

Sophia un environnement pour l'analyse de sûreté à partir de modèles Sophia framework for model-based safety analysis

Nataliya Yakymets, Julho Y. Munoz, Agnes Lanusse

▶ **To cite this version:**

Nataliya Yakymets, Julho Y. Munoz, Agnes Lanusse. Sophia un environnement pour l'analyse de sûreté à partir de modèles Sophia framework for model-based safety analysis. IMDR - 19ème congrès de maîtrise des risques et sûreté du fonctionnement, Oct 2014, Dijon, France. cea-01810452

HAL Id: cea-01810452
https://cea.hal.science/cea-01810452

Submitted on 7 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sophia un environnement pour l'analyse de sûreté à partir de modèles
Sophia framework for model-based safety analysis

Yakymets N., Munoz Julho Y., Lanusse A.
Commissariat à l'énergie atomique et aux énergies alternatives (CEA)
CEA, LIST, Laboratoire d'Ingénierie dirigée par les modèles pour les Systèmes Embarqués,
91191 Gif-sur-Yvette CEDEX, France
+33 (0)1 69 08 32 93
nataliya.yakymets@cea.fr

Résumé

Nous présentons Sophia un environnement de modélisation et d'analyse dédié aux analyses de sûreté étroitement couplé à l'outil de modélisation système Papyrus. Il permet d'exploiter les différentes facettes de modélisation offertes par SysML [7] et d'intégrer des fonctionnalités complémentaires pour conduire des analyses FTA (Fault Tree Analysis) et FMEA (Failure Mode and Effects Analysis) qui sont décrites dans la suite de ce papier.

Summary

We propose a modeling framework, called Sophia, for model-based safety analysis. Sophia uses the Papyrus editing tool to leverage features of SysML modeling language and includes facilities to perform FTA (Fault Tree Analysis) and FMEA (Failure Mode and Effects Analysis).

1. Objectifs

Les systèmes critiques doivent satisfaire de hauts niveaux d'exigences imposées par les normes de certification. Ces dernières préconisent un processus organisé en grandes étapes comme l'analyse des dangers[1][2] et l'analyse préliminaire de risques[3], et recommandent l'utilisation de méthodes classiques telles que l'analyse des modes de défaillance et de leurs effets (FMEA)[4], les analyses par arbres de défaillance (FTA) [5] ou par arbres d'événements, etc. Ces méthodes bien connues des ingénieurs de sûreté, sont cependant lourdes à mettre en œuvre et deviennent mal adaptées face à la croissance de la complexité des systèmes et aux contraintes de délais imposées par la compétitivité élevée dans les secteurs concernés. Il devient nécessaire d'accompagner l'activité d'analyse à l'aide d'outils adaptés et surtout plus proches du processus de conception.

Dans ce contexte, il peut être très intéressant de tirer parti des avancées dans le domaine de l'ingénierie dirigée par les modèles (IDM ou MBSE) pour mettre en place des stratégies coopératives d'évaluation de sûreté[1] (Safety Assessment ou SA) grâce à un couplage fin avec les environnements de modélisation système.

Nous présentons Sophia, un environnement de modélisation et d'analyse dédié aux analyses de sûreté étroitement couplé à l'outil de modélisation système Papyrus. Il permet d'exploiter les différentes facettes de modélisation offertes par SysML [7] et d'intégrer des fonctionnalités complémentaires pour conduire des analyses FTA et FMEA qui sont décrites dans la suite de ce papier.

2. Contexte

Les approches à base de modèles offrent un cadre intéressant pour développer des systèmes critiques. D'une part, elles facilitent une séparation des préoccupations qui permet une meilleure appréhension de la complexité des systèmes ; de plus, elles permettent en traitant les modèles à un haut niveau d'abstraction de pouvoir commencer à évaluer/valider et/ou simuler les modèles très tôt dans le processus de conception. Ces caractéristiques permettent d'enrichir l'environnement de modélisation avec des points de vue d'analyse spécifiques tels que l'évaluation de sûreté qui nous intéresse ici plus particulièrement et de l'interfacer avec des outils dédiés. Ainsi en utilisant les mécanismes d'extension d'UML, il est possible de définir un langage orienté analyse de sûreté et de l'intégrer facilement à un outil de modélisation existant tel que Papyrus puis si nécessaire d'exporter le modèle annoté vers des outils externes d'analyse formelle. Il devient donc possible d'effectuer des analyses de sûreté à base de modèles en intégrant les méthodes et outils classiques de SA dans un MBSE.

Durant les phases amont de modélisation système, les objectifs de l'évaluation de sûreté sont dirigés principalement vers l'identification des processus de propagation pouvant conduire à l'apparition des événements redoutés identifiés lors de l'analyse préliminaire des dangers. Le résultat attendu est de proposer des moyens de protection (barrières) et/ou des exigences de sûreté permettant de rendre plus robustes les architectures système. Traditionnellement les méthodes employées sont les analyses FTA et FMEA, le plus souvent outillées dans des outils bureautique de type tableur. Cette manière de faire atteint aujourd'hui ses limites.

Les approches à base de modèles paraissent attractives pour aller plus loin. Toutefois, il faut distinguer les approches à base de modèles formels, des approches à base de modèles système. Si les premières sont nombreuses et bien outillées Saml [8], AltaRica [9], HiP-HOPS[10], FSAP/NuSMV [11]), elles sont relativement difficiles à utiliser directement car elles nécessitent un effort important de transposition de la documentation système dans un langage formel. Les approches MBSE proposent une intégration fine de ces outils dans des environnements de modélisation système et donc semblent plus appropriées. Parmi les travaux les plus notables dans cette deuxième catégorie, nous pouvons citer les travaux autour du langage EAST-ADL et ceux effectués dans le cadre du projet [12] autour de SysML.

L'environnement Sophia propose de tirer parti des avancées dans le domaine des approches formelles en les associant dans une démarche d'analyse basée sur une modélisation système. Ainsi Sophia vise à réduire le fossé entre la modélisation système et la modélisation de sûreté et à permettre l'utilisation d'outils dédiés intégrant les techniques de FTA et FMEA dans un environnement unifié d'IDM construit sur l'outil Papyrus (éditeur UML/SysML).

3. Méthode

Le processus d'évaluation de sûreté supporté par Sophia est basé sur l'analyse formelle d'un système modélisé en UML/SysML (ou un DSL compatible). L'analyse s'effectue sur la base d'une abstraction du système (vue dysfonctionnelle) qui est transformé en machine à état puis exploré à l'aide de différentes techniques formelles de vérification. La méthodologie propose un certain nombre d'étapes (Figure 1).

[1] Par souci de concision nous avons choisi ici de traduire le terme anglais « safety » par « sûreté ». Nous aurions pu faire le choix de sécurité fonctionnelle ou sécurité-inocuité.

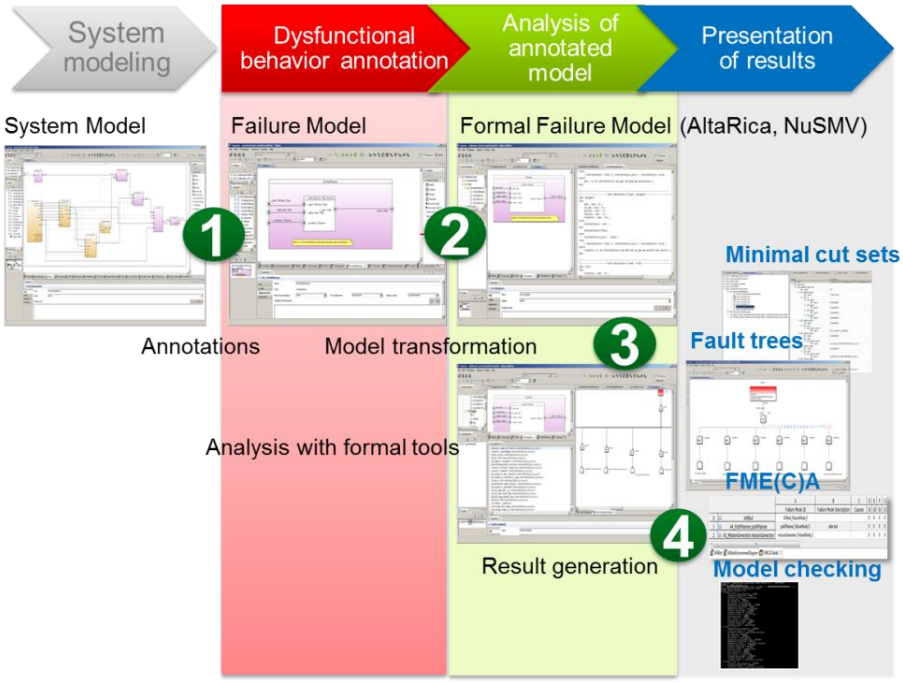


Figure 1. Sophia méthodologie

1. Conception Système. L'architecture système est modélisée avec la plateforme Papyrus en utilisant les diagrammes SysML et tout particulièrement les Internal Block Diagrams (IBD).
2. Annotation du Système. Le modèle SysMLest annoté de façon à définir un modèle dysfonctionnel du système. Pour ce faire, les différents modes de défaillances possibles associés aux composants du système (Blocks) sont identifiés et attachés au modèle et un modèle de propagation est défini.
3. Analyse de Sûreté. L'information entrée par l'ingénieur de sûreté est utilisée directement pour réaliser des tables FMEA mais est aussi exploitée avec des annotations de propagation pour produire les arbres de défaillance. Pour cela le modèle dysfonctionnel est transformé vers le langage AltaRica [9]. Sophia utilise ensuite l'outil ARC [14] pour générer des coupes minimales et créer automatiquement les arbres de défaillance. Lorsque des données statistiques sont disponibles pour les défaillances élémentaires, des analyses quantitatives peuvent être conduites en utilisant l'outil xFTA [15].
4. Présentation des Résultats. Les résultats des analyses peuvent être intégrés dans la documentation générée par l'outil ou présenté dans l'environnement de modélisation. Ainsi les arbres de défaillance peuvent être affichés dans Papyrus grâce à un profil dédié basé sur le format Open-Psa[2].

4. Résultats

Sophia est un environnement modulaire d'aide aux études d'évaluation de sûreté. Il offre plusieurs types de fonctionnalités : support à la modélisation, automatisation de l'exploitation de techniques formelles pour produire les résultats attendus, et support à la génération de documentation. Dans ce papier nous présentons plus particulièrement les fonctionnalités d'analyses FMEA et FTAsComme indiqué dans la Figure 2, les modules Sophia FMEA et FTA utilisent des profils pour annoter un modèle avec des informations liées aux analyses de risques (description des modes de défaillance, des événements redoutés,…) et permettent de produire des arbres de défaillances. Le module Sophia-FMEA permet d'accompagner la conduite d'analyses FMEA et de produire automatiquement des tables FMEA dans l'environnement de modélisation système, de les visualiser et les enrichir interactivement et de générer des rapports FMEA. Le module Sophia-FTA permet de conduire des analyses FTA qualitatives (arbres de défaillances et calcul des coupes minimales) et quantitatives (calcul de la probabilité de l'événement redouté sommet de l'arbre, et les analyses de sensibilité et importance).

Dans la suite, nous nous appuyons sur un cas d'utilisation robotique développé dans le cadre du projet Proteus pour illustrer la présentation des fonctionnalités de Sophia. L'exemple étudié concerne un scénario défini par un challenge robotique (RYC, Robotic Youth Challenge) et déployé sur un robot WifiBot. Ce challenge concerne une application de robotique mobile, dont le but est d'explorer un espace inconnu et de construire dynamiquement une carte de cet espace. Le scénario principal vise une exploration dans un environnement structuré en extérieur à la recherche d'une cible. L'architecture du modèle système est présentée dans la Figure 3 ci-dessous. Le niveau principal comporte neuf composants : Mission Generator génère les objectifs de missions RYC, Path Planner calcule un chemin pour le robot en utilisant une carte globale, des informations sur la mission et la position courante, Navigator définit une trajectoire locale pour le composant Pilot en utilisant une carte locale, Pilot calcule les vitesses des roues droite et gauche et détermine les points de passage que le robot doit atteindre pour suivre la trajectoire fournie en entrée, Servoings transforment les commandes de vitesse au format (tics) utilisé par le robot, Sensors capture les informations de l'environnement, Proximetry construit une carte en coordonnées polaires à partir des mesures des capteurs brutes, Local Map construit une carte en coordonnées Cartésiennes 2D sur laquelle sont placés les obstacles rencontrés, Global Map construit une carte absolue de la scène.

La modélisation du système est faite en RobotML un langage dédié (DSML ou Domain Specific Modeling Language) pour la modélisation de systèmes robotiques. L'implémentation de ce DSML est faite à l'aide d'un profil UML (mécanisme d'extension d'UML) qui permet dans ce cas de spécialiser le langage de modélisation pour un domaine métier, ici la robotique. Des types prédéfinis peuvent être utilisés ainsi que des bibliothèques de composants (par exemple le Type générique Sensor et des capteurs spécifiques). Il est important de noter que Sophia utilise aussi des profils mais pour annoter le modèle à des fins d'analyse. Sophia s'adapte très naturellement à des modélisations faites à l'aide de DSMLs réalisés sur la base d'UML et de son mécanisme d'extension.

2 Le format Open-Psa est une proposition de standard de modélisation pour les analyses de fiabilité. Il comprend notamment une section pour la modélisation d'arbres de défaillance (voir http://open-psa.org/).

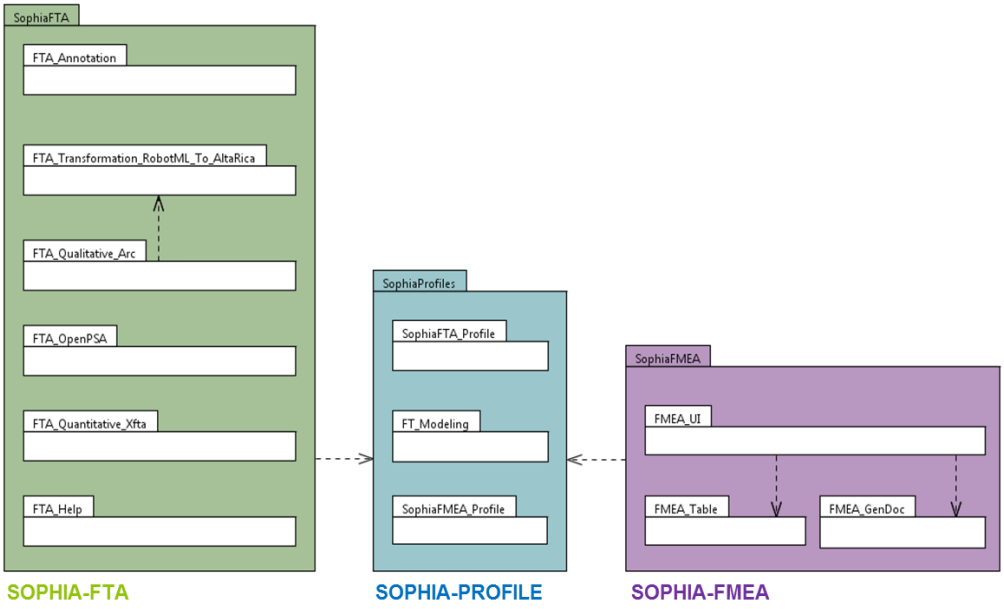


Figure 2. Architecture des modules FMEA et FTA de Sophia

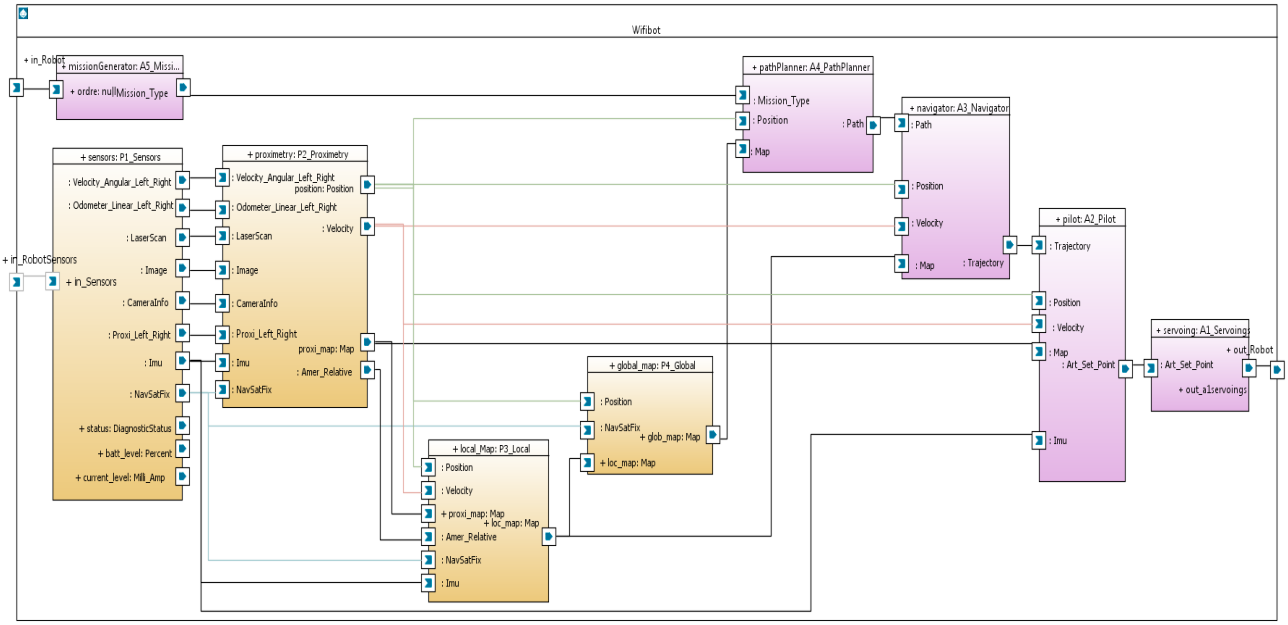


Figure 3. Architecture définie pour l'application Challenge RYC (modélisation RobotML)

4.1. FMEA

Dans le cas des FMEAs, la part essentielle de la contribution porte sur l'effort de modélisation et la production de tables de FMEAs qui sont construites sur la base de l'architecture du système et peuvent être enrichies automatiquement à chaque déclaration de nouveau mode de défaillance (Figure 4). Le module Sophia-FMEA étend l'approche FMEA et fournit également la possibilité d'étudier la criticité des modes de défaillances pour les composants matériels. Le résultat de cette analyse est présenté dans des tables d'AMDEC (Analyse des modes de défaillance de leurs effets et de leur criticité) que nous désignerons ici par FMECA(Failure Mode, Effects and Criticality Analysis).

La criticité d'un mode de défaillance est calculée sur la base du produit des paramètres de sévérité, détectabilité et occurrence. Ces critères sont définis relativement à des échelles qualitatives qui peuvent être définies dans les standards de sûreté de fonctionnement des domaines spécifiques (robotique, aéronautique, automobile …), mais sont également souvent redéfinies en interne par les industriels. Pour cette raison, Sophia-FMEA utilise des bibliothèques dédiées pour quantifier les valeurs de sévérité, détectabilité et occurrence en fonction d'un domaine spécifique et ou d'un contexte industriel. Au cours de l'analyse FMECA, un seuil de criticité est défini pour chaque bloc, il correspond à une borne supérieure de criticité inacceptable. Si la valeur de criticité estimée à travers l'analyse FMECA du bloc, celui-ci est considéré comme critique. L'analyse FMECA aide donc à déterminer les composants critiques de l'architecture qui peuvent ensuite être analysés, et protégés par la mise en place de barrières ou développés selon des règles de bonnes pratiques conformes aux recommandations des standrads de sûreté de fonctionnement.

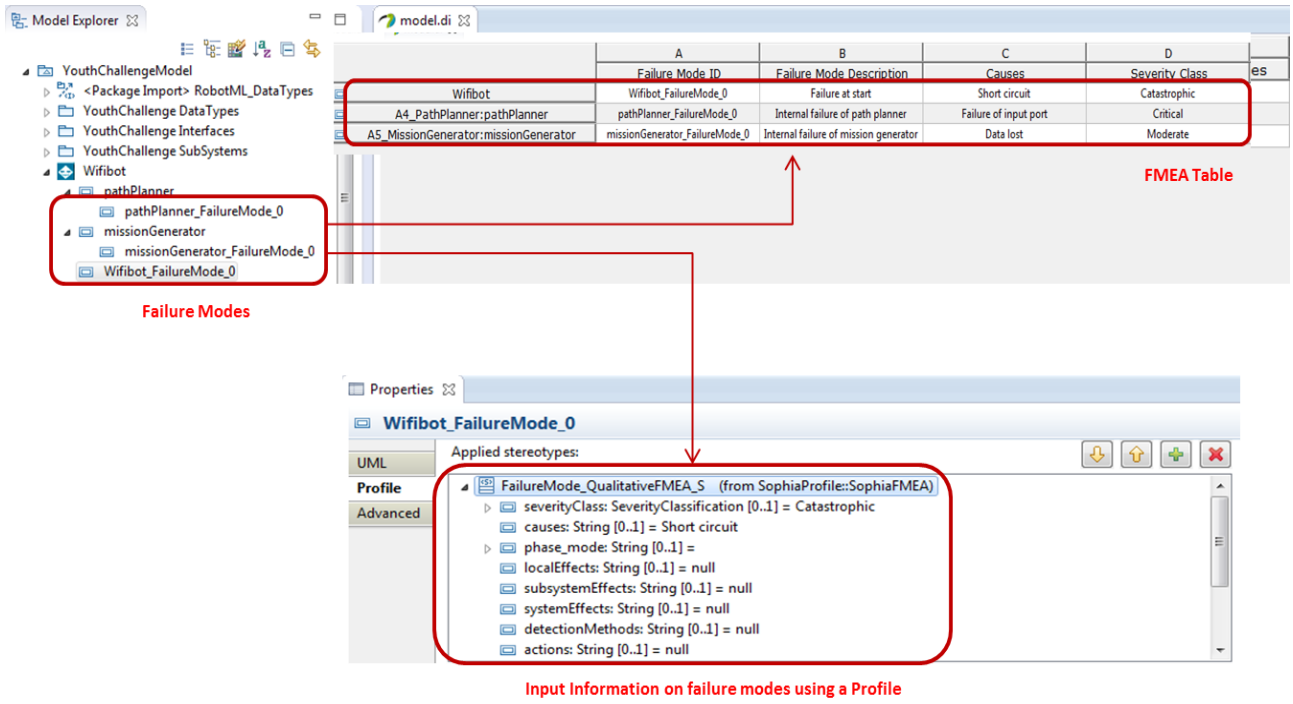


Figure 4. Sophia-FMEA modeleur

Les analyses FMEA conduites à l'aide de Sophia comportent plusieurs étapes :

1. Modélisation Système.

Chargement ou création d'un modèle de système à l'aide du langage SysML (ou d'un langage dédié à un domaine comme par exemple RobotML pour la robotique). Le système doit être décrit en utilisant au moins les diagrammes suivants :

- Les diagrammes de définition de blocks (Block Definition Diagrams ou BDD) qui définissent les types utilisés (Blocks) pour la modélisation du système et leur organisation hiérarchique par l'utilisation de relations de composition entre les blocks ;
- Les diagrammes internes de blocks (Internal Block Diagrams ou IBD) qui permettent la description des interactions entre les constituants du système (appelés Parts) via la définition de connecteurs entre les ports d'entrées/sorties de ces constituants.

2. Annotation du modèle pour la spécification dysfonctionnelle des Blocks.

Cette étape consiste à appliquer le profil Sophia-FMEA qui permet d'introduire les modes de défaillance possibles du système dans le modèle. Le comportement dysfonctionnel (failure behavior) du système est ainsi défini de manière modulaire en annotant chaque type de constituant du système. Le modeleur Papyrus est spécialisé de manière à permettre d'entrer les caractéristiques de ces modes de défaillance grâce aux propriétés de stéréotypes définies dans le profil. Ainsi, pour chaque Block système une liste de modes de défaillance possibles est créée. Il est à noter que si le système comporte plusieurs instances de ce type de constituant, cette caractérisation dysfonctionnelle du Block n'est faite qu'une fois.

3. Préparation de la configuration des seuils pour les analyses de criticité.

A cette étape on spécifie le seuil de criticité des Blocks à prendre en compte pour l'analyse. Un seuil de criticité est une valeur limite qui détermine le niveau au-delà duquel le Block est considéré comme critique. Si la valeur de criticité estimée lors de l'analyse FMEA pour le block est supérieure à ce seuil, le Block sera considéré comme critique.

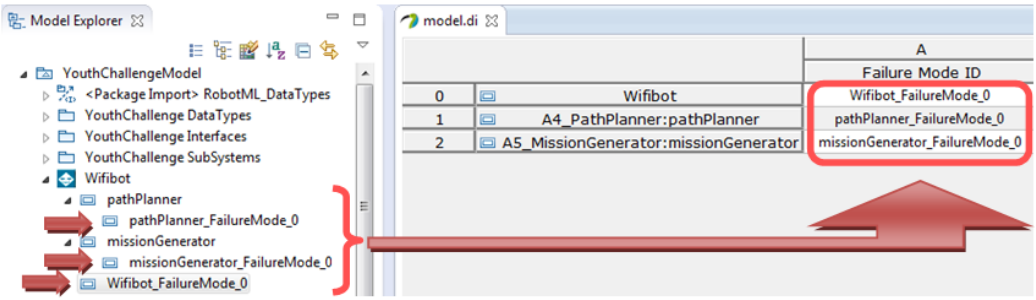
4. Analyse FMEA, FMECA et génération du rapport associé.

L'analyse consiste à compléter les informations de modes de défaillance qui peuvent être éditées dans une table du modeleur Sophia. Ensuite un rapport FMEA est généré automatiquement, il reprend toutes les informations de défaillance ajoutées au modèle en tenant compte de l'architecture du système.

Cette fonctionnalité réduit considérablement le temps nécessaire à la construction de FMEAs et garantit de plus la cohérence par construction entre celles-ci et l'architecture du système. Les tables FMEA peuvent ensuite être exportées dans des tableaux Excel ou exploitées directement dans l'outil de génération de documentation de Sophia. Ces tables utilisent par défaut les attributs correspondants à la norme EN 60812:2006 [NF 2006].

Le module FMEA implémenté présente les caractéristiques suivantes :

- Les modes de défaillance d'une fonction sont automatiquement réutilisables dans plusieurs systèmes où cette fonction est présente. Autrement dit, les modes de défaillance d'une fonction seront toujours réutilisables dans les systèmes où cette fonction est utilisée.
- Les tableaux sont toujours cohérents avec le modèle (Figure 5). Par exemple, si une fonction est supprimée du système, ses modes de défaillance sont enlevés du tableau FMEA.



- **Figure 5.** Synchronisation entre le modèle et le tableau FMEA

- Les tableaux FMEA et FMECA d'un même système sont toujours synchronisés entre eux (Figure 6). Donc, il n'est pas nécessaire de refaire entièrement un autre tableau.

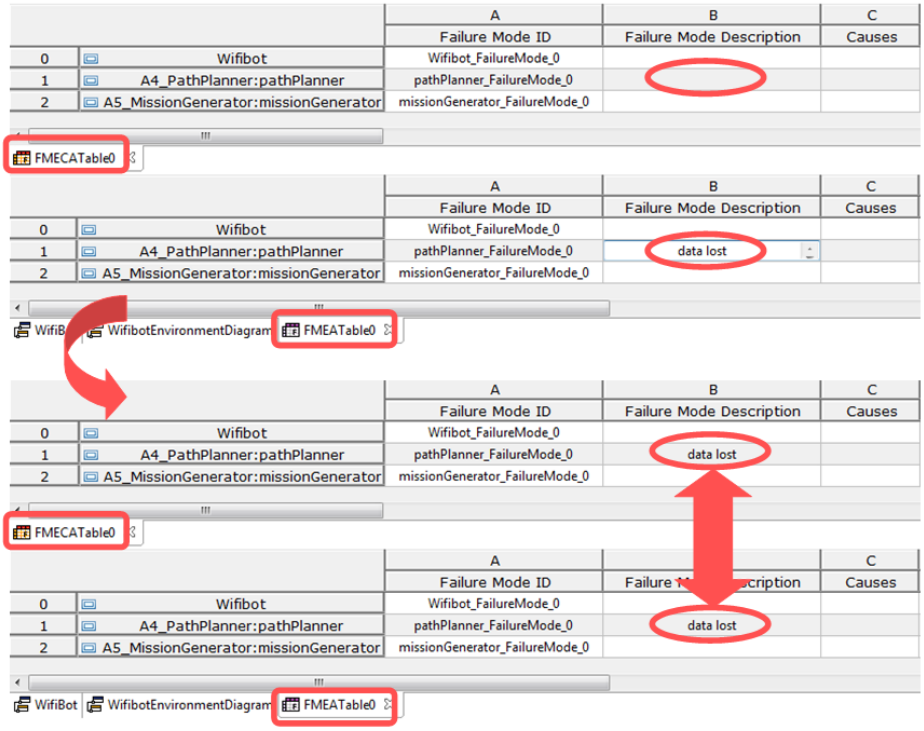


Figure 6. Synchronisation entre le tableau FMEA et FMECA

- Les tableaux utilisent par défaut les attributs inspirés de la norme EN 60812:2006 [NF 2006].
- Exportation de tableau en format Excel (.xls).

Grâce à ces caractéristiques, le temps nécessaire à la construction de FMEAs est réduit considérablement et garantit de plus la cohérence par construction entre celles-ci et l'architecture du système.

4.2. FTA

Le workflow supportant l'analyse FTA à l'aide de Sophia-FTA est décrit dans la Figure 7. Il respecte la méthodologie présentées ci-dessus et comporte plusieurs étapes :

- modélisation/conception système ;
- spécification dysfonctionnelle par annotation du modèle ;
- transformation du modèle vers le langage AltaRica (dataFlow) ;
- analyses FTA qualitative and quantitative ;
- présentation des résultats et génération de rapports.

Dans cet exemple, nous considérons l'événement redouté : "Le robot RYC ne suit pas la commande". Cet evénement sera l'événement sommet de l'arbre de défaillance.

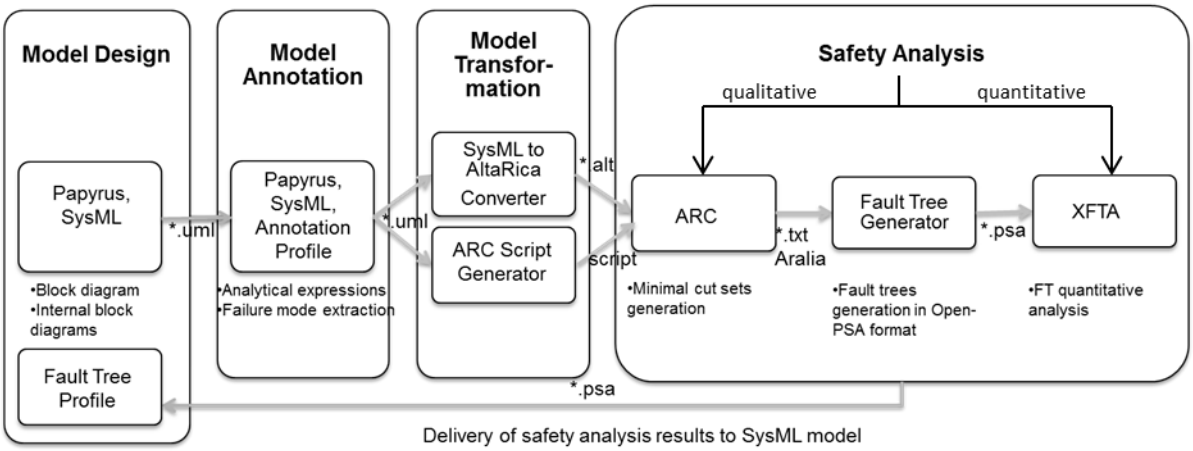


Figure 7. Sophia-FTA workflow

Pour effectuer une analyse FTA avec Sophia, le model RYC de la Figure 3 est annoté pour définir un comportement dysfonctionnel appelé *failure behavior*. Le *failure behavior* est repésenté de manière relativement classique à l'aide d'un ensemble d'expressions booléennes décrivant la manière dont des dysfonctionnements peuvent être observés en sortie d'un composant en fonctions de défaillances internes du composant et/ou des déviations éventuelles sur ses entrées. Dans notre approche, nous proposons une analyse multi-niveaux dont seuls les niveaux de détails les plus fins sont annotés de cette manière (les blocs de ce niveau sont

considérés comme des composants de base (BC ou basic components). Il est important de noter qu'il est possible de sélectionner la configuration d'analyse et donc également le niveau de raffinement choisi pour l'analyse.

Le principe d'annotation retenu repose sur le mécanisme de profil d'UML qui permet d'annoter directement le modèle pour conduire les analyses directement sur celui-ci. Les équations dysfonctionnelles sont associées aux ports de sortie des composants à l'aide des annotations définies dans le profil Sophia exploitant le mécanisme d'extension proposé par UML. Celui-ci contient des stéréotypes permettant d'annoter les composants. Par exemple, l'expression associée à la sortie outputPath du BC PathPlanner de la Figure 3 a la forme suivante.

(NOT f) AND Mission_Type AND Position AND Map.

Cette expression spécifie les conditions pour que le composant PathPlanner ne propage pas de comportement dysfonctionnel. La sortie outputPath ne propage pas de comportement dysfonctionnel si (i); la défaillance interne f ne s'est pas produite (i.e. PathPlanner_internal_failure is false), et si les entrées *Mission_Type et Position AND Map* sont toutes deux correctes (value is true).

Ces annotations dysfonctionnelles sont ensuite utilisées pour construire de manière automatique un système de machines à états. Les états de défaillance, les événements et conditions de passage vers ces états sont construits directement à partir des expressions. Le processus de transformation est adapté au langage formel cible. Dans le cas présenté nous utilisons AltaRica. Le comportement dysfonctionnel des composants est directement traduit en nœuds AltaRica (machines à états) et la modélisation du système est obtenue par composition des machines à états en prenant en compte les règles de propagation issues de la topologie des connecteurs du modèle système.
Cette automatisation de la transformation vers le langage formel AltaRica nous permet d'utiliser des outils formels (dans notre cas ARC du LaBRI) pour produire de manière transparente les arbres de défaillance et les coupes minimales à partir du modèle système annoté.
La transformation effectuée par Sophia s'appuie sur les technologies de l'ingénierie direigée par les modèles (IDM). La table I présente les grandes lignes du mapping utilisé. Il est intéressant de noter que cette transformation est exploitable sur des modèles SysSL ou RobotML.
La génération des arbres de défaillance est activée sur sélection d'un événement redouté système identifié lors d'une analyse préliminaire des dangers conformément aux recommandations du standard IEC61508.

TABLE I. Transformation Règles de transformation

Concept	SysML	RobotML	AltaRica	Description
Component type	System Block	Robot	Node main	RS under analysis
Component /Prototype	Block Part	Software, Hardware, RoboticSystem, SensorSystem, ActuatorSystem, CameraSystem, GPSSystem, Object-DetectionSensorSystem, Sensor-Driver, ImageSensorSystem, EngineSystem, WheelSystem, ObjectTrackingSensorSystem, LocalizationSensorSystem, SimulatedSystem Part	Node Field:sub	RS components
Flow variable /Type /Direction	FlowPort /Type /Direction	DataFlowPort / Type / Direction	Field: Flow /bool,integer, float,domain /In , Out	RS ports
Connection components	Connector	Connector	Assertion	Connection between components
Output deviation expression	Stereotyped FlowPort	Stereotyped DataFlowPort	Failure states and events, output assertions	Component failure behavior

En SysML ou RobotML, un système peut être considéré comme un ensemble de composants $C = \{c_0, c_1,..., c_m\}$. Chaque composant du niveau hiérarchique le plus fin d'analyse c_i est défini par un triplet $\{P_{in}, P_{out}, D\}$: un ensemble d'entrées P_{in}, un ensemble de sorties output P_{out}, et un ensemble d'expressions de déviations D liées aux ports de sortie du composant. Chaque expression contient un ensemble de modes de défaillances F et un sous-ensemble d'entrées incorrectes inputs P'_{in} du composant c_i: $\forall\ d_k \in D,\ d_k = \{M, P'_{in} \subseteq P_{in}\}$. Les composants de plus haut-niveau sont décrits comme $c_i = \{P_{in}, P_{out}, SC, I\}$, où SC est un ensemble de sous-composants (appartenant au sous-niveau) du composant c_i et connectés par un ensemble de connecteurs I.

En AltaRica, un système est représenté comme une machine à états composée d'un ensemble de nœuds (*nodes*) $N = \{n_0, n_1,..., n_m\}$. Au niveau de détail le plus fin de la configuration d'analyse, chaque noeud n_i est défini comme un quadruplet $\{In, Out, S, E, T, A\}$ qui contient un ensemble d'entrées In, un ensemble de sorties Out , un ensemble d'états S, un ensemble d'événements E, un ensemble de transitions Tet un ensemble d'assertions A. Un noeud composite d'un niveau supérieur $n_i = \{In, Out, SN, A\}$ contient un ensemble de sous-nœuds SN.

Au cours de la transformation, chaque composant c_i du modèle SysML ou RobotML model est traduit en un nœud AltaRica n_i de la manière suivante : les ports d'entrées et de sorties sont traduits dans les flows correspondants AltaRica $P_{in} \rightarrow In$, $P_{out} \rightarrow Out$; Si c_i contient des sous-composants ceux-ci sont traduits en sous-nœuds $SC \rightarrow SN$; les interconnections entre sous-noeuds sont traduites en ensembles d'assertions associées au nœud; $I \rightarrow A$. Si c_i est un composant du niveau le plus bas de la configuration d'analyse RS, ses modes de défaillance sont extraits de l'expression de déviation définie dans le modèle système annoté et converti en états, événements déclencheurs de transitions: $F \rightarrow S$, $F \rightarrow E$. Les transitions du nœud sont obtenues de la manière suivante $\forall\ t_j \in T\ t_j$ s'écrit: $(s_a=false)$ |- $e \rightarrow (s_a=true)$, où $e \in E$ est un événement résultant de l'occurrence d'un mode de défaillance dans l'état $s_a \in S$.

Dans le langage AltaRica, le comportement des noeuds est décrit en utilisant des expressions analytiques D→A.

Dans la Figure 8, nous montrons les résultats obtenus par cette transformation pour le composant Pilot. Celui-ci est représenté par une *part* dans un Internal Block Definition (IBD) de SysML. Cette *part* est transformée en un *sub-node* AltaRica. Des assertions sont définies à partir des connecteurs reliant les *ports* des *parts*. Par défaut, nous supposons que le robot fonctionne normalement dans l'état initial de l'analyse, Tous les états de défaillance sont positionnés à *false* dans le modèle AltaRica. Les expressions nous

permettent de définir les événements, les états et les transitions du modèle AltaRica. L'occurrence d'une défaillance produit un événement qui déclenche une transition vers un état défaillant.

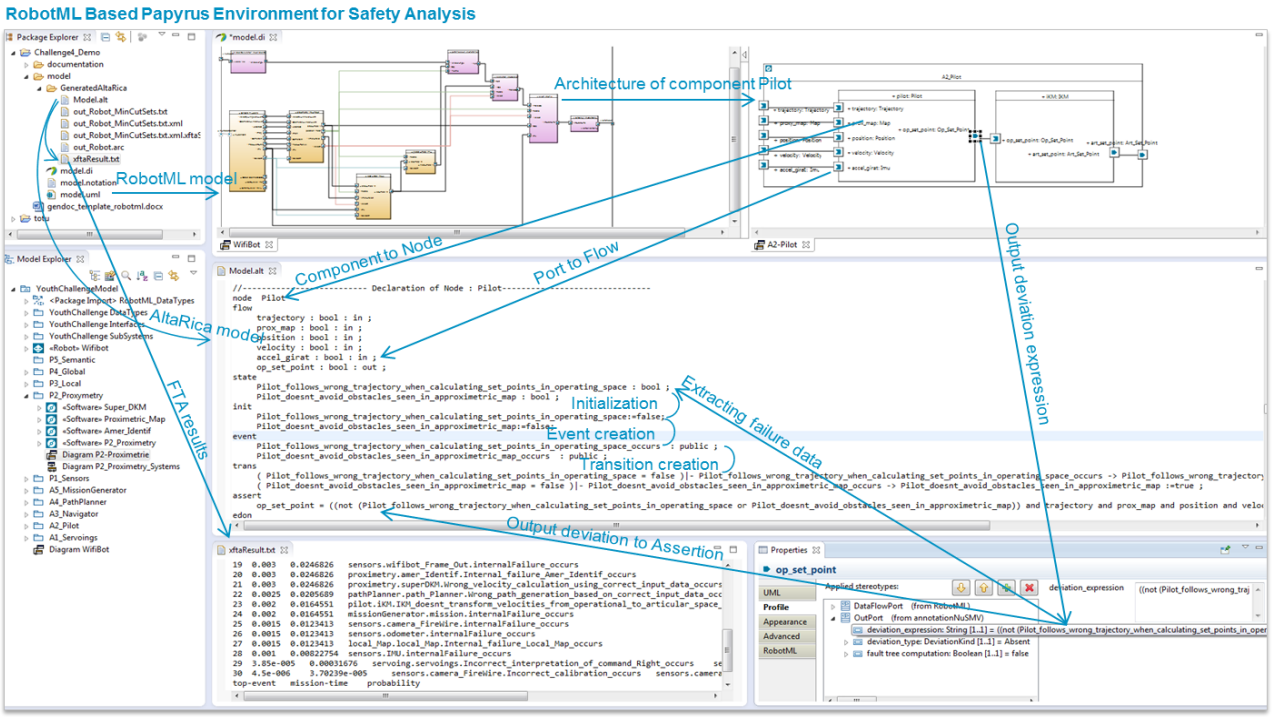


Figure 8. Interface de modélisation pour l'analyse de sûreté. Le modèle RobotML est annoté avec des expressions de deviations, converti en AltaRica et analyses avec xFTA.

Les arbres produits sont enregistrés au format open-PSA. Ce format est un format ouvert destiné aux analyses probabilistes de risques ; il permet de décrire différents types de constructions classiques utilisées par les ingénieurs de sûreté (arbres de défaillances, arbres d'événements, etc.). Nous permettons leur exploitation quantitative via xFTA. De plus, nous avons développé un éditeur graphique qui permet de les afficher dans l'environnement Papyrus grâce à un profil dédié qui implémente un sous-ensemble d'open-PSA pour les arbres de défaillance.

Concrètement, comme indiqué dans la Figure 7, l'environnement Sophia permet la génération automatique d'arbres de défaillance et des coupes minimales associées (analyse qualitative) ainsi que leur exploitation pour des analyses probabilistes de fiabilité (analyse quantitative). Les arbres de défaillances permettent d'identifier les causes possibles de l'apparition d'un événement redouté, cette étude permet de dégager les faiblesses d'une architecture et de proposer des mécanismes de protection (barrières) qui pourront la rendre plus robuste. A partir de ces arbres, il est ensuite possible en exploitant des données statistiques associées aux modes de défaillances identifiés lors de l'analyse FMEA de mener des études quantitatives : calcul de la probabilité d'occurrence d'un événement redouté (événement sommet d'un arbre de défaillance), des analyses de sensibilité et d'importance. L'analyse de sensibilité permet d'évaluer l'impact de la variation de valeurs de paramètres de fiabilité. L'analyse d'importance permet d'évaluer les contributions des différents constituants du système sur le risque global.

L'apport principal de l'environnement est le couplage fin entre les vues conception et analyse qui permet d'obtenir rapidement un retour sur une proposition d'architecture. Cet environnement permet de capitaliser les acquis en matière d'analyse formelle et d'exploiter au mieux les possibilités d'automatisation dans un contexte cohérent et rigoureux. L'effort de modélisation et de formalisation de la part de l'analyste de sûreté est réduit pour laisser la part principale de son activité à son expertise en analyse dysfonctionnelle. Son intégration dans l'environnement Papyrus permet de bénéficier de tous les outils associés d'IDM, notamment pour la vérification de modèles et bien sûr de capitaliser l'effort de modélisation qui peut être réutilisé.

Sophia est un prototype opérationnel. Les développements en cours portent sur la définition d'une méthodologie d'analyse itérative limitant la profondeur d'analyse selon des critères de pertinence de raffinement afin de réduire l'explosion combinatoire.

5. Discussion

La problématique adressée par Sophia est au cœur d'une préoccupation croissante face à l'accroissement du nombre des systèmes soumis à certification et de leur complexité. De nombreux travaux portent donc sur cette thématique. Historiquement, plusieurs démarches ont coexisté jusqu'ici : les démarches d'analyse ad-hoc avec des outils internes généralement basés sur des tableurs, les démarches basées sur des approches formelles pures, les démarches à partir de d'import de modèles.

Les premières les plus fréquentes nécessitent une saisie des données de description du système à partir de documents textuels, vers un fichier de tableur. Les analyses sont alors effectuées par automatisation partielle via des macros ; la deuxième approche consiste à transposer la modélisation du système dans un langage formel toujours à partir de la documentation ; c'est notamment une démarche très répandue pour l'utilisation d'outils industriels du commerce; la troisième approche consiste à automatiser l'export d'un modèle de système vers un outil d'analyse de safety, l'analyse est alors réalisée dans l'outil mais en aveugle puisque tout le modèle n'est pas importé. Outre le coût important de saisie du modèle, le problème majeur avec ces approches est la difficulté à rendre les résultats dans l'environnement de modélisation système.

Des approches plus prometteuses et qui se classent dans la même catégorie que Sophia consistent à annoter le modèle système et à utiliser ces annotations pour définir un modèle dysfonctionnel puis alimenter un outil d'analyse à partir de ces annotations. Cette approche a d'abord été utilisée pour connecter des modèles à l'outil HiP-HOPS en partant de modèles supportant des langages dédiés AADL pour l'avionique ou EAST-ADL pour le domaine automobile. Des expérimentations ont également été faites pour connecter ces modèles à des model-checkers tels que NuSMV. Les initiatives autour du langage EAST-ADL proposent aujourd'hui un modèle d'erreur, c'est aussi le cas de l'erreur annexe d'AADL ces systèmes d'annotations permettent de définir le comportement dysfonctionnel mais pas d'éditer les résultats directement dans le modeleur. Il est généralement proposé un éditeur xml séparé.

Parmi ces approches, on peut citer le projet MAENAD, ou encore le projet SAFE dans le domaine automobile. Par rapport à d'autres travaux, ils essaient d'intégrer une vision de processus d'ingénierie système très intéressante et notamment de prendre en compte différents niveaux d'analyse et les aspects liés aux allocations sur des architectures matérielles.
L'objectif de Sophia est de permettre d'outiller l'activité d'analyse de sûreté et de pouvoir donc aussi bien définir le modèle dysfonctionnel que présenter les résultats arbres de défaillance, FMEA, etc. directement dans l'outil de modélisation et de manière synchronisée avec le modèle d'architecture. Dans l'état actuel des développements, les arbres de défaillances sont affichés dans le modèle. Le format open-PSA utilisé permet l'échange transparent avec les outils d'analyse quantitative et la liaison avec les éléments annotés dans le modèle.
Les principaux choix méthodologiques mis en œuvre dans Sophia concernent les choix de système d'annotations (par équations dysfonctionnelles ou par machines à états profilées) ; le choix du système formel cible et la mise en œuvre de transformations conservant la sémantique du modèle source. Nous avons actuellement des traductions vers AltaRica, NuSMV et Diveristy, un outil interne du CEA. Les résultats de calculs d'arbres sont traduits en open-PSA.
Une démarche permettant une approche itérative d'analyse est implémentée, elle permet de réduire la complexité de l'analyse en sélectionnant des sous-systèmes à analyser.
Surtout, contrairement à toutes ces approche, Sophia intégré dans Papyrus offre un environnement de modélisation SysML complet (dont les exigences) permettant de modéliser des systèmes critiques en exploitant toutes les fonctionnalités de modélisation offertes par le standard et également MARTE pour les aspects dysfonctionnels. Ce support est commun aux ingénieurs système et aux ingénieurs safety, ce qui facilite le dialogue entre spécialistes et également l'intégration de diagrammes pertinents et synchronisés pour la présentation des résultats.

6. Conclusion

Les travaux décrits ici s'inscrivent dans un cadre général d'aide à l'activité d'analyse conduite par les ingénieurs de safety dans l'industrie. L'idée sous-jacente est de permettre d'exploiter les atouts des principes et techniques d'Ingénierie Dirigée par les Modèles en les intégrant dans un processus cohérent d'accompagnement des activités classiques d'analyse de sûreté et d'aide à la constitution de dossiers de sécurité. La méthodologie et l'environnement Sophia s'appuie sur le standard générique IEC61508 et propose des spécialisations compatibles avec des normes filles dans différents domaines (ferroviaire ISO 50128, automobile ISO 26262, robotique ISO/DIS 13482). La méthodologie s'intéresse tout particulièrement aux phases précoces de conception. Elle permet de réduire la complexité et la charge liée à ces activités en automatisant une partie du processus d'évaluation et en concentrant les efforts d'analyse sur son cœur de métier. Surtout, elle permet de conserver la visibilité sur le modèle système et notamment ses aspects comportementaux
L'utilisation de Sophia permet de réduire l'écart entre l'activité de modélisation système et les outils d'évaluation de sûreté permettant ainsi de mieux gérer les coûts et les contraintes d'ingénierie. En associant très tôt analyse et conception il est possible de proposer des solutions architecturales mieux adaptées et plus robustes. Les travaux effectués dans Sophia s'apparentent aux initiatives développées autour des outils tels que HiP-HOPS, FSAP/NuSMV, KB3, SAML, mais l'objectif n'est pas de développer de nouveaux outils d'analyse mais de créer des passerelles permettant à l'ingénieur de safety d'utiliser de tels outils de manière transparente tout en étant assuré de la garantie de la conservation sémantique de son modèle lors . Grâce à l'utilisation de mécanismes de profils et à la gestion des résultats d'analyse. L'utilisateur peut voir les résultats d'analyse affichés dans son environnement et produire de la documentation associée. Le support dans l'outil Papyrus permet de spécialiser encore l'affichage selon des points de vue propres au métier d'analyse de sûreté de fonctionnement et les processus métier internes.

Références

[1] IEC 61508, Functional Safety of Electrical/ Electronic/Programmable Electronic Safety-related Systems, International Electrotechnical Commission (2000).
[2] ARP-4754, Certification Considerations for Highly-Integrated Or Complex Aircraft Systems., Aerospace Recommended Practice (ARP) by Society of Automotive Engineers (SAE) (1996).
[3] IEC 61882, Hazard and operability studies (HAZOP studies) - Application guide, International Electrotechnical Commission (2001).
[4] IEC 60812, Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA), International Electrotechnical Commission (2006).
[5] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. M. III, J. Railsback, Fault Tree Handbook with Aerospace Applications, NASA, 2002.
[6] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies," Rev A, Incose MBSE Focus Group, May 2007.
[7] Object Management Group. OMG Systems Modeling Language, 1st Sept. 2007.
[8] M. Gudemann, F. Ortmeier, "A Framework for Qualitative and Quantitative Formal Model-Based Safety Analysis," Proc. of the 12th IEEE International Symposium on High-Assurance Systems Engineering (HASE), 2010, 132-141.
[9] T. Prosvirnova, M. Batteux, P.-A. Brameret, A. Cherfi, T. Friedlhuber, J.-M. Roussel, and A. Rauzy, "The AltaRica 3.0 project for Model-Based Safety Assessment," In Proceedings of 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS 2013. York (Great Britain). September, 2013.
[10] M. Walker et al., "Compositional Temporal Fault Tree Analysis. Computer Safety, Reliability, and Security," Lecture Notes in Computer Science, vol. 4680, 2007, pp 106-119.
[11] M. Bozzano, Ch. Jochim, "The FSAP/NuSMV-SA Safety Analysis Platform," Int. Journal on Software Tools for Technology Transfer, 2007,v.9, №1, pp. 5-24.
[12] A. Arnold, A. Griffault, G. Point, A. Rauzy, "The AltaRica language and its semantics," In Fundamenta Informaticae, vol. 34, pp 109–124, 2000.
[13] P. David et al., "Reliability study of complex physical systems using SysML," Reliability Engineering and system Safety, Elsevier, pp. 431-450, 2010.
[14] See ARC website: http://altarica.labri.fr/forge/projects/arc/wiki
[15] See XFTA website: http://www.lix.polytechnique.fr/ rauzy/xfta/xfta.htm