SÉCURITÉ
ACCEPTABILITÉ
MOBILITÉ AUTONOME

LIVRABLE L2.4-1

Etat des lieux sur les méthodologies de simulation numérique émanant des travaux réglementaires et normatifs

Version : 0.2

Date de version : 05-06-2020

Opération réalisée avec le concours des Investissements d'avenir de l'Etat confiés à l'ADEME

Informations du document

Périmètre de diffusion : Consortium / Public / Administration

Type : Intermédiaire / Final

Date prévue de livraison : T0+12

Statut : En cours / **Pour révision pairs** / Pour révision Bureau / Validé

Auteurs :

Resp. du livrable	Organisation	Rôle dans le projet
Hadi ZAATITI	IRT-SystemX	Ingénieur de recherche
Relecteurs	**Organisation**	

Table de révision :

Version	Date	Contenu de la modification
0.1	20/05/2020	Draft
0.2	03/06/2020	Version achevée initiale
0.2	05/06/2020	Relecture et correction

Table des matières

Table des illustrations et tableaux

Résumé exécutif

Contexte du projet SAM (Sécurité et l'Acceptabilité de la conduite et de la Mobilité autonome):

Afin de soutenir des projets d'Expérimentation du Véhicule Routier Autonome (EVRA), l'ADEME a lancé l'Appel à Projets EVRA dans le cadre du Programme d'investissements d'avenir (PIA), dont l'objectif est de concourir aux développements de méthodologies de validation de la sécurité, à l'amélioration des connaissances sur les usages, l'acceptabilité et les impacts sociétaux.

En réponse à cet appel à projets et en cohérence avec le programme national FVA, un consortium d'acteurs industriels (constructeurs, opérateurs de transport, systémiers et équipementiers, gestionnaires d'infrastructures), d'acteurs de la recherche et de partenaires territoriaux se rassemble pour réaliser un projet d'envergure sur la Sécurité et l'Acceptabilité de la conduite et de la Mobilité autonome (SAM), d'un budget consolidé de 114 M€.

Ce projet vise à élaborer un « bien commun », défini par l'ensemble des connaissances dont la mutualisation et le partage avec les autorités publiques bénéficient à l'élaboration des politiques publiques et à la construction d'un état de l'art, notamment en matière de sécurité, d'impacts et d'acceptabilité.

Ce bien commun sera construit autour d'une approche méthodologique commune et partagée, dans les trois domaines d'application ciblés : véhicule particulier autonome, système autonome de transport collectif et partagé, système autonome de transport de marchandises.

Sous le pilotage de la PFA, le projet SAM associe 11 industriels (Alstom, Cofiroute, EasyMile, Keolis, PSA, RATP, Renault, SNCF, Transdev, TwinswHeel, Valeo) pour réaliser des expérimentations et 8 partenaires pour en assurer la méthodologie et les évaluations dans leur domaine d'excellence (Cerema, IFP Energies Nouvelles, IGN, Le LAB, ENPC/LVMT, SystemX, UTAC CERAM, VEDECOM).

Le « bien commun » sera construit à partir d'expérimentations de véhicules autonomes sur routes ouvertes. Plus de 500 000 utilisateurs ou usagers testeront une centaine de véhicules autonomes sur 13 territoires d'expérimentation sélectionnés en France, sur une durée de 6 à 30 mois.

A propos du livrable

Le livrable ci-joint s'inscrit comme un des éléments de la tâche 2.4 du projet SAM : « Définir, outiller et tester une méthodologie de simulation numérique pour contribuer à la démonstration de la sécurité ».

La tâche consiste à définir une méthodologie de simulation numérique à base de scénario pour contribuer à la démonstration de la sécurité pour ensuite tester cette méthodologie.

Le présent livrable cite et décrit les éléments de l'état de l'art concerné par la simulation numérique émanant de travaux réglementaires et normatifs.

Introduction

Les véhicules autonomes sont une technologie émergente du futur. Les méthodes actuelles de validation et de vérification ne sont pas suffisantes pour la mise en opération de cette technologie pour plusieurs raisons :

- Le grand nombre de situations différentes qu'un véhicule autonome doit gérer ainsi que les contraintes temps réelles imposées pour prendre et effectuer les décisions.
- La conception classique à base de modèles avec des règles statiques est insuffisante pour décrire le comportement des véhicules ayant un niveau d'automatisation élevé. Ainsi, les méthodes d'intelligence artificielle à base de données sont nécessaires sauf qu'elles entrainent des difficultés pour la vérification et la validation.
- Les objectifs de sécurité nécessaires à atteindre par de telle technologie pour qu'elle soit mise en place est souvent difficilement atteignable par des tests et essais physiques. Par exemple, il est estimé qu'un très grand nombre de kilomètres de tests de roulage sont nécessaires pour valider une telle technologie. Ceci pouvant réellement prendre des années pour s'effectuer ainsi qu'un très grand coût.

Comme rappelé par le ‘’Position paper’’ (M. Brini, 2020), plus de 90% des causes critiques des accidents résultent du conducteur et peu de cas avec comme cause le véhicule lui-même. Les normes et réglementations parviennent aujourd'hui à cadrer efficacement les technologies classiques. Néanmoins, les véhicules autonomes devront remplacer les fonctions complexes du conducteur tel que la reconnaissance et la perception qui, quant à eux, sont responsables de 85% des causes critiques des accidents dues aux conducteurs (signalé par le même papier). Les normes et réglementation d'aujourd'hui doivent donc évoluer pour accompagner l'intégration de ces fonctions complexes dans les technologies existantes.

La simulation parait comme un moyen permettant de vérifier le comportement des véhicules autonomes au niveau modèle.

Dans le présent livrable, nous rappelons des éléments de la littérature concernée par la simulation, les normes et réglementation qui cadrent les moyens de simulation et leurs méthodologies sous-jacentes, ainsi que les principaux outils de simulation couramment utilisé dans l'industrie.

1. Objectif du livrable :

L'objectif du présent document est de présenter les méthodes de simulation actuelles et les travaux réglementaires et normatifs faisant intervenir de telles techniques. Nous présentons :

- Des notions générales introductives sur la simulation à base de modèle.
- Les cycles de développement des systèmes complexes et les modèles de conceptions issues de l'architecture système.
- Les normes et réglementation faisant intervenir la simulation numérique et les tests à base de scénarios.
- Une liste d'outils de simulation et leurs descriptions.
- Finalement, une partie discussion dressant les limitations des techniques actuelles ainsi que les attentes futures pour lesquelles on pense que la simulation numérique pourra y répondre notamment dans le domaine du transport intelligent.

2. Généralités sur la modélisation, les langages de modélisation et techniques de simulation:

Cette partie présente des informations d'ordre général provenant de l'état de l'art concernant la simulation numérique à base de modèle. Elle a pour but de familiariser le lecteur avec les définitions de modèle, de langage de modélisation, des principales techniques de simulation numérique et les traces d'exécution d'un modèle. Les concepts introduits sont illustrés par des exemples au fur et à mesure de leur présentation pour faciliter leur compréhension. Les lecteurs qui sont familiers par le domaine de la simulation peuvent passer directement au chapitre suivant.

2.1. Qu'est-ce qu'un modèle:

Un modèle est une **représentation abstraite** d'un système réel (ou d'un autre modèle) qui est exprimé par un **langage de modélisation**. Un tel langage peut avoir une **sémantique** très précise comme une équation mathématique ou une sémantique plus ambiguë telle que les langages naturels. Un modèle est construit pour décrire un aspect précis de l'objet modélisé, par exemple le portrait d'une personne est un modèle qui sert à illustrer les traits facials de la personne mais ne peut pas être utilisé pour représenter les traits de personnalité de la personne.

Afin de mieux illustrer les concepts introduits par cette partie, nous allons adopter **un exemple d'un modèle de feu de circulation**. L'exemple est modélisé d'une manière simple au départ et sera progressivement enrichi.

Exemple Feu de circulation :

Considérons le modèle M simplifié d'un feu de circulation. Un modèle peut se voir comme un ensemble de **variables** dont les **valeurs changent** sous certaines **conditions**. Dans un premier temps, nous proposons de modéliser le système de feu de circulation par une unique variable Feu. Celle-ci, à un instant de temps peut prendre l'une des trois valeurs permises $Rouge|Orange|Vert$. Le temps est une autre variable noté t qu'on supposera à ce stade comme ayant des valeurs entières positives $\{0,1,2,3,\dots\}$. Le modèle M comprend donc la variable Feu et temps t avec une contrainte indiquant l'espace des valeurs prises par la variable.

Figure 1 Modèle simplifié d'un feu de circulation

2.2. Qu'est-ce que la simulation numérique :

''La simulation numérique est un **calcul** qui est exécuté sur une **machine** et qui, à partir d'un modèle donnée et l'ensemble des entrées (ou paramètres) nécessaires au modèle, permet de fournir une représentation du comportement **permis** par le modèle.'' (Journal Nature)

On appellera cette représentation par la suite une (ou plusieurs) **trace d'exécution** du modèle. La simulation numérique intervient souvent dans les systèmes complexes pour lesquelles il est **difficile d'avoir des formes analytiques** connues permettant de décrire leur comportement. Les techniques de simulations numériques permettent donc **d'approximer le comportement** de tels systèmes complexes. Par conséquent, la simulation numérique n'est pas nécessaire pour décrire le comportement d'un modèle d'un système simple, dont le comportement peut être suffisamment décrit par un langage ayant une faible expressivité.

Il est donc important de constater que la simulation numérique peut entrainer des **erreurs** vu que ces méthodes approximent le comportement décrit par le modèle lui-même. Ces erreurs ne sont pas à confondre avec les erreurs dues à la **fidélité du modèle** simulé où naturellement ; il existe toujours un écart entre le modèle et la réalité entrainant des différences entre le comportement réel et celui simulé.

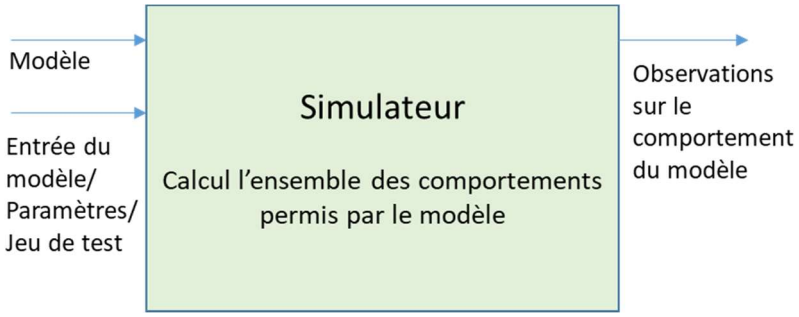

Figure 2 Schéma bloc général du fonctionnement d'un simulateur quelconque

Exemple Simulation du Feu de circulation :

Nous pouvons dès à présent simuler le modèle simple du feu avec l'unique variable Feu. Une simulation de ce modèle fourni une trace d'exécution, qui formellement peut s'exprimer le plus souvent par une séquence :

$$q_0 - t_1 \rightarrow q_1 - t_2 \rightarrow q_2 \ldots - t_n \rightarrow q_n$$

Où q_k est un vecteur contenant les valeurs de toutes les variables du modèle M au temps t_k. Notons par $t_{horizon} = \sum t_i$ le temps d'horizon aussi appelé temps de simulation indiquant la durée totale de simulation. La trace d'exécution indique les valeurs des variables du modèle pour chaque valeur de temps t en commençant par la valeur initiale (qu'on peut prendre nulle t_0 = 0) et jusqu'à un temps d'horizon (ou **temps de fin de simulation** $t_{horizon}$). Dans un premier temps, nous considérons les valeurs du temps comme discrètes $t = 0|1|2|...$ Notons qu'à ce stade nous n'avons pas encore modélisé comment les valeurs prises par la variable Feu changent avec le temps. Par conséquent, si on simule le modèle tel qu'on là actuellement, la variable Feu gardera la valeur initiale affectée à t_0 pour tous les temps successives. Nous verrons par la suite comment ajouter des contraintes supplémentaires dans M qui indiquent comment la variable Feu doit changer de valeur en fonction du temps.

		Etat initial					
Trace 1	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Rouge	Rouge	Rouge	Rouge	Rouge	Rouge
Trace 2	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Orange	Orange	Orange	Orange	Orange	Orange
Trace 3	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Vert	Vert	Vert	Vert	Vert	Vert

Figure 3 Traces d'exécutions fournies par la simulation du modèle de feu de circulation

2.2.1. Simulation par machine à état :

Corréler les changements de comportement au changement de temps : La table précédente montre les trois traces de simulations pour les trois valeurs initiales possibles de la variable Feu. Ces traces sont calculées par le simulateur et représentent les trois comportements possibles permis par M. Cependant, nous souhaitons ajouter à M des contraintes supplémentaires pour rendre le modèle **plus proche de la réalité** et donc d'avoir des traces de **simulations plus fidèles**. Par rapport à notre exemple, nous souhaitons que la variable Feu ne passe pas du $Vert$ au $Rouge$ sans avoir passé par $Orange$. Utilisons le langage de modélisation des machines à états pour enrichir M et modéliser un tel comportement:

Une machine à état ME est un couple $ME = (Q, T)$ où Q désigne l'ensemble des états et $T \subset Q \times Q$ l'ensemble des transitions.

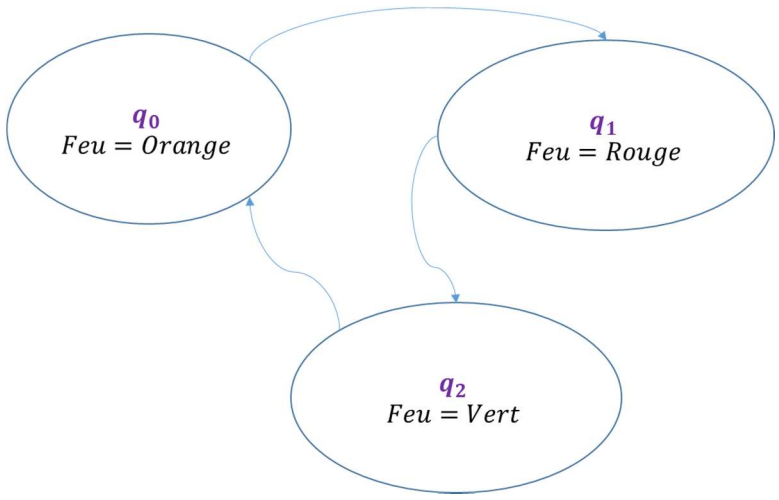

Figure 4 Machine à état simplifiée faisant partie du modèle de feu de circulation

Pour notre exemple, nous proposons la machine à état illustré dans la Figure 4 avec $Q = (q_1, q_2, q_3)$ et $T = (q_0, q_1), (q_0, q_2), (q_1, q_0), (q_2, q_1)$. La machine à état impose des contraintes supplémentaires pour restreindre le comportement du modèle, si le Feu est $Orange$, il peut soit resté $Orange$ sois devenir $Rouge$.

Exemple Simulation d'un modèle en machine à état : Simulons la machine à état en désignant par q_0 l'état initial:

		Etat initial q_0					
Trace 1	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Orange	Rouge	Rouge	Rouge	Vert	Vert
Trace 2	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Orange	Orange	Orange	Orange	Orange	Orange
Trace 3	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Orange	Orange	Rouge	Rouge	Rouge	Rouge

Nous avons générer trois traces de simulation à partir de M, notons que ces trois comportements ne sont pas les seuls possibles et une infinité de traces peut être généré par ce modèle (parce que $t_{horizon}$ peut être pris arbitrairement). Cependant, aucune de ces traces peuvent avoir un passage de la variable Feu de la valeur $Vert$ à la valeur $Rouge$ sans passer par l'état $Orange$ et vice versa. Nous avons donc réussi à corréler les changements des valeurs des variables du modèle (ici une seule variable) avec le changement de temps t.

Remarque : Notons qu'à ce stade, il est facile de prouver que toutes les traces générés par le simulateur vérifie que le Feu ne passe pas du $Vert$ au $Rouge$ sans passer par $Orange$. Ainsi des méthodes formelles de vérification de modèle peuvent automatiquement prouver qu'une telle propriété est vérifié et ceci pour toutes les traces de simulation (qui sont infinies). L'intérêt de la simulation ne se manifeste donc pas pour les modèles de haut niveau dont le comportement peut être quantifié d'une manière exacte par une machine. Cependant nous allons voir, que dès lors que le modèle devient plus complexe, et que la modélisation requiert l'emploi de méthodes d'approximations pour pouvoir générer les traces d'exécution, il ne sera plus possible de prouver formellement même les propriétés les plus simples. Il sera donc nécessaire de vérifier le modèle en générant uniquement **un ensemble finies** de traces d'exécution avec **un jeux de données 'bien choisies'** permettant de couvrir au mieux les comportements permis par le modèle. La vérification de la propriété souhaitée se fera sur cet ensemble de traces uniquement et pas sur toutes les traces possibles.

Des **contraintes temporelles** plus précises peuvent enrichir M pour plus de fidélité. Par exemple, nous pouvons modéliser le nombre d'unité de temps que la valeur du Feu reste sur $Rouge$. Supposons que la variable Feu doit rester au $Rouge$ exactement 2 unités de temps, 3 au $Vert$ et 1 à l'$Orange$. Afin de modéliser un tel comportement, nous aurons besoin d'une autre variable pour mesurer la durée pendant laquelle la variable Feu reste à une certaine valeur. Notons cette variable par d pour durée. On a supposé le temps comme des valeurs discrètes $0,1,\ldots,n$, considérons dans la suite le temps comme continu, sa dérivée par rapport à lui-même vaut 1 ou en d'autre terme, que la variable t vaut 1 si 1 unité de temps s'est écoulé, vaut 2 si 2 unité se sont écoulés etc...

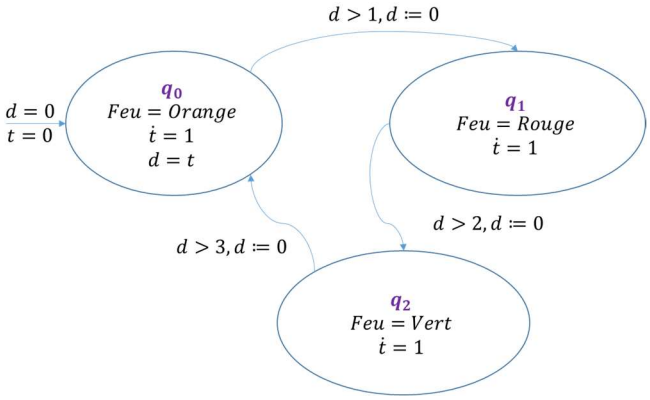

Figure 5 Machine à état temporisé, ajout de contraintes sur les durées de séjour et de changement d'état du feu de circulation

Exemple Simulation d'un modèle en machine à état temporisé : Simulons la machine à état en désignant par q_0 l'état initial du système donné par :

$$q_0 = (Feu = Orange, d = 0, t = 0)$$

Pour modéliser le fait que la variable Feu est soumise à des contraintes de durée quant à son changement de valeur, nous ajoutons des conditions dans M permettant d'exprimer ceci. Le passage de $Orange$ à $Rouge$ est conditionné par : $d > 1$ avec d mesurant le temps que la variable Feu est restée $Orange$ depuis son dernier changement. Le modèle précise aussi que la variable d est remise à 0 dès que la variable Feu change de valeur. Simulons la machine à état en variant la valeur initiale de la variable Feu et en respectant les nouvelles contraintes temporelles du modèle enrichie.

		Etat initial q_0					
Trace 1	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Orange	Rouge	Rouge	Vert	Vert	Vert
Trace 2	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Vert	Vert	Vert	Orange	Rouge	Rouge
Trace 3	Temps $t =$	0	1	2	3	4	5
	Observations $Feu =$	Rouge	Rouge	Vert	Vert	Vert	Orange

Voici trois traces de simulation. Notons que ces traces ne sont pas complètes, nous avons supposé le temps comme continu, alors qu'on montre uniquement les valeurs des variables à des instants précis seulement. Nous étions donc obliger de discrétiser par des pas de temps choisi pour pouvoir effectuer la simulation. Le pas choisi dans ces traces est de 1 unité de temps. Le pas de temps peu être élargie ou rétrécie. Le choix du pas de temps n'est pas toujours facile, s'il est trop grand, on ne voit plus le comportement souhaité (prenez un pas de 3 unités de temps,

nous verrons plus les changements vers le $Rouge$ dans la première trace par exemple) et s'il est très petit nous avons des traces très longue et précises mais qui ont demandé un temps de calcul et d'espace mémoire considérable sans avoir d'utilité supplémentaire qu'une trace avec un pas de temps plus grand.

2.2.2. Simulation numérique avec pas de temps variable :

La simulation numérique dérive les états futurs atteignables par le modèle étant donné l'état initiale, le modèle et l'ensemble des paramètres nécessaires. Les états futurs sont donnés par un ensemble de traces comme celles qu'on a vu dans les tableaux précédents. Nous avons pris un exemple simple ou la valeur Feu est discrète. Cependant pour des modèles plus compliqués les valeurs prises par les variables sont supposés continues.

Exemple Simulation d'un régulateur de vitesse d'un véhicule autonome : Prenons l'exemple de la régulation de vitesse d'un véhicule autonome A par rapport à un véhicule cible devant B. La loi de régulation est donnée par le système d'équations différentielles ordinaires suivant :

$$\dot{v_f} = 1$$
$$\dot{v} = a$$
$$\dot{gap} = v_f - v$$
$$\dot{a} = -3a - 3(v - v_f) + gap - (v + 10)$$

Où les variables v_f, v, gap, a désignent respectivement, la vitesse du véhicule B, la vitesse du véhicule A, l'inter-distance entre A et B, l'accélération du véhicule A. Le système d'équations différentielles dicte comment les variables du modèle changent en fonction du temps mais aussi en fonction d'eux même et ceci pour toute valeur du temps continu t et pour un état initial choisi.

Un simulateur numérique prend en entrée le système d'équations différentielles ainsi qu'un état initial et un ensemble de paramètre comme le temps d'horizon et le pas de temps et fourni en sortie les traces d'exécution partant de l'état initial, passant par tous les états atteignables jusqu'au temps d'horizon donné.

Exemple Traces de simulation d'un véhicule autonome avec pas de temps variable : Considérons l'état initial donnée par: $(v_f = 40, v = 60, gap = 10, a = 0)$, nous illustrons les traces de sorties du simulateur numérique qui donne une vision sur le comportement du modèle en question. Nous prenons $t_{horizon} = 50$. Remarquons comment v diminue dès les premiers instants permettant d'éviter la collision de A avec B. Le comportement simulé correspond bien à nos attentes d'un régulateur de vitesse. Mais est-ce que ceci reste vrai quand on change l'état initial ? Est-ce que ceci est vrai pour toutes les valeurs de: v, v_f, gap et a ? Cette question est difficile à répondre, nous ne pouvons pas prouver ceci d'une manière exacte. Cependant nous pouvons y répondre par une réponse approximé, en faisant plusieurs jeux de simulation et montrant que le comportement observable est bien celui attendu pour tous les jeux.

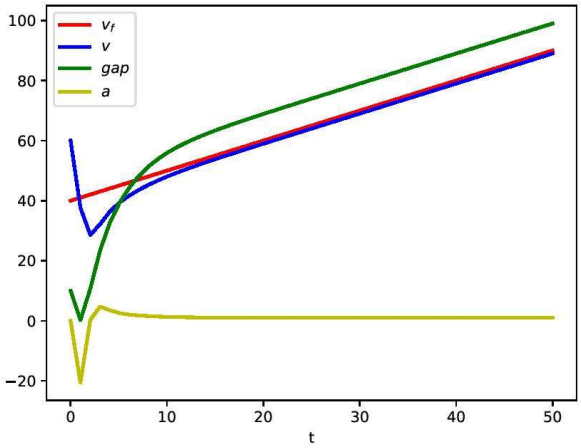

Figure 6 Variation des variables du modèle en fonction du temps durant la régulation de l'inter-distance entre un véhicule et une cible

Cette trace ne peut se générer par une machine que sur des valeurs discrète pour les variables temps et vitesse. Si l'on adopte un pas de temps fixe de 5 unités et on note la vitesse à chacun de ces instants nous allons rater l'évènement important qui se produit dans les premiers instants de la simulation : Cet évènement est celui de la baisse de la vitesse du véhicule A pour éviter sa collision avec le véhicule B. En effet A est assez rapide par rapport à B initialement (60>40) le véhicule est dicté par l'équation différentielles d'appliquer un freinage violent pour éviter que l'inter-distance gap devient négatif c.à.d. une collision. Il faut donc adopter un pas de temps adaptatif par rapport aux changements : Plus les changements se produisent dans un court intervalle de temps, plus le pas de temps doit être petit et vice versa. Ces méthodes sont connues comme **les schémas d'intégrations à pas variable d'Euler/Runge-Kutta** qui permettent d'adapter le pas d'une manière automatique aux changements pour avoir des traces de simulation qui approximent au mieux les traces réelles du modèle. De tels schémas sont à la base des outils de simulations numériques notamment le fameux algorithme ode45 de Matlab & Simulink.

2.3. Conclusion Partielle

Dans cette partie nous avons introduit des notions générales de simulation à base de modèle. Un simulateur prend un modèle en entrée et un ensemble de paramètres et fournit une ou plusieurs traces d'exécution en sortie. Un modèle simulable est une vue abstraite d'un système réelle ou d'un autre modèle et est décrit par un langage de modélisation ayant une sémantique précise. Les traces d'exécutions sont une représentation des comportements permis par le modèle et sont le plus souvent utilisés pour vérifier des propriétés de sûreté, pour valider des tests, des exigences et ainsi de suite. Nous avons vu que la simulation des systèmes complexes entraine des traces d'exécutions qui ne peuvent pas être calculé d'une manière exact et doivent être approximés. Des techniques bien connues de l'état de l'art comme les méthodes d'Euler/Runge-Kutta permettent d'avoir de bonnes approximations de ces traces et de réduire l'erreur d'approximation induite.

3. Niveau d'automatisation d'un véhicule autonome

Dans cette partie nous rappelons les cinq niveaux d'automatisation d'un véhicule autonome tel défini par la Société d'Ingénieurs d'Automobiles (SAE) et rappelons ce que c'est que les domaines d'opération de conception (ODD). Ces définitions nous seront utiles pour aborder et décrire la complexité engendrée par la simulation d'un véhicule autonome.

Plus le niveau d'automatisation d'un véhicule autonome est élevé, plus il est capable de gérer un plus grand nombre de situation de la vrai vie. Ceci entraine que le modèle sous-jacent sera donc plus compliqué et constituera ainsi un défi plus important quant à sa simulation.

3.1. Domaines d'opération de conception (Operational Design Domain [ODD])

Pour définir les différents niveaux d'automatisation, il faut rappeler ce que c'est qu'un ODD. Plus un véhicule possède un niveau d'autonomie élevé, plus son ODD est large.

- ***Système de conduite autonome*** (Automated Driving System [ADS]) système qui assure des fonctions de déplacement du véhicule dans un certain contexte sans l'intervention du conducteur. Ce terme ne permet pas de déterminer le niveau d'automatisation du système mais permet d'affirmer qu'un système ADS est doté, au moins, d'une fonction de conduite automatique.

- **Les *tâches de conduite dynamiques*** (Dynamic Driving Task [DDT]) : sont un ensemble de tâches opérationnelles, temps réel nécessaires pour opérer le véhicule sur une route en plein trafic (ces tâches excluent celles qui sont stratégiques comme l'ordonnancement du voyage, temps et position de départ/d'arrivée, les points de passage…). Des exemples de tâches DDT :

    - Contrôle du mouvement latéral du véhicule (déplacement vers la gauche ou la droite)
    - Contrôle du mouvement longitudinal (par accélération et freinage, déplacement rectiligne)
    - Reconnaitre l'environnement dynamique durant la conduite (détection d'obstacle et d'évènement, classification) et préparation de la réaction du système
    - Exécution de la réaction du système calculé en utilisant la loi de contrôle commande adaptée

- ***Les domaines d'opération de conception*** (Operational Design Domain [ODD]): Il s'agit des conditions spécifiques pour lesquelles un système de conduite autonome est conçu pour opérer. Ces conditions identifient un périmètre dans lequel se focalisera les analyses ultérieures de sûreté et de validation. Par ailleurs, le véhicule autonome conçu pour opérer dans un ODD donnée, doit pouvoir identifier s'il est à l'extérieur de son ODD ou pas afin de prendre des mesures préventives. Des exemples de domaines d'opération ODD :
    - Géographie du terrain
    - Type de voie
    - Intervalles de vitesse
    - Conditions météorologiques
    - Période du jour (matin - après midi - soir)
    - Les lois de circulation routières

3.2. Définition des Niveaux d'automatisation

Dans le tableau suivant, on résume la description des cinq **niveaux d'automatisation d'un ADS** tel définit par la SAE:

Niveau d'automatisation		Description
0	Aucune d'automatisation	Les tâches DDT sont assurées **uniquement** par le conducteur humain.
1	Assistance de conduite	Le ADS automatise les tâches de **contrôle** du mouvement **longitudinale** ou **latérale** (et pas simultanément) du véhicule sans faire les autres tâches DDT. Ces derniers seront toujours assurés par le conducteur humain pour ce niveau.
2	Automatisation de conduite partielle	Le ADS est capable d'automatiser le contrôle du mouvement **longitudinal** et **latéral simultanément** du véhicule. Le conducteur humain doit assurer la **détection d'obstacle** (ex : piétons) et **d'évènement** (ex : changement de couleur de feu) et **surveille** le ADS.
3	Automatisation de conduite conditionnel	Le ADS automatise tous les DDT relatives à un ODD mais **peut demander l'intervention du conducteur** à travers une signalisation. Le conducteur est supposé capable de prendre la main sur le véhicule.
4	Automatisation de conduite élevée	Le ADS **automatise tous les DDT** pour un périmètre ODD donnée et suppose qu'à aucun instant le conducteur peut intervenir pour une prise en main du véhicule. A ce niveau, le ADS est supposé, en cas de panne ou d'échec d'une fonction, de pouvoir conduire le véhicule à un arrêt sûr (un point à risque minimal) sans intervention humaine (à bord, ou à distance).
5	Automatisation de conduite totale	Suppose que le ADS automatise tous les DDT s**ans être limité à un périmètre ODD** spécifique et sans intervention du conducteur.

Remarque : Le niveau 3 d'automatisation entraine des défis vis-à-vis de la reprise en main du véhicule par le conducteur. En effet, plus le système de conduite est avancé (et donc proche du niveau 4), plus le demande d'intervention du conducteur se fera dans des contextes compliqués demandant des temps de reprise en main dans des délais de plus en plus petit. Cette opération peut avoir lieu sans que le conducteur possède une idée claire du contexte qui précède la reprise, lui permettant de prendre la main avec plus de sécurité. Ce problème n'existe pas au niveau 4, toutes les personnes à bord d'un véhicule de niveau 4 sont de simples passagers.

4. Cycle de développement d'un système complexe et architecture système

Dans cette partie nous rappelons les différentes phases de conception d'un système complexe et les concepts d'architecture système. La simulation peut accompagner le développement d'un système complexe telle qu'un véhicule autonome depuis les phases initiales de conception jusqu'à la mise en opération du système. Nous montrons dans cette partie comment la simulation accompagne ces différentes phases.

4.1. Cycle de développement

Nous revisitons le cycle en V ou « **V-Model »** (V-Model, s.d.) qui est une représentation graphique du cycle de développement d'un système complexe.

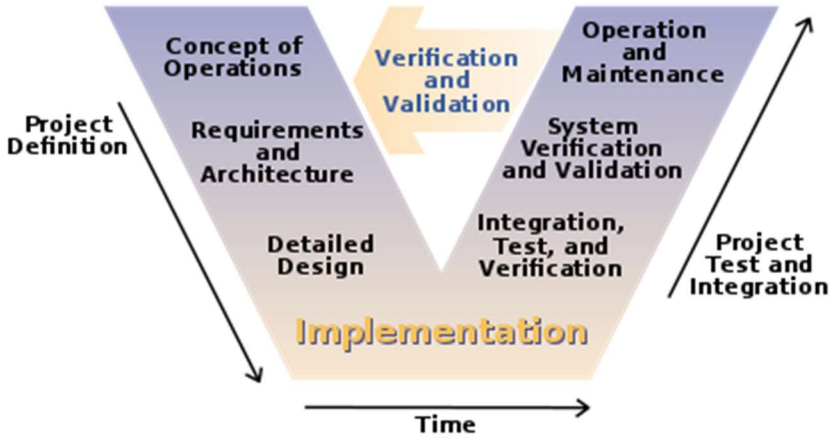

Figure 7 Cycle en V de développement d'un système complexe

Le cycle spécifie :

- un ensemble d'activités nécessaires au développement d'un système complexe
- les entrées nécessaires au bon déroulement de chaque activité et les résultats souhaités produits par chacune d'elle
- un enchainement suggérant un déroulement chronologique des différentes activités
- une définition hiérarchique des tests de validation du bas niveau (composant) au plus haut niveau (sous-système et système)

La partie gauche du **cycle en V** illustré sur la Figure 7 illustre en partant du besoin, l'élaboration et décomposition des exigences tandis que la partie droite représente l'intégration et validation du système. Le **cycle en V** permet d'organiser et situer l'avancée de la conception d'un système. Réellement les processus du cycle en V génériques sont le plus souvent adaptés au système spécifique en question. Plusieurs outils logiciels d'architecture et ingénierie système adoptent le schéma de conception en V (ces outils seront introduits ultérieurement). Rappelons certains avantages des **cycles en V :**

- **Séparation entre besoins, exigences, architecture et modélisation :** Cette séparation entre les différentes phases de conception d'un système permet de mieux évaluer la maturité du projet de conception et attribuer les différentes tâches à effectuer à des personnes spécialisées.
- **Processus générique et interdisciplinaire :** Le processus de développement décrit par le cycle en V est génériques à tout système, certains travaux étendent et adaptent le

cycle en V pour la conception des entreprises et des organisations. Par ailleurs, le cycle en V permet intervenir des domaines interdisciplinaires souvent très nécessaires pour la conception des systèmes complexes.

- **Tests et simulation pour chaque phase:** La conception incrémentale induite par le cycle en V permet d'établir des tests dès la conception jusqu'au développement et mise en opération du système ceci permet d'accompagner plus aisément la validation du système. Quand on dispose d'un modèle simulable, les cas de tests sont des cas de simulations.
- **Faciliter son exploitation par les outils logiciels :** Comme dit précédemment la structure claire du cycle en V permet aux outils logiciels de supporter efficacement ses différentes phases de développement.

4.2. Architecture système

L'architecture système est un champ interdisciplinaire qui se focalise sur la **conception**, la **gestion** et **l'intégration** d'un système complexe durant les différentes phases des **cycles de vie**. Dans les phases préliminaires de la conception d'un système, l'architecture système consiste à partir du besoin des partie-prenantes et les organiser en exigences. Dans ce qui suit nous rappelons quelques activités fondamentaux de l'architecture système en tentons de relier chacune de ces activités avec la simulation.

4.2.1. Besoins et Ingénierie des exigences

Dans cette phase, les parties-prenantes souhaitant concevoir le système se réunissent et mettent en commun leurs différents besoins. Ils procèdent par les étapes suivantes :

- **Lister le besoin**: les parties prenantes sont interrogées par les concepteurs du système à propos de leur besoin
- **Analyse** des exigences et négociation : Les exigences sont identifiés et toutes contradictions entre-elles sont réglés avec les parties-prenantes
- **Validation** des exigences : une fois le système est modélisé, une étape de validation des exigences consiste à tester leurs consistances et leurs prises en compte
- **Gestion** des exigences : assurer le changement des activités d'architecture système entrainées par une modification d'une exigence ou de son extension

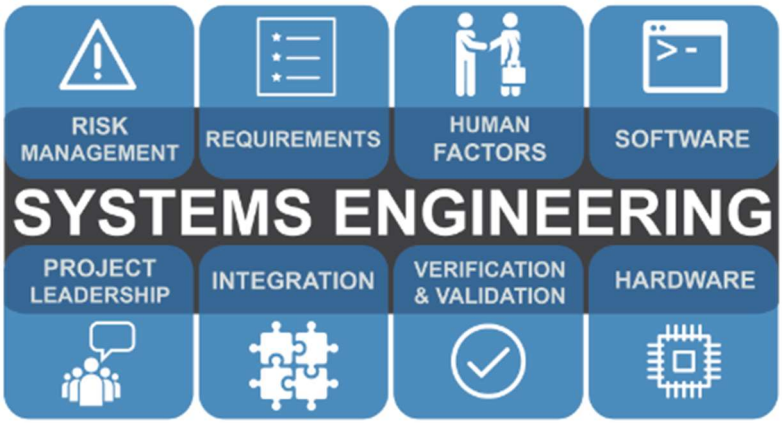

Figure 8 L'ingénierie système pluridisciplinaire (source INCOSE)

L'ensemble des exigences peuvent être classés en catégories facilitant leur exploitation par les activités d'architecture ultérieure. D'une manière générale nous pouvons classer les exigences en exigences :

- **Opérationnelle :** celles liés à préciser les situations dans lequel le système opère et les conditions nécessaires pour l'opération du système
- **Fonctionnelle :** décrivant de la capacité du système à faire une activité en précisant à partir des entrés du système les opérations permettant de fournir les sorties du système
- **Contrainte :** peuvent être de l'ordre technique indiquant l'obligation d'utiliser un moyen technique pour effectuer une activité mais aussi financier (coût pour une activité),
- **Performances :** dressant des métriques sur l'efficacité que le système doit atteindre

Simulation et exigences: Les exigences peuvent être considérées comme un point d'entrée pour la simulation. En effet, elles fournissent des éléments permettant de mettre en place un cas de test et donc de simulation du fait qu'une exigence décrit une fonctionnalité du système. Mais aussi les exigences donnent une idée sur le résultat attendu d'une simulation : le comportement simulé doit être celui attendu et conforme au besoin et donc aux exigences. Par exemple une exigence indiquant que la vitesse d'un véhicule système ne doit jamais dépasser une vitesse seuil implique que toutes les traces de simulation système ne doivent pas montrer une valeur pour la variable vitesse qui la dépasse. La traçabilité des exigences aux différentes parties du modèle est donc fondamentale.

Cycle en V et ADAS Le cycle en V est adopté pour la conception des systèmes d'aides à la conduite ou véhicule autonome (ADAS), ceci pour les raisons suivantes :

- Les fonctions ADAS deviennent de plus en plus interactives avec les différents organes des véhicules : volant, commandes d'accélération et de freinage, feux…
- De plus en plus de fonctions ADAS deviennent critiques en terme de sûreté de fonctionnement du fait qu'ils puissent agir, d'une manière autonome, sur la trajectoire du véhicule ;
- Le degré d'autonomie de plus en plus élevé (niveaux 2, 3, 4) des fonctions ADAS requiert un processus de développement fiable et traçable ;
- Le caractère de plus en plus complexe des fonctions ADAS et le prix relativement cher des différents composants liés à ces fonctions ADAS requière un processus de développement structuré et reproductible pour permettre de réduire les risques de l'impact économique lié au coût de développement.

4.2.2. Analyse opérationnelle :

Cette phase de conception du système consiste à regarder le système en tant que boite noire, c.à.d. de faire une analyse sur ce qu'on fournit en entrée au système, et ce qu'il fournit en sortie à son tour. Il s'agit aussi de préciser des ressources dont le système aura besoin au préalable. Plusieurs activités d'ingénierie système doivent avoir lieu pour avoir une architecture opérationnelle complète :

- **Acteurs externes et cas d'utilisation :** identifier tous les acteurs susceptibles d'interagir avec le futur système, leurs attentes du système, comment ils vont s'en servir, par quel moyen et pour quel objectif. Plus formellement, nous pouvons identifier un ensemble de flux entre le système et ces acteurs et détailler le contenu de chaque flux (matière, information, énergie,…)

- **Scénarios opérationnelles :** établit des scénarios durant lesquelles les acteurs externes identifiés interagissent avec le système et décrit le déroulement de ces interactions par rapport au temps

Liens avec la simulation : Cette activité décrit l'environnement externe du système et donc précise ce que l'environnement fourni au système. La simulation du système doit souvent être alimentée par ces entrées que ça soit par des simulateurs d'environnement ou des données entrée manuellement. Les différents jeux de simulation doit considérer les différents interactions identifiés par l'analyse opérationnelle du système. Cette phase précise également ce que le système fourni en sortie sans donner les détails sur comment il va le faire. Une partie des résultats de simulation attendues peuvent être tracés donc vers les sorties du système tel précisés par l'architecture opérationnelle.

4.2.3. Architecture Fonctionnelle ou Logique

L'architecture fonctionnelle est un modèle dont le but est d'identifier et représenter l'ensemble de capacités que le système doit posséder pour répondre aux exigences et donc aux besoins. Ces capacités sont représentées par un ensemble de fonctions. Une fonction prend des entrées, effectue un travail et fournit une sortie et est responsable d'assurer une part des capacités du système. Les fonctions interagissent entre-elles, les sorties de l'une peuvent être l'entrée pour une autre. Mais aussi les fonctions interagissent avec l'environnement, les fonctions reçoivent des éléments de l'environnement de nature physique ou digitale et peuvent fournir également des éléments à l'environnement.

Figure 9 présente une architecture fonctionnelle d'un système de transport intelligent générique. Cette architecture appelé Arc-IT est proposé par le département des transports des États-Unis (Etats-Unies, s.d.) et sera adopté dans les procédures d'homologation des futurs systèmes de transport intelligent.

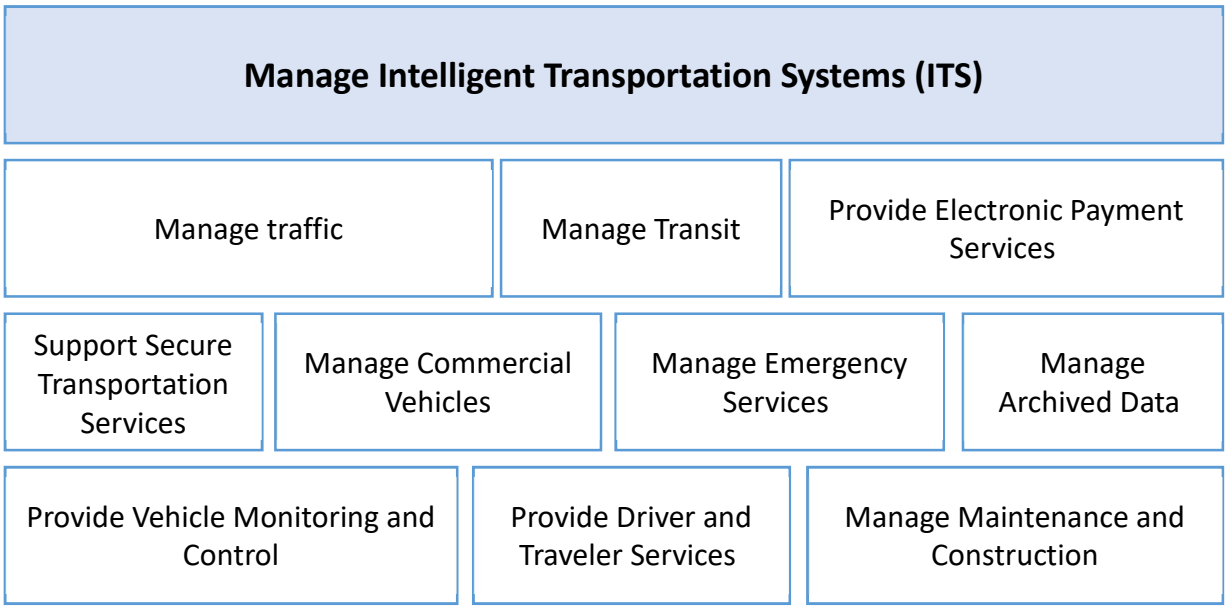

Figure 9 Exemple de décomposition fonctionnelles des capacités d'un système de transport intelligent proposé par Arc-IT (Etats-Unies, s.d.)

Liens avec la simulation : Le but de la simulation est de calculer les comportements permis par le modèle. Les modèles complexes peuvent exhiber un nombre infini de comportement possibles, la simulation doit donc viser de couvrir 'au mieux' les différentes situations dans lequel le système modélisé peut se retrouver et de calculer la réaction du système face à cette situation.

Cependant, comment pouvons-nous orienter 'au mieux' les simulations pour assurer une bonne couverture des différents comportements? En effet, il est impératif de simuler toutes les fonctionnalités du système. Par conséquent il est intéressent de dériver des cas de simulations en regardant les fonctions de l'architecture fonctionnelle. La capacité en test et en simulation est toujours limitée, ceci est dû à l'explosion combinatoire du nombre de situations possibles d'une part et au coût nécessaire pour faire une simulation d'autre part. Supposons que le nombre de simulation qu'on peut effectuer est de 20 simulations. Et nous voulons tester l'architecture fonctionnelle d'Arc-IT ayant 10 fonctions de haut niveau. Il sera donc intéressent de répartir cette capacité de simulation sur les différentes fonctions en attribuant pour chaque fonction une simulation au moins, et les 10 simulations restantes de les attribuer aux fonctions plus importantes (en terme de criticité par exemple). Mais il est important tout de même d'attribuer une part de cette capacité à simuler les interactions entre fonctions, quand l'une des fonctions fournit les entrées d'une autre.

4.2.4. Architecture Technique

L'architecture technique (aussi appelé organique ou physique) dresse un ensemble de moyens techniques, algorithmiques et technologiques qui permettent d'assurer l'ensemble des fonctions définis par l'architecture fonctionnelle. Elle définit les sous-systèmes du système global et les composants qui forment chaque sous-système ainsi que les interactions entre composants et sous-systèmes. La Figure 10 montre la décomposition d'une partie du système de transport intelligent proposé par Arc-IT (Etats-Unies, s.d.) en sous-systèmes, les interfaces de chacun ainsi que les flux d'interactions entre eux.

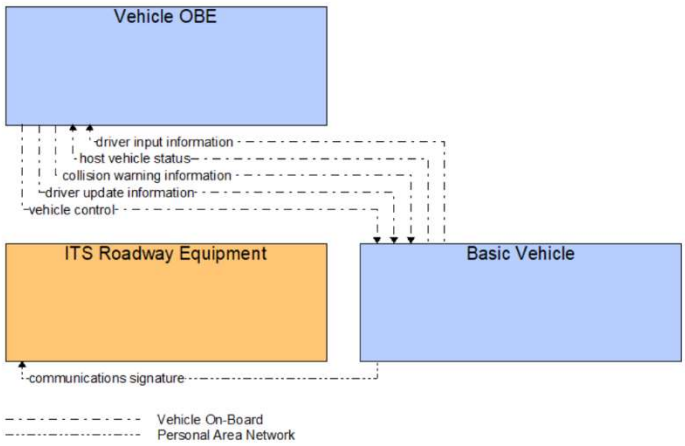

Figure 10 Interfaces entre sous-systèmes d'un système de transport intelligent proposé par Arc-IT

Liens avec la simulation : Comme pour l'architecture fonctionnelle, l'architecture technique peut préparer la simulation. Chaque composant possède des interfaces d'entrées/sorties, si nous possédons un modèle du composant réelle, nous pouvons alors simuler le modèle en fournissant les entrées nécessaires.

4.3. Conclusion partielle

Dans cette partie nous avons rappelé les modèles de conception de l'architecture système et du cycle en V dont les activités sont aujourd'hui nécessaires pour le développement des systèmes complexes. Nous avons établi des liens entre l'architecture système et la simulation. Nous avons vu que la simulation peut se préparer dès les phases préliminaires de conception (exigences/besoins) et jusqu'à la mise en opération du système. L'architecture système fournit des informations nécessaires pour orienter au mieux la simulation.

5. Simulation à base de scénarios

1. Motivation

Le raisonnement à base de composants n'est pas suffisant en ce qui concerne la validation des véhicules autonomes et plus généralement les systèmes complexes. L'échec du système peut venir d'une mauvaise interaction entre composants. En conséquence, des accidents peuvent avoir lieu sans avoir un mauvais fonctionnement des composants (panne, défaillance) séparément. Les approches à base de simulation permettent de mieux appréhender le comportement du véhicule en tant qu'ensemble. Une simulation donnée est une instanciation d'un scénario spécifié avec un grand niveau de détails. Le scénario permet d'illustrer un cas d'usage précis.

Ainsi, dans cette partie on reprend des éléments de l'état de l'art concerné pour la spécification de scénario à différents échelles de détails et donc différents niveaux d'abstraction.

Par rapport au langage, vocabulaire, taxonomie et ontologie de spécification de scénario : La spécification des scénarios nécessite un langage commun à tous les parties-prenantes avec des termes **distincts** afin de relever toutes ambigüités. Pour pouvoir exprimer des scénarios au niveau de modèle exécutable, il est important que le langage de spécification ait une sémantique claire et précise pour faciliter la formalisation des scénarios pour alimenter les simulateurs. Des sources très utiles pour définir un tel langage sont les standards tel que les guides de régulation des trafics ou ceux plus liés à la construction d'une autoroute, le code de la route etc.

2. Niveau d'abstractions des scénarios

Cette partie rappelle des éléments de la publication (Menzel, 2018) travail important qui indique le niveau de détails des scénarios durant les différentes phases de leur spécification et à des échelles d'abstraction différents. Les auteurs de l'article proposent trois classes d'abstraction du plus au moins abstrait: *fonctionnel*, *logique* et *concret*. Ils proposent également comment ces classes d'abstractions peuvent accompagner un suivi adéquat de la norme ISO 26262. L'ISO 26262 défini des phases pour le développement du système, les auteurs montrent comment les scénarios peuvent intervenir dans chacune de ces différentes phases. Indépendamment de la classe, le scénario est un déroulement séquentiel d'évènement qui se produit entre les acteurs du scénario.

Scénario fonctionnel : Un scénario fonctionnel est exprimé en langage naturel en **utilisant le vocabulaire** à élaborer du contexte. Il s'agit d'une spécification haut niveau, un ensemble de scénarios fonctionnels permet de représenter un cas d'usage. La spécification des scénarios fonctionnels nécessite d'avoir bien précisé le langage naturel relative au cas d'étude (langue, vocabulaire, taxonomie), ceci est important pour avoir des scénarios consistants même à ce niveau abstrait. A ce niveau, on peut préciser le contexte du scénario qui définit les différents éléments intervenant dans la description du scénario y compris les acteurs et ensuite de dresser le séquencement par rapport aux temps des évènements tout en précisant les acteurs intervenant durant chacun des évènements.

> *Exemple de scénario fonctionnel :*
>
> **Contexte :** La route est dédiée aux véhicules autonomes uniquement, elle comprend deux voies à sens opposés. Les entités qui interagissent durant ce scénario sont : le véhicule autonome, les

passagers à bord du véhicule, un lieu d'arrivée, les passagers en attente du véhicule dans un lieu de départ.

Scénario :

Le véhicule autonome circule sur la route dédiée aux véhicules autonomes jusqu'à son arrivée au lieu de départ. Une fois le véhicule arrivé, les portes du véhicule s'ouvrent et les passagers montent. Un signal sonore approprié permet d'indiquer la fermeture prochaine des portes. Les portes se ferment et le véhicule suit la route vers le lieu d'arrivée.

Scénario logique : Un scénario logique décrit à l'aide d'un ensemble de variables d'états les entités envisagées au niveau fonctionnel. Des contraintes sont également utilisées pour décrire la relation entre les différents variables, permettant ainsi de représenter les interactions entre les entités décrites précédemment au niveau fonctionnel. Pour chacune de ces variables est associé un domaine de valeurs. Des notations formelles peuvent être utilisées dans la description du scénario logique. La description du scénario logique couvre les éléments nécessaires pour pouvoir ensuite générer des tests et des cas de simulation. Il est possible d'attribuer des distributions de probabilités pour modéliser une non-certitude sur les valeurs prises par les variables d'états, quand celle-ci ne sont pas connues.

Exemple de scénario logique : Plusieurs scénarios logiques peuvent décrire le scénario fonctionnel précédent. On propose un scénario logique qui en dérive comme suit. La vitesse v du véhicule limitée à 70km/h commence à diminuer en s'approchant du lieu d'arrivée jusqu'à atteindre 0. Ceci se produit une fois que la distance relative entre le véhicule à une position x (par rapport à un référentiel) et la position du lieu de départ $x_{départ}$ devient inférieure à une certaine tolérance tol. Le tableau suivant résume les variables ainsi que leurs domaines et illustrent quelques contraintes à respecter. Les contraintes ne sont pas illustrées exhaustivement dans le tableau.

Variables	Domaines des Variables	Contraintes sur les Variables
x : position du véhicule à un temps t	$D_1: x > x_{départ}$ $D_2: x = x_{départ}$ $D_3: x < x_{départ}$	$x = x_{départ} + tol$ pendant la durée du transfert $d_{transfert}$ avec $0 < tol < 1e^{-2}$
v : vitesse du véhicule à un temps t	$v \in [0,70]$	$C_1: v = \frac{dx}{dt}$: la vitesse est la variation de la position vis-à-vis du temps
a: accélération du véhicule à un temps t	$a \in [-10,0]$ (freinage) $a \in [0,10]$ (accélération)	$C_2: a = \frac{dv}{dt}$: l'accélération est la variation de la vitesse par rapport au temps
feu	feu = rouge | vert | Orange	Changement périodique de période τ
$porte$: booléen état des portes	$porte = Faux \| Vrai$	

Scénario concret : A ce niveau des valeurs précises seront attribués aux variables d'états en respectant les domaines attribués par le scénario logique père. Les contraintes associées peuvent être également raffinée pour permettre une meilleure description des relations entre les

variables et vis-à-vis du temps (qui est également une des variables d'états). Le choix des valeurs concrètes doit être représentatif, en effet il existe une infinité de scénarios concrets possibles pour un même scénario logique. A cause de la combinatoire des situations différentes qui en déclinent il sera impossible de simuler l'infinité de scénarios concrets qui en découlent. Les scénarios concrets simulables doivent être choisis de telle sorte à caractériser les différences notables de comportements (pour assurer une 'bonne' couverture). Des métriques de similarité/différence entre les scénarios sont proposées dans l'état de l'art qui permet de fournir un score qui facilitera le choix des scénarios à simuler mais qu'on ne couvrira pas dans le présent document.

Exemple de scénario concret : On propose d'associer un scénario concret au scénario logique précédent. Pour cette fin on peut attribuer une position et une vitesse initiale du véhicule autonome et un exemple de contrainte plus raffinées.

Variables	Valeurs	Contraintes
x : position du véhicule	$x(t=0) = 0$ $x_{départ} = 500\ m$	$tol = 1e^{-3}$
v : vitesse du véhicule	$v(t=0) = 0$	
a: accélération du véhicule	$a(t=0) = 5$	Si $x(t) < x_{départ} - 200$ alors $a(t) = 10$ Si $x(t) < x_{départ} - 200$ alors $a(t) = f^{+}(t)$ avec f^{+} croissante Si $x(t) > x_{départ} - 200$ et $x(t) < x_{départ}$ alors $a(t) = f^{-}(t)$ avec f^{-} est décroissante Si $x(t) > x_{départ} + 200$ alors $a(t) = 10$
feu	feu = rouge | vert | Orange	$\tau = 15\ secondes$

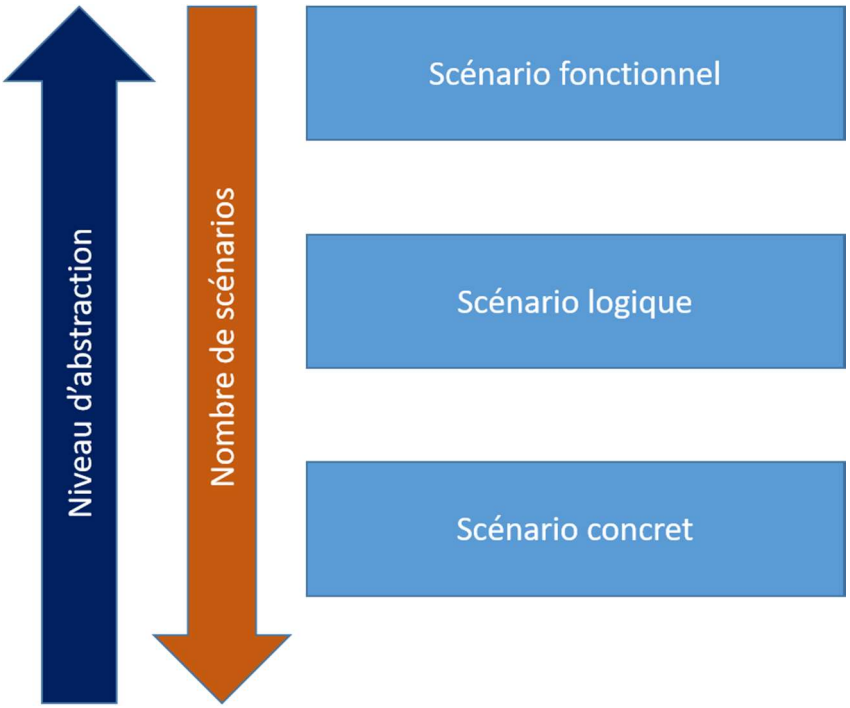

Figure 11 Niveau d'abstraction des scénarios

Niveau d'expressivité : Si on assimile l'expressivité des classes d'abstraction des scénarios à un automate simple (transitions et états), on peut décrire un scénario fonctionnel par un ensemble de chemin possible dans cet automate (chaque chemin est une séquence d'états coloriés en bleu dans la figure suivante) modélisant ainsi le non-déterminisme que celui-ci entraine. Un scénario logique correspondra à un ensemble plus restreint de ces chemins et un scénario concret par un chemin unique.

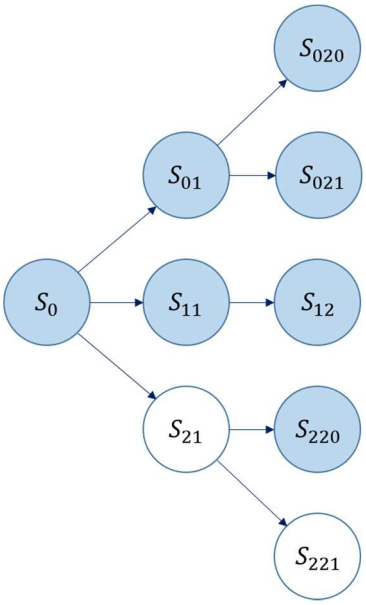

Figure 12 Scénario fonctionnel

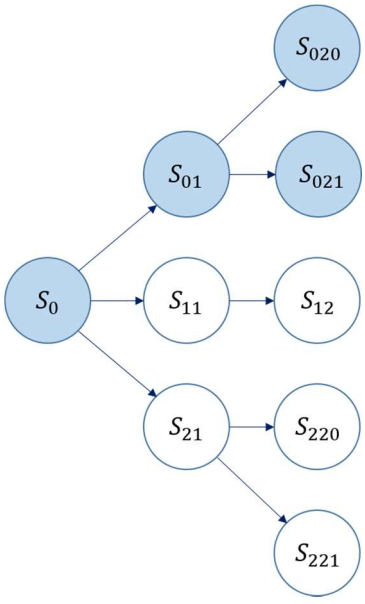

Figure 13 Scénario logique

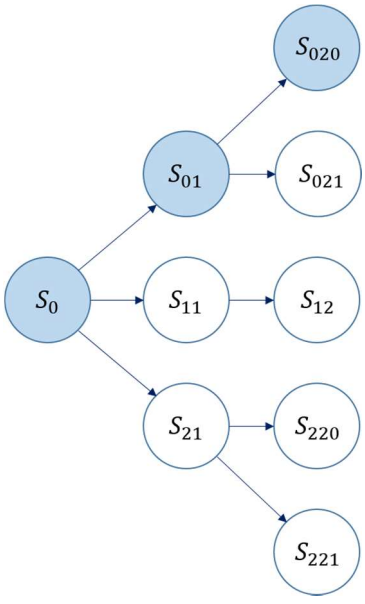

Figure 14 Scénario concret

5.1. Langages existants de spécification de scénarios

Les approches à base de scénarios sont de plus en plus utilisées pour décrire, caractériser et analyser le comportement des systèmes complexes. Des travaux important de l'état de l'art ont proposent des moyens pour pouvoir exprimer les scénarios et les exploiter à des fins de simulation. Nous rappelons dans cette section des langages de spécification de scénarios existants de l'état de l'art.

5.1.1. Openscenario (http://www.openscenario.org/) et ASAM:

Openscenario proposé par Vires et le projet Pegasus comme un standard de spécification de scénarios. Les raisons sur lesquelles les contributeurs s'appuient dessus pour motiver la nécessité d'un format générique de spécification de scénario sont diverses parmi lesquelles :

- Le besoin de faire un grand nombre de tests exhaustifs pour la validation des systèmes de conduits autonomes (ADAS)
- Les outils existants diffère largement les uns des autres réduisant ainsi la portabilité des fichiers : migrer un scénario d'un outil à un autre prend beaucoup d'effort.
- Les autorités réglementaires ne sont pas capables de fournir un format générique de spécification de scénario qui sera compatible avec les outils différents

En format XML, un scénario dans le format OpenScenario est composé d'éléments statiques et dynamiques capable d'être exporté dans des formats compatibles avec les outils existants. Il s'agit d'un projet en cours ; ses différentes phases sont présentées partant de la définition du format générique, à sa maintenance jusqu'à la définition plus précise comme la définition d'une manœuvre spécifique dans le format OpenScenario.

Plus récemment, l'ASAM (Association pour le Standardisation et l'Automatisation et Mesure des Systèmes) et OpenScenario ont fusionné leurs efforts https://www.asam.net/standards/detail/openscenario/ pour la spécification du format générique de description de scénarios. Les réunions de l'ASAM avec les différents collaborateurs sont régulières et un plan de travail périodique assure un suivi de la roadmap proposé sur leur site.

5.1.2. Projet SVA à l'IRT-SystemX :

Le projet SVA (IRT-SystemX, Projet SVA), Simulation pour les Véhicules Autonomes, propose dans le livrable « Choix des formats de descriptions » un format générique de description des scénarios.

Un scénario est une suite de scènes reliées entre elles par des évènements ou des actions.

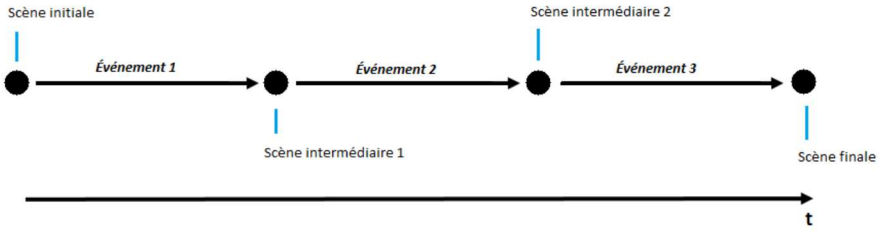

Figure 15 Description d'un scénario tel défini par le projet SVA à l'IRT-SystemX

Un scénario comprend une scène initiale, éléments statiques, dynamiques, conditions météorologiques, des évènements, des actions.

Etant donnée un scénario avec les éléments décrits précédemment, le cas d'usage tel défini par le livrable, désigne un des véhicules des éléments dynamiques comme le véhicule système en étude ou EGO dans un but de vérifier le comportement de celui-ci. Le cas d'usage permet de restreindre le nombre de situation possible décrit par le scénario. Comme dans les travaux de (Menzel, 2018), le projet définit ses propres classes d'abstraction des scénarios.

5.2. Normes concernées par les scénarios :

Cette partie a pour but de rassembler les normes concernées par la description, étude et simulation des scénarios. Cependant les normes présentées sont dans les phases préliminaires de rédaction et donc nous n'avons pas beaucoup de détails sur leurs contenues.

5.2.1. TC22/SC33/WG9 ISO WD 34501 (https://www.iso.org/standard/78950.html)

Cette norme intitulé « Termes et définitions de tests de scénarios pour les systèmes de conduite autonomes » (Terms and definitions of test scenarios for automated driving systems), spécifie des termes et des définitions pour les systèmes ADS (Systèmes de conduite autonomes, Autonomous Driving Systems). Le contenu de cette norme est applicable pour des ADS de niveau 3 et plus tel défini par l'ISO/SAE 22736. La rédaction de cette norme est suivi par les deux projets 3SA et SAM de l'IRT-SystemX.

5.2.2. TC22/SC33/WG9 ISO WD 34502 (https://www.iso.org/standard/78951.html):

Intitulé « Framework d'Ingénierie et processus d'évaluation de sûreté à base de scénario » (Engineering framework and process of scenario-based safety evaluation). Ce document fournit un guide et un framework d'ingénierie d'état de l'art pour les scénarios de test des ADS et une démarche d'évaluation de sûreté à base de scénarios. Le framework d'ingénierie clarifie le processus général d'évaluation de sûreté à base de scénarios à appliquer durant le développement du produit. Le guide et framework sont destinés pour être appliqué aux systèmes AD de niveau 3 et plus tel défini par l'ISO/SAE 22736.

5.2.3. TC22/SC33/WG9 ISO WD 34503 (https://www.iso.org/standard/78952.html):

Taxonomie pour les domaines d'opérations de conception des systèmes de conduite autonome (Taxonomy for operational design domain for automated driving systems) est un document qui spécifie les exigences de base pour une définition hiérarchique des taxonomies pour définir les domaines d'opérations de conception (ODD) d'un système de conduite autonome (ADS). Un

ODD inclus des attributs statiques et dynamiques et peut être utilisé pour développer des scénarios de tests dans lequel un ADS est conçu pour opérer. Ce document définit également les procédures de test de base pour les attributs des ODD. Ce document est applicable pour les systèmes ADS de niveau 3 et plus tel défini par l'ISO/SAE 22736

5.2.4. TC22/SC33/WG9 ISO WD 34504 (https://www.iso.org/standard/78953.html):

La norme « Attributs de scénario et catégorisation » (Scenario attributes and categorization) définit des packages concernant les différents attributs d'un scénario qui révèlent des informations nécessaires pour construire un scénario de test complet. Ces attributs peuvent référer à des sources de données complémentaires comme par exemple au les réseaux de route liés au scénario, des scènes 3D ou des modèles. Pour assurer une certaine qualité du standard, des mesures d'intégrité et d'intégrabilité de ces attributs sont définies. De plus, ce travail définit une catégorisation des scénarios en fournissant des tags qui comporte des informations qualitatives ou quantitatives sur ces scénarios.

6. Normes et réglementation concernées par la simulation numérique :

6.1. Normes

6.1.1. ISO 26262 Véhicules routiers - la sûreté fonctionnelle (Road Vehicles —Functional Safety https://www.iso.org/standard/43464.html):

L'augmentation en complexité des systèmes de l'industrie automobile implique des efforts croissants pour fournir des systèmes sûrs. L'ISO 26262 est un standard qui précise un ensemble de pratiques pour la conception et le test des produits. Il adresse particulièrement les applications en automobiles et notamment les composants avec des niveaux de sûretés critiques. Le standard dérive de l'IEC 61508, ce dernier étant générique pour les systèmes électroniques et électriques.

ISO 26262
Road Vehicles - Fuctional Safety

Ce standard permet :

- de *mesurer jusqu'à quel point* un système est sûr
- d'avoir un *vocabulaire commun* pour référencer des parties spécifiques du système
- d'accompagner le développement du système pendant tous ses phases (conception, développement, mise en production, opération,…) en donnant des recommandations de sûreté pour chacune de ces phases
- de déterminer la classe de risque du système et ses sous-parties ASIL (Automotive Safety Integrity Levels). Une fois identifié, le standard permet d'identifier les exigences de sûreté nécessaires pour avoir un résidu de risque acceptable.

Le standard spécifie donc les mesures de sûreté pour chacune des phases du système.

6.1.2. ISO 21448 Sûreté de la fonctionnalité visée (Road vehicles — Safety of the intended functionality https://www.iso.org/standard/70939.html)

L'industrie automobile utilise des processus de sûreté pour développer des systèmes sûrs dans le secteur automobile qui sont conformes au standard ISO 26262. Cependant ce dernier se focalise plus sur les défaillances dues aux sous-systèmes électriques. Or, la sûreté des véhicules autonomes ne peut pas être assurée uniquement en étudiant le comportement des sous-systèmes électriques et électroniques sous panne. En effet un mauvais fonctionnement du véhicule peut être dû à un mauvais usage d'une fonction d'opération par le conducteur ou encore des limitations de performance des capteurs du système ou des changements imprévus dans l'environnement routier. Pour adresser ces aspects, le SOTIF (Safety Of Intended Functionality) complémente l'ISO 26262. Un nouveau standard ISO 21448 cible l'étude de sûreté de l'automobile en absence de panne au niveau des sous-systèmes électriques et électroniques. Ce standard est en cours de développement et sera publié en début d'année 2019.

6.1.3. ISO 15622 :2018 Systèmes de transport intelligent – Système de contrôle adaptative de la vitesse – Exigences de performances et procédures de test (Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures https://www.iso.org/standard/71515.html)

Le document contient les stratégies de contrôle de base, les exigences de fonctionnalité minimale, les éléments de base de l'interface conducteur, les exigences minimum pour le diagnostic et la réaction et les procédures de test de performances pour les systèmes de contrôle adaptative de la vitesse (ACC).

Les systèmes ACC sont réalisés soit en tant que système agissant sur toute la plage de vitesse (Full Speed Range Adaptive Cruise Control [FSRA]) soit sur une plage limitée de vitesse (Limited Speed Range Adaptive Cruise Control [LSRA]). Les LSRA quant à eux, sont encore divisés en deux types nécessitant sois un embrayage manuel sois automatique. Les ACC visent fondamentalement le contrôle longitudinale des véhicules équipés sur les autoroutes (des routes ou les véhicules non-motorisés et les piétons sont interdits) sans congestion et pour les ACC de type FSRA avec congestion de trafic permis. L'ACC peut être augmenté d'autres capacités comme l'alerte aux obstacles devant le véhicule. Pour les ACC de type FSRA, le système va s'arrêter derrière un véhicule cible dans la limite de sa capacité de décélération et va reprendre son activité quand le conducteur demande au système de continuer le voyage à partir d'un état immobile du véhicule. Il n'est pas exigé que le système réagit aux objets stationnaires ou qui bougent lentement.

6.1.4. J2399_201409 : Contrôle de vitesse adaptative (ACC), caractéristiques d'opération et des interfaces utilisateurs (Adaptive Cruise Control (ACC) Operating Characteristics and User Interface https://www.sae.org/standards/content/j2399_201409/)

Les ACC sont une amélioration des contrôleurs de vitesse conventionnels qui permettent à un véhicule équipé ACC de suivre un véhicule devant à une plage de temps présélectionnée jusqu'à une vitesse maximal choisi par le conducteur en contrôlant le moteur, la transmission de puissance et les freins de service. Ce standard proposé par le SAE se focalise à spécifier les exigences minimales pour les caractéristiques des systèmes opérant en ACC et les éléments des interfaces utilisateurs. Ce document s'applique aux systèmes ACC dédiés aux véhicules avec passagers (motos inclus). Ce document ne s'applique pas aux véhicules à poids lourd (> 4536 kg). De plus, ce document n'adresse pas les ACC de type « stop et go » qui permettent aux véhicules équipés de ré-accélérer après un arrêt total du véhicule. Les futures révisions de ce document doivent considérer les ACC améliorés ainsi que l'intégration des systèmes d'alertes aux collisions avec les véhicules devant (Forward Vehicle Collision Warning Systems [FVCWS]).

6.1.5. ISO 19364 :2016 (https://www.iso.org/standard/64701.html)

L'ISO 19364 :2016 spécifie une méthode pour comparer les résultats de simulation d'un modèle mathématique par rapport à des données mesurées de test d'un véhicule existant en accord avec les tests des états stables circulaires de conduites tel spécifié par l'ISO 4138 ou le test à augmentation lente de direction qui est un alternative de l'ISO 4138. La comparaison est faite dans le but de valider les outils de simulation pour ce genre de test lorsqu'ils sont appliqués à des variantes des véhicules testés. La norme est applicable au véhicule de passagers tel défini par l'ISO 3833. Le test à augmentation lente de direction est décrit dans les réglementations comme le USA FMVSS 126 et le UN/ECE Regulation No. 13-H.

6.1.6. ISO 19365 :2016 (https://www.iso.org/standard/64702.html)

L'ISO 19365 :2016 spécifie une méthode pour comparer les résultats de simulation d'un modèle mathématique de véhicule pour tester les données mesurées d'un véhicule existant subissant des tests sinusoïdaux avec arrêt, typiquement utilisés pour évaluer la performance d'un système électronique de contrôle de stabilité (Electronic Stability Control [ESC]). Cette comparaison est établit dans le but de valider les outils de simulation pour ce type de test lorsqu'elle est appliquée à des variantes des véhicules de tests. La norme est applicable au véhicule passager tel défini par l'ISO 3833.

6.2. Standard concerné par la simulation :

6.2.1. FMI Functional Mock-up Interface:

FMI est un standard pour l'interopérabilité des outils de simulation. Il fournit une interface standardisé à être utilisé en simulation des systèmes complexes. Le projet MODELISAR mené par Dassault Systèmes avait pour objectif de définir les spécifications FMI.

La vision derrière l'approche FMI considère un système complexe comme un ensemble de composants interagissant ensemble et chacun régit par ses propres lois physiques. Il parait donc possible d'avoir un produit virtuel qui est un ensemble de modèles, où chacun représente un ensemble de composants avec les lois propres aux composants. A partir de cette philosophie, l'approche FMI consiste à avoir:

- un modèle de sous-systèmes décrit par des équations différentielles ordinaires, algébriques ou des équations discrètes en temps avec des états et des évènements par pas de temps
- des outils définissant les lois de contrôle
- ces outils permettent d'exporter un composant sous format d'unité fonctionnelle maquette ou FMU (Functional Mock-up Unit)
- ces FMU peuvent être importé dans d'autres environnements pour être exécuté
- Ainsi les différents FMU coopère en temps réel à travers un environnement de co-simulation

Ce processus est visuellement montré sur la Figure 16.

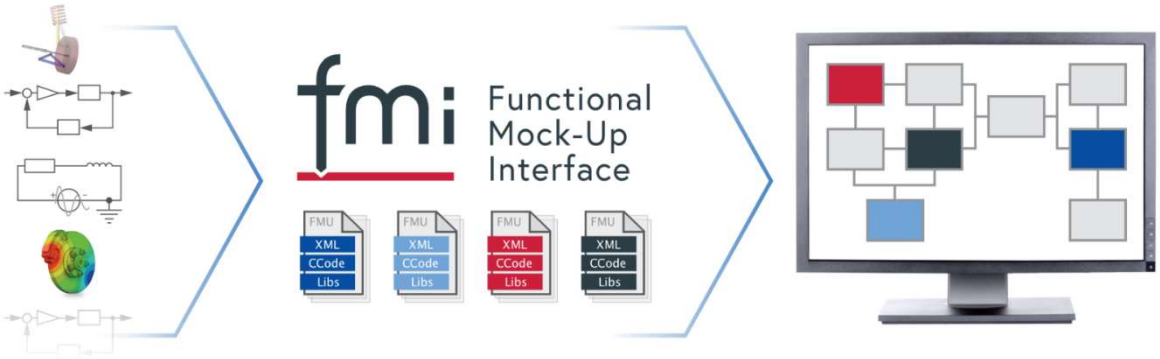

Figure 16 Le standard FMI définir une interface standardisé pour les échanges entre modèles dynamiques

6.2.2. Open Simulator Interface (OSI) :

(https://github.com/OpenSimulationInterface/open-simulation-interface) spécifie des moyens standardisés pour spécifier les conditions météorologiques, entrées de capteurs etc… Ce projet fournit un moyen générique pour décrire la perception de l'environnement par le véhicule autonome pour des scénarios virtuels. Le modèle de spécification proposé peut être utile pour combiner des spécifications issues de différents fournisseurs ou chaque fournisseur produit son interface OSI établit en suivant le format générique proposé.

6.3. Initiatives :

6.3.1. PEARS

PEARS (Prospective Effectiveness Assessment for Road Safety) est une initiative de plusieurs constructeurs européens (Renault, PSA, FCA, Volvo cars, BMW,…), de fournisseurs (Autoliv,…) et de centres de recherche sur la sécurité routière (TNO,...). Son objectif est d'harmoniser les méthodologies pour évaluer les ADAS en Europe. Parmi les objectifs, une proposition de norme ISO sur la méthode de validation par simulation numérique est en cours de rédaction.

Le document spécifie les lignes directrices et leur application pour l'évaluation prospective des performances de sécurité des technologies pré-collision dans les véhicules routiers par simulation virtuelle. Il traite des sujets suivants:

- Les objectifs d'évaluation : lignes directrices pour définir l'objectif d'une étude de simulation, y compris des mesures de performance de sécurité appropriées.

- Les données d'entrée : caractéristiques des différents types d'informations qui sont utilisées en entrée du processus. Cela comprend la définition de l'ensemble minimal d'informations nécessaires, en fonction de l'objectif d'évaluation et du cadre de simulation.

- Elément de base : méthodes permettant d'établir des cas de référence qui sont à la base de la simulation sans / avec la technologie en cours d'évaluation.

- Simulation virtuelle : caractéristiques des sous-modèles pour l'évaluation des performances de sécurité et le flux de travail de ses sous-modèles. Différentes techniques de simulation peuvent être adoptées.

- Évaluation des performances de sécurité : directives sur la façon de déterminer les performances de sécurité en termes de différentes métriques basées sur les résultats de la simulation.

- Validation et vérification des outils de simulation et de la méthode : lignes directrices pour la validation et la vérification de la méthode prospective d'évaluation des performances de sécurité. Pour chaque sous-modèle, des approches de validation et de vérification applicables sont définies.

- Documentation des études prospectives d'évaluation des performances de sécurité; lignes directrices pour documenter la manière dont les résultats ont été obtenus concernant les données d'entrée, l'établissement de la base de référence, les modèles et outils de simulation utilisés, les mesures d'estimation de l'efficacité appliquée et la validation et la vérification de la méthode et des outils.

Le document explicite la constitution du modèle de simulation (véhicule, infrastructure, trafic, environnement, technologie et conducteur) et le processus de simulation sous forme d'évaluation au cours du temps. Un modèle de collision est intégré dans ce processus.

Le principe de cette approche repose sur la comparaison entre une situation de référence (avec conducteur seul principalement) et une situation intégrant une technologie (avec contribution du conducteur éventuellement). L'objectif est de mesurer le bénéfice apporté par la technologie en termes de sécurité : collisions, blessés, sévérité,…

6.3.2. Safety First for Automated Driving, 2019 (SaFAD, 2019)

Ce document est un effort récent d'Intel et 10 autres entreprises spécialisés dans l'automobile et des entreprises du tiers X pour développer des technologies de conduites autonomes. L'objectif est de décrire un cadre de méthodologie pour le développement, le test et la validation des systèmes de conduite autonomes pour les intégrer dans les véhicules public larges.

La sûreté d'abord pour la conduite autonome (Safety First for Automated Driving [or SaFAD]) décrit en détail une approche commune sur la façon de développer et tester les véhicules autonomes, avec une focalisation d'affronter les problèmes de sécurité depuis la phase de la conception du système. En effet, le manque de standards fixes ralentit le développement des véhicules autonomes. Le document n'impose pas des contraintes qui doivent être satisfait dans le processus de développement mais vise plus à présenter un ensemble de bonnes pratiques pour développer des véhicules autonomes de niveau 3 et 4 (comme décrit par le SAE-J3016). 12 principes sont proposés sur lesquelles se base la conception sûre des véhicules autonomes.

L'initiative regroupe un grand nombre de constructeurs de véhicules Allemands comme Audi, BMW, Daimler et Volkswagen avec des entreprises chinoises comme Baidu. Néanmoins, les constructeurs français comme Renault, Nissan, Peugeot-Citroën et Japonais (Toyota, Renesas) sont entièrement absents.

Le document propose une architecture générique pour les véhicules autonomes rappel dans la

Figure 17 Architecture de haut niveau propose par Intel et 10 autres constructeurs automobiles dans le document La sûreté d'abord (Safety First). Nous pouvons identifiés les parties principales classique : Perception ; Planification et Actionnement. Ce dernier est un paradigme datant des années 1985 et qui était historiquement utilisé en robotique. Il est assez facile d'inférer la boucle percevoir ; planifier et actionner aux véhicules autonomes : De l'information provient des équipements des véhicules ; de l'infrastructure physique, le véhicule va ensuite traiter ces informations pour déduire la représentation de l'environnement et ensuite utilisera cette représentation pour planifier les actions au court et long terme qui doivent être effectuées.

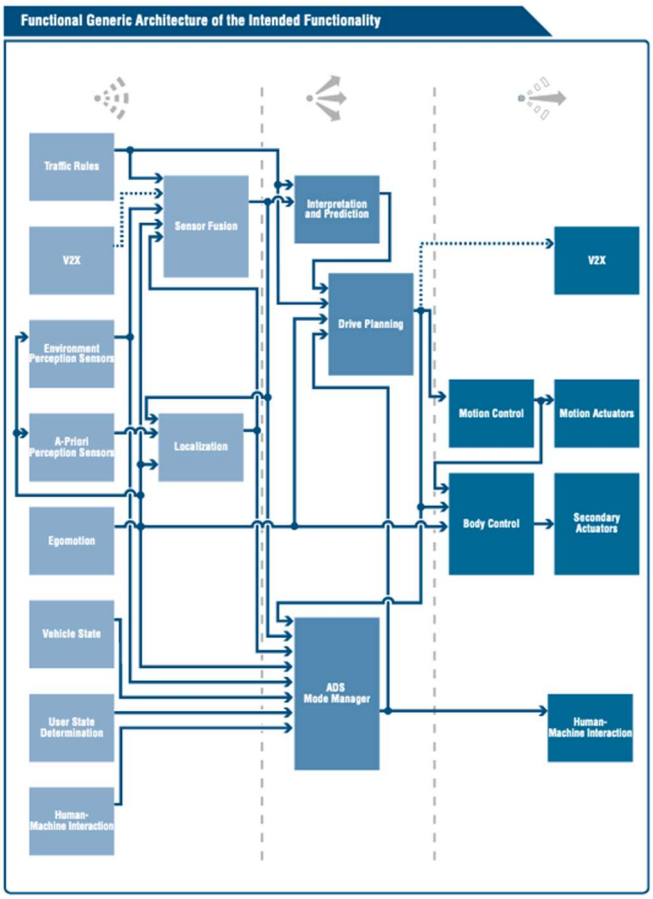

Figure 17 Architecture de haut niveau propose par Intel et 10 autres constructeurs automobiles dans le document La sûreté d'abord (Safety First)

7. Description d'outils de simulation existants:

Les outils de simulation sont indispensables pour pouvoir vérifier qu'un modèle de système complexe exhibe le comportement attendu de lui. D'une manière générale, un outil de simulation prend un modèle en entrée et un ensemble de paramètres et fournies un ou plusieurs traces de simulation en sortie. Ces traces peuvent être ensuite utilisées pour vérifier si les exigences du système sont bien satisfaites par le modèle.

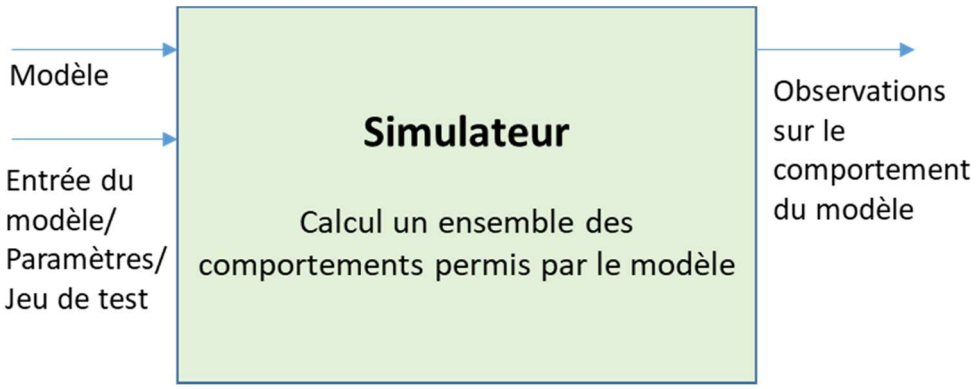

Figure 18 Schéma bloc générale d'un outil de simulation

Dans le tableau suivant nous organisons les outils de simulation par plusieurs catégories selon le genre de modèle qu'ils permettent de traiter. Nous reprenons et mettons à jour la liste des outils de simulation du livrable du projet SVA intitulé « Méthodes et outils de modélisation et de simulation des composants et des systèmes » (ISX-TA-SVA-LIV-0495, (IRT-SystemX)). Nous classons les outils de simulation selon le type de modèle proposé. Cette classification est attribuée en regardant purement les descriptions des outils et n'empêche qu'un des outils peut réellement figurer dans plusieurs catégories.

Catégories	Intitulé de l'outil
Perception, acquisition, traitement et fusion de données	RTMAPS (Intempora) ADTF (Elektrobit) vADAdeveloper (Vector) GOLD (VisLab- Université de Parme)
Simulation de dynamique de véhicules	Drive (State- Italie) CarSim (Coopération de simulation mécanique) veDYNA (TESIS DYNAware) VDL (Dymola – DASSAULT Systèmes) ASM (dspace) LMS (Siemens PLM)
Développement des IHM	Altia (altia) EB GUIDE (Elektrobit) NVIDIA DRIVE DESIGN (NVidia) Qt (Qt-digia)
Cartographie et système de navigation	SIVNav SDK (BeNomad) MacMap et Map Theme (MacMap) OpenStreetMap ADASRP
Simulation de capteurs, de conduite et d'environnement routier	Pro SiVIC PreScan CarMaker Virtual Test Drive SCANeR VR-Design Studio VRXperience
Génération et simulation de trafic routier	ASM Traffic Aimsun Sumo PTV Vissim
Algorithmes de contrôle commande	Matlab/Simulink
Génération de cas d'usages et de tests	Simulink Design Verifier MaTelo Certifylt
Architecture Système et bureautique	Microsoft Office Rational Doors arKItect Papyrus MagicDraw Capella

7.1. Outils de Perception, acquisition, traitement et fusion de données

7.1.1. Spécifications

Dans le cadre d'un développement efficace de systèmes embarqués en se basant sur des méthodes de développement à base de modèles (Model Based Development), des outils de perception et de développement d'applications de fusion de données représentent un élément principal. Ces outil soutiennent les concepteurs d'applications ADAS dans la création de nouveaux systèmes d'aide (ou délégation) à la conduite ou de sécurité active avec une multitude de modules et d'exemples prêts à l'emploi. Dans les points ci-dessous sont résumées les principales caractéristiques requises dans ce type d'outils.

- Une large librairie de modèle capteurs configurables ce qui permet de s'interfacer avec tous type de capteur (caméra, radar, GPS, capteur ultrason, …etc.) ;
- Capacité de connexion a de multiples interfaces (CAN, LIN, USB, Ethernet, WIFI, … etc.)
- Enregistrement, transfert et rejoue temps réel de données ;
- Acquisition synchrone de données ;
- Traitement en ligne ou hors ligne de données ;
- Interfaçage facile avec des modules externes (modules développés dans Matlab/Simulink ou en C/C++ par exemple) ;
- Une interface graphique simple et facile à utiliser ;
- Compatible avec un grand panel de système d'exploitation (Windows, Linux, Android …).
- Dans ce qui suit, un ensemble d'outils d'acquisition, de traitement et de fusion de données seront présentés.

7.1.2. RTMAPS- Real Time, Multisensor, Advanced Prototyping Software (Intempora)

RTMaps Studio est un logiciel qui permet de connecter facilement tout type de capteurs et actionneurs, d'acquérir, d'enregistrer, de traiter et de transmettre toute donnée, en temps réel.

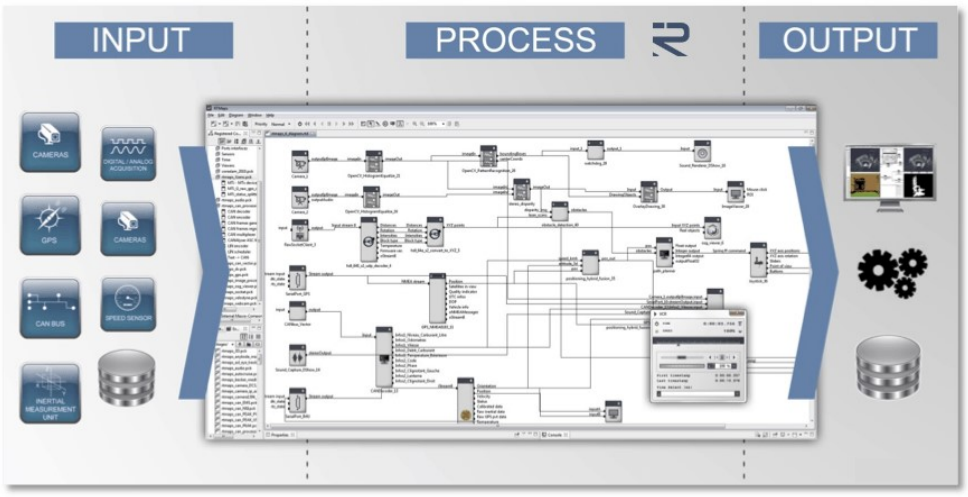

Figure 19 RTMaps Studio version 4

Le temps et les mesures jouent un rôle essentiel dans les applications industrielles (automobile, robotiques, …). De ce fait, l'acquisition et l'enregistrement des données capteurs ou véhicule sont réalisés d'une manière synchrone. En effet, les données sont datées précisément lors de leur acquisition par RTMaps. Cette datation procure une maîtrise complète des flux de données lors des traitements comme lors de la relecture des enregistrements par exemple. Au cours des essais, les situations et comportements peuvent être enregistrés pour une analyse fine ultérieure. La reproduction d'une situation est ainsi possible.

La **datation et la synchronisation** des données acquises de multiples capteurs vont permettre par exemple la **fusion de ces données.** Cette fusion va accroît la fiabilité et la robustesse des résultats tout en donnant accès à des capteurs éventuellement moins coûteux. En permettant d'explorer simplement des solutions différentes, il devient plus évident de valider rapidement les options choisies, avant chaque expérimentation réelle.

RTMaps Studio dispose d'une interface graphique de développement qui permet la mise en place rapide et l'évolution des différentes applications (traitement de données, fusion de données, contrôle, …etc.). Constituées de composants fonctionnels et indépendants permettent de gérer le développement d'algorithmes et le changement de capteurs. Les phases de tests et d'expérimentations sont alors simples de mise en œuvre.

Les composants sont mis en place par un simple glissé déposé sur la scène (l'espace de travail). Ils interfacent les capteurs, représentent les algorithmes (ou applications) et connectent les actionneurs. La souris permet de tirer des « fils » connectant la sortie d'un composant à l'entrée d'un autre. Les flux de données sont alors établis.

Pour chaque composant, un ensemble de paramètres est accessible par des boites de dialogue. Le choix des paramètres détermine le comportement du composant. Les composants sont regroupés dans des bibliothèques RTMaps. Ces composants proposent les fonctions élémentaires nécessaires à la plupart des applications abordées, ci-dessous un exemple de composants intégrés à RTMaps :

- Vidéo
- Communication et dispositifs d'entrée
- Navigation / inertielle
- Traitement du signal et des images Robotique
- BUS CAN et LIN
- Décodage de protocoles standard Affichage en temps réel
- Enregistrement et relecture de données Exportation vers des formats standards Interfaçage avec des logiciels tiers

L'enregistrement des données est déclenché par une simple pression sur le bouton record Figure 20. Le VCR Figure 20 permet la relecture des bases de données à la vitesse souhaitée. Un curseur permet d'accéder directement aux séquences pertinentes.

RTMaps Studio prend en compte la majorité des capteurs disponibles sur le marché.

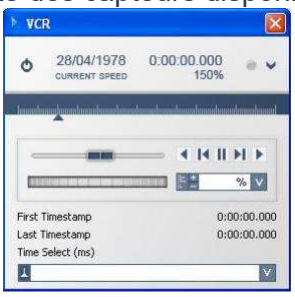

Figure 20 VCR

Intempora (Société qui a développée et qui commercialise RTMaps) fournit de très nombreux modules autorisant leur interfaçage. Il suffit qu'un appareil soit connectable à un système informatique pour permettre son intégration dans une application RTMaps. Exemples de capteurs supportés : Webcams, caméras, GPS, centrales inertielles, radars, télémètres laser, Bus CAN, ….etc.

RTMaps Studio dispose également d'un SDK « Software Development Kit » qui permet principalement de créer ses propres composants. La programmation se fait en C++ ; elle est facilitée par la présence de squelettes de codes et de macros, Figure 21.

Le SDK comprend une documentation ainsi que des exemples ou squelettes de code pour le développement de composants spécifiques. Des assistants intégrés aux environnements de développement (tels que Visual Studio de Microsoft) permettent la génération simple des projets de compilation.

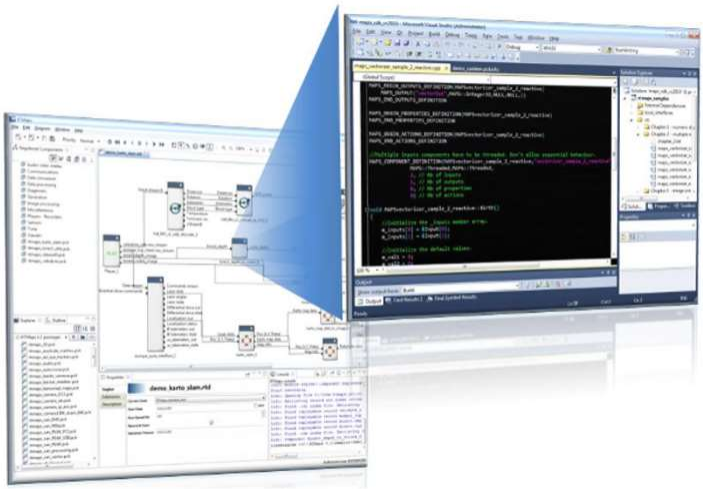

Figure 21 RTMAPS SDK

7.1.3. ADTF-Automotive Data and Time triggered Framework (Elektrobit)

EB Assist ADTF Figure 22 est un outil flexible et extensible qui est capable d'acquérir et de synchroniser les données de multiples capteurs. Il fournit des composants standards pour l'enregistrement et l'interprétation des données des réseaux CAN, FlexRay, LIN et MOST. Au-delà de l'enregistrement des données, ADTF propose des fonctionnalités temps réel de relecture, de traitement et de visualisation des données pour une utilisation en laboratoire ou sur un véhicule d'essai. L'environnement permet également le développement et la mise au point d'applications d'aide à la conduite et de sécurité active.

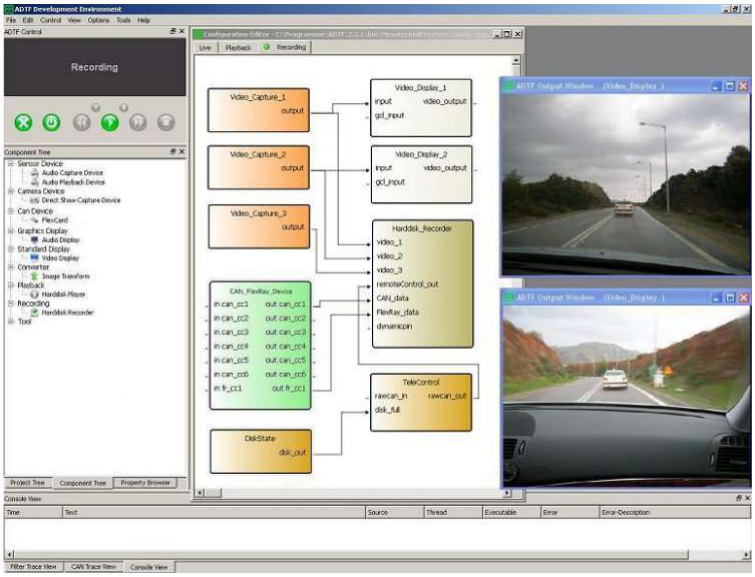

Figure 22 EB Assist ADTF

Les caractéristiques d'EB Assist ADTF peuvent être résumées dans les points suivants :

- Dispose d'une interface graphique pour la configuration et le contrôle ;
- Dispose d'une unité de capture qui permet un enregistrement précis de multiples flux vidéo de haute résolution et de bus de données (CAN, FlexRay et autres) ;
- Dispose d'une unité de lecture qui reproduit les enregistrements issus des phases d'acquisitions ;
- Permet l'enregistrement et la relecture en temps réel des données ;
- Regroupe un ensemble flexible et extensible de composants ;
- L'échange de données et des composants est simplifié ;
- Permet une visualisation en direct des données et des résultats ;
- Permet de générer et de reconstruire un horizon électronique basés sur le standard ADASISv2 ou des protocoles propriétaires ;
- Peux s'interfacer facilement avec d'autres logiciels et outils (MATLAB/Simulink par exemple).

Comme mentionné ci-dessus, EB Assist ADTF dispose de plusieurs composants qui facilitent l'acquisition, l'enregistrement et le traitement de données ainsi que le développement d'application sans avoir besoins de programmer. Ci-dessous une liste non exhaustive de composants disponible dans ADTF :

- Boite d'outils de connexion avec les composants matériels :
  - Les composants commercialisés par la société Vector[3] (CANCard, Peak CAN, MOST Vector, VN2610, SMSC Optolyzer, Vector VN3300, …) ;
  - Eberspächer FlexCard ;
  - DirectShow Video Devices ;
  - IDS µEye.

- Boite d'outils de visualisation :
    - Visualisation 3D de la scène;
    - Affichage 2D ;
    - Affichage X-Y.
- Boite d'outils de compression :
    - Compression et décompression du flux vidéo ;
- Boite d'outils d'information carte et ADASISv2 :
    - Trames GPS;
    - Simulation d'itinéraire;
    - Carte interactive.
- Boite d'outils du reconstructeur ADASISv2 :
    - Visualise l'arbre de l'horizon électronique ADASISv2 ;
    - Affichage des données textuelles ADASISv2 ;
    - Générer les statistiques d'utilisation de la mémoire.
    - Exécuter le test de conformité du reconstructeur ADASISv2.
- Boite d'outils d'enregistrement des données voiture :
    - Serveur d'enregistrement de données voiture ;
    - Connecter des interfaces utilisateurs distants
    - Connectivité à distance à des outils tiers
    - Contrôle et surveillance des fonctionnalités d'ADTF.

7.1.4. vADASdeveloper (Vector)

vADASdeveloper est un logiciel qui permet de développer et de tester des applications multi-capteurs. Dans la phase de développement d'algorithmes, vADASdeveloper est capable d'acquérir, enregistrer, visualiser et relire les données de tous les types de capteurs utilisés par les ADAS (caméras, Radar, Lidar,…). Le flux de données est manipulé graphiquement avec une mise à disposition claire. L'intégration de Visual Studio permet aux utilisateurs de développer et de tester les applications avec un seul outil.

Le type d'applications réalisables s'étend de la reconnaissance d'objet par traitement d'image à la fusion de données multi-capteurs (détection et suivi d'objets par exemple, Figure 23). vADASdeveloper est également adapté à des tâches d'analyse, d'évaluation et de prototypage des systèmes.

La bibliothèque modulaire "BASELABS4" peut être intégré de façon transparente ce qui donne aux développeurs des fonctionnalités utiles supplémentaires pour la configuration et le développement d'algorithmes de traitement du signal et de fusion de données. Autres modules en C, C++ ou MATLAB/Simulink peuvent être intégrées facilement.

Pour résumé, les fonctions et les champs d'application de vADASdeveloper peuvent être résumés dans les points suivant :

- Tous type de capteurs (caméra, radar, CAN, …etc.) peut être intégré et connecté d'une manière graphique ;
- Les données capteurs peuvent être enregistrées et rejouées d'une manière synchrone avec un horodatage précis ;

- Un affichage de résultats clair et configurable ;
- La rapide implémentation, de débogage et de test des applications dans un seul outil grâce à l'intégration de Visuel studio ;
- Une interface graphique personnalisée (utilisation de forme C#) ;
- Support de développement d'algorithmes de fusion de données via la bibliothèque "BASELABS" ;
- Génération de code C ;
- Développement d'applications multi-capteurs avec fusion de donnée en utilisant le langage de haut niveau C# ;
- Détection et suivi d'objets ;
- Traitement d'images (détection de marquages, reconnaissance des panneaux de signalisation) ;
- Détection précise de la position des véhicules.

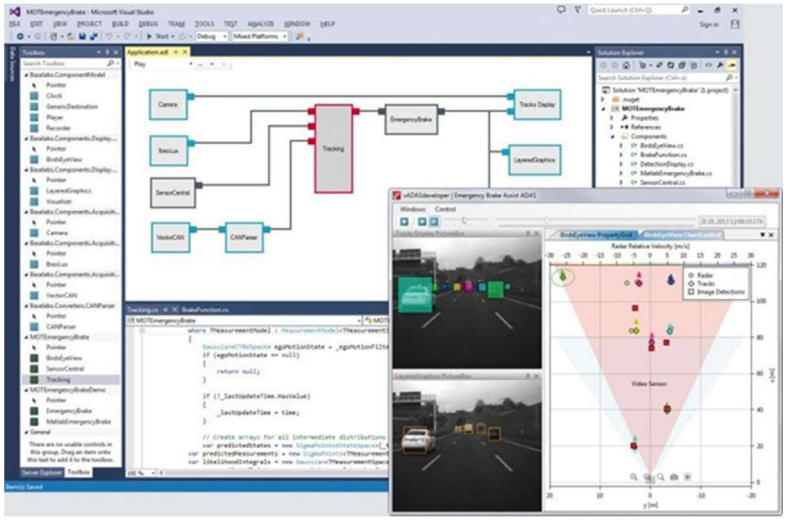

Figure 23 Implémentation, débogage et test d'une application multi-capteurs sous vADASdeveloper.

7.1.5. GOLD – General Obstacle and Lane Detection (VisLab–université de Parme)

GOLD est une structure logicielle C++ développée par le laboratoire VisLab de l'université de Parme. Il comprend des API qui lui permettent l'acquisition de données à partir de différents types de capteurs (caméra, radar, lidar, GPS, INS, CAN, …). Les données acquises sont synchronisées et horodatées ce qui garantit une haute précision lors de l'enregistrement et la relecture de ces données. Cette synchronisation permet également de développer des applications orientées fusion de données multi-capteurs.

La structure logicielle GOLD fournit des fonctions de traitement d'image bas niveau qui permettent de prétraiter les données acquises ou effectuer des opérations courantes (comme la stabilisation d'image ou la suppression de distorsion par exemple).

GOLD dispose d'une interface graphique d'utilisateur qui permet d'interagir avec les applications (voir la Figure 24).

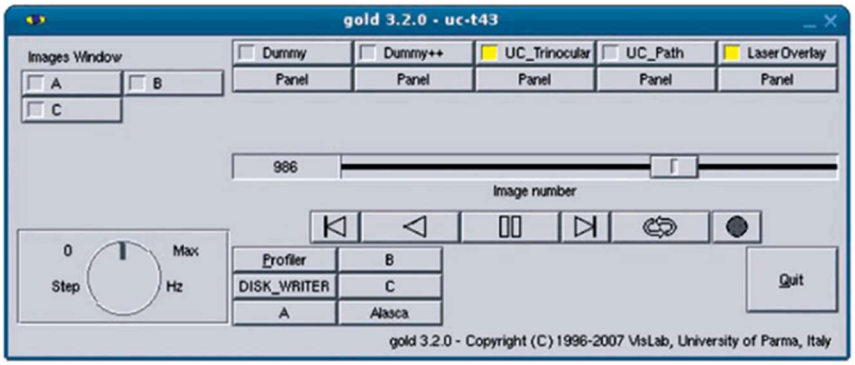

Figure 24 GOLD Framework GUI

L'architecture de GOLD a évolué, ces dernières années, principalement pour permettre aux développeurs de se libérés de la structuration bas niveau des architectures et des détails matériels ainsi que pour être utilisé comme logiciel embarqué grâce à sa flexibilité de fonctionner sur plusieurs plates-formes, même avec des ressources de calcul limitées telles que les processeurs embarqués OMAP3 ou ARM par exemple.

GOLD dispose d'une architecture modulaire (Figure 25) qui garantit un niveau élevé de séparation entre les composants. De ce fait, les sous-systèmes ou les nouvelles applications peuvent facilement être ajoutés.

Les applications développées via GOLD peuvent être déployées pour être portable sur plusieurs plates- formes (x86, x86_64, ARM, OMAP3), systèmes d'exploitation (Linux, Windows et MacOS) et compilateurs (GCC, Intel, Visuel Studio).

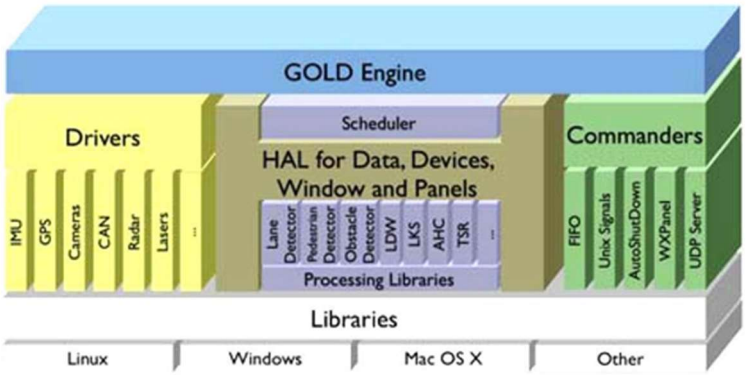

Figure 25 Architecture logiciel de GOLD

7.2. Outils de simulation de la dynamique du véhicule

7.2.1. Spécifications

La dynamique du véhicule est un élément primordial dans le processus de développement et de validation des ADAS. En effet, plusieurs systèmes ADAS (ACC, Park Assist, Lane Keeping,…etc.) ont besoins d'informations sur l'état du véhicule (tel que : la vitesse de déplacement, l'angle au volant, le rapport de vitesse engagé, …etc.). Et elle agit sur certains

actionneurs du véhicule (tel que : le couple moteur, les pédales de frein et d'accélération, le volant, …etc.). Pour cette raison le modèle du véhicule doit prendre en compte tous les éléments et les organes du véhicule (bloc moteur, transmission, boite de vitesse, bloc châssis, …etc.), voir Figure 26.

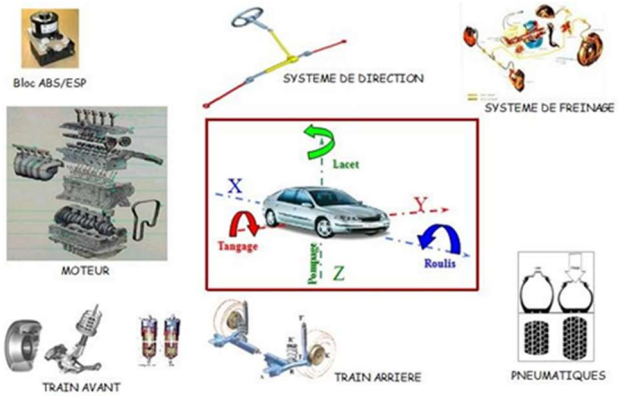

Figure 26 Principaux organes du véhicule

Les principales caractéristiques requises dans ce type d'outils sont résumées dans les points suivants :

- Prise en compte de tous les organes du véhicule ;
- La possibilité et la facilité d'intégration de modèles d'organes ou des sous-systèmes (physiques ou développé dans un tiers logiciel comme Matlab/Simulink par exemple) ;
- Dispose d'une interface graphique permettent d'accéder, de configurer et d'afficher tous les organes et leur paramètres ;
- Possibilité et facilité d'interfaçage avec des logiciels tiers (les simulateurs de conduite et Matlab/Simulink par exemple) ;
- L'outil doit être capable de tourner en temps réel pour des validations SIL ou HIL par exemple ;

7.2.2. Drive (Sate - Italy)

DRIVE est une application logicielle développée avec MATLAB/Simulink pour la simulation de la transmission. Il simule, en particulier, la dynamique de l'ensemble du véhicule en mouvement rectiligne variable pendant un changement de vitesses.

DRIVE fonctionne sous l'environnement MATLAB/Simulink de MathWorks. Il dispose d'une interface graphique conviviale basée sur des boutons et des menus à dérouler. La simulation dynamique de l'ensemble du véhicule est réalisée par la simulation de plusieurs sous-systèmes mutuellement interconnectés, tel que : la dynamique du bloc moteur, le couple du moteur, la dynamique de suspension, la dynamique de la transmission, la dynamique de la carrosserie du véhicule, la dynamique verticale des pneus ...etc.

A noter que la transmission est soigneusement modélisée de l'embrayage aux pneus, y compris la boîte de vitesse (Figure 27). L'interaction entre les pneus et le sol est exprimée par un diagramme non linéaire de force et du glissement, qui tient compte du calcul précis de forces de traction, y compris ceux avec des valeurs élevées de glissement.

Sate a conçue d'autres outils complémentaires comme Bench pour les suspensions, Clutch pour l'embrayage, Condiz pour la climatisation…Ces produits, fondés sur MATLAB/Simulink, peuvent être utilisés individuellement aussi bien que dans un environnement plus large de simulation.

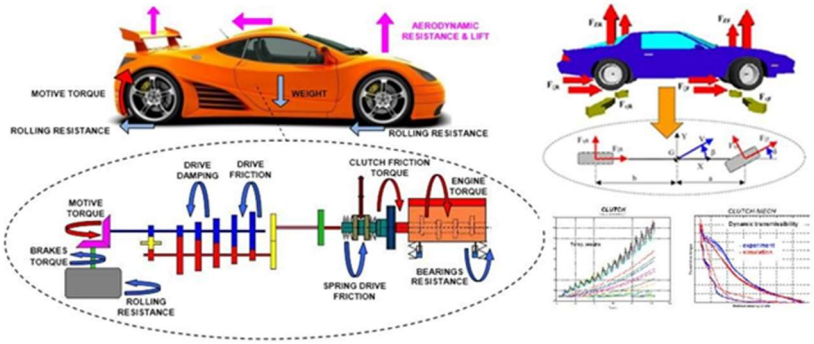

Figure 27 Drive – Simulation de la chaine de transmission

7.2.3. CarSim (Mechanical Simulation Corporation)

CarSim (Figure 28) est un logiciel commercialiser par Mechanical Simulation Corporation qui est un des leaders dans le développement et la distribution de logiciels pour simuler le comportement du véhicule impliquant des interactions entre la dynamique tridimensionnelle de ce dernier, les commandes du conducteur et la géométrie tridimensionnelle de la route.

CarSim simule le comportement dynamique de différents type de véhicules (tourisme, course, utilitaires…). Il permet de fournir en sortie plus de 800 variables ainsi que le tracé et l'analyse de ces dernières. Ces résultats sont exportables vers d'autres logiciels tels que MATLAB, Excel, ainsi que des outils d'optimisation.

Les modèles mathématiques de CarSim garantissent une couverture complète de tous les éléments du véhicule. Ces modèles sont extensibles à l'aide des commandes intégrées à l'outil et qui permettent l'import de modèles (ou sous-modèles) à partir de MATLAB / Simulink, LabVIEW ou développés sous Visual Basic, C. On peut utilisez ces options pour ajouter des contrôleurs ou étendre un détail dans un modèle, d'un sous-système ou d'un composant tels que les pneus, les freins, ... etc.

Les modèles disponibles dans CarSim peuvent fonctionner dix fois plus rapidement que le temps réel sur un ordinateur disposant d'un processeur de 3 GHz de fréquence. CarSim supporte également les outils (logiciels et matériels) temps réel (RT) et permet de réaliser des essais de validation matérielle dans la boucle (HIL).

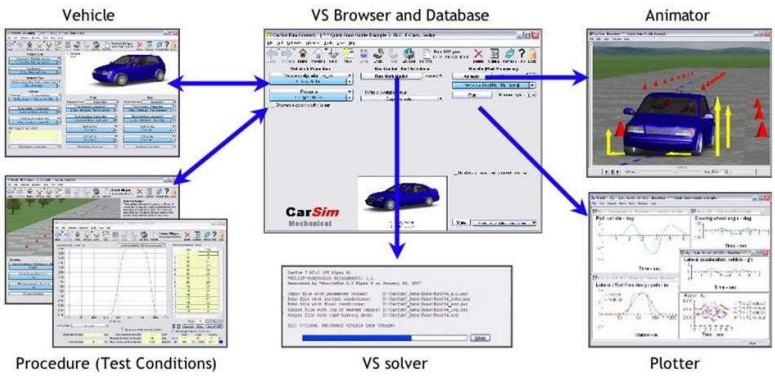

Figure 28 CARSIM

Les modèles mathématiques CarSim sont également intégrables dans les simulateurs de conduite, allant de systèmes peu coûteux basés sur un ordinateur aux simulateurs de conduite. Mechanical Simulation Corporation propose également d'autres outils tels que : BikeSim pour la simulation des motos et des scooters, TruckSim pour simuler les camions ...etc.

7.2.4. veDYNA (TESIS DYNAware)

veDYNA est un environnement de simulation temps réel de la dynamique du véhicule pour des essais de conduite virtuels. veDYNA peut être utilisé pour la conception d'un véhicule sur ordinateur comme il peut être utilisé également pour des tests de validation de composants sur banc de test virtuel ou physique. Le développement des commandes de la dynamique du véhicule (comme le contrôle global du châssis ou du lacet par exemple) est pris en charge à partir du prototypage rapide sur ordinateur et jusqu'aux tests de validation SIL et HIL.

veDYNA est appliquée dans divers domaines du développement du véhicule tels que:

- Études et optimisation des paramètres des composants du véhicule tels que : la transmission, la direction et la suspension ...etc ;
- Études et manipulation des paramètres du confort de conduite ;
- performances de conduite et calcul de la consommation ;
- Etude de la stabilité de conduite pour les véhicules utilitaires ;
- Développement et test des composants mécaniques en bancs de test virtuels et physiques ;
- La simulation en temps réel pour la conception, et la validation des systèmes de contrôle du véhicule tels que : ABS, ESP et ACC dans les phases SIL et HIL ;
- Tests de conduite virtuels dans le simulateur de conduite.

La mise en œuvre et l'évaluation des tests sont supportées par des procédures Matlab préconfigurées, dont les paramètres et les caractéristiques peuvent être modifiés via des interfaces graphiques d'utilisateur. La visualisation des résultats est possible au cours de l'exécution de la simulation avec l'outil d'animation DYNAanimation ou à la fin de la simulation via l'interface graphique d'utilisateur "GUI" de veDYNA (Figure 29 Vedyna).

Figure 29 Vedyna

7.2.5. VDL (Dymola – Dassault Systems)

VDL (Vehicle Dynamics Library) est un outil pour la modélisation, la simulation et l'analyse des véhicules. Ses principales caractéristiques sont les suivantes :

- Environnement complet pour l'analyse du véhicule et de ses sous-systèmes, y compris les routes, les conducteurs et les bancs de test ;
- Fondé sur Modelica® qui est un langage standard pour la modélisation de composants intuitif ;
- Dispose d'une interface Simulink pour la simulation en temps réel sur un ordinateur standard ou sur une cible matérielle.

VDL (Figure 30) dispose d'une architecture orientée objet ouverte avec un accès aux codes source des modèles dans Modelica. La route est décrite en 3D et plusieurs manœuvres de conduite sont disponibles : suivi de courbure, changement de voie, freinage en courbe, slalom, évitement d'obstacle…etc. La simulation en temps réel avec Dymola donne la possibilité d'exécuter les tests HIL sur les plates-formes communes comme dspace, RT-LAB, xPC Target et Cramas.

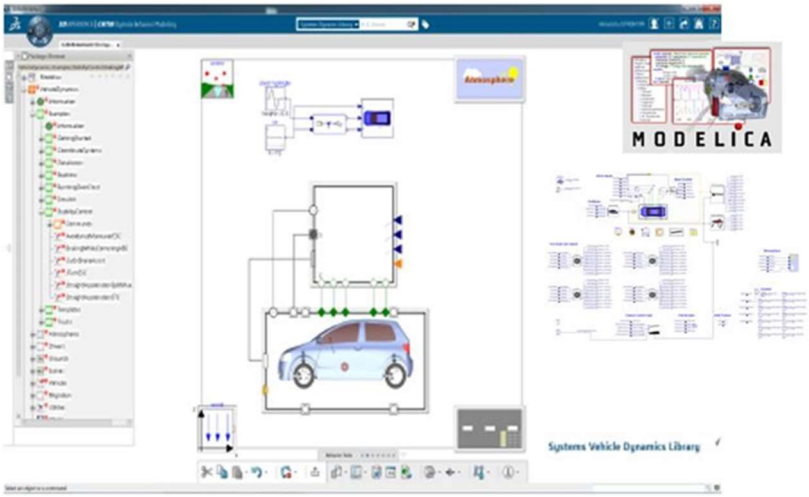

Figure 30 VDL

7.2.6. ASM Vehicle Dynamics Simulation Package (dspace)

ASM Vehicle Dynamics Simulation Package (Figure 31) est un modèle Simulink ouvert pour la simulation temps réel du comportement de la dynamique du véhicule dans un environnement spécifié. Le modèle est complet et indépendant, il peut intervenir dans toutes les phases du processus de développement du système. Il est généralement utilisé sur un simulateur dspace pour les tests de validation HIL (hardware- in-the-loop) des unités de contrôle électronique (ECU - Electronic Control Unit) ou pendant la phase de conception de la loi de commande pour une validation MIL (Model-in-the-loop).

Tous les blocs Simulink dans le modèle sont visibles, il est donc facile d'ajouter ou de remplacer des composants avec des modèles spécifiques (modèle fournisseur par exemple) pour adapter les propriétés du véhicule parfaitement aux besoins et aux exigences du système ou projet. Les routes et les manœuvres de conduite peuvent être créés à l'aide d'outils graphiques mises à disposition et qui permettent une visualisation claire.

Les caractéristiques du modèle physique du véhicule sont représentées par un système multi corps de 24 ddl (degrés de liberté). Il se compose d'une chaîne de traction, d'un moteur, de deux modèles de pneus, d'un modèle de frein hydraulique, d'un système multi-corps non linéaire de la cinématique des suspensions et de l'aérodynamique du véhicule, ainsi que d'un modèle de direction. Un environnement routier, des manœuvres de conduite et un conducteur en boucle ouverte ou fermée sont inclus également. Tous les paramètres des modèles peuvent être modifiés au cours de la simulation. La figure 18 représente l'architecture de l'outil et l'interface graphique d'utilisateur.

ASM Vehicle Dynamics Simulation Package peut être utilisé avec des composants physiques dans un environnement HIL (Hardware-In-the-Loop). Le modèle prend en charge en temps réel la génération du code via Real-Time Workshop (RTW) dans Matlab/Simulink et RTI de dspace pour la simulation en ligne sur une cible dspace temps réel.

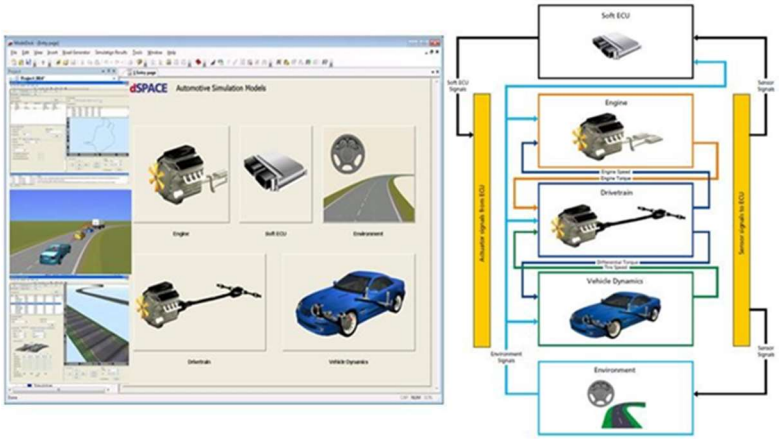

Figure 31 ASM-VDSP et ses composants

7.2.7. LMS Imagine.Lab AMESim (Siemens PLM)

LMS Imagine.Lab AMESim (Figure 25) est un logiciel de simulation pour la modélisation et l'analyse de systèmes 1D multi-domaines. Le logiciel permet de modéliser et d'analyser des systèmes multi-domaines ainsi que de valider leurs performances. Les composants du modèle sont décrits en utilisant des modèles analytiques validés qui représentent le comportement réel hydraulique, pneumatique, électrique ou mécanique du système.

Pour créer un modèle de simulation d'un système, un ensemble de bibliothèques (31 bibliothèques standard qui couvre tous les domaines physiques) contenant des composants prédéfinis pour différents domaines de la physique peut être utilisé.

LMS Imagine.Lab AMESim est fourni avec deux bibliothèques par défaut : mécanique et signal. Les autres bibliothèques doivent être achetées séparément. Le vaste ensemble de bibliothèques disponibles permet aux concepteurs de mettre au point des modèles de systèmes complexes qui couvrent plusieurs domaines physiques simultanément (hydraulique, pneumatique, mécanique, électrique, …). En ce sens, ces bibliothèques accélèrent la création de modèles et permettent de libérer du temps pour optimiser la conception.

La modélisation d'un système sous AMESim est réalisée en quatre étapes :

- Mode sketch : on assemble les différents composants ;
- Mode sous-modèle : on choisit le sous-modèle physique associé à chaque composant ;
- Mode paramètre : on choisit les paramètres pour les différents sous-modèles ;

- Mode simulation : on exécute la simulation.

Lors du passage du mode sous-modèle au mode paramètre, le sous modèle est compilé. Par conséquent, Il est nécessaire de disposer d'un compilateur. AMESim fonctionne avec le compilateur libre gcc (fourni avec AMESim), avec le compilateur de Microsoft Visual C++ et avec le compilateur Intel.

AMESim fonctionne sur la majorité des plateformes de type UNIX (dont Linux) et sous Windows.

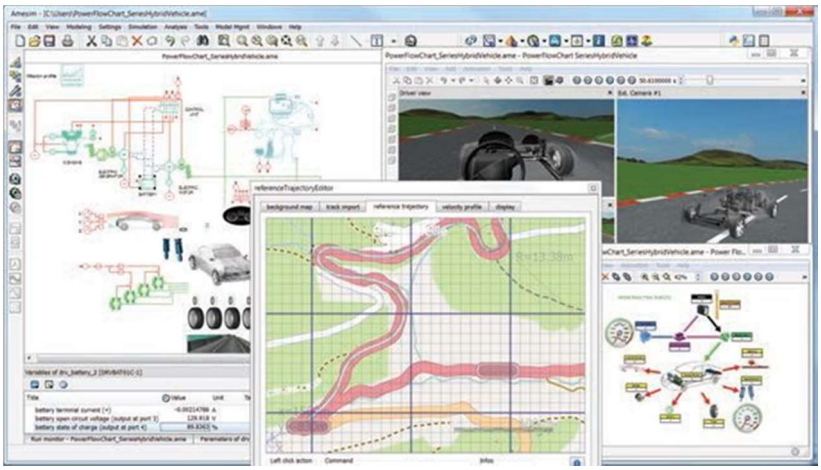

Figure 32 LMS Imagine.Lab AMESim

La suite logicielle LMS Imagine.Lab AMESim est composée de différentes applications :

- AMESim : Produit principal pour la modélisation et la simulation de systèmes dynamiques;
- AMECustom : Personnalisation de modèles et protection de la propriété intellectuelle ;
- AMERun : Simulation de modèles déjà existants ;
- AMESet : Développement de nouveaux composants.

7.3. Outils de Développement des IHM

7.3.1. Spécifications

Jusqu'à présent, les tableaux de bord de nos voitures étaient essentiellement analogiques. Cependant les choses sont en train de changer, notamment chez certains constructeurs haut de gamme qui se servent de leurs tableaux de bord comme argument de vente pour démontrer leurs capacités d'innovation (Figure 33).

Cette évolution dans conception de tableau de bord est accompagnée par une démocratisation des systèmes d'aide à la conduite (ACC, LKA, ParkAssist, ... etc.) et de l'émergence de nouvelles technologies connectées avec la promesse du véhicule autonome dans quelques années.

Tous ces éléments ont augmentés les enjeux pour le design d'interface utilisateur (nouvelles exigences, nouvelles contraintes, nouvelles interactions…) et pour répondre à toutes les exigences liées à ces enjeux de nouveaux outils de conception d'interfaces utilisateurs ont vu le jour. Les principales caractéristiques requises dans ce type d'outils sont suivants :

- Dispose d'une interface simple, dynamique et intuitive ;
- Dispose d'un large panel de formes et de couleurs ;
- Interfaçable avec des logiciels tiers ;
- Prend en compte la majorité des standards de communication (USB, Ethernet, CAN, LAN, ….) ;
- Capacité d'intégrer des modèles issus de logiciels tiers ;
- Portabilité sur tout type de plateforme logicielle et matérielle ;

Figure 33 Exemple d'IHM

7.3.2. Altia (altia)

Altia offre une suite d'outils de simulation graphique permettant aux ingénieurs de créer des interfaces graphiques réalistes pour leurs simulations. Cette suite est construite autour de quatre logiciels : PhotoProto, FlowProto, Altia Design et DeepScreen.

Le processus de développement d'une application d'interface graphique d'utilisateur est devisé en trois étapes (Figure 34). Dans l'étape une, les outils PhotoProto et FlowProto permettent d'étudier le besoin, définir les cas d'usage et de développer le concept de l'interface utilisateur ou l'interface home machine à réaliser. Dans l'étape deux, Altia Design permet de construire cette interface en se basant sur les cas d'usage et le concept développé précédemment. DeepScreen quant à lui, interviens dans l'étape trois et permet de générer le code de l'application.

Dans ce qui suit nous allons décrire brièvement chaque outil composant la suite altia.

Figure 34 Processus de développement d'une IHM

PhotoProto

PhotoProto permet de convertir un projet statique Photoshop en un prototype itératif avec seulement quelques clics souris. Il parcourt le fichier Photoshop et il exporte toutes les images et leurs références dans un fichier que PhotoProto affiche.

PhotoProto permet également :

- De transformer un graphique Photoshop statique en un prototype itératif ;
- De créer des boutons, curseurs, afficheurs, écrans, …etc. ;
- D'exporter automatiquement des illustrations (au format PNG) ;
- De transmettre des prototypes aux collègues ou aux clients pour les tester sur PC ;
- La lecture des fichiers mp3, avi et des modèles DirectX/OpenGL ;
- De préparer en avance de phase le développement sous Altia Design (import des sorties PhotoProto directement dans Altia Design).

FlowProto

FlowProto est ajout à Microsoft Visio, il permet :

- De créer des modèles de machine d'état simple dans Microsoft Visio ;
- De convertir la description du comportement créé dans Microsoft Visio en code C déployable ;
- De développer un code de commande qui donne à la conception Altia le comportement défini dans Visio ;
- De publier les organigrammes comme des fichiers XML qui peuvent être importés dans Altia Design pour le développement et génération de code.

Altia Design

Altia Design est l'outil principal de la suite, il permet d'obtenir une interface utilisateur détaillée, fonctionnellement complète, complètement intégrée et déployable. Altia Design, qu'est un outil de développement graphique, permet également :

- De construire des prototypes et des interfaces graphiques de qualité professionnelle sans codage.
- De décrire l'animation et simuler le comportement sans programmation ;
- De livrer un modèle GUI complet aux clients, partenaires, … etc. ;

- D'intégrer l'interface utilisateur aux modèles du système pour une validation complète de ce dernier ;
- De préparer l'interface graphique pour la génération du code via DeepScreen.

DeepScreen

Altia DeepScreen est un générateur de code graphique qui convertit les prototypes graphiques développés sous Altia Design en un code déployable. Pour générer le code, il suffit de sélectionner les objets qui représentent la partie d'affichage de l'interface et cliquez sur le bouton "Générer le code", un code C est généré en quelques secondes.

Le code généré est supporté par tous les systèmes d'exploitation (Windows, Linux, Unix, …).

7.3.3. EB GUIDE (Elektrobit)

EB GUIDE (Figure 35) est l'un des logiciels les plus utilisés pour la conception d'IHM dans l'automobile. Il permet de modéliser et de simuler une IHM sur un ordinateur ainsi que déployer facilement cette IHM sur une cible.

L'intégration des graphiques, des interfaces vidéo, des interfaces vocales y compris la reconnaissance vocale, des animations des effets ainsi que des objets 3D sont pris en charge dans EB GUIDE ce qui permet d'avoir des interfaces graphiques de haute qualité.

Les caractéristiques d'EB GUIDE sont les suivantes :

- Un outil tout-en-un qui permet la spécification, la modélisation, le prototypage et la production en série des IHM ;
- Fournit un outil pour la modélisation multimodale des graphiques et des interfaces vocales d'utilisateur ;
- Supporte le développement des IHM avec des graphiques 3D, des effets et des animations ;
- Permet d'utiliser la dernière technologie vocale ;
- Permet d'intégrer le contenu HTML 5 dans l'IHM native ;
- Permet d'évaluer l'IHM à tous les stades de développement ;
- ASIL / ISO 26262 expertise.

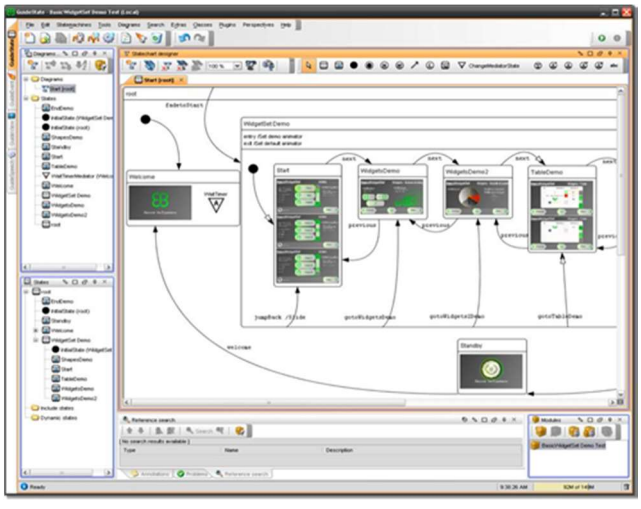

Figure 35 EB Guide

7.3.4. NVIDIA DRIVE DESIGN (NVidia)

NVIDIA DRIVE Design (Figure 36) est un outil de conception IHM permettant de développer des combinés d'instruments numériques et des systèmes d'info-divertissement pour les tableaux de bords. Conçu par NVIDIA, ce logiciel est utilisé par les constructeurs et les fournisseurs directs de l'industrie automobile pour développer de nouveaux concepts pour l'évaluation, l'étude de marché, le test et la production finale.

NVIDIA DRIVE Design inclut des capacités natives de rendu 2D/3D, d'imagerie et d'animation en haute résolution, des fonctionnalités avancées pour la gestion des sources de lumière et des caméras ainsi que des fonctions interactives qui permettent de créer des applications de haut niveau graphique. Enfin, grâce à la compatibilité MDL (Material Definition Language), des matériaux complexes comme la fibre de carbone, les métaux brossés, le cuir cousu et le verre peuvent être affichés en temps réel et de manière photo-réaliste.

NVIDIA DRIVE Design dispose d'une technologie avancée de traitement des signaux audio qui garantit une reconnaissance vocale dynamique et performante. Par conséquent, la précision des commandes vocales est renforcée et la concentration du pilote est préservée.

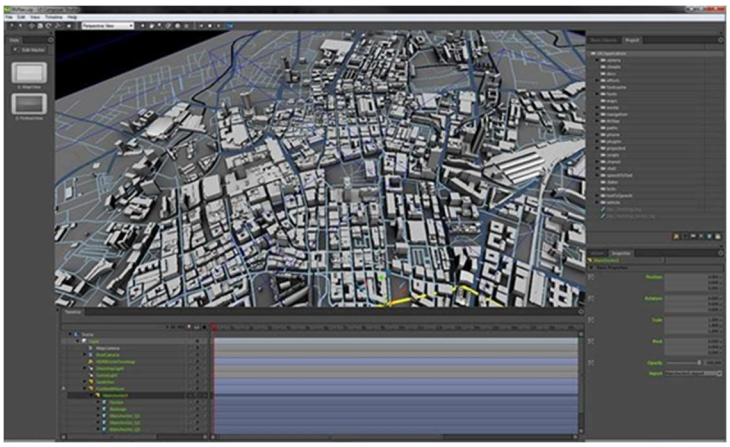

Figure 36 NVIDIA DRIVE Design

7.3.5. Qt (Qt - digia)

Qt est un Framework (ou un ensemble d'outils et de bibliothèques) destiné à la création des applications avec des composants graphiques (GUI). Son développement a débuté en 1994 et sa commercialisation en 1995-1996 par la firme Trolltech, cette dernière a été rachetée fin 2008 par Nokia. En 2011 Qt c'est fait racheter par Digia. Qt est complètement orienté objet est codé en langage C++. Toutefois, il existe des « bindings» pour d'autres langages tels que Perl, Python, C# et Java. Qt est portable et disponible sur plusieurs plateformes : MS/Windows, Unix/X11, Macintosh, Embedded OS (Figure 37).

Qt est disponible sous plusieurs licences :

- Qt Enterprise Edition et Qt Professional Edition : destinés aux développements à buts commerciaux.
- Qt Open Source Edition : cette version est distribuée avec la Q Public License et la GNU General Public License pour les développements libres.

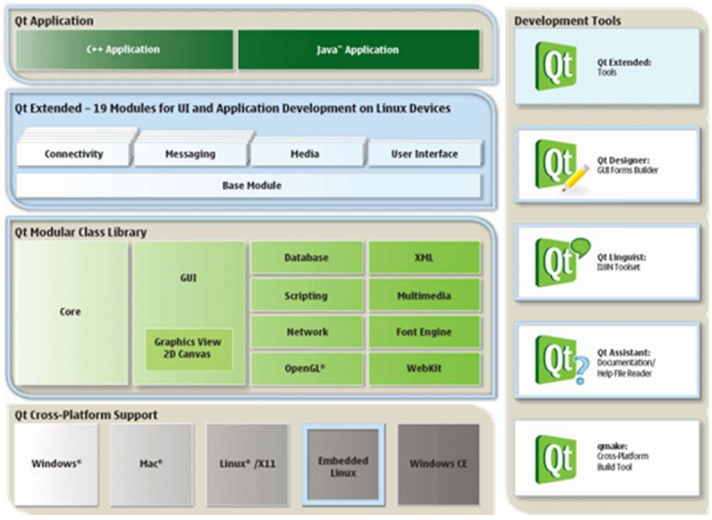

Figure 37 Architecture des bibliothèques Qt

En plus du développement des interfaces graphiques, Qt est constitué d'un ensemble de modules (Figure 37) qui lui permettent de disposer des fonctionnalités suivantes :

- Module GUI : module qui permet la création de fenêtres.
- Module OpenGL : Qt peut ouvrir une fenêtre contenant de la 3D gérée par OpenGL.
- Module de dessin : module qui permet de dessiner dans les fenêtres (en 2D), le module de dessin est très complet !
- Module réseau : Qt fournit un ensemble d'outils pour accéder au réseau, que ce soit pour créer un logiciel de Chat, un client FTP, un lecteur de flux RSS…etc.
- Module SVG : Qt permet de créer des images et des animations vectorielles, de la même manière que Flash.
- Module de script : Qt prend en charge le JavaScript ce qui permet de réutiliser ou d'ajouter des fonctionnalités sous forme de plugins par exemple.
- Module XML : pour l'échanger des données à partir de fichiers structurés à l'aide de balises, comme le XHTML par exemple.
- Module SQL : permet d'accéder aux bases de données (MySQL, Oracle, PostgreSQL…).
- Qt est utilisé par de nombreuses entreprises pour le développement de différentes applications ou GUI, à titre d'exemple on peut citer : Adobe, Archos, Boeing, Google, Skype, la NASA… etc.

Qt Creator

Qt Creator (Figure 38) est un environnement de développement intégré (IDE) multiplateforme faisant partie du Framework Qt. Il est orienté pour la programmation en C++ et optimisé pour compiler des projets utilisant Qt.

Il intègre les modules et les librairies cité ci-dessus, un débogueur ainsi que la documentation Qt. L'éditeur de texte intégré permet l'auto-complétion ainsi que la coloration syntaxique. Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows.

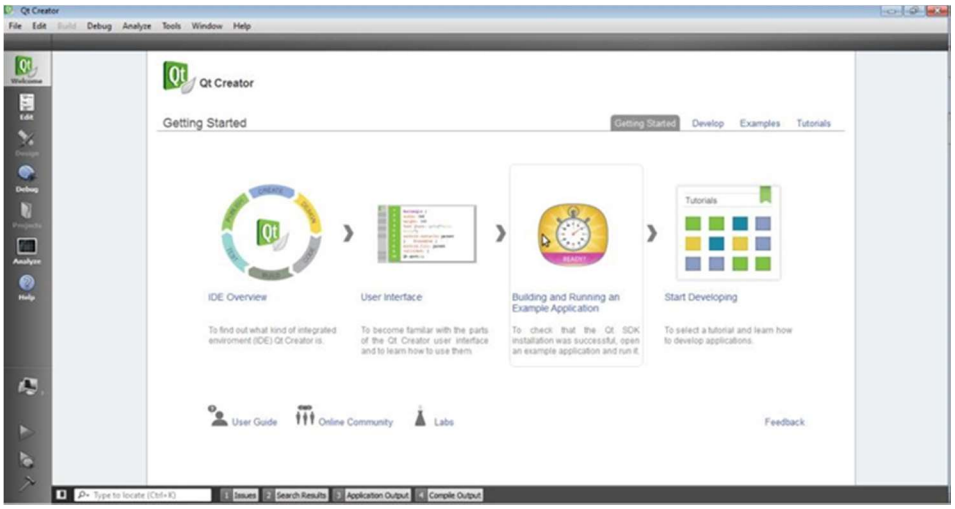

Figure 38 Qt Creator

Pour résumer, les principales caractéristiques de Qt sont les suivantes :

- Les applications Qt possèdent un aspect et un comportement natif sur toutes les plateformes supportées ;
- Pour le portage entre les systèmes les plus connus (Linux, Windows, Mac OS X, . . . etc.) une simple recompilation est nécessaire ;
- Le portage peut aussi être effectué pour des environnements embarqués ;
- Qt intègre le support de la 3D grâce à des composants OpenGL ;
- Qt utilise l'Unicode et possède des composants dédiés à l'internationalisation (QString, Qt Linguist) ;
- Qt permet l'utilisation de SGBD, indépendamment de la plateforme d'utilisation. Il est compatible notamment avec : Oracle, SQLite, PostgreSQL, MySQL, . . . ;
- Qt inclus un grand nombre de classes spécialisées pour un domaine ; il permet notamment la gestion directe du XML grâce à des parseurs SAX et DOM ;
- Qt inclus aussi des classes dédiées aux fonctionnalités réseaux et supporte les protocoles standard ;
- Pour faciliter la création des applications, un outil Qt Designer est mis à disposition pour permettre de créer rapidement les interfaces élémentaires ;
- Pour étendre Qt à de nouveaux composants et fonctionnalités, Qt propose un système d'extension (Meta Object System) et un compilateur (Meta Object Compiler (MOC)) qui permettent d'augmenter la puissance de la bibliothèque ;
- Pour faciliter la création des Makefiles et autres fichiers de compilation, Qt propose l'outil qmake qui génère tous les fichiers nécessaires à compilation «propre».

A noter que la compagnie Qt et ses partenaires (KDAB et Pelagicore) ont annoncés le lancement de la « Qt Automotive Suite », un nouveau produit qui répond aux besoins des équipementiers et des sous-traitants automobiles en terme de développement de systèmes d'info-divertissement embarquée dans les véhicules (In-Vehicle Infotainment - IVI). La Qt Automotive Suite a été présentée au Qt World Summit 2015 (5-7 octobre Berlin).

7.4. Outils de Cartographie et système de navigation

7.4.1. Spécifications

Dans le domaine des systèmes de transport intelligents, de nombreuses aides à la conduite nécessitent une localisation GPS précise du véhicule sur la route et une carte numérique de navigation précise et fiable. Cette localisation va permettre au système de navigation d'alimenter les systèmes d'aide à la conduite par des informations pertinentes (type de chaussée, vitesse limite,… etc.) aidant ces derniers à prendre la bonne décision ou à fournir la meilleure assistance.

Le but également est de fournir au conducteur des informations pertinentes liées à l'itinéraire, à la signalisation, et au trafic routier. L'intégration des outils logiciels qui permettent de simuler ces systèmes de navigation avec leurs cartographies dans les plateformes de développement et de validation des ADAS est un élément essentiel dans le processus de développement de ces ADAS. Les principales caractéristiques requises dans ce type d'outils sont suivants :

- Richesse de la carte en termes d'attributs ;
- Fiabilité du contenu (données précises, à jour, … etc.) ;
- Portabilité de la carte sur différents support logiciels ou matériel ;
- Interfaçable avec tout type de logiciels ou matériels,
- Prise en charge du protocole ADASISv2 ;

7.4.2. SIVNav SDK (BeNomad)

SIVNav SDK est un environnement de développement pour la création rapide et efficace d'applications de navigation professionnelles. Il fournit aux développeurs une gamme d'interfaces de programmation pour relier les fonctions de navigation, de décodage d'adresses et de rendu cartographique afin de créer leurs propres applications de navigation.

Les principales caractéristiques de SIVNav SDK sont les suivantes :

- Géocodage : adresses postales, points d'intérêt, codes postaux, recherche par zone géographique ;
- Routage : algorithmes de calcul d'itinéraire simples ou multiples, isochrones ;
- Affichage de cartes interactives ;
- Affichage aérien et images satellites ;
- Enrichissement cartographique avec des données cartographiques propriétaires ;
- Cartes et données multilingues ;
- La couverture de 78 pays sur 6 continents ;
- Langages de développement : C++ et C # ;
- Architecture modulaire (Figure 39) ;
- Systèmes d'exploitation : Android, Windows NT, 2000, XP, Vista, Seven, Windows Mobile, Windows CE, iPhone, et Linux.

BeNomad proposes également des outils complémentaires qui cibles des domaines d'utilisation spécifiques (BeNav, BeMap, GeoSDK, …).

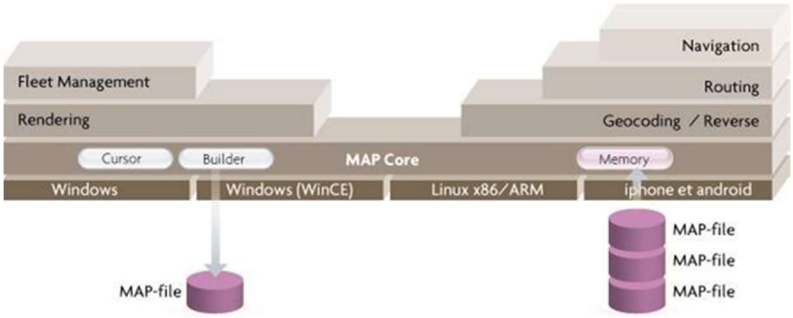

Figure 39 Architecture de SIVNav SDK

7.4.3. MacMap et Map Theme (MacMap)

MacMap (Figure 40) et MapTheme permet d'élaborer des cartes avec précision et souplesse. Chaque élément géographique dessiné est intégré à la base de données sous la forme d'un objet facilement manipulable à la souris : un clic dans la carte permet de sélectionner un élément, un double-clic d'ouvrir la fiche de l'objet pointé et de consulter ses données. Une présentation tableur de ces données est également disponible.

MacMap offre de nombreuses possibilités de représentation des objets géographiques en fonction des données qui leur sont attachées. Il intègre en standard de nombreuses fonctions de visualisation (Trame, Symbole, vue Proportionnelle,...) et les outils permettant de réaliser très simplement des travaux complexes comme les plans de villes : noms affichés le long des segmentations, rotations de symboles, choix des formes d'extrémité et d'angularité, détourage des textes... etc.

Chaque couche d'information peut être représentée autant de fois que nécessaire : une rue peut être affichée sous forme de trait, puis son nom sous forme de texte, en allant chercher les données utiles (nom, justification, interligne, ...) dans la base de données. MacMap® vous permet également de générer des rasters aux formats JPEG, PNG, et TIFF.

MacMap dispose en standard des traducteurs appropriés pour échanger des plans aux formats MIF-MID, DXF, ShapeFile, texte, …etc.

MacMap intègre aussi des fonctions de dessin vectoriel avancé (saisie, retouche, ...) qui permettent de constituer des bases de données propres à chaque utilisateur. La plupart de ces fonctions sont réunies dans des Packs Dessin et Transformations, et offrent de multiples aides à la saisie (rabattement des points saisis sur les objets existant, corrections topologiques de réseaux, génération de surfaces à partir de linéaires,...)

MacMap dispose d'un pack itinéraire qui regroupe des fonctions, ces dernières peuvent être utilisées dans différents contexte : calcul de temps d'accès et distance par la route, calculs d'isochrones, génération d'objets "trajets"...

Figure 40 MacMap

7.4.4. OpenStreetMap (OpenStreetMap)

OpenStreetMap (OSM, Figure 41) est un projet qui a pour but de constituer une base de données géographique libre du monde, en utilisant le système GPS et d'autres données libres.

À la manière de Wikipédia, tous les internautes naviguant dans le web peuvent contribuer à la création et à la numérisation de cartes. Des éditeurs permettent de réaliser en ligne des cartes en se basant sur un fond d'image satellitaire. Cependant, ces images satellitaires ne couvrent pas toujours en haute résolution l'ensemble du globe. C'est pourquoi il est possible d'introduire des données provenant de récepteurs GPS. Il suffit de réaliser un itinéraire et de positionner le récepteur GPS en mode enregistrement, puis de le restituer sur le serveur de données d'OpenStreetMap situé au Royaume-Uni et géré par la fondation OpenStreetMap.

Les points d'intérêt (POI, en anglais « point of interest »), c'est-à-dire, toutes les mentions utiles (noms, largeur, nature du revêtement, sens uniques, vitesse limite, barrières, pistes cyclables, … etc.) sont notés, soit en les écrivant, soit en les photographiant, soit en les décrivant sur un appareil d'enregistrement audio.

Les enregistrements de données GPS peuvent être rendus publics par l'intermédiaire du site d'OSM. Cela a pour avantage de les rendre visibles dans les outils d'édition des cartes. La carte principale est une carte routière comprenant des éléments figurés de manière plate, mais une carte du relief avec les courbes de niveaux est également disponible.

Les outils disponibles permettent d'utiliser les données d'OpenStreetMap pour :

- Alimenter la carte mondiale et en extraire certaines parties pour son propre usage (du globe complet à la carte locale) ;
- Créer des cartes interactives ou statiques ;
- Créer des cartes pour de nombreux terminaux GPS ;
- Alimenter certains SIG.

Deux principaux types d'outils informatiques sont utilisables : les logiciels d'édition de rendu de cartes (Mapnik, Osmarender, …) qui servent à élaborer les couches de la carte mondiale principale et de ses dérivés et les éditeurs de carte (iD, Potlatch, JOSM, Maperitive, Merkaartor, …) qui servent à modifier les couches existantes.

Ces logiciels sont tous sous licence libre et multiplateformes (GNU/Linux, MacOS X et Windows).

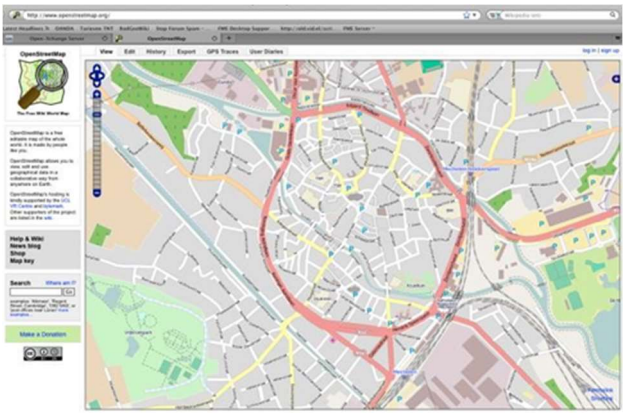

Figure 41 OpenStreetMap

7.4.5. ADASRP – ADAS Research Platform (HERE)

ADASRP (Figure 42) est une structure logicielle pour le prototypage des ADAS et des systèmes de navigation. Les éléments de base de cette structure sont la navigation (affichage de la cartographie, positionnement du véhicule, géocodage et routage) et la prédiction des chemins possible du véhicule (horizon électronique).

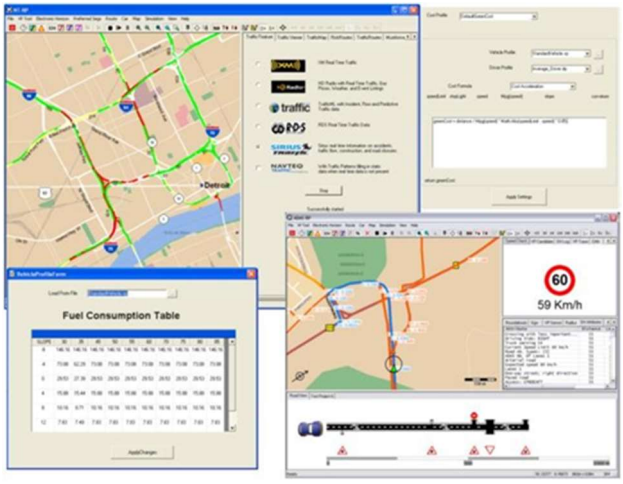

Figure 42 ADASRP

Les principales caractéristiques d'ADASRP sont les suivantes :

- Différentes bibliothèques sont disponibles pour permettre aux applications d'accéder facilement aux données, différents format son supportés ;
- Les données capteurs sont gérées par un module intégré à ADASRP. Toutefois, ADASRP peut également utiliser les données provenant d'un logiciel tiers d'acquisition des données capteurs ;
- Un outil de positionnement du véhicule sur la carte est intégré à ADASRP ;
- ADASRP permet de générer l'horizon électronique(*) du véhicule et de l'afficher ;
- ADASRP dispose d'un module de configuration qui permet de personnaliser l'horizon électronique généré ;
- ADASRP dispose d'une interface graphique (Figure 43) qui permet l'affichage des données capteurs, du positionnement du véhicule, l'horizon électronique, … etc. les applications ADAS peuvent être intégrer dans cette interface graphique (Figure 42).

NAVTEQ (HERE) propose également un composant physique sous le nom MPE (Map and Positioning Engine – Figure 44) qui permet de fournir l'horizon électronique via le bus CAN.

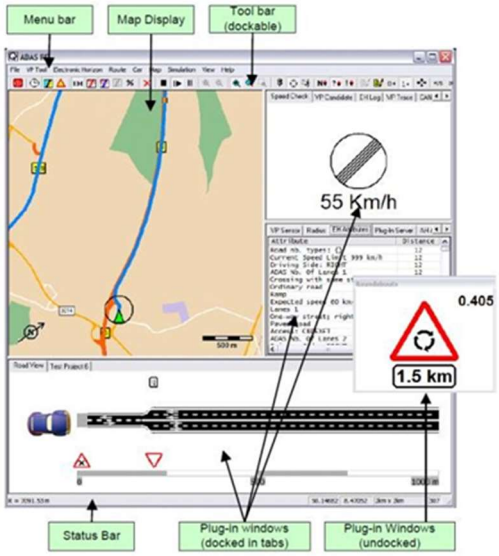

Figure 43 ADASRP GUI

Figure 44 MPE

7.5. Outils de Simulation de capteurs, de conduite et d'environnement routier

7.5.1. Spécifications

On compte plusieurs simulateurs à travers le monde développés dans les secteurs académiques, industriels et commerciaux. Dans la suite, nous présentons les simulateurs « les plus connus » avec leurs spécificités.

Les principales caractéristiques requises dans ce type d'outils sont suivants :

- Présence d'outils et de composants permettent de construire l'environnement routier :
  - Segments de routes prédéfinis : ligne droite, virage (arc de cercle, clothoïde, forme de Bézier, snake,…), trottoir, chicane, Voie d'insertion, voie de sortie, …etc.;

  - Librairie de marquages routiers (associé à la librairie de segment de route) afin de délimiter les voies, d'indiquer le sens de circulation, de marquer les passages piétons et les places de stationnement, …etc. ;
  - d'objets sous la forme de bâtiments, de maisons, d'arrêts de bus, de parkings privé, de stations d'essence, …etc. ;
  - Les panneaux de signalisation et du trafic routiers sont également indispensables;
- Doit permettre la construction automatique de l'environnement routier par import de données cartographiques (OpenStreetMap, here «Navteq», GoogleMaps, … etc.) ;
- Dispose d'un large panel d'objets mobiles (tout type de véhicule, piétons, animaux,…etc.);
- Dispose d'un large panel de modèles capteurs (caméra, GPS, radar, lidar, ultrason, odomètre,…etc.) ;
- Dispose d'un modèle dynamique du véhicule ;
- Interfaçable avec des logiciels tiers (Matlab/Simulink, AMESim, … etc.) ;
- Intégrable dans une configuration temps réel ;
- Capacité d'intégration de modèle ou de composant extérieur ;
- Accessibilité à toutes les données et les paramètres de l'environnement, des capteurs et du véhicule.

7.5.2. Pro SiVIC (ESI)

SiVIC (Figure 45) a d'abord été développé dans le but de fournir des données de simulation issues de modèles de capteurs bruitées et imparfaits assez similaires à celles fournies par des capteurs réels embarqués dans des conditions réelles (principalement les capteurs optiques). Au fil du temps, et grâce à son architecture modulaire, de nouvelles fonctionnalités ont été ajoutées pour fournir d'autres types de fonctionnalités tels que : la gestion des événements, les échanges de ressources ... etc.

Actuellement, SiVIC est capable de simuler la dynamique du véhicule et intègre une variété de capteurs (caméras, GPS, odomètre, lidar, radar, ...). Son objectif est de reproduire une situation aussi réaliste que possible. Les données de simulations générées peuvent être enregistrées via RTMaps (logiciel pour d'acquisition et l'exploitation des données).

RTMaps gère l'enregistrement et la synchronisation de toutes les données en provenance des capteurs SiVIC (caméras, GPS, Lidar, Centrale inertielle, odomètres, ...) ce qui permet de rejouer les scénarii enregistrés. Afin d'anticiper les évolutions des futures systèmes d'aide à la conduite, SiVIC intègre déjà de nouveaux types de capteurs tels que le Laser multifaisceaux ou les capteurs qui émule la communication véhicule- véhicule et véhicule-infrastructure (V2V et V2I).

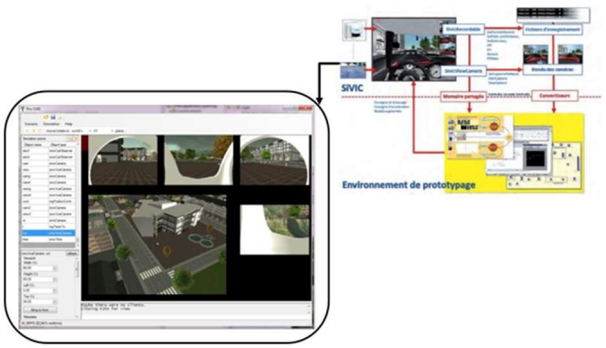

Figure 45 SiVIC dans E'MOTIVE

7.5.3. PreScan (TNO - TASS)

PreScan (Figure 46) est un simulateur d'environnement et de trafic routier développé par TNO et commercialisé par TASS. Il permet de construire des scénarii de simulation dans une interface graphique d'utilisateur (GUI) et d'exécuter ces scénarii afin de valider les systèmes en utilisant MATLAB/Simulink. A cet effet, des modèles de capteurs (Radar, Laser, GPS, et de communications V2V ou V2I) ainsi qu'un modèle dynamique de véhicule sont intégrés dans PreScan. Un modèle Matlab/Simulink représentant la dynamique du véhicule ainsi que des API dédiés aux capteurs permettent de récupérer toutes les données sur Matlab/Simulink.

PreScan est intégralement interfacé avec MATLAB/Simulink dans une configuration de cosimulation ce qui permet de développer et de valider la partie fonctionnelle des systèmes en boucle fermée. PreScan est interfacé et peut fonctionner, également, en mode esclave avec le logiciel ControlDesk de dSPACE (Figure 47). PreScan peut intégrer des dynamiques de véhicule issues des environnements logiciels : CarSim, veDYNA et ASM Vehicle Dynamics.

Figure 46 PreScan

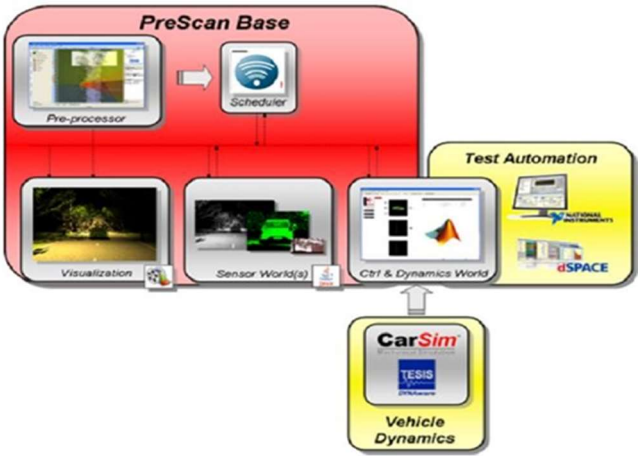

Figure 47 Interfaçage PreScan-CarSim-dSPACE

7.5.4. CarMaker (IPG)

La plateforme CarMaker est un environnement de simulation de conduite virtuelle à part entière. Il propose une large gamme d'applications qui vont des opérations de test fonctionnelles hors temps réel (MIL) à la validation matérielle avec des contraintes temps réel (HIL). En effet, CarMaker a été conçu pour soutenir le processus de développement de systèmes ou de composants du stade conceptuel aux essais sur prototype matériel. Par conséquent, la suite CarMaker est composé de deux éléments principaux, l'interface Toolbox CarMaker (CIT - CarMaker Interface Toolbox) et l'environnement virtuel du véhicule (VVE-Virtual Vehicle Environment).

L'interface Toolbox CarMaker contient une collection d'outils pour le contrôle, le paramétrage, l'analyse et la visualisation de la simulation ainsi que la gestion des données et des fichiers.

L'environnement virtuel du véhicule VVE intègre un modèle véhicule avec tous ses composants tels que le moteur, les pneus, le châssis, les suspensions, le système de direction, le système de frein, la transmission et l'aérodynamique ainsi que la route, le conducteur et les conditions météorologiques (vent, pluie, neige, ...). Les modèles des différents organes du véhicule sont groupés en sous-ensembles qui peuvent être modifiés ou remplacés par des modèles internes développés sous Simulink ou en langage C.

L'intégration complète de CarMaker (Figure 48) dans Matlab/Simulink fait de lui une plate-forme d'intégration et de validation ouverte pour la modélisation et la simulation pour toutes les phases du cycle en V (Model Based Design, Model Based Optimization, Model Based Validation) pour des développements qui concerne la dynamique du véhicule ou les systèmes d'aide à la conduite.

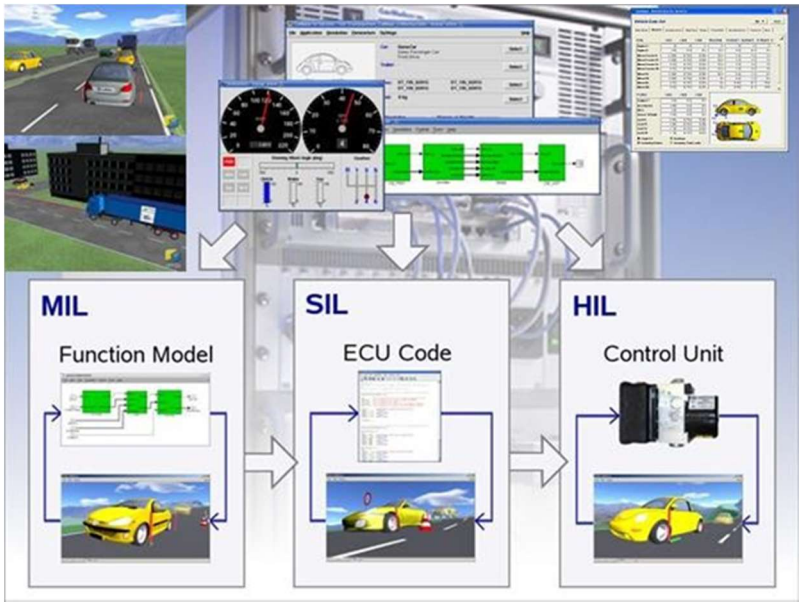

Figure 48 CarMaker

7.5.5. Virtual Test Drive - VTD (VIRES)

VIRES Virtual Test Drive (VTD) est un outil qui regroupe une suite logicielle pour les applications de conduite en simulation (Figure 49). Elle dispose d'outils pour la génération de route, génération de scénarii, génération d'image, simulation de trafic, simulation de son et simulation de contrôle.

VIRES Virtual Test Drive dispose d'une interface ouverte pour des composants tiers et pour des modules ou plug-in avec des API tiers.

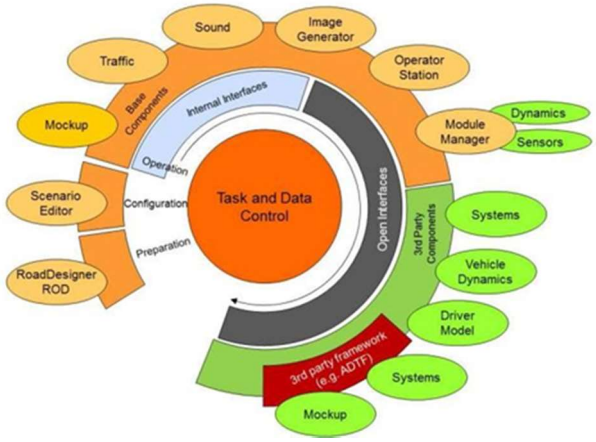

Figure 49 Structuration de la suite VIRES VTD

VTD offre une architecture unifiée pour fonctionner dans des environnements hétérogènes de complexité variable. Il peut fonctionner sur des plateformes simples type ordinateur de bureau pour des tests Model- In-the-Loop par exemple, comme il peut fonctionner sur des plateformes d'environnement complexe pour des tests Hardware-In-the-Loop.

VDT répond aux exigences temps réel pour des simulations interactives avec un conducteur dans la boucle ainsi que des composants physique du véhicule. Il peut tourner plus vite que le temps réel dans le cadre d'une simulation synchrone pour simuler un grand nombre de tests dans un court laps de temps.

VTD offre les fonctionnalités de base pour une gestion robuste des données de tous les composants de l'environnement virtuel. L'utilisateur peut utiliser les composants VTD pour la simulation du trafic, des scénarii, de la dynamique du véhicule, … etc.

VTD-DEV fournit un environnement de développement qui permet à l'utilisateur d'interagir avec les données à haute fréquence et pour contrôler l'ensemble de la simulation. De nouvelles commandes pour le contrôle des composants tiers peuvent être introduites dans VTD.

La liste des outils disponible dans VDT est la suivante :

- ROD - Road Designer : ROD est l'éditeur interactif du réseau routier. Il supporte différents formats d'import / export des données routière ce qui permet la construction des réseaux routiers basés sur des données réelles. Il est également compatible avec les règles de construction de routes et il dispose de vastes bibliothèques d'objets 3D, textures, … etc. (Figure 50.a) ;

- v-SCENARIO et v-TRAFFIC : fournissent un éditeur graphique interactif pour la définition et le suivi des scénarios du trafic pour la simulation sophistiqué du trafic routier (Figure 50.b) ;
- v-IG - Image Generator : v-IG est un générateur d'images temps réel. Il permet une synchronisation multicanal des applications et prend en compte un grand nombre d'effets spéciaux (ombres/soleil, réflexions sur des surfaces humides de la rue, rendu véhicule de haute qualité, pluie/neige/brouillard, module piétonne, module infrarouge, génération de flux vidéo, …), Figure 50.c ;
- v-IOS fournit une interface graphique utilisateur principale pour la gestion de la simulation;
- v-TaskControl : gère la tâche de contrôle centrale de la simulation et agit sur tous les flux de données. Il permet d'intégrer les modules tiers, gère le couplage de plusieurs simulateurs et pilote les états de toutes les entités de simulation ;
- Un module de son gère l'acoustique requis des véhicules pour une immersion de l'utilisateur dans la simulation. Les sons des véhicules et de la circulation sont simulés dans un environnement 3D.

VTD fonctionne sur les systèmes d'exploitation de type Linux et utilise les technologies de cartes graphiques NVIDIA.

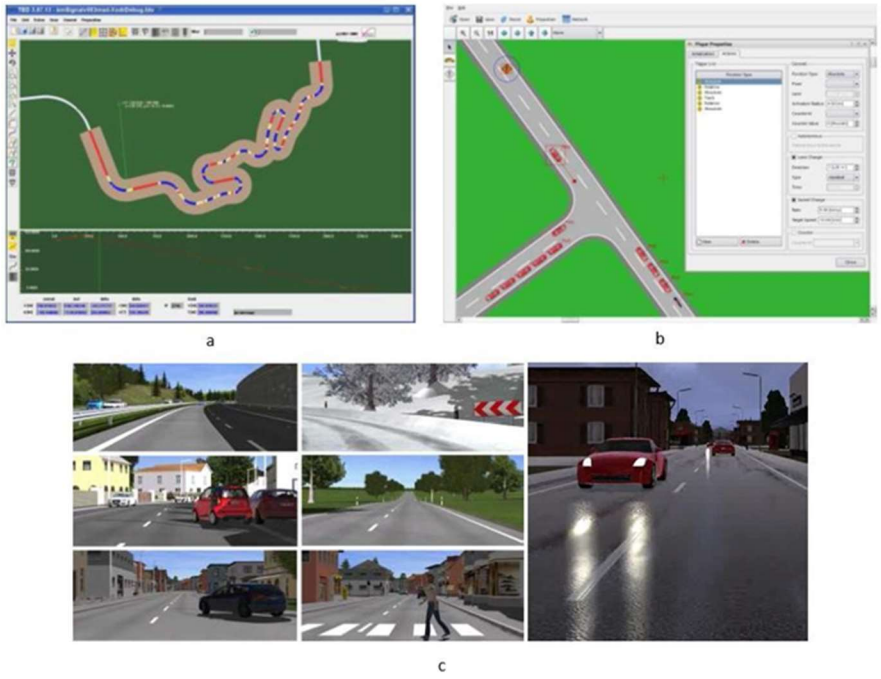

Figure 50 VIRES VDT, a. ROD, b. v-SCENARIO/v-TRAFFIC, c. v-IG

7.5.6. SCANeR (OKTAL)

Développé pour des experts de l'automobile, SCANeR studio (figure 44) est conçu pour répondre aux besoins spécifiques des professionnels de la simulation dynamique. L'utilisation du logiciel a été pensée, initialement, autour du processus d'utilisation des simulateurs de conduite et a été structuré autour de cinq modes dédiés accessibles depuis l'interface graphique:

- **TERRAIN** : Edition de réseau routier RoadXML sous format manuel ou automatique via l'import de différents type de base de données routières (GPX, XML, OSM, SHP, ..);
- **VÉHICULE** : Outil de mise au point et d'étude de modèles dynamiques dispose d'un modèle dynamique non linéaire et peut importer des modèles externes;
- **SCÉNARIO** : Editeur de scénario de simulation de conduite qui dispose de ses propres fonctions qui permettent de définir des scénarii complexes de simulation et il permet également l'import de scénarii développés sous python.
- **SIMULATION** : Outil de supervision de la simulation et du simulateur, couplé à Matlab/Simulink pour une cosimulation entre les deux environnements;
- **ANALYSE** : Outil graphique d'analyse fine des données de simulation. toutes les données du scénario de simulation sont enregistrées d'une manière synchrone et peuvent être affiché dans des graphiques ou exporter sous d'autres formats.

SCANeR studio dispose de plusieurs modèles génériques de capteurs (Radar, Caméra, GPS, Ultrason, …). Ces capteurs transmettent leurs données respectives à Matlab/Simulink via des API. SCANeR est couplé à l'environnement ADASRP ce qui permet d'interroger la base de données routière de ce dernier et récupérer l'horizon électronique* à chaque instant.

La modularité et la richesse de l'environnement SCANeR lui permettent d'être utilisé pour concevoir les systèmes avancés d'aide à la conduite (ADAS), les analyser, les optimiser et les valider :

- Conception des ADAS :
  - Navigation system;
  - Adaptive Cruise Control (ACC);
  - Lane Departure Warning (LDW);
  - Distance Warning (DW);
- Analyse et optimisation « Environnement-Véhicule-conducteur »
  - Environnement
    - Modéliser une infrastructure routière et la parcourir avec différents acteurs.
    - Gestion des conditions météorologiques et d'éclairage.
    - Modéliser le comportement dynamique d'un ou de plusieurs acteurs.
  - Véhicule
    - Gérer les capteurs (Radar, Caméra, GPS, …).
    - Mettre en situation et évaluer les ADAS.
  - Conducteur
    - Modéliser et analyser le comportement du conducteur.

- Validation
  - MIL
  - SIL
  - HIL

Figure 51 SCANeR studio

7.5.7. VR-Design Studio (FORUM 8)

VR- Design Studio (Virtual Reality Design Studio), anciennement appelé UC-Win/Road, est une suite logicielle commercialisée par la société japonaise FORUM 8 (Figure 45). VR-Design Studio est développé pour répondre à deux objectifs. Le premier est de permettre aux utilisateurs de simuler l'environnement réel aussi précisément que possible, est cela en ayant la possibilité de reproduire et de contrôler tous les effets liés à l'environnement tels que : la pluie, la neige, le vent, les ombres, l'éclairage, …etc. le changement de l'horaire (jour, matin, soir, nuit, …) et l'emplacement géographique sont inclus également. Le second objectif est de mettre à la disposition des utilisateurs une gamme de plug-ins qui leurs permettent de modéliser et de simuler des produit logiciels et qui leurs apportes également des sources de données telles que: Laser, Lidar, caméra.

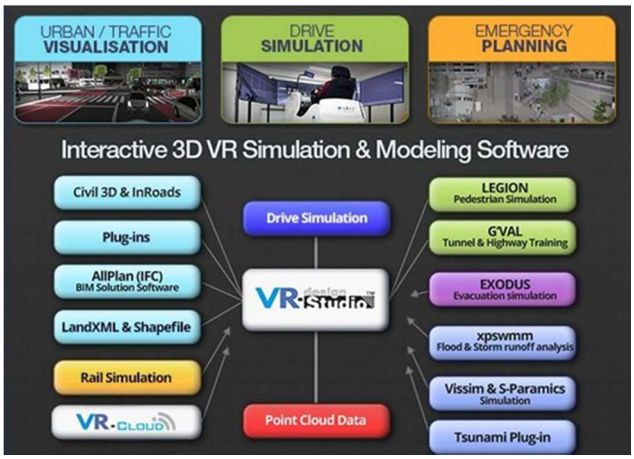

Figure 52 VIRTUAL REALITY DESIGN STUDIO

VR-Design Studio permet de construire une simulation 3D interactive et temps réel. Les utilisateurs peuvent manipuler l'espace 3D, importer ou éditer des données CAO, construire des modèles de textures, construire automatiquement des routes, des tunnels ou des ponts, ainsi que d'être en mesure de visualiser et de modifier le trafic.

VR-Design Studio dispose des fonctionnalités suivantes :

- Une large base de donnée de modèle est disponible : bâtiments, véhicules, personnes, objets urbains, …etc. (Figure 53);
- Créer facilement des structures routières compliquées (Figure 54)
- Affichage en temps réel des espaces à grande échelle : l'outil permet de visualiser un cube de de 1cm et une structure de routes de 20km dans le même espace. Les modèles 3D des végétations, des personnages, des objets et des véhicules sont réalisés en tenant compte le niveau de détail (Figure 54) ;
- Import des données du Cloud ainsi que les enregistrements d'une manière transparente ;
- Conduite dans des conditions réalistes ;
- Changer et contrôler les conditions météorologiques, l'heure du jour, l'éclairage, la géolocalisation, …etc. (Figure 55) ;
- Des déplacements réalistes de piétons d'une manière individuelle ou en foule.
- Générer est gérer le trafic ;
- Cosimulation avec CarSim : l'interfaçage avec CarSim8 permet de réaliser des simulations sur des modèles véhicules réalistes (Figure 56).

Figure 53 Exemple d'objets disponible dans VR-Design Studio

Figure 54 Exemple d'environnements routiers crées dans VR-Design Studio

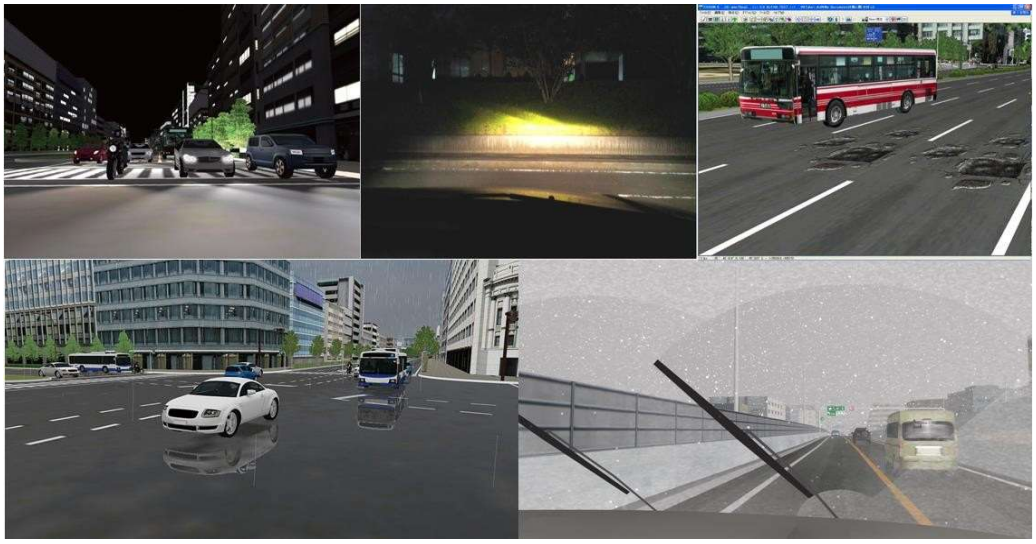

Figure 55 Exemple de conditions météorologiques et d'éclairage dans VR-Design Studio

Figure 56 Cosimulation VR-Design Studio - CarSim

7.5.8. VRXperience (OPTIS)

VRXperience est un simulateur de conduite qui permet de simuler en temps réel les conditions météorologique ou d'éclairage dans un environnement «réaliste», Figure 57. L'environnement de simulation est représenté en 3D et l'affichage peut être déployé sur des écrans 3D.

Cette solution est développée et commercialisée par OPTIS qui est un des leaders de l'édition de logiciels de simulation de la lumière et de la vision humaine.

Figure 57 Image issue de VRXperience

VRXperience permet de reproduire et de simuler différentes conditions climatiques (du brouillard à l'atténuation atmosphérique) ainsi que le flou des mouvements ce qui rend la perception de conduite réaliste à vitesse réelle. VRXperience permet de simuler aussi bien la perception visuelle du conducteur que du passager, mais également de vérifier l'adaptation de l'œil à des changements rapides de lumière.

L'utilisateur peut conduire et tester la conception de sa voiture sur route, dans toutes les conditions, mais également effectuer une simulation virtuelle complète pour valider l'intérieur de la voiture, les phares, les feux de tableau de bord, les rétroviseurs et l'affichage tête haute (HUD).

Le simulateur (Figure 58) inclut tous les éléments nécessaires à une reproduction virtuelle des conditions de conduite : optiques avant, panneau d'instrumentation, cockpits de verre, affichages tête haute et lumières adaptatives. L'utilisateur peut contrôler, gérer et valider virtuellement toutes ces fonctionnalités dans un environnement donné.

VRXperience est interfaçable avec Matlab-Simulink, ce qui permet de gérer les systèmes développés en toute simplicité. VRXperience peut être déployé sur n'importe quelle plate-forme matérielle. L'installation du simulateur est simple et rapide.

Les principales caractéristiques de VRXperience sont listées dans les points suivants :

- Gestion des optiques avant (AFS, éclairage matriciel, laser)
- Intégration du trafic et des piétons
- Pistes et circuits standards et personnalisés
- Simulateur virtuel complet intégrant la géométrie de l'habitacle
- Connexion par bus CAN aux simulateurs matériels et automobiles
- Rendu basé sur la physique
- Compatible avec plusieurs moteurs physiques
- Compatible avec murs d'affichage, moniteurs 3D stéréoscopique, affichages HDRI et centres multi- murs (centres de réalité virtuelle immersifs)
- Compatible Matlab et Simulink

Figure 58 Simulateur VRXperience

7.6. Outils de Génération et simulation de trafic routier

7.6.1. Spécifications

Dans le processus de validation des systèmes d'aide à la conduite il est primordial d'évaluer le comportement de ces derniers dans divers situations de conduite et de trafic.

Le but étant de confronter ces systèmes à toute sorte de situation de conduite (normal, sportive, dangereuse, …) et de trafic (faible, moyen, dense, …) en phase de simulation afin de détecter et de corriger toutes les anomalies avant une éventuelle intégration de ces système dans le véhicule et leurs validation sur route ouverte. Cette section va introduire quelques outils de génération et de simulation de trafic routier.

7.6.2. ASM Traffic (dspace)

ASM Traffic (Figure 59) de dSPACE, élément de la librairie de Modèles de simulation automobile (ASM), fournit toutes les fonctionnalités nécessaires à la simulation pour les tests des systèmes d'aide à la conduite. Grâce à cette technologie, les ingénieurs peuvent vérifier le fonctionnement des systèmes dès les premières phases de leur développement.

ASM Traffic permet d'introduire le véhicule de test dans des réseaux routiers urbains et ruraux avec un grand nombre de véhicules et d'objets environnants tels que des piétons et des panneaux de signalisation. Des capteurs virtuels installés dans le véhicule de test simulé, détectent les objets dans l'environnement simulé et avertissent les ADAS pour qu'ils puissent réagir en conséquence.

ASM Traffic fournit une série de tests prêts à l'emploi destiné aux fonctions de sécurité active d'après le protocole EuroNCAP (European New Car Assessment Program). L'objectif de ces tests est de démontrer que le temps de réaction et de ralentissement du véhicule testé suffisent à éviter ou à atténuer une potentielle collision. L'Euro NCAP test suite de dSPACE fournit des informations sur les notations de sécurité active du véhicule que l'EuroNCAP pourrait décerner pour l'évitement de collision de véhicule à véhicule et les constellations de véhicule à piéton.

ASM Traffic comprend un modèle de simulation temps réel et une interface utilisateur graphique permettant de définir les composants nécessaires tels que les réseaux routiers, les panneaux de signalisation, les véhicules et les capteurs. Il permet également d'importer des informations d'autres outils de cartographie et de simulation.

Figure 59 ASM Traffic

7.6.3. Aimsun (Transport Simulation System-TSS)

Aimsun est logiciel capable de reproduire les conditions réelles de trafic de n'importe quel réseau de transports. Il est entre autres utilisé pour élaborer et tester les systèmes de contrôle de trafic, les règles de gestion du trafic, les contrôles d'accès, l'emplacement des péages, les réseaux de transports publics, les voies réservées, les nouvelles infrastructures ou la prévision de l'état du trafic, et permet de travailler conjointement avec les systèmes de guidage de véhicules et d'autres applications en temps réel (Figure 53).

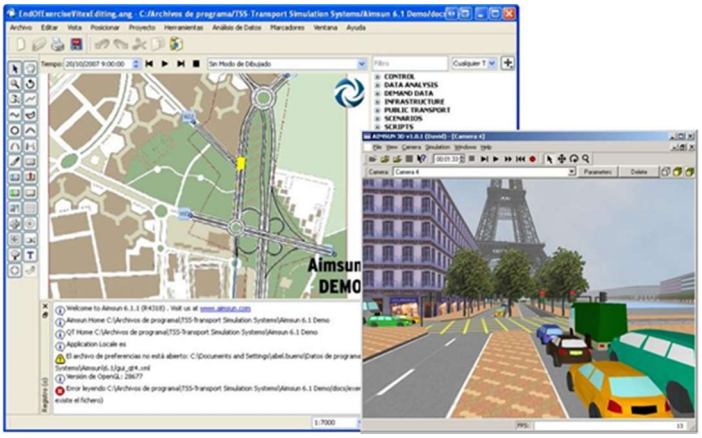

Figure 60 Aimsun

Aimsun modélise le trafic via une représentation simplifiée de la situation réelle avec un calibrage du modèle sur les données de comptages et les observations du fonctionnement (longueurs de file d'attente, temps de parcours).

Les véhicules du trafic suivent des lois dont les principales sont :

- La loi de poursuite (car following) ;
- Les lois de changements de voies (lane changing) ;
- Les lois d'acceptation des créneaux (gap acceptance).

Certains éléments du modèle de trafic changent de manière continue (véhicules, détecteurs) pendant la durée de la simulation. Les autres éléments changent de manière discrète (état des feux par exemple) à des instants donnés.

Un nombre important de paramètres sont tirés de manière aléatoire dans des distributions statistiques :

- Caractéristiques de véhicules ;
- Horaire d'entrée dans le modèle de trafic.

Chaque simulation avec sa génération aléatoire est appelée « réplication », pour un même scénario il faut lancer plusieurs réplications pour pouvoir caractériser correctement le fonctionnement du scénario.

Aimsun fournit des statistiques détaillées pouvant être imprimées : flux, temps de parcours, véhicules flottants, … etc. Des mesures environnementales comme la consommation de carburant et les émissions de pollution sont également disponibles. Aimsun inclut un outil de calibration et d'analyse des résultats qui permet de comparer les données graphiques et numériques des simulations avec les données du monde réel.

Il est possible de personnaliser l'outil Aimsun via un module API (Application Programming Interface). Ce module API est composé d'un ensemble de fonctions Python et C++ permettant à l'utilisateur:

- D'inclure des systèmes de transport intelligent pour les évaluer dans la simulation :
  - Systèmes de contrôle de feux de circulation ;
  - Gestion dynamique du trafic ;
  - Guidage de certains véhicules ;
- D'accéder à l'ensemble des données du simulateur pendant son déroulement pour y inclure :
  - Son propre modèle de consommation ou d'émissions de polluants ;
  - Ses propres indicateurs de trafic ;
  - Ses propres interfaces avec des applications externes.

Aimsun dispose également d'un module micro SDK (Aimsun Microscopic Simulator Software Development Kit) qui permet de remplacer les modèles de comportement d'Aimsun (loi de poursuite, loi de changement de voies, loi d'acceptation des créneaux, … etc.) par ses propres lois programmées en C++.

Ce module SDK peut être utilisé pour programmer les comportements des véhicules disposant de dispositifs spécifiques d'aide à la conduite.

7.6.4. Sumo (Simulation of Urban Mobility – Open source)

SUMO (Figure 61) est un simulateur de trafic routier libre et ouvert depuis sa mise en disposition en 2001. Il permet la modélisation des systèmes de trafic macro et microscopique. En d'autres termes, il intègre les véhicules, les transports publics et les piétons.

Différents outils sont inclus avec SUMO pour permettre par exemple de rechercher l'itinéraire, la visualisation, l'import du réseau routier, …etc.

La structure ouverte est libre de SUMO lui permet d'intégrer des modèles externes personnalisés, il fournit également des APIs qui permettent aux utilisateurs de contrôler la simulation à distance.

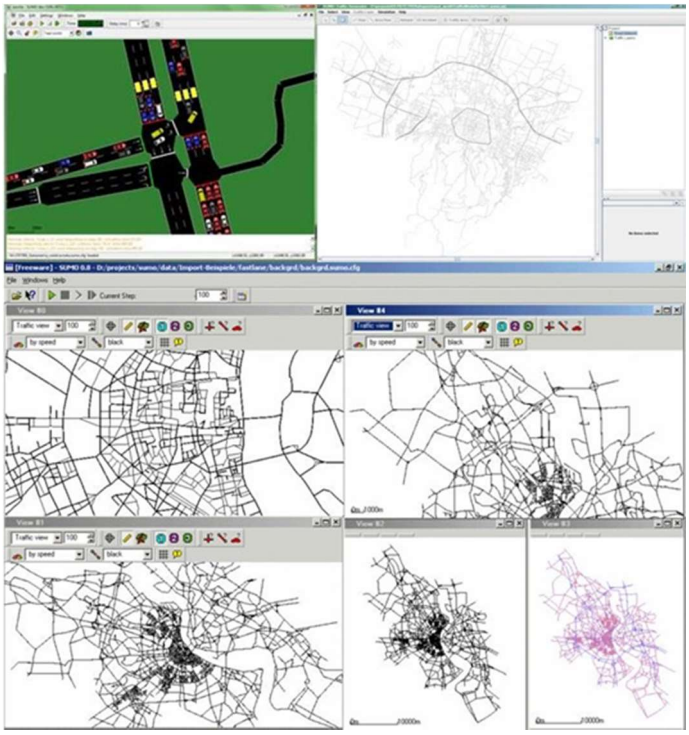

Figure 61 Sumo

Les principales caractéristiques de SUMO sont les suivantes :

- Simulation microscopique du trafic : les véhicules, les piétons et les transports publiques sont explicitement modélisés ;
- Interaction en ligne : contrôle de la simulation via l'outil TraCi ;
- Simulation du trafic multimodal ;
- Pas de limitation de taille du réseau routier ni du nombre de véhicules simulés ;
- Les formats d'imports supportés : OpenStreetMap, VISUM, VISSIM, here (anciennement appelée NavTeq) ;
- SUMO est implémenté en C++ et utilise des librairies portables.

SUMO a été utilisé dans divers projets pour répondre à un large panel de questions. A titre d'exemple on peut citer :

- Evaluation des performances des feux de signalisation ;
- Développement et évaluation de nouvelles méthodes de recherche de trajets ;
- Evaluation des systèmes de communication V2X.

7.6.5. PTV Vissim (PTV GROUP)

PTV Vissim (Verkehr In Städten – SIMulationsmodell) est un simulateur de trafic (microscopique et multimodal) routier. Développé par Planung Transport Verkehr (PTV), il permet «entre autres» de comparer des géométries de carrefours, d'analyser les priorités données aux transports publics ou d'identifier les effets de certaines signalisations, …etc.

PTV Vissim (Figure 62) permet de reproduire dans un seul modèle de simulation tous les usagers du transport (Transport individuel motorisé, transport de marchandises, transports en commun ferroviaires et routiers, piétons et cyclistes, …) ainsi que leurs interactions. Les modèles d'écoulement et de comportement sont basés sur des approches scientifiques garantissent une modélisation réaliste de l'ensemble des usagers.

Vissim offre une certaine flexibilité : Le concept de tronçons et de liaisons permet aux utilisateurs de modéliser des géométries complexes. Les paramètres décrivant les propriétés des conducteurs et des véhicules permettent une configuration personnalisée. Les propriétés de base, du type vitesse souhaitée et comportement d'accélération et de décélération, les comportements de succession des véhicules et de changement de voie déterminent sur une base opérationnelle le flux de trafic (Figure 63).

Figure 62 PTV VISSIM

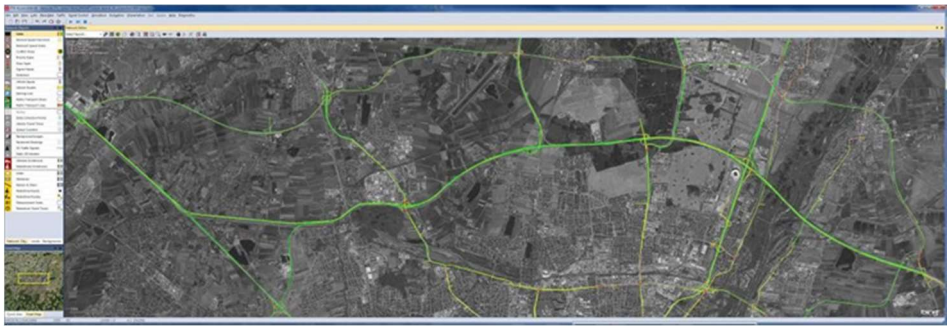

Figure 63 Simulation de réseaux autoroutiers avec PTV Vissim : visualisation des vitesses de segments de tronçons

PTV Vissim fait partie d'une suite logicielle appelée «Vision Traffic Suite» qui inclut : PTV Visum (pour l'analyse et la prévision du trafic) et PTV Vistro (pour l'optimisation de la signalisation et l'impact du trafic). Cette suite logicielle est très utilisée pour la planification, l'ingénierie et la simulation du trafic.

Les points suivants listent les domaines d'application de la suite logicielle Vision Traffic Suite :

- Planification du trafic ;
- Ingénierie du trafic ;
- Régulation du trafic par feux ;
- Transports collectifs ;
- Emissions et environnement.

La Figure 64 résume les fonctionnalités et les applications de la suite.

	PTV Visum Modélisation du trafic	PTV Vissim Simulation du trafic	PTV Viswalk Simulation de piétons
Planning level	macroscopique / mésoscopique	microscopique	microscopique
Model size	unlimited	unlimited	unlimited
Development of integrated multimodal networks, including data on travel demand and services for private and public transport	✓	✓	
Development of integrated multimodal networks for pedestrians	✓	✓	✓
Interactive analysis of travel demand and services and their variants	✓		
Analysis of individual intersections	✓	✓	
Traffic control inside and outside the city	✓	✓	
Optimisation of services and transport operations in PuT	✓	✓	
Modelling and analysis of traffic-related emissions	✓	✓	
Scenario analysis / Comparison of travel demand and services and their variants	✓	✓	
Realistic modelling of lane geometry & network element positions at any level of detail		✓	✓
Simulation of road and rail transport and pedestrian flows		✓	
Pedestrian simulation			✓
Result analysis and visualisation	✓	✓	✓

Figure 64 fonctionnalités de la « Vision Traffic Suite »

7.7. Outils d'implémentation d'Algorithmes de contrôle commande

7.7.1. Spécifications

Les outils de cette catégorie sont utilisés, entre autres, pour le développement des algorithmes de contrôle du véhicule. Le concepteur peut modéliser et régler l'algorithme de contrôle mis en œuvre selon les exigences du projet et de simuler le modèle avec l'ensemble du système dans des contextes d'usage définis.

Une fois que l'algorithme est fonctionnellement correct, le concepteur peut l'optimiser (en termes de sécurité, de performances, de maintenabilité, …etc.) via des outils qui permettent d'identifier les problèmes potentiels et de recommander les changements appropriés.

A partir du modèle développé, il est possible de générer automatiquement du code C et le télécharger sur une plate-forme matérielle. Le code source généré peut être utilisé pour des applications en temps réel ou en temps simulé, comme la simulation accélérée, le prototypage rapide, et les tests hardware-in-the- Loop, … etc.

En résumé, les outils pour la conception des algorithmes de contrôle prennent en charge toutes les étapes du processus de développement, de la modélisation du système jusqu'à son implantation sur cible grâce à la génération automatique du code C.

Dans les points ci-dessous sont résumées, les principales caractéristiques requises dans les outils de développement des algorithmes de contrôle :

- Large bibliothèque : présence de différent composant permettant simuler le comportement du système dans différentes conditions de fonctionnement ;
- Disponibilité de différents types de solveurs et des modèles mathématiques déjà développé ;
- Relier les exigences au modèle et au code : la possibilité de trouver les codes et les modèles qui mettent en œuvre une ou en ensemble d'exigences et vice versa ;
- La possibilité générer le code (C, C++, Java, …) pour une utilisation sur des cibles embarquées, cartes de prototypage rapide, …etc. ;
- Possibilité d'intégration de code externe ;
- Possibilité de détecter les erreurs, effectuer des analyses de couverture et d'autre vérifications pour améliorer la robustesse et la qualité du modèle et du code ;
- Possibilité d'interfacer l'outil avec d'autres modules ou environnement de développement;
- Disponibilité d'une interface graphique.

7.7.2. Matlab/Simulink (MathWorks)

MATLAB est un langage de haut niveau et un environnement interactif utilisés pour explorer et visualiser des idées et se prête à une collaboration inter-disciplines, parmi lesquelles le traitement du signal et de l'image, les communications et les systèmes de contrôle.

MATLAB dispose de plusieurs composants appelés « Toolbox » regroupés dans un environnement de diagramme blocs appelé Simulink. Ces Toolbox permettent de modéliser et de simuler un large panel d'événements dans différents domaines de la physique (la consommation de l'énergie électrique, le développement d'algorithmes de contrôle du véhicule (Figure 65), d'analyser et le traitement des données, …etc.).

Comme mentionné précédemment, Simulink est un environnement de diagramme fonctionnel destiné à la simulation multi-domaine et à l'approche de conception par modélisation Model-Based-Design. Il prend en charge la conception et la simulation au niveau système, la génération automatique de code, ainsi que le test et la vérification en continu des systèmes embarqués.

Création du modèle : Simulink propose un ensemble de blocs prédéfinis qu'ils peuvent être combinés pour créer un schéma fonctionnel détaillé du système. Les outils destinés à la modélisation hiérarchique, à la gestion des données et à la personnalisation des sous-systèmes permettent de représenter avec concision et précision les systèmes les plus complexes.

Simulation du modèle : Simulink permet de simuler le comportement dynamique du système et visualiser les résultats pendant l'exécution de la simulation. Pour garantir la vitesse et la précision du processus, Simulink propose des solveurs ODE à pas fixes ou variables, un débogueur graphique et un outil de profilage de modèle.

Analyse du résultat de la simulation : Après l'exécution d'une simulation, il est possible d'analyser le résultat dans MATLAB et Simulink. Simulink intègre des outils de débogage qui aident à comprendre le comportement de la simulation.

Gestion de projets : Simulink fournit des outils permettant de gérer les fichiers, les composants et les grandes quantités de données relatifs au projet.

Connexion à un équipement matériel : Il est possible de connecter le modèle Simulink à un équipement matériel afin d'effectuer du prototypage rapide, une simulation HIL (hardware-in-the-loop) ou un déploiement sur un système embarqué.

Simulink propose un éditeur graphique (Figure 65), un ensemble personnalisable de bibliothèques de blocs et des solveurs pour la modélisation et la simulation de systèmes dynamiques. Il est intégré à MATLAB, ce qui permet d'incorporer les algorithmes MATLAB dans les modèles et d'exporter le résultat des simulations vers MATLAB pour compléter les analyses.

Pour résumé, Simulink dispose des fonctionnalités suivantes :

- Éditeur graphique pour la création et la gestion de diagrammes fonctionnels hiérarchiques ;
- Bibliothèques de blocs prédéfinis pour la modélisation de systèmes en temps continu ou en temps discret ;
- Moteur de simulation avec des solveurs ODE à pas fixes ou variables ;
- Scopes et affichage de données pour la consultation du résultat de la simulation ;
- Outils de gestion de données et de projets pour les données et fichiers des modèles ;
- Outils d'analyse de modèles pour l'affinage de l'architecture de modèle et l'augmentation de la vitesse de simulation ;
- Bloc MATLAB Function pour l'import d'algorithmes MATLAB dans les modèles ;
- Outil Legacy Code Tool pour l'import rapide de code C et C++ existant dans les modèles.

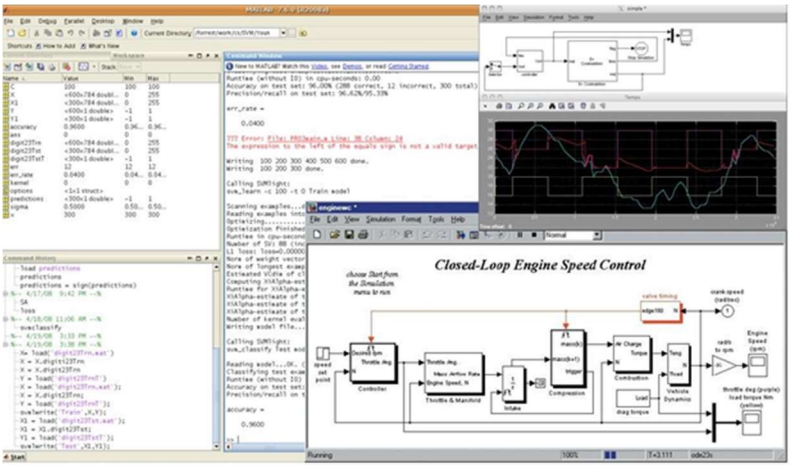

Figure 65 Matlab/Simulink

7.7.3. Autres

Quelques outils présentés dans les sections précédentes permettent également le développement des algorithmes de contrôle.

RTMaps: RTMaps dispose d'un module SDK (Software Development Kit) qui permet de développer des composants en C et C++ représentant des algorithmes de contrôle. Ces composants peuvent être intégrer dans des diagrammes RTMaps, alimenter par des données capteurs (enregistrées ou en directe) et agissent sur les actionneurs adéquats (Figure 66)

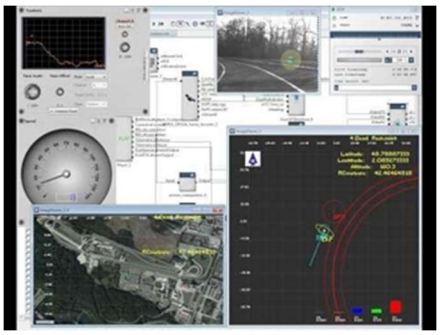

Figure 66 RTMaps pour le développement des ADAS

ADTF: ADTF dispose également d'un module SDK. De la même manière que RTMaps, il permet de développer des algorithmes de contrôle en C++ est les intégrer dans l'environnement de simulation.

7.8. Outils de Génération de cas d'usages et de tests

7.8.1. Spécifications

Le cycle en V est aujourd'hui bien connu des professionnels pour représenter les différentes phases de développement d'un produit. Les tests fonctionnels, réalisés sur la branche remontante (tests d'intégration, de système, d'acceptation), valident les réalisations et favorisent le retour d'informations aux équipes de conception. Malheureusement, on s'aperçoit souvent trop tard des manques ou incohérences contenues dans les spécifications. C'est pourquoi, l'idée de réaliser les tests au plus tôt (dès l'établissement des spécifications et la conception du système - branche descendante du cycle en V) a été mise en place.

Ces tests d'exigences sont conçus au travers de modèle de scenarios d'utilisation, ce qui permet des analyses fines du système à tester (couverture fonctionnelle, opérationnelle, du besoin).

Cette évolution a impactée le processus de conception et de validation de ces systèmes par l'apparition de de nouvelles méthodes (Model-Based-Development, Model-Based-Testing, ...) et d'outils permettant d'optimiser ces processus.

Model-Based-Testing (MBT) est une méthode systématique qui dérive du Model-Based-Design. Elle permet de générer des cas de test à partir des exigences du système. Elle permet également d'évaluer les exigences indépendantes de la conception et du développement des algorithmes du système (Figure 67).

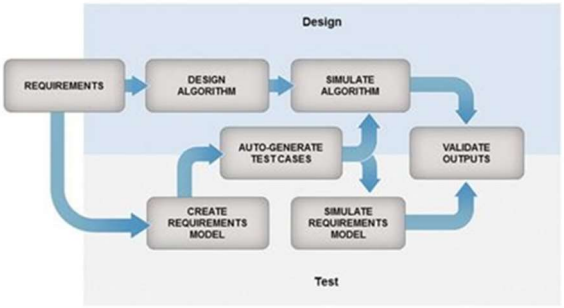

Figure 67 Interaction MBD - MBT

On trouve dans le commerce plusieurs outils permettant de générer automatiquement ces cas de tests. Certains de ces outils permettent également d'automatiser le processus de test.

Ces outils apportent un réel avantage aux équipes responsables des tests de validation :

- Permettent d'éviter des cas de test mal conçus, défectueux ou manquants, et du même coup, accroît la couverture des tests;
- Réduises les coûts pour les tests (délais, tests de non régression) ;
- Améliorent la qualité du processus de test ;
- Diminuent les efforts de mises à jour des cas de tests ;
- Renforcent la qualité de la documentation des exigences ;
- Créent une plateforme commune pour les concepteurs et les valideurs.

7.8.2. Simulink Design Verifier (MathWorks)

Simulink Design Verifier est un outil qui permet l'identification des erreurs de conception, la génération de cas de test et la vérification de la conformité des conceptions aux spécifications (Figure 68).

Simulink Design Verifier utilise des méthodes formelles afin de détecter dans des modèles des erreurs de conception difficiles à repérer sans nécessiter de tests étendus ou de simulations. Il met en évidence dans le modèle les blocs contenant ces erreurs et les blocs qui en sont exempts. Pour chaque bloc avec erreur, il calcule les limites de la plage de signaux et génère un vecteur de test qui reproduit l'erreur dans un environnement de simulation. Les vecteurs de test générés fournissent des entrées de simulation qui exercent la fonctionnalité capturée dans la structure du modèle et indiquée par les objectifs de test. Les vecteurs de test, associés aux propriétés de conception et aux objectifs de test, peuvent être utilisés pour vérifier l'exécution du code dans les configurations de test Software-In-the-Loop (SIL) et Processor-In-the- Loop (PIL).

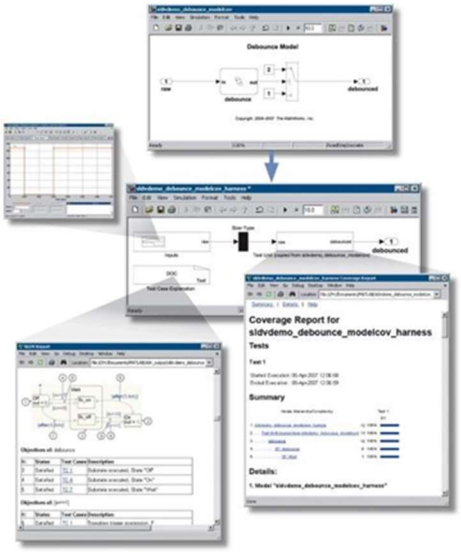

Figure 68 Simulink Design Verifier

Principales fonctionnalités :

- Moteurs d'analyse formelle Polyspace® et Prover Plug-In® ;
- Détection de logique morte, dépassement entier et virgule fixe, division par zéro et violation de propriétés de conception ;
- Blocs et fonctions pour la modélisation des spécifications fonctionnelles et sécuritaires ;
- Génération de vecteur de test à partir de spécifications fonctionnelles et d'objectifs de couverture du modèle, y compris la couverture de condition, de décision et de condition/décision modifiée (MC/DC - Modified Condition/Decision Coverage) ;
- Preuves de propriétés, avec génération d'exemples de violation pour l'analyse et le débogage ;
- Prise en charge des modèles en virgule fixe et flottante.

Simulink Design Verifier est certifié TÜV SÜD pour une utilisation dans le cadre de processus de développement devant se conformer aux normes ISO 26262, IEC 61508 ou EN 50128.

7.8.3. MaTeLo (all4tec)

MaTeLo (Markov Test Logic) est un outil logiciel basé sur l'approche Model-Based Testing (MBT). Il permet d'améliorer la performance de tout le cycle de validation et intègre la modélisation des tests, la génération automatique des tests avec leurs informations de traçabilité ainsi que l'analyse qualitative des campagnes effectuées.

MaTeLo met en œuvre cette approche MBT dans un environnement graphique. À partir des usages de l'application, les exigences ou les témoignages d'utilisateurs, MaTeLo est capable de générer automatiquement des cas de tests optimisés basés sur l'analyse des risques, la couverture et les résultats escomptés. Les cas de tests peuvent être exportés soit vers des outils automatiques d'exécution ou vers des gestionnaires d'outils de test pour une exécution manuelle.

Différentes approches peuvent être envisagées lors de la génération des cas de test suivant les objectifs de la campagne de tests et de la maturité du système à tester. Ainsi, MaTeLo peut générer des cas de test focalisés pour la validation des fonctions nominales, issus d'une analyse

de risques, optimum pour la couverture des exigences et enfin priorisant l'utilisation opérationnelle du système, pour en valider les objectifs de fiabilité.

Les cas de test générés automatiquement peuvent ainsi être traduits dans le format adéquat pour être par la suite exécutés grâce aux environnements d'automatisation liés aux bancs de tests et simulateurs utilisés : tels que TestStand de National Instruments, PROVEtech de MBTech, EXAM de MicroNova, XML ou tout langage de Scripting utilisable pour une adaptation personnalisée.

MATELO comprend 2 modules principaux :

1. Editeur de modèles d'usage (MaTeLo Editor, Figure 69) : représente les comportements du système et il permet de :
    1. Capture en entrée les éléments de spécification UML (XMI), SDL ou autre forme de spécification
    2. Réalisation du modèle :
        a. Création des états stables du système ;
        b. Définition de toutes les actions possibles dans un état donné, puis on réalise le modèle ;
        c. Intégration des données d'entrée et des résultats attendus ;
        d. Pose des fréquences d'usage ;
        e. Génération des cas de test « à blanc » pour valider le modèle ;
        f. Sauvegarde le modèle en XML pour en faciliter l'exploitation.
2. Générateur de cas de tests (MaTeLo Testor, Figure 70) :
    1. Génère automatiquement l'ensemble des scénarios de test nécessaire à la validation de vos systèmes :
        a. Tests de début de validation ;
        b. Tests d'usage ;
        c. Tests de couverture ;
        d. Tests aux limites.
    2. Permet une exécution des tests manuelle (HTML) ou automatique via les automates du marché (type TestStand, TTCN3 ou autres via l'export XML) ;
    3. Génère une analyse de la campagne de test (couverture fonctionnelle, …)

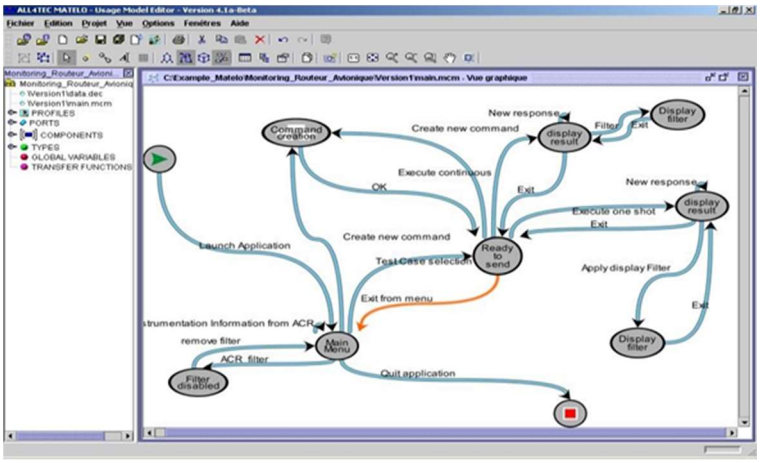

Figure 69 MaTeLo Editor

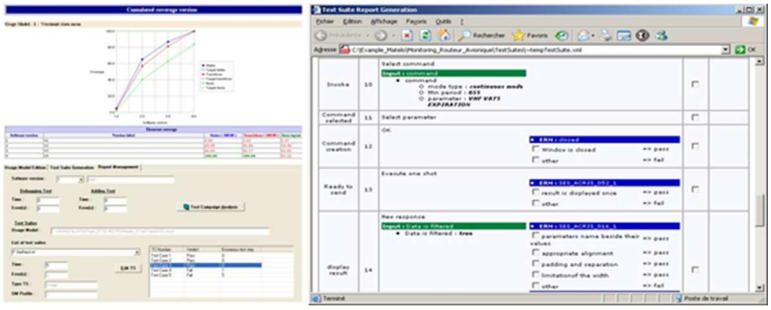

Figure 70 MaTeLo Testor

7.8.4. CertifyIt (Smartesting)

La solution CertifyIt (Figure 71) supporte la modélisation des processus et règles métier, des comportements applicatifs pour piloter la génération et la maintenance des tests. La gestion de la traçabilité est gérée automatiquement et les tests publiés et maintenus dans le référentiel de votre choix (tels que HP Quality Center, IBM RQM, Testlink, Squash TM, ...).

CertifyIt aide à industrialiser et à optimiser le processus de test : de façon systématique les processus et règles métier sont couverts par les tests générés et les évolutions fonctionnelles prises en compte avec un point de maintenance facilitant la prise en compte des évolutions (Figure 72).

- Faciliter le dialogue entre les parties (IT, Métier, Assistance à maîtrise d'ouvrage) :
  - Formaliser graphiquement ou réutiliser les processus métiers des applications à tester ;
  - Retranscrire les règles de gestion et le comportement attendu de l'application ;
  - Détecter d'éventuelles incohérences dans la description du fonctionnement de l'application.
- Améliorez la qualité des applications : moins d'erreurs, meilleur respect des exigences fonctionnelles :
  - Gérer les priorités de vos objectifs de test ;
  - Piloter par les risques ;
  - Valider que la conception des tests couvre bien les besoins fonctionnels ;

- Disposer dans CertifyIt de l'ensemble des informations nécessaires à la génération des scénarios de tests.

- Réduisez les coûts et délais de mise en place des applications :
  - Générer automatiquement le plan avec les scénarios de tests ;
  - Alimenter votre référentiel de test avec des scénarios adaptés au profil des personnes qui vont exécuter ces tests ;
  - Publier les scénarios de tests sous la forme de scripts exécutables par les robots, avec un glossaire de mots clefs réutilisables et accélérer l'automatisation de l'exécution ;
  - Intégrer des évolutions fonctionnelles de l'application grâce au point de maintenance unique ;
  - Visualiser l'impact des évolutions dans l'ensemble des scénarios et mettre à jour le référentiel de test.

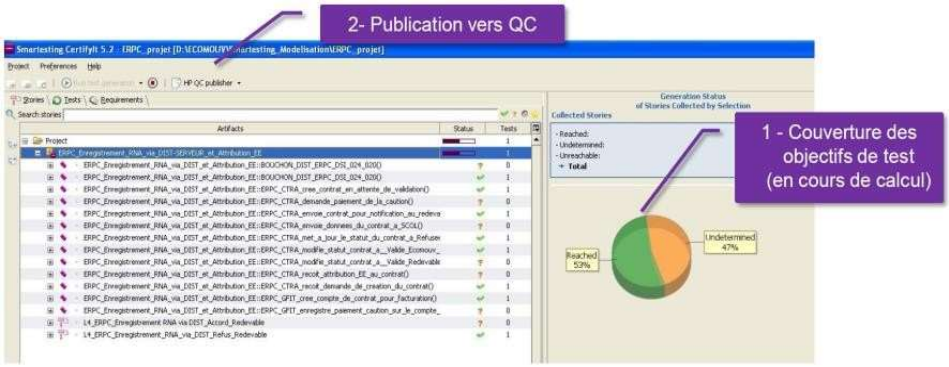

Figure 71 CertifyIt

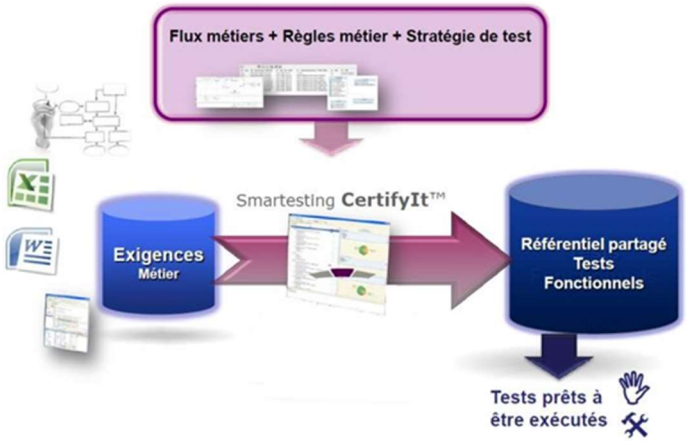

Figure 72 L'approche CertifyIt

7.9. Outils d'Architecture système et de bureautique

Dans cette section nous allons citer quelques outils, plus au moins connus, qui sont utilisés dans différentes étape du cycle en V. En particulier nous revisitons des outils pratiques de gestion des documents rédaction, ainsi que les outils de conception et d'architecture système. Il est vrai que ces outils ne sont pas jugés réellement comme des outils de simulation, mais certain permettent de faire de la simulation de haut niveau ou prépare les scénarios/tests pour la simulation.

7.9.1. Microsoft Office

La suite office est utilisée tout le long du cycle en v. à titre d'exemple :

- Excel est utilisé pour lister toutes les exigences du système, il peut également être utilisé pour décrire l'évolution temporelle des cas d'usage … ;
- Word est utiliser pour rédiger tous les livrables liés à chaque étape (CDCPF, DAS, RV, …etc.). il peut également être utilisé pour rédiger les exigences du système et pour décrire les cas d'usages… ;
- PowerPoint : présentation du projet, présentation du système, présentation de l'état d'avancement, … ;

7.9.2. Rational Doors (IBM)

Rational DOORS (Figure 66) est un outil qui permet d'optimiser la gestion des exigences en matière de communication, de vérification et de collaboration. Il permet également de capturer, d'effectuer le suivi, d'analyser et de gérer les changements d'informations et de démontrer la conformité aux réglementations et normes.

Les principales caractéristiques de Rational DOORS sont les suivantes :

- Gestion des exigences : Offre un environnement complet de gestion des exigences ;
- Traçabilité : Relie les exigences aux éléments de conception, aux plans de test, aux scénarios de test et à d'autres types d'exigences ;
- Evolutivité : S'adapte aux besoins changeants en matière de gestion des exigences ;
- Kit d'outils de suivi des tests : Inclut le kit d'outils de suivi des tests pour les environnements de tests manuels afin de relier les exigences aux scénarios de test ;
- Intégrations : Gère les changements des exigences soit avec un système simple de propositions de
- changements prédéfinis, soit avec un flux de travail de contrôle des changements plus complet et personnalisable s'intégrant aux solutions Rational de gestion des changements.

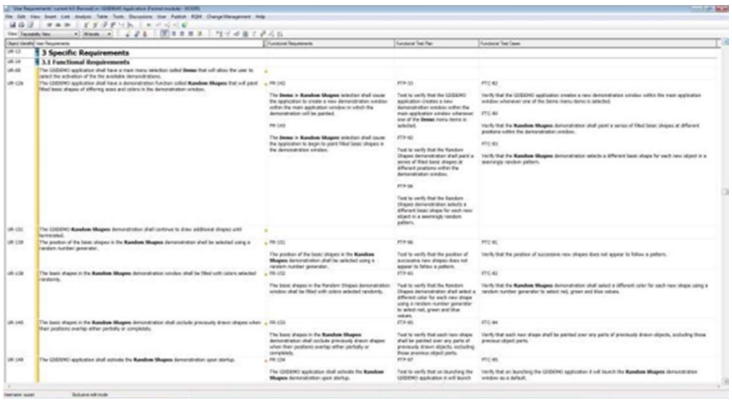

Figure 73 Rational DOORS

7.9.3. arKItect (KI – Knowledge Inside)

arKItect est une suite logicielle (arKItect SEA et arKItect Designer) qui permet de représenter des systèmes complexes hiérarchiques. Afin de s'adapter à différents domaines, arKItect dispose d'un éditeur de méta- modèles à la fois simple d'utilisation et permettant de décrire la plupart des domaines. Une fois le méta- modèle défini, l'utilisateur dispose directement d'un éditeur graphique permettant de développer son modèle (Figure 74).

arKItect SEA (Systems Engineering Advanced) propose un environnement graphique complet dédié à la mise au point des spécifications et des cahiers des charges de systèmes ou de sous-systèmes en relations. Ses fonctionnalités clés sont les suivantes :

- Description d'architectures fonctionnelles et physiques ;
- Allocation des exigences sur les fonctions ;
- Intégration du plan de validation ;
- Suivi des évolutions…etc.

arKItect Designer est l'outil dédié à la définition des méta-modèles graphiques. Il dispose de :

- Un éditeur de matrices de relations permet à l'utilisateur de définir ses propres objets et flux, de leur attribuer des attributs et de définir leurs règles de composition ;
- Un éditeur de vues permet de définir des projections spécifiques dans lesquelles seuls les éléments graphiques utiles ou autorisés sont disponibles ;
- Une API Python permet d'enrichir les objets et modèles de scripts réalisant des fonctions spécifiques : génération de rapport, interface à des produits tiers…

arKItect propose également des méta-modèles permettant de mettre en œuvre une démarche d'Ingénierie Système dans le cadre d'un processus type INCOSE (International Council on Systems Engineering).

arKItect dispose de modules ou d'interfaces qui lui permettent de s'interfacer avec des logiciels tiers. L'interface « Doors requirement » permet de créer un lien entre les exigences et

l'architecture système. L'intégration des exigences dans l'architecture globale du système permet de transformer efficacement le besoin client en une solution produit (Figure 75).

arKItect Excel Model Gateway permet de représenter graphiquement, dans un modèle synchronisé, des données importées d'un large éventail de feuilles de calcul.

Pour les besoins d'analyse de sûreté de fonctionnement, arKItect dispose d'un module safety.

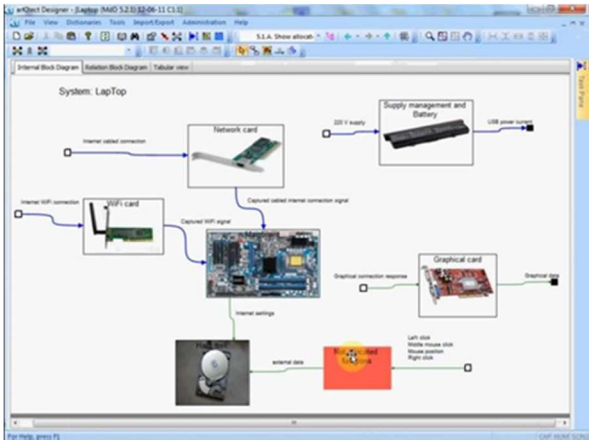

Figure 74 arKItect Designer

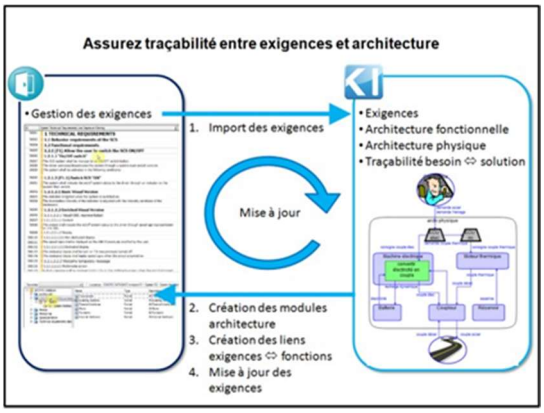

Figure 75 Interfaçage Doors-arKItect

7.9.4. Papyrus (Eclipse)

Papyrus (Figure 69) est un outil de modélisation UML intégré à Eclipse qui a la particularité d'implanter fidèlement le standard. En outre, il supporte les profils standards SysML pour la modélisation système et MARTE pour la modélisation de systèmes temps-réel embarqués. Sur ce dernier point, son grand avantage est de supporter la syntaxe du langage VSL (Value Specification Language) pour la spécification et la vérification de valeurs complexes (tuples, types physiques, etc.). Il facilite également la spécification et l'outillage de nouveaux profils.

L'intégration de Papyrus dans Eclipse permet de profiter des outils de transformation de modèles comme ATL (model-to-model) ou Acceleo (model-to-text) afin de créer des chaînes de

transformations et d'automatiser des bouts de processus de développements par l'ingénierie des modèles. L'outil fournit par défaut des templates pour la génération de code C++ ou java.

Papyrus offre également un support très avancé pour les profils UML qui permet aux utilisateurs de définir des éditeurs pour les DSL (Domain Specific Language) basés sur le standard UML 2.

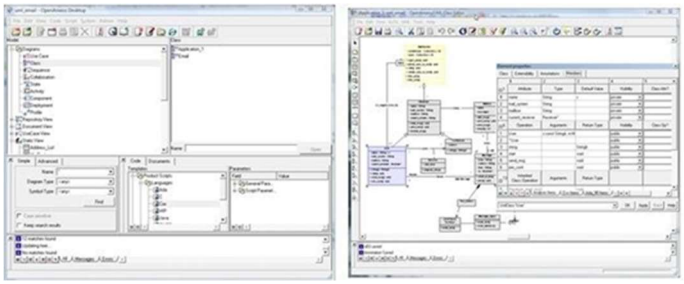

Figure 76 Papyrus

7.9.5. MagicDraw

MagicDraw est un outil de conception, d'architecture système et de gestion de documents. L'outil permet d'utiliser plusieurs langages comme :

- UML pour la conception de logiciels
- SysML pour la conception et architecture système
- BPMN pour la modélisation des entreprises et des processus du business/commerce

L'outil possède plusieurs plugins permettant d'étendre les fonctionnalités de base pour la génération automatique de documentation, la collaboration entre groupe sur les projets. Des outils de simulation haut niveau, comme les simulations par machine à états sont également implémentés.

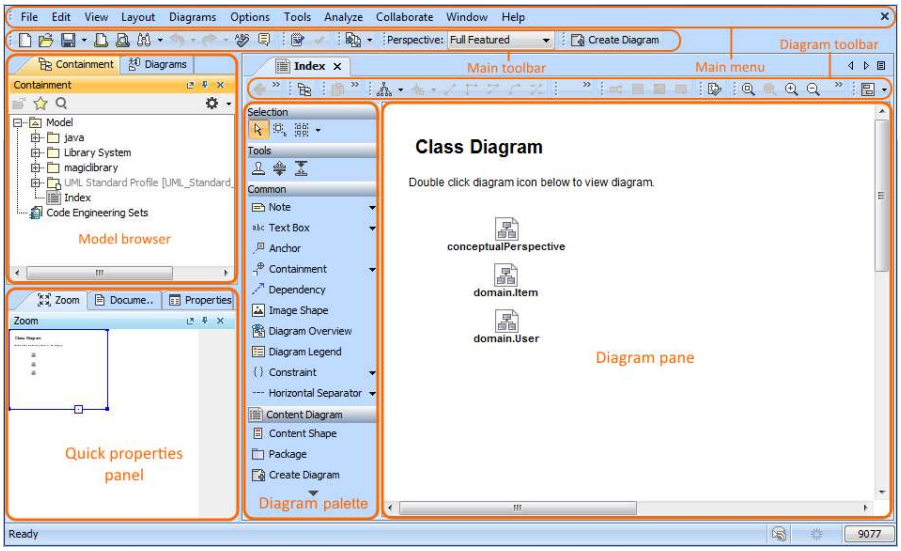

Figure 77 MagicDraw

7.9.6. Capella

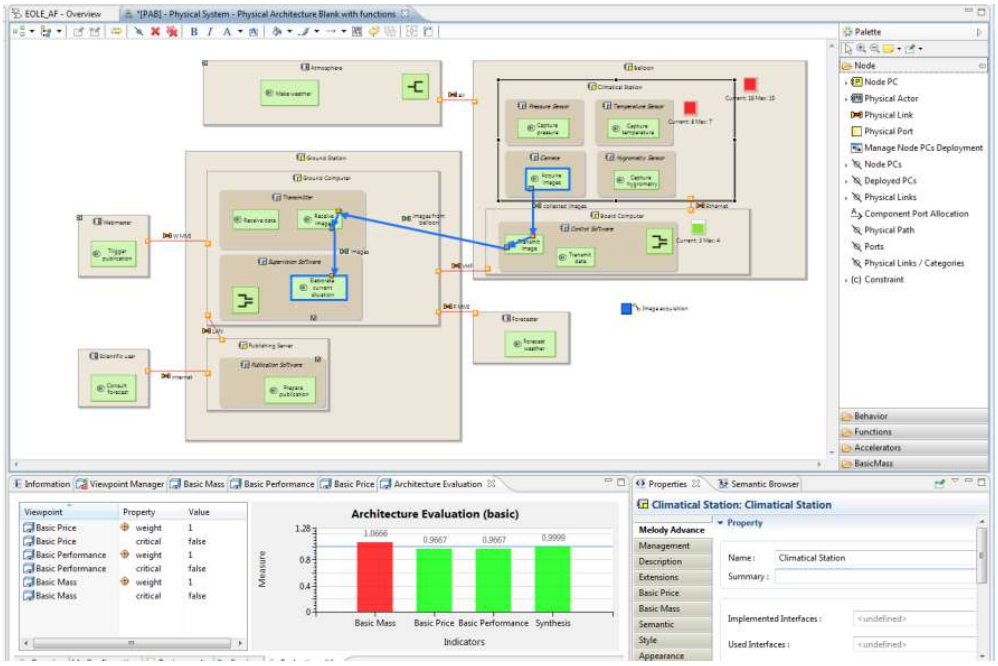

Figure 78 Environnement de développement système Capella

Capella est un logiciel en source ouverte développé par Thalès qui fournit des outils de modélisation graphiques de systèmes matériel ou logiciel.

L'outil est utilisé principalement pour modéliser les systèmes complexes et critiques et le développement des systèmes embarqués utilisés dans l'industrie comme en aéronautique, avionique, transport, la communication et sécurité des automobiles. Capella offre des vue correspondant à l'architecture système classique tel que l'architecture opérationnelle, logique ou physique.

8. Discussion sur les limitations et les attentes de la simulation

Des directions futures émergent quant à l'utilisation de la simulation pour le développement d'un système complexe.

Rappelons d'abord les avantages de la simulation lorsque l'on dispose d'un modèle simulable :

- La simulation permet **l'étude et l'analyse** du comportement d'un système **avant de l'avoir construit** et donc d'économiser les coûts de modification du système réel dans les phases en aval de conception et de réduire les risques qui peuvent être engendré par un test réel
- **Identifier et anticiper les défauts** du modèle et certains problèmes de compatibilité/intégration lorsque les traces de simulation ne montrent pas le comportement attendu du système.
- **Réduire le nombre des essais physiques nécessaires** (qui sont en général assez couteux) pour la validation du système lorsqu'il est possible de faire le même test en simulation avec des résultats précis
- **Contribue aux avancés rapide** de la technologie **en facilitant l'extension des modèles existants** pour ajouter de nouvelles fonctionnalités d'une manière incrémentale : il est facile d'expérimenter avec des ajouts de nouvelles fonctionnalités et de simuler le système global quand on dispose d'un modèle simulable

Cependant, la simulation a plusieurs limitations

- **La simulation n'est pas assez précise qu'un essai physique**, d'autant plus, quantifier cette précision (et par conséquent l'erreur induite par une précision trop large) est difficile.
- **Un modèle fidèle à la réalité peut s'avérer couteux**, voire même indisponible et pas faisable pour certains système. Les résultats de simulation d'un modèle infidèle sont en général inutilisables.
- **Nécessite des travaux théoriques importants :** les modèles doivent obéir aux lois physiques réels et s'appuie donc sur les travaux théoriques connues à ce jour. Or certains pour certains phénomènes, les scientifiques n'en possèdent pas des règles fondamentales qui les régissent et donc en disposer d'un modèle fidèle est difficile.
- **L'interprétation des résultats de simulation peut être difficile** dans certains cas.
- **La confrontation du système à la réalité sera toujours nécessaire** même quand un modèle est disponible, les modèles sont souvent incomplets et les tests physiques sont indispensables pour compléter les modèles.

Par la présente étude de l'état de l'art, nous proposons une liste courte de questions ouvertes entrainant plusieurs perspectives:

- Comment les normes peuvent évoluer pour cadrer la validation des nouvelles technologies complexes de perception d'environnement, de décisions critiques par la simulation ? Et comment les adapter pour viser la validation d'un sous-système en particulier (Navigation ou HMI ou Connectivité etc…)

- Comment valider les exigences et objectifs de sécurité (nombre de kilomètres traversés sans accident, nombre de scénarios testés …) indiqués par les normes en se servant de la

simulation ? Comment décliner ces objectifs pour assurer leurs prises en compte au niveau modèle et plus tard dans les tests et essais physiques ?

- Quel niveau de fidélité du modèle doit imposer les normes pour avoir confiance dans les résultats de simulation ? Et comment quantifier un tel niveau de fidélité ? Est-ce que les approches à base de scénarios et des critères de couverture peuvent s'avérer utile ?

- Comment juger la pertinence d'un outil de simulation ? Lorsque la simulation d'système complexe nécessite d'interfacer plusieurs outils de simulation à la fois (exemple : Scanner studio pour modéliser l'environnement et Matlab Simulink pour les lois de contrôle commande et de décision), quels cadres normatifs appliquer pour de telles interfaces ?

- Les véhicules autonomes nécessitent de la communication V2X, comment imposer un cadre normatif quant au protocole, moyen de communication et l'infrastructure nécessaire ?

9. Références

Etats-Unies, D. d. (s.d.). *Architecture Reference for Cooperative and Intelligent Transportation, .* Récupéré sur ARC-IT: https://local.iteris.com/arc-it/index.html

IRT-SystemX. (s.d.). *Méthodes et outils de modélisation et de simulation, Projet SVA, ISX-TA-SVA-LIV-0495.*

IRT-SystemX, Projet SVA. (s.d.). *L11.2 Choix des formats de descriptions, Projet SVA.*

M. Brini, E. A. (2020). *Position Paper PFA: SAFETY ARGUMENTATION FOR AUTOMATED VEHICLE.*

Menzel, T. B. (2018). Scenarios for Development, Test and Validation of Automated Vehicles. . *.arXiv preprint arXiv:1801.08598.*

SaFAD. (2019). *Safety First for Automated Driving.*

V-Model. (s.d.). Récupéré sur Wikipedia: https://en.wikipedia.org/wiki/V-Model