N° d'ordre : 997

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE
en
PRODUCTIQUE, AUTOMATIQUE ET INFORMATIQUE
INDUSTRIELLE

par

Jean-Christophe RIAT
Ingénieur de l'Ecole Centrale de Lille

VALIDATION PAR SIMULATION D'ARCHITECTURE DE COMMANDE D'ATELIER DANS L'INDUSTRIE DE PRODUCTION MANUFACTURIERE

Application aux systèmes automatisés de production d'un grand constructeur automobile

Soutenue le 12 Novembre 1992 devant la Commission d'Examen

Membres du jury :	M. MEIZEL	Rapporteur
	M. VALETTE	Rapporteur
	M. BOUREY	Examinateur
	M. GENTINA	Examinateur, Directeur de thèse
	M. STAROSWIECKI	Examinateur
	M. GIRARD	Invité
	M. DUMOUTIER	Invité
	M. PIERRE	Invité

A mes parents

A ma soeur

A mes amis

AVANT PROPOS

Le travail de recherche présenté dans ce mémoire a été réalisé au sein de la Direction Informatique, Télécommunications et Automatismes (D.I.T.A.) du groupe automobile PSA Peugeot Citroën, en collaboration avec le Laboratoire d'Automatique et d'Informatique Industrielle de l'Ecole Centrale de Lille (L.A.I.L.), dans le cadre d'une Convention Industrielle de Formation par la REcherche (C.I.F.R.E.).

A son terme, je tiens à remercier Monsieur Jean-Claude GENTINA, professeur à l'Ecole Centrale de Lille, pour avoir accepté d'assurer la direction de ces travaux et pour le soutien, les conseils et les critiques dont il m'a fait bénéficier tout au long de ces trois années de thèse. Je voudrais également exprimer toute ma reconnaissance à Monsieur Pascal DUMOUTIER, directeur de la division Stratégie et Méthodologies d'Automatismes du groupe P.S.A. Peugeot Citroën pour les conseils qu'il m'a fournis et pour la mise à disposition des moyens informatiques qui ont permis la réalisation de ces travaux.

Je tiens également à remercier

Monsieur MEIZEL, professeur à l'Université de Technologie de Compiègne,
Monsieur VALETTE, directeur de recherche au LAAS-CNRS

d'avoir accepté d'être les rapporteurs de cette thèse

ainsi que

Monsieur BOUREY, maître de conférences à l'Ecole Centrale de Lille,
Monsieur STAROSWIECKI, professeur à l'Université des Sciences et Techniques de Lille

pour l'honneur qu'ils me font en participant à mon Jury de Thèse.

Je suis en outre très honoré par la présence à ce Jury de Monsieur GIRARD, directeur de la Stratégie et des Technologies Avancées du groupe P.S.A, de Monsieur DUMOUTIER déjà cité, ainsi que de celle de Monsieur PIERRE Responsable Marketing du Centre International de Support Applications de la société Télémécanique.

Je voudrais ici remercier tous les membres du département d'Automatique et d'Informatique Industrielle, et particulièrement Monsieur ABRAMATIC, parmi lesquels j'ai été intégré pendant ces années de thèse, ainsi que l'ensemble des chercheurs du Laboratoire d'Automatique et d'Informatique Industrielle de l'Ecole Centrale de Lille.

Mes remerciements vont encore à toutes celles et ceux qui, à titre divers, ont participé à l'élaboration et à la réalisation de ce mémoire, et plus particulièrement à l'équipe de Monsieur Delignières grâce à qui j'ai pu appliquer sur un exemple industriel la démarche développée dans ce mémoire.

Pour illustrer les travaux présentés dans ce mémoire, nous avons été amenés à utiliser des spécifications, fournies par la Télémécanque, sur le fonctionnement de ses matériels d'automatisme. Certaines de ces spécifications sont détaillées dans ce mémoire, avec l'accord de cette société.

Ces spécifications ne doivent pas faire l'objet d'une reproduction, sous quelque forme que ce soit, sans l'accord écrit préalable de la Télémécanique.

Télémécanique se réserve le droit de faire évoluer ces spécifications sans information préalable. Télémécanique ne pourra, en aucun cas, être tenue responsable des erreurs pouvant être contenues dans ce document, ni de l'usage qui pourrait en être fait.

Les appellations TELWAY 7, UNI-TELWAY, UNI-TE et TSX 7 qui figurent dans ce mémoire sont des marques déposées Télémécanique. Les protocoles décrits dans les spécifications sont la propriété de Télémécanique. Il ne pourra y être fait référence sans l'appartenance à Télémécanique.

SOMMAIRE

Validation par simulation d'architecture de commande d'atelier dans l'industrie de production manufacturière

Application aux systèmes automatisés de production d'un grand constructeur automobile

Chapitre I

La commande et le pilotage des systèmes automatisés dans l'industrie manufacturière

Chapitre II

Modélisation et simulation d'architecture de commande et de pilotage d'atelier

Chapitre III

Application à un exemple industriel du groupe PSA Peugeot Citroën

ANNEXES

Annexe 1 : Formalisme des réseaux de Petri et description de l'outil SEDRIC

Annexe 2 : Etude de l'architecture de commande de l'atelier des mélanges de Rennes / La-Barre-Thomas

INTRODUCTION GENERALE

C'est en 1890 qu'Armand Peugeot (1849 - 1915) construit, de façon artisanale, le premier véhicule qui portera son nom en adaptant un moteur à explosion élaboré par l'allemand Gottlieb Daimler sur un quadricycle Peugeot [LOU 90]. Sept ans plus tard, il crée la Société Anonyme des Automobiles Peugeot dont l'atelier d'Audincourt produira 54 véhicules avec 125 ouvriers en 1897 ! En 1990, 1274 véhicules sont sortis chaque jour du centre de production Peugeot de Mulhouse avec un effectif de 12 509 personnes...

André Citroën (1878 - 1935) vint à l'automobile presque par hasard quand son ami André Haarbleicher lui demanda en 1907 de prendre la direction de la société Mors Automobiles qui connaît de graves difficultés financières. La production annuelle qui était de 120 unités en 1907 passa à 2810 véhicules en 1919 alors que l'entreprise comptait environ 4500 employés [WOL 91]. La production journalière, en 1990, du site Citroën de Rennes/La-Janais, a été de 1340 véhicules avec un effectif de 14 254 personnes...

Ces quelques chiffres permettent de mesurer l'ampleur des gains de productivité réalisés en moins d'un siècle dans le secteur automobile ! Cette industrie apparaît comme une "pionnière" qui a ouvert la voie à de nouveaux modes de fabrication et d'organisation du travail [SAU 84] pour la production manufacturière en grande série de produits de consommation courante.

Après quelques décennies de fabrication artisanale, l'industrie automobile s'est orientée, après la première guerre mondiale, vers un mode de production faisant appel à la division des tâches et au travail à la chaîne. Les pionniers en la matière ont notamment été Henry Ford et André Citroën. C'est grâce aux progrès accomplis dans le domaine des machines outils qu'une telle organisation a été rendue possible. Grâce à l'accroissement de la précision des opérations d'usinage, les pièces sont devenues interchangeables et ont pu prendre place dans une structure de véhicule standardisé. Ce type de travail a permis de diminuer dans des proportions importantes le prix de revient d'une automobile, ce qui a ouvert l'industrie du véhicule à un marché considérable.

Le mot automation fut utilisé pour la première fois en 1940 à la Ford Motor Company et a été mis en oeuvre sur un système de manutention de pièces mécaniques [BON 87]. Après la seconde guerre mondiale, les constructeurs généralisent les machines spéciales de plus en plus complexes et commencent à installer les premières machines

automatiques. C'est à cette période qu'apparaissent les premières machines transfert, qui fonctionnent selon les principes électro-mécaniques. Les opérations sont réalisées en séquence sans intervention manuelle.

A partir des années 1960-70, l'application de la micro-électronique aux outils de production permet de mettre au point les premières machines outils à commande numérique (MOCN). Celles-ci permettent des fabrications de pièces différentes lancées en lots répétitifs par simple changement de programme, ouvrant ainsi la voie à la production flexible [ROC 89]. Les robots industriels ont profité de cet acquis et se sont progressivement développés dans l'industrie [KOR 85].

Dans les années 70, l'utilisation de calculateurs à usage industriel et d'automates programmables se généralise sur les sites de production [BON 87]. Ces matériels, conçus pour fonctionner dans les ateliers, sont adaptés au traitement des signaux issus de capteurs et sont capables de produire des actions de commande. Ils ont contribué au développement important des automatismes grâce à la souplesse de la logique programmée, qui tend à remplacer la logique câblée.

Jusque dans les années 80, les automatismes étaient principalement affectés à la commande de machines isolées. Dans le but d'améliorer la productivité et la qualité, le nombre et la complexité des tâches automatisées s'accroissent. Afin d'assurer une meilleure maîtrise de leur outil de production, les industriels souhaitent disposer de fonctions d'assistance pour le suivi, la conduite et la gestion de leurs installations. Les automatismes sont donc progressivement intégrés dans un ensemble plus général : le système automatisé de production (S.A.P.) [LES 91].

Le concept C.I.M. (Computer Integrated Manufacturing) [WAL 90] apparaît de nos jours comme une référence pour la conception des moyens de production. Il conduit, pour la commande et le pilotage des systèmes automatisés, à la réalisation de systèmes hiérarchisés et distribués où les différentes fonctions de contrôle/commande sont réparties sur des équipements distants interconnectés par des réseaux locaux industriels.

Le groupe PSA Peugeot Citroën a engagé un plan de recherche pour des méthodes assistées par des outils pour la conception des architectures de commande de ses ateliers. Cette investigation concerne les domaines du matériel et du logiciel [BOR 90]. Si de nombreux travaux traitent de la production du code à implanter sur les équipements de commande, peu d'études existent pour assister la conception de l'architecture matérielle. Il s'agit pourtant d'un aspect important des installations automatisées, qui, avec la forte croissance du nombre de réseaux locaux industriels installés, soulève des difficultés importantes et nouvelles de conception.

Dans le cadre de ce projet, nous nous sommes intéressés à la validation a priori des performances de ces architectures. Ce point est en effet essentiel pour toute application contrainte par le temps, donc en l'occurrence la commande et le pilotage du procédé de fabrication. Il est pourtant encore mal maîtrisé par les concepteurs, et c'est souvent lors du démarrage des installations sur site que l'on découvre les performances effectives du système mis en place.

Les travaux présentés dans ce mémoire propose une démarche de modélisation/simulation d'architecture de commande avec un formalisme de réseaux de Petri. Après avoir identifié les divers éléments à prendre en compte dans le système à étudier, nous proposons une approche pour modéliser chaque composant avec le formalisme adopté. Les idées sont appuyées par la modélisation et la simulation effectives d'architectures de commande du groupe PSA Peugeot Citroën en utilisant l'outil SEDRIC.

Ce mémoire comporte trois chapitres :

- Dans une première partie intitulée "LA COMMANDE ET LE PILOTAGE DES SYSTEMES AUTOMATISES DE PRODUCTION DANS L'INDUSTRIE MANUFACTURIERE", nous présentons les éléments qui caractérisent les architectures de commande installées dans les ateliers de production du groupe automobile PSA Peugeot Citroën. Après un tour d'horizon des méthodes et outils actuellement disponibles pour assister la conception de tels systèmes, nous proposons une nouvelle démarche, qui inclue une étape de modélisation/ simulation, pour élaborer les architectures de commande d'atelier manufacturier.

- Le chapitre suivant dénommé "MODELISATION ET SIMULATION D'ARCHITECTURE DE COMMANDE ET DE PILOTAGE D'ATELIER" présente une façon de modéliser, avec un formalisme réseaux de Petri à structures de données, les différents équipements qui composent une architecture de commande. Cette approche est appliquée à différents équipements d'automatisme de la société Télémécanique.

- Dans une troisième et dernière partie nous appliquons la démarche de simulation à un cas industriel du groupe PSA Peugeot Citroën : la conception du système de suivi de fabrication de l'atelier de montage ANDON de Citroën à Rennes/La-Janais. Cet exemple, riche d'enseignements, démontre la faisabilité et l'intérêt de l'approche. Ce chapitre est intitulé "APPLICATION A UN EXEMPLE INDUSTRIEL DU GROUPE PSA PEUGEOT CITROEN".

CHAPITRE I

La commande et le pilotage des systèmes automatisés dans l'industrie manufacturière

INTRODUCTION

La compétition économique actuelle entraîne l'automatisation des ateliers de production pour augmenter la productivité des usines et améliorer la qualité des produits. L'ampleur des installations automatisées devient telle qu'il apparaît indispensable de disposer d'outils et de méthodes pour en assister la conception. Ce constat est en particulier valable pour les architectures de commande et de pilotage d'ateliers automatisés. C'est sur ce thème que portent les travaux qui sont présentés dans ce mémoire.

Le premier chapitre a pour objet de présenter le domaine d'application étudié, à travers la vision que nous en avons après trois années de thèse passées à la Direction Informatique Télécommunications et Automatismes (DITA) du groupe automobile PSA Peugeot Citroën. Ce chapitre est constitué de trois parties :

- Dans une première partie, nous dressons un bilan des installations automatisées implantées chez un manufacturier automobile tel que PSA Peugeot Citroën. Il nous paraît en effet important de mesurer l'ampleur que prennent actuellement les systèmes automatisés et les architectures de contrôle/commande associées, pour bien cerner les problèmes liés à leur conception.

- Les moyens actuellement utilisés pour élaborer et mettre en place les architectures de commande et de pilotage sont traités dans un second volet. Nous constatons que les outils actuellement disponibles sur le marché, ainsi que les travaux de recherche en cours sont principalement axés autour de la génération du code des programmes de contrôle/commande. Il est paradoxal de constater que, si l'aspect performances est un critère important dans le choix des architectures, très peu de chose existe actuellement pour assister les concepteurs sur ce point.

- C'est pourquoi nous proposons dans une troisième partie, une démarche de conception d'architecture de commande et de pilotage, qui intègre une évaluation par simulation : l'objectif est de modéliser l'architecture matérielle choisie avec les données échangées pour en évaluer les performances. Cette approche doit permettre au concepteur, de valider dès la phase de conception, les performances du système qu'il propose.

I. AUTOMATISATION DE LA PRODUCTION

I.1. Productivité... Productivité...

I.1.1. Quelques chiffres concernant la productivité

I.1.1.a Position du groupe PSA Peugeot Citroën au niveau mondial

En 1990, la production totale du groupe PSA Peugeot Citroën a été de plus de 2,2 millions de véhicules. Ce résultat positionne le constructeur généraliste français à la septième place des groupes automobiles mondiaux.

Rang mondial	Groupe automobile	Production totale
1	General Motors	7.425.000
2	Ford	5.541.000
3	Toyota	5.520.000
4	Nissan	3.185.000
5	Volkswagen-Audi-SEAT	3.053.000
6	Fiat	2.634.000
7	PSA Peugeot Citroën	2.235.000
8	Honda	2.000.000
9	Chrysler	1.899.000
10	Renault	1.843.000

Tableau I.1 : *classement des 10 premiers groupes automobiles mondiaux*

Dans une conjoncture internationale difficile et une compétitivité entre constructeurs de plus en plus forte, notamment en raison de l'efficacité de l'industrie automobile japonaise, le groupe PSA Peugeot Citroën doit maintenir un effort soutenu pour améliorer sa compétitivité et la qualité de ses produits [BEL 86]. En effet, selon un bilan sur l'industrie automobile mondiale réalisé par le MIT (Massachussetts Institute of Technology) [WOM 90], les constructeurs nippons auraient une avance significative dans tous les domaines : productivité, qualité et organisation.

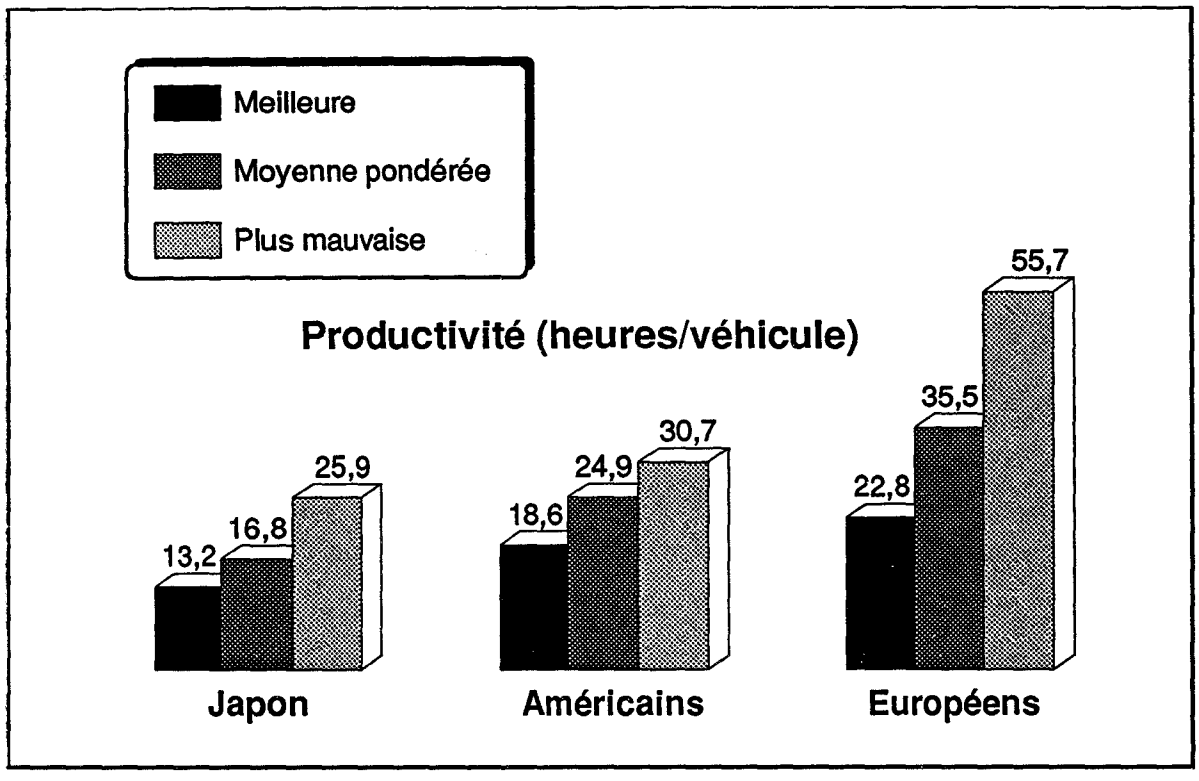


Figure I.1 : *productivité des chaînes d'assemblage des grands constructeurs en 1989*

Ainsi, chez les grands constructeurs, la productivité des usines Japonaises est en moyenne de 16,8 heures par véhicule, contre 20,9 heures pour les transplants japonais aux Etats-Unis, 24,9 heures pour les constructeurs américains et 35,5 heures pour les européens. Ces chiffres bruts sont cependant à interpréter avec prudence. Notamment les japonais achètent beaucoup plus de composants aux sous-traitants, ce qui fausse ainsi les comparaisons[1].

II.1.1.b Productivité pour le groupe PSA Peugeot Citroën

A partir des années 80, l'automatisation a été la solution adoptée en Europe, dans l'objectif de réaliser des gains de productivité et de qualité importants. Si, comme l'illustre le graphique de la figure I.2, des progrès significatifs ont été réalisés ces dernières années au sein du groupe PSA Peugeot Citroën, il est essentiel pour préparer l'avenir de maintenir un effort permanent pour l'amélioration des performances de l'outil industriel en terme de flexibilité, de fiabilité et de qualité. L'objectif fixé par la direction générale du groupe est d'augmenter la productivité de 50 % sur une période de 5 ans.

[1] d'après l'Usine Nouvelle, juillet 1991

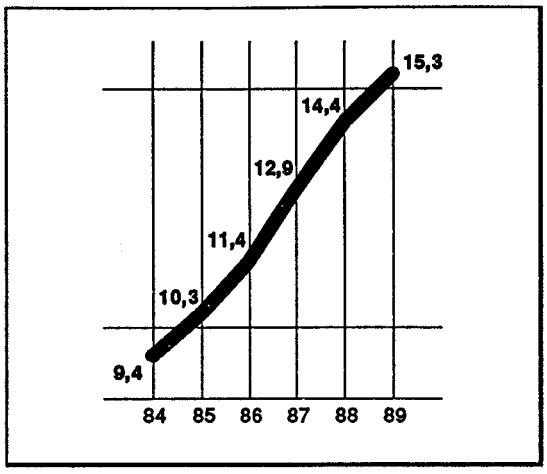


Figure I.2 : *nombre de voitures produites par employé chez PSA[2] (en équivalent 205)*

La figure I.3 juxtapose sur un même graphique, les courbes d'évolution de la production et des effectifs, pour le centre de production de Sochaux. Les tendances des courbes montrent clairement que jusqu'en 1980, l'automobile était principalement une industrie de main-d'oeuvre : l'évolution du volume de production était directement liée à celui des effectifs. Par contre, à partir du moment où les installations automatisées sont apparues, le nombre d'employés et celui des véhicules fabriqués ont évolués de façon inverse !

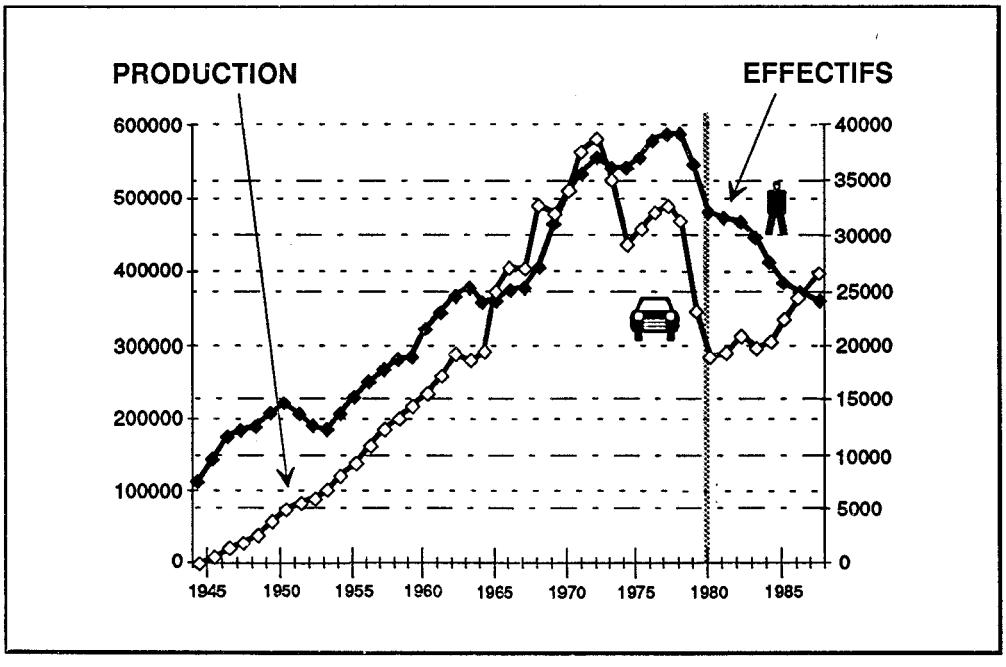


Figure I.3 : *courbes production/effectif du centre Peugeot de Sochaux*[3]

[2] d'après l'Usine Nouvelle, juillet 1991

[3] d'après la revue Economie et Humanisme, novembre-décembre 1988

I.1.2. Moyens automatisés chez PSA Peugeot Citroën

I.1.2.a Exemples d'installations automatisées [AUM 88]

Afin de mettre en évidence l'importance et la nature des installations automatisées chez PSA, nous présentons brièvement deux exemples du groupe. Le premier illustre les gains de productivité réalisés grâce aux automatismes dans l'usine Citroën de Borny. Le second concerne la Française de Mécanique et indique que l'automatisation ne se limite à la substitution des opérateurs humains par des machines, mais implique également de développer un système performant pour le pilotage de l'atelier de production.

CITROEN A BORNY[4] :

En 1987, cette usine, située dans la banlieue de Metz, a fabriqué 367.000 boîtes de vitesse. L'année suivante, suite à des investissements pour un montant d'environ un milliard de francs, la production à été de 678.000 boîtes, soit une augmentation de cadence de 87 %. 780 millions ont été consacrés à l'usinage et 160 au montage. Le totale représente 358 machines, 60 robots, 138 contrôles automatiques, 151 commandes numériques, 800 automates programmables, 23 applications informatiques, 2 kilomètres de convoyeurs sur une surface de fabrication de 33.000 m^2.

L'introduction de matériels performants a été accompagnée d'un effort important de formation : 5 % de la masse salariale et 80.000 heures par an (soit près d'une semaine par salarié). Les gains de productivité réalisés en l'espace d'une dizaine d'années, par l'introduction de systèmes automatisés associés à une main d'oeuvre qualifiée, sont considérables : vers la fin des années 70, le ratio était d'un homme par jour pour une boîte de vitesse fabriquée, fin 1988 celui-ci est descendu à 0,38 homme !

FRANÇAISE DE MECANIQUE[5] :

C'est dans cette usine, située à Douvrin près de Lille, qu'est fabriqué le petit moteur TUF (un 1300 cm^3 à bloc-moteur en fonte qui équipe entre autres les AX, 106 et 205) du groupe PSA Peugeot Citroën. La ligne de montage est automatisée à 50 %, avec un temps de cycle de vingt-six secondes par moteur.

---

[4] d'après l'Usine Nouvelle, juin 1988

[5] d'après l'Usine Nouvelle, novembre 1991

Une application essentielle de cet atelier est sa supervision. Elle regroupe dans une base de données toutes les informations sur la fabrication (programmes machines, suivi de qualité, descriptions des produits etc.). Les 53 commandes numériques et les 50 automates de la ligne sont connectés par un réseau industriel comportant cinq ordinateurs serveurs eux-mêmes reliés à l'ordinateur de supervision. Selon le type de moteur à fabriquer, ce dernier charge les informations nécessaires dans l'étiquette embarquée, associée au moteur et dans les automates et commandes numériques de la ligne.

Réciproquement, chaque poste transmet à la supervision les données relatives aux opérations effectuées. La supervision interroge également toutes les secondes les équipements pour contrôler les temps de cycle, détecter les pannes etc. Ce système permet de suivre constamment le fonctionnement du procédé automatisé pour disposer du maximum d'informations sur son évolution afin d'en optimiser le fonctionnement.

I.1.2.b Moyens d'automatisme

Le premier robot est apparu dans le groupe en 1974, mais c'est réellement à partir de 1981, que la croissance du parc installé est devenue forte. En 1991, le nombre de robots installés dans la division automobile était de 2155. La même année, le parc des directeurs de commandes numériques (CN), dans le groupe PSA représentait 1951 équipements.

Au niveau des moyens de commande et de pilotage d'atelier, il est intéressant d'observer l'évolution du parc installé des ordinateurs et automates programmables.

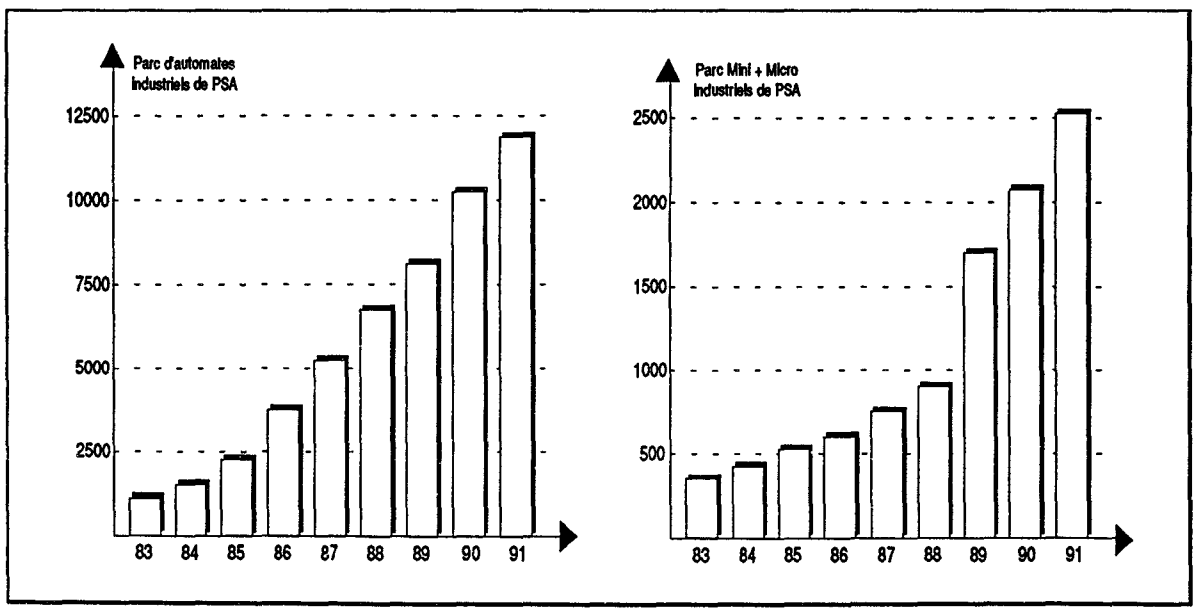


Figure I.4 : *parc installé d'automates programmables et d'ordinateurs pour le groupe PSA*

Si la croissance du nombre d'automates programmables est forte (16 % en 1991 et environ 25 % les années précédentes), celle du nombre de réseaux locaux industriels est encore plus élevées : 396 réseaux installés en 1990 et 532 en 1991, soit une croissance de près de 35 % !

I.1.3. Une automatisation maîtrisée

Comme nous venons de le voir, le groupe PSA a, depuis 10 ans, consacré des efforts considérables pour automatiser la production de ses véhicules. Cette démarche a permis de réaliser des gains de productivité très importants, notamment grâce à la réduction des coûts de main-d'oeuvre.

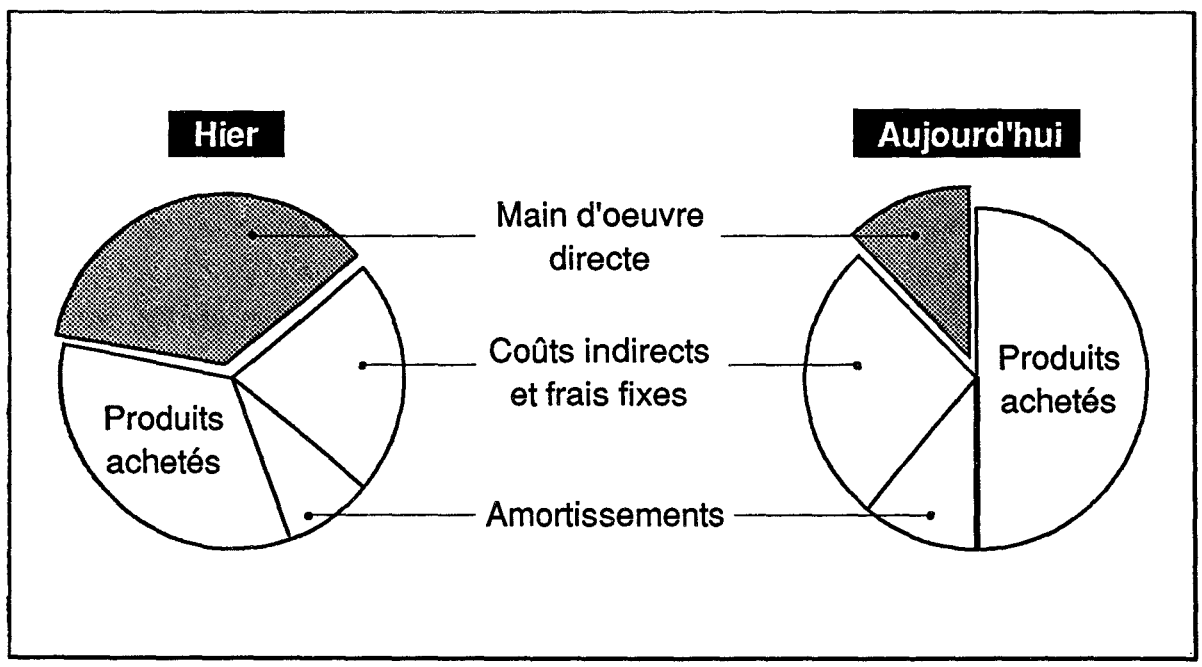


Figure I.5 : *une nouvelle répartition des coûts de production*[6]

Mais les moyens automatisés n'offrent pas toujours la fiabilité attendue. Ainsi, si le taux de disponibilité des équipements est de 80 à 85 % au Japon, il n'est que de 65 à 70 % en Europe[7] ! C'est pourquoi, la volonté actuelle est de mieux maîtriser les systèmes automatisés, afin de réaliser des gains de productivité nécessaires en fiabilisant les installations et en réduisant ainsi les pannes.

---

[6] d'après Ingersoll Engineers (Les Echos du12/03/92)

[7] d'après LA TRIBUNE de l'Expansion, mai 1992

Par ailleurs, la nouvelle charte de développement produit impose de réduire de cinq à quatre ans le temps nécessaire entre la décision de fabriquer un nouveau modèle et sa commercialisation. Comme le montre la figure I.6, le nouveau planning ne laisse alors que 112 semaines pour concevoir et réaliser les moyens de production.

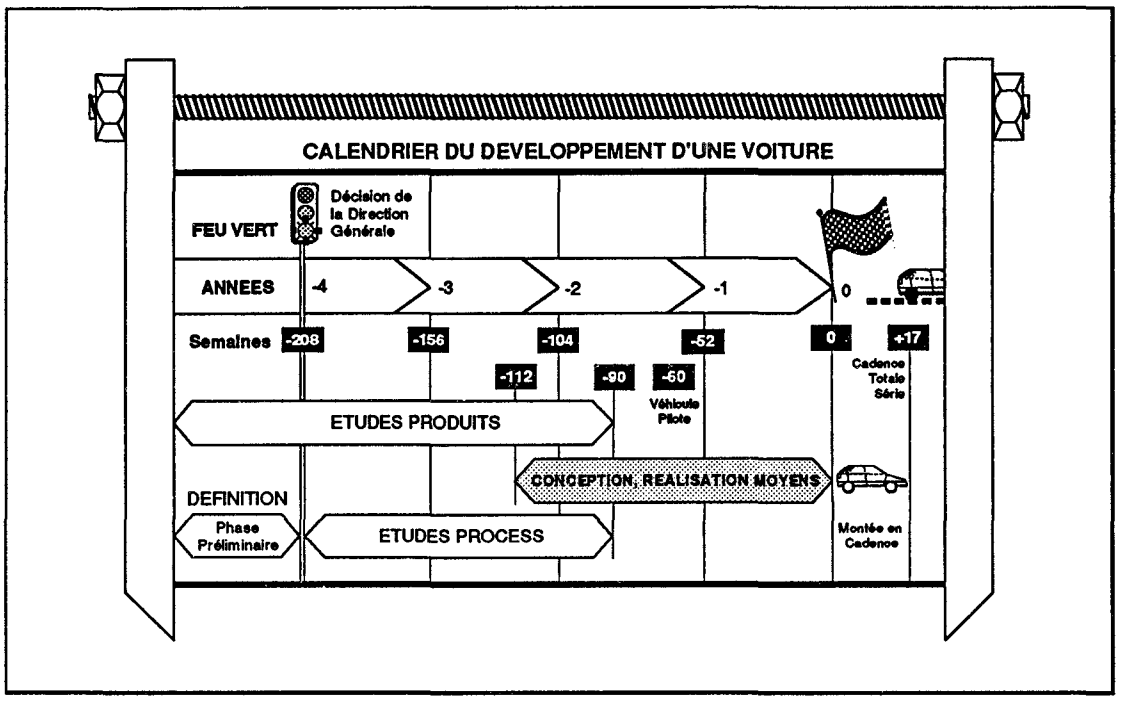


Figure I.6 : *charte de développement produit du groupe PSA Peugeot Citroën*[8]

Afin d'être en mesure de concevoir dans le cadre de ce délai, des installations automatisées avec les degrés de performances et de fiabilité attendus, le groupe PSA doit se doter de méthodes et d'outils efficaces pour assister la conception de ses ateliers automatisés.

I.2. Approche d'intégration du CIM

I.2.1. Objectifs

L'ambition du CIM (Computer Integrated Manufacturing) [WAL 90] vise à aider toutes les fonctions de l'entreprise, bien au-delà des seules fonctions de production, à établir des relations étroites, systématiques et fréquentes entre elles, en utilisant au mieux l'une des ressources fondamentales de l'entreprise : l'information.

[8] d'après l'ARGUS, janvier 1992

Le CIM peut être perçu comme un ensemble de systèmes d'informations intégrés, permettant à chaque personne dans l'entreprise de bénéficier d'un accès à toutes les données nécessaires à son travail.

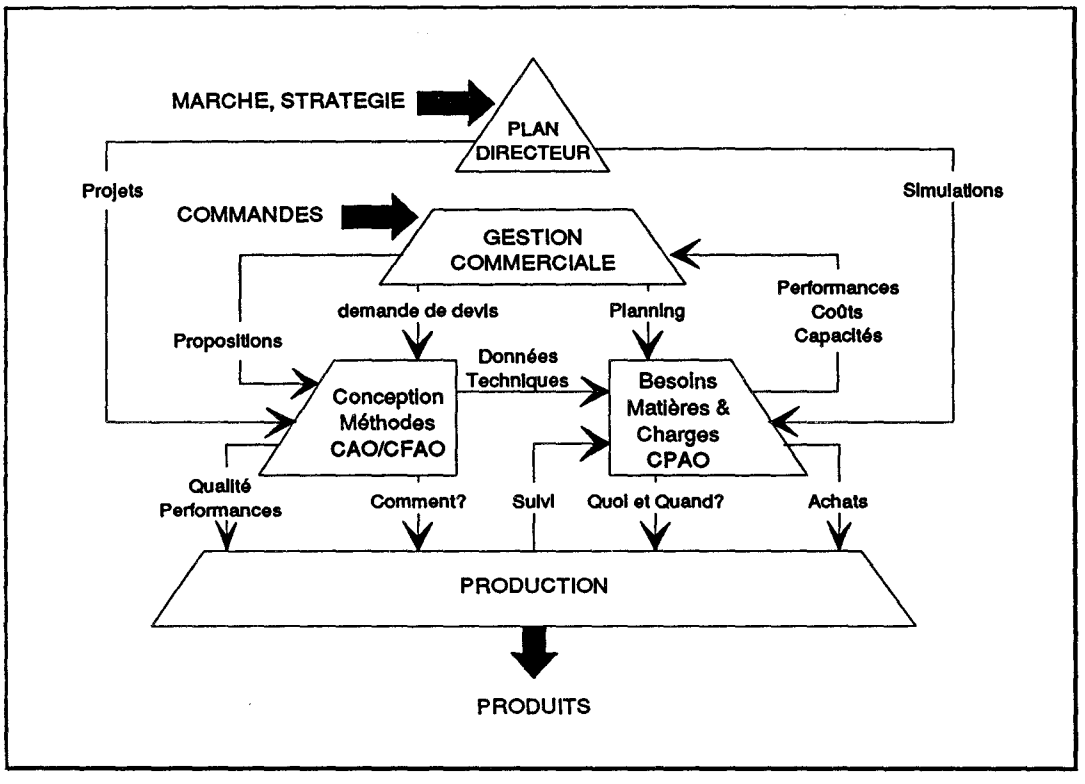


Figure I.7 : *architecture d'un système d'informations de l'entreprise*

I.2.2. Mise en oeuvre dans l'atelier

Une décomposition classique consiste à hiérarchiser les fonctions de l'entreprise selon cinq niveaux [BEU 88] :

niveau 4	Planification et gestion
niveau 3	Suivi de production
niveau 2	Conduite des moyens de production
niveau 1	Commande des moyens de production
niveau 0	Interface avec les moyens de production

Figure I.8 : *hiérarchisation des fonctions pour un atelier manufacturier*

Cette approche conduit à la réalisation de systèmes distribués, où les différentes fonctions de contrôle/commande sont réparties sur une architecture matérielle hiérarchisée, composée d'équipements distants interconnectés par des réseaux locaux industriels.

L'organisation humaine de l'atelier, découpé en lignes, modules et postes, se retrouve dans la hiérarchisation du système de pilotage [KRO 91]. De manière générale, nous pouvons représenter une architecture de commande d'atelier par le schéma suivant :

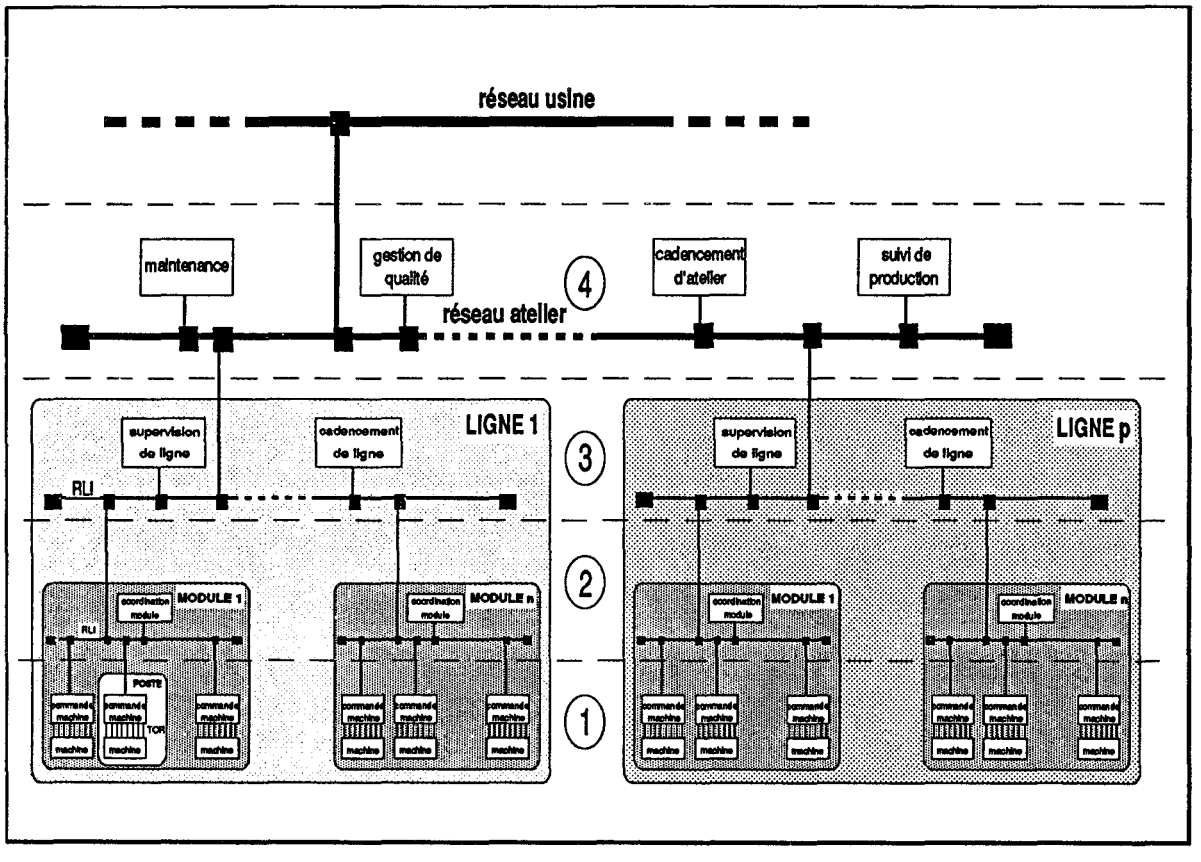


Figure I.9 : *schéma type d'architecture de commande et de pilotage d'atelier*

Les fonctions de niveaux 1 et 2 sont généralement implantées sur des automates programmables industriels (A.P.I.) [BOS 88], des armoires de commandes robots (C.R.) et des commandes numériques (C.N.) [COI 89]. Les fonctions de supervision et de cadencement sur la ligne sont réalisées par des ordinateurs de type industriel. Pour le niveau 4, on utilise des calculateurs de gestion industrielle [MAT 91]. Cette organisation n'est envisageable que pour les ateliers de taille importante. Dans le cas d'ateliers plus petits, elle est simplifiée.

Des réseaux locaux industriels [LEP 91] assurent la communication entre les équipements. Ils doivent permettre de transmettre des données dans des délais imposés par le procédé. Les contraintes de temps dépendent des spécificités de chaque installation mais pour fixer les idées, nous pouvons considérer que :

- un réseau de module doit garantir des délais compris entre 50 et 500 ms,
- un réseau de ligne doit assurer des temps de réponse de l'ordre de la seconde,
- un réseau d'atelier possède des contraintes de quelques secondes.

I.2.3. Conclusion

Il apparaît ainsi qu'assurer des communications entre les différents systèmes devient un axe majeur pour l'amélioration de la coordination et la coopération des fonctions de production. La technologie des réseaux locaux industriels (R.L.I.), véritable fédérateur du CIM est actuellement en passe de devenir le secteur phare de la productique, si on se réfère à la poussée du marché présent et de ses perspectives futures (Cf. figure I.10). D'ailleurs, comme nous l'avons vu dans le paragraphe I.1.2, cette croissance est en accord avec celle observée dans le groupe PSA Peugeot Citroën.

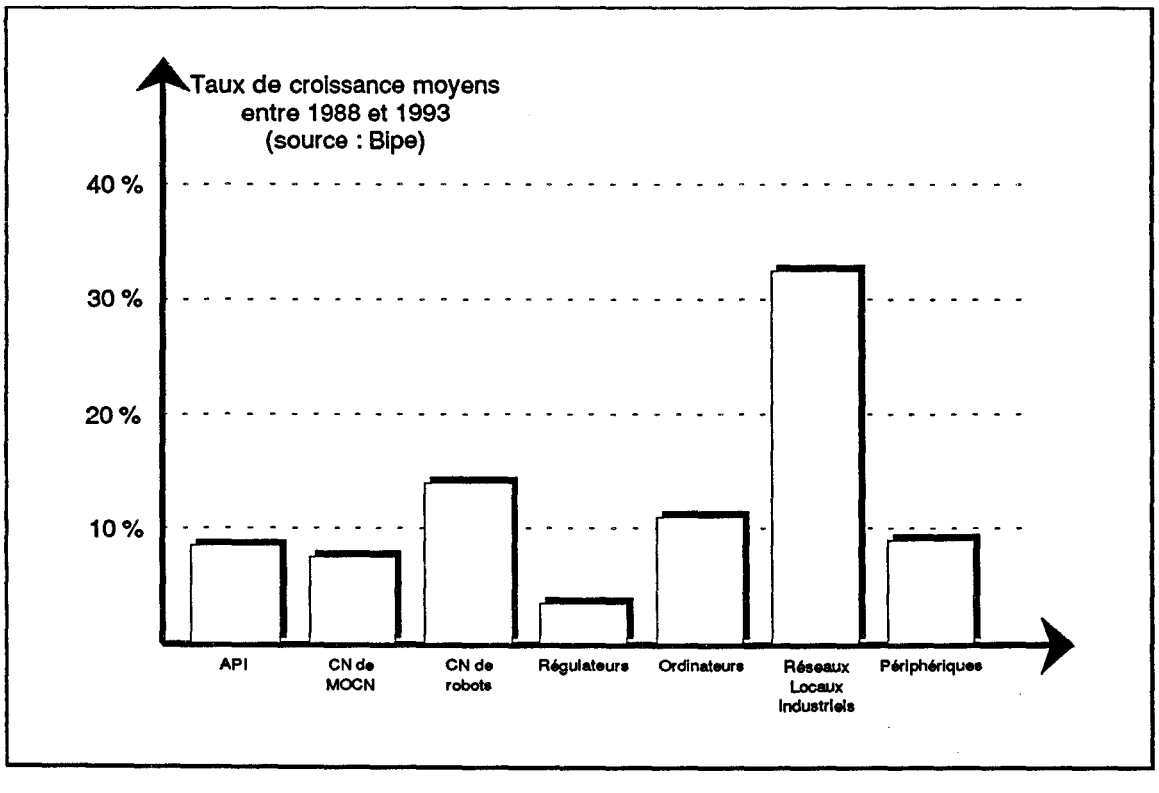


Figure I.10 : *taux de croissance des matériels d'automatisme*

I.3. Architecture de commande d'atelier chez PSA Peugeot Citroën

La réalisation de systèmes automatisés de production pour le groupe PSA Peugeot Citroën conduit à la réalisation d'installations qui peuvent comporter plusieurs centaines d'automates programmables et plusieurs dizaines de calculateurs industriels, qui concourent à la commande et au pilotage des moyens de production.

Afin de bien situer ce que représente une architecture de commande d'atelier chez un manufacturier automobile, nous allons décrire, le système de commande et de pilotage de l'atelier de tôlerie Peugeot Mulhouse S10 (Peugeot 106).

Cet exemple est intéressant à détailler dans le cadre de ce mémoire, car il est représentatif des systèmes basés sur le concept CIM pour les installations manufacturières. Il permet d'avoir une vision pratique sur les types et le volume des équipements utilisés. Les différentes fonctionnalités assurées par le système de commande et de pilotage sont explicitées pour ensuite décrire plus particulièrement les flux de données impliqués par le choix d'architecture et les médias de communications employés.

I.3.1. Atelier de tôlerie

Une installation de tôlerie permet, par assemblages d'éléments de carrosserie issus des presses, de réaliser les caisses de véhicules. Dans un but de flexibilité des moyens de production [YVA 90], une même tôlerie est capable de fabriquer plusieurs types de produits. Le processus d'assemblage fonctionne par consolidation d'étapes dépendantes de plusieurs flux. Les constituants d'une carrosserie donnée sont préparés dans des parties spécifiques, pour ensuite être assemblés sur la ligne principale polyvalente.

Les options retenues dans le projet ont été conditionnée par la découpe fonctionnelle de l'atelier basée sur trois entités inclusives :

- le **secteur** : c'est la plus grande entité qui recouvre une notion de PRODUIT. Le secteur contribue à la fabrication d'un élément de carrosserie. Il est géré de façon autonome sous la responsabilité d'un "Chef de secteur". On distingue les secteurs de préparation qui assurent la fabrication d'un sous-ensemble principal et les secteurs polyvalents qui réalisent l'assemblage des sous-ensembles principaux.

- la **zone** : un secteur est constitué de plusieurs zones qui sont chacune le champ d'action d'un "Conducteur d'installation". La zone recouvre une notion d'EXPLOITATION. En général, chaque zone a la capacité de fonctionner de façon autonome.

- l'îlot : une zone est constituée de plusieurs îlots. L'îlot est la plus petite entité incluant un ensemble d'équipements fonctionnellement très liés. Il recouvre une notion d'AUTOMATISME, généralement dépendant du champ d'action d'un arrêt d'urgence.

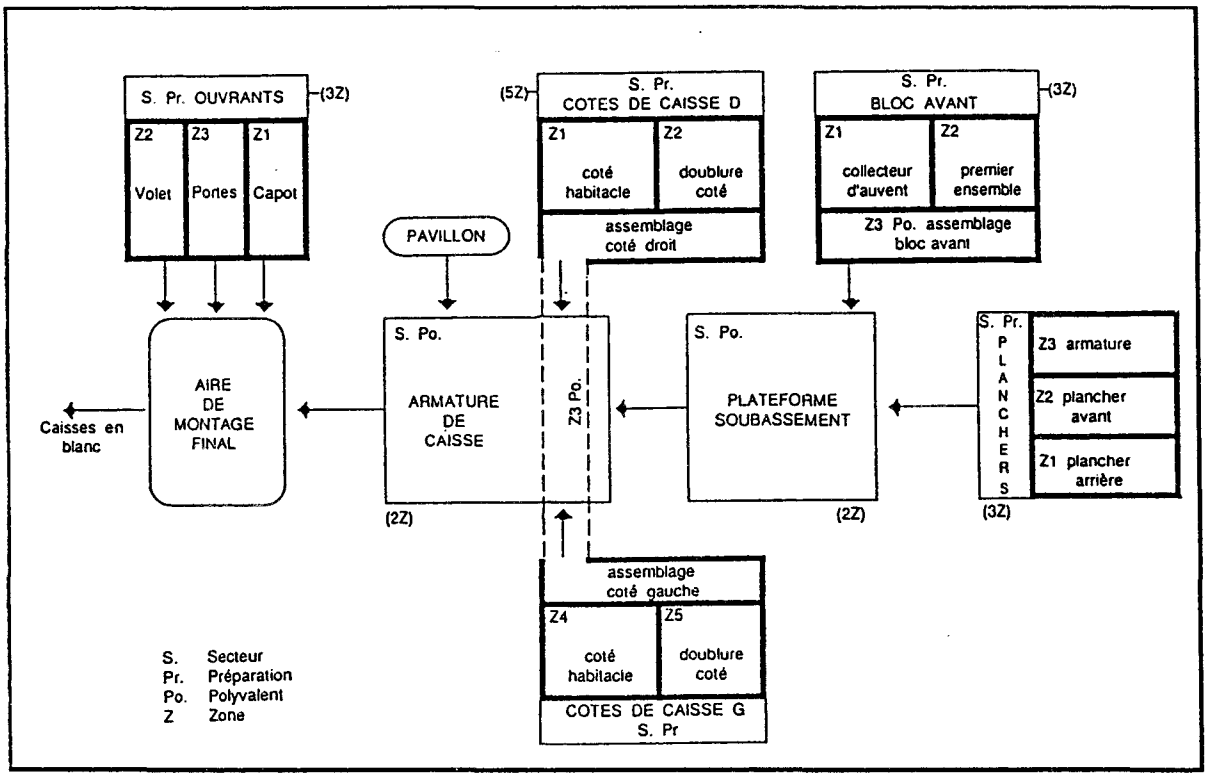


Figure I.11 : *Découpe d'un atelier de tôlerie*

I.3.2. Spécification des besoins

Outre la mise en oeuvre des automatismes liés aux machines de fabrication, les besoins essentiels concement :

- la **gestion de production** (cadencement) : au niveau de chaque secteur, l'engagement de la production est réalisée avec des calculateurs locaux de cadencement (CLC) et le calculateur central d'ordonnancement (CCO). L'utilisation de plots codés embarqués sur les lignes polyvalentes permet l'identification du produit.

- la **supervision des secteurs** : le besoin concerne le suivi et l'analyse des flux et de la fiabilité des équipements, d'où la nécessité de constituer une base de données propre à chacune des zones constituant les secteurs.

- la **conduite de zone** : les besoins exprimés sont ceux des conducteurs d'installation (analyse et suivi de production, téléchargement d'urgence des

programmes applicatifs API, baies robots et commandes numériques, commande centralisée des modes de fonctionnement etc.).

- la **conduite locale** : la recherche de la minimisation des temps d'arrêt entraîne des besoins d'aide au dépannage et d'aide à la remise en cycle.

I.3.3. Principes retenus

En fonction de l'analyse des besoins les principes retenus pour élaborer l'architecture de commande et de pilotage de l'atelier concernent essentiellement :

- **l'intégration dans l'existant** : il est apparu nécessaire de garantir la continuité avec l'existant tant vis à vis des matériels que de l'organisation humaine d'exploitation,

- **l'autonomie des procédés** : recherchée au niveau le plus bas, elle a conduit aux démarche de décentralisation des traitements, non dépendance des unités de contrôle et de supervision de niveau 2 et 3 et de suivi et d'identification des pièces sur les mobiles de la ligne.

- **le traitement et les flux** : afin de minimiser les temps de fabrication, une option d'anticipation des traitements a été retenue. Elle consiste à préparer un poste N à son initiative par prise d'informations sur la pièce au poste N-1. Il a par ailleurs été décidé de séparer les flux de cadencement des autres échanges de données.

- **la disponibilité et la fiabilité des équipements** : pour améliorer la disponibilité des équipements, les concepteurs ont opté pour la mise en oeuvre d'une fonction de surveillance, séparée des fonctions de commande.

I.3.4. Architecture CIM installée

I.3.4.a Présentation de l'architecture

La structure choisie est en parfaite adéquation avec l'organisation du personnel de production et la découpe fonctionnelle de l'atelier qui a été présentée au paragraphe I.3.1.

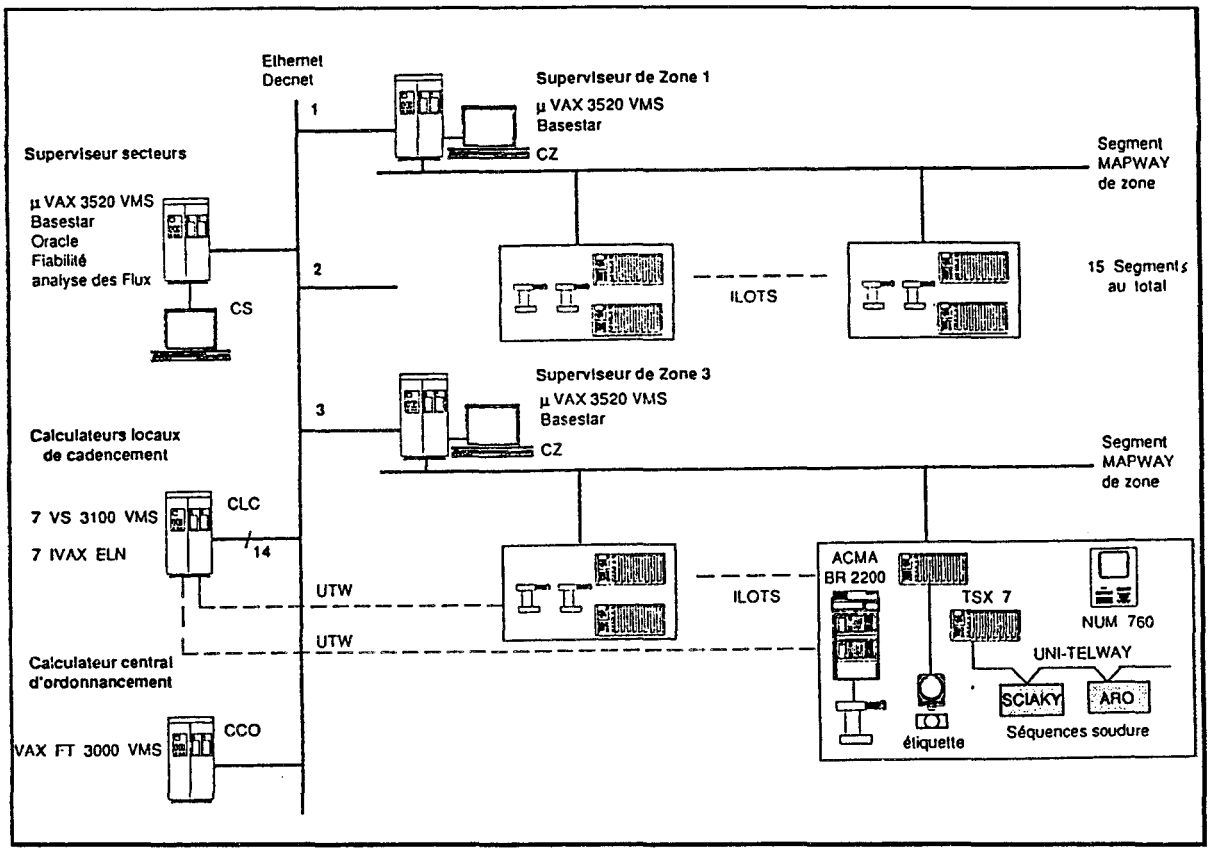


Figure I.12 : *architecture de commande et de pilotage de l'atelier de tôlerie Peugeot S10*

I.3.4.b Flux et médias de communication

Pour tous les secteurs, excepté celui des ouvrants, chaque zone comprend un segment de réseau MINI-MAP (Mapway). Celui-ci est fédérateur de tous les équipements d'automatisme de niveau 1 impliqués dans le flux principal des pièces propres à une zone. Il assure le premier flux d'informations nécessaires à la mise en oeuvre :

- de l'anticipation,
- de la supervision du suivi de production et de fiabilité des équipements,
- de conduite de zone,
- de téléchargement des programmes.

Dans le cas des lignes d'assemblages, le flux des informations de cadencement, second flux volontairement séparé du premier, est assuré par deux liaisons de type Uni-Telway entre calculateur de cadencement et deux automates en général, l'un en début et l'autre en sortie de ligne. Vis à vis de ce flux, ces automates servent uniquement de relais de communication aux plots mémoires (étiquettes) solidaires du mobile.

Au départ de chaque ligne, l'étiquette est codée puis à chaque poste elle est lue pour identifier le travail à effectuer. En retour, chaque poste y inscrit toutes les informations relatives à la bonne ou mauvaise exécution des tâches, mises à profit par le poste suivant pour autoriser ou interdire de travailler sur la pièce. Les étiquettes mettent donc en oeuvre un troisième flux d'informations de description du produit et de qualité.

Enfin un réseau Ethernet d'atelier assure un quatrième flux de données nécessaire pour la mise en oeuvre de l'ordonnancement et des fonctions propres aux consoles secteur.

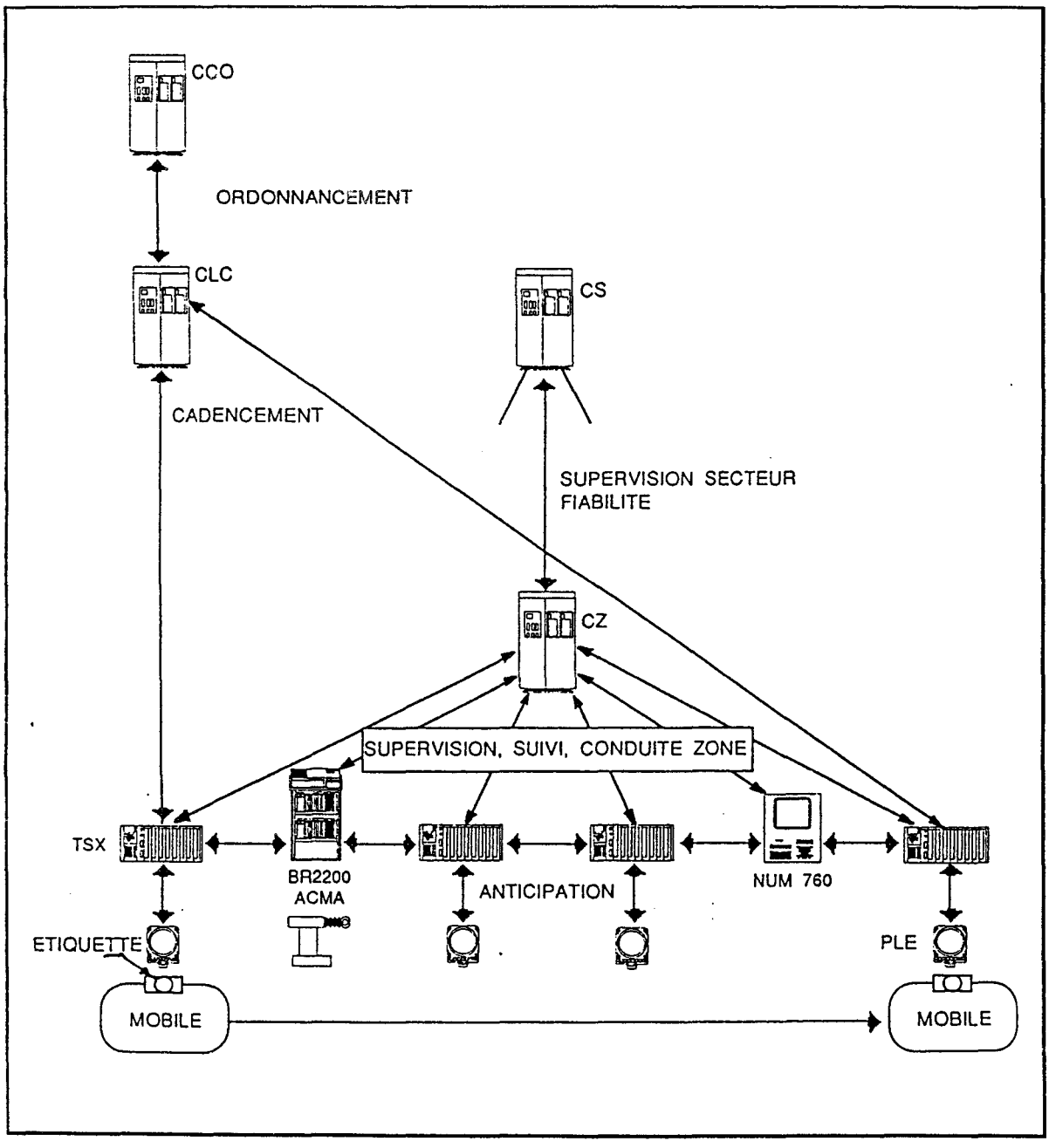


Figure I.13 : *besoins en communication de l'architecture*

I.3.4.c Description des composants de l'architecture

1 - Equipements d'automatisme de niveau 1 :

L'automatisation de l'atelier à nécessité la mise en oeuvre des équipements suivants :

- 15 segments Mapway,
- 342 automates TSX série 7
  - . 155 de type TSX 17 et 27
  - . 187 de type TSX 47, 67 et 87 dont 140 connectés au réseau Mapway,
- 153 baies de robots ACMA BR2200 dont 139 connectées au réseau Mapway,
- 6 commandes numériques NUM 760 dont 5 sur le réseau Mapway.

2 - Equipements informatiques de niveau 2 :

Chaque zone câblée en Mapway comporte une console de zone (CZ) basée sur un ordinateur VAX 3520. Outre leur connexion au réseau Mapway de zone, les 15 CZ sont reliés au équipements de niveau 3 via le réseau Ethernet. Elles mettent en oeuvre des applicatifs de conduite de zone et de remontée des informations de fiabilité.

3 - Equipements informatiques de niveau 3 :

Les équipements de niveau 3 sont fédérés sur un réseau Ethernet. La fonction d'analyse des flux et de fiabilité des équipements de tous les secteurs est consolidée dans une console de secteur (CS) basée sur un ordinateur VAX 3520. La fonction de cadencement locale (CLC) s'articule autour de deux calculateurs : une VAX Station VS 3100 pour l'aspect supervision du cadencement et un Micro-VAX sous le système d'exploitation Vax-Eln pour l'aspect "temps réel" du cadencement. Les "couples" CLC sont au nombre de 7 et sont en connexion avec le calculateur central d'ordonnancement (CCO) basé sur un système à tolérance de pannes VAX FT 3000.

II. ELABORATION DES ARCHITECTURES

II.1. Méthodes et outils disponibles sur le marché [ALA 88]

II.1.1. Solutions proposées par les constructeurs

Chaque constructeur d'équipements de contrôle/commande d'automatisme, propose un environnement de développement dédié à ses matériels. Afin d'en regarder les potentialités, nous présentons brièvement les outils de deux marques très implantées dans le groupe PSA Peugeot Citroën : Télémécanique et Digital.

X-TEL DE TELEMECANIQUE [TEL 90a][TEL 90b] :

X-TEL est un atelier pour le développement et la mise au point d'applications d'automatisme à base de produits Télémécanique. Il propose un certain nombre d'outils intégrés, qui se positionnent dans les différentes phases du cycle de vie d'une automatisation, de la conception à l'exploitation. Ces outils peuvent se classer en trois catégories :

- **les outils généraux de base** : il s'agit d'outils logiciels communs qui permettent d'améliorer la productivité en optimisant la gestion des projets, des modules, des symboles et des documentations,
- **les outils de programmation** : ils permettent de développer les fonctions d'automatismes à implanter dans les équipements de l'architecture (programmation des automates en langage PL7-2 ou PL7-3, paramétrage des coupleurs spécialisés, description et configuration de l'architecture de communication etc.),
- **les outils d'exploitation** : ils sont dédiés au suivi d'une installation automatisée pour visualiser, régler et diagnostiquer la bonne exécution de la partie opérative (analyse de trafics réseaux, visualisation et forçage d'état de variables dans un programme etc.)

VAXSET DE DIGITAL [DIG 89] :

VAXset est un ensemble d'outils intégrés proposés par Digital pour le développement d'application sous le système d'exploitation VMS. Six produits principaux sont disponibles :

- CMS (Code Management System) : gestion évoluée des versions de fichiers tout au long du développement d'une application,
- LSE (Language Sensitive Editor) : édition enrichie de programmes écrits en langage de haut niveau (Pascal, Ada, C etc.),
- SCA (Source Code Analyzer) : génération de références croisées et analyse statique de code source dans des langages de haut niveau (Pascal, Ada, C etc.),
- MMS (Module Management System) : automatisation des phases de compilation et de linkage pour produire le fichier exécutable,
- DTM (DEC/Test Manager) : automatisation des phases de test,
- PCA (Performance and Coverage Analyzer) : mesures de couverture et de performances lors de l'exécution de l'application.

II.1.2. Autres solutions disponibles

Les outils proposés en standard par les constructeurs restent très proches des machines cibles, et sont principalement axés sur l'écriture de programmes et la génération de code exécutable. Ils ne considèrent pas du tout les phases amont et en particulier l'étape de spécification. Pour pallier cette insuffisance, d'autres outils sont disponibles. Nous en avons retenu trois, qui illustrent les tendances actuelles d'évolution.

STP D'IGL TECHNOLOGIE [IGL 90] :

STP (Software Through Picture) est un environnement d'analyse et de conception de logiciels qui permet :

- **de modéliser** les traitements et les informations, les flots de contrôle, les structures de programmes et de données, et ce tant au niveau conceptuel que physique,
- **d'intégrer** ces différents modèles au sein d'un projet à l'aide d'un dictionnaire de données,
- **de vérifier** de manière extensive la cohérence et la complétude des modèles,
- **de produire** des schémas de bases de données ainsi que des squelettes de programmes,
- **de documenter** l'ensemble du travail réalisé, sous une forme directement publiable.

Ce produit est basé sur l'utilisation de plusieurs techniques de modélisation : analyse structurée (Yourdon/Demarco), Conception structurée, Analyse structurée temps réel (Hatley), Modèle entité/relation (Chen) [HAT 88]. Un tel environnement met l'accent sur la spécification, pour ensuite seulement assister la phase de conception. Le code généré en fin de cycle peut être du C, du Pascal ou de l'ADA, mais aucune ouverture vers les langages des automates programmables n'existe actuellement.

K-SYS DE GEOIDE [SCH 91] :

K-SYS apparaît comme une méthode, supporté par un outil logiciel, pour la conception des automatismes. Il permet de prendre en compte :

- **la spécification** : à partir d'une analyse fonctionnelle, le comportement des automatismes est déduit en tenant compte des aspects de séquence d'opérations, de traitement d'alarme et de traitement d'informations et de sécurité. La spécification est basée sur une description hiérarchique des installations (composant, équipement, poste etc.) avec la possibilité de réutiliser la spécification de sous-ensemble.
- **la validation** : pour chaque sous-ensemble, une simulation logique est possible, au travers d'un interface graphique qui permet de simuler les interactions avec l'environnement.
- **l'implantation** : à partir des spécifications fonctionnelles complètes, qui ont été validées par simulation, K-SYS génère automatiquement le code de l'application pour l'automate cible choisi. Actuellement des générateurs de code existent pour les automates Cegelec et Siemens.

SAGA DE VERILOG [PIL 92] :

SAGA est une boîte à outils intégrés autour d'un même format de représentation interne des données. Une application d'automatisme est décrite au moyen de dessins de type bloc-diagramme, ce qui permet une hiérarchisation des fonctions. Cette approche est donc adaptée à la culture des utilisateurs, surtout pour les processus continus. Il est élaboré autour du langage synchrone LUSTRE [HAL 91], ce qui permet de réaliser des vérifications de cohérence forte et ouvre la voie à l'utilisation de preuve logicielle.

Dans le même ordre d'idées, des travaux sont en cours pour passer d'une spécification Grafcet à un code intermédiaire issus des langages synchrones [AND 92]. Cette démarche devrait permettre à terme de disposer pour les automatismes séquentiels des avantages de l'approche synchrone (validation du code, exécution optimisée etc.).

II.1.3. Conclusion

Tous les outils qui ont été présentés dans le paragraphe II.2 ont pour objectif principal d'aboutir à l'écriture du code qui sera implanté dans les équipements de commande. Toutefois aucune assistance n'existe, ni pour répartir les fonctions de commande et de pilotage au sein d'une architecture, ni pour concevoir cette architecture.

II.2. Travaux de recherche

II.2.1. Projet CASPAIM [BOU 91][CRU 91]

Le projet CASPAIM (Conception Assistée des Systèmes de Production Automatisée dans l'Industrie Manufacturière) est développé par le laboratoire d'automatique et d'informatique industrielle (LAIL) de l'Ecole Centrale de Lille. Il a pour objectif le développement d'une méthodologie de conception des systèmes de production flexibles. Cette méthode intègre à la fois des aspects de commande et contrôle, de supervision et surveillance, ainsi que des aspects de gestion de production.

Les gammes opératoires, qui décrivent le chemin de chaque produit dans le système de production, constituent le point d'entrée de la démarche de CASPAIM. Elles permettent de construire un graphe de commande avec un formalisme de réseaux de Petri structurés. Des travaux ont permis d'aboutir, à partir de ce graphe, à la génération d'une commande exprimée en Grafcet [CRA 89].

Celle-ci détaille la structure des programmes de commande au niveau de la cellule. Mais c'est ensuite au concepteur de choisir les équipements et l'architecture matérielle pour implanter ces programmes. C'est également à lui de répartir les Grafcet entre les différents automates programmables.

II.2.2. Autres travaux proches du sujet traité

Face aux besoins ressentis par les concepteurs de systèmes automatisés, un groupe de travail a été créé en 1984, pour fédérer les travaux autour d'une nouvelle discipline, dénommée Génie Automatique. Leur objectif est de spécifier un poste de travail pour automaticien (projet PTA). Récemment, un projet de norme a été déposé [AFN 91] qui

définit un modèle de données normalisées, devant servir de référence pour le développement d'un ensemble d'outils modulaires qui formeront le poste de travail de l'automaticien.

Quelques outils existent déjà dans ce cadre, et nous pouvons par exemple mentionner le produit SPEX développé dans le cadre d'une collaboration entre les sociétés SPIE-TRINDEL, T.N.I. et le laboratoire d'automatique et de commande numérique du centre de recherche en automatique de Nancy (CRAN/LACN) [PAN 91]. Il s'agit d'un environnement de prototypage pour la commande, exprimée en Grafcet, d'un système automatisé. Il permet de décrire également la "partie commandée", ce qui permet dès la phase de conception la réalisation de simulations en boucle fermée.

II.2.3. Conclusion

Les quelques exemples présentés montrent que les travaux en cours concernant le génie automatique ont pour principale vocation de proposer des méthodes, des formalismes et des outils associés pour arriver à produire dans de meilleures conditions les programmes de commande des automatismes. Cependant le problème du choix des équipements pour supporter ces applicatifs n'est pas abordé.

Il s'agit d'un problème réel et majeur pour les industriels, qui disposent en effet d'une offre qui ne cesse de s'enrichir de la part les fournisseurs de matériels d'automatisme. L'aspect temps réel est une contrainte importante pour les systèmes de commande et de pilotage. Ce caractère est directement lié au choix des matériels d'automatisme, et à l'organisation des échanges de données au sein de l'architecture de commande et de pilotage. C'est dans ce contexte de préoccupations que nous avons orienté les travaux présentés dans ce mémoire.

II.3. Traitement de l'aspect performances

II.3.1. Performances des systèmes de contrôle/commande

La première fonction d'une architecture de commande et de pilotage est d'assurer la commande des moyens automatisés de production [VAL 80]. Par l'intermédiaire des capteurs, le procédé informe les équipements de niveau 1 de son évolution. Ces événements sont pris en compte par la commande qui génère de nouvelles consignes vers

le process. Il existe donc une interaction permanente entre les événements en provenance du procédé et les consignes de commande.

Un système de commande doit par conséquent être en mesure d'assurer des temps de réponse qui soient compatibles avec les temps de réaction du procédé commandé. Il est en particulier totalement inadmissible que la commande réalisée les automatismes pénalise la production. Lorsque les temps de cycles sont courts (typiquement quelques dizaines de secondes), la commande et en particulier les échanges de données entre les automates de chaque poste nécessite d'être particulièrement bien étudiée pour ne pas perturber la production théorique. Notamment, il est apparu, sur un exemple précis de machine transfert du groupe PSA Peugeot Citroën, qu'en remplaçant uniquement le réseau de communication par un réseau plus performant la productivité était augmentée !

Une architecture de commande et de pilotage assure l'échange d'informations avec des opérateurs humains chargés de piloter l'installation : états des en-cours, alarmes sur incidents, taux d'utilisation des moyens etc. Cette restitution de données est réalisée, en temps réel, avec des consoles opérateurs, des écrans de supervision, des imprimantes etc. De la même manière, le conducteur d'installation a la possibilité d'envoyer des commandes vers le process. Pour des raisons de souplesse d'utilisation, et parfois de sécurité, il est important que les temps de réponse associés soient admissibles pour un utilisateur. Il n'est par exemple guère acceptable que les temps de changement de vue sur un écran de supervision se chiffrent en dizaines de secondes !

Les temps de réponses constituent le critère le plus facilement mesurable vis-à-vis des performances d'un système. D'autres paramètres ont également leur importance, notamment en terme d'évolutivité : charge d'un réseau de communication, pourcentage d'utilisation d'une ressource etc. Ces critères sont importants, car ils permettent d'estimer des taux de charge et donc d'avoir une idée sur la capacité du système à admettre des fonctionnalités supplémentaires.

II.3.2. Moyens disponibles pour valider les performances

Les fabricants d'équipements d'architectures de commande et de pilotage fournissent, avec la documentation de leurs produits, des renseignements concernant les performances. On connaît par exemple le nombre de MIPS d'un ordinateur, le temps d'accès moyen pour un disque dur, ou encore le temps de transmission pour une requête sur un réseau de communication. Parfois les informations sont plus précises, et par exemple Télémécanique

ou Siemens indiquent, pour leurs automates programmables, le temps d'exécution de chaque instruction.

Cependant, chaque constructeur possède ses propres paramètres de mesures de performances, et il est parfois difficile d'établir des correspondances pour pouvoir comparer des équipements de marque différente. Les renseignements fournis ne sont pas forcément complets, ce qui peut obliger l'utilisateur à réaliser (ou à faire réaliser) des mesures complémentaires. La meilleure solution est alors de définir des benchmarks [BIN 90], qui permettent d'obtenir dans de bonnes conditions des mesures comparatives.

Ces informations, si elles sont importantes, ne sont toutefois pas suffisantes pour être en mesure d'évaluer les performances d'une d'architecture de commande. En effet, elles sont toujours réalisées dans des contextes particuliers d'utilisation (souvent simples) qui ne reflètent pas le fonctionnement réel lorsque l'équipement est intégré dans une architecture complète.

Une première solution consiste alors à "calculer", d'après les mesures disponibles, et en se basant sur le fonctionnement imaginé de l'architecture, les performances du système vu dans son ensemble. Cependant, une telle approche nécessite, d'une part de bien connaître le fonctionnement interne des équipements pour comprendre quels sont les critères qui influent sur leurs performances, et d'autre part d'appréhender l'évolution parallèle des différentes fonctions réparties sur l'architecture. Une telle approche permet d'avoir une idée sur le fonctionnement stationnaire de l'installation mais rend extrêmement difficile l'estimation des pointes de charge, où de cas singuliers dans lesquels peut se trouver le système étudié (goulets d'étranglements par exemple).

Une autre solution de type expérimentale consiste à développer en plate-forme un sous-ensemble, jugé représentatif de l'architecture envisagée. Il est alors possible de mesurer directement les performances. Cependant, Il ne peut s'agir que d'un "sous-ensemble" d'architecture et il n'est pas forcément simple d'extrapoler, à partir des mesures réalisées les performances du système global. De plus, une telle approche est coûteuse, à la fois du point de vue financier, mais aussi en temps, surtout si différentes solutions sont testées successivement.

II.3.3. Conclusion

Les performances d'une architecture de commande et de conduite d'atelier constituent une contrainte essentielle au moment de la conception d'un système. Il s'agit pourtant d'un aspect encore mal maîtrisé, et c'est souvent seulement lors du démarrage d'une installation sur site que l'on découvre les performances effectives du système mis en place.

Dans le cas où les performances sont inférieures aux performances spécifiées et attendues, il est toujours très difficile et surtout extrêmement coûteux d'effectuer des modifications dans l'atelier pour limiter les conséquences d'erreurs de conception. C'est pourquoi, on préfère souvent surdimensionner les architectures afin de s'affranchir des problèmes liés aux performances. Dans un contexte où les investissements sont évalués au plus juste, cette approche est inacceptable, car elle a évidemment pour conséquence d'augmenter les coûts.

C'est pourquoi nous avons choisi d'orienter nos travaux sur l'étude des performances d'architecture de commande et de conduite d'atelier. Notre objectif est de proposer quelques moyens pour valider a priori les choix, afin que le concepteur maîtrise, dès que possible en cours d'étude, les performances de l'architecture qu'il envisage.

III. DEMARCHE DE CONCEPTION PROPOSEE

III.1. Simulation dans l'industrie

III.1.1. Contexte

La simulation est une méthode souvent efficace d'aide à la décision, disponible pour les concepteurs et les gestionnaires de systèmes complexes [MIL 90]. L'idée de base est simple et naturelle, puisqu'elle consiste à obtenir un modèle dont le comportement reflète celui du système étudié. Il devient alors possible de mener des tests et expériences sur cette image de la réalité pour en comprendre le fonctionnement, en améliorer la conception et éventuellement détecter des dysfonctionnements.

Des approches de simulation sont développées dans des domaines très variés, qui vont du simulateur de vol pour la formation des pilotes, au modèle économique pour prévoir le cours du blé, en passant par la modélisation du "crash" sur les véhicules automobiles ou encore la simulation CAO/Robotique pour calculer les trajectoires de robots [DOM 92]. Pour les systèmes de production automatisés différentes approches de simulation sont actuellement utilisées avec succès dans deux domaines principaux : la simulation des flux de production et la simulation de partie opérative.

III.1.2. Simulation de flux de production [CER 88][CHA 90]

Un modèle de système de production inclut toutes les caractéristiques physiques du système représenté : implantation des machines, moyens de transport, zones de stockage etc. Il tient compte des temps opératoires, des temps de transport, de la disponibilité des machines, des outils, des chariots de transport et même éventuellement du personnel ainsi que des pannes éventuelles sur les moyens de production.

Le modèle simule le fonctionnement du système de production sur plusieurs jours voire plusieurs semaines, en quelques heures de temps CPU. Le diagramme des événements associé aux résultats de simulation peut ensuite être étudié et des changements appropriés sont susceptibles d'être apportés au système afin d'en améliorer le fonctionnement : par exemple changer la capacité d'une machine, la taille d'un stock tampon ou le nombre de chariots filoguidés.

Après une suite d'expériences et d'ajustements successifs, il devient possible d'optimiser la conception du système de production, l'objectif étant d'obtenir celui qui donnera les performances requises pour l'investissement le plus faible.

III.1.3. Simulation de partie opérative [GRZ 91]

Pour mettre au point les programmes automate, l'automaticien doit être en mesure de simuler le fonctionnement du procédé commandé. Dans l'industrie manufacturière, l'interaction entre commande et procédé est principalement réalisée au travers d'entrées/sorties TOR. Les premiers automaticiens avaient l'habitude de travailler avec une boîte à boutons qui leur permettait de produire manuellement des événements du process pour étudier de quelle manière réagissait la commande.

Cette solution étant beaucoup trop contraignante (aucune possibilité d'enregistrer des scénarios de test, nombre d'entrées/sorties limité, difficulté pour vérifier les temps de réponse etc.) une première "automatisation" de cette tâche a consisté à utiliser un autre automate programmable dont les cartes de sortie sont directement reliées à celles d'entrée de l'équipement à tester. Le comportement du process est alors modélisé par un programme automate.

Les langages d'automatisme (Grafcet, Ladder etc.) sont assez mal adaptés pour décrire ce type de comportement. C'est pourquoi des logiciels spécifiques ont été développés ces dernières années sur compatibles PC ou station de travail, pour permettre une description beaucoup plus facile de la partie opérative. L'ordinateur est alors connecté au moyen d'une carte spécifique à l'automate à tester.

Si cette approche ne diminue pas forcément le temps total de développement d'une application, elle permet de réaliser des validations beaucoup plus poussées en plate-forme de développement et limite donc d'autant les phases de mise au point dans l'atelier.

III.1.4. Conclusion

Les concepteurs des systèmes automatisés de production commencent à être largement sensibilisés au concept de modélisation/simulation. C'est pourquoi nous pensons qu'aller plus loin, en n'ayant cette fois-ci une vision non plus au niveau d'un équipement isolé comme pour la simulation de partie opérative, mais au niveau d'un réseau ou d'une

architecture hiérarchisée de réseaux constitue une démarche, certes de la recherche, car totalement nouvelle pour ce type de problème, mais également suffisamment proche des préoccupations des utilisateurs concernés pour susciter leur intérêt.

III.2. Analogie entre procédé et commande

III.2.1. Dualité entre moyens de fabrication et moyens de commande

La simulation est utilisée avec succès pour optimiser le dimensionnement des ateliers de production manufacturiers [DIM 90]. Il nous semble intéressant de remarquer qu'une dualité existe entre les entités rencontrées dans un système de production et celles du système de commande et de pilotage associé.

		PROCEDE de FABRICATION	SYSTEME de COMMANDE
FONCTION		Fabriquer des PRODUITS	Traiter des DONNEES
MOYENS	Traitement	MACHINE / POSTE MANUEL Opération sur le produit en cours	API / ORDINATEUR / CN / CR Traitement sur les données
	Stockage	MAGASIN / STOCKS TAMPON Stockage de produit en cours	MEMOIRE / UNITE SAUVEGARDE Stockage de données
	Transfert	MANUTENTION Transport des produits en cours	RESEAUX / POINT à POINT Echanges de données
INTERFACE		Capteurs / Actionneurs	Actionneurs / Capteurs

Tableau I.2 : *dualité entre le système de fabrication et celui de commande*

Dans l'industrie manufacturière, un système de production a pour vocation de fabriquer des produits, à partir de matériaux bruts, ou de produits semi-finis. Les différentes opérations nécessaires pour obtenir le produit final sont réalisées sur des postes qui peuvent être manuels ou automatisés (poste de soudage robotisé, machines d'usinage, d'assemblage etc.).

Le transfert des produits semi-finis entre les postes successifs sont assurés par des moyens de manutention (chariots filoguidés, robot sur portique etc.). Des stocks

intermédiaires sur les en-cours sont souvent nécessaires pour absorber les fluctuations de production entre les postes, ainsi que pour limiter les conséquences d'aléas de fabrication.

Lorsqu'il s'agit d'un système de production automatisé, l'interface entre le procédé et le système de commande et de pilotage associé est assuré par des capteurs/actionneurs. C'est à ce niveau, qu'une partie des données utilisées par le système de contrôle/commande est interfacée avec le système physique de l'atelier.

Ces données sont collectées et traitées par les équipements de l'architecture de commande : automates programmables, ordinateurs industriels, commandes numériques et commandes robots. Ceux-ci disposent de capacités de calcul pour transformer ces données et/ou en générer d'autres. Elles sont mémorisées de manière temporaire ou définitive en mémoire vive ou sur des mémoires de masse. Les différents matériels de commande échangent des informations via des réseaux locaux industriels ou des liaisons point à point.

III.2.2. Conception - Dimensionnement

Si une dualité existe au niveau macroscopique, entre les entités d'un système de production et celles d'une architecture de commande et de pilotage, on peut également remarquer que certaines contraintes, notamment liées au dimensionnement de ces systèmes, sont similaires lors des phases de conception. Ainsi, dans les deux cas, la meilleure solution sera celle qui assure, au moindre coût, les fonctionnalités et les performances demandées.

Un système de production sera notamment limité par la capacité des machines et/ou des opérateurs humains, la taille maximale des stocks et les temps imposés par lès moyens de transfert. De la même manière, une architecture de contrôle/commande doit garantir des temps de réponse compatibles avec les contraintes imposées par le procédé à commander. Ceci induit des contraintes de performances sur les unités de traitement et sur les moyens d'échange d'informations.

III.2.3. Conclusion

Des approches de simulation sont utilisées avec succès pour le dimensionnement d'ateliers de production. Etant donné, qu'il existe une dualité, à la fois au niveau des entités composants ces systèmes et des contraintes de performances à considérer lors de leur

conception, nous pensons qu'il est intéressant de regarder de quelle manière une démarche de simulation à événements discrets pourrait être développée pour assister la conception des architectures de commande et de pilotage d'atelier.

III.3. Nouvelle démarche de conception proposée

III.3.1. Présentation de la démarche

A partir des besoins exprimés par l'utilisateur des moyens de production, une spécification fonctionnelle détaillée précise les fonctionnalités attendues du système de commande. Après validation des spécifications, le concepteur imagine une (ou plusieurs) architecture matérielle susceptible de supporter les fonctionnalités demandées : choix d'automates programmables, de calculateurs industriels, de réseaux locaux industriels, de commandes numériques, de terminaux opérateurs etc.

Sur les équipements de cette architecture sont réparties les différentes fonctions de contrôle/commande. Cette organisation est conditionnée par les possibilités de traitement, d'archivage et de communication des équipements. Il est également tenu compte des fonctionnements dégradés de l'installation : un dysfonctionnement, matériel ou logiciel, dans le système de commande doit perturber le moins possible la production de l'atelier. Des fonctions implantées sur des équipements distants qui ont des informations à échanger dialoguent par réseau. Il est donc indispensable de définir le contenu et le séquencement des échanges de données au sein de l'architecture de commande.

Lorsqu'une solution est proposée il faut s'assurer qu'elle supporte les performances qui lui sont demandées : temps de prise en compte d'une commande, temps de remontée d'informations, charge d'un réseau de communication etc. On ne peut accepter que le système de commande ralentisse le process de fabrication ! Actuellement les concepteurs s'en remettent à leur expérience, ou, pour les installations très critiques, testent en plate-forme un sous-ensemble du système. **NOUS PROPOSONS ICI DE VALIDER CES PERFORMANCES PAR SIMULATION.**

On voit l'intérêt d'une telle démarche de modélisation durant la phase de conception : les différentes solutions envisagées sont successivement modélisées, simulées et évaluées. Les erreurs sont détectées dès la phase de conception, ce qui assure la conformité des performances du système aux spécifications exprimées.

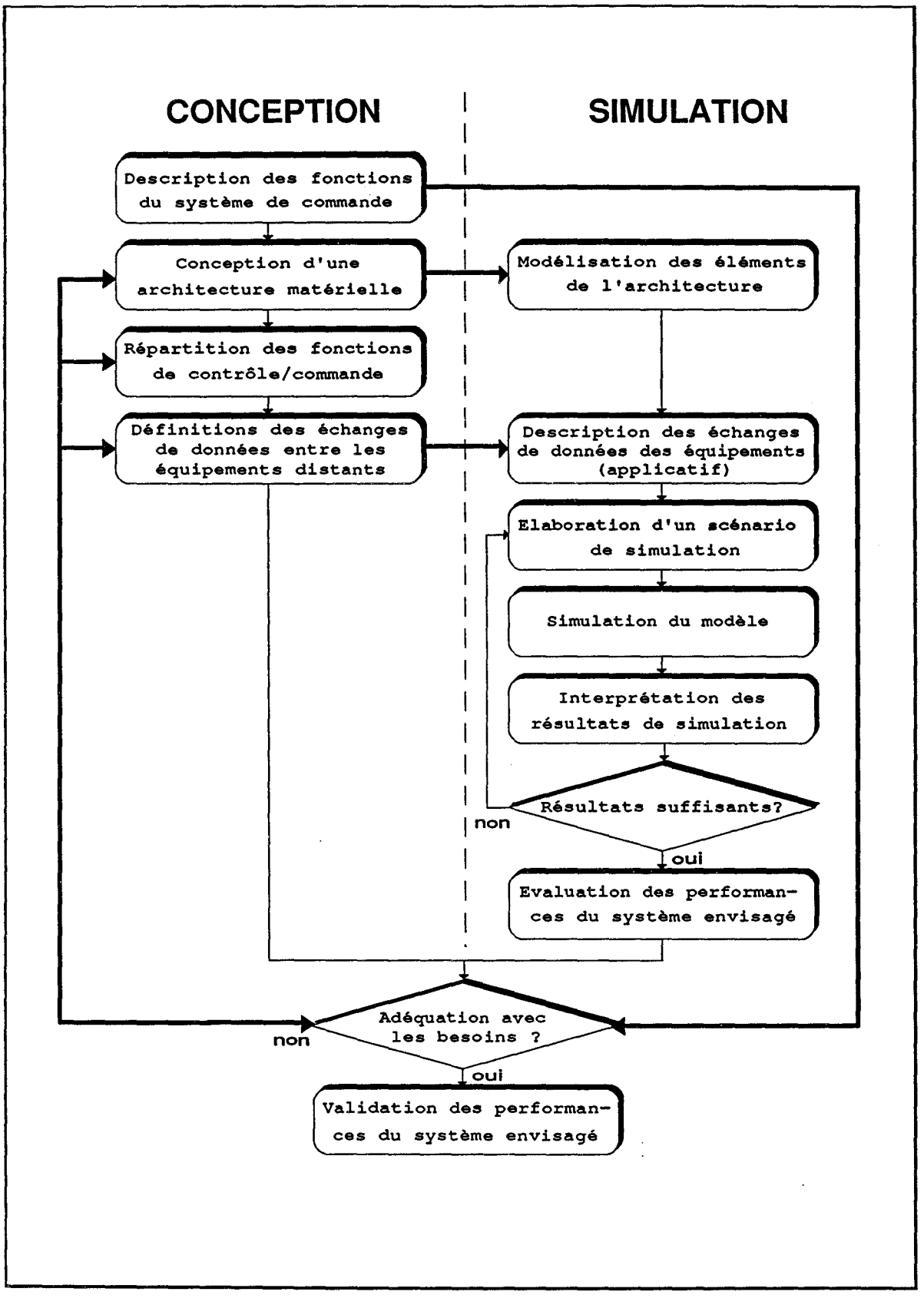


Figure I.14 : *démarche de conception d'une architecture de commande avec simulation*

La modélisation d'une architecture de commande commence par la modélisation de ses équipements. On retient essentiellement les caractéristiques liées au matériel utilisé qui ont une influence sur les performances : système d'exploitation pour un automate programmable ou un calculateur, protocole de communication pour un réseau local industriel, etc. Afin de disposer d'objets réutilisables, chaque équipement est modélisé séparément; on obtient ainsi un modèle d'automate programmable de type A, un modèle de coupleur de communication de type B etc.

Le modèle de l'architecture est alors obtenu par assemblage d'objets. Il est complété par une description des fonctions de commande implantées dans les équipements. Comme nous nous intéressons aux performances des échanges de données, il est nécessaire pour chaque fonction de décrire le séquencement des échanges avec les équipements distants; c'est ce que nous appelons "l'applicatif" de l'équipement.

Des scénarios sont ensuite définis pour simuler, avec le modèle de la solution envisagée le fonctionnement réel de l'installation. Ils ont pour objectif de représenter les événements de l'environnement qui influent sur le fonctionnement du système de commande : évolution de l'avancement du process, action opérateur etc. Le modèle complet peut alors être simulé, et les résultats interprétés. Si on juge que les informations obtenues sont suffisantes pour conclure sur les performances du système, l'étude est achevée. Dans le cas contraire, on fait d'autres simulations avec d'autres configurations de fonctionnement.

III.3.2. Positionnement dans la carte des activités de développement d'un SAP

Afin de bien situer à quels moments de l'étude et de la conception une telle approche doit être mise en oeuvre, nous allons nous appuyer sur la "carte des activités de développement d'un S.A.P." proposé par le C.C.G.A. (Centre Coopératif de Génie Automatique), qui reprend le traditionnel cycle de développement en "V" du génie logiciel pour l'adapter aux automatismes [VER 91].

Avec des niveaux de détails différents, la modélisation peut être mise en oeuvre dans trois phases distinctes :

1 - **suite à la réponse à un appel d'offres** : les différents partenaires contactés exposent chacun leur solution. Pour appuyer leur proposition, une démarche de modélisation/simulation leur permettrait de fournir des informations quantitatives

sur la solution qu'ils défendent. Les modèles seraient alors développés par des responsables du projet chez les intégrateurs ou les sociétés de service.

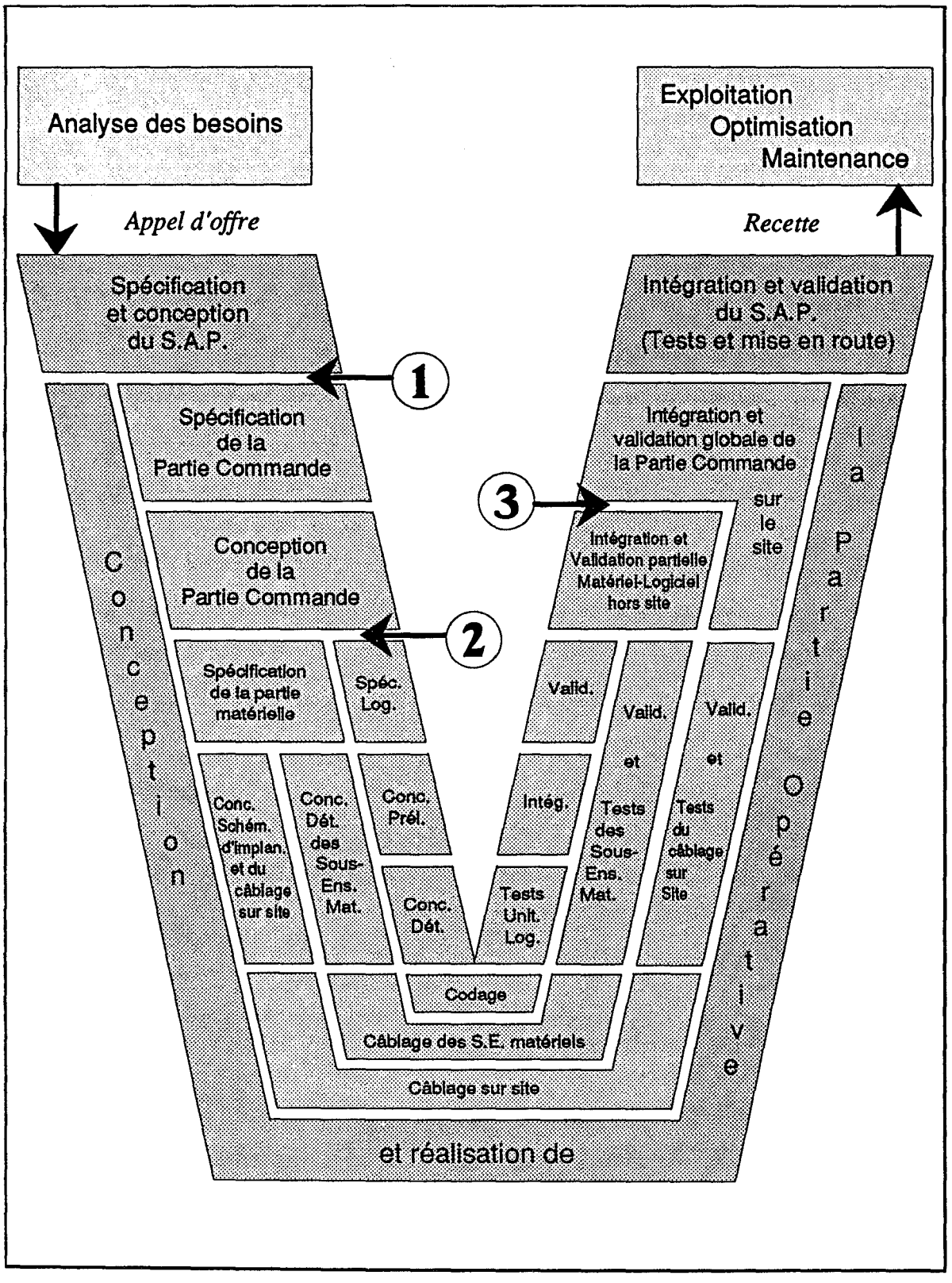


Figure I.15 : *intégration de la simulation dans le cycle de développement d'un SAP*

2 - **fin de phase de spécifications logicielle et matérielle** : à ce moment de l'étude, une architecture matérielle est proposée et l'implantation des fonctions de commande et de conduite sur les différents équipements est définie, avec les flux d'informations qui circulent sur les réseaux. Toutes les données nécessaires sont donc clairement fixées pour être en mesure de modéliser et de simuler la solution complète telle qu'elle est envisagée. Le chef de projet possède une vision d'ensemble sur la solution et est en mesure de l'évaluer par simulation.

3 - **durant la phase d'intégration et de validation hors site** : les développements et les mises au point sur plate-forme sont toujours réalisés avec un nombre d'équipements limités. Les tests unitaires ne permettent donc pas d'avoir une image réaliste de l'architecture dans son ensemble. L'intérêt de la simulation est alors d'extrapoler, à partir des informations de performances mesurées en plate-forme, les performances de l'architecture complète. Si des problèmes sont mis en évidence, des modifications dans l'organisation des échanges notamment pourront encore être réalisées, avant d'installer les équipements dans l'atelier. A cette étape d'avancement du projet, ce sont les développeurs des applications qui sont le mieux à même de mettre en oeuvre la démarche de simulation.

III.3.3 Conclusion

L'approche de modélisation/simulation, intégrée dans la démarche de conception d'architecture de commande et de pilotage d'atelier, est totalement nouvelle dans l'industrie de production manufacturière. Elle doit permettre de valider, avant implantation sur site, les performances et l'organisation des échanges de données du système étudié. Selon le degré d'avancement dans la conception de l'architecture, les informations disponibles pour élaborer le modèle seront plus ou moins précises. Il est important de garder constamment à l'esprit que les résultats obtenus par simulation dépendent des hypothèses retenues pour construire les modèles.

CONCLUSION

Après avoir exposé les caractéristiques des architectures de commande et de pilotage pour les systèmes automatisés de production installés chez PSA, nous avons mis l'accent sur les aspects relevant de l'évaluation des performances de tels systèmes. Il s'agit en effet d'une caractéristique importante qui nous semble à l'heure actuelle encore mal maîtrisée. C'est pourquoi nous avons choisi d'orienter nos travaux sur ce thème.

La simulation apparaît comme une méthode efficace permettant d'assister le dimensionnement de systèmes complexes. Elle est utilisée avec succès, notamment dans le domaine des flux de production. Nous proposons dans ce sens de développer une approche de modélisation pour les architectures de commande et de pilotage. Les simulations doivent permettre de valider les échanges de données et d'évaluer les performances des systèmes de commande et de pilotage selon le point de vue des "communications".

Cette démarche est nouvelle pour le domaine relatif aux performances des systèmes de commande d'ateliers manufacturiers. Il nous semble donc nécessaire de commencer l'étude par une analyse des caractéristiques des éléments de l'architecture à modéliser, pour ensuite choisir et justifier le formalisme adéquat.

CHAPITRE II

Modélisation et simulation d'architecture de commande et de pilotage d'atelier

INTRODUCTION

Après avoir, dans le premier chapitre de ce mémoire, mis en évidence l'intérêt d'une approche de modélisation/simulation lors de la conception d'architecture de commande d'atelier, nous présentons les principaux concepts sous-tendus par une telle démarche, totalement originale dans l'industrie de production manufacturière. Il est en effet indispensable de connaître les éléments caractéristiques d'un système de commande pour être en mesure de le modéliser de manière efficace.

Ce chapitre comporte quatre volets :

- Dans une première partie, nous justifions le choix d'un formalisme réseaux de Petri à structures de données pour l'étude de nos systèmes. Dans le cadre cette thèse, nous avons utilisé l'outil SEDRIC, qui implémente une variante de ce formalisme, pour construire et simuler les modèles détaillés dans ce mémoire.

- La deuxième partie expose les principes retenus pour modéliser de façon modulaire une architecture de commande et de pilotage. La modélisation des équipements d'automatisme est abordée au travers d'un graphe de processus communicant, qui permet de représenter le comportement interne de chaque matériel ainsi que les échanges de données entre équipements et avec l'environnement.

- Cette approche est appliquée dans une troisième partie à la modélisation d'équipements industriels d'automatisme. Les fonctionnements et les modèles associés d'un réseau local industriel et d'automates programmables de la société Télémécanique sont expliqués de façon détaillée. La généralisation de la démarche à d'autres matériels est également évoquée.

- Dans une dernière partie, nous décrivons ce que pourrait être, un outil industriel utilisable par les automaticiens, pour modéliser et simuler les architectures de commande qu'ils conçoivent, développent et mettent en oeuvre. Organisé autour d'une bibliothèque évolutive de modèles de composants, un tel outil impose deux niveaux d'utilisation. Une interface "modélisateur système" permet de compléter et d'enrichir la bibliothèque. Un environnement "concepteur d'architecture" autorise la modélisation, et la simulation d'architecture à partir des modèles de la bibliothèque.

I. UTILISATION DU FORMALISME RESEAUX DE PETRI

Dans le cadre de nos travaux de thèse, nous avons recherché, pour mettre en oeuvre l'approche de modélisation/simulation proposée, le formalisme qui nous paraissait le mieux adapté. Celui-ci devait impérativement être supporté par un outil car il est apparu indispensable, dès le commencement de notre étude, de valider nos idées par la construction et la simulation effectives de modèles.

I.1. Eléments d'architecture à modéliser

I.1.1. Principes généraux

Comme nous l'avons vu dans le premier chapitre de ce mémoire, une architecture de commande se compose d'équipements industriels (automates programmables, commandes numériques, commandes robots, calculateurs industriels) interconnectés par des réseaux locaux industriels.

Sur ces matériels fonctionnent des applicatifs (commande de machine, coordination au niveau d'un module, animation d'images de suivi de process etc.) qui concourent à la commande et au pilotage des moyens de production. Les consignes générées par la commande sont conditionnées par des informations en provenance du procédé ou de l'environnement (action opérateur par exemple).

Nous voyons donc qu'un modèle d'architecture se compose de trois types d'entités qu'il est important de bien dissocier :

- les modèles de fonctionnement interne des matériels,
- les descriptions des applicatifs implantés sur les équipements,
- une image réaliste de l'interaction entre la commande et son environnement.

Notre objectif est de simuler le comportement des flux de données échangées entre les équipements, au niveau de l'architecture de commande. Dans la représentation associée à chaque entité il ne faut donc retenir que les éléments qui ont une réelle incidence sur le déroulement de ces échanges. Il ne s'agit pas de modéliser complètement le fonctionnement spécifique d'un équipement mais d'en retenir les caractéristiques essentielles qui influent sur les performances des communications avec les matériels distants.

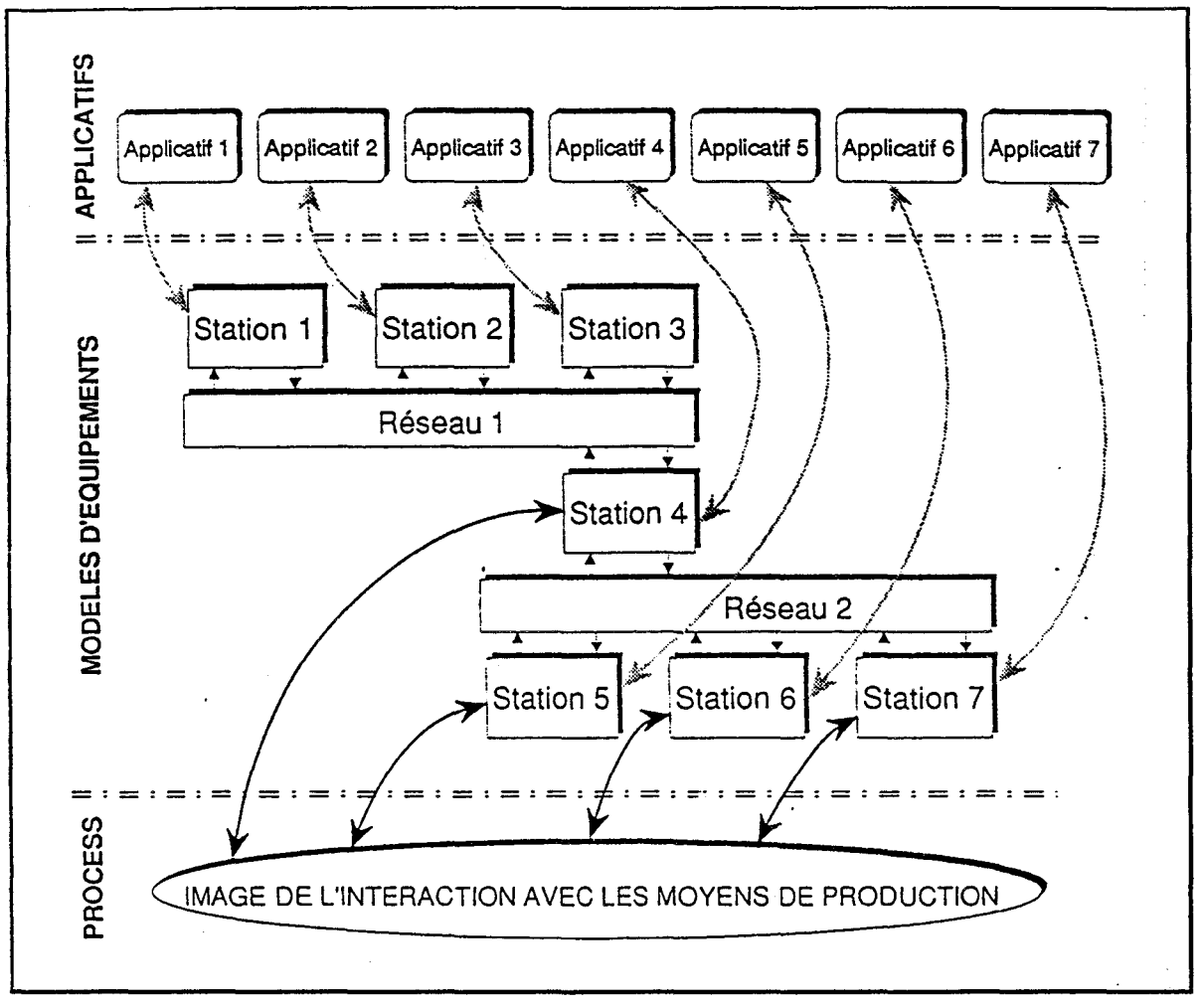


Figure II.1 : *Principes de modélisation d'une architecture de commande et de pilotage*

I.1.2. Caractères spécifiques à la modélisation des matériels d'automatisme

Pour les différents types d'équipements (automates programmables, calculateurs industriels etc.), un système d'exploitation, spécifique pour chaque marque et éventuellement type de matériel, fait office d'interface entre les programmes développés par l'utilisateur (ou éventuellement les logiciels ou progiciels utilisés et/ou adaptés) et la partie matérielle. Les performances de l'équipement, notamment en matière de communication, sont sensiblement dépendantes du fonctionnement interne de cette interface.

Chaque matériel se connecte sur un ou plusieurs réseaux par l'intermédiaire de carte ou de coupleur de communication. Dans certains cas, les échanges sont directement gérés par du code implémenté en ROM sur le périphérique (coupleurs de communication pour automates programmables par exemple). L'applicatif implanté sur l'équipement utilise alors, au travers du système d'exploitation, les services offerts par la carte ou le coupleur.

Dans d'autres cas, il est nécessaire de faire fonctionner un programme spécifique sur la station pour gérer le protocole imposé par le réseau. La tâche associée consomme alors des ressources, au même titre que les applicatifs implantés sur la station.

Sur l'exemple du chapitre précédent, des calculateurs Micro-Vax de marque Digital, fonctionnant sous le système d'exploitation temps réel Vax-Eln [DIG 87], dialoguent avec des équipements d'automatisme au travers d'un réseau Uni-Telway. Cette organisation a imposé le développement d'un driver de communication pour implémenter le protocole spécifique à ce réseau sur les calculateurs [RIA 88].

Du coté des automates TSX67/87, les communications sur le réseau sont directement prises en charge par les coupleurs Uni-Telway SCM 21.6 [TEL 88a][TEL 88b]. Le déroulement du programme écrit en langage Grafcet, qui est conditionné par le fonctionnement cyclique imposé par la tâche superviseur de l'automate, fait alors appels aux services proposés par le coupleur, au moyen de blocs fonction de type TxT [TEL 90c].

Les modèles des différents équipements utilisés ne peuvent être réalisés que par un spécialiste, qui connaît de manière approfondie le fonctionnement de chacun de ces systèmes. L'objectif est de modéliser chaque matériel une seule et unique fois et de pouvoir réutiliser le modèle associé (qui sera éventuellement paramétrable) dans chaque étude où le matériel intervient.

I.1.3. Spécification de la description des applicatifs

L'assemblage des différents modèles d'équipements intégrés dans une architecture de commande forme une modélisation complète de l'aspect matériel du système. Il s'agit ensuite, au niveau de chaque équipement, de décrire le séquencement de ses échanges de données au niveau de l'architecture, les caractéristiques de ces communications étant spécifiques pour chaque nouvelle application étudiée.

Le modèle d'un équipement est construit indépendamment de l'étude de toute architecture. Lors de l'analyse de chaque nouveau système de commande, il est donc nécessaire de détailler la partie communication de la commande qui sera implantée dans l'équipement. Cette approche nécessite de pouvoir détecter la réception et le contenu de messages émis par des stations distantes vers l'équipement concerné. Il est également indispensable de disposer d'une image des événements en provenance du process à piloter qui influent sur les règles d'émission de messages sur les réseaux.

Le formalisme utilisé doit donc offrir une certaine souplesse à ce niveau de l'étude, pour permettre de décrire et de modifier rapidement le séquencement des applicatifs. De plus, la modélisation des équipements doit permettre une interaction simple entre la description des applicatifs et le fonctionnement interne du matériel.

I.1.4. Image de l'interaction procédé/commande

Une architecture de commande et de pilotage d'un système automatisé de production réagit à des événements en provenance du process à piloter ainsi qu'à des consignes transmises par un niveau supérieur (cadencement par exemple). Après acquisition et traitement de ces informations le système de commande envoie des consignes vers le process pour le faire évoluer. La détermination de ces consignes est réalisée par les programmes de contrôle/commande implantés dans les équipements de l'architecture.

Les échanges de messages sur l'architecture sont notamment liés à des événements en provenance des moyens de production à commander. Il est donc nécessaire d'avoir une image réaliste de l'interaction entre la commande et le procédé, qui puisse être mise en relation avec la description des applicatifs, et ainsi conditionner l'évolution du séquencement des communications sur chaque station.

I.1.5. Exemple d'illustration

Pour appuyer les idées qui viennent d'être développées, nous présentons un exemple industriel du groupe PSA Peugeot Citroën, qui est par ailleurs plus amplement détaillé en annexe 2 de ce mémoire. Il s'agit d'une architecture organisée autour d'un réseau Telway 7 de la Télémécanique, sur lequel sont connectés des automates TSX 87/30 [TEL 89]. La topologie des équipements est présentée sur la figure II.2.

Pour des raisons de continuité avec l'existant, l'automatisation est réalisée de manière progressive : quatre automates sont installés dans une phase A, une cinquième station est ajoutée dans une phase B et une sixième dans une phase C. Le projet complet envisage à terme de mettre en place quinze automates sur le réseau.

Les choix de conception pour définir les échanges de données entre les équipements distants sont liés à la répartition sur l'architecture matérielle des fonctions de contrôle/ commande définies durant la phase préalable de spécification fonctionnelle, et aux possibilités de communication de l'architecture.

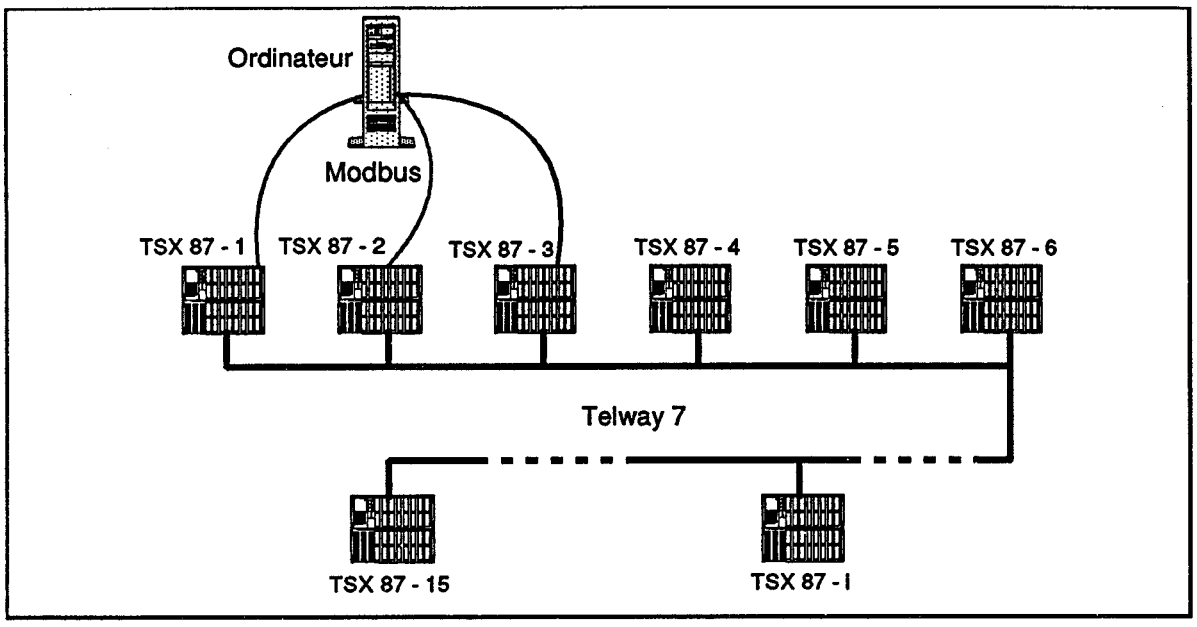


Figure II.2 : *architecture de commande étudiée*

Comme les automates 4, 5 et 6 ne sont pas directement reliés à l'ordinateur, les données qu'ils ont à échanger avec ce dernier doivent transiter par un des automates 1, 2 ou 3 qui jouent le rôle de passerelle. Les automates 1, 2 et 3 étant très chargés par les fonctions de commande qu'ils ont à réaliser, ils ne supportent aucun applicatif pour les échanges de données sur le réseau Telway 7. Ce sont donc les automates 4, 5 et 6 qui viennent lire et écrire dans ces automates.

Les échanges de données sur le réseau Telway 7 sont alors les suivants :

==> **Automates 1, 2 et 3** : aucun applicatif sur le réseau Telway 7,

==> **Automate 4** :

- télélecture de 2 mots d'état sur chaque station 1, 2 et 3 : de manière régulière,
- téléchargement des 2 mots d'état sur chaque station 1, 2 et 3 : de manière régulière,
- téléchargement des 10 mots d'alarmes sur le calculateur, via un des automates 1, 2 ou 3 : sur apparition d'une alarme,
- télélecture de tables de codes depuis le calculateur, via un des automates 1, 2 ou 3 : hors production lors du démarrage de l'installation.

==> **Automate 5** : idem automate 4 avec en plus :

- télélecture d'ordre de production (3 mots), sur chaque station 1, 2 et 3 sur demande de celui-ci : lectures régulières,

==> **Automate 6** : idem automate 4 avec en plus :

- télélecture d'ordre de production (3 mots), sur chaque automate 1, 2 et 3 sur demande de celui-ci : quelques ordres par jour,

Pour règlementer ces échanges de télélecture et téléchargement, un protocole applicatif est défini sur le réseau Telway 7. Le gestionnaire des échanges est l'automate 4. En plus de ses propres communications avec les autres stations, il désigne à tour de rôle l'automate autorisé à dialoguer sur le réseau. La réservation du bus étant ainsi faite, la station désignée peut, si elle a des échanges à effectuer, écrire ou lire des données dans les automates de son choix. Lorsqu'il a terminé ses échanges, l'automate autorisé à émettre statue la fin de ses émissions. L'automate 4 refait alors une nouvelle réservation pour une autre station.

Le projet complet prévoie que tous les automates, sauf les 1, 2 et 3 soient régulièrement interrogés par l'automate 4. L'attribution et la restitution du droit de parole est faite par l'intermédiaire des mots COM (table de mots partagée entre les différents automates du réseau [TEL 89]).

Au vu de la complexité du fonctionnement des échanges, il est extrêmement difficile d'appréhender les performances de ce système. Pourtant il est essentiel de connaître les conséquences de la mise en place du "protocole applicatif". En effet une station est autorisée à envoyer un message uniquement lorsque le droit de parole lui est attribué. Si le délai qui sépare deux autorisations successives est trop important, cela peut différer certains échanges, et éventuellement dans le pire des cas, entraîner un retard sur la production (s'il s'agit d'une demande de fabrication par exemple).

I.2. Formalismes de modélisation disponibles

I.2.1. Contexte

Notre objectif est de représenter des enchaînements temporisés d'échanges de données au sein d'une architecture de commande. Cette approche impose que nos modèles prennent en compte à la fois des éléments sur les fonctionnements internes des équipements et des descriptions d'applicatifs de commande. Le formalisme à utiliser doit donc présenter les trois caractéristiques suivantes :

1 - une ***prise en compte intégrée du parallélisme*** : une architecture de commande est un système distribué dans lequel différentes fonctions de commande et de pilotage s'exécutent simultanément sur des équipements distants. A certains stades de leur exécution, ces fonctions communiquent et se synchronisent au travers des moyens de communication disponibles sur l'architecture. Une composante essentielle d'un tel système est donc le caractère parallèle de son évolution et la

nécessaire prise en compte des caractères synchrones et asynchrones des communications.

2 - un ***aspect dynamique pour modéliser des changements d'états*** : dans le domaine du manufacturier et en particulier dans l'industrie automobile, un système de contrôle/commande peut être étudié comme un système discret. Le fonctionnement de chaque équipement de commande et de pilotage est alors assimilé à une succession d'états et de changements d'états. Ceux-ci sont conditionnés par la prise en compte d'événements en provenance de l'environnement (des moyens de production par exemple) ou générés par le système de commande lui-même (message au niveau applicatif, signal provoqué par le fonctionnement interne d'un équipement etc.)

3 - une ***prise en compte effective du temps*** : pour évaluer a priori les performances du système, il est nécessaire d'intégrer un aspect temporel dans les modèles. Des durées constantes ou stochastiques doivent pouvoir être associées à certains des états et à certaines des transitions entre états modélisés. Ces délais sont alors pris en compte lors de l'exploitation du modèle par simulation.

De manière générale, les environnements susceptibles a priori d'être employés pour construire et exploiter des modèles d'architectures, peuvent être positionnés selon quatre points de vue, comme l'illustre la figure II.3.

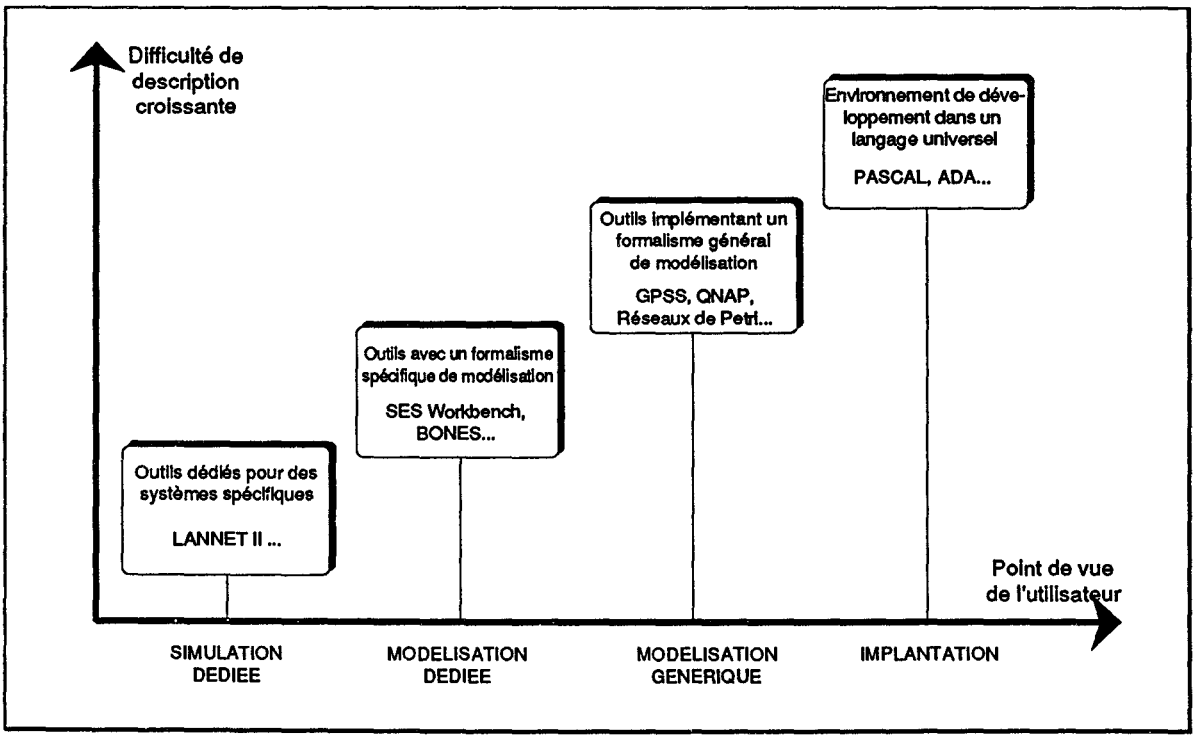


Figure II.3 : *environnements utilisables pour simuler des modèles*

D'emblée, il nous est apparu qu'adopter un formalisme graphique était souhaitable C'est pourquoi des langages de simulation à usage général tels que GPSS, SIMSCRIPT ou SLAM [CER 88] n'ont pas été envisagés. En effet une représentation sous forme de "graphes" présente à notre sens les avantages suivants pour nos travaux :

- capacité pour modéliser des comportements complexes et notamment pour prendre en compte le caractère parallèle de système,
- rapidité pour la mise au point des modèles grâce aux facilités de compréhension liées à l'aspect graphique,
- efficacité dans le cadre de travail en équipes avec des personnes habituées à des représentations graphiques (Grafcet).

Avant de présenter l'orientation retenue, nous justifions rapidement pourquoi les autres possibilités mentionnées sur le figure II.3 ont été écartées.

I.2.2. Outils dédiés

Un logiciel comme LANNET II.5 [CAC 90] de la société CACI permet l'étude par la simulation de réseaux de communication basés sur les protocoles Ethernet et Token-Ring. Les modèles font partie intégrante du logiciel, ce qui les rend accessibles, au travers d'une interface graphique conviviale, à des personnes qui n'ont pas besoin de connaître le fonctionnement interne du système qu'elles simulent. C'est un bon exemple d'outil dédié à l'étude de systèmes spécifiques. L'intérêt est de pouvoir tester rapidement différentes configurations de fonctionnement de ces réseaux pour obtenir notamment des informations sur la charge du système. Aucun outil de ce type n'existe actuellement pour l'étude des architectures de commande et le pilotage d'atelier de production.

Des outils plus généraux, comme SES Workbench [SES 91] ou BONES [BUR 89][SHA 90], proposent des formalismes spécifiques principalement dédiés à la modélisation de systèmes électroniques ou informatiques. Ils se sont d'abord développés sur le marché américain, où les approches de simulation semblent plus développées qu'en Europe, avant d'être distribués depuis peu de temps sur le marché français. L'approche paraît intéressante pour la modélisation du fonctionnement interne des équipements d'automatisme et des protocoles réseaux.

Cependant dans les exemples d'utilisation de ces outils, que nous avons eu l'occasion de consulter, les scénarios de charge des modèles sont représentés par des lois probabilistes. Dans le cadre de notre étude, il est indispensable (Cf. I.1) de pouvoir

modéliser de façon précise l'interaction entre le modèle et son environnement. Ces outils étant apparus bien après le début de nos travaux, il ne nous a pas été donné l'occasion de regarder en détails si les formalismes proposés pouvaient être employés dans le cadre de notre projet.

I.2.3. Langages de programmation

Une autre possibilité pour construire et exploiter des modèles, à l'extrême inverse des outils dédiés, est d'utiliser un langage universel de programmation. Comme les modèles que nous voulons élaborer possèdent une forte composante parallèle, nous avons regardé si des langages qui intègrent le parallélisme tels qu'ADA [BAR 88] pouvaient être employés. En effet, une idée qui semble séduisante de prime abord, consiste à exploiter le caractère parallèle de ce langage pour reproduire le fonctionnement parallèle du système à représenter.

ADA intègre la notion de parallélisme par l'intermédiaire des *tâches*. Une tâche est constituée d'une spécification qui décrit l'interface présentée aux autres tâches et d'un corps qui traduit son comportement dynamique. Celui-ci dont l'exécution est séquentiel, permettrait de spécifier le séquencement de chaque fonction de l'architecture modélisée.

Un mécanisme appelé *rendez-vous,* implémenté par les primitives ADA "entry", "accept" et "select" permet aux tâches d'interagir entre elles, en se synchronisant et en échangeant des données. Par l'utilisation de ces concepts il est possible de modéliser les échanges de messages et les synchronisations entre équipements distants au sein de l'architecture.

```
task T1 is                          task T2 is
....                                  Entry E2 (d2 : in INTEGER);
end T1;                             end T2;

task body T1 is                     task body T2 is
  d1 : INTEGER;                     begin
begin                                 accept E2 (d2 : in INTEGER) do
  { traitement A1}                       { traitement A2 }
  T2.E2 (d1);                         end E2;
  { traitement B1}                    { traitement B2 }
end T1;                             end T2;
```

Figure II.4 : *squelette du code de deux tâches ADA synchronisées par rendez-vous*

Le temps est représenté de façon explicite en ADA par l'instruction "**delay** temps". Celle-ci a pour effet de suspendre la tâche exécutant l'instruction, pendant une durée en secondes au moins égale à la valeur du paramètre "temps". Nous devons dire au moins, car après l'expiration de l'intervalle, le processeur n'est pas nécessairement immédiatement disponible pour poursuivre l'exécution de la tâche (il peut être utilisé par une autre tâche qui s'exécute en parallèle).

La manière dont le temps est pris en compte dans un programme ADA n'est pas à notre sens satisfaisante pour imaginer de modéliser des systèmes évoluant en parallèle avec des tâches ADA. D'une part, spécifier une action dont la durée est de 100 ms avec l'instruction delay 0.1 signifie que lors de la simulation du système, qui correspond en fait à l'exécution du programme ADA, la tâche attend effectivement au moins 100 ms. Cela signifie que si l'ensemble des temporisations rencontrées par une tâche correspondent à une durée cumulée de 100 heures, l'étude nécessitera au moins ce délai! D'autre part, l'imprécision sur la durée effective des temporisations lors d'une exécution du code rend nécessairement les résultats de l'étude imprécis, et n'offre aucune garantie sur la conservation des caractéristiques synchrones du modèle.

Pour utiliser de manière efficace un langage comme ADA pour réaliser des simulations, il est indispensable de contrôler à l'intérieur du modèle l'avancement du temps. C'est ce qui est fait par des outils qui proposent des formalismes généraux de modélisation, comme ceux détaillés dans le paragraphe suivant.

Remarque :

Le projet ELECTRE (Exécutif et LangagE de Contrôle Temps-réel REparti) propose un langage réactif asynchrone et son exécutif associé destinés à spécifier les comportements dans un système temps réel [ELL 86]. Une application est décrite en Electre comme un ensemble d'activités modulaires séquentielles, parallèles ou exclusives dont les instants d'activation et de préemption sont conditionnés par des occurrences d'événements émis par d'autres activités ou par l'environnement. Cette approche correspond bien à la philosophie de nos travaux mais le langage Electre ne permet pas de décrire le contenu procédural des différents modules et ne prend pas en compte de façon explicite le temps. C'est pourquoi ce formalisme n'a pas été retenu dans le cadre de cette étude.

I.2.4. Formalismes généraux de modélisation

Il existe deux formalismes graphiques principaux pour modéliser et évaluer les performances des systèmes discrets : le formalisme des files d'attente et celui des réseaux de Petri .

I.2.4.a Réseaux de files d'attente [GEL 82]

La modélisation par files d'attente repose sur deux notions principales :

1 - la station qui se compose d'une file d'attente et d'un ou plusieurs serveurs.

2 - le client qui chemine d'une station à l'autre.

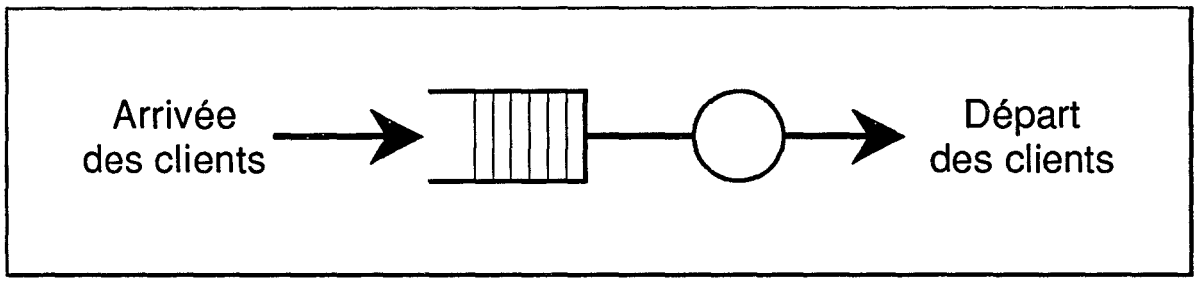


Figure II.5 : *représentation graphique d'une station à un serveur*

Une station est définie par : - le processus d'arrivée des clients à la station,
- la discipline de service
- le nombre de serveurs
- le processus de service,
- la capacité de la file d'attente.

Un client arrive dans une station suivant un certain processus d'arrivée. Si un guichet de service est libre, il y entre et est servi selon le processus de service associé. S'il n'est pas le seul dans la file d'attente à son arrivée, il lui est appliqué une discipline de service (LIFO, FIFO etc.). La capacité d'une file d'attente n'est pas forcément illimitée. Si sa capacité est limitée à k clients, et si elle est pleine il ne peut plus y avoir d'entrée de nouveau client.

Le formalisme permet de distinguer plusieurs classes de clients. Dans ce cas, il faut définir le processus d'arriver et le processus de service pour chaque classe, ainsi que la discipline de service entre classes. Pour modéliser des systèmes complexes, il faut combiner plusieurs files d'attente et ainsi former un réseau de files d'attente.

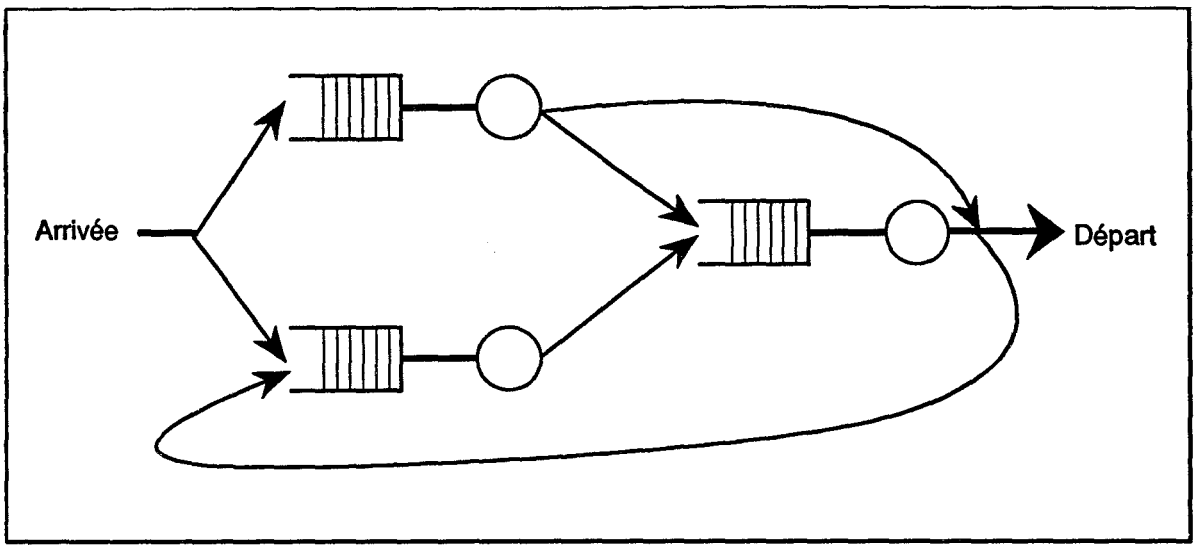


Figure II.6 : *exemple de réseau à files d'attente*

La société Simulog commercialise l'outil QNAP2 (Queing Network Analysis Package) qui permet de décrire et d'analyser des réseaux de files d'attente. Elaboré autour du langage QNAP, il permet de réaliser des simulations ou des résolutions analytiques de modèles. Un langage de type algorithmique permet une prise en compte détaillée de différents services autorisant ainsi un niveau de description plus fin pour les différents services.

I.2.4.b Réseaux de Petri [BRA 83][DAV 89]

Un modèle réseaux de Petri (RdP) classique se représente de façon totalement graphique (Cf. annexe 1) [VID 86]. Il est constitué de *places* symbolisées par des cercles et de *transitions* matérialisées par des rectangles (ou des segments). Des *arcs valués et orientés* relient les places aux transitions et les transitions aux places.

Chaque place du modèle RdP est susceptible de contenir des *jetons*. le nombre de jetons présents dans une place est appelé *marquage* de cette place. L'ensemble des marquages de toutes les places d'un modèle RdP constitue le marquage de ce réseau.

Le caractère dynamique du système se traduit par des règles d'évolution du marquage d'un RdP, qui permettent de passer d'un marquage à un autre. Un changement de marquage correspond au franchissement d'une transition du modèle :

- une transition est franchissable si et seulement si chacune des places amont de cette transition contient un nombre de jetons supérieur ou égal au poids de l'arc qui relie la place à cette transition.

- lors du franchissement d'une transition :

  1 - on retire de chacune des places amont un nombre de jetons égal au poids de l'arc reliant la place à cette transition

  2 - on ajoute dans chacune des places aval un nombre de jetons égal au poids de l'arc reliant la transition à la place.

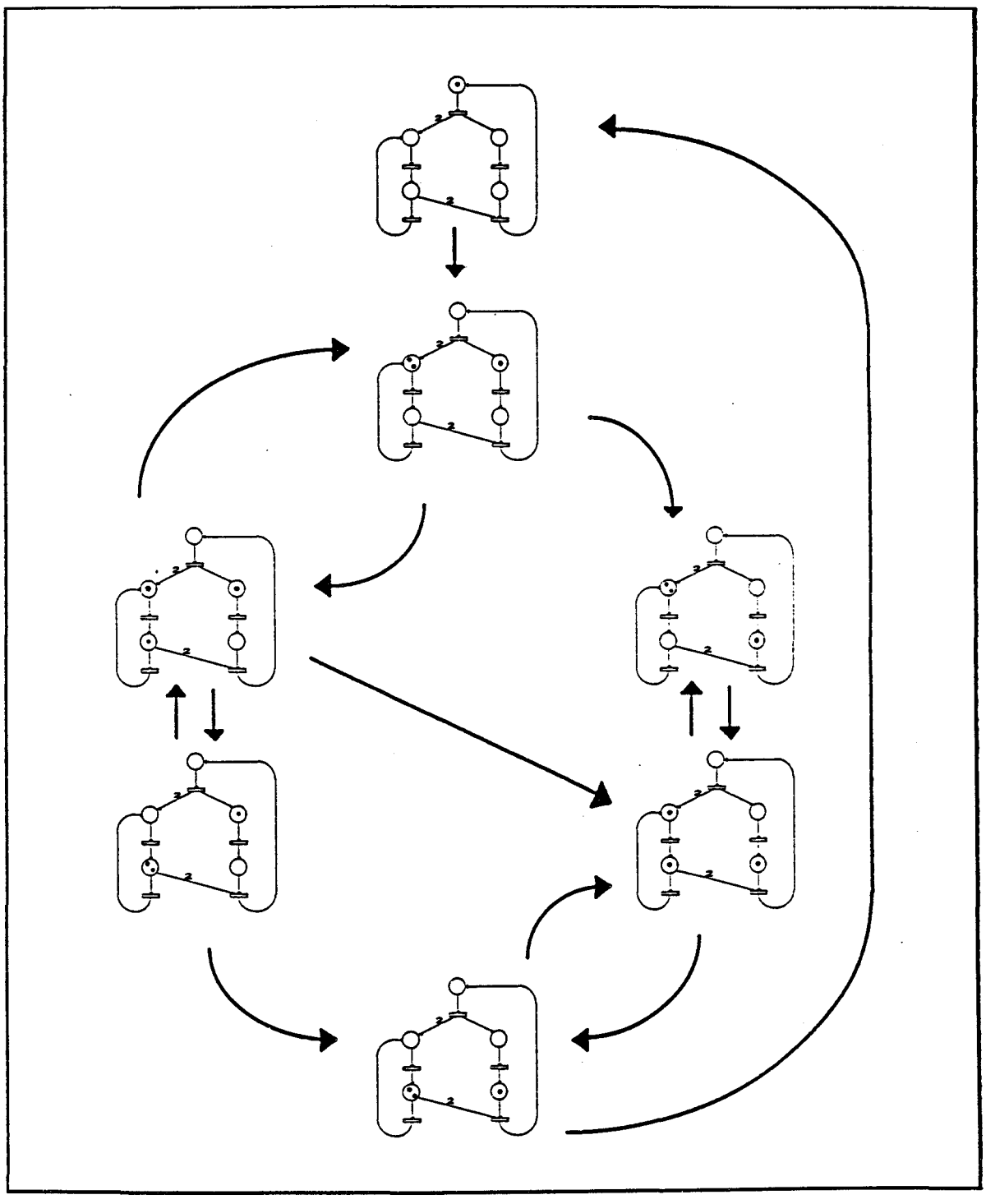


Figure II.7 : *possibilités d'évolution d'un modèle RdP*

I.2.4.c Conclusion

Les deux formalismes qui viennent d'être détaillés suscitent depuis plusieurs dizaines d'années de nombreux travaux de recherche, dans différents laboratoires en France et dans le monde. Les principaux sujets d'étude concernent les extensions des possibilités de modélisation, la recherche de nouvelle méthode de résolution ou d'exploitation ainsi que l'utilisation de ces formalismes dans des domaines d'applications variés.

I.3. Choix d'un formalisme réseaux de Petri

I.3.1. Travaux existants

Dans de nombreuses publications qu'il nous a été donné de consulter [PUJ 86], il est souvent admis que les réseaux de Petri sont plutôt utilisés pour traiter des problèmes d'analyse qualitative et les réseaux à files d'attente pour réaliser des études quantitatives.

Dans le domaine de l'évaluation de performances de systèmes informatiques, nous pouvons notamment faire référence aux travaux de recherche menés au sein du laboratoire CNRS-MASI (Méthodologie et Architecture des Systèmes Informatiques) de l'Université Pierre et Marie Curie (Paris VI) [FDI 89]. L'équipe Réseaux et Performance effectue depuis plusieurs années un transfert de technologie entre les recherches amont, dans le domaine de la modélisation des systèmes informatiques et de communication et ses applications industrielles [COR 91].

Fondés sur une base théorique éprouvée [REU 89], les réseaux de Petri permettent de réaliser des analyses formelles pour prouver des propriétés sur des systèmes modélisés avec ce formalisme. L'analyse formelle se distingue des techniques de simulation traditionnelles par son caractère exhaustif. Cette démarche a, par exemple, été mise en oeuvre lors de la conception de protocoles de communication pour valider des modes de fonctionnement [AYA 85].

Depuis quelques temps, les réseaux de Petri, auxquels ont été ajoutées ces dernières années un certain nombre d'extensions, ont également été utilisés avec succès pour faire de l'évaluation de performances de systèmes [MAR 87][JUA 90][REZ 90]. C'est pourquoi il nous a paru intéressant de regarder de quelle manière ce formalisme pouvait être utilisé dans le cadre de nos travaux. Cette idée s'est trouvée confortée par l'apparition, depuis quelques années, de plusieurs outils informatiques qui implémentent différents formalismes de RdP, et permettent ainsi la construction et l'exploitation de modèles.

I.3.2. Extensions du formalisme réseaux de Petri

Les réseaux de Petri, tels qu'ils avaient été définis par C.A. Petri en 1966, sont particulièrement bien adaptés pour la représentation des systèmes intégrant des caractéristiques de parallélisme, concurrence et synchronisation. C'est d'ailleurs dans cette optique qu'ils avaient initialement été imaginés.

I.3.2.a Représentation associée au jeton

Cependant, ce formalisme de base reste limité notamment à cause de l'impossibilité de distinguer les jetons. Ceci a pour conséquence de limiter le pouvoir d'expression et d'entraîner la construction de modèles de taille importante, souvent lourds à manipuler. C'est pourquoi des extensions ont été définies dans l'objectif d'individualiser les jetons en les rendant porteur d'informations, sur lesquelles il est possibles de réaliser des sélections et des actions.

Réseaux de Petri colorés [JEN 86]

Dans un réseau de Petri coloré, des couleurs sont associées aux jetons, ce qui permet de les différencier. A chaque place est associé un ensemble de couleurs qui correspondent aux classes de jetons susceptibles de marquer la place. Les règles de base d'évolution du marquage sont complétées par des couleurs de franchissement associées aux transitions et des fonctions associées aux arcs qui traduisent les relations qui existent sur les couleurs de sélection des jetons amont et de création des jetons aval.

Réseaux de Petri à Prédicat [GEN 87]

Les réseaux de Petri à Prédicat constituent une généralisation des réseaux de Petri colorés. Ce formalisme permet d'associer à un jeton, non plus une mais plusieurs couleurs. Une marque devient donc susceptible de représenter un n-uplet de valeurs dont les domaines sont des ensembles de constantes.

Réseaux de Petri à Structure de Données [SIB 85]

Un réseau de Petri à Structure de Données est un réseau dans lequel les jetons sont remplacés par des instances de classe de données structurées et typées (analogues aux RECORD dans le langage Pascal). Les règles de base d'évolution

d'un modèle sont complétées par des préconditions et des actions associées à chaque transition. Une précondition permet de spécifier des conditions booléennes sur les données associées aux jetons amont pour autoriser le franchissement. Une action permet de modifier les valeurs véhiculées par les jetons en sortie de transition.

Réseaux de Petri à Objets [SIB 87]

Les réseaux de Petri à objets sont une généralisation des réseaux à Structures de Données. Les classes de structures de données sont remplacées par des classes d'objets composés d'une liste d'attributs (propriétés) et de méthodes (opérations). Cette approche est à rapprocher des extensions objets apportées au langage Pascal, telles qu'elles ont par exemple été implémentées sur le compilateur Turbo-Pascal à partir de la version 5.5 [BOR 89]. Les méthodes associées aux jetons peuvent être utilisées au niveau des préconditions et actions sur les transitions. Il est possible de définir une classe par héritage d'attributs et de méthodes d'une classe pré-existante.

I.3.2.b Prise en compte du temps

Pour réaliser des évaluations de performances, il est nécessaire d'associer un aspect temporel à ces modèles. Il existe deux méthodes pour prendre en compte des délais avec le formalisme des réseaux de Petri. Des temporisations peuvent être associées aux places (on parle alors de RdP P-temporisés) ou aux transitions (on dira alors qu'un RdP est T-temporisés) [BRA 83].

Dans un RdP temporisé, une durée fixe est associée à chaque place ou transition. Cependant de nombreux processus nécessitent d'être modélisés par une variable aléatoire. On peut alors utiliser les RdP stochastiques [NAT 80] qui permettent d'associer un temps aléatoire au franchissement d'une transition.

I.3.3. Outils axés sur les réseaux de Petri

Le besoin d'outils informatiques permettant de construire et d'exploiter des modèles réseaux de Petri est ressenti depuis de nombreuses années [HOL 85]. Le premier logiciel de ce type a été le produit OVIDE commercialisé par SYSECA Temps Réel et réalisé en collaboration avec le LAAS. Actuellement il existe plusieurs réalisations qui fonctionnent sur station de travail. Nous présentons brièvement trois logiciels qui nous paraissent parmi les plus performants.

I.3.3.a Outil RdP de Verilog [VER 89b]

Cet outil permet la construction et l'analyse de réseaux de Petri classiques, temporels ou stochastiques. La construction d'un modèle est réalisée à partir d'une décomposition hiérarchique. Les modèles sont exploités à l'aide d'analyseurs (un par type de formalisme) qui calculent le graphe des marquages accessibles représentant le fonctionnement dynamique du système modélisé. A partir de ce graphe il est possible de vérifier des propriétés associées au modèle.

Les formalismes proposés par cet outil, ne permettent pas d'associer des informations aux jetons, ce qui restreint les possibilités d'expression. L'analyse réalisée par construction du graphe de couverture limite la taille et la complexité des modèles à cause de l'ampleur que peut rapidement prendre un tel graphe. Cet outil ne nous paraît donc pas adapté à nos travaux.

I.3.3.b Outil ELSIR de IBSI Electronique [REZ 90]

ELSIR est un outil pour spécifier, modéliser et évaluer les performances de systèmes répartis. Il a été développé au sein de la société IBSI Electronique, en coopération avec le Centre de Programmation de la Marine (CPM). Il intègre une méthode de décomposition hiérarchique de type SADT [IGL 89] : un système est vu comme la composition d'un ensemble de sous-systèmes. Pour détailler le fonctionnement temporel et logique des noeuds terminaux de cette décomposition, ELSIR propose un formalisme fondé sur des réseaux de Petri étendus : réseaux à prédicats avec typage et valuation des jetons et temporisations et actions associées aux transitions. L'évaluation de performances est réalisée par simulation du modèle.

I.3.3.c Outil SEDRIC de Techlog [VAL 85][TEC 89]

SEDRIC est un outil graphique, qui fonctionne sur station de travail, pour construire et simuler des modèles réseaux de Petri. Il a été conçu au L.A.A.S (Laboratoire d'Automatisme et d'Analyse des Systèmes) à Toulouse, et a été commercialisé par la société Techlog.

Le formalisme utilisé par ce produit est un formalisme réseaux de Petri interprétés, colorés et temporisés sur les places. A chaque jeton est associé une couleur, ainsi que dix paramètres réels. Pour résoudre des conflits, préciser des règles de décision au niveau d'un changement d'état, ou décrire des modifications subies par des entités modélisées par des jetons, des procédures peuvent être associées aux arcs. Extraites de la bibliothèque standard livrée avec SEDRIC, ou écrites par l'utilisateur en langage Pascal [LEB 80], elles sont de deux types :

conditions : pour introduire une condition supplémentaire, liée aux paramètres ou aux couleurs des jetons amont, pour autoriser ou interdire le franchissement d'une transition tirable au sens du marquage.

actions : activées lorsque la transition correspondante est effectivement franchie, elles autorisent des modifications sur les paramètres et les couleur des jetons franchissant la transition.

Le formalisme proposé est proche de celui des réseaux de Petri à structures de données, avec cependant la différence importante d'implémenter l'interprétation associée au formalisme sur les arcs et non sur les transitions. L'exploitation du modèle est réalisée par simulation.

I.3.4. Conclusion

Le formalisme réseaux de Petri proposé par l'outil SEDRIC nous a paru intéressant dans le cadre de nos travaux de modélisation d'architecture de commande. En effet, les données échangées sur l'architectures peuvent être représentées par les jetons. Les couleurs permettent de distinguer les classes de données et les paramètres de les décrire plus finement (émetteur, destinataire, taille...). Les traitements réalisés sur ces données sont matérialisés par des graphes réseaux de Petri. L'intérêt de l'interprétation avec des procédures écrites en langage Pascal est de permettre d'aller très précisément dans la description des traitements, grâce aux possibilités d'expression du langage.

Dans le paragraphe I.2. nous proposions d'utiliser un langage tel qu'ADA pour construire nos modèles. Le caractère textuel du langage autorise une grande précision dans la description des traitements, mais ne permet pas de matérialiser de manière simple des enchaînements d'états. Inversement, le formalisme des réseaux de files d'attente est mieux adapté sur ce point mais paraît plus limité quant à la finesse de description des traitements. Le formalisme de l'outil SEDRIC constitue une alternative intéressante en permettant de décrire de manière graphique des successions d'états et des flux de données et de manière textuelle des traitements. Toute la difficulté est alors de trouver le bon équilibre entre ce qui est modélisé avec l'aspect graphique des réseaux de Petri et ce qui est pris en compte par l'interprétation.

II. STRUCTURATION DU MODELE D'ARCHITECTURE

Le formalisme réseaux de Petri qui vient d'être présenté peut être considéré comme un langage graphique pour décrire et simuler le fonctionnement de systèmes qui intègrent une composante parallèle. Pour aborder la modélisation d'architecture dans les meilleures conditions, il est important de définir des règles d'utilisation de ce formalisme afin de structurer les modèles [BOU 91]. En effet, avec la capacité d'expression disponible avec les réseaux de Petri étendus, un même comportement peut se représenter de diverses manières.

II.1. Organisation du modèle d'architecture

Dans la paragraphe I.1, nous avons isolé trois types d'entités à modéliser pour décrire une architecture de commande et de pilotage d'atelier : le fonctionnement interne des équipements, la description des applicatifs de contrôle/commande et l'interaction entre l'architecture et son environnement. Nous allons donc détailler comment chacun de ces éléments est pris en compte avec le formalisme des réseaux de Petri de l'outil SEDRIC.

II.1.1. Modélisation des équipements

Un modèle est une représentation de la réalité soumise à des simplifications et obéissant à des hypothèses précises. Aborder la modélisation d'un composant suppose donc de connaître à un niveau d'abstraction donné ses modes de fonctionnement. Ce sont ces informations qui sont à intégrer de manière partielle ou totale dans le modèle que l'on veut élaborer. En effet, modéliser un système c'est réaliser un compromis entre précision et complexité.

Dans un objectif de modularité, il est nécessaire de construire un modèle pour chaque matériel. Le modèle peut être paramétrable pour tenir compte des différents modes de fonctionnement ou éventuellement des variations autour d'une même gamme de produits. C'est par exemple le cas d'un calculateur avec différentes fréquences de CPU, ou d'un coupleur de communication pour un même protocole, implémenté dans des coupleurs dédiés à différents types d'automates programmables.

Les modèles associés à chaque composant doivent pouvoir s'interconnecter pour former le modèle d'un équipement complet et ensuite d'une architecture. Ainsi pour

représenter un automate équipé d'un coupleur de communication réseau, on considérera le modèle du système d'exploitation de l'automate couplé avec celui du protocole réseau implémenté dans le coupleur.

Nous avons choisi d'aborder la modélisation des équipements de commande et de pilotage en les assimilant à des systèmes discrets : nous supposons que la description de leur fonctionnement interne peut se réduire à une succession d'états et de transitions entre états. Par exemple, pour un coupleur de communication qui implémente un protocole de type maître, on peut décrire les états suivants :

- interrogation de la première station,
- attente de la réponse de la première station,
- traitement de la réponse de la première station, etc.

De la même manière, sur un calculateur multitâche qui utilise du "swapping" avec une mémoire de masse, un process peut se trouver dans les états suivants [IBM 91] :

- en cours d'exécution,
- en attente et présent en mémoire,
- en attente et transféré sur disque,
- éligible et chargé en mémoire, etc.

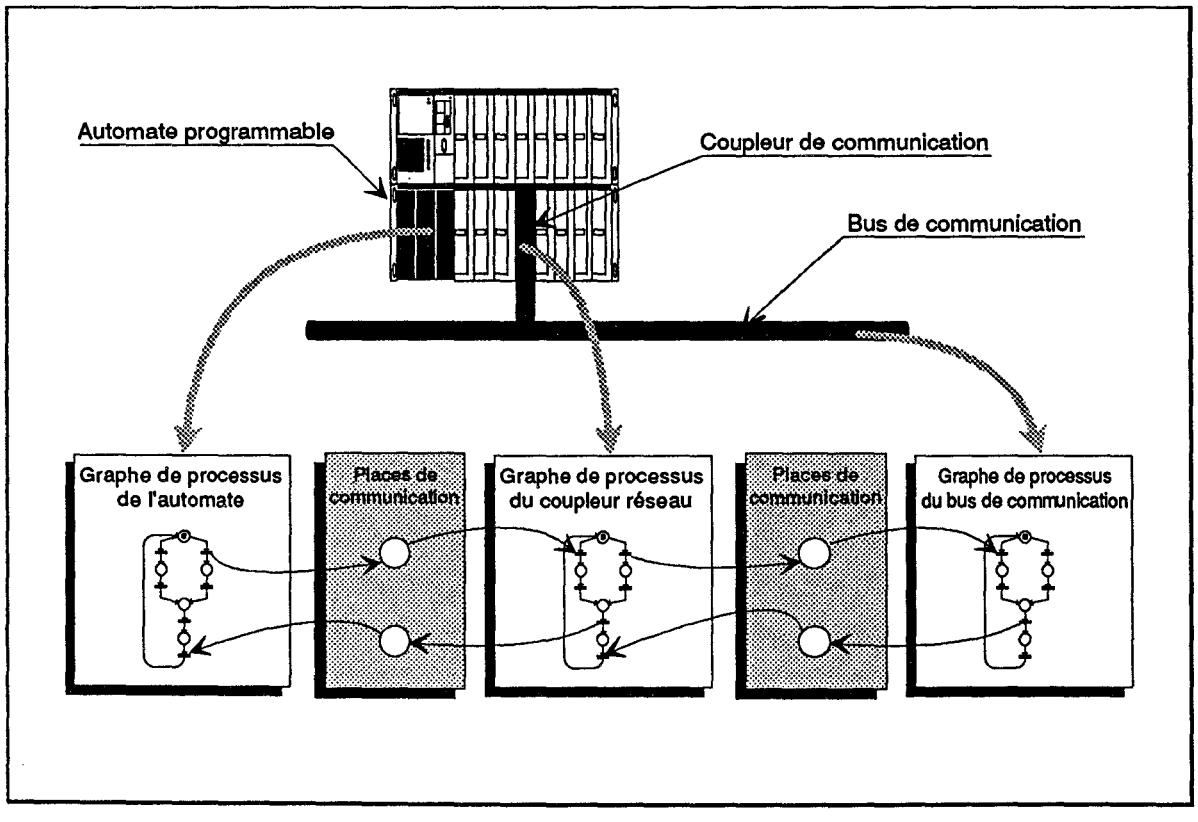


Figure II.8 : *principes de description du fonctionnement d'un automate programmable*

Nous avons choisi de représenter ces successions d'états par des graphes réseaux de Petri en les structurant en graphes de processus communicants. L'état du composant est alors matérialisé par la place du graphe qui est marquée en conjonction avec les valeurs de l'unique jeton du modèle. Le lien entre les graphes associés à différents équipements sont matérialisés par des places de communication.

II.1.2. Description des applicatifs

La description des applicatifs mis en place sur les stations de l'architecture permet de traduire au niveau du modèle, les séquencements des échanges de messages entre les matériels distants. Il s'agit donc de reproduire la partie communication des programmes développés sur les équipements. Celle-ci comprend d'une part l'émission et la réception effectives de messages mais également la prise en compte des événements qui provoquent ou conditionnent ces échanges.

Selon le degré de finesse souhaité de la modélisation, ou plus pratiquement en fonction des informations disponibles sur le fonctionnement du système à modéliser, la complexité de cette description peut être très variable. Pour certaines applications, il sera nécessaire de détailler de manière précise l'interaction entre procédé et commande pour que les scénarios de dialogues soient cohérents avec la réalité (Cf. chapitre III), alors que dans d'autres cas, une approche probabiliste d'émission de requêtes sera suffisante (Cf. annexe 2).

Dans un soucis de souplesse, nous proposons de modéliser le séquencement des échanges de niveau applicatif au moyen de l'interprétation du formalisme, c'est à dire en l'occurrence au moyen des procédures en langage Pascal disponibles avec l'outil SEDRIC. Le modèle du fonctionnement interne de l'équipement, construit à l'aide de graphes réseaux de Petri (Cf. II.1.1.) comprend alors certaines procédures laissées volontairement à compléter afin de décrire, pour chaque utilisation du composant dans un modèle d'architecture, les échanges de l'applicatif de la station considérée.

Les modèles du fonctionnement interne des équipements, i.e. le ou les graphes de processus réseaux de Petri avec les structures de données Pascal associées, doivent être construits en intégrant les liens avec les procédures Pascal de description des applicatifs. Au niveau de chaque procédure à compléter, les règles d'interfaçage définissent quelles sont les variables à utiliser pour décrire les émissions/réceptions de messages et assurer l'interaction avec l'image du procédé pour prendre en compte les scénarios de simulation.

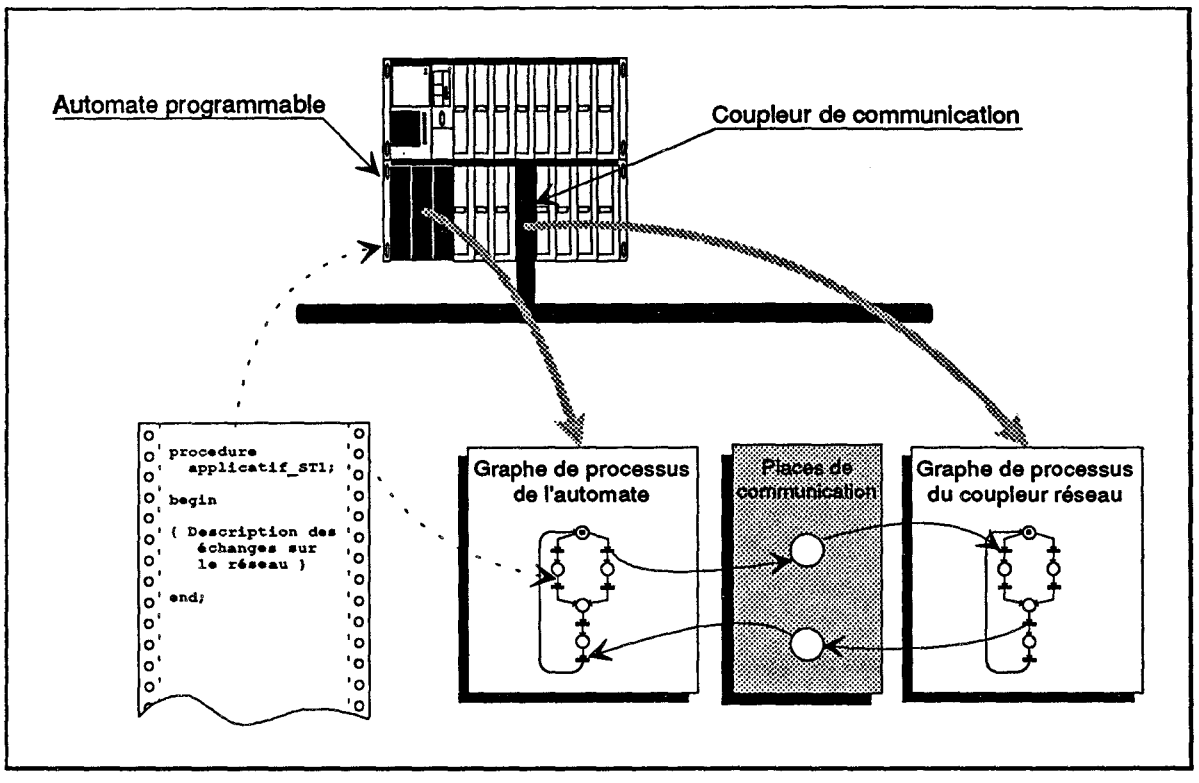


Figure II.9 : *description de l'applicatif dans un modèle d'automate programmable*

Dans la mesure du possible, il nous semble intéressant de proposer au niveau de la spécification des applicatifs, des formats de description et de structures de données qui soient proches de ceux utilisés lors du développement effectif des applications sur les équipements considérés. Cette philosophie doit permettre de rendre l'approche de modélisation plus facilement accessible aux personnes habituées à utiliser ces matériels.

A titre d'exemple, nous présentons dans le paragraphe III.2. des modèles d'automates TSX série 7 de la société Télémécanique [SOU 88]. Pour représenter les émissions/ réceptions de requêtes, nous avons choisi d'utiliser des enregistrements Pascal, dont la structure s'apparente à celle des blocs TxT utilisés dans les langages PL7-2 ou PL7-3 [TEL 88d][TEL 90c] utilisés pour programmer ces équipements. L'automaticien peut ainsi plus facilement faire le lien entre la description du modèle et le programme qu'il aura à développer.

II.1.3. Représentation de l'environnement

Une architecture de commande et de pilotage d'un système automatisé de production réagit à des événements en provenance du process à piloter ainsi qu'à des consignes transmises par un niveau supérieur (cadencement par exemple). Après acquisition et

traitement de ces informations le système de commande envoie des consignes vers le process pour le faire évoluer.

La détermination de ces consignes est réalisée par les programmes de contrôle/commande implantés dans les équipements de l'architecture. C'est donc notamment l'évolution de l'état du procédé qui conditionne l'évolution de la commande. C'est pourquoi il apparaît important de représenter les interactions entre le procédé et la commande qui conditionnent les échanges de données sur les réseaux.

Dans le cadre de ce mémoire, nous nous sommes limité à la représentation d'informations de type binaire. En effet, elles constituent une grande partie des données traitées dans l'industrie manufacturière, où l'interfaçage entre procédé et commande est réalisée au moyen de cartes d'entrées/sorties de type tout ou rien (T.O.R.).

Afin de tenir compte de l'asynchronisme entre l'évolution du process et celui de la commande, nous avons choisi de modéliser ces informations T.O.R. avec des places de communication dont le marquage représente l'état de ces données binaires. Bien entendu, il ne s'agit pas de représenter toutes les informations T.O.R. d'une application, mais de ne retenir que celles dont l'évolution provoque directement ou indirectement des échanges de messages sur les réseaux.

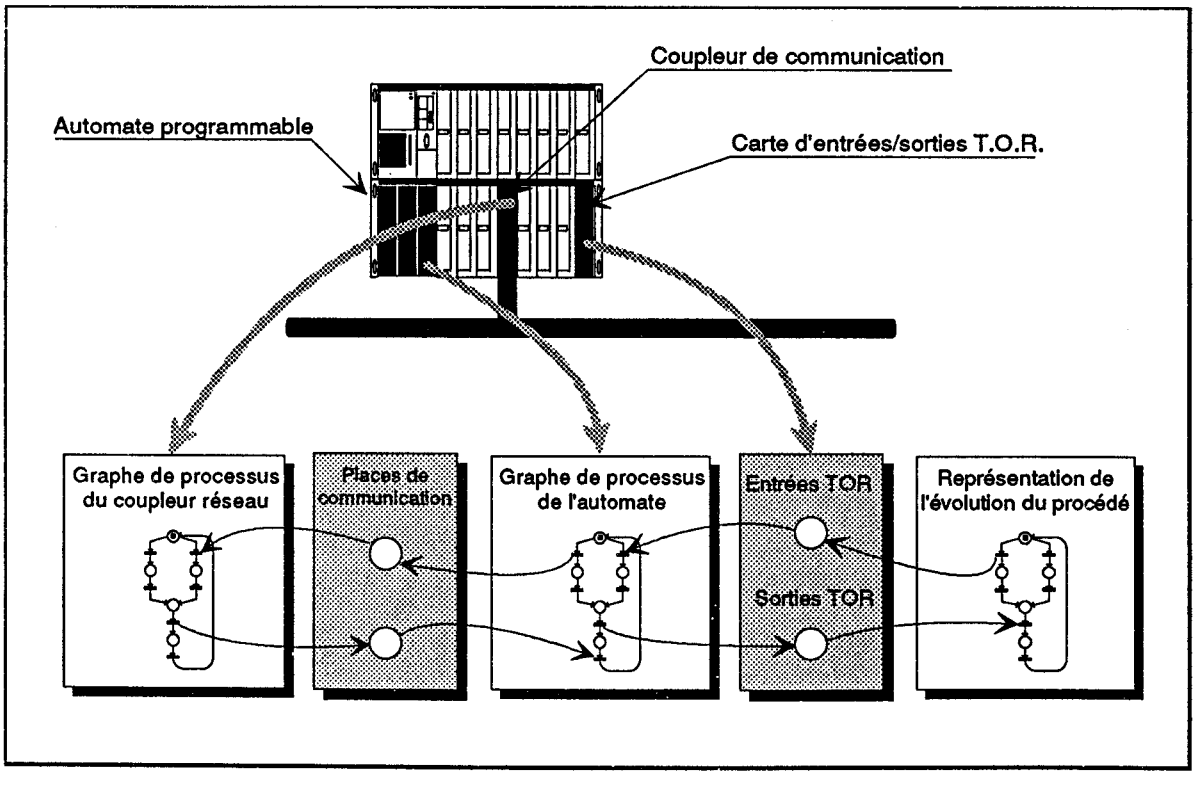


Figure II.10 : *interfaçage entre le modèle de l'architecture et celui du procédé*

La représentation de l'évolution du procédé est matérialisée par des changements d'état des valeurs T.O.R. associées. Il s'agit donc de modéliser un scénario réaliste de ces évolutions. Dans certains cas une approche probabiliste sera suffisante, alors que dans d'autres il pourra être nécessaire de modéliser plus finement le fonctionnement du procédé. Dans ce cas, on utilisera le même formalisme de réseaux de Petri que celui employé pour modéliser les équipements. En effet, les réseaux de Petri ont démontré leur capacité à modéliser ce type d'application [CER 88].

II.1.4. Conclusion

Nous venons de présenter dans les grandes lignes, la façon selon laquelle nous proposons de mettre en oeuvre le formalisme des réseaux de Petri. Il est important de remarquer que c'est grâce à la capacité d'expression des réseaux de Petri que les diverses entités intervenant dans une architecture de commande et de pilotage peuvent être modélisées. Dans les deux paragraphes qui suivent, nous présentons une méthode d'élaboration de graphes de processus structurés ainsi que des approches pour élaborer les scénarios de simulation.

II.2. Modélisation par graphes de processus communicants

II.2.1. Décomposition en graphes de processus

Nous supposons que le fonctionnement interne d'un équipement se ramène, à un niveau d'abstraction donné, à un ensemble d'états dont un seul est actif à un instant donné. La description dynamique d'un tel fonctionnement peut alors être assimilée à la représentation d'une succession de changements d'états. Nous avons adopté une décomposition en ***graphes de processus*** pour modéliser l'enchaînement de ces états : un jeton unique circule dans un graphe réseau de Petri qui boucle. Dans la suite de ce mémoire, ce jeton sera appelé ***jeton d'état***.

La couleur du jeton d'état indique le type de l'élément modélisé. Par exemple dans le graphe de processus associé à un coupleur de communication Uni-Telway configuré en maître, la couleur du jeton sera "Maître_UTW". Les paramètres de ce jetons permettent de caractériser finement l'état du composant, et complètent ainsi les informations d'état matérialisées par la place du graphe de processus qui est marquée.

La conjonction entre le marquage du réseau et les valeurs des paramètres du jeton identifie alors complètement l'état courant du composant modélisé. Un changement d'état correspond à une évolution du marquage donc au franchissement d'une transition du graphe, qui peut s'accompagner d'une modification des paramètres du jeton d'état.

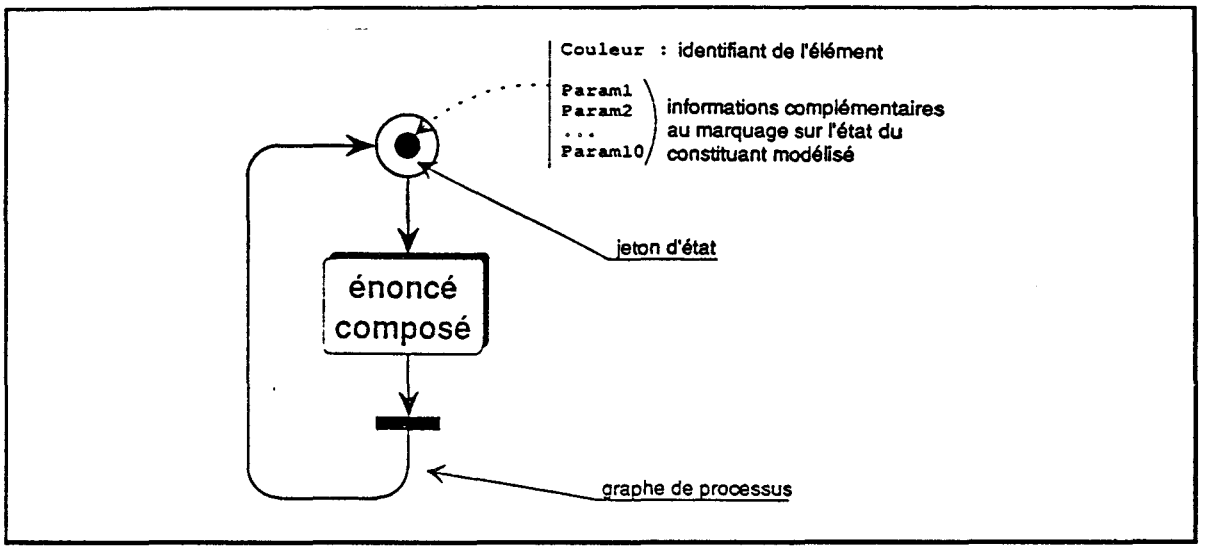


Figure II.11 : *structure générique d'un graphe de processus*

Les équipements communiquent entre eux, par exemple par bus interne dans le cas de coupleurs ou de cartes de communication connectés à une unité centrale. Les données échangées entre graphes de processus sont représentées par le marquage de places dénommées ***places de communication***. Les jetons, que nous appelerons ***jetons de message,*** matérialisent des données échangées et les valeurs des paramètres associés modélisent les informations véhiculées. La couleur du jeton permet de préciser le type du message. Une place de communication contient autant de jetons qu'il existe de messages en attente

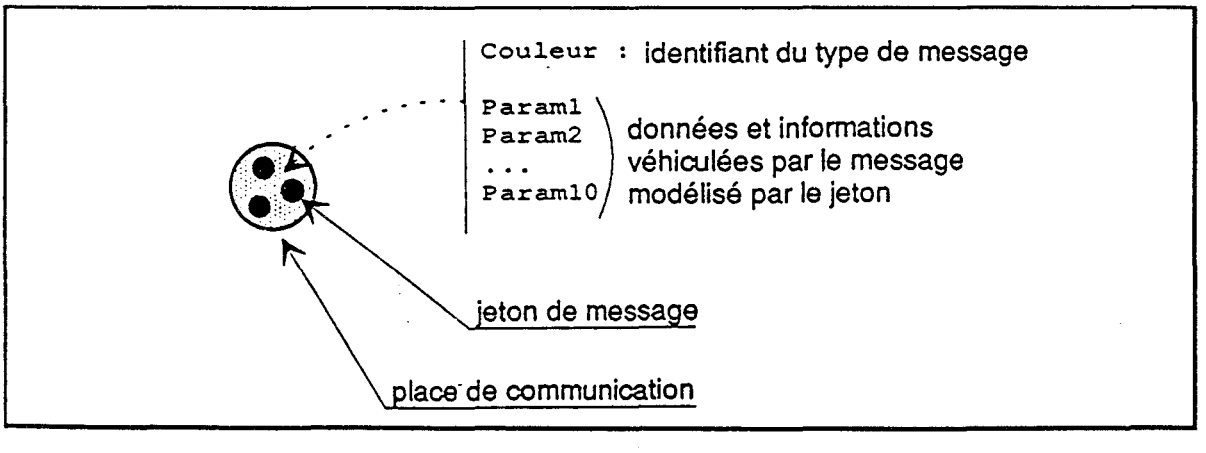


Figure II.12 : *place de communication entre graphes de processus*

Les jetons message sont créés et traités par les graphes de processus au travers d'énoncés réseaux de Petri qui seront décrits en détails dans le paragraphe II.2.3. C'est alors la présence ou l'absence de message dans les places de communication qui provoquent l'évolution du marquage des jetons d'état dans les graphes de processus. L'interprétation permet de conditionner le changement d'état en fonction des caractéristiques des messages, par exemple selon un numéro de destinataire.

Un graphe de processus est formé d'un énoncé composé (Cf. figure II.11). Celui-ci peut être un énoncé simple ou un énoncé simple suivi d'un énoncé composé (définition récursive [RIA 91b]). Un énoncé simple est, soit une structure de contrôle, soit un énoncé élémentaire. Les descriptions réseaux de Petri associées à ces énoncés sont détaillées dans les deux paragraphes qui suivent.

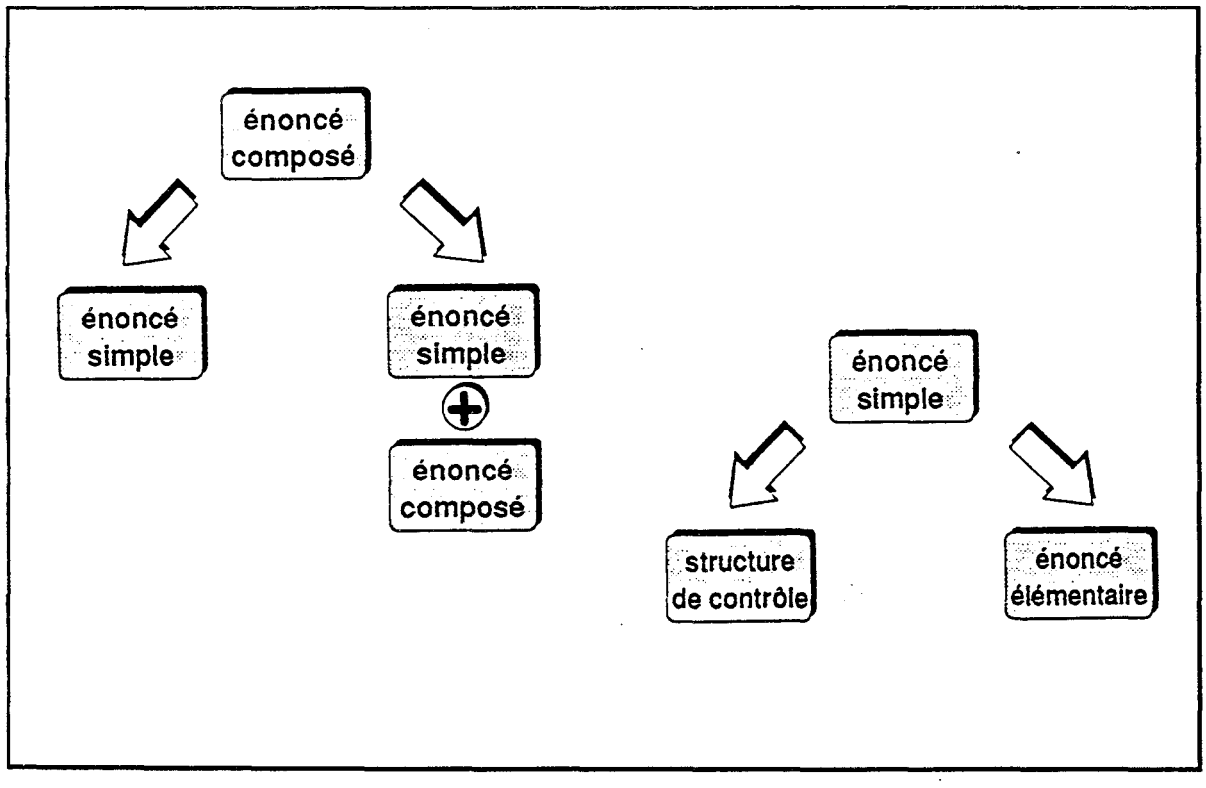


Figure II.13 : *définition récursive d'un énoncé composé*

II.2.2. Structures de contrôle

Trois structures de contrôle ont été définies, qui correspondent à celles disponibles dans un langage structuré, comme par exemple Pascal : l'alternative simple (IF...THEN...ELSE), l'alternative multiple (CASE...OF) et la répétitive (WHILE...DO ou REPEAT...UNTIL).

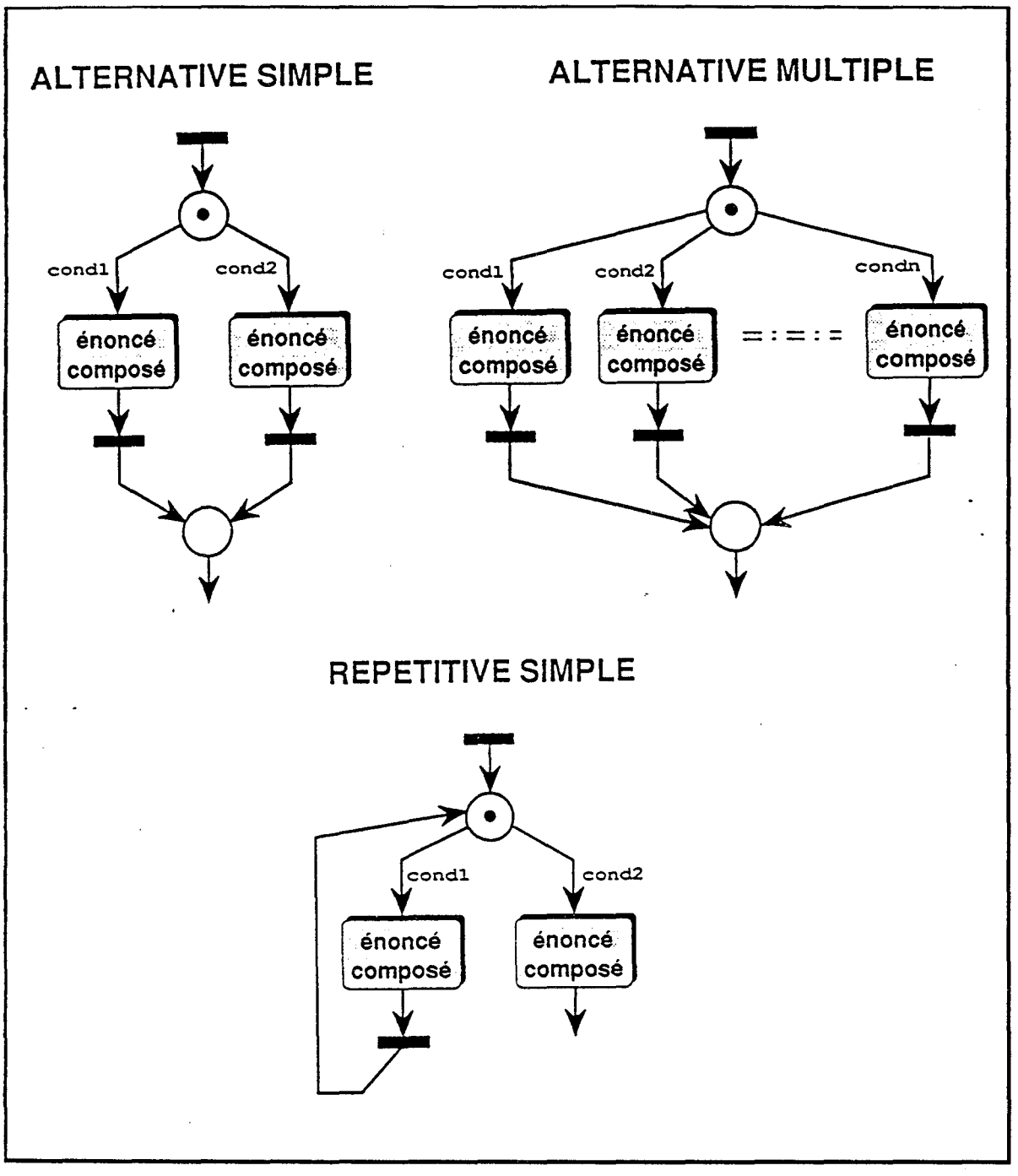


Figure II.14 : *structures de contrôle utilisées dans les graphes de processus*

II.2.3. Enoncés élémentaires

Les énoncés élémentaires permettent de traiter les parties d'évolution séquentielle à l'intérieur des structures de contrôle qui viennent d'être présentées. Il s'agit donc de décrire des enchaînements d'états, dont les transitions sont provoquées par des messages, ou comme nous le verrons dans le paragraphe suivant des activités constantes ou variables sur les places.

II.2.3.a Enoncés de base

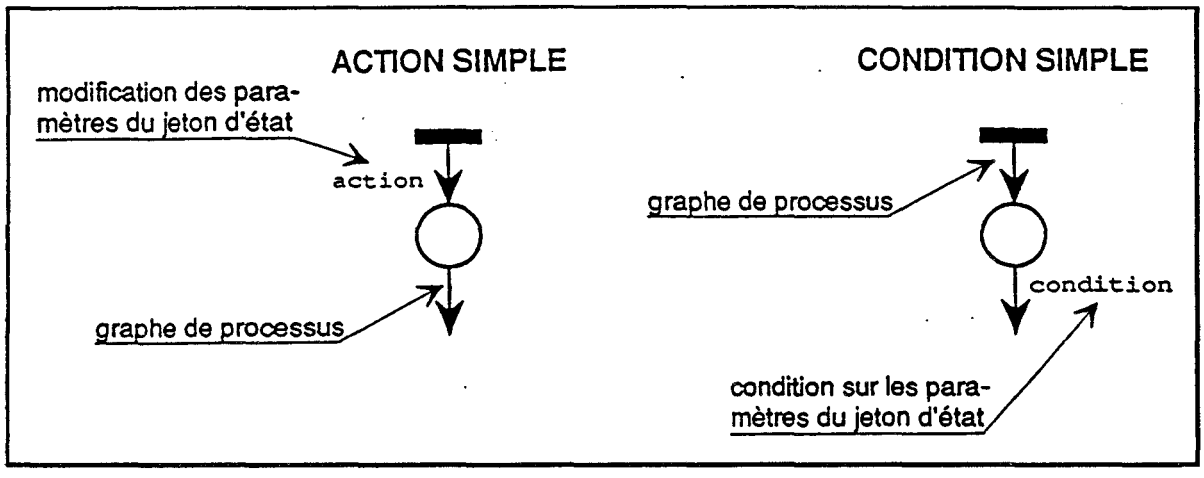


Figure II.15 : *énoncés élémentaires simples*

L'***action simple*** est utilisée, pour modifier les valeurs des paramètres du jeton d'état du graphe de processus. Cette opération est réalisée au moyen de la procédure appelée "`action`" sur la figure II.15. La ***condition simple*** permet de lier le franchissement de la transition aval à un test booléen. L'utilisation de cet énoncé impose que l'énoncé suivant soit une réception ou une lecture de message, et permet alors, comme nous le verrons dans les exemples du paragraphe II.2.5, de sélectionner les messages reçus en fonction des valeurs des paramètres du jeton d'état.

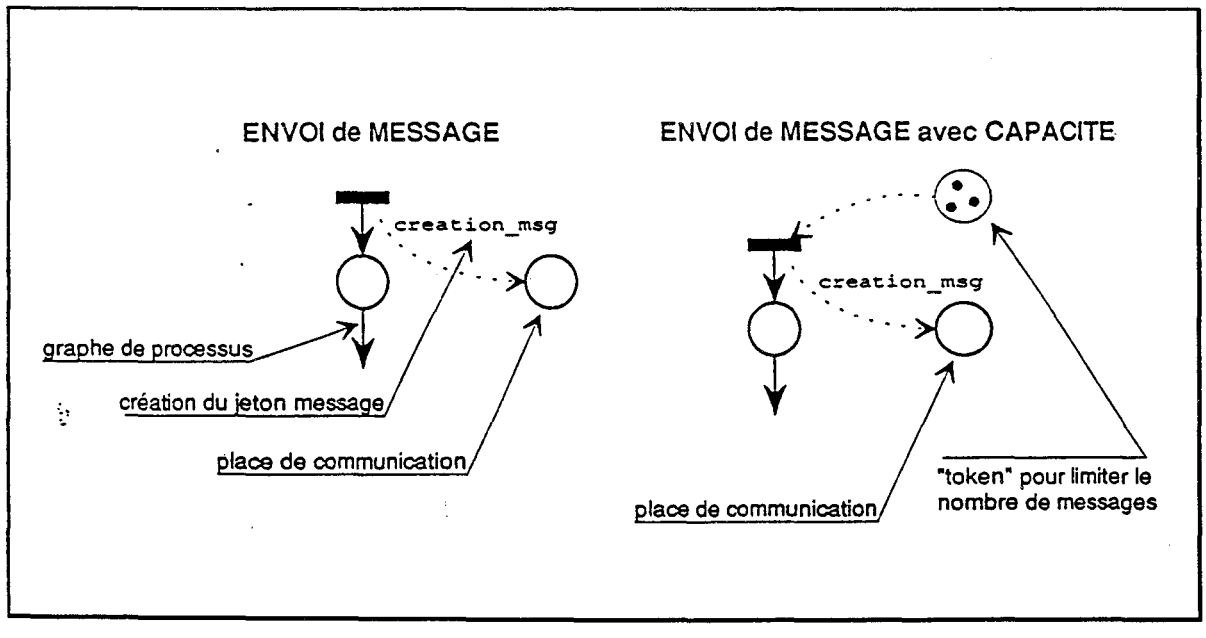


Figure II.16 : *énoncés d'envoi de message*

L'***envoi de message*** permet de créer des jetons de type message dans les places de communication. Si cette dernière représente un canal de communication limité en nombre de messages, on utilisera un énoncé ***envoi de message avec capacité***. Dans ce cas, le nombre de tokens présents lors du marquage initial correspond au volume maximal de messages simultanément en transit. A chaque envoi de message, un token est enlevé. Lorsqu'il n'en reste plus, cela signifie de le canal d'échange est saturé et donc la transition qui permet la création d'un jeton message n'est plus franchissable.

La procédure Pascal, dénommée "`creation_msg`" sur la figure II.16 permet de définir des caractéristiques du jeton message : couleur et valeurs des paramètres.

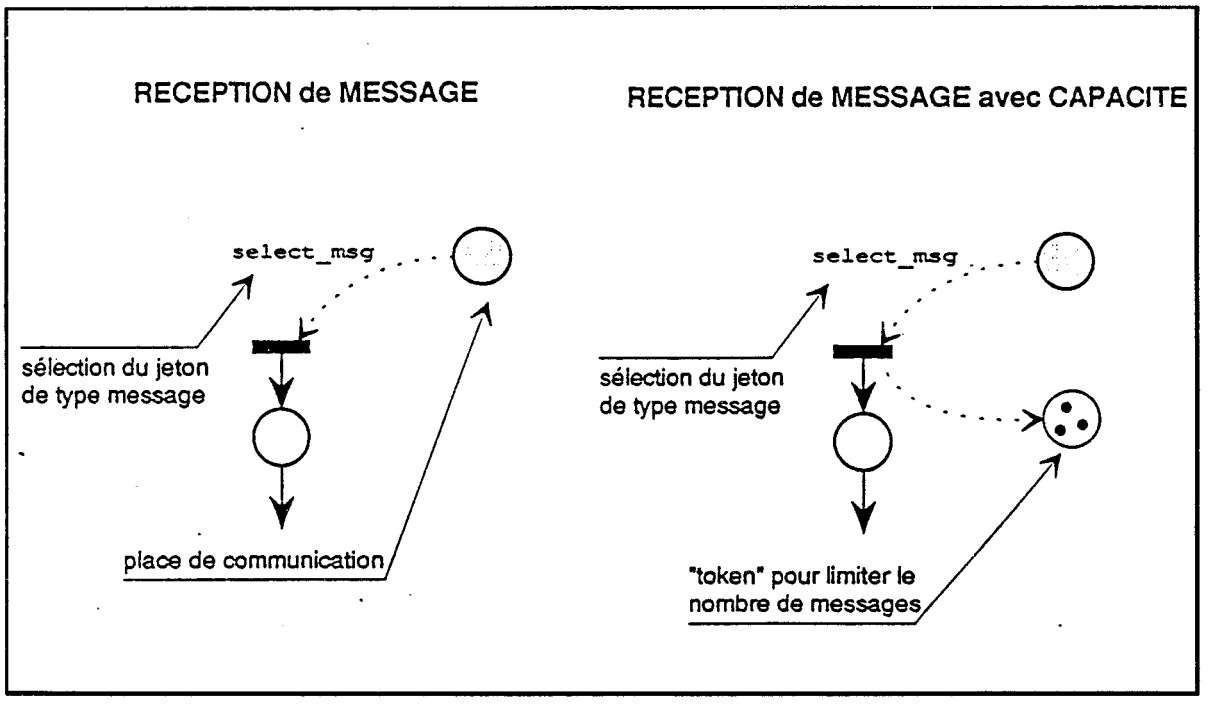


Figure II.17 : *énoncés de réception de message*

Les énoncés de ***reception de messages*** sont les complémentaires de ceux d'envoi, qui viennent d'être présentés. Nous retrouvons donc une primitive de réception simple et une autre de ***réception avec capacité*** pour tenir compte des phénomènes de zone d'échange de taille limitée. Dans ce cas, un token est ajouté à chaque réception de message, car cette action libère un emplacement dans la zone d'échange. La procédure Pascal appelée "`select_msg`" sur la figure II.17 permet de sélectionner les messages à prendre. On peut ainsi notamment créer des mécanismes de type FIFO ou LIFO avec les procédures Pascal "`pvjc`" ou "`npvjc`". Il est également possible de réaliser des choix selon des valeurs de paramètres, par exemple en fonction d'un numéro de destinataire, au moyen des procédures "`jpi`" et "`njpi`".

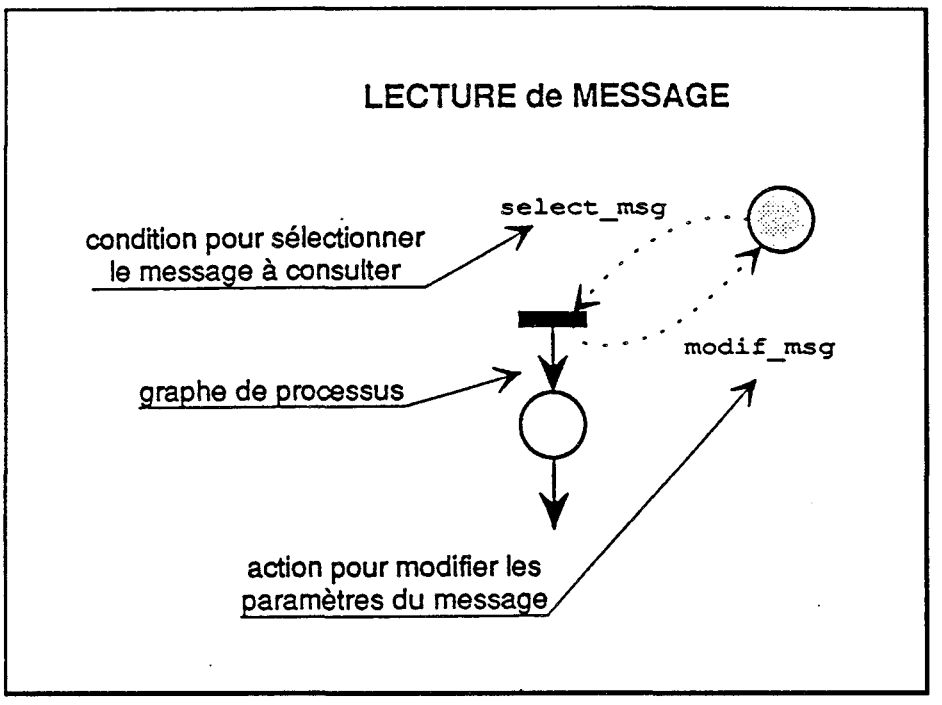


Figure II.18 : *énoncé de lecture de message*

L'énoncé de ***lecture de message*** permet de consulter et/ou de modifier les caractéristiques des paramètres de jetons messages présents dans des places de communication. Cette primitive autorise l'envoi de message en diffusion vers plusieurs graphes de processus. Chacun d'eux lit les paramètres du jeton associé au message et, en utilisant les priorité au niveau des transitions, un dernier processus supprime le message au moyen d'un énoncé réception de message.

II.2.3.b Enoncés combinés

Les énoncés de base qui viennent d'être détaillés constituent les primitives les plus simples pour décrire l'aspect séquentiel des graphes de processus. Celles-ci peuvent être conjuguées pour créer des énoncés élémentaires combinés. Ainsi, si la condition d'un changement d'état stipule qu'ils faut attendre deux messages présents dans des places de communication distinctes, on utilisera deux énoncés de type "réception message" (Cf. figure II.17).

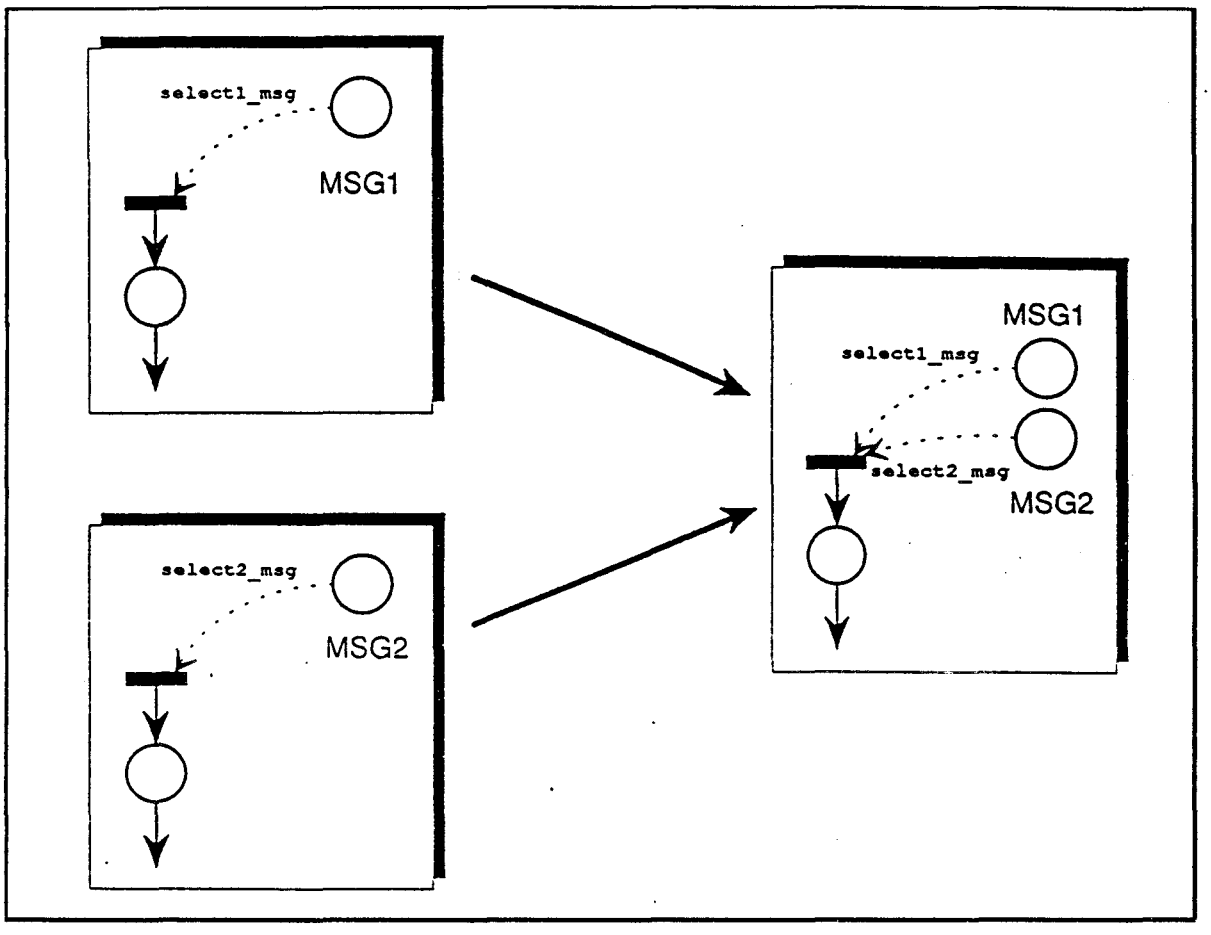


Figure II.19 : *exemple d'énoncé combiné*

II.2.4. Temporisation du modèle

Les temporisations sont des activités attachées aux places du modèle réseau de Petri : un jeton entrant dans une place temporisée ne devient disponible pour le franchissement d'une transition avale qu'une fois l'activité écoulée. Dans les deux paragraphes suivants, nous expliquons comment positionner les temporisations dans les modèles organisés en graphes de processus et de quelle manière décrire les activités.

II.2.4.a Positionnement des temporisations

Une activité dans un graphe de processus représente une durée associée à l'état modélisé par la place temporisée. Un délai sur une place de communication traduit une durée de transit pour un message, dans un canal de communication, comme par exemple un temps de transmission sur un bus. Dans ce mémoire, nous avons choisi de représenter graphiquement les places temporisées en mettant une croix dans la place.

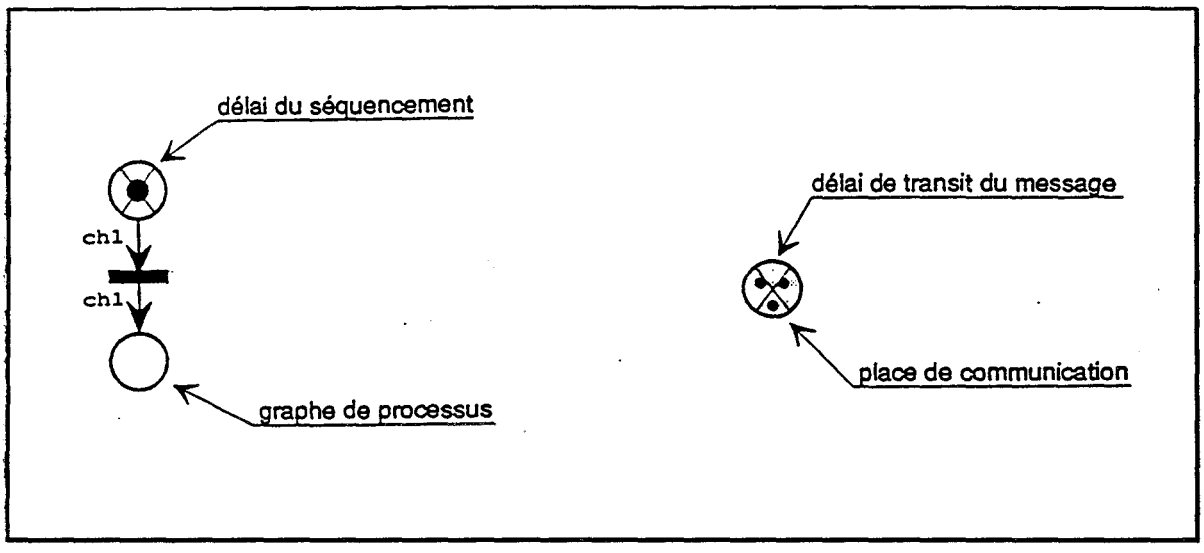


Figure II.20 : *activités associées au modèle réseau de Petri*

Pour les énoncés de traitement de messages (envoi, réception et lecture), les activités correspondent au délai nécessaire à l'équipement modélisé pour créer et émettre ou recevoir et traiter les données échangées. La figure II.21 détaille les énoncés temporisés possibles (les mêmes existent pour les énoncés équivalents avec capacité).

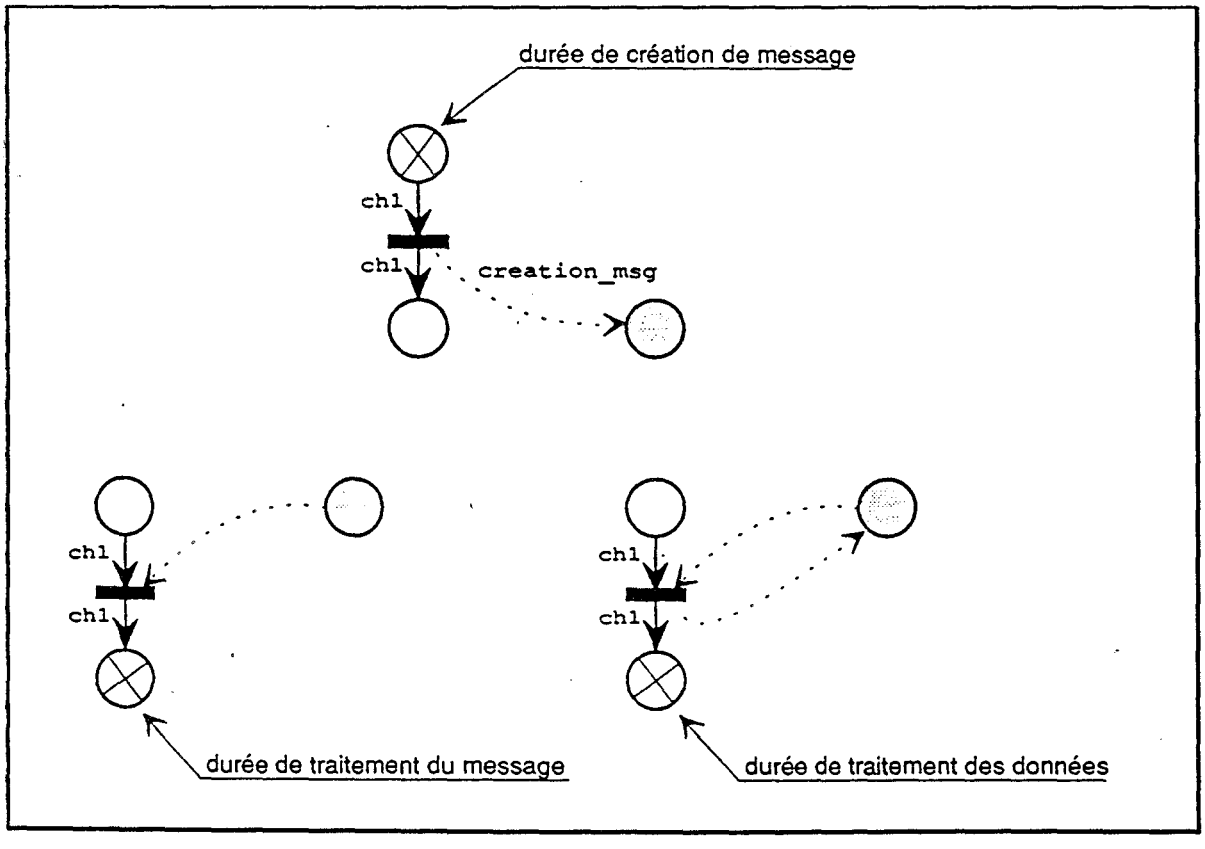


Figure II.21 : *énoncés temporisés d'échange de messages*

Pour compléter les énoncés disponibles, il est nécessaire de pouvoir modéliser le processus du time-out : un graphe de processus se trouve dans un état donné et si au bout d'un temps fixé (la durée du time-out), le marquage n'a pas changé, une alternative est déclenchée pour évoluer vers un état défini. Avec la structuration du modèle qui a été proposée, cela peut se produire lorsqu'un processus se trouve en attente de message et que ce dernier n'est pas disponible.

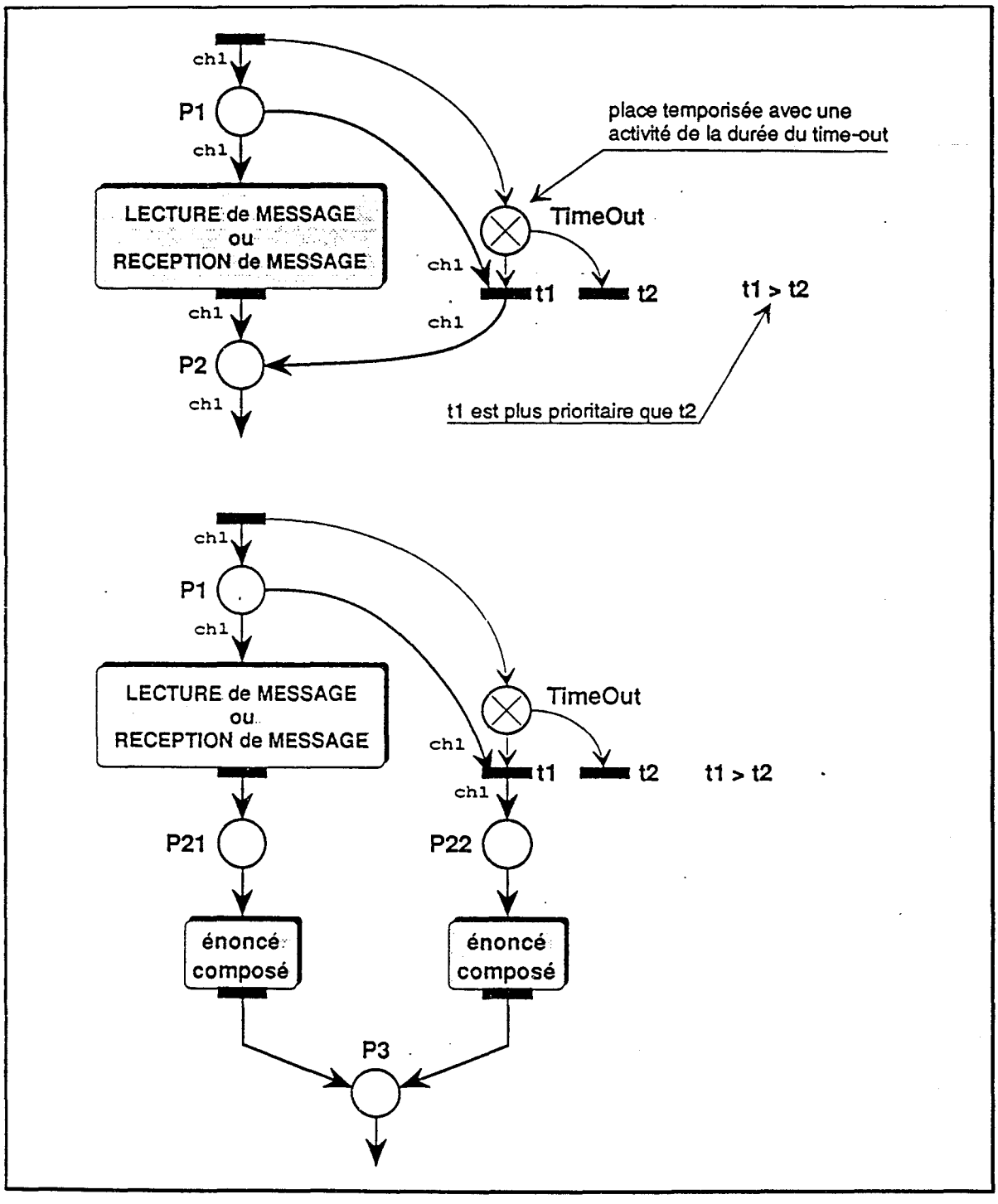


Figure II.22 : *énoncés modélisant un time-out*

La figure II.22 présente deux énoncés de time-out. Leurs principes sont identiques à la différence près que pour le premier le marquage généré sur time-out est le même que celui en fonctionnement normal (jeton d'état dans P2), alors que dans le second, les marquages sont différents (P21 ou P22). Dans le premier cas, la place marquée est la même, mais les états modélisés peuvent être différents en utilisant des actions pour modifier les paramètres du jeton d'état.

Sur les graphes de la figure II.22, la transition t2 est une transition puits. Lorsqu'elle est tirée (marquage de TimeOut et pas de marque dans P1), le jeton de la place TimeOut est détruit. Dans tous les cas, le jeton d'état du processus est conservé et l'ensemble des places du graphe de processus a un invariant de marquage de un; l'unique jeton est le jeton d'état (la place TimeOut doit être considérée comme une place auxiliaire qui reste marquée au plus pendant la durée du time-out).

Pour illustrer le fonctionnement du time-out, considérons le sous-ensemble d'un graphe de processus qui est reproduit sur la figure II.23. Le time-out est employé pour traiter l'absence de message dans la place MSG, lorsque le jeton d'état marque la place P2.

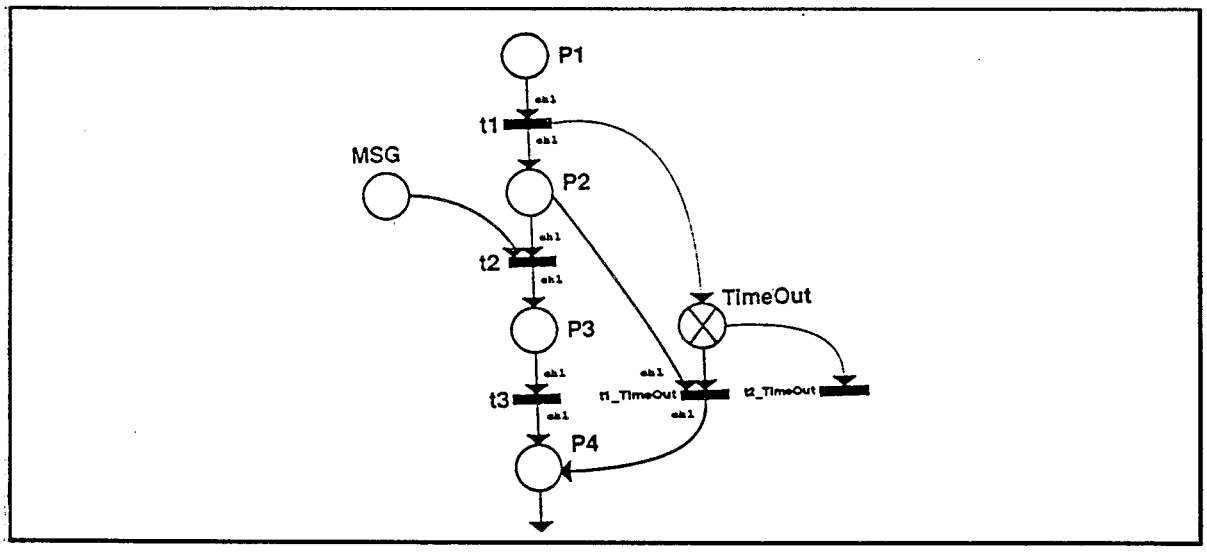


Figure II.23 : *sous-ensemble de graphe de processus utilisant un time-out*

Pour étudier l'évolution de ce modèle temporisé, supposons qu'à la date t, la place P1 soit marquée. La transition t1 est donc immédiatement tirée (P1 non temporisée), ce qui a pour conséquence de marquer P2 avec le jeton d'état (chemin ch1) et de créer un jeton dans la place TimeOut. Ce dernier ne deviendra disponible pour sensibiliser une transition (t1_TimeOut et t2_TimeOut) qu'au terme de la durée du time-out, soit à la date t + time_out.

Supposons qu'à la date t + delta, un jeton de type message (envoyé par un autre graphe de processus non représenté) soit disponible dans la place MSG. Pour l'étude du modèle, deux cas sont à envisager : 1 - delta < time-out,

2 - delta > time-out.

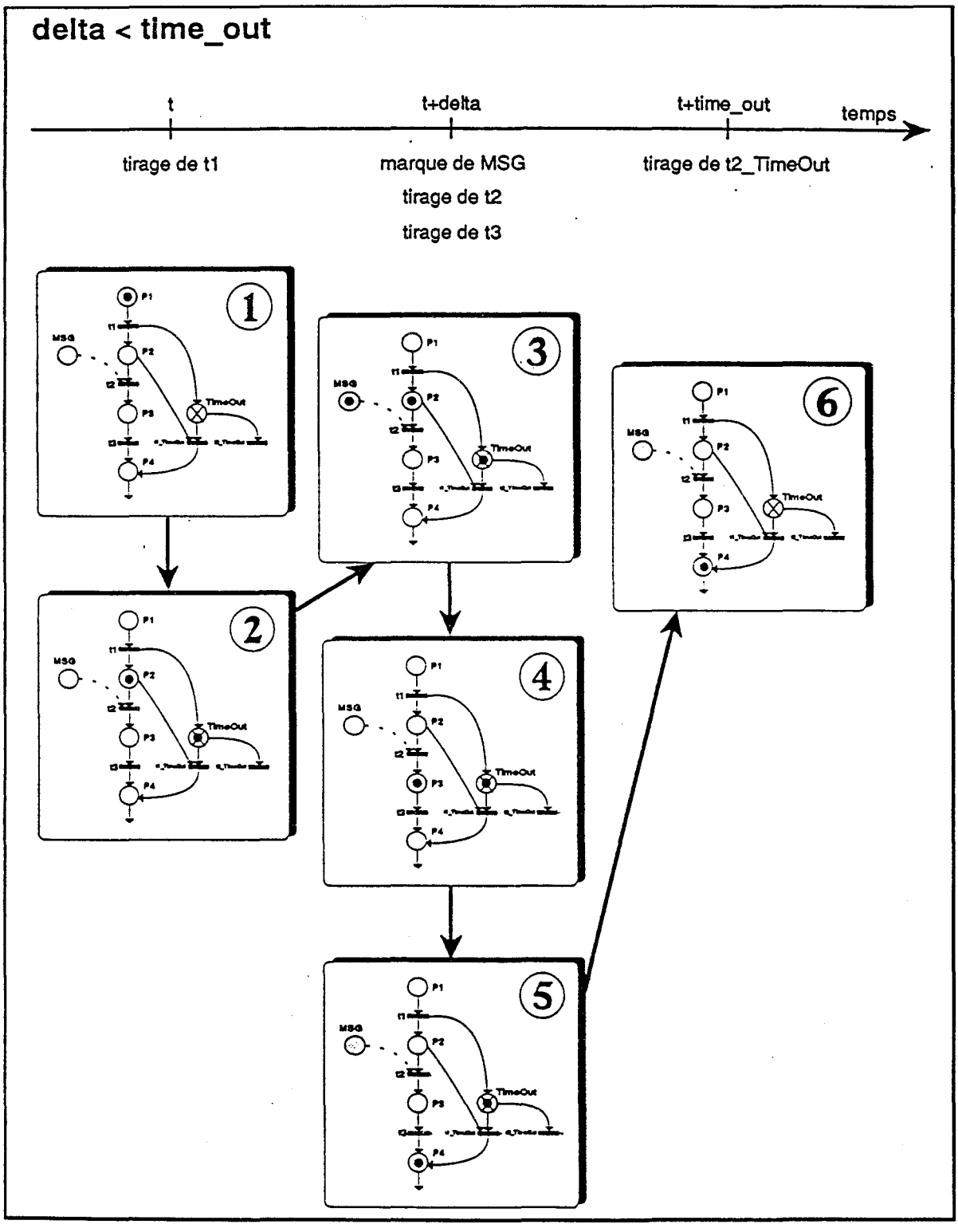


Figure II.24 : *chronogramme d'évolution lorsque delta < time_out*

L'évolution du marquage, lorsqu'un message devient disponible avant la fin du time-out, est représentée sur la figure II.24. Lorsque la place MSG est marquée, t2 puis t3 sont tirées à la date t + delta. A la fin de l'écoulement du time-out, le jeton de la place TimeOut devient disponible et la transition t2_TimeOut est donc franchie (transition puits qui provoque la suppression du jeton).

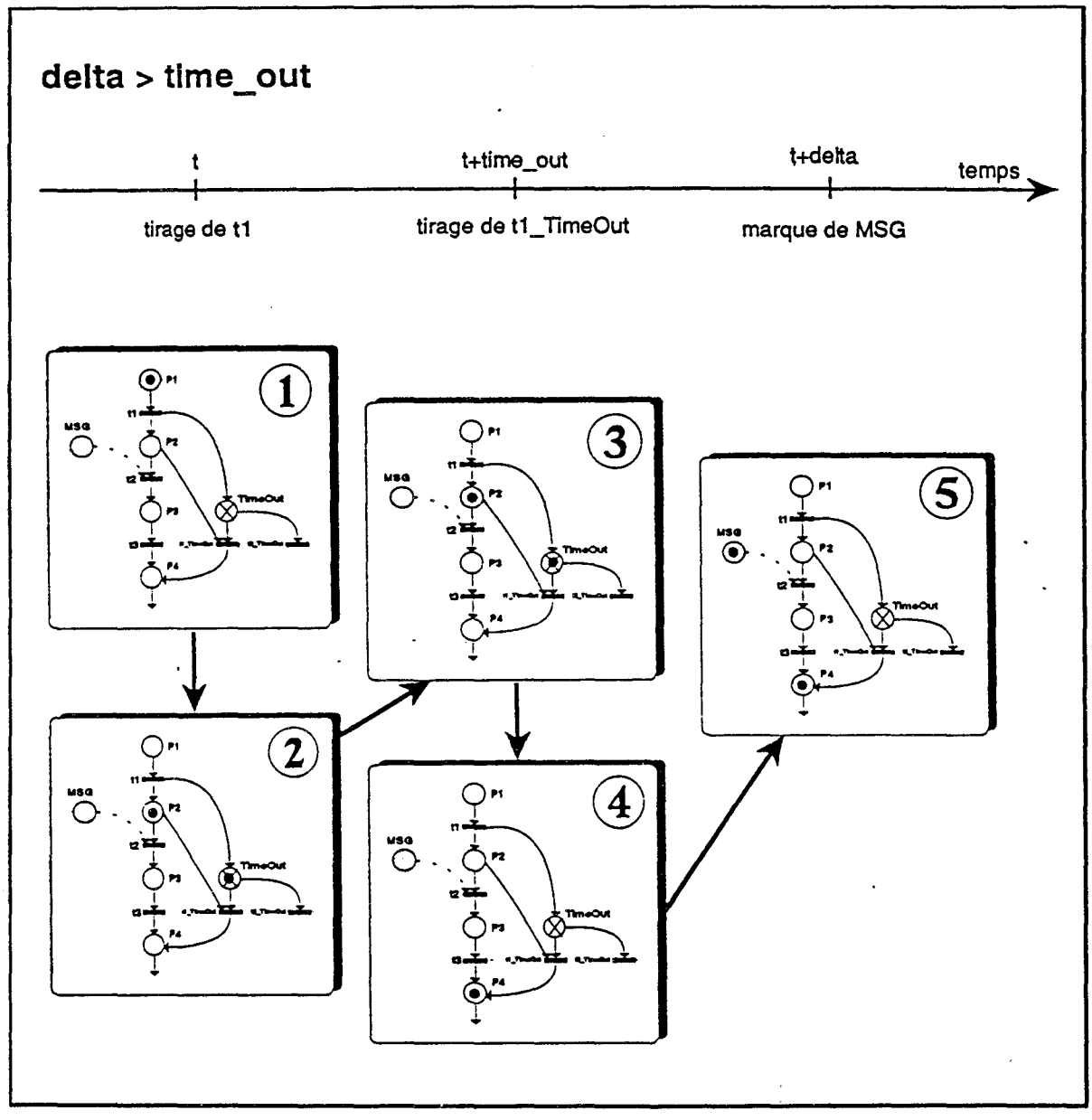


Figure II.25 : *chronogramme d'évolution lorsque delta > time_out*

Si à la fin du time-out, aucun message n'est disponible dans MSG, la transition t1_TimeOut est tirée (plus prioritaire que t2_TimeOut), et le jeton d'état marque la place P4 (chemin `ch1` de la transition t1_TimeOut). Dans le cas où un message devient disponible après la date t + time_out, le jeton restera dans la place MSG.

Remarque :

Avec le modèle, tel qu'il est détaillé sur la figure II.25, si un message marque la place MSG après la date t + time_out, il est disponible en lecture pour le graphe de processus, lorsque le jeton d'état est à nouveau dans la place P2. On peut aussi ne pas vouloir prendre en considération un tel message. Dans ce cas, il est nécessaire de le détruire. Une manière de procéder est alors d'utiliser un second message, comme proposé sur la figure II.26.

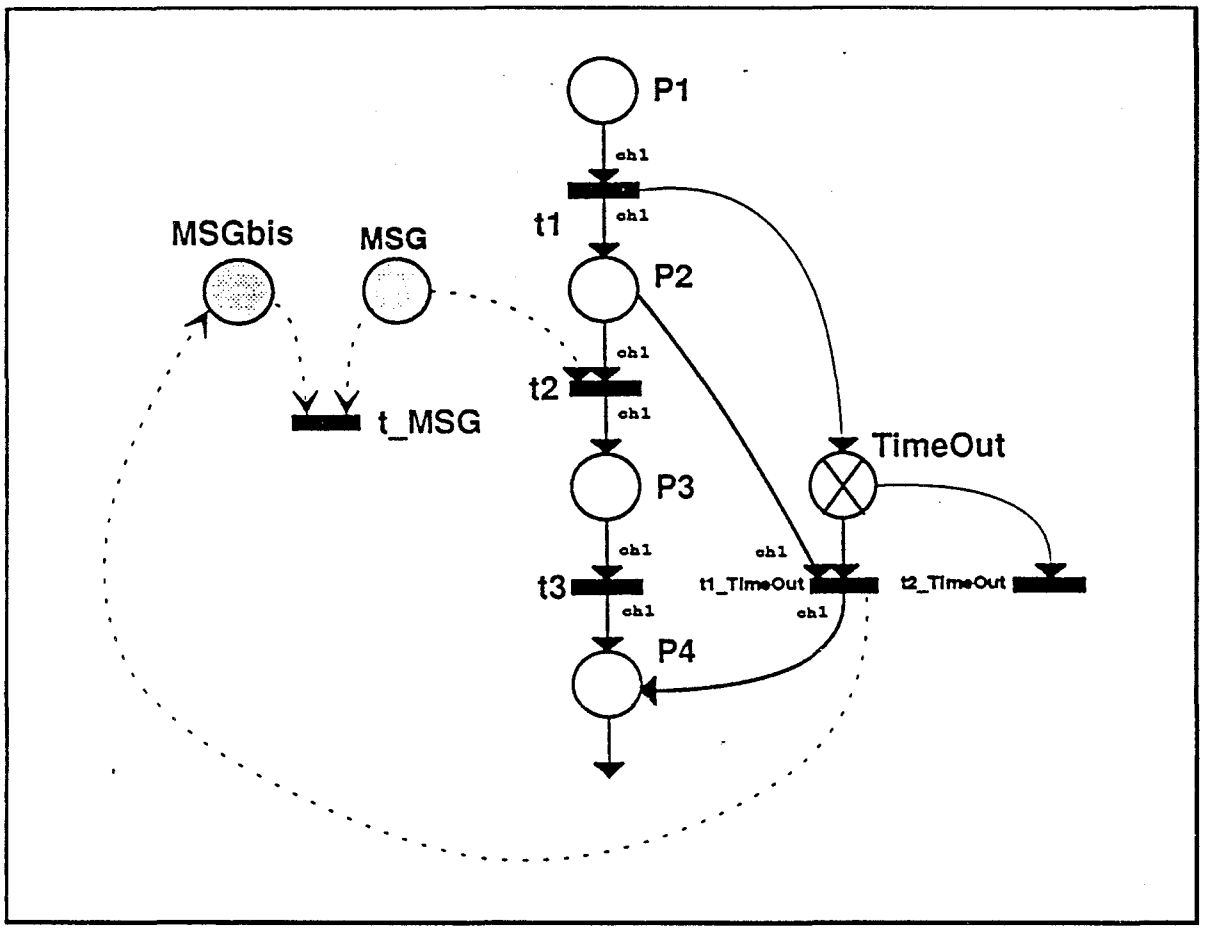


Figure II.26 : *time-out avec destruction du message retardé*

Dans ce cas, lorsque la transition t1_TimeOut est tirée, c'est à dire lorsque le message attendu n'est pas disponible à la fin du time-out, un jeton message est créé dans la place MSGbis. Ceci a pour conséquence de permettre le tir de la transition t_MSG (transition puits) dès qu'un message est disponible dans la place MSG. Ainsi, le message retardé, qui n'a pu être pris en compte par le graphe de processus, est effectivement supprimé.

II.2.4.b Valeurs des temporisations

Les valeurs numériques associées aux temporisations correspondent à des durées d'état de l'équipement modélisé. Il peut donc s'agir de valeurs constantes (temps de retournement pour un coupleur de communication par exemple) ou de grandeurs stochastiques (temps CPU consommé par un process). Dans ce cas, on utilisera des lois aléatoires : répartition uniforme, répartition triangulaire, répartition normale etc.

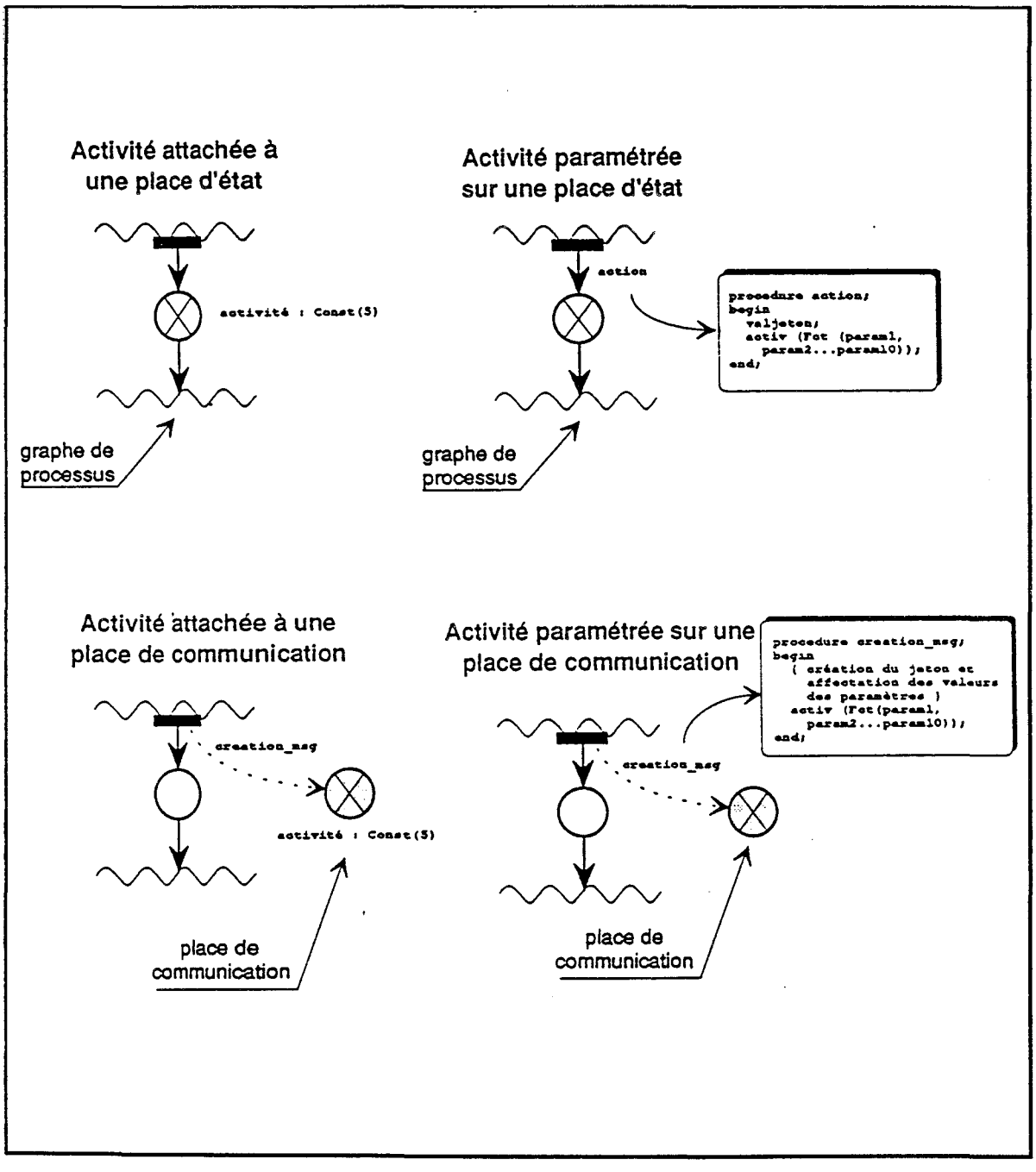


Figure II.27 : *mise en place des activités sur le modèle réseau de Petri*

Au niveau du modèle, la durée d'une activité peut être prise en compte de deux manières. Soit la temporisation est directement attachée à une place, soit elle est définie dans une action amont à la place au moyen de la procédure Pascal `'activ'`. L'intérêt de cette seconde approche est de pouvoir paramétrer l'activité, en fonction de valeurs d'attributs de jeton d'état ou de message. Cette fonctionnalité est très utile, par exemple pour temporiser une place de communication en fonction du type ou de la taille du message en transit.

Remarque :

Lorsque plusieurs temporisations sont consécutives dans un graphe de processus, sans émission ni réception de message, elles peuvent être regroupées. Cette philosophie a pour objectif de diminuer la taille des modèles, ce qui entraîne une réduction des durées de simulation (l'échéancier gère moins d'événements).

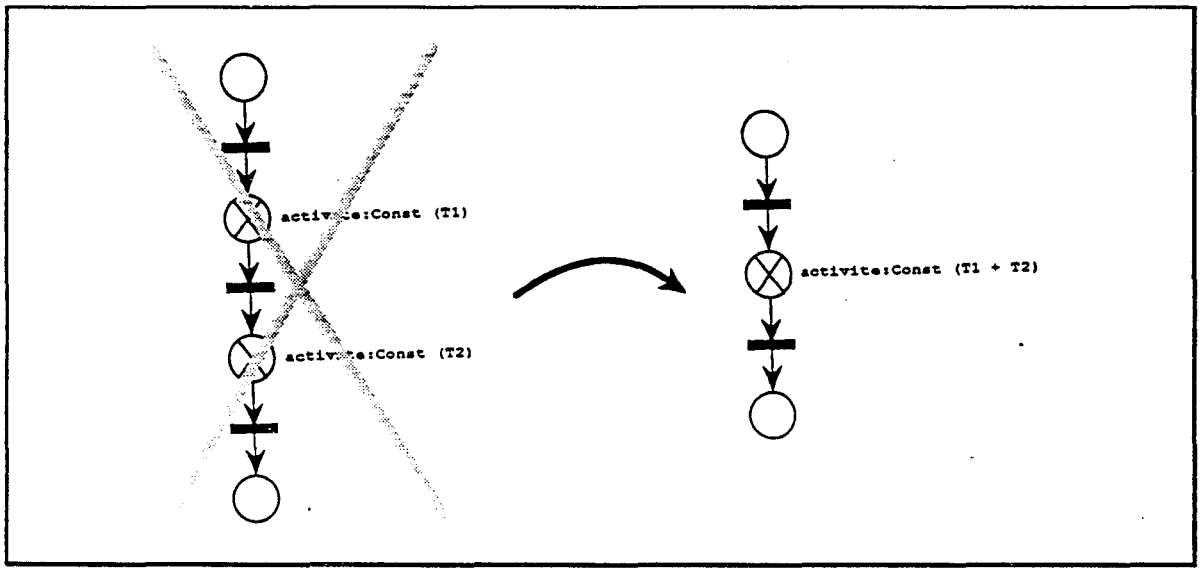


Figure II.28 : *regroupement de temporisations*

II.2.5. Caractéristiques des modèles

II.2.5.a Propriétés des modèles

Notre objectif, en proposant une structuration du modèle réseaux de Petri en graphes de processus communicants, est double. Cette approche permet d'une part de guider la construction des modèles en offrant un ensemble de primitives d'actions de base (émission/réception de message, time-out etc.) à "assembler" pour former des graphes de processus. Un effet induit de cette démarche est alors d'entraîner une homogénéité dans les différents modèles.

Mais la structuration a aussi pour conséquence d'assurer certaines "bonnes" propriétés au modèle. Comme nous utilisons un formalisme réseaux de Petri étendus, l'étude de ces propriétés doit être réalisée sur le modèle de base sous-jacent, c'est à dire sur le graphe de Petri binaire obtenu en supprimant les interprétations, priorités et caractères temporels du modèle.

Nous pouvons tout d'abord remarquer que, par construction, chaque graphe de processus est sauf [BOU 88]. En effet, pour tout marquage accessible, chaque place d'état peut contenir au plus un jeton, qui est en fait l'unique jeton d'état du graphe. Cette propriété est même plus forte, puisque dans un graphe de processus, l'ensemble des places d'état constitue un invariant de marquage égal à un. Ceci se démontre facilement en remarquant que dans chaque primitive utilisée dans la construction des graphes de processus, le jeton d'état du marquage initial est toujours conservé et n'est jamais dupliqué ni détruit. Par contre, les places du communication (sauf celles avec capacité) ne sont pas nécessairement bornées, et une étude spécifique doit être menée pour chaque modèle.

Avec la structuration proposée, aucune propriété relative à la vivacité ou aux possibilités de blocage du modèle ne peut être démontrée de manière générale. En effet, si le modèle est mal "pensé", un graphe de processus peut, par exemple, se trouver en attente d'un message qui n'arrive jamais. Il s'agit alors d'une mauvaise description du système. Cependant, dans un graphe de processus bien construit, doivent exister des enchainements d'arrivées de messages dans les places de communication, qui permettent au jeton d'état de marquer chacune des places du graphe.

Remarque :

Avec l'approche de décomposition en graphes de processus que nous proposons, l'évolution parallèle de systèmes est prise en compte par les graphes communicants. C'est pourquoi, un modèle ne doit pas comporter plus d'une place d'état en sortie d'une transition. Ceci conduirait en effet à un dédoublement du jeton d'état dans un graphe ce qui qui serait en opposition avec la définition d'un graphe de processus telle qu'elle a été exposée dans le paragraphe II.2.

Une transition peut cependant comporter plusieurs places de sortie. Mais dans ce cas, il s'agit nécessairement d'un énoncé d'émission ou de lecture de message, ou éventuellement d'un énoncé combiné. En sortie de la transition, il existe alors une et une seule place d'état et les autres sont des places de communication.

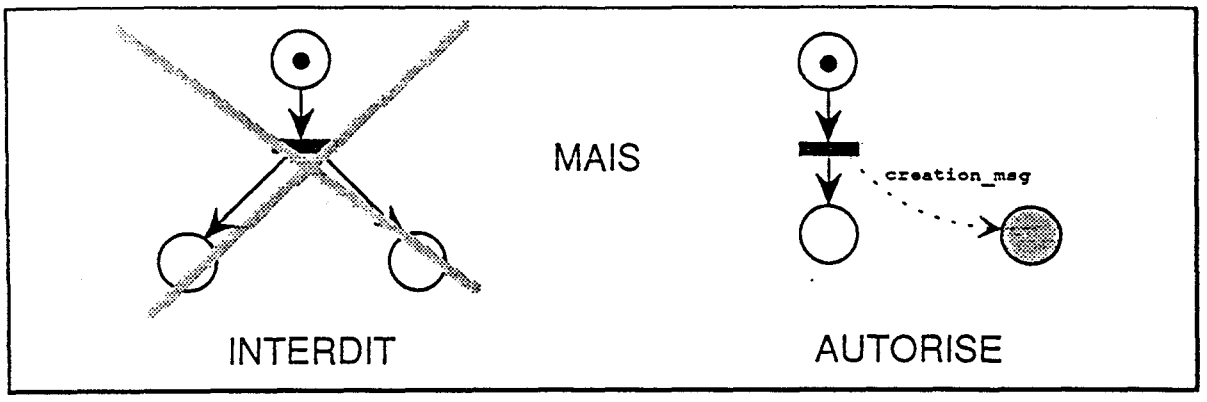


Figure II.29 : *structures autorisée et interdite dans un graphe de processus*

II.2.5.b Interprétation du modèle

L'interprétation (condition et action) ajoutée au modèle de base permet d'affiner le modèle en levant des indéterminismes, mais peut aussi engendrer des blocages qui n'existent pas sur le modèle sous-jacent. C'est pourquoi, il est indispensable de bien discerner, au niveau de l'évolution du modèle, les phénomènes qui sont pris en compte par le marquage et ceux qui se traduisent dans l'interprétation.

Pour qu'une transition soit franchissable, il faut qu'elle le soit au sens du marquage et que les conditions éventuelles sur chacun de ses arcs d'entrée soient vérifiées. Les préconditions sont utilisées pour résoudre des indéterminismes lorsque plusieurs transitions sont franchissables simultanément avec des places d'entrée communes, ou qu'une même transition est franchissable de différentes manières (plusieurs messages disponibles par exemple). L'interprétation, conjuguée avec les priorités entre transitions, doit alors permettre de préciser, sans ambiguïté pour les simulations, la façon dont le modèle évolue.

L'échéancier de l'outil SEDRIC détecte les transitions franchissables au sens du marquage et pour chacune d'elles exécute les procédures Pascal affectées aux arcs amont, dans un ordre qui est fonction des priorités entre transitions, des noms de chemin et de ceux des places, jusqu'à ce qu'une transition soit effectivement franchissable.

Afin de limiter les durées de simulation, nous interdisons au niveau de la construction du modèle, les cas de figure où plusieurs transitions en conflit sont simultanément franchissables, alors que l'interprétation n'autorise le tirage d'aucune d'entre elles. Ce choix nous paraît être en conformité avec la philosophie des réseaux de Petri de base. En effet, c'est toujours le marquage qui provoque l'évolution du modèle, les aspects priorité entre transitions et interprétation n'étant employés que pour rendre le modèle déterministe.

La première conséquence de cette approche est d'interdire la mise en place de condition sur des arcs uniques en entrée de transition. Sur l'exemple de la figure II.30, la condition appelée 'cond' liée aux paramètres du jeton d'état n'a aucun sens car si elle n'est pas vérifiée, la transition ne pourra jamais être tirée.

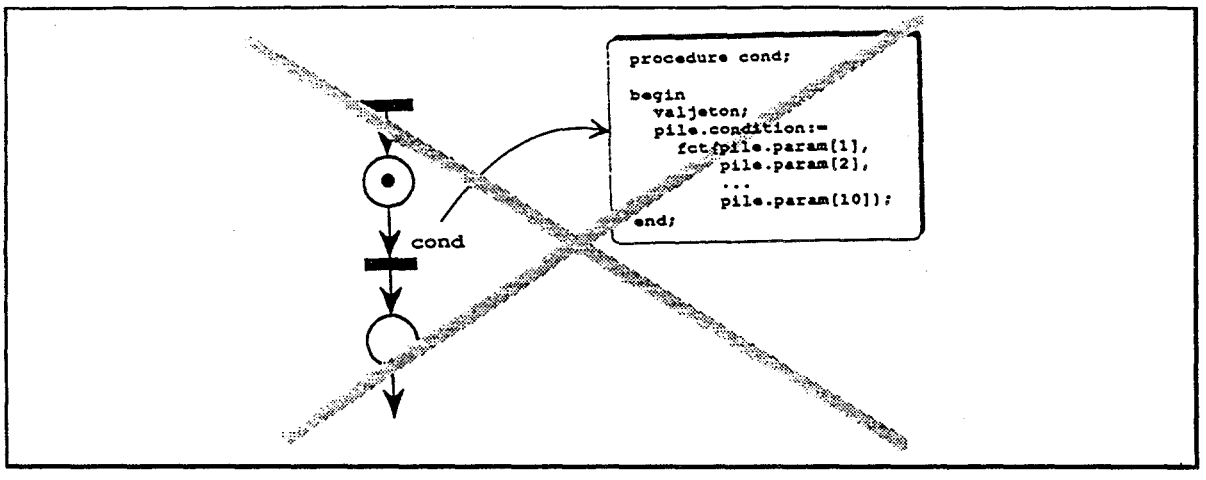


Figure II.30 : *graphe où l'utilisation de condition n'a pas de sens*

De la même manière, utiliser une variable Pascal, globale au modèle, pour en modifier la valeur dans une procédure de type action et l'inclure dans une précondition sur un autre graphe est déconseillé. En effet, dans l'exemple de la figure II.31, le jeton d'état pourrait se trouver bloqué dans la place P11 jusqu'à ce que t2 soit tirée et que le paramètre 'toto' soit modifié de façon à rendre 'cond1' vraie. On retrouve alors un cas de figure où le marquage autorise le tir d'une transition mais où l'interprétation l'interdit.

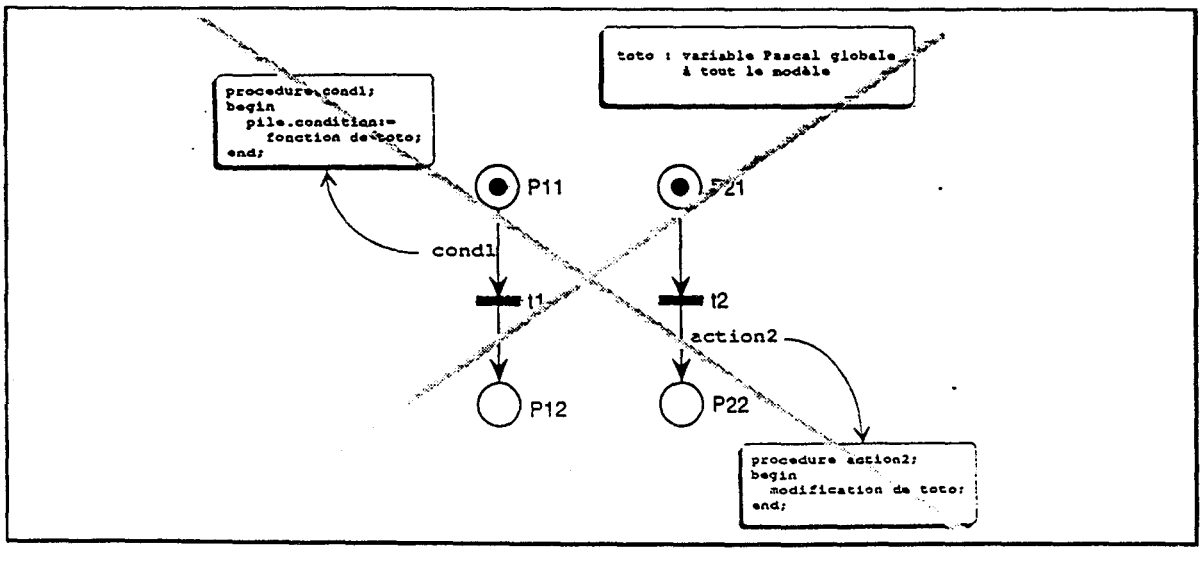


Figure II.31 : *variable globale partagée entre deux graphes de processus*

Dans le cas de l'alternative présentée sur la figure ci-dessous, t1 est plus prioritaire que t2. C'est donc 'cond1' qui est évaluée en premier. Si cette condition est vraie, t1 est franchie. Dans le cas contraire, 'cond2' est évaluée est doit donc forcément être vraie sans quoi le jeton d'état ne peut pas évoluer.

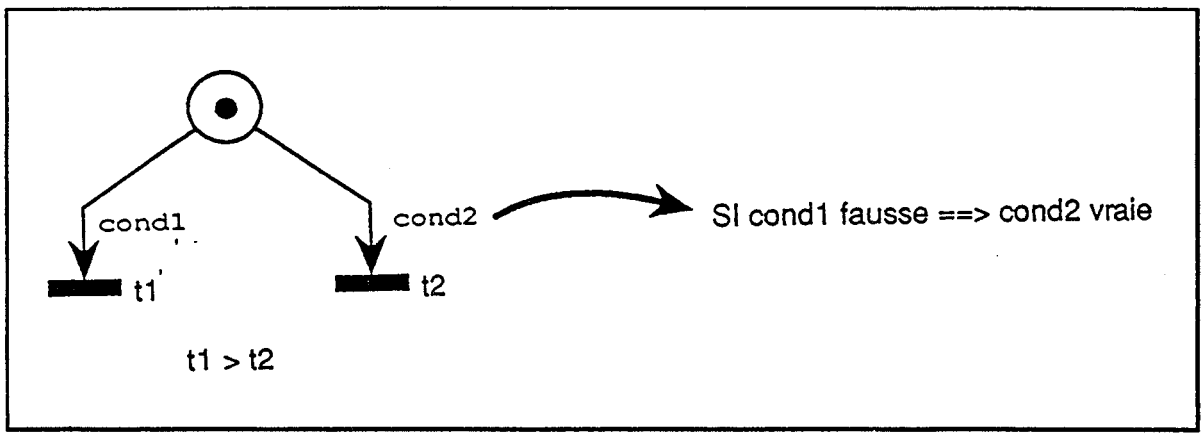


Figure II.32 : *contrainte sur une alternative pour ne pas bloquer un graphe de processus*

Pour une primitive de lecture de message, la procédure 'select_msg' doit permettre de sélectionner l'unique jeton de type message qui est choisi pour tirer la transition. Une solution simple consiste à employer des gestions de file standard comme la FIFO (procédure 'pvjc') ou la LIFO (procédure 'npvjc').

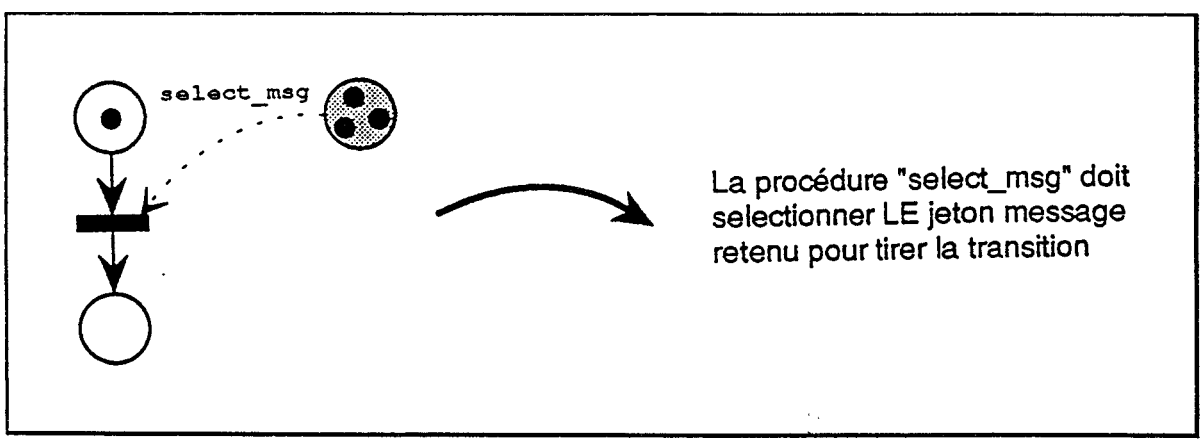


Figure II.33 : *contrainte sur un énoncé de lecture de message*

Lorsque le choix du message est fonction de paramètres liés au jeton d'état, il est nécessaire d'être en mesure de réaliser un lien entre des attributs de jetons présents dans différentes places amont à la transition. Il est alors indispensable d'employer des variables locales à une transition comme cela est expliqué dans le paragraphe ci-dessous.

II.2.5.c Variables locales aux transitions

La figure II.34 représente un cas générique de mise en oeuvre de variables locales à une transition, avec deux places amont et deux places aval. Cet exemple peut très facilement être généralisé à une transition comportant un nombre plus important de places amont et/ou aval.

Les variables aux11..aux1n, aux21..aux2p sont des variables déclarées de type réel et globales à tout le modèle. Cependant, leurs valeurs ne sont utilisées et n'ont une signification qu'au moment des tests des conditions de franchissement et du tirage de la transition étudiée. On peut donc voir ces variables comme étant locales à la transition.

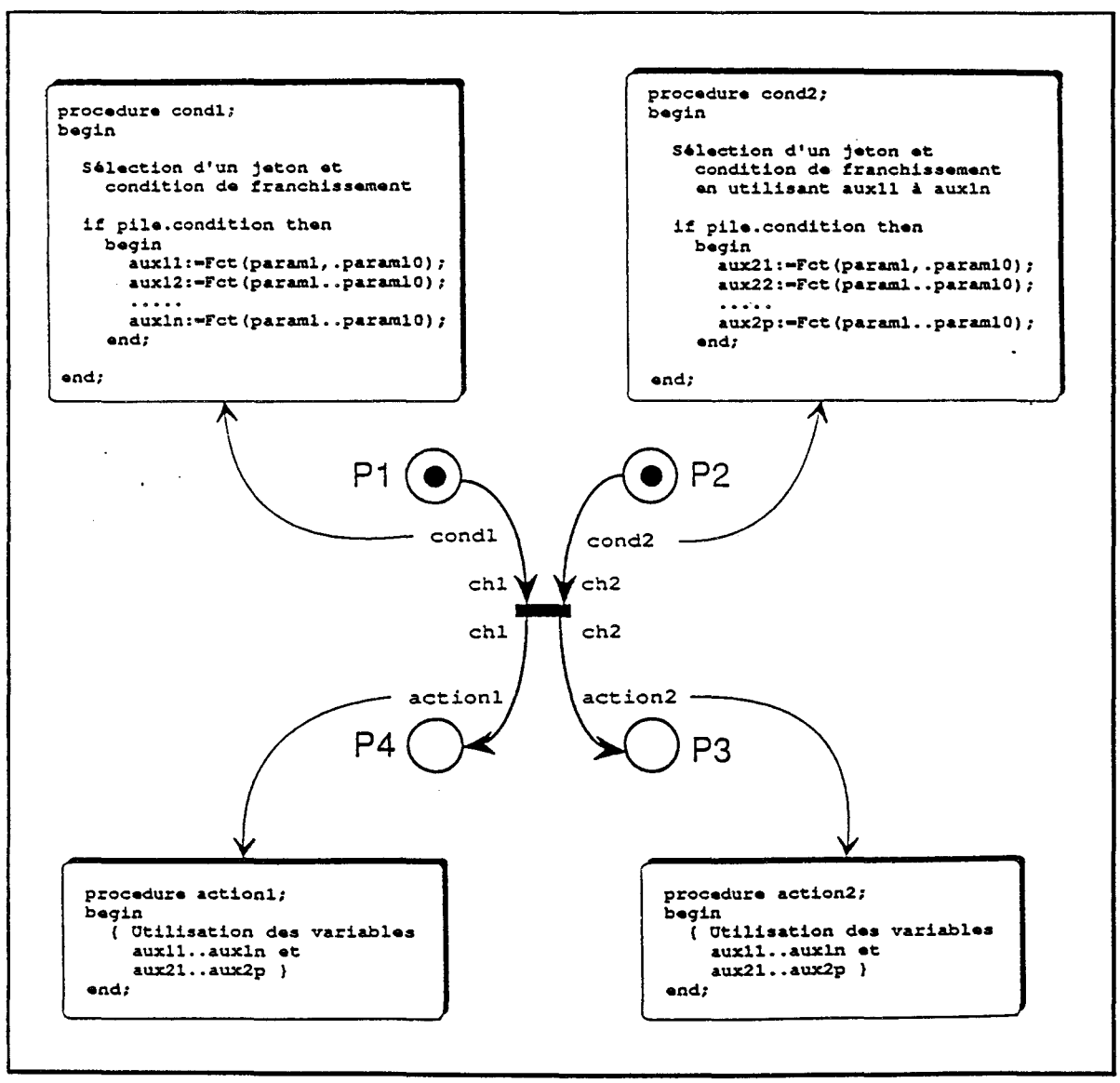


Figure II.34 : *exemple générique d'utilisation de variables locales à une transition*

L'utilisation des chemins ch1 et ch2 permet de s'assurer que la procédure 'cond1' est exécutée en premier, avant 'cond2'. La première partie de la procédure 'cond1' permet de sélectionner le jeton dans la place P1, au moyen de procédure comme 'pvjc', 'npvjc', 'jpi', 'njpi' etc. Si aucun jeton ayant les caractéristiques souhaitées n'est détecté, la variable booléenne 'pile.condition' est à FALSE et la transition n'est donc pas franchissable.

Dans le cas contraire, des valeurs sont stockées dans les variables aux11 à aux1n et la procédure 'cond2' est ensuite exécutée. Elle fonctionne de manière identique à 'cond1' à la différence près qu'elle à accès aux variables aux11 à aux1n pour sélectionner les jetons dans P2. Il devient donc possible d'associer un jeton de P1 à un autre de P2 en fonction des valeurs de leurs attributs. Si un jeton satisfaisant aux contraintes est disponible dans la place P2, la variable 'pile.condition' est à TRUE et les valeurs des variables aux21 à aux2p sont donc calculées.

Lorsque la transition est tirée, les procédures 'action1' et 'action2' sont exécutées. Celles-ci permettent de modifier les paramètres et la couleur des jetons qui vont marquer respectivement les places P4 et P3. Pour réaliser ces affectations, on dispose des paramètres du jeton qui suit le chemin affecté à l'arc mais aussi des valeurs des variables aux11 à aux1n et aux21 à aux2p. On a donc une vision globale de tous les paramètres qui interviennent dans le tir de la transition.

Afin de montrer l'utilisation de cette structure, nous allons présenter deux cas simples. Le premier est une primitive de lecture de message, où il s'agit de réaliser le lien entre le numéro de la station qui est le premier paramètre du jeton d'état et le numéro du destinataire du message qui est le premier paramètre du jeton de message.

On utilise donc une variable baptisée 'NoStation', qui reçoit une valeur dans la procédure 'cond1'. La procédure 'select_msg' sélectionne donc dans la place de communication le jeton message le plus ancien (gestion FIFO) dont le premier paramètre contient la même valeur que celle qui a été affectée à la variable 'NoStation'. Si un tel jeton existe, les données véhiculées par son deuxième paramètre sont stockées temporairement dans la variable 'Donnee' afin de pouvoir être récupérées par le jeton d'état au moyen de la procédure 'action1'.

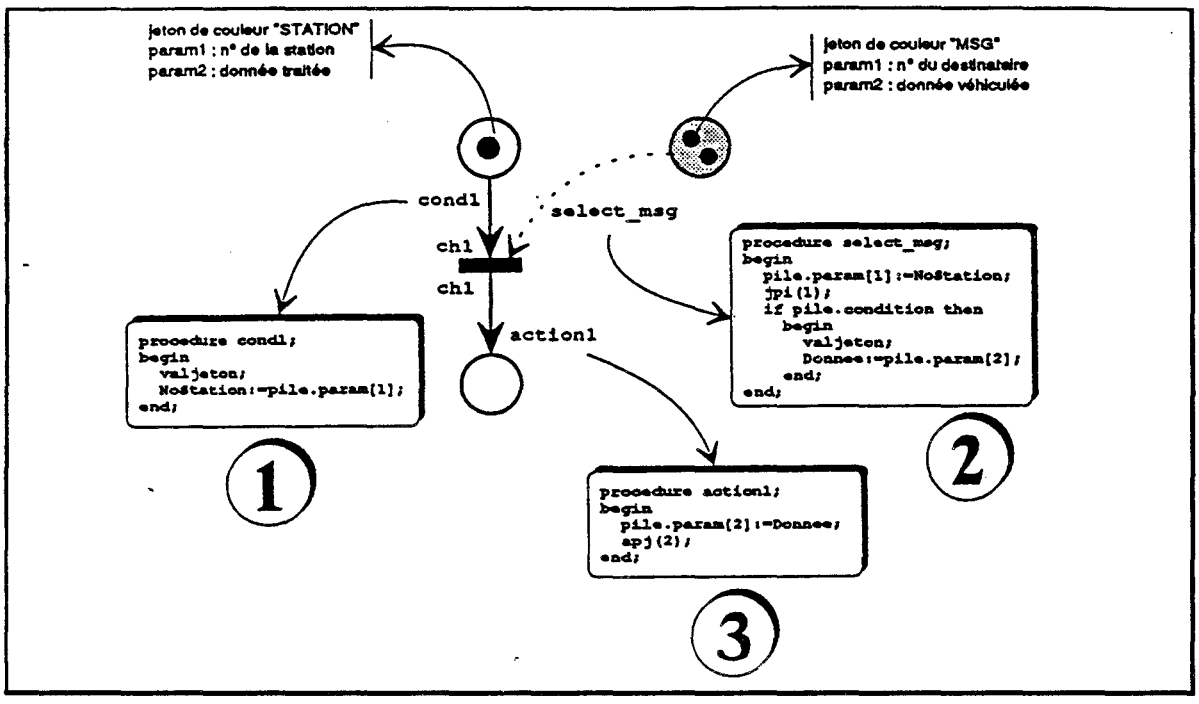


Figure II.35 : *exemple d'utilisation de variables locales à une transition*

Le second exemple proposé (Cf. figure II.36) est une primitive d'envoi de message où il s'agit de créer le jeton de message avec les attributs symbolisant le numéro du destinataire et les données véhiculées. La procédure '`cond1`' permet de stocker dans la variable '`Donnée`' la valeur du deuxième paramètre du jeton d'état qui symbolise les données traitées. Cette valeur peut ensuite être utilisée par la procédure '`creation_msg`' pour affecter le deuxième paramètre du jeton message créé.

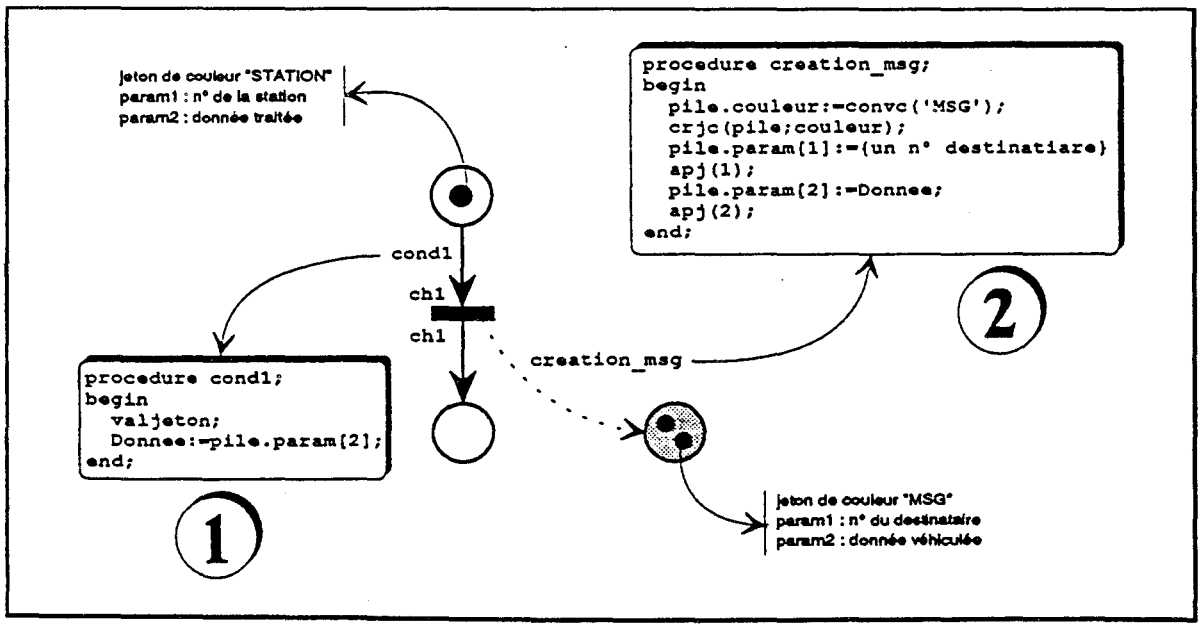


Figure II.36 : *exemple d'utilisation de variables locales à une transition*

II.3. Scénario - Environnement de simulation

II.3.1. Enjeux

A partir des modèles d'équipements d'automatisme que nous nous proposons d'élaborer selon les principes qui viennent d'être détaillés, il devient possible, par assemblage d'objets, de construire un modèle complet du point de vue des communications de l'aspect matériel d'une architecture de commande et de pilotage. Pour être en mesure d'évaluer les performances des communications d'un tel système, il est indispensable de le compléter par une image réaliste des échanges de données sur les réseaux. Cette approche consiste à définir des scénarios de charge de l'architecture, les caractéristiques des communications étant spécifiques pour chaque nouvelle application étudiée.

II.3.2. Description des applicatifs

Pour chaque station d'une architecture de commande, il est nécessaire de décrire les procédures d'émissions/réceptions de messages et données sur le ou les réseaux sur lesquels l'équipement est connecté. Il s'agit de représenter la partie communication de la commande implémentée dans l'équipement. Cette modélisation peut se concevoir de deux manières, avec des niveaux de détails différents.

Une première approche, que nous qualifierons de *macroscopique*, consiste à utiliser des lois probabilistes pour décrire l'émission des données. Pour chaque station, on définit la liste des messages que l'équipement est susceptible d'envoyer sur les réseaux. Le scénario d'émission de chaque message est alors assimilé à une loi aléatoire (loi de Poisson, loi de Gauss etc.). Si, pour certains messages, la taille des données émises est variable, la détermination de ce paramètre dans les simulations peut s'appréhender de la même façon.

Une seconde manière de procéder est d'adopter une approche plus *microscopique* qui consiste, pour chaque station, à détailler précisément le séquencement et le contenu des données échangées au niveau de son applicatif. Dans ce cas chaque station concernée doit pouvoir détecter la présence et les informations des messages dont elle est destinataire. Ceci permet de conditionner les traitements de son applicatif en fonction de l'évolution des autres stations de l'architecture et donc de modéliser aussi finement que nécessaire le comportement du système complet.

II.3.3. Image de l'environnement

Certains échanges sur une architecture de commande, comme par exemple des remontées d'informations d'état vers un système de supervision, ont lieu de manière régulière. Par contre, d'autres communications sont provoquées par des événements externes au système de commande. En effet, une architecture de contrôle d'un système automatisé de production réagit à des événements en provenance du process qu'il contrôle ainsi qu'à des consignes transmises par un niveau supérieur (cadencement par exemple). Après acquisition et traitement de ces informations le système de commande envoie des consignes vers le procédé pour le faire évoluer. La détermination de ces consignes est réalisée par les programmes de contrôle/commande implantés dans les équipements de l'architecture.

Dans ce cas, il est donc important d'incorporer dans les modèles une représentation réaliste de ces phénomènes, qui conditionnent les échanges sur les réseaux de communication. A nouveau deux approches sont possibles. La première consiste à représenter la génération des événements transmis à la commande selon une loi probabiliste. La seconde est de décrire les séquencements d'opérations du procédé ou de l'environnement qui conditionnent le déclenchement d'événements vers la commande. Dans le cadre de cette approche, il est alors également nécessaire de décrire les consignes générées par la partie commande vers le process et qui influent sur l'évolution de ce dernier. De cette manière, on modélise les interactions mutuelles entre l'architecture et son environnement.

II.3.4. Conclusion

La figure II.37 résume l'esprit selon lequel nous proposons d'appréhender la description des échanges de messages sur une architecture de commande. Lors d'une mise en oeuvre effective d'une approche de modélisation/simulation, on fera dans la majorité des cas appel à une solution hybride. En effet, cela dépendra des paramètres que l'on souhaite observer, de la précision des résultats attendus, et des informations dont on dispose au moment de la construction des modèles.

Dans une première étape (phase 1 de la figure I.15), la description des applicatifs sera macroscopique car on n'a qu'une vision globale des flux de données. Au cours de l'avancement du projet la description des séquencements des communication se précisent et il devient donc possible d'adopter un approche microscopique.

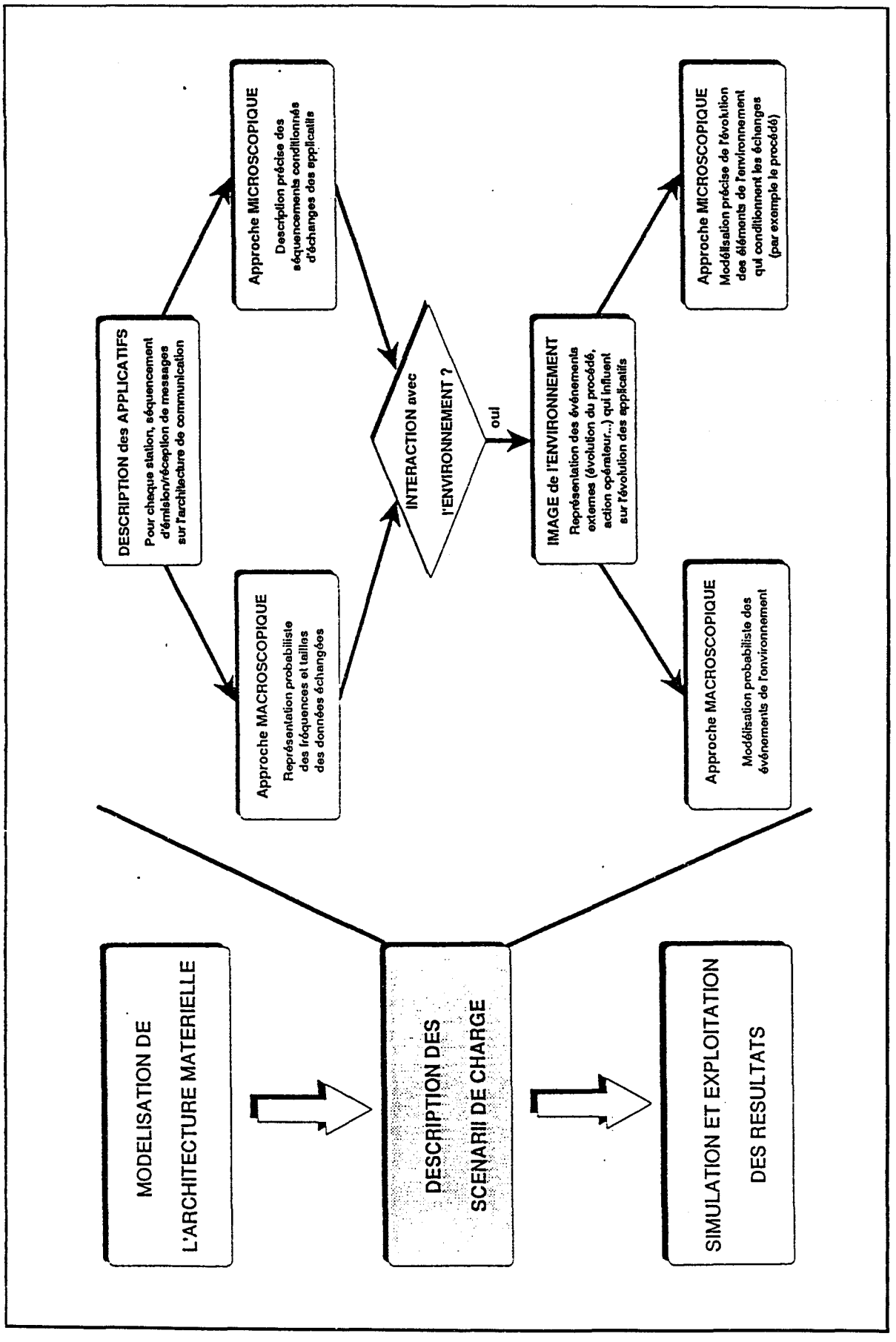


Figure II.37 : *Description des scénarios de charge d'une architecture de commande*

II.4. Limitations du formalisme de SEDRIC

Le formalisme réseaux de Petri proposé par l'outil SEDRIC, nous paraît, comme nous l'avons vu, intéressant pour aborder la modélisation et l'évaluation de performances d'architecture de commande d'atelier. Cependant, il reste limité sur quelques points qu'il est important de mentionner.

II.4.1. Interprétation au niveau de la transition

Dans SEDRIC, l'interprétation du formalisme réseaux de Petri est implémentée au travers de procédures Pascal associées aux arcs (Cf. annexe 1). Comme nous l'avons vu, cette approche est contraignante dans sa mise en oeuvre car elle ne permet pas d'avoir une vision locale à une transition, des jetons qui interviennent dans son franchissement : comme les procédures sont attachées aux arcs, on a directement accès uniquement aux jeton de la place (amont ou aval) attachée à l'arc. C'est la raison pour laquelle, il est apparu nécessaire d'avoir recours à l'utilisation de variables auxiliaires (Cf. § II.2.5.c).

Dans le formalisme des réseaux de Petri à structures de données, tel qu'il est défini par Sibertin-Blanc [SIB 85] (Cf. annexe 1), des variables sont utilisées au niveau des arcs pour instancier les jetons qui interviennent dans les franchissements. Les identifiants associés à tous ces jetons sont disponibles pour exprimer les préconditions et actions associées aux transitions. Une voie d'étude intéressante serait donc de regarder dans quelle mesure ce formalisme, au demeurant plus contraignant dans sa mise en oeuvre que celui employé (structure de données à définir de manière plus rigoureuse), pourrait être utilisé pour construire les modèles qui ont été détaillés dans ce mémoire.

II.4.2. Modélisation du mécanisme de préemption

II.4.2.a Principe

Le mécanisme de préemption est utilisé en informatique, notamment dans les systèmes multitâches pour gérer l'attribution du CPU entre les process [BEA 91]. Lorsque le processeur est utilisé par une tâche, et qu'une autre, de priorité supérieure passe dans un état qui requiert le CPU, le système d'exploitation peut décider de retirer momentanément le processeur à la première tâche pour l'attribuer à la seconde. On dit alors qu'il y a eu préemption du CPU.

Modéliser, avec le formalisme des réseaux de Petri, le partage d'une ressource se fait aisément au moyen du mécanisme d'exclusion mutuelle. Pour représenter la durée associée à l'utilisation de cette ressource, on utilise les activités associées aux places : une place est temporisée avec une durée égale au temps d'utilisation de la ressource. Mais il n'existe alors aucun moyen pour préempter, sur événement extérieur, cette ressource pour une autre utilisation momentanée et la restituer ensuite.

Pour prendre en compte ce type de phénomène avec des réseaux de Petri temporisés, nous proposons de mettre en oeuvre une méthode de discrétisation : la temporisation associée à l'utilisation de la ressource est discrétisée. A chaque pas, on peut ainsi remettre en cause l'attribution de la ressource, pour éventuellement la préempter.

II.4.2.b Exemple de modélisation

Afin d'illustrer cette approche, considérons le système multitâche suivant : plusieurs process, dont l'exécution est déclenchée par des événements extérieurs asynchrones, se partagent un CPU géré par un 'scheduler', dont la seule règle est d'attribuer le processeur à la tâche demandeur de plus haute priorité. On suppose de plus que tous les processus sont résidents en mémoire et ne requièrent aucune autre ressource que le CPU.

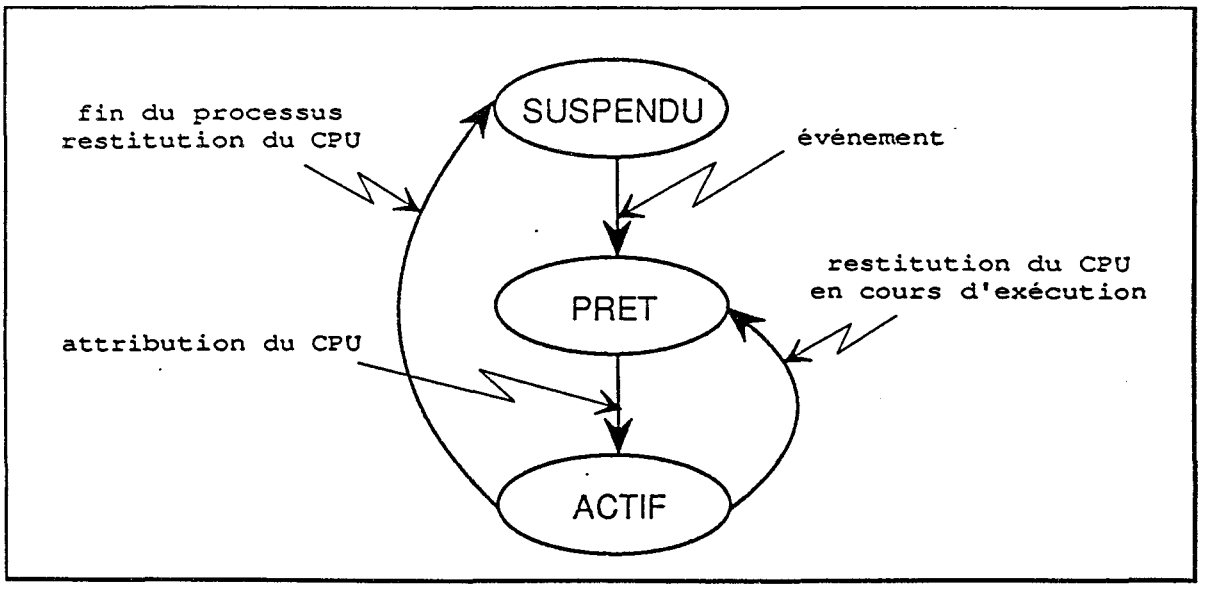


Figure II.38 : *graphe d'état d'un processus*

Un process peut alors se trouver dans les trois états suivants :

- **suspendu** : il est totalement inactif, jusqu'à temps qu'un événement extérieur provoque son réveil et le fasse passer à l'état prêt,

- prêt : il est en attente de CPU pour commencer, ou poursuivre (suite à une préemption de CPU) son exécution,
- actif : le CPU lui est attribué, et il est en cours d'exécution.

Pour représenter le fonctionnement détaillé de ce système multitâche simple, nous utilisons un graphe réseaux de Petri pour décrire le fonctionnement du scheduler, et un graphe pour chaque tâche. La synchronisation entre ces différents processus est réalisée au moyen des places de communication qui sont explicitées sur la figure suivante :

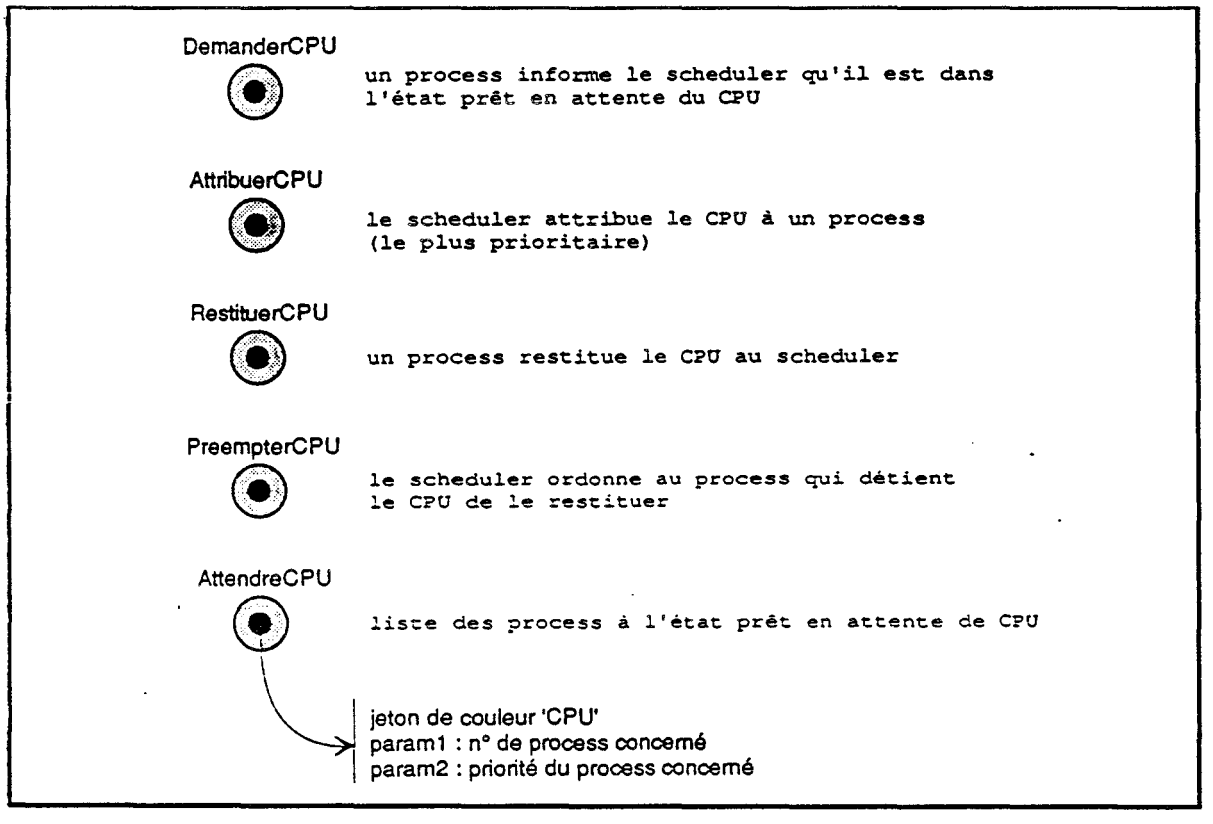


Figure II.39 : *places de communication entre le scheduler et les tâches*

Le fonctionnement d'une tâche est alors décrit par le modèle de la figure II.40. Le marquage de la place Suspendu indique que la tâche est inactive. Sur apparition d'un événement extérieur (jeton dans la place Evenement), la tâche passe dans l'état prêt et informe le scheduler qu'elle sollicite le CPU pour s'exécuter (message dans la place DemanderCPU). Au niveau de détail du modèle, on ne fait pas de distinction sur les appels, qui sont matérialisés par des jetons sans paramètre. La procédure 'CréerDemande' caractérise la requête en spécifiant dans les attributs du jeton créé le numéro et la priorité du process demandeur. La tâche passe ensuite dans l'état associé à la place Pret2, pour attendre que le scheduler lui attribue le CPU.

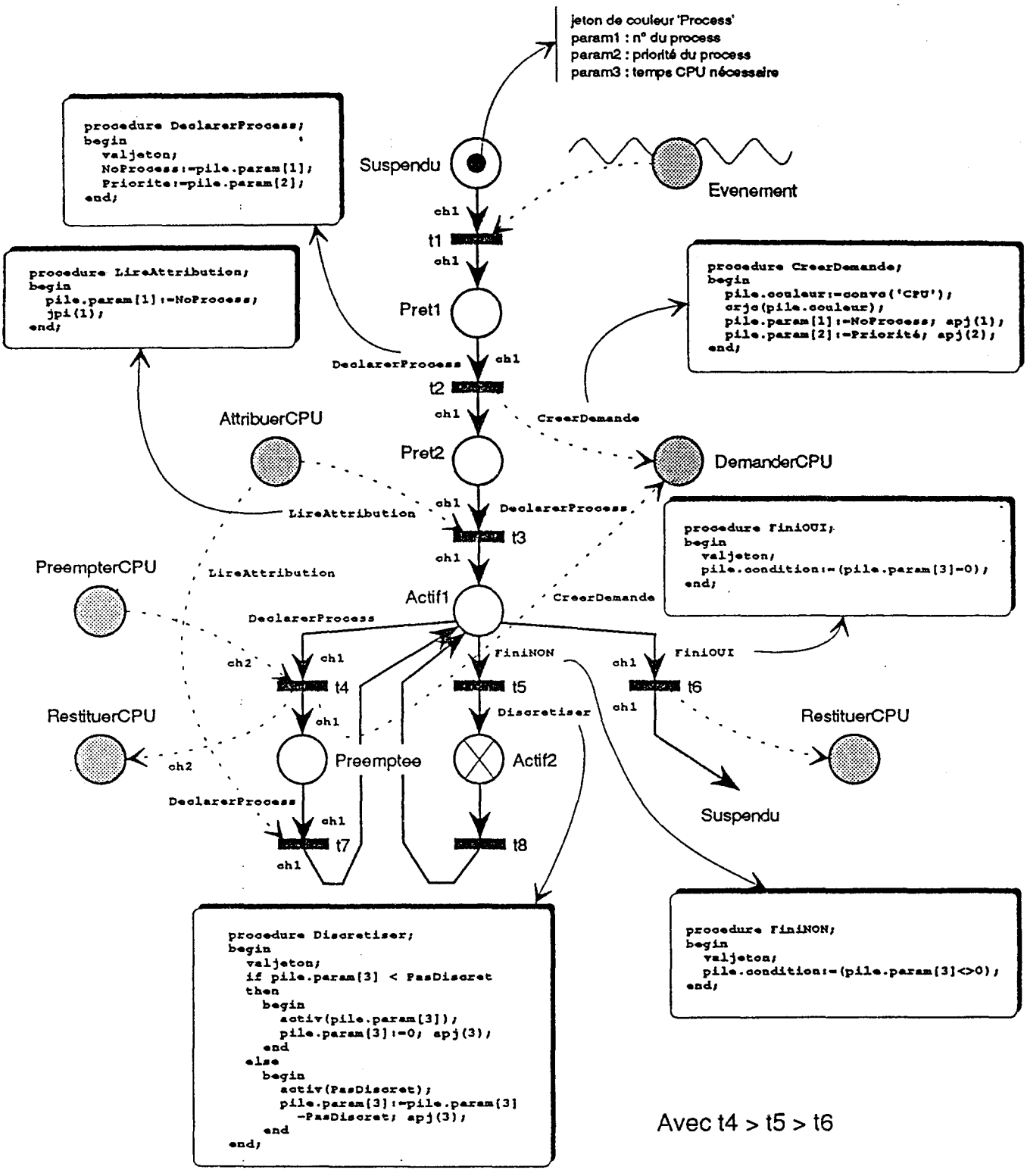


Figure II.40 : *graphe de processus d'une tâche du système*

Lorsqu'un jeton est présent dans la place AttribuerCPU, la transition t3 est sensibilisée. La condition, exprimée dans la procédure '`LireAttribution`', autorise le franchissement dans le seul cas où le process destinataire du message (paramètre 1 du jeton de la place AttribuerCPU) est bien celui considéré. Dans ce cas, t3 est tirée et la tâche passe dans l'état actif.

La tâche doit rester dans cet état actif pendant une durée qui représente le temps nécessaire au process pour s'exécuter. Cette valeur est stockée dans le troisième paramètre du jeton. Une constante, globale au modèle, appelée 'PasDiscret' contient la durée de discrétisation adoptée. Lorsque le CPU n'est pas retiré à la tâche en cours d'exécution, à chaque pas de discrétisation, t5 est tirée, et le jeton d'état marque la place Actif2 pendant une durée égale au temps de discrétisation.

A chaque franchissement, la procédure '`Discretiser`', soustrait au paramètre 3 la valeur du pas de discrétisation. Les procédures '`FiniOUI`' et '`FiniNON`' sont des conditions qui testent la valeur de cet attribut. Tant qu'il est non nul, t5 est franchie. Lorsqu'il vaut zéro, t6 est tirée, la tâche retourne dans l'état suspendu et un message est envoyé au scheduler pour lui indiquer que le CPU n'est plus utilisé (marquage de la place RestituerCPU).

Si lorsque la tâche est active, une autre tâche plus prioritaire, passe dans l'état prêt, le scheduler va préempter le CPU en envoyant un message dans la place PreempterCPU. Dans ce cas, au cours de la discrétisation, lorsque la place Actif1 est marquée, t4 devient sensibilisée. Comme elle est plus prioritaire que t5 et t6, elle est tirée, le CPU est restitué (marquage de la place RestituerCPU) et le process est dans l'état préempté (marquage de la place Preemptee).

Lors du tir de la transition t4, un message marque la place DemanderCPU, pour que le scheduler attribue à nouveau le processeur à la tâche préemptée. Le troisième paramètre du jeton d'état contient le temps CPU encore nécessaire pour terminer l'exécution de la tâche. Lorsque le scheduler lui affecte à nouveau le CPU (tir de la transition t7), le process se retrouve dans le même état qu'avant la préemption et restera donc dans l'état actif, pendant une durée correspondant au temps nécessaire pour terminer son exécution.

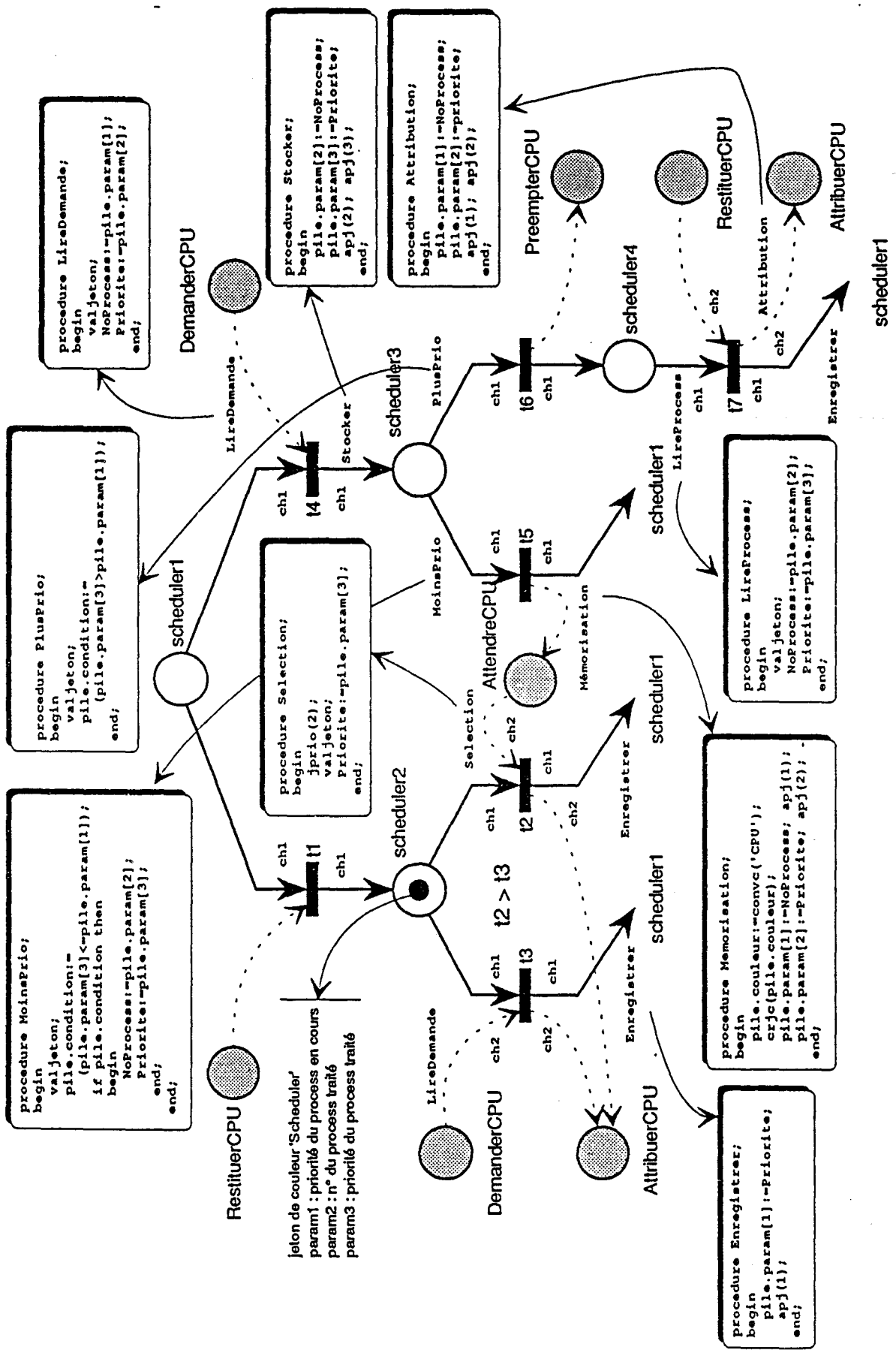


Figure II.41 : *modélisation du fonctionnement du scheduler*

Le modèle du scheduler est présenté en page précédente. Dans le marquage initial, le jeton d'état est dans la place scheduler2, aucun message n'étant présent dans les places de communication. Lorsqu'une tâche demande le CPU (message dans la place DemanderCPU) il lui est attribué et le scheduler passe dans l'état scheduler1. La priorité de la tâche en cours d'exécution est mémorisée dans le premier paramètre du jeton d'état.

Si le CPU est restitué (marquage de RestituerCPU), t1 est tirée et on se retrouve dans le marquage initial. Pendant que la tâche s'exécute, une autre tâche peut demander le CPU. Dans ce cas, la transition t4 est franchie. Si la tâche en cours est d'une priorité plus élevée, la transition t5 est tirée et un message dans la place AttendreCPU traduit le fait qu'une nouvelle tâche est dans l'état prêt. Si la tâche en cours est moins prioritaire, il faut lui retirer le CPU pour l'affecter à la nouvelle. Un message de préemption est donc envoyé, et lorsque le CPU est restitué, il est immédiatement réattribué au process demandeur.

Les process dans l'état prêt, qui sont mémorisés par des jetons dans la place AttendreCPU sont considérés par le scheduler lorsqu'une tâche termine sont exécution et restitue le CPU. Dans ce cas, c'est le process de plus haute priorité qui est sélectionné pour lui attribuer le CPU.

II.4.2.c Propriétés du modèle

Le modèle qui a été développé présente des propriétés intéressantes à observer, qui concernent d'une part les graphes de processus et d'autre part les places de communication.

Comme expliqué dans le paragraphe II.2.5, du fait de la structuration en graphes de processus, chaque graphe est sauf. On a deux invariants de marquage :

- les places du modèle du scheduler, où circule le jeton d'état 'Scheduler' :

  scheduler1, scheduler2, scheduler3 et scheduler4.

- les places du modèle du process, où circule le jeton d'état 'Process' :

  Suspendu, Pret1, Pret2, Actif1, Actif2 et Preemptee.

Par ailleurs toutes les places de communication sont bornées :

- DemanderCPU : lorsqu'un process envoie un message dans la place DemanderCPU (tir de la transition t2 sur la figure II.40), son jeton d'état marque la place Pret2 jusqu'à temps qu'un jeton de la place AttribuerCPU lui soit destiné. Or un tel jeton jeton est créé par le modèle du scheduler en enlevant une marque de

la place DemanderCPU (tir de la transition t3 sur la figure II.41). Donc la place DemanderCPU contient au plus autant de jetons qu'il existe de process modélisés.

- AttendreCPU : un raisonnement analogue à celui qui vient d'être développé montrerait que cette place est bornée par le nombre de process modélisés.
- AttribuerCPU : cette place est bornée à un. En effet, le scheduler crée un jeton dans cette place lors du franchissement d'une des transitions t2 ou t3. Lorsqu'un tel tirage est réalisé, pour que t2 ou t3 soient à nouveau tirées, il faut nécessairement que le jeton d'état du scheduler marque la place scheduler2 donc que t1 ait été franchie. Or pour que ce tirage soit possible, il faut qu'un process restitue le CPU donc nécessairement qu'il l'ai pris au préalable en utilisant le jeton de la place AttribuerCPU.
- RestituerCPU : par un raisonnement analogue à celui mené pour la place Attribuer CPU, on montre que cette place est également bornée à un.
- PreempterCPU : cette place est bornée à un. En effet, lorsqu'elle est marquée, le jeton d'état du scheduler se trouve dans la place scheduler4 en attente d'une marque dans la place RestituerCPU. L'unique process actif, ne peut créer une marque dans cette place que par le franchissement des transitions t4 ou t6 de la figure II.40. Comme t4 est plus prioritaire, elle est tirée ce qui supprime le jeton de la place PreempterCPU.

II.4.2.d Conclusion

Cet exemple, montre comment représenter, dans un modèle temporisé, un mécanisme de préemption. Le modèle proposé constitue une approche générique pour traiter des systèmes où plusieurs process se partagent le même processeur. L'approche de discrétisation qui a été développée doit pouvoir être étendue pour des systèmes plus complexes, comme par exemple ceux qui mettent en oeuvre des mécanisme de swapping [BEA 91] avec une unité de sauvegarde de masse.

Il est important de remarquer que pour que le modèle soit suffisamment représentatif de la réalité, il est nécessaire, lors des simulations, d'utiliser un pas de discrétisation suffisamment petit par rapport aux durées d'exécution des tâches. Ceci a pour conséquence d'augmenter fortement le nombre d'événements dans les simulations, et donc leur durée.

III. APPLICATION A DES EQUIPEMENTS INDUSTRIELS

Dans le paragraphe précédent, nous nous sommes attachés à détailler les règles d'utilisation des réseaux de Petri, pour construire des modèles. Nous allons maintenant montrer comment ces principes ont été appliqués pour la modélisation d'équipements industriels typiques utilisés par le groupe PSA Peugeot Citroën. Nous avons principalement travaillé sur des matériels de la société Télémécanique car ceux-ci constituent une part importante des équipements mis en place dans le groupe. De plus, nous avons, aux cours de nos travaux, pu obtenir les informations sur le fonctionnement des matériels étudiés, nécessaires pour construire les modèles. Nous avons cherché essentiellement à exprimer sur des cas concrets les caractères génériques à chaque équipement modélisé (automates programmables, réseaux locaux industriels etc.)

III.1. Modélisation d'un réseau local industriel

Les réseaux locaux industriels (RLI), utilisent des technologies proches de celles employées pour les réseaux informatiques d'entreprise [LEP 91]. Cependant, ils doivent présenter des caractéristiques "temps réel" garantissant des délais d'acheminement. Ils sont en effet utilisés pour synchroniser des tâches réparties et doivent donc de ce fait offrir une grande disponibilité. Les composants et les protocoles utilisés doivent tenir compte d'un fonctionnement en environnement perturbé, liés aux ateliers dans lesquels ils sont mis en place. Les performances de ces matériels de communication doivent pouvoir être évaluées afin de valider leur comportement à caractère temps réel.

Les débits sont faibles, dans la plupart des cas quelques dizaines à quelques centaines de Kbs/s. Les nouveaux produits atteignent des débits de 5 à 10 Mb/s, cette tendance étant liée à l'alignement sur les normes OSI [RUD 86]. Modéliser un réseau de communication consiste à modéliser le protocole qu'il implémente. Il est donc nécessaire d'en connaître les spécifications. Le modèle est élaboré en suivant la décomposition en 7 couches de l'ISO. Les temporisations sont pour certaines fournies par le protocole et pour d'autres sont liées aux matériels (coupleur, carte de communication etc.) employés.

III.1.1. Réseau Uni-Telway de Télémécanique

Uni-Telway est un bus industriel multipoint hétérogène qui assure une communication entre les divers constituants d'automatisme de l'offre Télémécanique, ainsi

qu'une ouverture vers des équipements tiers. Une analogie peut-être établie entre l'architecture de communication utilisée par Uni-Telway et le modèle de référence OSI.

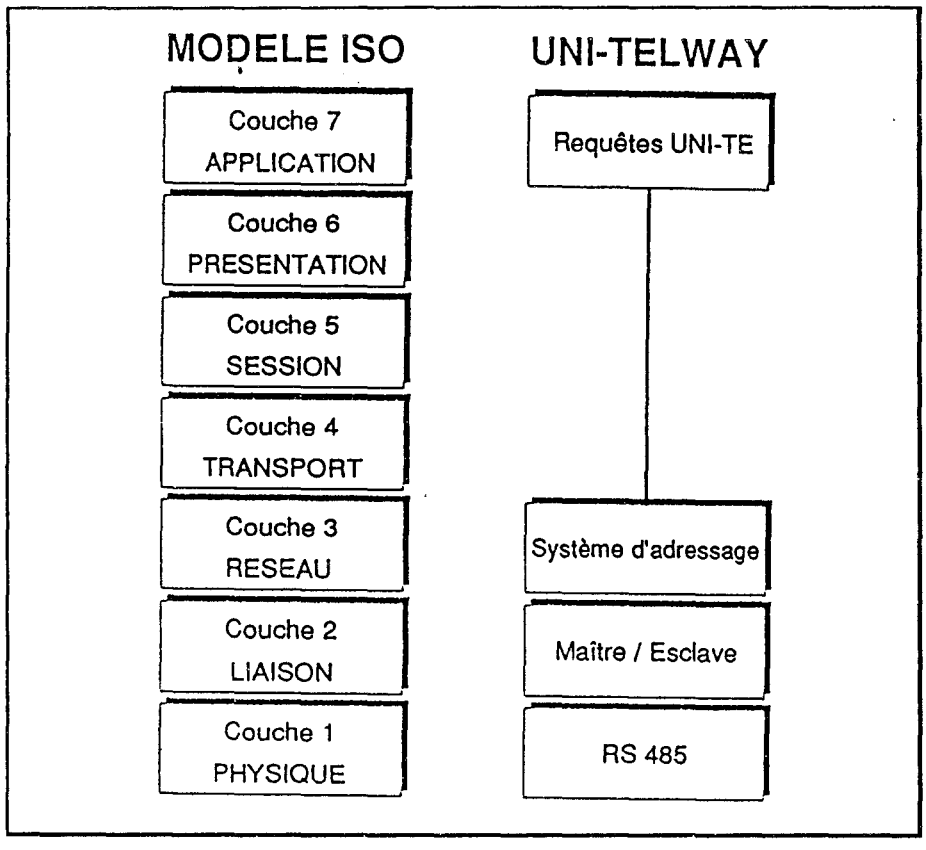


Figure II.42 : *analogie entre le protocole Uni-Telway et le modèle OSI*

I.1.1.a Couche application

La couche application offre une interface unique de communication pour l'ensemble des équipements conformes au protocole UNI-TE. Cette couche offre à l'utilisateur un ensemble de services standard pouvant être complétés par des services spécifiques aux automates programmables, aux commandes numériques etc.

La mise en oeuvre de ces services s'effectue par un mécanisme de question/réponse appelé requête/compte-rendu. Un équipement peut avoir les statuts suivants :

- CLIENT : c'est l'équipement qui prend l'initiative de la communication; il pose une question (lecture), transmet une information (écriture) ou envoi un ordre (run, Stop etc.).
- SERVEUR : c'est l'équipement qui réalise l'ordre du client.

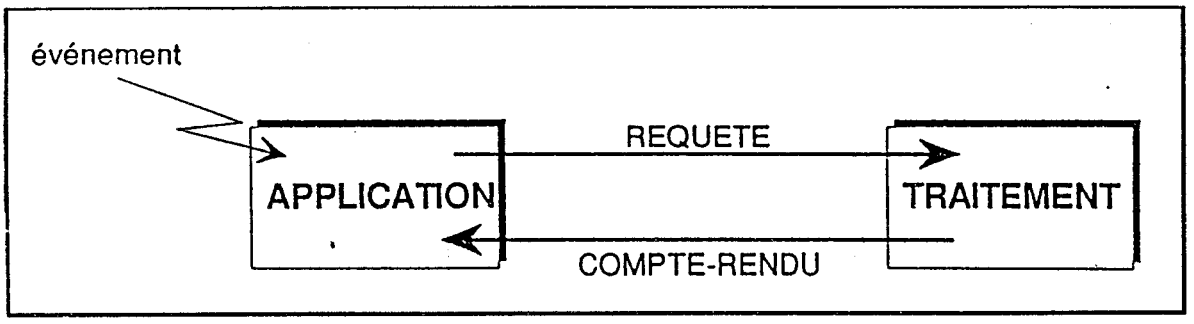


Figure II.43 : *fonctionnement client/serveur de la couche application*

Les requêtes UNI-TE offrent principalement les services suivants :

- lecture/écriture d'objets (bits, mots etc.),
- gestion des modes de marche (INIT, RUN, STOP),
- diagnostic bus et équipement,
- chargement et déchargement de fichier et programme.

Pour la construction de la trame, la partie TPDU (Transport Protocol Data Unit) comporte les rubriques suivantes :

Requête : TPDU = <code_requête> <code_catégorie> <données>

Le code_requête indique le service utilisé.

Le code_catégorie représente la catégorie de l'émetteur.

Les données sont les paramètres éventuels associés à la requête.

Compte-rendu : TPDU = <code_réponse> <données>

Le code_réponse indique le compte-rendu application de la requête.

Les données sont les paramètres éventuels associés à la réponse.

A titre d'illustration, la figure II.44 présente le format des TPDU associés à la requête de lecture de mot :

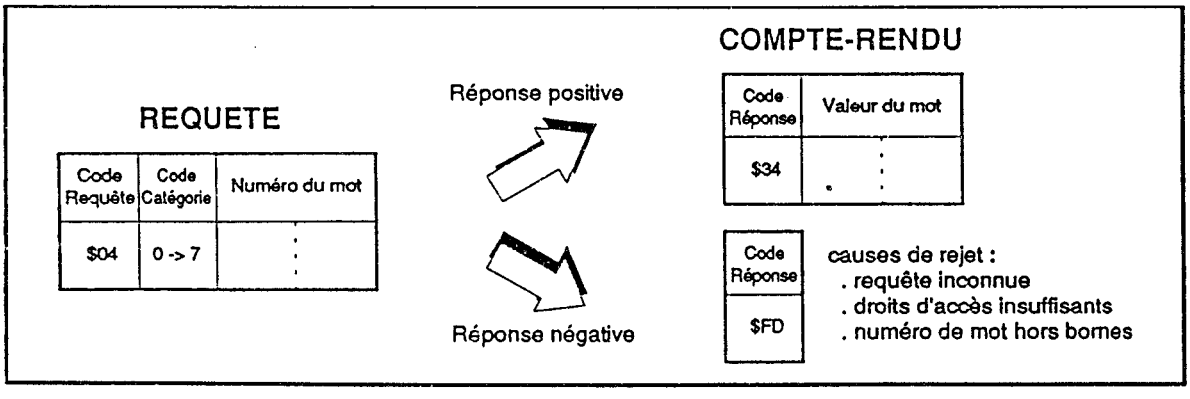


Figure II.44 : *format du service requête/compte-rendu de lecture de mot*

I.1.1.b Couche réseau

La couche réseau gère l'exploitation du réseau. Elle assure les fonctions de routage de l'émetteur vers le ou les destinataires, chaque interlocuteur étant identifié par une adresse réseau unique.

Le service d'adressage standard, qui est basé sur l'architecture des automates programmables TSX série 7 est hiérarchisé sur 5 niveaux, nécessitant chacun un octet :

- numéro de RESEAU,
- numéro de STATION,
- numéro de PORTE,
- numéro de MODULE,
- numéro de VOIE.

Une explication plus détaillée de la philosophie de cette organisation peut être consultée dans [TEL 88a].

Pour la construction de la trame, la partie NPDU (Network Protocol Data Unit) comporte pour l'adressage standard les rubriques suivantes :

NPDU = <type> <adresse_distante> <TPDU>

avec <adresse_distante> = <réseau> <station> <porte> <module> <voie>
et type = 20H pour l'adressage standard.

I.1.1.c Couche liaison [TEL 87]

La couche liaison de données transforme la transmission de bits en vrac en une ligne de communication qui, vue des couches supérieures, apparaît exempte d'erreurs. Son rôle est donc d'acheminer correctement des trames sur le bus. Le protocole implémenté au niveau de cette couche conditionne de manière importante les performances du réseau.

A. PRINCIPES DE FONCTIONNEMENT :

La couche liaison est asymétrique, avec un maître fixe et un ou plusieurs esclaves. Elle ne permet que les communications entre le maître et un esclave. C'est seulement par l'intermédiaire du routage au niveau réseau que la communication entre deux esclaves devient possible.

Le maître a l'initiative des échanges qui sont soit un message du maître vers l'esclave (*selecting*), soit l'invitation (*polling*) par le maître d'un esclave à émettre. L'ordre de polling est celui des numéros de station croissants. L'ordre de selecting est celui d'arrivée des messages à destination des esclaves. Nous pouvons résumer la gestion du bus par le maître ainsi :

```
TANT QUE VRAI
    POUR adresse_esclave:=1 A adresse_esclave_max FAIRE
        POLLING (adresse_esclave)
        SI j_ai_un_message_a_émettre ALORS
            SELECTING (destinataire)
    FinFAIRE
FinTANTQUE
```

Le maître de la liaison doit interroger périodiquement chaque esclave, même s'il n'attend pas de réponse (un esclave a le droit de poser une question). Tous les esclaves ont un même droit à la parole. Il est essentiel que le maître effectue un polling après chaque émission d'un message pour garantir ce droit à la parole et éviter ainsi les blocages du système de communication.

L'esclave subit la ligne. Il est constamment à l'écoute de celle-ci afin de sélectionner les trames qui lui sont destinées. Lorsque l'esclave a une message à émettre, il doit attendre que le maître l'interroge pour pouvoir l'envoyer. Nous pouvons résumer le travail de l'esclave de la façon suivante :

```
TANT QUE VRAI FAIRE
    écouter_la_ligne
    SI le_maître_m_invite_à_émettre ALORS
        SI j_ai_un_message_à_émettre
            ALORS émission_du_message
            SINON répondre_rien_à_émettre
FinTANTQUE
```

B. FORMAT DES TRAMES :

Un polling est une trame formée de trois caractères : <DLE> <ENQ> <adresse_liaison>
La réponse de l'esclave peut être :

. une information signifiant qu'il n'a pas de message à envoyer; sa réponse est alors une trame constituée du seul caractère <EOT>

. un message (requête ou compte-rendu) à émettre vers une autre adresse; sa réponse est alors une trame LPDU (Link Protocol Data Unit) qui à la forme suivante :

<DLE> <STX> <adresse_liaison> <longueur> <NPDU> <BCC>

Le maître acquitte ce message par un <ACK> ou éventuellement un <NACK> s'il ne peut pas le traiter faute de ressource.

Un message de selecting est constitué de (Cf. figure II.45):

. 2 caractères d'en-tête : <DLE> <STX>

. 1 caractère correspondant à l'adresse liaison du destinataire du message,

. 1 caractère indiquant la longueur du message,

. le message proprement dit qui comprend :

. l'adresse distante (1 octet de type et 5 octets d'adresse),

. le code requête suivi des données (1 à 128 caractères),

. un caractère de contrôle BCC.

Sur un selecting, la réponse de l'esclave peut être :

. une information signifiant que le message a été bien reçu : <ACK>

. une information signifiant que le message a été bien reçu, mais que faute de ressource il ne sera par traité : <NACK>

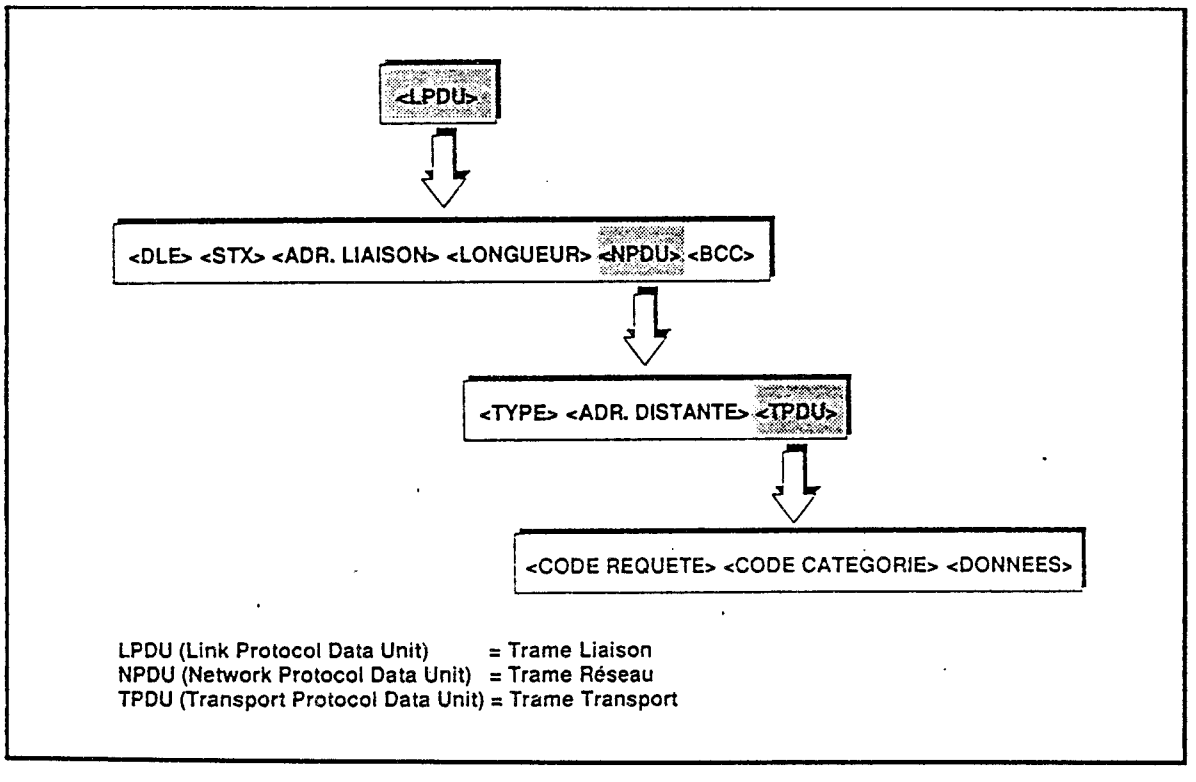


Figure II.45 : *décomposition du contenu d'une trame de message*

Remarques :

. Lorsqu'il s'agit d'une trame émise par le maître, le champ <adr_liaison> contient le numéro de l'esclave à qui la trame est destinée. Le maître a également la possibilité d'émettre des messages en diffusion; le champ <adr_liaison> contient alors 255. Dans ce cas chaque esclave réceptionne le message mais ne répond pas.

. Dans le cas d'un message émis par un esclave, le champ <adr_liaison> contient le numéro de l'esclave émetteur de la trame.

. Le caractère DLE ayant une signification particulière, pour éviter toute confusion, on opère ainsi (on parle de bourrage par DLE) :

. si le champ <long> vaut DLE, il est doublé

. tout caractère DLE dans le champ <NPDU> est dupliqué,

. les champs <adr_liaison> et <BCC> ne sont jamais doublés.

Ceci a pour effet d'allonger la longueur de la trame.

C. SEQUENCEMENT DES ECHANGES

Le protocole maître est détaillée sur les figures II.46 et II.47. La première décrit la partie envoie de message alors que la seconde traite le polling. Le schéma II.48 présente le protocole vu de l'esclave.

Nous voyons qu'un contrôle du flux est nécessaire, car un équipement récepteur d'un message peut ne pas avoir de buffer disponible ou suffisamment grand. Dans ce cas, il attend la fin du message, contrôle le BCC, et si le message reçu est correct émet un <NACK>. Si le message est incorrect, il ne répond rien.

Au niveau du réseau Uni-Telway, la détection des erreurs est multiple :

- **erreur caractère :** format de caractère incorrect, parité caractère incorrecte, ou écrasement caractère en réception,
- **erreur trame :** erreur de BCC, champ <longueur> incorrect ou dépassement du temps inter-caractère,
- **erreur protocole :** erreur de séquencement (réponse inattendue ou incohérente) ou temps enveloppe dépassé (sur émission de message ou de polling, le maître ne reçoit pas de réponse).

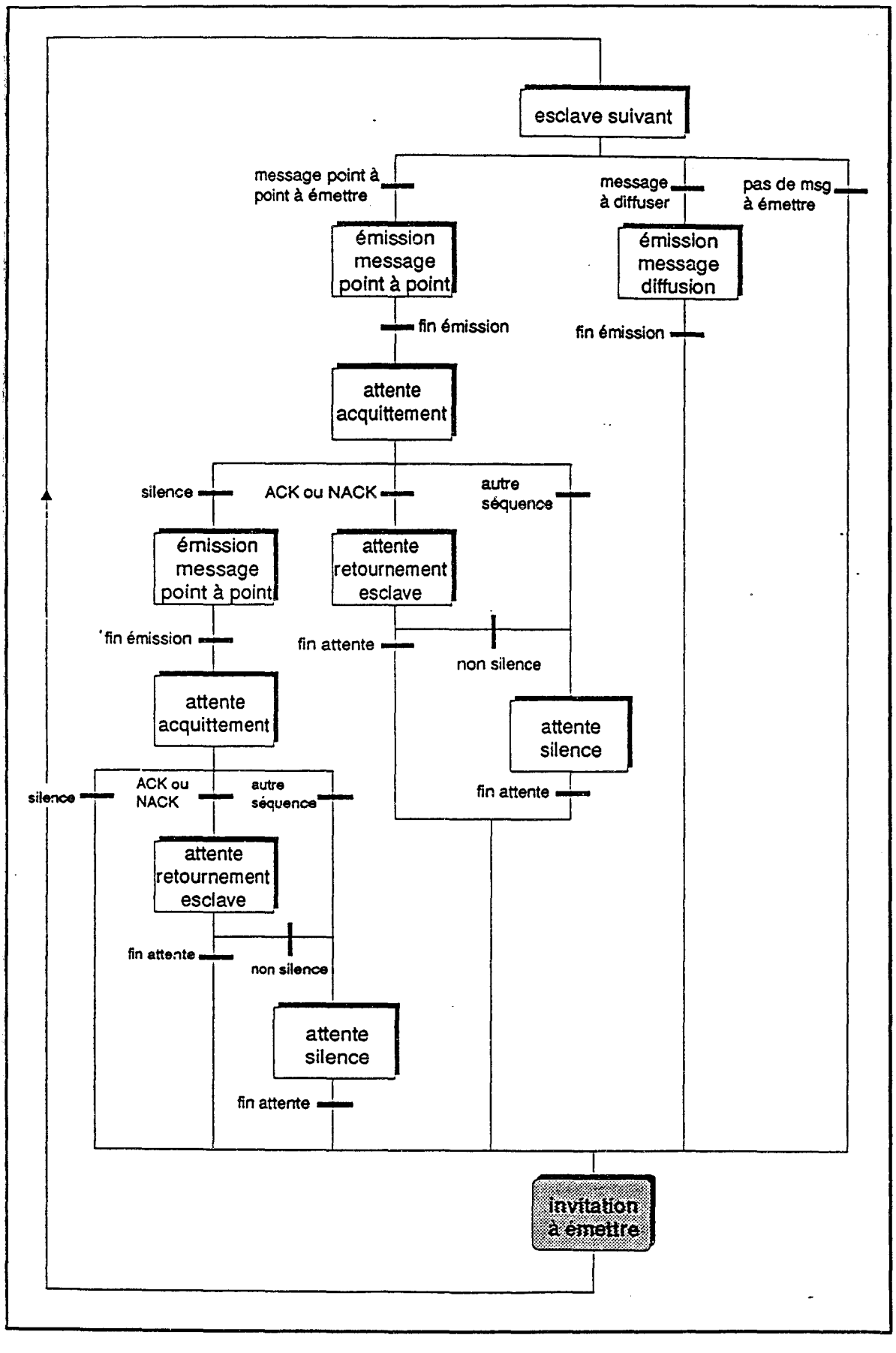


Figure II.46 : *protocole maître Uni-Telway (envoi de message)*

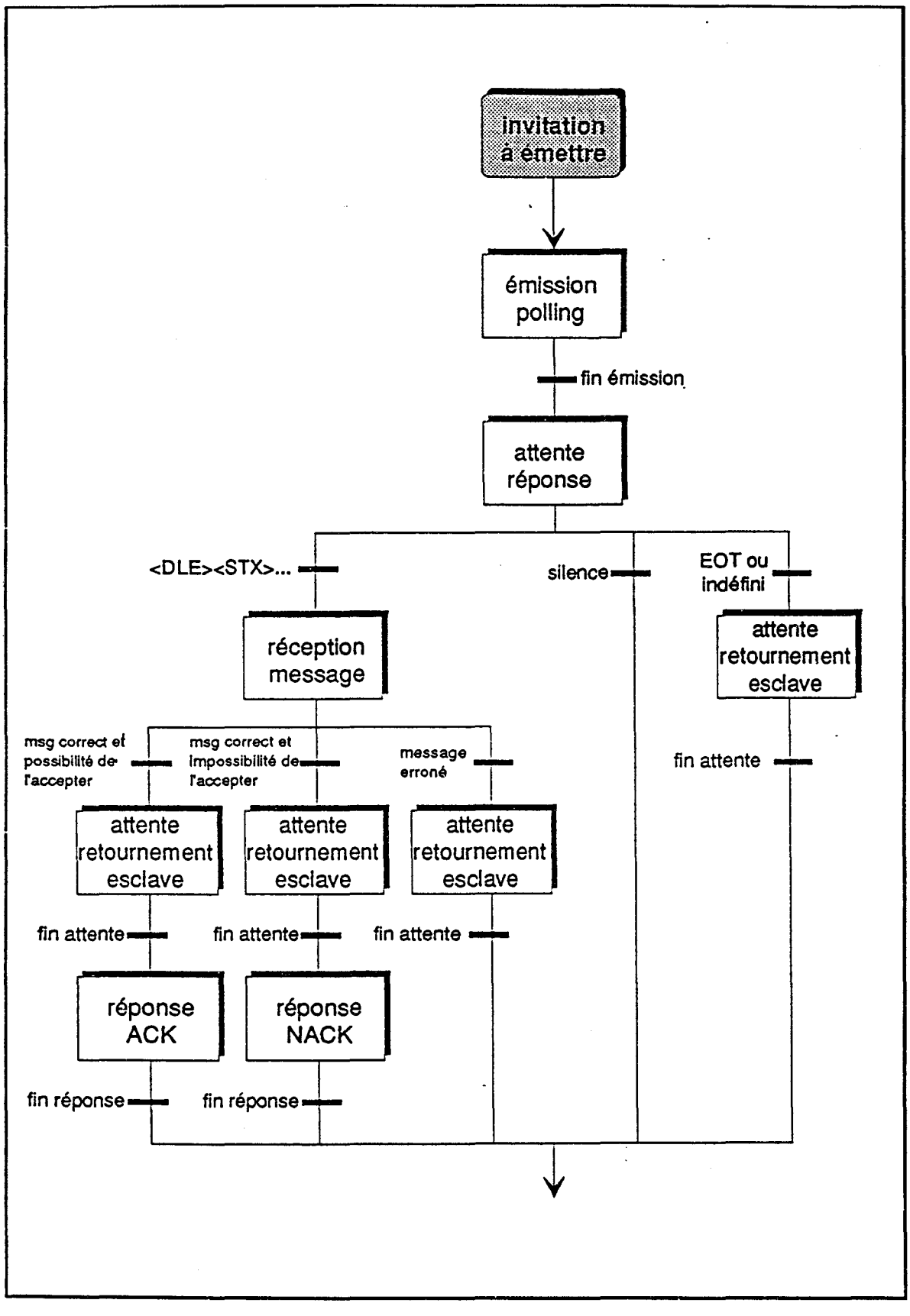


Figure II.47 : *partie "invitation à émettre" (polling) du protocole maître*

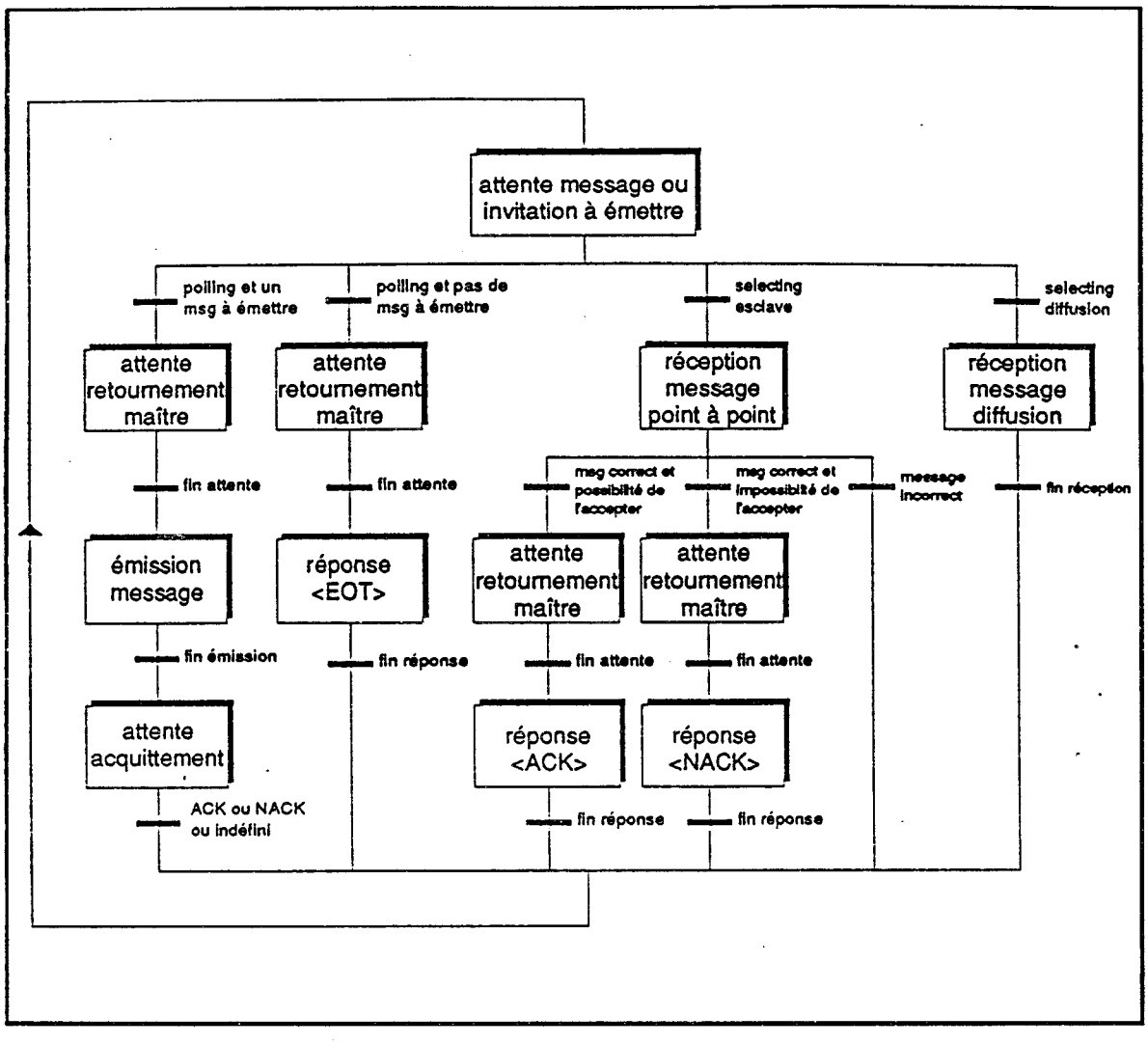


Figure II.48 : *protocole esclave Uni-Telway*

Les erreurs sont traitées de manière différente par le maître ou un esclave :

- **cas du maître** : sur absence de réponse (silence), le maître réémet immédiatement le message. Au second essai négatif, le message est perdu. Dans le doute (réception d'un caractère incorrect), le maître s'abstient de réémettre, pour éviter la duplication du message.

- **cas de l'esclave** : après émission d'un message, l'esclave écoute la ligne : si le premier caractère est <ACK>, le message a été bien reçu. En l'absence de réponse (silence), l'esclave réémettra le même message au prochain polling. Au second essai négatif, le message sera perdu. Dans les autres cas, l'esclave s'abstient de réémettre pour éviter la duplication du message.

D. EXEMPLE

Considérons l'exemple de la figure II.49. Trois automates sont connectés sur un réseau Uni-Telway. Par configuration, le coupleur de l'automate 1 est le maître de la liaison chacun des autres automates dispose de deux adresses esclave : une adresse serveur (Ad0) pour répondre à des requêtes et une adresse client (Ad1) pour envoyer des requêtes.

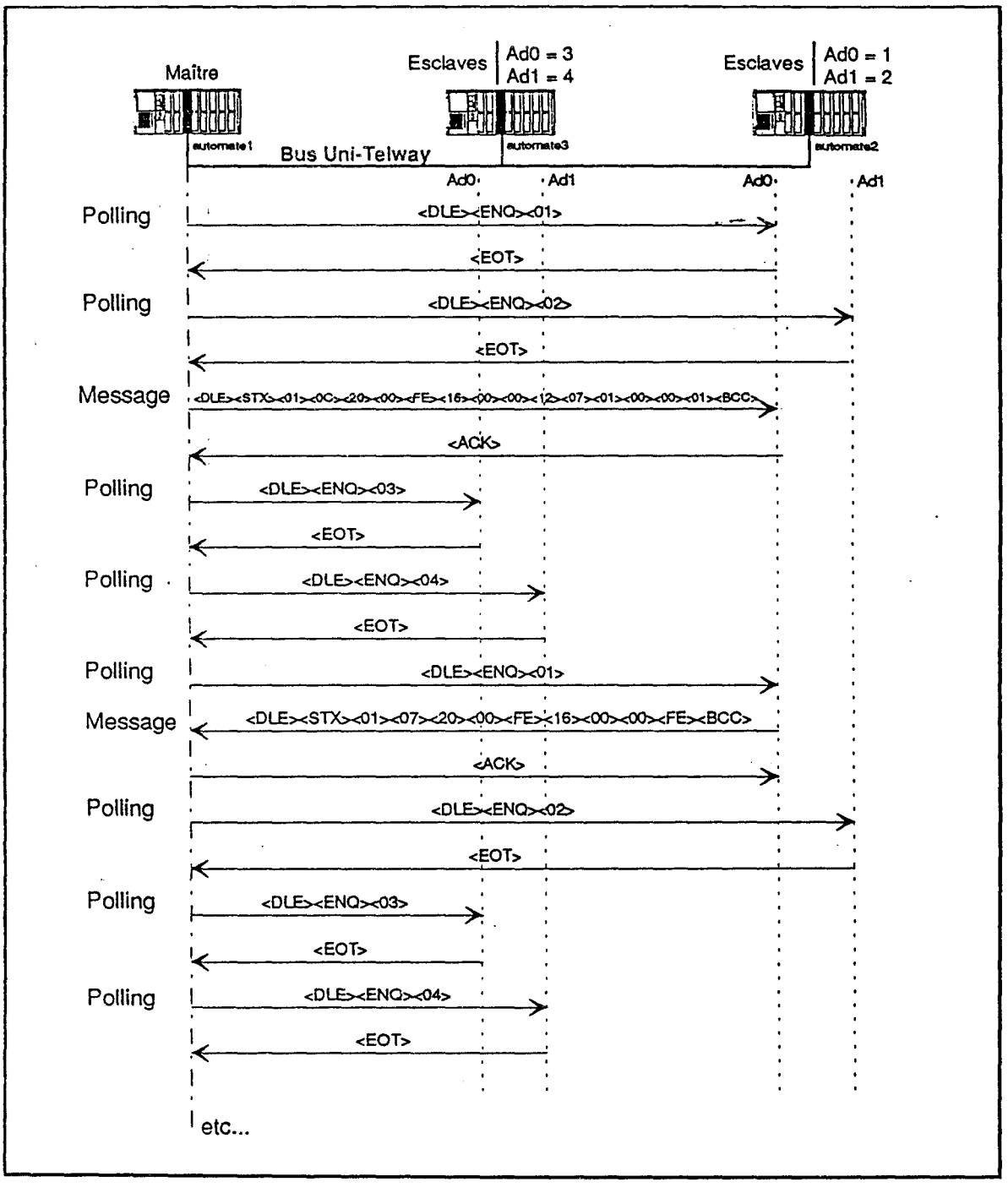


Figure II.49 : *exemples d'échanges sur le bus Uni-Telway*

Le maître interroge cycliquement ses quatre adresses esclave. Au cours de ce cycle, l'automate 1 envoie un message de requête vers l'automate 2 (écriture de l'image d'un bit d'E/S). Après réception, celui-ci est validé par l'esclave avec l'émission d'un ACK. Les pollings reprennent et lorsqu'il est à nouveau interrogé, l'automate 2 envoie un message de compte-rendu qui est validé par un ACK émis par le maître.

I.1.1.d Couche physique

La couche physique assure la transmission des bits sur un canal en vérifiant la bonne arrivée à destination de tous ceux qui ont été émis. Son rôle est donc de faire passer dans de bonnes conditions des caractères utiles sur le bus. Voici ses caractéristiques :

Topologie	- bus industriel hétérogène,
Interface	- RS 485 isolée,
Débit	- configurable de 300 à 19200 bds (9600 par défaut)
Format caractères	- 8 bits de données, parité impaire et 1 bit de stop.

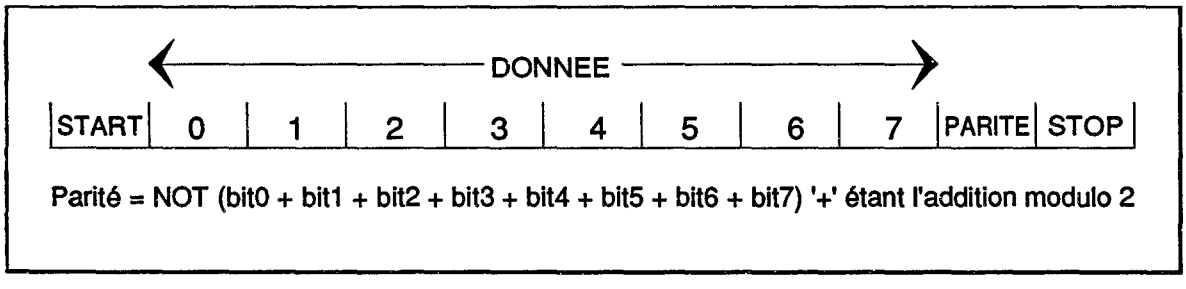


Figure II.50 : *format de représentation d'un caractère*

Nous voyons que pour transmettre un caractère utile, il faut 11 bits, et donc que le temps de transmission associé est : $\frac{11 \times 1000}{\text{débit en bits/seconde}}$ ce qui fait environ 1,15 ms pour un débit de 9600 bauds.

III.1.2. Modèle du bus de communication

Vu de la couche liaison, le bus apparaît comme un "transmetteur" de données : il reçoit des trames qui sont émises par les stations, et les rend disponibles en lecture pour les autres constituants du réseau avec un certain retard lié au nombre de caractères du message. Si l'on veut tenir compte des erreurs de transmission (parasite...), on peut modéliser ce phénomène dans le modèle de comportement du bus.

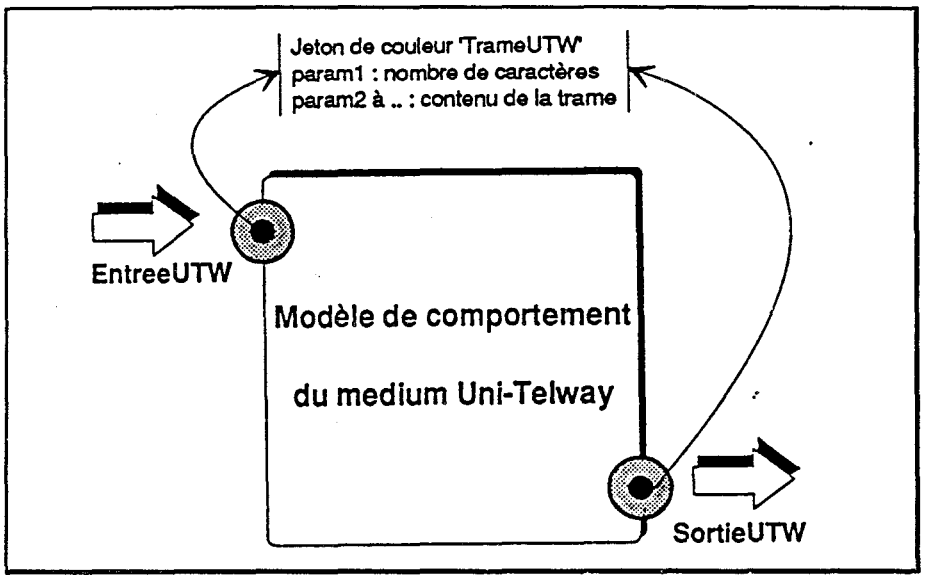


Figure II.51 : *interface du modèle du bus de communication*

Les places de communication EntreeUTW et SortieUTW permettent l'échange de messages avec les modèles de la couche liaison; chaque jeton de couleur TrameUTW modélise une trame en transit sur le bus. Les paramètres des jetons associés sont utilisés pour représenter les informations véhiculées dans la trame (type de trame, émetteur, destinataire etc.), le premier attribut contenant la longueur de la trame. Le modèle de la couche physique n'a besoin de connaître explicitement que cette donnée. Pour les autres paramètres la couche physique se contente de les véhiculer de la place EntreUTW vers SortieUTW sans les interpréter.

L'envoi de message vers la place EntreeUTW se modélise par un énoncé de "Envoi de Message". Pour la lecture des messages de la place SortieUTW, il faut représenter le fait que ces trames sont accessibles par tous les équipements du réseau (cette caractéristique est notamment utilisée pour l'émission de message en diffusion). On emploi donc un énoncé "Lecture de Message" dans le modèle de la couche liaison, en sélectionnant ou non le message en fonction de ses attributs. Une fois que le message a été consulté par tous les équipements, le modèle du bus doit le supprimer.

Le medium Uni-Telway est modélisé (voir figure II.52) par un graphe de processus, dans lequel circule un jeton de couleur 'MediumUTW'. Son premier paramètre contient le temps de transmission d'un caractère (par exemple 1,15 ms à 9600 bds). Le second paramètre, utilisé seulement si on veut tenir compte des erreurs de transmission, permet de de quantifier le taux d'erreur. Ces deux valeurs sont configurées lors du marquage initial. Les autres attributs servent à stocker momentanément les caractéristiques des trames.

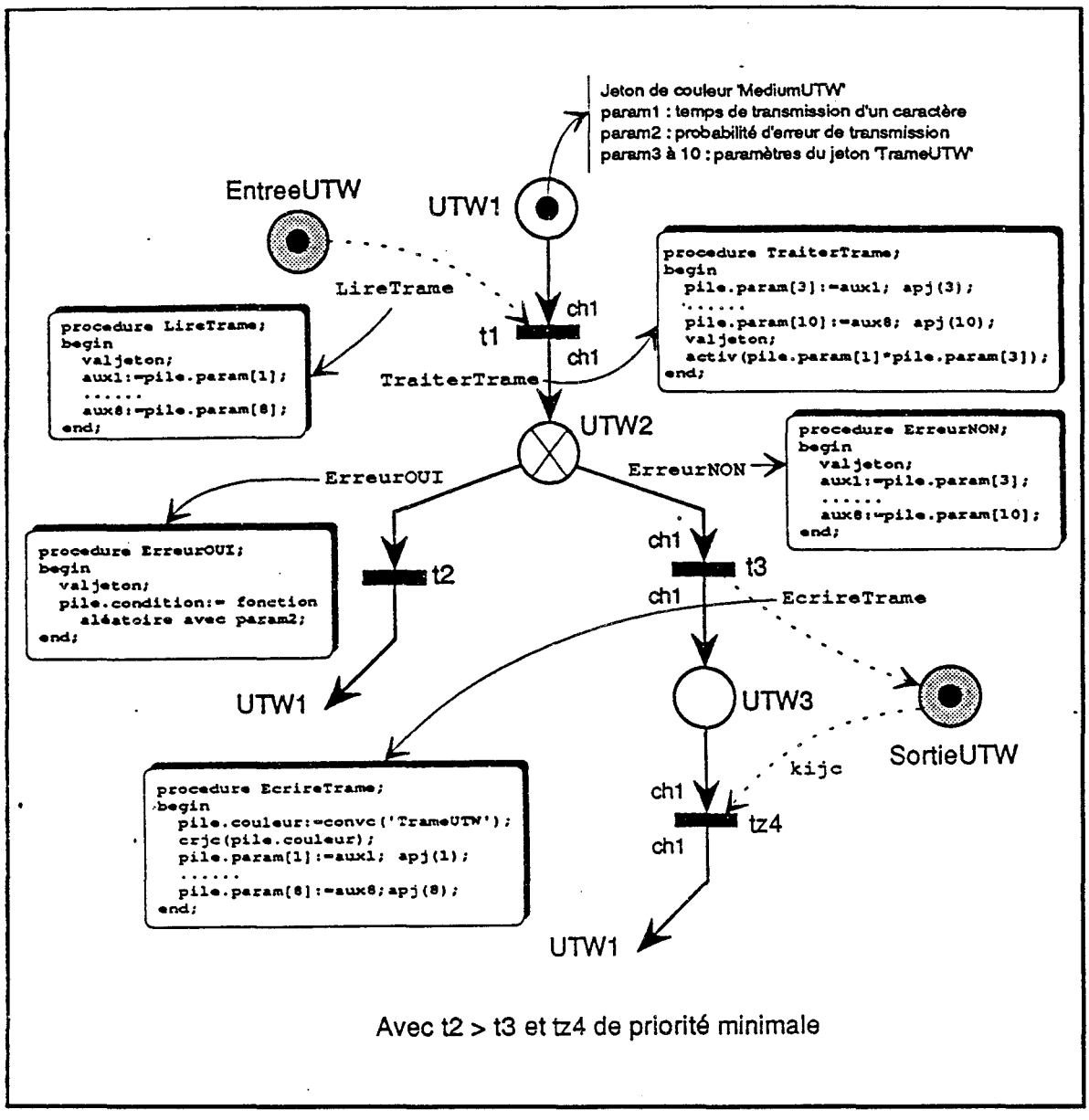


Figure II.52 : *modèle du bus de communication Uni-Telway*

Lorsqu'un message est disponible dans la place EntreeUTW, la transition t1 est tirée. Au moyen des procédures 'LireTrame'et 'TraiterTrame' les attributs de la trame sont transférés sur le jeton d'état. La place UTW2 est activée avec une durée égale au temps de transmission d'un caractère multiplié par le nombre de caractères de la trame.

A la fin de cette temporisation, t2 et t3 sont sensibilisées. Les procédures 'ErreurOUI'et 'ErreurNON'permettent de lever l'indéterminisme, c'est à dire de "décider" si la trame est sujette à une erreur de transmission. Comme t2 est plus prioritaire, 'ErreurOui'est exécutée en premier. Si elle retourne vrai, t2 est tirée et la trame est perdue.

Dans le cas contraire, t3 est franchie et la trame est recréée dans la place SortieUTW, au moyen de la procédure 'EcrireTrame' Chaque modèle de la couche liaison peut alors consulter le message et finalement, tz4 de priorité minimale est tirée et le modèle du bus retourne dans son état initial.

Dans le cas où l'on ne souhaite pas tenir compte des erreurs de transmission, il suffit de supprimer dans le modèle de la figure II.52 la branche gauche de l'alternative (transition t2 et procédure 'ErreurOui').

III.1.3. Modèles du coupleur Uni-Telway [TEL 88b]

III.1.3.a Interface du graphe de processus

Les coupleurs Uni-Telway permettent la connexion des automates sous ce protocole. Ils fonctionnent soit en tant que maître, soit en tant qu'esclave et jouent le rôle d'intermédiaires entre le bus et les API. Dans le modèle, le protocole liaison implémenté dans le coupleur est décrit avec un graphe de processus, qui échange des données, via des places de communication, avec le modèle de la couche physique (places EntreeUTW et SortieUTW) et avec celui de l'automate (places EntreeLiaison et SortieLiaison).

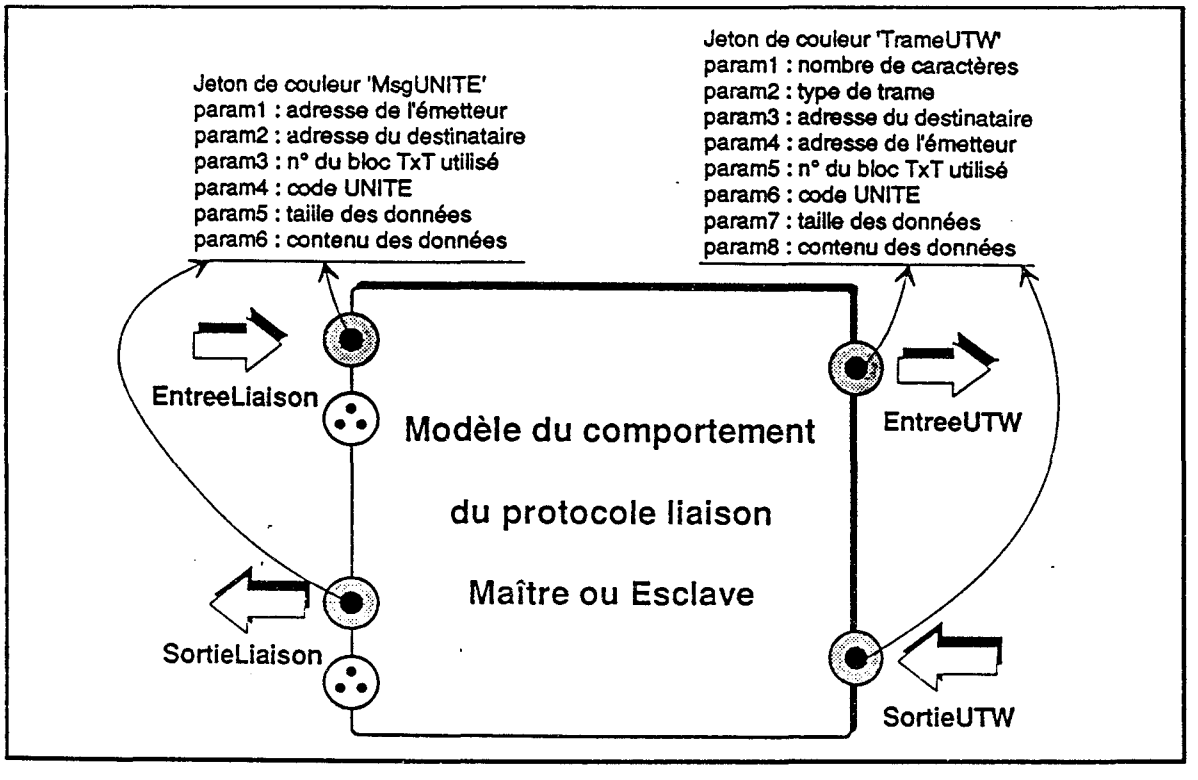


Figure II.53 : *interface du modèle de coupleur Uni-Telway*

Les places de communication avec l'automate sont des places à capacité car les zones réservées pour les échanges automate/coupleur sont limitées en nombre de messages. Leur taille est fonction du type de coupleur, du type d'automate et du protocole utilisé (maître ou esclave).

Les informations véhiculées par les jetons de couleur 'MsgUNITE', permettent de caractériser la requête ou le compte-rendu (émetteur, destinataire, code UNITE etc.). Ces paramètres 'traversent' la couche liaison et se retrouvent dans les attributs du jeton de couleur 'TrameUTW' qui modélise le message qui transite sur le bus.

III.1.3.b Modèle du coupleur configuré en maître

Le modèle du fonctionnement du coupleur maître est directement issu de la description de ce protocole présentée sur les figures II.46 et II.47. Le jeton qui circule dans le graphe est de couleur 'MaitreUTW'. Le premier paramètre de ce jeton, initialisé lors du marquage initial, contient le nombre d'esclaves connectés sur le réseau. Le second attribut représente le numéro du dernier esclave interrogé par le maître. Les autres paramètres sont employés pour stocker les caractéristiques des jetons de messages en provenance du bus ('TrameUTW') et du modèle de l'automate ('MsgUNITE').

Le modèle du comportement du maître se décompose en deux sous-ensembles. Le premier traite de l'invitation à émettre des esclaves et le second de l'émission de messages.

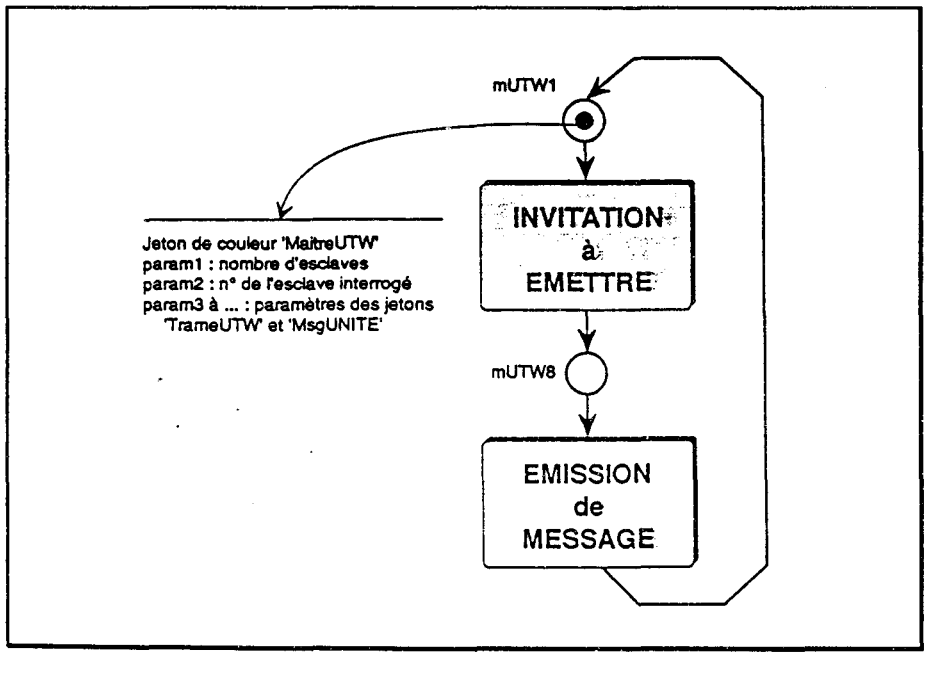


Figure II.54 : *structure du graphe de processus du modèle du protocole maître*

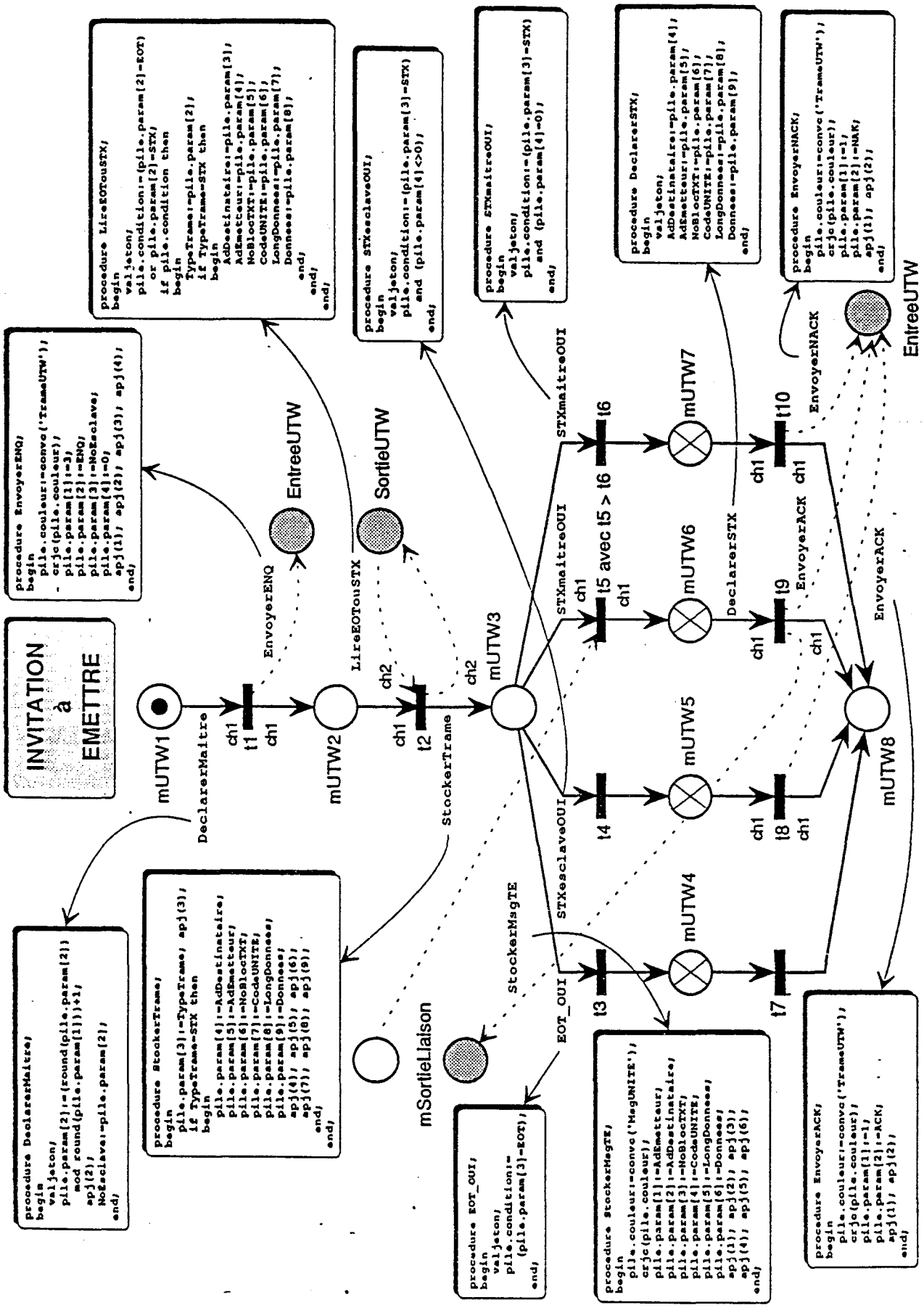


Figure II.55 : *modèle de la partie invitation à émettre du protocole maître*

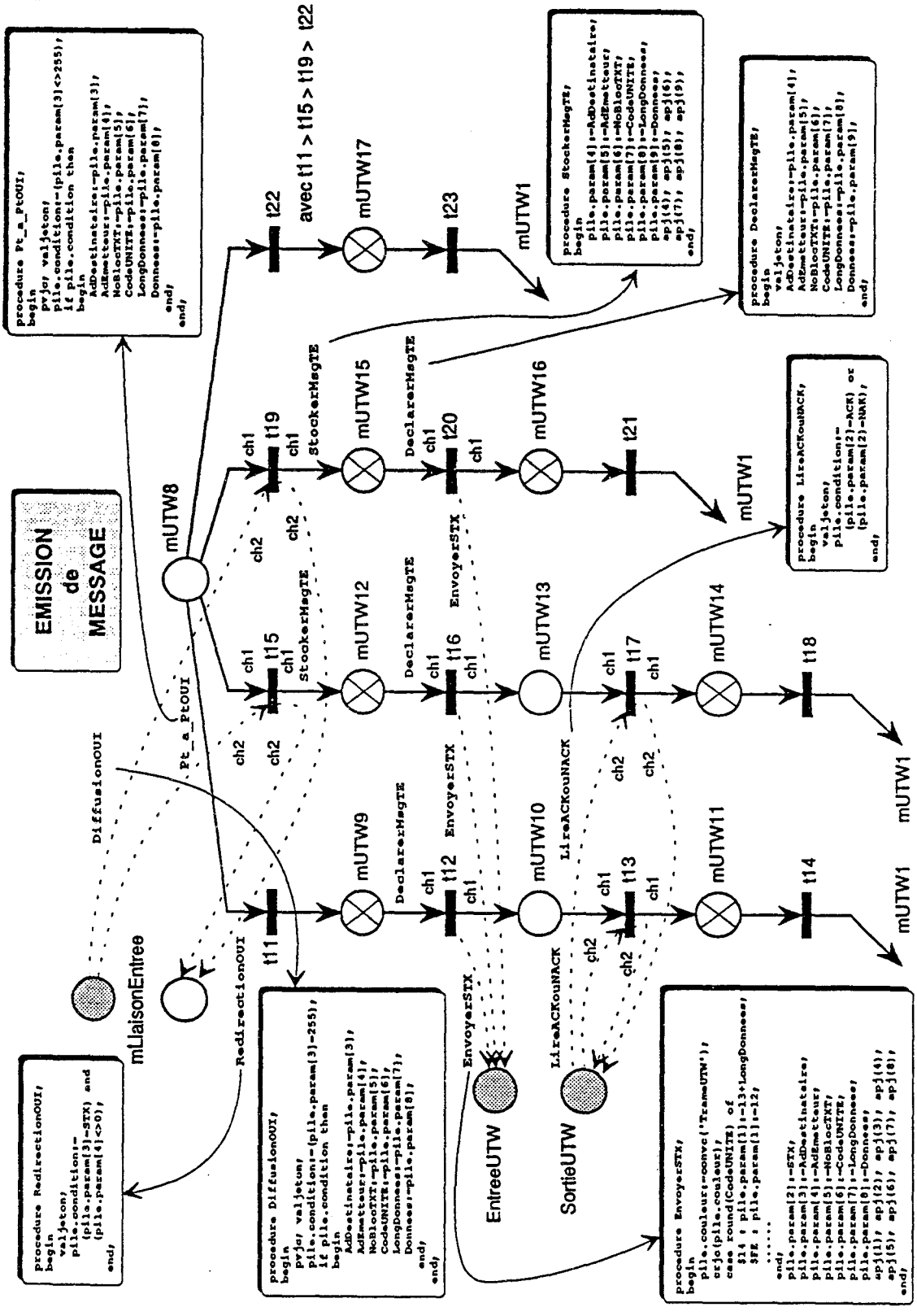


Figure II.56 : *modèle de la partie émission de message du protocole maître*

Lors du tir de la transition t1, un jeton de couleur 'TrameUTW' est envoyé vers le modèle du bus. Ce message modélise la trame de polling (<DLE><ENQ> <adresse_esclave>) émise par le maître à l'attention d'un esclave (ENQ est une constante définie en Pascal, tout comme STX, ACK, NAK, EOT qui sont également utilisées dans les modèles). La variable 'NoEsclave', locale à la transition t1, permet de faire le lien entre le numéro de l'esclave à interroger associé au jeton d'état et l'adresse du destinataire du jeton de message.

Après le franchissement de t1, le jeton d'état marque la place mUTW2 qui correspond à une attente d'un message de type EOT ou STX en provenance du bus. Lorsque la place SortieUTW est marquée avec un jeton qui correspond à un tel message, les paramètres associés sont mémorisés dans les attributs du jeton d'état (procédures 'LireEOTouSTX' et 'StockerTrame', qui utilisent des variables locales : AdDestinataire, AdEmetteur etc.). Celui-ci marque alors la place mUTW3 et quatre alternatives sont possibles pour faire évoluer le modèle.

Si la trame reçue était un caractère EOT (l'esclave interrogé n'a rien à émettre), les transitions t3 puis t7 (l'activité de mUTW4 représente le temps de traitement du message par le coupleur) sont tirées. Si la trame est un message (requête ou compte-rendu UNITE) à l'attention d'un esclave les transitions t4 puis t8 sont franchies, et les informations liées à ce message restent attachées au jeton d'état.

Dans le cas ou il s'agit d'un message à l'attention du maître, deux cas de figure sont possibles. Si la zone d'échange avec l'automate, modélisée par la place à capacité mSortieLiason n'est pas saturée, le message est traité (tir de t5), un jeton de couleur 'MsgUNITE' est créé dans la place mSortieLiaison et un ACK est envoyé sur le bus (franchissement de t9). Dans le cas contraire, le message est refusé et un NACK est diffusé sur le médium (tirs de t6 et t10).

A la fin du traitement de l'invitation à émettre d'un esclave, le jeton d'état du coupleur maître marque la place mUTW8. Commence alors la partie émission de message qui est modélisée sur la figure II.56. Quatre alternatives peuvent se produire, chacune d'elle correspondant à une branche du modèle. Les priorités entre les transitions t11, t15, t19 et t22 permettent lors des simulations, d'étudier leur franchissement dans cet ordre.

Le tir de t11 correspond au cas où, un message de type STX a été reçu, à destination d'un autre esclave. Le maître doit alors le réémettre vers la station concernée. C'est ce qui est réalisé lors du franchissement de la transition t12. Le maître se trouve alors en attente

d'un acquittement du message par l'équipement destinataire (ACK ou NACK). Cet état correspond au marquage de la place mUTW10. Lorsqu'un tel message est disponible, t13 est tirée, mUTW11 est marquée pendant une durée qui correspond au temps de retournement de l'esclave et le jeton 'MaitreUTW' retourne dans la place mUTW1.

Le franchissement des transitions t15 ou t19 correspond au cas où un message (requête ou compte-rendu) en provenance du maître est disponible dans la place mLiaisonEntree. Si le message est à émettre en point à point t15 est tirée (condition 'Pt_a_PtOUI'), s'il est à diffuser t19 est franchie (procédure 'DiffusionOUI'). Dans les deux cas, les paramètres du message sont transférés sur les attributs du jeton d'état au moyen de variables locales. Le franchissement des transitions t16 ou t20 crée la trame associée au message à destination du bus. Lorsque la trame est diffusée, il n'y a pas d'accusé de réception et après le temps de retournement, le jeton retourne en mUTW1. Lors d'une émission en point à point, le message doit être acquitté (comme précédemment) avant que mUTW1 soit à nouveau marquée. Si ni t11, ni t15 ni t19 ne sont franchissables, t22 est tirée, est après le délai associé à la place mUTW17, le jeton d'état marque à nouveau mUTW1.

PROPRIETES DES MODELES :

L'ensemble des places d'état du graphe de processus du protocole maître (mUTW1 à mUTW17) constitue un modèle réseau de Petri sauf. Il s'agit d'un invariant de marquage égal à un dont l'unique jeton est le jeton d'état de couleur 'MaitreUTW'.

Les places de communication mEntreeLiaison et mSortieLiaison étant à capacité sont, de fait, bornées. Les places EntreeUTW et SortieUTW sont bornées à un. Cette propriété est induite par le protocole modélisé. De type maître/esclave, un seul message peut donc être, à un instant donné, en transit sur le bus. Mais il s'agit là d'une propriété valable sur le modèle complet (et en particulier temporisé) qui n'est plus vérifiée sur le modèle binaire sous-jacent.

En effet, lorsque les transitions t9 ou t10 sont tirées, le maître acquitte de manière positive ou négative un message reçu en envoyant une trame de type ACK ou NACK dans la place EntreeUTW. Le jeton d'état MaitreUTW marque alors la place mUTW8. Dans le cas ou la place mLiaisonEntree n'est pas marquée, les transitions t22 et t23 sont tirées en séquence et le graphe est réinitialisé avec le marquage de mUTW1. t1 est donc franchissable, et son franchissement crée un autre jeton dans la place EntreeUTW.

Donc dans le réseau binaire sous-jacent la place EntreeUTW n'est plus bornée à un. Par contre dans le modèle temporisé, l'activité de la place mUTW17 permet d'attendre que le modèle du bus vide la place mEntreeUTW.

PRISE EN COMPTE DES ERREURS DE TRANSMISSION :

Le modèle qui vient d'être présenté est celui du protocole maître sans tenir compte des erreurs de transmission. Si on utilise un modèle de bus qui génère des pertes de trames, il est nécessaire de compléter le modèle du maître pour tenir compte de ces absences de message. On est alors amené à utiliser la primitive de time-out qui a été détaillée dans le paragraphe II.2.4.

Si nous reprenons le modèle de la figure précédente, il est nécessaire de traiter l'absence d'accusé de réception lors d'émission de trame de message en point à point. Dans ce cas, le maître réémet une seconde fois le message, et s'il n'est toujours pas acquitté le message est perdu. Le modèle qui correspond à ce cas de figure est détaillé sur la figure II.57; les places mUTW9 et mUTW10 correspondent à celles du modèle de la figure II.56.

Une primitive de time-out est utilisée après l'envoi du message (franchissement de t12). Si aucun accusé de réception n'est reçu après l'écoulement de la durée du time-out (activité de la place TimeOut1), la transition t26 est tirée et le message est émis une seconde fois. Une autre primitive de time-out est utilisée, et si à nouveau il n'y a aucune réponse, t29 est franchie et le message est définitivement perdu.

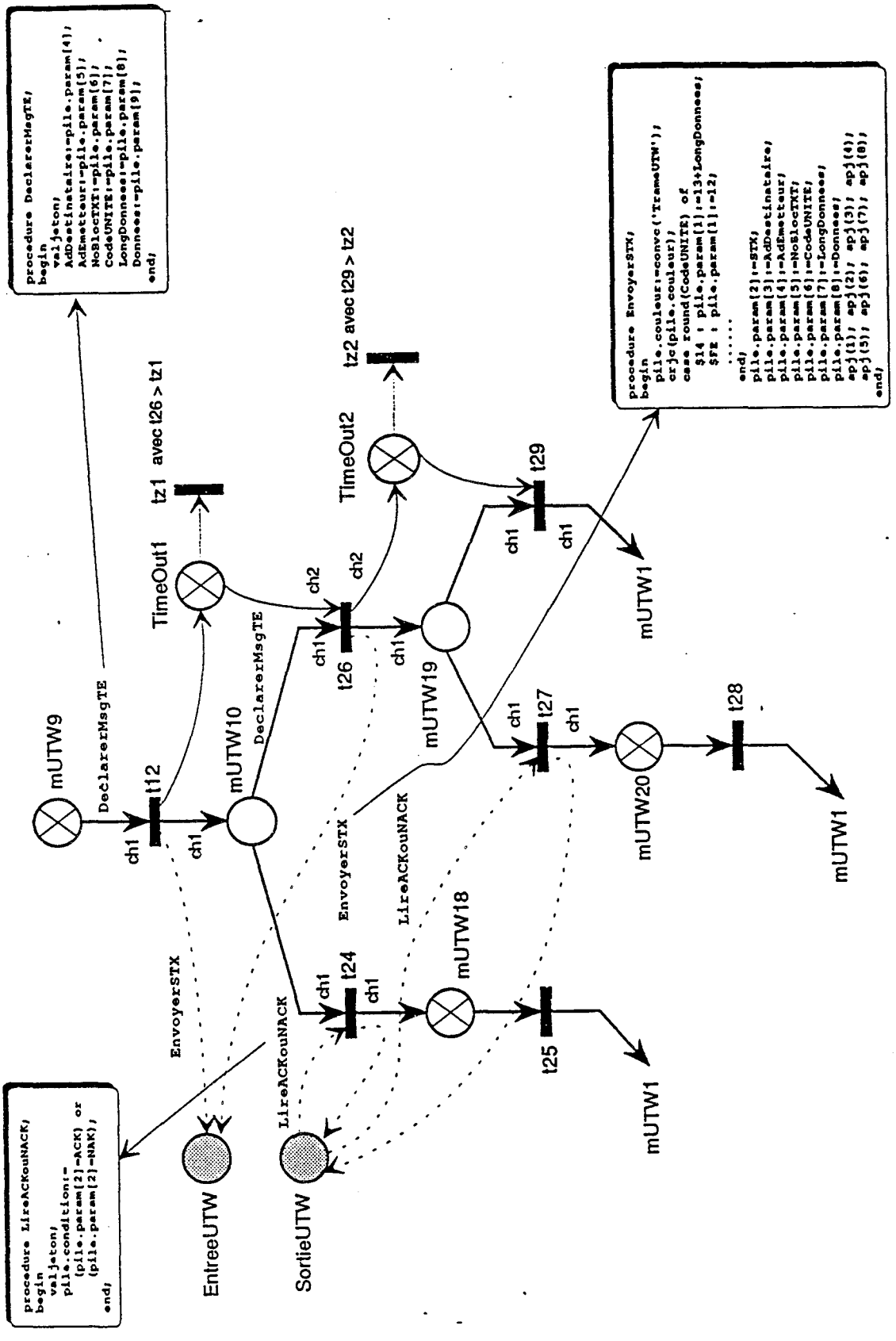


Figure II.57 : *exemple de prise en compte des pertes de message*

III.1.3.c Modèle du coupleur configuré en esclave

Le modèle du protocole esclave est construit d'une manière tout à fait analogue à celui du protocole maître. Le jeton d'état est de couleur 'EsclaveUTW', son premier paramètre représentant le numéro de l'esclave modélisé. Lorsque eUTW1 est marquée, l'esclave est en attente d'un message à son attention : polling (ENQ), message en point à point ou message en selecting. C'est la procédure 'LireENQouSTX' qui autorise le franchissement de t1 lorsqu'une telle trame est disponible.

Les attributs du message reçu sont transférés sur le jeton d'état au moyen de variables locales à la transition t1 (AdEmetteur, AdDestinataire etc.) et de la procédure 'StockerTrame'. Lorsque eUTW2 est marquée, deux alternatives sont possibles. Soit le message est un polling et dans ce cas t2 est franchie (condition attachée à la procédure 'PollingOUI'), soit il s'agit d'un selecting et c'est alors t3 qui est tirée (procédure 'SelectingOUI').

S'il s'agit d'un polling, le jeton d'état marque la place eUTW3 (Cf. figure II.59). Si un message est disponible dans la place eEntreeLiaison, il est lu (tir de t4), traité (activité attachée à la place eUTW4) et une trame de type STX est émise sur le bus (marquage de la place EntreeUTW). La place eUTW5 est alors marquée jusqu'à ce qu'un message d'accusé de réception soit disponible dans la place SortieUTW. La transition t6 est alors franchie et le jeton d'état du coupleur esclave retourne dans la place eUTW1. Lorsque t4 n'est pas franchissable, t7 est tirée et une trame de type EOT (pas de message à envoyer) est envoyée à l'attention du maître (tir de la transition t8).

S'il s'agit d'un selecting, le jeton d'état marque la place eUTW7 (Cf. figure II.60). La transition t9 est tirée lorsque le message est une émission en point à point (procédure 'Pt_a_PtOUI'). Dans ce cas, si une ressource est disponible pour stocker le message, il est traité (tir de t10 et t11) et un accusé ACK est envoyé sur le bus. Dans le cas contraire, t12 et t13 sont franchies et l'accusé est négatif (NACK). Lorsque le message a été envoyé en diffusion, ce n'est pas t9, mais t14 qui est tirée. Dans ce cas, le message n'est pas acquitté. Si une ressource est disponible, le message est mémorisé (tirage de t15 et t16) et dans le cas contraire le jeton d'état retourne simplement en eUTW1.

Remarque :

Lorsque plusieurs esclaves sont connectés sur le même réseaux, pour diminuer la taille du modèle, on introduit dans le même graphe autant de jetons d'état qu'il existe de stations connectées.

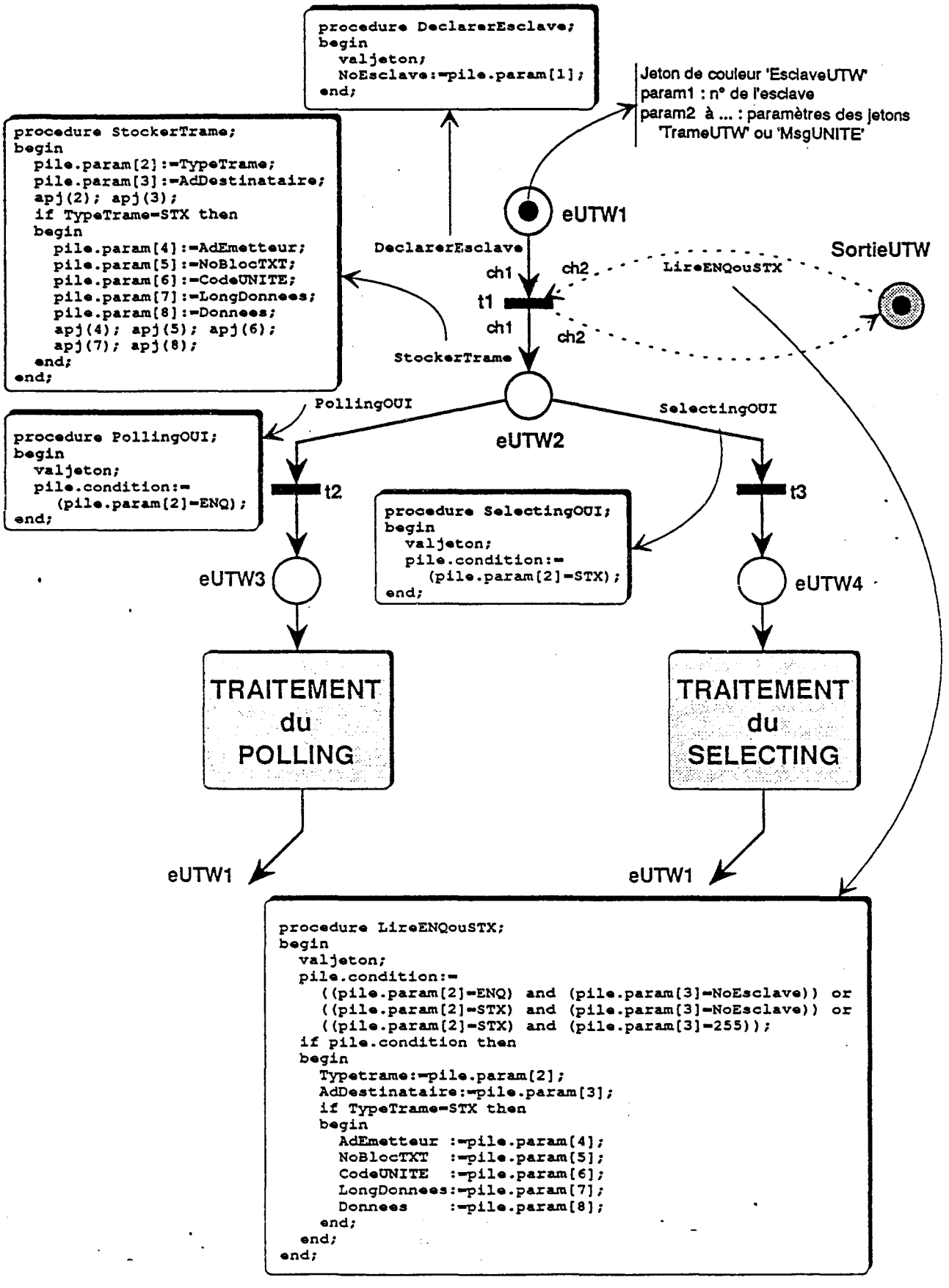


Figure II.58 : *modèle globale du protocole esclave*

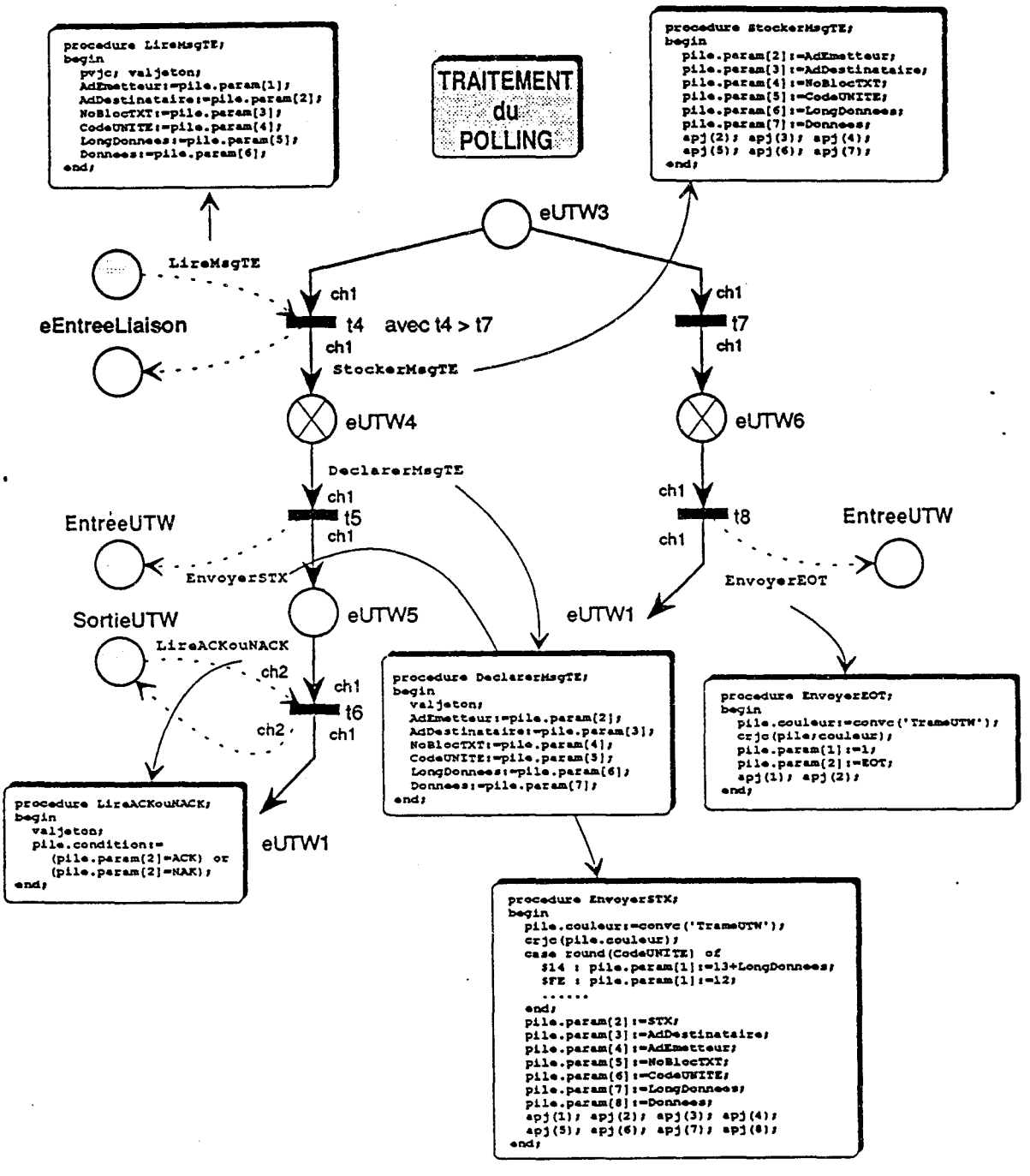


Figure II.59 : *modèle de la partie traitement du polling du protocole esclave*

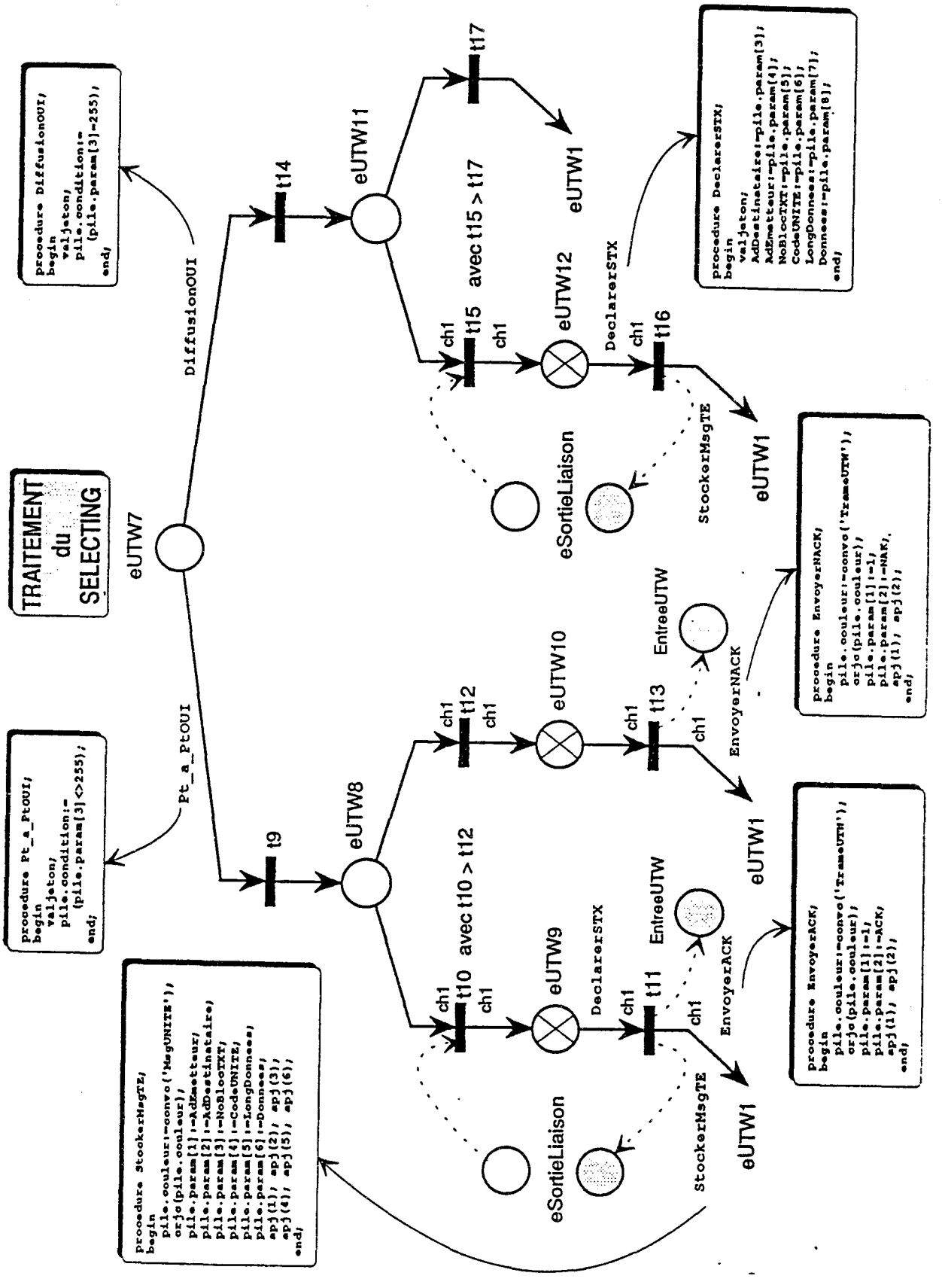


Figure II.60 : *modèle de la partie traitement du selecting du protocole esclave*

Mais les évolutions de ces différents jetons d'état, caractérisés par la valeur de leur premier paramètre (numéro d'esclave) et qui est fixée avec le marquage initial, sont totalement indépendantes. Les propriétés du modèle sont alors les mêmes que si on avait autant de graphes que d'équipements esclave. L'invariant de marquage pour le graphe de processus ne vaut plus un, mais est égal au nombre d'esclaves modélisés.

III.1.4. Modélisation d'autres réseaux locaux industriels

Au cours de nos travaux, nous avons été amenés, pour être à même d'étudier des architectures installées par le groupe PSA Peugeot Citroën (Cf. annexe 2), de modéliser d'autres réseaux locaux industriels, que celui qui vient d'être présenté.

Nous avons ainsi notamment étudié le réseau Telway 7 [TEL 89]. Il s'agit d'un bus inter-automates. Ses services sont complémentaires de ceux offerts par le réseau Uni-Telway. Il permet l'émission/réception de données entre stations, l'acheminement de requêtes système et le partage d'une table de mots, appelés mots COM, entre les automates du réseau.

Au niveau liaison, le protocole est de type maître/esclave, mais le maître est flottant, et non plus fixe comme dans Uni-Telway. Il est élu lors du démarrage du réseau. L'approche utilisée pour modéliser ce protocole est tout à fait similaire à celle développée précédemment. Le modèle du bus est identique, mais le débit est configuré à 19200 bauds. Le modèle des coupleurs Telway 7 reprennent les spécifications du protocole qui sont détaillées dans [LEP 91]. Des précisions sur ce modèle pourront être trouvées dans [RIA 91a] et [RIA 92].

De la même manière, un modèle de réseau Modbus (Cf. annexe2) a été élaboré en suivant la même approche que pour les protocoles Uni-Telway ou Telway 7. Il est d'ailleurs intéressant de remarquer que pour ces trois exemples, les modèles du bus sont identiques, à la différence près de la valeur du temps de transmission d'un caractère (et de la couleur du jeton d'état !).

III.2. Modélisation d'automates programmables

Les automates programmables industriels (API) assurent la commande de niveau 1 et la coordination de niveau 2 dans les architectures des systèmes de production automatisés. Ces équipements sont caractérisés par un système d'exploitation spécifique qui leur permet d'assurer des temps de réponse compatibles avec les contraintes temporelles imposées par le procédé qu'ils commandent.

Dans le cadre de nos travaux, il a été nécessaire d'identifier pour les automates étudiés, les caractéristiques de leur fonctionnement qui influent sur les échanges de données au niveau des réseaux de l'architecture. Nous avons donc été amenés à modéliser précisément la manière dont les messages en provenance et à destination des applicatifs (programmes de contrôle/commande implantés dans les API) étaient échangés avec les coupleurs de communication.

Ces éléments sont spécifiques à chaque modèle d'automate. Nous avons porté notre attention sur les produits de la société Télémécanique et en particulier sur les échanges avec les coupleurs de protocole Uni-Telway qui ont été modélisés dans le paragraphe précédent.

III.2.1. Fonctionnement des automates programmés en PL7-3

III.2.1.a Structure logicielle

Les automates programmables TSX 47/67/87/107 de la société Télémécanique [TEL 90c] ont un fonctionnement multi-tâches. Les tâches sont indépendantes et gérées par le superviseur de l'automate en fonction des priorités et périodicités propres à chacune d'elles.

La ***tâche interruption*** est une tâche utilisateur asynchrone, dont l'exécution est déclenchée par un ordre en provenance d'événement extérieur. Elle est traitée en priorité sur toutes les autres tâches.

La ***tâche rapide*** est la plus prioritaire des tâches périodiques. Elle doit être réservée aux traitements de courte durée à fréquence d'exécution élevée.

La ***tâche maître*** est LA tâche périodique obligatoire exécutant le traitement séquentiel. C'est la tâche de base de l'application. Elle est activée systématiquement par le superviseur et permet l'activation des autres tâches par l'intermédiaire du bloc CTRL.

Les ***tâches auxiliaires*** sont des tâches périodiques destinées aux traitements plus lents tels que mesure ou dialogue opérateur.

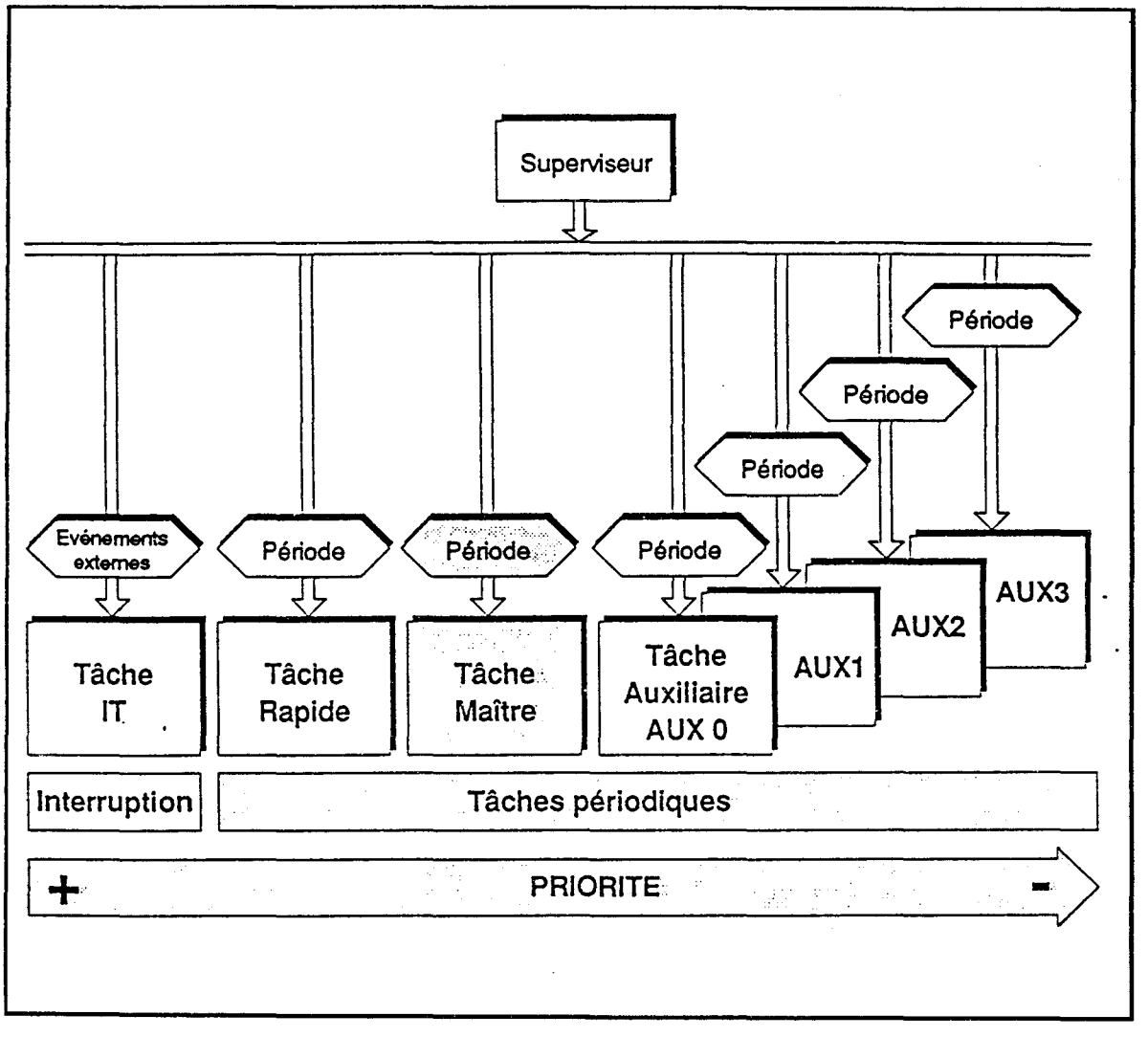


Figure II.61 : *structure logicielle multi-tâches de PL7-3*

Selon les besoins de son application, le concepteur peut opter pour un fonctionnement mono-tâche, et dans ce cas n'utiliser que la tâche maître, ou pour un fonctionnement multi-tâches en intégrant des tâches rapides, auxiliaires ou d'interruption.

III.2.1.b Traitement mono-tâche

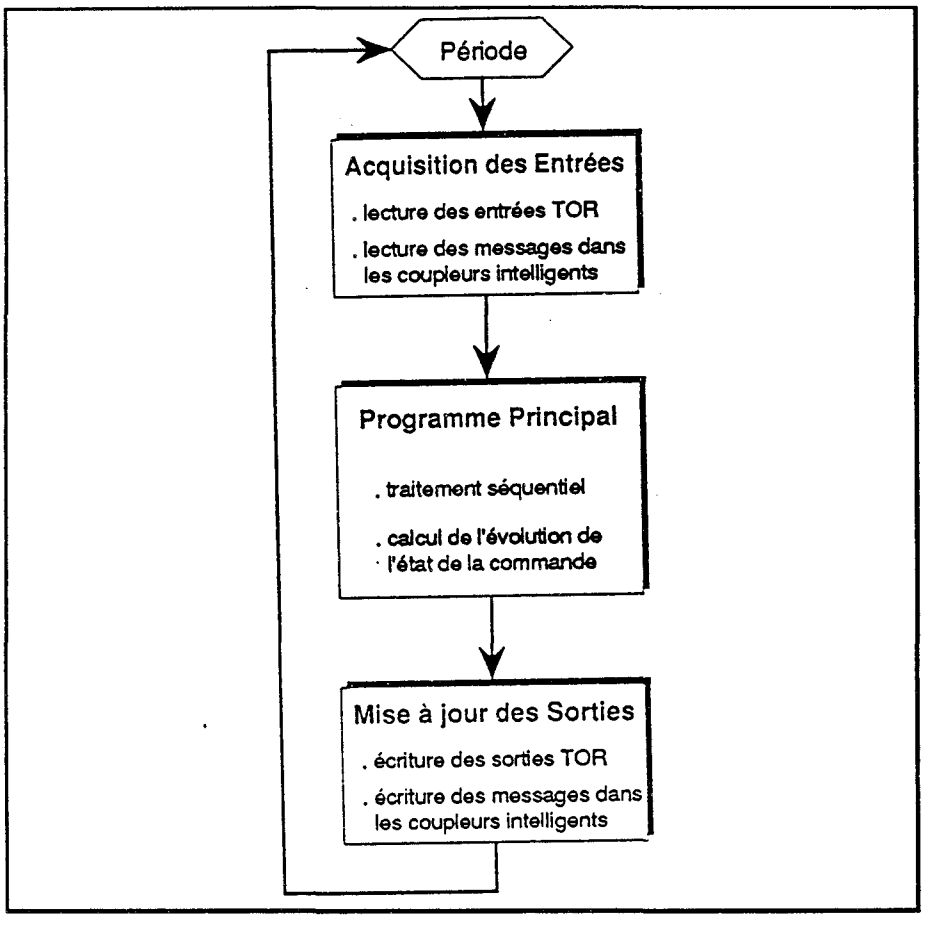


Figure II.62 : *fonctions de la tâche périodique maître*

Le programme d'une application mono-tâche est contenu dans une seule tâche utilisateur : la tâche maître. Elle est exécutée de façon périodique, selon un temps défini à la configuration par l'utilisateur (de 15 à 255 ms). Ce temps doit permettre le déroulement de l'ensemble des fonctions représentées sur la figure II.62.

L'acquisition des entrées correspond à la lecture implicite de l'état des entrées TOR et des messages en provenance des coupleurs intelligents déclarés dans la tâche.

Le traitement du programme correspond à l'exécution de l'applicatif écrit par le programmeur et implémenté en mémoire automate.

La mise à jour des sorties correspond à l'écriture implicite des sorties TOR et à l'émission des messages vers les coupleurs de communication utilisés.

III.2.1.c Traitement multi-tâches

Dans un soucis d'optimisation et de simplification de développement de son applicatif de commande, l'utilisateur peut répartir les traitements en fonction de leur fréquence et leur périodicité entre les tâches rapides, auxiliaires et d'interruption et la tâche maître. Dans ce cas, c'est toujours cette dernière qui constitue la base de l'application et qui permet l'activation ou non des autres tâches.

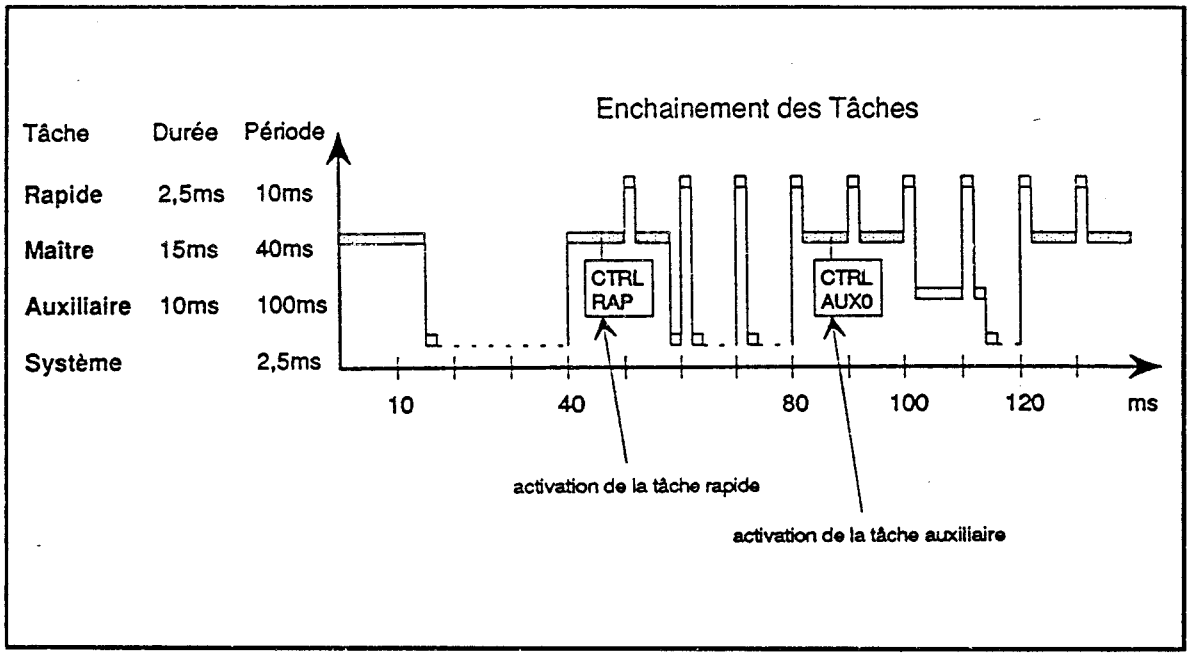


Figure II.63 : *exemple de chronogramme d'enchaînement de tâches en PL7-3*

A titre d'illustration, le chronogramme de la figure II.63 détaille l'enchaînement des tâches sur un automate où sont configurées une tâche maître, une tâche rapide et une tâche auxiliaire.

Dans un fonctionnement multi-tâches, le déroulement de la tâche maître est identique à celui présenté sur la figure II.62. Les tâches périodiques rapides et auxiliaires se décomposent de la même manière, avec une partie d'acquisition des entrées, une autre de traitement et une dernière de mise à jour des sorties. Cependant les échanges avec les entrées/sorties concernent uniquement les informations TOR et des registres internes. Le traitement des messages avec les coupleurs de communication s'effectuent exclusivement dans la tâche maître.

III.2.2. Modèle de la tâche maître

III.2.2.a Interface du graphe de processus

La tâche maître est la composante essentielle du fonctionnement de l'automate par rapport à ses échanges sur l'architecture de communication. En effet, c'est elle qui assure le dialogue entre l'applicatif et les coupleurs intelligents.

Le modèle de la tâche maître est un graphe de processus qui est issu du fonctionnement cyclique détaillé sur la figure II.62. Ce modèle s'interface d'une part avec celui des coupleurs de communication, d'autre part à la description de l'applicatif implémenté dans l'équipement et également au modèle simplifié du procédé commandé.

Afin de détailler le fonctionnement d'un tel modèle, nous allons présenter celui de la tâche maître d'un automate équipé d'un coupleur Uni-Telway.

Les échanges avec le modèle du coupleur sont réalisés au travers des places de communication EntreeLiaison et SortieLiaison, qui symbolisent les zones d'échanges entre l'U.C. de l'automate et son coupleur Uni-Telway.

Comme expliqué dans le paragraphe II.3.2, nous avons choisi de décrire l'applicatif de l'équipement au moyen de l'interprétation disponible dans le formalisme utilisé : la description des échanges de l'automates est réalisé avec une procédure Pascal. Pour faciliter cette description, deux structures de données Pascal sont attachées au modèle :

E/S_TOR : les variables I et O contiennent l'image des entrées/sorties de l'automate. Elles sont mises à jour à chaque cycle, en fonction des valeurs des attributs des jetons des places TSX87_I et TSX87_O, qui assurent l'interface avec le modèle de l'évolution du procédé. Cette approche a été choisi pour pouvoir définir de manière fine les scénarios de simulation (Cf. paragraphe II.3.3).

TXT : avec le langage PL7-3, les échanges de messages avec les coupleurs intelligents sont programmés avec des blocs TxT. Afin de rendre les descriptions des applicatifs proches des programmes implémentés dans les équipements, nous avons défini une structure TxT avec des champs analogues à ceux proposés par Télémécanique.

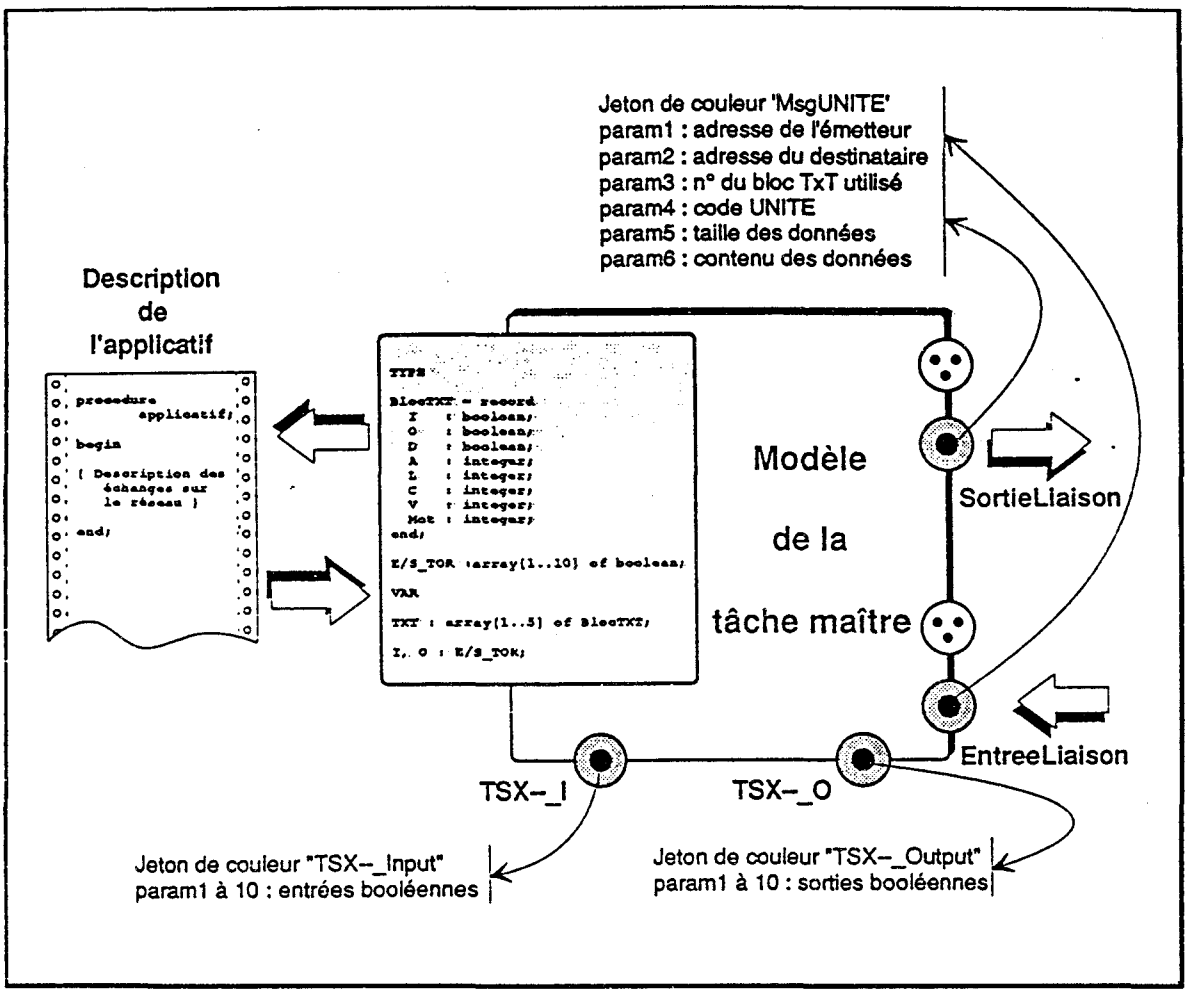


Figure II.64 : *interface du graphe de processus de la tâche maître*

C'est alors le modèle de la tâche maître qui réalise le lien entre l'utilisation de ces structures de données dans les descriptifs des applicatifs et le graphe de processus sous-jacent. Des exemples d'applicatifs, qui mettent en oeuvre ces principes, sont développés dans le chapitre III de ce mémoire.

III.2.2.b Description du modèle

Le graphe de processus complet du modèle de la tâche maître d'un automate programmable TSX 87 configuré avec un coupleur Uni-Telway et présenté sur les figures II.65 et II.66 (la place TSX87_4 est la même sur les deux schémas).

Le jeton d'état est de couleur 'TSX87'. Ses deux premiers attributs sont utilisés pour gérer le temps d'exécution de la tâche : le premier contient la durée effective consommée par la tâche et le second sa période d'activation, c'est à dire le temps de cycle de la tâche maître.

Au début de chaque cycle, la place TSX87_1 est marquée. Si aucun message n'est disponible en provenance du coupleur, la transition t2 est franchie. Dans le cas contraire, le message disponible est lu par le tir de la transition t1 et les paramètres associés sont mémorisés sur le jeton d'état. TSX87_2 est alors marquée. Si le message est une requête système, il est nécessaire de générer le compte-rendu associé. C'est alors t3 qui est tirée et le compte-rendu est mémorisé temporairement dans la place MSG, avant d'être transmis au coupleur en fin de cycle.

Si le message n'était pas une requête, c'est qu'il s'agit d'un compte-rendu associé à une requête émise préalablement par un des blocs TxT de l'automate. Dans ce cas, t4 est franchie et le bloc TxT concerné (dont le numéro a toujours été associé aux paramètres du message) est positionné avec les données associées.

Remarque : les procédure 'RequeteOUI', 'RequeteNON' et 'CompteRendu' sont à enrichir avec les codes des requêtes et comptes-rendus susceptibles d'être employés dans les applicatifs. Sur la figure II.65, seule la requête $14 (lecture de mot) et son compte-rendu positif ($FE) sont décrits.

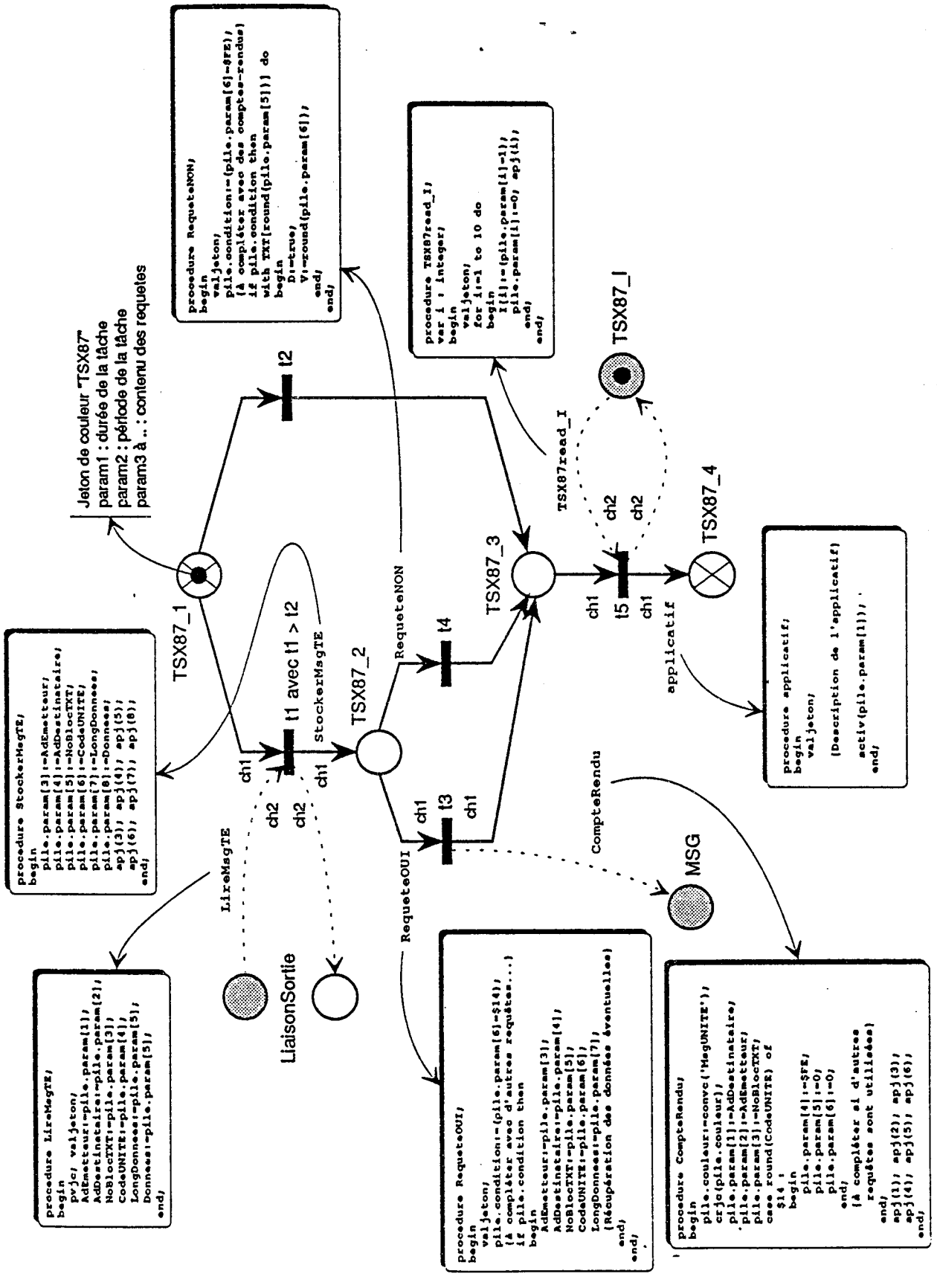


Figure II.65 : *première partie du modèle de la tâche maître*

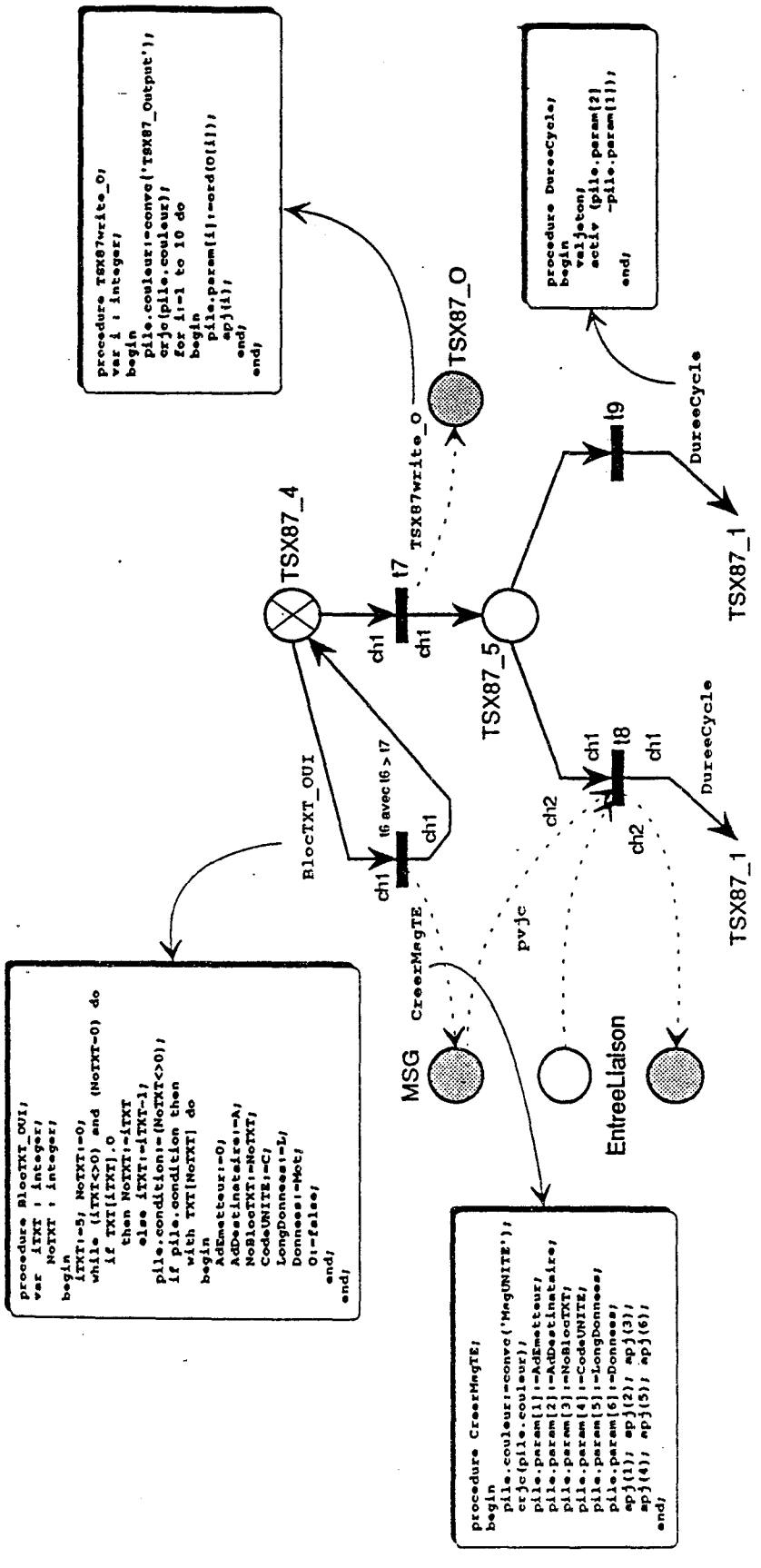


Figure II.66 : *seconde partie du modèle de la tâche maître*

Lorsque la place TSX87_3 est marquée, le franchissement de t5 modélise l'acquisition des entrées TOR : les paramètres du jeton de la place TSX87_I sont transmis dans la variable Pascal `I`. L'action '`applicatif`', en sortie de t5, est la procédure Pascal qui contient la description de l'applicatif. La place TSX87_4 est temporisée avec une activité égale à la durée d'exécution de la tâche. Dans le cas où ce temps n'est pas constant, il est possible au niveau du modèle de le représenter de manière plus fine, par une loi aléatoire par exemple, en modifiant le paramètre de la procédure '`activ`'.

Lorsque l'activité de la place TSX87_4 est écoulée, la transition t6 est tirée tant qu'un bloc TxT reste positionné en émission. La procédure '`BlocTXT_OUI`' parcoure les structures `TXT` et retient la première qui a été modifiée par l'applicatif pour envoyer une requête. Si un tel bloc est disponible, t6 est tirée, une requête est créée dans la place MSG et le bloc `TXT` n'est plus positionné en émission (`TXT.O = FALSE`). Dans le cas contraire, t7 est tirée et les sorties TOR sont mises à jour dans la place TSX87_O (transfert des valeurs de la variable O vers les paramètres du jeton).

La fin du graphe de processus concerne l'envoi de message à destination du coupleur. Si aucun message n'est disponible (place MSG non marquée) t9 est tirée. Sinon, si une place dans la zone d'échange avec le coupleur est disponible, le message le plus ancien est sélectionné, t8 est franchie et le jeton de message associé marque la place EntreeLiaison.

La procédure '`DureeCycle`', en entrée de la place TSX87_1, temporise cette place avec une activité égale au temps de cycle diminuée de la durée d'exécution de la tâche maître. Ainsi, lors des simulations, la tâche maître est activée précisément à chaque période.

Remarque : dans le modèle qui a été décrit, toute la durée d'exécution de la tâche correspond à l'activité de la place TSX87_4. Si on voulait être plus précis dans la modélisation, il est possible de décomposer cette durée en temps pour acquérir les entrées, durée de traitement et temps pour mettre à jour les sorties. Dans ce cas, il serait nécessaire de temporiser les places TSX87_2, TSX87_3 et TSX87_5. Cependant, comme les activités associées restent souvent faibles par rapport à la durée de traitement et ne sont pas connues, nous avons choisi de simplifier le modèle en regroupant toute la durée d'exécution dans l'activité de la place TSX87_4.

PROPRIETES DU MODELE :

Le graphe de processus du modèle de la tâche maître est sauf avec un invariant de marquage de un (le jeton d'état de couleur 'TSX87'). Il est intéressant de remarquer que ce graphe est réinitialisable et ne présente pas de cas de blocage possible.

Ces propriétés se montrent facilement en remarquant que les transitions qui correspondent à l'attente d'un message en lecture (t1 et t8) sont en conflit de tir avec des transitions franchissables sans condition (t2 et t9). Ceci implique que cycliquement le jeton d'état marque la place TSX87_1, ce qui constitue une conséquence directe du fonctionnement périodique de la tâche modélisée.

III.2.3. Extensions

III.2.3.a Structure logicielle multi-tâches

La tâche maître est moins prioritaire qu'une tâche rapide ou d'interruption. Elle est donc susceptible d'être interrompue par ces dernières. Dans ce cas son exécution reprend seulement après la fin de la routine la plus prioritaire.

Une première façon de prendre en compte ce phénomène dans le modèle de l'automate est d'ajouter dans la durée d'exécution de la tâche maître une fonction qui allonge ce temps d'une durée correspondante à l'exécution de la tâche plus prioritaire.

Comme durant la phase de modélisation d'architecture de commande, il est nécessaire d'estimer les temps de cycle et les durées d'exécution des tâches cette approche nous semble suffisamment fine par rapport à la précision des informations temporelles adoptées.

Cependant, dans le cas où il s'avérait nécessaire de modéliser d'une manière extrêmement précise le déroulement des tâches maître, auxiliaires et rapides, la limite du modèle proposé est atteinte. En effet, le traitement de la tâche maître est modélisé comme un tout, avec une procédure Pascal qui ne peut être interrompue. Dans ce cas, le modèle devra donc être revu, en adoptant une approche inspirée de celle que nous avons détaillée pour représenter le mécanisme de préemption (Cf. § II.4.2.).

III.2.3.b Automate multi-coupleurs

Le modèle qui vient d'être détaillé correspond à un automate équipé d'un seul coupleur Uni-Telway. Dans le cas où un API comporte plusieurs coupleurs, le modèle associé comportera autant de modules lecture de messages (places TSX87_1, TSX87_2 et TSX87_3, transitions t1, t2, t3 et t4) et écriture de messages (place TSX87_5, transitions t8 et t9) que de coupleurs.

Par exemple, dans le cas d'un automate avec deux coupleurs Uni-Telway, le modèle de la tâche maître aura l'allure suivante.

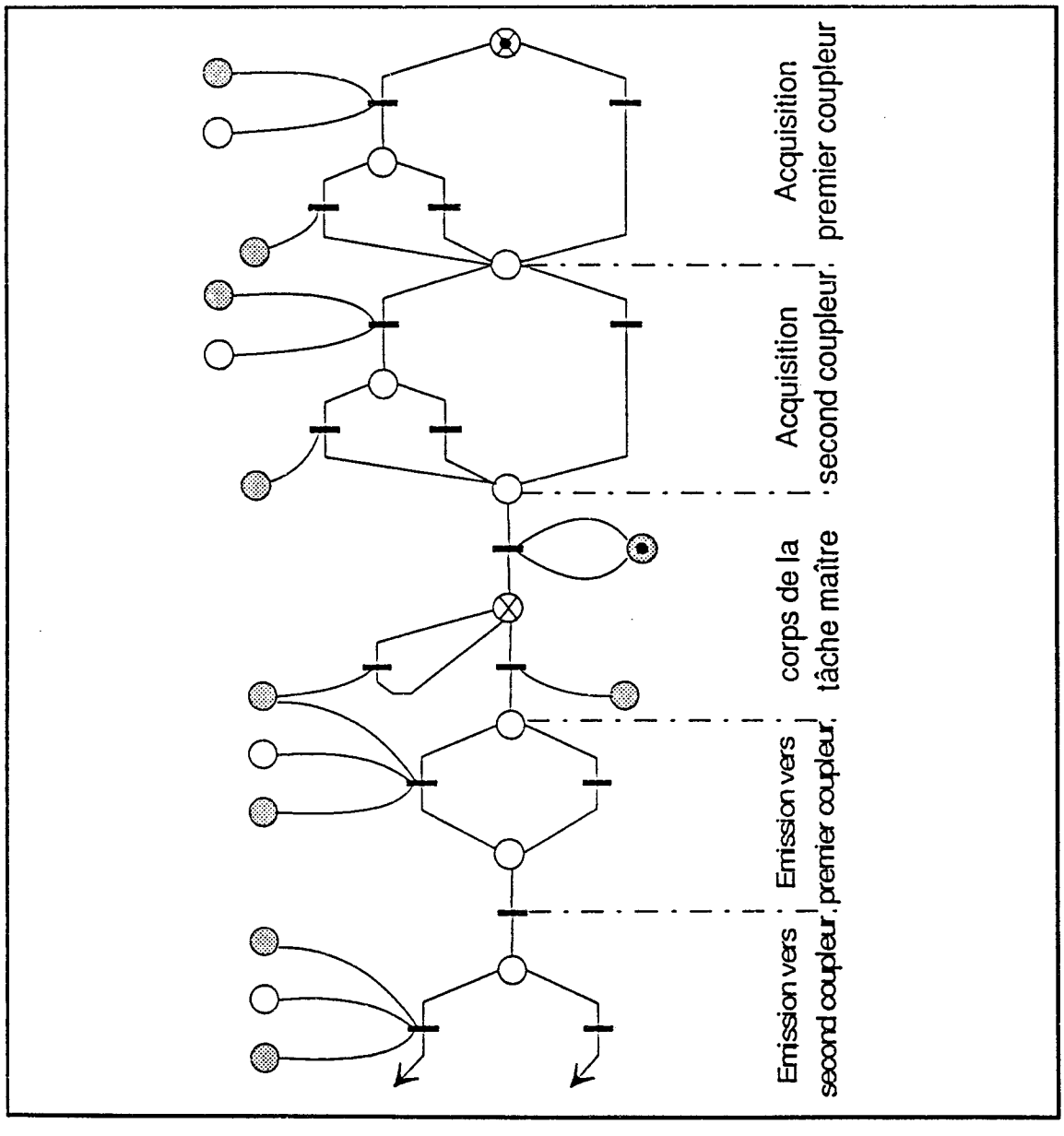


Figure II.67 : *modèle d'un automate avec deux coupleurs Uni-Telway*

III.3. Généralisation à d'autres équipements

III.3.1. Automates programmables Télémécanique programmés en PL7-2

PL7-2 est une structure logicielle, moins étendue que PL7-3, qui est disponible dans l'offre de Télémécanique pour les petits automates TSX17-20, TSX47-10 et TSX47-20. [TEL 88d]. Un tel automate peut fonctionner selon deux modes :

* **cycle monotâche :** l'automate exécute une seule tâche, la tâche maître qui est cyclique séquentielle mais non périodique. Le cycle automate est alors le suivant :
  - gestion du dialogue avec le terminal connecté,
  - acquisition des entrées,
  - traitement du programme écrit en PL7-2,
  - mise à jour des sorties.

* **cycle bitâche :** l'automate exécute la même tâche maître qu'en cycle monotâche avec en plus une tâche rapide périodique. A chaque signal de la base de temps, l'automate interrompt l'exécution de la tâche maître afin d'exécuter la tâche rapide.

Nous voyons que le fonctionnement est analogue à celui d'un automate programmé en PL7-3, avec une tâche maître (et éventuellement une tâche rapide) à la différence que cette fois-ci la tâche principale n'est plus périodique : dès qu'elle a terminé son exécution, elle reprend immédiatement un nouveau cycle.

Pour modéliser un tel comportement, nous pouvons reprendre le modèle des figures II.65 et II.66 en ne conservant que le paramètre de durée de la tâche, l'attribut lié à la période n'ayant plus de raison d'être.

Sur les micro-automates TSX17-20 équipés d'un coupleur Uni-Telway de type ACC5 [TEL 88c], celui-ci est toujours configuré en esclave. Les échanges entre U.C. et coupleur ne sont pas réalisés tous les cycles comme avec PL7-3, mais seulement un cycle sur deux. Dans la modélisation des échanges avec le coupleur, il est donc nécessaire d'ajouter une précondition supplémentaire pour autoriser la lecture ou l'écriture de message avec les places EntreeLiaison et SortieLiaison. Ces conditions doivent être liées à un nouveau paramètre du jeton d'état, modifié à chaque cycle, qui indique si le cycle courant autorise ou non les échanges avec le coupleur.

C'est cette approche qui a été mise en oeuvre dans les modèles des micro-automates TSX 17-20 qui ont été utilisés pour traiter l'exemple développé dans le chapitre III de ce mémoire.

III.3.2. Automates Siemens

Nous voyons que le modèle d'automate qui a été proposé permet avec quelques modifications, de représenter la gamme des automtes industriels TSX série 7, Il est donc intéressant de regarder dans quelle mesure, ce modèle qui apparaît générique pour les produits d'un constructeur s'adapte à des matériels d'un autre type.

Le groupe PSA Peugeot Citroën utilise dans ses ateliers des automates programmables de marque Siemens et Télémécanique. Nous avons donc regardé si les travaux réalisés sur les produits français pouvaient s'étendre aux matériels germaniques. Pour les deux marques d'automates, les principes de fonctionnement sont identiques : fonctionnement cyclique, lecture et mise à jour des entrées/sorties TOR, partie communication implantée dans des coupleurs spécifiques etc.

La structure cyclique du modèle avec la lecture et la mise à jour des entrées/sorties peut donc être reconduite. Cependant des adaptations sont nécessaires pour tenir compte des spécificités des automates Siemens. Celles-ci concernent principalement les modes d'échanges entre coupleur et tâche automate.

III.3.3. Autres équipements connectés aux réseaux

Pour un réseau donné, les drivers de protocole implémentés dans les différents équipements ont le même comportement : ils suivent les spécifications du protocole. Donc lorsqu'un modèle associé a été élaboré, il peut être réutilisé dans les modèles de chaque équipement connecté. Seules les valeurs de certaines temporisations peuvent être spécifiques à un matériel particulier.

Une fois construit, un modèle de protocole doit être considéré comme un modèle générique qui est reconduit, éventuellement avec quelques modifications, dans chaque matériel qui implémente ce protocole.

Par exemple pour un driver Uni-Telway développé sur un Micro-Vax de Digital, les modèles des coupleurs Uni-Telway qui ont été détaillés pourront être utilisés. Par contre, au niveau du système d'exploitation propre à l'équipement et de la description de l'applicatif, une étude spécifique du fonctionnement doit être menée.

III.4. Exemples de résultats

Dans le paragraphe I.1.5. de ce mémoire, nous avons présenté un exemple industriel d'architecture de commande où il était nécessaire de connaître les performances. Cette application est traitée en détails en annexe 2, mais nous présentons ici quelques résultats de simulation afin d'illustrer l'exploitation des modèles qui viennent d'être exposés.

III.4.1. Vérification de bon fonctionnement

Développer et mettre au point les mécanismes d'échanges de données au niveau d'une architecture de commande est une tâche toujours délicate, car durant les phases d'analyse et de développement, on ne dispose jamais de l'architecture complète pour faire des tests en vraie grandeur.

C'est donc dans la phase finale de mise au point sur site que les erreurs de conception apparaissent. Il est alors toujours long et fastidieux d'apporter les modifications nécessaires sur le logiciel. Et ceci d'autant plus que les causes des dysfonctionnements sont toujours difficiles à déceler à cause de l'espacement géographique des équipements.

Un modèle des échanges de données sur une architecture de commande autorise une observation simultanée du fonctionnement de tous les équipements. Durant les phases d'analyse et de conception, l'utilisation d'un tel modèle permet de découvrir rapidement les erreurs dans l'organisation des échanges entre les stations. Celles-ci peuvent provenir d'un mauvais séquencement logique des échanges, mais aussi de retards sur les données liés aux performances de l'architecture.

Sur notre exemple, la simulation du modèle a permis de détecter une mauvaise synchronisation, liée au caractère temporel des échanges, au niveau de la définition de l'organisation des flux de données.

III.4.2. Temps de réponse de l'applicatif

Pour évaluer les performances du protocole applicatif, nous avons mesuré le temps qui sépare deux attributions successives du bus à une même station. Ce délai est appelé Temps de Cycle Applicatif (TCA). Connaître le TCA est très important, car il a une influence directe sur les fréquences auxquelles les stations peuvent envoyer leurs requêtes : un automate n'est autorisé à émettre que lorsque le médium lui est attribué.

Nous avons dans un premier temps mesuré le TCA "à vide", c'est à dire sans échange de message : dès qu'une station dispose du droit de parole, elle le restitue immédiatement.

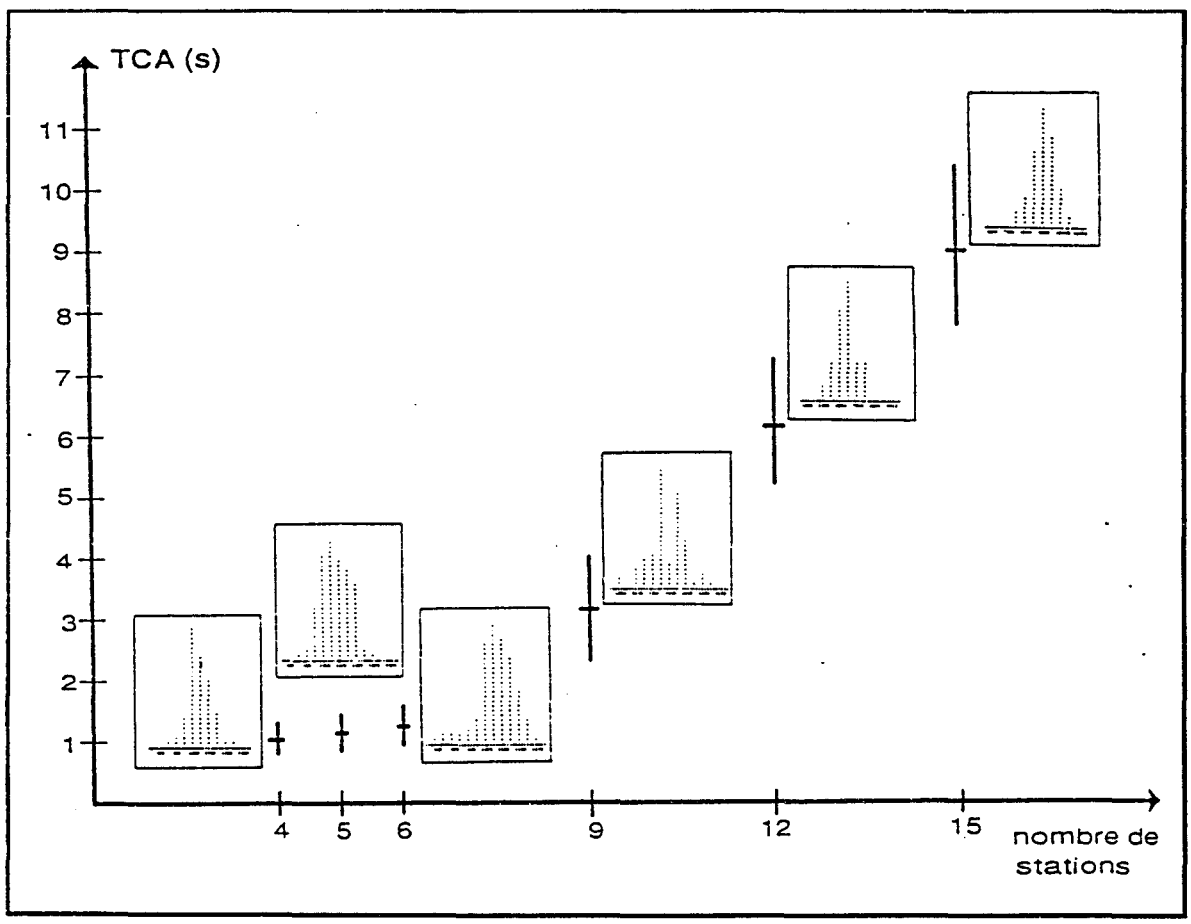


Figure II.68 : *Temps de Cycle Applicatif (TCA) "à vide"*

Nous avons ensuite mesuré le TCA "en charge", qui correspond au fonctionnement réel de l'installation : chaque station exécute ses requêtes de télélecture et de téléchargement.

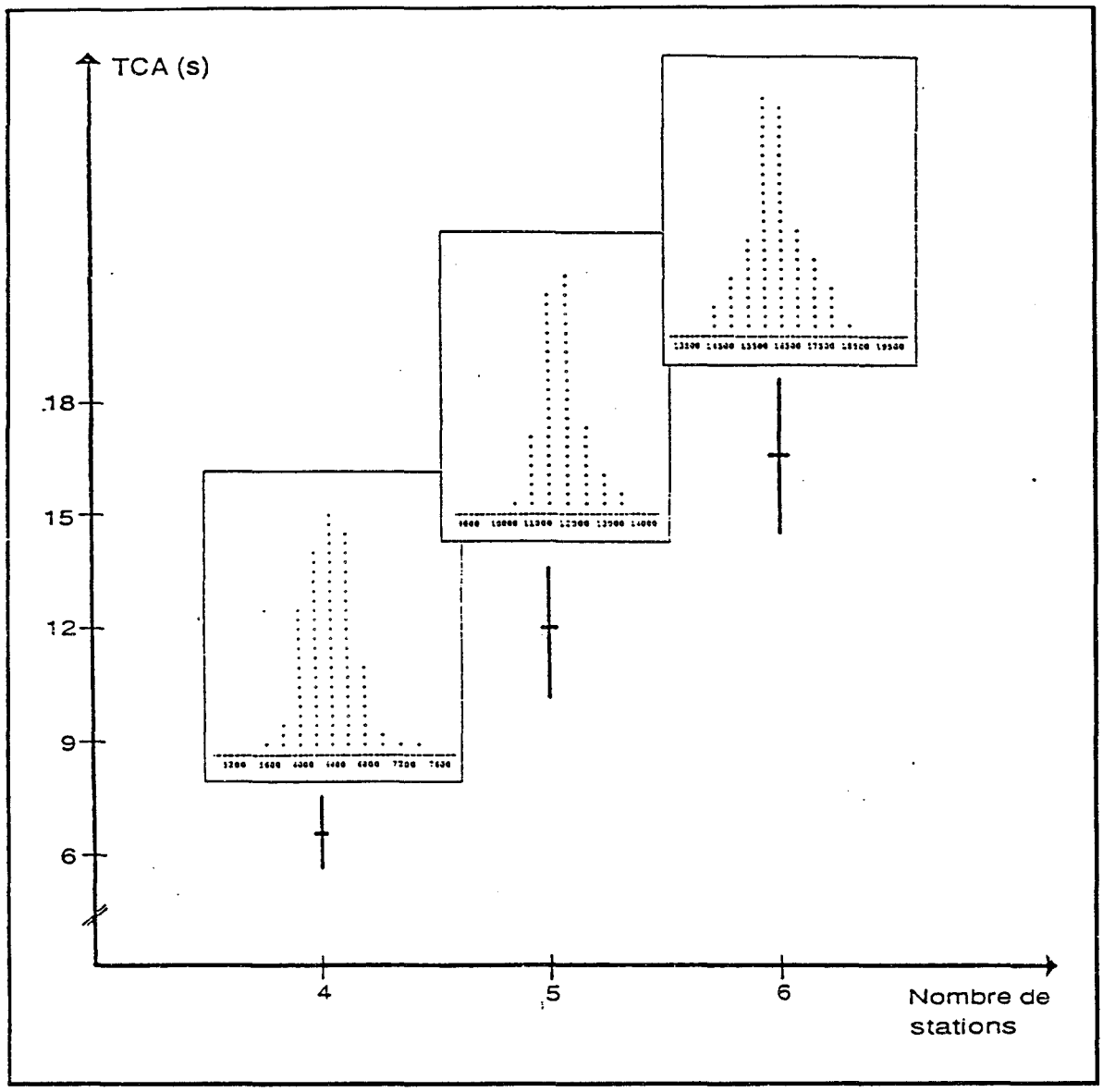


Figure II.69 : *Temps de Cycle Applicatif (TCA) "en charge"*

Chaque paramètre mesuré est caractérisé par sa valeur moyenne et la dispersion autour de cette moyenne. Afin d'observer l'influence du nombre de stations sur les performances, nous avons, pour chaque paramètre, regroupé sur un même graphique les valeurs obtenues pour des scénarios de simulation qui correspondent aux phases successives de l'automatisation.

Les valeurs élevées obtenues s'expliquent notamment par le temps de cycle de 100 ms adopté pour les automates. Dans cette application, les requêtes "critiques" du point de vue des temps de réponse sont envoyées par anticipation. Le délai d'anticipation étant supérieur au TCA évalué, les simulations montrent donc que le système proposé offre des performances adaptées au process commandé.

IV. VERS UN OUTIL D'ASSISTANCE A LA CONCEPTION

Nous avons présenté dans ce mémoire, une démarche permettant d'aborder la modélisation des communications d'architecture de commande d'atelier avec un formalisme réseaux de Petri étendus. Si elle donne des résultats tout à fait intéressants quant aux possibilités d'expression du formalisme employé et aux résultats obtenus après exploitation des modèles, il n'est pas envisageable de proposer aux utilisateurs concernés par cette démarche (Cf. chapitre I) d'utiliser l'outil SEDRIC. En effet, ces derniers doivent pouvoir construire un modèle d'architecture par simple assemblage de modules, lesquels masquent les modèles réseaux de Petri sous-jacents. Une telle approche n'est pas possible avec SEDRIC, qui s'adresse exclusivement à des spécialistes des réseaux de Petri.

C'est pourquoi ce travail de thèse constitue à notre sens une base de départ pour développer un outil industriel de modélisation/simulation d'architecture de commande et de pilotage, qui soit réellement utilisable et utilisé par les personnes en charge de cette fonction au niveau du groupe PSA Peugeot Citroën. A partir de l'expérience acquise lors de nos travaux de modélisation, nous proposons maintenant de présenter dans les grandes lignes les spécifications d'un tel outil.

IV.1. Spécification de l'outil

S'il y a encore quelques années, la simulation de flux de production était réalisée au moyen d'outils génériques, la tendance actuelle est de proposer à l'utilisateur des outils dédiés, simples d'emploi, où les matériels de production et de transport propres à chaque métier sont déjà modélisés dans l'outil [CER 88]. Les avantages d'une telle approche sont évidents : accessibilité de la démarche à un plus grand nombre de personnes, réduction des délais pour construire et exploiter un modèle, limitation voire suppression des erreurs de modélisation etc. Notre volonté est de proposer une approche comparable pour l'évaluation des performances des architectures informatiques de commande et de pilotage d'atelier.

L'outil doit être organisé autour d'une bibliothèque évolutive de modèles d'équipements de commande et de pilotage, pour garantir une adaptation aux nouveaux équipements d'automatisme qui vont apparaître sur le marché au fur et à mesure de l'évolution des techniques. Celle-ci doit être suffisamment complète pour être en mesure de décrire les architectures mises en oeuvre par le groupe PSA. Il est donc essentiel de définir des règles pour y adjoindre de nouveaux modèles qui restent compatibles avec ceux qui y existent déjà (généricité et maintenabilité).

Le fonctionnement d'un tel outil se situe donc sur deux niveaux qu'il est essentiel de bien distinguer :

1 - niveau "***modélisateur système***" pour créer et valider des nouveaux modèles génériques d'équipements et les archiver en bibliothèque,

2 - niveau "***concepteur d'architecture***" pour construire des modèles d'architecture de commande et de pilotage à partir des équipements modélisés en bibliothèque, simuler le système complet et exploiter les résultats ainsi obtenus.

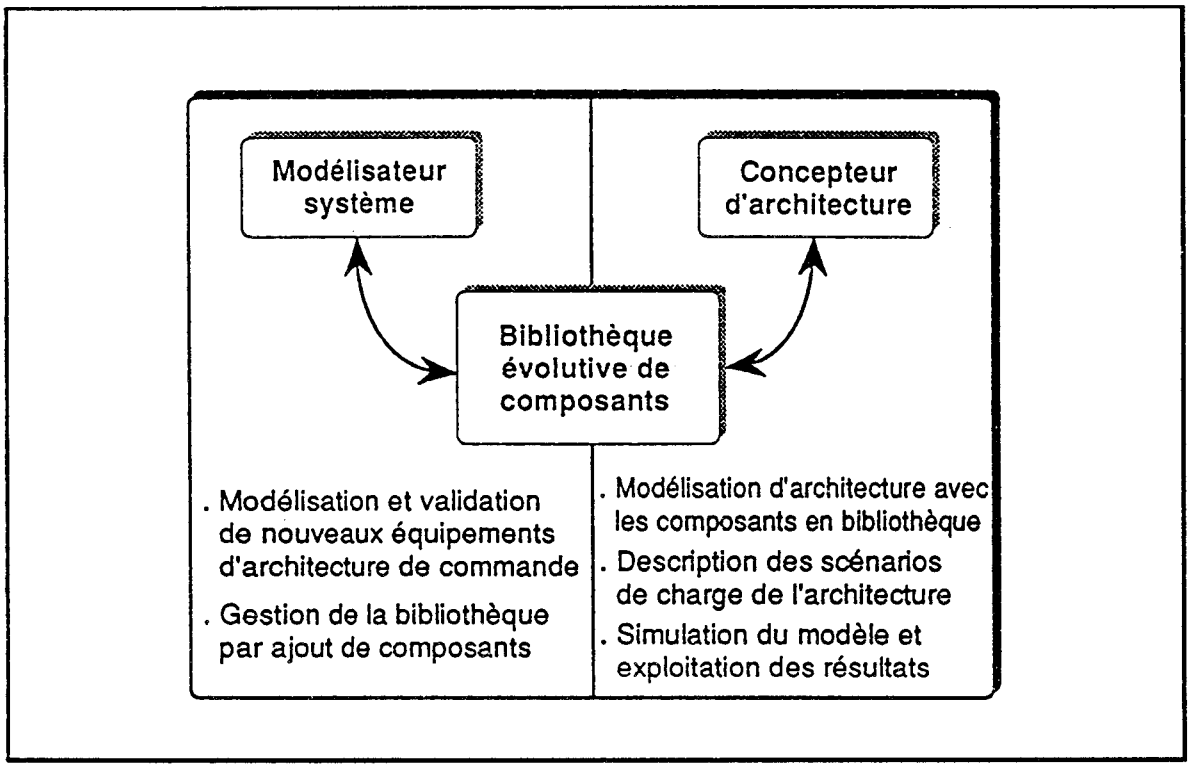


Figure II.70 : *Organisation d'un outil de modélisation d'architecture de commande*

IV.2. Principes d'utilisation de l'outil

Pour décrire l'architecture à étudier, l'utilisateur dispose d'une interface graphique conviviale à base d'icônes, de menus déroulants et de formulaires de saisie. La construction d'un modèle se décompose en deux parties :

1 - description de l'architecture matérielle : l'utilisateur dispose d'une palette d'icônes pour construire le graphe représentant l'architecture de commande. Chaque icône correspond à un équipement et est associé à un des modèles de la bibliothèque. Un formulaire de saisie est attaché à chacun des éléments; il peut être ouvert à tout moment pour consulter ou modifier les caractéristiques du composant considéré.

2 - définition des flux d'informations et des scénarios de charge : la description de la partie applicative consiste à définir les échanges de messages en provenance et/ou à destination du process à piloter. Pour cela, l'utilisateur dispose de menus déroulants et de formulaires de saisie pour la sélection et la caractérisation de modèles prédéfinis ainsi que d'un éditeur de texte traditionnel pour décrire finement les applicatifs au moyen d'un langage algorithmique.

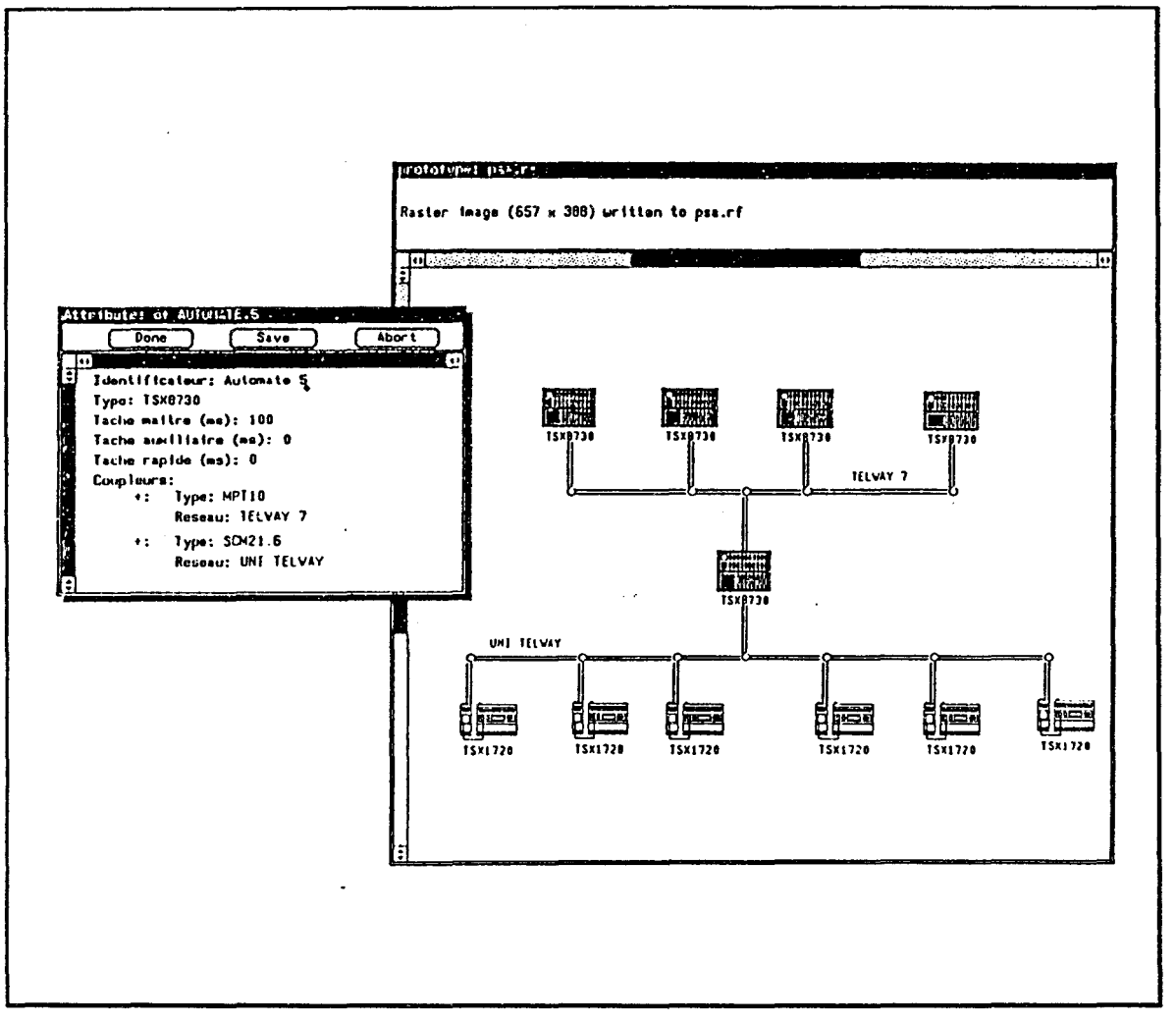


Figure II.71 : *exemple de vue graphique de l'outil proposé*

A partir de cette description, l'outil permet de générer le code correspondant pour lancer et contrôler les simulations. Celles-ci peuvent éventuellement être accompagnées d'animations graphiques, par exemple pour visualiser les données échangées entre les stations. Cette fonctionnalité peut s'avérer extrêmement utile durant les phases de mise au point des scénarios de dialogue inter-équipements. Un module de post-traitement aide à l'interprétation des résultats de simulation en proposant la visualisation et la consultation des résultats sous formes graphiques (courbes, histogrammes etc.) ou tableaux de valeurs.

Le schéma fonctionnel d'un tel outil est décrit sur la figure suivante :

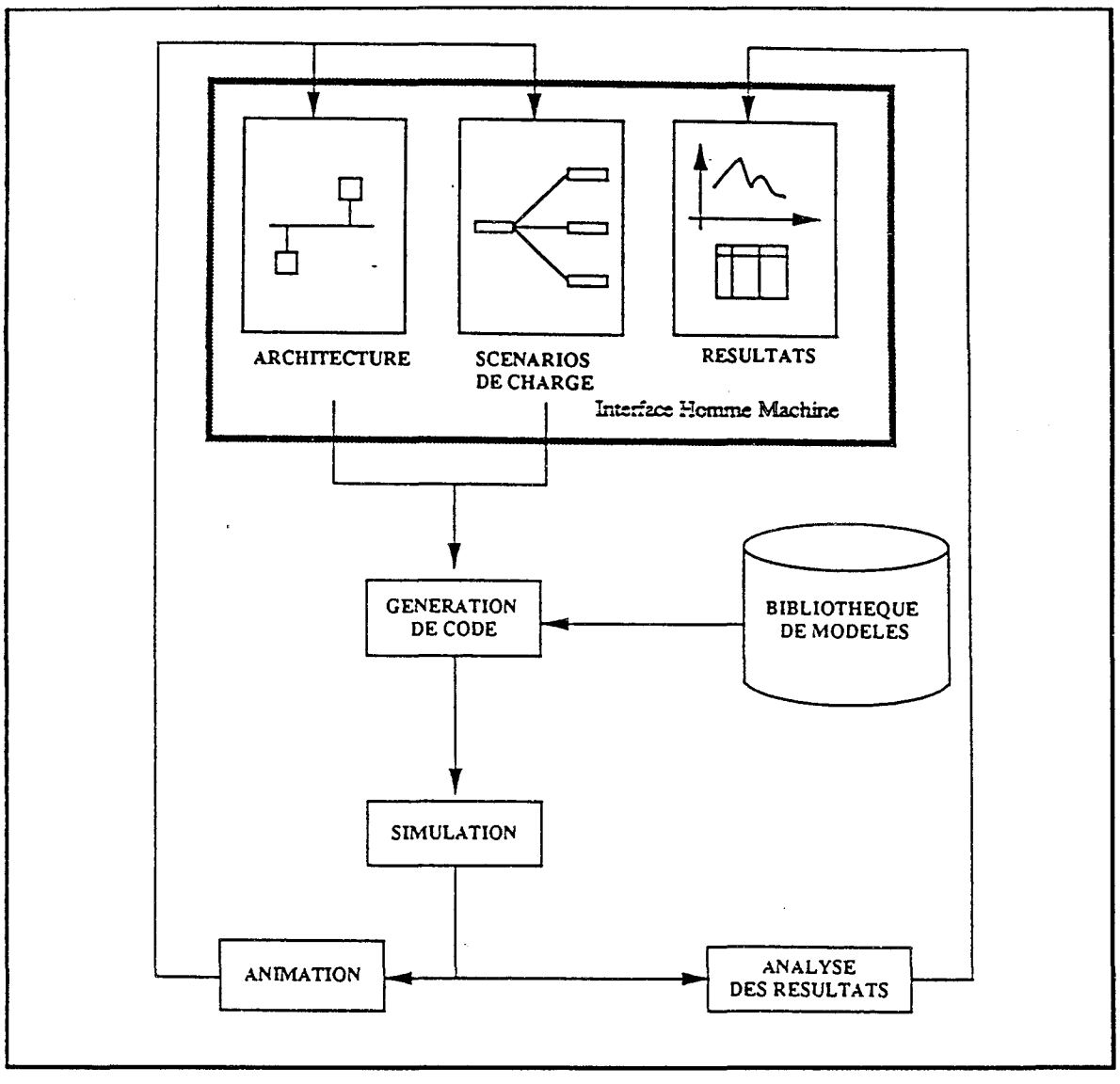


Figure II.72 : *schéma fonctionnel de l'outil proposé*

IV.3. Bibliothèque de composants

Dans le cadre de nos travaux de thèse, nous avons obtenu toutes les informations nécessaires de la part de la société Télémécanique sur le fonctionnement de ses équipements pour être en mesure de les modéliser de manière précise. Il est clair que d'autres fournisseurs d'équipements de contrôle/commande ne seront pas toujours disposés à divulguer l'ensemble des caractéristiques de leur matériel, soit pour des raisons de secret industriel, soit simplement parce que les informations requises ne sont pas disponibles telles quelles et que l'effort à fournir peut paraître trop important.

Il faut pourtant dans ce cas être capable de proposer un modèle suffisamment fidèle, du même niveau de détails que les autres, afin de ne pas remettre en cause le fonctionnement général de l'outil. Il nous paraît donc essentiel de bâtir une méthodologie de modélisation d'un composant dont le comportement est mal identifié. Cette méthodologie ne peut alors reposer que sur la prise de mesures sur l'équipement lui-même afin de caractériser son comportement de manière "externe".

Une caractéristique importante des architectures de commande d'atelier chez PSA Peugeot Citroën est l'hétérogénéité des matériels utilisés. Pour garantir une pérennité à l'outil il est indispensable que les nouveaux modèles qui seront progressivement introduits en bibliothèque restent totalement compatibles avec ceux déjà existants. En particulier, ils devront pouvoir s'intégrer, si nécessaire, dans une description d'architecture, comportant des modèles d'équipements plus anciens. Il est donc capital de définir sans ambiguité et de façon figée l'organisation et les principes à appliquer pour intégrer un nouveau composant en bibliothèque pour être en mesure de garantir la totale compatibilité avec l'existant.

En fonction de l'intérêt manifesté par les utilisateurs, et de la disponibilité de nouveaux produits, notamment basés sur des technologies FIP ou MAP, il sera nécessaire de se donner les moyens de les intégrer en bibliothèque. Comme ces matériel impliquent des philosophies nouvelles sur la conception des systèmes de commande et de pilotage, il est également primordial d'étudier et de définir de quelle manière l'approche de simulation proposée doit être utilisée pour aider à la configuration et au dimensionnement de telles architectures.

IV.4. Conclusion

L'outil doit être organisé autour d'un moteur de simulation de système à événements discrets. Celui-ci doit pouvoir s'intégrer facilement avec un environnement de développement de type objet pour être en mesure de coder rapidement toute la partie interface utilisateur. Comme les unités utilisées dans les modèles de systèmes informatiques sont faibles (milliseconde, voire microseconde) le nombre d'événements générés pour simuler quelques minutes de fonctionnement réel est important. Le moteur de l'outil doit donc être performant pour proposer des durées de simulation acceptables pour l'utilisateur. Choisir un moteur réseaux de Petri étendus constituerait à notre sens une bonne solution. Mais, la société Techlog ayant abandonné le produit SEDRIC, pour des raisons de marché insuffisant, il n'est pas forcément facile de trouver d'autres pôles de compétences dans le domaine.

Sur un plan commercial, nous sommes persuadé qu'un outil, tel que nous venons de le détailler, est susceptible d'intéresser d'autres utilisateurs que le groupe PSA. Par exemple, EDF s'intéresse également à l'évaluation de performances de ses systèmes de contrôle/commande par simulation [COR 91]. On pourrait très bien imaginer que la bibliothèque soit adaptée en fonction des besoins particuliers de chaque client, en y intégrant les modèles spécifiques des équipements qu'il utilise.

Cependant, même si un tel outil était développé, toute la difficulté est d'enrichir régulièrement la bibliothèque avec des modèles des nouveaux matériels disponibles. Ce travail ne peut être réalisé que par un spécialiste de la modélisation de tels systèmes. Nous ne pensons pas qu'il soit justifié de développer une telle compétence chez les utilisateurs potentiels de l'outil, car la vocation de PSA par exemple est de fabriquer des voitures et non des produits de simulation.

C'est pourquoi, la seule solution qui nous semble viable pour développer et assurer le succès d'un tel outil est d'en confier la réalisation et la promotion à une société de services ayant une activité dans le domaine de la simulation. La maintenance et la mise à jour régulière de la bibliothèque de composants seraient alors à sa charge, en fonction des besoins de ses clients.

CONCLUSION

Dans ce chapitre, nous avons exposé une manière pour aborder la modélisation des échanges de données au sein d'une architecture de commande et de pilotage d'atelier de production manufacturier. L'objectif des simulations est d'évaluer, dès la phase de conception, les performances relatives aux échanges de données, pour de tels systèmes. Cette approche doit permettre de valider les solutions adoptées avant leur installation sur site.

Après avoir justifié le choix d'un formalisme réseaux de Petri interprétés, colorés et temporisés, nous avons montré comment utiliser ce formalisme pour modéliser des équipements d'automatisme. Notre propos a été largement illustré par la mise en oeuvre de modèles de produits de la société Télémécanique (automates programmables et coupleurs de communication). Chaque modèle est construit une seule fois, et constitue ensuite un object générique, utilisé pour chaque étude d'architecture où le composant intervient.

Dans le troisième et dernier chapitre de ce mémoire, nous utilisons ces modèles pour étudier les performances d'une architecture de suivi de fabrication qui a été mise en place récemment dans le centre de production de Rennes/La-Janais. Au travers de cet exemple nous illustrons de quelle manière les modèles des équipements, construits avec des réseaux de Petri, s'interfacent avec une description des applicatifs et de l'environnement.

Les modèles complets d'architecture sont formés de quelques dizaines de places. En effet, les systèmes sont souvent constitués avec beaucoup d'équipements identiques : on utilise alors un seul graphe de processus dans lequel circulent indépendamment autant de jetons d'état qu'il existe d'équipements modélisés.

Les temps de simulation, obtenus avec l'outil SEDRIC sont importants. Selon la complexité des modèles étudiés, il nous a fallu sur une station SUN 4/110 entre une et dix heures de temps CPU pour simuler une heure de fonctionnement effectif. Ces résultats s'expliquent par le nombre très important d'événements générés, compte-tenu de la précision des modèles : les durées des états sont exprimés en millisecondes.

CHAPITRE III

Application à un exemple industriel du groupe PSA Peugeot Citroën

INTRODUCTION

Nous présentons dans ce chapitre une mise en oeuvre, sur un exemple industriel, de la démarche de conception d'architecture de commande d'atelier présentée au début de ce mémoire. Cette étude a été réalisée, durant la phase d'élaboration du système, en collaboration étroite avec des personnes de la direction des méthodes d'automobiles Citroën de Rennes/La-Janais [EVE 91].

Nous nous sommes attachés, tout au long de ce chapitre, à décrire les phases de réflexion successives qui ont conduit à la genèse du système installé sur site. Notre objectif est de montrer à quel moment de l'étude, la démarche de simulation présentée dans ce mémoire a pu être mise en oeuvre, quelles sont les informations nécessaires à la construction et à la simulation des modèles et quels sont les résultats d'une telle approche.

Ce chapitre comporte quatre volets :

- Dans une première partie, nous présentons l'atelier de montage ANDON, avec son système de suivi de fabrication. Les architectures matérielle et fonctionnelle des constituants d'automatisme à mettre en place sont détaillées afin de disposer de toutes les informations nécessaires à l'étude de cette installation.

- La deuxième partie expose la modélisation de l'architecture du système de suivi de fabrication ANDON. Ces travaux s'appuient sur les modèles réseaux de Petri d'équipements d'automatisme qui ont été exposés dans le deuxième chapitre de ce mémoire.

- Cette modélisation est exploitée dans la troisième partie pour simuler le fonctionnement de l'installation et en évaluer les performances. Les résultats ainsi obtenus sont comparés à des mesures réalisées sur le système réel.

- La dernière partie propose d'élargir l'étude en simulant l'architecture du système complet au niveau de l'atelier. Nous y présentons la façon dont ces travaux pourraient être menés, à partir de ceux déjà réalisés.

I. SYSTEME DE SUIVI DE FABRICATION "ANDON"

I.1. "Qualité Totale" chez Citroën

C'est en 1951 que le Docteur Armand Feigenbaum, alors vice-président du service qualité de la société américaine General Electric, inventa le qualificatif T.Q.C. (Total Quality Control) [HER 89]. Il avait observé que l'inspection classique, bien que rationalisée depuis 1940 par l'application de méthodes statistiques, ne suffisait pas à garantir la qualité. A. Feigenbaum élargit donc le concept de qualité en la considérant comme un choix stratégique concernant toute l'entreprise au même titre que les politiques financières ou commerciales.

C'est cette idée qui a servi de base au développement au Japon du C.W.Q.C. (Company Wide Quality Control). Ce système implique que les services et les personnels de tous niveaux soient concernés par la mission de l'entreprise qui est de fournir un produit de qualité répondant aux besoins des utilisateurs.

La méthode "Qualité Totale" Citroën, dont les premières études remontent à 1979, reprend ces principes en se basant sur le fait que la qualité doit être intégrée au processus de fabrication. Auparavant, le contrôle qualité et la fabrication étaient deux fonctions distinctes : il s'agissait essentiellement de vérifier a posteriori la qualité du produit.

L'idée directrice est de se donner les moyens de mener des actions préventives pour maîtriser la qualité. Pour cela, il est nécessaire de disposer d'informations précises et fiables, tout au long du processus de production. Les moyens de mesure doivent être simples et facilement exploitables par le personnel de production. Munis de ces informations, les agents d'atelier peuvent mesurer le niveau de qualité de leur production et pratiquer l'auto-contrôle. Après analyse des données relevées dans les ateliers, des remèdes avec des plans d'action peuvent être établis pour améliorer le processus de fabrication.

C'est dans ce contexte qu'est né, le système baptisé ANDON qui permet, depuis 1984, la surveillance en temps réel des anomalies et problèmes survenus lors des différentes phases opératoires du montage terminal d'un véhicule. A différents niveaux hiérarchiques, le système apporte les informations essentielles à l'accomplissement des tâches de chacun. Suite à l'extension et à la modification des lignes de production pour permettre l'assemblage d'un nouveau véhicule, le système ANDON existant a nécessité une refonte importante.

I.2. Lignes de montage ANDON

Le choix de l'architecture des systèmes d'automatisme est orienté par le flux produit et l'organisation humaine dans l'atelier. Pour comprendre les choix techniques réalisés et le contenu des informations présentées dans le modèle détaillé au paragraphe II de ce chapitre, nous allons donc commencer par décrire les principes de fonctionnement des lignes de montage ANDON.

I.2.1. Atelier de montage véhicules RM1/RM2

C'est vers l'atelier de montage que convergent, pour être assemblées, la carrosserie, la mécanique, et les pièces d'habillage et de garnissage. En sortie de ligne, le véhicule terminé, est stocké en attente d'une livraison au client. L'atelier de montage Citroën RM1/RM2, sur lequel porte notre étude, abrite deux lignes de fabrication modulables, qui assurent des opérations manuelles et d'autres automatisées ou robotisées.

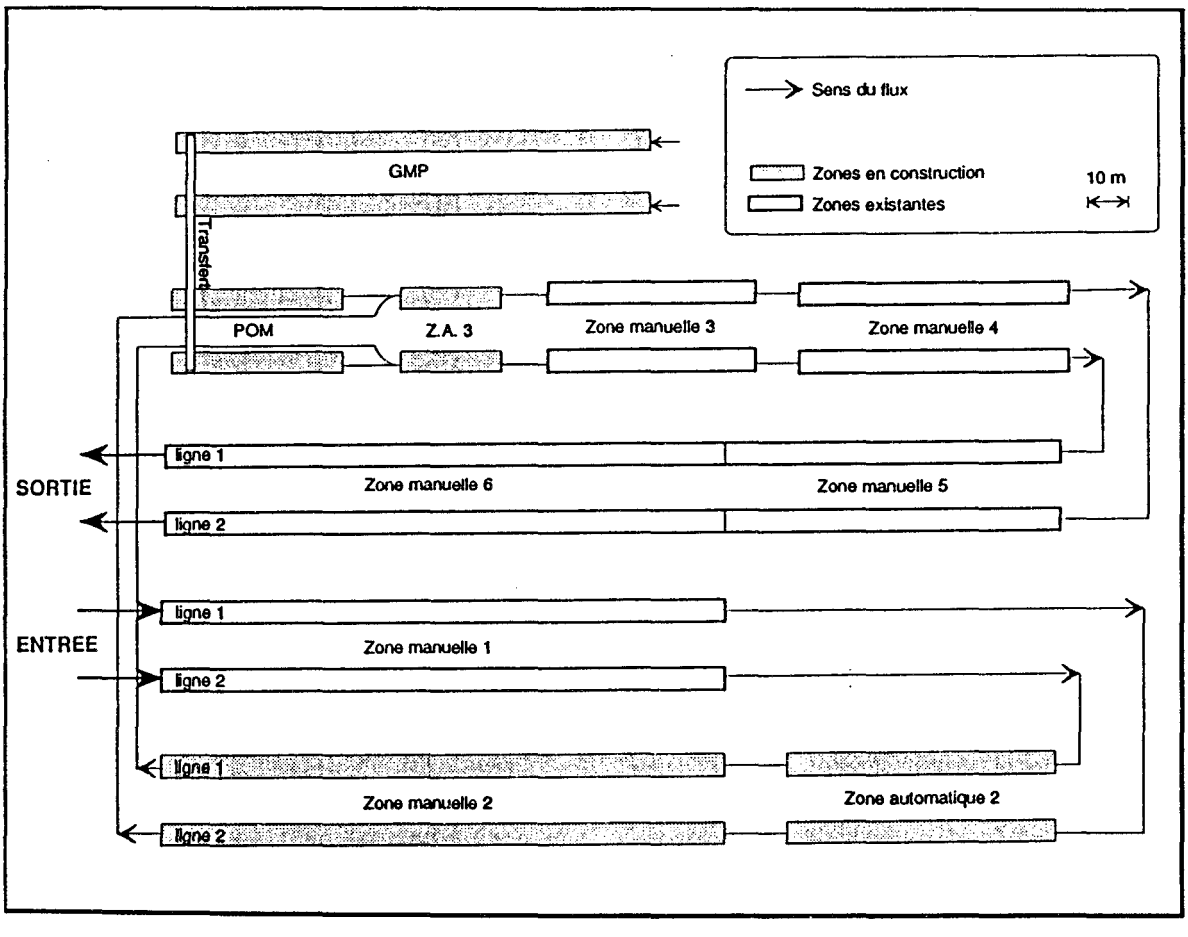


Figure III.1 : *Localisation des zones techniques au sein du montage RM1/RM2*

Le flux de production, identique pour les deux lignes, apparaît comme le cheminement d'une caisse véhicule au travers de zones techniques (Z.T), manuelles (Z.M.) ou automatiques (Z.A.). Le transfert entre les zones est assuré par des convoyeurs aériens. Quelques sous-ensembles sont préalablement assemblés sur des zones de préparation voisines, puis acheminés sur la ligne pour être montés. La figure III.1 représente le circuit des deux lignes, dont les zones techniques sont détaillées dans le tableau suivant.

Zone	Type et fonctionnalité de la zone
ZM1	Montage des faisceaux électriques et de certaines insonorisations
ZA2	Montage du poste de conduite, du pare brise et du pavillon
ZM2	Insonorisation du véhicule
GMP	Préparation et assemblage du groupe motopropulseur (trains de roue, échappement...)
POM	Préparation du bloc moteur, réalisation de l'ensemble de motorisation
ZA3	Mariage de la carrosserie avec l'ensemble de la motorisation
ZM3	Mise en place des liaisons entre la caisse et le groupe motopropulseur
ZM4	Assemblage d'éléments sous le véhicule et dans le compartiment moteur
ZM5	Pose de divers éléments (batterie, garnissage habitacle...)
ZM6	Finitions (sièges, capot, feux AR, garnissage habitacle...)

Tableau III.1 : *Zones techniques de l'atelier RM1/RM2*

Les Z.T. manuelles sont divisées en U.H.T. (unité homogène de travail). Chaque U.H.T. est décomposée en plusieurs pas de travail délimités par un marquage au sol. Les pas sont désignés par un numéro et une lettre (D ou G) en fonction de leur situation à droite ou à gauche de l'axe de la ligne. En général on trouvera un opérateur par pas et par côté de ligne.

I.2.2. Description du système ANDON

Le système ANDON est mis en place sur toutes les zones manuelles de l'atelier de montage qui vient d'être présenté. Il est utilisé par les opérateurs de ligne qui effectuent eux-même l'auto-contrôle de leur travail. Il s'agit d'établir un contact permanent entre ces derniers et leurs assistants intervenants afin de régler le plus rapidement possible tous les problèmes susceptibles de perturber le déroulement normal de leur phase opératoire, de mémoriser ces problèmes et d'en permettre un traitement par une hiérarchisation.

Chaque côté d'un pas de travail est équipé d'un dispositif d'appel sous la forme d'une corde pendulaire située sur le bord extérieur de la ligne. En cas de problème ou de défaut constaté, l'opérateur tire sur la cordelette de son pas : c'est l'appel ANDON. Par un dispositif sonore et lumineux (tableau synoptique affecté à chaque U.H.T., avec deux cases lumineuse par pas), le système alerte une personne d'assistance pour qu'elle intervienne.

L'assistant effectue si possible la retouche sur le pas de travail, et termine par une action, le réarmement ANDON. Pour cela, chaque pas est équipé d'une commande de réarmement située sur un des côtés de la ligne. S'il n'est pas disponible pour intervenir aussitôt, le véhicule continue d'avancer sur un nombre de pas variable suivant le type de ligne. Si l'intervention n'est pas réalisée dans le délai correspondant, la ligne est arrêtée automatiquement et un gyrophare signale l'arrêt ANDON.

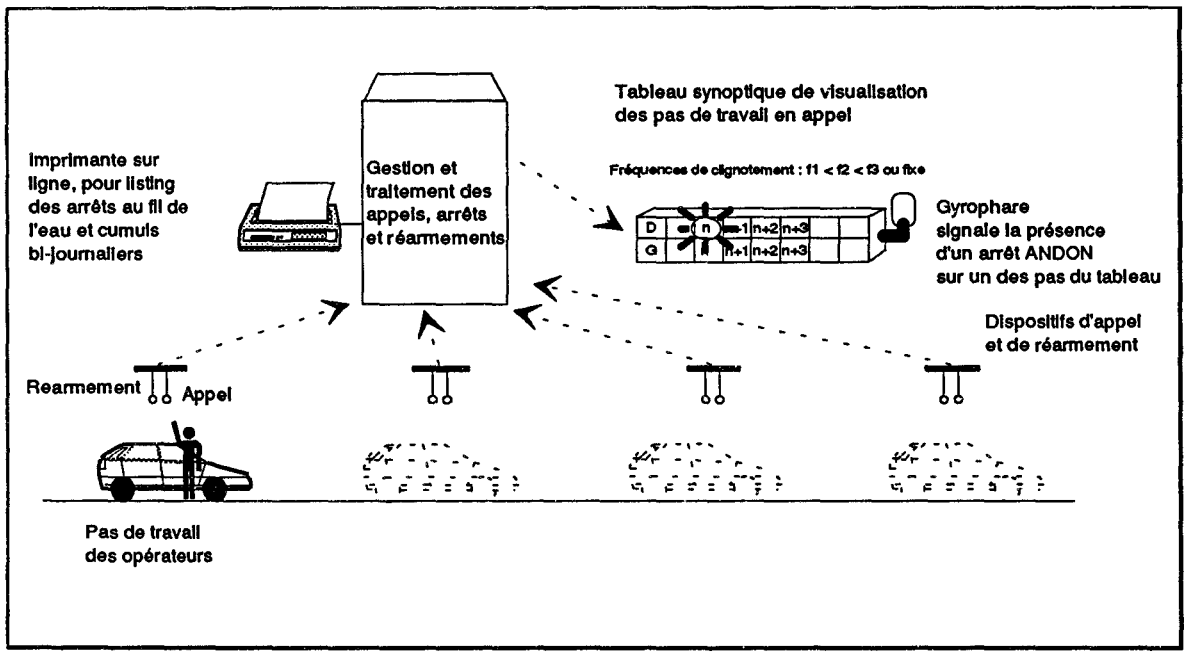


Figure III.2 : *Organisation du système ANDON sur quelques pas de travail*

I.2.3. Fonctionnement de l'installation

Lors du déclenchement d'un appel ANDON au pas n, un klaxon est actionné et la lampe n du tableau synoptique correspondant à ce pas se met à clignoter avec la fréquence f1. A la détection du véhicule sur le pas suivant, la lampe n s'éteint et celle du pas n+1 se met à clignoter à la fréquence f2 qui est supérieure à f1. Lorsque le véhicule atteint le pas n+2 sans qu'aucun réarmement ne soit survenu, la lampe n+1 s'éteint et la lampe n+2 se met à clignoter à une fréquence f3 encore plus rapide que la précédente.

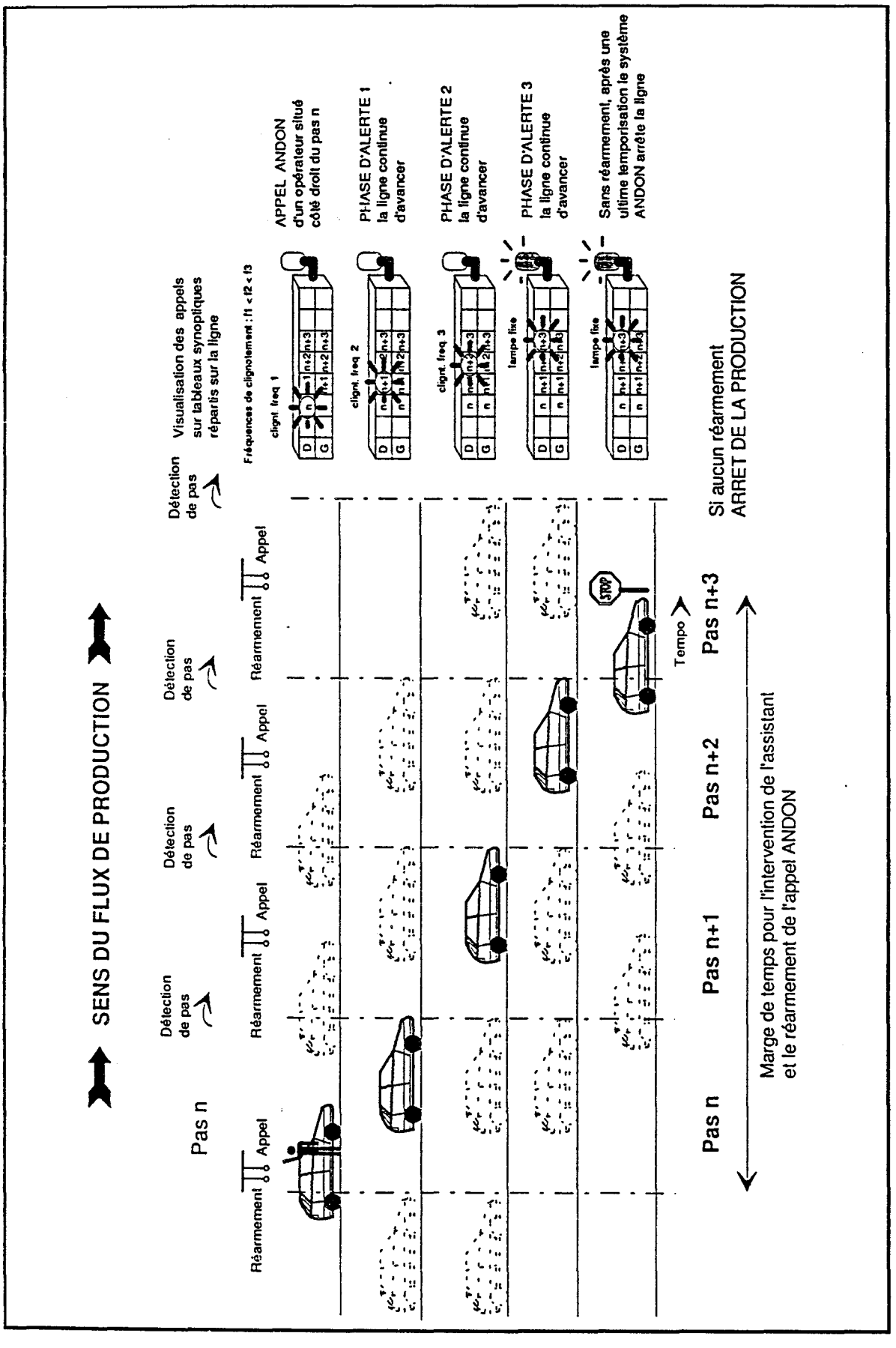


Figure III.3 : *Différentes phases d'alerte du système ANDON*

Comme le montre la figure III.3, le système ANDON permet quatre niveaux d'alerte possibles. Au franchissement du pas n+3, la lampe n+2 s'éteint et la lampe n+3 est allumée fixe. Le gyrophare monté sur le tableau concerné est validé et la ligne arrêtée après une temporisation équivalente au parcours d'un demi-pas.

Lorsqu'une lampe doit clignoter à deux fréquences différentes, la priorité est donnée à l'appel le plus ancien, tout en mémorisant les deux appels. De même, la commande de réarmement efface en premier lieu l'appel le plus ancien.

Tous les appels ANDON sont comptabilisés pour pouvoir ensuite être exploités de manière statistique. Après chaque séance de travail, (à 14 heures pour les équipes du matin et 23 heures pour celles du soir), une synthèse bi-journalière du cumul des arrêts de la zone technique est éditée.

date :02/03 23h00 ZM1 ligne 2

N° affectation	Nb appels	Nb arrêts	Tps arrêt
ANDON 1 G	03		
ANDON 7 G	07	01	2mn03s
ANDON 11 G	15	01	15s
ANDON 16 G	01		
ANDON 17 G	05	02	2mn01s
ANDON 20 G	02	02	1mn18s
ANDON 21 G	10		
ANDON 23 G	01	01	5mn06s
ANDON 25 G	01		
ANDON 1 D	02		
ANDON 12 D	01		
ANDON 19 D	02	01	58s
ANDON 20 D	03		
ANDON 26D	01		
ANDON 30D	02	01	1mn
TOTAL		08	12mn41s

Figure III.4 : *Listing de suivi et synthèse des défauts*

I.3. Architecture du système de suivi de fabrication

I.3.1. Installation existante

Les zones de fabrication équipées du système ANDON sont les Z.T. manuelles des deux lignes de l'atelier. Le traitement des différentes étapes du processus d'assistance est confié à un automate programmable industriel de marque Télémécanique (TSX 67 ou 87); huit automates sont nécessaires pour gérer l'ensemble des zones manuelles.

En plus des informations ANDON, l'automate traite des défauts techniques qui proviennent d'armoires à relais (A.R.) et d'automatismes voisins. Toutes les stations ANDON sont connectées à un automate centralisateur. Cette interconnexion permet d'assurer une visualisation et une centralisation des défauts vers le service de maintenance, qui suivant le cas peut intervenir sur les moyens. Aujourd'hui, ces anciennes zones représentent les 2/3 des zones de l'atelier auxquelles sont ajoutées des zones automatiques et des zones manuelles plus modernes.

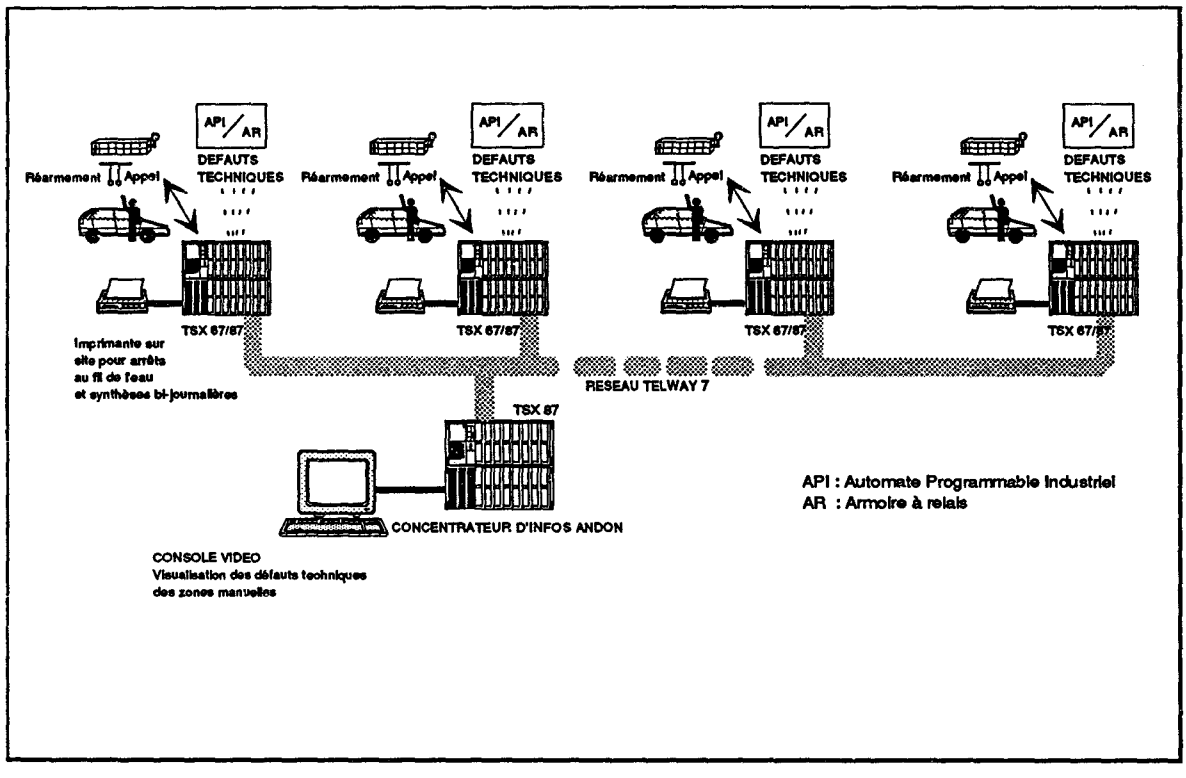


Figure III.5 : *Architecture du système existant*

I.3.2. Nouvelles fonctionnalités demandées

Les travaux réalisés dans l'atelier de montage RM1/RM2 permettront d'assembler un nouveau véhicule sur deux lignes de production. A cause de l'augmentation du nombre de pas de travail il a été nécessaire de créer des zones manuelles entièrement nouvelles. Les réaménagements imposés par les contraintes de montage du nouveau produit ont également imposé des modifications sur les zones déjà existantes.

Le système ANDON, qui était en place, a donc nécessité une refonte importante. De plus, dans un soucis d'efficacité, de nouveaux besoins ont été émis par les services de maintenance et de fabrication.

I.3.2.a Supervision de l'atelier

Jusqu'à présent, les défauts techniques, et les informations ANDON étaient regroupées sur une même console de visualisation reliée à un automate concentrateur. Pour rendre la fonction maintenance plus efficace, il est apparu nécessaire de centraliser tous les problèmes techniques de l'atelier dans une unique salle de contrôle, où toutes les informations sont enregistrées et analysées.

I.3.2.b Suivi de fabrication ANDON

Afin de mieux traiter les informations enregistrées par chacun des automates ANDON, les responsables de fabrication souhaitent désormais obtenir une gestion des données à un niveau supérieur. D'une part, tous les résultats obtenus pour chacune des zones manuelles doivent être centralisés au niveau de l'atelier RM1/RM2. D'autre part, l'assistant de production affecté à une U.H.T. donnée doit disposer du bilan des appels et arrêts propre à son U.H.T. C'est ainsi que l'application ANDON devra renseigner les utilisateurs en proposant les documents de synthèse de suivi périodique et journalier sur les appels et arrêts de chaque U.H.T.

I.3.3. Nouvelle architecture proposée

L'architecture complète du système de maintenance et de suivi de fabrication, pour l'atelier RM1/RM2 est représentée sur les figures des pages 13 et 14 de ce chapitre.

I.3.3.a Système de supervision d'atelier SIMONA

Excepté le système ANDON, l'atelier RM1/RM2 est équipé d'automates programmables de marque Siemens [SIE 89]. C'est donc le réseau propriétaire de ce constructeur, SINEC H1, qui a été retenu pour relier ces équipements à un superviseur, en l'occurrence le produit SIMONA [SIE 86]. Ce dernier fonctionne sur un mini-ordinateur SICOMP M56 et assure le traitement en temps réel et l'archivage des données.

Une interface spécifique a du être définie pour remonter les défauts techniques depuis les automates ANDON existants vers le superviseur. Après une analyse de la disposition géographique des équipements dans l'atelier, la solution adoptée consiste à mettre en place un automate Siemens 135U comme interface pour la transmission des défauts vers la supervision générale.

I.3.3.b Reconduction pour les zones existantes

Pour les zones manuelles existantes, les automates ANDON sont conservés. Leur programme applicatif doit cependant subir des modifications pour s'adapter à la nouvelle architecture. Les défauts techniques ne sont plus acheminés via Telway 7 vers l'automate concentrateur ANDON, mais sont transmis vers le réseau SINEC H1. Les TSX67/87 dialoguent par des liaisons tout ou rien avec l'automate Siemens 135U qui véhicule l'ensemble des défauts des zones manuelles ZM3, ZM4, ZM5 et ZM6 vers le superviseur. Le réseau Telway 7 est utilisé par les stations ANDON pour envoyer, pour synthèse, les événements survenus sur la ligne vers l'automate concentrateur.

I.3.3.c Etude d'une solution adaptée aux nouvelles zones

Le critère économique de cette étude étant des plus important, une analyse comparative de différentes solutions a été effectuée pour minimiser les coût de l'installation. Trois possibilités ont été envisagées pour assurer l'échange d'informations entre les automates ANDON des nouvelles zones manuelles et les lignes de montage (appel, réarmement, tableau synoptique etc.) :

1 / Raccordement par liaison T.O.R :

En reprenant la solution existante sur les zones rénovées, il apparaît, au vue de la longueur des câbles nécessaires (plusieurs centaines de mètres !) que les coûts de raccordement sont très élevés. En effet, pour ces zones rénovées, l'ensemble des capteurs sont reliés directement par des câbles vers une armoire principale ANDON affectée généralement à une, voire deux zones techniques.

2/ Entrées-sorties déportées par liaison série :

Les différents systèmes d'entrées-sorties déportées permettent de réduire dans des proportions importantes les coûts de câblage. Pour le système ANDON sur les nouvelles zones manuelles, une étude de faisabilité a été réalisée avec des boîtiers d'entrées-sorties de marque Entrelec. Cette solution présente un intérêt économique du point de vue du matériel, mais nécessite le développement d'un programme d'application (acquisition, protocole d'échange, traitement etc.). De plus, le grand nombre de boîtiers nécessaires pour l'installation laisse entrevoir des temps de réponse élevés, qu'il est difficile d'évaluer sans une étude approfondie.

3 / Mise en place de mini-automates TSX 17-20 [TEL 88d]

En reprenant la même philosophie que celle des entrées-sorties déportées, cette troisième solution consiste à répartir les traitements sur des petits automates Télémécanique de type TSX 17-20. Ceux-ci assurent alors la gestion des informations ANDON sur quelques pas de travail des nouvelles zones manuelles.

La compatibilité matérielle de ces automates avec la gamme des produits Télémécanique permet leur interconnexion via le bus Uni-Telway. Quelques automates TSX 17 répartis sur une ou plusieurs zones assurent la détection des événement sur un nombre limité de pas et la transmission des données vers l'automate ANDON de la zone considérée.

En comparaison avec la première solution, celle-ci fait apparaître un gain financier de l'ordre de 20 %. Elle est donc retenue pour équiper les nouvelles zones ZM2.1, ZM2.2, GMP.1, GMP.2, POM.1 et POM.2. En fonction du nombre de pas à gérer et des temps de dialogue nécessaires lors des échanges entre automates, la configuration suivante est envisagée :

- 1 mini-automate TSX 17 pour la gestion d'un ensemble de 14 pas,
- 1 automate TSX 87 pour centraliser les informations provenant de 8 mini-automates TSX 17
- 2 réseaux Uni-Telway de 8 TSX 17 chacun pour permettre la connexion avec les automates ANDON

Le tableau suivant donne la répartition des pas entre ces automates sur les nouvelles zones techniques :

Zone Technique	Nombre de TSX 17	Nombre de pas
ZM2 ligne 1 (ZM2.1)	4	55
ZM2 ligne 2 (ZM2.2)	4	55
GMP ligne 1 (GMP.1)	3	39
POM ligne 1 (POM.1)	1	13
GMP ligne 2 (GMP.2)	3	39
POM ligne 2 (POM.2)	1	13

Tableau III.2 : *Répartition des automates sur les nouvelles Z.T.*

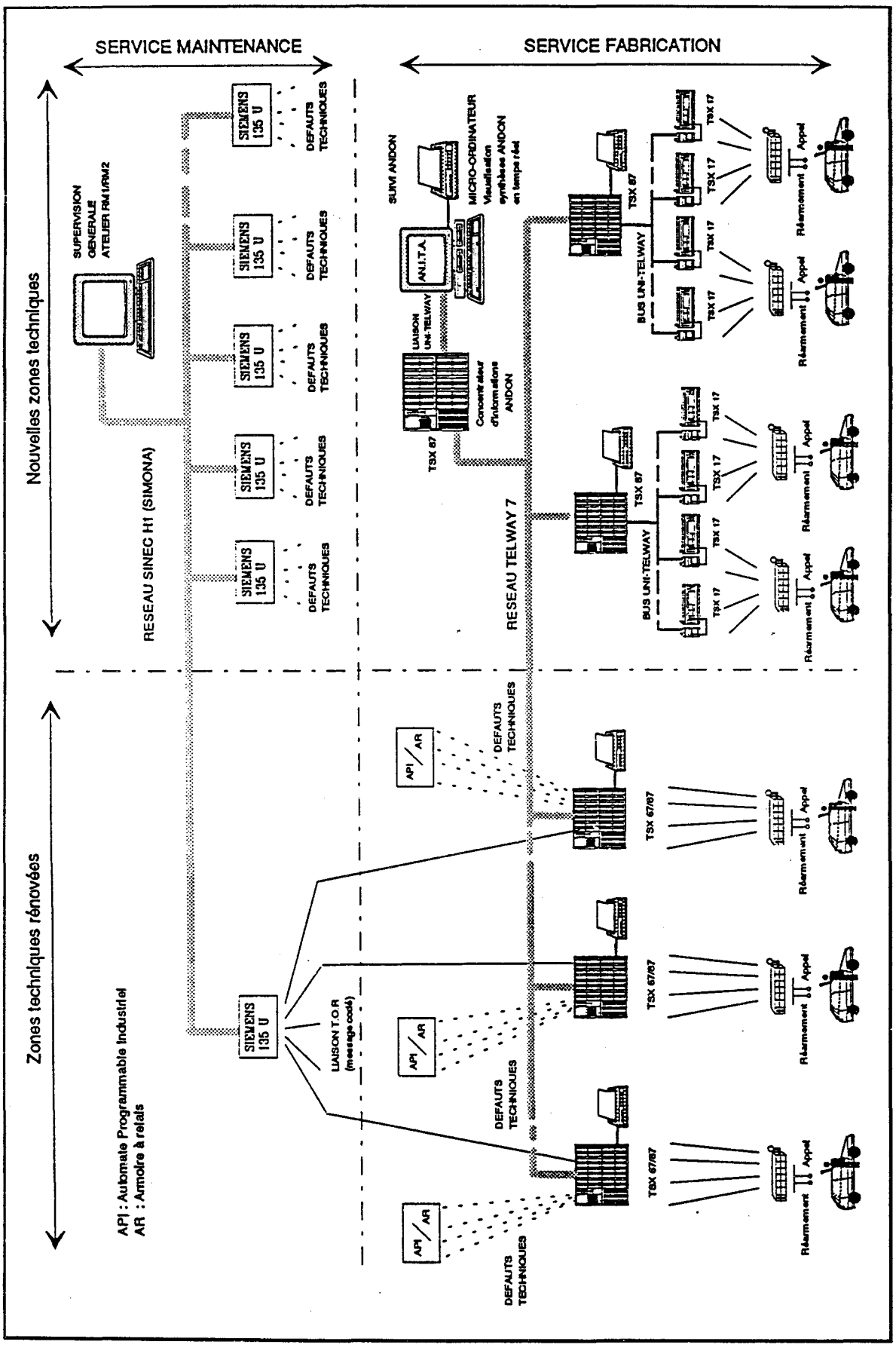


Figure III.6 : *Architecture complète de l'atelier RM1/RM2*

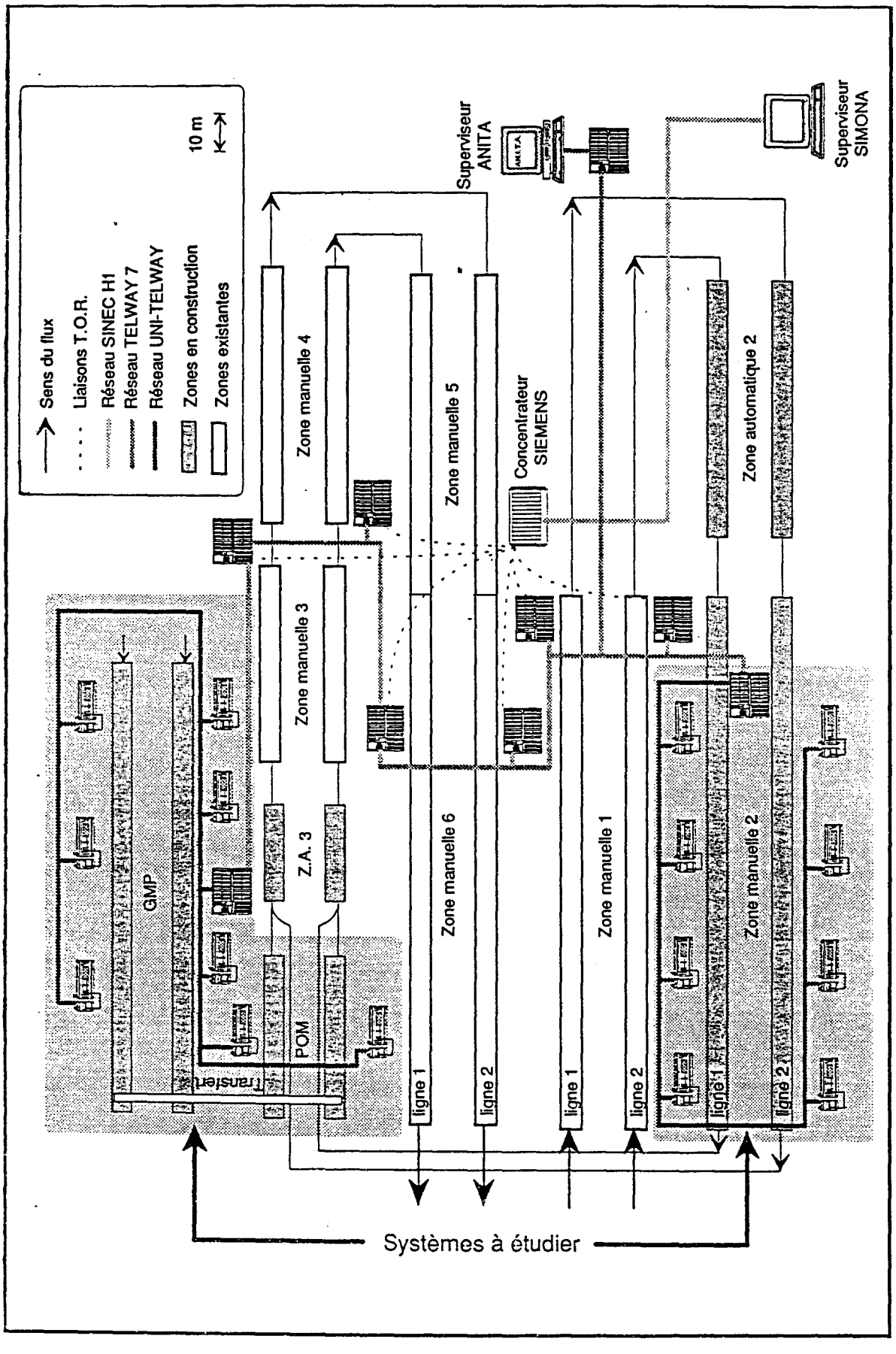


Figure III.7 : *Implantation des réseaux dans l'atelier RM1/RM2*

I.3.3.d Mise en place d'un micro-ordinateur de type PC

Après analyse des fonctionnalités demandées par les utilisateurs (Cf. I.3.2.b), il est apparu nécessaire de mettre en place un micro-ordinateur de type PC pour assurer la mémorisation et la mise en forme des divers résultats de synthèse ANDON. Celui-ci, installé dans les locaux des services de fabrication, est connecté à l'automate concentrateur ANDON via une liaison Uni-Telway.

Toutes les informations ANDON, communiquées au PC, sont traitées par un logiciel développé en dBASE III+ et baptisé AN.I.T.A. (ANdon, Information, Traitement, Analyse). Par un dialogue opérateur simple, il permet d'obtenir rapidement les différents tableaux de résultats attendus en fonction du choix de l'opérateur

I.4. Echanges fonctionnels sur les nouvelles zones

I.4.1. Système étudié

Le système étudié, analogue pour les zones ZM2 et GMP/POM est représenté sur la figure III.8. Il se compose d'un automate Télémécanique TSX 87 équipé d'un coupleur de communication SCM 21.6 et de huit mini-automates TSX17-20 configurés avec des coupleurs de communication ACC5.

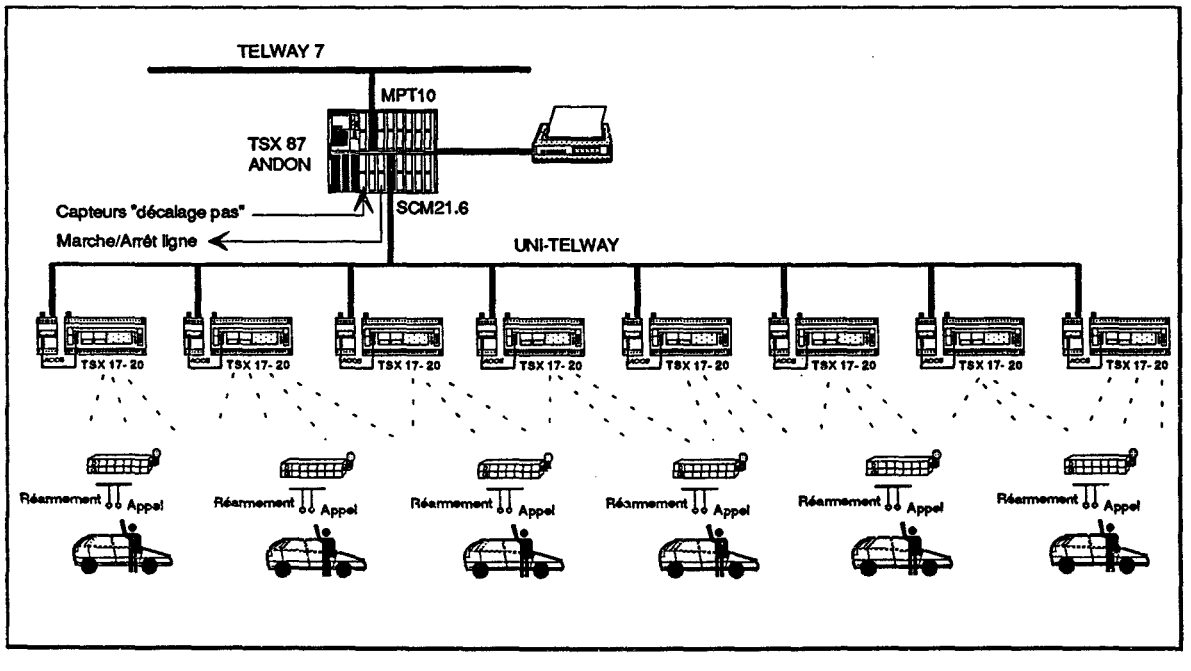


Figure III.8 : *Système installé pour les nouvelles zones manuelles*

I.4.2. Fonctions affectées aux automates

I.4.2.a Automates TSX 17-20

Compte-tenu de l'étendue des lignes et des caractéristiques techniques du matériel employé (limitation du nombre d'entrées/sorties notamment), un automate TSX 17-20 gère au maximum 14 pas. Connecté directement aux actionneurs, il assure le traitement des appels et des réarmements et visualise des différentes phases ANDON sur les tableaux synoptiques des pas (Cf. figure III.3).

Chaque événement sur les pas (appel, arrêt ou réarmement) est communiqué à l'automate principale ANDON de la zone via le bus Uni-Telway. Comme le nombre de pas visualisés sur ces tableaux (10 au maximum) est inférieur au nombre de pas connus par un TSX 17-20, il existe une dissymétrie pour répartir les pas sur les synoptiques et sur les mini-automates.

Les deux figures ci-dessous montrent que selon la configuration, à une même zone technique, peuvent être affectés un, trois ou quatre TSX 17-20.

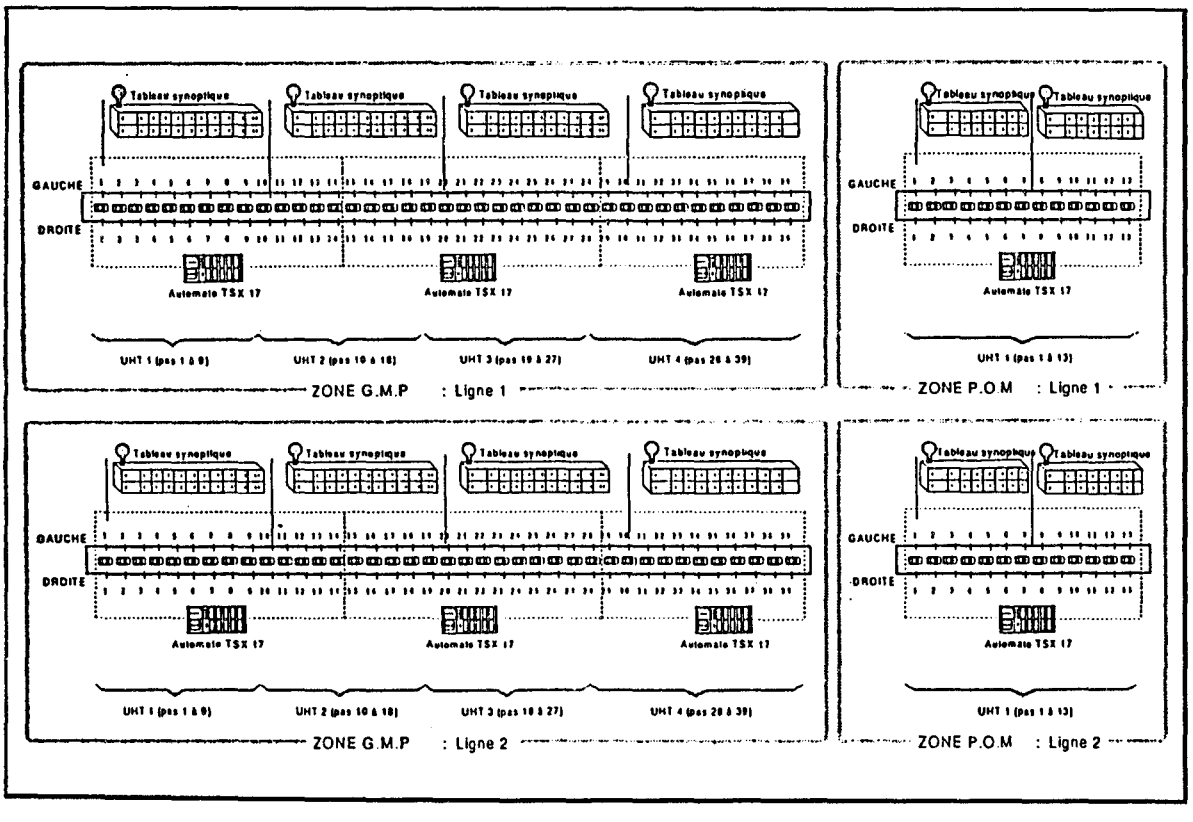


Figure III.9 : *Configuration des nouvelles zones manuelles GMP et POM*

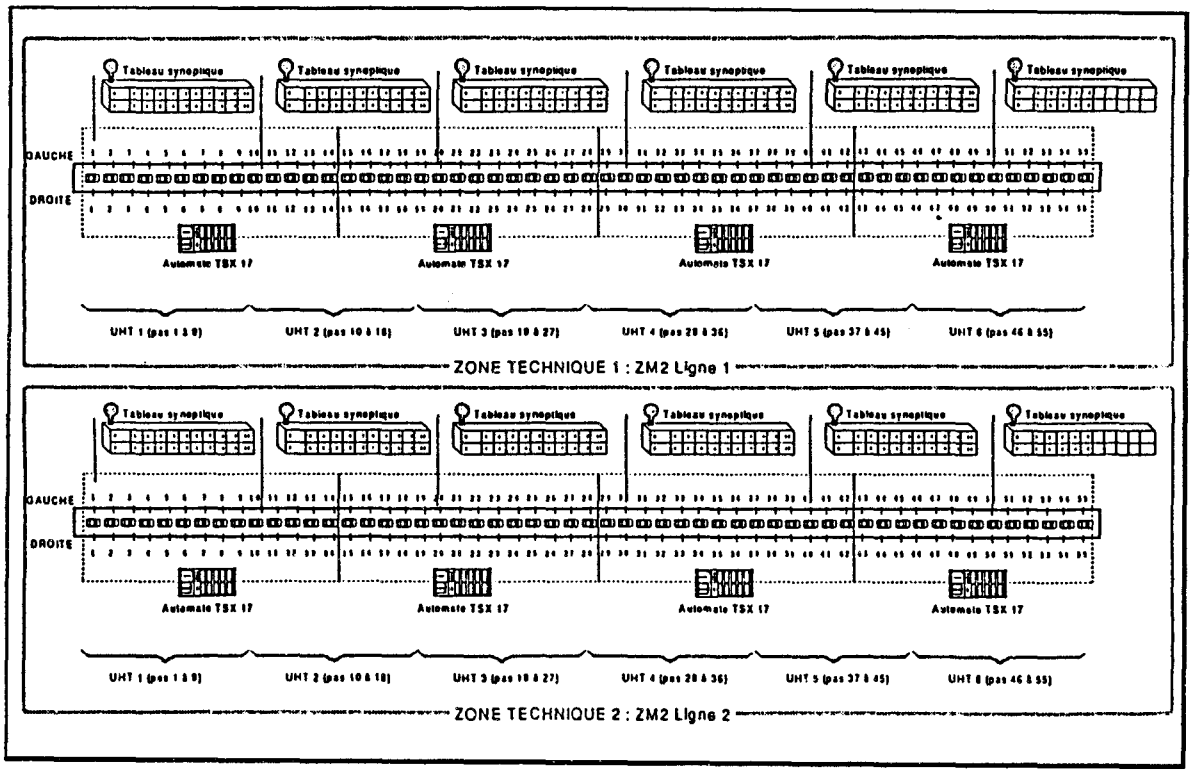


Figure III.10 : *Configuration des nouvelles zones manuelles ZM2*

I.4.2.b Automate ANDON de la zone

Cet automate, contrôle le fonctionnement de la ligne. En liaison directe avec le système de manutention, il peut, suite à l'aboutissement d'un appel ou d'un arrêt ANDON, sur une zone technique stopper la production de cette zone.

Pour les nouvelles zones techniques, les dispositifs d'appel et de réarmement ne sont plus, comme pour les zones rénovées, connectés directement aux automates ANDON. En conséquence, ceux-ci deviennent des collecteurs d'informations en provenance des pas de travail affectés aux mini-automates TSX 17-20 qui leurs sont connectés au travers du réseau Uni-Telway. Ayant une connaissance globale de la configuration des pas qui le concernent, il assure le calcul des temps d'arrêts, et les cumuls des nombres d'appel et d'arrêt pour chaque pas.

En fonction de l'état du capteur de "décalage de pas", qu'il est le seul à connaître, l'automate ANDON a un rôle de maître et synchronise le traitement de chaque TSX 17. L'information "décalage de pas" est envoyée à destination du TSX 17-20 de début de zone (automate qui gère le premier tableau synoptique d'une zone). Ensuite cette information est propagée de proche en proche entre les mini-automates pour assurer la continuité des appels en cours.

La gestion du gyrophare présent sur les tableaux synoptiques ne peut être faite par les TSX 17-20 car, cette fonctionnalité implique une connaissance globale de la ligne et un traitement particulier que seul l'automate ANDON peut effectuer.

Toutes les données qu'il possède permettent d'éditer les synthèses bi-journalières et l'impression des arrêts au fil de l'eau. En "temps réel", l'automate dialogue avec le concentrateur ANDON sur le réseau Telway 7 pour lui transmettre l'ensemble des données nécessaires à l'élaboration des tableaux de synthèses pour l'application AN.I.T.A.

Les échanges sur le réseau Uni-Telway peuvent se résumer par le schéma suivant :

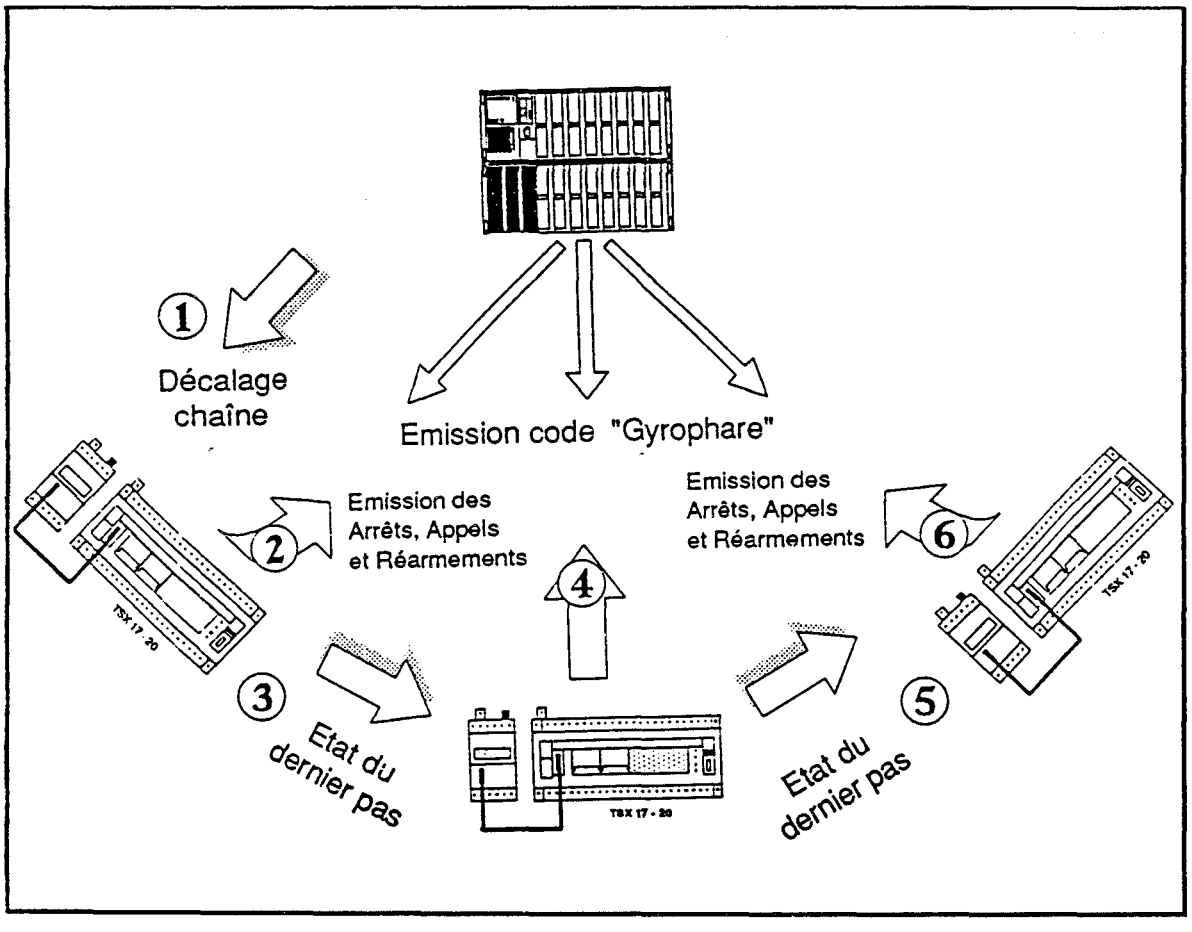


Figure III.11 : *Cycle des échanges entre les automates TSX 17 et l'automate ANDON*

Nous venons de définir les besoins en communication entre les automates des nouvelles zones. Nous allons, dans le paragraphe suivant, définir précisément la manière dont ces échanges sont implémentés dans les automates qui dialoguent sur le réseau Uni-Telway.

I.4.3. Description des échanges

Notre objectif étant d'évaluer les performances du système, il est nécessaire de connaître, pour chaque station, les données envoyées et reçues, ainsi que les séquencements de ces échanges.

I.4.3.a Contraintes matérielles

Puisque les coupleurs ACC5 ne peuvent être qu'esclave, c'est le coupleur SCM 21.6 qui est configuré en maître sur le réseau Uni-Telway. L'organisation fonctionnelle impose des échanges du TSX 87 vers les TSX 17 (information de décalage de pas, gestion du gyrophare), des échanges dans le sens inverse (apparition appel, arrêt et réarmement), ainsi que des échanges entre TSX 17 (continuité des synoptiques).

Chaque coupleur Uni-Telway ACC5 des automates TSX 17 doit donc être configuré en esclave avec une adresse application SERVEUR et application écoute (Ad0) et une adresse application CLIENT (Ad1). Comme celles-ci doivent être consécutives, le $i^{ème}$ automate TSX 17-20 est affecté des adresses $Ad0 = 2 \times i - 1$ et $Ad1 = 2 \times i$.

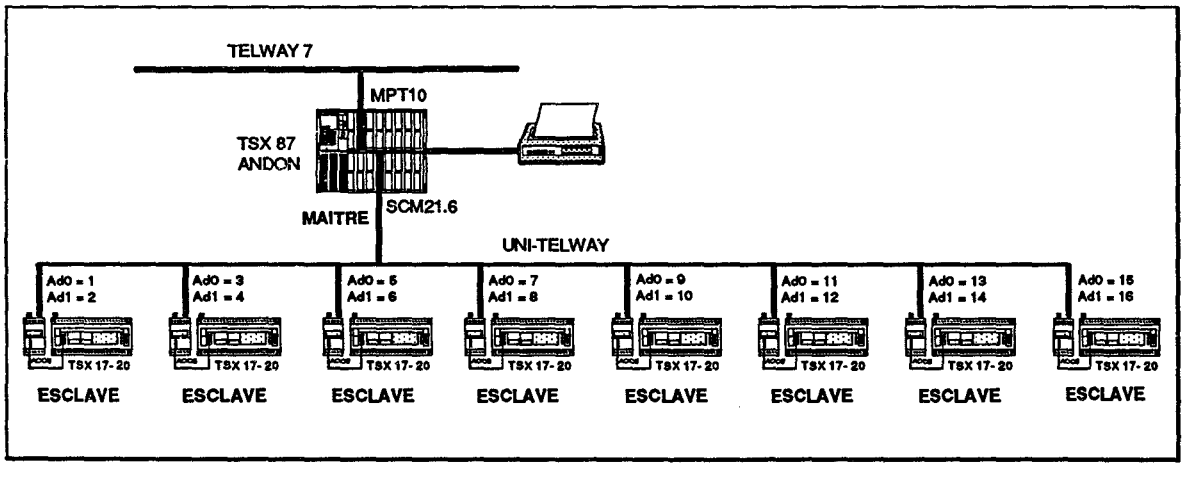


Figure III.12 : *Adressage sur le réseau Uni-Telway*

Pour les différents échanges de messages, le principe adopté est identique. L'automate émetteur de l'information écrit un mot de 16 bits dans une zone mémoire réservée de la station destinatrice. Ce mot véhicule les données relatives à l'échanges (le codage exact est détaillé dans les paragraphes suivants). L'implémentation dans les automates est réalisée au moyen de la requête standard Télémécanique UNI-TE "écriture d'un mot" (code $14). La programmation en langage PL7-2 pour les automates TSX 17-20 et PL7-3 pour les TSX 87 utilise les blocs Texte pour l'émission de cette requête.

Pour la gestion des numéros de pas, chaque automate TSX 17-20 connaît par configuration, les numéros "relatifs" des pas qui lui sont affectés. Ceux-ci sont convertis, à un niveau supérieur, en numéro de pas "absolu", pour permettre une gestion globale au niveau de l'atelier (1200 pas prévus).

I.4.3.b Echanges des TSX 17 vers le TSX 87

Chaque automate TSX 17 scrute les actionneurs d'appel de ses pas et gère l'état des lampes des tableaux synoptiques associés. Lorsqu'un appel est déclenché, il affecte un niveau d'alerte 0 au pas en défaut et assure le clignotement de la lampe correspondante sur le tableau synoptique. Il signale alors l'apparition de l'appel à l'automate ANDON par l'émission d'une requête d'écriture de mot. Les informations transmises sont le code message (1 pour signaler un appel) et le numéro relatif du pas associé à l'appel.

Sur réception de l'information d'avance de ligne, l'automate TSX 17 décale les défauts en cours sur les pas et incrémente les niveaux d'alerte associés. Si le niveau d'alerte maximum est atteint pour l'un des pas, il envoie à l'automate ANDON une requête d'écriture de mot pour signaler l'arrêt obligatoire de la ligne. Les informations transmises sont le code message (2 pour signaler un arrêt) et le numéro du pas incriminé.

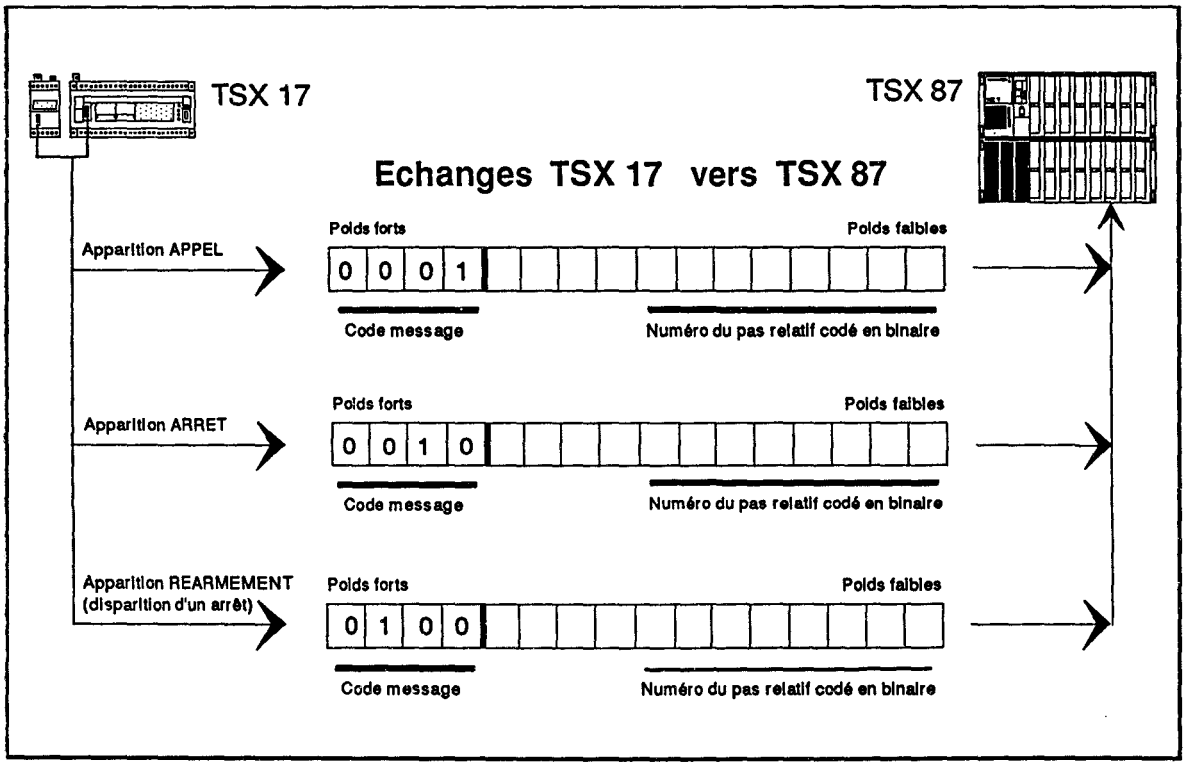


Figure III.13 : *Echanges TSX 17 vers TSX 87*

Suite à un réarmement opérateur, l'automate TSX 17 met à jour le voyant associé sur le synoptique. Si le défaut avait, préalablement, provoqué un arrêt de la ligne le mini-automate envoie un message au concentrateur ANDON. Les informations transmises sont le code message (4 pour signaler un réarmement) et le numéro du pas associé.

I.4.3.c Echanges du TSX 87 vers les TSX 17

Pour chaque Z.T., l'automate ANDON reçoit l'information de "décalage pas" signifiant le franchissement d'un pas sur la ligne. Cette information est transmise au premier automate TSX 17 de la zone considérée (TSX 17 de début de zone ou autonome). Cet échange est réalisé par l'écriture de mot avec un code message 4.

A la réception d'un message d'arrêt ou de réarmement de la part d'un TSX 17, l'automate ANDON rafraîchit en mémoire l'état des gyrophares et envoie une requête d'écriture mot vers le ou les automates TSX 17 chargés de la commande des gyrophares à modifier. Les informations transmises sont le code message (15 pour signaler une commande gyrophare) et l'état logique de chaque gyrophare.

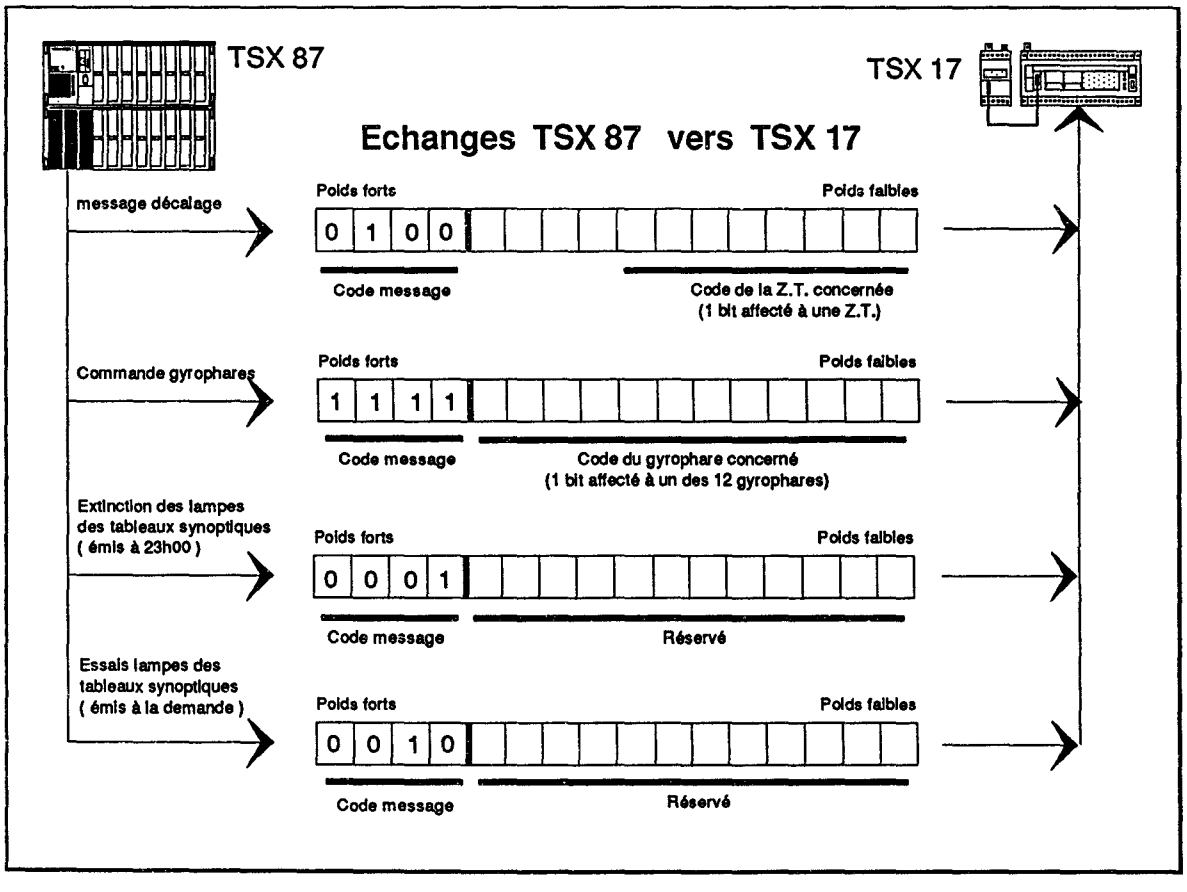


Figure III.14 : *Echanges TSX 87 vers TSX 17-20*

Deux autres messages nécessaires au système ANDON ont été développés :

- le message "extinction de toutes les lampes" émis par le TSX 87 vers chaque mini-automate à une heure programmée en fin de journée (23 H 00). A la réception de ce message, les TSX 17 stoppent toute visualisation en cours sur leurs tableaux synoptiques et gyrophares,
- le message "essai de toutes les lampes", émis à la demande du personnel de maintenance par action sur un bouton poussoir. La réception de ce message par chaque TSX 17 active une séquence de visualisation sur les tableaux synoptiques.

Ces échanges étant occasionnels, ne seront pas considérés pour la modélisation et la simulation présentées dans les paragraphes suivants.

I.4.3.d Echanges entre TSX 17

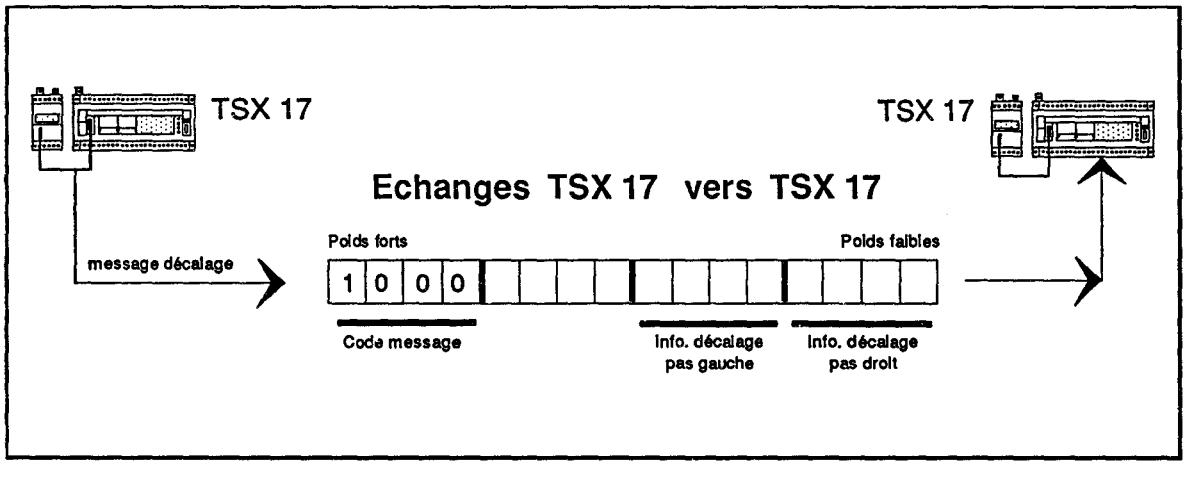


Figure III.15 : *Echanges TSX 17-20 vers TSX 17-20*

Les échanges inter-automates TSX 17 existent lorsque plusieurs équipements sont nécessaires pour couvrir une même Z.T. Ils assurent la continuité du traitement des appels en cours entre deux TSX 17. En effet, tout signal de décalage ligne émis par l'automate ANDON vers un TSX 17 de début de ligne entraîne la propagation de cette information entre chaque TSX 17 de cette ligne. Le message se traduit par l'envoi vers le TSX 17 suivant d'une requête d'écriture d'un mot. Les informations transmises sont le code message (8 pour signaler un décalage) et l'état courant des derniers pas gauche et droit.

Pour la ligne ZM2, l'enchaînement des échanges entre les équipements, suite à un décalage de la ligne est présenté sur la figure suivante :

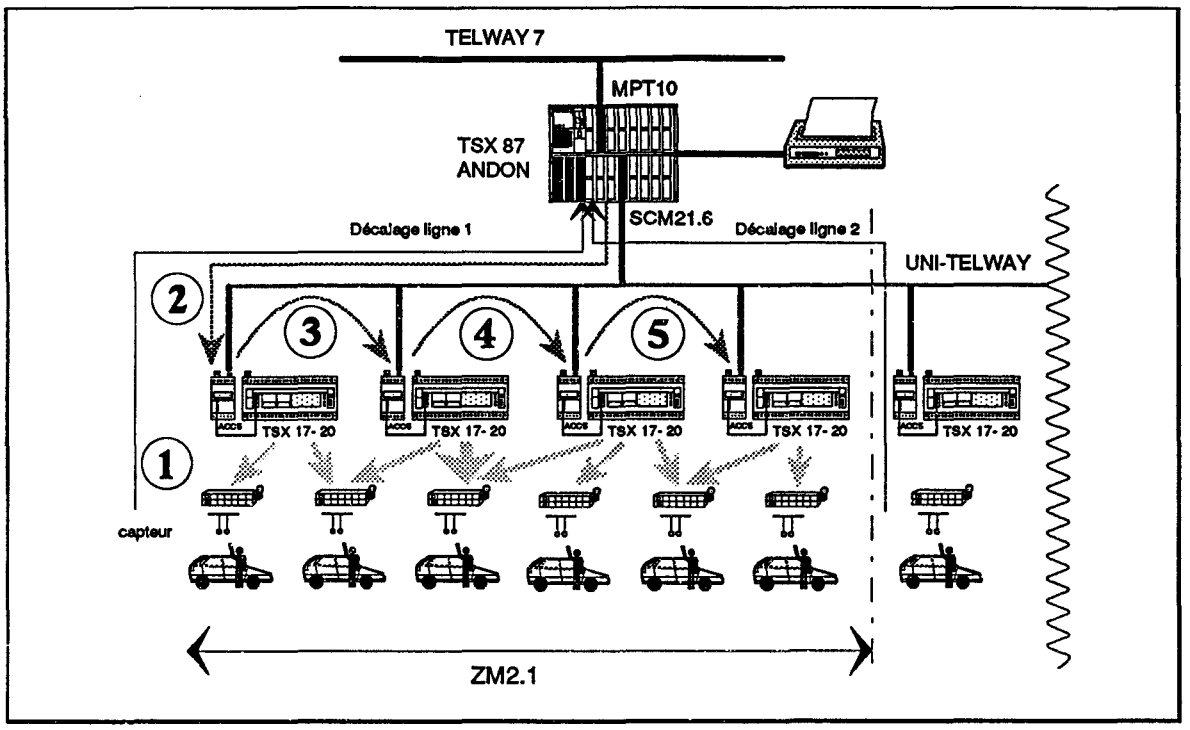


Figure III.16 : *Propagation de l'information "décalage ligne" sur ZM2.1*

Lorsque la ligne franchit un pas, un capteur transmet un information T.O.R. appelée "avance ligne" à l'automate ANDON (flèche n°1). Ce dernier envoie alors un message de décalage (type 4) à l'attention du premier TSX 17-20 de la zone considérée (flèche n°2). Le mini-automate mets donc à jour l'état des pas dans sa mémoire, ce qui provoque une réactualisation sur les voyants lumineux gérés par l'équipement.

Lorsque le TSX 17-20 n'est pas un automate de fin de ligne, il propage l'information "avance ligne" à son successeur. Dans le message associé sont indiqués les états courants des derniers pas gauche et droit gérés par le mini-automate à l'initiative du message. Ces informations permettent d'assurer la continuité au niveau de la signalisation de l'état des appels sur les tableaux synoptiques. Il y a donc un enchainement de messages successifs (flèches n°3, 4 et 5).

Chaque mini-automate, lorsqu'il reçoit l'information de "décalage ligne", teste l'état des différents pas qu'il gère pour voir si l'un d'eux n'a pas atteint de degré d'alerte maximum. Si c'est le cas, il envoie un message d'arrêt (type 2) à l'attention de l'automate ANDON (flèche n°5 sur la figure III.17). Ce dernier met à jour dans sa mémoire les états des gyrophares et émet une consigne (type 15) à l'attention du TSX 17-20 chargé de la commande du gyrophare à allumer pour signaler l'arrêt à l'opérateur (flèche n°6 sur la figure III.17).

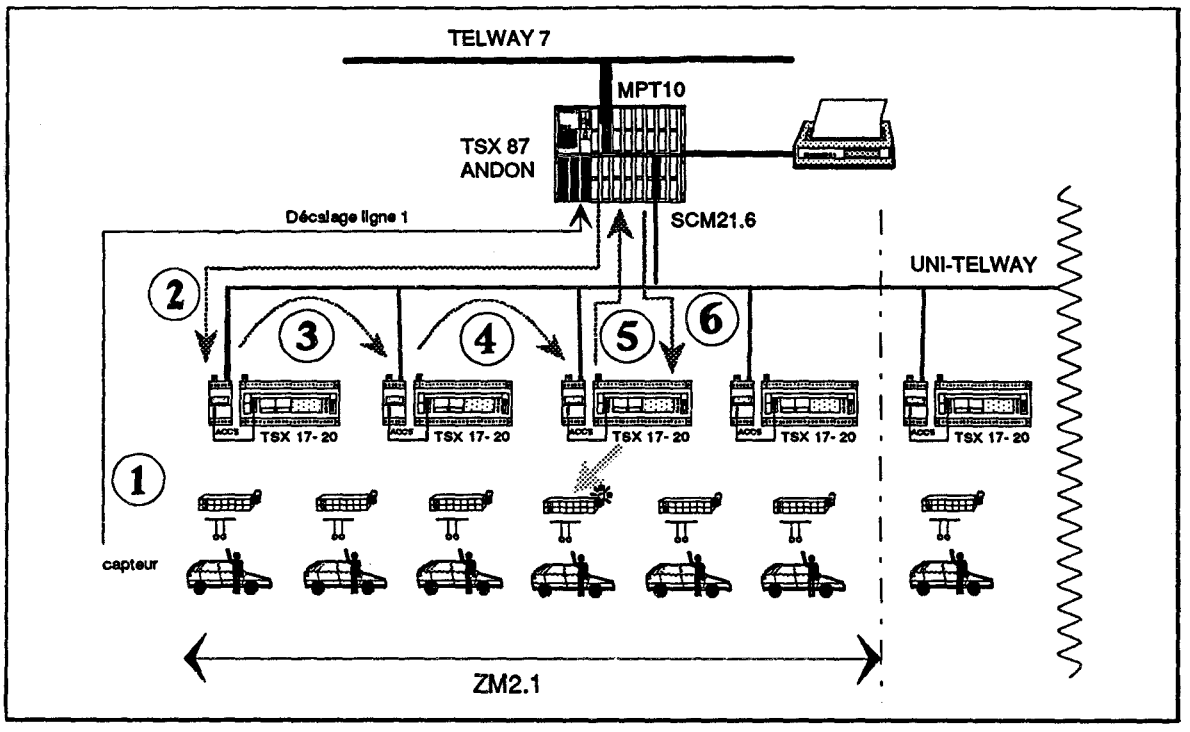


Figure III.17 : *Mise en oeuvre du gyrophare suite à un arrêt ANDON sur ZM2.1*

I.4.4. Applicatifs des équipements

I.4.4.a Automate ANDON TSX 87

Le programme de l'automate ANDON, vis à vis de ces échanges sur le réseau Uni-Telway, consiste à gérer l'état des gyrophares de chaque tableau synoptique à partir des messages reçus des TSX 17-20, et à informer les mini-automates du décalage de la ligne. Le séquencement des opérations qu'il doit réaliser est résumé ci-dessous.

APPLICATIF de l'AUTOMATE TSX 87

{ LECTURE des mots reçus de la part des TSX 17-20 }

POUR chaque automate TSX 17-20 FAIRE
  SI un mot a été reçu du TSX 17-20 considéré ALORS
    SI mot de type 1 (appel andon sur le TSX 17-20) ALORS
      Comptabiliser l'appel en mémoire automate
    FinSI

.../...

```
SI mot de type 2 (arrêt andon sur le TS 17-20) ALORS
        Mémoriser l'arrêt en mémoire automate
        Déclencher le calcul de la durée de l'arrêt
        Emission du nouvel état gyrophares aux TSX 17-20 concernés
    FinSI

    SI mot de type 4 (réarmement andon) ALORS
        Mémoriser le réarmement en mémoire automate
        Calculer la durée de l'arrêt
        Emission du nouvel état gyrophares aux TSX 17-20 concernés
    FinSI
  FinSI
FinFAIRE


{ Gestion du DECALAGE "avance ligne" }

POUR chaque ligne FAIRE
  SI avance d'un pas de la ligne considérée ALORS
    Envoi du mot de type 4 (avance ligne) au TSX 17-20 tête de ligne
  FinSI
FinFAIRE
```

I.4.4.b Automates TSX 17-20

Chaque mini-automate gère localement l'état des pas qui lui sont affectés. Pour chaque événement (appel, arrêt ou disparition d'arrêt), il envoie un message à l'automate ANDON pour l'en informer. A la réception d'un message de "Décalage Ligne", il met à jour l'état local de ses pas et propage l'information à son successeur éventuel.

APPLICATIF des AUTOMATES TSX 17-20

```
{ Prise en compte des APPELS ANDON }

POUR chaque appel FAIRE
  Mise à jour du pas associé en mémoire automate
  Mettre un mot de type 1 (appel) dans la FIFO des messages à émettre
FinFAIRE
```

.../...

```
{ Prise en compte des REARMEMENTS ANDON }

 POUR chaque réarmement FAIRE
     Mise à jour du pas associé en mémoire automate
     SI la ligne est arrêté à cause du pas courant ALORS
         Mettre un mot de type 4 (réarmement) dans la FIFO des messages à émettre
     FinSI
 FinFAIRE


{ DECALAGE de la ligne ? }

 SI "avance ligne" reçu du 87 ou du TSX 17-20 précédent ALORS
     Mise à jour de l'état des pas en mémoire automate
     SI un pas a atteint le niveau d'alerte maximum ALORS
         Mettre un mot de type 2 (arrêt) dans la FIFO des messages à émettre
     FinSI
     SI je ne suis pas un 17-20 de fin de ligne ALORS
         Mémoriser un mot de type 8 (avance) comme message à émettre
     FinSI
 FinSI


{ Message GYROPHARE ? }

 SI "commande gyrophare" reçue du TSX 87 ALORS
     Modifier la sortie du gyrophare
 FinSI


{ EMISSION des MESSAGES }

 SI aucun échange en cours ALORS
     SI un mot dans la FIFO des messages à émettre ALORS
         Emettre le message avec un bloc TXT
         Déclencher un timer
     SINON
         SI un message de type 8 à envoyer ALORS
             Emettre le message de type 8 avec un bloc TXT
             Déclencher un timer
         FinSI
 SINON
     SI le timer a dépassé le time-out ALORS
         Réémettre le dernier message
     FinSI
```

Ces instructions sont exécutées à chaque cycle de la tâche de l'automate TSX 17-20. Il est absolument nécessaire de gérer une FIFO des messages à émettre, car "*l'UC esclave ne peut gérer qu'un échange à la fois avec le programme application. Pour entamer un autre échange, il faut attendre impérativement que le premier soit terminé (réception d'un*

compte-rendu faisant remonter le bit D du bloc texte)" (Cf. documentation Interface TSX 17 ACC5 Bus UNI-TELWAY Micro-automate TSX 17-20 page 32).

Lorsqu'un bloc Texte émet une requête, il se positionne en attente d'une réponse. En cas de mauvaise communication, ou de déconnexion du destinataire pendant l'échange, ce bloc Texte reste bloqué. Le programme application tient compte de ce phénomène en incluant un time-out lors de l'envoi d'une requête. Si aucune réponse n'a été reçue à l'expiration du time-out, le bloc Texte est réinitialisé. Dans ce cas, le programme essai de relancer la requête. Dès la disparition du défaut, l'émission des requêtes reprend normalement. Ainsi aucune information n'est perdue.

II. MODELISATION DU SYSTEME D'UNE ZONE

Suite à une présentation des travaux de modélisation/simulation d'architecture de commande d'atelier appliqués sur un autre exemple industriel du groupe PSA Peugeot Citroën (Cf. annexe 2), les concepteurs de l'architecture qui vient d'être détaillée ont demandé d'évaluer les performances de leur système. Ils ne disposent en effet d'aucun moyen d'estimation a priori lors de la phase de conception. Nous avons donc appliqué ensemble la démarche présentée dans le premier chapitre de ce mémoire aux réseaux Uni-Telway des nouvelles zones, dont les performances étaient difficiles à appréhender en l'absence de toute référence, eu égard à la nouveauté de l'application.

II.1. Modélisation de l'architecture matérielle

Le système étudié a été présenté sur la figure III.8. Il se compose d'un automate Télémécanique TSX 87-30 avec une carte de communication Uni-Telway configurée en maître et de huit automates TSX 17-20 équipés de coupleurs Uni-Telway ACC5 esclaves. Ces équipements sont reliés par un bus Uni-Telway.

Compte-tenu de la nature de l'application à réaliser, l'utilisation unique d'une tâche maître dans chaque automate a été jugée suffisante. Le modèle de cette architecture s'obtient alors par assemblage de graphes réseaux de Petri qui ont été détaillés dans le chapitre II de ce mémoire (Cf. figure III.18).

Le système comporte un seul automate TSX 87. Dans le graphe réseau de Petri associé à cet équipement circule donc un unique jeton qui matérialise l'état de la tâche principale. De même, l'état du coupleur SCM21.6 configuré en maître Uni-Telway est représenté par le marquage et les valeurs des paramètres du seul jeton présent dans le graphe du modèle de ce matériel.

Huit mini-automates TSX 17-20 sont connectés au réseau Uni-Telway. Pour modéliser ces matériels identiques on utilise un seul graphe de processus, qui décrit les comportements de la tâche principale, dans lequel circulent, de manière totalement indépendante, huit jetons d'état. Chaque jeton est associé à un des huit équipements. Pour les distinguer, la valeur du premier paramètre de ces jetons indique le numéro de l'automate programmable associé (nombres impairs de 1 à 15).

Chaque coupleur ACC5 gère deux adresses liaison (Ad0 et Ad1). Dans le graphe de processus associé au comportement de l'esclave Uni-Telway circulent donc 16 jetons, dont la valeur du premier paramètre correspond au numéro de l'esclave (de 1 à 16). Les échanges de données entre le graphe de processus de la tâche maître du TSX 17-20 et le graphe de processus du coupleur ACC5 sont réalisés en associant les jetons de la tâche maître dont le premier paramètre vaut i avec les jetons du coupleur esclave dont le premier paramètre est égal à i ou i+1.

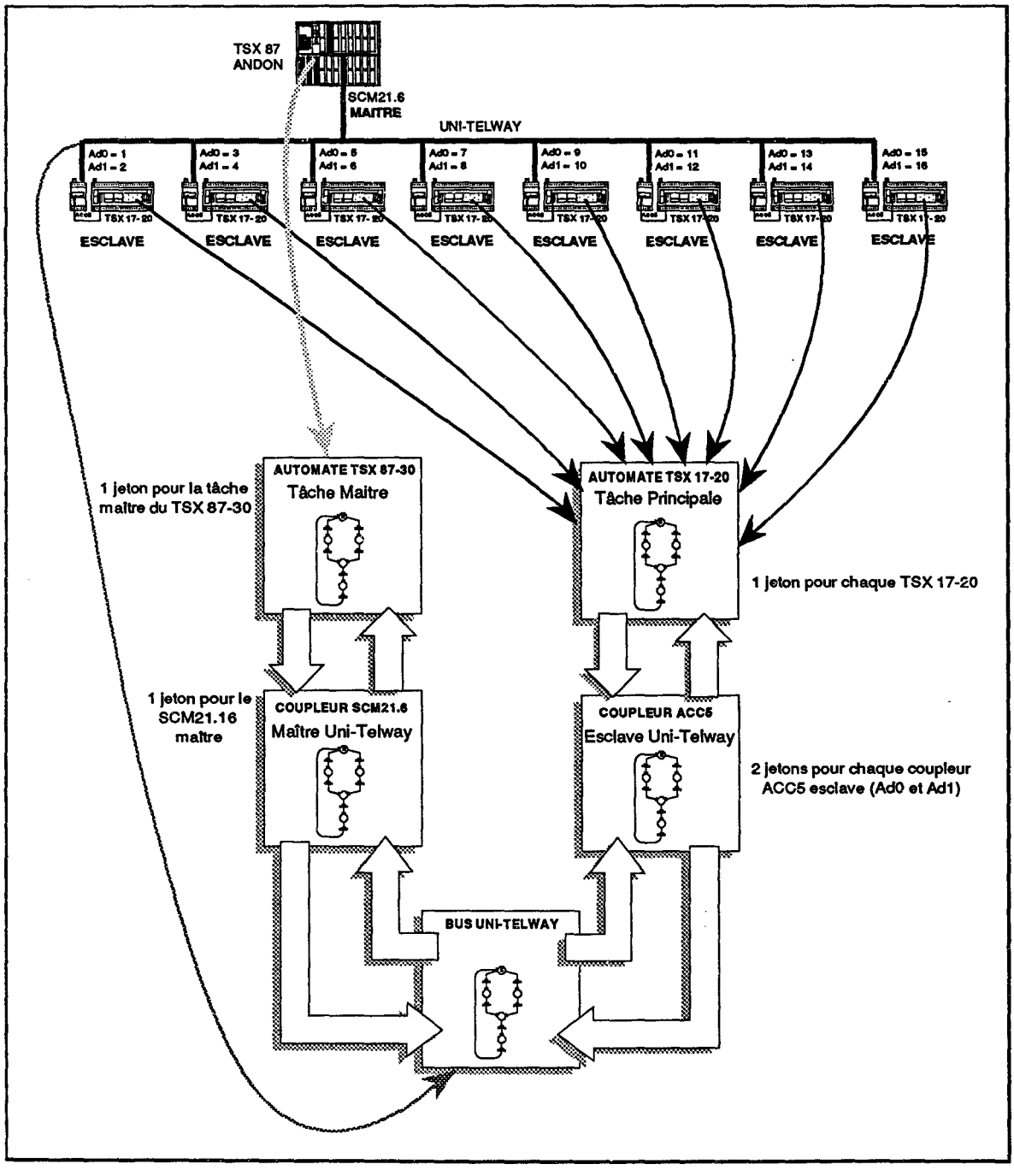


Figure III.18 : *Construction du modèle du système à étudier*

II.2. Description des applicatifs

II.2.1. Principes

Il s'agit de décrire, pour chaque équipement, le séquencement de ses communications sur le réseau Uni-Telway. Ainsi, lors de simulations, il est possible d'observer les échanges sur ce réseau, d'en évaluer les performances et de détecter d'éventuelles erreurs de conception dans l'organisation des dialogues (mauvaise synchronisation par exemple).

Comme expliqué dans le chapitre II de ce mémoire, ces applicatifs sont décrits, par des procédures écrites en langage Pascal. L'interprétation du formalisme lors de la simulation des modèles réseaux de Petri permet de faire de lien entre ces procédures et le modèle du fonctionnement interne de l'équipement.

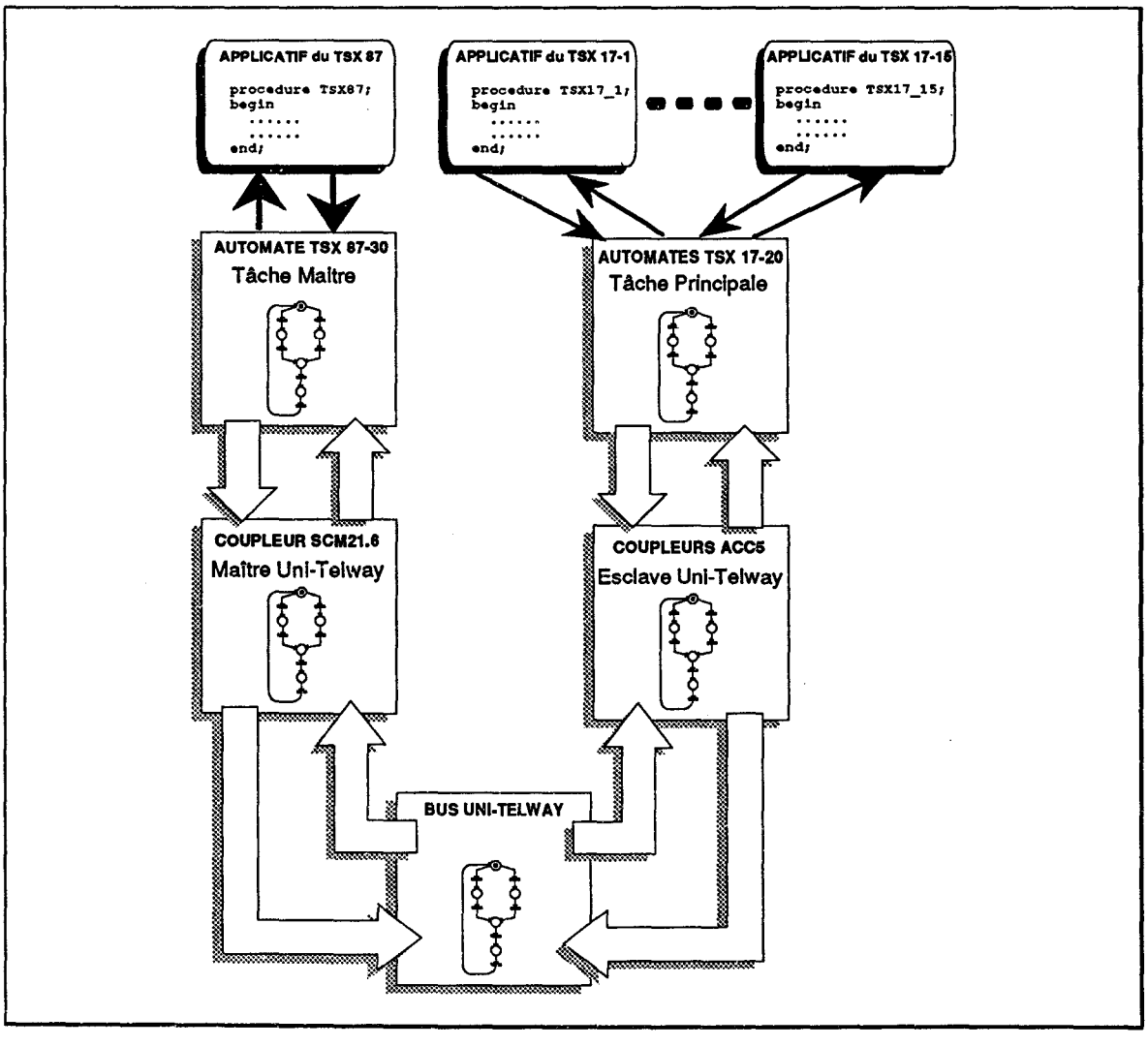


Figure III.19 : *Ajout de la description des applicatifs sur le modèle des équipements*

Une structure de données "Bloc TXT", définie avec les modèles des automates programmables Télémécanique, permet de décrire l'utilisation de requêtes UNI-TE. Ainsi, les échanges de données sont spécifiés d'une manière analogue à celle utilisée dans les programmes automates écrits en langages PL7-2 ou PL7-3.

De même, des tableaux de booléens "I" et "O" sont accessibles, pour chaque automate, à la procédure Pascal de description de son applicatif. Ils représentent les images des entrées/sorties T.O.R. dont dispose la tâche maître de l'automate. "I" est utilisé en lecture, pour recevoir des informations de l'environnement et "O" en écriture pour envoyer des consignes vers l'extérieur.

Pour chaque station, le déroulement de son applicatif est conditionné par les messages reçus des autres stations et par les informations T.O.R. en provenance du procédé. Il réagit en envoyant des messages à destination d'autres équipements et en modifiant ses sorties T.O.R. vers le procédé.

II.2.2. Modèle de l'applicatif du TSX 87 ANDON

II.2.2.a Description des Entrées/Sorties T.O.R.

L'automate ANDON reçoit de chaque zone ZM2.1 et ZM2.2 une information de décalage de ligne. En fonction des messages en provenance des TSX 17-20, il envoie à destination de chaque ligne des consignes de marche ou d'arrêt du flux de production.

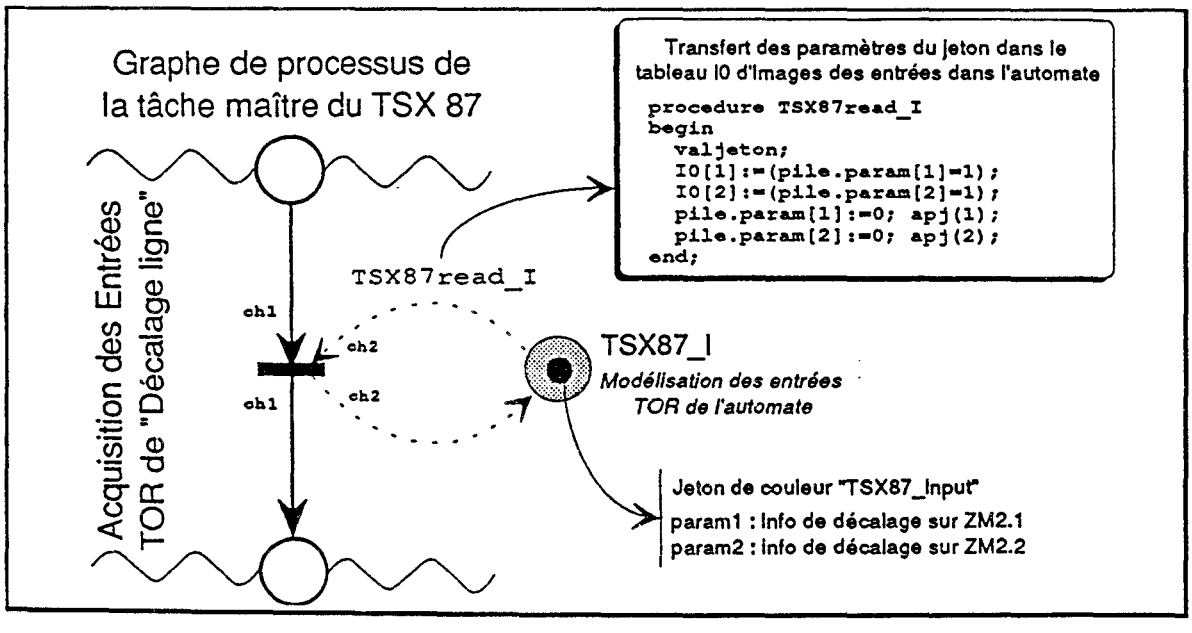


Figure III.20 : *Modélisation des entrées TOR de l'automate TSX 87*

Comme expliqué dans le chapitre II, les entrées/sorties T.O.R. sont représentées dans le modèle par les paramètres des jetons présents dans les places de communication TSX87_I et TSX87_O du graphe de la tâche maître.

En début de chaque cycle, la tâche maître du TSX 87 lit les paramètres du jeton présent dans la place TSX87_I et les transfère dans une table interne, dénommée `IO`, d'état des entrées T.O.R. La traduction en réseaux de Petri est présentée sur la figure III.20.

En fin de chaque cycle, la tâche maître met à jour les sorties TOR à destination du process. Comme détaillé sur la figure ci-dessous, cette opération est modélisée par la création d'un jeton dans la place TSX87_O dont les paramètres prennent les valeurs de la table interne `OO` des sorties TOR du modèle de la tâche maître.

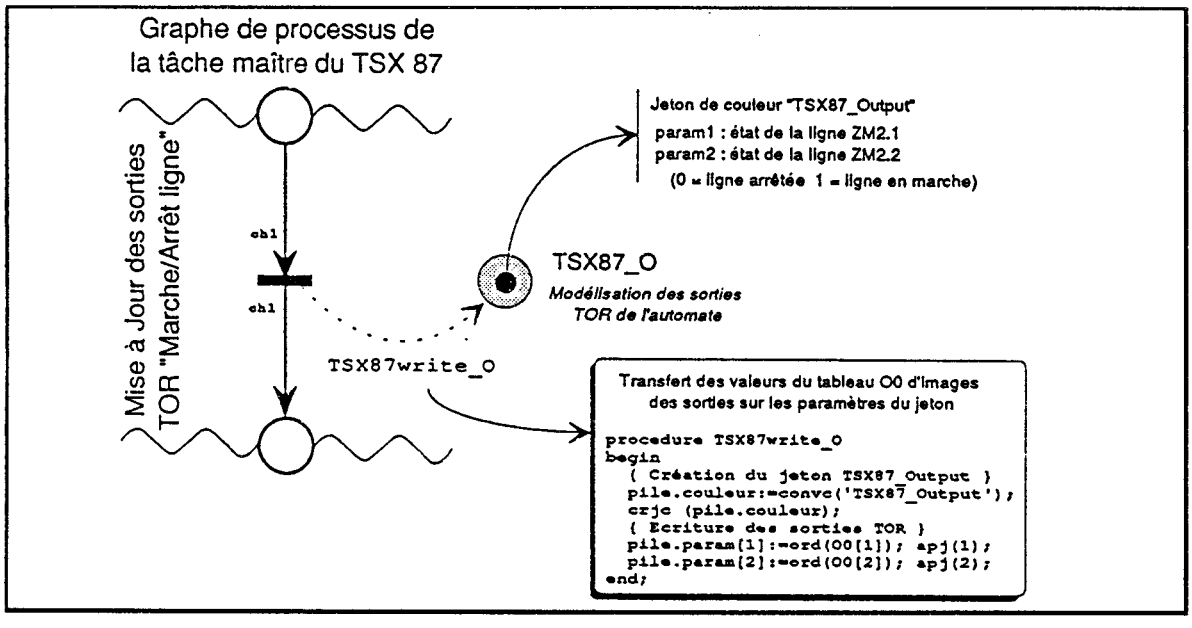


Figure III.21 : *Modélisation des sorties TOR de l'automate TSX 87*

II.2.2.b Description de l'applicatif

Ce modèle est directement déduit de la description de l'applicatif présentée au paragraphe I.4.4.a. La procédure Pascal du modèle de l'automate ANDON de la zone ZM2 est reproduite en page suivante.

Cette routine est appelée à chaque cycle de la tâche maître. Pour stocker l'image, locale à l'automate modélisé, de l'état du procédé on utilise des variables (`Nb_Appels`, `Nb_Arrets`, `ZM21_Arretee` etc.), qui doivent être globales au graphe de processus de la tâche maître, pour conserver leurs valeurs entre deux appels de la procédure.

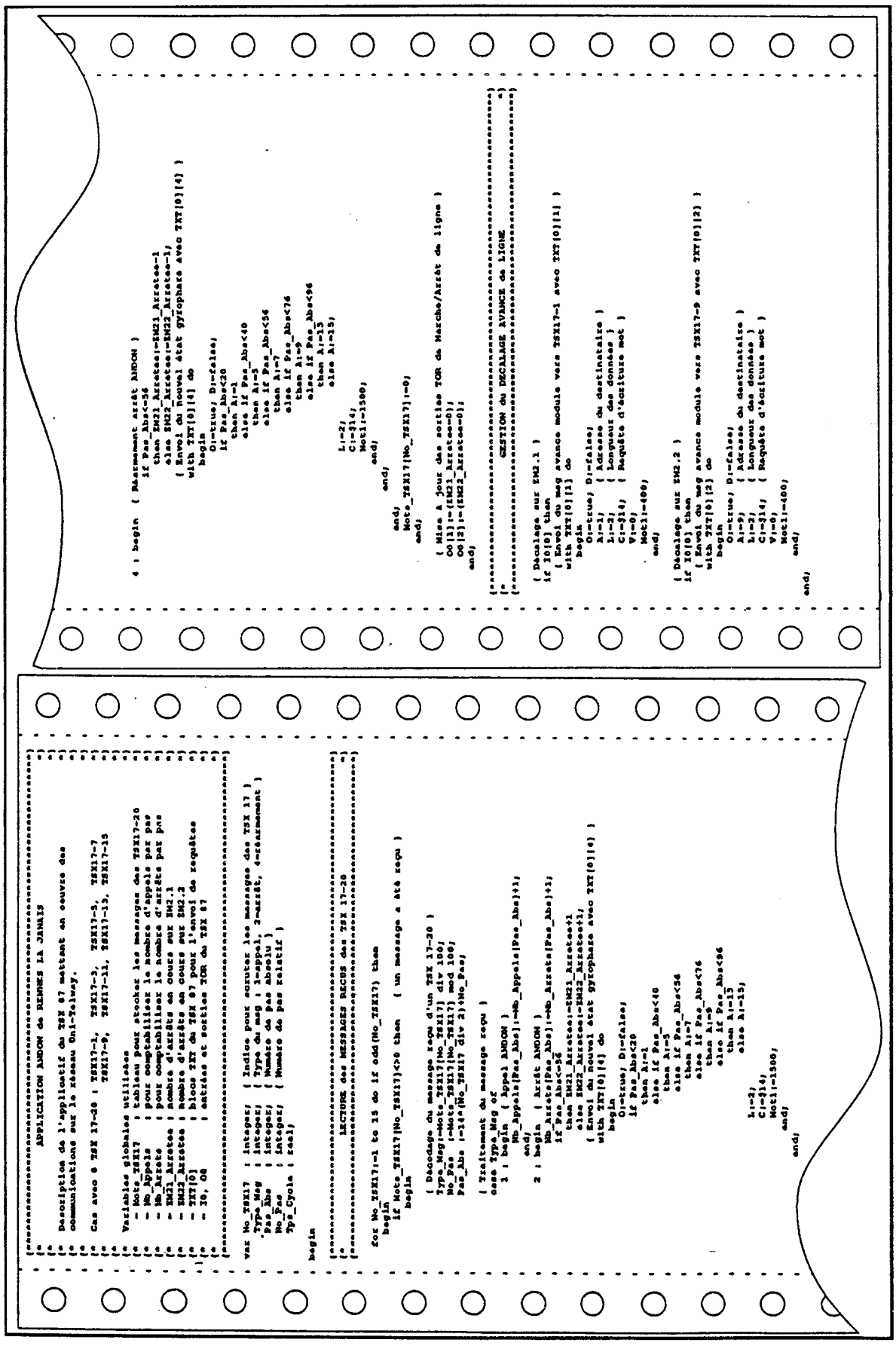


Figure III.22 : *Description de l'applicatif de la tâche maître du TSX 87 ANDON de ZM2*

II.2.3. Modèle de l'applicatif des TSX 17-20

II.2.3.a Description des Entrées/Sorties T.O.R.

Les TSX 17-20 reçoivent de chaque pas qui leur est affecté, des informations d'appel et de réarmement ANDON. Le principe de représentation de ces entrées TOR est le même que pour le TSX 87. Les micro-automates n'envoient aucune consigne vers le procédé.

II.2.3.b Modèle des applicatifs des TSX 17-20

Ce modèle est directement déduit de la description de l'applicatif présentée au paragraphe I.4.4.b, et sa description fait appel aux mêmes concepts que celle de l'applicatif du TSX 87 ANDON.

II.3. Modélisation de l'environnement

II.3.1. Principes

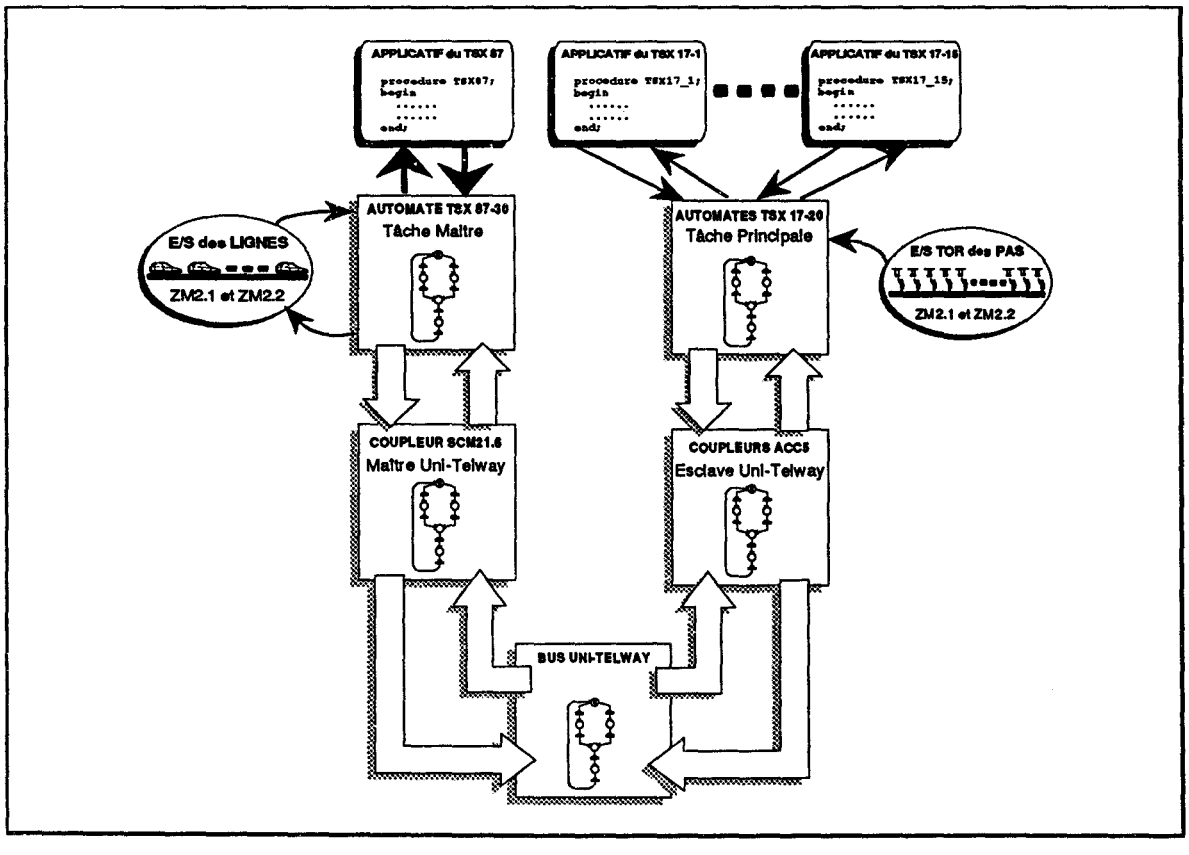


Figure III.23 : *Ajout de la description du comportement du procédé*

Il s'agit de décrire le comportement du procédé, vis à vis de ses interaction avec les entrées/sorties des automates. C'est en effet l'évolution de l'état du process qui conditionne les actions du système de suivi de fabrication. Pour les nouvelles zones techniques, l'environnement est constitué des commandes Marche/Arrêt des lignes, des l'informations de décalage ligne et des actions d'appel ou de réarmement des opérateurs.

II.3.2. Interaction entre le TSX 87 et les lignes de montage

II.3.2.a Approche simpliste

Une manière simple de procéder est de positionner, de manière régulière, les paramètres de décalage ligne à 1 sur le jeton de la place TSX87_I. Cette approche correspond au fonctionnement normal des lignes sans jamais aucun arrêt.

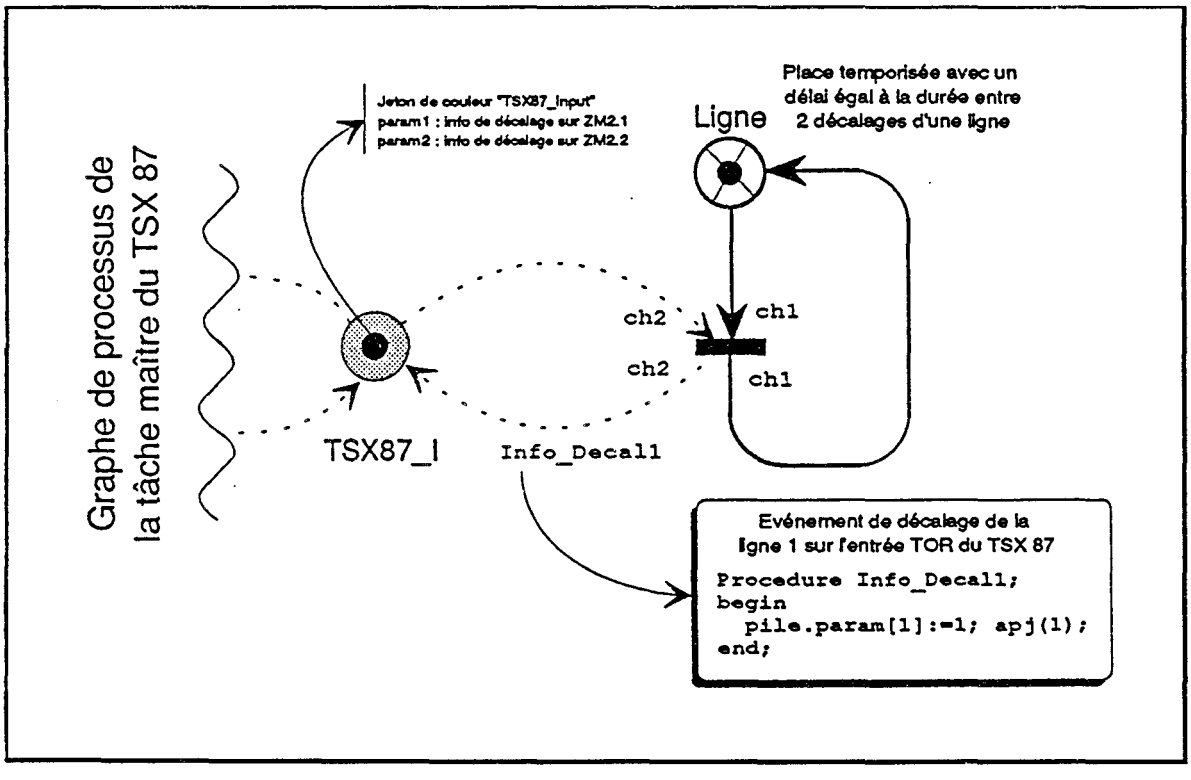


Figure III.24 : *Représentation simpliste du "Décalage Ligne"*

Pour tenir compte, de manière probabiliste, des arrêts il est possible d'introduire un caractère stochastique dans la temporisation de la place Ligne : une première loi détermine la fréquence d'apparition d'un arrêt et une seconde loi la durée de l'arrêt. Lors de l'apparition d'un arrêt, la temporisation de la place Ligne est égale à la durée normale nécessaire à la chaîne pour changer de pas augmentée du temps estimé de l'arrêt.

Cette approche ne convient cependant pas pour le cas que nous avons à traiter. En effet, il est nécessaire que les événements du procédé soient corrélés avec ceux du modèle de la commande. L'apparition d'un arrêt doit, par exemple, correspondre à un appel qui atteint le degré d'urgence maximum. C'est pourquoi un modèle plus détaillé du process a été élaboré.

II.3.2.b Approche détaillée

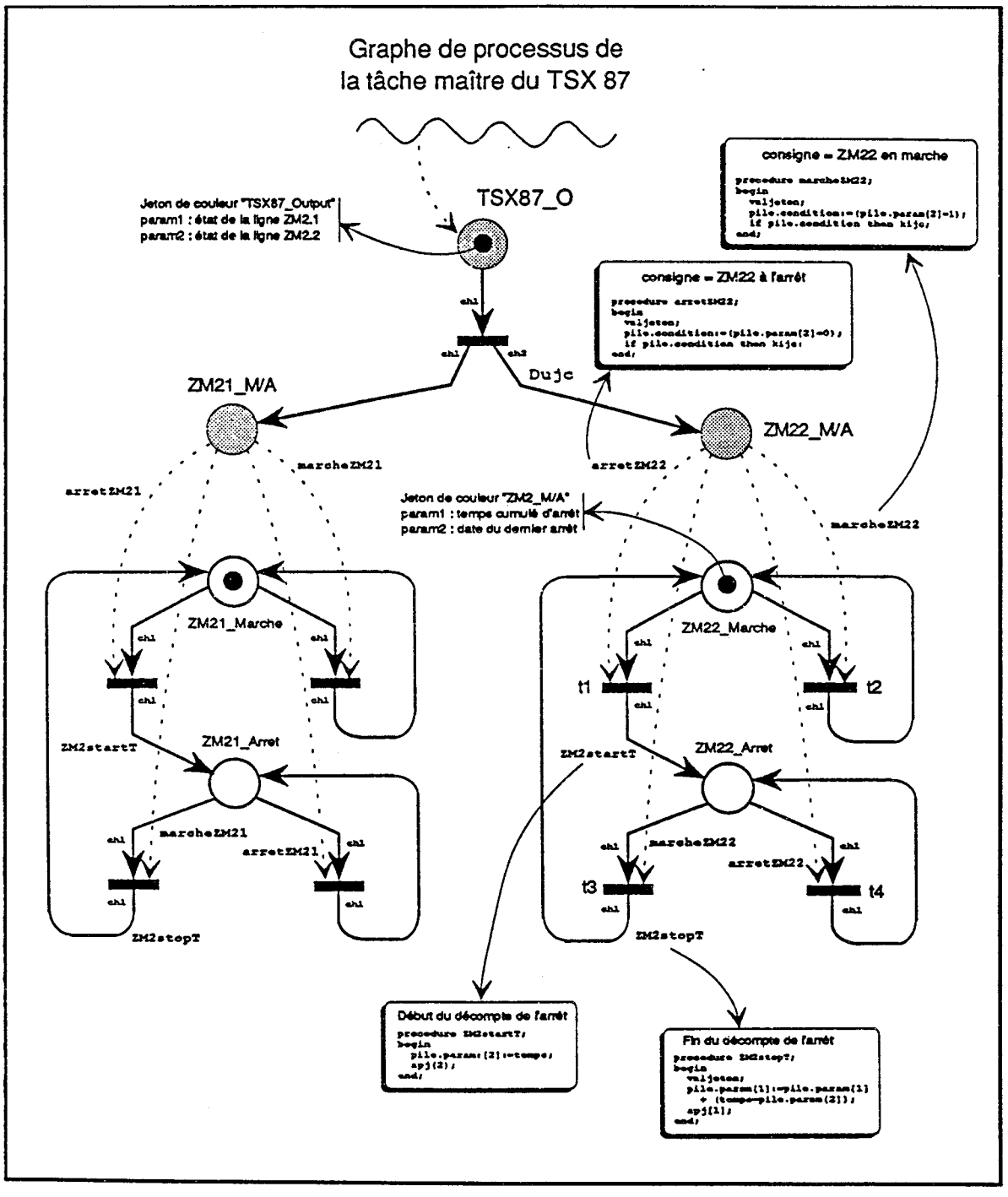


Figure III.25 : *Modèle du procédé de l'état des lignes ZM2.1 et ZM2.2*

Chaque zone (ZM2.1 et ZM2.2) est décrite par un graphe réseaux de Petri spécifique. Un premier graphe permet de matérialiser l'état de la ligne : marche ou arrêt. Un jeton dans la place ZM22_Marche (resp. ZM22_Arret) traduit que la ligne ZM22 est en marche (resp. en arrêt).

Les changements d'états sont provoqués par les consignes de fonctionnement de la ligne du TSX 87, qui sont représentées par les paramètres 1 et 2 du jeton de couleur "TSX87_Output". Celui-ci est dupliqué dans les deux places ZM21_M/A et ZM22_M/A spécifiques à chacune des zones.

L'interprétation sur les valeurs des paramètres de ces jetons permet de conditionner la transition franchissable. Par exemple si ZM22_Marche est marquée et que param2 du jeton présent dans ZM22_M/A est positionné à 0 (ligne à arrêter), `t1` est tirée (`t2` n'est pas franchissable au sens de l'interprétation de la procédure `marcheZM22` !) et ZM22_Arret devient marquée avec le jeton provenant de ZM22_Marche.

Comme nous le verrons dans les pages suivantes, il est nécessaire de connaître la durée des arrêts. Celle-ci correspond au temps de séjour du jeton "ZM2_M/A" dans la place ZM22_Arret, qui se calcule par différence entre l'instant de simulation lors du franchissement de `t1` et l'instant de tir de `t3`. Les informations de début et de temps cumulé d'arrêt sont mémorisées sur les paramètres 1 et 2 du jeton. Ces opérations sont réalisées par les procédures `ZM2startT` et `ZM2stopT`.

Le modèle qui vient d'être présenté permet de prendre en compte les consignes T.O.R. de marche/arrêt ligne du TSX 87. Il est nécessaire de le compléter par un autre modèle pour simuler l'envoi de l'information de décalage de ligne vers l'automate. Ce second modèle est présenté sur la figure III.26. C'est un graphe de processus dans lequel circule un jeton de couleur "ZM2_Flux". Ce modèle utilise le marquage des deux places ZM22_Marche et ZM22_Arret du modèle précédent pour conditionner son évolution (représentation par des arcs incomplets vers ces places).

Pour expliquer le fonctionnement de ce modèle, supposons qu'une information de décalage vienne d'être envoyée. La place ZM22_A est alors marquée et temporisée avec un délai égal au temps nécessaire à la ligne pour avancer d'un pas (cette durée est une constante car la ligne progresse à vitesse constante). Lorsque le jeton "ZM2_Flux" devient disponible, si la ligne est en marche `t2` est franchie et la place ZM22_C marquée. Lors de ce franchissement, le param1 du jeton "ZM2_M/A" a été lu et réinitialisé et le premier paramètre de "ZM2_Flux" a pris sa valeur (procédures `NoteArret` et `MemoArret`).

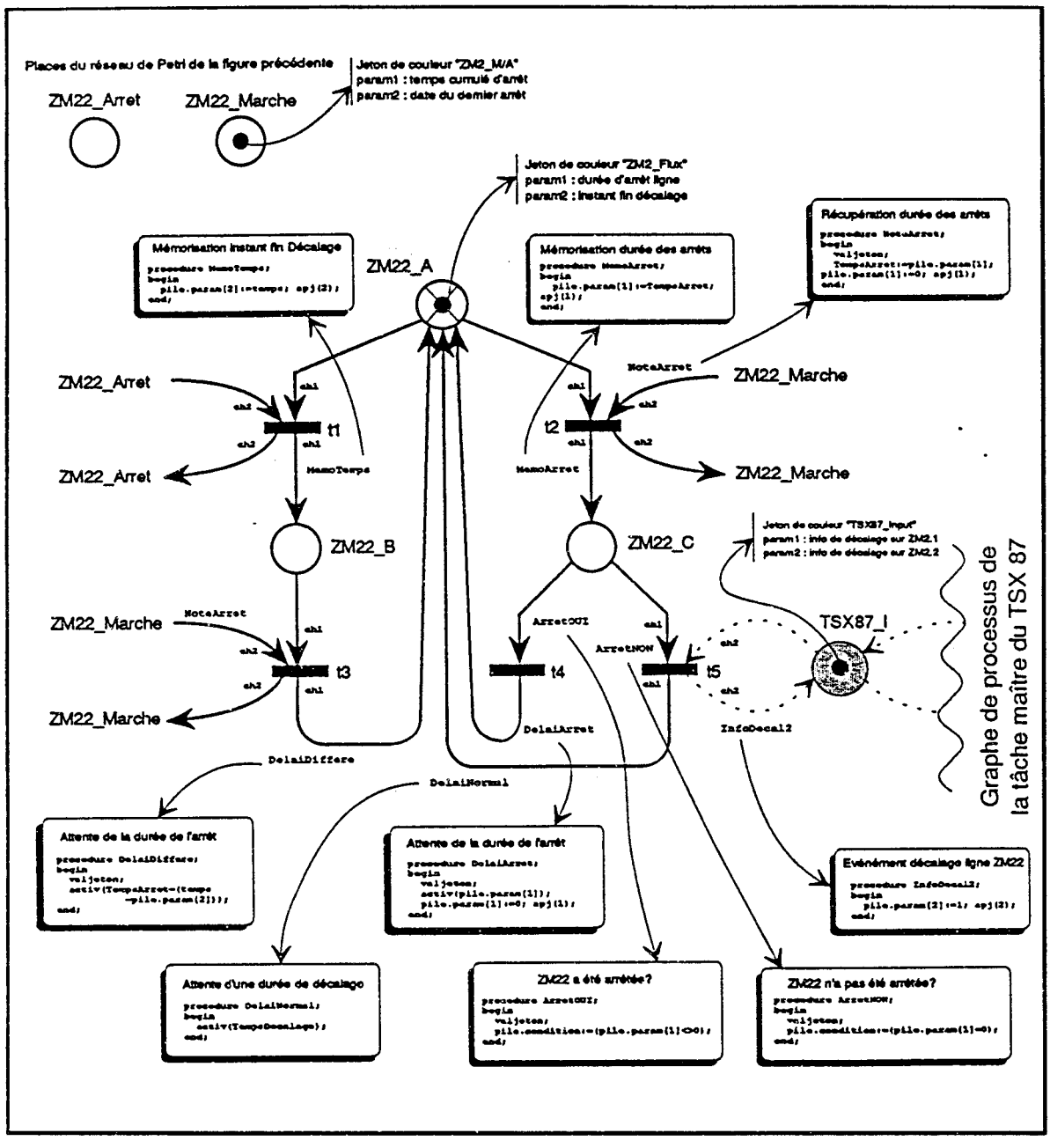


Figure III.26 : *Modèle de TempsDecalage de la ligne ZM2.2 avec les entrées du TSX87*

Si celle-ci est nulle, `t5` est franchit, l'information de décalage ligne envoyée a "TSX87_Input" et le jeton "ZM2_Flux" retourne en ZM22_A avec un état identique à celui du début (la procédure `DelaiNormal` fixe la temporisation de ZM22_A à la valeur constante `TempsDecalage`).

Dans le cas contraire, cela signifie que la ligne à été arrêtée pendant la durée du décalage de la ligne. Il est donc nécessaire d'attendre encore un délai égal à la durée de l'arrêt avant d'envoyer un signal d'avance ligne. La transition `t4` est tirée et la procédure `DelaiArret` temporise la place ZM22_A avec cette durée.

A l'expiration de ce délai, si la ligne n'a pas été arrêtée entre-temps `t2` puis `t5` sont franchies, le signal d'avance ligne transmis à "TSX87_Input" et la place ZM22_A marquée avec le jeton "ZM2_Flux" et temporisée avec une durée égale à `TempsDecalage` comme au début.

Le franchissement de `t1` correspond au cas où le jeton présent dans ZM22_A devient disponible, alors que la ligne est à l'arrêt. Ce jeton vient donc marquer la place ZM22_B et attend que la ligne se remette en marche. La procédure `MemoTemps` stocke dans le second paramètre du jeton l'instant du début de cette attente. Lorsque ZM22_Marche est à nouveau marquée, `t3` est franchie et la procédure `DelaiDiffere` temporise la place ZM22_A avec une durée égale au temps d'arrêt (récupéré par la procédure `NoteArret`) diminué du temps de séjour du jeton "ZM2_Flux" dans la place ZM22_B.

III.3.3. Modélisation des appels/réarmements opérateur

La modélisation des apparitions des appels/réarmements opérateurs sur les entrées T.O.R. des TSX17-20, consiste, pour chaque automate, à représenter sous la forme d'une loi probabiliste l'occurrence de tels événements. Le choix de la loi est exposé dans le paragraphe suivant de ce chapitre.

III. RESULTATS DE SIMULATION

Nous présentons dans cette partie le bilan de nos travaux de modélisation/simulation, appliqués à la zone ZM2 (Cf. figure III.8). Le modèle exposé dans le paragraphe précédent a été simulé et les résultats ainsi obtenus ont été comparés à des mesures réalisées sur l'installation de Rennes/La-Janais.

III.1. Simulation du modèle

III.1.1. Paramètres observés

Nous nous proposons d'observer les paramètres suivants :

- TCR = Temps de Cycle du Réseau Uni-Telway : temps qui sépare deux interrogations successives d'un même esclave,
- PCA = Prise en Compte Appel : délai entre l'apparition d'un événement (appel ou réarmement) sur l'entrée T.O.R. d'un TSX 17-20 et sa prise en compte effective dans le TSX 87 ANDON,
- DAL = Délai Avance Ligne : délai entre l'apparition de l'information de "décalage pas" sur l'entrée T.O.R. du TSX 87 et la mise à jour effective des voyants lumineux sur le dernier TSX 17-20 de la zone considérée (Cf. figure III.27),
- CCM = Charge du Coupleur Maître : nombre de messages présents dans le coupleur SCM 21.6 en attente de transfert vers la tâche maître du TSX 87.

III.1.2. Scénario de simulation

Le temps de cycle des automates est un paramètre important pour les performances du système. Pour réaliser les simulations, nous avons adopté les valeurs suivantes :

- TSX 87-30 : 100 ms (valeur supposée compte-tenu du système déjà existant),
- TSX 17-20 : la durée du temps de cycle n'est pas constante (tâche cyclique mais non périodique). Comme les TSX 17-20 sont très chargés (au dire des concepteurs !), nous avons pris une durée de cycle maximale, d'environ 150 ms. Pour tenir compte des fluctuations de cette valeur, elle est approximée par la fonction : TempsCycleTSX17 = 130+20*RANDOM(1).

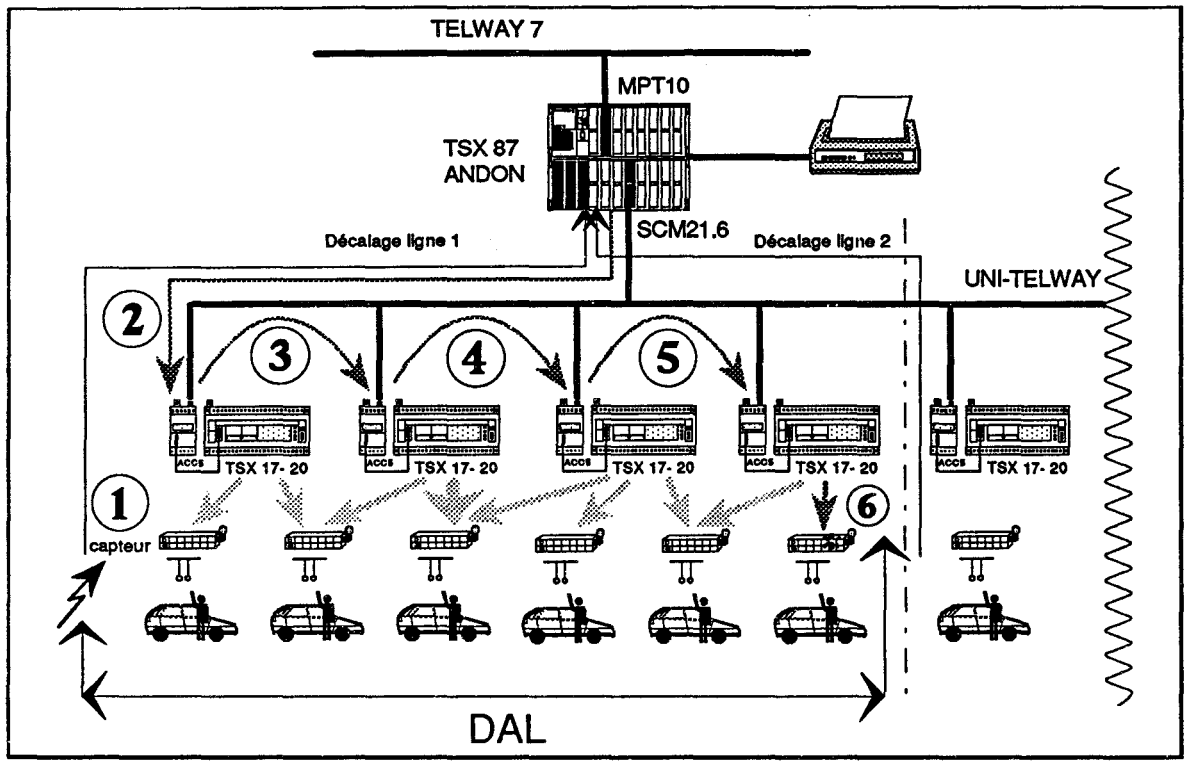


Figure III.27 : *Echanges liés au paramètre DAL (Délai Avance Ligne)*

Les échanges de messages sur cette architecture de commande sont déclenchés suite à des informations T.O.R. en provenance du procédé. Il est donc nécessaire d'en restituer une image aussi fidèle que possible. Pour cette étude, nous avons adopté :

- pour le **TSX 87-30** : une information T.O.R. "avance ligne" arrive à fréquence régulière, car la ligne évolue à vitesse constante (le délai entre 2 positionnements correspond au temps pour que la ligne avance d'un pas, soit environ 2 mn),
- pour chaque **TSX 17-20** : nous raisonnons sur la fréquence d'apparition des appels sur un côté (droit ou gauche), qui compte 14 pas. Après concertation avec les utilisateurs, nous avons supposé qu'un appel survenait en moyenne toute les deux minutes, et que le délai de réarmement pouvait être variable entre 0 et 10 minutes.

III.1.3. Résultats de simulation

La première chose à remarquer est que le fait de construire un modèle des échanges sur l'architecture impose de décrire précisément l'organisation des communications, travail qui n'est souvent réalisé que lors de la phase de programmation de l'application. Lorsque des erreurs sont détectées à ce stade avancé des travaux, il est alors toujours long et fastidieux de les corriger.

Pour le projet de l'atelier ANDON, la construction du modèle a permis de mettre en évidence un dysfonctionnement dans l'organisation des échanges telle qu'ils étaient envisagés au départ. Pour la gestion de l'avance des pas il avait été proposé que l'information de "décalage de ligne" soit envoyée en diffusion vers tous les TSX 17-20. Après réflexion, il est apparu que cette solution ne convenait pas car, pour garder une intégrité dans les données, il est nécessaire que l'information de décalage pas se propage de proche en proche entre les mini-automates.

Tous les résultats présentés dans ce paragraphe ont été obtenus par simulation du modèle décrit dans le paragraphes III.2. Il a été complété avec des "sondes" pour relever les paramètres à observer. Durant la simulation, ces données sont stockées dans des fichiers. Un programme de tracé d'histogramme permet d'en faciliter l'interprétation. Nous avons mené trois études successives:

a/ 2 automates TSX 17-20,

b/ 4 automates TSX 17-20 sur une zone (ZM2.1 ou ZM2.2),

c/ 8 automates TSX 17-20 sur deux zones (ZM2.1 et ZM2.2).

Chaque paramètre mesuré est caractérisé par sa valeur moyenne et la dispersion autour de cette moyenne. Afin d'observer l'influence du nombre de stations sur les performances, nous avons, pour chaque paramètre, regroupé sur un même graphique les valeurs obtenues pour chacune des études a, b et c.

Durant la simulation, nous stockons sur fichiers les suites d'événements qui correspondent au fonctionnement du système. Sur la figure III.28 nous reproduisons un exemple de sortie pour une simulation avec huit TSX 17-20. Ce document est indenté sur trois colonnes avec les significations suivantes (l'unité utilisée pour exprimer le temps est la milliseconde) :

- 1ère colonne : événement sur le procédé (appel opérateur,...),
- 2ème colonne : prise en compte de l'entrée T.O.R. par la tâche maître de l'automate,
- 3ème colonne : réception et émission de messages par les tâches principales.

Exemple :

```
APPEL  TSX17-1 cote 1 pas  8  instant     43841
  APPEL  TSX17-1 cote 1 pas  8 instant     43938
    APPEL emis  TSX17-1 pas  8 instant     43938
    APPEL recu  TSX87  du TSX17-1 pas  8 instant     44301
```

```
APPEL  TSX17-7 cote 1 pas  1  instant    49269
     APPEL  TSX17- 7 cote 1 pas  1 instant    49352
          APPEL emis  TSX17- 7 pas  1 instant    49352
          APPEL recu  TSX87  du TSX17- 7 pas  1 instant    49784
DECAL ZM2  instant    60000
     DECAL  ZM2 TSX87  instant    60040
          DECAL ZM2 emis  TSX87 vers TSX17-9 instant    60040
          DECAL recu  TSX17- 9 instant    60211
          DECAL emis  TSX17- 9 instant    60211
          DECAL recu  TSX17-11 instant    61086
          DECAL emis  TSX17-11 instant    61086
          DECAL recu  TSX17-13 instant    61774
          DECAL emis  TSX17-13 instant    61774
          DECAL recu  TSX17-15 instant    62400
APPEL  TSX17-1 cote 1 pas  1  instant    87679
     APPEL  TSX17- 1 cote 1 pas  1 instant    87754
          APPEL emis  TSX17- 1 pas  1 instant    87754
          APPEL recu  TSX87  du TSX17- 1 pas  1 instant    88239
DECAL ZM1  instant   120000
     DECAL ZM1  TSX87  instant   120057
          DECAL ZM1 emis  TSX87 vers TSX17-1 instant   120057
          DECAL recu  TSX17- 1 instant   120268
          DECAL emis  TSX17- 1 instant   120268
          DECAL recu  TSX17- 3 instant   121081
          DECAL emis  TSX17- 3 instant   121081
          DECAL recu  TSX17- 5 instant   121714
          DECAL emis  TSX17- 5 instant   121714
          DECAL recu  TSX17- 7 instant   122366
REARM  TSX17-1 pas  2  instant   143076
     REARM  TSX17- 1 pas  2 instant   143126
APPEL  TSX17-9 cote 1 pas 12  instant   152555
     APPEL  TSX17- 9 cote 1 pas 12 instant   152557
          APPEL emis  TSX17- 9 pas 12 instant   152557
          APPEL recu  TSX87  du TSX17- 9 pas 12 instant   153076
APPEL  TSX17-11 cote 1 pas  7  instant   153332
     APPEL  TSX17-11 cote 1 pas  7 instant   153411
          APPEL emis  TSX17-11 pas  7 instant   153411
          APPEL recu  TSX87  du TSX17-11 pas  7 instant   153695
APPEL  TSX17-15 cote 1 pas  6  instant   175434
     APPEL  TSX17-15 cote 1 pas  6 instant   175529
          APPEL emis  TSX17-15 pas  6 instant   175529
          APPEL recu  TSX87  du TSX17-15 pas  6 instant   175758
DECAL ZM2  instant   180000
     DECAL  ZM2 TSX87  instant   180070
          DECAL ZM2 emis  TSX87 vers TSX17-9 instant   180070
          DECAL recu  TSX17- 9 instant   180360
          DECAL emis  TSX17- 9 instant   180360
          DECAL recu  TSX17-11 instant   181219
          DECAL emis  TSX17-11 instant   181219
          DECAL recu  TSX17-13 instant   181947
          DECAL emis  TSX17-13 instant   181947
APPEL  TSX17-15 cote 0 pas  3  instant   182212
     APPEL  TSX17-15 cote 0 pas  3 instant   182338
          APPEL emis  TSX17-15 pas  3 instant   182338
          DECAL recu  TSX17-15 instant   182767
          APPEL recu  TSX87  du TSX17-15 pas  3 instant   182941
APPEL  TSX17-7 cote 1 pas  3  instant   185055
     APPEL  TSX17- 7 cote 1 pas  3 instant   185192
          APPEL emis  TSX17- 7 pas  3 instant   185192
          APPEL recu  TSX87  du TSX17- 7 pas  3 instant   185716
```

Figure III.28 : *Séquence temporisée d'événements de simulation avec 8 TSX 17-20*

L'opérateur déclenche un appel au pas 8 du 1er TSX17 à l'instant 43 s 841 ms. Cet appel est pris en compte par le TSX17-1 à l'instant 43 s 938 ms. Le TSX 17-1 envoie un message pour signaler l'appel au TSX87 à l'instant 43 s 938 ms. Le TSX 87 reçoit un message du TSX17-1 pour l'appel au pas 8 à l'instant 44 s 301 ms.

III.1.3.a Temps de Cycle Réseau : TCR

Le temps de cycle du réseau Uni-Telway est le temps qui sépare deux interrogations successives d'un même esclave par le maître de la liaison. Ce temps est lié au séquencement du protocole d'accès au médium utilisé (maître/esclave) et à la taille et la fréquence des requêtes qui sont échangées.

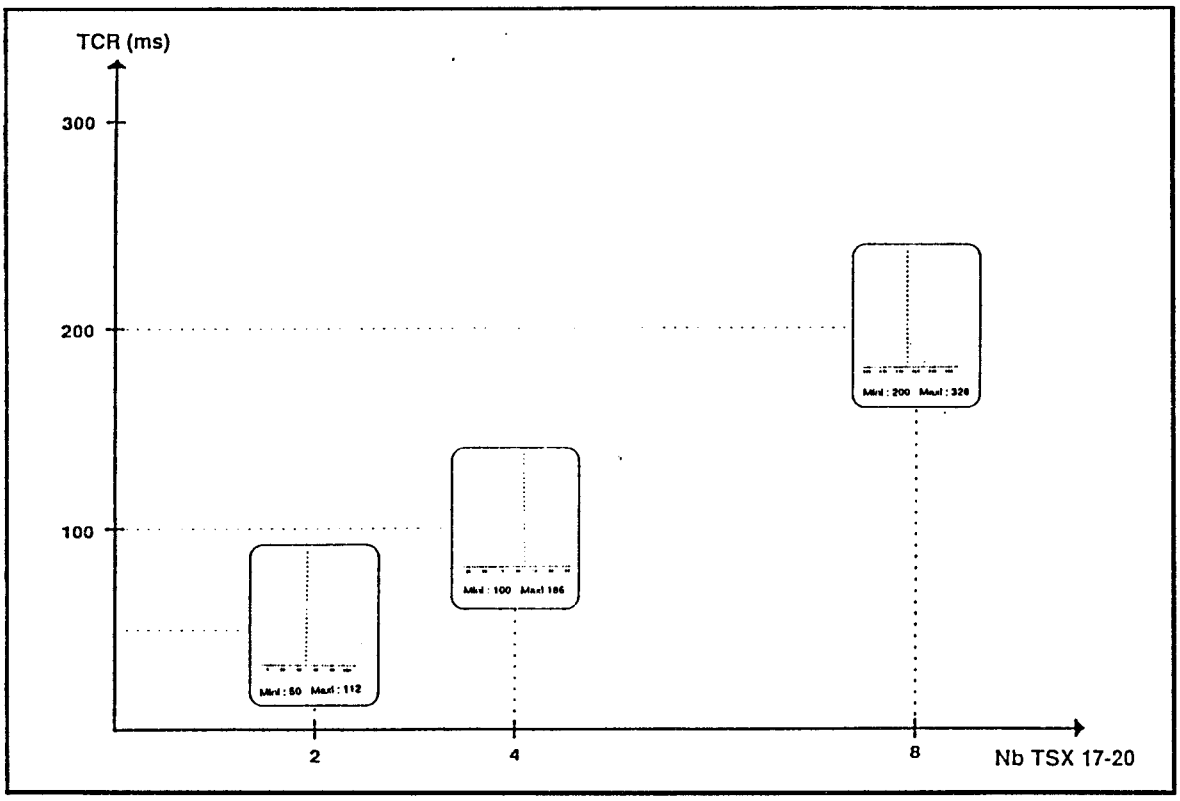


Figure III.29 : *Résultats sur le Temps de Cycle Réseau (TCR)*

Ces résultats montrent que le réseau est peu chargé. En effet, dans la grande majorité des cas, la valeur mesurée du TCR correspond à un cycle sans échange de message. Lorsque des données utiles sont véhiculées sur le réseau (requête ou compte-rendu UNI-TE), le TCR est augmenté des temps de transfert et de traitement associés à ces informations. C'est par exemple les cas avec huit mini-automates TSX 17-20 lorsque le cycle réseau vaut 328 ms.

III.1.3.b Prise en Compte Appel : PCA

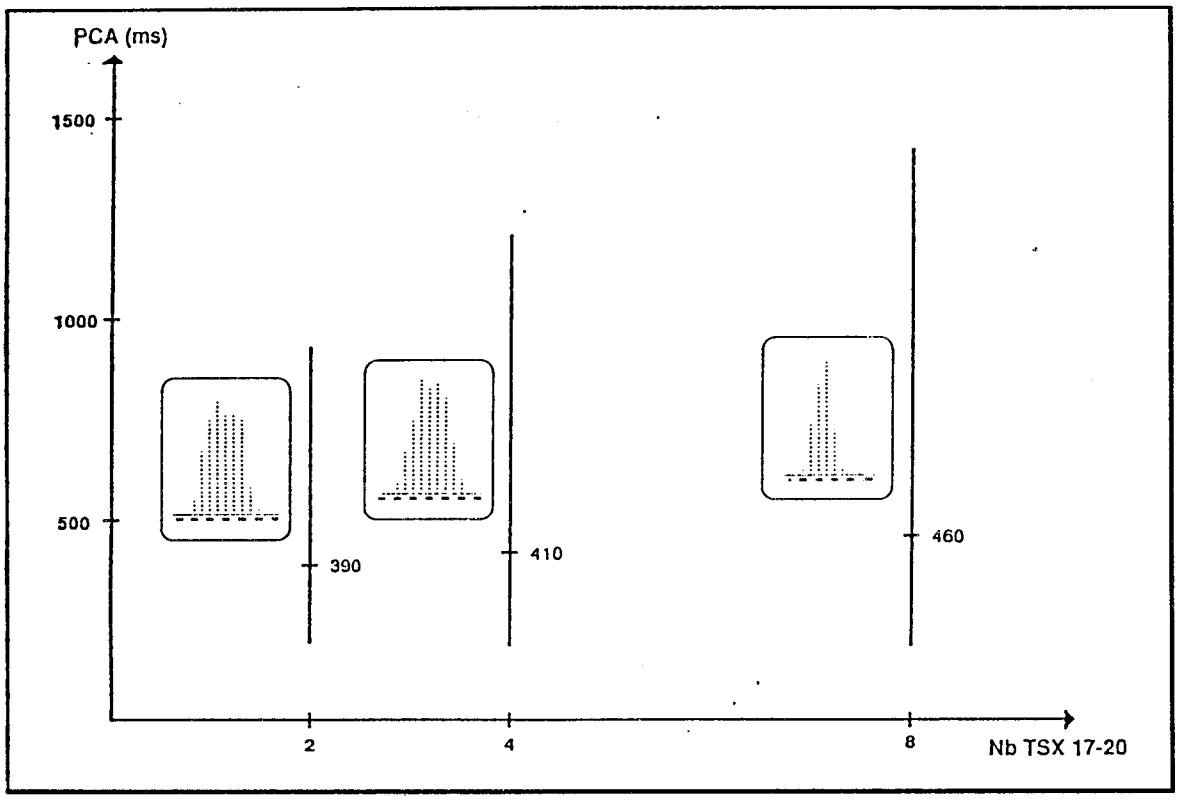


Figure III.30 : *Résultats sur le temps de Prise en Compte d'un Appel (PCA)*

Le temps de prise en compte d'un appel correspond au délai entre l'apparition d'un événement (appel ou réarmement) sur l'entrée T.O.R. d'un TSX 17-20 et sa prise en compte effective dans le TSX 87 ANDON. Il se décompose ainsi :

- prise en compte de l'entrée T.O.R. sur le 17-20 (au maximum un cycle automate),
- traitement de l'entrée et armement d'un bloc TXT en émission (au minimum un cycle automate, mais éventuellement plus si d'autres messages sont déjà dans la FIFO),
- prise en compte de la requête par le coupleur ACC 5 (entre 0 et 2 cycles automate),
- transit de la requête vers le maître (au minimum 20 ms, au maximum TCR + 20 ms),
- transfert de la requête vers l'UC du TSX 87 (entre 0 et n cycles automates si d'autres messages sont déjà présents dans la mémoire du coupleur SCM 21.6).

Nous voyons que ce temps varie peu avec le nombre de TSX 17-20. Ceci est normal, puisqu'avec deux ou huit automates, les durées liées aux traitements dans les coupleurs ou les automates sont les mêmes. Seul varie le TCR qui passe de 50 à 200 ms ; on retrouve bien en moyenne un écart égal à la moitié de l'augmentation du TCR.

III.1.3.c Décalage Avance Ligne : DAL

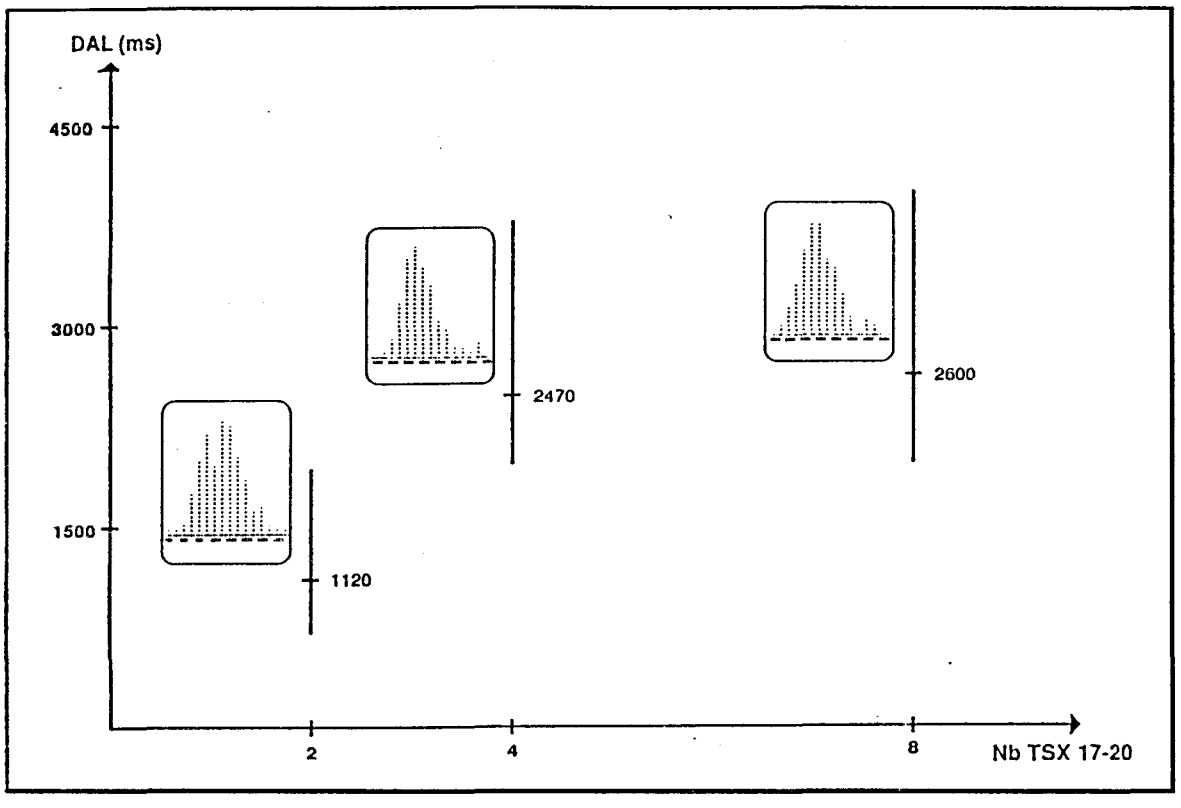


Figure III.31 : *Résultats sur les temps de Décalage Avance Ligne : DAL*

Le temps de décalage ligne correspond au délai entre l'apparition de l'information avance ligne sur l'entrée T.O.R. du TSX 87 et la mise à jour effective des voyants lumineux sur le dernier TSX 17-20 de la zone considérée. Ce décalage nécessite plusieurs communication successives d'esclave à esclave (Cf. figure III.27) ce qui explique pourquoi le délai se mesure en secondes. On remarque qu'avec quatre ou huit mini-automates, ce temps varie peu, ce qui est normal puisqu'avec huit TSX 17-20, la ligne se décompose en deux zones (ZM2.1 et ZM2.2) de quatre automates chacune. La différence de 130 ms s'explique par l'augmentation du TCR.

III.1.3.d Charge Coupleur Maître : CCM

La charge du coupleur maître correspond au nombre de messages en attente dans le coupleur SCM 21.6 d'une prise en compte par la tâche maître du TSX 87. Ce dernier ne peut en effet lire qu'un seul message (requête ou compte-rendu) par cycle automate.

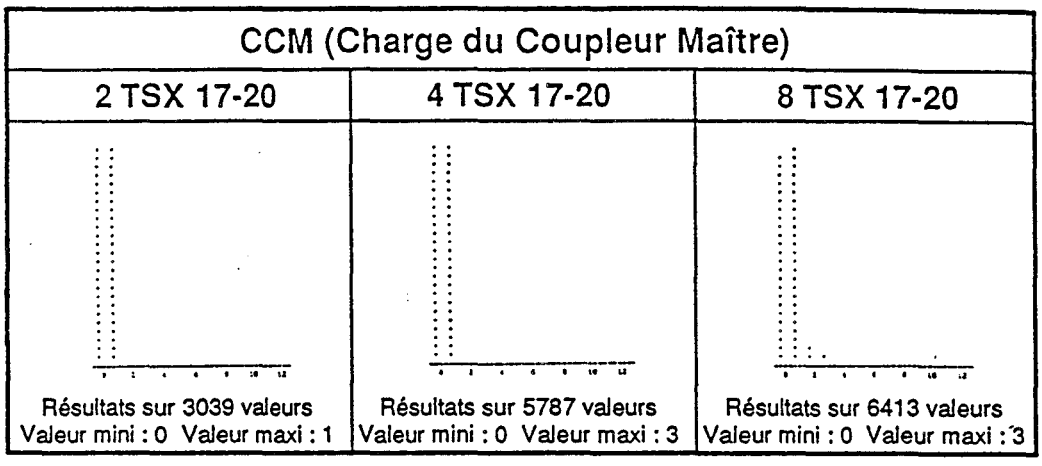


Figure III.32 : ***Résultats sur la Charge du Coupleur Maître (CCM)***

Ce paramètre est intéressant car la taille de cette zone d'échange est de 3 messages. Si un message arrive alors que la zone est pleine, celui-ci est refusé (NACK). Nous voyons qu'avec deux TSX 17-20 il y a au maximum un message en attente, mais qu'avec huit mini-automates, il peut y en avoir jusqu'à 3. Ceci s'explique facilement car plus il y a de TSX 17-20, plus il y a de messages émis vers le TSX 87.

Il est donc nécessaire au niveau de la programmation de l'envoi des messages dans les TSX 17-20 de prévoir un time-out avec réémission éventuelle ; en effet, si un mini-automate envoie un message alors que 3 requêtes sont déjà en attente dans le coupleur maître, celui-ci ne sera pas pris en compte. Il est donc vital de le renvoyer au bout d'un certain temps d'attente !

Remarque :

Ces graphiques représentent des nombres d'états, avec 1, 2 ou n messages en attente. C'est pourquoi pour CCM avec deux TSX 17-20 par exemple, il y a autant d'états avec aucun message que d'états avec un message. Cette représentation n'est pas similaire à celle qui montrerait la durée cumulée de chacun des états !

III.2. Adéquation modèle / Réalité

Pour vérifier les résultats qui viennent d'être présentés, nous avons réalisé des mesures sur l'installation, une fois celle-ci mise en place. Notre attention s'est portée sur le paramètre DAL car il est représentatif du fonctionnement global du système sur le réseau Uni-Telway : il inclut des communications entre le TSX 87 et un TSX 17-20 et entre deux mini-automates.

III.2.1. Mesures sur l'installation

III.2.1.a Mesures avec deux TSX 17-20

Les mesures ont été effectuées sur la plate-forme de développement avec un réseau Uni-Telway sur lequel sont connectés deux automates TSX 17-20 et un TSX 87. Le temps de cycle du TSX 87 est fixé par configuration à 100 ms. Pour les TSX 17-20, il est variable selon les cycles et a été mesuré entre 90 et 110 ms.

1200	1450	1300	900	1300	1150	950	900	1300	1050
1000	950	1000	1500	1000	1000	1400	100	900	950
1700	950	1100	950	1000	1600	1050	1000	950	950
1400	1250	1000	1000	950	1100	900	1050	900	1400

Tableau III.3 : *Valeurs mesurées (en ms) pour DAL avec deux TSX 17-20*

III.2.2.b Mesures dans l'atelier avec huit TSX 17-20

Les mesures ont été effectuées sur le site de production, sur la zone ZM2 de l'atelier RM1/RM2 (Cf. figure III.7). Un automate TSX 87-30 est connecté à huit automates TSX 17-20. Réaliser ces mesures n'a pas été une tâche facile, puisque cela a notamment nécessité de tirer 500 m de câble! Le temps de cycle du TSX 87 est fixé par configuration à 80 ms. Celui des TSX 17-20 est variable et fluctue entre 90 et 110 ms

2250	1950	2300	2720	2400	1960	2150	2100	1970	2100	2000	1890
1900	2460	2350	2120	2100	2320	2250	2020	2320	1900	2310	2090
2600	1940	2000	1920	2680	1930	2200	2310	1900	2590	2420	2280
2080	2410	2020	2160	2230	2270	1920	2140	2120	1880	2130	2190
1930	2640	2120	2380	2220	2110	2340	2220	2100	1930	2010	1890
2360	1960	1990	2810	2540	2080	1940	2520	2000	2080	1930	2420
2010	2020	2340	2050	2070	2030	1930	1880	2430	2430	2600	2420
2610	2120	2520	2330	2230	2610	2280	2390	2080	2580	1900	2280
1900	2070	1980	2000	2020	9300	6150	6200	8950	9050	8600	6700
7900	6500	9400	8850	9140	7300						

Tableau III.4 : *Valeurs mesurées (en ms) pour le paramètre DAL avec huit TSX 17-20*

II.2.2. Comparaison simulation / mesures

Afin de comparer les résultats mesurés a ceux obtenus par simulation, nous avons, pour chaque cas étudié, représenté sur le même graphique, les deux types de valeurs.

II.2.2.a Configuration avec deux TSX 17-20

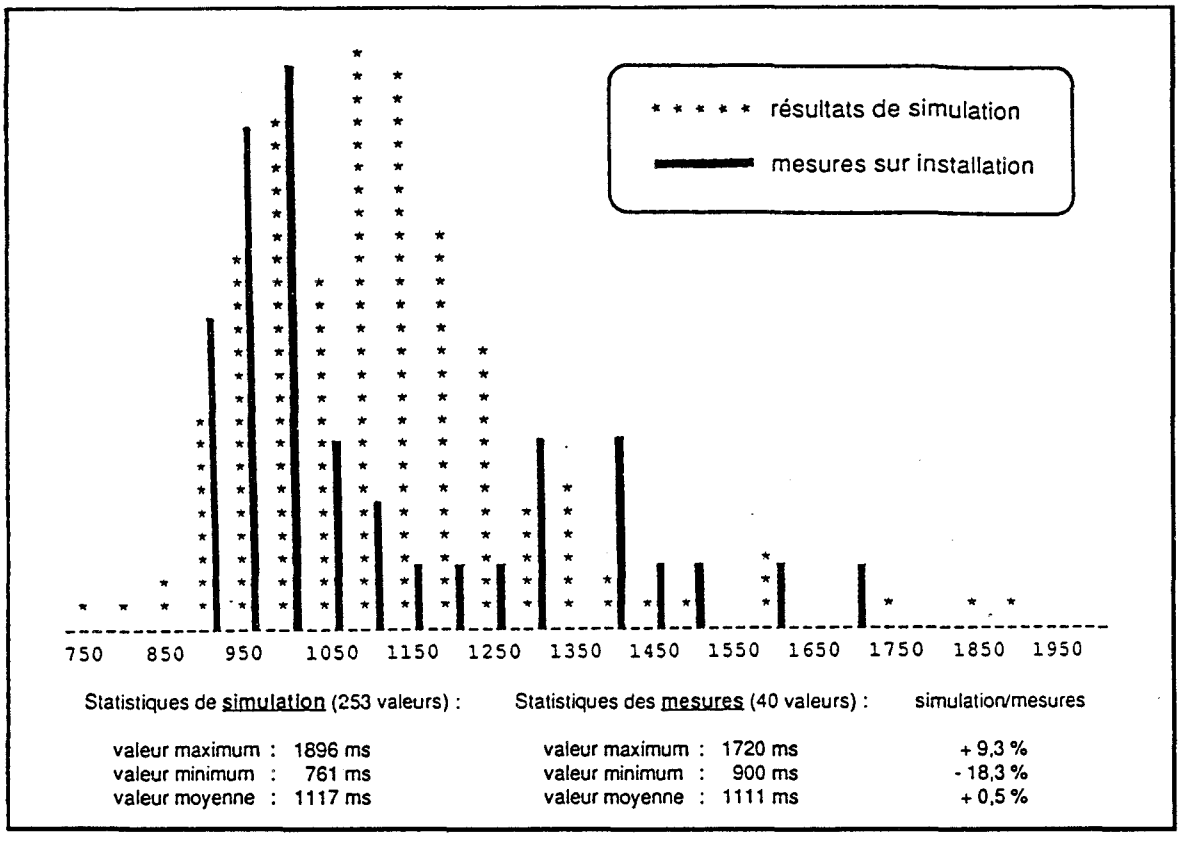


Figure III.33 : *Comparaison des valeurs simulées et mesurées du paramètre DAL*

Nous voyons qu'avec les mesures sur l'installation, nous retrouvons une répartition similaire à celle qui avait été mise en évidence par la simulation. L'aspect plus "grossier" de la répartition des valeurs mesurées s'explique par le nombre de mesures effectuées (296 pour la simulation et 40 sur l'installation). Les valeurs mesurées apparaissent globalement un peu plus faibles (écart de l'ordre de 10 à 20 %) que celles obtenues par simulation.

A priori nous ne connaissions par le temps de cycle des TSX 17-20. Comme nous savions qu'ils étaient chargés, nous avons choisi de prendre pour notre modèle un temps de cycle élevé : 130 + 20 x RANDOM(1) ms.

Nous avons été "pessimistes", puisqu'une fois les tâches de commande programmées, ce temps de cycle s'est trouvé être compris entre 90 et 110 ms! Plus les temps de cycles sont élevés, plus les durées des échanges sont importantes ; l'imprécision sur l'estimation des temps de cycle augmente donc les valeurs du DAL obtenues par simulation.

Dans nos modèles, nous avons supposé que les entrées T.O.R. étaient prises en compte dans le premier cycle automate qui suivait leur apparition. Pour des raisons pratiques de mise en place, il a été nécessaire d'imposer un délai de 300 ms avant de considérer effectivement l'information d'avance ligne sur l'entrée T.O.R. de l'automate ANDON. Ce délai, qui n'a pas été simulé, augmente la valeur du DAL.

La conjonction de ces deux phénomènes ne permet pas d'avoir une estimation suffisamment fine de la précision de notre modèle. C'est pourquoi nous l'avons corrigé pour réaliser une simulation a posteriori qui est présentée dans le paragraphe II.2.3.a

III.2.2.b Configuration avec huit TSX 17-20

Les résultats sont présentés en page suivante. Nous voyons que les valeurs mesurées se divisent en deux paquets bien distincts. Un premier groupe, composé de 101 mesures, comporte des temps entre 1880 ms et 2810 ms. Un second groupe de 13 valeurs présente des durées variant de 6200 ms à 9400 ms.

Les programmes implantés dans les automates sont configurés avec un time-out de 5 secondes pour les émissions de messages sur le réseau Uni-Telway. Lorsqu'une station émettrice d'une requête n'a pas reçu de compte-rendu au bout de 5 secondes elle réémet une seconde fois le message. Nous pensons donc que les 13 valeurs élevées sont dues à des phénomènes d'erreurs de transmission sur le bus Uni-Telway dans l'atelier.

Ce phénomène n'a pas été détecté par la simulation, car dans nos modèle nous supposons que les transmissions se passent bien. Le seul cas où il peut y avoir réémission de message est celui où le message est refusé suite à un dépassement de capacité de la file d'attente pour stocker les messages dans un coupleur.

Pour la centaine de mesures autour de 2 secondes les mêmes remarques que pour le système avec deux TSX 17-20 s'appliquent : estimation trop élevée du temps de cycle des TSX 17-20 et non prise en compte du délai de 300 ms. De plus, les mesures dans l'atelier ont été réalisées "à vide", c'est à dire avec uniquement des échanges de messages de "décalage de ligne".

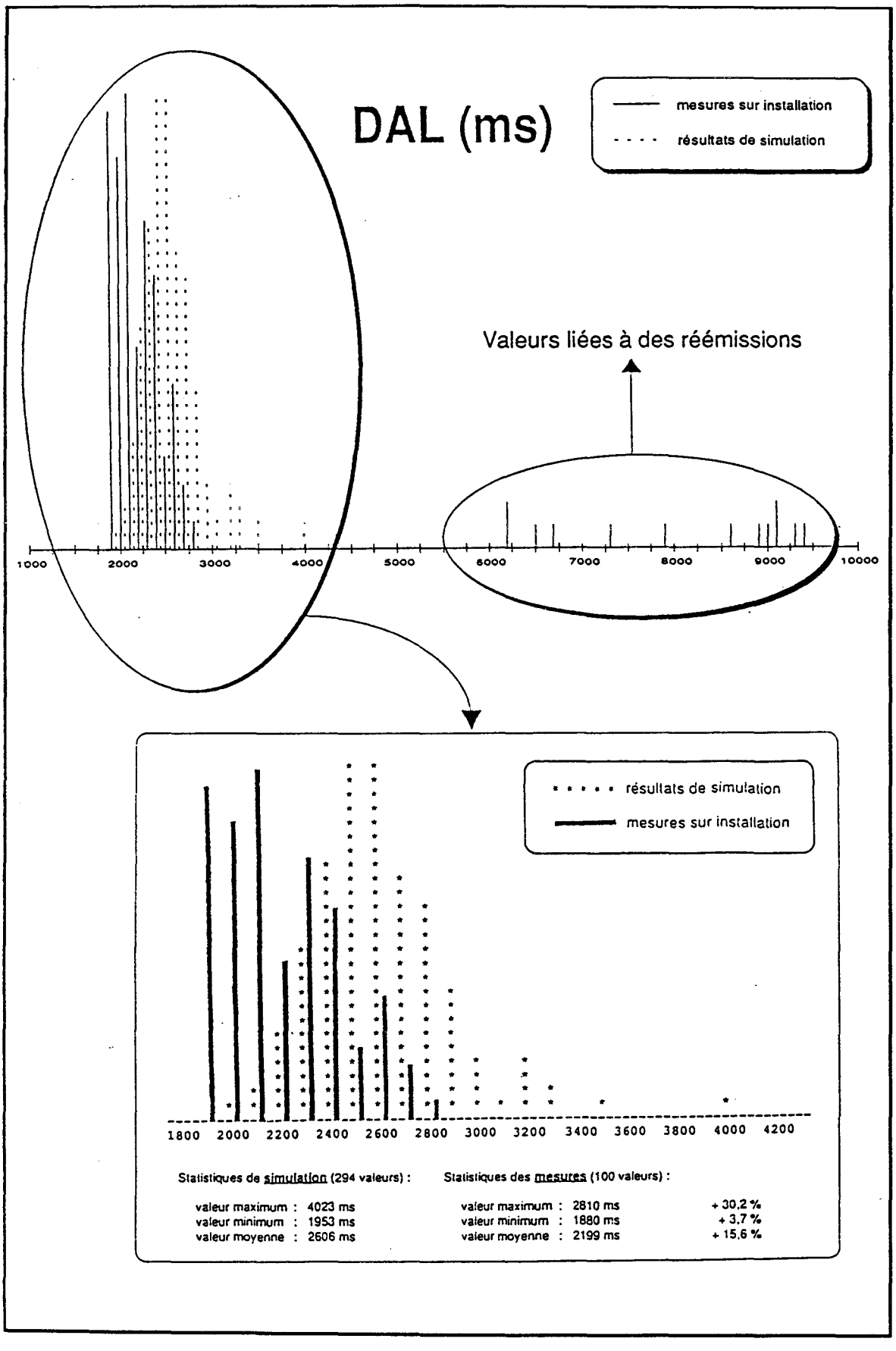


Figure III.34 : *Comparaison des valeurs simulées et mesurées du paramètre DAL*

Dans nos scénarios de simulations, nous avons étudié le système "en charge", c'est à dire avec des échanges de messages d'appel, d'arrêt ou de réarmement. Nous avons simulé en moyenne un appel opérateur toutes les minutes sur chaque TSX 17-20. Cette hypothèse créé un nombre important de messages d'appel, ce qui explique pourquoi, au cours de la simulation, ces messages entrent parfois en conflit avec ceux de "décalage ligne". Les valeurs élevées, notamment celle de 4023 ms, sont dues aux "encombrements" crées par ces messages supplémentaires.

III.2.3. Simulations a posteriori

III.2.3.a Configuration avec deux TSX 17-20

Afin de pouvoir mieux juger la fiabilité du modèle, nous avons refait une simulation en prenant en compte le temps de cycle réel des TSX 17-20, soit 90 + 20 x RANDOM(1), ainsi que le délai de 300 ms imposé pour la prise en compte de l'information d'avance ligne sur l'entrée T.O.R. du TSX 87. La spécification de ce délai a nécessité une modification au niveau de la description de l'applicatif du TSX 87. Nous voyons cette fois-ci que la précision est de l'ordre de 10 %.

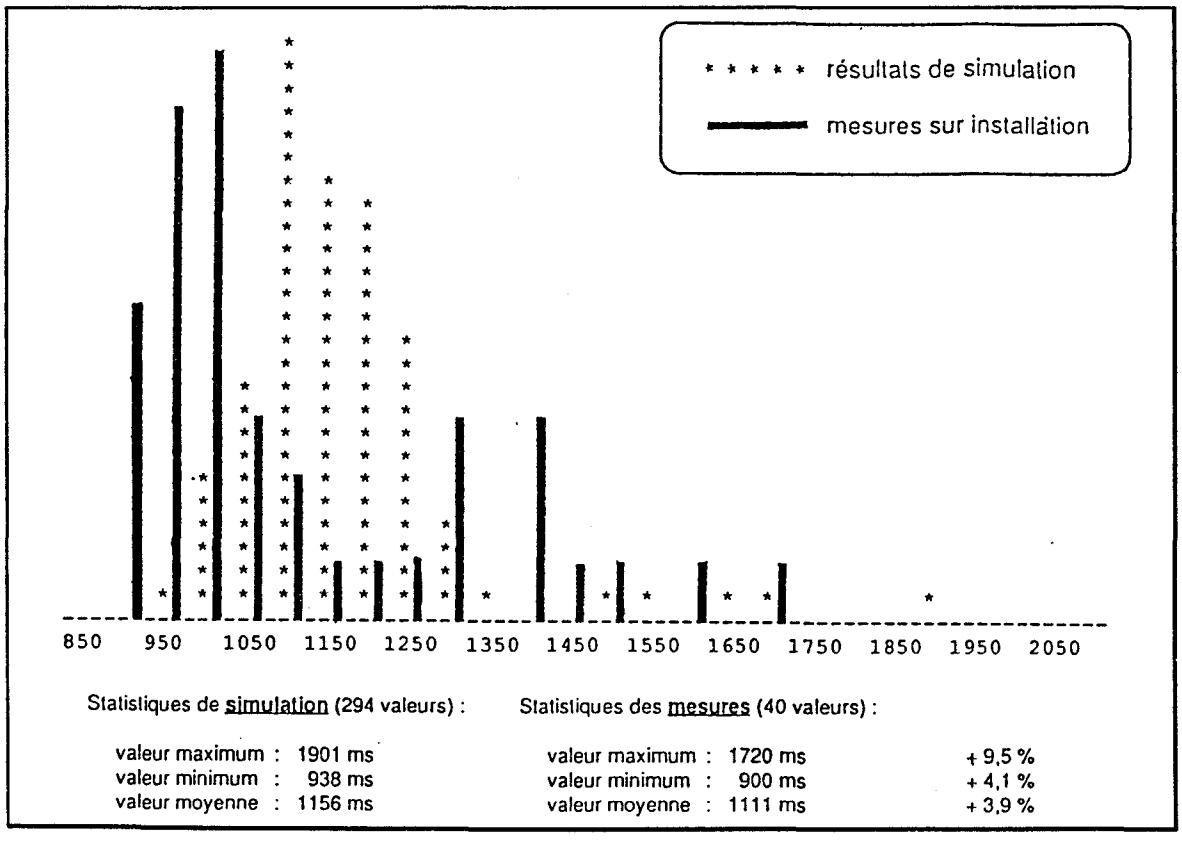


Figure III.35 : *Comparaison a posteriori des valeurs simulées et mesurées de DAL*

III.2.3.b Configuration avec huit TSX 17-20

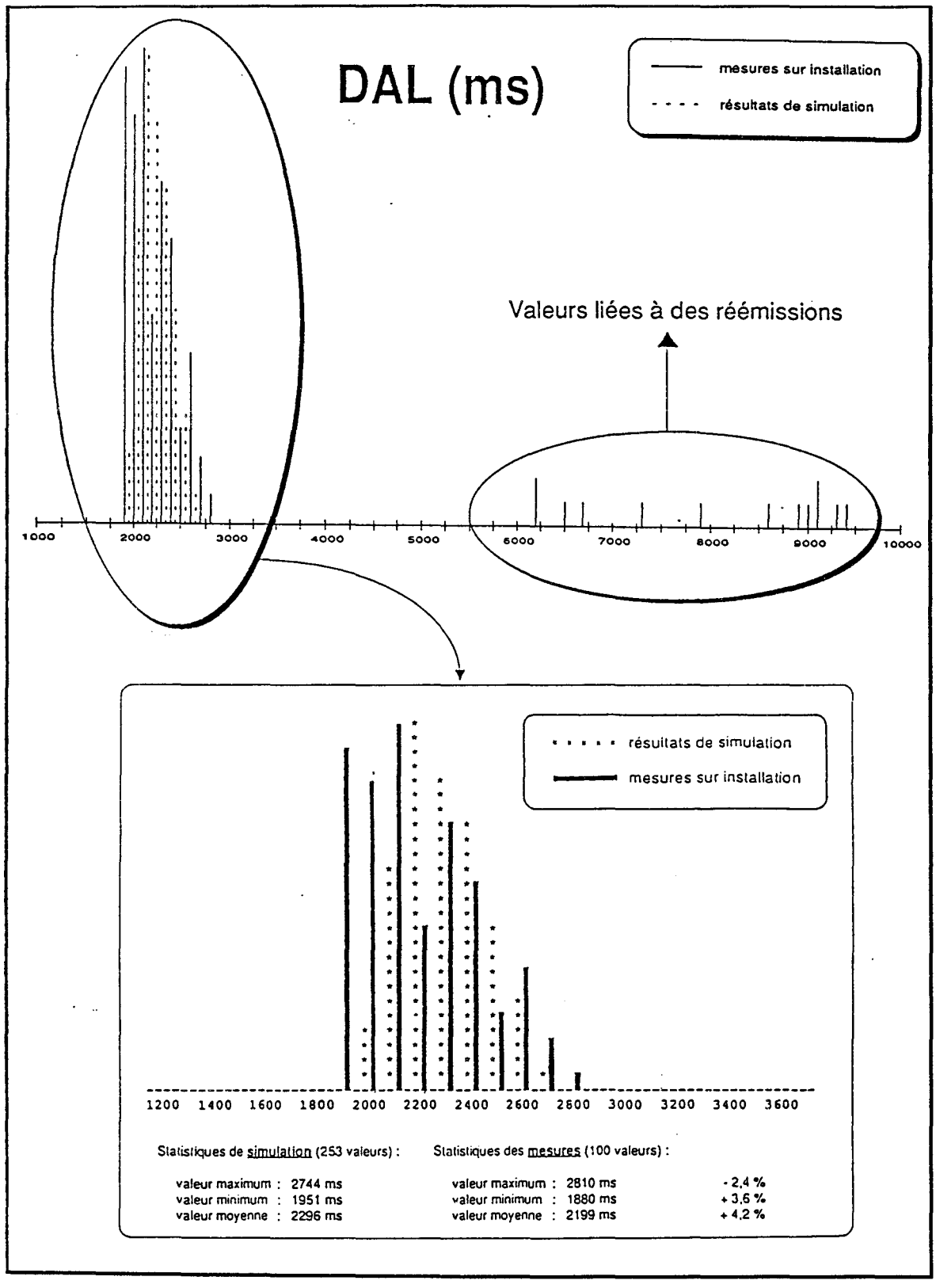


Figure III.36 : *Comparaison a posteriori des valeurs mesurées et simulées de DAL*

Nous avons réalisé une simulation avec cette fois-ci le temps de cycle réel des TSX 17-20 et le délai de 300 ms pour la prise en compte de l'information "avance ligne". Celle-ci a été réalisée "à vide", c'est à dire avec uniquement des messages de décalage de pas. Nous voyons alors que, comme dans le cas précédent, la précision est inférieure à 10 %.

III.3. Conclusion de cette étude

Ces travaux constituent une première étude complète de faisabilité d'une approche de modélisation/simulation consistant à l'évaluation des performances des architectures de commande du groupe PSA. Les résultats sont résumés sur le graphique suivant :

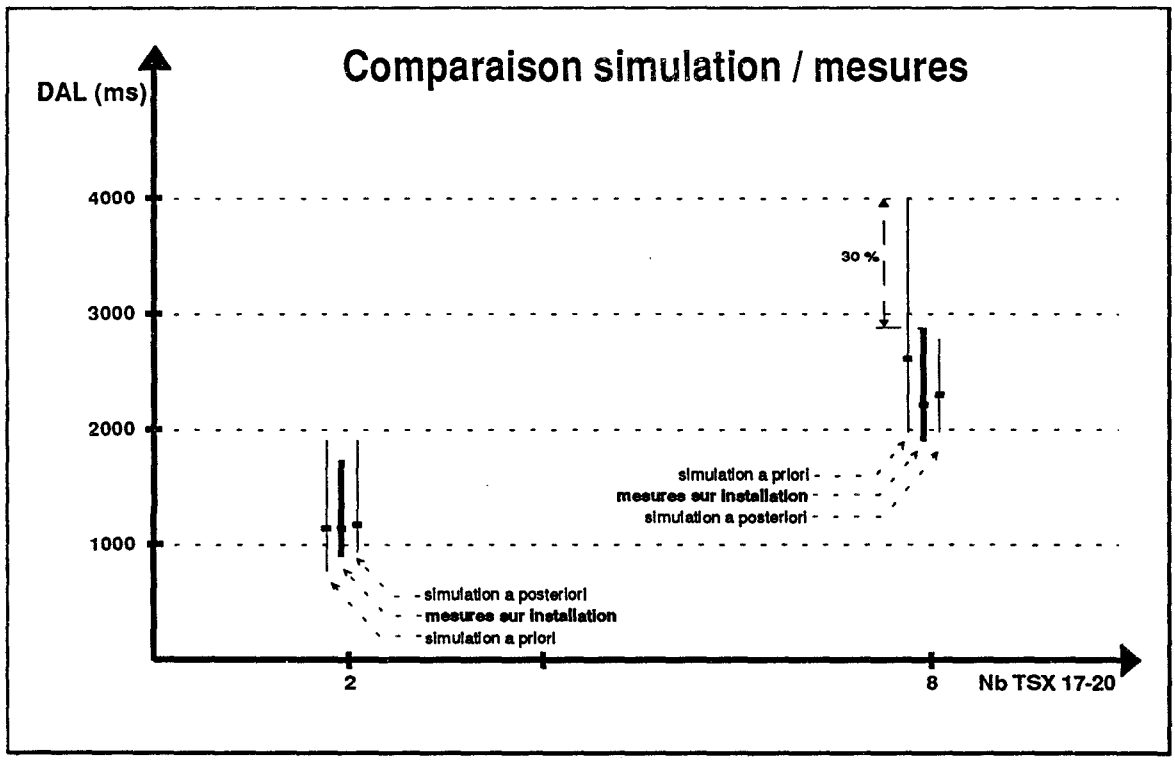


Figure III.37 : *Synthèse de la comparaison des valeurs mesurées et simulées de DAL*

Il est tout d'abord intéressant de remarquer que dans tous les cas, l'écart maximum entre les durées estimées par simulation et celles mesurées dans l'atelier est de 30 %. Lorsque les hypothèses prises sur le système pour la simulation sont effectivement celles adoptées pour l'installation réelle, les écarts relevés entre la simulation et les mesures sur site sont inférieures à 10 %.

IV. VERS UNE ETUDE DU SYSTEME COMPLET

IV.1. Architecture multi-réseaux

L'étude présentée dans les paragraphes précédents de ce mémoire considère un sous-ensemble de l'architecture du système de suivi de fabrication de l'atelier de montage RM1/RM2. Nous allons maintenant montrer comment, à partir des travaux réalisés, le système pourrait être analysé dans sa globalité.

IV.1.1. Architecture matérielle

Il s'agit d'étudier l'architecture ANDON complète qui se compose de deux réseaux Uni-Telway pour les zones ZM2 et GMP/POM et d'un réseau Telway 7 qui relient les automates ANDON au concentrateur. Ce dernier est ensuite connecté au micro-ordinateur PC par une liaison Uni-Telway.

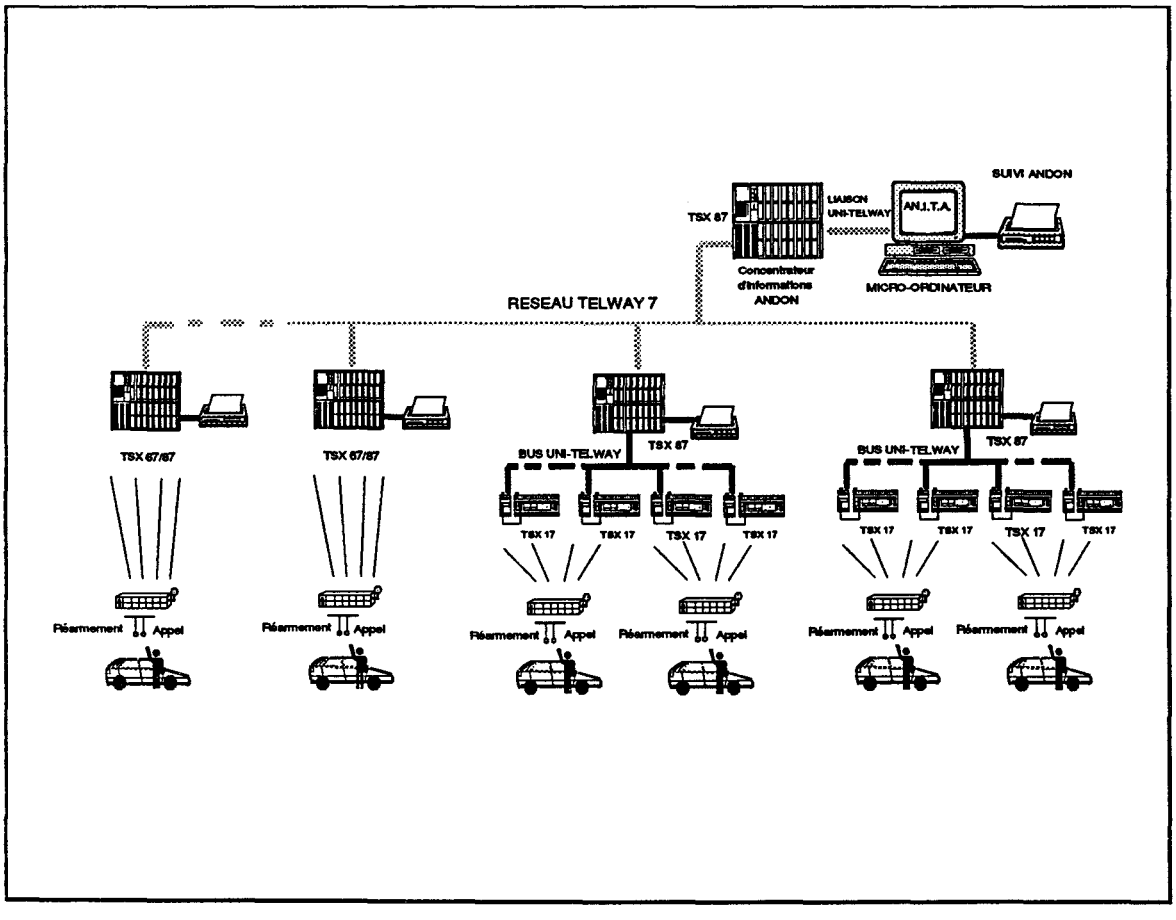


Figure III.38 : *Architecture multi-réseaux à étudier*

IV.1.2. Fonctionnement des échanges

Connectés au réseau Telway 7, les huit automates ANDON transmettent les informations de leur zone (apparition d'un appel, d'un arrêt ou disparition d'un arrêt) vers l'automate concentrateur. Pour les secteurs rénovés, ces informations viennent directement des entrées T.O.R. de l'automate ANDON. Pour les nouvelles zones (ZM2 et GMP/POM), ces données sont issues de messages reçus en provenance des TSX 17-20.

La modélisation du système complet impose de connaître les applicatifs de chaque station. Les échanges sur le réseau Telway 7 utilisent les mots communs (COM) et l'envoi de données par bloc texte [TEL 89].

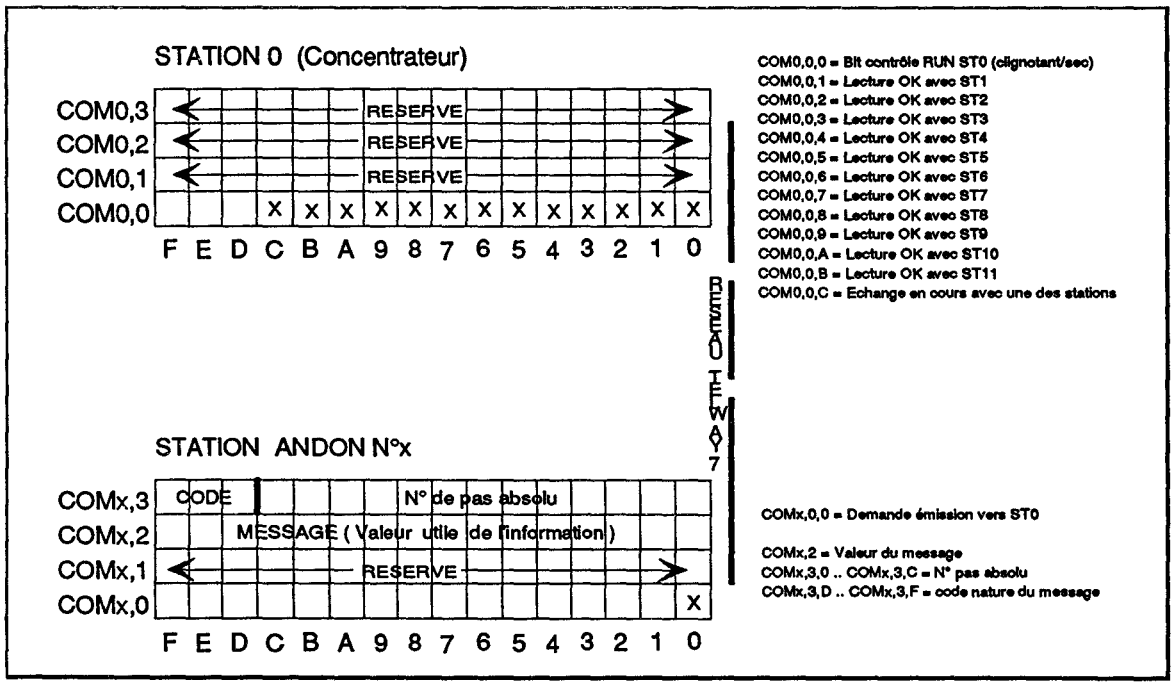


Figure III.39 : *Structure des mots COM pour le dialogue Concentrateur/Station ANDON*

Trois informations sont transmises par les mots COM :

- le code du message : 001 pour un appel, 010 pour un arrêt et 100 pour un réarmement
- le numéro du pas où s'est produit l'événement,
- le contenu du message (le nombre d'appels ou d'arrêts cumulés ou le temps d'arrêt).

Pour compléter l'information de disparition d'arrêt (code 100), quatre mots de 16 bits sont envoyés sur Telway 7 par bloc texte, pour remonter au concentrateur des informations sur des temps de cumul d'arrêt pour le pas, l'U.H.T. et la Z.T.

Pour émettre un message vers le concentrateur, chaque automate ANDON vérifie qu'aucun dialogue n'est en cours (COM0,0,C = 0) et fait une demande d'émission via ses mots COM. Si dans un délai fixé (environ 3 secondes), le concentrateur n'a pas reconnu le message (COM0,0,n°station = 1) alors un nouvel essai est réalisé avec le même message. Lorsque le message est compris, le concentrateur positionne le bit "lecture OK" de la station à 1, ce qui entraîne la disparition de la demande d'émission de l'automate de cette station. Les applicatifs des stations sont résumés ci-dessous.

EMISSION de l'automate ANDON

```
SI pas "échange en cours" avec une autre station ALORS

    { Traitement d'une émission vers le concentrateur }

    SI pas apparition "nouvel essai d'émission" (bit interne) ALORS
        SI pas "demande d'émission" (COM x,0,0 = 0) ET
                                Appel, Arrêt ou Réarmement (file d'attente) ALORS
            Charger le message à émettre
                (COM x,3, COM x,2 et Bloc TxT si message de disparition d'arrêt)
            Se positionner en demande d'émission vers concentrateur (COM x,0,0 <- 1)
            Lancer la temporisation du temps enveloppe
        FinSI
    SINON
        Réémettre message
        Relancer la temporisation du temps enveloppe
    FinSI

    { Traitement  d'une réponse du concentrateur ou d'un time-out }

    SI acquittement des données par le concentrateur (COM 0,0,x = 1) ALORS
        Supprimer demande d'émission  (COM x,0,0 <- 0 et "nouvel essai d'émission" <- 0)
    SINON
        SI temps enveloppe dépassé et "demande d'émission" ALORS
            Essayer une nouvelle émission ("nouvel essai émission" <- 1)
        SINON
            Traiter  l'erreur dans les échanges
        FinSI
    FinSI

FinSI
```

Figure III.40 : *Applicatif de l'automate ANDON*

RECEPTION du CONCENTRATEUR

{ Traitement d'une demande d'émission d'une station ANDON (n° x) }

SI "demande d'émission" d'une des stations *(COM x,0,0 = 1)* **ALORS**
Déclaration d'un échange en cours *(COM 0,0,C <- 1)*
Lecture et traitement des informations
(COM x,3, COM x,2 et Bloc TxT si message de disparition d'arrêt)
Acquittement de la lecture *(COM 0,0,x <-1)*
FinSI

{ Traitement de la fin des échanges avec la station ANDON (n° x) }

SI suppression de "demande d'émission" de la station **ALORS**
Statuer la fin de l'échange *(COM 0,0,x <- 0 et COM 0,0,C <- 0)*
FinSI

Figure III.41 : *Applicatif de l'automate concentrateur*

IV.1.3. Paramètre à observer

Un paramètre intéressant à observer sur l'architecture complète est le délai entre l'apparition d'un appel sur un pas d'une nouvelle zone et sa prise en compte effective par la tâche maître de l'automate concentrateur. Nous appelerons cette durée DRA (Délai Remontée Appel).

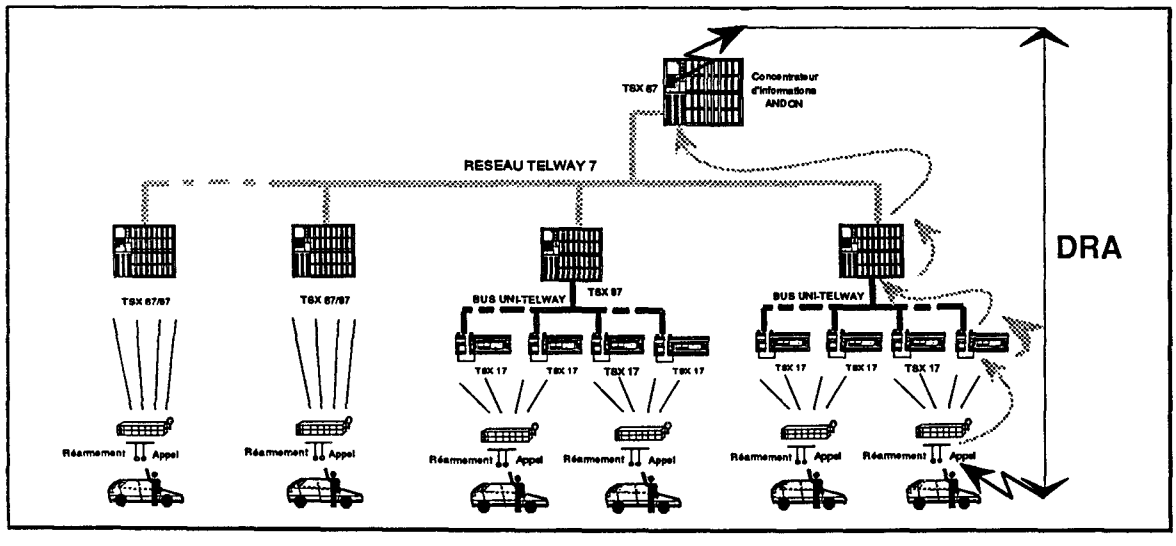


Figure III.42 : *Echanges de messages liés au paramètre DRA (Délai Remontée Appel)*

IV.2. Manière de conduire l'étude

IV.2.1. Principes

La modélisation d'un réseau d'automates TSX 7 connectés à un réseau Telway 7 avec des coupleurs de communication MPT 10 a été évoquée dans le chapitre II. Avec le modèle de réseau Uni-Telway qui a été présenté, l'étude du système complet pourrait se faire en assemblant les graphes réseaux de Petri de chaque équipement pour obtenir un modèle global de toute l'architecture.

Cependant cette approche impose de construire un réseau de Petri de grande taille (plus de 100 places). L'architecture comporte en effet 25 automates et autant de coupleurs de communication. Les temps de simulation risquent alors d'être importants et les risques d'erreurs nombreux.

Il nous paraît préférable d'adopter une autre démarche. Dans une première phase chaque réseau Uni-Telway est simulé de façon isolée et durant ces simulations, les informations nécessaires par la suite sur le fonctionnement de l'automate ANDON sont recueillies dans des fichiers. Dans une seconde phase, le réseau Telway 7 est simulé de manière autonome en considérant les résultats collectés dans les premières simulations comme des événements de l'environnement.

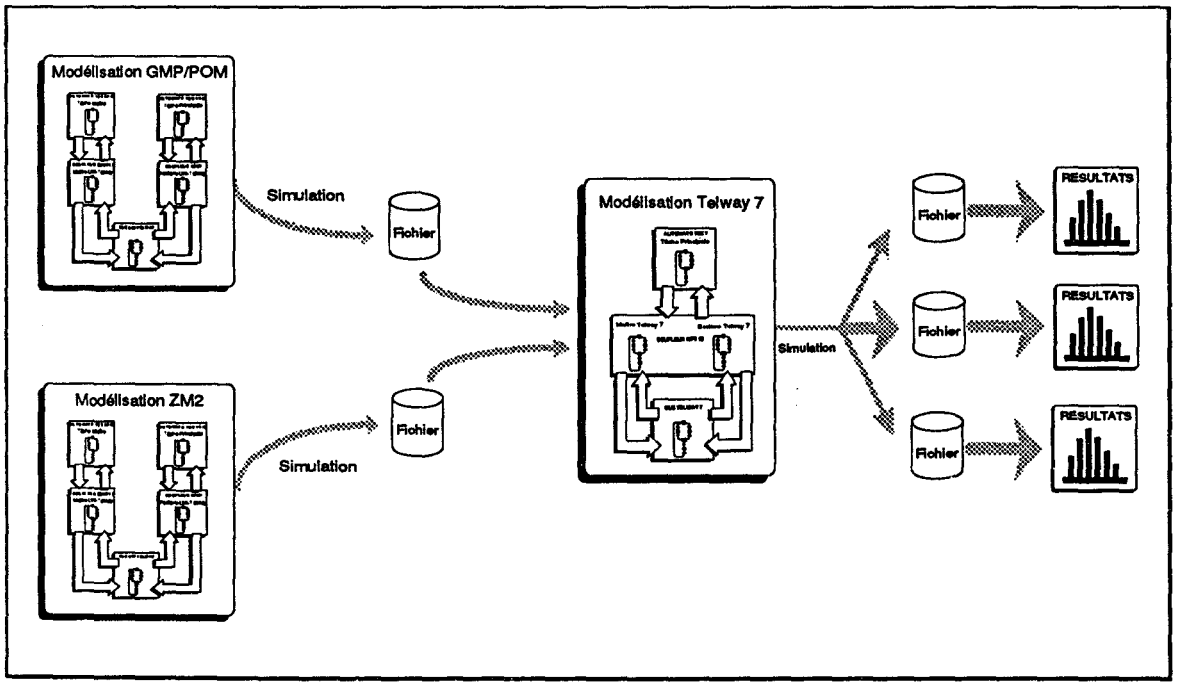


Figure III.43 : *Principes pour simuler l'architecture complète*

Cette approche est réalisable pour l'application que nous avons à traiter. Elle ne l'est cependant pas dans tous les cas, notamment lorsque des stations connectées sur des réseaux différents se synchronisent mutuellement en échangeant des messages par l'intermédiaire d'équipements "passerelle". Dans notre exemple, les communications se font des TSX 17-20 vers l'automate concentrateur, mais pas dans le sens inverse.

IV.2.2. Simulations de ZM2 et GMP/POM

Il s'agit du modèle construit dans le paragraphe II de ce chapitre. Une simulation doit être réalisée pour la zone ZM2 et une autre pour GMP/POM. Au cours de ces simulations, il est nécessaire de stocker dans des fichiers l'instant de prise en compte par la tâche maître de l'automate TSX 87 ANDON des différents messages en provenance des TSX 17-20, avec les informations véhiculées (Cf. figure III.13). Pour pouvoir mesurer le paramètre DRA, avec chaque message de type 1 (apparition d'un appel) doit figurer l'instant absolu où est survenu l'appel.

IV.2.3. Simulation du réseau Telway 7 de l'atelier ANDON

Les descriptions des applicatifs des huit automates ANDON et du concentrateur sont la traduction en procédures Pascal des séquencements d'échanges présentés sur les figures III.40 et III.41. La modélisation de l'environnement doit prendre en compte, d'une part les informations d'appels, d'arrêts, de réarmements et de décalage ligne pour les six automates ANDON des zones rénovées, et d'autre part, les messages en provenance des réseaux Uni-Telway pour les deux TSX des nouvelles zones.

Les résultats sur le paramètre DRA sont obtenus par différence entre l'instant où un message d'appel est pris en compte par la tâche maître de l'automate concentrateur et le moment où est survenu l'appel, cette valeur étant stockées dans les fichiers résultants des simulations réalisées préalablement.

IV.2.4. Conclusion

Nous avons présenté dans ce dernier paragraphe, une méthode pour étudier les performances d'une architecture de taille déjà importante. Il nous paraît essentiel, avant d'aborder la modélisation d'un système complexe, de toujours chercher, lorsque cela est possible, à le décomposer pour faire des simulations séparées de chaque sous-ensemble.

CONCLUSION GENERALE

Dans ce mémoire, nous avons proposé une approche de modélisation d'architecture de commande et de pilotage d'atelier de fabrication, visant à évaluer par simulation les performances des systèmes de communication mis en oeuvre à ce niveau. L'intérêt principal est de permettre la validation des performances ainsi que l'organisation des échanges de données impliquées par les choix de conception.

Si la simulation est employée avec succès dans divers domaines de l'industrie (flux de production, CAO/robotique etc.), il s'agit d'une approche relativement nouvelle pour les systèmes de commande d'installations automatisées dans l'industrie manufacturière. C'est pourquoi il s'est avéré nécessaire, dans une première étape, de définir de quelle manière aborder la modélisation des échanges d'informations d'une architecture de commande et de pilotage. A partir de l'étude d'applications industrielles du groupe PSA Peugeot Citroën, nous avons mis en évidence trois niveaux de modélisation :

1 - modélisation du fonctionnement propre des équipements participant à l'architecture : il s'agit de représenter les caractéristiques du fonctionnement interne des équipements (système d'exploitation, protocole de communication etc.), qui ont une incidence significative sur les performances des échanges de données au niveau de l'architecture. Pour chaque matériel, le modèle est réalisé une seule fois, de manière à pouvoir être employé de façon générique dans chaque système où intervient ce type d'équipement.

2 - modélisation du comportement des applicatifs implantés sur les équipements : pour une architecture donnée, on isole, dans le programme de contrôle/commande à implanter sur l'équipement, la partie qui traite de la communication. La modélisation associée doit générer les échanges séquencés de messages au niveau de l'architecture.

3 - image de l'environnement : l'objectif est d'interfacer le modèle de l'architecture (matériels et applicatifs) avec un modèle de l'environnement de manière à produire des scénarios de charge proches de la réalité. Il nous paraît en effet indispensable de disposer de modèles suffisamment précis pour que les résultats de simulation soient fiables.

A partir de ces spécifications, nous avons choisi un formalisme de modélisation qui permet de construire et de simuler les modèles souhaités. Les réseaux de Petri à structures de données se sont révélés être appropriés pour cette étude, car ils permettent, avec un formalisme unique de représenter l'architecture de commande dans sa globalité. Chaque équipement est modélisé sous la forme d'un graphe de processus communiquant, qui décrit les états successifs que peut prendre le matériel. Ce modèle s'interface d'une part avec un modèle simplifié du procédé et d'autre part avec une description des applicatifs.

Pour nos travaux, nous avons utilisé l'outil SEDRIC. Il permet de construire et de simuler les modèles grâce à une implémentation du formalisme des réseaux de Petri à structures de données; l'interprétation est traduite à l'aide de procédures écrites en langage Pascal. Nous avons modélisé plusieurs équipements de commande (automates programmables et coupleurs de communication) de la société Télémécanique.

Ces modèles isolés ont été utilisés pour élaborer des modèles complets d'architecture de commande du groupe PSA Peugeot Citroën. La comparaison entre les résultats de simulation et les mesures réalisées sur site ont permis de démontrer la faisabilité de la démarche proposée et d'en saisir l'intérêt du point de vue validation.

Il reste maintenant à rendre l'approche plus industrielle. En effet, il est difficile d'envisager de proposer, aux automaticiens, l'utilisation de l'outil SEDRIC tel quel. Pour leur permettre de mettre en oeuvre une telle démarche, il est nécessaire de développer un outil adapté au contexte d'application, permettant de décrire rapidement l'architecture de commande et de pilotage qu'ils souhaitent valider. Il convient en effet de mettre à la disposition des concepteurs une bibliothèque de modèles génériques d'équipements utilisés chez PSA Peugeot Citroën.

Ce travail de thèse constitue un point de départ, pour un projet de plus grande ampleur, dont l'objet serait de définir, puis de développer un outil répondant aux spécifications. Cette voie ouvre de nombreuses perspectives dans la continuité de l'étude validée par notre travail, notamment :

- l'extension des modèles disponibles à une plus large gamme d'équipements,
- l'intégration des modèles au sein d'une bibliothèque structurée,
- la définition et le développement d'un outil d'aide à la validation par simulation.

BIBLIOGRAPHIE

[AFN 91] Projet de norme AFNOR
"Représentation des systèmes de contrôle et de commande des systèmes automatisés - Contribution à l'intégration des outils XAO du Génie Automatique : modèle conceptuel Base-PTA"

[ALA 88] P. Alanche
"Les automatismes et leur CAO"
Editions Hermès, collection Technologies de Pointe, Paris 1988

[AND 92] C. André, M.A. Peraldi
"Grafcet et langages synchrones"
GRAFCET'92, Paris mars 1992

[AUM 88] M. Aumiaux, G. Rodde
"Automatiser la production"
Editions Masson, Paris 1988

[AYA 85] J.M. Ayache, J.P. Courtiat, M. Diaz, G. juanole
"Utilisation des réseaux de Petri pour la modélisation et la validation de protocoles"
T.S.I. volume 4, N°1, 1985, pp. 51-71

[BAR 88] J. Barnes
"Programmer en ADA"
InterEditions, Paris 1988

[BEA 90] F. Beau
"Identification des contraintes et scénarios d'échange dans une application manufacturière validée par une simulation"
Les exigences du temps réel sur les réseaux locaux de communication
Journée SEE, Grenoble juin 1990

[BEA 91] J. Beauquier, B. Bérard
"Systèmes d'exploitation - concepts et algorithmes"
Editions McGraw Hill, Paris 1991

[BEL 86] L. Bellenger
"Les nouveaux défis de l'automobile"
Chotard et associés éditeurs, Paris 1986

[BEU 88] G. Beuchot, R. Josserand
"Les réseaux locaux industriels"
Editions Hermès, collection Technologies de Pointe, Paris 1988

[BIN 90] C. Binot
"Les benchmarks pour évaluer les performances des systèmes informatiques"
Minis & Micros N° 334, février 1990

[BON 87] R. Bonetto
"Les ateliers flexibles de production"
Editions Hermès, Paris 1987

[BOR 89] Borland
"Turbo-Pascal - Programmation orientée objet"
Documentation Borland 1989

[BOR 90] P. Borrel
"Méthodes et outils pour la conception et la réalisation de systèmes temps réel"
Le temps réel - Colloque AFCET, Nantes octobre 1990

[BOS 88] J.C. Bossy
"Les automates programmables"
Editions Hermès, collection Technologies de Pointe, Paris 1988

[BOU 88] J.P. Bourey
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Docteur de l'Université de Lille 1988

[BOU 91] J.P. Bourey, J.C. Gentina
"Conception structurée et modulaire d'architecture de contrôle répartie en production flexible manufacturière"
APII, volume 25, N°4, 1991, pp. 349-376

[BRA 83] G.W. Brams
"Réseaux de Petri : théorie et pratique"
Editions Masson, Paris 1983

[BUR 89] D. Bursky
"Simulator eases communication network design"
Electronic Design, October 1989

[CAC 90] CACI
"LANNET II.5 - User's Manuel"
CACI Products Company, 1990

[CER 88] A. Cernault
"La simulation des systèmes de production"
Cepadues éditions, Paris 1988

[CHA 90] D. Chabbert
"Simuler pour investir à coup sur"
Techniques & équipements de production N°10, juin 1990

[COI 89] P. Coiffet
"La productique et ses outils"
Editions Hermès, collection Technologies de Pointe, Paris 1989

[COR 91] P. Corvisier, G. Dupuy, S. Fdida, K.L. Thai
"La modélisation des réseaux de communication industriels des centrales à EDF-GDF"
Nouvelles Architectures pour les communications, Paris octobre 1991

[CRA 89] E. Craye
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Docteur de l'Université de Lille, 1989

[CRU 91] D. Cruette, J.P. Bourey, J.C. Gentina
"Method of structural and functional analysis of complex processes, and aided design of command under constraints in the manufacturing"
Mathematical and intelligent models in system simulation, IMACS 1991

[DAV 89] R. David, H. Alla
"Du Grafcet aux réseaux de Petri"
Editions Hermès, Paris 1989

[DIG 87] Digital
"Introduction to VAXELN"
Documentation Digital, 1987

[DIG 89] Digital
"Using VAXset"
Documentation Digital, AA-HB16E-TE 1989

[DIM 90] M. Di Mascolo
"Modélisation et évaluation de performances des systèmes de production gérés en Kanban"
Thèse de Docteur de l'INPG, Grenoble 1990

[DOM 92] E. Dombre, A. Fournier, J.F. Armand, F. Beauchaints, V. Carré, D. Gehors
"Méthodologie de programmation des robots-manipulateurs à partir de systèmes CAO"
APII, volume 36, n° 1, 1992, pp. 49-58

[ELL 86] J.P. Elloy, O. Roux
"Electre : a language for control structuring in real time"
The Computer Journal, Vol. 29, N°3,1986, pp.229-234

[EVE 91] P. Eveillard
"Etude et réalisation d'un système de suivi de qualité de production
Application à l'utilisation des réseaux locaux dans l'industrie automobile"
Mémoire CNAM, Rennes décembre 1991

[FDI 89] S. Fdida, G. Pujolle
"Modèles de systèmes et de réseaux"
Tome 1 : Performance - Tome 2 : Files d'attente
Editions Eyrolles, Paris 1989

[GEL 82] E. Gelende, G. Pujolle
"Introduction aux réseaux de files d'attente"
Editions Eyrolle, Paris 1982

[GEN 87] H.J. Genrich
"Predicate/transition nets"
Lectures notes in computer science, Springer Verlag 1987

[GRZ 91] F. Grzesiak
"Les enjeux techniques et financiers de la simulation de partie opérative dans un projet d'automatisme"
Nouveaux outils méthodologiques de conception des systèmes automatisés
AUTOMATION 91, Paris mai 1991, pp.157-195

[HAL 91] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud
"The synchronous dataflow programming language LUSTRE"
Proceedings of the IEEE, 79 (9), september 1991

[HAT 88] Derek J. Hatley, Imtiaz A. Pirbhai
"Strategies for real-time system specification"
Dorset House Publishing, New York 1988

[HER 89] P. Hermel
"Qualité et management stratégiques - Du mythique au réel"
Editions d'Organisation, Paris 1989

[HOL 85] D. Hollinger
"Utilisation pratique des réseaux de Petri dans la conception des systèmes de production"
T.S.I. volume 4, n°6, 1985, pp.509-522

[IBM 91] IBM Formation
"Noyau et programmation du système AIX"
Cours de formation IBM, 1991

[IGL 89] IGL Technologies
"SADT Un langage pour communiquer"
Editions Eyrolles, 1989

[IGL 90] IGL Technologies
"Software Trough Pictures"
User's manuels, 1990

[JEN 86] K. Jensen
"Coloured Petri nets"
Lectures Notes in Computer Science, Springer-Verlag 1986

[JUA 90] G. Juanole
"Modèles pour l'analyse qualitative et/ou quantitative d'architectures de communication temps réel"
Les exigences du temps réel sur les réseaux locaux de communication
Journéee SEE, Grenoble juin 1990

[KOR 85] Y. Korem
"Robotique pour ingénieurs"
Editions McGraw-Hill, Paris 1986

[KRO 91] S. Krowiak
"Automatisation et conduite d'un atelier de tôlerie"
La pratique du C.I.M. ou la mise en oeuvre de l'usine communicante
AUTOMATION 91, Paris juin 1990, pp. 53-72

[LEB 80] P. Le Beux
"Introduction au Pascal"
Editions Sybex, 1980

[LEP 91] F. Lepage, F. Afilal, P. Antoine, E. Bajic, J.Y. Bron, T. Divoux
"Les réseaux locaux industriels"
Editions Hermès, Paris 1991

[LES 91] J.J. Lesage
"Les outils de modélisation pour le cycle de vie des systèmes automatisés de production"
Génie automatique et production industrielle, ISMCM Saint-Ouen, mars 1991

[LOU 90] J.L. Loubet
"Automobiles PEUGEOT Une réussite industrielle 1945 - 1974"
Editions Economa, Paris 1990

[MAR 87] J. Martinez, P. Muro, M. Silva
"Modelling, validation and software implementation of production systems using high level Petri-nets"
IEEE, International conference on robotics and automation
Raleigh (USA), 1987, pp. 1180-1185

[MAT 91] P. Mathieu
"Les systèmes automatisés de production dans l'industrie manufacturière"
Génie automatique et production industrielle
Journées de synthèse, ISMCM Saint-Ouen mars 1991

[MIL 90] R.I. Mills
"Simultaneous engineering - the role of process modelling"
The international review of advanced manufacturing technology and management, pp. 60-63

[NAT 80] S. Natkin
"Les réseaux de Petri stochastiques et leur application à l'évaluation des systèmes informatiques"
Thèse de doctorat 3ème cycle, CNAM 1980

[PAN 91] H. Panetto, F. Corbier, P. Lhoste
"Structuration et réutilisation en conception des machines et systèmes automatisés de production manufacturière"
Nouveaux outils méthodologiques de conception des systèmes automatisés
AUTOMATION 91, Paris mai 1991, pp.125-141

[PIL 92] D. Pilaud
"Conception et validation de logiciel zéro défaut en automatisme"
Réseaux de terrain - Sureté des automatismes
AUTOMATION 92, Paris 1992, pp. 153-160

[PUJ 86] G. Pujolle
"L'évaluation de performance des systèmes informatiques"
AFCET/Interfaces N°42, avril 1986, pp.13-17

[REU 89] C. Reutenauer
"Aspects mathématiques des réseaux de Petri"
Etudes et recherches en informatique (ERI)
Editions Masson, Paris 1989

[REZ 90] C. Reze, P. Rioux, F. beaufils
"ELSIR un formalisme réseau de Petri adapté à la modéliastion de systèmes"
Les exigences du temps réel sur les réseaux locaux de communication
Journéee SEE, Grenoble juin 1990

[RIA 88] J.C. Riat
"Réalisation d'une serveur d'équipements industriels sous protocole Uni-Telway"
D.E.A. de productique, Université de Lille, septembre 1988

[RIA 91a] J.C. Riat
"Performance a priori d'une architecture de commande d'atelier"
Nouveaux outils méthodologiques de conception des systèmes automatisés
AUTOMATION 91, Paris mai 1991, pp.15-31

[RIA 91b] J.C. Riat
"La récursivité démystifiée..."
DEBUG Magazine, N°14, juin 1991

[RIA 92] J.C. Riat
"Evaluating the performance of a workshop command architecture by simulation"
Third International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Lyon France march 1992, pp. 129-146

[ROC 89] A. Roche, M. Jubin
"Les cellules et îlots flexibles d'usinage"
Editions Hermès, collection Technologies de Pointe, Paris 1989

[RUD 86] M. Rudniansky
"Architecture de réseaux : le modèle OSI rôle et fonctionnalités"
Editions Editest, Paris 1986

[SAU 84] J. Sauvy
"L'industrie automobile"
Editions Que sais-je ? 1984

[SCH 91] O. Schmid, J.C. Michaud
"Utilisation de K-SYS chez Elf Aquitaine"
Congrès MESUCORA 91, session N°5, pp.25-32

[SES 91] Scientific and Engineering Software
"SES/Workbench Introductory Overview"
Documentation SES, release 2.0, 1991

[SHA 90] K.S. Shanmugan, W. LaRue
"Modeling and simulation of communication networks using the Block Oriented Network Simulator (BONeS)"
System design and networking conference, Santa Clara, May 1990

[SHI 83] S. Shingo
"Maîtrise de la production et méthode Kanban"
Editions d'Organisation, Paris 1983

[SIB 85] C. Sibertin-Blanc
"High level Petri nets with data structure"
6th European Workshop on Petri net and applications, Espoo (Finland) 1985

[SIB 87] C. Sibertin-Blanc
"Les réseaux de Petri à objet comme langage de programmation récursif"
Bigre + Globule 59, Avril 1988

[SIE 86] Siemens
"Processeur de communication CP 535 avec logiciel de programmation COM 535"
Manuel de service, C79000-B8577-C320-02 1986

[SIE 89] Siemens
"Automate programmable S5-135U (CPU 928)"
Manuel, 6ES5 998-1UL32 1989

[SOU 88] C. Sourisse
"Les automatismes industriels"
Editions Hermès, collection Technologies de Pointe, Paris 1988

[TEC 89] Techlog
"SEDRIC manuel de référence"
Documentation Techlog, 1989

[TEL 87] Télémécanique
"Spécifications Uni-Telway"
Document Télémécanique 1987

[TEL 88a] Télémécanique
"Bus UNI-TELWAY - Manuel de référence"
Documentation Télémécanique, TSX D24 004F 1988

[TEL 88b] Télémécanique
"TSX SCM 21.6 voie 1 - Bus UNI-TELWAY"
Documentation Télémécanique, TSX D24 005 F 1988

[TEL 88c] Télémécanique
"Interface TSX 17 ACC5 - Bus UNI-TELWAY Micro-automate TSX 17-20"
Documentation Télémécanqiue, TSX D24 006F 1988

[TEL 88d] Télémécanique
"Langage PL7-2 Synthèse"
Documentation Télémécanique, TSX D12 002F 1988

[TEL 89] Télémécanique
"Réseau inter-automates TELWAY 7"
Documentation Télémécanique, TSX D41704 1989

[TEL 90a] Télémécanique
"X-TEL Guide méthodologique de l'atelier logiciel"
Documentation Télémécanique, TSX DG XTEL V4F 1990

[TEL 90b] Télémécanique
"X-TEL atelier logiciel"
Documentation Télémécanique, TXT DM XTEL V4F 1990

[TEL 90c] Télémécanique
"Langages PL7-3 Manuel de référence V4"
Documentation Télémécanique, TXT DR PL7 3 V4F

[VAL 80] R. Valette, M. Courvoisier
"Systèmes de commande en temps réel"
Editions SCM, Paris 1980

[VAL 85] R. Valette, V. Thomas, S. Bachman
"Sedric, un simulateur à événements discrets basé sur les réseaux de Petri interprétés et colorés"
Conference on manufacturing Systems, INRIA Paris 1985

[VER 89a] C. Verlinde
"Contribution à l'étude des architectures de systèmes automatisés"
Thèse de Docteur de l'INPL, Nancy 1989

[VER 89b] Verilog
"RdP Aide à la conception et l'évaluation de systèmes par réseaux de Petri"
Notice technique 1989

[VER 91] L. Verdin
"De la spécification à l'exploitation : le 'cycle de vie' des SAP"
Génie automatique et production industrielle
Journées de synthèse, ISMCM Saint-Ouen mars 1991

[VID 86] G. Vidal-Naquet
"Les réseaux de Petri et leurs applications"
AFCET/Interfaces N°43, mai 1986, pp. 5-13

[WAL 90] J.B. Waldner
"CIM les nouvelles perspectives de la production"
Editions Dunod, Paris 1990

[WOL 91] J. Wolgensinger
"André Citroën"
Editions Flammarion 1991

[WOM 90] J.S Womack, D. Jones, D. Roos
"The machine that changed the world"
Rawson Associates 1990

[YVA 90] P.A. Yvars
"Poste de travail du roboticien pour l'étude, la simulation et la reconfiguration des lignes de production flexibles dans l'industrie manufacturière"
Thèse de Doctorat, Lille 1990

ANNEXE 1

Formalismes des réseaux de Petri
et description de l'outil SEDRIC

I. RESEAUX DE PETRI SIMPLES

I.1. Définition d'un réseau de Petri [BRA 83] [DAV 89]

Un réseau de Petri (RdP) est un quadruplet R = < P, T, Pré, Post > où :

. P est un ensemble fini et non vide de places; on note m = Card (P),

. T est un ensemble fini et non vide de transitions; on note n = Card (T),

. Pré : P x T --> N est la fonction d'incidence avant qui définit le poids de l'arc orienté reliant une place p de P à une transition t de T (ce poids est nul lorsque l'arc n'existe pas),

. Post : P x T --> N est la fonction d'incidence arrière qui définit le poids de l'arc orienté reliant une transition t de T à une place p de P (ce poids est nul lorsque l'arc n'existe pas)

Si Pré (p, t) > 0 on dit que p est une place d'entrée de t.
Si Post (p, t) > 0 on dit que p est une place de sortie de t.

I.2. Définition d'un réseau de Petri marqué

Un réseau de Petri marqué est un couple N = < R, M > où :

. R est un réseau de Petri,

. M : P --> N est une fonction marquage du RdP; M (p) est le marquage de la place p, on dit aussi le nombre de marques (ou de jetons) contenues dans p.

I.3. Règles d'évolution du marquage d'un réseau de Petri

Les règles d'évolution, qui permettent de modifier le marquage d'un RdP par le franchissement de transition, traduisent la dynamique du système modélisé. Une transition t est franchissable pour un marquage M si et seulement si :

$$\forall p \in P,\ M(p) \geq Pré(p, t)$$

On note M (t > cette relation dans N^n x T : c'est la précondition de franchissement de t.

Si t est franchissable pour M, le franchissement de t permet d'obtenir le nouveau marquage M' tel que :

$\forall p \in P,\ M'(p) = M(p) - Pré(p, t) + Post(p, t)$

On note $M\ (\ t\ >\ M'$ la relation correspondante dans $N^n \times T \times N^m$ et on dit que M donne M' par le franchissement de t.

Remarque :

Si t est franchissable, son franchissement devient une opération possible mais non obligatoire; le marquage d'un RdP peut rendre plusieurs transitions franchissables simultanément et dans ce cas, une seule transition sera franchie à la fois.

II. RdP A STRUCTURE DE DONNEES

II.1. Notations

Un réseaux de Petri à structures de données est constitué d'un RdP et d'un structure de données associée. Pour la définir, il est nécessaire de définir les concepts suivants :

- **PROPRIETE** : attribut relatif à un objet. Une propriété est définie par son nom et un domaine de valeurs, qui peut être soit un ensemble de constantes prédéfinies, soit l'ensemble des entités d'une classe d'entité. On appelle valeur d'une propriété un élément de son domaine.

- **CLASSE D'ENTITE** : liste de propriétés distinctes.

- **ENTITE** : instance d'une classe d'entité. Une entité est définie par son nom, sa classe d'entité et la valeur qui est associée à chacune des propriétés de sa classe. Pour désigner la valeur d'une propriété d'une entité, on utilise la notation pointée : <nom d'entité> . <nom de propriété>.

Pour un ensemble E, on notera :

. E^* l'ensemble des suites finies aussi appelées n-uplets de E :

$$E^* = \{ <e_1, e_2 \ldots e_i \ldots e_n, n \in N, e_i \in E \}$$

. 1 la suite vide, c'est-à-dire l'élément de E^* pour lequel $n = 0$,

. P (E) l'ensemble des parties de E^*.

II.2. Définition formelle d'un RdPSD [SIB 85]

Un réseau de Petri à structure de données (en abrégé RdPSD) est défini par les neufs éléments suivants :

1 - un ensemble P de places;

2 - un ensemble T de transitions;

3 - un ensemble CE de classes d'entités, qui sont celles des entités du marquage du réseau;

4 - un ensemble V de variables qui permettent de spécifier les entités déplacées par le franchissement d'une transition,

5 - une fonction cl, qui définit la classe des variables et des places :

$cl : V \rightarrow CE$ et $cl : P \rightarrow CE^*$

La classe d'une variable détermine la classe d'entité à laquelle appartiennent les entités qui peuvent lui être substituées. La classe d'une place détermine la classe des entités des n-uplets qu'elle peut contenir.

6 - une fonction d'incidence avant : $Pré : T \times P \rightarrow P(V^*)$, qui respecte la classe des variables et des places : $v^* \in Pré(t,p) \Rightarrow cl(v^*) = cl(p)$.

Si $Pré(t,p) \neq 0$, on dira que p est une place d'entrée de t. Si une variable v figure dans l'un des Pré (t,p) on dira que v est une variable d'entrée de t.

7 - A chaque transition t, on peut associer une expression booléenne, dont les seules variables libres sont ses variables d'entrées; cette expression, est appelée la précondition de t.

8 - une fonction d'incidence arrière : $Post : T \times P \rightarrow P(V^*)$ qui respecte la classe des variables et des places : $v^* \in Post(t,p) \Rightarrow cl(v^*) = cl(p)$

Si $Post(t,p) \neq 0$, on dira que p est une place de sortie de t. Si une variable v figure dans l'un des Post (t,p) on dira que v est une variable de sortie de t.

9 - A chaque transition t on peut associer une action, c'est-à-dire un ensemble d'affectations de la forme : $x.nprop \leftarrow f(y_1, y_2 \ldots y_n)$

où : x est une variable de sortie de t,
nprop est une propriété des entités de cl (x),
$y_1, y_2 \cdots y_n$ sont des variables d'entrée de t,
f est une fonction quelconque.

II.3. Marquage d'un RdPSD

Un état d'un réseau de Petri à structures de données, que l'on appelle marquage, est défini par la donnée de :

1 - une fonction de répartition : $m : P \rightarrow P(E^*)$, qui indique pour chaque place les n-uplets d'entités qu'elle contient; ceux-ci doivent vérifier la classe de chaque place, c'est-à-dire en notant cl (e) la classe d'une entité e :

$$\forall e^* = \langle e_1 \ldots e_n \rangle \in m(p),\ \langle cl(e_1) \ldots cl(e_n) \rangle = cl(p)$$

2 - la valeur des entités figurant dans les places; deux marquages ayant la même fonction de répartition des entités, mais avec des valeurs différentes sont considérés comme distincts.

II.4. Règles d'évolution d'un RdPSD

Pour qu'une transition t soit armée pour un marquage donné, il faut que chacune de ses variables d'entrée v puisse capturer une entité S (v) du marquage de telle sorte que :

$$\forall p,\ S(Pré(t,p)) < m(p)$$

La fonction S, qui détermine à quelle entité correspond chaque variable est appelée une substitution.

Pour qu'une transition t soit franchissable, il faut d'une part qu'elle soit armée pour une substitution S, et que d'autre part les entités capturées par S satisfassent sa précondition.

Le franchissement d'une transition t franchissable par une substitution S s'effectue alors en quatre temps :

1 - dans chaque place d'entrée de t, on retire un n-uplet d'entités pour chaque n-uplet de variables de Pré (t,p);

2 - pour chaque variable de sortie qui n'est pas une variable d'entrée, on crée une nouvelle entité;

3 - l'action de t est appliquée aux entités capturées par les variables;

4 - dans chaque place de sortie de t, on dépose un n-uplet d'entités pour chaque n-uplet de variable de Post (t,p).

III. DESCRIPTION DE L'OUTIL SEDRIC

III.1. Formalisme réseaux de Petri implémenté dans SEDRIC

Afin de bien comprendre les modèles qui ont été construits avec l'outil SEDRIC, dans le cadre de ce travail de thèse, nous proposons de détailler le formalisme proposé par cet outil. Nous allons présenter les caractéristiques attachées aux cinq éléments de base d'un modèle réseaux de Petri, à savoir le jeton, la place, la transition l'arc et les procédures pour l'interprétation du modèle.

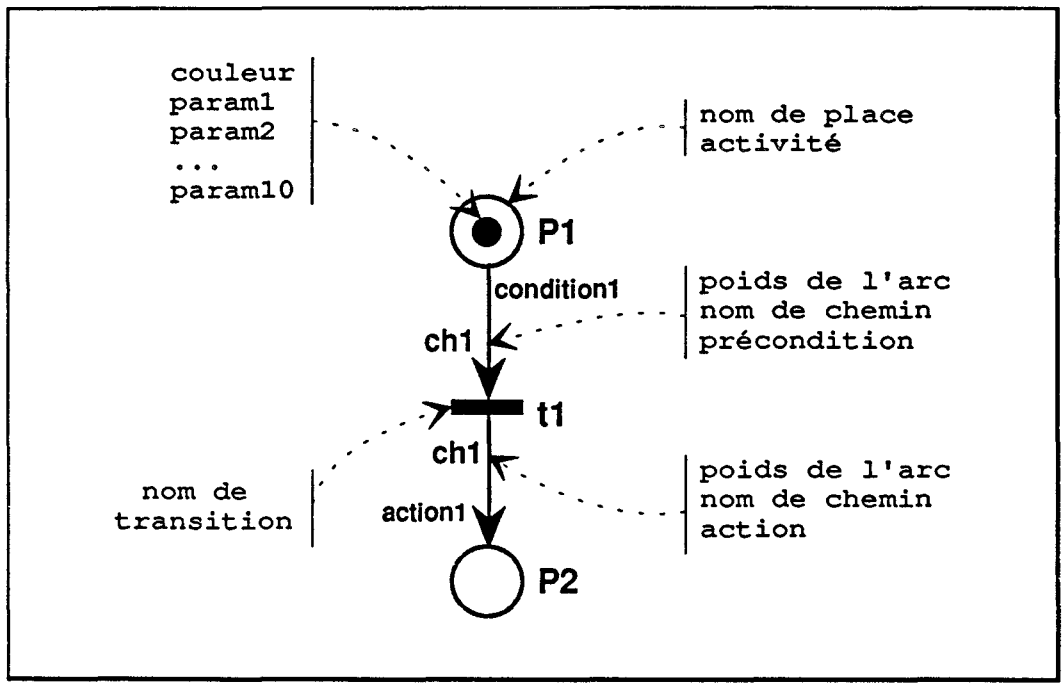


Figure A1.1 : *éléments attachés à un modèle réseaux de Petri avec SEDRIC*

Jeton : SEDRIC permet de traiter deux types de jetons : des jetons non-colorés, qui sont en fait de simples "token" et des jetons colorés. Chaque jeton coloré est caractérisé par une couleur et une liste de dix paramètres réels. Lors de la création du marquage initial associé à un modèle, on définit l'ensemble des couleurs, ainsi que les valeurs des attributs de chaque jeton.

Place : On distingue deux types de places : les places colorées qui ne peuvent contenir que des jetons colorés et les places non-colorées pour les jetons simples. Une place est caractérisée par son nom; deux places d'un même modèle ne peuvent pas être homonymes. Pour temporiser les places on leur

associe une activité qui fixe la durée de temporisation. Celle-ci peut-être constante ou stochastique (différentes lois aléatoires sont disponibles). Il est important de remarquer qu'aucune couleur n'est associée aux places. Des jetons colorés de tous types peuvent donc marquer n'importe quelle place colorée.

Transition : Chaque transition est caractérisée par son nom, qui doit être unique. Lors d'une simulation, lorsque plusieurs transitions sont tirables au sens du marquage, l'ordre des tentatives de franchissement (évaluation des préconditions associées à chaque transition) correspond à celui alphabétique des noms de transition. Lors d'un conflit, c'est donc la première transition dans l'ordre alphabétique, dont les préconditions sont vraies qui est tirée.

Arc : Le poids d'un arc permet de définir le nombre de jetons de la place correspondante qui sont impliqués dans le tir de la transition. Des noms de chemin (par exemple "ch1" sur la figure A1.1) sont employés pour spécifier le cheminement des jetons lors du tir de transition, ce qui est indispensable avec des jetons individualisés. En général, pour chaque transition, les arcs entrant sont appariés avec les arcs sortant en utilisant des noms de chemin identiques de telle sorte qu'il n'existe aucune ambiguïté sur les places destination des jetons impliqués dans le franchissement.

Des procédures sont associées aux arcs et constituent l'interprétation du formalisme réseaux de Petri. Des conditions sur les arcs amont, activées lorsque la transition est franchissable au sens du marquage, permettent de conditionner le tir aux attributs (couleurs et paramètres) des jetons de la place. Il est ainsi possible de sélectionner les jetons qui participent au franchissement. Des actions, activées lorsqu'une transition est effectivement tirée, permettent de modifier les attributs des jetons impliqués dans le tir. L'ordre dans lequel sont évaluées les conditions liées à une transition correspond à l'ordre alphabétique des noms de chemin des arcs associés. La même règle s'applique pour l'ordre d'exécution des actions après le tir d'une transition.

Procédure : On distingue deux catégories de procédures. Des procédures standard sont disponibles avec le formalisme. Au cas où les fonctionnalités offertes par celles-ci seraient insuffisantes, l'utilisateur peut se définir ses propres routines. Ecrites en langage Pascal, elles permettent d'accéder directement à la structure de données du modèle, pour en accroître les possibilités d'expression.

Les principales procédures standard sont détaillées dans le tableau ci-dessous :

Procédures standard : CONDITION	
jprio (numéro)	sélection du plus vieux jeton coloré dont le paramètre 'numéro' a la plus grande valeur
njprio (numéro)	sélection du plus vieux jeton coloré dont le paramètre 'numéro' a la plus petite valeur
jcfi (nom de couleur)	sélection du plus vieux jeton coloré de la couleur spécifiée
njcfi (nom de couleur)	sélection du plus vieux jeton coloré de couleur différente de celle spécifiée
jpi (numéro, valeur)	sélection du plus vieux jeton coloré dont le paramètre 'numéro' est égal à 'valeur'
njpi (numéro, valeur)	sélection du plus vieux jeton coloré dont le paramètre 'numéro' est différent de 'valeur'
pvjc	sélection du plus vieux jeton coloré (fonctionnement en FIFO)
npvjc	sélection du jeton coloré le plus récent (fonctionnement en LIFO)
Procédures standard : ACTION	
dujc	duplication d'un jeton coloré
kijc	destruction d'un jeton coloré
crjc (nom de couleur)	création d'un jeton coloré de la couleur spécifiée
acj (nom de couleur)	modification de la couleur d'un jeton coloré avec la couleur spécifiée

Tableau A1.1 : *principales procédures standard*

Le code Pascal de procédures utilisateur peut faire appel aux procédures standard ainsi qu'à des routines spécifiques, qui permettent d'exploiter la structure de données du modèle. Le tableau A1.2 présentent ces principales routines complémentaires :

Routines pour les procédures UTILISATEUR	
convp (nom de place)	fonction de conversion de nom de place en numéro interne utilisé par le simulateur
convt (nom de transition)	fonction de conversion de nom de transition en numéro interne utilisé par le simulateur
convc (nom de couleur)	fonction de conversion de nom de couleur en numéro interne utilisé par le simulateur
memejeton	utilisée pour mettre en oeuvre des choix multicritères sur des jetons
valjeton	récupération des caractéristiques du jeton coloré courant
apj (numéro)	validation de modifications sur le paramètre 'numéro' du jeton courant
activ (durée)	modification de la durée d'activité de la place aval

Tableau A1.2 : *principales routines disponibles pour les procédures utilisateur*

III.2. Exploitation de l'outil

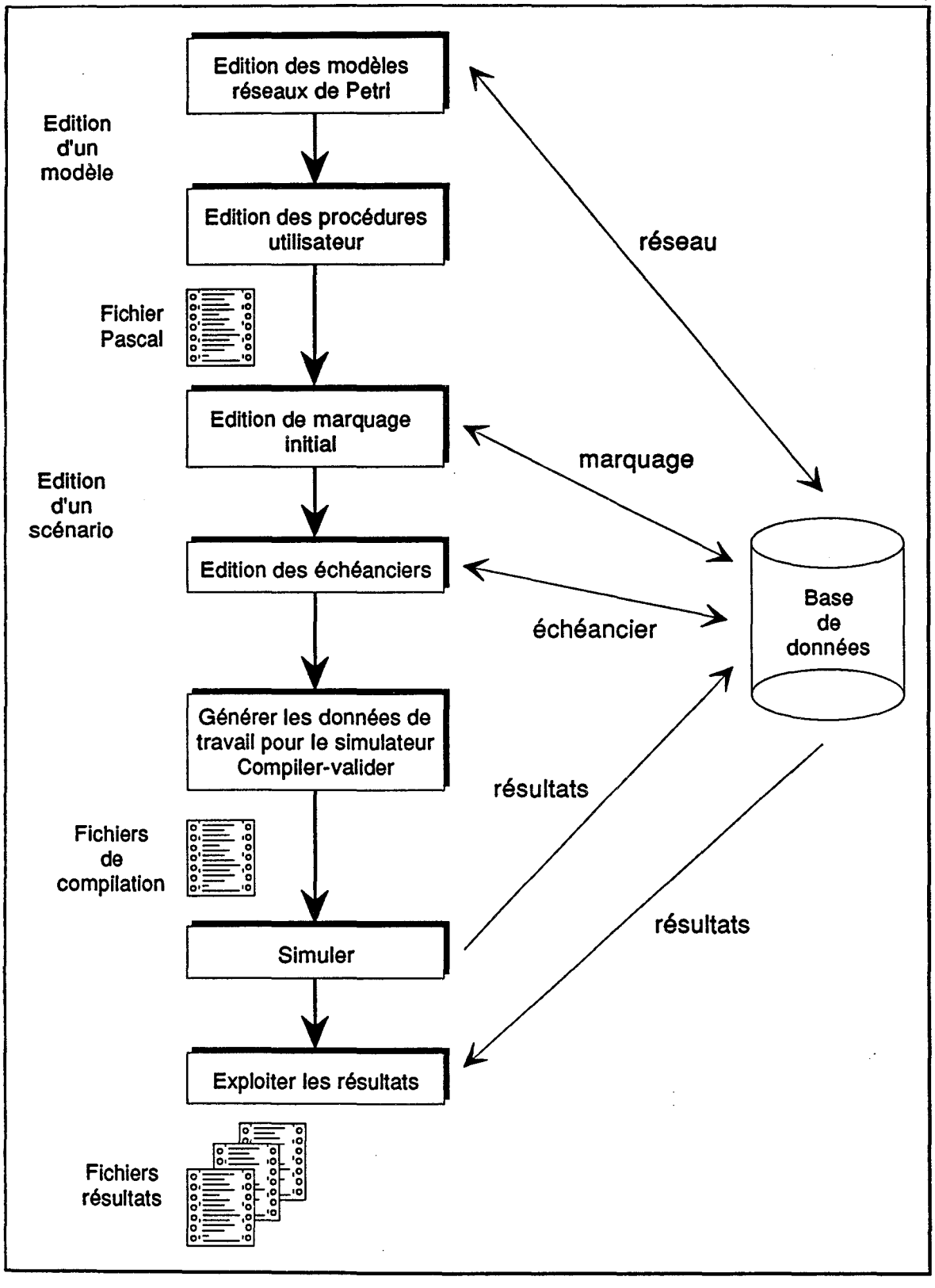


Figure A1.2 : *cheminement de construction et d'exploitation d'un modèle avec SEDRIC*

A l'aide d'un éditeur graphique, on construit le modèle réseau de Petri. Lorsque des procédures utilisateur sont utilisées, le code associé est déclaré au moyen d'un éditeur de texte. Dans l'éditeur de marquage, sont définis les identificateurs des couleurs, ainsi que le marquage initial des places du réseau. L'éditeur d'échéancier permet de déclarer les événements qui vont rythmer la simulation (tirs de transition correspondant à des événements extérieurs au système modélisé par exemple).

Avant de pouvoir exécuter une simulation, SEDRIC vérifie lors d'une phase de compilation certaines cohérences au niveau du modèle complet (procédure utilisateur utilisée mais non définie par exemple). Durant cette phase, les procédures Pascal utilisateur sont compilées et le scénario (marquage initial et échéancier) est traduit pour pouvoir être utilisé par le simulateur.

Lorsque la compilation est réussie, le modèle peut être simulé. Chaque simulation peut être réalisée en mode interactif (l'utilisateur a la main après chaque événement) ou en mode automatique. La simulation génère en sortie des résultats qui sont stockés sur disque en vue d'une exploitation ultérieure.

III.3. Fonctionnement du simulateur

Le fonctionnement du simulateur de SEDRIC est résumé sur la figure suivante :

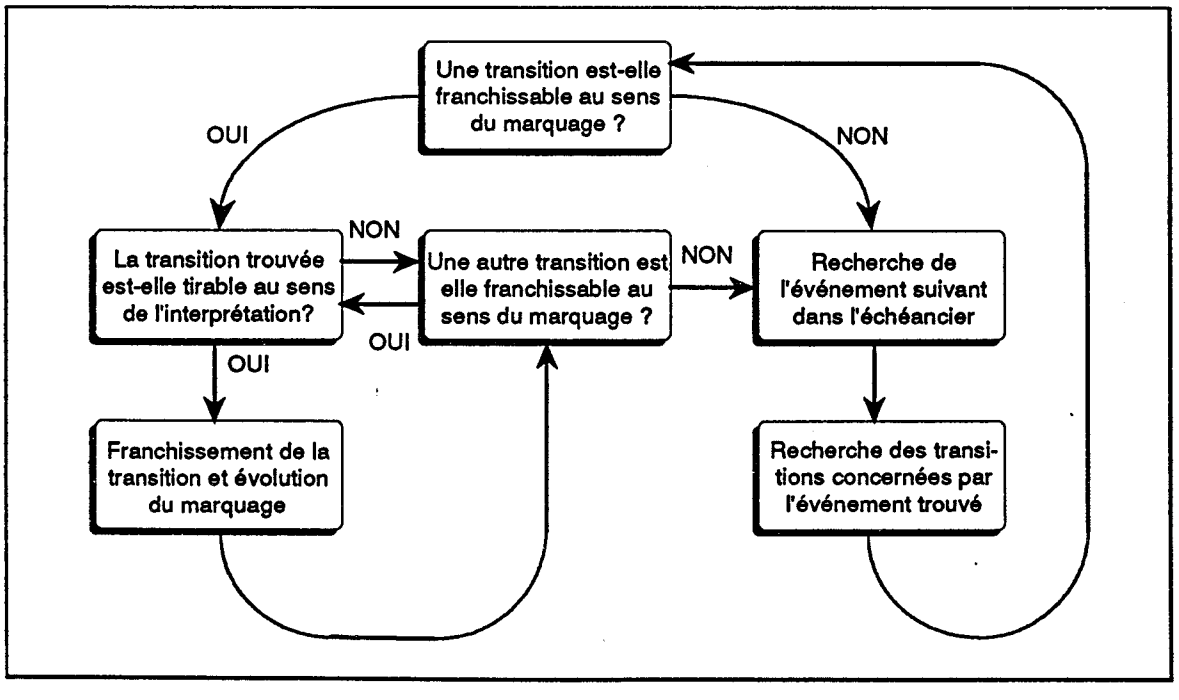


Figure A1.3 : *principe de fonctionnement du joueur*

On appelle joueur de réseau de Petri l'algorithme qui permet de passer d'un marquage à un autre dans la logique d'évolution du modèle. Dans un réseau de Petri interprété, une transition est franchie lorsque les deux conditions suivantes sont simultanément vérifiées :

- elle est franchissable au sens du marquage du graphe réseau de Petri,
- les conditions associées aux arcs amont de la transition sont vérifiées.

Les événements de simulation sont gérés par un échéancier. L'intérêt d'une telle approche est de pouvoir "sauter" des intervalles de temps lorsqu'on sait que rien ne se passe. Comme le montre la figure A1.3, lorsqu'on arrive à un état stable, c'est-à-dire quand aucune transition n'est franchissable, on prend en compte directement l'événement suivant de l'échéancier. La durée effective de simulation n'est alors plus liée au temps réel de fonctionnement simulé, mais dépend du nombre d'événements pris en compte lors de la simulation.

ANNEXE 2

Etude de l'architecture de commande de l'atelier des mélanges de La-Barre-Thomas

Etude de l'architecture de commande de l'atelier des mélanges de La-Barre-Thomas

I. ATELIER DES MELANGES

I.1. Produit fabriqué : le caoutchouc

L'usine de Rennes/La-Barre-Thomas produit les éléments à base de caoutchouc qui sont montés sur les véhicules du groupe PSA : joints de porte, joints de pare-brise etc. Dans l'atelier des mélanges sont fabriqués les différents types de caoutchoucs qui servent de matière première pour ces éléments. Stocké dans des bacs, le caoutchouc brut doit "vieillir" quelques jours, dans le magasin des mélanges terminés, avant de pouvoir être utilisé dans l'atelier de production.

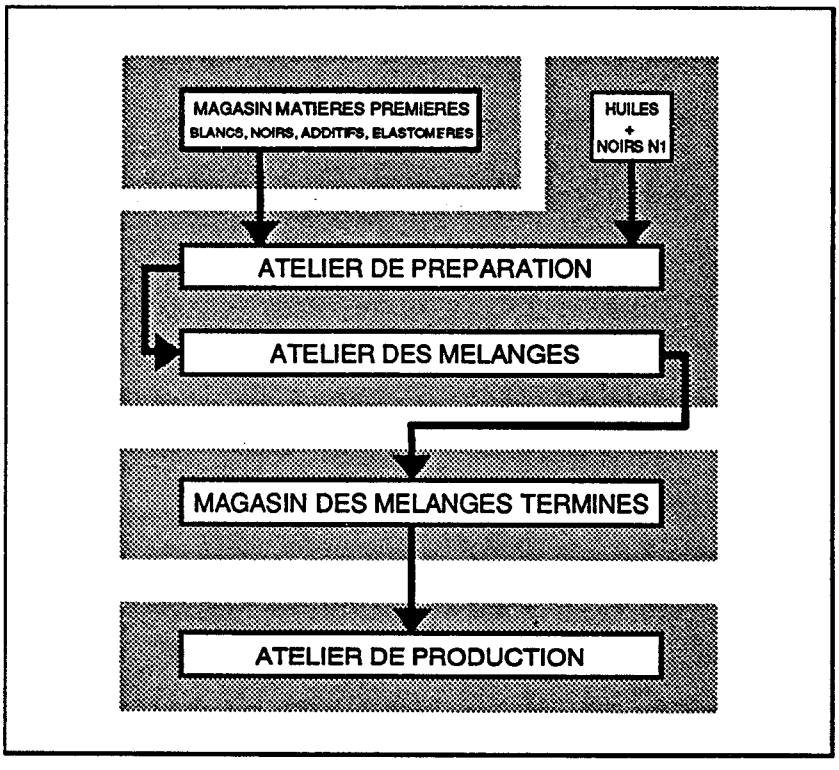


Figure A2.1 : *Organisation de l'usine de Rennes/La-Barre-Thomas*

Pour confectionner du caoutchouc brut, il faut doser et malaxer dans des mélangeurs des constituants de base , qui appartiennent aux cinq types suivants :

- charges blanches : 9 types - 4400 kg/Jour,
- charges noires : 8 types - 22800 kg/Jour,
- plastifiants ou huiles : 12 types - 11200 kg/Jour,
- additifs : 81 types (répartis en 5 familles) - 6100 kg/Jour,
- élastomères : 61 types - 28500 kg/Jour.

Les dosages sont réalisés par pesées des produits, avant introduction dans les mélangeurs.

L'atelier des mélanges se caractérise par les éléments suivants :

- plus de 300 formules de caoutchouc à gérer,
- une durée moyenne du cycle de mélange de 6 minutes,
- un nombre moyen de 14 composants par formule introduits en 4 phases,
- la réalisation quotidienne d'environ 30 charges (73000 kg).

I.2. Organisation de l'atelier

L'atelier des mélanges se compose de :

- 1 installation de distribution des blancs,
- 1 installation de distribution des noirs,
- 1 installation de distribution des huiles,
- 1 installation de distribution des élastomères,
- 4 installations de distribution des additifs (une par famille d'additifs),
- 3 mélangeurs internes (MI) A, B et C,
- 3 mélangeurs de réception,
- 6 lignes de finition comportant chacune un mélangeur externe (ME), un mélangeur de conformation et une loveuse,

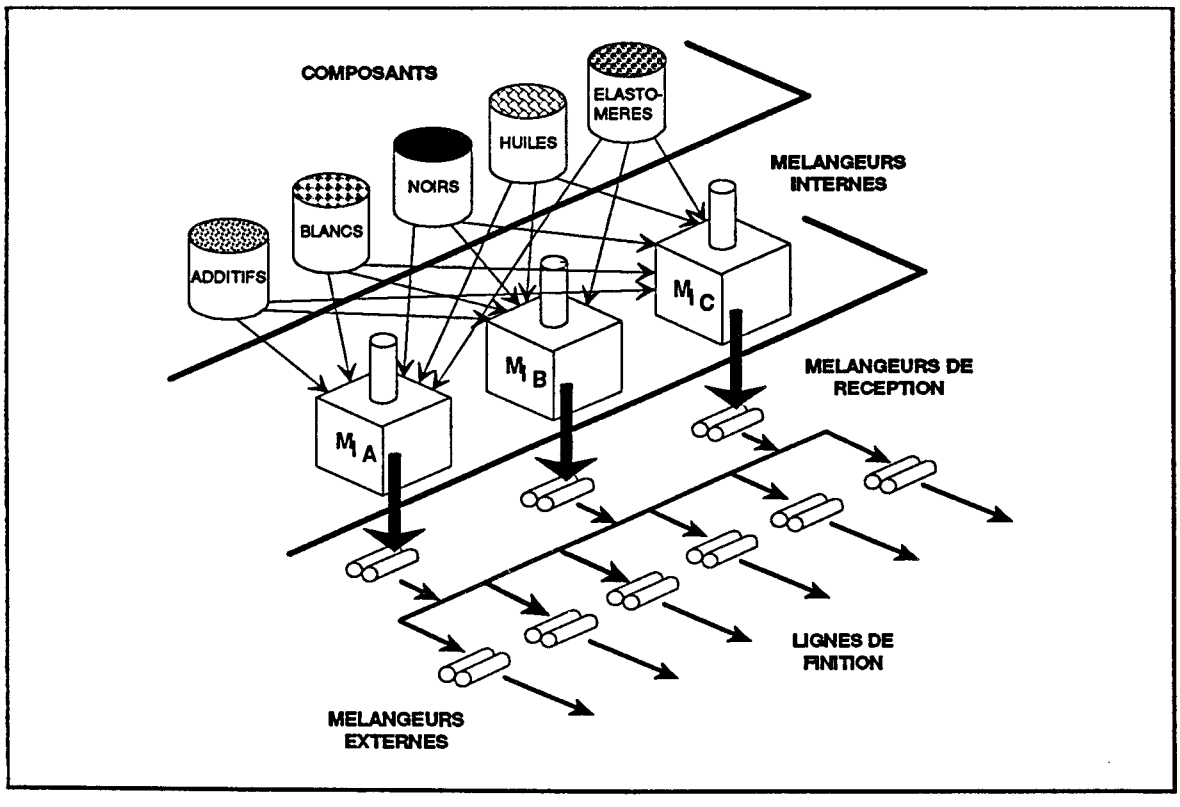


Figure A2.2 : *Organisation de l'atelier des mélanges de Rennes/La-Barre-Thomas*

I.3. Objectifs de l'automatisation

L'automatisation de l'atelier des mélanges a pour buts :

- d'alimenter les mélangeurs internes pour assurer un travail en 3x8 par semaine,
- d'assurer le transport, les pesées et l'introduction automatique des constituants des mélanges caoutchouc,
- de contrôler et de conduire les mélangeurs internes et externes,
- de gérer l'ensemble de l'atelier (réalisation automatique des programmes de travail),
- d'améliorer la régularité des charges,
- de permettre une meilleure vision de la qualité.

Cette automatisation doit se faire:

- de façon modulaire pour une mise en place progressive,
- en s'intégrant dans l'installation pour ne pas modifier l'implantation des mélangeurs.

Afin de faciliter l'intégration de l'automatisation dans l'installation existante, il a été choisi de mettre en place les systèmes de commande et de pilotage de manière progressive. Pour le projet quatre tranches successives ont été identifiées :

1ère tranche : automatisation des huiles, des blancs, des noirs
et du mélangeur interne A,

2ème tranche : automatisation du mélangeur interne C,
et supervision par Monitor 77,

3ème tranche : automatisation du mélangeur interne B,
des 3 mélangeurs externes A, B et C,
des additifs 2, des additifs 3/5 et des additifs 4,

4ème tranche : automatisation des élastomères (non retenue à ce jour).

II. ARCHITECTURE DE COMMANDE

II.1. Besoins fonctionnels

A partir des objectifs définis pour l'automatisation de l'atelier des mélanges, les fonctions à supporter par l'architecture de commande et de pilotage ont été détaillées :

1 - Gestion des données :

Il s'agit de préparer et de gérer des données techniques nécessaires pour la fabrication (description des nomenclatures, gammes et paramètres de commande et de contrôle).

2 - Gestion des approvisionnements :

Seul l'approvisionnement des noirs de type N1 et des huiles pour lesquels le process commence dès la porte de l'usine génère une interface avec le NGP (système de gestion usine). Les autres constituants consommés par l'atelier des mélanges sont appelés à partir d'un magasin de matières premières, géré par un système informatique existant, qui lui-même gère ses interfaces avec le NGP. L'approvisionnement de l'atelier des mélanges étant réalisé en kanban, il n'existe aucun lien entre le système informatique de l'atelier et celui du magasin.

3 - Gestion des mélanges terminés :

La gestion du magasin se fait par un système kanban [SHI 83]. Organisée autour d'un système informatique, elle doit permettre une mise à jour de la disponibilité des charges entrées au magasin, en fonction des résultats de contrôle décisionnel. Une liaison avec le NGP est nécessaire pour transmettre les mouvements comptables de déclaration de production de l'atelier des mélanges.

4 - Contrôle qualité :

Des tests sont réalisés à différentes étapes de la fabrication :

Mesures sur le processus liées au produit : effectuées par les automatismes des mélangeurs, elles sont transmises au système informatique par le compte-rendu de cadencement (exemple : température de mélangeur, vitesse mélangeur etc.).

Contrôles fabrication : au dernier poste de chaque ligne de finition, l'opérateur effectue un contrôle fabrication sur un rhéomètre qui transmet les mesures au système informatique.

Contrôles laboratoire : effectués de manière fréquentielle, ils sont réalisés sur des machines de laboratoire, toutes connectées au système informatique.

5 - Ordonnancement :

La procédure d'ordonnancement est effectuée à la demande du responsable de production, au moins une fois par équipe, en tenant compte des réalisations de l'équipe précédente (priorité au "recyclage" des ordres non effectués), et avec pour horizon les trois équipes suivantes. Elle reçoit en entrée un portefeuille non ordonnancé d'ordres d'exécution pour chacun des trois mélangeurs internes et génère une liste d'ordres de production.

6 - Cadencement :

Il consiste en une transmission des listes ordonnancées de production vers les automatismes des mélangeurs internes. Cette distribution doit être effectuée de telle sorte que ces automatismes aient toujours en mémoire la partie de la liste correspondant aux quelques heures suivantes, pour pallier les éventuelles défaillances du cadencement. Un compte-rendu est fourni par les automatismes pour chaque charge réalisée. Celui-ci véhicule les mesures effectuées sur le processus pendant les phases de mélange, ainsi que les pesées réelles effectuées.

7 - Commande du processus de fabrication :

L'automatisme de commande de chaque mélangeur interne exécute les ordres de fabrication du cadencement. La charge est réalisée selon les opérations successives définies dans la gamme. Chaque opération se décompose en trois étapes :

- mise en condition du mélangeur interne suivant les paramètres de conditionnement lus dans la gamme,
- introduction des produits selon les temps donnés dans la gamme,
- mélange des constituants.

A la fin de la dernière opération, un message signale à l'opérateur du mélangeur de réception que "la charge est prête pour évacuation". L'opérateur donne son accord dès que le mélangeur de réception est libre. La réception de la charge sur ce dernier provoque l'affichage de l'état de disponibilité des lignes de finition. Après choix d'une de ces lignes par l'opérateur, la charge est transférée sur le mélangeur externe où sont ajoutés des additifs. Une fois passée sur le mélangeur conformateur de bande, la charge est rangée en caisses pour être stockée en magasin.

L'introduction progressive des différents composants dans le cycle de production implique qu'ils soient délivrés à temps et dans les bonnes quantités par les installations de distribution. Les pesées sont donc effectuées en parallèle à la production, et les produits sont temporairement stockés dans des trémies tampon pour être déversés au bon moment dans les mélangeurs.

8 - Commande des installations de distribution :

Ces installations doivent permettre l'acheminement des composants vers les systèmes de pesée. Pour les produits solides (élastomères et certains additifs), le convoyage s'effectue sur un tapis roulant muni d'un système de pesée. Les chargements sur ce tapis sont manuels.

Les constituants sous forme de poudre ou de granulés (blancs, noirs et certains additifs) sont stockés dans des silos. Le transport se fait sous pression dans des tuyaux. Les pesées sont réalisées automatiquement avec des trémies peseuses situées en amont des trémies tampons.

Les huiles, qui sont sous forme liquide, sont emmagasinées dans des cuves et circulent dans des canalisations. Comme pour les poudres, les pesées se font de manière automatique avec des trémies peseuses.

Pour les produits non solides, le système doit assurer la gestion du remplissage des cuves/silos, ainsi qu'une surveillance des alarmes, défauts. Pour les huiles, une circulation permanente dans les canalisations, à des températures données, doit également être assurée.

9 - Exploitation des résultats de production :

Elle consiste à réaliser diverses statistiques (consommations réelles des produits, répartition de la production par formule etc.), et des traitements sur les données qualité.

10 - Assistance à la conduite de l'installation :

Il s'agit de mettre en oeuvre des procédures de démarrage et d'arrêt de l'atelier, ainsi que de gestion des modes de marche des moyens.

11 - Assistance à la maintenance locale et centralisée :

Elle consiste à traiter les alertes en provenance des automatismes, en offrant aux opérateurs une assistance au diagnostic au dépannage, et à la remise en cycle. Des statistiques d'incidents et de fonctionnement sont établies en liaison avec le système de suivi de l'activité maintenance (SAM).

II.2. Architecture adoptée

Suite à l'étude fonctionnelle qui vient d'être détaillée, nous pouvons distinguer quatre catégories de fonctions à implanter:

1 - des applications de gestion centrale déjà existantes sur ordinateur IBM :
- . gestion des nomenclatures et gammes NGP,
- . suivi des activités maintenance SAM.

2 - un ensemble de transactions, permettant les créations, mises à jour et exploitation des données de base liées au pilotage en temps réel de l'atelier, ainsi que la gestion des résultats. Le choix a été fait de regrouper toutes ces fonctionnalités sur un calculateur à tolérance de panne de marque Tandem,

3 - des applications de commande directe du procédé, implantées sur des automates programmables industriels Télémécanique TSX série 7,

4 - des applications basées sur des superviseurs d'automates : la supervision centrale est implantée sur un compatible PC avec le progiciel Monitor 77, et la supervision locale sur des PC industriels de type FICAM.

Nous obtenons alors une architecture de commande sur quatre niveaux, comme l'illustre la figure suivante :

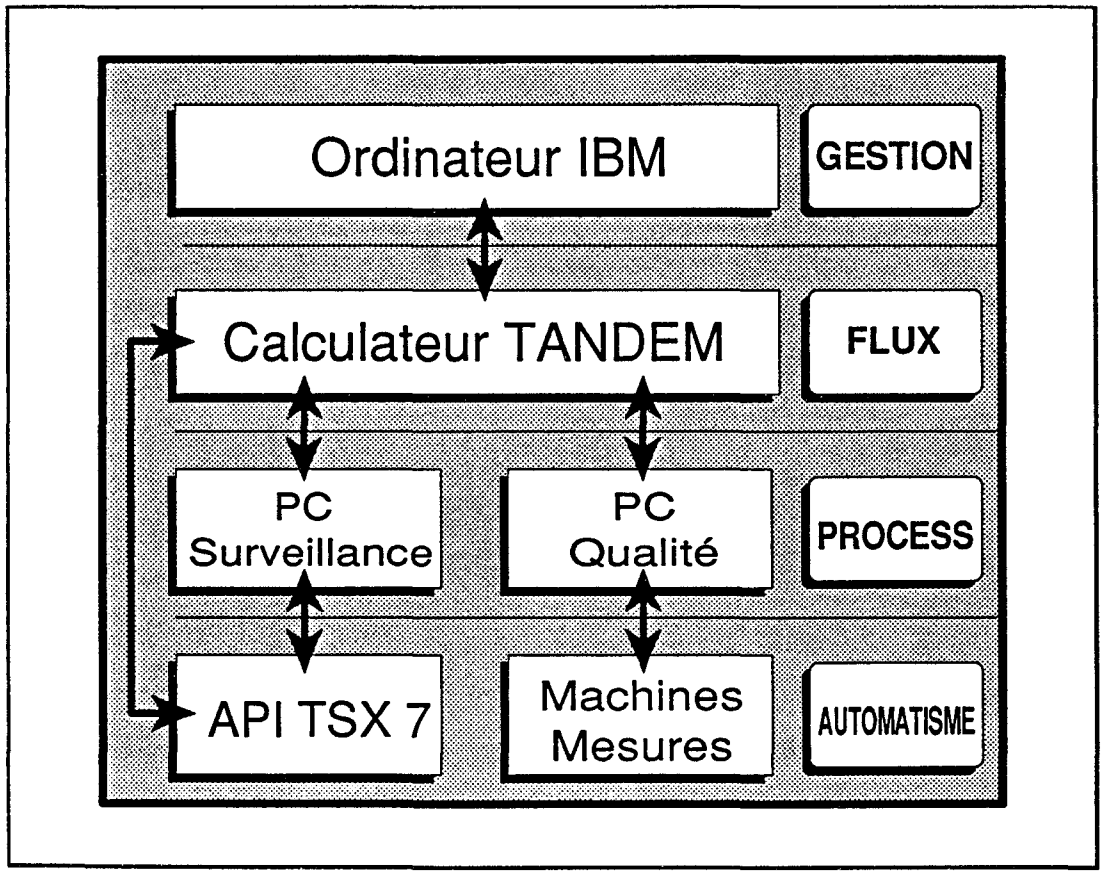


Figure A2.3 : *Organisation de l'architecture de commande et de pilotage de l'atelier*

La répartition des fonctions sur cette architecture est détaillée dans le tableau présenté en page suivante :

	Tandem	Automates	Supervision	Qualité
Interface NGP - Gestion données	X			
Approvisionnement Huiles + Noirs N1	X	X		
Gestion des mélanges terminés	X			
Contrôle qualité	X			X
Ordonnancement	X			
Cadencement	X	X		
Commande du processus de fabrication		X		
Commande des installations de distribution		X		
Exploitation des résultats de production	X	X		
Assistance à la conduite		X	X	
Assistance à la maintenance		X	X	

Tableau A2.1 : *Répartition des fonctions sur l'architecture de commande et de pilotage*

III. MODELISATION DES AUTOMATISMES

Dans l'architecture présentée dans le paragraphe précédent, nous avons isolé la partie commande du procédé qui est la plus contrainte du point de vue des temps de réponse. C'est ce sous-ensemble que nous avons modélisé et simulé pour en vérifier les performances. Dans une première étape, nous avons élaboré un modèle de l'architecture matérielle étudiée. Celui-ci a ensuite été complété par une description des applicatifs de chaque équipement. Le modèle complet a alors été simulé pour obtenir les résultats souhaités sur le fonctionnement et les performances de cette architecture de commande.

III.1. Architecture matérielle

L'automatisation de façon progressive de l'atelier des mélanges a nécessité un découpage de l'atelier en trois parties:

- DISTRIBUTION :
  . elle concerne l'acheminement des différents produits aux mélangeurs internes demandeurs
  . chaque famille de produits constitue un sous-ensemble (noirs, blancs, huiles, additifs)

- FABRICATION :
  . elle s'occupe de réaliser le programme de fabrication grâce aux 3 fonctions :
    - d'organisation et de gestion des procédés de fabrication (gammes),
    - de pesage des produits,
    - de pilotage des mélangeurs.
  . chaque mélangeur interne constitue un sous-ensemble

- FINITION :
  . elle s'occupe de synchroniser les mélangeurs internes, les comatiques de réception et les mélangeurs externes, ainsi que les conformateurs de bande
  . deux mélangeurs externes associés à un mélangeur interne constituent un sous-ensemble

Un automate programmable est affecté à la commande de chaque sous-ensemble. L'architecture de commande complète des automatismes est présentée sur la figure A2.4 :

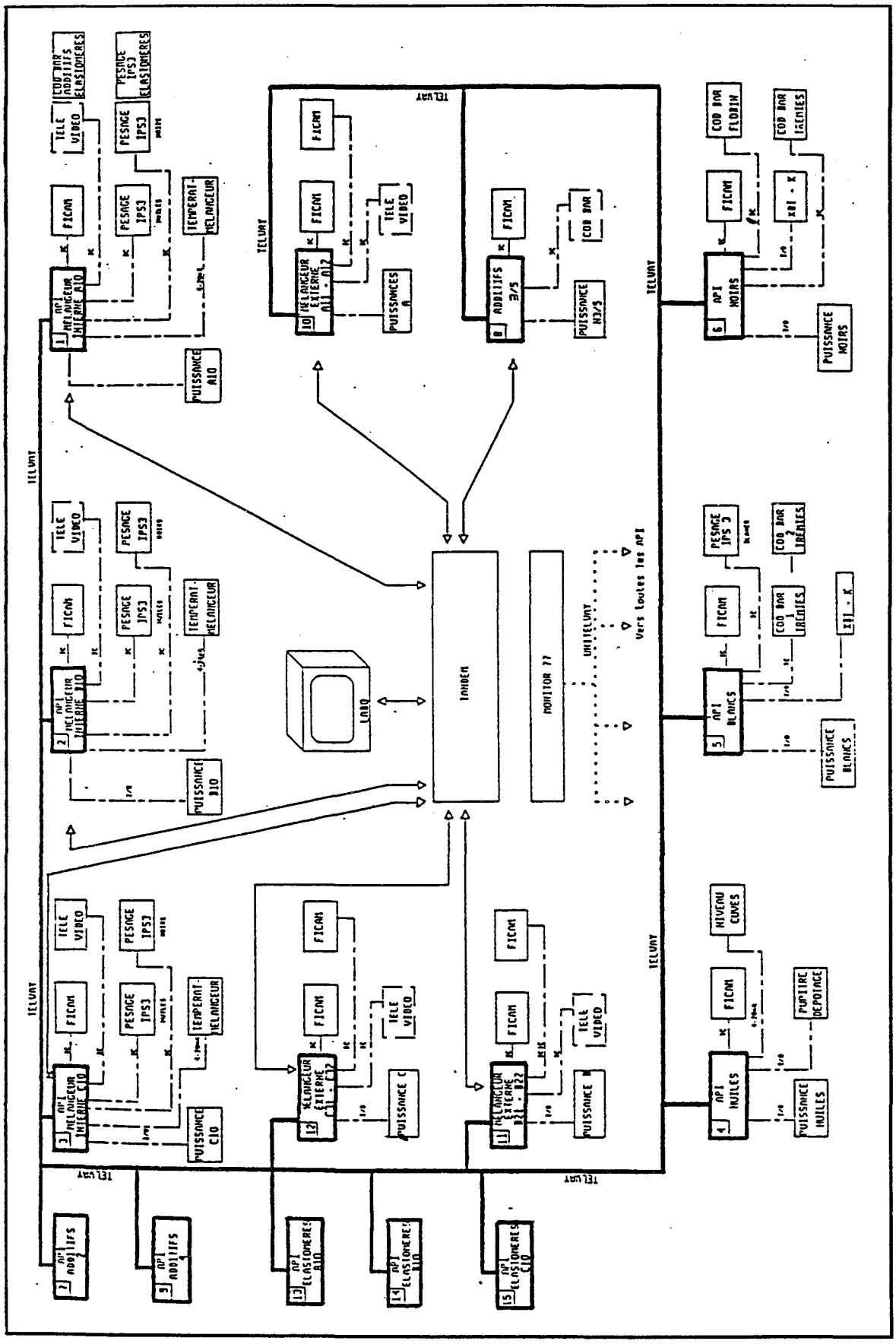


Figure A2.4 : *Architecture des automatismes de l'atelier des mélanges*

La synchronisation entre les automates de l'atelier est totalement réalisée par les mots COM et les échanges de messages sur le réseau Telway 7 :

- entre un automate MI ou ME et les automates de distribution pour l'apport des composants liés aux phases de mélange,
- entre un automate MI et l'automate de finition correspondant à la ligne choisie par l'opérateur pour transférer l'identité de la charge évacuée sur le mélangeur de réception.

Les communications entre les automates mélangeurs et le Tandem sont réalisées avec des liaisons point à point sous protocole Modbus. Les échanges permettent de:

- télécharger les gammes (hors production),
- télécharger les codes composants (hors production),
- télécharger les ordres de production (cadencement),
- remonter les alarmes vers le Tandem,
- remonter les état d'avancement de la charge en cours vers le Tandem,
- remonter les consommations de constituants vers le Tandem.

Pour les fonctions de supervision centralisée sur le Monitor 77, les échanges de données sont faits sur un réseau Uni-Telway. L'assistance à la maintenance locale est réalisée sur PC FICAM (un par automate) relié par une liaison point à point sous protocole Modbus.

Après avoir choisi l'architecture matérielle, le concepteur doit organiser les échanges de données sur les réseaux. Ses choix sont conditionnés par les besoins fonctionnels en communication entre les sous-ensembles de commande ainsi que par les possibilités de transmission et de services des réseaux utilisés. Par exemple, dans l'application étudiée, les automates de distribution sont connectés au réseaux Telway 7 mais ne sont pas reliés au Tandem. Pourtant ils ont des données à échanger avec ce dernier (alarmes, état d'avancement etc.). Il est donc nécessaire d'utiliser, au niveau de l'organisation des flux de données, un des automates de distribution comme passerelle pour assurer ces échanges.

Lorsque le séquencement des échanges est défini, il n'existe aucun moyen, ni pour le valider "logiquement" (détection de blocage etc.), ni pour en connaître a priori les performances. Il s'agit pourtant d'une caractéristique importante du système, car il n'est pas admissible que les échanges ralentissent le process. Pour notre application, il est par exemple utile d'avoir une idées du délai d'acheminement d'une demande de composant émise par un MI à l'attention d'un automate de distribution. C'est sur ce point que notre démarche de modélisation/simulation doit apporter une assistance significative au concepteur.

III.2. Modélisation de l'architecture matérielle

Lorsque cette étude a été réalisée, les tranches 1 et 2 étaient installées. Elles concernent les mélangeurs internes A et C, ainsi que les installations de distribution des blancs, des noirs et des huiles. Nos travaux ont été tout d'abord centrés sur cette architecture, pour ensuite faire des extrapolations sur les extensions des tranches 3 et 4.

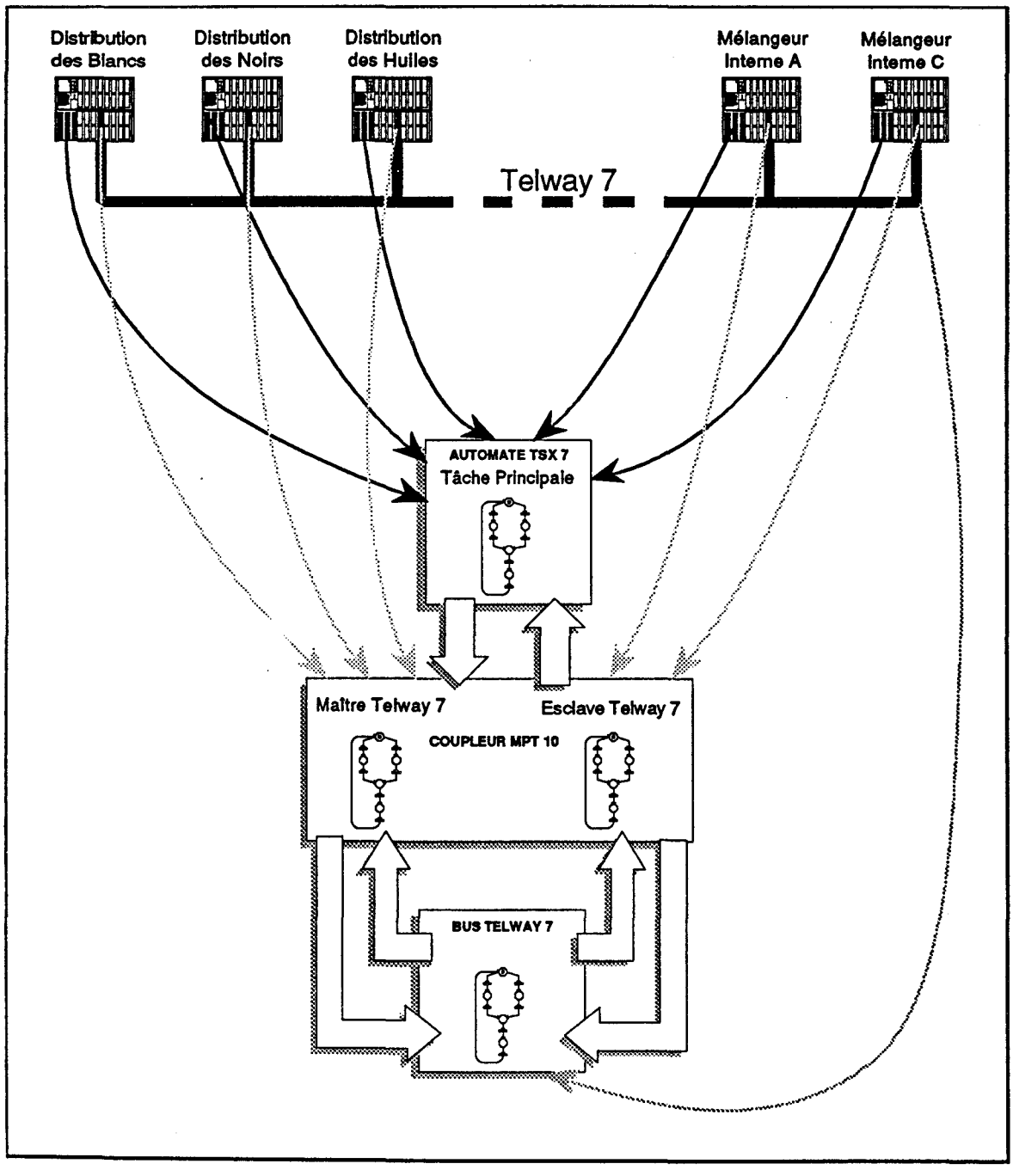


Figure A2.5 : *Construction du modèle de l'architecture matérielle*

L'architecture est organisée autour d'un réseau Telway 7 sur lequel sont connectés tous les automates TSX 7 équipés de coupleurs de communication MPT 10. Le modèle de ce système a été évoqué dans le chapitre II de ce mémoire. Comme tous les équipements utilisés sont identiques, nous avons un unique graphe de processus qui représente la tâche maître des automates et un seul modèle du coupleur Telway 7. Dans chaque graphe circulent autant de jetons qu'il y a de stations reliées au réseau (4 pour la première tranche, 5 pour la deuxième et jusqu'à 15 pour le projet final). Chaque jeton est identifié par son premier paramètre qui correspond au numéro de station Telway 7.

III.3. Echanges entre les automates

III.3.1. Données échangées

Le fonctionnement du procédé impose d'organiser les échanges inter-automates pour l'acheminement et la pesée des constituants de la façon suivante :

- pour les huiles : approvisionnement et pesée directs par les MI demandeurs,
- pour les noirs courants : approvisionnement et pesée directs par les MI demandeurs, à partir des trémies journalières affectées,
- pour les noirs N1 : demande de distribution du MI à l'automate des noirs,
- pour les blancs : demande de distribution du MI à l'automate des blancs.

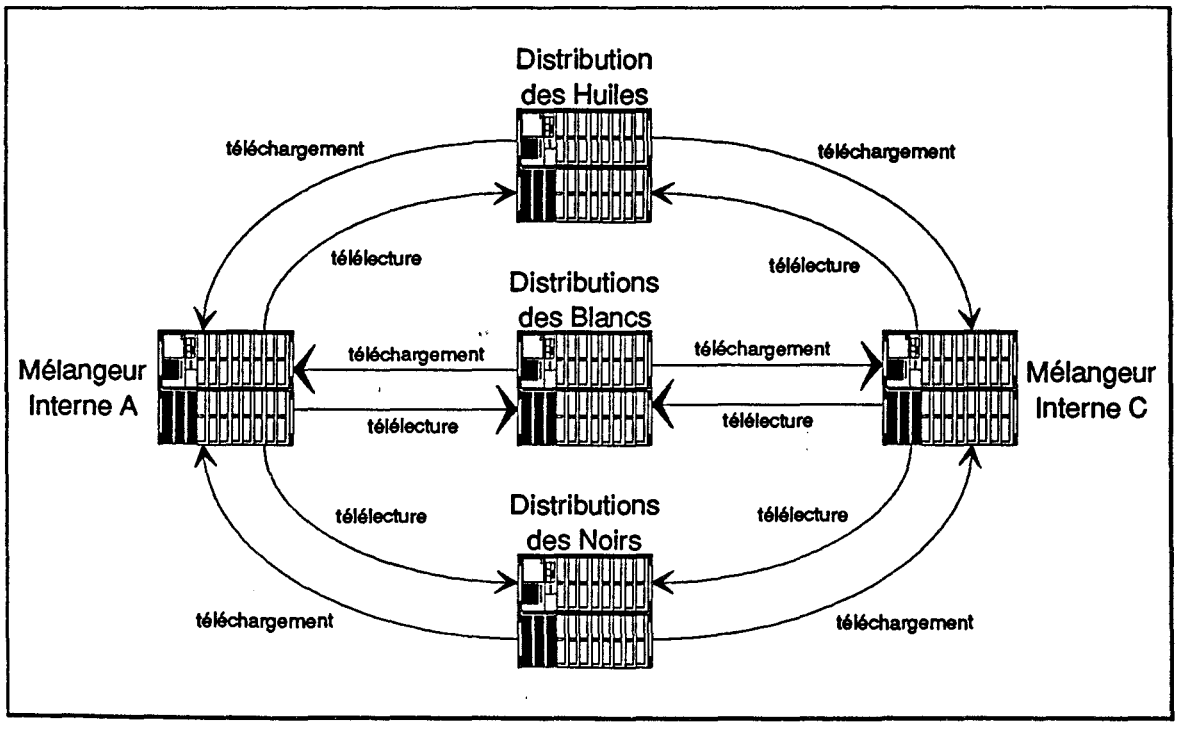


Figure A2.6 : *Organisation des échanges inter-automates (2ème tranche)*

Seuls les automates mélangeurs sont connectés au calculateur Tandem. Tous les échanges d'informations entre les automates de distribution et le Tandem (alarmes approvisionnement, état des installation de distribution, codes composants etc.) se font donc en utilisant un des mélangeurs internes comme passerelle.

Les automates des mélangeurs internes étant très chargés, il a été décidé qu'ils ne supporteraient aucun applicatif pour les échanges de données sur le réseau Telway 7. Ce sont donc les automates de distribution qui viennent lire et écrire dans les automates mélangeurs.

Les données qui transitent sur le réseau Telway 7 sont véhiculées à la fois par des mots COM et des requêtes UNITE.

1/ Infomations transmises par mots COM :

- présence défaut sur module,
- compte-rendu de mise en service,
- compte-rendu d'arrêt des installations,
- demande et validation de traitements distants,
- etc.

2/ Informations transmises par requête UNITE de lecture et d'écriture de mots :

Automate des huiles :

- télélecture de 2 mots d'état mélangeur sur CHAQUE MI : de manière régulière,
- téléchargement des 2 mots d'état des huiles sur CHAQUE MI: de manière régulière,
- téléchargement des 10 mots d'alarmes approvisionnement sur le Tandem, via UN des MI : sur apparition d'une alarme,
- télélecture des 120 mots de codes composants sur le tandem, via UN des MI : hors production lors du démarrage de l'installation.

Automate des blancs :

- télélecture des pesées blanches à réaliser (3 mots), sur CHAQUE MI, sur demande de celui-ci : demandes régulières pour chaque charge (moyenne 6 mn)
- télélecture de 2 mots d'état mélangeur sur CHAQUE MI : de manière régulière,
- téléchargement de 2 mots d'état des blancs sur CHAQUE M: de manière régulière,
- téléchargement des 10 mots d'alarmes approvisionnement sur le Tandem, via UN des MI : sur apparition d'une alarme,
- télélecture des 42 mots de codes composants sur le tandem, via UN des MI : hors production lors du démarrage de l'installation.

Automate des noirs :

- télélecture des pesées noires N1 à réaliser (3 mots), sur CHAQUE MI sur demande de celui-ci : quelques demandes par jour,
- télélecture de 2 mots d'état mélangeur sur CHAQUE MI : de manière régulière,
- téléchargement des 2 mots d'état des noirs sur CHAQUE MI : de manière régulière,
- téléchargement des 10 mots d'alarmes approvisionnement sur le Tandem, via UN des MI : sur apparition d'une alarme,
- télélecture des 72 mots de codes composants sur le tandem, via UN des MI : hors production lors du démarrage de l'installation.

Automate MI A, B et C :

- aucun échange applicatif sur le réseau Telway 7.

III.3.2. Organisation des échanges sur le réseau Telway 7

Un protocole applicatif est défini sur le réseau Telway 7 pour règlementer les échanges de télélecture et téléchargement qui viennent d'être détaillés. Le gestionnaire des échanges est l'automate des huiles. En plus de ses propres communications avec les autres stations, il désigne à tour de rôle l'automate autorisé à dialoguer sur le réseau Telway 7. La réservation du bus étant ainsi faite, la station désignée peut, si elle a des échanges à effectuer, écrire ou lire des données dans les automates de son choix. Lorsqu'il a terminé ses échanges, l'automate autorisé à émettre statue la fin de ses émissions. L'automate des huiles refait donc une nouvelle réservation pour une autre station.

Tous les automates, sauf ceux des mélangeurs internes, sont régulièrement interrogés par l'automate des huiles. L'attribution et la restitution du droit de parole sont faites par l'intermédiaire de mots COM, selon la figure suivante :

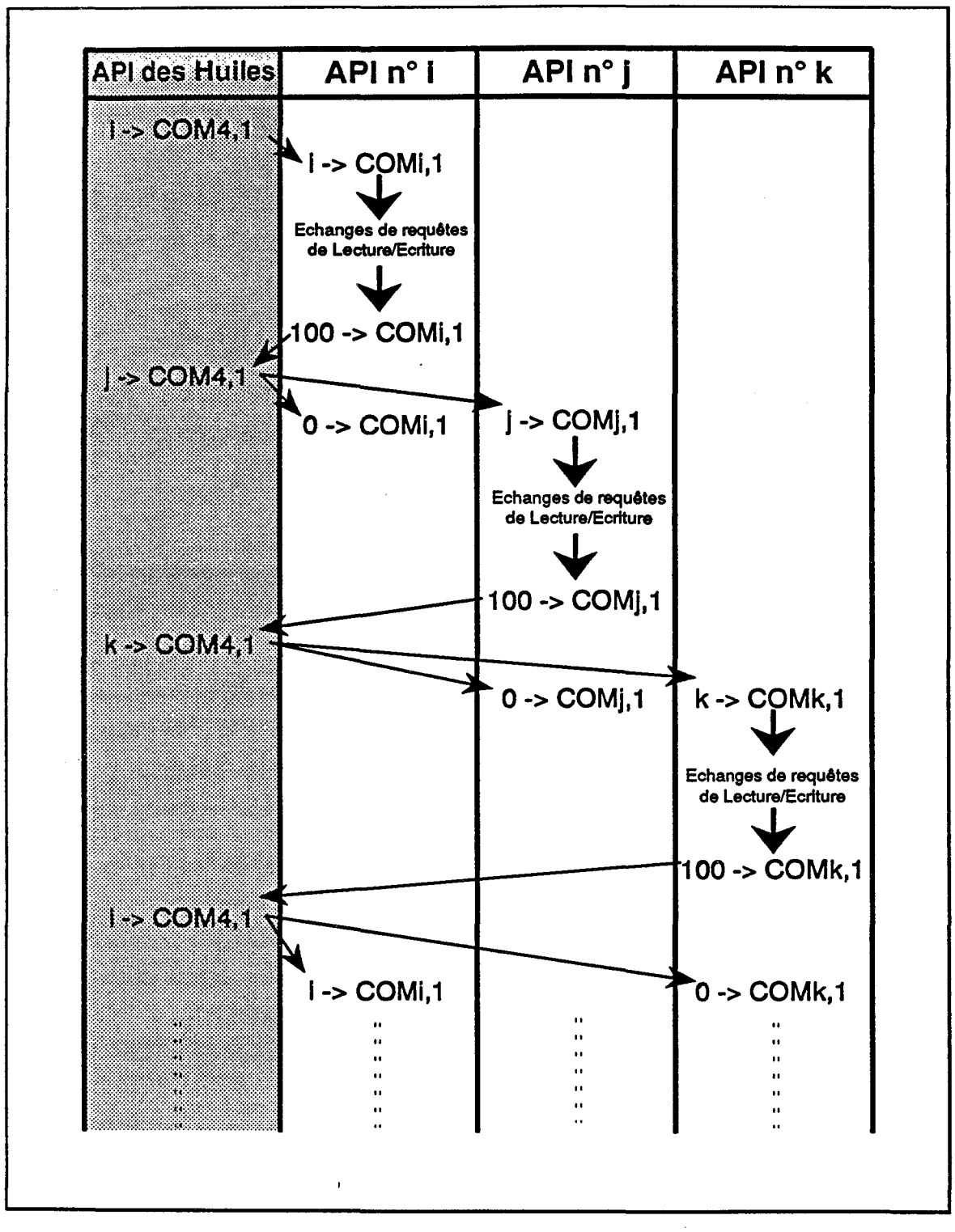


Figure A2.7 : *Fonctionnement du protocole applicatif sur Telway 7*

Compte-tenu de cette organisation des communications et des données qu'il a à échanger avec les autres stations, le fonctionnement des échanges de l'automate des huiles se résume par l'organigramme suivant (ceux des blancs et des noirs sont similaires) :

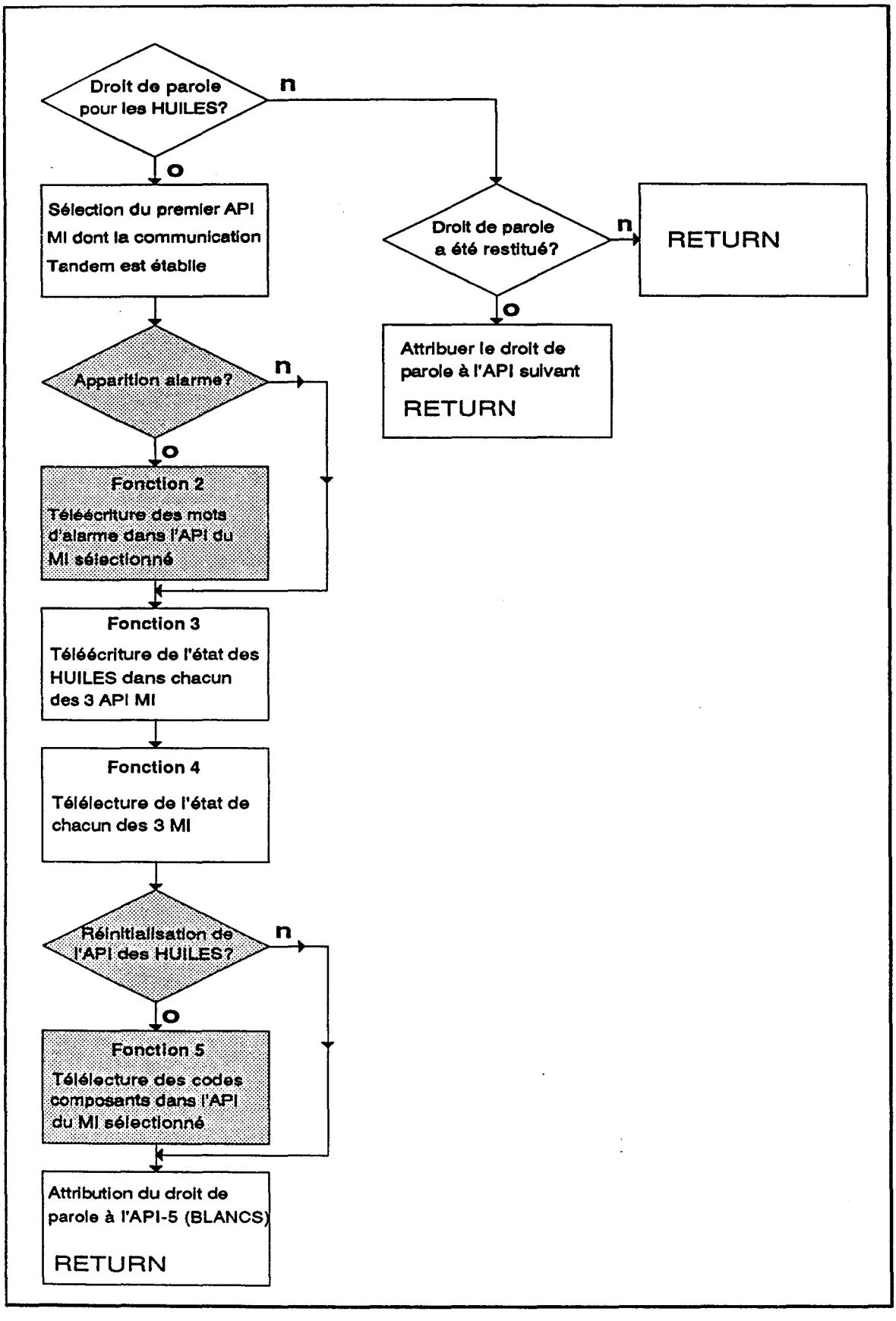


Figure A2.8 : *Séquence de dialogue de l'automate des huiles*

III.4. Modélisation des échanges entre les automates

III.4.1. Approche générale

Il s'agit de compléter le modèle de l'architecture matérielle présenté au paragraphe III.2. avec une description des applicatifs de l'application. Comme ceux-ci dépendent de chaque équipement, nous obtenons un modèle complet qui est organisé de la façon suivante :

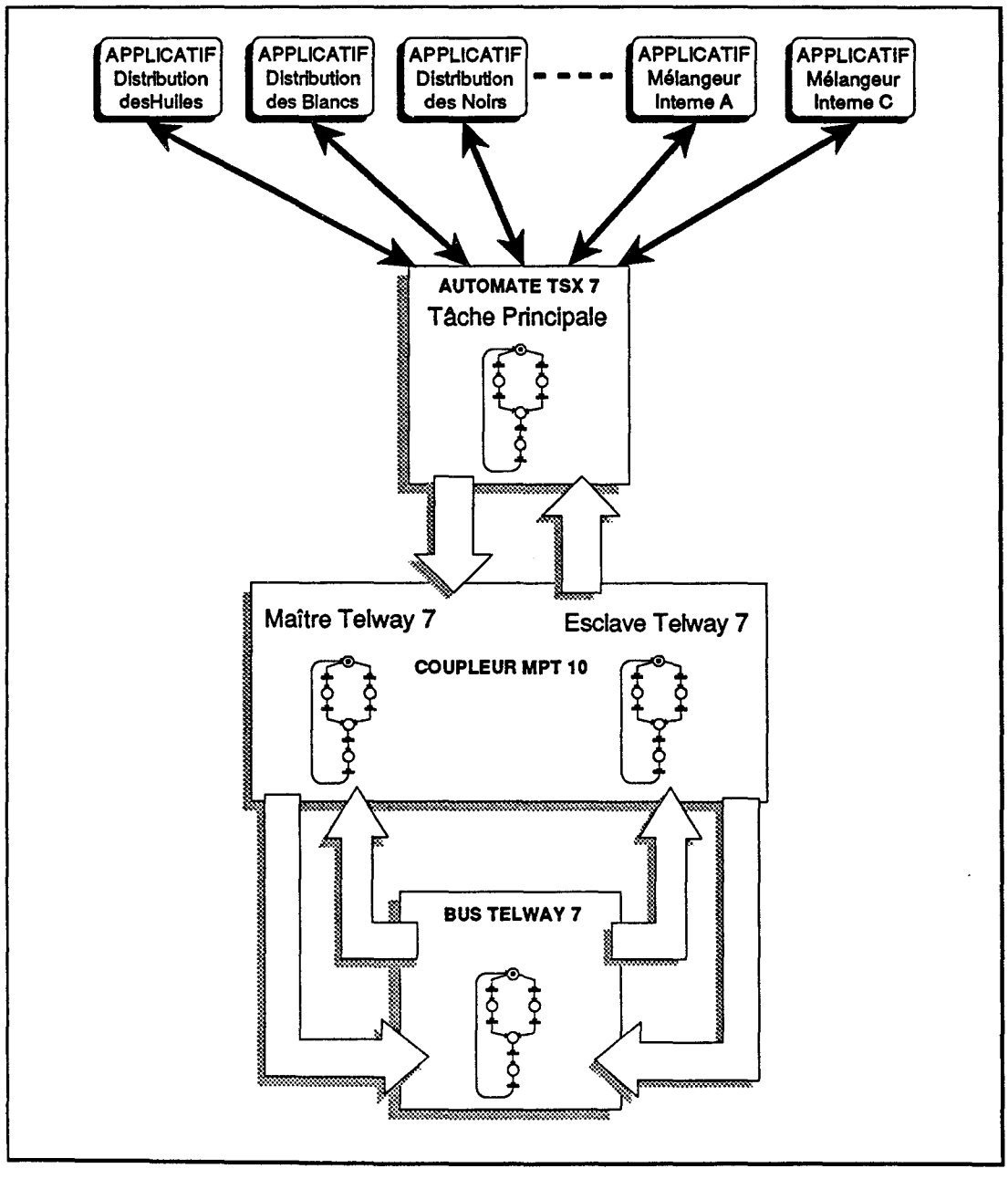


Figure A2.9 : *Ajout de la description des applicatifs sur le modèle des équipements*

Comme nous l'avons proposé dans le chapitre II de ce mémoire, les applicatifs sont détaillés par des procédures écrites en langage Pascal. L'interprétation du formalisme lors de la simulation des modèles réseaux de Petri permet de faire le lien entre ces procédures et le modèle du fonctionnement des automates TSX série 7.

Des structures de données, définies avec le modèle de la tâche maître de l'automate, permettent de manipuler directement les mots COM, les E/S TOR ainsi que les blocs TXT pour l'émission et la réception des requêtes UNITE. Pour cet exemple nous avons essayé de rapprocher le modèle le plus possible du code automate PL7-3, qui pour cette application a été développé en langage littéral Télémécanique.

III.4.2. Application sur l'architecture de La-Barre-Thomas

Sur cet exemple, toutes les routines de communication sont implantées dans le PRELIMINAIRE : code écrit en langage littéral qui est exécuté au début de la tâche maître, à chaque cycle automate. Les fonctionnalités de ces tâches sont donc simplement reprises dans notre modèle. Afin d'illustrer la façon dont les applicatifs ont été décrits, nous avons détaillé la fonction 4 de l'automate des huiles (Cf. figure A2.8) :

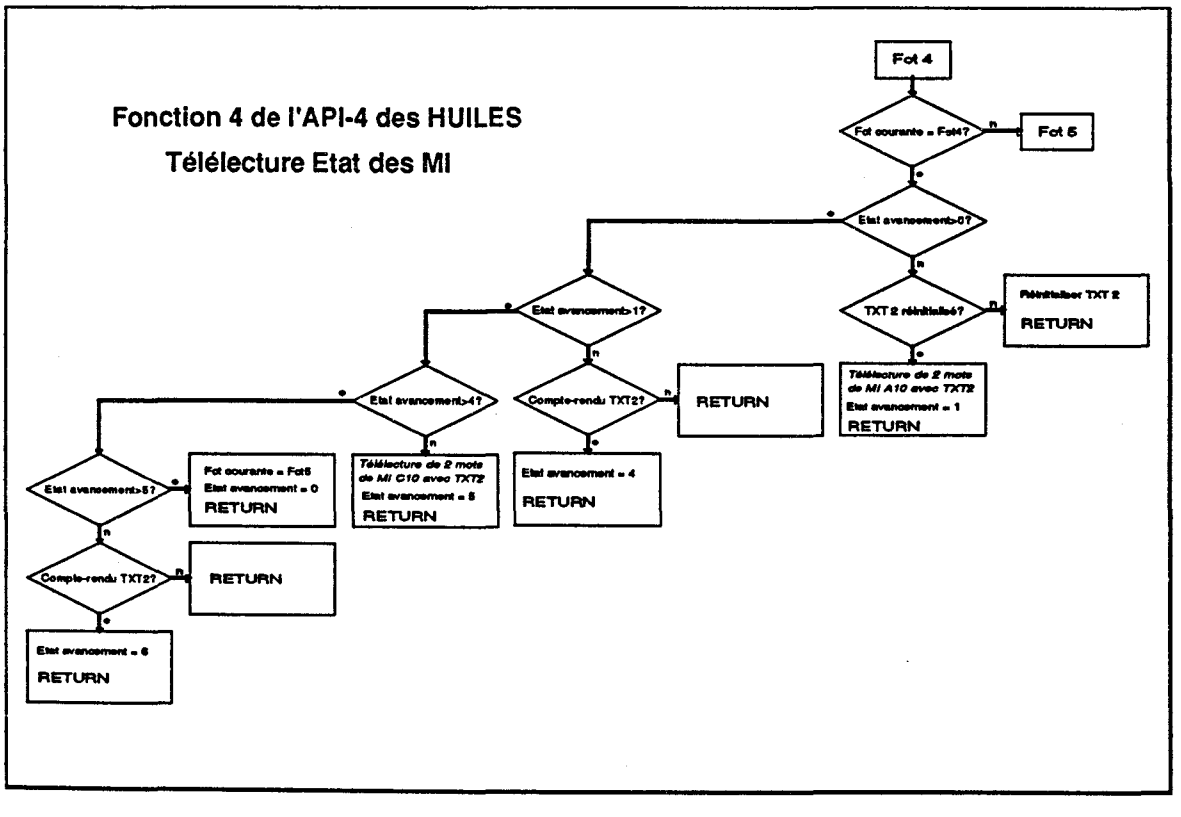
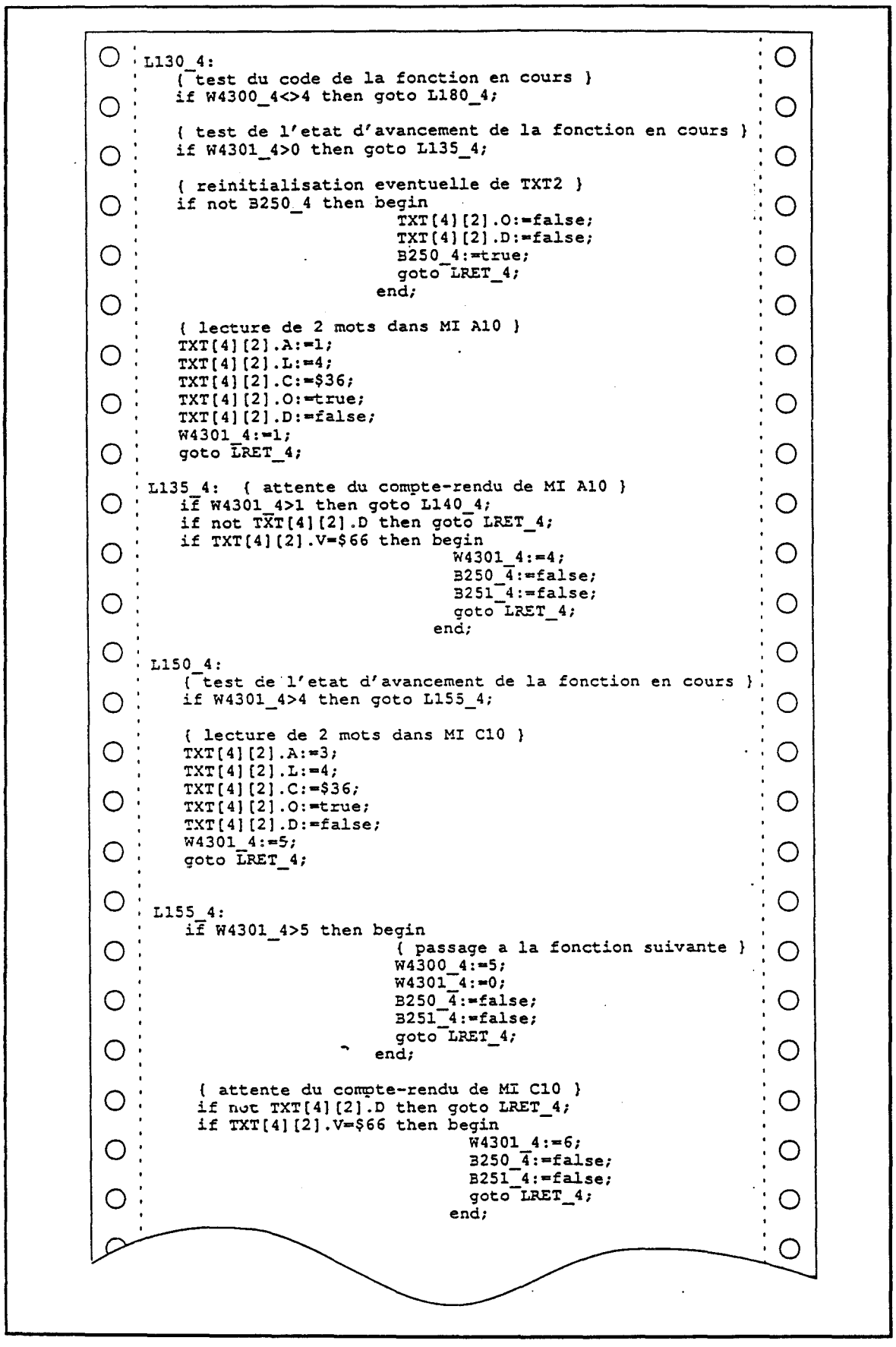


Figure A2.10 : *Télélecture de l'état des mélangeurs internes A et C (fonction 4)*



```
L130_4:
   { test du code de la fonction en cours }
   if W4300_4<>4 then goto L180_4;

   { test de l'etat d'avancement de la fonction en cours }
   if W4301_4>0 then goto L135_4;

   { reinitialisation eventuelle de TXT2 }
   if not B250_4 then begin
                        TXT[4][2].O:=false;
                        TXT[4][2].D:=false;
                        B250_4:=true;
                        goto LRET_4;
                      end;

   { lecture de 2 mots dans MI A10 }
   TXT[4][2].A:=1;
   TXT[4][2].L:=4;
   TXT[4][2].C:=$36;
   TXT[4][2].O:=true;
   TXT[4][2].D:=false;
   W4301_4:=1;
   goto LRET_4;

L135_4:  { attente du compte-rendu de MI A10 }
   if W4301_4>1 then goto L140_4;
   if not TXT[4][2].D then goto LRET_4;
   if TXT[4][2].V=$66 then begin
                             W4301_4:=4;
                             B250_4:=false;
                             B251_4:=false;
                             goto LRET_4;
                           end;

L150_4:
   { test de l'etat d'avancement de la fonction en cours }
   if W4301_4>4 then goto L155_4;

   { lecture de 2 mots dans MI C10 }
   TXT[4][2].A:=3;
   TXT[4][2].L:=4;
   TXT[4][2].C:=$36;
   TXT[4][2].O:=true;
   TXT[4][2].D:=false;
   W4301_4:=5;
   goto LRET_4;

L155_4:
   if W4301_4>5 then begin
                       { passage a la fonction suivante }
                       W4300_4:=5;
                       W4301_4:=0;
                       B250_4:=false;
                       B251_4:=false;
                       goto LRET_4;
                     end;

    { attente du compte-rendu de MI C10 }
    if not TXT[4][2].D then goto LRET_4;
    if TXT[4][2].V=$66 then begin
                              W4301_4:=6;
                              B250_4:=false;
                              B251_4:=false;
                              goto LRET_4;
                            end;
```

Figure A2.11 : *Description de la fonction 4 de télélecture de l'état des MI A et C*

Le préliminaire est exécuté à chaque cycle automate, il est donc indispensable de disposer de variables automate pour mémoriser les états d'avancement des échanges:

- W4300 : numéro de fonction actuellement en cours (entre 1 et 5)
- W4301 : état d'avancement de la fonction en cours

L'émission et la réception des requêtes de télélecture et de téléchargement se font en utilisant des blocs TXT, avec les paramètres suivants :

- TXT.O : variable booléenne à positionner pour émettre la requête,
- TXT.D : variable booléenne positionnée lors de la réception du compte-rendu,
- TXT.A : adresse de la station destinataire de la requête,
- TXT.L : longueur de la table des données émises,
- TXT.C : code de la requête ($36 = télélecture - $37 = téléchargement),
- TXT.V : code du compte-rendu.

IV. RESULTATS DE SIMULATION

IV.1. Vérification de bon fonctionnement

Un modèle des échanges de données sur une architecture de commande permet une observation simultanée du fonctionnement de tous les équipements. Durant les phases d'analyse et de conception, l'utilisation d'un tel modèle peut permettre de découvrir rapidement les erreurs dans l'organisation des échanges entre les stations. Celles-ci peuvent provenir d'un mauvais séquencement logique des échanges, mais aussi de retards sur les données liés aux performances de l'architecture.

Sur l'exemple de La-Barre-Thomas, nous allons montrer l'intérêt d'une telle approche en étudiant le fonctionnement du protocole utilisateur. Pour spécifier cet applicatif dans notre modèle, nous avons utilisé le code source implanté sur les automates TSX 7.

La première application simulée consiste à attribuer "à vide", le droit de parole successivement aux différents automates du réseau. Dès qu'une station reçoit l'autorisation d'échanger des messages, elle statue immédiatement la fin de ses échanges pour que le gestionnaire de réseau attribue le médium à la station suivante.

En simulant ce modèle, nous nous sommes aperçu qu'il existait un dysfonctionnement dans le séquencement du protocole applicatif. En effet, une même station peut prendre et rendre plusieurs fois de suite le médium sans que le gestionnaire du réseau ait eu le temps d'attribuer le bus à une autre station. Ce phénomène se détecte sur le modèle lorsqu'on observe, au travers des valeurs des mots COM, quel automate prend et restitue le bus pour ses échanges.

En fait, l'implémentation du protocole applicatif serait parfaitement correcte si les mots COM constituaient réellement une table de données partagées entre les stations du réseau. Le fonctionnement "logique" tel qu'il est décrit sur la figure A2.7 est correct. Mais, à cause des communications sur le bus Telway 7, il existe toujours un retard entre la modification d'un mot COM dans un automate, et la prise en compte effective de cette modification par les autres stations.

Ce retard est lié a trois phénomènes :

a/ les mots COM modifiés par une station au cours d'un cycle automate ne sont transférés qu'en fin de cycle au coupleur de communication Telway 7,

b/ un coupleur Telway 7 ne peut émettre ses mots COM sur le réseau à destination des autres stations que lorsqu'il est interrogé par le maître (le protocole qui règlemente l'accès au médium est de type maître/esclave). Ces interrogations se font cycliquement, avec un intervalle d'environ 120 ms (avec 5 API sur le réseau) entre deux interrogations successives de la même station Telway 7,

c/ les mots COM reçus par un coupleur Telway 7 ne sont pris en compte par la tâche maître qu'au début du cycle suivant de l'automate.

Si les temps de cycle automate sont de 100 ms, le retard entre la modification d'un mot COM par une station et sa prise en compte effective par les autres stations peut dépasser 500 ms (valeur confirmée par la documentation Telway 7 de la Télémécanique).

La figure A2.12 détaille le séquencement **TEMPORISE** des échanges entre les automates des huiles, des blancs et des noirs. Nous nous sommes placés dans le cas où il y a cinq automates sur le réseau. Le maître du protocole Telway 7 doit donc interroger cycliquement cinq stations, et le gestionnaire applicatif trois. Pour chaque automate de distribution, nous avons détaillé les cycles de la tâche maître, avec la lecture des COM depuis le coupleur en début de cycle, et la mise à jour des COM en fin de cycle dans ce même coupleur. La colonne de droite symbolise le séquencement des échanges sur le bus Telway 7, avec l'interrogation cyclique des stations et les valeurs des mots COM transférés.

Nous voyons alors que l'automate des blancs prend et restitue quatre fois de suite le médium Telway 7 avant que l'automate des noirs n'ait accès au bus! Cela vient du fait qu'il existe un délai entre le moment où le bus est restitué par l'automate des blancs et celui où ce même automate des blancs prend en compte l'attribution du médium à l'automate des noirs.

Ce dysfonctionnement n'entrave a priori pas le bon déroulement des échanges de télélecture et de téléchargement. Deux stations ne peuvent jamais disposer simultanément du bus car au début de chaque cycle, les automates testent si le médium leur est toujours attribué. Cependant, il peut arriver qu'une station exécute plusieurs fois de suite les fonctions qui lui sont imparties. Ceci peut s'avérer très pénalisant pour les performances de l'application.

Pour empêcher ce phénomène, il suffit d'apporter une légère modification dans l'implémentation du protocole applicatif : une station n'est autorisée à émettre sur le médium que si le gestionnaire lui attribue le droit d'accès au médium ET n'a pas restitué ce droit précédemment.

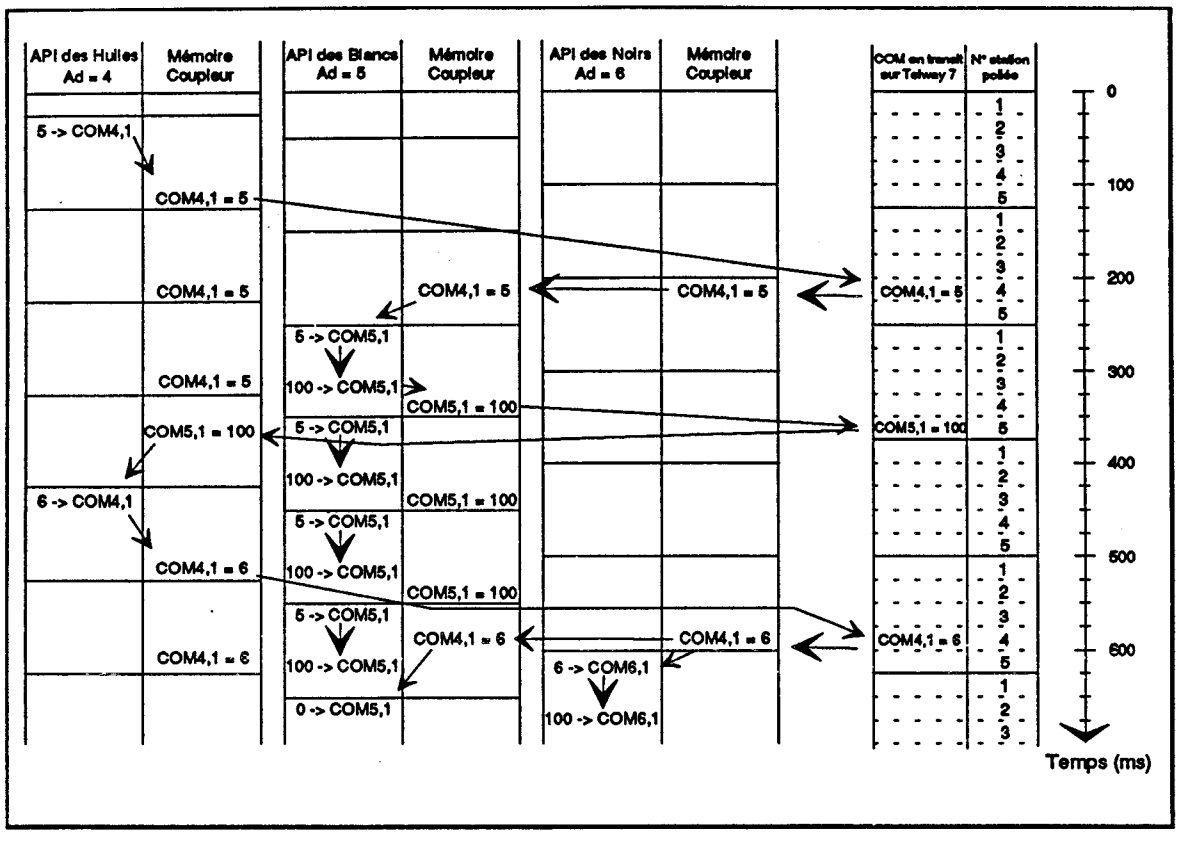


Figure A2.12 : *séquencement temporisé des échanges de mots COM inter-automates*

IV.2. Validation des performances du système

Tous les résultats présentés dans ce paragraphe ont été obtenus par simulation du modèle décrit précédemment. Nous avons mené trois études successives :

a/ 4 automates : API des Huiles, des Blancs et des Noirs + MI A,

b/ 5 automates : API des Huiles, des Blancs et des Noirs + MI A et C,

c/ 6 automates : API des Huiles, des Blancs et des Noirs + MI A, B et C.

Pour chaque automate, nous avons pris un temps de cycle de la tâche maître de 100 ms. Durant la simulation, les données mesurées sont stockées dans des fichiers. Elles sont ensuite exploitées en représentant la distribution des valeurs de chaque paramètre. On obtient donc des histogrammes avec en abscisse des durées discrétisées en millisecondes et en ordonnée un taux proportionnel au nombre de fois où la valeur a été mesurée dans la simulation (sous-graphes des figures A2.13 à A2.16). Chaque paramètre est caractérisé par sa valeur moyenne et la dispersion autour de cette moyenne (traits verticaux sur les figures A2.13 à A2.16). Afin d'observer l'influence du nombre de stations sur les performances, nous avons, pour chaque paramètre, regroupé sur un même graphique les différentes valeurs obtenues pour chacune des études a, b et c

IV.2.1. Performances du protocole applicatif "à vide"

Pour évaluer les performances du protocole applicatif, nous avons mesuré le temps qui sépare deux attributions successives du bus à une même station. Ce délai est appelé Temps de Cycle Applicatif (TCA). Cette valeur est importante lors de la mise au point du protocole car elle permet très rapidement d'avoir une première idée sur les temps des échanges.

Nous avons fait des mesures, avec 4 (1ère tranche), 5 (2ème tranche), 6, 9, 12 et finalement 15 automates (projet complet) sur le réseau. Le TCA atteint 9 secondes avec 15 stations! Ceci s'explique par le fait que le temps de transfert des COM augmente avec le nombre d'automates, pour atteindre environ 400 ms avec 15 stations. Comme le temps de cycle est grand (100 ms), on obtient des résultats qui se chiffrent en secondes.

Connaître le TCA est important, car il a une influence directe sur les fréquences auxquelles les stations peuvent émettre leurs requêtes et sur les temps d'attente avant que l'envoi d'un message soit possible. Ceci est simplement lié au fait qu'un automate n'est autorisé à émettre que lorsque le médium lui est attribué.

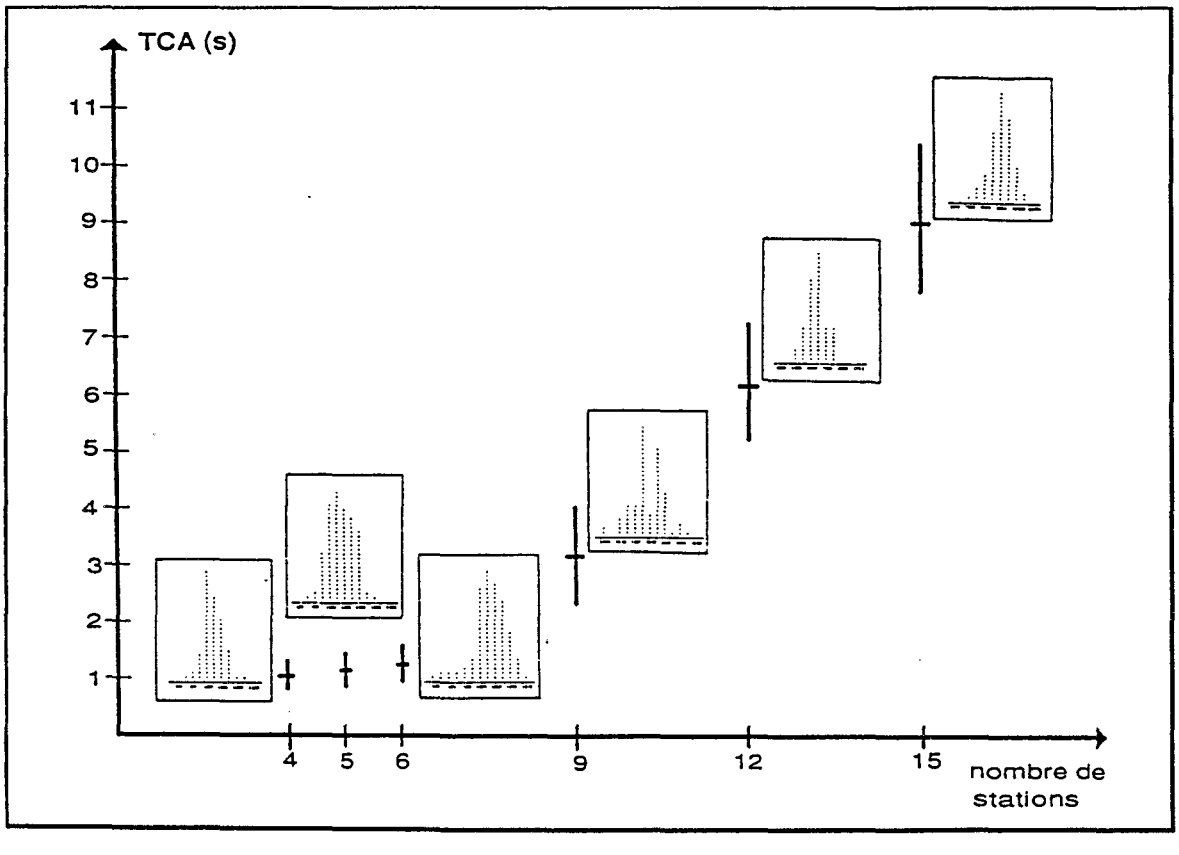


Figure A2.13 : *Temps de Cycle Applicatif à vide (TCA)*

IV.2.2. Performances du protocole applicatif "en charge"

Cette mesure correspond au fonctionnement réel de l'installation. Chaque station exécute ses requêtes de télélecture et de téléchargement. Le TCA "en charge" est sensiblement plus important que celui "à vide" du paragraphe précédent. Cette différence est due au temps nécessaire à chaque station pour émettre ses requêtes de lecture/écriture et attendre les comptes-rendus associés.

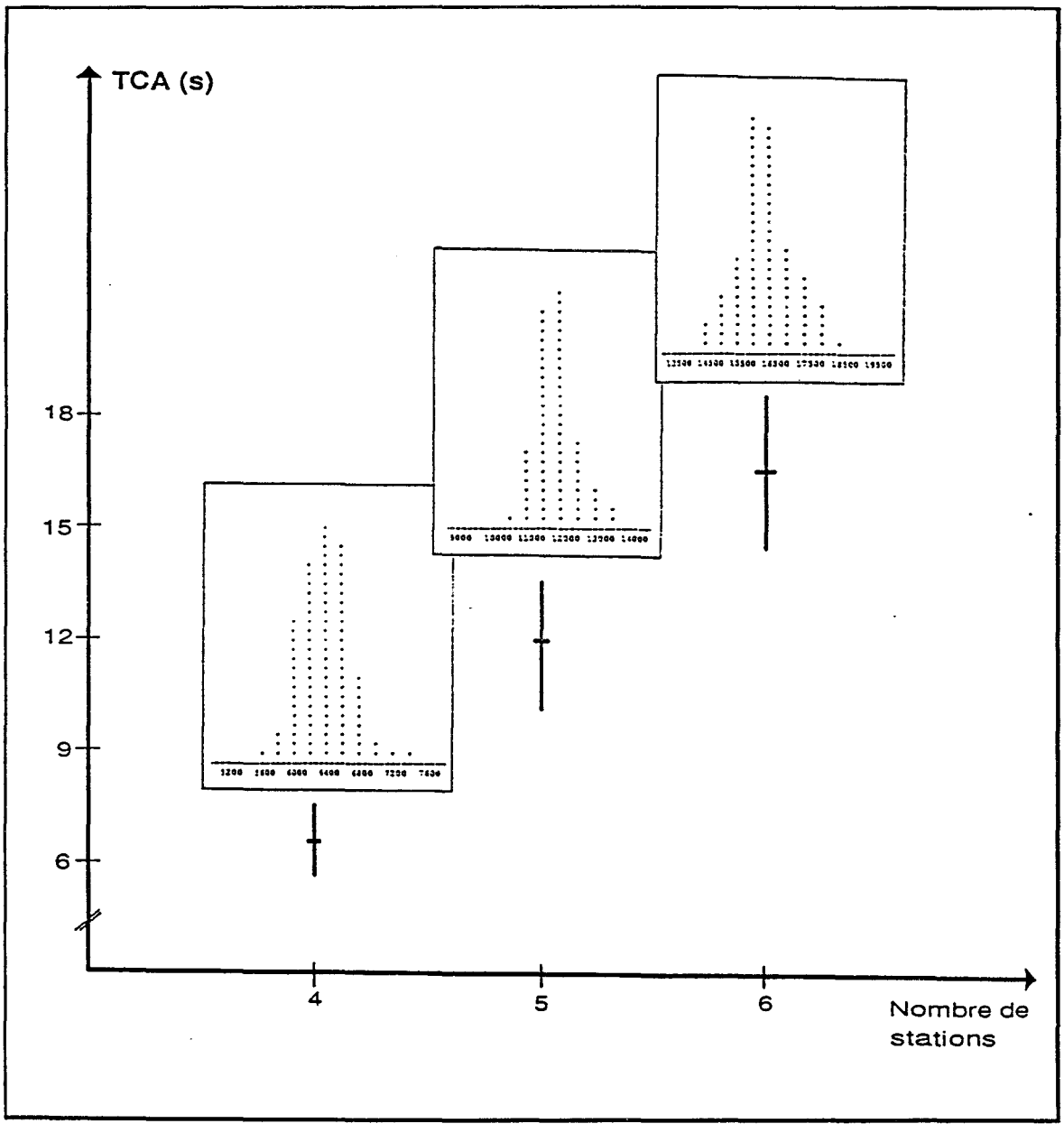


Figure A2.14 : *Temps de Cycle Applicatif en charge (TCA)*

IV.2.3. Fréquence de mise à jour des mots d'état

Chaque automate de distribution envoie régulièrement ses mots d'état vers chaque mélangeur interne et lit les mots d'état de ces mêmes mélangeurs. Ces lectures/écritures se font le plus rapidement possible en fonction des possibilités du protocole applicatif. Il est intéressant de mesurer la fréquence de ces lectures/écritures pour vérifier qu'elle reste compatible avec les délais maximum autorisés pour avertir une station d'un changement d'état. Le graphique ci-dessous représente la Fréquence d'Etat des Huiles (FEH). Vue la symétrie de l'installation, les valeurs sont les mêmes pour la fréquence d'état des Noirs ou des Blancs, ainsi que pour l'état des mélangeurs internes.

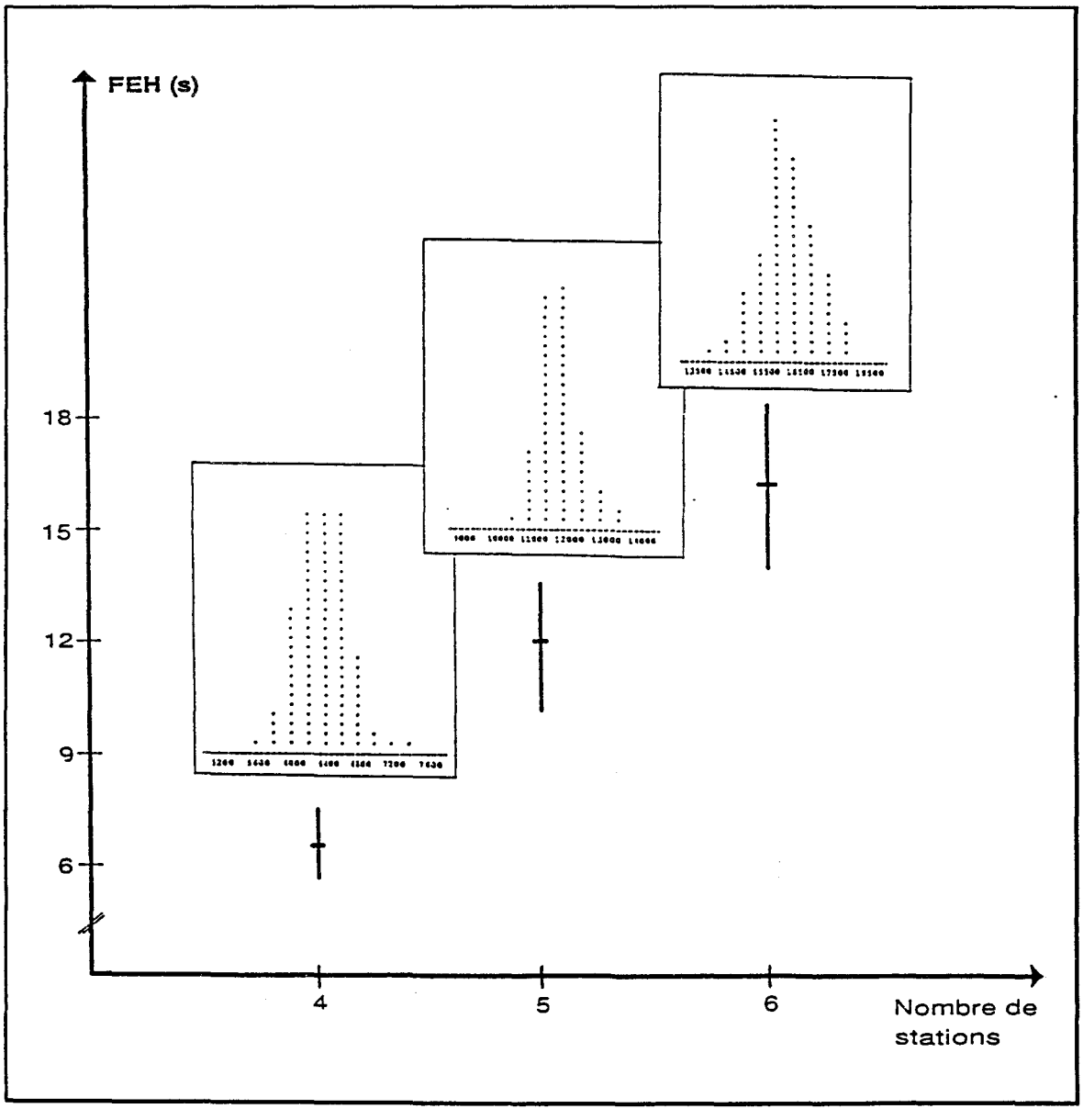


Figure A2.15 : *Fréquence d'état des huiles (FEH)*

IV.2.4. Temps de cycle réseau

Le Temps de Cycle Réseau (TCR) est le temps qui sépare deux interrogations successives d'un même esclave sur le bus Telway 7. Ce temps est lié au séquencement du protocole d'accès au médium employé (polling de type maître/esclave) et à la taille des requêtes qui sont échangées. Nous observons que TCR augmente avec le nombre d'automates, et on peut affirmer qu'avec 15 stations il sera autour de 400 à 500 ms, ce qui aura une influence importante sur les performances globales du système. C'est parce que TCR croît avec le nombre d'automates, que toutes les valeurs mesurées jusqu'ici augmentent de la même façon.

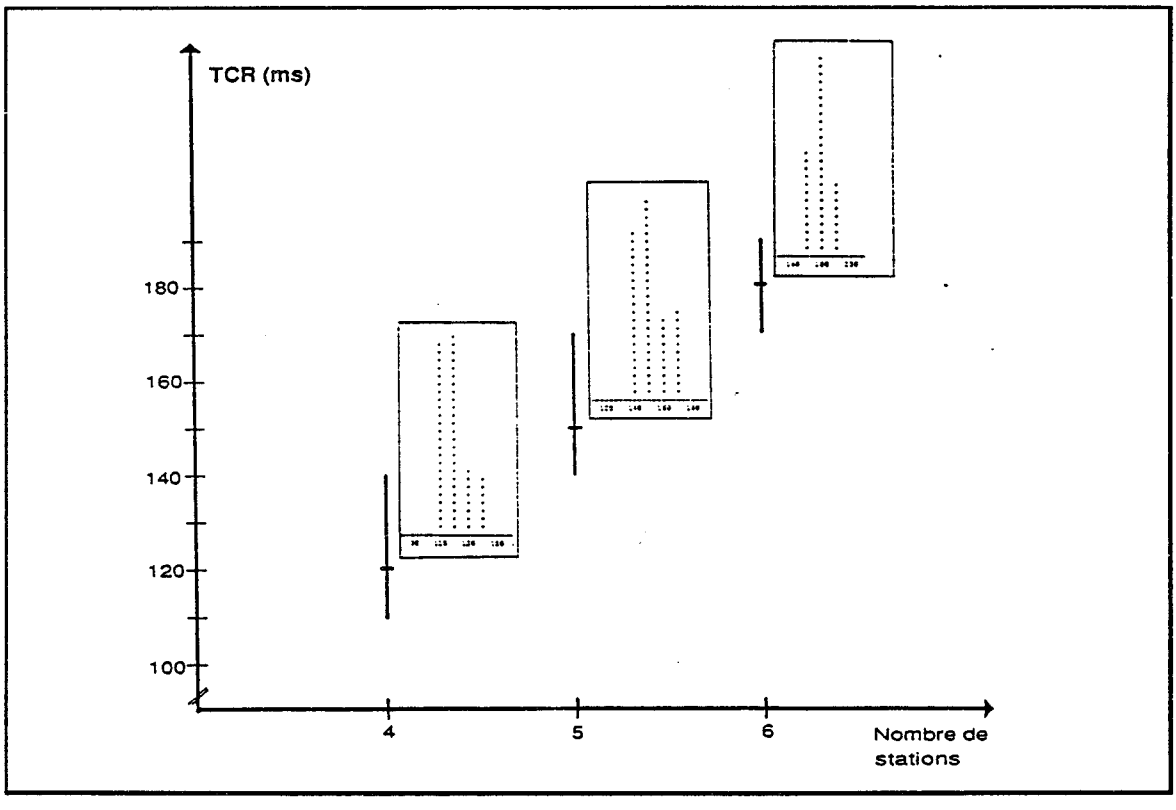


Figure A2.16 : *Temps de Cycle Réseau (TCR)*

IV.2.5. Zone d'échange des requêtes

Le tableau ci-dessous représente le nombre de Messages en Attente dans le coupleur d'un automate Mélangeur interne (MAM). Ce nombre oscille entre zéro message et un message, ce qui est parfaitement normal car le protocole applicatif mis en place n'autorise l'envoi que d'une requête à la fois !

MAM (Messages en Attente Mélangeur)		
4 TSX 87	5 TSX 87	6 TSX 87
		
Valeur mini : 0 Valeur maxi : 1	Valeur mini : 0 Valeur maxi : 1	Valeur mini : 0 Valeur maxi : 1

Figure A2.17 : *Nombre de Message en Attente dans le Mélangeur (MAM)*

V. CONCLUSION

Après avoir obtenu les résultats de simulation qui viennent d'être présentés, nous nous sommes rendus dans l'atelier de Rennes pour confronter nos résultats à la réalité du site.

Le dysfonctionnement dans l'organisation des échanges, qui a été mis en évidence par la simulation, a bien été constaté sur l'installation. En effet une correction avait été apportée dans les sous-routines de communication des automates pour empêcher suite à une restitution du droit de parole par une station, la reprise immédiate du médium par le même automate lorsque le gestionnaire des échanges n'a pas encore détecté la restitution du bus.

Entre temps l'installation avait sensiblement évoluée, notamment par l'ajout sur le réseau Telway 7 d'un nouvel automate pour assurer la commande d'un tapis de convoyage. Cette station supplémentaire participe au protocole applicatif d'attribution du droit de parole sur le bus.

Les hypothèses prises lors des simulations ne correspondent donc plus exactement à l'application installée dans l'atelier. Nous avons cependant réalisé des mesures sur le paramètre TCA en essayant de voir l'influence de l'automate supplémentaire sur ce paramètre. Les valeurs ainsi obtenues par recoupement, correspondent à l'ordre de grandeur des valeurs simulées. Nous retrouvons en effet des résultats de plus de dix secondes pour le TCA.

Cette première mise en oeuvre, sur un cas industriel du groupe PSA Peugeot Citroën, de la démarche de modélisation/simulation développée dans ce mémoire, a permis d'obtenir des résultats qualitativement corrects, mais qui restent à affiner du point de vue quantitatif. Les conditions de mesures n'ont en effet pas permis d'évaluer finement la précision au modèle. C'est la raison pour laquelle nous avons traité un autre cas industriel, qui est détaillé dans le chapitre III de ce mémoire.

RESUME :

Nous présentons dans ce mémoire une approche de modélisation d'architecture de commande et de pilotage d'atelier. L'objectif est de simuler les communications de tels systèmes afin d'en valider les performances avant installation sur site. Pour construire et exploiter les modèles, nous avons utilisé un formalisme de réseaux de Petri à structures de données. Trois niveaux de modélisation on été mis en évidence. Chaque équipement de commande (automate programmable, coupleur de communication etc.) est modélisé de manière générique, afin de pouvoir être réutilisé dans chaque étude où ce composant intervient. La juxtaposition de ces modèles permet de d'élaborer des modèles complets d'architecture matérielle. Pour tenir compte des spécificités de chaque installation, ces derniers sont interfacés avec une description du séquencement des échanges pour représenter les applicatifs de commande et avec une image de l'environnement pour créer des scénarios de charge réaliste. Des études, menées sur des installations automatisées d'un constructeur automobile, ont permis de démontrer la faisabilité de la démarche et d'en saisir l'intérêt en terme de validation.

MOTS-CLES :

Modélisation - Simulation - Réseaux de Petri - Performances - Temps réel - Architecture de commande d'atelier- Réseau local industriel - Automate programmable.

ABSTRACT :

We present in this document an approach of modelling workshop command and piloting architecture. The goal is to simulate the communications of such systems in order to validate the performances prior to their on-site installation. To construct and exploit the models, we have used a Petri nets with data structures formalism. Three modelling levels have been indentified. Each command equipment (programmable logic controller, communication controller etc.) is generically modelled in order to be re-used in each study where this component appears. The model of the complete hardware architecture is produced by assembling those elementary models. To take the specificities of each installation into account, those models are interfaced with a description of the exchanges sequences to represent command applications and with an image of the environment to create simulation scenario. Studies, on automated installations of a car builder, have proved the faisability of the approach and have shown the interest in validation order.

KEY-WORDS :

Modelling - Simulation - Petri nets - Performances - Real time - Workshop command architecture - Industrial local area network - Programmable logic controller.