


THESE DE DOCTORAT DE
L'UNIVERSITE PIERRE ET MARIE CURIE

SPECIALITE

SCIENCES POUR L'INGENIEUR

PRESENTEE PAR

M. FABIEN BONNEFOI

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

SUJET DE LA THESE :

VÉRIFICATION FORMELLE DES SPÉCIFICATIONS DE SYSTÈMES COMPLEXES PAR RÉSEAUX DE PETRI: APPLICATION AUX SYSTÈMES DE TRANSPORT INTELLIGENTS

SOUTENUE LE 27 SEPTEMBRE 2010 DEVANT LE JURY COMPOSE DE :

Nom	Qualité	Titre
Fabrice KORDON	Directeur	Professeur à l'Université P. & M. Curie
Yvon KERMARREC	Rapporteur	Professeur à l'École Nationale Supérieure des Télécommunications de Bretagne
Jean-Claude ROYER	Rapporteur	Professeur à l'École des Mines de Nantes
Christine CHOPPY	Examinateur	Professeur à l'Université de Paris-Nord
Béatrice BÉRARD	Examinateur	Professeur à l'Université P. & M. Curie
Guy FREMONT	Examinateur	Innovative solution manager, S.A.N.E.F.
Franck POMMEREAU	Examinateur	Professeur à l'Université d'Évry

Université Pierre & Marie Curie - Paris 6
Bureau d'accueil, inscription des doctorants
Esc G, 2ème étage
15 rue de l'école de médecine
75270-PARIS CEDEX 06

Tél. Secrétariat : 01 44 27 28 10
Fax : 01 44 27 23 95
Tél. pour les étudiants de A à EL : 01 44 27 28 07
Tél. pour les étudiants de EM à MON : 01 44 27 28 05
Tél. pour les étudiants de MOO à Z : 01 44 27 28 02
E-mail : scolarite.doctorat@upmc.fr

Vérification formelle des spécifications de systèmes complexes par Réseaux de Petri: application aux Systèmes de Transport Intelligents

Fabien Bonnefoi

27 septembre 2010

"As far as the laws of mathematics refer to reality, they are not certain,as far as they are certain, they do not refer to reality"
Albert Einstein

Résumé

Les Systèmes de Transport Intelligent (STI) pour la route sont indispensables à la gestion du trafic et à la sécurité des usagers. Complexes et critiques, leur comportement dépend de nombreux phénomènes continus et leur conception nécessite la collaboration de nombreuses équipes. La vérification formelle de ces systèmes pose différents problèmes d'intégration au cycle de développement, d'explosion combinatoire, de vérification des phénomènes continus et de coût de production des modèles.

En utilisant le STI SAFESPOT comme cas d'étude, cette thèse propose différentes solutions aux problèmes d'utilisation des méthodes formelles au cours du cycle de développement d'un système complexe. D'abord en proposant une méthode de génération de modèles formels à partir des spécifications du système. Nous présentons un ensemble de règles de transformation permettant de produire des modèles formels (en Réseaux de Petri) à partir des spécifications semi-formelles du système (des diagrammes UML). Cela permet de minimiser le coût de production des modèles formels tout en favorisant une bonne communication entre les différents acteurs d'un projet.

Différents aspects de composition du modèle formel sont ensuite étudiés afin de permettre une approche modulaire. Grâce à l'élaboration d'un patron générique pour notre modèle formel, différents scénarios de composition et d'analyse peuvent être effectués à moindre coût tout en réduisant les problèmes d'explosion combinatoire.

Enfin, nous proposons une technique d'intégration de phénomènes continus dans les modèles formels par discrétisation. Cette approche permet la production de modèles sur lesquels les propriétés de logique temporelle sont décidables. Nous présentons une méthode de calcul de la propagation des incertitudes liées à la discrétisation. Ces incertitudes peuvent ensuite être prises en compte en modifiant les modèles ou les propriétés à vérifier afin de garantir la validité des preuves produites.

Abstract

Intelligent Transportation Systems (STI) for the road are essential for traffic management and security of road users. Complex and critical, their behavior depends on many continuous phenomena and their design requires the cooperation of many teams. Formal verification of these systems raises several problems like integration into a development cycle, combinatorial explosion, verification of continuous phenomena and production costs of models.

Using SAFESPOT ITS as a case study, this thesis proposes different solutions to integrate formal methods in the development cycle of a complex system. First, by proposing a method for generating formal models from the system specifications. We present a set of transformation rules to produce formal models (in Petri nets) from semi-formal specifications of system (UML diagrams). This minimizes the cost of production of formal models while promoting good communication between project stakeholders.

Different aspects of composition of formal model are then studied to allow a modular approach. With the development of a generic pattern for our formal model, different composition and analysis scenarios can be performed at lower cost while reducing the problems of combinatorial explosion.

Finally, we propose a technique for integrating continuous phenomena in formal models by discretization. This approach allows production models on which the properties in branching-time logic is decidable. We present a method to compute propagation of uncertainties related to the discretization. These uncertainties can then be taken into account by modifying models or verification properties to ensure the validity of produced proofs.

Remerciements

Je tiens d'abord à remercier Fabrice Kordon et Guy Fremont qui m'ont donné leur confiance en me proposant d'effectuer cette thèse. Ils m'ont soutenu tout au long de mes travaux, ils ont su critiquer mes propositions de manière constructive et soutenue dans mes initiatives quand elles étaient pertinentes. C'est avant tout grâce à eux si j'ai mené à bien ces travaux.

Je tiens ensuite à remercier les équipes du Lip6 et en particulier l'équipe Move. Je n'oublierai jamais votre disponibilité et nos nombreuses discussions où, souvent autour d'un café, les idées étaient débattues simplement dans une optique d'enrichissement réciproque. Cet environnement m'a aidé à affiner ma démarche. Je rends donc ici hommage à Alban Linard, Jean-Baptiste Voron, Alexandre HAMEZ à l'humour salvateur. À Yann Thierry-Mieg et Denis Poitrenaud aux jugements scientifiques si justes et si cruels. On se sent tout petit à leurs côtés mais c'est justifié.

À Nicolas Gibelin et Jean-Luc Mounier, mes problèmes machines ou logiciels n'ont jamais été aussi agréables et pour la première fois de ma vie j'étais heureux de ne pas connaître le mot de passe admin de ma machine : bravo les gas !

Je tiens à rendre hommage à Lom Messan Hillah, mon premier compagnon de publication avec qui j'ai tissé des liens qui vont au delà du simple cadre professionnel et qui je l'espère dureront longtemps. Tu as su être à mes côtés dans les moments les plus durs, merci Lulu !

Enfin Claude Girault, maître Yoda des thésards, vous m'avez sauvé la mise dans la dernière ligne droite. J'aurais aimé vous rencontrer plus tôt. Merci du fond du coeur.

Je voudrais remercier l'équipe du Lipn et Christine Choppy, la rigueur 'old school' c'est la classe, merci !

Je voudrais remercier les équipes de Cofiroute et particulièrement la DSETI : Marie, Antonio, Franck, la bonne ambiance c'était grâce à vous. Je tiens à rendre un hommage tout particulier à Michelle Bounegab : je n'étais qu'un "jeune" étudiant (du moins dans ma tête) et tu m'as montré, non sans difficulté, le chemin pour devenir un professionnel responsable. Quelque chose que je n'aurais pas trouvé dans la bibliothèque d'informatique de l'université et d'une importance bien plus grande ! Allez, on se fait un plateau de fruits de mer ;-)

Ensuite, Nicolas Deckner, qui n'a jamais baissé les bras dans la production de nos 2000 lignes de code dans le foutoir du projet européen SAFESPOT. On a mis a l'amende pas mal d'équipes, bravo mec !

Je tiens aussi à remercier l'équipe COSSIB du projet SAFESPOT. Malgré quelques tensions inévitables dans un projet de cet ampleur j'ai beaucoup appris à vos côtés et vous n'avez jamais remis en question mes jeunes compétences en management. J'espère un jour travailler de nouveau avec vous. Merci Angela Spence, Tobias Schendzielorz, Filippo Visintainer, Francesco Bellotti, Dominique Vilain et les autres.

Merci aussi à notre grand chef à tous Roberto Brignolo, faudra un jour que vous m'expliquiez comment vous avez pu garder votre calme et votre bienveillance dans le management d'un projet de plus de 50 partenaires européens sur quatre ans... Respect !

Quand, au bout de la nuit, les forces me manquaient, Radio Nova avait le dont pour balancer la sauce qui vous reboost. Mes oreilles vous seront à toujours reconnaissantes.

Je tiens à remercier mes parents, mon frère, ma soeur et ma famille. Tout ce long chemin, c'est aussi grâce à vous.

Enfin, rien n'aurait de sens sans elle. Elle est mon jour, ma nuit, mes joies et mes peines. Ma plus belle histoire. Cette thèse t'es dédiée mon amour.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

Sommaire

Les spécifications des systèmes complexes sont la pierre angulaire de la maîtrise de leur fiabilité. Elles jouent à la foi le rôle d'outil de communication entre les différentes équipes impliquées dans le projet de développement et servent de référence pour l'implémentation du système.

L'utilisation des méthodes formelles est alors le meilleur moyen d'assister la conception et la validation de ces spécifications, mais pose encore différents problèmes de production des modèles, d'explosion combinatoire et de décidabilité lors de la vérification des évolutions continues du système.

Cette thèse propose des solutions pour la vérification formelle des systèmes complexes en introduisant une approche modulaire de production et de vérification des modèles ainsi qu'une méthode de prise en compte des évolutions continues du système.

Notre approche est présentée autour d'un cas d'étude tiré d'un projet industriel d'implémentation d'un système complexe et critique dans lequel j'ai eu la responsabilité de développer un de ses composants. Il s'agit d'un *système de transport intelligent* (S.T.I.) pour la route issu du projet Européen SAFESPOT [99, 20] : impliquant une cinquantaine de partenaires européens (entreprises ou laboratoires de recherche) sur une durée de quatre ans, ce projet a pour objectif d'étendre dans l'espace et le temps, la perception que les conducteurs ont de leur environnement. Cela est rendu possible entre autre grâce à l'utilisation d'une nouvelle technologie de communication sans fil adaptée aux contraintes de mobilité, le 802.11p [66], qui réduit les délais liés à l'échange d'informations entre les véhicules et l'infrastructure routière.

Cette thèse est le fruit d'une collaboration entre le Département des Systèmes d'Exploitation et des Techniques Innovantes de la société COFIROUTE, et l'équipe Modélisation et Vérification du Laboratoire d'Informatique de Paris 6. Nous présentons dans ce chapitre le contexte, les problématiques et objectifs de nos travaux.

1.1 Contexte : Systèmes coopératifs pour la route et vérification formelle

Les domaines où les systèmes informatiques et électroniques jouent un rôle crucial pour la sécurité des biens et des personnes sont de plus en plus nombreux. Ce sont, par exemple, les domaines de la médecine, du transport, de l'aérospatiale ou de la finance. Avant la mise en place et l'utilisation de ces systèmes critiques, il est indispensable de pouvoir évaluer leur fiabilité. Cette évaluation nécessite de soumettre le système à une série de tests et d'analyses qui prennent souvent la forme de simulations, de tests logiciels ou plus rarement d'une analyse formelle.

L'analyse formelle est pourtant le meilleur moyen de prouver tout ou partie de la logique sous-jacente d'un système ou d'un logiciel. Pourtant, parmi les dizaines de méthodes de développement utilisées de nos jours, rares sont celles qui en font usage, préférant la mise en place de procédures de tests ou de simulations qui analysent le système comme un élément extérieur suspecté de contenir des erreurs, plutôt que de raisonner directement sur sa logique interne.

Ce phénomène à première vue étonnant, relève de causes multiples. En dehors de quelques spécialistes, peu d'ingénieurs s'intéressent aux méthodes formelles et celles-ci ne sont que trop rarement enseignées dans les écoles et universités. À première vue abstraites et complexes à mettre en œuvre, elle sont souvent considérées comme une charge de travail supplémentaire.

En effet, de la fin des années 60 aux années 80, la méthode de vérification formelle la plus utilisée était basée sur la logique de Floyd-Hoare. Cette logique définit un ensemble d'axiomes et de règles d'inférences pour l'ensemble des instructions de base d'un langage de programmation impératif. En utilisant ces axiomes et ces règles d'inférences dans un système déductif formel, il est possible de produire manuellement des preuves de propriétés d'un programme. Cette approche nécessite la construction manuelle de preuves beaucoup plus longues et complexes que le programme initial. Bien que cette approche ait été utilisée avec succès, sa complexité d'utilisation a fait naître un ensemble de mythes concernant les méthodes formelles [58, 17], comme par par exemple : "*les méthodes formelles augmentent le coût du développement des systèmes*" "*elles ralentissent le processus de développement*", "*elle ne sont pas utilisées pour des systèmes réels et complexes*", "*elles ne sont pas outillées*", ou encore "*elles sont inutiles*".

Dans les années 80, E. Clarke, A. Emerson et J.Sifakis fondent une nouvelle approche de vérification des programmes : le "Model Checking". Utilisant un algorithme de preuve, cette méthode permet la production automatique de preuves concernant l'évolution d'un système dans le temps [93]. Les années 90 ont permis, grâce aux développement des méthodes de vérification formelle et particulièrement celles de Model Checking, de réfuter la majorité des mythes que nous avons évoqués.

Mais, pendant longtemps, l'utilisation des méthodes formelles s'est faite presque exclusivement dans les domaines de l'armement, de l'aérospatial, en cryptographie ou en sécurité informatique. Cela pour des raisons d'exigence en termes de fiabilité et de sécurité. Mais alors que les domaines où l'informatique joue un rôle essentiel en termes de sécurité sont de plus en plus nombreux et les systèmes de plus en plus complexes, les méthodes de développement s'adaptent et évoluent dans le sens d'une utilisation accrue des méthodes formelles.

Les systèmes de transport intelligents (STI) sont utilisés depuis plusieurs années dans l'aviation civile, le trafic ferroviaire ou l'exploitation des infrastructures routières. Ils accomplissent des tâches essentielles au maintien de la sécurité et à l'efficacité du trafic. Le développement de nouvelles technologies permet la multiplication des tâches prises en charge par ces systèmes, et l'amélioration de leur efficacité. Ces systèmes gagnent en complexité, ce qui rend difficile leur conception et l'évaluation de leur fiabilité.

Les STI constituent donc un cas d'étude intéressant pour l'élaboration et l'évaluation de nouvelles pratiques des méthodes formelles.

1.2 Problématique : Conception et vérification formelle des systèmes complexes

Les systèmes complexes impliquent souvent un grand nombre d'entités et des calculs exécutés en parallèle. De plus ces systèmes contiennent souvent des applications critiques pour les vies humaines. De telles contraintes requièrent une attention particulière lors de la conception et de l'implémentation. Mais alors que les systèmes complexes et critiques se multiplient, le coût lié à l'emploi des méthodes formelles est un obstacle à leur utilisation.

Les principales difficultés liées à l'utilisation efficace des méthodes formelles sont diverses.

- 1 : **La complexité d'un système rend la tâche de modélisation formelle coûteuse**. La vérification formelle de certaines propriétés nécessite la modélisation de l'ensemble du système. Constitué d'un grand nombre de composants, sa modélisation formelle devient longue et coûteuse. Cela représente un obstacle à l'utilisation efficace de ces méthodes et nécessite de développer une nouvelle approche basée sur des méthodes de génie logiciel.

- 2 : **La vérification des systèmes complexes engendre de l'explosion combinatoire**. Lors de l'analyse des modèles, en particulier lorsqu'il s'agit d'étudier les comportements dynamiques du système par *model checking*, la quantité et la complexité des modèles, des paramètres et des propriétés à vérifier engendrent de l'explosion combinatoire [49].

- 3 : **Des problèmes de décidabilité empêchent la vérification des phénomènes continus**. La majorité des techniques de vérification de modèle concerne les systèmes discrets et finis. Or, un grand nombre de propriétés des systèmes complexes reposent sur un ensemble de variables et paramètres continus, on parle alors de système *hybrides*. La gestion des systèmes hybrides n'est pas aisée [31] et mène souvent à l'élaboration de modèles non finis sur lesquels il n'est pas possible d'obtenir des résultats d'analyse dans le cas général. Les évolutions continues d'un système et de ses composants sont difficiles à vérifier même avec les langages formels possédant ce pouvoir d'expressivité. Soit le formalisme est trop rigide pour être adapté à la méthode de développement, soit il ne propose pas la modélisation des aspects continus. De plus, la vérification formelle des paramètres continus d'un système pose des problèmes de décidabilité.

- 4 : **La diversité des spécialités et des méthodes impliquées dans un projet de réalisation d'un système complexe rend difficile la mise en place d'une méthode de spécification.** Les projets de développement de systèmes complexes impliquent un grand nombre de partenaires et spécialistes dans des domaines variés. La mise en place du processus de spécification implique une sélection parmi une multitude de méthodes et de formalismes. C'est une tâche critique pour le bon déroulement du projet [87, 121]. L'utilisation des méthodes formelles représente alors un coût supplémentaire.

- 5 : **La cohérence entre différents formalismes n'est pas toujours garantie**. Les formalismes utilisés pour la conception, la spécification puis le développement des systèmes

ne sont pas les mêmes que ceux utilisés pour la modélisation et la vérification formelle. La relation entre ces formalismes n'est pas directe, et rien ne garantit la cohérence entre ces différentes représentations du système.

En 2007, J. Sifakis [41] présente un ensemble de perspectives pour palier aux problèmes liés à l'utilisation des méthodes de Model Checking.

- Garantir que le modèle représente *fidèlement* le système. Le terme "*fidèlement*" signifie ici que la relation entre le modèle et le système puisse être définie au travers d'une sémantique formelle vérifiable.
- Améliorer le passage à l'échelle. Les algorithmes de vérification actuels sont assez efficaces, mais souffrent tous de limitations liées à la complexité inhérente aux systèmes composés d'un nombre élevé de composants. Deux approche alternatives sont alors envisagées. La première consiste à calculer des propriétés sur le système à partir de propriétés calculées à partir de ses composants et de règles d'assemblages. La deuxième consiste à élaborer des preuves de propriétés autour d'architectures formelles concernant une classe de systèmes.

1.3 Objectifs : Aide à la modélisation et la vérification formelle dans un contexte industriel

Cette thèse propose un ensemble de solutions aux problèmes d'utilisation des méthodes formelles au cours du cycle de développement d'un système complexe.

1. Le premier objectif est de faciliter la production de modèles formels sans pénaliser la communication au sein d'un projet. Les langages de spécification semi-formels comme UML ont l'avantage d'être simples à comprendre, ce qui favorise une bonne communication entre différentes équipes au sein d'un projet. Mais ils ne permettent pas d'effectuer des analyses formelles du système. Les modèles formels eux, sont plus compliqués à interpréter et sont moins adaptés à une communication efficace au sein d'un projet. Notre approche est donc de définir un ensemble de règles de transformation permettant de générer les modèles formels du système à partir de spécifications semi-formelles. Cela permet de produire des modèles formels à moindre coût sans entraver la communication entre les différentes équipes du projet.

2. Le deuxième objectif est d'accompagner le processus de spécification en produisant des modèles formels à différents niveaux d'abstraction. Il est ainsi possible de valider les choix faits aux différentes étapes de la spécification du système.

3. Le troisième objectif est de permettre l'analyse formelle de différentes stratégies d'implémentation du système à moindre coût. Notre approche est de définir une architecture d'assemblage de composants formels. Il est ainsi possible de définir différentes versions des composants formels, de les analyser séparément avant de les assembler. Cette approche permet aussi la réutilisation de certains composants réduisant ainsi le coût

de production du modèle formel complet. L'étude de plusieurs projets de Systèmes de Transport Intelligents nous permet de définir une architecture formelle pour cette classe de système au sein de laquelle la réutilisation de composants formels est possible.

4. Enfin, notre dernier objectif est de permettre la modélisation des paramètres et des fonctions continues ayant un impact sur le comportement du système tout en conservant la capacité d'analyse de propriétés exprimées en logique temporelle. Pour ce faire nous adoptons une approche par discrétisation des aspects continus tout en calculant l'intervalle de confiance associé au modèle formel discrétisé.

Notre approche est testée sur un cas d'étude tiré d'un projet industriel d'implémentation d'un système complexe. Il s'agit du projet SAFESPOT [99, 20] : impliquant une cinquantaine de partenaires européens (entreprises ou laboratoires de recherche) sur une durée de quatre ans, ce projet a pour objectif d'étendre dans l'espace et le temps, la perception que les conducteurs ont de leur environnement.

Nous avons, au sein de ce projet, conçu et implémenté l'application d'alerte aux conducteurs en cas d'obstacles sur la route, qui utilise les dernières innovations dans le domaine des systèmes d'alerte aux conducteurs.

- Une approche coopérative (pair-à-pair) entre l'infrastructure et les véhicules permet d'étendre et d'améliorer la diffusion d'informations aux conducteurs [20].
- Une nouvelle technologie de communication permet de réduire considérablement les temps de connexion entre des véhicules circulant à des vitesses supérieures à 200km/h [66].
- De nouvelles bases de données événementielles et temps réel basées sur la fusion d'informations rendent plus rapide l'exploitation de données fiables, mais la rendent aussi plus complexe [6].

Au cours de ce projet, des tests sur pistes d'essais ont été effectués (voir Annexe A.1 page 151) avant une démonstration en situation réel de trafic. Le système SAFESPOT est alors déployé coté infrastructure dans une borne présentée figure 1.1. Le système est aussi déployé dans des véhicules de tests équipés de l'ensemble des composants nécessaires au fonctionnement du système, un exemple est présenté figure 1.2

1.4 Plan du mémoire

Chapitre 2 : État de l'art et cas d'étude Dans ce chapitre nous présentons le contexte de cette thèse : la conception des systèmes complexes, critiques, et leur vérification formelle. Puis, nous détaillons les problématiques liées à cette activité. Enfin, nous présentons notre cas d'étude : le système SAFESPOT.

Chapitre 3 : Présentation des contributions Dans ce chapitre, nous présentons notre technique de vérification formelle des spécifications des systèmes complexes et critiques. Nous expliquons d'abord quelles sont les contraintes liées à la spécification des système complexes.

FIGURE 1.1 – La borne de l'infrastructure

FIGURE 1.2 – Le véhicule équipé

Puis nous présentons les différents niveaux d'abstraction qui nous ont permis de mettre en place notre technique : cas d'utilisation, architecture, composants et algorithmes.

Chapitre 4 : Élicitation des besoins utilisateur Le cahier des charges fonctionnel (CdCF) est la première ébauche du système où sont définies ses principales caractéristiques et fonctionnalités. Il joue un rôle essentiel en début de projet en formalisant les choix de réalisation et les responsabilités des différents acteurs du projet.

Ce chapitre présente d'abord les problèmes posés par l'analyse de l'accidentologie européenne. Celle-ci sert de base à l'analyse des besoins des utilisateurs. Nous présentons ensuite comment nous l'avons effectuée au sein du projet SAFESPOT. Enfin, une méthode pour la conception des cas d'utilisation est présentée. Elle est fondée sur notre concept de scénarios accidentogènes

Chapitre 5 : Analyse formelle des spécifications de STI Des modèles semi-formels de spécification de l'architecture du système et de son comportement peuvent être utilisés pour produire des modèles formels.

Dans ce chapitre, nous présentons un ensemble de règles de transformation permettant de produire des Réseaux de Petri à partir de diagrammes UML. Grâce à l'élaboration d'un patron générique pour notre modèle formel, il est possible de construire notre modèle formel par l'assemblage de composants formels. Les composants formels peuvent être vérifiés avant d'être assemblés. Différents scénarios d'analyse peuvent alors être effectués à moindre coût en testant différentes versions d'un même composant.

Chapitre 6 : Vérification formelle des contraintes continues par discrétisation Enfin, cette thèse propose une méthodologie d'intégration de phénomènes continus basée sur la dis-

crétisation de Réseaux de Petri Colorés en Réseaux de Petri Symétriques. La méthodologie présentée offre une solution originale qui traite les problèmes de propagation d'erreurs lors de la discrétisation des aspects continus, et les problèmes d'explosion combinatoire qui y sont liés.

Chapitre 2

État de l'art et cas d'étude

Sommaire

Les *"Systèmes de Transport Intelligents"* (STI) pour la route ont un rôle de plus en plus important dans nos sociétés en assurant la bonne gestion du trafic routier. Grâce à l'utilisation des technologies de l'information et de la communication, ils permettent une meilleure gestions des flux, la détection automatique d'incidents ainsi que le recueil et la diffusion d'informations concernant l'état du trafic. Ils accomplissent de nombreuses tâches liées par exemple à la sécurité ou à la gestion des péages automatiques. Leur développement s'accélère depuis un vingtaine d'années, et les différents systèmes mis en place communiquent difficilement entre eux. Ceci n'est pas surprenant quand on sait que les standards et les technologies sur lesquels ils reposent varient d'un système à l'autre.

La mise en place progressive de mécanismes de coopération permet d'augmenter l'efficacité et la fiabilité de ces systèmes, ouvrant la voie à de nouveaux services. Par exemple, en améliorant la disponibilité des informations dans l'espace et dans le temps, la sécurité des usagers peut être renforcée. Mais la coopération des STI pour la route implique l'élaboration de "systèmes de systèmes" dont le comportement devient difficile à analyser et à vérifier, rendant difficile la garantie de la fiabilité du système résultat.

Le développement des STI pour la route n'aurait pu se faire aussi rapidement sans l'amélioration des outils et pratiques de développement. Ces avancées permettent aux concepteurs et aux développeurs de se détacher progressivement des contraintes liées à l'utilisation des outils et pratiques de développement pour mieux se concentrer sur les services et la nature du programme qu'ils doivent développer. Ils permettent aussi l'automatisation de certaines tâches comme la génération de code source ou encore l'analyse du système.

Ce chapitre présente le contexte et le cas d'étude de cette thèse. Dans un premier temps nous présentons les Systèmes de Transport Intelligents (STI) pour la route, leurs différents aspects et leur dynamique de développement aux cours de ces dernières années. Puis nous présentons différentes avancées dans le domaine de l'ingénierie des systèmes informatiques et de la vérification formelle. Enfin nous concluons par une présentation du cas d'étude qui a été utilisé pour nos différentes expérimentations.

2.1 STI pour la route

Les STI [1] et l'ingénierie des systèmes sont des domaines d'activité qui s'enrichissent mutuellement. D'un côté, les STI profitent des dernières innovations liées à la détection vidéo, la géo-localisation ou encore les télécommunications. Ils coopèrent de plus en plus, formant des systèmes de systèmes aux structures et aux comportements complexes. D'un autre côté, l'évolution des outils et méthodes de développement offre les abstractions et mécanismes adaptés à la conception de systèmes de plus en plus complexes.

Cette section présente le contexte de cette thèse. Dans un premier temps nous présentons les Systèmes de Transport Intelligents pour la route, leurs missions, les différentes formes qu'ils

1. Afin de faciliter la lecture, nous utilisons parfois l'acronyme "STI" pour désigner les "*Systèmes de Transport Intelligents pour la route*". Bien que beaucoup de ce qui se rapporte aux STI pour la route soit aussi valable pour les STI en général, ça n'est pas toujours le cas.

peuvent prendre et la dynamique de développement qui les anime. Puis nous présentons l'ingénierie des systèmes informatiques, son rôle, ses dernières avancées et ses liens avec les STI pour la route.

2.1.1 Définition

Le transport des personnes et des biens joue un rôle central dans le développement de nos sociétés, que se soit pour la maîtrise des énergies, des ressources naturelles, la sécurité ou encore l'activité économique.

La Commission Européenne estime par exemple que le secteur des transports et les industries qui y sont liées couvrent 7% du PIB européen, et emploient environ dix millions de personnes [100, 67] .

Définition 2.1.1. *Le terme Système de Transport Intelligent (STI) (en anglais ITS pour Intelligent Transportation Systems) désigne les systèmes qui utilisent les technologie de l'information et de la communication dans le domaine des transports. Depuis longtemps indispensables dans la gestion des trafics aérien et ferroviaire, les STI connaissent depuis quelques années un formidable développement dans le domaine du transport routier, principalement grâce au développements de la géo-localisation, des moyens de communications et des solutions logicielles.*

2.1.2 Les missions des Systèmes de Transport Intelligents pour la route

La sécurité : Selon l'OMS [115], les accidents de la circulation sont la principale cause de décès chez les 10-24 ans : près de 400 000 jeunes de moins de 25 ans sont tués dans des accidents de la circulation chaque année et des millions d'autres blessés ou handicapés. Les chiffres collectés par les Nations Unies entre 1994 et 2004 font état de 4 millions d'accidents par an dont 150 000 accidents mortels en Europe et en Amérique du Nord [80]. Grâce à une détection rapide des événements dangereux et à des mécanismes d'alerte ou de contrôle des véhicules, la mise en place de STI liés à la sécurité permet de réduire le nombre d'accidents [112] .

La gestion du trafic routier : Selon les études prospectives menées en France en 2000 par la *Direction des Routes*, la croissance du trafic routier a augmenté de 230% en 25 ans. Les taux correspondants pour les 15 années suivantes devaient être supérieurs à 50% que ce soit pour les voyageurs ou les marchandises [108]. Les crises économiques de 2001 et 2007 ont légèrement réduit cette perspective de croissance qui reste néanmoins importante [91]. Une meilleure gestion des flux grâce aux STI permet de réduire les embouteillages et de repousser la construction de nouvelles routes dont le prix est élevé et dont l'impact environnemental est non négligeable [112]. Il convient néanmoins de rappeler que l'amélioration de la fluidité sur le réseau routier le rend plus attractif, ce qui participe finalement à l'augmentation du trafic.

La gestion du réseau et son exploitation : Pour assurer l'entretien en condition de fonctionnement, assurer la communication avec les utilisateurs et les organisations extérieures, ou encore la gestion des pannes ou des accidents, il est nécessaire d'utiliser des systèmes d'assistance adaptés. Ces systèmes sont utilisés par les *exploitants* d'infrastructures de transport et sont appelés *Systèmes d'Aide à l'Exploitation* (SAE). Ces systèmes sont indispensables pour une gestion efficace d'une infrastructure de transport. On estime par exemple qu'un SAE pour

la route capable de détecter l'occurrence d'un accident et d'alerter les usagers (par des panneaux à messages variables par exemple) permet de réduire de 30% à 10% la part des accidents en chaîne [2] sur le total des accidents [116].

Le respect de l'environnement : Préserver notre environnement est devenu une préoccupation majeure, et l'impact environnemental du transport routier peut être réduit grâce aux transports intelligents. En permettant une meilleur gestion des flux, ils permettent de réduire la durée et la fréquence des embouteillages, diminuant ainsi à la fois l'énergie consommée et les gaz émis dans l'atmosphère [44].

2.1.3 Les différentes formes de STI

De l'Europe à la Chine en passant par les États Unis d'Amérique et le Japon, de nombreux projets concernent le développement des systèmes de transport intelligents pour la route. Il serait dérisoire de vouloir brosser un tableau exhaustif des innovations et STI ayant vu le jour ces dernières années, mais les exemples suivants donne une idée de l'importance du sujet : les États-Unis, le Mexique et le Canada utilisent désormais un système de suivi des véhicules de marchandises sensibles ou dangereuses grâce aux communications par satellites ; le centre de Londres est équipé d'un système de péage sans barrière basé sur des communications courte portée et de la détection vidéo ; les mesures du trafic de la ville de Pékin et de sa banlieue sont basées sur les remontés automatiques d'informations des taxis en temps réel. Sur de nombreuses routes la régulation du trafic est basée sur des limitations de vitesse variables et des péages à tarification évoluant en fonction de la demande. Enfin, certains véhicules sont capables de détecter les sorties de route, les excès de vitesse ou de faire du suivi de véhicule puis de freiner automatiquement en cas de danger. Les STI prennent donc des formes et des architectures variées en fonction des missions qu'ils doivent remplir, et font appel à différents types de technologies.

Les éléments que nous venons de présenter donnent une idée de l'importance des enjeux sociaux et économiques liés aux transports routiers. A l'instar des États Unis, du Japon ou de nombreux autres pays, l'Union Européenne promeut et finance le développement des STI. En 2006 par exemple, la Commission Européenne lance plusieurs projets de STI pour la route au travers de son sixième "*Programme Cadre pour la Recherche et le Développement*" (i.e. le 6ème PCRD). Les trois plus gros projets de STI de ce 6ème PCRD cumulent un budget supérieur à quatre-vingt quinze millions d'euros. Leur objectif est d'améliorer la coopération entre les systèmes développés dans les véhicules et ceux mis en place dans les infrastructures du réseau routier, et de poser ainsi les bases des futurs systèmes de sécurité pour la route.

2.1.4 La dynamique de développement des STI

La stratégie de recherche et de développement pour les systèmes de transport intelligents pour la route en Europe illustre des stratégies mises en place de nos jours. Elle suit deux axes principaux.

2. Un véhicule accidenté représente un danger pour les autres usagers et pourrait provoquer de nouveaux accidents. On parle alors d'accidents en chaîne.

Le premier consiste à harmoniser les technologies et les normes utilisées pour la conceptions des STI dans les différents pays de l'Union grâce aux projets Euro-régionaux. Ces projets favorisent l'utilisation de technologies permettant les échanges d'informations à l'échelle européenne.

Le deuxième axe s'articule autour de projets de recherche dont les objectifs sont l'élaboration de systèmes de sécurité, la mise en place de services d'information aux usagers, la régulation du trafic, le développement de technologies de communication ou encore l'élaboration de couches logicielles (des *framework*) adaptées aux problématiques de la route.

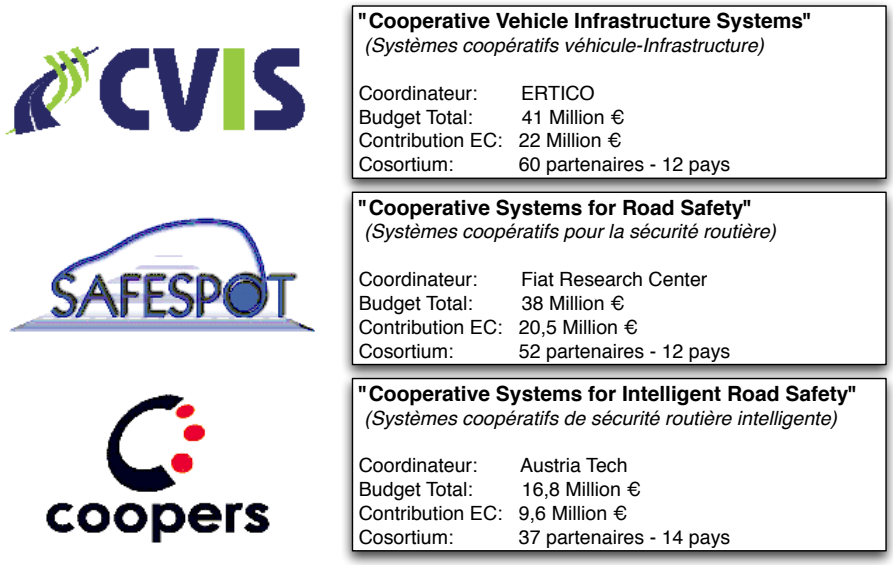


FIGURE 2.1 – Les *Projets Intégrés* (IP) européens du 6èmePCRD. Période du 01/02/2006 au 31/12/2010.

Parmi ces projets, trois mettent en place des systèmes coopératifs avec pour objectif l'amélioration de la sécurité routière et l'efficacité du trafic : CVIS (Cooperative Vehicle Infrastructure Systems) [96], SAFESPOT (Cooperative systems for Road Safety) [20] et COOPERS (COOPerative systEms for Intelligent Road Safety) [95]. Un aperçu de leur ampleur par leurs budgets et la taille de leurs consortiums, est présenté figure 2.1.

Ces projets s'inscrivent dans une stratégie européenne de prise en charge de la sécurité du conducteur à différentes échelles de temps : de la planification de son trajet à la gestion en temps réel des dangers sur la route. Au total, c'est un ensemble de six projets de STI différents qui permet cette approche. En voici la présentation :

– Les projets CVIS et COOPERS ont pour objectif d'améliorer l'efficacité et la sécurité du réseau routier en aidant le conducteur à *choisir un itinéraire* évitant les routes présentant un danger temporaire (comme la présence d'un accident ou un problème d'adhérence). Les recommandations fournies concernent les prochaines minutes du trajet.

– SAFESPOT se concentre lui sur la *détection des situations de danger potentiel* et fournit en temps réel des alertes au conducteur. Son objectif est de permettre au conducteur de réagir à un événement imprévu pouvant survenir dans la prochaine minute.
– Le projet PREVENT a pour objectif d'améliorer la sécurité en *contrôlant le véhicule* quand le conducteur n'a pas le temps de le faire lui-même. Par exemple pour éviter des manœuvres dangereuses ou en prenant le contrôle des freins pour un arrêt d'urgence. Ce projet travaille sur une échelle de temps inférieure à 10 secondes.
– Les systèmes APROSYS et eCall ont été conçus pour *atténuer les conséquences d'un accident* en resserrant les ceintures de sécurité juste avant une collision, ou en envoyant un signal d'alerte aux services d'urgence après un incident.

Cette approche européenne de gestion de la sécurité implique donc plusieurs systèmes à différentes étapes d'un trajet et sur différentes échelles de temps. C'est ce qui est illustré sur la figure 2.2.

Les projets CVIS, SAFESPOT, COOPERS et plus généralement ceux présentés sur la figure 2.2 sont complémentaires. La solution technologique adoptée pour permettre la coopération de ces systèmes est l'architecture CALM [12]. Cette architecture permet une communication véhicule vers infrastructure (V2I), infrastructure vers véhicule (I2V), véhicule vers véhicule (V2V). Elle permet aussi un accès continu à Internet via différentes technologies, qui peuvent être utilisées séparément ou simultanément en fonction de la couverture radio disponible [3].

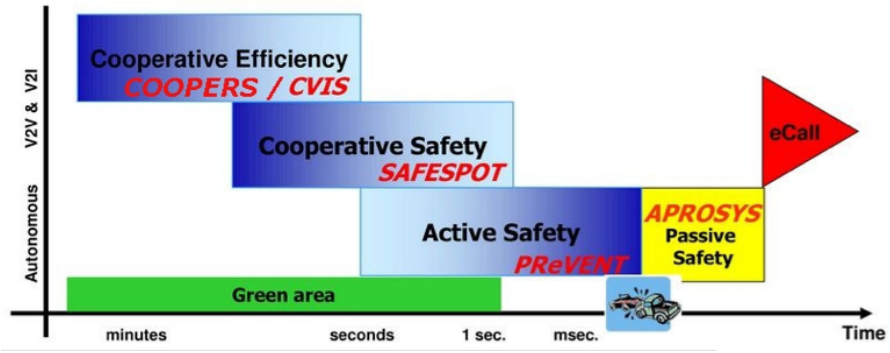


FIGURE 2.2 – Les projets européens de transports coopératifs et la gestion des accidents.

Le développement continu de nouvelles technologies permet la multiplication des tâches prises en charge par les STI et l'amélioration de leur efficacité. Mais cela implique des systèmes plus complexes et rend leur conception et leur réalisation difficiles. Depuis les années 1970, la complexité des systèmes informatiques et électroniques est identifiée comme la source principale des difficultés liées à leur réalisation. Depuis, de nombreux efforts sont investis dans l'amélioration des pratiques et méthodes de conception et réalisation de ces systèmes. C'est ce que nous présentons dans la section suivante.

3. Le développement et l'évaluation de l'architecture CALM sont des objectifs du projet CVIS.

2.2 L'ingénierie des systèmes

La complexité des gros systèmes croît plus vite que notre capacité à les appréhender [81]. C'est en particulier le cas des systèmes répartis ou l'exécution simultanée de plusieurs flots d'exécution rend extrêmement difficile l'élaboration de techniques de test fiables. La complexité des systèmes représente un obstacle à leur développement et à leur fiabilité. On parle alors de la *crise du logiciel*. Ce concept évoqué pour la première fois en 1968 par F.L.Bauer [102] est toujours d'actualité.

Définition 2.2.1. *Un système complexe est défini de manière informelle par quelques propriétés :*

1. *un grand nombre d'entités et d'interactions locales et simultanées,*
2. *des décisions décentralisées (locales),*
3. *des boucles de rétroaction (l'état d'une entité a une influence sur son état futur via d'autres entités),*
4. *il est ouvert et soumis à un extérieur, de sorte qu'un observateur ne peut en prévoir l'évolution par un raccourci de calcul.*

2.2.1 La crise du logiciel

Les problèmes liés à la complexité croissante des systèmes et logiciels ne sont pas nouveaux. L'histoire de l'informatique est depuis toujours liée à l'apparition de nouveaux modèles et méthodes de développement repoussant les limites de la conception des systèmes complexes. Ces innovations, liées à l'évolution des capacités de calcul et de communication, continuent de stimuler l'imagination des concepteurs, et les projets semblent aller vers toujours plus de complexité. Ainsi, les causes de la "crise du logiciel" identifiées dans les années 70 sont encore d'actualité [49].

De nombreux rapports et publications attestent que la complexité de systèmes croît [21], que les problèmes de conception et de réalisation liés à cette complexité persistent [25, 87, 52], que les méthodes de développement actuelles n'apportent que des solutions partielles [9]. Ces problèmes sont d'autant plus gênant pour les systèmes ayant un rôle en terme de sécurité : systèmes de contrôle d'une centrale nucléaire ou de guidage des avions. Les Systèmes de Transport Intelligents (STI) sont un exemple de tels systèmes dont il est difficile de garantir la fiabilité de leur fonctionnement.

2.2.2 De l'intérêt de l'ingénierie des systèmes

De nombreuses études soulignent qu'une bonne ingénierie des systèmes amène une réduction des coûts et des délais de réalisation d'un système. Par exemple, le *"International Council of Systems Engineering"* (INCOSE) a mesuré les coûts prévus et effectifs des projets et la part investie en ingénierie [64]. Pour les 44 projets analysés, l'étude montre une augmentation moyenne des coûts de 50% sans ingénierie, et une meilleur maîtrise des coût quand l'investissement en ingénierie s'approche des 20% du budget total du projet. C'est ce qui est illustré sur la figure 2.3 où l'on peut voir en bleu la distribution moyenne du rapport entre le taux de dépassement de budget et le pourcentage de budget investi en ingénierie.

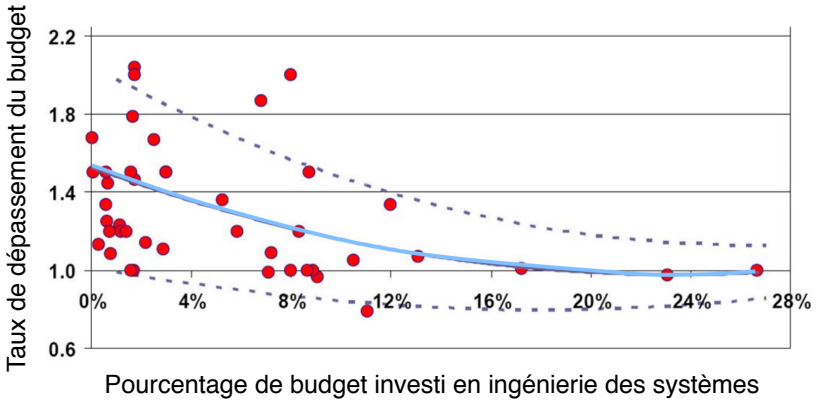


FIGURE 2.3 – L'ingénierie des systèmes améliore la maîtrise des coûts des projets informatiques

2.2.3 Les corrections tardives coûtent plus cher

Les projets rencontrent toujours à un moment ou à un autre des problèmes ou erreurs qu'ils faut traiter. Si ce n'est pas une erreur, c'est un changement de directives ou d'objectifs qu'il va falloir prendre en compte. Le problème est qu'une erreur ou une fonctionnalité manquante détectée à la phase de réalisation coûtera toujours plus cher à corriger qu'au moment de la rédaction du cahier des charges.

Des études montrent que les délais de détection et de gestion des erreurs augmentent le coût de leur prise en charge et de leur correction de manière dramatique [94]. Le principe illustré figure 2.4 [86] est par exemple qu'une mauvaise contrainte de fonctionnement sera simple à corriger à la phase de rédaction des contraintes, mais coûtera de plus en plus cher à mesure que le projet avance, en suivant une évolution exponentielle. En effet, tant qu'une erreur n'est pas corrigée, son impact se propage à chaque nouvelle étape du développement.

2.2.4 Cycles de développement

Une *méthode de conception* est constituée :
- d'un *cycle de développement* qui décrit l'enchaînement et la nature des différentes étapes de production,
- et de différents *formalismes* qui sont utilisés, par exemple, pour la spécification et d'implémentation.

Le *cycle de développement* d'un système se décompose en plusieurs étapes : de l'analyse des besoins à la livraison du produit, intégrant même parfois les étapes de mise à jour du système. Voici les principale étapes communes à la majorité des cycles de développement :

1. l'analyse des besoins des utilisateurs et des contraintes de fonctionnement,
2. la spécification,
3. la réalisation (ou production),
4. les tests et vérifications,

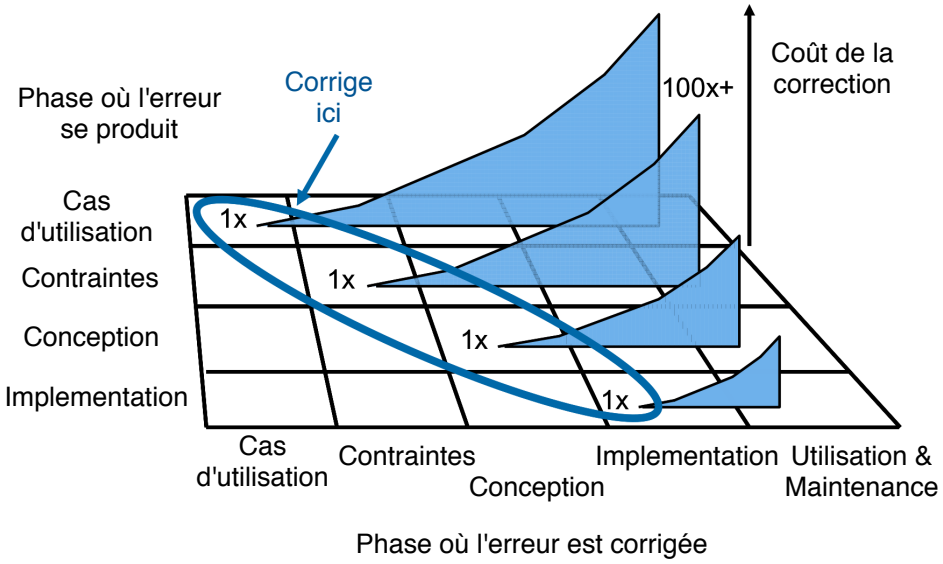


FIGURE 2.4 – Coûts des corrections aux différentes étape du développement

5. la validation [4],
6. la livraison et la maintenance.

Chacune de ces étapes peut être encore décomposée. Il est par exemple courant que la spécification d'un système se décompose en 1) la spécification de l'architecture 2) les spécifications détaillées du système. De plus, l'enchaînement de ces étapes et les relations qu'elles ont entre elles varient d'un projet à l'autres.

Cycle en cascade : Le cycle de développement des systèmes électronique et informatique s'est d'abord inspiré des pratiques utilisées pour la construction des bâtiments. Dans ce secteur, les étapes de conception et de réalisation doivent se succéder les unes après les autres, et il n'est pas possible de commencer les fondations d'un bâtiment tant que ses plans ne sont pas terminés et ainsi de suite. Chaque étape doit être validée avant de pouvoir passer à la suivante et chaque retour en arrière dans le cycle de développement est compliqué et coûteux. On parle alors de cycle en cascade (présenté à gauche sur la figure 2.1).

Progressivement, de nouvelles approches sont apparues intégrant les problématiques liées au développement des logiciels. Sans en faire une présentation exhaustive, on peut identifier deux famille principales : celle du cycle en "*V*" et celle du développement "*en spirale*".

Cycle en "V" : L'idée sous-jacente est que le développement d'un logiciel se fait d'abord par une décomposition progressive du problème initial jusqu'à la production du code. Puis la production du système se fait par recomposition progressive. Chaque étape de décomposition correspond à une étape de recomposition, donnant au cycle de production une forme en "*V*" (au centre sur la figure 2.1). Cette approche permet aussi, à chaque étape de décomposition, de préparer l'étape de recomposition correspondante. Par exemple, lors de l'étape d'*analyse des*

4. Le terme de "validation" fait ici référence à une évaluation du système vis-à-vis du cahier des charges en impliquant les clients et les utilisateurs du système. Il ne faut pas le confondre avec le même terme utilisé dans le contexte de la vérification formelle, qui consiste à tester des propriété formelles sur un modèle.

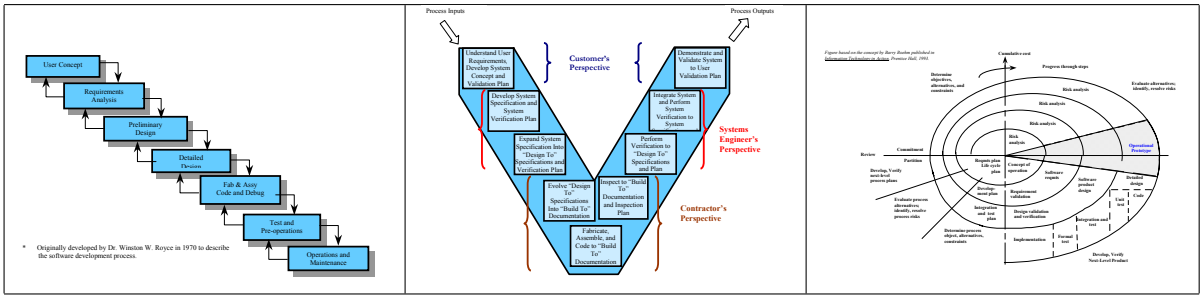


TABLE 2.1 – Les principaux modèles de développement : en cascades, en V et en spirale

besoins utilisateur, sont produit les scénarios de *validation par l'utilisateur*.

Cycle en spirale : Le *cycle en spirale* propose de commencer les projets par une itération rapide du cycle en"V" afin de produire rapidement un prototype partiel du projet. Puis, plusieurs itérations en "V" de plus en plus longues sont effectuées jusqu'à la production du système final. L'intérêt de cette approche est d'abord qu'elle permet de détecter des problèmes potentiels à chacune des étapes du cycle. Elle permet aussi d'impliquer plus fortement les différentes parties prenantes du projet comme les clients ou les utilisateurs. Elle nécessite néanmoins une coopération accrue des équipes impliquées dans le projet, ce qui peut s'avérer problématique par exemple pour les projets aux dimensions internationales.

2.2.5 Méthodes de développement pour STI

Les *méthodes de développement* décrivent les différentes étapes du développement, les formalismes à utiliser, les interactions possibles entre les différentes étapes et leur succession dans le temps. On distingue souvent les *cycles de développement*, qui décrivent l'enchaînement des différentes étapes, des *formalismes* utilisées pour décrire les productions de ces différentes étapes (textes, tableaux, modèles ou codes).

Des méthodes spécifiques aux STI ont vu le jour pour assister le développement des Systèmes de Transport Intelligents. On peut citer par exemple l'approche européenne *FRAME* [97] ou l'approche américaine : *National ITS Architecture* [119].

L' architecture de modélisation pour les STI européens (FRAME) Cet outil regroupe un certains nombre d'exemples et de recommandations à adopter lors de **la spécification** d'un système de transport intelligent. Ces recommandations sont élaborées autour d'un langage de spécification : le langage FRAME. Différentes bases de données de besoins utilisateurs ou de contraintes de fonctionnement liées aux STI sont aussi proposées. Il est ainsi possible de sélectionner les modèles ou les contraintes de fonctionnement parmi ceux qui sont proposés pour spécifier un STI.

Frame définit une norme et un langage commun pour la spécification des STI européens. La sémantique du langage concerne les aspects statiques de composition du système, mais la composition interne des block fonctionnels n'est pas abordée. C'est une bonne approche car cela rend la méthode facile à utiliser dans une première phase de modélisation. Regrettons cependant l'absence d'outils d'analyse et de vérification des modèles élaborés par cette méthode.

Le guide de développement américain pour les STI Contrairement à l'approche européenne, le guide de développement américain [119] met l'accent sur le cycle de développement plutôt que sur le langage de spécification.

Basée sur un cycle en "V", la méthode décrit les différentes caractéristiques de chacune des étapes de développement. Elle détaille le rôle de acteurs impliqués dans la réalisation du STI : utilisateur, client, concepteur ou développeur ainsi que leurs relations avec les étapes de développement. Elle explique aussi les éléments à produire et à valider à chaque étape du cycle. Sans proposer de langage de spécification ou de norme à suivre, elle met l'accent sur la gestion d'un projet de STI pour faciliter son élaboration par plusieurs partenaires.

2.3 Développement basé sur les modèles

Afin de maîtriser la complexité croissante des systèmes, il sont conçus sous la forme d'un ensemble de modèles. On parle alors de développement ou d'ingénierie basée sur les modèles (en anglais *Model Driven Development M.D.D. - Model Driven Engeneering M.D.E. - Model Driven architecture M.D.A.*)[107].

L'utilisation de modèles présente plusieurs avantages :

- n'étant pas matériel leurs coûts de mise en œuvre est faible,
- ils permettent de s'abstraire du système physique,
- ils permettent de mieux comprendre le système en décrivant ses différents aspects comme son architecture, ses comportements ou encore ses interactions avec son environnement,
- ils améliorent la communication au sein du projet,
- ils peuvent être analysés par des outils informatiques et/ou servir à préparer des test, ou effectuer des simulations ou des vérifications formelles.
- enfin, ils constituent une documentation précise qui facilite la maintenance du système [5].

Dès lors, une vaste communauté s'est constituée au fil des années, réunissant les experts de différents domaines, autour du concept de développement par modèle. Que ce soit par exemple pour effectuer des vérifications formelles, spécifier des STI ou réaliser des simulations, il existe une grande variété de formalismes permettant de modéliser un système.

2.3.1 Formalismes de modélisation

Le formalisme le plus utilisé de nos jour est le Langage de Modélisation Unifié (en anglais UML pour *"Unified Modeling Language"*). Sa célébrité est due au fait qu'il est simple à comprendre et qu'il permet de spécifier les aspects statiques et dynamiques d'un système, de la description des cas d'utilisation à la conception des algorithmes en passant par le déploiement. Sa large utilisation est aussi due au fait qu'il n'est pas spécifique à un domaine d'application particulier. En ressemblant autour de lui une large communauté de chercheurs et d'utilisateur, UML a joué un rôle majeur pour le progrès des méthodes de développement basées sur les modèles. Mais c'est aussi un des langages les plus critiqués :

- il est difficile de définir l'ensemble minimal de diagrammes suffisant pour un projet donné,
- son méta-modèle est très compliqué, ce qui rend parfois son utilisation difficile et pose des problèmes lorsqu'on cherche à faire évoluer ce formalisme[49],

5. Cela nécessite que les modèles soient régulièrement mis à jour.

– sa sémantique reste floue, et malgré les possibilités d'extension de sa syntaxe et de sa sémantique, il est compliqué d'éviter les ambiguïtés d'interprétation des modèles.

Il existe d'autre langages de modélisation dont il serait difficile d'établir une liste exhaustive tant leur nombre est élevé. Ils sont souvent restreint à une problématique comme les laquages de spécification d'interfaces (par exemple WSDL et SOAP pour les services web) ou adapté à un domaine spécifique comme AADL [110].

Ces langages plus restreints sont souvent plus adaptés à certains secteurs d'activités ou besoins et offrent des outils de vérification plus puissants. Par exemple la Society of Automotive Engineers (S.A.E) publie en 2004 le standard AADL spécifique aux systèmes embarqués en automobile et aérospatiale afin de permettre des vérifications de propriétés comme l'ordonnançabilité, la bonne transmission des messages ou le bon dimensionnement du matériel (comme la capacité mémoire) à partir de diagrammes .

L'expressivité du langage de modélisation est ainsi souvent en opposition avec sa capacité d'analyse et plus généralement avec la prise en compte des besoins spécifiques d'un domaine. Ainsi, il reste à la charge des managers de projet de choisir les normes à respecter, le cycle de développement et les langages de modélisation qui seront utilisés en fonction des besoins spécifiques du système.

2.3.2 Maturité des modèles

L'ingénierie basée sur les modèles propose une approche où les modèles jouent un rôle majeur lors de la conception, de l'implémentation et de la maintenance du système. En fonction de la nature des modèles, et des outils associés, il est possible de faire différentes utilisations :

1. de la conception et de la communication au sein du projet,
2. de la spécification,
3. de la génération de code ou de circuits électroniques.

Pour décrire le niveau de maturité des modèles utilisés dans les projets, Warmer et Klleppe ont établi une échelle graduée de 0 à 4[126]. Au niveau 0, tout est dans l'esprit des développeurs. Au niveau 1, les modèles servent à la conception et à la communication. Au niveau 2, ils servent à la spécification du système. Puis, aux niveaux 4 et 5, les modèles peuvent être synchronisés avec le code de manière partielle (niveau 4) ou totale (niveau 5).

2.4 Vérification formelle de modèles

Un langage formel repose sur un ensemble de définitions strictes et non ambiguës qui permettent de constituer un environnement de raisonnement logique. Il permet de constituer des formules, des algorithmes, ou plus généralement de *formaliser* un problème qui pourra ensuite être *résolu* ou *calculé* à l'aide des règles de déduction logiques définies par ce langage. Cette démarche ne repose ni sur l'intuition ni sur des sentiments humains, mais sur l'application de règles formellement définies. Le calcul d'une proposition ou d'un algorithme peut donc être automatisé.

C'est Alonzo Church et Alan Turing qui dans les années 30 théorisent cette automatisation du calcul grâce aux *Machines de Turing* et au *Lambda Calcul*. Ils donnent ainsi la première définition de ce qui peut être calculé en résolvant le problème de la *calculabilité*.

Deux familles de méthodes formelles sont nées de ces travaux :

- **La preuve de théorèmes** (ou *Theorem proving*) : la première, qui découle du *Lambda Calcul*, est basée sur une approche par preuves. Le modèle est décrit au moyen d'axiomes, les propriétés sont des théorèmes qui sont alors vérifiés par un *assistant de preuve*.
- **La vérification de modèles** (ou *Model Checking*) : la seconde, issue des *Machines de Turing*, est basée sur la vérification de modèles. Le modèle est transformé en automate dont on calcule les états atteignables dans un *graphe d'états*. Il est alors possible d'explorer le graphe et de vérifier si une propriété est satisfaite.

Ces deux approches sont complémentaires. Les approches basées sur les preuves permettent l'analyse de systèmes dont l'espace d'états est infini. Cependant, l'utilisation d'un assistant de preuve est une tâche technique qu'il est difficile d'automatiser. Les preuves obtenues sont aussi complexes à analyser.

Au contraire, la vérification de modèles est a priori dédiée aux systèmes finis, et la plupart des étapes sont complètement automatisées, ce qui facilite leur utilisation par un utilisateur qui n'est pas expert dans le domaine. Mais cette technique est confrontée aux problèmes d'explosion combinatoire.

Depuis l'invention des *Machines de Turing* et du *Lamda calcul*, de nombreux formalismes et outils de vérification formelle ont vu le jour. Les plus utilisés ont été classés en 2008 par Kordon, Hugues et Renault [79] en fonction des propriétés qu'ils permettent d'analyser (figure 2.5).

Voici une liste non exhaustive de ces propriétés couramment recherchées sur un modèle et que nous vérifieront sur les cas d'études présentés dans ce mémoire :

- La cohérence des interfaces (en anglais *Interface consistency*) : garantit que les différentes interfaces des modèles sont cohérentes entre elles.
- La présence de blocages (en anglais *deadlocks*) : garantit que le modèle peut repasser infiniment par le même état, et donc qu'il ne va pas se bloquer sur un état donné.
- La causalité (en anglais *Causality*) : permet de garantir qu'un état (ou une famille d'états) du modèle sera atteignable à partir d'un état (ou d'une famille d'états) donné.

2.4.1 Les réseaux de Petri

Le parallélisme soulève de nombreux problèmes de modélisation et de validation des mécanismes de communication et de synchronisation, pour lesquels les automates classiques ne sont pas adaptés. Dans les années 60, Carl Adam Petri crée un formalisme spécifique pour modéliser, simuler et vérifier les systèmes concurrents : les réseaux de Petri.

Sémantiquement, les réseaux de Petri permettent d'analyser des systèmes concurrents capables d'interagir et de s'échanger des ressources. L'analyse se fait souvent par l'étude de l'ensemble des états accessibles par les systèmes étudiés. Les propriétés structurelles du réseau peuvent aussi être analysées.

Dans cette section nous présentons le formalisme des Réseaux de Petri et certaines de ses évolutions adaptées à l'analyse des systèmes complexes.

Frameworks	Interface consistency	System invariants	Fault-Tree Analysis	Schedulability	Liveness	Causality/deadlocks	Performance analysis	Available tools
Simulation				×				Cheddar , CPNTOOLS , Rhapsody , Renew , SCADE , Simulink
Semantic Analysis	×			×			×	Cheddar , MAST , SPARK , TRAIAN
Type checking	×							EiffelStudio , FuZZ , Z/EVES
Theorem proving	×	×	×		×	×		Atelier B Coq , Z/EVES , PVS
Model Checking...	×	×		×	×	×		CHARON , CPN-AMI , FAST , SMV , SPIN , SCADE , SPOT
...timed							×	CADP , Kronos , TINA , UPPAAL
...stochastic							×	GreatSPN , PRISM , QPME

FIGURE 2.5 – Outils de vérification formelle et portée des vérifications

Les Réseaux Place-Transition [6] :

Un réseau de Petri permet une modélisation formelle des **relations causales** entre les **états** (appelés places) et les **actions** (appelées transitions) d'un système en fonction d'un ensemble de **ressources** (appelées jetons ou marquages). Les changements d'état, déclenchés par des actions, sont conditionnés par la disponibilité de ressources. Les ressources peuvent être produites ou détruites lors d'un changement d'état.

Definition 2.4.1. ***Présentation d'un réseau Place-Transition :*** *Un réseau de Petri est un graphe biparti orienté valué qui a deux types de nœud : la place et la transition ;*

- *un arc est soit il est un lien d'une place à une transition, soit un lien d'une transition à une place.*
- *L'arcs est étiquetés par une valeur (ou un poids), qui est un nombre entier positif. L'arc ayant k poids peut être interprété comme ensemble de k arcs parallèles.*
- *L'étiquette du poids égale à 1 est ignorée.*
- *Un marquage assigne un nombre entier à chaque place. Si un marquage assigne un entier k à une place p, on dit que p est marqué par k jetons.*

Definition 2.4.1. *Un PT-NET et un 5-tuple* $Q = \langle P, T, Pre, Post, m_o \rangle$ *ou*

- P *est un ensemble fini de places ;*
- T *est un ensemble fini de transitions ;*
- $P \cap T = \emptyset$*, i.e. les ensembles* P *et* T *sont disjoints ;*
- $Pre : P \times T \to \mathbb{N}$ *est l'application d'incidence avant ;*
- $Post : P \times T \to \mathbb{N}$ *est l'application d'incidence arrière ;*

6. aussi appelés 'P/T nets'

– $m_o : P \times \mathbb{N}$ *est l'application de marquage initiale qui à chaque place* P_i *associe un nombre de jetons.*

Cette sémantique est adaptée à de nombreux domaines.
– La description de transformations chimiques de molécules, les différents atomes ou molécules produites étant représentées par des ressources. Les différentes réactions possibles entre atomes/molécules consommant et produisant des ressources sont modélisées par les règles de transition entre états.
– La modélisation d'une chaîne de production.
– La modélisation de systèmes informatiques comme un protocole, un programme ou encore un *framework*.
– Tout système à transition discrète qui peut être modélisé par un ensemble d'état et de règles de production et de consommation de ressources.

Extensions des réseaux de Petri Il existe un grand nombre d'extensions au formalisme des réseaux de Petri. Certaines permettent des approches probabilistes ou la modélisation des phénomènes continus. La section suivante présente les extensions aux réseaux de Petri que nous avons utilisés.

Les réseaux de Petri Colorés Les réseaux de Petri Colorés (CP-NET) ont été introduits [74] afin d'améliorer l'expressivité du formalisme en permettant une définition plus riche des ressources manipulées. Les ressources, ou jetons du réseau, sont définies par des classes qui décrivent l'ensemble des valeurs que peut prendre chacun des jetons, qui s'apparentent maintenant à des variables dont la valeur peut évoluer entre les différents états du modèle.

Cette amélioration a de multiples conséquences. Les contraintes exprimées sur les arcs entrant et sortant des transitions peuvent maintenant porter sur certaines classes de jetons et leurs valeurs. De même, la production ou destruction de ressources est maintenant conditionnée par ces classes et permet l'utilisation de fonctions complexes lors de leur manipulation (comme les opérations arithmétiques : l'addition, la multiplication, la division ou la soustraction).

Les réseau Colorés facilitent grandement la modélisation des systèmes complexes et leur compréhension au travers du modèle. Il est par exemple possible de définir différentes classes de ressources pour les flots de contrôle et de données, ce qui est plus complexe à modéliser avec des réseaux simples de type "Place / Transition" où un seul type de ressource est disponible.

Definition 2.4.1. *Un CP-NET et un 9-tuple* $CPN = \langle \Sigma, P, T, A, N, C, G, E, I \rangle$ *ou*

(i) Σ *est un ensemble non vide de types, aussi appelé ensemble des couleurs.*
(ii) P *est un ensemble fini de places.*
(iii) T *est un ensemble fini de transitions.*
(iv) A *est un ensemble fini d' arcs tel que :*

$$\bullet P \cap T = P \cap A = T \cap A = \emptyset.$$

(v) N et une fonction de nœud. N est définie de A vers $P \times T \cup P \times T$.
(vi) C est une fonction de couleur. C est définie de P vers $\sum$.
(vii) G est une fonction de garde. G est définie de A vers des expressions telles que :

$$\bullet \forall t \in T : [Type(G(t)) = \mathbb{B} \wedge Type(Var(G(t))) \subseteq \Sigma]$$

(viii) E est une expression de fonction d'arc. Elle est définie de A vers des expressions telles que :

$$\bullet \forall a \in A : [Type(E(a) = C(P)_M S \wedge Type(Var(E(a))) \subseteq \Sigma]$$

(ix) I est fonction d'initialisation. Elle est définie à partir de P vers des expression telles que :

$$\bullet \forall p \in P : [Type(I(p)) = C(p)_M S].$$

En améliorant grandement l'expressivité du formalisme, l'introduction des "Réseaux Colorés" permet de créer des automates infinis et l'utilisation de fonctions arithmétiques qui rendent impossible l'exploration de l'espace des états accessibles du système. Ainsi, c'est tout un ensemble de propriétés formelles qu'il est dès lors impossible de vérifier. Dans la majorité des cas, seule la simulation du modèle est possible.

Pour pallier ce problème, les réseaux Symétriques ont été définis. En apportant des restrictions quant à l'utilisation des réseaux Colorés principalement sur la définition des *classes de ressources* et des fonctions arithmétiques utilisables, ils forcent la modélisation de modèles finis, permettant l'exploration de l'espace d'états et la vérification des propriétés formelles associées.

Les réseau de Petri Symétriques Les réseaux de Petri Symétriques (SN) [7] ont été introduits [29, 27] dans le but d'exploiter les symétries des systèmes distribués afin d'obtenir une représentation plus compacte de l'espace d'état.

Le concept des réseaux symétriques est similaire à celui des réseaux colorés. Mais les types de places autorisées et les fonctions de couleurs sont plus restreintes. Ces restrictions permettent de capturer les symétries du modèles et d'obtenir une représentation compacte de l'espace d'état, permettant ainsi l'analyse de systèmes complexes [65].

Les réseaux de Petri Symétriques (SN) sont des modèles de réseaux de Petri de haut niveau dont la syntaxe permet d'exprimer la symétrie d'un système par la définition de domaines de couleurs et de fonctions de couleur.

Les jetons du réseau contiennent une information composite exprimée par un tuple de couleurs (appelés objets) issues de *classes de couleur de base* potentiellement ordonnées. Chaque classe de couleur représente des composants du système d'un type donné (ex. une classe de processus, de processeur, etc.).

Si les objets d'une classe se comportent de manière différente, la classe doit être partitionnée en sous-classes statiques. Les objets d'une même sous-classe statique représentent des entités

7. en anglais "*Symetric Nets*", que l'on peut trouver aussi sous l'appellation plus ancienne de *"Well Formed Nets"*.

se comportant de manière identique (symétrique), alors que les objets appartenant à des sous-classes statiques différentes peuvent avoir des comportements différents. Cet aspect s'exprime dans la syntaxe des fonctions de couleurs du modèle, qui permettent de distinguer les objets d'une même sous-classe statique.

Les fonctions de couleur du modèle SN sont construites à partir de trois types de fonctions de base : la fonction de *projection*, la fonction *successeur* et la fonction $diffusion/synchronisation$.

- La syntaxe de la fonction de *projection* est x, où x est une variable de transition. Elle est appelée projection par qu'elle sélectionne un élément du tuple des valeurs du paramètre, qui représentent l'instance de couleur de la transition.
- La syntaxe utilisée pour la fonction *successeur* est $!x$ où x est une variable de la transition. Elle s'applique uniquement au classes ordonnées et retourne le successeur de la couleur assignée à x dans l'instance de couleur de la transition.
- La syntaxe de la fonction de $diffusion/synchronisation$ est $C_i.all$ (ou $C_i^j.all$) où C_i est une classe de couleur et C_i^j une sous-classe statique de C_i. C'est une fonction constante qui retourne l'ensemble des couleurs de la classe C_i (ou de la sous-classe statique $C_i^j \subset C_i$). Elle est appelée *synchronisation* quand elle est utilisée sur un arc entrant, car elle implémente la synchronisation de l'ensemble de jetons colorés contenus dans une place. Elle est appelée diffusion quand elle est utilisée sur un arc sortant d'une transition, car elle met un jeton de chacune des couleurs de la (sous)classe à laquelle elle s'applique dans une place.

Definition 2.4.1. *Soit SN un réseau Symétrique [26] [40].*

Un SN est un 7-tuple $= \langle P, T, Pre, Post, Cl, C, \phi \rangle$ *où*

- $Cl = \{C_1, ..., C_k\}$ *est un ensemble de classes élémentaires ; chaque classe élémentaire* C_i *est un ensemble fini et non vide pouvant être partitionné en* s_i *sous-classes statiques* $(C_i = \uplus_{q=1..s_i} C_{i,q})$ *; On note* $n_i = |C_i|$ *et* $n_{i,q} = |C_{i,q}|$ *;*
- P *est un ensemble non vide et fini de places ;*
- T *est un ensemble non vide et fini disjoint de P de transitions ;*
- C *est la fonction de couleur de domaine* $P \cup T$ *et de codomaine* ω*, où* ω *est un ensemble contenant les produits cartésiens finis d'éléments de Cl (classes élémentaires). Un élément de* $C(s)$ *est un tuple* $< c_1, c_2, ..., c_l >$ *et appelé couleur de s ;*
- $Post(resp.Pre)$ *est la fonction d'incidence avant (resp. arrière) qui associe à tout couple* p,t *de* $P \times T$ *une fonction de couleur gardée de* $C(t)$ *vers* $Bags(C(p))$ *;*
- $\phi(t)$ *est une fonction qui associe à chaque transition une garde.*

Ce formalisme permet de prendre en compte le typage des entités et de définir des ensembles d'entités de cardinalité finie. Ceci permet une définition concise et paramétrée du système, tout en préservant sa sémantique.

Un exemple de réseau de Petri Symétrique est présenté figure 2.6. Il modélise une classe de processus (identifiée par le type P) qui accèdent à une ressource partagée **CR**. Les processus obtiennent la valeur de la ressource de type *Val* à partir de la place **CR**. Les constantes **PR** et **V**

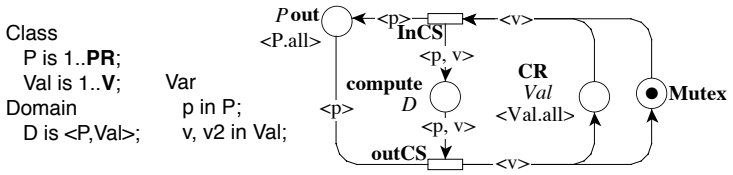


FIGURE 2.6 – Exemple de réseau Symétrique

sont des paramètres entiers du système. La classe de processus est modélisée par les places **out** et **compute**.

La place **compute** représente un traitement effectué sur la valeur fournie par la place **CR**. Chaque processus possède une valeur qui est restituée quand le calcul est terminé. La place **Mutex** modélise une exclusion mutuelle entre les processus et possède un jeton sans valeur. La place **out** est initialement marquée avec un jeton pour chaque valeur dans *P* (indiqué par <P.all>) et la place **CR** possède un jeton pour chacune des valeurs du type *Val*.

L'intérêt principal des réseaux Symétriques est la possibilité de générer un espace d'états symbolique. Un état symbolique représente un ensemble d'états concrets ayant une structure identique. Ceci est illustré figure 2.8 où l'état symbolique en haut sur la figure, représente les quatre états concrets en haut sur la figure 2.7.

Sur la figure 2.8, l'état représenté en bas correspond au marquage initial du modèle quand **out** contient un jeton pour chaque valeur du type **P** et *CR* un jeton par valeur du type *Val*. L'état du haut représente un ensemble d'états où les valeurs des variables *P* et *V* peuvent être permutées. Dans cet état la place **compute** contient un seul jeton, alors que les places **out** et **CR** contiennent un jeton par valeur du type de ces places moins la valeur utilisée pour construire le jeton de la place **comput**.

Dans certains cas, comme ici, l'espace d'état symbolique ne change pas en fonction des types *P* et *V* ou du marquage initial du modèle. Cela permet d'obtenir une représentation compact du comportement du système. La vérification de propriétés peut être effectuée par une analyse structurelle sur l'espace d'état symbolique ou sur le modèle Place/Transition correspondant obtenu par dépliage du modèle symétrique.

Des travaux récents ont automatisé l'analyse des symétries admises par un système [117] et des symétries d'une propriété [4]. Ces techniques, couplées à des outils de vérification automatique nous autorisent une vérification du modèle par des propriétés de logique temporelle LTL tout en combattant efficacement l'explosion combinatoire de l'espace d'états inhérente à

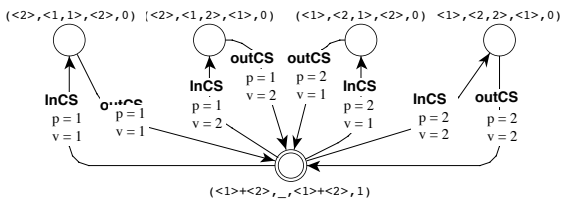


FIGURE 2.7 – Espace d'état du modèle de la figure Fig. 2.6

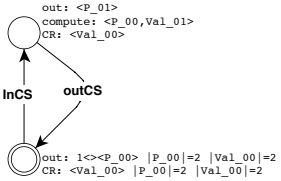


FIGURE 2.8 – Espace d'état symbolique du modèle de la figure Fig. 2.6

ce type d'analyse, et en limitant ses effets.

2.5 Cas d'étude : le projet SAFESPOT

SAFESPOT est un projet intégré, co-financé par la commission européenne, dont les objectifs sont la conception, le développement et l'évaluation de "Systèmes de Sécurités Coopératifs pour le Transport Routier". L'enjeu de SAFESPOT est de comprendre comment des véhicules "intelligents" et des routes "intelligentes" peuvent coopérer pour réduire de manière significative le nombre d'accidents sur la route.

2.5.1 Objectif et concept du projet

En combinant les informations collectées par les capteurs des véhicules et ceux de l'infrastructure, le projet SAFESPOT permettra d'étendre dans l'espace et le temps la perception des dangers potentiels. La transmission d'alertes et de recommandations aux véhicules par le biais de communications véhicule à véhicule ou entre l'infrastructure et les véhicules, améliorera la perception que les conducteurs ont de leur environnement.

L'architecture fonctionnelle est conçue pour être la même pour les deux entités du système, les véhicules et l'infrastructure, formant ainsi un réseau pair-à -pair. Cela permet une approche décentralisée des échanges d'informations concernant l'état des véhicules, l'état du réseau, des informations relatives à la météo ou l'état de la chaussée. Cette approche coopérative permet de définir de nouvelles applications de sécurité pour la route [106, 14] .

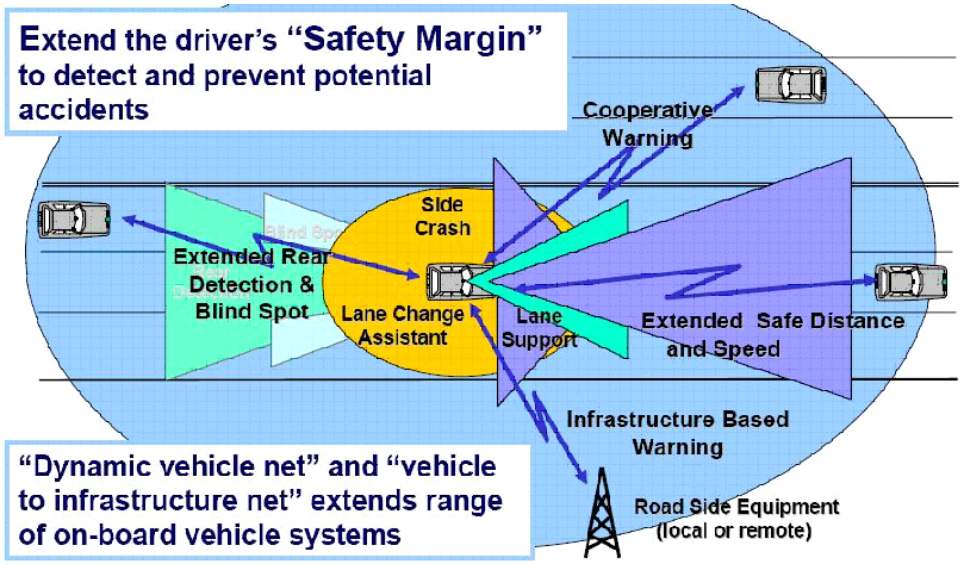


FIGURE 2.9 – Le concept de "Safety Margin" dans SAFESPOT

En combinant les informations provenant des capteurs embarqués dans les véhicules et déployés sur l'infrastructure, le projet SAFESPOT permet une amélioration du délai de détection des risques d'incidents potentiels. La transmission d'alertes et de recommandations aux véhicules, permet d'améliorer dans le temps et l'espace la vision qu'a le conducteur de son environnement. Le concept de "marge de sécurité" *"Safety Margin"* développé dans

SAFESPOT est présenté sur la figure 2.9. Le véhicule central de cette figure est équipé de nombreux systèmes d'assistance à la conduite (ADAS), il communique avec les autres entités (véhicules et infrastructure) afin d'étendre sa marge de sécurité (en bleu ciel sur la figure).

Ce projet repose sur un certain nombre d'innovations comme :
- un réseau Ad-Hoc entre véhicules et avec l'infrastructure (le V.A.NET. pour "Vehicular Ad-Hoc Network") basé sur la technologie IEEE-802.11p qui permet les communications à grande vitesse et réduit les délais de connections,
- une carte dynamique locale (la L.D.M. pour "Local Dynamic Map") qui permet le stockage et l'échange d'informations dynamiques relatives à la sécurité,
- des réseaux de capteurs sans fil pour l'infrastructure ("wireless sensor network"),
- un ensemble d'applications tirant parti de ces innovations et définissant le concept de marge de sécurité ("*Safety Margin*") mis en place dans SAFESPOT .

Le consortium

Le projet SAFESPOT est le fruit de la collaboration d'une cinquantaine d'entreprises, de laboratoires et d'organismes publics. Il est découpé en huit sous-projets. La société COFIROUTE à la responsabilité de la direction du sous-projet *SP5 - Cooperative Systems Infrastructure Based Applications* consacré à l'élaboration des applications mises en place au niveau de l'infrastructure. J'ai pu participer à la coordination des différentes étapes de conception, de réalisation, de test et de validation de ce sous-projet.

Le rôle de COFIROUTE m'a permis d'être au cœur des décisions prises au niveau des groupes de direction principaux du projet : les groupes *SP7- Core Architecture* et *SP8 - IP Management*.

La figure 2.10 présente les différents sous-projets de SAFESPOT et leurs responsabilités aux travers des différentes étapes du projet.

2.5.2 Architecture du système et technologies impliquées

Concept de "Local Dynamic Map'

Le concept de carte dynamique, présenté figure 2.11, est une innovation importante du projet SAFESPOT.

En combinant les informations provenant des capteurs des véhicules et de l'infrastructure, la carte dynamique de SAFESPOT [6] permet une représentation en temps réel des véhicules et de leur environnement. Ainsi, les applications embarquées dans les véhicules ou celles déployées dans l'infrastructure ont un accès aux informations concernant la présence de piétons, la vitesse des véhicules, les conditions météo, la présence d'obstacles ou de véhicules roulant à contresens, la présence d'accident ou de chantier, la présence de véhicules d'urgence en déplacement ou encore les cycles des feux de signalisation.

Le réseau déployé dans SAFESPOT (le "V.A.Net.") est aussi une innovation clé de ce projet. Les détails relatifs à ce réseau ainsi qu'aux applications sont développés dans les parties suivantes de cette section.

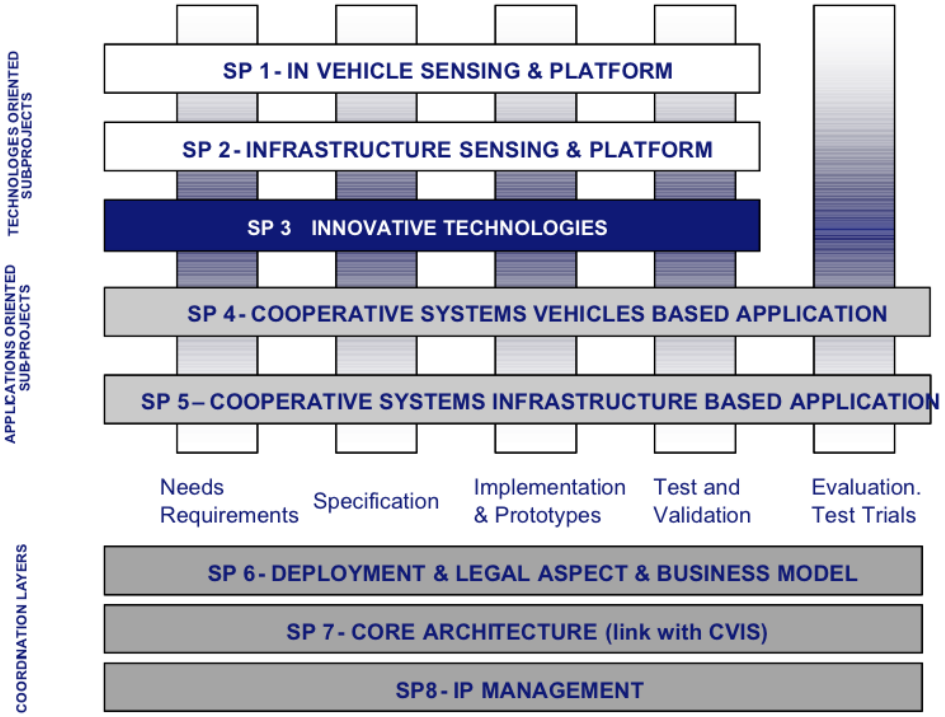


FIGURE 2.10 – Structure des sous-projets de SAFESPOT

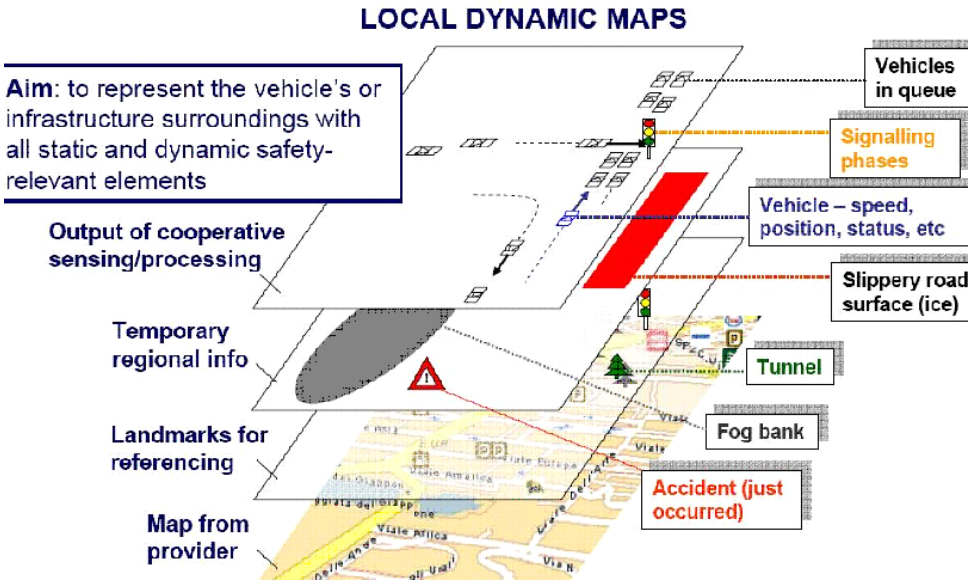


FIGURE 2.11 – Carte dynamique du projet SAFESPOT

Les technologies de communication

Le concept de l'architecture CALM (Communication Architecture for Land Mobile environment) [12] permet le développement d'applications, indépendamment du média de communication utilisé. Dans CALM, le CME (CALM Management Entity) est chargé de choisir la meilleure technologie de communication en fonction de sa disponibilité et des besoins des utilisateurs. Cette architecture permet, d'une part, des communications de véhicule à véhicule ainsi qu'entre les véhicules et l'infrastructure, et d'autre part, un accès à Internet continu à travers différentes technologies potentiellement utilisées de manière simultanée. CALM inclut d'ores et déjà les standards suivants : Cellular (CALM 2G/3G), Infrared light (IR), Microwave (CALM M5), IEEE 802.11 a/b/g (WIFI), IEEE 802.11p (mobile WIFI) [36], Millimeter waves (CALM MM), Microwaves CEN DSRC. D'autres médias comme le Wimax pourraient y être intégrés dans le futur. Dans CALM, l'unification de ces technologies se fait à travers la couche IPv6, les problématiques de réseau mobile étant prises en charge par le protocole NEMO (Network Mobility).

L'architecture CALM développée dans le projet CVIS sert aussi de référence aux projets COOPERS et SAFESPOT. Le tableau 2.5.2 montre comment l'architecture CALM fournit des technologies de communication adaptées aux besoins de ces différents projets.

Interfaces	COOPERS	CVIS	SAFESPOT
CALM-IEEE802.11p		X	X
CALM-IEEE802.11a/b/g			X
CALM-GPRS/UMTS	X	X	
CALM-IR	X	X	
CALM-M5	X		
CALM-CEN DSRC		X	
WIMAX	X		
DAB/BVB-H	X		

TABLE 2.2 – Communications dans COOPERS, CVIS et SAFESPOT

Le réseau ad-hoc de SAFESPOT [18] est chargé d'assurer les échanges de messages entre véhicules et avec l'infrastructure. Il inclut des moyens de maintien du réseau par envoi de balises en continu, et de gestion de la congestion en fonction du nombre d'entités présentes dans une zone géographique. Ce réseau permet différents modes d'envoi des messages, par diffusion géographique (ou "geo-cast") ou en fonction des adresses réseaux des entités grâce à des 'sauts multiples' (ou "multihop routing"). Il propose aussi un mode de contention des messages en cas de faible pénétration du système. Enfin, ce réseau met en place un système de priorités afin de garantir une qualité de service aux messages d'alerte.

La technologie choisie pour le réseau de SAFESPOT est l'IEEE-802.11p. Cette technologie permet la communication à grande vitesse en utilisant une bande de fréquence élevée, et des délais d'établissement des connexions réduit en synchronisant les horloges des cartes de communication avec l'horloge du système GPS. Cette technologie fait partie de l'architecture CALM. Cela permet au système SAFESPOT être compatible avec les autres projets utilisant CLAM, comme CVIS et COOPERS.

Architecture fonctionnelle pair-à-pair :
Grâce aux communications sans fil et à l'utilisation d'une base de données dynamique, l'architecture fonctionnelle de SAFESPOT est basée sur une approche pair-à-pair où chaque entité, véhicule ou l'infrastructure, fournit et requiert les mêmes services :

- échanges automatiques des données dynamiques entre entités,
- transmission des informations au travers des véhicules ou de l'infrastructure qui peuvent servir de relais de manière transparente,
- une chaîne de traitement de l'information et de génération des alertes à l'intérieur des entités,
- une politique commune de gestion des messages basée sur leur priorité.

Cette architecture pair-à-pair n'empêche pas pour autant des variations quant à leurs implémentation logique ou physique. La nature des capteurs mis en place n'est pas la même dans toutes les entités. Aussi, les applications de sécurité dépendent des entités même si certaines sont implémentées aussi bien côté infrastructure que côté véhicules.

2.5.3 L'application "Hazard and Incident Warning" :

La coopération entre l'infrastructure routière et les véhicules du projet SAFESPOT par échange d'information à différents niveaux de traitement (des données brutes des capteurs aux commandes générées par les applications) permet l'émergence de nouvelles applications de sécurité dans le domaine des STI (cet aspect est détaillé chapitre 2 section 2.1).

Nous présentons ici l'application "Hazard and Incident Warning" (H&IW) que j'ai développée au sein du projet SAFESPOT et que nous utilisons comme cas d'étude dans ce mémoire.

L'objectif de cette application est d'alerter les conducteurs en cas d'événement dangereux survenant sur la route. Les danger considérés vont de la plaque de verglas aux embouteillages en passant par le chantier, le piéton ou un véhicule accidenté.

La stratégie d'alerte de H&IW améliore les méthodes de signalisation actuelles pour ce type d'obstacles. En cas de chantier ou d'accident sur la route, deux stratégies de signalisation sont actuellement utilisées.

La première consiste à déployer une série de panneaux en amont du danger présentant un certain nombre d'indications comme une réduction de voie ou une limitation de vitesse. Cette méthode de signalisation a l'inconvénient de reposer sur un ensemble de panneaux en général de petite taille et parfois difficiles à voir de loin ou en cas de trafic dense.

Une deuxième stratégie est alors souvent utilisée, en particulier sur autoroute, basée sur l'utilisation de véhicules de maintenance équipés d'une *flèche lumineuse de rabattement* de grande taille, fixée en hauteur sur le véhicule. Cette méthode, bien que fournissant moins d'information, est plus visible et donc souvent plus efficace. De plus, elle est plus simple et plus rapide à mettre en place.

Ces deux stratégies sont illustrées figure 2.12, avec à gauche la stratégie basée sur l'utilisation de panneaux déposés sur la chaussée, et à droite, la stratégie basée sur l'utilisation de véhicules d'intervention munis de *flèches lumineuses de rabattement*.

Comme le montre ces deux stratégies de signalisation un choix doit être fait entre : fournir une information détaillée au conducteur, ou fournir une information plus visible. L'application

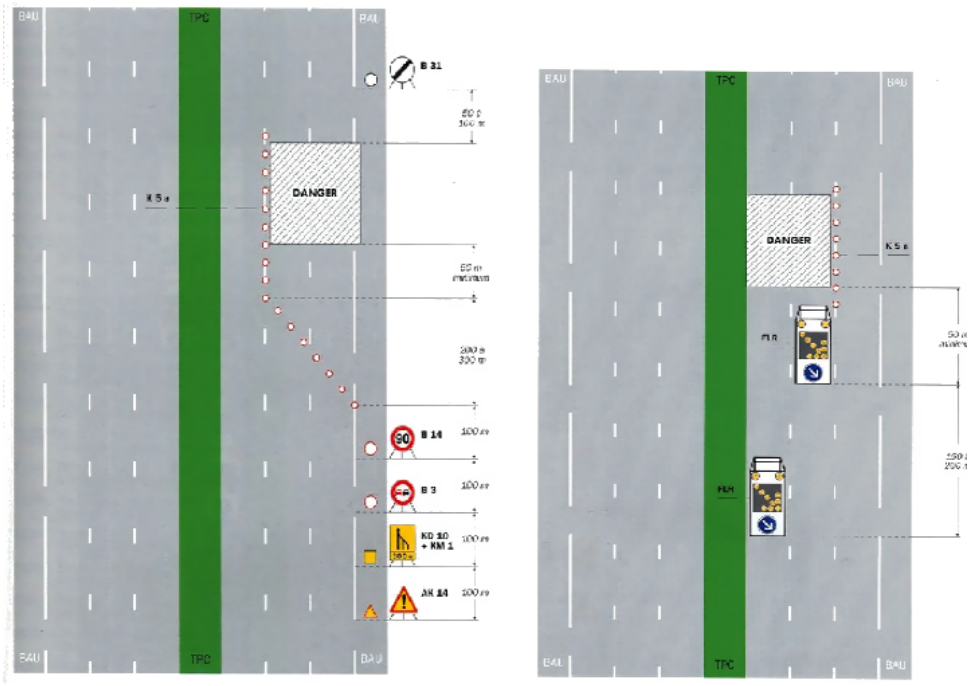


FIGURE 2.12 – Extrait du guide de signalisation routière : signalisation de danger sur autoroute

H&IW améliore la prise en charge de ce type de situation car elle permet :

- une **gestion automatique des dangers** sur la route, de la détection à l'affichage,
- la **réduction des problèmes de visibilité** causés par exemple par le brouillard, la présence de poids lourds ou d'un virage, en permettant un affichage d'alertes à bord des véhicules,
- **une diffusion quasi instantanée de l'information** par l'intermédiaire du réseau de communication formé par les véhicules et l'infrastructure,
- la **prise en compte des caractéristiques des véhicules** comme leur poids ou leur type (marchandise, services, tourisme),
- un **ajustement du niveau d'alerte en fonction du danger** en changeant l'aspect visuel ou audio des alertes affichées dans les véhicules.

2.6 Synthèse

L'application H&IW que nous avons développée est une application critique car elle prend en charge la sécurité des usagers de la route. Elle fait partie du système SAFESPOT qui permet, grâce à son approche coopérative entre les véhicules et l'infrastructure, d'étendre dans l'espace et le temps la perception que les conducteurs ont de leur environnement. C'est donc une application complexe et critique pour laquelle il est important de pouvoir garantir la fiabilité. Ainsi que nous l'avons présenté dans ce chapitre, la vérification formelle de ses spécifications permet :

– de fournir une preuve logique de leur validité,
– la correction d'erreurs qui, à ce stade du cycle de développement, coutera potentiellement moins cher que lors des phases ultérieures du projet.

L'application H&IW nous sert de cas d'étude tout au long de ce mémoire. Au chapitre 4 nous présenterons ses différents cas d'utilisation et comment ils ont été élaborés. Aux chapitres 5 et 6 nous présentons ses spécifications et les méthodes que nous avons élaborées afin de fournir une preuve formelle de leur validité.

Chapitre 3

Contributions : Vérification des spécifications de systèmes complexes

Sommaire

Notre objectif est de proposer une solution pour intégrer les méthodes formelles au développement des systèmes complexes et critiques. En partant de langages semi-formels de spécification **utilisés dans l'industrie**, nous définissons des règles de transformation afin de produire des modèles formels. Ces modèles sont analysés et les résultats peuvent ensuite être utilisés pour modifier les spécifications de départ. L'utilisation de **patrons génériques** et de **bibliothèques de modèles** nous permet de modéliser formellement des systèmes constitués d'un grand nombre de composants. L'utilisation de **différentes extensions des réseaux de Petri** permet d'adapter nos modélisations à **différents niveaux d'abstraction** et de réduire le problème de l'explosion combinatoire. L'utilisation des **techniques de discrétisation** nous permet de modéliser les phénomènes continus qui entrent en jeu dans l'évolution du système.

Notre approche permet ainsi la **vérification des spécifications** d'un système et s'intègre aux **cycles de développement en "V"** bien qu'elle puisse être utilisée avec d'autres cycles de développement.

Dans ce chapitre nous présentons, dans la première section, les problèmes liés à la vérification formelle des spécifications de systèmes complexes. Puis comment notre méthode de vérification formelle des spécifications s'intègre au cycle de développement d'un système. Nous expliquons, dans une deuxième section, les contraintes de spécification de systèmes complexes et critiques et comment nous les avons prises en compte dans le projet SAFESPOT. Enfin, dans une troisième partie, nous présentons comment ces spécifications sont transformées à différents niveaux d'abstraction pour permettre la vérification formelle des propriétés qualitatives d'un système.

3.1 Objectif : Intégration des méthodes formelles au cycle de développement

L'objectif de notre méthodologie est de guider les concepteurs vers une situation où ils pourraient vérifier des propriétés qualitatives de leurs spécifications, et ainsi les rendre plus fiables. Cela implique l'utilisation de méthodes formelles, mais cette approche pose différents problèmes que nous présentons dans la première partie de cette section. Nous expliquons ensuite comment notre méthode s'intègre à un cycle de développement en "V".

3.1.1 Problématique

L'utilisation des méthodes formelles au cours de la conception des systèmes complexes pose les problèmes suivants :

Problème 1 Une compétence nouvelle est nécessaire : la maîtrise d'une méthode formelle. Cependant de nombreuses autres sont déjà indispensables à la réalisation d'un système complexe. Par exemple la connaissance de différents langages de modélisation, du processus de spécification ou encore des langages de programmation.

Maîtriser la complexité d'un système passe par l'utilisation des méthodes formelles, seules à même de fournir la rigueur mathématique nécessaire à la production de preuves logiques. Or cela implique une complexification préalable du processus de conception [1].

Problème 2 Une mauvaise communication au sein d'un projet peut générer des erreurs de conception. Cependant les méthodes formelles sont souvent exprimées au travers de modèles complexes qui ne facilitent pas la communication entre plusieurs équipes.

Problème 3 Si les méthodes de vérification de modèles permettent d'utiliser des outils de vérification automatique, elles impliquent aussi le calcul exhaustif des multiples états d'un système et/ou le calcul de propriétés impliquant une importante complexité algorithmique. Elles posent un problème d'explosion combinatoire [2] qui limite leur capacité à être utilisées.

Problème 4 Choisir l'étape appropriée, l'abstraction nécessaire et suffisante, et les modèles pertinents pour l'utilisation d'une approche formelle est d'autant plus important que l'utilisation de ces méthodes a un coût non négligeable. Un problème d'intégration au processus de développement se pose donc.
Une technique doit donc être mise en place pour faciliter l'utilisation des méthodes formelles et gérer ces problèmes d'explosion combinatoire, de flexibilité et d'intégration aux méthodes de conception classiques. C'est ce dernier aspect que nous présentons dans la section suivante.

3.1.2 Relation avec le cycle de développement

Un cycle de développement peut être décomposé en trois parties : d'abord la spécification, puis l'implémentation et enfin la phase d'intégration, de tests et de validation. Nous avons expliqué chapitre 2 section 2.2.3 que toute modification ou erreur détectée au cours le la réalisation d'un système coûte d'autant plus cher qu'elle survient tardivement. Nous avons donc décidé d'intégrer les méthodes formelles dans la première partie d'un cycle en V : sa phase de spécification. Il existe alors plusieurs alternatives pour vérifier formellement des spécifications.

Alternative no1 Il est possible de spécifier directement le système à l'aide de modèles formels. Mais cette approche a un défaut majeur. En effet, l'intérêt principal de l'ingénierie basée sur les modèles est que leur représentation graphique est simple à comprendre et synthétique. Cela facilite la compréhension du système et les interactions entre les différents acteurs du processus de développement. Utiliser des modèles formels, austères et compliqués à comprendre, aurait un effet négatif sur la coopération des équipes de développement.

Alternative no2 La deuxième approche consiste à utiliser les spécifications existantes du système, puis à en extraire une spécification formelle : en ajoutant des informations supplémentaires, et/ou en ne conservant que les informations pertinentes en fonction des propriétés

1. Cette approche est dans la continuité de la logique de développement des systèmes complexes : on cherche à gérer la complexité et à prendre les décisions au cours de la conception du système plutôt que pendant sa réalisation.

2. Nous avons ici fait le choix d'utiliser les méthodes de vérification de modèles plutôt que les méthodes de preuve de théorèmes pour des raisons que nous expliquons dans la dernière section de ce chapitre.

que l'on souhaite vérifier. Cette approche est plus souple. Elle permet de combiner les avantages des modèles formels et des modèles semi-formels. C'est donc l'approche que nous avons choisie.

Dans cette optique, le développement basé sur les modèles (Model Driven Development (MDD)) [57] (présenté en section 2.3) est une approche appropriée. Son principal intérêt est l'accent mis sur les modèles comme premiers éléments de description du système.

Si des modèles formels sont produits, il est possible d'en vérifier et d'en prouver des propriétés comme l'équité d'accès à des ressources partagées, la tolérance aux fautes ou l'occurrence d'un événement donné (pour plus de détails, voir la figure 2.5, présentée section 2.4). Dans ce cas, les techniques formelles sont particulièrement adaptées pour évaluer et vérifier des choix d'implémentation sur des systèmes complexes [53].

On souhaite profiter des premiers résultats de vérification formelle le plus tôt possible sur des modèles simples, minimiser le coût de leur élaboration, et maximiser l'efficacité de notre approche. Pour cela, nous produisons des modèles formels à différents niveaux d'abstraction.

Nous avons défini quatre niveaux d'abstraction permettant de couvrir l'ensemble des spécifications. Ils présentent chacun des spécificités qui nécessitent une approche particulière.

- **Niveau 1 - Cas d'utilisation** Cette première ébauche du système constitue la partie fonctionnelle du cahier des charges et nécessite la coopération des différentes parties prenantes du projet : le client , les utilisateurs et les équipes de réalisation. Elle est effectuée en langage naturel ce qui constitue un obstacle à sa formalisation. Pour autant, elle est à la base des spécifications et de la validation finale du système. Nous utilisons dans un premier temps, une approche qui permet d'obtenir des **cas d'utilisation détaillés**. Ceux-ci présentent entre autres les principales opérations effectuées par le système. Dans un second temps, l'utilisation de la méthode FRAME (présentée chapitre 2) nous permet de produire la **liste des acteurs des entités du système** ainsi que la **liste de ses contraintes de fonctionnement**.
- **Niveau 2 - Architecture** La spécification de l'architecture permet la définition de la structure et des interfaces du système. Beaucoup d'éléments la constituent, ce qui rend sa modélisation formelle longue et complexe. D'abord, nous définissions des **règles de transformation** permettant de produire des modèles formels à partir des modèle semi-formels utilisés pour les spécifications. Puis l'utilisation d'un **patron générique pour modèles formels** et l'élaboration d'une **bibliothèque de modèles**, permettent une approche modulaire de composition du modèle formel.
- **Niveau 3 : Composants** Cette étape de spécification permet la définition de la structure interne des différents éléments du système et de leurs comportements. Nous utilisons là aussi des règles de transformation. Il est plus difficile ici d'exploiter la modélisation complète du système à des fins de vérification formelle. Ceci, pour des raisons d'explosion combinatoire. Dans ce cas nous modélisons un **environnement dédié** à chacune des propriétés que l'on souhaite analyser .
- **Niveau 4 : Algorithmes** C'est le niveau de détail le plus élevé. Il permet de décrire les algorithmes critiques du système. Ces algorithmes sont spécifiés en langage mathématique, et les données doivent être modélisées à leur niveau de détail le plus élevé. Cela pose un problème de modélisation formelle lorsque les paramètres et les fonctions impliqués dans l'évolution du système ne sont pas discrets mais continus. L'utilisation d'une

méthode de discrétisation permet d'intégrer ces aspects continus du système à nos modèles formels.

Ces quatre niveaux d'abstraction, et leur place dans un cycle en V, sont présentés sur la figure 3.1.

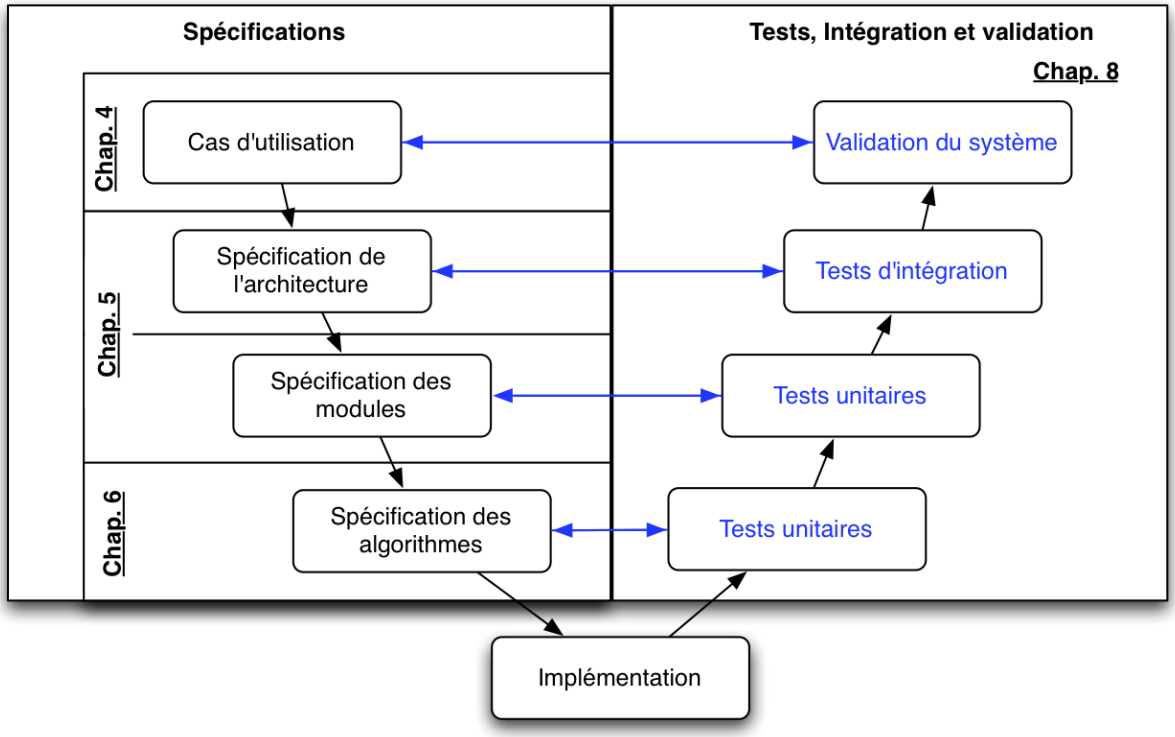


FIGURE 3.1 – Les quatre niveaux d'abstraction des spécifications dans un cycle en "V"

Dans la section suivante, nous présentons les langages de spécification semi-formelle à partir desquels nous avons élaboré notre méthode. Cette présentation est faite autour d'un cas concret de spécification d'un système complexe et critique, le STI SAFESPOT. Ce que nous présentons s'applique à la majorité des systèmes complexes et critiques.

3.2 Contraintes de spécification industrielles

Notre approche consiste à utiliser les spécifications existantes du système, puis à en extraire une spécification formelle. Dans cette section nous présentons les contraintes qui dirigent l'élaboration des spécifications d'un système complexe.

L'élaboration d'un système complexe implique la coordination de spécialistes de différents domaines qui utilisent des formalismes et procédures de spécification différents. Proposer une méthode de spécification commune au sein de ce type de projet pose donc trois problèmes majeurs [49] :

- le choix des formalismes à utiliser qui permettent à tous ces spécialistes de communiquer,
- le choix des modèles à produire,
- et le choix d'une procédure de spécification qui permettent la production des différents éléments dans un ordre cohérent.

Pour résoudre ces problèmes nous avons choisi un **ensemble minimal de modèles**, des **éléments d'harmonisation**, et mis en place une **procédure de spécification** qui permet la coopérations des différentes équipes au sein du projet.

Notre approche est appliquée à un projet de STI : le projet SAFESPOT [20].

Cette section présente la méthode et l'environnement de spécification que nous avons élaborés au sein d'un groupe transversal à ce projet. Il est composé des partenaires venant des différents sous-projets pour prendre en compte les besoins et le spécificité [3].

3.2.1 Objectifs et principales contraintes de spécification

Le projet SAFESPOT fait partie d'une approche Européenne pour les STI incluant d'autres projets que nous avons présentés chapitre 2 section 2.1. Afin de faciliter la coopération et l'harmonisation de ces projets, ainsi que pour favoriser l'émergence de standards européens pour les STI, la commission européenne a fixé deux recommandations concernant la méthode de spécification :

1. l'utilisation de la méthode FRAME [97] (présentée section 2.1) pour l'élaboration des contraintes de fonctionnement détaillées,
2. et l'utilisation d'un formalisme de diagrammes communs : l'UML [88].

Pour le reste, c'est aux différents projets de proposer une méthode de spécification adaptée à leurs spécificités. Les contraintes prises en compte pour le projet SAFESPOT sont communes aux projets impliquant un nombre élevé de partenaires pour l'élaboration de systèmes complexes et critiques. Cette partie présente les contraintes les plus significatives et les choix qui en découlent (leur ordre de présentation n'est pas lié à leur importance).

Contrainte no 1 Gérer la complexité du système et les interactions au sein du projet
Les systèmes complexes comme SAFESPOT nécessitent l'implication de spécialistes de différents domaines comme ceux des communications radio, de la fusion de données, de la cartographie, etc. De nombreuses interactions sont nécessaires entre les différentes équipes du projet. Ces équipes peuvent aussi évoluer au cours du projet pour différentes raisons (comme le passage d'une étape du cycle de développement à une autre). Ces interactions peuvent être fortement pénalisées par un manque de documentation concernant les différents objectifs et productions de chaque équipe.

Contrainte no 2 : Limiter les modifications
Un retard ou un changement des spécifications peut avoir des conséquences sur des pans entiers du système, entrainant des retards en cascades. Ceci est dû à la taille des équipes, au nombre de sous-projets travaillant en parallèle et aux nombreuses interdépendances entre les composants du système.

Un ensemble de principes ont été appliqués pour prendre en compte ces contraintes.

1. Obtenir des contraintes de fonctionnement et des spécifications revues, corrigées, validées puis acceptées afin qu'elles puissent être figées. C'est ce qui est appelé par la communauté de développement Agile le "Big Model Up Front" BMUF [105]. Cette méthode

3. Il s'agit du sous-projet SCORE : *SAFESPOT Core Architecture*.

est souvent critiquée car elle rend la tâche de spécification longue et coûteuse sans pour autant empêcher totalement l'apparition tardive de nouvelles contraintes ou le changement de certaines spécifications [71, 90]. Mais c'est surtout un moyen efficace de réduire les chances que ces modifications ne surviennent.

2. Chaque document de spécification doit être vérifié et validé par des personnes extérieures au processus de rédaction. Ils est alors accepté ce qui permet de passer à l'étape suivante du cycle de développement. C'est l'approche utilisée dans les développements suivant un cycle en *cascade* [104].
3. La taille du projet impliquant une cinquantaine de partenaires, elle rend toutes modifications des spécifications compliquées à valider et à mettre en œuvre. L'approche par cycles courts de développement n'est pas applicable car elle nécessiterait trop d'interactions entre les partenaires du projet. Il s'agit donc de mettre en place un seul et unique cycle de développement [4]. L'objectif est de limiter au maximum les retours en arrière dans le cycle.
4. Le fait de produire le système au terme d'une seule itération n'empêche pas d'utiliser certains aspects des cycle en V. Ainsi, à chaque phase de décomposition seront préparées les procédures de tests et de validation des phases de recomposition (comme nous l'avons présenté chapitre 2).

La méthode de spécification choisie consiste donc à prendre en charge la complexité du système et à résoudre le maximum de conflits et sources d'erreurs avant de passer au développement. Elle permet aussi de limiter les retours en arrière dans le cycle de développement.

3.2.2 Les différentes étapes de la méthode de spécification

Le guide : Le projet SAFESPOT implique un nombre élevé de partenaires venant de pays et d'organisations différentes, ayant des compétences, des connaissances et des pratiques liées à leur domaine d'expertise. Pour aider ces différents partenaires à comprendre et à utiliser les recommandations de spécification, un guide [111] est élaboré. Les diagrammes et descriptions requis y sont détaillés avec des exemples concrets tirés du projet. La suite de cette section en présente le résumé.

Cahier des charges fonctionnel Cette première étape permet la définition des besoins utilisateurs, la production des **cas d'utilisation** et la définition des **contraintes de fonctionnement**. Cela est fait suivant la méthode FRAME. L'analyse des différents environnements d'exécution correspondant aux différents types de routes amène différentes contraintes et choix d'implémentation [124].

Spécification de l'architecture : L'objectif est ici l'identification des principaux **composants**, des **interfaces** entre composants et des **flots de contrôle et/ou de données** du système. Cette tâche est mise sous la responsabilité de représentants de chacun des sous-projets et nécessite une coopération accrue. L'architecture est ici spécifiée à l'aide d'un diagramme UML

4. C'est l'inverse des approches *"Agile"* ou *"Extreme Programming"* qui consistent entre autres à produire le plus tôt possible des parties du système, puis à les valider par de nombreuses itérations de cycles courts de conception, implémentation, tests et validation.

de composants. Ce diagramme de haut niveau produit pour le projet SAFESPOT est présenté figure 3.2 [5]. Les différents composants du systèmes sont colorés en fonction des sous-projets (SP1...SP5) qui ont la responsabilité de leur conception et de leur implémentation.

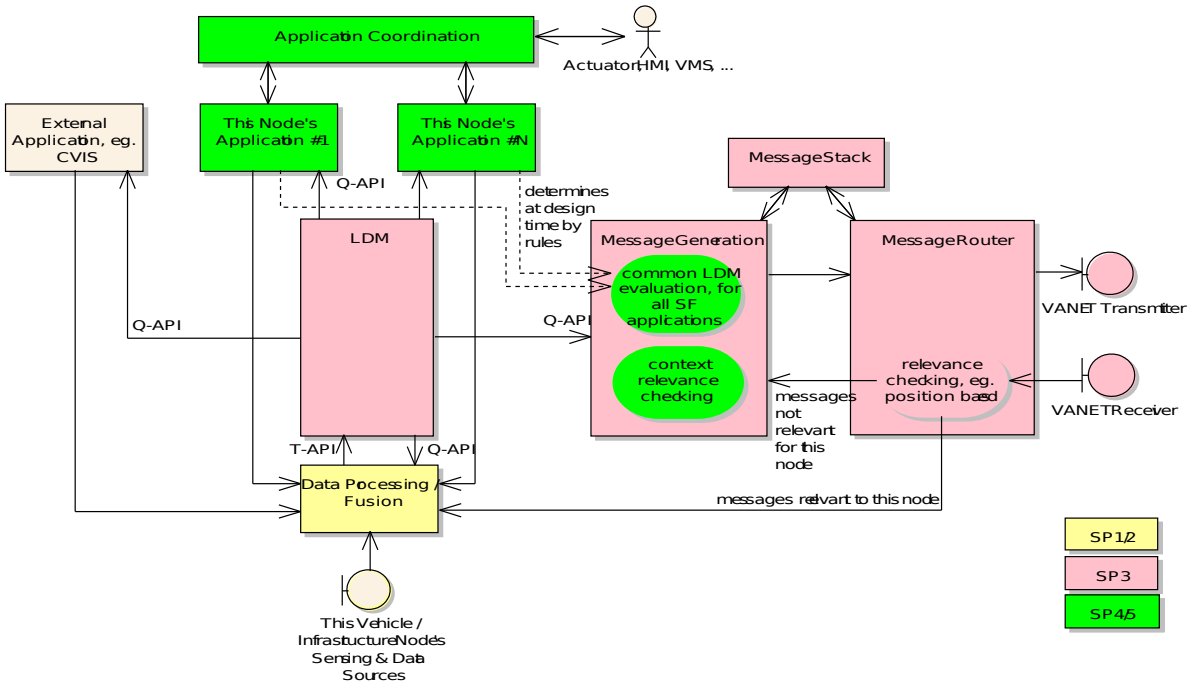


FIGURE 3.2 – L'architecture de haut niveau de SAFESPOT(UML1.0)

Il est possible aussi de spécifier l'architecture fonctionnelle de haut niveau de SAFESPOT grâce aux diagrammes fonctionnels FRAME et ainsi de préciser son rôle et sa place dans un contexte d'harmonisation des STI européens. Cet aspect est présenté en annexe (voir section A.3 page 163).

Les spécifications détaillées : Chaque composant doit ensuite être décrit par différents diagrammes UML accompagnés de descriptions informelles en langage courant. Différents aspects statiques et dynamiques des composants doivent être modélisés.

- **La cohérence avec l'architecture du système**. Elle peut être validée et détaillée en spécifiant les données échangées entre composants et les méthodes d'accès à ces composants. Les diagrammes d'interface UML permettent la modélisation de ces aspects.
- **Les interactions entre composants**. Se mettre d'accord sur les interactions possibles entre composants nécessite la coopération des partenaires du projet. Les diagrammes de séquence UML sont dans ce cas particulièrement adaptés. Ils sont à la fois simples à comprendre et suffisamment précis pour que l'on puisse vérifier leur cohérence avec le reste des spécifications.

5. nommé "The Guyancourt Diagram" dans le projet en référence à la réunion de travail qui l'a vu naître, et qui eut lieu dans les locaux de RENAULT à Guyancourt

– **La structure interne**. Pour permettre l'implémentation des composants, le détail de leur structure interne est nécessaire. Les diagrammes de classe UML ont l'avantage de pouvoir être mis en relation directe avec : d'une part le reste des spécifications et d'autre part le code source du composant.
– **Les comportements internes**. Pour permettre aux développeurs d'implémenter les composants, il faut pouvoir spécifier leur comportement interne. Les diagrammes d'activités UML permettent cette spécification.

Les données : La définition des données échangées entre les composants [127] de haut niveau est une tâche qui perdure pendant toute la phase de spécification du système, et même alors que les développements ont commencé. La contrainte principale sur ces données repose sur leur capacité à être échangées en temps réel. Il est donc naturel qu'elles soient définies à travers leur format d'échange entre composants, et que celui-ci soit économe en bande passante. Le choix est donc inspiré de ce qui est à l'heure actuelle utilisé sur les bus de communication à l'intérieur des véhicules (les bus CAN par exemple). Les données y subissent les mêmes contraintes d'efficacité en bande passante et suivent les grandes lignes de la norme ASN1 [113, 69]. Chaque donnée est donc étudiée séparément et son intervalle de valeur défini. Ainsi, chaque donnée est représentée numériquement par le plus petit type logique possible : par exemple, une donnée n'ayant pas plus de 256 valeurs possibles sera représentée sur un octet. Elles sont ensuite transférées par l'intermédiaire de "Datagram UDP".

Par exemple les latitudes (**LatPosition** sur la figure 3.2.2) doivent pouvoir prendre des valeurs allant de +90 à -90 degrés, avec une précision de 1/8 de micro degré, soit environ 1/10cm, ce qui nécessite 31 bits de données. Les informations temporelles doivent offrir une précision de l'ordre de la milliseconde, et sont donc représentées sur 6 octets avec 2 octets pour la précision en millisecondes.

La figure 3.2.2 montre un extrait des spécifications des données du système. On voit comment la syntaxe choisie permet d'expliquer la sémantique, la précision et la taille des données.

```
    Speed ::= SEQUENCE

/*speed is represented as a vector by (composant and direction(heading))*/
       speedcomposant       Speedcomposant,
       heading           Heading

    Speedcomposant ::= INTEGER (-32768..32767) - Units of 0.01 m/s (16 bits)

                                          -- Actual range (-327,68..327,67) m/s
                                          -- Speed = 327,65 means no speed available
/* Negative values imply the vehicle is moving in reverse. Data not available
 or not valid expressed through the use of Variance */

    Heading ::= INTEGER (0..65535) - LSB of 0.0054931640625 0.005493247 degrees

/*The current heading of the vehicle, expressed in signed units of 0.005493247 degrees
 from North (such that 65,535 such degrees represent 360 degrees ? 1LSB).  North shall
 be defined as the axis defined by the WSG-84 coordinate system and its reference
 ellipsoid. Increasing when turning clockwise.Data not available or not valid expressed
 through the use of Variance */

    LatPosition ::= INTEGER (-720000000..720000000) - in LSB = 1/8 micro degree (31bits)

                                        -- Actual range: ( -90..+90) degrees
```

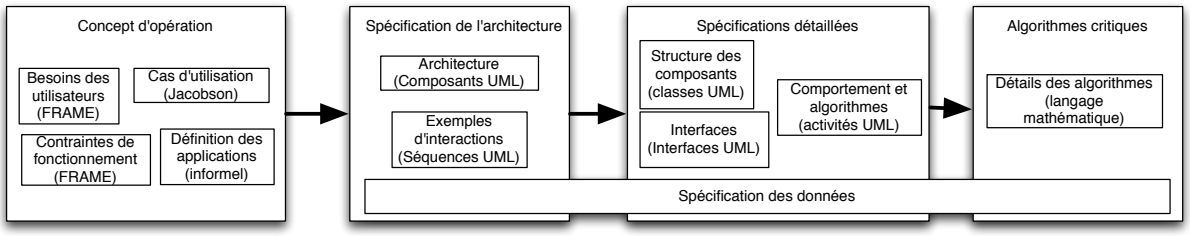


FIGURE 3.3 – Processus de production des spécifications

Les algorithmes critiques : Certains algorithmes font appel à des connaissances avancées qui ne sont pas à la portée des développeurs. C'est le cas avec les applications qui calculent les niveaux d'alerte en fonction de données spatiales et temporelles, ou encore pour les algorithmes de fusion de données. Les extensions d'UML sont trop contraignantes pour être préférées à une expression plus directe en langage mathématique. Ces algorithmes complexes sont donc spécifiés à l'aide d'explications textuelles et d'équations mathématiques. Une vue d'ensemble des étapes de spécification et des formalismes utilisés est présentée figure 3.3.

3.2.3 Les éléments d'harmonisation

Les interfaces : Les différents éléments du systèmes sont connectés à travers des interfaces. Certaines d'entre elles sont considérées comme stratégiques pour l'ensemble du système. Par exemple, l'accès à la base de données centrale (la "Local Dynamic Map" LDM présentée section 2.5), ou l'accès au réseau ad-hoc (le "Vehicular Ah-hoc NETwork" VANET aussi présenté section 2.5) sont partagés par la majorité des composants du système. Leurs spécifications sont donc utilisées par l'ensemble des sous-projets de SAFESPOT. Par conséquent, un planning spécial est mis en place pour valider leur définition dans le projet.

Le flot de données : Un autre point clé d'harmonisation entre sous-projets est construit autour des différents types de données et messages échangés (dans les différents flots de données et interfaces). Cette définition évolue durant la phase de spécification et implique les différentes équipes du projet.

La vérification des contraintes : Chaque composant du système doit être associé à une liste de contraintes de fonctionnement définie au format FRAME. Ces contraintes doivent ensuite être validées au sein du projet. Cette étape implique une forte coopération entre les sous-projets. En effet, une grande partie des contraintes exprimées impactent plusieurs sous-projets et composants. Par exemple, une contrainte de délais conditionnant le bon fonctionnement d'une application peut induire une contrainte de délais sur le composant réseau du système. Après vérification, si des contraintes ne peuvent être satisfaites, les spécifications de certains composants doivent être changés.

3.3 Vérification formelle des spécifications

Nous avons présenté au chapitre 2 section 2.4 les différentes techniques de vérification formelle. Nous expliquons ici pourquoi et comment notre approche est fondée sur l'utilisation des méthodes dites de *vérification de modèles*.

3.3.1 Vérifier les modèles

L'analyse de modèles peut être effectuée principalement à l'aide de deux méthodes : la simulation ou l'analyse formelle.

D'un côté, la simulation peut intégrer un grand nombre de paramètres et ne requiert pas forcément une puissance de calcul très élevée [6]. Mais la simulation a le défaut d'être partielle et certains comportements, bon ou mauvais, peuvent ne pas être exhibés. Elle peut aussi être effectuée de manière intensive sans jamais révéler une trame comportementale formellement exploitable.

D'un autre côté, les méthodes formelles, au sens mathématique du terme, permettent une analyse exhaustive. Il existe deux approches [122] :

- *La preuve de théorème* est utilisée par les méthodes algébriques pour la preuve de propriétés sur le système, potentiellement infini, défini par des axiomes. Cette méthode est difficile à utiliser car elle est rarement automatisée. Elle requiert une grande expertise et des ingénieurs expérimentés. Il est par conséquent très long d'obtenir une preuve.
- *La vérification de modèle* (plus connue sous son appellation anglaise de *model checking*) consiste à explorer de manière exhaustive l'espace d'état d'un système. La vérification est totalement automatisée mais se confronte souvent à un problème d'explosion combinatoire. Par conséquence, cela requiert une puissance de calcul importante et un espace mémoire conséquent. Mais cette technique est bien outillée et différents mécanismes permettent de maîtriser une partie de l'explosion combinatoire grâce, entre autres, à la détection des symétries du modèle.

3.3.2 Maîtriser la complexité du système

Les STI et autres systèmes complexes ont en commun d'impliquer un grand nombre de composants électroniques et informatiques, mais aussi mécaniques ou humains, évoluant dans un environnement complexe. Le nombre d'éléments à modéliser formellement rend la tâche complexe et fastidieuse. De plus, il est important de pouvoir traiter une classe de problèmes plutôt qu'un seul et unique problème. Notre approche est de proposer un **patron générique** pour une classe de systèmes, puis de constituer une **bibliothèque de modèles formels** de composants pouvant être configurés puis assemblés suivant le patron afin de produire un modèle complet et analysable. Ce patron doit donc permettre des variations d'architecture, de composants, ou de paramètres.

Le patron décrit l'architecture du modèle formel à partir de l'architecture du système décrite dans les spécifications UML. Il est générique pour un ensemble de STI et est appliqué au projet SAFESPOT.

6. en comparaison avec une vérification formelle sur le même modèle

3.3.3 Choisir les formalismes adaptés

L'objectif principal de notre approche est de produire des modèles formels analysables à partir de spécifications non formelles. Dans ce sens, l'utilisation combinée d'UML et des méthodes formelles n'est pas nouvelle [109]. Plusieurs approches ont étés proposées qui prouvent la faisabilité et l'intérêt de cette démarche, particulièrement pour l'évaluation de performance que ce soit par vérification de modèles [24] ou par preuve de théorème [16] :

- **Analyse des cas d'utilisation** : en partant des cas d'utilisations définis par Jacobson [72], C. Choppy et G.Reggio montrent comment obtenir des modèles formels en CASL-LTL analysables [30].
- **Analyse de performance :** par transformation de modèles UML vers des modèles comme les réseaux de file d'attente (LQN) ou les réseaux de Petri stochastiques A. d'Ambrogio et J.P. Lopez-Grao ont montré comment procéder à des analyse de performances [35, 84].
- **Analyse structurelle :** par utilisation des diagrammes structurels d'UML comme support à l'élaboration de modèles en langage B, C. Snook propose un moyen de faire une analyse structurelle du système [109].
- **Analyse du comportement :** par la transformation de diagrammes d'activité UML [84], ou par transformation des diagrammes de séquences et de machines à états [8] vers des réseaux de Petri, il est possible d'analyser le comportement d'un système.

Ces transformations reposent sur une spécification des relations entre les **méta-modèles** source et destination, puis sur des **règles de transformation**. Elles permettent principalement l'analyse de propriétés quantitatives ou des simulations. Mais elles ne prennent pas en compte le besoin d'une approche **modulaire** et **hiérarchisée** nécessaire quand le système est trop complexe pour être analysé d'un seul bloc.

Pour proposer une solution adaptée aux systèmes complexes, notre approche de production des modèles formels consiste à utiliser : d'une part les spécifications UML ainsi que les formules mathématiques décrivant les algorithmes critiques ; d'autre part, différentes extensions des réseaux de Petri.

UML offre une notation semi-formelle flexible et extensible pour la description structurelle et comportementale d'à peu près tout type de système industriel. Sa maturité en fait le formalisme le plus approprié à notre approche [118].

Les réseaux de Petri, offrent un cadre formel pour différentes analyses [51].

Il existe en effet un grand nombre d'outils permettant la vérification automatique de divers aspects du système. En particulier, les réseaux symétriques [26], qui capturent les symétries comportementales dans les systèmes distribués, ce qui réduit les problèmes d'explosion combinatoire [117]. Ils sont donc particulièrement adaptés à la vérification des systèmes complexes.

3.4 Synthèse

Les méthodes formelles permettent de vérifier différents aspects d'un système. Mais leur utilisation tout au long d'un cycle de développement pose des problèmes de flexibilité, de passage à l'échelle, d'explosion combinatoire ou encore de modélisation des phénomènes continus.

Pour gérer ces différents problèmes dans le cadre des systèmes complexes et critiques, nous définissons quatre niveaux d'abstraction différents. Pour chacun d'entre eux, nous transformons les spécifications correspondantes en modèle formel. Chaque niveau d'abstraction nécessitant une approche particulière.

Nous présentons les contraintes liées à la spécification des systèmes complexes. Puis, nous expliquons les raisons qui nous amènent à proposer

- l'utilisation de cas d'utilisation détaillés et la méthode FRAME pour rédiger le cahier des charges fonctionnel ;
- l'utilisation de diagrammes UML pour la spécification de l'architecture et des composants du système ;
- l'utilisation de formules mathématiques pour spécifier les algorithmes critiques du système.

Enfin, nous expliquons pourquoi et comment nous décidons d'utiliser les réseaux de Petri et leurs extensions pour la vérification des modèles formels.

Chapitre 4

Élicitation des besoins des utilisateurs pour STI

Sommaire

Les cas d'utilisation (CU) constituent la première ébauche d'un système. Leur élaboration permet la définition de ses principales caractéristiques et fonctionnalités attendues. Afin d'arbitrer le choix des CU qui seront implémentés et ceux qui ne le seront pas, on a recours à une analyse de valeur (AV) dont la nature dépend du projet.

Dans le cas du projet SAFESPOT par exemple, l'AV repose sur une évaluation de l'efficacité en terme de sécurité routière. Une *analyse de l'accidentologie routière européenne* est donc nécessaire, mais pose différents problèmes identifiés par le projet PENDANT [98] comme l'exploitation de données hétérogènes ou encore l'accès à des données détaillées.

L'AV pose aussi un problème méthodologique car elle suppose plusieurs itération du processus de définition des UC et de leur sélection afin de couvrir l'ensembles des besoins initialement prévus.

Enfin pour favoriser une bonne coopération et la créativité des concepteurs, l'élaboration des CU doit se faire en langage naturel au détriment d'une approche formelle. Il est alors difficile d'en garantir la cohérence et la fiabilité alors qu'ils jouent un rôle clé tout au long du processus de développement, et que des erreurs de conception à ce stade du projet seront potentiellement les plus coûteuses [86].

Dans ce chapitre nous présentons les problèmes posés par l'analyse de l'accidentologie européenne. Puis comment nous avons effectué cette analyse au sein du projet SAFESPOT. Ensuite nous présentons notre méthode de conception des CU à partir de cette AV, et définissons le concept central de notre méthode : le **scénario accidentogène**.

4.1 Contexte : le cahier des charges fonctionnel et sa place dans un cycle de développement

Un *cahier des charges fonctionnel* (CdCF) permet de détailler les objectifs et contraintes de réalisation d'un projet. C'est un des éléments constitutif du *cahier des charges* au même titre que la description du planning du projet, de ses ressources financières ou encore des clauses juridiques de sa réalisation. Même s'il ne constitue pas à lui seul un contrat commercial, il peut être considéré comme un document contractuel entre la maîtrise d'ouvrage (i.e. le commanditaire) et la maîtrise d'œuvre (i.e. le prestataire). Il joue à ce titre un rôle essentiel en début de projet en formalisant les choix de réalisation et les responsabilités des différents acteurs du projet.

La qualité du CdCF et sa prise en compte tout au long du processus de réalisation est un critère important de succès du projet comme nous l'avons présenté au chapitre 2. C'est pourquoi, dans le cas de projets aux enjeux économiques ou sociaux importants, ou dans le cas des systèmes complexes et critiques (SCC), le CdCF fait l'objet d'une attention particulière. Il est détaillé, formalisé, et des procédures impliquant le client et les équipes de développement sont mises en place pour en garantir le suivi et la mise à jour à différentes étapes du cycle de développement [1].

C'est une tâche complexe qui nécessite :

– de la créativité pour formuler des propositions,

1. Sa validation tout au long du projet est d'ailleurs à la base des méthodes de développement *en spirale* comme nous l'avons présenté au chapitre 2 section 2.2.4.

- l'utilisation de connaissances dans des domaines d'expertise différents pour évaluer ces propositions,
- une formulation suffisamment simple et claire pour être comprise de tous,
- mais il doit aussi être suffisamment précis et détaillé pour servir de document contractuel entre les différentes parties prenantes du projet.

Mais cette tâche pose plusieurs problèmes :

- L'analyse de valeur (AV) introduite vers la fin des années 40 par Lawrence D. Miles a pour objectif l'évaluation des différents choix de réalisation. Il permet entre autres d'arbitrer le choix des cas d'utilisation qui seront implémentés. La nature et la forme de l'AV dépend du projet et de ses objectifs. Dans le cas du projet SAFESPOT, l'AV repose sur une évaluation de l'efficacité en terme de sécurité routière. Elle nécessite une *analyse de l'accidentologie routière européenne* qui pose différents problèmes identifiés par le projet PENDANT, [98] comme l'exploitation de données hétérogènes ou encore l'accès à des données détaillées.
- Le processus de réalisation et de sélection des cas d'utilisation (CU) décrit dans les méthodes de développement, telles que celles spécifiques aux STI aux États Unis [120] ou en Europe [75], consiste à définir d'abord différents cas d'utilisation puis à en évaluer la pertinence par l'analyse de valeur et des besoins utilisateurs. Ce processus doit être réitéré jusqu'à l'obtention de CU satisfaisants. Or cette approche générique est potentiellement plus coûteuse et moins efficace qu'une approche prenant en compte les spécificités du projet.
- L'élaboration des cas d'utilisation introduits par Jacobson dans les années 90 [72] nécessite la coopération de différents acteurs du projet : les clients, les concepteurs et les utilisateurs. Pour favoriser une bonne coopération et la créativité des concepteurs, leur élaboration doit se faire en langage naturel au détriment d'une approche formelle. Il est alors difficile de garantir que le CdCF sera cohérent et fiable alors qu'il joue un rôle clé tout au long du processus de développement, et que des erreurs de conception à ce stade du projet seront potentiellement les plus coûteuses [86].

4.1.1 Parti pris de l'expressivité et de la communication

Un cahier des charges fonctionnel (CdCF) détaille les limites de ce que doit faire le système et de ce qu'il ne doit pas faire. Sa conception doit, par raffinements successifs, permettre la définition de différents aspects du système :

- La liste des entités internes ou externes au système : respectivement les acteurs et les principales fonctionnalités du système.
- Les besoins des utilisateurs : ils donnent une vision concrète des objectifs du système et des attentes des utilisateurs.
- Les cas d'utilisation : ils mettent en scène le système dans différentes situations d'utilisation et décrivent les différentes actions qui mènent à la satisfaction des besoins des utilisateurs.
- Les contraintes de fonctionnement : elles décrivent les conditions nécessaires à l'implémentation des cas d'utilisation.

Cette étape du cycle de développement d'un système nécessite la coopération entre les clients du système, ses concepteurs et ses futurs utilisateurs. Elle doit donc permettre le dia-

logue entre ces différents acteurs du projets, tout en favorisant la créativité des concepteurs. Ainsi, le niveau de détail doit rester faible pour ne pas perturber les discussions et négociations lors de leur mise en place. Une certaine souplesse d'expression quant à leur formulation est encouragée.

Les méthodes de développement pour STI élaborées aux États-Unis ou en Europe insistent sur l'importance des échanges entre les différents acteurs du projet [121, 75] à cette étape d'un cycle de développement. Pour illustrer notre propos, nous présentons ici un extrait des recommandations que nous avons dans le cadre du projet SAFESPOT [85].

« Les besoins utilisateurs, les cas d'utilisation et les contraintes de fonctionnement sont définis comme suit :

- *Les besoins des utilisateurs sont des énoncés en langage naturel (ou diagrammes informels) de ce que le système est censé fournir et des contraintes dans lesquelles il doit fonctionner.*
- *Les cas d'utilisation définissent un sous-ensemble des fonctionnalités d'un système. Ils sont principalement utilisés pour définir le comportement d'un système sans en préciser la structure interne. (...). Ils ne sont pas nécessairement cohérents, et sont susceptibles d'être exprimés en texte clair et à l'aide de schémas informels si cela est nécessaire pour les rendre plus compréhensibles (...)*
- *Les contraintes de fonctionnement sont axées vers le système et son implémentation, et sont capturées à l'aide de techniques (semi) formelles ; Elles ne sont pas nécessairement faciles à comprendre par l'utilisateur. Les contraintes principales découlent des cas d'utilisation (...) »*

TABLE 4.1 – Extrait des recommandations pour l'élaboration du cahier des charges du projet SAFESPOT

4.1.2 La première étape d'un cycle de développement

L'analyse des besoins, puis la définition des cas d'utilisation et des applications sont les premières étapes du cycle de développement d'un système complexe et critique. Elles fournissent les informations qui permettront ensuite la définition plus détaillée du système au travers des spécifications. Quand l'expression des besoins des utilisateurs par le client est trop abstraite ou trop floue ceux-ci doivent être explicités par les équipes du projet. C'est le cas du projet SAFESPOT où la rédaction du CdCF est à la charge de l'équipe de développement (la maîtrise d'œuvre), qui joue alors le rôle d'assistant à la maîtrise d'ouvrage. La figure 4.1 montre comment la définition des cas d'utilisation et des applications s'inscrit dans son cycle de développement.

Les cas d'utilisation (étape 2 sur la figure 4.1) sont définis à partir de l'analyse des besoins (étape 4 sur la figure 4.1), et présentent une première vision de l'utilisation du système. Afin de valider cette première définition du système, il faut pouvoir évaluer son impact potentiel (étape 3 sur la figure) en mesurant le nombre de besoins utilisateurs satisfaits ou, dans le cas de systèmes de transport intelligents, en pourcentage d'accidents pris en compte. C'est ce qui est appelé l'analyse de valeur (AV) des cas d'utilisation. Si les résultats de l'analyse de valeur n'est pas satisfaisante, la définition des cas d'utilisation et des applications qui en découlent est revue

et corrigée jusqu'à obtenir une AV satisfaisante (c'est à dire plusieurs itérations des étapes 2 et 3 sur la figure). Une première définition des contraintes de fonctionnement du système peut ensuite être déduite de ces cas d'utilisation (étape 4 sur la figure).

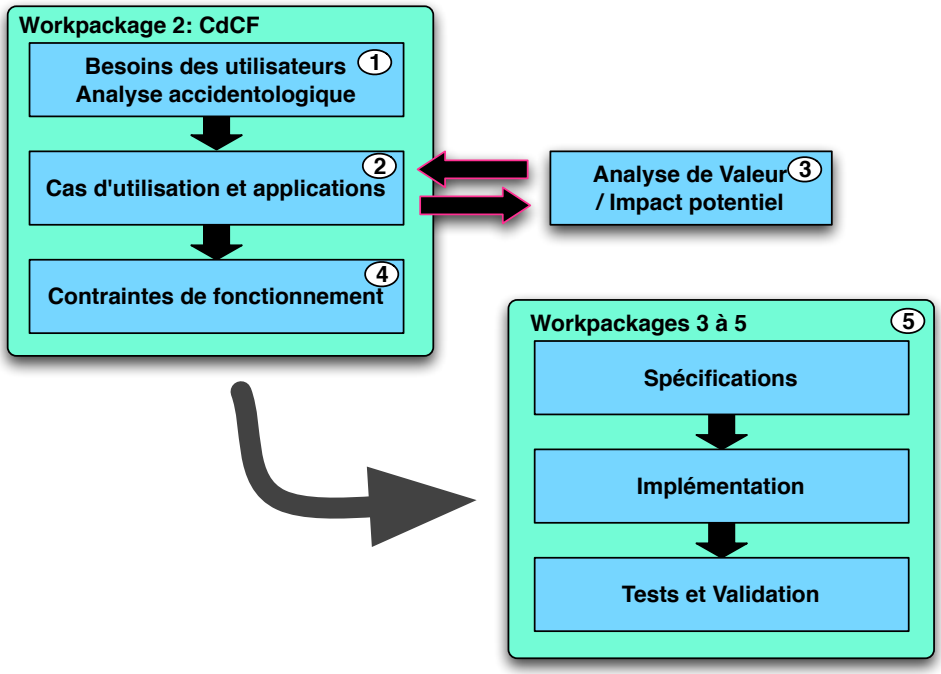


FIGURE 4.1 – Processus de définition du cahier des charges fonctionnel (CdCF) dans le cycle de développement du projet SAFESPOT

La définition des cas d'utilisation nécessite de prendre en compte les besoins des utilisateurs et les caractéristiques du système visé. Par exemple, dans le cas d'un système de transport intelligent visant à améliorer la sécurité des usagers, une *analyse accidentologique* est nécessaire pour valider l'intérêt du système (i.e. pour procéder à l'analyse de valeur).

Ainsi, en fonction du domaine d'application et des objectifs d'un système, la définition des cas d'utilisation doit être effectuée à partir de différents types d'analyses des besoins et différents types d'analyses de valeur. Mais cette approche n'est pas efficace. Elle nécessite plusieurs itérations de l'évaluation de l'impact (étapes 2 et 3 sur la figure 4.1) alors qu'il est possible dans certains cas de déduire directement les cas d'utilisation de l'analyse des besoins, et de réduire le nombre d'itération des étapes 2 et 3 comme nous le présentons dans ce chapitre. Mais d'abord, dans ce qui suit, nous présentons les différents problèmes liés à l'analyse de l'accidentologie sur les routes européennes et ceux liés à la définition des cas d'utilisation.

4.2 Problématique : les difficultés de l'analyse des besoins des utilisateurs

Dans le cas du projet SAFESPOT, l'analyse des besoins des utilisateurs et l'analyse de valeur sont liées à une étude des accidents de la route en Europe. Basée sur l'utilisation de

différentes sources de données accidentologiques européennes cette étude pose différents problèmes.

– D'abord de nombreux détails sur l'origine, les conditions et le déroulement des accidents est nécessaire pour pouvoir proposer des solutions. Or, ce niveau de détail n'est pas disponible au niveau Européen, mais seulement pour certains états ou régions de l'Union Européenne.
– Les sources de données locales sont souvent hétérogènes quant aux détails qu'elles présentent et les méthodes de mesures utilisées.
– Enfin, il n'existe pas de méthode décrivant comment cette analyse peut permettre la définition des cas d'utilisation et des applications.

Dans cette section, nous présentons les problèmes rencontrés pour l'élaboration des cas d'utilisation et l'analyse des besoins dans le cadre du développement d'un système complexe de transport intelligent pour la route.

4.2.1 Problème de l'hétérogénéité des données européennes

Les données accidentologiques disponibles au niveau européen sont fournies par l'institut Eurostat, les bases de donnée CARE et EACS [45, 43] ou encore les projet SAFETYNET, RANKERS ou PENDANT [103, 98, 46]. Elles mettent en évidence les principales similitudes et différences entre pays de l'Union Européenne, et confirment que les réseaux urbains, ruraux et autoroutiers présentent des caractéristiques différentes. La distribution des accidents dans ces zones suit principalement les valeurs moyennes présentées tableau 4.2. Les autres détails disponibles concernent surtout des critères comme l'âge et le genre des conducteurs, ou encore le jour de la semaine où ont lieu les accidents.

Indicateurs / classification	Accidents	Blessés	Morts	Niveau de risque[1]	Indice de mortalité[2]
1er	Urbain 75%	70%	55%	0.57-1.10	18,7%
2ème	Rural 20%	20%	35%	0.43	4,6%
3ème	Autoroute 5%	10%	10%	0.06	1,4%

■ Urbain ■ Rural ■ Autoroute
[1] (morts par million de véhicules/km)
[2] (pourcentage d'accidents mortels sur le nombre total d'accident)

TABLE 4.2 – Distribution moyenne des accidents sur les différents types de routes européennes

Mais des détails plus pertinents pour la définition des cas d'utilisation comme : les causes principales et secondaires, la trajectoire des véhicules, la topologie du réseau, l'état de la route ou de la météo, ne sont pas encore disponibles à un niveau européen, car pas encore pris en compte dans certains pays ou bien mesurés de manières différentes [98, 15].

Un exemple typique des différences de mesures des accidents à travers l'Europe concerne les accidents mortels : dans certains pays, la mort est retenue comme conséquence de l'accident si elle survient jusqu'à trente jours après l'hospitalisation, pour d'autres pays, seuls les morts survenant au moment de l'accident sont prises en compte.

Les données détaillées disponibles au niveau des pays européens sont hétérogènes. Par exemple, seules l'Allemagne et l'Italie fournissent des statistiques concernant les infractions au code de la route au niveau des intersections urbaines. Cela donne une vision restreinte difficile à généraliser au niveau européen. D'autant plus que ces données présentent des disparités : ces infractions représentent 14,8% des accidents mortels en zone urbaine en Allemagne, et 27% en Italie.

4.2.2 Problème de la définition des cas d'utilisation

La conception des applications du projet SAFESPOT suit une approche centrée sur les besoins des utilisateurs. L'annexe technique du projet [68] décrit l'enchainement des différentes étapes de conception, et liste les éléments à définir dans un cas d'utilisation. Mais elle ne donne pas d'indication sur la manière dont les cas d'utilisation peuvent être déduits de cette analyse accidentologique.

En effet, le seul moyen de valider une liste de cas d'utilisation et la définition des applications qui en découle est d'en évaluer l'impact potentiel. Plusieurs itérations sont parfois nécessaires pour valider la définition des applications. En quelque sorte, les concepteurs travaillent par tâtonnements. Dans ce qui suit nous proposons une méthode pour faciliter la conception des cas d'utilisation et simplifier leur évaluation.

4.3 Contribution : une méthode pour l'analyse de l'accidentologie et l'élaboration des cas d'utilisation

Recueil d'informations détaillées
Afin de fournir une analyse des accidents sur les routes européennes qui soit utile au projet, certains détails concernant ces accidents sont nécessaires. En attendant que ces informations soient disponibles au niveau européen, des données plus locales mais plus précises concernant un pays, une région ou une ville européenne doivent être utilisées.

Ces données fournissent différentes caractéristiques des accidents comme : leur impact, leurs sources, leurs causes, les types de véhicules impliqués, l'état du trafic, les trajectoires des véhicules, les conditions météorologiques ou les infractions au code de la route [7, 50].

Par exemple, les données fournies par l'institut Néerlandais de recherche pour la route [47] font la distinction entre les différents réseaux et fournissent aussi des informations sur les conditions des accidents (voir figure 4.2).

Les données de la ville de Turin en Italie [70], des autoroutes françaises [3], de la région Bretagne en France ou des zones urbaines allemandes, belges ou néerlandaises [125, 47], ont été utilisées. Des études spécifiques concernant les sorties de route [5], les intersections [73] ou les vitesses excessives [101] sont aussi prises en compte.

Définition 4.3.1. *La **cause principale** d'un accident de la route est le comportement humain, la condition environnementale ou la défaillance mécanique qui est considérée comme étant à l'origine de l'accident. On la distingue des **causes secondaires** qui favorise l'occurrence d'une **cause principale** ou en aggrave les conséquences.*

On constate, par exemple, qu'une vitesse excessive et le manque d'attention des conducteurs sont observés comme étant des causes principales dans un grand nombre d'accidents,

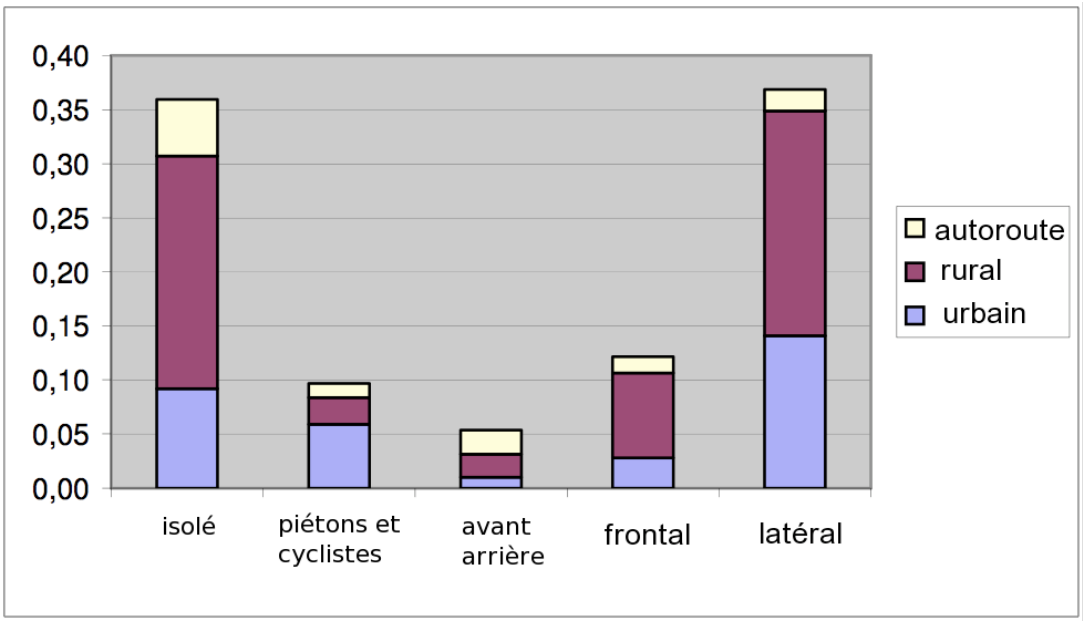


FIGURE 4.2 – Distribution des accidents par zone aux Pays-Bas en 2000

qu'ils sont souvent combinées et accompagnés d'autres facteurs secondaires, comme une faible adhérence de la chaussée ou un manque de visibilité. Mais leur impact sur les trois types de réseaux (urbain, rural et autoroutier) n'est pas le même.

L'analyse détaillée des accidents de la route en Europe est effectuée en distinguant trois types de routes : les réseaux urbain, rural et autoroutier, car ils présentent des caractéristiques distinctes en terme de dynamique de circulation, topologie du réseau, types d'entités rencontrées et comportement des conducteurs.

La vitesse excessive est un problème plus important sur les routes rurales et les autoroutes, où elle est la cause de 40% des accidents mortels, contre 10% pour les zones urbaines. Par contre le manque d'attention est particulièrement critique en zone urbaine à cause de la complexité du réseau et du nombre de véhicules, piétons ou cyclomoteurs [48]. Sur autoroute, l'inattention est plutôt due à la longueur des trajets et à la monotonie des routes [3].

Maîtrise de l'hétérogénéité des données

Pour pallier le problème de disponibilité de certaines informations au niveau européen, l'utilisation de données plus locales est donc nécessaire. Mais ces données restent très hétérogènes quant aux détails qu'elles fournissent et à leur portée. En outre, rien n'indique comment s'en servir pour définir les cas d'utilisation du système.

La méthode que nous proposons s'effectue en plusieurs étapes que voici (la méthode est aussi présentée figure 4.3).

1. Les données doivent être classées par type de réseau : rural, urbain ou autoroutier car leurs caractéristiques accidentogènes sont très différentes.
2. Elles sont ensuite regroupées à partir : soit des causes principales des accidents, soit en fonction de facteurs secondaires souvent impliqués.

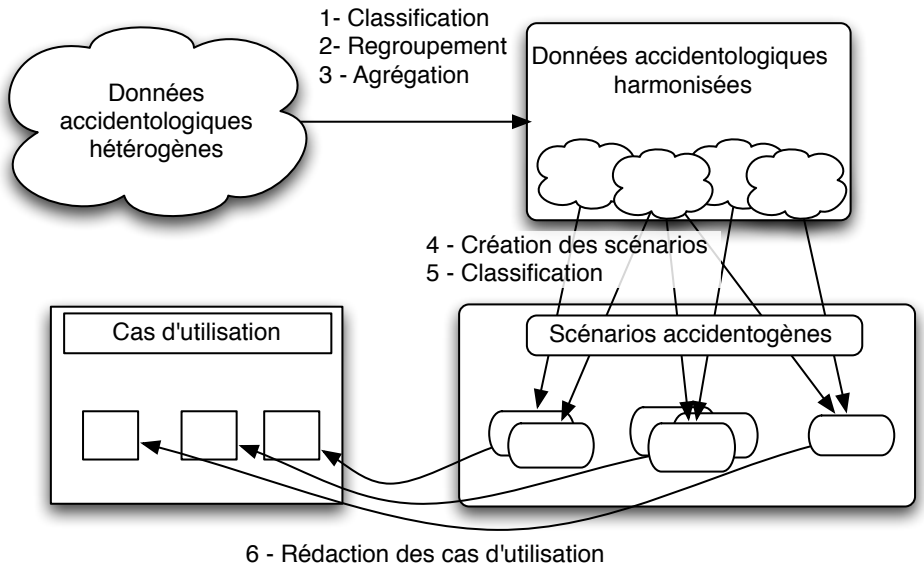


FIGURE 4.3 – Méthode basée sur les scénarios accidentogènes

3. Si plusieurs sources de données sont disponibles pour un même facteur principal ou secondaire, alors la moyenne est faite entre ces deux sources [2].
4. Les données sont ensuite regroupées dans des tableaux présentant le facteur principal considéré et ses facteurs secondaires liés (en faisant toujours la distinction entre les différents types de réseaux, urbain, rural ou autoroutier). On obtient alors ce nous appelons un **scénario accidentogène**.
5. Ces scénarios accidentogènes sont alors classés en différentes catégories en fonction des **paramètres** et des **acteurs** impliqués : Obstacles, Erreurs de jugement et inattention, Infractions, Conditions environnementales extrêmes et Collisions, etc.
6. Puis, pour chaque **scénario accidentogène** défini, on définit un cas d'utilisation qui le prend en compte.

Définition 4.3.2. *Un **scénario accidentogène** est une liste de causes principales et/ou secondaires d'accidents liées par un paramètre commun, et présentant le pourcentage d'accidents dans lesquels ces causes sont impliquées.*

Cette relation entre **scénario accidentogène** et cas d'utilisation garantit la pertinence du cas d'utilisation et permet une évaluation de son impact potentiel.

Pour illustrer cette méthode, voici un exemple de son utilisation. Une vitesse excessive est la cause principale de nombreux accidents impliquant très souvent une cause secondaire comme une mauvaise évaluation des conditions météorologiques ou une mauvaise évaluation des distances. Cela justifie la création d'un scénario dédié à ce facteur, que nous présentons figure 4.3.

D'un autre côté, la mauvaise évaluation des conditions météorologiques est un facteur secondaire souvent impliqué dans les accidents, ce qui justifie aussi l'élaboration d'un scénario accidentogène pour ce facteur.

2. Cette approche est bien sûr imparfaite mais dans la mesure où cette moyenne n'est pas disponible au niveau européen, c'est la meilleure approche possible.

Nom	Vitesse et conduite dangereuse	
Facteur	Vitesse inadaptée, conduite dangereuse	42,00%
Facteur aggravants	Vitesse	
Type de route	Autoroute	
Extension 1		
Type	Vitesse excessive	19,00%
Extension 2		
Type	Dépassement dangereux	12,00%
Extension 3		
Type	Vitesse inadaptée au trafic	5,90%
Extension 4		
Type	Vitesse inadaptée à la météo	3,40%
Extension 5		
Type	Inter-distance dangereuse	1,70%

TABLE 4.3 – Scénario accidentogène "Vitesse et Conduite Dangereuse" pour autoroutes

Les scénarios sont ensuite définis en détaillant les facteurs secondaires ou principaux liés les plus souvent rencontrés et le pourcentage d'accidents où ces facteurs sont impliqués. Un exemple est présenté au tableau 4.3 qui montre le scénario élaboré pour les vitesses excessives qui représentent 42% des accidents mortels sur autoroute.

Les scénarios accidentogènes sont définis séparément pour chacune des zones du réseau routier : urbaine, rurale ou autoroutiaire. Voici par exemple les scénarios accidentogènes que nous avons définie pour la zone urbaine, présentés tableau 4.4. Ils présentent le pourcentage d'accidents mortels qui y sont liés par rapport au total d'accidents mortels sur les trois zones du réseau. Dans la mesure où plusieurs causes sont impliquées dans l'occurrence d'un accident, la somme des pourcentages présentés ici est supérieure au pourcentage d'accidents mortels dans la zone urbaine.

Nous présentons dans la section suivante la manière dont les cas d'utilisation et des applications du système sont déduits de ces scénarios accidentogènes.

4.4 Résultats : définition des cas d'utilisation et des applications

4.4.1 Cas d'utilisation

Les cas d'utilisation sont une description détaillée de la manière dont le système doit répondre aux besoins des utilisateurs décrits dans les scénarios accidentogènes définis en section 4.3.

Inattention et fatigue du conducteur	
- fatigue et réaction trop lente	5,1%
- distraction et inattention	14,3%
Mauvaise utilisation de la route	
- demi-tour dangereux	16,9%
- conduite sur la mauvaise voie	6,1%
- changement de voie	1,1%
Interaction avec d'autres véhicules	
- mauvaise distance de sécurité	12,5%
- dépassement dangereux	2%
- arrêt brutal	1%
Vitesse inadaptée	
- non respect de la vitesse autorisée	11,5%
Intersection	
- non respect du code	21%
- piéton en train de traverser	4%

TABLE 4.4 – Scénarios accidentogènes en zone urbaine

Durant ce processus, vingt-quatre cas d'utilisation concernant l'infrastructure sont définis [85] et classés en catégories : Obstacles, Erreurs de jugement et inattention, Infractions, Conditions environnementales extrêmes, Collisions, etc.

La définition d'un cas d'utilisation est constituée de trois parties :

– **La première partie** présente les principaux aspects d'un cas d'utilisation, à savoir : son titre, sa finalité, sa pertinence, la source du problème, les conditions de succès ou d'échec, la condition de déclenchement et le lien avec les données accidentologiques. Un exemple est fourni tableau 4.5. Il concerne les sur-accidents dus à la présence d'un véhicule à l'arrêt sur une partie de la chaussée. Son objectif est double : éviter les sur-accidents et démarrer la prise en charge de l'obstacle (ici, un véhicule accidenté). Son impact potentiel découle du *scénario accidentogène* dont il est issu, ce qui rend son évaluation automatique. En l'occurrence, les sur-accidents concernent 14% des accidents mortels sur autoroute.
  Cette partie du cas d'utilisation décrit aussi l'événement déclencheur, les conditions de succès et d'échec du scénario. Ici, le cas d'utilisation est un succès si les véhicules arrivant à proximité de l'obstacle sont alertés, si une alerte est envoyée au centre de contrôle pour la prise en charge de l'obstacle, et enfin, si un embouteillage est évité.

– **La deuxième partie** donne une description, étape par étape, des interactions entre les entités du système pour résoudre le problème (tableau 4.6). En l'occurrence, cinq étapes sont identifiées et peuvent se résumer ainsi : détection, recueil d'informations supplémentaires, analyse, envoi des alertes et recommandations, modification éventuelle des conditions de circulation.
  Ces deux premières parties présentent une vue à la fois discrète et continue du système. C'est un moyen de capturer une discrétisation gros grain, une abstraction des compor-

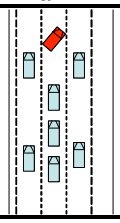
Case Name	Accident as an obstacle
Case ID	SP5_UC13_v1.0
Status	Final
Short description	 A vehicle has an accident on the road. All vehicles concerned (in the vicinity or incoming) are warned and possibly asked to change lane in order to avoid cumulative accidents. Furthermore, infrastructure manager or assistance manager must be warned to react accordingly.
Purpose	The first objective is to avoid cumulative accidents, secondary objective is to quickly manage the accident (assistance vehicle …)
Rationale	The problem of accident and cumulative accident on interurban road presents high gravity with respect to speed. Furthermore it has a large impact in the media and generate large traffic jams
Authors	Sébastien Glaser – LCPC / Fabien Bonnefoi - Cofiroute
Driving environment	Inter-Urban and Rural Roads
Vehicle probe type	All
Risk's source	First level off risk in the vicinity is the lack, or late, of reaction of driver. Second level of risk is cumulative accidents, with vehicles crashing on stopped ones
Successful end condition	Warn the driver of accident, quick emergency response, cumulative accidents avoided
Failed end condition	First kind of failed end condition could be cumulative accident, second kind is traffic jam
Trigger	Detect an accident condition
Frequency of occurrence	Accident as an obstacle represents about 14.4% of fatal accident on highway. Moreover, it is a large source of congestion

TABLE 4.5 – Le cas d'utilisation no 13 : introduction

Primary Actor	Broken vehicle	
Secondary Actor(s)	Oncoming vehicle	
Scenario Description	**Step**	**Action**
	1	Detection of accident and stopped car : ▪ The vehicle is equipped with V2V/V2I communication and can send a message with position and relevant information, either automated or driver activated ▪ A vehicle crosses the accident zone and send a warning, either automated or driver activated ▪ The infrastructure detects the accident using sensors (camera…)
	2	The infrastructure grabs the information and acquires local map of vehicle in the vicinity
	3	An infrastructure based application analyses the situation and detects possible conflict at lane level
	4	Warning message is sent to concerned vehicle, with conflict zone and possible actions, as lane changing manoeuvre.
	5	Depending on the situation, upstream traffic could be re-routed.
Extensions	**Step**	**Action**
	4b	Infrastructure sends the message to assistance or emergency centre and starts the safety margin for emergency vehicle use case

TABLE 4.6 – Le cas d'utilisation no 13 : détails

tements attendus du système. Ce niveau d'abstraction très élevé est particulièrement propice à une première analyse formelle du système. Mais ils sont exprimés en langage courant. L'exploitation des cas d'utilisation pour procéder à une analyse formelle du système est présentée dans la deuxième partie de ce chapitre.

– **La troisième partie** contient la définition détaillée des besoins utilisateurs et des contraintes sur le système (tableau 4.7). Les *besoins utilisateurs détaillés* correspondent aux besoins des utilisateurs dans la situation décrite par le cas d'utilisation. Ils peuvent être évoqués par différents cas d'utilisation. Ils détaillent des contraintes sur le système du point de vue le plus important : celui de l'utilisateur. Les contraintes de fonctionnement décrivent les conditions techniques que doit remplir le système pour être capable de prendre en charge le cas d'utilisation. Les besoins utilisateurs détaillés et les contraintes de fonctionnement présentent respectivement les objectifs et conditions de réalisation du cas d'utilisation.

User Needs	*Broken vehicle :* ▪ To avoid a collision with another vehicle ▪ To limit speed of oncoming vehicles *Oncoming vehicles:* ▪ Be warned of an accident in the vicinity, especially in low visibility condition ▪ Be warned on a possible pedestrian on the road event ▪ Getting recommendation on lane to follow and speed recommendations
Corresponding requirements	▪ Emergency call with extended data (localisation, type…) ▪ Accident detection ▪ Warning Message management ▪ Lane Level positioning ▪ Local map around accident ▪ V2I Communication
Related UCs	Pedestrian as an obstacle Assistance vehicle signalling an event
Open issues	▪ Can the "emergency call" be always available, with extended data ▪ What is the confidence on vehicle and driver activation
Comments	

TABLE 4.7 – Le cas d'utilisation no 13 : expression des besoins et contraintes

Les cas d'utilisation peuvent concerner des situations différentes, mais proposer la même solution en terme de réaction du système. Par exemple les différents types d'obstacles, tels qu'un véhicule lent, accidenté ou un embouteillage, suscitent le même comportement du système. Ces cas d'utilisation sont présentés en annexe A.4 page 165.

Comme on le voit, l'élaboration des cas d'utilisation suit une approche méthodique. Elle permet entre autres :

– de faire le lien entre les scénarios accidentogènes et le système SAFESPOT,
– de donner une description des principales étapes de fonctionnement du système,
– de définir une liste détaillée des besoins utilisateurs,
– de définir les contraintes de fonctionnement.

Les cas d'utilisation permettent une approche à mi-chemin entre un discours informel et une vision plus détaillée et plus technique du système. Ils facilitent donc les discussions et prises de décision en début de projet, tout en préparant les prochaines étapes du processus de développement. Ils ont donc une grande importance pour la suite du projet mais leur aspect informel

rend difficile leur évaluation, leur vérification et leur validation. Dans la deuxième partie de ce chapitre nous montrerons comment il est néanmoins possible de les valider formellement.

Dans ce qui suit nous présentons des exemples d'applications SAFESPOT déduites de ces cas d'utilisation et leur justification d'un point de vue accidentologique.

4.4.2 Définition des applications

SAFESPOT est un système coopératif entre véhicules et infrastructure. Il est composé d'applications basées sur l'infrastructure ou sur le véhicule. C'est-à-dire, qu'au delà des échanges de données recueillies par les capteurs, l'infrastructure et les véhicules peuvent générer et échanger des alertes. Quand une alerte est déclenchée par l'infrastructure (la décision est prise côté infrastructure), on parle d'application basée sur l'infrastructure. Il en va de même pour les véhicules.

Les applications SAFESPOT se répartissent donc en deux catégories principales : les applications mises en place dans les véhicules [114] et celles déployées dans l'infrastructure [124, 106]. Certaines applications de ces deux catégories couvrent le même type de danger, et d'autres sont spécifiques aux caractéristiques de l'entité où elles sont implémentées. Ainsi, les applications peuvent être déclenchées dans différentes situations de couverture réduite ou de faible pénétration du système.

ID	**Application Name**
SP5_H&IW	Hazard and Incident Warning
SP5_Spa	Speed Alert
SP5_Rdep	Road Departure
SP5_IRIS	Intelligent Cooperative Intersection Safety System
SP5_SMAEV	Safety Margin for Assistance and Emergency Vehicle
SP4_VRUAA	Vulnerable Road User Detection and Accident Avoidance
SP4_RCS	Road condition Status
SP4_SLSD	Speed Limitation and Safety Distance
SP4_CUWA	Curve Warning
SP4_LCM	Lane Change Manoeuvre
SP4_SO	Safe Overtaking
SP4_HOCW -RECW-FCW	Head on / Rear End /Frontal Collision Warning
SP4_RIS	Road Intersection Safety

TABLE 4.8 – Les applications SAFESPOT

Le tableau 4.8 présente la liste des applications de SAFESPOT. Les applications dont l'identifiant est préfixé par 'SP5' sont celles déployées dans l'infrastructure. Celles qui sont préfixées par 'SP4' sont déployées dans les véhicules.

Comme nous pouvons le constater, ces applications concernent des situations dangereuses à court terme, et ne prennent pas en charge les tâches de choix d'itinéraires ou d'information concernant le nombre de places de parking libres. Ces applications montrent l'orientation de

SAFESPOT, dédié aux problèmes de sécurité, contrairement aux projets CVIS et COOPERS (comme nous l'avons présenté au chapitre 2).

INITIAL LIST OF APPLICATIONS (Technical Annex)	FINAL LIST OF APPLICATIONS
Speed alert and road departure prevention	Speed Alert – SpA
Smart signalling for safety enhancement	Road Departure Prevention – RDep
Safety margin for Assistance and emergency vehicles	Safety Margin for Assistance and Emergency Vehicles – SMAEV
Safe urban intersection	Co-operative Intersection Collision Prevention System – CICPS
Hazard and incident warning	Hazard and Incident Warning – H&IW

TABLE 4.9 – Définition des applications de l'infrastructure avant et après l'analyse des besoins

Une liste initiale d'applications a été définie au lancement du projet avant que ne soit effectuée l'analyse accidentologique. L'élaboration des scénarios accidentogènes nous a permis de définir plusieurs cas d'utilisation d'où découle une nouvelle vision de ces applications. Le tableau 4.9 présente un extrait de l'amendement au projet qui a été rédigé pour faire état de cette nouvelle définition des applications. Nous présentons ci-dessous la définition de trois d'entre d'entre elles, modifiées suite à l'élaboration des scénarios accidentogènes.

"Intersection Collision Avoidance" :
L'analyse des données accidentologiques montre le besoin d'une application spécifique afin d'améliorer la sécurité des carrefours urbains ou ruraux car 30% des accidents mortels et 40% des accidents avec blessés surviennent à ces carrefours sur le réseau Italien par exemple.

L'analyse accidentologique et l'élaboration des scénarios accidentogènes a mené à étendre le champ d'action de cette application initialement dédiée aux zone urbaines comme on peut le constater sur le tableau 4.9.

Cette application est clairement centrée sur des zones (points noirs ou *"black spots"*) statiques et exploite des communications et un calcul en temps réel à travers un réseau local à haute vitesse. Néanmoins, l'infrastructure va diffuser des informations aux conducteurs non seulement à des points spécifiques, mais aussi dans des zones plus larges au delà de l'intersection.

L'application prend aussi en compte les piétons ou d'autres utilisateurs vulnérables de la route qui ne sont pas équipés du système SAFESPOT, mais ont besoin de recevoir des informations à travers les panneaux de signalisation ou les feux.

"Speed Alert" :
L'application "Speed Alert" prend en charge l'ensemble des problèmes liés à la vitesse mis

en évidence par les scénarios accidentogènes. Elle fournie des recommandations de vitesse aux conducteurs à partir d'évaluations en temps réel de paramètres tels que : les conditions météorologiques, l'état de la surface de la route, la topologie de la route, la vitesse du trafic et la présence d'événements (comme un chantier, un accident, un embouteillage, la présence de piétons, une déviation, etc., qui lui sont signalés par l'application H&IW).

Par exemple, si un véhicule arrive sur une section de route sujette à un embouteillage, cette application alerte le conducteur s'il ne ralentit pas suffisamment tôt.

Cette application est justifiée dans le sens où une vitesse excessive est une cause principale dans 40% des accidents survenant sur les routes rurales ou les autoroutes. Cette application devait initialement prendre en charge les problèmes d'excès de vitesse et de sortie de route. L'élaboration des scénarios accidentogènes nous a amenés à la séparer en deux applications distinctes dans la mesure où les paramètres et conditions d'occurrence de ces problèmes sont différents. Les solutions proposées par les cas d'utilisation pour prendre en charge ces problèmes ont mis en évidence le besoin de distinguer ces deux applications.

"Hazard and Incident warning" :
L'objectif de cette application est d'alerter les conducteurs en cas d'éléments ou d'événements dangereux sur la route. Les événements considérés sont les plus pertinents en termes de sécurité : accidents, obstacles divers, embouteillage, présence de piétons ou d'animaux, véhicule circulant à contresens ou présence de chantier. Cette application analyse aussi les conditions environnementales qui pourraient réduire le coefficient d'adhérence de la route ou réduire la visibilité des conducteurs.

Basée sur la coopération entre véhicules et avec l'infrastructure, cette application diffuse des alertes aux conducteurs et fournit des informations détaillées de la situation au système SAFESPOT. En fonction des dangers qu'elle a détectés, elle fournit aux autres applications et aux conducteurs une description détaillée et actualisée de la route : voie bloquée, limitation de vitesse temporaire , déviation, etc.

Cas d'utilisation et sous-applications de H&IW : L'application H&IW couvre un certains nombre de cas d'utilisation à travers différentes sous-applications.

- **H&IW_01** : Obstacle sur la route (en anglais :"*Obstacle on the road*") alerte le conducteur de la présence d'obstacles statiques ou dynamiques sur la route comme un véhicule accidenté, un véhicule lent, un piéton ou un embouteillage.
- **H&IW_02** : Conduite à contresens (en anglais :"*Wrong way driving*") alerte les conducteurs en cas de véhicules circulant à contre sens sur leur voie à cause d'un dépassement ou d'une d'une erreur de conduite.
- **H&IW_03** : Condition météorologique dangereuse (en anglais :"*Abnormal weather conditions*") alerte le conducteur en cas de présence de danger du aux conditions météorologiques (i.e. pluie, verglas, brouillard) qui induisent une réduction de l'adhérence de la route ou une réduction de la visibilité.

Cette décomposition est présentée figure 4.4 (la liste des cas d'utilisation couverts par l'application est non exhaustive).

H&IW	H&IW_01:obstacles (dyamic)	SP5_UC12: Assistance vehicle signaling a trafic event
		SP5_UC11: Safety margin for maintenance vehicle
		SP5_UC14: Traffic jam as obstacle
		SP5_UC15: Traffic jam as obstacle 2
		SP5_UC17: Pedestrain on the road
	H&IW_01:obstacles (static)	SP5_UC13: Accident as obstacle
		SP5_UC16: Deviation for road work
	H&IW_02:wrong way drinving	SP5_UC23: Overtaking on a two way road
		SP5_UC34: Ghost driver
	H&IW_03:Reduced Friction or	SP5_UC41: Prevention of the lack of adherence
		SP5_UC43: Entering a tunnel
		SP5_UC44: Exiting a tunnel
		SP5_UC45: Sudden reduced visibility

FIGURE 4.4 – Sous-application de H&IW et cas d'utilisation couverts

4.4.3 Évaluation de l'impact des applications

Pour évaluer l'importance des applications, nous avons effectué une étude de leur impact potentiel.

Impact potentiel

Le système SAFESPOT fournit des alertes aux conducteurs mais ne contrôle pas les véhicules. Il est donc délicat de connaître à l'avance le niveau d'acceptation de ces alertes par les conducteurs. Par exemple, un conducteur en excès de vitesse peut ignorer les alertes qui lui sont délivrées. De plus, la définition des applications ne permet pas de connaître le niveau de détection effectif des capteurs. Par exemple, les systèmes de détection automatique d'obstacle sur la route à l'aide de caméras vidéo ne détectent pas la totalité des obstacles.

Ce que nous appelons impact potentiel correspond au nombre d'accidents pris en charge par l'application et non à ceux qui seront effectivement évités. Pour chacune des applications, nous avons regroupé dans un tableau la liste des cas d'utilisation qu'elle couvre et le détail de son impact potentiel, pour les trois types de routes (urbain, rural, autoroute). Un exemple pour l'application *Road Departure Prevention* est présenté tableau 4.10.

Nous avons ensuite évalué l'importance de ces applications les unes par rapport aux autres. Cette évaluation est basée sur la comparaison des impacts potentiels des cas d'utilisation. D'autres facteurs entrent en jeu. En effet, le système SAFESPOT n'est déployé que sur certaines parties du réseau. Par exemple, en zone urbaine, il n'est déployé qu'aux abords d'une intersection. Sur le reste du réseau urbain l'importance des cas d'utilisation est donc nulle. Par ailleurs, certains cas d'utilisation ont une importance qui n'est pas liée à leurs conséquences en termes d'accidentologie. Par exemple, garantir la sécurité du personnel travaillant sur les routes.

Les tableaux 4.11 et 4.12 (pages 85 et 86) présentent l'importance (nommée "*priority*" dans les colonnes à droite du tableau) des cas d'utilisation déduits des scénarios accidentogènes. Chaque type de route est évalué séparément.

Road departure prevention	
Scope:	Accident causes: All causes where a road departure can be the main or an aggravating factor and where the application speed alert is not strongly relevant.
Related Use cases:	- Obstacles : SP5_UC16 - Misjudgement : SP5_UC21 - Rule violation: SP5_UC32 - Critical environment conditions : SP5_UC41, SP5_UC42, SP5_UC45 - Safety improving driver assistance :
Potential impact (The Road Departure application will impact on all the 3 main environments)	
Urban area	Scenario : Driver's lack of vigilance - leaving the carriageway: 20% Supposing a 100% reliable system working on all the cars, then we could expect that the Road Departure application would cover around 20% of the German urban lethal accidents.
Rural area	The French data analysis on rural roads reports that between 30% and 40% of the accidents are of "Run-off road" type. These figures give an idea of the potential impact of a 100% reliable SAFESPOT system working on all the cars and in all the areas.
Motorway	Scenario : Driver's lack of vigilance - Veer of the lane 25% A 100% reliable SAFESPOT system should be able to manage about 25% of road departure accidents in Italian and French motorways.

TABLE 4.10 – Analyse de l'impact potentiel de l'application "Road Departure Prevention"

L'importance s'interprète comme suit :
– l'échelle est de 0 à 10, 10 correspondant à l'importance d'implémentation la plus élevée ;
– avec une importance supérieure à 6 sur un type de route, le cas d'utilisation est automatiquement implémenté (ce qu'indiquent les couleurs du tableau) ;
– avec une importance inférieure à 7 pour tous types de route, le cas d'utilisation n'est implémenté qu'en fonction de son classement total ;
– un "S" indique que le cas d'utilisation doit être implémentés car il concerne une catégorie spéciale d'usagers (comme le personnel travaillant sur les routes), son importance est alors arbitraire.

Ces tableaux indiquent, pour chaque cas d'utilisation, la ou les applications le couvrant. Ils présentent également les facteurs principaux et secondaires liés aux cas d'utilisation et issus des scénarios accidentogènes.

4.4.4 Liste des acteurs et contraintes de fonctionnement

L'élaboration des cas d'utilisation permet d'identifier la liste des acteurs et entités du système et de définir la liste des contraintes de fonctionnement. Dans le cadre du projet SAFESPOT, ce travail est effectué suivant la méthodologie FRAME (que nous avons présentée au chapitre 2 page 34).

Liste des acteurs et entités du système

Les cas d'utilisation ont été définis par différentes équipes au sein du projet. Celles-ci utilisent des termes différents pour évoquer le système et les acteurs impliqués dans les cas d'utilisation.

La méthodologie FRAME propose un moyen d'harmoniser les cas d'utilisation au travers de la définition des acteurs et entités du système. Un exemple est présenté figure 4.5. Il s'agit de l'acteur nommé *SAFESPOT end-user*. La définition de son nom (*name*) comporte l'indication des termes utilisés dans les différents cas d'utilisation (*also named*). Comme on peut le constater, les termes *Vehicle*, *Driver*, *Car*, *Truck* utilisés dans les différents cas d'utilisation, désignent tous le *SAFESPOT end-user*. La spécification des acteurs et entités suivant la méthode FRAME doit aussi comporter la description de :
– leur rôle,
– un exemple de son évocation dans un cas d'utilisation,
– sa relation avec le système.

Cette approche permet d'harmoniser les cas d'utilisation et facilite la coopération entre les équipes du projet.

Liste des contraintes

Les cas d'utilisation décrivent un ensemble de contraintes de fonctionnement du système. La méthodologie FRAME offre un moyen de les formaliser. Un exemple est présenté figure 4.6. Chaque contrainte doit d'abord indiquer sur quelle partie du système elle porte. Elle peut aussi porter sur l'ensemble du système. La contrainte doit ensuite être exprimée de manière concise. Une liste est ensuite constituée présentant :

Actor	**Name**	SAFESPOT end-user Also named : Vehicle, Driver, Car, Truck
	Definition	A road user that is using the SAFSPOT System and is generally driving a motor vehicle.
Role	**Name**	SAFESPOT client
	Definition	The SAFESPOT end-user is the main client of the SAFESPOT system. He will receive warning, recommendations and any other kind of safety service.
Comments		If no particular precision is given, any car, vehicle, truck or driver mentioned in the use cases are to be considered as a SAFESPOT End User.
Example context (optional)		Driver in car with on-board SAFESPOT system receives warnings related to an accident
Relation to System component		SAFESPOT Client (SC)

FIGURE 4.5 – Exemple de définition des acteurs et entités du système SAFESPOT

– un identifiant pour chacune des contraintes (*ID*),
– un nom donné à la contrainte (*Name*),
– la définition de la contrainte (*Requirement Definition*),
– l'élément du système qui requiert la contrainte (*Application*),
– un commentaire au travers desquels les différentes équipes impactées par la contrainte, peuvent interagir pendant la phase de validation de ces contraintes (*Observation*),.

L'utilisation de la méthodologie FRAME permet ainsi de spécifier de manière semi-formelle les contraintes de fonctionnement du système.

ID	Name	Requirement Definition	Application	Observation
SP5_RQ 04_36_v1 .0	Prediction of Trajectories	The system shall be able to predict the vehicles' trajectories.	H&IW_01 EXTENDED	Considered: This is foreseen for the dangerous overtaking sub-application (intention to overtake).
SP5_RQ 05_36_v1 .0	Identify Safety Critical Situations	The system shall be able to identify safety critical situations surrounding a critical point e.g. an urban intersection.	H&IW ALL	Considered: Identification of the safety-critical conditions is the role of the Scenario Analysis module in all H&IW sub-applications
SP5_RQ 52_36_v1 .0	Static Vehicle Data	The system shall receive static vehicle data like width, length, type of vehicle, mass.	H&IW ALL	Considered: This data (deriving from the VANET beaconing) will be obtained by querying the LDM.

FIGURE 4.6 – Exemple de contraintes de fonctionnement au format FRAME

Use Case ID	Use Case Headline	Related Application	Accident cause, type, kind or other	Priority			
				U	R	M	Total
Obstacles							
SP5_UC11	Safety margin for maintenance vehicle on snow removal or salting operation.	Safety margin for assistance and emergency vehicles	Weather		5	4(S)	9
SP5_UC12	Assistance vehicle patrolling or signalling a traffic event on a road	Safety margin for assistance and emergency vehicles	weather / obstacle		4	7(S)	11
SP5_UC13	Accidents as obstacles	Hazard and incident warning / Safety margin for assistance and emergency vehicles	Vehicle obstacle / stationary vehicle		7	8	15
SP5_UC14	Traffic jams as obstacles (slow moving vehicles)	Hazard and incident warning	Vehicle obstacle		6	7	13
SP5_UC15	Traffic jams as obstacles, from an accident with low visibility	Hazard and incident warning	Vehicle obstacle / stationary vehicle		7	7	14
SP5_UC16	Deviations for road-works	Hazard and incident warning, Road Departure	Obstacle / collision with infrastructure		5	5	10
SP5_UC17	Pedestrian as an obstacle (non urban area)	Hazard & incident warning	Pedestrian obstacle		8	8	16
Misjudgement ("wrong behaviour")							
SP5_UC21	Prevention and management of the inattention of the driver	Road Departure, Road Departure	Drowsiness / Inattention		2	10	12
SP5_UC22	Safe signalized intersection (crossing, turning)	Safe urban intersection	Intersection black spot	8	8		16
SP5_UC23	Overtaking on a two way road (infra-scenario only)	Speed alert	Side / Head tail / Frontal		8		8

TABLE 4.11 – Liste des cas d'utilisation et priorités d'implémentation - Partie 1/2

Use Case ID	Use Case Headline	Related Application	Accident cause, type, kind or other	Priority			
				U	R	M	Total
Rule violation							
SP5_UC31	Safe signalized intersection (red light violation)	Safe urban intersection	Intersection black spot	6	5		11
SP5_UC32	Prevention of Driver Excessive Speed	Speed alert / Road Departure	Rule violation / Road departure / Drowsiness /		8	9	17
SP5_UC33	Right of way (stop sign), not signalized roads	Safe urban intersection	Priority / Stop signs	8	7	2	17
SP5_UC34	Ghost drivers (wrong way driving)	Speed alert, Hazard and incident warning	Wrong way		4	7(S)	11
Critical environment conditions (incl. infrastructure)							
SP5_UC41	Prevention for the Lack of adherence of the road	Hazard and incident warning / Road departure	Weather: Rain/ snow / etc…		8	5	13
SP5_UC42	Prevention of Driver excessive Speed (critical environment.)	Speed alert / Road Departure	Weather / Road departure / Drowsiness /		8	7	15
SP5_UC43	Entering into a tunnel	Hazard and incident warning	Luminosity / head tail		6	5	11
SP5_UC44	Exiting from a tunnel	Hazard and incident warning	Luminosity / head tail		6	5	11
SP5_UC45	Sudden reduced visibility (geometric)	Hazard and incident warning / Road Departure	Luminosity / head tail		7	5	12
Safety improving driver assistance (incl. safety margin)							
SP5_UC51	Assistance vehicle signalling a traffic event on a road	Hazard and incident warning / Safety margin for assistance and emergency vehicles			5	5(S)	10
SP5_UC52	Emergency vehicle is approaching a controlled intersection	Safe urban intersection / Safety margin for assistance and emergency vehicles	Intersection	5(S)			5
SP5_UC53	Junction of two motorways (merging and separation)	Speed alert	Side / Head tail			5	5
SP5_UC54	Access or exit of a motorway (at junction)	Speed Alert / Road Departure	Excessive speed			5	5
SP5_UC55	Driver assistance for complex urban intersections	Safe urban intersection	Intersection black spot	7			7

TABLE 4.12 – Liste des cas d'utilisation et priorités d'implémentation - Partie 2/2

4.5 Synthèse

Partant des recommandations de la Commission Européenne, l'analyse accidentologique a prouvé son efficacité en donnant une vision détaillée des dangers que peuvent rencontrer les conducteurs sur les routes européennes.

La difficulté principale est de trouver des données détaillées, puis de gérer l'hétérogénéité de ces données venant de différents pays. Ce problème est surmonté grâce à notre définition et à notre classification de *scénarios accidentogènes* d'où découlent la définition des cas d'utilisation. Nous montrons aussi comment cette analyse des besoins permet d'effectuer l'analyse de l'impact potentiel des applications.

Les bases de données CARE et EACS, ainsi que les projets SAFETYNET, RANKERS ou PENDANT, visent à donner une vision détaillée et harmonisée de l'accidentologie européenne. Ils constituent une perspective intéressante en vue d'améliorer l'étude que nous venons de présenter.

Chapitre 5

Vérification de l'architecture et des spécifications détaillées

Sommaire

Au chapitre 3 nous avons présenté notre approche de vérification formelle de spécifications complexes et exposé la première des quatre étapes du processus. Le présent chapitre concerne les étapes 2 et 3, à savoir les spécifications de l'architecture du système et de ses composants.

La modélisation formelle de ces spécifications pose deux problèmes.

– Le nombre de composants du système étant élevé, la complexité du modèle formel obtenu rend difficile le test d'une nouvelle configuration du système ou de l'un de ses composants.
– Les spécifications du système sont basées sur plusieurs diagrammes semi-formels qu'il faut mettre en relation avec le modèle formel.

Pour résoudre ces problèmes notre approche repose sur :

1. la transformation des modèles des spécifications en modèles formels,
2. l'utilisation d'un patron modélisation,
3. l'élaboration d'une bibliothèque de modèles formels.

Cette approche permet de définir une architecture pour notre modèle formel, qui rendra possible de tester différentes implémentations d'un même composant en ne modifiant qu'une partie limitée du modèle.

La première section de ce chapitre présente les problèmes, objectifs et principes de notre méthode de modélisation. Dans la seconde section nous présentons les différentes étapes de notre méthode. Dans la troisième section nous l'appliquons à un cas concret : le système SAFESPOT et son application H&IW (*"Hazard & Incident Warning"*).

5.1 Introduction

La vérification des spécifications de l'architecture et des composants d'un système complexe pose différents problèmes.

Problèmes

1. Les systèmes complexes sont caractérisés par un nombre élevé de composants et d'interactions. Cela rend long et coûteux production des modèles formels.
2. Pour une vérification formelle des spécifications d'un système, il est important d'analyser sa structure ou son comportement dans différents cas d'utilisation. Or la complexité du modèle formel rend difficile sa modification.

Afin de produire le modèle formel des spécifications et d'interpréter les résultats d'analyse, il est nécessaire d'établir une relation entre les spécifications semi-formelles et le modèle formel.

Objectifs

Deux objectifs peuvent être déduits de ces problèmes :

1. faciliter la production du modèle formel et réduire son coût de production ;

2. rendre possible le test de différents cas d'utilisation (ou différentes versions d'un composant) du système, en minimisant les modifications du modèle formel.

Pour satisfaire ces objectifs notre méthode est basée sur les principes suivants.

Principes

1) *Utiliser la spécification de l'architecture pour composer le modèle formel.*

En fonction du cas d'utilisation, certains composants du systèmes peuvent ne pas être impliqués. L'environnement d'exécution du système peut aussi varier. Mais l'architecture du système elle, ne change pas. Cette définition de l'architecture peut être exploitée pour définir un *patron de modélisation formel*. Ce patron définit un ensemble de sous-modèles formels et leurs relations.

L'utilisation de ce patron nécessite la mise en place de règles d'assemblage des sous-modèles formels. Celles-ci peuvent être définies en fonction des interfaces spécifiées dans les diagrammes UML. Une approche modulaire de composition du modèle formel est ainsi possible. On peut ensuite définir une bibliothèque de sous-modèles formels où l'on choisira ceux que l'on souhaite utiliser dans la composition du modèle formel complet.

2) *Utiliser des règles de transformation pour produire les modèles formels.*

Ce principe permet de réduire le coût de production du modèle formel. Différentes approches ont été proposées pour la transformation de modèles UML vers des modèles de performance comme les Réseaux de File d'Attente (LQN) ou les Réseaux de Petri Stochastiques [35, 84, 8] comme nous l'avons présenté au chapitre 2, section 2.1. Ces transformations reposent sur une spécification des relations et des équivalences entre les méta-modèles sources et destination.

Néanmoins, ces approches n'utilisent pas les Réseaux de Petri Symétriques qui permettent de réduire les problèmes d'explosion combinatoire (voir chapitre 2, section 2.1). Nous avons donc décidé d'utiliser UML comme langage de spécification et les Réseaux de Petri Symétriques comme formalisme de modélisation formelle.

Les outils d'analyse

Les Réseaux de Petri Symétriques peuvent être vérifiés à l'aide de différents outils. Nous en citerons deux : PROD [123] et GreatSPN [28] qui permettent l'analyse comportementale du modèle en construisant le graphe des états accessibles.

Prod permet :

1. de vérifier des formules CTL ou LTL,
2. de détecter les situations de blocage du modèle (*deadlocks)*,
3. d'exhiber l'ensemble des événements (i.e. les tirages de transitions) liant deux états du système.

GreatSPN permet :

1. d'exploiter le symétries du modèle,
2. de produire le graphe d'accessibilité symbolique du modèle,
3. de tester des formules LTL.

La transformation de contraintes sur le système en formules LTL ou CTL est un sujet d'étude à part entière qui n'est pas traité dans ce mémoire. Nous montrons néanmoins un exemple de propriétés que nous avons vérifiées grâce à cette méthode chapitre 6.

Les situations de blocage du modèle (*deadlocks*) permettent d'exhiber, par exemple, des problèmes d'accès à des ressources partagées entre différents processus s'exécutant en parallèle. C'est en observant ce phénomène que nous avons pu optimiser l'application H&IW comme nous le montrons dans la deuxième section de ce chapitre.

Les deux outils PROD et GreatSPN sont intégrés à la plateforme CPN-AMI[59], avec d'autres outils comme un vérificateur de syntaxe ("*syntax checker*"), un outil d'analyse structurelle ou encore, l'outil PetriScript [61].

Ce dernier est particulièrement intéressant dans notre cas, car il permet de manipuler les Réseaux de Petri à l'aide de scripts. Il est ainsi possible de créer des places ou des transitions et de procéder à des fusions de places ou de transitions. C'est à l'aide de cet outil que nous appliquons nos règles d'assemblage au sein de notre patron du modèle formel.

La plateforme CPN-AMI est utilisable en invoquant ses services (i.e. les outils qu'elle intègre) à travers internet, ou bien de manière locale si elle est installée sur une machine. Les outils de modélisation Macao et Coloane possèdent une interface vers CPN-AMI. Ils répondent ainsi à nos besoins de modélisation et de vérification des réseaux de Petri Symétriques.

5.2 Les étapes de notre méthode

Notre processus de modélisation comporte quatre étapes (présentées figure 5.1) :

- **étape 1 :** définition de l'architecture du modèle formel que nous appelons *patron de modélisation* ;
- **étape 2 :** modélisation des différents *composants* du modèle formel rassemblés dans une *bibliothèque* ;
- **étape 3 :** sélection et *configuration* des composants, puis utilisation des *règles d'assemblage* pour obtenir le modèle complet en fonction du cas d'utilisation ou des propriétés à vérifier ;
- **étape 4 :** analyse du modèle à l'aide d'un ensemble d'*outils appropriés*.

5.2.1 Étape 1 : Définition du patron de modélisation

Le patron générique du modèle formel est produit par transformation de la spécification de l'architecture qui repose sur deux types de diagrammes UML :

- le diagramme de composants, qui spécifie l'architecture du système ;
- les diagrammes d'interfaces, qui décrivent les relations entre deux composants du système.

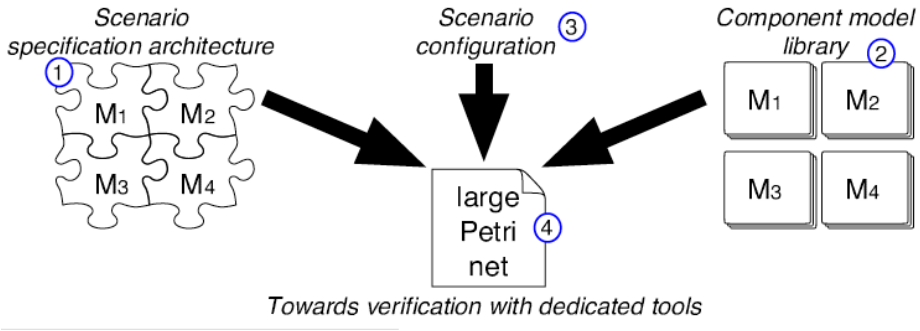


FIGURE 5.1 – Vue d'ensemble de l'approche modulaire

Nous utilisons les diagrammes d'interfaces et de composants tels que définis dans les spécifications UML 2.0 [88] [1]. La figure 5.2 montre un exemple de diagramme de composant, dans sa partie (a), et un diagramme d'interface dans sa partie (b).

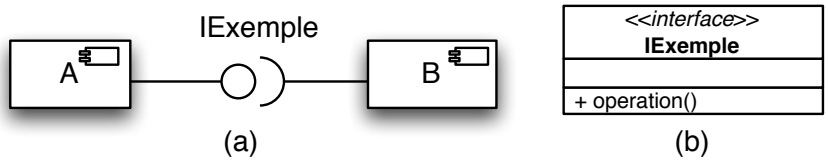


FIGURE 5.2 – Représentation graphique d'une interface en UML

Ces deux diagrammes UML décrivent la partie structurelle des relations entre les composants du système. Ils indiques :

- quels composants sont liés par une interface,
- qui fournit et qui requiert cette interface,
- quelles opérations sont impliquées dans ces relations,
- et quels paramètres entrent en jeu.

La transformation de ces relations en Réseaux de Petri Symétriques se fait suivant les principes que voici.

- Chaque interface UML est transformée en un ensemble de places, transitions et fonctions de couleur. Ces éléments serviront à connecter les sous-modèles formels entre eux à l'étape 3 de notre méthode.
- Chaque opération d'une interface est modélisée par un ensemble de places et de transitions.
- Les paramètres et la valeur de retour d'une opération sont modélisés par des fonctions de couleurs. Ils déterminent aussi les domaines de couleurs des places entrant en jeu dans l'interface.

Voici quelques conventions que nous utilisons :

- l'ensemble des opérations d'une l'interface, nommé *"ownedOperation"* en UML 2, est noté $O = \{o_1, ..., o_n\}$;
- l'ensemble des paramètres d'une opération, nommé *"ownedParameter"* en UML 2, est noté $Param = \{par_1, ..., par_j\}$;

1. Les éléments que nous considérons sont spécifiés de manière identique dans la version 2.2 des spécifications d'UML [89].

– la valeur de retour d'une opération, nommée *"\type"* en UML 2, est notée *ret*.

Nous utilisons des conventions pour nommer les places, transitions et fonctions de couleurs pour :

– faciliter la compréhension du modèle formel, et mettre en évidence sa relation avec les modèles UML,
– permettre l'automatisation du processus de transformation,
– faciliter la mise en place des règles d'assemblage des sous-modèles formels.

Les places et transitions qui correspondent à la transformation d'une interface sont nommées comme suit : on utilise d'abord la première lettre du module, puis la lettre *I* pour interface, suivie de la première lettre du nom de l'interface et des trois premières lettres du nom de l'opération. Par exemple, une opération nommée $opration$ au seins d'une interface nommée $Exemple$, impliquée dans la composition d'un composants *A*, donnera des places ou des transitions nommées A_IE_ope.

Plus formellement, nous définissons un ensemble de règles. Les règles 1.1 et 1.2 définissent comment transformer les flots de contrôle et de données des diagrammes UML. Les règles 2.1, 2.2 et 2.3 définissent comment transformer les interfaces. Les règles 3.1 et 3.2 définissent la transformation des diagrammes d'activités UML.

Règle 1.1
Transformation des flots de contrôle en domaines de couleur
Soit *A* et *B* deux composants UML liés par une ou plusieurs interfaces. Pour chacun des composants, on définit un domaine de couleur, *C_a* et *C_b* représentant leurs flots de contrôle, ainsi que des variables contenant une instance de ces couleurs. Avec la notation des Réseaux de Petri Symétriques on obtient : $Var < a > in\ C\_a$ et $Var < b > in\ C\_b$. On crée ensuite un domaine de couleur $C_a \times C_b$ que l'on nomme $cflux\_ab$ dont chaque instance $< a,b >$ permet de décrire une relation entre deux instances des composants *A* et *B*.

Règle 1.2
Transformation des flots de données en domaines de couleur
Soit *A* et *B* deux composants UML liés par une ou plusieurs interfaces. Les flots de contrôle liés à ces composants sont définis conformément à la règle 1.1. Si ces composants sont liés par l'intermédiaire d'une opération $opration()$ d'une interface, mettant en jeu des paramètres, on crée des domaines de couleur pour chacun des paramètres ou valeur de retour : $C\_ope\_par1...C\_ope\_parnetC\_ope\_ret$ On crée ensuite deux domaines de couleur $C\_a \times C\_b \times C\_ope_{par1}... \times C\_ope\_parn$ et $C\_a \times C\_b \times C\_ope\_ret$ que l'on nomme respectivement $dflux\_ab\_ope$ et $dflux\_ab\_ret$ dont chaque instance $< a,b,parm1...,param >$ et/ou $< a,b,ret >$ sert à décrire la relation entre deux instances des composants *A* et *B* au travers de l'opération de l'interface.

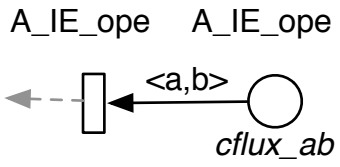
Règle 2.1

Transformation des opérations d'interface sans paramètre ni valeur de retour

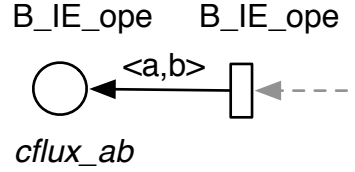
Soit $IExemple$ une interface constituée d'une opération $operation()$ sans paramètre ni valeur de retour.

1. Interface fournie L'opération de l'interface implémentée par le composant A est modélisée par une place et une transition. Elles sont nommées toutes les deux A_IE_ope sur la figure (a). Le domaine de la place est $C\_a \times C\_b$ et est noté $cflux\_ab$. La place est reliée à la transition par un arc allant de la place vers la transition. Cet arc est valué par les fonctions d'identité sur les variables représentant les flots de contrôles conformément à la règle 1.1 et notées $< a, b >$. Un arc sortant, représenté par une flèche grise en pointillés, indique le lien avec le reste du modèle du composant.

2. Interface requise L'opération de l'interface requise par le composant B est modélisée par une place et une transition. Elle sont nommées toutes les deux B_IE_ope sur la figure (b). Le domaine de la place est $C_a \times C_b$, et est noté $cflux\_ab$. La place est reliée à la transition par un arc allant de la transition vers la place. Cet arc est valué par les fonctions d'identité sur les variables représentant les flots de contrôle conformément à la règle 1.1, et notées $< a, b >$. Un arc entrant, représenté par une flèche grise en pointillé, indique le liens avec le reste du modèle du composant.



(a)



(b)

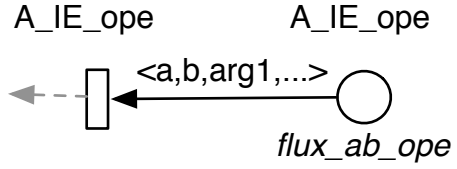
Règle 2.2

Transformation des opérations d'interface avec paramètres, sans valeur de retour

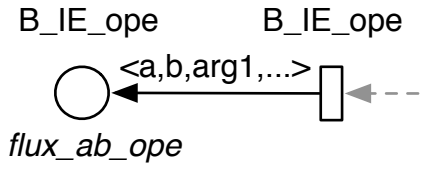
Soit *IExemple* une interface constituée d'une opération $operation()$ avec arguments, sans valeur de retour. Soit $Param = \{par_1, ..., par2\}$ l'ensemble des paramètres de l'opération. Pour chaque paramètre de l'opération on crée un domaine de couleur, nommé C_{argi}.

1. Interface fournie L'opération de l'interface implémentée par le composant *A* est modélisée par une place et une transition. Elle sont nommées toutes les deux *A_IE_ope* sur la figure (a). Le domaine de la place est $C_a \times C_b \times C_{arg1} \times C_{arg2}...$, et est noté $dflux\_ab\_ope$. La place est reliée à la transition par un arc allant de la place vers la transition. Cet arc est valué par les fonctions d'identité sur les variables représentant les flots de contrôle est de données et notées $< a, b, arg_1, arg_2, ... >$.

2. Interface requise L'opération de l'interface requise par le composant *B* est modélisée par une place et une transition. Elle sont nommées toutes les deux *B_IE_ope* sur la figure (b). Le domaine de la place est le même que pour le composant fournissant l'interface. La place est reliée à la transition par un arc allant de la transition vers la place. Cet arc est valué par les fonctions d'identité sur les variables représentant les flux de contrôles conformément à la règle 1.1, et notées $< a, b, arg_1, arg_2, ... >$.



(a)



(b)

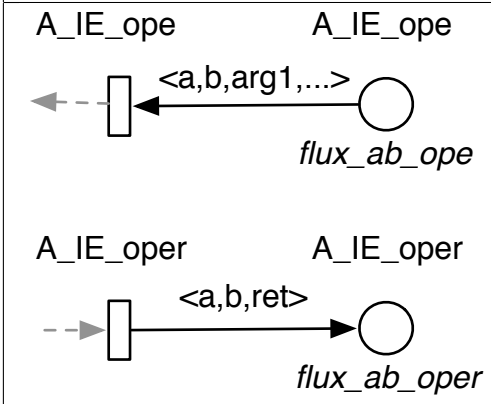
Règle 2.3

Transformation des opérations d'interface avec paramètres, et valeur de retour

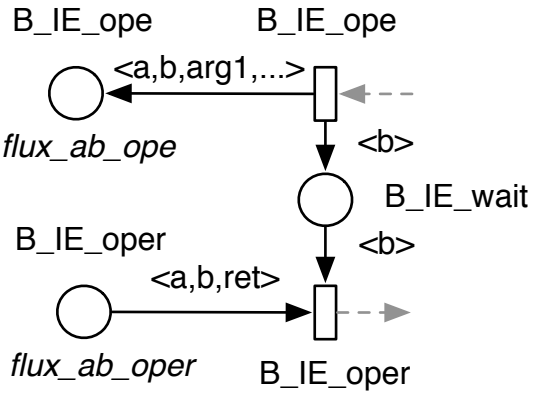
Soit *IExemple* une interface constituée d'une opération $operation()$ avec arguments, et valeur de retour. Soit $Param = \{par_1, ..., par2\}$ l'ensemble des paramètres de l'opération. Pour chaque paramètre de l'opération on crée un domaine de couleur nommé C_{argi}.

1. Interface fournie L'opération de l'interface implémentée par le composant *A* est modélisée par une place et une transition. Elle sont nommées toutes les deux *A_IE_ope* sur la figure (a). Le domaine de la place est $C_a \times C_b \times C_{arg1} \times C_{arg2}...$, et est noté $dflux\_ab\_ope$. La place est reliée à la transition par un arc allant de la place vers la transition. Cet arc est valué par les fonctions d'identité sur les variables représentant les flots de contrôle est de données, et notées $< a, b, arg_1, arg_2, ... >$.

2. Interface requise L'opération de l'interface requise par le composant *B* est modélisée par une place et une transition. Elle sont nommées toutes les deux *B_IE_ope* sur la figure (b). Le domaine de la place est le même que pour le composant fournissant l'interface. La place est reliée à la transition par un arc allant de la transition vers la place. Cet arc est valué par les fonctions d'identité sur les variables représentant les flots de contrôle conformément à la règle 1.1, et notées $< a, b, arg_1, arg_2, ... >$.



(a)



(b)

L'application des règles de transformation 1 et 2 permet de définir les parties des sous-modèles formels représentant les interfaces fournies ou requises des composants UML.

On définit ensuite les règles d'assemblage. Les places de l'interface (partie requise et fournie) sont fusionnées. Pour ce faire, on définit un ensemble d'instructions en langage Petri Script. Ces instructions sont ensuite utilisées avec l'outils FrameKit.

Prenons par exemple, le cas d'un client et d'un serveur liés par une interface nommée "QueryAPI", possédant une opération nommée "query" comme celle présentée figure 5.3. L'opération possède un paramètre "mess" et une valeur de retour "ret".

La transformation des flots de contrôle et de données de cette interface, conformément aux règles 1.1 et 1.2, produit des domaines de couleur $dflux\_cs\_que = C_c \times C_s \times C_{mess}$ et $dflux\_cs\_quer = C_c \times C_s \times C_{ret}$. Ces règles impliquent aussi la création d'instances de ces domaines de couleur : $< s, c, mess >$ et $< s, c, ret >$.

L'application de la règle de transformations 2.3 produit ensuite les places et transitions colorées en noir sur la figure 5.4. Sur cette figure, les interfaces sont reliées grâce aux règles d'assemblage. En l'occurrence, les places *Client_IQ_que* et *Serveur_IQ_que* ont été fusionnées dans une place nommée *IQ_que*, qui est remplie de gris sur la figure 5.4. La place *IQ_quer* est construite de la même manière, sur le flot de données transportant la valeur de retour de l'opération.

Les places et transitions en gris sur la figure 5.4 représentent les parties des composants connectées à ces interfaces. Il est possible de modifier cette partie du client (ou du serveur), sans modifier le serveur (ou le client) tant que la partie modélisant l'interface *Query* reste conforme aux spécifications.

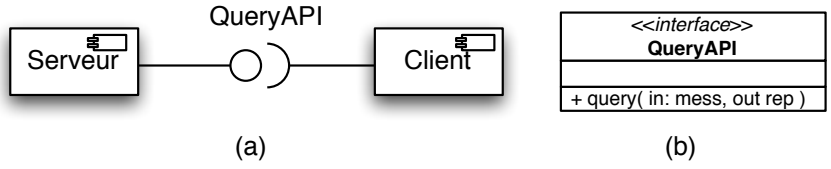


FIGURE 5.3 – Exemple d'une interface UML entre un client et un serveur.

5.2.2 Étape 2 : Création d'une bibliothèque de composants

La bibliothèque de composants peut être constituée de deux manière différentes.

1. En transformant les diagrammes d'activités UML en Réseau de Petri Symétriques.
2. En modélisant formellement les composants directement en Réseaux de Petri Symétriques.

Dans cette section, nous présentons les règles 3.1 et 3.2 de transformation utilisées pour transformer un diagramme d'activités en modèle de Réseau de Petri Symétrique.

La relation entre les activités UML et les réseaux de Petri Symétriques sont plus simples à définir que pour les interfaces UML. En effet, la sémantique des diagrammes d'activités UML est en partie inspirée des Réseaux de Petri. Une activité UML est activée si l'activité précédente sur le diagramme a produit une ressource. Elle va alors à son tour produire une ressource quand elle se terminera. La principale difficulté est donc d'identifier la nature de ces ressources (ce sont, dans la majorité, des flots de contrôle).

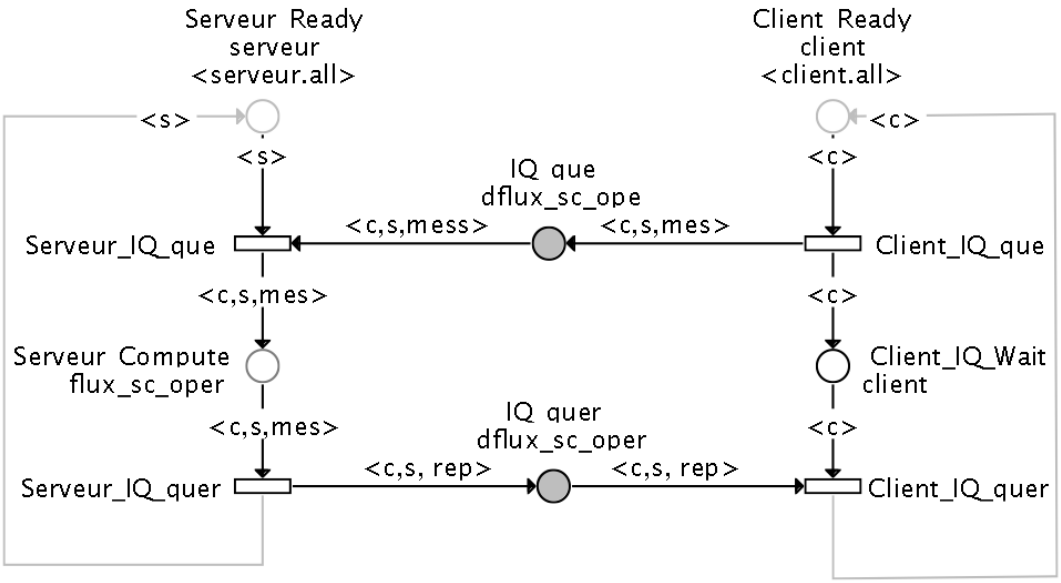


FIGURE 5.4 – Exemple de la transformation en Réseaux de Petri Symétriques du modèle de la figure 5.3.

Les règles de transformation que nous avons définies sont présentées dans les tableaux de règles 3.1 (page 100) et 3.2 (page 101).

5.2.3 Étape 3 : Assemblage du modèle formel

La première étape consiste à établir le ou les cas d'étude à utiliser comme base de l'analyse et à définir l'ensemble des propriétés à analyser. On choisit alors les versions de composants appropriés au cas d'étude.

Les communications entre composants correspondent généralement à un échange de message ou le récepteur fournit une pile de réception. Les messages sont traités par le récepteur dans un délai indéterminé après la réception et ne bloque pas l'évolution du module émetteur. C'est le cas dans SAFESPOT dans la majorité des échanges de messages à travers des datagrammes UDP. Ces communications sont modélisées en réseau de Petri par la fusion des places des interfaces définies dans les règles 2.1, 2.2 et 2.3. La figure 5.5 montre un exemple de fusion de places.

Les contraintes liées à cette fusion portent sur les domaines des places concernées, qui doivent être identiques. Cette contrainte est garantie si les règles 2.1, 2.2 et 2.3 sont appliquées. Le modèle formel complet est ensuite assemblé en utilisant l'outil PetriScript [61].

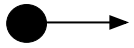
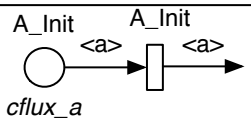
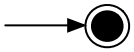
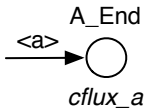
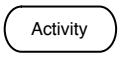
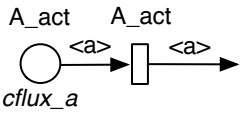
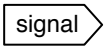
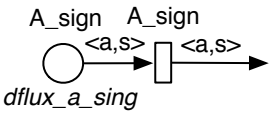
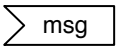
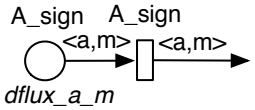
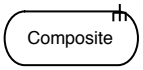
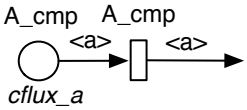
Il est parfois nécessaire de vérifier un composant avant de l'assembler au reste du modèle. Dans ce cas, un environnement du modèle formel d'un composant est représenté par des places qui contiennent un ensemble de variables d'entrées et qui servent aux tests préalables. Un exemple est fourni dans la seconde section de ce chapitre.

5.2.4 Étape 4 : Analyse et vérification

L'analyse peut être effectuée sur deux niveaux :

Règle 3.1

Transformation d'éléments d'activité en Réseaux Symétriques

Élément d'activité	Représentation UML	Représentation Réseau Symétrique
noeud initial		
noeud final		
acivité		
envoi de signal		
reception de message		
acivité composite		

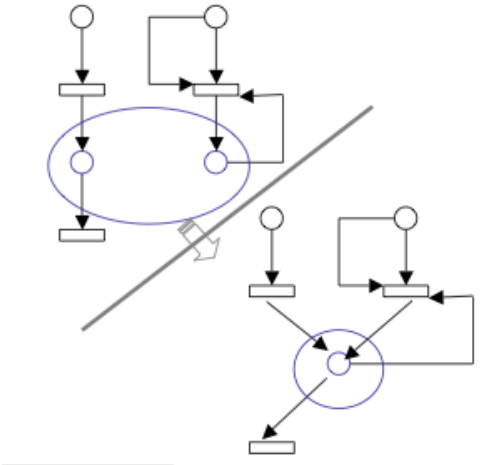
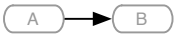
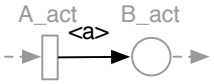
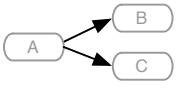
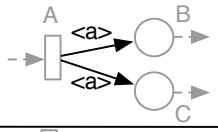
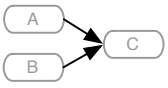
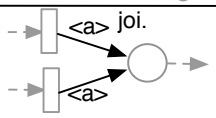
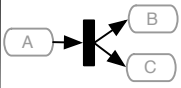
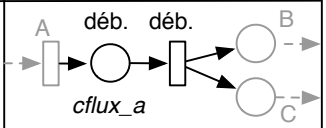
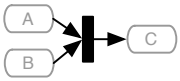
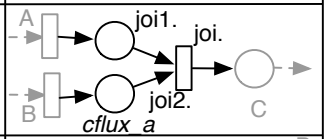
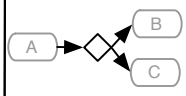
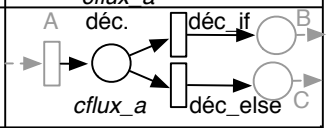
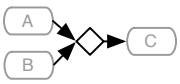
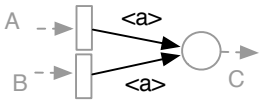


FIGURE 5.5 – Fusion de places

Règle 3.2

Transformation de motifs UML en réseaux symétrique

Motif d'activité	Représentation UML	Représentation Réseau Symétrique
flot de contrôle		
flot de contrôle (2 flots sortants)		
flot de contrôle (2 flots entrants)		
débranchement		
jointure		
décision		
fusion		

1. sur chaque composant pris individuellement,
2. ou sur le modèle assemblé.

Nos tests montrent qu'il est préférable de tester au préalable les comportements attendus des composants avant de les assembler. Cela permet d'effectuer un *"débuggage"*.

Comme nous l'avons présenté dans la première section de ce chapitre, les outils que nous utilisons permettent d'effectuer différentes vérifications sur le modèle. ll est ainsi possible de vérifier par exemple si le modèle est *vivant*, ou d'analyser les séquences d'évènements menant à un blocage du modèle (*deadlock*).

Les cas d'utilisation comme ceux que nous avons présentés chapitre 4 sont liés à un ensemble de contraintes qui peuvent être interprétées comme des propriétés à analyser. Aussi, la deuxième partie de ces cas d'utilisation permet de définir un ensemble de relations de cause à effet que l'on souhaite vérifier ou au contraire éviter.

Dans ces deux cas, l'écriture des propriétés dans un langage de logique temporelle comme LTL ou CTL est nécessaire. C'est un sujet d'étude à part entière qui n'est pas traité dans ce mémoire. Nous montrons néanmoins, au chapitre 6, un exemple de contraintes sur le système transformées en formule CTL puis vérifiées.

5.3 Application : vérification de l'architecture SAFESPOT et de l'application H&IW

Différents cas d'étude issus du projet SAFESPOT ont été étudiés. Le premier consiste à valider les aspects dynamiques de l'architecture du système présentée section 3.2, et d'en vérifier la conformité avec les cas d'utilisation et les contraintes de fonctionnement présentés section 3.2. Le deuxième cas d'étude consiste à valider le comportement d'un composant important du système SAFESPOT : l'application H&IW présentée section 2.5.3.

5.3.1 Analyse de l'architecture

L'observation de différents projets de STI permet d'identifier un ensemble de composants de base utilisés dans un grand nombre de projets de STI comme SAFESPOT, TrafficView ou encore les systèmes présentés chapitre 3.

Il s'agit de pouvoir réutiliser un ensemble de modèles formels pour analyser différents projets de STI. Notre approche modulaire permet alors de changer certains composants en fonction du projet que l'on souhaite analyser. Cette architecture, applicable à plusieurs projets est définie dans un **patron générique pour les STI**.

Ce patron, présenté figure 5.6, est composé de :

1. le *STI (véhicule / infrastructure)*
2. l'*environnement du système* (i.e. *System environnement* sur la figure 5.6). Dans notre cas, cet environnement inclut le canal de communication et la modélisation des conducteurs.

La vue détaillée de l'architecture d'une entité STI (véhicule ou infrastructure) est présentée figure . 5.7. Les modules impliqués sont :

– Les capteurs (*sensors*), qui mesurent l'état d'un véhicules et de l'environnement.

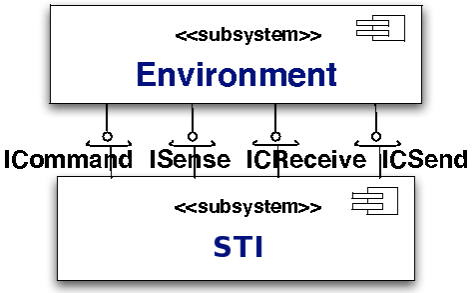


FIGURE 5.6 – Présentation des deux composants principaux du patron générique pour STI

– La base de données (*Perceived State*) qui stocke les informations nécessaires au calcul d'une commande.
– La stratégie (*Strategy*) chargée de calculer une commande en fonction de l'état perçu et de la transmettre à l'environnement.
– Le composant de communication qui conditionne l'échange de messages entre les véhicules et/ou l'infrastructure.

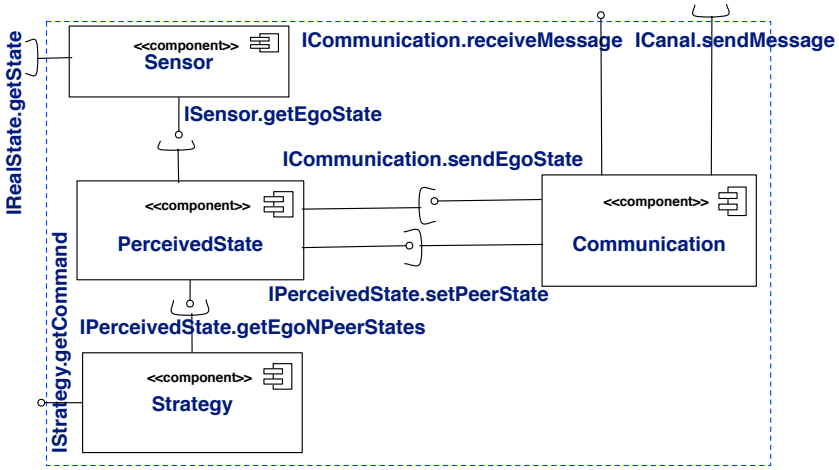


FIGURE 5.7 – Composants d'une entité STI (véhicule ou infrastructure)

L'architecture du système SAFESPOT (voir section 3.2.2) peut être modélisée à l'aide du canevas générique pour STI que nous venons de présenter. La figure 5.8 présente l'architecture de SAFESPOT. Initialement modélisée en UML1 dans le projet SAFESPOT, nous présentons ici sont équivalent en UML2 pour être plus conforme à notre méthode.

Les hypothèses sont alors les suivantes.

– Les composants Gateway, Sensors et Data Fusion de l'architecture SAFESPOT sont considérés comme des composants Sensor du patron générique.
– Le composant APPCoordinator qui gère les affichages pour le conducteur est intégré au composant Strategy avec le composant Applications.
– Le composant LDM correspond au composant PerceivedState du patron générique.
– Le composant Message Manager correspond au composant Communication du patron générique.

– Enfin, la relation entre les composants Applications et DataFusion doit être implémentée dans l'environnement du patron (alors qu'elle est théoriquement interne au système).

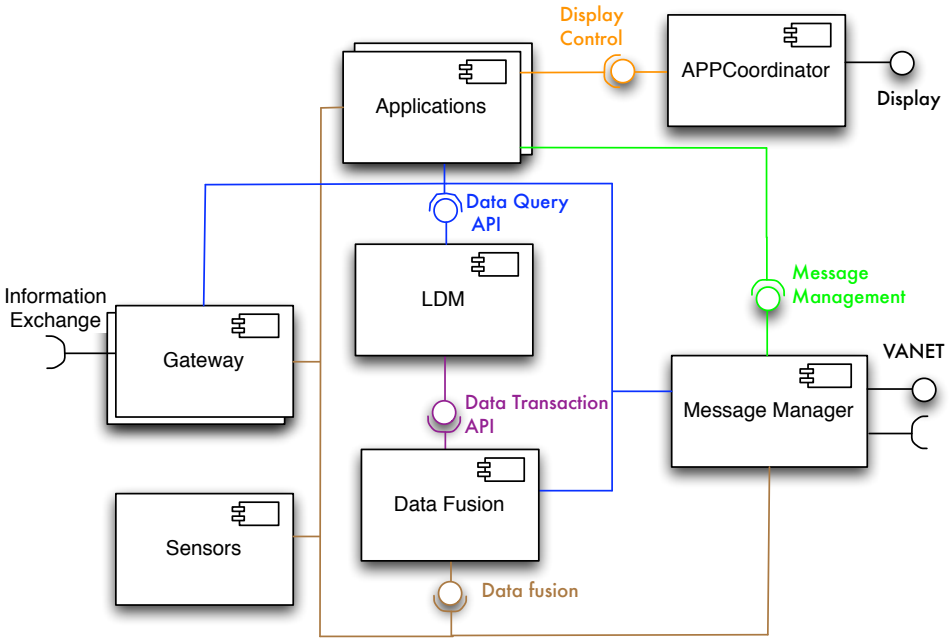


FIGURE 5.8 – Diagramme de composants UML de SAFESPOT(UML2.0)

Si les diagrammes UML de composants, d'interfaces et d'activités sont fournis, il est possible d'obtenir le modèle formel en réseau de Petri d'un composant ou de l'ensemble du système. Ceci est facilité par l'application des règles de transformation que nous avons présentées dans la deuxième section de ce chapitre.

Le modèle complet du système est assemblé au niveau des interfaces grâce à l'utilisation de l'outil PetriScript.

En fonction de la propriété du système à analyser et des abstractions utilisées, certains modules peuvent être ignorés ou simplifiés.

Modélisation de l'environnement

Le composant environnement, présenté dans cette section, est constitué de plusieurs éléments. Pour faciliter la lecture de ce mémoire, les modèles en réseaux de Petri Symétriques sont présentés en annexe A.5 page 169.

Les éléments constitutifs de l'environnement sont les suivants.

1. Une description de l'état réel des véhicules. Cet état sera perçu par les véhicules avec un décalage temporel. Cela permet de modéliser le fait que l'état perçu par le véhicule est différent de l'état réel en raison du retard introduit par les capteurs et les communications.
2. Une description du conducteur. C'est lui qui, en fonction des commandes émises par le système, change l'état réel d'un véhicule.

3. Un canal de communication. Il relie les différentes entités STI entre elles.
4. Une horloge globale. Elle est conçue afin de garantir une exécution équitable des entités STI et de l'environnement au cours du temps. Cette horloge est donc une abstraction du temps symbolisé par un cycle. Elle est connectée à l'environnement et aux entités STI afin de conditionner l'ordre d'exécution des actions : utilisation des capteurs, envoi de messages, réception de messages, envoi de commandes, mise à jour de l'environnement réel. La figure 5.9 présente l'ordonnanceur de cette horloge. Cette horloge n'est pas connectée aux autres composants au travers d'interfaces du fait de sa nature que nous venons d'exposer. Les transitions *VehicleMain_IVSAuthorizeSensor*, *VehicleMain_IVSAuthorizeSend*, *VehicleMain_IVSAuthorizeReceive*, *VehicleMain_IVSAuthorizeCommand* et *VehicleMain_EndCycle* sont alors fusionnées dans les différents composants du modèle.

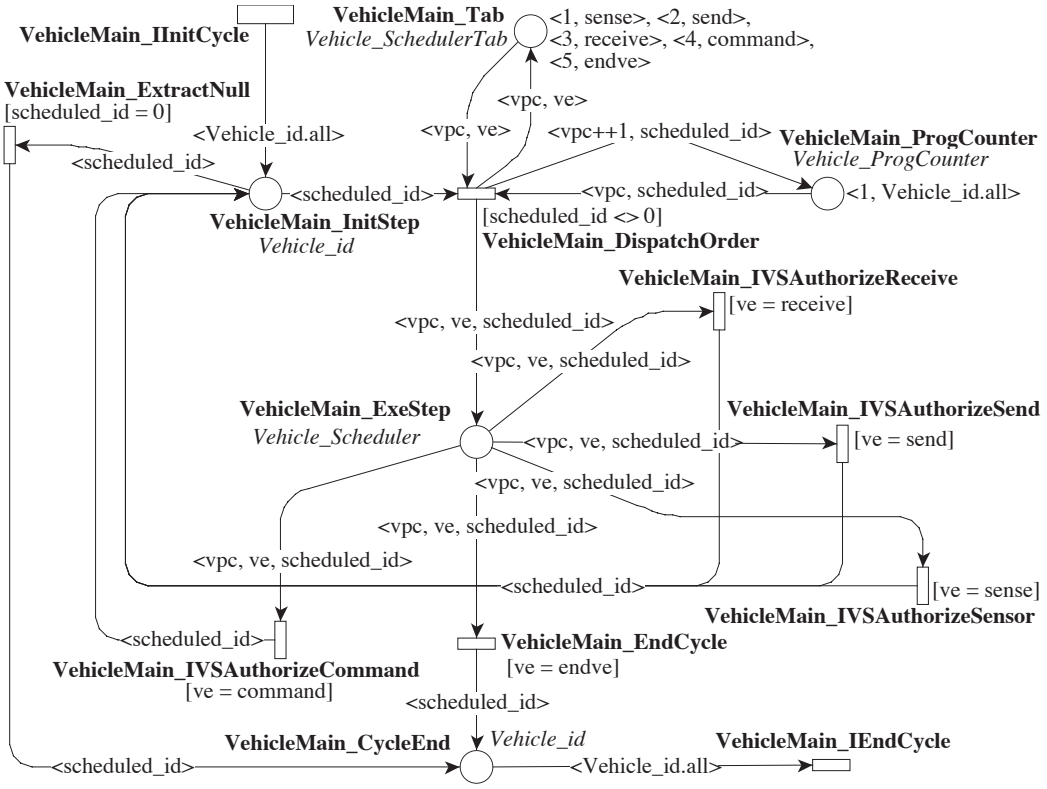


FIGURE 5.9 – Ordonnanceur de l'horloge du modèle

Cas d'étude

Nous étudions, dans ce premier cas d'étude, le déclenchement par une entité STI d'alertes en rapport à certains types de conflits : inter-distances trop faibles, manoeuvres de dépassement inappropriées ou changements de file dangereux.

Le système SAFESPOT est conçu pour permettre à un groupe de véhicules de coopérer. Ces véhicules échangent des informations relatives à l'état du trafic et en informent les conducteurs de telle sorte qu'ils puissent prendre les décisions appropriées : éviter un embouteillage, changer de file dans une zone d'accident, ralentir, etc.

Pour rester dans un univers discret permettant de procéder à des analyses formelles, il faut procéder à des abstractions des variables continues comme l'espace et de temps.

Pour ce faire, la route est considérée comme une ressource partagée et divisée en cellules de tailles variables et contiguës, se déplaçant à la vitesse moyenne du trafic (voir figure 5.10). Chaque véhicule occupe une cellule entière, mais certaines cellules peuvent ne pas être occupées et sont considérées comme libres si elles offrent un espace suffisant pour qu'un véhicule s'y insère.

Un véhicule désirant se déplacer dans une cellule libre prévient les véhicules adjacents à cette cellule. Ces véhicules étant des candidats potentiels à cette cellule, un consensus doit être obtenu pour permettre à un véhicule d'arriver dans cette cellule. Les véhicules partagent le médium de communication et la portée des communications est limitée en nombre de cellules.

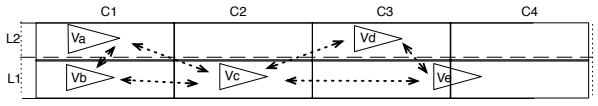


FIGURE 5.10 – Abstraction de la route

Sur la figure 5.10 le véhicule *Va* est dans la cellule *L2C1*. Il communique avec les véhicules *Vb* et *Vc* respectivement dans les cellules *L1C1* et *L1C2*. Le véhicule *Vd* est en dehors de son champ de communication. Pour pouvoir se déplacer dans la cellule *L2C2* il va donc devoir obtenir un consensus avec les véhicules *Vb* et *Vc*.

Ce type de modèle peut être utilisé pour tester et implémenter différentes stratégies de contrôle [37]. Un certain nombre d'hypothèses sont nécessaires afin de spécifier ce modèle :

- Les véhicules communiquent grâce à des communications sans fils dont la capacité peut être limitée par le nombre de connections ouvertes simultanément.
- Les communications sont basées sur l'échange de messages : les informations à échanger sont agrégées dans des messages transmis aux autres véhicules.
- Les informations échangées concernent l'état des véhicules. Elles sont représentées à l'aide d'une abstraction mais peuvent être raffinées par la suite pour des études plus précises (c'est ce que nous présentons au chapitre suivant).
- Chaque véhicule perçoit son état à travers ses capteurs et sa balise GPS. Ces informations sont stockées dans sa base de données locale.
- Chaque véhicule reçoit l'état de ses voisins au travers des communications sans fils. Ces informations sont aussi stockées dans les bases de données des véhicules.
- La représentation de l'environnement perçue par les véhicules présente un certain décalage par rapport à la situation réelle : les informations stockées sont mises à jour de telle sorte qu'à un instant i la base de données d'un véhicule et la situation réelle ne sont pas totalement identiques, du fait des délais de mise à jour.

Analyses

Pour générer le modèle complet, les diagrammes UML d'architecture, d'interfaces et d'activités sont suffisants. Après l'application des règles de transformations que nous avons définies, il est possible de vérifier les spécifications de l'architecture d'un système.

Ce cas d'étude nous permet de valider :
- notre patron générique pour STI,
- la modélisation de l'architecture de SAFESPOT grâce à ce patron,
- la modularité de notre approche : la modification des composants au sein du modèle formel ne remet pas en cause le reste du modèle,
- que le modèle produit est syntaxiquement correct et analysable.

Dans ce cas d'étude nous avons testé différentes versions du composant *stratégie* des véhicules. Pour ce faire, nous avons vérifié :
- l'absence d'inter-bloquage,
- l'absence de place contenant un nombre infini de ressources,
- des propriétés de logique temporelle.

Ces dernières sont soit écrites à partir des "contraintes d'exécutions " présentées 4 soit rédigées directement en logique temporelle. Cela permet de prouver par exemple que le module stratégie ne donnera pas certaines recommandations si l'ensemble des états des véhicules adjacents n'a pas été reçu.

Dans la cas d'étude suivant, nous présentons la validation de notre approche de transformation des diagrammes d'activités.

5.3.2 Analyse de l'application H&IW

Nous présentons dans cette section l'analyse de l'application "Hazard and Incident Warning" à partir de ses diagrammes d'activités. Cette étude consiste à isoler ce composant de l'architecture du système. Ce cas d'étude concerne la validation de sa stratégie de gestion de threads.

La première partie présente les spécifications de l'application. Puis nous présentons la modélisation formelle est les résultats d'analyse du modèle.

Spécification de l'application

L'application "Hazard and Incident Warning" (H&IW) est constituée d'un ensemble de classes effectuant de manière générique différentes tâches. Voici leur présentation.

Le contrôleur de zone ("**Black Spot Monitor**")

Il est chargé de détecter les sources de dangers considérées par l'application et, le cas échéant, d'activer leur prise en charge par un "**scénario Manager**".

Sa définition est générique et applicable à tous les cas d'utilisation de H&IW que nous avons présentés au chapitre 4. La partie générique devant être implémentée différemment

pour chaque cas d'utilisation est la partie définissant les conditions de détection d'un danger. Elle est implémentée par la fonction "***eventCaching***".

Le gestionnaire de scénario ("**Scenario Manager**")

Il prend en charge la stratégie d'alerte des usagers relative au danger détecté. Il permet d'analyser la situation et d'alerter les véhicules au travers de la fonction "***scenarioAnalysis()***". Il est composé de sous-modules :

- l'évaluateur de danger ("**Threat assessor**") qui évalue le niveau de risque par la fonction "***riskEvaluation()***".
- le gestionnaire de décision ("**Decision manager**") qui calcule la stratégie d'alerte au travers de la fonction "***safetyMarginComputation()***".
- l'activateur d'alerte ("**Warning system actuator**") qui est responsable du déclenchement et de l'arrêt des alertes sur différents types de supports : réseau, panneau à messages variables, écran embarqué dans le véhicule, etc. Il le fait au travers de la fonction "***warningStrategyRealisation()***".

Spécification du comportement de l'application

Les diagrammes d'activités du *BlackSpotMonitor* et du *ScénarioManager* et de l'application H&IW sont présentés figure 5.11. Ces diagrammes d'activités présentent les différents flots de contrôle impliqués lors de l'exécution de l'application.

Une exécution type se déroule ainsi :

1. Quand l'application H&IW est exécutée, elle démarre un "*BlackSpotMonitor*" pour chaque type de danger pris en charge par l'application : chantier, plaque de verglas, accident, etc. Cette action est modélisée par le nœud initial du diagramme d'activités du *BlackSpotMonitor*.
2. Le "*BlackSpotMonitor*" s'enregistre auprès de la base de données pour un type de danger, puis se met en attente. Ceci est modélisé par les activités *bsm.start()* et *eventCatching()*.
3. la base de données du système (la *LDM*) se met en attente de l'occurrence de l'événement. Lorsque celui-ci survient, elle envoie un message au *BlackSpotMonitor*.
4. celui-ci, analyse le danger, puis vérifie si un "*ScenarioManager*" déjà actif prend en charge un danger dont la zone d'impact chevauche celle du nouveau danger. Ceci est modélisé par l'activité *findRelevantSc()*.
5. sinon, il démarre un nouveau "*ScenarioManager*", ce qui est modélisé par les activités *newSc()*, *addScenario()* puis *smg.start()*.

La boucle principale du "*ScenarioManager*" se décompose en cinq activités :

1. *scenarioAnalysis()* : qui représente l'analyse des événements fournis par le "*BlackSpotMonitor*".
2. *riskEvÉvaluation()* : qui calcule si un risque existe ou non.
3. *safetyMarginComputation()* : qui calcule les alertes à déclencher.
4. *warningStrategyRealisation()* : qui envoie les alertes au conducteur, puis recommence la séquence d'activités.

5. si les dangers ne sont plus présents, le "*ScenarioManager*" s'arrête, ce qui est représenté par l'activité *stop()*.

L'application des règles de transformation définies section 5.1 permet de produire les modèles en Réseaux de Petri Symétriques présentés figures 5.12 et 5.12.

Analyse

Ce deuxième cas d'étude présente les résultats obtenus sur l'analyse de l'application *"Hazard and Incident Warning"*. Il s'agit ici de l'analyse d'un module isolé.

L'application des règles de transformation présentées section 5.1 permet une première modélisation des diagrammes d'activités qui peuvent déjà être analysés avec les outils que nous avons présentés section 5.1. Mais la portée de ces vérifications reste limitée.

Pour effectuer l'analyse de la gestion des Threads par exemple, il est nécessaire de raffiner ces modèles et d'y ajouter des éléments supplémentaires, ainsi qu'un environnement d'exécution.

Voici les principales actions effectuées pour produire le modèle permettant cette analyse.

1. La base de données est représentée par une place LDM_API envoyant des événements au *BlackSpotMonitor*.
2. La donnée modélisant la liste des ScénarioManager actifs est modélisée par une place Bsm_SmList.
3. La liste des *ScénarioManager* inactifs est modélisée par la place Sys_FreeThreads.
4. L'activité *Sm.addEvent()* est modélisée par la place et la transition SM_addEvent.
5. La liste des événements gérés par un scénario manager est modélisé par la place Smg_EventList.
6. Enfin, les deux modèles sont connectés. L'activité *smg.start()* du BlackSpot monitor génère une ressource dans le nœud initial du diagramme d'activité du ScénarioManager. Ce qui est modélisé en reliant la transition *Smg__startt* du modèle du *BlackSpotMonitor* avec la place Smg_start du *ScénarioManager*.

Nous obtenons alors le modèle présenté figure 5.13.

L'analyse de ce réseau montre que le *"Scenario Manager"* peut se terminer après le test *isRisk()* du *"BlackspotMonitor"* alors qu'un nouvel événement (un danger type travaux ou accident) à été ajouté à la liste des dangers à traiter.

Ceci est mis en évidence lors de l'analyse formelle de l'existence de *deadlocks*. On constate alors que le modèle peut s'arrêter avec un jeton $< sm, ev >$ dans la place Smg_EventList, alors que le jeton $< sm >$ modélisant le flot de contrôle du ScénarioManager correspondant est dans la place Smg_end (il a terminé).

En d'autres termes, certains dangers peuvent être ignorés par l'application. De plus, s'il n'y a plus de danger à traiter il est important de pouvoir le signaler en passant par l'activité *"WarningStrategyRealisation"*.

Pour résoudre ce problème, une solution simple consiste à ajouter un verrou conditionnant l'accès à la liste des dangers pris en charge par un *"Scenario Manager"*. Ainsi entre le moment

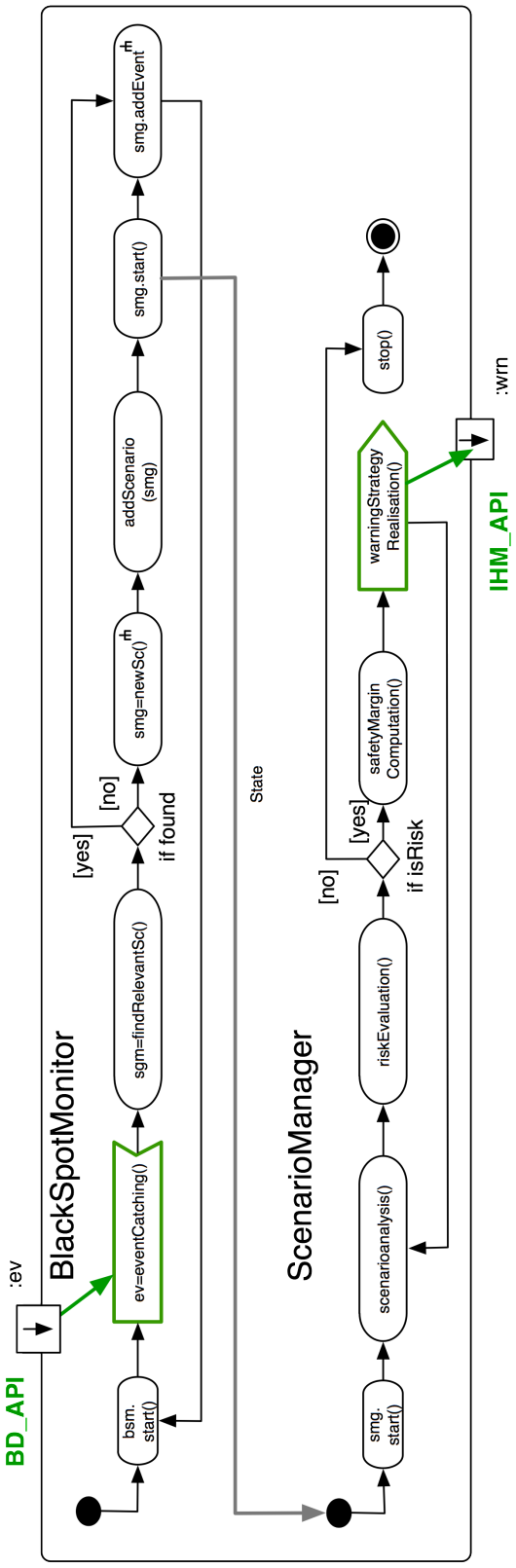


FIGURE 5.11 – Présentation des diagrammes d'activités de l'application H&IW

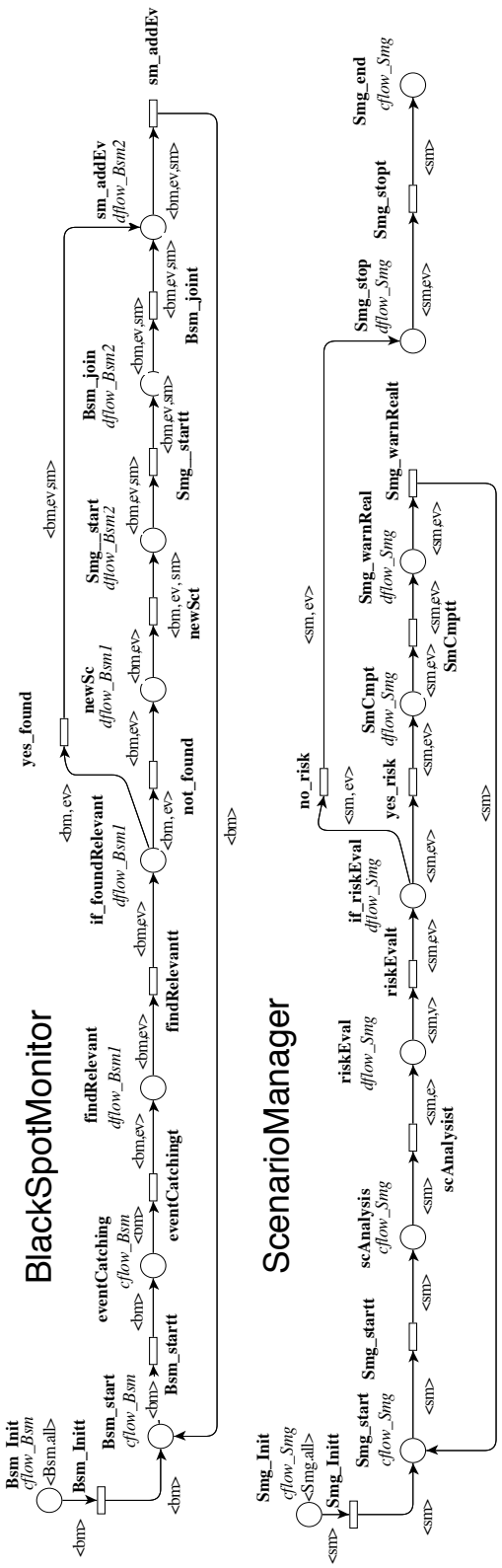


FIGURE 5.12 – Activités de H&IW en Réseau de Petri Symétrique

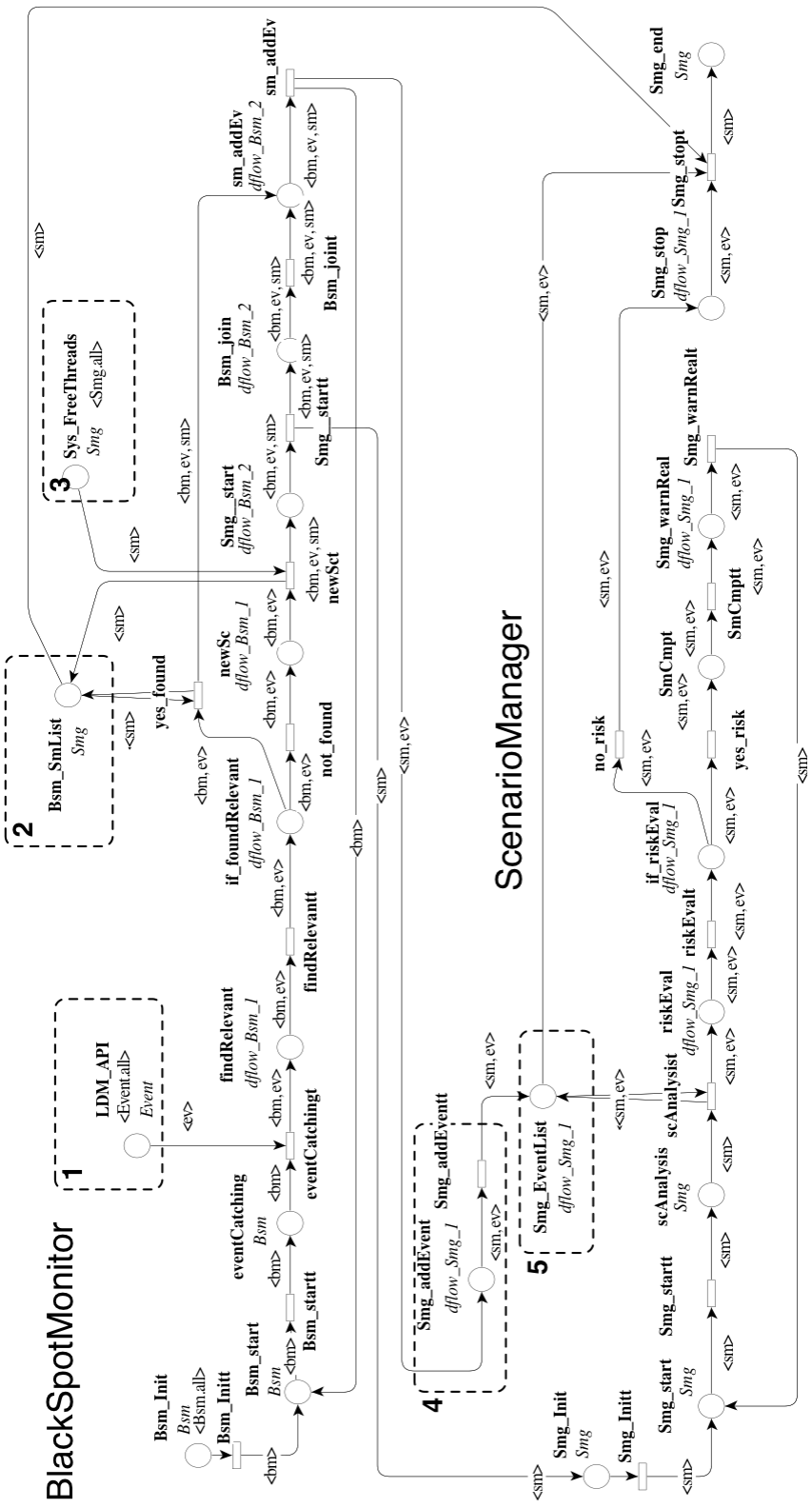


FIGURE 5.13 – Présentation du réseau symétrique de H&IW assemblé avec un environnement minimal de test

ou le *"BlackspotMonitor"* test les dangers pris en charge par un *"Scenario Manager"* et le moment ou il ajoute un danger à sa liste, il est impossible pour le *"Scenario Manager"* de faire le test isRisk() et de se terminer.

Enfin, pour optimiser l'algorithme du *"Scenario Manager"* il est possible de tester la liste des dangers encore actifs avant de commencer un nouveau cycle d'analyse et d'alerte. Ce test est moins puissant que le premier car il n'analyse que la présence des dangers et ne mesure pas leur dangerosité comme le fait le premier test.

Une autre optimisation a été obtenue sur l'algorithme de l'application H&IW (et déjà inclue dans les modèles que nous venons de présenter). En analysant le diagramme d'activités présentant ses interactions avec la base de données, on constate qu'une approche événementiel de la gestion de la boucle principale du *"Scenario Manager"* est possible et permet de supprimer des accès inutiles à la base de données (améliorant donc les performances de l'application). En définissant les conditions de déclenchement des alertes (la présence d'utilisateurs en situation de danger) et en utilisant le mécanisme d'enregistrement et de notification de la base de données, il n'est plus nécessaire de répéter continuellement la boucle principale du *"Scenario Manager"* effectuant ces requêtes à la base de données.

Ce cas d'étude permet de valider :

1. que les règles de transformation permettent de générer rapidement des modèles en Réseaux de Petri Symétriques,
2. que les modèles produits sont syntaxiquement corrects et analysables,
3. qu'il peuvent être facilement raffinés pour effectuer des tests supplémentaires.

5.4 Conclusion

Dans ce chapitre nous avons présenté notre méthode de modélisation formelle des spécifications de l'architecture et des composants d'un système complexe. Notre objectif est d'assister le concepteur de systèmes complexes et de faciliter la production des modèles formels. Nous définissons à cet effet un ensemble de règles de transformation des diagrammes UML vers les Réseaux de Petri Symétriques.

La transformation des diagrammes de composants et d'interfaces permet la mise en place d'un patron générique pour notre modèle formel. La définition de ce patron apporte plusieurs avantages.

1. Il permet une approche modulaire de la composition du modèle formel. Cela permet de tester différents modèles, et différents scénarios d'analyse en ne modifiant qu'une sous-partie du modèle formel.
2. Il permet aussi de définir un ensemble de composants dans une bibliothèque qui peuvent, par la suite, être réutilisés dans différents scénarios d'analyse.

La définition de règles de transformation des diagrammes d'activités vers les Réseaux de Petri Symétriques facilite la modélisation des composants du système.

1. Une fois produits, ces modèles peuvent être analysés ou intégrés au patron générique.
2. Ils peuvent aussi facilement être adaptés pour des analyses plus fines sur le composant.

Un point intéressant de notre méthode est que certaines transformations de modèles peuvent être automatisées, ce qui facilite la production des modèles formels.

Nous avons obtenu des résultats d'analyses sur deux cas d'étude issus du projet SAFESPOT. Les propriétés qualitatives analysées ont permis des optimisations du système et de ses composants.

Chapitre 6

Vérification formelle des contraintes continues par discrétisation

Sommaire

Au chapitre 5 nous avons montré comment modéliser et vérifier formellement l'architecture et les composants du système à partir de diagrammes UML. Dans ce chapitre 6, nous nous intéressons aux algorithmes du système à un niveau de détail supérieur. La spécification initiale de ces algorithmes est basée sur des formules mathématiques impliquant des variables et fonctions continues, comme nous l'avons expliqué chapitre 3 section 3.2.

La plupart des systèmes de transport intelligents impliquent des paramètres physiques continus dans leur comportements, comme l'espace ou le temps. On parle alors de systèmes hybrides. La vérification formelle de tels systèmes dans un contexte industriel pose différents problèmes.

1. D'abord un problème de disponibilité d'outils permettant l'analyse formelle des modèles.
2. Ensuite, celui de la disponibilité d'une méthode de modélisation pouvant être appliquée à différents systèmes.
3. Enfin, un problème de calculabilité. La modélisation des systèmes hybrides devant répondre à de nombreuses contraintes pour rendre le modèle décidable.

Dans ce chapitre nous présentons une méthodes **d'intégration des phénomènes continus** dans la modélisation et la vérification des systèmes complexes. Notre méthode est basée sur l'utilisation des Réseaux de Petri Colorés pour décrire les comportements dépendant de paramètres continus. Ces modèles sont ensuite transformés en Réseaux de Petri Symétriques à l'aide d'une **méthode de discrétisation**.

Nous étudions enfin les **incertitudes** introduites par la discrétisation et leurs conséquences sur le modèle et ses propriétés formelles. Nous obtenons ainsi des modèles formels discrets et potentiellement finis permettant de vérifier des propriétés exprimées en **logique temporelle comme LTL ou CTL**.

La première partie de ce chapitre présente notre méthodologie de modélisation des phénomènes continus. Puis nous appliquons cette méthode à un cas concret : le module de freinage d'urgence de l'application "Hazard and Incident Warning" issue du projet SAFESPOT. Ce cas d'étude est présenté dans trois sections. D'abord nous présentons les spécifications de l'application et ses contraintes de fonctionnement. Puis, nous montrons les modèles produits et les conséquences de la discrétisation sur le modèle et ses contraintes de fonctionnement. Enfin, nous présentons les résultats d'analyse dans une troisième partie. En conclusion de ce chapitre nous étudions les conséquences de notre approche et les perspectives qu'elle apporte pour la modélisation des systèmes complexes et hybrides.

6.1 Méthodologie

Cette section présente notre méthodologie de modélisation et d'analyse de systèmes hybrides. Dans une première partie nous présentons notre méthodé. Puis nous présentons ses différentes étapes : la modélisation, la discrétisation et l'analyse.

6.1.1 Présentation de la méthodologie

Les systèmes de transports intelligents sont des systèmes complexes et hybrides. Plusieurs décisions sont calculées en parallèles par différentes entités comme les véhicules ou l'infrastructure. Une partie de ces calculs est basée sur des variables et des comportement continus comme l'adhérence de la chaussée ou la position des véhicules. L'entrelacement des comportements discrets et continus, propre aux systèmes hybrides, pose des problèmes pour l'application des méthodes formelles. Si ces paramètres continus ne sont pas pris en compte et modélisés, seule une sous-partie des contraintes du système peut être vérifiée.

Des techniques sont dédiées à l'analyse des systèmes continus comme les méthodes algébriques comme la méthode B [1]. Mais ces approches sont difficiles à mettre en place et à paramétrer. Leur utilisation dans un contexte industriel a donc une conséquence en termes de coût et se combine difficilement avec l'approche élaborée dans ce mémoire qui considère différents niveaux d'abstraction.

Les méthodes de vérification de modèles (en anglais *model checking*) sont plus simples à utiliser mais prennent difficilement en charge les aspects continus. La majorité de ces techniques de vérification concerne les modèles discrets. La prise en charge des phénomènes continus dans ce contexte introduit de nombreuses contraintes pour garantir que le modèle soit fini et analysable.

Introduit par Alur et al. et Maler et al. au début des années 90 les automates hybrides [2] permettent de modéliser et de spécifier formellement des systèmes hybrides. Les états des automates hybrides sont constitués d'un état discret et d'un vecteur de variables continues qui leur donne la puissance du continu (i.e. le cardinal du continu). Il est prouvé depuis le milieu des années 90 que dans ce cas l'ensemble des états atteignables n'est pas décidable dans le cas général et son calcul par approximation n'est pas trivial [62].

Les réseaux de Petri Hybrid [38] offrent une approche intéressante, mais il n'existe pas à l'heure actuelle d'outils permettant d'analyser les formules de logique temporelle [92].

Nous présentons ici une méthodologie de gestion des aspects hybrides d'un système par vérification de modèles avec les Réseaux de Petri et des méthodes algébriques. Notre méthode est basée sur la transformation de Réseaux de Petri Colorés (CPN) en Réseaux de Pétri Symétriques (SN) (que nous avons présentés au chapitre 2 section 2.1). Une approche similaire de transformation a été élaborée [11] à partir de CPN vers des systèmes à compteurs. L'objectif y est de procéder à l'analyse de spécifications exprimées en CPN, mais sans la composante hybride que nous introduisons ici.

Notre méthode utilise :

- les CPN pour leur capacité à modéliser les aspects continus d'un système,
- les SN pour leur capacité à produire l'espace d'état symbolique d'un modèle, ce qui facilite l'analyse des systèmes complexes.

Mais les SN n'offrent qu'un ensemble limité de fonctions de couleurs, nécessitant donc la mise en place d'une méthode de transformation.

Un avantage important de notre méthode et qu'elle prenne en compte le problème fondamental lié au calcul de la propagation d'erreur introduit par le processus de discrétisation. Ce problème est bien décrit dans [23] :

"En science, les termes d'imprécision ou d'erreur ne font pas forcement référence à une méprise ou à une bévue. Au contraire, ils évoquent souvent les imprécisions inhérentes à toutes mesures et qui ne peuvent jamais être complètement éliminées.(...) Une part importante de l'effort scientifique est consacré à la compréhension de ces imprécisions (analyse d'erreur) afin que des conclusions appropriées puissent être déduites des différentes observations. Les étudiants se plaignent souvent que le calcul de l'erreur est plus fastidieux que le calcul des nombres qu'ils essayent de mesurer. Ce qui est souvent vrai. Néanmoins, les mesures peuvent perdre toute signification sans la connaissance des erreurs qui y sont associée."

Notre approche est de vérifier les spécifications d'algorithmes d'un système exprimées en langage mathématique, comme ceux spécifiés dans le projet SAFESPOT (voir chapitre 2 page 60). Puis d'en produire un modèle formel analysable. Les propriétés à analyser doivent aussi être transformées en propriétés formelles. Dans notre méthode nous utilisons le langage CTL.

Nous utilisons donc ici des spécifications constituées de :

– la *spécification* décrivant les algorithmes, exprimées à l'aide de formules mathématiques,
– les *contraintes* qui définissent un ensemble d'assertions à vérifier sur le système (voir chapitre 4 page 83).

Ces spécifications sont modélisées à l'aide de CPN, ce qui permet l'utilisation de fonctions complexes comme celles utilisant les nombres réels. Les fonctions issues des spécifications du système décrivent principalement ses aspects physiques ou les algorithmes et équations qui utilisent des variables physiques du système.

Ces fonctions sont insérées dans le modèle CPN en tant qu'expressions d'arc (*arcs labels*) lors de l'étape de **modélisation**. Le modèle ainsi obtenu est difficile voir impossible à analyser formellement à cause de sa complexité (principalement due aux fonctions complexes et continues). L'étape de **discrétisation** a pour objectif de produire un modèle équivalent en SN. Les SN sont bien adaptés à la spécification de tels systèmes qui sont intrinsèquement symétriques [13]. La contrepartie est une perte de précision sur les fonctions et variables continues qui sont alors discrétisées. La perte de précision, ou propagation d'erreur, est alors calculée lors de cette étape, et répercutée sur les propriétés à vérifier. L'analyse du modèle est enfin effectué lors de l'étape de **vérification formelle**.

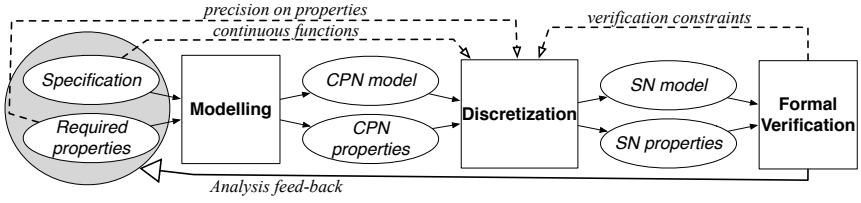


FIGURE 6.1 – Vue d'ensemble de la méthode

La section suivante présente les trois étapes principales de cette méthodologie.

6.1.2 Modélisation

Notre objectif est ici de fournir une représentation plus précise du système en incluant les paramètres continus dans le modèle formel. Le travail principal consiste à identifier les flots de

contrôles et de données impliqués dans les sous-parties du système impliquant des phénomènes continus, ainsi que les opérations effectuées sur ces flots.

Pour simplifier notre recherche de méthode et la compréhension des modèles, nous avons décidé de nous concentrer sur ces sous-parties du système. Ainsi, plutôt que de les intégrer au modèles produits chapitre 5, ce qui produirait des modèles excessivement complexes pour la problématique qui nous motive ici, nous avons décidé de modéliser l'environnement de ces sous partie avec le minimum d'éléments nous permettant de procéder aux analyses.

Les paramètres et variables du système identifiés sont modélisés à l'aide de types CPN. Par exemple, les variables continues du système sont modélisées par le type *réel* du formalisme des CPN. Les fonctions du système qui manipulent les variables continues sont modélisées à l'aide d'expressions d'arcs colorées.

6.1.3 Discrétisation

L'étape de discrétisation transforme le modèle en CPN vers un modèle en SN. Pour ce faire, les variables continues exprimées à l'aide de types *réels* et les fonctions qui les manipulent doivent être discrétisées. Différentes étapes mènent à la discrétisation de ces variables et fonctions continues :

1. discrétisation des éléments (variables et fonctions) continus,
2. calcul de la propagation d'erreur,
3. transformation des types et des fonctions colorées CPN en SN.

Première étape : discrétisation des éléments continus

La discrétisation consiste à transformer un signal, une équation ou un modèle continu en son équivalent discret. En fonction du domaines dans lequel ce processus prend place, on parle de 'numérisation', d' 'échantillonnage', de 'quantification', d' 'encodage' ou de 'discrétisation'.

Les techniques de discrétisation diffèrent en fonction de leur domaine d'application.Voici quelques définitions préliminaires afin de réduire le risque d'ambiguïté :

Définition 6.1.1. *Une **région** est un polygone n-dimensionnel (i.e. un polytope) constitué des points adjacents d'une fonction discrète à n dimensions.*

Définition 6.1.2. *Un **maillage** est un ensemble de régions représentant une fonction discrète à n dimensions, à des fins d'analyse et de modélisation.*

Il existe un grand nombre de méthodes de discrétisation qui peuvent être classées entre méthodes globales et locales, supervisées et non-supervisées, statiques ou dynamiques [39].

- Les **méthodes locales** produisent un partitionnement en régions locales de l'espace d'instance. Ces méthodes utilisent en général des arbres de décision pour produire les partitions (i.e. la classification).
- Les **méthodes globales** (comme le textitbinning) [39] produisent un maillage sur l'intégralité de l'espace d'instance continu à n-dimension où chaque éléments est partitionné en régions. Le maillage contient $\prod_{i=1}^{n} k_i$ régions, avec k_i le nombre de partitions du $i^{\text{ème}}$ élément.

Nous utiliserons ici la méthode dite de **binning à intervalles égaux** pour une première approche de discrétisation. La méthode de binning à intervalles égaux est une méthode globale non-supervisée qui implique la division du domaine des valeurs observées d'une variable en k intervalles de tailles égales, ou k est le paramètre de discrétisation. Si une variable x est bornée dans le domaine défini par x_{min} et x_{max}, la taille de l'intervalle est :

$$\Delta = \frac{x_{max} - x_{min}}{k} \tag{6.1}$$

Deuxième étape : Calcul de la propagation d'erreur

La discrétisation d'une fonction continue introduit une perte de précision (ou marge d'erreur) qui doit être calculée et prise en considération lors des vérifications effectuées sur le modèle.

Il existe plusieurs méthodes de calcul de la propagation d'erreur dans une fonction [82, 23]. La plus courante consiste à déterminer les contributions introduites par les erreurs sur chacune des variables d'entré, puis de combiner ces contributions individuelles en calculant leur moyenne géométrique (i.e. la racine carrée de la somme des carrés).

$$\Delta_{f(x,y,..)} = \sqrt{\Delta_{fx}^2 + \Delta_{fy}^2 + ...} \tag{6.2}$$

Il existe aussi différentes méthodes de calcul des contributions de chacune des variables d'entrées, comme l'utilisation de dérivées partielles ou le calcul de différences .

– La méthode des dérivées partielles calcule la contribution d'une variable x à l'erreur sur la fonction f en faisant le produit de l'erreur sur x (i.e. Δ_x) avec la dérivée partielle de $f(x,y,...)$ par rapport à x :

$$\Delta_{fx} = \frac{\partial f(x,y,..)}{\partial x}\Delta_x \tag{6.3}$$

– La méthode par différences calcule la valeur absolue de la différence suivante :

$$\Delta_{fx} = \mid f(x+\Delta_x,y,..) - f(x,y,..) \mid \tag{6.4}$$

L'utilisation de ces contributions individuelles dans une moyenne géométrique repose sur l'hypothèse que les variables sont indépendantes et qu'elles ont une distribution Gaussienne de leurs valeurs moyennes. Cette méthode donne une bonne estimation de l'erreur sur la fonction. Néanmoins, cette approche reste probabiliste est repose sur des hypothèses qui ne peuvent être garanties ici.

Nous choisissons une autre méthode qui n'introduit pas de probabilité et ne fait aucune hypothèse sur les variables. L'approche est ici d'obtenir la valeur exacte de l'erreur maximum propagée même si sa probabilité d'occurrence est faible.

Soit f une fonction de $\mathbb{R}$ dans $\mathbb{R}$ qui à x associe $f(x)$, x une variable continue ou réelle, et x_{disc} la valeur discrète de x.

Si nous choisissons un pas de discrétisation de $2 * \Delta_x$ nous pouvons dire que pour chaque x_{disc} image de x par discrétisation : $x \in [x_{disc} - \Delta_x, x_{disc} + \Delta_x]$ (ce qui est souvent écrit de manière simplifiée $x = x_{disc} \pm \Delta_x$). On peut calculer les bornes de $\Delta_{f(x)}$ introduit par la discrétisation :

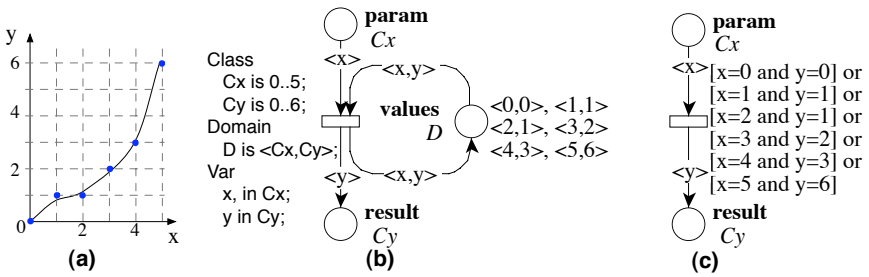


FIGURE 6.2 – Exemple de discrétisation de fonction dans une place ou dans la garde d'une transition

$$f(x) = f(x_{disc}) \pm \Delta_{f(x)} \qquad \Delta_{f(x)} = f(x \pm \Delta_x) - f(x) \tag{6.5}$$

On peut aussi dire que l'erreur sur $f(x)$ est dans l'intervalle :

$$\Delta_{f(x)} \in [Min(f(x \pm \Delta_x) - f(x)), Max(f(x \pm \Delta_x) - f(x))] \tag{6.6}$$

Cette méthode s'applique aussi aux fonctions à plusieurs variables. Pour une fonction f à n variables $f(x \pm \Delta_x, y \pm \Delta_y, ..)$ et à 2^n solutions. Les bornes maximales d'erreurs sur f sont :

$$\Delta_f \in [Min(f(x \pm \Delta_x, y \pm \Delta_y, ..) - f(x, y, ..)), Max(f(x \pm \Delta_x, y \pm \Delta_y, ..) - f(x, y, ..))] \tag{6.7}$$

C'est cette méthode que nous utilisons dans l'exemple présenté dans les sections suivantes de ce chapitre concernant le module de freinage d'urgence de l'application H&IW (implémenté dans l'activité *SafetyMarginComputation* présentée au chapitre 5).

La perte de précision peut être prise en considération de deux manières. Dans le modèle en SN, afin de conserver une cohérence avec les contraintes exprimées sur le système. L'autre solution consiste à modifier les propriétés à vérifier (dérivées des contraintes du système) en prenant en compte cette imprécision.

Troisième étape : Transformation en Réseaux de Petri Symétriques

Certaines variables du réseau CPN n'ont pas à être transformées, car elles sont déjà exprimées à l'aide de types énumérés (discrets). La transformation des variables réelles résulte de la discrétisation des domaines infinis vers des domaines énumérés.

Pour effectuer la modélisation des fonctions complexes dans les SN (comme par exemple pour le calcul d'une distance de freinage en fonction de la vitesse d'un véhicule), nous devons les discrétiser et les représenter soit dans une place spécifique, soit comme une garde de transition.

La figure 6.2 représente un exemple de discrétisation de fonction. La partie gauche (a) montre la fonction ($f(x) = y$) qui est discrétisée et la partie droite montre les SN correspondant : dans le modèle (b) la fonction est discrétisée à l'aide d'une place, dans le modèle (c) la fonction est discrétisée dans une garde de transition. Dans les deux cas, les associations des variables x et y sont sélectionnées au tirage de la transition. Il est à noter que dans le modèle (b), le marquage reste constant.

Cette technique peut être généralisée à toute fonction $f(x_1, x_2, ..., x_n)$ quelle que soit sa complexité.

Il est alors important de remarquer que :

– la discrétisation d'une fonction devient une hypothèse de modélisation qui doit être validée séparément (pour évaluer l'impact de l'imprécision introduite par la discrétisation),
– pour une fonction, il est relativement simple de générer automatiquement la liste des valeurs qui doivent être stockées dans le marquage initial de la place ou dans la garde de la transition correspondante.

Le seul inconvénient de cette technique est la perte de précision par rapport aux systèmes continus qui nécessitent une approche hybride [31]. Ainsi, le choix de la méthode de discrétisation et son paramétrage doivent être étudiés, par exemple pour garantir que l'imprécision introduite reste dans un intervalle correct. Il est aussi possible de modéliser les fonctions en utilisant des inéquations dans la garde de la figure 6.2(c). Néanmoins, l'introduction de variables libres dans les deux parties de la comparaison casse les symétries dans le modèle. C'est pourquoi dans la figure 6.2(c), la garde effectue des comparaisons entres une variable libre et une constante.

Une fois les variables et fonctions discrétisées le modèle CPN peut être transformé en réseau Symétrique.

6.1.4 Vérification

Nos modèles sont vérifiés à l'aide de :
– *Techniques structurelles* (calcul d'invariants, borne structurelles, etc.) sur le réseau déplié P/T. Comme notre réseau est coloré, un outil de dépliage adapté aux gros systèmes [78] est utilisé pour obtenir le réseau P/T et procéder aux analyses structurelles.
– *Model checking*, nous avons élaboré des techniques efficaces de model checking dédié à ce type de systèmes et utilisant de manière intensive les symétries et les diagrammes de décision. Ces techniques se sont révélées très efficaces pour ce type de système en exploitant leur régularité [65, 13].

Si les classes symétriques colorées sont trop grandes (i.e. l'intervalle de discrétisation est trop petit), alors se pose le problème d'explosion combinatoire (aussi bien pour le modèle checking que pour les analyses structurelles). Inversement, si l'erreur introduite par la discrétisation est trop grande, les propriétés perdent leur "précision" et peuvent devenir insignifiantes. Un compromis entre explosion combinatoire et précision du modèle doit être trouvé.

6.2 Modélisation

Pour illustrer cette méthode nous prenons pour cas d'étude le module de freinage d'urgence de l'application *Hazard and Incident Warning* qui implémente la stratégie de sécurité de l'application [1]. Sa vérification nécessite la prise en charge de paramètres et de fonctions continues.

Dans cette section nous présentons les spécifications de ce module et leur modélisation en SN à l'aide de notre méthode.

1. Le concept de *Safety Margin* du projet SAFESPOT que nous avons présenté chapitre 3 fait écho aux calculs relatifs à la sécurité des usagers, en opposition aux autres calculs nécessaires au fonctionnement du système comme l'accès à la base de données ou le routage des messages.

6.2.1 Spécification détaillée des algorithmes : le concept de la marge de sécurité

L'objectif d'*Hazard and Incident Warning* est d'alerter les conducteurs en cas d'événements dangereux sur la route. Les événements pris en compte sont par exemple l'accident, la présence d'obstacles, d'embouteillage, de piétons ou encore un véhicule roulant à contre-sens. L'application analyse aussi les conditions environnementales qui pourraient influencer le coefficient d'adhérence de la chaussée ou réduire la visibilité des conducteurs. Basée su les informations fournies par les capteurs des véhicules et de l'infrastructure, l'application fournit des alertes aux conducteurs.

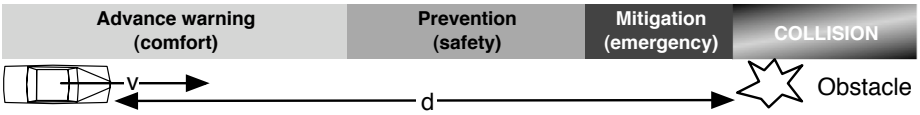


FIGURE 6.3 – Stratégie de sécurité du module de freinage d'urgence

Le module de stratégie que nous étudions ici est appelé *module de freinage d'urgence* (*Emergency Braking Module*). Il est implémenté dans l'activité *SafetyMarginComputation()* du *ScenarioManager* présentée chapitre 5.

En cas d'obstacle sur la route, le module récupère la vitesse, la capacité de freinage et la distance relative à l'obstacle du véhicule considéré. Avec ces données, il calcule la commande de sécurité qui doit être transmise au conducteur et aux autres applications de SAFESPOT. Cette commande représente le niveau de sécurité du véhicule. Les trois types de commandes (ou alertes) délivrés par le module sont "Comfort" si aucune action du conducteur n'est requise, "Safety" si le conducteur est supposé commencer une décélération, et "Emergency" si le conducteur doit commencer rapidement un freinage d'urgence. La figure 6.3 présente ces trois commandes pour une situation donnée (vitesse, capacité de freinage, etc.).

Les réseaux de Petri sont particulièrement bien adaptés pour décrire et analyser ce type d'application. Néanmoins, une partie de l'application "Hazard and Incident Warning" est basée sur l'analye de variables continues comme la vitesse d'un véhicule ou sa distance à un obstacle. Ces données font partie du flot de données du système, mais elles sont aussi déterminantes pour son flot de contrôle.

Beaucoup de propriétés peuvent être vérifiées grâce aux réseau de Petri sans que les variables continues ne soient modélisées. Mais pour certaines propriétés, comme celles présentés section 6.2.3, leur modélisation est indispensable.

6.2.2 Modèle mathématique du module de freinage d'urgence

Le *module de freinage d'urgence* implémente une fonction pour déterminer le niveau de sécurité d'un véhicule. Cette fonction calcule la *distance de freinage* d'un véhicule grâce à sa vitesse et à sa capacité de freinage. Soit $v \in V$ la vitesse d'un véhicule tel que $V \subset \mathbb{R}^+$.
Soit $b \in B$ la capacité de freinage du véhicule tel que $B \subset \mathbb{R}^+$.
La distance de freinage est alors :

$$f(v,b) = \frac{v^2}{2b} \tag{6.8}$$

Soit $d \in D$ la distance entre l'obstacle et le véhicules avec $D \subset \mathbb{R}^+$.

L'algorithme principal du module de freinage d'urgence définit deux seuils pour déterminer quand un véhicule passe d'un état "Confort" (*"Comfort state"*) à un état "Sécurité" (*"Safety state"*), et d'un état "Sécurité" à un état "Urgence" (*"Emergency state"*). Ces seuils sont définis à partir du temps qu'il reste au conducteur pour réagir. Si le conducteur à plus de trois secondes disponibles pour réagir, il est dans l'état "confort". S'il a entre trois secondes et une seconde pour réagir il est dans l'état "Sécurité". S'il a moins d'une seconde pour réagir, il est alors dans l'état "Urgence". Ces seuils sont définis ainsi :

$$EB\_Safety = \frac{v^2}{2b} + v * 3 - d \tag{6.9}$$

$$EB\_Emergency = \frac{v^2}{2b} + v * 1 - d \tag{6.10}$$

L'algorithme de la fonction est :

```
function Eb_Strategy(d,v,b){
  Eb_Safety = (v^2)/(2b) + v * 3 - d;
  Eb_Emergency = (v^2)/(2b) + v * 1 - d;
  if (Eb_Safety < 0) then  Command = 'Comfort';
  else if (Eb_Emergency < 0) then Command = 'Safety';
     else  Command = 'Emergency'  endif
  return Command;}
```

Pour l'application "Hazard and Incident Warning"(H&IW), et donc pour son module de freinage d'urgence, les valeurs de v sont supposées appartenir à l'ensemble $[0,46]m/s$, b dans $[3,9]m/s^{-2}$ et d dans $[0,500]m$. Si les variables sont en dehors de ces ensembles, elle doivent être prises en charge par d'autres applications : on est en dehors du champ d'action de l'application. Par exemple, les vitesses supérieures à $46m/s$ sont prises en charge par l'application "Speed Arlert".

6.2.3 Contraintes de fonctionnement

Les spécifications de SAFESPOT en général et de H&IW en particulier, comprennent un ensemble de contraintes de fonctionnement (*"required properties"*) structurées conformément à la méthode FRAME [97] que nous avons présentée chapitre 4. L'analyse de ces contraintes pour l'application H&IW montre que 18 des 47 contraintes principales (i.e. 38%) impliquent des notions continues comme l'espace ou le temps. L'étude présentée concerne ces contraintes à composantes continues. En voici un exemple pour le module de freinage d'urgence :

Propriété 1 : Les commandes doivent être activées de manière appropriée. Ce qui peut être détaillé comme suit :

- **1.1** : Quand la distance de freinage d'un véhicule est inférieure à sa distance le séparant de l'obstacle plus la distance parcourue en une seconde (temps de réaction du

conducteur) par le véhicule, l'application H&IW doit déclencher une alerte de type "Urgence".
- **1.2** : Quand la distance de freinage d'un véhicule est en dessous de sa distance à l'obstacle plus la distance parcourue en trois secondes par le véhicule, l'application H&IW doit déclencher une alerte de type "Sécurité".
- **1.3** : Quand plusieurs obstacles sont présents sur la route, l'application H&IW doit déclencher les alertes associées à chacun des obstacles.

Propriété 2 : les alertes doivent être déclenchées progressivement du niveau le plus faible au niveau le plus élevé.
- Quand un véhicule s'approche d'un obstacle, l'application H&IW doit déclencher les alertes *Confort, Sécurité, Urgence* dans l'ordre correspondant au niveau de danger menaçant le véhicule. Ainsi, si un conducteur ne réagit pas à la première alerte de "Confort", les alertes de "Sécurité" puis d'"Urgence" doivent être déclenchées.

6.2.4 Spécification en Réseaux de Petri Symétriques

Pour simplifier l a modélisation du module de freinage d'urgence de l'application H&IW, son environnement est simplifié : les données sont récupérées d'une interface, une commande est alors calculée et transmise à une autre interface. Le modèle de la figure Fig. 6.4 présente ce comportement générique.

La transition `Get_Data` a deux arcs entrant venant des places `Interface_Call` et `Interface_Data`. La place `Interface_Call` est typée par un sous-ensemble d'entiers `PROCESSID` (il s'agit ici d'un jeton de valeur "1") modélisant un flot contôle. Une fois le processus invoqué et les données récupérées, la place `Step1` contient un couple de jetons (pid,data). La transition `Process_Strategy` fournit une commande résultant d'un calcul sur les données.

Ce schéma générique, présenté Fig. 6.5, est ensuite instancié pour le module de freinage d'urgence en changeant les types `generic_data` et `generic_command` par respectivement `EB_DATA` et `EB_COMMAND`.

Les données utilisées par ce module sont composées d'une distance (`Distance`) , d'une vitesse (`Velocity`) et d'un indice de freinage(`Braking_Factor`) : le domaine de couleurs `EB_DATA = produit Distance * Velocity * Braking_Factor`.

Ces données modélisant l'état des véhicules appartiennent, dans les spécifications, à l'ensemble $\mathbb{R}^*$.

Pour la modélisation en CPN, nous pouvons utiliser ce typage afin de conserver ce niveau d'expressivité, mais le modèle est ainsi inexploitable dans le cadre d'une analyse exhaustive.

Le type `EB_COMMAND` a trois valeurs possibles : `EB_COMMAND = Comfort | Safety | Emergency`.

La commande appropriée résulte du calcul de la fonction `EB_Strategy`.

6.3 Discrétisation

En partant du modèle CPN, nous appliquons dans cette section la méthode de discrétisation présentée section 6.1.3 afin de produire les modèles SN.

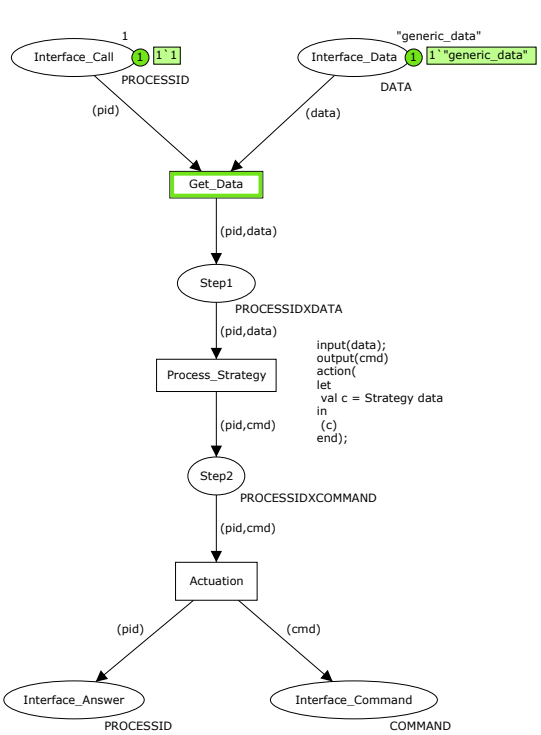


FIGURE 6.4 – Patron de l'application H&IW en Réseau de Petri Coloré

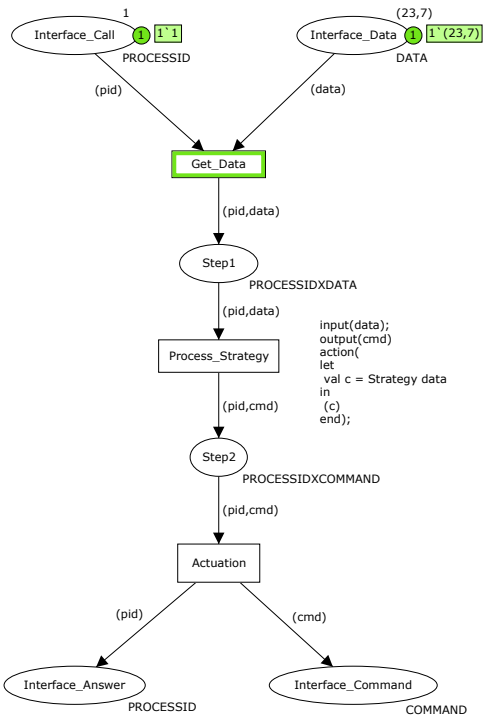


FIGURE 6.5 – Instanciation du module de freinage d'urgence en Réseau de Petri Coloré

6.3.1 Implémentation de fonctions complexes en Réseau de Petri Symétriques

D'abord, les types CPN doivent être discrétisés. En utilisant la méthode de discrétisation de binning à intervalles égaux (présentée section 6.1.3) avec des intervalles de k_v, k_b et k_d pour chacune des variables, nous obtenons un maillage de $k_v \times k_b \times k_d$ régions (voir les définitions 6.1.2 et 6.1.1) pour la fonction discrétisée. Les ensembles de définition des variables d, v et b sont maintenant composés de k éléments ordonnés. Par exemple, avec $k = k_v = k_b = k_d = 10$ le type discétisé de v est $[0, 4.6, 9.2, ..., 46]$ et la fonction de freinage d'urgence discrétisé contient 10^3 régions. Avec $k = 100$, le domaine de v est $[0, 0.46, 0.92, ..., 46]$ le maillage est composé de 10^6 régions.

La section 6.1.3 présente deux solutions de modélisation des fonctions complexes en réseaux symétriques. Nous choisissons la solution b (6.2) car elle est plus performante lors du calcul de l'espace d'état symbolique. Ainsi, nous ajoutons la place **EB_Strategy_Table** dans le réseau symétrique (son marquage initial est présenté en section 6.3.4).

Nous avons choisi une méthode de discrétisation simple et générique qui ne prend pas en compte la spécificité des fonctions à discrétiser. D'autres méthodes de discrétisation comme celles utilisant des intervalles de tailles variables permettraient de réduire le nombre de marquages tout en conservant le même niveau de précision moyennant la prise en compte des spécificités de la fonction. Ces aspects sont discutés en sections 6.5.1 et 6.5.2.

Valeurs de v, b, d k / $card(EBData)$	$v = 13m/s,\ b = 8m/s^{-2},$ $d = 500m$	$v = 36m/s,\ b = 4m/s^{-2},$ $d = 100m$
10 / 10^3	$\Delta_{Eb\_Saf} \in [-70.83m, 74.84m]$ $\Delta_{Eb\_Emerg} \in [-61.64m, 65.64m]$	$\Delta_{Eb\_Saf} \in [-118.9m, 144.5m]$ $\Delta_{Eb\_Emerg} \in [-109.7m, 135.3m]$
20 / $8 * 10^3$	$\Delta_{Eb\_Saf} \in [-35.87m, 36.81m]$ $\Delta_{Eb\_Emerg} \in [-31.27m, 32.26m]$	$\Delta_{Eb\_Saf} \in [-61.97m, 68.28m]$ $\Delta_{Eb\_Emerg} \in [-57.37m, 63.68m]$
50 /$12.5 * 10^3$	$\Delta_{Eb\_Saf} \in [-14.45m, 14.61m]$ $\Delta_{Eb\_Emerg} \in [-12.62m, 12.77m]$	$\Delta_{Eb\_Saf} \in [-25.47m, 26.47m]$ $\Delta_{Eb\_Emerg} \in [-23.63m, 24.63m]$
100 /10^6	$\Delta_{Eb\_Saf} \in [-7.25m, 7.29m]$ $\Delta_{Eb\_Emerg} \in [-6.33m, 6.37m]$	$\Delta_{Eb\_Saf} \in [-12.85m, 13.10m]$ $\Delta_{Eb\_Emerg} \in [-11.93m, 12.19m]$

TABLE 6.1 – Bornes d'erreurs pour différents paramètres de discrétisation

De plus, en fonction des propriétés à analyser, il est aussi possible de calculer et d'utiliser des classes d'équivalence. Cet aspect est discuté en section 6.4.2.

6.3.2 Calcul de la propagation d'erreurs dans les SN

Nous calculons l'erreur de précision introduite par la discrétisation sur les "seuils" du module de freinage d'urgence. L'incertitude produite est :

$$\Delta_{Eb\_Safety} = Eb\_Safety(v \pm \Delta_v, b \pm \Delta_b, d \pm \Delta_d) - Eb\_Safety(v, b, d) \tag{6.11}$$

$$\Delta_{Eb\_Safety} = (\frac{(v \pm \Delta_v)^2}{2(b \pm \Delta_b)} + 3(v \pm \Delta_v) - (d \pm \Delta_d)) - (\frac{v^2}{2b} + 3v - d) \tag{6.12}$$

$$\Delta_{Eb\_Safety} = \frac{(v \pm \Delta_v)^2}{2(b \pm \Delta_b)} - \frac{v^2}{2b} \pm 3\Delta_v \pm \Delta_d \tag{6.13}$$

Par exemple, considérons (cf. Table 6.1) un véhicule léger classique roulant à $v = 13m/s$ (i.e. $50km/h$), sur route sèche (i.e. $b = 8m/s^2$) à $d = 500m$ d'un obstacle. Si l'on prend un pas de discrétisation de $k = 100$ et une erreur de $\pm 0.45m/s$ pour v, $\pm 0.06m/s^2$ pour d et $\pm 5m$ pour p. Nous obtenons : [2] $\Delta_{Eb\_Safety} \in [-7.25m, +7.29m]$ $\Delta_{Eb\_Emergency} \in [-6.33m, +6.37m]$

Pour le même véhicule situé à 100 mètres de l'obstacle, circulant à $v = 36m/s$ (i.e. $130km/h$), sur une route mouillée (i.e. $b = 4m/s^2$), nous obtenons :

$\Delta_{Eb\_Safety} \in [-12.85m, +13.10m]$ $\Delta_{Eb\_Emergency} \in [-11.93m, +12.19m]$

Ces résultats fournissent le niveau de précision que l'on pourra obteni rlors de la validation des propriétés sur le modèle SN. Le tableau 6.1 présente différentes marges d'erreurs calculées pour quatre valeurs différentes du paramètre k. La précision des seuils dépend de ce paramètre, mais aussi de la valeur des variables. Par exemple, les valeurs de v et b sont plus déterminantes dans le calcul de l'intervalle d'erreur que d. L'exploitation de ces erreurs pour valider le modèle en Réseaux de Petri Symétriques et ses propriétés, est présentée dans la section suivante.

2. Dans la table 6.1, Δ_{Eb_Saf} et Δ_{Eb_Emerg} signifient Δ_{Eb_Safety} et $\Delta_{Eb\_Emergency}$ respectivement.

6.3.3 Validation de la discrétisation dans les Réseaux de Petri Symétriques

La discrétisation des variables et fonctions dans le modèle en SN de la figure 6.6 introduit des imprécisions (ou erreurs). En fonction des propriétés qui doivent être vérifiées, cette imprécision doit être prise en compte.

Par exemple, les propriétés présentées sections 6.2.3 peuvent être vérifiées par l'utilisation de formules CTL (Computation Tree Logic) [42]. Avec un facteur de discrétisation de $k = 100$ valeurs sur les variables d'entrée (voir la dernière ligne de la table 6.1), la propriété 1 peut être vérifiée avec une précision inférieure à $\pm 7,3m$ sur la distance avec une vitesse de $13m/s$ sur route sèche ($b = 8m/s^{-2}$).

Si l'imprécision est acceptable au regard des propriétés à vérifier, le concepteur du système peut valider la discrétisation valide pour ces propriétés. Sinon, une plus grande précision peut être requise et nécessiter une nouvelle discrétisation.

On peut aussi propager cette imprécision dans les formules CTL. Pour ce faire, des valeurs d'entrée plus contraignantes doivent être choisies dans la formule (i.e. une vitesse plus élevée, un indice de freinage plus faible ou un obstacle plus proche). Dans notre cas, le plus simple est de choisir un obstacle plus proche en y intégrant la marge d'erreur. Par exemple, la formule CTL modélisant une partie de la propriété 1 :

```
AG((EB_Data_Retrieved == <13,8,500>) => AX(EB_Cmd_Cpt==<Safety>))
```
devient :
```
AG((EB_Data_Retrieved == <13,8,(500-7.29)>) => AX(EB_Cmd_Cpt==<Safety>)).
```

Dans certains cas, il est possible de calculer le facteur de discrétisation des variables en fonction de la précision désirée. Cette approche est décrite section 6.5.2.

6.3.4 Transformation pour obtenir le Réseau de Petri Symétrique

Cette section présente l'étape de transformation du modèle CPN vers le modèle SN. Nous présentons d'abord les principes généraux qui sont ensuite appliqués aux modèles dédiés à la vérification des deux propriétés définies en section 6.2.3.

Calcul de la discrétisation

Notre objectif est d'obtenir un réseau symétrique à partir du modèle présenté figure 6.5. Pour ce faire, nous devons *i)* discrétiser les types continus et, *ii)* générer la table correspondant à la fonction qui doit être modélisée. Le résultat est présenté en figure 6.6.

- Le type `EB_DATA` du modèle 6.5 est associé à *EBData* (figure 6.6) qui est le produit cartésien des trois types discrets `Distance`, `Velocity` et `Braking_Factor`. La figure 6.6 ne présente pas la définition de ces types car ils dépendent du critère de discrétisation.
- Comme nous l'expliquons section 6.1.3, la fonction complexe `EB_Strategy` de la figure 6.5 est associée à la place **EB_Strategy_Table** du modèle présenté en 6.6. Son marquage initial est une table d'association pour la fonction discrète. L'étiquetage de l'arc entrant

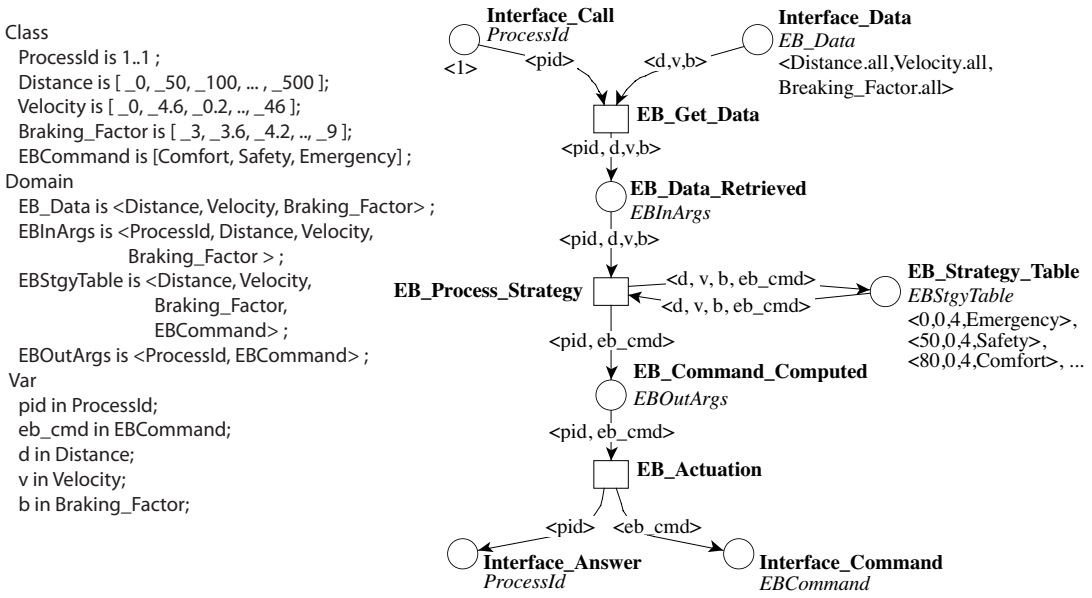


FIGURE 6.6 – Réseau de Petri Symétrique du module de freinage d'urgence (la marquage de la place **EB_Strategy_Table** n'est pas complet pour simplifier la lecture)

de la transition **EB_Process_Strategy** avec les variables d, v, b permet la sélection de la commande adéquate dans la variable eb_cmd.

– Le marquage de la place **EB_Strategy_Table** est généré à partir des valeurs discrètes des domaines `Distance`, `Velocity`, et `Braking_Factor` de la fonction `EB_Strategy` présentée en section 6.2.2.

Modèle de vérification de la propriété 1 :

Le modèle de la figure 6.6 peut être utilisé directement (i.e. sans être modifié) pour vérifier la propriété 1.

Le marquage de la place **Interface_Data** doit fournir toutes les valeurs possibles en entrée afin que l'on puisse vérifier que dans tous les cas la commande adéquate sera émise conformément aux propriétés 1.1 et 1.2. Ce marquage est obtenu à l'aide de fonctions de diffusion sur les domaines : <Distance.all,Velocity.all,Braking_Factor.all>.

Pour modéliser plusieurs threads fonctionnant en parallèle sur différents obstacles afin de vérifier la propriété 1.3 (6.2.3), il suffit d'adapter le marquage de la place **Interface_Call** en modifiant la constante **NBT**. Ainsi, le marquage initial de la place **Interface_Call** peut être obtenu par la fonction de diffusion <ProcessId.all>.

Cela ajoute peu de complexité au modèle, n'a pas d'impact sur le calcul de l'imprécision et permet la vérification de la propriété 1.3.

Modèle de vérification de la propriété 2 :

Cette propriété implique une interaction avec l'environnement fonctionnel (*runtime*)du module et une interaction avec un conducteur (*a passive driver*).

La figure 6.7 présente le modèle sur lequel ont été ajouté :

– un conducteur passif (ne prenant aucune décision) sur la droite du modèle,

– l'environnement fonctionnel sur la gauche, représenté par une boucle de rétro-action.

Comme le conducteur ne réagit pas aux alertes, les différents niveaux d'alerte (Comfort Safety Emergency) seront déclenchés successivement à mesure qu'augmente le dangerosité de la situation.

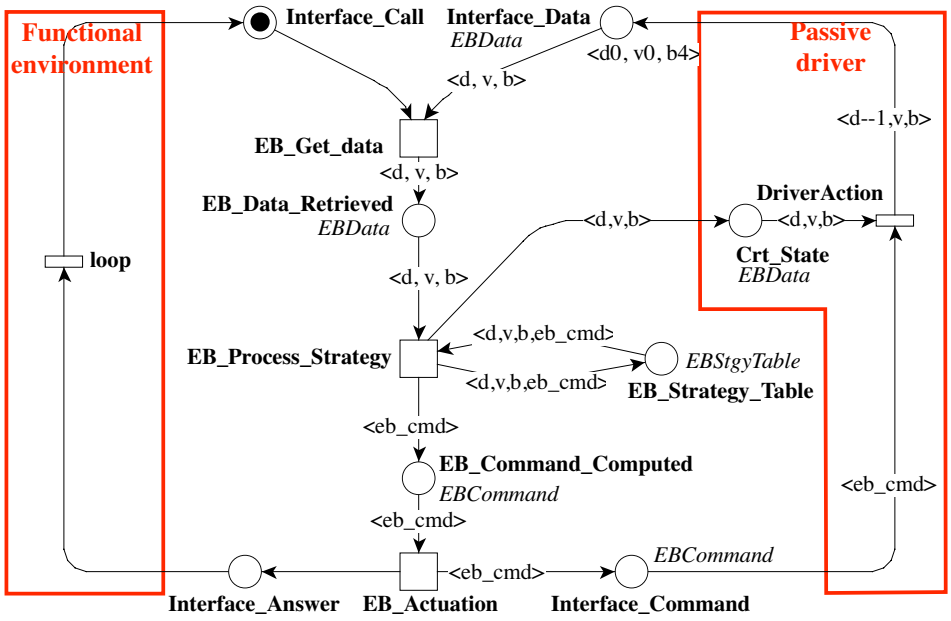


FIGURE 6.7 – Modèle de la figure 6.6 enrichi avec un conducteur passif (le marquage de la place **EB_Strategy_Table** et celui du modèle de la figure 6.6)

6.3.5 Synthèse

Cette section présente la modélisation nécessaire à la vérification des propriétés sur le module de freinage d'urgence. Les modèles sont basés sur le canevas proposé en figure 6.5 et modifiés en fonction des contraintes des propriétés 1 et 2 présentées en section 6.2.3.

Ces modifications ont un impact sur la complexité de la vérification. Par exemple, le graphe d'accessibilité du second modèle (celui avec le driver passif) croît de manière linéaire avec le niveau de discrétisation du domaine `Distance`. À l'inverse, la complexité du premier modèle croît plus rapidement.

Dans ces exemples, la boucle de rétroaction contient uniquement des variables discrètes comme une commande ou un identifiant de processus, mais pas de variable continue discrétisée. Cela empêche l'accumulation d'erreurs introduites par la discrétisation. Si une variable discrétisée était impliquée dans une boucle de rétroaction, cela introduirait une contrainte supplémentaire de discrétisation pour garantir d'une certaine précision. Il serait aussi nécessaire de garantir que l'accumulation d'erreurs ne se produise pas à l'infini.

La section suivante présente les résultats d'analyse. Le premier modèle posant plus de problèmes d'explosion combinatoire, il est le plus étudié.

6.4 Analyse

Cette section présente les analyses effectuées sur le modèle de vérification de la propriété 1 (figure 6.6) avec différentes configurations.

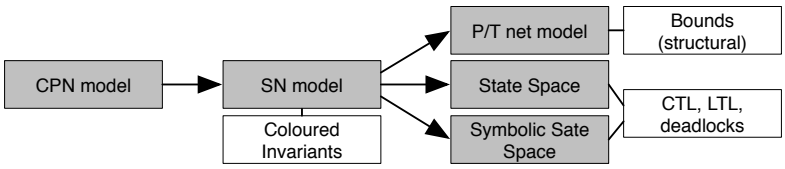


FIGURE 6.8 – Vue d'ensemble de l'analyse

La discrétisation de fonctions dans un SN génère des places avec des marquages de grande taille. Il est important d'en étudier les conséquences sur l'analyse et sur les outils de vérification.

Objectifs Les objectifs de l'analyse sont de vérifier les propriétés tout en gérant le problème d'explosion combinatoire.

Protocole expérimental La figure 6.8 présente les différentes techniques qui peuvent être utilisées pour vérifier les propriétés 1 et 2. Les analyses structurelles comme les invariants de places, les limites du réseau, etc., peuvent être effectuées directement sur le réseau déplié en réseau de Petri Place/Transition. Les invariants colorés peuvent aussi être calculés directement à partir du réseau coloré. Il est aussi possible de calculer les espaces d'états symboliques afin de procéder à une analyse comportementale.

Considérations techniques L'analyse des réseaux de Petri est une tâche complexe impliquant différentes transformations du modèle comme le dépliage ou des réductions. Plusieurs outils sont utilisés pour mener à bien ces analyses. Les modèles en Réseaux de Petri Colorés sont réalisés avec l'outil CPN-Tools [34], puis les Réseaux de Petri Symétriques sont réalisés à l'aide de Coloane et de ses interfaces vers CPN-AMI [83] et PetriScript (i.e. un langage de script pour la modélisation de Réseaux de Petri Symétriques) pour la génération des marquages initiaux. L'outil CPN-AMI est utilisé pour l'analyse des réseaux symétriques.

6.4.1 Analyse structurelle

La première analyse effectuée est l'analyse structurelle. Ne nécessitant pas le calcul de l'espace d'état du modèle, elle ne pose pas de problème d'explosion combinatoire.

Analyse des Réseaux de Petri Symétriques Nous avons calculé les invariants colorés sur le réseau symétrique avec différents niveaux de discrétisation. Le seul invariant implique la place "EB_Strategy_Table" comme prévu (son marquage est stable par définition). La discrétisation n'a pas d'impact significatif sur la mémoire nécessaire à ce calcul.

Réduction structurelle L'utilisation de réductions structurelles sur le réseau de Petri [10, 54, 56] permet de réduire le réseau de la figure 6.6 vers le réseau 6.9. Ils sont structurellement équivalents vis-à-vis de la propriété 1. Le réseau réduit permet d'éliminer les entrelacements inutiles quand plusieurs threads sont exécutés en parallèle.

Le réseau réduit de la figure 6.9 possède la même déclaration (i.e. le même marquage initial) que celui de la figure 6.6.

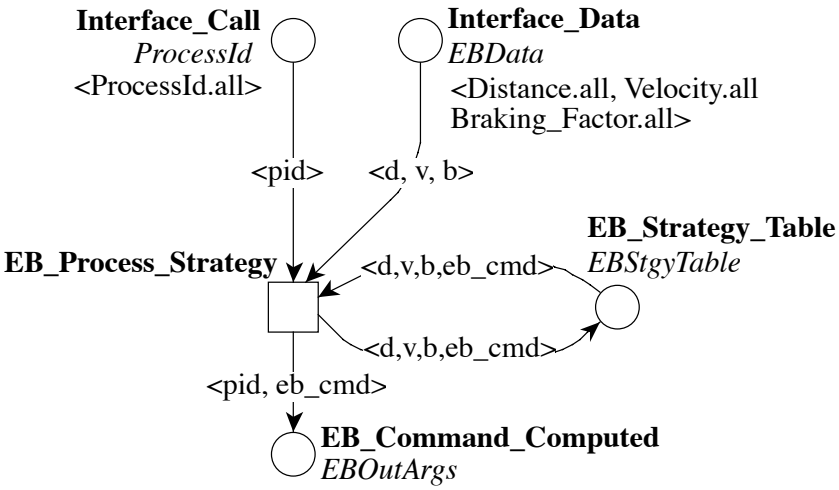


FIGURE 6.9 – Réseau réduit à partir du modèle de la figure 6.6 (sans le marquage de la place **EB_Strategy_Table**)

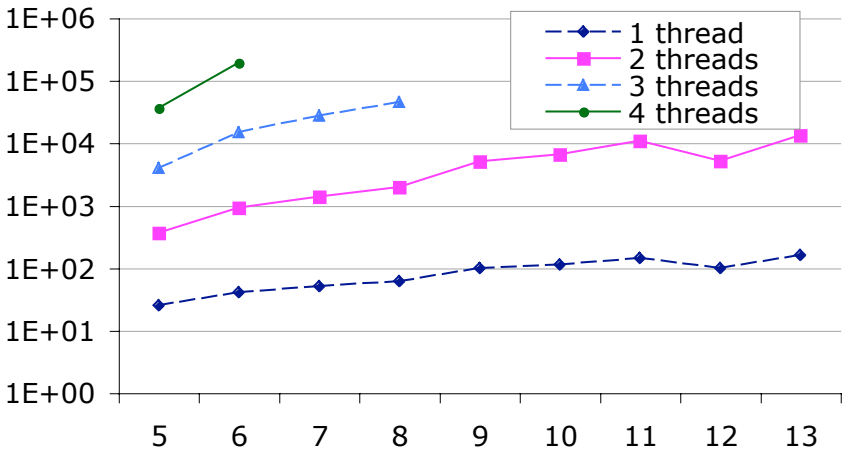


FIGURE 6.10 – Évolution de l'espace d'état symbolique du réseau de la figure 6.9 quand la discrétisation et le nombre de threads varient

Analyses des réseaux dépliés Le paramètre de discrétisation (k) a un impact sur la mémoire nécessaire au dépliage du réseau. La taille de la place **EB_Data** qui contient la fonction discrétisée a une croissance cubique par rapport au paramètre de discrétisation ce qui se répercute directement sur le réseau déplié en réseau P/T. Le nombre de places dépliées est de $5 * (k)^3 + 8$ (avec k le paramètre de discrétisation). Ce qui est confirmé par les outils de dépliage de CPN-AMI.

Il est possible de prouver que les réseaux (6.6 et 6.9) dépliés sont bornés. Dans la mesure où le marquage du réseau n'a pas d'impact sur les propriétés structurelles, notre méthodologie est adaptée à ce type d'analyse.

6.4.2 Analyse comportementale

Les deux outils de *model checking* PROD [123] et GreatSPN [28] sont utilisés pour l'analyse comportementale. Nous présentons ici les résultats de l'analyse de la complexité liée à la construction des graphes d'accessibilité et à la vérification des formules CTL.

Calcul de l'espace d'états La complexité de la génération de l'espace d'état est semblable à celle du dépliage en temps et en mémoire. L'espace d'état symbolique généré par GreatSPN montre une optimisation mineure comparée au calcul explicite effectué avec PROD.

La complexité du tirage de la transition **EB_Process_Strategy** est :

$$\sum_{i=1..NbP} C^i_{IDM} \tag{6.14}$$

où $NbP = |ProcessId|$ (i.e. la cardinalité du type *ProcessId* modélisant nombre de threads) et $IDM = |M_0(\textbf{Interface\_Data})|$ (le nombre d'éléments/jetons dans la place **Interface_Data**). Comme $M_0(\textbf{Interface\_Data})$) contient $|Distance| \times |Velocity| \times |Braking\_Factor|$ jetons, le réseau génère rapidement un gros marquage.

Il n'est donc pas surprenant que PROD ne puisse générer l'espace d'état que pour un nombre de threads limité ou un nombre restreint de valeurs pour les types `Distance` et `Velocity` [3]. Cela est dû au fait que PROD écrit en mémoire une représentation explicite de l'espace d'état.

Des mesures effectuées avec GreatSPN sont fournies figure 6.10. Ce sont les valeurs obtenues dans un délais de calcul inférieur à 24h sur un processeur à 2,33 GHz.

La figure 6.10 montre que GreatSPN résiste mieux à l'explosion combinatoire induite par le parallélisme quand plusieurs threads accèdent à la fonction de freinage d'urgence (nous utilisons la détection des symétries présentée dans [117]).

Certaines discrétisations ne peuvent être calculées quand plus de deux threads sont implémentés pour des problèmes de temps plus que pour des problèmes de mémoire. Cela est dû à la complexité du tirage de la transition **EB_Process_Strategy** qui n'est symétrique qu'au regars du domaine de `ProcessId`. À titre d'exemple, le nombre de tirages pour **EB_Process_Strategy** (calculé grâce à la formule 6.14) sur le modèle avec 7 valeurs pour les domaines `Distance` et `Velocity` est :

3. Dans nos expérimentations nous avons limité la capacité de freinage à deux valeurs correspondant aux valeurs moyennes sur route sèche et route humide.

$$C_{98}^{1} + C_{98}^{2} + C_{98}^{3} + C_{98}^{4} = 3\,769\,227 \tag{6.15}$$

vu que les places **Interface_Data** et **Interface_Call** contiennent initialement 98 et 4 jetons. Le temps de calcul excessif est dû à la fonction de canonisation nécessaire à la génération de l'espace d'états symboliques. L'utilisation de la mémoire n'a jamais dépassé 100Mbyte.

6.4.3 Gestion du problème de complexité

Notre approche par discrétisation nous permet de réduire un problème infini en un problème discret. Néanmoins, la vérification pour une discrétisation raisonnable pose encore des problèmes d'explosion combinatoire. Cette section présente différentes pistes pour gérer ce problème.

La première solution proposée consiste en une technique spéciale de modélisation utilisable pour les calculs d'accessibilité. La seconde est basée sur l'introduction d'algorithmes dédiés dans le *model checker*.

Classes d'équivalence sémantiques

Dans un Réseau de Petri Symétrique l'espace d'états symboliques est calculé à partir des *classes d'équivalence*. Elles sont soit définies par le concepteur du modèle sur le domaine de couleurs, soit calculées automatiquement à partir de la structure du réseau [117]. Ici, la structure de notre réseau n'est pas compatible avec ces méthodes.

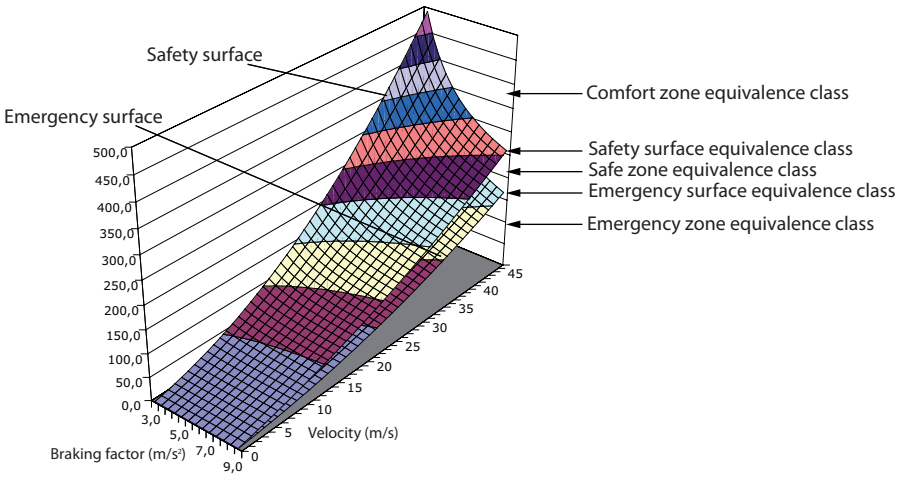


FIGURE 6.11 – Classes d'équivalence sémantique déduites des surfaces des équations 6.9 et 6.10

Néanmoins, les solutions des équations 6.9 et 6.10 définissent des classes d'équivalence au travers des limites entre les différents états du système : "Comfort", "Safety", "Emergency". La figure 6.11 présente les cinq classes d'équivalence déduites des surfaces calculées à partir des équations du système.

Il est ainsi possible de considérer un élément par classes d'équivalence dans l'espace d'états. La projection de ces éléments dans les dimensions impliquées permet de réduire les domaines de couleurs à des ensembles de valeurs très réduits. Sur le modèle de la figure 6.6, nous pouvons conserver uniquement cinq points dans l'espace d'états du système, et ainsi obtenir des domaines de couleurs de *distance*, *velocity* et *braking factor* ne contenant que cinq valeurs.

Cette nouvelle discrétisation n'est pertinente que pour les propriétés d'accessibilité mais reste correcte car toutes les classes d'équivalence possibles sont accessibles. La figure 6.12 montre l'espace d'états réduit du modèle de la figure 6.6.

Contrairement aux classes d'équivalence "classiques" des réseaux de Petri symétriques, ces classes d'équivalence sont calculées à partir de la définition du système et de sa structure, c'est pourquoi nous les appelons *Classes d'Équivalence Sémantique*.

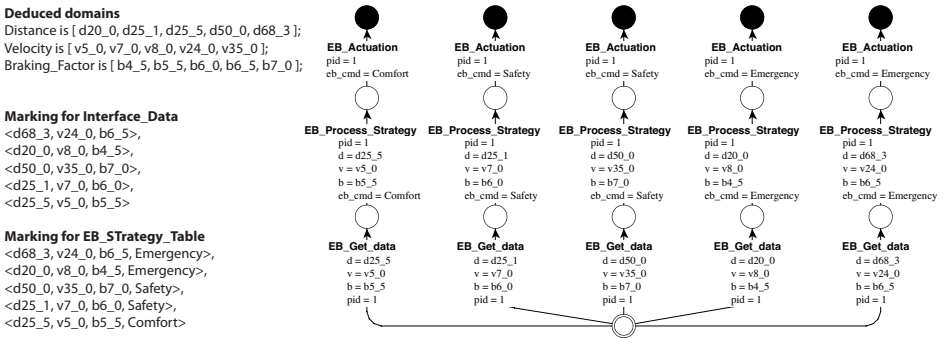


FIGURE 6.12 – Espace d'états réduit calculé à l'aide des classes d'équivalence sémantique

Élaboration d'un *Model Checker* dédié Afin de pallier le problème de l'explosion combinatoire survenant lors du calcul de l'espace d'états du système, nous avons identifié différentes solutions à mettre en place dans un *model checker*.

1. L'utilisation des ordres partiels comme décrit dans [19] qui permet d'éviter l'exploration de chemins redondants.
2. L'identification des places à marquage stable pour améliorer l'utilisation des diagrammes de décision. Typiquement, lorsque PROD calcule l'espace d'états, la place **EB_STrategy_Table** est systématiquement réécrite pour chacun des états du système alors que, étant stable, elle n'apporte pas d'information pertinente pour les distinguer les uns des autres. Si ce type place était identifié et les variables qui les encodent placées au sommet du diagramme de décision (par exemple avec un diagramme de décision binaire : *BDD* [22]), son marquage ne serait représenté qu'une seule fois.
3. L'utilisation de diagrammes de décision adaptés. Certains diagrammes de décision comme les DDD [32], MDD [77] ou SDD [33] ré-encodent les données discrètes (dans le cas des DDD et des MDD) ou prennent en compte la hiérarchie (les SDD) ce qui permet un encodage réduit des espaces d'états. On parle alors de *symbolic model checking* [4]. Ces diagrammes de décision gèrent entres autres l'optimisation précédemment citée (i.e. l'identification des places à marquage stable).
4. L'utilisation des Réseaux de Petri Symétriques avec *bags*. Dans cette extension récente des Réseaux de Petri [76], les jetons contiennent des ensembles de jetons. Cela permet de réduire la complexité du calcul du tirage de certaines transitions comme la transition **EB_Process_Strategy** [55].
5. L'utilisation de *super calculateurs* ou de *grilles de calculateurs* . L'utilisation d'un super calculateur composé de plusieurs processeurs n'a d'intérêt qu'avec un *model checker*

4. à ne pas confondre avec l'appellation *symbolic state space* utilisé dans le cadres des réseaux symétriques

écrit spécifiquement pour tirer parti de cette répartition du calcul. Le logiciel GreatSPN est déjà partiellement ré-écrit pour cet usage [60]. Ainsi, le calcul de l'espace d'états peut être réparti sur plusieurs processeurs.

Ces techniques permettent de résoudre les principaux problèmes liés à l'explosion combinatoire que nous avons rencontrés. Et même si elles n'existent qu'à l'état de prototypes ou ne sont que partiellement implémentées nous sommes convaincus qu'elles simplifieront l'analyse de systèmes discrétisés complexes comme celui que nous avons présenté.

6.5 Perspectives

Dans ce chapitre, nous avons décrit et appliqué une méthode de discrétisation permettant de vérifier formellement les systèmes hybrides en modélisant ses composantes continues. Nous présentons ici les perspectives liées à notre approche.

6.5.1 Optimisation des paramètres de discrétisation

L'algorithme de discrétisation que nous avons choisi repose, comme la plupart des autres algorithmes de discrétisation, sur des paramètres. Ces paramètres peuvent être optimisés pour un ensemble de variables ou fonctions continues données. Le paramètre utilisé ici définit le nombre de valeurs discrètes de chacune des variables de la fonction. Plus ce paramètre est élevé, plus la discrétisation de la fonction est précise.

Mais, dans la fonction de freinage d'urgence par exemple, les variables de la fonction n'ont pas toutes la même influence sur la *vitesse* d'évolution de la fonction.

Il est donc possible d'optimiser la discrétisation en choisissant un paramètre différent pour chacune des variables : un paramètre élevé pour les variables influentes et petit pour les variables peu influentes.

Une manière simple de procéder est de calculer séparément l'influence de chacune des variables sur l'évolution de l'erreur induite par la discrétisation de la fonction, ce qui se calcule à l'aide de dérivées partielles.

Par exemple, la dérivé partielle de la fonction d'erreur sur le seuil *EB_Safety* (équation 6.13) par rapport à la variable v est :

$$\frac{\partial \Delta_{Eb\_Safety}}{\partial v} = \frac{v \pm \Delta_v}{b \pm \Delta_b} - \frac{v}{b} \tag{6.16}$$

En procédant ainsi pour chacune des variables de la fonction, on constate que les variables v et d sont plus influentes que la variable b.

Cela permet de définir des paramètres de discrétisation optimisés pour la fonction étudiée. Le tableau 6.2 présente l'erreur induite par la discrétisation quand les paramètres sont optimisés en utilisant les dérivés partielles [5].

Cette approche donne des résultats d'optimisation différents en fonction de la fonction discrétisée. Nous montrons ici que l'on peut obtenir une réduction de 10% de l'erreur induite par la discrétisation du seuil *EB_Safety* .

5. Δ_{Eb_Saf} et Δ_{Eb_Emerg} signifient Δ_{Eb_Safety} et$\Delta_{Eb_Emergency}$ respectivement.

Valeurs de v, b, d Paramètres	$v = 13m/s, b = 8m/s^{-2}$, $d = 500m$	$v = 36m/s, b = 4m/s^{-2}$, $d = 100m$
$k_v = 114, k_b = 73$ $k_d = 120$ $card(EBData) < 10^6$	$\Delta_{Eb\_Saf} \in [-6.167m, 6.203m]$ $\Delta_{Eb\_Emerg} \in [-5.367m, 5.403m]$	$\Delta_{Eb\_Saf} \in [-12.09m, 12.40m]$ $\Delta_{Eb\_Emerg} \in [-11.29m, 11.60m]$

TABLE 6.2 – Discrétisation avec paramètres optimisés

L'étude des paramètres de discrétisation pour une fonction donnée est une tâche complexe qui permet d'obtenir des résultats intéressants.

6.5.2 Discrétisation par contrainte

Il peut être dans certains cas nécessaire de déterminer les paramètres de discrétisation des variables (ici k_b, k_v et k_d) en fonction d'une erreur maximale tolérée sur la fonction définie à priori.

Prenons par exemple la fonction de calcul de la distance de freinage (6.8) présentée en section 6.2.2. Il est possible de calculer le paramètre de discrétisation, des variables v et b en partant de la précision désirée sur la fonction Δ_f. Soit $\pm\Delta_f$ l'erreur tolérée sur la fonction f, et $\pm\Delta_v$, $\pm\Delta_b$ les erreurs résultantes sur v et b. En utilisant le calcul de l'intervalle d'erreur présenté en section 6.1.3 nous avons :

$$\pm\Delta_f = \frac{(v \pm \Delta_v)^2}{2 * (b \pm \Delta_b)} - \frac{v^2}{2 * b} \tag{6.17}$$

donc [6] :

$$\Delta_b = -\frac{2 * b^2 * \Delta_f - 2 * b * \Delta_v * v - b * \Delta_v^2}{2 * b * \Delta_f + v^2} \tag{6.18}$$

et deux solutions pour Δ_v un peu plus complexes.

Soit v_{min}, v_{max}, b_{min} et b_{max} les intervalles de v et b. La cardinalité des ensembles discrets V et B sont :

$$Card(V) = \frac{v_{max} - v_{min}}{2 * \Delta_v} \qquad Card(B) = \frac{b_{max} - b_{min}}{2 * \Delta_b} \tag{6.19}$$

Si nous désirons par exemple la même cardinalité (le même paramètre de discrétisation) pour les ensembles de couleurs V et B ($k_b = k_v$), nous obtenons [7] :

$$\Delta_v = \frac{(v_{max} - v_{min})\Delta_b}{b_{max} - b_{min}} \tag{6.20}$$

En utilisant la valeur de Δ_v de l'équation (6.20) dans l'équation (6.18), il est possible de déduire Δ_b pour un Δ_f donné.

Pour une fonction à deux variables, cette méthode reste complexe et donne plusieurs solutions qui doivent être étudiées pour choisir celle qui est appropriée. Néanmoins, elle permet le calcul des paramètres de discrétisation en fonction de la précision désirée.

6. Les opérateurs $\pm$ ont été enlevés de l'équation pour plus de lisibilité.
7. Il possible de choisir des paramètres différents comme présenté en section 6.5.1.

6.6 Conclusion

Ce chapitre présente une méthode d'intégration des aspects continus d'une spécification par discrétisation dans un modèle de Réseau de Petri, à des fins de vérification formelle par *model checking*. Elle est présentée dans un contexte concret d'une fonction de freinage d'urgence d'un STI.

Les méthodes de discrétisation dépendent des équations décrivant le problème. Ici, ces équations sont tirées du modèle physique du système. Il est important, dans ce contexte d'utilisation d'une méthode de discrétisation, de pouvoir évaluer la précision des démonstrations de propriétés.

À cette fin, nous calculons une abstraction discrète des équations. La qualité de l'abstraction est évaluée par rapport à la propriété qui doit être vérifiée. C'est un aspect important de la modélisation et de l'évaluation d'un système à l'aide de modèles formels. L'erreur induite par la discrétisation (l'abstraction) peut être réduite en utilisant une discrétisation plus précise. On peut aussi intégrer cette erreur par la modification des formules décrivant les propriétés.

Les équations sont d'abord modélisées dans un canevas en Réseaux de Petri Colorés. La discrétisation permet d'obtenir ensuite un modèle fini. Le réseau est transformé en Réseaux de Petri Symétriques pour permettre la détection des symétries et procéder aux analyses.

Même si la précision des analyses est contrainte par le problème d'explosion combinatoire, il est possible d'analyser les principales propriétés du modèle comme ses bornes et ses propriétés d'accessibilité. Cela pour des domaines de couleurs de tailles raisonnables, ce qui ne serait pas possible avec le modèle continu en Réseau de Petri Coloré.

Dans le contexte du projet SAFESPOT, de nombreux modules s'exécutent en parallèle et peuvent introduire de nouvelles variables continues. L'analyse des contraintes de discrétisation et des erreurs propagées devient alors très complexe.

Notre méthode permet l'utilisation de différents algorithmes de discrétisation. Pour une première approche, nous utilisons un algorithme simple et générique. Mais d'autres algorithmes à intervalle de discrétisation variable dédiés aux fonctions à discrétiser permettent de bonnes optimisations. Néanmoins, ils introduisent de nouvelles contraintes de vérification formelle ou de calcul de propagation d'erreur qui ne sont pas abordées ici.

En conclusion, la notion de classe d'équivalence sémantique tirée des équations et injectée dans le modèle au travers des marquages initiaux est très intéressante. Elle permet de réduire de manière drastique le problème d'explosion combinatoire pour la vérification de propriétés liées à l'accessibilité des états du modèle. L'exploitation de ces informations, quand elle possible, semble l'approche la plus intéressante dans la perspective de futurs travaux autour de cette méthodologie.

Chapitre 7

Conclusion générale et perspectives

La sécurité humaine est de plus en plus confiée aux systèmes de l'information et de la communication. Une attention particulière est donc nécessaire pour assurer la fiabilité de ces systèmes toujours plus complexes.

Les Systèmes de Transport Intelligent pour la route sont un exemple de ces systèmes complexes et critiques jouant un rôle pour la sécurité humaine. Ils nécessitent la collaboration de différents acteurs, chacun spécialisé dans une partie du processus de développement. La communication entre ces différents acteurs est la cause de nombreuses difficultés.

L'approche la plus répandue dans l'industrie est d'utiliser des spécifications semi-formelles pour assurer la communication au sein du projet ainsi que pour spécifier l'implémentation du système. Dans ce cas, les contraintes qui permettraient d'effectuer la vérification formelle de ces spécifications ne sont pas prises en compte. Or, les méthodes formelles sont seules à même de fournir des preuves mathématiques de la fiabilité des spécifications.

Quand les spécifications du système sont directement rédigées dans un langage formel, leur production est complexe, longue et coûteuse. De plus la complexité des modèles produits nuit à la communication nécessaire entre les différentes équipes impliquées dans le projet.

Pour introduire les méthodes formelles sans nuire à la coopération des équipes, et satisfaire à l'exigence de fiabilité, nous proposons de produire les modèles formels à partir des spécifications semi-formelles.

Contributions

Dans le cadre de la spécification puis du développement d'une application de sécurité pour la route, notre travail s'est articulé autour de deux contributions complémentaires :

1. la mise en place d'une **technique de production des cas d'utilisation pour les STI**
2. **l'aide à la vérification des spécifications de systèmes complexes** par *model checking*.

1 - Élaboration des cas d'utilisation pour STI dans un contexte européen

L'élaboration de cas d'utilisation pour un STI de sécurité routière passe par une analyse de l'accidentologie. Or, les sources de données accessibles au niveau européen ou au niveau de chaque pays sont soit hétérogènes, soit insuffisamment détaillées. Aucune méthode n'existe à l'heure actuelle pour prendre en compte ces problématiques.

Dans le chapitre 4, nous proposons un moyen de surmonter cette difficulté en définissant le concept de **scénarios accidentogènes**.

1. En classant les données accidentologiques selon différents paramètres définissant les causes d'un accident (lieu, conditions environnementales, attitude du conducteur etc.), ces scénarios permettent d'homogénéiser les données accidentologiques.
2. Nous utilisons ensuite ces scénarios pour définir les cas d'utilisation du système. Cela nous permet de passer de l'étude accidentologique à la définition des applications.
3. Enfin, nous avons montré comment cette analyse des besoins permet de déterminer la priorité d'implémentation des cas d'utilisation.

Partant des recommandations de la Commission Européenne, cette approche de l'étude accidentologique a prouvé son efficacité en donnant une vision détaillée des dangers que peuvent rencontrer les conducteurs sur les routes européennes. Seul un niveau de détail élevé a permis d'améliorer la conception des applications de SAFESPOT. Notre méthode offre une solution pour l'élaboration des cas d'utilisation pour les STI de sécurité routière à partir de données accidentologiques hétérogènes. Elle a permis de redéfinir les applications initialement envisagées pour SAFESPOT en augmentant le nombre de situations accidentogènes prises en compte.

2 - Vérification formelle des spécifications de systèmes complexes

Les diagrammes UML sont les diagrammes les plus utilisés dans l'industrie pour la spécification de l'architecture et des composants d'un système. Ils sont adaptés à la spécification des systèmes complexes, mais leur analyse formelle nécessite la mise en place d'une méthode de transformation vers des modèles formels (i.e. leur formalisation). Différentes méthodes existent pour formaliser UML mais aucune n'offre une approche modulaire pourtant nécessaire pour la prise en compte des systèmes complexes et leur analyse tout au long du cycle de développement.

Nous montrons, au chapitre 5, comment il est possible de tirer partie des diagrammes UML, pour fournir un cadre à l'élaboration de modèles formels modulaires conformes au système étudié.

1. La transformation des diagrammes UML d'architecture, d'interface et d'activités facilite la production de modèles modulaires en Réseaux de Petri. Cela permet ensuite d'effectuer la vérification formelle de nombreuses propriétés comme celles exprimées en logique temporelle, et ainsi d'améliorer la fiabilité des spécifications.
2. En définissant un patron formel générique de l'architecture d'un système, notre méthode offre une solution pour prendre en charge les systèmes complexes. Elle permet modifier des sous-parties du modèle formel sans avoir à modifier le reste du modèle. Cette approche permet de tester différentes version des composants du système, ou de tester différents cas d'utilisation facilement.

3. L'application de cette méthode au cas d'étude SAFESPOT montre sa faisabilité et sa capacité à détecter des erreurs de conception. Elle propose des solutions pour la modélisation de systèmes composés d'un grand nombre d'éléments en introduisant une approche modulaire inédite, et permet l'analyse des évolutions discrètes du système.

Une partie importante du système est néanmoins régie par un ensemble de phénomènes continus qu'il est important de pouvoir modéliser afin de prouver les propriétés qui y font référence. L'intégration de ces phénomènes continus dans des méthodes formelles comme les automates hybrides est un domaine de recherche très actif, mais peu intégré aux outils de vérification actuels. Il est difficile de produire des modèles analysables dans la mesure ou l'on sait que la décidabilité de ce type d'automate n'est pas décidable dans le cas générale, et il n'existe pas d'outils pour analyser les formules de logique temporelle (LTL ou CTL).

À cela, nous proposons au chapitre 6 une solution basée sur la discrétisation afin de produire des modèles analysables sur lesquels la logique temporelle peut être utilisée.

1. Des Réseaux de Petri Colorés sont utilisés pour représenter les aspects continus du système. Ils sont ensuite discrétisés en Réseaux de Petri Symétriques formellement analysables.
2. Nous traitons les problèmes de propagation d'erreurs lors de la discrétisation des aspects continus. Ces erreurs peuvent ensuite être corrigées à l'intérieur du modèle ou dans la formulation de propriétés à vérifier.
3. Appliquée au projet SAFESPOT, cette méthode démontre sa capacité à effectuer des analyses formelles impliquant des phénomènes continus.

Perspectives

La mise en place des scénarios accidentogènes répond à différents problèmes liés à la définition des cas d'utilisation et à l'analyse de l'accidentologie européenne. La difficulté est de trouver des données détaillées, et de gérer l'hétérogénéité de ces données venant des différents pays.

L'amélioration des bases de données CARE et EACS, ou l'avancement des projets SAFETYNET, RANKERS ou PENDANT seront d'une aide précieuse pour les ingénieurs développant les futures applications de sécurité pour les routes européennes. En effet, leur objectifs sont l'unification des données accidentologiques européennes et la mise en place d'indices permettant d'évaluer la sécurité sur les routes.

La vérification des spécifications que nous avons présentée dans ce mémoire, rencontre principalement un problème d'explosion combinatoire dû à la complexité du système étudié. Afin de pallier le problème de l'explosion combinatoire survenant lors du calcul de l'espace d'état du système, nous avons identifié différentes solutions à mettre en place dans un *model checker*.

1. L'utilisation des réseaux de Petri Symétriques avec *bags* est une perspective intéressante. Dans cette extension des réseaux de Petri [76], les jetons contiennent des ensembles de jetons. Cela permet de réduire la complexité du calcul du tirage des transitions comme celles utilisées dans notre méthode de discrétisation.

2. Grâce à l'encapsulation fournie par les *Instanciable Petri nets* [63], il n'est pas nécessaire de connaître la structure interne d'un composant. Les définitions de types et de types-composites qu'ils introduisent rendent possible la composition hiérarchique. Cela permet de mettre en exergue une certaine régularité dans les modèles. Cette régularité est notamment exploitée par les SDD pour gérer plus efficacement le problème de l'explosion combinatoire rencontré lors du calcul de l'espace d'état.
3. Dans le contexte de la discrétisation des Réseaux de Petri Colorés en Réseaux de Petri Symétriques, les places servant à la modélisation des fonctions continues sont stables mais systématiquement réécrites pour chacun des états du système lors du calcul de l'espace d'état. Alors que, étant stables, elles n'apportent pas d'information pertinente pour distinguer ces états entre eux. Si une telle place est identifiée, elle peut être placée au sommet du diagramme de décision, par exemple avec un diagramme de décision binaire : *BDD* [22], son marquage ne serait représenté qu'une seule fois.
4. Les diagrammes de décision comme les DDD [32], MDD [77] ou SDD [33] ré-encodent les données discrètes (dans le cas des DDD et des MDD) ou prennent en compte la hiérarchie (les SDD) ce qui permet un encodage réduit des espaces d'états. On parle alors de *symbolic model checking* [1].
5. L'utilisation des ordres partiels, comme décrit dans [19], permet quant à elle d'éviter l'exploration de chemin redondants lors des analyses d'accessibilité, ce qui permettrait aussi de limiter les problèmes d'explosion combinatoire.
6. L'utilisation d'un super calculateur composé de plusieurs processeurs n'a d'intérêt qu'avec un *model checker* écrit spécifiquement pour tirer partie de cette répartition du calcul. Le logiciel GreatSPN est déjà partiellement ré-écrit pour cet usage [60]. Ainsi, le calcul de l'espace d'états peut être réparti sur plusieurs processeurs, ce qui permet de gérer des modèles plus complexes.

Ces solutions n'existent qu'à l'état de prototypes ou ne sont que partiellement implémentées. Toutes permettent de réduire le problème de l'explosion combinatoire, principal obstacle liés à la vérification des systèmes complexes. Nous sommes convaincus qu'elles faciliteront l'analyse de ces systèmes.

1. à ne pas confondre avec l'appellation *symbolic state space* utilisée dans le cadre des Réseaux de Petri Symétriques

Bibliographie

[1] Jean-Raymond Abrial. *The B Book - Assigning Programs to Meanings*. Cambridge University Press, August 1996.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid automata : An algorithmic approach to the specication and verication of hybrid systems. *Hybrid Systems, LNCS, Springer*, pages 209–229, 1992.

[3] Association des sociétés françaises d'autoroute. Lethal accident analysis on private motorway. Statistical studies, ASFA, 2004.

[4] S. Baarir, S. Haddad, and J-M. Ilié. Exploiting Partial Symmetries in Well-formed nets for the Reachability and the Linear Time Model Checking Problems. In *Proc. of WODES'04 - IFAC Workshop on Discrete Event Systems, part of 7th CAAP*, Reims - France, 2004. Springer Verlag.

[5] Floréal Le Bar and Yves Page. Les sorties de voies involontaires. Technical report, ARCOS, CEESAR LAB, 2002.

[6] Christine Bartels. Safespot local dynamic maps - virtual worlds for safety applications. In *14th World Congress and Exhibition on Intelligent Transport Systems and Services*, 2007.

[7] E.G.H.J. Bastiaensen and A. De Hoog. Automated Vehicle Guidance with ADA ; Technology Perspectives for safety and infrastructure utilisation, February 2001.

[8] Simona Bernardi, Susanna Donatelli, and José Merseguer. From UML sequence diagrams and statecharts to analysable petri net models. In *WOSP '02 : Proceedings of the 3rd international workshop on Software and performance*, pages 35–45, New York, NY, USA, 2002. ACM Press.

[9] Daniel M. Berry. The inevitale pain of software development : Why there is no silver bullet. *RISSEF 2002*, LNCS 2941 :50–74, 2004.

[10] G. Berthelot. Transformations and decompositions of nets. In *Advances in Petri Nets*, LNCS 254, pages 359–376. Springer Verlag, 1986.

[11] Jonathan Billington, Guy Edward Gallasch, and Laure Petrucci. Verification of the class of stop-and-wait protocols modelled by coloured Petri nets. *Nord. J. Comput.*, 12(3) :251–274, 2005.

[12] Catherine Bonari and Christophe Desnouailles. Calm : Un bouquet de normes relatif aux communications véhicules-infrastructures. *T.E.C. ISSN 0397-6513*, 195 :47–51, 2007.

[13] F. Bonnefoi, L. Hillah, F. Kordon, and X. Renault. Design, modeling and analysis of ITS using UML and Petri Nets. In *10th International IEEE Conference on Intelligent Transportation Systems (ITSC'07)*, pages 314–319. IEEE Press, 2007.

[14] Fabien Bonnefoi, Francesco Bellotti, Tobias Schendzielorz, and Filippo Visintainer. SAFESPOT Applications for Infrasructure-based Co-operative Road Safety. In *14th World Congress on Intelligent Transport Systems*, 2007.

[15] Fabien Bonnefoi, Tobias Schendzielorz, et al. D5.2.4 Accident data review and potential impact of each function. Deliverable, SAFESPOT by Cofiroute and Technical University of Munich, et al., 2007.

[16] B. Bordbar, L. Giacomini, and D.J. Holding. UML and Petri nets for design and analysis of distributed systems. In *IEEE Conference on Control Applications*, pages 610–615, 2000.

[17] Jonathan P. Bowen and Michael G Hinchey. Seven more myths of formal methods. *IEEE Software*, 12 :34–41, 1995.

[18] Achim Brakemeier, Michele Provera, Giuliana Zennaro, Tim Edwards, Markus Shawky, Dzmitry Kliazovich, Fabrizio Granelli, et al. D3.3.4 Vehicular Ad Hoc Networks. Technical report, SAFESPOT, 2007.

[19] Robert Brgan and Denis Poitrenaud. An efficient algorithm for the computation of stubborn sets of well formed Petri nets. In Giorgio De Michelis and Michel Diaz, editors, *Application and Theory of Petri Nets*, LNCS 935, pages 121–140. Springer, 1995.

[20] Roberto Brignolo. Co-operative road safety - the SAFESPOT integrated project. In *APSN - APROSYS Conference*. Advanced Passive Safety Network, May 2006.

[21] Seth Bullock and Dave Cliff. Complexity and emergent behaviour in ICT Systems. Technical report, Digital Media Systems Laboratory, 2004.

[22] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking : 10^{20} states and beyond. *Inf. Comput.*, 98(2) :142–170, 1992.

[23] R. Brown C. Covault and D. Driscoll. *Uncertainties and Error Propagation - Appendix V of Physics Lab Manual*. Case Western Reserve University, 2005.

[24] J. Campos and J. Merseguer. On the integration of UML and Petri nets in software development. In S. Donatelli and P.S. Thiagarajan, editors, *27th ICATPN - Petri Nets and other models of concurrency*, volume 4024, pages 19–36. Springer-Verlag Berlin Heidelberg, June 2006.

[25] Robert N. Charette. Why software fails. Technical report, Spectrum - IEEE, 2009.

[26] G. Chiola, C. Dutheillet, G. Franceschini, and S. Haddad. On Well-Formed Coloured Nets and their Symbolic Reachability Graph. *High-Level Petri Nets. Theory and Application, LNCS*, 1991.

[27] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. A symbolic reachability graph for coloured Petri nets. *Theoretical Computer Science*, 176(1–2) :39–65, 1997.

[28] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7 : Graphical Editor and Analyzer for Timed and Stochastic Petri Nets Performance Evaluation. *special issue on Performance Modeling Tools*, 24((1&2)) :47–68, November 1995.

[29] Giovanni Chiola, Claude Dutheillet, Giuliana Franceschinis, and Serge Haddad. Stochastic well-formed colored nets and symmetric modeling applications. *IEEE Trans. Computers*, 42(11) :1343–1360, 1993.

[30] C. Choppy and G. Reggio. Improving use case based requirements using formally grounded specifications. *FASE 2004. M. Wermelinger and T. Margaria-Steffen, Eds. LNCS*, 2984 :244–261, 2004.

[31] P. Christofides and N. El-Farra. *Control Nonlinear and Hybrid Process Systems : Designs for Uncertainty, Constraints and Time-delays*. Springer Verlag, 2005.

[32] J.-M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud, and P.-A. Wacrenier. Data decision diagrams for Petri net analysis. In *Proc. ICATPN'2002*, LNCS 2360, pages 101–120. Springer Verlag, June 2002.

[33] J-M. Couvreur and Y. Thierry-Mieg. Hierarchical Decision Diagrams to Exploit Model Structure. In *Proc. FORTE'05*, LNCS 3731, pages 443–457. Springer, 2005.

[34] The CPN Tools Homepage, 2007. `http://www.daimi.au.dk/CPNtools`.

[35] Andrea D'Ambrogio. A model transformation framework for the automated building of performance models from UML models. In *WOSP '05 : Proceedings of the 5th international workshop on Software and performance*, pages 75–86, New York, NY, USA, 2005. ACM Press.

[36] J. Daniel and D. Luca. IEEE 802.11p : Towards an International Standard for Wireless Access in Vehicular Environments. In *Proceedings of Vehicular Technology Conference,VTC Spring, IEEE*, pages 2036–2040, May 2008.

[37] S. Dashtinezhad, T. Nadeem, B. Dorohonceanu, C. Borcea, P. Kang, and L. Iftode. TrafficView : A Driver Assistant Device for Traffic Monitoring based on Car-to-Car Communication. In IEEE Computer Press, editor, *IEEE Semiannual Vehicular Technology Conference*, 2004.

[38] René David and Hassane Alla. On Hybrid Petri Nets. *Discrete Event Dynamic Systems : Theory and Applications*, 11(1-2) :9–40, 2001.

[39] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Int. Conf. on Machine Learning, pp 194-202*, 1995.

[40] C. Dutheillet. *Symétrie dans les réseaux colorés : définition, analyse et application à l'évaluation de performance*. PhD thesis, Université Pierre et Marie Curie, Paris, 1992.

[41] A. Emerson E. Clarke and J. Sifakis. Model checking : algorithmic verification and debugging. *Commun. ACM 52*, 74-84, Nov. 2009.

[42] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1) :1–24, 1985.

[43] European Automobile manufacturers Association (ACEA) and European Commission. EACS, European Accident Causation Survey : accurate information on the causes of road accidentsin europe, 2004.

[44] European Commission. European energy and transport - trends to 2030. *Luxembourg : Office for Official Publications of the European Communities*, ISBN 92-894-4444-4, 2003.

[45] European Council. CARE : a community database on road accidents resulting in death or injury (no statistics on damage - only accidents). Council Decision 93/704/EC, Oj No L329 of 30.12.1993, pp. 63-65, 1993-2008.

[46] ETSC European Transport Safety Council. Transport accident and incident investigation in the european union, http ://www.etsc.be/accinv.pdf. Statistical studies, European Transport Safety Council, 2001.

[47] Federal Highway Research Institute (Bundesanstalt für Straßenwesen). www.bast.de. Statistical studies, BAST, 2007.

[48] Federal Statistical Office Germany. Federal Statistical Surveys, http ://www.destatis.de/. Statistical studies, Federal Statistical Office Germany, 2004.

[49] Robert France and Bernhard Rumpe. Model-driven development of complex software : A research roadmap. In *FOSE '07 : 2007 Future of Software Engineering*, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society.

[50] Schermers G. Advanced Driver Assistance (ADA) systemen ; Perspectief verkeersveiligheid. Technical report, AVV Transport Research Centre, Rotterdam, 2000.

[51] C. Girault and R. Valk. *Petri Nets for Systems Engineering : A Guide to Modeling, Verification and Applications*. Springer, 2003.

[52] Robert L. Glass. The standish report : Does it really describe a software crisis ? In *Communications of the ACM*, page p. 15, 2006.

[53] J. Gogen and Luqi. Formal methods : Promises and problems. *IEEE Software*, 14(1) :75–85, 1997.

[54] S. Haddad. A reduction theory for coloured nets. In *Advances in Petri Nets 1989*, LNCS 424, pages 209–235. Springer-Verlag, 1990.

[55] S. Haddad, F. Kordon, L. Petrucci, J-F. Pradat-Peyre, and N. Trèves. Efficient State-Based Analysis by Introducing Bags in Petri Net Color Domains. In *28th American Control Conference (ACC'09)*, St-Louis, USA, June 2009. IEEE.

[56] Serge Haddad and Jean-François Pradat-Peyre. New efficient Petri nets reductions for parallel programs verification. *Parallel Processing Letters*, 16(1) :101–116, March 2006.

[57] B. Hailpern and P. Tarr. Model-driven development : The good, the bad and the ugly. *IBM Systems Journal*, 45(3) :451, 2006.

[58] Anthony Hall. Seven myths of formal methods. *IEEE Softw.*, 7(5) :11–19, 1990.

[59] A. Hamez, L. Hillah, F. Kordon, A. Linard, E. Paviot-Adet, X. Renault, and Y. Thierry-Mieg. New features in CPN-AMI 3 : focusing on the analysis of complex distributed systems. In *6^{th} International Conference on Application of Concurrency to System Design (ACSD'06)*, Turku, Finland, June to be published in 2006. IEEE Computer Society.

[60] A. Hamez, F. Kordon, Y. Thierry-Mieg, and F. Legond-Aubry. dmcG : a distributed symbolic model checker based on GreatSPN. In *28th International Conference on Petri Nets and Other Models of Concurrency (ICATPN'07)*, LNCS 4546, pages 495–504. Springer, 2007.

[61] A. Hamez and X. Renault. *PetriScript Reference Manual*. LIP6, www-src.lip6.fr/logiciels/mars/CPNAMI/MANUAL_SERV.

[62] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata ? In *In Proc. of the 27th ACM Symp. on the Theory of Computing (STOCS '95),*, pages 373– 382, 1995.

[63] Lom Messan Hillah. *Intégration des méthodes formelles au développement dirigé par les modèles, pour la conception et la vérification des systèmes et applications répartis*. PhD thesis, Université Pierre et Marie Curie - Laboratoire d'informatique de Paris 6, 2010.

[64] Eric Honour. *Understanding the Value of Systems Engineering*. The International Council on Systems Engineering (INCOSE), 2004.

[65] J. Hugues, Y. Thierry-Mieg, F. Kordon, L. Pautet, S. Baarir, and T. Vergnaud. On the Formal Verification of Middleware Behavioral Properties. In *Proceedings of the 9th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'04)*, volume ENTCS 133, pages 139 – 157. Elsevier, 2004.

[66] IEEE 802.11 Working Group for WLAN Standards. *IEEE 802.11 tm Wireless Local Area Networks*. IEEE, 2008.

[67] FP7 Cooperation Work Programme : Information and Communication Technologies. Updated Work Programme 2009 And Work Programme 2010 - Theme 3 ICT – Information and Communications Technologies. Technical Report C(2009) 5893, European Commission, July 2009.

[68] SAFESPOT Integrated Project Consortium. Annex I : "Description of Work". Technical Annexs, Sixth Framework Programme – Information Society Technologies, 2006.

[69] ISO. Specifications of Abstract Syntax Notation One (ASN-1). ISO 8824, 1987.

[70] Italian Institute of Statistics ISTAT. Incidenti stradali – anno 2003. Technical report, International salon of road safety, 2004.

[71] M.A. Jackson. The Role of Architecture in Requirements Engineering. In *First internationnal conference on requirements engineering*. IEEE Computer Society, Colorado Springs, 1994.

[72] Ivar Jacobson. *Object-Oriented Software Engineering : A Use Case Driven Approach*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[73] S.T.M.C. Janssen. Safety on urban through-road intersections ; comparison of crash rates. Technical Report R-2003-36, Institute for Road Safety Research, SWOV, 2004.

[74] Kurt Jensen. *Basic Concepts, Analysis Methods and Practical Use*, volume 3, Practical use ; ISBN 3-540-62867-3. Springer Verlag, 1997.

[75] Peter H Jesty and Richard A P Bossom. Using the FRAME Architecture for Planning Integrated Intelligent Transport Systems. In *Euro American Conference On Tellematics And Information Systems EATIS*, 2009.

[76] Tommi Junttila. *On the symmetry reduction method for Petri Nets and similar formalisms*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2003.

[77] T. Kam, T. Villa, R. Brayton, and A. L. Sangiovanni-Vincentelli. Multi-Valued Decision Diagrams : Theory and Applications. *International Journal on Multiple-Valued Logic*, 4(1-2) :9–62, 1998.

[78] F. Kordon, A. Linard, and E. Paviot-Adet. Optimized Colored Nets Unfolding. In *FORTE'06*, LNCS 4229, pages 339–355. Springer Verlag, 2006.

[79] Fabrice Kordon, Jérôme Hugues, and Xavier Renault. From model driven engineering to verification driven engineering. In *SEUS '08 : Proceedings of the 6th IFIP WG 10.2 international workshop on Software Technologies for Embedded and Ubiquitous Systems*, pages 381–393, Berlin, Heidelberg, 2008. Springer-Verlag.

[80] Commission Économique Pour l'Euope. Statistiques des accidents de la circulation routière en europe et en amérique du nord. Technical report, United Nations, 2007.

[81] Nancy Levenson. Software Engineering : Stretching the Limits of Complexity . *Communication of the ACM*, 40(2) :275–282, 1997.

[82] Vern Lindberg. *Uncertainties and Error Propagation - Part I of a manual on Uncertainties, Graphing, and the Vernier Caliper*. Rochester Inst. of Technology, 2000.

[83] LIP6-MoVe. *The CPN-AMI Home page.* http ://www.lip6.fr/cpn-ami.

[84] Juan Pablo Lopez-Grao, José Merseguer, and Javier Campos. From UML activity diagrams to Stochastic Petri nets : application to software performance engineering. In *WOSP '04 : Proceedings of the 4th international workshop on Software and performance*, pages 25–36, New York, NY, USA, 2004. ACM Press.

[85] Paul Mathias, Fabien Bonnefoi, Tobias Schendzielorz, et al. D5.2.1 Definition of use case and user. Deliverable, SAFESPOT by Siemens, Cofiroute et al., 2006.

[86] Steve McConnell. *Code Complete, Second Edition*. Microsoft Press, Redmond, WA, USA, 2004.

[87] The Royal Academy of Engineering and The British Computer Society. The challenges of complex it projects. Technical report, The Royal Academy of Engineering and The British Computer Society, 2004.

[88] OMG. *Unified Modeling Language : Superstructure - Version 2.0 formal/05-07-04*. OMG, March 2006.

[89] OMG. *Unified Modeling Language : Superstructure - Version 2.2 formal/09-02-03*. OMG, February 2009.

[90] D.L. Parnas and P.C. Clements. A Rational Design Process : How and Why to Fake It. In *IEEE Transactions on Software Engineering SE-12 :2*, pages 196–257, 1986.

[91] Simo Pasi. Global economic crisis hits european road freight transport in the fourth quarter of 2008. Technical report, Eurostat, 2009.

[92] Petri Net Steering Commitee. Petri Net Tool Database : Quick and up-to-date overview of existing tools for Petri Nets, http ://www.informatik.uni-hamburg.de/tgi/petrinets/tools/db.html.

[93] A. Pnueli. A temporal logic of concurrent programs. *Theoretical Computer Science*, 13 :45-60, 1981.

[94] Pressman, R S. *Software engineering : a practioner's approach (2nd. edition)*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[95] COOPERS Project. COOPERS : Co-operative Systems for Intelligent Road Safety. Research and Development project, European Commission Information Society and Media – Sixth PCRD – http ://www.cvisproject.org/, 2006.

[96] CVIS Project. Co-operative Vehicle-Infrastructure Systems. Research and Development project, European Commission Information Society and Media – Sixth PCRD – http ://www.cvisproject.org/, 2006.

[97] E-FRAME project. FRAME : The European ITS Framework Architecture. Support Action project, European Commission Information Society and Media – http ://www.frame-online.net/, 2008.

[98] PENDANT Project. Pendant Pan-European Co-ordinated Accident and Injury Database. Research and Development project, European Commission Information Society and Media – http ://www.vsi.tugraz.at/pendant/, 2003-2005.

[99] SAFESPOT Project. SAFESPOT : Cooperative Vehicles and Road Infrastructure for Road Safety. Research and Development project, European Commission Information Society and Media – Sixth PCRD – http ://www.safespot-eu.org/, 2006.

[100] The ASSESS project. Impact assessment of the communication "Keep Europe Moving" Sustainable mobility for our continent ; a Mid-term review of the European Commission's 2001 Transport White Paper. Commission Staff Working Document, 2006.

[101] A. Quimby, G. Maycock, C. Palmer, and Grayson. G.B. Drivers' speed choice : an in-depth study. Technical Report TRL 326, Transport Research Laboratory, Crowthorne, UK, 1999.

[102] Brian Randell. The 1968/69 nato software engineering reports. Technical report, University of Newcastle upon Tyne, 1969.

[103] RANKERS Project. RANKERS : Europe's most comprehensive research initiative to date in the field of road safety engineering. Research and Development project, European Commission Information Society and Media – http ://www.rankers-project.com/, 2007.

[104] W. W. Royce. Managing the Development of Large Software Systems : Concepts and Techniques. In *WesCon*, 1970.

[105] S. W. Amber, Agile Software Development : The Official Agile Modeling (AM) Site. http ://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm, 2001.

[106] Tobias Schendzielorz, Fabien Bonnefoi, et al. D5.2.3 Application Scenarios and System Requirements. Technical report, SAFESPOT by Technical University of Munich, Cofiroute et al., 2007.

[107] B. Selic. The pragmatics of model-driven development. *IEEE Software 20(5)*, pages 19–25, 2003.

[108] Sénat français. Proposition pour un Schéma Directeur National des Autoroutes de liaison. Proposition de loi relative au financement du schéma directeur national des autoroutes. no 359, Sénat, Chambre haute du Parlement français., 2000.

[109] C. Snook and M. Butler. UML-B : Formal Modeling and Design Aided by UML. In *ACM Transaction on Sofware Engineering and Methodology*, volume 15, pages 92–122, January 2006.

[110] S.A.E. Society of Automotive Engineers. The SAE Architecture Analysis & Design Language standard (AS5506). http ://www.sae.org, Septembre 2004.

[111] Angela Spence, Abdel Kader Mokaddem, Gino Franco, Sergio Sapienza, et al. D7.3.2 SAFESPOT Architecture Building Guide. Deliverable, SAFESPOT by Mizar, Renault et al., 2008.

[112] Dominic Stead. Mid-term review of the european commission's 2001 transport white paper. *European Journal of Transport and Infrastructure Research*, EJTIR, 6(no. 4) :pp. 365–370, 2006.

[113] Douglas Steedman. Abstract Syntax Notation One : ASN.1. The Tutorial and Reference. *British Library Catalogue in Publication Data*, 1990.

[114] Matthias Strauss, Ulrich Staehlin, et al. D4.2.2 Safety Margin Concept. Technical report, SAFESPOT by Continental Teves AG and CO OHG et al., 2007.

[115] Toroyan T. and Peden M. (editeur). Les jeunes et la sécurité routière. Technical Report ISBN 978 92 4 2595116, NLM WA275, Organiation Mondiale de la Santé (OMS), Genève, 2007.

[116] Jean-Pierre Taroux and Jean-Noël Chapulut. Évaluation des systèmes d'exploitation de la route en milieu urbain. Technical Report 2002-0180-01, Conseil Général des Ponts et Chaussées, 2004.

[117] Y. Thierry-Mieg, C. Dutheillet, and I. Mounier. Automatic symmetry detection in well-formed nets. In W. M. P. van der Aalst and E. Best, editors, *24th International Conference on Applications and Theory of Petri Nets 2003*, volume 2679 of *LNCS*, pages 82–101. Springer Verlag, 2003.

[118] J. Trujillo. A Report on the First International Workshop on best practices of UML (BP-UML'05). In *SIGMOD Record*, volume 35/3, September 2006.

[119] U.S. Department of Transportation. National ITS Architecture : a common framework for planning, defining, and integrating intelligent transportation systems, version 6.1 http ://www.iteris.com/itsarch/, March 2009.

[120] U.S. Department of Transportation, Federal Highway Administration, and Federal Transit Aministration. *Systems Engineering for Intelligent Transportation Systems - An Introduction For Transportation Professionals*. Number FHWA-HOP-07-069 in EDL-14340. Department of Transportation, Office of Operations, January 2007.

[121] U.S. Department of Transportation, Federal Highway Administration, and Federal Transit Aministration. Systems Engineering for Intelligent Transportation Systems : An Introduction for Transportation Professionals, 2007.

[122] A. Valmari. The State Explosion Problem. In *Lectures on Petri Nets I : Basic Models*, number 1491 in Lecture Notes in Computer Science, pages 429–528. Springer-Verlag, 1998.

[123] K. Varpaaniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo. Prod reference manual. Technical report, Helsinki University of Technology, 1995.

[124] Filippo Visintainer, Francesco Bellotti, Fabien Bonnefoi, and Tobias Schendzielorz. Infrastructure-based co-operative architectures : How safespot deals with different road network areas. In *14th World Congress on Intelligent Transport Systems*, 2007.

[125] Belgisch Instituut voor de Verkeersveiligheid. www.bivv.be. Statistical studies, Minister van Mobiliteit en Vervoer en van de Federale Overheidsdienst Mobiliteit en Vervoer, 2007.

[126] Jos Warmer and Anneke Kleppe. *The Object Constraint Language : Getting Your Models Ready for MDA*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[127] Silvia Zangherati, Roberto Brignolo, Giuliana Zennaro, et al. SF SP7 Data Format and Messages. Technical report, SAFESPOT by Centro Ricerche FIAT et al., 2009.

Annexe A

Annexes

A.1 Tests d'intégration de l'application H&IW

Cette annexe présente les résultats des tests d'intégration de l'application H&IW. Ces tests ont été effectués sur une piste d'essais où le système SAFESPOT a été déployé.

FIGURE A.1 – Site de test : la piste d'essais du LIVIC

A.1.1 Objectifs

L'objectif des tests d'intégration est de valider l'implémentation de l'application H&IW dans un environnement semblable à celui de son utilisation en situation réelle. Ces tests nécessitent donc un déploiement du système SAFESPOT sur une piste d'essais, et la reproduction des cas d'utilisation de l'application H&IW.

Ces tests permettent aussi, de valider les hypothèses portant sur le système et l'application H&IW (i.e. les contraintes de fonctionnement) qui ont été utilisées lors des différentes phases du cycle de développement.

Dans cette section nous détaillons les objectifs principaux et secondaires de ces tests et leur place dans le cycle de développement de SAFESPOT.

Intégration dans le cycle de développement

Suite aux phases de conception, spécification et vérification formelle du système et de ses modules, vient la phase de tests, validation et vérification. Cette phase de tests se décompose en trois étapes principales :

– **Les tests en laboratoire** permettent de valider la majorité des choix techniques en testant les modules séparément et dans un environnement fonctionnel restreint. Les comportements des différents composants de l'environnement fonctionnel du module à vérifier (capteurs, modules ou utilisateurs) sont en partie simulés.
– **Les tests de validation sur piste (intégration)** permettent de valider l'application dans un environnement semblable à celui de son utilisation en situation réelle en procédant au déploiement d'un maximum de logiciels et matériels composant le système SAFESPOT. C'est ce que nous présentons dans cette annexe.
– **Les tests d'évaluation** ont pour objectifs d'évaluer le système SAFESPOT et son impact sur la sécurité dans une situation de trafic réelle. Le système SAFESPOT est d'une part testé sur des tronçons de route représentatives de la diversité du réseau et conformes aux cas d'utilisation. D'autre part, le système est testé par un panel d'utilisateurs. Finalement, l'impact potentiel en terme de sécurité est évalué grâce aux résultats des tests et à leur mise en relation avec l'étude accidentologique (présentée au chapitre **??**).

Ces différentes étapes permettent de tester le système en procédant à un déploiement progressif. Les protocoles expérimentaux de ces différentes étapes étant progressivement plus complexes et coûteux à mettre en place, il est important de tirer un maximum de résultats de chacune des étapes avant de passer à la suivante. Les tests et résultats obtenus au cours des ces trois étapes sont différents mais se rapportent toujours aux objectifs et contraintes de fonctionnements élaborés dans les différentes phases du cycle de développement du système.

L'étape de validation que nous présentons dans cette annexe, s'effectue en testant les prototypes d'implémentation afin de valider les contraintes de fonctionnement du système, ce qui est présenté figure A.2.

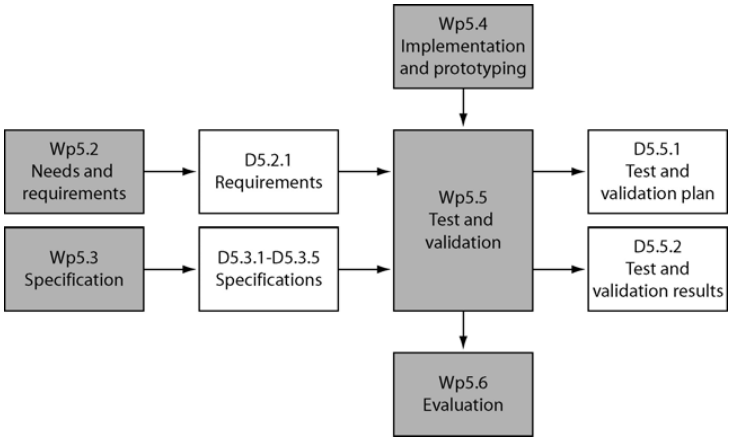


FIGURE A.2 – Place de l'étape de validation dans le cycle de développement du système SAFESPOT

A.1.2 Protocole expérimental

Nous présentons dans cette section le protocole expérimental utilisé pour la validation de l'application H&IW.

Méthode : Scénario de test et méthode de mesure

Scénarios de tests

L'application H&IW est définie par des cas d'utilisation qui présentent le comportement attendu de l'application en situation réelle (présentés au chapitre 4 page 81). C'est a partir de ces cas d'utilisation que sont élaborés les procédures de validation.

Les procédures de validation doivent donc décrire :

– comment seront effectués les tests,
– le matériel nécessaire,
– les cas d'utilisation concernés,
– ce qui sera mesuré et comment,
– la liste des contraintes et exigences de fonctionnement concernées par les tests.

Par exemple, pour les cas d'utilisation "SP5_UC16 : Déviation for road work", les tests doivent mettre en scène un véhicule circulant sur une route contenant un chantier. Le véhicule doit passer plusieurs fois à proximité du chantier, sur la voie fermée par le chantier et sur la voie non fermée. Un schéma est fourni figure A.3.

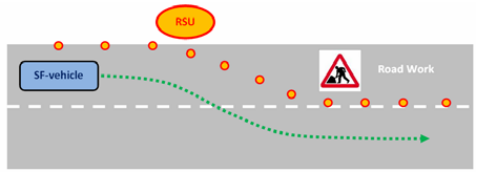


FIGURE A.3 – Schéma de description des tests

Puis, un récapitulatif du comportement attendu de l'application est donné. En l'occurrence, pour l'application H&IW, il s'agit de la description de la stratégie d'alerte. La figure A.4 montre comment sont calculées les zones de destination des alertes et leur modes de routage pour un véhicule circulant à 90km/h pour le cas d'utilisation SP5_UC16.

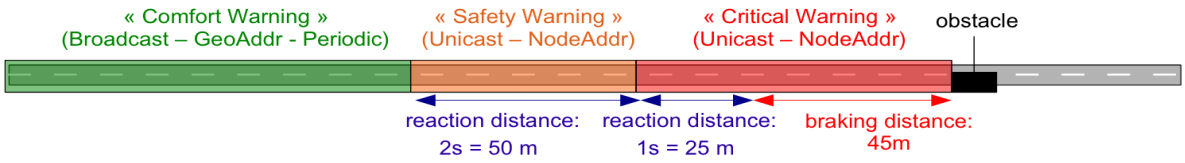


FIGURE A.4 – Stratégie d'alerte du cas d'utilisation SP5_UC16 à 90km/h

Enfin, un tableau récapitulatif des tests à effectuer et des mesures attendues est fourni. Il est présenté figure A.5.

Méthode de mesure

Pour procéder à la validation de l'application, un certain nombre de mesure doivent être effectuées.

DESCRIPTION		PARAMETERS				EXPECTED MESURES				
		VEHICLE SPEED		ROAD WORK		Comfort	Safety		Critical	
id	description	km/h	brespeed adapt.	speed limitation		distance	time t	dist. to obs	time	dist. to ob
1.1	Obstacle Warning t	20	7	30km/h	8,33	under rsu cov	never	never	never	never
1.2	Obstacle Warning t	40	7	30km/h	8,33	under rsu cov	3	42,15	1	19,93
1.3	Obstacle Warning t	50	7	30km/h	8,33	under rsu cov	3	55,45	1	27,67
2.1	Obstacle Warning t	40	7	50km/h	13,89	under rsu cov	3	never	never	never
2.2	Obstacle Warning t	50	7	50km/h	13,89	under rsu cov	3	55,45	1	27,67
2.3	Obstacle Warning t	70	7	50km/h	13,89	under rsu cov	3	85,34	1	46,45
2.4	Obstacle Warning t	90	7	50km/h	13,89	under rsu cov	3	119,64	1	69,64
2.5	Obstacle Warning t	110	7	50km/h	13,89	under rsu cov	3	158,36	1	97,24
3.1	Ability to stop	70	7 breaks at Safety	50km/h	13,89	under rsu cov	3	85,34	never	never
3.2	Ability to stop	70	7 breaks at Critica	50km/h	13,89	under rsu cov	3	85,34	1	46,45
3.3	Ability to stop	90	7 breaks at Safety	50km/h	13,89	under rsu cov	3	119,64	never	never
3.4	Ability to stop	90	7 breaks at Critica	50km/h	13,89	under rsu cov	3	119,64	1	69,64
4.1	Ability to slow down	70	7 Safe. down to 40	50km/h	13,89	under rsu cov	3	66,05	never	never
4.2	Ability to slow down	70	7 Critic. down to 4	50km/h	13,89	under rsu cov	3	66,05	1	27,16
4.3	Ability to slow down	90	7 Safe. down to 40	50km/h	13,89	under rsu cov	3	94,84	never	never
4.4	Ability to slow down	90	7 Critic. down to 4	50km/h	13,89	under rsu cov	3	94,84	1	44,84

FIGURE A.5 – Paramètres et valeurs attendues des tests

– D'abord à l'aide des traces (logs) produites par l'application et les autres composants du système. Pour l'application H&IW nous avons choisi, les traces produites par l'application et celles produites par le composant réseau (le "VANET" présenté).
– Ensuite à l'aide d'un observateur extérieur au système. En effet, il est important de considérer le système testé comme pouvant contenir des erreurs, même dans ces traces, que les capteurs peuvent être défaillants, et que le conducteur peut avoir un comportement différent de celui attendu. Sur la figure A.6, la partie gauche montre ce qui est mesuré par un observateur extérieur au système. Pour effectuer ces mesures, un ensemble de points de repères sont disposés sur la piste d'essais. La figure A.7 montre ces points de repère, disposés tout les dix mètres sur la piste de test.

La figure A.6 présente, à la manière d'un diagramme de séquence UML, un exemple d'échange d'informations entre un véhicule et l'infrastructure déclenchant une alerte. Les flèches en rouge montrent les mesures obtenues dans les "traces" du système.

Matériel et logiciels

Un ensemble de matériel et de logiciels est nécessaire au déploiement du système SAFESPOT pour ces tests. Le tableau A.8 présente l'ensemble des logiciels déployés lors des tests (le terme RSU signifie Road Side Unit et désigne l'infrastructure, le terme OBU signifie On Board Unit et désigne le véhicule).

Les figures A.9 et A.10 présentent respectivement la borne de l'infrastructure et le véhicule utilisé lors des tests.

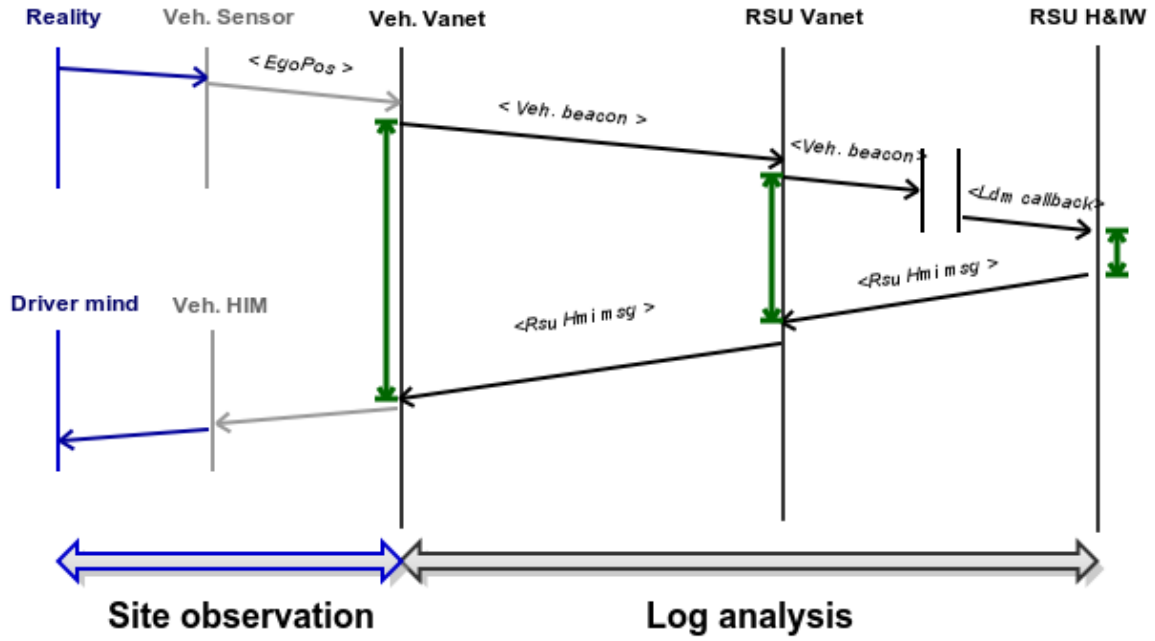


FIGURE A.6 – Méthode de mesure

FIGURE A.7 – Méthode de mesure : les marquages visuels

Référence RSU	Test RSU	Reference OBU	Test OBU
GATEWAY	Cofiroute Gateway	Pos Sensor / SP1 framework	Livic sensor / Cofi EgoPosGenerator
DF	FULL *DataFusion v 1.0.10*	DF	Cofiroute Test Viewer
LDM	FULL *LDM server v 11.7.1* *LDM client v 17.1* *Ldm Schema v24* *Map FR04-03.sql*	LDM	FULL
VANET	FULL cvis-image-20090506 sntrouter-0.9-985.deb	VANET	FULL cvis-image-20090506 sntrouter-0.9-985.deb
APP	Partial : H&IW_01	APP	Not required
APPCOORD	Integrated in H&IW_01	APP Client	Not required
IHM/PMV	Integrated in H&IW_01	IHM	Not required

FIGURE A.8 – Logiciels déployés lors des tests

FIGURE A.9 – La borne de l'infrastructure

FIGURE A.10 – Le véhicule équipé

A.1.3 Analyse des résultats

La session de tests préliminaires dont nous présentons les résultats partiels ici, avait plusieurs objectifs : tester le déploiement des modules du système, et tester l'applications H&IW. Le temps nécessaire au déploiement du système et du matériel expérimental, puis les tests du système déployé, n'ont laissé qu'un temps limité pour procéder aux premiers tests sur piste de l'application H&IW.

Par exemple, il n'a pas été possible d'effectuer plus de deux passages pour une même vitesse, et il faut donc interpréter les mesures fournies avec une certaine précaution. Néanmoins, les résultats obtenus sont cohérents avec ce qui était attendu, et permettent donc une première validation partielle de l'application H&IW.

Mesures préliminaires de l'environnement de l'application :

Pour permettre une analyse pertinente des résultats de tests de l'application H&IW, il est important de bien connaître le système à l'intérieur duquel elle est implémentée. Un certain nombre de tests préliminaires ont donc été effectués sur le système.

Ces tests du système concernent principalement :

- la précision des capteurs et leur fréquence d'échantillonnage.
- performance du réseau en terme de portée, taux de perte de message, débit, vitesse.
- les temps de traitement des différents composants du système nécessaires aux tests.

Si la majorité des mesures effectuées étaient concordantes avec ce qui était attendu, l'analyse des traces et des mesures effectuées par un observateur extérieur a révélé quelques comportements inattendus :

- La portée des communications n'était pas celle escomptée (en moyenne 450 mètres de rayon pour environ 800 espérés) avec un taux de pertes de message de plus de 2% . Après différent tests et analyses, plusieurs facteurs ont été identifiés pour expliquer ces résultats :
  - il est apparu qu'un des câbles reliant les antennes aux cartes réseaux, avait un mauvais blindage, et dissipait une bonne partie de la puissance émise par la carte.

  - ensuite, les antennes choisies avaient plusieurs défauts : en concentrant la puissance sur un angle d'ouverture faible pour obtenir une bonne porté (80% de l'énergie est concentrée dans un angle d'ouverture de 10°) elles sont d'autant plus sensibles aux écarts de fréquence. Or, les cartes 802.11p émettent dans la bande des 5.9GHz entre 5.875GHz et 5.925GHz alors que les antennes choisies étaient adaptées à la bande du 802.11a entre 5.15 et 5.875 GHz. Ce léger écart de fréquence ne pose en général pas de problème significatif sauf quand la puissance du signal est concentré dans un angle d'ouverture faible comme c'était le cas lors des tests.
- le capteur de vitesse n'offrait pas la précision attendue. Après analyse, il apparaît que le module de lecture du capteur tronquait cette valeur à la valeur entière la plus proche, provoquant une perte significative de précision.
- le module de fusion de données était réglé avec une fréquence d'échantillonnage de 3Hz, ce qui est relativement faible au vu des performances attendues. En effet, dans le pire des cas, l'information parvenants à l'infrastructure pouvait avoir un "retard" allant jusqu'à 300ms.

Ces observations mettent en évidence l'intérêt des tests grandeur nature avec déploiement du système, mais aussi la nécessité d'avoir un observateur extérieur au système qui permet de valider ou non les mesures fournies par les traces du système.

Tests de l'application H&IW

L'application H&IW possède un certains nombre de paramètres permettant de compenser la précision des capteurs ou les délais de communication et de traitement. Ces paramètres agissent sur le calculs des seuils de déclenchement des alertes de différentes manières :

- les différents délais impliqués dans le transfert des information sont compensés en étant ajoutés au paramètre décrivant le temps de réaction du conducteur.
- la précision du capteurs de position du véhicule (le GPS du véhicule) est compensée en "rapprochant l'obstacle" (c.a.d. en modifiant la variable contenant le calcul de la distance du véhicule à l'obstacle).
- enfin, la précision sur la vitesse est compensée en modifiant la valeur fournie à l'application.

Pour procéder aux réglages de ces différents paramètres, plusieurs tests sont effectués à différentes vitesses sans paramètre correctif afin de pouvoir observer les écarts avec les valeurs théorique d'un système parfait.

Puis, l'analyse des traces et des observations doit permettre de calculer ces paramètres correctifs. Les tests doivent donc être effectués en deux phases. Le temps nécessaire à l'analyse des traces et observations n'a pas permis d'effectuer la deuxième phase de tests, sauf pour une vitesse.

La figure A.11 montre une synthèse des observations et des mesures effectués les plus significatives. La dernière ligne montre le résultat pour le tests avec l'utilisation des paramètres correctifs.

Ces tests préliminaires, même s'il ne permettent pas de valider entièrement l'application H&IW, en particulier parce que les paramètres correctifs n'ont pas été complètement testés, ont montré que l'application avait le comportement attendue en terme de déclenchement des alertes, stratégie de routage et autres interactions avec le système SAFESPOT.

	Log loop time	Log loop dist.	First Observed warrning	Observed error
Test at 50km/h	av.:333ms [312,353]ms	4.07m [3.81,4.31] m	~ 20 (for 55m expected)	~35m
Test at 90km/h	av.:522ms [458,707]ms	12.83m [10.75,17.86] m	~ 80m (for 120 expected)	~40m
Test at 120km/h	av.:441ms [240,751] ms	14.89m [8.09,25.52]m	~100m (for 150 expected)	~50m
Test at 50km/h with adaptd parameters	352 ms [222,485]ms	5.10m [2.91,8.39]m	~60m (for 55m expected)	~5m

FIGURE A.11 – Résultats avant et après application des paramètres correctifs

Il ressort que sur 24 contraintes concernés par le cas d'utilisation testé, 19 ont été validé dont 4 partiellement et 5 non testés. Les tableaux récapitulatifs A.12, A.13 et A.14 sont fournis dans la section suivante de cette annexe.

A.2 Listes des contraintes validées lors des tests de H&IW

H&IW functions			
Message Management SP5_RQ02_36_v1.0	The system shall be able to decide if and to whom a message has to be send.	OK	
Identify Safety Critical Situations SP5_RQ05_36_v1.0	The system shall be able to identify safety critical situations at the surrounding of a critical point e.g. an urban intersection.	OK	
SP5_RQ06_19_v1.0 / Priority Level of Message:	The system shall be able to provide a priority level for each transmitted message. This is valid for V2I as well as for I2V communication	OK	
SP5_RQ07_19_v1.0 / Validity of Messages:	The system shall be able to determine the period of validity of the transmitted and received messages.	OK	
Message Management 2 SP5_RQ09_19_v1.0	The system must not send warnings concerning severe dangerous situations (that might cause drivers to extreme driving actions), which does in fact not happen.	Partial	Test site adaptassion is required to avoid all
Data exchange: SP5_RQ12_33_v1.0	The system shall be able to transmit information towards several supporters (e.g. VMS, radio, PND)	OK	Tested during previous integrassion test
Query from the LDM SP5_RQ16_36_v1.0	The system shall be able to query needed data from the LDM.	OK	
Receive data from the LDM SP5_RQ17_36_v1.0	The system shall be able to receive and handle the LDM data.	OK	

FIGURE A.12 – Contraintes portant sur H&IW

LDM functionalities			
SP5_RQ49_19_v1.0 / Position and speed of vehicles:	The RSU subsystem shall be able to receive at minimum the current vehicle position and speed.	OK	
SP5_RQ68_10_v1.0 / Road status:	The system shall acquire data on the weather conditions in the 'critical area' (extent TBD) of the obstacle or accident,	Not tested	
SP5_RQ65_10_v1.0 / Obstacle description	The system shall provide other details of the obstacle where possible : type of object (accident, queue, rocks, dropped load, etc), lanes affected, speed (for moving object), precise location etc. i.e. wet/dry road surface, rain, fog, ice, which may affect stopping distances and visibility.	Ok	Type of object and preceise location tested and ok.
TIMING and Performance			
Simultaneity SP5_RQ18_19_v1.0	The system shall be able to manage all vehicles in the vicinity of a critical point (e.g. the intersection) simultaneously.	Not tested	Test made with only one vehicle
SP5_RQ20_19_v1.0 Time of Warning Generation	The system shall generate and transmit warning information to the drivers timely enough, so that the reaction time left to the drivers is appropriate.	Ok, partialy tested	Callibrassion of H&IW has been proved efficient for tested use cases
SP5_RQ22_10_v1.0 R/S Warning	The system shall provide a roadside warning in fewer than 2 sec (TBC) in the critical zones to permit vehicles to reduce speed and/or change lane in order to avoid a (secondary) collision.	OK	With calibrassion of H&IW
SP5_RQ35_19_v1.0 /Time of Data Delay 2:	In the roadside sub-system the delay between generation of warnings and transmission of this data to the vehicles shall be smaller than 50 ms.	Ok	

FIGURE A.13 – Contraintes portant sur la base de donnée

MESSAGES EXCHANGE and Filtering			
SP5_RQ34_19_v1.0 / Directly address a Vehicle:	The roadside sub-system must be able to address data directly to one vehicle distinctively (or a group of vehicles) or to broadcast to all vehicles in the vicinity of the RSU.	OK	
SP5_RQ50_36_v1.0 / Transmission of warnings:	The system shall be able to transmit the warning / recommendation to equipped vehicles.	OK	
SP5_RQ69_10_v1.0: V Warning:	The system shall send appropriate warnings to equipped vehicles within the critical (e and a) zones to permit them to reduce speed and/or change lane and avoid a (secondary) collision.	OK	
SP5_RQ37_19_v1.0 / Filter for relevant data:	The vehicle subsystem should be able to filter the relevant information to the driver.	Partial	Filtering at vanet level, but no HMI implemented
SP5_RQ51_19_v1.0 / HMI:	In vehicle assistance HMI shall present warnings to the driver in an intelligent way without distracting him. Example: Use of different actuators to smooth signalizes hazards.	Not tested	
SP5_RQ48_36_v1.0 / Environmental Data:	The system shall receive data from environmental sensors about weather (rain, snow, fog...).	Not tested	
SP5_RQ52_36_v1.0 / Static Vehicle Data:	The system shall receive static vehicle data like width, length, type of vehicle, mass.	Partial	Only vehicle type has been tested
SP5_RQ60_36_v1.0 / Lack of Friction:	The system shall receive information about the lack of friction of a road segment.	Not tested	
SP5_RQ61_36_v1.0 / Data Reception:	The system must be able to receive data from the vehicles as well as the infrastructure.	OK	
TOTAL	**24 requirement, 19 ok (with 4 partial) and 5 not tested**		

FIGURE A.14 – Contraintes portant sur les échanges de messages

A.3 SAFESPOT dans le contexte Européen d'harmonisation des STI

Les applications d'assistance au conducteur du projet SAFESPOT (comme H&IW), ne sont pas automatiquement compatibles au standard décrit dans FRAME pour plusieurs raisons :

- SAFESPOT prend en charge les conducteurs des véhicules, les piétons et les cyclistes. Ces différents types d'utilisateurs sont réunis au sein de l'utilisateur SAFESPOT nommé " SAFESPOT end User". Or, l'utilisateur type proposé dans l'architecture Frame, et dont la sémantique se rapproche le plus du "SAFESPOT end User", est le conducteur (*Driver*), ce qui ne permet pas une équivalence stricte avec les acteurs de SAFESPOT. Le problème se pose aussi avec les véhicules de maintenance et d'intervention qui n'ont pas d'équivalent direct dans FRAME.
- Si les domaines fonctionnels de Frame correspondant à ceux de SAFESPOT peuvent être identifiés. Leurs relations, tel que spécifiées dans FRAME, ne correspondent pas au découpage fonctionnel choisi dans SAFESPOT (et présenté dans le diagramme de composant chapitre 3 figure 3.2).

Il est néanmoins possible de montrer un niveau de compatibilité entre l'architecture de SAFESPOT et l'architecture générique de FRAME, et ainsi spécifier un niveau de compatibilité entre SAFESPOT et les autres systèmes ITS compatibles FRAME. Ainsi , SAFESPOT est compatible "FRAME" dans les cas d'utilisation n'impliquant pas les adaptations que nous venons de citer. La figure A.15 montre comment les services proposés par SAFESPOT pour l'aide et l'assistance au conducteur (*"Adas" : Advanced Diver Assistance Systems)* sont compatibles avec FRAME. Pour ce faire il faut restreindre SAFESPOT au cas où les conducteurs vulnérables ne sont pas pris en charge, et les services pour les véhicules de maintenance et d'intervention ne sont pas considérés.

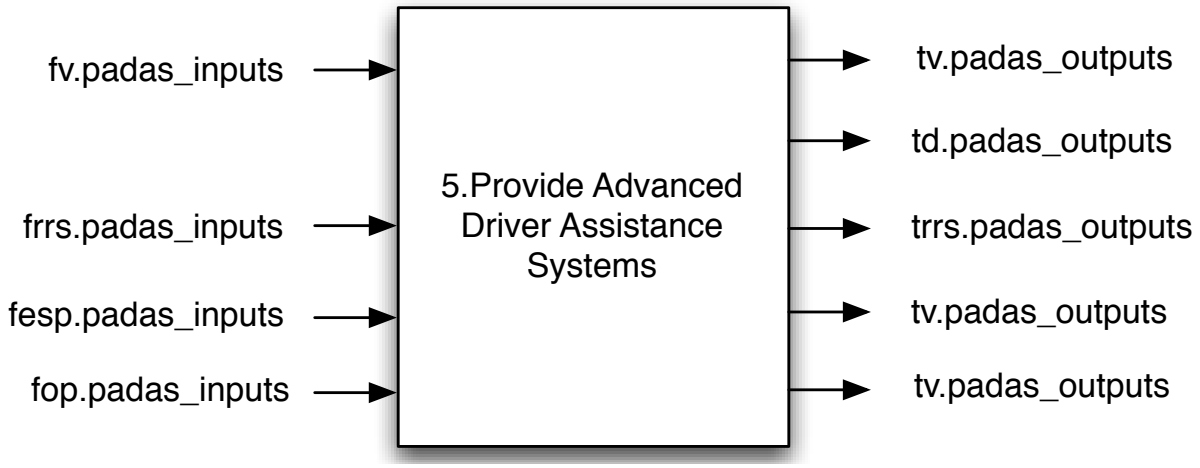


FIGURE A.15 – L'Architecture fonctionnelle de haut niveau de la partie "Adas" de SAFESPOT au format FRAME

Sur cette figure, les flots de données sont nommés conformément à l'architecture FRAME, indiquant ainsi les différents éléments mis en jeux à chacune des extrémités du flot.

– La première partie du nom, qui finit par un point, indique avec la première lettre *f* (pour *"from"*) ou *t* (pour *"to"*) si le flot est entrant ou sortant. Cette partie est suivie des initiales du module fonctionnel ou acteur à l'origine du flot.
– La deuxième partie indique la nature du flot. Elle est déterminée par le champ fonctionnel impliqué et la nature sortante ou entrante du flux.

Par exemple, le premier flot en haut à droite, est nommé *fv.padas_inputs* ce qui indique
– un flux venant (*f*)
– de l'acteur "véhicule" (*v*),
– de type "données d'assistance au conducteur" (*padas* pour *Provide Advanced Driver Assistance Système*)
– entrant (*inputs*).

Les autres acteurs impliqués dans les flot de ce modèle sont :
– les systèmes liés à la route (*rrs* pour *Road Related System*)
– les fournisseurs de services externes (*esp* pour *External Services Provider*)
– les opérateurs (*op* pour *Operator*)
– le conducteur (*d* pour *Driver*)

A.4 Cas d'utilisation

Voici des exemples de cas d'utilisation concernant les services pris en charge par l'infrastructure du système SAFESPOT, et qui sont liés aux scénarios no 13 "Accident as Obstacle" présentés chapitre 4.

A.4.1 UC 12 : "Véhicule d'assistance"

Le premier exemple est une exception à la démarche initiale consiste à relier les cas d'utilisation à des justifications accidentogènes. Ici, les préoccupations des gestionnaires d'infrastructure quant à la sécurité des équipes travaillant sur le réseau sont prises en compte. Ce cas d'utilisation est présenté figure A.16.

A.4.2 UC 14 : "Embouteillage potentiellement mal anticipé"

Ce cas d'utilisation est intéressant car il est très proche de celui présenté chapitre 4. Alors que la situation prise en charge par ce UC est différente, et que les informations nécessaires à son fonctionnement ne sont pas les mêmes, les interactions entre les différentes entités du système sont identiques. Il est présenté figure **??**.

A.4.3 UC 17 : " Piéton sur autoroute"

Dans ce cas d'utilisation, on voit comment le système SAFESPOT peut réagir à une suite d'événements dangereux. En effet, la situation prise en charge ici peut survenir à la suite d'un accident comme celui décrit dans le cas d'utilisation présenté au chapitre 4. Les cas d'utilisation peuvent donc s'enchaîner et fournir des services de sécurité en cas de défaillance de la prise en charge du cas d'utilisation précédent. Ce cas d'utilisation est présenté A.18.

Case Name	Assistance vehicle patrolling or signalling a traffic event on a road	
Case ID	SP5_UC_12_v1.0	
Status	Final	
Short description	An assistance vehicle patrols or signals a traffic event (e.g.: accident, traffic jam, road work…). The vehicle is stopped or it moves slowly (backward or forward) on an emergency or a principal lane. All vehicles arriving or these that are already in the vicinity of the assistance vehicle are warned about the presence of a slow or stopped vehicle. It may be associated with a speed limit and the additional alert that may be carried by the assistance vehicle.	
Purpose	To inform driver who approaches the assistance vehicle in order to avoid an eventual collision.	
Rationale	Assistance vehicles patrol or signal a traffic event frequently. No specific data, but strong support from road operators in order to protect their employees	
Authors	Joël RECEVEUR – SODIT – receveur sodit.info (Responsible) Fabien BONNEFOI – COFIROUTE – fabien.bonnefoi cofiroute.fr	
Driving environment	Highway and expressway	
Vehicle probe type	All	
Risk's source	▪ A stopped or slowly moving assistance vehicle ▪ Presence of an assistance vehicle in the emergency lane ▪ Maladjusted speed of an approaching vehicle ▪ Drivers' behaviour, inattention, environment conditions ▪ Bad visibility - No respect of inter-distance	
Successful end condition	In approach of the assistance vehicle, drivers adapt their speed and respect a security distance in order to avoid a collision.	
Failed end condition	Collision with an assistance vehicle. Excessive speed of a vehicle that overtakes an assistance vehicle.	
Trigger	An assistance vehicle beginning a patrol or signalling an event on a road.	
Frequency of occurrence	No further information until now	
Primary Actor	Assistance vehicles	
Secondary Actor(s)	▪ All other vehicles in approach of the assistance vehicles ; ▪ Road operators.	
Scenario Description	Step	Action
	1	Assistance vehicle is stopped on an emergency lane or on a principal lane and signals an event on the motorway.
	2	Assistance vehicle indicates its localisation and the raison of perturbation
	3	The infrastructure base application analysis this information and transmits it towards the vehicle approaching the assistance one
	4	The approaching driver is informed about the presence of the assistance vehicle and the eventual event perturbation on a road. He may also receive information about speed limit.
	5	
Extensions	Step	Action
	1a	The assistance vehicle moves slowly on a road (backward or forward)
User Needs	▪ Road operators want to reduce the number of accidents between assistance vehicles and other vehicles ▪ Road operators want to transmit warning information towards the driver approaching the assistance vehicle that signals a traffic event or that carries out a snow-removal or a salting	

FIGURE A.16 – UC12 : Cas d'utilisation : "Embouteillage potentiellement mal anticipé"

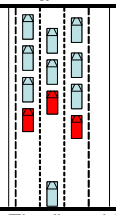
Case Name	Traffic Jams as an obstacle (extension to Slow Moving Vehicle)	
Case ID	SP5_UC14_v1.0	
Status	Final	
Short description		The end of a traffic jams shall be interpreted as an obstacle. In fact, especially on Highways where the speed limit is high, an incoming vehicle with some degraded conditions, either driver drowsiness or bad weather, may react lately. The driver must be warned few seconds before arriving on the end of the jams.
Purpose	The first objective is to avoid accident at the end of the traffic jams. Next, another objective could be more traffic oriented with traffic re-routing	
Rationale	On high density of population area, each day, traffic jams are a large cause of loss of time. Moreover, when other parameters occurred, as bad weather condition, low visibility, … driver can perceive the end line of a traffic jam too late, resulting in a car crash.	
Authors	Sébastien Glaser – LCPC / Fabien Bonnefoi - COFIROUTE	
Driving environment	Highway for high speed related problem, Rural Road for lack of visibility problem	
Vehicle probe type	All	
Risk's source	Lake of reaction or late reaction of the driver. This may be worsen with respect to environment situation as visibility	
Successful end condition	Vehicle stops at the end of the traffic jam	
Failed end condition	Not certainly, but may result in a front-end collision at high speed	
Trigger	Traffic jams emerging	
Frequency of occurrence	The problem of wrong speed with traffic represents about 5.5% of car accident on highway	
Primary Actor	Last stopped vehicle or infrastructure	
Secondary Actor(s)	Incoming vehicle	
Scenario Description	**Step**	**Action**
	1	Detection of the traffic jams
	2	Detection of the end line of the traffic jam, depending on the detectors: ▪ Vehicles emits a stop condition with position, collected by infrastructure ▪ Vehicle send local map to the infrastructure ▪ Infrastructure detects the end line of the traffic jams
	3	Infrastructure based application gathers data and analyses the end line of the traffic jams
	4	Infrastructure creates a safe area at the end line of the traffic jams, and sends message
Extensions	**Step**	**Action**
	4b	Incoming traffic may be re-routed
User Needs	*Last stopped vehicle:* Prevent collision with oncoming vehicles *Oncoming vehicles:* To be warn of incoming traffic jam	
Corresponding requirements	▪ V2I communication ▪ Local Map ▪ Infrastructure based sensors to detect end line of a traffic jams	
Related UCs		

FIGURE A.17 – UC14 : Cas d'utilisation : "Véhicule d'assistance"

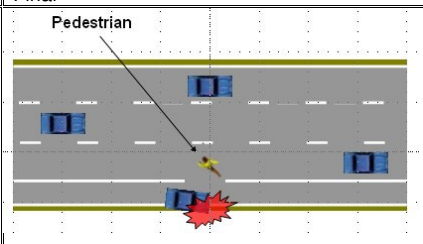
Case Name	Pedestrian on motorway	
Case ID	SP5_UC17_v1.0	
Status	Final	
Short description	 Pedestrians can be encountered on motorway roads when they get out of a broken down vehicle, during road works or for other reasons. They represent a high risk for the traffic flow and for them self. When the sensors of the infrastructure or those of equipped vehicles detect the presence of a pedestrian on the road, a warning is triggered to enhance their security and the security of oncoming vehicles.	
Purpose	Enhance the protection of pedestrians on motorway.	
Rationale	About 15% of lethal accidents on French motorway involve a pedestrian.	
Authors	Fabien Bonnefoi, COFIROUTE	
Driving environment	Motorway roads	
Vehicle probe type	All	
Risk's source	Due to an accident, a broken down vehicle or road works, a pedestrian is present on the motorway.	
Successful end condition	During the presence of pedestrians on the road, vehicles are warned to adapt their speed. No pedestrian collided with a vehicle.	
Failed end condition	An accident occurs with a pedestrian.	
Trigger	The SAFESPOT system detects the presence of a pedestrian.	
Frequency of occurrence	About 15% of lethal accidents on French motorway involve a pedestrian.	
Primary Actor	A pedestrian	
Secondary Actor(s)	Oncoming vehicles	
Scenario Description	**Step**	**Action**
	1	The system (i.e. the infrastructure or equipped vehicle) detect the presence of a pedestrian on the road.
	2	The system sends a warning about the presence of a pedestrian to the oncoming vehicles.
	3	The drivers take action accordingly by slowing down.
User Needs	The presence of pedestrian needs to be detected by the system. Oncoming vehicles need to be warned about the presence of a pedestrian.	
Corresponding requirements	SP58RQ20, SP5_RQ32, SP5_RQ47	
Related UCs	Deviation for road work, accident as an obstacle.	
Open issues		
Comments		

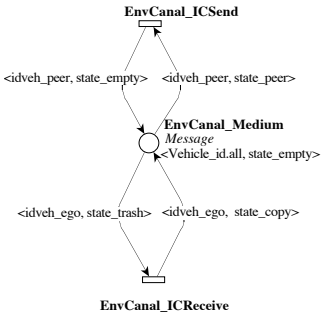
FIGURE A.18 – UC17 : Cas d'utilisation : " Piéton sur autoroute"

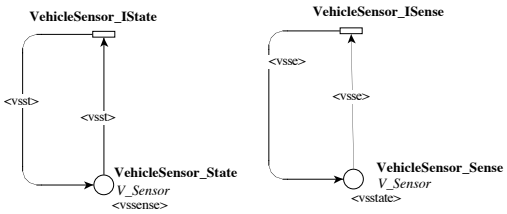
A.5 Modélisation formelle de l'architecture "SAFESPOT"

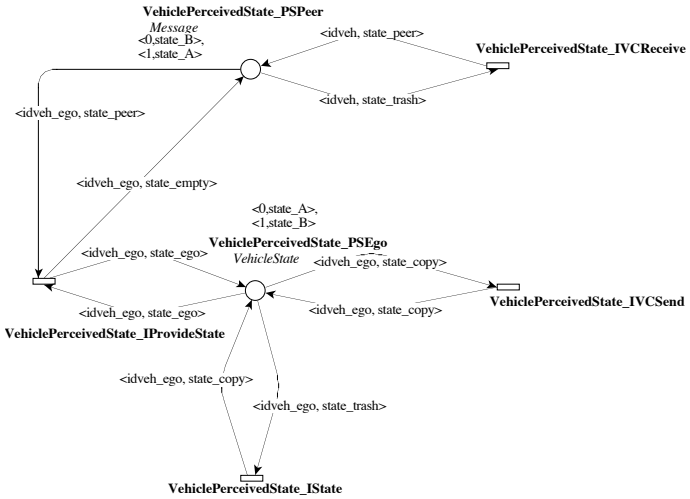
Nous présentons ici la relation entre les composants UML du système SAFESPOT et le patron de modélisation formelle pour le cas d'étude présenté chapitre 5.

Modèle 1.45

Correspondance Visuelle entre les composants du diagramme UML et ceux du modèle RdP - (Page 1)







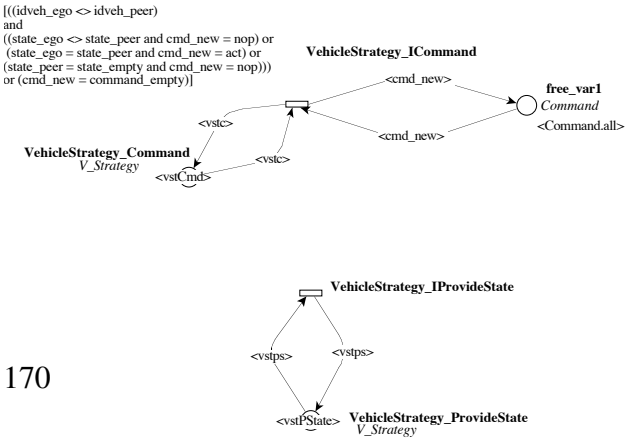
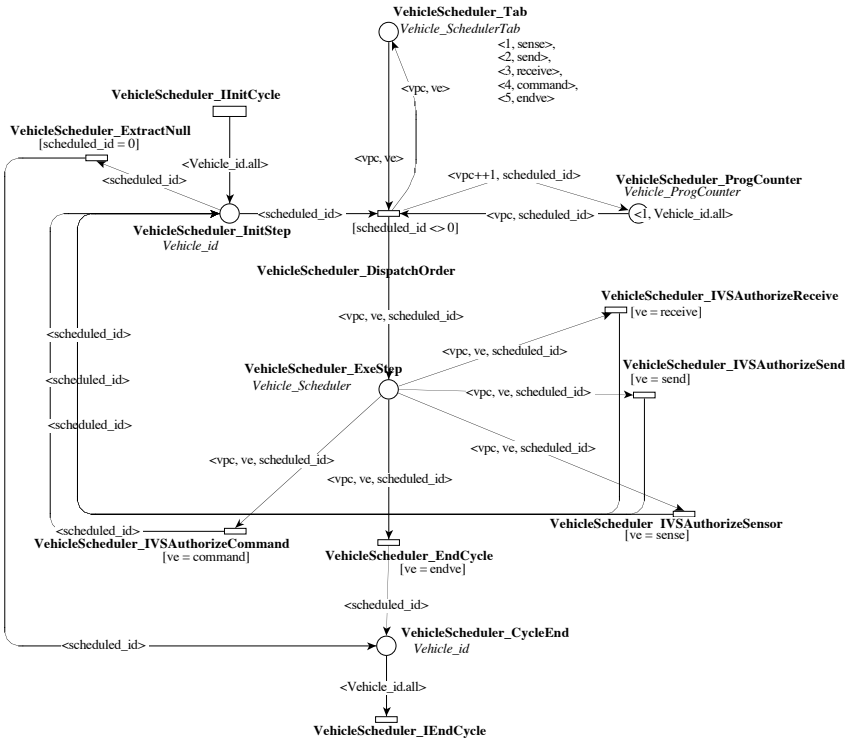
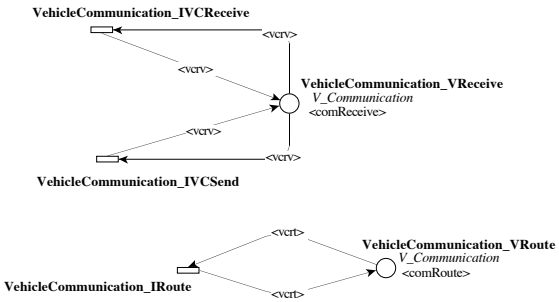


FIGURE A.19 –

Modèle 1.45

Correspondance Visuelle entre les composants du diagramme UML et ceux du modèle RdP - (Page 2)





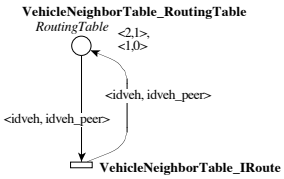
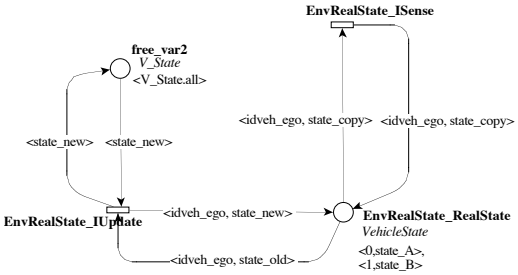
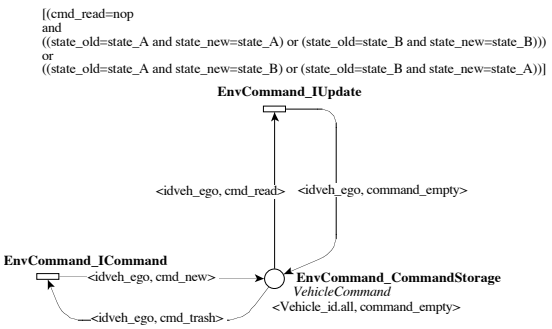


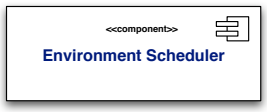
FIGURE A.20 –

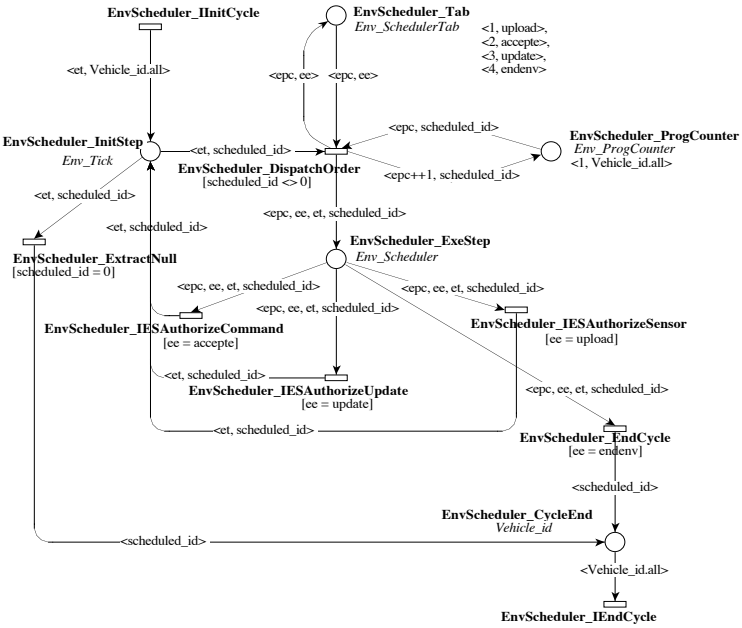
Modèle 1.45

Correspondance Visuelle entre les composants du diagramme UML et ceux du modèle RdP - (Page 3)









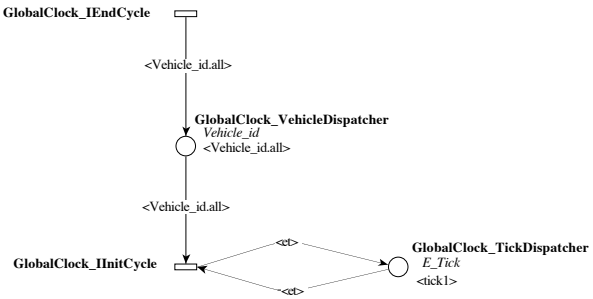


FIGURE A.21 – 173

A.6 Liste de publications

A.6.1 Systèmes de Transport Intelligents

From user needs to application, the SAFESPOT approach based on roads data analysis.Fabien Bonnefoi, Francesco Bellotti, and Tobias Schendzielorz. *In 6th European Congress and Exhibition on Intelligent Transport Systems and Services*, Aalborg, Denmark, June 2007.

Infrastructure-based co-operative architectures : How safespot deals with different road network areas. Filippo Visintainer, Fabien Bonnefoi, Francesco Bellotti and Tobias Schendzielorz. *In 14th World Congress on Intelligent Transport Systems*, Beijing, China, October 2007.

Safespot applications for infrastructure-based co-operative road safety. Fabien Bonnefoi, Francesco Bellotti, Tobias Schendzielorz, and Filippo Visintainer. *In 14th World Congress on Intelligent Transport Systems*, Beijing, China, October 2007.

SAFESPOT Specification method : an example with infrastructure based applications. Fabien Bonnefoi and Fahim Belarbi. *In 15th World Congress and Exhibition on Intelligent Transport Systems and Services*, New York, U.S.A., October 2008.

Fabien Bonnefoi, Gwenaëlle Toulminet, Jacques Boussuge et Claude Laurgeau. **Synthèse comparative des projets européens CVIS, SAFESPOT, COOPERS ; et participation de l'A.S.F.A.**.*ATEC-ITS France International Congress*, Versailles, France, Février 2009.

A.6.2 Méthodes Formelles

Fabien Bonnefoi, Lom Messan Hillah, Fabrice Kordon, et Guy Frémont. **An approach to model variations of a scenario : Application to Intelligent Transport Systems.** *In Workshop on Modelling of Objects, Components, and Agents (MOCA'06)*, Turku, Finland, June 2006.

Fabien Bonnefoi, Lom Messan Hillah, Fabrice Kordon, et Xavier Renault. **Design, modeling and analysis of ITS using UML and Petri Nets.** *In 10th International IEEE Conference on Intel ligent Transportation Systems (ITSC'07)*, pages 314 à 319. IEEE Press, 2007.

Fabien Bonnefoi, Christine Choppy, and Fabrice Kordon. **A discretisation method from coloured to symetric net : application to an industrial example.** *LNCS Transactions on Petri Nets and Other Models of Concurrency III (ToPNoC),* Lecture Notes in Computer Science,Vol. 5800, ISBN : 978-3-642-04854-8

Sélectionné parmi les meilleurs papier du :
9th Workshop and Tutorial on Practical Use of Coloured Petri Nets and CPN Tools.
Première publication :

Fabien Bonnefoi, Christine Choppy, et Fabrice Kordon. **«A discretisation method from Coloured to Symetric Nets : Application to an industrial exemple»** in *9th Workshop and Tutorial on Practcal use of Coloured Petri Nets and CPN tools*, ISSN :0105-8517, pages 183-202, 2008.