SYNTHESE DE COUPES MINIMALES FONCTIONNELLES EN COUPES MINIMALES COMPOSANT

SYNTHESIS OF MINIMAL FUNCTIONAL CUTSETS AS MINIMAL COMPONENT CUTSETS

André Leblond
Thales Avionics - 18 avenue du Maréchal Juin BP 49 - 92362 Meudon-la-Forêt Cedex +33 (0) 1 39 45 52 82
andre-l.leblond@fr.thalesgroup.com

Résumé

La construction de la safety pour un système complexe est une tâche qu'il est préférable de commencer lors des étapes initiales de définition de ce système. L'impact d'une correction d'erreur sur les coûts et les délais est en effet d'autant plus grand que cette erreur est détectée plus tard. Par ailleurs, les pratiques courantes du métier sont fondées sur des tâches manuelles, fortement sujettes à erreur, qui peuvent effectivement induire des surcoûts de certification. Ceci explique l'apparition, à partir des années 1990, des techniques de « construction de la safety à base de modèles » (Model-Based Safety Assessment). La démarche proposée dans le présent article s'inscrit dans le cadre de ces techniques. L'objectif est de décrire sur un exemple réel une méthode d'évaluation de performances de safety, applicable aux étapes amont (« early validation ») du développement de ce système. L'architecture n'étant, à ce stade, pas figée, la méthode permet d'effectuer des comparaisons entre variantes du point de vue des performances de safety, et au final de choisir l'option la plus intéressante de ce point de vue. La méthode est supportée par le logiciel SYNTHECOUPE développé par Thales.

Summary

Safety assessment for complex systems is a critical task, which should begin as soon as system definition is started. The impact of error correction in terms of cost and delay is all the more important since error detection is late. Furthermore, current practices are mainly based on manual, error prone tasks, which may indeed lead to additional certification costs. This is the reason why, starting from the nineties until now, Model-Based Safety Assessment (MBSA) techniques have emerged. The approach we propose in this paper comes within the scope of these techniques. Our objective is to describe on a real example a method for providing safety performances estimates. That method can be applied to the early stages of development process. At that time, system architecture is not yet frozen, and the method enables comparisons between architecture variants from safety point of view. At the end, the most attractive choice can be selected. That method is supported by a software tool called SYNTHECOUPE, which has been developed by Thales.

1. Introduction et contexte

La construction de la safety pour un système complexe est une tâche qu'il est préférable de commencer lors des étapes initiales de définition de ce système. L'avantage principal est de valider cette définition en amont de la phase de développement. L'impact d'une correction d'erreur sur les coûts et les délais est en effet d'autant plus grand que cette erreur est détectée plus tard. Ceci est particulièrement vrai pour les systèmes avioniques hautement intégrés.

Par ailleurs, les pratiques courantes du métier sont fondées sur des tâches manuelles, fortement sujettes à erreur, qui peuvent au final augmenter les coûts de certification. Ceci explique l'apparition, à partir des années 1990, des techniques de « construction de la safety à base de modèles » (Model-Based Safety Assessment, MBSA) dans les milieux universitaires et industriels. Le terme « MBSA » est à prendre ici dans son acceptation la plus large : il couvre à la fois les modèles de type « fonctionnel », constitués par des réseaux de « fonctions » interconnectées, et les modèles de type « physique », constitués par des « composants » matériels et/ou logiciels interconnectés. Un objet « fonction » correspond à un sous-ensemble plus ou moins vaste des fonctionnalités du système modélisé. Un objet « composant » correspond à un sous-ensemble, matériel ou logiciel, de l'architecture physique de ce système. Les modèles MBSA peuvent également être de nature « statique » ou « dynamique ». Les outils associés aux modèles statiques comprennent notamment la génération automatique d'arbres de défaillances à partir du modèle ; les outils associés aux modèles dynamiques offrent, en plus de ces fonctionnalités, des capacités de simulation. Les dernières décennies ont vu non seulement le développement de nombreuses méthodes, techniques et outils, mais aussi, pour certaines d'entre elles, leur adoption progressive par les organismes de certification. Par exemple, le système de commandes de vol de l'avion 7x de Dassault a été certifié sur la base de modèles écrits dans une version « dataflow » du langage Altarica [1].

L'utilisation de modèles est particulièrement recommandée lors de la phase dite « early validation » mentionnée ci-dessus. En permettant à l'équipe de développement de découvrir rapidement certains problèmes, elle augmente son degré de confiance dans la solution choisie. Ces modèles doivent pouvoir être conçus, mis à jour et supprimés facilement, de manière à écarter au moindre coût les solutions fautives. Le développement du système peut ensuite se poursuivre d'une façon moins chaotique que si l'on avait procédé autrement.

Un critère important pour classifier les techniques MBSA concerne la relation existant entre le modèle safety et le processus principal de développement du système. On peut distinguer deux approches : (a) les modèles de safety sont des *extensions* des modèles utilisés pour l'ingénierie, et (b) les modèles de safety sont des *modèles dédiés* venant s'ajouter aux éventuels modèles système.

Un exemple de la première approche MBSA est la construction d'un « modèle système étendu avec injection de pannes » (Extended System Model with Failure Injection), technique développée dans le cadre des projets ESACS et ISAAC [2, 3]. Les experts safety reçoivent de la part de l'équipe de développement, des modèles d'ingénierie système écrits dans des langages du type SCADE ou Matlab Simulink. Ces modèles sont ensuite enrichis par des « modèles de défaillances » élémentaires (Failure Mode Models) qui viennent s'insérer dans les flux d'information du modèle principal.

Un autre exemple de cette première approche est le « viewpoint safety » de Thales, actuellement en cours de développement. Les modèles utilisés pour l'ingénierie sont construits sous l'outil Thales Capella [8]. Le développeur rajoute ensuite des « attributs safety » à certains objets du modèle. Ces attributs comprennent notamment les niveaux DAL et les défaillances « perte » ou « erroné » envisagés pour les composants. La propagation de pannes individuelles peut alors être simulée grâce au « viewpoint ». L'allocation par l'expert des niveaux DAL aux composants du modèle peut aussi être vérifiée automatiquement, au regard des recommandations de l'ARP4754 et de la DO178B.

L'avantage principal de l'approche « modèle étendu » est la cohérence reliant par construction les analyses de safety et le processus de développement. Ces deux processus partagent en effet le même environnement de modélisation, fondé sur des langages et des outils communs. Malheureusement, les modèles système détaillés requis pour l'analyse de safety apparaissent parfois relativement tard dans la

conception du système, les surcoûts liés à d'éventuelles évolutions étant déjà élevés. Enfin, les niveaux de granularité pertinents pour la conception du système et pour l'analyse de safety ne sont pas forcément compatibles. En d'autres termes, la complexité du modèle étendu peut dépasser la capacité de traitement des outils d'analyse de safety disponibles sur le marché.

La deuxième approche MBSA, fondée sur l'utilisation pour la safety de modèles dédiés distincts des modèles système, allège sensiblement les difficultés mentionnées ci-dessus. Les modèles étant spécialement conçus pour les besoins de la safety, l'expert métier peut choisir les composants qu'il juge pertinents pour son analyse, et décrire leur comportement et leurs interfaces au niveau de détail juste nécessaire. D'inutiles complexités d'analyse peuvent ainsi être évitées. La cohérence entre modèles système et modèles de safety n'est plus garantie, elle doit être maintenue par l'expert tout au long du processus de développement. Cette deuxième approche comprend notamment les techniques de « Failure Logic Modeling ».

Les techniques et notations MBSA avancées, telles que HiP-HOPS [4], FPTN [5], et Altarica [1], appartiennent souvent à cette dernière catégorie. Le comportement des composants modélisés est vu comme une relation de dépendance entre d'une part l'état des sorties, et d'autre part l'état des entrées et le statut actif/inactif des défaillances internes. A noter toutefois que dans certains cas, le modèle doit contenir des informations dites « globales », non rattachables à l'un quelconque des composants individuels.

La démarche proposée dans le présent article s'inscrit dans le cadre des techniques MBSA. Pour l'essentiel, elle consiste à générer, à partir d'un ensemble important de coupes ou de séquences minimales, des informations plus synthétiques. Les coupes ou séquences minimales sont en général obtenues par compilation ou simulation d'un modèle MBSA sur un outil adapté tel que Cécilia fondé sur le langage Altarica. Les informations synthétiques sont ensuite générées par l'outil SYNTHECOUPE. Elles permettent d'estimer les performances de safety d'un système complexe, notamment pendant la phase d'« early validation ».

Mentionnons également que pour compléter cette chaîne d'outils, Thales développe actuellement une « passerelle » permettant de créer un modèle Cécilia à partir d'un modèle Capella. L'utilisateur sélectionne sous Capella les objets (composants, fonctions hébergées, liens fonctionnels) qu'il souhaite voir apparaître sous Cécilia. La passerelle permet alors d'exporter ces objets puis de les réimporter sous Cécilia sous la forme de « nodes » Altarica interconnectés, qui sont des boîtes vides. La tâche suivante de l'expert safety consiste à saisir sous Cécilia le code Altarica correspondant à chacune de ces boîtes. Après compilation du modèle ainsi complété, les coupes minimales pourront alors être automatiquement générées par l'outil.

2. Objectifs

L'objectif général de cet article est de décrire sur un exemple réel une méthode d'évaluation de performances de safety, applicable aux stades amont (« early validation ») de la conception d'un système complexe. Cette méthode comprend deux étapes principales. La première étape repose sur l'utilisation des techniques de « Failure Logic Modeling ». Elle consiste, pour chacune des architectures envisagées pour le système, à construire un modèle dysfonctionnel, à l'aide d'un outil dédié de type SD9 (Dassault Systèmes) ou Cécilia (Dassault Aviation). L'expert modélise le comportement dysfonctionnel des composants du système, puis combine ces modèles pour aboutir à un modèle système. L'outil permet également, pour chacun des modèles système concernés et pour chaque événement redouté *ER*, de générer le jeu de coupes (ou de séquences) minimales associé, que nous nommerons par la suite *coupes minimales individuelles*, ou *coupes minimales fonctionnelles*.

La deuxième étape de la méthode consiste à évaluer, à partir des coupes minimales individuelles produites par la première étape, les performances safety qui caractérisent ces modèles système. Ces performances safety comprennent notamment des majorants des probabilités par heure d'occurrence des événements redoutés. A l'issue de cette seconde étape, on dispose, pour chacun des modèles système et chaque événement redouté, d'un jeu de critères qualitatifs et quantitatifs. Les comparaisons entre architectures du point de vue de la safety sont alors possibles et s'appuieront sur ces critères.

La construction de tels modèles (première étape) soulève des difficultés, par exemple la définition, lors des phases préliminaires de l'étude, de comportements dysfonctionnels pertinents pour les composants du système. A ce stade, la structure physique et les fonctionnalités de certains composants ne sont que grossièrement connues. En l'absence de FMEA suffisamment détaillées, les modes de défaillances physiques de ces composants ne sont pas non plus clairement identifiés. Les données dysfonctionnelles dont dispose l'expert pour ces composants se réduisent souvent à des estimations des taux de défaillances totaux. Les logiques de défaillances (Failure Logic Models) à établir pour les composants du système doivent donc s'appuyer sur d'autres types de défaillances.

En supposant établies d'une manière ou d'une autre ces logiques de défaillances, l'exploitation des modèles système à des fins d'évaluation de la safety (deuxième étape) se heurte à d'autres difficultés, notamment la taille excessive des jeux de coupes minimales individuelles, souvent non quantifiées, produits par la première étape. Des jeux comprenant plusieurs milliers, voire dizaines ou centaines de milliers, de coupes minimales individuelles sont fréquemment rencontrés. Il existe donc un réel besoin d'exprimer un grand nombre de coupes minimales individuelles sous une forme synthétique, quantifiée, et facilement interprétable.

Dans ce cadre, le deuxième objectif de l'article est d'une part de décrire la démarche de construction des modèles système utilisés pour notre exemple (première étape) et d'autre part de montrer comment le problème posé par le foisonnement des coupes minimales individuelles a pu être maîtrisé grâce à l'outil SYNTHECOUPE développé par Thales, au profit d'une vue plus synthétique de ces coupes (deuxième étape). Le troisième objectif de l'article est de montrer sur notre exemple comment la comparaison, puis le choix, d'une architecture système parmi diverses variantes possibles, ont pu être facilités par l'usage de SYNTHECOUPE.

3. Le système exemple

Le système exemple considéré dans l'article est constitué par 4 senseurs interconnectés embarqués sur un avion (Figure 1), dont les sorties sont présentées en entrée d'autres équipements, par exemple un PFD (Primary Flight Display). L'ensemble des 4 senseurs doit globalement tenir des exigences très dures de safety, d'une part pour l'intégrité des données générées en sortie, d'autre part pour la disponibilité globale de ces données. Le but recherché pour l'intégrité est de permettre au PFD de sélectionner n'importe laquelle des 4 voies, avec une probabilité de perte d'intégrité inférieure à 1E-9 par heure de vol. Le but recherché pour la disponibilité est d'autoriser le PFD à sélectionner, en cas de perte d'une ou de plusieurs voies, l'une des voies encore disponibles, la probabilité de perte simultanée des 4 voies restant inférieure à 1E-9 par heure de vol. Ces deux objectifs doivent aussi être tenus en configuration « dispatch », i.e. quand l'un des 4 senseurs reste arrêté pendant toute la durée de la mission.

Le principe fonctionnel retenu pour tenir ces objectifs est la surveillance exercée par chaque senseur sur lui-même et sur les 3 autres. Des informations issues de chaque senseur Si sont réinjectées en entrée des 3 autres senseurs Sx, Sy et Sz. Chaque senseur Si dispose donc, en plus du paramètre ADRCi qui lui est propre et qu'il propage normalement vers l'extérieur, d'autres signaux en provenance des Sx lui permettant de détecter une éventuelle perte d'intégrité du paramètre ADRCi. Dans ce dernier cas, la sortie du senseur Si véhiculant ADRCi sera coupée. Ce mécanisme permet donc, pour chaque senseur Si, d'augmenter l'intégrité du paramètre ADRCi généré vers l'extérieur, au prix d'une perte en disponibilité de ce signal.

Trois variantes d'architecture pour les senseurs critiques ont été envisagées à des fins d'évaluation et de comparaison (Figures 1, 2 et 3). Un senseur est constitué d'un composant COM effectuant le calcul des paramètres critiques, et d'un composant MON surveillant la partie COM. Les MON et COM sont eux-mêmes décomposés en *fonctions*, ou *opérateurs fonctionnels*, les deux termes étant considérés comme

synonymes dans notre article. En cas de détection par MON d'une perte d'intégrité sur COM, le switch situé sur le MON des Figures 1, 2 et 3 coupe la sortie ADRCi qui lui est associée. Les trois variantes correspondent à des répartitions différentes des mécanismes de consolidation, de validation et de surveillance mutuelle mis en œuvre dans les parties COM et MON. La troisième variante est économiquement la plus intéressante, car le composant MON qui lui est propre est suffisamment simple pour pouvoir être implémenté en hardware uniquement.

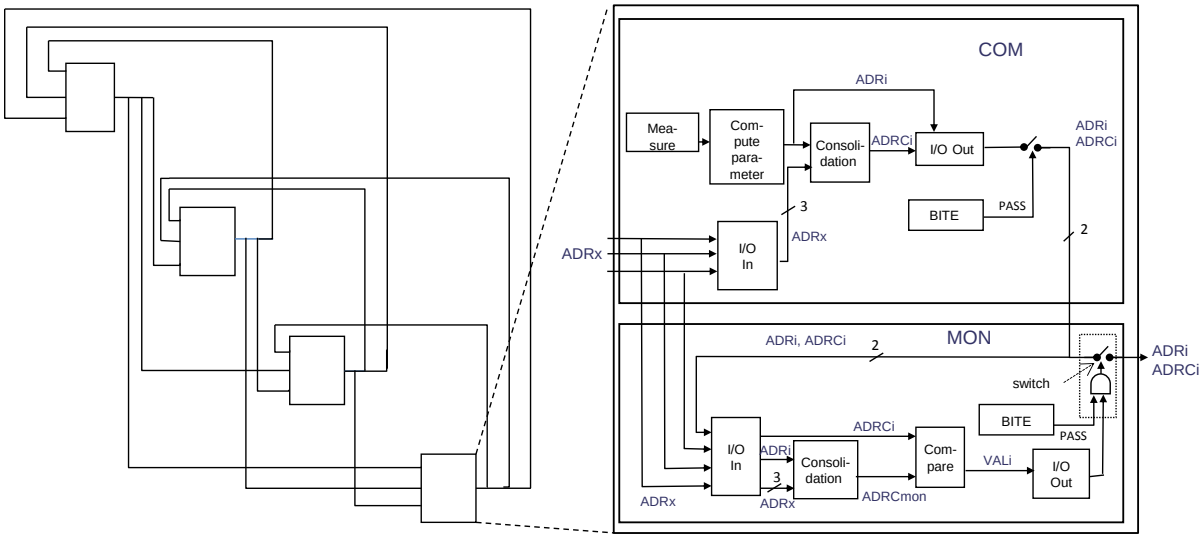

Figure 1. Les 4 senseurs interconnectés et l'architecture COM-MON « en parallèle »

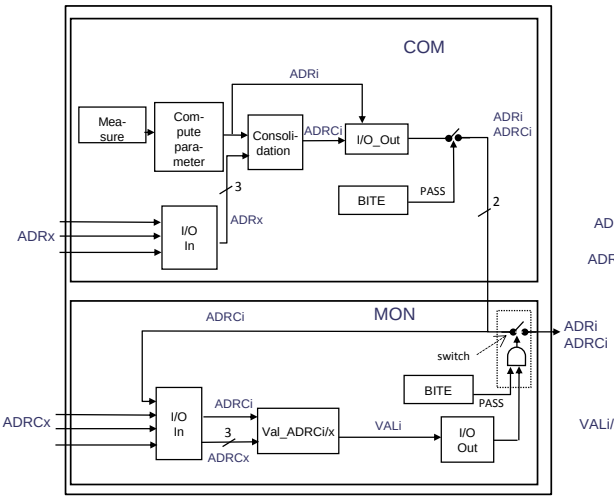

Figure 2. L'architecture COM-MON « en série »

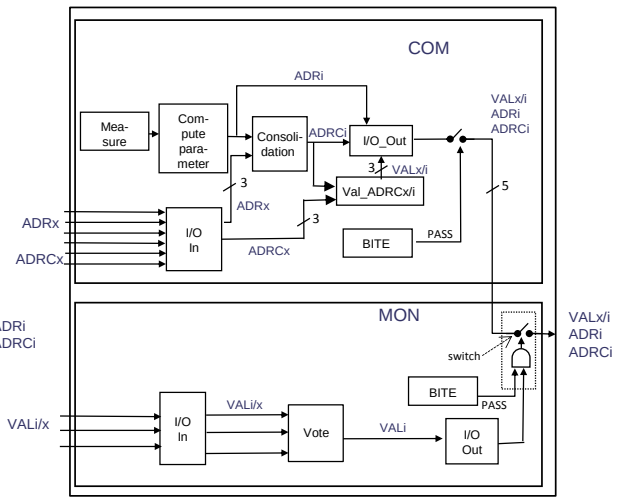

Figure 3. L'architecture « réduite »

3.1 Architecture COM-MON « en parallèle »

Dans cette architecture (Figure 1), la partie COM du senseur Si génère, en plus du paramètre consolidé ADRCi, un paramètre ADRi non consolidé, à destination de la partie MON de Si et des parties COM et MON des 3 autres senseurs Sx. Les fonctions « Consolidation » incluses dans les parties COM et MON sont identiques et partagent les mêmes entrées : le ADRi local et les ADRx issus des 3 autres senseurs. Elles génèrent les paramètres consolidés ADRCi et ADRCmon respectivement. Le paramètre ADRCi (resp. ADRCmon) est la recopie de ADRi quand les 4 entrées ADRi et ADRx sont cohérentes entre elles. Dans le cas contraire, le paramètre ADRCi (resp. ADRCmon) est coupé en interne de la fonction « Consolidation ».

La partie MON contient en plus un comparateur dont les entrées sont le ADRCi généré par la partie COM, et le paramètre ADRCmon calculé en parallèle par la partie MON. Sa sortie est le booléen VALi, qui vaut **false** quand ces entrées sont en désaccord, auquel cas le switch immédiatement situé en sortie de MON est ouvert. A noter pour cette architecture le parallélisme des traitements, et le fait que les seules données échangées entre senseurs sont les paramètres non consolidés ADRi et ADRx.

3.2 Architecture COM-MON « en série »

La partie COM de cette architecture est identique à la partie COM de l'architecture précédente. Le paramètre non consolidé ADRi généré par le COM de Si est injecté en entrée des COM des autres senseurs. Le paramètre consolidé ADRCi généré par le COM de Si est lui-même injecté en entrée du MON de Si et des MON des autres senseurs.

La fonction « Val_ADRCi/x » du MON de Si a pour objet la validation de la voie i par les 3 autres voies x, y et z. Elle prend pour entrées le ADRCi issu du COM de Si, et les 3 signaux ADRCx issus des autres senseurs. Elle génère le booléen VALi qui vaut **true** quand les 4 entrées sont en accord (le switch est passant), et **false** dans le cas contraire (le switch est ouvert). Le booléen VALi représente donc la validité de la voie i « vue globalement » par les autres voies. A noter pour cette architecture la « mise en série » des traitements effectués par

les fonctions « Consolidation » et « VAL_ADRCi/x », et le fait que les données échangées entre senseurs sont l'ensemble des paramètres consolidés ADRCi et non consolidés ADRi.

3.3 Architecture COM-MON avec MON « réduit »

Dans cette architecture, les paramètres consolidés ADRCi et non consolidés ADRi issus du COM de Si sont présentés en entrée du COM des autres senseurs. En plus de la fonction « Consolidation », le COM d'un senseur comporte une fonction « Val_ADRCx/i » dont l'objet est la validation par la voie i des 3 autres voies x. Cette fonction prend pour entrées le ADRCi issu de la consolidation locale, et les 3 paramètres consolidés ADRCx issus des voies x. Elle génère en sortie les 3 booléens VALx/i correspondant à ces voies. Le booléen VALx/i relatif à la voie x vaut **true** quand les deux entrées ADRCi et ADRCx de la fonction sont en accord, et **false** dans le cas contraire. Les 3 signaux VALx/i pour un Si donné sont présentés en entrée des MON des 3 autres senseurs.

La partie MON de Si comporte une fonction « vote », dont les entrées sont les 3 VALi/x et la sortie VALi. Le booléen VALi vaut **true** quand 2 au moins des entrées VALi/x valent **true** (le switch est passant), et **false** dans le cas contraire (le switch est ouvert). Les sorties ADRi et ADRCi du senseur sont donc coupées quand le senseur Si est « vu » comme invalide par au moins 2 autres senseurs. Notons pour finir la logique très simple de la partie MON pour cette architecture, qui pourrait être implémentée en hardware uniquement.

Compte tenu de la distribution, assez diversifiée, des fonctions de consolidation, de validation et de vote sur les 4 senseurs, la comparaison des performances safety entre les 3 variantes n'est pas faisable à la main. Elle passe nécessairement par l'usage de modèles.

4. Construction et exploitation des modèles système (première étape)

Nous présentons dans cette section la première étape de la démarche décrite dans cet article, telle qu'elle s'applique sur notre système exemple. Cette première étape comprend la création, selon les principes de « Failure Logic Modeling », des 3 modèles système correspondant aux 3 variantes mentionnées plus haut. Elle couvre également l'extraction des coupes minimales individuelles associées à ces modèles. Pour finir, elle résume sous forme de tableau les différents jeux de coupes individuelles correspondant aux 3 architectures, aux 2 configurations (nominale, dispatch), et aux 2 événements redoutés (intégrité, disponibilité) considérés. L'outil utilisé pour effectuer ces tâches est Cécilia (Dassault Aviation), basé sur le langage Altarica.

4.1 Le réseau d'opérateurs fonctionnels sous-jacents à un composant

En l'absence d'informations relatives à la *structure physique* des composants, nous nous appuierons sur leur *structure fonctionnelle*. Dans le contexte de cet article, la structure fonctionnelle associée à un composant est le réseau des fonctions interconnectées (opérateurs fonctionnels) qui lui sont sous-jacentes.

Dans notre système exemple ces structures fonctionnelles nous sont fournies comme données d'entrée (Figures 1, 2 et 3). Citons comme exemples d'opérateurs fonctionnels les fonctions « Consolidation », « Compute parameter » et « Vote ». A noter que pour d'autres systèmes exemples, la décomposition des composants en réseaux d'opérateurs peut ne pas être fournie d'emblée, mais constituer une tâche préliminaire à la construction du modèle proprement dite.

4.2 Catégories associées aux opérateurs, aux informations véhiculées et aux défaillances individuelles

Il nous faut maintenant distinguer 2 catégories d'information : la *catégorie continue* et la *catégorie discrète*. La catégorie continue correspond aux signaux tels que ADRCi ou ADRi, pouvant prendre l'une quelconque des valeurs comprises dans un intervalle. Nous définirons 3 états pour ces informations : OK (présente et correcte), KO (présente et erronée) et NO (information absente). La catégorie discrète correspond aux signaux ne pouvant prendre qu'un nombre fini et généralement faible de valeurs, par exemple les signaux booléens VALx/i générés par un senseur (Figure 3). Les 2 valeurs possibles pour ces signaux sont **true** et **false**.

Un opérateur fonctionnel comportant une sortie Out unique sera de même catégorie continue ou discrète que Out. Si l'opérateur est de catégorie continue, nous lui associerons les défaillances individuelles de base « Out_LO » et « Out_EO », auxquelles d'autres peuvent éventuellement s'ajouter. La défaillance « Out_LO » supprime l'information Out (valeur NO). La défaillance « Out_EO » corrompt cette information (valeur KO). Si l'opérateur est de catégorie booléenne (cas particulier de la catégorie discrète), nous lui associerons les défaillances individuelles de base « Out_STT » (sortie forcée à **true**) et « Out_STF » (sortie forcée à **false**).

A noter pour finir qu'un opérateur comportant plusieurs sorties possèdera autant de couples de défaillances individuelles (Out_LO, Out_EO) ou (Out_STT, Out_STF) que de sorties Out continues ou booléennes. Selon la catégorie de ces sorties, cet opérateur sera donc de catégorie continue, booléenne, ou mixte.

4.3 L'établissement des logiques de défaillances

Nous établissons ensuite la logique de défaillances de chaque opérateur hébergé par un composant en nous basant sur les catégories d'information qu'il manipule et la nature de l'opération qu'il effectue. La logique de défaillances du composant proprement dit résulte de l'assemblage de ces logiques partielles. De même, la logique de défaillances associée au système complet résulte de l'assemblage des logiques de défaillances de ses composants.

Un opérateur étant donné, les états de chacune de ses sorties sont exprimés sous forme d'expressions booléennes dont les termes sont les états des entrées et le statut actif/inactif des défaillances individuelles de cet opérateur. L'ensemble de ces expressions booléennes constitue la logique de défaillances de l'opérateur. Dans le contexte de cet article, les logiques de défaillances sont exprimées en Altarica,

```
// I/O_Out operator (COM, Figure 3)
trans
 ~ ADRi_Out_Lost |-  ADRi_Out_LO → ADRi_Out_Lost := true;
 ~ ADRi_Out_Err |-  ADRi_Out_EO → ADRi_Out_Err := true;
 ~ ADRCi_Out_Lost |-  ADRCi_Out_LO → ADRCi_Out_Lost := true;
 ~ ADRCi_Out_Err |-  ADRCi_Out_EO → ADRCi_Out_Err := true;
 ~VALx/i_Out_StkT |- VALx/i_Out_STT → VALx/i_Out_StkT := true;
 ~VALx/i_Out_StkF |- VALx/i_Out_STF → VALx/i_Out_StkF := true;
 ~VALy/i_Out_StkT |- VALy/i_Out_STT → VALy/i_Out_StkT := true;
 ~VALy/i_Out_StkF |- VALy/i_Out_STF → VALy/i_Out_StkF := true;
 ~VALz/i_Out_StkT |- VALz/i_Out_STT → VALz/i_Out_StkT := true;
 ~VALz/i_Out_StkF |- VALz/i_Out_STF → VALz/i_Out_StkF := true;
```

```
// I/O_Out operator (suite )
assert
    (if ADRi_In = NO or ADRi_Out_Lost then ADRi_Out = NO
     else
       (if ADRi_In = KO or ADRi_Out_Err then ADRi_Out = KO
        else ADRi_Out = OK
       )
    ) ;
    (if ADRCi_In = NO or ADRCi_Out_Lost then ADRCi_Out = NO
     else
       (if ADRCi_In = KO or ADRCi_Out_Err then ADRCi_Out = KO
        else ADRCi_Out = OK
       )
    ) ;
    VALx/i_Out = ((Valx/i_In and ~VALx/i_Out_StkF) or VALx/i_Out_StkT);
    VALy/i_Out = ((Valy/i_In and ~VALy/i_Out_StkF) or VALy/i_Out_StkT);
    VALz/i_Out = ((Valz/i_In and ~VALz/i_Out_StkF) or VALz/i_Out_StkT);
```

Figure 4. Le code Altarica pour l'opérateur I/O_Out

A titre d'exemple, l'opérateur « I/O_Out » situé dans la partie COM du senseur de la Figure 3, est de catégorie mixte (cf. code Altarica ci-dessus). Les défaillances individuelles définies pour cet opérateur sont les événements ADRi_Out_LO, ADRi_Out_EO, ADRCi_Out_LO**,** ADRCi_Out_EO, VALx/i_Out_STT et VALx/i_Out_STF, etc. ou d'une manière équivalente, les variables « mémoire » ADRi_Out_Lost, ADRi_Out_Err, ADRCi_Out_Lost, ADRCi_Out_Err, VALx/i_Out_StkT et VALx/i_Out_StkF, etc. Notons que les variables Altarica « de flux » ADRi_In, ADRi_Out, ADRCi_In, ADRCi_Out, etc. modélisent les entrées et les sorties de l'opérateur fonctionnel.

4.4 Evénements redoutés et coupes minimales individuelles

L'étape suivante de la démarche est la saisie sur le modèle des expressions booléennes caractérisant les événements redoutés. Le premier *ER* est l'émission d'un paramètre critique erroné ADRCi par l'un quelconque des 4 senseurs, dont la probabilité doit rester inférieure à 1E-9 par heure de vol. Cette condition s'exprime en Altarica sous la forme

((ADRC1 = KO) ***or*** (ADRC2 = KO) ***or*** (ADRC3 = KO) ***or*** (ADRC4 = KO)) = ***true***

Le deuxième *ER* est la perte simultanée des paramètres critiques ADRCi générés par les 4 senseurs, dont la probabilité doit rester inférieure à 1E-9 par heure de vol. Cette condition exprimée en Altarica est la suivante :

((ADRC1 = NO) ***and*** (ADRC2 = NO) ***and*** (ADRC3 = NO) ***and*** (ADRC4 = NO)) = ***true***

Les coupes minimales individuelles, i.e. les combinaisons minimales de défaillances individuelles provoquant l'*ER* (événements Altarica), sont ensuite extraites du modèle (en passant éventuellement comme étape intermédiaire par la génération automatique d'un arbre de défaillances).

4.5 Les 12 jeux de coupes minimales individuelles

La démarche décrite ci-dessus s'applique à chacune des 3 variantes envisagées pour notre architecture. Elle aboutit à la création de 3 modèles système. Pour chacun de ces modèles, 2 configurations sont à prendre en compte : la configuration « nominale » (les 4 senseurs sont actifs), et la configuration « dispatch » (l'un d'entre eux est éteint).

Par ailleurs, les 2 événements redoutés (intégrité et disponibilité) sont à considérer pour chacun des 6 couples (variante, configuration). Il existe donc au total 12 triplets (variante, configuration, *ER*), soit 12 jeux de coupes minimales individuelles finalement générés au terme de la première étape de notre approche.

La Table 1 exprime ces résultats. Les coupes minimales se comptent par milliers, voire dizaines ou centaines de milliers. Les défaillances individuelles n'étant pas quantifiées, ces coupes ne sont elles-mêmes pas quantifiées.

	Ordre	Parallèle	Parallèle Dispatch	Série	Série Dispatch	Réduit	Réduit Dispatch
INTEGRITE	1						
	2	34	21	62	54	8	123
	3	168	105	648	225	1508	
	4	108	279	1612		2320	
	5	1664	3	1884		2112	
	6	36		2964		240	
	7	4		672		168	
	8			584			
	9			24			
	10			8			
	TOTAL	**2014**	**408**	**8458**	**279**	**6356**	**123**

	Ordre	Parallèle	Parallèle Dispatch	Série	Série Dispatch	Réduit	Réduit Dispatch
DISPONIBILITE	1						
	2		39		72	12	
	3	256	5433	624	797	144	637
	4	87957	2835	10549	666	14656	3843
	5	77832	1296	17720	117	151516	9063
	6	190348	216	35696	27	464642	13824
	7	7912		21548			
	8	169680		15962			
	9	32		1196		> 500000	
	10	131712		5566			
	11						
	12	38416		2402			
	TOTAL	**704145**	**9819**	**111263**	**1679**	**> 1000000**	**27367**

Table 1. Les coupes minimales individuelles

5. Evaluation de base des performances safety (deuxième étape)

5.1 Problématique

Les coupes minimales fonctionnelles obtenues lors de la première étape de notre démarche fournissent de nombreux détails sur les comportements dysfonctionnels critiques des composants, mais ne permettent pas de conclure sur la tenue des objectifs quantitatifs de safety. D'une part, ces coupes minimales n'étant pas quantifiées, ne fournissent pas directement des estimations des probabilités d'occurrence des *ER*. D'autre part, du fait du grand nombre de ces coupes, nous ne disposons pas d'une vue synthétique des performances safety atteignables par le système. Nous ne savons notamment pas identifier les composants les plus critiques, pouvant provoquer un *ER* à eux tous seuls ou en association avec un autre composant.

La méthode que nous proposons pour résoudre ces problèmes est la principale innovation décrite dans cet article. Sa mise en œuvre passe par l'utilisation du logiciel SYNTHECOUPE qui a été développé spécifiquement chez Thales pour cet usage.

5.2 Principe de la méthode

Elle consiste à « synthétiser » chacun des 12 jeux de coupes minimales individuelles en un ensemble plus réduit de « coupes minimales composant » quantifiées, qui permettent d'estimer un majorant de la probabilité d'occurrence par heure de l'événement redouté.

Nous introduisons d'abord la notion de « défaillance composant ». Un composant *Ci* sera considéré comme « défaillant » d'un point de vue modèle s'il « contient » au moins un opérateur défaillant. La variable booléenne *FLi* représentant la défaillance composant sera donc le OU logique de toutes les défaillances individuelles *Dij* « comprises » dans *Ci* :

$$FLi = \sum_{j} Dij$$

Une coupe minimale fonctionnelle étant donnée, l'ensemble des défaillances individuelles *Dij* comprises dans cette coupe et appartenant au même composant *Ci* sera remplacé par la « défaillance composant » *FLi*. Nous obtenons ainsi, pour chaque coupe minimale fonctionnelle, une combinaison de défaillances composant qui sera nommée « coupe composant ». L'ensemble de ces coupes composant sera ensuite compressé, en éliminant les doublons et les coupes non minimales, i.e. englobant d'autres coupes composant contenant moins d'éléments.

Cette opération s'applique, pour chacun des 12 jeux, à l'ensemble des coupes fonctionnelles minimales appartenant à ce jeu. Nous générons ainsi, pour chaque jeu, un ensemble de coupes minimales composant, duquel nous déduirons un majorant de la probabilité par heure d'occurrence de l'événement redouté.

5.3 Examen de la méthode sur un exemple simple

5.3.1 Défaillances fonctionnelles et coupes minimales fonctionnelles associées

Soient C_1, C_2, C_3 et C_4 des composants physiquement ségrégués. Les défaillances individuelles qu'ils contiennent (i.e. qui sont associées à des opérateurs inclus dans le composant) sont représentées dans la Table 2. Le composant C_1 contient (D_{11}, D_{12}, D_{13}, D_{14}, D_{15}). Les composants C_2, C_3 et C_4 contiennent (D_{21}, D_{22}, D_{23}, D_{24}), (D_{31}, D_{32}, D_{33}, D_{34}) et (D_{41}, D_{42}, D_{43}, D_{44}) respectivement. Supposons également que l'événement redouté *ER* ne comprenne que 4 coupes minimales fonctionnelles MFC_1, MFC_2, MFC_3 et MFC_4 définies comme suit (Table 2) :

$$MFC_1 = D_{12}\,D_{13}\,D_{14}\,D_{24}\,D_{32} \qquad MFC_2 = D_{11}\,D_{14}\,D_{33}\,D_{41} \qquad MFC_3 = D_{31}\,D_{32}\,D_{41} \qquad MFC_4 = D_{32}\,D_{42}$$

les *Dij* et les *MFCk* étant considérées comme des variables booléennes. La variable booléenne *ER* représentant l'événement redouté est une fonction logique des défaillances *Dij* individuelles. Une telle fonction N'EST PAS égale à la somme booléenne de ses coupes minimales [6]. Par contre, elle implique logiquement ladite somme :

$$ER \Longrightarrow MFC_1 + MFC_2 + MFC_3 + MFC_4$$

i.e.
$$ER \Longrightarrow D_{12}\,D_{13}\,D_{14}\,D_{24}\,D_{32} + D_{11}\,D_{14}\,D_{33}\,D_{41} + D_{31}\,D_{32}\,D_{41} + D_{32}\,D_{42} \qquad \{1\}$$

5.3.2 Défaillances composants et coupes minimales composant associées

Les défaillances composant sont les suivantes :

$$FL_1 = D_{11} + D_{12} + D_{13} + D_{14} + D_{15} \qquad FL_2 = D_{21} + D_{22} + D_{23} + D_{24}$$
$$FL_3 = D_{31} + D_{32} + D_{33} + D_{34} \qquad FL_4 = D_{41} + D_{42} + D_{43} + D_{44}$$

Sachant que $Dij \Longrightarrow FL_i$ pour tout *i* et *j*, nous avons les implications logiques suivantes :

$$MFC_1 = D_{12}\,D_{13}\,D_{14}\,D_{24}\,D_{32} \Longrightarrow FL_1\,FL_2\,FL_3$$
$$MFC_2 = D_{11}\,D_{14}\,D_{33}\,D_{41} \Longrightarrow FL_1\,FL_3\,FL_4$$
$$MFC_3 = D_{31}\,D_{32}\,D_{41} \Longrightarrow FL_3\,FL_4$$
$$MFC_4 = D_{32}\,D_{42} \Longrightarrow FL_3\,FL_4$$

et aussi

$$ER \Longrightarrow MFC_1 + MFC_2 + MFC_3 + MFC_4 \Longrightarrow FL_1\,FL_2\,FL_3 + FL_1FL_3\,FL_4 + FL_3\,FL_4 + FL_3\,FL_4 \qquad \{2\}$$

et finalement

$$ER \Longrightarrow FL_1\,FL_2\,FL_3 + FL_3\,FL_4 \qquad \{3\}$$

La Table 2 illustre ces étapes. Les coupes composant générées avant réduction sont $FL_1\,FL_2\,FL_3$, $FL_1\,FL_3\,FL_4$, $FL_3\,FL_4$ et $FL_3\,FL_4$. Après réduction, il nous reste les 2 coupes minimales composant $FL_1\,FL_2\,FL_3$ et $FL_3\,FL_4$. Le passage de {2} à {3} n'est autre que la réduction logique de l'expression booléenne {2}, compte tenu des règles de simplification A + A = A, A + AB = A, et A + (~A)B = A + B. Le processus permettant de générer {3} à partir de {1} est appelé *synthèse des coupes minimales fonctionnelles en coupes minimales composant*, c'est le titre de notre article.

Pour finir, nous désignerons par le terme « coupes minimales fonctionnelles sous-jacentes à une coupe minimale composant » l'ensemble des coupes fonctionnelles ayant donné naissance à cette coupe composant. Pour notre exemple, l'unique coupe fonctionnelle sous-jacente à $FL_1\,FL_2\,FL_3$ est MFC_1. Les 2 coupes fonctionnelles sous-jacentes à $FL_3\,FL_4$ sont MFC_3 et MFC_4. La coupe MFC_2 n'est sous-jacente à aucune coupe composant.

	C1					C2				C3				C4			
	D11	D12	D13	D14	D15	D21	D22	D23	D24	D31	D32	D33	D34	D41	D42	D43	D44

C1	C2	C3	C4
FL1	FL2	FL3	FL4

C1	C2	C3	C4	
FL1	FL2	FL3	FL4	
5E-5	5E-5	5E-5		1,3E-13
		5E-5	5E-5	2,5E-09
				2,5E-09

Table 2. La synthèse des coupes sur un exemple simple

5.3.3 Quantification des coupes minimales composant et de l'événement redouté

Les variables booléennes *ER* et *FLi* intervenant dans {3} sont des fonctions logiques des défaillances individuelles de base *Dij*. Elles n'ont de sens, en toute rigueur, que d'un point de vue « modèle ». Pour aller plus loin, nous conviendrons de donner à *ER*, à *FLi* et donc à {3} une signification « physique ».

La variable booléenne *ER* représentera la présence de l'événement redouté. De même, *FLi* représentera la présence d'une panne matérielle dans *Ci*. *ER* et les *FLi* seront donc vus comme des variables booléennes aléatoires satisfaisant l'implication {3}.

Par ailleurs, nous supposerons que la durée de la mission est de une heure, et que toutes les pannes matérielles apparues pendant la mission sont réparées avant le début de la mission suivante. La probabilité par heure d'occurrence de l'*ER* sera notée *p(ER)*. De même, *p(FLi)* sera la probabilité pour que *Ci* tombe en panne au cours d'une heure, ce composant étant initialement exempt de pannes (absence de pannes « dormantes »). Avec ces conventions, *p(FLi)* est égal au taux de défaillances total λi du composant *Ci*, et nous déduisons de {3} l'inégalité

$$p(ER) \le p(FL_1\,FL_2\,FL_3 + FL_3\,FL_4) \le p(FL_1\,FL_2\,FL_3) + p(FL_3\,FL_4)$$

Les composants *Ci* étant physiquement ségrégués, les variables aléatoires *FLi* peuvent être considérées comme indépendantes. Les probabilités des produits apparaissant ci-dessus sont donc égales aux produits des probabilités, d'où l'inégalité

$$p(ER) \le p(FL_1)\,p(FL_2)\,p(FL_3) + p(FL_3)\,p(FL_4)$$

i.e.
$$p(ER) \le \lambda_1\lambda_2\lambda_3 + \lambda_3\lambda_4 \qquad \{4\}$$

Le produit $\lambda_1\lambda_2\lambda_3$ est un majorant de la probabilité associée à la coupe $FL_1\,FL_2\,FL_3$. De même, $\lambda_3\lambda_4$ est un majorant de la probabilité associée à FL_3FL_4. Un majorant de la probabilité d'occurrence par heure de *ER* sera simplement la somme $\lambda_1\lambda_2\lambda_3 + \lambda_3\lambda_4$ de ces majorants.

Supposons maintenant que l'exigence quantitative concernant *ER* soit *p(ER)* ≤ 1E-9. Si les valeurs des λi sont telles que $\lambda_1\lambda_2\lambda_3 + \lambda_3\lambda_4 \le$ 1E-9, nous aurons effectivement démontré que *p(ER)* ≤ 1E-9.

Rappelons pour finir que ce résultat suppose l'absence de pannes dormantes persistant d'une mission à la suivante, et la ségrégation physique des composants. Des travaux dépassant le cadre de cet article sont en cours qui permettront d'étendre le processus de synthèse à d'autres contextes, levant ainsi ces restrictions.

5.4 Application de la synthèse des coupes au système exemple

Les étapes successives de la synthèse nous conduisent de {1} à {3} et à {4}, elles sont facilement automatisables. Le logiciel SYNTHECOUPE a été implémenté qui réalise cette synthèse. Ses données d'entrée comprennent d'une part la liste des coupes minimales fonctionnelles à synthétiser, d'autre part la liste des défaillances individuelles intervenant dans ces coupes. Elles comprennent également, pour chacune de ces défaillances individuelles, l'indication du « groupe » (i.e. du composant) auquel elle appartient. La liste des défaillances et la liste des coupes fonctionnelles sont générées automatiquement à partir du modèle par l'outil Cécilia, sous forme de 2 fichiers XML. L'indication, pour chaque défaillance fonctionnelle, du composant auquel elle appartient, est fournie directement par l'utilisateur à SYNTHECOUPE. Les taux de défaillances Λi des composants *Ci* sont également fournis par l'utilisateur.

Les données de sortie comprennent, pour un ensemble donné de coupes minimales fonctionnelles, la liste des coupes minimales composant issues de la synthèse. Elles comprennent également, pour chaque coupe composant, la valeur du majorant de la probabilité qui lui est associée, et pour l'événement redouté, un majorant de sa probabilité d'occurrence par heure. SYNTHECOUPE peut aussi générer, pour chacune des coupes minimales composant, la liste des coupes minimales fonctionnelles qui lui sont sous-jacentes.

La synthèse des coupes a été réalisée par SYNTHECOUPE sur chacun des 12 jeux (variante, configuration, *ER*) de coupes minimales fonctionnelles. Les taux totaux de défaillances Λ_{COM_PAR}, Λ_{COM_SER}, Λ_{COM_RED} fournis en entrée pour les composants COM des 3 variantes sont estimés à 5E-5. De même, les taux totaux de défaillances Λ_{MON_PAR}, Λ_{MON_SER} fournis en entrée pour les composants MON des variantes « parallèle » et « série » sont estimés à 5E-5. Le MON de la variante « réduite » étant peu complexe, son taux total de défaillances est estimé à 1E-5.

La table 3 présente les résultats des 12 synthèses. On notera le nombre réduit des coupes composant eu égard au nombre important des coupes individuelles. La dernière ligne des tableaux fournit les majorants des probabilités d'occurrence des *ER* pour chacun des 12 jeux. Les cases grisées correspondent aux cas où ce majorant dépasse 1E-9.

Les seuls cas où l'on puisse effectivement conclure à la tenue des objectifs de safety, correspondent aux deux triplets (COM-MON parallèle, nominal, disponibilité) et (COM-MON série, nominale, disponibilité). Dans ces deux cas, en effet, les coupes minimales composant sont d'ordre 3 au minimum, alors que partout ailleurs des coupes composant d'ordre 2 sont présentes.

		Parallèle		Parallèle Dispatch		Série		Série Dispatch		Réduit		Réduit Dispatch	
INTEGRITE	Ordre	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER
	1												
	2	10	2,5E-08	6	1,5E-08	10	2,5E-08	6	1,5E-08	4	2,0E-09	6	9,0E-09
	3									4	5,0E-13		
	TOTAL	**10**	**2,5E-08**	**6**	**1,5E-08**	**10**	**2,5E-08**	**6**	**1,5E-08**	**8**	**2,0E-09**	**6**	**9,0E-09**

		Parallèle		Parallèle Dispatch		Série		Série Dispatch		Réduit		Réduit Dispatch	
DISPONIBILITE	Ordre	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER
	1												
	2			3	7,5E-09			3	7,5E-09	6	1,5E-08	3	7,5E-09
	3	4	5,0E-13	4	5,0E-13	4	5,0E-13	4	5,0E-13			4	1,6E-14
	4	11	6,9E-17			11	6,9E-17			5	2,1E-19		
	TOTAL	**15**	**5,0E-13**	**7**	**7,5E-09**	**15**	**5,0E-13**	**7**	**7,5E-09**	**11**	**1,5E-08**	**7**	**7,5E-09**

Table 3. Résultats de la synthèse des coupes sur le système exemple

La Table 4 montre les résultats détaillés fournis par la synthèse pour la variante « réduite », associée à l'*ER* d'intégrité. La présence de coupes d'ordre 2 explique la valeur 2E-9, légèrement trop élevée, pour le majorant de la probabilité de l'ER. A l'inverse, la Table 5 présente des résultats détaillés pour la variante « parallèle » associée à l'*ER* de disponibilité, qui permettent de conclure à la tenue de l'objectif.

		COM1	MON1	COM2	MON2	COM3	MON3	COM4	MON4	Probas
REDUITE NOMINALE / INTEGRITE	1	■	■							5,0E-10
	2			■	■					5,0E-10
	3					■	■			5,0E-10
	4							■	■	5,0E-10
	5	■		■		■				1,3E-13
	6	■		■				■		1,3E-13
	7	■				■		■		1,3E-13
	8			■		■		■		1,3E-13
										2,0E-09

Table 4. Coupes composant – réduite, nominale, intégrité

		COM1	MON1	COM2	MON2	COM3	MON3	COM4	MON4	Probas
PARALLELE NOMINALE / DISPONIBILITE	1	■		■		■		■		1,3E-13
	2	■		■		■		■		1,3E-13
	3	■				■		■		1,3E-13
	4	■		■		■		■		1,3E-13
	5	■		■		■		■	■	6,3E-18
	6	■		■		■	■	■	■	6,3E-18
	7	■				■		■	■	6,3E-18
	8	■		■		■		■	■	6,3E-18
	9		■	■		■		■	■	6,3E-18
	10		■	■		■		■	■	6,3E-18
	11		■	■		■		■	■	6,3E-18
	12		■		■	■		■	■	6,3E-18
	13		■		■	■			■	6,3E-18
	14		■		■		■	■		6,3E-18
	15		■		■		■		■	6,3E-18
										5,0E-13

Table 5. Coupes composant – parallèle, nominale, disponibilité

Il convient de souligner le caractère pessimiste des chiffres présentés en Table 3. Les probabilités d'occurrence par heure des défaillances composant sont systématiquement majorées par les valeurs Λ_{COM} ou Λ_{MON} des taux globaux. Dans la suite de cet article, nous distinguerons 3 types de défaillances composant. La démarche de synthèse de coupes sera modifiée de manière à prendre en compte cette distinction. Au final, nous aboutirons à des majorants pour les probabilités d'ER nettement moins pessimistes que les valeurs présentées en Table 3.

5.5 Comparaison entre la synthèse dite « de base » et d'autres approches existantes

Les outils tels que Cécilia (Dassault Aviation), SD9 (Dassault Systèmes) et Simfia (APSYS) fondés sur le langage Altarica « dataflow », comprennent des fonctionnalités voisines de celles fournies par notre synthèse SYNTHECOUPE dite « de base ». La notion de « composant » est remplacée par la notion de « zone » (Simfia) ou de « classe d'équivalence » (Cécilia). La notion de « coupe composant NON MINIMALE » est remplacée par la notion « d'ensemble de zones » ou « d'ensemble de classes d'équivalence ». La notion de « coupe minimale sous-jacente » existe dans les trois contextes sous des noms différents.

Notons pour finir la différence principale : la réduction des coupes composant non minimales en coupes composant minimales est sans équivalente dans le contexte de Cécilia ou de Simfia. Il s'agit là d'un degré de synthèse supplémentaire offert par SYNTHECOUPE.

6. Evaluation plus fine des performances safety (deuxième étape typée)

6.1 Les défaillances composant typées d'un point de vue « physique »

Nous n'avons jusqu'à présent défini pour un composant physique qu'une seule défaillance « générique » produite par l'ensemble des pannes matérielles, à laquelle nous associons une probabilité d'occurrence par heure égale au taux de défaillance total Λ du composant. Les termes « défaillance » et « panne » correspondent ici aux définitions données par Laprie [7]. Cette défaillance « générique » sera dite « non typée », par opposition aux défaillances composant « typées » définies ci-dessous. Pour notre exemple simple, la défaillance « générique » associée au composant *Ci* est modélisée par le booléen *FLi*.

Les études de safety effectuées en interne de Thales font fréquemment apparaitre des classifications plus fines. Sans être exhaustif, nous distinguerons 3 défaillances composant supplémentaires, de types *TL*, *PL* et *CR* respectivement. Ces défaillances correspondent aux comportements dysfonctionnels « perte totale », « perte partielle » et enfin à « calcul corrompu ». La « perte totale » du composant (type *TL*) peut être vue comme sa « passivation » : les sorties sont absentes ou figées à des valeurs fixes. Entrent notamment dans ce type toutes les pannes détectées par les tests en ligne BITE, dont la sanction est l'arrêt du composant, et les pannes d'alimentation provoquant son extinction. La FMEA du composant, si elle existe, permet d'affecter une valeur Λ_{TL} à la probabilité d'occurrence par heure de « perte totale », le composant étant initialement exempt de pannes. Dans le cas contraire, une valeur conservatrice $\Lambda_{TL} = 90\%\Lambda$ est fréquemment utilisée. Une autre valeur souvent utilisée est le majorant Λ.

Le « calcul corrompu » dans le composant (type *CR*) correspond à la génération d'erronés en interne de ce composant. En termes de FMEA, il s'agit des pannes physiques mentionnées dans la liste qui provoquent des erreurs de valeurs. La défaillance « perte partielle » (type *PL*) correspond aux pertes de fonctionnalités du composant. En termes de FMEA, il s'agit des pannes physiques mentionnées dans la liste qui suppriment des traitements.

Un choix possible pour l'application de notre méthode, est d'ignorer la distinction entre « calcul corrompu » et « perte partielle », ce qui revient à ne considérer que 2 types *TL* et *CR/PL*. La valeur $\Lambda_{CR/PL}$ de la probabilité d'occurrence par heure de « calcul corrompu » OU « perte partielle », le composant étant initialement exempt de pannes, sera $\Lambda - \Lambda_{TL}$.

Une approche moins majorante serait de conserver les 3 types *TL*, *CR* et *PL*, en leur affectant des valeurs Λ_{TL}, Λ_{CR} et Λ_{PL}. La FMEA du composant, si elle existe, peut servir à chiffrer ces valeurs. Dans le cas contraire, une valeur $\Lambda_{CR} = 5\%\Lambda$, voire $2\%\Lambda$ pourrait être retenue. La probabilité d'occurrence par heure de « perte partielle » serait alors $\Lambda_{PL} = \Lambda - \Lambda_{TL} - \Lambda_{CR}$.

Dans la suite de cet article, et pour rester général, nous nous appuierons sur cette classification en 3 types *TL*, *CR* et *PL*. Mais les développements qui vont suivre restent aisément transposables au cas d'une classification en 2 types *TL* et *CR/PL*.

6.2 Les défaillances fonctionnelles typées

Nous conviendrons, par analogie avec ce qui précède, d'affecter un type *L*, *E* ou *F* aux défaillances fonctionnelles. Un composant *C* comportera ZERO OU UNE SEULE défaillance de type *L*. Cette défaillance (perte totale) sera la perte simultanée de TOUTES les fonctions incluses dans le composant (plus précisément, la perte de toutes les sorties continues des fonctions, et le collage à une valeur par défaut des sorties booléennes).

En Altarica, nous pouvons la modéliser par une *synchronisation.* Cette synchronisation est une défaillance supplémentaire appartenant à C, qui vient s'ajouter aux défaillances fonctionnelles déjà prévues pour ce composant. Elle lie entre elles toutes les défaillances individuelles xx_LO et xx_STT (ou xx_STF selon la valeur par défaut choisie) des fonctions de ce composant. Le tirage de cet événement (perte totale) provoquera le tirage simultané des xx_LO et des xx_STx choisis, donc la perte ou le collage à une valeur par défaut de toutes les sorties primaires. Les notions de « perte totale » d'un point de vue « modèle » et d'un point de vue « physique » sont donc cohérentes entre elles.

Pour compléter cette définition, nous prendrons dans le code Altarica du composant une précaution supplémentaire, qui nous sera utile par la suite : le tirage de la synchronisation interdira le tirage de toutes les autres défaillances individuelles du composant. A l'inverse, le tirage d'une ou de plusieurs défaillances individuelles interdira le tirage de la synchronisation.

Une défaillance individuelle attachée à la sortie d'une fonction sera du type *E* si suite à son activation, des informations erronées sont générées sur cette sortie. Une défaillance individuelle sera du type *F* si elle n'est pas déjà typée *E* ou *L*.

La Table 6 montre les types que nous avons affectés aux défaillances individuelles du COM de l'architecture réduite (Figure 3). La synchronisation Altarica (type *L*) n'apparaît pas dans ce tableau, mais elle existe et lie entre elles toutes les défaillances annotées « F / Synchro ».

Fonction	**I/O_In**								
Défaillance	ADRx_EO	ADRx_LO	ADRy_EO	ADRy_LO	ADRz_EO	ADRz_LO	ADRCx_EO	ADRCx_LO	ADRCy_EO
Type	E	F / Synchro	E	F / Synchro	E	F / Synchro	E	F / Synchro	E
Fonction	**I/O_In**			**I/O_Out**				**Compute_parameter**	
Défaillance	ADRCy_LO	ADRCz_EO	ADRCz_LO	ADRi_EO	ADRi_LO	ADRCi_EO	ADRCi_LO	ADRi_EO	ADRi_LO
Type	F / Synchro	E	F / Synchro	E	F / Synchro	E	F / Synchro	E	F / Synchro
Fonction	**Consolidation**		**Val_ADRCx/i**						**BITE**
Défaillance	ADRCi_EO	ADRCi_LO	VALx/i_STT	VALx/i_STF	VALy/i_STT	VALy/i_STF	VALz/i_STT	VALz/i_STF	PASS_STF
Type	E	F / Synchro	F / Synchro	F	F / Synchro	F	F / Synchro	F	F / Synchro

Table 6. Les défaillances fonctionnelles typées pour le COM de l'architecture réduite

6.3 Les défaillances composant typées d'un point de vue « modèle »

Nous conviendrons que pour un composant *Ci* donné, les défaillances fonctionnelles *Dij* de types *L*, *E* ou *F* s'appelleront désormais *Lij*, *Eij* et *Fij* respectivement. La « perte totale », définie plus haut d'un point de vue « physique », sera représentée d'un point vue « modèle » par la variable booléenne *TLi = Lij*, *Lij* étant l'unique défaillance fonctionnelle de type *L* du composant *Ci*. Le « calcul corrompu », défini plus haut d'un point de vue « physique », sera représenté d'un point vue « modèle » par la variable booléenne *CRi*. Cette variable sera le OU logique de toutes les défaillances fonctionnelles de type *E* appartenant au composant *Ci*, la défaillance *Lij* restant inactive. Enfin, la défaillance « perte partielle » définie d'un point de vue « physique » sera représentée d'un point de vue « modèle » par le booléen *PLi*. Cette variable sera le OU logique de toutes les défaillances fonctionnelles de type *F* du composant *Ci*, les variables *Lij* et *Eij* restant inactives. Nous pouvons donc poser

$$TL_i = Lij \qquad CR_i = (\sim TLi)\sum_j Eij \qquad PL_i = (\sim TLi)\left(\sim\sum_j Eij\right)\sum_j Fij$$

Pour assurer la cohérence entre les deux points de vue, nous conviendrons que les booléens *TLi*, *CRi* et *PLi* sont aussi des variables aléatoires, de probabilités d'occurrence par heure Λ_{TLi}, Λ_{CRi} et Λ_{PLi} respectivement, *Ci* étant initialement exempt de pannes.

Voici maintenant quelques implications logiques qui nous seront utiles par la suite : $Lij \Rightarrow TLi$ de manière triviale, $Eij \Rightarrow (\sim TLi)\ Eij$ et $Fij \Rightarrow (\sim TLi)\ Fij$, puisque l'activation des défaillances *Eij* ou *Fij* exclue l'activation de $Lij = TL_i$. Nous en déduisons les implications suivantes :

$$\sum_j Eij \Rightarrow (\sim TLi) \sum_j Eij = CRi \qquad \{5\}$$

$$\sum_j Fij \Rightarrow (\sim TLi)\left(\sum_j Fij\right) = (\sim TLi)\left(\sum_j Eij\right)\left(\sum_j Fij\right) + (\sim TLi)\left(\sim\sum_j Eij\right)\left(\sum_j Fij\right)$$

i.e.
$$\sum_j Fij \Rightarrow CRi\left(\sum_j Fij\right) + PLi \Rightarrow CRi + PLi = CR/PLi \qquad \{6\}$$

6.4 Principe de la synthèse des coupes fonctionnelles en coupes composant typées

Une coupe minimale fonctionnelle étant donnée, le produit (coupe partielle) des défaillances individuelles typées comprises dans cette coupe et appartenant au même composant *Ci* sera remplacé par l'une des trois « défaillances composant » *TLi*, *CRi* ou *CR/PLi*. Si la coupe partielle comprend LA défaillance *Lij*, elle sera remplacée par *TLi*. Si elle ne comprend pas *Lij*, ni aucune des défaillances *Fij*, mais au moins l'une des défaillances *Eij*, la coupe partielle sera remplacée par *CRi*. Si enfin la coupe partielle ne comprend pas *Lij* mais l'une au moins des *Fij*, et éventuellement des défaillances *Ei*, elle sera remplacée par *CR/PLi = CRi + PLi*. Nous obtenons ainsi, pour chaque coupe minimale fonctionnelle, une combinaison de défaillances composant typées qui sera nommée « coupe composant typée ».

Cette opération s'applique, pour chacun des 12 jeux, à l'ensemble des coupes fonctionnelles minimales appartenant à ce jeu. Nous générons ainsi, pour chaque jeu, un ensemble de coupes composant typées, duquel nous ôtons les coupes redondantes et celles qui ne sont pas minimales (i.e. contiennent une autre coupe composant avec moins d'éléments et cohérence des types).

Pour chacun des 12 jeux, nous disposons au final d'un ensemble réduit de « coupes minimales composant typées », duquel nous pourrons déduire un majorant de la probabilité d'occurrence par heure de l'événement redouté.

6.5 Examen de la synthèse typée sur l'exemple simple

6.5.1 Défaillances fonctionnelles et coupes minimales fonctionnelles associées

Le composant C_1 contient (F_{11}, E_{12}, E_{13}, F_{14}, L_{15}). Les composants C_2, C_3 et C_4 contiennent (F_{21}, E_{22}, L_{23}, F_{24}), (E_{31}, E_{32}, E_{33}, L_{34}) et (L_{41}, F_{42}, F_{43}, E_{44}) respectivement (Table 7). Les 4 coupes minimales sont maintenant :

$$MFC_1 = E_{12}\,E_{13}\,F_{14}\,F_{24}\,E_{32} \qquad MFC_2 = F_{11}\,F_{14}\,E_{33}\,L_{41} \qquad MFC_3 = E_{31}\,E_{32}\,L_{41} \qquad MFC_4 = E_{32}\,F_{42}$$

L'implication {1} devient alors

$$ER \Rightarrow E_{12}\,E_{13}\,F_{14}\,F_{24}\,E_{32} + F_{11}\,F_{14}\,E_{33}\,L_{41} + E_{31}\,E_{32}\,L_{41} + E_{32}\,F_{42} \qquad \{7\}$$

6.5.2 Défaillances composants et coupes minimales composant associées

Sachant que d'après {5} et {6} $Lij \Rightarrow TL_i$, que $Eij \Rightarrow CR_i$ et que $Fij \Rightarrow CR/PL_i$ pour tout *j*, nous avons les implications logiques suivantes :

$$MFC_1 = E_{12}\,E_{13}\,F_{14}\,F_{24}\,E_{32} \Rightarrow CR/PL_1\ CR/PL_2\ CR_3$$
$$MFC_2 = F_{11}\,F_{14}\,E_{33}\,L_{41} \Rightarrow CR/PL_1\ CR_3\ TL_4$$
$$MFC_3 = E_{31}\,E_{32}\,L_{41} \Rightarrow CR_3\ TL_4$$
$$MFC_4 = E_{32}\,F_{42} \Rightarrow CR_3\ CR/PL_4$$

et aussi

$$ER \Rightarrow MFC_1 + MFC_2 + MFC_3 + MFC_4 \Rightarrow CR/PL_1\ CR/PL_2\ CR_3 + CR/PL_1\ CR_3\ TL_4 + CR_3\ TL_4 + CR_3\ CR/PL_4 \qquad \{8\}$$

et finalement
$$ER \Rightarrow CR/PL_1\ CR/PL_2\ CR_3 + CR_3\ TL_4 + CR_3\ CR/PL_4 \qquad \{9\}$$

La Table 7 illustre ces étapes. Après réduction, il nous reste 3 coupes minimales composant typées. Le passage de {8} à {9} n'est autre que la réduction logique de l'expression booléenne {8}, compte tenu des règles de simplification A + A = A, A + AB = A, et A + (~A)B = A + B. Le processus permettant de générer {9} à partir de {7} est appelé *synthèse des coupes minimales fonctionnelles en coupes minimales composant typées.*

C1					C2				C3				C4			
F11	E12	E13	F14	L15	F21	E22	L23	F24	E31	E32	E33	L34	L41	F42	F43	E44
	E	E	F					F		E						
F			F								E		L			
									E	E			L			
										E				F		

C1	C2	C3	C4
CR/PL	CR/PL	CR	
CR/PL		CR	TL
		CR	TL
		CR	CR/PL

C1	C2	C3	C4	
CR/PL / 5E-6	CR/PL / 5E-6	CR / 2,5E-6		6,3E-17
		CR / 2,5E-6	TL / 4,5E-5	1,1E-10
		CR / 2,5E-6	CR/PL / 5E-6	1,3E-11
				1,3E-10

Table 7. La synthèse en coupes composant typées sur un exemple simple

6.5.3 Quantification des coupes minimales composant typées et de l'événement redouté

Rappelons que les booléens *ER*, *TLi*, *CRi* et *CR/PLi* sont des variables aléatoires qui satisfont {9}. Nous supposerons toujours que la durée de la mission est de une heure, et que toutes les pannes matérielles apparues pendant la mission sont réparées avant le début de la mission suivante (absence de pannes dormantes). Avec ces conventions, nous déduisons de {9} l'inégalité suivante :

$$p(ER) \le p(CR/PL_1\ CR/PL_2\ CR_3 + CR_3\ TL_4 + CR_3\ CR/PL_4) \le p(CR/PL_1\ CR/PL_2\ CR_3) + p(CR_3\ TL_4) + p(CR_3\ CR/PL_4)$$

Les composants *Ci* étant physiquement ségrégués, les variables aléatoires correspondant à des composants différents peuvent être considérées comme indépendantes. Les probabilités des produits apparaissant ci-dessus sont donc égales aux produits des probabilités. Nous obtenons

$$p(ER) \le p(CR/PL_1)\ p(CR/PL_2)\ p(CR_3) + p(CR_3)\ p(TL_4) + p(CR_3)\ p(CR/PL_4)$$

Nous savons également que $p(CR/PLi) = p(CRi + PLi) \leq p(CRi) + p(PLi) = \Lambda_{CRi} + \Lambda_{PLi}$, qui implique en posant $\Lambda_{CR/PLi} = \Lambda_{CRi} + \Lambda_{PLi}$

$$p(ER) \leq \Lambda_{CR/PL1}\Lambda_{CR/PL2}\Lambda_{CR3} + \Lambda_{CR3}\Lambda_{TL4} + \Lambda_{CR3}\Lambda_{CR/PL4} \quad \{10\}$$

Nous supposerons que Λ_{TLi} = 4.5E-5, Λ_{CRi} = 2.5E-6 et $\Lambda_{CR/PLi}$ = 5E-6 pour tout *i*. Avec ces valeurs nous obtenons la quantification des coupes composant typées indiquée dans la table 7.

6.6 Application de la synthèse des coupes typées au système exemple

Les données d'entrée de SYNTHECOUPE comprennent, en plus des informations mentionnées en section 5.4, l'indication, pour chacune des défaillances individuelles du modèle système, du type *L*, *E* ou *F* qui lui est associé. Ces informations sont fournies directement par l'utilisateur à SYNTHECOUPE. Les taux de défaillances Λ_{TLi}, Λ_{CRi} et $\Lambda_{CR/PLi}$ des défaillances composant associées aux *Ci* sont également fournis par l'utilisateur (Table 8).

	COM Parallèle	COM Série	COM Réduit	MON Parallèle	MON Série	MON Réduit
Λtotal	5,0E-05	5,0E-05	5,0E-05	5,0E-05	5,0E-05	1,0E-05
ΛTL	4,5E-05	4,5E-05	4,5E-05	4,5E-05	4,5E-05	9,0E-06
ΛCR	2,5E-06	2,5E-06	2,5E-06	2,5E-06	2,5E-06	5,0E-07
ΛCR/PL	5,0E-06	5,0E-06	5,0E-06	5,0E-06	5,0E-06	1,0E-06

Table 8. Les taux de défaillances utilisés sur le système exemple

Les données de sortie comprennent, pour un jeu donné de coupes minimales fonctionnelles, la liste des coupes minimales composant typées issues de la synthèse. Elles comprennent également, pour chaque coupe composant typée, la valeur du majorant de la probabilité qui lui est associée, et pour l'événement redouté, un majorant de sa probabilité d'occurrence par heure.

		Parallèle		Parallèle Dispatch		Série		Série Dispatch		Réduit		Réduit Dispatch	
INTEGRITE	Ordre	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER
	1												
	2	22	2,6E-10	15	1,9E-10	34	4,1E-10	9	7,5E-11	8	3,0E-11	18	7,8E-10
	3			12	1,9E-14			12	1,9E-14	80	7,9E-14		
	4	32	1,3E-18			5	9,1E-19						
	TOTAL	**54**	**2,6E-10**	**27**	**1,9E-10**	**39**	**4,1E-10**	**21**	**7,5E-11**	**88**	**3,0E-11**	**18**	**7,8E-10**

		Parallèle		Parallèle Dispatch		Serie		Serie Dispatch		Réduit		Réduit Dispatch	
DISPONIBILITE	Ordre	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER	Nombre de coupes	Majorant proba ER
	1												
	2			18	8,5E-10			12	7,5E-10	24	3,4E-10	12	1,7E-10
	3	92	9,3E-14	109	1,1E-12	68	7,6E-14	140	1,1E-12			66	2,1E-13
	4	944	1,2E-16			958	1,1E-16	12	4,7E-20	269	1,3E-17		
	TOTAL	**1036**	**9,3E-14**	**127**	**8,5E-10**	**1026**	**7,6E-14**	**164**	**7,5E-10**	**293**	**3,4E-10**	**78**	**1,7E-10**

Table 9. Les coupes composant typées sur le système exemple

La synthèse des coupes minimales fonctionnelles en coupes composant typées a été réalisée par SYNTHECOUPE sur chacun des 12 jeux (variante, configuration, *ER*). Les valeurs utilisées pour les divers taux de défaillances sont indiquées sur la Table 8.

La Table 9 présente les résultats, les coupes d'ordre supérieur à 4 n'étant pas mentionnées. Les tables 10 et 11 montrent les coupes d'ordre 2 fournies par la synthèse typée pour les deux triplets (réduite, dispatch, intégrité) et (parallèle, dispatch, disponibilité). Le taux d'occurrence des événements redoutés est maintenant inférieur à 1E-9 dans tous les cas. Les coupes composant typées sont plus nombreuses que les coupes composant non typées (Table 3) mais les probabilités associées sont plus faibles. La synthèse en coupes composant typées est donc une approche moins majorante que la synthèse non typée.

Il découle de ce qui précède que les 3 variantes étudiées sont acceptables du point de vue de la safety, avec des résultats voisins. Nous choisirons la variante « réduite », pour laquelle la partie MON est suffisamment simple pour être implémentée entièrement en hardware.

6.7 Comparaison entre la synthèse dite « typée » et d'autres approches éventuelles

A notre connaissance, le processus de synthèse typée n'existe pas dans d'autres contextes que SYNTHECOUPE. La notion de « type » pour les défaillances fonctionnelles ou composant est propre à notre démarche, ainsi que le processus permettant d'évaluer des majorants pour les probabilités d'occurrence des événements redoutés.

7. Conclusion

Dans cet article, nous avons présenté une méthode d'évaluation des performances de safety atteignables par un système complexe. Cette méthode est particulièrement pertinente pour la phase « d'early validation » du processus de développement. Elle est illustrée sur un exemple réel comprenant 4 senseurs redondants générant des paramètres critiques avec des exigences très dures d'intégrité et de disponibilité.

Dans la section 3, nous avons décrit ces exigences et les trois variantes d'architecture COM-MON envisagées pour notre système exemple. Dans la section 4, nous avons présenté la première étape de la démarche, qui inclue notamment la construction d'un modèle orienté safety pour chacune des trois variantes. Ces modèles appartiennent à la classe des *modèles dédiés*, par opposition aux *modèles étendus* mentionnés dans la section 1. La modélisation des composants physiques est fondée sur la modélisation, puis l'assemblage, des *opérateurs fonctionnels élémentaires* compris dans ces composants. Le modèle système est lui-même obtenu par l'assemblage des modèles des composants physiques. La première étape de la démarche se conclue par la génération automatique des 12 jeux de coupes ou séquences fonctionnelles minimales provoquant les *ER*, correspondant aux 12 triplets (variante parallèle/série/réduite, configuration nominale/dispatch, *ER* intégrité/disponibilité). Ces coupes sont non quantifiées, leur nombre est important (dizaines voire centaines de milliers).

Dans la section 5, nous avons présenté l'évaluation dite « de base » des performances de safety correspondant à chacun des 12 jeux. Cette évaluation s'effectue grâce au logiciel SYNTHECOUPE développé par Thales. Elle se fonde, pour chaque jeu, sur la *synthèse des coupes fonctionnelles de ce jeu en coupes minimales composant quantifiées*. Chacune de ces coupes composant « couvre » l'ensemble des coupes fonctionnelles dont les éléments (défaillances fonctionnelles) appartiennent à l'un des composants intervenant dans la coupe

composant. La probabilité associée à une coupe composant est le produit des probabilités de défaillance associées à chacun des composants la constituant. La probabilité de défaillance associée à un composant est majorée par le taux de défaillance total de ce composant. Enfin, la probabilité par heure d'occurrence de l'*ER* est majorée par la somme des probabilités associées aux coupes composant. Nous avons constaté au terme de cette seconde étape dite « de base » que la plupart des majorants ainsi obtenus pour l'*ER* étaient supérieurs à l'objectif de 1E-9, ce qui nous interdit de conclure à la tenue de cet objectif.

			COM1	MON1	COM2	MON2	COM3	MON3	COM4	MON4	Probas
REDUITE DISPATCH	INTEGRITE	1	CR		CR						6,3E-12
		2	CR		CR/PL						1,3E-11
		3	CR		TL						1,1E-10
		4	CR				CR				6,3E-12
		5	CR				CR/PL				1,3E-11
		6	CR				TL				1,1E-10
		7	CR	CR/PL							2,5E-12
		8	CR/PL		CR						1,3E-11
		9	CR/PL				CR				1,3E-11
		10	TL		CR						1,1E-10
		11	TL				CR				1,1E-10
		12			CR		CR				6,3E-12
		13			CR		CR/PL				1,3E-11
		14			CR		TL				1,1E-10
		15			CR	CR/PL					2,5E-12
		16			CR/PL		CR				1,3E-11
		17			TL		CR				1,1E-10
		18					CR	CR/PL			2,5E-12
											7,8E-10

Table 10. Coupes composant typées – réduite, dispatch, intégrité

			COM1	MON1	COM2	MON2	COM3	MON3	COM4	MON4	Probas
PARALLELE DISPATCH	DISPONIBILITE	1	CR		CR						6,3E-12
		2	CR		CR/PL						1,3E-11
		3	CR		TL						1,1E-10
		4	CR				CR				6,3E-12
		5	CR				CR/PL				1,3E-11
		6	CR				TL				1,1E-10
		7	CR/PL		CR						1,3E-11
		8	CR/PL		CR/PL						2,5E-11
		9	CR/PL				CR				1,3E-11
		10	CR/PL				CR/PL				2,5E-11
		11	TL		CR						1,1E-10
		12	TL				CR				1,1E-10
		13			CR		CR				6,3E-12
		14			CR		CR/PL				1,3E-11
		15			CR		TL				1,1E-10
		16			CR/PL		CR				1,3E-11
		17			CR/PL		CR/PL				2,5E-11
		18			TL		CR				1,1E-10
											8,5E-10

Table 11. Coupes composant typées – parallèle, dispatch, disponibilité

Dans la section 6, nous avons présenté une version dite « typée » et moins majorante de la synthèse des coupes. Nous avons considéré 3 défaillances composant au lieu d'une seule générique pour l'approche « de base ». Ces 3 défaillances sont la « perte totale » (type *TL*) qui désigne la « passivation » du composant, le « calcul corrompu » (type *CR*) qui désigne la présence d'informations corrompues au sein du composant, et enfin la « perte partielle » (type *PL*). Les probabilités d'occurrence associées à ces défaillances sont fournies par l'utilisateur. De manière analogue nous avons considéré les 3 types *L*, *E* ou *F* caractérisant les défaillances fonctionnelles de chaque composant. Pour chacune de ces défaillances fonctionnelles, l'utilisateur de SYNTHECOUPE doit donc spécifier le type auquel elle appartient. L'algorithme de synthèse est ensuite redéfini pour tenir compte de ces nouvelles données d'entrée, les types caractérisant les défaillances composant étant déterminés automatiquement par l'outil. Les coupes minimales composant ainsi générées sont des combinaisons de défaillances composant typées. En règle générale, les valeurs choisies pour les probabilités relatives à « calcul corrompu » et à « perte partielle » pourront être nettement moins élevées que les taux de défaillances totaux des composants.

Au terme de la démarche, nous avons obtenu un ensemble de 12 majorants pour les probabilités d'occurrence des *ER*, correspondant aux 12 jeux de coupes fonctionnelles minimales. Ces majorants étant inférieurs à l'objectif de 1E-9, nous pouvons conclure à la tenue des exigences de safety pour nos 3 variantes, nos 2 configurations et nos 2 événements redoutés. Au final, la troisième variante dite « réduite » est préférable aux deux autres. Ses performances de safety sont en effet suffisantes, et la partie MON qu'elle comprend est réalisable en hardware uniquement, ce qui n'est pas le cas des MON des deux autres variantes.

Rappelons pour conclure que ces résultats supposent l'absence de pannes dormantes persistant d'une mission à la suivante, et la ségrégation physique des composants. Des travaux futurs, déjà très avancés, permettront d'étendre le processus de synthèse à d'autres contextes, levant ainsi ces restrictions.

Remerciements

L'auteur remercie Dominique Bouard, Christophe Garnavault et Marion Morel pour leurs contributions à ces travaux, et la société Thales pour leur financement.

Références

1. Arnold A., G. Point, A. Griffault, A. Rauzy, 2000, The Altarica Formalism for Describing Concurrent Systems, in Fundamenta Informaticae, Vol. 34, p. 109-124.
2. Åkerlund O., P. Bieber *et al*, 2006, ISAAC, a Framework for Integrated Safety Analysis of Functional, Geometrical and Human Aspects, in 3rd European Congress on Embedded Real Time Systems (ERTS), Toulouse, France.
3. Bozzano M., A. Villafiorita, *et al*, 2003, ESACS, an Integrated Methodology for Design and Safety Analysis of Complex Systems, in European Safety and Reliability Conference (ESREL), Maastricht. Balkema Publishers.
4. Papadopoulos Y., 1999, Hierarchically Performed Hazard Origin and Propagation Studies, in 18th International Conference on Computer Safety, Reliability and Security (SAFECOMP), Toulouse, France. Springer-Verlag.
5. Fenelon P., J. McDermid, July 1993, An Integrated Toolset for Software Safety Analysis, Journal of Systems and Software, p. 279-290.
6. Rauzy A., December 2001, Mathematical Foundation of Minimal Cutsets, in IEEE Transactions on Reliability. IEEE Reliability Society, Vol. 50, Num. 4, p. 389–396.
7. Avizienis, Laprie, Randell and Landwehr, Jan-March 2004, Basic Concepts and Taxonomy of Dependable and Secure Computing, in IEEE Transactions on Dependable and Secure Computing, Vol 1, n°1.
8. Voirin J.-L., 2010, Method & Tools to Secure and Support Collaborative Architecting of Constrained Systems, 27th International Congress of the Aeronautic Sciences.