VERS UNE METHODOLOGIE DE MODELISATION ALTARICA POUR DES SYSTEMES PHYSIQUES

TOWARD A METHODOLOGY FOR THE ALTARICA MODELLING OF PHYSICAL SYSTEMS

Romain ADELINE, Pierre DARFEUIL, Sophie HUMBERT
TURBOMECA groupe SAFRAN
Boite N°6, 64511 BORDES Cedex
Prénom.Nom@turbomeca.fr

Janette CARDOSO
ISAE
10 Avenue Edouard Belin,
31055 TOULOUSE
Janette.Cardoso@isae.fr

Christel SEGUIN
ONERA
2 Avenue Edouard Belin,
31055 TOULOUSE
Christel.Seguin@onera.fr

Résumé

L'ingénierie Dirigée par les Modèles (IDM) est aujourd'hui en développement pour réaliser les études de sécurité de systèmes en s'appuyant sur des langages de propagation de défaillance (*e.g.* AltaRica). Le problème de la modélisation (*i.e.* de l'obtention du modèle) étant devenu un problème clef, l'objectif poursuivi ici est l'obtention systématique d'une spécification informelle de modèles AltaRica. Pour cela, l'approche est fondée sur la méthodologie de modélisation présentée dans (Humbert 2006). Par rapport à ces travaux, nous améliorons cette méthodologie en tenant compte d'une gamme plus riche de systèmes et en définissant de nouvelles abstractions pour la construction de modèles AltaRica de systèmes mécaniques et hydromécaniques.

Summary

Model Based Engineering (MBE) is developed to assess the safety of a system by using languages such as AltaRica. Today, a crucial problem deals with the modelling phase of systems into such languages. The objective of the paper is to systematically realize an informal specification to help the AltaRica modelling of physical systems. The methodology described in (Humbert 2006) is used. We improve this methodology by defining new abstractions for mechanics and hydromechanics systems.

Introduction

Le processus de développement d'un système aéronautique comprend une phase d'évaluation de sa sécurité. Pour la réaliser, divers référentiels aéronautiques proposent des pratiques communément admises pour démontrer le fonctionnement sûr d'un système. On citera notamment l'ARP4754 (Aerospace Recommended Practice) recommandée pour guider le développement d'un système aéronautique et démontrer la tenue des exigences de navigabilité (SAE 1996a) et l'ARP 4761 qui recommande des méthodes d'évaluation de la Sûreté de Fonctionnement (SAE 1996b). Parmi ces méthodes, citons notamment la construction d'arbres de défaillance ou d'AMDE (Analyse des Modes de Défaillance et de leurs Effets).
Cependant, si ces méthodes sont très largement maitrisées et éprouvées, les systèmes actuels (systèmes complexes, reconfigurables et de taille de plus en plus importante) en font apparaître certaines limites. Pour aller au-delà, l'Ingénierie Dirigée par les Modèles (IDM) fonde le développement de systèmes sur la création de modèles. Initialement, l'IDM fut crée dans des optiques telles que la simplification de la conception de systèmes (principalement des systèmes logiciels) ou l'amélioration de la communication entre différentes équipes d'un même projet. Aujourd'hui, l'utilisation de l'IDM s'est rependue à la réalisation des études de sécurité de systèmes. L'objet d'une telle utilisation est alors la simulation de systèmes en présence de défaillances, la génération automatique d'arbres de défaillance ou la génération automatique des scénarios de défaillance conduisant à un évènement redouté. Parmi les différentes solutions outillant ces analyses et pour des raisons de maturité et de disponibilité d'outils, nous utilisons le langage de propagation de défaillance AltaRica.
Trois étapes peuvent classiquement être définies lors de l'étude de comportements de système : 1) la modélisation du système qui s'intéresse à la description de celui-ci et de ses composants dans un langage adapté (ici, AltaRica), 2) la validation du modèle qui assure que celui-ci est une abstraction correcte du système réel, 3) la vérification de la tenue des exigences du système par le modèle. Dans ce papier, nous abordons le premier point. A ce sujet, divers travaux ont déjà étudié l'approche pour la modélisation de systèmes logiciels, électriques ou hydrauliques (Kehren 2004). Nous souhaitons aujourd'hui améliorer cette méthodologie en tenant compte d'une gamme plus riche de systèmes et en définissant de nouvelles abstractions pour la construction de modèles AltaRica de systèmes physiques. Dans ce papier, nous nous intéressons à la modélisation de systèmes mécaniques et présentons les résultats obtenus sur les systèmes mécaniques, hydromécaniques et logiciels.
La suite du papier est organisée de la façon suivante : la première partie présente le contexte de travail et notre problématique ; la seconde partie introduit le langage AltaRica ; l'approche proposée est décrite dans la troisième partie ; la quatrième partie s'intéresse à la présentation de notre cas d'étude ; la cinquième partie présente l'application de l'approche proposée à différents systèmes de notre cas d'étude (Section 4, 5, 6 et 7) ; la dernière partie conclue sur le travail effectué.

Contexte de l'étude

1 Analyses classiques de Sûreté de Fonctionnement (SdF)

L'ingénierie de la SdF a le double rôle de 1) définir les exigences de sécurité que le système doit tenir et 2) assurer que le système satisfait effectivement ces exigences. En pratique, ces étapes sont fortement couplées et leurs réalisations se

formalisent avec différents types d'analyse. L'évaluation qualitative des exigences de sécurité a pour objectif de démontrer qu'aucune combinaison de défaillances (cessation de l'aptitude d'un bien à accomplir une fonction requise ; à ne pas confondre avec le terme panne - Une défaillance est un évènement, une panne est un état) d'ordre inférieur à *N* ne mène à l'occurrence d'un évènement *E* (*N* étant dépendant de la sévérité de *E*). L'évaluation quantitative des exigences de sécurité a pour but d'estimer la probabilité d'occurrence d'un évènement redouté (ER).

Deux méthodes utilisées sont l'analyse par arbre de défaillance et l'AMDE (Analyse des Modes de Défaillance et de leurs Effets). Une AMDE est une méthode d'analyse d'un système étudiant systématiquement les dysfonctionnements potentiels des composants de ce système. Un arbre de défaillance décompose un évènement en une combinaison booléenne d'évènements élémentaires (e.g. les défaillances issues de l'AMDE). Les chemins de propagation de défaillance sont alors représentés par des chemins logiques et il est possible de réaliser sur ces arbres des études qualitatives (combinaisons minimales de défaillances conduisant le système dans un état redouté) et quantitatives (probabilité d'occurrence de l'ER considéré).

2 Problématique

Bien que les arbres de défaillance soient aujourd'hui une méthode maitrisée et éprouvée, ils présentent aujourd'hui certaines limites. Parmi celles-ci, citons par exemple :

- les systèmes actuels sont des systèmes complexes (selon (Hutzler 2000), un tel système *« implique de nombreux composants qui interagissent dynamiquement à plusieurs niveaux ou échelles »*) hautement reconfigurables et la création des arbres de défaillance devient lourde et couteuse ;
- un arbre de défaillance est centré sur un unique évènement redouté ;
- le formalisme et la taille des arbres peuvent en rendre l'accès difficile aux non-initiés.

Quatre besoins majeurs ont été identifiés ; le premier étant un besoin de standardisation. (Nizet et al. 2001) définit ce concept comme *« la programmation à l'avance de certains aspects d'un travail de manière telle que les standards ainsi prévus aient une certaine stabilité dans le temps et s'appliquent de manière similaire »*. On retiendra que, généralement, la standardisation est un procédé qui permet de gagner du temps et de réduire les coûts. Plus rapide, moins cher et souvent plus fiable, ce procédé est aujourd'hui répandu dans de nombreux domaines. Cette notion de standardisation est sans aucun doute une des philosophies fondatrices de l'IDM et a conduit à des notions comme celles de généricité et de réutilisation de modèle. Le second besoin identifié est celui de la traçabilité entre les informations implémentées dans les modèles et leurs origines. Le troisième besoin est celui de posséder des modèles compositionnels. Ainsi, les chemins de propagation de défaillance ne sont plus représentés explicitement (comme sur les arbres de défaillance) mais sont définis à partir de compositions de composants élémentaires. Le quatrième et dernier besoin est un besoin d'outils permettant de supporter et d'automatiser la réalisation d'analyses de SdF.

En vue de satisfaire les besoins évoqués, divers travaux (Joshi, 2006) ont montré qu'il était possible d'utiliser l'IDM pour réaliser les études de SdF. L'idée directrice est alors de se munir d'un modèle formel des propagations de défaillance du système à étudier et d'utiliser ce modèle pour réaliser les études de SdF du système (on parlera alors et notamment de génération automatique d'arbre de défaillance). Parmi les différents langages disponibles, le langage AltaRica a été adopté et sera présenté dans une section postérieure.

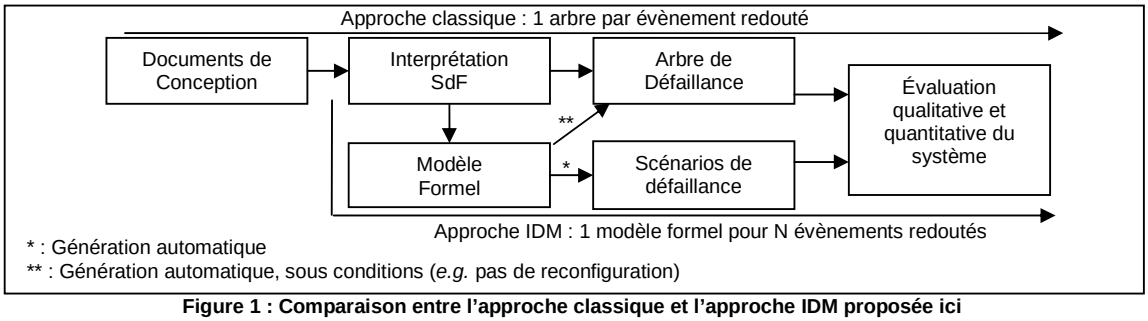

Figure 1 : Comparaison entre l'approche classique et l'approche IDM proposée ici

3 D'autres langages / méthodes / outils pour la réalisation d'études SdF

Les arbres de défaillance (et leurs équivalents les diagrammes de fiabilité) sont sans nul doute possible une des méthodes les plus répandues lorsqu'on parle d'analyses SdF de systèmes. Comme déjà évoqué, cette méthode permet des analyses qualitatives et quantitatives de systèmes mais présente des limites d'utilisation lors de son application à des systèmes complexes. Pour dépasser ou tenter de dépasser ces limites, divers travaux ont été menés pour automatiser la réalisation des études de SdF. Ces approches ont pour fondement commun une description préalable du système (la forme de cette description est fonction de l'approche) permettant de tracer la propagation de défaillance entre les composants du système. Si les travaux présentés plus loin dans ce papier utilisent une description du système en langage AltaRica, nous présentons ici d'autres travaux ayant pour vocation l'évaluation des propagations de défaillance dans un système.

Pour commencer, des travaux s'intéressent à l'utilisation de Réseaux de Petri (RdP) stochastiques pour réaliser des analyses qualitatives et quantitatives de systèmes (par exemple, (Rugina 2006) s'intéresse à l'utilisation de tels RdP et d'AADL). Brièvement, un RdP représente le système sous une vue état / transition et est purement qualitatif. Un RdP stochastique permet l'affectation de probabilités aux passages des transitions. Ces RdP stochastiques permettent donc les analyses qualitatives permises par les RdP simples (notamment identifier si le système peut se trouver dans un état indésirable) ainsi que la réalisation de simulation de Monte Carlo pour évaluer quantitativement le système (probabilité d'arriver dans un état particulier du système). Si les RdP et les RdP stochastiques sont bien adaptés à la modélisation de systèmes industriels (par exemple, pour des systèmes soumis à une politique de maintenance), une des difficultés majeures est liée à la construction des modèles. L'analyste doit à tout moment prendre garde au problème d'explosion combinatoire et doit donc gérer le compromis expressivité – taille du modèle. L'usage d'une telle méthode est donc souvent limité à des systèmes de petites tailles.

D'autres travaux (Papadopoulos et al. 2001) présentent une tout autre approche à travers la méthode HIP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies). Cette méthode se propose de générer automatiquement un arbre de défaillance. Pour cela, un modèle hiérarchique est construit à partir d'une décomposition du système sous forme de schémas blocs successifs. Les modes de défaillance (conséquence observable de la défaillance, par exemple dans un circuit électrique, un court circuit ou un circuit ouvert) de chaque « bloc » identifié sont étudiés et introduit dans un tableau représentant comment ces modes de défaillance se propagent à leurs environnements. Un algorithme est ensuite appliqué pour générer l'arbre de défaillance correspondant à un évènement redouté. HIP-HOPS est supporté par un outil supportant des analyses qualitatives et quantitatives de systèmes.

Pour finir, le langage Figaro (Bouissou et al. 1991) a été développé par EDF pour modéliser la propagation des défaillances au sein d'un système. Son objectif est similaire à celui du langage AltaRica : Figaro est orienté vers la création de bases de connaissance décrivant des classes de composant ; chacune de ces classes inclut le comportement nominal et dysfonctionnel du composant qu'elle représente ; le système global est ensuite construit par instanciation de composant. Figaro est supporté par l'outil KB3 contenant un éditeur graphique de modèles ainsi que différents algorithmes permettant d'évaluer qualitativement et quantitativement le modèle (contrairement à AltaRica, Figaro est plus tourné vers le quantitatif que vers le qualitatif). Dans (Bouissou et al. 2006), les langages Figaro et AltaRica sont comparés : bien qu'ayant un objectif commun, ils diffèrent profondément dans leurs philosophies et leurs utilisations ; AltaRica, plus simple, est destiné aux analystes en charge des études de système ; Figaro, plus sophistiqué, est plus adapté à qui veut concevoir des outils de modélisation « clef en main » (une fois la base de connaissance crée, la réalisation de modèle n'imposera l'écriture d'aucune ligne de code Figaro).

Le langage AltaRica

Le langage AltaRica (Arnold et al. 2000) est un langage de modélisation conçu au LaBRI (Laboratoire Bordelais de Recherche en Informatique), fondé sur la notion d'automates à contraintes, et qui permet de modéliser formellement le comportement de système en présence de défaillances. Son caractère formel, hiérarchique et compositionnel lui confère une capacité à étudier des systèmes complexes et à en faire les analyses. Aussi, le langage est dit évènementiel puisque la dynamique d'un modèle n'est pas liée au temps mais est régie par l'occurrence d'évènements internes au système. AltaRica intéresse de nombreux industriels (Airbus, Dassault, Turbomeca) et est doté de nombreuses variantes. Parmi celles-ci, citons en une très fortement inspirée de AltaRica Data-Flow (Rauzy 2002) et supportée par l'outil Cecilia OCAS. C'est de cette version qu'il s'agira dans la suite de ce papier lorsque nous utiliserons le terme AltaRica.

En AltaRica, la description d'un système met en œuvre différentes étapes parmi lesquelles : la description des différents éléments composant le système, la description des interactions entre ces composants ou la description des hiérarchies à l'intérieur du système (*i.e.* notion de système / sous-système). Dans cette optique, la construction d'un modèle met en jeu la création de plusieurs unités élémentaires que l'on appellera *nœud*. Ainsi, la définition d'un nœud contient une partie statique avec la définition des champs suivants :

- *flow* : permet la déclaration des variables d'entrée / sortie (E/S) et de leurs types (principalement booléen ou énuméré). Ces variables représentent les informations échangées entre le nœud et son environnement (ou réciproquement) ;
- *state* : permet la déclaration des variables d'état interne du nœud et de leurs types (principalement booléen ou énuméré). L'état courant du nœud est une valuation de ces variables d'états ;
- *event* : permet la déclaration de l'ensemble des évènements provoquant un changement d'état ; il est possible d'associer à chacun de ces évènements une loi de probabilité (*e.g.* Exponentielle ou instantanée) ;

La partie dynamique du nœud est définie quant à elle grâce aux champs :

- *init* : permet d'assigner une valeur initiale aux variables d'états du nœud ;
- *trans* : permet la description des changements d'état suite à l'occurrence d'un évènement. Une transition se formalise sous la forme « *G(s, v)* |− *E* -> *s_* » où *G* est une condition booléenne sur les variables d'état *s* et les variables d'entrée *v* du nœud, *E* est l'évènement modifiant l'état du nœud si sa configuration vérifie *G*, *s_* décrit la nouvelle valuation des variables d'état du nœud ;
- *assert* : permet de définir les valeurs des variables de sortie du nœud en fonction de celles des variables d'entrée et des variables d'état.

Étant donnée la description du langage ci-dessus, modéliser en AltaRica consiste alors à identifier les composants du système (*i.e.* les nœuds), leurs interfaces (variables d'E/S) ainsi que leurs comportements fonctionnels et dysfonctionnels.
Chacun des nœuds modélise un composant du système. Celui-ci est composé de variables de flux et de variables d'état dont le type permet la représentation du niveau de détail modélisé (on utilisera parfois dans cet article le terme granularité pour parler de ce niveau de détail). En effet, AltaRica ne permettant pas la représentation directe de variables continues, il nous faut extraire du domaine de définition de ces variables certaines classes de valeur permettant la propagation correcte des défaillances et l'observation des évènements redoutés. Cette question est un point clef dans le processus de modélisation AltaRica et sera traitée en détail dans une prochaine section. Au niveau du comportement modélisé à l'intérieur du composant : les variables d'état représentent les modes de fonctionnement et de dysfonctionnement du composant ; les évènements modélisent les défaillances ou reconfigurations qui provoquent, à travers les transitions, l'évolution des variables d'état ; les assertions décrivent le calcul de la valeur des variables de sortie en fonction de celles d'entrée et de celles d'état. Une fois tous les composants modélisés au sein d'une bibliothèque, l'architecture globale du modèle est constituée d'instances de ces composants interconnectées via leurs E/S.

Pour illustrer la construction d'un nœud AltaRica, prenons l'exemple d'un tuyau dont le but est d'amener de l'eau d'un point A à un point B. Ici, le cas est simplifié à l'extrême puisque : nous considérerons l'eau comme étant présente ou absente, le seul mode de défaillance considéré est la fuite (ici, analogue à une rupture de tuyau). Pour modéliser ce composant, on modélise donc une variable d'entrée *IN* et une variable de sortie *OUT*, toutes deux booléennes (sans originalité, Vrai = eau présente / Faux = eau absente). On considère que le tuyau a deux états : un état de fonctionnement nominal dans lequel l'eau présente en A est conduite jusqu'en B, un état de dysfonctionnement représentant l'état du tuyau après l'occurrence de la fuite (et qu'il y ait de l'eau en A ou pas, il n'y aura pas d'eau en B). Le nœud contiendra donc une unique variable d'état *ST* qui prendra ses valeurs dans l'ensemble {ok, ko}. L'évènement modélisant une fuite du tuyau sera appelé *Leakage*. Fondé sur ces informations, le code AltaRica du composant *Tuyau* est décrit ci-dessous.

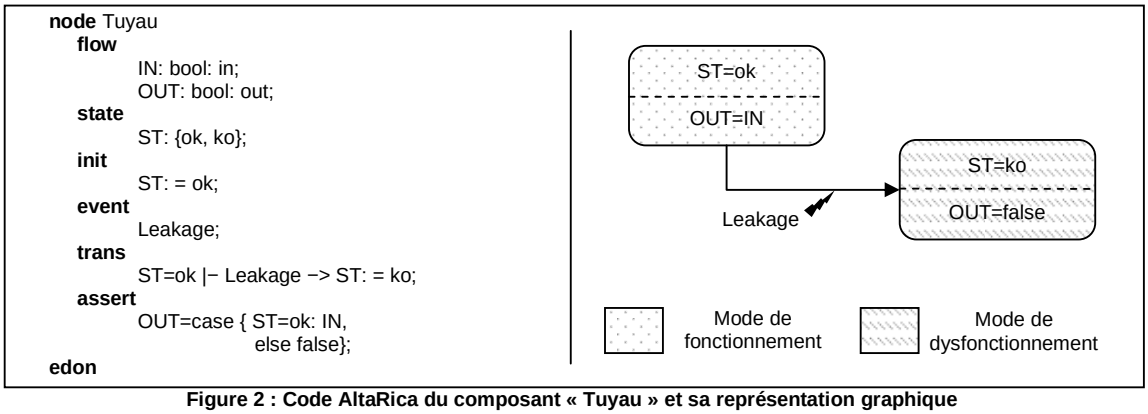

Figure 2 : Code AltaRica du composant « Tuyau » et sa représentation graphique

Spécification informelle de modèle AltaRica

Différentes approches, déjà disponibles dans la littérature, traitent de la modélisation en langage AltaRica de systèmes logiciels, électriques ou encore hydrauliques (Kehren et al. 2004) ; peu traitent de l'unification de ces approches pour la modélisation de systèmes physiques. Le but de l'approche présentée dans cette partie est de produire, de manière systématique, une spécification informelle de ce que devra être le modèle AltaRica, *i.e.* identifier ses objectifs, les évènements qu'il devra propager et les informations que celui-ci devra contenir pour permettre cette propagation. Pour cela, l'approche présentée se propose 1) d'identifier l'information à extraire du système considéré et 2) de définir des abstractions permettant de propager cette information. Si le lecteur informé pourra reconnaître (Humbert 2006) dans les fondations de la démarche proposée, l'originalité de nos travaux se situe selon nous dans son application à des systèmes mécaniques et hydromécaniques.

Caractérisation des objectifs du modèle : analyse préliminaire

(Donnadieu et al. 2003) décrit la modélisation comme *« un processus technique qui permet de représenter un objet dans un but de connaissance »*. Cette définition étant prise en compte, il serait sans doute déraisonnable et hasardeux de commencer notre approche sans préciser les objectifs (*i.e.* le but) du modèle souhaitant être construit. De manière simple, on exprimera dans un premier temps ces objectifs en précisant que le modèle devra permettre l'étude des évènements redoutés (ER) considérés et que pour cela, son périmètre devra couvrir le système d'étude (prendre en compte l'ensemble des comportements ayant un impact sur l'occurrence des ER). Pour expliciter davantage ces objectifs, le modèle se doit de modéliser les propagations de défaillance à l'intérieur du système considéré : les dépendances fonctionnelles et physiques entre composants devront ainsi être identifiées ; une attention particulière devra également être donnée à l'identification des mécanismes de détection et de tolérance aux défaillances (ces mécanismes devant ensuite et bien entendu être inclus dans le modèle). Également, les ER considérés devront pouvoir être observés par le modèle.

Par conséquent et préalablement à la modélisation, nous proposons de nous intéresser à l'identification de trois informations : 1) le périmètre du modèle et son architecture (*i.e.* l'identification des composants à modéliser), 2) l'expression des ER sur le système d'étude considéré (*i.e.* décliner si besoin les ER de haut niveau au système d'étude) et 3) le comportement fonctionnel et dysfonctionnel de chacun des composants identifiés.

Pour rentrer sans plus attendre dans le vif du sujet, l'approche commence par une analyse préliminaire du système d'étude pouvant être présentée comme une succession de plusieurs sous-étapes :

- analyse fonctionnelle du système ;
- décomposition du système global en sous-systèmes physiquement homogènes ;
- identification des interfaces entre ces sous-systèmes ;
- identification des évènements redoutés (ER) système et leurs exigences respectives ;
- déclinaison des ER système (et leurs exigences) aux sous-systèmes ;
- pour chaque sous-système, identification des composants ayant un impact sur l'occurrence des ER considérés ;
- pour chaque sous-système, identification des interfaces entre ces composants ;
- pour chaque composant, identification des modes de défaillance et des évènements nominaux potentiels.

Clarifions maintenant comment les résultats de cette analyse préliminaire sont utilisés pour spécifier les données à inclure dans le modèle.

Caractérisation de l'architecture globale du modèle

Dans un modèle AltaRica, le choix des composants à inclure dans le modèle est crucial dans le sens où il détermine le périmètre et la granularité (i.e. le niveau de détail) du modèle. En accord avec l'analyse préliminaire ci-dessus, la démarche débute par une analyse fonctionnelle (typiquement, de philosophie similaire à SADT). Lorsque le système peut être décrit comme une hiérarchie de sous-systèmes et de composants, la structure du modèle reflètera autant que possible la structure du système. *A contrario*, lorsque le système ne présente que peu ou pas de hiérarchie, une option intéressante à ce stade est d'identifier des sous-systèmes d'étude en groupant des ensembles de composant d'un même domaine physique (comme le sera présenté dans de prochaines sections, la propagation de défaillance entre composants d'un même domaine physique peut se faire en propageant un ensemble de grandeurs spécifiques au domaine).
Toujours au niveau du choix des composants à inclure dans le modèle, ce choix peut être raffiné dans le but de rendre explicite certaines fonctions réalisées par un composant (ainsi, si un composant réel a deux fonctions, il est possible de modéliser chaque fonction par un composant AltaRica). Au contraire et pour limiter la taille du modèle, il peut être parfois bénéfique de grouper un ensemble de composants réels en un unique nœud AltaRica (par exemple, lorsqu'un ensemble de composants est pris en compte pour le bon fonctionnement du modèle mais n'impacte pas l'occurrence de l'ER). Dans tous les cas, les choix de modélisation devront être justifiés (majoritairement, grâce à l'analyse fonctionnelle et à l'AMDE) et tracés.

Ensuite, pour chaque composant identifié comme étant à modéliser, le choix des ports d'entrée-sortie (E/S) se doit de refléter et de modéliser les dépendances fonctionnelles et physiques entre le composant et son environnement. Nous verrons dans la suite de ce papier que l'analyse fonctionnelle nous permet d'identifier des ports d'E/S de haut niveau qui seront ensuite raffinés selon le domaine physique considéré.

Caractérisation des objectifs du modèle : Évènements Redoutés (ER) à observer

Un autre résultat de l'analyse préliminaire réalisée est la caractérisation des ER à observer sur le modèle. Il nous faut donc 1) les exprimer sur le périmètre du modèle puis 2) les observer sur le modèle. La première étape décline les ER système au sous-système considéré. Une fois cela fait, l'observation des ER passe, dans le modèle AltaRica, par la création de nœuds appelés *observateurs*. Ces nœuds sont similaires à des nœuds AltaRica classiques. Cependant, s'ils possèdent des variables d'E/S, ils ne possèdent en général pas d'évènement et donc pas de variable d'état. Typiquement, un tel observateur possèdera N variables d'entrée (N>0) et une unique variable de sortie booléenne représentant l'occurrence, ou non, de l'ER. Son unique assertion permettra de modéliser l'occurrence de l'ER : la variable de sortie sera vraie si les N variables d'entrées prennent une certaine combinaison de valeur.

Caractérisation d'un composant unitaire

Une fois l'architecture et le but du modèle identifiés, nous souhaitons à présent 1) caractériser le comportement de chacun des composants identifiés comme étant à modéliser et 2) définir des abstractions permettant de modéliser ce comportement. Pour cela, il nous est nécessaire d'identifier :

- les comportements fonctionnels : identifier les évènements ayant attrait au fonctionnement nominal du composant ainsi que les modes de fonctionnement nominaux (*i.e.* les états de bon fonctionnement) ;
- les comportements dysfonctionnels : identifier les modes de défaillance du composant ainsi que ses modes de dysfonctionnement (*i.e.* les états que le système peut atteindre après l'occurrence d'une défaillance) ;
- les lois de propagation à l'intérieur du composant (comment réagit le composant à une entrée non nominale ? Comment propage t-il l'information à son tour ?).

Les évènements relatifs au fonctionnement nominal du composant et les modes de fonctionnement (état de fonctionnement nominal du composant) sont identifiés grâce aux différents documents de conception disponibles (par exemple, la spécification du composant). Ici, il s'agira par exemple de modéliser l'ouverture (ndlr : souhaitée) d'une valve ou le changement de configuration d'utilisation du système.

Les modes de défaillance et les modes de dysfonctionnement (état de fonctionnement non nominal du composant) sont identifiés à partir de l'analyse fonctionnelle du composant et de l'AMDE. La première, en décrivant les fonctions du composant, fournira les modes de défaillance de haut niveau (par négation des fonctions devant être remplies); la seconde fournira les modes de défaillance organiques. Cependant, bien que plus exhaustive dans la description des modes de défaillance, l'AMDE est excessivement détaillée pour être candidate à un export de l'ensemble de ces modes de défaillance dans le modèle AltaRica. En effet, deux de ces modes de défaillance peuvent avoir une conséquence fonctionnelle identique. Ainsi, le modèle AltaRica sera muni, autant que possible, de modes de défaillance de haut niveau ; chacun d'entre eux pouvant être mis en correspondance avec au moins un mode de défaillance issu de l'AMDE.

Concernant la modélisation des propagations de défaillance, nous nous devons d'identifier comment le composant réagit à une défaillance de son environnement, *i.e.* nous devons modéliser la réaction de ce composant à la défaillance ainsi que la façon dont il propage l'information à son tour (propagation directe, mitigation de la défaillance, envoi d'un signal d'alarme si détection…).

Propager l'ensemble de ces comportements revient à être capable, pour le composant, de « lire » l'information provenant de son environnement et de « transmettre » l'information à ce même environnement. Pour cela, il nous faut définir des abstractions suffisamment expressives permettant de propager les informations dans le modèle (*i.e.* définir des abstractions pour les ports d'E/S). Dans cette optique et comme déjà explicité, nous utilisons l'analyse fonctionnelle pour identifier des ports d'E/S de haut niveau (mouvement, puissance, fluide, courant électrique…). Cependant, si le concept d'E/S orientées, propre à AltaRica (DataFlow OCAS), s'adapte bien aux systèmes logiciels (les composants ont des E/S bien définies), il est plus difficilement applicable aux systèmes électriques, mécaniques ou hydrauliques (ou en général, il n'y a pas de causalité claire entre ce qui pourrait être une entrée et une sortie). Pour identifier les E/S de composants appartenant à de tels domaines physiques, nous choisissons de raffiner ces E/S selon le domaine physique considéré. Ce raffinement se fait par l'intermédiaire d'un ensemble de grandeurs physiques dont le domaine de définition (ndlr : la plupart du temps énuméré) est fonction des évènements redoutés à observer (*i.e.* des objectifs du modèle) et des évènements à propager. Parfois et pour propager convenablement les évènements, la propagation se devra d'être bidirectionnelle (dans un système physique, souvent, « l'entrée » influe sur la sortie et réciproquement). Aussi et selon le but du modèle, il est possible de propager la valeur d'une grandeur physique et/ou sa qualité (correcte ou erronée).

AltaRica étant fondé sur la notion d'automate de mode (Rauzy 2002), il convient ordinairement de faire un compromis entre expressivité et complexité afin d'éviter un temps de calcul trop important. La plupart du temps, une solution acceptable est de s'assurer que les valeurs manipulées (*i.e.* le domaine de définition des variables manipulées) soient nécessaires et suffisantes à l'observation des ER considérés.

Une fois les évènements et les ports d'E/S AltaRica identifiés, on s'intéresse à la dynamique du composant, *i.e.* les transitions et les assertions. En ce qui concerne les transitions, il faut s'intéresser 1) aux conditions sous lesquelles les évènements peuvent avoir lieu et 2) au potentiel caractère transitoire des évènements. Les premières définiront les gardes des transitions AltaRica (par exemple, un évènement peut n'être tirable que dans un certain état) et permettront, entre autres choses, de modéliser des pannes en cascade. Pour le second, on identifie si l'évènement est permanent ou transitoire. Dans le cas d'évènements transitoires, on ajoute au modèle un évènement « inverse ».

// Transition – Évènement transitoire	*// Transition – Évènement « inverse »*
ST=ok |- Evènement_transitoire -> ST:= transitoire;	ST:= transitoire |- Evènement_inverse -> ST:=ok;

Enfin, pour spécifier les assertions AltaRica, on réalise différentes tables de décision où les valeurs des variables de sortie sont définies en fonctions de celles des variables d'entrée et de l'état interne du composant. Ces tables de décisions sont écrites en se fondant sur les effets décrits dans l'AMDE ainsi que sur sa propre compréhension du système.

Les différentes étapes ci-dessus nous permettent, pour chacun des composants à modéliser, d'identifier l'ensemble des informations qui devront être *a priori* implémentées dans le modèle. C'est en ce sens que nous appelons cela une « spécification informelle » du modèle AltaRica (informelle puisqu'il s'agit pour l'instant uniquement de méthodes et/ou de conseils). En utilisant ensuite et par exemple, l'atelier Cecilia OCAS de Dassault Aviation, il est possible de construire, à partir de l'ensemble de ces spécifications informelles, une bibliothèque de modèle de composant.

Non présenté ici mais étant selon nous une problématique intéressante, nous nous intéressons également à la formalisation de cette « spécification informelle ». Le double but avoué étant de 1) faciliter l'obtention du modèle AltaRica et 2) mettre en place une méthode de validation de la bibliothèque de composants obtenue (Adeline et al. 2010).

Cas d'étude

Dans un souci de clarté et de lisibilité, le cas d'étude proposé est inspiré de la réalité mais n'en reprend pas toute la complexité. Dans cette section, nous présentons le cas d'étude sur lequel nous appliquerons la méthodologie proposée à la section précédente ainsi que les évènements redoutés à étudier sur ce cas d'étude.

En phase de vol, il est du domaine du possible qu'un turbomoteur (moteur d'hélicoptère) soit potentiellement et exceptionnellement soumis à une phase de survitesse (*i.e.* le moteur tourne à une vitesse supérieure à la normale). La raison peut être d'origines diverses : mécanique en cas de rupture d'arbre de transmission, hydromécanique si la quantité de carburant envoyée est supérieure à la consigne, logicielle si cette même consigne est calculée de manière erronée, ou bien opérationnelle en cas d'action pilote inappropriée par exemple. Dans tous les cas et quelle qu'en soit l'origine, l'effet de cette survitesse sur le moteur doit être limité. Pour cela, un turbomoteur possède différentes protections possibles, du blindage (protection mécanique) à la coupure électronique survitesse dont le but est de détecter cette survitesse puis d'éteindre le moteur « proprement », *i.e.* sans autre effet.

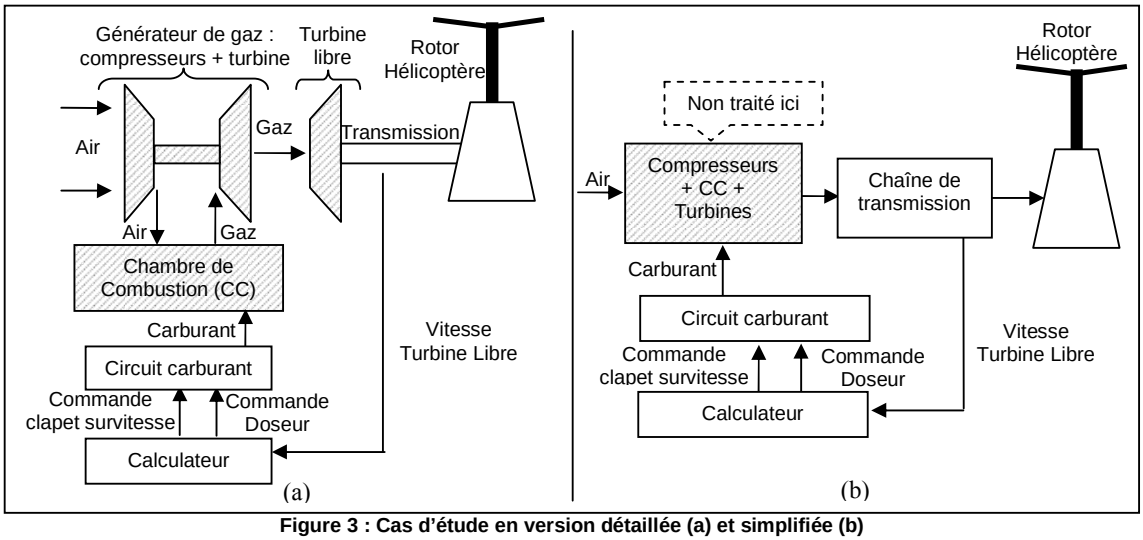

Figure 3 : Cas d'étude en version détaillée (a) et simplifiée (b)

Description du cas d'étude

Pour décrire notre cas d'étude, nous nous intéressons à sa version simplifiée présentée sur la Figure 3-b. Trois sous-systèmes nous intéressent particulièrement : l'ensemble {chaîne de transmission}, le circuit carburant et le calculateur. Ici, l'ensemble constitué des compresseurs, de la chambre de combustion et des turbines est uniquement utilisé pour la compréhension du cas d'étude mais ne sera pas traité dans ce papier.

L'ensemble {chaîne de transmission} est constitué de composants mécaniques de type engrenages ou roulements. Son rôle principal est de transmettre la puissance mécanique de la sortie de la turbine libre au rotor de l'hélicoptère. Deux capteurs, aussi présents dans cet ensemble, permettent de mesurer la vitesse de la turbine libre et de transmettre cette vitesse au calculateur.

Le calculateur traite les différentes mesures provenant du moteur (dont la mesure de vitesse évoquée ci-dessus) afin de 1) élaborer la consigne de débit carburant à injecter dans le moteur en fonctionnement nominal et 2) de couper l'arrivée de carburant si une survitesse est détectée. Dans un tel cas, le calculateur commande l'ouverture d'un clapet d'arrêt dans le circuit carburant.

Le circuit carburant assure, entre autres, les fonctions de dosage du carburant et d'alimentation de la chambre de combustion. Brièvement, il s'agira d'injecter la juste quantité (consigne élaborée par le calculateur) de carburant dans la chambre de combustion afin que celle-ci produise les gaz entraînant la turbine à la vitesse voulue. C'est dans ce circuit carburant que se situe le clapet d'arrêt permettant de forcer le retour du carburant dans le réservoir, l'empêchant ainsi d'être injecté dans le moteur. Lorsque l'ouverture de ce clapet est commandée (par exemple, dans un cas de survitesse détectée), le moteur s'arrête.

Évènements redoutés à étudier

Nous nous intéressons ici particulièrement à l'observation de deux évènements redoutés. Ces deux évènements sont :

- l'occurrence d'une survitesse du moteur ;
- l'Arrêt en Vol Non Commandé (AEVNC) qui survient lorsque le moteur s'arrête de manière non souhaitée par le pilote (en particulier, un arrêt du moteur par la protection survitesse rentrera dans cette catégorie) ;

Ces deux évènements serviront de support dans les prochaines sections et nous permettrons d'illustrer l'approche proposée dans la partie précédente (Spécification informelle de modèle AltaRica).

Obtention d'une spécification informelle pour la modélisation AltaRica du cas d'étude

4 Analyse préliminaire du cas d'étude

4.1 Décomposition du système global en sous-systèmes d'étude

En accord avec la Figure 3 et avec la partie précédente, nous décomposons le cas d'étude proposé en trois sous systèmes d'étude : 1) un système mécanique : l'ensemble {chaîne de transmission} ; 2) un système hydromécanique : le circuit carburant ; et 3) un système logiciel : le calculateur.

Concernant les interfaces entre ces différents sous-systèmes, celles-ci ont d'ores et déjà été mentionnées dans la partie précédente et sont :

- l'ensemble {chaîne de transmission} transmet la vitesse de rotation de la turbine libre au calculateur ;
- le calculateur transmet 1) la consigne de quantité de carburant à envoyer dans la chambre de combustion et 2) la commande d'ouverture du clapet d'arrêt en cas de survitesse moteur détectée ;
- le circuit carburant injecte le carburant dans la chambre de combustion ;
- l'ensemble {Compresseurs + CC + Turbines} est présent pour la compréhension générale de l'ensemble et agit ici comme une « boite noire ». On retiendra qu'il reçoit du carburant du circuit carburant et qu'il transmet de la puissance mécanique à l'ensemble {chaîne de transmission}.

Les travaux présentés ici sont principalement tournés vers la modélisation formelle des propagations de défaillance au sein des systèmes mécaniques. Un retour d'expérience sur une telle modélisation au sein de systèmes hydromécaniques et logiciels sera proposé dans les prochaines sections.

4.2 Déclinaison des évènements redoutés système aux sous-systèmes d'étude

Il s'agit ici de décliner aux trois sous-systèmes identifiés les évènements redoutés système explicités dans la partie « Cas d'étude ». Pour rappel, il s'agit de 1) l'occurrence d'une survitesse moteur et 2) l'Arrêt en Vol Non Commandé. Le Tableau 1 a pour vocation de présenter la déclinaison de ces deux évènements redoutés aux trois sous-systèmes d'étude.

	Arrêt en Vol Non Commandé	Survitesse
Sous-système mécanique	Perte de la transmission	Survitesse de la transmission
Sous-système hydromécanique	Plus de carburant n'est injecté dans la chambre de combustion	Trop de carburant est injecté dans la chambre de combustion
Sous-système logiciel	La quantité de carburant à injecter calculée est trop faible (trop peu de carburant sera injecté pour maintenir le moteur allumé)	La quantité de carburant à injecter calculée est trop importante (trop de carburant sera injecté)

Tableau 1 : Déclinaison des évènements redoutés systèmes aux sous-systèmes d'étude

---

Il nous faut maintenant nous intéresser à la modélisation de chacun des composants identifiés comme étant à modéliser. Pour cela, nous particularisons l'étude en fonction du domaine physique considéré. Pourquoi ? Deux arguments étayeront ici notre choix : 1) les grandeurs physiques manipulées sont différentes d'un domaine à l'autre, 2) les évènements redoutés à observer sont de granularités différentes (dans le cas réel, plus complexe que le cas d'étude présenté ici, tous les évènements redoutés considérés n'ont pas leurs occurrences influencées par l'ensemble des sous-systèmes d'étude).

5 Spécification informelle du modèle du sous-système mécanique

5.1 Architecture du sous-système mécanique

L'idée générale est ici d'identifier, pour le sous-système d'étude considéré, l'architecture permettant de modéliser la propagation de défaillance à l'intérieur de ce sous-système. Dans cette optique, nous identifions les composants dont au moins un mode de défaillance a un impact sur au moins un des évènements redoutés (ER) considérés (puis, nous ne modéliserons dans le modèle du composant que les modes de défaillance ayant un impact sur au moins un des ER considérés ; ceux n'ayant d'impact sur aucun de ces ER ne seront pas modélisés). L'objectif à terme étant d'inclure dans le modèle les informations utiles à la représentation des propagations de défaillance dans le système et de justifier la non-inclusion des autres informations (*i.e.* justifier que l'on n'oublie rien d'utile dans le modèle).

Ainsi et sans davantage de détail ici, l'architecture du modèle du sous-système mécanique sera composée d'engrenages, de roulements (billes, rouleaux…), d'arbres de transmission ou encore de vis.
Une fois l'architecture identifiée, nous nous intéressons aux interfaces (fonctionnelles et physiques) entre les composants de ce sous-système. Pour cela, les différents documents disponibles comme les documents de conception (spécifications, coupes machines…) ainsi que l'analyse fonctionnelle réalisée lors de cette analyse préliminaire (si celle-ci est d'un niveau de détail suffisamment bas) sont utilisés. Ainsi, l'architecture du modèle est déterminée.

Dans la suite de cette section, nous supposerons connue la liste des composants à modéliser ainsi que la liste des évènements redoutés à considérer (*i.e.* les objectifs de nos modèles). Pour illustrer la démarche sur un système mécanique, nous prendrons l'exemple d'un arbre de transmission mécanique. La fonction, simple, de cet arbre et de transmettre un mouvement de rotation d'un point d'entrée à un point de sortie. Également à noter, cet arbre de transmission est guidé en rotation (ndlr : par un roulement de type supposé quelconque ici).

5.2 Une première spécification informelle des E/S : Analyse fonctionnelle de l'arbre de transmission

Nous commençons par une analyse fonctionnelle de l'arbre de transmission en vue d'identifier l'ensemble des interactions que celui peut avoir avec son environnement. Identifier ces interactions nous permettra ensuite de déterminer un premier ensemble

d'E/S dont l'objectif sera d'autoriser la modélisation des propagations de défaillance. Ces E/S seront ensuite et si besoin raffinées au cours d'une prochaine étape. Pour l'arbre de transmission, le résultat de l'analyse fonctionnelle est décrit Figure 5-a. Nous identifions donc trois entrées et deux sorties.

5.3 Spécification informelle du comportement interne du composant

Ici, il s'agit d'identifier deux informations : 1) les évènements qui seront à intégrer dans le modèle (*i.e.* les évènements relatifs au fonctionnement nominal du composant ainsi qu'à son dysfonctionnement, *i.e.* les défaillances) ; 2) les modes de fonctionnement et de dysfonctionnement ainsi que les transitions entre ces modes.

Comme déjà explicité dans la partie « Spécification informelle de modèle AltaRica », les évènements (relatifs au fonctionnement et au dysfonctionnement) du composant sont obtenus à partir de l'analyse fonctionnelle et de l'AMDE. Sans revenir sur l'obtention de ces évènements, ceux retenus ici seront, et on l'admettra, la perte de la transmission de mouvement par rupture de l'arbre et la production de particules métalliques dans l'huile (il n'y a pas d'évènement relatif au comportement nominal). On s'intéresse ici particulièrement au premier.

Une fois ces évènements identifiés, les modes de fonctionnement et de dysfonctionnement sont en mesure d'être déterminés. Ici et en relation avec les deux évènements retenus, nous avons :

- un unique mode de fonctionnement nominal, lorsque la transmission de mouvement est correcte et qu'aucune particule métallique n'est libérée dans l'huile ;
- 3 modes de dysfonctionnement : la perte de la transmission de mouvement sans émission de particule mécanique, l'émission de particules métalliques tout en conservant la transmission de mouvement, la combinaison des deux.

En combinant la connaissance des évènements et des modes de fonctionnement / dysfonctionnement, il est du domaine du possible d'écrire les différentes transitions entre ces modes (par exemple, on passe du fonctionnement nominal à la perte de transmission par l'évènement « Rupture »). Une représentation possible à ce stade est un diagramme état – transition UML classique (Figure 4).

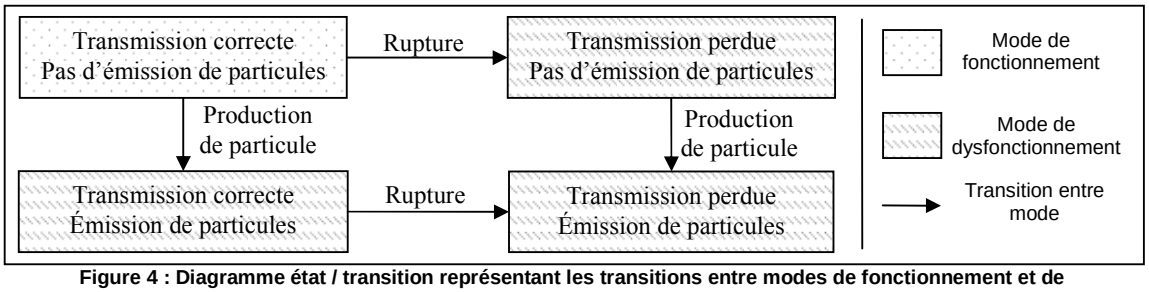

Figure 4 : Diagramme état / transition représentant les transitions entre modes de fonctionnement et de dysfonctionnement

5.4 Spécification informelle des évènements externes à propager

Le composant se doit de propager à l'environnement non seulement ses propres évènements (Production de particules et Rupture) mais aussi les évènements d'autres composants qui parviennent jusqu'à lui (si, par exemple, il y a déjà une défaillance dans le système). La propagation peut alors se résumer en deux questions : « comment informer l'environnement d'un évènement du composant ? » et « comment informer le composant d'un évènement de l'environnement ? ».

Pour cela, un inventaire des possibles modes de défaillance est fait dans le modèle. De cet inventaire sont extraits des modes de défaillance génériques. Ici et pour cet article, on considérera concernant la transmission de mouvement un unique mode de défaillance générique : la rupture de la transmission entre le moteur et l'hélicoptère. Aussi à propager pour modéliser notre arbre de transmission, une perte de guidage ainsi qu'une perte de lubrification (plus d'huile en entrée).

5.5 Spécification informelle des E/S raffinées

Ici, l'objet est d'identifier les E/S qui permettront la propagation des évènements (internes et externes au composant) de et vers l'environnement du composant. Notre approche a pour but de raffiner les E/S de haut niveau identifiées section 5.2 et de propager ces évènements par l'intermédiaire d'abstractions de grandeurs physiques. Pour cela, notre choix s'appuie sur l'AMDE qui décrit les modes de défaillance du domaine mécanique (typiquement une rupture) comme ayant des conséquences sur 1) le couple transmis et 2) la vitesse de rotation transmise. Ainsi, toujours selon l'AMDE, une défaillance peut avoir des effets sur le couple, la vitesse ou encore les deux simultanément. Il nous faut donc propager ces deux informations.
Aussi et compte tenu qu'un système mécanique est un système « continu », une défaillance de l'arbre n'a pas uniquement des conséquences sur les composants situés fonctionnellement en aval mais aussi sur les composants situés en amont (une rupture de l'arbre entrainera non seulement un arrêt de l'entraînement du rotor (si l'hélicoptère possède un unique moteur) mais conduira aussi à une perte du couple résistant conduisant à une survitesse de la turbine (et entraînera l'activation de la protection survitesse)). Par conséquent, il nous faut propager le couple de variable {couple mécanique, vitesse de rotation} de façon bidirectionnelle, *i.e.* en amont et en aval (Figure 5-b).

Au sujet de l'huile et rapidement, ici une propagation monodirectionnelle (de l'amont vers l'aval) suffit pour propager le couple de variable {présence d'huile, présence de particules dans l'huile}. Ces deux variables seront booléennes.

Au sujet du guidage, nous propageons la réalisation correcte de la fonction en propageant, de manière monodirectionnelle, une unique variable booléenne.

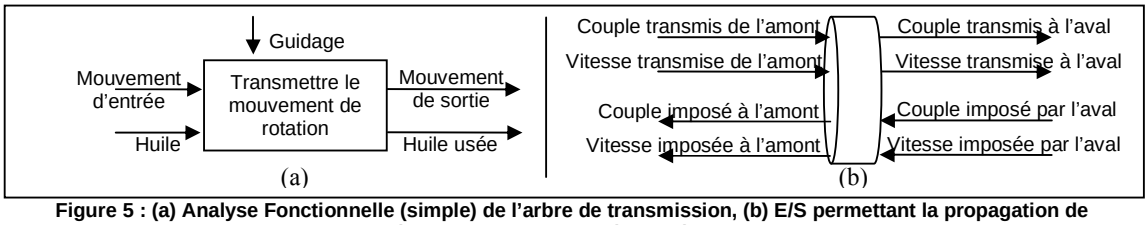

Figure 5 : (a) Analyse Fonctionnelle (simple) de l'arbre de transmission, (b) E/S permettant la propagation de défaillance dans un système mécanique

Nous nous intéressons maintenant à la détermination du niveau de granularité des variables « couple mécanique » et « vitesse de rotation ». Pour déterminer ce niveau de granularité, nous prenons le parti d'inclure les informations nécessaires et suffisantes à la propagation des défaillances considérées (rupture) et à l'observation des évènements redoutés à étudier. Ici et en adéquation avec le Tableau 1, nous souhaitons observer une perte de la transmission et une survitesse de cette même transmission. Nous choisissons donc comme niveau de granularité :

- {ok, nul, trop important} pour la variable couple ;
- {ok, nulle, survitesse} pour la variable vitesse.

Ensuite et à un niveau supérieur à celui de la modélisation du simple composant, il est alors possible de définir les modes de défaillance génériques en termes d'E/S (ou, d'un autre point de vue, de définir les effets de chaque mode de défaillance sur les E/S propagées dans le modèle). Il s'agira alors de propager dans le modèle des informations telles que « une rupture entraîne un couple transmis nul et une vitesse transmise nulle ». Ces effets peuvent être définis comme présenté sur le Tableau 2.

Mode Nominal		**Rupture de l'arbre**	
Sortie	*Valeur*	*Sortie*	*Valeur*
Couple transmis à l'aval	Couple transmis de l'amont	Couple transmis à l'aval	nul
Vitesse transmise à l'aval	Vitesse transmise de l'amont	Vitesse transmise à l'aval	nulle
Couple imposé à l'amont	Couple imposé par l'aval	Couple imposé à l'amont	nul
Vitesse imposée à l'amont	Vitesse imposée par l'aval	Vitesse imposée à l'amont	nulle

Tableau 2 : Modélisation d'un mode de défaillance courant du domaine mécanique

Ce travail permet alors de définir, avec la prise en compte du fonctionnement nominal (ici, s'il n'y a pas eu rupture, le fonctionnement nominal se résume à « sortie = entrée », *i.e.* pas de mécanisme de mitigation de défaillance), la valeur des sorties dans chaque mode de fonctionnement / dysfonctionnement.

5.6 Bilan sur la spécification informelle de modèles de systèmes mécanique

D'une manière générale et la remarque est valable pour l'ensemble du travail présenté ici, on rappelle ici que doivent être propagées les grandeurs permettant de propager les défaillances du système. Le travail relaté ici et les résultats présentés sont donc à mettre en relation avec le cas d'étude considéré (système + ER considérés) mais ne peut être appliqué directement.

Composants type	Roue dentée : Élément d'un train d'engrenage Roulement : permet le positionnement, la transmission des efforts et la rotation entre deux pièces par le remplacement du glissement en un roulement Arbre de transmissions : transmet le mouvement de rotation et le couple Vis : Réalise la fixation d'une ou plusieurs pièces			
Évènement à propager	Rupture			
Grandeurs modélisées	Couple	bidirectionnelle	Ok, nul, trop important	Dans le système réel, la granularité est plus large pour observer davantage d'ER
	Vitesse	bidirectionnelle	Ok, nulle, survitesse	
	Huile	monodirectionnelle	booléenne	
	Particules	monodirectionnelle	booléenne	

Tableau 3 : Bilan sur la spécification informelle du système mécanique

Nous avons ainsi modélisé dans le cas réel deux sous-systèmes mécaniques : un système comportant 20 composants appartenant à 11 classes différentes ; un second comportant 36 composants de ce 11 mêmes classes. Sur ces sous-systèmes, deux évènements redoutés ont été étudiés. Par rapport aux arbres de défaillance, les résultats qualitatifs obtenus sont bons, validés par jugement d'expert et par comparaison avec ceux issus des arbres. Les résultats quantitatifs sont identiques si l'on s'assure que les périmètres modèle – arbre sont bien équivalents.

6 Bilan sur la spécification informelle de modèle de sous-système hydromécanique

L'approche d'obtention des informations étant similaire à la section précédente pour les sous-systèmes mécaniques, on ne présente ici qu'un récapitulatif des résultats obtenus (Tableau 4).

Composants type	Pompe : Fournir du fluide sous pression Filtre : Protéger le circuit des particules		Valve : distribuer le fluide Tuyau : transporter le fluide	
Évènement à propager	Il s'agira principalement d'évènements similaires à des fuites (de différentes sévérités) et des colmatages (de différentes sévérités pouvant aller jusqu'à l'obturation)			
Grandeurs modélisées	Débit	bidirectionnelle	Ok, nul, trop important	Dans le système réel, la granularité est plus large pour observer davantage d'ER
	Pression	bidirectionnelle	Ok, nulle, surpression	
	Température	monodirectionnelle	Ok, trop importante	

	Comportement dynamique	monodirectionnelle	Régulation, débit gelé, oscillation

Tableau 4 : Bilan sur la spécification informelle du système hydromécanique

En suivant cette méthodologie, nous avons modélisé dans le cas réel un sous-système hydromécanique comportant 31 composants appartenant à 26 classes différentes. Sur ce sous-système, trois évènements redoutés ont été étudiés. Par rapport aux arbres de défaillance, les résultats qualitatifs obtenus sont bons, validés par jugement d'expert et par comparaison avec ceux issus des arbres. Les résultats quantitatifs sont identiques si l'on s'assure que les périmètres modèle – arbre sont bien équivalents.

7 Quelques mots sur les systèmes logiciels

Cette section décrit le retour d'expérience provenant du travail présenté dans (Humbert 2006) et dans (Castel et al. 2002). Le domaine logiciel est relativement différent des deux autres puisque, si nous pensons l'approche présentée dans les sections précédentes applicable aux systèmes logiciels, l'étude est faite uniquement au niveau fonctionnel et l'information propagée est généralement limitée à la qualité des données échangées. Des travaux déjà effectués nous tirons deux règles principales :

- trois modes de défaillance génériques (perte de la fonction, fonction intempestive et réalisation erronée de la fonction) peuvent être définis pour une erreur logicielle (Castel et al. 2002) ;
- une erreur logicielle n'a de conséquence que sur les composants utilisant spécifiquement les résultats de la fonction ; ainsi, l'information (*i.e.* si le signal est correct, erroné ou perdu) doit l'être uniquement aux composants étant fonctionnellement en aval de celui considéré.

Ainsi, dans un système logiciel, l'information propagée l'est par une variable monodirectionnelle de domaine de définition {correct, erroné, perdu} représentant la qualité du signal transmis par une fonction à une autre.

Conclusion

L'objectif des travaux présentés ici est de proposer une approche permettant l'obtention de spécifications informelles de modèles AltaRica. Fondée et inspirée de la méthodologie de modélisation proposée par (Humbert 2006), l'approche proposée ici est applicable à une gamme plus riche de systèmes (mécaniques et hydromécaniques) et définit, pour ces systèmes, de nouvelles abstractions permettant la modélisation des propagations de défaillance en leurs seins.

La démarche proposée permet donc de « préparer » la modélisation AltaRica des systèmes considérés. Un de ses avantages est d'améliorer la traçabilité entre le système réel et ce que sera le modèle. Cependant, un des problèmes majeurs et récurrent d'une telle approche se situe au niveau de la validation des modèles obtenus. Aujourd'hui, la représentativité des modèles AltaRica effectués a été établie en comparant les résultats obtenus à partir des modèles avec ceux extraits des AMDE et des arbres de défaillance existants chez Turbomeca. Dans un processus, où les modèles viendraient à remplacer les arbres, des moyens complémentaires de validation doivent être trouvés. En l'état, cette validation ne peut être autre qu'une vérification informelle de la cohérence entre les informations implémentées et les informations présentées ici. Pour cette raison, nous nous intéressons très fortement à la formalisation de cette spécification informelle sous une forme suffisamment expressive, lisible et compréhensible pour permettre une validation par jugement d'expert (la Figure 4 fait partie de cette réflexion). Une fois cette spécification formelle (ou au minimum semi-formelle) obtenue, le problème de la validation est transformé en un problème de vérification entre une implémentation et sa spécification. Ces travaux sont présentés dans (Adeline et al. 2010).

8 Références

R. Adeline, J. Cardoso, P. Darfeuil, S. Humbert, C. Seguin, Toward a validation process for Model Based Safety Analysis, ERTSS 2010, Toulouse, 2010.

A. Arnold, A. Griffault, G. Point, and Antoine Rauzy, The Altarica formalism for describing concurrent systems, Fundamenta Informaticae, 40 :109–124, 2000.

M. Bouissou, H. Bouhadana, M. Bannelier, N. Villatte, Knowledge modelling and reliability processing: presentation of the Figaro language and associated tools Safecomp'91, Trondheim (Norvège), novembre 1991.

M. Bouissou, C. Seguin, Comparaison des langages de modélisation AltaRica et Figaro, Lambda Mu 15, Lille, 2006.

C. Castel, C. Seguin, Modèles formels pour l'évaluation de la sûreté de fonctionnement des architectures logicielles d'avionique modulaire intégrée, AFADL : Approches Formelles dans l'Assistance au Développement de Logiciels, 2002.

G. Donnadieu, D. Durand, D. Neel, E. Nunez, L. Saint Paul, L'approche systémique : De quoi s'agit-il ?, Synthèse des travaux du groupe AFSCET, Diffusion de la pensée systémique, 2003.

S. Humbert, C. Castel, C. Seguin, Y. Dutuit, J.-M. Bosc, P. Darfeuil, Méthodologie de modélisation AltaRica pour la Sûreté de Fonctionnement d'un système de propulsion d'un hélicoptère incluant une partie logicielle. Lambda Mu 15, Lille, 2006.

G. Hutzler, 2000, Du jardin des hasards au jardin des données : une approche artistique et multi-agent des interfaces hommes / systèmes complexes, thèse de doctorat, Université Pierre et Marie Curie, Paris 6, janvier 2000.

A. Joshi, M. Whalen, M. Heimdahl, Model-based safety analysis final report, NASA contractor report, NASA/CR-2006-213953, 2006.

C. Kehren, C. Seguin, P. Bieber, C. Castel, C. Bougnol, J.-P. Heckmann, S. Metge, Advanced Multi-System Simulation Capabilities with AltaRica, International System Safety Conference, 2004.

J. Nizet, J. Pichault , Introduction à la théorie des configurations. Du "one best way" à la diversité organisationnelle, De Boeck Université, 2001.

Y. Papadopoulos, J. McDermid, R. Sasse, G. Heiner, Analysis and synthesis of the behaviour of complex programmable electronic system in condition of failure, Reliability Engineering and System Safety, 229-247, 2001.

SAE, ARP4754: Certification Considerations for Highly Integrated or Complex Avionics Systems, Society of Automotive Engineers, 1996.

SAE, ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, Society of Automotive Engineers, 1996.

A. Rauzy. Mode automata and their compilation into fault trees. Reliability Engineering and System Safety, 78 :1–12, 2002.

A.-E. Rugina, K. Kanoun, M. Kaaniche, Modélisation de la sûreté de fonctionnement de systèmes à partir du langage AADL, Lambda Mu 15, Lille, 2006.