N° d'ordre : 512

THESE

présentée à

L' UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR DE L' UNIVERSITE

par

Béatrice FAYOLLE

Ingénieur HEI

SPECIFICATION ET VALIDATION D'AUTOMATISMES A HAUT NIVEAU DE SURETE DE FONCTIONNEMENT APPLIQUEES AUX TRANSPORTS TERRESTRES AUTOMATISES

Soutenue le 24 AVRIL 1990 devant la Commission d'Examen

Membres du jury:	MM.	R.	GABILLARD	Président
		J.	DEFRENNE	Rapporteur
		F.	PRUNET	Rapporteur
		J.M.	TOULOTTE	Invité
		Y.	DAVID	Invité
		P.	FORIN	Invité

SOMMAIRE

INTRODUCTION

Ce travail de recherche a été fait dans le cadre d'une convention CIFRE (Convention Industrielle de Formation par la Recherche) pour la société Matra Transport. Ce type de contrat offre à l'entreprise, qui désire réaliser un travail de Recherche et de Développement, des collaborations extérieures. Il permet, dans d'excellentes conditions, l'établissement d'une liaison étroite entre une entreprise à vocation strictement définie et un laboratoire compétent.

Les laboratoires ayant soutenu ce travail sont l'INRETS-CRESTA et le Centre d'Automatique de l'USTLFA.

L'INRETS est l'Institut National de Recherche sur les Transports et leur Sécurité, dépendant du Ministère des Transports.

Son laboratoire, le Cresta, Centre de Recherche et d'Evaluation des Systèmes de Transports Automatisés, s'intéresse plus particulièrement à l'automatisation des systèmes de transports. Outre ses activités de certification des systèmes français, elle poursuit des recherches sur le traitement d'images, la transmission immatérielle... Sa compétence dans le domaine nous a conseillé dans le choix des orientations pour mener à bien cette thèse.

Le Centre d'Automatique de l'Université des Sciences et Techniques de Lille-Flandre-Artois a offert par la formation de DEA tout d'abord, puis par l'encadrement de cette thèse, une aide pédagogique pour ce travail.

Cette thèse s'intéresse au développement des automatismes dans les transports terrestres automatisés.

L'automatisation intégrale des systèmes de transport guidé tend à devenir une nécessité dans la mesure où elle apporte un gain important au niveau de la qualité de service (sécurité, rentabilité).

Cette tendance, favorisée par l'évolution technologique, a fait naître un état d'esprit qui consiste à chercher des performances de plus en plus élevées tout en améliorant le niveau de sécurité.

Cela nécessite un effort considérable d'étude et d'analyse de sécurité dès la phase de spécification du système. Pour alléger les études tant au niveau exhaustivité qu'au niveau temps, on est amené à se doter d'outils et de méthodes capables de satisfaire ces exigences. Cependant, on se heurte au choix des outils et des méthodes soit par méconnaissance, soit par manque d'études comparatives. On se contente alors d'utiliser les recettes habituelles par mesure d'assurance.

Il semble utile d'évaluer quelques outils de spécification vis à vis du besoin des transports terrestres automatisés. Cette thèse s'inscrit dans ce cadre.

Dans une première partie, on rappelera les exigences en matière de sûreté de fonctionnement dans les Transports Terrestres Automatisés et on précisera les critères d'évaluation des objectifs de sécurité.

La deuxième partie tentera de définir les caractéristiques demandées à une spécification de qualité. Elle positionnera la spécification dans le suivi de projet et décrira les analyses préliminaires à faire avant d'aborder les modèles. Nous utiliserons dans la partie suivante quelques modèles de spécification (Réseau de Petri, GRAFCET, Automates à Etats Finis). Nous préciserons les avantages, les inconvénients et les méthodes d'utilisation de chacun des modèles. Sur un exemple tiré des automatismes du VAL de Lille, on présentera les outils utilisés pour modéliser et analyser les automatismes fixes.

L'analyse et la validation du modèle seront traitées dans la dernière partie. Elles cherchent à prouver que le modèle formé est sans erreur. Nous préciserons l'intérêt d'utiliser des outils de simulation.

CHAPITRE 1
SURETE DE FONCTIONNEMENT DANS LES TRANSPORTS TERRESTRES AUTOMATISES

1-INTRODUCTION

De plus en plus, on cherche à automatiser les systèmes de transport urbain en site propre afin d'apporter une bonne compétitivité économique et technologique en ce domaine. Le VAL en est un exemple puisqu'il se déplace sans conducteur à son bord .

L'automatisation intégrale a nécessité un effort important dans les études de sécurité du système. La notion d'intégralité signifie que la sécurité des mouvements de train ne repose plus sur le conducteur de la rame. En effet l'homme ne peut surveiller en spectateur le fonctionnement d'automatismes sophistiqués sans immanquablement laisser vagabonder son esprit. Moins il sera sollicité, moins il sera attentif et prêt à réagir rapidement.

Lors d'automatisation intégrale pour les lignes de métro, l'accent a de ce fait été porté sur l'autonomie des équipements en matière de sécurité. L'idée principale est de ne plus demander au conducteur de réagir dans des cas particuliers où un risque se présenterait mais de rendre les automatismes capables de mettre la ligne dans un état de sécurité en cas de problème: c'est-à-dire sans risque d'accident.

La principale difficulté est d'évaluer le niveau de sécurité atteint par le système.

2-DESCRIPTION GENERALE DES AUTOMATISMES DANS LES TRANSPORTS TERRESTRES

Les automatismes gèrent en sécurité les déplacements des véhicules en ligne et dans les zones de garage, ainsi que l'échange des voyageurs lors des arrêts aux stations.

De cette façon, les conducteurs à bord ne sont plus nécessaires et leur poste s'est transformé en poste d'agent d'intervention en ligne. Il est basé sur la notion d'agent itinérant, tant dans les stations que dans les rames, chargé du contact direct avec les voyageurs et chargé d'intervenir chaque fois que nécessaire en actions simples pour maintenir ou rétablir le service. L'accent est porté sur la sécurité et le confort des voyageurs vis à vis des agressions ou de la détérioration du matériel [AMP80].

La surveillance du système est également assurée par des télémesures, des caméras vidéo, l'écoute et la sonorisation des stations.

Ces données sont traitées au Poste Central de Commande (PCC) afin de transmettre les informations nécessaires aux opérateurs. Elles reflètent, en permanence, l'état de la ligne et signalent les équipements en panne.

Les opérateurs ont ainsi connaissance d'éventuelles perturbations. Ils peuvent se mettre en contact à distance avec les voyageurs.

Ces dispositifs de surveillance s'intègrent facilement dans les automatismes.

Les opérateurs du PCC ou itinérants sur la ligne s'assurent de la bonne marche du système et agissent en conséquence. [PER87]

L'automatisme intégral apporte une plus grande souplesse dans l'exploitation. La mise au point de fonctions de plus en plus compliquées permet d'adapter le comportement du système à la demande du client ou à des cas particuliers d'exploitation.

Les tracés de la ligne peuvent être diversifiés. Il est en effet aisé d'automatiser une ligne comprenant une fourche ou un croisement. On cherche ainsi à diminuer le temps de trajet en limitant les changements de ligne. L'augmentation de la cadence des rames est quasi instantanée. Des rames sont en permanence disponibles aux deux bouts de la ligne sans immobiliser de personnel. La capacité offerte peut varier autour du programme nominal en fonction des imprévus (conditions météorologiques, événements sociaux, etc ...).

D'autre part, l'évolution de la société fait que plus on avance dans les développements et plus l'on devient exigeant sur le service fourni. En exploitation, les longues interruptions de service ne sont pas acceptables. Il faut garantir un temps de trajet constant et supprimer les retards. Les causes de panne et de retard seront prises en compte dans les études dès la phase de conception.

Pour prouver la confiance à accorder au système lorsqu'il sera terminé, le concepteur est alors amené à s'intéresser à **la disponibilité, la sécurité et la maintenabilité**. Ces notions de **sûreté de fonctionnement** prennent une part non négligeable dans les études.

Pour garantir un service correct, les pannes et les défauts doivent être corrigés avant qu'ils ne gênent l'exploitation. Ils sont alors à prendre en compte dès les premières phases du projet et pendant tout le développement.

L'étude de la sécurité vise à évaluer les risques d'accidents même en l'absence totale de statistiques globales représentatives. En effet par cette analyse, on cherche à se prémunir contre toutes les causes de risques et pas seulement contre les accidents que l'on a déjà rencontrés. La notion de sécurité est présente à chaque étape depuis la définition du besoin jusqu'à la mise en service et l'exploitation.

La disponibilité et la maintenabilité caractérisent les performances auxquelles répond le système. Elles prouvent son assurance de qualité.

La sûreté de fonctionnement se calcule pour chaque composant du système. Elle aide ainsi à mettre en évidence les parties sensibles de la chaîne, qui demandent des améliorations. Elle vise donc à équilibrer les efforts de qualité et les choix de conception, et à

supprimer les causes de panne fréquente. Il est reconnu qu'un élément fragile dans une chaîne tend à diminuer la qualité de toute la chaîne. Il est par conséquent important d'identifier cet élément et de le corriger.

En proposant un système, le concepteur s'engage à respecter ces objectifs de la **sûreté de fonctionnement.**

3-LES OBJECTIFS DE LA SURETE DE FONCTIONNEMENT DANS LES TRANSPORTS TERRESTRES AUTOMATISES.

Chaque système suivant son mode de fonctionnement et suivant son environnement d'utilisation a des critères qui lui sont propres. A partir de ces données, il se fixe des objectifs de sûreté de fonctionnement à respecter. Ces objectifs ne sont pas des valeurs à atteindre mais des limites à ne pas dépasser.

3-1 DEFINITION DE LA SURETE DE FONCTIONNEMENT [LAP85][VIL88]

La sûreté de fonctionnement (= dependability) d'un système est la qualité du service délivré. Une qualité telle que les utilisateurs du système puissent lui accorder une confiance justifiée. L'objectif est de réaliser un système où la faute, considérée comme naturelle, est prévue et devient tolérable.

La mesure de la sûreté de fonctionnement est définie par:

- la fiabilité (Reliability)
- la sécurité système et utilisateur (Safety and Security)
- la maintenabilité (Maintenability)
- la disponibilité (Availability)

La fiabilité est l'aptitude du système à accomplir une fonction requise dans des conditions données, et pendant un intervalle de temps donné. C'est une grandeur mathématique qui mesure la continuité de service.

v(t) = Probabilité (fonctionnement correct jusque t)

De cette valeur, on en déduit le temps moyen de bon fonctionnement avant la défaillance: MTBF (Mean Time Between Failure).

La mesure de temps de bon fonctionnement sans défaillance catastrophique est la sécurité.

La maintenabilité est la mesure de l'interruption de service ou mesure du temps jusqu'à restauration.

De la fiabilité et de la maintenabilité, on déduit la disponibilité. C'est une mesure de l'accomplissement du service par rapport à l'alternance accomplissement-interruption (temps de réparation).

$$\text{Disponibilité asymptotique} = \frac{\text{Temps Moyen de Bon Fonctionnement}}{\text{MTBF + Temps de réparation}}$$

Tous ces termes (fiabilité, sécurité, maintenabilité, disponibilité), définis ici comme grandeurs mathématiques et variables aléatoires, représentent, dans le sens général, un ensemble de techniques et de méthodes appliqué pour la conception et la validation du système. Elles cherchent à en améliorer l'exploitation. L'ensemble des définitions de la sûreté de fonctionnement est réuni dans l'annexe 1.

3-2 DISPONIBILITE ET MAINTENABILITE

Un système de transport urbain desservant des lieux de travail, des gares ou des aéroports... doit offrir un service très ponctuel, car la moindre perturbation crée des dérangements importants pour les utilisateurs. De plus, vis à vis de l'opinion publique, des interruptions fréquentes du service pendant l'exploitation peuvent entraîner un désintérêt des usagers pour ce mode de transport.

La disponibilité ou la qualité de service prend donc une place importante dans un système tel qu'une ligne de métro.

Il faut distinguer la disponibilité en exploitation, de la disponibilité globale. Pour les usagers, la disponibilité apparente est celle calculée pendant l'exploitation. Le temps de réparation est le temps mis pour restaurer le service après une interruption de trafic. C'est le temps pendant lequel l'exploitation est perturbée. Pour la disponibilité en exploitation, le temps passé à l'atelier pour la réparation complète est transparent.

$$\text{Disponiblité pour l'exploitation} = \frac{\text{temps d'exploitation - } \Sigma \text{ temps de perturbations}}{\text{temps d'exploitation}}$$

La disponibilité globale sert pour l'exploitant. Elle permet, entre autre, de calculer le nombre de rames nécessaires, selon le temps moyen passé à l'atelier par chacune d'elles.

Pour améliorer la disponibilité en exploitation, il faut minimiser, voire supprimer les pannes en ligne. Connaissant l'usure des pièces pendant une durée donnée d'exploitation, on peut prévoir à quel moment la défaillance risque de se produire. On

effectue la réparation ou le contrôle avant qu'elle n'arrive. C'est la **Maintenance Préventive**. Elle est planifiée grâce à des méthodes probabilistes.

Une pièce est remplacée avant qu'elle ne soit complètement hors d'usage et risque d'engendrer une perturbation. Elle coûte ainsi plus cher en heures globales passées à l'atelier et en matériel, mais elle entraîne un gain sur la qualité de service offert pour l'exploitation.

Il faut également noter que la maintenance préventive contribue également à la sécurité. Elle sert, en particulier, à détecter d'éventuelles pannes dormantes, pannes ne provoquant pas de défaillances apparentes mais, qui combinées avec d'autres pannes, peuvent devenir dangereuses.

Pour programmer la maintenance préventive, il faut être capable de calculer le temps moyen de bon fonctionnement. Or les équipements tels que les cartes électroniques ne présentent pas de phénomènes d'usure mais un vieillissement tel, que la maintenance préventive n'a pas de sens. L'usure n'existe qu'à très long terme. Dans ce cas, on utilise la redondance d'équipement. Un deuxième équipement prend le relais dès que le premier tombe en panne. Le temps de réparation, vis à vis de la disponibilité en exploitation, correspond alors au temps de commutation d'un équipement sur l'autre.

La redondance est également utilisée pour des équipements dont la moindre panne provoque une forte perturbation pour l'exploitation. Nous cherchons à éviter les dépannages en ligne et donc les perturbations. La redondance des équipements contribue à rendre le système tolérant aux fautes [HEC79].

Une autre façon d'améliorer la disponibilité est de la prendre en compte dès la phase de conception. Si le système conçu a moins de défauts, les pannes seront *a fortiori* moins nombreuses. Dans la deuxième partie de ce document, nous verrons comment prendre en compte la sûreté de fonctionnement dès l'écriture de la spécification.

3-3 SECURITE

3-3-1 METHODOLOGIE DE LA SECURITE [MAR85]

Nous allons dans ce paragraphe présenter les étapes que l'on trouve dans la méthodologie de la sécurité.

a-Profil de la mission

L'étude de sécurité d'un système commence par la définition du profil de la mission.

Il comprend l'analyse de l'environnement climatique (naturel ou induit), des conditions d'exploitation avec la prévision de trafic, le calcul *a priori* d'une périodicité de maintenance.

b-Définition du système

La description du système doit être faite très tôt afin que chaque intervenant travaille avec la même définition et les mêmes bases.

Elle décrit les modes de fonctionnement correct ou dégradé, les pannes et les accidents.

c-Analyse de sécurité du système

L'analyse de sécurité de l'ensemble du système identifie les situations dangereuses, les accidents potentiels, les éléments dangereux, ainsi que les conséquences qui s'ensuivent. Cette liste a été écrite à partir des bilans des accidents et des incidents d'exploitation. Elle a ensuite été étoffée par des experts ayant une parfaite connaissance du système.

L'analyse détermine la gravité de ces conséquences et surtout définit des règles de conception et les procédures permettant d'éliminer ou de maîtriser les situations dangereuses et les accidents potentiels ainsi mis en évidence. Les conséquences peuvent être classées suivant 4 niveaux critiques:

- C1 - BENIN -sans effet réel sur le service.
  exemple : la panne d'une partie des éclairages de la rame.

- C2 - MINEUR - Modifie le service rendu sans modifier de façon significative l'application.
  exemple : la panne du chauffage de la rame. Elle reste toujours accessible et utilisable mais le service est dégradé.

- C3 - MAJEURE - Modifie le service rendu de telle sorte que des moyens soient nécessaires dans l'application pour pallier l'effet.
  exemple: destruction d'une carte des automatismes fixes qui empêche l'exploitation sur une portion de la ligne et nécessite un mode dégradé d'exploitation.

- C4 - CATASTROPHIQUE - Non tolérée par l'application
  exemple: Déraillement d'une rame de métro pouvant engendrer de la détérioration de matériel, voire des blessés.

Les défaillances C1 à C3 altèrent la disponibilité. Elles ne peuvent pas toujours être évitées. Par contre, des moyens sont prévus pour rétablir un service normal au plus vite.

Les défaillances du type C4 concernent la sécurité. Ce sont celles que l'on cherche absolument à supprimer. Elles entrent dans l'objectif que l'on ne doit pas dépasser. Selon les solutions retenues pour la réalisation des équipements, la sécurité s'exprime *a priori* par un objectif chiffré dont l'ordre de grandeur est de 10^{-9} défaillances catastrophiques par heure d'exploitation.

A l'issue de la définition de ces objectifs, le système dans son ensemble pourra être considéré comme ayant une bonne conception vis à vis de la sécurité. Toute possibilité d'accident aura qualitativement une parade prévue et ceci de façon exhaustive: soit par le passage dans un état de sécurité et de fonctionnement plus ou moins dégradé; soit par un

dimensionnement électrique ou mécanique suffisant pour rendre pratiquement impossible l'accident.

Les étapes suivantes de la méthodologie de la sécurité conduisent à une analyse de sécurité détaillée pour chaque partie du système. Ce sont plus particulièrement ces étapes qui nous intéresseront dans cette thèse.

d-Définition des allocations d'objectifs de sécurité

Pour la sécurité d'un nouveau système de transport, les objectifs que l'on se fixe, doivent être au moins égaux voire meilleurs que ceux des autres réseaux actuellement en service. Les systèmes de transport en France prennent leurs références sur les statistiques d'accidents du métro de Paris, système le plus ancien en service.

Une étude du B.C.E.O.M. ([BCE74] et [GAB74]) fait une analyse comparative des causes et des conséquences d'accidents pour les différents modes. Elle tente ensuite d'en tirer un bilan quant à la connaissance et à la prévention des accidents et à l'intérêt de la typologie d'accident.

Les statistiques du B.C.E.O.M. de 1974 ont permis de déduire des objectifs de sécurité pour le projet VAL. Ceux-ci sont basés sur une évaluation de la sécurité du métro de Paris.

Les types d'accidents sont répertoriés en trois classes [DHA86]:

- **accidents systèmes** : les voyageurs passifs subissent des accidents imputables à des défaillances, à un mauvais entretien du système de transport, à des fautes d'exploitation ou à un environnement anormal.

- **accidents voyageurs-systèmes** : le voyageur est actif, le système provoque un accident du fait de cette action du voyageur.

- **accidents voyageurs** : le système n'est que le cadre du déroulement de l'accident (agression sur des personnes, vandalisme ...).

La dernière classe d'accident n'est pas liée au système et est du ressort de la police urbaine. Elle n'est pas comptabilisée pour l'allocation des objectifs de sécurité.

Pour un système donné, les causes possibles d'accidents collectifs ou individuels sont répertoriées ainsi que tous les équipements entrant en jeux dans la sécurité de cette fonction.

A partir des limites d'accidentés, on déduit pour chaque équipement des objectifs de sécurité sous forme de "Probabilité d'avoir une panne pouvant conduire à un accident par heure de fonctionnement ".

e-Analyse de la sécurité au niveau équipement

L'analyse de la sécurité au niveau équipement est introduite par une analyse préliminaire de risque. A partir de l'analyse sécurité système, on étudie quelles sont les situations dangereuses qui peuvent être amenées par l'équipement considéré [VIL88].

Le cas qui nous intéresse dans cette thèse est celui de l'analyse de sécurité des automatismes. Nous voulons obtenir des fonctions automatisées sans erreur de conception mettant en jeu la sécurité du système. Or le test de la sécurité est impossible à faire sur le site, le nombre de configurations différentes à tester étant beaucoup trop grand. Il faut donc vérifier la sécurité des automatismes conçus dès la phase de spécification. L'analyse préliminaire détaille les risques de la fonction. Elle décrit des critères à respecter lors de l'écriture de la spécification.

A posteriori, une validation de la spécification est faite. Le responsable sécurité [AMP88] passe en revue toutes les causes possibles d'accidents et s'attache:

- à les empêcher de se produire. C'est l'évitement des fautes par une conception sûre du système et par une maintenance préventive. Cette démarche consiste à assurer une "sécurité primaire".

- à minimiser les conséquences corporelles d'un accident (utilisation de matériaux non propagateurs d'incendie par exemple). Il s'agit alors de mesures de "sécurité secondaire".

- à minimiser des blessures (mise en place d'extincteurs proches des zones à plus haut risque: les moteurs, les poubelles et cendriers de stations, etc...).

Pour la conception des automatismes, le responsable de la validation utilise une méthode de sécurité primaire. Il cherche à rendre le système sûr en supprimant les erreurs ou en écrivant des procédures d'exploitation.

f-Règles et précautions

Des règles d'exploitation et de maintenance peuvent êtres déduites de l'analyse de la sécurité. Elles donnent les limites des droits des opérateurs sur le site.
On peut également définir des précautions de fabrication et des essais sur le site.

g-Synthèse des résultats

La synthèse réunit tous les résultats obtenus au cours des analyses. Elle s'assure que l'ensemble des règles d'exploitation et de maintenance, ainsi que les précautions de fabrication et les essais sur le site sont homogènes.
Elle démontre, enfin, que les objectifs chiffrés sont respectés.

h-Le suivi opérationnel

La dernière étape de la méthodologie de la sécurité s'intéresse à l'exploitation du système: le suivi opérationnel.

Les accidents et les incidents d'exploitation sont minutieusement analysés. Ils peuvent dans certain cas remettre en cause des analyses précédentes.

3-3-2 SECURITE INTRINSEQUE ET SECURITE PROBABILISTE

Pour les transports terrestres, la sécurité est traditionnellement assurée de manière intrinsèque, c'est à dire que chaque équipement assure lui-même sa sécurité [GAB79].

Contrairement aux avions qui ne tolèrent aucune panne les empêchant d'atteindre l'aéroport le plus proche, les véhicules de métro peuvent subir des défaillances (perte de la puissance des moteurs, délocalisation ...) à condition qu'elles soient sans conséquence grave pour le système, c'est à dire qu'elles conduisent à un état restrictif, appelé état de sécurité. Cet état est celui où l'on arrête l'exploitation.

Les véhicules de transport terrestre, conçus en sécurité intrinsèque, possèdent donc deux états possibles :

- il n'y a pas de risque et le véhicule est autorisé à se déplacer
- il y a un problème technique, le comportement du système est tel que le véhicule est arrêté et sa protection est assurée par l'immobilisation du système. C'est le fonctionnement en sécurité.

Ce concept est appelé sécurité positive. Il est acceptable pour des systèmes continûment réparables comme les transports terrestres.

On le distingue du concept de redondance (sécurité probabiliste) pour lequel on calcule la probabilité de défaillance catastrophique d'un équipement et on prévoit des redondances jusqu'à atteindre l'objectif voulu. Cette dernière méthode est plus adaptée à l'aviation où l'on cherche à tolérer certains modes de pannes et à restaurer le service. La sécurité positive est impossible pour l'aviation.

En sécurité positive, tous les équipements sont conçus avec deux états: un état de fonctionnement et un état de repos. L'état de repos est l'état de sécurité. Toute panne du composant ou de son environnement doit conduire à cet état de sécurité.

On l'appelle également état de basse énergie. La panne simple qui consiste à perdre l'alimentation conduit dans cet état de sécurité ou de basse énergie.

Cette conception de la sécurité se fait au détriment de la disponibilité. En effet le système n'est pas du tout tolérant aux fautes puisqu'à la moindre panne il s'arrête de fonctionner. Cette disponibilité peut alors être améliorée par la redondance des équipements. Les équipements sont fabriqués en deux exemplaires identiques. L'un des deux est prêt à prendre le relais si l'autre tombe en panne.

La sécurité intrinsèque est démontrée et assurée jusqu'à atteindre l'objectif de sécurité. Il s'exprime en "Probabilité d'avoir une panne non détectée par heure d'exploitation" (PND). Cette probabilité englobe les pannes ne conduisant pas à l'état de sécurité défini plus haut.

4- CONCLUSION

Dans la méthodologie, nous avons parlé des allocations d'objectifs de sécurité liées à des pannes identifiées d'équipements. Pour fixer ces objectifs de sécurité, nous avons considéré que la conception était correcte et sans erreur. Il est évidemment impossible d'estimer la probabilité d'une panne de conception. Il est du ressort du constructeur de s'assurer que son système est correctement conçu et assure donc la sécurité.

Pour les automatismes, il faut vérifier que les fonctions ne peuvent en aucun cas mettre le système en danger. C'est-à-dire que dans toutes les conditions d'exploitation, les automatismes protègent le système contre des risques comme la collision...C'est une méthode de sécurité primaire.

Or la complexité des automatismes ne laisse pas la possibilité matérielle de tester sur le site tous les cas possibles d'exploitation. D'autre part, plus une défaillance est détectée tard, plus la faute est difficile à corriger. En intégration le grand nombre d'équipements mis en jeu rend la faute difficile à détecter et à cerner. De plus l'impact sur l'ensemble du système d'une modification même peu importante est quasiment impossible à appréhender. Elle impose une nouvelle validation complète du système.

Il est donc important d'appliquer des méthodes d'analyse de la **sûreté de fonctionnement** au plus tôt et de valider chaque étape de développement.

Pour le concepteur, les études de sécurité commencent dès la phase de définition des besoins et suivent le système durant sa vie opérationnelle. Dans cet objectif, les analyses fonctionnelles et les spécifications jouent un rôle capital dans la suite du développement du projet en matière de coût et de gestion de projet.

Dans la suite de la thèse, nous nous intéresserons à l'application de la sûreté de fonctionnement dans les premières étapes du projet et plus particulièrement à la spécification des automatismes. Notre objectif est de fournir en fabrication une spécification sans erreur.

On s'attachera à évaluer quelques méthodes de spécification capables de répondre aux exigences spécifiques au milieu des transports terrestres. On évaluera leur efficacité sur l'analyse et la modélisation d'une fonction particulière du métro automatique du type VAL.

Nous regarderons dans le dernier chapitre, ce que ces méthodes nous apportent dans le test des spécifications afin de vérifier qu'elles ne contiennent pas de fautes de conception.

CHAPITRE 2
LES SPECIFICATIONS
EN VUE DE
LA SURETE DE FONCTIONNEMENT

1-INTRODUCTION

Les technologies utilisées actuellement permettent de développer des automatismes de plus en plus performants. Les installations ainsi réalisées ont un degré de complexité qui rend difficile des analyses et des évaluations fondées sur le savoir-faire et l'expérience accumulée.

En effet, il n'est pas possible pour vérifier la sécurité d'une fonction de ne se fier qu'à l'intuition du concepteur ou à l'expérience que l'on a de l'exploitation du système. La variété des disciplines et le grand nombre d'hypothèses à vérifier et à prendre en compte nécessitent, lors de la phase de définition et de conception, une précision et une netteté excluant toute méprise et toute équivoque.

Ceci ne peut être obtenu qu'en s'appuyant sur une méthode rigoureuse de spécification.

Nous écrivons des spécifications de type formel. La spécification est la définition, la détermination des caractères de quelque chose. La notion de caractère formel insiste sur la précision et sur l'utilisation d'un langage évitant les ambiguïtés.

D e plus ces outils de formalisation permettent une simulation des spécifications écrites. La simulation facilite le test de la spécification. L'utilisation de ces méthodes est améliorée par l'emploi d'un support informatique. L'informatique est un bon outil pour la gestion des documents (la spécification et les résultats de tests) et la réalisation des calculs répétitifs.

Plus ces méthodes sont rigoureuses, plus elles manquent de souplesse. Les techniques et les outils utilisés sont donc spécifiques à chaque étape de la spécification.

2-DEFINITION DE LA SPECIFICATION

2-1 LA SPECIFICATION DANS LE PROJET

La gestion de projet a notamment pour rôle d'améliorer la qualité du projet et vise à atteindre à chaque étape les objectifs de sûreté de fonctionnement que l'on s'est fixé. La qualité est applicable à toute la durée de vie de l'équipement [PAR86].

2-1-1 LE CYCLE DE VIE DE L'EQUIPEMENT

Le cycle de vie de l'équipement comporte plusieurs phases. Pour les cas des automatismes, nous proposons la décomposition [BRA88] selon les 4 étapes suivantes:

1- Une phase d'étude préalable, durant laquelle le concept du système est développé. Ceci peut conduire à des études de faisabilité, à des développements de maquettes, ou à des simulations de fonctionnement. Elle doit aboutir à un document que nous appelons **Analyse Préliminaire** du système, c'est-à-dire à la description des services attendus, des contraintes et des performances à satisfaire, et à l'ébauche de l'architecture en sous-ensemble. C'est la description du problème.

2-Une phase de développement, qui commence à la rédaction de la **spécification fonctionnelle**. Celle-ci consiste à mieux formuler et à préciser les services, les contraintes et les performances à satisfaire. C'est à partir de cette spécification que sera faite la conception du système puis la réalisation; en parallèle on procède à l'analyse et à la vérification du sous-ensemble par rapport à sa spécification (test de validation) puis à sa conception (test d'intégration). Cette spécification peut être rédigée suivant une méthode naturelle, semi-formelle, ou formelle.

3-Une phase d'intégration et de validation, qui a pour but de regrouper les divers sous-ensembles développés, d'en vérifier la bonne coordination et de s'assurer que le système offre bien dans son environnement les services voulus, conformément à ses spécifications.

4-Une phase de mise en service et de support du système, qui regroupe les activités de formation, d'assistance aux utilisateurs, de suivi de problèmes d'exploitation, de réparation ou de maintenance...

Dans cette étude, nous nous intéresserons à l'écriture de l'analyse préliminaire et de la spécification fonctionnelle et formelle ainsi qu'à leur validation.

2-1-2 LA QUALITE DU PROJET

L'assurance qualité du projet est une activité présente dans toutes les phases de vie du projet. Elle assure d'abord que les divers intervenants possèdent les mêmes normes et les mêmes documents. Ces dispositions doivent également être prises en ce qui concerne les outils d'aide à la conception et à la réalisation [TROY86].

Elle a un regard sur l'avancement du projet par un contrôle permanent de ce qui est conçu ou modifié au cours de la réalisation du produit. Elle est chargée de vérifier qu'aucune action ou manque d'action ne s'oppose à la bonne marche du projet.

Les fournitures que le constructeur doit mettre en place pour satisfaire les exigences de la qualité, sont les suivantes [PAR86]:

-a- description des spécifications fonctionnelles contenant :

- un plan d'assurance qualité
- un plan de vérification

-b- description des spécifications formelles

-c- plan de tests détaillés

-d- dossier de fabrication

-e- rapport de qualité, sécurité, fiabilité...

-f- manuel utilisateur et de maintenance

Nous pouvons remarquer que les premiers documents à mettre en place sont la spécification fonctionnelle suivi de la spécification formelle. Ils serviront de documents de travail de base lors de l'avancement des développements et leur qualité joue un rôle important pour la suite du projet.

En effet, une erreur contenue dans la spécification sera reportée dans toute la suite du projet. D'autre part, s'il est aisé de trouver la cause d'une erreur dans une spécification, il sera beaucoup plus difficile de mettre cette cause en évidence lorsque l'on intègre et lorsqu'on met en jeu de nombreux équipements.

C'est pourquoi il est nécessaire de tester ces spécifications et de les corriger avant de passer à la phase de fabrication.

Dans ce travail, nous nous intéressons essentiellement à ces deux premières phases du cycle de développement dans le but d'analyser et d'évaluer des méthodes de modélisation pour améliorer la sûreté de fonctionnement.

Nous verrons d'une part, la définition et l'écriture des spécifications formelles, puis dans une troisième partie, la validation au plus tôt de ces spécifications.

2-2 PRESENTATION DE LA SPECIFICATION

2-2-1 QU'EST-CE QU'UNE SPECIFICATION

Par spécification, nous entendons ici uniquement la description précise des fonctionnalités d'un système. Nous distinguons deux types de spécifications ([GAU88] et [DEC88]).

Les spécifications **semi-formelles** sont habituellement exprimées à l'aide d'une notation graphique annotée en langage naturel et respectant un certain nombre de règles syntaxiques.

Les spécifications **formelles** sont exprimées à l'aide d'une notation complètement définie, précise et non ambiguë (un langage, un graphisme, un tableau de chiffres...). Cette notation a non seulement une syntaxe mais aussi une sémantique. On doit donc pouvoir décider par un raisonnement rigoureux si une mise en œuvre satisfait ou non une spécification.

Ces deux spécifications peuvent être différenciées dans leur domaine d'application.

La **spécification semi-formelle** est souvent utilisée dans l'industrie lors des phases de conception de haut niveau. Elle permet de décrire l'ensemble du problème de manière hiérarchique. Cette approche assez ancienne connait un regain d'intérêt consécutif à l'avènement de nombreux environnements capables de la supporter (écran graphique, souris, amélioration de la puissance de calcul, outils pour générer l'ensemble de la documentation...)

La syntaxe et la sémantique de certaines parties des notations sont souvent floues. Une telle spécification, destinée à un public assez large, est d'une lecture relativement aisée.

Elle donne une vision globale du système pour contrôler certaines propriétés telle que la cohérence des informations échangées.

La **spécification formelle**, plus difficile d'utilisation, est souvent déconnectée des phases de définition des exigences et de conception. Elle est d'un niveau macroscopique et ne permet de spécifier qu'un aspect du problème.

Sa syntaxe et sa sémantique précises sont destinées à un public averti et formé qui seul peut comprendre ce qui est demandé.

Elle conduit le responsable de la spécification à être plus précis et donc à décrire le système au plus haut niveau. Elle doit être suffisamment fine pour décrire avec précision certains mécanismes délicats tels que les interactions et les conflits élémentaires entre différentes actions du système. La spécification formelle est également appelée modèle.

La spécification formelle comme la semi-formelle modélisent et expriment, chacune à leur niveau, les propriétés qu'on attend du système. Le travail effectué peut ainsi être vérifié et validé. C'est pourquoi de nombreux responsables des spécifications éprouvent le besoin de mettre sur pied une méthode de conception de systèmes, d'étudier des modèles mathématiques permettant de se convaincre, ou mieux de prouver certaines propriétés des systèmes modélisés, et ainsi, d'estimer certaines grandeurs soit par calcul, soit par simulation.

Les spécifications semi-formelles diminuent globalement le coût de réalisation d'un projet. Elles augmentent les coûts de conception détaillée mais diminuent sensiblement les coûts de fabrication, d'intégration, de documentation et en particulier de maintenance. L'utilisation des méthodes semi-formelles est facilitée par les outils informatiques.

Il n'est intéressant d'utiliser la spécification formelle que pour une petite partie d'un projet et en particulier dans les fonctions très critiques. Celle-ci est difficile d'approche et nécessite une maîtrise parfaite de la méthode.

2-2-2 LA QUALITE DE LA SPECIFICATION

Une bonne spécification doit répondre aux critères de qualité que nous énumérons ici [BRA88].

La principale qualité d'une spécification est d'avoir été écrite au début du projet et mise à jour au fur et à mesure de l'évolution du projet. Il est difficile de vérifier que la réalisation est conforme à ce qui est demandé si cette demande n'a pas été écrite et exprimée clairement.

Avoir un système qui marche en mode nominal ne prouve pas qu'il est capable de réagir à des agressions de l'environnement. Si on ne définit pas précisément les conditions de fonctionnement et les erreurs qu'il doit tolérer, on ne peut pas le protéger contre ces

agressions. La spécification doit donc être **complète** et **conforme au besoin** tout en restant concise. Elle exprime le "quoi" mais non le "comment". Il ne faut pas entrer trop loin dans le détail et il faut définir la frontière avec la conception.

La spécification doit être **cohérente**. Il est difficile de réaliser un produit à partir d'une spécification qui contient des informations contradictoires. Pourtant, il est fréquent de trouver des contradictions dans une spécification de volume important et souvent rédigée par plusieurs personnes. C'est également une conséquence possible des modifications apportées à un document au fil du temps.

Enfin la spécification doit être **significative** et doit pouvoir exprimer de manière **précise** le besoin. Par exemple, pour un système de transport en commun dans une ville, le besoin dépendra du nombre de personne à transporter. Il faut chiffrer, de la façon la plus précise possible, les prévisions d'utilisation de ce système de transport pour pouvoir faire le choix entre un moyen de transport personnalisé, un réseau de bus ou une ligne de métro.

2-2-3 LES TYPES DE SPECIFICATION FORMELLE

La spécification formelle peut être du type [GAU88]:

- **axiomatique**. Elle décrit l'ensemble des propriétés que doit satisfaire la mise en œuvre. Elle est basée sur des axiomes.

- **relationnelle**. La spécification représente un ensemble d'objets et de relations entre ces objets. Des règles décrivent la façon dont cet univers peut évoluer suite à certaines actions.

- **opérationnelle**. Cette spécification décrit les actions à réaliser au moyen d'algorithmes ou d'autres systèmes de transition. Mais elle ne décrit pas la mise en œuvre du système. C'est en quelque sorte une "spécification par l'exemple". Elle se prête au prototypage.

Le choix du type de spécification va essentiellement dépendre du niveau de complexité de la fonction à définir. Il est difficile d'avoir une spécification à la fois détaillée et facilement transmissible.

Une fonction simple sera définie de façon axiomatique. Une fonction compliquée, dont on ne connait pas *a priori* tous les états possibles, devra être testée et sera plus facile à aborder avec une spécification opérationnelle.

C'est pourquoi plusieurs outils de spécification seront utilisés, chacun étant adapté à la fonction qu'il spécifie.

2-2-4 LE TEST DES SPECIFICATIONS FORMELLES

Les spécifications formelles du type opérationnel peuvent devenir exécutables. Une spécification est exécutable si elle est exprimée dans un langage pour lequel il existe un interprète. Le modèle spécifié possède plusieurs états. En simulant des commandes en entrée,

il est possible d'obtenir, par cet interprète, le passage d'un état à un autre. La spécification peut calculer les sorties du système lorsque ses entrées évoluent. Cette spécification est dite dynamique.

Si une spécification formelle est exécutable, elle permet d'obtenir rapidement un prototype ou une maquette du système à développer.

Le prototype simule le comportement du système. Il est alors possible de voir comment évolue le système dans des configurations bien particulières et pour un grand nombre de cas possibles.

Après l'écriture de la spécification, une analyse est nécessaire pour vérifier et pour prouver que le modèle est sans erreur. La simulation est une aide précieuse pour cette mise au point. L'analyse définit une série de tests. Ces tests sont déroulés dynamiquement sur le prototype. La simulation est souvent lourde en calcul et n'est acceptable que si elle est automatisée.

La figure 2-1 situe la spécification et le prototype par rapport à la phase de conception et de développement. Le prototype est développé depuis la spécification formelle. Un jeu de tests sera exécuté sur ce prototype.

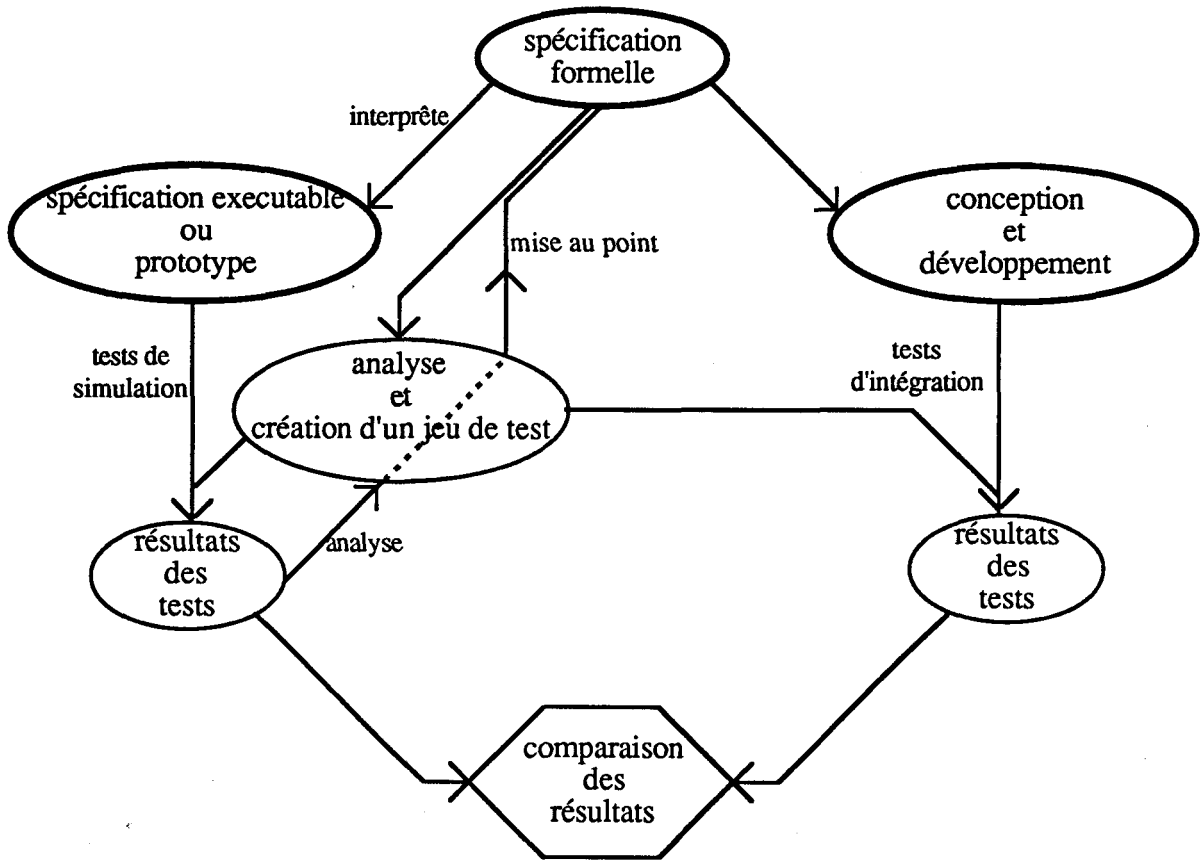


FIGURE 2-1

La spécification est validée par expérimentation de son comportement sur un ensemble de jeux de tests. Ceci contribue à la rendre sûre de fonctionnement.

L'objectif est de détecter au plus tôt les erreurs de spécification pour ne pas les répercuter dans tout le développement du projet. Plus une erreur est détectée tard dans le projet et plus elle sera difficile à identifier et à corriger.La conception ne doit être faite qu'à partir d'une spécification testée, modifiée et validée [MOA76].

Ce même jeu de tests pourra ultérieurement être exécuté sur le système développé. Les résultats des tests sur le système développé seront comparés à ceux obtenus par le prototype. Par cette comparaison, on montre que le développement correspond au prototype. On valide donc la chaîne qui passe de la spécification formelle à la conception et au développement.

Il faut distinguer spécification formelle et prototype. La spécification est une description cohérente et complète du système alors que le prototype n'en est qu'une approximation. Il peut être limité aux parties critiques que l'on veut mieux surveiller. Seule la spécification peut servir de document de référence pour le constructeur.

Le prototype n'est pas un point de départ pour la conception. Il est parallèle. Le développement pourrait être fait à partir du prototype s'il existait une fonction de transformation de ce prototype en dossier de conception.

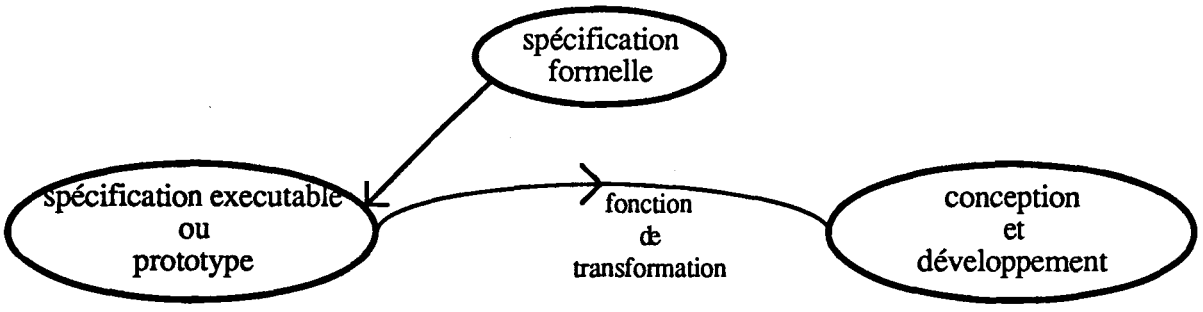


FIGURE 2-2

Pour l'écriture de cette fonction de transformation, deux problèmes se posent. Tout d'abord le prototype n'a pas nécessairement toutes les informations pour la conception. La prise en compte des autres critères (performance, mise en œuvre) peut difficilement être explicitée dans la phase de construction du prototype.

Il est ensuite difficile d'optimiser la conception lors de la transformation automatique ou assistée du prototype. En effet les regroupements et les découpages en sous-systèmes, ainsi que les problèmes de décision, sont difficiles à automatiser.

Par contre la fonction de transformation, une fois validée, nous affranchirait des problèmes qui peuvent se présenter lors de l'interprétation de la spécification pour la conception.

3-ANALYSE PRELIMINAIRE A L'ECRITURE DE LA SPECIFICATION

Pour obtenir une spécification sûre, outre la validation de la spécification, l'analyse préliminaire réunit les informations nécessaires au responsable de la spécification pour bien concevoir son équipement. Elle vise à éliminer les erreurs dues à l'oubli, la méconnaissance ou la mauvaise compréhension de la demande.

C'est la description du problème. Elle dit ce qu'on veut faire et comment. Elle liste les fonctions que devra réaliser l'équipement, les critères à respecter et les outils qui seront utilisés.

Le responsable de la spécification dispose alors de toutes les informations pour écrire sa spécification dans les meilleurs conditions et sans impasse.

Dans cette partie, nous illustrons nos propos par deux exemples complémentaires. Ils sont basés autour de fonctions d'un métro entièrement automatisé. Le premier traitera de la logique de canton d'une ligne alors que l'autre aura pour but de gérer le déplacement de véhicules dans une zone de communication.

Le sens des termes techniques est détaillé dans le glossaire à la fin du document après les annexes.

3-1 QUE VEUT-ON SPECIFIER ?

Ce paragraphe doit définir ce que nous voulons spécifier. Il limite exactement le domaine de la spécification. Le fonctionnement de notre équipement une fois terminé et les performances attendues y sont expliqués.

Dans l'exemple 1 est spécifié la logique de canton d'une ligne de métro. Celle-ci est divisée en plusieurs zones appelées tronçons. Chaque tronçon est lui-même découpé en cantons. Sur chaque canton, il ne peut y avoir qu'une seule rame à la fois. La logique doit suivre les mouvements des trains sur un canton tout en évitant les collisions de rames par rattrapage.

Le découpage de la ligne se fait selon la figure 2-3:

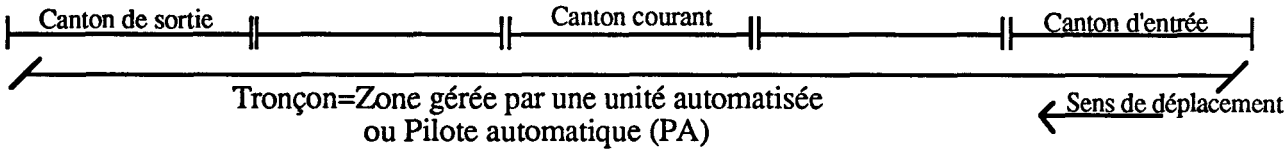


FIGURE 2-3

L'exemple 2 traite d'une zone de rebroussement en ligne implantée comme dans la figure 2-4:

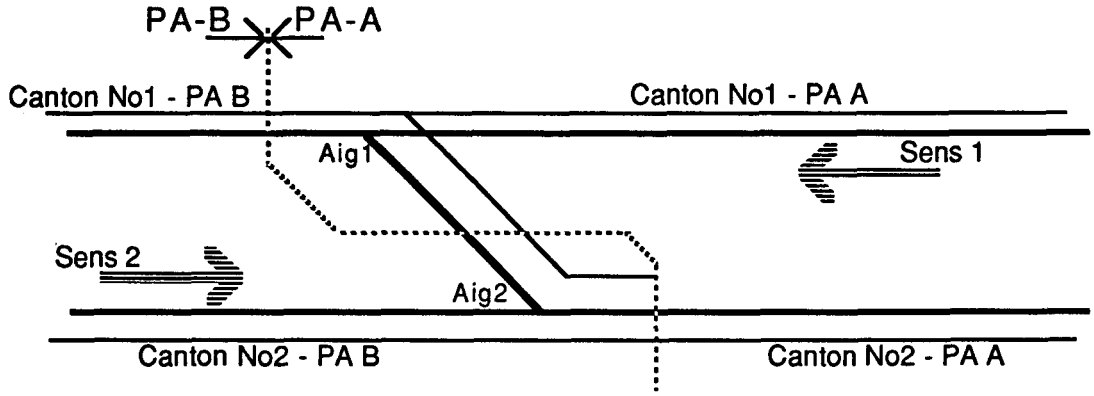


FIGURE 2-4

L'exploitant veut utiliser cette zone de deux façons :

- Les rames se déplacent normalement suivant le sens 1 pour la voie d'en haut, et suivant le sens 2 pour la voie d'en bas. C'est le service normal.
- Une rame arrivant du canton 1 du PAA, emprunte la communication pour se diriger vers le canton 2 du PAA. Elle ne passe pas par le PAB. C'est le service avec rebroussement.

La fonction à spécifier doit commander les déplacements du train dans une communication. Ces déplacements dépendront du choix de l'exploitant. La fonction commande également les deux aiguilles suivant l'itinéraire qu'elle met en place.

3-2 AVEC QUOI VEUT-ON SPECIFIER ?

Il faut donner au responsable de la spécification tous les moyens disponibles pour décrire la fonction. Ce sont tout d'abord les interfaces avec les autres équipements. On lui liste les signaux envoyés à l'équipement et sous quelle forme ils sont fournis.

Si le découpage en fonctions a été complètement fait, cette liste peut être complète dès l'écriture de l'analyse préliminaire.

Cette liste de signaux en interface nous donne également les informations envoyées par les capteurs.

Dans les deux exemples, les moyens pour réaliser la fonction sont tout d'abord la reconnaissance des véhicules. Les véhicules ont des antennes émettrices appelées Emetteurs Anti-Collision (EAC). Ces émissions sont détectées au niveau du sol par des boucles appelées Détecteurs Positifs (DP). Ce signal est à 1 si un EAC émet juste au dessus de la boucle.

Une autre sorte de détecteur placé au sol détecte le passage des rames. Ce sont des barrières du type faisceaux lumineux qui sont occultés par les trains. Ceux-ci sont appelés Détecteurs Négatifs (DN) et sont à zéro lors du passage du train, même si celui-ci n'a pas d'EAC.

On commande également le déplacement des véhicules par des boucles au sol. Un train ne pourra se déplacer que si la boucle sous le train est alimentée. Ces boucles sont appelées FS.

Pour l'exemple 1, il n'a été défini à ce niveau aucune interface avec les autres équipements. Cette unité à gérer est autonome. La spécification déterminera si il faut des télécommandes envoyées par l'exploitant.

Pour la communication, la zone est surveillée par la logique de canton classique. La fonction peut disposer de toutes les informations de cette logique. Elle dispose également de deux commandes vers chaque aiguille (commande à droite et commande à gauche) et de deux télémesures (contrôle à droite et contrôle à gauche).

3-3 SUIVANT QUEL OUTIL, VEUT-ON SPECIFIER ?

L'analyse préliminaire décrit le contenu et la forme de la spécification. Elle doit définir le langage d'écriture de cette spécification. Le langage choisit doit avoir des qualités de mise en œuvre [TRO84]. Il dépend du matériel de fabrication puisque la spécification doit pouvoir se traduire en dossier de fabrication. Ce langage dépend également des outils de validation qui peuvent être utilisés. Il peut être très différent du produit fini. Nous allons dans la suite étudier différents langages de spécification que nous décrivons à ce moment-là.

3-4 CRITERE DE SECURITE A RESPECTER OU ANALYSE PRELIMINAIRE DE RISQUES

L'analyse préliminaire de risques sert à mettre en évidence les risques contre lesquels il faut se protéger. Elle identifie les situations dangereuses et détermine les conséquences qui s'ensuivent ainsi que leur gravité. Cette analyse aide à concevoir le système tout en conservant la notion de sûreté [BEO88].

La plupart des techniques employées actuellement pour l'évitement des fautes les éliminent selon leur type. Elles ne permettent pas de porter un effort plus important sur les fautes conduisant à des défaillances catastrophiques puisqu'elles ne prennent pas en compte les conséquences des défaillances mais seulement leur origine.

Il parait donc nécessaire de se doter d'une méthode d'analyse de la sécurité ou de la disponibilité permettant l'identification des fonctions critiques.

3-4-1 FONCTIONS CRITIQUES

La première étape de cette analyse est de mettre en évidence les fonctions critiques. Celles-ci sont connues par l'expérience et par l'intuition. Reprenons l'exemple de l'automatisation d'une ligne de métro.

L'expérience dans l'exploitation d'un métro, même non automatisé, et la connaissance et l'analyse des incidents permettent d'énumérer un certain nombre de risques contre lesquels on veut se protéger. Mais l'automatisation apporte de nouvelles données. Et ce n'est pas parce qu'on n'a jamais rencontré un accident dû à une certaine panne, que le risque n'existe pas.

On veut se protéger contre les risques même peu probables. On utilise alors la méthode intuitive. L'imagination à la fois du concepteur et du "valideur" intervient pour définir des risques. Tout spécialiste connait bien les risques de sa spécialité, dit-on souvent.

Il n'en est rien: des spécialistes interrogés fournissent effectivement une liste de risques, mais elle est très rarement exhaustive. De plus, de nombreux risques sont estimés à tort comme sans importance ou sans conséquence grave. Cette mégarde est souvent due au fait que les concepteurs se limitent au domaine très restreint de leur spécialité et ne voient pas les conséquences possibles sur l'environnement.

Cette étude de risques est donc pilotée par des spécialistes de la validation en collaboration étroite avec les concepteurs. A partir des modèles de pannes de composants, on peut, en les combinant entre eux, définir la conséquence sur le système et mettre en évidence un nouveau risque.

Enfin les fonctions critiques identifiées seront classées suivant leur niveau défini dans la première partie (bénin, mineur, majeur, catastrophique).

Ces risques étant identifiés, le concepteur devra s'en protéger dans tous les cas, à la fois dans l'état normal du système mais aussi lorsque des pannes surgissent. L'analyse préliminaire cherchera à énumérer des critères de conception afin de se protéger contre ces risques. Ces critères serviront de document guide pour l'écriture de la spécification. Le responsable de la spécification cherchera à réaliser sa fonction pour pouvoir exploiter mais aussi pour respecter ces critères et donc se protéger contre les risques. On vise à écrire une spécification sûre.

a-Exemple de la logique de canton

Le principal risque contre lequel on veut se protéger est la collision entre deux rames.

La ligne est découpée en cantons. On ne veut qu'une seule rame sur chaque canton. Pour respecter ce critère, il faut que chaque rame soit reconnue, et que l'on sache en permanence si un canton est occupé ou non.

Ces critères peuvent s'écrire de la façon suivante:

1- Une rame doit toujours être correctement reconnue par la logique.
   a- Elle doit pour cela occuper au moins un canton. Elle ne peut libérer le canton d'où elle sort que si elle occupe déjà le suivant.
   b- Il faut détecter les déplacements des rames sans Emetteurs Anti-collision (EAC) puisque ce sont ces derniers qui par émission sur les boucles indiquent les déplacements des rames.
2- Conséquence du critère 1b, il faut détecter les rames qui perdent leurs EAC.
3- Un canton ne peut être occupé que par une seule rame.
   a- Il faut détecter les rames qui pénètrent sur un canton déjà occupé.
   b- Il faut détecter les rames qui reculent d'un canton sur l'autre.
   c-Au début de l'exploitation, on doit savoir dans quel état on prend la ligne (absence ou présence de rame). Cette logique doit donc avoir un état initial qui respecte le critère n°1.

b- exemple de la zone de rebroussement

En plus du risque de collision par choc frontal, on met en évidence le risque de collision par prise en écharpe (dans une zone de communication, collision entre deux rames

venant de deux voies différentes), le risque de bi-voie (dans une communication, une rame engage un essieu dans une voie et l'autre dans une voie de direction différente) et le risque de déraillement.

Ceci nous amène à définir les critères suivants:
1- Il ne faut pas bouger une aiguille sous une rame.
2- Il ne faut pas envoyer une rame vers une aiguille mal positionnée.
3- Une rame doit toujours se déplacer vers une zone libre et qui le restera tout au long de son déplacement.

Enfin pour les deux exemples, il faut se protéger contre les actions humaines préjudiciables à la sécurité. Pour les automatismes, ce sont essentiellement des erreurs d'exploitation. On considère qu'un opérateur bien formé peut toujours se tromper lors de l'envoi d'une télécommande. Ce type d'erreur ne doit pas aller à l'encontre de la sécurité du système. Par contre, on ne prend pas en compte les erreurs multiples n'ayant pas de causes communes, ni les conducteurs "fous".

3-4-2 MODE DE PANNE

Pour prendre en compte cette analyse préliminaire, il faut connaître les modes de panne du matériel utilisé.

Dans notre étude d'automatisation d'une ligne de métro, nous nous basons sur le concept de sécurité intrinsèque: "Tout composant ne peut pas être dans un état actif sans qu'on lui ait commandé" (voir la première partie). Ce principe implique qu'on connaisse l'état actif et l'état inactif du composant.

Prenons comme exemple les Détecteurs Négatifs. Ces barrières de faisceaux lumineux sont généralement placées aux entrées et sorties d'une zone. Leur but est de détecter le passage de n'importe quel type de rame. Pour la sécurité, aucun train ne doit être invisible vis à vis de ces barrières. Le cas de panne admis est DN=0. On n'aura jamais DN=1 alors qu'un véhicule passe devant la barrière.

Pour les DPs, l'état de sécurité est DP=0. En effet la détection positive DP capte un signal émis par les antennes de la rame. L'état de sécurité est l'état où il n'y a plus d'énergie, c'est-à-dire que le signal n'est pas reçu.

De même pour le pilote, l'état de panne des interfaces et des automatismes est l'état 0. Une panne matérielle ne force pas un signal à 1.

Ce document d'analyse préliminaire définit ce qu'on veut faire, comment le faire et dans quelles conditions. Il est approuvé par toutes les parties: responsables de la spécification, concepteurs et spécialistes de validation. Il sert enfin de document de base pour l'écriture de la spécification. Bien qu'écrit en langage naturel, il a le mérite d'être complet et de s'être intéressé aux risques contre lesquels on veut se protéger.

C'est un document qui doit évoluer au cours de l'avancement du projet en particulier si de nouveaux risques sont mis en évidence. Il reste un document très général et peut être utilisé dans d'autres projets.

Cette analyse préliminaire est donc un bon outil pour l'écriture et la validation d'une spécification sûre.

4-ECRITURE D'UN MODELE DE DECOMPOSITION SEMI-FORMELLE: SADT, ET DE REPRESENTATION FORMELLE AU PLUS BAS NIVEAU: AUTOMATES A ETATS FINIS.

La méthode que nous allons analyser ici est une méthode d'aide à la spécification des systèmes. Celle-ci a deux supports: d'une part la décomposition hiérarchique, d'autre part une méthode semi-formelle et dynamique.

La première étape de travail consiste à décomposer la fonction en sous-fonctions puis en sous-sous-fonctions. La spécification est éclaircie car elle est divisée en petites unités de spécification. La méthode utilisée est la méthode SADT de type semi-formelle: Structured Analysis and Design Technique. Elle nous amène à faire un découpage hiérarchique de la fonction. Elle a l'avantage de gérer les interfaces entre les sous-fonctions de même niveau.

Au plus bas niveau de la décomposition, la spécification est modélisée suivant le principe des Automates à Etats Finis. L'ensemble des Automates à Etats Finis, complété des interfaces entre les sous-fonctions, forment un modèle formel de toute la spécification.

Nous avons expérimenté cette fonction sur l'outil ASA. Cet outil fait évoluer dynamiquement le modèle formel.

Pour la spécification de nouvelles fonctions, il est actuellement un des plus utilisé dans le domaine des Transports Terrestres Automatisés

4-1 DECOUPAGE HIERARCHIQUE

La méthode consiste à modéliser la structure et le comportement d'un système sous forme de décomposition hiérarchique descendante de plus en plus précise.

Le point de départ est la décomposition hiérarchique du système sous forme de modules. Cette représentation modélise à la fois les **objets** (données, documents...) et les **activités** (réalisées par des hommes, des programmes ou des mécanismes). C'est un langage graphique qui permet de montrer les modules, leurs relations et leur intégration dans une structure hiérarchique.

Dans le cas général, un module est associé à une activité et on représente celle-ci par une boîte rectangulaire. Cette activité sera décomposée en d'autres sous-activités de niveau de détail de plus en plus fin et ainsi de suite jusqu'aux modules terminaux de fonctions élémentaires.

Une boîte comporte comme interfaces:

- Des entrées d'état (à gauche dans la représentation graphique): ce sont des données utilisées par une activité.

- Des entrées de contrôle (au-dessus dans la représentation graphique): elles précisent comment et en fonction de quoi se déroule l'activité. Ces entrées sont souvent écrites sous forme de messages fugitifs qui conditionnent l'évolution de l'automate à états finis. Ces messages sont consommés par l'automate.

- Des sorties: ce sont des données produites par l'activité.

La simplicité de la structure d'un modèle dépend à la fois des interfaces entre deux niveaux et des regroupements des fonctions de même niveau.

Nous avons choisi de traiter l'exemple de la logique de canton. Cette fonction gère le suivi des trains sur une portion de voie d'environ 30 mêtres. La ligne a été découpée en plusieurs zones géographiques appelées tronçons. Chaque tronçon est découpé en cantons. Nous conserverons ce découpage pour spécifier la fonction de la logique de canton. Nous distinguerons trois types de canton: le Canton d'Entrée de tronçon; le Canton de Sortie de tronçon; le Canton Courant ou Canton Intermédiaire.
L'implantation des boucles pour cette fonction est expliquée dans le glossaire.

Le schéma 2-5 reproduit la décomposition hiérarchique de la logique de canton. Cette fonction a été découpée en autant d'étapes de calcul qu'il y a à faire. Pour les interfaces avec l'extérieur (Partie Commande), un module PCC a été décrit.

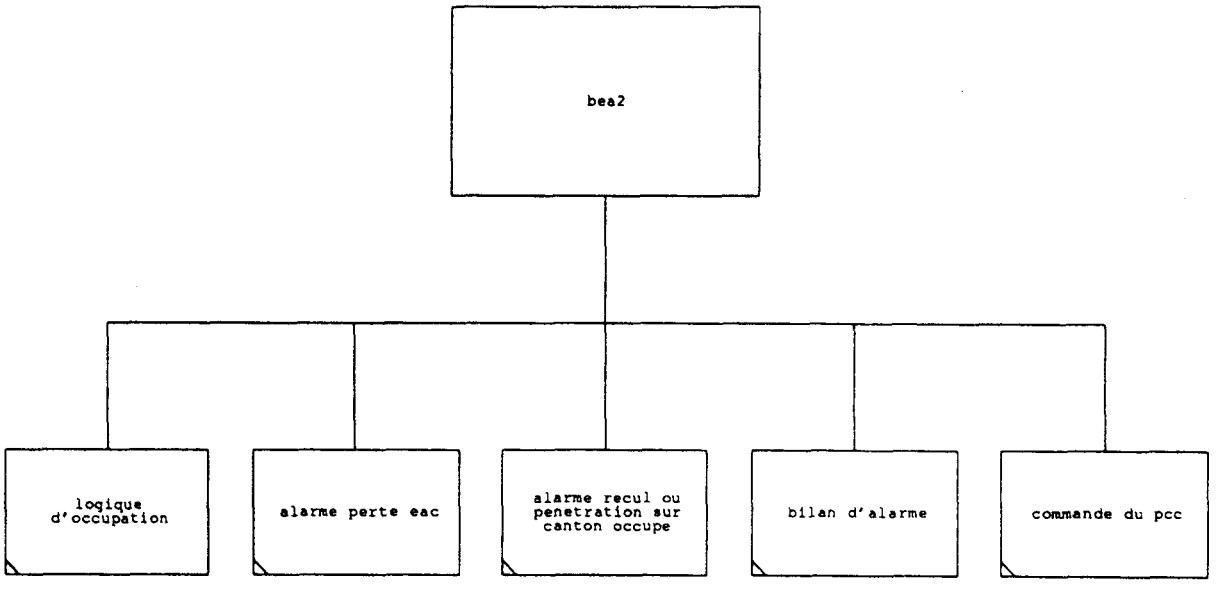


FIGURE 2-5

La figure 2-6 décrit les communications qu'il y a entre les fonctions du deuxième niveau.

Le découpage choisi a été fait en trois niveaux. Il n'est pas intéressant de trop multiplier les niveaux, car le suivi des informations deviendrait très difficile. Il ne faut pas passer à plus de cinq niveaux de hiérarchie.

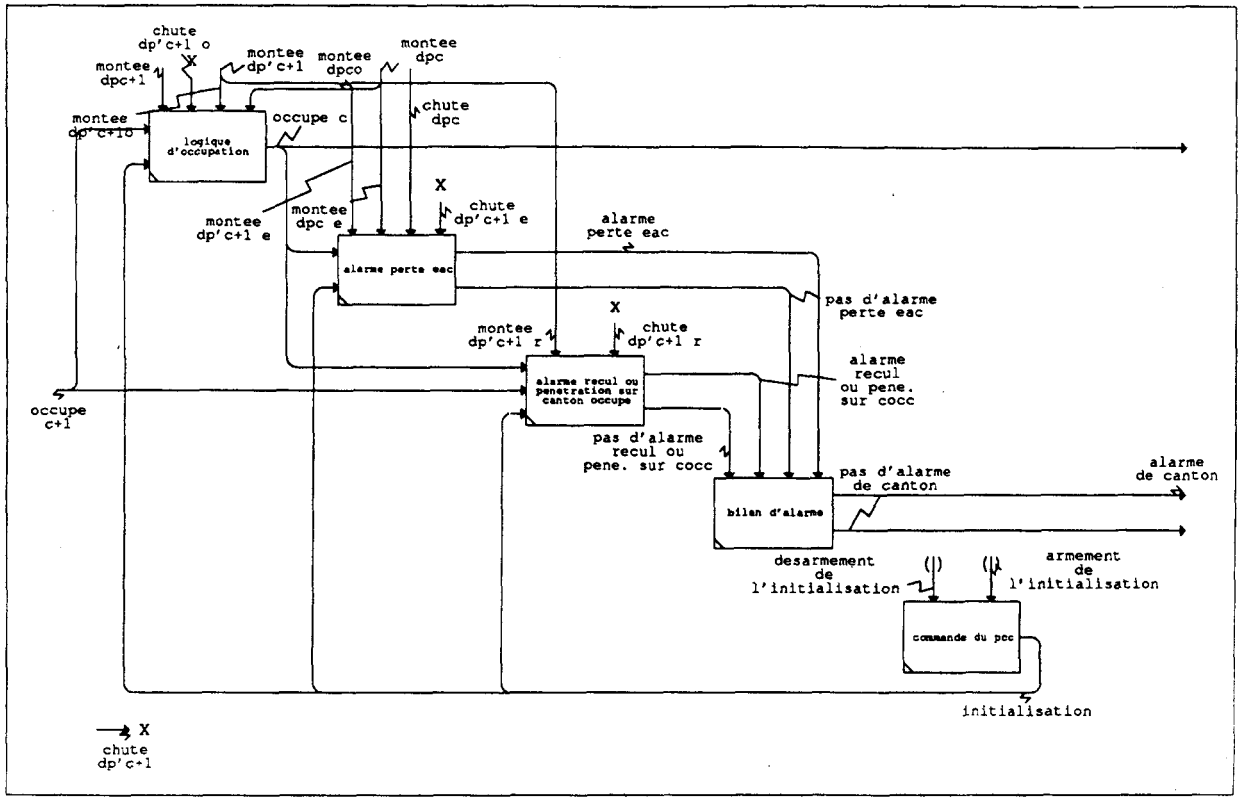


FIGURE 2-6

Pour s'assurer que la simulation peut tourner, il faut vérifier que les interfaces sont homogènes. Le tenant et l'aboutissant d'une interface doivent être de même type. Dans notre exemple, le signal "occupation c" à la sortie de la boîte "logique d'occupation" est un booléen. Il est redistribué aux boîtes d'alarme sous le même nom. C'est une donnée que peut lire la fonction.

Les signaux du type *"montée d'une boucle"* indique qu'une rame commence à émettre sur cette boucle. Inversement le signal *"chute nom d'une boucle"* indique que la boucle ne recoit plus.

Le signal "montée DP'_{c+1}" est du type message. C'est un signal de ce type qui active les Automates à Etats Finis. La méthode ne permet pas la simultanéité dans l'envoi des messages. Ce message se divise donc en trois messages différents vers chacune des boîtes d'alarme et de logique d'occupation. Ceci permet de tester l'influence de l'ordre d'arrivée des messages sur la logique spécifiée. On retrouve donc à l'extrémité de cette interface, trois autres interfaces: "montée DP'_{c+1o}"; "montée DP'_{c+1e}"; "montée DP'_{c+1r}". Il est par contre impossible de tester notre modèle lorsque les messages évoluent simultanément.

4-2 AUTOMATES A ETATS FINIS

Au niveau le plus bas de la décomposition hiérarchique, on obtient un ou plusieurs automates dont on définit les interfaces, les canaux de liaison, les états, les événements et les données.

Les automates à états finis sont des graphes composés :

- de places: elles indiquent l'état dans lequel le système se trouve;
- d'arcs qui permettent d'évoluer d'une place vers une autre.

La logique de l'automate est basée sur l' émission et la réception de messages.
Les arcs sont franchis lorsque l'automate recoit un message donné. L'automate consomme alors ce message (type boîte à lettre). Ces messages sont fugitifs. L'évolution d'une entrée d'état (du type booléen par exemple) ne décidera pas à elle seule du franchissement d'un arc. Le principe d'évolution de l'automate est donc basé sur des séquences d'événements, plutôt que sur des combinaisons de variables d'états.

Lors du franchissement d'un arc, l'automate élabore à son tour des messages. Ces messages sont alors envoyés vers l'automate d'un autre module. Une transition activée par un message peut également être conditionnée par des assertions. Les actions sont ainsi réalisées au niveau des places.

L'évolution d'une place vers une autre se passe de la façon suivante:

```
Aller de la place P1 à la place P2,          | transition
 [quand je recois le message M1,             |
 et si le booléen B est vrai]                | condition
       alors [envoyer le message M2          |
              effectuer le calcul suivant...] | action
 fin.
```

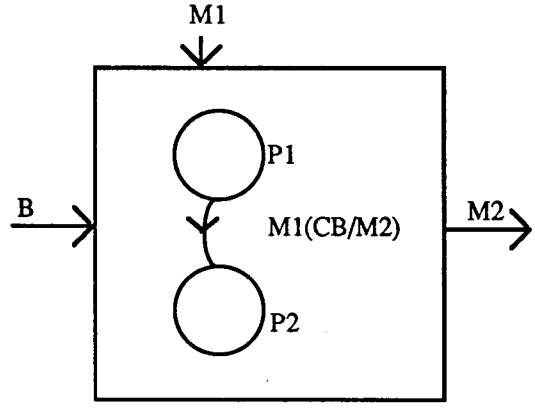


FIGURE 2-7

L'évolution de l'automate se voit donc principalement dans la réception de messages. L'activation ou non d'une place est transparente vis à vis de l'extérieur. Les informations transmises sont des informations de type lettre. Elles ne sont consommées qu'une seule fois.

Pour exemple l'automate à états finis du module logique d'occupation est donné dans la figure 2-8. La matrice de l'automate est décrite dans la figure 2-9.

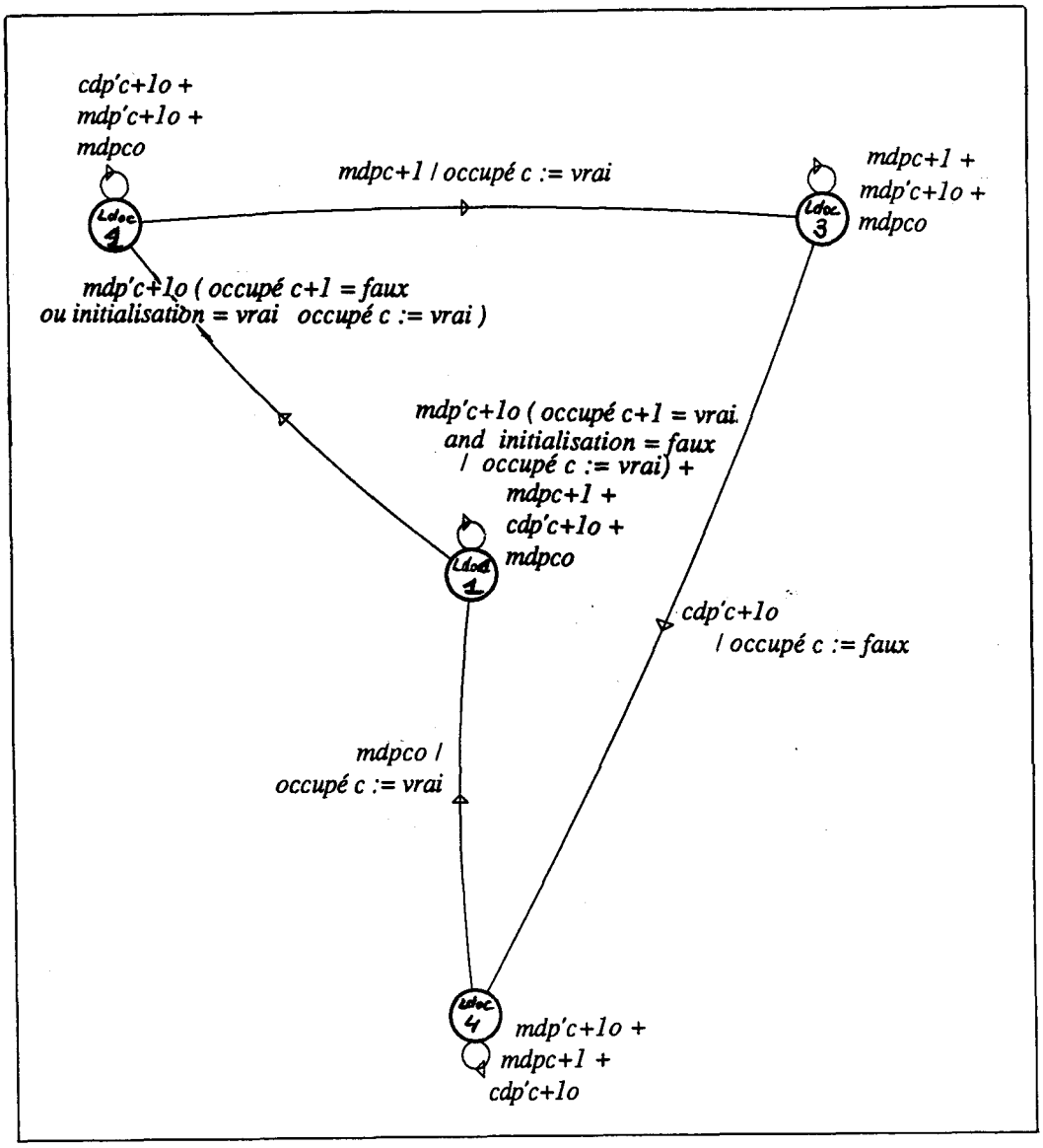


FIGURE 2-8

message		etat	E1	E2	E3	E4
			ldoc1	ldoc2	ldoc3	ldoc4
Me1	montee dpc+1		v/.	A0/E3	v/.	v/.
Me2	montee dp'c+1o	C1	A0/E2	v/.	v/.	v/.
		C2	A0/.			
Me3	montee dpco		v/.	v/.	v/.	A0/E1
Me4	chute..c+1 o		v/.	v/.	A1/E4	v/.

v : sans message — *• rebouclage sur la même place*

C1 : occupé c+1=Faux ou initialisation = Vrai — *A0 : occupe c := Vrai*

C2 : occupé c+1=Vrai et initialisation = Faux — *A1 : occupe c := Faux*

FIGURE 2-9

La matrice est un autre moyen de description de l'automate. Pour chaque état et pour chaque message, elle explique comment évolue l'automate. Une case vide indiquerait que dans cet état là, si le message est envoyé vers le module, l'automate ne sait pas comment réagir. Une spécification est complète si toutes les cases de la matrice sont pleines. Tous les cas possibles ont alors été pris en compte. Pour éviter la sur-spécification, une case ne peut rester vide qu'à condition de justifier que ce cas est impossible. Par contre, si le cas est possible, mais qu'il ne fait pas évoluer l'automate, il faut qu'en même faire une boucle de l'état sur lui-même.

Nous avons vu que cet automate évolue par une méthode de réception et d'émission de messages. L'inconvénient est qu'on ne peut pas faire évoluer l'automatisme uniquement sur des variables d'états et des assertions. Or dans notre cas d'automatisation d'une ligne de métro, nos entrées sont l'état des capteurs et non des messages de type lettre (ils ne disparaissent que si ils sont consommés).

Ce concept de message est difficile à appliquer pour des automatismes basés sur l'état des variables.

Les automates à états finis s'adaptent mieux à la spécification de logique micro-programmées où un message symbolise une interruption du microprocesseur.

4-3 LA SIMULATION

Nous avons décrit précédemment la modélisation par une démarche SADT puis par des Automates à Etats finis. La construction par SADT est un modèle statique alors que les Automates à Etats Finis spécifient la dynamique du modèle.

De plus, tous les Automates à Etats Finis d'un même modèle peuvent communiquer entre eux grâce aux interfaces définis dans les modules.

La simulation de ces automates permet de vérifier que la conception est correcte.

Le simulateur propose plusieurs messages. L'opérateur peut sur le clavier choisir un de ces messages puis lancer ensuite la simulation. Il faut noter que seuls les messages susceptibles de faire évoluer les Automates à Etats Finis sont proposés. On ne retrouve pas à chaque fois tous les messages. Le menu est donc facile à lire. Si on veut choisir un message et qu'il ne se trouve pas dans le menu, c'est que ce cas n'a pas été spécifié.

Dans l'exemple ci-dessous, on teste le cas d'une rame pénétrant sur un canton occupé:

```
SYSTEME : bea2
                                                          Couverture (%)
    Modules                                          Transitions  Etats
-------------------------------------------------------------------------

+   M bea2
+     M1 logique d'occupation                            35          100
+     M2 alarme perte eac                                25          100
+     M3 alarme recul ou penetration sur canton occupe   50          100
+     M4 bilan d'alarme                                  19          75
+     M5 commande du pcc                                 75          100


SCENARIO : *penetration sur canton deja occupe par une autre rame*
           *on part de l'etat initialise avec le canton c+1 non occupe*

>montee dpc>montee dpco
    \occupe c(Vrai)

>montee dpc>montee dpc e

>montee dp'c+1>montee dp'c+1o
    /occupe c+1(Vrai)
    \occupe c(Vrai)

>montee dp'c+1>montee dp'c+1 e

>montee dp'c+1>montee dp'c+1 r
    /occupe c+1(Vrai)
    <alarme de canton

*creation d'une alarme lors de la montee sur la dp'c+1
        dans le module d'alarme recul*
```

On peut remarquer que le message "Alarme de Canton" est émis lorsque le message "montée sur DP'c+1" est envoyé au module "Alarme recul ou pénétration sur canton occupé".

L'outil SADT et Automates à Etats Finis est un outil qui réunit à la fois le statique, pour aider le responsable de la spécification à faire une décomposition hiérarchique et le dynamique pour simuler la spécification.

L'inconvénient des AEF est qu'ils réagissent par messages et non par condition sur des états. Or les Transports Terrestres Automatisés fonctionnent sur des variables d'états construites à partir des capteurs. Le principe de message consommable est donc difficile à reproduire. Cette méthode a par contre le mérite d'être supportée par un outil informatique (ASA) distribué dans le commerce.

5- CONCLUSION

Dans ce chapitre, nous avons mis en évidence l'importance de la spécification dans le projet. La spécification est le point de départ du projet. Elle sert de référence depuis la mise en œuvre jusqu'à l'exploitation du système.

Pour garder son intérêt tout au long du projet, une spécification doit être de bonne qualité: complète et conforme au besoin, cohérente et significative.

Dans des problèmes complexes tels que l'automatisation d'une ligne de métro, il est difficile de bien spécifier. Des outils formels dans leur syntaxe et leur sémantique sont nécessaire. Ils aident à écrire la spécification, et à vérifier le contenu de la spécification. Ils limitent les risques d'erreurs lorsqu'on passe en phase de fabrication et d'intégration.

Dans le chapitre suivant, nous décrirons deux outils de spécification formelles et leur méthode d'utilisation.

Nous vérifierons par la suite que les modèles écrits se protègent contre les risque définis.

CHAPITRE 3
CREATION DE MODELES
DE SPECIFICATION FORMELLE

L'analyse préliminaire du chapitre précédent décrit le problème à traiter. La spécification formelle, quant à elle, décrit une solution au problème.

Pour tester et valider la solution choisie, nous avons vu l'importance d'une spécification exécutable. Les facilités d'exécution dépendront de l'outil de modélisation utilisé. nous allons nous intéresser à deux outils de spécification formelle.

En matière de modélisation, les outils existants sont très nombreux. Chacun d'eux cherche à créer un modèle de bonne qualité. Ils doivent aider le responsable de la spécification à écrire un "bon" modèle suivant le sens que l'on a défini dans le paragraphe "qualité de la spécification", c'est-à-dire un modèle complet, conforme au besoin, cohérent, significatif et précis. Et ils doivent par la suite aider le responsable de la spécification à valider son modèle.

Les principales qualités que nous mettrons en évidence pour le choix d'un outil sont [PAR86]:

- les possibilités d'expression d'un modèle, notamment l'aptitude à la modularité et à la puissance des liens inter-modulaires (**souplesse**);

- son aptitude à la validation, c'est-à-dire à la preuve que le système satisfait bien les besoins attendus (**preuve**). Dans notre cas, où nous cherchons à obtenir une spécification sûre, cette propriété prendra une place non négligeable dans nos études.
  On distinguera:
  -la validation de **cohérence** qui consiste à vérifier que les modules sont cohérents entre eux et qu'il n'y a pas d'informations contradictoires. Elle montrera que la syntaxe de l'outil de modélisation est très rigoureuse et se base sur la vérification de règles décrites et sur la gestion de dictionnaires de données. Elle peut aisément être automatisée.
  -la validation **fonctionnelle** qui consiste à vérifier que la fonction réalise correctement son travail. Celle-ci sera étudiée dans la troisième partie;

- son aptitude à la matérialisation du système modélisé, c'est-à-dire sa mise en œuvre sur une machine (**portabilité**);

- son aptitude à constituer un dossier clair et compréhensible par un maximum de personnes. Un même modèle doit avoir la même signification pour les différents lecteurs (**documentation**).

La valeur (qualité ou intérêt) d'un outil dépendra de son aptitude à satisfaire les qualités que nous venons de décrire.

Actuellement la plupart de nos spécifications d'automatismes à haut niveau de sûreté de fonctionnement sont écrites en logique utilisant des booléens. Les opérateurs utilisés sont les opérateurs ET,OU, NON et les temporisations. Les variables ont deux états Vrai et Faux.

Cette méthode a l'avantage d'être parfaitement maîtrisée par tout le monde et d'être simple à lire et à comprendre.

D'autre part, cette méthode de modélisation, dite en porte logique, est proche de la réalité. Les spécifications sont directement écrites dans ce langage. Il n'y a aucune interprétation à faire pour passer de la spécification à un modèle de validation et pour passer à la réalisation. On s'affranchit donc de tous les problèmes d'interprétation.

Autour de la méthode de modélisation en porte logique, nous avons développé un simulateur de ces automatismes et de leur environnement. Ce simulateur cherche à mettre le système spécifié dans un état proche de la réalité. Nous simulons le comportement des automatismes fixes en fonction des événements extérieurs: déplacements d'un train sur la zone, sélection du mode manuel ou automatique du train, occultation des capteurs...

Nous pouvons ainsi rapidement tester des cas d'exploitation compliqués mettant en jeu plusieurs trains. Nous verrons lors de la validation, les avantages d'un simulateur.

Par contre ce type de modèle n'est pas suffisant pour spécifier des automatismes réalisés en logiciel. Il ne permet pas d'écrire toutes les fonctions d'un langage de programmation. D'autres outils de modélisation sont donc devenus nécessaires.

Ces outils sont nombreux et possèdent de plus une très grande panoplie de propriétés. Ces propriétés font varier le modèle suivant l'application. Si on travaille sur une fonction où le temps réel a une place importante ou sur une fonction statique et répétitive par exemple, les propriétés de la méthode de modélisation ne seront pas les mêmes.

Plus les propriétés sont nombreuses, plus le modèle sera rigoureux et facile à vérifier. En contre partie, plus il sera difficile à créer. Il faut donc trouver un compromis entre la nécessité d'une propriété et les difficultés de mise en œuvre qu'elle ajoute au modèle.

La portabilité du modèle dépendra des outils informatiques qui ont été développés autour de la méthode.

Certaines de ces méthodes ont été normalisées (exemple des réseaux de Petri). Grâce à cette normalisation, tout le monde parle le même langage et comprend le modèle de la même façon.

Les propriétés acceptées par la normalisation sont parfois limitées par l'outil informatique qui la supporte. Ces supports informatiques deviennent alors une référence pour l'outil de spécification. On dit que l'on utilise un outil de spécification du type de celui qui est développée pour un support existant. Le développement d'outil informatique a forcement accéléré le phénomène de normalisation. Ceux-ci s'adressent à plusieurs utilisateurs de sociétés différentes. Il fallait obligatoirement que tout le monde parle le même langage et que donc on définisse des normes d'écriture de ce langage.

Pour un langage informatique, tel que le langage C par exemple, on ne comprendrait pas que lorsque l'on change de compilateur, il faille revoir complètement la façon d'écrire le programme. Pour un outil de modélisation, le problème est le même. Il faut opter pour une normalisation afin que tout le monde parle le même langage.

Dans ce paragraphe, nous allons comparer quelques outils de modélisation des spécifications. Nous nous orientons vers des modèles formelles: réseaux de Petri, Grafcet. Ces modèles sont utilisées par la suite, pour tester et valider la spécification.

Lors de l'écriture d'un modèle, le piège dans lequel il ne faut pas tomber est d'obtenir un modèle trop grand et trop complexe accessible au seul créateur du modèle. Si le modèle est trop complexe, le concepteur ne sera pas capable de le développer et le risque d'erreur est grand. Pour remédier à ce problème, nous définirons pendant la création des modèles quelques règles à suivre pour obtenir un modèle correct. Ces règles seront très personnalisées et dépendront de l'application que l'on étudie.

1-LES RESEAUX DE PETRI

Les systèmes informatiques ou de commande d'automatismes logiques sont conçus comme des sous-ensembles de sous-systèmes.

Pour augmenter la puissance de calcul d'un système, on fait coopérer plusieurs sous-systèmes entre eux. Pour optimiser le nombre de composants, les temps de réponse des automatismes ou les paramètres (ex: temps de parcours sur une inter-station de métro), on est également amené à concevoir des fonctions de plus en plus compliquées. Ces contraintes sont nécessaires pour répondre à des exigences de disponibilité mais également de sécurité.

Par exemple: une rame lancée à 80 km/h parcourt 22 m en une seconde. S'il y a un risque de collision, et que le train doit être arrêté, le temps nécessaire pour élaborer l'alarme et arrêter le train en freinage d'urgence doit se compter en milli-secondes plutôt qu'en secondes.

En contrepartie apparaissent des problèmes liés à l'expression de la fonction ou au comportement d'un système. Evoquons ici les problèmes de blocage, de cohérence de données, de contraintes temporelles...L'étude de ces problèmes se pose dès la spécification. Il est indispensable de pouvoir vérifier et valider le travail effectué ce qui nécessite de savoir modéliser et d'exprimer les propriétés qu'on attend.

Les réseaux de Petri constituent un modèle correctement défini d'un point de vue théorique. De nombreux ouvrages décrivent les réseaux de Petri et tentent d'unifier les notations et le vocabulaire employé [BRA83]. De plus ils possèdent des propriétés telles qu'ils peuvent être utilisés pratiquement d'un bout à l'autre du cycle de vie d'un projet [ALA85].

1-1 PRESENTATION DE L'OUTIL

Les réseaux de Petri sont utilisés afin de modéliser le comportement dynamique de systèmes discrets. Ils sont composés de deux types d'objets: les places et les transitions. A chaque transition est associée un événement dont l'occurrence provoque la modification de l'état du système. Plus précisément, les places jouent le rôle de variables d'état du système et sont à valeur entière. De façon imagée, ces valeurs entières sont représentées par autant de marques affectées à une place. L'état du système est alors associé à un marquage définissant pour toutes les places le nombre de marques qui lui est affecté (ou bien qu'elle contient). A l'occurrence d'un événement correspond le franchissement de la transition. Elle dépend de la satisfaction de pré-conditions.

L'ensemble des définitions et des notations des réseaux de Petri sont reprises dans l'annexe 2. Nous y avons également décrit les propriétés qui permettent de valider les réseaux de Petri. Ce sont en général des limites qu'on impose aux réseaux. Elles prouvent que l'automatisme conçu est sans risque de blocage. Elles évitent à l'utilisateur d'écrire des modèles trop compliqués à relire et difficiles à valider.

Les réseaux de Petri utilisés dans l'exemple sont des réseaux ordinaires à arcs inhibiteurs. Les arcs inhibiteurs "testent à zéro" le marquage d'une place. Ils n'autorisent le franchissement d'une transition que si le marquage de la place est nul.

Nous porterons également un regard sur les réseaux à prédicats.

1-2 EXEMPLE 1: LOGIQUE DE CANTON

Nous avons choisi de montrer, comme application, la mise en réseaux de Petri de deux fonctions d'automatismes fixes d'un métro: la logique de canton ainsi qu'une fonction de rebroussement. La logique de canton est celle qui a été traitée dans le chapitre 2 des Automates à Etats Finis. La fonction de rebroussement commande l'itinéraire que doit suivre le train. Leur spécification axiomatique est donnée dans le paragraphe d'analyse préliminaire. Elles sont intégrées dans la partie contrôle du Pilote Automatique Fixe (PAF). Dans le premier modèle, les réseaux utilisés sont des réseaux classiques avec des arcs inhibiteurs.

Afin d'avoir un modèle le plus clair possible, on s'est imposé une méthode de construction. Tout d'abord les fonctions ont été découpées très finement. Chacune de ces fonctions est traitée séparément et représentée sous forme d'un réseau de Petri qui représente un bloc.

Un bloc est donc un réseau de Petri pour lequel le marquage des places définit l'état du système (canton occupé ou non; valeur de l'alarme...).

Les transitions sont conditionnées :

- par des valeurs de booléens. Ce sont les interfaces avec les équipements adjacents: Télécommandes du poste central, émission sur les boucles de détection positives (DP), occultation de détecteurs négatifs (DN)...;
- par le marquage des places du réseau;
- par le marquage des places d'un réseau adjacent. Ce sont les informations élaborées par d'autre blocs et qui peuvent être utilisées dans ce bloc. En effet on garde la possibilité de transmettre d'un bloc à l'autre l'état marqué ou non d'une place donnée.

Les entrées et les communications entre les blocs seront donc définies pour chacun d'eux, en précisant le type d'information attendu et la provenance de cette information. Nous définirons des contraintes à suivre pour l'utilisation des communications.

Dans cet exemple nous traitons la logique de canton de la ligne. Nous allons spécifier la logique pour les trois types de cantons (canton d'entrée, canton de sortie, canton courant).

1-2-1 LE RESEAU DE PETRI POUR UN CANTON D'ENTREE OU UN CANTON COURANT

Le schéma 3-1-1 et 3-1-2 représente le réseau de Petri de la logique de canton avec son traitement d'alarme. Nous donnons tout d'abord le réseau pour un canton du type canton d'entrée de tronçon ou canton courant.

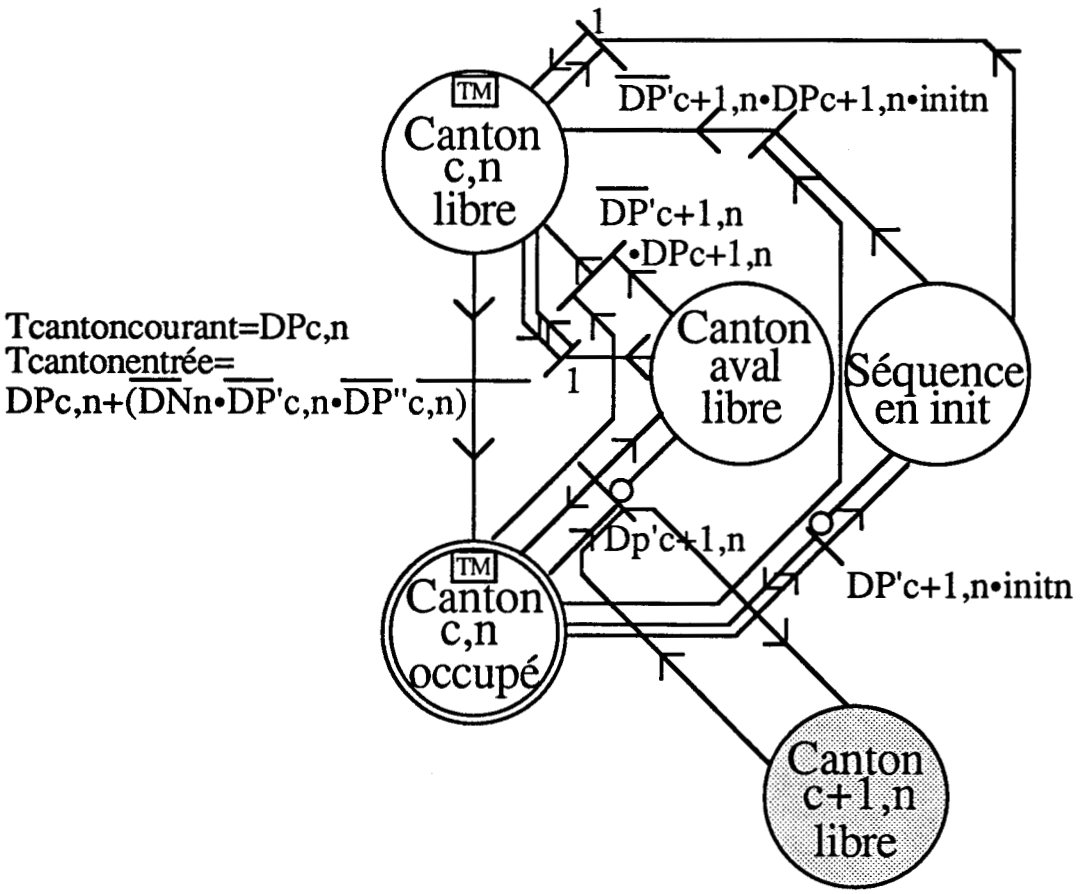


FIGURE 3-1-1

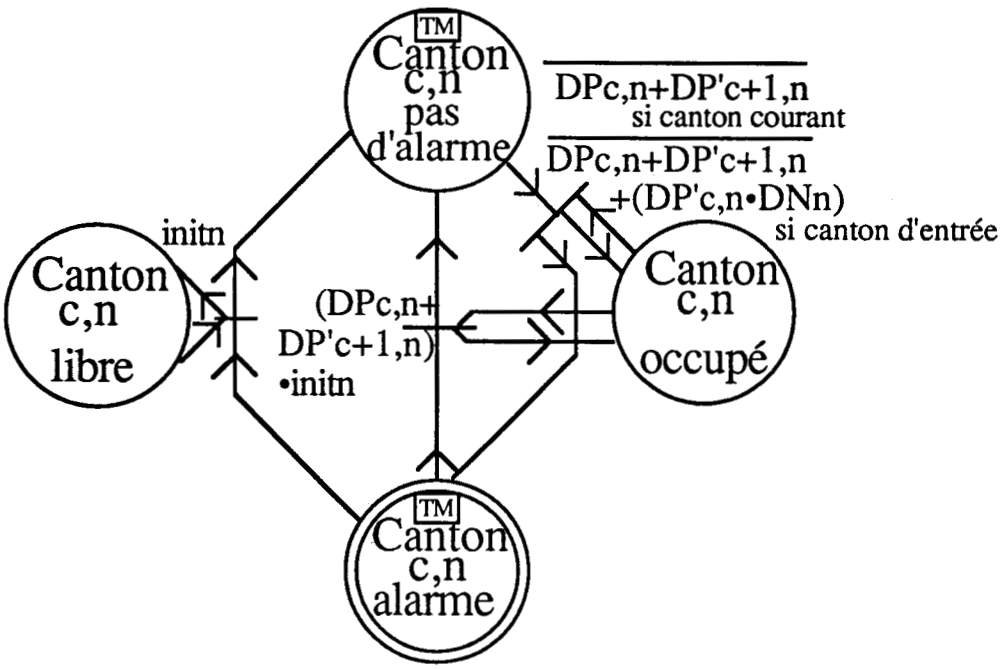


FIGURE 3-1-2

1-2-2 PRISE EN COMPTE DE L'ANALYSE PRELIMINAIRE

Les explications suivantes mettent en évidence le fait que la démarche suivie est rigoureuse et que lorsqu'on crée une transition le souci principal est de respecter les critères de l'analyse préliminaire:

- •Les boucles DP' se justifient pour distinguer les sorties du canton des pertes des EAC.
- •On ne libère le canton que si le précédent était libre, pour le respect du critère 3-a (pénétration sur canton occupé).
- •Dans la condition de libération: Non $DP'_{c+1,n} \cdot DP_{c+1,n}$ le terme $DP_{c+1,n}$ se justifie à cause du critère 2 (détecter les pertes d'EAC); il ne faut pas libérer le canton si la rame perd ses EACs.
- •Pour créer une alarme lorsqu'on perd les EACs, on part du principe que si le canton est occupé et que si les boucles $DP_{c,n}$ et $DP'_{c+1,n}$ ne recoivent aucune émission alors on met le canton en alarme.
- •Pour le respect du critère 1-b (détecter les mouvements de rame sans EAC), la condition d'occupation est différente pour un canton d'entrée. On utilise une barrière négative à l'entrée.
- •Les transitions égales à 1 sont nécessaires pour les risques de blocage après la commutation de init à normal et vice versa.

Cependant la méthode de construction des schémas laisse une large place à l'imagination. Un réseau n'est pas unique et est propre à son spécifieur. Il peut malgré toute l'attention portée contenir encore des erreurs.

Lorsque le spécifieur crée sa spécification, il essaie de prendre en compte tous les critères. Il peut éventuellement oublier un cas d'exploitation pour lequel le critère ne s'applique pas.

1-2-3 LE BLOC

Il faut remarquer que pour traiter la fonction "logique de canton et traitement d'alarme du canton c,n", nous avons dessiné un réseau de Petri qui parait indépendant du reste de la fonction. En réalité, ce réseau utilise la place "Canton c+1,n libre". Celle-ci est créée dans la fonction équivalente mais pour le canton c+1,n. Ce réseau du canton c+1,n n'a pas été complètement dessiné car le schéma serait trop lourd et illisible. De plus, c'est exactement le même réseau mais avec d'autres paramètres. Le réseau n'est ainsi dessiné qu'une seule fois. Toutes les informations sont écrites en fonction des paramètres c et n (ex: $DP_{c,n}$) et le même schéma peut être reporté tout au long de la ligne.

Lors de l'écriture du modèle, nous voulions utiliser une décomposition fine de la fonction et dessiner un réseau pour chacun des petits modules. Bien entendu les modules peuvent communiquer entre eux. Pour pouvoir vérifier les propriétés de construction (vivant, borné...) sur chaque réseau plutôt que sur l'ensemble de la fonction, nous avons limité les communications entre les réseaux. Nous acceptons qu'un réseau "consulte" l'état d'une place d'un autre réseau, mais il ne doit pas modifier le marquage de ce réseau.

Nous appelons chaque réseau un bloc. Le réseau de Petri donné ci-dessus constitue donc un bloc.

Dans ce schéma, les places des autres réseaux dont on a besoin de connaître l'état ont été notées par un rond sur fond pointillé. L'état d'une place du réseau qui peut être lu par un autre réseau est une télémesure du bloc que l'on est en train de traiter. Cette télémesure est indiquée par le sigle suivant dans la place concernée:

TM

Pour en terminer avec les notations utilisées, nous soulignons que les places initiales ont été notées par des doubles ronds. Nous considérons qu'à l'initialisation, ces places contiennent une seule marque et les autres places zéro marque.

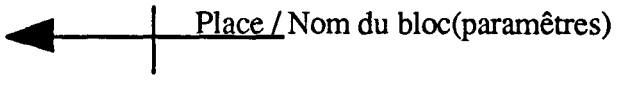
Un bloc est décrit par les informations suivantes:

- **Le nom du bloc** qui permet de le différencier des autres et qui indique la fonction qui est traitée. *Ex: logique de canton et traitement d'alarme.*

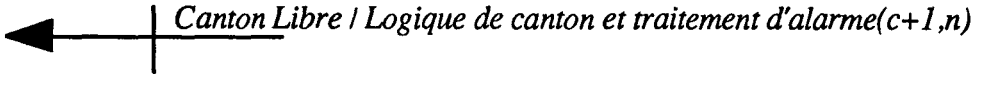
- **Les paramètres du bloc.** Si le bloc traite une fonction que l'on retrouve dans chaque canton, il n'est écrit qu'une seule fois. Les paramètres définiront dans ce cas, le numéro du PA et le numéro du canton. Les paramètres sont à saisir lorsque l'on utilise ce bloc. Les variables d'entrées ou de sorties en dépendront. *Ex: paramètres c et n .*

- **Télémesures du bloc**: ce sont les informations de sortie données par le marquage des places du réseau. Le marquage d'une place indique l'état du système ou est utilisé pour passer une commande vers les actionneurs *(exemple: le marquage de la place alarme entraîne la coupure de la haute tension et commande d'un freinage d'urgence).* Elles peuvent être envoyées au PCC mais également être utilisées par d'autres blocs. On peut considérer les télémesures comme étant les sorties du bloc. *Ex: canton c,n occupé.*

-**Les entrées du bloc**:
  - **Les télémesures venant d'un autre bloc.**Ce sont les places extérieures à ce bloc et dont on a besoin de connaître l'état. On les notera de la façon suivante: "nom de la place / Bloc de provenance (paramètres)" et devant une flèche barrée par un trait symbolisant la transition:



Exemple :



- **Les variables venant du système**: ce sont les informations fournies par les capteurs ou les télécommandes envoyées par d'autres équipements.

Chaque bloc sera représenté suivant le schéma 3-1-3:

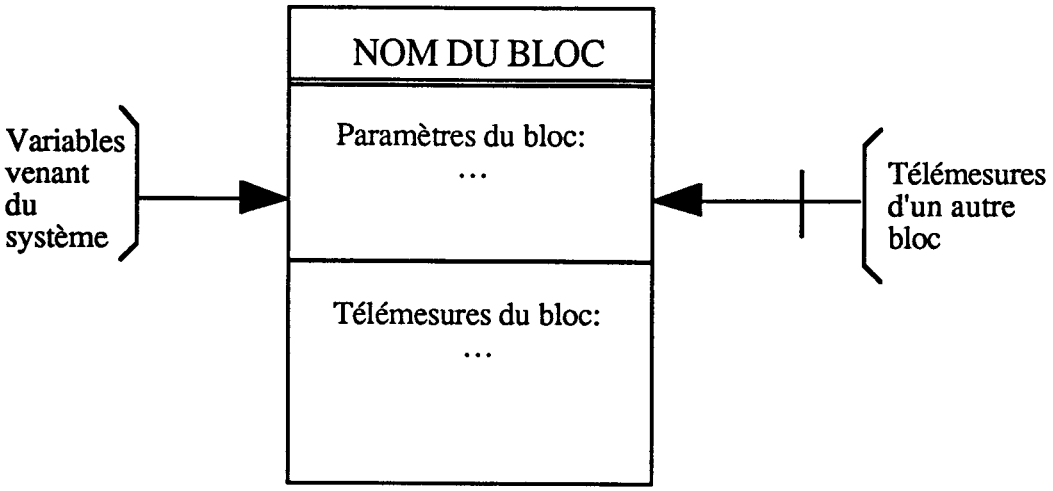


FIGURE 3-1-3

Le bloc du réseau de Petri de la logique de canton pour un canton d'entrée ou un canton courant est représenté dans la figure 3-1-4.

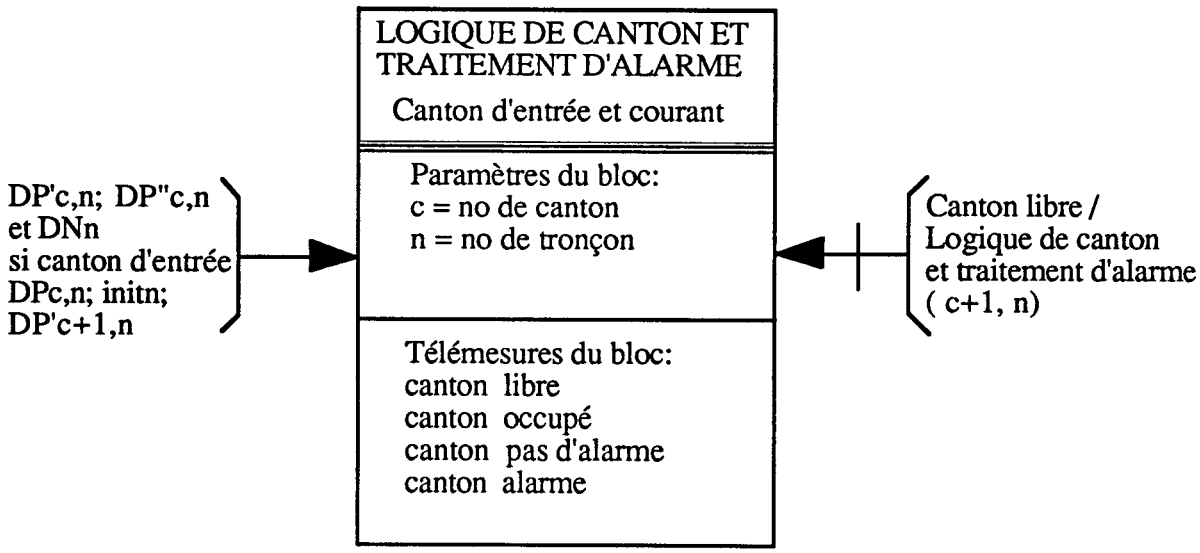


FIGURE 3-1-4

Cette représentation comprend toutes les informations nécessaires pour relier les blocs entr'eux lorsqu'on crée un équipement. Il faut noter que la flèche de droite doit toujours être reliée à un bloc et il faudra donc vérifier l'existence de ce bloc. De même pour la flèche de gauche, il faut vérifier que les variables sont disponibles.

A chaque bloc est associé un réseau de Petri. Ce réseau est un quadruplet: R=<P;T;Pré;Post> où :

- P est un ensemble fini de places,
- T est un ensemble fini de transitions,
- Pré est application PxT $\rightarrow$ M appelée d'incidence avant,

- Post est application PxT → M appelée d'incidence après.

En général M = ℕ ou ℕ ∪{∅}. ∅ est l'élément vide utilisé pour les arcs inhibiteurs.

Dans notre cas, on n'utilise pas de marquage multiple dans les places. M est donc égal à l'ensemble {0;1;∅}.

L'ensemble P des places est divisé en deux sous-ensembles:

- $P_{interne}$ est l'ensemble des places propres au réseau
- $P_{externe}$ est l'ensemble des places des autres réseaux et dont on a besoin de connaître l'état.

1-2-4 LES COMMUNICATIONS ENTRE LES BLOCS

Les blocs peuvent communiquer entre eux puisqu'un réseau utilise le marquage d'une place d'un autre bloc (ce sont les télémesures venant d'un autre bloc).

Pour faciliter la vérification des propriétés du réseau de Petri, on s'est imposé une limitation dans les communications entre les blocs. On accepte qu'un bloc "consulte" l'état d'une place d'un autre bloc par contre ce bloc ne doit pas modifier le marquage de la place de cet autre bloc.

Si on veut changer le marquage d'une place P d'un autre bloc, alors l'état de P est conditionné par la fonction que l'on est en train de traiter et la place P doit donc être incluse dans le bloc de cette fonction. En effet le marquage d'une place représente une commande à passer pour cette fonction ou un état du système significatif pour la fonction traitée. Nous considérons alors que cette restriction n'est pas pénalisante pour l'écriture du modèle.

Cette propriété s'écrit comme suit :
pour un réseau R=<P;T;Pré;Post>
∀ p ∈ Pexterne (Pexterne ⊂ P est l'ensemble des places d'un autre réseau)
∀ t ∈ T
M(p)⟨t⟩ reste inchangé.

Pour respecter cette propriété, les seules configurations admises sont :
- les boucles.

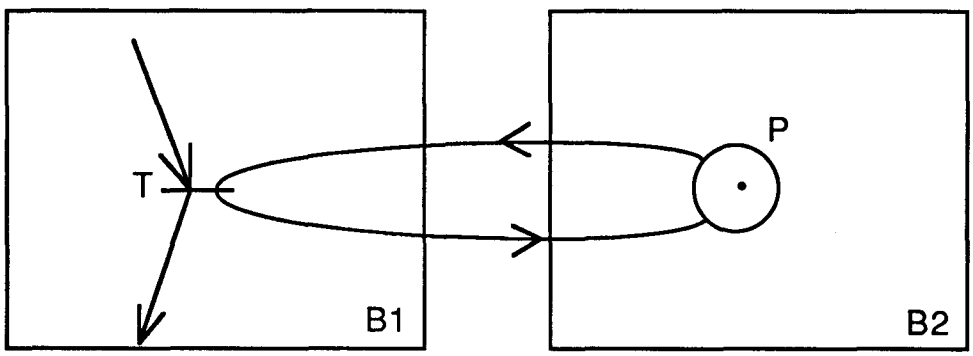


FIGURE 3-1-5

B1 et B2 sont deux blocs distincts. T1 ne sera franchie que si P2 est marquée. Quand T1 sera franchie, P2 sera toujours marquée.

-et les arcs inhibiteurs.

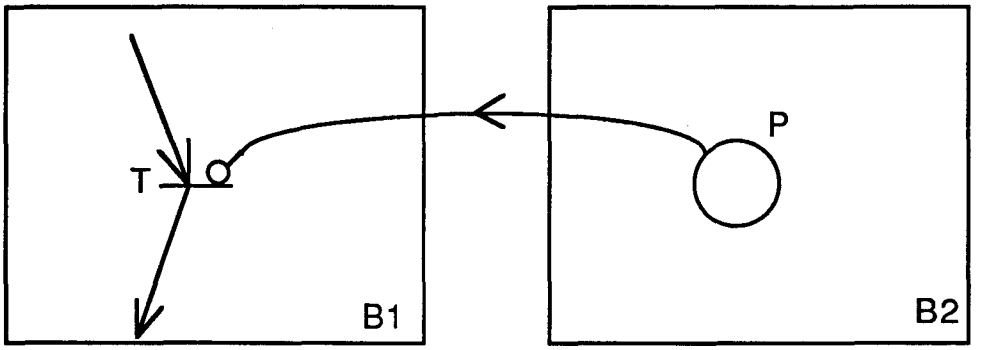


FIGURE 3-1-6

Un arc inhibiteur ne modifie pas le marquage de la place.

Cette contrainte à l'avantage de faciliter la vérification des propriétés du réseau. Etant donné qu'on ne modifie pas le marquage des autres blocs, il est possible d'étudier les propriétés d'un bloc en considérant que l'on a un réseau indépendant, l'évolution de ce bloc se faisant à l'intérieur de lui-même.

Il y a malgré tout une précaution à prendre. Pour les communications avec les boucles, si la place P du bloc B2 n'est jamais marquée au cours de l'évolution du réseau, on risque de bloquer le bloc B1. De même pour les arcs inhibiteurs, si P est toujours marquée.

Il faut alors vérifier, pour valider la propriété de non blocage du réseau de Petri, que les places sont :

- au moins une fois marquées pour les communications en boucle

- au moins une fois de marquage nul pour les communications avec les arcs inhibiteurs

et ceci pour l'évolution du marquage qui nous intéresse pour faire fonctionner notre système.

Outre le découpage en module des réseaux de Petri, et la facilité de vérifier les propriétés syntaxiques du réseau, l'utilisation des blocs ne présentent pas que des avantages. Elle impose des limites dans la construction du réseau. Par exemple, la synchronisation entre les tâches de deux blocs différents n'est pas possible. Dans la communication entre les blocs, on perd ainsi certains avantages des réseaux de Petri et en particulier ceux concernant le comportement dynamique. Cette limite ne nous a pas gêner pour des systèmes tels que ceux que l'on a étudiés ici. Si d'autres types de communication sont nécessaires entre deux places de deux blocs alors ces deux places doivent être réunies dans le même bloc.

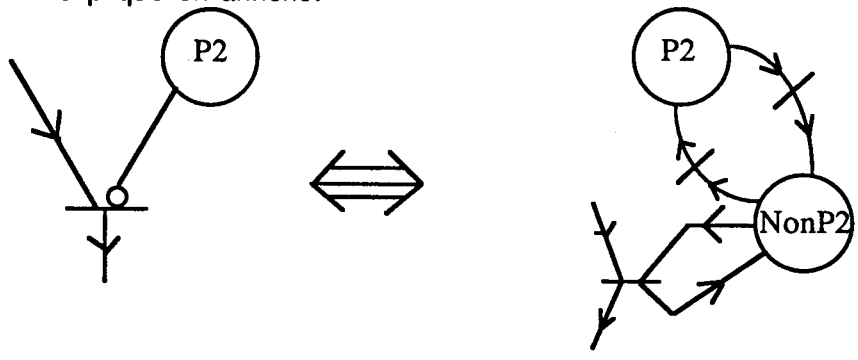
1-2-5 VERIFICATION DES PROPRIETES DU RESEAU

La vérification des propriétés du réseau ne sert pas à prouver la bonne réalisation. Elle vise simplement à vérifier que le réseau est viable et qu'il peut être exécuté. C'est une validation syntaxique.

Afin de vérifier les propriétés du réseau de la logique de canton, nous allons transformer le réseau obtenu.

Les communications entre les blocs seront ignorées suivant ce qui a été choisi lors de la création des blocs.

Quant aux arcs inhibiteurs, ils seront remplacées par des boucles suivant le schéma 3-1-7 expliqué en annexe.

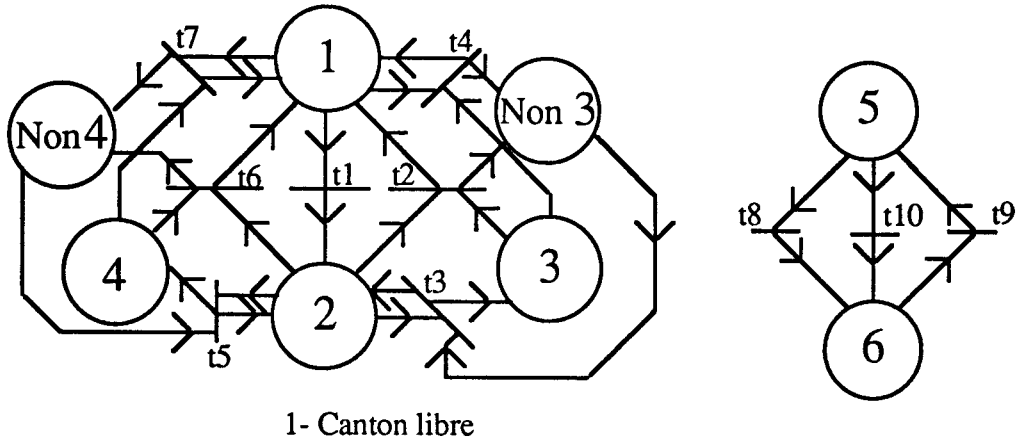


P2 et NonP2 sont incompatibles. M(P2) + M(NonP2) = 1

FIGURE 3-1-7

Du point de vue simulation, le réseau transformé reste le même que le réseau initial. Il contient simplement plus de places et de boucles. Les propriétés des réseaux de Petri sont ainsi plus faciles à vérifier.

Le réseau transformé est représenté dans la figure 3-1-8



1- Canton libre
2- Canton occupé
3- Canton aval libre
4- Séquence en init
5- Alarme
6- Pas d'alarme

FIGURE 3-1-8

Le réseau des places 5 et 6 reste indépendant car les communications avec les places 1 et 2 sont du type boucle et consultation sans modifier le marquage.

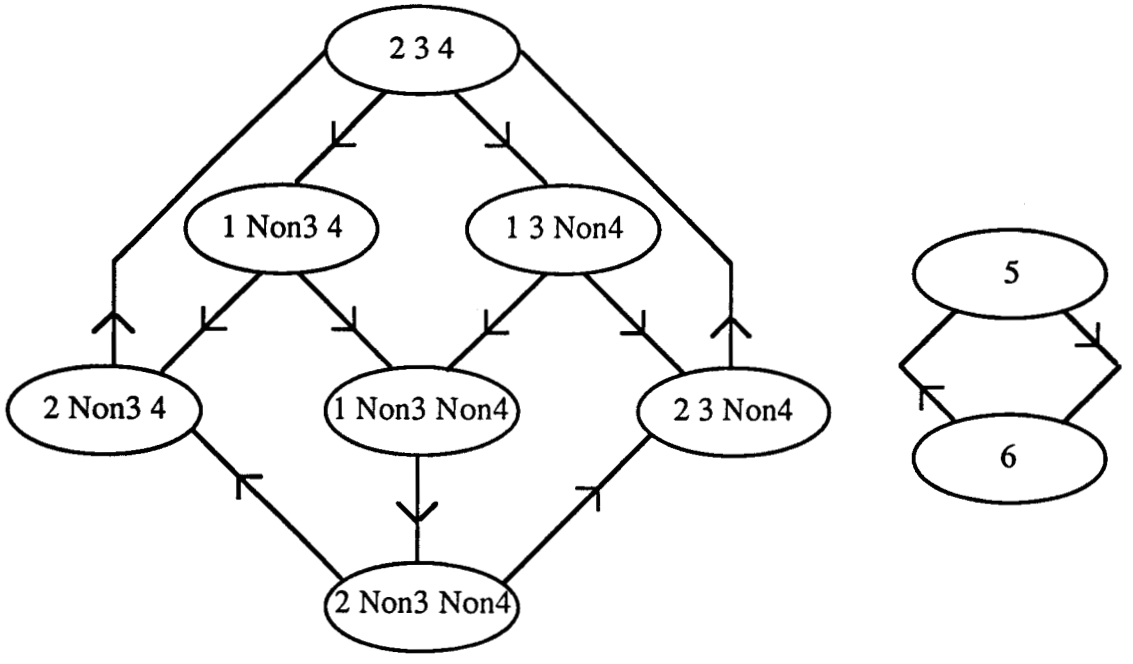
Pour ce réseau transformé, on écrit la matrice Pré/Post et on regarde les marquages accessibles.

	1	2	3	4	Non3	Non4
t1	1	1				
t2	1	1	1		1	
t3		1	1		1	
t4	1		1		1	
t5		1		1		1
t6	1	1		1		1
t7	1			1		1

	5	6
t8	1	1
t9	1	1
t10	1	1

MATRICE PRE/POST

FIGURE 3-1-9



MARQUAGES ACCESSIBLES ET LEURS TRANSITIONS

FIGURE 3-1-10

On part du marquage initial: canton occupé; les places 2, Non3 et Non4 sont marquées. On franchit les transitions une à une pour voir les marquages qui peuvent être atteints. On ne se soucie pas des prédicats associés aux transitions. Ils n'entrent pas dans la validation syntaxique mais dans la validation fonctionnelle.

Dans la matrice Pré/post la somme des lignes suivant chaque transition est nulle. Par conséquent, pour chaque transition, le nombre de marques tirées est égale au nombre de marques envoyées dans les places.

Le réseau transformé est donc borné.

Dans la liste des marquages accessibles, on remarque qu'à partir de n'importe quel marquage il est possible d'atteindre tous les autres marquages. De cette façon

$\forall M \in A<R,M_0>,\ A<R,M>=A<R,M_0>$

De plus pour toutes les transitions t du réseau, t peut être franchie pour le réseau $<R,M_0>$. En effet dans l'ensemble des marquages accessibles, toutes les transitions sont présentes au moins une fois.

1-2-6 LE RESEAU DE PETRI POUR UN CANTON DE SORTIE

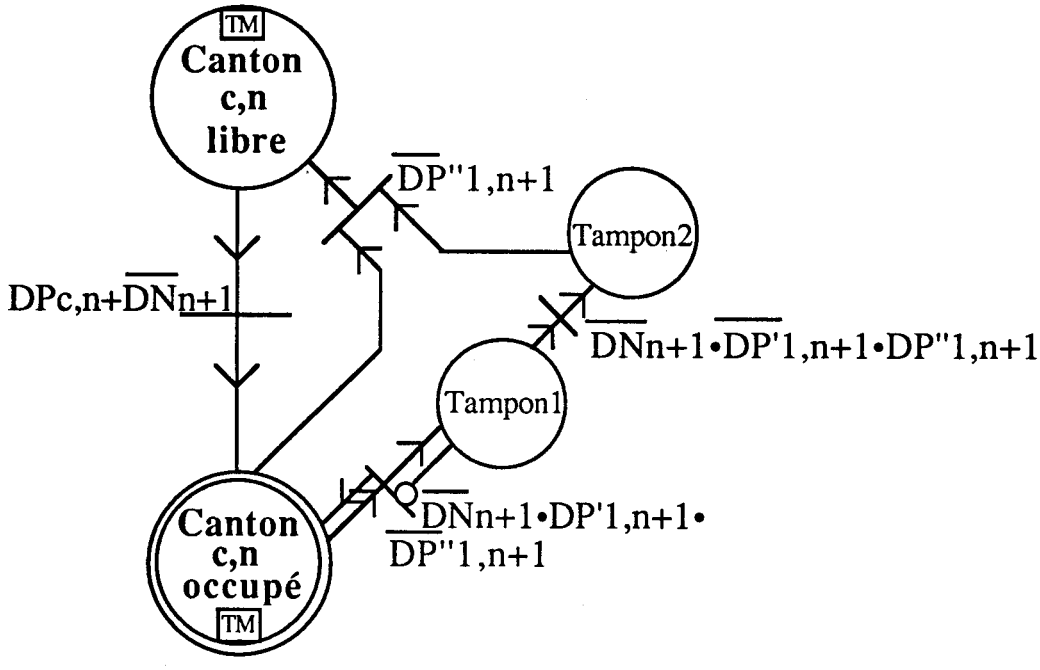


FIGURE 3-1-11

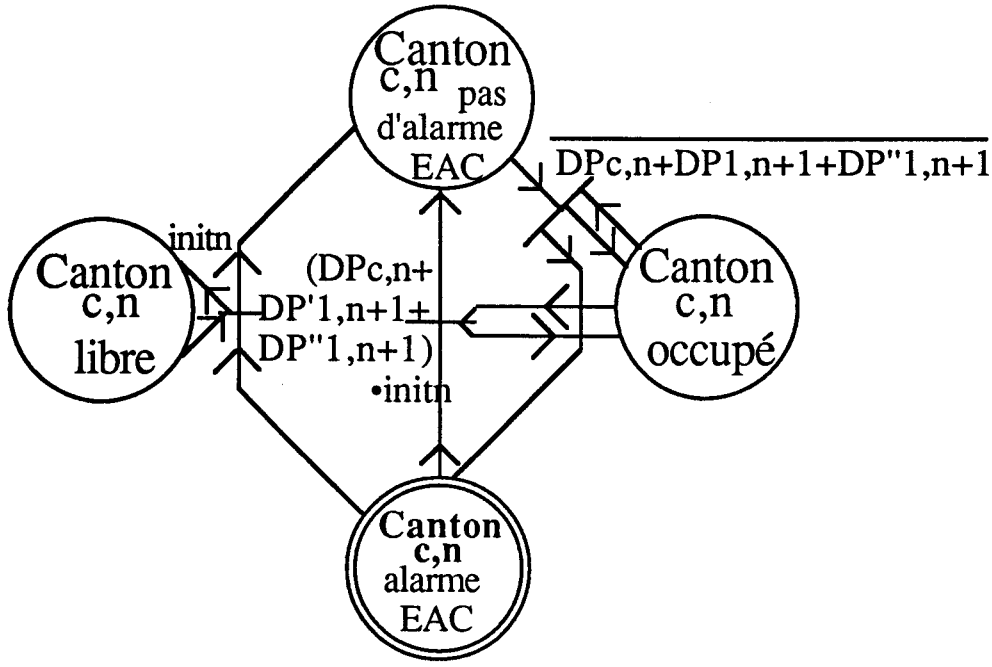


FIGURE 3-1-12

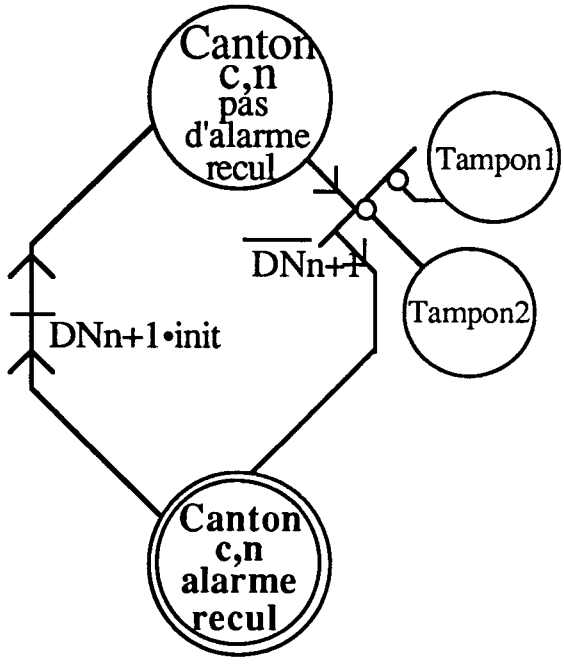


FIGURE 3-1-13

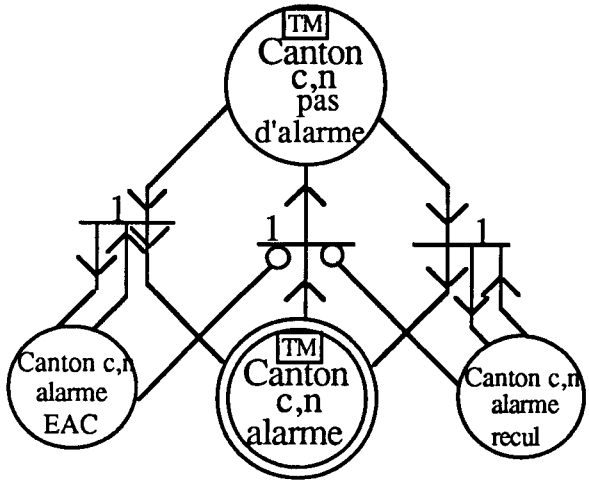


FIGURE 3-1-14

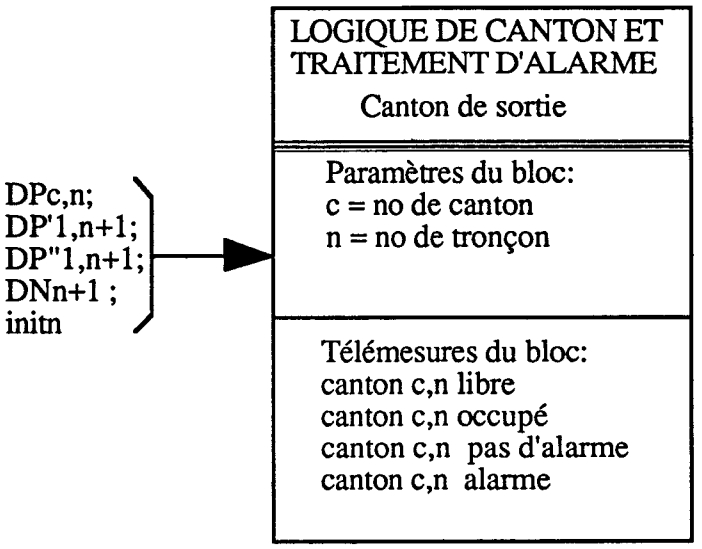


FIGURE 3-1-15

Ce réseau n'a pas de communication avec les autres réseaux. Il traite la zone de sortie du tronçon. Comme on a considéré que la gestion des tronçons était indépendante, il n'y a pas de communication avec l'autre réseau.

C'est pour cette raison que l'on a été obligé d'implanter une barrière de détection négative. Elle détecte les pénétrations de rame sans EAC par la sortie (cas de recul).

Dans le traitement d'alarme, la place tampon 1 est utilisée pour suivre la séquence sur les boucles de sortie. La place tampon 2 se justifie par le scénario suivant: un rame émet sur la boucle DP'1,n+1 puis occulte le DNn+1. La place tampon 1 devient marquée. A ce moment, on considère que la rame perd ses EAC et recule légèrement au point de désocculter le DNn+1. Sans place tampon 2 et avec une transition du type

$$DNn+1 \cdot \overline{DP'1,n+1} \cdot \overline{DP"1,n+1}$$

on libère ce canton c,n sans en occuper d'autre, ce qui est contraire au critère 1-a (une rame doit toujours occuper au moins un canton). On utilise donc de cette place tampon 2 qui ne sera marquée qu'après l'occupation du canton 1,n+1.

1-3 EXEMPLE 2: LA ZONE DE REBROUSSEMENT.

1-3-1 L'IMPLANTATION DES BOUCLES

Cette zone doit commander les itinéraires suivants:

-itSN1 et itSN2, les itinéraires normaux sens 1 et 2.

-itFDT, l'itinéraire de fond de tiroir. La rame arrive par le sens 1 et s'arrête au niveau du fond de tiroir.

-itRBT, l'itinéraire de rebroussement. La rame part du fond de tiroir et emprunte la communication dans le sens 2.

Elle commande également les aiguilles 1 et 2.

Les équipements au sol ont été implantés suivant les schémas que l'on retrouve dans le glossaire.

1-3-2 SPECIFICATION EN RESEAU DE PETRI

Cette logique a été découpée suivant les réseaux suivants:

-Passage du service normal (SN) au service de rebroussement (SRBT). Ce réseau gère le changement de mode d'exploitation entre le service normal (SN) et le service de rebroussement (SRBT). Il vérifie essentiellement avant de passer d'un service à l'autre que la zone est libre et que l'on ne risque pas d'interrompre un mouvement de rame alors qu'il n'est pas terminé.

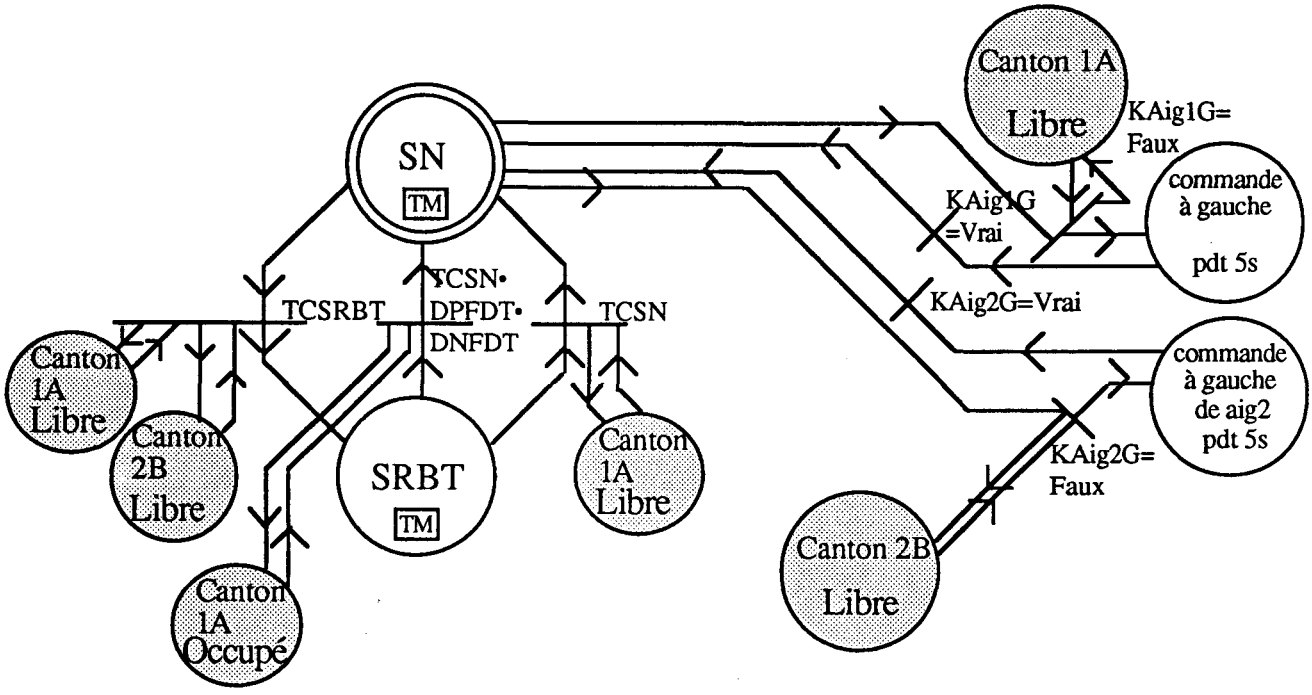


FIGURE 3-1-16

- l'alimentation des FS

Ces boucles commandent le déplacement des trains. Pour respecter les critères de l'Analyse Préliminaire, une FS ne sera alimentée qu'à condition que les aiguilles soient bien positionnées. On s'assure que la zone vers laquelle le train se dirige est libre et le restera. Cette dernière contrainte est réalisée soit par la logique de canton pour les FS de ligne, soit grâce aux réseaux du cycle de rebroussement pour les FS de la communication.

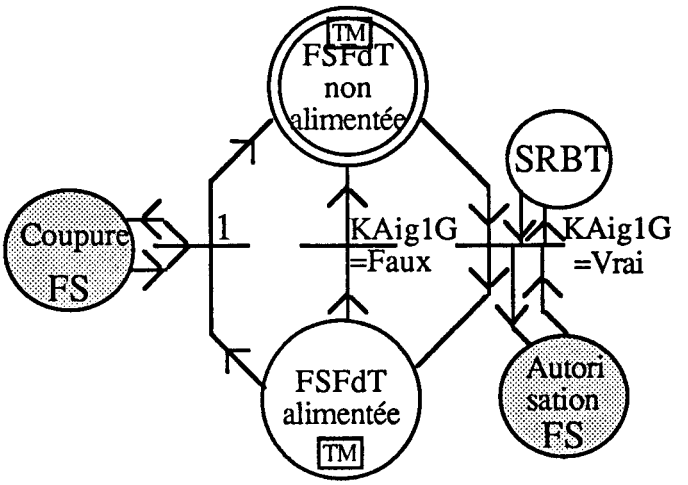


FIGURE 3-1-17

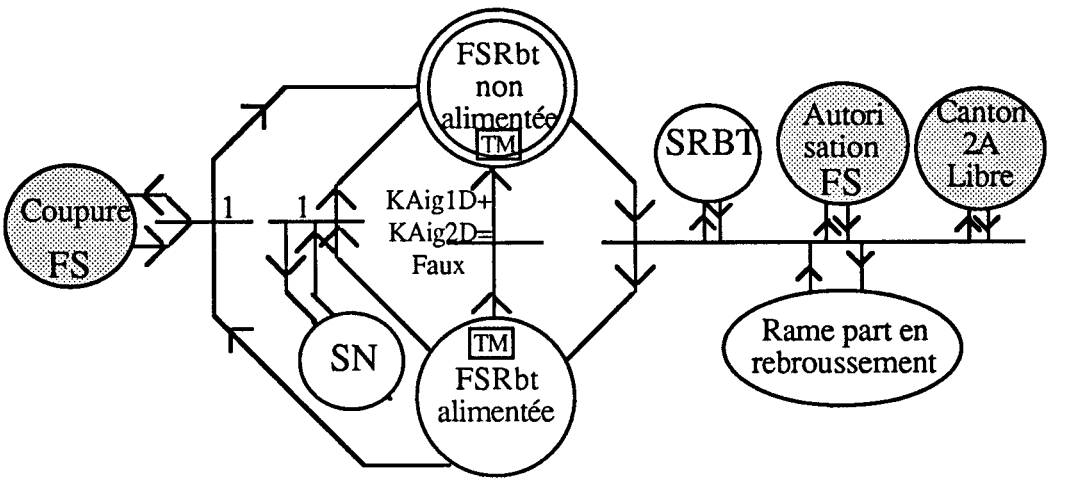


FIGURE 3-1-18

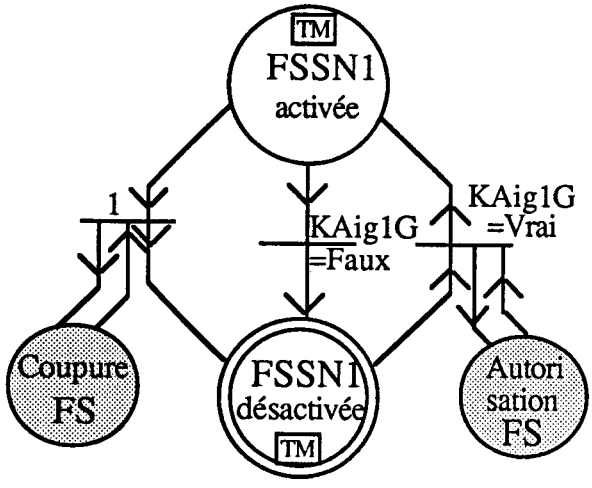


FIGURE 3-1-19

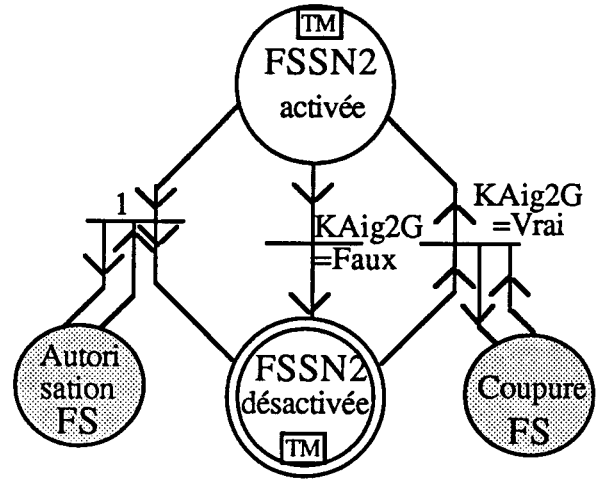


FIGURE 3-1-20

-l'alimentation générale des FS

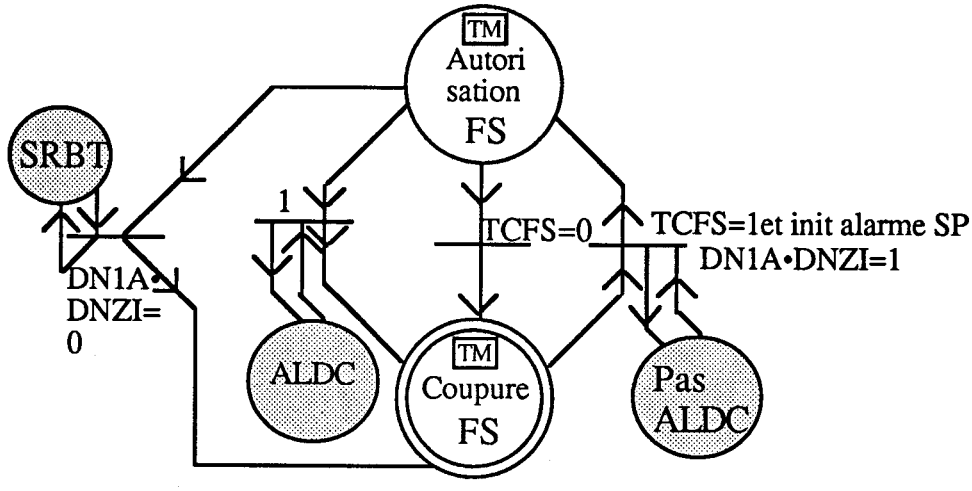


FIGURE 3-1-21

-Le cycle de rebroussement

Il suit l'évolution d'une rame lors d'un rebroussement.

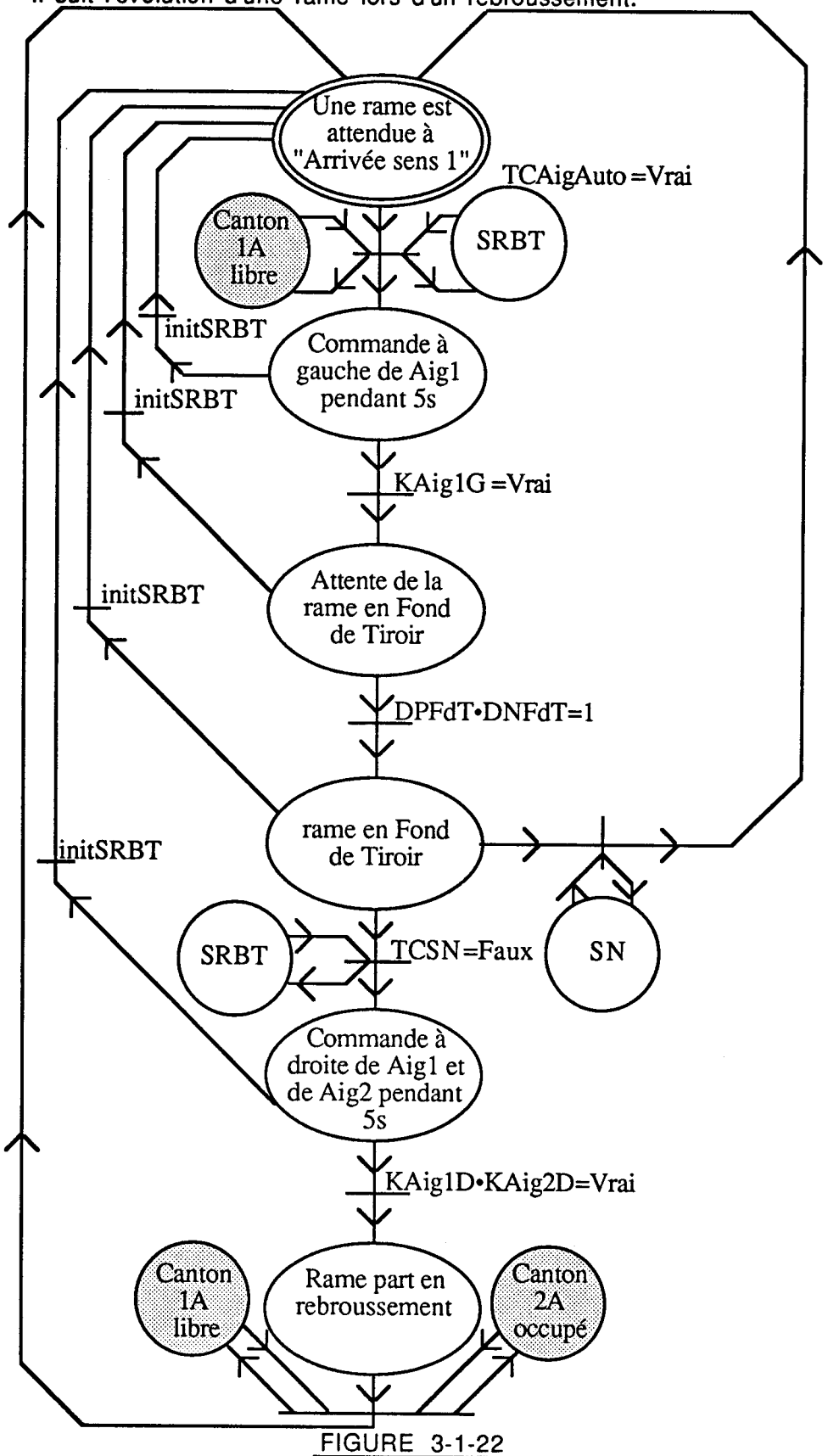


FIGURE 3-1-22

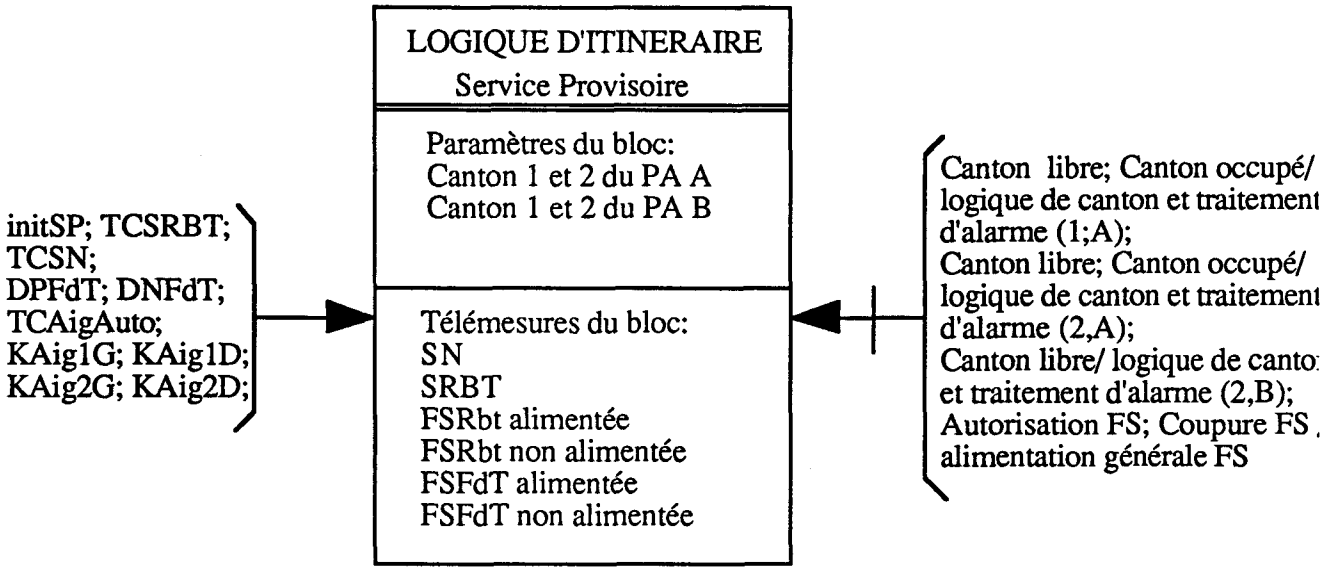


FIGURE 3-1-23

1-4 EXTENSION AUX RESEAU DE PETRI A PREDICATS

Dans l'exemple étudié, nous avons décrit la logique de canton. Cette logique est répétée pour chaque canton. Le réseau était donc réécrit autant de fois qu'il y a de cantons. Avec les réseaux de Petri à prédicats, on ne dessine qu'un seul réseau pour tout le PA. Les paramètres des arcs et des transitions désigneront le nom du canton concerné. En dehors de ces paramètres, le schéma du réseau n'est pas modifié. A chaque marque d'une place sera associé un paramètre.

Soit un PA à cinq cantons désignés de la manière suivante: Canton 1,n; C 2,n; C 3,n; C 4,n; C 5,n.

Les blocs 1 à 4 sont regroupés en un seul bloc pour lequel on paramètre les arcs et les transitions. Cette étape consiste à superposer les 4 réseaux identiques. Les 4 places ou arcs superposés sont distingués par leur paramètre. Le réseau de Petri du canton de sortie étant légèrement différent, on ne peut pas regrouper ce bloc avec les autres. Le schéma devient celui de la figure 3-1-24.

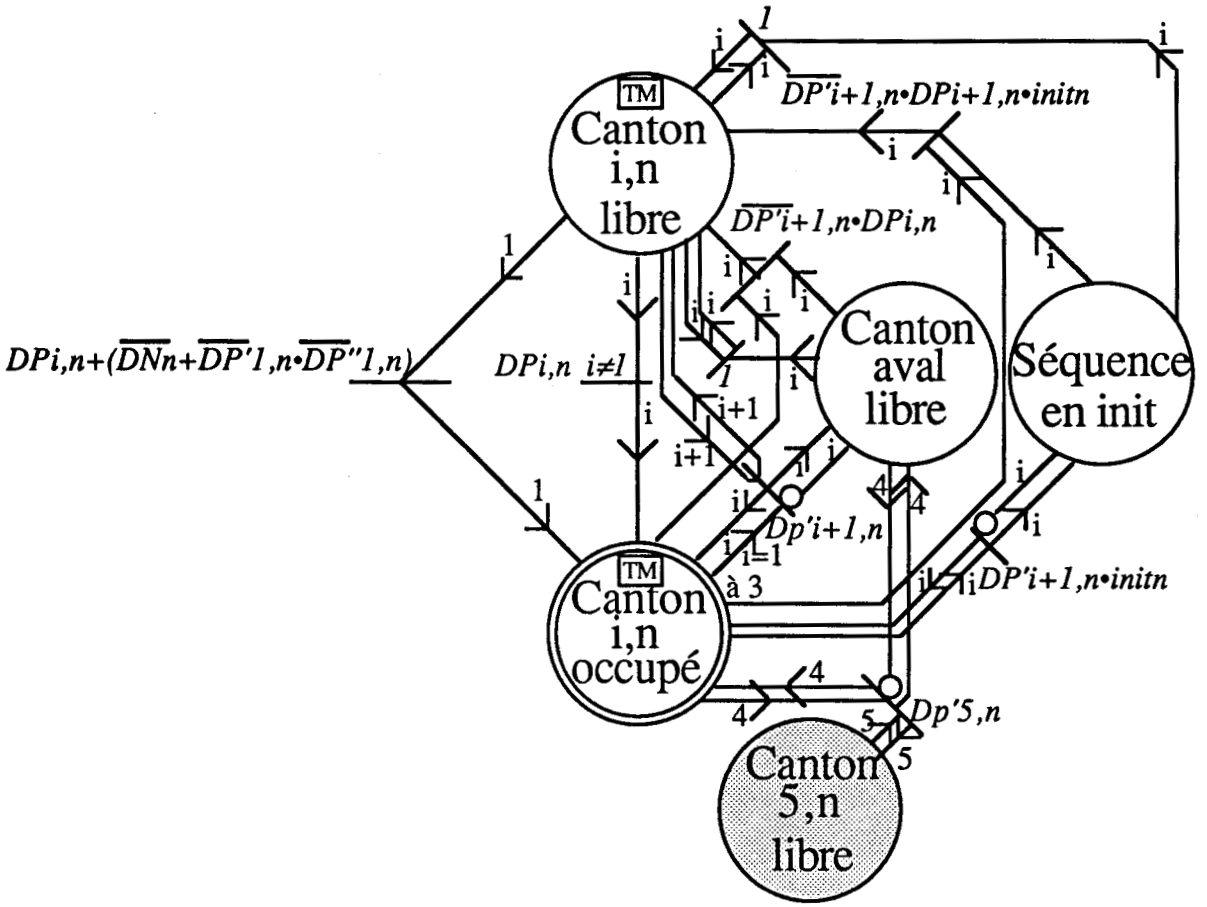


FIGURE 3-1-24

Dans ce schéma, i devient variable et peut prendre les valeurs de 1 à 4. La communication avec le bloc adjacent n'existe donc que pour le dernier canton.

Les réseaux de Petri à prédicats sont également caractérisés par: R=<P,T,Pré,Post>.

Pré est l'application d'incidence avant. Elle précise si il existe ou non un arc entre une place et une transition.

$$Pré : P \times T \rightarrow M \quad \text{où } M=\{0;1;\varnothing\}$$

Pour les réseaux de Petri à prédicats, à chaque fois que Pré(p;t)$\in\{1;\varnothing\}$, il faudra donner le paramètre correspondant. Ce paramètre précise le type de la marque à tirer. A chaque transition est associée une assertion pour la condition de franchissement de la transition. Elle dépend du paramètre i de la marque qui a été tirée.

Pour la fonction Post, on pondère de même tous les arcs par un paramètre . Celui-ci est fonction de la marque tirée et il détermine la marque à mettre dans la place d'arrivée.

L'ensemble de définition de ces marques doit être parfaitement défini et fini.

Dans l'exemple ci-dessous, l'ensemble des paramètres utilisés est: {1;4;5;i où $i\in[1;4]\cap\mathbb{N}$}

L'application Pré peut s'écrire sous forme d'une matrice. Dans l'écriture de la matrice nous utilisons les notations de la figure 3-1-25.

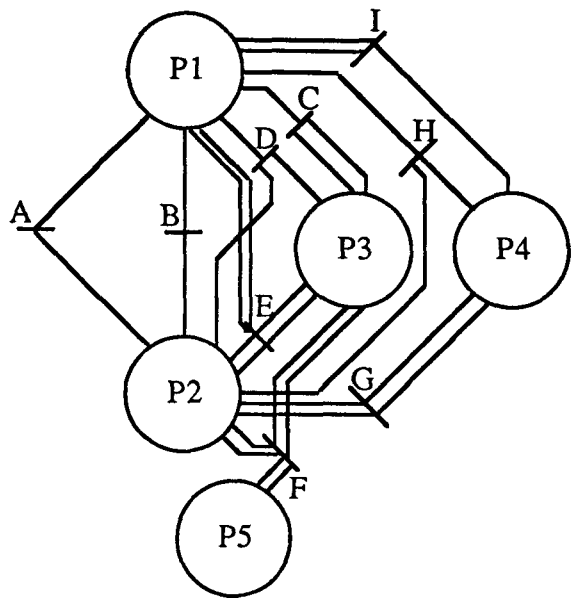


FIGURE 3-1-25

P \ T	A	B	C	D	E	F	G	H	I
P1	1 / 1	1 / i,i≠1	0	1 / i	1 / i+1	0	0	0	1 / i
P2	0	0	1 / i	0	1 / i	1 / 4	1 / i	1 / i	0
P3	0	0	1 / i	1 / i	Ø / i	Ø / 4	0	0	0
P4	0	0	0	0	0	0	Ø / i	1 / i	1 / i
P5	0	0	0	0	0	1 / 5	0	0	0

MATRICE PRE/PARAMETRES

FIGURE 3-1-26

La matrice Post s'écrira de la même façon.

La méthode des prédicats appliquées aux réseaux de Petri diminue globalement le nombre de places et de transitions du réseau. Elle ne simplifie pas le bloc en apparence. Quatres réseaux identiques ont été réunis en un seul et on a ajouté des paramètres sur chaque arc et dans les conditions de franchissement de chaque transition pour spécifier le canton qui est concerné. Les connections entre les blocs sont moins visibles et la validation est donc plus difficile.

Pour la programmation, on diminue globalement la taille du programme puisque les équations sont "paramètrées" et écrites qu'une seule fois pour les quatres cantons. Par contre le temps d'exécution du programme est le même. Les quatre transitions des quatre cantons ont été superposées en une seule mais pour cette transition il faut vérifier ses conditions de franchissement pour chacun des paramètres (4x1 = 1x4). La quantité de calcul est donc globalement la même.

La description d'un seul réseau et le nombre réduit de places écrites constituent l'avantage de ces réseaux; avantage intéressant lorsqu'on travaille sur un outil informatique limité en nombre de places. Encore faut-il que cet outil supporte les réseaux à prédicats.

Pour le cas considéré, les réseaux à prédicats sont pratiques d'utilisation si on veut visionner toute une ligne d'exploitation. Il deviennent lourd d'utilisation dès qu'on concentre l'étude sur un canton et ses deux cantons adjacents.

1-5 CONCLUSION SUR LES RESEAUX DE PETRI

La représentation graphique des réseaux de Petri permet souvent une bonne visualisation du système étudié. L'utilisation de modèle graphique a pour but d'éliminer le maximum d'ambiguïté. Il faut garder une taille raisonnable pour éviter d'écrire un modèle trop compliqué et inutilisable pour des personnes non initiées.

Dans l'exemple traité, on s'en est protégé en découpant la fonction en modules plus ou moins indépendants. Pour les modules, des règles strictes de construction sont à respecter. Cette méthode rend les propriétés plus faciles à vérifier.

La vérification des propriétés du réseau (vivant, borné...) traite les problèmes de blocage, de coalition et de cohérence des données. Si l'une de ces propriétés n'est pas vérifiée, cela peut signifier qu'une erreur de conception a été faite. Dans notre exemple, la propriété de vivacité a été montrée à partir de l'ensemble des marquages accessibles. Il est clair que dans la pratique cette énumération devient très vite impossible à cause d'une explosion combinatoire. Il est alors très intéressant de réduire le réseau avant de l'analyser.

Il existe d'autres moyens que l'analyse pour valider une spécification et détecter les éventuels erreurs de conception. Le plus utilisé est la simulation qui reste un outil puissant et convaincant malgré ses limites (un modèle ne peut jamais représenter exactement la réalité et des problèmes restent indécidables). Les réseaux de Petri présentent à ce niveau par rapport à d'autres modèles, l'avantage d'être visuels et dynamiques. La simulation permet ainsi de vérifier que le modèle correspond bien à ce qu'on veut qu'il fasse, les réseaux de Petri étant particulièrement adaptés pour traiter les problèmes de synchronisation et de fonctionnement en parallèle. Les réseaux de Petri ne peuvent par contre pas traiter les problèmes temps réel.
Nous reparlerons de la simulation dans la troisième partie de cette thèse.

Les supports informatiques des réseaux de Petri sont encore assez rares. Peu acceptent la saisie par des graphiques.
Par contre, les réseaux de Petri de base peuvent facilement se traduire en logiciel exécutable. Le passage aux réseaux à prédicats est plus difficile.

En conclusion les réseaux de Petri offrent une excellente décomposition des problèmes et facilitent la conception. Par la rigueur d'écriture, le modèle créé est clair et peu ambigu. Mais ils ne doivent pas être utilisés n'importe comment. Une méthode d'utilisation guide l'utilisateur pour diminuer la taille des schémas et garder une bonne lisibilité. Les réseaux de Petri sont un bon outil de spécification.

2- LE GRAFCET

Le GRAFCET, Graphe de Commande Etape Transition, a été crée par soucis d'uniformiser et de normaliser les méthodes graphiques utilisées pour la conception des automatismes [TOU82] [BLA82].

L'objectif de départ était de créer un outil unique de représentation des spécifications qui peut ainsi s'appliquer à toutes les machines existantes dans le commerce.

Aujourd'hui le GRAFCET est beaucoup utilisé dans les lycées techniques comme dans les bureaux d'études, et des automates programmables se sont adaptés à cette technique.

Pour notre application, après avoir étudié les réseaux de Petri nous avons constaté l'utilisation de nombreuses boucles. Ces boucles sont essentiellement dues à la fonction suivante: l'activation d'une place P conditionne le franchissement d'une transition, mais l'état de la place n'est pas modifié. Ce sont typiquement les transitions que l'on retrouve dans la transition entre les blocs.

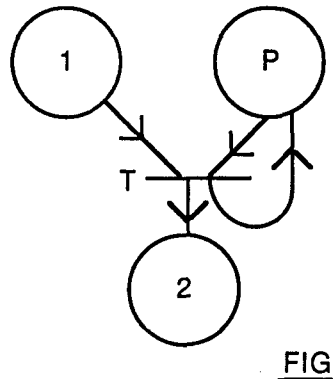


FIGURE 3-2-1

Or dans le GRAFCET, il est possible de faire intervenir l'état d'une place dans le prédicat de la transition.

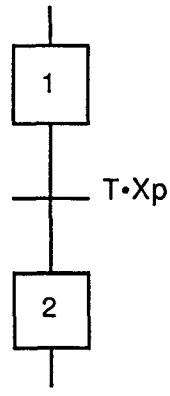


FIGURE 3-2-2

Le but de ce paragraphe est donc de comparer les performances des réseaux de Petri et du GRAFCET pour l'application aux automatismes d'un métro automatique.

2-1 DESCRIPTION DU GRAFCET PAR RAPPORT AUX RESEAUX DE PETRI

En général le GRAFCET se lit de haut en bas sauf si une flèche indique le contraire.
Les places ne peuvent comporter qu'une seule marque. Une deuxième marque qui arrive dans une place disparait.
Comme il a été expliqué dans l'introduction, le marquage d'une place peut apparaître dans le prédicat d'une transition. Cette caractéristique aura l'avantage d'éclaircir la représentation graphique.

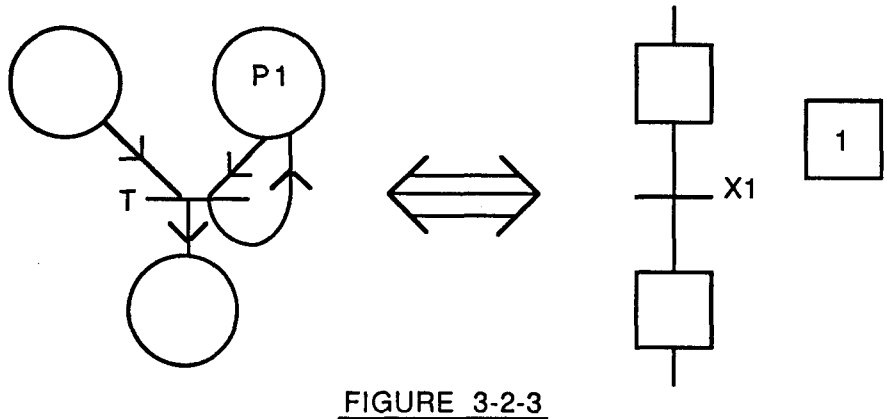


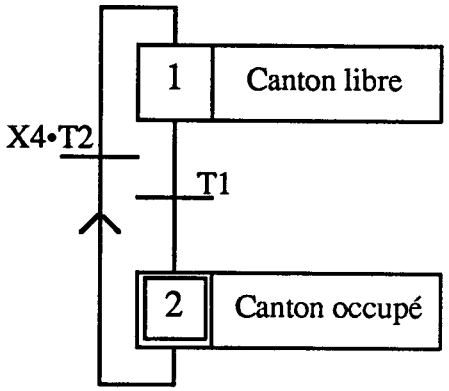
FIGURE 3-2-3

Les principales caractéristiques du GRAFCET sont reportées dans l'annexe 2. Les notations sont représentées par rapport aux réseaux de Petri.

2-2 SPECIFICATION DE LA LOGIQUE DE CANTON EN GRAFCET

Nous allons dans ce paragraphe spécifier la logique de canton en GRAFCET. Les schémas sont donnés ci-dessous.

2-2-1 LOGIQUE DE CANTON POUR UN CANTON D'ENTREE ET UN CANTON COURANT



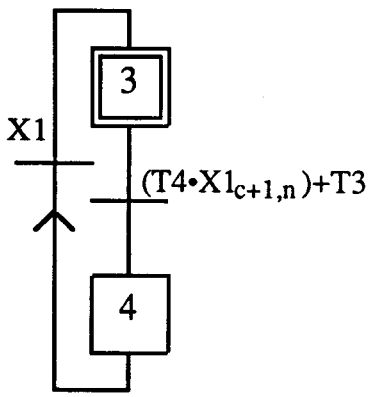
$T1 = DP_{c,n}$ si Canton courant

$= DP_{c,n} + \overline{DN_n + DP'_{c,n} + DP''_{c,n}}$ si Canton d'entrée

$T2 = \overline{DP'_{c+1,n}} + DP_{c,n}$

$T3 = DP'_{c+1,n} \bullet INIT_n$

$T4 = DP'_{c+1,n}$



$T5 = \overline{DP_{c,n} + DP'_{c+1,n}}$ perte des EAC ou recul sur canton amont.

$T6 = DP'_{c,n} + DN_n + \overline{DP''_{c,n}}$ pénétration sur canton d'entrée occupé

$T7 = \overline{DP'_{c,n} + DP''_{c,n} + DN_n}$ pénétration sans EAC sur canton d'entrée occupé ou non .

$T8 = DP_{c,n} + DP'_{c+1,n}$

$T9 = INIT_n$

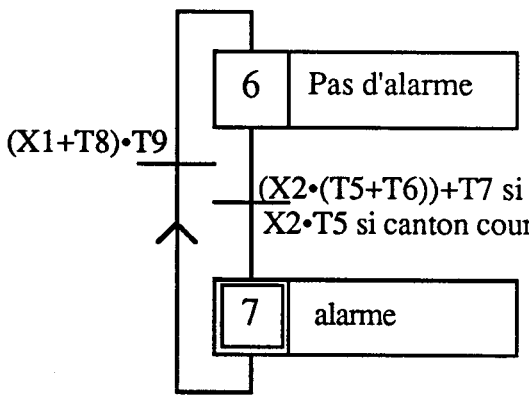
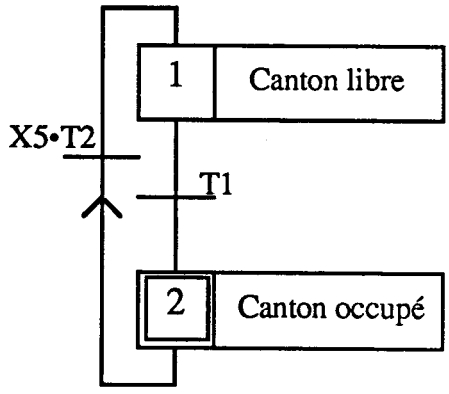


FIGURE 3-2-4

Si on compare ces graphes à un réseau de Petri, on peut remarquer qu'un seul réseau de Petri a son équivalence dans plusieurs graphes apparemment indépendants. Cette apparence est trompeuse puisque le passage de la place 3 à la place 4, par exemple, est conditionné par l'activation de la place 1 extérieure à ce réseau.

Par contre deux graphes n'ayant schématiquement aucune transition qui les relient, ne se modifient pas l'un l'autre. La marque d'un graphe ne peut pas être envoyée à un autre graphe. Cette propriété est intéressante pour les règles de communication que l'on s'était fixées lors de la construction de réseaux de Petri. (On accepte qu'un bloc "consulte" l'état d'une place d'un autre bloc par contre ce bloc ne doit pas modifier le marquage de la place de cet autre bloc)

2-2-2 LOGIQUE DE CANTON POUR UN CANTON DE SORTIE



$T1 = DP_{c,n} + DN_{n+1}$ Recul sur CS

$T2 = \overline{DP''_{1,n+1}}$ Rem: si perte des EAC à ce moment , alarme sur canton 1,n+1

$T3 = DP'_{1,n+1} \cdot \overline{DP''_{1,n+1}} \cdot \overline{DN_{n+1}}$

$T4 = \overline{DP'_{1,n+1}} \cdot DP''_{1,n+1} \cdot \overline{DN_{n+1}}$

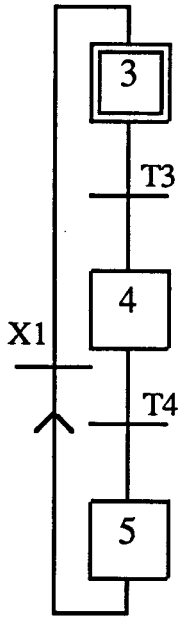
$T5 = \overline{DP_{c,n} + DP'_{1,n+1} + DP''_{1,n+1}}$ perte des EAC ou recul sur canton amont.

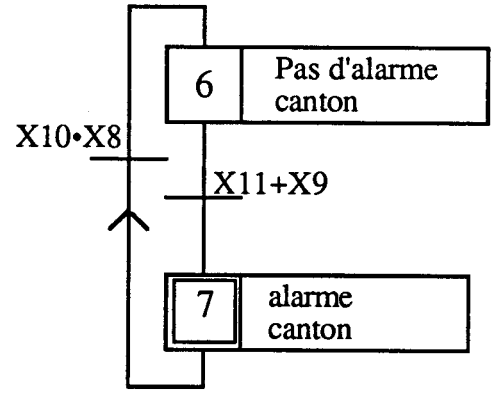
$T6 = \overline{DN_{n+1}}$

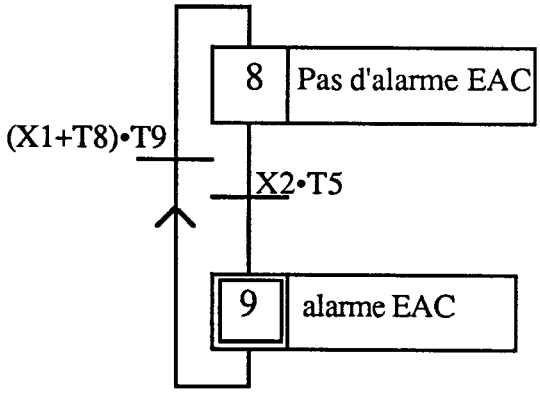
$T7 = DN_{n+1} \cdot INIT_n$

$T8 = DP_{c,n} + DP'_{1,n+1} + DP''_{1,n+1}$

$T9 = INIT_n$







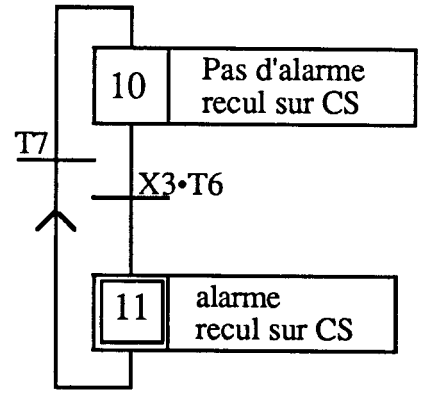


FIGURE 3-2-4

2-3 LE PARAMETRAGE DES ARCS: LE GRAFCET A PREDICATS

Nous pouvons noter que la fonction écrite en Grafcet est répétitive. En paramètrant les transitions, on superpose les modèles de chaque canton pour n'en faire qu'un seul.

Ce type de paramètre s'apparente à celui effectué sur les réseaux de Petri à prédicats.

Un paramètre sur l'arc indique le type de marque à tirer de la place. Les équations des transitions dépendront de ce paramètre. A la condition de marquage d'une place, on ajoutera le type de la marque que doit contenir cette place. La grande nouveauté par rapport aux Grafcets classiques est que chaque place pourra contenir plusieurs marques mais toutes de type différent.

Nous avons choisi pour cet exemple, un paramètre à deux dimensions. Le premier terme désigne le numéro de canton, le second le numéro du Pilote Automatique. Le paramètre sera de la forme (c, n).

Pour mettre sur un même schéma tous les cantons (entrée, courant, sortie), on distinguera les paramètres des cantons de sortie.

Exemple: Pour une ligne à trois PA où le PA1 a 3 cantons,
le PA2 a 2 cantons,
le PA3 a 4 cantons,

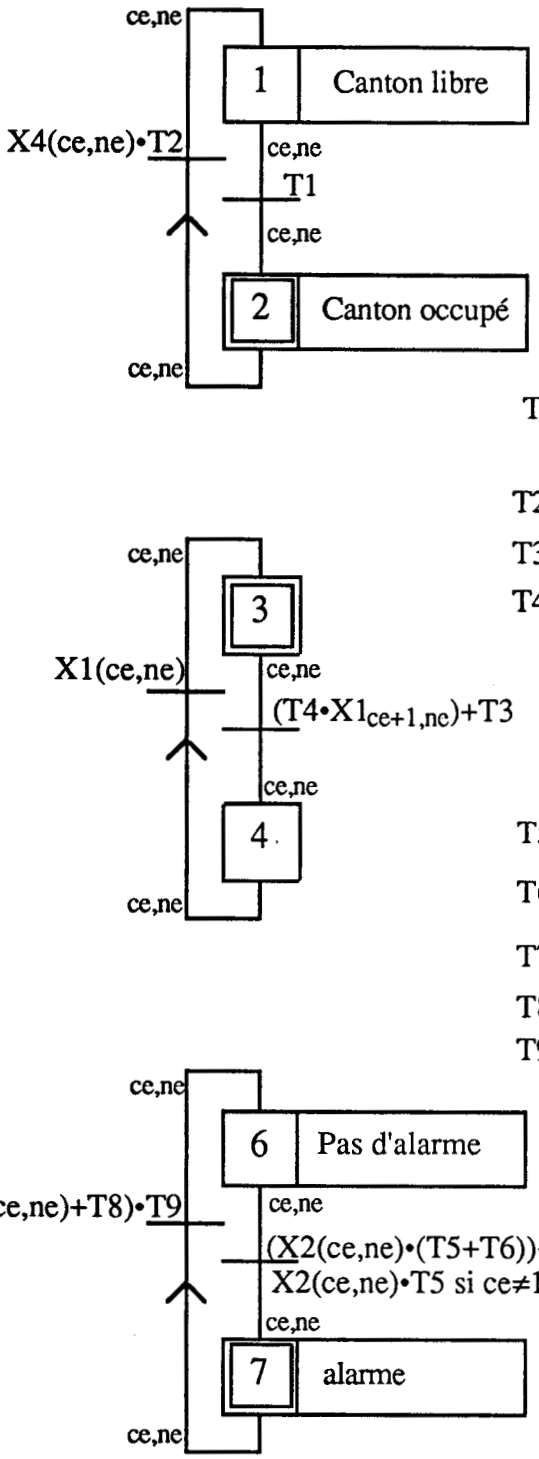
on distingue deux ensembles de paramètres:

CEC={(1,1);(2,1);(1,2);(1,3);(2,3);(3,3)} pour les cantons d'entrée et les cantons courants et

CS={(3,1);(2,2);(4,3)} pour les cantons de sortie.

Le GRAFCET paramètré de cette ligne est décrit dans les figures 3-2-6 et 7.

où (ce,ne) ∈ CEC et (cs,ns) ∈ CS



$T1 = DP_{ce,ne}$ si ce≠1

$= \overline{DP_{ce,ne}+DN_{ne}+DP'_{ce,ne}+DP''_{ce,ne}}$ si ce=1

$T2 = \overline{DP'_{ce+1,ne}} + DP_{ce,ne}$

$T3 = DP'_{ce+1,ne} \bullet INIT_{ne}$

$T4 = DP'_{ce+1,ne}$

$T5 = \overline{DP_{ce,ne}+DP'_{ce+1,ne}}$

$T6 = DP'_{ce,ne} + DN_{ne} + \overline{DP''_{ce,ne}}$

$T7 = \overline{DP'_{ce,ne}+DP''_{ce,ne}+DN_{ne}}$

$T8 = DP_{ce,ne} + DP'_{ce+1,ne}$

$T9 = INIT_{ne}$

FIGURE 3-2-6

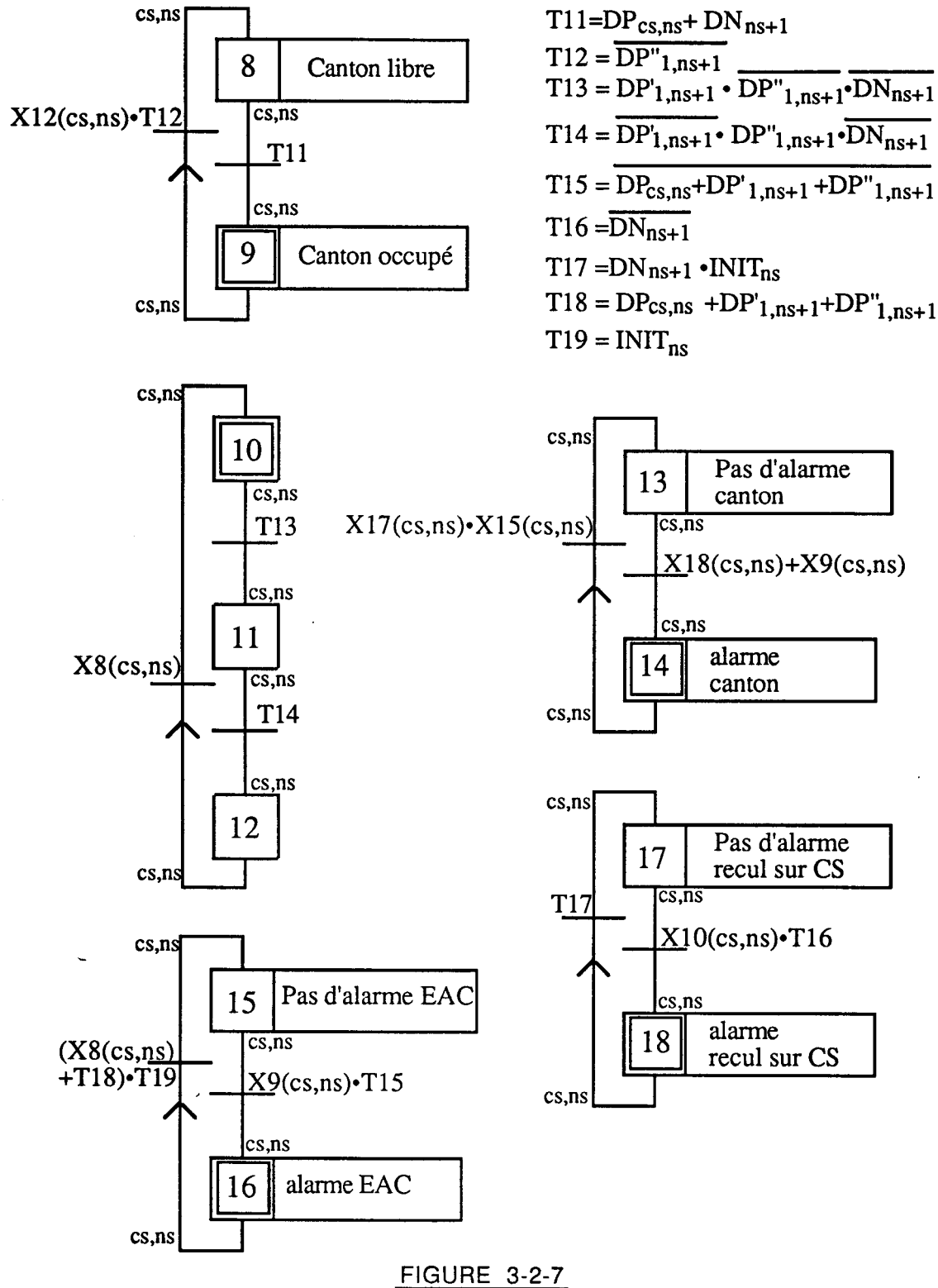


FIGURE 3-2-7

Ce modèle comprend la fonction logique de canton pour toute la ligne. Le schéma est beaucoup plus petit mais comprend pourtant la même quantité d'information. Il est par contre fortement chargé.

2-4 CONCLUSION

Il est évident que l'écriture du GRAFCET est claire. Le nombre de places et d'arcs est diminué puisqu'on peut conditionner la transition par les marques d'une place.

Un défaut est cependant à noter: dans une transition, il est possible d'entrer la condition "étape active" sans modifier cette étape et sans tracer schématiquement la relation entre cette place et la transition. Cette technique sert à éclaircir les schémas. Mais de ce fait, quand on active une étape, on ne voit pas immédiatement toutes les transitions qui deviennent ainsi franchissables et on ne dessine pas de façon claire les chemins utilisables.

Ce défaut n'est plus gênant si le Graphe est programmé puisqu'il peut y avoir une recherche automatique des étapes accessibles depuis un état ou des transitions conditionnées par un état.

Il semble que le GRAFCET se rapproche très bien de la gestion de programmes et de sous-programmes:

- un GRAFCET représente un morceau de programme
- l'activation peut être conditionnée par l'état d'un autre GRAFCET, ce qui est équivalent à un programme principal qui gère l'accès à plusieurs sous-programmes ou équivalents à une interruption envoyée vers le microprocesseur.

Par contre un ordinateur n'effectue pas plusieurs tâches simultanément. Il faut donc lors de l'écriture du GRAFCET garder à l'idée que les tâches ne peuvent pas être effectuées simultanément. Les actions sont alors traitées de façon séquentielle.

Pour la mise en œuvre du GRAFCET, une place ne pouvant avoir que deux états (0 ou 1 marque) se traduit par un booléen. Le calcul des transitions contenant des conditions de marquage des places se fait alors sur des booléens.

3-CONCLUSION

Nous avons décrit dans ce chapitre des outils de spécification des automatismes. Ces outils (SADT, Réseaux de Petri, Grafcet) permettent l'écriture de spécifications formelles. Celles-ci sont claires et sans ambiguïté.

Nous avons également mis l'accent sur les précautions à prendre lors de l'utilisation de ces outils si on ne veut pas obtenir une spécification illisible. Nous avons, de ce fait, décrit le découpage en réseaux de Petri et leur mise en blocs presque indépendants. Cette organisation visait à faciliter la vérification des propriétés du réseau et à faciliter sa validation fonctionnelle. Notre choix s'est donc porté sur l'outil Réseau de Petri grâce à sa souplesse et à son adaptabilité. La méthode d'utilisation que nous avons décrite, guide le concepteur pour créer une spécification de bonne qualité.

Une fois le modèle écrit, on ne peut pas être sûr qu'il soit sans erreur. Un effort important s'appuyant sur l'analyse préliminaire a pourtant été porté lors de l'écriture du modèle de spécification. Pour respecter les contraintes de sécurité et de disponibilité, il faut prouver que les critères de l'analyse préliminaire sont suffisants et sont vérifiés dans tous les cas. La conformité de la spécification ne peut être prouvée que par son test. Dans la troisième partie, nous nous intéresserons au test des spécifications pour les valider.

Nous décrirons également un outil de simulation de la spécification. Nous avons vu jusqu'à maintenant que l'écriture de la spécification nécessite des méthodes rigoureuses en raison de la complexité des automatismes imaginés.

Ces méthodes manipulent généralement un grand nombre de données. Un outil informatique apporte un support appréciable dans la manipulation de ces données.

La demande d'outils est au départ essentiellement due à l'importance qu'a pris le logiciel pour le développement d'automatismes dans le milieu industriel.

L'implantation de logiques micro-programmées dans des équipements embarqués (métros, avions, fusées...) a nécessité la mise au point de logiciels de si grande taille qu'ils doivent nécessairement être développés par une équipe et non plus par un seul homme. Ce travail ne pouvait réussir qu'avec l'aide d'outils de modélisation mais également de gestion.

Nous avons étudié certains des outils existants actuellement sur le marché. La synthèse de l'étude se trouve dans le document [FEK88].

Tous utilisent des méthodes gérant des bases de données. Ces méthodes ont le mérite de créer un environnement ouvert. La base de données est unique pour l'ensemble du projet. Celle-ci servira à la production d'une documentation fiable. Elle aidera la gestion du projet et l'assurance qualité du projet. Ces méthodes sont certainement les plus faciles à mettre en œuvre et peuvent être supportées par des petits systèmes.

Pour l'écriture de la spécification, ces outils utilisent souvent des méthodes de décomposition structurée des tâches. Partant de l'ensemble du problème à traiter, on le décompose en sous-fonctions permettant de résoudre ce problème. Chaque sous-fonction est elle-même découpée en "sous sous-fonctions" et ainsi de suite jusqu'à en arriver à une unité

suffisamment petite pour être traitée entièrement dans une seule tâche et très généralement par une seule personne. C'est une décomposition du type hiérarchique comme celle qui existe dans la description SADT.

Ces méthodes gèrent généralement les interfaces entre fonctions et à chaque niveau de décomposition. Les flots de données et flots de contrôle décrivent la manipulation des données. Les signaux en interface sont ainsi correctement définis et de la même façon pour les deux côtés de la chaîne (l'envoyeur et le récepteur).

Ces méthodes de décomposition structurée des tâches du système se prêtent parfaitement à l'analyse statique d'un système.

Par contre peu d'outils s'intéressent à la sûreté de fonctionnement et peu offrent également du prototypage. La plupart sont des outils de développement de logiciel. Le prototypage se fait sur des morceaux de logiciel, c'est-à-dire sur une sous-sous-fonction. Les tests engendrés sont alors du genre tests unitaires. Ce sont des tests qui ont essentiellement pour but de faire une vérification syntaxique du module écrit.

Le prototypage et la simulation sont donc peu disponibles dans ces outils. Cette absence s'explique sans doute par la spécificité de chaque application.

Les langages utilisés, la présentation de la spécification et des tests, la gestion des tests dépendent fortement de l'application envisagée et des méthodes de fabrication qui seront utilisées. Il semble impossible de faire un outil universel de simulation.

Par contre, le développement pour un projet donné d'un outil de simulation peut être fait de façon suffisamment modulaire pour que l'outil soit réadapté dans d'autres projets. D'autant plus que le développement d'un tel outil coûte cher, et que l'investissement ne peut être envisagé que s'il est amorti sur plusieurs projets.

Nous reviendrons sur les avantages et les performances d'un outil dans la dernière partie.

CHAPITRE 4
VALIDATION
DE
LA SPECIFICATION

1-INTRODUCTION

La phase de validation fonctionnelle suit l'écriture de la spécification. Elle consiste à vérifier que les critères sont respectés. Elle peut se concentrer sur les critères concernant les problèmes de sécurité. A la fin de cette étape, la spécification est validée et donc sans erreur par rapport aux objectifs que l'on s'était fixé. Elle peut alors partir en fabrication. Le risque de reprise est nettement diminué.

1-1 POURQUOI VALIDER ?

Le but de la validation est de montrer que la conception est correcte. Elle améliore et prouve la sûreté du fonctionnement du système [MYE79].

Comme nous l'avons déjà expliqué précédemment, il est indispensable de valider une étape avant de passer à la suivante.

Lors de la fabrication d'un meuble, il faut s'assurer que les dimensions indiquées sur le plan soient correctes avant de découper les planches. Une planche trop petite ne pourra pas être corrigée.

La validation doit être faite au fur et à mesure du développement pour plusieurs raisons :

- Plus une faute est vue tard, et plus elle sera difficile à identifier. Si beaucoup de composants entrent en jeu dans la transmission de l'information, on ne sait pas lequel induit l'erreur. L'expérience a montré qu'en matière de validation des logiciels, les problèmes les plus difficiles à mettre en évidence étaient ceux liés à un défaut de spécification [CHA89].
- Une fois que cette faute est identifiée, elle est difficile à corriger. En phase d'intégration par exemple, les spécifieurs ne peuvent pas toujours répondre immédiatement à l'anomalie qui s'est présentée. Il se pose des problèmes de disponibilité temporelle, ou même géographique du spécifieur. Or cette défaillance peut bloquer l'intégration tant qu'elle n'est pas corrigée. Le projet prend ainsi du retard.

D'autre part, la modification apportée peut remettre en cause les tests et les vérifications qui ont été faites par ailleurs. En effet, l'influence de la modification sur le système doit être surveillée. C'est ce qui s'appelle le test de non régression. Il impose souvent de refaire des tests déjà faits.

Une modification demandée tard coûte cher au projet et retarde l'avancement. Lors de la validation, on cherche à éliminer les fautes de conception. On veut en particulier prouver que les fautes du type C4, catastrophiques (voir le chapitre 1) sont toutes éliminées.

Le modèle de spécification écrit dans la première partie de cette thèse constitue une étape du projet. Ce document sera vérifié et validé avant de passer à la phase de conception détaillée. Nous allons tout d'abord fixer les objectifs de la validation.

1-2 QUE VALIDER ?

Une première façon de valider est de tester le principal le plus rapidement possible pour détecter les erreurs. On simule au hasard des cas d'exploitation qui, intuitivement semblent importants.

On arrête la validation quand le temps que l'on s'est imparti pour cette mise au point est écoulé. Dans ces conditions, on a toutes les chances de passer à côté d'un défaut gênant mais très particulier dans sa façon d'apparaître, de telle sorte qu'il soit difficile de s'en apercevoir par une analyse intuitive. De plus, on est incapable de prouver que la validation est exhaustive.

Pour le test de la spécification, on commence donc par se poser la question "Que veut-on valider et jusqu'à quelle précision doit-on aller ?"

L'analyse préliminaire a été écrite pour orienter cette validation.

D'une part, elle explique par rapport à quoi on va valider. Elle dit ce qu'on veut trouver dans la spécification et ce que doit réaliser le système terminé. Elle nous donne les critères nécessaires pour juger si le résultat d'un test est ou non acceptable.

D'autre part, l'analyse préliminaire liste les risques que l'on veut éliminer ainsi que leur niveau de criticité. Pour la sécurité du système, le risque de collision n'est pas acceptable. C'est le plus fort niveau de criticité : C4 qualifié de catastrophique. Un effort de validation plus important doit être mené pour le risque de collision plutôt que pour le risque de ne pas arriver à la station dans les temps. Ce dernier serait plutôt qualifié de mineur (C2).

C'est l'analyse préliminaire de risque qui mettra en évidence tous les risques contre lesquels on veut se protéger. La validation devra suivre le plan de cette analyse. Point par point, on testera notre modèle vis à vis du scénario que l'on regarde. Lorsque tous les points auront été prouvés, on pourra assurer que notre modèle se protège correctement contre les risques que l'on a envisagés. L'analyse préliminaire doit donc être complète pour assurer la sécurité du système [BEO88].

1-3 COMMENT VALIDER ? : LA SIMULATION

Pour tester la spécification fonctionnelle vis à vis de risques, on déroule des jeux de tests. Ceux-ci sont déduits de l'analyse préliminaire de risque. Ils comprennent les scénarios à dérouler sur le modèle et les résultats que l'on attend. Ces scénarios sont simulés sur le modèle formel. On regarde à chaque événement comment évoluent les états du modèle et on compare les résultats aux objectifs fixés au départ [LAM85].

La simulation manipule une grosse quantité de données, ce qui nécessite un support informatique. Suite à l'étude menée sur les "outils de spécification de systèmes numériques" [FEK 88], nous avions constaté que peu de ces outils offraient des possibilités de simulation et de tests dynamiques des spécifications.

Devant les possibilités apportées par la simulation, nous avons donc choisi de développer notre propre simulateur. Nous expliquerons ce choix dans le paragraphe sur la simulation.

1-4 QUE RESTE-T-IL DE LA VALIDATION ?

La question qui est posée ici est: "Quelles informations doivent rester d'une validation une fois qu'elle est terminée ?"

Les tests ont été déroulés sur la spécification. Les critères ont été vérifiés. Ils ont sans doute remis en cause certains aspects de la conception ou mis en évidence des contraintes pour l'exploitation. Cette validation peut également ne pas trouver de problème et accepter ce qui est fait tel quel.

Mais à partir du moment où on a décidé que la spécification fonctionnelle était correcte, doit-on conserver des traces des problèmes rencontrés et du travail effectué?

Les entreprises souffrent souvent d'un manque de documentation ou d'une documentation mal écrite. Or la documentation de la validation est nécessaire pour deux raisons :

- Elle peut faire l'objet d'un document contractuel vis à vis d'un organisme certificateur ou en cas de litige. Le document de validation est alors une preuve que le système est correct. Il contient tous les tests et les vérifications faites sur la spécification ainsi que des règles à respecter pour l'implantation ou la mise en œuvre. Il atteste le travail effectué et contient les résultats.

- Elle peut être ultérieurement relue au cas où on réutiliserait cette spécification dans une génération future de l'équipement. Supposons que l'on veuille réutiliser une spécification en y apportant quelques modifications: pour prouver que la nouvelle version est correcte, il faut pouvoir rapidement reprendre le document de validation. Les tests doivent donc pouvoir être déroulés à nouveau et les résultats comparés à ceux de l'ancienne version. Toutes les vérifications à faire doivent être contenues dans le document de validation. On ne devrait pas se reposer de nouvelle question en matière de sûreté de fonctionnement pour le nouvel équipement, sauf si l'analyse préliminaire est elle-même remise en cause.

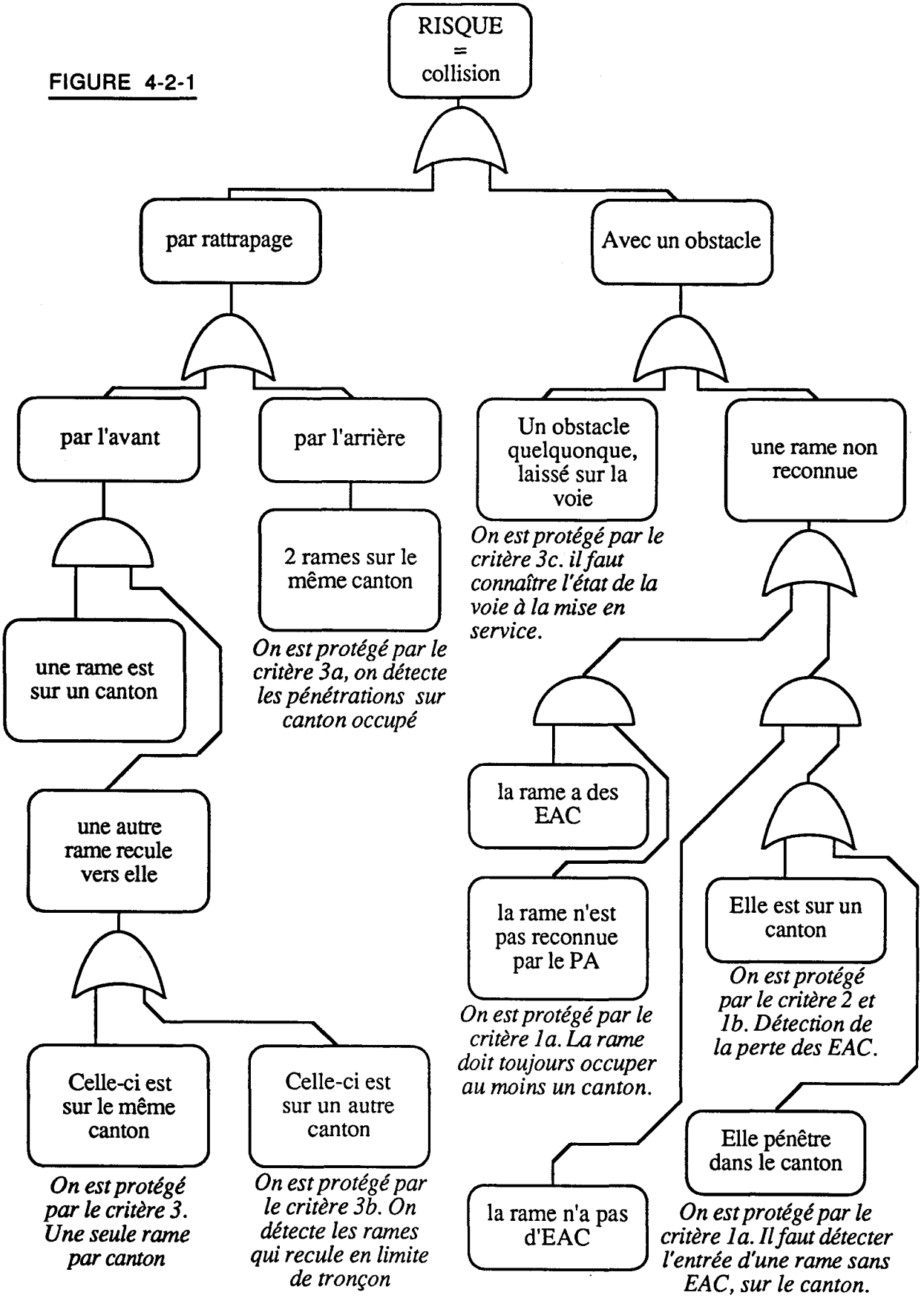
2-VALIDATION DE LA LOGIQUE DE CANTON

2-1 RESPECT DES CRITERES DE L'ANALYSE PRELIMINAIRE

Pour cette fonction, le risque qui a été défini dans l'analyse préliminaire est le risque de collision. Il est développé suivant l'arbre de la figure 4-2-1. Chaque extrémité de branche représente une configuration qui nous amène à ce risque. En dessous de chaque configuration, un commentaire renvoie aux critères que l'on a listés au cours de l'analyse préliminaire de risques.

Cet arbre prouve que le risque de collision est toujours évité à condition que les critères de l'analyse préliminaire soient respectés.

FIGURE 4-2-1


RISQUE = collision
par rattrapage
Avec un obstacle
par l'avant
par l'arrière
Un obstacle quelquonque, laissé sur la voie
une rame non reconnue
2 rames sur le même canton
On est protégé par le critère 3a, on détecte les pénétrations sur canton occupé
On est protégé par le critère 3c. il faut connaître l'état de la voie à la mise en service.
une rame est sur un canton
une autre rame recule vers elle
la rame a des EAC
la rame n'est pas reconnue par le PA
Elle est sur un canton
On est protégé par le critère 2 et 1b. Détection de la perte des EAC.
On est protégé par le critère 1a. La rame doit toujours occuper au moins un canton.
Celle-ci est sur le même canton
Celle-ci est sur un autre canton
Elle pénètre dans le canton
la rame n'a pas d'EAC
On est protégé par le critère 3. Une seule rame par canton
On est protégé par le critère 3b. On détecte les rames qui recule en limite de tronçon
On est protégé par le critère 1a. Il faut détecter l'entrée d'une rame sans EAC, sur le canton.

2-2 VERIFICATION DES CRITERES

L'étape suivante consiste donc à vérifier ces critères.
Prenons comme exemple la perte des Emetteurs Anti-collision d'une rame (critère 2) et vérifions que pour un canton d'entrée la logique réagit correctement. La perte des Emetteurs Anti-collision doit arrêter la rame, la logique doit donc se mettre en alarme.

Pour cette démonstration, nous utiliserons une simulation des réseaux de Petri. Nous modifierons les entrées du réseau et nous regarderons l'évolution du marquage.

Nous voulons vérifier notre critère dans tous les cas possibles. Nous allons donc commencer par définir tous les cas à examiner.

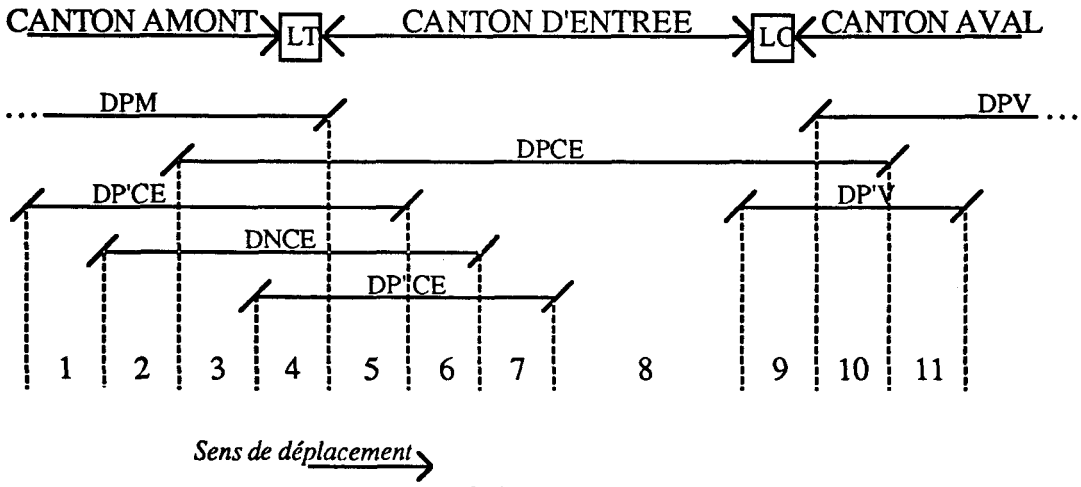


FIGURE 4-2-2

Dans le schéma 4-2-2, nous mettons en évidence 11 configurations différentes d'exploitation. Nous remarquons qu'avec 7 variables, le nombre d'arrangements est de 2^7, soit 128 cas. Mais toutes ces combinaisons ne sont pas possibles. Certaines ne peuvent pas être reproduites dans la réalité.
Il est évident que le test systématique des 128 cas n'est pas acceptable. Avant d'écrire les tests, il faut donc regarder dans le contexte de l'exploitation quels cas peuvent être réellement reproduits et ne travailler qu'à partir de ces cas. On peut également noter que ce nombre de cas dépend de la configuration et de la façon dont sont implantées les boucles. La réalisation du réseau n'apporte rien dans le choix des tests.

Dans l'exemple, nous avons donc 11 configurations différentes. L'ordre dans lequel ces configurations apparaissent, est important. Le véhicule se déplace de gauche à droite. Il se place d'abord dans la configuration 1 puis 2 puis 3 etc...

2-3 CONDITIONS DE REALISATION DES CRITERES

Nous ferons la vérification des 11 cas en considérant que le critère 3 est vrai: il n'y a qu'une seule rame par canton.

Dans un premier temps, nous n'utiliserons qu'une seule rame sur les trois cantons. Les résultats sont les mêmes avec plusieurs rames.

En effet quand une rame est dans une zone de transition entre deux cantons, elle occupe ces deux cantons. Il ne peut donc pas y avoir d'autre rame sur les deux cantons concernés. D'autre part, quand une rame est sur un canton, il peut y avoir d'autres rames sur les cantons amont et aval. Mais ces deux rames ne sont pas dans la zone de transition des deux cantons. Elles sont donc uniquement sur les boucles DP des cantons amont et aval. Ces données n'entrent pas dans la logique de canton. La présence d'autres rames dans la zone ne change donc rien à l'étude, en considérant qu'il n'y a pas de pénétration sur canton occupé.

Nous allons donc tester 11 cas différents représentés dans le tableau ci-dessous. On part avec une rame sur le canton aval. Le canton d'entrée n'est pas occupé puisqu'une rame est autorisée à y pénétrer.Dans chacun des cas, nous simulons la coupure des Emetteurs Anti-collision de la rame. Nous vérifions que la logique passe dans un état d'alarme.

état du détecteur \ cas	1	2	3	4	5	6	7	8	9	10	11
DPM=1	7	7	7	7							
DP'CE=1	7	7	7	7	7						
DNCE=0		7	7	7	7	7					
DP"CE=1				7	7	7	7				
DPCE=1			7	7	7	7	7	7	7	7	
DP'V=1									7	7	7
DPV=1										7	7

FIGURE 4-2-3

2-4 TESTS

Le test de la spécification est fait sur le modèle des Réseaux de Petri. Dans le tableau suivant, nous indiquons à chaque étape les places marquées du réseau de Petri. Les deux premières colonnes contiennent le marquage des réseaux dans le cas normal d'une rame qui suit le canton amont, le canton d'entrée puis le canton aval. Les deux colonnes suivantes donnent la réaction de la logique après la coupure des émetteurs en partant de l'état normal. Enfin la dernière colonne contient les commentaires sur les résultats de la logique. Elle indique en particulier si la réaction est acceptable ou non.

coupure des EAC

CAS	CANTON AMONT	CANTON D'ENTREE	CANTON AMONT	CANTON D'ENTREE	REMARQUES
1	canton occupé pas d'alarme	canton libre pas d'alarme	canton occupé alarme EAC alarme		OK, arrêt de la rame sur le canton amont
2	canton occupé tampon 1 pas d'alarme	idem	canton occupé tampon 1 alarme EAC alarme	canton occupé alarme	Le canton d'entrée devient occupé par l'occultation du DN. L'alarme est établie car le canton est occupé et la transition DPCE+DP'V=1 devient franchissable. OK, alarme sur les deux cantons.
3	idem	canton occupé pas d'alarme	canton occupé tampon 1 alarme EAC alarme	canton occupé alarme	OK.
4	idem	idem	idem	idem	OK.
5	idem	idem	idem	idem	OK.
6	canton ocupé tampon 2 pas d'alarme	idem	canton ocupé tampon 2 alarme EAC alarme	idem	OK.
7	idem	idem	idem	idem	OK.
8	canton occupé pas d'alarme	idem		idem	OK.

CAS	CANTON AVAL	CANTON D'ENTREE	CANTON AVAL	CANTON D'ENTREE	REMARQUES
9	canton libre pas d'alarme	canton occupé canton aval libre pas d'alarme		canton occupé canton aval libre alarme	alarme sur le canton d'entrée, rame arrêtée. OK.
10	canton occupé pas d'alarme	idem	canton occupé alarme	idem	OK.
11	idem	idem	idem	idem	OK.

FIGURE 4-2-4

Nous avons donc prouvé par ces scénarios que dans le cas nominal le critère 2 était vérifié. Le cas nominal signifie que cette panne de perte des EAC n'est assortie d'aucune autre.

La validation fonctionnelle faite est donc du type boîte noire. On ne s'intéresse pas à la façon dont est réalisée la fonction. Les jeux de scénarios doivent évidement reproduire tous les cas réellement possibles.

La logique sera ainsi fonctionnellement validée et le jeu de scénarios ou jeu de tests témoignent des vérifications faites. Ce travail est long et manipule un grand nombre de données. Il doit être supporté par un outil informatique. Dans le paragraphe suivant nous décrivons l'outil de simulation que nous avons développé.

Une fois la logique validée, on peut passer aux phases de conception détaillée et de réalisation.

Si des modifications sont décidées pendant les phases suivantes, elles remettent en cause la validation fonctionnelle. Pour s'assurer que ces modifications ne dégradent pas la fiabilité de la spécification, les tests fonctionnels doivent être redéroulés sur la nouvelle spécification fonctionnelle. Le support informatique permet de faire ce travail automatiquement et donc rapidement.

Enfin en intégration, on s'assure que le système fabriqué est conforme à la spécification en lançant le jeu de tests sur l'équipement fabriqué via une baie de tests.

3-SIMULATION

Lors de la validation de la spécification fonctionnelle, nous avons souhaité disposer d'un outil de simulation pour manipuler l'ensemble des données. Cet outil est destiné dans un premier temps à faciliter le travail du "valideur" et du "spécifieur".

Dans le cadre de nouveaux projets au sein de Matra transport, nous avons donc développé un outil de simulation des Pilotes Automatiques et de leurs interfaces. Cet outil est destiné aux projets en cours de conception et simule les spécifications écrites en porte logique ET, OU, NON. C'est la méthode qui est actuellement utilisée à l'intérieur de la société pour l'écriture des spécifications fonctionnelles.

Cet outil simule non seulement le comportement du Pilote Automatique, mais également de tous les équipements environnants: le véhicule, les appareils de voie, le poste central de contrôle, les capteurs... En simulation, un véhicule se déplace automatiquement le long de la voie comme pendant une exploitation normale. Le simulateur calcule alors les réactions des capteurs et des automatismes. L'état des signaux est observable en temps réel sur l'écran.

Nous avons développé le simulateur de Pilote Automatique Fixe sur un outil d'I.A.O.: Ingénierie Assistée par Ordinateur. L'IAO simule le comportement logique de cartes électroniques. Nous avons étendu les possibilités de cet outil en créant des composants représentant les objets des automatismes et de la voie et pouvant communiquer entre eux. Une association d'objets simule donc le déplacement d'un métro sur la ligne. Nous expliquons ce choix dans la suite de ce chapitre.

Ce simulateur des automatismes fixes fonctionne actuellement pour des spécifications écrites en porte logique ET, OU. Il est possible de faire évoluer ce langage de modélisation et d'utiliser un langage qui s'apparente au pseudo-code. Nous voyons à la fin du chapitre suivant quel principe ce simulateur doit évoluer pour supporter un autre langage de spécification.

3-1 CAHIER DES CHARGES

Un effort important a été investi dans l'écriture du cahier des charges. Il fallait prévoir dès le départ comment on souhaite utiliser cette simulation ainsi que les options dont on voulait disposer. Le simulateur étant destiné à la validation des spécifications, nous devions simuler des automatismes dans un état nominal ainsi qu'en cas de panne diverse d'un des composants. Ce cahier des charges était un compromis entre ce qu'il est possible de faire sur la machine support que l'on a choisie et ce qu'on souhaite faire en simulation.

Cet outil a donc été réalisé par deux ingénieurs de qualifications complémentaires. L'un formé au système d'IAO assura la programmation du logiciel, le

rédacteur de la présente thèse par la connaissance du système a assuré la définition des fonctions à remplir par le simulateur, le test du logiciel sur des exemples, ainsi que son utilisation sur un exemple concret.

Lors de l'écriture du cahier des charges du simulateur, nous avons cherché à être le plus proche possible de la réalité. Nous voulions d'une part simuler l'exploitation normale de la ligne, d'autre part, simuler des modes dégradés d'exploitation ainsi que des pannes sur l'automatisme spécifié ou sur son environnement (panne du train ou de ses antennes, panne d'un capteur, panne d'un composant,...). L'outil devait, à notre avis, garder cette souplesse, puisqu'il devait servir à simuler des configurations particulières et compliquées.

De plus nous voulions profiter des qualités temps réel du logiciel de base d'IAO. Elles nous permettaient de simuler les temps réels de réponse des composants et donc de traiter des problèmes de blocage et de synchronisation. Il est possible de programmer un temps de réponse des objets et de le faire évoluer aisément. Pour un signal provenant d'un capteur, suivant qu'il est transmis directement à l'équipement ou qu'il transite par plusieurs interfaces, le temps de transfert peut varier. Le simulateur, grâce à ses capacités temps réel, met en évidence les conséquences d'une modification du temps de réponse.

L'autre impératif était de faire une simulation modulaire pour qu'elle évolue avec les nouvelles générations de métro. Nous avons donc:

- séparé la simulation de voie, qui décrit l'environnement, de la simulation des automatismes. De cette façon, on n'est pas tenu par une méthode de spécification des automatismes. Celle-ci peut aisément évoluer.

- décrit notre simulation de voie par objet, de façon à distinguer: le train, la voie, et les capteurs sur la voie. Si on change de système, pour les générations futures de métro, il est aisé de modifier ou d'ajouter un objet. Le simulateur, quant à lui, ne sera pas modifié.

L'outil simule le comportement du Pilote Automatique fixe et de ses interfaces. Le train est avancé automatiquement. En fonction de la position et de l'état du train, le simulateur calcule la réaction des capteurs au sol. Certains dialogues avec la station sont également simulés. Quant à l'interface avec le Poste Central, elle devient l'interface entre le simulateur et l'opérateur pendant la simulation. Le Pilote Automatique recoit toutes ces informations et fait évoluer son comportement. Il envoie donc de nouvelles commandes vers le train et vers les aiguillages. Enfin l'opérateur peut introduire au cours de la simulation des pannes sur les équipements. Tous les résultats de simulation sont affichés à l'écran ou peuvent être enregistrés et archivés.

L'opérateur peut enregistrer et archiver toutes les commandes qu'il envoie. Ce jeu de test pourra par la suite être déroulé automatiquement.

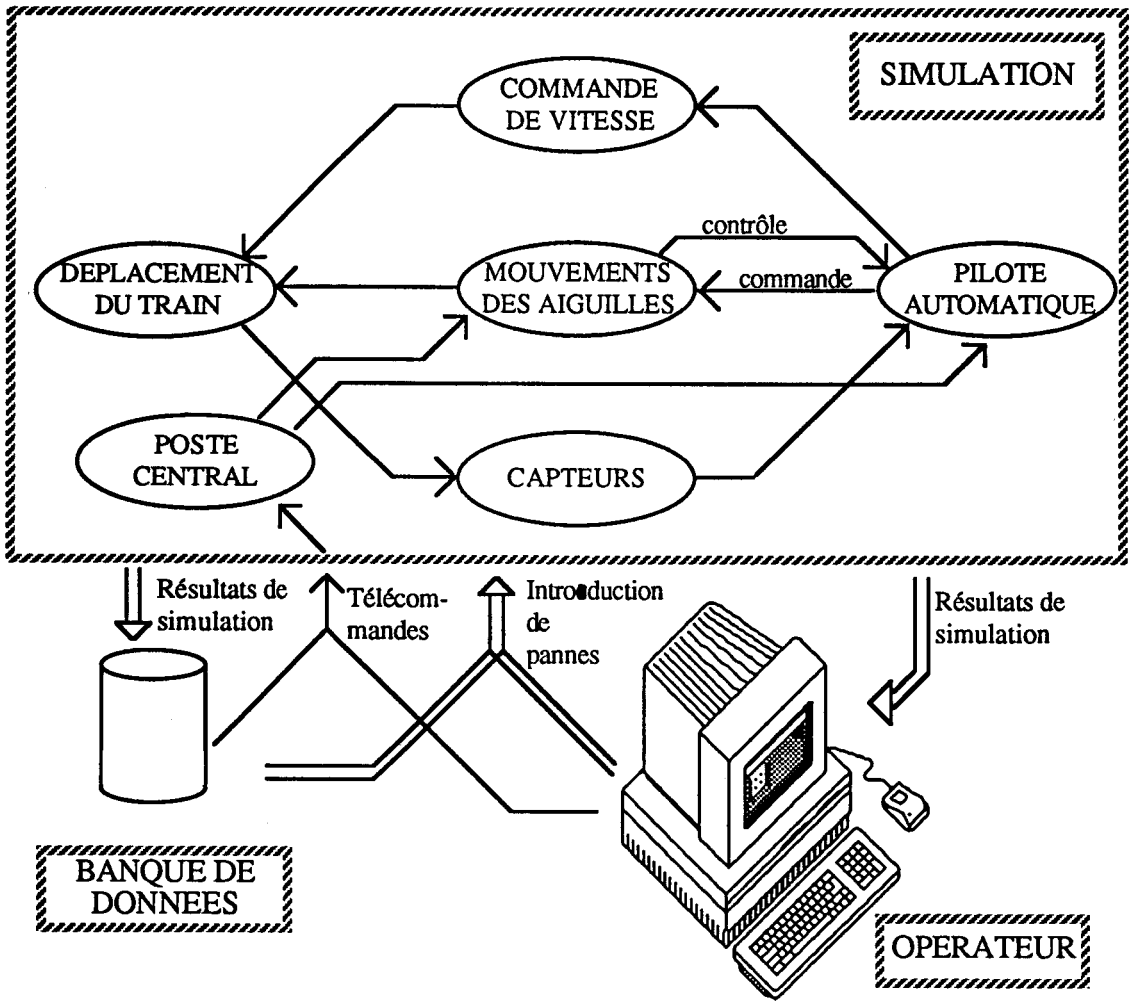


FIGURE 4-3-1

3-2 DEVELOPPEMENT

Pour développer ce simulateur, nous sommes partis d'un outil IAO. Cet outil simule le comportement logique de cartes électroniques. Il dispose d'une interface graphique de saisie disponible sur console graphique, d'un compilateur, d'un simulateur temps réel avec une présentation des résultats sur chronogramme. Le support que nous avons plus particulièrement utilisé est développé par la société VALID.

En simulation, l'outil d'IAO calcule les sorties en fonction de l'évolution des entrées. La logique du simulateur est de type "événementiel". Cela signifie qu'à chaque fois qu'une entrée varie le simulateur calcule la fonction de transfert du composant concerné. Les résultats logiques sont représentés par des chronogrammes fonction du temps. Il est possible d'observer toutes les équipotentielles auxquelles on a donné un nom. Une équipotentielle est la valeur d'un signal que l'on trouve sur un fil ou une piste entre deux ou plusieurs connexions. Ce sont les données qui sont transmises entre les composants.

L'outil d'IAO a une très forte puissance de calcul. Il simule le comportement de tous les signaux en temps réel. Il n'y a pas de risque d'oubli d'un signal ni d'erreur de calcul.

Les temps de réponse sont donnés par l'abscisse du signal sur le chronogramme. Le temps de calcul du processeur du simulateur est totalement transparent pour les résultats.

Cet outil d'IAO a l'avantage de pouvoir tester les problèmes de synchronisation. Il peut gérer les informations temps réel. Comme notre découpage a été fait de façon analogue à la réalisation et sans interprétation du système, nous pouvons saisir les temps de réponse de chaque composant et donc tester les problèmes dus à des retards ou à des pics. Ce point est important en période d'intégration et de maintenance. Il limite les interventions sur le site ainsi que les essais. Un problème particulier peut être reproduit sur le simulateur et analysé.

Enfin le logiciel de simulation offre des capacités de présentation et d'archivage des résultats ainsi qu'un dialogue permanent entre l'utilisateur et la machine. L'opérateur peut à tout moment introduire une panne ou envoyer une télécommande vers le modèle et en observer l'influence. La simulation est un dialogue permanent entre l'opérateur et la machine.

Le principe de fonctionnement de ce logiciel est le suivant: un composant électronique est équivalent à une boîte possédant des entrées, des sorties, et une fonction de transfert qui calcule la valeur des sorties en fonction des entrées. Cette fonction peut être créée à partir de plusieurs composants électroniques disponibles en bibliothèque ou peut être écrite dans un langage de programmation classique.

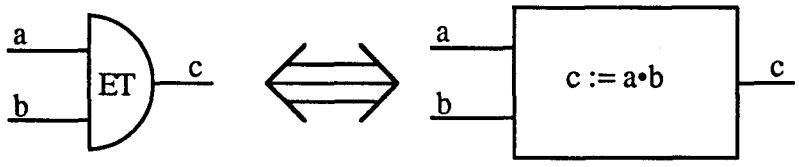


FIGURE 4-3-2

Dans le schéma de gauche, nous avons réalisé la fonction c=a•b à partir du composant ET de la bibliothèque. Dans le schéma de droite, nous avons créé un nouveau composant à trois broches et dont la fonction est c:=a•b, écrite en logiciel. Ces deux schémas réalisés de façon différentes, sont équivalents.

En utilisant le langage de programmation des fonctions (type Pascal), il est donc possible de créer toute sorte de composants manipulant des données logiques ou numériques. Les données numériques sont codées en binaire et transmises par des bus.

Les entrées et les sorties des boîtes sont ensuite reliées entre elles comme on relie, par des pistes ou des fils, la patte de deux composants. Ces fils sont les équipotentielles (ou signaux) du schéma.

En partant de ce principe, nous avons programmé des composants. La programmation nous permet de transmettre des données d'un composant vers l'autre mais également d'effectuer des calculs et de réaliser des algorithmes tels que "si...alors...", "tant que...faire...".

Nous avons créé un composant par objet que l'on retrouve sur une ligne: train, capteur, aiguillage... Ces objets ou composants calculent eux-mêmes leur état et transmette aux autres objets les informations qui sont nécessaires.

Par exemple, le composant de Voie a été dessiné suivant le schéma 4-3-3.

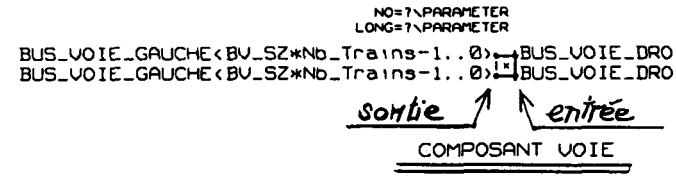


FIGURE 4-3-3

Il comprend une entrée (à droite) et une sortie (à gauche) matérialisées par des points. Les deux points de connexion inférieure ne sont que schématiques. L'entrée et la sortie sont de la taille d'un bus appelé Bus_Voie et dimensionné en fonction du nombre de trains à simuler (Nb_Trains) et d'un paramètre fixe (BV_SZ). La paramètre Nb_Trains est saisi sur un autre composant lors de l'écriture de la spécification. Le simulateur peut alors dimensionner son bus.

Les composants de voie sont reliés entre eux par le bus de voie. Leur connexion donne le plan de la ligne. L'objet voie simule le mouvement des trains sur la portion de voie le concernant. Il augmente de un pas la position du train chaque fois qu'il en recoit la commande. Il connait donc la position exacte d'un ou de plusieurs train sur sa portion. Lorsque le train arrive en fin de portion de voie, le composant Voie informe le composant VOIE suivant qu'un train arrive sur sa portion. Le nouveau composant VOIE prend le relais pour gérer la portion du train.

Le bus Voie transmet les informations sur la position du train par rapport au début de la portion de voie.

A ce bus Voie, on connecte les capteurs: DP, DN, FS,... Ils recoivent du Bus_Voie l'information de position du train. Ils peuvent alors calculer leur état occulté ou non, et envoyer l'information aux automatismes. Le bus transmet également les informations sur l'état du train. Le capteur peut ainsi savoir si le train a ou non des émetteurs Anti-collision et s'il émet sur les boucles. Inversement via ce bus, le capteur de commande de vitesse (FS) enverra au composant Voie son état. Il informera le composant voie sur les commandes d'autorisation de rouler et le sens de déplacement.

Au début de la ligne, on introduit le composant Train qui simule une rame de métro entrant sur la voie et se déplaçant sur la ligne. Ce modèle possède plusieurs modes de fonctionnement :

- simulation en mode manuel. L'opérateur au clavier simule alors les déplacements qu'il souhaite. Il commande la marche ou l'arrêt du train. Il choisit le sens de déplacement.

- simulation en mode automatique. Les mouvements du train sont asservis aux consignes envoyées par les capteurs FS. Ces capteurs indiquent l'autorisation de déplacement et le sens de déplacement.

Enfin le composant train a une option qui permet de simuler des pannes sur les antennes.

Dans la simulation de voie, les informations sont transmises à travers le Bus_Voie. Chaque composant ou chaque objet calcule son propre état.

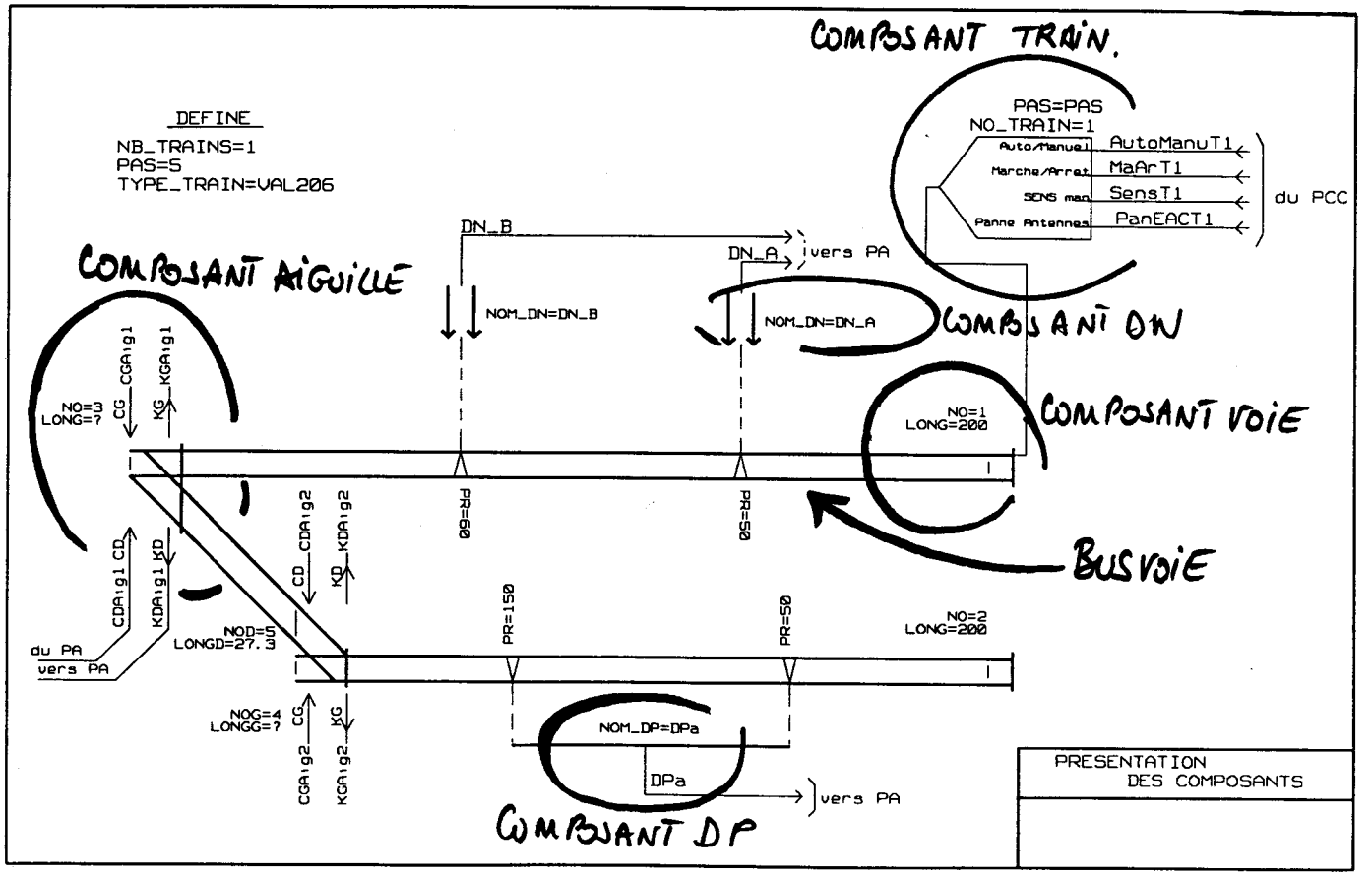


FIGURE 4-3-4

Dans le schéma 4-3-4, nous représentons pour exemple, la saisie d'une portion de voie qui peut être simulée.

On y retrouve:

- les composants *voies* et les composants *aiguilles* sont reliés par le bus voie pour former le plan de la ligne. Les composants aiguilles gèrent en plus la position de l'aiguille en fonction des commandes. Il dirige le train vers la portion qu'il faut.
- les composants *DP* et *DN* reliés au bus voie pour connaître la position du train et qui ont une sortie représentant leur état, envoyée vers les automatismes.
- le composant *train* relié au bus voie pour indiquer à tous les composants son état.

Enfin les automatismes sont saisis avec des portes logiques ET, OU, NON.
Exemple:

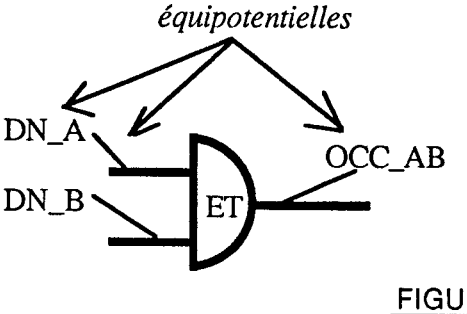


FIGURE 4-3-5

Grâce à l'interface graphique, nous pouvons constituer un document complet et clair de notre spécification. Elle détecte les fils mal connectés et fournit une liste d'équipotentielles. Ces outils aident à détecter les erreurs dues à la saisie.

Le développement du simulateur d'automatismes fixes a donc consisté à décrire un composant pour chaque objet de voie ou d'automatismes et à décrire les interfaces entre ces composants. Pour chaque composant, il fallait trouver l'algorithme qui réalise au mieux la fonction et est correct dans tous les cas possibles de simulation et d'exploitation de l'automatisme. La mise au point a consisté à vérifier ces algorithmes même dans des configurations exceptionnelles.

On retrouve donc les composants DP, DN, FS, Voie, Aiguille, Train, pour la description de la voie et les composants ET, OU, temporisations, bascules... pour les automatismes. Les interfaces transmettent les données entre composants. Elles servent également à envoyer des messages informant les composants qu'un état a bougé.

Avec tous ces composants, on saisie la spécification fonctionnelle d'un automatisme et on simule son comportement.

3-3 SAISIE DE LA SPECIFICATION FONCTIONNELLE

La saisie se fait en deux parties séparées. Il y a la saisie du Pilote Automatique Fixe et la saisie de la partie voie.

Dans notre cas, le Pilote Automatique Fixe a été saisi en portes logiques du type ET, OU, NON. L'ensemble des composants utilisés pour la spécification sont réunis dans une bibliothèque particulière. Celle-ci évite ainsi l'utilisation à tort de composants inexistants. Ce langage de spécification a l'avantage d'être très proche de la réalité. Il n'y a pas à faire de traduction de la spécification en un modèle et du modèle en un logiciel exécutable. Cet atout diminue le risque d'erreur lors de la traduction.

Dans cet outil, les automatismes peuvent également être spécifiés en pseudo-code tout en gardant le même simulateur. L'utilisation du pseudo-code s'approche très près du développement du logiciel.

Par contre, aucun langage formel de spécification fonctionnelle ne peut être supporté suivant sa méthode graphique.

Le modèle de voie est construit autour de ses points caractéristiques : appareil de voie, nez de quai,... Il est découpé en portions et les distances sont données en relatif par rapport à ce point.

Pour saisir ce modèle, on utilise les composants que l'on a créés et que l'on connecte entre eux. Les composants sont archivés en bibliothèque commune. En cas de changement dans leur définition, il suffit d'intervenir sur la bibliothèque. Ces composants peuvent évoluer facilement et, avec la bibliothèque, il n'y a pas de risque de travailler avec des composants obsolètes ou inexistants.

Le schéma de la zone de rebroussement et des capteurs implantés est représenté suivant les figures 4-3-6 et 4-3-7. La simulation peut se faire avec deux trains.

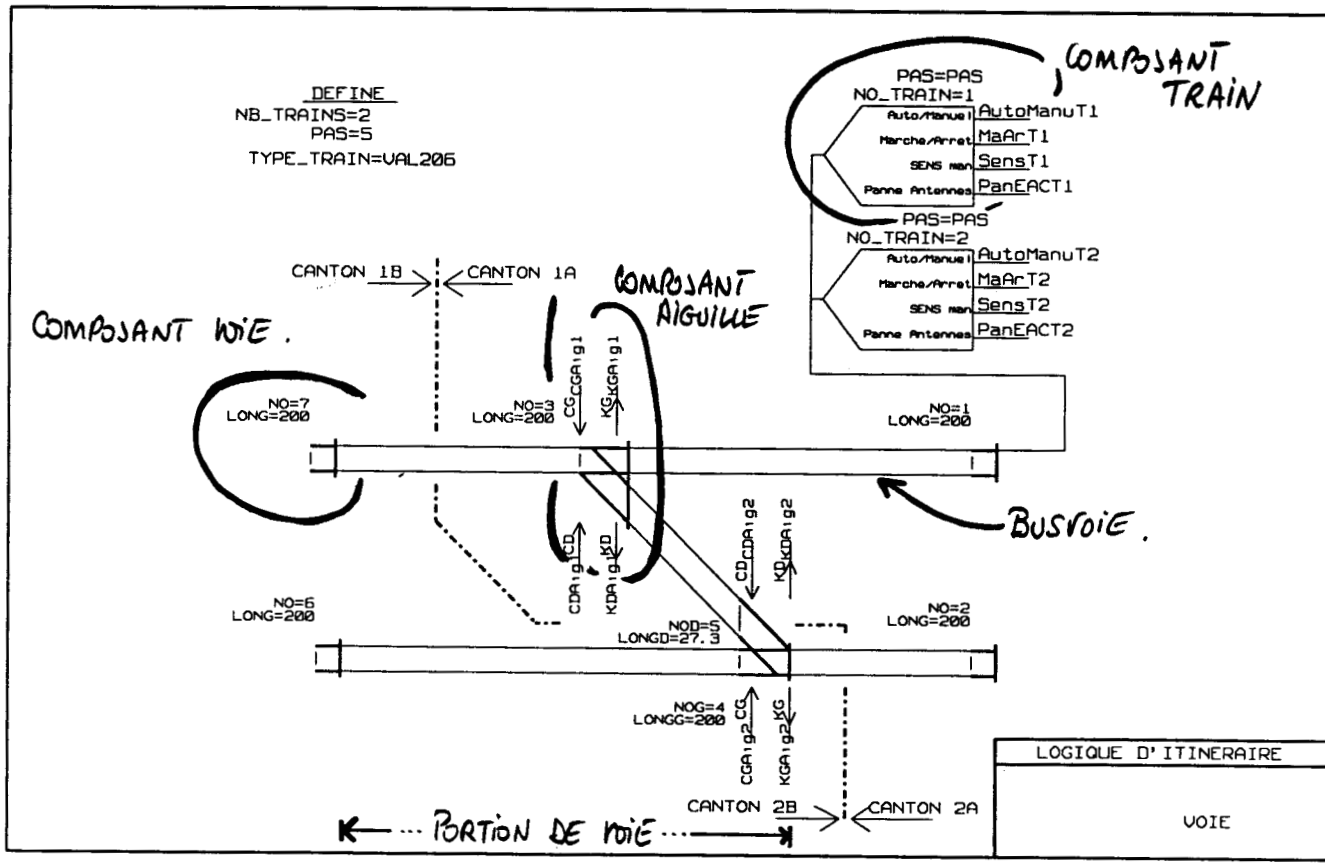


FIGURE 4-3-6

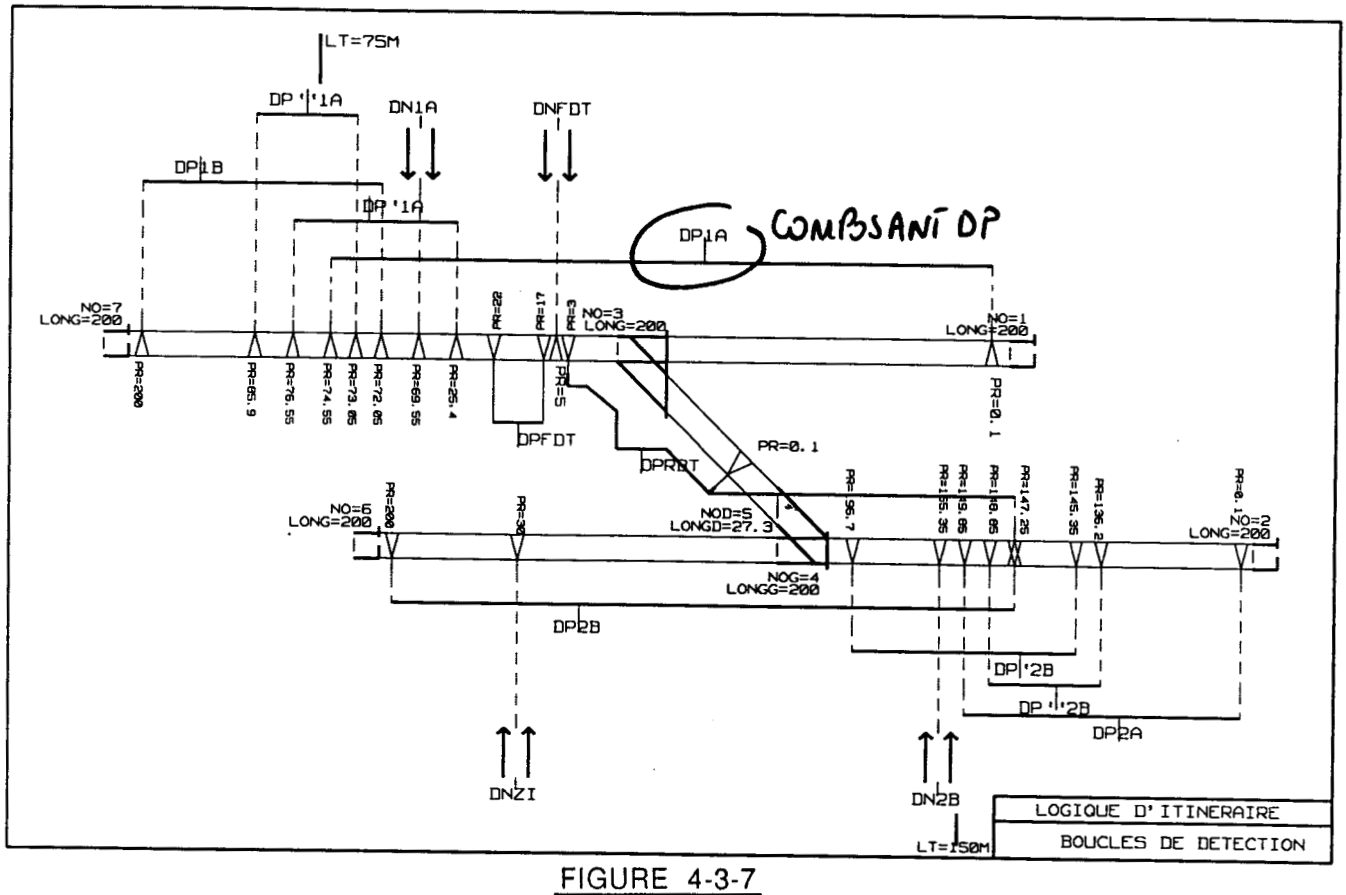


FIGURE 4-3-7

La spécification une fois qu'elle est saisie, constitue un document précis et complet de la spécification. On connait les fonctions décrites par les automatismes. On connait également les choix d'implantation des capteurs. Le simulateur est capable de gérer plusieurs versions de la spécification.

3-4 SIMULATION

3-4-1 FONCTIONNEMENT DU SIMULATEUR

Nous profitons du simulateur disponible sur l'outil d'IAO. Il s'utilise sur un poste de travail graphique avec un dialogue temps réel. L'opérateur entre des commandes du type: changement d'état d'un signal d'entrée, mise en place des points d'arrêt (ce sont des conditions logiques qui arrêtent la simulation dès qu'elles sont vérifiées), affichage des signaux. Il lance ensuite la simulation pour une durée choisie. Celle-ci s'arrête si elle rencontre un point d'arrêt ou dès que le temps est écoulé. Les changements d'état sont alors indiqués sur le chronogramme.

L'opérateur dispose de plusieurs outils pour observer les résultats:

- lire un signal ou un bus sur le chronogramme et connaître son état précis à n'importe quel instant;
- calculer un intervalle de temps entre deux événements;
- enregistrer une période d'observation soit sous forme de chronogramme soit sous forme d'une liste chronologique d'événements. Ce dernier fichier est utilisé pour la comparaison automatique de deux résultats.

Nous donnons ci-dessous un exemple de simulation de la zone de rebroussement.

Le train arrive par le canton 1A, s'arrête dès qu'il a désoccculté le DNFdT. Les aiguilles sont commandées. Le train repart dans l'autre sens de marche.

L'exemple ci-dessous nous montre comment les scénarios sont déroulés.

Les commandes sont passées en utilisant la souris et le menu affiché à l'écran. Toutes les commandes passées sont mémorisées dans un fichier que l'on peut archiver et par la suite dérouler afin de tester à nouveau ce scénario. Le fichier de commande mémorisé par le simulateur et après la mise au propre est donné dans le schéma suivant:

```
wavefor 0 3000
record all

op  cgaig1
op  kgaig1
op cdaig1
op kdaig1

op  cgaig2
op  kgaig2
op cdaig2
op kdaig2

radix 10
op troncons<7..0>
op positions<14..0>

op automanut1
op maart1
op senst1
op paneact1
op dnfdt
op dpfdt
```

```
op dp1a
op dp'1a
op dn1a
op dp''1a
op dp1b

op dprbt

op dp2b
op dp'2b
op dn2b
op dp''2b
op dp2a

op dnzi

logi 0
sim 0
```

```
DEPOSIT CGAIG1,1
sim 2570
set br dpfdt=1
DEPOSIT MAART1,1
sim 15600
set br dnfdt = 1
sim 2000

DEPOSIT CDAIG1,1
DEPOSIT CDAIG2,1
sim 50

DEPOSIT CGAIG1,0

DEPOSIT SENST1,1
sim 21500

wavefor 0 19000
plot 7000 19000
```

FICHIER DE COMMANDE DU SCENARIO DONNE EN EXEMPLE

Celui-ci ouvre les signaux que l'on veut voir apparaître sur l'écran (op = open) sous forme de chronogramme. Il commande ensuite à une rame venant du canton 1A d'avancer jusqu'à ce qu'elle soit en fond de tiroir, puis de repartir sur la voie 2 vers le canton 2B. La position des aiguilles est commandée par le scénario (Deposit CDAig1,1). Enfin, les résultats sont mis en mémoire (Plot 7000,19000). Ceux-ci apparaissent sous la forme de chronogramme nous donnant exactement la séquence qui a eu lieu sur les signaux.

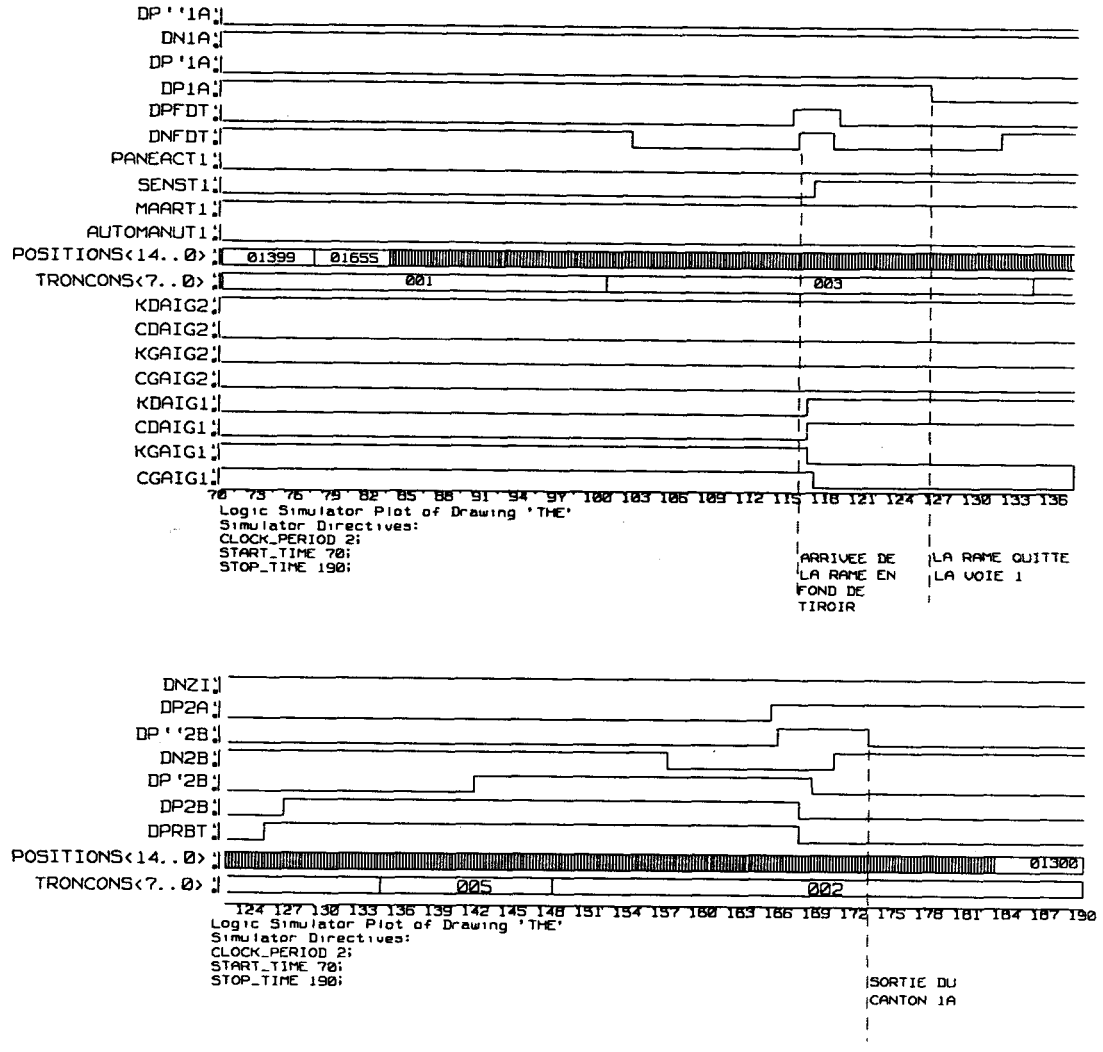


FIGURE 4-3-8

Dans le fichier de commande, il est tout à fait possible d'intervenir sur le signal panne des antennes (signal PanEACT1) et de simuler une coupure des émetteurs. On peut donc tester automatiquement des scénarios du même type que ceux que l'on a traités dans le deuxième paragraphe de ce chapitre.

Dans le chronogramme de la figure 4-3-8, le signal "Positions" est le seul qui soit illisible. En effet il représente un bus à 15 fils et tous ses états ne peuvent pas être représentés sur le chronogramme à deux états (1 ou 0). De plus il change trop souvent d'état pour que ceux-ci puissent être tous écrits sur le chronogramme. Par contre, pendant la simulation, il est tout à fait possible de connaître la valeur exacte du bus position à une abscisse donnée. Mais lorsque l'on mémorise un chronogramme, on utilise assez peu ce signal. Nous nous intéressons plus particulièrement aux positions relatives entre les fronts. Nous regardons par exemple la création d'une alarme par rapport à l'événement qui l'a crée (l'occultation d'un DN par exemple).

Le simulateur a l'avantage d'être facile d'utilisation grâce à la souris et à un menu pratique. Ce menu évolue en passant des commandes au clavier ou par un fichier de

commandes. On a donc pu lors de la création d'une interface utilisateur faire un menu adapté à nos besoins. On retrouve ainsi des commandes telles que:

- faire avancer le train sur une distance ou pendant une durée choisie
- lire la position d'un train
- mettre des points d'arrêt de la simulation en fonction de l'état d'un capteur, de la position du train...

3-4-2 VALIDATION DE L'OUTIL

Cet outil sert à prouver que l'automatisme spécifié est sans erreur. Il faut donc éviter que cet outil, par un défaut de conception, introduise d'autres erreurs dans la spécification ou masque des erreurs.

Nous avons, lors des premières utilisations, fait un test de notre simulateur.

Ce travail a consisté essentiellement à tester des configurations limites par des essais: association d'aiguillages; superposition de capteurs; portion de voie inférieure à la longueur d'un train... Nous avons fait à ce niveau une validation du type boîte noire.

D'autre part pour nous protéger complètement contre les erreurs dues à l'outil, les tests réalisés sur le simulateur doivent être déroulés ultérieurement sur l'équipement fabriqué. Une baie de test activera les entrées des cartes. Les résultats seront comparés à ceux obtenus avec le simulateur. Lors de divergence, on peut remettre en cause soit la fabrication, soit le simulateur.

3-4-3 SIMULATION POUR LA VALIDATION DES PA

La simulation consiste à dérouler automatiquement des scénarios et à vérifier que les résultats sont acceptables. Ces scénarios sont définis à partir de l'analyse préliminaire de risque.

Pour un risque donné, on est amené à analyser la réaction du PA dans des conditions bien particulières. On écrit alors un scénario simulant ce type de configuration. La définition des scénarios comprend également le critère d'acceptation de ce scénario. Ce critère définit, par exemple, à quel moment on doit commander l'arrêt de la rame compte-tenu de sa courbe de décélération et des limites imposées par le gabarit.

L'outil VALID aide à l'exécution des scénarios. **Les scénarios sont faciles à saisir sur le simulateur.** En effet, les données d'entrée du simulateur sont des commandes de déplacement du train. Après la sélection d'un itinéraire, on commande au train un déplacement pendant un temps donné ou jusqu'à un certain point. Ces points se représentent par une équation logique. Exemple: avancer jusqu'à ce que DP22 = 1.

A partir de là, le simulateur calcule automatiquement l'état des équipements de voie en fonction de la position du train, ainsi que des valeurs du PAF.

Le simulateur est :

- un outil ayant une **très grande puissance de calcul** qui est forcément **plus précise et meilleure que celle du cerveau humain.** Il libère l'opérateur de l'écriture des séquences sur les boucles lorsqu'on déplace un train, et du calcul des sorties sur le PAF.

- un **outil exhaustif** : ses calculs sont réalisés sur tout le PA. Il n'y a donc pas de risque d'oublier des changements d'état de certains signaux. Il a **une puissance de calcul sur une grande échelle.**

3-5 EXPLOITATION DES RESULTATS

Les résultats de chaque scénario sont regroupés dans des fiches.

La fiche de résultats est présentée sous forme d'un chronogramme des signaux concernés par le scénario. Ces chronogrammes permettent de vérifier que le PA correspond aux critères d'acceptation de ce scénario.

Les scénarios et leurs résultats peuvent constituer un document contractuel vis-à-vis du client ou de l'organisme certificateur. Ce document a l'avantage d'être complet. Il prouve que les scénarios ont été déroulés et contrôlés. Il garde une trace de tous les problèmes qui ont été traités. Ce document, bien que lourd, peut être facilement relu. Chaque scénario y est décrit par :

- les positions et les mouvements du train,
- les pannes envisagées et les télécommandes envoyées.

Un des objectifs est également de faire un test de non régression sur le PAF en cas de modification. Les fichiers de commandes des scénarios conservés sur bandes magnétiques sont déroulés à nouveau sur la spécification. Les résultats seront comparés à ceux contenus dans les fiches de résultats du document de sécurité. Cette fiche doit contenir les signaux modifiés et ceux qui ne doivent absolument pas être modifiés. Ce test de non régression est long à cause de la quantité de scénarios à dérouler, mais il peut être fait automatiquement.

3-6 AUTRES UTILISATIONS DE L'OUTIL

Nous avons surtout parlé de l'utilisation de l'outil pour la validation.

Il permet de tester rapidement des configurations non critiques (défaillances bénignes ou mineures). On vérifie surtout la disponibilité et l'exploitabilité de l'équipement. En validation très critique, il permet de dérouler des cas compliqués d'exploitation.

L'utilisation de l'outil aide à la compréhension de l'automatisme spécifié. L'outil montre comment fonctionne l'équipement, c'est donc un bon atout pour la formation à tous les niveaux: spécification, formation de l'exploitant, formation pour la maintenance.

Enfin, en intégration, il permet de chercher les causes des défauts rencontrés. Grâce à cette simulation temps réel, on peut reproduire la configuration mise en place. L'observation de tous les signaux internes peut aider au diagnostic d'une anomalie, d'autant plus que l'observation de signaux n'est pas toujours possible sur l'équipement réel et sur le site puisqu'il faut disposer de sondes et de points de mesure.

3-7 EXTENSION AUX RESEAUX DE PETRI

Nous avons pour cet outil de simulation saisi nos automatismes sous forme de portes logiques. Mais il est tout à fait possible de faire évoluer le langage du modèle de spécification. Nous avons vu que les portes logiques pourraient être remplacées par des modules écrits dans un langage proche de la programmation en Pascal. De cette manière, les réseaux de Petri pourraient être programmés. Nous allons regarder comment cette technique serait mise en œuvre.

La simulation de voie est gardée telle quelle. On conserve le même traitement pour les mouvements du train et la transmission des informations par le bus de voie. Les interfaces entre les automatismes spécifiés et la simulation de voie ne varient pas non plus. La simulation reste du même type que ce qui est effectivement réalisé sur le site.

Autour de cette simulation de voie nous intégrons alors un nouveau type de spécification.

La simulation du logiciel d'IAO est du type événementiel. Ce principe s'adapte parfaitement aux réseaux de Petri qui évoluent seulement lors de la réception d'un événement modifiant les transitions. Nous profitons de ce principe pour décrire les réseaux de Petri. Nous décrivons les réseaux en deux parties. Un premier module calcule les assertions des transitions pour un réseau. Ce module recoit les états des capteurs nécessaires pour calculer les transitions. Ces données sont booléennes ou numériques. Pour se référer au bloc des réseaux de Petri (se reporter au paragraphe 1-2-3 du chapitre 3), les états des capteurs correspondent aux variables venant du système.

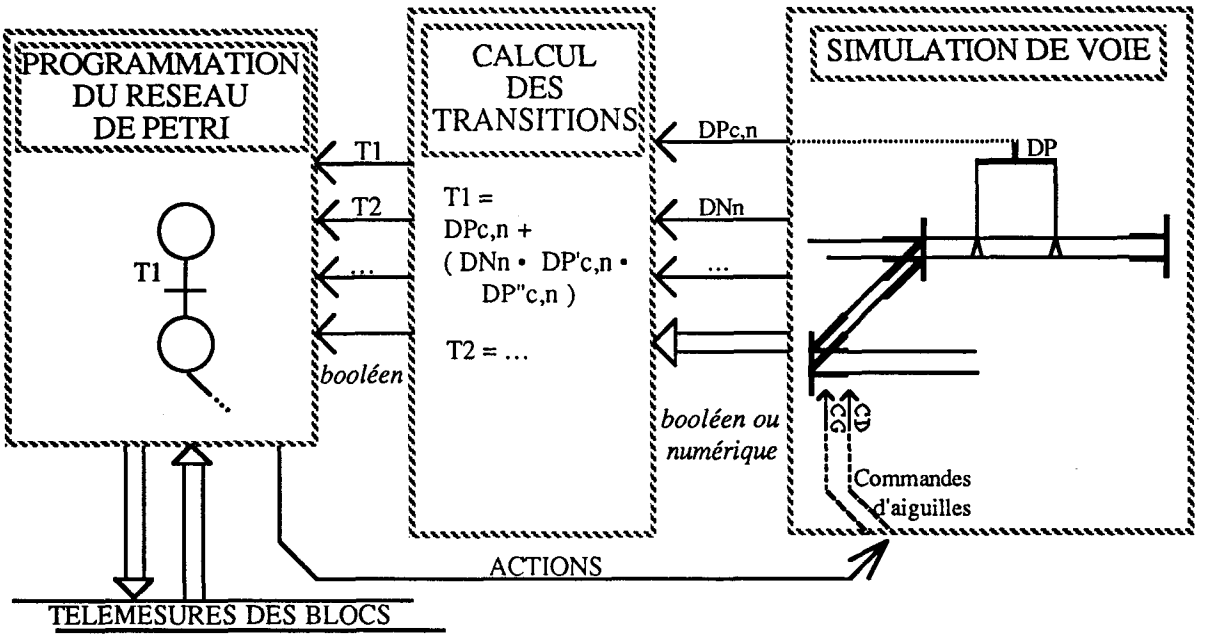


FIGURE 4-3-9

A chaque fois qu'un des états changera, ce module sera activé et les transitions seront calculées.

Les assertions des transitions sont ensuite envoyées vers le module contenant le réseau de Petri. Elles sont du type booléen. A chaque changement d'état d'une transition, le module du réseau de Petri est activé.

Ce module recoit également les télémesures des autres blocs c'est-à-dire l'état du marquage de places d'autres réseaux.

Etant donné que les places des réseaux simulés représentent des états du système, nous n'avons qu'une ou zéro marque par place et nous représentons ces places par un booléen.

Enfin les places des réseaux de Petri peuvent lancer des actions et donc agir sur la simulation de voie.

La programmation des réseaux de Petri est actuellement courante et a déjà été étudiée dans différents ouvrages.

4-CONCLUSION

Malgré les recommandations faites au départ et les moyens employés pour écrire la spécification, il est nécessaire de vérifier que la conception du modèle est sans erreur et en particulier qu'elle ne risque pas de créer des situations contraires à la sécurité.

Cette vérification est faite en déroulant un jeu de tests sur la spécification. Ce jeu de tests est inspiré de l'analyse préliminaire. On s'assure que chaque critère est respecté dans toutes les situations d'exploitation. Un test est composé d'un scénario, des critères d'acceptation du test, et des résultats nécessaires pour se conformer aux critères d'acceptation. L'ensemble des tests constitue le jeu de tests. Ce jeu a permis de vérifier l'état de la spécification qui devient alors une spécification validée et bonne pour fabrication. Cette spécification est sans erreur.

Ce même jeu de tests sera ultérieurement déroulé sur l'équipement fabriqué. Cette étape assure que la construction est correcte. Elle nous protège contre les erreurs d'interprétation du modèle. Elle prouve que le passage du modèle à la réalisation n'a pas introduit de nouvelles erreurs.

Pour dérouler l'ensemble de ces tests, nous avons utilisé un outil de simulation. Il nous aide à calculer les réactions du système et à manipuler les fichiers de données. C'est également un bon outil pour archiver les problèmes étudiés et les résultats obtenus sous forme de jeu de tests.

Enfin, parce qu'elle favorise la compréhension de l'automatisme spécifié, la simulation est un outil utile à l'intégration et à la formation.

CONCLUSION

Le cadre de notre travail se situe dans les premières années de développement d'un produit (matériel ou logiciel). Nous avons cherché à mettre au point une méthode d'écriture de spécification qui puisse faciliter ce travail par l'absence d'erreur et permettre la validation fonctionnelle du produit fini.

Dans cette perspective, nous avons analysé et évalué les outils et les méthodes les plus employées afin d'en dégager "la méthode" qui s'adapterait à la spécification et à l'analyse des automatismes de sécurité de type pilote automatique dans les transports terrestres.

Cette analyse a montré que la méthode SADT (Structured Analysis Design Technique) facilite la décomposition et l'analyse des automatismes mais doit être accompagnée d'outils de type GRAFCET, Réseaux de Petri, Automates à Etats Finis...pour être efficace.

Nous avons étudié ces outils ainsi que des moyens informatiques adaptés (IAO, outils d'aide à la spécification).

Le GRAFCET est un bon outil grâce à sa compacité. Le modèle représenté est plus petit qu'en réseau de Petri. Par contre la compréhension n'est pas aussi immédiate que dans les Réseaux de Petri qui eux représentent de façon schématique tous les chemins.

Notre choix s'est porté sur les Réseaux de Petri qui avec ses extensions représentent le modèle le plus fin et le plus souple à manier ainsi que le plus complet. En effet, les relations entre les événements sont toutes visibles par les transitions.

Il est à noter qu'une utilisation anarchique de ces potentialités nuit à la lisibilité du modèle et à l'analyse de la sécurité des automatismes complexes. Pour ce faire, il est nécessaire de structurer le modèle. Nous avons émis une méthode de construction de ces réseaux qui consiste à découper la fonction en petits modules et leur lecture en est plus facile.

Ce découpage nous a amené, pour faciliter la synchronisation et la lisibilité, à adopter des règles de construction: un module ne doit pas modifier l'état d'un autre module; un module ne peut que consulter l'état de variables de modules adjacents. Les Réseaux de Petri et l'évolution de leur marquage sont ainsi observables module par module.

Par cet outil de spécification et par la méthode mise en place, nous avons réalisé un modèle complet de l'automatisme à spécifier. Il a été écrit avec une méthode rigoureuse et réalisé à partir de l'analyse préliminaire.

Celle-ci définit les besoins et les exigences aussi bien du point de vue disponibilité que sécurité. Malgré ces recommandations faites au départ et les moyens

employés, il est nécessaire de vérifier que la conception du modèle est sans erreur et en particulier qu'elle ne risque pas de créer des situations dangereuses.

Cette vérification est faite en déroulant des jeux de tests sur le modèle. On prouve ainsi que le comportement du modèle est bien celui qu'on avait défini dans l'analyse préliminaire.

Pour chaque test, on mémorise le scénario déroulé et les résultats de ce scénario. On établit également un critère d'acceptation de ce scénario qui définit les conditions nécessaires pour affirmer que les résultats du test sont bien ceux qu'on attendait. Un test est donc composé d'un scénario, des critères d'acceptation et des résultats à obtenir pour se conformer aux critères d'acceptation.

L'ensemble des tests déroulés sur la spécification constitue le jeu de tests.

En simulant des tests sur le modèle, on manipule un grand nombre de données. il est donc indispensable de disposer d'un support informatique. L'outil que nous avons développé a l'avantage de faire la gestion des données et d'avoir une grande puissance de calcul.

L'outil gère l'ensemble des données du jeu de tests. Il sera en interface avec la baie de test pour dérouler ultérieurement les scénarios sur l'équipement fabriqué. Il offre un prototypage tel que la spécification est proche de la réalité et simule le comportement de l'automatisme spécifié et de l'environnement. Cette étape de validation assure que la construction est correcte et nous protège ainsi contre les erreurs d'interprétation du modèle.

Nous nous sommes également intéressés aux outils disponibles sur le marché. Beaucoup offrent une aide à la spécification. Ils orientent l'utilisateur dans le découpage de ses fonctions. Ces outils se situent plutôt au niveau organisationnel de la spécification. Ils conduisent souvent le concepteur à construire des Réseaux d'automates communicants.

Par contre nous n'avons pas, parmi ces outils, trouvé ceux qui répondaient aux exigences de sécurité pour obtenir une spécification sûre.

ANNEXE 1

DEFINITION DE

LA SURETE DE FONCTIONNEMENT

1-FIABILITE, SECURITE, MAINTENABILITE, DISPONIBILITE.

La sûreté de fonctionnement (= dependability) d'un système est la qualité de service qu'il délivre. Une qualité telle que les utilisateurs du système puissent lui accorder une confiance justifiée. L'objectif est de réaliser un système où la faute est naturelle, prévue et tolérable [LAP85].

L'ensemble des définitions est donné dans le chapitre 1.

2- ALTERATION DE LA QUALITE DE SERVICE

2-1 Définition de l'altération de la qualité de service [TROY86]

La qualité de service peut être altérée par des circonstances indésirables qui créent des défaillances dans le système.

La **défaillance** est un dysfonctionnement qui se manifeste sur le comportement du système. C'est le passage d'un état de fonctionnement à un état de non fonctionnement.

Une défaillance à pour origine une **erreur** dans le système fabriqué. L'erreur est la partie d'un équipement qui est non conforme à ce qu'elle devrait être pour que le système puisse produire le service attendu. Elle modifie la structure du système.

L'erreur est provoquée par une **faute** dont la nature est:

- physique - phénomènes physiques adverses, intérieurs ou extérieurs.

- humaine - imperfections humaines qui sont des fautes de conception ou des fautes d'interaction (violation des procédures d'exploitation ou de maintenance).

Les fautes sont classiquement réparties en trois catégories:

- **les fautes aléatoires**: absentes à la conception du système, elles apparaissent à un moment plus ou moins aléatoire. Après être passées par une phase intermittente, elles deviennent permanentes.

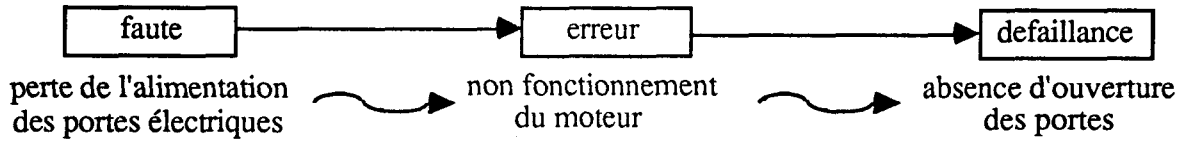
De leur probabilité d'apparition, on en déduira une période de contrôle du système afin de les éviter. Ces contrôles sont appelés maintenance préventive.

- **les fautes transitoires**: elles apparaissent de façon plus ou moins aléatoire. Elles passent par une phase intermittente puis disparaissent. C'est le cas des fautes physiques externes, des fautes d'interaction ou de manipulation, des fautes de maintenance.

- **les fautes permanentes**: elles sont présentes, dès la mise en service du système. Elles demeurent dans le système et par conséquent, sont susceptibles d'en altérer le bon fonctionnement aussi longtemps qu'elles n'en sont pas retirées. Le risque de masquage et de multiplicité de fautes est ici plus élevé puisque leur temps de présence dans le système est plus long.

Une faute seule ne provoque pas forcément une erreur. Il faut parfois accumuler plusieurs fautes pour obtenir une erreur puis une défaillance.

En résumé, on distingue trois niveaux d'imperfection du système:



2-2 Mesure de l'altération de la qualité de service

La défaillance est une mesure possible du dysfonctionnement. Elle affecte le service demandé et est ressentie par l'utilisateur. Elle est caractérisée par le niveau de criticité auquel elle est rattachée. Cette criticité peut se rapporter à la disponibilité ou à la sécurité. Les quatre niveaux critique que l'on distingue sont expliqués dans le chapitre 1.

Les circonstances indésirables, sources d'erreurs dans le système puis de défaillances du service, peuvent être soit évitées soit tolérées suivant l'application du système.

La conception et la réalisation s'appuient sur un ou plusieurs des outils suivants:

- **évitement des fautes** : elle minimise, à la conception et à la construction, la possibilité d'apparition des fautes.

- **prévision des erreurs** : suite à la conception, elle estime, par évaluation du système, la présence d'erreurs et leurs conséquences. Elle contribue à la construction d'un dossier de **maintenance préventive** qui cherche à supprimer les erreurs latentes avant qu'elles ne deviennent effectives.

exemple: une pièce mécanique peut être changée régulièrement avant qu'il n'y ait rupture.

- **suppression des erreurs** : elle minimise la présence d'erreurs par vérification systématique des fonctions. C'est la maintenance corrective curative.

  exemple: vérification du bon fonctionnement des cartes électroniques.

- **tolérance aux fautes** : C'est un traitement d'erreurs, automatique ou assisté par ordinateur. Il s'applique sous forme de :
  - compensation d'erreur. On retrouve l'état correct grâce à la redondance d'information (voteur).
  - On remplace l'état erroné par un état différent sans erreur. Le nouvel état est soit un état permissif (arrêt des machines), soit un état précédent dans lequel on se trouvait avant l'apparition du défaut.

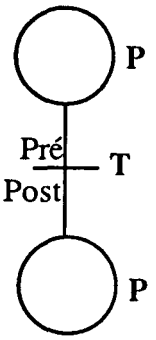
Ces méthodes sont à programmer et à prévoir dès la phase de conception.

ANNEXE 2

DEFINITION DES RESEAUX DE PETRI

1-DEFINITION

Déf.1 : Un réseau de Petri est un quadruplé R = < P, T; Pré, Post > où :



P est un ensemble fini de places (cercles)

T est un ensemble fini de transitions (traits)

Pré est l'application (PxT→ ℕ) d'incidence avant (arc)

Post est l'application (PxT→ ℕ) D'incidence arrière (arc)

Déf.2 : Un réseau marqué est le couple N = <R;M> formé d'un réseau R et d'une application M : P → ℕ appelée Marquage.

M(p) est le marquage de la place p .

Notation matricielle

Transitions / Places	T1 T2 ...
P1	
P2	
.	MARQUAGE
.	Pré/Post
.	

Matrice Pré/Post

Transitions / Places	T1 T2 ...
P1	
P2	
.	MARQUAGE
.	(Post-Pré)
.	

Matrice d'incidence

C(p, t) = Post(p, t) - Pré(p, t)

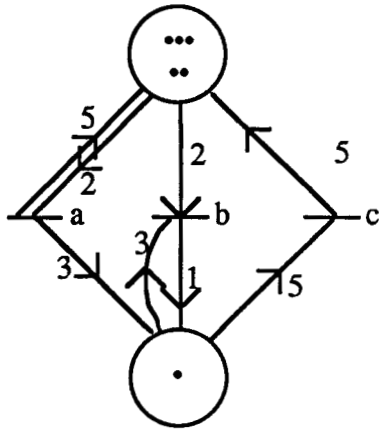
FIGURE A-2-1

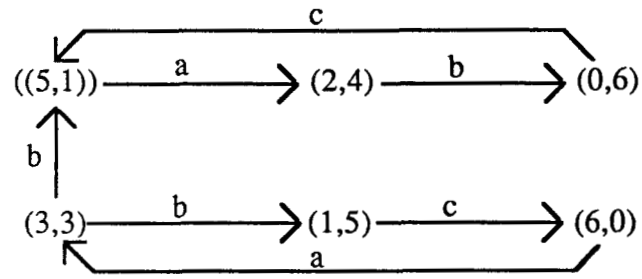
Déf.3 : une transition t est franchissable ou tirable pour un marquage M si et seulement si ∀ p ∈ P, M(p) ≥ Pré(p,t)

Déf.4: l'ensemble des marquages accessibles d'un réseau à m places à partir du marquage M est:

$A(R,M) = \{ M' \in \mathbb{N}^m / \exists$ une séquence telle que $M/s > M' \}$

Exemple :





MATRICES

Pré/Post	a	b	c
P1	5/2	2/0	0/5
P2	0/3	1/3	5/0

Incidence	a	b	c
P1	-3	-2	+5
P2	+3	+2	+5

FIGURE A-2-2

Equation fondamentale :

$$M/s = M' \Rightarrow M' = M + C.\overline{s}$$

Notation : M/s > M'

si s est tirable pour M, on le notera M/s >

2-PROPRIETES GENERALES DES RESEAUX DE PETRI

L'analyse des réseaux de Petri laisse apparaître un grand nombre de propriétés. Ces propriétés sont en général des limites que l'on impose aux réseaux. Elles prouvent que l'automatisme conçu est capable d'être exécuté sans risque de blocage. Celles-ci empêchent l'utilisateur d'écrire des modèles trop compliqués à relire et donc pas facile à valider.

Pour chaque utilisation, on fait un choix des propriétés qui s'appliquent au réseau. Ce choix dépend également du type de validation envisagée.

2-a Réseaux bornés

Une place p d'un réseau marqué <R, M_O> est k-borné si

$\forall M \in A(R,M) \qquad M(p) \leq k$.

Un réseau marqué est borné si $\forall p, \exists k$ tel que p soit k-borné.

Lors de l'implantation d'un réseau sur un outil informatique, on vérifie que celui-là est borné par l'écriture de l'arbre de couverture.

2-b Réseaux vivants

Définition: une transition t d'un réseau marqué <R, M_0> est vivante si pour tout marquage M appartenant à l'ensemble des marquages accessibles de <R, M_0>, t pourra être franchie pour le réseau <R, M>. C'est-à-dire qu'il existe un marquage M' de A<R, M> tel que M'/t >.

Un réseau est vivant si quelle que soit la transition t du réseau t est vivante.

La propriété de vivacité d'un réseau montre qu'il est dans un état permanent de fonctionnement sans toutefois tenir compte des transitions franchies. Dans un réseau vivant pour tout marquage atteint, il est possible d'évoluer vers un autre marquage.Le franchissement à terme de toutes les transitions n'est jamais définitivement impossible.

Cette caractéristique est souvent indispensable pour tous les systèmes en fonctionnement permanent et pour lesquels l'indisponibilité d'une certaine façon signifie une erreur ou une panne. Cette propriété caractérise le non blocage du réseau. Elle met en évidence le fait que l'on ne se retrouvera pas dans une place dont on ne pourra jamais sortir et que le réseau est viable.

2-c Réseaux persistants

Définition: Un réseau est persistant si :

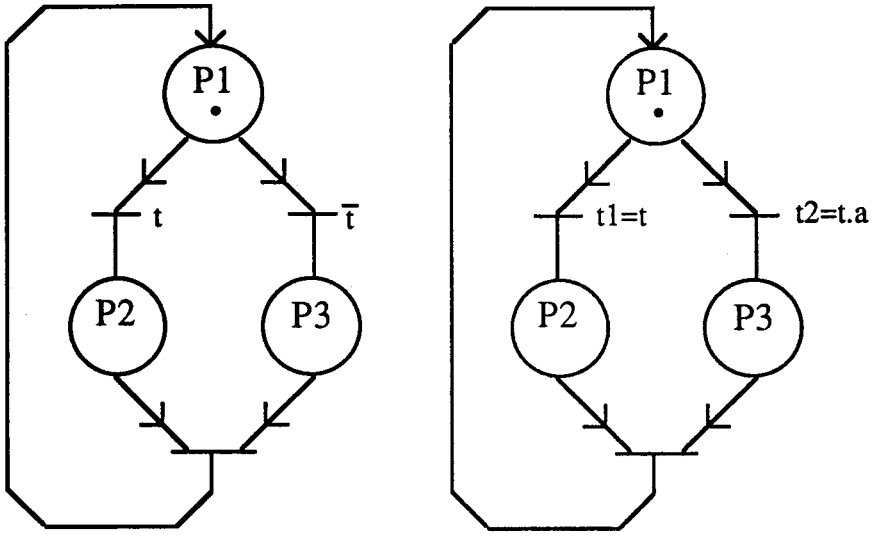
- pour tout marquage accessible M
- pour toutes transitions distinctes t et t'
- si (t est tirable pour M et t' est tirable pour M) alors (tt' est tirable pour M)

Cette propriété signifie que si deux transitions ont une place d'entrée commune, le marquage de la place est suffisant pour franchir les deux transitions, ou alors il faut montrer que les deux transitions ont des conditions incompatibles (elles ne sont jamais validées en même temps).

Dans un réseau persistant toutes les transitions peuvent être traitées sans conflit.

Si la propriété n'est pas vérifiée, il faut définir un ordre de priorité dans le tirage des transitions. Cet ordre se définit assez facilement dans la programmation des réseaux de Petri. Le test des transitions étant réalisé de façon séquentiel, la transition prioritaire sera testée avant l'autre.

Exemple:



Réseau 1 **Réseau 2**

Dans le réseau 1, t et $\bar{t}$ sont incompatibles. Il n'y a donc pas de problèmes de conflit.
Dans le réseau 2, t et t.a sont validées en même temps quand t=a=1. Le réseau n'est pas persistant. On peut dans ce cas donner un ordre de priorité à t2 par exemple (remarque: si l'ordre de priorité est à t1, t2 ne sera jamais franchie). on peut également remplacer la condition de transition t1 par t1=t.$\bar{a}$

FIGURE A-2-3

3- LES TYPES DE RESEAUX

3-a Réseaux à arc inhibiteur

Par les réseaux de Petri, on peut vérifier qu'une condition est validée (test du marquage d'une place pour tirer une transition). Par contre on ne peut pas exprimer directement qu'une transition est soumise à la non vérification d'une condition (test à zéro du marquage d'une place).

Il faut alors utiliser les arcs inhibiteurs. Ceux-ci n'autorisent le franchissement d'une transition que si le marquage de la place est nul.

Définition: Soit un réseau R =<P, T, Pré, Post>
P l'ensemble des places
T l'ensemble des transitions
Pré définit les arcs d'une place vers une transition avec le nombre de marques nécessaires
Post les arcs depuis une transition vers une place

Par rapport aux réseaux classiques où le nombre de marques de Pré appartient à $\mathbb{N}$, dans ce-cas le nombres de marque peut être égal à l'élément vide Ø tel que :

t est franchissable

si M(p) ≥ pré(p, t) quand Pré(p, t) ∈ ℕ

et si M(p) = 0 quand Pré(p, t) = Ø.

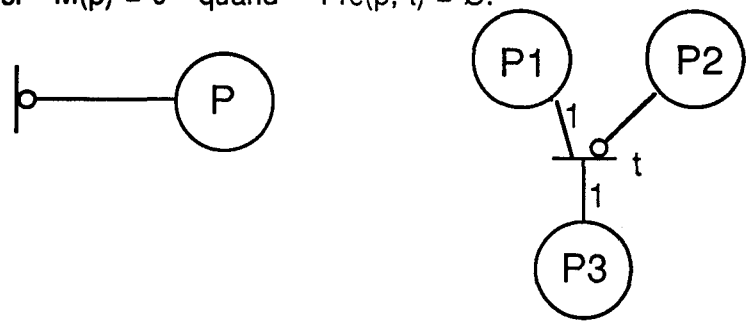


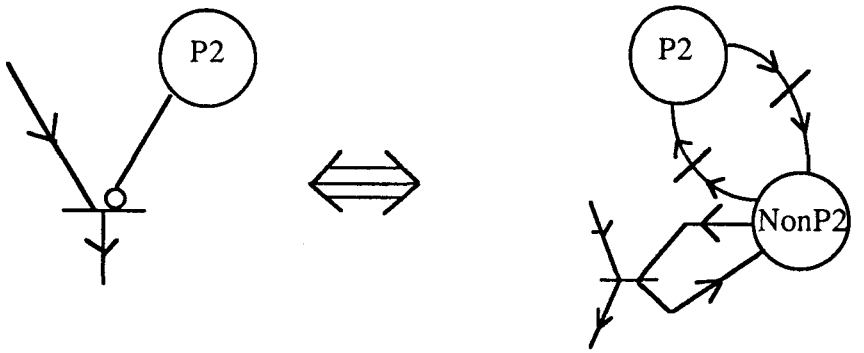
FIGURE A-2-4

L'arc P_2-t est un arc inhibiteur. Pré(P_2, t) = Ø .
t sera franchissable si $M(P_2)=0$ et $M(P_1)\geq 1$.

Les arcs inhibiteurs amènent des inconvénients. Certaines propriétés deviennent indécidables. En effet une transition pouvant être franchie même si le marquage d'une place est nul, la propriété suivante :

si t est franchissable pour un marquage M alors t est franchissable M'≥M

n'est plus valable pour les réseaux à arcs inhibiteurs. Par conséquent, la propriété de vivacité devient difficile à démontrer. Dans ces réseaux on peut remplacer les arcs inhibiteurs par une place supplémentaire représentant la condition inverse de celle qu'on veut tester.



P2 et NonP2 sont incompatibles. M(P2) + M(NonP2) = 1

FIGURE A-2-5

Ce test introduit une boucle supplémentaire entre P2 et Non P2. Le réseau sera alors peu réductible et la recherche des propriétés devra se faire en énumérant des séquences de franchissement et en développant des preuves par assertion.

3-b Réseaux colorées

Lors de la modélisation d'un système réel, il arrive souvent que la taille d'un réseau devienne très importante. Ce phénomène de croissance peut être dû à l'emploi répété de certaines sous-formes de sous-réseaux. Il peut alors être utile d'enrichir la définition d'un réseau pour qu'un tel mécanisme devienne une opération primitive d'un nouveau modèle que nous considérons alors comme abréviation d'un réseau. Sans en changer la puissance algorithmique, ni les propriétés une abréviation de réseau permet une économie d'écriture et de lecture souvent nécessaire à la description de grands systèmes.

Nous allons ici examiner les réseaux dont les places peuvent contenir des marques de différents types ou couleurs .

Le principe est le suivant: le marquage représente le nombre de réalisations d'un seul événement d'un même type. Dans un réseau normal le marquage de la place "Canton occupé" indique uniquement que le canton de ce réseau est occupé. Il existe alors une place canton occupé pour chaque canton de la ligne. Le réseau est alors répétitif. Le même réseau "occupation et libération de canton" est répété autant de fois qu'il y a de canton. En associant à chaque canton une couleur, on ne représentera plus qu'un seul réseau pour la ligne. La couleur de la marque indiquera de quel canton il s'agit. Le marquage d'une seule place peut alors modéliser plusieurs réalisation d'événements de types différents.

A chaque transition, en plus des conditions de franchissement habituelles, on ajoute une condition sur la couleur de la marque qui sera tirée.

Exemple:

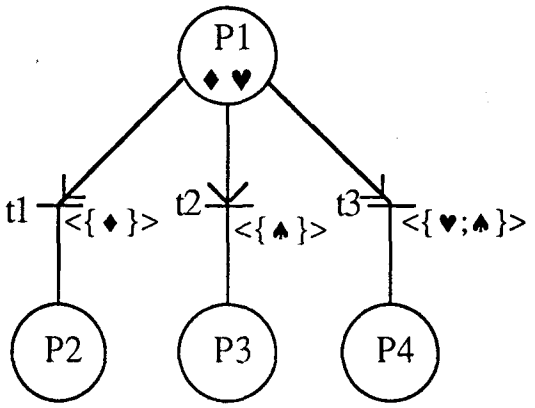


FIGURE A-2-6

<{♦}> représente le domaine de couleur qui autorise le franchissement de la transition t_1.

Dans cet exemple, t_1 et t_2 sont tirables mais pas t_3.

Exemple de l'occupation et de la libération de canton pour trois cantons:

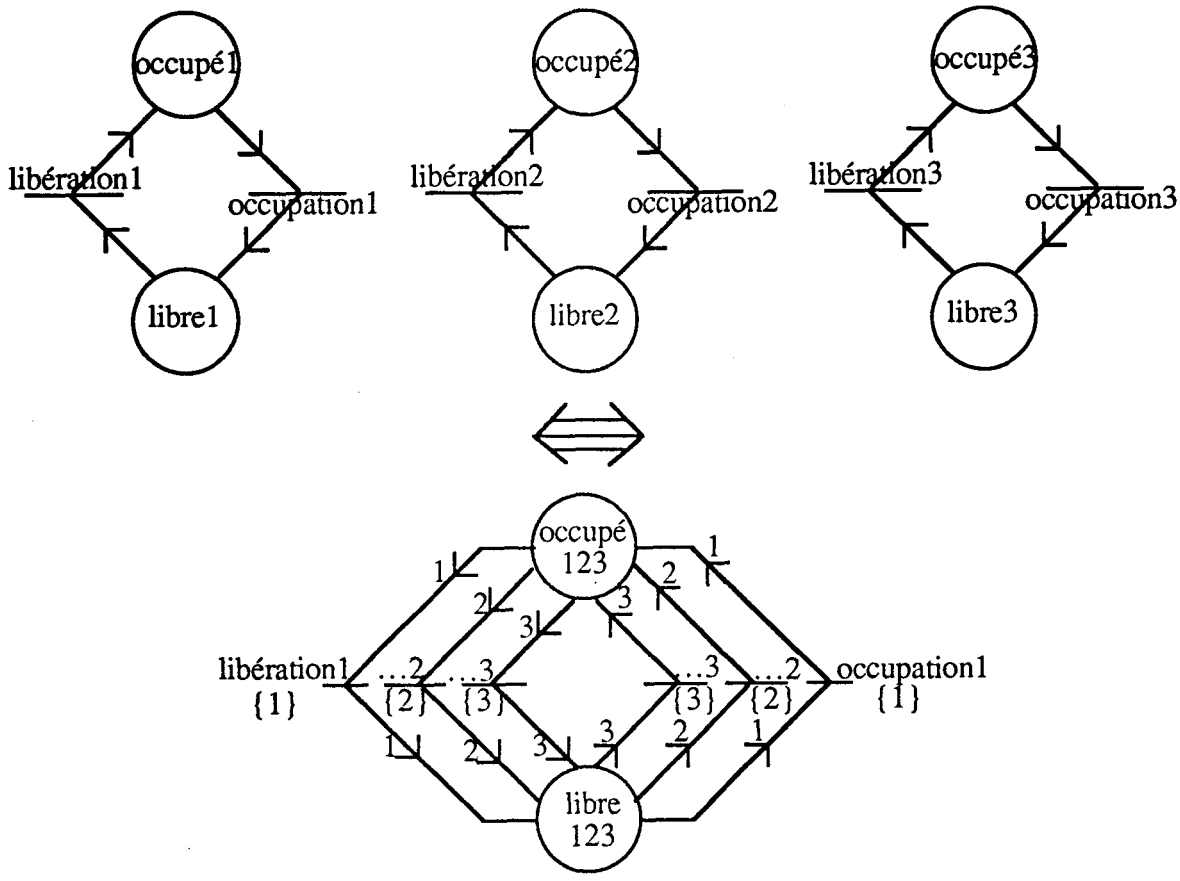


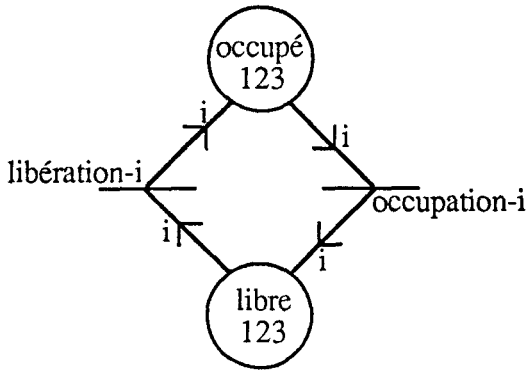
FIGURE A-2-7

Cet exemple montre que le nombre de places est diminué mais le nombre de transitions reste le même. On peut alors utiliser les réseaux à prédicats.

3-c réseaux à prédicats

Avec les réseaux de Petri à prédicats, on va pouvoir attribuer à une marque non plus une couleur comme dans les réseaux de Petri colorés mais plusieurs paramètres. Une marque devient ainsi susceptible de représenter un n-uplet de valeurs. Les arcs seront étiquetés par le type de la marque que l'on désire tirer puis envoyer vers la place destination et les transitions seront conditionnées par certains des paramètres de la marque. Ces conditions s'écrivent sous la forme de prédicats pour lesquels nous pouvons utiliser les variables.

Le réseau "occupation et libération" pour trois cantons devient:



Cette méthode permet de construire des réseaux de Petri réduits lorsqu'une fonction est répétitive.

Les références bibliographiques sur les réseaux de Petri sont les suivantes:
•[BRA83],[THE78],[VAL85] pour la théorie et la pratique;
•[CRE84],[HOL85],[VID86],[TSI85] pour les travaux.

ANNEXE 3

LE GRAFCET

SES PARTICULARITES PAR RAPPORT

AUX RESEAUX DE PETRI

Nous allons dans cette annexe décrire l'utilisation du Grafcet par rapport aux réseaux de Petri.

Pour l'application qui nous intéresse, la principale différence entre ces deux méthodes est la suivante: en Grafcet, le marquage d'une place s'écrit et se lit sous forme d'un booléen. En effet les places ne peuvent contenir qu'une seule marque. Si une deuxième marque est envoyée vers une place, celle-ci ne conserve la trace que d'une seule. Une place n'a donc que deux états: marqué ou non marqué. Le booléen de marquage d'une place apparaît donc dans le prédicat d'une transition. Cette option supprime l'emploi des boucles. Ce sont justement ces boucles que nous avons beaucoup utilisées dans les réseaux de Petri de la logique de canton.

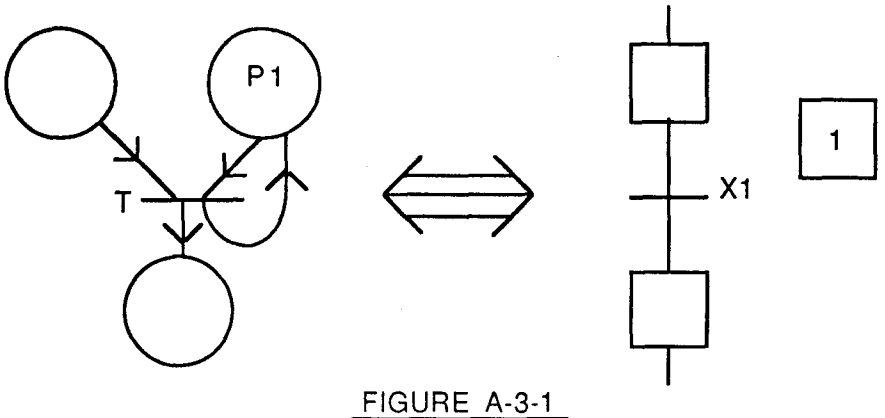


FIGURE A-3-1

Dans le tableau de la page suivante, nous avons réuni les notations utilisées dans les Grafcets. Pour chaque type de représentation, nous donnons l'équivalence en réseaux de Petri.

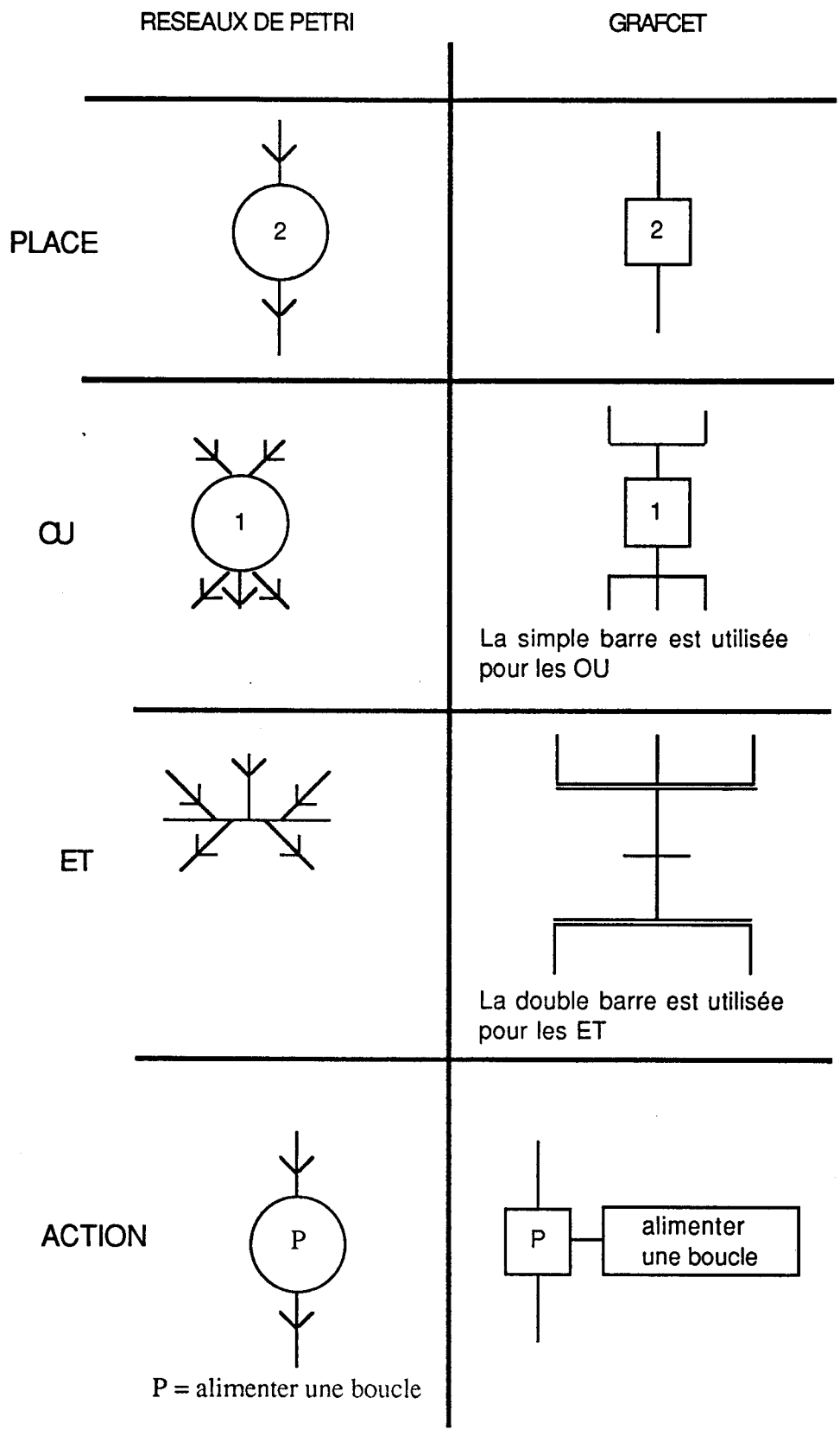
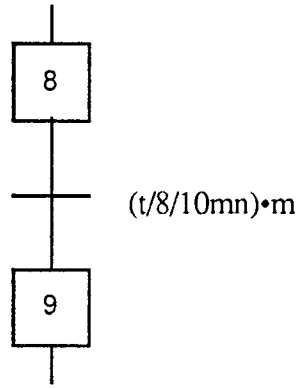

RESEAUX DE PETRI
GRAFCET
PLACE
2
2
OU
1
1
La simple barre est utilisée pour les OU
ET
La double barre est utilisée pour les ET
ACTION
P
P
alimenter une boucle
P = alimenter une boucle

TABLEAU A-3-2

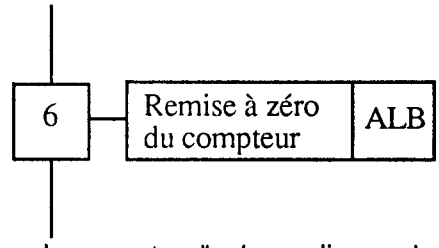
Une action ne se fait qu'une seule fois pendant que l'étape est active (ex: mettre un jeton dans une boîte, ou alors elle est permanente (ex: alimenter un programme de vitesse pour commander le déplacements des rames).

1-TEMPORISATION DES TRANSITIONS



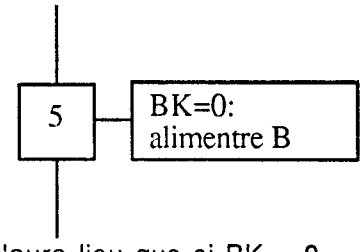
t/8/10 mn signifie que la transition sera franchie au plus tôt 10 mn après la dernière activation de l'étape 8. Cette transition nous assure qu'on reste au moins dix minutes dans la place 8. Elle nous laisse suffisamment de temps pour réaliser l'action correspondant à cette place.

2-ACTION IMPULSIONNELLE



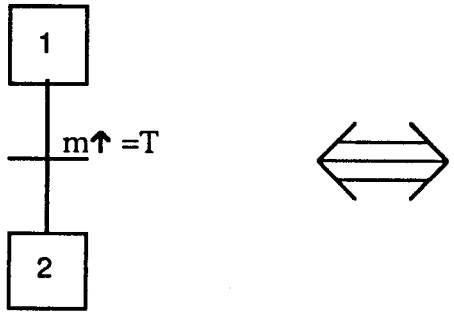
L'action "remise à zéro du compteur" n'aura lieu qu'une seule fois à chaque activation de la place 6.

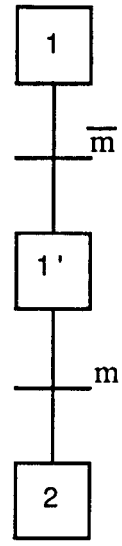
3-ACTION CONDITIONNELLE



L'action "alimenter B" n'aura lieu que si BK = 0.

4-TRANSITION IMPULSIONNELLE





T est activée quand m passe de 0 à 1
T' = m↓ est activée quand m passe de 1 à 0

1' est une place fictive

Ces quelques définitions que nous avons réunies ici seront suffisantes pour comprendre notre étude sur le GRAFCET.

GLOSSAIRE

Dans ce glossaire, sont expliqués les termes techniques et spécifiques à notre exemple. On fait référence au paragraphe de la thèse où ils sont employés ou expliqués plus en détail.

CANTON: (Chapitre 2, § 3-1) Portion de voie sur laquelle une seule rame est autorisée à pénétrer.

TRONÇON: (Chapitre 2, § 3-1) Ensemble de plusieurs cantons. Le tronçon est une zone gérée par une unité automatisée ou Pilote Automatique (PA).
Dans un tronçon, on a un canton d'entrée, un canton de sortie et un ou plusieurs cantons intermédiaires ou cantons courants.

ZONE DE REBROUSSEMENT: (Chapitre 2, § 3-1) Portion de voie comprenant une communication. La communication, composée de deux aiguillages, permet à la rame de passer d'une voie sur l'autre et de repartir dans l'autre sens. C'est le rebroussement.

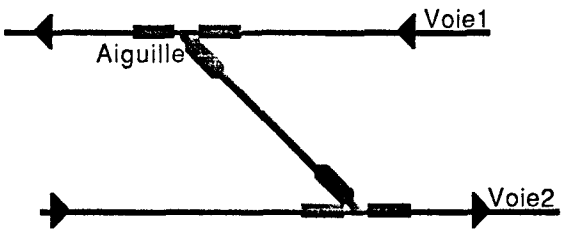


FIGURE G-1

SERVICE NORMAL: (Chapitre 2, § 3-1) Mode d'exploitation ordinaire sur la voie 1 et sur la voie 2 sans emprunter la communication.

SERVICE AVEC REBROUSSEMENT (Rbt): (Chapitre 2, § 3-1) Mode d'exploitation où la rame emprunte la communication pour passer de la voie 1 à la voie 2 et repart dans l'autre sens.

EQUIPEMENTS D'AUTOMATISMES

1-Détecteurs (Chapitre 2, § 3-2)

EAC-EMETTEURS ANTI-COLLISION: Antennes émettrices posées sur le véhicule. Ces antennes émettent en permanence un signal qui permet à la rame de se faire reconnaître des automatismes placés au sol.

3-Implantation des boucles

L'implantation des boucles dépend de la fonction à réaliser.

Pour l'exemple de la logique de canton, l'implantation des boucles est la suivante:

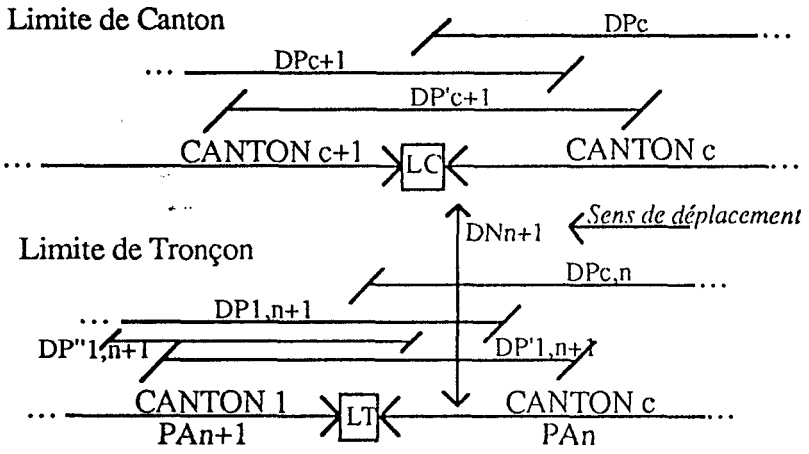


FIGURE G-3

Pour l'exemple de la zone de rebroussement, on a implanté les boucles FS pour commander les déplacements des trains (FDT: Fond de Tiroir):

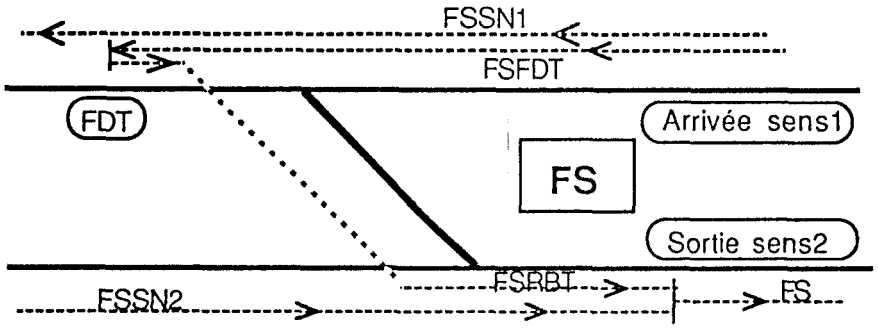


FIGURE G-4

et les boucles DP et les barrières DN pour détecter la position des trains:

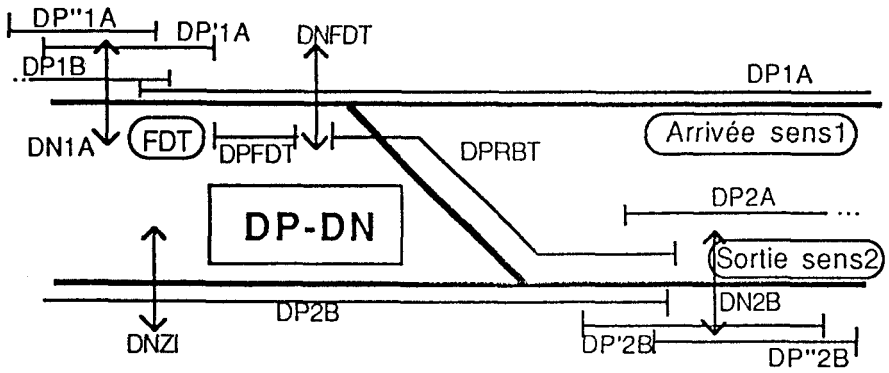


FIGURE G-5

BIBLIOGRAPHIE

SURETE DE FONCTIONNEMENT

[AMP80] *"Quelques réflexions sur les études de sécurité des systèmes de transport en commun urbain"*
André Ampelas,
Ingénieur chef de division à la direction du réseau routier.

[BCE74] *"La sécurité dans les transports collectifs:*
- analyse des accidents dans les modes actuels, Mars 74;
- Réglementation, Janvier 74";
Organisme B.C.E.O.M.

[DHA86] *"Sur la réglementation de la sécurité des transports nouveaux par objectifs quantifiés"*
Henri-Bertrand Thibault - Jean-François Dhalluin.

[FAU78] *"Fiabilité et renouvellement des équipements"*
R. Faure - J.L. Laurière .
Collection programmation GAUTHIER-VILLARS.

[GAB74] *"Réflexion sur l'applicabilité aux modes de transports nouveaux de l'étude de la sécurité dans les modes actuels du B.C.E.O.M."*
R. Gabillard,
Mai 1974.

[GAB79] *"Sur les rôles respectifs de l'homme et du matériel dans la sécurité d'un système de transport"*
Robert Gabillard - Michel Ficheur,
Annales des Ponts et Chaussées - 2ème Trimestre 1979.

[HEC79] *"Fault tolerance software"*
Herbert Hecht,
IEEE on reliability- R28 -nº3, Août 1979.

[LAP85] *"Sûreté de fonctionnement des systèmes informatiques et tolérants aux fautes: concept et base".*
J.C. Laprie,
TSI, Vol 4, nº5, 1985.

[MAR85] *"Méthodologie d'étude de la sécurité"*
Claude Marcovici,
Note Sécurité - Matra Transport, 1985.

[PER87] *"L'automatisation intégrale d'un métro existant à grand gabarit- Métro de Lyon - MAGGALY"*
J. Pernot - C. Teillon,
P.R.D.T.T.T. 1987.

[SCH69] *"Traité de fiabilité"*
Maurice Schowb - Guy Pérache,
ed. Masson & C_{ie}, 1969.

[VIL88] *"Sûreté de fonctionnement des systèmes industriels. Fiabilité. Facteurs humains. Informatisation."*
Alain Villemeur,
ed. Eyrolles, 1988.

SPECIFICATION

[ALA85] *"Point sur l'utilisation des Réseaux de Petri et du Grafcet à la RNUR"*
Pierre Alanche,
TSI vol nº4-1985.

[BAU86] *"Génie logiciel: Contribution à la définition et à la mise en œuvre d'un poste d'automatisation et d'instrumentation"*
Christine Baudel,
Thèse de 3ème cycle - Université de Lille-Flandres-Artois, Avril 86.

[BLA82] *"Comprendre, Maîtriser et Appliquer le Grafcet"*
M. Blanchard,
ed. Cepadeus 1982.

[BRA83] *"Réseaux de Petri: théorie et pratique" tome 1 et 2*
G.W. Brams,
ed. Masson 1983.

[BRA88] *"Spécification du logiciel dans l'industrie"*
G.Bracon,
Bigre - Janvier 88 - Spécification du logiciel.

[DEC88] *"«Software Throught Pictures» de IDE. Outils et méthodes"*
Olivier Declerfayt,
Université Catholique de Louvain,
Bigre nº58, janvier 1988.

[FEK88] *"Outils de spécification de systèmes numériques. Analyse Bibliographique"*
Béatrice Fayolle - Miloudi El Koursi
Octobre 1988-137.

[GAU88] *"Impact de spécifications formelles sur le développement des logiciels"*
Marie Claude Gaudel et Christine Choppy,
Bigre nº58, janvier 88.

[HOL85] *"Réseaux de Petri dans les systèmes de production"*
Hollinger ,
TSI vol4,6, Nov-Dec 85.

[JFD86] *"Spécification d'un projet: une aide automatique"*
Jean-François Dhalluin, Bureaux d'études Automatismes n°28, Octobre 1986.

[KWA87] *"Recherche en automatisme: les travaux français"*
Nadine Kwasny,
Bureaux d'études Automatismes nº32,
Mars 87.

[MOA76] *"L'approche fonctionnelle dans la vérification des systèmes informatiques. Proposition d'un ensemble de méthodologies"*
Mohamed Moalla,
Thèse de Docteur Ingénieur, Institut National polytechnique de Grenoble, Décembre 1986.

[MOA81] *"Spécification et conception sûre d'automatismes discrets complexes, basées sur l'utilisation du Grafcet et des Réseaux de Petri."*
Mohamed Moalla,
Thèse d'état, juillet 1981.

[PAR86] *"GAM T17 (Projet indice 4)*
Méthodologie de développement des logiciels intégrés"
CELAR, Centre d'Electronique de l'ARmement
IPA Parayre, 1986.

[REC86] *"La prévision du risque technologique"*
La Recherche nº183, Décembre 1986.

[THE78] *"Pratique séquentielle et Réseaux de Petri"*
Sylvain Thélliez,
ed . Eyrolles, 1978.

[TOU82] *"Grafcet et logique industrielle"*
Jean-Marc Toulotte - Sylvain Thélliez, ed . Eyrolles, 1982.

[TROY86] *"Projet de guide d'évaluation de la fiabilité des logiciels"*
R. Troy,
Journée AFCIQ " Fiabilité des logiciels: évolution internationale de la normalisation", 19 Février 1986 Toulouse.

[TSI85] *"Technique et Sciences informatiques: nº Spécial Réseaux de Petri"*
AFCET Informatique - Edition Dunod
Volume 4 nº1, janvier-février 1985.

[VAL85] *"SEDRIC: un simulateur à événements discrets basé sur les réseaux de Petri"*
R. Valette - V. Thomas - S. Bachmann.
APII-RAIRO, vol19, nº5,1985.

[VID86] *"Les réseaux de Petri et leurs applications"*
G. Vidal - Naquet
interfaces, nº43, Mai 86.

[AGI89] *"Journée d'étude*
Conception et Validation des logiciels de sécurité dans les transports terrestres"
AFCET-GRRT-INRETS-CRESTA-USTLFA-8 juin 89.

VALIDATION

[BEO88] *"Une méthode d'analyse de la sûreté des logiciels"*
Beounes, LAAS - Cheilan, Aerospatiale - Hourtolle, Aerospatiale,
6ème colloque internationale de fiabilité et de maintenabilité- λμ
Strasbourg - France , 1988.

[CHA89] *"Projet SACEM*
Application de méthodes formelles à la validation ds logiciels."
Journée AFCET, GRRT, INRETS-CRESTA, USTLFA du 8 juin 89.

[CRIS85] *"Exception, défaillance et erreur"*
F. Christian,
TSI vol4-3, mai, juin 1985.

[LAM85] *"Utilisation des Réseaux de Petri pour le test des programmes temps réels"*
Guy LAMARCHE, Patrick TAILLIBERT.
TSI, vol nº1-1985.

[MYE79] *"The art of the software testing"*
Glennford J. Myers
edition Wiley-Interscience, 1979.

[RAN75] *"System structure of software fault tolerance"*
Brian Randell,
IEEE on software engeneering, vol SE-1, nº2, juin 1975.

[TOU83] *"Applications industrielles du GRAFCET"*
Jean-Marc Toulotte - Sylvain Thelliez, ed . Eyrolles, 1983.

[TROY84] *"Guide d'acquisition des logiciels: aspects exigences qualité."*
J. Meekel, R. Troy,
Agence de l'informatique, Février 1984.

TABLE DES MATIERES

CHAPITRE 4

RESUME: Le cadre d'étude de cette thèse est la sûreté de fonctionnement dans les transports terrestres automatisés. Le développement d'automatismes de plus en plus complexes nécessite de nouvelles méthodes prouvant la sûreté de fonctionnement aussi bien du point de vue sécurité que disponibilité et "maintenabilité". Le travail de recherche s'est orienté vers les méthodes de spécification des automatismes et la validation de ces spécifications.

Dans un premier temps, on étudie différents outils de spécification qui s'appuient sur des méthodes formelles et semi-formelles de définition du système. On a analysé leur apport au niveau des études de sécurité.

Avant de passer à la phase de conception, il est important de s'assurer que le modèle conçu est conforme aux objectifs. L'analyse préliminaire définit les objectifs à atteindre et les critères à respecter. Elle sert de document de référence pour valider la spécification.

Par la suite, nous étudions la simulation du modèle. Nous décrivons dans ce document l'outil qui a été développé pour simuler les spécifications d'automatismes au sol. Il aide la mise au point de la spécification et sa validation. Il simule le comportement de l'équipement dans un environnement proche de la réalité.

Mots clés: simulation; modèle spécification; transport en site propre; transport; automatisation; sécurité; fiabilité (sûreté de fonctionnement); validation test.

ABSTRACT: This thesis is dealing with dependability in terrestrial transportation. New methods are needed to prove the dependability of more and more complex automatisms working for safety as well as for security, availability and reliability.

We get the research in automatic working specification methods. By these methods, the specification model will be clear and unambiguous.

On one hand, different specification tools are studied. These tools lean on formal and semi-formal system definition methods. Their contribution on safety studies concerning the automatised terrestrial transportation have been analysed.

Before the conception phase, we must insure that the model reflects ours objectives. We must avoid rectifying the specification during the following phases. The objectives and criterions are defined by the preliminary analysis. It is our reference to validate the specification.

On the other hand, we have studied the model simulation. The tool that have been developped to simulate ground automatic working specification is described in this thesis. it helps to perfect and to validate specification and simulates the equipement behaviour in an almost real environment.

Key words: Simulation; Specification model; Corridor transport; Transport; Automation; Safety; Reliability; Test validation.