les bases de connaissances contenant le modèle et ses dérivés. La gestion du modèle et des connaissances qu'il génère impose l'utilisation d'un Environnement de Développement Système (EDS) réunissant la suite d'outil employée au cours de l'ISBM.

Les méthodologies d'ISBM doivent une grande part de leur efficacité à la puissance de l'EDS outillant la démarche. [10] donne une définition complète des EDS : un EDS fait référence aux outils et bases de connaissances utilisés pour le développement d'un système. Les outils communément utilisés dans un EDS permettent : la modélisation, l'analyse des composants physiques et logiciels, le test et le management de projet. Les bases de connaissances doivent alors permettre le transfert et la cohérence des informations. Nous proposons l'ajout cohérent d'outils spécifiques d'analyse de SdF ; MéDISIS rassemble les nouveaux processus d'ISBM associés à leur emploi.

2.2 Intégrer les analyses SdF à l'ISBM et à l'EDS

Au cours du processus d'ISBM, le modèle commun d'ingénierie utilisé doit permettre de connecter et initier les différentes activités d'analyse et de conception détaillée nécessaires à la production du système désiré. Pour l'analyse de systèmes à fortes contraintes de SdF, le processus passe par d'importantes analyses du comportement du système, incluant l'observation des activités fonctionnelles et dysfonctionnelles du système. La validation formelle et la recherche de scénarios de défaillance sont les deux grands axes de travail relevés dans la littérature.

Le processus d'agrégation des connaissances nécessaire à une analyse de SdF peut être décomposé en deux phases.

Dans un premier temps (phase qualitative), il nécessite la création et l'incorporation de nouvelles connaissances dépassant la vue fonctionnelle du système. Il s'agit du recensement du comportement dysfonctionnel de chaque élément du système. Cette partie requiert l'emploi de méthodes de création ou de recouvrement d'informations, procédant par analyse de la description fonctionnelle du système et utilisation de jugement d'experts. Ce sont d'une part l'analyse fonctionnelle menée dans les premières phases du processus d'IS, et la réalisation d'Analyse des Modes de Défaillance de leurs Effets et Criticité (AMDEC) et d'Analyse Préliminaire des Risques (APR) d'autre part.

Dans un second temps (phase quantitative), l'analyse de SdF du système se poursuit par l'exploitation des connaissances précédemment rassemblées. Le but est alors la qualification et la quantification des aspects plus généraux du modèle, tel que l'atteignabilité d'un état redouté ou le calcul de la disponibilité globale du système. Durant cette phase, la connaissance nécessaire a déjà été produite, il reste à l'analyser pour produire les vérifications souhaitées. Ceci nécessite la création d'un modèle formel du système reflétant les facettes

fonctionnelles et dysfonctionnelles de celui-ci. Les analyses produites à partir de ces modèles procèdent de deux manières distinctes : l'analyse par scénarios de fonctionnement du système (scénarios unitaires ou ensembles de scénarios), la preuve mathématique de propriétés du modèle. De nombreux outils, méthodes, langages et formalismes ont été développés pour réaliser cette partie de l'analyse. On peut distinguer deux approches principales, de par leur diffusion : l'utilisation, les automates à états finis et les Réseaux de Petri (RdP).

Nous avons bâti une méthode pour intégrer efficacement le sous-processus d'analyse de SdF des systèmes au processus plus global d'ISBM. Cette méthode intitulée MéDISIS (Méthode D'Intégration des analyses de Sdf à l'Ingénierie Système) a fait l'objet de diverses publications [5,6,7,8]. Son but est de proposer un enchaînement de traitements de l'information et des connaissances, incluant la création, l'expression, l'analyse, la pérennisation et la réutilisation répondant aux besoins décrits précédemment. Ces traitements doivent être supportés par des outils et répertoires formant un EDS cohérent. Les objectifs à atteindre par la méthode sont les suivants :

- Faciliter la transmission de connaissances entre équipes.
- Accélérer la réalisation des études de SdF.
- Organiser l'exploitation commune des connaissances sous forme de modèles.
- Permettre la réutilisation des connaissances entre projets.
- Identifier les besoins d'analyse et réaliser le suivi de leurs résultats.
- Améliorer la cohérence et la qualité des analyses SdF.

Nous considérons que la source d'information initiale à MéDISIS est un modèle SysML purement fonctionnel, résultat de l'analyse fonctionnelle du système étudié. La méthode a pour objectif de maximiser l'efficacité des analyses SdF durant la conception, en les rendant plus rapides, cohérentes et pertinentes. Pour cela, une Base de données des Comportements Dysfonctionnels (BCD) constitue l'élément central de MéDISIS. Cette BCD regroupe les connaissances collectées sur les mécanismes de défaillances des entités composant le système analysé. MéDISIS inclut des services de génération automatique d'analyse ou de modèle, utilisés pour réaliser chaque étape de la démarche. La figure 1 schématise MéDISIS et présente l'enchaînement des procédures effectuées et l'utilisation de la BCD.

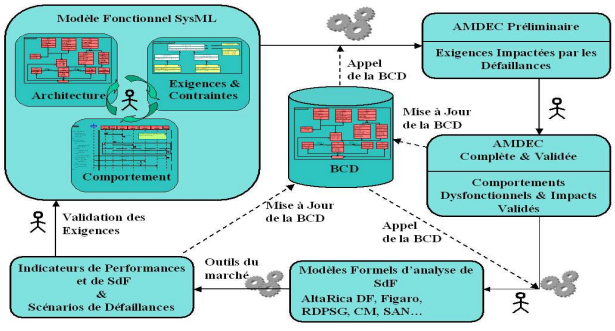

Figure 1. La méthode MéDISIS

MéDISIS est une méthode déductive et incrémentale se déroulant en 3 phases majeures :

- Recherche du comportement dysfonctionnel des composants et des influences entre composants voisins. Identification des exigences impactées. Cette phase est finalisée par une AMDEC munie d'une assistance automatisée et correspond à la phase qualitative.
- Construction assistée d'un modèle utilisant une représentation formelle du comportement fonctionnel et dysfonctionnel du système (phases quantitative).
- Analyse outillée des modèles formels, validation des exigences, retour sur la conception.

3 L'ASSISTANCE A LA CREATION D'AMDEC PAR L'ANALYSE AUTOMATISEE DE MODELES SYSML

L'automatisation et la réalisation d'AMDEC sur des modèles utilisant un langage spécifié sont des leviers importants pour la réussite de l'intégration des AMDEC à l'ISBM. L'AMDEC pourra bénéficier des mêmes avantages que ceux relevés lors de l'utilisation de méthodes basées sur les modèles, à savoir : meilleure qualité d'étude, meilleure communication entre équipes, meilleure productivité et meilleur transfert des connaissances. En effet, par l'analyse de modèles purement fonctionnels exprimés en SysML, il devient possible d'automatiser des étapes comme la recherche de composants et de guider l'analyse de l'expert lors de la rédaction de l'AMDEC finale.

Lors d'une AMDEC, le processus de prise de décision est mené pour établir des relations de cause à effet à partir de vues structurelles et fonctionnelles du système pour générer un tableau AMDEC. L'étude AMDEC passe par les étapes suivantes :

1. Recensement des composants à étudier.
2. Pour chaque composant, rechercher les MdD.
3. Pour chaque MdD, rechercher les causes possibles.
4. Pour chaque couple MdD/causes possible, rechercher les effets possibles sur les composants périphériques.
5. Pour chaque effet, réaliser la cotation du risque (RPN), proposer des moyens de détection et spécifier les actions à mener pour diminuer le risque.

Toutes ces étapes engendrent la constitution d'un tableau regroupant les résultats de l'AMDEC. L'étape 1 se base sur une vue structurelle du système, les éléments à identifier sont les composants du système. Cette vue est couverte par l'axe structurel de SysML. L'étape 2 fait appel aux connaissances des experts (organisées ou non dans des bases de données), les éléments relevés sont le nom des MdD, cette étape est réalisée dans MéDISIS par la connexion à la BCD. L'étape 3 est la première faisant appel au raisonnement de causalité, elle est basée sur l'expérience des experts et l'analyse des vues du système. En effet, comme nous le constatons dur la figure 4 la défaillance peut provenir de l'entourage du composant affecté. Pour cela, il convient d'identifier les interfaces du composant ouvertes à l'influence des composants périphériques. Pour l'étape 3, on recense les causes connues par les experts, les interfaces offertes par le composant et les composants périphériques, puis on conserve ceux dont la fonction peut provoquer le MdD. Pour cela les diagrammes de l'axe structurel de SysML sont analysés pour découvrir les chemins de propagation. L'étape 4 fait également appel au raisonnement causal :

le MdD du composant induit un changement de l'exécution de ses fonctions. Ces fonctions dégradées produisent leurs sorties, transmises par les interfaces du composant aux composants périphériques. La réception de ces flux potentiellement erronés provoque des changements d'état des composants périphériques et de leurs traitements, ces informations sont les effets du MdD. Les éléments à produire ici sont les comportements des fonctions altérées, les flux altérés, les interfaces les relayant et les composants périphériques touchés ainsi que leur comportement modifié. Pour cela on étudie plus en détail les diagrammes de l'axe comportemental de SysML, ces derniers montrent les enchaînements de traitement réalisés ainsi que les paramètres impactés. Enfin, l'étape 5 fait appel aux jugements et connaissances des experts pour produire le RPN et communiquer les solutions adéquates pour la gestion du MdD.

Nous avons ainsi énoncé 7 règles d'analyse des modèles SysML utilisées pour la préparation des AMDEC :

Règle N°1 : Les composants identifiés par l'AMDEC sont les *parts* du modèle. Leur nom donne le nom du composant. Leur type est donné par le *block* dont le *part* est une instance.

Règle N°2 : Les fonctions de chaque composant sont identifiées en relevant : les *opérations* du *block* typant le *part*, les *actions* allouées par une *swimlane* au *part* ou au *block* le typant, es *actions* de type « call operation action » appelant une *opération* du *block* typant le *part*.

Règle N°3 : Les interfaces et relations structurelles sont identifiées par l'analyse des *ports*. Les *ports* reliés entre eux par les *connecteurs* et les *parts* qui les possèdent sont respectivement les interfaces et les composants périphériques. Le sens de l'interface est déduit en considérant la *propriété direction* des *flow ports*.

Règle N°4 : Les composants périphériques sont, en complément de ceux identifiés par la règle N°3, les *parts* recevant/émettant des *messages* du/vers le *part* étudié et les *parts* responsables d'*actions* en relation avec celle du *part* étudié.

Règle N°5 : L'analyse de la propagation des défaillances des composants à travers sa représentation fonctionnelle, s'appuie sur les diagrammes de l'axe comportemental, afin de rechercher les causes et effets de ces défaillances. La propagation est réalisée dans le sens permis par les interfaces identifiées par la règle N°3 et les interactions relevées par la règle N°4.

Règle N°6 : Les exigences relatives aux composants sont identifiées en considérant les relations *satisfy* entre les *parts* et les *requirements*, entre les *typing blocks* ou *owning blocks* et les *requirements*, et entre les *properties* des *parts* ou *typing blocks* et les *requirements*. Les *requirements* dérivés (*derived*) ou possédant un lien de *containment* avec les *requirements* satisfaits par le *part* sont également liés au composant.

Règle N°7 : Les contraintes en relation avec les composants sont identifiées en considérant les *constraint properties* dont les *constraint parameters* sont liés aux *values properties* des *parts*. Ce sont en particulier toutes les *constraint properties* du *typing block* des *parts*.

Ces règles fournissent un support à l'analyse du système à travers son modèle SysML en vue de la réalisation de son AMDEC. Ces dernières permettent d'isoler dans le modèle complet les éléments d'informations nécessaires pour la réalisation des étapes successives constituant le raisonnement produit lors d'une AMDEC. Le tableau 1 résume les phases d'utilisation de chaque règle vis-à-vis des étapes de l'AMDEC.

Tableau 1. Analyse de modèle SysML et création d'AMDEC

Étape de réalisation AMDEC / Règle d'analyse SysML/elts MéDISIS	1 : Recensement composants	2 : Recensement MdD	3 : Recherche Effet/Cause	5 : Cotations Actions correctives
Règle 1	●			
Règle 2		●	●	
Règle 3			●	
Règle 4			●	
Règle 5			●	
Règle 6			●	●
Règle 7		●	●	
BCD		●		●
Jugement d'expert		●	●	●

4 DE SYSML À ALTARICA DATA FLOW

La constitution d'un modèle formel du système reflétant son comportement dysfonctionnel est un point crucial pour ses phases de validation. Les qualités que l'on cherche à atteindre au cours de cette étape sont en tous points identiques à celles recherchées pour le processus AMDEC : traçabilité envers le modèle central du système et ses exigences. La volonté principale est donc d'assister, voire d'automatiser la création du modèle formel à partir du modèle fonctionnel du système et des connaissances du comportement défaillant. Nous proposons pour cela de réaliser une traduction des modèles SysML vers le langage AltaRica Data Flow.

Nous nous plaçons dans une phase de l'étude du système où l'AMDEC a été réalisée. Par cette analyse, constituant le lancement de MéDISIS, les analystes ont obtenu les Modes de Défaillance de chaque composant et ont pu cibler leurs effets potentiels sur le système. L'étude à mener est alors de qualifier et quantifier les effets de ces défaillances à l'échelle du système global. Ces validations viennent en réponse aux exigences formulées par les parties prenantes ou en prévision d'une demande de certification. Il s'agit donc d'utiliser toutes les connaissances contenues dans le modèle fonctionnel commun au processus d'Ingénierie Système, mais également d'exploiter les connaissances relevées sur le comportement dysfonctionnel au cours de l'AMDEC.

Dans cette partie, nous étudions l'aide à la saisie du modèle dysfonctionnel en AltaRica Data Flow, en ciblant la traduction des éléments du modèle SysML en AltaRica Data Flow. Cette construction se fait en trois phases et sous deux vues : la vue fonctionnelle et la vue dysfonctionnelle. Nous détaillons la retranscription de la structure du système, puis de son comportement fonctionnel et enfin l'adjonction du comportement dysfonctionnel. Pour les deux premières parties,

nous travaillons à partir du modèle SysML issu de l'analyse fonctionnelle. La partie dysfonctionnelle est constituée en exploitant la BCD constituée à partir de l'expérience de l'institution développant le système et surtout par les éléments recensés lors de l'AMDEC.

Afin de résoudre les problèmes de complexité de modélisation des grands systèmes multi-technologiques, SysML propose de suivre la voie d'une représentation hiérarchique et compositionnelle. Le concept de *block* se retrouve donc au centre de la représentation. Dans un second temps, la spécification met l'accent sur la représentation des liens constituant le maillage de l'architecture étudiée. Les concepts de flux se sont donc vus attribuer une modélisation structurée par leur typage, orientation et valeur. Enfin, SysML propose de nombreux concepts de modélisation offrant la possibilité de caractériser les changements d'état et le séquencement des traitements réalisés. Ces stratégies pour aborder la modélisation d'un système se retrouvent dans le langage AltaRica DF, dont la notion de composants s'échangeant des flux influencés par et influençant leur comportement est fondatrice. Nous avons ainsi pu proposer une méthode de traduction entre ces modèles pouvant aller jusqu'à l'automatisation du passage d'une représentation à l'autre.

La construction du modèle AltaRica est effectuée en deux étapes et a été discutée dans [8]. Dans un premier temps, le modèle fonctionnel est constitué. Ensuite, la partie dysfonctionnelle est ajoutée sur le modèle précédemment constitué dans lequel apparaissent les composants physiques de l'architecture et leur comportement nominal. Cette seconde étape est réalisée grâce aux informations glanées lors de l'étude AMDEC et par l'utilisation de la BCD commune au processus d'ingénierie global. Pour chaque élément de la transformation de modèles, nous identifions les éléments du modèle SysML à prendre en compte et leur expression en AltaRica. Pour chacune des phases de la transcription, nous analyserons les restrictions ou adaptations nécessaires en matière de modélisation SysML pour automatiser le processus.

Les tableaux qui suivent expriment les correspondances existant entre les deux langages. Le tableau 2 présente l'obtention des concepts représentant la structure du système, le tableau 3 s'intéresse au comportement fonctionnel. Nous recensons les concepts à modéliser pour représenter le système et nous donnons les artefacts employés dans les deux langages pour refléter ces derniers.

Tableau 2 Traduire la structure du système

Concept	AltaRica DF	SysML
Type de composant / Instanciation	Node Field : sub	Block Part
Variable d'état / Type	Field state Bool, integer, float, domain	Value properties Value Type
Variable de flux / Type / Direction	Field : flow Bool, integer, float, domain In ou Out	Value properties Value Type ∅
Port (Interface) / Type / Direction	Une variable de flux est aussi un port	Flow/standard port Value Type/Block Flow port Direction/Interface
Connexion Inter-composants	Assertion : égalités entre variables de flux	Connectors des Internal Block Diagrams

Cette mise en correspondance des deux langages fait apparaître de nombreux points de recouvrement entre eux. Néanmoins, certains points de la traduction nécessitent l'utilisation de méthodes de traduction pour obtenir le modèle complet en AltaRica DF. De manière générale, SysML dispose d'artefacts plus nombreux et laisse une plus grande latitude au concepteur dans le typage des éléments du modèle.

Pour obtenir une traduction directe, il est nécessaire de définir soit des routines d'interprétation du modèle SysML, soit de définir des règles d'utilisation de SysML lors de la saisie du modèle. Nous avons pu constater que de telles restrictions loin d'alourdir la modélisation SysML, rendaient celle-ci plus cohérente d'un point de vue de l'ingénierie globale. Ces règles de saisie concernent par exemple la déclaration de liens explicites entre variables et interfaces convoyant le phénomène physique qu'elles caractérisent. De même, en ce qui concerne la description du comportement fonctionnel (tableau 3), il est nécessaire de déclarer des allocations entre les composants et les comportements qu'ils exécutent.

Tableau 3. Traduire le comportement fonctionnel

Concept	AltaRica DF	SysML
Comportement interne	Assertion	Operations Constraint properties
Comportement événementiel	Transition : guard	Ordonnancement messages Constraint properties Gardes d'activités
	Transition : event	Opérations Activités Actions
	Transition : affectation	Constraint properties
Synchronisation des comportements	Field : sync	Interaction Activités

L'adjonction de la partie dysfonctionnelle peut être gérée de deux façons, pouvant également être complémentaires. La première solution est de transmettre le « squelette » fonctionnel du modèle, comportant la définition de la structure et du comportement fonctionnel, aux experts de SdF et de confier à ces derniers la saisie de la partie dysfonctionnelle. Cette saisie reprend les techniques employées actuellement à l'exception près que la partie commune au modèle du concepteur est déjà traduite et a priori cohérente, car résultat d'une traduction automatique. La deuxième solution est d'ajouter de façon automatique certaines parties du comportement dysfonctionnel. Pour cela, il faut disposer au préalable d'une base de données de type BCD, comme nous le préconisons dans MéDISIS. Cette base doit être formée pour accueillir les résultats de l'AMDEC et le pérenniser en vue de leur utilisation pour la création du modèle d'analyse de SdF. L'initialisation d'une telle base a été discutée dans [7].

Tableau 4. Traduire le comportement dysfonctionnel

Concept	AltaRica DF	SysML/MéDISIS
Condition de défaillance	Transition : guard	Jugement d'expert BCD : • Diagrammes paramétriques • Value properties • Operations • Allocations • Statemachines
Déclenchement	Transition : event	
Premier effet	Transition : affectation	
Comportement	Assertion	
Caractéristiques	Field : state Field : extern : law	

La traduction entre modèles SysML et AltaRica DF que nous pouvons fournir permet de retirer différents avantages pour le processus d'analyse de SdF. Dans un premier temps, elle permet d'adosser le modèle d'analyse SdF au modèle commun de l'EDS, pour ainsi disposer d'une représentation cohérente avec le système développé. Deuxièmement, une telle transformation offre l'occasion de créer rapidement une majeure partie du modèle formel nécessaire aux validations inhérentes à la SdF. Enfin, cette équivalence entre SysML et AltaRica DF autorise le développement d'une base de connaissances commune sur les dimensions dysfonctionnelles du système, capable d'interagir avec les modèles des concepteurs et des experts chargés de la validation. Cette traduction dispose des mêmes possibilités d'implémentation que le service de création d'AMDEC. Nous avons en outre noté que les règles de modélisation SysML à mettre en place pour faciliter la traduction se montraient cohérentes avec celles énoncées pour l'analyse AMDEC et participaient également à obtenir un modèle cohérent et expressif pour l'ingénierie.

5 DE SYSML À AADL

AADL est d'abord un langage textuel formel dont la première version est parue en 2004, [1]. Son méta-modèle, sa définition sous forme graphique et son extension de modélisation d'erreurs sont publiées en tant qu'annexe en 2006, [2]. La récente révision [3] marque tout l'intérêt de la communauté à maintenir ce langage opérationnel, notamment dans le domaine de l'embarqué avionique. Chaque composant du système embarqué est décrit en AADL suivant deux aspects. Le premier, le *type*, correspond à l'interface fonctionnelle du composant. Le second, l'*implémentation*, décrit le contenu du composant (sous-composants, propriétés temporelles, connexions, etc.). Dans la pratique les descriptions du *type* et de l'*implémentation* peuvent être faites par des personnes différentes, chacune ayant en charge une étape dans le raffinement de la description de l'architecture, de la spécification jusqu'à la conception détaillée.

Chaque composant appartient à une des 10 catégories prédéfinies en AADL, classifiées elles-mêmes en 3 sous-ensembles : matériel (Execution Platform), logiciel (Application Software) et hybride (Composite). Chaque *type* de composant AADL possède une notation graphique reprise sur la figure 2.

Le modèle SysML recèle déjà de nombreuses informations sur le système ; il est incompatible avec la notion d'EDS, de les recréer lors de la modélisation en AADL. Toutefois, les informations spécifiques à la dimension temporelle du système ne peuvent pas être déduites du modèle SysML.

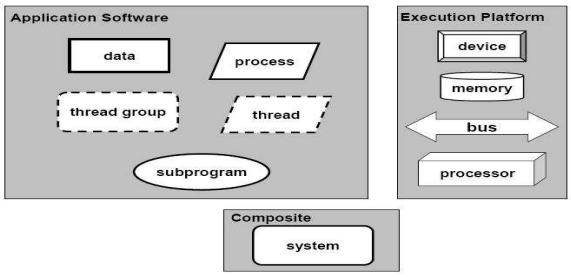

Figure 2. Représentation graphique des différentes entités du langage AADL

En effet peu d'artéfacts sont prévus en SysML pour modéliser les données liées au temps. De plus, ces données sont généralement utilisées lors des étapes de conception détaillées, or ici nous nous situons aux étapes de spécification et de conception préliminaire du cycle de développement d'un système. Par contre nous devons assurer la meilleure traçabilité possible entre les deux modèles et éviter toute saisie inutile. Il sera donc important de bien cerner les possibles liens entre ces deux langages résumés par le tableau 5.

Tableau 5 Traduction des concepts de bases

Concepts	**AADL**	**SysML**
Composant logiciel /instanciation	Software components implementation	Block Part
Composant materiel /instanciation	Hardware components implementation	Block Part
Association	Binding	Block Association
Sous composants	Subcomponents	Part
Port (Interface) / Type / Direction	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

SysML fournit l'architecture du système embarqué, représentée par l'ensemble des connexions, relations et échanges entre composants du système. De plus, le principe des BDD et IBD de SysML s'adapte naturellement au mécanisme de *type* et d'*implémentation* de AADL. En effet, les informations élicitées lors de la création d'un bloc dans un BDD en SysML correspondent à la définition de l'interface fonctionnelle du composant, c'est-à-dire le *type* d'un composant en AADL. L'IBD d'un bloc en SysML représente exactement le principe d'*implémentation* en AADL : l'organisation interne du composant. Ainsi, les *parts* deviennent des composants en AADL, les *flow ports* deviennent des *ports* typé de AADL selon les informations échangées, les *flow spécifications* seront identifié a des *flows* AADL,…

Cependant pour ce qui est de la classification des composants, un problème se pose, puisqu'en en SysML le bloc sert aussi de « type », et donc les composants sont définis selon leur type/bloc et leur nom/part. Alors qu'en AADL il est nécessaire de classifier nos composants suivant les 10 catégories prédéfinies (Memory, processor,…). Ainsi, afin de permettre la transcription d'un langage à l'autre, il nous faut envisager l'intervention d'une source d'information supplémentaire, et pour cela trois possibilités sont envisageables :

- Mettre en place une méthode de modélisation SysML qui préciserait la catégorie à laquelle appartiennent les blocs.
- Avoir recours à une décision de spécialiste pour catégoriser les blocs, un questionnaire permet de faciliter cette étape.
- Utiliser une base de données des correspondances entre les blocs SysML et leur catégorie en AADL pour permettre de centraliser l'expérience des transcriptions antérieures et faciliter les suivantes.

Notons que cette classification des parties du système en AADL est du domaine de l'expert métier et non de l'architecte. Cette limite est bien évidemment liée à la composition des équipes de concepteurs et doit être définie dans les procédures de conception ou dans la définition des livrables entre parties prenantes du projet.

Le dimensionnement d'un système en AADL, nécessite la quantification des « properties » des composants extraits, en effet ces propriétés permettent l'emploi des outils d'analyse temporelle, tel que cheddar [17] ou RMA [13]. Comme cela a été évoqué, les *value properties* de SysML permettront partiellement de renseigner les *properties* du modèle AADL, mais il nous faudra encore une fois faire appel à une source extérieure d'information pour avoir un renseignement complet de toutes nos *properties.*

Pour nos besoins d'informations extérieurs, on privilégiera une approche similaire à celle choisie pour la création d'AMDEC et de modèle d'Altarica DF, à savoir, ne pas imposer de méthodologie de modélisation du système en SysML, avoir recours à un expert en système embarqué qui renseigne les propriétés manquantes ainsi que les catégories inconnues, et enfin mettre en place une base de données qui stocke ces réponses. Par la suite, la base de données permettra de limiter le nombre de questions posées à l'expert en tirant parti des informations précédemment recueillies.

Comme pour la création d'AMDEC à partir de SysML, nous suivons une succession d'étapes d'analyse et de traitement :

Etape N°1 : Identifier la totalité des blocks et parts et établir les hiérarchies entre ces entités, en tenant compte des notions de contenant/contenu.

Etape N°2 : Mettre au point les, liens structurels et compositionnels, des différentes entités qui composent notre système.

Etape N°3 : Effectuer l'opération de complétion du *type* des entités (ex : ce block « shared_memory » est du type memory), avec l'aide d'une source d'information extérieure (Spécialiste et base de données).

Etape N°4 : Réaliser le modèle AADL structurel au niveau textuel et son équivalent graphique.

Etape N°5 : Effectuer l'opération de complétion (sélection) des variables dimensionnantes, avec l'aide d'une source d'information extérieure (idem étape 3)

Etape N°6 : Intégrer ces informations au code du modèle, sous forme de variables associées aux entités précédemment créées (en 2).

L'automatisation possible des étapes 1,2,4,6 est visible, cependant pour les étapes 3 et 5, même avec l'utilisation de la base de données, une intervention humaine experte est nécessaire. Le tableau 6 indique pour chaque phase du processus usuel de modélisation en AADL, les étapes de traduction utilisable, indiquant par le fait l'aide fournie.

Tableau 6 allocation des étapes d'analyse aux phases de modélisation

Phase de modélisation AADL / Etape d'analyse SysML/elts MéDISIS	1 : Recensement composants	2 : Classification des composants	3 : Dimensionner le système	4 : Rédiger un modèle AADL complet
Etape 1	●			
Etape 2	●	●	●	
Etape 3		●		
Etape 4				●
Etape 5			●	
Etape 6				●
BCD		●		●
Jugement d'expert		●	●	●

6 RETOUR D'EXPÉRIENCE ET CONCLUSION

MéDISIS augmentée des fonctionnalités nécessaires aux systèmes embarqués est actuellement utilisée dans le cadre d'un partenariat avec la société MBDA, afin de réaliser le système de contrôle de vol d'un véhicule hypersonique. Plus particulièrement, nous relevons, ici, les avantages rencontrés lors de la phase d'analyse du projet.

Nous nous plaçons au moment où les exigences fonctionnelles sont encore exprimées sous forme littérale. Le premier travail à effectuer est de réécrire ces exigences sous une forme aisément compréhensible pour les utilisateurs et les concepteurs. Cette étape que l'on appelle le raffinement des exigences permet de tendre vers des exigences unitaires correspondant à des fonctions elles-mêmes unitaires de notre système. Elle consiste à diviser nos exigences. Afin de préserver de l'intégrité et la cohérence des exigences, 3 règles simples sont à respecter :

- Le raffinement des exigences ne doit pas contenir de boucle.
- Éviter de raffiner une exigence par un trop grand nombre d'exigences. Situation rencontrée, lorsque des étapes dans le processus itératif ont été omises.
- Ne pas raffiner une exigence par une seule exigence. Cela est synonyme de perte d'information, ou d'un manque de précision de l'exigence initiale.

Dès que la spécification du système le permet, c'est-à-dire dès que les lois physiques définissant les réactions du système sont connues, nous pouvons traduire les exigences fonctionnelles unitaires sous forme de diagrammes paramétriques. Il en va de même pour toute fonction devant être réalisée par le système. Le modèle SysML du système contient alors une description hiérarchisée des contraintes statiques mathématiques d'un système sous forme de diagrammes paramétriques.

Le module de synchronisation nous permet de créer un schéma bloc Simulink de chaque diagramme paramétrique. Nous pouvons alors poursuivre le développement du modèle mathématique du système par simulation et tester ses performances, la fiabilité et l'exactitude de l'algorithme. En reprenant des travaux antérieurs [11] et suite à une génération

d'AMDEC, il est d'ores et déjà possible à ce stade d'introduire dans le modèle les comportements dysfonctionnels majorants afin de tester des choix d'architecture fonctionnels. Le modèle générique Matlab permet de reprendre la plupart des modes de défaillance utilisés lors d'une AMDEC. La BCD permet ici d'identifier le type de défaut et de renseigner les paramètres du modèle générique de défaillance.

Dans le cas de notre application nous avons pu établir que la redondance prévue des capteurs autour du fuselage du véhicule permettait d'établir avec un post traitement ad hoc une redondance d'ordre variant entre 2 et 3 en fonction des capteurs et des grandeurs mesurées.

Le module de synchronisation permet alors la mise à jour du modèle Simulink tant au niveau des « Parametric Diagrams » déjà créés qu'au niveau des nouveaux issus des blocs fonctionnels Matlab. Les artefacts transmis d'un modèle à l'autre sont les suivants : « Constraint Properties », « Constraint Parameters », « Value Properties » et les connecteurs. Plusieurs voies d'amélioration du processus restent à réaliser : l'insertion automatique des modèles de défaillance à partir de la « pré » AMDEC générée, ainsi que le paramétrage automatique des modes de défaillance en fonction de l'importance de leur criticité. Du point de vue des exigences l'établissement du modèle en Simulink et son analyse par simulation, permet dès les phases amont du projet de raffiner naturellement et de façon cohérente les exigences. Les exigences de sûreté de fonctionnement sont également introduites ou quantifiées de façon cohérente lors de cette simulation. La traçabilité est alors assurée par la synchronisation permettant de sélectionner les blocs devant se traduire par de nouvelles exigences sur le modèle SysML.

Simulink, et plus particulièrement la toolbox « Verification and Validation » met à notre disposition plusieurs outils de tests de nos exigences ainsi modélisés. On distinguera les tests fonctionnels et les tests structurels. Les tests fonctionnels nous renseigner quant à la conformité du comportement des composants de notre système, assimilant les blocs unitaires de traitement de notre modèle comme des boites noires dont il testera la réponse en sortie par rapport à des jeux de test prédéfini en entrée. À l'opposé on trouvera les tests structurels qui analyseront le comportement réel de notre modèle en traitant l'ensemble comme une boite blanche, vérifiant par exemple le taux de couverture du système correspondant à une simulation. La définition des jeux de test et leur gestion sont générés en SysML dans le modèle du système.

Lors que l'étape d'analyse fonctionnelle est achevée, la nature des flux de donnée manipulée par le système est connue, ainsi que ses principales fonctions de traitement. Lors de l'étape de spécification, il convient de réaliser une allocation des flux et de fonctions sur une structure organique du système. Cette technique est bien connue et est reprise également par les techniques d'ingénierie dirigée par les modèles. Le langage Sysml, se prête particulièrement bien à cette activité, car les flux de donnée sont définis par des blocs auxquels sont associées les contraintes, elles-mêmes définies par les « Parametric Diagrams ». C'est également la structure de modélisation que nous avons établie pour tous composants physiques de notre système. Chaque composant créé lors de cette étape se voit alors attribuer automatiquement les exigences issues des données qu'il manipule.

L'attribution des fonctions à la vue organique du système, nécessite quant à elle, l'utilisation d'allocateur spécifique (i.e. l'association stéréotypée « allocate ») ou l'emploi des diagrammes d'interaction tels que les diagrammes de séquence (ou diagramme d'activité), comme avec UML l'allocation devient implicite au processus de placement des fonctions sous forme de message (ou « swim lane »).

À ce stade de conception du système, plusieurs activités sont à réaliser, nous les avons identifiées sous le terme générique de choix d'architecture. Quels sont les enjeux réels de cette étape ? Ils consistent à : choisir les sous composants ad hoc, spécifier pour chacun d'eux les fonctionnalités et contraintes à remplir, les critères à prendre en compte sont évidemment les exigences fonctionnelles et non fonctionnelles telles que les contraintes de sûretés de fonctionnement.

Concernant les contraintes de sureté de fonctionnement reliée aux flux, elles impactent naturellement les composants physiques fournissant un port typé avec ce flux. Lors de la génération automatique d'AMDEC, nous identifions ainsi les contraintes, donc les exigences pouvant êtres impactées par l'occurrence d'une défaillance. Nous avons ainsi souligné, dans l'étude du véhicule, l'absolue nécessité de prendre une structure de communication conforme aux sollicitations imposées au système lors des phases de largage et d'accélération.

De par leur nature, les systèmes de contrôle embarqués sont également très sensibles aux choix architecturaux réalisés trop tôt ou sans considération des contraintes temporelles. Nous avons réalisé l'analyse temporelle du modèle fonctionnel de notre système en deux étapes : la définition des chemins critiques et la complétion du modèle temporelle.

La définition des chemins critiques est réalisée à partir de la modélisation structurelle et statique de notre système (figure 3). Nous définissons la dynamique des échanges de données au sein de ce système entre le calculateur de vol (composant qui contrôle le moteur du véhicule) et le codeur (composant par lequel transite l'ensemble des données capteurs), sous la forme de diagramme de séquences. Une de ces séquences est illustrée par la figure 4, afin d'alimenter notre raisonnement. En particulier, nous déduisons plusieurs formules de l'âge des données des trames circulant entre le Codeur et le Calculateur au moment de leur traitement au sein du Calculateur :

Tadc = T_ech + Tv_wlEth_co + T_CEth_co + T_PE_co.
(Pire cas) Tadc = T_ech + (1/f) + T_CEth_co + T_PE_co
(Meilleur cas) Tadc = T_ech + T_CEth_co + T_PE_co
(Moyenne) Tadc = T_ech + (1/2f) + T_CEth_co + T_PE_co

Avec Tv_wlEth_co le temps d'attente de la donnée dans la mémoire du calculateur avant d'être utilisée pour répondre à une requête Ethernet. Tv_wlEth_co varie entre 0 et 1/f, où f est la fréquence d'échantillonnage de la donnée. Cette analyse de séquence de traitement dynamique nous permet de comparer nos différentes structures de traitements et ainsi dégager des critères de performances.

Une fois ces séquences de traitements parfaitement renseignées dans notre modèle SysML, et connaissant les constantes temporelles de chacune des actions de notre système, nous

pouvons réaliser la complétion du modèle temporel. Après traduction du système en AADL, l'étude temporelle suivant une analyse Rate Monotonic Analysis avec Cheddar [17] est réalisée permettant la sélection des composants envisagée en fonction de leur performance.

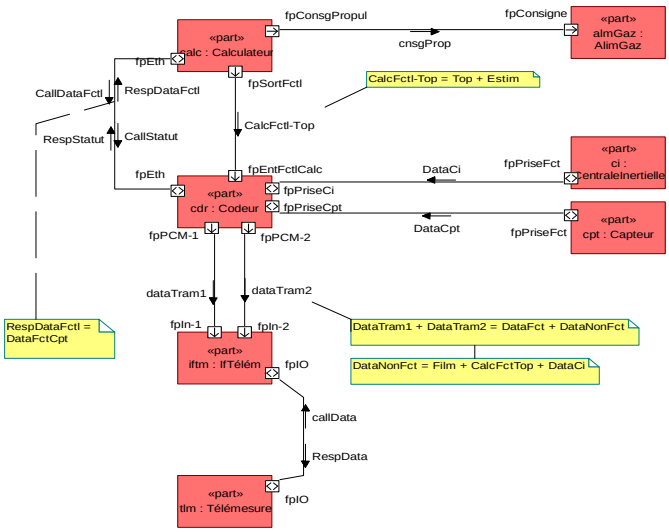

Figure 3. IBD représentant l'ensemble du système simplifié

Chacun des processus, décrits dans cet article, n'a pas le même niveau de développement, cependant tous ont montré leur intérêt dans notre EDS. Lors de la phase d'analyse de notre système embarqué, ils nous ont permis d'effectuer une réelle prise en compte des analyses de sûreté de fonctionnement et également de dégager une architecture organique maximisant les performances du système. Nous continuons actuellement notre projet en utilisant des spécifications exécutables offrant également la possibilité d'intégrer les comportements dysfonctionnels. Le développement des outils se poursuit également, facilité par l'aspect modulaire des EDS.

REFERENCES

[1] « Architecture Analysis and Design Language ». SAE. AS 5506 standard. Version 1.0 2004.

[2] « Architecture Analysis and Design Language ». SAE. AS 5506/1 standard, Annexes. 2006.

[3] « Architecture Analysis and Design Language ». SAE. AS 5506 standard. Version 2.0. 2009.

[4] R. Cressent. Comparaison et mise en place d'une méthode d'analyse et de modélisation de systèmes complexes, temps réel et embarqués. Rapport de stage Ingénieur de l'ENSIB, soutenu à Bourges le 1er septembre 2009.

[5] P. David, V. Idasiak & F.Kratz. *Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models*. Proceedings of ESREL 2008, 22-25 septembre 2008.

[6] P. David, V. Idasiak & F.Kratz. *Etude pour une meilleure intégration des données de conception dans les analyses de fiabilité*. Actes du 16ème Congrès Lambda, 7-9 octobre 2008.

[7] P. David, V. Idasiak & F.Kratz. *Use and improvements of SysML in reliability study*. Proceedings of RAMS 2009, Jan. 2009.

[8] P. David, V. Idasiak & F.Kratz. *Automating the synthesis of AltaRica Data-Flow models from SysML*. Proceedings of ESREL 2009, 7-10 septembre 2009.

[9] Estefan. Survey of Model-Based Systems Engineering (MBSE) Methodologies, Rev. B. *INCOSE MBSE Initiative, 23 Mai 2008*. 2008.

[10] S. Friedenthal, A. Moore & R. Steine. A Practical Guide to SysML : The Systems Modeling Language. *The MK/OMG press, Elsevier*. 2008

[11] V. Idasiak & P. David. Accident simulation: Design and results. Proceedings of ESREL 2007, 25-27 juin 2007

[12] International Council on Systems Engineering. Systems Engineering, INCOSE, Vision 2020. Version 2.03, TP-2004-004-02, Septembre 2007.

[13] Mark H. Klein, Thomas Ralya, Bill Pollak, Ray Obenza, Michael González Harbour, « A practitioner's Handbook for Real-Time Analysis », Kluwer Academic Publishers, 1993

[14] Object Management Group. OMG Systems Modeling Language (OMG SysML) V1.1. 1er novembre 2008.

[15] A. Rauzy. Mode automata and their compilation into fault trees. Reliability Engineering & System Safety, Vol. 78, pp. 1-12, 2002.

[16] A. Rauzy, Z. Brik & E. Arbaretier. Sûreté de Fonctionnement et Analyse de Performance. Actes du 16ème Congrès Lambda Mu, 7-9 octobre 2008

[17] F. Singhoff, « The Cheddar AADL Property sets (Release 2.x) » LISyC technical report number singhoff-03-07, February 2007.

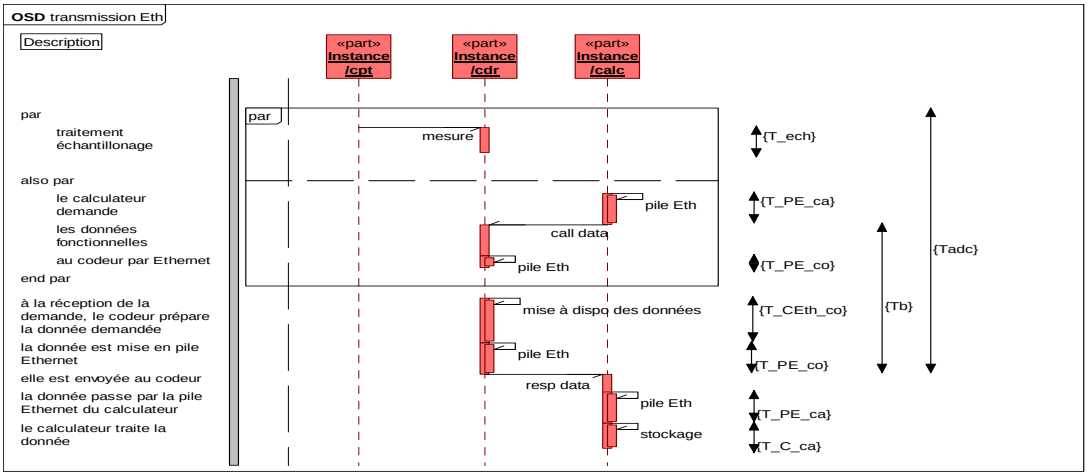

Figure 4. Exemple de séquence de traitement