Modélisation synchrone appliquée à la sûreté de fonctionnement

Tullio Joseph TANZI
Ecole des Mines de Paris, Pôle Cindyniques
06904 Sophia Antipolis cédex, France.
Tullio.Tanzi@cindy.cma.fr

Charles ANDRÉ
Laboratoire I3S, UNSA/CNRS
BP 121, 06903, Sophia Antipolis cédex, France.
andre@unice.fr

Résumé
Dans des applications critiques touchant à la sûreté, une solution doit être validée avant d'être déployée. On interdit le test des solutions sur le système réel pour des raisons de sûreté. Toute tentative de modification d'un système de sûreté, ou d'introduction d'un nouveau système doit être examinée avec soin. Les solutions proposées doivent être validées, et si possible, d'une manière formelle. C'est le cas des systèmes qui sont en charge, sur les autoroutes, de la prévention des accidents et de la collecte d'informations lorsqu'un accident se produit.
Cet article décrit une approche pour valider un système touchant à la sûreté pour les autoroutes avant d'entreprendre sa réalisation effective. Le système est modélisé comme un ensemble d'agents répartis dans l'espace et qui coopèrent en temps réel. Le comportement des agents est exprimé par un modèle synchrone graphique (syncCharts). Grâce aux outils disponibles dans l'environnement de programmation du langage synchrone « Estérel », enrichis de quelques outils développés spécialement, nous pouvons animer ces modèles et tester leur comportement sur des scénarios de test (prototypage rapide d'application). Les tests ne couvrant pas toutes les évolutions possibles nous vérifions ensuite formellement des propriétés critiques pour l'application. Cette analyse formelle est rendue possible par les fondements mathématiques du modèle utilisé. Les propriétés analysées sont essentiellement des propriétés de sûreté (safety properties) ; des résultats concernant des propriétés de performance sont également accessibles.

Mots clés : Modélisation du fonctionnement, Sûreté de fonctionnement, Évaluation de performances, Prototypage, Modèle synchrone

1. Introduction

Dans le cadre d'un marché classique, les exigences de performances sont fournies par le donneur d'ordre en fonction de la finalité de l'application, du respect des règles de l'art, des aspects normatifs et réglementaires, de son expérience acquise et de ses contraintes d'intégration vers ses autres systèmes. Ces performances sont définies au plus tôt dans le cycle de vie du système. Après dépouillement des offres et passation du marché, elles ne sont plus remises en question et deviennent contractuelles, constituant un engagement et un référentiel sur lequel se greffent toutes les actions de spécifications et ultérieurement de validation du système.
Se pose alors la question de savoir si le système ainsi conçu ou choisi satisfait ses exigences. Pour ce qui concerne les aspects fonctionnels, la solution consiste généralement à mener une campagne d'essais dans des conditions aussi proches que possible des limites prévues. Ceci est beaucoup plus difficile pour les aspects de sûreté de fonctionnement. Il n'est en général pas possible de prendre en compte toutes les situations correspondant aux exigences fixées.
Pour les applications critiques, c'est à dire celles pour lesquelles une défaillance du système serait susceptible d'entraîner des atteintes importantes aux personnes et/ou aux biens, il est possible d'obtenir une confiance suffisante dans le fonctionnement du système en imposant des méthodes de développement et de validation adaptées. Dans la plupart des secteurs concernés il existe des normes préconisant les méthodes à utiliser.
Les techniques appliquées dans le cadre de l'analyse fonctionnelle de sécurité permettent de mettre en évidence des défauts au niveau de spécifications externes qui seraient restés inaperçues lors des diverses phases de l'étude. Les progrès de la sûreté de fonctionnement ont été importants ces dernières années pour les composants techniques des systèmes à risques. Néanmoins il se pose encore de nombreux problèmes :

- Quel niveau de fiabilité doit-on affecter à chaque composant?
- Comment évaluer cette fiabilité?
- Quels sont les principaux risques de défaillance?
- Comment les identifier ?
- Comment les mesurer ?
- …

La conception de systèmes à risques exige de répondre à ces interrogations bien avant le début de l'exploitation. Dans ce contexte, il est important pour les industriels de disposer d'outils permettant d'évaluer :

- La faisabilité et l'efficacité des politiques de production et de soutien,
- Les performances opérationnelles des systèmes, en particulier disponibilité, sécurité, robustesse, …

La simulation se justifie dans cette démarche, pour étayer l'approche technologique en amont et aider à répondre à plusieurs typologies de difficultés :

- Complexité du système : dynamique, nombreuses ressources en interaction, déploiement progressif,
- Faible retour d'expérience,
- Nécessité de réaliser des approches multiples et partielles,

…

Pour illustrer notre démarche dans le cadre de cet article, nous l'avons appliquée à la mise en œuvre de matériel de sécurité déployé sur les réseaux routiers et autoroutiers. Dans le contexte moderne de la route, ou le ruban d'asphalte s'enrichit de plus en plus d'équipements électroniques sur le bord de la chaussée, la place de l'information devient primordiale. Les dispositifs d'information, qui héritent de la complexité des systèmes de communication modernes, voient leur fonctionnement influer fortement sur le niveau de sécurité des biens et des personnes utilisant l'autoroute.
Dans ce contexte, il est important d'avoir des systèmes fiables, robustes, mais cela est indissociable de la nécessité de disposer de la preuve que ces objectifs sont atteints.

1.1. Contexte de l'étude

Dans cet article, nous adoptons une approche synchrone pour modéliser et analyser les comportements de dispositifs liés à la sécurité sur les autoroutes. Cette étude a été réalisée en commun par une équipe Ecole des Mines de Paris et une équipe de l'Université de Nice/CNRS, en relation avec une société d'autoroutes. Nous expliquons comment cette approche apporte des garanties sur le comportement de parties du système et permet de tester le comportement du système global dès les phases de conception. Le modèle synchrone graphique retenu, qui est fondé sur les notions d'états et transitions, permet de plus une communication avec les utilisateurs du système, sans avoir recours explicitement au langage synchrone sous-jacent. Les outils informatiques associés facilitent l'exécution et l'exploitation de scénarios de test. Ces tests peuvent être également menés de façon interactive. Plus intéressante encore est la possibilité d'établir formellement des propriétés de sûreté (safety properties). Ces connaissances acquises pendant la phase de conception, c'est-à-dire avant la réalisation effective du système, peuvent être répercutées immédiatement vers le bureau d'études.

1.2. Présentation de la problématique

Classiquement, la Sûreté de Fonctionnement (SdF) se décompose en quatre composantes qui sont décrites par la figure 1. Lors de la conception d'un système, il est nécessaire de les prendre en compte dès l'origine, et de répondre au mieux à ces préoccupations.

La mise en œuvre classique de la démarche de sécurité est réalisée selon le schéma décrit ci-dessous :

- Identifier les scénarios d'accidents possibles,
- Évaluer, qualitativement ou quantitativement, les risques correspondants,
- Attribuer à chaque risque identifié un niveau de gravité,
- Hiérarchiser les risques, à partir d'une échelle de criticité,
- Sélectionner les risques non acceptables.

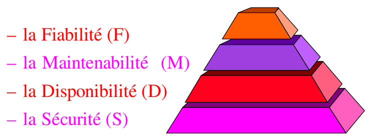

Figure 1: Les composantes de la SdF

Une aide efficace

- à la conception
- à l'évaluation du potentiel de fiabilité du produit (modèles, bases de données)
- à la validation : analyses qualitatives, essais, ...

demande la disponibilité d'un système adapté. Ce système doit être capable de générer exhaustivement tous les scénarios possibles, d'évaluer les interactions entre les acteurs, de tenir compte de leur vraisemblance. Il doit s'appuyer sur des modèles rigoureux et puissants, adaptés aux différents stades de la vie du système (spécification, conception, tests, exploitation). En effet, les causes essentielles d'échecs ou de défaillances constatées en utilisation, ou à l'inverse les cas de sur-spécification, s'expliquent très souvent par des conditions réelles d'emploi différentes de celles spécifiées, soit par méconnaissance au stade du développement des situations réelles d'exploitation, soit par insuffisance d'analyse

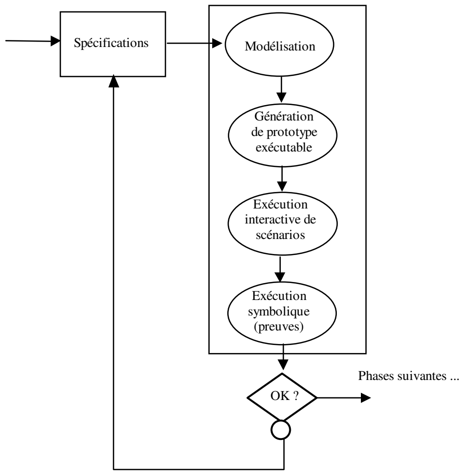

Figure 2: Phase analyse/concept préliminaire.

1.3. Justification des choix

La sûreté de fonctionnement se construit dès l'expression des besoins à l'aide de méthodes et outils spécialisés. Chacun des éléments constituant le système se mesure ou s'évalue. Dans ce contexte nous avons défini notre approche qui se caractérise par les points suivants :

- Mise en œuvre dès les premiers stades (analyse et conception préliminaire), afin de réduire la difficulté et les coûts d'application des modifications sur le système.
- Obtention rapide de prototypes (logiciels) exécutables. Leurs buts est de trouver dès les premières phases du cycle de développement les erreurs de conceptions et les choix non pertinents.
- Choix d'une modélisation par une approche synchrone qui a été introduite, entre autre, pour écrire des programmes sûrs et offrant des possibilités de vérifications formelles.

La vérification des diverses options retenues dans le cadre de la conception d'un projet n'est pas toujours possible avant sa réalisation. Bien souvent, elle s'appuie sur la réalisation et l'utilisation d'un prototype. Cependant, le prototype peut être difficilement exhaustif et totalement représentatif du système final. Il en est de même pour les conditions d'essais.

En réalisant un *prototype virtuel* du système, capable d'accueillir les modèles des comportements les plus critiques du système étudié, il est possible de reproduire les fonctionnements les plus difficiles à mettre en œuvre dans la réalité. La prise de compte de ce processus dès les premières phases du projet permet de rétro agir sur la conception.

1.4. Plan du document

Après une introduction sur l'étude et son contexte, nous présentons la problématique.
La modélisation synchrone qui est privilégiée dans notre approche, fait l'objet de la troisième partie. On y présente les hypothèses sous-jacentes, les variantes graphiques et l'usage que nous voulons en faire dans le problème autoroutier étudié.
La quatrième partie illustre les principes exposés précédemment. Il s'agit d'une étude de cas qui traite d'un problème touchant à la sécurité routière, lié à une utilisation optimisée et sûre du réseau d'appel d'urgence. Les modèles de comportement pour divers agents de l'application sont présentés et commentés.
La partie suivante est consacrée aux modèles exécutables et aux simulations interactives qui peuvent être menées sur nos prototypes logiciels.
Les aspects preuves formelles sont abordés dans la sixième partie. Les techniques mises en œuvre sont justifiées. Les limitations existantes sont dégagées, ainsi que nos propositions pour les surmonter.
Les leçons tirées de cette expérience et les perspectives terminent cet article.

2. Acuité des problèmes de congestion

Chaque jour, il y a en France environ 86 millions de déplacements de personnes en automobile, et une dizaine de millions de mouvements de véhicules de transports de marchandises. Plus de 90 % des trajets effectués par la route ne sont victimes d'aucune congestion [8].
Une voiture en panne, un accident ou des travaux créent un bouchon dès que la circulation dépasse la capacité des voies disponibles. La plupart des bouchons sont dus à de simples ralentissements non provoqués par un obstacle physique. Si un véhicule réduit sa vitesse, par prudence ou par curiosité[1], ceux qui suivent vont ralentir de plus en plus. Lorsque la circulation est dense, ce phénomène peut provoquer l'arrêt du trafic quelques kilomètres en arrière du point où a eu lieu le premier ralentissement. Le bouchon se forme, et dure tant que la densité des véhicules arrivant reste supérieure à la capacité résiduelle des voies.

---

1 Ce phénomène de curiosité explique la constitution de bouchons sur la voie inverse de celle où à eu lieu un accident, et qui est donc libre. Dans ces circonstances, on observe la constitution de bouchons parfois plus importants sur le voie inverse, que sur celle dont la capacité est réduite suite à l'accident.

On peut constater le même type de phénomène lorsqu'un usager change de voie, dans un régime de trafic dense et où la différence de vitesse entre les voies est faible. Il est, dans ce cas, le résultat de la difficulté d'insertion de l'usager dans la nouvelle file.
Les informations dont on peut disposer, de diverses façons pour évaluer un phénomène n'ont pas toutes la même influence sur la conséquence. La variation de certains paramètres, même si elle représente une valeur significative, peut présenter parfois un effet minime sur le phénomène lui-même. A l'inverse, la moindre modification d'un autre paramètre pourra avoir un résultat sans commune mesure avec cette variation.
Aujourd'hui, nous mettons en œuvre de plus en plus de systèmes complexes pour acquérir les informations nécessaires à une bonne gestion du niveau de sûreté des infrastructures routières et autoroutières. L'étude et le déploiement de tels systèmes sont très lourds et prennent beaucoup de temps. En fournissant ces informations aux personnels d'exploitation ils autorisent la mise en œuvre des actions préventives nécessaires au niveau de l'exploitation pour endiguer l'évolution des conditions vers une situation de risque, et si possible d'inverser la tendance pour revenir à des conditions de sécurité optimale. Dans le cas où le recours aux actions curatives s'avérerait nécessaire, la connaissance des conséquences de l'incident permet d'anticiper le plan de réponse et de mieux gérer la crise. Cela aide à comprendre la problématique et la cinématique des existants sur le réseau autoroutier.

3. L'approche Synchrone

La programmation synchrone est apparue dans le milieu des années 80 pour répondre aux problèmes posés par la programmation des « systèmes réactifs ».

3.1. Langages synchrones

C'est David Harel [10] qui a introduit ce qualificatif pour bien les distinguer des programmes « transformationnels ». Les programmes réactifs sont en interaction permanente avec leur environnement et leurs évolutions sont contraintes par cet environnement. Les applications dans lesquelles le contrôle joue un rôle prépondérant sont typiquement des applications réactives. Beaucoup de systèmes réactifs sont de plus soumis à des contraintes dites temps réel. Le fort couplage programme-environnement rend nécessaire une confiance accrue dans les programmes : comportement parfaitement prévisible, ne conduisant en aucun cas à des situations non désirées. Ce souci se retrouve dans les applications critiques.
Les langages bien adaptés à la programmation « transformationnelle » donnent généralement des résultats décevants pour les applications réactives et temps réel. Il est surtout très difficile de garantir les comportements attendus. La raison majeure est la complexité des langages et une sémantique qui ne capture pas l'essence même des interactions.
La solution adoptée par les langages synchrones est la simplicité et une sémantique rigoureuse. La simplicité résulte du fait qu'il s'agit de langages spécialisés : ils se concentrent sur les aspects « contrôle » et ne prétendent pas concurrencer les autres langages dans les traitements algorithmiques classiques. La lecture de l'article de G. Berry [5] qui compare langages généraux et langages spécialisés est très instructive sur ce point. Quant à la sémantique, elle est définie mathématiquement et s'appuie sur quelques concepts simples. Il en résulte des possibilités effectives de raisonnements formels (et automatisables) sur les programmes.
Un numéro spécial des proceedings des IEEE a été consacré à la présentation des langages synchrones [4].

3.2. Les hypothèses synchrones

Les modèles synchrones sont des abstractions qui expriment de façon fine et non ambiguë les comportements des systèmes réactifs. Ces abstractions sont des simplifications de la réalité. Ces modèles se prêtent à la construction d'applications et leur analyse. Il convient, naturellement, de vérifier si ces hypothèses sont acceptables pour les implémentations retenues.

Pour présenter ces hypothèses nous nous basons sur les concepts que l'on retrouve dans le langage Esterel [7] qui est un langage synchrone impératif. Une présentation plus détaillée de l'approche synchrone et des hypothèses sous-jacentes a déjà été publiée par un des auteurs [1].
La première abstraction porte sur les communications : tout échange d'information dans un modèle synchrone se fait par des signaux. Un signal est caractérisé par sa présence à un instant donné. Cette présence est une information fugace, elle rejoint la notion plus classique d'événement. Un signal réduit à cet aspect événementiel est dit signal pur. Un signal porte généralement une information qui présente un caractère plus rémanent. On parle alors de signaux valués. Du point de vue de la programmation ces signaux sont typés. Les signaux valués peuvent être vus comme des messages.
Dans les modèles synchrones, les signaux sont diffusés instantanément (instantaneous broadcasting). La conséquence est que tout sous-système perçoit exactement la même information à chaque instant ; il n'existe pas les distorsions introduites par les communications asynchrones.
La seconde hypothèse est que les réactions se font en temps nul. Ceci signifie que les sorties sont disponibles instantanément. Ce mode de fonctionnement est particulièrement judicieux pour les systèmes réactifs. L'instantanéité de la réaction fait que tous les traitements se font de façon atomique, les problèmes habituels dus aux évolutions concurrentes sont alors évités.

Ces hypothèses semblent au premier abord irréalistes. En fait, elles apportent un grand confort au concepteur car elles permettent de travailler au niveau des modèles dans un cadre idéalisé. De plus, elles ont conduit à une sémantique simple et rigoureuse, bien adaptée aux systèmes réactifs. La validité des hypothèses pour les systèmes réels peut être facilement testée sur les réalisations.

3.3. Modèle synchrone graphique

Dans le cadre de l'application traitée, nous souhaitions bénéficier des avantages de l'approche synchrone mais sans toutefois imposer à nos partenaires (autoroutiers, bureau d'études, experts) l'apprentissage de nouveaux langages. Nous avons adopté une approche plus familière à la majorité des ingénieurs : une approche graphique. Le modèle retenu : les « syncCharts » [2] est un modèle graphique synchrone. Il s'appuie sur les concepts d'états et de transitions. Un tel modèle permet de « visualiser » les comportements attendus. Par rapport à un graphe d'états ordinaire, le modèle offre des possibilités de description d'évolutions parallèles et de hiérarchie. En ce sens, ce modèle est proche du Grafcet [3] utilisé dans les automatismes logiques et des statecharts de David Harel [9]. Les syncCharts supportent en plus les préemptions. La préemption est la possibilité donnée à un agent d'interdire définitivement (avortement) ou temporairement (suspension) les activités d'un autre agent. Les préemptions sont courantes dans les comportements réactifs. Elles sont associées aux exceptions.
Le passage à ce modèle graphique n'enlève rien à la rigueur du modèle qui est tout à fait conforme aux hypothèses synchrones. En fait tout syncChart peut être traduit automatiquement en un programme Esterel équivalent. L'utilisateur fait ainsi de la programmation Esterel sans le savoir !

3.4. Application au problème autoroutier

L'idée est de représenter le système à concevoir comme un ensemble d'agents interagissant. Le comportement de chacun des agents est exprimé par un syncChart. Les échanges d'informations se font par signaux.
L'utilité des syncCharts est multiple :

- Il s'agit d'une représentation graphique des comportements attendus de l'agent. Ce modèle étant compréhensible de la part des utilisateurs, il est possible de cerner les interprétations erronées ou les incomplétudes, ceci par dialogue avec les professionnels.
- Il existe des outils pour compiler et simuler ces modèles. On peut donc très vite expérimenter sur des prototypes logiciels et les soumettre à des scénarios de test.

- Ces comportements peuvent être vérifiés formellement par des techniques décrites plus loin. Nous ne sommes plus alors tributaires du choix de scénarios de test : l'étude devient exhaustive. Il en résulte une confiance accrue dans la conception.

Le souci majeur que l'on peut avoir avec l'approche synchrone est la vision globale qu'elle impose. Ceci paraît incompatible avec le caractère très distribué (géographiquement) du système étudié. En fait, notre étude porte sur le contrôle des échanges d'informations entre des utilisateurs et le poste de commande. Ce qui nous importe c'est de monter que ce contrôle est assuré de façon prévisible et correcte. Les liaisons passent sur un réseau dédié, qui fonctionne suivant un protocole simple de partage de temps et sur lequel les informations sont diffusées. Dans ces conditions les hypothèses synchrones sont tout à fait réalistes.

4. Étude de cas : Application au RAU

4.1. Introduction

Il s'agit de définir un système d'information destiné aux données techniques nécessaires au maintien du niveau de viabilité et de sécurité d'un réseau d'autoroute. Pour cela il faut connaître en permanence la position des divers mobiles correspondant aux moyens d'interventions. En cas d'incident, le système doit être capable de fournir des informations complémentaires sur l'événement, telle que, par exemple l'image photographique du site de l'incident.
La solution étudiée propose d'utiliser le réseau de communication privé, enterré le long de la chaussée, pour transmettre ces informations. Le réseau s'appuyant sur une base RAU (réseau d'appel d'urgence), le fonctionnement doit privilégier cet usage. Si un usager vient à utiliser ce mode de communication en même temps que les secours, sa communication devient prioritaire au détriment de toute autre opération en cours.
En cas de dysfonctionnement du RAU classique, le système étudié doit pouvoir offrir une alternative de secours en prenant le relais et en acheminant le message jusqu'au prochain point de connexion opérationnel.
La figure 3 reprend ces divers composants du système. Les boîtes C_k modélisent des ruptures possibles de la ligne enterrée.

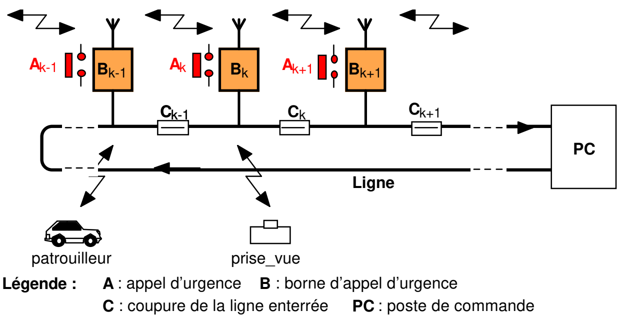

Figure 3: Architecture générale du système

4.2. Définitions générales

Un réseau RAU est composé d'environ 50 bornes téléphoniques séparées de 2 km en terrain découvert et 500 mètres en tunnel. Dans cette première étude on considère une zone géographique de type Var ou Estérel, c'est à dire sans tunnel.
On identifie les acteurs suivants :

- Les patrouilleurs, qui fonctionnent de manière similaire à un rondier,
- Les opérateurs à pied. Dans cette étude, on ne prend en compte que l'opérateur chargé de la prise de vue en cas d'incident,
- Les autres véhicules d'intervention de types divers,
- L'usager, qui est susceptible en cas de problèmes (accident, pannes, malaise, …) d'utiliser la borne d'appel d'urgence (RAU).

Le réseau doit assurer deux types de tâches : la tâche de fond, qui est le suivi des véhicules de patrouille, et les tâches ponctuelles ou exceptionnelles qui sont :

- Le suivi d'autres véhicules,
- Le transfert d'image d'accident,
- L'appel RAU normal,
- L'appel RAU en mode secours du RAU.

Dans notre solution nous distinguons trois modes de fonctionnement :

- Normal,
- Urgent,
- Critique.

4.3. Rôle du patrouilleur

Le patrouilleur parcourt à intervalles de temps réguliers le réseau autoroutier sur une zone donnée, qui correspond à un district et à un linéaire d'environ 50 km. Il communique avec le poste de contrôle via les bornes d'appel d'urgence situées le long de l'autoroute. Nous proposons d'incorporer des « cellules de communication » dans ces bornes. Ces cellules offrent des possibilités de liaison radio faible distance, faible débit entre le patrouilleur et les bornes voisines, ainsi qu'entre bornes voisines elles-mêmes. Nous avons retenu différents modes de fonctionnement pour le patrouilleur :

- Arrêt : Le mobile est en attente, par exemple en position en un endroit donné. On n'émet pas (on peut envisager une émission "dead or live" à récurrence lente, par exemple toutes les heures ou les deux heures). Le mobile se remet en émission dès son départ (détection de la mise en mouvement),
- Patrouille normale : Le mobile est en patrouille normale, sans problème particulier. On émet au minimum. Des récurrences de l'ordre de 5 à 10 minutes sont admissibles. La vitesse de progression de la patrouille est de l'ordre de 60 à 80 kilomètres par heure. Une émission de sa position toutes les 2 à 3 minutes (différence de position de l'ordre de 2 à 3 kilomètres) est suffisante.
- Patrouille rapide : Il s'agit du (ou des) état(s) d'alerte. Dans cet état, le mobile est dans une phase de progression accélérée, pour se rendre sur le site de l'incident. On peut estimer la vitesse de progression dans cette phase dans une fourchette de 100 à 120 kilomètres par heure. Le suivi de la progression doit être plus fin. On passe à des récurrences de l'ordre de 2 minutes (voire 1 mn si besoin).
- Intervention : Le patrouilleur est sur le site d'un incident et il intervient. Dans ce mode de fonctionnement, il est nécessaire de distinguer les opérations d'intervention en protection classique, de la situation de recul en protection de queue de bouchon. En effet dans cette dernière opération, le recul du patrouilleur peut fournir des informations sur la vitesse de formation de la congestion.

Nous imposons l'envoi d'information de position et d'identification à des fréquences dépendant du mode dans lequel se trouve le patrouilleur. On considère que le message envoyé respecte le format suivant : [ID|Position|Horodatage] (voir figure 5). Pour chacun de ces modes des fréquences d'émission des informations sont définies.

État	Vitesse	Récurrence	Distance
Arrêt	0	Détection de mouvement dead or live (1-2 heures)	0
Patrouille	60-80 km/h	5 à 10 mn selon horaire	6 à 10 km
Pat. rapide	100-120 km/h	1 à 2 mn	3 à 4 km
Intervention	Diverses	2 mn	faible
Recul	Faible (< 10 km/h)	1 mn	< 200 m

Figure 4 : Gestion des émissions

ID	Identificateur du véhicule
POS	Position
HORO	Datation de la position

Figure 5 : Message patrouilleur

4.3.1. Description des émissions du patrouilleur

Les émissions sont déclenchées par une temporisation tous les T_0 (voir figure 6). Avec la récurrente interne de t_0, la patrouilleur ré-émet son message tant qu'il n'a pas reçu un acquittement de la cellule la plus proche. Le nombre de ré-émissions est un paramètre à choisir. Nous avons retenu 4 essais. En cas d'échec (dû par exemple à une panne d'émission ou de réception côté cellule ou côté patrouilleur) une alarme est activée sur le patrouilleur (voyant rouge, sonnerie, …) afin que les opérateurs sachent que leur position n'a pu être transmise. De plus la défaillance est enregistrée pour exploitation ultérieure par les services de maintenance. Par contre en cas de réception d'un acquittement le système entre en sommeil, et attend le prochain T_0. La figure 6 précise le chronogramme de l'opération.

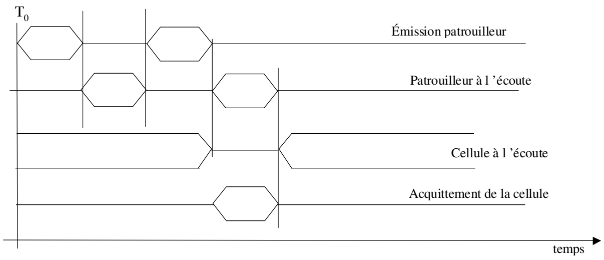

Figure 6: Chronogramme patrouilleur

À tout moment le patrouilleur peut recevoir une requête de position qui lui parvient du centre de contrôle. Dans ce cas, il termine le traitement en cours et répond par l'envoi de la position correspondant à l'endroit où il se trouve à ce moment. Le traitement de la réponse sur requête est la même que l'émission de message en fonctionnement normal.

Les chronogrammes sont très utiles pour exprimer ces comportements temporels. Il est toutefois difficile de couvrir ainsi tous les cas. Le syncChart de la figure 8 exprime le comportement complet de façon beaucoup plus concise et exploitable par les outils d'analyse de programmation synchrone. Ce syncChart décrit le comportement de façon hiérarchique (états emboîtés). Les préemptions sont représentées par des arcs. Comme ce modèle est parfaitement déterministe, il faut savoir prévoir ce qui se passe en cas d'occurrences simultanées d'événements. La structure exprime des priorités : par exemple si il y a occurrence de `t0` et de `acc_p` (accusé de réception en provenance du patrouilleur) simultanément, c'est `acc_p` qui l'emporte car c'est une préemption plus externe. Lorsque les préemptions sont de même niveau, la priorité est donnée explicitement (entier placé à l'origine de l'arc) : `req` (requête venant du poste de commande via une cellule de communication) a priorité sur `T0`.

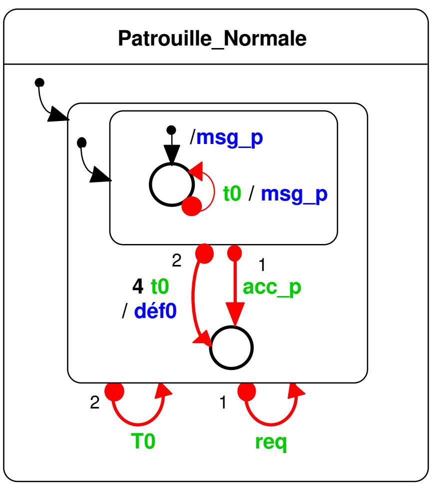

Figure 7 : Comportement du patrouilleur

L'accès au système d'émission est géré par partage de temps synchronisation temporelle. Cela correspond au technique de type TDMA utilisée en radio (voir figure 8). Il est nécessaire de définir le nombre maximum de véhicules autorisés à émettre sur le réseau. On affecte à chacun des véhicules un numéro d'ordre. on divise ensuite l'unité de temps (1 seconde) par le nombre de véhicules et on obtient le nombre équivalent de slots d'émission. chacun des véhicules n'émet que pendant la durée du slot qui lui est attribué.

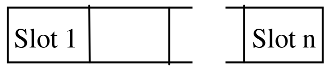

Figure 8: Synchronisation temporelle des émissions

Si l'on retient un nombre maximal de 30 véhicules, pour un district qui couvre une zone géographique correspondant à un linéaire de chaussée d'environ 40 à 50 km, cela nous permet d'avoir des slots d'une durée de 1000/30 = 33.33 millisecondes. Dans le cas où l'on retient un débit de 9600 bauds, cela correspond à des tailles de message maximum correspondant à 320 octets, ce qui est largement suffisant pour contenir un message tel que défini à la figure 5.

4.4. Cellule de communication

La cellule est en attente de réception de message. Lorsqu'elle reçoit une message elle exécute les actions ci-dessous :

- Réception et mémorisation du message,
- Envoi d'un message d'acquittement tel que défini par la figure 9,
- Envoi du message reçu au centre de contrôle,
- Attente de l'acquittement du centre de contrôle.

4.4.1. Description du fonctionnement de la cellule

En fonctionnement normal la cellule est en attente de réception de message du patrouilleur (`msg_p`). Lorsque elle reçoit un message, elle le mémorise puis envoie un acquittement (`acc_p`) à destination du patrouilleur. Le format du message d'acquittement est représenté à la figure 9. Si le patrouilleur reçoit cet acquittement il va arrêter de transmettre le message tous les t0. Dès qu'il ne transmet plus la cellule ne va plus recevoir de message, donc ne plus émettre d'acquittement.

IDREC	Identificateur du véhicule
IDCEL	Identificateur de la cellule

Figure 9: Message d'acquittement de la cellule

Une fois l'acquittement envoyé, la cellule transmet le message reçu au poste de contrôle (`msg_c`). Au passage elle rajoute sur le message son propre identificateur IDCEL. Elle attend ensuite l'acquittement du poste de contrôle (`acc_c`). Le fonctionnement mis en œuvre est du même type que celui du patrouilleur. Attente, ré-émissions jusqu'à réception de l'acquittement. Pour limiter les ré-émissions, on peut fixer une limite maximale au delà de laquelle la cellule passe en erreur. On fixe pour cela un taux de reprise maximal (par exemple 5).

La figure 10 montre le chronogramme de la cellule.

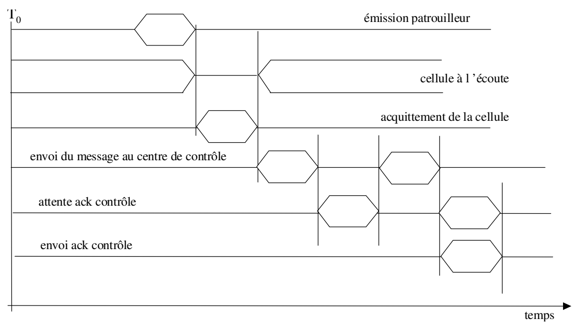

Figure 10: Chronogramme cellule

La figure 11 est le syncChart de comportement de la cellule (pour la partie communication avec le patrouilleur et le poste de contrôle). Le syncChart exprime des comportements plus élaborés. En particulier, il montre des activités concurrentes dans le contrôleur : elles sont séparées par une ligne en tirets. Le macro-état de droite, appelé « `Gestion_réponses` » correspond au fonctionnement décrit ci-dessus. Il comprend lui-même des activités en parallèle : gestion des dialogues avec le patrouilleur et avec le poste de contrôle. La partie gauche spécifie l'attente de réception d'un signal diffusé (`Diff`) par le poste de contrôle et demandant immédiatement la position du patrouilleur. A la réception de `Diff`, la cellule émet `req` (requête) à destination du patrouilleur. Ici encore on applique une technique de ré-émissions multiples en l'attente d'accusé de réception. En cas d'échec, le poste de contrôle est notifié par un accusé de réception négatif (`nacc_c`) qui doit lui-même faire l'objet d'un accusé de réception de la part du poste de contrôle. On peut noter l'usage du signal local `ré_init` qui permet de synchroniser les activités parallèles : la partie gauche demande, par ce signal, la ré-initialisation du gérant de réponses. Le comportement de chaque partie du système a été ainsi décrit à l'aide de syncCharts, non insérés dans cette présentation.

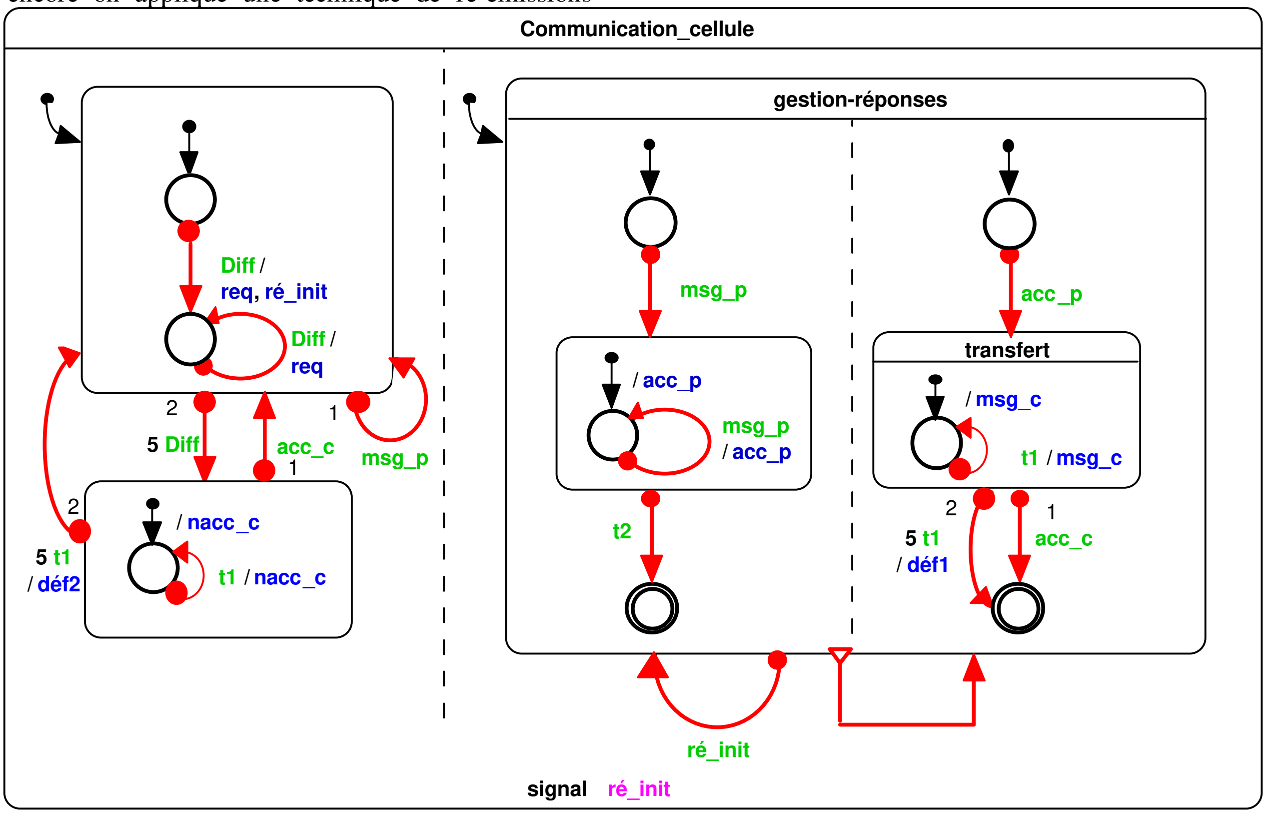

Figure 11: Comportement de la cellule de communication

4.4.2. Cellules et sûreté de fonctionnement

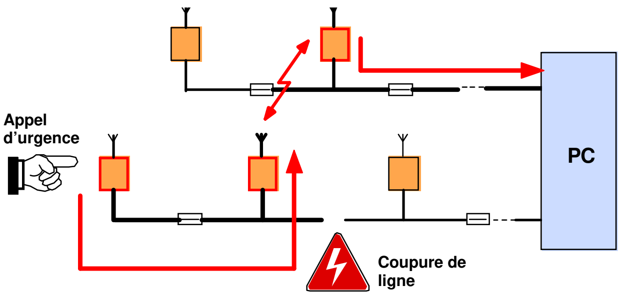

Figure 12 : Exemple de tolérance aux pannes

Sur la figure 3 d'architecture générale du système, nous avons fait apparaître les possibilités de rupture accidentelle sur le RAU. Une des fonctionnalités que nous voulons ajouter aux bornes d'appel d'urgence est la prise en charge de l'acheminement des appels d'urgence en cas de rupture de ligne, par les cellules. La figure 12 donne le principe de cette solution de secours : la cellule située en amont de la rupture retransmet l'appel à une cellule voisine par la liaison radio locale.

Ainsi, outre la gestion des communications avec le patrouilleur (§ 4.4.1), le contrôle de la cellule doit assurer la surveillance de la continuité de la ligne. En présence de rupture, la cellule adopte des modes de fonctionnement dégradés. Le syncChart de la figure 13 précise ces changements de modes. Le corps des macro-états est lui-même décrit par d'autres syncCharts.

Il reste, enfin, une autre fonctionnalité des cellules que nous ne détaillerons pas : la gestion de la transmission des images depuis un appareil de prise de vues mobile vers le poste de contrôle.

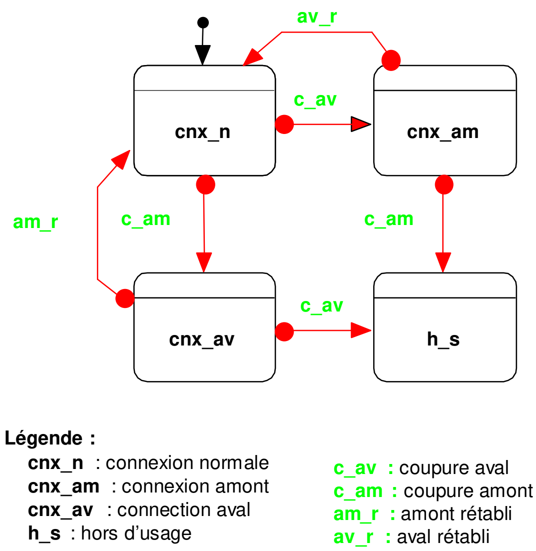

Figure 13 : Modes de fonctionnement d'une cellule

5. Modèles exécutables

Partant des syncCharts, une chaîne de compilation permet de créer des prototypes logiciels exécutables.

5.1. Chaîne de compilation

Grâce à leur sémantique formelle, les syncCharts peuvent être traduits systématiquement en programmes Estérel qui expriment rigoureusement les mêmes comportements. On peut ensuite, sur ces programmes, appliquer les outils développés pour le langage Estérel.
Les principales étapes de la compilation sont les suivantes :

1. Traduction SyncCharts vers Estérel. Il s'agit d'une traduction structurelle. Les programmes Estérel produits sont humainement lisibles et incluent des commentaires de génération. Leur contenu est généralement ignoré de l'utilisateur ; seuls des experts du langage peuvent y trouver des possibilités d'optimisations.
2. Compilation des programmes Estérel en un code intermédiaire. Le code intermédiaire actuellement le plus utilisé est un format qui contient des tables (de signaux, de types, d'actions, …) et des équations booléennes de contrôle. Les actions sont déclenchées en fonction des valeurs prises par ces variables de contrôle.
3. Instrumentation du code pour la simulation. Le code intermédiaire est traduit dans un langage séquentiel d'usage général comme le langage C. Du code est rajouté automatiquement pour l'interfaçage programme/utilisateur : ceci couvre les dialogues d'entrée/sortie mais également les facilités de remonter dans le source. L'outil Xes (X-Esterel Simulator) de l'environnement Estérel engendre automatiquement, à partir de programmes Estérel, un exécutable avec panneaux de contrôle en Tcl-Tk, qui permettent une simulation interactive des programmes.
4. Facultativement, l'utilisateur peut définir sa propre interface graphique pour rendre les simulations plus conviviales. Nous avons opté pour une telle solution.

Un prototype universitaire d'éditeur/compilateur de syncCharts est disponible pour l'enseignement et la recherche. Il se trouve sur le site web d'Esterel [7]. Il existe maintenant un environnement complet commercialisé appelé « Estérel Studio » [11], qui intègre tous les outils Estérel et offre des fonctionnalités accrues, comme l'animation directe des syncCharts en simulation.

5.2. Simulation

Il s'agit d'observer le comportement du système conçu. Pour cela, des scénarios peuvent être créés de façon interactive par appuis sur des boutons associés aux signaux. Xes inclut également un « magnétophone logiciel » qui permet de mémoriser des scénarios de test et de les rejouer. Les réactions aux stimuli sont clairement visualisées par des changements de couleur des signaux sur les panneaux.
En cas de réaction non prévue, on peut grâce aux possibilités de « remonté dans le source », savoir dans quel état se trouve le contrôleur au moment de la réaction erronée. Cette information est très précieuse car les programmes expriment de nombreuses évolutions en parallèle. Il n'est pas facile pour un humain de tracer des programmes parallèles. Le simulateur l'aide grandement en permettant de visualiser, après chaque réaction, les points d'arrêt atteints pour chaque fil d'exécution.
Dans la distribution d'Estérel Studio, ces points d'arrêt sont directement visualisés sur les syncCharts : le contour des états actifs est visualisé en rouge, les transitions qui ont été réalisées sont colorées en vert. Comme le modèle est à la fois parallèle et hiérarchisé ces indications sont quasiment indispensables pour localiser rapidement les erreurs de spécifications de comportement.
Un inconvénient que l'on rencontre parfois avec ces simulateurs est qu'ils ne prennent pas en compte les relations de dépendance existant entre les signaux d'interface. Un actionneur provoque souvent le changement d'un capteur auquel il est lié dans l'environnement. Si nous ne voulons pas tester des scénarios complètement irréalistes, il faut tenir compte de ces dépendances. C'est ce que nous avons fait dans nos simulations en introduisant une modélisation de l'environnement sur lequel agissent les contrôleurs à tester.
La figure 14 représente la chaussée (rectangle intérieur, couvrant un district), 5 bornes reliées par les lignes enterrées et le poste de contrôle. Il s'agit du panneau principal de simulation. Le patrouilleur est visualisé sous forme de petit rectangle qui se déplace de façon automatique sur le rectangle. Lors de sa progression, il apparaît de manière dynamique les liaisons radio entre le patrouilleur et les bornes. Ces liaisons sont orientées de l'émetteur vers le récepteur. Sur ce simple schéma il est aisé de voir qui émet et quand il émet : sur la figure le patrouilleur émet sa position vers les bornes 3 et 4. La série de boutons étiquetés « cuts » permet de simuler les ruptures de lignes. Au cas où un des ces boutons est activé, la partie de ligne associée disparaît sur le panneau. Le bouton « Tick » sert à la personne qui mène la simulation à demander la réaction suivante. Il s'agit alors d'une simulation en pas-à-pas. Il existe également la possibilité de générer le Tick périodiquement (mode automatique).
Notons que pour des problèmes de lisibilité il n'était pas question de visualiser la cinquantaine de bornes du district. Avec 5 bornes on peut déjà tester les problèmes de conflits et de concurrence.

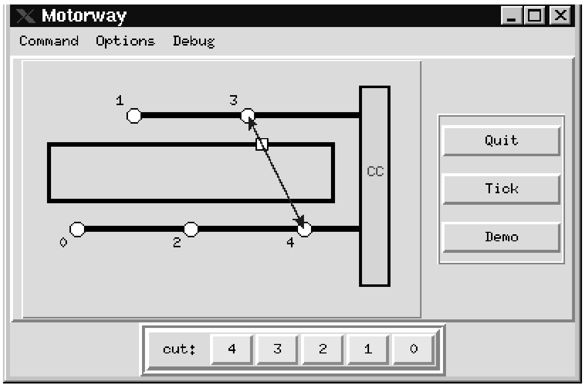

Figure 14 : Panneau de simulation principal

Nous avons également conçu des panneaux pour le patrouilleur (figure 15), pour chaque borne (figure 16) et pour le poste de contrôle (figure 17). Nous expliquons quelques uns des signaux représentés.

Figure 15 : Panneau de simulation pour le Patrouilleur

Par des boutons radio, on sélectionne l'allure du patrouilleur (recul lent, arrêt, avance normale, avance rapide) conformément aux spécifications du paragraphe 4.3. On peut ajuster plus finement la vitesse à l'aide du curseur. Les boutons de la partie inférieures correspondent à des signaux échangés. Certains boutons peuvent être directement activés par la personne qui effectue la simulation : « T » pour transmission du message de position, « B » (breakdown) pour panne de la radio. Les boutons non activables manuellement sont entièrement pris en charge par le simulateur. « R » s'allume quand une requête de position est émise par une borne à destination du patrouilleur. « A » est activé lors de la réception d'un accusé de réception (signal acc_p du paragraphe 4.4.1).

Figure 16 : Panneau de simulation pour une Borne

Sur le panneau de simulation d'une borne on a visualisé à gauche les signaux reçus et à droite les signaux émis. Tout ces signaux n'ont pas été détaillés. Explicitons ceux relatifs à l'utilisateur (automobiliste en difficulté). « C » (Call) est pour l'appel d'urgence, quand le bouton d'appel est maintenu appuyé. « L » (Listen) est pour l'écoute des réponses, le bouton d'appel doit alors être relâché. Ces signaux sont porteurs d'information, ils sont donc des signaux valués. Pour positionner ou lire cette valeur il suffit de cliquer avec le bouton droit de la souris sur le bouton du panneau. Une petite fenêtre de dialogue apparaît alors. Elle peut être cachée et visualisée à volonté, par simples clics.

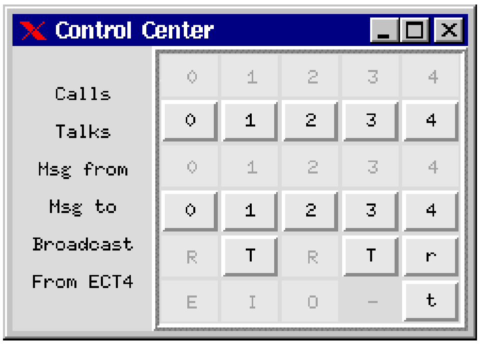

Figure 17 : Panneau de simulation pour le Poste de contrôle

Sur le panneau du poste de contrôle on a reporté tous les appels d'urgences (Calls). Ce sont des signaux valués. Les signaux valués « Talks » portent les réponses. La requête de position du patrouilleur est faite en activant le bouton « r » de la ligne Broadcast.

5.3. Apprentissage

L'animation du prototype est également un excellent outil d'apprentissage. Grâce aux interfaces développées, un utilisateur non spécialiste en programmation réactive peut facilement expérimenter le fonctionnement du système. Il peut éventuellement suggérer des améliorations non seulement dans l'interface, mais aussi dans le comportement. Comme pour les simulateurs de systèmes complexes et coûteux, l'utilisateur pourra être confronté, en simulation, à des situations critiques et des scénarios catastrophiques. Ces scénarios auront été établis au préalable par les experts du domaine.

6. Validation

La simulation est un outil précieux pour tester des scénarios et donc trouver rapidement des erreurs de spécification ou de conception. Toutefois, cette technique est insuffisante dans le cas des systèmes critiques. Les scénarios ne sont que des évolutions possibles du système. Des erreurs peuvent rester cachées dans les évolutions non couvertes.

6.1. Vérification symbolique de modèle

Compte tenu du caractère formel du modèle utilisé, il est possible de *prouver* des propriétés relatives au modèle. La technique que nous avons adopté est celle de la vérification symbolique de modèle (symbolic model checking). L'idée est d'explorer de manière exhaustive l'espace d'état du modèle. Ceci suppose que notre modèle ait un nombre d'état fini. Toutefois, cet espace tout en étant fini peut être si grand que les programmes de vérification sont dans l'incapacité de l'analyser. L'utilisation d'une évaluation symbolique évite une construction explicite de l'espace d'états et permet donc de s'attaquer à des espaces potentiellement très grands. Comme nous l'avons expliqué dans le paragraphe 5.1, nos modèles sont traduits en un système d'équations booléennes qui représente le contrôle du programme. Comme il existe des techniques très efficaces basées sur les BDD (Binary Decision Diagrams) pour évaluer symboliquement les systèmes d'équations booléennes, nous pouvons appliquer des techniques de vérification symbolique à nos modèles. L'outil Xeve de l'environnement Estérel a été spécialement développé par A. BOUALI [6] pour la vérification symbolique des programmes synchrones.

6.2. Propriétés vérifiables

Quels types de propriétés peut on établir avec Xeve ?
Tout d'abord des propriétés touchant aux *problèmes d'accessibilité*. On peut demander à l'outil Xeve de calculer le nombre d'états (de contrôle) accessibles. Il peut également calculer une version minimale de cet espace. On peut aussi savoir si un état « dangereux » ne fait pas partie des états effectivement accessibles.
Les propriétés que l'on demande à Xeve de vérifier sont généralement des *propriétés de sûreté* (safety properties). De façon simplifiée, une propriété de sûreté exprime le fait que, quelles que soient les évolutions du systèmes, on n'aura pas de « mauvaises nouvelles », c'est à dire des situations non désirées. Pouvoir prouver ce genre de propriété est naturellement de la première importance dans les systèmes critiques. Dans nos applications, nous distinguons différentes classes de propriétés de sûreté :

- Les propriétés de type *combinatoire*. Elles expriment que pour tout état accessible un prédicat est toujours vrai (par exemple une exclusion mutuelle entre actions).
- Les propriétés de type *séquentiel*. Ces propriétés sont généralement exprimées par des formules de logiques temporelles. Il s'agit, par exemple de montrer qu'un enchaînement dangereux d'actions ne se produira jamais.
- Les propriétés de *réponse bornée* (bounded responsiveness). Il s'agit d'établir qu'à une cause donnée, succèdera sa conséquence avant l'occurrence d'une échéance (timeout). Ces comportements sont typiquement ceux attendus pour des systèmes temps réel.

Les propriétés combinatoires peuvent être exprimées en Estérel. Les propriétés séquentielles et de réponse bornée s'expriment plus aisément en syncCharts.

Pour tester une propriété de sûreté avec Xeve, il faut construire un *observateur* de la propriété et le composer en parallèle avec le modèle de contrôleur à tester. Un observateur est un système réactif qui se contente d'observer les signaux d'entrée et de sortie du contrôleur et émet un signal particulier en cas de violation de la propriété. La propriété est satisfaite si et seulement si Xeve montre que le signal de violation n'est jamais émis. Il est bien sûr possible de tester plusieurs propriétés simultanément. En cas de violation d'une propriété, Xeve génère une séquence conduisant à la violation. Cette séquence est un scénario directement exécutable par Xes. On a donc une aide très précieuse, car les scénarios ainsi créés correspondent à des évolutions auxquelles le testeur n'avait même pas pensé. Il est clair que cette technique peut conduire l'utilisateur à découvrir des anomalies profondément cachées dans ses spécifications ou sa conception.

6.3. Quelques propriétés de l'application

Les bornes d'appel d'urgence ont été conçues pour mettre en contact un usager et le poste de contrôle. Cette mission doit être assurée dans tous les cas. Comme nous avons enrichi les fonctionnalités de ces bornes (cellules de communication) il faut *établir formellement* que la

gestion de ces appels d'urgence est prioritaire par rapport à tout autre traitement. Il s'agit d'une propriété combinatoire que nous avons pu vérifier avec Xeve.
Une propriété séquentielle est qu'en cas de rupture de ligne, les appels vont être reliés par radio (figure 12). Quant aux propriétés de réponse bornées, elles nous permettent de quantifier des temps de réponse, donc concernent des aspects performance.

6.4. Limitations

Xeve semble être un outil incontournable pour les systèmes critiques. Malheureusement, il existe des limitations dans les propriétés analysables. D'abord, Xeve travaille sur des équations booléennes. Lorsqu'on utilise des signaux valués (signaux dont les valeurs sont typées), la compilation produit un ensemble d'équations booléennes pour le contrôle, mais également des fonctions informatiques générales dont l'exécution est déclenchée par le contrôle booléen. Dans une telle situation, Xeve ne sait pas couvrir tous les états, même pour des types de données finis.

La deuxième limitation est due aux équations booléennes elles-mêmes. Malgré l'utilisation des BDD pour calculer symboliquement l'espace des états atteignables, dans certains cas défavorables, on peut dépasser les possibilités de calcul (phénomène d'*explosion combinatoire).*

On peut reculer la première limitation par une *abstraction des données du modèle*. Cette tâche ne peut être faite que par un expert. Elle consiste, par exemple, à ignorer un signal valué qui n'a pas d'influence directe sur le contrôle. Si la valeur des données a une influence sur le contrôle, on peut alors remplacer ce signal par plusieurs signaux purs (séparation de cas). Ces techniques de simplification du modèle ne sont pas encore automatisées. Dans notre application nous avons procédé au cas par cas. Par exemple, le message envoyé par l'usager en cas de problème, n'a pas d'importance vis à vis du contrôle. Par contre il faut savoir qu'il y a un message et d'où il est issus. Nous avons remplacé le signal valué de message d'appel d'urgence par le signal pur `Callk` qui exprime l'occurrence d'un tel message en provenance de la borne numérotée k.

Pour éviter l'explosion combinatoire, il est possible de traiter les sous-ensembles séparément. La difficulté est de savoir si les propriétés locales sont conservées au niveau global. C'est un problème très général et difficile. L'évolution du langage Estérel va dans le bon sens. La nouvelle version du compilateur (v6) autorisera une compilation modulaire. Les techniques de preuves, également modulaires, vont être développées. Elles seront applicables à l'étude présentée dans cet article.

7. Conclusions et perspectives

Dans cet article nous avons proposé une approche synchrone à la conception d'un système de sûreté critique. Les comportements attendus sont exprimés en *syncCharts*, un modèle synchrone graphique. Par des exemples partiels nous avons montré comment ce modèle pouvait décrire des comportements réactifs complexes tout en restant compréhensible par les non-spécialistes en programmation synchrone. Nous avons ensuite expliqué comment *valider la solution proposée*, sans faire la mise en place réelle, qui serait trop risquée. La validation emploie des techniques de simulations sur scénarios de test, mais également des techniques de preuves formelles de propriétés, essentiellement des propriétés de sûreté.

7.1. État des travaux

Nous sommes au début de notre étude. Notre premier objectif a été d'identifier les agents du système, de spécifier leur *comportement* et de valider ces comportements. Dès ces premiers stades, l'approche synchrone se révèle être un excellent outil pour trouver rapidement les erreurs (early bug finding). L'étape suivante, en cours d'évaluation, concerne le *comportement émergent* du système, c'est à dire l'analyse des interactions entre agents. L'analyse portera sur des propriétés fonctionnelles mais aussi des propriétés non fonctionnelles touchant à la sûreté et à certains aspects de performance.
Cette nouvelle approche permet de chercher les limites et de valider la cohérence de dispositifs complexes sans avoir besoin de mettre en œuvre un prototype réel (au sens électronique). Elle permet aussi d'acquérir des informations sur un nouveau système avant de continuer plus loin son étude, et donc de répercuter cette connaissance à un stade du projet où sa prise en compte est possible à faible coût.

7.2. Perspectives

La conception de systèmes complexes est une tâche difficile. L'histoire nous montre un grand nombre de dysfonctionnements lors de l'utilisation effective du système, dans les conditions où il a été conçu. Cela est d'autant plus vrai pour des dispositifs de sécurité qui ne devraient jamais être mis en action dans les conditions extrêmes pour lesquelles ils sont prévus.
Pour être vraiment utiles, les prototypes logiciels que nous proposons, doivent être capables de représenter le système réel à différents degrés de finesse. Les méthodes de validations que nous avons préconisées pourraient être mises en échec par des représentations trop détaillées. Ces représentations étant certainement plus proches de la réalité, il faut être capable de les analyser. Notre approche ne pourra être une contribution aux tests sur la capacité d'action des systèmes complexes que si nous développons des outils et méthodes qui « passent à l'échelle ». Ce souci est partagé par la communauté synchrone : les nouveaux compilateurs sont destinés à des applications « industrielles ». Nous espérons en tirer le meilleur partie sur le problème de sécurité autoroutière qui nous concerne.

Références

[1] ANDRÉ C. Representation and Analysis of Reactive Behaviors: A Synchronous Approach**,** IEEE-SMC, Computational Engineering in System Applications, Lille (F), July, 1996, p 19-29.
[2] ANDRÉ C, BOUFAÏED H, DISSOUBRAY S. SyncCharts : un modèle graphique synchrone pour systèmes réactifs complexes. Real Time Systems 98 (RTS'98), Paris, janvier 1998, Teknea, p 175-195.
[3] AFNOR. Diagrammes fonctionnels Grafcet pour la description des systèmes logiques de commande. Norme Française, NF C 03 190, juin, 1982.
[4] BENVENISTE A., BERRY G. Another Look at Real-Time Programming. Proceedings of the IEEE, vol 79, n 9, Sept, 1991.
[5] BERRY G. Real-Time Programming: Special Purpose or General Purpose Languages. 11th IFIP World Congress, San Francisco (CA), 1989.
[6] BOUALI A. Xeve : an Esterel Verification Environment. CMA-ENSMP, Dec 1997
[7] ESTEREL Esterel v5, disponible sur le site web http://www-sop.inria.fr/meije/esterel
[8] GERONDEAU C**.** Congestion et fluidité. Paris (Fr.) *Revue des Associations des Ingénieurs et Anciens Elèves de l'Ecole Nationale des Ponts et Chaussées.* Décembre 1994. PCM LE PONT N° 12. p33-35.
[9] HAREL D. Statecharts: a Visual Formalism for Complex Systems. Science of Computer programming, vol 8, 1987, pp 231-274.
[10] HAREL D. , PNUELI A. On the development of Reactive Systems in Logics and Models of Concurrent Systems. NATO ASI Series, K.R APT Ed. Springer Verlag, vol 13, 1985, pp 477-498.
[11] SIMULOG, Estérel Studio, visitez le site web http://www.simulog.fr/iesterelf.htm
[12] TANZI T.J, SERVIGNE S, GUIOL R. Real Time Information system for Risk Management on Motorway. CAISE'98: the 10th Conference on Advanced Information Systems Engineering, Pisa, Italy, June 8-12 1998. 11p.