Modélisation comportementale et sémantique pour la sûreté de fonctionnement des systèmes autonomes critiques

Nicolas Méric, Julien Niol, Mohamed Tlig

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Modélisation comportementale et sémantique pour la sûreté de fonctionnement des systèmes autonomes critiques

Behavioral and semantic modeling for the safety of critical autonomous systems

Nicolas Méric
Dép. Ingénieries système et logicielle
IRT SystemX
Saclay, France
Email : nicolas.meric@irt-systemx.fr

Julien Niol
Dép. Transport autonome
APSYS
Elancourt, France
Email : julien.niol@apsys-airbus.com

Mohamed Tlig
Dép. Ingénieries système et logicielle
IRT SystemX
Saclay, France
Email : mohamed.tlig@irt-systemx.fr

***Résumé*—Afin de pallier certaines limites liées à la complexité de la simulation du Véhicule Autonome (VA) et rencontrées à l'IRT SystemX, nous proposons un workflow qui repose sur l'utilisation conjointe d'une ontologie et d'un langage de modélisation comportementale.**

***Mots clés*—Sûreté de fonctionnement, Model Based Safety Assessment, Scénarios de validation, Système de conduite autonome, Ontologie, AltaRica**

***Abstract*—To overcome some limitations linked to the complexity of the autonomous vehicle simulation and exposed at the Institute for Technological Research SystemX, we put forward a workflow which uses an ontology and a behavioral modeling language.**

***Index Terms*—Dependability, Safety, Model Based Safety Assessment, Validation Scenarios, Autonomous Driving System, Ontology, AltaRica**

I. Introduction

Le développement du Véhicule Autonome (VA) représente une perspective incontournable pour les constructeurs automobiles. La réalisation du VA exige aussi de lever des verrous scientifiques.

L'un des principaux sujets d'étude concerne la sécurité du VA. Le projet Simulation pour la sécurité du Véhicule Autonome (SVA) de l'IRT SystemX s'inscrit au cœur de cette problématique et donne la possibilité, au sein de ces différentes activités, d'entrevoir les difficultés scientifiques latentes posées par la sécurité du VA. En effet, il a pour objectif de répondre par la simulation numérique au défi posé par la complexité de la démonstration de la sécurité. Cette complexité, liée à la fois au grand nombre de situations que le conducteur rencontre sur la route, leur incertitude, et aux technologies embarquées, rend les validations par des tests en usages réels extrêmement coûteuses, voire impossibles dans certain cas.

Un VA est un véhicule qui est constamment en interaction avec son environnement :

— il interprète son environnement à l'aide de capteurs et d'algorithmes de décision ;
— il agit en fonction de ces interprétations.

Ainsi, l'un des enjeux majeurs est d'être capable de qualifier la sécurité des algorithmes de décision du VA, algorithmes pouvant être perturbés par :

— des perturbations internes : défaillances et fautes aléatoires, pouvant amener le système à avoir un comportement non désiré et potentiellement dangereux ;
— des perturbations externes : événements et phénomènes environnementaux, pouvant conduire le système dans des états ou situations difficilement décidables et potentiellement dangereux pour les occupants d'un VA et les autres usagers.

Ceci implique que la sécurisation du VA englobe plusieurs aspects qui nécessitent des approches spécifiques afin de sécuriser le système vis-à-vis de dysfonctionnements internes (*safety*), protéger le système des attaques extérieures (cybersécurité ou *security*), éviter des prises de décisions erronées par le système dues par exemple à une mauvaise interprétation de son environnement au travers de ses capteurs (« fonctionnel sûr » ou « Safety Of The Intended Functionality (SOTIF) »), et enfin, empêcher le conducteur de faire une mauvaise utilisation du système (les mésusages potentiellement prévisibles) [1].

Ce travail s'inscrit dans la continuité de travaux déjà effectués [2] qui présentent un exemple d'étude de la sûreté de fonctionnement sur un VA avec un modèle comportemental fondé sur le Model Based Safety Assessment (MBSA). Cette approche a montré des résultats encourageants quant à sa capacité à générer des scénarios pertinents. Néanmoins, elle a aussi fait apparaître les limites d'une telle représentation

qui entraîne un problème de complexité. De plus, le formalisme utilisé est peu pratique pour définir la cohérence des concepts environnementaux.

Dans ce qui suit, nous commençons par présenter les travaux connexes qui introduisent le contexte spécifique du VA et les travaux réalisés au sein du projet SVA que nous prolongeons. Nous émettons ensuite une proposition qui dégage les problématiques sous-jacentes du sujet étudié, puis nous présentons nos réalisations et nos résultats pour enfin suggérer quelques perspectives possibles.

II. Travaux connexes

Afin d'interpréter son environnement, le VA dispose de capteurs (radars, caméras…), d'informations cartographiques et de données venant de l'infrastructure ou d'autres véhicules. La norme SOTIF [3] a pour but de saisir les problématiques associées à ce type de défaillances, comme éviter des prises de décisions erronées par le système dues par exemple à une mauvaise interprétation de son environnement [1].

A. Présentation de la norme SOTIF

En effet, la norme SOTIF cherche à répondre à l'absence de risque déraisonnable en raison de dangers résultant d'insuffisances fonctionnelles de la fonctionnalité prévue ou d'une mauvaise utilisation raisonnablement prévisible par des personnes. Nous utiliserons aussi la terminologie « fonctionnel sûr » pour désigner les éléments se rapportant à la description, la prise en compte et la correction des insuffisances fonctionnelles ou du mésusage prévisible par des personnes.

Pour certains systèmes, qui reposent sur la perception de l'environnement externe et interne, il peut y avoir potentiellement des comportements dangereux causés par la fonctionnalité prévue ou une limitation de performance du système qui sont exempts des défauts traités par la série ISO 26262 [4]. Des exemples de telles limitations fonctionnelles correspondant au « fonctionnel sûr » incluent :

— l'incapacité de la fonction à comprendre correctement la situation et à fonctionner en toute sécurité (cela comprend également des fonctions qui utilisent des algorithmes d'apprentissage automatique) ;
— la robustesse insuffisante de la fonction vis-à-vis des variations d'entrée du capteur ou des conditions environnementales diverses.

La notion de limitation de performance traitée recouvre les insuffisances dans la mise en œuvre de la fonctionnalité prévue, c'est-à-dire, perception incomplète de la scène, insuffisance de l'algorithme de décision, performances insuffisantes de l'action.

B. Approches de sécurisation

Cette section reprend les éléments qui sont le fruit de deux approches de sécurisation réalisées auparavant et appliquées sur un système de conduite autonome Traffic Jam Chauffeur (TJC) [5], [2], [6]. Ce dernier s'occupe de la gestion de la trajectoire et de la vitesse du véhicule, dans des conditions de trafic dense (bouchons sur autoroute) et à des vitesses ne dépassant pas les $70\,\mathrm{km\,h^{-1}}$. Pour réaliser ces tâches, le système TJC utilise principalement deux capteurs : une caméra et un radar. Nous présentons donc ci-après une synthèse des deux approches de sécurisation suivantes :

— une étude d'Arbres De Défaillances (ADD) qui est un livrable classiquement recommandé par la norme ISO 26262 dans le cadre des études de sûreté de fonctionnement automobiles. Cette étude revient à proposer une approche par allocation des défaillances ;
— une modélisation comportementale réalisée à l'aide du langage AltaRica et de l'outil SimFia. Cette étude correspond à la modélisation de la propagation des défaillances.

1) Allocation des défaillances: L'objectif du concept de sécurité fonctionnelle est de développer des exigences fonctionnelles sécuritaires à partir des objectifs de sécurité issus de l'Analyse Préliminaire des Risques. Ces exigences sont ensuite allouées à une ou plusieurs fonctions de l'architecture système. Il est à noter qu'il n'y a pas qu'un concept de sécurité mais plusieurs en fonction du niveau d'abstraction où l'on se situe. Dès lors que l'architecture technique qui réalise les fonctions est définie, ces exigences fonctionnelles sont elles-mêmes raffinées en exigences techniques qui sont-elles mêmes raffinées en exigences matérielles et logicielles lorsque la conception a le niveau de détail suffisant. Généralement pour créer ces exigences, des analyses de sécurité sont d'abord réalisées afin de définir des mesures de maîtrise des risques, incluant des mécanismes de sécurité. Pour les systèmes les plus critiques, l'ISO 26262 exige d'utiliser des méthodes complémentaires de types inductives et déductives afin de garantir l'exhaustivité des analyses.

Tous ces éléments permettent une appréhension globale des dysfonctionnements ainsi que de leurs causes. L'objectif de l'Analyse Préliminaire des Risques est d'identifier et classifier les risques qui peuvent être causés par une défaillance du système et de déterminer les objectifs de sécurité associés aux événements redoutés afin de maîtriser l'apparition de ces risques. À partir de l'analyse des défaillances des fonctions du système, on peut établir une liste d'événements redoutés. Ces événements redoutés doivent ensuite être classifiés. Pour cela on utilise la cotation Automotive Safety Integrity Level (ASIL) (figure 1). La cotation ASIL est établie à partir de trois critères : la sévérité, la contrôlabilité et l'exposition [1].

Severity of the harm	Probability of exposure	Controllability C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Figure 1: Matrice de Définition des Niveaux d'ASIL – ISO 26262 (2011) [7]

Ces critères sont divisés en plusieurs niveaux croissants, allant de S1 à S3 pour la sévérité, de C1 à C3 pour la contrôlabilité et de E1 à E4 pour l'exposition. Le plus bas niveau de sécurité est le niveau « Quality Management (QM) ». Ce niveau signifie que le risque associé à un événement dangereux n'est pas déraisonnable et ne nécessite donc pas

de mesures de sécurité. Les ASIL sont rangés par ordre croissant de niveau d'exigence de sécurité de A à D.

Le défaut de cette approche est que, pour définir des objectifs de sécurité (*Safety Goals*), la norme s'appuie sur la notion de contrôlabilité des situations dangereuses, ce qui est inapproprié dans le cadre des systèmes autonomes. Ainsi, si on applique la norme dans le contexte du VA, alors n'importe quelle situation courante sera qualifiée comme une situation à risque et se verra attribuer un niveau élevé d'ASIL. De plus, la complexité de l'environnement du VA rend le formalisme des ADD inadapté, car il apparaît impossible de considérer cet environnement de manière exhaustive. Enfin, la norme ISO 26262 ne fournit pas de pistes pour gérer les problèmes liés aux interactions avec l'environnement et à l'interprétation des données issues des capteurs, qui restent des sujets de préoccupation majeurs pour tout VA.

Pour remédier aux diverses limites évoquées, il a été envisagé d'adapter une méthode particulièrement utilisée dans l'aéronautique appelée MBSA pour traiter des phénomènes concomitants internes et externes, notamment en ce qui concerne la perception de l'environnement. Cette méthode a abouti à la conception d'un modèle comportemental qui permet de considérer la propagation des défaillances. Cette modélisation a été effectuée avec l'outil SimFia, un logiciel développé par APSYS et fondé sur le langage AltaRica. Nous présentons dans la section suivante le langage AltaRica avant d'exposer ces résultats.

2) Présentation du langage AltaRica: AltaRica [8] est un langage de modélisation comportementale dédié à la sûreté de fonctionnement. Son utilisation donne lieu à la création d'un modèle qui est un ensemble de briques (aussi appelées nœuds) dont la structure interne est décrite dans la figure 2.

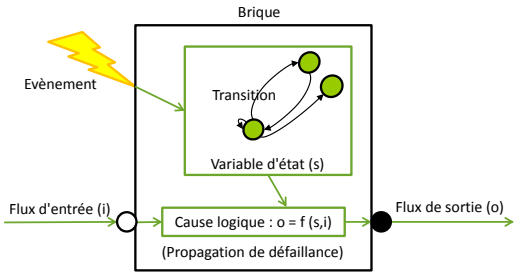

FIGURE 2: Brique AltaRica

Les briques ont des connecteurs d'entrée et de sortie par lesquels elles communiquent entre elles. La valeur d'un connecteur de sortie est déterminée par une assertion logique, fonction des connecteurs d'entrée et des variables d'état de la brique. Une variable d'état est une variable interne à la brique qui évolue grâce à des événements. La brique contient donc une machine à état, dont les états sont définis par les valeurs des variables d'état et dont les transitions sont déclenchées par les événements. Dans l'exemple de la figure 3, un événement « défaillance » fait passer la variable d'état « s » de la valeur « nominal » à la valeur « défaillant ». Par l'intermédiaire de la cause logique, le connecteur de sortie prend alors la valeur « ko », même si l'entrée de la brique est correcte.

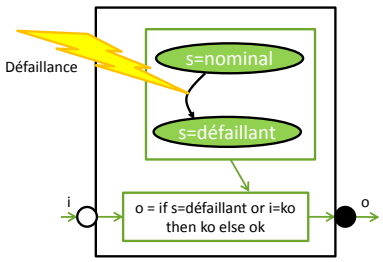

FIGURE 3: Exemple d'arrivée d'une défaillance

Enfin, le langage permet d'organiser les briques dans une hiérarchie : une brique composite contient d'autres briques (composites ou non). Étant donné une variable du modèle et une valeur, un compilateur d'AltaRica génère tous les ensembles d'événements qui conduisent à ce que la variable prenne la valeur considérée. Par exemple, il est intéressant de savoir s'il existe une défaillance de composant qui peut entraîner la défaillance du système entier. L'outil SimFia offre la possibilité de générer les coupes ou les séquences minimales d'apparition d'un événement donné (c'est-à-dire une combinaison de variables d'état). Ceci se fait grâce aux algorithmes de *model checking* implantés dans l'outil. Par exemple, pour le modèle de la figure 4, avec pour chaque nœud une défaillance simple :

— les coupes trouvées seront $\{C\}$, $\{D\}$ et $\{A, B\}$;
— les séquences trouvées seront (C), (D), (A, B) et (B, A).

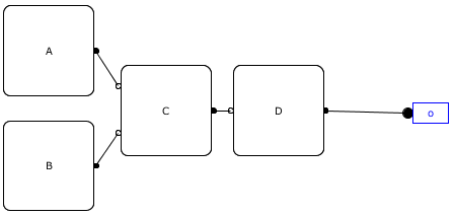

FIGURE 4: Exemple de modèle AltaRica avec quatre briques

3) Propagation des défaillances: La méthode MBSA requiert de construire un modèle du système avec un haut niveau d'abstraction et permet de générer toutes les séquences minimales menant à une situation dangereuse, ce qui permet d'avoir une approche plus riche et plus détaillée que les démarches par coupes minimales habituellement utilisées en sûreté de fonctionnement [6]. De ce fait, le résultat recherché consiste à utiliser la méthode MBSA pour générer des scénarios potentiellement critiques liés à des problèmes d'interaction ou d'interprétation de l'environnement. L'intérêt d'une approche MBSA est de se focaliser sur des classes de scénarios critiques, à travers la sélection de facteurs ayant un impact direct sur des situations dangereuses définies de manière macroscopique. Le comportement du TJC a été étudié en utilisant un modèle comportemental [6]. La modélisation comportementale est une modélisation qui a pour but de permettre l'étude d'un système à travers l'observation de ses changements d'état en fonction d'occurrence d'événements.

L'objectif principal de la modélisation comportementale était de créer un modèle global (scène, perception, commandes...) sous SimFia se rapprochant au maximum des spécifications des capteurs et des entrées-sorties du TJC. Le modèle, une fois complété, peut être exploité pour la génération des séquences d'événements menant aux situations

redoutées. Dans le but de modéliser l'ADAS (Advanced Driver-Assistance System), le VA (appelé aussi véhicule Ego dans la suite de cet article) et son environnement (les autres véhicules, les conditions météorologiques, etc.) ont été pris en compte comme illustré sur la figure 5. La structure du modèle est la suivante :

— *Environment* : tous les éléments de l'environnement intervenant dans le fonctionnement du TJC ;
— *Perception* : capteurs de l'Ego utilisés par le TJC ;
— *Fusion* : combinaison de différentes informations renvoyées par différents capteurs ;
— *Control* : décision du mouvement de l'Ego ;
— *Activation conditions of TJC* : conditions d'activation du TJC.

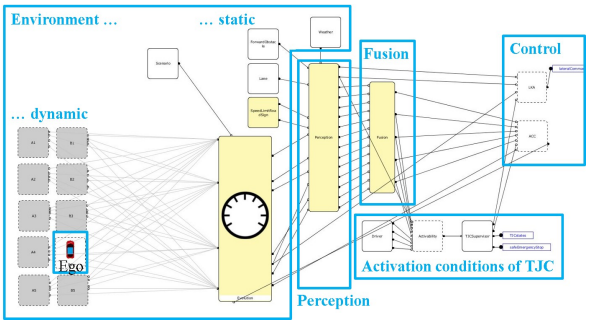

FIGURE 5: Vue générale du modèle comportemental [2]

L'*Environment* modélise les éléments de l'environnement qui sont utilisés par le TJC ou qui l'influencent, et qui se trouvent dans la limite de la portée des capteurs. L'environnement statique comprend les lignes de marquage, les conditions météorologiques, la signalisation et, possiblement, des obstacles. L'environnement dynamique est constitué de la circulation routière. Le modèle peut contenir jusqu'à deux véhicules dans l'environnement du VA. La partie *Control* modélise les modules de commandes permettant de décider du mouvement de l'Ego. Le modèle contient deux modules de commande :

— l'Adaptative Cruise Control (ACC) commande l'accélération et donc la vitesse longitudinale ;
— le Lane Keeping Assist (LKA) commande la trajectoire latérale.

La partie *Control* porte les logiques de décision. Ces logiques sont assez complexes à concevoir car elles doivent assurer la stabilité du modèle, c'est-à-dire :

— lorsqu'il n'y a pas de changement dans l'état du système, la consigne doit rester identique ;
— lorsqu'il y a un changement, la consigne doit réagir et remettre le système dans un état stable.

Par exemple, si la distance au véhicule de devant se réduit, l'ACC commande de freiner.

À travers ce modèle, il fallait illustrer la prise en compte des spécificités expliquées dans la norme SOTIF. Cela s'est concrétisé en modélisant les conditions météorologiques, qui peuvent influencer les fonctions des capteurs, en un nœud *Weather*. Une fois que la variable d'état du nœud *Weather* passe dans une condition de perturbation, des événements peuvent se déclencher avec une forte probabilité.

Cette approche qui permet de modéliser la propagation des défaillances a donné des résultats intéressants, mais des limitations sont également apparues. La première limitation concerne le pouvoir d'expressivité : avec AltaRica, on ne peut s'exprimer qu'en utilisant des variables discrètes alors que la modélisation de la dynamique de l'environnement est d'ordre continu (par exemple la variation des positions des véhicules les uns par rapport aux autres, ou bien les variations des vitesses et des accélérations). De même avec AltaRica, il n'est pas possible de créer des concepts qui sont l'intersection de plusieurs concepts. Par exemple, supposons que l'on définisse un modèle météo. Ce modèle pourra avoir les valeurs « pluie », « brouillard », « nuages ». Si l'on souhaite avoir un modèle météo qui représente à la fois la pluie et le brouillard, il faudra concrètement créer des boîtes spécifiques pour chaque état de la météo au sein de la boîte météo. Ce formalisme complexifie la modélisation.

Une seconde limitation est en rapport avec la complexité calculatoire. Au sein d'une coupe ou d'une séquence, le nombre d'événements présents est appelé *ordre*. Plus la coupe ou la séquence comporte d'événements, moins elle est probable et donc critique. Quand on augmente la complexité de l'environnement, le temps de calcul de la génération des coupes (ou des séquences) augmente exponentiellement. En consultant le tableau I, on peut constater que le calcul des coupes et séquences d'ordre 4 est déjà très long ($\simeq$ 2,5 jours) avec une station de calcul standard. De plus, on rencontre un problème d'explosion combinatoire à partir de l'ordre 5, pour le modèle de la figure 5.

TABLE I: TEMPS DE CALCUL DES COUPES ET DES SÉQUENCES EN FONCTION DE L'ORDRE [5]

	Ordre	Temps
Coupes	1	1 s
	2	34 s
	3	52 min
	4	60 h
	5	Non dét.
Séquences	1	1 s
	2	32 s
	3	50 min
	4	50 h
	5	Non dét.

Enfin, une troisième limitation est en rapport avec l'une des idées qui ont donné lieu à la proposition de ce travail. Il s'agit de la possibilité de discriminer les scénarios en identifiant leurs classes afin de pouvoir les classer. Ceci permettrait par exemple d'optimiser la couverture de test. En effet, le formalisme du langage AltaRica ne permet pas de vérifier la cohérence des concepts qui sont définis, et donc une discrimination des scénarios n'est pas possible.

C. Utilisation des ontologies

Les scénarios de validation du VA sont spécifiques à leurs systèmes. L'identification de nouvelles situations intéressantes à rejouer en simulation s'appuie aujourd'hui essentiellement sur l'expertise des ingénieurs en sûreté de fonctionnement et des ingénieurs systèmes. L'accidentologie et les roulages sont quant à eux des sources d'information et d'inspiration qui s'appuient sur des données réelles. Ces données recueillies

doivent permettre d'identifier des scénarios intéressants, et de construire une approche statistique des paramètres représentatifs autour de ces scénarios. Cependant, ces paramètres sont très nombreux, et ce constat conduit inévitablement à un grand nombre de combinaisons. Parmi ces scénarios possibles, certains sont pertinents à rejouer, c'est-à-dire les scénarios menant à des conséquences potentiellement graves, et d'autres ne le sont pas. La question est de savoir comment identifier les premiers et rejeter les autres. Or, cette approche ne permet pas de répondre complètement à ces questions, et n'offre aucune garantie quant à la couverture de l'ensemble des situations critiques.

Pour apporter des éléments de réponses à ces questions, nous avons proposé de créer des scénarios via une approche théorique [9]. En effet, il s'agit de systématiser la création de scénarios, et d'en extraire les plus pertinents parmi le champ des possibles. Cette méthodologie de génération s'appuie d'abord sur une ontologie des éléments représentatifs de l'environnement du VA. Construire une ontologie permet non seulement de définir et de structurer une base de connaissances (par exemple de hiérarchiser des éléments de la météo, de l'infrastructure, etc.), mais aussi d'introduire des règles qui permettent de faire des vérifications et du tri sur la génération des scénarios. À cette fin, un graphe de dépendance entre les éléments de l'ontologie a été créé afin d'établir des relations additionnelles notamment des relations probabilistes et mathématiques.

Cette méthode nous a permis de générer un nombre fini de scénarios différents et de minimiser encore ce nombre grâce à un raisonneur se basant sur les règles ontologiques. Cependant, une ontologie riche et détaillée sera associée à un graphe complexe et plus difficile à maîtriser. Nous avons aussi constaté qu'il est difficile de maîtriser la génération de scénarios redondants ou superflus. En effet, il faut générer un très grand nombre de scénarios pour couvrir l'ensemble des cas possibles. Ces quantités pèsent sur les ressources allouées aux moyens de simulation disponibles et nécessitent un temps d'analyse des résultats considérable.

III. Réalisations

L'approche de sécurisation par propagation des défaillances a été à l'origine de la conception d'un modèle comportemental se rapprochant au maximum des spécifications des capteurs et des entrées sorties du TJC. Le modèle comportemental peut être exploité pour la génération des séquences d'événements menant aux situations redoutées. Cette approche permet de prendre en compte les aspects « fonctionnel sûr » contrairement à l'approche par allocation de défaillance. Cependant l'application de l'approche par propagation des défaillances a révélé plusieurs limitations telles que le pouvoir d'expressivité du langage AltaRica et les problèmes de complexité inhérents ou l'impossibilité d'assurer la cohérence des concepts définis.

Nous sommes conscients qu'il serait impossible de décrire toute la complexité de l'environnement du VA dans le modèle comportemental sans être confronté aux limitations évoquées. Une possibilité pour contourner cette limitation est d'utiliser un autre formalisme pour le modèle comportemental. L'utilisation d'un système de Représentation de Connaissance (RC) formel permettrait de vérifier la cohérence des concepts définis. Le caractère formel des ontologies nous permet de constater que ce formalisme présente des spécificités qui pourraient permettre de remédier aux limitations soulevées par l'application de l'approche par propagation des défaillances. En effet, en utilisant une ontologie, il est possible de saisir formellement les concepts, en s'appuyant sur un formalisme mathématique. Cependant le formalisme des ontologies n'est pas adapté pour définir un modèle de propagation des défaillances de la méthode MBSA qui, lui, présente des avantages comme la génération des séquences d'événements menant aux situations redoutées. Il paraît donc intéressant d'utiliser les deux formalismes et de voir comment les combiner en un workflow, pour tenter de transcender le caractère aporétique de l'approche par propagation des défaillances.

Il est donc envisagé dans la suite de ce papier d'extraire des éléments conceptuels décrivant l'environnement du modèle comportemental et de les définir dans une ontologie tout en conservant le système TJC, qui s'occupe de la tâche de conduite, dans le modèle comportemental et donc défini avec le langage AltaRica. Cela permettra de simplifier le modèle comportemental afin d'éviter les problèmes de complexité lors de la génération des séquences.

En analysant le modèle comportemental (figure 5), nous pouvons constater que la modélisation de l'environnement se compose de deux parties principales, une partie statique et une partie dynamique.

> L'environnement statique comprend les lignes de marquage, les conditions météorologiques, la signalisation et, possiblement, des obstacles. L'environnement dynamique est constitué de la circulation routière. Le modèle peut contenir jusqu'à deux véhicules dans l'environnement du véhicule autonome. [6]

L'environnement statique est plus simple à modéliser. En effet, la modélisation de l'environnement dynamique a nécessité un bouclage depuis la partie *Control* vers la partie *Environment*.

> Comme le trafic est dynamique et que la situation dépend du comportement du véhicule Ego, nous avons bouclé la consigne ACC et la vitesse longitudinale du véhicule Ego. Lorsqu'un changement apparaît dans l'environnement ou l'état du système, l'ACC fournit une nouvelle consigne et un événement est déclenché pour mettre à jour la valeur de la vitesse. Ainsi nous pouvons visualiser l'effet des logiques de l'ACC et les différentes étapes du mouvement. [2]

Cela signifie donc que l'Ego peut avoir une influence sur l'environnement modélisé. La conséquence sur la définition du workflow est grande. Il faut qu'il soit possible de transmettre également des informations depuis le modèle comportemental vers l'ontologie et mettre à jour l'ontologie. Cette caractéristique est propre à la partie dynamique. La partie statique, a priori, nécessite seulement de pouvoir transmettre des informations depuis l'ontologie vers le modèle comportemental.

Nous commencerons donc d'abord par traiter de la partie statique de l'environnement. La résolution des difficultés rencontrées lors de la réalisation de cette partie du workflow

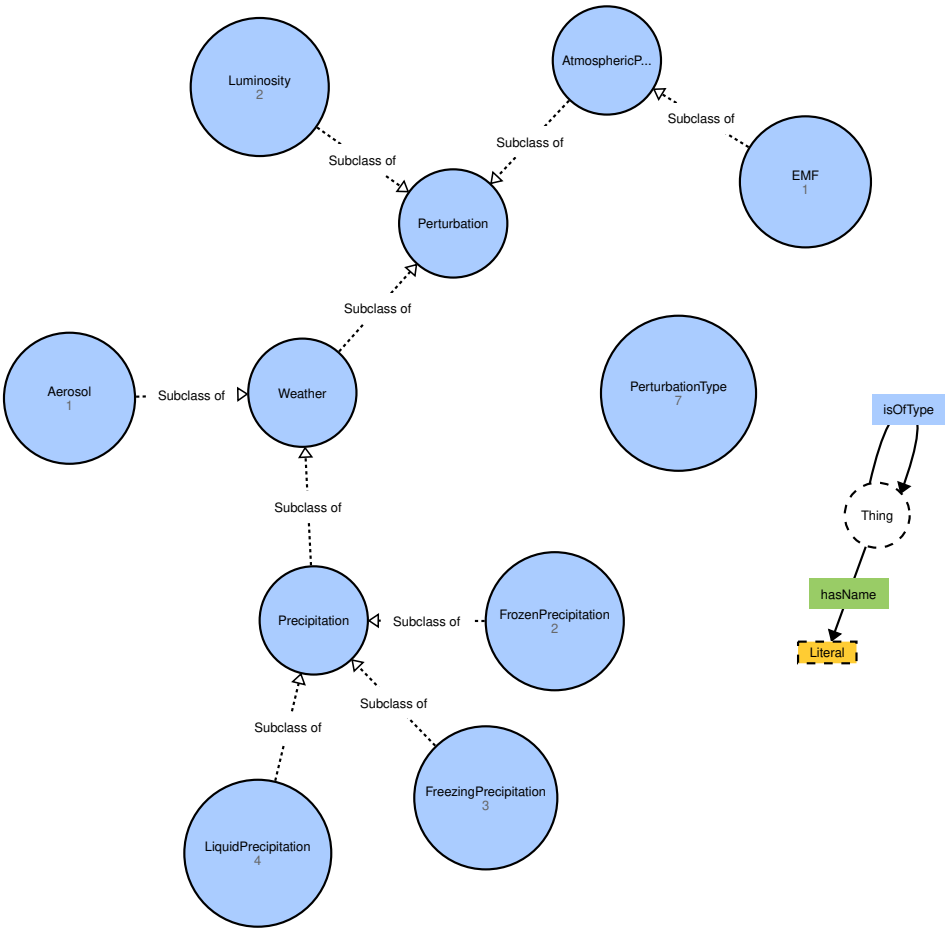

FIGURE 6: Ontologie représentant la météo du modèle comportemental

nous servira pour aborder la partie dynamique, plus complexe a priori à mettre en œuvre.

A. *Gestion de l'environnement statique*

« L'environnement statique comprend les lignes de marquage, les conditions météorologiques, la signalisation et, possiblement, des obstacles. » [6]. Parmi ces éléments statiques, la définition des conditions météorologiques nous intéresse plus particulièrement parce que, comme nous l'avons vu, la météo a été utilisée pour illustrer la prise en compte du « fonctionnel sûr » par le modèle comportemental.

Dans cette section nous utilisons l'exemple de la météo pour proposer une méthodologie qui permet de définir un workflow à même d'intégrer à notre réflexion le problème de l'explosion combinatoire inhérente au modèle comportemental et de traiter cette explosion combinatoire. Ce workflow consiste à remplacer dans le modèle défini en AltaRica la modélisation de la météo par un type énuméré qui correspond aux différentes valeurs que peut prendre la variable d'état du nœud météo. Ce type énuméré est le résultat de l'interrogation d'une ontologie qui modélise la météo et comporte les différentes valeurs du type énuméré.

Il existe différentes familles de langages d'ontologies qui ont chacune des spécificités. Nous nous sommes arrêtés sur le langage d'ontologie Web Ontology Language (OWL). Nous définissons le nœud météo dans une ontologie illustrée dans la figure 6.

Une fois que nous avons défini l'ontologie de la météo, il faut trouver un moyen de transmettre les informations de l'ontologie vers le modèle comportemental.Le workflow nécessite de définir une manière d'interroger l'ontologie pour récupérer les informations adéquates, car en fonction des choix décidés pour représenter la connaissance dans l'ontologie, les informations à transmettre au modèle comportemental ne seront pas nécessairement les mêmes. Les informations à transmettre sont déterminées par les choix de modélisation. Il faut donc bénéficier d'un moyen paramétrable pour interroger l'ontologie et récupérer les informations adéquates.

O'CONNOR et DAS ont eu l'idée de réutiliser le langage Semantic Web Rule Language (SWRL) pour définir un langage de requête, Semantic Query Web Rule Language (SQWRL) [10]. Nous décidons d'utiliser SQWRL pour interroger l'ontologie et sélectionner les informations à transmettre à l'ontologie. En utilisant SQWRL, nous bénéficions d'un moyen simple et paramétrable pour interroger l'ontologie.

Il nous reste à définir le lien proprement dit entre l'ontologie et le modèle comportemental en étudiant les éléments nécessaires à conserver dans le modèle.

Nous choisissons de nous appuyer sur une version figée du modèle comportemental qui servira de référence pour l'étude de la définition du lien entre l'ontologie et le modèle. En considérant les parties adéquates du modèle comportemental nous adaptons le modèle AltaRica pour qu'il soit capable de prendre en compte les valeurs initiales des variables d'état

qui seront transmises depuis l'ontologie. Cela nous permet de supprimer le nœud météo pour simplifier le modèle AltaRica.

Nous disposons à présent de tous les éléments nécessaires pour transmettre les informations de la partie météo définie dans l'ontologie vers le modèle comportemental. Ainsi, pour extraire les éléments de la partie statique du modèle comportemental, nous pourrons suivre le même procédé et utiliser les mêmes technologies. Il faudra suivre les étapes suivantes :

— définir la partie du modèle souhaitée dans une ontologie ;
— écrire les requêtes SQWRL nécessaires pour pouvoir récupérer les informations adéquates dans l'ontologie ;
— adapter le modèle comportemental pour traiter les valeurs transmises par l'ontologie et supprimer les nœuds qui sont maintenant définis dans l'ontologie pour simplifier le modèle.

Une fois ces tâches effectuées, nous disposons d'un workflow (figure 7)

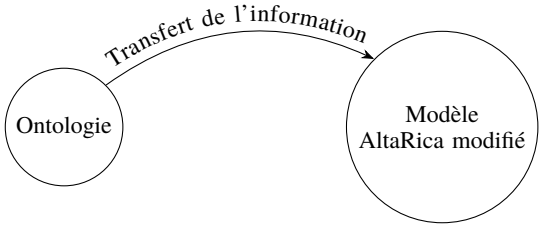

FIGURE 7: Workflow de la partie statique

qui permet la communication entre une ontologie et un modèle comportemental défini en AltaRica.

Nous devons maintenant analyser la partie dynamique de l'environnement pour évaluer les capacités de ce workflow à saisir adéquatement les potentielles particularités de cette partie dynamique, et envisager de l'enrichir si nous nous heurtons à ses limites dans notre tentative.

B. Analyse de l'environnement dynamique

« L'environnement dynamique est constitué de la circulation routière. Le modèle peut contenir jusqu'à deux véhicules dans l'environnement du véhicule autonome. » [6].

La partie dynamique est définie dans un nœud *Evolution* qui comporte l'état de l'évolution de la position des véhicules (le nœud *StateEvolution*) et l'intelligence du calcul de cette évolution (le nœud *LogicalCauseEvolution*). Cette partie contient de nombreuses dépendances internes qui permettent de stabiliser le modèle. En effet, la grille dans la partie dynamique de la figure 5 représente le placement relatif des véhicules et non un repère absolu. Les automates de chaque véhicule ont ainsi besoin des informations de vitesse et de position des autres véhicules pour se placer correctement sur la grille. Cette cohérence impose de gérer l'évolution dynamique de la scène dans un seul formalisme. Une gestion entre deux formalismes distincts imposerait de multiples aller-retours entre eux et risque de mener à un modèle non représentatif. Cette partie est ainsi extraite en intégralité du modèle pour être gérée par l'ontologie, à l'image de l'environnement statique. Il est cependant nécessaire de gérer les interactions de l'Ego avec son environnement, à savoir la commande en boucle fermée. Par exemple, en modifiant sa consigne de vitesse, l'Ego agit sur son environnement et modifie à minima les distances aux autres véhicules qu'il perçoit, variables d'entrées de la commande. La prise en compte de la partie dynamique de l'environnement nécessite d'enrichir le workflow (figure 8) pour intégrer cette boucle.

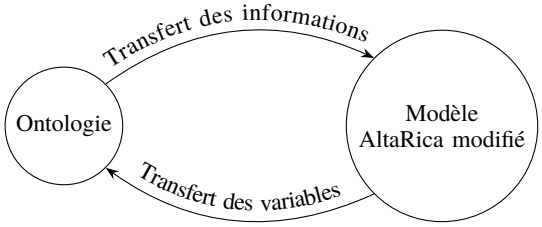

FIGURE 8: Workflow de la partie dynamique

L'exécution du modèle AltaRica permettra de générer une valeur de consigne dépendante de l'environnement perçu par l'Ego. Cette valeur de la variable de consigne est transmise à l'ontologie qui modifie alors l'état de l'environnement en calculant une nouvelle scène mettant à jour les vitesses et positions respectives de chaque véhicule. Les informations de cette nouvelle configuration sont exportées de l'ontologie vers le modèle qui pourra ainsi recalculer la nouvelle consigne à partir de cette configuration.

À travers cet exemple, nous pouvons appréhender les modifications que le workflow que nous proposons pour gérer la partie statique de l'environnement doit souffrir pour être en mesure de saisir les spécificités de la partie dynamique de l'environnement. Il s'agira de considérer la génération de séquences à chaque tirage d'événement ou groupe d'événements consécutifs s'étant déclenchés et appartenant au modèle comportemental, c'est-à-dire à la définition du TJC, pour récupérer les valeurs des variables associées aux bouclages et mettre à jour l'ontologie de l'environnement dynamique, pour enfin récupérer les valeurs dans l'ontologie et les transférer au modèle comportemental. Les groupes d'événements consécutifs appartenant au modèle comportemental correspondent à des sous-séquences qui peuvent être générées en utilisant SimFia car ils sont tous définis dans le modèle comportemental, et à chaque bouclage, il faudra revenir vers l'ontologie pour la mettre à jour avant de retourner vers le modèle comportemental.

Ce workflow permet de générer des séquences d'ordre élevé, par exemple une séquence d'ordre sept (figure 9).

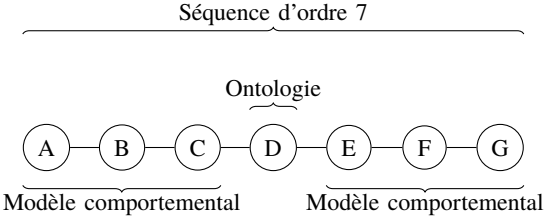

FIGURE 9: Découpage d'une séquence d'ordre 7

Cette séquence est constituée de trois événements se déclenchant dans le modèle comportemental (les événements A, B et C), suivis d'un événement qui doit être évalué dans l'ontologie de l'environnement dynamique (l'événement D), suivi lui-même par trois événements du modèle comportemental (les événements E, F et G). Pour générer cette séquence nous devons nécessairement effectuer un aller-retour entre le modèle comportemental et l'ontologie. Nous

commençons par générer la sous-séquence (A, B, C) avec SimFia. Puis nous mettons à jour l'ontologie avec les valeurs de bouclage modifiées. Cette étape correspond à la sous-séquence (D). Enfin nous transférons les informations de l'ontologie au modèle comportemental et générons la sous-séquence (E, F, G) avec SimFia.

IV. Résultats

Nous présentons dans cette section les résultats de l'application des deux parties du workflow présentées précédemment.

A. Application du workflow statique

Pour évaluer ce workflow nous allons nous appuyer sur l'exemple suivant : nous définissons deux concepts statiques du modèle AltaRica dans une ontologie. Ce sont la météo et le marquage au sol. En appliquant la méthodologie pour définir le workflow, nous obtenons l'ontologie, les requêtes SQWRL pour interroger l'ontologie et le modèle AltaRica modifié dont les nœuds qui modélisent la météo et le marquage au sol ont été supprimés.

Le temps de calcul de la génération de certaines séquences en s'appuyant sur ce workflow est amplement réduit. Dans notre exemple, cette diminution du temps de calcul est immédiatement perceptible dès l'ordre 4. Le temps de calcul des séquences d'ordre 4 commençant par un événement correspondant à une modification de la météo ou du marquage au sol est équivalent à un temps de calcul des séquences d'ordre 2 du modèle original, c'est-à-dire quelques dizaines de secondes. De même le temps de calcul des séquences d'ordre 5 commençant par un événement correspondant à une modification de la météo ou du marquage au sol est de 14 h, c'est-à-dire un temps de calcul qui s'inscrit entre l'ordre 3 et 4 du modèle original. Pour obtenir le temps de calcul de l'ensemble des séquences d'ordre 4, il faut ajouter le temps de calcul des autres séquences qui commencent par des éléments de l'environnement qui ne sont pas encore définis dans l'ontologie, c'est-à-dire les éléments statiques autres que la météo et le marquage au sol. Nous pouvons cependant extrapoler les calculs précédents. En effet, le calcul de ces séquences se rapportera aux calculs précédents à chaque fois que nous ajouterons un élément statique de l'environnement à l'ontologie et que nous supprimerons le nœud correspondant dans le modèle AltaRica. Il est aussi nécessaire d'ajouter le temps de calcul des séquences qui ne font pas intervenir l'environnement, c'est-à-dire les séquences dont les événements correspondent entièrement à des éléments du TJC, ce qui représente 1 h environ.

Table II: Comparaison des temps de calcul des séquences entre le modèle original et le modèle modifié

	Ordre	Temps	
		Modèle original (n nœuds)	Modèle modifié ($n-2$ nœuds)
Séquences	1	1 s	
	2	32 s	
	3	50 min	
	4	50 h	1 h
	5	Non dét.	$14\,\text{h} + \alpha$

Enfin, il faut ajouter d'une part le temps de calcul des séquences qui contiennent un événement relatif à la partie statique qui ne se situe pas au début des séquences, et d'autre part le temps calcul des séquences relatives aux éléments dynamiques. Le temps de calcul total est consultable dans le tableau II. Le calcul des séquences d'ordre 4 est d'1 h, car, en s'appuyant sur le workflow, les temps de calcul à ajouter que nous venons d'évoquer sont négligeables. Par contre à partir de l'ordre 5, il devient nécessaire de prendre en compte le temps de calcul de ces catégories de séquences, que nous appellerons α. Le temps de calcul total des séquences d'ordre 5 est donc de $14\,\text{h} + \alpha$. La problématique du calcul de α posée par cette catégorie de séquences nécessite l'utilisation du workflow enrichi, dont nous évoquons l'application dans la section suivante.

B. Application du workflow dynamique

La problématique du bouclage constitue en définitive l'unique spécificité de l'environnement dynamique par laquelle il se différencie de l'environnement statique. La mise en application de ce bouclage consiste donc à récupérer d'une part la valeur de consigne et d'autre part à réinjecter les nouveaux paramètres de la scène dans le modèle comportemental.

Actuellement il est impossible de récupérer la valeur de consigne à chaque pas de la génération des séquences ou coupes avec le logiciel SimFia. Celui-ci n'est pas conçu pour permettre l'interruption de l'algorithme de génération de séquence. Mais, pour les besoins de la réalisation du démonstrateur, nous proposons une manière de contourner cette limitation.

Connaissant les valeurs possibles que peut prendre la consigne, nous en choisissons une, augmenter la vitesse par exemple. Nous pouvons alors générer tous les séquences minimales aboutissant à cette valeur de consigne à partir de la configuration initiale. En parallèle, nous pouvons calculer à partir de l'ontologie une nouvelle configuration initiale à appliquer au modèle comportemental. Enfin, nous pouvons générer toutes les séquences qui mènent à l'événement redouté à partir de cette seconde configuration initiale.

En outre, comme il n'est pas possible d'interrompre la génération de séquence, ni de forcer de l'extérieur la mise à jour des valeurs de l'environnement, il est nécessaire de sauvegarder l'état global du modèle comportemental à la fin de la première séquence, afin de pouvoir reprendre la génération de séquence où elle s'était arrêtée. L'ensemble permet de reconstituer alors des séquences complètes aboutissant à une situation d'accident et tenant compte de l'évolution de l'environnement.

L'exemple utilisé s'articule donc en trois temps. Dans un premier temps, nous générons toutes les séquences minimales d'ordre 3 conduisant à une consigne d'augmentation de la vitesse suite à un défaut du TJC. Dans un deuxième temps, nous calculons l'évolution de la scène résultant de l'accélération du véhicule Ego. Dans un troisième temps, nous poursuivons les séquences en générant les séquences minimales jusqu'à l'ordre 3, menant à l'événement redouté à partir de la nouvelle scène et de la configuration intermédiaire du TJC. Cette troisième étape ne s'est effectuée qu'à partir de quelques

sous-séquences représentatives permettant d'extrapoler les résultats.

L'observation de l'événement redouté nécessite d'être modifiée par rapport au modèle comportemental unique. Celui-ci définissait la collision par la présence de deux véhicules sur une même case de la grille de la figure 5. En l'absence d'évolution dynamique, il devient nécessaire de définir celui-ci comme une erreur de consigne menant à une situation dangereuse, par exemple ne pas ordonner de diminution de la vitesse lorsque le véhicule cible est devenu trop proche et ainsi continuer à se rapprocher.

L'exécution de l'exemple permet de reconstruire la majorité des séquences obtenues à l'ordre 4 avec le modèle comportemental unique. Une classe de séquences fondées sur une surestimation de la vitesse du véhicule cible disparaît cependant. Celle-ci se justifie par les modalités d'observation de l'événement redouté. Dans le modèle unique, le véhicule ne ralentit pas suffisamment pour éviter la collision, menant à l'événement redouté. Ceci ne constitue pas une mauvaise décision et n'est pas perçu comme un événement redouté dans cet exemple. Une quatrième étape avec un second passage par l'ontologie permettrait cependant d'atteindre la collision.

L'exemple n'a pas mis en évidence de séquence d'ordre supérieur à 4 compte tenu de l'architecture simple du modèle. Il faut néanmoins considérer que l'exemple a permis d'explorer des séquences allant jusqu'à l'ordre 7. Ces séquences n'avaient pas pu être explorées dans le modèle comportemental unique.

Avec la simplification du modèle, le temps de génération de séquences d'ordre 3 est d'environ 5 min à 10 min. La reconstitution de l'ensemble des séquences nécessite dans cet exemple 622 générations de séquence d'ordre 3. La première génération aboutit à 69 séquences, tandis que l'évolution depuis l'ontologie peut mener à 9 configurations. Ceci représente environ 1 h à 2 h de calcul, un temps bien inférieur aux 50 h qui avaient nécessaires pour atteindre l'ordre 4 dans le modèle comportemental unique.

La définition de l'environnement dans l'ontologie et sa suppression du modèle comportemental permet de se restreindre à plusieurs générations de séquence à un ordre plus petit pour arriver aux mêmes résultats qu'une seule génération à un ordre plus élevé (en termes de satisfaction de risque résiduel acceptable).

L'analyse actuelle ne tient cependant pas compte des limitations techniques empêchant le fonctionnement du workflow d'un bout à l'autre de son exécution sans interruption, même s'il paraît manifeste que nous puissions extrapoler depuis cet exemple pour la généraliser.

Ce découpage offre la possibilité de générer des séquences d'ordre très élevé, autorisant à évacuer dans certains cas la problématique de la complexité.

V. Conclusion

L'approche proposée dans cet article s'appuie sur l'utilisation combinée d'une ontologie pour décrire l'environnement du véhicule autonome et d'un modèle comportemental pour décrire sa logique de fonctionnement et les défaillances pouvant l'affecter. Cette approche s'est montrée capable d'unir les avantages des deux formalismes tout en résolvant une partie des inconvénients. Ainsi, la génération de scénarios fondés sur un enchaînement de scènes tirées d'une ontologie peut être orientée par un point de vue sûreté de fonctionnement. L'utilisation de la génération de séquences permet de construire un enchaînement de scènes cohérentes vis-à-vis de défaillances pouvant survenir et mener à un accident. Cette cohérence était partiellement assurée par le modèle comportemental qui intégrait le changement de scène dans les séquences générées. Cependant, ceci exigeait une exploration exhaustive de l'espace d'état composé de toutes les combinaisons possibles d'états du véhicule autonome et de son environnement, menant à une explosion combinatoire pour les ordres élevés.

La séparation des deux parties du modèle et la mise en place d'un workflow pour les faire communiquer a permis de réduire cette explosion combinatoire sans réduire l'exhaustivité des résultats obtenus. Les temps de génération sont ainsi bien inférieurs pour des résultats équivalents. Ces points validés sur le principe nécessitent cependant d'être vérifiés avec un outillage plus abouti et sur un jeu de test plus grand que quelques exemples.

Cela permet également d'utiliser les points forts de chaque formalisme. Ainsi, l'utilisation d'une ontologie permet de trier les séquences générées en classes et ainsi de pouvoir déterminer si plusieurs scénarios couvrent ou non des situations différentes. L'objectif est ainsi d'arriver à une génération optimisée ne retenant que les scénarios les plus pertinents couvrant un maximum de situations différentes.

La couverture devient un critère important à mettre en place pour cette génération de scénarios. En effet, en réduisant la taille des modèles et en permettant de générer des scénarios complexes qui sont en fait l'enchaînement de multiples scénarios plus petits, le workflow démultiplie les possibilités. Il y a alors un risque de reporter l'explosion combinatoire sur la quantité de scénarios générés en augmentant leur longueur. Se pose alors la question de la pertinence de rechercher des scénarios d'ordre très élevé. La mise en place d'une heuristique est ainsi nécessaire pour piloter la génération de scénarios et les allers-retours entre les deux formalismes afin de maximiser la couverture des situations rencontrées sans négliger l'identification des scénarios à enjeux sécuritaires. Cette heuristique permettra ainsi d'éviter une exploration de l'infinité des cas d'usage possibles.

Acronymes

ACC Adaptative Cruise Control

ADAS Advanced Driver-Assistance System

ADD Arbre De Défaillance

ASIL Automotive Safety Integrity Level

IRT Institut de Recherche Technologique

LKA Lane Keeping Assist

MBSA Model Based Safety Assessment

OWL Web Ontology Language

RC Représentation de Connaissance

SOTIF Safety Of The Intended Functionality

SQWRL Semantic Query Web Rule Language

SVA Simulation pour la sécurité du Véhicule Autonome

SWRL Semantic Web Rule Language

TJC Traffic Jam Chauffeur

VA Véhicule Autonome

Références

[1] L. Zhao, E. Arbaretier et M. TLIG, « Virtualization and simulation validations : new methodological and technical fields for safe design engineering of autonomous systems, » anglais, in *Congrès Lambda Mu 21 "Maîtrise des risques et transformation numérique : opportunités et menaces"*, Reims, France, oct. 2018.

[2] M. Tlig, M. Machin, R. Kerneis, E. Arbaretier, L. Zhao, F. Meurville et J. Van Frank, « Autonomous driving system : model based safety analysis, » anglais, in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, L. O'Conner, éd., IEEE, Luxembourg City, Luxembourg : IEEE, 25-28 juin 2018, p. 2-5.

[3] Technical Committee ISO/TC 22 et Subcommittee SC 32, « Road vehicles — safety of the intended functionality, » anglais, International Organization for Standardization, Chemin de Blandonnet 8 1214 Vernier, Geneva, Switzerland, rapp. tech. ISO/PAS 21448:2019, jan. 2019.

[4] ——, « Road vehicles – functional safety, » anglais, International Organization for Standardization, Chemin de Blandonnet 8 1214 Vernier, Geneva, Switzerland, rapp. tech. ISO 26262:2018, déc. 2018.

[5] R. Kerneis, « Modélisation comportementale pour la sécurisation du véhicule autonome, » mém. de mast., Institut National des Sciences Appliquées (INSA) - Rouen, 2017, 44 p.

[6] M. Tlig, M. Machin, R. Kerneis, E. Arbaretier, L. Zhao, F. Meurville et J. Van Frank, « Contribution à la sécurisation du véhicule autonome : Modélisation comportementale avec AltaRica, » in *Congrès Lambda Mu 21 "Maîtrise des risques et transformation numérique : opportunités et menaces"*, L. Marle, éd., Reims, France, 4 oct. 2018.

[7] D. Loche, « Architecture E/E et Logicielle pour les Systèmes Temps-Réels à Criticité Multiple : Étude Bibliographique, » working paper or preprint, avr. 2018.

[8] A. Arnold, G. Point, A. Griffault et A. Rauzy, « The AltaRica formalism for describing concurrent systems, » *Fundamenta Informaticae*, t. 40, n° 2,3, p. 109-124, 1999.

[9] W. Chen et L. Kloul, « An ontology-based approach to generate the advanced driver assistance use cases of highway traffic, » anglais, in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2018, Volume 2 : KEOD, Seville, Spain, September 18-20, 2018*, D. Aveiro, J. L. G. Dietz et J. Filipe, éd., SciTePress, 2018, p. 73-81.

[10] M. O'Connor et A. Das, « SQWRL : a query language for owl, » anglais, in *Proceedings of the 6th International Conference on OWL : Experiences and Directions - Volume 529*, R. Hoekstra et P. F. Patel-Schneider, éd., sér. OWLED'09, Aachen, Germany, Germany : CEUR-WS.org, 2009, p. 208-215.