CNRS UMR 7039

UFR Ecole Supérieure des Sciences et Technologies de l'Ingénieur de Nancy
Ecole Doctorale IAE+M
DFD Automatique et Production Automatisée

Thèse

Présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy 1

Spécialité Automatique

Par **Pavol BARGER**

EVALUATION ET VALIDATION DE
LA FIABILITE ET DE LA DISPONIBILITE
DES SYSTEMES D'AUTOMATISATION A INTELLIGENCE DISTRIBUEE,
EN PHASE DYNAMIQUE

Soutenue publiquement le 15 décembre 2003 devant la commission d'examen

Membres du jury :

Président :	J.-F. AUBRY	Professeur, Institut National Polytechnique de Lorraine, Nancy
Rapporteurs :	R. VALETTE	Directeur de Recherche CNRS, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse
	P. YIM	Professeur, Ecole Centrale, Lille
Examinateurs :	I. AUGE-BLUM	Maître de Conférences, Institut National des Sciences Appliquées, Lyon
	M. ROBERT	Professeur, Université Henri Poincaré Nancy 1 (directeur de thèse)
	J.-M. THIRIET	Maître de Conférences, Université Henri Poincaré Nancy 1 (co-directeur de thèse)
Invité :	P. CHARPENTIER	Docteur, Ingénieur, Institut National de Recherche et de Sécurité, Nancy

Centre de Recherche en Automatique de Nancy, 2 rue Jean Lamour, 54519 Vandœuvre cedex

A Alexandra et mes parents

Remerciements

Je voudrais remercier Messieurs Michel ROBERT et Jean-Marc THIRIET, mes encadrants, pour l'aventure à laquelle ils m'ont invité. Si je devais la revivre, c'est eux que je prendrais pour guides.

Un merci sincère s'impose aux rapporteurs, Messieurs Robert VALETTE et Pascal YIM qui ont su, à travers leurs conseils, apporter un regard constructif à ce travail.

Jean-François Aubry a beaucoup contribué à ma thèse à la fois sur le contenu scientifique par des échanges encourageants et également sur l'environnement matériel. Je tiens à le remercier pour tout cela.

Les travaux d'Isabelle Augé-Blum ont été une source d'inspiration pour l'avancée de ce travail de recherche.

Philippe Charpentier, Pascal Lamy et Marc Kneppert pour la coopération avec l'INRS qui a permis la validation d'une partie de notre travail sur un problème concret.

L'ensemble de l'équipe du CRAN à l'ESSTIN m'a supporté avec beaucoup de patience et de gentillesse. Merci : Marie-Christine, Frédérique, Marion, Sylvie, Sandrine, Jean-Luc, Jean-Marie, Christophe, Gilles, Gérard, Eric, Philippe, Blaise, Jean-Claude, Davy, Fabien, Mohamed et Alexandre.

Je remercie l'ensemble du personnel de l'ESSTIN pour toute aide, en particulier Florence, Virginie, Guillaume, Eric, Stéphane et tous les autres.

Mon épouse Alexandra et mes parents ont eu quelques temps difficiles avec moi, surtout dans la dernière ligne droite. Un grand merci leur est dû.

Sommaire

Introduction 10

1 Systèmes distribués 13
1.1 Introduction générale 13
1.2 Positionnement du problème 15
1.2.1 Systèmes hybrides 15
1.2.2 Systèmes dynamiques 16
1.2.3 Systèmes réunissant l'automatique et l'informatique 16
1.3 Méthodes d'études des SAR 18
1.4 Présentation de la méthode 22
1.5 Conclusion 23

2 Sûreté de fonctionnement 25
2.1 Introduction 25
2.2 Limites de la Méthode de l'arbre des causes 26
2.3 Sûreté de fonctionnement des composants intelligents 27
2.4 Sûreté de fonctionnement dynamique 28
2.5 Sûreté de fonctionnement et réseaux de Petri 29
2.5.1 Utilisation des réseaux de Petri stochastiques 30
2.5.2 Utilisation des réseaux de Petri colorés 31
2.6 Analyse par simulation de Monte Carlo 32
2.6.1 Les techniques de réduction de variance 34
2.6.2 Le temps et la simulation 35
2.7 Simulation par sauts 37
2.8 Enjeu de taille des modèles 40

3 Réseaux de Petri colorés 43
3.1 Introduction 43
3.2 Définitions 44
3.3 Choix du logiciel 44
3.4 Introduction à la modélisation Design/CPN 46
3.4.1 Une place CPN 47
3.4.2 Une transition CPN 48
3.4.3 Un arc CPN 49
3.4.4 Temporisation des arcs 52
3.5 Quelques briques de Design/CPN 52
3.5.1 Introduction au langage ML 52
3.5.2 Tuples et trames de communication 53
3.5.3 Listes et tampon d'émission 53
3.5.4 Différences entre tuples et listes 57
3.5.5 Arc inhibiteur 57
3.5.6 Arc probabiliste 58
3.5.7 Places tampons 59
3.5.8 Temporisation synchrone (instants d'échantillonnage) 60
3.5.9 Expression de durée (temporisation asynchrone) 61

4 Modèles de base 62
4.1 Système echantillonné 62
4.2 Conception hiérarchique et premier capteur 68
4.3 Modèle d'un capteur défaillant 71
4.4 Modèle d'un canal de communication 74

5 Application de la méthode à l'usure des composants 79
5.1 Introduction 79
5.2 Modèles des composants 81
5.3 Usure des composants 84
5.3.1 Redondance capteurs 84
5.3.2 Critères d'évaluation 86
5.4 Résultats de Monte Carlo 87
5.5 Conclusion 93

6 Etude des scénarios défaillants 95
6.1 Introduction 95
6.2 Présentation des composants 97
6.2.1 Capteur 97
6.2.2 Régulateur 98
6.2.3 Actionneur 99
6.2.4 Réseau de communication 100
6.3 Dépendances des défaillances 101
6.4 Simulation de Monte Carlo 103
6.5 Scénarios 105
6.6 Dépendances dynamiques 108
6.7 Résultats de la méthode, conclusions 109

7 Modélisation d'un réseau de sécurité réel 111
7.1 Introduction 111
7.2 AS-i et AS-i Safety at Work 112
7.3 Monte Carlo, graphe d'occurrence et leur complémentairité 112
7.4 Transposition d'une norme vers un modèle CPN 114
7.5 Méthode d'analyse 118
7.6 Synthèse des résultats de modélisation et d'analyse 121
7.7 Conclusion 121

8 Conclusion 123

Glossaire 126

Bibliographie 128

Annexe A : Sûreté de fonctionnement 136
A.1 Evaluation de la sûreté de fonctionnement 136
A.1.1 Définitions de base 136
A.1.2 Evaluation des systèmes 137
A.2 Méthodes d'évaluation 141
A.2.1 Arbres des causes 142
A.2.2 Analyse de Markov ou Méthode de l'espace d'état 143

Annexe B : Bases des réseaux de Petri 144
B.1 Définitions de base 144
B.2 Méthodes d'analyse 146
B.2.1 Simulation 146
B.2.2 Graphe de marquage 147
B.2.3 Algèbre linéaire 147
B.2.4 Méthodes de réduction 148
B.3 Propriétés de base 149
B.4 Extensions des réseaux de Petri 151
B.4.1 Réseaux de Petri généralisés 153
B.4.2 Réseaux de Petri à capacité 154
B.4.3 Réseaux de Petri à prédicats 154
B.4.4 D'autres extensions 155

Introduction

Ce travail s'inscrit dans la thématique de l'évaluation de la sûreté de fonctionnement de systèmes d'intelligence distribuée. Par essence, la problématique se trouve à l'intersection de plusieurs champs disciplinaires traditionnels : les problèmes de modélisation et d'évaluation des performances tels que les pratiquent les automaticiens, l'évaluation de la sûreté de fonctionnement qui est souvent appliquée, avec des méthodologies reconnues, sur des systèmes où les aspects dynamiques sont peu pris en compte, et enfin les sciences et technologies des réseaux et de la communication, où l'essentiel des recherche se focalise sur l'assurance de la bonne réalisation de la fonction communication.

Ce travail se propose, dans le contexte décrit dans le paragraphe précédent, d'aborder et de proposer des pistes qui permettront d'avoir une approche globale, permettant d'évaluer des systèmes d'automatisation à intelligence distribuée, ou des systèmes à réseaux (SAR, systèmes d'automatisation à réseaux, "networked control systems") ; ces systèmes sont composés, dans notre hypothèse, de capteurs et actionneurs intelligents, c'est-à-dire dotés de capacités de traitement et d'interfaces de communication, organisés autour d'un réseau de communication, souvent un réseau de terrain.

Le but de ce travail est de proposer une approche d'étude des systèmes d'automatisation à réseau en gardant un formalisme unique toute au long de l'étude et en étudiant ces systèmes à la fois du point de vue de la commande et du point de vue de la sûreté de fonctionnement tout en gardant l'aspect temporel dans les modèles pour exprimer des phénomènes tels que le vieillissement des composants ou l'encombrement du canal de communication.

Dans la suite de travaux précédent, décrits dans le chapitre 1, notamment les travaux de Blaise CONRARD qui proposait une approche d'évaluation d'architectures distribuées a priori en phase statique, sont inclus ici des aspects liés à la dynamique du système, à la fois pour prendre en compte les évolutions du système au cours du temps, son historique, son vieillissement, mais aussi les divers modes de fonctionnement et configurations par lesquels passe le système au cours de son fonctionnement.

Les chapitres 1 et 2 de la thèse situent le contexte du travail, en définissant la problématique en regard de travaux passés ou en cours, les spécificités et caractéristiques des systèmes d'automatisation

distribués et systèmes d'automatisation à réseaux, un état des lieux de méthodes d'évaluation de la sûreté de fonctionnement et une discussion sur l'utilisation des réseaux de Petri dans des évaluations de la sûreté de fonctionnement ; en effet, les extensions stochastiques des réseaux de Petri permettent d'inclure une dimension probabiliste, nécessaire à l'évaluation de la sûreté de fonctionnement. Le chapitre 2 se termine par des considérations sur la simulation, notamment la simulation de Monte Carlo, ses possibilités et ses limites.

Le Chapitre 3 développe les Réseaux de Petri colorés, qui sont l'outil de modélisation qui a été choisi puisque répondant potentiellement à nos questions, à savoir pouvoir modéliser un système dans lequel apparaissent généralement des variables à temps continus (en général discrétisées) et des événements discrets, pouvant survenir de manière aléatoire. Après la définition des RdP colorés, nous présentons l'outil Design-CPN, proposé par l'Université de Århus (Danemark) et le langage ML standard qui en fait une partie. Cette partie permet ensuite de mieux comprendre l'élaboration des modèles.

Le chapitre 4 commence par un tutoriel de modélisation des composants d'automatisation dans le logiciel Design/CPN. Divers modèles de base du système d'automatisation à intelligence distribuée sont présentés, notamment le système discrétisé, un capteur "intelligent" communicant et potentiellement défaillant, et un canal de communication. Ces composants de base sont ensuite utilisés dans les modèles plus complexes proposés dans les chapitres 5 à 7.

Les chapitres 5 et 6 illustrent l'utilisation de la méthode sur un système, concrètement une cuve dont le remplissage est commandée via un réseau. La première évaluation proposée dans le chapitre 5 montre comment par l'utilisation qui est faite du système, l'occurrence d'un événement peut entraîner un changement de mode de fonctionnement qui se traduit ensuite par un vieillissement ou une dégradation différente des composants du système. La conséquence est que la mise en panne d'un composant ou la perte de sécurité d'une fonction, peut se produire de manière tout à fait différente de ce qui avait été initialement prévu et prévisible. Le chapitre 6, dans la même idée, illustre l'évolution du système d'automatisation pendant son cycle de fonctionnement ; cette application rassemble les résultats de l'étude de scénarios d'événements en vue de l'identification des scénarios redoutés. Enfin la dernière application, présentée dans le chapitre 7 est un peu en marge ; elle propose l'utilisation de réseaux de Petri à des fins de vérification des propriétés d'un réseau de terrain de sécurité, en l'occurrence AS-i Safety at Work, en regard des spécifications données par la norme.

Deux annexes complètent cette thèse. La première contient des rappels de base du domaine de la sûreté de fonctionnement. La deuxième présente les Réseaux de Petri, leurs propriétés et leurs extensions.

1 Systèmes distribués

1.1 INTRODUCTION GENERALE

Cette thèse étudie les systèmes d'automatisation à intelligence distribuée qui sont évalués du point de vue de leur sûreté de fonctionnement. Un accent particulier est mis sur l'étude des systèmes d'automatisation à réseau à cause de leur forte émergence actuelle mais également à cause de la possibilité d'appliquer les méthodes d'études des systèmes d'automatisation à réseau sur des systèmes d'automatisation à intelligence distribuée.

Les systèmes d'automatisation à intelligence distribuée (SAID dans la suite) sont une extension des systèmes d'automatisation. Un système d'automatisation est un ensemble de composants organisés pour accomplir une mission en minimisant les interventions humaines. La mission du système d'automatisation est exercée sur un autre système, dit processus sous contrôle (processus dans la suite). Le système d'automatisation et le processus forment ensemble un système automatisé.

Les principaux composants d'un système d'automatisation sont :

- les capteurs, servant à déterminer l'état actuel du processus,
- les régulateurs avec le rôle d'établir une nouvelle commande à appliquer au processus,
- les actionneurs destinés à appliquer la commande au processus.

Les liaisons de ces composants peuvent être assurées de plusieurs manières. Une des plus simples est la boucle 4 – 20 mA. Il s'agit d'une connexion analogique où la plus petite des valeurs à transmettre est associée avec 4 mA et la plus grande avec 20 mA. D'autres moyens de communication (dont les réseaux de communication) peuvent être présents et seront discutés dans la suite.

Un système d'automatisation à intelligence distribuée est constitué de composants dotés d'une intelligence. Par intelligence nous comprenons une capacité de traitement de l'information associée à la possibilité de communiquer cette information aux autres composants. Si un réseau de communication est utilisé pour transmettre cette information, nous parlerons alors de systèmes d'automatisation à

réseau (SAR, « networked control systems », [Wal 01]). Les réseaux de terrain sont le plus couramment utilisés dans ces systèmes.

[Fen 02] justifie l'évolution vers des systèmes d'automatisation à réseau par l'amélioration de l'efficacité, de la flexibilité et de la fiabilité en réduisant les temps et les coûts de l'installation, de la reconfiguration et de la maintenance.

La figure 1.1 présente un exemple de SAR qui pour assurer la commande de niveau du liquide dans la cuve est composé de :

- un capteur analogique de niveau,
- deux capteurs binaires,
- un régulateur,
- un actionneur,
- un réseau de communication,
- d'autres composants pouvant générer un trafic sur le réseau de communication.

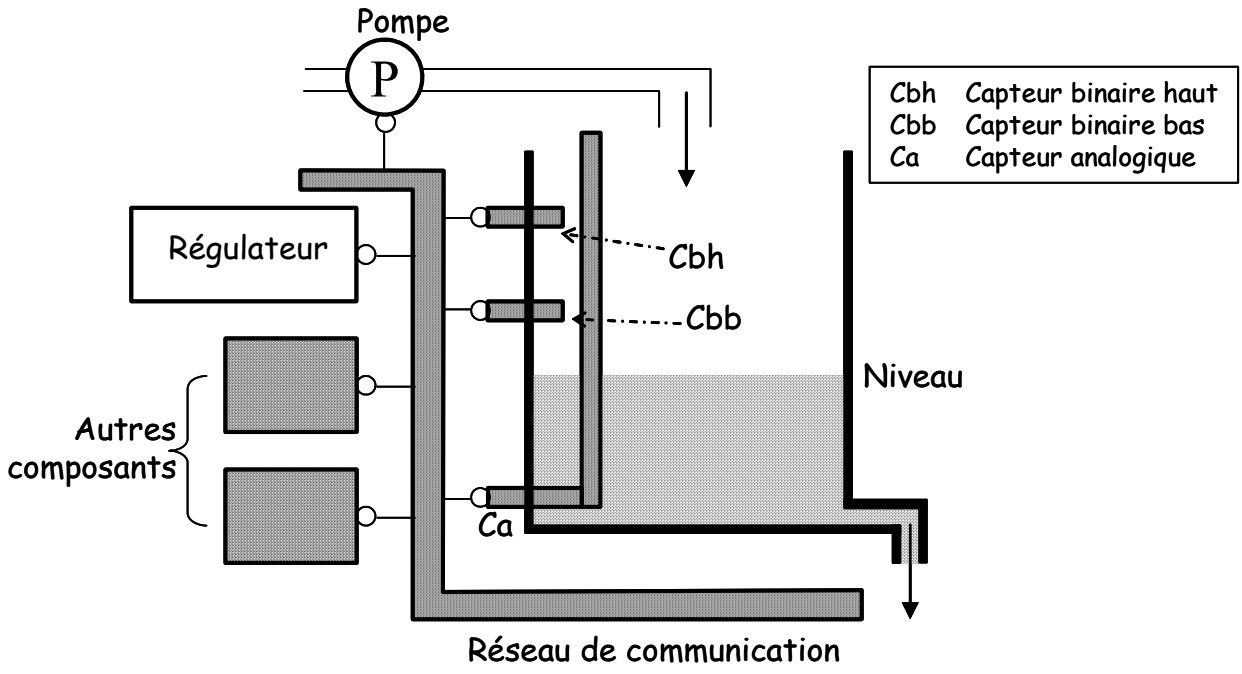


Figure 1.1 : Exemple d'un système d'automatisation à réseau

1.2 POSITIONNEMENT DU PROBLEME

Plusieurs propriétés caractérisant les SAID en général peuvent être observées, dont :

1.) leur composition hybride, qui se traduit par la cohabitation de sous-systèmes continus et/ou échantillonnés et de sous-systèmes à événements discrets,
2.) leurs aspects dynamiques intégrant les changements dynamiques de leur architecture, par exemple la reconfiguration,
3.) l'intégration des fonctionnalités de l'automatique avec la mission orientée processus et des fonctionnalités informatiques avec la mission orientée données.

Ces 3 points sont développés dans la suite.

1.2.1 Systèmes hybrides

Un système automatisé est un ensemble hybride [Rie 02] avec des sous-systèmes continus et/ou échantillonnés et des sous-systèmes à événements discrets. Cette dualité peut être retrouvée même au sein d'un composant unique. Par exemple un capteur logique qui envoie périodiquement sa mesure peut au dépassement d'un seuil envoyer une information sur ce dépassement. Puisque ce dépassement n'arrive pas forcément à l'instant de l'échantillonnage, cette information est générée d'une manière asynchrone par rapport à la période d'échantillonnage.

Il devient alors évident qu'un appel aux techniques des modélisations hybrides peut être nécessaire. Les réseaux de Petri en font partie. Néanmoins pendant la conception des modèles nous avons voulu vérifier l'exactitude de nos modèles des parties échantillonnés. Pour cela nous les avons confronté aux résultats de simulation de Scilab et Matlab/Simulink. Mais on ne peut pas oublier que ces deux outils ont une utilisation restreinte pour la modélisation des systèmes hybrides. Pendant la thèse ils sont alors exclusivement utilisés pour vérifier les parties synchrones des modèles et également pour visualiser des résultats des simulations RdP à partir des fichiers sauvegardés pendant la simulation.

1.2.2 Systèmes dynamiques

L'aspect dynamique est très important dans les SAID. La configuration logique et matérielle peut être très variable au cours de leurs cycles de vie. Les changements par rapport à l'état initial peuvent être très rapides, tels que l'arrêt de fonctionnement d'un composant, par exemple sa casse, ou plutôt lents comme la diminution de vitesse d'action pour un actionneur qui subit une usure.

L'importance de l'aspect dynamique implique le besoin d'études dynamiques du comportement de ces systèmes, puisque les études statiques se montrent insuffisantes. Les méthodes classiques de leurs études, notamment dans le domaine de la sûreté de fonctionnement, tels que la méthode de l'arbre des causes ne permettent pas une prise en compte suffisante de phénomènes tels que les liaisons entre les défaillances, la reconfiguration ou l'altération des propriétés au cours du temps (ex. : vieillissement).

1.2.3 Systèmes réunissant l'automatique et l'informatique

Dans les SAR on retrouve un autre aspect hybride important. On y observe la présence d'une double base de temps. D'un côté sont rassemblés les phénomènes de l'automatique avec des constantes de temps caractérisant les composants, c'est-à-dire les dynamiques lentes. A l'opposé se trouvent les contraintes temporelles associées au traitement des données, tels que le filtrage et la mémorisation. La vitesse de transmission sur le réseau de communication appartient également à ce groupe. L'avantage de notre approche est qu'elle permet d'exprimer grâce aux horloges multiples présentes dans le modèle l'expression distincte de ces deux bases de temps.

L'ensemble des études présentées dans notre travail traite les problèmes que pose la présence d'un réseau de communication dans un système de régulation. Il ne s'agit nullement de l'étude de systèmes de gestion du réseau de communication, comme s'est souvent le cas quand les deux mots clés : *réseau* et *commande*, sont présents ensemble.

L'objectif premier de la conception d'un système SAR est de faire un usage efficace de la capacité finie du réseau de communication en maintenant une bonne performance de la boucle de régulation y compris la stabilité, le temps de montée, le dépassement et d'autres critères [Wal 99]. Ajoutons ici qu'un « bon » système doit avoir un niveau de sûreté de fonctionnement correspondant.

Le but de ce travail est d'étudier les SAR dans leur globalité sans se focaliser sur les composants indépendants mais plutôt sur leurs interactions mutuelles. Pour une étude sur les réseaux de communication à utiliser dans les SAR, [Lia 01] présente une comparaison de trois réseaux utilisables dans les SAR : Ethernet, ControlNet et DeviceNet. L'étude de l'Ethernet dans une application d'automatique est présentée dans [Ota 02a]. Des travaux sur la conception d'un régulateur SAR notamment en vue de sa stabilité sont fortement émergents et une synthèse peut être trouvée dans [Ota 02] ou [Lia 02]. L'analyse spectrale de la sortie du SAR est décrite dans [Lin 02].

Les avantages de tels systèmes par rapport aux systèmes traditionnels avec une connexion câblée point à point sont indiscutables. Ne citons que quelques uns d'entre eux :

- réduction du câblage,
- augmentation de la modularité, flexibilité, reconfigurabilité,
- potentiel de communication des informations sur la supervision ou le diagnostic.

Néanmoins, la présence d'un réseau de communication complexifie l'analyse et la conception des SAR [Zha 01]. Plusieurs facteurs influencent le fonctionnement d'un SAR :

1. Le délai de transmission de données. Ce délai peut être constant (avec ou sans gigue) pour les réseaux déterministes tels que AS-i ou variable dans le temps comme c'est le cas d'Ethernet. L'impact de ce délai peut considérablement modifier le comportement du système par rapport à la référence. Il peut dans certains cas déstabiliser le SAR.
2. Le réseau de communication peut être vu comme un ensemble de chemins potentiellement défaillants. Il faut donc considérer l'impact d'une éventuelle perte d'une trame de communication.
3. Une information peut être pour des raisons propres à la spécification du réseau de communication divisée en plusieurs trames. Il faut alors prendre en compte qu'à l'arrivée une partie des trames peut être perdue.
4. Dans certains cas l'ordre d'arrivée des trames peut être différent de l'ordre de leur émission.

Lors de la conception d'un SAR il faut prendre en compte non seulement les caractéristiques du processus sous contrôle ainsi que l'ensemble des éléments de commande mais aussi les propriétés du réseau de communication, notamment le taux de transmission, le protocole d'accès au médium (MAC), la longueur des trames, etc. Deux approches principales existent :

1. Concevoir le système d'automatisation sans prise en compte du délai de transmission ni du taux de pertes de trames et utiliser le protocole de communication qui minimise de tels événements.

2. Introduire explicitement les mécanismes de traitement de ces événements perturbateurs directement dans la conception des algorithmes de commande.

La meilleure façon d'améliorer la performance d'un SAR est de réduire le volume des informations transmises [Ota 02]. [Wall 99] propose dans ce but un protocole de communication appelé TOD (Try-Once-Discard) qui n'envoie que les dernières valeurs quitte à parfois perdre une valeur si sa transmission est tardive. Cette communication introduit également un algorithme d'ordonnancement de messages dynamiques sous le nom « Maximum-Error-First » ce qui signifie que le noeud gagnant est celui dont l'écart pondéré de la dernière valeur transmise est le plus important.

1.3 METHODES D'ETUDES DES SAR

Comme c'est dit auparavant, les systèmes d'automatisation à réseau sont une extension des systèmes d'automatisation distribués classiques. La différence principale est la présence d'un réseau de communication, ce qui se traduit par l'intégration d'un nouvel élément en interaction constante avec tous les autres composants du système en question. Ainsi, on peut dire que le réseau de communication devient un élément central. Pour cette raison, les aspects de la sûreté de fonctionnement doivent être examinés avec une diligence particulière [Jua 02]. Ces études portent souvent sur le réseau de communication lui-même dans le cadre d'étude de performances. Les aspects sûreté de fonctionnement n'ont alors qu'une importance secondaire. La probabilité d'altérations de bits est l'aspect le plus souvent mentionné.

Différentes approches sont utilisées dans la littérature pour étudier les systèmes SAR. On peut les classer dans deux catégories :

1.) Celles qui établissent un modèle d'un SAR complet y compris le réseau de communication, avec une vision système.
2.) Celles qui ne modélisent que le réseau de communication et prennent en compte les données transportées, avec une vision centrée réseau.

[Wal 99] fait partie de la première catégorie. Une boucle complète de régulation est considérée pour un système multi-entrée multi-sortie. Les dynamiques du processus sous contrôle et du régulateur sont décrites comme des systèmes continus d'équations d'états. Le réseau adopté suppose la diffusion de

toutes les informations et donc l'état du processus sous contrôle est connu à tous les éléments en permanence. Ni le capteur, ni l'actionneur ne sont considérés explicitement. [Bus 98] compare ce système à un système hybride.

Pour l'analyse [Wal 99] utilise la simulation de Monte Carlo à cause de la nature aléatoire du trafic sur le réseau de communication. 3200 simulations ont été effectuées pour plusieurs valeurs d'un paramètre τ, dit l'intervalle maximal de transfert admissible. C'est le paramètre qui résume en lui-même la présence du réseau de communication.

Le problème de la mise en place de l'algorithme de commande dans un système à réseau est abordé dans [Lig 03] et l'influence de méthodes d'initialisation des composants fait l'objet de [Lig 04]. Dans [Wal 99] le régulateur et l'actionneur sont combinés en une seule entité. Le réseau de communication n'est présent qu'entre le capteur et le régulateur/actionneur. Un modèle interne du processus (équations d'états continues) est également incorporé dans le régulateur/actionneur. Il sert à assurer la commande pendant l'absence de nouvelles mesures du capteur. Trois hypothèses principales sont faites et les deux dernières concernent le réseau de communication :

1. Le modèle « compensé » de l'installation est stable,
2. Le délai de transmission est négligeable,
3. La fréquence de transmission des mesures est constante.

La deuxième hypothèse est levée pour une partie de l'étude mais la période d'échantillonnage reste toujours supérieure au délai de transmission, les deux sont toujours constants. Sous ces conditions la simulation de Monte Carlo n'est pas nécessaire, une simulation du fonctionnement suffit et est utilisée dans la partie d'analyse de la communication.

L'approche proposée dans [Bra 03] est différente des deux précédentes. Il fait usage de la co-simulation. La partie transmission est simulée dans l'outil *ns-2* (l'outil interne des auteurs) et la partie opérative dans un solveur ODE. L'objectif de l'étude est de déterminer si une certaine architecture de réseau et autres composants de commande permettent d'obtenir la sortie du processus sous contrôle proche de la trajectoire de référence. Le but de la co-simulation est de permettre de traiter et d'étudier les propriétés d'un réseau de communication telles que la bande passante, la fiabilité ou le délai, simultanément avec les caractéristiques de la boucle de commande : la stabilité, la robustesse ou la performance. La modélisation d'un SAR est séparée en trois parties utilisant trois formalismes différents :

1. La dynamique des agents (éléments physiques ou virtuels) qui sont des systèmes dynamiques hybrides influencés a.) directement par les états ou les sorties continus ou discrets des autres agents, b.) indirectement par la réception d'une information sur l'état ou la sortie d'un autre agent par le réseau de communication, c.) par des événements ou des perturbations stochastiques internes ou externes.
2. Le flux d'information dans le réseau est représenté comme un graphe orienté qui résume les relations d'informations détaillées dans les équations dynamiques.
3. La topologie du réseau est un multigraphe orienté et coloré. Ce graphe contient les ressources de communication disponibles. Il s'agit bien d'un multigraphe puisque deux nœuds peuvent être connectés par plusieurs liaisons directes.

L'approche proposée permet la prise en compte de réseaux de communication sophistiqués et de les exprimer dans un outil bien connu (*ns-2*). L'inconvénient est le besoin d'utiliser deux formalismes différents ce qui pourrait nuire aux méthodes d'analyse formelle de vérification de sécurité de l'installation.

[Fen 02] propose de développer le modèle SAR à l'aide de Matlab/Simulink. Pour cela il divise le modèle en deux parties : un simulateur du réseau et un simulateur de commande. La partie réseau est exprimée dans le NCsimulator (logiciel propres aux auteurs) et ensuite incorporée dans le modèle de la partie opérative, développée quant à elle sous Matlab/Simulink. L'avantage de cette approche est la possibilité de prendre des mesures réelles des caractéristiques du réseau et de les insérer dans le modèle pour améliorer sa précision.

La modélisation distincte de la partie opérative et du réseau de communication peut être également trouvée dans [Pat 03]. Il propose d'abord de modéliser la partie opérative avec des équations d'état. Les capteurs et les actionneurs défaillants y sont intégrés ainsi que les interfaces de communication. Le protocole concerné est le TTP/C et il est modélisé comme une séquence de communications individuelles. Pour faire cohabiter les deux parties du modèle on utilise les branchements (switching).

Le même protocole de communication est adressé dans [Pap 01]. Il s'agit d'une étude portant sur le système de freinage de voiture. La méthode d'étude s'appelle Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) et elle est basée sur les arbres des causes, l'analyse des défaillances fonctionnelles et l'analyse de modes de défaillances et d'effets. L'approche consiste à concevoir un modèle de système hiérarchique, puis à effectuer une analyse SdF grâce à l'analyse des

défaillances fonctionnelles. Pour assurer la liaison fiable entre différents niveaux hiérarchiques on propose d'utiliser l'analyse des modes de défaillances et d'événements avec une extension focalisée sur les interfaces. L'étape finale de l'étude est la détermination de propagation des défaillances dans le système. Dans la méthode HiP-HOPS, cette étape est automatisée avec l'aide d'un algorithme systématique pour la synthèse des arbres des fautes.

Une approche mixte est utilisée dans [Blu 98]. La boucle de régulation est d'abord étudiée en remplaçant le réseau de communication par un bloc à retard pur et sa marge de phase est déterminée pour garantir la stabilité de la boucle. Le retard est ensuite étudié avec un modèle RdP stochastiques et puis réintégré dans l'expression de la boucle de régulation.

Une seconde catégorie d'approches de modélisation SAR sont des modèles du réseau de communication.

Des modèles pour la représentation formelle du comportement d'activités de communication doivent avoir au minimum les attributs ci-dessous en termes de pouvoir d'expression et de pouvoir d'analyse [Jua 95].

1.) Le pouvoir d'expression doit permettre :
   a.) L'expression du parallélisme et de la synchronisation par échange de messages,
   b.) L'expression de la synchronisation par contraintes temporelles (cet attribut prend de plus en plus d'importance, compte tenu du développement actuel des systèmes distribués temps réels).

2.) Le pouvoir d'analyse doit permettre :
   a.) Une analyse qualitative, c'est-à-dire l'étude des propriétés « logiques » du comportement d'une architecture de communication ; en particulier, il est essentiel de pouvoir caractériser la synchronisation : a.) par échange de message ; b.) par contrainte temporelle.
   b.) Une analyse quantitative, c'est-à-dire l'évaluation de performances en fonctionnement normal et en fonctionnement anormal (sûreté de fonctionnement) ; cette analyse peut se faire en termes de bornes ou de valeurs moyennes.

1.4 PRESENTATION DE LA METHODE

La méthode retenue pour notre étude des SAID peut se résumer en deux étapes :

1.) modélisation,
2.) exploitation des modèles.

La construction des modèles de réseaux de Petri colorés (CPN, 'colored Petri nets', [Jen 97]) commence par la création des modèles fonctionnels des composants. Ceci intègre l'expression de leur évolution dynamique en modes de fonctionnement nominaux, entre autre la transmission d'une trame de communication ou encore l'évolution des variables d'état d'un système échantillonné.

Parallèlement aux modèles fonctionnels, il est nécessaire de développer les parties dysfonctionnelles. Ainsi il faut exprimer les différents modes de défaillance pris en compte pour chaque composant du SAID. L'exemple de cette partie est la modélisation de l'usure de l'actionneur en fonction de son activité.

Le pas suivant de la méthode est l'intégration des parties fonctionnelles et dysfonctionnelles dans un seul modèle. Il s'agit d'une étape délicate puisque les deux parties doivent fonctionner simultanément et en aucun cas l'exécution d'une fonctionnalité ne peut nuire à l'exécution de l'autre. Exemple : Un actionneur doit fonctionner en même temps que ces propriétés comme l'usure ou le vieillissement sont modifiées.

L'interconnexion des différents composants est la dernière marche dans l'étape de la modélisation. Une attention doit être portée aux places partagées entre plusieurs composants, notamment à celles qui relient des composants échantillonnés aux composants réagissant sur événement. Il faut impérativement assurer un fonctionnement simultané de tous les composants. Cette phase de conception est spécialement évoquée dans le sous-chapitre traitant des Places – Tampons.

Après avoir conçu le modèle CPN d'un SAID et également celui du processus une analyse peut être faite. Pour cela il est d'abord impératif de définir des critères d'évaluation. Pour étudier les aspects de la SdF il faut alors définir la notion de panne de système. Elle fait naturellement partie des événements significatifs. D'autres événements de cette classe peuvent être des défaillances des composants, des dépassements des seuils, des instants de changements de mode de commande, etc. Il s'avère que même

pour des systèmes relativement simples, le nombre d'événements à prendre en compte peut être assez conséquent.

Les critères à évaluer du point de vue dynamique ne sont pas clairement formulés dans la littérature. Il faut donc souvent que le concepteur en propose des appropriés pour donner la bonne signification à son analyse. Nous sommes actuellement en train d'évaluer l'influence des différentes définitions d'une stabilisation de processus sur son évaluation globale.

Les modèles sont ensuite analysés par une simulation de Monte Carlo. Elle est abordée dans le premier chapitre.

Les modèles CPN peuvent être analysés également par un graphe d'occurrence. Sa construction est automatique et directement intégrée dans l'outil Design/CPN. Mais à l'état actuel, l'explosion combinatoire beaucoup trop importante empêche toute analyse significative des modèles. Différentes pistes sont explorées et celle de regroupement d'états vers des classes se montre relativement prometteuse.

1.5 CONCLUSION

Cette étude bibliographique a montré une forte émergence de différentes études dans le domaine des systèmes distribués. Tout d'abord, ceci peut être considéré comme une migration des études classiques sur les systèmes distribués vers ceux qui intègrent les réseaux de communication. L'étude de la sûreté de fonctionnement de ce type de système relève pour l'instant de deux communautés : celle des automaticiens et celle des informaticiens. Sur ce type de systèmes complexes, ces communautés ont tendance à se rapprocher mutuellement avec pour but l'étude du système dans son ensemble. Ce nouvel axe de recherche nécessite l'intégration de méthodes de travail communes et les RdP en sont incontestablement une des possibilités. Leur utilisation est en effet généralement acceptée par les deux communautés.

Un aspect des systèmes distribués est récurrent dans cette recherche. Il s'agit de la prise en compte de l'aspect dynamique du problème. Nous verrons dans le chapitre suivant que ce problème du comportement dynamique se pose et des solutions commencent à émerger dans le domaine de la sûreté

de fonctionnement, notamment dans le domaine de la fiabilité dynamique. On retrouve ici une synthèse bénéficiant des avancées dans le domaine des systèmes distribués et dans la SdF en général pour une meilleure intégration des propriétés temporelles des systèmes dès leur conception.

2 Sûreté de fonctionnement

2.1 INTRODUCTION

Parallèlement au développement de la distribution des systèmes, la Sûreté de Fonctionnement a été amenée à formuler les problèmes liés à cette nouvelle caractéristique et à évaluer l'influence de la distribution sur les propriétés du système distribué. Selon [Dai 03] la fiabilité d'un service distribué est la probabilité d'assurer avec succès le service dans un environnement distribué.

Le travail présenté s'inscrit dans la ligne du travail de la thèse de Blaise Conrard [Con 99] effectué au CRAN entre 1996 et 1999. L'extension principale est la prise en compte des aspects dynamiques dans la considération des systèmes d'automatisation.

[Con 99] traite la question de l'influence des composants dits intelligents et de réseaux de terrain sur les critères de la sûreté de fonctionnement. Il présente un avis sur le besoin d'intégration de ces critères dès la phase de la conception des systèmes. La thèse établit également une liaison entre la sûreté de fonctionnement et la rentabilité. L'estimation est faite a priori.

Pour pouvoir établir les modèles deux architectures sont utilisées : fonctionnelle et matérielle. L'avantage de cette séparation consiste en la facilité d'expression des phénomènes tels que la redondance.

La modélisation fonctionnelle est divisée en :

- méthodes orientées fonctions, dont SADT et SART,
- méthodes orientées données, dont le diagramme entité – association,
- méthodes orientées objets.

La méthode proposée consiste à :

1.) modéliser l'architecture fonctionnelle du système,
2.) l'enrichir par des informations relatives à la sûreté de fonctionnement,

3.) établir une architecture matérielle préliminaire,
4.) faire la projection de l'architecture fonctionnelle sur l'architecture matérielle et optimiser cette projection.

Cette méthode permet de prendre en compte différentes caractéristiques des composants dits intelligents. Son point le plus fort est l'optimisation des choix de composants en maximisant des critères de la sûreté de fonctionnement aussi bien que ceux de la rentabilité. Néanmoins son inconvénient majeur est sa limitation aux caractéristiques statiques dans le temps. Alors des propriétés comme le vieillissement ou le changement de mode d'utilisation ne peuvent pas être considérées dans l'étude.

D'autres travaux effectués au laboratoire servent également de point d'appui. [Mor 01] présente la liaison entre le domaine de la sûreté de fonctionnement et les systèmes de production, notamment leurs politiques de maintenance. [Web 01] analyse également la maintenance des systèmes de production à l'aide des réseaux Bayesiens.

2.2 LIMITES DE LA METHODE DE L'ARBRE DES CAUSES

Les méthodes classiques de l'étude de la SdF ne sont pas adaptées pour ces nouveaux phénomènes. Elles se heurtent à leurs limites de pouvoir d'expression et de capacité d'analyse. Une des méthodes les plus répandues est la Méthode de l'Arbre des Causes (MAC). Cette méthode a fait largement ses preuves dans le domaine de la SdF classique.

Cette réflexion n'a pas pour but de lister toutes les difficultés si la méthode MAC devrait être appliquée pour l'analyse des systèmes SAR. Son objectif est d'argumenter sur quelques exemples pratiques les limites de la méthode. Le problème de modélisation d'une défaillance du réseau de communication est utilisé pour illustrer deux aspects problématiques : les conséquences diverses d'un seul événement (causes communes) et l'historique d'apparition des événements.

Il n'est pas aisé d'intégrer la défaillance de transmission du réseau de communication dans une étude. Il s'agit bien d'une défaillance intermittente dont la durée peut être comparable à la durée de transmission d'un bit ou supérieur. La probabilité d'occurrence pour un tel événement est dans la plupart des cas précisée par le constructeur du réseau de communication. Elle pourrait alors être considérée comme

l'événement élémentaire ne nécessitant pas de développement futur et être représentée par un cercle dans l'arbre des causes. Le problème rencontré subsiste dans les différents effets produits par cet événement. En fait, ces conséquences sont déterminées selon l'information endommagée. Dans certains cas cette défaillance peut être négligée, comme par exemple si le message transmis ne contient pas de données liées à la sûreté de fonctionnement, et dans d'autres cas elle peut être la cause d'un arrêt intempestif de l'installation, par exemple s'il s'agit d'un bit de synchronisation du réseau.

Supposons à présent la possibilité de distinguer pendant la conception de l'arbre des causes le classement de toutes les défaillances du réseau selon le contenu endommagé, ce qui est en réalité une tâche particulièrement difficile. Le deuxième problème vient de l'historique du vécu du système. En fait, les mêmes défaillances du réseau ne produisent pas forcément toujours les mêmes effets. Ceci dépend en partie de la présence ou de l'absence du même événement ou d'un autre événement pertinent dans le passé plus ou moins proche de l'instant d'occurrence. Encore une fois, ce phénomène, proche de la fiabilité dynamique, reste difficilement exprimable dans un arbre des causes.

Il est bien connu que la méthode MAC n'est pas adaptée aux systèmes dont les défaillances ne sont pas indépendantes. [Vill 88] propose pour de tels systèmes l'utilisation de méthodes comme la Méthode de l'Arbre des Conséquences (MACQ) et la Méthode de l'Espace des Etats (MEE). Une brève description des méthodes de base de la SdF se trouve dans la première partie de l'annexe de cette thèse.

2.3 SURETE DE FONCTIONNEMENT DES COMPOSANTS INTELLIGENTS

La présence des instruments intelligents peut être très bénéfique pour l'amélioration de la sûreté de fonctionnement des systèmes d'automatisation. Mais à la différence des systèmes électromécaniques pour lesquels les mécanismes de défaillances sont relativement bien connus, les modes de défaillances de la partie intelligente sont plus difficile à prévoir. [Gar 02] présente une étude de l'influence de la partie intelligente sur la sûreté de fonctionnement du système complet. L'ensemble du système est modélisé avec les graphes de flux dynamiques (Dynamic Flowgraph Methodology). L'analyse d'un tel modèle se fait généralement en retraçant les séquences d'événements à partir des effets vers leurs causes. C'est une démarche analogique à la méthode des arbres des causes avec l'extension vers des systèmes logiques non booléens.

[Bat 00] présente une étude de la fiabilité des composants d'une installation pilote. Pour cela la méthode des arbres des causes est développée. Il s'agit d'une application simple modélisant l'ensemble du système et une partie de ces composants, notamment un actionneur. La redondance des composants est également considérée.

2.4 SURETE DE FONCTIONNEMENT DYNAMIQUE

Pour étudier les aspects de la SdF des systèmes distribués de nombreuses méthodes sont disponibles : l'AMDEC, les arbres de défaillances, les graphes de Markov. La plupart d'entre elles se heurtent de plus en plus à leurs limites. En effet, ces méthodes ne sont pas adaptées à prendre en compte les aspects dynamiques des systèmes tels que les reconfigurations, les modes dégradés ou encore les dépendances très variées des défaillances. Néanmoins, ces méthodes restent utiles puisque non remplacées [Leb 02]. Leur application est toujours valable si on s'intéresse à l'aspect statique d'études de la sûreté de fonctionnement.

Plusieurs nouvelles méthodes ont fait leur apparition pour traiter les problèmes en dehors du champ des caractéristiques statiques. Elles font partie d'un ensemble appelé 'fiabilité dynamique' [Lab 00].

La fiabilité dynamique est un domaine d'évaluation probabiliste de risque qui étudie le comportement des systèmes hommes-machines-logiciels sous influence de la dynamique sous-jacente d'une façon intégrée [Smi 98].

Pour une large classe de modèles leur état de fonctionnement est déterminé par des occurrences de deux types d'événements [Duf 02] :

- le premier type d'événements est lié directement à l'évolution déterministe des variables physiques du processus,
- le deuxième type d'événements est purement stochastique. Il correspond habituellement aux requêtes aléatoires ou aux défaillances des composants.

La simulation de Monte Carlo et les Réseaux de Petri hybrides sont deux méthodes proposées pour l'étude de la fiabilité dynamique par [Duf 02]. L'autre approche proposée est le calcul analytique du modèle de Markov.

La solution des problèmes de la fiabilité dynamique demande l'utilisation de techniques numériques. Sa difficulté vient de l'importance du nombre d'états et d'un nombre potentiellement élevé de transitions entre les états [Smi 98]. La simulation de Monte Carlo est une voie de solution, mais malgré son insensibilité aux dimensions de problèmes traités elle peut être particulièrement lente à converger.

2.5 SURETE DE FONCTIONNEMENT ET RESEAUX DE PETRI

Comme il est en partie montré dans le chapitre précédent, les méthodes classiques d'évaluation de la sûreté de fonctionnement ne sont pas bien adaptées pour l'application aux systèmes d'automatisation à intelligence distribuée, notamment par les difficultés à exprimer les caractéristiques temporelles et dynamiques. C'est pourquoi nous avons examiné la façon d'utiliser les réseaux de Petri à travers toutes leurs extensions aux études liées à la sûreté de fonctionnement. Une méthode de traduction d'un arbre des causes vers les RdP peut être trouvée dans [Kne 01].

[Mor 02] et [Mih 02] précisent que les RdP se distinguent des autres méthodes d'étude de la SdF. En effet, ils peuvent être employés tout au long du cycle de développement d'un processus : on peut ainsi conserver le même formalisme pour comprendre l'architecture et le comportement d'un processus (au même titre que l'analyse fonctionnelle) mais aussi pour modéliser ses comportements défaillants (comme les graphes de Markov ou les arbres de défaillance). Cette polyvalence est encore accrue par leur représentation à la fois graphique et mathématique. Le premier formalisme facilite la compréhension et le dialogue entre les concepteurs et les ingénieurs SdF, alors que le second possède un pouvoir d'expression des méthodes formelles.

[Che 96] propose une synthèse de l'utilisation des RdP dans le domaine de l'ingénierie des protocoles ainsi que les relations entre les RdP et d'autres formalismes tels que SDL, ESTELLE, LOTOS.

L'usage des RdP a fait ces preuves aussi dans [Kam 99] pour le modèle d'une gestion décentralisée de plusieurs réseaux de télécommunications. Il s'agit d'une application ADLN pour laquelle une disponibilité totale est calculée grâce au passage des RdP vers la chaîne de Markov.

2.5.1 Utilisation des réseaux de Petri stochastiques

Les RdP stochastiques sont utilisés largement pour toutes sortes d'études sur les systèmes. [Jel 02] propose par exemple leur utilisation pour évaluer les performances d'une chaîne logistique en présence d'un environnement incertain. L'évaluation des performances est un des deux grands domaines de leur application. Le deuxième champ d'application est l'évaluation des caractéristiques de la sûreté de fonctionnement.

Il existe une méthode de passage automatique des Blocs Diagrammes de Fiabilité (BDF) vers un modèle RdP équivalent décrit dans [Sig 02]. L'intérêt principal de cette démarche est de pouvoir bénéficier de la puissance de modélisation des réseaux de Petri en partant de la spécification BDF. Un autre avantage est la possibilité de créer les modules génériques pour pouvoir les réutiliser.

Dans certains travaux, les modèles de réseaux de Petri sont utilisés pour servir comme base de génération des modèles de chaînes de Markov. C'est le cas de [Ram 02], qui utilise cette approche pour l'analyse de la fiabilité des systèmes avec des contraintes temporelles spécifiques qui influent sur la criticité des défaillances. La méthode est appliquée à l'étude des scénarios opérationnels de gestion de temps de vol.

Une autre façon de passer d'un modèle RdP vers les graphes de Markov est l'utilisation des résultats de la simulation de Monte Carlo. Cette approche est décrite dans [Leb 02]. Sur un exemple de commande de l'ouverture des portes de métro, on y présente la méthodologie de transformation d'un modèle SART vers les RdP. L'analyse se fait par la simulation de Monte Carlo grâce aux scénarios obtenus. Une étude fonctionnelle et dysfonctionnelle de vérification que le système fait tout et rien que ce qui est prévu fait également partie de cette communication qui se termine par une étude sur les difficultés d'analyse de taux de fiabilité réels par la simulation.

La simulation de Monte Carlo est utilisée aussi dans [Lar 02], qui sur l'exemple d'une station de pompage des gaz, présente une méthode d'évolution d'évaluation de performances vers l'évaluation de fiabilité. Il s'agit d'une extension d'un modèle RdP existant focalisé sur la performance du système avec les caractéristiques de la sûreté de fonctionnement.

Le problème de la politique complexe de maintenance est abordé dans [Châ 02]. Parmi les avantages de l'approche RdP cités dans cette communication, on peut mentionner notamment la possibilité d'intégrer

différents paramètres dans les modèles, ce qui permet de modéliser non seulement la politique de maintenance mais également le système de support logistique à la maintenance. L'autre avantage est la conception hiérarchique et modulaire de modèles, qui permet la réutilisation des sous modèles à plusieurs reprises.

[Wan 02] propose une méthode de modélisation d'un système de remplissage des cuves à l'aide des RdP hybrides non-colorés. Le modèle est composé hiérarchiquement et il inclut entre autre le modèle des cuves et des vannes assurant le transfert du liquide. Le capteur n'est pas considéré pour l'analyse de la SdF pour la raison qu'il est considéré comme trop peu défaillant. Les vannes peuvent être dans deux états : ouvertes ou fermées et les défaillances sont le collage (la vanne reste bloquée dans son état actuel) et la fermeture intempestive. Pour obtenir l'ensemble des scénarios redoutés, il propose l'ajout d'un injecteur de défaillances dans le modèle. Cette publication mentionne également l'analyse du modèle avec des arcs inhibiteurs grâce au calcul de l'équation fondamentale [Dav 97]. La limitation majeure de ce travail est la considération des composants binaires du type Tout-ou-Rien.

La proposition de [Kab 97] est de faire évoluer les RdP vers les E-nets, qui ont une possibilité de reproduire la nature dynamique du processus.

Les réseaux de Petri stochastiques sont également appliqués dans le domaine de la fiabilité dynamique [Châ 98] pour une analyse de débordement ou d'assèchement d'une cuve. Il s'agit d'un système hybride avec une partie continue (par exemple le niveau du liquide) et une partie discrète, notamment l'apparition des alarmes. Deux approches sont proposées :

- combinaison d'un modèle RdP stochastiques avec une méthode de calcul de la partie continue,
- utilisation des RdP hybrides pour modéliser l'ensemble du système.

Ce travail a été implémenté sur le logiciel MOCA-RP, qui a permis une analyse de Monte Carlo avec 1 million de simulations d'une durée de 1000 heures par simulation.

2.5.2 Utilisation des réseaux de Petri colorés

L'utilisation des RdP colorés dans la sûreté de fonctionnement reste minoritaire par rapport aux autres extensions. Néanmoins plusieurs communications sur le sujet existent. Leur utilisation est proposée par exemple dans les perspectives de travail de [Jel 02].

Selon [Son 03], qui utilise les RdP colorés pour la vérification des logiciels, les RdP colorés présentent de nombreux avantages comme le prototypage rapide ou la facilité de visualisation. Parmi les inconvénients, il faut nommer l'explosion combinatoire du graphe d'occurrence et la complexité des tâches liées à l'analyse des invariants. Il propose alors de compléter la méthode de RdP colorés avec la méthode de vérification des prototypes (Prototype Verification System) qui sera utilisée comme un outil de preuve mathématique.

[Mon 02] intègre les RdP colorés dans l'étude de conception d'un régulateur de graissage des roulements dans l'aérospatial. Les raisons de choix de RdP cités sont :

- L'amélioration de la lisibilité et de la compréhension des systèmes modélisés grâce à l'interface graphique,
- La proximité des caractéristiques discrètes du système de commande numérique avec le formalisme des transitions d'états,
- La possibilité d'obtenir une analyse formelle d'un modèle RdP.

Les propriétés suivantes sont étudiées :

- L'absence de l'état de blocage,
- La vivacité du système,
- Le comportement déterministe du RdP autonome correspondant.

2.6 ANALYSE PAR SIMULATION DE MONTE CARLO

On appelle méthode de Monte Carlo toute technique numérique de résolution de problèmes mathématiques qui utilise des nombres aléatoires ou des nombres pseudo-aléatoires [Dod 93].

La simulation dynamique montre l'évolution du système dans le temps. Une autre approche est appelée Simulation de Monte Carlo. Son but principal est de fournir l'information sur la probabilité d'occurrence de certains événements apparus sur le système. Pour les systèmes stochastiques la simulation dynamique (ou la simulation de fonctionnement) se rapproche fortement de la simulation de Monte Carlo [Pag 80]. Pour les systèmes déterministes la simulation dynamique est suffisante.

La confusion entre ces deux termes (simulation dynamique et simulation de Monte Carlo) provient alors du fait que dans le cas d'une simulation de fonctionnement, on peut prendre en compte un certain nombre de scénarios qui ont chacun une certaine probabilité d'occurrence [Pag 80]. Selon [Raj 00] l'avantage de la simulation est que les défaillances non markoviennes peuvent être modélisées facilement.

Le principe de la simulation de Monte Carlo est l'obtention d'un certain nombre de chronologies (simulations individuelles) suivies par leur traitement statistique. Cette méthode semble séduisante par la grande variété des hypothèses qu'elle permet de prendre en compte. Cependant, elle présente un inconvénient sérieux. Il s'agit du temps de calcul. Le temps nécessaire pour une seule chronologie peut être excessivement long (en cas d'un événement très rare) et de plus un nombre non négligeable de chronologies est demandé pour minimiser les erreurs statistiques. Le tableau 2.1 montre le nombre de simulations nécessaires pour obtenir l'écart type ε désiré dans le cas d'une variable aléatoire exponentielle :

$$n = \left(\frac{u_{\alpha/2}}{\varepsilon} \right)^2$$

où α (confiance symétrique)= 0.1 et $u_{\alpha/2}$ = 1.645

ε	0.3	0.2	0.1	0.05
n	30	68	270	~1000

Tab. 2.1 : Nombre de simulations nécessaires pour obtenir un écart type désiré [Pag 80]

Par exemple, pour l'analyse d'un modèle de Markov à trois états [Raj 00] a utilisé 100 simulations et retrouvé au final une disponibilité autour de 0.9986, donc très proche de la solution analytique.

Pour pallier la difficulté du temps de simulation [Pag 80] propose la méthode de réduction de variance et [Mon 98] la simulation par sauts. Les deux méthodes sont présentées dans la suite.

2.6.1 Les techniques de réduction de variance

Les techniques de réduction de variance ont un double but : d'une part diminuer la durée d'une chronologie, d'autre part augmenter la précision pour le nombre de chronologies donné. L'échantillonnage conditionnel et l'échantillonnage d'importance (appelé aussi stratification) sont deux méthodes applicables pour raccourcir la durée d'une simulation.

L'échantillonnage conditionnel se fait en mettant en évidence une variable aléatoire auxiliaire Z telle que φ s'écrive :

$$\varphi(X1, X2, ..., Xn) = f(X1, X2, ..., Xk, Z)$$

avec

$$Z=\Psi(Xk+1, Xk+2, ..., Xn).$$

L'estimation de φ se fait alors en deux étapes :

1.) Détermination de la loi de la variable aléatoire Z par simulation ou par une méthode analytique et
2.) Simulation (ou méthode analytique) pour la fonction $f(X_1, X_2, \ldots, X_k, Z)$. Cet échantillonnage peut être fait en cascade un certain nombre de fois.

[Pag 80] cite encore 3 autres méthodes qui peuvent être utilisées pour améliorer la mise en œuvre de la simulation de Monte Carlo dans la pratique :

- le procédé de la roulette russe ou de séparation, qui consiste à réduire la durée de la simulation en stoppant toute expérience qui aboutirait à un échec,
- l'utilisation des variables antithétiques, des méthodes de régression qui diminuent la variance du résultat grâce à l'estimation par des procédés qui se recoupent,
- la méthode de comparaison conduisant à une simplification, en neutralisant partiellement l'influence des variables aléatoires.

[Mel 02] présente une autre méthode de l'amélioration de Monte Carlo et plus particulièrement la sensibilité de la sortie en fonction des entrées en utilisant une estimation des gradients des critères étudiés pendant la simulation.

La simulation de Monte Carlo est largement utilisée pour l'évaluation des critères de la SdF des systèmes très divers. Citons par exemple [Mar 98], [Mon 98], [Bor 00], [Roc 02], [Lab 02], [Tuf 02] ou [Dev 02].

2.6.2 Le temps et la simulation

Le système d'automatisation à réseau est un mélange de deux bases de temps différents. D'un côté le réseau de commutation est un élément extrêmement rapide en juxtaposition aux constantes de temps des autres composants du système de l'autre côté. De plus, les constantes de temps des composants paraissent négligeables par comparaison avec le temps moyen de bon fonctionnement (temps entre deux pannes successives) des composants. Ceci est une caractéristique générale de processus défaillants qui peuvent selon [Châ 98] appartenir dans deux grandes classes : a.) les processus déterministes gouvernés par des petites constantes de temps et de l'autre côté b.) les processus stochastiques, notamment les défaillances, qui produisent des changements rapides de la dynamique mais qui sont caractérisés par des délais assez importants.

La particularité de l'étude de la sûreté de fonctionnement sur un système d'automatisation à réseau est une cohabitation entre des phénomènes très rapides, tels que les événements produits sur le bus de communication et des événements rares ou exceptionnels comme des défaillances. De plus toutes les caractéristiques des composants sont susceptibles d'être dynamiques et donc de dépendre non seulement de l'état actuel du système mais également de son historique. Dans la suite nous voulons montrer que le temps d'activité d'un SAR peut être considéré comme épars.

A titre d'exemple, illustrons le cas d'un SAR constitué de 3 composants (capteur, régulateur, actionneur). Tous les composants sont discrets, échantillonnés avec la même période d'une seconde. Dans chaque période deux informations sont alors échangées (mesure vers le régulateur et commande vers l'actionneur). Le réseau doit donc transmettre deux paquets contenant les deux informations. Soit 1 milliseconde la durée de transmission d'un paquet, 2 ms sont nécessaires pour transmettre toutes les informations dans le SAR. 998 ms restent inutilisées et disponibles pour d'autres communications éventuelles dans une période d'échantillonnage. On suppose que le réseau n'est pas partagé par d'autres applications.

Les caractéristiques des composants ont tendance à être dynamiques. Ainsi le taux d'utilisation de l'actionneur est sujet à l'ensemble de l'historique d'évolution du système. Par exemple le taux d'usure des composants dépend de l'algorithme de commande qui peut s'adapter à la configuration actuelle du système. Par conséquent il est très difficile d'estimer le temps de bon fonctionnement des composants et donc ils ne peuvent pas être facilement remplacés par un modèle mathématique. Un tel remplacement, utilisés par exemple dans [Mon 98], pourrait raccourcir d'une manière significative le temps nécessaire pour la simulation, mais son application réduirait l'aspect dynamique et stochastique du modèle d'une manière trop grossière et nous ramènerait vers l'étude de la sûreté de fonctionnement statique.

Si on préfère ne pas sacrifier la finesse du modèle nous proposons de travailler sur des échelles de temps modifiées. Au lieu de considérer la durée de transmission d'un paquet à une ms, elle sera fixée à 100 ms. Cette modification change un certain nombre de paramètres du modèle, notamment le taux d'occupation du réseau. Alors si l'étude portait sur ce paramètre, un tel échange ne pourrait pas être utilisé. L'autre point important modifié avec le changement de la vitesse de transmission est le comportement des composants réagissant sur les événements tels que l'arrivée d'une trame de communication. Dans ce cas, on doit être très attentifs aux conséquences de ce changement. Néanmoins, notre hypothèse de départ est la présence exclusive de composants échantillonnés donc non dépendant de l'instant d'arrivée des messages. Une synthèse des problèmes d’échantillonnage se trouve dans [Cas 03].

D'une manière tout à fait semblable la période d'échantillonnage peut être modifiée et augmentée en changeant les modèles des composants. L'augmentation de la période d'échantillonnage permet à son tour l'agrandissement de la durée de transmission du réseau. Pour ce faire, la condition suivante doit être respectée : toute information nécessaire pour la régulation doit être arrivée au destinataire dans un délai inférieur à la période d'échantillonnage des composants.

Concrètement au lieu d'utiliser la période d'une seconde, portons-la à cinq secondes et la durée de transmission d'un paquet peut alors devenir une seconde. Une seconde devient alors l'unité de base du temps dans le modèle. Elle peut être suffisante pour le modèle d'un réseau aussi bien que des composants utilisés. Pour étudier la sûreté de fonctionnement d'un système où les caractéristiques intéressantes apparaissent autour de mille heures, 5 millions de secondes restent à simuler. Le modèle d'un système d'automatisation comme décrit auparavant nécessite pour une telle simulation à peu près 500 minutes (ou approximativement 4 heures) de calcul sur un PC standard. Pour lancer 1000

simulations, la simulation demande donc 6 mois de temps de calcul sous condition que le temps nécessaire pour chaque simulation est constant et ne dépende pas du nombre de simulations.

Exemple : la durée d'un bit sur le réseau AS-i est de 6 µs. La constante de temps d'un composant est typiquement dans la plage de quelques secondes. Et le temps moyen de bon fonctionnement d'un composant peut atteindre des milliers d'heures. Le système et aussi son modèle intègrent donc des effets cadencés entre 10^{-6} sec et 10^{6} sec. Par conséquent, la modélisation du système n'est pas aisée.

Parmi les éléments présents dans le modèle, il y a le capteur analogique. Son utilisation est constante et ne varie pas. Il envoie une nouvelle mesure continuellement. Son taux de défaillance est constant. Il peut alors être remplacé par un modèle mathématique qui nous donne l'information sur l'instant de l'apparition de la défaillance. La question se pose de simuler ou non tous les moments jusqu'à la panne. Cela semble possible à la condition que le reste du système soit stationnaire[1] y compris l'entrée (valeur désirée), les perturbations et la sortie atteintes par le système. Dans ce cas on pourrait sauter jusqu'à l'instant estimé de la défaillance. A partir de cet instant, toutes les caractéristiques des composants doivent être changées.

Une piste est peut-être d'estimer les caractéristiques des composants sous toutes les configurations et après de les recombiner pour obtenir des valeurs globales. Est-ce qu'il s'agit dans ce cas d'une simulation par sauts ? Il ne semble pas, parce que dans ce cas la simulation n'est nécessaire que pour les intervalles ou le comportement du système n'est pas stationnaire.

2.7 SIMULATION PAR SAUTS

Le comportement d'un système d'automatisation à réseau peut être caractérisé comme un travail minutieux autour d'un point de fonctionnement qui s'étale dans l'intervalle entre des événements qui induisent le changement important dans le comportement du système : la défaillance d'un composant, le changement de consigne ou une rafale de pertes des trames de communication. Le travail minutieux consiste à relever des mesures, les envoyer, déterminer la nouvelle commande, souvent très proche de la valeur précédente, d'envoyer la commande et de l'appliquer par l'actionneur. La charge du système ne

[1] Dans ce contexte le mot stationnaire est utilisé dans son acception habituelle dans le domaine des signaux aléatoires : les propriétés statistiques sont invariantes dans le temps.

varie pas beaucoup à ce stade. Pour cette raison on peut faire appel à la méthode de simulation par sauts, utilisée entre autre dans [Mon 98].

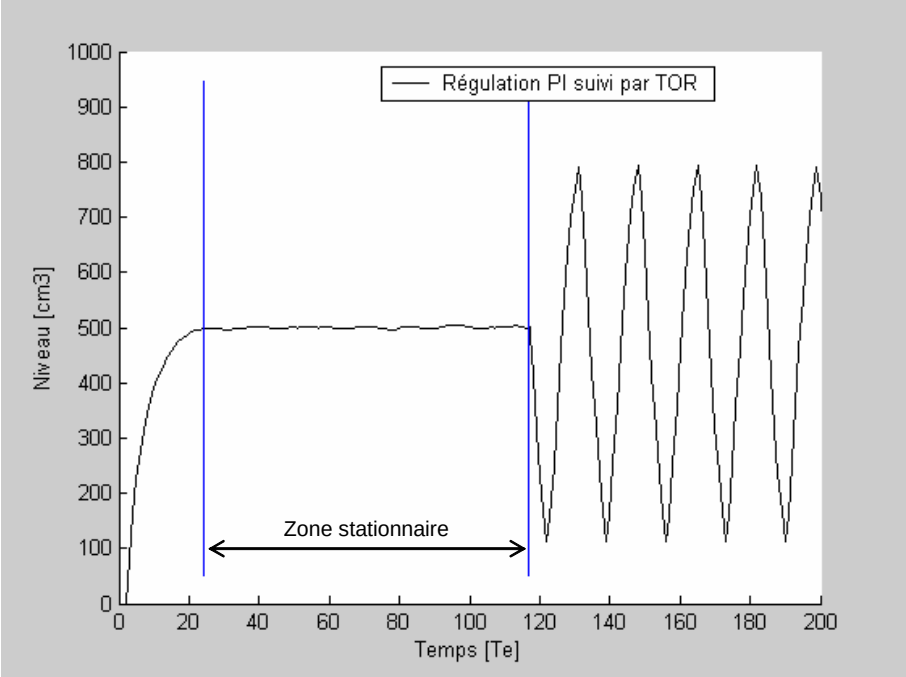


Figure 2.1 : Simulation de deux algorithmes

Un exemple qui montre ce phénomène est relevé sur la figure 2.1. Ce graphe contient le résultat d'une simulation de niveau de liquide obtenue grâce à un algorithme PI dans un premier temps, suivi par la régulation Tout-ou-Rien. La modification de la loi de commande est une conséquence d'une défaillance survenue au niveau du capteur analogique et l'utilisation de deux capteurs binaires pour un mode de fonctionnement dégradé. Après la montée initiale, pendant laquelle la cuve se remplit, le niveau est maintenu constant, jusqu'à l'apparition d'une défaillance du capteur. Un algorithme de contrôle dégradé se met alors en place et le niveau est maintenu en oscillations autour de la valeur désirée. La zone située entre les deux traits verticaux peut être considérée comme étant stationnaire. En effet, aucune variation importante n'est apparente. La valeur désirée reste constante ainsi que le niveau atteint et les perturbations au système sont stationnaires. Cette zone est délimitée à gauche par la fin de montée (phase transitoire) du niveau et à droite par la défaillance du capteur qui imposera l'entrée du système dans un mode de fonctionnement dégradé.

Vu la stationnarité supposée du système, il est possible, dans cette zone stationnaire, de remplacer les valeurs obtenues par la simulation par leurs estimations. La taille du saut dépend de l'instant de

l'occurrence du prochain événement qui entraînera le changement de comportement et des caractéristiques du système. La possibilité de prévision du prochain événement majeur est la première contrainte pour que la simulation par sauts puisse être utilisée.

Pour la mise en place de la simulation, il faut être capable d'identifier pendant la simulation l'arrivée du système dans une zone stationnaire. Cet estimateur est relativement facile à déterminer si le nombre d'événements importants est limité. Sa complexité ne dépend donc pas de la taille du système, mais seulement des critères de définition de la zone stationnaire et de l'ensemble des événements importants à prendre en compte.

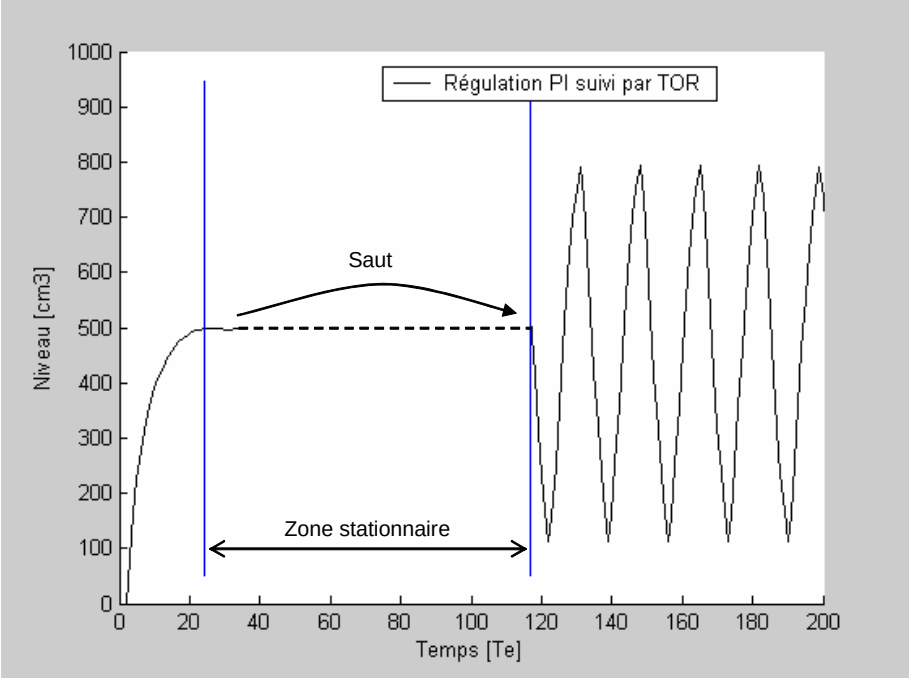


Figure 2.2 : Principe de simulation par sauts

Le saut pendant la simulation est montré sur la figure 2.2. Cette figure représente la zone stationnaire qui est composée d'une partie d'identification, où l'entrée en zone stationnaire est détectée et les paramètres stationnaires tels que l'ouverture de la vanne, sont calculés, suivi par le saut lui-même. Dans le cas présent par exemple le niveau du liquide est considéré constant et les états internes du modèle ne changent pas. Bien sûr, des estimateurs plus complexes peuvent être imaginés, entre autre pour détecter la stationnarité pendant les oscillations autour de la valeur désirée.

Les études de la sûreté de fonctionnement sur des systèmes d'automatisation à réseau constituent un champ d'application idéale pour la simulation par sauts, si la méthode d'analyse choisie est la

simulation de Monte Carlo. Le cycle de vie d'un tel système est composé en majorité de phases stationnaires. De plus, sous l'hypothèse de connaissances des lois de distribution de défaillances, les instants de la défaillance peuvent être calculés pour chaque phase stationnaire, qui peut être considérée comme un mode particulier de fonctionnement. Le problème majeur pendant la mise en place de cette approche est le développement de l'estimateur des zones stationnaires, notamment leur identification si la forme du comportement stationnaire n'est pas connue à l'avance.

2.8 ENJEU DE TAILLE DES MODELES

Un autre facteur est lié au problème de la vitesse de simulation : c'est le niveau de détail inclus dans le modèle. Nous nous sommes posés cette question avant de commencer à modéliser un réseau de communication. A ce stade la question fondamentale se pose : Quel réseau ou protocole faut-il modéliser ? Les réponses peuvent être classées en deux groupes principaux :

- Modélisation d'un protocole de communication réel,
- Modélisation d'un protocole imaginaire.

La modélisation d'un protocole réel mène à des modèles très complexes même pour des réseaux les plus simples, tel que AS-interface, alors que la modélisation d'un protocole définie par le concepteur pose des questions sur la pertinence de la définition du protocole.

L'autre grande difficulté consiste à déterminer le niveau de détail exprimé dans le modèle, autrement dit de sa finesse. C'est une question peut-être encore plus complexe à résoudre que la précédente. En effet, le niveau le moins fin est de travailler sur les trames d'une manière symbolique où par exemple l'adresse de destination est un symbole et l'adresse d'émetteur un autre symbole. Dans la programmation, ce niveau représenterait les langages évolués, telle que C ou encore Java. Le niveau intermédiaire représente alors la modélisation de chaque trame comme une série de bits, de chaînes booléennes qui parfois peuvent atteindre des dimensions très importantes. On peut comparer cette représentation avec les langages de bas niveau comme l'Assembleur ou le langage machine. Le niveau le plus fin est par conséquent la modélisation des valeurs des signaux électriques (tension et/ou courant) sur le câble assurant la liaison.
L'avantage d'une représentation des trames à l'aide des symboles est une amélioration de la lisibilité du modèle ainsi que sa conception accélérée. Néanmoins on perd un certain nombre de détails

potentiellement important. Ainsi le choix du niveau de détail dans le modèle est à choisir selon l'objectif de l'étude. Pour vérifier les concepts de base d'acheminement des trames le modèle de niveau symbolique s'avère suffisant, pour des études sur les courants parasites la modélisation la plus fine est nécessaire.

Une question supplémentaire reste toujours posée : Quel niveau de détail la méthode de modélisation utilisée peut-elle encore représenter ? Il s'avère que les Réseaux de Petri colorés peuvent aussi bien exprimer le niveau symbolique que le niveau des courants présents sur le bus de communication. De plus, la conception hiérarchique permet de commencer le développement des modèles sur le niveau symbolique et ensuite en cas de besoin de progressivement affiner les modèles.

Dans la plupart des modèles présentés dans cette thèse, le réseau est modélisé sur le niveau symbolique, ce qui est largement suffisant pour montrer le potentiel de la méthode utilisée pour l'analyse d'un système d'automatisation à réseau. Il reste donc à montrer que même les signaux électriques peuvent être représentés en cas de besoin. Pour cela nous allons nous servir de la spécification des signaux électriques présentés dans la norme EN 50295 (spécification du réseau AS-i, voir [NF 50295]).

Le principe de codage de la transmission doit se présenter comme indiqué sur la figure 2.3. L'état inactif doit être représenté par un "1". La requête maître ou la réponse esclave doit être codée au format Manchester II avec modulation d'impulsions alternatives (APM). L'intervalle bit doit être de 6 µs. Un "0" doit être codé par un intervalle de 3 µs au niveau fort suivi par un intervalle de 3 µs au niveau faible. Un "1" doit être codé par un intervalle de 3 µs au niveau faible suivi par un intervalle de 3 µs au niveau fort. Le transmetteur doit être installé comme un puits de courant. L'amplitude du courant de modulation doit être comprise entre 55 mA et 68 mA. Avec les inductances du circuit de découplage de l'alimentation, ce courant générera une impulsion négative à chaque front montant et une impulsion positive à chaque front descendant. La forme d'onde idéale des impulsions est (EN 50295): $u(t) \approx \pm U_{send} \times \sin^2\left(\frac{2\pi}{6\mu s} t\right)$ où U_{send} est une constante, approximativement égale à 2 V.

Il devient ainsi clair, que pour modéliser le réseau de la manière la plus détaillée, il faut modéliser des signaux sur le câble. Pour faire cela, il suffit de discrétiser l'intervalle bit et échantillonner la valeur de la tension présente sur le bus. On obtient alors une séquence numérique qui peut être exprimée à l'aide des RdP colorés. Néanmoins, le choix de niveau de détail dans le modèle reste toujours un compromis entre la précision et le temps de l’analyse.

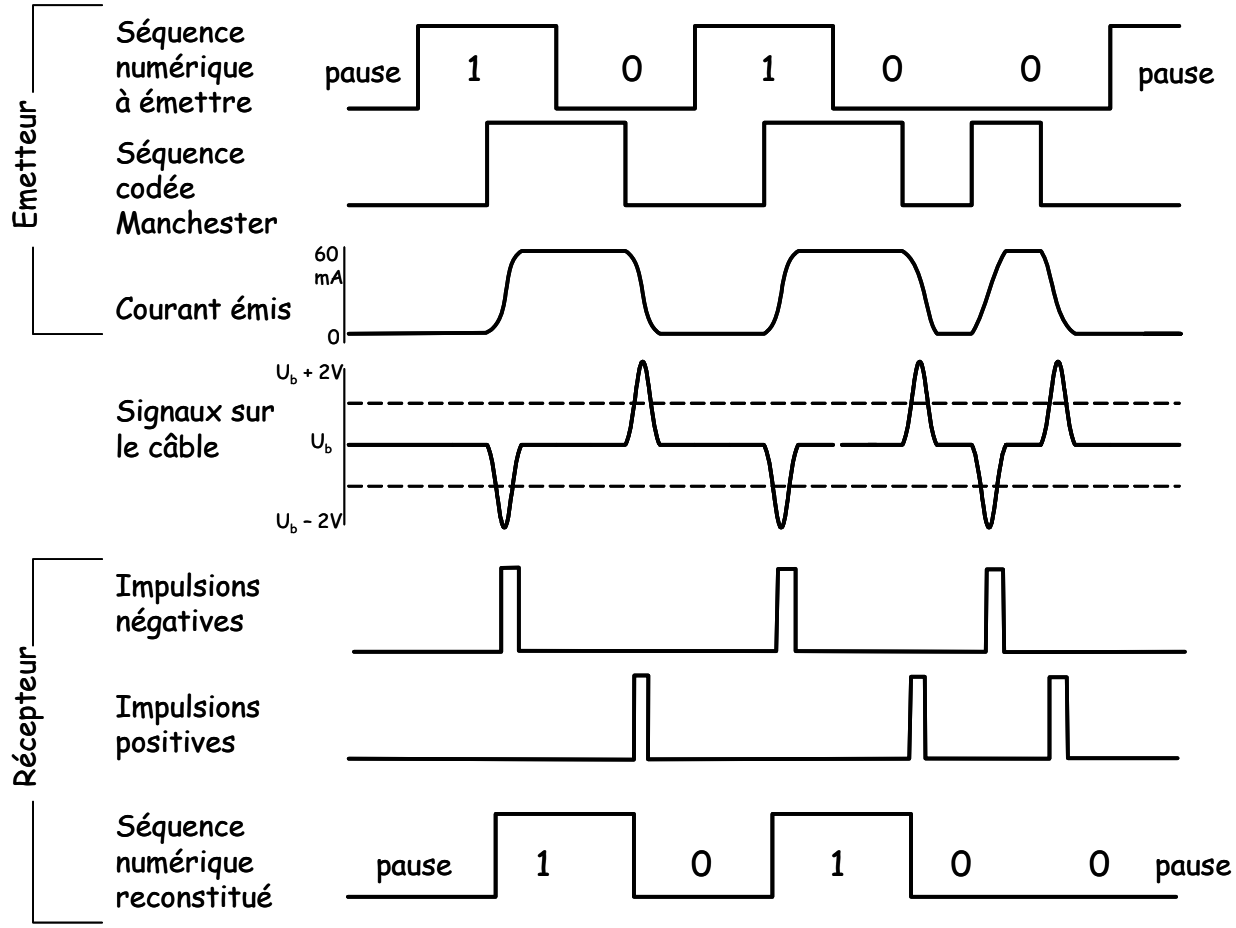


Figure 2.3 : Principe de codage de transmission

3 Réseaux de Petri colorés

3.1 INTRODUCTION

Les Réseaux de Petri (RdP) ont été choisis dans cette étude pour plusieurs raisons. Premièrement il permettent de représenter statiquement les systèmes grâce à leur structure (ensemble : places-transitions-arcs). Deuxièmement, ils peuvent modéliser le comportement dynamique des systèmes avec l'évolution des jetons dans les places.

Une autre raison de ce choix est leur reconnaissance. Ils sont reconnus comme un outil de modélisation puissant aussi bien par la communauté des automaticiens que celle des informaticiens. Il s'agit d'un outil largement accepté pour l'étude des performances des systèmes, mais aussi leur sûreté de fonctionnement. D'une manière traditionnelle, les modèles et les méthodes d'analyse du bon fonctionnement et des performances sont étudiés par des approches distinctes. Mais à présent, il devient de plus en plus difficile de fixer une frontière entre ces domaines. On peut alors observer une tendance à l'unification des approches combinant les avantages des méthodes formelles (focalisées sur le fonctionnement) et celles d'analyses de performances [Mor 02].

Le choix des RdP colorés est conditionné entre autre par un besoin d'exprimer des valeurs analogiques ou discrètes, comme le niveau, la température ou la vitesse. De plus il est nécessaire de transmettre ces informations au sein d'une trame de communication. Il faut alors que le jeton représentant la trame puisse exprimer des valeurs différentes (telles que le destinataire du message, les données utiles, etc.). Ces demandes ont alors imposé l'utilisation des RdP colorés.

Dans ce chapitre nous rappelons d'abord la définition des Réseaux de Petri Colorés, ensuite nous résumons brièvement la recherche de logiciel de travail. La plus grande partie est consacrée à l'introduction à Design/CPN et à la présentation très détaillée de quelques concepts de base. Ce chapitre est complété par un rappel des RdP classiques se trouvant dans la deuxième partie de l'annexe.

3.2 DEFINITIONS

Un **Réseau de Petri Coloré** non hiérarchique est un produit cartésien CPN=(Σ, P, T, A, N, C, G, E, I) qui satisfait les conditions ci-dessous [Jen 97] :

(i) Σ et un ensemble fini de types non vides, appelés **ensembles des couleurs**,

(ii) P est un ensemble fini de **places**,

(iii) T est un ensemble fini de **transitions**,

(iv) A est un ensemble fini d'**arcs,** tel que :

- $P \cap T = P \cap A = T \cap A = \varnothing$,

(v) N est une fonction des **nœuds**. Elle est définie de A vers PxT ∪ TxP,

(vi) C est une fonction des **couleurs**. Elle est définie de P vers Σ,

(vii) G est une fonction des **gardes**. Elle est définie de T vers les expressions telles que :

- $\forall t \in T : [Type(G(t)) = B \wedge Type(Var(G(t))) \subseteq \Sigma]$,

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

- $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$, où p(a) est la place de N(a).

(ix) I est une fonction **d'initialisation**. Elle est définie de P vers les expressions fermées telles que :

- $\forall p \in P : [Type(I(p)) = C(p)_{MS}]$.

Un pas Y dans un marquage M est **permis** si et seulement si la propriété suivante est respectée :

$$\forall p \in P : \sum_{(t,b) \in Y} E(p,t) < b > \leq M(p) .$$

3.3 CHOIX DU LOGICIEL

Une fois la décision de travailler avec les Réseaux de Petri colorés prise, il reste à choisir l'outil de modélisation et d'analyse avec le potentiel de satisfaire la plupart de nos besoins. Il est à souligner que ce choix est un compromis et l'outil idéal n'existe pas. Le choix d'un logiciel existant a été préféré au développement d'un logiciel pour une simple question de coût de développement en temps.

Le choix se fait sur deux ensembles de critères. Le premier ensemble contient des critères imposés. Si le logiciel ne répond pas à un, il est automatiquement éliminé. Le deuxième ensemble est représenté par des critères qui peuvent rendre le travail plus efficace, mais dans ce cas le choix peut être très subjectif.

Parmi les critères imposés sont comptés :

- la possibilité d'exprimer un arc inhibiteur,
- le travail avec des données structurées,
- la présence des expressions stochastiques.

Critères apportant des plus-values :

- disponibilité du logiciel,
- coût de son acquisition,
- conception modulaire,
- son acceptation par les différentes communautés concernées par ce travail.

A la fin, 3 logiciels ont été retenus :

1. Design/CPN
2. GreatSPN
3. Miss-RdP

En pratique, il existe deux façons à l'intérieur du domaine Réseaux de Petri de haut niveau pour construire les modèles et les analyser ensuite. Il s'agit d'utiliser soit des Réseaux de Petri colorés [Jen 97], dont le logiciel Design/CPN [design] est l'implantation de référence. L'autre approche consiste en utilisation des Réseaux de Petri stochastiques généralisés [Mar 95] tels que présents dans le logiciel GreatSPN [greatspn].

GreatSPN permet de modéliser et d'évaluer les RdP stochastiques généralisés et aussi leur extension colorée : les réseaux stochastiques bien formés. Il intègre des modules comme l'interface graphique, un analyseur des propriétés structurelles (pour le moment non disponible pour des réseaux colorés), un module de programmation linéaire (pour le moment non disponible pour des réseaux colorés), un générateur de graphe de marquage, un solveur Markovien et des modules de simulation (pour le moment non disponibles pour des réseaux colorés), un analyseur des réseaux bien formés (simulation et graphe de marquage).

L'inconvénient majeur de l'utilisation de ce logiciel dans notre travail est l'absence de couleurs prédéfinies comme le type réel ou entier. En fait, GreatSPN repose sur la définition de chaque couleur par énumération ce qui est contraire aux caractéristiques requises. De l'autre côté il faut noter la présence d'un solveur Markovien efficace, ce qui n'est pas le cas de Design/CPN.

Quant à Miss-RdP, la conception hiérarchique dans ce logiciel est actuellement en voie d'implémentation, donc indisponible.

Design/CPN à l'origine n'est pas conçu pour modéliser les systèmes stochastiques. Mais avec son grand pouvoir d'expression, il permet d'exprimer plus ou moins confortablement les modèles équivalents aux RdP stochastiques. C'est pour cette raison que son utilisation est privilégiée dans le travail présent. Il a fait ses preuves entre autre dans [Meu 00] et [Son 03].

Même si les trois logiciels mentionnés possèdent des caractéristiques conformes aux critères imposés Design/CPN a été choisi pour l'ensemble de cette étude à cause de sa disponibilité et de son acceptation par une large communauté scientifique. Une description plus détaillée du processus de sélection du logiciel peut être trouvée dans [Bar 00].

En absence d'un solveur Markovien, l'association des RdP avec la simulation de Monte Carlo est une façon efficace de traiter les problèmes liés à la SdF des systèmes complexes actuellement présents dans industrie [Sig 02], [Mor 02].

3.4 INTRODUCTION A LA MODELISATION DESIGN/CPN

Le but de cette partie est de présenter les mécanismes de base qui par la suite permettent de développer des modèles plus complexes.

Un **RdP coloré** (CPN dans la suite, ‘colored Petri net’, [Jen 97]) est composé des **places**, des **transitions** et des **arcs**. Tous ces composants créent la structure statique d'un CPN. D'autre part, les **jetons** forment la partie dynamique d'un CPN. A cela s’ajoute au moins un **champ de déclarations globales** (Global Declaration Node). Différents attributs (tels que nom, couleur, etc.) sont associés à ces objets. Certains d’entre eux sont obligatoires, d’autres facultatifs. Tous ces objets sont disposés sur

des feuilles ou pages. Une feuille est simplement un espace graphique vide, qui peut contenir tout objet utilisé dans Design/CPN.

3.4.1 Une place CPN

Comme dans tous les RdP, une place est graphiquement présentée comme un cercle ou plutôt une ellipse. 3 attributs peuvent lui être associés : 1.) le **nom** de la place, 2.) la **couleur** et 3.) le **marquage initial**. Le nom et le marquage initial sont des attributs facultatifs. Sous certaines conditions, telle que la construction d'un graphe d'occurrence (voir plus tard), le nom devient un attribut obligatoire. Dans ce cas il est exigé que le nom de la place soit unique sur sa feuille.

La couleur de chaque place doit lui être attribuée. Une place ne peut contenir que des jetons qui appartiennent à cette couleur. L'insertion de tout autre jeton provoque une erreur de compilation. Chaque couleur associée aux places doit être déclarée dans le champ des déclarations globales.

Le marquage initial reste, sous toutes les conditions, un attribut facultatif. Il peut contenir un ou plusieurs jetons dont les valeurs appartiennent à la couleur de la place.

Par convention dans nos modèles le nom de la place est toujours (dans la mesure du possible) écrit en **gras** directement à l'intérieur de la place. La couleur est indiquée dans un des quatre coins virtuels de l'ellipse (en haut à gauche, en bas à droite, etc.) et toujours en *italique*. Le marquage initial est placé de préférence à l'intérieur de la place en-dessous du nom. Si le marquage initial est trop grand pour être mis à l'intérieur il est placé à proximité de la place. Il est toujours écrit sans aucune accentuation (police normale).

3.4.2 Une transition CPN

Il est devenu une habitude de représenter une transition dans les RdP comme un rectangle. Il en est ainsi avec les CPN. Une transition peut posséder 5 attributs, tous facultatifs (sauf pour l'utilisation spécifique où le nom devient obligatoire, voir les attributs d'une place) : 1.) le **nom** de la transition, 2.) l'expression de **garde**, 3.) la définition du **code** (programme), 4.) le **log** ou champs des traces et 5.) la **temporisation**.

L'expression de garde d'une transition limite sa validation aux cas où cette expression est vraie. Son utilisation est expliquée sur des exemples dans la suite.

Le segment de code permet d'associer l'exécution d'un sous-programme en langage ML à l'instant de l'exécution de la transition. Il est composé d'une définition des entrées, des sorties et de la fonction à exécuter, précédée par la formule *action*. La présence du segment de code est indiquée par la lettre C placée dans un petit carré dans l'onglet bas droit de la transition.

Le champ des traces *log* permet de garder les traces des franchissements de la transition. Il n'est pas utilisé dans nos modèles.

Une transition CPN peut être temporisée. Ceci correspond exactement aux RdP T-temporisés dans [Dav 97].

Par convention les règles suivantes sont observées dans nos modèles : 1.) le nom est inscrit à l'intérieur de la transition en **gras**, 2.) l'expression de garde est placé entre crochets [] à proximité de la transition, 3.) le segment de code, à part son indication par la lettre C, est écrit (dans la mesure du possible, ce qui est rarement le cas pour les grands modèles) dans un rectangle pointillé dans le voisinage de la transition et 4.) la temporisation est écrite à l'intérieur de la transition en dessous de son nom, elle est facilement reconnaissable par l'arobasse (@) par lequel elle commence.

3.4.3 Un arc CPN

Un arc relie toujours une place et une transition. Son seul attribut, **l'inscription**, permet de profiter pleinement des avantages des RdP. Une inscription sur l'arc peut contenir soit une ou plusieurs **valeurs** de la couleur appropriée, soit une **variable**, soit une **fonction**.

L'association d'une valeur à l'arc résulte en la restriction de la "bande passante" de l'arc. Ainsi pour que le jeton puisse passer l'arc il faut qu'il contienne cette valeur, sinon le passage lui est interdit. Si un tel arc se trouve en amont de la transition il dévalide alors la transition si la place ne contient pas de valeur précisée sur l'arc.

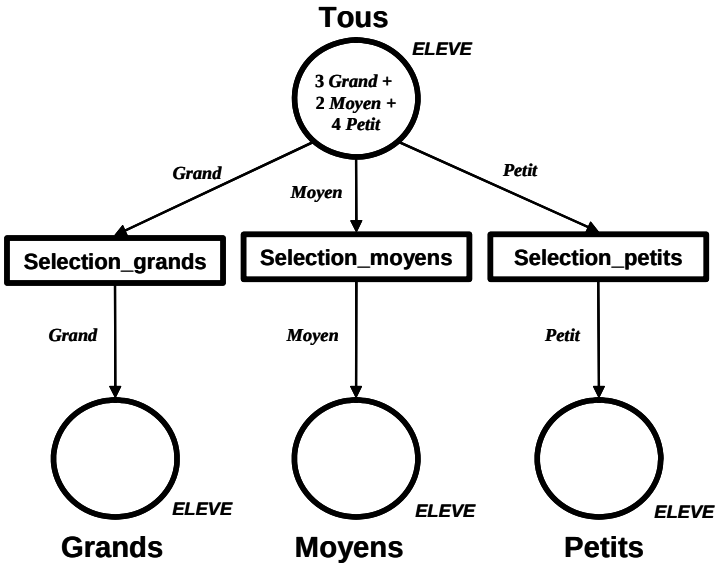


Figure 3.1 : Des valeurs associées aux arcs

Sur la figure proposée (figure 3.1) il y a quatre places : *Tous*, *Grands*, *Moyens*, *Petits*. Le but est de trier les élèves selon leur taille. Au début tous les élèves sont dans la place *Tous* (3 *Grands*, 2 *Moyens* et 4 *Petits*). Seules les *Grands* peuvent passer par l'arc vers la transition *Selection_grands*. Ainsi pour les autres. A la fin, chacun des élèves se trouve dans la place qui lui correspond.

L'inscription d'une variable sur l'arc permet de faire passer par l'arc des différentes valeurs, par exemple, si tous les élèves doivent passer un par un à travers le couloir jusqu'à la cantine (voir figure 3.2). Nous définissons pour cela une variable *élève* du type *ELEVE*. A lui on associe une valeur parmi celles disponibles dans la place *Tous* et ainsi la transition *Couloir* peut être franchie. Une fois que la

valeur est jointe à la variable, cette valeur ne change pas pendant tout le franchissement de la transition. Ainsi à la fin de la simulation tous les élèves se trouvent à la *Cantine*.

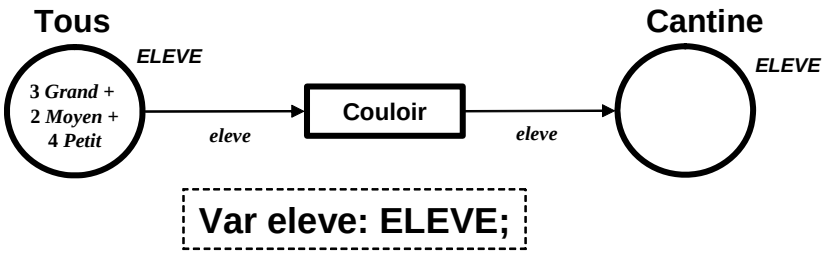


Figure 3.2 : Association des variables aux arcs

Pour illustrer l'utilisation des fonctions sur les arcs, nous reprenons l'exemple du triage des élèves de la figure 3.1. Sur la figure suivante (3.3) on retrouve alors les quatre places *Tous*, *Grands*, *Moyens*, *Petits*, mais il ne reste qu'une seule transition *Selection*. Pour son franchissement un élève sera associé à la variable *élève*. Après le franchissement il passe par les trois conditions : si *élève=Grand* then *Grand* else *empty*, etc. Ainsi la taille de chaque élève sera comparée trois fois avec forcément une réponse positive et deux négatives. Il faut souligner que dans des cas d'utilisation des conditions sur les arcs, il faut impérativement inclure tout composant possible de la couleur exactement une fois. Si par exemple il y avait un *Mauvais élève* dans la place *Tous* sur la figure 3.3 et il n'est pas pris en compte par aucune des conditions en aval de la *Selection*, cet élève passera la *Selection*, mais comme *Mauvais* n'est ni *Grand*, ni *Moyen*, ni *Petit*, alors il obtiendra 3 réponse négatives et ne sera ajouté dans aucune place. Un *élève* se serait alors volatilisé. Ceci peut être souhaité dans certains cas, mais en général un tel comportement est à éviter. D'une manière proche la multiplication des élèves est aussi possible. Bien sûr, beaucoup d'autres possibilités d'utilisation des fonctions inscrites sur des arcs existent.

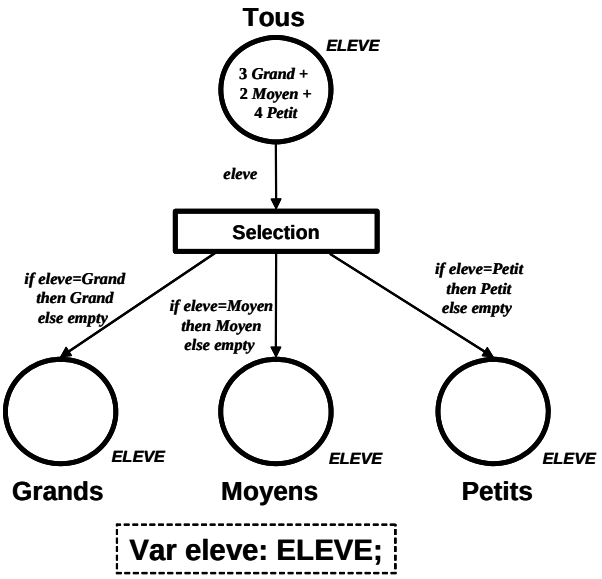


Figure 3.3 : Association des fonctions aux arcs

Ici il est nécessaire de mentionner l'utilisation des variables libres. Sur la figure 3.4 pour franchir la transition *Transformation* un élève est associé à la variable *élève1*. Mais en aval de cette transition ne se trouve pas la variable *élève1* mais *élève2*. Or jusqu'à présent aucune valeur ne lui est attribuée. Alors le logiciel va lui attribuer une valeur choisie 'au hasard' parmi les valeurs de la couleur *Elèves*. Puisque la probabilité de chaque valeur est identique (dépendant du choix d'algorithme de génération des nombres aléatoires) à la fin de la simulation, la place *Transformé* va tendre à contenir autant de *Grands*, que de *Moyens* et *Petits*.

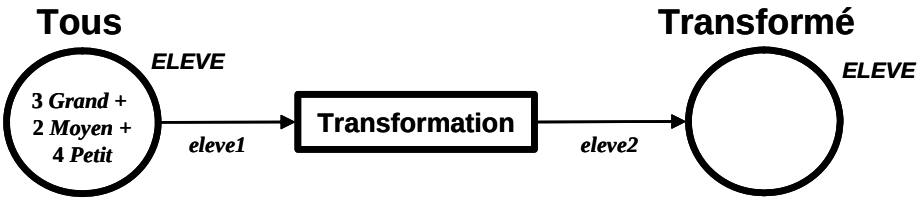


Figure 3.4 : Exemple d'une variable libre

3.4.4 Temporisation des arcs

Dans la théorie classique des RdP deux grands types de temporisations sont reconnus :

1.) Celles où le temps est associé à la place, c'est-à-dire RdP P-temporisés et
2.) Celles où le temps est un attribut de la transition et on les nomme RdP T-temporisés.

Le logiciel Design/CPN permet d'inclure le temps dans la définition de chaque transition. Il ne le permet pas pour les places. Néanmoins il donne à l'utilisateur également la possibilité de temporiser les arcs. Les inscriptions sur les arcs de sortie peuvent contenir des spécifications de durée de 'traversée' du jeton par l'arc. Il s'agit d'un moyen simple et efficace de réduire la taille des modèles puisque chaque transition peut avoir plusieurs arcs temporisés avec des délais différents ce qui remplace plusieurs transitions temporisées. La combinaison des arcs temporisés et des arcs non temporisés est également possible pour chaque transition.

3.5 QUELQUES BRIQUES DE DESIGN/CPN

3.5.1 Introduction au langage ML

Le langage ML est associé à Design/CPN. Il s'agit d'un langage de programmation fonctionnelle et pas de programmation procédurale telle que C ou Pascal.

Différentes variantes de ce langage existent. Design/CPN implémente une version proche du SML/NJ (Standard ML of New Jersey) très bien décrite dans [Ull 98]. Néanmoins un certain nombre de modifications par rapport à cette version est présent dans Design/CPN et elles sont expliquées sur le site [design].

C'est cette variante du langage ML que nous allons dans la suite appeler ML.

3.5.2 Tuples et trames de communication

Un tuple (produit cartésien) dans ML est formé de deux ou plusieurs éléments de types quelconques et il est délimité par des parenthèses. Ainsi un tuple ressemble à un 'record' avec la différence de s'adresser aux éléments par leur position et non par le nom du composant.

Plus formellement nous pouvons écrire que le tuple est un produit cartésien d'arité (dimension) n :

$$A_1 \, x \, A_2 \, x \, \ldots \, x \, A_n = \{(a_1, a_2, \ldots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n\}$$

Sous ML les ensembles élémentaires $A_1, A_2, \ldots, A_n$ peuvent être soit des types simples (comme booléen, nombres entiers, chaînes de caractères) ou des types composés (tuples, listes et autres (fonctions)).

Le tuple entier peut représenter un jeton dans Design/CPN. L'exemple typique de son utilisation est la trame de communication. Nous la définissons formellement en tant qu'un produit cartésien de type *String x String x Data x String* où *data* lui-même est un produit cartésien de type *Int x String*. Un exemple de la trame est : ("Source1", "Desti2", (34,"OK"), "CRC_OK"). La signification de cette trame est que la source "Source1" (originaire de la trame) envoie un message au destinataire "Desti2" un tuple des datas 34 et "OK". La fin de la trame est une chaîne de caractères représentant le statut de vérification de la trame.

Vis-à-vis des Réseaux de Petri colorés les tuples peuvent être considérés comme des couleurs complexes.

3.5.3 Listes et tampon d'émission

Une liste est une séquence finie de zéros ou plusieurs éléments d'un type donné [Aho 93]. Elles permettent d'exprimer entre autre l'ordonnancement. Dans l'exemple du réseau de communication nous les utilisons pour ordonnancer les paquets avant leur transmission.

Une liste est symbolisée dans ML par des crochets [ ]. Ainsi par exemple [2, 5, 9] est une liste de nombres entiers ou encore ["Ma", "première", "liste"] est une liste de chaînes de caractères. ML exige que tous les composants d'une liste appartiennent à la même classe. Ainsi la liste [2, "premier", 4] n'est pas permise car elle mélange des nombres entiers et des chaînes de caractères. Les tuples sont autorisés

comme éléments des listes et par conséquent la liste [(2, "premier", 4)] est valide car elle contient un élément – tuple composé de nombres entiers et de chaînes de caractères.

Une place particulière est réservée à la liste vide ([ ] ou *nil* dans ML). Une liste vide fait partie de chaque liste et elle est toujours son dernier élément. Sa longueur est zéro (la longueur d'une liste étant le nombre d'éléments qu'elle contient).

Sur les listes unidirectionnelles (dans la suite appelées simplement listes), il n'y a que deux opérateurs définis :

- la concaténation (exprimée dans ML par ^^),
- l'ajout de tête à une liste (écrit comme ::).

Une concaténation de deux listes a pour résultat une liste qui commence par les éléments de la première liste et finit par les éléments de la deuxième liste. Exemple : [4, 9, 3]^^[2, 93] donne [4, 9, 3, 2, 93].

L'ajout d'une tête (une tête étant le premier élément d'une liste) à une liste produit une liste de longueur incrémentée de 1. Exemple : 5::[4, 9, 3] produit [5, 4, 9, 3]. La différence de la concaténation, qui elle fusionne deux listes, est que l'ajout d'une tête prend en arguments un composant et une liste (et non pas deux listes).

Cet opérateur peut aussi être utilisé pour sélectionner la tête d'une liste. Exemple : Ayons une liste [4, 9, 3, 2, 93] que nous voulons rendre égale à A::L. Le résultat obtenue est qu'A est égal à 4 et L est une liste égale à [9, 3, 2, 93]. L'élément A est appelé la tête de liste et la liste L est nommée la queue de liste. C'est la seule façon d'accéder aux composants d'une liste. Pour sélectionner par exemple le troisième composant de la liste [4, 9, 3, 2, 93] il faut d'abord enlever la tête en faisant A::L (A égal à 4, L égal à [9, 3, 2, 93], puis enlever la tête du reste (L) B::L1 (B devient 9 et L1 égal à [3, 2, 93]) et le troisième composant est maintenant devenu la tête de la liste L1 et on peut l'extraire par C::L2 (C est 3 et L2 [2, 93]).

Il est évident que pour travailler sur les listes la récursivité s'impose. A titre d'exemple nous présentons une fonction récursive pour déterminer la longueur d'une liste :

```
len([ ])=0;
| len (A::L) = 1+ len(L);
```

Cette fonction définit dans sa première ligne que la longueur d'une liste vide est égale à zéro. La ligne suivante dit que la longueur d'une liste composée d'une tête et d'une queue est égale à la longueur de la queue incrémentée d'un.

Après avoir brièvement présenté les listes nous allons maintenant illustrer leur utilisation pour modéliser un tampon d'émission d'un réseau de communication.

Soit un tampon d'émission où tous les composants voulant émettre placent leurs paquets de communication dans une file d'attente de type FIFO (first-in first-out, premier venu premier servi). C'est à dire que le premier paquet (la tête) de la file d'attente sera transmis dès que le réseau de communication est libre (voire la figure 3.5).

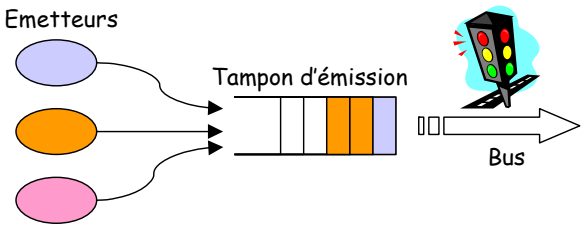


Figure 3.5 : Une liste comme un tampon d'émission

Il y a quatre composants qui accèdent exclusivement (un par un) au tampon d'émission : 3 émetteurs pour y écrire et le bus pour y lire et libérer les positions. La lecture et l'écriture ne peuvent pas se passer simultanément. Aucune priorité n'est définie. Si plusieurs composants veulent accéder au tampon un d'eux est choisi au hasard et commence son interaction avec le tampon. Pendant l'interaction le tampon est réservé et les autres composants ne peuvent pas y accéder.

Sur la figure suivante (figure 3.6) nous présentons le modèle CPN du tampon d'émission défini auparavant.

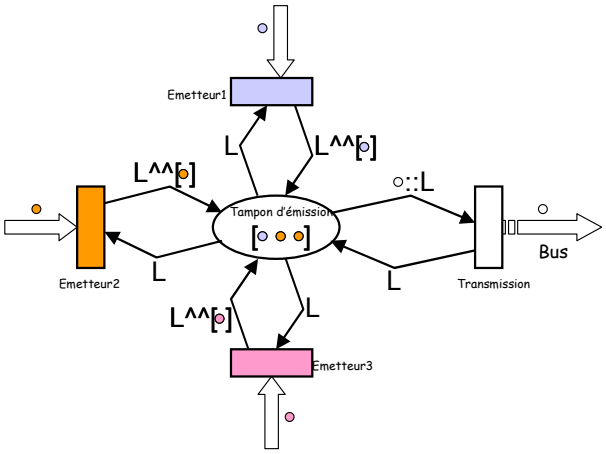


Figure 3.6 : Modèle CPN d'un tampon d'émission

A l'instant présenté sur la figure il y a déjà trois trames de communication présentes dans le tampon : une trame provenant de l'émetteur1 (à la tête de la liste) et deux de l'émetteur2. Sous la syntaxe ML et selon notre définition des trames de communication ce contenu peut-être exprimé comme : [("Emetteur1", "Desti1", DATA1, "CRC"), ("Emetteur2", "Desti2", DATA2, "CRC"), ("Emetteur2", "Desti3", DATA3, "CRC")].

Présentons d'abord le mécanisme de l'écriture dans le tampon. Un émetteur veut donc rajouter sa trame dans le tampon. Pour cela il faut premièrement que le tampon soit disponible. Si cette condition est respectée alors l'émetteur peut prendre la liste de la place "Tampon d'émission" (représenté par l'arc avec l'inscription "L"). Ainsi il dévalide toutes les autres transitions (composants), ce qui correspond à la réservation du tampon pour l'écriture. Pour écrire il ajoute sa trame au contenu du tampon (qui se traduit par la concaténation de la liste L et une liste de longueur 1 contenant la trame à transmettre). A l'arrivée du jeton à sa place "Tampon d'émission", toutes les transitions sont à nouveau validées et un nouveau cycle peut commencer.

D'une manière analogue s'effectuent la lecture et la libération des places par le bus. Si le bus est disponible (ce mécanisme n'est pas représenté ici) et le tampon d'émission contient au moins une trame à transmettre, alors la transmission peut commencer. On récupère le contenu du tampon en séparant la première trame (tête, de provenance quelconque, représenté par un jeton (petit cercle) vide) et le reste du contenu du tampon (représenté par L). Après l'émission la première trame est transmise sur le bus tandis que le reste du contenu retourne au "Tampon d'émission" finissant ainsi le cycle élémentaire de lecture du tampon.

Le mécanisme de l'écriture systématique à la fin du tampon et de la lecture de la tête du tampon garantit que les messages seront transmis dans l'ordre dans lequel ils sont arrivés au tampon donc on retrouve bien la file d'attente type FIFO.

3.5.4 Différences entre tuples et listes

Même si au premier regard les tuples et les listes pourraient paraître semblables, les deux concepts sont d'un fonctionnement bien différent et ne peuvent en aucun cas être substitué l'un à l'autre. D'une part, la longueur d'un tuple est fixe, celle d'une liste est variable. Ceci est utile pour représenter par exemple une pile dont la taille n'est pas connue en avance. D'autre part, par la position nous pouvons accéder à chaque composant d'un tuple, mais seulement à la tête d'une liste.

3.5.5 Arc inhibiteur

Parmi les extensions de RdP particulièrement utiles se trouve l'arc inhibiteur. La définition exacte de cette extension peut être trouvée dans [Dia 01]. Il s'agit d'un arc qui valide le franchissement de la transition si le nombre des jetons dans la place est strictement inférieur à son poids. Une de ces applications principales est le test de la fin d'une action. Dans notre cas, nous avons voulu en faire l'usage pour la remise automatique des compteurs à zéro à la fin d'un cycle complet de comptage.

Design/CPN ne revendique pas l'intégration de cette extension, il a alors fallu trouver une façon de la remplacer. La solution retenue met en œuvre les listes. La place vide est alors représentée par la place contenant une liste vide. Et une place non vide par la place avec la liste non vide, c'est-à-dire une liste qui contient au moins un élément. Pendant les franchissements des transitions au lieu de changer les contenus des places on change plutôt le contenu de leurs listes. On obtient ainsi une représentation équivalente aux arcs inhibiteurs classiques qui n'alourdit que légèrement la conception du modèle.

3.5.6 Arc probabiliste

L'arc probabiliste (par exemple entre la transition *Usure* et la place *Etat* sur la figure 3.7) est utilisé pour exprimer la nature stochastique des événements. Dans nos travaux il s'agit notamment de la modélisation des pannes survenues aux composants.

Pour indiquer l'état du composant il existe une place appelée *Etat*. A l'origine le composant se trouve dans un état de fonctionnement nominal symbolisé par un jeton initial "Ok". Ce jeton est utilisé pour franchir la transition *Usure*.

Après avoir franchi cette transition la valeur de la variable "*aléatoire*" est examinée. Cette valeur est générée par un générateur de nombres aléatoires avec une distribution uniforme dans l'intervalle entre 1 et 100. Si cette valeur est inférieure à 90 le composant reste dans son état de fonctionnement nominal et le jeton "Ok" est remis dans la place *Etat*. Dans le cas contraire (la valeur de la variable "*aléatoire*" est supérieure ou égale à 90) une panne du composant se manifeste par l'ajout du jeton "Err" dans la place *Etat*.

La panne du composant survient donc en moyenne 10 fois sur 100 franchissements de la transition *Usure*. Cette représentation d'occurrence des pannes ne peut être considérée comme complète sans l'intégration du temps dans le modèle. Celle-ci est décrite dans les paragraphes suivants.

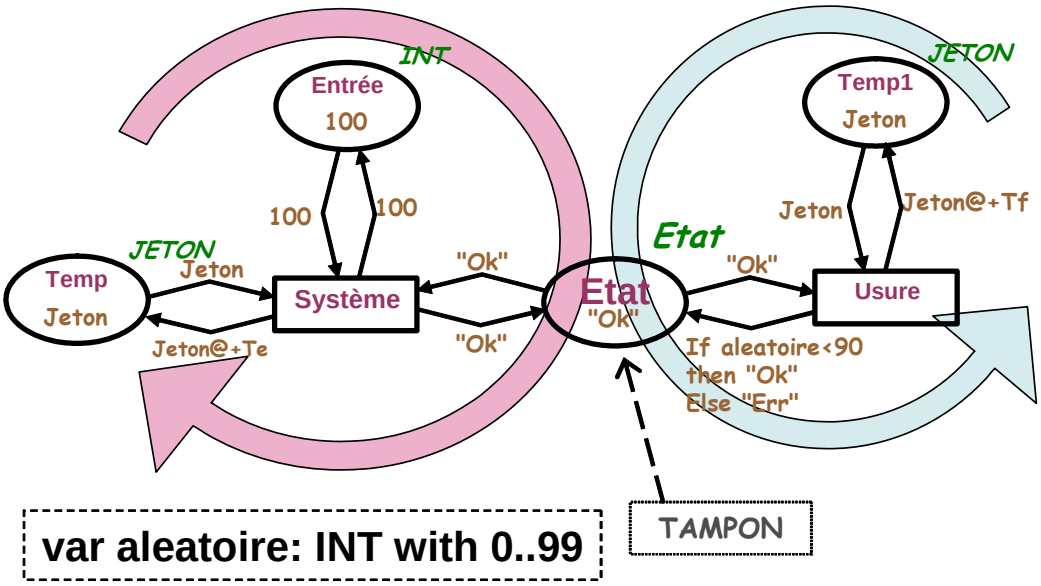


Figure 3.7 : Arc probabiliste et place tampon

3.5.7 Places tampons

Les tampons sont des places qui contiennent toujours des jetons et valident ainsi toujours les transitions en aval de ces places. Des exemples typiques de leur utilisation sont les tampons d'émission ou de réception de réseaux de communication ou encore les états utilisés pour exprimer l'état fonctionnel ou dysfonctionnel des composants.

L'enjeu particulier de ces places est qu'elles sont en amont de deux transitions qui doivent être synchrones. Une place exprimant l'état fonctionnel ou dysfonctionnel est utilisée pour illustrer un tel cas (place *Etat* sur la figure 3.7). Cette place a une double utilisation :

- participer à la modélisation d'une défaillance,
- conditionner le traitement des informations dans le reste du composant.

Il est évident que l'exécution d'une de ces deux fonctionnalités ne doit pas empêcher l'autre fonctionnalité de continuer à s'exercer[2]. Pour pouvoir assurer cette exigence nous proposons la

[2] La présence de deux boucles de temporisation (une pour *Système* avec la période *Te* et l'autre pour *Usure* avec la période *Tf*) représente l'idée du fonctionnement simultané de ces deux fonctionnalités. En effet, pendant que le *Système* fonctionne (la transition est franchie) *l'Etat* du composant est périodiquement mis à jour (franchissement de *l'Usure*).

condition suivante : chaque jeton enlevé d'une place tampon doit être remis immédiatement, c'est-à-dire que la durée de son traitement et ainsi de sa disparition de la place tampon doit être impérativement nulle.

Cette condition est interprétée dans Design/CPN par l'interdiction d'utilisation des transitions temporisées en aval des places tampons et des arcs temporisés en amont des places tampons.

Ces places sont particulièrement importantes dans la modélisation de cohabitation des composants synchrones (time-triggered) et des composants réagissant sur événement (event-triggered).

3.5.8 Temporisation synchrone (instants d'échantillonnage)

Pour être conforme à la théorie des systèmes discrets le régulateur et le processus sous contrôle doivent être échantillonnés au même moment.

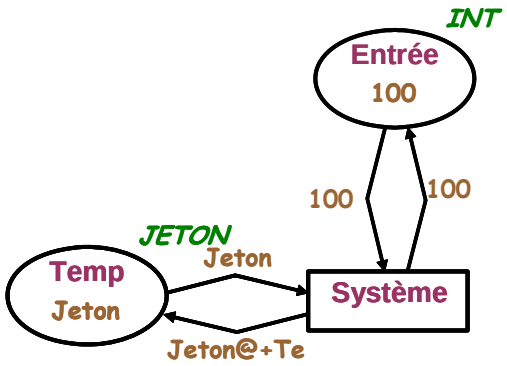


Figure 3.8 : Echantillonnage synchrone

La satisfaction de cette exigence est assurée par la présence exclusive des places tampons en amont des transitions qui sont censées être synchrones. En aval de la transition d'autres places que des places tampons peuvent être utilisées. L'intégration du temps dans le modèle se fait à travers une place *Temp* (figure 3.8) qui initialement contient un *Jeton* (dans le cas où le premier instant d'échantillonnage doit avoir lieu au début de la simulation du modèle).

A l'instant d'échantillonnage synchrone de la transition *Système* les jetons correspondants sont enlevés de toutes les places en amont de la transition. Après le traitement dont la durée est impérativement

nulle, les jetons sont immédiatement placés dans les places de sortie de la transition franchie et également dans les places tampons avec une seule exception qui est la place *Temp*. Le *Jeton* à destination de la place *Temp* sera retardé de *Te* unités de temps (*Te* étant la période d'échantillonnage).

Puisque toutes les places en amont de la transition sauf la place *Temp* ont été déjà réalimentées la seule condition pour que la transition *Système* soit de nouveau valable (et donc franchie) est la présence du *Jeton* dans la place *Temp*, qui est la seule place en amont de la transition qui n'est pas du type tampon. A l'arrivée du *Jeton* dans la place *Temp* la transition *Système* est à nouveau validée est selon le principe des réseaux de Petri temporisés sera franchie à l'instant où elle est devenue franchissable. Ceci implique que la période des franchissements de la transition *Système* est toujours égale à *Te* unités de temps.

3.5.9 Expression de durée (temporisation asynchrone)

Le problème d'une temporisation asynchrone, c'est-à-dire que l'instant de franchissement de la transition suivante dépend du franchissement de la transition précédente, est présent par exemple dans le réseau de communication. Pour pouvoir transmettre il faut entre autre qu'une information à transmettre arrive dans le tampon d'entrée du réseau de communication.

Une fois que cette information est arrivée et le réseau est prêt pour son envoi la transmission elle même dure un certain temps avant d'être placée dans la place tampon qui représente le tampon de sortie du réseau de communication.

Il est donc possible sous le formalisme des réseaux de Petri de faire cohabiter les parties synchrones et asynchrones dans le même modèle.

Design/CPN intègre deux types différents de temporisation soit en associant le temps aux transitions soit aux arcs. Dans nos modèles nous préférons la temporisation des arcs puisqu'elle permet de réduire la taille des modèles. Un type de temporisation peut être facilement transformé vers l'autre type d'une manière semblable à celle de [Dav 97] pour les temporisations des places et transitions.

4 Modèles de base

La composition des systèmes d'automatisation à réseau peut être variable, mais il est néanmoins possible de distinguer certains types ou classes de composants. Le plus simple à identifier est le réseau de communication. Il s'agit du composant cardinal de tout système d'automatisation à réseau. Pour les autres composants il faut faire une distinction entre la fonctionnalité elle-même et l'emplacement physique du composant.

4.1 SYSTEME ECHANTILLONNE

Le système échantillonné représente le composant de base de ce travail. Il est utilisé dans les modèles de tous les régulateurs, capteurs et actionneurs.

Pour la modélisation d'un système échantillonné le point de départ est son expression sous la forme d'équations d'état ou sous sa forme d'équations de récurrence :

$$x(k+1)=Ax(k)+Bu(k)$$
$$y(k)=Cx(k)$$

x – vecteur d'état (n,1)
y – vecteur de sortie (m,1)
u – vecteur de commande (r,1)
A – matrice d'évolution
B – matrice de commande
C – matrice d'observation

Dans la réalisation du modèle d'un système échantillonné il sera considéré que le système lit et écrit les valeurs d'entrée *u* et de sortie *y* dans une mémoire publique (c'est-à-dire partagée avec les autres composants) et stocke les variables d'état *x* dans une mémoire privée.

Le point essentiel de la modélisation d'un système échantillonné est la garantie du respect de synchronisation de l'échantillonnage. Pour cette raison il est préférable de séparer le dispositif de l'échantillonnage de toutes les autres fonctionnalités du système. En plus il faut constamment garantir la possibilité d'accès du système aux mémoires publiques aussi bien en lecture qu'en écriture.

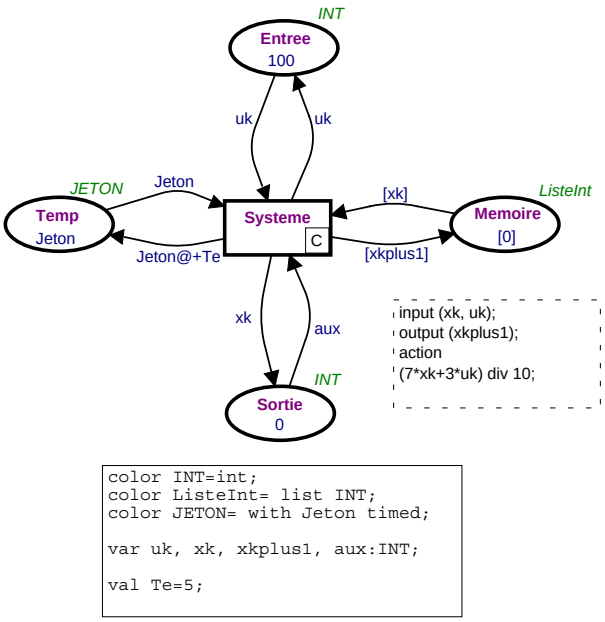


Figure 4.1 : Modèle CPN d'un système échantillonné

Sur la figure 4.1 est présenté un exemple d'un système échantillonné correspondant au système d'équations d'état[3] :

$$x(k+1)= 0.7\ x(k) + 0.3\ u(k)$$

$$y(k)=x(k)$$

Le nœud de déclaration fait partie intégrante de chaque modèle CPN. Il est reconnaissable par sa bordure. Dans cette étude 3 couleurs sont définies : INT, ListInt et JETON. La couleur INT est simplement une activation de l'ensemble des nombres entiers. La couleur ListInt définit l'ensemble des listes composées de nombres entiers. La couleur JETON est créée par énumération de toutes les valeurs qui lui sont propres. Ceci est spécifié par le mot-clé "with". Pour les besoins de la présente étude, la couleur JETON n'est composée que d'une seule valeur *Jeton*. Elle permet ainsi l'équivalence avec les

[3] La version actuelle de Design/CPN ne permet pas de travailler avec des nombres réels. Nous utilisons donc les nombres entiers. Ce dispositif est fonctionnel si les valeurs des variables sont importantes (100 et plus). Dans ce cas, l'influence des arrondis est petite et les résultats sont proches de la réalité.

RdP non-colorés. Le mot-clé "timed" qui est aussi présent dans la définition de cette couleur indique que tout instance créée à sa base, aura un attribut qui lui associe une expression du temps. Cet attribut s'hérite pour tout objet composé de la couleur JETON.

Seules les variables du type INT sont utilisées dans l'exemple développé. Le besoin d'utilisation des variables du type ListInt est évité par la présences des variables INT (xk et xkplus1) qui sont placées dans des crochets. Il n'est nullement nécessaire de préciser l'utilisation de la constante Jeton qui a été initialisée à l'instant de la définition de la couleur JETON.

La place *Entrée* est déclarée du type INT (nombres entiers). Par définition, elle peut alors contenir un nombre quelconque de nombres entiers ou n'en contenir aucun. Si la place *Entrée* est vide elle empêche le bon fonctionnement du franchissement de la transition *Système*. Si son contenu est plus grand qu'un nombre entier, le choix de la valeur d'entrée est aléatoire. Par exemple en cas de deux valeurs présentes dans la place, n'importe laquelle peut être prise comme l'entrée du *Système*. Dans ce cas, le fonctionnement peut aussi être corrompu. Pour garantir que la place va toujours contenir une seule valeur, la condition suivante doit être respectée : à chaque modification du contenu de cette place, une valeur doit y être immédiatement restituée. Pour cette raison il y non seulement un arc menant de la place *Entrée* vers la transition *Système*, mais aussi un arc dans la direction opposée. Dans le cas présent la valeur prise dans *l'Entrée* est associée à la variable *uk*. Puisque la même variable est associée aux deux arcs liés à *l'Entrée*, la valeur de son contenu ne sera pas modifiée par le franchissement de la transition *Système*.

La place *Mémoire* est du type ListInt (liste de nombres entiers). Elle peut alors contenir un nombre de listes quelconque : chaque liste ayant une taille quelconque. Pour des raisons identiques à celles concernant la place *Entrée*, il est impératif qu'elle ne contienne qu'une seule liste. La liste dans cette place sur la figure 4.1 contient une liste composée d'une seule valeur (*0*). L'arc reliant la *Mémoire* avec le *Système* est associé avec l'inscription [*xk*] où *xk* est un nombre entier. Cette construction impose que la liste dans la *Mémoire* soit d'une longueur unitaire. Sinon la transition système devient non validée et son franchissement serait arrêté. D'autres constructions des inscriptions aux arcs permettent d'éviter ce problème.

L'utilisation d'une liste est privilégiée dans cette place pour son développement futur. En effet, les listes permettent une représentation à la fois simple et efficace des variables à mémoriser. Ainsi pour garder en mémoire par exemple trois variables nécessaires pour un système du $2^{ème}$ ordre, la place serait

initialisée avec une liste comprenant 3 valeurs initiales (ex. : [*0, 0, 0*]). L'arc vers la transition *Système* pourrait alors être associé avec l'inscription "[*xk, xkmoins1, ukmoins1*]" et l'arc de retour avec "[*xkplus1, xk, uk*]". Par conséquent le changement de l'ordre du système modélisé n'exige point le changement de la structure du modèle déjà existant.

Le type de la place *Sortie* sur la figure 4.1 est INT. La valeur contenue dans cette place n'est pas importante pour le franchissement de la transition *Système*. Mais pour respecter le principe qu'une seule valeur peut se trouver dans la place *Sortie*, il faut "écraser" ou plus précisément enlever la valeur antérieure. La variable *aux* assure que la valeur contenue dans *Sortie* sera enlevée et ensuite remplacée par la nouvelle valeur de la sortie du *Système xk*.

La couleur de la place *Temp* est du type JETON. Pour le franchissement de la transition *Système*, le *Jeton* est enlevé de la place *Temp*. Il y est retourné avec un décalage par rapport à l'instant de franchissement. Le décalage est de *Te* unités de temps (*tu*). Dans le présent exemple *Te* est fixé à 5 *tu* dans le nœud de déclaration global. Dans le cas du fonctionnement nominal (normal) le *Jeton* apparaît dans la place *Temp* seulement aux instants *k*Te (k=0, 1, 2, ...)*. Dès son apparition à l'instant initial il est pris pour le franchissement du *Système* et immédiatement après son retour dans la place *Temp* il repart de nouveau pour le franchissement du *Système*. C'est le seul élément 'environnant' le *Système* qui est temporisé.

Pour son franchissement la transition *Système* récupère les informations suivantes : *uk* de la place Entrée, *xk* de la place Mémoire et *Jeton* de la place Temp. La variable *aux* enlevée de la place *Sortie* ne sera pas utilisée pour les calculs liés au franchissement de la transition *Système*. Néanmoins, il est à noter que le jeton sera enlevé de la place *Sortie*, qui par conséquent va se retrouver vide au même titre que les places *Entrée* et *Mémoire*, c'est-à-dire momentanément.

A l'instant du franchissement de la transition *Système*, le code qui lui est associé est exécuté. Le segment de code, repérable par sa bordure pointillée et la lettre 'C' placé dans la transition correspondante, est composé d'une spécification des entrées (*Input*), d'une spécification des sorties (*Output*) et d'une partie précisant le programme à exécuter. Cette dernière partie est introduite par le mot ACTION. Dans le cas présent elle est composée d'une seule ligne "*(7*xk+3*uk) div 10;*". Elle est équivalente à l'expression "*0.7*x(k)+0.3*u(k)*", donc à un système discret du premier ordre. *x(k)* et *u(k)* sont des entrées pour la section d'action et sont récupérées sur les arcs entrant à la transition *Système*.

De la même manière *xkplus1*, correspondant à *x(k+1)*, est la sortie calculée dans la dernière ligne de la section d'action et il est attribué aux arcs de sortie de la transition *Système*.

Le système représenté se trouve dans son état initial. *L'Entrée* est fixée à *100*, la *Mémoire* et la *Sortie* sont initialisées à *0* et le *Jeton* est présent dans la place *Temp*. A partir de ce point, le système va évoluer aux instants k**Te (k=0, 1, 2, ...)*. Cette évolution peut être observée par simulation. Dans ce cas le modèle est exécuté et les variables significatives sont stockées soit dans la mémoire soit directement dans un fichier. La deuxième possibilité permet l'exploitation des résultats d'une simulation par d'autres logiciels tels que Matlab ou Scilab [Scilab]. Les résultats de quelques premiers pas d'une telle simulation sont relevés sur la figure suivante (figure 4.2).

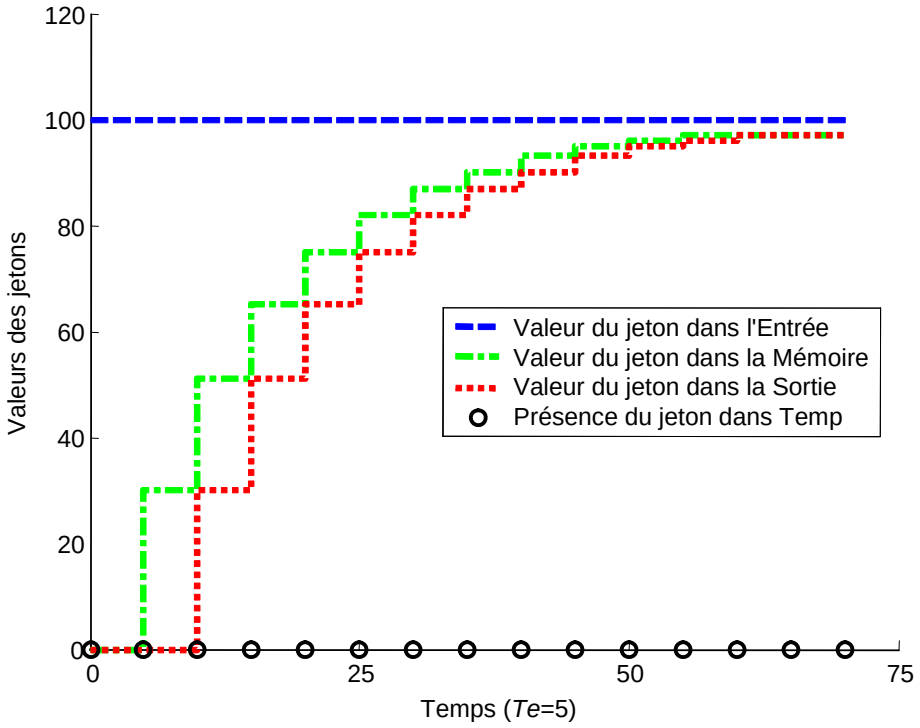


Figure 4.2 : Contenu de chaque place

Le graphe d'occurrence (Occ graphe) est également disponible pour l'analyse du modèle. Ce graphe contient tous les états atteignables à partir de l'état initial. Les nœuds représentent les marquages du modèle (vecteurs dont les éléments expriment le marquage individuel de chaque place). Les arcs reliant les nœuds correspondent aux franchissements d'une seule transition qui implique le changement de l'état global du modèle. Il est à souligner que la taille d'un tel graphe peut être largement supérieure à la

taille du modèle à partir duquel il est créé. La figure 4.3 montre le début du graphe d'occurrence du modèle présenté dans cette section.

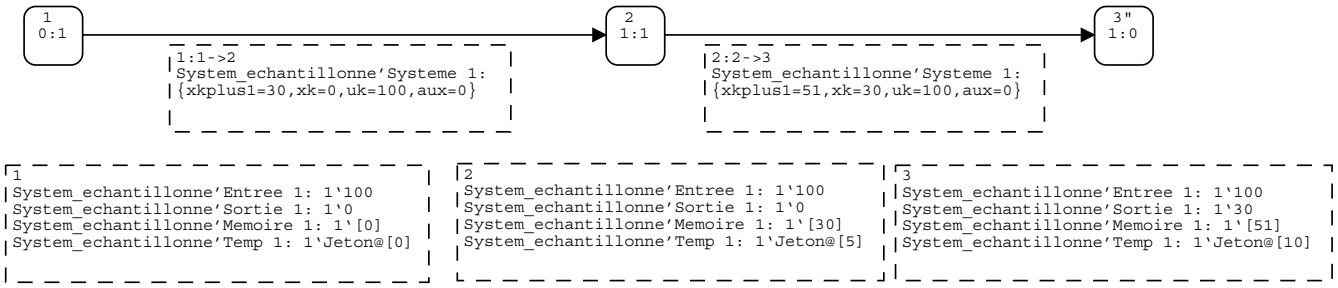


Figure 4.3 : Portion d'un graphe d'occurrence d'un système échantillonné

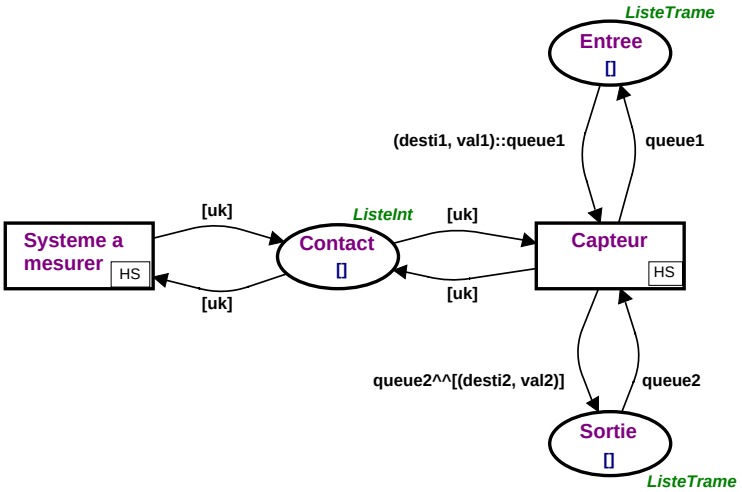
Le graphe d'occurrence montre trois états dans une séquence linéaire. Ceci implique que, à partir de l'état initial 1, le comportement du modèle est déterministe. L'état 1 et suivi par l'état 2. Il n'y a pas d'autres évolutions possibles. L'état 1 possède un successeur et aucun prédécesseur, puisque c'est l'état initial (les informations inscrites à l'intérieur du nœud 1 le sont sous forme *prédécesseurs:successeurs*). Le passage de l'état 1 vers l'état 2 consiste en le franchissement de la transition *Système* du modèle du système échantillonné. Les valeurs des variables utilisées pendant le franchissement sont les suivantes (présenté dans les accolades) : *xkplus1*=30, *xk*=0, *uk*=100 et *aux*=0. Ainsi les valeurs des jetons sont mises à jour et constituent l'état 2 dont les marquages individuels sont *Entrée*=100, *Sortie*=0, *Mémoire*=[30]. La place *Temp* contient un Jeton@[5]. Cette inscription signifie que malgré la présence apparente du *Jeton* dans la place *Temp*, celui-ci ne sera disponible pour un nouveau franchissement qu'à l'instant 5, ce qui correspond à la fin de la première période d'échantillonnage.

Dans le dernier état représenté, l'inscription 3 est suivie par un drapeau <">. Ce drapeau signifie que les successeurs de cet état ne sont pas encore calculés. Par conséquent, cet état peut avoir des successeurs mais n'est nullement contraint. Pour le déterminer, il est impératif de vérifier l'existence de ces successeurs. Dans la suite il sera montré que ce "détail" présente un des problèmes majeurs d'utilisation des graphes d'occurrence.

Dans les paragraphes précédents un simple modèle ainsi que sa simulation et son graphe d'occurrence sont présentés d'une manière très détaillée. Le souci principal était de faire comprendre en profondeur et faire une première connaissance avec les significations des différents objets d'un modèle CPN et faciliter la compréhension de modèles présentés dans les parties suivantes.

4.2 CONCEPTION HIERARCHIQUE ET PREMIER CAPTEUR

La suite présentera les notions de base de la conception hiérarchique d'un modèle CPN. Le but est de montrer une des intégrations possibles par exemple d'un système échantillonné (dans le rôle d'un processus physique) avec un capteur mesurant la valeur atteinte par le processus physique. Pour les besoins de cet exemple, le modèle de système échantillonné présenté auparavant est repris et légèrement modifié.




```
color INT= int;
color ListeInt= list INT;
color JETON= with Jeton timed;
color DESTINATION= string;
color DONNEE= int;
color TRAME= product DESTINATION * DONNEE;
color ListeTrame= list TRAME;

var uk, xk, xkplus1: INT;
var li: ListeInt;
var desti1, desti2: DESTINATION;
var val1, val2: DONNEE;
var queue1, queue2: ListeTrame;

val Te=5;
```

Figure 4.4 : Interconnexion du système et du capteur

Du point de vue macroscopique, il s'agit ici d'une interaction entre deux systèmes indépendants qui sont représentés sur la figure 4.4 en tant que les transitions : "Système à mesurer" et "Capteur". L'interaction entre ces deux transitions est assurée grâce à la place "Contact". Le drapeau "HS" signifie que la transition, qui lui est associée, n'est pas une transition standard, mais qu'elle est une transition de substitution. Un double clic sur une telle transition ouvre une nouvelle page écran sur laquelle un modèle peut être construit. Par défaut cette page contient les places qui sont connectées à la transition

de substitution originale. Pour le "Système à mesurer" une seule telle place existe. Un arc mène de la place "Contact" vers la transition et un autre de la transition vers "Contact". C'est la raison pour laquelle "Contact" est une place d'entrée et de sortie pour cette transition. Il est à noter que la mise d'une nouvelle valeur dans la place de "Contact" écrase sa valeur précédente.

La figure 4.5 présente le modèle d'un système échantillonné décrit dans les lignes précédentes qui est placé non pas comme un système indépendant mais comme une partie prenante d'un système plus complexe. Par rapport à la vue macroscopique du système, ici on se trouve à un niveau plus détaillé à quoi on va se référencer comme le niveau inférieur. Pour mesurer la valeur atteinte par le système, il faut accéder à sa valeur de sortie. C'est pour cela, que la place originalement appelée "Sortie" est renommée pour devenir "Contact". Le reste du modèle reste intact.

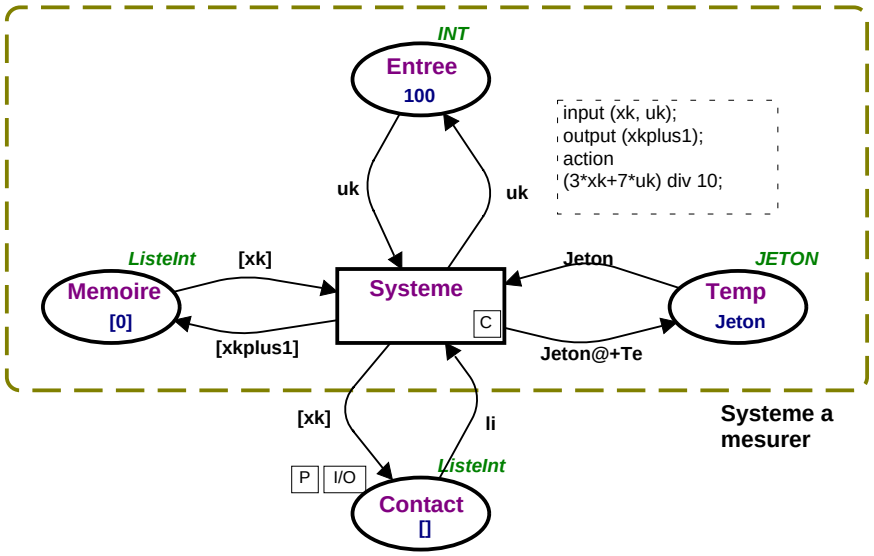


Figure 4.5 : Détail sur un composant partagé – Système échantillonné

Un drapeau P sur la figure 4.5 indique que la place "Contact" est une "porte" vers le niveau supérieur du modèle. De plus, la nature de cette "porte" est précisée comme Entrée/Sortie (drapeau I/O). Bien sûr, la "porte" peut être également du type Entrée (ou Sortie) seulement.

L'autre composant "caché" derrière une transition de substitution sur la figure 4.4 est le capteur. 3 places (Contact, Entrée, Sortie) sont liées dans les deux sens à la transition de substitution. La figure 4.6 montre le détail d'un capteur élémentaire doté d'une capacité de communication.

En général, un capteur peut fonctionner sous deux modes différents :

- time-triggered (mesures envoyées périodiquement),

- event-triggered (mesures envoyées en réaction à un événement).

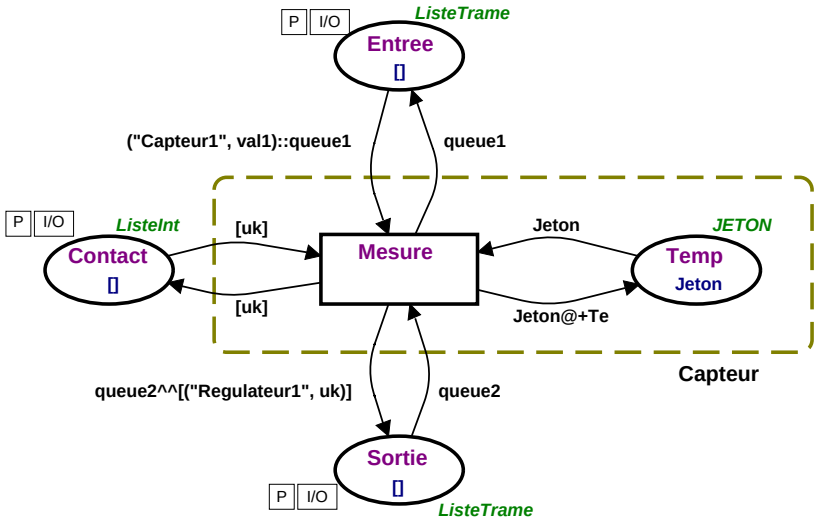


Figure 4.6 : Détail sur un composant partagé – Capteur

Sur la figure présentée (figure 4.6), le capteur est un hybride de ces deux approches. Il envoie la mesure sur demande, mais pas plus souvent que toutes les *Te* unités de temps (selon le nœud de déclaration *Te*=5). La place "Entrée" du modèle du capteur contient une liste composée des différentes trames de communication présentes sur le réseau de communication (celui-ci n'est pas montré sur l'illustration). Une trame de communication dans cet exemple contient une adresse de destination (du type chaîne de caractères) et une valeur numérique (du type nombre entier) qui se traduisent comme des données utiles. Pour qu'une demande puisse être reçue par le capteur, il est impératif que le premier élément de la liste soit égal à "'("Capteur1", val1)". La chaîne "Capteur1" est une constante et la valeur "val1" est une variable. Pour la validation de la transition "Mesure", on demande que l'adresse de la destination de la trame soit égale au "Capteur1" et les données de cette demande seront stockées dans la variable "val1". La "queue1" contient éventuellement d'autres trames présentes dans la place "Entrée". En franchissant la transition "Mesure" le premier élément de la liste est alors séparé du reste de la liste et consommé pendant le franchissement tandis que le reste de la liste est retourné dans "Entrée" pour un traitement ultérieur.

La présence d'une boucle de temporisation entre la place "Temp" et la transition "Mesure" limite le délai minimal entre deux franchissements successifs à *Te*. Comme l'état "Contact" contient la valeur de

sortie du "Système à mesurer", cette valeur est placée dans la variable "uk". Le franchissement de la 'Mesure' forme une nouvelle trame de communication, dans le cas présent c'est la trame '("Regulateur1", uk)' et cette nouvelle trame est ajoutée au contenu actuel de la place de 'Sortie', qu'on appelle 'queue2'. A partir de cette place la trame est prête à être envoyée sur le réseau de communication, ce qui n'est pas montré sur la figure.

La composition hiérarchique facilite largement le travail de conception des modèles. Entre autre, elle permet d'échanger une partie d'un modèle pour une autre. C'est le cas dans la section suivante qui montrera la possibilité de remplacement du simple capteur décrit auparavant, par un capteur un peu plus complexe intégrant notamment des défaillances aussi bien en fonction du temps qu'en fonction du nombre de sollicitations.

4.3 MODELE D'UN CAPTEUR DEFAILLANT

Le capteur présenté dans les paragraphes précédents et sur la figure 4.6 est un capteur simple, qui après avoir reçu une demande de mesure et après vérification du respect d'un délai minimal, récupère la valeur atteinte par le 'Système à mesurer' et la prépare pour l'envoi. Dans cette section nous allons montrer comment un certain nombre de défaillances peuvent être exprimées dans un modèle du capteur. Il est à noter, que par souci de faciliter la compréhension, seuls des systèmes non réparables sont considérés dans le présent chapitre. Néanmoins, les systèmes réparables peuvent également être modélisés par cette méthode.

Par rapport à la figure 4.6 les changements ont été faits. En premier lieu, il s'agit d'un élargissement des définitions et déclarations dans le nœud de déclaration. Seules les lignes ajoutées sont montrées sur la figure 4.7, sachant que toute déclaration et définition de la figure précédente reste valable. Une nouvelle couleur 'Fonctionnement' est définie par énumération de ces deux valeurs : 'Ok' pour symboliser le bon fonctionnement du composant et 'Err' pour représenter sa panne. La couleur 'PROBA' contient les nombres entiers compris entre 1 et 100. Au travers de deux variables 'proba1' et 'proba2' deux types de défaillances seront modélisés. Enfin une nouvelle constante temporelle est définie : 'Tf' (d'une valeur égale à dix unités de temps) est utilisée pour préciser la période avec laquelle évolue l'état fonctionnel du capteur. Dans le modèle 'Te' représente l'écart minimal imposé entre deux échantillons de mesure successifs.

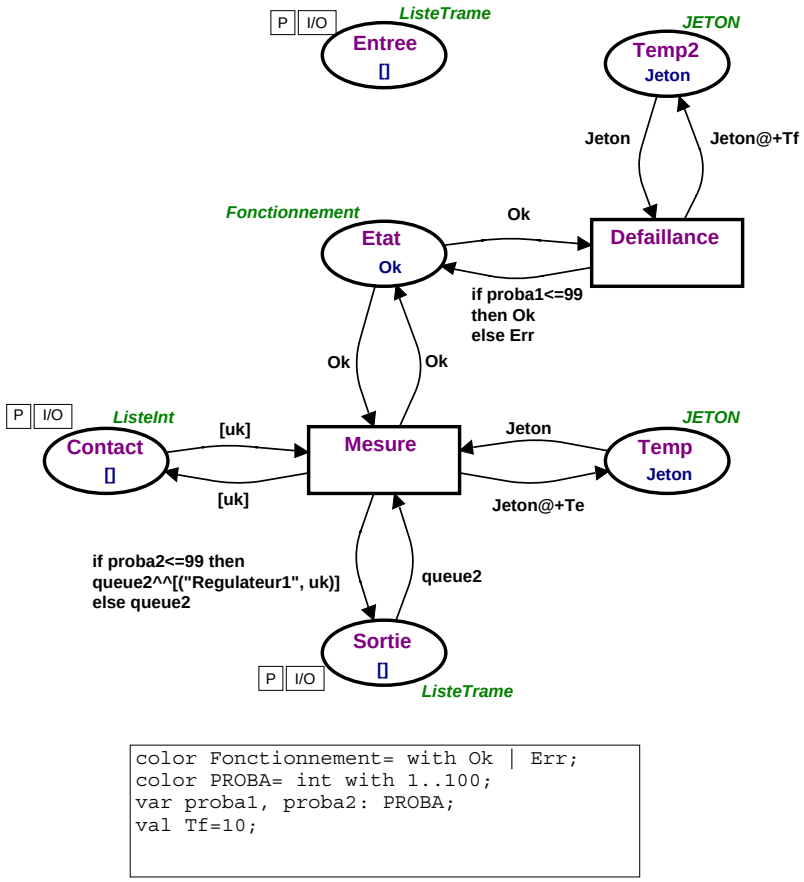


Figure 4.7 : Détail sur un capteur défaillant

Dans le schéma du fonctionnement, le changement principal consiste en l'arrêt de prise en compte de l'entrée du capteur. Pour cette raison, la place 'Entrée' n'est connectée nulle part (aucun arc sortant de la place). Sous condition du bon fonctionnement les mesures établies pourraient donc être préparées pour un envoi périodique avec 'Te' comme période. Le fonctionnement entre la transition 'Mesure' et les places 'Contact' et 'Temp' reste identique à celui de la figure précédente. Les changements touchent principalement l'ajout d'une nouvelle place 'Etat' et d'une modification de l'inscription sur l'arc entre la 'Mesure' et la 'Sortie'. Le premier changement sert à modéliser une panne dont l'occurrence est une fonction du temps de fonctionnement du capteur, le deuxième changement exprime une défaillance où l'apparition est une fonction du nombre de sollicitations.

Pour intégrer la panne qui ne dépend que du taux de défaillance sur un intervalle de temps, la transition 'Défaillance' est employée. Cette transition possède deux états d'entrée et de sortie ('Etat' et 'Temp2'). La place 'Temp2' est utilisée d'une manière semblable à celle de la place 'Temp', c'est-à-dire pour

assurer la fréquence temporelle qui sépare deux évaluations consécutives de l'état de fonctionnement du capteur.

La place 'Etat' contient la représentation de l'état de fonctionnement actuel du capteur. Le jeton de la valeur 'Ok', présent dans l'état initial du modèle, valide aussi bien le franchissement de la transition 'Mesure' que celui de la transition 'Défaillance'. Au franchissement de la transition 'Mesure' ce jeton est pris et aussitôt remis dans l''Etat' sans aucune modification. Pendant le franchissement de la transition 'Défaillance', qui se produit toutes les 'Tf' unités de temps, le jeton est pris et remis en fonction de la valeur de la variable 'proba1'. Cette valeur est automatiquement générée selon la loi uniforme à chaque franchissement de la transition 'Défaillance' par un générateur de nombres aléatoires implémenté dans Design/CPN. Une fois sur 100 la valeur générée est supérieure à 99 (à cause de l'intervalle de la couleur 'PROBA') et l'état fonctionnel du capteur passera de 'Ok' vers 'Err'. Dans les autres cas ('proba1'<=99) le capteur reste en bon fonctionnement. Puisque la 'Défaillance' est franchie toutes les 'Tf', le taux de défaillance du capteur est fixé à 10^{-3} [unités de temps^{-1}].

Pour modéliser la défaillance, dont l'occurrence ne dépend pas du temps écoulé, mais plutôt du nombre de sollicitations, la variable 'proba2' est ajoutée au modèle. Sa définition est identique à celle de la variable 'proba1'. Elle est appliquée sur l'arc qui relie la transition 'Mesure' à la place 'Sortie'. L'inscription est la suivante : 'if proba2<=99 then queue2^^[("Regulateur1",uk)] else queue2'. Comme pour la variable 'proba1', où une nouvelle valeur est générée à chaque franchissement de la transition 'Défaillance', une nouvelle valeur est mise dans 'proba2' à chaque franchissement de la 'Mesure'. Si cette valeur est inférieure ou égale à 99, alors une nouvelle trame sera ajoutée aux trames déjà contenues dans la 'Sortie'. L'ajout de la trame aux trames existantes est représenté comme une concaténation (symbolisée par ^^) de deux listes : queue2, qui contient les trames déjà présentes dans la 'Sortie', et une nouvelle trame, dont le destinataire est le "Régulateur1" et les données expriment la valeur mesurée 'uk'. Si la valeur calculée pour la 'proba2' est supérieure à 99 alors le contenu original, c'est-à-dire 'queue2' inchangé, est remis dans la 'Sortie'.

Tandis que la première défaillance décrit l'usure du capteur ou un endommagement de son corps de mesure qui par leur nature est irréversible, la deuxième défaillance exprime l'échec du capteur à envoyer la dernière mesure obtenue. Cette défaillance est considérée comme ponctuelle. Alors son apparition ne doit pas empêcher le capteur de fonctionner et celui-ci doit retenter sa chance au prochain instant d'échantillonnage où la défaillance peut mais ne doit pas forcément réapparaître.

Le capteur décrit dans ce chapitre est largement utilisé dans nos travaux. Sa caractéristique principale est l'intégration de deux classes de défaillances dans le modèle d'une manière facile à comprendre et également à modifier en cas de besoin. Grâce à la constitution des places 'Entrée' et 'Sortie' comme des "portes" vers le niveau supérieur du modèle il peut être connecté au réseau de communication. Un exemple d'un réseau est présenté dans la partie suivante.

4.4 MODELE D'UN CANAL DE COMMUNICATION

Dans les sous-chapitres précédents, le modèle d'un capteur défaillant ainsi que le modèle du système à mesurer par le capteur sont présentés. Le capteur contient deux places ('Entrée' et 'Sortie') qui agissent comme les tampons (« buffers ») permettant la connexion du capteur au réseau de communication. Un réseau qui permet le transport de l'information, sous sa forme simplifiée, appelée "canal", est présenté dans la suite.

L'interconnexion du capteur et du réseau de communication se fait d'une manière tout à fait semblable à la jonction entre le capteur et le système à mesurer. Des places 'Entrée' et 'Sortie' sont partagées par les deux composants. La transition 'Canal' est rajoutée au modèle initial, qui est montré sur la figure 4.4. Il s'agit d'une transition de substitution, qui "cache" le modèle du canal de communication. A ce point, il faut souligner la facilité de la conception hiérarchique du modèle. Le modèle ainsi construit est présenté sur la figure 4.8. Il est composé d'un système à mesurer, d'un capteur et d'un canal de communication. En cas de besoin, l'utilisateur peut accéder aux détails cachés par les transitions de substitution, mais le niveau hiérarchique le plus élevé permet de vite comprendre la composition générale du système. Cette facilité de compréhension est accentuée par la présence d'une page hiérarchique qui récapitule les liaisons entre différentes parties, appelées "pages", du modèle global. L'exemple d'une telle page, pour le modèle représenté sur la figure 4.8, est montré sur la figure 4.9.

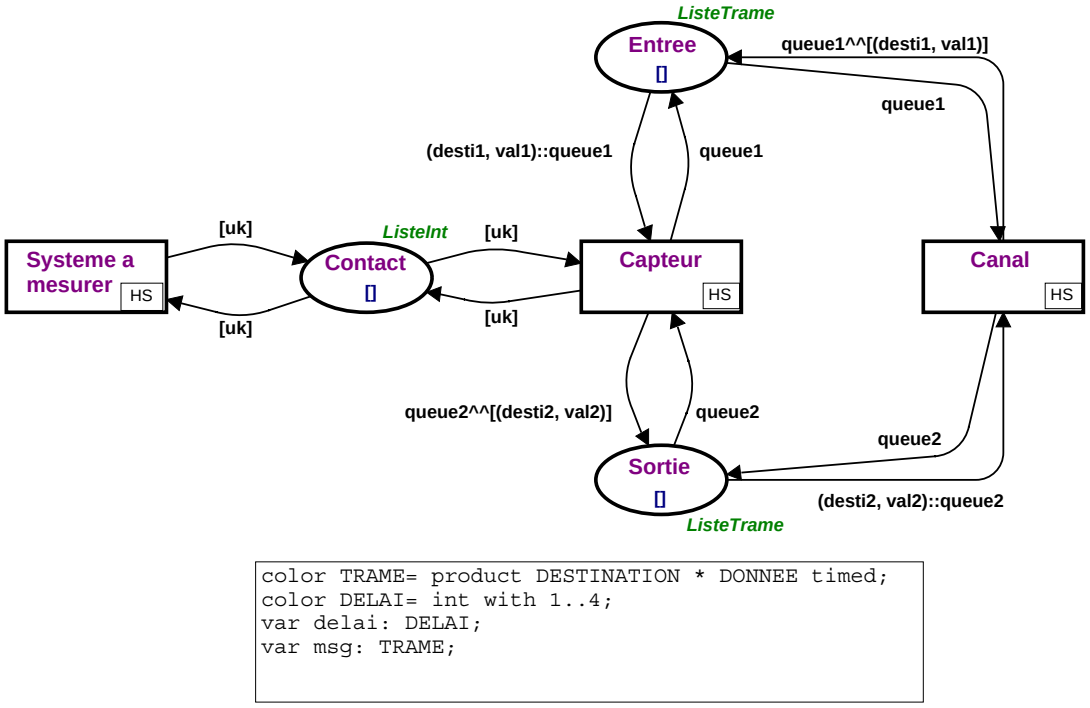


Figure 4.8 : Interconnexion du capteur et du canal de communication

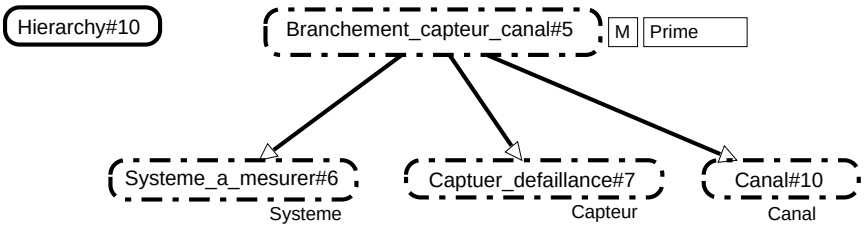


Figure 4.9 : Récapitulation de la hiérarchie du modèle

Pour pouvoir réaliser cet agrandissement du modèle, le nœud de déclaration doit être modifié et élargi. Sur la figure 4.8 seules les modifications et ajouts sont montrés. Le nœud complet peut être obtenu en combinant le présent schéma et ceux des figures 4.4 et 4.7. La première modification concerne la définition de la couleur 'TRAME', qui à partir de maintenant est temporisée (*timed*). Ceci est nécessaire pour exprimer la durée de transmission non nulle pour chaque trame. Cette couleur est définie comme un produit cartésien de la 'DESTINATION' et de 'DONNEE'.

Une couleur supplémentaire est définie : 'DELAI' est composée de nombres entiers compris dans l'intervalle 1 à 4. La variable 'delai', définie sur cette couleur, est utilisée pour exprimer la durée de transmission d'une trame.

En dernière ligne de cette section du nœud de déclaration, la variable 'msg' du type 'TRAME' est créée. L'ajout de cette variable facilite la représentation de la trame transférée et augmente le confort du travail de modélisation.

Le principe de fonctionnement du canal de communication est illustré sur la figure 4.10. Le canal de communication est démarqué par deux tampons (d'émission et de réception). Les deux sont des files d'attente classique du type FIFO (First In, First Out, premier venu, premier servi). Entre les deux il y a le bus, sur lequel la transmission dure un certain temps. Pour le moment, le bus considéré ne modifie ni ne perd de messages transmis. La présente définition suppose, à part l'existence des tampons globaux qui n'est pas un phénomène généralisé, que tous les émetteurs peuvent écrire à tout moment dans le tampon d'émission et tous les récepteurs peuvent lire dans le tampon de réception.

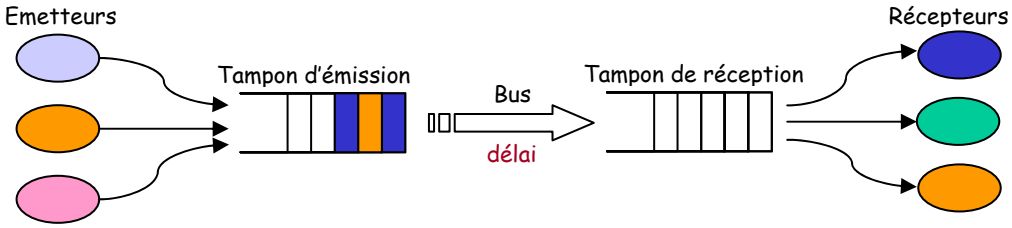


Figure 4.10 : Fonctionnement du canal de communication

Il faut noter que le sens de génération de l'information jusqu'à maintenant est du haut vers le bas. Ainsi, la place 'Entrée' du capteur est en haut et 'Sortie' en bas. Néanmoins, le rôle du canal est de transporter l'information produite par des composants vers les entrées des autres composants. Cela implique que l'entrée du canal doit se trouver en bas et sa sortie en haut. Pour ne pas le confondre, la place 'Sortie' de la figure 4.8 est renommée pour devenir 'C_entrée' sur la figure 4.11 et la place 'Entrée' devient alors 'C_sortie' dans cette portion du modèle.

La place 'C_entrée' contient donc le tampon d'émission, c'est-à-dire toutes les trames qui doivent être transportées par le canal de communication. Le tampon est représenté par une liste, qui à l'instant initial se trouve vide ([ ]). Pour valider la transition 'Transmission', il est impératif que le contenu de la place 'C_entrée' puisse s'inscrire sous la forme 'msg::queue1', ou 'msg' est une trame de communication et 'queue1' une liste contenant le reste des trames. Elle peut être vide si 'Sortie' ne contient qu'une trame à communiquer. L'autre condition de validation de la transition 'Transmission' est que la place 'Bus'

contienne une liste vide, ce qui veut représenter l'état dans lequel le canal est libre et disponible pour une transmission.

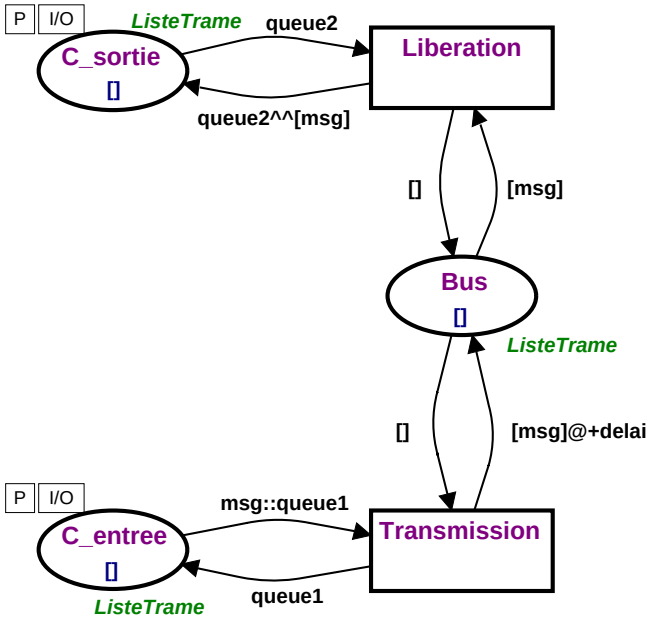


Figure 4.11 : Canal de communication

Quand la transition 'Transmission' est franchie, 'queue1' est remis dans la place 'C_entrée' et une liste contenant une seule trame 'msg' (la liste est alors égale au '[msg]') est mise dans la place 'Bus'. Il reste à déchiffrer la signification de l'inscription '@+délai'. Le symbole '@' dans Design/CPN représente le temps actuel. '@+délai' signifie alors, que '[msg]' sera mis dans 'Bus' seulement 'délai' unités de temps après l'instant du franchissement de la transition 'Transmission'. Puisque aucune valeur précise n'est associée à la variable 'délai', une valeur aléatoire comprise entre 1 et 4 (voir le nœud de déclaration, figure 4.8) sera générée à chaque franchissement de la transition correspondante. C'est ainsi que la durée aléatoire de la transmission est représentée.

Une fois que le message se trouve dans 'Bus', la transition 'Transmission' n'est plus validée et les autres trames doivent attendre jusqu'à la libération du 'Bus' qui se fait en franchissant la transition 'Libération'. Son franchissement remet une liste vide dans 'Bus' et par conséquent la 'Transmission' devient de nouveau franchissable. La liste '[msg]' est par concaténation ajoutée à la fin du contenu actuel de la place 'C_sortie'. Ainsi s'achève la transmission d'une trame de communication par le canal considéré.

Dans les paragraphes précédents un certain nombre de composants d'un système d'automatisation sont décrits d'une manière très détaillée avec le souci d'une compréhension approfondie par le lecteur. Le modèle du système d'automatisation complet doit encore contenir les modèles des actionneurs et des régulateurs. Leurs développements sont laissés à l'initiative du lecteur, sachant qu'ils sont très semblables aux composants déjà présentés. Dans la suite, certains autres modèles modifiés seront introduits pour les besoins de chaque étude du fonctionnement du modèle du système d'automatisation.

5 Application de la méthode à l'usure des composants

5.1 INTRODUCTION

L'application, sur laquelle cet étude est présentée, contient la commande du niveau du liquide dans une cuve. La mission du SAR est de remplir et de maintenir le niveau désiré dans la cuve. Une mesure analogique est alors prise par le capteur. Sa valeur est envoyée via le réseau de communication au régulateur qui détermine la nouvelle commande et la communique à l'actionneur qui représente une pompe placée à l'entrée de la cuve et qui gère le flux d'entrée à la cuve. Une perturbation supplémentaire (fuite ou évaporation) agit également sur le système et elle n'est pas mesurée. Dans l'état initial la cuve est vide et prête à être remplie. La valeur finale désirée du liquide est fixée à 500 unités de hauteur (*hu*).

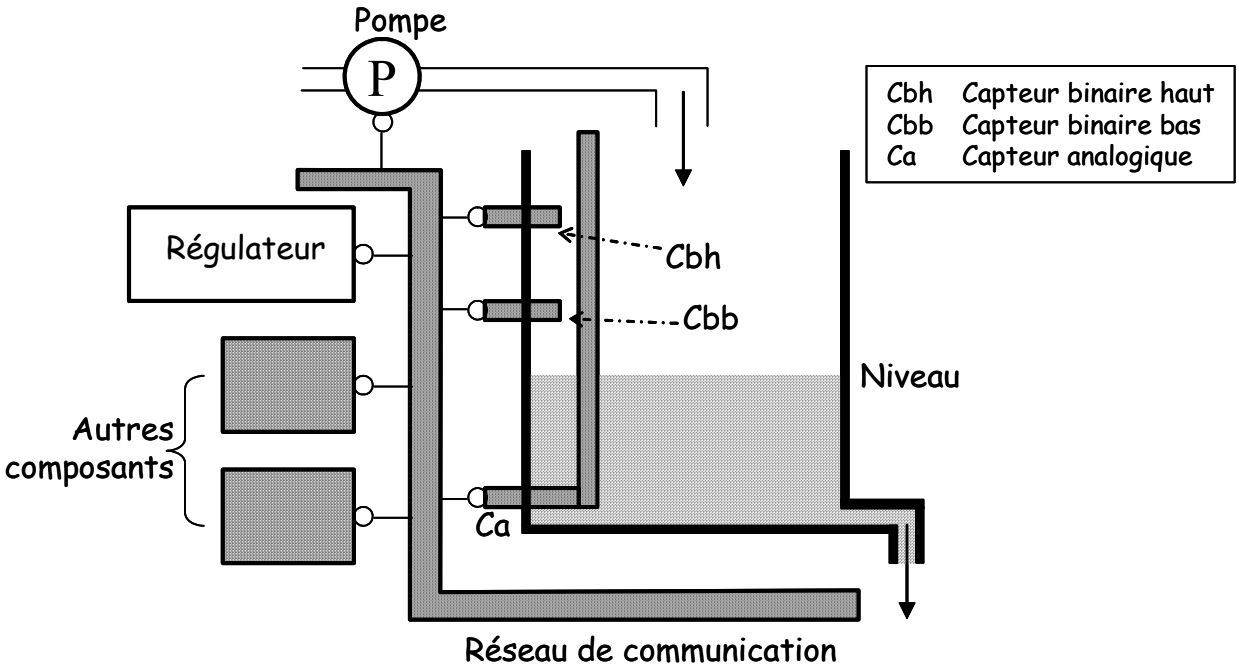


Figure 5.1 : Schéma du système étudié

Pour pouvoir assurer son fonctionnement, le SAR (figure 5.1) est composé de :

- un capteur analogique de niveau,

- deux capteurs binaires,
- un régulateur,
- un actionneur,
- un réseau de communication,
- d'autres composants éventuels générant un trafic supplémentaire sur le réseau de communication.

Différentes défaillances peuvent apparaître sur ces composants. Leur choix est arbitraire et seulement les cas non réparables sont pris en compte dans cette étude. La suite présente une description détaillée de fonctionnement de tous les composants.

Trois capteurs sont utilisés pour mesurer le niveau : un capteur analogique et deux capteurs binaires, qui assurent les mesures en redondance avec le capteur analogique. Les données obtenues du capteur analogique sont utilisées pour déterminer la nouvelle valeur de la commande calculée par l'algorithme de commande nominale, principalement du type P.

Le régulateur est séparé aussi bien des capteurs que des actionneurs et l'ensemble des composants est relié par un réseau de communication. D'autres composants pourraient éventuellement être branchés sur le même réseau, si l'étude portait sur l'influence de la charge du réseau sur la qualité de la commande.

Pour agir sur le niveau, une pompe est disponible à l'entrée du liquide dans la cuve. A la sortie de la cuve, il y a une ouverture pour modéliser une fuite aléatoire. Ce dispositif représente la perturbation à la commande.

Le réseau de communication est défini comme un canal par lequel transitent des trames contenant des données sur le système. Des queues FIFO sont à l'entrée et à la sortie de ce canal.

Cette étude s'intéresse à l'influence de l'algorithme de commande sur l'usure et par conséquent sur la disponibilité de l'actionneur. Deux algorithmes sont étudiés : 1.) un algorithme P pour le fonctionnement nominal et 2.) un algorithme Tout-Ou-Rien pour le cas de la défaillance du capteur analogique. Dans cette étude, une seule défaillance est considérée. Tous les composants à l'exception du capteur analogique sont considérés non défaillants. Ceci par souci de mettre l'accent sur l'influence de l'apparition d'un événement connu auparavant.

Tous les composants sont représentés comme des systèmes échantillonnés avec *Te* comme période d'échantillonnage. Dans l'ensemble de cette étude la période d'échantillonnage *Te* est constante et égale à 10 unités de temps (*tu*). Le capteur, l'actionneur et le régulateur sont des composants discrets avec cette période d'échantillonnage. Le réseau de communication est un système réagissant aux événements discrets. Pour assurer la coexistence de ces deux types de composants, il est recommandé d'utiliser les places tampons à l'entrée et à la sortie du réseau de communication.

5.2 MODELES DES COMPOSANTS

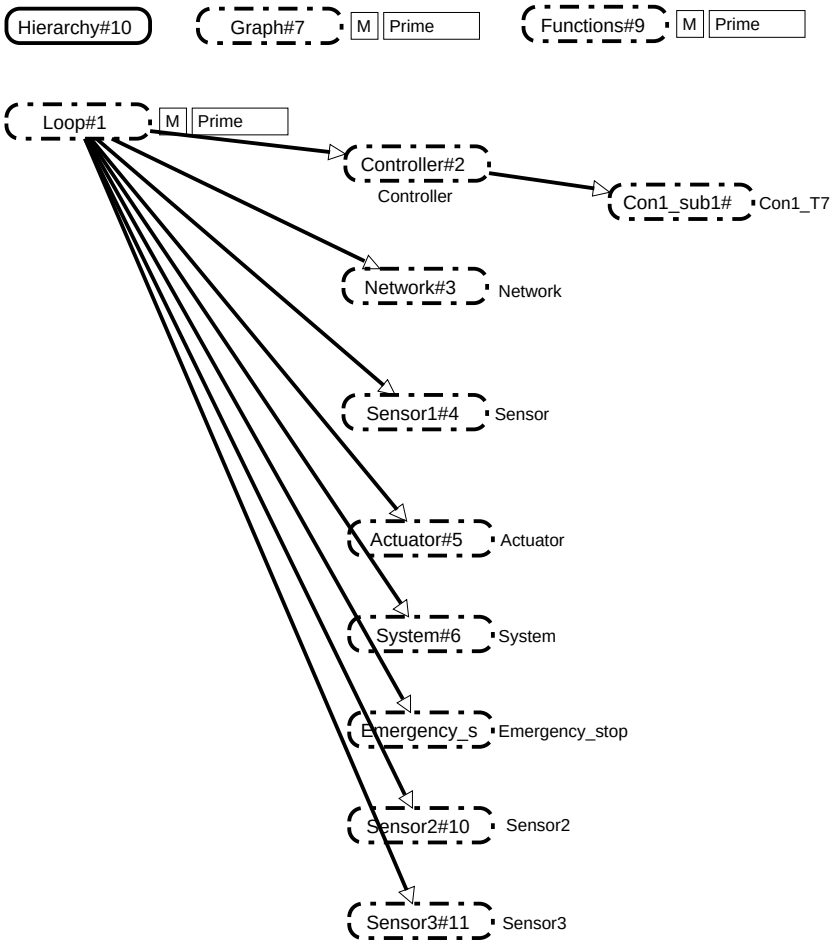
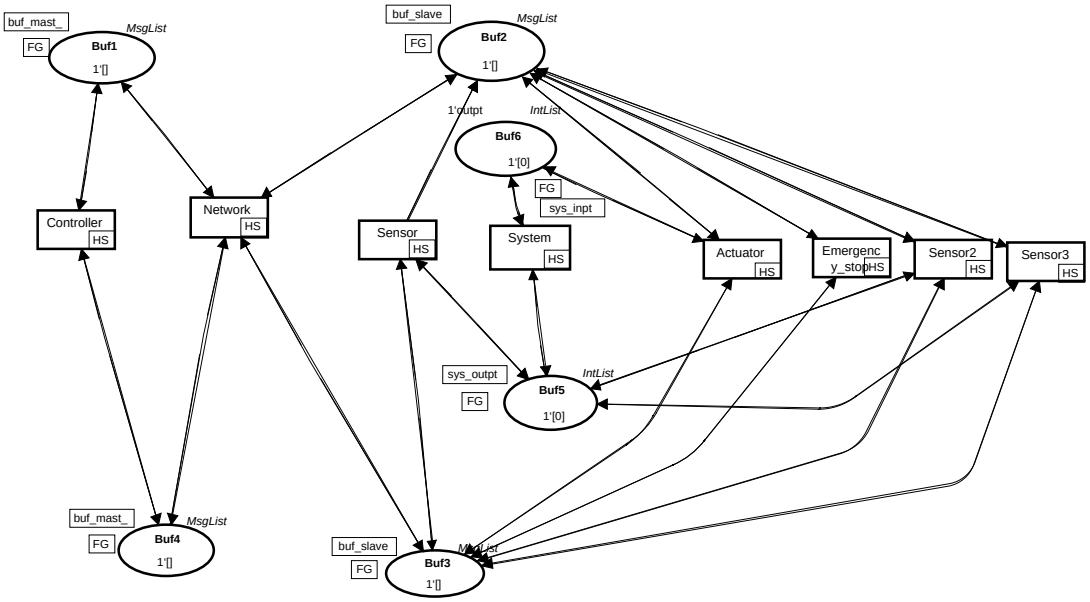


Figure 5.2 : Hiérarchie du modèle

Dans cette partie une description globale du modèle CPN est donnée et les modèles de plusieurs composants du système automatisé sont présentés. Le modèle est conçu d'une manière hiérarchique et sa composition est relevée sur la figure 5.2.

Notre modèle est composé donc d'une page de la spécification de sa hiérarchie, d'une page, qui contient les graphes de simulation, d'une page des fonctions et du modèle d'un système automatisé « Loop ». Cette dernière fait l'usage de plusieurs sous-pages contenant les modèles de tous les composants du SAR et aussi le modèle du processus « System ». La page « Loop » est reportée sur la figure 5.3.




```
color INT=int timed;
color PRB=int with 1..100;
color STRING=string timed;
color DATA= product INT*STRING;
color ADR=string timed;
color MSG=product ADR*DATA;
color MsgList=list MSG;
color IntList=list INT;
color AdrList=list ADR;

var inpt,outpt:MsgList;
var rest,r,r1,r2,r3,r4:MsgList;
var a,b,c,d: MSG;
var i1,i2,i3,i4:INT;
var xk,uk,xk_1,xkpl1:INT;
var s1,s2,s3,s4:STRING;
var proba:PRB;
```

Figure 5.3 : Contenu de la page « Loop »

Sur cette page on peut voir tout d'abord une partie de nœud de déclaration qui contient les définitions des couleurs et des variables utilisées dans le modèle. Outre ce nœud on y trouve également 8 transitions, dont « Controller », « Network » et « Sensor3 ». Ce sont toutes des transitions de substitution qui 'cachent' chacune un modèle représenté par une sous page. Ces sous pages, ou autrement dit ces composants, sont connectées entre elles grâce aux places « Buf1 » à « Buf6 ». Ces places tampons servent à partager et à échanger les informations entre les composants. C'est le cas par exemple pour la place « Buf5 » qui contient la valeur de sortie de processus et tous les capteurs y sont connectés.

A titre d'exemple nous montrons à présent deux des composants du système automatisé. Premièrement, il s'agit du modèle du processus. On sait à présent qu'il est connecté aux capteurs à l'aide de la place « Buf5 ». Par élimination on peut attribuer la place « Buf6 » à l'entrée du processus. A part ces deux places et une transition, le modèle contient encore une place « Sys_Tim » pour le temporisateur et une place « Sys_P1 » pour la mémoire interne. Le fonctionnement de ce modèle est expliqué au Chapitre 4.

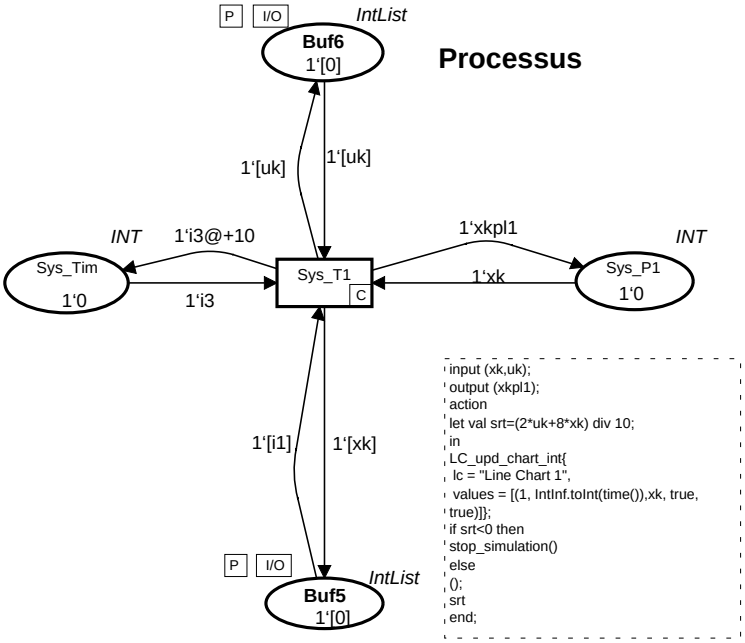


Figure 5.4 : Modèle du processus

Le deuxième modèle présenté est le réseau de communication. Ce réseau comporte deux canaux de communication. L'un dédié à l'envoi des messages du contrôleur aux autres composants. Ceci est fait entre les places « Buf1 » et « Buf2 » en franchissant la transition « Net_T1 ». L'autre canal de communication est dédié aux réponses des composants au contrôleur entre les places « Buf3 », « Buf4 » et la transition « Net_T2 ». La place « Net_P1 » exprime la spécification du protocole utilisé : Le message du contrôleur envoyé à un autre composant réserve le réseau, le marque comme occupé et à ce moment seulement une réponse du composant peut le libérer. Sur le modèle présenté, on ne vérifie pas si la réponse provient du bon composant. Toute réponse d'un composant libère le réseau.

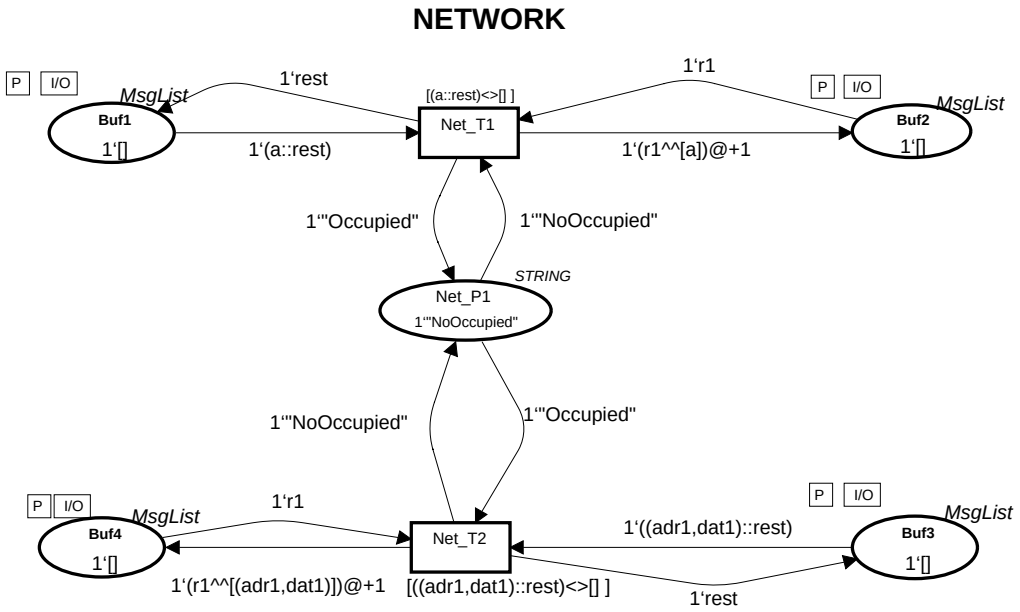


Figure 5.5 : Modèle du réseau de communication

5.3 USURE DES COMPOSANTS

5.3.1 Redondance capteurs

Le critère étudié dans le chapitre présent porte sur la manière d'utiliser la pompe.

On considère une pompe, dont le fonctionnement est garanti jusqu'à un certain volume de pompage et sort de la garantie dès que cette limite est atteinte.

Comme il est expliqué auparavant, un capteur analogique défaillant est présent dans cet exemple. Dans le cas du bon fonctionnement, un algorithme proportionnel est utilisé pour calculer la valeur de la commande. Dans le cas contraire, un algorithme de repli basé sur une régulation Tout-ou-Rien (TOR) est activé. La figure 5.6 montre un exemple de résultat de simulation d'un tel système sans l'influence des perturbations pour un système stabilisé à 200 *hu* et les capteurs binaires placés à 150 et 250 *hu*. Dans la suite de l'étude les perturbations agissant sur le niveau sont considérées.

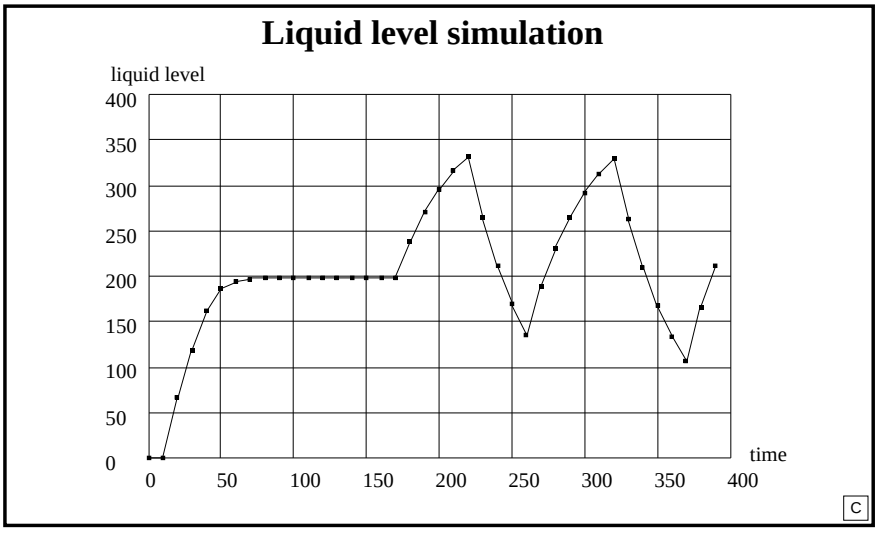


Figure 5.6 : Résultat d'une simulation

Après avoir conçu le modèle CPN fonctionnel du système considéré, une possibilité de défaillance est ajoutée au modèle du capteur analogique. A l'instant initial de la simulation l'actionneur est en état de bon fonctionnement. L'occurrence de la défaillance dépend de la loi exponentielle. Dans le cas d'une défaillance l'algorithme TOR assure le fonctionnement en mode dégradé grâce aux deux capteurs binaires.

D'autres paramètres du modèle sont :

période d'échantillonnage *Te*	1 *s*
conditions initiales	0
valeur désirée	500 *hu*
perturbations	0 – 10 par *Te*
position du capteur haut	600 *hu*
position du capteur bas	400 *hu*
fonctions de transfert (3 capteurs)	1
taux de défaillance	1000 *s*
délai réseau (une trame)	0.1 *s*
taux de pertes trames	0

Différents critères peuvent être évalués sur ce modèle. Cette étude porte sur l'usure de l'actionneur.

5.3.2 Critères d'évaluation

La simulation de Monte Carlo est utilisée pour l'analyse du comportement du système. Elle consiste en un grand nombre de répétitions de simulations individuelles. Le point de départ est toujours l'état initial. L'arrêt de la simulation peut se faire selon des critères variés. Par exemple la durée maximale de chaque simulation peut être définie. Dans ce cas les données obtenues sont censurées à droite. L'impact de la censure sur l'ensemble des données obtenues est une fonction de la durée de simulation choisie. Pour éviter la censure il est possible de continuer avec les simulations jusqu'à ce que l'événement étudié se produise. L'inconvénient de cette proposition est l'existence du cas limite où la défaillance ne se produit pas et donc la simulation devrait continuer indéfiniment. La durée de chaque simulation individuelle dans cette étude est choisie pour que le volume des données censurées ne dépasse pas 5% de la totalité des données disponibles.

Pendant chaque simulation un certain nombre d'informations est mémorisé et ensuite sauvegardé dans un fichier pour une analyse *a posteriori*. Parmi ces informations se trouve notamment la date d'occurrence de la défaillance du capteur analogique. L'autre événement important est la date de l'application de l'algorithme de régulation dégradée. Ces deux dates peuvent être voisines mais la situation peut se produire que malgré la défaillance du capteur, l'algorithme de régulation dégradée n'est appliqué que très tardivement voire jamais. C'est notamment le cas si après une défaillance du capteur analogique les perturbations sont minimales et le niveau du liquide reste longtemps dans l'intervalle entre les deux capteurs binaires utilisés par la commande de repli.

Le critère évalué dans cette étude est l'usure de l'actionneur. Elle peut être définie de différentes façons. Dans le présent exemple elle est définie comme la somme de changement de taux de pompage, c'est-à-dire $sum|(u_{i+1} - u_i)|$, où u_i est le taux de pompage à l'instant i. Une autre possibilité est de la définir comme la totalité du volume pompé, $sum|u_i|$. Par conséquent les différences $u_{i+1} - u_i$ sont également relevées pendant la simulation.

Il est intéressant de noter que le modèle propose aussi la sauvegarde de l'évolution du niveau de liquide en fonction du temps pendant toute la simulation. Ce renseignement pourra servir par exemple pour distinguer entre les modes transitoires et stationnaires, comme c'est le cas dans l'étude du chapitre 6.

5.4 RESULTATS DE MONTE CARLO

Après les simulations on obtient un ensemble de données censurées. Dans un certain nombre de cas le capteur analogique n'est pas tombé en panne. Dans d'autres cas, l'algorithme de repli n'a jamais été utilisé, ce qui est dû à la grande inertie du système.

La figure 5.7 montre une réponse indicielle (avec perturbations) choisie dans l'ensemble des simulations effectuées. La phase transitoire pendant laquelle la cuve est remplie n'est pas visible pour une simple raison, que sa durée est très petite vis-à-vis de la durée totale de simulation. La deuxième phase, maintien du niveau grâce aux informations fournies par le capteur analogique, est bien identifiable entre 0 et 8400 *Te*. Elle est limitée par un trait vertical. Cette phase est immédiatement suivie par l'application de l'algorithme de repli qui utilise les informations en provenance de deux capteurs binaires. Cette phase s'arrête (deuxième trait vertical) à la fin de la simulation à 10800 *Te*. La défaillance n'est pas représentée sur cette image, mais elle est apparue autour de 1200 *Te*.

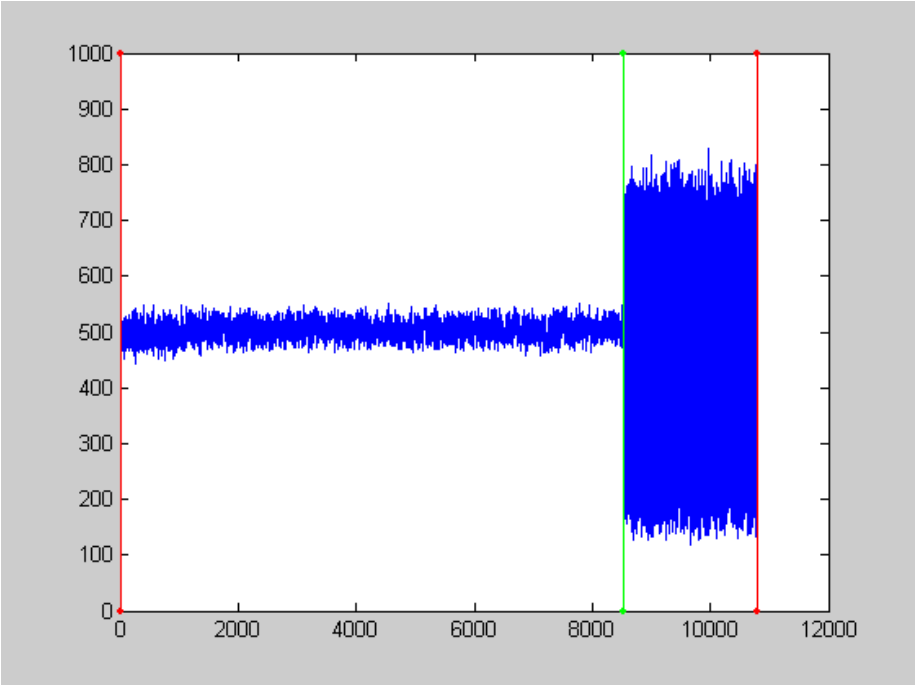


Figure 5.7 : Exemple d'une réponse indicielle

Les défaillances sur le capteur analogique ont été générées selon la loi exponentielle avec le taux de défaillance lambda=2 heures^{-1} et donc le MTTF= 1/2 heure. Il est impératif qu'après la simulation de

Monte Carlo on retrouve ces mêmes valeurs pour la défaillance. Si tel n'était pas le cas, cela remettrait en question la façon de modéliser la défaillance.

Le MTTF théorique est donc ½ heure. Le MTTF obtenu par la simulation et après avoir censuré les données est 0.49357 heure. Pour l'intégration des données censurées dans l'estimation du MTTF voir [Dod 99]. Alors il est possible de constater que les deux MTTF sont proches. Il est encore possible de vérifier que le MTTF simulé observe bien la loi exponentielle. Les deux tests principaux sont le test du khi-deux et le test de Kolmogorov-Smirnov [Vil 88]. [Dod 99] propose plusieurs méthodes graphiques simples pour la vérification de l'ajustement par la loi exponentielle qui sont applicable à l'ensemble des mesures disponibles y compris les données censurées. Un intervalle de confiance pour la valeur de lambda pourrait également être calculé, mais il faut prendre en compte un grand nombre d'occurrences de défaillances ce qui rend le calcul délicat.

La distribution du MTTF et des caractéristiques en dépendant sont relevées sur la figure 5.8.

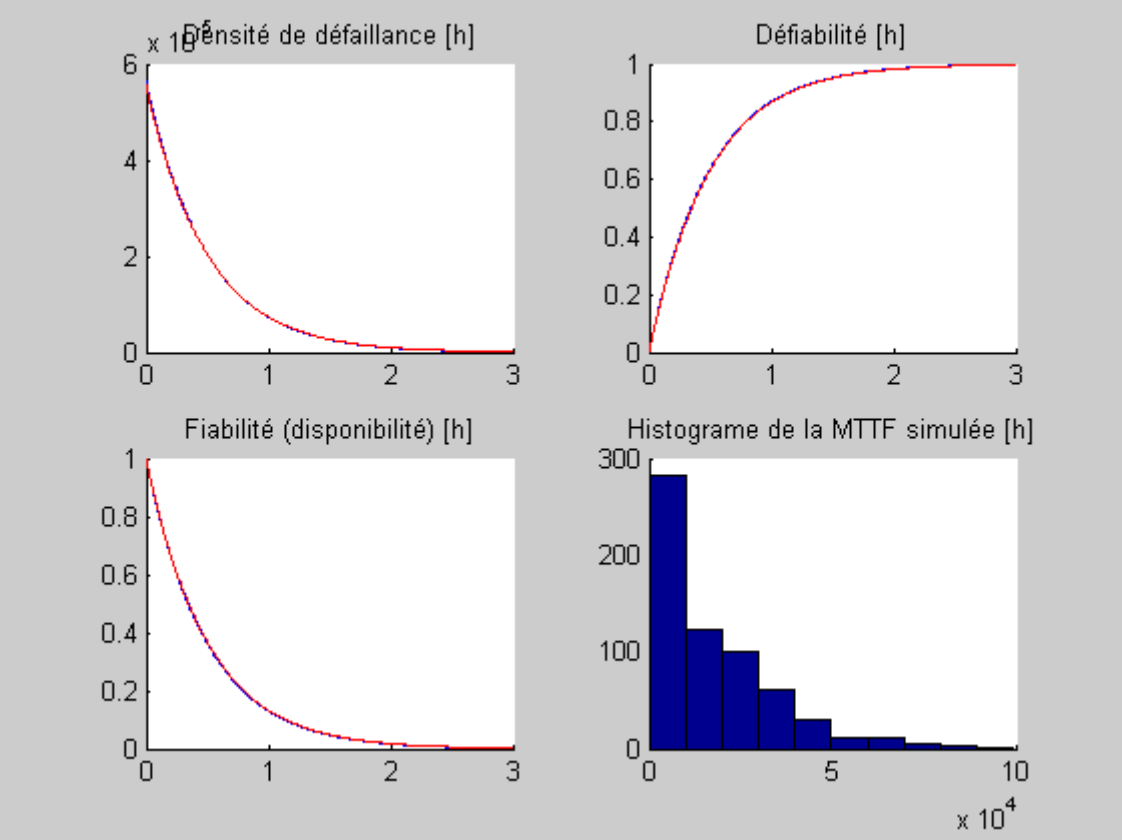


Figure 5.8 : Distribution de taux de défaillance

Pour faciliter les analyses suivantes, tous les cas où l'algorithme de repli n'a jamais été appliqué sont enlevés des résultats et ne sont plus considérés dans la suite. Bien sûr, une méthode alternative de traitement des données censurées pourrait être utilisée.

Le but de cette étude est d'estimer le taux d'usure de la pompe en fonction de l'état global du système. Du point de vue contrôle/commande, deux états peuvent être distingués :

- transitoire, pendant le remplissage de la cuve,
- stationnaire où le niveau est maintenu constant.

Le taux d'usure de l'actionneur varie entre autre en fonction de ces deux états. L'état stationnaire est considéré comme l'intervalle où le niveau est constamment maintenu entre le niveau bas (400 *hu*) et le niveau haut (600 *hu*). L'état transitoire est le complément de l'état stationnaire.

Du point de vue de la sûreté de fonctionnement, une autre classification des états s'impose. Elle est une fonction de la configuration des composants en bon fonctionnement et comporte deux états :

- état de bon fonctionnement du capteur analogique,
- état de panne du capteur analogique.

Ces deux états conditionnent également le taux de l'usure de l'actionneur.

Ainsi le taux d'usure peut être classé dans une des quatre catégories :

- transitoire et bon fonctionnement du capteur analogique,
- stationnaire et bon fonctionnement du capteur analogique,
- stationnaire et défaillance du capteur analogique,
- transitoire et défaillance du capteur analogique.

Il est à noter que le taux d'usure n'est pas exclusivement une fonction de ces quatre états, mais d'autre facteurs d'influence affectent sa valeur (amplitude des perturbation, écart entre consigne et le niveau atteint, etc.). Pour simplifier l'étude, il ne sera pas considéré le taux d'usure instantané mais le taux moyen.

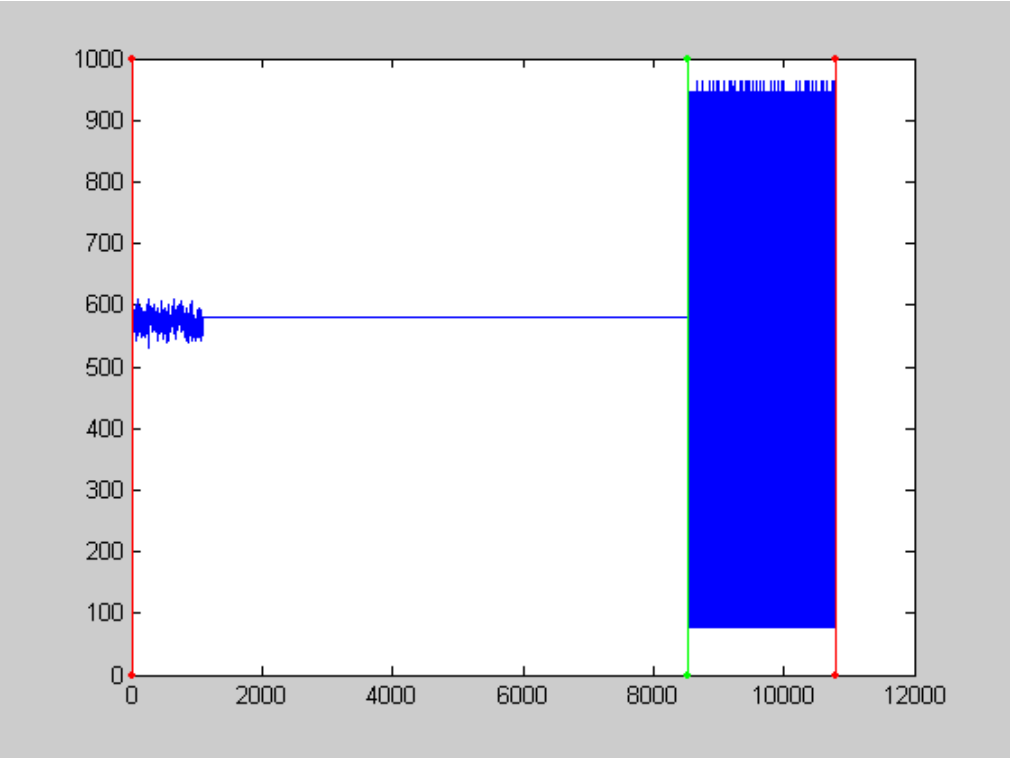


Figure 5.9 : Exemple de la commande (l'entrée de l'actionneur)

La figure 5.9 montre un exemple de la valeur de commande appliquée à l'actionneur. C'est la commande correspondante à la réponse indicielle présentée sur la figure 5.7. On peut y distinguer la phase stationnaire avec le bon fonctionnement du capteur analogique et la phase où ce capteur est en panne. Dans la première phase, il faut remarquer la présence d'un intervalle pendant lequel le taux de pompage ne change pas. Ce phénomène se traduit par une ligne horizontale sur la figure. La défaillance s'est produite autour de 1200 *Te*. Mais l'algorithme de repli n'est pas immédiatement appliqué. Comme la valeur de niveau reste entre les bornes définies pour la commande Tout-ou-Rien, aucun changement n'est demandé à l'actionneur. La première demande lui arrive vers 8400 *Te*. Ce n'est qu'à cet instant que on considère que on a quitté la phase du bon fonctionnement du capteur analogique. Il faut prendre en compte que cette période de non-activité de l'actionneur peut d'une manière significative changer les résultats numériques présentés dans la suite.

En premier lieu, la phase stationnaire sous la condition de bon fonctionnement du capteur analogique est analysée. Cet intervalle est borné à gauche par l'entrée du système dans la phase stationnaire (typiquement le dépassement du niveau de 400 *hu*) et à droite par l'instant de l'application de l'algorithme de repli. Sur la figure 5.9, cette phase s'étale entre les deux premiers traits verticaux. Les écarts entre la consigne et le niveau réel sont petits et par conséquent les variations du taux de pompage

sont également petites. La date moyenne d'entrée dans cette zone est 11.29 *Te* et la durée moyenne de séjour dans cette phase est de 2297 *Te*. La figure 5.10 montre les distributions de ces deux caractéristiques.

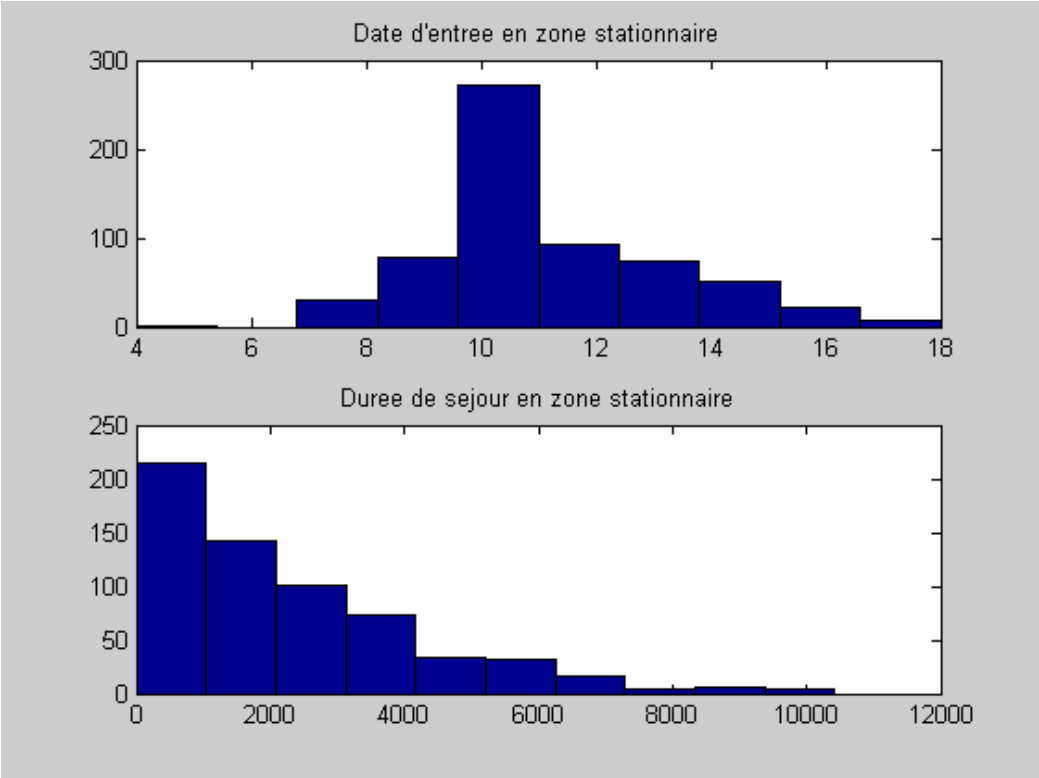


Figure 5.10 : Caractéristiques de la zone stationnaire

En deuxième partie, on considère l'état stationnaire en présence de panne du capteur analogique. La date moyenne d'entrée dans cette zone est 2309 *Te*. La distribution correspondante est précisée sur la figure 5.11.

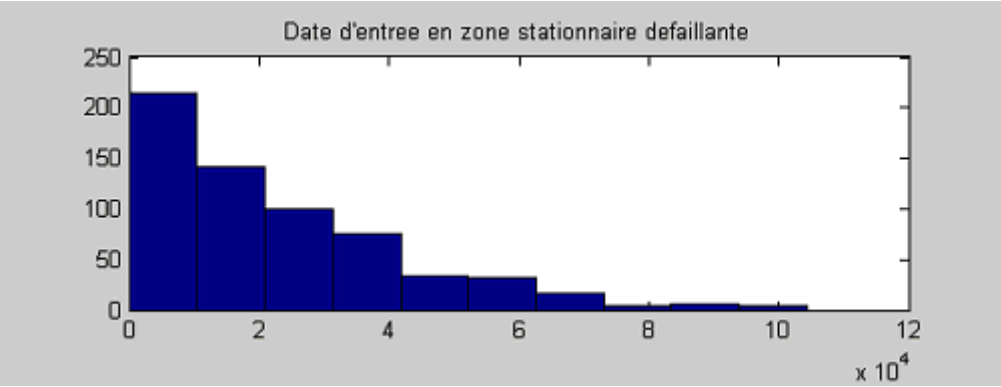


Figure 5.11 : Caractéristiques de la zone stationnaire avec une défaillance

Une fois les durées et dates d'entrées établies, on peut alors identifier les taux d'usure dans les phases respectives. Le taux d'usure est défini comme $\tau m = sum(ui+1-ui)/durée_de_sejour$. Pour la phase stationnaire avec bon fonctionnement du capteur analogique cette usure moyenne est 5.0 *hu/Te*. Sa distribution est représentée sur la figure 5.12a. Les quelques valeurs "parasites" qui représentent le taux d'usure compris entre 8 et 19 *hu/Te* sont des cas pour lesquels le niveau venait d'atteindre la phase stationnaire et immédiatement une défaillance s'est produite et l'algorithme de repli a été utilisé. Ces cas sont peu nombreux et peuvent être négligés. Les valeurs significatives se trouvent alors entre 0 et 8 *hu/Te*. La taille de cet intervalle est entre autre une fonction du repos de l'actionneur, où celui-ci n'est pas sollicité.

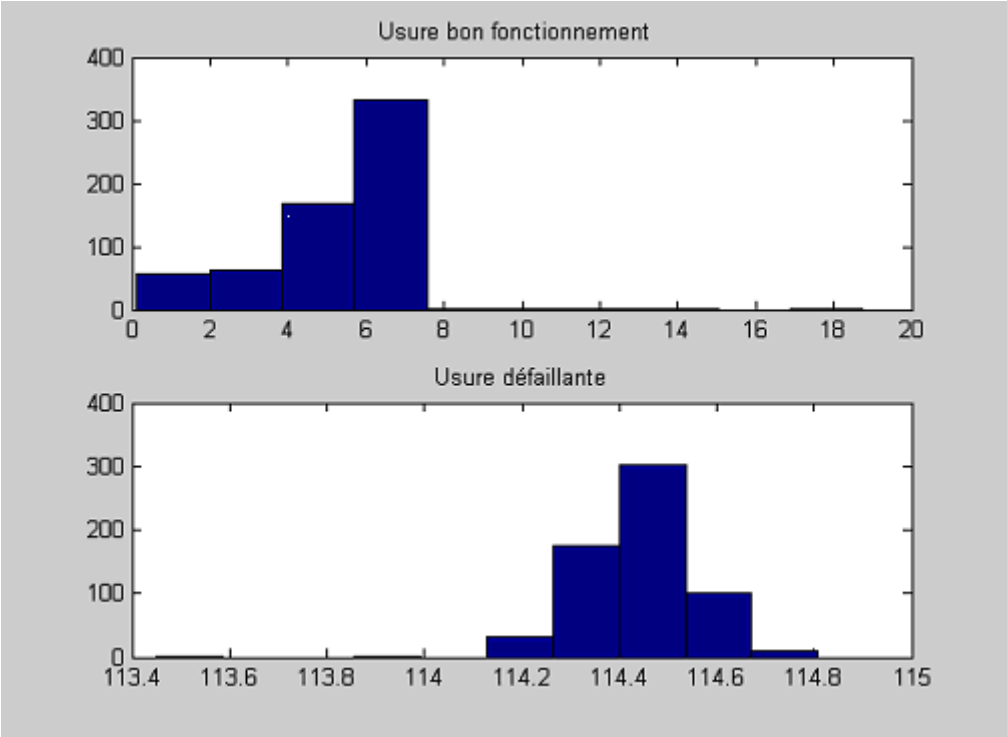


Figure 5.12 : Zone stationnaire avec une défaillance :

a.) Usure sous fonctionnement nominal,

b.) Usure en présence d'une défaillance.

Ceci n'est pas le cas de l'usure sous l'algorithme Tout-ou-Rien. La distribution de l'usure dans ce cas (figure 5.12b) est beaucoup plus étroite que dans le cas précédent. La valeur moyenne est de 114.44 *hu/Te*.

5.5 CONCLUSION

L'intérêt de cette étude porte sur l'estimation du taux de l'usure en fonction de la configuration dynamique du système. Elle permet d'obtenir une comparaison quantitative des comportements dynamiques des composants. Ainsi elle peut être utilisée pour comparer les fonctions critères des différentes architectures et topologies et les classer selon leurs valeurs quantitatives.

Un deuxième résultat intéressant est l'estimation de durée de séjour dans des configurations individuelles. Ainsi en connaissant la façon à laquelle les composants seront utilisés ainsi que leur durées, il est possible par exemple d'évaluer les différentes architectures sur les critères du coût de

maintenance et évaluer son apport à l'amélioration des performances du système. Les résultats sont synthétisés dans le tableau suivant :

Le MTTF pour les donnés non censurées est de	0.494 heures (1777 *Te*)
Date d'entrée dans la zone stationnaire moyenne	11.29 *Te*
Durée de séjour stationnaire moyenne	2297 *Te*
Date d'entrée dans la zone stationnaire défaillante moyenne	23089 *Te*
Usure moyenne bon fonctionnement	5.0 *hu/Te*
Usure moyenne défaillante	114.4 en *hu/Te*

D'un point de vue plus général, cette étude nous a permis d'identifier de nombreuses questions qui ont besoin d'être prises en compte pendant la conception du modèle d'un SAR. Il est question notamment de la coexistence des parties échantillonnées avec des systèmes à événements discrets, du choix du régulateur, des modes de défaillances des composants.

Elle a également permis de voir les possibilités et les limites d'analyse par simulation de Monte Carlo et d'en déduire le besoin d'utiliser les taux de défaillance très élevés pour pouvoir obtenir des résultats d'un laps de temps/calcul donné.

6 Etude des scénarios défaillants

6.1 INTRODUCTION

Dans cette étude ([Bar 03], [Bar 03a], [Thi 03]) on parlera des scénarios d'événements qui précèdent la défaillance globale du système d'automatisation. Le problème de l'influence du réseau de communication est également largement présenté et une solution possible est discutée. Les caractéristiques de la SdF de ces composants sont très variables et leurs défaillances sont indépendantes entre elles. Le taux de défaillance pour chaque composant est supposé connu a priori, mais les relations avec les autres composants du système restent à évaluer à posteriori. Une estimation d'un taux de défaillance globale du SAR complet basée sur la connaissance des taux de défaillances individuels est le but de cette étude. Une contribution à l'étude des dépendances dynamiques est également présentée. L'approche toute entière est illustrée par un exemple facilitant la compréhension de la méthode appliquée.

Cette étude s'inspire de l'exemple du chapitre précédent. Des changements ont été effectués parmi lesquels l'arrêt de l'exploitation de deux capteurs binaires (même s'ils restent présents dans le modèle leurs informations ne sont plus traitées par le régulateur), le changement de régulateur pour un prédicteur de Smith, une nouvelle fonction de transfert de la cuve ou encore l'ajout d'une deuxième boucle de régulation (« Process2 » et « Controller2 ») fonctionnant d'une manière indépendante de la première boucle, mais partageant le réseau de communication et ainsi générant du trafic supplémentaire sur le réseau. Les modes de défaillance des composants ont été élargis et retravaillés.

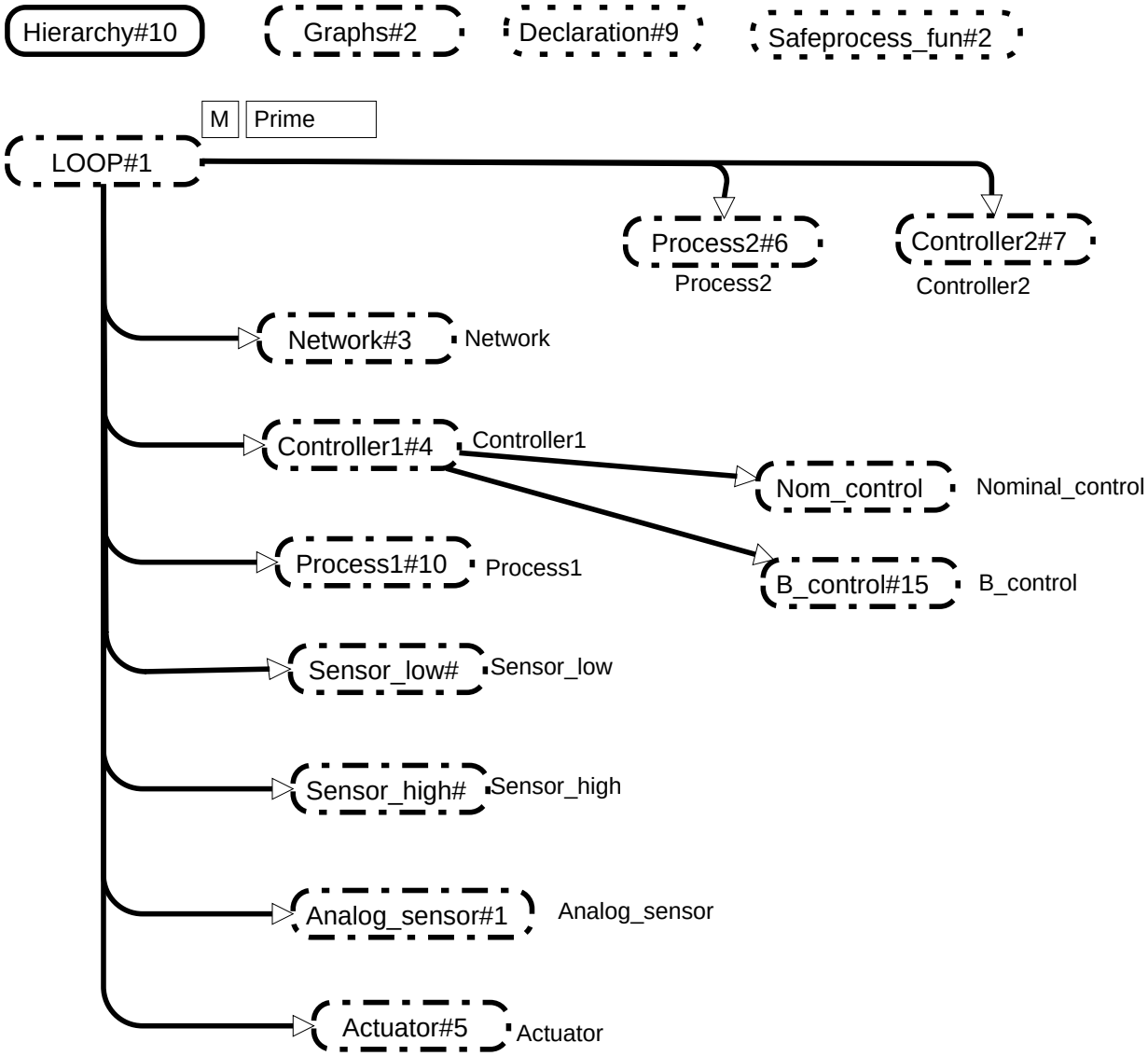


Figure 6.1 : Hiérarchie des modèles

La figure 6.1 résume la hiérarchie du modèle. Il comporte la page des graphes. Une page entière est réservée au nœud de déclaration et une autre aux fonctions d'analyse. La page « Loop » ressemble à celle du chapitre précédent et contient en plus une deuxième boucle de régulation (« Process2 » et « Controller2 »). Dans le régulateur « Controller1 » il reste une sous-page « B_control » pour représenter l'algorithme de commande Tout-ou-Rien, mais cette page n'est pas activée pendant les simulations. La suite propose la description des composants.

6.2 PRESENTATION DES COMPOSANTS

6.2.1 Capteur

Cette étude porte sur un capteur qui envoie ses messages périodiquement. Une hypothèse de mesures non biaisées est faite mais n'affecte pas la portée de la méthode exposée. A chaque période d'échantillonnage *Te* une nouvelle mesure est établie et préparée pour l'envoi. Il est alors possible de différencier deux fonctionnalités du capteur : 1.) Mesures (transition « Measuring ») et 2.) Communication (transition « Sending »).

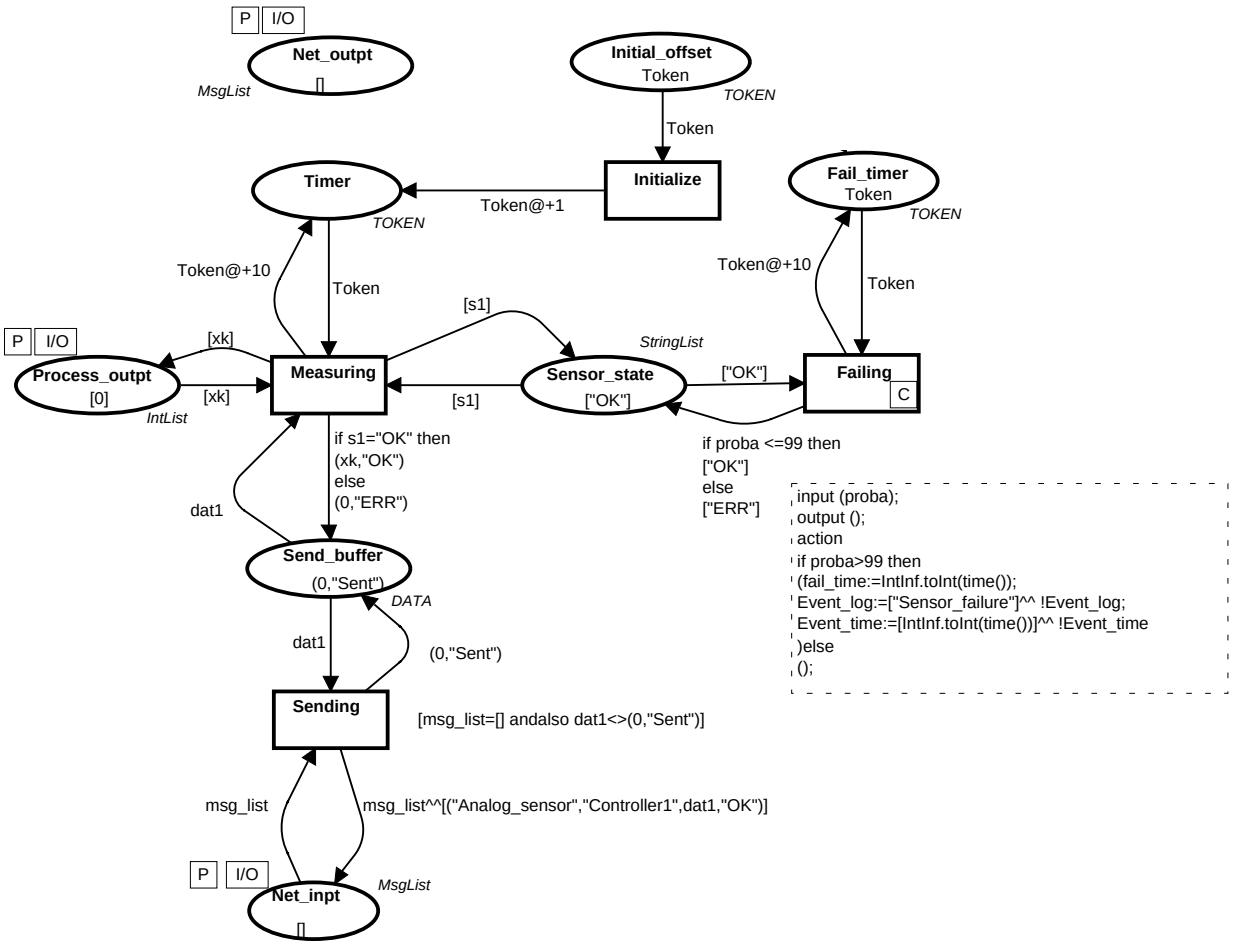


Figure 6.2 : Modèle du capteur analogique

La figure 6.2 montre le modèle du capteur dans son état initial. On peut y voir que la place « Timer » est vide. Elle est initialisée à partir de la place « Initial_offset » qui contient un jeton. L'arrivée du jeton vers la place « Timer » est retardée d'une unité de temps (inscription « Token@+1 »). Ce retard permet

de déphaser la mesure (franchissement de « Measuring ») par rapport à l'instant d'évolution de processus (mise à jour du contenu de la place « Process_outpt »). Ce décalage garantit que le capteur va toujours mesurer l'état du processus après son instant de modification et pas avant. Sans ce décalage les mesures prises à l'instant *k*Te* pouvaient être égales à l'état du processus à l'instant *k*Te* juste avant son évolution ou immédiatement après. C'est pourquoi nous avons volontairement déphasé les instants de mesure du capteur.

La défaillance ne modifie pas la capacité de communication du capteur. Elle arrête exclusivement la prise des mesures. Par conséquent le capteur n'envoie pas la valeur mesurée mais seulement une trame indiquant l'apparition de défaillance (avec « 0,'ERR' » dans les champs de données). La probabilité de cette défaillance est fixée constante et égale à 1/100 *Te* avec une distribution uniforme. Dans la suite, cette défaillance est appelée *Def_capt*.

6.2.2 Régulateur

La synthèse du régulateur pour les systèmes à retard variable est un problème hors de portée de la présente étude. Une proposition pour une telle synthèse peut être trouvée dans [Wal 01]. Le régulateur utilisé est un prédicteur de Smith (figure 6.3) avec à la base un algorithme PI. Le délai pur du régulateur de Smith est supposé égal à *2*Te*, ce qui correspond au délai d'une période d'échantillonnage pour le passage de chaque information du capteur au régulateur et de l'autre période entre le régulateur et l'actionneur. Dans la totalité de cette étude, le réseau de communication garantit les délais de transmission inférieurs à la période d'échantillonnage.

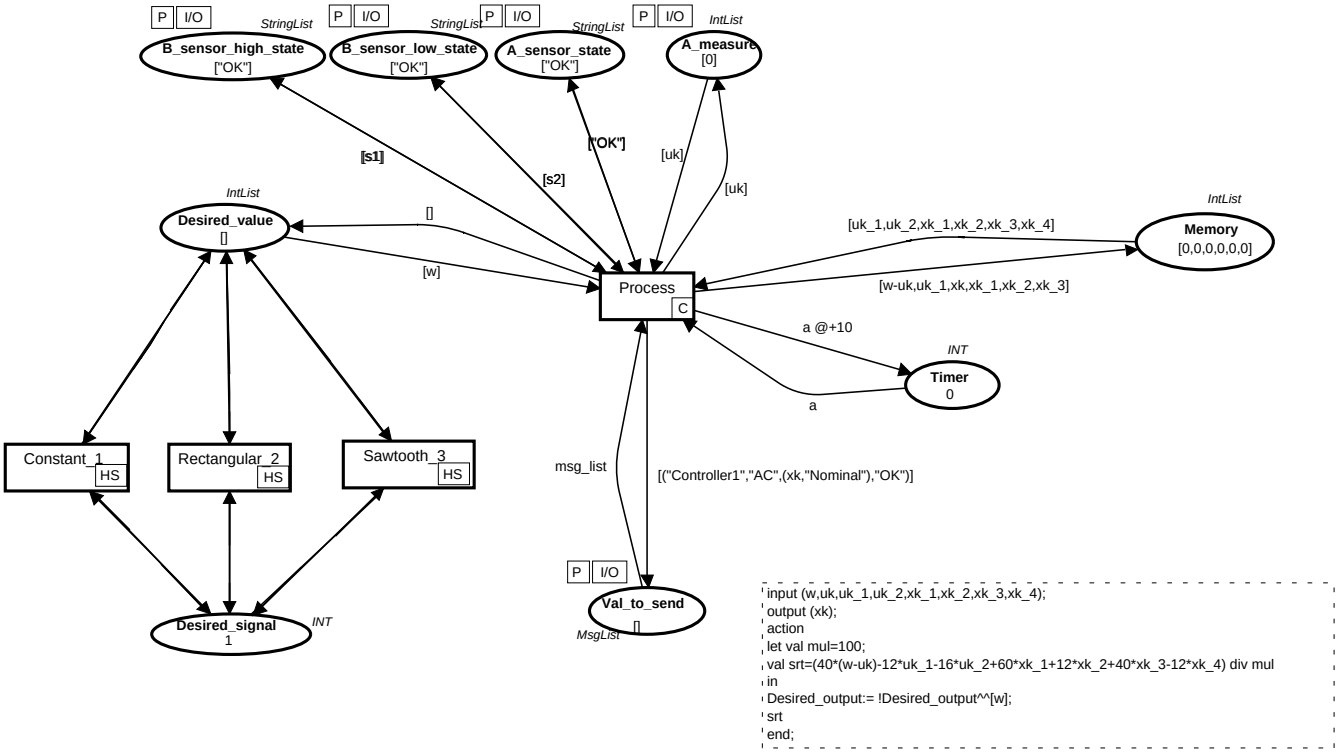


Figure 6.3 : Modèle du régulateur

Pendant la défaillance du régulateur, la fonctionnalité de communication est arrêtée. Cette fonctionnalité se trouve sur le niveau supérieur dans la hiérarchie du modèle par rapport à la partie présentée et donc ne se trouve pas sur la figure 6.3. Ainsi, aucun message n'est envoyé à l'actionneur. Comme dans le cas du capteur, le taux de défaillance est constant et placé à 1/100 *Te*. Cette défaillance est nommée *Def_reg* dans la suite.

Si le régulateur ne reçoit pas une mesure valable du capteur, soit parce que le capteur est en panne, soit parce que l'information a été altérée ou perdue pendant la transmission, il continue à établir les nouvelles commandes en fonction de la dernière mesure obtenue.

6.2.3 Actionneur

Un système discret du 1er ordre modélise l'actionneur qui représente la pompe du système d'origine. L'actionneur peut se trouver dans un des deux états possibles : *action* et *attente*. L'état *attente* est déclaré si l'entrée de l'actionneur (la commande du régulateur) et sa sortie (taux de pompage) sont égales. Dans tous les autres cas, il est déclaré en *action*.

Deux modes de défaillance sont considérés sur l'actionneur : élément *collé* et *usure*. Les probabilités des deux modes sont constantes avec des distributions uniformes. Le premier est dû au blocage de la pompe au (re)démarrage après que la pompe a été arrêtée. Le deuxième représente un problème typique de l'usure d'un composant pendant son fonctionnement.

Le problème de collage, qu'on appellera *Act_bloc*, est une défaillance qui n'est pas une fonction du temps mais plutôt du nombre de sollicitations. Cette défaillance apparaît en moyenne 1 fois sur 100 requêtes de démarrage et par conséquent sa fréquence d'occurrence dépend aussi de l'algorithme de commande choisi. Il est possible de supposer que la régulation Tout-ou-Rien produirait un nombre plus important des occurrences de cette défaillance que la régulation PI auquel cas le démarrage reste un événement peu fréquent.

L'*usure* (ou *Act_usu*) ne peut apparaître que si l'actionneur est en *action*. Une loi uniforme est utilisée pour représenter cette défaillance est son taux est fixé à 1/50 *Te*.

La coexistence de ces deux types de défaillances pose un certain nombre de problèmes dans les méthodes de modélisation et analyse classiques, mais peut être résolue aisément dans un modèle RdP.

6.2.4 Réseau de communication

Le réseau de communication est le seul composant dans le système modélisé qui fonctionne d'une manière événementielle. Le délai de transmission d'un paquet est considéré constant est égal à 1/10 *Te*. Dans le fonctionnement nominal pendant chaque macrocycle deux trames doivent être communiquées. Une du capteur vers le régulateur et l'autre du régulateur vers l'actionneur. En cas de besoin, le présent modèle peut être modifié pour représenter un délai de transmission aléatoire ou avec une gigue. Le mécanisme d'accès au média n'a pas beaucoup d'importance dans ce cas, puisque la bande passante du réseau est largement supérieure à la bande passante nécessaire.

L'intérêt de ce réseau de communication ne réside donc pas dans l'étude d'influences des retards, mais plutôt dans les erreurs qui apparaissent pendant la transmission est leur influence sur le comportement global du système. Pour cette raison deux erreurs possibles sont intégrées dans le modèle : 1.) *Pert_tramme* si la trame est perdue et 2.) *Alt_trame* si elle est altérée.

Dans l'état actuel, ces deux erreurs auront les mêmes conséquences sur les composants mais peuvent en avoir différentes dans les travaux futurs. Leurs probabilités d'occurrences sont identiques et égales à une perte de paquet sur 20 paquets envoyés et modification d'un paquet sur 20.

6.3 DEPENDANCES DES DEFAILLANCES

Une fois que les composants et leurs caractéristiques ont été définis, l'accent peut être mis sur la défaillance du système global et ensuite sur les relations entre les défaillances des composants individuels et la défaillance globale. Rappelons que la défaillance d'une entité est définie comme son incapacité d'accomplir la tâche demandée sous les conditions imposées. Dans le cas présent, la mission du système global est de remplir la cuve et de maintenir le niveau du liquide dans l'intervalle désiré. Ici, la valeur désirée est placée à 500 unités de hauteur (*hu*). De l'énoncé de la mission du système deux types de comportement peuvent être déclarés défaillants :

1.) le remplissage de la cuve échoue et à la fin de l'intervalle d'observation le niveau ne s'est jamais approché de la valeur désirée et
2.) le maintien échoue et le niveau quitte l'intervalle cible après s'y être approché.

Le premier mode est nommé *idle* et le deuxième *failed*. Pour le mode *failed*, l'instant d'occurrence est défini comme la coordonnée temporelle de la sortie définitive du niveau de ± 10% intervalle autour de la valeur désirée.

Dans les paragraphes suivants, est abordé le problème des relations entre les défaillances des composants individuels et celle du système global.

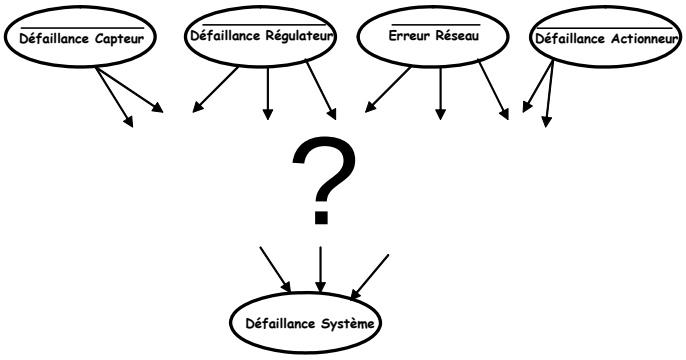


Figure 6.4 : Dépendances des défaillances

A l'instant initial, tous les composants sont dans leurs états nominaux, c'est-à-dire qu'aucune défaillance n'est présente (fig. 6.4). Les états finaux du système à la fin de chaque intervalle d'observation dans notre étude peuvent être soit l'état de bon fonctionnement, soit *idle* ou *failed*. Les deux derniers sont des états dysfonctionnels. Ces états peuvent être atteints de différentes façons à partir de l'état initial du système. L'organisation des dépendances n'est pas linéaire mais plutôt parallèle. La structure logique d'un SAR peut être vue comme un enchaînement série du capteur, du réseau de communication, du régulateur, encore du réseau et de l'actionneur qui sont placés dans une boucle fermée. Néanmoins, la défaillance du capteur n'entraîne pas forcément la défaillance ni du réseau de communication ni de celle du régulateur. Mais même si le régulateur fonctionne correctement, si le capteur est hors service et donc les entrées du régulateur sont obsolètes, le fonctionnement du SAR se dégrade et il tend alors vers un état global de panne. De plus, la défaillance d'un composant peut entraîner des changements dans la façon d'exploiter les autres composants et par conséquent leurs caractéristiques seront modifiées. La complexité de ce problème de dépendances entre les défaillances est encore accentuée par la présence des erreurs de communications qui sont difficilement intégrables dans un schéma et leur influence reste difficile à prévoir.

Pour proposer une solution partielle à ce problème, nous proposons dans la suite une étude basée sur des scénarios (séquences d'événements) qui mènent vers une défaillance du SAR. Les scénarios peuvent être obtenus d'un modèle CPN de deux manières différentes : 1.) simulation et 2.) graphe d'occurrence. Les deux méthodes sont exigeantes en temps et mémoire de calcul. L'avantage du graphe d'occurrence est la construction d'un ensemble complet et donc exhaustif des scénarios possibles, mais l'inconvénient de cette méthode est son explosion combinatoire importante, notamment sur des systèmes stochastiques, qui rend son utilisation difficile, voir impossible. Pour cette raison, nous avons préféré l'utilisation d'une simulation de Monte Carlo [Pag 80, Mon 98].

Les taux de défaillance sont d'une importance cruciale pour toute étude de la sûreté de fonctionnement. Dans cette étude, ils sont choisis arbitrairement très élevés pour accélérer la simulation. Ceci n'est pas très gênant ici dans la mesure où nous parvenons malgré tout à visualiser des états dysfonctionnels qui peuvent donc se produire, et à en déduire les conséquences. Une méthode pour travailler avec des taux plus proches de la réalité est proposée et appliquée dans [Mon 98] et la méthode s'appelle simulation par sauts (voir chapitre 2.7). Son idée de base est de remplacer un intervalle de simulation par l'estimation du comportement du modèle. Mais, comme seules les relations dynamiques logiques entre les défaillances sont étudiées ici et non leurs apparitions dans le temps, les taux choisis peuvent être considérés comme acceptables.

6.4 SIMULATION DE MONTE CARLO

La simulation de Monte Carlo [Lab 02] est effectuée sur le modèle du système étudié. Elle consiste à répéter la simulation de fonctionnement un certain nombre de fois. Pour des raisons évoquées auparavant (chapitre 2.6) ce nombre doit être suffisamment élevé. Pour les besoins de cette étude 6734 simulations de fonctionnement ont été effectuées. Chaque simulation s'intéresse à l'intervalle du temps entre 0 et 2000 unités de temps (*tu*) avec la période d'échantillonnage des composants *Te* = 10 unités de temps. Chaque simulation démarre avec une cuve vide. L'algorithme de contrôle stabilise le niveau du liquide à 210 unités de temps (*tu*) si les perturbations supplémentaires n'étaient pas considérées. Les perturbations intégrées dans le modèle représentent une petite fuite aléatoire et leur valeur est comprise entre 0 et 10 unités de hauteur (*hu*) à chaque période d'échantillonnage *Te*.

Quelques exemples de simulation de fonctionnement sont relevés sur la fig. 6.5. Le temps sur l'axe *x* est représenté en *Te*. L'intervalle cible du niveau se situe entre 450 et 550 unités de hauteur (deux traits horizontaux). En principe, chaque réponse indicielle est composée de deux parties principales : transitoire et stationnaire. La première phase dure habituellement à peu près 20 *Te* et par conséquent la phase stationnaire dure 180 *Te*. Dans la suite, il sera montré que les relations entre les défaillances individuelles sont différentes selon la phase dans laquelle se trouve le SAR. Il s'agit donc bien de dépendances dynamiques.

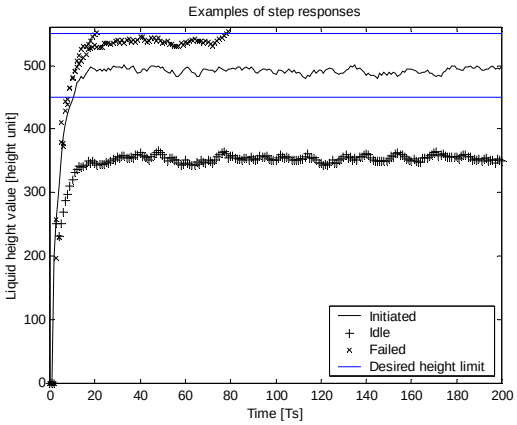


Figure 6.5 : Quatre exemples de simulation de fonctionnement

A la fin de chaque simulation de fonctionnement, le SAR peut se retrouver dans un de trois états : *idle*, *initiated* ou *failed*. L'état *idle* signifie que le niveau du liquide n'est jamais pendant la simulation entré dans l'intervalle désiré. Il s'agit donc d'une défaillance de la fonctionnalité de remplissage. C'est un état dysfonctionnel. Cet état peut apparaître si par exemple l'actionneur est bloqué à la première demande de démarrage. Sur l'ensemble de 6734 simulations, le système a fini dans cet état 389 fois (~6%).

L'état *initiated* représente que le SAR à accompli sa double mission (remplissage et maintien) avec succès pendant toute la durée de l'intervalle d'observation et une fois entré dans le voisinage autour de la valeur désirée, il ne l'a jamais quitté. Il faut souligner qu'aucune supposition sur les états fonctionnels des composants individuels n'est faite dans ce cas. A la fin de la simulation, les composants peuvent être aussi bien en fonctionnement nominal qu'en panne. 5620 (~83%) simulations observées ont fini dans l'état *initiated*.

Failed arrive si le niveau quitte le voisinage de la valeur désirée après y avoir entré. Aucune supposition sur la durée de séjour dans le voisinage n'est faite. Cette situation s'est produite 725 fois (~11%) sur l'ensemble des simulations. Le tab. 6.1 présente une synthèse des états finaux de la simulation.

Total de simulations	Simulation finie comme		
	Idle	*Initiated*	*Failed*
6734	389	5620	725
100%	~6%	~83%	~11%

Tab. 6.1 : Résultat de la simulation de Monte Carlo

Pour pouvoir déterminer le « Mean Time To Failure » (MTTF) du système global, les deux états dysfonctionnels (*idle* et *failed*) devraient être pris en compte. L'instant d'apparition de l'état *failed* est défini d'une manière exacte mais pour l'état *idle* il existe une diversité de définitions possibles quant à son temps d'occurrence. Cela peut être soit une valeur arbitraire (dont *0*) soit le résultat d'un algorithme quelconque et sa valeur est entièrement dépendante du choix du concepteur. Cette diversité est la raison pour laquelle aucune valeur du MTTF n'est donnée ici.

La probabilité d'aboutir à la panne du système avant la fin de l'intervalle d'observation peut quant à elle être calculée facilement comme :

$$\frac{389 + 725}{6734} = 16,54\%$$

6.5 SCENARIOS

Une fois l'information sur la probabilité de défaillance moyenne obtenue, on peut procéder à l'étude des dépendances entre les défaillances. Il est évident que la défaillance globale du système est une conséquence d'une suite d'événements afférents. Ces séquences sont appelées des scénarios et sont présentées dans [Mon 98].

Dans le modèle de la présente étude, 6 événements ont un impact sur la défaillance globale (table 6.2). Ils sont enregistrés tout au long de la simulation de Monte Carlo ensemble avec les instants de leurs apparitions.

Evénement	Son Code	Taux d'occurrence
Def_capt	6	1/100 T_e
Blocage_actionneur	5	1/100 requêtes
Usure_actionneur	4	1/50 T_e
Def_contrelleur	3	1/100 T_e
Pert_trame	2	1/20 trames
Alt_trame	1	1/20 trames

Tab. 6.2 : Evénements significatifs

Les différents scénarios de ces événements sont ensuite analysés selon l'algorithme suivant : pour chaque simulation qui s'est terminée dans l'état *failed*, prendre toujours un scénario de N événements (N : longueur de scénario; N=1, N=2, …, N=25) qui précède la défaillance globale et compter le nombre de ses apparitions dans l'ensemble des événements enregistrés ainsi que le nombre de fois où ce scénario est directement suivi par la défaillance globale. Pour obtenir la probabilité que le scénario (de longueur N) concerné mène à la défaillance il suffit de faire un rapport de ces deux nombres.

Le but de cette opération est d'examiner le potentiel de cette méthode de séparer les événements 'importants' de ceux qui ne le sont pas. L'inclusion d'un événement important dans le scénario devrait se traduire par l'augmentation de la probabilité d'une défaillance globale.

Passons maintenant à l'analyse d'un scénario concret. La séquence considérée est <1 1 2 1 2 1 2 2 2 1 1 2 6 2 2 3 2 1 1 2 4 1 2 2 1>, qui sont les codes correspondants aux différents événements et leur ordonnancement est chronologiquement inverse. Ce scénario indique que le dernier événement enregistré avant la défaillance globale est *Alt_trame* (code 1), l'avant dernier *Alt_trame* et l'antépénultième *Pert_trame* (code 2). Le treizième événement avant la défaillance globale est *Def_capt* (code 6).

La modification de trame (*Alt_trame*) est un prédécesseur direct de 342 défaillances globales sur les 725 *failures* observées pendant la simulation. On pourrait être tenté de dire, que *Alt_trame* cause presque une défaillance sur deux. Mais ce n'est pas une bonne conclusion. On ne peut que constater que presque 50% des défaillances sont directement précédées par une *Alt_trame*.

La même *Alt_trame* est apparue 319043 fois dans les 6734 simulations. De ce point de vue, il est clair que seul un pourcentage négligeable des *Alt_trame* précède la défaillance globale. On pourrait être tenté de prononcer qu'une *Alt_trame* ne mène presque jamais à la défaillance globale. Nous verrons dans la suite que cela serait une conclusion prématurée.

Si on considère maintenant un couple d'événements précédant la défaillance globale, ce qui dans le scénario étudié est la paire *Alt_trame*, *Alt_trame*, les résultats obtenus sont les suivants : cette paire précède directement 150 sur 725 défaillances observées et il est apparu en tout 154600 fois dans l'ensemble des simulations de fonctionnement.

La table 6.3 récapitule l'ensemble des résultats pour le scénario étudié en fonction de sa longueur N ($1 \leq N \leq 25$). La probabilité que la suite directe de séquence de N événements soit la défaillance globale peut être obtenue en divisant la troisième ligne par la quatrième. Le résultat de cette opération (ligne 5) est relevé sur la figure 6.6. L'axe horizontal montre le nombre des événements pris en compte pour le calcul de la probabilité. Sur l'axe *y* se trouve la valeur de la probabilité calculée.

Longueur du scénario	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Evenement concerné	1	1	2	1	2	1	2	2	2	1	1	2	6	2	2	3	2	1	1	2	4	1	2	2	1
Nb pannes directes	342	150	77	27	11	4	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Nb total	319043	154600	74911	36220	17506	8522	4144	1991	974	455	212	212	2	1	1	1	1	1	1	1	1	1	1	1	1
%	0.11	0.1	0.1	0.07	0.06	0.05	0.05	0.05	0.1	0.22	0.47	0.47	50	100	100	100	100	100	100	100	100	100	100	100	100

Tab. 6.3 : Nombre d'occurrences de parties de scénario

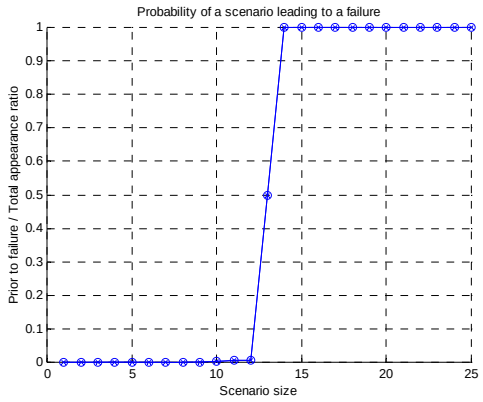


Figure 6.6 : Probabilité d'occurrence d'une défaillance globale en fonction de la longueur de scénario

Plusieurs informations intéressantes peuvent être déduites de ce graphe. Premièrement, il s'agit des conséquences des pertes et des modifications des trames de communication. Dans le scénario étudié, la séquence de douze événements *Alt_trame* et *Pert_trame* est présente et ne conduit que rarement à la défaillance globale. Ici encore il faut éviter de faire des conclusions prématurées comme par exemple de dire que les pertes et modifications des paquets sont des événements insignifiants, même si dans notre étude une séquence de douze de ces événements n'a mené à la défaillance qu'une fois sur 212 apparitions au total (~0.5%).

La situation change rapidement si on rajoute le treizième événement au scénario. Cette nouvelle séquence a apparue 2 fois pendant la simulation et dans un cas elle a abouti directement au débordement de la cuve. La deuxième fois, la simulation s'est terminée dans l'état *idle* à cause d'une défaillance ultérieure. Il est donc évident que sur ce scénario la défaillance du capteur augmente considérablement la probabilité de l'apparition du débordement de la cuve.

Le scénario présenté est un choix arbitraire dans l'ensemble des scénarios observés pendant la simulation. Les longueurs des scénarios relevés ont varié entre 3 et 50 événements pour chaque simulation. Le scénario étudié devient unique si plus que 13 événements sont considérés. Dans le cas d'une simulation de durée plus grande, il est probable que l'unicité de sa présence disparaîtrait. Mais son unicité ne limite en aucun cas la portée de cette étude. Le problème serait l'unicité des événements dans le scénario, mais tel n'est pas le cas.

Une situation semblable peut être également observée pour les autres scénarios, notamment pour les défaillances de l'actionneur (collage et usure) et du régulateur. La présence d'un de ces événements entraîne souvent mais pas toujours immédiatement la défaillance du SAR. L'intervalle de temps qui s'écoule entre la défaillance d'un composant et le débordement effectif de la cuve peut être très variable. Dans la continuation de cette étude, on pourrait montrer que la présence d'un de ces 4 événements mène à un des deux états défaillants avec une probabilité proche de 1.

6.6 DEPENDANCES DYNAMIQUES

Des propos dits auparavant, on pourrait conclure que les modifications ou pertes des trames ne sont pas des événements primordiaux au bon fonctionnement du SAR. Dans le scénario présenté, il y a une

séquence de 12 tels événements qui n'aboutit pas à une probabilité de défaillance globale élevée. Mais sur 725 *failed* scénarios 359 (~50%) ne contiennent que les deux événements liés au réseau de communication. Il en résulte que presqu'un débordement sur deux est une conséquence des erreurs survenues au niveau du réseau de communication. Ceci est une conséquence directe de la définition de la défaillance globale dans le SAR. S'il se produit une rafale d'erreurs de communication pendant la phase transitoire de la réponse indicielle (remplissage), le régulateur calculera des valeurs qui mènent directement et très rapidement à un débordement du niveau de liquide.

Les taux d'erreur choisis pour le réseau de communication font que le réseau est le composant le plus défaillant parmi l'ensemble des constituants du SAR. Mais il faut prendre en compte le fait que les erreurs de communication sont limitées dans le temps et pas permanentes comme c'est le cas avec d'autres composants. La proportion des défaillances dues au réseau reste quand même très importante. Il peut être supposé que les changements dans les taux de probabilité de défaillance des constituants va d'une manière significative changer les résultats présentés ici.

6.7 RESULTATS DE LA METHODE, CONCLUSIONS

La méthode proposée ici permet d'obtenir des résultats intéressants dont les plus importants sont synthétisés dans la suite.

Premièrement, l'approche décrite permet l'estimation du taux de défaillance global pour un système complexe sous les conditions que les modes de fonctionnement et les taux de défaillance des composants soient connus a priori. Ceci peut être une contribution à la phase de conception des systèmes distribués complexes où les composants sont bien connus mais leur interactions et dépendances reste difficiles à estimer.

En deuxième lieu, l'étude confirme le résultat intuitif que la défaillance du capteur, du régulateur et de l'actionneur mène presque inévitablement à la défaillance globale et la méthode permet l'évaluation quantitative de ce comportement, et donc, des situations critiques. Par conséquent, ces défaillances doivent être constamment considérées comme importantes et ainsi surveillées en permanence.

Troisièmement, bien que les modifications et les pertes de trames soient les événements les plus fréquents, ils ne causent pas forcément toujours la défaillance globale. Leur degré d'importance varie en fonction de l'état/phase du système. Dans la phase stationnaire, leur importance est moindre, mais pendant le régime transitoire celle-ci accroît. En effet, un des pires cas recensés est la présence de quatre pertes ou modifications relativement proches dans le temps, mais pas sur des trames consécutives tout au début de la simulation, ce qui entraîne directement un débordement.

Le quatrième point souligne l'importance de l'étude intégrant le réseau de communication. Dans l'exemple présenté le réseau lui-même est la cause unique de 50% des défaillances globales. Ceci montre bien l'importance de l'étude de la sûreté de fonctionnement pour ce type de cas.

En dernier lieu, la méthode proposée permet non seulement une classification statique des événements à prendre en compte, mais également leur association avec l'état/phase du système global. Les fonctions d'importance dynamiques des différents événements peuvent être établies. Cette information trouve son application entre autre dans l'allocation dynamique des ressources de communication dans les systèmes complexes basée sur la connaissance de l'état/phase de chaque sous-système.

Actuellement, seules les relations logiques ont été analysées. Il ne faut pas oublier que l'aspect temporel est crucial dans ce domaine et son étude est une perspective majeure avec le but d'estimer l'importance temporelle (ex. : facteur d’oubli) et dynamique des différents événements.

7 Modélisation d'un réseau de sécurité réel

7.1 INTRODUCTION

L'utilisation des réseaux de communication a de nombreux avantages. Néanmoins leur implémentation industrielle dans des installations à forte criticité ou dans des installations de sécurité est limitée par l'approbation par des organismes indépendants. L'Institut National de Recherche et de Sécurité (INRS) fait partie de ces organismes, particulièrement dans le domaine des machines. Nous avons eu l'occasion de coopérer avec l'INRS afin de vérifier les spécifications techniques du réseau de sécurité AS-i Safety at Work, en regard des contraintes de sécurité attendues.

Dans cette étude nous présentons tout d'abord les concepts de base du protocole AS-i et son extension AS-i Safety at Work. Une partie du protocole est ensuite modélisé par l'approche des RdP colorés et le modèle est analysé grâce au graphe d'occurrence construit à partir du modèle [Bar 03b]. Cette étude est différente des précédentes en plusieurs points :

- le but n'est pas d'estimer les caractéristiques par simulation de Monte Carlo, car cette méthode n'est pas adaptée pour l'étude de la sécurité,
- les modèles construits doivent correspondre intégralement à la spécification de la norme, avec une possibilité de vérification de cette correspondance à tout moment,
- il est impératif que le graphe d'occurrence puisse être analysé. En définissant les critères d'évaluation il faut donc limiter l'explosion combinatoire. Dans notre cas, c'est fait grâce à une séquence d'excitation déterministe.

En résultat de ce travail on récupère :

- les modèles des différents composants du réseau AS-i,
- la liste des ambiguïtés et incohérences de la norme et par conséquent la liste des choix que nous avons fait pour aboutir à des modèles finaux,
- le graphe d'occurrence prêt pour une analyse. Dans notre cas, la séquence d'excitation de l'esclave est périodique et donc infinie. C'est pourquoi seulement une partie du graphe est construite.

Il est à noter que modéliser une norme avec des Réseaux de Petri n'est pas une idée novatrice. Elle est largement abordée entre autre dans [Bla 00]. L'avantage de notre approche est de modéliser avec un outil logiciel existant et ensuite d'intégrer le modèle dans le reste du système d'automatisation.

7.2 AS-I ET AS-I SAFETY AT WORK

AS-i, une abbréviation pour Actuator – Sensor interface, appartient à la catégorie des réseaux de terrain. Déjà de son appellation il est clair que le domaine de son application principale est l'interconnexion des capteurs et des actionneurs aux régulateurs. La topologie du réseau est soit un bus, soit un arbre, soit une étoile et elle doit contenir un maître unique et un nombre d'esclaves (62 pour AS-i et 31 pour AS-i Safety at Work maximum). Tous les échanges d'informations sont initiés par le maître auquel répondent potentiellement les esclaves. La réponse de l'esclave contient une information codée sur 4 bits.

Une couche supplémentaire est ajoutée au protocole AS-i pour former AS-i Safety at Work. Ainsi le protocole AS-i devient une partie centrale dans un réseau de communication dédié à la sécurité. Selon sa présentation par le consortium, toutes les fonctions de sécurité peuvent y être implémentées puisque AS-i Safety at Work est certifié pour être conforme à la catégorie 4 de la EN (Norme Européenne) 954 – 1. L'un des avantages principaux est la possibilité d'acheminer les données standard et de sécurité sur le même médium.

Ce travail est un exemple de modélisation et d'analyse d'une partie du protocole de communication AS – i, tel qu'il est décrit dans la référence principale NF EN 50295 [NF 50295].

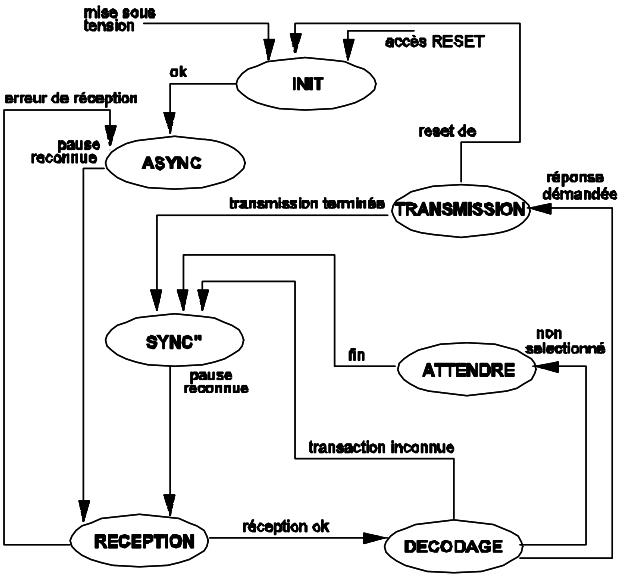
7.3 MONTE CARLO, GRAPHE D'OCCURRENCE ET LEUR COMPLEMENTAIRITE

Comme c'est expliqué dans les chapitres précédents, la simulation de Monte Carlo est une méthode basée sur la répétition des simulations de fonctionnement du modèle. Le nombre de répétitions est plutôt élevé, habituellement placé entre 10^3 et 10^5. Les résultats ainsi obtenus sont ensuite sujet à une analyse statistique pour calculer dans la plupart des valeurs et comportements moyens tel que le MTTF, sa déviation ou des fréquences d'apparition.

De l'autre côté, le graphe d'occurrence (Occ graphe) contient tous les états atteignables à partir du marquage initial et cette méthode est donc bien adaptée pour la recherche sur les conséquences d'apparitions des événements rares qui sont difficilement observables pendant la simulation. La construction du graphe d'occurrence commence à l'état initial du système à partir duquel tous les états immédiatement atteignables à l'instant même sont déterminés après quoi le temps avance d'un pas et les nouveaux états atteignables sont identifiés. C'est pour cette raison que la méthode de construction s'appelle « width-first » (d'abord en largeur). Par définition, un graphe d'occurrence pour des systèmes temporisés est infini puisque le temps ne s'arrête jamais. Il est donc nécessaire pour une analyse raisonnable de choisir un intervalle d'observation ou d'imposer d'autres contraintes [Liu 01] pour borner l'évolution possible du système. Il faut remarquer que la nature stochastique du système analysé contribue d'une manière non négligeable à l'explosion combinatoire du graphe d'occurrence.

L'utilisation de graphe d'occurrence peut être recommandée pour les analyses de sécurité et sûreté de fonctionnement puisqu'elle peut être considérée comme un outil de vérification formelle à condition que le modèle corresponde parfaitement au système modélisé. Pour cela, l'utilisation du graphe d'occurrence est au cœur de cette étude.

7.4 TRANSPOSITION D'UNE NORME VERS UN MODELE CPN



Après mise sous tension ou après reset de l'esclave par l'accès RESET ou après réception d'une requête reset_AS-i_slave l'esclave doit être mis à l'état INIT.

Dans l'état INIT les fonctions suivantes doivent être exécutées:

- charger les registres de sortie de données et de sortie de paramètres avec la valeur par défaut 0xF;
- remettre le registre d'état à 0x0;
- charger l'adresse, la configuration E/S et le code identification de la mémoire non volatile dans les registres appropriés et contrôler les erreurs de lecture;
- mettre la marque data_exchange_disable (rendre échange de données inutilisable) à TRUE (vrai);
- passer à l'état ASYNC.

Figure 7.1 : Extrait de la norme NF EN 50295 (version française)

La NF EN 50295 contient une spécification textuelle pour le protocole AS-i. Le texte est largement illustré avec des schémas qui facilitent sa compréhension. La NF EN 50295 est composé de 9 chapitres et deux annexes. Elle couvre tout le domaine concernant la définition du protocole, dès les définitions et propriétés de bases jusqu'aux contraintes de construction et tests de compatibilité. Une couche supplémentaire, qui n'est pas précisée dans cette norme peut être rajoutée à cette spécification pour former un protocole étendu AS-i Safety at Work.

La partie développée dans la présente étude est le modèle des états internes de l'esclave et leur évolution, comme c'est décrit dans le paragraphe 8.2.2.6 de la norme. Un extrait de ce paragraphe est

montré sur la fig. 7.1. Les états internes sont décrits sur deux pages à l'aide d'un diagramme et une description textuelle des actions associées à chaque état accompagnée de conditions d'évolution pour les passages entre les états.

Chaque esclave de communication doit se trouver impérativement dans un des états : INIT, ASYNC, RECEIVING, DECODING, WAIT, TRANSMIT et SYNC. Les états internes sont transformés vers des places (ellipses) dans le modèle CPN. Les passages entre les états internes, des arcs orientés sur la figure 7.1, sont dans le modèle CPN représentés comme des transitions (rectangles). La structure de base du modèle CPN ainsi obtenu se trouve sur la figure 7.2.

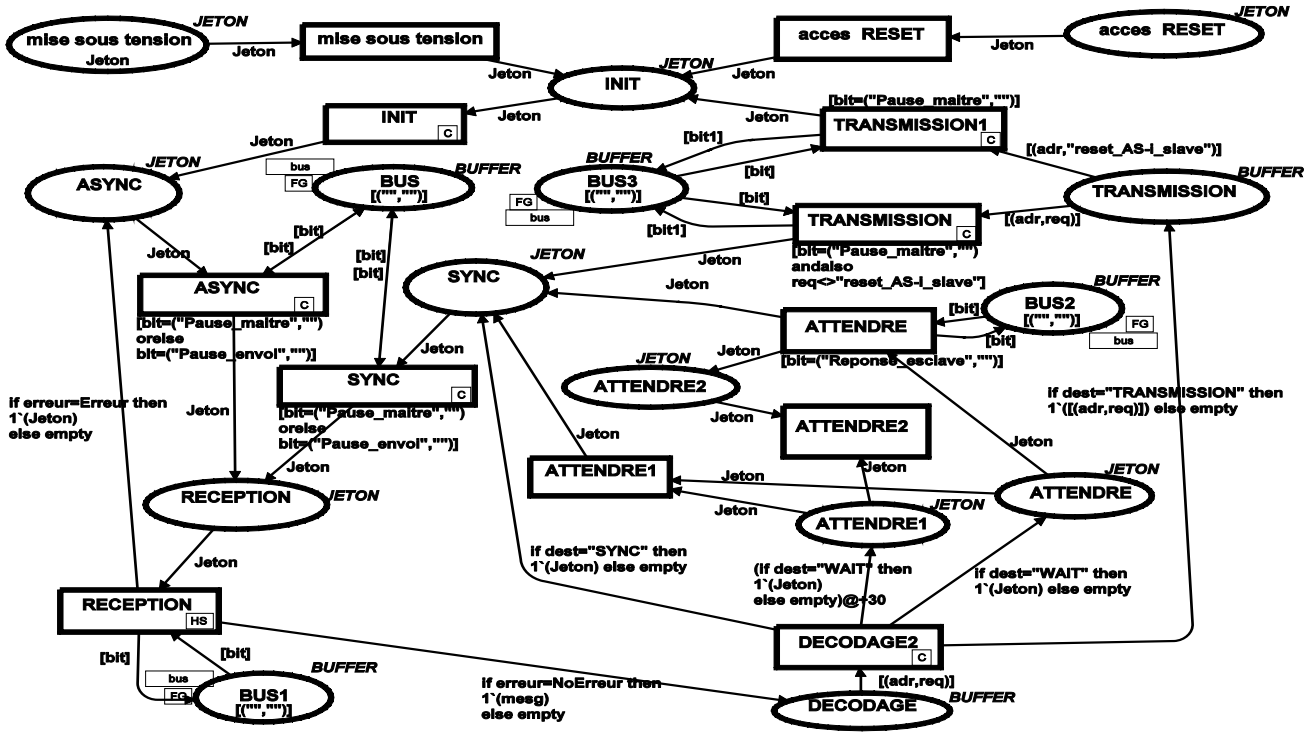


Figure 7.2 : Structure du modèle CPN

Dans la suite le passage de l'état INIT vers l'état ASYNC et sa transposition vers un modèle CPN sont expliqués en détail. Selon la norme (figure 7.1), une fois que le système arrive dans l'état INIT il est censé charger les registres *sortie* et *paramètre* avec la valeur par défaut *0xF*. Dans Design/CPN aucune action ne peut être associée avec les places, mais uniquement avec les transitions. Pour cette raison, les actions d'initialisation sont placées dans le segment du code de la transition reliant les états INIT et ASYNC. Les registres peuvent être représentés de manières différentes dans Design/CPN. Celle retenue pour ce modèle est l'utilisation des déclarations des variables globales à l'aide du déclaratif

`globref` dans le nœud de déclarations globales. Chaque modèle CPN doit impérativement contenir au moins un nœud de déclaration global, mais il peut en avoir plusieurs et d'autres nœuds de déclarations sont possibles (locales ou limitées). Un exemple d'une partie d'un nœud de déclaration est montré sur la figure 7.3. Une autre façon de représenter des registres est l'utilisation des places CPN. Chaque place représenterait alors un registre. L'inconvénient de cette méthode est l'augmentation du nombre des arcs et des places, ce qui réduit la lisibilité des modèles.

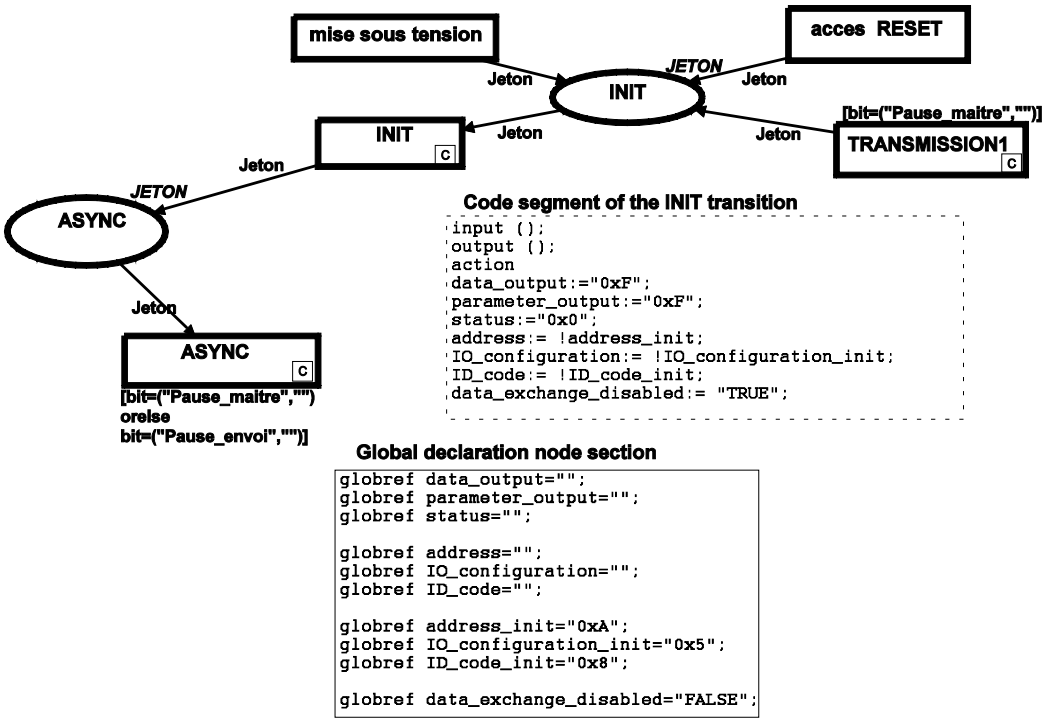


Figure 7.3 : Détail de la transition INIT – ASYNC

Les registres requis sont actualisés dans la section d'actions dans le segment du code associé à la transition, comme c'est montré sur la fig. 7.3. Une fois cette actualisation effectuée, le registre *état* doit être remis à 0 (valeur *0x0*) d'une manière analogue à l'attribution précédente. De la même façon les valeurs *I/O_configuration* et *code_identification* sont chargées de la mémoire interne vers les registres appropriés. La mémoire interne est représentée comme un ensemble de références globales (`globref`) définies dans le nœud de déclaration global. Pour simplifier la construction du modèle, tous les registres sont déclarés de type chaîne de caractères.

Selon la norme à ce point d'avancement, il faut vérifier les erreurs de réception. Ce mécanisme n'est pas intégré dans le présent exemple, car les auteurs préfèrent rester à un niveau macroscopique (niveau

des trames) et ne pas modéliser le niveau des bits qui est nécessaire pour pouvoir correctement vérifier les erreurs de réception, ceci par souci de faciliter la compréhension du modèle par un public élargi. Néanmoins, le modèle peut en cas de besoin être modifié et intégrer également le niveau des bits.

Finalement, un indice *data_exchange_disable* doit être mis à un avant que l'état interne de l'esclave ne passe à ASYNC. Ainsi, toutes les actions et conditions pour l'évolution d'état (à l'exception de la vérification d'erreur de réception) sont faites et l'esclave se trouve à présent dans l'état ASYNC où il peut commencer à vérifier l'état du bus et attendre la pause maître.

Dans les paragraphes précédents, on voit une description détaillée de la transposition d'un passage entre deux états internes de l'esclave. Le modèle des autres passages a été développé d'une manière analogue et n'est pas présenté ici. Quelques mécanismes supplémentaires ont dû être utilisés pour accomplir cela et les plus importants parmi eux sont :

- les places de fusion et transitions de substitution pour mettre en place la composition hiérarchique,
- les fonctions logiques : comparaisons, tests et conditions,
- les expressions de garde sur les transitions pour permettre/interdire leur validation,
- l'implémentation du temps dans le modèle.

Pendant la conception du modèle quelques passages problématiques de la norme, notamment plusieurs incohérences, ont été recensés. Premièrement, on peut trouver une séquence dans la norme :

```
Dans l'état DECODING les fonctions suivantes doivent être exécutées°:
   - comparer l'adresse reçue avec le registre d'adresses; si elles ne sont pas
     égales passer à l'état WAIT;
   - analyser l'information; si la requête est inconnue, passer à l'état SYNC;
   - … .
```

La transformation de ce prédicat vers une structure if-then-else standard est plutôt ambiguë. On peut la comprendre soit comme :

```
1.) si l'adresse reçue <> registre d'adresses alors passer à l'état WAIT, sinon (si
requête = inconnue alors passer à l'état SYNC, sinon (…))
```

ou

```
2.) si l'adresse reçue <> registre d'adresses alors passer à l'état WAIT,
sinon ???, si requête = inconnue alors passer à l'état SYNC, sinon ???).
```

Il est évident que ces deux interprétations de la norme mènent vers des résultats différents. Ce problème est une conséquence du choix fait pour la composition de la partie textuelle de la norme. En effet, la structuration et la relation entre les différentes parties du texte devraient, à notre avis, être précisées d'une manière plus explicite que par leur simple succession chronologique.

Un autre problème vient d'une forte ressemblance entre les états ASYNC et SYNC. Dans l'état SYNC l'indice *sync* doit être mis à 1. Dans l'état ASYNC le même indicateur doit être remis (à un). Alors *sync* est mis deux fois à 1 et jamais à 0. C'est pour quoi nous soupçonnons un problème de traduction. En effet, l'interprétation et le comportement du modèle changent si on établit la correspondance entre le mot anglais *set* et le mot français *mettre* (à 1) et *reset* et *remettre* à 0 (réinitialiser).

7.5 METHODE D'ANALYSE

Après avoir créé le modèle, survient la phase de son analyse. Ceci signifie l'observation des réactions du modèle et évolution de l'état d'esclave en fonction des signaux extérieurs comme par exemple la présence d'un bit sur le bus. Le choix de l'ensemble des événements observés est un facteur crucial pour le concepteur. En effet, ce choix est une fonction du savoir-faire du concepteur et son 'intuition' est un facteur déterminant pour la valeur de l'analyse.

Pour observer le comportement de l'esclave – récepteur un modèle d'émetteur est nécessaire. A ce stade de notre travail, l'émetteur maître n'est pas un modèle conforme aux spécifications AS-i mais seulement un émetteur simplifié qui envoie une séquence pseudo-aléatoire de requêtes de communication. L'envoi sur le bus est modélisé en plaçant la trame de communication dans une place appelée BUS. Les auteurs sont conscients qu'il s'agit ici d'une approximation grossière, mais elle est entièrement suffisante pour le but de l'étude.

La séquence pseudo-aléatoire de requêtes envoyée par le maître est conçue pour mettre en évidence le comportement de l'esclave dans le plus grand nombre de situations et elle comporte :

- des requêtes standard adressées à l'esclave observé (avec l'adresse de destination correspondante), par exemple *data_exchange* ou *write_parameter*,
- des trames standard adressées à un autre esclave que celui observé, pour vérifier les réactions de l'esclave observé,

- des messages corrompus, par exemple un bit de parité non valide ou la longueur de trame mal formée.

Ces trois types de trames sont organisés par l'émetteur dans une séquence pseudo-aléatoire et copiés progressivement sur le bus, qui est représenté sur la fig. 7.2 comme une place fusion composée des places BUS, BUS1, BUS2 et BUS3. Une place fusion est une copie identique de la place d'origine. En effet, elle peut être considérée comme une seule place avec plusieurs copies de sa représentation graphique.

Design/CPN permet une génération automatique du graphe d'occurrence (figure 7.4). Après l'avoir créé, il y a plusieurs requêtes prédéfinies pour son analyse. Il y a également une possibilité de construire des requêtes selon les spécifications de l'utilisateur.

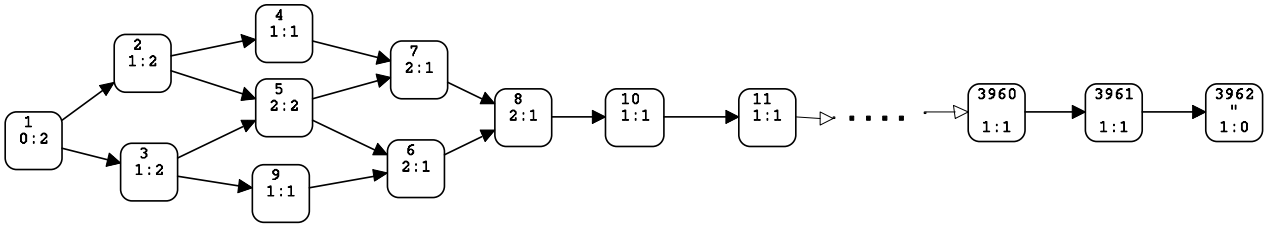


Figure 7.4 : Graphe d'occurrence

La première information fournie par le logiciel est la présence d'un état puits dans le graphe d'occurrence. Un état puits est un état à partir duquel le modèle ne peut plus évoluer [Dav 97], [Dia 01]. Bien sûr la présence d'un tel état est indésiré dans la mesure où l'esclave doit toujours continuer à réagir sur les requêtes du maître. Pour trouver le nombre d'états puits, il faut parcourir chaque état du graphe d'occurrence et vérifier le nombre de ses successeurs. Si ce nombre est égal à 0 alors un état puits peut être présent.

Dans cette expérience, un état puits parmi plus que 3900 états atteints à été trouvé. Mais si on examine cet état en détail, on trouve qu'il s'agit du dernier état calculé et pour lui les successeurs ne sont encore pas calculés. Alors la valeur par défaut 0 est utilisée. Les successeurs peuvent être calculés facilement, mais cela ne résout pas le problème, mais le décale simplement. En conclusion il faut dire que si seulement une partie (coupe) de graphe d'occurrence est calculée, il faut toujours vérifier si les places puits le sont réellement ou s'il s'agit seulement des limites de la coupe calculée.

La présence d'un seul état puits peut renseigner sur le comportement du système communiquant : son comportement est presque toujours déterministe. La seule exception à cette règle est l'instant d'initialisation du modèle où plusieurs actions sont effectuées quasi simultanément pour initialiser tous les composants. Après ce bref intervalle, le comportement du système converge rapidement vers une évolution complètement déterministe. Ce propos est encore confirmé par le nombre d'arcs, qui est légèrement supérieur (de moins de 10) au nombre des états atteignables. Dans le cas d'un comportement purement déterministe (séquence linéaire des états) le nombre d'arcs était inférieur d'une unité au nombre des états.

Toutes ces informations peuvent être obtenues du compte-rendu standard établi automatiquement par le logiciel. Entre autre, il y a aussi des renseignements sur la « bornitude » ou la vivacité. Une information est également donnée, si le graphe a été calculé en intégralité ou seulement en partie (c'est le cas de cette étude).

Toutes les informations résumées dans le compte-rendu standard restent très générales. Mais le potentiel de l'analyse de graphe d'occurrence est beaucoup plus important. Les critères définis par l'utilisateur peuvent être effectués moyennant la capacité de programmation de ces requêtes. Le passage entre les états TRANSMIT et INIT peut être pris pour un exemple d'une telle analyse. Selon EN 50295, ce passage doit exclusivement avoir lieu si la trame *reset_AS-i_slave* a été correctement reçue par l'esclave. Alors pour vérifier que chaque demande de reset est traitée correctement il suffit de comparer le nombre de passages entre TRANSMIT et INIT avec le nombre d'apparitions de cette requête sur le BUS. Rappelons qu'aucune corruption des messages sur le bus n'est considérée à ce stade de l'étude.

Après avoir appelé la fonction SearchNodes qui parcourt tous les états du graphe d'occurrence, 106 états contenants les demandes de reset sont trouvés. Il reste à déterminer les passages entre TRANSMIT et INIT. Pour cela il faut répertorier tous les états où l'esclave se trouve dans l'état TRANSMIT et après examiner les successeurs directs de ces états. Il faut compter seulement les cas où INIT suit immédiatement le TRANSMIT. 53 cas sont ainsi recensés.

Il est évident que même si les deux résultats ne sont pas identiques, ils sont fortement corrélés. Dans la suite de l'analyse il a été découvert que les 106 états avec la trame reset sur le BUS se produisent toujours en couples de deux états successifs. Il est relativement facile à démontrer que la répétition de

ces états est due aux tests effectués sur le BUS. Ainsi on peut conclure que toutes les demandes *reset_AS-i_slave* sont reconnues et traitées correctement.

7.6 SYNTHESE DES RESULTATS DE MODELISATION ET D'ANALYSE

Cette section propose une synthèse des résultats principaux des parties précédentes :

1. CPN et Design/CPN permettent la conception d'un modèle formel d'une spécification comme EN 50295.
2. Le modèle peut être simulé et permet ainsi au concepteur d'acquérir une meilleure compréhension du modèle et de son comportement.
3. La simulation de Monte Carlo permet l'analyse des performances.
4. Un graphe d'occurrence peut être généré automatiquement. Dans le présent exemple le graphe est infini et, par conséquent, seule une partie est calculée.
5. L'absence de blocage (états puits) est vérifiée d'une manière non exhaustive.
6. La différence entre le nombre d'arcs et d'états procure des renseignements sur la structure du graphe.
7. Les fonctions de recherche définies par l'utilisateur offrent une grande souplesse pour l'analyse des nombreux critères sur le graphe d'occurrence (ex. : *reset_AS-i_slave*).

7.7 CONCLUSION

La plus grande partie de cette étude porte sur la modélisation et l'analyse d'un protocole de communication réel. Les CPN ont validé leur potentiel dans ce type d'application. Le problème le plus important dans la phase de modélisation est l'absence d'une spécification suffisamment approfondie. En présence des ambiguïtés le concepteur doit interpréter la spécification et introduit ainsi une subjectivité dans le modèle, ce qui peut nuire à une analyse objective. Répertorier les ambiguïtés et les incohérences permet à son tour d'améliorer la qualité de la spécification [Bla 00].

Un modèle CPN peut être analysé de deux façons : 1.) performance et critères QdS et 2.) Occ graphe, qui est approprié pour les études de sûreté/sécurité et peut être considéré comme une méthode d'analyse formelle.

Un modèle constitué d'un esclave et d'un maître est présenté dans cette étude. Mais la méthode n'est pas limitée à la modélisation des composants seuls. Chaque sous-système modélisé peut être utilisé ensuite comme une brique à intégrer dans un système plus complexe. Touts les composants peuvent d'abord être vérifiés séparément et après une analyse du système complet est toujours possible. Ceci est un atout majeur de la composition hiérarchique d'un modèle CPN et peut amplement faciliter le travail avec des systèmes de tailles conséquentes. Ainsi ce modèle peut directement devenir une partie des modèles présentés dans les études précédentes.

Le protocole complet n'est pas modélisé dans son intégralité à l'état actuel, puisque le but est de vérifier la faisabilité d'une analyse d'une norme par les CPN et de préparer une étude plus poussée. Cette deuxième étude, qui récupère les travaux déjà accomplis, est en cours actuellement.

La vérification d'absence de blocage, telle qu'elle est présentée ici, ne peut pas être considérée comme une preuve formelle. Une preuve par induction pour toute taille de la portion calculée serait nécessaire. Mais il est possible qu'en faisant usage des caractéristiques structurelles, ce problème puisse être évité [Chr 01]. Ceci notamment par une transformation du graphe infini vers un graphe fini.

Ce travail a été repris dans [Seb 03] pour son stage de DEA. Son travail a été en partie l'extension des modèles existant et de l'autre côté une proposition de méthodes d'évaluation de la sécurité du protocole de communication. Le constat final est favorable à l'utilisation de Design/CPN pour une étude de sécurité grandeur nature.

8 Conclusion

La problématique de l'évaluation de la sûreté de fonctionnent des systèmes d'automatisation à intelligence distribuée dépasse largement le cadre d'une thèse. De nombreuses questions attendent des réponses. Par conséquent, cette thèse ne se veut ni exhaustive ni complète.

Le but de la thèse était de vérifier la pertinence d'une approche orientée RdP pour les systèmes d'automatisation à intelligence distribuée, notamment les systèmes comprenant un réseau de communication pour en estimer les critères de la sûreté de fonctionnement.

La méthode utilisée est constituée des étapes suivantes :

- le développement des modules des composants,
- l'association des composants pour former un système complet,
- l'analyse du système par simulation de Monte Carlo.

La modularité de cette approche permet de créer des bibliothèques de composants et ensuite d'alléger la conception d'un système aussi bien en terme de temps d'analyse qu'en terme de coût. Elle s'inscrit alors pleinement dans le projet d'étude du laboratoire qui est le développement d'un outil d'aide à la conception des systèmes distribués.

En l'état actuel, un recensement des problèmes a été effectué. Il a permis de voir de nombreuses facettes du problème de synthèse et d'analyse des systèmes d'automatisation distribués. Ces problèmes vont de la conception d'un régulateur, via le choix des composants et des architectures jusqu'à la certification des installations.

L'étude bibliographique du sujet a confirmé le potentiel des RdP dans le domaine de la modélisation des systèmes distribués, ainsi que dans le cadre d'application aux problèmes de la sûreté de fonctionnement en permettant entre autre une expression facile du comportement stochastique des systèmes.

Le travail pratique se divise en deux parties. La première est la modélisation et la seconde l'analyse des modèles. La modélisation grâce à l'outil Design/CPN offre des possibilités inégalées quant à la prise en compte des différents points de vue relatifs aux systèmes distribués. Toutes les nuances de tous les

composants d'un système peuvent être exprimées. C'est par exemple le cas du mécanisme de défaillance de la pompe lorsqu'elle reste bloquée au démarrage. L'avantage principal de la modélisation CPN est donc la possibilité de modéliser l'architecture d'un système à intelligence distribuée sous les points de vue fonctionnel (commande) et dysfonctionnel (sûreté de fonctionnement) dans un modèle unique.

L'analyse des modèles s'est avérée quant à elle beaucoup plus délicate. La vitesse de simulation sur un ordinateur standard ne permet pas l'utilisation de constantes proches de la réalité. Nous sommes alors obligés de travailler avec des constantes très élevées, ce qui peut convenir aux études des scénarios défaillants, mais ne permet pas d'étudier pleinement la disponibilité. Des méthodes d'accélération et d'améliorations de performances existent [Aub 03] mais n'ont pas encore été intégrées à ce travail.

La méthode utilisée a permis d'évaluer des applications distribuées aussi bien au niveau des performances de commande que du point de vue de la sûreté de fonctionnement. L'usure des composants ainsi que des scénarios défaillants peuvent être estimés. La thèse présente aussi une méthodologie détaillée de modélisation des composants des systèmes distribués et s'intéresse au problème de cohabitation de sous-ensembles à événements discrets avec des sous-ensembles échantillonnés [Lig 04].

Le présent travail a mis en évidence les avantages et les inconvénients d'une telle méthode ainsi que de pressentir ses limites. Parmi les avantages on peut inclure l'uniformité de l'approche de modélisation tout au long de l'étude ou encore la finesse des modèles CPN. Mais la finesse a un coût qui est celui d'une lenteur de simulation. Alors le choix du niveau de détail est un compromis entre le modèle qui exprime tout et celui dont la simulation est rapide.

Le problème principal rencontré est donc la vitesse de simulation. En effet, à la construction de l'outil Design/CPN, les concepteurs étaient loin d'imaginer des modèles aussi conséquents. Alors leur moteur d'analyse n'est pas optimisé pour des grands modèles. C'est pourquoi ils ont proposé une extension du logiciel, dite NewSim, dotée d'un nouveau solveur optimisé et par conséquent beaucoup plus performant. L'utilisation de ce solveur n'est pas faite dans ce travail à cause de l'état avancé des modèles et quelques nuances d'incompatibilité d'un modèle Design/CPN classique avec celui de NewSim. A présent, nous considérons que tous nos modèles peuvent être adaptés pour cette nouvelle extension.

Pour permettre à ce travail de continuer dans la suite, il est impératif de revoir la vitesse d'analyse. Pour cela il faudrait refaire un essai comparatif de vitesse d'analyse d'un modèle équivalent par plusieurs logiciels disponibles. Il est à noter que certains d'entre eux ont fait l'objet d'une évolution importante depuis le début de la thèse. Si aucun des logiciels pour les RdP colorés ne permettait une analyse acceptable, la méthode d'analyse par sauts pourrait être envisagée. En tout cas, le retour vers les RdP non colorés n'est pas possible si l'intégralité d'un système doit être prise en considération.

Ces travaux s'inscrivent dans la démarche de construction d'un outil d'aide à la conception des systèmes sûrs de fonctionnement. Dans cette optique nous disposons à présent d'une méthodologie optimisant le choix des composants du système d'automatisation du point de vue statique. Un modèle de ce système peut ensuite être réalisé et son fonctionnement peut être validé du point de vue dynamique. Il peut également être comparé d'une manière quantitative avec les comportements d'autres systèmes. L'étape suivante dans cette démarche générale est la génération des preuves formelles garantissant le respect des critères de la sûreté de fonctionnement imposés.

Glossaire

ADLN	Advanced Distributed Learning Network
AMDE	Analyse des Modes de Défaillance et de leurs Effets
APD	Analyse Préliminaire des Dangers
APM	Modulation d'Impulsions Alternatives
AS-i	Actuator – Sensor Interface, réseau de terrain
AS-i Safety at Work	extension de sécurité du réseau AS-i
BDF	Blocs Diagrammes de Fiabilité
CPN	Coloured Petri Net, Réseaux de Petri coloré
E-net	Evaluation nets, proposés par G. Nutt comme une extension des RdP
ESTELLE	Extended State Language
FT	Fonction de Transfert
HiP-HOPS	Hierarchically Performed Hazard Origin and Propagation Studies
hu	unités de hauteur
LOTOS	Language of Temporal Ordering Specifications, langage de spécifications d'ordonnancement temporel
MAC	Méthode de l'Arbre des Causes
MACQ	Méthode de l'Arbre des Conséquences
MCPR	Méthode des Combinaisons de Pannes Résumées
MDCC	Méthode du Diagramme Causes-Conséquences
MDS	Méthode du Diagramme de Succès
MEE	Méthode de l'Espace d'Etats
MISS-RdP	logiciel RdP colorés, commercialisé par la société IXI – GFI consulting
ML	langage de programmation ML standard [Ull 98]
Moca-RP	logiciel RdP stochastiques, commercialisé par la société IXI – GFI consulting
MTV	Méthode de la Table de Vérité
NCS	Networked Control System, système d'automatisation à réseau
NewSim	extension proposée pour Design/CPN avec un solveur optimisé
ns-2	simulateur des événements discrets, http ://www.isi.edu/nsnam/ns/
ODE	algorithme de solution numérique des équations différentielles
PI	régulateur Proportionnel et Intégral
QdS	Qualité de Service
RdP	Réseaux de Petri
SADT	Structure Analysis and Design Technique
SAID	Système d'Automatisation à Intelligence Distribuée

SAR	Système d'Automatisation à Réseau
SdF	Sûreté de Fonctionnement
SDL	Specification and Description Language
SML/NJ	langage de programmation ML standard [Ull 98], version New Jersey
Te	période d'échantillonnage
TOD	Try-Once-Discard, protocole de communication
TOR	régulateur Tout-Ou-Rien
tu	unités de temps

Bibliographie

[Aho 93] A. Aho, J. D. Ullman, Concepts fondamentaux de l'informatique, Dunod, Paris, 1993.

[Aub 03] J.-F. Aubry, E. Guilhem, T. Hutinet, R. Schoenig, Exemple de Calcul de la Fiabilité d'un Système Hybride, 4th International Conference on Industrial Automation, École de technologie supérieure, Montréal, June 9-11 2003.

[Bar 00] P. Barger, Modélisation en vue de l'analyse de performances et de la simulation d'un système d'automatisation à intelligence distribuée, raport de DEA ATNS, CRAN, Université Henri Poincaré, Nancy 1, 2000.

[Bar 02] P. Barger, J.-M. Thiriet, Evaluation of networked control systems thanks to a dynamical reliability approach, CCF 2002, Sinaia, Roumanie, septembre 2002.

[Bar 02a] P. Barger, J.-M. Thiriet, M. Robert, Performance and dependability evaluation of distributed dynamical systems, European Conference on System Dependability and Safety (ESRA 2002/lambda-Mu13), Lyon (France), 19-21st March 2002, pp. 16-22.

[Bar 02b] P. Barger, J.-M. Thiriet, M. Robert, Dynamical reliability and availability evaluation and validation of distributed control systems, IEEE Instrumentation and Measurement Technology Conference Anchorage, AK, USA, 21-23 May 2002.

[Bar 03] P. Barger, J.-M. Thiriet, M. Robert, Safety analysis and reliability estimation of a networked control system, IFAC-Safeprocess 2003, Washington (USA), June 2003.

[Bar 03a] P. Barger, J.M. Thiriet, M. Robert, Dependability Analysis of a Distributed Control or Measurement Architecture, 20th IEEE Instrumentation and Measurement Technology Conference (IEEE/IMTC2003), Vail (Colorado, United States), 20-22nd May 2003, pp. 473-477, ISBN 0-7803-7705-2.

[Bar 03b] P. Barger, J.M. Thiriet, M. Robert, Modeling and analyzing Safety Criteria of a real communication protocol, 5ème Congrès International Qualité et Sûreté de Fonctionnement, Qualita 2003, Nancy, France, 19-21 Mars 2003.

[Bat 00] M. Battaini, F. Casciati, L. Faravelli, Some reliability aspects in structural control, Probabilistic Engineering Mechanics 15, 2000, pp. 101-107.

[Bla 00] J.-C. Blaise, Apport d'une modélisation de l'information normative à l'intégration des règles de sécurité des machines en conception, thèse de doctorat, Université Henri Poincaré, Nancy 1.

[Blu 98] I. Blum, G. Juanole, Modélisation et evaluation du protocole MAC du réseau CAN et étude de l'influence de sa qualité sur les performances d'une boucle d'asservissement, Conference on Real-time and Embedded Systems (RTS'99), Paris (France), 10-12 Février 1999, pp.22-32.

[Bor 00] E. Borgonovo, M. Marseguerra, E. Zio, A Monte Carlo methodological approach to plant availability modeling with maintenance, aging and obsolescence, Reliability Engineering and System Safety 67, 2000, pp. 61-73.

[Bra 03] M. S. Branicky, V. Liberatore, S. M. Phillips, Networked control system co-simulation for co-design, Proc. American Control Conf., Denver, June 4-6, 2003.

[Bus 98] L. Bushnell, O. Beldiman, G. C. Walsh, An Equivalence Between a Control Network and a Switched Hybrid Systems, Lecture Notes in Computer Science, v. 1386, Springer-Verlag 1998, pp. 64-80.

[Cas 03] P. Caspi, Automatique continue, automatique discrète, informatique industrielle : le triangle des Bermudes ?, Modélisation des systèmes réactifs (MSR), Metz, 2003, pp. 243-260.

[Châ 98] E. Châtelet, J. L. Chabot, Y. Dutuit, Events Representation in Dynamic Reliability Analysis using Stochastic Petri Nets, International Workshop Series on Advanced Topics in Reliability and Risk Analysis, Greenbelt, Maryland, USA, septembre 1998, pp. 101-122.

[Châ 98a] E. Châtelet, E. Zio, S. Pasquet, The Use of Neural Networks in the Reliability Analysis of Dynamic Systems: An overview, International Workshop Series on Advanced Topics in Reliability and Risk Analysis, Greenbelt, Maryland, USA, septembre 1998, pp. 123-145.

[Châ 02] E. Châtelet, C. Bérenguer, O. Jellouli, Performance assessment of complex maintenance policies using stochastic Petri nets, Lambda-Mu 13 – Esrel 2002 European conference, Lyon, 2002.

[Che 96] To-yat Cheung, Petri Nets for protocol engineering, Computer communications 19, 1996, pp. 1250-1257.

[Chr 01] S. Christensen and K. Jensen and T. Mailund and L. M. Kristensen. State – Space Methods for Timed Coloured Petri Nets. In Proceedings of 2nd International Colloquium on Petri Net Technologies for Modeling Communication Based Systems. Berlin Germany, September 14-15, 2001.

[Con 99] B. Conrard, Contribution à l'évaluation quantitative de la sûreté de fonctionnement des systèmes d'automatisation en phase de conception, thèse de doctorat, CRAN, Université H. Poincaré, Nancy 1, 1999.

[Dai 03] Y. S. Dai, M. Xie, K. L. Poh, G. Q. Liu, A study of service reliability and availability for distributed systems, Reliability Engineering and System Safety 79, 2003, pp. 103-112.

[Dav 97] R. David, H. Alla, Du Grafcet aux réseaux de Petri, 2e édition revue et augmentée, Editions Hermès, Paris, 1992, 1997.

[design] http ://www.daimi.aau.dk/designCPN

[Dev 02] N. Devictor, M. Marques, S. Boulègue, P. Lamagnère, M.P. Valeta, M. Eid, Analyse statistique de la ténacité : utilisation de méthodes de Monte-Carlo pour estimer l'incertitude sur la distribution de la ténacité, Lambda-Mu 13 – Esrel 2002 European conference, Lyon, 2002.

[Dia 01] M. Diaz (sous la direction de), Les réseaux de Petri – Modèles fondamentaux, Hermès Sciences Publications, Paris, 2001.

[Dod 93] Y. Dodge, Statistique, Dictionnaire encyclopédique, Dunod, Paris, 1993.

[Dod 99] B. Dodson, D. Nolan, Reliability Engineering Handbook, QA Publishing, LLC, Tucson, USA, 1999.

[Duf 02] F. Dufour, Y. Dutuit, Fiabilité dynamique : un nouveau modèle, Lambda-Mu 13 – Esrel 2002 European conference, Lyon, 2002.

[Fen 02] F.-L. Lian, J. R. Moyne, D. M. Tilbury, Network Design Consideration for Distributed Control Systems, IEEE Transactions on Control Systems Technology 10(2), March 2002, pp. 297-307.

[Gar 02] C. J. Garrett, G. E. Apostolakis, Automated hazard analysis of digital control systems, Reliability Engineering ans System Safety 77, 2002, pp. 1-17.

[greatspn] http ://www.di.unito.it/~greatspn/index.html

[Had 99] S. Haddad, D. Poitrenaud, Theoretical Aspects of Recursive Petri Nets, 20th International Conference on Application and Theory of Petri Nets. Williamsburg. June 1999.

[Jel 02] O. Jellouli, E. Châtelet, P. Lallement, Application of Stochastic Synchronized Petri Nets in Suplly Chain Inventory Management, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Jen 97] K. Jensen, Coloured Petri Nets. Basic Concepts, Analysis Methodes and Practical Use, Monographs in Theoretical Computer Science, Springer Verlag, 2nd corrected printing, 1997.

[Jua 95] G. Juanole, Réseaux de Petri et communications, Rapport LAAS No95296, Réseaux de Communications et Conception de Protocoles, Coordonnateurs: G. Juanole, A. Serhrouchni, D. Seret. Hermès, N°ISBN 2-86601-478-2, 1995, pp.265-287.

[Jua 02] G. Juanole, Quality of service of communication networks and distributed automation, models and performances, Invited paper, 15th Triennial World Congress of the IFAC, Barcelona, Spain, 21-26 juillet 2002.

[Kab 97] I. V. Kabashkin, Dynamic Reliability Models of Complex Engineering Systems based on Petri Nets, Automatic Control and Computer Sciences, Vol. 31, No. 3, 1997, pp. 46-51.

[Kam 99] M. Kamei, Y. Sugasawa, Petri Net Model and Performance Evaluation of an Autonomous Decentralized Network System, Computers and Mathematics with Applications 38, 1999, pp. 165-173.

[Kha 02] S. Khalfaoui, E. Guilhem, H. Demmou, R. Valette, Une méthode pour obtenir des scénarios critiques dans les systèmes mécatroniques, 13ème Colloque Européen de Sûreté de Fonctionnement (ESREL'2002), Lyon (France), 19-21 Mars 2002, pp. 455-461.

[Kne 01] J. Knezevic, E. R. Odoom, Reliability modelling of repairable systems using Petri nets and fuzzy Lambda-Tau methodology, Reliability Engineering and System Safety 73, 2001, pp. 1-17.

[Lab 00] P. E. Labeau, C. Smidts, S. Swaminathan, Dynamic reliability: towards an integrated platform probabilistic risk assessment, Reliability Engineering and System Safety 68, 2000, pp. 219-254.

[Lab 02] P.E. Labeau, E. Zio, Procedures of Monte Carlo transport simulation for applications in system engineering, Reliability Engineering and System Safety 77, 2002, pp. 217-228.

[Lar 02] E. Lardeux, Medical Gases Filling Center Petri net Modeling : From performance assessment to safety assessment, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Leb 02] A. Leblond, P. Cordat, R. Schoenig, J.-F. Aubry, E. Guilhem, Méthodologie outillée d'aide à la conception de systèmes embarqués « sûrs de fonctionnement », Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Li 01] G. Li, Y. Song, P. Xiao, H. Jin, Monitoring System of Car Information Central Control System, 4th IFAC International Conference on Fieldbus Systems and their Applications – FeT'2001, Nancy, France, Elsevier Scientific, November 2001.

[Lia 01] F.-L. Lian, J. R. Moyne, D. M. Tilbury, Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet, IEEE Control Systems Magazine, Vol. 21, No. 1, February 2001, pp. 66-83.

[Lia 02] F.-L. Lian, James R. Moyne, D. M. Tilbury, Optimal Controller Design and Evaluation for a Class of Networked Control Systems with Distributed Constant Delays, 2002 American Control Conference, May 2002.

[Lig 03] Ligušová, J. , Liguš, J., Barger, P., Modification of the Smith predictor for random delays treatment in the network control systems, 2nd IFAC Conference CONTROL SYSTEMS DESIGN (CSD'03) Bratislava, Slovak Republic, September 7 – 10, 2003.

[Lig 04] J. Ligušová, J.-M. Thiriet, J. Liguš, P. Barger, Effect of Element's Initialization in Synchronous Network Control System to Control Quality, Annual Reliability and Maintainability Symposium – RAMS 2004, Los Angeles, USA, 27-30 Janvier 2004, (à paraître).

[Lin 02] Q. Ling, M.D. Lemmon, Robust performance of soft real-time networked control systems with data dropouts, 2002 IEEE conference on decision and control, Las Vegas, Nevada, December 2002.

[Liu 01] L. Liu, J. Billington. *Modeling and Analysis of the CES Protocol of H.245*. Third Workshop and tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, August 29-31, 2001, pp. 95-114.

[Mar 95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, Modelling with Generalized Stochastic Petri Nets Wiley Series in Parallel Computing John Wiley and Sons, ISBN : 0 471 93059 8, 1995.

[Mar 98] M. Marseguerra, E. Zio, J. Devooght, P.E. Labeau, A concept paper on dynamic reliability via Monte Carlo simulation, Mathematics and Computers in Simulation 47, 1998, pp. 371-382.

[Mel 02] R. E. Melchers, M. Ahammed, Gradient estimation for applied Monte Carlo analysis, Reliability Engineering and System Safety 78, 2002, pp. 283-288.

[Meu 00] P. Meunier, B. Denis and J.J. Lesage, Safety Analysis During the Control Architecture Design of Automated Systems, Safeprocess 2000 (IFAC), Budapest, Hungary, June 14-16, 2000.

[Mih 02] A. Mihalache, A. Todoskoff, M. Barreau, F. Guerin, J.-Y. Morel, I. C. Bacivarov, Reliability Analysis for Mechatronic Systems. Application on ABS, 8th International Conference Quality, Reliability, and Maintainability, CCF'02, Sinaia, Roumanie, 18-20 sept 2002.

[Mon 98] G. Moncelet, Application des Réseaux de Petri à l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1998.

[Mon 02] G. Moncelet, A. Texier, N. Raimarckers, New concepts for a dependable design of aeronautic engine lubrification system, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Mon 02a] L. A. Montestruque, P. J. Antsaklis, State and Output Feedback Control in Model-Based Networked Control Systems, Proc. of the 41st IEEE Conf. on Decision and Control, Las Vegas, Nevada, Dec. 10 – 13, 2002, pp. 1620-1625.

[Mor 01] G. Morel, B. Iung, M.-C. Suhner, J.-B. Léger, Maintenance Holistic Framework for Optimising the Cost/Availability compromise of Manufacturing Systems, IFAC Cost Oriented Automation, Berlin, Germany, 8-9 October 2001, pp. 7-19.

[Mor 02] J.-Y. Morel, M. Barreau, A. Todoskoff, Petri Nets : a tool adapted to computer system dependability and safety, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[NF 50295] Norme française, Norme européenne, NF EN 50295, Appareillage à basse tension, Systèmes d'interface appareil de commande – appareils. Interface capteur – actionneur (AS-i), mars 1999.

[Ota 02] P. G. Otanez, J. R. Moyne, D. M. Tilbury, Using Deadbands to Reduce Communication in Networked Control Systems, 2002 American Control Conference, May 2002.

[Ota 02a] P. G. Otanez, J. T. Parrott, J. R. Moyne, D. M. Tilbury, The Implications of Ethernet as a Control Network, Global Powertrain Conference, Ann Arbor, MI, September 2002.

[Pag 80] A. Pagès, M. Gondran, Fiabilité des systèmes, Editions Eyrolles, Paris, 1980.

[Pap 01] Y. Papadopoulos, J. McDermid, R. Sasse, G. Heiner, Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure, Reliability Engineering and System Safety 71, 2001, pp. 229-247.

[Pat 03] R. P. Patankar, A model for fault-tolerant networked control system using TTP/c communication, ACC 2003, accepted.

[Raj 00] D. V. Raje, R. S. Olaniya, P. D. Wakhare, A. W. Deshpande, Availability assessment of a two-unit stand-by pumping system, Reliability Engineering and System Safety 68, 2000, pp. 269-274.

[Ram 02] A. V. Ramesh, D. W. Twigg, U. R. Sandadi, T. C. Sharma, Reliability analysis of systems with operation-time management, IEEE Transactions on reliability, Vol. 51, No. 1, March 2002.

[Rie 02] P. Riedinger, C. Iung, J. Daafouz, Commande optimale des systèmes dynamiques hybrides : Théorie et pratique. Conférence Internationale Francophone d'Automatique, CIFA 2002, Nantes, 8 - 10 juillet 2002.

[Roc 02] C.M. Rocco, J.A. Moreno, Network reliability assessment using a cellular automata approach, Reliability Engineering and System Safety 78, 20002, pp. 289-295.

[Scilab] http://www-rocq.inria.fr/Scilab/

[Seb 03] Q. Sébastien, Evaluation du niveau de sécurité d'une architecture distribuée à l'aide d'une modélisation par réseaux de Petri, Rapport de DEA, Ecole Nationale Supérieure d'Electricité et de Mécanique, INPL de Lorraine, 2003.

[Sig 02] J.-P. Signoret, J.-L. Chabot, T. Hutinet, Comment cacher un réseau de Petri derrière un bloc diagramme de fiabilité, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Smi 98] C. Smidts, J. Devooght, P. E. Labeau, Dynamic reliability: Future Directions, International Workshop Series on Advanced Topics in Reliability and Risk Analysis, Greenbelt, Maryland, USA, septembre 1998.

[Son 03] H. S. Son, P.H. Seong, Development of a safety critical software requirements verification method with combined CPN and PVS: a nuclear power plant protection system application, Reliability Engineering and System Safety 80, 2003, pp. 19–32.

[Thi 03] J.-M. Thiriet, P. Barger, M. Robert, Dynamic evaluation of dependability for distributed intelligence automation systems, 4th International Conference on Industrial Automation, École de technologie supérieure, Montréal, June 9-11 2003.

[Tuf 02] B. Tuffin, S. Collas, Creation of a dynamic model language and application of Monte Carlo methods for the reliability survey of industrial complex systems, Lambda-Mu 13 – Esrel 2002 European conference, Lyon 2002.

[Ull 98] J. D. Ullman, Elements of ML programming – ML97 edition, Prentice Hall, New Jersey, 1998.

[Vid 92] G. Vidal-Naquet, A. Chowuet-Geniet, Réseaux de Petri et systèmes parallèles, Armand Colin, Paris, 1992.

[Vil 88] A. Villemeur, Sûreté de fonctionnement des systèmes industriels, Editions Eyrolles, Paris, 1988.

[Wal 99] G. C. Walsh, H. Ye, L. Bushnell, Stability Analysis of Networked Control Systems, Proceedings of American Control Conference, June 1999.

[Wal 01] G. C. Walsh, H. Ye, Scheduling of Networked Control Systems, IEEE Control Systems Magazine, February 2001, pp. 57-65.

[Wan 02] Yi-Feng Wang, Jer-Yu Wu, Chuei-Tin Chang, Automatic hazard analysis of batch operations with Petri Nets, Reliability Engineering and System Safety 76, 2002, pp. 91-104.

[Web 01] P. Weber, M.-C. Suhner, B. Iung, System Approach-based Bayesian Network to Aid Maintenance of Manufacturing Process, IFAC Cost Oriented Automation, Berlin, Germany, 8-9 October 2001, pp. 33-39.

[Web 03] P. Weber, L. Jouffe, Reliability modelling with dynamic Bayesian networks, IFAC-Safeprocess 2003, Washington (USA), June 2003.

[Zha 01] W. Zhang, M. S. Branicky, S. M. Phillips, Stability of Networked Control systems, IEEE Control Systems Magazine, February 2001, pp. 84-99.

Annexes

Annexe A : Sûreté de fonctionnement

A.1 EVALUATION DE LA SURETE DE FONCTIONNEMENT

A.1.1 Définitions de base

Fiabilité [Vil 88] : C'est l'aptitude d'une entité à accomplir une fonction requise dans des conditions données, pendant une durée donnée, mathématiquement :

$$R(t)=P[\text{E non défaillante sur } [0,t]]$$

Défaut [Vil 88] : Tout écart entre une caractéristique d'une entité et la caractéristique voulue dépassant des limites d'acceptabilité dans les conditions données. Autrement dit, un défaut est une non-conformité à des objectifs ou des clauses de spécifications.

Défaillance [Vil 88] : Cessation de l'aptitude de l'entité à accomplir une fonction requise, qui est liée à un écart entre la caractéristique réelle et la caractéristique espérée de la fonction.

Panne [Vil 88] : Inaptitude d'une entité à accomplir une fonction requise.

Après apparition d'une défaillance, on considère donc que l'entité est en panne ; une panne résulte toujours d'une défaillance. Les pannes peuvent être de natures différentes :

- Panne intermittente : Panne d'une entité subsistant pendant une durée limitée après laquelle l'entité redevient apte à accomplir une fonction requise sans avoir été soumise à une opération de maintenance corrective.
- Panne fugitive : Panne d'une entité qui est intermittente et difficilement constatable.
- Panne permanente : Panne d'une entité qui persiste tant que n'ont pas eu lieu des opérations de maintenance corrective.
- Panne latente : Panne qui existe mais qui n'a pas encore été détectée.

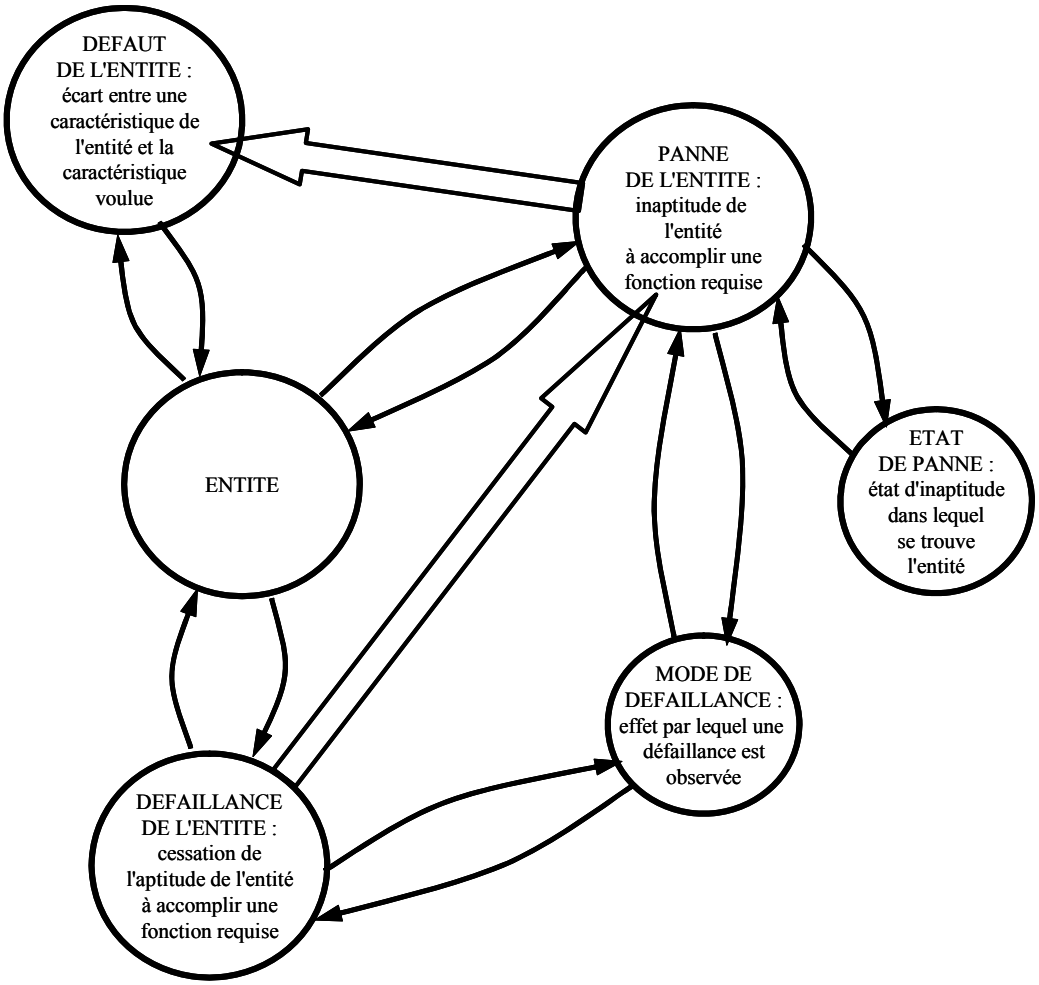


Figure A.1 : Défaut, défaillance et panne [Vil 88]

Un point important à définir ici est la cohérence des systèmes du point de vue de la sûreté de fonctionnement. D'une manière générale, un système est dit cohérent si [Vil 88] :

- la panne de tous les composants entraîne la panne du système,
- le fonctionnement de tous les composants entraîne le fonctionnement du système,
- lorsque le système est en panne, aucune défaillance supplémentaire ne rétablit le fonctionnement du système,
- lorsque le système est en fonctionnement, aucune réparation n'induit la panne du système.

A.1.2 Evaluation des systèmes

Les méthodes d'analyse de sûreté de fonctionnement travaillent pour la plupart par la décomposition des grands systèmes en composants individuels dont les caractéristiques sont supposées connues. Les

systèmes considérés peuvent être construits à partir de leurs composants grâce à des branchements logiques en série, en parallèle, en redondance ou par une combinaison des précédents [Dod 99].

A.1.2.1 Système série

Pour qu'un système en série (figure A.2) fonctionne correctement, tous les composants doivent fonctionner correctement.

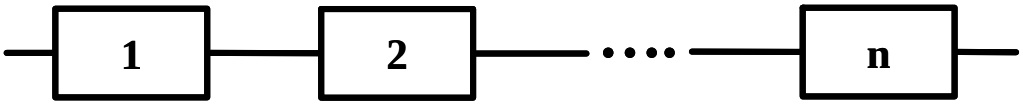


Figure A.2 : Diagramme bloc d'un système en série [Dod 99]

Si chaque composant est indépendant, la fiabilité d'une architecture série est :

$$R_s = \prod_{i=1}^{n} R_i$$

Si les défaillances des composants observent une loi exponentielle, le temps moyen avant la première défaillance du système est :

$$MTTF_s = \frac{1}{\sum_{i=1}^{n} \lambda_i}$$

où λ_i est le taux de défaillance du composant i.

A.1.2.2 Système parallèle

Pour qu'un système en parallèle (figure A.3) fonctionne correctement, au moins un des composants doit fonctionner correctement.

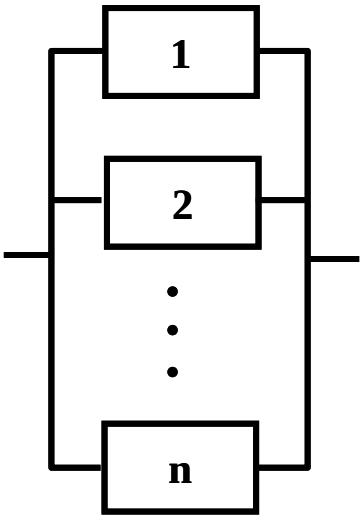


Figure A.3 : Diagramme bloc d'un système parallèle [Dod 99]

La classification des systèmes parallèles est la suivante :

- systèmes actifs (tous les systèmes marchent en parallèle, *n-1* composants en redondance)
  - purs (la défaillance d'un composant n'affecte pas les taux de défaillances des autres)
  - partagés (le taux de défaillance évolue en fonction de l'état des autres composants)
- systèmes passifs (les composants redondants démarrent à la défaillance du composant en marche).

La fiabilité d'un système parallèle pur est :

$$R_s = 1 - \prod_{i=1}^{n}(1 - R_i)$$

Si le système parallèle est constitué de *n* composants identiques avec un taux de défaillance exponentiel le temps moyen entre les défaillances est :

$$MTBF_s = \sum_{i=1}^{n} \frac{1}{i\lambda}$$

où λ est le taux de défaillance des composants.

A.1.2.3 Système r sur n

Ce système est une modification des systèmes parallèles. Pour qu'il fonctionne correctement, il est nécessaire que *r* composants sur *n* au total fonctionnent correctement. Son taux de fiabilité est :

$$R_s = \sum_{x=r}^{n} C_n^x R^x (1-R)^{n-x}$$

et pour le cas ou les taux de défaillance des composants sont constants, le temps moyen avant la première défaillance est :

$$\theta_s = \sum_{i=r}^{n} \frac{1}{i\lambda}$$

où λ est le taux de défaillance des composants.

A.1.2.4 Système redondant passif

Un composant unique est actif à l'instant donné dans les systèmes redondants passifs (figure A.4). Un autre composant est mis en marche en cas de défaillance du premier. Cette architecture nécessite donc l'utilisation d'un commutateur.

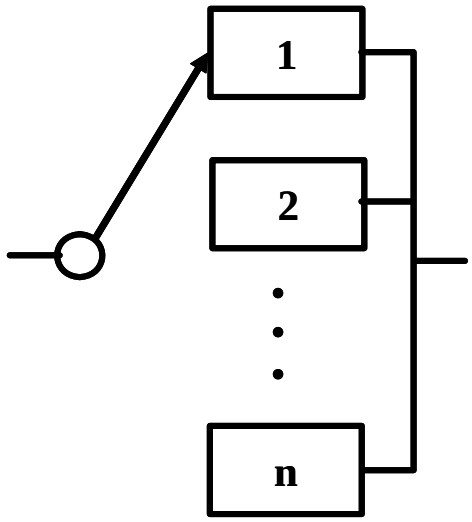


Figure A.4 : Diagramme bloc d'un système parallèle [Dod 99]

Les expressions de la fiabilité pour de tels systèmes dépendent des caractéristiques des composants, mais également du commutateur et de l'évolution du taux de défaillance en présence de défaillances d'autres composants. Pour le cas le plus simple avec un commutateur non-défaillant et égalité de tous

les taux de défaillance avec une distribution exponentielle constante λ dans la présence de n composants, la fonction de fiabilité est :

$$R(x) = e^{-\lambda x} \sum_{i=0}^{n-1} \frac{(\lambda x)^i}{i!}$$

A.2 METHODES D'EVALUATION

De nombreuses méthodes existent. [Dod 99] propose 4 méthodes pour l'étude de la sûreté de fonctionnement :

- Arbres des causes,
- Simulation de Monte Carlo,
- Analyse de Markov,
- Allocation de fiabilité.

[Vil 88] en propose 9 :

- Analyse Préliminaire des Dangers (APD),
- Analyse des Modes de Défaillance et de leurs Effets (AMDE),
- Méthode du Diagramme de Succès (MDS),
- Méthode de l'Arbre des Causes (MAC),
- Méthode de la Table de Vérité (MTV),
- Méthode des Combinaisons de Pannes Résumées (MCPR),
- Méthode de l'Arbre des Conséquences (MACQ),
- Méthode du Diagramme Causes-Conséquences (MDCC),
- Méthode de l'Espace d'Etats (MEE).

D'autres méthodes existent mais sont actuellement relativement minoritaires. Il s'agit par exemple de méthodes basées sur les réseaux bayésiens [Web 03], les réseaux de Petri [Sig 02], [Châ 98], les réseaux de neurones [Châ 98a].

A.2.1 Arbres des causes

Il s'agit d'une méthode parmi les plus classiques de la SdF. Elle a les objectifs suivants [Vil 88] :

- déterminer les diverses combinaisons possibles d'événements qui entraînent la réalisation d'un événement indésirable unique ;
- représenter graphiquement ces combinaisons au moyen d'une structure arborescente.

Un arbre des causes est composé de portes et d'événements. Les portes peuvent être classées en deux groupes : 1. des portes de base (ET et OU (avec ou sans condition), OU EXCLUSIF) et 2. des portes étendues : SI, COMBINAISON de *m* sur *n,* DELAI, MATRICIELLE, QUANTIFICATION, SOMMATION, COMPARAISON.

Les événements peuvent être :

- événement primaire ou élémentaire,
- événement conditionnel qui est le résultat de la combinaison d'autres événements,
- événement 'maison' équivalent du commutateur on/off,
- événement 'losange' qui ne peut être considéré comme élémentaire mais dont les causes ne sont pas et ne seront pas développées,
- événement conditionnel,
- événement 'triangle' servant pour continuer l'arbre à l'autre page de la spécification.

L'analyse d'arbre des causes doit rechercher, à partir d'un événement indésirable, les pannes ou les défaillances de composants dont la combinaison entraîne l'apparition de l'événement indésirable. L'analyse des causes doit être faite jusqu'à ce que l'on obtienne des événements de base.

L'analyse d'un arbre des causes se fait à l'aide des coupes minimales et des implicants premiers. Une coupe minimale [Vil 88] est la plus petite combinaison d'événements entraînant l'événement indésirable. Les implicants premiers, utilisés dans l'analyse des systèmes non cohérents, sont définis comme étant des produits d'événements de base ou de leurs complémentaires ne contenant aucun autre implicant.

A.2.2 Analyse de Markov ou Méthode de l'espace d'état

Il s'agit de la méthode peut-être la plus connue dans le domaine de la sûreté de fonctionnement. Comme son nom l'indique, elle peut être appliquée aux systèmes qui peuvent être définis comme un ensemble d'états.

Les états sont classés dans deux catégories [Vil 88] :

- des états de fonctionnement : ce sont les états où la fonction du système est réalisée, des composants du système pouvant être en panne ; l'état de bon fonctionnement est l'état où aucun composant n'est en panne ;
- des états de panne : ce sont les états où la fonction du système n'est plus réalisée, un ou plusieurs composants du système étant en panne.

Comme la description détaillée est donnée dans plusieurs ouvrages de référence dont [Vill 88] et [Dod 99], elle n'est pas présentée ici. L'accent sera mis sur les problèmes liés à l'application de cette méthode sur l'évaluation des systèmes SAR.

Selon [Vil 88] le processus d'analyse comprendra 3 parties :

a.) *Le recensement et le classement de tous les états du système en états de bon fonctionnement ou en états de panne.* A priori, il est pratiquement impossible d'estimer tous les états d'un système SAR. Donc cette étape ne peut être accomplie que partiellement.

b.) *Le recensement de toutes les transitions possibles entre ces différents états et l'identification de toutes les causes de ces transitions.* Comme l'étape précédente était difficilement faisable, cette étape l'est aussi. En effet, les transitions entre les états sont en général dynamiques dans un SAR. De plus, leurs lois d'évolution sont difficiles à exprimer.

c.) *Le calcul des probabilités de se trouver dans les différents états au cours d'une période de vie du système ou le calcul de caractéristiques de sûreté de fonctionnement.* Sous la condition de réussite des étapes précédentes, cette évaluation reste possible.

Annexe B : Bases des réseaux de Petri

B.1 DEFINITIONS DE BASE

Il y a différentes façons de définir les Réseaux de Petri (RdP dans la suite). [Dav 97] utilise l'appellation de RdP ordinaires. Il s'agit du même concept qui est dans [Dia 01] nommé RdP places – transitions et généralement connu sous le nom de RdP (tout court) [Vid 92].

Définition 1 [Dav 97] **:** Un RdP ordinaire non marqué est un quadruplet Q= <P, T, Pré, Post> tel que :
$P = \{P_1, P_2, \ldots, P_n\}$ est un ensemble fini et non vide de places ;
$T = \{T_1, T_2, \ldots, T_n\}$ est un ensemble fini et non vide de transitions ;
P ^ T = O, c'est-à-dire que les ensembles P et T sont disjoints ;
Pré : P x T → {0, 1} est l'application d'incidence avant ;
Post : P x T → {0, 1} est l'application d'incidence arrière.

Un marquage initial est une distribution des marques (jetons) dans l'ensemble des places P à l'instant 0.

Définition 2 [Dav 97] **:** Un RdP marqué est un doublet R = <Q, M_0> dans lequel Q est un RdP non marqué et M_0 un marquage initial.

Les définitions analogues se trouvent aussi dans [Dia 01].

La présentation graphique des RdP est une autre forme de définition. Les places sont représentées par des cercles, les transitions par des rectangles, les applications d'incidence avant et arrière par des arcs orientés et les marques par des petits cercles pleins (voire figure B.1).

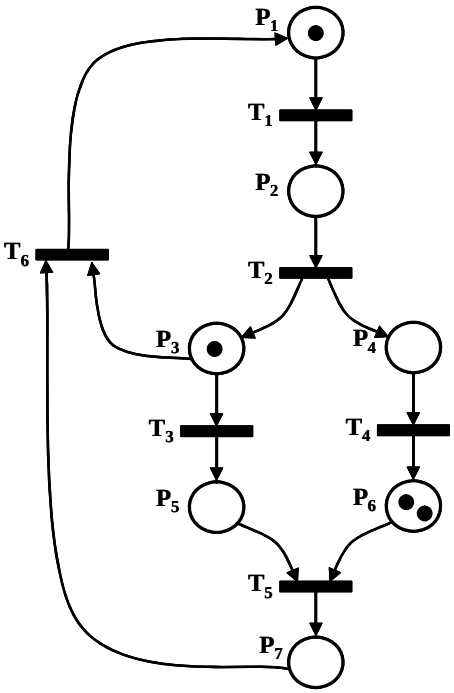


Figure B.1 : Réseaux de Petri marqué [Dav 97]

Définition 3 [Dia 01] **:** Pour un marquage m, une transition t est sensibilisée (et, pour les réseaux de Petri places-transitions, tirable ou franchissable) si et seulement si, pour tout $P_i \in P$, on a $m(P_i) \geq$ Pré(pi,t).

Définition 4 [Dia 01] **:** Dans un réseau de Petri Places-transitions, toute transition tirable t peut être tirée (ou franchie) et son tir (ou franchissement) conduit à un nouveau marquage m' défini, pour tout p, par :

m'(p) = m(p) – Pré(p,t) + Post(p,t)

Le franchissement ou tir d'une transition t sera noté m[t>m' ou m – t → m'.

B.2 METHODES D'ANALYSE

La construction du modèle d'un système permet sa meilleure compréhension. Mais les modèles formels proposent en plus différentes possibilités d'analyse. Un grand avantage des techniques formelles est le potentiel d'automatisation de leur analyse par des algorithmes.

Un des intérêts majeurs des modèles formels est la possibilité de définir sans ambiguïté le comportement d'un système, de développer des algorithmes de vérification des propriétés et de les intégrer dans un outil logiciel [Dia 01].

Pour les RdP trois moyens d'analyse sont particulièrement bien développés :

- la simulation,
- le graphe de marquage,
- l'algèbre linéaire (analyse des invariants).

Parmi d'autres méthodes d'analyse se trouvent la réduction, la décomposition, l'analyse directe du RdP [Kha 02] ou encore la conception en respectant certaines contraintes pour garantir que le modèle final possède des caractéristiques exigées [Mon 02].

B.2.1 Simulation

La simulation est une méthode efficace d'analyse des RdP. Dès la phase de conception des modèles, elle permet de vérifier le fonctionnement du modèle pas-à-pas. Ainsi elle contribue à une meilleure compréhension du modèle et facilite son développement.
Après avoir complété le modèle, la simulation devient un moyen d'analyse de son comportement. La visualisation de l'évolution d'états du modèle peut être observée. On peut ainsi percevoir la quasi globalité du comportement du modèle.

La dernière méthode décrite ici pour l'analyse des modèles RdP par simulation est la méthode de Monte-Carlo.

B.2.2 Graphe de marquage

Définition [Dia 01] **:** L'ensemble des marques accessibles ou ensemble d'accessibilité, $A(R,m_0)$ ou A, se définit par le plus petit ensemble :
$M_0 \in A$ et, si $m \in A$ et $m[t>m'$ alors $m' \in A$

Définition [Dia 01] **:** Le graphe des marquages accessibles ($G(R, m_0)$ (ou graphe d'accessibilité) est défini comme le graphe dons les nœuds sont les marquages accessibles de $A(R,m_0)$ et dont les arcs, étiquetés par les noms des transitions, sont définis par la relation de tir entre les marquages. Pour m, m' $\in$ A, on a $m - t \rightarrow m' \in G$ si est seulement si $m[t>m'$.

L'analyse par un graphe de marquage nécessite la connaissance d'un marquage initial du modèle ce qui peut poser un problème dans certains cas.

L'analyse par parcours d'ensemble ou d'une partie du graphe de marquage appartient aux méthodes comportementales. A cause des problèmes liés à la taille du graphe de marquage, des modifications de ces méthodes existent. Elles sont basées sur la prise en considération de la structure du RdP initial. De telles méthodes sont appelés structurelles.

[Chr 01] présente une méthode implémentée dans l'outil Design/CPN qui permet sensiblement de diminuer le coût temporel du calcul d'un graphe d'occurrence. Il s'agit d'effacer les parties déjà explorées pour libérer de l'espace mémoire. La condition d'effacement est l'exploration totale de la partie correspondante et la mémorisation des résultats significatifs.

B.2.3 Algèbre linéaire

L'approche d'analyse des RdP par l'algèbre linéaire est basée sur l'équation fondamentale :
Mk= Mi+ W*S
c'est-à-dire que le marquage Mk s'exprime comme la somme du marquage initial Mi et du produit de la matrice d'incidence par le vecteur des occurrences de transition dans la séquence de franchissements.

L'étude de la matrice d'incidence W est utilisée pour la détermination des composantes conservatives et répétitives.

La composante conservative P(F) (autrement appelé support) est un sous-ensemble de l'ensemble des places P, tel que la somme des marquages dans les places P(F) reste identique pour toute séquence de franchissement.

Propriété [Dav 97] **:** Soit D un ensemble de transitions. L'ensemble D est une composante répétitive (stationnaire si et seulement s'il existe une séquence de franchissements S telle que T(S)=D et W*S=0).

L'idée d'analyse des RdP colorés par l'algèbre linéaire et plus particulièrement par des matrices symboliques est également en partie possible. Elle a été développée par H. Genrich. Néanmoins son implantation dans des outils informatiques se trouve au stade de prototypes et n'est pas actuellement intégré dans les logiciels les plus répandus.

Une composante répétitive est une séquence des franchissements sur un sous-ensemble de l'ensemble des transitions qui à partir d'un marquage Mi va à travers d'autres marquage revenir au marquage Mi. Cette propriété est utilisée pour la détermination des comportements cycliques.

B.2.4 Méthodes de réduction

Les méthodes d'analyse des RdP par la réduction [Dav 97] profitent de la propriété qu'il est possible sous certaines contraintes d'enlever ou de substituer certaines places ou des groupes de places tout en gardant la propriété étudiée. Mais il faut faire grande attention car les règles de réduction peuvent être différentes d'une propriété à l'autre. Les propriétés les plus souvent étudiées sont la vivacité et la bornitude ainsi que les invariants de marquage.

En général, quatre règles sont utilisées :

- substitution d'une place,
- place implicite,
- transition neutre,
- transitions identiques.

B.3 PROPRIETES DE BASE

Dans la suite, nous proposons quelques définitions de base liées à l'analyse des RdP par un graphe de marquage ou par l'algèbre linéaire à l'aide de calcul des invariants.

Les définitions suivantes sont prises dans [Dav 97] et dans [Dia 01].

Une **Place** P_i est dite **bornée** pour un marquage initial M_0 s'il existe un entier naturel k, tel que pour tout marquage accessible à partir de M_0, le nombre de marques dans P_i est inférieur ou égal à k (on dit que P_i est k-bornée).

Un **RdP est borné** pour un marquage initial M_0 si toutes les places sont bornées pour M_0 (le réseau est k-borné si toutes les places sont k-bornées).

Définition : Un RdP est dit **sauf** (ou binaire) pour un marquage initial M_0 si pour tout marquage accessible, chaque place contient au plus une marque.

Définition (pseudo-vivacité) : Un réseau de Petri (R, m_0) est dit **pseudo-vivant** si :

$$\forall m \in A(R, m_0),\ \exists t \in T \quad t.q.\ m\left[t\right\rangle$$

C'est-à-dire que pour tout marquage faisant partie du graphe de marquage, c'est-à-dire tout marquage atteignable, il existe au moins une transition qui peut être franchie.

Définition (quasi-vivacité) : Un réseau de Petri (R, m_0) est **quasi-vivant** si :

$$\forall t \in T,\ \exists m \in A(R, m_0) \quad t.q.\ m\left[t\right\rangle$$

Ce qui veut dire que pour chaque transition appartenant au RdP il existe au moins un marquage, tel que la transition en question est franchissable.

Définition (vivacité) : Un réseau de Petri (R, m_0) est vivant si, pour tout marquage $m \in A(R, m_0)$, le réseau (R,m_0) est quasi vivant. Autrement dit :

$$\forall m \in A(R, m_0), \forall t \in T, \exists m' \in A(R, m) \ \ t.q.\ m'\left[t\right\rangle$$

C'est-à-dire que pour tout marquage accessible du marquage initial et pour toute transition, il existe un nouveau marquage appartenant au graphe de marquage telle que toute transition appartenant au RdP est franchissable.

Définition : Un **blocage** (ou état puits) est un marquage tel qu'aucune transition n'est validée (franchissable).

Définition : Un RdP est dit **sans blocage** pour un marquage initial M_0 si aucun marquage accessible M_i à partir de M_0 n'est un blocage.

Définition :

a) Un RdP a un état d'**accueil** M_a pour un marquage initial M_0 si pour tout marquage accessible M_i M_0, il existe Si tel que M_i $[S_i>M_a$.

b) Un RdP est **réinitialisable** pour un marquage initial M_0 si M_0 est un état d'accueil.

Définition : Un **conflit effectif** est l'existence d'un conflit structurel, K, et d'un marquage, M, tel que le nombre de maques dans P_i est inférieur au nombre de transitions de sortie de P_i qui sont validées par M.

Définition : Un RdP est **sans conflit** effectif pour un marquage initial M_0, si pour tout marquage accessible M M_0 il n'a pas de conflit effectif.

Définition : Un RdP est **persistant** pour un marquage initial M_0 si pour tout marquage accessible Mi M0, on a la propriété suivante : si T_j et T_k son validées par le marquage M_i, alors T_jT_k est une séquence de franchissements à partir de M_i (ainsi que T_kT_j par symétrie).

Définition : Soit $\sigma \in T^*$ une **séquence de transitions**. Son vecteur d'occurrence $\vec{\sigma} \in \square^T$ est défini par : $\vec{\sigma}(t)$ est le nombre d'occurrences de t dans σ.

Proposition (Equation de changement d'état, équation fondamentale) : Soit R un réseau de Petri et m[σ>m' une séquence de franchissement. On a alors :

$$m' = m + C * \vec{\sigma}.$$

Remarque : *C* est la matrice d'incidence du RdP.

Définition (Flots d'un réseau) : Les différents annulateurs que nous considérons sont les suivants :
- Un P-flot est un vecteur non nul v de ZP qui vérifie vt.C=0.
- Un P-semiflot est un vecteur non nul v de NP qui vérifie vt.C=0.
- Un T-flot est un vecteur non nul v de ZT qui vérifie C.v=0.
- Un T-semiflot est un vecteur non nul v de NT qui vérifie C.v=0.

Les semiflots sont utilisables pour déterminer par exemple des composantes répétitives (stationnaires), notamment par le calcul algébrique.

B.4 EXTENSIONS DES RESEAUX DE PETRI

Le pouvoir d'expression des RdP classiques reste limité et par conséquent leur utilisation trop compliquée voir impossible si par exemple on a besoin d'un arc inhibiteur. Pour remédier à ce problème de nombreuses extensions ont été proposées dans la littérature. [Dia 01] présente une synthèse que quelques extensions principales d'abord selon l'ordre chronologique (figure 3.2) puis selon leur sémantique (figure B.3).

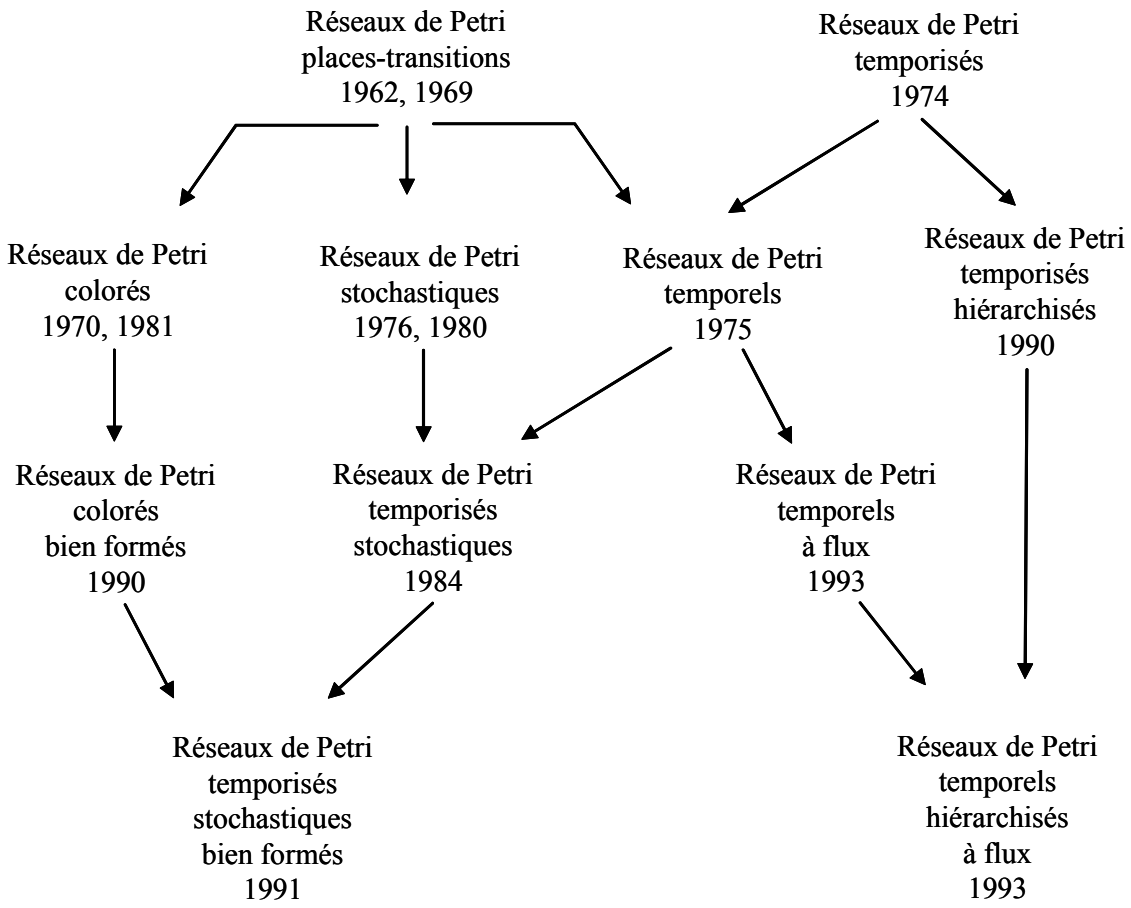


Figure B.2 : Différentes familles de réseaux de Petri [Dia 01]

Le classement de ces extensions n'est pas une chose aisée. En effet, les extensions se combinent et croisent souvent entre elles et créent ainsi des nouvelles classes. Un très bon exemple de ce phénomène se trouve sur la figure B.3 où les RdP colorés se trouve aussi bien à gauche qu'à droite du diagramme. A gauche en tant que des RdP colorés et à droite en tant que RdP colorés stochastiques.

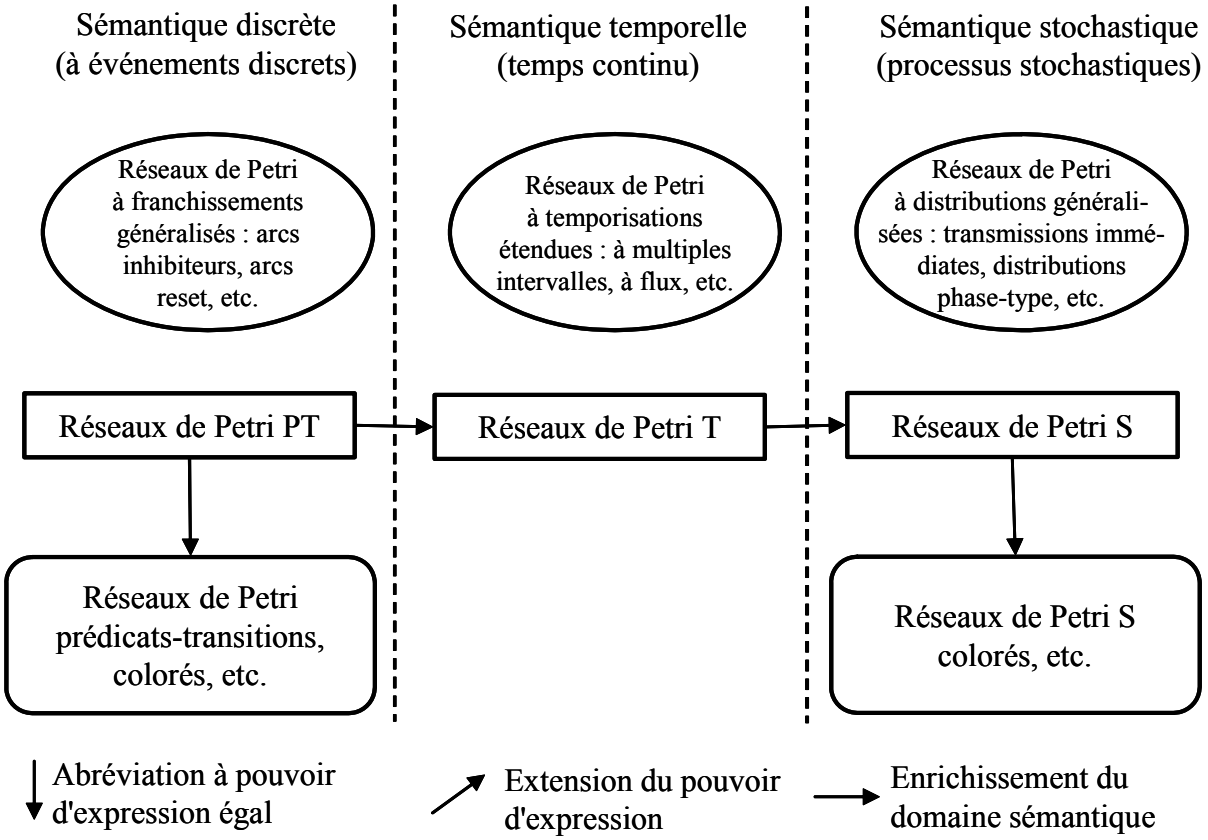


Figure B.3 : Les domaines sémantiques des réseaux de Petri [Dia 01]

Nous n'allons que brièvement mentionner quelques types de RdP bien connus et puis enchaîner directement avec les RdP colorés tels qu'ils sont implémentés dans l'outil Design/CPN en justifiant tout d'abord ce choix.

B.4.1 Réseaux de Petri généralisés

Les RdP généralisés sont une extension permettant d'associer "un poids" aux arcs. (Ce poids indique le nombre de jetons qui doivent passer par l'arc lors du franchissement de transitions correspondantes.) Le poids implicite est fixé à 1. Pour franchir une transition d'un RdP généralisé, il est nécessaire que toutes les places P_i en amont de la transition T_j contiennent au moins le nombre de jetons associés aux arcs (Pré(P_i,T_j)). Après le franchissement, les nombres de jetons dans les places en aval de la transition T_j sont augmentés par Post($_{Pi}$,T_j).

Si auparavant la matrice d'incidence est composée de nombre entiers –1, 0, 1, dans le cas de RdP généralisé elle peut contenir tout nombre entier –n, …, -2, -1, 0, 1, 2, …, n.

Propriété : Tout réseau de Petri généralisé peut être transformé en RdP ordinaire.

Cette extension des RdP permet dans certains cas de réduire d'une manière significative la taille du RdP ordinaire correspondant. Ainsi elle augmente le potentiel d'utilisation des RdP.

B.4.2 Réseaux de Petri à capacité

Dans le cas précédent un attribut est ajouté aux arcs de RdP. Les RdP à capacité associent aux places de RdP la capacité, c'est-à-dire les limites de nombres de jetons qui peuvent être contenus dans une place. Pour que cette condition soit applicable, la condition de capacité doit être précisée explicitement.

Cette extension modifie les conditions de franchissement des transitions. Pour qu'un franchissement soit possible, toutes les place à capacité en aval de la transition doivent contenir un nombre de jetons strictement inférieur aux capacités mentionnées. Dans le cas d'une combinaison des RdP à capacité et RdP généralisés, cette différence doit être supérieure ou égale à Post ($_{Pi}$, T_j).

Propriété : Tout réseau de Petri à capacité peut être transformé en RdP ordinaire.

B.4.3 Réseaux de Petri à prédicats

Selon [Dav 97] les RdP à prédicats comportent des marques auxquelles on attribue des paramètres. Les arcs portent les étiquettes et les prédicats sont associés aux transitions. Le prédicat peut pendant le franchissement de la transition modifier la valeur des marques utilisées pour le franchissement.

Il s'agit d'un rapprochement aux RdP colorés, présentés dans la suite. En effet, les RdP à prédicats peuvent être considérés comme une sous-classe des RdP colorés.

B.4.4 D'autres extensions

Avec les quelques exemples d'extensions nous sommes loin d'avoir fait le tour du monde RdP. Des nombreuses autres extensions existent, mais restent souvent aux utilisations bien particulières. Les RdP récursifs [Had 99] en sont un exemple. Ils sont adaptés pour la modélisation des systèmes multi-agents et combinent les RdP avec les langages sans contexte.