Modélisation de la propagation des fautes dans les systèmes de production

Timothée Kombé

N° d'ordre 2011ISAL0057 Année 2011

THÈSE

Modélisation de la propagation des fautes dans les systèmes de production

Présentée devant
L'Institut National des Sciences Appliquées de Lyon
et
L'Ecole Nationale Supérieure Polytechnique de Yaoundé

Pour obtenir

Le grade de Docteur/PhD

École doctorale : Électronique, Électrotechnique, Automatique
Spécialité : Automatique Industrielle

Par

Timothée KOMBE

Soutenue le 30 juin 2011 devant la commission d'examen

Jury

Adolphe MOUKENGUE IMANO	Professeur à l'IUT Université Douala (Cameroun)	Rapporteur
Bernard GRABOT	Professeur à l'ENIT de PAU (France)	Rapporteur
Pierre ELE	Professeur à l'ENSP Université Yaoundé I (Cameroun)	Examinateur
Antoine RAUZY	Chercheur au LIX, Polytechnique (France)	Examinateur
Eric NIEL	Professeur à l'INSA de Lyon (France)	Directeur
Charles AWONO ONANA	Professeur à l'ENSP Université Yaoundé I (Cameroun)	Directeur

Laboratoire Ampère de l'INSA de Lyon
et
Laboratoire de Matériaux et de Mécanique de l'ENSP de Yaoundé

À Carol Marcelle

mon l'autre moi

Remerciements

Je tiens à remercier Monsieur AWONO ONANA, Professeur à l'Ecole Nationale Supérieure Polytechnique de l'Université de Yaoundé I au Cameroun pour son soutien pendant toute cette thèse. Je n'ai pas assez de mots pour dire Merci à Monsieur Eric NIEL, Professeur au laboratoire Ampère de l'INSA de Lyon, pour m'avoir accepté, accueilli et intégré dans son équipe de recherche "Actionneurs et Systèmes"" et grâce à qui j'ai pu obtenir deux subventions de la bourse MIRA (Mobilité Internationale Rhône Alpes) grâce à laquelle mon séjour en France a été possible. Sa disponibilité, son humilité, ses remarques, ses conseils, ses orientations m'ont mis en confiance et ont permis d'aboutir à la rédaction de cette thèse. Je tiens à remercier du fond de mon cœur Monsieur Antoine RAUZY de Dassault Systems, pour m'avoir gracieusement offert le logiciel de simulation AltaRica Data-Flow grâce auquel mes modélisations et simulations ont été possibles. Ses conseils et ses remarques m'ont été d'un très grand apport. Je remercie Monsieur Benoit NDZANA Directeur du Laboratoire de Matériaux et de Mécanique de l'Ecole Nationale Supérieure Polytechnique de l'université de Yaoundé I au Cameroun pour m'avoir accepté dans ce laboratoire et pour son soutien. Je remercie Monsieur Louis Max AYINA OHANDJA Directeur de l'Institut Universitaire de Technologie de l'Université de Douala au Cameroun, pour m'avoir donné l'opportunité d'aller m'inscrire en thèse à l'INSA de Lyon, grâce à l'octroi des bourses de formation des formateurs.

J'exprime ma reconnaissance envers tous les membres de ce jury qui, nonobstant leurs multiples occupations, m'ont fait honneur en acceptant de venir juger les travaux de cette thèse.

Je remercie tous les membres du laboratoire Ampère de l'INSA de Lyon pour m'avoir facilité l'intégration dans cette équipe. Monsieur Laurent PIETRAC, pour ses remarques et ses critiques, je fais un clin d'œil à Monsieur Emil DUMITRESCU, pour ses sympathiques remarques, Messieurs Grégory FARAULT, Mingming REN et Gabor KOVACS, tous Doctorants dans l'équipe "Actionneurs et Systèmes", pour la bonne ambiance qui avait toujours régné dans le groupe.

Je remercie tous ceux qui de près ou de loin ont contribué à l'élaboration de ce travail. Monsieur Jean Pierre NGUELE AYONG et son épouse Cécile, Emmanuel ESSOUONG et son épouse Sandrine, Timothée BITOUNOU et son épouse Nicole, Françoise ETONG et son époux Eric, tous membres de la famille résidant en France, pour leur hospitalité.

Je ne saurais terminer sans dire grandement Merci à mes enfants, Fabrice KOMBE NANGA, Nastia KOMBE BELOMBE, Thierry Henri KOMBE et Cédric KOMBE, pour leur compréhension. Ils ont su accepté que je les abandonne à leur maman seule, pour des séjours assez longs, afin de commencer et de terminer ces travaux de thèse. Je remercie mon grand frère Monsieur Engelbert NGOULA et ma petite sœur Anastasie NANGA pour leurs encouragements.

Table des matières

Table des figures

Table des tableaux

Liste des abréviations

AdD - Arbre de Défaillance
ADL - Langage de Description d'Architecture
AFNOR - Association Française de NORmalisation
AMDE - Analyse des Modes de Défaillance et de leurs Effets
AMDEC - Analyse des Modes de Défaillance et de leurs Effets et de leur Criticité
CA - Conséquences Acceptables
CCS - Calcul des Systèmes de Communication
CF - Coefficient de Fiabilité
CI - Conséquences Inacceptables
CIMENCAM - Les Cimenteries du Cameroun
CNRS – Centre National de Rechreche Scientifique
CU - Coefficient d'Utilisation
DCAI - Durée Cumulée d'Arrêt sur Incident
D_{eff} – Disponibilité efficiente
Do - Disponibilité Opérationnelle
FSM - Machines à états finis
GRAI – Graphe de Résultats et Activités Interreliés
HM - Heure de Marche
HO - Heure d'Ouverture
IML - Institut de mathématique de Luminy
LaBRI - Laboratoire Bordelais de Recherche en Informatique
MACQ - Méthode d'Arbre des Conséquences
MdD - Modes de Défaillances
MECS – Mécanisme Commande Entrée Sortie
MEF - Machines à Etats Finis
MTTF - Mean Time To Failure
NPB – Nombre de Pièces Bonnes
NPR – Nombre de pièces réalisées
NPTR - Nombre de Pièces Théoriquement Réalisables
OEE - Overall Equipment Effectiveness
PdM - Processus de Markov
P_{eff} - Performance efficiente
PRG - Panne Résumée Globale

Q_{eff} – Qualité efficiente
RdP - Réseaux de Petri
RdPS - Réseaux de Petri Stochastiques
RdPSG - Réseaux de Petri Stochastiques Généralisée
SADT – System Analysis and Design Technic
SAP – Système de Production Automatisé
SdF – Sûreté de Fonctionnement
SdP – Système de Production
SED – Système à Evénement Discret
t_{AF} – Temps d''arrêts fonctionnels
t_{AI} – Temps d''arrêts induits
t_{AP} – Temps d''arrêts propres
TF – Temps de Fonctionnement
TGV – Train à grande vitesse
TN – Temps Net
TO – Temps d''Ouverture
T_p - Taux de performance
T_q - Taux de qualité
TR – Temps Requis
TRE – Taux de Rendement Economique
TRG – Taux de Rendement Global
TRS – Taux de Rendement Synthétique
TRS_G – Taux de Rendement Synthétique Global
TT – Temps Total
TU – Temps Utile
UMR- Unité Mixte de Recherche

INTRODUCTION GÉNÉRALE

Tout au long de leur vie opérationnelle, les systèmes industriels simples ou complexes sont soumis aux fluctuations issues d'origines multiples et variés : indisponibilté des systèmes due aux arrêts propres et induits, perte de performance due aux ralentissements et aux écarts de cadence, perte de qualité. Ces fluctuations entrainent une perte de leur efficience, engendrant une baisse de rendement et une augmentation des coûts indirects (maintenance, reconditionnement, reconfiguration…).

La bonne tenue de ces systèmes dépend étroitement de leur efficience caractérisée par un ensemble d'indicateurs de performance permettant d'avoir une vision plus globalisante du rendement et des moyens engagés. L'évaluation de performance de tels systèmes industriels par une étude comportementale nominale et dysfonctionnelle de leurs ressources élémentaires (humaines, organisationnelles, techniques) permet de disposer d'indicateurs décisionnels qui assureront une optimisation des politiques de maintenance et de production.

Nous présentons dans cette thèse une méthode d'évaluation de l'efficience des systèmes industriels de production de biens basée sur la modélisation comportementale (temporelle et stochastique) complétée par la simulation des dysfonctionnements. Le but de notre travail est d'appréhender au travers de structures classiques fonctionnelles de type série, parallèle et complexes (assemblage et expansion) la propagation des fautes, afin de caractériser l'indicateur d'efficience par le Taux de Rendement Synthétique (TRS), principal indicateur viable au niveau industriel.

L'efficience d'une entité de production est un indicateur de performance global-local, c'est-à-dire pouvant être calculé pour n'importe quel niveau de décomposition hiérarchique (système ou sous système). En considérant l'ensemble des niveaux, le lien entre les indicateurs d'efficiences d'un système de production sera défini à partir des liaisons intrinsèques entre tâches et ressources de production [KOM 06]. On retiendra dans ces travaux que l'efficience est fonction de la Disponibilité, de la Qualité et de la Productivité. On peut ainsi exprimer un lien direct entre l'impact des activités de maintenance et celles de production dans une entreprise. Bien que ces deux activités manipulent le même objet, elles ont rarement les mêmes objectifs. La « Production » s'attache à l'amélioration de la productivité, en se fiant à un indicateur unique de performance globale de type TRS, alors que la « Maintenance » relève du bon fonctionnement du système en contrôlant la fréquence d'apparition des pannes, leur gravité, ainsi que leur recouvrement. Dès lors, l'efficience peut donc être considérée comme une combinaison d'indicateurs tel le TRS et la Disponibilité [KOM 09b].

Évaluer la performance globale d'un système de production de biens est un problème ardu qui nécessite la prise en compte de la bonne interaction de ses différents constituants (humains, organisationnels, techniques) [INN 06]. Avec la complexité croissante des systèmes industriels et l'importance que l'on attache à leur capacité à fonctionner correctement et d'une manière continue, le besoin de modéliser fidèlement leurs comportements fonctionnel et dysfonctionnel pour ensuite évaluer leur performance globale, se fait de plus en plus pressant. Divers indicateurs sont déjà communément utilisés dans le domaine de la Sûreté de Fonctionnement des systèmes [CAB 99] et [VIL 88]. Au-delà des concepts classiques de Fiabilité [PAG 80], [LYO 93], [LAB 00], [DAV 01]et [COT 99] de Disponibilité Instantanée [CHA 01] et [CAO 02], d'autres indicateurs plus généraux, car ne relevant pas exclusivement

du comportement binaire (marche/panne) du système, sont apparus ces dernières années [SOR 06]. Parmi ces indicateurs, le TRS [WIL 06], [NOT 03] et [MEU 02] devenu au travers de la norme NFE 60-182 [AYE 03a], un indicateur pertinent de l'efficience des systèmes de production.

De nombreux travaux [BOU 07], [DIS 02], [HUA 03], [KOM 06], [QMA 08] et [vRT 03] présentent aujourd'hui le TRS comme l'indicateur majeur de performance des systèmes de production, mais on ne retrouve aucune proposition de modélisation. Mahadevan (2004) et Huang et al. (2003) ont présenté le TRS comme combinaison de trois facteurs efficients à savoir : la Qualité, la Performance et la Disponibilité. Leur raisonnement n'est basé que sur l'aspect productivité. Le point de vue Sûreté de Fonctionnement n'est pas intégré dans cette évaluation. Or, nous savons que pour évaluer l'efficience d'un système, le seul aspect productivité ne suffit pas, il faut tenir compte du comportement des machines. D'autres travaux, bien que basés sur les temps d'état d'un moyen de production, semblent n'orienter leurs objectifs que sur la démarche de mesure du TRS [AYE 03a], [AYE 03b] et [WON 07]. Or la mesure n'est faite qu'après occurrence des événements ayant entraîné la dégradation de l'efficience, ce qui ne permet pas d'envisager une véritable politique d'optimisation de l'efficience et de fourniture d'indicateurs décisionnels. Quant aux méthodes d'évaluation de performance, elles existent et exploitent des modèles comportementaux de type réseaux de Petri [KRO 01], [FER 04], [DEC 03], [JAM 01], [MAZ 95], [GRO 97], [LIM 99], [SAR 99], [SAL 01], [LAB 04] et [DJE 03] ou systémiques de type de modèle GRAI (Graphe de Résultats et Activités Interreliés) [LUP 06] et [MAH 04]. Aucune de ces méthodes ne présente par modélisation ni par simulation, l'impact de la variation du TRS et l'influence du comportement humain sur l'efficience des systèmes. Elles ne permettent donc pas une fourniture d'indicateurs décisionnels. Les travaux de cette thèse permettent non seulement de modéliser la variation du TRS sous la dynamique de ses trois composantes (Performance, Qualité et Disponibilité Opérationnelle) par des modèles automates d'état, mais aussi les comportements du Réparateur et de l'Opérateur et de simuler sous AltaRica Data-Flow [RAU 06] cette dynamique. Cette simulation stochastique permet d'obtenir les différents seuils du TRS qui sont comparés à ceux imposés par la World Class Performance. Ces seuils sont déclencheurs des prises de décision.

Le but de nos travaux est de représenter la dynamique des trois composantes du TRS que sont : le taux de Qualité, le taux de Performance et la Disponibilité Opérationnelle [KOM 09a] et [KOM 09b] afin d'évaluer la variation de l'efficience des systèmes de production pour guider les prises de décision opérationnelle (maintenance, conception…). La Disponibilité Opérationnelle [SMA 03] et [CAO 02] est définie selon la norme NFE 60-182 comme étant le rapport du temps de fonctionnement sur le temps requis. Dans le cas d'un processus en exploitation, l'identification de la Disponibilité Opérationnelle peut être obtenue par l'usage des données statistiques issues de l'exploitation réelle du processus durant une période de temps T en usant de la relation suivante :

$$A = \int^{T} \frac{Q_r(\tau)}{Q_0}\, d\tau$$

Où Q_0 est la capacité de production nominale ;
Q_r est la capacité réelle de la chaîne de production.

Les valeurs seuil du TRS sont celles de la World Class Performance. Tout dépassement de ces seuils est déclencheur d'une prise de décision pour les politiques de maintenance et d'exploitation des systèmes de production.

Si le calcul du TRS pour un composant (sous-système) simple isolé est assez aisé, sa modélisation pour une évaluation globale comprenant l'aspect multi état du système, de son environnement, et du facteur humain l'est beaucoup moins. Pour permettre une prise en compte de toutes les ressources constitutives d'un système (humaines, organisationnelles et techniques), nous avons présenté chaque composante du TRS sous trois modèles à base d'automates d'états: modèle de mode, modèle de calcul, modèle de composantes humaines.

Le premier modèle est basé sur les états (marche (m), marche dégradée (m_d) et hors service (hs)) dans lesquels peut se trouver un système en fonction de la valeur du TRS, et donc de son efficience,

Le second modèle est dédié au calcul afin de compter le temps de séjour dans chaque état, et donc de suivre la variation du TRS en fonction des temps d'arrêts (T_A) (connus d'une part pour les arrêts fonctionnels, et aléatoires d'autre part pour les pannes, les pertes en Qualité et les pertes en Performance) et du temps requis (TR),

Le dernier modèle est un modèle des composantes humaines (Opérateur et Réparateur) responsables générateurs respectivement des événements d'arrêts fonctionnels ainsi que des remises en service et des réparations.

Les modèles d'automates d'états ont été élaborés afin de décrire la dynamique de la variation du TRS (au travers de la Qualité, Performance et Disponibilité Opérationnelle) en fonction de ses trois vues. Ces modèles ont ensuite été traduits en AltaRica Data-Flow [RAU 06], langage à la fois formel (basé sur la notion de garde des systèmes de transitions (automate de mode)), de description (conçu pour écrire et analyser les modèles) et hiérarchique. L'utilisation de ce langage nous a permis de nous affranchir du problème d'explosion combinatoire qu'aurait présenté l'usage d'un outil classique comme les réseaux de Petri. L'usage des réseaux de Petri ne présente pas de problème si l'on recherche simplement le temps de séjour dans les places. Si par contre l'on recherche l'évaluation au travers de son graphe d'état, on obtient une explosion. De plus, AltaRica Data-Flow manipule facilement plusieurs modèles automates établissant des liens de synchronisation, et des liens de flux [ARN 05]. Cette manipulation sera importante dans le résultat recherché au niveau de l'entreprise (passage du local au global, politique des interventions…). L'association des lois de probabilité aux événements de commutation d'un état de fonctionnement à un autre est également un des apports indéniables du langage AltaRica Data-Flow. Il permet de gérer avec simplicité les processus stochastiques propres à la dégradation fonctionnelle du système.

Ce travail est scindé en deux grandes parties. La première partie est consacrée à la présentation de l'état de l'art, elle est constituée de 3 chapitres.

Le premier est dédié à la représentation d'un système de production (sa structure, sa composition, ses différentes fonctions, sa nature, ses conditions d'exploitation et son environnement) et les différents critères d'évaluation de performance. Il présente également la notion de propagation des fautes et dysfonctionnement.

Le chapitre 2 présente la notion d'événements distinguant ainsi les différentes occurrences pouvant se produire dans un système de production en fonctionnement et notamment ceux qui seraient intéressants pour l'étude du TRS.

Le troisième chapitre aborde la méthodologie de la modélisation des systèmes de production, avec ses formalismes. Il présente également les principes de l'analyse prévisionnelle : l'analyse structurelle et fonctionnelle, l'analyse qualitative et enfin l'analyse quantitative ou évaluation de performance.

La deuxième partie de ce mémoire de thèse concerne plus particulièrement notre contribution. Elle regroupe trois chapitres.

Le chapitre 4 présente les procédures de calcul du TRS. Il s'agit ici des formules de calcul d'indicateurs de performance en tenant compte de la structure du système. La première partie de ce chapitre présente le TRS comme indicateur pertinent de performance. En deuxième partie de ce chapitre, nous présentons la norme NFE 60-182 introduisant les différents temps d'état de production. En troisième partie de ce chapitre, nous présentons des formules de calcul du TRS en tenant compte de l'aspect productivité. En quatrième et dernière partie de ce chapitre sont présentées les méthodes d'évaluation du TRS sous le seul aspect Sûreté de Fonctionnement. La proposition dans ce chapitre de formules de calcul du TRS global et de la considération des structures des systèmes complexes (série, parallèle, assemblage et expansion) est d'un apport indéniable.

Le chapitre 5 présente la modélisation temporelle et stochastique du TRS. Il s'agit de présenter le TRS et ses composantes sous forme d'automates d'états. Ces automates donnent une description de la dynamique de la variation du TRS. En fonction de la valeur du TRS, nous avons défini trois états dans lesquels est affecté le système : l'état marche (m) si la valeur du TRS est $\geq 85\%$, marche dégradé (md) si $25\% \geq TRS \geq 85\%$, et hors service (os) si $TRS \leq 25\%$. Ce chapitre présente ensuite le modèle de l'efficience en fonction de la Disponibilité Opérationnelle, en fonction de la Qualité et de la Performance. Un modèle automate décrivant les comportements de l'Opérateur et du Réparateur est également donné.

Le chapitre 6 donne une modélisation sous AltaRica Data-Flow pour sa procédure de calcul. Il s'agit en fait de traduire en AltaRica Data-Flow, tous les modèles automates d'états décrivant la dynamique du TRS et de ses composantes. La simulation stochastique de chaque modèle ainsi traduit permet de calculer la valeur du TRS en fonction des taux de transition caractérisant les événements retenus. La valeur obtenue permet alors d'évaluer l'efficience du système en tenant compte des seuils de la World Class Performance [QMA 08], [AYE 04], [CIM 04], [vTR 03] et [CLE 00].

Nous concluons enfin ce travail en faisant le bilan de notre étude, et en donnant également des ouvertures pour les travaux à venir.

État de l'art

INTRODUCTION

Cette première partie est consacrée à la présentation de l'état de l'art. Pour évaluer la performance d'un système, il est important de connaître sa structure, sa composition, ses différentes fonctions, sa nature, ses conditions d'exploitation et son environnement. Elle est constituée de 3 chapitres.

Le premier est dédié à la représentation d'un système de production (sa structure, sa composition, ses différentes fonctions, sa nature, ses conditions d'exploitation et son environnement), ainsi que les différents critères d'évaluation de performance. Il présente également la notion de propagation des fautes et dysfonctionnement.

Le chapitre 2 présente la notion d'événements distinguant ainsi les différentes occurrences pouvant se produire dans un système de production en fonctionnement et notamment ceux qui seraient intéressants pour l'étude du TRS.

Le troisième chapitre aborde la méthodologie de la modélisation des systèmes de production, avec ses formalismes. Il présente également les principes de l'analyse prévisionnelle : l'analyse structurelle et fonctionnelle, l'analyse qualitative et enfin l'analyse quantitative ou évaluation de performance.

Chapitre I

MODÉLISATION ET ÉVALUATION DE PERFORMANCE DES SYSTÈMES DE PRODUCTION (SdP)

1.1 INTRODUCTION

Le but de notre travail est de modéliser la propagation des fautes dans un système comportant plusieurs éléments montés en séries, en parallèle en assemblage ou en expansion, afin de caractériser le Taux de Rendement Synthétique (TRS), indicateur de son efficience. Les différentes machines qui composent chaque système sont caractérisées par un taux de chargement ε et de service δ et exécutent une opération à la fois [MOH 06]. Chacune des machines puise des produits à usiner dans un stock d'entrée et dépose les produits finis dans un stock de sortie. Chaque stock de sortie correspond au stock d'entrée de la machine suivante et ainsi de suite. Le processus respecte le code MECS (Mécanisme, Entrée, Commande, Sortie) caractérisant la modélisation SADT (System Analysis and Design Technic) de chaque cellule élémentaire de la chaîne. Cette modélisation, bien que n'étant plus d'actualité, représente le système sous forme d'Actigramme (boîte noire) à l'intérieur duquel se déroule le processus. L'entrée du système correspond à la matière d'œuvre à l'état brut, les sorties des cellules élémentaires correspondent à la matière d'œuvre dotée de la valeur ajoutée issue de l'action du processus à l'intérieur de la boîte. La sortie du système représente la matière d'œuvre à l'état final à laquelle sont ajoutés les rebuts. Ainsi, chaque produit doit passer dans un ordre bien défini par différentes machines entre lesquelles il reste en attente dans les zones de stockages. Tout Système de Production Automatisé (SAP) étant séquentiel, une machine ne peut commencer à exécuter une opération que si le produit est présent dans le stock d'entrée et qu'il y a de la place disponible pour entreposer le produit une fois usiné dans le stock de sortie. La dynamique du système (prise en compte des synchronisations, des valeurs d'échelle etc.) est modélisée par SADT^{+} [ZAY 93] et [NOW 97] qui consiste à intégrer les contraintes de temps sur l'enclenchement ou le déclenchement d'activité interconnectées [NIE 94]. La « *qualité totale* » du produit à la sortie du système permet ainsi de définir le TRS global, principal indicateur de sa performance.

1.2 NATURE D'UN SYSTÈME

Les caractéristiques et les contenus d'un modèle appelé à décrire le comportement dynamique des systèmes réels varient selon les objectifs de l'étude et le degré de finesse de représentation recherché [MOH 06]. Selon le but de la modélisation et la logique de changement d'états, le modèle développé est de nature différente [DAV 93] et [OUN 99] et peut être :

- *Continu* : Lorsque l'état du système est caractérisé par des paramètres dont la valeur évolue de façon continue dans le temps (figure 1.1.a). La loi d'évolution de ces variables est alors généralement décrite par un système d'équations algébriques ou différentielles ;

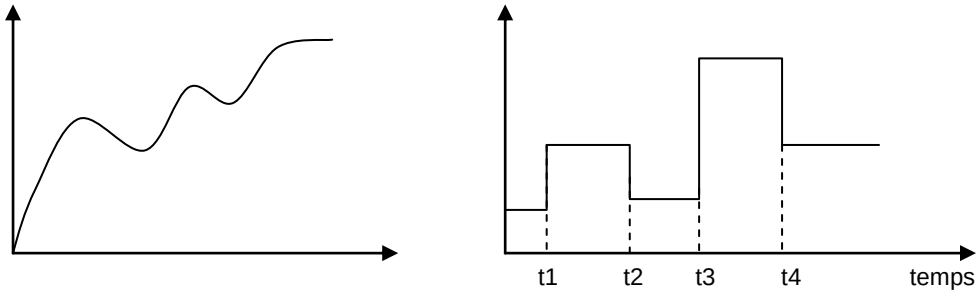
- *Discret* : Lorsque l'espace d'état est discret (les systèmes évoluent de façon discrète dans le temps), c'est-à-dire qu'il existe une suite strictement croissante de nombres réels positifs *t(i), i* = 0, 1, 2, …, nommés instants, telle que durant la quantité $d(i) = t(i+1) - t(i)$

unité de temps, l'état d'un tel système n'évolue pas. L'état d'un système discret n'est modifié que lors de l'occurrence de certains événements (figure 1.1.b). Les méthodes d'identification des événements sont décrites au § 1.5 du chapitre 1.

Deux modèles sont alors distingués [OUN 99] :

- Les modèles discrétisés qui permettent l'observation de l'état à des instants réguliers prédéterminés selon une loi parfaitement connue. Ceci correspond à une représentation discrète de systèmes continus ;
- Les modèles à événements discrets sont utilisés lorsque les changements d'états se font lors d'instants dont la loi d'apparition est apériodique. Un Système à Evénement Discrets (SED) [CAS 08] évolue conformément à l'arrivée d'événements caractéristiques de changement d'états du système.

Pour anticiper les problèmes de performance avant la construction d'un premier prototype, il est impératif de prévoir les performances du système avant de l'implémenter dans les conditions optimales, dans un souci de gain en coût et en temps lors du développement du système. Pour ce faire, le système est modélisé, et son analyse indique les problèmes potentiels qui peuvent survenir. De plus, nous pouvons ainsi dimensionner le système de manière à en tirer les meilleures performances possibles pour une architecture donnée [BEN 03].



a) Evolution continue b) Evolution discrète

Figure 1.1 : Evolution d'un système dans le temps

Devant la complexité des nouvelles générations de systèmes à concevoir, plusieurs techniques de modélisation ont été développées. Toutes sont basées sur le même principe, consistant à définir successivement :

– les **états** du système (on suppose que les systèmes étudiés ont un nombre fini d'états) ;
– les **transitions** entre chaque état (dynamique du système) ;
– la **temporisation** des transitions (temps mis dans un état donné).

Ainsi, les modèles permettent d'analyser le comportement dynamique d'un système en spécifiant l'ensemble de toutes les transitions possibles entre les différents états du système. Les systèmes étudiés s'exécutent dans un environnement aléatoire. Lors de la modélisation d'un tel système, la principale difficulté consiste à modéliser les transitions, à savoir le temps que l'on passe dans chaque état en fonction des statistiques sur l'environnement et le comportement du système lui même. Dans la plupart des cas, nous supposons que le système est markovien, donc sans mémoire. L'état futur ne dépend alors que de l'état présent, et non

du passé. Les chaînes de Markov, que ce soit à temps continu ou à temps discret, facilitent donc l'analyse des performances des systèmes dynamiques dans de nombreux domaines d'application.

Les hypothèses suivantes sont posées :

- le stock d'entrée et le stock de sortie ont des capacités infinies, alors que les stocks intermédiaires sont supposés de capacité unitaire ;
- on suppose que les systèmes sont réparables, et que la distribution des temps entre défaillances (ou les arrêts), et la distribution des durées de réparation suit une loi exponentielle de paramètres respectivement taux de défaillance λ et de réparation μ constants ;
- on suppose aussi que les taux de chargement et de service sont des paramètres de loi de probabilités distribuées exponentiellement.

1.3 CRITÈRES D'ÉVALUATION D'UN SYSTÈME DE PRODUCTION

Les systèmes de production devenant de plus en plus complexes suite à de nouveaux impératifs de l'appareil de production, telles que la flexibilité, la réactivité, la productivité, la qualité et la robustesse, la conception et l'exploitation de ces systèmes nécessitent des techniques d'évaluation se basant sur deux principaux concepts : l'Efficience et la Sûreté de Fonctionnement.

1.3.1 L'Efficience

Dans le domaine industriel, l'*Efficience* d'un système peut être définie comme étant un attribut mesurable et observable par lequel se définit sa performance et la qualité de ses produits. C'est la constance d'un système à fournir la même qualité d'un produit à tout instant *t* donné. Elle est liée à tout le cycle de vie du système de production, à sa productivité, aux stocks et en-cours, aux coûts de production, aux délais de livraison etc.

L'Efficience d'un système est mesurable et appréciable à partir d'un indicateur de performance qui donne une vision temps réel de son fonctionnement. L'indicateur de performance permet également d'avoir une accessibilité aux paramètres de fonctionnement du système à chaque instant donné. En pratique, l'atteinte de l'objectif de l'entreprise est un indicateur de performance crédible dans la mesure où les objectifs sont eux-mêmes définis de manière volontariste [SAS 98]. L'Efficience d'un système permet donc de mettre en jeu des critères quantitatifs (rentabilité désirée) ou qualitatifs (une meilleure image de marque par exemple).

Selon l'Association Française de NORmalisation (AFNOR), l'Efficience se définit de la façon suivante : « *rapport entre les résultats obtenus et l'utilisation des moyens dédiés pour les obtenir* ».

La définition de l'Efficience proposée par le laboratoire Ampère UMR CNRS 5005 en 2003 est la suivante : « *l'Efficience est un indicateur qui sous-entend plusieurs notions : une notion de cadence et une notion de disponibilité* ». L'Efficience peut donc être considérée comme une combinaison d'indicateurs tel le TRS qui lui même comporte les notions de cadence et de Disponibilité.

L'Efficience d'un outil de production est donc un indicateur de performance global-local, pouvant être calculé pour n'importe quel système ou sous système de l'outil de

production et pour n'importe quel niveau de décomposition. Le lien entre toutes les Efficiences d'un outil de production est crée directement à partir des liaisons intrinsèques entre les tâches et/ou composants physiques de l'outil de production. L'Efficience prend en compte à minima la Disponibilité, le rendement et l'efficacité de l'outil de production et peut être étendu à la Qualité, la Sécurité ou tout autre élément périphérique affectant les performances de l'outil.

a) Calcul de l'Efficience

Les calculs effectués pour la définition de l'Efficience dépendent de la norme utilisée. Les formules qui vont suivre sont définies à partir de la norme CNOMO AFNOR. La formule proposée par le laboratoire Ampère est la suivante :

Efficience* = *Taux de Qualité* x *Taux de Disponibilité* x *Taux de Productivité

$$E_{ff} = T_q \text{ x } T_d \text{ x } T_p \qquad (1)$$

En fonction des différents temps utilisés, on peut encore définir l'efficience par :

$$Eff = \frac{TF}{TF + TAP} \times \frac{T_C}{T_C + TdTC} = \frac{T_C}{TF + TAP}$$

TF **:** Temps de fonctionnement
TAP **:** Temps d'arrêt propre
Tc **:** Temps de cycle
TdTC **:** Temps de dépassement de temps de cycle

b) Combinaison des Efficiences

L'Efficience d'un système est considérée comme une combinaison des Efficiences de tous ses sous-systèmes. Ceci signifie qu'il faut recueillir les informations aux niveaux les plus bas de la modélisation puis remonter par couches pour obtenir une Efficience globale du système. Ceci n'est pas si simple. D'où l'intérêt de notre travail.

Considérons un système global composé de *n* sous-systèmes interagissant en séquence ou en parallèle. Il faut mentionner que si deux sous-processus sont en parallèle, seul un sera pris en compte dans le calcul d'Efficience du système global.

Soit *i* appartenant à l'ensemble des tâches du chemin critique (ensemble des sous-processus constituant le temps de cycle le plus long possible), par définition, l'Efficience du système global est égale à :

$$Eff_G = \frac{TC_G}{TF_G + TAP_G} = \frac{\sum_i TC_i}{\sum_j \left(TF_j + TAP_j\right)} \qquad (2)$$

On considère en effet que le temps de cycle global du système est la somme des temps de cycle de chacun des sous-systèmes appartenant au chemin critique. Si l'on transforme le dénominateur à l'aide de la formule de base de l'Efficience, on obtient :

$$TF_i + TAP_i = \frac{TC_i}{Eff_i}$$

D'où l'efficience du système global :

$$Eff_G = \frac{\sum_i TC_i}{\sum_j \left(\frac{TC_j}{Eff_j} \right)} = \frac{\sum_i TC_i}{\frac{\sum_j \left[TC_j \times \prod_{k \neq j} Eff_k \right]}{\prod_j Eff_j}}$$

Pour *i, j,* et *k* allant de 1 à *n*, soit :

$$Eff_G = \frac{\sum_i (TC_i) \times \prod_j (Eff_j)}{\sum_j \left[TC_j \times \prod_{k \neq j} (Eff_k) \right]} \times \prod_m \left[\frac{Tt - TAF_m}{Tt} \right] \quad (3)$$

Où *i* et *j* sont des actions appartenant au chemin critique et *m* des actions permanentes.

En conséquence, la mesure quantitative de la performance économique reste très souvent l'élément principal qui guide l'évaluation du système de production. D'autre part, la rentabilité ou le rendement d'un processus de fabrication peut s'analyser en termes de formation du résultat et en termes de ratios caractéristiques.

1.3.2 La Sûreté de Fonctionnement (SdF)

La Sûreté de Fonctionnement (SdF) d'un système permet d'établir le degré de fonctionnement que l'on peut lui attribuer dans le cadre de la mission qu'il doit assurer [SAS 98]. C'est une notion générique qui mesure la qualité de service délivré par un système, de manière à ce que l'utilisateur ait en lui une confiance justifiée [SCH 04]. Les objectifs fixés actuellement par la production industrielle intégrée visent à optimiser un site de production au sens de la productivité, de manière à ce qu'il assure convenablement le service pour lequel il est conçu (produire en qualité, au moindre coût et dans un temps limité). Cette notion d'assurance et de confiance accordée au service délivré définit le concept de Sûreté de Fonctionnement [NIE 94]. Appelée également « Science des Défaillances » [VIL 88], elle est liée à tout le cycle de vie du système de production, à sa Disponibilité, à sa Fiabilité et la Maintenabilité de ses machines, etc. Elle consiste à connaître, évaluer, prévoir, mesurer et maîtriser les défaillances des systèmes technologiques et les défaillances humaines [ZWI 08]. Elle est l'étude structurelle (statique) et dynamique des systèmes du point de vue prévisionnel, mais aussi opérationnel et expérimental en tenant compte des aspects probabilités et conséquences des défaillances.

La Sûreté de Fonctionnement est donc la propriété qui permet à ses utilisateurs d'accorder une confiance justifiée dans le service qu'il leur délivre [LAP 96]. La notion de service se définit comme le comportement du système tel qu'il est perçu par ses utilisateurs (autres systèmes humains et/ou physiques en interaction). Un système durant sa vie

opérationnelle d'un point de vue SdF, peut être conçu par ses utilisateurs comme une alternance d'états de service :

- *service approprié*, où le service délivré est conforme au service spécifié ;
- *service inapproprié*, où le service délivré est différent du service spécifié.

Les principales techniques d'analyse de la SdF et leur adéquation possible aux différentes étapes de développement d'un SdP préconisent l'enchaînement des étapes suivantes :

- *L'analyse préliminaire* : basée sur une approche causale (exemple : Analyse des Modes de Défaillance et de leurs Effets (AMDE), Analyse des Modes de Défaillance de leurs Effets et de leur Criticité (AMDEC), l'Arbre de Défaillance (AdD), l'objectif étant de constituer un répertoire des différents types de scénarios de pannes possibles. Ces méthodes sont considérées comme statiques ;
- *L'analyse qualitative* : basée aussi sur une approche causale, consiste à exploiter tous les cheminements possibles entre les événements initiateurs et les événements redoutés (objectif de sûreté) ;
- *L'analyse quantitative* : basée sur une approche causale ou une approche d'état (exemple : les graphes de Markov, les réseaux de Petri stochastiques (RdPS)) dont le but est la quantification probabiliste de l'occurrence des évènements redoutés.

La difficulté d'analyser et d'évaluer la Sûreté de Fonctionnement est fonction de la complexité du système de production; notamment du fait qu'il génère et sollicite de nombreux services à nombreux utilisateurs. Les particularités d'intégration du concept de la Sûreté de Fonctionnement dans le domaine de la productique sont donc caractérisées par l'hétérogénéité du milieu (en machines et outils, en informations, en énergies, en principes de commande etc.), par la prise en compte des facteurs humains (l'Opérateur s'expose au danger, mais fait partie intégrante du système en représentant lui-même une entité fonctionnelle il en est de même du Réparateur), ainsi que par la nature des diverses procédures d'exploitation.

C'est ainsi que fortement intégrée dans le processus de production, la SdF doit être mise en place à différentes étapes de l'automatisation (conception, spécification, réalisation, mise au point, exploitation) ; son rôle consiste à identifier les dysfonctionnements, à les prévenir et réduire leur impact s'ils se produisent. La SdF est un concept générique, généralisant et englobant plusieurs composantes définies par rapport à une certaine perception du service délivré par le système, telle que la Maintenance, la Disponibilité, la Fiabilité et la Sécurité.

Les limites et les recouvrements de ces différentes composantes ne sont pas toujours très bien définis et dépendent fortement du processus de production et de la façon dont il est exploité. Néanmoins, dans le domaine productique, les performances d'un site de production sont évaluées essentiellement en termes de *Disponibilité* et de *Sécurité* (figure 1.2) ; la première résulte de la bonne fiabilité du matériel et des conditions de sa maintenance, la seconde, de son comportement vis-à-vis des défaillances critiques.

Une défaillance d'un système est observée lorsque le service délivré dévie du service spécifié. La défaillance est la conséquence d'un comportement erroné du (ou d'une partie du) système : une erreur est une composante valuée de l'état du système (par rapport au processus de traitement) susceptible d'entraîner une défaillance. La cause adjugée ou supposée d'une erreur est une faute dont l'origine est due à un phénomène physique ou à un comportement

humain erroné. Une erreur est donc la manifestation d'une faute dans le système, alors qu'une défaillance est l'effet d'une erreur sur le service. *La propagation des fautes dans un système est donc génératrice des dysfonctionnements, principales causes de la baisse de l'efficience*.

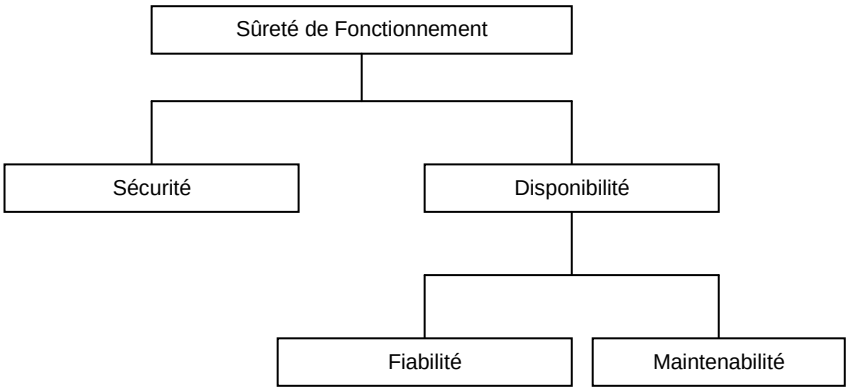


Figure 1.2 : Vecteur Sûreté de Fonctionnement

1.4 PROPAGATION DES FAUTES ET DYSFONCTIONNEMENTS

Un *dysfonctionnement* peut être considéré comme toute dérive de service des activités d'un système par rapport à la fonction requise. Tout dysfonctionnement s'accompagne toujours de perturbations d'activités sur les flux de sortie des fonctions du système [NIE 94].

[ZAY 93] part d'une observation du processus des dérives de service des activités en aval consommatrices ou influencées par la sortie dégradée pour proposer une AMDE fonctionnelle spécifique à chaque activité. Les tableaux AMDE sont alors déterminés à partir des annotations du modèle statique et de règles particulières de production. La figure 1.3 montre les effets d'une défaillance fonctionnelle d'une activité sur ses sorties et les activités aval.

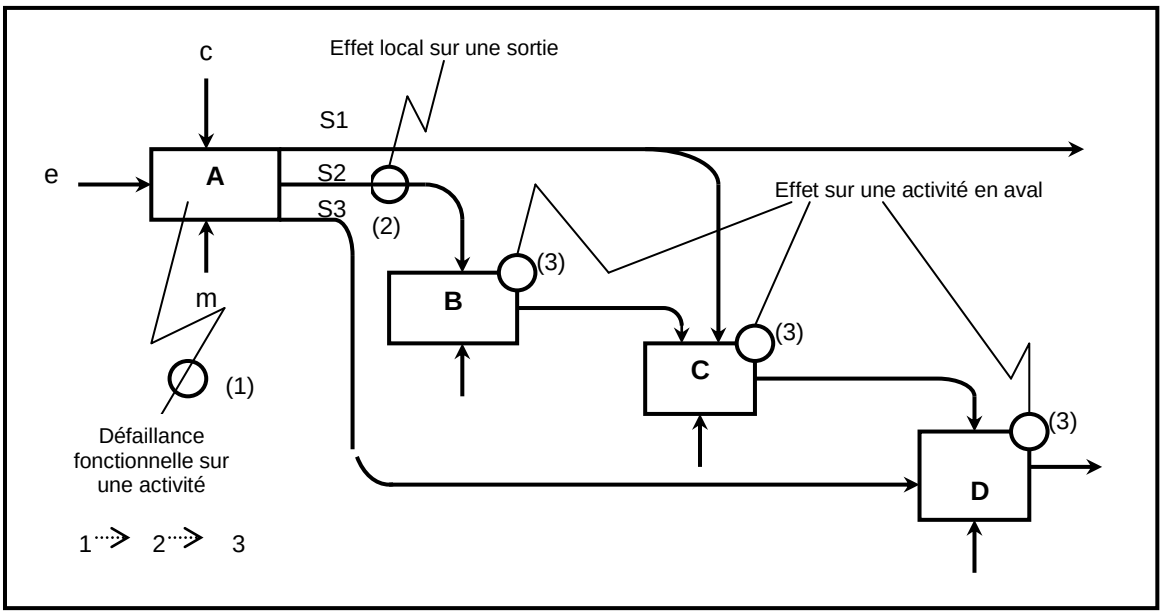


Figure 1.3: Effets d'une défaillance fonctionnelle d'une activité sur ses sorties et les activités en aval

Les interactions entre les activités du modèle SADT ainsi que les annotations établies, fournissent des informations concernant le séquencement des activités ainsi que le niveau de risque local associé à une activité. Ceci permet de proposer pour chaque activité du modèle SADT et pour les Modes de Défaillances (MdD) retenus :

- une gravité intrinsèque représentant le niveau d'insécurité du mécanisme ;
- les perturbations sur les sorties ;
- la propagation de l'effet de défaillance et l'effet de gravité sur les activités en aval.

Deux approches d'AMDE sont alors étudiées concernant la propagation des effets locaux d'une activité sur les activités en interaction.

1.4.1 AMDE-Activités

Objectif : caractériser l'état du service délivré en situation de défaillance. La dégradation du service s'observe directement en sortie de l'activité en défaillance.
Les quatre principaux modes de défaillances sont :

- MdD1 : fonctionnement prématuré
- MdD2 : ne fonctionne pas au moment prévu
- MdD3 : ne s'arrête pas au moment prévu
- MdD4 : défaillance en fonctionnement

Chaque MdD est appliqué sur une activité dont les flux entrants (entrées et contrôles) sont considérés comme non perturbés, on constate directement sur les sorties, l'effet local de ces défaillances.

1.4.2 AMDE-Flux :

Objectif : propager les effets locaux observés à partir des AMDE-Activités afin de relever les relations cause effet des défaillances fonctionnelles.

Le SADT n'est qu'une représentation statique du système, et par conséquent ne ressort pas l'aspect dynamique de la propagation des fautes. Il apparaît donc impératif d'enrichir le modèle fonctionnel SADT par des primitives graphiques, dédiées aux aspects comportementaux, afin d'obtenir un modèle interprétable permettant de suivre le comportement dynamique de la structure fonctionnelle. [NOW 97] propose une approche systémique (hiérarchique et structurée) qui consiste à utiliser au mieux les connaissances structurelles et fonctionnelles du système étudié pour appréhender la propagation des fautes. Elle est basée d'une part sur la modélisation des fautes et à la poursuite de leur propagation dans le système, elle fait appel aux relations causales, support indiqué pour la modélisation comportementale. D'autre part, elle permet de déterminer quantitativement la probabilité d'accès aux situations critiques ; cette étape utilise les modèles dynamiques.

1.4.3 Génération d'un modèle SADT temporel : le SADT^{+}

Les propositions relatives à la représentation dynamique dans SADT se classent de la manière suivante :

- utilisation de séquencements et équations d'activation ;
- utilisation de formalismes pour la description des comportements internes des activités ;
- utilisation d'interfaces typées pour caractériser les flux de production.

L'utilisation de *Séquencements et équations d'activation* est une méthode peu pratique due à la perte d'informations à cause de la surcharge du diagramme.

La méthode par *Formalisme de description de comportement* consiste à associer un modèle SADT à un modèle mathématique (réseaux de Petri ou automates communicants). La figure 1.4 en donne une illustration.

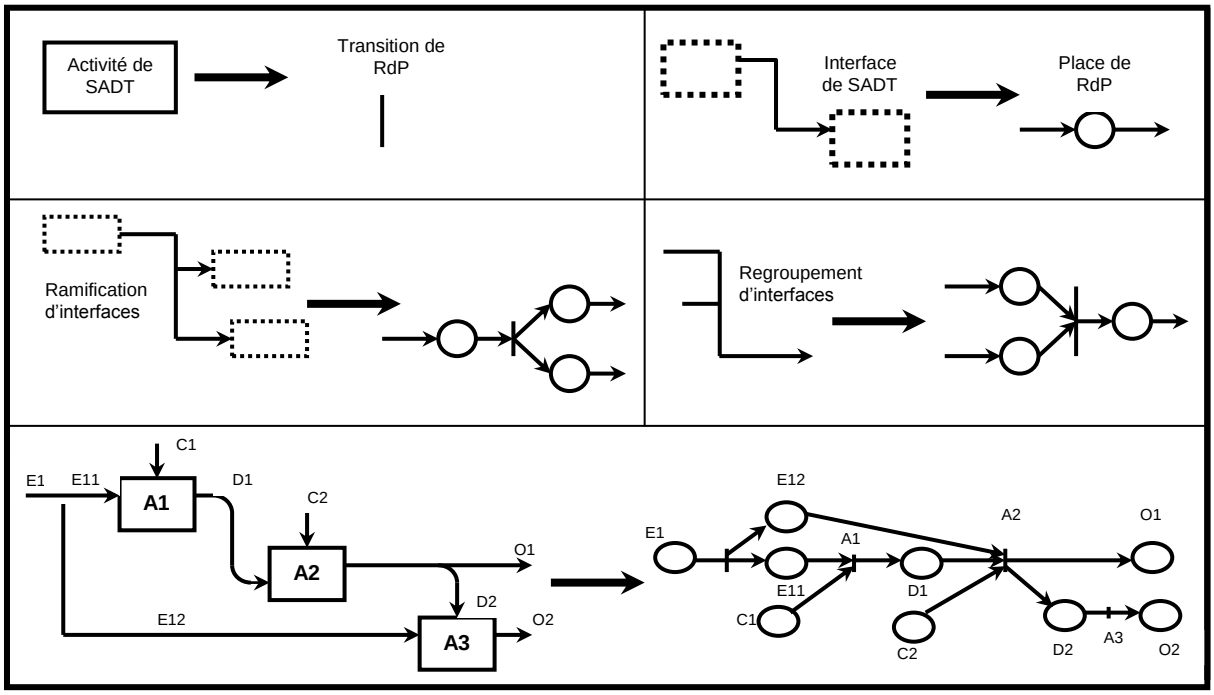


Figure 1.4 : Transformation SADT-Réseaux de Petri

La méthode par *Utilisation d'interfaces typées* intègre les notions suivantes :

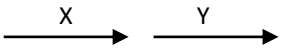
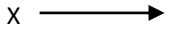
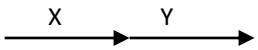
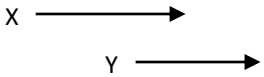
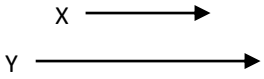
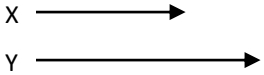
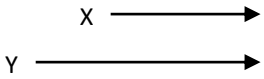
- utilisation de flèches en trait « interrompu » pour représenter un flux de données ;
- utilisation de flèches en pointillé pour représenter un flux de contrôle avec les conventions suivantes :

1) Une flèche étiquetée représente un échange des signaux ;
2) Une flèche comportant le symbole E/D pour la notion d'activation/désactivation ;
3) Une flèche comportant le symbole T pour le déclenchement d'un processus qui s'arrête de lui-même.

Le modèle SADT fonctionnel est ensuite enrichi par des primitives graphiques, dédiées aux aspects comportementaux, afin d'obtenir un modèle interprétable permettant de suivre le comportement dynamique de la structure fonctionnelle.

Dans la phase de spécification, les problèmes de synchronisation d'un SdP consistent à imposer des contraintes de temps sur des actions effectuées par un ensemble d'activités. Le temps sera ainsi modélisé par l'utilisation des relations d'intervalles (tableau 1.1). Les intervalles de temps seront définis par un instant de *début* (d) et un instant de *fin* (f) ; avec « d < f ».

Tableau 1.1 : Relations entre deux intervalles

Relations temporelles	Relations temporelles	Expression graphique	Expression événements synchrones
X précède Y	Y précédé par X		fX < dY
X égale Y	Y égale X		dX = dY
X rencontre Y	Y rencontré par X		fX = dY
X chevauche Y	Y chevauché par X		dX < dY dY < fX fX < fY
X durant Y	Y contient X		dY < dX fX < fY
X commence Y	Y commencé par X		dX = dY fX < fY
X termine Y	Y terminé par X		dY < dX fX = fY

Le but de la modélisation par SADT temporel est d'intégrer des relations temporelles dans les flèches (figure 1.5), afin de rendre plus compréhensible la lecture du schéma, tant du point de vue déplacement des flux, que ordonnancement (séquencement).

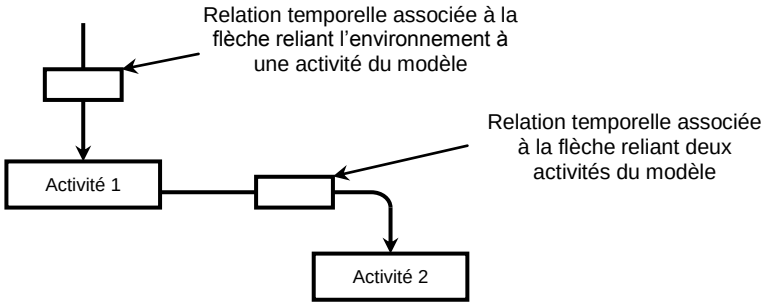


Figure 1.5 : Intégration des relations temporelles sur les flèches

Sept relations sont à prendre en compte : *précède, égale, rencontre, chevauche, contient, commence, termine*. L'intégration de toutes ces nouvelles informations dans le modèle se fait en trois phases :

Phase 1 : *annotation des canaux*

Un canal est la flèche qui relie deux activités du modèle SADT ou une activité et son environnement. L'annotation d'un canal est typée de la manière suivante :

Type = (données, event-synchronisation, event-inhibition, matière)
Phase 2 : *comportements des activités ambiguës*

Utilisation des équations d'activation dans le but d'affiner le comportement de l'activité dans le cas où il y a ambiguïté (relative à une décision, à une sélection ou à plusieurs modes d'activation). Les équations d'activation peuvent intervenir dans les cas suivants :

- L'activité a *plusieurs entrées et plusieurs sorties* : Dans ce cas, il faut préciser la transformation du sous-ensemble d'entrée en sous-ensemble de sortie, par l'utilisation d'équations d'activation de la forme :
  «sortie ←--- entrée» (*lire : sortie provoquée par entrée*)

- L'activité a *plusieurs modes d'activation* dus à la présence de plusieurs interfaces de contrôle de type «event-synchronisation» et plusieurs interfaces de sortie. Dans ce cas, il faut spécifier le contrôle participant à la génération de la sortie en question dans un mode d'activation particulier.
  «sortie ←--- contrôle» (*lire : sortie provoquée par contrôle*)

- Un des modes d'activation de l'activité consiste à *consommer une matière sans rien produire* jusqu'à un mode d'activation ultérieur. Dans ce cas, l'équation d'activation est de la forme :
  «←--- entrée et contrôle» (*lire : aucune sortie provoquée par (entrée et contrôle)*).

- La relation *entrée(s) – sortie(s) dépend des conditions externes* issues d'une interface de contrôle de type «données». Dans ce cas, il faut spécifier le sous-ensemble d'entrées-sorties correspondant. L'équation d'activation est de la forme :
  « (sortie 1 ou sortie 2 ou ...) ←--- (entrée 1 et contrôle)» (*lire : une et seulement une des sorties (sortie 1 ou sortie 2 ou ...) est provoquée par (entrée et contrôle)*).

Phase3 : *génération de SADT temporel (SADT$^+$)*

La génération du SADT temporel est maintenant possible après l'annotation des canaux, et la spécification des équations d'activation.

1.4.4 Le modèle fonctionnel

L'approche classique du SADT est complétée avec les éléments permettant d'exprimer le parallélisme, le séquencement, la synchronisation et l'exclusion mutuelle entre activités. Le modèle comportemental est obtenu en associant à chaque composant SADT$^+$ sa correspondance en RdP. Il faut rappeler que le concept SADT temporel développé par [ZAY 93] permet de caractériser l'état du service délivré par l'activité elle-même et d'observer les dérivés de service des activités en aval, influencées par la sortie dégradée. Mais l'analyse de sécurité ne peut se limiter uniquement aux sorties locales des éléments du système. Par rapport au SADT temporel ayant pour but d'identifier les MdD, d'observer leurs causes et effets, d'étudier leur propagation, SADT$^+$ cherche à évaluer le système dans son ensemble en utilisant le même formalisme de départ, mais servira de base à la définition systématique des états critiques vis à vis des spécifications fonctionnelles. Les relations temporelles ont donc été réorganisées et complétées, pour les uniformiser avec la logique des intervalles et des

durées. Pour conserver la logique d'intervalles, [NOW 97] associe les RdP temporels tant aux boîtes fonctionnelles qu'aux opérateurs de séquencement.

1.4.5 Le modèle dysfonctionnel

Il repose sur la représentation d'activités soumises aux défaillances exprimées en terme de désynchronisation. Les défaillances sont perçues ici comme intervalles temporels affectés aux différentes fonctions du système. La modélisation dysfonctionnelle prend en compte les modes de défaillance suivants :

- MdD1 **:** fonctionnement prématuré ;
- MdD2 **:** ne fonctionne pas au moment prévu ;
- MdD3 **:** panne au cours de fonctionnement ;
- MdD4 **:** ne s'arrête pas au moment prévu.

Un ou plusieurs modes de défaillance doivent être affectés aux activités par l'intermédiaire d'annotations dysfonctionnelles. Une structure réseaux de Petri stochastiques généralisée (RdPSG) est générée pour chaque activité perturbée, elle est ensuite modifiée selon ces annotations.

[NIE 94] présente les Machines à Etats Finis (MEF) comme principale logique appropriée aux changements d'états des services et des flux, car elles correspondent au mieux à la notion de processeur, présente dans les Systèmes à Evénements Discrets (SED) [LHO 03]. Le principe est le suivant : pour chaque MEF appartenant au modèle fonctionnel initial, est mise en place une structure additionnelle traduisant les effets des MdD. Par hypothèse, on suppose que pour toute MEF, il est possible de distinguer deux classes d'états, l'une dite *de veille* où aucune action n'est enclenchable, l'autre dite *de travail* où l'enclenchement d'action est permis. Le mécanisme de dysfonctionnement répond aux procédures suivantes :

- MdD1 **:** exécute la prochaine transition à partir de l'état de veille;
- MdD2 **:** n'exécute pas la transition d'arrêt à partir d'un état actif;
- MdD3 **:** n'exécute pas la transition de marche à partir d'un état de veille ;
- MdD4 **:** reste dans l'état opérationnel courant, mais n'exécute aucune nouvelle transition. La MEF est dormante.

L'occurrence des MdD (MdD1 à MdD4) est exprimée par des événements externes d'entrée, impliquant la réécriture de nouvelles transitions.

Plutôt que de surcharger les spécifications initiales des MEF décrivant le comportement dynamique de chaque activité feuille A_{ij}, il faut conserver l'avantage de la décomposition hiérarchique de SADT. Ainsi, pour chaque A_{ij} seront substituées deux MEF connectées au modèle initial.
Les variables propres à chaque $A_{ij}1$ sont distribuées vers les MEF perturbatrices $A_{ij}2$. Celles-ci héritent d'un nouvel ensemble de variables d'état et d'événements à travers leurs canaux d'entrant (*i'* pour les entrées, *c'* pour les contrôles et *m'* pour les mécanismes).

Le principe retenu pour induire chaque $A_{ij}2$ en erreur consiste à distribuer, à partir des $A_{ij}1$ initiaux (figure 1.6) :

- une valeur caractéristique de la perturbation par l'intermédiaire de variables cognitives ;
- des valeurs erronées aux variables d'entrée ;
- des événements incorrects.

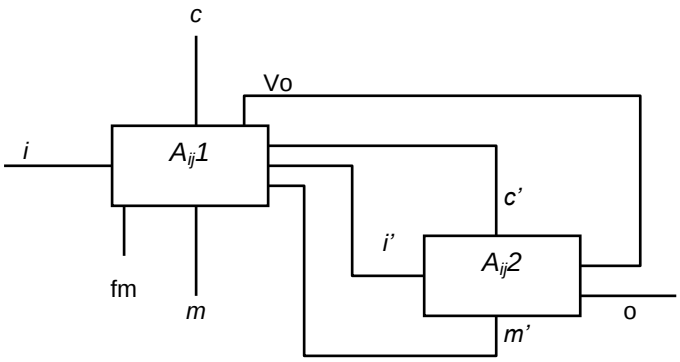


Figure 1.6 : Principe de la décomposition dysfonctionnelle

La détermination des mécanismes de perturbation diffère selon A_{ij} (activités feuilles) considérées, celles-ci sont de trois types (primitive, régulière ou complexe). Une activité sera dite *primitive*, si elle ne possède pas d'autocontrôle ; elle est alors activée de l'extérieur seulement par des ordres de marche-arrêt et fonctionne continûment. Une activité sera dite *régulière*, s'il existe des mécanismes internes de contrôle lui permettant de poursuivre ou d'interrompre l'exécution de sa tâche. Une activité *complexe* est décrite par sa capacité d'autonomie, tant pour son activation que pour sa désactivation. A ces classes d'activité se trouvent associés les mécanismes de perturbation suivants :

- *Activité primitive* : l'induction en erreur s'établit au travers des messages de commande de marche-arrêt, tels que la transmission de :

- MdD1 correspondra à un ordre de départ temporellement incohérent à travers *c'* ;
- MdD2 correspondra à filtrer le message de commande d'arrêt ;
- MdD3 correspondra à filtrer le message de commande de marche ;
- MdD4 correspondra à un ordre d'arrêt temporellement incohérent à travers *c'*.

Lorsque aucun MdD n'est injecté, le modèle dysfonctionnel est alors transparent et transmet correctement les commandes de marche-arrêt de *c* vers *c'*.

- *Activité régulière ou complexe* : La transcription de MdD1 pour ce type d'activité est plus compliquée du fait de l'existence de gardes (traduction de la décidabilité de l'activation associée aux transitions de marche) qui devront être forcées suivant la nature des spécifications.

La transcription des MdD2 à MdD4 est identique à l'activité primitive. Un message sera filtré et non transmis de l'état actif quand MdD2 sera injecté, de l'état de veille pour MdD3 et de l'état courant pour MdD4.

La propagation des défaillances est réalisée lorsqu'une activité saine reçoit :
- une information erronée (à travers une variable cognitive ou un message) ;
- une mauvaise commande, temporellement mal distribuée. Dans ce cas, on constate la robustesse du système si la condition de décidabilité de la réception pour la MEF prévient toute exécution de la transition correspondante ;
- un message non intentionnel, lorsque l'activité réceptrice n'a pas ses gardes activées.

Cette méthode a permis de fournir un modèle dysfonctionnel simulable, réalisé par l'intermédiaire de MEF perturbatrices. Les perturbations considérées se traduisent en termes de filtrage d'informations ou de leurres. Le problème majeur concerne alors l'exploitation d'un tel simulateur (exhaustivité des scénarios d'entrée) et l'analyse des résultats de simulation (validité, représentativité). La similitude avec l'analyse de performance par TRS est perceptible.

1.5 CONCLUSION

Les systèmes de production devenant de plus en plus complexes suite à de nouveaux impératifs de l'appareil de production, telles que la flexibilité, la réactivité, la productivité, la qualité et la robustesse, la conception et l'exploitation de ces systèmes nécessitent des techniques d'évaluation se basant sur deux principaux concepts : l'Efficience et la Sûreté de Fonctionnement.

L'évaluation de l'efficience d'un système basée sur l'étude comportementale fonctionnelle et dysfonctionnelle de ses éléments constitutifs (matériel et humain) repose sur la modélisation et la simulation. Cette évaluation donne des résultats dont l'analyse permet de mettre sur pied des éléments décisionnels et des politiques de maintenance.

La Sûreté de Fonctionnement quant à elle permet d'établir le degré de fonctionnement que l'on peut attribuer à un système dans le cadre de la mission qu'il doit assurer. C'est une notion générique qui mesure la qualité de service délivré par un système, de manière à ce que l'utilisateur ait en lui une confiance justifiée. Les objectifs fixés actuellement par la production industrielle intégrée visent à optimiser un site de production au sens de la productivité, de manière à ce qu'il assure convenablement le service pour lequel il est conçu (produire en qualité, au moindre coût et dans un temps limité).

La difficulté d'analyser et d'évaluer la Sûreté de Fonctionnement est fonction de la complexité du système de production; notamment du fait qu'il génère et sollicite de nombreux services à nombreux utilisateurs. Les particularités d'intégration du concept de la Sûreté de Fonctionnement dans le domaine de la productique sont donc caractérisées par l'hétérogénéité du milieu (en machines et outils, en informations, en énergies, en principes de commande etc.), par la prise en compte des facteurs humains (l'Opérateur s'expose au danger, mais fait partie intégrante du système en représentant lui-même une entité fonctionnelle il en est de même du Réparateur), ainsi que par la nature des diverses procédures d'exploitation. C'est ainsi que fortement intégrée dans le processus de production, la SdF doit être mise en place à différentes étapes de l'automatisation (conception, spécification, réalisation, mise au point, exploitation) ; son rôle consiste à identifier les dysfonctionnements, à les prévenir et réduire leur impact s'ils se produisent. La SdF est un concept générique, généralisant et englobant plusieurs composantes définies par rapport à une certaine perception du service délivré par le système. Les principaux concepts de la SdF que sont : la Maintenabilité, la Disponibilité, la Fiabilité et la Sécurité permettent, tout comme les indicateurs de performance de type TRS, d'évaluer la performance des systèmes de production.

Chapitre II

NOTION D'ÉVÉNEMENTS

2.1 INTRODUCTION

Un système de production simple ou complexe est caractérisé par les différents états qu'il occupe durant tout son cycle de vie. Le changement d'état d'un système de production résulte généralement de l'occurrence d'un événement. Ces événements sont de deux types : probabilistes et déterministes. Dans le domaine de la Sûreté de Fonctionnement, les événements sont généralement de type « *défaillance* » ou « *réparation* ». Au chapitre 5, nous avons représenté le système par des modèles d'états. Le passage d'un état à l'autre se fait non seulement sous l'occurrence des événements probabilistes de type « *défaillance* » et « *réparation* », déterministes de type « *arrêtDétecté* » ou encore « *arrêtProlongé* », mais également sous l'occurrence des événements liés à la perte ou à l'amélioration de la Qualité, à la perte ou à l'amélioration de la Performance et au passage de la valeur du TRS à travers les seuils fixés par la World Class Performance. Afin de permettre une bonne mise en place d'indicateurs décisionnels efficients, tous les événements se produisant dans un système doivent être identifiés et classifiés.

2.2 IDENTIFICATION DES ÉVÉNEMENTS

A chaque instant, l'état d'un système de production est constitué d'opérations à exécuter (un produit peut être à l'état disponible, en cours, etc.), et des ressources qui doivent les exécuter (une ressource peut être en état de veille, active ou en arrêt). Les événements seront de deux types :

- les *événements attendus*, dont on sait qu'ils doivent normalement se produire dès que le système de production est dans un certain état ;
- les *événements inattendus*, qui correspondent à une perturbation associée à une ressource ou à une opération.

2.2.1 Événements attendus

Les événements attendus qui déclenchent le processus de prise de décision sont généralement les fins d'exécution d'opération, les fins de panne et les fins de blocage d'opération :

- la fin d'exécution d'opération libère une ou plusieurs ressources et rend disponible l'opération suivante ;
- compte tenu de l'état du système de production et des objectifs de production, et lorsqu'une ressource est libérée, il faut décider à quelle opération elle doit être affectée ;
- lorsqu'une opération devient disponible, il faut décider avec quelle(s) ressource(s) elle peut être exécutée ;
- lors d'une fin de panne, une ressource peut de nouveau exécuter une opération. Deux cas se présentent :
    - soit cette ressource reprend l'exécution de l'opération arrêtée si celle-ci a attendu la fin de la panne ;

- ou alors, si aucune opération n'est affectée, une décision d'affectation est à prendre.

- deux situations sont possibles lors d'une fin de blocage d'opération :
  - soit l'opération est restée couplée à une ou plusieurs ressources, dans ce cas l'exécution reprend ;
  - ou alors l'opération a été découplée d'une ressource, dans ce cas elle doit être réaffectée.

D'autres types d'événements attendus non spécifiés dans ce contexte peuvent également intervenir dans un système de production (par exemple les reprises secteur avec notamment les reprises à froid, et les démarrages à chaud). Dans le cas d'espèce, un « *chien de garde logiciel*» est impératif dans le système pour déterminer le type de reprise.

2.2.2 Événements inattendus

Un système de production est un milieu extrêmement exposé à des perturbations. Ces perturbations peuvent être liés tant aux ressources, aux opérations qu'aux événements extérieurs au système de production. Ces événements sont généralement de type : *rebuts* ; *non-conformité* ou *défaillance machine*. [VIL 88] montre que la complexité des SdP actuels, leurs nombreuses interactions et les redondances fonctionnelles existantes, rendent difficiles les tâches d'élaboration et d'évaluation des séquences d'événements. Une approche systématique est donc nécessaire pour comprendre et mettre en évidence les nombreux facteurs qui peuvent influencer de telles séquences. L'*arbre des conséquences* se présente donc comme étant l'outil le plus approprié pour la caractérisation et la détermination des événements potentiels pouvant entraîner le dysfonctionnement voire la mise hors d'usage d'un système.

2.2.3 Identification d'événements par la Méthode d'Arbre des Conséquences (MACQ)

Une *séquence d'événements* est une succession d'événements dont le premier dit « *initiateur* » est générateur d'autres événements dits « *génériques* ». Cette succession d'événements conduit à une combinaison de défaillances et de fonctionnements des systèmes de sûreté. Lorsque cette succession ou cette combinaison d'événements conduit à :

- des conséquences jugées inacceptables (CI) : elle sera dénommée « *séquence inacceptable* » ;
- des conséquences acceptables (CA) : elle sera dénommée « *séquence acceptable* ».

a) Événement initiateur et séquence

Par définition, un *événement initiateur* est le premier événement d'une séquence d'événements. Deux approches sont utilisées pour identifier les événements initiateurs :

- l'évaluation technologique prenant en compte la conception de l'installation, les études de risque précédentes, les accidents déjà survenus et plus généralement l'expérience d'exploitation de ces systèmes… ;
- la construction d'un arbre des causes à partir de l'événement indésirable défini au niveau de l'ensemble de l'installation industrielle.

Une séquence (d'événements) est une combinaison temporelle d'événements (fonctionnement ou panne de systèmes élémentaires) conduisant à des événements indésirables.

Une séquence inacceptable est une combinaison temporelle d'événements (fonctionnement ou panne de systèmes élémentaires) conduisant à des conséquences inacceptables (ou événement indésirable).

Un événement initiateur est la première Panne Résumée Globale (PRG) d'une séquence inacceptable. Le principe de PRG permet de fixer le niveau de décomposition (ou de regroupement) auquel est attaché ce premier événement. La définition d'un événement initiateur reste assez délicate, car elle dépend du contexte, de la décomposition du système en systèmes élémentaires qui est très souvent significative du degré de finesse de l'analyse. De plus, dans la pratique, on peut être submergé par le nombre de séquences imaginables. Souvent beaucoup de ces séquences sont improbables et ne méritent pas d'être considérées. Aussi est-il intéressant de coupler la notion de séquence à celle d'une probabilité. Les définitions suivantes sont finalement retenues :

- une séquence inacceptable est une séquence d'événements (fonctionnement ou PRG de systèmes élémentaires) conduisant à des événements indésirables et de probabilité supérieure à une probabilité limite donnée ;
- un événement initiateur est la première PRG d'une séquence inacceptable.

b) Analyse quantitative de l'Arbre des Conséquences

L'analyse quantitative de l'Arbre des Conséquences dépend beaucoup de la nature des dépendances entre les événements constitués par l'événement initiateur et les événements génériques ; très simple lorsque tous les événements sont indépendants entre eux, elle peut devenir très complexe pour certaines dépendances.

D'une manière générale, considérons l'Arbre des Conséquences de la figure 2.1.

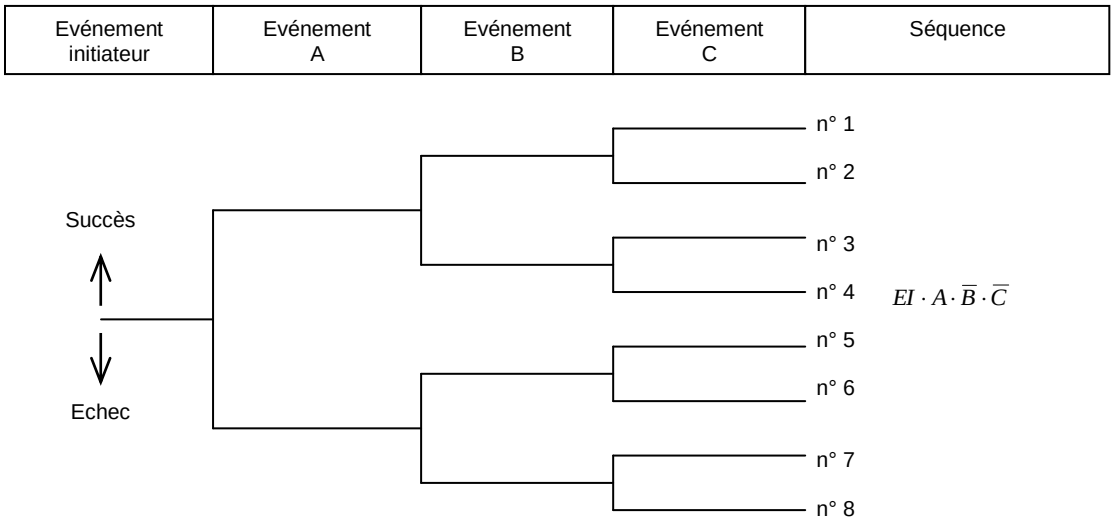


Figure 2.1 : Arbre des Conséquences

La séquence n° 4 s'écrit : $Sq = EI \cdot A \cdot \overline{B} \cdot \overline{C}$ L'utilisation du théorème des probabilités conditionnelles permet d'écrire :

$$P[Sq] = P[EI] \times P[A/EI] \times P[\overline{B}/EI \cdot A] \times P[\overline{C}/EI \cdot A \cdot \overline{B}] \quad (4)$$

Examinons maintenant le cas des événements indépendants et celui des événements dépendants.

*** Événements indépendants :** Supposons que les événements génériques sont indépendants entre eux. La probabilité de la séquence n° 4 s'écrit :

$$P[Sq] = P[EI] \times P[A/EI] \times P[\overline{B}/EI] \times P[\overline{C}/EI] \quad (5)$$

Admettons que l'événement initiateur a un taux d'apparition Λ_{EI} et qu'il n'est pas susceptible de disparaître (pas de réparation). De plus, les événements A, B, C correspondent aux missions de systèmes démarrant (avec des probabilités $\Gamma_A, \Gamma_B, \Gamma_C$) et devant fonctionner (avec des taux de défaillance $\Lambda_A, \Lambda_B, \Lambda_C$) pendant des temps T_A, T_B, T_C.

Calculons la probabilité d'apparition de la séquence n° 4 sur [0, t] :

$$P[Sq] \approx P[EI](\Gamma_B + \Lambda_B T_B)(\Gamma_C + \Lambda_C T_C)$$
$$P[Sq] \approx (1 - e^{-\Lambda_{EI} t})(\Gamma_B + \Lambda_B T_B)(\Gamma_C + \Lambda_C T_C)$$

Avec les conditions :

$$P[A/EI] \approx 1; \quad \Lambda_B T_B \prec\prec 1; \quad \Lambda_C T_C \prec\prec 1$$

Cette formule se généralise facilement lorsque plusieurs systèmes élémentaires sont impliqués dans une séquence accidentelle :

$$P[Sq] \approx P[EI] \times \pi[\text{Probabilités de défaillance des systèmes élémentaires impliqués dans } Sq]$$

On a implicitement admis que les périodes de fonctionnement des systèmes sont petites devant *t*.

Ainsi, de manière approchée, *la probabilité de chaque séquence est le produit de la probabilité de l'événement initiateur par la probabilité d'échec des événements génériques*.

*** Événements dépendants :** Le calcul dépend beaucoup de la nature de ces dépendances. On s'intéresse ici à trois types de dépendances :

- les dépendances de type *cause commune* ;
- les dépendances de type *séquentiel* ;
- les dépendances de type *cause commune* et de type *séquentiel*.

c) Dépendance de type cause commune

Si les Arbres des Causes ont été réalisés pour modéliser les défaillances des systèmes élémentaires intervenant dans les séquences ; des événements communs apparaissent répétés dans différents Arbres des Causes. La réduction booléenne permet d'obtenir les séquences minimales dont les probabilités sont ensuite calculées.

d) Dépendance de type séquentiel

Il est important de mentionner dans cette partie que certains Arbres de Conséquences ayant des dépendances séquentielles peuvent être assimilés à des processus semi-markoviens ; ceci facilite alors le calcul d'"Arbre des Conséquences où existent des réparations des systèmes correspondant à l'"événement initiateur et aux événements génériques.

- *Premier cas* : **systèmes élémentaires fonctionnant en séquence**

Considérons l'"Arbre des Conséquences de la figure 2.2 :

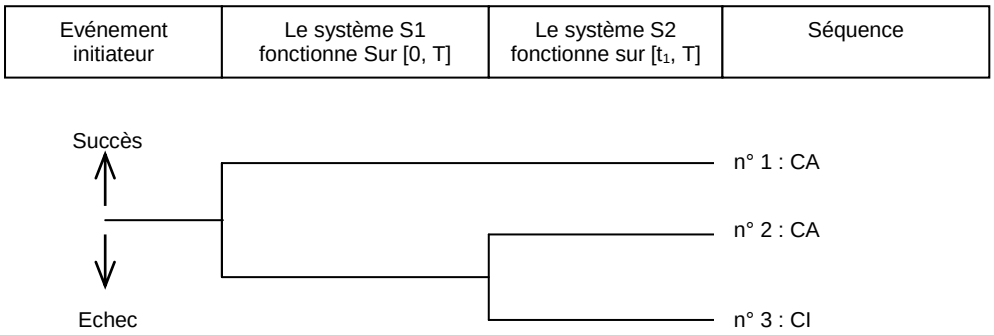


Figure 2.2 : Arbre des Conséquences ayant des dépendances de nature séquentielle

On suppose qu'"après apparition de l'"événement initiateur, le système S_1 doit démarrer et fonctionner sur l'"intervalle de temps [0, T] ; en cas de défaillance, le système S_2 fonctionne en tant que secours. Les conséquences inacceptables (CI) sont atteintes si le système S_2 est à son tour défaillant.

Afin de simplifier l'"écriture des équations, les hypothèses suivantes sont faites :

- l'"événement initiateur est indépendant des événements génériques ;
- les systèmes S_1 et S_2 ne connaissent pas de défaillances au démarrage ;
- le système S_2 est supposé, au début, en attente et sans possibilité de défaillance dans cette phase ;
- les taux de défaillance des systèmes ($\Lambda_1, et\ \Lambda_2$) sont supposés constants ; les systèmes sont irréparables durant leur mission.

La probabilité de la séquence n° 3 se calcule de la manière suivante :

$$P[Sq] = P[EI \cdot \bar{S}_1 \cdot \bar{S}_2]$$
$$P[Sq] = P[EI] \times P[\bar{S}_1 \cdot \bar{S}_2]$$
$$P[Sq] = P[EI] \int_0^T U_1(t_1)\, \bar{R}_2(T - t_1)\, dt_1$$

Où $U_1(t)$ est la densité de défaillance du système S_1, et $\bar{R}_2$ la défiabilité du système S_2, soit :

$$U_1(t_1) = \Lambda_1\, e^{-\Lambda_1 t_1}; \quad \bar{R}_2(T - t_1) = 1 - \exp[-\Lambda_2 (T - t_1)]$$

On obtient :

$$P[Sq] = P[EI]\left[1 + \frac{\Lambda_2}{\Lambda_1 - \Lambda_2} e^{-\Lambda_1 T} - \frac{\Lambda_1}{\Lambda_1 - \Lambda_2} e^{-\Lambda_2 T}\right] \quad (6)$$

Dans le cas où $\Lambda_1 = \Lambda_2 = \Lambda$:

$$P[Sq] = P[EI][1 - e^{-\Lambda T} - \Lambda T\, e^{-\Lambda T}]$$

Il est alors aisé de généraliser les formules de calcul de la probabilité d'une séquence mettant en jeu, de manière séquentielle, *n* systèmes élémentaires dont les densités de défaillance sont :

$U_i(t) = \Lambda_i\, e^{-\Lambda_i t}$ Avec les Λ_i tous distincts :

$$P[Sq] = P[EI]\left[1 + \sum_{i=1}^{n} B_i \exp[-\Lambda_i T]\right] \quad (7)$$

Où

$$B_i = \frac{\prod_{i=1}^{n} \Lambda_i}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (\Lambda_j - \Lambda_i)} \quad avec \quad \Lambda_0 = 0$$

- *Deuxième cas* : **disparition de l'événement initiateur**

Considérons le schéma de la figure 2.3 représentant l'Arbre des Conséquences avec disparition de l'événement initiateur:

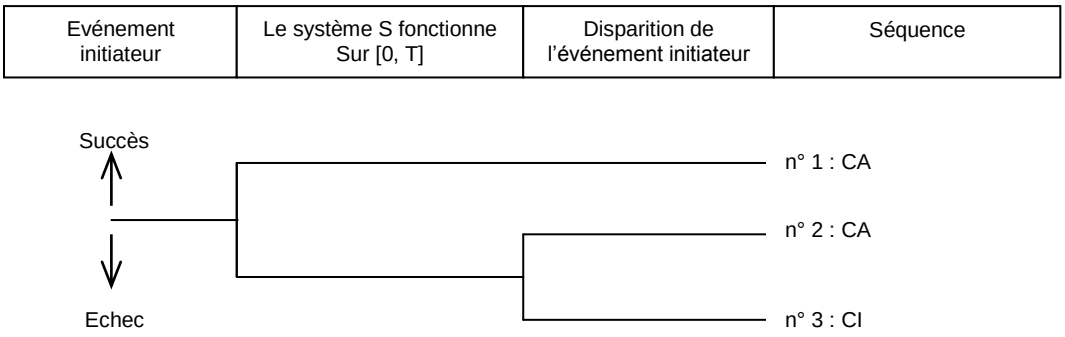


Figure 2.3 : Arbre des Conséquences avec disparition de l'événement initiateur

On suppose que l'événement initiateur peut disparaître ; il correspond par exemple à la perte d'un système élémentaire réparable dont le taux de réparation est μ_i. En cas de défaillance du système *S*, les conséquences inacceptables ne sont atteintes que si l'événement initiateur n'a pas disparu. Calculons la probabilité de la séquence n° 3 avec les hypothèses suivantes :

- l'événement initiateur est indépendant du premier événement générique ;

- le système *S* ne connaît pas de défaillance au démarrage ;
- les taux de défaillance et de réparation sont constants :

$$P[Sq] \approx P[EI] \int_0^T U(t)\, e^{-\mu_i t}\, dt \quad avec \quad P[EI] \prec\prec 1$$

Où *U(t)* est la densité de défaillance du système S $(\Lambda\, e^{-\Lambda t})$ *et* $e^{-\mu_i t}$ la probabilité de non-disparition de l'événement initiateur (ou l'immaintenabilité du système élémentaire associé). Si la condition n'est pas respectée, la formule n'est plus valable ; il faut en effet tenir compte de la possible répétition de l'événement initiateur avant que n'apparaisse la séquence n° 3. La considération de processus semi-markovien permettrait de calculer ce cas. La formule précédente devient :

$$P[Sq] \approx P[EI] \frac{\Lambda}{\Lambda + \mu_i} \left[1 - e^{-(\Lambda + \mu_i) T}\right] \qquad (8)$$

En considérant maintenant que le système *S* est réparable (taux de réparation *M*) et qu'il existe un délai d_c après la perte du système *S* avant d'atteindre les conséquences inacceptables : ce délai est ainsi mis à profit pour tenter de finir de réparer le système *S* ou de faire disparaître l'événement initiateur :

$$P[Sq] \approx P[EI] \int_0^T U(t)\, e^{-M d_c}\, e^{-\mu_i (t + d_c)}\, dt$$

$$P[Sq] \approx P[EI] \frac{\Lambda}{\Lambda + \mu_i} e^{-(M + \mu_i) d_c} \left[1 - e^{-(\Lambda + \mu_i) T}\right] \qquad (9)$$

De tels calculs se généralisent facilement à la perte de plusieurs systèmes élémentaires en séquence en tenant compte de leurs indisponibilités au démarrage.

e) Dépendance de type cause commune et de type séquentiel

Ces dépendances sont les plus difficiles à traiter. Considérons le schéma de la figure 2.4 représentant une dépendance de type « *cause commune* » et de type « *séquentiel* »:

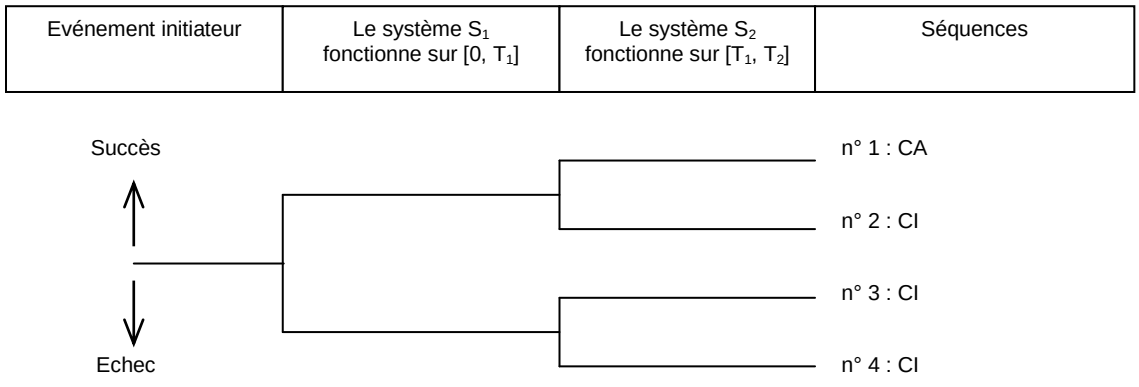


Figure 2.4 : Arbre des conséquences ayant des dépendances de nature « cause commune » et « séquentielle »

On suppose connaître les Arbres des Causes des systèmes S_1 et S_2 ; de plus, il existe des composants, communs aux deux systèmes S_1 et S_2 entraînant l'existence d'événements de cause commune dans les deux arbres. Par contre, il y a indépendance entre l'événement initiateur et chacun des événements génériques.

Appelons S le système constitué par les deux systèmes élémentaires (S_1 et S_2) et abordons le calcul de la probabilité de la séquence n° 1.

$$P[Sq_1] = P[EI] \times P[S \ \ fonctionne \ sur \ [0, T_2]]$$

Le deuxième terme du produit est la fiabilité du système S multi-phases. Elle ne dépend que des indisponibilités absolues et conditionnelles de chaque composant.

Un système multi-phases est un système qui accomplit une mission qui peut être divisée en périodes temporelles consécutives ou *phases* ; il effectue une tâche bien spécifiée. La configuration du système – c'est-à-dire un sous-ensemble de composants et de leurs relations fonctionnelles – change de phase en phase.

2.3 CONCLUSION

La notion d'événements est importante pour notre étude car ce sont les événements qui décrivent la dynamique du système. L'occurrence d'un événement est toujrs initiateur de changement d'état. La présentation de dépendance événementielle est importante car, elle permet de mieux comprendre les notions de propagation de flux et de synchronisation introduites dans le calcul et la modélisation du TRS, tant pour les systèmes série que pour les systèmes parallèles et complexes.

Chapitre III

MÉTHODOLOGIE DE LA MODÉLISATION

3.1 INTRODUCTION

Le modèle est une représentation de la réalité dans un formalisme. Il est développé pour répondre à des questions déterminées et comporte certaines limitations, c'est une abstraction du système réel [TUF 06]. Ce *système réel* n'existe pas forcément lors du processus de modélisation, il se peut que ce soit un système que nous cherchons à développer. Pour cela nous désirons effectuer une étude préliminaire de ses performances.

A partir du modèle, nous voulons obtenir des résultats sur le système étudié, c'est l'étape de résolution. Certains résultats peuvent nous amener à modifier notre vision du système, et nous inciter à calculer de nouveaux résultats sur le modèle. Pour cela, nous pouvons être amenés à *complexifier* le modèle pour rajouter des informations. De même, nous pouvons nous rendre compte que certains résultats ou éléments du modèle sont inutiles pour calculer les indices de performances recherchés, et nous pouvons ainsi parfois *simplifier* le modèle pour faciliter sa résolution. Les résultats obtenus sont comparés avec le comportement du système réel (comportement espéré si le système n'est pas encore créé), le système qui donne les performances optimales peut alors être conçu.

3.2 MÉTHODOLOGIE DE LA MODÉLISATION

3.2.1 Notion de modèle

L'étude d'un SdP en vue de l'évaluation de ses performances conduit à la notion de modèle et de processus de modélisation [MOH 06]. [MIN 68] définit un modèle de la manière suivante :
«*Pour un observateur A, β est un modèle pour B si A peut, à partir de β, apprendre quelque chose d'utile sur le fonctionnement de B* » (figure 3.1).

Une autre définition du modèle donnée par [BEN 03] est la suivante :

« *Le modèle est une représentation de la réalité dans un formalisme, c'est une abstraction d'un système réel qui existe déjà ou n'existe pas* ». [POP 73] présente la notion de modèle par trois concepts relatifs qui sont :

- un modèle doit avoir un caractère de ressemblance avec le système réel ;
- un modèle doit constituer une simplification du système réel ;
- un modèle est une idéalisation du système réel.

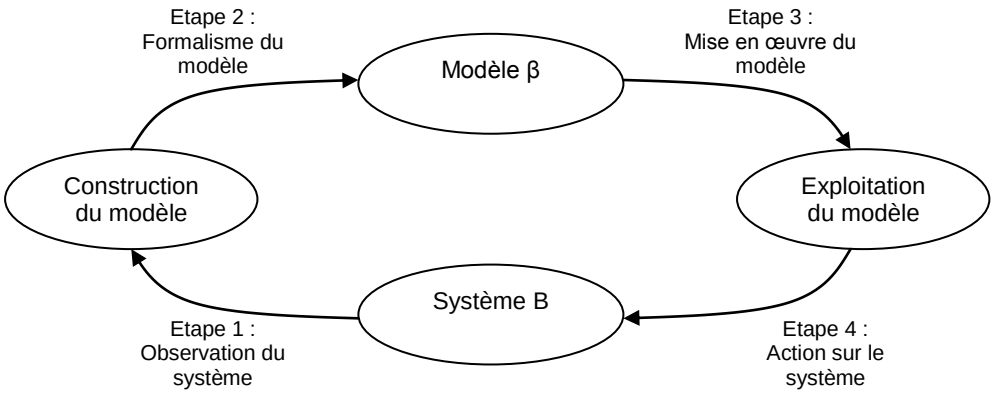


Figure 3.1: Illustration de l'utilisation d'un modèle

En pratique, la construction d'un modèle s'effectue à partir d'observations du système réel à modéliser, mais, elle tient compte également des objectifs que le modèle doit permettre d'atteindre (questions auxquelles on souhaite pouvoir répondre) [DAV 93]. Le degré de finesse d'un modèle est un critère de qualité économiquement important, entraînant ou non des surcoûts en temps et en effort de travail [MOH 06].

A chaque stade de vie du système, des techniques de différentes natures peuvent être utilisées. Ainsi, avant la construction du premier prototype du système, nous sommes amenés à modéliser le système, ou bien à simuler son comportement. Une fois le premier prototype construit, nous pouvons l'observer pour tester les performances du système alors qu'il est soumis à des conditions proches de l'exploitation, mais les résultats sont alors parfois difficiles à interpréter et à extrapoler. Cette approche est utilisée tardivement dans le cycle de vie.

Au niveau des différentes techniques, une première approche informelle, basée sur l'intuition et l'expérience, est nécessaire. Cependant, elle ne se base pas sur une méthodologie et sur des données quantifiées et validées, et des effets secondaires non détectés peuvent apparaître par la suite. Une étude plus systématique et formelle des performances s'avère donc indispensable, par modélisation (début du cycle de vie) puis par observation du comportement (fin du cycle de vie). Si nous utilisons uniquement l'observation, nous serons peut être amené à modifier radicalement un système qui a déjà engendré beaucoup de coûts, car cette technique s'emploie alors que le système a déjà été construit.

a) Modèle de connaissance

Le modèle de connaissance ou de fonctionnement d'un système est une formalisation dans un langage naturel ou graphique de la structure et du fonctionnement de ce système. Si le système existe, le modèle de connaissance contient l'ensemble des connaissances acquises lors de phases d'observation. Si le système n'existe pas, le modèle de connaissance contient les spécifications de topologie et de fonctionnement des concepteurs.

b) Modèle d'action

Le modèle d'action est une traduction du modèle de connaissance dans :

- un formalisme mathématique : par exemple une méthode analytique qui exploite le processus de Markov (PdM) ;
- un langage de programmation ou un système expert: il est directement exploitable sur calculateur et fournit les performances du système modélisé sans recourir à la mesure directe (par exemple un langage de simulation).

L'exploitation du modèle de connaissance et du modèle d'action est appelée « *processus de modélisation* ». Ce processus est généralement itératif (figure 3.2).

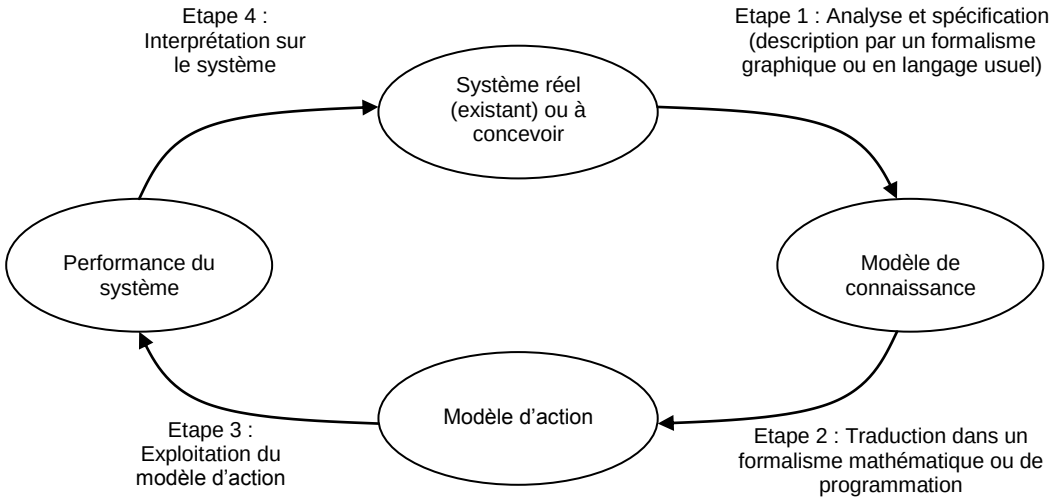


Figure 3.2 : Exploitation du Modèle de Connaissance

3.2.2 Processus de modélisation

Plusieurs techniques de modélisation formelles ont été développées ces dernières années [SAL 06]. Toutes sont basées sur le même principe qui consiste à définir :

– les états du système.
– les transitions entre les états.
– les délais des transitions.

Les états sont représentés graphiquement par des nœuds et les transitions par des flèches étiquetées reliant les nœuds entre eux. Les transitions sont étiquetées par des événements dont l'occurrence engendre une valuation de la transition et un changement d'état du système, après validation de sa garde. Le graphe résultant est connu sous le nom de système de transitions représentant le comportement du système. En supposant que le système étudié a un nombre fini d'états, le changement de l'état du système est engendré par l'exécution d'une action qui déclenche une transition. Mais, il est difficile de réaliser une modélisation manuelle de l'ensemble de toutes les transitions possibles entre les différents états d'un système large et complexe, car souvent le système de transitions sous-jacent d'un système réel est de l'ordre de quelques milliers voire même centaines de milliers d'états. Plusieurs formalismes de haut niveau ont été proposés pour permettre la modélisation d'un système complexe par le biais des formules algébriques compactes, et la dérivation automatique au plus bas niveau, du système de transitions et de la chaîne de Markov sous

jacente, qui ont un très grand nombre d'états et de transitions en utilisant des sémantiques définies en termes de transitions.

Chaque formalisme a sa propre syntaxe et dont la dérivation de la chaîne de Markov n'est autre qu'une application de sa sémantique généralement définie en termes de transitions entre les états. Un logiciel est alors utilisé pour générer l'espace d'états et la matrice « générateur infinitésimal» de la chaîne de Markov, ainsi que pour calculer les solutions stationnaires et transitoires. Quel que soit le formalisme utilisé, l'objectif est le calcul des paramètres de performances du système en question. Certaines méthodes de résolution sont cependant spécifiques à un formalisme, mais des techniques peuvent être adaptées d'un formalisme à un autre.

Le processus de modélisation peut être illustré par la figure 3.3.

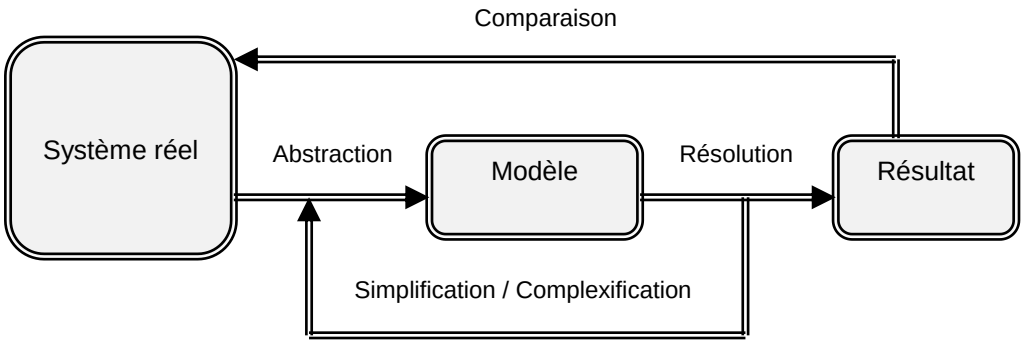


Figure 3.3 : Illustration du Processus de Modélisation

Un grand nombre d'approches a adopté les réseaux de Petri [BAS 00] et [CHA 01] comme formalisme de base pour la modélisation des systèmes dynamiques en ajoutant des extensions qui vont de simples temporisations constantes, jusqu'à des mécanismes beaucoup plus sophistiqués. Les réseaux de Petri temporisés [TOU 91] ont été les premières approches pour l'introduction de la notion de temps dans ce formalisme. Les réseaux de Petri stochastiques (RdPS) [FLO 85] avec leurs nombreux dérivés ont ensuite été définis, comme : Réseaux de Pétri Stochastiques Généralisés [ABB 01] et [AJM 84], Réseaux de Petri Stochastiques et Déterministes, Réseaux de Petri Stochastiques à temps discret, etc. [KHA 97]. Ils permettent une description fonctionnelle et temporelle intégrée dans le même modèle, avec une différence au niveau de la distribution du temps d'exécution. Les réseaux de Petri n'offrent cependant ni la compositionalité ni l'abstraction.

Ces travaux ont permis l'intégration d'outils d'évaluation de performance à l'ensemble des outils d'analyse structurelle développés pour les réseaux de Petri.

Le Calcul des Systèmes de Communication (CCS - *Calculus of Communicating Systems*) a inspiré d'autres approches de modélisation. Ceci est le cas:

– des algèbres des processus stochastiques [GÖT 95];

– des réseaux d'automates stochastiques [ATI 93].

Ces approches n'ont pas la notion d'*entités* et de *flot* (respectivement *clients* et *routage* dans les réseaux de files d'attente ou *jetons* et *arcs* dans les réseaux de Petri). En revanche,

elles offrent une vision compositionnelle de sous-systèmes qui interagissent entre eux (concept de modèles modulaires) [FER 98].

Enfin, des outils algébriques ont été développés sur les graphes afin de décrire le comportement temporel d'événements.

– les algèbres (*max, +*) [ROC 01], [LHO 03] et [NAI 01];

– les algèbres exotiques [CAO 84] et [MOL 87].

L'utilisation de ces méthodes offre une vision très distincte des approches précédentes. Les algèbres (*max, +*) et les algèbres exotiques en général ne sont pas basées sur les chaînes de Markov [STE 94] et [TRI 82]. Les autres formalismes sont en majorité basés sur des hypothèses Markoviennes. Le niveau de détail nécessaire à la description des modèles réduit l'utilisation, au moins pour le moment, de ce formalisme à des cas réels.

Les réseaux de files d'attente [GOU 03] sont un formalisme largement utilisé pour l'évaluation de performance, mais qui est jugé moins expressif à cause du manque de formel. Des algorithmes pour la solution analytique sont extrêmement rapides, et peuvent être utilisés pour évaluer les paramètres de performance du système, en plus de l'évaluation par simulation.

Les Réseaux d'Automates Stochastiques sont un formalisme de haut niveau qui permet la modélisation de chaînes de Markov très grandes et très complexes d'une façon compacte et structurée. Ils ont été construits pour prendre en compte la taille de l'espace d'états lors de la conception d'un modèle. En effet, un système est modélisé à partir de plusieurs composants, qui évoluent souvent en parallèle (d'une façon indépendante) sauf en certaines actions de synchronisation, où les composants interagissent entre eux. Dans ce formalisme, le comportement d'un composant du système est représenté par un ensemble d'états qui constitue un automate (une chaîne de Markov avec un espace d'état raisonnable). Chaque automate est décrit par une matrice locale. L'ensemble de tous les automates ainsi constitué est représenté par une chaîne de Markov multidimensionnelle, dont les états sont ceux de l'espace produit, avec une matrice sous-jacente (appelée descripteur) obtenue en appliquant les opérateurs de l'algèbre tensorielle (ou de Kronecker) sur les matrices locales de chaque automate. La structure de générateur permet d'avoir un gain considérable de l'espace mémoire puisqu'elle évite le stockage de la matrice générateur tout entière. Des techniques de résolution efficaces sont alors disponibles pour l'analyse quantitative. Mais, l'utilisation des composants indépendants reliés par l'intermédiaire des fonctions de synchronisations peut produire un espace d'états avec beaucoup d'états inaccessibles.

La figure 3.4 présente un résumé de la classification présentée.

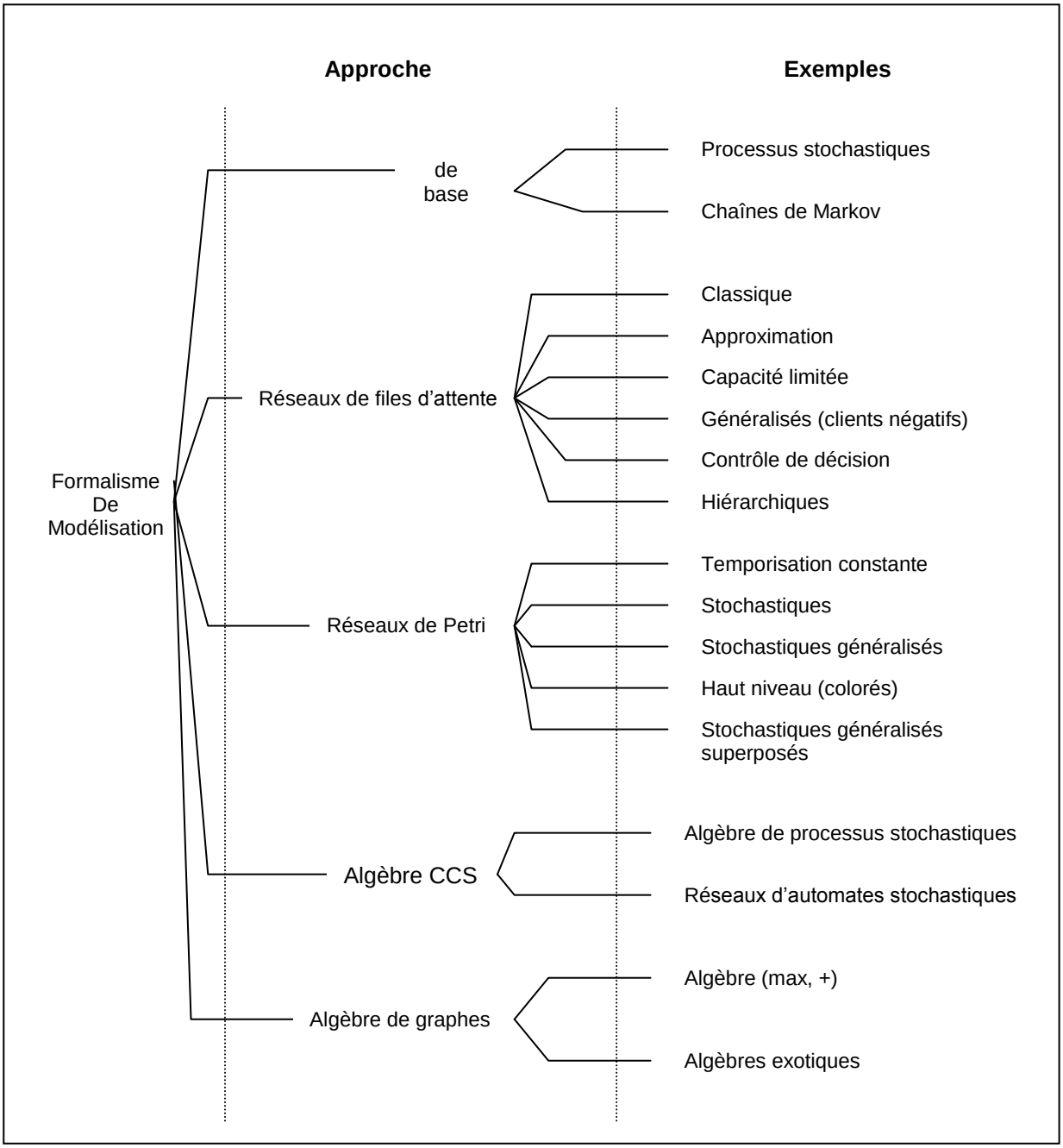


Figure 3.4 : Formalisme de modélisation

3.3 PRINCIPES DE L'ANALYSE PRÉVISIONNELLE

L'analyse d'un système est un processus orienté vers l'acquisition, l'investigation et le traitement ordonnés d'informations spécifiques au système, à son exploitation, et pertinentes vis-à-vis d'une décision ou d'un objectif donné [VIL 88] ; ce processus aboutit à l'obtention d'un modèle du système. Selon [MOH 06], la phase d'analyse est une étude des propriétés du modèle pour en déduire des propriétés associées du système initial. On distingue

habituellement et schématiquement quatre étapes principales dans l'analyse de la Sûreté de Fonctionnement (SdF) d'un système.

3.3.1 Analyse structurelle et fonctionnelle

Cette étape est celle du recueil des premières informations relatives à la structure du système et à ses caractéristiques techniques et fonctionnelles. On cherchera notamment à recueillir les informations relatives aux constituants du système et à sa configuration.

Pour anticiper les problèmes qualitatifs et quantitatifs avant l'implémentation et la fabrication d'un prototype, il est nécessaire de vérifier les comportements fonctionnels et de prévoir les performances du système dans un souci de gain en coût et en temps lors du développement de ce dernier [SAL 06]. Aujourd'hui, des langages de spécifications formelles avec une expressivité temporelle sont mis en œuvre pour fournir un formalisme intégré. Ces derniers offrent différentes méthodes pour la construction de modèles d'un système et pour la réalisation de ces deux types d'analyses, afin de détecter les problèmes potentiels qui peuvent survenir, et répondre aux questions de performance et de surcoût de reconstruction du prototype.

Une première analyse fonctionnelle du système doit aboutir à identifier et à définir les principales fonctions du système. Un découpage structurel SADT permet alors de bien définir ses limites extérieures et intérieures, en fonction des différentes fonctions de chaque sous-système [ZAY 93]. Ce découpage peut d'ailleurs descendre à un niveau plus fin (niveau composant), tout dépend de l'information recherchée. Cette étape est très souvent préliminaire à l'analyse qualitative.

3.3.2 Analyse qualitative

A partir des modèles de type événementiel comme les réseaux de Petri ou les automates d'état, on peut étudier des situations telles que :

- les possibilités de blocage du modèle, généralement révélatrices d'erreurs de conception dans le système réel ;
- certaines propriétés d'atteignabilité pour des états donnés du système ;
- la détermination de certains invariants du modèle qui ont généralement des propriétés physiques intéressantes ;
- la bornitude des états du modèle dont l'absence révèle très souvent une possibilité d'explosion du nombre de certaines entités du système.

Les limites de résolution de l'analyse doivent être précises. Faut-il aller dans l'analyse jusqu'aux composants ? Faut-il aller jusqu'à un niveau de détail plus fin c'est-à-dire jusqu'à des pièces de composants ? Ou alors s'arrêter au niveau des sous-systèmes ou du système lui-même ?

La considération des éléments précédents doit aboutir à la proposition d'une décomposition du système en composants pour l'analyse. Bien évidemment, cette décomposition peut être différente de celle retenue dans la description du système. Il faut, en effet, disposer sur chaque composant d'informations relatives tant aux modes de défaillances et à leurs causes (principale conséquence des temps d'arrêts), qu'aux données de Sûreté de Fonctionnement associées.

Les méthodes de l'analyse quantitative ont surtout pour objectif la recherche de toutes les causes de dysfonctionnement pouvant affecter la Sûreté de Fonctionnement du système, et par conséquent son efficience. De nombreuses méthodes existent et l'art du spécialiste consiste alors à choisir les méthodes les plus adaptées aux objectifs de l'étude, au système à analyser et aux moyens dont il dispose.

L'utilisation des ces méthodes aboutit à une modélisation de la Sûreté de Fonctionnement du système et des dysfonctionnements l'affectant. Cette modélisation est basée sur la décomposition structurelle retenue pour le système et sur un certain nombre d'hypothèses relatives, par exemple, au caractère catalectique des dysfonctionnements, aux phases ou configurations de fonctionnement reconnues a priori importantes.

D'une manière générale, l'analyste est conduit à effectuer un certain nombre d'hypothèses spécifiques à chaque étude. Ce sera le cas entre autres, pour l'impact des facteurs diminuant graduellement la performance du système telles que l'usure et les réparations répétées, de l'environnement ou des autres systèmes sur le système étudié, les politiques de maintenance, le comportement de l'Opérateur et du Réparateur en situation normale, incidentelle ou accidentelle…

3.3.3 Analyse quantitative ou évaluation des performances

La phase d'analyse quantitative consiste à caractériser par les mesures (probabilités, par exemple) la Sûreté de Fonctionnement du système. Ces probabilités sont obtenues par le traitement mathématique du modèle et par la prise en compte des données relatives aux événements élémentaires [VIL 88]. Ce sont des méthodes qui permettent de connaître a priori le comportement d'un système de production pour une configuration donnée [MOH 06]. Le comportement est défini par les performances calculées à partir des caractéristiques du système de production.

Evaluer signifie « *déterminer une quantité par le calcul sans recourir à la mesure directe* ». L'évaluation est donc toujours effectuée à l'aide d'un modèle.

Dans la phase de conception, le problème du dimensionnement (choix du nombre de machines, de la configuration du système, de la capacité de stocks, des modes de marche et d'arrêts…) est un point crucial pour une évaluation de performance. En effet, l'atelier a un objectif qui est de réaliser une certaine production :

- s'il est sous dimensionné, il ne remplit pas sa fonction ;
- s'il est sur dimensionné, il y a gaspillage.

Pour résoudre ce problème de dimensionnement, on passe par la résolution d'un problème plus simple qui est l'évaluation de performance du système pour une configuration donnée. En répétant l'opération sur des configurations différentes du système, on peut espérer trouver celle qui constitue le meilleur compromis entre productivité et investissement.

Dans la phase d'exploitation, il faut en assurer l'efficience, la Disponibilité et tous les autres paramètres de la Sûreté de Fonctionnement. Là aussi l'évaluation de performance est un élément important pour la prise de certaines décisions. La performance du système devient sujette à ses changements d'état.

En se basant sur les deux cas précédemment cités, on distingue deux approches pour la modélisation des systèmes complexes:

- La première approche ou *modélisation à priori* est utilisée pour la conception des systèmes qui n'existent pas encore physiquement, afin d'en déterminer les principales caractéristiques.
- La deuxième approche ou *modélisation à posteriori* consiste à améliorer, à adapter ou à modifier les systèmes existants.

3.3.4 Les méthodes de prédiction de performance

Le calcul de l'état stationnaire d'un modèle, c'est à dire la proportion de temps que la chaîne de Markov reste dans chacun des états sur une trajectoire de durée infinie (théorème ergodique), est exprimé par un vecteur de probabilité associant une probabilité à chaque état de la chaîne [HEY 82]. À partir de ce vecteur nous pouvons calculer plusieurs informations sur le système modélisé, par exemple le nombre moyen de tâches traitées, les délais moyens, etc.
Cette résolution correspond à l'obtention du vecteur solution du système:

$$\pi Q = 0 \qquad \|\pi\|_2 = 1 \tag{10}$$

Où Q est une matrice décrivant le processus de Markov appelée matrice de transition ou générateur infinitésimal.

Parmi les méthodes de résolution nous pouvons faire la distinction entre:
- les méthodes analytiques [BAS 75], [KLE 75] et [REI 80];
– les méthodes numériques [SAA 95] et [STE 94] ;
– les simulations [ROB 90], [YCA 97], [HOU 03], [MOR 02] et [PLA 97].

a) Les méthodes analytiques

Les méthodes analytiques sont les méthodes qui donnent une solution sans passer par la résolution numérique du système linéaire $\pi Q = 0$. Elles sont très efficaces et beaucoup plus rapide que les simulations [FER 98] et [SAL 06]. Elles sont purement structurelles. Ces méthodes ont l'avantage d'éviter la résolution du système linéaire, système généralement très grand. Cependant, leurs hypothèses sont très restrictives, elles nécessitent généralement de simplifier le fonctionnement du système pour leur mise en œuvre.

Leur principe est de modéliser un SdP par un certain nombre d'équations. Les performances du système sont alors calculées en résolvant ce système d'équations, mais la solution n'existe que pour une classe très restreinte des systèmes. Parmi ces méthodes, nous pouvons citer : les processus de naissance et de mort, la forme produit…

b) Les méthodes numériques

Les méthodes numériques servent de modèles abstraits à partir desquels des résultats exacts peuvent être obtenus, tout en résolvant un ensemble d'équations linéaires dérivées de la chaîne du Markov sous-jacente du modèle [SAL 06]. Ces méthodes offrent un compromis

entre la simulation et les méthodes analytiques. Elles sont beaucoup plus précises et dans certains cas beaucoup plus rapides que la simulation, mais moins rapides que les méthodes analytiques.

Les méthodes numériques peuvent être partagées en trois groupes:
– les méthodes numériques directes;
– les méthodes numériques itératives avec analyse structurelle;
– les méthodes numériques itératives pures.

Il faut néanmoins tenir compte du fait que le générateur infinitésimal d'une chaîne de Markov est une matrice possédant des caractéristiques particulières, favorisant l'application de certaines méthodes mais empêchant l'utilisation d'autres [FER 98]. Les méthodes numériques de résolution de systèmes linéaires applicables aux chaînes de Markov sont généralement des méthodes itératives. Les méthodes directes, comme la méthode de Gauss, ne sont pas utilisables pour des modèles de grande taille (nombre d'états) [LAS 94].

Étant donné la grande taille des chaînes de Markov à résoudre, il est important de faciliter l'application des méthodes de résolution de systèmes linéaires. Un très grand nombre de solutions sont disponibles dans la littérature. Elles peuvent être divisées en deux grands groupes (qui ne sont pas forcément exclusifs):

– les méthodes qui font des analyses structurelles du générateur Q avant de résoudre le système [DAL 94], [FOU 95], [MOR 96], [NEU 81] et [NEU 89] ;
– les méthodes qui ne font que des analyses numériques pour tirer profit du type de stockage utilisé pour le générateur [BUC 94], [DON 93] et [PLA 91].

Si pour des raisons de coût (généralement dû à la taille du problème) il n'est pas possible d'obtenir une solution exacte, il existe une solution alternative: la recherche de bornes [FER 98].

Le calcul de bornes de performance [COU 77 et 94] est basé sur des simplifications du générateur en déterminant une solution aussi proche que possible de la solution exacte pour une partie pertinente du modèle. Ce genre de résolution peut être fort intéressant lors de problèmes très grands (éventuellement même infinis) [ABU 96] et [MAH 97]. Cette résolution approchée est généralement employée en partant d'analyses structurelles de la chaîne de Markov [FOU 95], [MOK 97] et [MOR 96]. La figure 3.5 présente un résumé de la classification présentée.

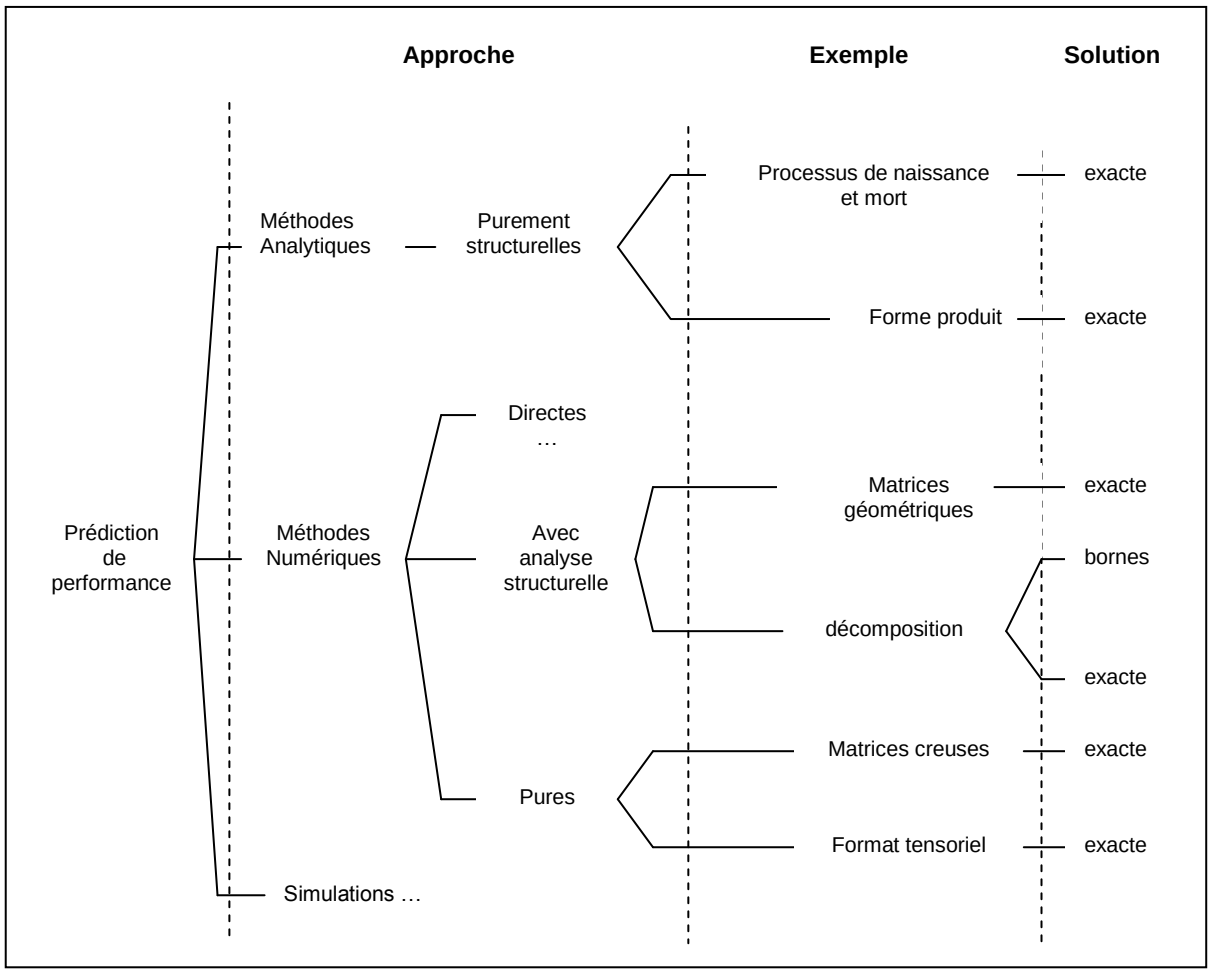


Figure 3.5 : Méthodes d'obtention de performances

c) Les simulations

La simulation permet d'imiter le comportement d'un système en traitant des modèles généralement plus proches de la réalité que les méthodes formelles, et elle sert souvent à valider les modèles de ces derniers [FER 98] et [SAL 06]. Elle présente moins d'hypothèses [TUF 06], et mesure le comportement d'un modèle du système en simulant son exécution, par exemple par une génération aléatoire d'événements répartis de manière discrète dans le temps. Elle recouvre aujourd'hui des enjeux considérables dans tous les domaines d'ingénierie (technique, commerciale, financière…) car elle constitue une aide indispensable à la décision et à la maîtrise des incertitudes [CAB 01]. La simulation à événements discrets est généralement utilisée pour les modèles stochastiques sans aucune restriction temporelle (distribution générale ou non-Markovienne), où les probabilités d'exécution des actions sont remplacées par des échantillons des lois de distributions associées [OUA 01]. Elle possède un domaine d'application quasi-illimité mais reste économiquement très coûteuse (temps humain et machine) [CHE 03]. En plus, la simulation a l'avantage d'être insensible à la taille de l'espace d'état. Mais, l'utilisation directe de la simulation est difficile et sa complexité temporelle croit avec les détails investis dans le modèle et le degré de précision (ou

l'intervalle de confiance) requis. Elle ne repose sur aucune théorie mathématique, il convient donc d'être prudent lors de son utilisation (qualité des générateurs de nombres pseudo-aléatoire, validité des noyaux de synchronisation,…) au risque d'obtenir des résultats absurdes.

Cette technique permet donc de simuler les systèmes existants ou non et ceci pour une charge donnée (études de fonctionnement transitoires, tests de manière discrète dans le temps) est opposée à la simulation continue (le système change continuellement d'états). Pour faciliter la mise en place d'une simulation, il sera plus judicieux de construire un modèle formel pour s'assurer du bon fonctionnement du modèle avant sa mise en œuvre dans le simulateur.

De toutes ces méthodes d'évaluation de performance, nous avons utilisé la simulation pour l'évaluation des modèles d'automates d'états du système en fonction des temps d'états définis d'après la norme NFE 60-182, et de la valeur du TRS. La simulation stochastique de ces modèles d'états, malgré un espace d'état considérable pour un système complexe, a été possible grâce à la grande performance qu'offre le langage de modélisation AltaRica Data-flow.

3.4 CONCLUSION

La résolution des problèmes d'évaluation de performance des systèmes nécessite que l'on connaisse sa structure, ses fonctions (secondaires et principales), les différents éléments qui le composent etc. Les méthodes de résolution adoptées sont fonction de la complexité du système. Le modèle est une représentation de la réalité dans un formalisme. Il est développé pour répondre à des questions déterminées et comporte certaines limitations, c'est une abstraction du système réel. Ce *système réel* n'existe pas forcément lors du processus de modélisation, il se peut que ce soit un système que nous cherchons à développer. Pour cela nous désirons effectuer une étude préliminaire de ses performances. A partir du modèle, nous voulons obtenir des résultats sur le système étudié, c'est l'étape de résolution. Certains résultats peuvent nous amener à modifier notre vision du système, et nous inciter à calculer de nouveaux résultats sur le modèle. Pour cela, nous pouvons être amenés à *complexifier* le modèle pour rajouter des informations. De même, nous pouvons nous rendre compte que certains résultats ou éléments du modèle sont inutiles pour calculer les indices de performances recherchés, et nous pouvons ainsi parfois *simplifier* le modèle pour faciliter sa résolution. Parmi les méthodes de prédiction de performance, les simulations semblent mieux adaptées car elles représentent une abstraction du système réel, et sont peu sensible à la taille de l'espace d'état du système.

BILAN DE L'ÉTAT DE L'ART

En résumé, nous pouvons retenir au travers de ce chapitre que les systèmes de production sont assez complexes, car influencés à la fois par les comportements humains, les aspects organisationnel et technique, l'environnement, les limites du système etc. La structure du système, sa nature et ses fonctions sont autant d'éléments dont il faut avoir connaissance avant d'envisager toute procédure de modélisation et d'évaluation de performance. Evaluer signifie « *déterminer une quantité par le calcul sans recourir à la mesure directe* ». L'évaluation de performance est donc toujours effectuée à l'aide d'un modèle. Elle permet de résoudre le problème de dimensionnement à la conception (choix du nombre de machines, de la configuration du système, des modes de marche et d'arrêts etc.). Dans la phase d'exploitation, il faut en assurer l'efficience, la disponibilité et tous les autres paramètres de la Sûreté de Fonctionnement. Là aussi l'évaluation de performance est un élément important pour la prise de certaines décisions.

Les systèmes de production complexes sont caractérisés par un espace d'états considérable, et une explosion combinatoire, ce qui rend difficile leur résolution par des méthodes classiques. Parmi les méthodes de résolution des systèmes à grand espace d'état, nous pouvons retenir les trois plus importantes :

- les méthodes analytiques;
- les méthodes numériques;
- les simulations.

Les méthodes analytiques sont les méthodes qui donnent une solution sans passer par la résolution numérique du système linéaire $\pi Q = 0$. Elles sont très efficaces et beaucoup plus rapide que les simulations. Elles sont purement structurelles. Ces méthodes ont l'avantage d'éviter la résolution du système linéaire, système généralement très grand. Cependant, leurs hypothèses sont très restrictives, elles nécessitent généralement de simplifier le fonctionnement du système pour leur mise en œuvre. Leur principe est de modéliser un système de production par un certain nombre d'équations. Les performances du système sont alors calculées en résolvant ce système d'équations, mais la solution n'existe que pour une classe très restreinte des systèmes. Parmi ces méthodes, nous pouvons citer : les processus de naissance et de mort, la forme produit…

Les méthodes numériques servent de modèles abstraits à partir desquels des résultats exacts peuvent être obtenus, tout en résolvant un ensemble d'équations linéaires dérivées de la chaîne du Markov sous-jacente du modèle. Ces méthodes offrent un compromis entre la simulation et les méthodes analytiques. Elles sont beaucoup plus précises et dans certains cas beaucoup plus rapides que la simulation, mais moins rapides que les méthodes analytiques.

La simulation permet d'imiter le comportement d'un système en traitant des modèles généralement plus proches de la réalité que les méthodes formelles, et elle sert souvent à valider les modèles de ces derniers. Elle mesure le comportement d'un modèle du système en simulant son exécution, par exemple par une génération aléatoire d'événements répartis de manière discrète dans le temps. En plus, la simulation a l'avantage d'être insensible à la taille de l'espace d'état. Pour faciliter la mise en place d'une simulation, il sera plus judicieux de construire un modèle formel pour s'assurer du bon fonctionnement du modèle. Un langage de type formel et structurel tel AltaRica Data-Flow présenté en annexe A convient mieux à cet exercice.

Nous proposons donc dans la deuxième partie de notre travail, une méthode d'évaluation de performance des systèmes de production basée sur la modélisation de tous ses éléments constitutifs. Notre méthode utilise un indicateur de performance type TRS, par modélisation par automate d'état de ses composantes (Qualité, Performance et Disponibilité opérationnelle). Les modèles automates ainsi constitués sont traduits en AltaRica Data-Flow. Une simulation stochastique sur une mission donnée permet alors d'obtenir la valeur de l'indicateur calculé. Le résultat obtenu est comparé aux valeurs seuils définies par la World Class Performance. Les formules de calcul utilisées sont proposés par approche temporelle basée sur le découpage des temps d'état d'un moyen de production, norme NFE 60-182. Nous avons modélisé des systèmes simples, séries, parallèles avec redondance chaude et froide. Dans le cas des redondances froides, nous avons considéré les systèmes sans redémarrage et avec redémarrage. L'intégration des comportements de l'Opérateur et du Réparateur dans les modèles AltaRica Data-Flow a permis de mettre en évidence les politiques de maintenance utilisées dans les entreprises.

Notre Contribution

INTRODUCTION

Cette deuxième partie présente notre contribution. Elle regroupe les chapitres 4, 5 et 6.

Le chapitre 4 présente les procédures de calcul du TRS. Il s'agit ici des formules mathématiques permettant de calculer cet indicateur de performance en tenant compte de la configuration du système. Dans ce chapitre, nous présentons la norme NFE 60-182 qui répartit les différents temps d'état d'un moyen de production. Les formules utilisées seront basées sur ces temps d'état. À la suite du chapitre, nous présentons les formules de calcul du TRS en tenant compte de l'aspect productivité c'est-à-dire basé sur le nombre de produit réalisé. La dernière partie décrit le calcul du TRS sous le seul aspect Sûreté de Fonctionnement, c'est-à-dire en ne tenant compte que des comportements fonctionnel ou dysfonctionnel des machines. Nous avons également proposé dans cette partie des formules de calcul du TRS global des systèmes complexes (série, parallèle, assemblage et expansion).

Le chapitre 5 présente la modélisation temporelle et stochastique du TRS. Il s'agit de présenter le TRS et ses composantes sous forme d'automate d'état. Ces automates donnent une description de la dynamique de la variation du TRS d'un mode à l'autre. En fonction de la valeur du TRS, nous avons défini trois modes dans lesquels peut se trouver un système : le mode marche (m) si la valeur du TRS est $\geq 85\%$, marche dégradé (md) si $25\% \geq TRS \geq 85\%$, et hors service (os) si $TRS \leq 25\%$. Ainsi, nous avons modélisé l'efficience d'un système en fonction de la Disponibilité opérationnelle, en fonction du taux de Qualité et du taux de Performance. Le TRS global est également modélisé en prenant en compte tous les événements permettant de passer d'un état du système à l'autre. Un modèle automate d'état permettant de décrire les comportements de l'Opérateur et du Réparateur est également donné. Dans les modèles stochastiques, nous ressortons la dynamique à l'intérieur de chaque état par des modèles Coxians.

Le chapitre 6 qui est le dernier permet maintenant de faire une modélisation AltaRica Data-Flow de l'efficience d'un système de production. Il s'agit en fait de traduire en AltaRica Data-Flow, tous les modèles automates d'états décrivant la dynamique du TRS et de ses composantes. La simulation stochastique de chaque modèle ainsi traduit permet de calculer la valeur du TRS en fonction des taux de transition caractérisant chaque événement. La valeur obtenue permet alors d'évaluer l'efficience du système en tenant compte des valeurs de la World Class Performance.

Chapitre IV

PROCÉDURES DE CALCUL DU TRS

4.1 INTRODUCTION

L'optimisation des processus de production réside dans la bonne appréciation des indicateurs de performance. Le TRS (encore appelé en anglais «*Overall Equipment Effectiveness* OEE ») [HUA 03] est devenu au travers de la norme NFE 60 – 182, l'un des indicateurs de performance des systèmes de production [KOM 06]. Il a été reconnu comme une méthode fondamentale pour mesurer la performance d'un équipement. Son étude commence vers la fin des années 1980. Actuellement, il est reconnu comme outil fondamental d'évaluation de performance des systèmes de production. L'utilisation du TRS et l'établissement d'un matériel de performance discipliné qui donne une lecture du système aideront à se concentrer sur les paramètres critiques à son succès. Analyser les catégories de TRS peut révéler les plus grandes limites à succès. Le TRS rend compte de l'utilisation effective d'un moyen de production, et permet d'identifier les pertes. Il est l'un des indicateurs important pour la prise des décisions [AYE 03a]. Il permet de mieux comprendre et d'optimiser les flux de production, de se doter d'outils d'aide à la décision dynamiques et appropriés, d'organiser une politique de maintenance adaptée et de sauvegarder les savoir-faire, ce qui permet de pérenniser la valeur ajoutée de l'entreprise. Il se présente comme un puissant outil d'investigation [AYE 03b].

La connaissance de cet indicateur est importante pour l'appréciation de l'incompatibilité *coûts - délais – qualité*. La qualité d'un moyen de production s'obtient non seulement par une meilleure fiabilité des décisions, la transparence des données, la diminution des temps de réaction, mais aussi et surtout par un contrôle efficace des dysfonctionnements du système dus aux facteurs diminuant graduellement l'efficience telles que les réparations répétées et l'usure, ainsi que par l'application d'une bonne politique de maintenance. La meilleure productivité quant à elle n'est obtenue que s'il y a amélioration des moyens de production que sont : un taux d'utilisation optimal conditionnée par la disponibilité des ressources, une réduction des délais de fabrication, une amélioration de la qualité des produits et des services conditionnée par la performance des équipements, une bonne flexibilité et une amélioration de la rentabilité.

Le TRS mesure la performance d'un moyen de production. Il permet d'identifier les pertes, il représente un excellent outil d'investigation. Il est la „température"" du moyen de production. Mais pour progresser, savoir ne suffit pas, il faut comprendre. C'est pourquoi, on associe toujours à la mesure du TRS, un recueil détaillé et factuel des causes de non rendement synthétique [GEN 08]. Les causes serviront à déterminer les temps d'état qui servent de base au calcul des indicateurs. Afin de permettre une meilleure évaluation de performance des systèmes de production, nous avons utilisé comme procédure de calcul du TRS, deux grands aspects : d'une part, l'aspect productivité qui le défini comme un rapport du nombre de produits bons sur le nombre de produit théoriquement réalisables. D'autre part, nous avons considéré l'aspect Sûreté de Fonctionnement, qui permet de définir le TRS comme un rapport du temps utile (temps pendant lequel on fabrique les produits bons) sur le

temps requis (temps pendant lequel on aurait théoriquement réalisé un certain nombre de produits). Cette deuxième approche utilise des taux (taux de Performance, taux de Qualité et Disponibilité Opérationnelle), lesquels serviront de données pour la modélisation de l'efficience des systèmes de production par automates d'états dans le chapitre 5.

4.2 LE TRS : INDICATEUR DE L'EFFICIENCE DES SYSTÈMES DE PRODUCTION

4.2.1 Les qualités requises pour un indicateur pertinent

Un indicateur doit être techniquement et conceptuellement apte à mesurer, avec une précision acceptable, le phénomène qu'il est censé mesurer et demeurer pertinent [BOU 07].

Les qualités nécessaires pour y parvenir sont de 3 ordres :

1. *les qualités d'usage* : Il s'agit ici d'un ensemble de qualités de bon sens qu'il n'est pas toujours évident de réunir toutes ensemble, dès lors que le champ à instrumenter intègre des aspects humains, ce qui est le cas très général de la qualité ;
2. *les qualités métrologiques* : un indicateur est un instrument de mesure, il doit satisfaire aux qualités requises de tout instrument de mesure ;
3. *les qualités systémiques* : l'indicateur s'insère dans un ensemble visant à assurer une conduite du système qualité dans la bonne direction (celle de la politique définie).

a) Les qualités d'usage

*** Simplicité :** La politique doit être comprise (norme). Il faut donc que l'indicateur soit d'interprétation simple, du point de vue des acteurs chargés de le produire et responsables de son niveau.

- On peut être amené à instrumenter deux grands types de champs :
  - *un champ propre aux activités opérationnelles concernant le métier de l'entreprise* : La qualité de la production, l'optimisation du processus, le bon usage des matières, tous phénomènes de nature non financière ;
  - *un champ proprement financier concernant l'équilibre économique de l'entreprise*: la définition des enjeux pour fixer les priorités puis apprécier l'efficacité économique des programmes d'amélioration.
- Nous pouvons fixer comme principe général qu'il faut préférer, chaque fois que cela ne comporte pas de contre-indication, une mesure dans l'unité native de la donnée, c'est-à-dire naturelle du point de vue du champ instrumenté. Préférer la mesure des phénomènes non financiers par des unités non financières et vice-versa.

Deux avantages :

- la qualité de la mesure est meilleure puisque la donnée initiale n'a pas à être convertie dans une autre unité ;
- elle est directement "parlante" pour les hommes qui sont dans ce champ puisqu'elle correspond à leur langage habituel.

*** Représentativité***:* Pour être correctement représentatif de l'objectif dont il mesure la performance ou de l'action à piloter, le bon indicateur doit rassembler simultanément 3 qualités :

- *quantification* : c'est la définition même d'un indicateur ;

- *exhaustivité* : une représentation complète de l'objectif ou de l'avancement de l'action ;

- *objectivité* : être exempt de conventions de calcul pouvant faire l'objet de débat.

Chaque fois que cela a un sens et ne présente pas d'inconvénients, il est préférable de construire les indicateurs de telle façon qu'ils augmentent quand la situation s'améliore.
Il vaut mieux calculer un taux de qualité qu'un taux de rebut. Pourquoi ?

- parce que ce qui monte est jugé intuitivement de manière positive;

- parce que la combinaison d'indicateurs entre eux est plus simple et directe si tous les indicateurs constituants sont construits dans le même sens;

- parce qu'un repérage réflexe des indicateurs à surveiller sera plus commode si leur représentation graphique est homogène.

Illustrons les 2 derniers points par l'exemple de l'efficacité d'˝un moyen de production (figure 4.1). Les facteurs de succès sont les temps d'˝état. Nous disposons d'un but et de 3 facteurs clés de succès. Intuitivement, on perçoit que pour chacun de ces 3 facteurs, le paramètre mesurable idéal est le temps perdu du fait de l'insuccès de ce point de vue. Mais comment chiffrer ?

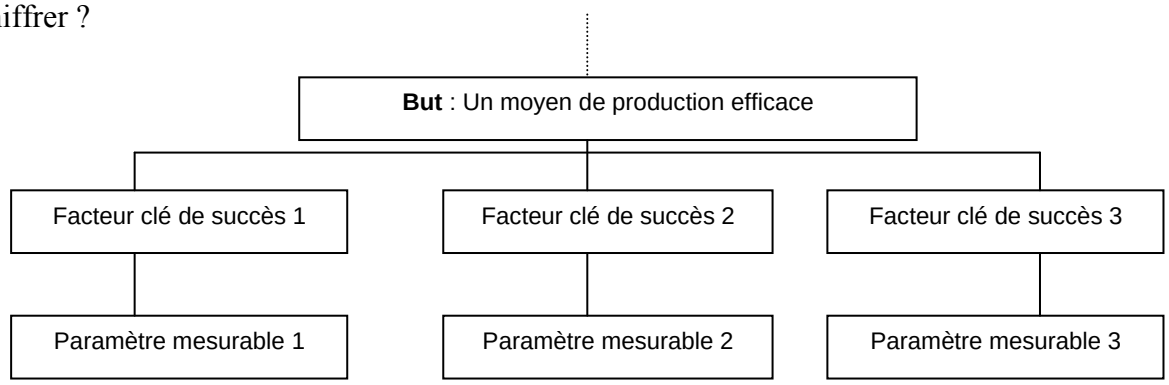


Figure 4.1 : Illustration graphique de quelques facteurs clés du succès

Clarifions la question en analysant les temps d'˝état. Ils peuvent se décomposer comme suit :

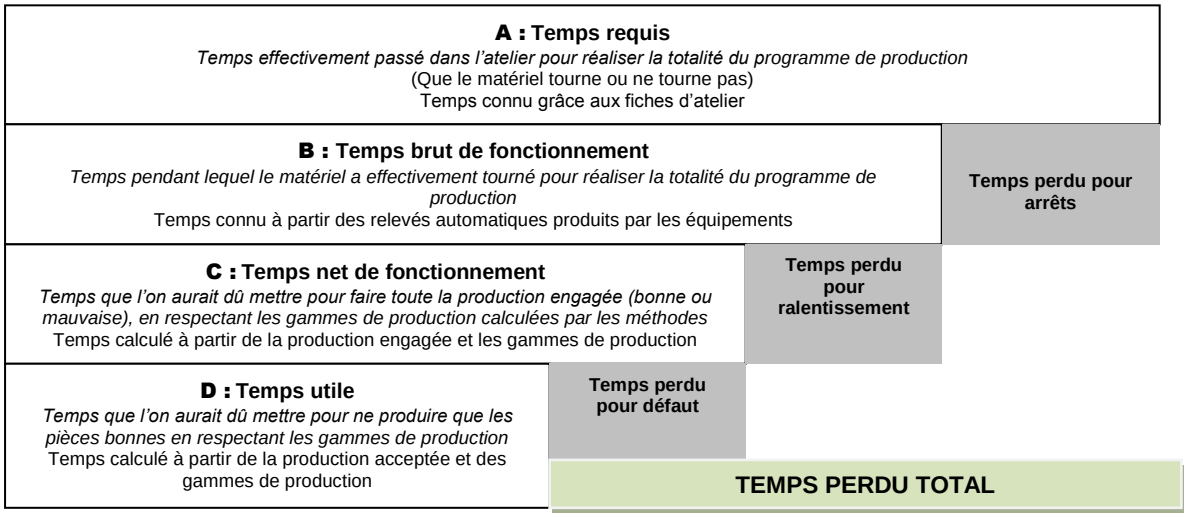


Figure 4.2 : Illustration des temps perdus

Ces 4 niveaux de temps étant connus, la figure 4.1 peut être complétée par les indicateurs :

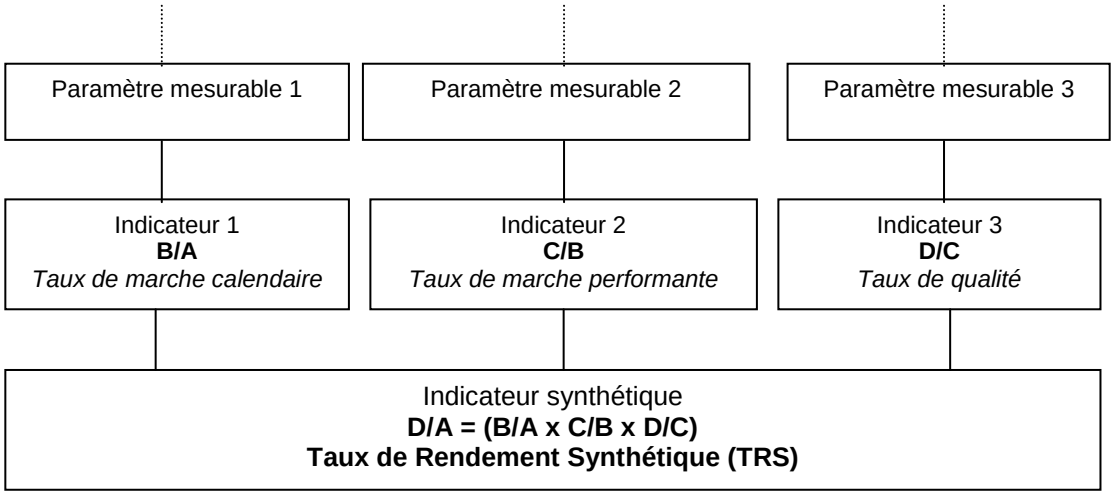


Figure 4.3 : Illustration de quelques indicateurs de performance

On peut remarquer qu'une augmentation de chacun des indicateurs entraîne l'amélioration de la performance associée. La combinaison des trois indicateurs permet également d'avoir une vision synthétique de la performance du système. Chaque indicateur partiel permet à chaque partie prenante du système de production d'apprécier sa performance. Le dernier indicateur pourra faire partie du tableau de bord du Directeur de production, par exemple, le deuxième, celui du Directeur technique [BOU 07].

*** Opérationnalité :** Un indicateur a pour vocation *d'orienter* l'action (indicateurs de performance) ou d'*ajuster* (indicateurs de pilotage). Pour cela, il faut que l'information qu'il fournit à l'instant *t* soit valide au moment où l'action sera conduite. On parle de décision en temps réel. Le temps réel peut se définir comme la mise en concordance de deux fenêtres de temps : celle pendant laquelle l'information est valide et celle pendant laquelle la décision est possible.

b) Les qualités métrologiques

Un indicateur est un moyen de mesure. Il doit donc satisfaire aux conditions requises d'un bon instrument de mesure, à savoir :

- *la justesse* : c'est la capacité à donner la valeur vraie (dans une fourchette de précision déterminée) ;
- *la fidélité, constance* ou *reproductibilité* : c'est la capacité de l'instrument à donner la même mesure lorsqu'il est mis dans des situations identiques ;
- *la précision* : c'est la fourchette contrôlée à l'intérieur de laquelle se trouve la valeur mesurée.

On remarquera que la justesse inclut la fidélité.

*** Spécificité métrologique des indicateurs**

Les instruments de mesure courants ont été construits spécifiquement pour un usage déterminé et en fonction de cet usage, ils sont parfaitement adaptés par définition à ces mesures. On retrouve cette perfection dans des indicateurs élémentaires, tels que le nombre de pièces produites par une machine munie d'un compteur. On ne la retrouve pas souvent dans les indicateurs qualité car ils sont la plupart du temps construits à partir d'un "détournement" de données. Aussi, faut-il juger les qualités de l'indicateur en fonction du champ qu'il prétend éclairer (le point clé dont il prétend mesurer le degré de maîtrise) et bien avoir en tête les hypothèses souvent implicites qui garantissent sa validité.

c) Les qualités systémiques

Les indicateurs ont vocation à mesurer la performance (pour les uns) et à piloter (pour les autres), le système qualité de l'entreprise. L'ensemble des indicateurs doit satisfaire 3 qualités.

- *Pertinence* : C'est leur capacité à mesurer l'efficacité de la politique qualité par rapport à l'environnement, c'est-à-dire leur aptitude à "satisfaire les exigences des clients… et autres parties intéressées". Au niveau de la politique générale, il s'agira du client externe, à des niveaux intermédiaires ou opératoires, il pourra s'agir d'un client interne. Des indicateurs pertinents se définissent en identifiant bien quels sont tous les "clients" de l'entité mise sous contrôle.
- *Cohérence* : C'est l'assurance qu'une amélioration constatée sur tel indicateur contribue bien à une amélioration à un niveau plus général et au final à une amélioration du point de vue de la politique qualité dans son ensemble. Des indicateurs cohérents signifient un déploiement efficace de la politique qualité à travers la structure.

- *Convergence* : C'est l'assurance qu'une amélioration constatée sur tel indicateur ne s'est pas faite au détriment de la performance d'une autre entité de même niveau. Les problèmes de convergence sont fréquents entre services d'approvisionnement et services de production (une minimisation exagérée des stocks peut entraîner des difficultés de fonctionnement en production) et entre services de production et services commerciaux (une personnalisation excessive des offres peut entraîner d'autres difficultés de production).

4.2.2 Le TRS : indicateur pertinent de performance

La réelle valeur ajoutée de l'entreprise est le « processus de fabrication » qui intègre les différentes étapes mettant en œuvre les ressources humaines et matérielles. Les nouveaux défis liés à l'environnement exigent des entreprises, une capacité constante d'innovations, un niveau toujours plus haut de la qualité, la sauvegarde des marques. Tout ceci nécessite d'optimiser les processus de production. Optimiser c'est tout d'abord, mesurer avant d'agir ensuite sur les points critiques. Les besoins en optimisation sont multiples et recouvrent des enjeux souvent considérables [CAB 99c] et [CAB 00b] . Dans le domaine de la Sûreté de Fonctionnement, cette optimisation cherche à améliorer la robustesse des produits aux aléas (défaillances, erreurs humaines…). Pour un système de production, il faut mettre en place des indicateurs liés au bilan matière, au bilan production, en fonction des temps d'état du système. La maîtrise de la relation a*ction – phénomène – état* permet d'avoir une vision transversale temps réel de la production et de prendre des mesures conséquentes pour la réduction des facteurs diminuant graduellement l'efficience telles que les réparations répétées, et l'usure.

Dès lors, les actions à mettre en place apparaissent comme évidentes et elles impactent l'ensemble du système de production, personnel, machine et processus. L'action sur les processus est délicate, car c'est le processus qui donne un sens à la production. Une machine devient une ressource à partir du moment où le processus qui la mobilise la met en relation avec d'autres ressources (personnel, ou autre machine…). De plus, le processus capitalise le savoir-faire, une véritable valeur ajoutée de l'entreprise. Ainsi, assurer la traçabilité du processus par poursuite des indicateurs de performance tel le TRS, permet de garantir la haute qualité et la compétitivité d'un système de production [KOM 06].

Les principales fonctionnalités qui permettent de caractériser la performance d'un système de production sont : l'ordonnancement, la gestion du personnel, la gestion des ressources, le cheminement des produits et des lots, l'acquisition des données, le contrôle de la qualité, la gestion des procédés, l'analyse des performances, la gestion des documents, la gestion de la maintenance. On remarque tout de même que certaines d'entre elles sont directement opératoires, comme l'ordonnancement ou le contrôle de la qualité, tandis que d'autres apparaissent plutôt comme des fonctions transversales, c'est le cas de la gestion des ressources.

Un indicateur va servir à réagir pour réguler l'action dans le sens des orientations fournies par la politique et les objectifs qualité. Il faut donc trouver la meilleure cohérence possible entre les objectifs et les indicateurs pour que cette réaction soit opportune. Le choix des indicateurs constitue une "instrumentation", une quantification, des objectifs pour rendre la mise sous contrôle (au sens de la gestion) efficace. Pour ce faire, deux orientations sont possibles :

- dans le sens de l'action, il s'agit de définir puis de piloter les plans d'action qualité permettant d'assurer la maîtrise des facteurs clés de succès ;
- dans le sens du contrôle de l'action, il s'agit d'aboutir à la définition des indicateurs pertinents permettant d'assurer que l'action mène bien vers l'objectif arrêté par le cahier des charges.

Le TRS est un indicateur de performance qui regroupe toutes ces qualités requises. Du point de vue productivité, le TRS intègre un indicateur de qualité : le Taux de Qualité. Cet indicateur est d'ailleurs le plus exigible d'après la World Class Performance qui le situe à 99%. Du point de vue Sûreté de Fonctionnement, le TRS reste pertinent car il intègre la Disponibilité Opérationnelle qui prend en compte, non seulement la disponibilité des machines, mais aussi celle du Réparateur et de l'Opérateur. Il intègre également le Taux de Performance qui prend en compte les ralentissements et les écarts de cadence. Les modèles automates d'états du TRS que nous proposons au chapitre 5 permettent de compter le temps que le système passe dans chaque état. Ceci donne une vision synthétique du comportement fonctionnel et dysfonctionnel du système, ce qui favorise une réactivité décisionnelle efficiente, et un suivi performent.

4.3 DÉFINITIONS DU TRS EN FONCTION DES TEMPS D'ETAT (Norme NFE 60-182-mai 2002)

La norme NFE 60–182 s'applique en priorité aux machines ou aux groupes de machines en fonctionnement automatique ou semi-automatique (ligne de conditionnement, centres d'usinage, îlots robotisés). Elle peut facilement être étendue, avec bénéfice, aux

fabrications manuelles ou aux activités de process. Les temps d'état des systèmes de production sont définis en fonction de cette norme comme le présente la figure 4.4.

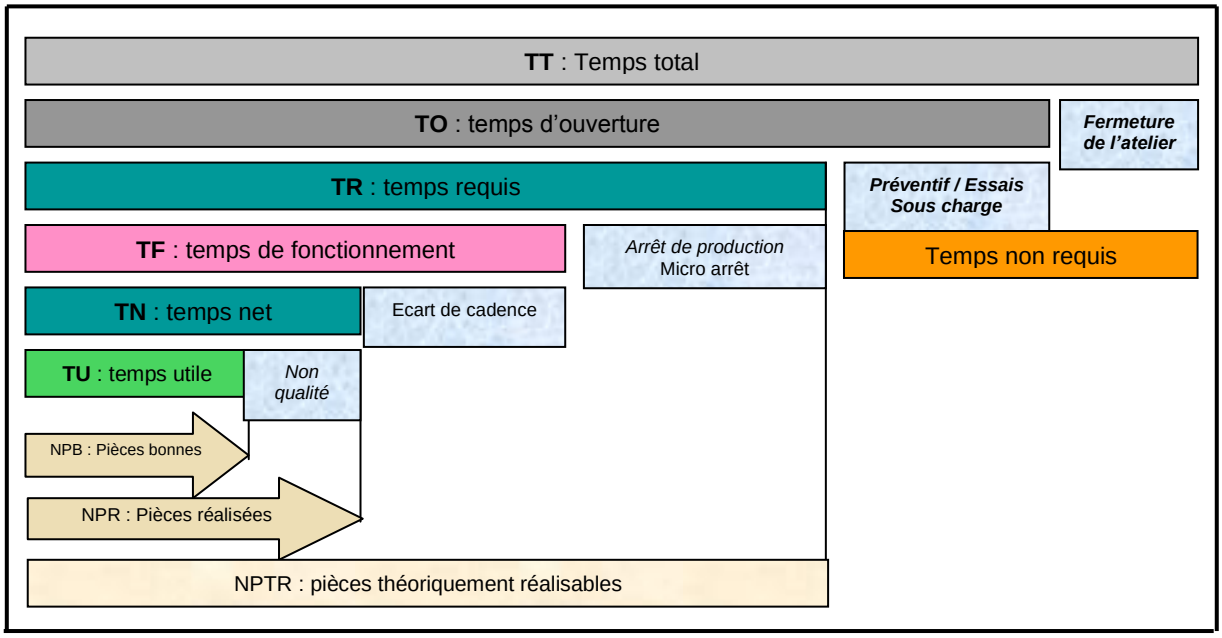


Figure 4.4 : Les temps d'état d'un moyen de production : la norme NFE 60-182 – mai 2002 (source : [AYE 04])

Le temps total (TT): temps de référence intégrant l'ensemble des états possibles du moyen. Pour une journée, le temps total est de 24 h ; pour une semaine, le temps total est de 168 h ; pour un an, le temps total est de 365 jours × 24 h soit 8760 h, etc.

Le temps d'ouverture (TO): partie du temps total correspondant à l'amplitude des horaires de travail du moyen de production et incluant les temps d'arrêt de désengagement du moyen de production par exemple (nettoyage, sous charge, modification, essai, formation, réunion, pause, maintenance préventive,…).

Le temps requis (TR) : partie du temps d'ouverture pendant lequel l'utilisateur engage son moyen de production avec la volonté de produire comprenant les temps d'arrêt subis et programmés (par exemple pannes, changement de série, réglage, absence de personnel).

$$TR = Temps\ d'ouverture\ - temps\ de\ sous\ charge\ (essai + pause + préventif) \qquad (11)$$

Le temps de fonctionnement (TF) : partie du temps requis pendant lequel le moyen de production produit des pièces bonnes et mauvaises dans le respect ou non du temps de cycle de référence (T_{CR}) et avec toutes ou parties des fonctions en service.

$$TF = Temps\ requis - temps\ d'arrêts\ de\ production \qquad (12)$$

Le temps net (TN) : partie du temps de fonctionnement pendant lequel le moyen de production aurait produit des pièces bonnes et mauvaises, dans le respect du temps de cycle de référence.

$$TN = \frac{Nombre\ de\ pièces\ réalisées}{cadence} \quad (13)$$

Le temps utile (TU) : partie du temps net correspondant au temps non mesurable obtenu en multipliant le nombre de pièces bonnes par le temps de cycle de référence ou en divisant le nombre de pièces bonnes par la cadence.

$$TU = \frac{Nombre\ de\ pièces\ bonnes}{cadence} = \frac{Nombre\ de\ pièces\ réalisées - rejets}{cadence}$$

Le temps d'arrêts propres (t_{ap}) : partie du temps requis correspondant au temps d'arrêt imputable au moyen de production.

Le temps de panne (t_p) : partie du temps d'arrêt propre due à un dysfonctionnement.

Le temps d'arrêt d'exploitation (t_{AE}) : partie du temps d'arrêt propre provoquée par l'utilisateur par exemple pour les arrêts de service dus à l'impossibilité du personnel de remplir sa fonction, à des problèmes de qualité,…

Le temps d'arrêts fonctionnels (t_{AF}) : partie programmée du temps d'arrêt propre qui peut se décomposer en :

- t_{COP} : Temps de changement d'outil programmé
- t_{RF} : Temps de réglage fréquentiel
- t_{dC} : Temps de contrôle
- t_{CF} : Temps de changement de fabrication
- t_{EF} : Temps d'entretien fréquentiel

Le temps de micro arrêts (t_{ma}) : partie du temps d'arrêt propre constituée de temps d'arrêt difficilement mesurables dont le seuil est défini par l'entreprise.

Le temps d'arrêts induits (t_{AI}) : partie du temps requis correspondant au temps d'arrêt pendant lequel le moyen de production ne peut accomplir sa fonction pour des causes externes : défaut d'approvisionnement, saturation de pièces, manque de personnel, manque de ressources extérieures, défaut d'énergie.

4.3.1 Autres indicateurs de performance définis en fonction de la norme NFE 60 - 182

Les informations de la figure 4.4 permettent d'obtenir les définitions des indicateurs de performance suivants :

a) Taux de Rendement Economique (TRE) : C'est l'indicateur stratégique d'engagement des moyens de production, il permet aux dirigeants d'affiner la stratégie d'organisation de l'entreprise [AYE03a].

Le taux stratégique d'engagement des moyens se définit par la relation :

$$T_S = \frac{Temps\ d'ouverture}{Temps\ Total} = \frac{TO}{TT} \quad (14)$$

Le Taux de Rendement Economique sera défini par :

$$TRE = \frac{Temps\ Utile}{Temps\ Total} = \frac{TU}{TT} \quad (15)$$

b) Taux de Rendement Global (TRG) : C'est un indicateur de productivité de l'organisation industrielle. C'est un indicateur économique qui intègre la charge effective d'un moyen de production [AYE 03a]. C'est l'outil de mesure du gain de productivité de l'entreprise. Il est employé dans la majeure partie des cas dans des industries manufacturières [PIE 05]. Il permet de répondre à de nombreuses questions stratégiques (actions à engager pour optimiser la production, efficience de l'organisation, besoin d'investissement…). Il exprime la réalité de fonctionnement par rapport à un idéal de fonctionnement et il permet de visualiser les différentes pertes de rendement d'utilisation, de performances et de qualité.

Le Taux de Rendement Global est un ratio entre deux quantités de temps, ou deux quantités de pièces produites. Il y a deux types de temps :

- les temps nécessaires à la fabrication de pièces bonnes (temps utile) ;
- le temps disponible (ou temps d'ouverture).

Mathématiquement, le Taux de Rendement Global se définit par :

$$TRG = \frac{Temps\ Utile}{Temps\ d'Ouverture} = \frac{TU}{TO} \quad (16)$$

Le Taux de charge se définit par :

$$T_C = \frac{Temps\ \mathrm{Re}quis}{Temps\ d'ouverture} = \frac{TR}{TO} \quad (17)$$

4.3.2 Définition du Taux de Rendement Synthétique (TRS)

C'est un indicateur de productivité qui rend compte de l'utilisation effective d'un moyen de production. Du point de vue productivité, Il se présente comme étant le rapport du nombre de pièces bonnes (NPB) sur le nombre de pièces théoriquement réalisables (NPTR).

Le Nombre de Pièces Théoriquement Réalisables (NPTR) se définit de la manière suivante:

$$TRS = \frac{Nombre\ de\ pièces\ bonnes}{Nombre\ de\ pièces\ théoriquement\ réalisables} = \frac{NPB}{NPTR} \quad (18)$$

$$TRS = \frac{NPB}{NTPR} = NPB \times \frac{T_{CR}}{TR}$$

$$NPTR = \frac{TR}{T_{CR}}$$

où

TR – temps requis
T_{CR} – temps de cycle de référence
NPB – Nombre de Pièces Bonnes
NTPR – Nombre de Pièces Théoriquement Réalisables

Le TRS mesure la performance d'un moyen de production. Il permet d'identifier les pertes, il représente un excellent outil d'investigation. Il est la "température" du moyen de production. Mais pour progresser, savoir ne suffit pas, il faut comprendre. C'est pourquoi, on associe toujours à la mesure du TRS, un recueil détaillé et factuel des causes de non rendement synthétique. Les causes serviront à déterminer les temps d'état qui servent de base au calcul des indicateurs.

Du point de vue Sûreté de Fonctionnement, le TRS se définit par le produit du taux de qualité (Tq) par le taux de performance (Tp) et par la disponibilité opérationnelle (Do).

$$\mathrm{TR}S = Tq \times Tp \times Do \tag{19}$$

a) Taux de Qualité (Tq)

C'est le rapport du nombre de bonnes pièces sur le nombre de pièces réalisées.

$$T_q = \frac{Nombre\ de\ pièces\ bonnes}{Nombre\ de\ pièces\ réalisées} = \frac{NPB}{NPR} \tag{20}$$

Le taux de qualité peut encore être exprimé comme le rapport du temps utile sur le temps net.

$$T_q = \frac{TU}{TN} = \frac{temps\ utile}{temps\ net} = \frac{temps\ utile}{temps\ utile + temps\ non\ qualité}$$

$$T_q = \frac{TU}{TU + t_{nq}} \tag{21}$$

b) Taux de performance (Tp)

C'est le rapport du temps net sur le temps de fonctionnement. Il permet de définir l'efficience réelle du moyen de production.

$$T_p = \frac{Temps\ Net}{Temps\ de\ Fonctionnement} = \frac{Temps\ Net}{Temps\ Net + Temps\ écart\ de\ cadence}$$

$$T_p = \frac{TN}{TF} = \frac{TN}{TN + t_{Eca}} \tag{22}$$

c) Disponibilité Opérationnelle (Do)

La Disponibilité Opérationnelle intègre le temps requis (TR) et le temps de fonctionnement (TF). La figure 4.5 montre que le temps requis est composé du temps de fonctionnement (TF) plus les arrêts de production (t_{AP}). Ces arrêts de production sont la composition des arrêts induits (t_{AI}) et des arrêts propres (t_{ap}).

La disponibilité opérationnelle Do se définit par :

$$D_o = \frac{Temps\ de\ \ fonctionnement}{Temps\ \mathrm{Re}quis} = \frac{Temps\ de\ \ fonctionnement}{Temps\ de\ \ fonctionnement + Temps\ Arrêts\ de\ \mathrm{Pr}oduction}$$

$$D_o = \frac{TF}{TR} = \frac{TF}{TF + t_{AP}} \qquad (23)$$

$$Temps\ d'arrêts\ de\ \ production\ \ = \ \ Temps\ d'arrêts\ induits + Temps\ d'arrêts\ \ fonctionnels +$$
$$+ Temps\ d'arrêts\ d'\mathrm{exp}loitation + Temps\ de\ \ panne$$

$$t_{AP} = t_{AI} + t_{ap} \ \ = \ \ t_{AI} + t_{AF} + t_{AE} + t_P$$

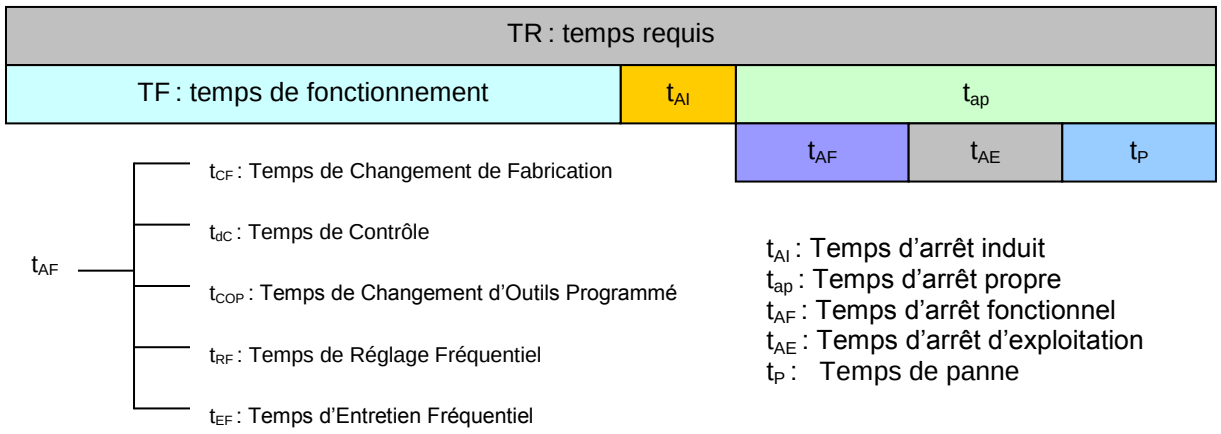


Figure 4.5 : Représentation du temps requis

Les principaux arrêts de production (tableau 4.1) peuvent être imputés aux causes suivantes:

- arrêts induits (causes externes au moyen de production) ;
- arrêts propres (imputable au moyen, à l'outillage, au produit, à l'exploitation du moyen).

Tableau 4.1 : Caractérisation des temps d'arrêts

Type d'arrêt	Causes	
Arrêts Induits	Manque de pièces	
	Saturation de pièces	
	Manque de personnel	
	Défaut d'énergie	
	Manque de ressources extérieures	
Arrêts Propres	Pannes	
	Arrêts d'exploitation	
	Arrêts Fonctionnels	Changement de fabrication
		Contrôle
		Changement d'outils programmé
		Réglage fréquentiel
		Entretien fréquentiel

4.3.3 Formulation du TRS (vue temporelle) à partir de la norme NFE 60-182

D'après la norme AFNOR NFE 60-182 - mai 2002 (figure 4.4), et d'après la formule (19), nous pouvons caractériser le TRS (vue temporelle) de la manière suivante :

$$TRS = \frac{TU}{TU + t_{nq}} \times \frac{TN}{TN + t_{Eca}} \times \frac{TF}{TF + t_{AP}} \tag{24}$$

La formule (24) montre que le TRS d'une entité (composant, sous-système ou système), est une composition de trois indicateurs que sont : la Qualité, la Disponibilité Opérationnelle, la Performance. Chaque indicateur étant fonction des temps d'état de l'entité à l'instant *t* donné.

L'amélioration du TRS ainsi défini passe par l'amélioration de chaque indicateur, et donc des principaux temps d'état :

- amélioration du taux de qualité par diminution du « temps de non qualité » ;
- amélioration du taux de performance par diminution du « temps des écarts de cadence » ;
- amélioration de la Disponibilité Opérationnelle par diminution du « temps d'arrêts de production ».

Il ressort de ces indicateurs que l'amélioration de la capacité productive d'un système passe par la maîtrise des différents temps d'état. Pour atteindre un TRS de l'ordre de 85% (valeur optimale internationale régulièrement citée) [QMA 08], [VOR 08], [vTR 03], [CLE 00] et [AYE 03a], il faut :

- diminuer les arrêts induits (t_{AI}) notamment par une bonne organisation de la gestion des stocks (éviter le manque de pièces ou la saturation de pièces), une bonne organisation du service du personnel par un recrutement suffisant de personnels afin de planifier les tâches. Diminuer les causes extérieures telles que l'absence d'énergie, par l'achat des groupes électrogènes, afin de suppléer l'alimentation secteur en cas de coupure. Il est avantageux d'automatiser le fonctionnement des systèmes afin d'optimiser le temps de réaction.

- diminuer les arrêts propres (t_{AP}), par une bonne organisation du service maintenance afin de minimiser les pannes, le choix d'une bonne politique de maintenance devient alors déterminant pour atteindre ce but. Il serait intéressant d'orienter cette politique vers la maintenance préventive, l'objectif étant d'assurer une bonne disponibilité et une bonne fiabilité des équipements. Les arrêts fonctionnels (t_{AF}) doivent être programmés et rigoureusement respectés. Les arrêts d'exploitation (t_{AE}) doivent également être diminués.
La diminution de tous ces temps d'arrêts augmente le temps de fonctionnement (figure 4.5), améliorant ainsi la Disponibilité Opérationnelle (D_o) qui tend alors vers une valeur maximale de 90% (valeur internationale). La Disponibilité Opérationnelle rend compte de l'utilisation effective des moyens de production.

- augmenter le taux de performance T_p par une diminution des micros arrêts (écart de cadence), en améliorant l'organisation, la fiabilité des équipements et le temps de rémissance (temps de mise en marche de l'équipement) [EFA 03]. Ceci permet d'augmenter le temps net *TN* (figure 4.4). Le taux de performance pourra donc tendre vers 95% (valeur internationale).

- augmenter le taux de qualité T_q par diminution de la non qualité, en améliorant la qualité totale, l'assurance qualité, en agissant sur les paramètres environnementaux et sur la qualité de formation des opérateurs. Ceci augmenterait le temps utile, faisant ainsi tendre le taux de qualité vers une valeur optimale de 99%. Le taux de qualité rend compte de la réelle valeur ajoutée d'un moyen de production.

Toutes les causes responsables de la baisse du TRS peuvent être regroupées dans un diagramme dit « Arbre des causes de faiblesse du TRS » (figure 4.6).

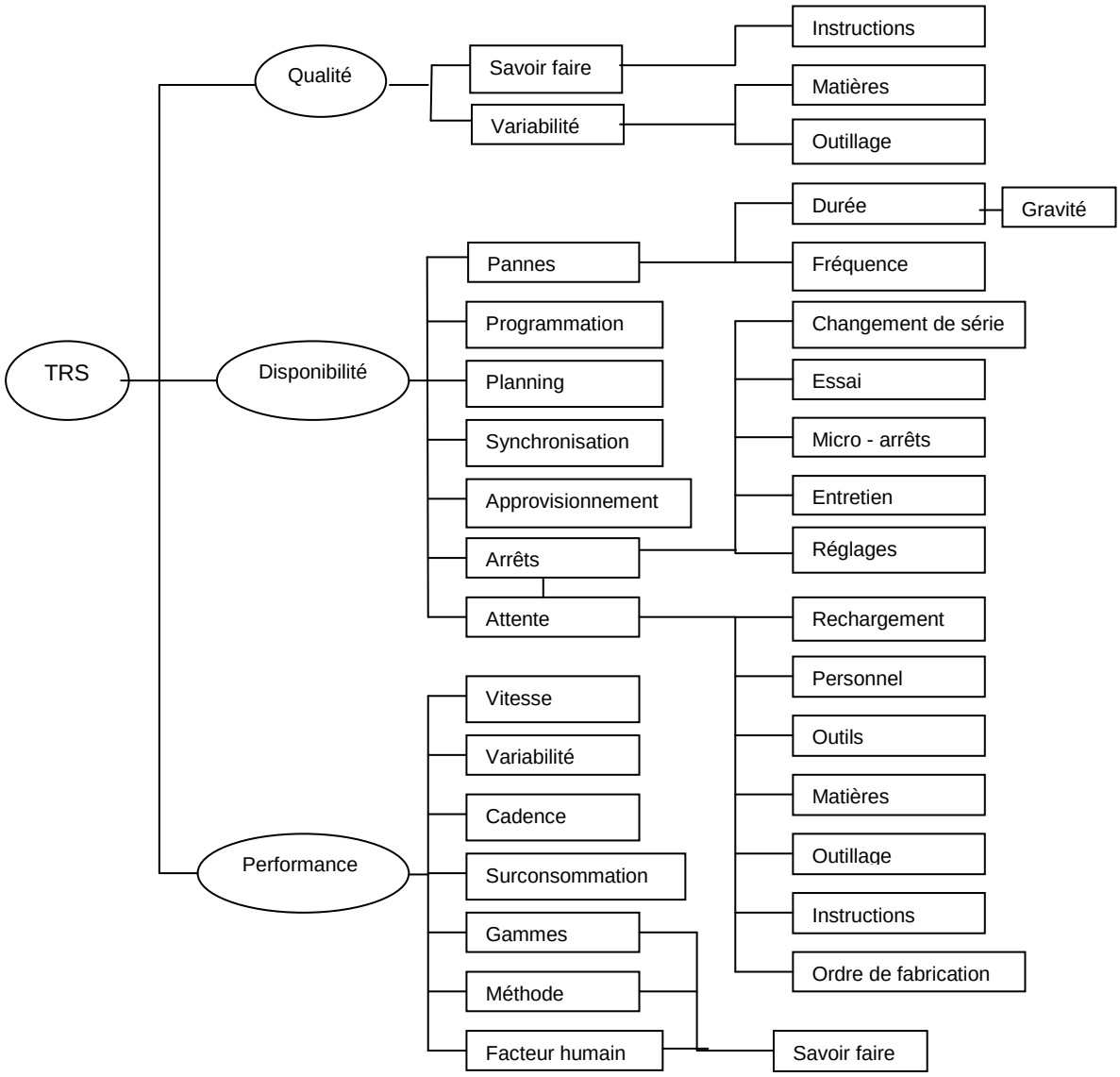


Figure 4.6 : Arbre des causes de faiblesse du TRS

4.3.4 Validation des concepts théoriques dans une PME Camerounaise : cas des Cimenteries du Cameroun (CIMENCAM)

Créée le 16 décembre 1963, la société « Les Cimenteries du Cameroun » (CIMENCAM) est une filiale du groupe LAFARGE. La raison sociale de cette entreprise est de produire et commercialiser le ciment (types CPA 55 et CPJ 35) sur le marché camerounais et dans la sous région Afrique Centrale. En décidant la création de ce complexe industriel, les pouvoirs publics répondaient à une motivation fondamentale : prendre pied dans le secteur de l'industrie en dotant le Cameroun de deux unités, une cimenterie complète dans la localité de Figuil (au Nord Cameroun), et une station de broyage à Bonabéri (Douala).

Nous voulons déterminer le TRS global de la société « Les Cimenteries du Cameroun » (usine de Bonabéri), afin de le comparer à d'autres indicateurs de performance habituellement utilisés à savoir : le Coefficient de Fiabilité (CF) et le Coefficient d'Utilisation

(CU). L'objectif est de ressortir les principaux postes goulets, afin d'améliorer la productivité. Nous nous sommes intéressés au calcul du TRS du broyeur n° 2 d'une société qui en compte trois, pour la période allant du 1er au 23 mai 2005. Notre choix est justifié par la simple raison que ce broyeur est celui qui cause le plus de problèmes de fonctionnement dus aux arrêts. Nous avons travaillé sur un échantillon aussi réduit à cause des difficultés de collecte des données, et surtout à la nécessité d'avoir des informations temps réel.

Les travaux présentés dans cette partie se limitent à la mesure. Les travaux sur la modélisation seront développés au chapitre 5. La méthodologie de calcul utilisée est la suivante : Après la collecte des données de fonctionnement et leur enregistrement automatique en temps réel, nous avons acquis le rapport mensuel de marche de l'atelier. Notre étude s'est ensuite orientée vers un découpage de ce rapport mensuel en périodes hebdomadaires. Une analyse et une étude de chaque sous rapport nous ont permis de déduire les temps d'état de chaque sous période, en mettant un accent sur les temps d'arrêts. Une évaluation des principales composantes du TRS (Taux de Qualité, Taux de Performance, Disponibilité Opérationnelle) a ensuite été effectuée, avant le calcul du TRS hebdomadaire, puis celui du TRS global du système. L'algorithme de cette démarche est présenté à la figure 4.7.

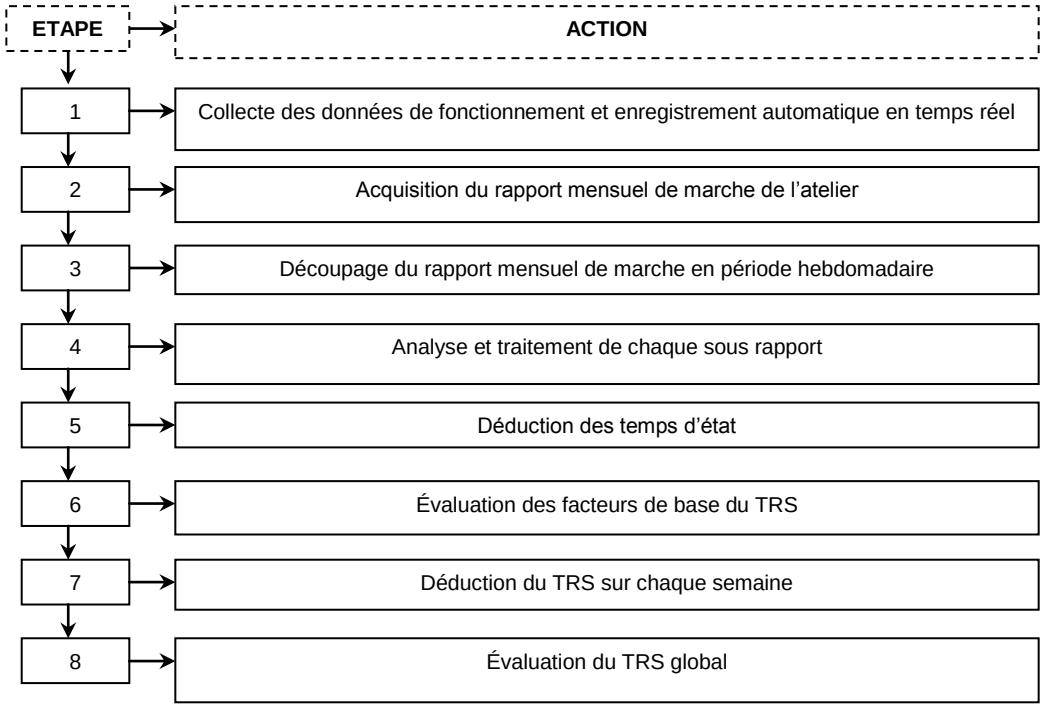


Figure 4.7 : Algorithme de la procédure de calcul du TRS global de la Société « les Cimenteries du Cameroun »

a) Présentation des résultats obtenus

Toutes les mesures faites pendant les trois semaines d'étude (période allant du 1er au 23 mai 2005) ont été regroupées dans les tableaux 4.2 et 4.3. La variation hebdomadaire des différents temps d'état est représentée sur la figure 4.8, tandis que la proportion des temps

d'arrêt durant toute la période d'étude est donnée à la figure 4.9. La figure 4.10 représente l'évolution du TRS et de ses composantes sur les trois semaines.

Tableau 4.2 : Bilan des temps d'état pour la période allant du 1er au 23 mai 2005

Temps d'état	TO	TR	Tnr	Tap	TF	Te	TN	TU	Tre
Bilan du 01-05-2005 au 07-05-2005									
Durée en heure	168.00	168.00	0	7.94	160.0	0	160.06	160.06	0
Bilan du 08-05-2005 au 14 -05-2005									
Durée en heure	72.00	72.00	0	3.74	68.26	0	68.26	0	68.26
Bilan du 15-05-2005 au 23 -05-2005									
Durée en heure	144.00	136.00	8.00	3.43	132.5	9.01	123.56	0	123.56

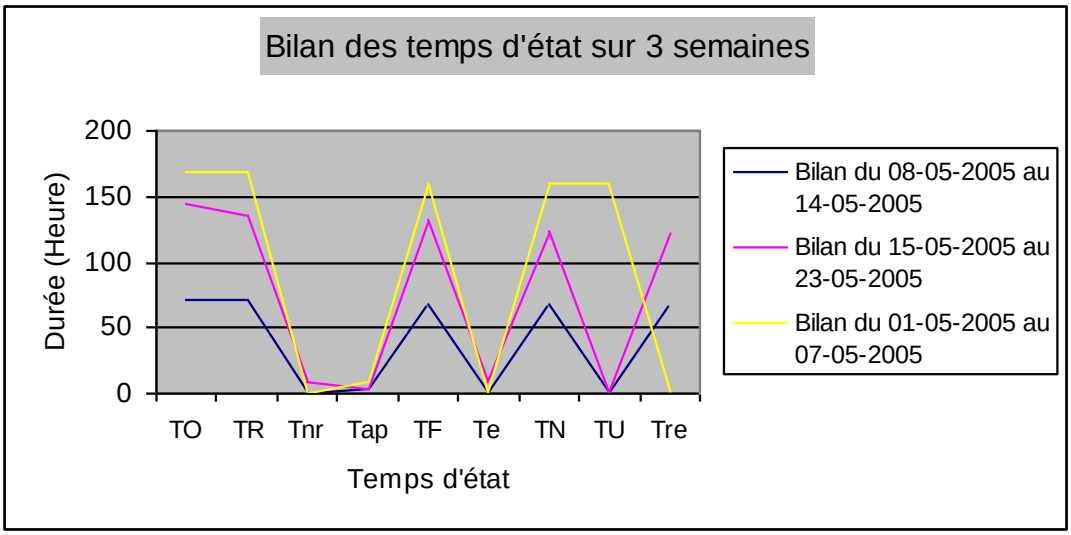


Figure 4.8 : Bilan des temps d'état sur 3 semaines

Tableau 4.3: Bilan des temps d'arrêt durant la période d'étude

N°	Désignation	Durée (Heure)
1	Arrêt sans signalisation	0.48
2	Défaut coffret pompe Fuller	0.97
3	Arrêt NH Silo 5 et passage au Silo 6	0.47
4	Manque KK : travaux sur Redler 1 et 3	0.82
5	Arrêt du BK3 par bourrage du palier G28	0.45
6	Arrêt du BK2 par NH Silo 4	0.21
7	Défaut température G2 côté G28	3.42
8	Défaut alimentation : présence mottes	0.75
9	Défaut asservissement circuit fermé BK2	0.17
10	Arrêt par discordance volet 42292 suite au changement de Silo	0.77
11	Boulon tombé sur BK3 et défaut température sortie BK2	1.16
12	Bourrage élévateur 42239 et élévateur 42237	1.21
13	Bourrage vis 42249	0.51
14	Travaux SAMU sur pompe 42213	0.12
15	Arrêt suite NH Silo5 sans signalisation	0.15

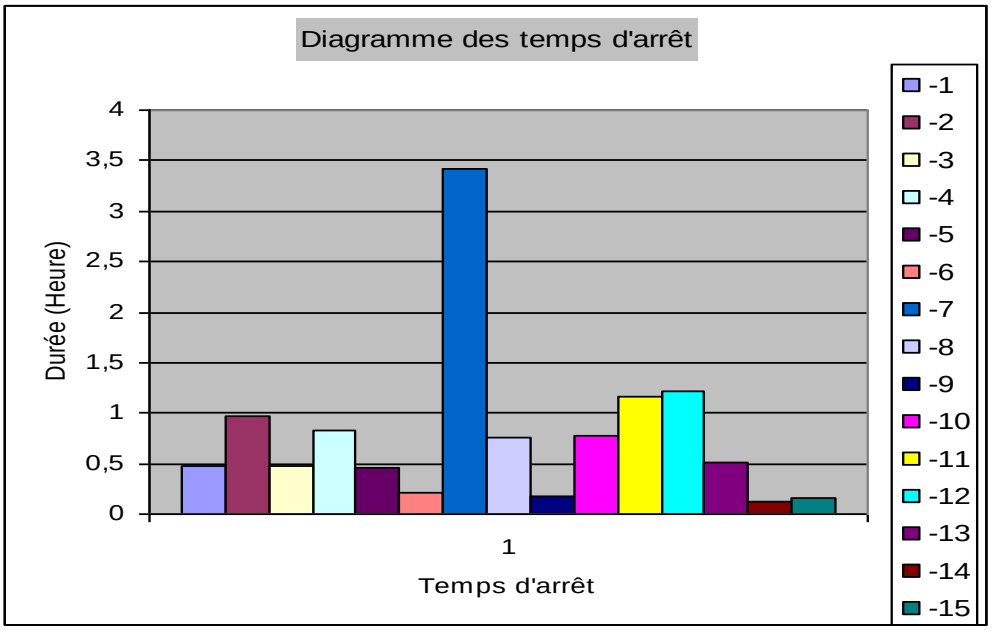


Figure 4.9 : Diagramme des temps d'arrêt

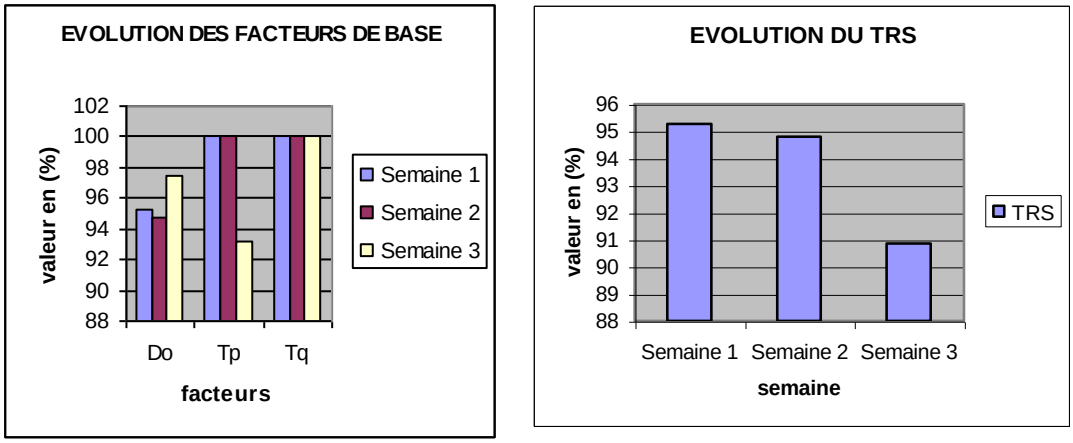


Figure 4.10 : Evolution du TRS et ses composantes par semaine

Tableau 4.4 : Facteurs de base du TRS par période

PÉRIODE	*Do* (%)	Tp (%)	*Tq* (%)	TRS (%)
01-05-2005 au 07-07-2005	95.27	100.00	100.00	95.27
08-05-2005 au 14 -05-2005	94.80	100.00	100.00	94.80
15-05-2005 au 23-05-2005	97.47	93.20	100.00	90.85

TRS du mois = 93,59 %
(Des trois semaines)

b) Analyse des résultats

Notre objectif dans cette partie du travail est de calculer le TRS global du broyeur n° 2 de la société « Les Cimenteries du Cameroun » (CIMENCAM) usine de Bonabéri, afin de déterminer les principaux postes goulets, les principales causes de la baisse de productivité, pour enfin proposer des mesures servant à améliorer les facteurs diminuant graduellement la performance du système, traquer les causes de dysfonctionnement et faire la chasse au gaspillage, afin d'améliorer la productivité.

Les mesures faites sur trois semaines ont montré une variation hebdomadaire des principales composantes du TRS. La figure 4.8 montre une bonne tenue des temps d'état, surtout lors de la première semaine d'étude. On constate une égalité entre le temps d'ouverture et le temps requis, ce qui se justifie par un temps non requis nul. De la même façon, il y a égalité entre le temps utile et le temps net, ce qui se traduit par un temps de non qualité nul et un temps d'arrêts propres nul. La même tendance s'observe à la deuxième semaine bien que les valeurs soient en baisse par rapport à la première semaine. La troisième semaine par contre présente une diminution du temps requis, conséquence directe d'une augmentation du temps non requis, et des micros arrêts.

Le calcul du TRS hebdomadaire pour chaque période d'étude montre un Taux de Qualité constant sur les trois semaines, un Taux de Performance constant les deux premières semaines, mais qui diminue la troisième semaine, et une variation de la Disponibilité Opérationnelle, avec une nette amélioration la troisième semaine (tableau 4.4). Les résultats de la figure 4.10 montrent un TRS diminuant graduellement de la première à la troisième semaine d'étude. On obtient finalement un TRS global de l'ordre de 93,59% sur les trois semaines. A première vue, ce résultat nous semble assez satisfaisant, car largement supérieur à la World Class Performance qui est de 85%. Ce résultat ne garantie tout de même pas la capacité productive de la société qui peut être sujette à des fluctuations. Cette inquiétude est d'ailleurs ressentie avec l'augmentation des temps d'arrêts la troisième semaine et un TRS de l'ordre de 90,85% dans la même période.

Une comparaison faite avec d'autres indicateurs de performance utilisés à CIMENCAM montre la pertinence du TRS qui présente un découpage temporel fin avec une prise en compte de tous les temps d'arrêt. Avec un Coefficient de Fiabilité (CF) supérieur à 97%, le service maintenance semble assez performant, tandis que le Coefficient d'Utilisation (CU) se situe dans l'ordre de 93%. Le Coefficient d'Utilisation est l'objectif du service de fabrication.

Ces deux indicateurs se définissent de la manière suivante :

$$CF = \frac{HM}{HM + DCAI} \qquad \text{et} \qquad CU = \frac{HM}{HO} \tag{25}$$

Avec
HM : Heure de Marche
$DCAI$: Durée Cumulée d'Arrêt sur Incident
HO : Heure d'ouverture

La pertinence du TRS par rapport à ces deux indicateurs est signifiée par le fait que les composantes de ces deux indicateurs semblent globales, et ne prennent pas en compte les détails des différents temps d'état. Par exemple, nous constatons que le CF qui se définit par le rapport des heures de marche sur la somme des heures de marche plus la durée cumulée d'arrêt sur incident, ne présente pas de finesse et de précision sur le découpage des temps de marche et d'arrêt. Certains temps d'arrêts semblent ne pas être pris en compte dans ces calculs, par exemple les arrêts propres, les arrêts fonctionnels, les arrêts induits. Ceci a pour corollaire l'augmentation du temps requis, et la diminution du temps net. C'est ce qui justifie un CF assez élevé. De la même façon, le paramètre « Heure d'Ouverture (HO) » utilisé pour le calcul du Coefficient d'Utilisation (CU), ne précise pas s'il s'agit du temps de fonctionnement (TF), du temps net (TN), ou du temps utile (TU). Ceci diminue automatiquement le dénominateur du quotient de CU, et son augmentation vers une tendance à 100%.

Nous constatons à partir du découpage des temps d'état que pour augmenter le TRS global d'un système (sous système), il faut diminuer les temps d'arrêt. En ce moment, nous pouvons établir une relation étroite entre le service de production influencé par les arrêts induits et le service maintenance gestionnaire des arrêts propres. On peut remarquer qu'une bonne tenue du service de production permet d'annuler les arrêts d'exploitation, et que la bonne tenue du service maintenance diminue les pannes. Les arrêts fonctionnels doivent également être gérés de façon rationnelle et programmée. Il en découle qu'une amélioration

du TRS d'un système de production passe par un rapprochement entre le service de la maintenance et celui de la production.

Nous pouvons, pour conclure cette partie dire que : le TRS est un indicateur intégrateur, qui donne une visibilité directe et optimale sur le comportement des performances des services associés à la production. Mais si son évaluation peut être relativement simple à obtenir pour un élément, son expression pour un système complexe l'est beaucoup moins (prise en compte des redondances, des facteurs d'échelle temporelle, des désynchronisations…). Les modèles automates d'états proposés au chapitre 5, ainsi que leur traduction en AltaRica Data-flow au chapitre 6, permettent, à partir de la simulation stochastique de la dynamique des composantes du TRS, de faire une poursuite du comportement fonctionnel du système.

Afin d'avoir une vision synthétique du comportement individuel de chaque élément constitutif du système, nous présentons dans les lignes qui suivent, les procédures de calcul du TRS pour un système complexe (comportant des éléments montés en série, en parallèle, en expansion et assemblage).

4.4 DÉTERMINATION DU TRS D'UN SYSTÈME REDONDANT

4.4.1 Le TRS global

Le TRS conventionnel tel que défini par Nakajima en 1988 dont l'étude a été faite au § 4.2.2 chapitre 4 permet de calculer la performance individuelle de chaque équipement d'un système. L'extension de cette étude dans le cadre global d'un système de production comportant plusieurs sous systèmes nécessite la prise en compte de certains paramètres tels que l'efficacité de chaque composant de la chaîne. La notion de TRS global (TRS_G) a donc été introduite pour permettre l'évaluation des contributions individuelles des différents éléments constitutifs des systèmes de base (série, parallèle, expansion et assemblage). Le TRS_G est une fonction du TRS conventionnel qui peut être évalué sous deux aspects : l'aspect Productivité et l'aspect Sûreté de Fonctionnement. L'un met l'accent sur la quantité de produit réalisé, l'autre sur les comportements fonctionnel et dysfonctionnel des équipements mis en œuvre. Le calcul du premier est basé sur l'efficience des différentes composantes que sont : la Qualité, la Disponibilité Opérationnelle et la Performance, celui du second est basé sur les taux de ces mêmes composantes (Taux de Qualité, Taux de Performance et Disponibilité Opérationnelle). La différence entre le TRS conventionnel et le TRS_G point de vue productivité réside dans l'introduction de la composante « Performance efficiente » (P_{eff}). Il permet de capturer les pertes de la productivité causées uniquement par le matériel seul [MUT 07] et [MAH 04].

Du point de vue productivité, le TRS_G est développé pour permettre une comparaison entre la production à un instant donné à la production théoriquement réalisable de tout le système. Nous avons vu précédemment pour le cas d'un composant que le TRS conventionnel se définit comme le rapport du nombre de pièces bonnes sur le nombre de pièces théoriquement réalisable.

$$TRS_i = \frac{Nombre\ de\ pièces\ bonnes(composant)}{Nombre\ de\ pièces\ théoriquement\ réalisables(composant)} = \frac{NPB_{(C)}}{NPTR_{(C)}} \qquad (26)$$

De la même façon, on peut définir le TRS_G d'un système complexe comme étant le rapport du nombre de pièces bonnes à la sortie du système par le nombre de pièces théoriquement réalisables par le système.

$$TRS_G = \frac{Nombre\ de\ pièces\ bonnes\ (système)}{Nombre\ de\ pièces\ théoriquement\ réalisables\ (système)} = \frac{NPB_{(S)}}{NPTR_{(S)}} \tag{27}$$

Le TRS_G pour les sous systèmes (série, parallèle, expansion et assemblage) est basé sur une approche prenant en compte le temps que passe un équipement dans un état inactif (état d'attente). L'idée est la suivante : si un processus a besoin d'être traité par l'équipement A avant d'être traité par l'équipement B (par exemple A et B sont montés en série), alors la qualité et la quantité du produit à la sortie de B sont contraintes par le comportement de l'équipement A. Si l'équipement B ne connaît pas de temps d'inactivité (attente) causé par le manque de ressources à la sortie de l'équipement A, alors, le nombre de produit bon à la sortie de B dépend de son TRS individuel, de sa cadence théorique, ainsi que du temps requis. Autrement, le nombre de bonnes pièces est contraint par ce que A peut produire, et par l'efficience de l'équipement B.

La formule permettant de déterminer le TRS de chaque composant d'une chaîne point de vue productivité est la suivante :

$$TRS = D_{eff} \times P_{eff} \times Q_{eff} \tag{28}$$

où

D_{eff} – Disponibilité efficiente (pertes associées incluant les arrêts induits, les arrêts propres, les pannes etc.) ;

P_{eff} – Performance efficiente (pertes associées incluant les ralentissements, les écarts de cadence, les blocages etc.) ;

Q_{eff} – Qualité efficiente (pertes associées incluant la non qualité, les reprises, les défauts de fabrication etc.).

$$D_{eff} = \frac{TF}{TR} \tag{29}$$

$$P_{eff} = \frac{TN}{TF} \times \frac{t_p^{(r)}}{t_p^{(th)}} \tag{30}$$

$$Q_{eff} = \frac{NPB}{NPR} = \frac{nombre\ de\ pièces\ bonnes}{nombre\ de\ pièces\ réalisées} \tag{31}$$

Où

TF – Temps de fonctionnement

TR – Temps requis

TN – Temps net

$t_P^{(th)}$ - Taux de productivité théorique

$t_P^{(r)}$ - Taux de productivité réel

NPB – Nombre de pièces bonnes

NPR – Nombre de pièces réalisées

Puisqu'un équipement ne peut pas produire à la cadence théorique durant le temps net (TN), le taux de productivité réel ($t_P^{(r)}$) se formule de la façon suivante :

$$t_P^{(r)} = \frac{NPR}{TN} \tag{32}$$

En considérant les formules (28), (29), (30), (31) et (32) on obtient une formule importante pour la définition du TRS point de vue productivité:

$$TRS = \frac{NPB}{t_P^{(th)} \cdot TR} = \frac{NPB}{NPTR} = NPB \cdot \frac{T_{CR}}{TR} \tag{33}$$

L'égalité (33) montre que le nombre de produit théoriquement réalisable peut encore se définir :

$$NPTR = \frac{TR}{T_{CR}} = t_P^{(th)} \cdot TR \tag{34}$$

De la même façon, on peut constater que le taux de productivité théorique est égal à l'inverse du temps du cycle de référence :

$$t_P^{(th)} = \frac{NPTR}{TR} = \frac{1}{T_{CR}} \tag{35}$$

Par analogie, la formule (27) peut être mise sous forme de l'équation (33), ce qui nous permet de définir le TRS_G :

$$TRS_G = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{NPB_{(S)}}{t_{P(S)}^{(th)} \cdot TR} \tag{36}$$

On peut également tirer le nombre de produit théoriquement réalisable par le système pendant le temps requis et le définir sous forme de l'équation (34).

$$NPTR_{(S)} = t_{P(S)}^{(th)} \cdot TR \tag{37}$$

Où $t_{P(S)}^{(th)}$ est le taux de productivité théorique de tout le système. Il définit la fonction d'inter connectivité du système.

La performance d'un système de production dépend de plusieurs facteurs. Elle dépend du mode d'exploitation, qui lui même dépend de la politique de production mise sur pied, du degré de qualification du personnel exploitant… Elle dépend aussi de la complexité du système, dont la structure peut prendre des configurations allant du simple (série, parallèle, expansion ou assemblage) au complexe (combinaison de certaines des configurations de base). Si un système de production comporte *n* composants, le TRS individuel et le nombre de produit bon de chaque composant durant une période d'observation correspondant au temps requis se définissent par :

$$TRS_i = D_{eff_i} \times P_{eff_i} \times Q_{eff_i} \qquad i = 1, ..., n \tag{38}$$

L'équation (33) nous permet encore d'écrire :

$$NPB_i = TRS_i \cdot t_{P_i}^{(th)} \cdot TR \qquad i = 1,...,n \tag{39}$$

L'égalité (39) définit le nombre de produit bon pour chaque composant de la chaîne. De la même façon, l'équation (36) nous permet de définir le nombre de produit bon de tout le système par la relation :

$$NPB_S = TRS_G \cdot t_{P_S}^{(th)} \cdot TR \tag{40}$$

Il faut mentionner que TRS_i et NPB_i pourraient être des variables aléatoires dans une longue période d'observation. La raison est que pour une période d'observation différente du temps requis ou une période d'observation de même longueur mais qui commence à des temps différents, dans la plupart des cas, les valeurs mesurées du TRS_i et du NPB_i seront différentes à cause du caractère aléatoire de la disponibilité, de la performance et du rendement du composant. Par conséquent, les valeurs TRS_i et NPB_i ne sont pas connues avec certitude avant qu'elles ne soient mesurées pendant la période d'observation. Pour être significatives et utilisables, les valeurs mesurées du TRS_i et du NPB_i doivent être associées à un temps d'observation correspondant au temps requis TR. Cependant, si pendant la période d'observation, le composant *i* peut atteindre un état stable de fonctionnement, alors en utilisant des approches statistiques, les valeurs attendues du TRS_i et du NPB_i peuvent être déterminées.

4.4.2 Détermination du TRS global (TRS_G) d'un système du point de vue productivité

La structure de tout système de production épouse l'une des configurations de base à savoir : Série, Parallèle, Expansion, Assemblage ou complexe (combinaison de certaines des configurations de base).

Basé sur le TRS individuel (TRS_i) de chaque composant du système, le TRS global (TRS_G) peut être déduit. Les formules permettant de calculer le TRS_G pour les systèmes de base Série, Parallèle, Expansion et Assemblage présentent tout de même des limites d'information concernant les „reprises"" dans la perte de la qualité. Ceci rend très complexe l'étude des systèmes combinés.

a) Détermination du TRS_G d'un système comportant *n* composants monté en Série

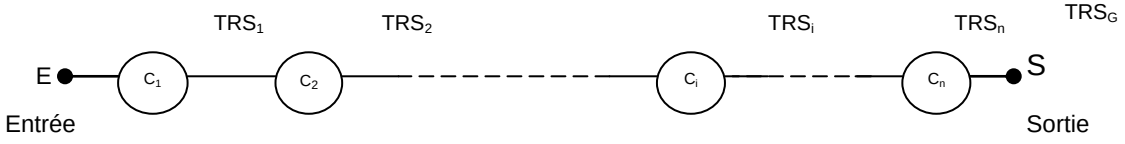


Figure 4.11 : Système de production comportant *n* éléments en série

Le schéma de représentation d'un système comportant *n* composants montés en Série est donné à la figure 4.11. Pour tout système série basé sur la théorie de la conservation du flux de matière, pendant un temps d'observation correspondant au temps requis TR, le nombre de produit bon (NPB) à la sortie du n^{ième} composant doit correspondre à celui de tout le système. Autrement dit :

$$NPB_S = NPB_n \tag{41}$$

Où

NPB_n - nombre de produit réalisé à la sortie du n^{ième} composant

NPB_S - nombre de produit réalisé à la sortie du système

Donc

$$NPB_S = TRS_n \cdot t_{P_n}^{(th)} \cdot TR \tag{42}$$

Dans un système série, la production est dominée par la cadence de la machine la plus lente de la chaîne. Ainsi, le taux de productivité global d'un système comportant *n* composants montés en série pendant le temps requis TR se définit par :

$$t_{P_S}^{(th)} = \min\left\{t_{P_i}^{(th)}\right\} \qquad i = 1,...,n \tag{43}$$

En utilisant les équations (27), (37), (40), (42) et (43), le TRS_G pour un système Série se définit par :

$$TRS_{G(série)} = \frac{NPB_S}{NPTR_S} = \frac{NPB_S}{t_{P_S}^{(th)} \cdot TR} = \frac{TRS_n \cdot t_{P_n}^{(th)}}{t_{P_S}^{(th)}} = \frac{D_{eff_n} \cdot P_{eff_n} \cdot Q_{eff_n} \cdot t_{P_n}^{(th)}}{\min\left\{t_{P_i}^{(th)}\right\}} \tag{44}$$

Il faut mentionner que le taux de productivité théorique d'un système (sous système) série pour le nombre de produit réalisé $t_{P_S}^{(th)}$ dépend du nombre de types de produits, du taux de productivité de chaque composant de la chaîne pour chaque type de produit et du temps d'observation correspondant au temps requis TR.

La formule (44) peut encore se mettre sous la forme :

$$TRS_{G(série)} = \frac{\min\left\{\min_{i=1,2,...,n-1}\left\{TRS_i \cdot t_{P_i}^{(th)} \cdot \prod_{j=i+1}^{n} Q_{eff_j}\right\}, TRS_n \cdot t_{P_n}^{(th)}\right\}}{\min_{i=1,2,...,n}\left\{t_{P_i}^{(th)}\right\}} \tag{45}$$

Où

TRS_i - Taux de Rendement Synthétique du $i^{ème}$ composant

TRS_n - Taux de Rendement Synthétique du $n^{ième}$ composant

$t_{P_i}^{(th)}$ - Taux de productivité théorique du $i^{ème}$ composant, généralement défini par l'exploitant

Q_{eff_j} - Qualité efficiente du $j^{ème}$ composant

b) Détermination du TRS_G d'un système comportant *n* composants monté en Parallèle

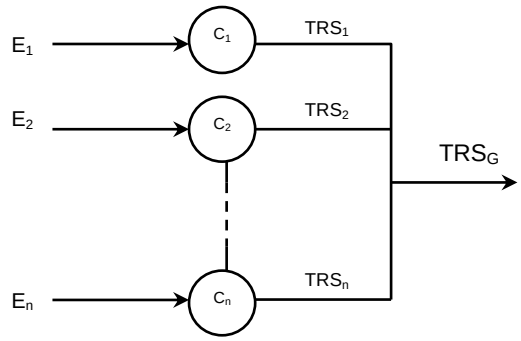


Figure 4.12 : Système de production comportant n éléments en parallèle

Le schéma d'un système (sous système) comportant *n* composants montés en parallèle est donné à la figure 4.12. Pour un système Parallèle basé sur la théorie de la conservation du flux de matière, pendant la période d'observation correspondant au temps requis TR, le nombre de produit bon à la sortie du système est égal à la somme des bons produits de chaque machine montée en parallèle, et le nombre de produit réalisé doit également correspondre à la somme des nombres de produits réalisés par chaque machine montée en parallèle. Deux cas peuvent alors se présenter :

- *Redondance active (ou encore redondance chaude)*: tous les composants sont normalement en fonctionnement permanent ;
- *Redondance passive (ou encore redondance froide)* : il est nécessaire de mettre en état d'éveil au moins un élément de la chaîne afin de prendre le relais lors de la défaillance du composant en fonctionnement.

- **Redondance chaude**

Dans le cas d'une redondance chaude, deux situations peuvent se présenter : Toutes les machines sont identiques, c'est-à-dire fonctionnent avec la même cadence et fabriquent le même type de produit, ou alors au moins une est différente, c'est-à-dire qu'elle fonctionne avec une cadence différente de celle des autres. Elle peut alors, soit fabriquer le même type de produit mais avec une cadence différente, soit fabriquer un produit d'une autre nature. Dans les deux cas, le nombre de produit à la sortie du système est égal à la somme des productions de chaque branche. L'efficience du système est alors égale à la somme des efficiences de chacune des branches montées en parallèle.

Le nombre de produits bons à la sortie du système est égal à la somme des nombres de produits bons de chaque machine, et le nombre de produits réalisés du système est également égal à la somme du nombre de produits réalisés par chaque machine.

$$NPB_S = \sum_{i=1}^{n} NPB_i \quad et \quad NPR_S = \sum_{i=1}^{n} NPR_i \tag{46}$$

Ainsi, on a

$$NPR_S = \sum_{i=1}^{n} \left(t_{P_i}^{(r)} \cdot TN_i \right) \tag{47}$$

En remplaçant le premier membre de l'équation (46) par l'expression de l'équation (39), on obtient :

$$NPB_S = \sum_{i=1}^{n} \left(TRS_i \cdot t_{p_i}^{(th)} \cdot TR \right) \tag{48}$$

Dans un système parallèle, le taux de productivité théorique du système est égal à la somme des taux de productivités théorique de chaque composant. On a :

$$t_{P_S}^{(th)} = \sum_{i=1}^{n} t_{P_i}^{(th)} \tag{49}$$

En utilisant les équations (27), (40), (47), (48) et (49), le taux de rendement synthétique global d'un système parallèle ayant des composants identiques se définit par :

$$TRS_{\substack{G(\text{parallèle actif}) \\ C_i = C_j}} = TRS_{\substack{G(\text{parallèleactif}) \\ C_i \neq C_j}} = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n} \left(TRS_i \cdot t_{P_i}^{(th)} \right)}{t_{P_S}^{(th)}}$$

$$TRS_{\substack{G(\text{parallèle actif}) \\ C_i = C_j}} = TRS_{\substack{G(\text{parallèleactif}) \\ C_i \neq C_j}} = \frac{TRS_1 \times t_{P1}^{(th)} + TRS_2 \times t_{P2}^{(th)} + \cdots + TRS_n \times t_{Pn}^{(th)}}{t_{P1}^{(th)} + t_{P2}^{(th)} + \cdots + t_{Pn}^{(th)}}$$

$$TRS_{\substack{G(\text{parallèle actif}) \\ C_i = C_j}} = TRS_{\substack{G(\text{parallèleactif}) \\ C_i \neq C_j}} = \frac{\sum_{i=1}^{n} \left(D_{eff_i} \cdot P_{eff_i} \cdot Q_{eff_i} \cdot t_{P_i}^{(th)} \right)}{\sum_{i=1}^{n} t_{P_i}^{(th)}} \tag{50}$$

Où C_i et C_j sont le $i^{ème}$ et le $j^{ème}$ composants du système

- **Redondance froide**

Dans le cas d'une redondance froide, au moins une machine du système est mise en attente pour prendre le relais en cas de défaillance de la machine en fonctionnement. Deux cas de figure peuvent alors se présenter: d'abord, le système fonctionne sans redémarrage des composants (cas d'un système non réparable). Dans ce cas, tout composant qui tombe en panne est définitivement exclu du système. Puis, on considère que le système fonctionne avec redémarrage (cas d'un système réparable) c'est-à-dire que lorsqu'un composant tombe en panne, il est réparé et remis en attente. Il peut dans ce cas être à nouveau sollicité si le dernier composant en attente tombe en panne à son tour. Le système fonctionnera ainsi tant que la période d'observation (temps requis) n'est pas encore atteinte.

- **Redondance froide sans redémarrage**

Dans le cas d'un système comportant plusieurs composants montés en redondance froide sans redémarrage, le nombre de produits théoriquement réalisables par chaque composant est exactement le même et égal à celui du système tout entier. Tout composant du système est supposé fonctionner pendant la durée d'observation s'il n'est pas défaillant.

$$t_{P_1}^{(th)} = t_{P_2}^{(th)} = \cdots = t_{P_i}^{(th)} = t_{P_S}^{(th)} = \frac{NPTR_i}{TR} = \frac{NPTR_{(S)}}{TR} \tag{51}$$

Où

$t_{P_i}^{(th)}$ - Taux de productivité théorique du $i^{ème}$ composant

$t_{P_S}^{(th)}$ - Taux de productivité théorique du système

TR – Temps requis

$NPTR_i$ – Nombre de produits théoriquement réalisable du $i^{ème}$ composant

$NPTR_{(S)}$ – Nombre de produits théoriquement réalisable du système

Le TRS global du système est égal au rapport du nombre de produits bons du système divisé par le nombre de produits théoriquement réalisables du système. Or le nombre de produits bons du système est égal à la somme des nombres de produits bons de chaque composant. Le nombre de produits bons de chaque composant lui-même étant égal au nombre de produit réalisé par le composant (NPR_i) moins les rejets enregistrés pour le même composant.

$$TRS_{\substack{G(\text{parallèle passif})\\ \text{sans redémarrage}}} = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n} NBP_i}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n} \left(NPR_i - rejets_i\right)}{NPTR_{(S)}} \tag{52}$$

En utilisant les équations (27), (37) et (48) on obtient :

$$TRS_{\substack{G(\text{parallèle passif})\\ \text{sans redémarrage}}} = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n} \left(TRS_i \cdot t_{P_i}^{(th)}\right)}{t_{P_S}^{(th)}} \tag{53}$$

En considérant les conditions définies dans la relation (51), l'équation (53) devient :

$$TRS_{\substack{G(\text{parallèle passif})\\ \text{sans redémarrage}}} = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n} \left(TRS_i \cdot t_{P_i}^{(th)}\right)}{t_{P_S}^{(th)}} = \frac{\sum_{i=1}^{n} \left(D_{eff_i} \cdot P_{eff_i} \cdot Q_{eff_i} \cdot t_{P_S}^{(th)}\right)}{t_{P_S}^{(th)}} = \sum_{i=1}^{n} TRS_i \tag{54}$$

L'équation (54) montre que le TRS global d'un système parallèle passif sans redémarrage des composants est égal à la sommes des TRS locaux de chaque composant ayant fonctionné dans la limite du temps d'observation (temps requis).

- **Redondance froide avec redémarrage**

Dans un système comportant plusieurs composants montés en redondance froide avec redémarrage (cas d'un système réparable), le TRS local du $i^{ème}$ composant au $j^{ème}$ démarrage se définit par :

$$\underset{\substack{i=1,2,\ldots,n\\ j=1,2,\ldots,k}}{TRS_{ij}} = D_{eff_{ij}} \times P_{eff_{ij}} \times Q_{eff_{ij}} \tag{55}$$

Où

TRS_{ij} - Taux de rendement synthétique du $i^{ème}$ composant au $j^{ème}$ démarrage

D_{effij} - Disponibilité efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage

P_{effij} - Performance efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage

Q_{effij} - Qualité efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage

- *Disponibilité efficiente* : La Disponibilité efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage est égale au rapport du temps de fonctionnement du même composant au démarrage considéré sur le temps requis.

$$D_{eff_{ij}} = \frac{TF_{ij}}{TR} \tag{56}$$

- *Performance efficiente* : Tout comme dans l'équation (30), la Performance efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage se définit de la façon suivante :

$$P_{eff_{ij}} = \frac{TN_{ij}}{TF_{ij}} \times \frac{t_{P_{ij}}^{(r)}}{t_{P_{ij}}^{(th)}} \tag{57}$$

Avec

$$t_{P_{ij}}^{(r)} = \frac{NPR_{ij}}{TN_{ij}}$$

Et

$$t_{P_{ij}}^{(th)} = \frac{NPTR_{ij}}{TR}$$

- *Qualité efficiente* : La qualité efficiente du $i^{ème}$ composant au $j^{ème}$ démarrage est égale au rapport du nombre de produits bons sur le nombre de produits réalisés des mêmes composants dans lesdits états.

$$Q_{eff_{ij}} = \frac{NPB_{ij}}{NPR_{ij}} = \frac{NPR_{ij} - rejets_{ij}}{NPR_{ij}} \tag{58}$$

Le TRS global d'un système comportant plusieurs composants montés en redondance froide avec redémarrage est égal à la somme des TRS locaux de chaque composant à chaque démarrage.

$$TRS_{\substack{G(parallèle\ passif)\\ avec\ redémarrage}} = \frac{NPB_{(S)}}{NPTR_{(S)}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k}\left(TRS_{ij}\cdot t_{P_{ij}}^{(th)}\right)}{t_{PS}^{(th)}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k}\left(D_{eff_{ij}}\cdot P_{eff_{ij}}\cdot Q_{eff_{ij}}\cdot t_{P_S}^{(th)}\right)}{t_{P_S}^{(th)}} = \sum_{i=1}^{n}\sum_{j=1}^{k} TRS_{ij} \tag{59}$$

c) Détermination du TRS_G d'un système comportant *n* composants monté en Assemblage

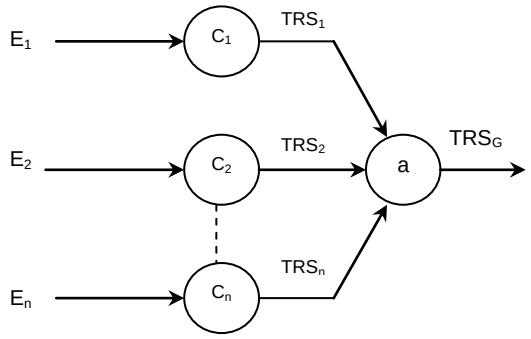


Figure 4.13 : Diagramme Assemblage

Le schéma qui permet de représenter un système comportant des composants montés en Assemblage est donné à la figure 4.13. Dans ce schéma, C_1, C_2, …, C_n sont les composants, *a* est le composant assembleur. Pour un système Assemblage, le taux de productivité théorique du système se définit par:

$$t_{P_S}^{(th)} = \min\left\{\min_{i=1,2,\ldots,n}\left\{\frac{t_{P_i}^{(th)}}{k_i}\right\}, t_{P_{(n+1)}}^{(th)}\right\} \tag{60}$$

Ainsi le Taux de Rendement Synthétique global pour un système comportant *n* composants montés en Assemblage se définit par :

$$TRS_{G(assemblage)} = \frac{\min\left\{\min_{i=1,2,\ldots,n}\left\{TRS_i \cdot \frac{t_{P_i}^{(th)}}{k_{A_i}} \cdot Q_{eff_a}\right\}, t_{P_a}^{(th)} \cdot TRS_a\right\}}{\min\left\{\min_{i=1,2,\ldots,n}\left\{\frac{t_{P_i}^{(th)}}{k_{A_i}}\right\}, t_{P_a}^{(th)}\right\}} \tag{61}$$

Où

n – nombre de composants dans le système

$t_{P_i}^{(th)}$ - taux de productivité théorique du $i^{ème}$ composant

TRS_i - Taux de Rendement Synthétique du $i^{ème}$ composant

Q_{eff_a} - Qualité efficiente du composant assembleur

k_{A_i} - nombre de parts requises du $i^{ème}$ composant pour la réalisation du produit final dans le système assemblage

d) Détermination du TRS_G d'un système comportant *n* composants monté en Expansion

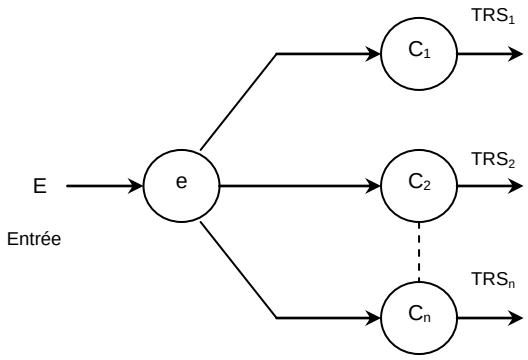


Figure 4.14 : Diagramme Expansion

Le schéma de représentation d'un système comportant *n* composants montés en expansion est donné à la figure 4.14. Dans ce schéma, C_1, C_2, …, C_n sont les composants, *e* est le composant assurant la fonction expansion. Dans ce système, le taux de productivité théorique est défini par la relation :

$$t_{P_S}^{(th)} = \sum_{i=1}^{n} \min\left\{t_{P_e}^{(th)} \cdot k_{E_i} \cdot t_{P_i}^{(th)}\right\} \tag{62}$$

Ainsi, le taux de rendement synthétique global pour un système comportant *n* composants montés en Expansion sera défini par :

$$TRS_{G(\exp ansion)} = \frac{\sum_{i=1}^{n} \min\left\{t_{P_e}^{(th)} \cdot TRS_e \cdot k_{E_i} \cdot Q_{eff_i}, t_{P_i}^{(th)} \cdot TRS_i\right\}}{\sum_{i=1}^{n} \min\left\{t_{P_e}^{(th)} \cdot k_{E_i} \cdot t_{P_i}^{(th)}\right\}} \tag{63}$$

Où

n – nombre de composants dans le système

$t_{P_i}^{(th)}$ - taux de productivité théorique du $i^{ème}$ composant

TRS_i - Taux de Rendement Synthétique du $i^{ème}$ composant

k_{E_i} - le nombre de parts de produit réalisés par le composant en expansion qui doit être envoyée au $i^{ème}$ composant

Q_{eff_i} - qualité efficiente du $i^{ème}$ composant

D'une façon générale, le taux de productivité théorique $t_P^{(th)}$ et la Qualité efficiente théorique Q_{eff} de chaque système dans chacune des configurations série, parallèle, assemblage et expansion peuvent être récapitulés dans le tableau 4.5 ci-dessous.

Tableau 4.5 : Taux de productivité et qualité efficiente des différentes configurations

Système	**Taux de productivité théorique du système** $t_{P_S}^{(th)}$	**Qualité efficiente théorique du système** Q_{eff_S}
Série	$\min_{i=1,2,\ldots,n}\left\{t_{P_i}^{(th)}\right\}$	$\prod_{i=1}^{n} Q_{eff_i}$
Parallèle	$\sum_{i=1}^{n} t_{P_i}^{(th)}$	$\frac{\sum_{i=1}^{n} Q_{eff_i}}{n}$
Assemblage	$\min\left\{\min_{i=1,2,\ldots,n}\left\{\frac{t_{P_i}^{(th)}}{k_{A_i}}\right\}, t_{P_a}^{(th)}\right\}$	$\frac{\sum_{i=1}^{n} k_{A_i} \cdot Q_{eff_i}}{\sum_{i=1}^{n} k_{A_i}} \cdot Q_{eff_A}$
Expansion	$\sum_{i=1}^{n} \min\left\{t_{P_e}^{(th)} \cdot k_{E_i}, t_{P_i}^{(th)}\right\}$	$Q_{eff(E)} \cdot \frac{\sum_{i=1}^{n} k_{E_i} \cdot Q_{eff_i}}{\sum_{i=1}^{n} k_{E_i}}$

4.4.3 Détermination du TRS global d'un système du point de vue Sûreté de Fonctionnement (SdF)

Du point de vue Sûreté de Fonctionnement, les procédures de calcul permettant de déterminer le TRS global (TRS_G) d'un système redondant dépendent de la réparabilité du système. L'évaluation du TRS_G d'un système sera différente selon que les composants du système sont réparables ou non.

a) TRS d'un système nominal comportant *n* composants montés en série

Lorsqu'un système de production comporte *n* éléments montés en série, la performance du système dépend en entier du comportement individuel de chaque composant. Il va de soit que la mise hors d'usage ou le dysfonctionnement d'un seul composant de la chaîne de production compromet le fonctionnement de tout le système. Ainsi, l'état de la sortie *S* dépend du comportement de chacun des composants de la chaîne à l'instant *t* donné.

Chaque Cellule élémentaire de la chaîne étant déclenchée par un événement Ev_i (par exemple la présence d'une ressource), le TRS_i élémentaire à la sortie de la cellule est le résultat de l'action de l'événement sur le processus local. La figure 4.11 peut alors être modifiée, et on obtient la figure 4.15 suivante :

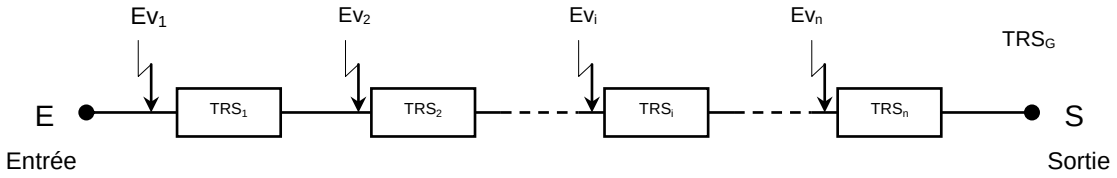


Figure 4.15 : Caractérisation de l'évaluation du TRS dans chaque cellule élémentaire d'un système

Le schéma de la figure 4.15 montre que le TRS global (TRS_G) à la sortie du système n'est obtenu que si tous les événements Ev_i se réalisent. Nous considérons qu'aucune cellule n'est ni défaillante, ni en attente, et par conséquent le système ne connaît aucun temps d'arrêts de production ($t_{AP} = 0$). Nous considérons également qu'il existe une ressource de quantité infinie à l'entrée *E* du système. Ces conditions optimisent le calcul du TRS_i pour chaque cellule. La probabilité d'obtenir un TRS_G à la sortie du système dépend de la probabilité que tous les évènements se réalisent. On peut alors l'exprimer par la relation :

$$TRS_G = P\left[Ev_1 \cdot Ev_2 \cdot \ldots \cdot Ev_i \cdot \ldots \cdot Ev_n\right]$$

Lorsque les événements sont indépendants :

$$TRS_G = \prod_{i=1}^{n} P\left[Ev_i\right] \qquad (64)$$

Or nous savons que chaque action de l'événement Ev_i sur le processus génère un TRS_i, donc :

$$TRS_i = P\left[Ev_i\right]$$

D'où :

$$TRS_G = \prod_{i=1}^{n} TRS_i \quad (65)$$

Or, d'après la figure 4.4, le TRS élémentaire se définit par la relation :

$$TRS_i = \frac{TU}{TR} = \frac{TU}{TU + t_{Ami}} = \frac{TR - t_{Ami}}{TR} \quad (66)$$

Où

TU – Temps utile
TR – Temps Requis
t_{Ami} – Temps arrêts du i^{ème} composant.

$$t_{Ami} = Somme\ des\ temps\ d'arrêts = t_{AP} + t_{Eca} + t_{nq}$$

Où

t_{AP} – temps d'arrêts de production du i^{ème} composant
t_{nq} – temps perdu pour non qualité du i^{ème} composant
t_{Eca} – temps perdu pour ralentissement (écart de cadence) du i^{ème} composant

Mais puisque dans un système comportant plusieurs éléments montés en série, la production à la sortie du n^{ième} composant est dominée par la cadence du composant le plus lent de la chaîne, c'est-à-dire celui qui connaît le plus grand temps d'arrêts, le temps d'arrêt machine du système $t_{Am(S)}$ se définit par :

$$t_{Am_S} = \max\left(t_{Am_i}\right)$$

Le TRS global du système comportant n éléments en série du point de vue Sûreté de Fonctionnement se définit alors par la relation :

$$TRS_{G(série)} = \frac{TR - \max\left(t_{Am_i}\right)}{TR} \quad (67)$$

b) TRS d'un système nominal comportant *n* composants montés en Parallèle

Le schéma d'un système comportant *n* composants montés en parallèle est donné à la figure 4.12. Avec les annotations précédentes, on peut écrire :

$$TRS_G = P\left[Ev_1 + Ev_2 + ... + Ev_i + ... + Ev_n\right]$$

Pour développer cette formule, on définit les événements complémentaires aux événements Ev_i. Le TRS global du système à tout instant *t* donné sera alors défini :

$$TRS_G = 1 - P\left[\overline{Ev_1 + Ev_2 + ... + Ev_i + ... + Ev_n}\right]$$

$$TRS_G = 1 - P\left[\overline{Ev}_1 \cdot \overline{Ev}_2 \cdot ... \cdot \overline{Ev}_i \cdot ... \cdot \overline{Ev}_n\right]$$

Lorsque les événements sont indépendants entre eux, on a :

$$TRS_G = 1 - \prod_{i=1}^{n} (1 - TRS_i) \quad (68)$$

En remplaçant la formule (66) dans (68) on obtient :

$$TRS_G = 1 - \prod_{i=1}^{n} \left(1 - \frac{TR - t_{Am_i}}{TR}\right) \quad (69)$$

Dans un système comportant plusieurs éléments montés en parallèle, le TRS global à la sortie du système est égale à la somme des TRS élémentaires de chaque composant.

$$TRS_G = \sum_{i=1}^{n} TRS_i \quad (70)$$

Deux cas de fonctionnement sont alors possibles :

- *Redondance active (ou encore redondance chaude)*: tous les composants sont normalement en fonctionnement permanent ;

- *Redondance passive (ou encore redondance froide)* : il est nécessaire de mettre en état d'éveil au moins un élément de la chaîne afin de prendre le relais lors de la défaillance du composant en fonctionnement.

- **Redondance chaude**

Dans le cas d'une redondance chaude, deux situations peuvent se présenter : Toutes les machines sont identiques, c'est-à-dire fonctionnent avec la même cadence et fabriquent le même type de produit, ou alors au moins une est différente, c'est-à-dire qu'elle fonctionne avec une cadence différente de celle des autres. Dans les deux cas, le nombre de produit à la sortie du système est égal à la somme des productions de chaque branche. L'efficience du système est alors égale à la somme des efficiences de chacune des branches montées en parallèle. L'expression du TRS_G dans ce cas de figure est donnée par la relation suivante :

$$TRS_G = \sum_{i=1}^{n} TRS_i = \frac{TR - t_{Am1}}{TR} + \frac{TR - t_{Am2}}{TR} + \cdots + \frac{TR - t_{Amn}}{TR}$$

$$TRS_{G\,(parallèleactif)}{}_{c_i = c_j} = TRS_{G\,(parallèleactif)}{}_{c_i \neq c_j} = \frac{nTR - \sum t_{Am_i}}{TR} \quad (71)$$

Où

n - nombre de machines en parallèle

TR – temps requis

t_{Ami} – temps d'arrêts de la $i^{ème}$ machine

- **Redondance froide**

Le schéma d'un système comportant n composants montés en redondance froide est donné à la figure 4.16.

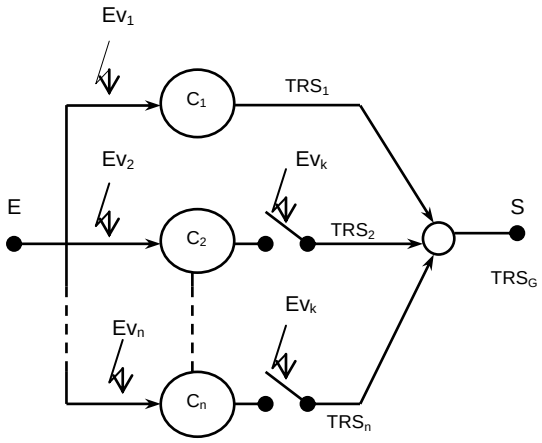


Figure 4.16 : Diagramme parallèle avec redondance froide

Dans le cas de la redondance froide, les commutations peuvent être la cause des dysfonctionnements. Cette fois-ci, les événements à considérer ne sont plus indépendants. Nous allons considérer que les commutateurs sont parfaitement fiables, et que la probabilité de réalisation des événements Ev_k est égale à un (figure 4.16).

La disponibilité d'un composant c_i est égale à la probabilité que ce composant fonctionne à un instant t donné. Or, le TRS_i d'un composant ne peut être évalué que si le composant est en état de fonctionnement, et donc est disponible. Les hypothèses suivantes sont à prendre en compte :

- on suppose qu'un composant fonctionne et que $(n - 1)$ composants sont en attente. Chaque composant en attente est supposé démarrer dès qu'apparaît l'occurrence d'un événement incidentel ou dysfonctionnel entraînant l'arrêt du composant en fonctionnement. Il sera donc caractérisé par une disponibilité instantanée prévisionnelle (en attente à l'arrêt et avant sollicitation) $A_k(\delta)$ à l'instant δ donné. Le composant en attente sera périodiquement testé afin d'évaluer cette disponibilité instantanée prévisionnelle ;
- Il faut connaître la disponibilité générale du système qui elle-même dépend de la disponibilité de chaque composant ;
- la disponibilité du premier composant est évidente, puisqu'il est en fonctionnement ;
- la disponibilité des éléments en attente dépend du résultat du test périodique effectué ;
- tout composant en attente déclaré en panne d'arrêt lors du test est supprimé (cas d'un système non réparable). Cet élément devient alors indisponible ;
- la disponibilité d'un composant en attente est déterminée par deux conditions :
  * La sollicitation due à l'apparition d'un événement incidentel ou dysfonctionnel ayant entraîné l'arrêt du composant en fonctionnement ;
  * La validation de la disponibilité prévisionnelle instantanée du composant en attente juste avant la sollicitation, qui est caractérisée par la probabilité qu'il démarre à l'instant δ donné ;
- nous considérons que les commutations sont parfaitement fiables ;
- nous ne gérons pas ici le choix (l'ordre) du composant à commuter ;
- une ressource de valeur infinie est toujours présente à l'entrée E du système.

Au départ, le composant en attente est déclaré mis en service au temps $t = 0$ et le premier test est effectué au temps T ; le composant était donc en état de fonctionnement au temps $t = 0$. Le paramètre temps est donné à la figure 4.17 [VIL 88].

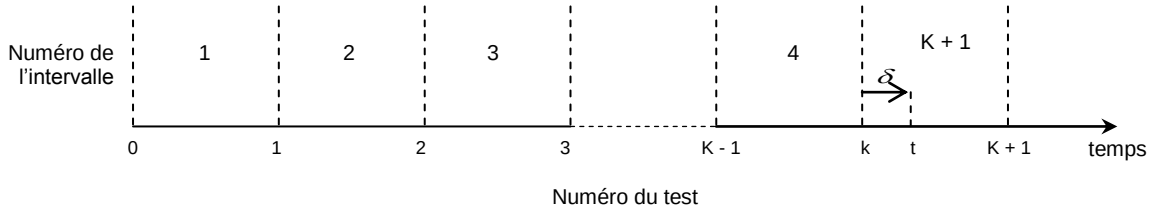


Figure 4.17 : Paramètres temps servant aux tests périodiques des composants en attente dans un système en redondance passive

Considérons l'intervalle de numéro $k+1$: $t = \delta + kT$. Notons $A_k(\delta)$ la disponibilité instantanée prévisionnelle du composant (en attente à l'arrêt et avant sollicitation) à l'instant δ de l'intervalle de temps $[kT, (k+1)T]$. Plaçons-nous immédiatement avant le k-ième test du composant ; il est susceptible d'être indisponible de deux manières différentes :

- il est en réparation à la suite d'une défaillance antérieure (cas d'un système réparable) ;
- il est en panne à la suite d'une défaillance à l'arrêt survenue depuis le test précédent $(k-1)$.

Globalement, cette disponibilité est calculée compte tenu des hypothèses suivantes :

- le composant est testé à intervalles réguliers sous forme d'un démarrage ; celui-ci sollicite le composant qui peut être défaillant lors de cette sollicitation ;
- le composant à l'arrêt est susceptible de connaître des défaillances qui ne seront révélées que par le test suivant l'apparition de cette défaillance ;
- le composant est réparé après constatation d'une défaillance lors d'un test.

Les paramètres qui caractérisent le composant en attente sont :

λ_a : Taux de défaillance à l'arrêt (due à une panne latente) ;

γ_d : Taux de défaillance (à la sollicitation) due au démarrage ;

μ : Taux de réparation ;

T : Durée de l'intervalle entre tests.

La disponibilité instantanée prévisionnelle s'exprime:

$$A_k(\delta) = [1 - (\overline{A}_{k-1}(T) + \gamma_d)(1 - \mu\delta)]e^{-\mu\delta} \qquad (72)$$

Lorsque le système est non réparable, le taux de réparation μ est égal à zéro. D'autre part, le test ne permettant de prendre en compte que les composants disponibles $\overline{A}_{k-1}(T) = 0$. La formule (71) devient :

$$A_k(\delta) = (1 - \gamma_d) \qquad (73)$$

La probabilité que le composant démarre à l'instant δ est :

$$A'_k(\delta) = A_k(\delta)(1-\delta)$$
$$A'_k(\delta) = (1-\gamma_d)(1-\delta) \quad (74)$$

Ces hypothèses nous amènent à conclure que ; la probabilité d'avoir un TRS_G à la sortie du système (figure 4.16) est égale à la probabilité qu'au moins un composant fonctionne, et donc soit disponible.

$$TRS_G = P[Ev_1] + P[Ev_2]\cdot A'_k(\delta) + P[Ev_3]\cdot A'_k(\delta) + \ldots + P[Ev_i]\cdot A'_k(\delta)$$
$$TRS_G = P[Ev_1] + A'_k(\delta)[P[Ev_2] + P[Ev_3] + \ldots + P[Ev_i]]$$

Le système étant en redondance froide, il est impossible que deux événements Ev_i se produisent au même moment. Ils sont incompatibles. On peut donc appliquer le théorème de Poincaré. On obtient :

$$TRS_G = P[Ev_1] + A'_k(\delta)\,P[Ev_2 + Ev_3 + \ldots + Ev_i]$$
$$TRS_G = P[Ev_1] + A'_k(\delta)[1 - P[\overline{Ev_2 + Ev_3 + \ldots + Ev_i}]]$$
$$TRS_G = P[Ev_1] + A'_k(\delta)[1 - P[\overline{Ev}_2 \cdot \overline{Ev}_3 \cdot \ldots \cdot \overline{Ev}_i]]$$

Puisque chaque événement Ev_i à l'entrée de chaque composant génère un TRS_i à sa sortie, on peut écrire :

$$TRS_G = TRS_1 + A'_k(\delta)[1 - \prod_{i=2}^{n}(1 - TRS_i)] \quad (75)$$

Deux cas de figure peuvent alors se présenter: d'abord, le système fonctionne sans redémarrage des composants (cas d'un système non réparable). Dans ce cas, tout composant qui tombe en panne est définitivement exclu du système. Puis, on considère que le système fonctionne avec redémarrage (cas d'un système réparable) c'est-à-dire que lorsqu'un composant tombe en panne, il est réparé et remis en attente. Il peut dans ce cas être à nouveau sollicité si le dernier composant en fonctionnement tombe en panne à son tour. Le système fonctionnera ainsi tant que la période d'observation (temps requis) n'est pas encore atteinte.

- **Redondance froide sans redémarrage**

Dans un système comportant plusieurs composants montés en parallèle en redondance froide sans redémarrage, le TRS_G est égale au ratio de la somme des temps de fonctionnement de chaque composant sollicité, moins la somme des temps de non qualité des mêmes composants, le tout divisé par le temps requis.

$$TRS_{\substack{G(parallèle\ passif)\\ sans\ redémarrage}} = \frac{\sum_{i=1}^{n} TF_i - \sum_{i=1}^{n} t_{nq_i}}{TR} \quad (76)$$

La somme des temps de fonctionnement de tous les composants doit être inférieure ou égale au temps requis.

$$\sum_{i=1}^{n} TF_i \le TR$$

- **Redondance froide avec redémarrage**

Dans un système comportant plusieurs composants montés en redondance froide avec redémarrage, le TRS global du système est égal au rapport de la somme des temps de fonctionnement de tous les composants à chaque démarrage moins la somme des temps perdu pour la non qualité des différents éléments dans les mêmes démarrages, sur le temps requis.

$$TRS_{\substack{G(\text{parallèle passif}) \\ \text{avec redémarrage}}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} TF_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{k} t_{nq_{ij}}}{TR} \tag{77}$$

La somme des temps de fonctionnement de tous les composants à tous les démarrages doit être inférieure ou égale au temps requis.

$$\sum_{i=1}^{n}\sum_{j=1}^{k} TF_{ij} \leq TR$$

Dans un système redondant froid avec redémarrage des composants, l'efficience ne dépend que du temps de fonctionnement, du temps perdu pour non qualité de chaque composant, ainsi que du temps requis. Il ne dépend ni des arrêts de production, ni du temps perdu pour ralentissement (écart de cadence).

c) TRS d'un système nominal comportant *n* composants monté en Assemblage

Le schéma représentant un système comportant des composants montés en assemblage est donné à la figure 4.13. Dans le système Assemblage, le sous système parallèle (C_1, C_2, …, C_n) est monté en série avec le composant assembleur. Et puisque dans un système série, la production est dictée par la machine la plus lente, le TRS global d'un système Assembleur est défini par la relation:

$$TRS_{G(\text{assemblage})} = \min\left\{\frac{nTR - \sum_{i=1}^{n} t_{Am_i}}{TR},\quad TRS_a\right\} \tag{78}$$

Où

n – nombre de composant dans le sous système parallèle

TR - Temps requis

t_{Ami} – Temps d'arrêts machine du $i^{ème}$ composant incluant les pannes, les arrêts fonctionnels, le temps perdu pour non qualité et le temps perdu pour ralentissement (écart de cadence)

TRS_a – TRS du composant assembleur

d) TRS d'un système nominal comportant *n* composants monté en Expansion

Le schéma représentant un système comportant des composants montés en expansion est donné à la figure 4.14. Dans le système Expansion, le sous système parallèle (C_1, C_2, …, C_n) est monté en série avec le composant assurant la fonction d'expansion. Et puisque dans un

système série, la production est dictée par la machine la plus lente, le TRS global d'un système Expansion est défini par la relation:

$$TRS_{G(expansion)} = \min\left\{ TRS_{(e)},\ \frac{nTR - \sum_{i=1}^{n} t_{Am_i}}{TR} \right\} \quad (79)$$

Où

n – nombre de composant dans le sous système parallèle

TR - Temps requis

t_{Ami} – Temps d'arrêts machine du $i^{ème}$ composant incluant les pannes, les arrêts fonctionnels, le temps perdu pour non qualité et le temps perdu pour ralentissement (écart de cadence)

TRS_e – TRS du composant assurant la fonction d'expansion

4.5 CONCLUSION

Les procédures de calculs développées dans ce chapitre ont permis de chercher et de valider les formules mathématiques permettant de calculer le TRS d'un système simple, celui d'un système complexe (série, parallèle, assemblage et expansion). Nous avons tenu compte de l'aspect Productivité et de l'aspect Sureté de Fonctionnement c'est-à-dire en tenant compte des temps d'état d'un moyen de production dont le découpage est fait selon la norme NFE 60-182. S'il est vrai que le TRS avait toujours été calculé de façon globale c'est-à-dire en faisant simplement le produit de ses trois composantes que sont la Qualité, la Performance et la Disponibilité Opérationnelle, nos travaux permettent aujourd'hui de faire une évaluation locale de chaque composant du système. On peut évaluer le TRS global d'un système série en tenant compte des TRS locaux, on peut évaluer le TRS global d'un système parallèle en tenant compte non seulement des TRS locaux, mais également du type de redémarrage. On peut également calculer le TRS global d'un système en expansion ou en assemblage.

Chapitre V

MODÉLISATION TEMPORELLE ET STOCHASTIQUE DU TRS

5.1 INTRODUCTION

L'étude de la Sûreté de Fonctionnement des systèmes complexes, ayant des capacités de reconfiguration, des redondances passives, ou d'autres types de dépendances requiert l'utilisation de modèles comportementaux (ou dynamiques), dans lesquels on modélise explicitement le processus aléatoire qui fait évoluer le système d'état en état, jusqu'à ce qu'il atteigne une catégorie d'états indésirable [BOU 05].

Sur ce type de modèles, on a toujours un phénomène d'explosion combinatoire, dû au nombre d'états à considérer et une fonction exponentielle du nombre de composants du système à étudier. Pour faire face à ce problème, on a souvent recours à la simulation Monte-Carlo [CHA 01] qui est une méthode à la fois très générale et insensible au nombre d'états, ou alors aux réseaux de Petri stochastiques [LIM 99], [BEN 05], [SAL 01], [DJE 03], [ABB 01] et [MOH 06]. Cependant, les résultats qu'ils produisent peuvent être imprécis et demander des temps de calcul prohibitifs pour des systèmes très fiables. En outre, les outils habituels de simulation Monte-Carlo ou de réseaux de Petri stochastiques ne donnent pas de résultats qualitatifs permettant de valider le modèle, tels que les séquences d'événements amenant à un état indésirable [BOU 05]. Cette limitation explique pourquoi nous avons préféré des modèles de type « graphes d'états » [RAK 05], qui, non seulement permettent des calculs des temps de séjour et de comptage des variables, mais aussi peuvent facilement être traduits en AltaRica Data-Flow, haut formalisme de modélisation/simulation.

L'objectif de ce chapitre est de décrire la dynamique des événements caractérisant l'efficience d'un système par des modèles comportementaux (fonctionnel et dysfonctionnel). Cette représentation est faite par des graphes d'état. Ces modèles serviront de support aux calculs (de temps de séjour du système dans chaque état, de comptage de variable) par les modèles de simulation AltaRica Data-Flow (chapitre 6).

Comme dans le cas des réseaux de Petri, chaque graphe est constitué de nœuds et d'arcs. Chaque nœud définit l'état dans lequel se trouve le système. Le passage d'un état à l'autre est possible grâce aux arcs représentant les transitions. A chaque transition est associée une garde qui définit les conditions de franchissement et de valuation pour le passage à l'état suivant. Chaque modèle ainsi défini représente soit un *système de transition* (ensemble de configurations et de transitions permettant de décrire des processus dynamiques), un *automate à contraintes* (automate à états finis usuel dont les états et transitions ne sont pas définis explicitement mais par des contraintes sur les variables) ou un *automate de mode* (9-uplet comprenant : un domaine fini, un ensemble fini de variables (d'état, de flux d'entrée et de sortie), une fonction associant à une variable son domaine, un ensemble fini d'événements, une fonction partielle appelée *transition*, une fonction d'assertion et une fonction définissant les conditions initiales).

La modélisation temporelle et stochastique du TRS permet de représenter la dynamique de sa variation à travers les seuils de la World Class Performance en fonction des fluctuations de ses composantes que sont : la Qualité, la Performance et la Disponibilité Opérationnelle. Cette variation est fonction de la structure du système (série, parallèle, expansion ou assemblage). Une modélisation par automate d'état de la Disponibilité, de la Qualité efficiente ou de la Performance efficiente sera plus facile à réaliser que celle du TRS global d'un composant simple (prise en compte des trois composantes à la fois). Cette modélisation devient plus complexe pour un système série tout comme pour un système redondant.

5.2 MODÉLISATION DE L'EFFICIENCE D'UN SYSTÈME

Globalement, et en tenant compte des temps d'arrêts, on peut définir le TRS d'un système de la manière suivante :

$$TRS = T_q \times T_P \times D_o \tag{80}$$

$$TRS = \frac{T_U}{T_U + t_{nq}} \times \frac{T_N}{T_N + t_{ma}} \times \frac{T_F}{T_F + t_{AP}} \tag{81}$$

Ces deux formulations permettent de présenter le TRS sous deux approches :

- *l'approche stochastique* (formule (80)) qui le présente comme une composition de trois facteurs caractérisés par des taux de variation (dégradation et amélioration) permettant de définir la dynamique de l'efficience du système.
- *l'approche temporelle* qui présente le TRS comme un rapport du Temps Utile sur le Temps Requis (formule (81)). Les comportements fonctionnel et dysfonctionnel du système caractérisés par la variation du TRS comme indicateur de son efficience sont fonction de ces temps d'état. On peut observer à partir de la formule (81) que le TRS est maximal lorsque les temps d'arrêt sont tous nuls et il diminue proportionnellement avec l'augmentation de ceux-ci.

Notre objectif dans ce travail étant d'évaluer l'efficience du système par l'étude comportementale du TRS, nous démontrerons dans la suite des travaux, l'impact des différentes configurations (Série, Parallèle) sur le comportement (fonctionnel et dysfonctionnel) du système.

Deux approches sont alors possibles pour atteindre ces objectifs : *l'approche temporelle* et *l'approche probabiliste*.

Dans l'approche temporelle [CAB 00c] de l'efficience d'un système, le temps est utilisé comme principale variable pour l'évaluation du TRS.

$$TRS = \frac{TU}{TR} = \frac{TR - T_A}{TR} \tag{82}$$

Avec :

TU –Temps Utile ;

TR –Temps Requis ;

T_A –Temps d'Arrêts qui regroupe : les Arrêts induits (manque de pièces, saturation de pièces, manque de personnel, défaut d'énergie et manque de ressource

extérieure), les Arrêts propres qui eux-mêmes comprennent : les Arrêts d'exploitation, les Pannes et les Arrêts fonctionnels (changement de fabrication, contrôle, changement d'outils programmé, réglage fréquentiel et entretien fréquentiel). Ces différents temps d'arrêts sont représentés dans le tableau 4.1 du chapitre 4 § 4.3.2.

D'après la World Class Performance [QMA 08], [AYE 04], [CIM 04], [vTR 03] et [CLE 00], un système est dit efficient lorsqu'il est caractérisé par un $TRS \geq 85\%$. Cette valeur non absolue est celle régulièrement citée dans les littératures dans le cas des systèmes manufacturiers. Elle n'est donc pas standard, tout dépend des objectifs fixés par chaque système d'exploitation. Ce ne sont que des indications dont l'avantage est de caractériser globalement une efficience de production. Ainsi, les exigences pour les trois composantes du TRS sont : $T_q \geq 0,99$, $T_P \geq 0,95$ et $D_O \geq 0,90$. Pour modéliser l'efficience d'un système, nous allons considérer ces valeurs. Les différents seuils de fonctionnement dans chacun des trois modes (*m, m_d, hs*) sont donnés au tableau 5.1.

Tableau 5.1 : Différentes valeurs seuil des composantes du TRS

TRS / Paramètre	≥ 0.85	0.85 < TRS > 0.25	≤ 0.25
T_q	≥ 0.99	$0.99 < T_q > 0.29$	≤ 0.29
T_p	≥ 0.95	$0.95 < T_p > 0.28$	≤ 0.28
D_o	≥ 0.90	$0.90 < D_o > 0.26$	≤ 0.26

Pour modéliser l'efficience d'un système, nous allons tenir compte de cette valeur. La recherche des limites de fonctionnement peut se faire en résolvant l'équation de la formule (82). On obtient alors :

$$TRS = \frac{T_R - T_A}{T_R} \quad \Rightarrow \quad O,85 = \frac{T_R - T_A}{T_R} \quad \Rightarrow \; T_A = \frac{1}{7} T_R \tag{83}$$

D'après la formule (83), tout système reste efficient tant que la durée de ses temps d'arrêts est inférieure à $0,143T_R$. Dès que cette valeur est atteinte, le système change d'état et commence un fonctionnement dégradé. Recherchons maintenant les limites du fonctionnement « acceptable » $T_A = f(T_R)$. La recherche des seuils peut être représentée dans le tableau 5.2 suivant :

Tableau 5.2 : Valeur du TRS en fonction du Temps Requis

TA	0	$\frac{1}{10}TR$	$\frac{1}{9}TR$	$\frac{1}{8}TR$	$\frac{1}{7}TR$	$\frac{1}{6}TR$	$\frac{1}{5}TR$	$\frac{1}{4}TR$	$\frac{1}{3}TR$	$\frac{1}{2}TR$	$\frac{3}{4}TR$	TR
TRS	1	0,90	0,88	0,87	0,85	0,83	0,80	0,75	0,66	0,50	0,25	0

Le tableau 5.2 montre que le TRS d'un système décroît jusqu'à une valeur de 25% lorsque la durée des Temps d'Arrêts est égale aux trois quarts du Temps Requis $T_A = \frac{3}{4}T_R$. Cette valeur peut être considérée comme valeur seuil pour laquelle il vaut mieux arrêter le système (le système est considéré hors service). Nous obtenons finalement trois états de fonctionnement caractéristiques de l'efficience du système:

- l'état de fonctionnement nominal (m): $T_A \leq \frac{1}{7}T_R \Rightarrow TRS \geq 85\%$

- l'état de fonctionnement dégradé (m_d): $T_A > \frac{1}{7}T_R \Rightarrow TRS < 85\%$

- l'état hors service (hs) : $T_A \geq \frac{3}{4}T_R \Rightarrow TRS \leq 25\%$

L'automate d'état correspondant à cette distribution temporelle des temps d'état d'un moyen de production et représentant la vue efficiente du système est donné à la figure 5.1.

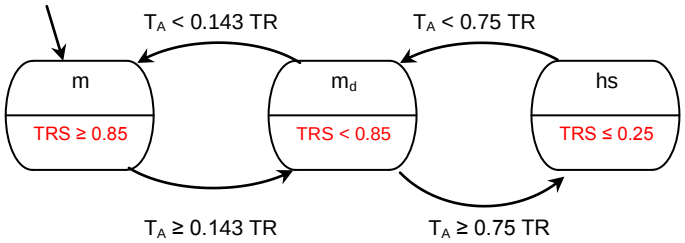


Figure 5.1: Automate d'état d'un système représentant sa vue efficiente

5.3 MODÉLISATION DU TRS D'UN COMPOSANT PAR AUTOMATE

À travers un composant actif circulent des informations de production et les informations de son efficience définies à travers son indicateur de performance qu'est le TRS. Pour des besoins de simplicité et de compréhension, nous allons nous intéresser seulement aux variables TRS dont le comportement renseigne sur l'état du système et sur la dynamique du processus.

L'automate d'état d'un composant (système) simple caractérisé par les valeurs de TRS données au tableau 5.2 est représenté à la figure 5.2. Le système reste dans l'état « *marche* » tant que la valeur du TRS est supérieure ou égale à 85%. À la suite de l'occurrence d'un facteur dégradant la performance du système (perte de Qualité, perte de Disponibilité ou de Performance), la valeur du TRS devient inférieure à 85% et le système bascule dans l'état « *marche dégradée* ». Si l'influence des facteurs dégradants persiste, la valeur du TRS atteint alors un seuil qu'on pourrait juger d'inacceptable (inférieure ou égale à 25%). Le système passe alors dans l'état « *hors service* ». Mais le TRS est un indicateur de performance qui rend compte de l'utilisation effective des moyens de production. Il doit être utilisé dans une politique d'amélioration progressive. Une amélioration (diminution des temps d'arrêts, augmentation de la Disponibilité ou amélioration de la Qualité) permet de ramener le système de l'état « hors service » à celui de « marche dégradée ». Enfin, une reprise effective permettra de passer de l'état « marche dégradée » à l'état « marche ». Il se peut qu'une dégradation sévère (perte totale de la Qualité, de la productivité ou panne grave) survienne

alors que le système est en fonctionnement nominal. Le système passe alors de l'état « marche » à l'état « hors service ». Une remise à l'état neuf ou une amélioration totale de la Qualité ou de la productivité peut également ramener le système de l'état « hors service » à l'état « marche ».

Chacun des trois facteurs (Qualité, Performance et Disponibilité) peut être cause de ce changement d'état. Ces facteurs pourront formellement être associés dans une composition automate - produit synchrone d'automates. Ils seront pris en compte individuellement dans la phase de modélisation, c'est-à-dire qu'on devra modéliser chaque état du système en tenant compte de chaque facteur, puis les associer pour ressortir le comportement global de tout le système. Le raisonnement sera basé sur les temps d'état des moyens de production (vérification par la norme NFE 60-182).

Nous continuons de mentionner que les valeurs seuil représentées dans le tableau 5.2 ne sont prises ici que comme exemple. Il n'existe pas de seuil prédéfini pour la considération du TRS. Nous sommes partis d'une valeur régulièrement citée qui est de 85% [QMA 08], [AYE 04], [WON 07] et [CLE 00]. D'autres valeurs peuvent également être prises en compte, tout dépend du système de production et du domaine dans lequel on se trouve.

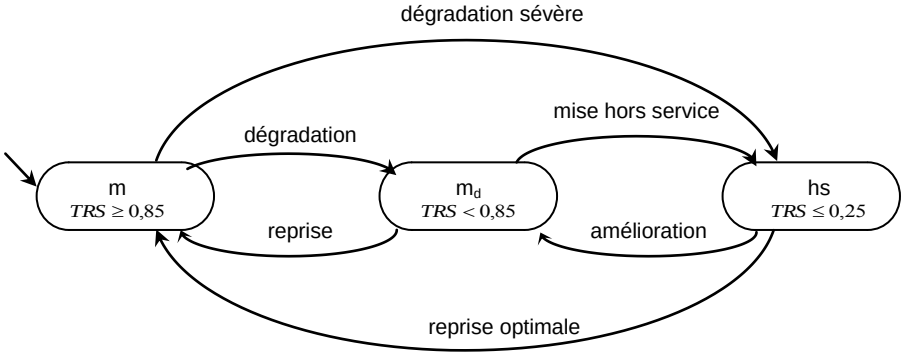


Figure 5.2: Le modèle automate du TRS d'un composant réparable

Dans l'exemple présenté en figure 5.2, nous avons considéré les valeurs suivantes :

- TRS_{max} = 85%, TRS_{min} = 25% ;
- état « *marche* » → TRS ≥ 0,85 c'est-à-dire TA ≤ 0,143 TR;
- état « *marche dégradée* » → 0,85 < TRS > 0,25 c'est-à-dire 0,143 TR < TA > 0,75 TR ;
- état « *hors service* » → TRS < 0,25 c'est-à-dire TA ≥ 0,75 TR.

Les événements et les gardes associées aux différentes transitions de la figure 5.2 sont représentés dans le tableau 5.3.

Tableau 5.3 : Evénements et gardes associées aux transitions de la figure 5.2

Evénement	Garde associée à chaque transition
dégradation	$T_A > 0{,}143 T_R$
reprise	$T_A \leq 0{,}143 T_R$
mise hors service	$T_A \geq 0{,}75 T_R$
amélioration	$T_A < 0{,}75 T_R$
dégradation sévère	$T_A \geq 0{,}75 T_R$
reprise optimale	$T_A \leq 0{,}143 T_R$

Les différents événements et gardes associées représentés au tableau 5.2 permettent d'obtenir le système de transitions représenté à la figure 5.3.

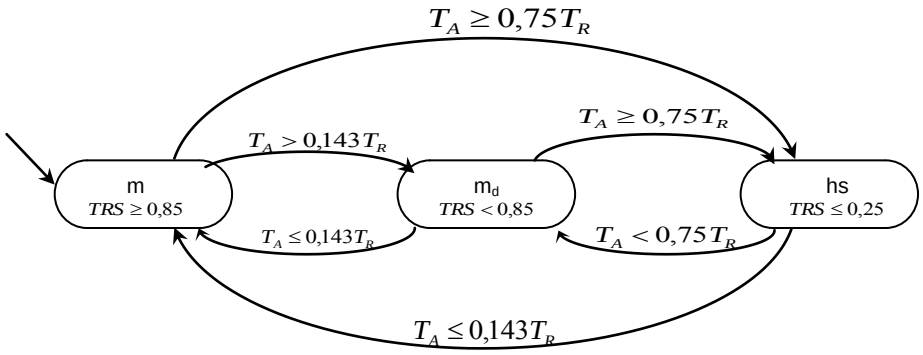


Figure 5.3: Système de transitions du modèle TRS d'un composant réparable

5.3.1 Cas d'un composant non réparable

Le TRS est un indicateur (global / local) de l'efficience d'un système, il renseigne sur l'état du système et/ou de ses composants. Il sera donc modélisé comme variable d'état. Ses paramètres qui sont définis par les temps d'état, pourront à la fois être considérés comme variables d'état (valuation locale du TRS) et de flux (mise à jour des transitions) (figure 5.4).

Figure 5.4: Modélisation du TRS d'un composant

Un composant non réparable n'a pas d'état de fonctionnement dégradé ; Soit il est en fonctionnement normal (TRS > 0,25), soit il est hors service à la suite d'une dégradation sévère (TRS < 0,25). On peut donc le modéliser de la façon suivante (figure 5.5):

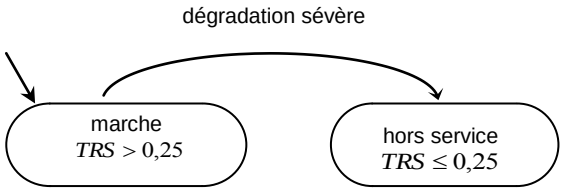


Figure 5.5 : Automate d'état du TRS d'un composant non réparable

5.3.2 Cas d'un composant réparable

Dans ce cas, les trois états de fonctionnement sont pris en compte. L'automate d'état d'un tel composant est donné à la figure 5.1.

En pratique, l'état m_d est un état important, car il est déclencheur de la prise de décision. Le passage à cet état dans le sens de la dégradation est indicateur de l'occurrence d'un événement redoutable ou critique entraînant l'arrêt ou la perte totale ou partielle du système.

5.3.3 Modélisation du TRS d'un système comportant plusieurs composants montés en série

Pour les besoins de simplicité, nous allons considérer pour ce cas que les deux composants sont soit réparables, soit non réparables. Nous considérons que chaque composant est caractérisé par un TRS local (TRS_i) qui renseigne sur la sortie locale o_i. Le TRS global (TRS_G) à la sortie du système est le produit des TRS_i de tous les composants de la chaîne (formule (65), chapitre 4, § 4.4.3).

La sortie o du système est égale à : $o = o_1 \wedge o_2 \wedge ... \wedge o_n$.

a) Cas d'un système comportant deux composants non réparables

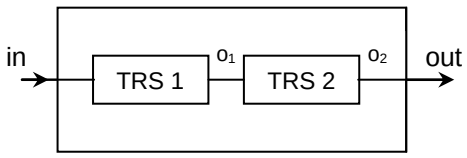


Figure 5.6 : Système comportant deux composants en série

Tout comme dans le cas d'un seul composant non réparable, le graphe d'état de ce système ne présente pas de mode dégradé. On obtient l'automate d'état de la figure 5.7. Le seul état délivrant une sortie nominale est l'état initial correspondant à celui où chacun des composants délivre un TRS_i local. Dans les deux autres situations, dès que l'un au moins des

deux composants de la chaîne connait une sévère dégradation, tout le système passe à l'état « hors service », et aucune sortie n'est délivrée.

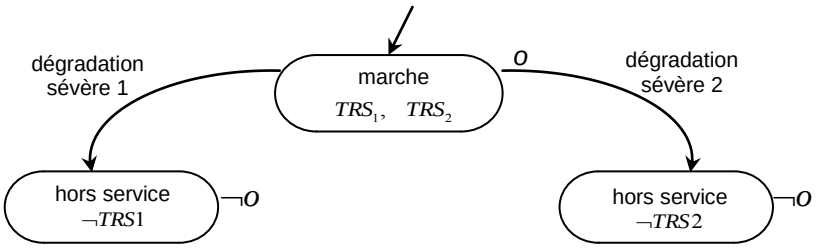


Figure 5.7: Automate d'état d'un système comportant deux composants non réparables montés en série

b) Cas d'un système comportant deux composants réparables

Pour un système réparable, les trois états (m, m_d, hs) sont pris en compte pour chaque composant de la chaîne. Ceci signifie que chaque composant est modélisé par un automate à 3 états. La prise en compte des différents états de chaque composant donne finalement 9 combinaisons d'états pour le système et une explosion de transitions entre états. D'une façon générale, pour n composants dans la chaîne, on obtient 3^n états. Pour éviter cette explosion d'états due au fait qu'on part de 3 états pour un seul composant, et afin de rendre le graphe plus lisible, nous représenterons graphiquement la dynamique du processus en modélisant chaque état du système par une « *macro état* » comportant en interne chacune des combinaisons d'états des n composants en série. La « *macro état* » est un état composite dont la valuation entraîne soit la marche, soit l'arrêt du système.

Le symbole de cette macro état est donné par la figure 5.8.

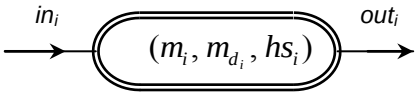


Figure 5.8 : Symbole d'une macro état

Dans le cas d'un système comportant deux composants réparables, les différents états probables dans lesquels peut se trouver le système sont représentés dans la figure 5.9.

C_2 \ C_1	m_1	m_{d1}	hs_1
m_2	O	O_d	arrêt
m_{d2}	O_d	O_d	arrêt
hs_2	arrêt	arrêt	arrêt

Figure 5.9 : Différents états d'un système série à deux composants réparables

Les différents états de fonctionnement dans lesquels peut se trouver un pareil système sont regroupés dans le tableau 5.4.

Tableau 5.4 : Etat d'un système série à deux composants réparables

État	TRS_G	Sortie du syst.	État du système
m_1m_2	TRS_1, TRS_2	o	Fonctionnement normal
$m_{d1}m_2$	TRS_{d1}, TRS_2	o_d	Fonctionnement dégradé
hs_1m_2	$\neg TRS_1, TRS_2$	$\neg o$	Hors service
m_1m_{d2}	TRS_1, TRS_{d2}	o_d	Fonctionnement dégradé
$m_{d1}m_{d2}$	TRS_{d1}, TRS_{d2}	o_d	Fonctionnement dégradé
hs_1m_{d2}	$\neg TRS_1, TRS_{d2}$	$\neg o$	Hors service
m_1hs_2	$TRS_1, \neg TRS_2$	$\neg o$	Hors service
$m_{d1}hs_2$	$TRS_{d1}, \neg TRS_2$	$\neg o$	Hors service
hs_1hs_2	$\neg TRS_1, \neg TRS_2$	$\neg o$	Hors service

Le système de transitions de ce système est donné à la figure 5.10.

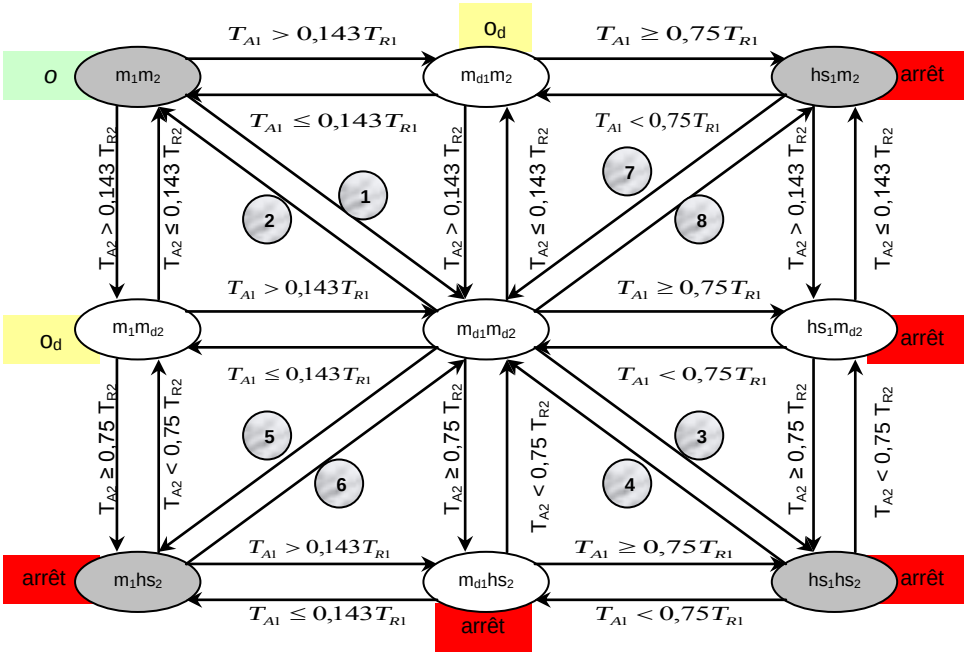


Figure 5.10 : Système de transitions de la macro état d'un système série à deux composants réparables

Le tableau 5.5 complète les transitions ne pouvant pas être mentionnées dans la figure 5.10 (numérotées de 1 à 8).

Tableau 5.5 : Transitions non mentionnées sur la figure 5.10

N°	Transitions	État de départ	État d'arrivée
1	$(TA_1 > 0{,}143TR_1) \wedge (TA_2 > 0{,}143TR_2)$	(m_1m_2)	$(m_{d1}m_{d2})$
2	$(TA_1 \leq 0{,}143TR_1) \wedge (TA_2 \leq 0{,}143TR_2)$	$(m_{d1}m_{d2})$	(m_1m_2)
3	$(TA_1 \geq 0{,}75TR_1) \wedge (TA_2 \geq 0{,}75TR_2)$	$(m_{d1}m_{d2})$	(hs_1hs_2)
4	$(TA_1 < 0{,}75TR_1) \wedge (TA_2 < 0{,}75TR_2)$	(hs_1hs_2)	$(m_{d1}m_{d2})$
5	$(TA_1 \leq 0{,}143TR_1) \wedge (TA_2 \geq 0{,}75TR_2)$	$(m_{d1}m_{d2})$	(m_1hs_2)
6	$(TA_1 > 0{,}143TR_1) \wedge (TA_2 < 0{,}75TR_2)$	(m_1hs_2)	$(m_{d1}m_{d2})$
7	$(TA_1 < 0{,}75TR_1) \wedge (TA_2 > 0{,}143TR_2)$	(hs_1m_2)	$(m_{d1}m_{d2})$
8	$(TA_1 \geq 0{,}75TR_1) \wedge (TA_2 \leq 0{,}143TR_2)$	$(m_{d1}m_{d2})$	(hs_1m_2)

L'automate d'états de ce système est présenté à la figure 5.11. Puisque le système est réparable, les événements *perte* TRS_i et *reprise* TRS_i permettent de définir la dynamique comportementale fonctionnelle et dysfonctionnelle du système. Chaque macro état est caractérisé par la délivrance (ou la non délivrance) d'une sortie en fonction de la délivrance ou non d'un TRS_G qui lui-même le produit des TRS_i dans chaque macro état. La délivrance d'un TRS_i dépend de la combinaison des états (m_i, md_i, hs_i) des deux composants (figure 5.9).

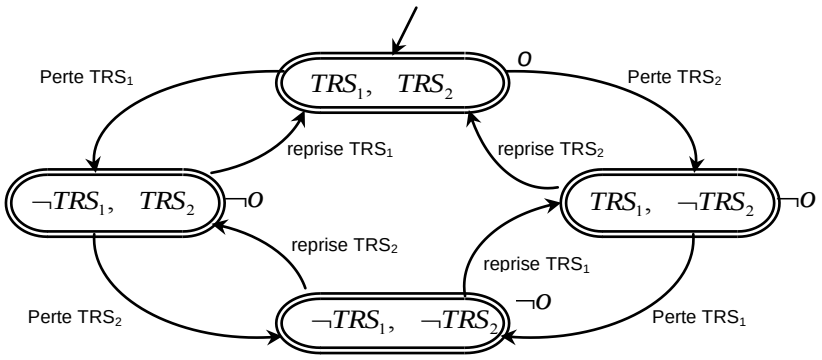


Figure 5.11 : Automate d'état d'un système comportant deux composants réparables montés en série

5.3.4 Automate d'un système comportant plusieurs composants montés en parallèle

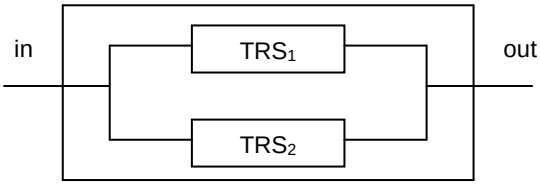


Figure 5.12 : système comportant deux composants en parallèle

a) Cas d'une redondance active

Dans ce cas, les deux composants sont actifs en même temps. Pour des raisons de simplicité, nous allons considérer dans la suite des hypothèses que les deux composants sont non réparables. Les différents états dans lesquels peut se trouver le système sont résumés dans la figure 5.13.

C_2 \ C_1	m_1	m_{d1}	hs_1
m_2	o	o_d	o_d
m_{d2}	o_d	o_d	o_d
hs_2	o_d	o_d	arrêt

Figure 5.13 : Différents états d'un système redondant actif à deux composants non réparables

La figure 5.13 montre une amélioration fonctionnelle du système en redondance active par rapport au système série (figure 5.9). On constate une diminution des états d'arrêt du système au profit d'états dégradés.

Le système de transitions d'un système redondant actif sera le même que celui de la figure 5.10 à la seule différence que toutes les fonctions en ET sur les gardes des transitions seront remplacées par des fonctions en OU. L'automate d'état correspondant à ce système est donné à la figure 5.14.

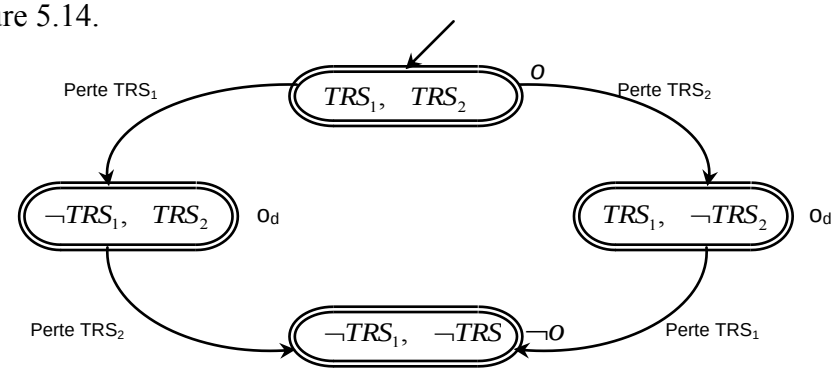


Figure 5.14 : Automate d'état d'un système comportant deux composants en redondance active

b) **Cas d'une redondance passive**

Dans ce cas, un composant au moins est au repos et prend la relève dès que celui en fonctionnement devient défaillant. Nous allons considérer que les deux composants sont non réparables. La figure 5.15 présente les différents états de ce système.

C_2 \ C_1	m_1	m_{d1}	hs_1
m_2	o	o	o
m_{d2}	o	o_d	o_d
hs_2	o	o_d	arrêt

Figure 5.15 : Différents états d'un système en redondance passive à deux composants non réparables

La figure 5.15 nous permet d'éliminer les modes non autorisés. En effet, Les conditions initiales considèrent les deux composants non défaillants avant la sollicitation, puisque les deux composants ne peuvent pas être actifs au même moment et qu'un composant non sollicité ne peut pas être en mode dégradé, les modes (m_1m_2, $m_{d1}m_{d2}$, $m_{d1}m_2$, m_1m_{d2}) ne sont pas autorisés. Les modes (a_1m_{d2}, $m_{d1}a_2$) donnent des sorties dégradées. Le graphe d'états se réduit à trois macros états. Il est représenté à la figure 5.16.

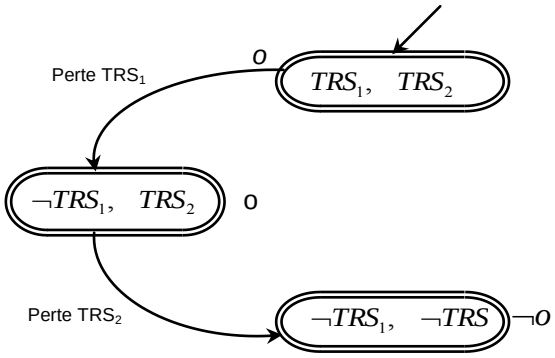


Figure 5.16 : Automate d'état d'un système comportant deux composants en redondance passive

Comme nous l'avons annoncé au chapitre 5 § 5.2, l'évaluation de l'efficience d'un système par l'étude comportementale du TRS est basée sur deux approches: *l'approche temporelle* et *l'approche stochastique*.

5.4 MODÉLISATION COMPORTEMENTALE TEMPORELLE DE L'EFFICIENCE D'UN SYSTÈME : APPROCHE TRS « *TEMPOREL* »

La modélisation comportementale temporelle de l'efficience d'un système de production peut se faire en tenant compte des trois composantes du TRS que sont : la Qualité, la Performance et la Disponibilité Opérationnelle.

5.4.1 Modélisation de l'efficience en fonction de la Disponibilité Opérationnelle

Pour modéliser un système de production en ne tenant compte que de sa Disponibilité Opérationnelle, on peut le considérer dans l'un des états suivants : *marche*, *arrêt*, *panne*, *marche dégradée* et *hors service*. Le passage de l'état marche dégradée se produit lorsque le système a longtemps séjourné soit dans l'état panne, soit dans l'état arrêt. Cette commutation s'effectue lorsque la garde $T_A \geq 0{,}143TR$ de la figure 5.1 est tirée. Le passage de l'état marche dégradée à l'état hors service s'effectue lorsque la garde $T_A \geq 0{,}75TR$ est tirée.

Deux types de transitions sont utilisés pour modéliser l'efficience d'un système en ne tenant compte que de cette composante du TRS : les transitions probabilistes dont les événements sont caractérisés par des valeurs constantes du taux de défaillance λ, ainsi que du taux de réparation μ. Ceci concerne les événements *panneDétectée, réparation et réparationOptimale*. Ces événements sont caractérisés par des lois exponentielles. D'autres transitions sont dites déterministes. Il s'agit des transitions dont les gardes sont des valeurs constantes du temps. On citera par exemple les événements *arrêtDétecté*, *remiseEnMarche*, *arrêtProlongé, panneProlongée,* etc. Ces événements sont caractérisés par des fonctions de Dirac. L'automate d'état correspondant à ce modèle est donné à la figure 5.17.

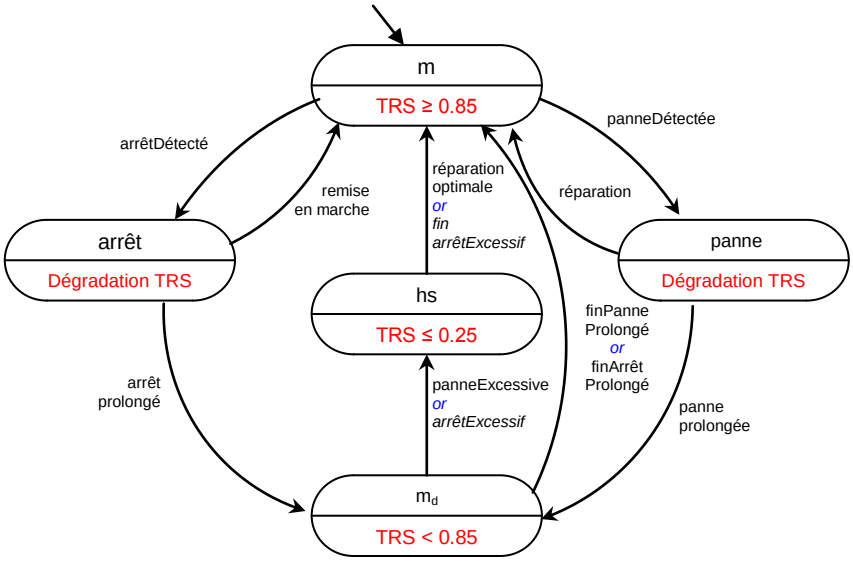


Figure 5.17 : Modèle automate de l'efficience d'un système en fonction de sa Disponibilité Opérationnelle

Initialement, le système est dans l'état *marche*. La valeur du TRS dans cet état est nominale et supérieure ou égale à quatre vingt cinq pour cent. Dès qu'une panne est détectée, le système passe à l'état *panne* sur événement *panneDétectée*. La valeur du TRS connait

alors une dégradation. Le système est supposé rester dans cet état tant que la valeur du TRS n'est pas inférieure à quatre vingt cinq pour cent et tant que l'intervention du réparateur reste dans la limite du temps de réparation imparti. Il revient à l'état de marche dès la fin de la réparation sur événement *réparation*. Lorsque l'échéance d'un arrêt programmé arrive et que l'arrêt est exécuté par l'Opérateur, le système passe à l'état *arrêt* sur événement *arrêtDetecté*. La valeur du TRS connaît également une dégradation. Il est supposé rester dans cet état tant que la valeur du TRS n'est pas inférieure à quatre vingt cinq pour cent et tant que la durée d'intervention de l'Opérateur reste dans le temps imparti. Il pourra revenir à l'état de marche dès la fin de l'arrêt sur événement *remiseEnMarche*. Une faute ou une négligence de l'Opérateur maintient le système dans l'état *arrêt* au-delà du temps imparti. La valeur du TRS chute alors en deçà de 85% et le système bascule à l'état *marche dégradé* sur événement *arrêtProlongé*. Une incompétence du Réparateur fait également basculer le système dans cet état sur événement *pannеProlongée*. Lorsque les gardes des événements *finPanneProlongé* et *finArrêtProlongé* sont tirées, le système revient à l'état de marche, si non il bascule à l'état *hors service* sur événement *panneExcessive* ou *arrêtExcessif*. La valeur du TRS dans cet état atteint alors un seuil que l'on pourrait juger d'inacceptable de 25%. Une réparation optimale ou une fin d'arrêt excessif ramène le système à l'état marche sur événements *réparationOptimale* ou *finArrêtExcessif*.

5.4.2 Modélisation de l'efficience en fonction de la Qualité

La Qualité est l'une des trois composantes du TRS. C'est le plus important car son poids dans la valeur du TRS doit être assez élevé pour un système efficient ($T_q \geq 99\%$). Le modèle automate permettant de caractériser l'efficience d'un système en fonction de la Qualité est donné à la figure 5.18. Le système passe de l'état *marche* à l'état *qualitéDégradée* sur événement *perteQualité*. L'événement *finPerteQualité* peut ramener le système à l'état marche, si non, il bascule à l'état *marcheDégradé* sur événement *perteQualitéProlongée*. L'événement *finPerteQualité* ramène le système en état de marche, si non il bascule à l'état *hors service* sur événement *perteQualitéExcessive*. La garde de cette transition est tirée lorsque la valeur du TRS chute à 25%. Le retour à l'état *marche* est conditionné par l'événement *finPerteQualitéExcessive*.

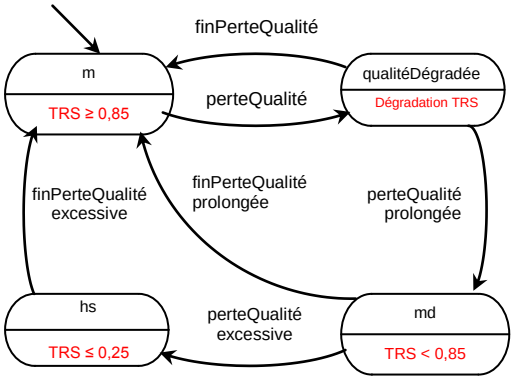


Figure 5.18: Modèle automate de l'efficience d'un système en fonction de la Qualité

5.4.3 Modélisation de l'efficience en fonction de la Performance

D'après la World Class Performance, tout système est dit efficient lorsqu'il est caractérisé par un taux de performance $T_P \geq 0{,}95$. Le modèle automate de l'efficience d'un système en fonction de la Performance est similaire à celui de la figure 5.18. Le système passe de l'état *marche* à l'état *performanceDégradée* sur événement *pertePerformance*. Dès que cessent les écarts de cadence et les ralentissements, le système revient à l'état marche sur événement *finPertePerformance.* Si les causes ayant entraîné les ralentissements persistent, le système bascule à l'état *marcheDégradé* sur événement *pertePerformanceProlongée.* L'événement *finPertePerformanceProlongé* peut ramener le système à l'état marche, si non il bascule à l'état *hors service* sur événement *pertePerformanceExcessive.* La garde de cette transition est tirée lorsque la valeur du TRS chute à 25%. Le retour à l'état marche est conditionné par l'événement *finPertePerformanceExcessive*.

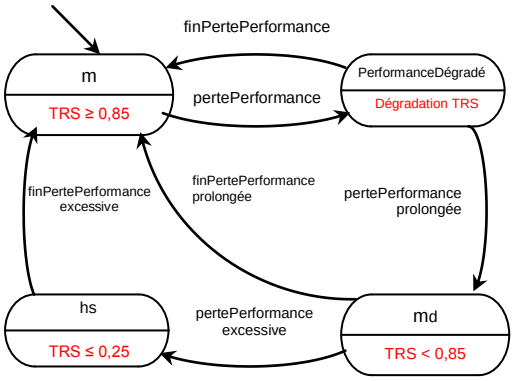


Figure 5.19: Modèle automate de l'efficience d'un système en fonction de la Performance

5.4.4 Modélisation temporelle de l'efficience d'un système simple réparable

Le modèle automate de l'efficience d'un système en fonction de la valeur du TRS et de la dynamique de ses 3 composantes est donné à la figure 5.20. Formellement, ce modèle est le produit synchrone G = G1 x G2 x G3 des figures 5.17 (automate étendu G1), 5.18 (automate étendu G2) et 5.19 (automate étendu G3). Les paramètres et les différentes lois associés aux transitions de l'automate étendu de la figure 5.20 sont représentés au tableau 5.6.

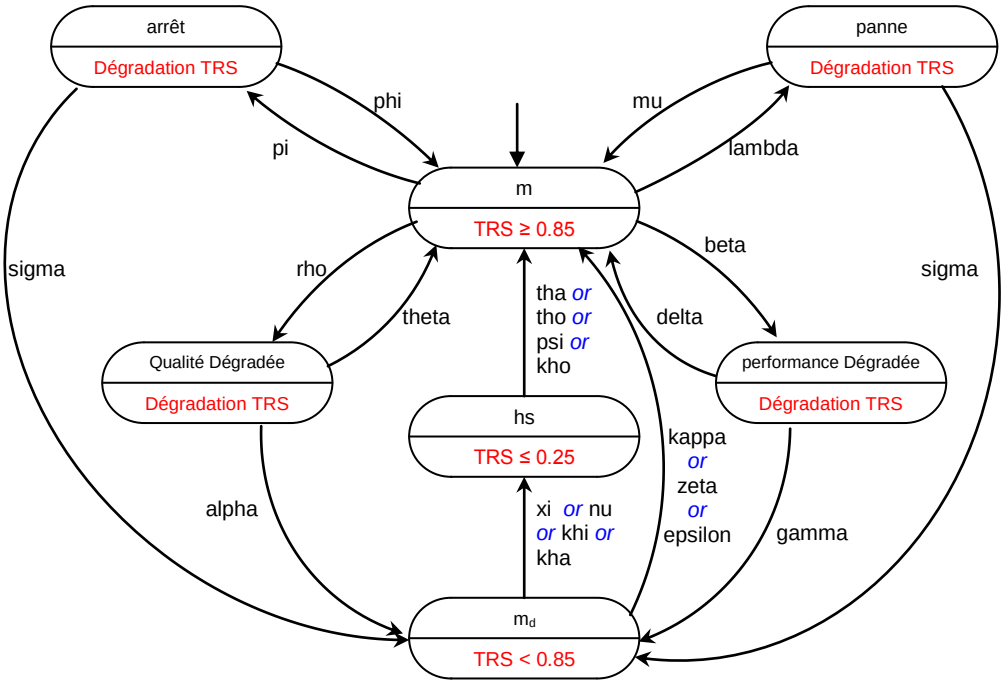


Figure 5.20: Modèle automate de l'efficience d'un système simple réparable

Tableau 5.6: Paramètres et différentes lois associés aux transitions de l'automate de mode de la figure 5.20

Evénement	Paramètre	Fonction
panneDetectée	lambda	exponential
reparation	mu	exponential
arrêtDetecté	pi	Dirac
remiseEnMarche	phi	Dirac
panneProlongée	sigma	Dirac
finPanneProlongée	kappa	Dirac
arrêtProlongé	sigma	Dirac
finArrêtProlongé	kappa	Dirac
arrêtExcessif	xi	Dirac
finArrêtExessif	tha	Dirac
panneExcessive	nu	Dirac
réparationOptimale	tho	exponential
perteQualité	rho	exponential
finPerteQualité	theta	exponential
perteQualitéProlongée	alpha	Dirac
finPerteQualitéProlongée	zeta	Dirac
perteQualitéExcessive	khi	Dirac
finPerteQualitéExessive	psi	Dirac
pertePerformance	beta	exponential
finPertePerformance	delta	exponential
pertePerformanceProlongée	gamma	Dirac
finPertePerformanceProlongée	epsilon	Dirac
pertePerformanceExcessive	kha	Dirac
finPertePerformanceExcessive	kho	Dirac

Le séjour du système dans les états *panne* et *arrêt* est tributaire des compétences de l'Opérateur et du Réparateur. Une faute ou la négligence de l'un et l'incompétence de l'autre fait basculer le système à l'état *marche dégradée* puis finalement à l'état *hors service*. Les comportements de l'Opérateur et du Réparateur sont modélisés par les automates des figures 5.21 et 5.22 respectivement.

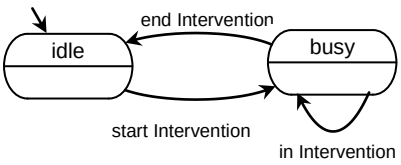


Figure 5.21: Automate de mode de l' Opérateur

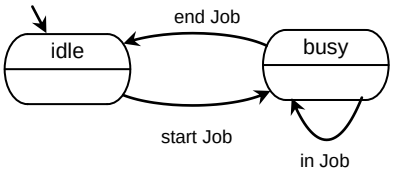


Figure 5.22: Automate de mode du Réparateur

La concordance d'événements entre le fonctionnement du système (figure 5.20) et les comportements du Réparateur et celui de l'Opérateur (figure 5.21 et 5.22) est assurée par une synchronisation entre les transitions associées. Par exemple, une synchronisation doit être établie entre les événements *arrêtDétecté* et *startIntervention*, de même qu'entre *remiseEnMarche* et *endIntervention*. Une autre synchronisation doit exister entre *panneDétectée* et *startJob* de même qu'entre *réparation* et *endJob*. Cette synchronisation permet de déclencher simultanément la mise en service de l'Opérateur et du Réparateur et les événements du système qui les interpellent. On considère qu'ils interviennent dès que la panne ou l'arrêt sont détectés, ce qui n'est pas toujours vrai dans le cas pratique. Dans ce dernier cas, on devrait modéliser le système en ajoutant un état intermédiaire qui permet de gérer l'attente de l'intervention comme illustré en figure 5.23. Cet automate est également un produit synchrone des automates étendus des figures 5.21 et 5.22.

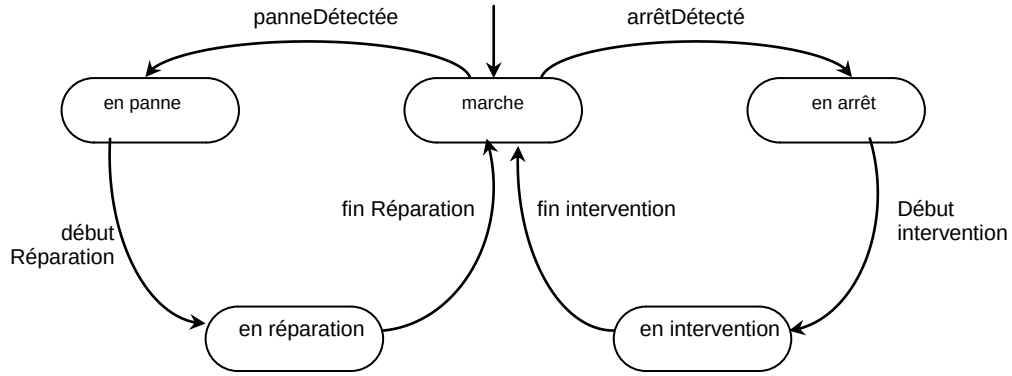


Figure 5.23 : Automate de mode des différents temps d'arrêt du système en tenant compte des indisponibilités du Réparateur et de l'Opérateur

5.5 MODÉLISATION COMPORTEMENTALE STOCHASTIQUE D'UN SYSTÈME : APPROCHE TRS « *STOCHASTIQUE* »

Dans l'approche TRS stochastique, les trois composantes du TRS que sont : la Qualité, la Performance et la Disponibilité (formule (80)), sont prises en compte individuellement. Il est intéressant de connaître l'impact de la variation de chacun de ces composants sur les comportements fonctionnel et dysfonctionnel du système. Cette modélisation stochastique de la variation du TRS à l'intérieur et à l'extérieur de chaque état permet de mieux appréhender la dynamique de la simulation stochastique de chaque modèle en AltaRica Data-Flow. Ces trois composantes peuvent se définir de la façon suivante :

La Qualité

La Qualité est définie comme étant la mesure dans laquelle un produit spécifique est adapté aux exigences auxquelles il est destiné à répondre [VIL 88]. Elle est caractérisée, non seulement par sa conformité aux spécifications qui la définissent, mais aussi et surtout par son aptitude à rester conforme à ces spécifications pendant sa durée de vie. La Qualité désigne aussi la conformité du produit à sa spécification à sa sortie d'usine.

La Qualité est un indicateur pertinent de l'efficience d'un système. Avec un taux d'évaluation (rapport du nombre de produit bon sur le nombre de produit fabriqué) de 99% (valeur internationale), elle se présente comme étant la composante la plus exigible du TRS. La moindre variation de la Qualité entraîne automatiquement la perte de l'efficience du système.

La Performance

La Performance peut être considérée comme l'efficacité mais aussi l'efficience. Autrement dit, la réalisation du but final ne suffit pas, la manière d'atteindre ce but doit être également jugée. Ainsi on converge vers une autre vision de la performance : le triptyque *objectif – résultats – moyens* dont l'efficacité mesure les résultats par rapport aux objectifs. L'efficience tient compte des moyens employés pour obtenir les résultats et la pertinence prend en balance les moyens par rapport aux objectifs. Un projet doit fournir plus de richesse qu'il en a reçu (effet surgénérateur), tant au niveau humain, financier et technique, que pour tous les acteurs, internes ou externes, à l'entreprise.

La notion de performance intègre également plusieurs autres aspects. Ainsi l'approche processus et le capital intellectuel (qui intègre l'aspect humain) sont des démarches à tenir compte lors d'une évaluation de la performance. La Performance est donc différente d'un système à l'autre. Elle dépend fortement d'un ensemble de paramètres, numériques ou non, tels que la taille de l'entreprise, les priorités fixées par l'entreprise, etc. Configurer de tels systèmes consiste à attribuer une valeur à chaque paramètre afin d'optimiser un critère de performance.

La Disponibilité

La Disponibilité (encore appelée *Availability* en anglais) est l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données et à un instant donné [VIL 88]. Elle est généralement mesurée par la probabilité qu'une entité E soit en état d'accomplir une fonction requise dans des conditions données et à une instante *t* donnée :

$$A(t) = P[E\ non\ défaillante\ à\ l'instant\ t]$$

La Disponibilité Opérationnelle est fonction des temps d'état des moyens de production. Considérée comme le rapport du temps de fonctionnement sur le temps requis, elle est influencée notamment par les arrêts de production. Ces arrêts de production comprennent : les Arrêts Induits (manque de pièces, saturation de pièces, manque de personnel, défaut d'énergie et manque de ressource extérieure), les Arrêts propres qui eux-mêmes comprennent : les Arrêts d'exploitation, les Pannes et les Arrêts fonctionnels (changement de fabrication, contrôle, changement d'outils programmé, réglage fréquentiel et

entretien fréquentiel). La Disponibilité Opérationnelle reste donc la composante la plus délicate de cette composition car dépendant des états dont les événements initiateurs sont dans la plupart aléatoires.

Les trois états de fonctionnement du système (marche, marche dégradée et hors service) restent valables tel que illustré à la figure 5.1. L'interprétation de ce schéma se fait de la façon suivante :

L'état *marche*

Dans cet état, la valeur du TRS est nominale et reste supérieure à une valeur minimale admissible selon le domaine d'exploitation (par exemple 85%). Pour cette valeur, les trois composantes ont des contributions différentes. Le Taux de Qualité doit être au moins égale à 99%, le Taux de Performance de 95% et la Disponibilité Opérationnelle de 90%. Toutes ces valeurs sont des valeurs internationales admises pour un TRS de 85% [CIM 04], [AYE 03a], [QMA 08], [CL E00] et [vRT 03].

L'état *marche dégradé*

Le système passe dans cet état lorsque, en dégradation, l'une au moins des composantes passe en dessous des valeurs indiquées précédemment. Il peut également revenir dans cet état lorsqu'après avoir séjourné dans l'état *hors service*, sur amélioration en qualité de l'un au moins des ces composantes.

L'état *hors service*

C'est l'état dans lequel le système ne produit plus aucune activité. Toutes les interventions faites dans cet état vont dans le sens de l'amélioration des facteurs dégradants l'efficience. Un long séjour du système dans cet état présente des conséquences néfastes énormes pour l'entreprise.

5.5.1 Le modèle stochastique de l'efficience d'un système de production

La variation de chacune des composantes : Qualité, Performance et Disponibilité Opérationnelle obtenue à partir des formules de calcul présentées au chapitre 4, entraîne automatiquement celle du TRS. Cette variation peut donc être conséquente aux comportements individuels ou combinés de ces composantes. Puisque les événements initiateurs de ces variations peuvent soit s'améliorer soit se détériorer, on assiste à une dynamique interne dans chaque état du système. Le passage à l'un ou l'autre état n'est effectif que si la valeur du TRS (garde de chaque état) est franchie. Le système ainsi défini dans le cas d'un composant simple est représenté à la figure 5.24.

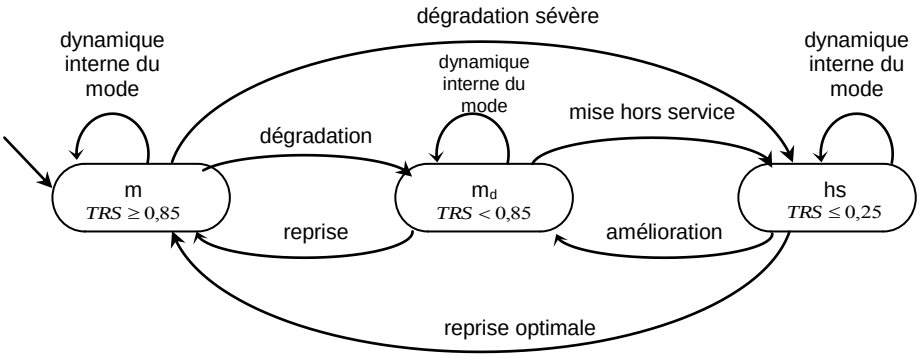


Figure 5.24 : Le modèle stochastique du TRS

La figure 5.25 représente le système de transition associé au modèle automate de la figure 5.24. Elle ressort non seulement la dynamique entre les différents états du système, mais aussi celle à l'intérieur de chaque état. Cette variation de chaque paramètre non seulement détermine le passage d'un état à l'autre, mais aussi définit la durée du séjour dans chacun des états. Au niveau global du système, la variation interne n'est pas visible. Elle n'est appréciée de l'extérieur que par la variation du TRS.

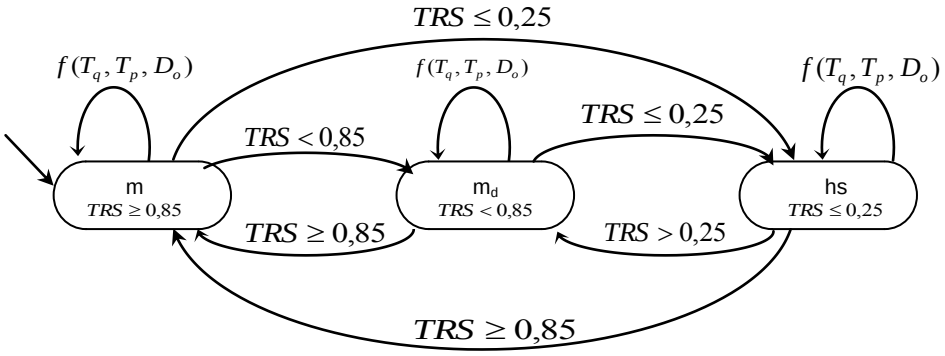


Figure 5.25 : Le modèle stochastique du TRS en fonction des dynamiques transitoires

Le modèle d'état associé à un pareil système devient très complexe, car chaque état représente en soit un macro système composé de micro systèmes qu'il faut également modéliser. Le facteur dégradant du comportement interne de chaque état est une combinaison de facteurs dégradants dont il convient de modéliser les événements individuellement. Nous utiliserons pour cette modélisation la méthode dite « *d'états fictifs* » [COC 92], [CAB 99b] et [CAB 00a] représentant le modèle de *Cox* qui lui-même est une généralisation du modèle d'*Erlang* [SAL 06].

5.5.2 Modèle Coxian de l'efficience d'un système

La perte ou l'amélioration d'une des composantes du TRS dans un état est automatiquement observée de l'extérieur par sa variation. Nous appellerons « *défaillance* », la

perte des caractéristiques d'au moins une composante du TRS entraînant le passage d'un état nominal amont vers un état dégradé aval. Nous appellerons de la même façon « *réparation* », le passage d'un état dégradé aval vers un état nominal amont. Les valeurs seuil permettant la variation de la valeur du TRS en fonction de celle des trois composantes (Qualité, Performance et Disponibilité Opérationnelle) sont données au tableau 5.1.

Ainsi, la fonction « défaillance » comprendra les paramètres : perte de la Qualité, perte de la Disponibilité et perte de la Performance. Pour le passage de l'état « marche » à l'état « marche dégradée », on aura le paramètre Λ tel que $\Lambda = f(\lambda_q, \lambda_p, \lambda_d)$ avec :

- λ_q : perte de la Qualité ;
- λ_p : perte de la Performance ;
- λ_d : perte de la Disponibilité.

De même, le passage de l'état « marche dégradée » à l'état « hors service » sera tributaire du paramètre Λ' tel que $\Lambda' = f(\lambda'_q, \lambda'_p, \lambda'_d)$.

Le passage d'un état dégradé aval vers un état nominal amont se fait sur amélioration d'au moins une des composantes du TRS. Ainsi, la fonction « *réparation* » comprendra les paramètres : amélioration de la Qualité, amélioration de la Performance et amélioration de la Disponibilité. Pour le passage de l'état « marche dégradée » à l'état « marche », on aura le paramètre M tel que $\mathrm{M} = f(\mu_q, \mu_p, \mu_d)$ avec :

- μ_q : amélioration de la Qualité ;
- μ_p : amélioration de la Performance ;
- μ_d : amélioration de la Disponibilité.

Le passage de l'état « hors service » à l'état « marche dégradée » se fera également sur amélioration de l'un au moins de ces paramètres. Il sera tributaire de la fonction M' tel que $\mathrm{M}' = f(\mu'_q, \mu'_p, \mu'_d)$.

Le passage d'un état à l'autre n'est observé que lorsque la variation d'au moins une des composantes lambda - mu a atteint le seuil admissible de séjour dans cet état. Ces variations peuvent être considérées comme étant de « *premier degré* ». Les variations dites de « *second degré* » quant à elles permettent de maintenir la valeur de l'indicateur de performance à l'intérieur d'un état.

La composante « Disponibilité Opérationnelle » dépend de deux types de paramètres : les *pannes* et les *arrêts fonctionnels*. Ces deux états du système dépendent de deux événements différents. Le premier est aléatoire et peut survenir à tout moment. Le second est programmé et géré par un Opérateur.

Une panne sévère de même qu'une perte sévère de la Qualité ou de la Performance peuvent faire basculer le système de l'état « marche » à l'état « hors service ». IL en est de même pour une perte excessive de temps par l'opérateur dans l'état d'arrêts fonctionnels. Ce passage sera défini par un paramètre Λ_s tel que $\Lambda_s = f(\lambda_{qs}, \lambda_{ps}, \lambda_{ds})$, avec :

- λ_{qs} : perte sévère de la Qualité ;

- λ_{ps} : perte sévère de la Performance ;
- λ_{ds} : perte sévère de la Disponibilité (panne sévère).

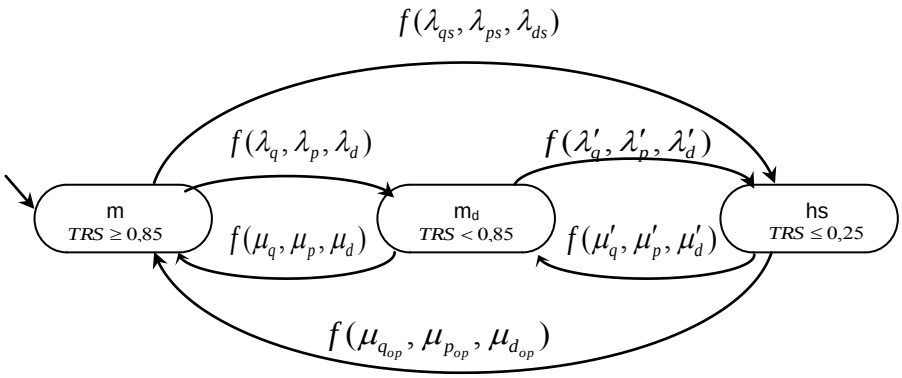


Figure 5.26 : Le modèle stochastique du TRS en fonction des dynamiques transitoires

L'automate étendu modélisant un pareil système est donné à la figure 5.27.

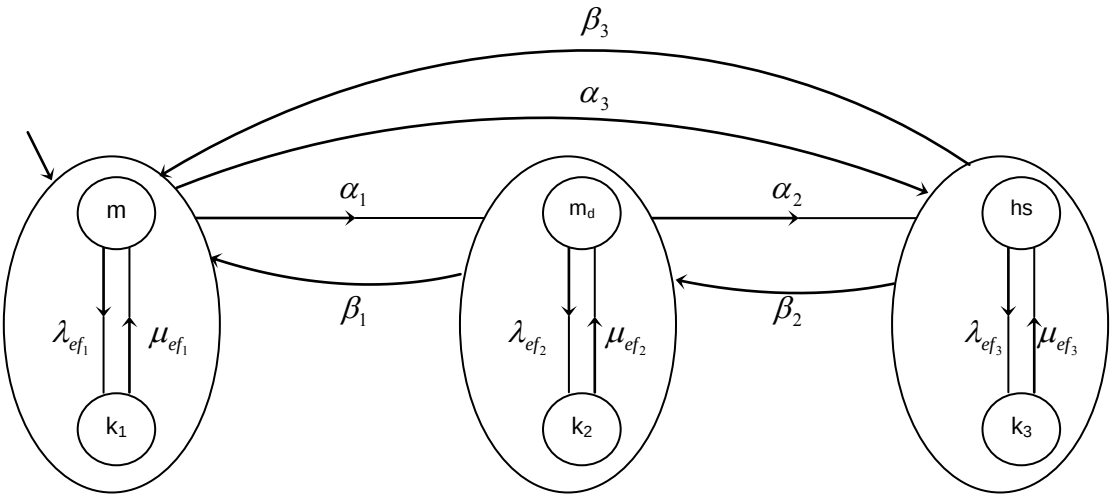


Figure 5.27: Automate étendu du modèle stochastique

Avec :

$\lambda_{ef_1} = f(\lambda_{q_1}, \lambda_{p_1}, \lambda_{d_1})$ $\quad$ $\mu_{ef_1} = f(\mu_{q_1}, \mu_{p_1}, \mu_{d_1})$

$\lambda_{ef_2} = f(\lambda_{q_2}, \lambda_{p_2}, \lambda_{d_2})$ $\quad$ $\mu_{ef_2} = f(\mu_{q_2}, \mu_{p_2}, \mu_{d_2})$

$\lambda_{ef_3} = f(\lambda_{q_3}, \lambda_{p_3}, \lambda_{d_3})$ $\quad$ $\mu_{ef_3} = f(\mu_{q_3}, \mu_{p_3}, \mu_{d_3})$

$\alpha_i - Taux\ de\ dégradation$

$\beta_i - Taux\ d'amélioration$

On obtient finalement un modèle de Cox pour modéliser la dynamique du système à l'intérieur de chaque état. Ces différents modèles sont représentés par les figures suivantes :

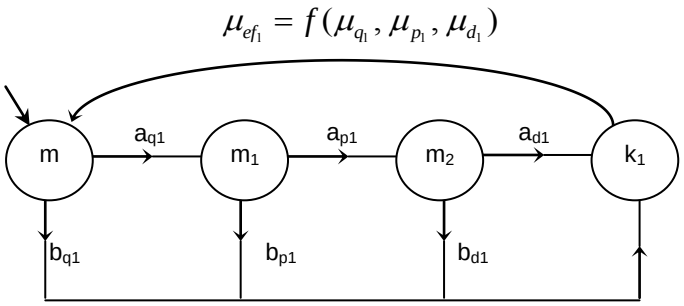


Figure 5.28: modèle de Cox de l'état « marche »

Avec :

m_1, m_2 – états fictifs

$\lambda_{q_1} = a_{q_1} + b_{q_1}$

$\lambda_{p_1} = a_{p_1} + b_{p_1}$

$\lambda_{d_1} = a_{d_1} + b_{d_1}$

a_{qi}, a_{pi}, a_{di} – faibles pertes en Qualité, Performance et Disponibilité respectivement ;
b_{qi}, b_{pi}, b_{di} – fortes pertes en Qualité, Performance et Disponibilité respectivement ;

NB : on considère que les fortes pertes ont une forte prédominance sur les faibles pertes. Ceci permet de négliger les combinaisons *faibles pertes – fortes pertes*, et par conséquent de réduire le nombre de transitions à l'intérieur de chaque état.

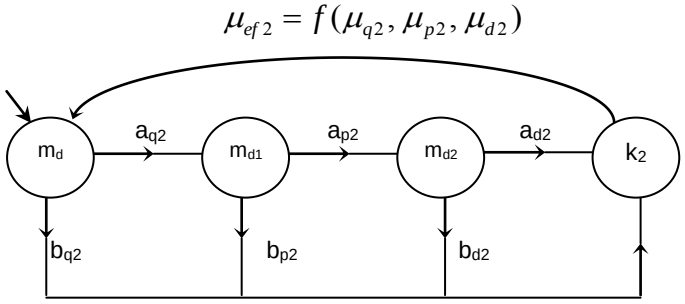


Figure 5.29 : modèle de Cox de l'état « marche dégradée »

Avec :

m_{d1}, m_{d2} – états fictifs

$\lambda_{q_2} = a_{q_2} + b_{q_2}$

$\lambda_{p_2} = a_{p_2} + b_{p_2}$

$\lambda_{d_2} = a_{d_2} + b_{d_2}$

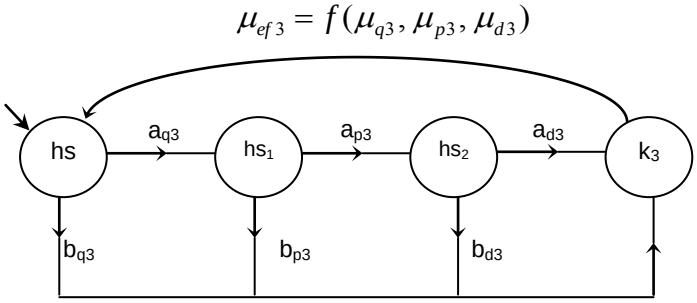


Figure 5.30 : modèle de Cox de l'état « hors service »

Avec :

hs_1, hs_2 – états fictifs

$\lambda_{q_3} = a_{q_3} + b_{q_3}$

$\lambda_{p_3} = a_{p_3} + b_{p_3}$

$\lambda_{d_3} = a_{d_3} + b_{d_3}$

5.6 CONCLUSION

La modélisation par automate d'état des composantes du TRS a permis de ressortir la complexité de la dynamique de ces composantes, lorsqu'on fixe un seuil d'appréciation de l'efficience d'un système. Les macros représentations utilisées ont permis de modéliser par des modèles Coxians la dynamique dans chaque état. Ces modèles permettent ainsi une facile interprétation en AltaRica Data-Flow de la dynamique du TRS dans chaque état. Elle permet également d'apprécier la complexité de cette dynamique dans le cas d'un système complexe (série, parallèle, assemblage et expansion). L'intégration des modèles d'automates étendus de l'Opérateur et du Réparateur dans la modélisation du système est également d'une très grande importance. Elle permet de définir les politiques de maintenance à mettre sur pied en fonction de la complexité du système. Elle permet également de ressortir la pertinence de la fiabilité humaine dans la Sureté de Fonctionnement de tout système.

Chapitre VI

MODÉLISATION ALTARICA DATA-FLOW DE L'EFFICIENCE D'UN SYSTÈME DE PRODUCTION

6.1 INTRODUCTION

La modélisation de l'efficience des systèmes de production est d'une importance indéniable. Elle permet, non seulement de faire une poursuite de la production à travers le diagramme de fiabilité du système, afin de ressortir les postes goulets, mais surtout de définir la dynamique de cette évaluation en fonction de la structure de chaque système. Si cette évaluation est assez facile pour les systèmes simples, elle devient moins évidente pour les systèmes complexes (série, parallèle, assemblage et expansion), car il faut tenir compte des facteurs d'échelle, des séquencements et des synchronisations. Pour résoudre ce problème, nous utilisons le langage de modélisation AltaRica Data-Flow, langage à la fois formel et graphique. L'usage de ce langage favorise une utilisation des modèles d'automates simples et étendus, permettant l'intégration de la Qualité, de la Performance et de la Disponibilité des systèmes. Cet usage permet, par l'intégration de la clause *sync*, de synchroniser les automates (composition d'automates). Ceci facilite la construction des modèles combinant un type de fonctionnement avec les comportements de l'Opérateur et du Réparateur. Les modèles automates temporels et stochastiques présentés aux §5.4 et 5.5 du chapitre 5 sont explicitement utilisés dans les modèles AltaRica Data-Flow. Ils permettent, non seulement de compter le temps de séjour du système dans chaque état, mais aussi de calculer en même temps le TRS, en utilisant les formules de calcul proposées au §4.4.3 du chapitre 4. Les résultats obtenus sont comparés avec les valeurs exigées par la World Class Performance, facilitant ainsi les prises de décision sur les politiques d'exploitation et de maintenance des systèmes.

Le caractère compositionnel de AltaRica Data-Flow permet de modéliser chaque composante du TRS (Qualité, Performance et Disponibilité Opérationnelle) dans un nœud appelé *node*. Chaque node permet de calculer soit le taux de Qualité, soit le taux de Performance, soit la Disponibilité Opérationnelle, à partir des différents taux de transition associés aux événements des automates étendus de chaque composante. Un node peut être décomposé en *sub-node* (sous nœud) afin de ressortir la hiérarchisation du système. Le TRS global est calculé dans le nœud principal appelé *main,* à l'intérieur duquel, grâce à la clause *extern,* les paramètres des différentes lois de probabilité associées aux événements liées aux transitions de l'automate d'état peuvent être définis. Dans le même nœud, la clause *sync* permet de contraindre deux événements simultanés appartenant à deux sous nœuds, par exemple entre les événements de l'automate étendu modélisant la Disponibilité Opérationnelle, et ceux des automates étendus de l'Opérateur et du Réparateur.

6.2 MODÉLISATION ALTARICA DATA FLOW DE L'EFFICIENCE D'UN SYSTÈME SIMPLE RÉPARABLE

La notion d'efficience a été développée au chapitre 1 §1.3.1. L'efficience d'un système est définie comme étant un attribut mesurable et observable par lequel se définit sa

performance et la qualité de ses produits. L'Efficience est mesurable et appréciable à partir d'un indicateur de performance qui donne une vision temps réel du fonctionnement du système. L'indicateur de performance permet également d'avoir une accessibilité aux paramètres de fonctionnement à chaque instant donné. En se basant sur la norme NFE 60-182, trois critères de performance permettent de définir l'efficience d'un système. Il s'agit des critères de Disponibilité, de Performance et de Qualité.

6.2.1 Modélisation AltaRica Data Flow de la Disponibilité d'un système simple réparable à deux états : « marche » et « panne »

Il s'agit ici d'un critère de Disponibilité. Le système ne pouvant se trouver que dans deux états : un état dans lequel le système fourni un service nominal : c'est l'état de marche. Le deuxième état est celui dans lequel le système fournit un service dégradé, ou ne délivre rien du tout à sa sortie suite à un événement dégradant sa performance : c'est l'état de panne. L'automate d'état d'un système simple réparable à deux états « marche » et « panne » est donné à la figure 6.1. Au départ, le système est à l'état « marche ». A la suite d'une défaillance, il passe de l'état de marche à l'état « panne ». Le retour de l'état de panne à celui de marche peut se faire à la suite d'une réparation. Les événements *panneDétectée* et *réparation* sont caractérisés par des taux de transition *lambda* et *mu* respectivement. Lambda est appelé *taux de défaillance* et mu *taux de réparation*. Dans la pratique, lambda doit être le plus petit possible, afin de minimiser l'apparition des défaillances, et mu, le plus grand possible afin d'optimiser les réparations. La somme de ces deux paramètres doit être inférieure ou égale à un. Pour des raisons de simplicité des calculs, lambda et mu doivent être pris dans la période de vie utile du système. Ils sont donc constants et sont caractérisés par des lois exponentielles.

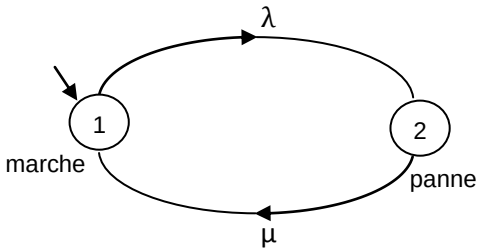


Figure 6.1 : automate d'état d'un composant simple réparable

Le modèle AltaRica Data Flow de l'automate de la figure 6.1 est donné à la figure 6.2. Les résultats de la simulation stochastique sont donnés à la figure 6.3.

```
/*
 * Un composant simple
 * -------------------
 */

domain etatComposant = {marche, panne};

node composant
state etat:etatComposant;
event panneDétectée, reparation;
init
        etat := marche;
trans
   (etat=marche) |- panneDétectée -> etat := panne,
   (etat=panne)   |- reparation ->        etat := marche;
extern
    law <event panneDétectée> = exponential(lambda);
    law <event reparation>        = exponential(mu);
    parameter lambda              = 0.00001;
    parameter mu                  = 0.1;
edon
```

```
/*
 * Le controleur
 * -------------
 */

node controleur
state etat:etatComposant;
        dateDePanne  :float;
        tempsDePanne:float;
 event panneDétectée, reparation;
 init
       etat                := marche;
       dateDePanne   := 0;
       tempsDePanne := 0;
 trans
     (etat=marche) |- panneDétectée ->  etat := panne,
     dateDePanne := %date();
     (etat=panne)   |- reparation ->          etat := marche,
     tempsDePanne := tempsDePanne +
                           (%date() - dateDePanne);
edon
```

```
 /*
 * Le système
 * ----------
 */
node main
sub Cmp:composant; Ctr:controleur;
event panneDétectée, reparation;
sync <panneDétectée,Cmp.panneDétectée,Ctr.panneDétectée>,
       <reparation,Cmp.reparation,Ctr.reparation>;
extern
   property efficienceInstantanee = <term ((%date()- Ctr.tempsDePanne)/%date())>;
   observer efficience = end_value(<term ((%date()- Ctr.tempsDePanne)/%date())>);
edon
```

Figure 6.2 : Modèle AltaRica Data Flow d'un composant simple réparable à 2 états « marche » et « panne »

Dans le modèle donné à la figure 6.2, le nœud sommet „*Système*“ comporte deux nœuds feuilles „*Composant simple*“ et „*Contrôleur*“, définissant ainsi la hiérarchie du système. Chaque nœud commence par la déclaration de son nom „*node*“ et finit par la clause „*edon*“. Le contrôleur permet de compter le temps. Il détermine la durée du séjour du composant dans l'état « panne » en calculant la différence des temps de marche et d'arrêt. Le modèle ainsi défini permet de calculer la Disponibilité du composant. Le modèle est calculé sans tenir compte de l'intervention du Réparateur. Seuls les paramètres lambda et mu permettent de passer d'un état à l'autre.

Chaque nœud feuille est manipulé par :

- les variables d'état : par exemple, le composant ne peut être que dans deux états „*marche*“ et „*panne*“.
- les événements : deux événements sont possibles pour le composant : „*panneDétectée*“ et „*réparation*“
- les transitions : Elles décrivent la dynamique du nœud, c'est-à-dire ce qui va changer à l'occurrence de l'un des événements. Elles sont caractérisées par une garde dont la valuation entraîne le changement d'état du système.

- la clause „*extern*“: elle permet de définir les principales lois de probabilité associées aux événements. Elle permet aussi de transmettre des morceaux d'informations structurées à des outils d'évaluation. Par exemple les lois de probabilité associées à des événements ne font pas partie du langage AltaRica Data-Flow. Toutefois, divers outils, y compris les simulateurs Monte-Carlo et compilateurs des Arbres de Défaillance, ont besoin de cette information.

- la clause „*sync*“: Elle permet de contraindre la simultanéité entre des événements du nœud et de ses sous nœuds direct.

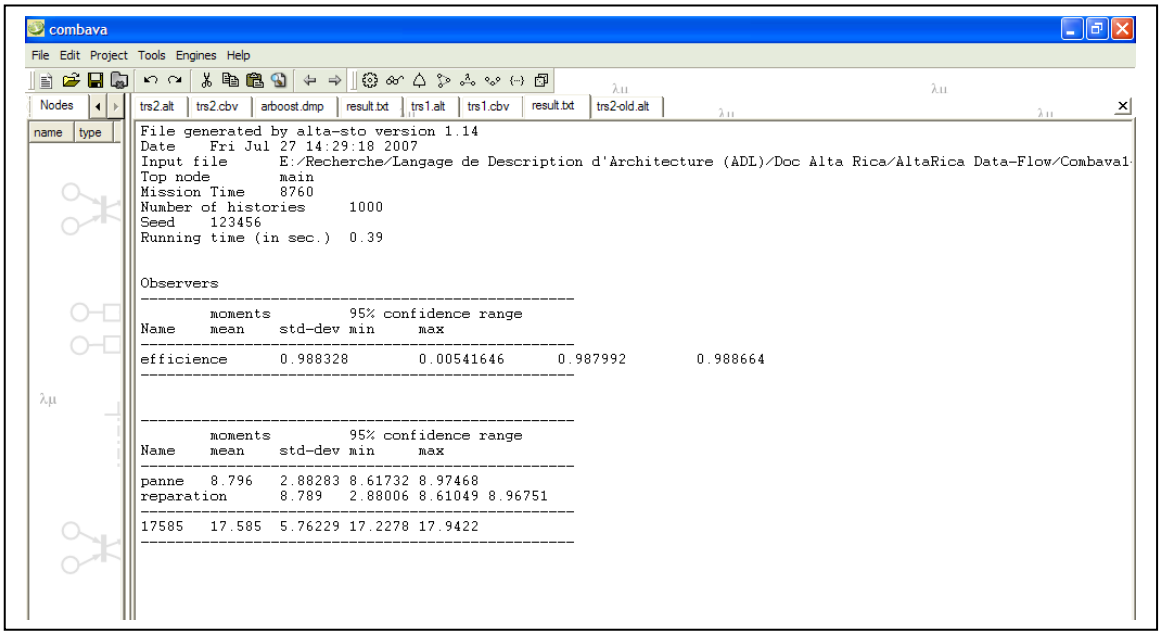


Figure 6.3 : Résultat de la simulation stochastique du modèle AltaRica d'un système simple réparable à deux états « marche » et « panne »
lambda = 0.00001, tau = 0.1

Le résultat de la figure 6.3 présente le calcul de la Disponibilité du système comportant un élément simple réparable à deux états „*marche*“ et „*panne*“. Le nœud feuille „*Contrôleur*“ permet de calculer le temps passé dans l'état „*panne*“ et génère le „*TempsDePanne*“. Le nœud sommet „*Système*“ calcule le rapport de ce „*TempsDePanne*“ sur le temps requis qui ici est égal à 8760 heures, soit un an d'activité du service. La valeur de la Disponibilité est alors définie à partir de ce rapport. Les paramètres lambda et mu ont pour valeur respective: lambda = 0.00001 et mu = 0.1. Ces valeurs sont optimales, ce qui permet d'obtenir une Disponibilité de 98,7992%.

L'influence de la variation des paramètres lambda et mu sur la valeur de la Disponibilité du composant simple réparable à deux états « marche » et « panne » est donnée au tableau 6.1.

Tableau 6.1 : Influence de la variation des paramètres lambda et mu sur la valeur de la Disponibilité

Paramètre		Disponibilité
Lambda	mu	
0.00001	0.1	0.9878
0.0001	0.01	0.8087
0.001	0.01	0.7853

6.2.2 Modélisation AltaRica Data Flow de l'efficience d'un système simple réparable à trois états « marche », « arrêt » et « panne »

Il s'agit également d'un critère de l'efficience. Deux types d'événements caractérisent les transitions de ce système : non seulement les événements probabilistes (*panneDetectée* et *réparation*), mais aussi ceux déterministes (*arrêtDétecté* et *remiseEnMarche*). La classification des différents types d'arrêts en tenant compte des temps d'état d'un moyen de production a été faite au tableau 4.1 du chapitre 4 § 4.3.2. L'automate d'état de l'efficience d'un système de production simple réparable à trois états « marche », « arrêt » et « panne » est donné à la figure 6.4. Au départ, le système est à l'état « marche ». A la suite de la détection d'une panne, il passe de l'état de marche à l'état de panne. Son retour à l'état de marche n'est possible qu'après une réparation. De même, il passera de l'état de marche à l'état « arrêt » sur détection d'un arrêt. Son retour à l'état de marche se fera après remise en état de fonctionnement par l'Opérateur.

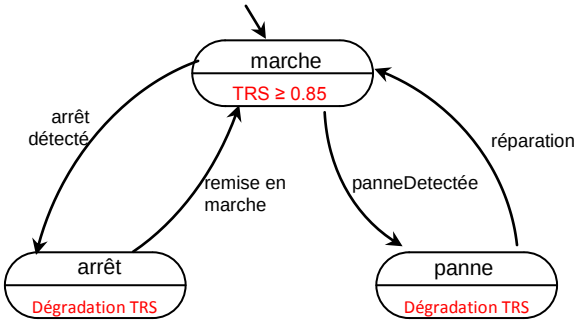


Figure 6.4 : automate d'état d'un système simple réparable à 3 états « marche », « arrêt » et « panne »

Le modèle AltaRica Data-Flow de l'efficience du système représenté à la figure 6.4 est donné à la figure 6.5. Comme dans le cas précédent, le nœud principal comprend deux sous nœuds. Le contrôleur permet de déterminer la durée du séjour passée dans les états « panne » ou « arrêt ». Les événements probabilistes sont caractérisés par une loi exponentielle tandis que les événements déterministes sont caractérisés par des fonctions de Dirac. La durée de la mission est de 8720 heures, soit un an d'activité. Le temps requis pris en exemple est égal à 420 heures [VOR 08] soit environ 3 mois d'activité. Les caractéristiques des autres paramètres sont données au tableau 6.2 suivant.

Tableau 6.2 : caractéristiques des événements liés aux transitions de la figure 6.4

Evénement	Paramètre	Fonction	Valeur
panneDétectée	lambda	exponentielle	0.001
réparation	mu	exponentielle	0.05
arrêtDétecté	pi	Dirac	7 000
remiseEnMarche	phi	Dirac	5

```
node   Unit

state  s                          :{working, failed, stopped};
       dateOfFail                  : float;
       timeOfFail                  : float;
       dateOfStop                  : float;
       timeOfStop                  : float;
       timeOfUnitStops             : float;
       plannedProductionTime       : int;

event  fail, repair, stopDetected, discountInWalk, end;
init   s                           := working;
       dateOfFail                  := 0;
       timeOfFail                  := 0;
       dateOfStop                  := 0;
       timeOfStop                  := 0;
       timeOfUnitStops             := 0;
       plannedProductionTime       := 420;
 trans
      (s = working)          |- fail              -> s := failed,
      (s = failed)           |- repair            -> s := working,
      (s = working)          |- stopDetected      -> s := stopped,
      (s = stopped)          |- discountInWalk    -> s := working;

true                         |- end ->;

extern
       law <event fail>            = exponential(lambda);
       law <event repair>          = exponential(mu);
       law <event end>             = Dirac(tau);
       law <event stopDetected>    = Dirac(pi);
       law <event discountInWalk>  = Dirac(phi);
       parameter lambda            = 0.001;
       parameter mu                = 0.05;
       parameter tau               = 8760;
       parameter pi                = 7000;
       parameter phi               = 5;

edon
```

```
node Controlor

state  s                          :{working, failed, stopped};
       dateOfFail                  : float;
       timeOfFail                  : float;
       dateOfStop                  : float;
       timeOfStop                  : float;
       timeOfUnitStops             : float;
       plannedProductionTime       : int;

event  fail, repair,  stopDetected, discountInWalk, end;
init   s                           := working;
       dateOfFail                  := 0;
       timeOfFail                  := 0;
       dateOfStop                  := 0;
       timeOfStop                  := 0;
       timeOfUnitStops             := 0;
       plannedProductionTime       := 420;
trans
      (s = working)          |- fail                  -> s := failed,
           dateOfFail := %date();
      (s = failed)           |- repair                -> s := working,
           timeOfFail := timeOfFail + (%date() - dateOfFail);
      (s = working)          |- stopDetected          -> s := stopped,
           dateOfStop := %date();
      (s = stopped)          |- discountInWalk        -> s := working,
           timeOfStop := timeOfStop + (%date() - dateOfStop),
           timeOfUnitStops := timeOfFail + timeOfStop;
true                         |- end ->;

edon
```

```
node   main
sub    U:Unit; Ctr:Controlor;
event  fail, repair, stopDetected, discountInWalk,  end;
state  s                           :{working, failed, stopped};
       timeOfUnitStops             : float;
       plannedProductionTime       : int;
init
       s                           := working;
       timeOfUnitStops             := 0;
       plannedProductionTime       := 420;

sync
       <fail:                (U.fail                and Ctr.fail)>,
       <repair:              (U.repair              and Ctr.repair)>,
       <stopDetected:        (U.stopDetected        and Ctr.stopDetected)>,
       <discountInWalk:      (U.discountInWalk      and Ctr.discountInWalk)>,
       <end:                 (U.end                 and Ctr.end)>;


extern
       property  OpAvailEfficiency = <term((plannedProductionTime - Ctr.timeOfUnitStops)/plannedProductionTime)>;
       observer  OpAvailEfficiency = end_value (<term((plannedProductionTime -
Ctr.timeOfUnitStops)/plannedProductionTime)>);
edon
```

Figure 6.5 : Modèle AltaRica Data Flow de l'efficience du système simple réparable à trois états : « marche », « arrêt » et « panne »

L'influence de la variation des paramètres *lambda*, *mu*, *pi* et *phi* sur la valeur de l'efficience du système simple réparable à trois états « marche », « panne » et « arrêt » est donnée au tableau 6.3.

Tableau 6.3 : Influence de la variation des paramètres lambda, mu, pi et phi sur la valeur de l'efficience

Paramètre				Efficience
Lambda	mu	pi	phi	
0.001	0.05	7 000	5	0.9785
0.0001	0.01	6 000	5	0.9626
0.0001	0.001	5 000	40	0.7579

Les résultats obtenus et présentés au tableau 6.3 montrent l'influence de la variation des paramètres lambda, mu, pi et phi sur l'efficience d'un système simple réparable à trois états « marche », « panne » et « arrêt ». Cette efficience diminue considérablement avec la diminution du paramètre pi, et donc avec l'augmentation du nombre d'arrêts au cours de la même mission. Elle s'améliore par contre lorsque cette période tend vers la durée de la mission. D'autre part, une augmentation du temps de remise en marche du système diminue également l'efficience du système.

Cette analyse montre bien l'intérêt d'avoir un Opérateur et un Réparateur fiables. Elle permet de mettre en évidence les politiques de maintenance mises sur pied par les chefs d'entreprises. Il va de soit qu'un Réparateur mal formé prendra plus de temps à remettre le système en état de marche à la suite d'une panne. Il en est de même pour l'Opérateur. L'incompétence de l'un et de l'autre fait apparaître un état supplémentaire dit de « marche dégradée » dans le schéma de la figure 6.4. Nous appelons état de « marche dégradée », un état du système généré par le temps de séjour supplémentaire que le système connait lorsqu'à la suite d'une défaillance, il passe plus de temps de réparation que prévu, ou encore à la suite d'un arrêt programmé, il n'est pas remis en marche au moment prévu. La valeur du TRS devient alors inférieure à 85%. Les événements qui permettent de passer à cet état sont (figure 6.6) : „'*panneProlongée''* pour le passage de l'état de panne à l'état « marche dégradée », et „'*arrêt prolongé*'''pour le passage de l'état d'arrêt à l'état « marche dégradée ».

6.2.3 Modélisation AltaRica Data Flow de l'efficience d'un système simple réparable à quatre états « marche », « arrêt », « panne » et « marche dégradée »

L'automate d'état d'un système de production simple réparable à quatre états « marche », « panne », « arrêt » et « marche dégradée » est donné à la figure 6.6. Le passage de l'état de panne à la marche dégradée survient lorsque le système a séjourné longtemps dans cet état, et que la Disponibilité Opérationnelle (Do) est passée en deçà de 90% d'après les valeurs adoptées par la World Class Performance et présentées au tableau 5.1 du chapitre 5. Il en est de même du passage de l'état d'arrêt à la marche dégradée. Le retour de l'état de marche dégradée à l'état de marche se fait soit à la fin de la réparation à la suite de l'événement « *finPanneProlongée* » ou alors à la remise en marche à la suite de l'événement « *finArretProlongé* ».

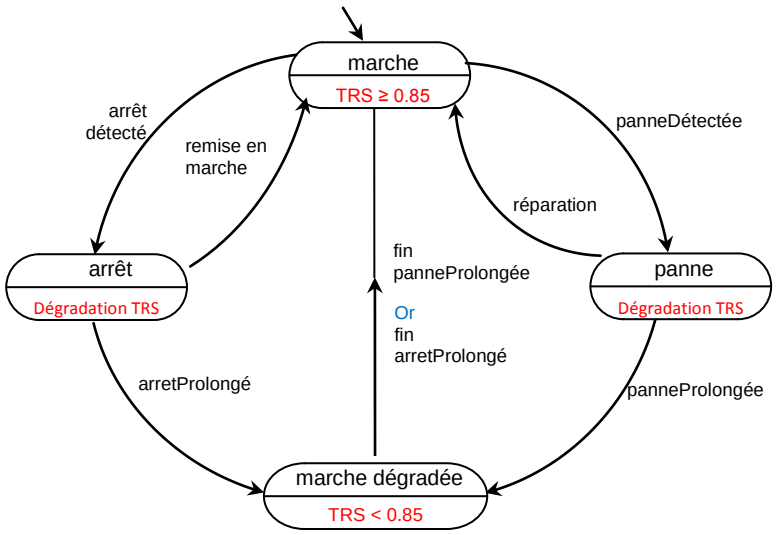


Figure 6.6 : automate d'état d'un système simple réparable à 4 états « marche », « arrêt », « panne » et « marche dégradée »

Le modèle AltaRica Data Flow de l'automate d'état de la figure 6.6 est donné à la figure 6.7. Comme dans les cas précédents, le nœud principal comprend deux nœuds feuilles. Le contrôleur permet de déterminer la durée du séjour passée dans les états « panne », « arrêt » ou « marche dégradée ». Les événements probabilistes sont caractérisés par une loi exponentielle tandis que les événements déterministes sont caractérisés par des fonctions de Dirac. La durée de la mission est de 8760 heures, soit un an d'activité. Le temps requis est égal à 420 heures, soit trois mois d'activité. Les caractéristiques des autres paramètres sont données au tableau 6.4. Les passages des états panne et arrêt à l'état marche dégradée sont tributaires des gardes, qui elles mêmes sont fonction de la durée des temps d'arrêts. Le système passe à la marche dégradée lorsque le temps d'arrêt devient égal au dixième du temps requis (T_A = TR/10). Cette valeur est déduite à partir des hypothèses de la World Class Performance fixant les valeurs seuil de l'efficience d'un système de production (tableau 5.2 chapitre 5 § 3.2).

Tableau 6.4 : caractéristiques des événements liés aux transitions de la figure 6.6

Evénement	Paramètre	Fonction	Valeur
panneDétectée	lambda	exponentielle	0.0001
réparation	mu	exponentielle	0.5
arrêtDétecté	pi	Dirac	1 000
remiseEnMarche	phi	Dirac	30
panneProlongée	sigma	Dirac	timeOfUnitFail = 0.1 x PPT
arretProlongé	sigma	Dirac	timeOfUnitStop = 0.1 x PPT
finPanneProlongée	kappa	Dirac	timeOfUnitFail < 0.1 x PPT
finArretProlongé	kappa	Dirac	timeOfUnitStop < 0.1 x PPT

```
node   Unit

state  s                        :{working, failed, stopped, Wd};
       dateOfFail               : float;
       timeOfFail               : float;
       dateOfStop               : float;
       timeOfStop               : float;
       timeOfUnitStops          : float;
       plannedProductionTime    : int;

event  fail, repair, stopDetected, discountInWalk, prolongedStop, prolongedFail, endOfProlongedFail,
       endOfProlongedStop, end;


init   s                        := working;
       dateOfFail               := 0;
       timeOfFail               := 0;
       dateOfStop               := 0;
       timeOfStop               := 0;
       timeOfUnitStops          := 0;
       plannedProductionTime    := 420;

trans
     (s = working)      |- fail                                      -> s := failed,
     (s = failed)       |- repair                                    -> s := working,
     (s = working)      |- stopDetected                              -> s := stopped,
     (s = stopped)      |- discountInWalk                            -> s := working,
     (s = failed)       |- prolongedFail                             -> s := Wd,
     (s = stopped)      |- prolongedStop                             -> s := Wd,
     (s = Wd)           |- endOfProlongedStop or endOfProlongedFail  -> s := working;


true                    |- end ->;

extern
       law <event fail>                 = exponential(lambda);
       law <event repair>               = exponential(mu);
       law <event end>                  = Dirac(tau);
       law <event stopDetected>         = Dirac(pi);
       law <event discountInWalk>       = Dirac(phi);
       law <event prolongedStop>        = Dirac(<term(plannedProductionTime/10)>);
       law <event prolongedFail>        = Dirac(<term(plannedProductionTime/10)>);
       law <event endOfProlongedFail>   = Dirac(<term((plannedProductionTime/10) - 1)>);
       law <event endOfProlongedStop>   = Dirac(<term((plannedProductionTime/10) - 1)>);
       parameter lambda                 = 0.0001;
       parameter mu                     = 0.5;
       parameter tau                    = 8760;
       parameter pi                     = 1000;
       parameter phi                    = 30;

edon
```

```
node Controlor

state  s                          :{working, failed, stopped};
       dateOfFail                  : float;
       timeOfFail                  : float;
       dateOfStop                  : float;
       timeOfStop                  : float;
       timeOfUnitStops             : float;
       plannedProductionTime       : int;

event  fail, repair,  stopDetected, discountInWalk, end;

init   s                           := working;
       dateOfFail                  := 0;
       timeOfFail                  := 0;
       dateOfStop                  := 0;
       timeOfStop                  := 0;
       timeOfUnitStops             := 0;
       plannedProductionTime       := 420;


trans
      (s = working)          |- fail                -> s := failed,
            dateOfFail := %date();
      (s = failed)           |- repair              -> s := working,
            timeOfFail := timeOfFail + (%date() - dateOfFail);
      (s = working)          |- stopDetected        -> s := stopped,
            dateOfStop := %date();
      (s = stopped)          |- discountInWalk      -> s := working,
            timeOfStop := timeOfStop + (%date() - dateOfStop),
            timeOfUnitStops := timeOfFail + timeOfStop;

true                         |- end ->;

edon


node   main
sub    U:Unit; Ctr:Controlor;
event  fail, repair, stopDetected, discountInWalk,  end;
state  s                           :{working, failed, stopped};
       timeOfUnitStops             : float;
       plannedProductionTime       : int;
init
       s                           := working;
       timeOfUnitStops             := 0;
       plannedProductionTime       := 420;

sync
       <fail:                 (U.fail                 and Ctr.fail)>,
       <repair:               (U.repair               and Ctr.repair)>,
       <stopDetected:         (U.stopDetected         and Ctr.stopDetected)>,
       <discountInWalk:       (U.discountInWalk       and Ctr.discountInWalk)>,
       <end:                  (U.end                  and Ctr.end)>;
extern
       property  OpAvailEfficiency  = <term((plannedProductionTime - Ctr.timeOfUnitStops)/plannedProductionTime)>;
       observer  OpAvailEfficiency = end_value (<term((plannedProductionTime –
       Ctr.timeOfUnitStops)/plannedProductionTime)>);

edon
```

Figure 6.7 : Modèle AltaRica Data Flow de l'automate d'état de la figure 6.6

Les résultats de la simulation stochastique du modèle AltaRica Data-Flow de la figure 6.7 sont donnés à la figure 6.8. L'influence de la variation des paramètres lambda, mu, pi et phi sur la valeur de l'efficience du système simple réparable à quatre états « marche », « panne », « arrêt » et « marche dégradée » est donnée au tableau 6.5.

Tableau 6.5 : Influence de la variation des paramètres lambda, mu, pi et phi sur la valeur de l'efficience

Paramètre				Disponibilité
Lambda	mu	pi	phi	
0.0001	0.5	1 000	30	0.9965
0.0001	0.01	6 000	5	0.9977
0.0001	0.001	5 000	40	0.9998
0.001	0.01	6 000	5	0.9999

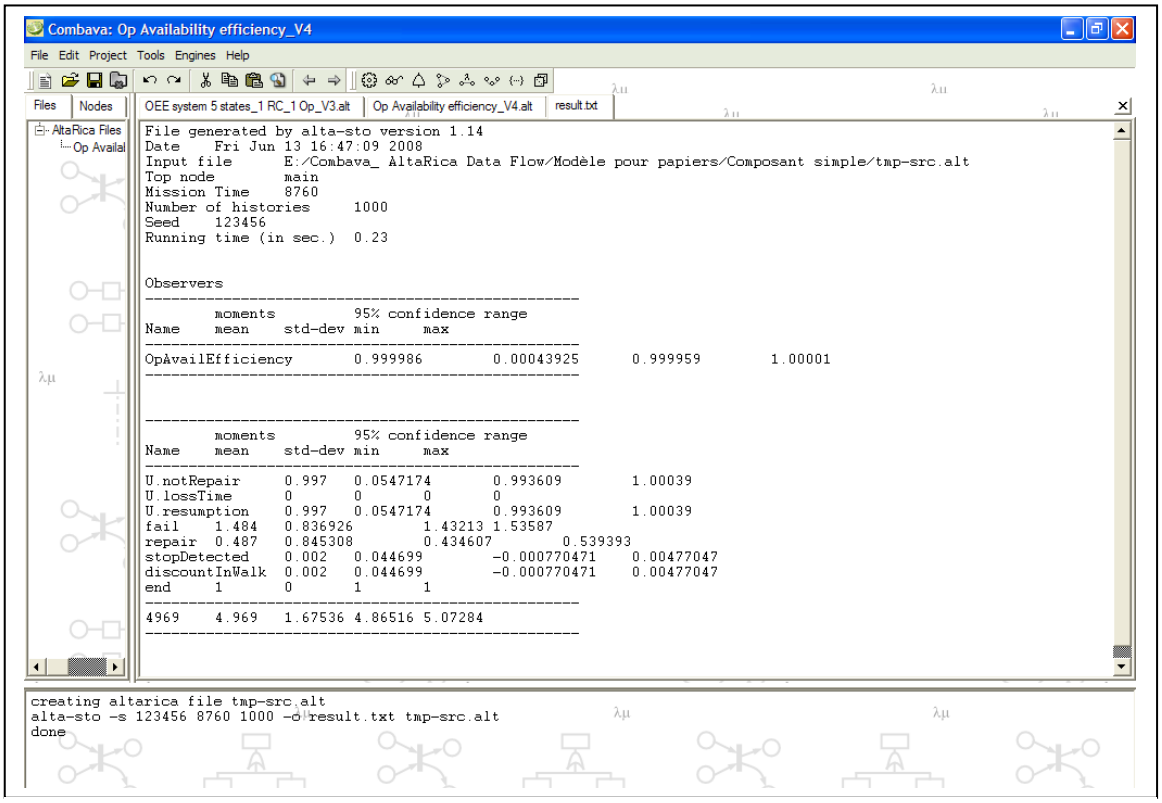


Figure 6.8: Résultat de la simulation stochastique du modèle AltaRica de l'Efficience pour un composant simple réparable
lambda = 0.001 ; *mu* = 0.01 ; *tau* = 8760 ; *pi* = 6000 ; *phi* = 5

6.2.4 Modélisation AltaRica Data-Flow de la Qualité Efficiente d'un système simple à trois états « marche », « non qualité » et « marche dégradée »

Il s'agit d'un critère de Qualité. L'automate d'état de la Qualité Efficiente (QE) d'un système simple à trois états « marche », « non Qualité » et « marche dégradée » est donné à la figure 6.9. Au départ, le système est à l'état « marche ». Le taux de Qualité est alors supérieur ou égal à 99% (tableau 5.2 du chapitre 5). Dès que le système commence à connaître des pertes en Qualité, il bascule en mode « non Qualité » sur événement « *perteQualité* ». On évalue alors le temps perdu suite à cette perte de Qualité. Ce temps va faire chuter le temps utile, diminuant également le taux de Qualité qui est le rapport du temps utile sur le temps net, c'est-à-dire le rapport du temps utile sur le temps utile plus le temps perdu pour la non Qualité (formule (21) du chapitre 4 § 4.3.2). Lorsque le taux de qualité devient inférieur à 99%, le système passe de l'état de non Qualité à celui de la marche dégradée sur événement « *perteQualitéProlongée* ». Le retour à l'état de marche se fait sur événements « *finPerteQualité* » ou « *finPerteQualitéProlongée* ».

Les événements « *perteQualité* » et « *finPerteQualité* » sont probabilistes. Ils sont caractérisés par des lois exponentielles. Les autres événements : « *perteQualitéProlongée* » et « *finPerteQualitéProlongée* » sont déterministes. Ils sont caractérisés par des fonctions de Dirac.

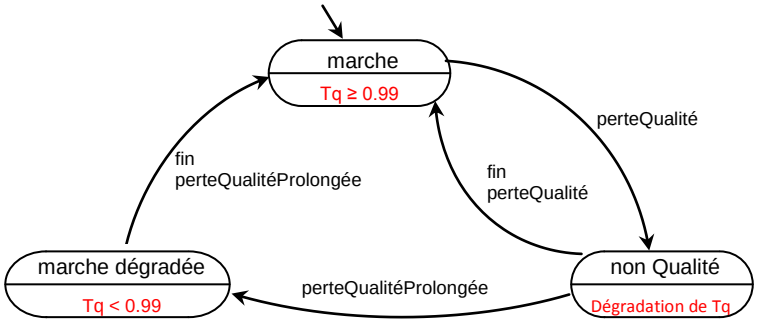


Figure 6.9 : automate d'état de la Qualité Efficiente d'un système à 3 états « marche », « non Qualité » et « marche dégradée »

Le modèle AltaRica Data-Flow de l'automate d'état de la figure 6.9 est donné à la figure 6.10. Le nœud principal comprend deux nœuds feuilles. Le contrôleur permet de déterminer la durée du séjour passé dans les états « non qualité» et « marche dégradée ». La durée de la mission est de 8760 heures, soit un an d'activité. Le temps net est égal à 321,183 heures, temps obtenu à partir de l'exemple traité dans [VOR 08]. Les caractéristiques des autres paramètres sont données au tableau 6.6. Le passage de l'état « non Qualité » à l'état « marche dégradée » est tributaire de la garde de cette transition, qui elle même est fonction de la durée du temps perdu pour non Qualité. Le système passe à la marche dégradée lorsque le temps d'arrêt pour non Qualité devient égal au centième du temps net (t_{nq} = TN/100). Cette valeur est obtenue en résolvant l'équation de la formule (21) du chapitre 4, et en tenant compte des valeurs de la World Class Performance qui exige un taux de Qualité de 99% pour un système efficient.

```
node   Unit

state  s                        :{working, Nq, Wd};
       dateOfNoneQuality        : float;
       timeOfNoneQuality        : float;
       NetOperatingTime         : float;

event  lossOfQuality, endLossOfQuality, qualityDegradation, qualityResumption, end;

init   s                        := working;
       dateOfNoneQuality        := 0;
       timeOfNoneQuality        := 0;
       NetOperatingTime         := 321.183;

trans
      (s = working)          |- lossOfQuality          -> s := Nq,
      (s = Nq)               |- endLossOfQuality      -> s := working,
      (s = Nq)               |- qualityDegradation    -> s := Wd,
      (s = Wd)               |- qualityResumption     -> s := working;

true                         |- end ->;

extern
       law <event lossOfQuality>           = exponential(rho);
       law <event endLossOfQuality>        = exponential(kappa);
       law <event end>                      = Dirac(tau);
       law <event qualityDegradation>      = Dirac(<term(NetOperatingTime/100)>);
       law <event qualityResumption>       = Dirac(<term((NetOperatingTime/100) - 1)>);
       parameter rho        = 8.91e-5;
       parameter kappa      = 3.4e-5;
       parameter tau        = 8760;

edon


node Controller

state  s                        :{working, Nq};
       dateOfNoneQuality        : float;
       timeOfNoneQuality        : float;
       NetOperatingTime         : float;

event  lossOfQuality, endLossOfQuality,  end;

init   s                        := working;
       dateOfNoneQuality        := 0;
       timeOfNoneQuality        := 0;
       NetOperatingTime         := 321.183;

trans
      (s = working)          |- lossOfQuality          -> s := Nq,
         dateOfNoneQuality := %date();
      (s = Nq)               |- endLossOfQuality      -> s := working,
         timeOfNoneQuality := timeOfNoneQuality + (%date() - dateOfNoneQuality);

true                         |- end ->;

edon
```

```
node   main
sub    U:Unit; Ctr:Controller;
event  lossOfQuality, endLossOfQuality,  end;
state  s                        :{working, Nq};
       timeOfNoneQuality        : float;
       NetOperatingTime         : float;

init   s                        := working;
       timeOfNoneQuality        := 0;
       NetOperatingTime         := 321.183;
sync
       <lossOfQuality:        (U.lossOfQuality          and Ctr.lossOfQuality)>,
       <endLossOfQuality:  (U.endLossOfQuality      and Ctr.endLossOfQuality)>,
       <end:                   (U.end                    and Ctr.end)>;

extern
       property  QualityEfficiency = <term((NetOperatingTime - Ctr.timeOfNoneQuality)/NetOperatingTime)>;
       observer  QualityEfficiency = end_value (<term((NetOperatingTime - Ctr.timeOfNoneQuality)/NetOperatingTime)>);
edon
```

Figure 6.10 : Modèle AltaRica Data-Flow de l'automate d'état de la figure 6.9

Tableau 6.6 : caractéristiques des événements liés aux transitions de la figure 6.9

Evénement	Paramètre	Fonction	Valeur
perteQualité	rho	exponentielle	$8.91{,}10^{-5}$
finPerteQualité	kappa	exponentielle	$3.4{,}10^{-5}$
perteQualitéProlongée	alpha	Dirac	timeOfUnitNoneQuality = 0.01 x NOT
finPerteQualitéProlongée	zeta	Dirac	timeOfUnitNoneQuality < 0.01 x NOT

Le résultat de la simulation stochastique du modèle AltaRica Data-Flow du modèle automate de la figure 6.9 en tenant compte des données du tableau 6.6 donne une Qualité Efficiente QE = 0.9993. Ce résultat est satisfaisant, car correspond bien aux valeurs de la World Class Performance.

6.2.5 Modélisation AltaRica Data-Flow de la Performance Efficiente d'un système simple à trois états « marche », « non Performance » et « marche dégradée »

Il s'agit d'un critère de Performance. Le système peut être soumis à des pertes de cadence, suite à un dysfonctionnement d'un de ses composants, ou alors au démarrage si l'opérateur exploitant est mal formé. L'automate d'état de la Performance Efficiente (PE) d'un système simple à trois états « marche », « non Performance » et « marche dégradée » est donné à la figure 6.11. Au départ, le système est à l'état « marche ». Le taux de Performance est alors supérieur ou égal à 95%. Dès que le système commence à connaître des ralentissements et des écarts de cadence, il bascule en mode « non Performance » sur événement « *pertePerformance* ». On évalue alors le temps perdu suite à cette perte de Performance. Ce temps va faire chuter le temps net, diminuant également le taux de Performance qui est le rapport du temps net sur le temps de fonctionnement, c'est-à-dire le rapport du temps net sur le temps net plus le temps perdu pour la non Performance (formule (22) du chapitre 4 § 4.3.2). Lorsque le taux de Performance devient inférieur à 95%, le

système passe de l'état de non Performance à celui de la marche dégradée sur événement « *pertePerformanceProlongée* ». Le taux de Performance prend alors une valeur inférieure à 95%. Le retour à l'état de marche se fait sur événements « *finPertePerformance* » ou « *finPertePerformanceProlongée* ».

Les événements « *pertePerformance* » et « *finPertePerformance*» sont probabilistes. Ils sont caractérisés par des lois exponentielles. Les autres événements : « *PertePerformenceProlongée* » et « *finpertePerformanceProlongée* » sont déterministes. Ils sont caractérisés par des fonctions de Dirac.

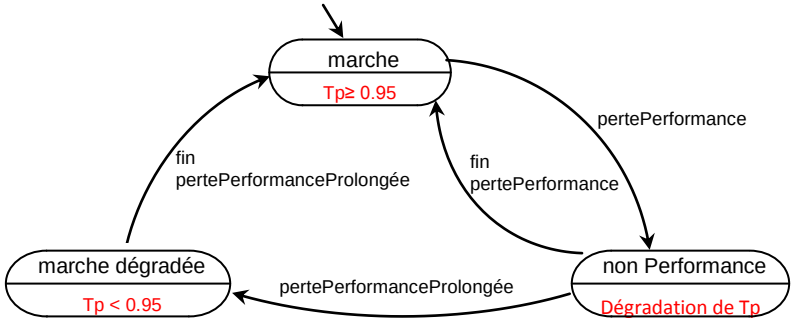


Figure 6.11 : automate d'état de la Performance Efficiente d'un système à 3 états « marche », « non Performance » et « marche dégradée »

Le modèle AltaRica Data-Flow de l'automate d'état de la figure 6.11 est donné à la figure 6.12. Le nœud principal comprend deux nœuds feuilles. Le contrôleur permet de déterminer la durée du séjour passé dans les états « non Performance» et « marche dégradée ». La durée de la mission est de 8760 heures, soit un an d'activité. Le temps de fonctionnement est égal à 373 heures, temps obtenu à partir de l'exemple traité dans [VOR 08]. Les caractéristiques des autres paramètres sont données au tableau 6.7. Le passage de l'état « non Performance » à l'état « marche dégradée » est tributaire de la garde de cette transition, qui elle même est fonction de la durée du temps perdu pour non performance. Le système passe à la marche dégradée lorsque le temps d'arrêt pour non Performance (écarts de cadence) devient égal au vingtième du temps de fonctionnement (t_{Eca} = TF/20). Cette valeur est obtenue en résolvant l'équation de la formule (22) du chapitre 4, et en tenant compte des valeurs de la World Class Performance qui exige un taux de Performance de 95% pour un système efficient.

```
node   Unit

state  s                              :{working, Np, Wd};
       dateOfNonePerformance    : float;
       timeOfNonePerformance    : float;
       OperatingTime                : float;
event  lossOfPerformance, endLossOfPerformance, performanceDegradation, performanceResumption, end;
init   s                              := working;
       dateOfNonePerformance    := 0;
       timeOfNonePerformance    := 0;
       OperatingTime                := 373;
trans
      (s = working)            |- lossOfPerformance          -> s := Np,
      (s = Np)                 |- endLossOfPerformance    -> s := working,
      (s = Np)                 |- performanceDegradation  -> s := Wd,
      (s = Wd)                 |- performanceResumption   -> s := working;
true                           |- end ->;
extern
       law <event lossOfPerformance>            = exponential(sigma);
       law <event endLossOfPerformance>      = exponential(zeta);
       law <event end>                                = Dirac(tau);
       law <event performanceDegradation>    = Dirac(<term(OperatingTime/20)>);
       law <event performanceResumption>     = Dirac(<term((OperatingTime/20) - 1)>);
       parameter sigma   = 2.11e-5;
       parameter zeta      = 3.5e-2;
       parameter tau       = 8760;
edon
```

```
node Controlor

state  s                              :{working, Np};
       dateOfNonePerformance    : float;
       timeOfNonePerformance    : float;
       OperatingTime                : float;
event  lossOfPerformance, endLossOfPerformance,  end;

init   s                               := working;
       dateOfNonePerformance    := 0;
       timeOfNonePerformance    := 0;
       OperatingTime                := 373;
trans
      (s = working)            |- lossOfPerformance          -> s := Np,
         dateOfNonePerformance := %date();
      (s = Np)                 |- endLossOfPerformance    -> s := working,
         timeOfNonePerformance := timeOfNonePerformance + (%date() - dateOfNonePerformance);

true                           |- end ->;
edon

node   main
sub    U:Unit; Ctr:Controlor;
event  lossOfPerformance, endLossOfPerformance,  end;
state  s                              :{working, Np};
       timeOfNonePerformance    : float;
       OperatingTime                : float;
init   s                              := working;
       timeOfNonePerformance    := 0;
       OperatingTime                := 373;
sync
       <lossOfPerformance:      (U.lossOfPerformance            and Ctr.lossOfPerformance)>,
       <endLossOfPerformance:   (U.endLossOfPerformance    and Ctr.endLossOfPerformance)>,
       <end:                 (U.end                                    and Ctr.end)>;
extern
       property  PerformanceEfficiency = <term((OperatingTime - Ctr.timeOfNonePerformance)/OperatingTime)>;
       observer  PerformanceEfficiency = end_value (<term((OperatingTime -
Ctr.timeOfNonePerformance)/OperatingTime)>);
edon
```

Figure 6.12 : Modèle AltaRica Data-Flow de l'automate d'état de la figure 6.10

Tableau 6.7 : caractéristiques des événements liés aux transitions de la figure 6.10

Evénement	Paramètre	Fonction	Valeur
pertePerformance	sigma	exponentielle	$2.11{,}10^{-5}$
finPertePerformance	zeta	exponentielle	$3.5{,}10^{-2}$
pertePerformanceProlongée	gamma	Dirac	timeOfUnitNonePerformance = 0.01 x OT
finPertePerformanceProlongée	epsilon	Dirac	timeOfUnitNonePerformance < 0.01 x OT

Le résultat de la simulation stochastique du modèle AltaRica Data-Flow du modèle automate de la figure 6.10 en tenant compte des données du tableau 6.7 donne une Performance Efficiente PE = 0.9534. Ce résultat est satisfaisant, car correspond bien aux valeurs de la World Class Performance.

6.2.6 Modélisation AltaRica Data-Flow du Réparateur et de l'Opérateur

L'Opérateur et le Réparateur font partie du système de production. Par conséquent, leurs comportements doivent être intégrés dans la modélisation de l'efficience du système. Les automates d'état correspondant à leurs comportements ont été donnés aux figures 5.21 et 5.22 du chapitre 5. Au départ, l'Opérateur est à l'état « *repos* ». Dès qu'une échéance d'arrêt est détectée, il passe à l'état « *occupé* » sur événement « *débutIntervention* ». A la fin de l'arrêt, le système est remis en marche. L'Opérateur passe de l'état « *occupé* » à l'état « *repos* » sur événement « *finIntervention* ». De la même façon, le Réparateur est à l'état repos au départ. Il passera à l'état occupé sur événement « *débutTravail* ». A la fin de la réparation, le système est remis en marche et le Réparateur revient à l'état repos sur événement « *finTravail* ». Ces automates sont des sous systèmes de l'automate d'état de la figure 6.6. Dans cette figure, le passage à l'état marche dégradée est tributaire des gardes de ces transitions, déclencheurs des événements « *arrêtProlongé* » et « *pannePrologée* ». Ces deux événements sont générés à cause de l'incompétence de l'Opérateur et du Réparateur. Le non respect des échéances de fin de mission par ces derniers est modélisé par une boucle à l'état « *occupé* » caractérisée par les événements « *auTravail* » et « *enIntervention* » respectivement.

Afin d'assurer une simultanéité entre les événements liés aux transitions de la figure 6.6 et ceux liés aux transitions des figures 5.21 et 5.22, une synchronisation doit être faite entre ces événements dans le modèle AltaRica Data-Flow. Par exemple, une synchronisation doit être faite entre les événements « *panneDétectée* » et « *débutTravail* », ou encore entre « *réparation* » et « *finTravail* ». De même, il faut synchroniser les événements « *arrêtDétecté* » et « *debutIntervention* » tout comme « *remiseEnMarche* » et « *finIntervention* ». La clause *sync* introduite dans le modèle AltaRica Data-Flow de la figure 6.14 permet de contraindre les événements *panneDétectée* et *débutTravail,* ainsi que les événements *arrêtDétecté* et *debuIntervention* à se produire simultanément, afin de respecter le temps imparti dans chaque état d'indisponibilité de la figure 6.4. Ceci évite au système le passage à l'état marche dégradée qui apparaît dans le modèle automate de la figure 6.6.

Les modèles AltaRica Data-Flow de l'Opérateur et du Réparateur sont donnés à la figure 6.13.

```
node    RepairCrew
state   s: {idle, busy};
event   startJob, endJob;
trans
    (s = idle)      |- startJob      -> s := busy,
    (s = busy)      |- endJob        -> s := idle;
init
    s := idle;
edon
```

```
node    Operator
state   s: {inactive, active};
event   startIntervention, endIntervention;
trans
    (s = inactive)      |- startIntervention    -> s := active,
    (s = active)        |- endIntervention      -> s := inactive ;
init
    s := inactive;
edon
```

Figure 6.13 : Modèle AltaRica Data-Flow de l'Opérateur et du Réparateur

Le modèle AltaRica Data-Flow de l'efficience d'un système simple réparable en tenant compte des comportements de l'Opérateur et du Réparateur est donné à la figure 6.14.

```
node  Unit
state s                         :{working, failed, stopped};
      dateOfFail                : float;
      timeOfFail                : float;
      dateOfStop                : float;
      timeOfStop                : float;
      timeOfUnitStop            : float;
      plannedProductionTime     : int;
event fail, repair, stopDetected, discountInWalk, end;
init  s                         := working;
      dateOfFail                := 0;
      timeOfFail                := 0;
      dateOfStop                := 0;
      timeOfStop                := 0;
      timeOfUnitStop            := 0;
      plannedProductionTime     := 420;
trans
    (s = working)       |- fail               -> s := failed,
    (s = failed)        |- repair             -> s := working,
    (s = working)       |- stopDetected       -> s := stopped,
    (s = stopped)       |- discountInWalk     -> s := working;
true                    |- end ->;
extern
      law <event fail>              = exponential(lambda);
      law <event repair>            = exponential(mu);
      law <event stopDetected>      = Dirac(pi);
      law <event discountInWalk>    = Dirac(phi);
      law <event end>               = Dirac(tau);
      parameter lambda  = 0.0001;
      parameter mu      = 0.01;
      parameter pi      = 6000;
      parameter phi     = 5;
      parameter tau     = 8760;
edon

node    RepairCrew
state   s: {inactive, active};
event   startRepair, endRepair;
trans
    (s = inactive)      |- startRepair     -> s := active,
    (s = active)        |- endRepair       -> s := inactive;
init
    s := inactive;
edon

node    Operator
state   s: {inactive, active};
event   startIntervention, endIntervention;
trans
    (s = inactive)      |- startIntervention    -> s := active,
    (s = active)        |- endIntervention      -> s := inactive;
init
    s := inactive;
edon
```

```
/*
 * The controller
 * --------------
 */

node Controller
state  s                          :{working, failed, stopped};
       dateOfFail                 : float;
       timeOfFail                 : float;
       dateOfStop                 : float;
       timeOfStop                 : float;
       timeOfUnitStop             : float;
       plannedProductionTime      : int;
event  fail, repair, stopDetected, discountInWalk, end;
init   s                          := working;
       dateOfFail                 := 0;
       timeOfFail                 := 0;
       dateOfStop                 := 0;
       timeOfStop                 := 0;
       timeOfUnitStop             := 0;
       plannedProductionTime      := 420;
trans
      (s = working)          |- fail                -> s := failed,
            dateOfFail := %date();
      (s = failed)           |- repair              -> s := working,
            timeOfFail := timeOfFail + (%date() - dateOfFail);
      (s = working)          |- stopDetected        -> s := stopped,
            dateOfStop := %date();
      (s = stopped)          |- discountInWalk      -> s := working,
            timeOfStop := timeOfStop + (%date() - dateOfStop),
            timeOfUnitStop := timeOfFail + timeOfStop;
true                         |- end ->;

edon

node   main
sub    U:Unit; Ctr:Controller; R:RepairCrew; O:Operator;
event  fail, repair, stopDetected, discountInWalk, end;
state  s                         :{working, failed, stopped};
       timeOfUnitStop             : float;
       plannedProductionTime      : int;
init
       s                         := working;
       timeOfUnitStop            := 0;
       plannedProductionTime     := 420;

sync
       <fail:                (U.fail                  and  Ctr.fail                and  R.startRepair)>,
       <repair:              (U.repair                and  Ctr.repair              and  R.endRepair)>,
       <stopDetected:        (U.stopDetected          and  Ctr.stopDetected        and  O.startIntervention)>,
       <discountInWalk:      (U.discountInWalk        and  Ctr.discountInWalk      and  O.endIntervention)>,
       <end:                 (U.end                   and  Ctr.end)>;

extern
       property efficiency = <term((plannedProductionTime - Ctr.timeOfUnitStop)/plannedProductionTime)>;
       observer efficiency = end_value (<term((plannedProductionTime - Ctr.timeOfUnitStop)/plannedProductionTime)>);
edon
```

Figure 6.14 : Modèle AltaRica Data-Flow de l'automate d'état de la figure 6.6 en tenant compte du comportement de l'Opérateur et du Réparateur

L'influence de la variation des paramètres *lambda*, *mu*, *pi* et *phi* sur la valeur de l'efficience du système simple réparable à trois états « marche », « panne » et « arrêt » est donnée au tableau 6.8.

Tableau 6.8 : Influence de la variation des paramètres lambda, mu, pi et phi sur la valeur de l'efficience en tenant compte du comportement de l'Opérateur et du Réparateur

Paramètre				**Disponibilité**	
lambda	mu	pi	phi	Sans Opérateur ni Réparateur	Avec 1 Opérateur et 1 Réparateur
0.001	0.05	7 000	5	0.9785	0.9785
0.0001	0.01	6 000	5	0.9626	0.9626
0.0001	0.001	5 000	40	0.7579	0.7579

En pratique, la compétence du Réparateur seule ne suffit pas. Il faut aussi tenir compte de sa disponibilité. Son indisponibilité lorsque survient l'événement « *panneDétectée* » met le système dans un état d'attente d'intervention, ce qui augmente le temps d'arrêt et diminue la disponibilité du système. L'automate d'état d'un pareil système est donné à la figure 6.15. Les caractéristiques liées à ses transitions sont données au tableau 6.9, et les résultats de la simulation stochastique des modèles AltaRica Data-Flow en tenant compte des comportements du Réparateur et de l'Opérateur sont représentés au tableau 6.10. Ce tableau ressort clairement l'impact de l'état d'attente du système sur la valeur de l'efficience.

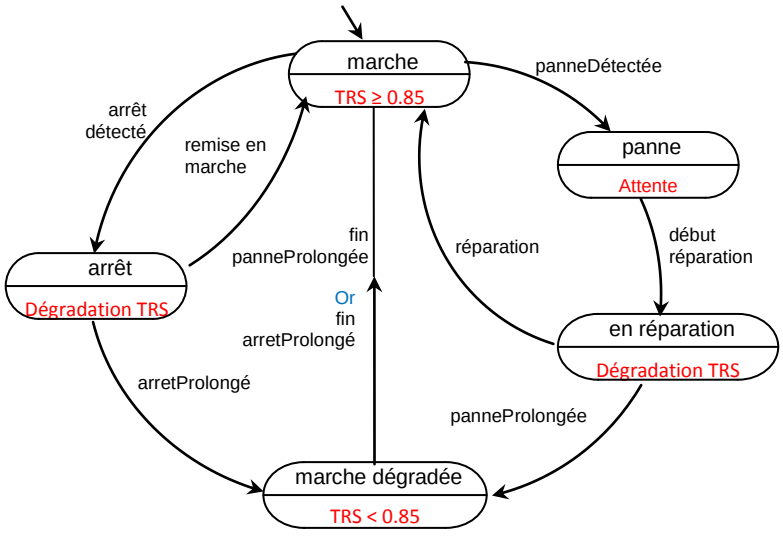


Figure 6.15: automate d'état d'un système simple réparable à 5 états « marche », « arrêt », « panne », « marche dégradée » et « en réparation » avec un état d'attente

Tableau 6.9 : caractéristiques des événements liés aux transitions de la figure 6.15

Evénement	Paramètre	Fonction	Valeur
panneDétectée	lambda	exponentielle	0.0001
réparation	mu	exponentielle	0.01
débutRéparation	pha	Dirac	15
arrêtDétecté	pi	Dirac	1 000
remiseEnMarche	phi	Dirac	30
panneProlongée	sigma	Dirac	timeOfUnitFail = 0.1 x PPT
arretProlongé	sigma	Dirac	timeOfUnitStop = 0.1 x PPT
finPanneProlongée	kappa	Dirac	timeOfUnitFail < 0.1 x PPT
finArretProlongé	kappa	Dirac	timeOfUnitStop < 0.1 x PPT

Tableau 6.10 : Influence de l'intégration des comportements de l'Opérateur et du Réparateur dans le calcul de l'efficience d'un système de production simple réparable

Modèle AltaRica Data-Flow du système	Efficience
Sans Réparateur ni Opérateur et sans état d'attente	0.8035
Sans Réparateur ni Opérateur mais avec un état d'attente	0.7747
Avec 1 Réparateur et 1 Opérateur et sans état d'attente	0.8035
Avec 1 Réparateur et 1 Opérateur et avec un état d'attente	0.7885

Les résultats du tableau 6.10 montrent bien l'influence de l'intégration des comportements de l'Opérateur et du Réparateur dans le calcul de l'efficience d'un système de production simple réparable. Nous constatons que pour un modèle AltaRica Data-Flow sans Opérateur ni Réparateur et sans état d'attente, on obtient une Efficience de 80,35%. Cette Efficience chute à 77,47% dès qu'on introduit un état d'attente dans le système. L'intervention immédiate de l'Opérateur et du Réparateur dès l'apparition des événements « *panneDétectée* » et « *arrêtDétecté* » maintient l'efficience à 80,35%. Cette valeur chute également dès qu'on introduit un état d'attente dans le système. Elle passe alors à 78,85%.

Ces résultats permettent de mettre un accent non seulement sur les politiques de maintenance mises sur pied et sur la formation du personnel, mais aussi et surtout sur les politiques sociales de l'entreprise. La motivation du personnel, l'instauration de la prime de rendement, le départ en retraite des séniors, l'assurance maladie et les visites médicales permanentes des employés, les conditions ergonomiques sont autant de paramètres qui

peuvent avoir un impact sur le comportement de l'Opérateur et du Réparateur, et donc sur l'efficience d'un système de production.

Le concept de motivation représente le construit hypothétique utilisé afin de décrire les forces internes et/ou externes produisant le déclenchement, la direction, l'intensité et la persistance du comportement au travail. L'imprécision dans les côtes de rendement est à l'origine des effets négatifs sur la motivation des employés. En effet, le sentiment d'insatisfaction et d'iniquité diminue la motivation au travail des employés. Ceci peut s'expliquer entre autre par le taux d'absentéisme élevé entrainant à chaque fois l'apparition de l'état d'attente dans le modèle de simulation de l'efficience d'un système.

La motivation du personnel est un facteur clé dans le succès d'une organisation. L'avantage concurrentiel d'une entreprise passe par l'immense potentiel offert par les hommes qui la composent. Pour bénéficier de ce potentiel d'une part les décisions d'ordre stratégiques et opérationnelles doivent tenir compte de ce que chaque personne a de mieux à offrir, c'est-à-dire le rendement individuel, et d'autre part du degré de motivation de chacun: les forces internes et/ou externes qui produisent le déclenchement, la direction, l'intensité et la persistance de son comportement. Ainsi, le rendement d'un employé qui est sa valeur et son efficacité à accomplir un travail est fonction de son degré de motivation au travail.

6.2.7 Modélisation AltaRica Data-Flow du Taux de Rendement Synthétique (TRS) d'un système simple à sept états « marche », « panne », « arrêt », « non qualité », « non performance », « marche dégradée » et « hors service »

Le TRS est un indicateur de performance complexe, car il est la composition des trois indicateurs précédemment étudiés à savoir : la Performance, la Qualité et la Disponibilité Opérationnelle. Sa valeur pour un système simple correspond au produit de ces trois indicateurs. L'automate d'état du TRS d'un système simple réparable à 7 états « *marche* », « *arrêt* », « *panne* », « *non qualité* », « *non performance* », « *marche dégradée* » et « *hors service* » est donné à la figure 5.20 du chapitre 5. Au départ, le système est en état de marche. Le passage d'un état à un autre est fonction non seulement de la garde au niveau de chaque transition, mais aussi de la valeur du TRS dont les seuils sont fixés d'après la World Class Performance (tableau 5.1 chapitre 5). Le fonctionnement d'un tel système devient stochastique puisque chaque composante peut se détériorer ou s'améliorer individuellement ou de façon combinée. Les caractéristiques liées aux différentes transitions sont données au tableau 6.11.

Le modèle AltaRica Data-Flow correspondant à l'automate d'état de la figure 5.20 du chapitre 5 est donné à la figure 6.16. Afin de profiter du caractère compositionnel qu'offre le langage AltaRica Data-Flow, nous avons fait un découpage de modèle automate en sous modèles indépendants, lesquels sont traduits en nœuds feuilles dans le langage AltaRica Data-Flow afin de caractériser la variation de l'efficience du système d'un état amont moins dégradé, vers un état aval plus dégradé et vice versa. Ce découpage s'est fait en huit nœuds feuilles de la manière suivante :

- *Unit 1* : Nœud feuille, modélise le passage de l'état de marche à l'état de panne et vice versa, en tenant compte de l'état d'attente suite à l'indisponibilité du Réparateur (figure 6.15) ;
- *Unit 2* : Nœud feuille, modélise le passage de l'état panne à l'état marche dégradée et vice versa ;

- *Unit 3* : Nœud feuille, modélise le passage de l'état de marche à l'état d'arrêt, puis à celui de marche dégradée suite à un arrêt prolongé et vice versa ;
- *Unit 4* : Nœud feuille, modélise le passage de l'état marche dégradée à l'état hors service suite à une panne prolongée excessive, ou à un arrêt prolongé excessif, puis à l'état de marche ;
- *Unit 5* : Nœud feuille, modélise le passage de l'état de marche à l'état non Qualité, puis à celui de marche dégradée et vice versa ;
- *Unit 6* : Nœud feuille, modélise le passage de l'état marche dégradée à hors service, puis à l'état marche à la suite d'une perte excessive de Qualité ;
- *Unit 7* : Nœud feuille, modélise le passage de l'état de marche à l'état non Performance, puis à celui de marche dégradée et vice versa ;
- *Unit 8* : Nœud feuille, modélise le passage de l'état marche dégradée à hors service, puis à l'état marche à la suite d'une perte excessive de Performance ;

Le calcul du TRS s'effectue dans un nœud principal appelé *main* (figure 6.16 h) à l'intérieur duquel les résultats de tous les autres sous modèles sont combinés. Il faut remarquer dans ce nœud principal l'important rôle que joue la clause *sync* afin de contraindre une simultaneité entre le comportement de l'Opérateur et la mise en arrêt et en marche du système, de même que entre celui du Réparateur et la mise en états de panne, en réparation et en marche. Les résultats de la simulation stochastique en tenant compte des données du tableau 6.11 et du comportement de l'Opérateur et du Réparateur sont donnés au tableau 6.12.

Tableau 6.11: Paramètres et différentes lois associés aux transitions de l'automate d'état des figures 5.20 et 6.15

Evénement	Paramètre	Fonction	Valeur
panneDetectée	lambda	exponentielle	0.0001
reparation	mu	exponentielle	0.01
debutReparation	pha	Dirac	15
arrêtDetecté	pi	Dirac	6 000
remiseEnMarche	phi	Dirac	5
panneProlongée	sigma	Dirac	timeOfUnitFail = 0.1 x PPT
finPanneProlongée	kappa	Dirac	timeOfUnitFail < 0.1 x PPT
arrêtProlongé	sigma	Dirac	timeOfUnitStop = 0.1 x PPT
finArrêtProlongé	kappa	Dirac	timeOfUnitFail < 0.1 x PPT
perteQualité	rho	exponentielle	$8.91,10^{-5}$
finPerteQualité	theta	exponentielle	$3.4,10^{-5}$
perteQualitéProlongée	alpha	Dirac	timeOfUnitNoneQuality = 0.01 x NOT
finPerteQualitéProlongée	zeta	Dirac	timeOfUnitNoneQuality < 0.01 x NOT
pertePerformance	beta	exponentielle	$2.11,10^{-3}$
finPertePerformance	delta	exponentielle	$3.5,10^{-3}$
pertePerformanceProlongée	gamma	Dirac	timeOfUnitNonePerformance = 0.05 x OT
finPertePerformanceProlongée	epsilon	Dirac	timeOfUnitNonePerformance < 0.05 x OT
panneExcessive	nu	Dirac	timeOfUnitFail = 0.74 x PPT
reparationOptimale	tho	exponentielle	$2.5,10^{-3}$
arrêtExcessif	xi	Dirac	timeOfUnitStop = 0.74 x PPT
finArrêtExcessif	tha	Dirac	timeOfUnitStop < 0.74 x PPT
pertePerformanceExcessive	kha	Dirac	timeOfUnitNonePerformance = 0.72 x OT
finPertePerformanceExcessive	kho	Dirac	timeOfUnitNonePerformance < 0.72 x OT
perteQualitéExcessive	khi	Dirac	timeOfUnitNoneQuality = 0.71 x NOT
finPerteQualitéExcessive	psi	Dirac	timeOfUnitNoneQuality < 0.71 x NOT

```
node   Unit1

state  s                    :{working, failed, inRepair, repaired};

       dateOfFail           : float;
       timeOfFail           : float;
       PPT                  : int;
event  fail,  repair,  startRepair, endRepair, end;
init   s                    := working;
       dateOfFail           := 0;
       timeOfFail           := 0;
       PPT                  := 420;
trans
      (s = working)         |- fail                -> s := failed,
          dateOfFail        := %date();
      (s = failed)          |- startRepair         -> s := inRepair,
      (s = inRepair)        |- repair              -> s := repaired,
      (s = repaired)        |- endRepair           -> s := working,
           timeOfFail       := timeOfFail + (%date() - dateOfFail);

true                        |- end ->;

extern
       law <event fail>                 = exponential(lambda);
       law <event repair>               = exponential(mu);
       law <event startRepair>          = Dirac(pha);
       law <event endRepair>            = Dirac(0);
       law <event end>                  = Dirac(8760);
       parameter lambda                 = 0.0001;
       parameter mu                     = 0.01;
       parameter pha                    = 15;

edon


node Unit2

state  s                    :{working, failed, Wd};
       dateOfPrologedFail   : float;
       timeOfProlongedFail  : float;
       timeOfFail           : float;
       PPT                  : int;
event   prolongedFail, endOfProlongedFail, end;
init   s                    := failed;
       dateOfPrologedFail   := 0;
       timeOfProlongedFail  := 0;
       timeOfFail           := 0;
       PPT                  := 420;
trans
      (s = failed)          |- prolongedFail          -> s := Wd,
          dateOfPrologedFail   := %date();
      (s = Wd)              |- endOfProlongedFail   -> s := working,
          timeOfProlongedFail := timeOfProlongedFail + (%date() - dateOfPrologedFail );

true                        |- end   ->;
extern

       law <event prolongedFail>        = Dirac(sigma);
       law <event endOfProlongedFail>   = Dirac(kappa);
       law <event end>                  = Dirac(8760);
       parameter sigma                  = <term(timeOfFail = 0.1*PPT)>;
       parameter kappa                  = <term(timeOfFail < 0.1*PPT)>;

edon
```

Figure 6.16 a : Modèle AltaRica Data-Flow de la Disponibilité des nœuds feuilles Unit1 et Unit2 de l'automate de la figure 6.15 en tenant compte de l'état d'attente

```
node   Unit3

state  s                          :{working, stopped, Wd};
       dateOfStop                 : float;
       timeOfStop                 : float;
       timeOfProlongedStop        : float;
       dateOfProlongedStop        : float;
       timeOfUnitStop             : float;
       PPT                        : int;

event  stopDetected,  discountInWalk,  prolongedStop, endOfProlongedStop, end;

init   s                          := working;
       dateOfStop                 := 0;
       timeOfStop                 := 0;
       timeOfProlongedStop        := 0;
       dateOfProlongedStop        := 0;
       timeOfUnitStop             := 0;
       PPT                        := 420;

trans
      (s = working)          |- stopDetected            -> s := stopped,
           dateOfStop  := %date();
      (s = stopped)          |- discountInWalk          -> s := working,
           timeOfStop  := timeOfStop + (%date() - dateOfStop);
      (s = stopped)          |- prolongedStop           -> s := Wd,
           dateOfProlongedStop := %date();
      (s = Wd)               |- endOfProlongedStop      -> s := working,
           timeOfProlongedStop := timeOfProlongedStop + (%date() - dateOfProlongedStop),
           timeOfUnitStop      := timeOfStop + timeOfProlongedStop;

true                         |- end ->;

extern
       law <event stopDetected>          = Dirac(pi);
       law <event discountInWalk>        = Dirac(phi);
       law <event prolongedStop>         = Dirac(sigma);
       law <event endOfProlongedStop>    = Dirac(kappa);
       law <event end>                    = Dirac(tau);
       parameter pi        = 6000;
       parameter phi       = 5;
       parameter sigma     = <term(timeOfUnitStop = 0.1*PPT)>;
       parameter kappa     = <term(timeOfUnitStop < 0.1*PPT)>;
       parameter tau       = 8760;

edon
```

Figure 6.16 b : Modèle AltaRica Data-Flow de la Disponibilité du nœud feuille Unit3 de l'automate de la figure 5.20 en ne tenant compte que des temps d'arrêt

```
node   Unit4

state  s                          :{working, Wd, OS};
       timeOfUnitFail              : float;
       timeOfUnitStop              : float;
       dateOfExProlFail            : float;
       timeOfExProlFail            : float;
       dateOfExProlStop            : float;
       timeOfExProlStop            : float;
       timeOfUnitExProlStop        : float;
       timeOfFail                  : float;
       timeOfProlongedFail         : float;
       PPT                         : int;

event  exessiveProlongedFail, optimalRepair, exessiveProlongedStop, endOfExProlongedStop, end;

init   s                           := Wd;
       timeOfUnitFail              := 0;
       timeOfUnitStop              := 0;
       dateOfExProlFail            := 0;
       timeOfExProlFail            := 0;
       dateOfExProlStop            := 0;
       timeOfExProlStop            := 0;
       timeOfUnitExProlStop        := 0;
       timeOfFail                  := 0;
       timeOfProlongedFail         := 0;
       PPT                         := 420;

trans
     (s = Wd)          |- exessiveProlongedFail       -> s := OS,
            dateOfExProlFail := %date();
     (s = Wd)          |- exessiveProlongedStop       -> s := OS,
            dateOfExProlStop  := %date();
     (s = OS)          |- optimalRepair               -> s := working,
            timeOfExProlFail := timeOfExProlFail + (%date() - dateOfExProlFail);
     (s = OS)          |- endOfExProlongedStop        -> s := working,
           timeOfExProlStop     := timeOfExProlStop + (%date() - dateOfExProlStop),
           timeOfUnitExProlStop := timeOfExProlFail + timeOfExProlStop,
           timeOfUnitFail       := timeOfFail + timeOfProlongedFail;

true                       |- end ->;

extern
       law <event exessiveProlongedFail>     = Dirac(nu);
       law <event exessiveProlongedStop>     = Dirac(xi);
       law <event endOfExProlongedStop>      = Dirac(khi);
       law <event optimalRepair>             = exponential(theta);
       law <event end>                       = Dirac(8760);
       parameter theta   = 2.5e-3;
       parameter nu      = <term(timeOfUnitFail = 0.74*PPT)>;
       parameter xi      = <term(timeOfUnitStop = 0.74*PPT)>;
       parameter khi     = <term(timeOfUnitStop < 0.74*PPT)>;

edon
```

Figure 6.16 c : Modèle AltaRica Data-Flow du nœud feuille Unit4 de l'automate de la figure 5.20 pour la mise hors service à partir de l'état marche dégradée

```
node   Unit5

state  s                          :{working, Nq, Wd};
       dateOfNoneQuality           : float;
       timeOfNoneQuality           : float;
       NOT                         : float;
       dateOfProlongedQLoss        : float;
       timeOfProlongedQLoss        : float;
       timeOfUnitNoneQuality       : float;

event  qualityLoss, endOfQualityLoss, prolongedQLoss, endOfProlongedQLoss, end;

init   s                          := working;
       dateOfNoneQuality          := 0;
       timeOfNoneQuality          := 0;
       dateOfProlongedQLoss       := 0;
       timeOfProlongedQLoss       := 0;
       timeOfUnitNoneQuality      := 0;
       NOT                        := 321.183;


trans
      (s = working)        |- qualityLoss                -> s := Nq,
           dateOfNoneQuality     := %date();
      (s = Nq)             |- endOfQualityLoss          -> s := working,
           timeOfNoneQuality     := timeOfNoneQuality + (%date() - dateOfNoneQuality);
      (s = Nq)             |- prolongedQLoss            -> s := Wd,
           dateOfProlongedQLoss  := %date();
      (s = Wd)             |- endOfProlongedQLoss       -> s := working,
           timeOfProlongedQLoss  := timeOfProlongedQLoss + (%date() - dateOfProlongedQLoss),
           timeOfUnitNoneQuality := timeOfNoneQuality + timeOfProlongedQLoss;

true                       |- end ->;

extern
       law <event qualityLoss>              = exponential(rho);
       law <event endOfQualityLoss>         = exponential(theta);
       law <event end>                      = Dirac(tau);
       law <event prolongedQLoss>           = Dirac(alpha);
       law <event endOfProlongedQLoss>      = Dirac(zeta);
       parameter rho       = 8.91e-5;
       parameter theta     = 3.4e-5;
       parameter alpha     = <term(timeOfUnitNoneQuality = 0.01*NOT)>;
       parameter zeta      = <term(timeOfUnitNoneQuality < 0.01*NOT)>;
       parameter tau       = 8760;

edon
```

Figure 6.16 d : Modèle AltaRica Data-Flow du nœud feuille Unit5 de l'automate de la figure 5.20 pour le calcul de la Qualité Efficiente

```
node   Unit6

state  s                          : {working, Wd, OS};
       timeOfExQLoss              : float;
       dateOfExQLoss              : float;
       timeOfUnitNoneQuality      : float;
       NOT                        : float;
event  excessiveLossOfQuality, endOfExessiveQLoss, end;
init   s                          := Wd;
       timeOfExQLoss              := 0;
       dateOfExQLoss              := 0;
       timeOfUnitNoneQuality      := 0;
       NOT                        := 321.183;
trans
      (s = Wd)             |- excessiveLossOfQuality     -> s := OS,
           dateOfExQLoss := %date();
      (s = OS)             |- endOfExessiveQLoss         -> s := working,
           timeOfExQLoss := timeOfExQLoss + (%date() - dateOfExQLoss);
true                       |- end ->;
extern
       law <event excessiveLossOfQuality>      = Dirac(khi);
       law <event endOfExessiveQLoss>          = Dirac(psi);
       law <event end>                         = Dirac(8760);
       parameter khi     = <term(timeOfUnitNoneQuality = 0.71*NOT)>;
       parameter psi     = <term(timeOfUnitNoneQuality < 0.71*NOT)>;
edon

node   Unit7

state  s                          :{working, Np, Wd};
       dateOfNonePerformance      : float;
       timeOfNonePerformance      : float;
       OT                         : float;
       dateOfProlongedPLoss       : float;
       timeOfProlongedPLoss       : float;
       timeOfUnitNonePerformance : float;
event  performanceLoss, endOfPerformanceLoss, prolongedPLoss, endOfProlongedPLoss, end;
init   s                          := working;
       dateOfNonePerformance      := 0;
       timeOfNonePerformance      := 0;
       OT                         := 373;
       dateOfProlongedPLoss       := 0;
       timeOfProlongedPLoss       := 0;
       timeOfUnitNonePerformance := 0;
trans
      (s = working)        |- performanceLoss            -> s := Np,
           dateOfNonePerformance     := %date();
      (s = Np)             |- endOfPerformanceLoss    -> s := working,
           timeOfNonePerformance     := timeOfNonePerformance + (%date() - dateOfNonePerformance);
      (s = Np)             |- prolongedPLoss             -> s := Wd,
           dateOfProlongedPLoss  := %date();
      (s = Wd)             |- endOfProlongedPLoss      -> s := working,
           timeOfProlongedPLoss  := timeOfProlongedPLoss + (%date() - dateOfProlongedPLoss),
           timeOfUnitNonePerformance := timeOfNonePerformance + timeOfProlongedPLoss;
true                       |- end ->;
extern
       law <event performanceLoss>          = exponential(beta);
       law <event endOfPerformanceLoss>  = exponential(delta);
       law <event end>                      = Dirac(tau);
       law <event prolongedPLoss>           = Dirac(gamma);
       law <event endOfProlongedPLoss>    = Dirac(epsilon);
       parameter beta                       = 2.11e-3;
       parameter delta                      = 3.5e-3;
       parameter gamma    = <term(timeOfUnitNonePerformance = 0.05*OT)>;
       parameter epsilon   = <term(timeOfUnitNonePerformance < 0.05*OT)>;
       parameter tau         = 8760;
edon
```

Figure 6.16 e : Modèle AltaRica Data-Flow des nœuds feuilles Unit6 et Unit7 de l'automate de la figure 5.20 pour la mise hors service suite à une perte excessive de Qualité et pour le calcul de la Performance Efficiente

```
node   Unit8

state  s                              : {working, Wd, OS};
       dateOfExPerformanceLoss        : float;
       timeOfExPerformanceLoss        : float;
       timeOfUnitNonePerformance      : float;
       OT                             : float;

event  excessiveLossOfPerformance, endOfExPerformanceLoss, end;

init   s                              := Wd;
       dateOfExPerformanceLoss        := 0;
       timeOfExPerformanceLoss        := 0;
       timeOfUnitNonePerformance      := 0;
       OT                             := 373;

trans
      (s = Wd)              |- excessiveLossOfPerformance      -> s := OS,
           dateOfExPerformanceLoss  := %date();
      (s = OS)              |- endOfExPerformanceLoss          -> s := working,
           timeOfExPerformanceLoss  := timeOfExPerformanceLoss + (%date() - dateOfExPerformanceLoss);

true                        |- end ->;

extern
       law <event excessiveLossOfPerformance>    = Dirac(kha);
       law <event endOfExPerformanceLoss>        = Dirac(kho);
       law <event end>                           = Dirac(8760);
       parameter kha     = <term(timeOfUnitNonePerformance = 0.72*OT)>;
       parameter kho     = <term(timeOfUnitNonePerformance < 0.72*OT)>;

edon
```

Figure 6.16 f : Modèle AltaRica Data-Flow du nœud feuille Unit8 de l'automate de la figure 5.20 pour la mise hors service suite à une perte excessive de Performance

```
node     RepairCrew
state    s: {idle, busy};

event    startJob, inJob, endJob;

trans
      (s = idle)        |- startJob        -> s := busy,
      true              |- inJob           -> s := busy,
      (s = busy)        |- endJob          -> s := idle;
init
      s := idle;
edon


node     Operator
state    s: {idle, busy};

event    startIntervention, inIntervention, endIntervention;

trans
      (s = idle)        |- startIntervention     -> s := busy,
      true              |- inIntervention        -> s := busy,
      (s = busy)        |- endIntervention       -> s := idle;
init
      s := idle;
edon
```

Figure 6.16 g : Modèle AltaRica Data-Flow de l'Opérateur et du Réparateur de l'automate de la figure 5.20

```
node   main

sub    A:Unit1; B:Unit2; C:Unit3; D:Unit4; E:Unit5; F:Unit6; G:Unit7; H:Unit8; R:RepairCrew; O:Operator;

event  startRepair, endRepair, prolongedFail, endOfProlongedFail, stopDetected,  discountInWalk,  prolongedStop,
       endOfProlongedStop, exessiveProlongedFail, optimalRepair, exessiveProlongedStop, endOfExProlongedStop;

state  s                              :{working, failed, inRepair, repaired, stopped, Nq, Np, Wd, OS};
       timeOfFail                     : float;
       timeOfProlongedFail            : float;
       timeOfUnitStop                 : float;
       timeOfUnitExProlStop           : float;
       timeOfUnitNoneQuality          : float;
       timeOfExQLoss                  : float;
       timeOfUnitNonePerformance      : float;
       timeOfExPerformanceLoss        : float;
       PPT                            : int;

init   s                              := working;
       timeOfFail                     := 0;
       timeOfProlongedFail            := 0;
       timeOfUnitStop                 := 0;
       timeOfUnitExProlStop           := 0;
       timeOfUnitNoneQuality          := 0;
       timeOfExQLoss                  := 0;
       timeOfUnitNonePerformance      := 0;
       timeOfExPerformanceLoss        := 0;
       PPT                            := 420;

sync
       <startRepair:                  (A.startRepair                 and  R.startJob)>,
       <endRepair:                    (A.endRepair                   and  R.endJob)>,
       <prolongedFail:                (B.prolongedFail               and  R.inJob)>,
       <endOfProlongedFail:           (B.endOfProlongedFail          and  R.endJob)>,
       <stopDetected:                 (C.stopDetected                and  O.startIntervention)>,
       <discountInWalk:               (C.discountInWalk              and  O.endIntervention)>,
       <prolongedStop:                (C.prolongedStop               and  O.inIntervention)>,
       <endOfProlongedStop:           (C.endOfProlongedStop          and  O.endIntervention)>,
       <exessiveProlongedFail:        (D.exessiveProlongedFail       and  R.inJob)>,
       <optimalRepair:                (D.optimalRepair               and  R.endJob)>,
       <exessiveProlongedStop:        (D.exessiveProlongedStop       and  O.inIntervention)>,
       <endOfExProlongedStop:         (D.endOfExProlongedStop        and  O.endIntervention)>;

extern
       property Efficiency = <term((PPT - (A.timeOfFail + B.timeOfProlongedFail + C.timeOfUnitStop +
D.timeOfUnitExProlStop
                     + E.timeOfUnitNoneQuality + F.timeOfExQLoss  + G.timeOfUnitNonePerformance +
H.timeOfExPerformanceLoss))/PPT)>;
       observer Efficiency = end_value (<term((PPT - (A.timeOfFail + B.timeOfProlongedFail + C.timeOfUnitStop +
D.timeOfUnitExProlStop
                     + E.timeOfUnitNoneQuality + F.timeOfExQLoss  + G.timeOfUnitNonePerformance +
H.timeOfExPerformanceLoss))/PPT)>);

edon
```

Figure 6.16 h : Modèle AltaRica Data-Flow du nœud principal de l'automate de la figure 5.20 pour le calcul du TRS du système en tenant compte des données du tableau 6.11

Tableau 6.12 : Influence de l'intégration des comportements de l'Opérateur et du Réparateur dans le calcul TRS d'un système de production simple réparable

Modèle AltaRica Data-Flow du système	TRS
Sans Réparateur ni Opérateur et sans état d'attente	0.7392
Sans Réparateur ni Opérateur mais avec un état d'attente	0.7119
Avec 1 Réparateur et 1 Opérateur et sans état d'attente	0.8199
Avec 1 Réparateur et 1 Opérateur et avec un état d'attente	0.7817

Les résultats du tableau 6.12 montrent une fois de plus l'influence de l'intégration des comportements de l'Opérateur et du Réparateur dans la modélisation de l'efficience d'un système de production. Le TRS est élevé et de l'ordre de 82% lorsque les comportements de l'Opérateur et du Réparateur sont pris en compte dans le modèle AltaRica Data-Flow. Il chute à 73,92% dans le modèle sans prise en compte de ces comportements. Nous pouvons également constater une diminution de la valeur du TRS lorsqu'un état d'attente est pris en compte dans le modèle AltaRica Data-Flow.

6.3 MODÉLISATION ALTARICA DATA-FLOW DE L'EFFICIENCE D'UN SYSTÈME SÉRIE

Dans un système comportant n composants montés en série, le TRS global du système dépend des TRS locaux de chaque composant. Il a été démontré au § 4.4.3 du chapitre 4 que le TRS global d'un système comportant n éléments en série est égal au produit des TRS locaux. D'autre part, dans un tel système, l'efficience globale est dictée par celle du composant le plus lent de la chaîne, c'est-à-dire celui qui perd le plus de temps possible. Généralement, le TRS global d'un système série est égal au TRS du dernier composant de la chaîne.

L'automate d'états d'un système série réparable à 2 composants est donné à la figure 5.11 du chapitre 5. Chaque composant ne peut générer de TRS local que s'il est en fonctionnement. Le TRS global étant le produit des TRS locaux, le système ne peut fournir de sortie que lorsque les deux composants de la chaîne sont en fonctionnement. Le cas échéant, le système ne fournit aucune sortie, et le TRS est nul. Le TRS étant une grandeur compositionnelle, l'implication de chaque composante dans son évaluation a un impact pouvant être valué positivement ou négativement. Les comportements de l'Opérateur et du Réparateur influencent fortement l'efficience d'un tel système.

6.3.1 Modélisation AltaRica Data-Flow de l'efficience d'un système comportant deux composants montés en série

L'efficience d'un système comportant deux éléments montés en série dépend de celle de chaque composant de la chaîne. Elle dépendra non seulement des caractéristiques de

chaque composant, mais aussi de la politique de maintenance mise sur pied. L'automate d'état de chaque composant est celui de la figure 6.15 du chapitre 6. Les caractéristiques des événements liés aux transitions de la figure 6.15 sont données au tableau 6.13 en tenant compte des caractéristiques de chaque composant. Tous les composants identiques auront les mêmes taux de transition, tandis que pour les composants différents, ces taux seront différents. Le modèle AlataRica Data-Flow permettant de calculer l'efficience d'un système série à deux composants est donné à la figure 6.17. La formule utilisée pour ce calcul est la formule (67) du chapitre 4. Les résultats de la simulation stochastique du modèle AltaRica Data-Flow de la figure 6.17 sont donnés au tableau 6.14.

Tableau 6.13: caractéristiques des événements liés aux transitions de la figure 6.15 pour un système réparable série à 2 composants

Evénement	Paramètre	Fonction	Valeur	
			Composant 1	Composant 2
panneDetectée	lambda	exponentielle	0.0001	0.0001
réparation	mu	exponentielle	0.5	0.5
débutRéparation	pha	Dirac	15	15
arrêtDétecté	pi	Dirac	1 000	1 000
remiseEnMarche	phi	Dirac	30	30
panneProlongée	sigma	Dirac	timeOfUnitFail = 0.1 x PPT	timeOfUnitFail = 0.1 x PPT
arretProlongé	sigma	Dirac	timeOfUnitStop = 0.1 x PPT	timeOfUnitStop = 0.1 x PPT
finPanneProlongée	kappa	Dirac	timeOfUnitFail < 0.1 x PPT	timeOfUnitFail < 0.1 x PPT
finArretProlongé	kappa	Dirac	timeOfUnitStop < 0.1 x PPT	timeOfUnitStop < 0.1 x PPT

Les politiques de maintenance mises sur pied par l'entreprise jouent un rôle important sur les résultats du calcul de l'efficience d'un système réparable comportant deux éléments montés en série.

Le premier cas envisageable est celui où le système a un seul Réparateur. Dans ce cas, si le Réparateur est disponible, il devient sollicité chaque fois qu'un des composants de la chaîne tombe en panne. On comprend tout de suite que pour un système non fiable, le Réparateur sera fréquemment sollicité, car il est probable que le système connaisse des pannes régulières. La situation devient préoccupante lorsque les deux composants tombent en panne en même temps. Et ceci est plus probable pour les composants identiques pour lesquels les taux de défaillance et de réparation sont les mêmes. Dans ce cas, le Réparateur devra intervenir pour assurer la maintenance des deux composants l'un après l'autre, ce qui prendrait évidemment assez de temps et augmenterait l'indisponibilité du système. Si en plus le Réparateur n'est pas bien formé, le résultat serait plus mauvais. En tenant compte de son indisponibilité, l'apparition d'un état d'attente tel que traité au § 6.2.6 du chapitre 6 dégrade également le résultat.

Le second cas est celui où chaque composant a son Réparateur. Dans ce cas, on constate une amélioration conséquente de l'efficience. Cependant, il faudrait que les deux

Réparateurs soient disponibles, c'est-à-dire qu'ils interviennent dès que le composant correspondant tombe en panne. L'indisponibilité de l'un ou de l'autre entraine l'apparition de l'état d'attente, ce qui augmenterait également l'indisponibilité du système. Si en plus les deux Réparateurs sont indisponibles lorsque les deux composants tombent en panne, la situation devient paralysante pour le système.

```
node   Unit

state  s                          :{working, failed, stopped, Wd};
       dateOfFail                  : float;
       timeOfFail                  : float;
       dateOfStop                  : float;
       timeOfStop                  : float;
       plannedProductionTime       : int;
       timeOfComponentStops        : float;

event  fail, repair, stopDetected, discountInWalk, prolongedStop, prolongedFail, end;

init   s                           := working;
       dateOfFail                  := 0;
       timeOfFail                  := 0;
       dateOfStop                  := 0;
       timeOfStop                  := 0;
       plannedProductionTime       := 420;
       timeOfComponentStops        := 0;

trans
      (s = working)        |- fail                -> s := failed,
           dateOfFail := %date();
      (s = failed)         |- repair              -> s := working,
           timeOfFail := timeOfFail + (%date() - dateOfFail);
      (s = working)        |- stopDetected        -> s := stopped,
          dateOfStop  := %date();
      (s = stopped)        |- discountInWalk      -> s := working,
          timeOfStop  := timeOfStop + (%date() - dateOfStop);
      (s = stopped)        |- prolongedStop       -> s := Wd,
      (s = failed)         |- prolongedFail       -> s := Wd,
          timeOfComponentStops  := timeOfFail + timeOfStop;

true                       |- end ->;

extern
       law <event fail>              = exponential(lambda);
       law <event repair>            = exponential(mu);
       law <event prolongedFail>     = Dirac(<term (plannedProductionTime/7)>);
       law <event stopDetected>      = Dirac(pi);
       law <event discountInWalk>    = Dirac(phi);
       law <event prolongedStop>     = Dirac(<term (plannedProductionTime/7)>);
       law <event end>               = Dirac(tau);

edon
```

Figure 6.17a: Modèle AltaRica Data-Flow permettant de calculer les temps d'arrêt d'un système réparable série à 2 composants Unit 1 et Unit 2

```
node   main

sub    U1:Unit; U2:Unit; R1:RepairCrew; R2:RepairCrew; O1:Operator; O2:Operator;

event  fail, repair, stopDetected, discountInWalk;

state  s                          :{working, failed, stopped, Wd};
       PPT                        : int;
       timeOfComponentStops       : float;

init
       s                          := working;
       timeOfComponentStops       := 0;
       PPT                        := 420;

sync
     <fail:              (U1.fail                  and  R1.startRepair)>,
     <fail:              (U2.fail                  and  R2.startRepair)>,
     <repair:            (U1.repair                and  R1.endRepair)>,
     <repair:            (U2.repair                and  R2.endRepair)>,
     <stopDetected:      (U1.stopDetected          and  O1.startIntervention)>,
     <stopDetected:      (U2.stopDetected          and  O2.startIntervention)>,
     <discountInWalk:    (U1.discountInWalk        and  O1.endIntervention)>,
     <discountInWalk:    (U2.discountInWalk        and  O2.endIntervention)>;

extern
       parameter U1.lambda  = 0.0001;
       parameter U2.lambda  = 0.0001;
       parameter U1.mu      = 0.02;
       parameter U2.mu      = 0.02;
       parameter U1.pi      = 6000;
       parameter U2.pi      = 6000;
       parameter U1.phi     = 5;
       parameter U2.phi     = 5;
       parameter U1.tau     = 8760;
       parameter U2.tau     = 8760;
       parameter U1.rho     = 8.91e-5;
       parameter U2.rho     = 8.91e-5;
       parameter U1.sigma   = 3.11e-6;
       parameter U2.sigma   = 3.11e-6;
       parameter U1.kappa   = 8.91e-5;
       parameter U2.kappa   = 8.91e-5;
       parameter U1.zeta    = 3.5e-3;
       parameter U2.zeta    = 3.5e-3;
       property efficiency  = <term ((PPT - max(U1.timeOfComponentStops, U2.timeOfComponentStops))/PPT)>;
       observer efficiency = end_value (<term ((PPT - max(U1.timeOfComponentStops,
                                          U2.timeOfComponentStops))/PPT)>);

edon
```

Figure 6.17b: Modèle AltaRica Data-Flow permettant de calculer l'efficience d'un système réparable série à 2 composants Unit 1 et Unit 2 en tenant compte du comportement des Opérateurs et des Réparateurs

Lorsque les deux composants sont différents, l'efficience dépend non seulement des différents taux de transition liés aux événements de chaque automate d'état, mais également de la compétence et de la disponibilité du Réparateur et de l'Opérateur. L'automate d'état de l'équipe de Réparateurs montrant la dynamique de leurs interventions dans le cas d'un système complexe réparable est donné à la figure 6.18. Au départ, les deux réparateurs sont au repos, le système fonctionne normalement, d'où l'état « *idleMen = 2* » dans le graphe. Dès que l'un des composants tombe en panne, un des Réparateur entre en fonction pendant que le deuxième continue de se reposer. Ceci est modélisé par l'état « *idleMen = 1* ». Lorsque les deux composants tombent en panne, les deux Réparateurs entrent en fonction. L'état « *idleMen = 0* » représente cette situation dans le graphe. Evidemment, en cas

d'incompétence de l'un ou l'autre des Réparateurs, le temps imparti à la réparation sera dépassé et le système basculera dans un état de réparation prolongé. Ceci est modélisé par la boucle « *inJob* » dans chacun des états où au moins un Réparateur est en activité. Le modèle AltaRica Data-Flow permettant de modéliser le comportement de chaque Réparateur est donné à la figure 6.13. Celui permettant de calculer l'efficience d'un système série réparable à deux composants en tenant compte du comportement des Opérateurs et des Réparateurs est donné à la figure 6.17b. Une fois de plus, la claus *sync* est utilisée pour contraindre la simultanéité des événements entre le comportement du système te ceux du Réparateur et de l'Opérateur.

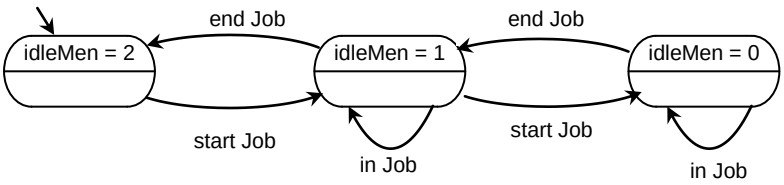


Figure 6.18: Automate d'état d'une équipe de Réparateurs

Les résultats de la simulation stochastique du modèle AltaRica Data-Flow d'un système série à deux composants en tenant compte des données du tableau 6.13 sont donnés au tableau 6.14. L'influence de l'intégration du comportement des Opérateurs et des Réparateurs ainsi que les caractéristiques des deux composants sont également pris en compte dans ces calculs.

Tableau 6.14 : Influence de l'intégration des comportements des Opérateurs et des Réparateurs dans le calcul de l'efficience d'un système réparable série à 2 composants et en tenant compte des caractéristiques de chaque composant

Modèle AltaRica Data-Flow du système	Efficience	
	Composants identiques	Composants différents
Sans Réparateur ni Opérateur	0.9745	0.9635
Avec 1 Réparateur et 1 Opérateur	0.9775	0.9714
Avec 2 Réparateurs et 1 Opérateur	0.9888	0.9853
Avec 1 Réparateur et 1 Opérateur et un état d'attente	0.9565	0.9478
Avec 2 Réparateurs et 1 Opérateur et deux états d'attente	0.9274	0.9235

Les résultats de la simulation stochastique des modèles AltaRica Data-Flow présentés au tableau 6.14 ressortent bien l'influence de l'intégration des comportements des Opérateurs et des Réparateurs dans le calcul de l'efficience d'un système réparable série à 2 composants et en tenant compte des caractéristiques de chaque composant. Lorsque les deux composants de la chaîne sont identiques, l'efficience s'améliore avec l'augmentation du nombre de Réparateurs, mais elle diminue avec l'apparition des états d'attente dans le modèle. Pour les

composants différents, on obtient exactement la même tendance, mais avec des valeurs plus faibles que dans le cas précédent. Ce qui est tout à fait normal, car l'un des composants peut présenter des caractéristiques beaucoup moins fiables : les taux de transition par exemple.

6.3.2 Modélisation AltaRica Data-Flow de la Qualité Efficiente et de la Performance Efficiente d'un système comportant deux composants montés en série

Les automates d'états d'un système à deux composants réparables montés en série et les modèles AltaRica Data-Flow correspondant sont les mêmes que ceux des figures 6.9, 6.10 et 6.11 pour chaque composant. La différentiation des composants se fait en changeant tout simplement les paramètres *rho* et *kappa* (tableau 6.6) pour la Qualité Efficiente, et *sigma* et *zêta* (tableau 6.7) pour la Performance Efficiente.

6.3.3 Modélisation AltaRica Data-Flow du TRS d'un système comportant deux composants montés en série

L'automate d'état de chaque composant de la chaîne est donné à la figure 5.20 du chapitre 5. Les paramètres et différentes lois associés aux transitions de cet automate d'états sont donnés au tableau 6.15. La dynamique transitoire est modélisée dans le nœud feuille de la figure 6.19a, les différentes lois de probabilité associées aux événements de la figure 5.20 sont données dans le modèle du nœud feuille de la figure 6.19b. Le TRS global du système est calculé dans le nœud main du modèle AltaRica Data-Flow de la figure 6.19c, en faisant le produit des TRS locaux (formule 65 du chapitre 4), eux même calculés par la formule 67 du chapitre 4. Les comportements de l'Operateur et du Réparateur sont également pris en compte dans ces modèles.

Tableau 6.15: Paramètres et différentes lois associés aux transitions de l'automate d'état de la figure 5.20 pour un système série à 2 composants

Evénement	Param	Fonction	Valeur	
			Composant 1	Composant 2
panneDetectée	lambda	exponentielle	0.0001	0.0001
reparation	mu	exponentielle	0.01	0.01
débutReparation	pha	Dirac	15	15
arrêtDetecté	pi	Dirac	6 000	6 000
remiseEnMarche	phi	Dirac	5	5
panneProlongée	sigma	Dirac	timeOfUnitFail = 0.1 x PPT	timeOfUnitFail = 0.1 x PPT
finPanneProlongée	kappa	Dirac	timeOfUnitFail < 0.1 x PPT	timeOfUnitFail < 0.1 x PPT
arrêtProlongé	sigma	Dirac	timeOfUnitStop = 0.1 x PPT	timeOfUnitStop = 0.1 x PPT
finArrêtProlongé	kappa	Dirac	timeOfUnitStop< 0.1 x PPT	timeOfUnitStop < 0.1 x PPT
perteQualité	rho	exponentielle	$8.91,10^{-5}$	$8.91,10^{-5}$
finPerteQualité	theta	exponentielle	$3.4,10^{-5}$	$3.4,10^{-5}$
perteQualitéProlongée	alpha	Dirac	timeOfUnitNoneQuality = 0.01 x NOT	timeOfUnitNoneQuality = 0.01 x NOT
finPerteQualitéProlongée	zeta	Dirac	timeOfUnitNoneQuality < 0.01 x NOT	timeOfUnitNoneQuality < 0.01 x NOT
pertePerformance	beta	exponentielle	$2.11,10^{-3}$	$2.11,10^{-3}$
finPertePerformance	delta	exponentielle	$3.5,10^{-3}$	$3.5,10^{-3}$
pertePerformanceProlongée	gamma	Dirac	timeOfUnitNonePerformance = 0.05 x OT	timeOfUnitNonePerformance = 0.05 x OT
finPertePerformanceProlongée	epsilon	Dirac	timeOfUnitNonePerformance < 0.05 x OT	timeOfUnitNonePerformance < 0.05 x OT
panneExcessive	nu	Dirac	timeOfUnitFail = 0.74 x PPT	timeOfUnitFail = 0.74 x PPT
reparationOptimale	tho	exponentielle	$2.5,10^{-3}$	$2.5,10^{-3}$
arrêtExcessif	xi	Dirac	timeOfUnitStop = 0.74 x PPT	timeOfUnitStop = 0.74 x PPT
finArrêtExcessif	tha	Dirac	timeOfUnitStop < 0.74 x PPT	timeOfUnitStop < 0.74 x PPT
pertePerformanceExcessive	kha	Dirac	timeOfUnitNonePerformance = 0.72 x OT	timeOfUnitNonePerformance = 0.72 x OT
finPertePerformanceExcessive	kho	Dirac	timeOfUnitNonePerformance < 0.72 x OT	timeOfUnitNonePerformance < 0.72 x OT
perteQualitéExcessive	khi	Dirac	timeOfUnitNoneQuality = 0.71 x NOT	timeOfUnitNoneQuality = 0.71 x NOT
finPerteQualitéExcessive	psi	Dirac	timeOfUnitNoneQuality < 0.71 x NOT	timeOfUnitNoneQuality < 0.71 x NOT

```
node   Unit
state  s                          :{working, failed, inRepair, stopped, Nq, Np, Wd, OS};
       dateOfFail                 : float;
       timeOfFail                 : float;
       dateOfStop                 : float;
       timeOfStop                 : float;
       timeOfUnitStop             : float;
       dateOfQualityLoss          : float;
       timeOfQualityLoss          : float;
       dateOfPerformanceLoss      : float;
       timeOfPerformanceLoss      : float;
       PPT                        : int;
event  fail, startRepair, repair, stopDetected, discountInWalk, QualityLoss, endOfQualityLoss, prolongedQLoss,
         endOfProlongedQLoss, performanceLoss, endOfPerformanceLoss, prolongedPLoss, endOfProlongedPLoss,
         excessiveFail, optimalRepair, excessiveStop, endOfExcessiveStop, excessivePLoss, endOfExcessivePLoss,
         excessiveQLoss, endOfExcessiveQLoss,  prolongedStop, endOfProlongedStop, prolongedFail,
         endOfProlongedFail, end;
init
       s                          := working;
       dateOfFail                 := 0;
       timeOfFail                 := 0;
       dateOfStop                 := 0;
       timeOfStop                 := 0;
       dateOfQualityLoss          := 0;
       timeOfQualityLoss          := 0;
       dateOfPerformanceLoss      := 0;
       timeOfPerformanceLoss      := 0;
       timeOfUnitStop             := 0;
       PPT                        := 420;
trans
       (s = working)        |- fail                    -> s := failed,
            dateOfFail   := %date();
       (s = failed)         |- startRepair             -> s := inRepair,
       (s = inRepair)       |- repair                  -> s := working,
            timeOfFail   := timeOfFail  + (%date() - dateOfFail);
       (s = inRepair)       |- prolongedFail           -> s := Wd,
       (s = Wd)             |- endOfProlongedFail      -> s := working,
            timeOfFail   := timeOfFail  + (%date() - dateOfFail);
       (s = Wd)             |- excessiveFail           -> s := OS,
       (s = OS)             |- optimalRepair           -> s := working,
           timeOfFail   := timeOfFail  + (%date() - dateOfFail);
       (s = working)        |- stopDetected            -> s := stopped,
            dateOfStop   := %date();
       (s = stopped)        |- discountInWalk          -> s := working,
            timeOfStop   := timeOfStop + (%date() - dateOfStop);
       (s = stopped)        |-  prolongedStop          -> s := Wd,
       (s = Wd)             |-  endOfProlongedStop     -> s := working,
            timeOfStop   := timeOfStop + (%date() - dateOfStop);
       (s = Wd)             |-  excessiveStop          -> s := OS,
       (s = OS)             |-  endOfExcessiveStop     -> s := working,
            timeOfStop   := timeOfStop + (%date() - dateOfStop);
       (s = working)        |- qualityLoss             -> s := Nq,
            dateOfQualityLoss := %date();
       (s = Nq)             |- enOfQualityLoss         -> s := working,
            timeOfQualityLoss := timeOfQualityLoss + (%date() - dateOfQualityLoss);
       (s = Nq)             |- prolongedQLoss          -> s := Wd,
       (s = Wd)             |- endOfProlongedQLoss     -> s := working,
            timeOfQualityLoss := timeOfQualityLoss + (%date() - dateOfQualityLoss);
       (s = Wd)             |- excessiveQLoss          -> s := OS,
       (s = OS)             |- endOfExcessiveQLoss     -> s := working,
            timeOfQualityLoss := timeOfQualityLoss + (%date() - dateOfQualityLoss);
       (s = working)        |- performanceLoss         -> s := Np,
            dateOfPerformanceLoss := %date();
       (s = Np)             |- enOfPerformanceLoss     -> s := working,
            timeOfPerformanceLoss := timeOfPerformanceLoss + (%date() - dateOfPerformanceLoss);
       (s = Np)             |- prolongedPLoss          -> s := Wd,
       (s = Wd)             |- endOfProlongedPLoss     -> s := working,
           timeOfPerformanceLoss := timeOfPerformanceLoss + (%date() - dateOfPerformanceLoss);
       (s = Wd)             |- excessivePLoss          -> s := OS,
       (s = OS)             |- endOfExcessivePLoss     -> s := working,
            timeOfPerformanceLoss := timeOfPerformanceLoss + (%date() - dateOfPerformanceLoss),
           timeOfUnitStop := timeOfFail + timeOfStop + timeOfQualityLoss + timeOfPerformanceLoss;
true                        |- end ->;
```

Figure 6.19a : Modèle AltaRica Data-Flow de l'automate de la figure 5.20 permettant de gérer la dynamique des transitions en tenant compte des paramètres du tableau 6.15

```
extern

    law <event fail>                    = exponential(lambda);
    law <event startRepair>             = exponential(pha);
    law <event repair>                  = exponential(mu);
    law <event prolongedFail>           = Dirac(<term(timeOfUnitFail = 0.1 x PPT)>);
    law <event enOfProlongedFail>       = Dirac(<term(timeOfUnitFail < 0.1 x PPT)>);
    law <event excessiveFail>           = Dirac(<term (timeOfUnitFail = 0.74 x PPT)>);
    law <event optimalRepair>           = exponential(tho);
    law <event stopDetected>            = Dirac(pi);
    law <event discountInWalk>          = Dirac(phi);
    law <event prolongedStop>           = Dirac(<term(timeOfUnitStop = 0.1 x PPT)>);
    law <event enOfProlongedStop>       = Dirac(<term(timeOfUnitStop < 0.1 x PPT)>);
    law <event excessiveStop>           = Dirac(<term timeOfUnitStop = 0.74 x PPT)>);
    law <event endOfExcessiveStop>      = Dirac(<term timeOfUnitStop < 0.74 x PPT)>);
    law <event qualityLoss>             = exponential(rho);
    law <event endOfQualityLoss>        = exponential(theta);
    law <event prolongedQLoss>          = Dirac(<term timeOfUnitNoneQuality = 0.01 x NOT)>);
    law <event enOfProlongedQLoss>      = Dirac(<term timeOfUnitNoneQuality < 0.01 x NOT)>);
    law <event excessiveQLoss>          = Dirac(<term timeOfUnitNoneQuality = 0.71 x NOT)>);
    law <event endOfExcessiveQLoss>     = Dirac(<term timeOfUnitNoneQuality < 0.71 x NOT)>);
    law <event performanceLoss>         = exponential(beta);
    law <event endOfPerformanceLoss>    = exponential(delta);
    law <event prolongedPLoss>          = Dirac(<term timeOfUnitNonePerformance = 0.05 x OT)>);
    law <event enOfProlongedPLoss>      = Dirac(<term timeOfUnitNonePerformance < 0.05 x OT)>);
    law <event excessivePLoss>          = Dirac(<term timeOfUnitNonePerformance = 0.72 x OT)>);
    law <event endOfExcessivePLoss>     = Dirac(<term timeOfUnitNonePerformance < 0.72 x OT)>);
    law <event end>                     = Dirac(tau)

edon


node    RepairCrew

state   s       :{inactive, active};
event   startRepair, endRepair;
init    s := inactive;
trans
    (s = inactive)    |- startRepair    -> s := active;
    (s = active)      |- endRepair      -> s := inactive;

edon


node    Operator

state   s       :{inactive, active};
event   startIntervention, endIntervention;
init    s := inactive;
trans
    (s = inactive)    |- startIntervention    -> s := active;
    (s = active)      |- endIntervention      -> s := inactive;

edon
```

Figure 6.19b : Modèle AltaRica Data-Flow permettant de définir les événements liés aux transitions de l'automate de la figure 5.20 en tenant compte des paramètres du tableau 6.15

```
node   main
sub   U1:Unit; U2:Unit; R1:RepairCrew;  R2:RepairCrew; O1:Operator; O2:Operator;

event fail, repair, stopDetected, discountInWalk;

state  s                    : {working, failed, stopped, Wd, OS};
      PPT                   : int;
      timeOfUnitStop        : float;
init
      s                           := working;
      timeOfUnitStop              := 0;
      PPT                         := 420;
sync
    <fail:                (U1.fail                  and     R1.startRepair)>,
    <fail:                (U2.fail                  and     R2.startRepair)>,
    <repair:              (U1.repair                and     R1.endRepair)>,
    <repair:              (U2.repair                and     R2.endRepair)>,
    <stopDetected:        (U1.stopDetected          and     O1.startIntervention)>,
    <stopDetected:        (U2.stopDetected          and     O2.startIntervention)>,
    <discountInWalk:      (U1.discountInWalk        and     O1.endIntervention)>,
    <discountInWalk:      (U2.discountInWalk        and     O2.endIntervention)>;

extern
     parameter U1.lambda  = 0.0001;
      parameter U1.pha       = 15;
      parameter U1.mu        = 0.01;
      parameter U1.tho       = 2.5,10^-3;
      parameter U2.lambda = 0.0001;
      parameter U2.pha       = 15;
      parameter U2.mu        = 0.01;
      parameter U2.tho       = 2.5,10^-3;
      parameter U1.pi        = 6 000;
      parameter U1.phi       = 5;
      parameter U2.pi        = 6 000;
      parameter U2.phi       = 5;
      parameter U1.rho       = 8.91e-5;
      parameter U1.theta     = 3.4e-5;
      parameter U2.rho       = 8.91e-5;
      parameter U2.theta     = 3.4e-5;
      parameter U1.beta      = 2.11e-3;
      parameter U1.delta     = 3.5e-3;
      parameter U2.beta      = 2.11e-3;
      parameter U2.delta     = 3.5e-3;
      parameter U1.tau       = 8760;
      parameter U2.tau       = 8760;

      property efficiency = <term (((PPT - U1.timeOfUnitStop)/PPT) * ((PPT - U2.timeOfUnitStop)/PPT))>;
      observer efficiency = end_value (<term(((PPT - U1.timeOfUnitStop)/PPT) * ((PPT - U2.timeOfUnitStop)/PPT))> );

edon
```

Figure 6.19c : Modèle AltaRica Data-Flow du nœud principal de l'automate de la figure 5.20 permettant de calculer le TRS global d'un système à 2 composants identiques en tenant compte des comportements des Opérateurs et des Réparateurs

Les résultats de la simulation stochastique du modèle AltaRica Data-Flow de la figure 6.19 correspondant à un système série à 2 composants identiques en tenant compte du comportement des Opérateurs et des Réparateurs sont donnés à la figure 6.20. Le système ainsi défini permet d'obtenir un TRS = 0.9256. Cette valeur devient très sensible aux variations comportementales des Opérateurs et des Réparateurs, de même qu'à celles des paramètres liés aux transitions de chaque composant. Les résultats montrant l'influence de ces variations sont donnés au tableau 6.16.

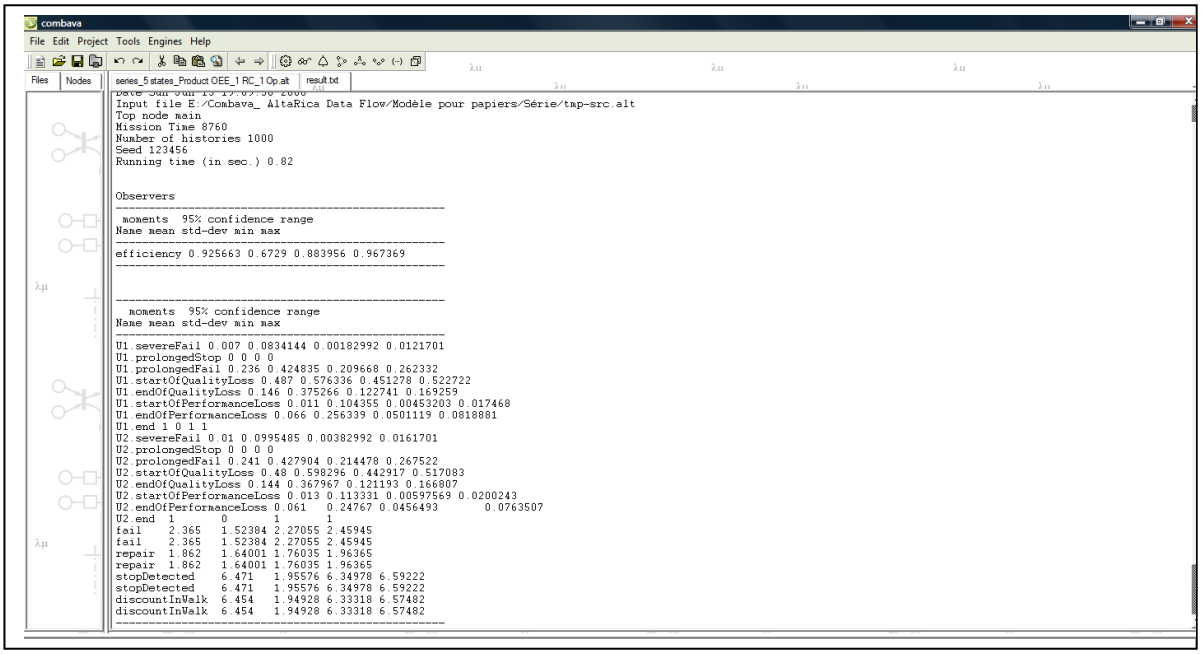


Figure 6.20: Résultats de la simulation stochastique du modèle AltaRica Data-Flow de la figure 6.18

Tableau 6.16 : Influence de l'intégration des comportements des Opérateurs et des Réparateurs dans le calcul TRS d'un système réparable série à 2 composants et en tenant compte des caractéristiques de chaque composant

Comportement du Réparateur	Modèle AltaRica Data-Flow du système	TRS
Sans retard du Réparateur (paramètre pha = 0)	Sans Réparateur ni Opérateur (composants identiques)	0.9235
	Avec 1 Réparateur et 1 Opérateur (composants identiques)	0.9314
	Avec 2 Réparateurs et 1 Opérateur (composants identiques)	0.9471
	Avec 1 Réparateur et 1 Opérateur (composants différents)	0.8874
	Avec 2 Réparateurs et 1 Opérateur (composants différents)	0.9188
Avec retard du Réparateur (paramètre pha ≠ 0)	Sans Réparateur ni Opérateur et un état d'attente (composants identiques)	0.7538
	Avec 1 Réparateur et 1 Opérateur et un état d'attente (composants identiques)	0.7625
	Avec 2 Réparateurs et 1 Opérateur et un état d'attente (composants identiques)	0.7815
	Avec 1 Réparateur et 1 Opérateur et un état d'attente (composants identiques)	0.7484
	Avec 2 Réparateurs et 1 Opérateur et deux états d'attente (composants différents)	0.7189

6.4 MODÉLISATION ALTARICA DATA-FLOW DE L'EFFICIENCE D'UN SYSTÈME PARALLÈLE

Il a été dit au § 4.4.3 du chapitre 4 que l'évaluation du TRS global d'un système comportant plusieurs éléments montés en parallèle dépend de plusieurs facteurs. Elle dépend du type de redondance (chaude ou froide), de la réparabilité des composants (système réparable ou pas), du nombre de composants que contient le système. La formule (71) permet de déterminer le TRS global d'un système parallèle dans le cas d'une redondance chaude. Pour la redondance froide, deux cas sont envisageables : d'abord si le système n'est pas réparable, ce qui signifie qu'il n'y a pas redémarrage des éléments en panne, dans ce cas, le TRS global est défini par la formule (76); d'autre part, si le système est réparable, et que tout composant qui tombe en panne peut être réparé et mis en attente pour un autre démarrage, le TRS global sera défini par la formule (77). Toutes ces formules sont obtenues à partir des temps d'état d'un moyen de production (norme NFE 60-182), et donc ne tiennent compte que de l'aspect sûreté de fonctionnement.

6.4.1 Modélisation AltaRica Data-Flow de la Disponibilité d'un système parallèle à 2 composants identiques en redondance chaude

L'automate d'état de chaque composant du système est donné à la figure 6.1. Le composant n'a que deux états : « *marche* » et « *panne* ». Les deux composants sont identiques c'est-à-dire qu'ils ont les mêmes taux de transitions. Le modèle AltaRica Data-Flow d'un tel système est donné à la figure 6.21. La formule utilisée pour le calcul de la Disponibilité est la formule 71 du chapitre 4.

```
node   Composant

state  etat                :{marche, panne};
       dateDePanne         : float;
       tempsDePanne        : float;
       tempsRequis         : int;

event  panneDetectée, reparation, fin;

init
      etat                 := marche;
      dateDePanne          := 0;
      tempsDePanne         := 0;
      tempsRequis          := 5000;

trans
      (etat = marche)       |- panneDetectée      -> etat:= panne,
         dateDePanne  := %date();
      (etat = panne)        |- reparation         -> etat:= marche,
         tempsDePanne := tempsDePanne + (%date() - dateDePanne);
true                        |- fin ->;

extern
      law <event panneDetectée>   = exponential(lambda);
      law <event reparation>      = exponential(mu);
      law <event fin>             = Dirac(5000);

edon
```

Figure 6.21a : Modèle AltaRica Data-Flow d'un composant réparable à 2 états « marche » et « panne»

```
node   main
sub   C1:Composant; C2:Composant;

state  etat                : {marche, panne};
       tempsRequis         : int;
       tempsDePanne        : float;
init
       etat                := marche;
       tempsDePanne        := 0;
       tempsRequis         := 5000;
extern
       parameter C1.lambda   = 0.0001;
       parameter C1.mu       = 0.02;
       parameter C2.lambda   = 0.0001;
       parameter C2.mu       = 0.02;
       property efficience = <term ((2*tempsRequis - (C1.tempsDePanne + C2.tempsDePanne))/tempsRequis)>;
       observer efficience = end_value (<term ((2*tempsRequis - (C1.tempsDePanne + 2.tempsDePanne))/tempsRequis)>);
edon
```

Figure 6.21b : Modèle AltaRica Data-Flow permettant de calculer la Disponibilité d'un système parallèle à 2 composants identiques en redondance chaude

6.4.2 Modélisation AltaRica Data-Flow du TRS global d'un système parallèle à 2 composants en redondance froide

En redondance froide, deux cas de fonctionnement sont possibles :

- *Redondance froide sans redémarrage des composants (système non réparable)* :

Dans ce cas, le système est non réparable. Les composants sont mis en fonctionnement l'un après l'autre, et le système s'arrête lorsque le dernier élément encore en fonctionnement tombe en panne. La dynamique d'un tel système est donnée à la figure 6.22. Au départ, le système est à l'état « *marche* ». Le composant 1 fonctionne et le second attend. A la suite de l'événement *lambda1*, le composant1 tombe en panne et le second se met en fonctionnement. Il faudrait pour cela que les commutateurs soient fiables. Le système continue donc de fonctionner. Dès que l'événement *lambda 2* se produit, les deux composants sont en panne et le système arrête de fonctionner. Chaque composant en fonctionnement est susceptible de connaitre des pertes en Qualité. Ces pertes peuvent également cesser et le système revient à l'état de fonctionnement normal. Tous les événements liés à la perte de Qualité et à sa cessation sont représentés dans le graphe de la figure 6.22 par les paramètres rho_i et $theta_i$.

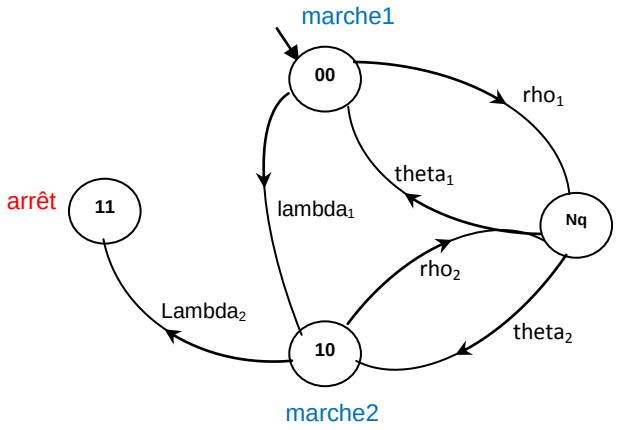


Figure 6.22 : automate d'état d'un système à 2 composants non réparables en redondance froide sans redémarrage

- *Etat de marche 1* (symbolisé par 00) : ici, le premier composant est en fonctionnement, et le second composant est en attente.
- *Etat de marche 2* (symbolisé par 10) : le premier composant est en panne, le système fonctionne toujours car le second composant a démarré dès que le premier est tombé en panne.
- *Etat arrêt* (symbolisé par 11) : le second composant en fonctionnement tombe lui aussi en panne. Puisque le système est non réparable, les deux composants sont en panne, et le système lui-même reste dans cet état.
- *Etat non qualité* (symbolisé par Nq) : Dans cet état, le composant en fonctionnement connait une perte en Qualité.

Le temps de fonctionnement du système est égal à la somme des temps de fonctionnement de tous les composants. Puisque nous sommes dans la période de vie utile de chaque composant, tous les lambdas sont constants, ce qui justifie d'ailleurs l'usage de la loi exponentielle comme principale loi de probabilité. Le temps de fonctionnement de chaque composant est égal à son temps moyen de bon fonctionnement avant la première défaillance (Mean Time To Failure (MTTF)). Pour simplifier le modèle, nous avons considéré que le système ne connaît pas de panne à la sollicitation. La formule (76) montre que le TRS global d'un système parallèle à n composants en redondance froide sans redémarrage ne dépend que de la somme des temps perdus pour non Qualité, de la somme des temps de fonctionnement de tous les composants et du temps Requis. Le modèle AltaRica Data-flow d'un pareil système est donné à la figure 6.23.

```
node   Composant

state  etat                    :{marche1, marche2, arrêt, Nq};
       dateDeNonQualité        : float;
       tempsDeNonQualité       : float;
       tempsRequis             : int;

event  panneDetectée1, panneDetectée2, perteQualité1, perteQualité2, finPerteQualité1, finPerteQualité2, fin;

init
      etat                     := marche1;
      dateDe NonQualité        := 0;
      tempsDe NonQualité       := 0;
      MTTF1                    := 0;
      MTTF2                    := 0;
      tempsRequis              := 5000;

trans
      (etat = marche1)      |- panneDetectée1       -> etat:= marche2,
           MTTF1            := 1/lambda1 ;
      (etat = marche2)      |- panneDetectée2       -> etat:= arrêt,
           MTTF2            := 1/lambda2 ;
      (etat = marche1)      |- perteQualité1        -> etat:= Nq,
         dateDeNonQualité   := %date();
      (etat = Nq)           |- finPerteQualité1     -> etat:= marche1,
         tempsDeNonQualité := tempsDeNonQualité + (%date() - dateDeNonQualité);
       (etat = marche2)     |- perteQualité2        -> etat:= Nq,
         dateDeNonQualité   := %date();
      (etat = Nq)           |- finPerteQualité2     -> etat:= marche2,
         tempsDeNonQualité := tempsDeNonQualité + (%date() - dateDeNonQualité);

true                        |- fin ->;
```

Figure 6.23a : Modèle AltaRica Data-Flow permettant de calculer la dynamique d'un système à 2 composants en redondance froide sans redémarrage

```
extern
    law <event panneDetectée1>    = exponential(lambda1);
    law <event panneDetectée2>    = exponential(lambda2);
    law <event perteQualité1>     = exponential(rho1);
    law <event perteQualité2>     = exponential(rho2);
    law <event finPerteQualité1>  = exponential(theta1);
    law <event finPerteQualité2>  = exponential(theta2);
    law <event fin>               = Dirac(5000);
    parameter lambda1   = 0.004;
    parameter lambda2   = 0.02;
    parameter rho1      = 8.91e-5;
    parameter rho2      = 8.91e-5;
    parameter theta1    = 3.4e-5;
    parameter theta2    = 3.4e-5 ;

    property efficience = <term ((MTTF1 + MTTF2) - tempsDeNonQualité)/tempsRequis)>;
    observer efficience = end_value (<term ((MTTF1 + MTTF2) - tempsDeNonQualité)/tempsRequis)>);

edon
```

Figure 6.23b : Modèle AltaRica Data-Flow permettant de calculer le TRS d'un système à 2 composants en redondance froide sans redémarrage

- *Redondance froide avec redémarrage des composants*:

Dans ce cas de figure, les deux composants du système sont réparables, c'est-à-dire qu'ils peuvent tomber en panne, être réparés puis remis en attente. Le système reste donc en marche tant qu'au moins un des composants est en fonctionnement. Le seul état d'arrêt du système est lorsque le dernier composant en attente tombe en panne au moment de la sollicitation, ou encore si le réparateur n'est pas fiable, et que la réparation du composant en panne n'est pas terminée avant que celui en fonctionnement tombe en panne lui aussi. Chaque composant en fonctionnement est susceptible de perdre en Qualité, ce qui crée également dans le modèle un état de non Qualité. L'automate d'état d'un pareil système est représenté à la figure 6.24.

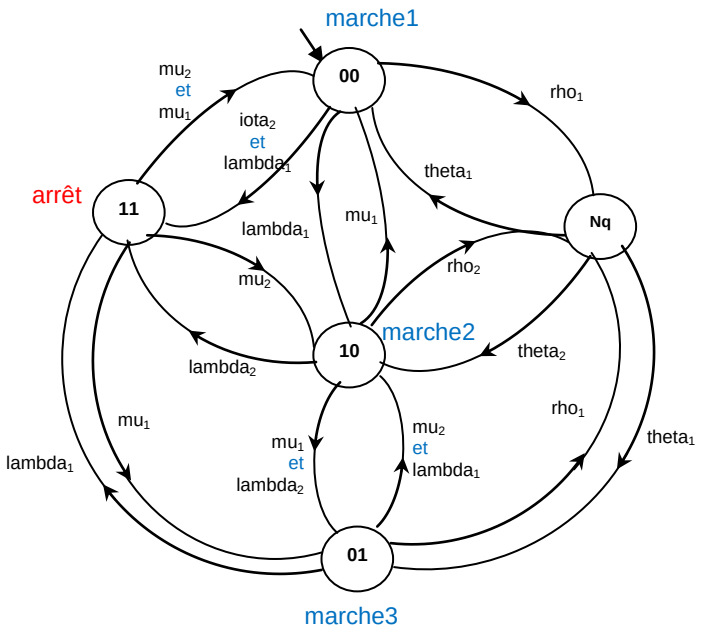


Figure 6.24 : automate d'état d'un système à 2 composants réparables en redondance froide avec redémarrage

Tous les événements liés au graphe d'état de la figure 6.24 sont probabilistes. Ils seront donc caractérisés par des fonctions exponentielles. Les paramètres et les différentes lois associés à cet automate sont donnés au tableau 6.17.

Tableau 6.17: Paramètres et différentes lois associés aux transitions de l'automate d'état de la figure 6.24

Evénement	Paramètre	Fonction
panneDetectée du premier composant	$lambda_1$	exponentielle
reparation du premier composant	mu_1	exponentielle
perteQualité du premier composant	rho_1	exponentielle
finPerteQualité du premier composant	$theta_1$	exponentielle
panneDetectée du second composant	$lambda_2$	exponentielle
reparation du second composant	mu_2	exponentielle
panne à la sollicitation du second	$iota_2$	exponentielle
perteQualité du second composant	rho_2	exponentielle
finPerteQualité du second composant	$theta_2$	exponentielle

Le modèle AltaRica Data-Flow de l'automate d'état de la figure 6.24 est donné à la figure 6.25. Au départ, le système est à l'état *marche1* (symbolisé par 00), c'est-à-dire que les deux composants sont fonctionnels, mais le second est mis en attente et le premier fonctionne. Dès que le composant 1 tombe en panne, le second est sollicité et se met en marche (état *marche2* symbolisé par 10 dans le graphe). Si le composant 2 connait une panne au moment de la sollicitation, le système bascule à l'état *arrêt* (symbolisé par 11). Le composant 2 en fonctionnement peut également tomber en panne à son tour, on passe alors à l'état *marche3* (symbolisé par 01) si la réparation du composant 2 s'est terminé avant cet événement. Dans le cas contraire, le système bascule à l'état *arrêt*. Le système est susceptible de connaître des pertes en Qualité lorsqu'il est en fonctionnement. Un état *non Qualité* (symbolisé par Nq) existe dans le graphe à cet effet.

Chaque fois que le système passe à l'état *arrêt*, le modèle compte ce temps d'arrêt. Il en est de même chaque fois que le système perd en Qualité, le modèle AltaRica Data-Flow calcule le temps perdu à cet effet. Le temps d'arrêt total du système sera égal à la somme des temps d'arrêt et de celui perdu pour non Qualité. Le TRS global du système sera égal au rapport du temps Requis moins le temps d'arrêt système sur le temps Requis (norme NFE 60-182). Cette configuration est assez fiable, et donne un TRS très élevé du fait que le système ne connait pratiquement pas d'arrêt s'il n'y a pas de panne à la sollicitation, et si le réparateur est fiable.

```
node   systeme

state  etat                    :{marche1, marche2, marche3, arret, Nq};
       dateDeNonQualité        : float;
       tempsDeNonQualité       : float;
       dateArret               : float ;
       tempsArret              : float ;
       tempsArretSysteme       : float ;
       tempsRequis             : int;

event  panneDetectée1, panneDetectée2, panneALaSollicitation2, reparation1, reparation2, perteQualité1, perteQualité2,
       finPerteQualité1, finPerteQualité2, fin;

init
     etat                      := marche1;
     dateDe NonQualité         := 0;
     tempsDe NonQualité        := 0;
     dateArret                 := 0 ;
     tempsArrêt                := 0 ;
     tempsArretSysteme         := 0 ;
     tempsRequis               := 8760;

trans
     (etat = marche1)      |- panneDetectée1 and (not panneALaSollicitation2)   -> etat:= marche2,
     (etat = marche1)      |- panneDetectée1 and panneALaSollicitation2          -> etat:= arret,
           dateArret     := % date() ;
    (etat = marche1)       |- perteQualité1                                     -> etat:= Nq,
          dateDeNonQualité  := %date();
    (etat = Nq)            |- finPerteQualité1                                  -> etat:= marche1,
          tempsDeNonQualité := tempsDeNonQualité + (%date() - dateDeNonQualité);
    (etat = marche2)       |- reparation1 and (not panne2)                      -> etat:= marche1,
    (etat = marche2)       |- panneDetectée2 and reparation1                    -> etat:= marche3,
    (etat = marche2)       |- panneDetectée2 and (not reparation1)              -> etat:= arret,
          dateArret     := % date() ;
    (etat = marche2)       |- perteQualité2                                     -> etat:= Nq,
          dateDeNonQualité  := %date();
    (etat = Nq)            |- finPerteQualité2                                  -> etat:= marche2,
          tempsDeNonQualité := tempsDeNonQualité + (%date() - dateDeNonQualité);
    (etat = marche3)       |- panneDetectée1 and reparation2                    -> etat:= marche2,
    (etat = marche3)       |- panneDetectée1 and (not reparation2)              -> etat:= arret,
          dateArret     := % date() ;
    (etat = marche3)       |- perteQualité1                                     -> etat:= Nq,
          dateDeNonQualité  := %date();
    (etat = Nq)            |- finPerteQualité1                                  -> etat:= marche3,
          tempsDeNonQualité := tempsDeNonQualité + (%date() - dateDeNonQualité);
    (etat = arret)         |- reparation1 and reparation2                       -> etat:= marche1,
          tempsArret := tempsArret + (%date() - dateArret);
    (etat = arret)         |- reparation2 and panne1                            -> etat:= marche2,
          tempsArret := tempsArret + (%date() - dateArret);
    (etat = arret)         |- reparation1 and panne2                            -> etat:= marche3,
          tempsArret := tempsArret + (%date() - dateArret);
          tempsArretSysteme := tempsArret + tempsDeNonQualité ;

true                       |- fin ->;


node    RepairCrew

state   s       :{inactive, active};
event   startRepair, endRepair;
init    s := inactive;
trans
    (s = inactive)    |- startRepair    ->  s :=  active;
    (s = active)      |- endRepair     ->  s :=  inactive;

edon
```

Figure 6.25a : Modèle AltaRica Data-Flow permettant de calculer la dynamique d'un système redondant à 2 composants avec redémarrage

```
sync
    <panneDetcetée:  (panneDetectée1              and    RepairCrew.startRepair)>,
    <panneDetectée:  (panneDetectée2              and    RepairCrew.startRepair)>,
    <reparation:     (reparation1                 and    RepairCrew.endRepair)>,
    <reparation:     (reparation2                 and    RepairCrew.endRepair)>,

extern
     law <event panneDetectée1>           = exponential(lambda1);
     law <event panne2Detectée>           = exponential(lambda2);
     law <event panneALaSollicitation2>   = exponential(iota2);
     law <event reparation1>              = exponential(mu1);
     law <event reparation2>              = exponential(mu2);
     law <event perteQualité1>            = exponential(rho1);
     law <event perteQualité2>            = exponential(rho2);
     law <event perteQualité3>            = exponential(rho3);
     law <event finPerteQualité1>         = exponential(theta1);
     law <event finPerteQualité2>         = exponential(theta2);
     law <event finPerteQualité3>         = exponential(theta3);
     law <event fin>                      = Dirac(8760);
     parameter lambda1   = 0.0001;
     parameter lambda2   = 0.002;
     parameter mu1       = 0.01;
     parameter mu2       = 0.03;
     parameter iota2     = 0.00001;
     parameter rho1      = 8.91e-5;
     parameter rho2      = 8.91e-5;
     parameter theta1    = 3.4e-5;
     parameter theta2    = 3.4e-5;


     property efficience = <term ((tempsRequis - tempsArretSysteme)/tempsRequis)>;
     observer efficience = end_value (<term ((tempsRequis - tempsArretSysteme)/tempsRequis)>);

edon
```

Figure 6.25b : Modèle AltaRica Data-Flow permettant de calculer le TRS d'un système redondant à 2 composants avec redémarrage

6.5 CONCLUSION

Nous avons démontré dans ce chapitre que le langage de modélisation AltaRica Data-Flow est à la fois compositionnel, graphique et formel. Le formalisme AltaRica Data-Flow nous a permis de nous passer des difficultés de modélisation des systèmes à grand espace d'états, en utilisant les outils classiques tels que les réseaux de Petri et le graphe de Markov. Ces outils dont l'usage présente des limites à cause du classique problème d'explosion combinatoire. Cette vue compositionnelle permet plus aisément de décrire la dynamique de la variation de l'efficience, tant dans les systèmes simples (à un seul composant), que dans les systèmes complexes (série et parallèle). L'apport indéniable dans l'utilisation de ce langage est l'intégration du comportement de l'Opérateur et du Réparateur dans cette modélisation, et la gestion des flux notamment avec la clause « *assert* ». L'utilisation de la clause « *sync* » pour établir un synchronisme entre les événements simultanés est également d'un grand intérêt. Elle contraint une assertion simultanée de deux événements. Les modèles développés dans le cas des systèmes série a permis de vérifier l'intérêt de la fiabilité humaine, mais aussi de mettre en évidence les différentes politiques de maintenance. Les systèmes parallèles avec et sans redémarrage permettent également de mettre en évidence les problèmes de conception.

CONCLUSION GÉNÉRALE

Le but de notre étude était de faire une modélisation de la propagation des fautes dans les systèmes de production en vue d'évaluer leur efficience. Nous avons pour ce faire utilisé un indicateur de performance de type TRS. Tous les travaux jusque là menés dans l'étude du TRS le calculaient de façon globale, c'est-à-dire sans tenir compte de la structure et de la configuration du système étudié. La majorité de ces travaux se limitaient à la mesure du TRS, et nul ne faisait allusion à sa modélisation, encore moins à l'aspect local de cet indicateur, c'est-à-dire de la contribution individuelle de chaque composant du système à son efficience.

Beaucoup de travaux existent cependant pour l'évaluation de performance des systèmes. On peut citer les outils classiques tels que les réseaux de Petri, les graphes de Markov, le Arbres de Défaillance etc. Mais toutes ces méthodes deviennent peu efficaces dès que l'espace d'état devient considérable, à cause du classique problème d'explosion combinatoire. Les méthodes de résolution tels que les méthodes analytiques, les méthodes numériques présentent également des limites à cause du même problème. Dès lors, seule la simulation s'avère efficace. Elle présente l'avantage d'être insensible à l'espace d'état. Elle permet d'imiter le comportement d'un système en traitant des modèles généralement plus proches de la réalité que les méthodes formelles, et elle sert souvent à valider les modèles de ces derniers. Elle mesure le comportement d'un modèle du système en simulant son exécution, par exemple par une génération aléatoire d'événements répartis de manière discrète dans le temps. Elle possède un domaine d'application quasi-illimité mais reste économiquement très coûteuse (temps humain et machine). Mais pour faciliter la mise en place d'une simulation, il sera plus judicieux de construire un modèle formel pour s'assurer du bon fonctionnement du modèle avant sa mise en œuvre dans le simulateur.

Le modèle est une représentation de la réalité dans un formalisme. Il est développé pour répondre à des questions déterminées et comporte certaines limitations, c'est une abstraction du système réel. Ce *système réel* n'existe pas forcément lors du processus de modélisation, il se peut que ce soit un système que nous cherchons à développer. Pour cela nous désirons effectuer une étude préliminaire de ses performances. Plusieurs techniques de modélisation formelles ont été développées ces dernières années. Toutes sont basées sur le même principe qui consiste à définir :

– les états du système.
– les transitions entre les états.
– les délais des transitions.

Les états sont représentés graphiquement par des nœuds et les transitions par des flèches étiquetées reliant les nœuds entre eux. Les transitions sont étiquetées par des événements dont l'occurrence engendre une valuation de la transition et un changement d'état du système, après validation de sa garde. Le graphe résultant est connu sous le nom de système de transitions représentant le comportement du système. En supposant que le système étudié a un nombre fini d'états, le changement de l'état du système est engendré par l'exécution d'une action qui déclenche une transition. Mais, il est difficile de réaliser une modélisation manuelle de l'ensemble de toutes les transitions possibles entre les différents états d'un système large et complexe, car souvent le système de transitions sous-jacent d'un système réel est de l'ordre de quelques milliers voire même centaines de milliers d'états. Plusieurs formalismes de haut niveau ont été proposés pour permettre la modélisation d'un

système complexe par le biais des formules algébriques compactes, et la dérivation automatique au plus bas niveau, du système de transitions et de la chaîne de Markov sous jacente, qui ont un très grand nombre d'états et de transitions en utilisant des sémantiques définies en termes de transitions. Nous avons opté pour l'usage d'un langage de description d'architecture : AltaRica Data-Flow. Après modélisation du TRS en automate d'état afin de représenter la dynamique de sa variation en fonction de ses trois composantes que sont la Qualité, la Performance et la Disponibilité Opérationnelle, nous avons traduit ces automates d'état en AlataRica Data-Flow, langage à la fois formel et graphique et véritable moteur de modélisation/simulation. L'utilisation de ce langage nous a permis de nous affranchir du classique problème d'explosion combinatoire. De plus, AltaRica Data-Flow permet de manipuler plusieurs modèles automates permettant d'établir des liens de synchronisation entre les sous systèmes évoluant simultanément, et des liens de flux entre ceux évoluant en série ou parallèlement. Cette manipulation de plusieurs modèles d'automates sera importante dans le résultat recherché (passage du local au global, politique des interventions…). L'association des lois de probabilité aux différents événements associés aux transitions permettant le passage d'un état de fonctionnement du système à l'autre est également un des apports indéniables du langage AltaRica Data-Flow. Il permet de gérer avec facilité les processus stochastiques évoluant dans le système sous l'influence des facteurs dégradants de son efficience. L'intégration du comportement de l'Opérateur et du Réparateur dans les modèles AltaRica data-Flow permet également de mettre en évidence les politiques de maintenance et sociale des entreprises. Nous avons simulé les systèmes simples, série et parallèle. Les formules mathématiques utilisées ont été proposées au chapitre 2 en tenant compte des temps d'état d'un moyen de production, répartis selon la norme NFE 60-182. Les résultats de la simulation stochastique de chaque modèle AltaRica ont permis d'évaluer l'efficience du système en tenant compte des valeurs seuils de la World Class Performance.

S'il est vrai que les systèmes de production manufacturiers utilisés comme modèle dans notre étude restent moins complexes que les systèmes critiques (très peu de composants redondants), la perspective de ces travaux est d'évaluer par les mêmes méthodes, l'efficience des systèmes critiques tels que les avions, les TGV etc., qui eux présentent un espace d'état beaucoup plus grand.

ANNEXE A

LE LANGAGE DE MODÉLISATION ALTARICA DATA-FLOW

A.1 Introduction

AltaRica Data-Flow est un langage formel de description de haut niveau consacrée à l'étude de la fiabilité. Il s'appuie sur la notion d'automates de mode. Les automates de mode sont des entrées/sorties des machines à états. Ils généralisent des formalismes tels que des machines à états finis (FSM), les machines Mealy, Moore machines, réseaux de Petri... AltaRica Data-Flow peut être vu comme un moyen pratique de décrire et de combiner les automates de mode.

Le projet AltaRica a débuté en 1997 au Laboratoire Bordelais de Recherche en Informatique (LaBRI, France). Il existe, depuis le tout début, un partenariat fort entre les laboratoires universitaires et les entreprises (dont Total et Dassault Aviation a joué un rôle central). L'objectif principal du projet était de donner une base formelle à un atelier de fiabilité et d'étudier comment l'ingénierie et la fiabilité des méthodes formelles (model-checking) peuvent être croisées. Rapidement, il est devenu clair qu'une telle base formelle ne peut être obtenue que par le biais d'un langage dédié. La première version du langage AltaRica a été conçue par l'équipe LaBRI pendant les années 1998-2000. Cette première version était fortement inspirée des travaux effectués au LaBRI sur le model-checking d'une part (avec notamment le model checker MEC) et la programmation logique, d'autre part.

Au départ, les développeurs d'AltaRica ont cherché uniquement à décrire des comportements qualitatifs, ne faisant intervenir aucune loi de probabilité. Ils ont avant tout recherché les concepts minimaux permettant une projection aisée et rigoureuse des modèles AltaRica vers les modèles traditionnels de fiabilité (Arbre de Défaillances, réseaux de Petri stochastiques, automate à états finis, etc.). Ainsi, ils ont choisi de représenter la dynamique d'un composant par un automate de modes. Dans un automate de modes, des transitions d'automate qui modélisent les enchaînements des modes de fonctionnement et défaillances, cohabitent avec des formules booléennes qui contraignent les valeurs que peuvent prendre les paramètres du système dans chaque mode.

A.2 Présentation du langage AltaRica Data-Flow

Au début des années 2000, Dassault Aviation a décidé de créer son propre atelier basé sur la fiabilité AltaRica (Cécilia OCAS). De sévères restrictions ont été imposées sur le langage pour faire de la compilation dans les arbres de défaillances. Avec le même objectif, Antoine Rauzy qui s'est déplacé du LaBRI pour l'Institut de mathématique de Luminy (IML, Marseille, France) et qui a créé avec d'autres collègues ARBoost Technologies, a conçu une version simplifiée de AltaRica. Dans ce fragment d'AltaRica appelé **AltaRica Data-Flow** [RAU 06], les flux sont orientés et ceci permet de tester plus aisément la complétude de chaque description. Seules des modifications mineures ont été réalisées depuis lors à ce langage, principalement par le biais de la normalisation de la clause "*extern*".

AltaRica Data-Flow est un Langage de Description d'Architecture (ADL) qui a une sémantique basée sur les automates à contraintes. Il permet une modélisation à la fois fonctionnelle et dysfonctionnelle des systèmes, en vue d'une analyse en Sûreté de

Fonctionnement et d'évaluation de performance. La vue dysfonctionnelle est permise grâce à l'introduction d'événements modélisant la défaillance des composants.

Face à la complexification des systèmes à étudier, et à la croissance des exigences de rapidité et exhaustivité des études de Sûreté de Fonctionnement, les fiabilistes ont dû se doter de nouvelles techniques de modélisation. En effet, la construction manuelle de modèles classiques tels que les Arbres de Défaillances, les graphes de Markov, les réseaux de Petri, etc. demande à la fois beaucoup de temps et d'expertise, comporte des risques d'erreur, et ne permet pas la capitalisation des connaissances autrement que dans la tête des analystes.

Pour répondre à tous les besoins, un langage de modélisation idéal (pour la Sûreté de Fonctionnement des systèmes) devrait avoir les caractéristiques suivantes : être compositionnel, pouvoir être partagé par ceux qui font les analyses de Sûreté de Fonctionnement et les concepteurs de systèmes, permettre des études qualitatives et quantitatives de même type que celles que l'on fait avec des méthodes classiques : recherche exhaustive de causes, d'effets, démonstration de propriétés qualitatives, quantification probabiliste. Un tel langage devrait permettre une bonne structuration des connaissances de façon à permettre la capitalisation et la réutilisation des modèles, posséder une syntaxe et une sémantique claires. Il devrait être facile à documenter et à associer à des visualisations graphiques variées.

La modélisation AltaRica Data-Flow de l'efficience d'un système bâtie sur son TRS local et global, se justifie par le fait que:

1. AltaRica Data-Flow est un langage à la fois formel et graphique.
2. AltaRica Data-Flow a été conçu pour modéliser la propagation de pannes au sein d'un système. Il permet donc de modéliser au premier chef des états d'erreur, les événements qui les ont causés, les modes de défaillances induits par ces états. Il se distingue des autres langages dans les moyens fournis pour structurer ces informations et dans les primitives offertes pour décrire la dynamique du système.
3. Les modèles classiques présentent l'inconvénient d'être peu robustes aux changements d'hypothèses car ils ne sont pas "compositionnels". Leur caractère monolithique fait qu'ils sont lisibles uniquement par des experts, et qu'ils sont très peu réutilisables.
4. AltaRica Data-Flow permet de définir des modèles composés d'une liste fixée (il n'y a pas de création ni destruction d'objets en cours d'évolution du système) d'instances d'objets (appelés "**nodes**") en interaction. Ces objets sont organisés de manière hiérarchique : chaque objet contient un certain nombre de caractéristiques propres (états internes d'erreur, événements cause, flux en interfaces) et la déclaration d'objets de niveau inférieur. Par défaut, les règles de visibilité et d'accès aux attributs des objets suivent la hiérarchie. Au niveau le plus abstrait, le système global est représenté par un objet unique. Chaque type d'objet peut être instancié plusieurs fois dans un modèle, mais il n'existe pas de relation d'héritage permettant de factoriser des informations communes à plusieurs types. Les interfaces d'un type sont partitionnées en un nombre fixé d'entrées et un nombre fixé de sorties. Les deux seuls moyens pour faire interagir des objets sont : soit identifier une sortie d'un objet avec une entrée d'un autre, soit définir un "vecteur de synchronisation" capable de provoquer un changement d'état simultané sur deux ou plusieurs objets.
5. Le langage AltaRica Data-Flow est orienté vers la création de modèles qualitatifs pouvant être enrichis par des informations de nature probabiliste.

6. L'exploitation des produits d'automates laisse penser que la composition de systèmes élémentaires par des opérations d'association permet de passer plus formellement du local au global. De plus, la description de ces interactions gagne également en clarté en profitant des notions de liens de flux et de synchronisation entre ces composants. D'autre part, il permet d'intégrer le comportement humain (celui de l'Opérateur et du Réparateur) dans le calcul et la modélisation de l'efficience des systèmes. Ceci est un apport indéniable dans l'évaluation de performance des systèmes de production. La plupart des méthodes et études y afférent se sont toujours limité au simple calcul d'indicateur de performance.

A.3 De l'automate de mode au modèle AltaRica Data-Flow

La notion d'automates de modes a été développée à l'annexe B. Pour illustrer cette notion, considérons le schéma de la vanne de la figure A.1 (partie a)). Son comportement peut être décrit par l'automate de mode de la figure A.1 (partie b). L'état de la vanne est décrit au moyen de deux variables booléennes «*ouvert*» et «*fermé*». Ces variables pourraient également être des entiers ou des réels. Initialement, l'automate de mode est par exemple dans l'état «fermé = vrai». Il peut changer d'état quand un événement se produit. Dans notre exemple, il y a deux événements possibles: «fermé» et «ouvert». Les flux d'entrée et de sortie sont également décrits au moyen de variables. Il n'existe ici qu'un seul flux d'entrée "input" et un seul flux de sortie "output". Les valeurs de flux de production sont entièrement déterminées par les valeurs des flux d'entrée et des variables d'état. Les transitions de l'automate de mode sont caractérisées par une garde, c'est-à-dire une condition booléenne qui indique si la transition est tirable, un événement qui marque la transition, et, enfin, une règle de mise à jour des variables d'état lorsque la transition a été tirée. Le code AltaRica Data-Flow pour cet automate de mode est donné à la figure A.2.

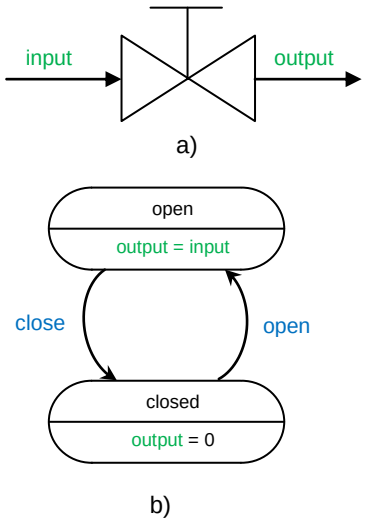


Figure A.1 :
a) Une valve
b) Automate de mode correspondant à la valve

```
node valve
  state closed:bool;
  flow  input:float:in; output:float:out;
  event open, close;
  trans
    closed |- open -> closed:=false;
    not closed |- close -> closed:=true;
  init
    closed := true;
  assert
     output = if closed then 0 else input;
edon
```

Figure A.2 : le modèle AltaRica de l'automate de mode de la figure A.1 b)

A.4 Le méta-modèle AltaRica Data-Flow

Par définition, la méta-modélisation est l'activité de réalisation des méta-modèles. Un méta-modèle expose les constructions syntaxiques, leurs sémantiques, les liens qu'elles entretiennent entre elles, et les règles de bonne formation, nécessaires pour la définition des modèles donnés. Il représente donc la grammaire du langage de modélisation. Pour certains auteurs, le modèle est considéré comme étant une abstraction du système réel, permettant de prédire la qualité du système, d'analyser certaines propriétés lors de l'évolution ou de communiquer avec les autres intervenants dans le cycle de vie du système. D'autres ne distinguent pas le code du modèle. Un modèle est donc défini comme une spécification formelle des fonctions, structure ou comportement du système. Par spécification formelle, on sous-entend que le langage utilisé pour établir cette spécification doit avoir une syntaxe et une sémantique bien définies, avec éventuellement des règles d'analyse, de preuve ou d'inférence pour ces constructions. Le langage AltaRica Data-Flow vérifie cette définition. Son méta-modèle est donné à la figure A.3.

AltaRica
- node Main
- node Intermédiaire [0..*]
- node Feuille [0..*]
- node (Feuille et intermédiaire)
- name -- *unique* --
- flow [0..*] -- *variables de flux* --
  Domain -- *domaine de valeurs, type* --
  Way-- *in, out* --
- state [0..*] -- *variables d'états* --
  Domain-- *domaine de valeurs, type* --
- event [0..*] -- *événements (constantes symboliques)* --
- trans [0..*] -- dynamique des états du nœud --
  --triplet <garde, événement, affectation sur les variables d'état> --
- assert [0..*] -- *contraintes sur les configurations (valeurs des variables)* --
  (node Intermédiaire uniquement)
- sub [0..*] -- *sous-nœuds (=instances de nœuds) de la forme Id.nodeName*--
- sync [0..*] -- *contraintes de simultanéité entre les événements du nœud père et ceux des nœuds fils*--

Figure A.3: Le méta-modèle AltaRica Data-Flow

A.5 Syntaxe et sémantique du langage AltaRica Data-Flow

Un modèle AltaRica est un automate qui génère un graphe des états atteignables à partir d'états initiaux, appelé structure de Kripke (W, W0,Var, Dom(Var), Evt, m, t) définie comme suit :

– W = ensemble des états atteignables et W0, partie de W représentant les états initiaux possibles ;

– Var = ensemble fini de variables dénotant les flux (entrées, sorties) ou les états internes des composants du système ;

– Dom(Var) = ensemble des valeurs que peuvent prendre les variables ;

– Evt = ensemble fini d'événements ;

– m: W × Var → Dom(Var) = fonction d'assignation de valeurs aux variables ;

– t: W × Evt → W = relation de transition entre états étiquetée par les événements.

Cette structure est construite à partir d'un modèle AltaRica de la manière suivante :

– Dans les états initiaux de W0, on alloue tout d'abord une valeur aux variables d'état internes initialisées par une clause « **init** ». Pour les autres variables de Var, on considère toutes les allocations compatibles avec les contraintes établies dans les clauses « **assert** » et l'on peut obtenir ainsi plusieurs états initiaux possibles.

– Dans un état (défini par l'ensemble des valeurs des variables de Var) donné, on examine l'ensemble des transitions des clauses « **trans** ». Elles sont de la forme « garde |- nom_evt -> postconditions ».

- Les gardes sont des expressions booléennes définies à partir de conditions sur les flux et les états internes. Les postconditions affectent explicitement une nouvelle valeur à certaines variables d'état internes et implicitement laissent inchangées les autres. Si dans un état w les valeurs assignées aux flux et aux états internes vérifient les conditions exprimées dans la garde d'une transition t. Les transitions peuvent être groupées par des vecteurs de synchronisation. Selon le type de synchronisation, le groupe sera tirable si toutes ses transitions sont tirables ou si l'une d'elle au moins l'est. Des priorités peuvent être attribuées aux transitions. Parmi les transitions ou groupes de transitions synchronisées, seuls les plus prioritaires sont tirables. Pour chaque transition tirable, on construira tous les successeurs possibles qui assignent aux flux et aux variables internes des valeurs compatibles avec l'effet de la postcondition et avec les assertions. Pour un groupe de transitions synchronisées, le successeur devra satisfaire simultanément les postconditions de chacune des transitions du groupe.

– Le processus est itéré indéfiniment dans le cas le plus général ou jusqu'à ce que l'on ne construise plus de nouveaux états de W dans la plupart des cas traités.

A titre d'illustration, on souhaite modéliser une source électrique. Son état binaire (fonctionnement ou dysfonctionnement) sera représenté par une variable d'état notée *s*. Une sortie de ce système, notée *o*, fournie une valeur booléenne en fonction de l'état de la source. Une défaillance *fail* peut entraîner la perte du système. Dans ce cas le système ne fournit plus de sortie. Le code AltaRica correspondant à ce composant est donné à la figure A.4.

```
node   source
state  s: bool ;
flow   o: out : bool;
event  fail;
trans  s |- fail -> s:=false;
assert o=s;
init     s:=true;
edon
```

Figure A.4 : Modèle AltaRica définissant
le formalisme d'automate à contraintes

Un nœud peut être considéré comme un composant réutilisable de modélisation. Plusieurs instances d'un nœud peuvent apparaître dans le modèle. Une telle description hiérarchique peut être aplatie en un seul d'automate de mode au moyen de simples opérations syntaxiques. Chaque nœud peut être décrit comme un automate à contraintes.

Comme dans les réseaux de Petri, les valeurs des variables (états et flux) ne peuvent évoluer que si la transition est tirée. Lorsqu'une transition est tirée, d'abord les variables d'état sont mises à jour, ensuite les valeurs de flux de sortie sont recalculées en fonction de l'assertion. L'ensemble est supposé être infiniment rapide. Par conséquent, dans AltaRica Data-Flow, le temps est juste considéré comme une séquence d'événements.

Il ya une différence entre la mise à jour des variables d'état et celle des variables de flux. Les variables d'état sont assignées en parallèle, alors que les variables de flux sont assignées en série, en fonction de leurs dépendances. Une illustration en est faite à la figure A.5.

```
...
trans
  (s1=1) and (s2=1) |- e -> s1:=s2+1, s2:=s1;
   ...
assert
  o1 = o2+s1,
  o2 = s2;
...
```

Figure A.5 : Illustration de la mise à jour des variables d'état et de flux

La transition est tirable lorsque les deux variables d'état s_1 et s_2 sont égales à 1. Lorsque la transition est tirée, s_1 et s_2 sont mis à jour en parallèle. S_1 prend la valeur précédente de $s_2 + 1$, alors que s_2 prend la valeur précédente de s_1. Ainsi, les nouvelles valeurs de s_1 et s_2 sont respectivement de 2 et 1. Ensuite, les variables de flux o_1 et o_2 sont mises à jour en fonction de l'assertion et de leurs dépendances. Ici, o_1 dépend de o_2 et s_1, et s_2 dépend de o_2. Par conséquent, o_2 est mise à jour en premier. Sa nouvelle valeur est 1. Enfin, o_1 est mise à jour. Sa nouvelle valeur est 3. Cet exemple illustre également une règle très importante: les assertions ne doivent pas comporter de boucle.

A.6 Interprétation des transitions

Le comportement du composant est décrit par des contraintes sur les changements d'états, appelés *transitions* (trans). Il existe quatre types de transitions en AltaRica Data-Flow :

A.6.1 Les transitions Stochastiques

Les transitions Stochastiques peuvent se produire à tout moment si leur garde est satisfaite, selon certaines distributions probabilistes. Par défaut, les événements et par conséquent les transitions auxquelles ils sont associés sont stochastiques.

A.6.2 Les transitions immédiates

Une transition immédiate est tirée immédiatement dès que sa garde est satisfaite. Les transitions immédiates sont généralement plus prioritaires. Une priorité est un entier. Les transitions immédiates sont tirées dans un ordre décroissant des priorités. Elles sont utilisées pour modéliser la reconfiguration du système en étude.

A.6.3 Les transitions conditionnelles

Les transitions conditionnelles sont un type particulier de transitions immédiates. Elles sont regroupées en paquets. Toutes les transitions d'un paquet doivent avoir la même garde. Lorsque les transitions d'un paquet deviennent tirables, l'une d'entre elles est choisie au hasard selon certaines distributions probabilistes, et est tirée. Les transitions conditionnelles ont été introduites pour modéliser les choix non déterministes (par exemple dans les cas de succès ou de panne à la sollicitation des systèmes redondants).

A.6.4 Les transitions de type Dirac

Les transitions de type Dirac sont tirées après un délai fixe. Elles doivent être utilisées avec précaution. Elles ne peuvent être manipulées que par simulation de Monte-Carlo (les compilateurs d'Arbres de Défaillances, ou de graphes de Markov ne sont pas compatibles avec les transitions de ce type). Une transition de type Dirac avec un retard nul est considéré comme une transition immédiate.

Une illustration de tous ces types de transitions est faite dans l'automate de mode d'un composant réparable en attente dans le cas d'une redondance froide (figure A.6).

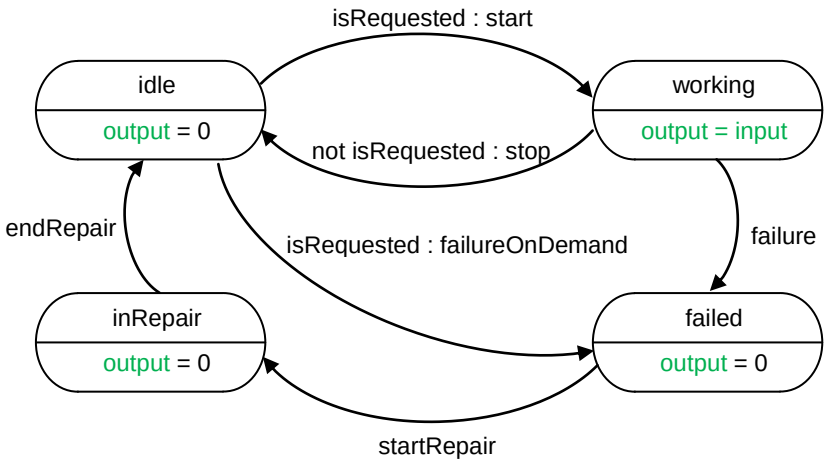


Figure A.6 : Automate de mode d'un composant réparable redondant : Illustration des différents types de transitions

Le composant peut être dans quatre états: "idle", "working", "failed" et "inRepair". Les flux d'entrée booléenne "isRequested" indiquent si le composant est actif ou non. S'il est dans l'état inactif et "isRequested" devient vrai (pour une raison externe), le composant est immédiatement activé. Ici, il ya deux possibilités: soit le composant commence la mission à la sollicitation ou alors il est défaillant. Ces deux transitions sont donc immédiates et conditionnelles réciproquement. Elles ont les mêmes gardes et sont regroupées en paquet. Le taux de transition associé à l'événement "failureOnDemand" est de 0,02 tandis que celui associé à la transition "start" est de 0,98. La somme des taux de transitions d'un paquet devrait être égale à 1.

Dans l'état "working", le flux d'entrée "isRequested" est censé être vrai. Si, pour une raison externe, le composant n'est pas sollicité, la transition "stop" est tirée. Cette transition est immédiate (et non conditionnelle). Pour définir son type, il ya deux possibilités.

Premièrement, l'événement "stop" peut être associé à un Dirac (0) direct. Ensuite, il peut se voir accorder une priorité (ce qui indirectement le décrit comme un événement immédiat).

Toujours dans l'état "working", le composant peut tomber en panne à la suite d'une défaillance. Les défaillances sont des événements stochastiques. Ici, l'événement "failure" est associé à une distribution probabiliste exponentielle de paramètre 0,001.

Pour sortir de l'état "failed", le composant doit emprunter la transition "startRepair". Cette transition est également immédiate. Toutefois, cette transition est généralement synchronisée avec les transitions d'une unité de réparation, ceci conditionne son passage dans l'état "inRepair".

La réparation du composant peut prendre un temps donné. Ce temps est fixe et connu (10 unités de temps par exemple). Ainsi, l'événement "endRepair" est associé à un Dirac (10) de distribution.

Le modèle AltaRica Data-Flow correspondant à l'automate de mode de la figure A.6 est représenté à la figure A.7.

```
node    SpareUnit
state   s:{idle, working, failed, inRepair};
flow    isRequested:bool:in; input:float:in output:float:out
event   failureOnDemand, start, stop, failure, startRepair, endRepair;
trans
        (s=idle)       and isRequested     |- failureOnDemand -> s := failed;
        (s=idle)       and isRequested     |- start           -> s := working;
        (s=working)                        |- failure         -> s := failed;
        (s=working)    and not isRequested |- stop            -> s := idle;
        (s=failed)                         |- startRepair     -> s := inRepair;
        (s=inRepair)                       |- endRepair      -> s := idle;
assert
        output = if (s=working) then input else 0;
init
        s := idle;
extern
        law <event failure>                = exponential(0.001);
        law <event endRepair>              = Dirac(10);
        law <event failureOnDemand>        = constant(0.02);
        law <event start>                  = constant(0.98);
        bucket {<event failureOnDemand>, <event start>}   = onDemand;
        priority {<event stop>, <event startRepair>}      = 1;
edon
```

Figure A.7 : Modèle AltaRica Data Flow de l'automate de mode de la figure A.6

A.7 Déclarations des domaines et les nœuds

Une description AltaRica est faite d'une série de déclarations de domaines, de nœuds (node) et de fonctions. Elle doit contenir au moins la déclaration d'un nœud appelé «**main**». La forme générale d'une description AltaRica est donc la suivante.

```
description ::= declaration*
declaration ::= domain-decl
            ::= node-decl
```

Les domaines doivent être déclarés avant leur utilisation. Le reste de cette section détaille les déclarations ci-dessus.

A.7.1 Déclaration des domaines

Un domaine est soit un ensemble de constantes symboliques ou une gamme numérique. La syntaxe des déclarations de domaine est la suivante.

```
domain-decl ::=  domain identifier = domain
domain  ::=  bool | int | float
        ::=  { identifier (, identifier)* }
        ::=  [ int, int ]
```

Les domaines "int" et "float" doivent être manipulés avec précaution, car ils sont considérés comme infinis (même si un nombre fini d'entiers ou de flottants peut être représentés dans les ordinateurs).

A.7.2 Déclaration des nœuds (node)

La déclaration d'un nœud est constituée de trois parties: la déclaration du nœud éléments (variables d'état, variables de flux, les événements et les sous-nœuds), le nœud corps (état initial, les transitions, l'assertion et la synchronisation des vecteurs) et une clause „extern“.

```
node-decl ::=
 node identifier
   node-elem
   node-body
   extern-cls
edon

node-elem    ::= state-cls? flow-cls? event-cls? sub-cls?

state-cls      ::= state state-decl+
state-decl     ::= identifier : domain ;
flow-cls       ::= flow flow-decl+
flow-decl      ::= identifier : domain : orientation ;
orientation    ::= in | out | local
event-cls      ::= event event-decl (, event-decl)* ;
event-decl     ::= identifier
sub-cls        ::= sub sub-decl+
sub-decl       ::= identifier : identifier ;
```

Figure A.8 : Déclaration des nœuds (node)

Les clauses "state", "flow", "event" et "sub" peuvent être utilisées dans n'importe quel ordre. Les nœuds peuvent contenir des sous-nœuds. Leurs définitions peuvent être données dans n'importe quel ordre. Les définitions des nœuds ne doivent comporter aucune boucle.

Les variables de flux sont typées. Elles sont "in", "out" ou "local". Bien que ce typage soit obligatoire dans la version actuelle de la boîte de dialogue, il pourrait facilement être induit du reste de la description. Une variable de flux est une variable de sortie si elle se situe à gauche d'une assertion. Dans le cas contraire, c'est une variable d'entrée. Les variables locales sont comme des variables de sortie.

Comme pour les éléments, les clauses "init", "trans", "assert" et "sync" peuvent être donnés dans n'importe quel ordre. Les affectations, initialisations et assertions sont effectuées "en parallèle". Cela signifie que la variable d'état ne peut se produire qu'une fois seulement comme membre de gauche d'une série d'assertions ou d'initialisations. De même, une variable de flux ne peut se produire qu'une fois comme membre de gauche d'une série d'assertions.

```
node-body  ::=  init-cls? trans-cls? assert-cls? sync-cls?

init-cls        ::= init (initialization (, initialization)*)  ;
initialization ::=  state := constant

trans-cls      ::= trans transition
transition     ::=
  bool-expr  |- event -> (assignment (, assignment)*)  ;
assignment ::=  state := expr

assert-cls     ::= assert (assertion (, assertion)*)  ;
assertion      ::= flow = expr

sync-cls       ::=  sync vector-decl
vector         ::= < event (, event)* >
               ::= < event : sync-term >

sync-term      ::= true | false
               ::= event
               ::= sync-term (or sync-term)
               ::= sync-term (and sync-term)
```

Figure A.9 : Déclaration des clauses : "init", "trans", "assert" et "sync"

A.8 À propos de la synchronisation relationnelle

Les synchronisations sont un moyen puissant pour modéliser les interactions entre les composants d'un système. AltaRica Data-Flow met en œuvre un mécanisme de synchronisation d'origine, à savoir la notion de synchronisation relationnelle. En AltaRica Data-Flow, un vecteur de synchronisation peut être considéré comme une équation de la forme „‟event = expression‟‟, où „‟event‟‟ est un événement du nœud ou de ses nœuds secondaires et „‟expression‟‟ est une expression booléenne monotone construite sur les constantes vrai et faux, d'événements, et les deux connecteurs ET et OU. Une clause de synchronisation peut donc être considérée comme un ensemble de ces équations. La syntaxe

actuelle ne reflète pas cette idée dans un souci de compatibilité avec d'autres versions d‟AltaRica. Toutefois, il est préférable de considérer la synchronisation en termes d'équations, comme présenté ci-dessus. Nous donnons ci-dessous deux vecteurs de synchronisation et de leurs interprétations sous forme d‟équation.

```
<startRepair,Unit1.startRepair, RepairCrew.startJob>
  --> startRepair = Unit1.startRepair and RepairCrew.startJob

<CCF: Unit1.failure or Unit2.failure>
  --> CCF = Unit1.failure or Unit2.failure
```

Figure A.10 : Illustration d'une synchronisation relationnelle

Le mécanisme de synchronisation a trois effets:

* tout d'abord, il supprime toutes les transitions marquées par un des événements qui se produisent dans les équations.

* deuxièmement, il introduit de nouvelles transitions construites à partir de celles supprimées et la synchronisation des équations.

* troisièmement, il permet d'effacer (ou de masquer) tous les événements associés à la clause synchronisation qui ne se trouvent pas dans le membre gauche de l'équation.

A.9 La clause "extern"

La clause "extern" a été ajoutée aux déclarations des nœuds afin de transmettre des morceaux d'informations structurées à des outils d'évaluation. Par exemple les lois de probabilité associées à des événements ne font pas partie du langage AltaRica Data-Flow. Toutefois, divers outils, y compris les simulateurs Monte-Carlo et compilateurs des Arbres de Défaillance, ont besoin de cette information.

A.9.1 La syntaxe de la clause "extern"

La syntaxe de la clause „'extern‟" doit répondre à deux exigences. Tout d'abord, il doit être possible de se référer à des éléments de descriptions comme des événements ou des expressions. Ces éléments doivent être mis à jour lors d‟une insertion d‟une instance du nœud dans une hiérarchie. Deuxièmement, il doit être possible de décrire tout morceau d‟information relevant du contexte.

La clause „"extern‟" est donc une liste de termes externes. Un terme externe peut être soit une valeur (une constante, un identificateur ou une chaîne, ou un ensemble de termes externes, ou un identifiant suivi d'une liste de paramètres qui sont eux-mêmes externes. Des fragments de descriptions sont joints entre "<" et ">". Un mot clé indique le type de fragment joint.

```
extern-cls ::=  extern extern-decl+
extern-decl ::= type extern-term [= extern-term]?;

type ::= identifier

extern-term ::= number | string | identifier
          ::= identifier “(” extern-term (,extern-term)* “)”
          ::= { extern-term (, extern-term)* }
          ::= <event event>
          ::= <term expr>
          ::= <local expr>
          ::= <global expr>

string ::=  any text between two quotes ".
```

Figure A.11 : Syntaxe de la clause “extern”

A.9.2 Prédicats et propriétés

L'une des utilisations les plus simples de la clause „extern“ est la définition de propriétés et des prédicats, c'est-à-dire que des quantités sont calculées à partir des modèles et observées par les différents outils. Ces quantités peuvent être définies comme des variables d'état ou de flux, mais il est souvent préférable de séparer le comportement de l'observation. Dans la boîte de dialogue du langage AltaRica Data-Flow, les prédicats sont une valeur booléenne, tandis que les propriétés sont des entiers ou des valeurs réelles. Puisque le rôle des prédicats et des propriétés est d'observer le modèle, il doit y avoir un moyen de connecter la clause „extern“ avec les variables d'état et de flux. La directive <term expr> est utilisée dans ce but. L'expression à l'intérieur de cette directive est évaluée comme des expressions dans des transitions et des assertions. Considérons par exemple ce système de la figure A.10 constitué de deux composants réparables et d'un réparateur. Supposons que nous voulons observer lorsque les deux composants sont défaillants, et lorsqu'au moins un est en réparation. Pour ce faire, il suffit de définir deux prédicats (puisque les quantités observées sont booléennes). Par exemple :

```
node System
  ...
sub Unit1:RepairableUnit; Unit2:RepairableUnit; Crew:RepairCrew;
  ...
extern
   predicate failed          = <term ((Unit1.s=failed) and (Unit2.s=failed))>;
   predicate oneInRepair = <term ((Unit1.s=inRepair) or (Unit2.s=inRepair))>;
edon
```

Figure A.12 : Déclaration des prédicats

A.10 Événements stochastiques

Les transitions stochastiques peuvent être franchies à tout moment si leur garde est satisfaite en fonction de certaines distributions probabilistes. Par défaut, les événements et les transitions qu'elles marquent sont stochastiques.

A.10.1 Distributions probabilistes

Aux événements stochastiques doivent être associées des distributions probabilistes. Il existe essentiellement deux distributions probabilistes qui peuvent être interprétées dans le contexte AltaRica: la distribution exponentielle et la distribution de Weibull (les constantes et les distributions de Dirac ne peuvent pas être associées à des événements stochastiques). Les deux donnent la probabilité $p_e(t)$ que l'événement *e* est tiré avant un certain délai *t*, ou inversement, dans un contexte de simulation, $d_e(z)$ le retard avant le tir de *e* donné par la probabilité *z*. Le tableau A.1 suivant donne ces deux fonctions ainsi que la description des paramètres des distributions de Weibull et exponentielle.

Tableau A.1 : Interprétations des distributions probabilistes

Distribution	**Paramètres**	$p_e(t)$	$d_e(z)$
exponentielle	λ: taux	$1 - e^{-\lambda t}$	$-\frac{\log(z)}{\lambda}$
Weibull	α: paramètre d'échelle β: paramètre de forme	$1 - \exp\left[-\left(\frac{t}{\alpha}\right)^{\beta}\right]$	$\alpha \times [-\log(z)]^{\frac{1}{\beta}}$

Les déclarations permettant d'associer les distributions exponentielle et de Weibull à un événement *e* sont les suivantes.

```
law <event e> = exponential(lambda);
law <event e> = Weibull(alpha,beta);
```

Dans les déclarations ci-dessus, lambda, alpha et beta sont des paramètres.

A.10.2 Paramètres des distributions probabilistes

Comme distributions probabilistes, certains paramètres peuvent être interprétés dans le contexte AltaRica tandis que d'autres ne peuvent passés que comme outils d'évaluation. Les paramètres qui peuvent être interprétés sont les suivants.

- les constantes: 0.001, 1.6e-4...
- les paramètres nommés: lambda, mu...
- les sommes et produits des paramètres : somme (lambda1, lambda2), produit (3, lambda)...

Les paramètres nommés sont définis par le biais des déclarations „paramètre“. Leur syntaxe est la suivante.

```
parameter name = parameter;
```

Où *name* est le nom du paramètre et *parameter* est un paramètre valide (en effet, les définitions de paramètres ne doivent pas contenir de boucle).

A.10.3 Évènement immédiat

Une transition immédiate est tirée immédiatement dès que sa garde est validée. Elle est plus prioritaire que les transitions stochastiques. Il y a deux façons de typer un événement immédiat. La première consiste à lui donner une priorité (supérieur à 0). Par exemple :

```
priority <event startRepair> = 1;
```

La deuxième façon est d'associer l'événement à une distribution probabiliste avec un Dirac (0). Par exemple :

```
law <event endRepair> = Dirac(0);
```

A.10.4 Évènement conditionnel

Les transitions conditionnelles sont un type particulier de transition immédiate. Elles sont regroupées (bucketées). Toutes les transitions d'un même groupe (bucket) doivent avoir la même garde. Lorsque les transitions d'un groupe deviennent tirables, l'une d'elles est choisie aléatoirement selon certaines distributions probabilistes, et est tirée. Pour typer un événement comme conditionnel, il faut l'associer à une distribution probabiliste et l'insérer dans un bucket (groupe de transition). Par exemple :

```
bucket { <event start>, <event failureOnDemand> } = onDemand;
law <event start> = constant(0.98);
law <event failureOnDemand> = constant(0.02);
```

La somme des probabilités des événements d'un bucket doit être égale à 1.

A.10.5 Évènements Dirac

Les transitions de type Dirac sont tirées après un délai fixe. Un événement est un événement Dirac s‟il est associé à une distribution Dirac non nulle. Par exemple :

```
law <event endRepair> = Dirac(10);
```

A.10.6 Préemption

Quand une transition stochastique ou une transition Dirac (non nul) devient tirable, un retard est défini (tiré au hasard dans le cas de transitions stochastiques, et fixé dans le cas de transitions Dirac) et la transition est supposée plus longue à cet instant avec la durée du retard. Si la transition reste tirable au cours de la durée du retard, elle est tirée à l‟échéance (date prévue). Cela pourrait être le cas, cependant, à cause du fait que d‟autres transitions sont tirées, elle est désactivée avant la fin du délai. Ici, il ya deux politiques:

* le temps passé dans le mode tirable est oublié. La prochaine fois que la transition devient tirable, un nouveau délai est fixé. C'est le défaut de cette politique dans AltaRica Data-Flow.

* le temps passé dans le mode tirable est mémorisé. La prochaine fois que la transition devient tirable, le délai avant le tir aura déjà diminué à cause du temps passé dans le mode tirable. Dans ce cas, l'événement est dit préemptible.

La déclaration permettant d'indiquer qu'un événement *e* est préemptible est la suivante :

```
preemptible <event event>;
```

A.10.7 Attributs des événements

Il est parfois utile de regrouper les événements en catégories. Une façon d'y parvenir consiste à associer des attributs aux événements. Les déclarations permettant d'associer un attribut à un événement et une série d'événements sont les suivants :

```
attribute name(<event event>) = value;
attribute name({ <event event> [, <event event>]* }) = value;
```

Dans la déclaration ci-dessus, le nom est un identificateur et la valeur est soit une constante, une chaîne, un identificateur ou une déclaration globale englobant une constante (l'intérêt de celui-ci est d'éviter le préfixage d'identification au cours de l'aplatissement des hiérarchies). Par exemple :

```
attribute zone(<event failure>) = "room 65";
attribute type({<event fail1>, <event fail2>}) = <global severe>;
```

A.11 CONCLUSION

AltaRica Data-Flow hiérarchise les composants qui sont considérés comme des nœuds (node). On peut ainsi utiliser un nœud à l'intérieur d'un autre nœud (sub) et les faire communiquer. Chaque nœud possède un nombre fini de variables internes dites d'état (state), dont les valeurs sont locales et isolées de l'environnement dans lequel elles se trouvent, et également un nombre fini de variables de flux (flow) qui font le lien entre le composant et son environnement. Très souvent ces deux types de variables sont liés. Un point important d'AltaRica Data-Flow est que les valeurs des variables sont toujours discrètes. De plus, chaque nœud peut être décrit comme un automate à contraintes. En effet, le comportement du composant est décrit par des contraintes sur les changements d'états, appelés *transitions*, où la satisfaction d'une garde composée de variables d'entrée et d'état, ainsi que l'occurrence d'un événement (event) permet de changer l'état. Il peut également changer d'état interne par des transitions causées par des événements spéciaux (notés ε) qui ne sont pas spécifiées dans le modèle lui-même mais définis au niveau sémantique et qui permettent de garantir qu'un nœud ne bloque pas. On peut également utiliser des contraintes globales sur les variables, appelées *assertions* qui permettent de valuer les sorties du composant en fonction de variables d'entrée et d'état. Notons que trois types de constantes peuvent être utilisées : les constantes entières, énumérées et booléennes. L'état initial du composant est déclaré après la clause *init*. Il est également possible de fournir, dans un code AltaRica, des informations utiles à des outils de traitement externe. Ces informations pourront correspondre à des probabilités d'occurrence associées à des événements. Elles seront ensuite traitées par des outils de traitement de coupes minimales. Ces données supplémentaires seront entrées après la clause *extern* dans le code AltaRica.

ANNEXE B

LES AUTOMATES D'ÉTAT

B.1 Système de transitions

Un système de transitions consiste en un ensemble de configurations et de transitions permettant de décrire des processus dynamiques. Les configurations correspondent aux états du processus et les transitions renseignent sur son évolution. Il existe plusieurs types de systèmes de transition comme les systèmes étiquetés ou paramétrés. L'objectif principal de cette description formelle des processus est de pouvoir les analyser et analyser leurs propriétés. Traditionnellement, deux types de propriétés sont intéressantes : les propriétés dites de *sûreté* et celle dites de *vivacité*. Une propriété de *sûreté* est utilisée pour exprimer que quelque chose de non souhaitée ne doit jamais se produire, comme, par exemple, vérifier qu'en loi normale les commandes de vol d'un avion ne permettent jamais d'atteindre un angle de roulis à 67°. Les propriétés de *vivacité* expriment le fait que quelque chose de souhaitée finira toujours par se réaliser au moins une fois, par exemple, la perte totale du système électrique d'un A320 doit aboutir au déploiement de l'éolienne de secours située sous l'appareil. Définissons maintenant les systèmes de transition sur lesquels il est possible d'analyser ce genre de propriétés.

Un système de transitions est un quadruplet
$A = \langle S, S_0, T, F \rangle$ avec:

- S un ensemble fini ou infini d'états
- $S_0 \subseteq S$ un ensemble fini ou infini d'états initiaux
- $T \subseteq S \times S$ un ensemble fini ou infini de transitions
- $F \subseteq S$ un ensemble fini ou infini d'états finaux

Notons que des transitions particulières appelées ε-transitions initiées par un événement ε, interne au processus et inobservable, permettent de garantir qu'un processus ne bloque pas. Si *S* et *T* sont finis on parle de systèmes de transitions finis. Si *S* est fini on parle alors aussi d'automate.

B.2 Chemin d'un système de transitions

On appelle chemin de longueur n (n > 0) d'un système de transitions fini A une suite de transitions t_1, …, t_n telle que $\forall i, 1 \le i \prec n, t_i = (s_i, s_{i+1})$ Alors s_1 appartient à l'ensemble $S_0 \subseteq S$ des états initiaux de *A* et s_n appartient à l'ensemble *F* des états finaux de *A*.

De même, un chemin infini est une suite infinie de transitions t_1, …, t_n telle que $\forall i \ge 1, t_i = (s_i, s_{i+1})$. Comme dans le cas des chemins finis, $s_1 \in S_0$. Etant donné qu'il n'est pas possible d'imposer à l'état terminal d'un chemin infini d'appartenir à *F*, une autre contrainte doit être définie et satisfaite pour ce type de chemins. Dans le cas des systèmes de transitions infinis, cette contrainte correspond au fait de devoir passer une infinité de fois par l'ensemble des états de *F*.

Exemple : Considérons un système à deux composants en redondance froide. Un des composant n'est pas activé initialement (appelé secondaire). Il ne s'active qu'après détection de la défaillance du composant initialement activé (appelé primaire). Chaque composant peut défaillir à tout moment. Le système de transitions correspondant à ce processus est représenté à la figure B.1. L'état 1 correspond à l'état initial où les deux composants sont opérationnels, le composant 1 est actif alors que le 2 ne l'est pas. Après défaillance du composant 1, le système passe dans l'état 2 où il est possible d'activer le composant 2 pour passer dans l'état 5. Si l'on observe une panne cachée du composant 2 le système passe dans l'état 3. La défaillance du système dans les états 2 ou 3 mène le système dans l'état 4. On passe finalement de l'état 5 à l'état 6 sur défaillance du composant 2.

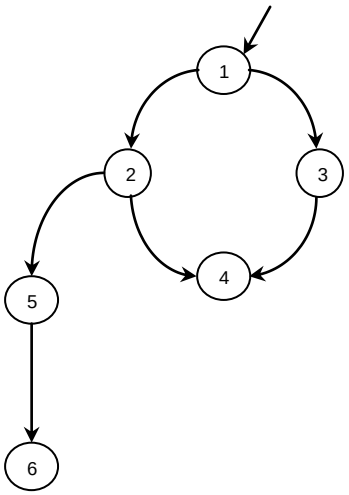


Figure B.1 : Système de transitions

L'avantage de ce formalisme réside dans sa simplicité de conception et de compréhension pour les systèmes de taille raisonnable. En effet, il est nécessaire d'expliciter l'ensemble des états du processus ce qui devient rapidement problématique pour des cas pratiques réaliste. On est alors face au problème très connu d'explosion combinatoire du nombre d'états.

Cependant, à ce stade, les systèmes de transitions ne permettent pas une représentation détaillée des processus. Une transition, par exemple correspond à une action précise, il serait intéressant de représenter le nom de cette action sur le graphe. De la même manière il serait également intéressant de représenter l'ensemble des propriétés satisfaites en un état du graphe. On passe alors dans le domaine des systèmes de transitions étiquetés.

B.3 Système de transition étiqueté

Un système de transition étiqueté est un quintuplé

$A = \langle S, S_0, \Sigma, T, F \rangle$ Avec:

- S un ensemble fini ou infini d'états
- $S_0 \subseteq S$ un ensemble fini ou infini d'états initiaux
- Σ un ensemble fini ou infini d'événements
- $T \subseteq S \times \Sigma \times S$ un ensemble fini ou infini de transitions

- $F \subseteq S$ un ensemble fini ou infini d'états finaux

Le système de transition étiqueté de l'exemple de la figure A.1 est représenté à la figure B.2.

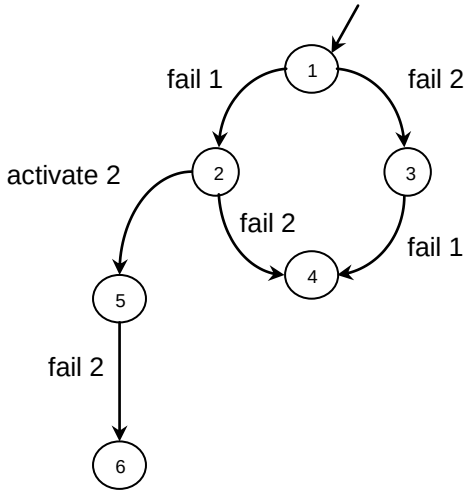


Figure B.2 : Système de transitions étiqueté

Afin d'augmenter l'expressivité des systèmes de transitions pour faciliter la modélisation de systèmes réels, il est possible d'étendre leur modèle. Ainsi, les automates à propriétés permettent d'associer une propriété logique à chaque état du graphe. Une autre extension possible consiste à introduire un ensemble fini de variables discrètes dans le modèle des systèmes de transitions. On peut ainsi définir des gardes aux transitions qui imposent certaines conditions sur les variables pour autoriser leur franchissement. Dans cette quête croissante d'expressivité, il est possible de généraliser le concept d'automates à variables en introduisant des paramètres entiers en quantité éventuellement infinie. Un jeu de valuation de ces paramètres permet de caractériser un état et à chaque transition est toujours associée une contrainte sur ces paramètres. C'est le principe des *automates à contraintes*.

B.4 Automates à contraintes

Un *automate à contraintes* est un automate à états finis usuel dont les états et transitions ne sont pas définis explicitement mais par des contraintes sur les variables. Intuitivement, un automate à contraintes se décrit à l'aide d'un ensemble de transitions de la forme :

$G(V) \xrightarrow{e} V = \sigma(V)$ avec V un ensemble de variables. Cette transition est franchie si la garde *G(V)* définissant une contrainte sur les variables de *V* est satisfaite et l'événement *e* se produit. On passe alors dans un nouvel état défini par la valuation de ses variables $V = \sigma(V)$.

Un automate à contraintes est un 7-uplet $A = \langle D, S, F, \Sigma, T, A, I \rangle$ avec :

- D est le domaine fini ou infini
- S et F sont deux ensembles disjoints de variables appelés respectivement variables de d'état et de flux.
- Σ est un ensemble d'événements

- T est un ensemble de transitions. Une transition est un triplet (g, e, a) avec $g \subseteq D^{|S \cup F|}$ une contrainte sur $S \cup F$ appelée *garde*, $e \in \Sigma$ et a est une application qui définit les états successeurs : $a : D^{|S \cup F|} \to D^{|S|}$
- $A \subseteq D^{|S \cup F|}$ est un ensemble d'assertions i.e. de contraintes sur les valeurs de variables
- $I \subseteq D^{|S \cup F|}$ est l'ensemble des états initiaux

Une restriction intéressante des automates à contraintes permet de travailler sur un nombre fini de variables et d'orienter les variables flux qui ne sont plus alors bidirectionnelles. Cette restriction correspond au modèle des automates de mode.

B.5 Automate de mode

Un automate de mode est un 9-uplet $A = \langle D, S, F^{in}, F^{out}, dom, \Sigma, \delta, \sigma, I \rangle$ avec :

- D est un domaine fini
- Notons V l'ensemble fini de variables scindées en trois catégories S, F^{in}, F^{out}. Ces variables sont des sous ensembles de V deux à deux disjoints appelées respectivement variables d'état, de flux d'entrée et de flux de sortie
- $dom : V \to 2^D$ tel que $\forall \upsilon \in V, dom(\upsilon) \neq \emptyset$ associe à une variable son domaine
- Σ est un ensemble fini d'événements
- δ est une fonction partielle appelée transition: $dom(S) \times dom(F^{in}) \times \Sigma \to dom(S)$, $dom(S) \times dom(F^{in})$ est la garde de la transition
- σ est une fonction totale appelée assertion : $dom(S) \times dom(F^{in}) \to dom(F^{out})$
- $I \subset \sigma$ est une fonction partielle qui définit les conditions initiales

Exemple : Reprenons l'exemple de la redondance froide de deux composants identiques. Dans cet exemple, les composants C_1 et C_2 sont des sources électriques qui redondent passivement. Chaque composant C_i possède un état noté s_i ainsi qu'une défaillance $fail_i$, et C_2 possède un événement d'activation (*activate2*) permettant de changer son état a_2 d'activation.

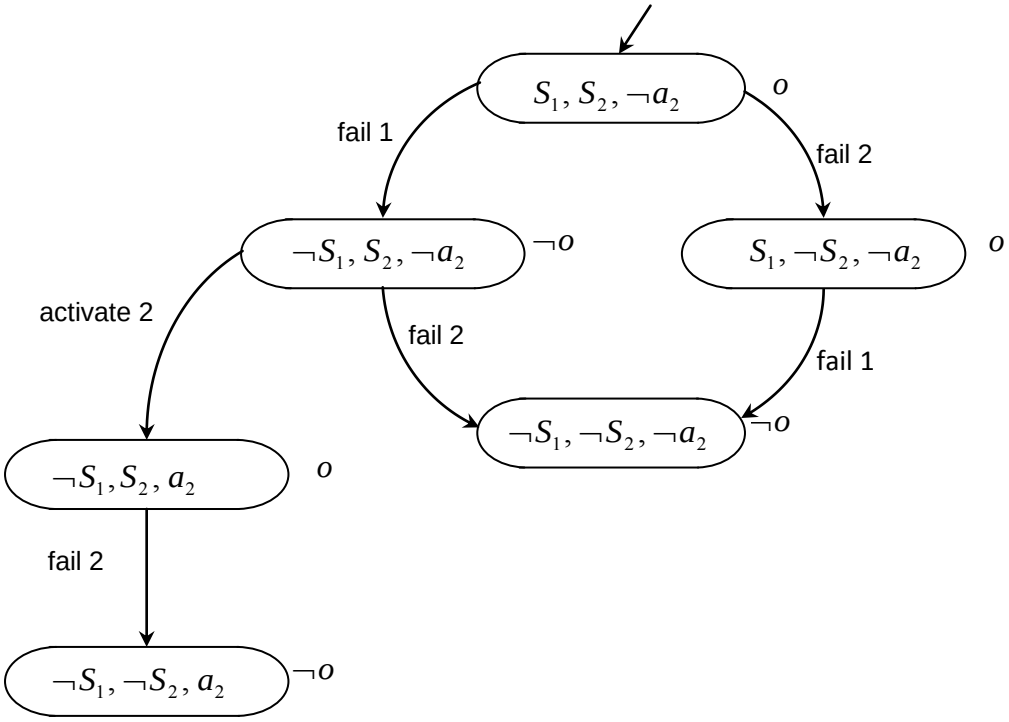


Figure B.3: Automate de mode

Les sorties o_1 et o_2 des composants permettent de construire une sortie unique notée $o = o_1 \vee o_2$. Toutes ces variables sont booléennes. Initialement les composants sont non-défaillants (i.e. $s_i = 1$) et le composant C_2 n'est pas activé. Dans cet état o est vraie. Si C_1 défaille alors C_2 peut être activé et prend le relai. Dans le cas où les deux composants sont défaillants ($fail_1$ et $fail_2$ ont été franchies) o devient faux. L'automate équivalent est présenté à la figure B.3. Il faut noter que pour des raisons de simplicité, les ε-transitions n'ont pas été modélisées.

Les caractéristiques de cet automate sont les suivantes :

- $V = S \cup F^{in} \cup F^{out}$ *avec* $S = \{S_1, S_2, a_2\}$, $F^{in} = \{\emptyset\}$, $F^{out} = \{o\}$
- Toutes ces variables sont booléennes :
$dom(S_i) = dom(a_2) = dom(o) = \{true,\ false\}$
- Les événements sont : $\Sigma = \{ fail_1,\ fail_2, activate_2\}$
- Les transitions sont :
$\delta(S_1, S_2, \neg a_2,\ fail_1) = \langle \neg S_1, S_2, \neg a_2 \rangle$,
$\delta(S_1, S_2, \neg a_2,\ fail_2) = \langle S_1, \neg S_2, \neg a_2 \rangle$,
$\delta(S_1, \neg S_2, \neg a_2,\ fail_1) = \langle \neg S_1, \neg S_2, \neg a_2 \rangle$,
$\delta(\neg S_1, S_2, \neg a_2\ fail_2) = \langle \neg S_1, \neg S_2, \neg a_2 \rangle$,
$\delta(\neg S_1, S_2, \neg a_2, activate_2) = \langle \neg S_1, S_2, a_2 \rangle$ *et*
$\delta(\neg S_1, S_2, a_2,\ fail_2) = \langle \neg S_1, \neg S_2, a_2 \rangle$
- L'assertion est : $o = (S_1 \vee (S_2 \wedge a_2))$ donc σ est telle que :

$\sigma(S_1, S_2, \neg a_2) = \langle o \rangle$

$\sigma(S_1, \neg S_2, \neg a_2) = \langle o \rangle$

$\sigma(\neg S_1, S_2, a_2) = \langle o \rangle$

$\sigma(\neg S_1, S_2, \neg a_2) = \langle \neg o \rangle$

$\sigma(\neg S_1, \neg S_2, \neg a_2) = \langle \neg o \rangle$

$\sigma(\neg S_1, \neg S_2, a_2) = \langle \neg o \rangle$

- L'état initial est : $(S_1, S_2, \neg a_2)$

B.6 Connexion de composants

La connexion de composants, i.e. d'automates de modes, selon le concept AltaRica Data-Flow consiste à contraindre certaines variables d'entrée d'un composant à être égales à une sortie d'un autre composant. Pour que cette connexion soit valide, il est nécessaire que ces variables soient indépendantes, c'est-à-dire que les variables d'entrée n'interviennent pas dans les calculs de la valeur de la variable de sortie. D'un point de vue pratique, une connexion peut être assimilée à un renommage de certaines variables.

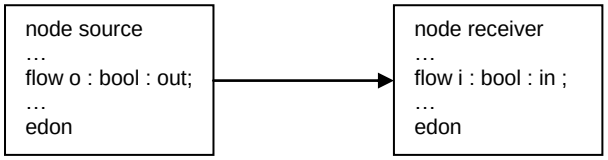


Figure B.4 : connexion de composants en AltaRica Data-Flow

Soit ν un jeu de valuation des variables, nous noterons dans la suite $\nu[u]$ la valeur de la variable u dans cette valuation ν. Soit $A_i = \langle D, S_i, F_i^{in}, F_i^{out}, dom, \Sigma_i, \sigma_i, \delta_i, I_i \rangle$ un automate de mode, $o \in F^{out}$ et $i_1, ..., i_k \in F^{in}$ tel que o et $i_1, ..., i_k$ soient indépendants et $dom(o) \subseteq dom(i_1), ..., dom(o) \subseteq dom(i_k)$. Soient $F_*^{in} = F^{in} \setminus \{i_1, ..., i_k\}$, $s \in dom(s)$, et $I \in dom(F_*^{in})$. Notons $I_{s, o|i_1, ..., o|i_k}$ la valuation de F^{in} telle que :

$$I_{s, o|i_1, ..., o|i_k}[u] \stackrel{def}{=} \begin{cases} I[u] & si\ u \in F_*^{in} \\ \sigma(s, I')[o] & sinon \end{cases}$$

Avec I' n'importe quelle extension de I dans la valuation de F^{in}. L'automate de mode A où $i_1, ..., i_k$ sont connectés à o est l'automate de mode $A_{o|i_1, ..., o|i_n} = \langle D, S, F_*^{in}, F^{out}, dom, \Sigma, \delta', \sigma', I \rangle$ avec δ' et σ' tels que :

- $\forall s \in dom(S), I \in dom(F_*^{in})$ et $\varepsilon \in \Sigma$ si $\delta(s, I_{s, o|i_1, ..., o|i_n}, e)$ est défini, alors $\delta'(s, I, e) = \delta(s, I_{s, o|i_1, ..., o|i_n}, e)$

- $\forall s \in dom(S), I \in dom(F_*^{in}), \sigma'(s, I) = \sigma(S, I_{s, o|i_1, ..., o|i_n})$

B.7 Composition parallèle

La composition consiste à placer plusieurs nœuds AltaRica dans un même cadre, une même vue afin de pouvoir les faire communiquer et interagir.

Soient $A_1, ..., A_n$ n automates de mode, avec $A_i = \langle D, S_i, F_i^{in}, F_i^{out}, dom, \Sigma_i, \delta_i, \sigma_i, I_i \rangle$ et tels que leurs vocabulaires soient distincts,
i.e. $\forall i, j, 1 \le i \le j \le n, (S_i \cup F_i^{in} \cup F_i^{out}) \cap (S_j \cup F_j^{in} \cup F_j^{out}) = \emptyset$ et $\Sigma_i \cap \Sigma_j = \emptyset$
Alors la composition parallèle (parfois appelée aussi produit libre) de $A_1, ..., A_n$ est l'automate de mode $A = \langle D, S, F^{in}, F^{out}, dom, \Sigma, \delta, \sigma, I \rangle$ tel que :

- $S = \bigcup_{i=1}^{n} S_i$, $F^{in} = \bigcup_{i=1}^{n} F_i^{in}$, $F^{out} = \bigcup_{i=1}^{n} F_i^{out}$, $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$
- δ est obtenu en remontant les δ_i dans A, i.e. soient $s_1 \in dom(S_1), ..., s_n \in dom(S_n)$ soient $i_1 \in dom(F_1^{in}), ..., i_n \in dom(F_n^{in})$, soit $t_i \in dom(S_i)$ et $\varepsilon \in \Sigma_i$ tel que $t_i = \delta_i(S_i, I_i, e)$
Alors $\delta(s_1, ..., s_n, i_1, ..., i_n, e) = \langle s_1, ..., s_{i-1}, t_i, s_{i+1}, ..., s_n \rangle$
- de la même façon, σ est obtenu en remontant les σ_i dans A,
$\sigma(s_1, ..., S_n, i_1, ..., i_n) = \langle \sigma_1(s_1, i_1), ..., \sigma_n(s_n, i_n) \rangle$
- $I = \langle I_1, ..., I_n \rangle$

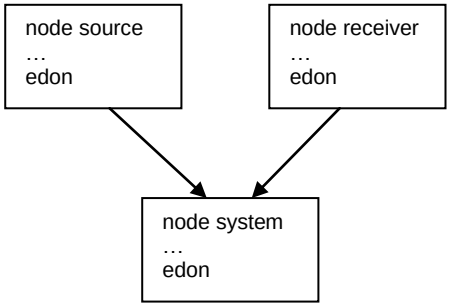


Figure B.5 : Composition parallèle de composants en AltaRica Data-Flow

B.8 Synchronisation

Les transitions dans les automates de mode sont asynchrones. L'opération de synchronisation permet de tirer plusieurs transitions simultanément. Les événements ainsi synchronisés sont regroupés au sein d'un vecteur de synchronisation. Nous allons tout d'abord présenter la notion de composition de valuations puis l'opération de synchronisation.

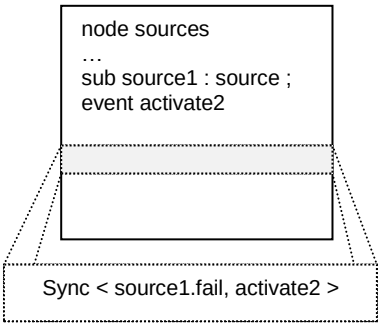


Figure B.6 : Synchronisation en AltaRica Data-Flow

Soit $S \subseteq V$ (ensemble fini de variables de l'automate), $V, V' \text{ et } V''$ des jeux de valuation des variables. On dit que $V' \text{ et } V''$ sont incompatibles par rapport à V s'il existe une variable $\upsilon \in S$ telle que $V[\upsilon]$, $V'[\upsilon]$ et $V''[\upsilon]$ soient distincts. La composition des deux valuations V' et V'' par rapport à V, notée $V' \circ_\upsilon V''$ est la valuation définie par :

$$\forall \upsilon \in S, V' \circ_\upsilon V'' \overset{def}{=} \begin{cases} V'[\upsilon] & si\ V'[\upsilon] \neq V[\upsilon] \\ V''[\upsilon] & sinon \end{cases}$$

Soit $A = \langle D, S, F^{in}, F^{out}, dom, \Sigma, \delta, \sigma, I \rangle$ un automate de mode et $\vec{e}_1, ..., \vec{e}_r$ r vecteurs de synchronisation. Σ' est le sous ensemble de Σ qui contient les événements présents dans les vecteurs de synchronisation. La synchronisation de A par les $\vec{e}_i$ est un automate de mode :

$A \setminus \vec{e}_1, ..., \vec{e}_r = \langle D, S, F^{in}, F^{out}, dom, (\Sigma \setminus \Sigma') \cup \{\vec{e}_1, ..., \vec{e}_r\}, \delta', \sigma, I \rangle$ avec :

- $\forall e \in \Sigma \setminus \Sigma', s \in dom(S)$ et $I \in dom(F^{in})$ si $\delta(s, I, e)$ est définie alors $\delta'(s, I, e)$ est aussi définie et $\delta'(s, I, e) \overset{def}{=} \delta(s, I, e)$
- $\forall \vec{e}_i = \langle e_1, ..., e_k \rangle, s \in dom(S)$ et $I \in dom(F^{in}), si\ \delta(s, I, e_1), ..., \delta(s, I, e_k)$ sont définies et compatibles deux à deux, alors

$$\delta'(s, I, \vec{e}_i) \overset{def}{=} \delta(s, I, e_1) \circ_s ... \circ_s \delta(s, I, e_k)$$

En AltaRica Data-Flow, la notion de temps s'exprime par un ordonnancement des événements. Ainsi, il est possible d'exprimer des priorités en introduisant un ordre partiel sur les événements. On obtient alors un automate de mode avec priorités qui est un couple $(A, \prec)$ avec A un automate de mode et $\prec$ un ordre partiel. Bien entendu ces priorités ne priment pas sur les gardes des transitions.

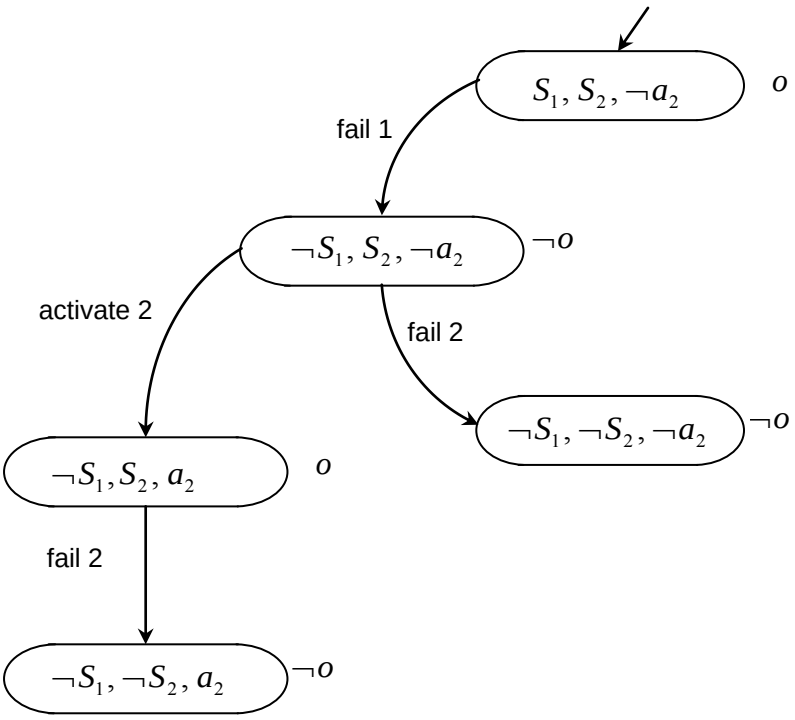


Figure B.7: Automate de mode avec priorité *fail1 > fail2*

ANNEXE C

PRINCIPALES LOIS DE PROBABILITÉ UTILISÉES À LA SÛRETÉ DE FONCTIONNEMENT

C.1 La loi exponentielle

La loi exponentielle est très fréquemment utilisée car elle est une des seules qui permette de réaliser facilement les calculs.

La densité de probabilité est :

$$f(t) = \lambda e^{-\lambda t} \quad \lambda > 0 \quad t > 0$$

Où λ est une constante et t le temps.

La fonction de répartition est :

$$F(t) = 1 - e^{-\lambda t}$$

Par ailleurs :

$$moyenne = \frac{1}{\lambda} \qquad \text{var}\,iance = \frac{1}{\lambda^2}$$

La loi exponentielle est très utilisée pour caractériser la période durant laquelle le taux de défaillance est constant ; elle décrit alors l"intervalle de temps entre deux défaillances.

C.2 La loi de Weibull

Cette loi dépend de trois paramètres. Elle est très intéressante car elle permet de représenter un grand nombre de distributions expérimentales.
La densité de probabilité est.

$$f(t) = \frac{\beta (t-\gamma)^{\beta-1}}{\sigma^{\beta}} \exp -\left(\frac{t-\gamma}{\sigma}\right)^{\beta} \quad \beta > 0, \quad \sigma > 0, \quad t > \gamma$$

La fonction de répartition est :

$$F(t) = 1 - \exp\left[-\left(\frac{t-\gamma}{\sigma}\right)^{\beta}\right]$$

Par ailleurs :

$$moyenne = \gamma + \sigma \Gamma\left(\frac{1+\beta}{\beta}\right)$$

$$\text{var}\,iance = \sigma^2 \left[\Gamma\left(\frac{2}{\beta}+1\right) - \Gamma^2\left(1+\frac{1}{\beta}\right)\right]$$

La loi exponentielle est une loi de Weibull dont les paramètres sont égaux à :

$$\beta = 1, \quad \gamma = 0, \quad \sigma = \frac{1}{\lambda}$$

C.3 La loi gamma

C'est une loi à deux paramètres λ et β. La densité de probabilité est :

$$f(t) = \frac{\lambda^{\beta} t^{\beta-1}}{\Gamma(\beta)} e^{-\lambda t} \quad t \geq 0, \quad \lambda > 0, \quad \beta > 0$$

Et $\Gamma(\beta)$ est une fonction gamma définie par l'équation suivante :

$$\Gamma(\beta) = \int_0^{\infty} x^{\beta-1} e^{-x} dx$$

Pour les valeurs entières de $\beta : \Gamma(\beta) = (\beta - 1)!$

La fonction de répartition est :

$$F(t) = \int_0^{t} \frac{\lambda^{\beta} x^{\beta-1}}{\Gamma(\beta)} e^{-\lambda x} dx$$

Par ailleurs :

$$La\ moyenne = \frac{\beta}{\lambda}; \quad la\ \text{var}iance = \frac{\beta}{\lambda^2}$$

Trois cas particuliers de la loi gamma sont à mentionner :

- $\beta = 1; \quad f(t) = \lambda e^{-\lambda t}$; on retrouve la loi exponentielle
- β est un entier *k* : on obtient la loi d'Erlang :

$$f(t) = \frac{\lambda^k t^{k-1}}{(k-1)!} e^{-\lambda t}$$

$$F(t) = 1 - e^{-\lambda t} \sum_{i=1}^{k} \frac{(\lambda t)^{i-1}}{(i-1)!}$$

Pour *k = 1*, on retrouve la loi exponentielle.

Considérons l'événement A défini par l'apparition du *k-ième* événement aléatoire, ces événements ayant un taux d'apparition constant λ. La probabilité pour que A se produise entre les instants *t* et *t + dt* est égale au produit de la probabilité de *k – 1* apparitions entre 0 et t par la probabilité d'une apparition entre *t* et *t + dt*, soit :

$$f(t)\,dt = e^{-\lambda t} \frac{(\lambda t)^{k-1}}{(k-1)!} \lambda\, dt$$

On retrouve la loi d'Erlang. *F(t)* apparaît ainsi comme la probabilité pour que la durée d'attente de *k* apparitions soit inférieure à *t*.

- $\lambda = \frac{1}{2}; \quad \beta = \frac{\upsilon}{2}$ et υ est un entier

Nous obtenons la loi de χ^2 (khi-deux) à υ degrés de liberté.

ANNEXE D

LA MÉTHODE DES ÉTATS FICTIFS (LOI D'ERLANG)

La méthode d'états fictifs est l'une des méthodes proposées pour la résolution analytique des processus non markoviens. Si l'hypothèse du taux de transition constant est souvent vérifiée pour le taux de défaillance, elle peut être fausse pour le taux de réparation. La méthode d'états fictifs permet de ramener le processus dans un contexte markovien.

Elle consiste à remplacer une transition à taux non constant entre deux états par une combinaison de transitions à taux constants entre des états fictifs. Elle permet de modéliser des transitions quelconques (durée pseudo-déterministe, taux de défaillance croissant relatif à un phénomène d'usure…etc.). Elle offre, également, la possibilité de réduire le nombre d'états d'un système en remplaçant des modèles de sous-ensembles indépendants par des modèles simplifiés équivalents.

Il a été démontré que n'importe quel type de transition peut être approximé de cette manière, et que le jeu de transitions entre états fictifs peut se limiter au modèle de Cox présenté dans la figure D.1.

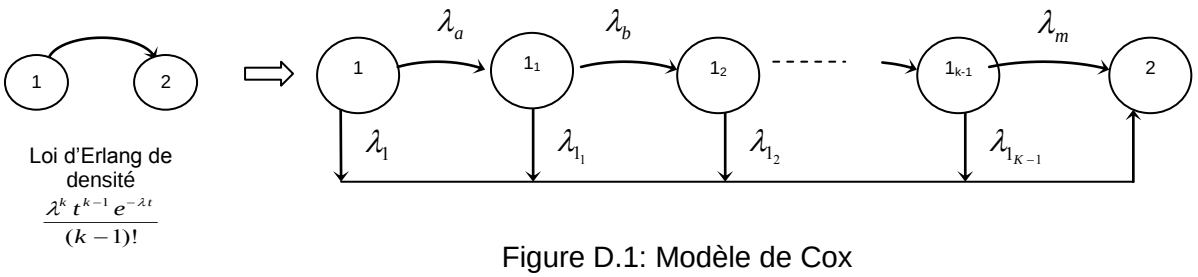


Figure D.1: Modèle de Cox

Si les taux de transition de la chaîne directe sont identiques ($\lambda_a = \lambda_b = \ldots = \lambda_m$), la loi est dite d'Erlang généralisée. Si, de plus, le jeu de transitions est limité à la chaîne directe, la loi qui en résulte est une loi d'Erlang simple de paramètre *k*, correspondant à la convolution de *k* lois exponentielles, c'est-à-dire qu'une transition (défaillance ou réparation) correspondant à une loi d'Erlang peut être décomposée en *k* transitions correspondant à une même loi exponentielle et *k-1* états fictifs (figure D.2).

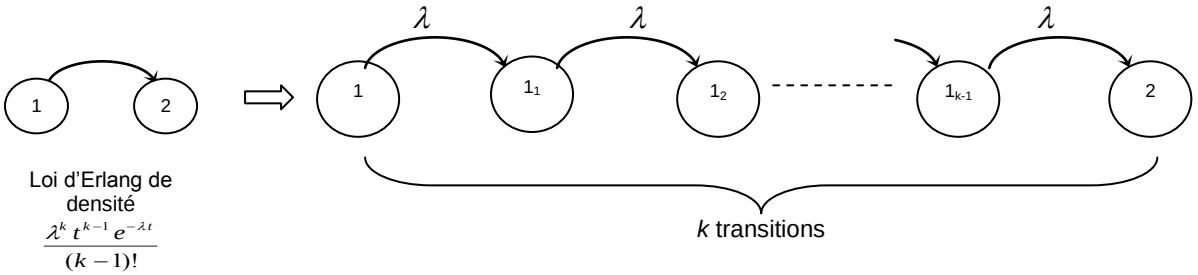


Figure D.2: Décomposition d'une transition correspondant à la loi d'Erlang en *k* transitions

Les états 1_1, 1_2,…1_{k-1} sont fictifs mais permettent de transformer le processus en un processus markovien. On peut généraliser l'utilisation des états fictifs ; tout d'abord les taux de transition peuvent être différents. On obtient alors le graphe de la figure D.3. Les états fictifs sont alors dits en série.

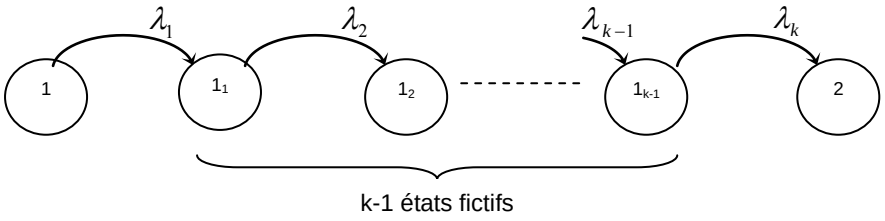


Figure D.3 : *k-1* états fictifs en série

On peut également concevoir une transition de l'état 1 vers l'état 2 qui soit telle que le temps de séjour dans l'état 1 soit distribué suivant une loi exponentielle de paramètre λ_i avec la probabilité α_i (figure D.4):

$$p(X = X_i) = \alpha_i, \quad \alpha_i > 0, \quad \sum_{i=1}^{k} \alpha_i = 1$$

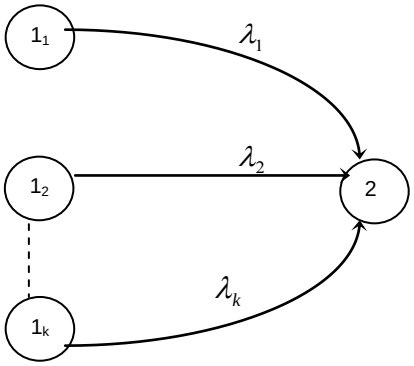


Figure D.4 : Décomposition parallèle de *k* états fictifs

Les états fictifs sont dits en parallèle. Naturellement il est possible d'utiliser simultanément les deux représentations.

Bibliographie

[ABB 01] A., Abbas-Turki, O.,Grunder, A.,El Moudni (2001). *Modélisation et évaluation d'une correspondance élémentaire d'un système de transport en commun par Réseaux de Petri Stochastiques.* 3e Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM"01 – du 25 au 27 Avril 2001 – Troyes (France).

[ABU 96] O., Abu-Amsha, J., M., Vincent (1996). *An algorithm to bound fonctionnalise of Markov chains large state space*. Grenoble: Rapport de Recherche du projet MAI, n° 25, 1996.
Anonymous ftp *ftp://ftp.image.fr/pub/MAI/Rapports/R_MAI025.ps.gz*

[AJM 84] M., Ajmone-Marsan, G., Balbo, G., Conte (1984). *A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems*. ACM Transactions on Computer systems, vol 2, n° 2, 1984, pp 93-122.

[ARN 05] A. Arnold, G. Point, A. Griffault, A. Rauzy, (2005), *AltaRica, Manuel méthodologique*, LaBRI, Université Bordeaux I et CNRS (UMR 5800), 96 pages, 26 septembre 2005.

[ATI 93] K., Atif (1993). *Modélisation du parallélisme et de la synchronisation*. Thèse en vue de l"obtention du grade de Docteur de l"INPG, Grenoble, 1993.

[AVA 04] B., Avanzi (2004). *Chaînes de Markov : Eléments de Théorie et Application au Domaine de la Finance.* Travail de séminaire. Ecole des HEC Université de Lausanne Mai 2004.

[AYE 03a] A., Ayel, X., Fontenelle, (2003). *Le Taux de Rendement Synthetique: Un indicateur, deux fonctions*, 23ième journee regionale de la productique, Amberieu en Bugey.

[AYE 03b] A., Ayel, B., Davier (2003). *Le TRS indicateur de la performance : un guide pratique à l'usage des responsables de production.* Centre Technique des Industrie Mécaniques (CETIM), 2003, 91 pages.

[AYE 04] A. Ayel (2004). *La mesure de performance des machines de production*. CETIM, TRS performance 2004.

[BAS 00] R., Bastide (2000). *Spécification comportementale par réseaux de Petri : Application aux systèmes distribués à objets et aux systèmes interactifs.* Habilitation à diriger des Recherches de l"Université Toulouse 1, spécialité : Informatique, janvier 2000. 74 pages.

[BAS 75] F., Baskett, K., M., Chandy, R., R., Muntz, F., G., Palacios (1975). *Open, closed and mixed networks of queues with different classes of customers*. Journal of the ACM, vol. 22, n° 2, 1975, pp 248-260.

[BEN 03] A., Benoit (2003). *Méthodes et algorithmes pour l'évaluation des performances des systèmes informatiques à grand espace d'états.* Thèse en vue de l"obtention du grade de Docteur de l"INPG, spécialité : Informatique : Systèmes et Communications, juin 2003. 170 pages.

[BEN 05] A., Benoit, B., Plateau, W., J., Stewart (2005). *Réseaux d'automates stochastiques à temps discret*. RSTI-TSI-24/2005. Evaluation de performance, pp 229-248.

[BIL 03] A., Billaud, P.,Ganzin, L., Gillet, M., Haziza, O., Roth (2003). *Le soutien logistique intégré : La prise en compte des besoins d'un système tout au long de son cycle de vie*. Gestion de production et Logistique : Projet - IFI 2003 – Option Génie Industriel.

[BOU 05] M., Bouissou (2005). *Dix petits problèmes de modélisation (en sûreté de fonctionnement des systèmes)*. Revue Phoebus n° 34 juillet-août-septembre 2005.

[BOU 07] M., Boutry (2007). *Construction d'indicateurs* [online]. Available from http://www.univ-nancy2.fr/Amphis/images/19/64/Gest-Qual_ConstructionIndicateurs.pdf [accessed 8 may 2008]

[BUC 94] P., Buchholtz (1994). *A class of hierarchical queueing networks and their analysis*. Queueing Systems**,** vol. 15, 1994, pp 59-80.

[CAB 98a] A., Cabarbaye (1998). *Modélisation et évaluation des systèmes* - Cours de technologies spatiales, Edition Cepadues Toulouse 1998.

[CAB 98b] A., Cabarbaye (1998). *Apports, difficultés et prospectives dans le domaine de l'évaluation quantitative en Sûreté de Fonctionnement*, Qualité Espace, supplément au N°33, septembre 1998.

[CAB 99a] A., Cabarbaye, L., Tomasini, L.,Ngom, S., Allibe (1999). *Apport des algorithmes génétiques à la Sûreté de Fonctionnement et à l'optimisation des systèmes* – Congrès Qualita 99, 25-26 mars 99, Paris.

[CAB 99b] A., Cabarbaye, L., Ngom (1999). *Mise en oeuvre de la méthode des états fictifs et génération automatique des matrices de Markov* - Congrès Qualita 99, 25-26 mars 99, Paris.

[CAB 99c] A., Cabarbaye, J., Séroi, L., Tomasini (1999). *Optimisation de la Sûreté de Fonctionnement des systèmes spatiaux* - 3e Congrès International de Génie Industriel, Montréal 26 - 28 mai 1999.

[CAB 00a] A. Cabarbaye, L. Ngom (2005), *Mise en œuvre de la méthode des états fictifs et génération automatique des matrices de Markov.* Congrès International Pluridisciplinaire de Markov.doc

[CAB 00b] A., Cabarbaye, J., Séroi (2000). *Optimisation dans le domaine de la Sûreté de Fonctionnement* 12e Colloque National de Sûreté de Fonctionnement (λ/μ 12), Montpellier 28 - 30 mars 2000.

[CAB 00c] A., Cabarbaye, L., Ngom (2000). - *Simulation dynamique des arbres d'événements* - 12e Colloque National de Sûreté de Fonctionnement (λ/μ 12), Montpellier 28 - 30 mars 2000.

[CAB 01] A., Cabarbaye (2001). *SIMCAB : un outil générique de simulation sous Microsoft Excel.* 3e Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM"01 – du 25 au 27 avril 2001 - Troyes (France).

[CAB 99] E., Cabau (1999). *Introduction à la conception de la sûreté*. Schneider Electric. Cahier technique n° 144. Edition juin 1999.

[CAO 02] Y., Cao, H., Sun (2002). *System Availability With Non-Exponentially Distributed Outages*. IEEE Transactions on Reliability, Vol. 51, n° 2, June 2002, pp 193 – 198.

[CAO 84] Z., Q., Cao, K., H., Kim, F., W., Roush (1984). *Incline algebra and applications***.** Ellis Horwood, 1984.

[CAS 08] C., G., Cassandras, S., Lafortune (2008). *Introduction to Discrete Event Systems, second edition*. ISBN-13: 978-0-387-33332-8. Springer Science & Business Media, LLC.

[CHA 01] J. L., Chabot, Y., Dutuit, A., Rauzy, (2001). *De l'usage de la simulation Monte-Carlo couplee aux reseaux de Petri en surete de fonctionnement*. 3è Conference Francophone de Modelisation et Simulation, Troyes-France, du 25 au 27 avril 2001.

[CHE 03] N.,Cheikhrouhou, R., Glardon (2003). *Optimisation de lignes de production via simulation : une approche basée sur l'analyse de perturbation*. 4e Conférence Francophone de MOdélisation et SIMulation "Organisation et Conduite d"Activités dans l"Industrie et les Services" MOSIM"03 – du 23 au 25 avril 2003 - Toulouse (France).

[CIM 04] CIMNET (2004). *Improve your Bottom Line Through Manufacturing.* 2004 CIMNET, Inc. web: www.cimnet.com

[CLE 00] J. W., Clemons, (2000). *Overall Equipment Effectiveness*, EnteGreat, Inc. Birmingham, Alabama, Mountain Conference.

[COC 92] C., Cocozza-Thivent, F. Kervegant (1992), *Quantification en Sûreté de fonctionnement par approximation des systèmes de grande taille : méthode des états fictifs.* Revue de statistiques appliquées, tome 40, n°2, pp 17-30.

[COT 99] P. N., Cotaina (1999). *Méthodologie d'aide à la décision et à la mise en place de politiques de maintenance pour les P.M.E. L'apport de la MBF (Maintenance Basée sur la Fiabilité).* Thèse, en vue de l"obtention du grade de Docteur de l"Université Henri Poincaré, Nancy 1, spécialité Automatique, soutenue le 22 octobre 1999.

[COU 77] P., J., Courtois (1977). *Decomposability: queueing and computer systems applications***.** London: Academic Press, 1977.

[COU 94] P., J., Courtois, P., Semal (1994). *Bounds for positive eigenvectors of non-negative matrices and for their approximations by decomposition.* Journal of the ACM**,** vol. 31, n° 4, pp 804-825, 1994.

[DAL 94] Y., Dallery, Z., Liu, D. Towsley (1994). *Equivalence, reversibility, symmetry and concavity properties in fork-join queueing networks with blocking*. Journal of the ACM, vol. 41, n° 5, 1994, pp 903-942.

[DAV 01] W., David (2001). *Prioritizing System-Reliability Prediction Improvements*. IEEE Transactions on Reliability, Vol. 50, n° 1, march 2001. pp 17 – 25.

[DAV 93] R., C., David (1993). *Analyse orientée objets et modélisation par simulation*. Editions Addison-Wesley, France, juillet 1993, 362 pages.

[DEC 03] D., Decotigny (2003). *Une infrastructure de simulation modulaire pour l'évaluation de performances de systèmes temps réel.* Thèse en vue de l"obtention du grade de Docteur de l"Université de Rennes. spécialité : Informatique. Avril 2003. N° d"ordre 2831, 244 pages.

[DIS 02] J. P., Dismukes (2002). *Factory level metrics : Basis for productivity improvement.* International Conference on Modelling and Analysis of Semiconductor Manufacturing (MASM), Arizona USA, 2002.

[DJE 03] M., Djebabra, L., Bendada, N., Bennoui (2003). *Capacité des réseaux de Petri pour l'analyse et la modélisation des systèmes de production.* Département Hygiène & Sécurité, Faculté des Sciences de l"Ingénieur. Université de Batna – Algérie, CPI"2003, pp 1 – 13.

[DON 93] S., Donnatelli (1993). *Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space.* North-Holland: Performance Evaluation, vol. 18, 1993, pp 21-36.

[EFA 03] E. D. Efaga, R. Danwe, F. Braun, M.-C. Suhner, Awono Onana (2003). *Analyse des données du retour d'expérience pour l'organisation de la maintenance des outils de production des PME et PMI dans le cadre de la MBF (Maintenance Basée sur la Fiabilité).* QUALITA 2003. 5ième Congrès international pluridisciplinaire Qualité et Sûreté de Fonctionnement. Actes du Congrès. Institut de Sûreté Industrielle Nancy France PP 360-367.

[FER 04] J.-L., Ferrier, J.-L., Boimond (2004). *Systèmes dynamiques à événements discrets : du modèle à la commande.* ISTIA - Université d'Angers Laboratoire d'Ingénierie des Systèmes Automatisés (LISA). Juin 2004. web : http://www.istia.univ-angers.fr/LISA.

[FER 98] P. H. L., Fernandes (1998). *Méthodes Numériques pour la Solution de Systèmes Markoviens à Grand Espace d'États.* Thèse en vue de l'obtention du grade de Docteur de l'Institut National Polytechnique de Grenoble, spécialité : Informatique. février 1998. 260 pages.

[FLO 85] G., Florin, S., Natkin, (1985). *Les réseaux de Petri stochastiques*. Techniques et Sciences Informatiques, vol 4, n°1, pp 143 – 160.

[FOU 03] M., Fouladirad, I., Nikiforov (2003). *Statistical Fault Detection in linear Systems whith nuisance parameters*. Qualita 2003-5ème congrès pluridisciplinaire Qualité et Sûreté de Fonctionnement Nancy-France pp 452-459.

[FOU 95] J., M., Fourneau, F., Quessette (1995). *Graphs and stochastic automata networks.* In: Computations with Markov chains**.** Ed: W. J. Stewart. Boston: Kluwer Academic Publishers, 1995.

[GEN 08] Géniop (2008). *Calcul du TRS-Norme NFE 60-182* [online]. Available from http://www.geniop.com [accessed 8 may 2008].

[GOU 03] R., Goullioud, N., Abouchi, R., Grisel, (2003). *Analyse des performances des machines parallèles : approche par simulation à évènements discrets*. CPE-LISA-CNRS 0092, Lyon cedex 02.

[GOS 00] K., Goseva Popstojanova (2000). *Failure Correlation in Software Reliability Models.* IEEE Transactions on Reliability, Vol. 49, n° 1, March 2000, pp 37 – 48.

[GÖT 95] N., Götz, H., Hermanns, U., Herzog, V., Mertsiotakis, M., Rettelbach (1995). *Constructive specification techniques-integrating functional performance and dependability aspects.* In: Quantitative Methods in Parallel Systems**.** Springer, 1995.

[GRO 97] E., Grolleau, A., Choquet-Geniet, F., Cottet (1997). *Ordonnancement Optimal des Systèmes de Tâches Temps Réel à l'Aide de Réseaux de Petri.* proc. AGIS, Automatique Génie informatique Image Signal, Angers, 1997 pp.239-246.

[HAC 02] L., Hacker (2002). *Critical Examination of a Common Assumption in System Availability Computations.* Reliability Edge, Volume 3, Issue 2, quarter 3, 2002.

[HEY 82] D., P., Heyman, M., J., Sobel (1982). *Stochastic models in operations research*. Vol 1, New York, McGraw-Hill, 1982.

[HUA 03] S. H., Huang, J. P., Dimukes, J., Shi, Q., Su, M. A., Razzak, R., Bodhale, D. E., Robinson, (2003). *Manufacturing Productivity Improvement Using Effectiveness Metrics and Simulation Analysis*, International Journal of Production Research, Vol. 41, No. 3, pp 513-527.

[HOU 03] C., Houlbert, A., Cabarbaye (2003). *Apport d'un logiciel générique d'optimisation et de simulation à la résolution de problématiques d'ordonnancement et à la maîtrise des risques*. 4è Conférence Francophone de Modélisation et SIMulation « Organisation et conduite d'Activités dans l'Industrie et les Services » MOSIM'03 – du 23 au 25 avril 2003 – Toulouse (France).

[INN 06] F., Innal, Y. Dutuit, (2006). *Évaluation de la performance d'un systeme de production des contributions individuelles de ses unites constitutives*. 6è Conference Francophone de Modelisation et Simulation, Rabat, Maroc, du 3 au 5 avril 2006.

[JAM 01] D., Jampi, E., Guilhem, J-F., Aubry (2001). *Conception et sûreté de fonctionnement : deux activités indissociables.* 3e Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM'01 – du 25 au 27 avril 2001 - Troyes (France).

[KAM 04] O., Kamach (2004). *Approche Multi-Modèle pour les systèmes à Evénements Discrets : application à la gestion des modes de fonctionnement.* Thèse en vue de l'obtention du grade de Docteur de l'INSA de Lyon, spécialité : Automatique Industrielle, Décembre 2004, 176 pages.

[KEH 04] C., Kehren, C., Seguin, P., Bieber, C., Castel (2004). *Analyse des exigences de sûreté d'un système électrique par model-checking*. Actes du congrès Lambda-MU 14, colloque de Maîtrise des risques et Sûreté de Fonctionnement 2004.

[KHA 97] W., Khanza (1997). *Réseaux de Petri P-Temporels : Contribution à l'étude des systèmes à événements discrets.* Thèse en vue de l'obtention du grade de Docteur. Soutenue le 7 mars 1997 à l'Université de Savoie. Spécialité EEA. 190 pages.

[KLE 75] L., Kleinrock (1975). *Queueing theory (vol. 1& 2)*. Chichester: John Wiley & Sons, 1975.

[KOM 06] T., Kombé, E. D., Efaga, B., Ndzana, E., Niel, (2006), *Efficience d'un système Bâtie sur le TRS global par poursuite du diagramme de fiabilité*. Revue internationale Afrique Science volume 2 n°2, juin 2006, pp 198 – 211.

[KOM09a] T., Kombé**,** E. Niel, L. Pietrac, A. Razy (2009), *Global Efficiency Assessment based on Component Composition of OEE Using AltaRica Data-Flow Language* 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM' 09) June 3 – 5, 2009 Moscow-Russia.

[KOM09b] T., Kombé**,** E. Niel, L. Pietrac, A. Razy (2009), *Modélisation Temporelle et Stochastique du TRS pour l'Évaluation de l'Efficience des Systèmes de Production.* 3èmes Journées Doctorales du GDR MACS (Groupement de Recherche en Modélisation, Analyse et Conduite des Systèmes Dynamiques) JDMACS 2009, Angers - France, du 17 au 18 mars 2009.

[KRO 01] H. Kromm, JC. Deschamps, G. Doumeingts (2001), *Modélisation de processus pour évaluation par niveaux de détail successifs.* 3e Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM'01 – du 25 au 27 avril 2001 - Troyes (France).

[LAB 04] K. Labadi, L. Amodeo et H. Chen, (2004), *Modélisation et analyse des performances des chaînes logistiques par les Réseaux de Petri,* Laboratoire d'Optimisation des Systèmes Industriels (LOSI) Université de Technologie de Troyes – France.

[LAB 00] P. E., Labeau (2000). *Dynamic reliability: towards an integrated platform for probabilistic risk assessment.* Reliability Engineering and System Safety 68 (2000), pp 219–254.

[LAC 03] H., Lacresse, A., Grall, I., Nikiforov (2003). *Statistical Fault Detection with nuisance parameters and nonlinearities*. Qualita 2003-5ème congrès pluridisciplinaire Qualité et Sûreté de Fonctionnement Nancy-France pp 460-467.

[LAP 96] J., C., Laprie, J., Arlat, J., P., Blanquart, A., Costes, Y., Crouzet, Y., Deswarte, J., C., Fabre, H., Guillermain, M., Kaâniche, K., Kanoun, C., Mazet, D., Powell, C., Rabéjac et P., Thévenod (1996). *Guide de la Sûreté de Fonctionnement*. ISBN 2-85428-382-1, Cépaduès-Editions, janvier 1996.

[LAS 94] P., Lascaux, R., Théodor (1994). *Analyse numérique matricielle appliquée à l'art de l'Ingénieur (vol. 1 & 2)*. Paris, Masson, 1994.

[LHO 03] M., Lhommeau (2003). *Étude de systèmes à événements discrets dans l'algèbre (*max*;+). Synthèse de correcteurs robustes dans un dioïde d'intervalles. Synthèse de correcteurs en présence de perturbations***.** Thèse en vue de l'obtention du grade de Docteur. Soutenue le 16 décembre 2003 à l'ISTIA - Université d'Angers. N° 597, 178 pages.

[LIM 99] S., Limam (1999). *Contribution à la modélisation et à la simulation des systèmes de production de services : Proposition d'une méthode basée processus, UML et réseaux de Petri objets.* Thèse en vue de l'obtention du grade de Docteur de l'Institut National Polytechnique de Grenoble, spécialité : Génie Industriel préparée au Laboratoire d'Automatique de Grenoble (LAG). Décembre 1999. 233 pages.

[LUP 06] R., Lupan, A., Kobi, C., Robledo, A., Delamarre, H., Christofol, (2006). *Modelisation et evaluation de la performance en conception.* 6è Conference Francophone de Modelisation et Simulation, Rabat - Maroc, du 3 au 5 avril 2006.

[LYO 93] P., Lyonnet, (1993). *Optimisation d'une politique de maintenance*. Technique et Documentation Lavoisier, Paris, 166 p.

[MAH 04] S., Mahadevan, (2004). *Automated Simulation Analysis of Overall Equipment Effectiveness Metric*. Master Of Science Thesis in Industrial Engineering University of Cincinnati, USA.

[MAH 97] S., Mahévas (1997). *Modèles Markoviens de grande taille : calculs de bornes*. Thèse en vue de l'obtention du grade de Docteur de l'université de Rennes I, 1997.

[MAZ 95] B., Mazigh, J., Gresser (1995). *Evaluation de la sûreté de fonctionnement des systèmes de production par les réseaux de Petri stochastiques généralisés.* Diagnostic et sûreté de fonctionnement, volume 5 - n° 3 /1995, pp 255 – 272.

[MEU 02] M., Meunier (2002). *Optimiser les processus et les ressources de production.* MES: Pilotez votre atelier en temps réel pour produire plus et mieux. Pôle Productique Rhôle-Alpes, les dossiers n° 20 avril 2002.

[MOH 06] H., Mohamed (2006). *L'utilisation conjointe des réseaux de Petri stochastiques et des processus de Markov pour la modélisation, l'analyse et l'évaluation des performances d'un système de production : Ligne d'emboutissage de l'entreprise B.A.G. Batna*. Thèse en vue de l'obtention du grade de Magister. Soutenue le 14 juin 2006 à l'Université El-Hadj Lakhdar Batna. Spécialité Génie Industriel. 168 pages.

[MOL 87] P., Moller (1987). *Notions de rang dans les dioïdes vectoriels*. In: CNRS/CNET/INRIA Seminar: Algèbres Exotiques et Systèmes à Evènements Discrets, Issy-les-Moulineaux, France, 1987.

[MOR 02] F., Morata (2002). *Simulation ''Parfaite'' de grands modèles markoviens: Implantation de la méthode d'Aliasing*. Institut d'Informatique et Mathématique Appliquées de Grenoble IMAG.

[MOR 96] P., Moreaux (1996). *Structuration des chaines de Markov des réseaux de Petri stochastiques-décomposition tensorielle et agrégation*. Thèse en vue de l'obtention du grade de Docteur de l'Université Paris Dauphine, 1996.

[MOK 97] N., L., Mokdad (1997). *Méthodes et outils pour l'évaluation des performances des réseaux informatiques*. Thèse en vue de l'obtention du grade de Docteur de l'Université de Versailles-Saint-Quentin-en-Yvelines, 1997.

[MUT 07] K. M. N. Muthiah, S. H. Huang (2007), *Overall Throughput Effectiveness (OTE) metric for factory level performance monitoring and bottleneck detection*, International Journal of Production Research, vol. 45, N° 20, 15 october 2007, pp. 4753 – 4769.

[NAI 01] A., Nait-Sidi-Moh, A.-M., Manier, A., Elmoudni (2001). *Modélisation et évaluation d'un système de transport par l'algèbre max-plus*. 3e Conférence Francophone de MOdélisation et SIMulation «Conception, Analyse et Gestion des Système Industriels» MOSIM'01- du 25 au 27 avril 2001 – Troyes (France), pp 99–104.

[NEU 81] M., F., Neuts (1981). *Matrix geometric solutions in stochastic models, an algorithmic approach*. John Hopkins University Press, 1981.

[NEU 89] M., F., Neuts (1989). *Stochastic geometric matrices of G/ M/ 1 type and their applications***.** Marcel Dekker inc., 1989.

[NIE 94] E. Niel, (1994) *De la Sécurité Opérationnelle des Systèmes de Production*, Habilitation à diriger les recherches en sciences (HDR 94 – 013), Université Claude Bernard, Lyon 1 et Institut National des Sciences Appliquées de Lyon, France.

[NOT 03] D. Noterman (2003). *Le MES dans la lignée du modèle CIM*. Pôle Productique Rhône-Alpes, Saint Etienne.

[NOW 97] G. Nowack (1997). *De l'apport des opérateurs temporels dans la modélisation dysfonctionnelle des systèmes*. Thèse N° 97 ISAL 0072 en vue de l'obtention du grade de Docteur. Laboratoire d'Automatique Industrielle de l'INSA de Lyon, France.

[OUA 01] M., Ouabiba, N., Mebarki, P., Castagna (2001). *Couplage entre des méthodes d'optimisation itératives et des modèles de simulation à événements discrets*. 3e Conférence Francophone de MOdélisation et SIMulation «Conception, Analyse et Gestion des Systèmes Industriels».

[OUN 99] F., Ounnar (1999). *Prise en compte des aspects décision dans la modélisation par réseaux de Petri des systèmes flexibles de production*. Thèse en vue de l'obtention du grade de Docteur en Automatique et

[PAG 80] A., Pagès, M., Gondran (1980). *Fiabilité des systèmes*. Editions Eyrolles, Paris 5è, 323 pages.

[PER 04] F. Peres, D. Noyes (2004), *Formulation d'un problème d'optimisation d'une stratégie de maintenance par l'analyse Markovienne*, Laboratoire Génie de Production Ecole Nationale d'Ingénieurs de Tarbes.

[PIE 05] E., Pierrein (2005). *Le TRG, un élément clé du TPM*. Caterpillar, Librapport, Creative Commons du 27-08-2005.

[PLA 97] B., Plateau, B., Ycart (1997). *Fast simulation kernel for urban traffic.* Rapport de recherche MAI/IMAG, n° 41, 1997. Anonymous ftp *ftp://ftp.imag.fr/pub/MAI/Rapports/R_MAI041.ps.gz*

[PLA 91] B., Plateau, K., Atif (1991). *Stochastic Automata networks for modelling parallel systems.* IEEE transactions on software engineering**,** vol.17, n° 10, pp 1093-1108, 1991.

[POP 73] J., Poper (1973). *La dynamique des systèmes, principes et applications*. Editions d'organisations, Paris, 1973.

[QMA 08] Q-mation, *OEE-World Class Performance Reporting* [online]. A Division of Q-mation, Inc. Available from: http://www.qmation.com/webpages/oee.htlm [Accessed 6 may 2008].

[RAK 05] E., Rakotomalala (2005). *Spécifications robustes du système de pilotage d'une fonction électronique automatique*. Thèse en vue de l'obtention du grade de Docteur. Spécialité : Automatique et Informatique Appliquée. Ecole Centrale de Nante, France, soutenue le 5 décembre 2005.

[RAU 06] A., Rauzy, (2006). *AltaRica Data-Flow Language Reference*, combava Reference Documentation, ARBoost Technologies.

[REI 80] M., Reisig, (1980). *Mean value analysis of closed multichain queueing networks*. Journal of the ACM, vol. 27, n° 2, pp 313-322.

[RIG 03] E. Rigaud, F. Guarnieri, J. Riout (2003). *Vers une méthode d'auto-diagnostic des défaillances liées aux produits et aux moyens de production à destination des PME-PMI.* QUALITA 2003. 5ième Congrès international pluridisciplinaire Qualité et Sûreté de Fonctionnement. Actes du Congrès. Institut de Sûreté Industrielle Nancy – France PP 222-230.

[ROB 90] T., G., Robertazzi (1990). *Computer networks and systems : queueing theory and performance evaluation*. New York, Springer-Verlag, 1990.

[ROC 01] Rocquencourt (2001). *Méthodes, algorithmes et logiciels pour l'automatique*. Institut National de Recherche en Informatique et en Automatique INRIA. Projet METALAU.

[SAA 95] Y., Saad (1995). *Iterative methods for sparse linear systems*. Boston, PWS publishing Company, 1995.

[SAL 06] O. Salem (2006), *Modélisation algébrique et évaluation de performance des mécanismes de gestion de la qualité de service dans les réseaux Ad Hoc.* Thèse pour obtenir le grade de Docteur de l'Université de Toulouse III, soutenue le 9 octobre 2006, discipline : Informatique.

[SAL 01] N., Salmi, M., Ioualalen (2001). *Méthode conjointe de décomposition et de calcul des bornes pour l'évaluation des performances d'un Réseau de Petri Stochastique non borné.* 3e Conférence Francophone de MOdélisation et SIMulation "Conception, Analyse et Gestion des Systèmes Industriels" MOSIM'01 – du 25 au 27 Avril 2001 – Troyes (France), pp 639 – 646.

[SAR 99] P. Sarri, (1999). *Stabilisation Optimale des systèmes à Événements Discrets à Structure Vectorielle : Application à la Sécurité Opérationnelle des systèmes de production*, Thèse n° 99 ISAL 0016 en vue de l'obtention du grade de Docteur de l'Institut National des Sciences Appliquées de Lyon – France.

[SAS 98] C. Sassine, (1998) *Intégration des politiques de maintenance dans les systèmes de production manufacturiers,* Thèse en vue de l'obtention du grade de Docteur de l'Institut National Polytechnique de Grenoble - France, Discipline : Automatique et Productique.

[SCH 04] Schneider, (2004). *La Sûreté de Fonctionnement*. Guide Technique, revue InterSection, novembre 2004.

[SCH 03] R., Schoenig, J.-F., Aubry, E., Guilhem, T., Hutinet (2003). *An example of reliability assessment by an aggregation merhod of markov graphs for hybrid systems*. Qualita 2003-5ème congrès pluridisciplinaire Qualité et Sûreté de Fonctionnement Nancy-France pp 637-644.

[SMA 03] A., Smati, K., Younsi, N., Zeraibi et N., Zemmour, (2003). *Modélisation de la disponibilité d'une chaîne de GNL sur la base d'une approche bayésienne d'estimation des indices de fiabilité*. Oil & Science and Technology-Rev. IFP, vol. 58 (2003), n° 5, pp 531-549.

[SOR 06] W. I., Soro, M., Nourelfath, D., Aït-Kadi, (2006). *Evaluation des indices de performance d'un systeme multi-etats degradable*. 6è Conference Francophone de Modelisation et Simulation, Rabat - Maroc, du 3 au 5 avril 2006.

[STE 94] W., J., Stewart (1994). *Introduction to the numerical solution of Markov chains***.** Princeton University Press, 1994.

[TOM 00] L., Tomasini, A., Cabarbaye, J., Séroi, F., Garcia (2000). *Optimisation de la maintenance d'une constellation de satellites* - 12e Colloque National de Sûreté de Fonctionnement (λ/μ 12), Montpellier 28 – 30 mars 2000.

[TOU 91] L., Toutain (1991). *SAMSON: Un simulateur pour systèmes répartis et temps réel.* Thèse en vue de l'obtention du grade de Docteur en Science de l'Université du Havre. Spécialité : Instrumentation et commande, Novembre 1991, 132 pages.

[TRI 82] K., Trivedi (1982). *Science applications Probability & statistics with reliability, queueing and computer*. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

[TUF 06] B. Tuffin (2006), *Modélisation mathématique pour la conception, l'analyse quantitative et le contrôle de la qualité de service des systèmes*. HDR présentée à l'Université de Rennes le 10 avril 2006.

[VIL 88] A., Villemeur, (1988). *Sûrete de fonctionnement des systemes industriels*, editions Eyrolles, Paris 5è, 795 pages.

[vRT 03] vTRM (2003). *OEE – World Class Performance Reporting*. vRTM.OEE R5 a division of Q-mation, Inc. web www.vRTM.com.

[VOR 08] Vorne, (2008). *Calculating OEE* [online]. Vorne Industries. Available from: http://www.oee.com/calculating_oee.html [Accessed July 28, 2008].

[WIL 06] R. M., Williamson, (2006). *Using Overall Equipment Effectiveness: the Metric and the Measures* [online]. Strategic Work Systems, Inc. Columbus NC 28722. Available from: www.swwpitcrew.com/articles/OEE 0206.pdf [Accessed 10 may 2008].

[WON 07] Wonderware, (2007). *DT Analyst 2.0: Visualiser, Analyser, Optimiser* [online]. Factory Systems. Available from: http://www.factory-syst.fr/upload/docMenuDroit/DT%20Analyst%20V2.pdf [Accessed 10 June 2008].

[YCA 97] B., Ycart (1997). *Simulation de modèles markoviens***.** Grenoble : Université Joseph Fourier, 1997. (DESS d'Ingénierie Mathématique)

[ZAY 93] J. Zaytoon (1993). *Extension de l'Analyse Fonctionnelle à l'étude de la Sécurité Opérationnelle des Systèmes Automatisés de Production*. Thèse N° 93 ISAL 0016 en vue de l'obtention du grade de Docteur. Laboratoire d'Automatique Industrielle de l'INSA de Lyon – France.

[ZWI 08] G. Zwingelstein (2008). *Sûreté de fonctionnement des systèmes industriels complexes*. Technique de l'Ingénieur numéro S 8 250.