Université de REIMS CHAMPAGNE-ARDENNE

THÈSE

Présentée pour obtenir le grade de

Docteur de l'Université de Reims Champagne-Ardenne

Discipline : Génie informatique, Automatique et Traitement du signal

Par

Noureddine MALKI

Contribution au diagnostic des Systèmes à Evénements Discrets par modèles temporels et distributions de probabilité

Soutenue publiquement le 15 juillet 2013

Membres du Jury

Rapporteurs

Pr. Pascal BERRUET	Université de Bretagne Sud
Pr. Serge DEBERNARD	Université de Valenciennes

Examinateurs

Pr. Jean-François PETIN (président du jury)	Université de Lorraine
Pr. Bernard RIERA	Université de Reims Champagne-Ardenne
Pr. Noureddine MANAMANNI	Université de Reims Champagne-Ardenne
Dr. Alexandre PHILIPPOT	Université de Reims Champagne-Ardenne

Centre de Recherche en Science et Technologie de l'Information et de la Communication
UFR Sciences Exactes et Naturelles
Moulin de la Housse
51687 REIMS cedex 2

Remerciements

Ce travail de thèse a été mené au sein de l'équipe de recherche Auto du Centre de Recherche en Sciences et Technologies de l'Information et de la Communication (CReSTIC) de l'Université de Reims Champagne-Ardenne. Je remercie le directeur du CReSTIC Monsieur le Professeur Michaël Krajecki de m'avoir accueilli au sein de son laboratoire.

Le bon déroulement de ce travail n'aurait pas été possible sans le soutien, la disponibilité et les précieux conseils de mes encadrants, Messieurs MANAMANNI et PHILIPPOT. Ils trouveront ici l'expression de ma gratitude et ma reconnaissance profondes.

Je tiens à remercier chaleureusement Monsieur Pascal BERRUET, Professeur à l'Université de Bretagne Sud, et Monsieur Serge DEBERNARD, Professeur à l'Université de Valenciennes et du Hainaut Cambrésis, pour l'honneur qu'ils m'ont fait en acceptant de rapporter ce travail de thèse.

Mes vifs remerciements s'adressent également à Monsieur Jean-François PETIN, Professeur à l'Université de Lorraine, et Monsieur Bernard RIERA, Professeur à l'Université de Reims Champagne-Ardenne, pour m'avoir fait honneur d'examiner ce travail.

Je tiens aussi à remercier tous les membres du CReSTIC et tous les doctorants pour les moments agréables que nous avons partagés au cours de ces années.

Je dédie ce travail particulièrement à ma chère mère, à qui je dois beaucoup pour l'amour et le réconfort donnés durant ces années, et d'avoir supporter ma très longue absence. Mes remerciements vont également à mes frères et sœurs, à qui je témoigne mon affection et ma profonde reconnaissance.

Enfin, mes remerciements vont à tous ceux qui de près ou de loin, d'une manière ou d'une autre, ont contribué à l'aboutissement de ce travail.

Table des matières

Introduction Générale 1

Chapitre 1 : Introduction au diagnostic des défauts et aux SEDs 6

1.1 INTRODUCTION 8
1.2 TERMINOLOGIE 9
1.2.1 Etats du système et ses signaux 9
1.2.2 Opération autour du système 11
1.2.3 Propriétés du système 12
1.2.4 Modélisation du système 13
1.3 CLASSIFICATION DES DEFAUTS 13
1.3.1 Selon le composant responsable 14
1.3.1.1 Défauts actionneurs 14
1.3.1.2 Défauts capteurs 14
1.3.1.3 Défauts du procédé 15
1.3.2 Selon la dynamique d'occurrence 15
1.3.2.1 Défauts abrupts 15
1.3.2.2 Défauts graduels 16
1.4 CLASSIFICATION DES SYSTEMES DYNAMIQUES 16
1.4.1 Systèmes Continus 17
1.4.2 Systèmes à Evénements Discrets 17
1.4.3 Systèmes Dynamiques Hybrides 18
1.5 CADRE GENERALE DU DIAGNOSTIC DE DEFAUTS 18
1.6 INTRODUCTION AUX SYSTEMES A EVENEMENTS DISCRETS 20
1.6.1 Réseaux de Petri 21
1.6.2 GRAFCET 22
1.6.3 Automates à états finis 24
1.6.3.1 Notions de langages 24
1.6.3.2 Généralités sur les automates 26
1.6.3.3 Notions de contrôlabilité et observabilité 28
1.6.3.4 Opérations sur les automates 29
1.6.4 Automates temporisés 32
1.6.4.1 Notions de base 32
1.6.4.2 Composition des automates temporisés 34
1.7 CONCLUSION 36

Chapitre 2 : Diagnostic des Systèmes à Evénements Discrets 37

2.1 INTRODUCTION 39
2.2 APPROCHES SANS ET AVEC MODELE(S) 40

2.2.1 Approches sans modèle ... *40*
2.2.1.1 Arbre de défaillances ... 41
2.2.1.2 Système expert ... 42
2.2.1.3 Approches à base de classification de données ... 42
2.2.1.4 Approches à base des Réseaux Bayésiens ... 43
2.2.2 Approches à base de modèle(s) ... *44*
2.3 REPRESENTATION DES DEFAUTS DANS LE MODELE ... 45
2.3.1 Approche avec représentation des défauts dans le modèle ... *45*
2.3.2 Approche sans représentation de défauts dans le modèle ... *46*
2.4 OUTILS DE REPRESENTATION DES APPROCHES DE DIAGNOSTIC ... 47
2.4.1 Modèles à base d'automates à états finis ... *47*
2.4.1.1 Approches à base d'événements ... 48
2.4.1.2 Approches à base d'états ... 49
2.4.2 Approches à base de Réseaux de Petri (RdP) ... *52*
2.4.3 Approches à base de Templates ... *54*
2.4.4 Approches à base de Chroniques ... *55*
2.5 STRUCTURE DU DIAGNOSTIC ... 57
2.5.1 Structure centralisée ... *57*
2.5.2 Structure décentralisée ... *59*
2.5.3 Structure distribuée ... *62*
2.5.4 Choix d'une structure ... *64*
2.6 NOTION DE DIAGNOSTICABILITE ... 66
2.7 CONCLUSION ... 67

Chapitre 3 : Diagnostic des SEDs par distributions de probabilité ... 69

3.1 INTRODUCTION ... 71
3.2 CONTEXTE ET OBJECTIFS ... 71
3.3 DESCRIPTION GENERALE DE L'APPROCHE ... 72
3.4 CONSTRUCTION DU MODULE DE DIAGNOSTIC (HORS LIGNE) ... 76
3.4.1 Décomposition du système et Définition des contraintes ... *76*
3.4.1.1 Principe de décomposition du système ... 76
3.4.1.2 Apprentissage des séquences d'événements ... 78
3.4.2 Construction des Templates et Chroniques ... *79*
3.4.3 Construction des modules de diagnostic ... *84*
3.4.3.1 Caractérisation de l'occurrence d'événements par distribution de probabilité ... 84
3.4.3.2 Construction des conditions d'autorisation et des fonctions de non-occurrence ... 86
3.5 DEPLOIEMENT DU DIAGNOSTIC (EN LIGNE) ... 90
3.5.1 Procédures de détection des défauts ... *94*
3.5.2 Procédure de localisation des défauts ... *97*
3.6 CONCLUSION ... 99

Chapitre 4 : Illustration de l'approche et résultats de simulation 101

4.1 INTRODUCTION 103
4.2 DESCRIPTION DU SYSTEME 103
4.3 CONSTRUCTION DU DIAGNOSTIQUEUR (HORS LIGNE) 104
4.3.1 Décomposition du système 105
4.3.2 Modélisation du système 106
4.3.2.1 Construction des distributions de probabilité 111
4.3.2.2 Construction des conditions d'autorisation et des fonctions de non-occurrence 112
4.4 DEPLOIEMENT DU DIAGNOSTIQUEUR (EN LIGNE) 115
4.4.1 Procédure de détection, localisation et identification des défauts 115
4.4.1.1 Défauts abrupts dans des composants de type Tout ou Rien « ToR » 116
4.4.1.2 Défauts graduels dans des composants de type analogique ou dans le processus 118
4.4.2 Décision et dialogue Homme-Machine 123
4.5 CONCLUSION 125

Conclusion Générale et Perspectives 127

Bibliographie..... 132

Introduction Générale

Introduction Générale

Contexte et objectifs

La modernisation des différents procédés technologiques, visant principalement à leur automatisation, augmente d'une façon incessante durant ces dernières années. De ce fait, ces procédés deviennent de plus en plus complexes et sophistiqués. Cependant, cette complexité implique des besoins croissants en termes de fiabilité, disponibilité et sûreté de fonctionnement. La complexité de telles installations technologiques peut les conduire dans des états de dysfonctionnement ne permettant plus de réaliser leurs tâches correctement. Afin de pouvoir éviter ce genre de situation et d'améliorer les performances de ces installations, l'élaboration d'un module de diagnostic est désormais nécessaire. Ce module va permettre d'envisager des actions correctives (maintenance, reconfiguration de la commande…etc.) pour que le système retourne à son état de fonctionnement nominal. La nécessité du module de diagnostic n'est pas liée seulement à l'amélioration des performances et de la productivité des systèmes, mais également à la limitation des conséquences des défauts pouvant être catastrophiques ; que ce soit au niveau des biens ou des personnes (Figure 1).

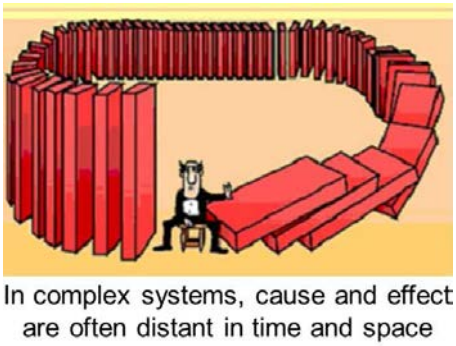


Figure 1. Représentation de la conséquence d'un défaut.

Le problème du diagnostic de défauts dans les systèmes automatisés complexes a reçu une attention considérable par le monde scientifique. En effet, une littérature bien riche, que ce soit théorique ou pratique, concernant ce domaine a été développée pour les systèmes complexes et particulièrement ceux ayant des modèles à événements discrets. Les Systèmes à Evénements Discrets (SEDs) occupent plusieurs domaines d'application incluant les chaînes

de production manufacturière, la robotique, les systèmes véhiculaires, la logistique, les réseaux de communication…etc. Les SEDs sont définis comme des systèmes dynamiques asynchrones dont l'espace d'états peut être décrit par un ensemble dénombrable [Cassandras et Lafortune, 2008]. Leur évolution se fait selon l'occurrence d'événements qui se produisent de manière instantanée à des points isolés dans le temps. Cette occurrence caractérise généralement un changement d'état du système.

Les outils ainsi que les techniques utilisées pour atteindre l'objectif du diagnostic changent d'une communauté à une autre. Ils dépendent souvent du système à diagnostiquer. Par conséquent, la diversité des approches proposées au long de ces dernières années rend la problématique de diagnostic de défauts également complexe. En général, un module de diagnostic de défauts est construit afin de réaliser trois tâches consécutives : la détection, la localisation et l'identification. La première tâche consiste à percevoir si le système fonctionne correctement ou si un défaut a eu lieu dans ce dernier. La tâche de localisation vise à déterminer la partie défectueuse du système en représentant des candidats responsables de leurs occurrences. Finalement, la tâche d'identification consiste à déterminer le type de défaut en éliminant ou minimisant au maximum le nombre de candidats qui ont été fournis dans la tâche de localisation mais en fournissant également des caractéristiques à ce défaut (amplitude, gravité, date d'apparition, …). Cet ensemble de tâches permet alors de choisir ou proposer à l'opérateur un type d'action à mettre en œuvre (maintenance corrective ou préventive, reconfiguration de la commande…etc.).

L'objectif de cette thèse est de proposer une approche de diagnostic de défauts pour les Systèmes à Evénements Discrets. Pour atteindre cet objectif, il faut tout d'abord décomposer le système global que l'on souhaite diagnostiquer en un ensemble de sous-systèmes indépendants au niveau événementiel et géographique. Par la suite, chacun de ces sous-systèmes est modélisé par des modèles temporels appartenant au formalisme des automates temporisés et représentant un comportement normal ou défaillant. Ces modèles sont construits hors ligne et par apprentissage en se basant sur les signaux d'entrées/sorties du système. Cette information temporelle est représentée par des intervalles de temps traduit par la suite en distributions de probabilité. Ces distributions peuvent être utilisées pour confirmer le fonctionnement normal ou la détection d'un fonctionnement anormale du système.

Organisation du mémoire

Ce mémoire de doctorat est structuré en quatre chapitres et organisé comme suit :

Dans le **premier chapitre**, une présentation de la terminologie utilisée pour le diagnostic de défauts est présentée. Dans ce contexte, une classification des défauts qui peuvent avoir lieu dans un système est également faite. Dans un deuxième temps, nous présentons de manière générale les différentes classes de systèmes dynamiques en s'intéressant particulièrement aux systèmes à évènements discrets. Ainsi, nous évoquons quelques notions relatives aux langages formels et quelques outils de représentation pour la modélisation des SEDs tels que les automates à états finis par exemple.

Le **deuxième chapitre** est consacré à un état de l'art non exhaustif des méthodes de diagnostic des SEDs rencontrées dans la littérature. Ces différentes méthodes sont représentés au travers une classification qui a été faite selon plusieurs critères (approches sans et avec modèle, selon la représentation des défauts dans le modèle, structure du diagnostic…etc.). L'état de l'art présenté dans ce chapitre va nous permettre de positionner notre travail de recherche par rapport aux approches présentées.

Le **troisième chapitre** constitue notre contribution principale. Il est consacré à la mise en œuvre d'un module de diagnostic en deux phases. La première phase correspond à la construction hors ligne du module de diagnostic. Elle est réalisée en trois étapes :

- décomposition du système en sous-systèmes indépendants,
- construction des *Templates* et *Chroniques* et
- construction des diagnostiqueurs.

La deuxième phase vise au déploiement en ligne du module de diagnostic. Cette phase est effectuée également en trois étapes qui sont :

- l'implémentation des modules de diagnostic,
- la détection de défauts et
- la localisation (voire identification) de ceux-ci.

Dans le **quatrième chapitre,** les concepts théoriques développés dans ce mémoire sont illustrés au travers un benchmark simulé afin de pouvoir aussi bien représenter un comportement normal du système que générer des défauts. Le cas d'étude et les différents scénarios de validation de l'approche sont simulés sous l'environnement *Matlab/Simulink* afin d'en évaluer sa pertinence.

Une conclusion sur ce travail et des perspectives de recherche et d'application marquent la fin de ce mémoire.

Chapitre 1

Introduction au diagnostic des défauts et aux Systèmes à Evénements Discrets

1.1 INTRODUCTION 8
1.2 TERMINOLOGIE 9
1.2.1 Etats du système et ses signaux 9
1.2.2 Opération autour du système 11
1.2.3 Propriétés du système 12
1.2.4 Modélisation du système 13
1.3 CLASSIFICATION DES DEFAUTS 13
1.3.1 Selon le composant responsable 14
1.3.1.1 Défauts actionneurs 14
1.3.1.2 Défauts capteurs 14
1.3.1.3 Défauts du procédé 15
1.3.2 Selon la dynamique d'occurrence 15
1.3.2.1 Défauts abrupts 15
1.3.2.2 Défauts graduels 16
1.4 CLASSIFICATION DES SYSTEMES DYNAMIQUES 16
1.4.1 Systèmes Continus 17
1.4.2 Systèmes à Evénements Discrets 17
1.4.3 Systèmes Dynamiques Hybrides 18
1.5 CADRE GENERALE DU DIAGNOSTIC DE DEFAUTS 18
1.6 INTRODUCTION AUX SYSTEMES A EVENEMENTS DISCRETS 20
1.6.1 Réseaux de Petri 21
1.6.2 GRAFCET 22
1.6.3 Automates à états finis 24
1.6.3.1 Notions de langages 24
1.6.3.2 Généralités sur les automates 26
1.6.3.3 Notions de contrôlabilité et observabilité 28
1.6.3.4 Opérations sur les automates 29
1.6.4 Automates temporisés 32
1.6.4.1 Notions de base 32
1.6.4.2 Composition des automates temporisés 34
1.7 CONCLUSION 35

Chapitre 1 : Introduction au diagnostic des défauts et aux SEDs

1.1 Introduction

Notre société moderne dépend fortement des procédés technologiques complexes. La fiabilité, la disponibilité et la sûreté de fonctionnement des équipements sont ainsi devenues de véritables défis pour les entreprises actuelles. La complexité de systèmes technologiques peut conduire à l'occurrence de défauts. Un défaut est une grandeur qui change le comportement d'une partie du processus de telle sorte que cette partie ne remplit plus son fonctionnement [Blanke et al., 2003]. Dans les systèmes complexes, les effets d'un défaut peuvent rapidement se propager en conduisant à la dégradation de leurs performances ou à leurs défaillances. Cela signifie que les défauts doivent être détectés le plus tôt possible, et des décisions doivent être prises pour arrêter la propagation de leurs effets et, par conséquent, minimiser la dégradation du système. Afin de pouvoir atteindre cet objectif, il est nécessaire de mettre en place un module de diagnostic qui serait sensible à l'occurrence de défauts qui devrait être intégré au système [Meera et al., 1996] [Genc et Lafortune., 2003]. Le but du diagnostic est de déterminer si un défaut s'est produit dans le système ou non, cette tâche est appelée « *détection de défauts* », et de définir le type et l'emplacement de ce défaut, cette tâche est appelée « *localisation de défauts* ».

La première partie de ce chapitre est consacrée à la présentation de la terminologie utilisée pour le diagnostic de défauts et qui sera adoptée par la suite dans ce mémoire. Dans la deuxième partie, une classification non exhaustive des défauts auxquels les systèmes sont sujets est présentée. Cette classification a été faite selon plusieurs critères. Ensuite, nous présenterons les différentes classes des systèmes dynamiques en se focalisant sur les Systèmes à Evénements Discrets (SEDs). Avant d'aller plus loin dans l'étude de ces systèmes, nous allons introduire le diagnostic de défauts dans son cadre générale ainsi que les phases principales qui le constituent. Finalement, Nous présentons les différentes définitions et notions de base concernant les SEDs ainsi que quelques outils utilisés pour leur modélisation. Nous jugeons que la présentation de ces outils est nécessaire pour une meilleure appréhension des travaux présentés dans l'ensemble du mémoire de thèse.

1.2 Terminologie

Avant d'aborder le sujet de thèse portant essentiellement sur le diagnostic des Systèmes à Evénements Discrets « SEDs », nous allons tout d'abord présenter et éclaircir quelques définitions concernant la terminologie qui sera adoptée par la suite dans le présent manuscrit. La terminologie autour du diagnostic n'est pas souvent la même selon le domaine d'étude. Par exemple en médecine, le diagnostic est la démarche par laquelle le médecin, généraliste ou spécialiste va déterminer, à partir de symptômes (effets) et/ou d'étiologie (causes), l'affection dont souffre un patient afin de proposer un traitement. Son diagnostic est réalisé grâce à son expérience et l'étude académique des différentes maladies ou affections. Il est très souvent réalisé en deux phases : (i) l'historique du patient et de la maladie, (ii) puis l'examen physique avec des observations globales (auscultation, palpation, température, tension, …) et parfois plus précises selon le cas (analyse de sang, d'urine, …). Il est facilement compréhensible que l'ensemble des termes de diagnostic utilisés en médecine, bien que dans la philosophie de réflexion très proche, ne pourront pas se prêter directement au diagnostic immobilier (évaluation énergétique, sanitaire d'un bâtiment, …).

Dans le cadre des travaux autour des systèmes manufacturiers à tendance discrète, il existe également différentes définitions dans la littérature [Riascos et al, 2008] [Philippot et al, 2005].

En 2009, le conseil scientifique de l'association Diag21 (www.diag21.com/), créée à l'initiative d'un groupe d'industriels de l'Aéronautique et de Défense impliqués dans les problématiques du diagnostic de systèmes complexes, a proposé un ensemble de terminologie française couramment utilisé en diagnostic [Ly et al., 2009]. Par ailleurs, le comité technique SAFEPROCESS de l'IFAC (International Federation of Automatic Control) a également présenté une liste comportant différentes définitions [Isermann et Ballé, 1997]. Dans la suite de ce rapport, nous nous baserons sur ces deux comités afin de faire une synthèse de ces définitions regroupées selon plusieurs critères que nous allons voir.

1.2.1 Etats du système et ses signaux

Des attributs externes au système peuvent avoir des grands impacts sur son état de fonctionnement en lui apportant des changements qui influencent, d'une façon négative, ses performances en le mettant dans un état de dysfonctionnement. Ces changements peuvent

avoir des effets soit sur son état interne ou soit sur les interactions entre ses différents composants en modifiant certains de ses signaux.

Anomalie: Déviation par rapport à ce qui est attendu. Une anomalie justifie une investigation qui peut déboucher sur la constatation d'une non-conformité ou d'un défaut.

Défaut: un changement inattendu d'un paramètre caractéristique réel d'une unité fonctionnelle surveillée par rapport au paramètre caractéristique habituel. L'écart est considéré comme défaut lorsqu'il dépasse les limites d'acceptabilité. Un défaut dans le système n'affecte pas en général l'aptitude du système à accomplir une fonction requise. Par conséquent, on peut constater que le défaut ne conduit pas toujours à une défaillance. De ce fait, le défaut est vu comme une opinion sur le bon fonctionnement. Par contre une défaillance résulte systématiquement d'un défaut.

Erreur: Changement d'état d'une unité fonctionnelle pouvant entraîner une défaillance. Une erreur est effective quand une procédure d'exploitation fait intervenir la faute qui en est la cause.

Faute: Action, volontaire ou non, dont le résultat est la non prise en compte correcte d'une directive, d'une contrainte exprimée par le cahier des charges. On peut distinguer deux classes de faute : les fautes physiques qui sont à l'origine de perturbations internes ou externes au système, et les fautes d'ordre humain, dues à l'imperfection humaine. Les fautes physiques peuvent être soit de conception liées au cycle de vie d'un produit depuis sa phase de spécification jusqu'à sa phase d'exploitation, soit d'interaction liées à l'exploitation du système (demande de services inadéquats ou inexistants, violation de fonctionnement normal du système).

Défaillance: Altération ou cessation de l'aptitude d'une unité fonctionnelle à accomplir une fonction requise. Après défaillance d'une unité fonctionnelle, celle-ci est en état de panne. La défaillance est un passage d'une unité fonctionnelle d'un état de fonctionnement normal à un état de fonctionnement anormal ou de panne.

Panne: C'est l'incapacité d'une unité fonctionnelle à accomplir une fonction requise ou à assurer le service approprié à la suite d'une défaillance. L'occurrence d'une panne dans un système peut provoquer son arrêt complet. Une panne est généralement la conséquence d'une

défaillance ; néanmoins, elle peut exister sans défaillance préalable. C'est l'ensemble des défaillances des composants. La cause supposée d'une panne est un défaut physique ou une erreur humaine.

Symptôme: Événement ou ensemble de données au travers desquels le système de détection identifie le passage dans le fonctionnement anormal. Il s'agit d'un effet qui est la conséquence d'un comportement anormal. Un symptôme est le seul élément dont a connaissance le système de surveillance au moment de la détection de la défaillance.

Acquisition de données: Collecte des informations concernant le fonctionnement actuel du système. Elle peut être ponctuelle, périodique ou événementielle.

Bruit: Toute entrée inconnue et incontrôlable qui peut avoir lieu dans le système.

1.2.2 Opération autour du système

Une unité fonctionnelle lorsqu'elle est dans son mode opérationnel son fonctionnement peut être subi des changements suite aux différents facteurs externes (anomalie, défaut, défaillance…). Ces facteurs peuvent impliquer des graves conséquences sur l'état du système. En effet, plusieurs opérations autour du système seront envisageables (Figure 1.1) afin de savoir quel type de ces facteurs est le responsable de ces changements. Par conséquence, selon ces connaissances acquises nous pouvons agir d'une façon qui nous permet de garantir la sûreté du système ainsi que la sécurité des personnels.

Diagnostic: Consiste à déceler l'apparition d'un aléa et rechercher ses causes et ses conséquences. La première phase dans ce processus consiste à détecter l'occurrence d'une anomalie. Dans une deuxième phase, explicative, elle consiste à localiser cette anomalie et à en rechercher les causes. Enfin, la troisième phase, plus prospective, consiste à analyser les conséquences de l'anomalie sur le système global afin, par exemple, d'éviter les déclenchements d'alarmes en cascade ou de déterminer l'état réel dans lequel se trouve le système.

Maintenance: Ensemble des actions techniques et administratives correspondantes, y compris les opérations de surveillance et de contrôle, destinée à maintenir (maintenance préventive) ou à rétablir (maintenance corrective) une unité fonctionnelle dans un état spécifié ou dans des

conditions données de sûreté de fonctionnement (disponibilité, fiabilité, maintenabilité et sécurité) lui permettant d'accomplir une fonction requise.

Surveillance: Ensemble des moyens mis en œuvre (opérations manuelles ou automatiques, étapes, fonctions et mécanismes) destinées à observer l'état d'une unité fonctionnelle (en ligne, en temps réel) dans le but de faire face aux aléas d'un système au cours de la phase d'exploitation.

Pronostic : Consiste à déterminer la durée de vie restant sous contrainte temporelle. Il s'agit d'anticiper l'apparition des anomalies sur des horizons de temps de prédiction étendus. Le terme "horizon de temps de prédiction" signifie l'intervalle de temps s'étendant de l'instant où une prédiction est réalisée jusqu'à l'instant d'apparition d'une anomalie.

Reconfiguration : Consiste à réorganiser les lois de commande de manière à amener le système dans un état le plus proche possible de celui dans lequel il se trouvait en fonctionnement normal.

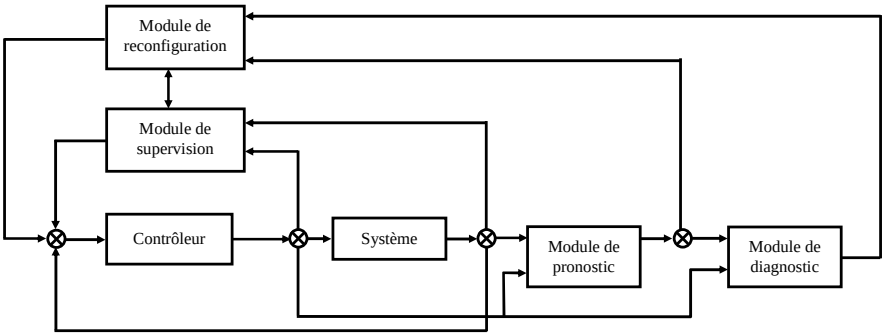


Figure 1.1. Schéma général des modules assurant la sûreté du système.

1.2.3 Propriétés du système

Un système doit vérifier certaines propriétés afin de garantir une bonne productivité et remplir sa fonction dans la présence des exigences posées par un cahier de charge. En fait, Parmi ces propriétés, que le système est contraint d'avoir, nous citons : la maintenabilité, la disponibilité et la fiabilité.

Maintenabilité: Aptitude d'une unité fonctionnelle à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise lorsque la maintenance est accomplie dans des conditions données avec des procédures et des moyens prescrits.

Disponibilité: Aptitude d'une unité fonctionnelle à être en état d'accomplir une fonction requise dans des conditions données et à un instant donné, en supposant que la fourniture des moyens extérieurs nécessaires sont assurés. C'est une grandeur qui intègre la fiabilité et la maintenabilité. Elle exprime la probabilité pour que le système accomplisse sa fonction, donc qu'il soit exempté de fautes, à l'instant *t*, sachant qu'il a pu en receler auparavant.

Fiabilité: Aptitude d'une unité fonctionnelle à accomplir une fonction requise ou à satisfaire les besoins des utilisateurs, dans des conditions données, pendant une durée donnée.

1.2.4 Modélisation du système

Un système est un ensemble déterminé d'éléments (ou composants) interconnectés, en interaction ou coopérant entre eux. Ces éléments représentent une unité fonctionnelle élémentaire considérée comme indivisible. Afin de représenter le comportement normal, ou anormal, du système, une étape de modélisation est souvent nécessaire à travers sa globalité ou un ensemble de ces composants.

Modèle quantitatif : Il décrit le comportement et les relations reliant les variables et les paramètres d'un système sous forme analytique tels que les équations différentielles.

Modèle qualitatif : Ce modèle permet d'abstraire le comportement du procédé avec un certain degré d'abstraction à travers des modèles non pas mathématiques mais des modèles de type symbolique/abstrait (automates à états, réseaux de Petri, …).

1.3 Classification des défauts

Dans le contexte du diagnostic, un défaut est considéré comme tout événement qui peut agir sur différentes parties d'un système complexe, et qui influence d'une façon inacceptable la performance globale de ce dernier. L'occurrence de défauts dans un système peut être très coûteuse en termes de production, disponibilité des équipements et de coût économique et humain. Dans de nombreux cas, un système peut passer par un état « dégradé » pendant une certaine durée avant qu'il tombe dans un état « défaillant ». Dans la littérature, les défauts sont classés selon plusieurs critères [Isermann, 2005]:

- Composant responsable,
- Dynamique d'occurrence.

1.3.1 Selon le composant responsable

En prenant comme critère de classification « le composant responsable de l'occurrence de défauts », on peut distinguer trois catégories principales de défauts (Figure 1.2).

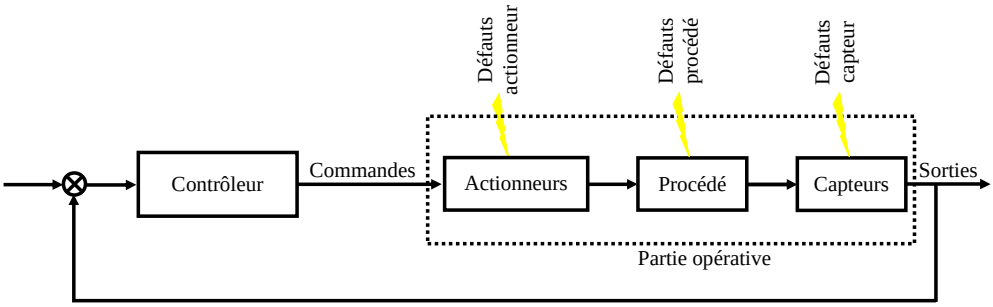


Figure 1.2. Classification des défauts selon le composant responsable.

1.3.1.1 Défauts actionneurs

Ces défauts représentent une perte partielle ou totale de l'actionneur. Une perte totale (défaillance) d'un actionneur peut se produire, par exemple, suite à un bris de l'actionneur lui-même, suite à un blocage dû à la présence d'un corps extérieur ou à cause d'un câble coupé, etc. Ce genre de défauts peut avoir comme conséquence la non-réaction de l'actionneur à une commande envoyée par le contrôleur ou une réaction non attendue. Une perte partielle (dégradation) d'un actionneur conduit celui-ci dans un mode de fonctionnement dégradé. Elle peut avoir lieu, par exemple, suite à une fuite hydraulique ou pneumatique, une résistance accrue ou une chute de tension d'alimentation, etc.

1.3.1.2 Défauts capteurs

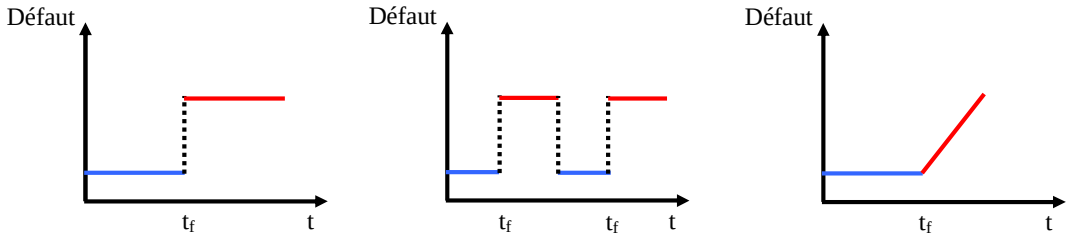
Ils représentent les lectures incorrectes des capteurs dans un système. Ils peuvent être aussi subdivisés en défauts partiels et défauts totaux. Un défaut partiel d'un capteur peut se traduire par son passage dans un mode de fonctionnement dégradé. Dans ce mode le capteur fournit des valeurs inexactes ou inattendues pour une grandeur physique. Un défaut total d'un capteur conduit ce dernier dans un mode de fonctionnement défaillant. Ce mode de fonctionnement est représenté soit par l'incapacité du capteur à fournir une information concernant la grandeur physique à mesurer soit par des informations erronées. Cela peut être dû, par exemple, à cause d'une perte de sensibilité du capteur, suite à une augmentation d'un signal de bruit ou à cause de faux contacts.

1.3.1.3 Défauts du procédé

Ce sont des défauts qui peuvent avoir lieu dans le procédé lui-même, partie opérative hors capteurs et actionneur, et donc qui n'appartiennent pas aux deux premières catégories. Ils représentent des changements dans la dynamique du système, comme la présence d'une fuite dans un système hydraulique ou la perte de puissance d'un système électrique, etc. Due à leur diversité, les défauts procédé couvrent une classe très large des situations, et sont considérés souvent comme les plus difficiles à traiter.

1.3.2 Selon la dynamique d'occurrence

Cette caractéristique concerne essentiellement le paramètre temporel de l'apparition d'un défaut. En effet, les défauts peuvent être aussi classifiés en deux principales catégories [Gertler, 1998] (Figure 1.3).



a-1) Défauts permanents a-2) Défauts intermittents

a) Défauts abrupts b) Défauts graduels permanents

Figure 1.3. Classification des défauts selon les caractéristiques temporelles.

1.3.2.1 Défauts abrupts

Ce type de défauts sont souvent dus à un dommage matériel et se caractérisent par un changement instantané de l'état normal vers un état défaillant. Habituellement, ils sont très critiques car ils affectent directement la performance et la fiabilité du système. Les défauts abrupts peuvent être soit permanents ou soit intermittents.

Les défauts abrupts permanents sont caractérisés par leur présence persistante dans le temps après leur apparition. De ce fait, suite à ce type de défaut, il n'est pas possible de revenir en fonctionnement normal sans intervention sur le composant défectueux.

Pour les défauts intermittents, ils apparaissent et disparaissent d'une façon aléatoire et surtout rapidement. Ils passent d'un état normal à défaillant plusieurs fois dans les deux sens. Ils peuvent être la conséquence de mauvais contacts, câbles abimés, d'usure du capteur lui-même, etc.

1.3.2.2 Défauts graduels

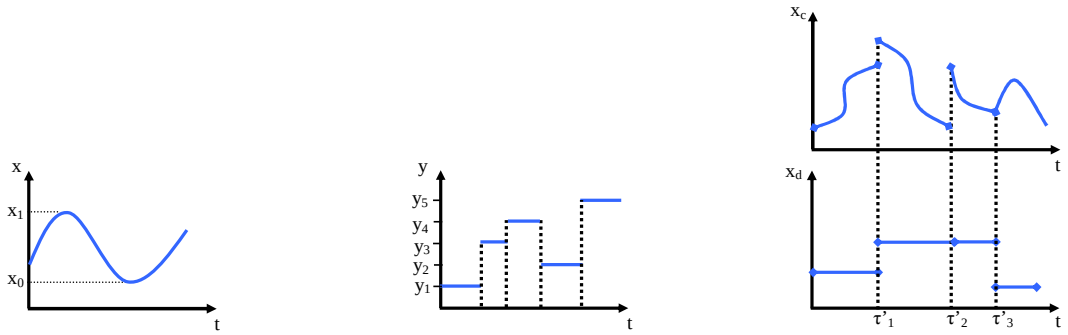
Ce type de défauts a un comportement temporel lent (dérive) et permanent dans le temps. Ils causent des changements dans les paramètres du système. Ils sont souvent difficiles à détecter car leur évolution temporelle peut avoir la même signature que celle d'une modification paramétrique lente représentant une non-stationnarité du procédé. Ces défauts peuvent avoir lieu dans le système suite au vieillissement, à l'usure de composants, à une fuite de tuyauterie, etc.

1.4 Classification des Systèmes dynamiques

Un système dynamique consiste en un ensemble d'états possibles, avec une loi qui détermine de façon unique l'état présent du système en fonction de ses états passés [Alligood et al., 1997]. Ces systèmes sont caractérisés particulièrement par la nature de leurs variables d'état. En conséquent, on peut distinguer deux types différents de variables d'état (Figure.1.4):

- Les variables d'état continues : Elles sont définies sur un ensemble de valeurs réelles comme par exemple le temps, une température, une pression, un niveau de liquide dans un réservoir, etc.
- Les variables d'état discrètes : Elles prennent leurs valeurs dans un ensemble dénombrable telles que l'ensemble des entiers ou des valeurs booléennes. On cite comme exemples pour ce type de variables d'état : L'état d'un actionneur ou capteur tout ou rien, le nombre de pièces dans un stock, etc.

Selon le type de variables d'état, les systèmes dynamiques peuvent être classés en trois catégories : (i) les systèmes continus (SC) qui nécessitent uniquement des variables d'état continues pour leur modélisation, (ii) les systèmes à événements discrets (SEDs) dont la modélisation nécessite des variables d'état discrètes, et enfin (iii) les systèmes dynamiques hybrides (SDH) qui font appel simultanément à des variables d'état continues et des variables d'état discrètes afin de pouvoir les modéliser.



a) Espace d'états continus b) Espace d'états discrets c) Espace d'états hybrides

Figure 1.4. Représentation de l'espace d'états selon la dynamique.

1.4.1 Systèmes Continus

Ces systèmes traitent des entités continues comme la température, la pression, le débit, etc [Cohen et Coon, 1953] [Brauer et Nohel, 1969] [Branicky, 1994] [Chow et Fang, 1994]. La modélisation de l'évolution dynamique de ces systèmes en fonction du temps est représentée par des modèles mathématiques tels que des équations récurrentes, une fonction de transfert, des équations d'état, etc. De manière générale, la modélisation de tel système se fait par des équations différentielles de la forme :

$$\dot{x} = f(x) \qquad (1.1)$$

Le comportement du système continu est caractérisé par la solution de l'équation différentielle (1.1) à partir d'un état initial $x(t_0)= x_0$.

1.4.2 Systèmes à Evénements Discrets

Un système à événements discrets (SED) est un système dynamique où l'espace d'état est un ensemble discret et l'état change seulement à certains instants du temps et de façon instantanée [Cassandras et Lafortune, 2008]. Les SEDs englobent une grande variété de systèmes modernes. Il s'agit notamment des systèmes manufacturiers, de transport, logistique (tel que la distribution et le stockage de marchandises, la prestation de services), de communication, etc. Dans sa forme de base, ils sont décrits par un ensemble d'états liés entre eux par des transitions auxquelles des événements sont associés. Dans les systèmes réels, les événements correspondent aux actions (par exemple : mise en marche/arrêt d'un moteur électrique…) qui peuvent changer l'état du système. Ces actions valident une transition liant deux états (par exemple : état de marche et état d'arrêt du moteur électrique…). Cette forme

de description est généralement basée sur l'observation de la dynamique interne du système. Alternativement, un SED peut être décrit par des prédicats logiques sur un ensemble de variables d'état. Dans ce cas, un événement correspond à un changement d'une ou plusieurs variables d'état. Par conséquent, ce changement conduit à une transition du système d'un état à un autre.

Une étude plus détaillée concernant ces systèmes sera présentée dans la section 1.6.

1.4.3 Systèmes Dynamiques Hybrides

Pendant plusieurs années, la communauté automaticienne a traité séparément les systèmes continus et les SEDs. Chacune de ces deux communautés ayant ses propres théories, méthodes et outils pour résoudre les problèmes qui se posent à eux. Cependant, il est parfois nécessaire de représenter/traiter simultanément les aspects continus et discrets d'un système. L'ensemble des variables d'état de cette nouvelle classe de systèmes dynamiques peut contenir des variables événementielles et continues simultanément et dans ce cas, leur interaction détermine des comportements qualitatifs et quantitatifs. Les systèmes dynamiques de cette nouvelle catégorie sont appelés *systèmes dynamiques hybrides* [Zaytoon, 2001] [Antsaklis et al, 1999].

1.5 Cadre générale du diagnostic de défauts

Cette partie est consacrée à l'introduction de quelques notions de base concernant le diagnostic de défauts dans les systèmes dynamiques en générale. Dans ces systèmes, les effets d'un défaut peuvent rapidement se propager et conduire à une dégradation dans les performances du système ou pire encore, entrainant des dégâts irrémédiables à cause d'une défaillance. Par conséquent, les défauts doivent être détectés et localisés aussi vite que possible et les décisions concernant leur criticité doivent être prises afin d'éviter la défaillance complète du système. Ce but peut être atteint en arrêtant la propagation des défauts et en minimisant des dégradations dans les performances du système. Pour prendre les bonnes décisions, il ne suffit pas seulement de savoir qu'un défaut s'est produit, mais il est également nécessaire de savoir sur quel composant il se trouve, quel est son type, mais aussi d'identifier l'élément responsable de son occurrence. Le dispositif qui nous permet de connaître ces informations est appelé « *module de diagnostic* ».

Diagnostiquer un système dynamique complexe consiste à trouver les défauts qui ont eu lieu dans ce dernier. Pour atteindre cet objectif une certaine compréhension de la structure interne du système ainsi que le niveau d'indépendance, s'il existe, entre ses différents composants est exigée. Il est aussi nécessaire de connaître les conséquences d'occurrences de défauts dans le système. Si un défaut survient, le système change souvent de comportement car il n'est plus en mesure d'effectuer toutes ses fonctionnalités désignées dans le cahier des charges. Toutes ces informations sont exprimées à travers un modèle représentant le fonctionnement normal voir défaillant du système. Toutefois, la modélisation des systèmes complexes, tels que les systèmes manufacturiers, de télécommunications…, et la détermination des relations liant les différentes observations et les états de fonctionnement qui lui correspondent sont devenues de plus en plus difficile puisque les relations sont souvent masquées. Construire, et avec précision, des modèles pour des systèmes dynamiques complexes est un objectif qui n'est pas toujours atteignable dû à la complexité des procédés eux-mêmes et aussi aux interactions qui peuvent être présents entre leurs différents composants.

Le diagnostic de défauts peut se faire en trois phases principales : *détection*, *localisation* et *identification de défauts* (Figure. 1.5).

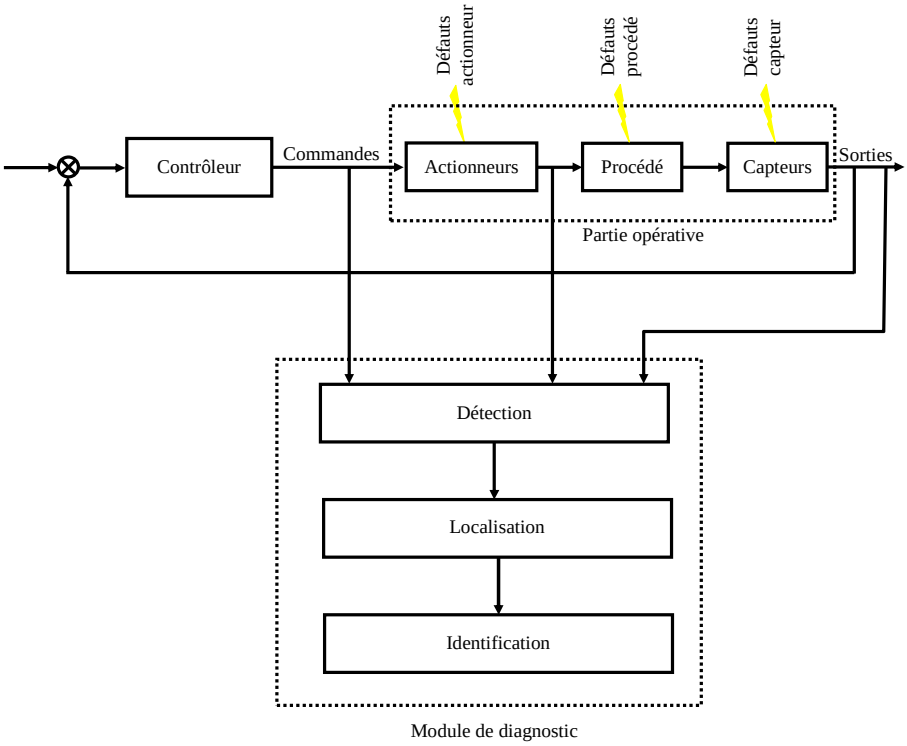


Figure 1.5. Phases principales de diagnostic.

Détection : cette étape consiste à découvrir l'apparition de toute anomalie de fonctionnement dans le système surveillé et de déterminer l'instant de son occurrence.

Localisation : cette phase vient après la phase de détection de défauts. Étant donné qu'un défaut est survenu dans le système, sa localisation se fait en se basant sur l'écart détecté entre le comportement prédit par le modèle du système et celui observé afin de fournir des composants candidats dans lesquels le défaut a eu lieu. Dans cette étape, l'objectif est de déterminer un seul candidat en utilisant les observations disponibles. Malheureusement, cet objectif n'est pas toujours possible à atteindre. Le but sera alors de minimiser l'ensemble des composants candidats à ce défaut.

Identification : c'est la dernière phase du diagnostic. Elle consiste à déterminer le type et la nature des défauts qui ont eu lieu dans le système. Apres avoir récupéré les informations nécessaires concernant les défauts, cette phase vise également à connaitre, avec exactitude, la gravité des défauts et leurs impacts sur les performances et la fiabilité du système à diagnostiquer.

Dans le cadre de nos travaux développés dans le chapitre 2, nous nous focaliserons sur le diagnostic des défauts autour des systèmes à événements discrets. Par conséquent, nous allons présenter rapidement certaines notions et utilisations de ces systèmes dans le paragraphe suivant.

1.6 Introduction aux Systèmes à Evénements Discrets

Ces systèmes ont été représentés d'une façon introductive dans la partie « 1.4.2 ». Une étude plus détaillée leur est consacrée dans cette section. Un système à événements discrets est décrit à travers son comportement observé. Ceci est représenté formellement par l'ensemble de toutes les séquences d'événements conduisant le système à changer son état au fil du temps. Ces séquences d'événements définissent le *langage généré* par le système.

En général, les propriétés du comportement d'un SED semblent plus intuitives pour les représenter et les analyser dans le cadre d'un langage formel. Ceci est basé sur le fait que le comportement d'un SED est défini comme le langage généré par son modèle représenté via un outil de modélisation (automates à états finis, réseaux de Petri, Statecharts ….).

Les SEDs comportent des modules de contrôle par lesquels des modifications peuvent être engendrées sur le comportement de certaines parties du système. En se basant sur les

caractéristiques physiques du système, les événements peuvent être considérés comme contrôlables, activé et désactivé par le contrôleur, ou incontrôlables dans le cas contraire. Une théorie du contrôle d'une classe générale de systèmes à événements discrets a été initiée par Ramadge et Wonham en 1987 [Ramadge et Wonham, 1987].

Plusieurs outils de modélisation des SEDs ont été proposés afin de pouvoir étudier et analyser ces systèmes convenablement dans divers contextes. Dans cette section, nous abordons quelques outils utilisés pour la représentation des SEDs qui sont les automates à états finis, les réseaux de Petri, le Grafcet, …. Une étude plus détaillée est porté aux automates à états finis suite à leurs liens forts avec notre travail.

1.6.1 Réseaux de Petri

Les réseaux de Petri (RdPs) sont un outil de modélisation pour les systèmes à événements discrets qui a été développé, dans les années 60, par Carl Adam Petri dans sa thèse de doctorat intitulée « Communication avec Automates » [Petri, 1962]. Les RdPs sont considérés parmi les outils les plus utilisés pour représenter et analyser les SEDs [René et Alla, 1992].

Définition 1.1. Un réseau de Petri est un graphique constitué de places connectées entre elles via des transitions. Formellement, un RdP est défini par un 5-uplet $N = (P, T, Pré, Post, m_0)$ avec :

- $P = \{P_1, P_2, \ldots, P_m\}$ est l'ensemble de places,
- $T = \{T_1, T_2, \ldots, T_n\}$ est l'ensemble des transitions,
- Pré : $P \times T \rightarrow \aleph$ est la matrice d'incidence avant,
- Post : $P \times T \rightarrow \aleph$ est la matrice d'incidence arrière,
    - C=Post-Pré est la matrice d'incidence.
    - où $\aleph$ est l'ensemble des entiers non négatifs,
- $m_0 : P \rightarrow \aleph$ est le marquage initial du réseau.

Le marquage

Pour représenter la dynamique d'un système modélisé par un réseau de Petri, il est nécessaire de compléter le RdP par un marquage (Figure 1.6).

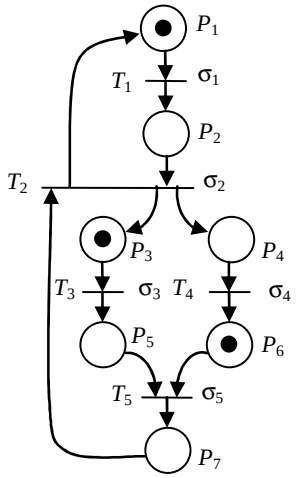


Figure 1.6. Réseau de Petri marqué.

Définition 1.2. Le marquage d'un RdP consiste à disposer un nombre entier (positif ou nul) de marques ou jetons dans chacune de ses place. Le nombre de marque contenu dans une place P_i sera noté $M(P_i)$ soit m_i.

Pour l'exemple considéré figure 1.6, on a $m_1 = m_3 = m_6 = 1$ et $m_2 = m_4 = m_5 = m_7 = 0$. Le marquage initial du réseau, M_0, est défini par le vecteur de ses marquages initiaux correspondant à chaque place. En conséquence, le marquage initial du RdP de la figure 1.6 est donc : $M_0^T = [1, 0, 1, 0, 0, 1, 0]$. Le marquage représente l'état du système à un certain instant. L'évolution de l'état correspond donc à une évolution du marquage, évolution qui se produit par le franchissement d'une transition T_i.

1.6.2 GRAFCET

Le GRAFCET est un outil de synthèse de modélisation du comportement des systèmes logiques dans l'optique de réaliser une partie commande. En 1975, une « commission de normalisation de la représentation du cahier des charges d'un automatisme logique » (animée par Michel Blanchard) [Blanchard, 1979] est créée au sein de l'AFCET (Association Française pour la Cybernétique Economique et Technique) ; après deux ans de travail, le rapport de la commission définit le Grafcet. Le groupe EPA (Equipements de Production Automatisée) de l'ADEPA (Agence pour le DÉveloppement de la Productique Appliquée à l'industrie) reprend les travaux de la commission de normalisation, son travail de mise en forme permet la parution de la norme NF C03-190 en 1982.

En 1988, le Grafcet est l'objet d'une norme internationale : CEI/IEC 848 [CEI/IEC 848, 1988]. L'année 1993 voit la publication de deux documents importants mais souvent mal utilisés :

• le projet de norme UTE C03-191 [UTE C03-191, 1993] rédigé à partir des travaux du groupe « GRAFCET » de l'AFCET,

• la norme CEI/IEC 1131 [CEI/IEC 1131-3, 1993] concernant les automates programmables industriels, et en particulier la partie 3 définissant les langages de programmation.

Depuis 1998, la norme CEI/IEC 60848 était en cours de révision, la seconde édition de la norme est maintenant parue (02/2002). Cet outil a été créé pour faire face à la complexité croissante des automatismes logiques. Il permet la représentation du cahier des charges d'un système en palliant les inconvénients des différentes méthodes existantes (principalement la lourdeur).Un Grafcet est représenté par un graphe qui comporte deux types de nœuds, les étapes et les transitions. Des arcs (ou liaisons) orientés relient soit une étape à une transition, soit une transition à une étape.

Une étape dans le Grafcet joue un rôle similaire à une place dans le RdP. Une étape peut prendre trois état différents : active, inactive ou étape initiale. Ces ensembles d'étapes sont représentés dans la figure 1.7.

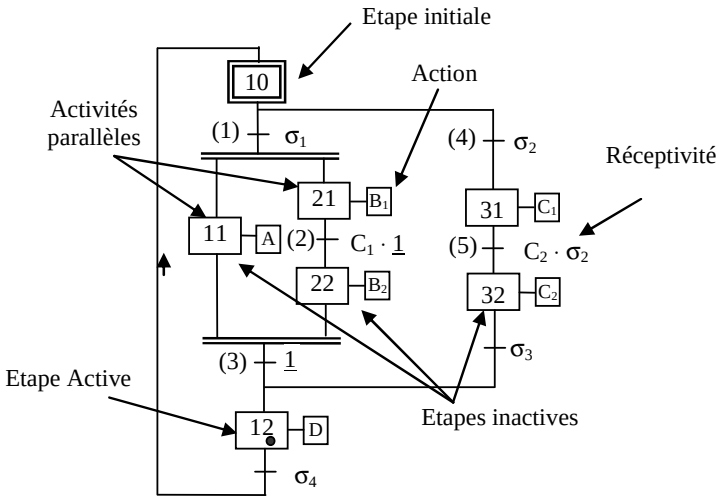


Figure 1.7. Exemple d'un Grafcet.

L'évolution d'un Grafcet s'appuie essentiellement sur des règles de syntaxe et d'évolution que le lecteur pourra retrouver en annexe 1, tout comme la description des différents éléments graphiques de celui-ci.

1.6.3 Automates à états finis

Un automate est une machine à états qui permet de décrire un système à événements discrets. Un automate est représenté par une succession d'états et de transitions associées à des événements. De façon générale, un automate est constitué des entrées et des sorties discrètes et qui changeant ses sorties suite à une modification survenu à ses entrées [Cassandras et Lafortune, 2008] [Shallit, 2008].

Avant de représenter formellement les automates à états finis, nous allons d'abord introduire une notion de base qui nous sera utile dans la définition des automates. Cette notion est appelée *langage*.

1.6.3.1 Notions de langages

L'évolution d'un SED peut être décrite par un ensemble de couples (*e*, *t*) où « *e* » représente un événement et « *t* » représente l'instant d'occurrence de cet événement. L'évolution de l'état d'un système peut être définie par les couples : (d, t_1), (f, t_2), (d, t_3), (t, t_4), (r, t_5),…etc. Cet ensemble ordonné de couples constitue ce que l'on appelle une séquence. Une telle description se place à un niveau temporel dans le sens où l'instant d'occurrence des événements est une information considérée comme pertinente. En revanche, si l'on considère un modèle logique pour décrire le SED, seul l'ordre d'occurrence des événements importe. Dans ce contexte, le fonctionnement du système est décrit par la séquence des événements : *dfdtr*….etc. L'évolution d'un SED sera en général décrite par un ensemble de séquences d'événements. Cet ensemble de séquences constitue un *Langage* sur l'ensemble des événements possibles dans notre système.

Pour formaliser des SEDs sous forme de langage, on représente les événements par des symboles. L'ensemble de tous ces symboles (σ_1 , …, σ_n) est fini et constitue un alphabet noté Σ. Toutes les séquences finies d'événements ou traces, peuvent alors être représentées par une séquence de symboles $s = \sigma_1 . \sigma_2 \ldots$ appelée chaîne (ou mot) sur l'alphabet Σ [Cassandras et Lafortune, 2008].

Définition 1.3. On appelle *alphabet* (ou vocabulaire), un ensemble fini de symboles noté Σ. Dans le cas d'un SED, l'alphabet pourra représenter l'ensemble des événements possibles dans le système. Cet ensemble est composé de tous les événements qui font évoluer le SED.

Définition 1.4. Un *mot* (ou *chaîne* ou *séquence*) noté *s* défini sur un alphabet Σ est une suite finie d'éléments de Σ. La longueur d'un mot *s*, notée |*s*|, représente le nombre de symboles de *s* (par exemple, |*dfr*| = 3).

Etant donné un alphabet Σ, on notera Σ^n où *n* est un entier naturel, l'ensemble des mots sur Σ de longueur *n*. Par exemple, pour l'alphabet Σ= {*d*, *f*}, nous obtenons : Σ^0 = { ε } où ε représente le mot vide (qui ne comporte aucun symbole) ; Σ^1 = {*d*, *f*} et Σ^2 = { *dd*, *df*, *fd*, *ff* }, ….etc. Par extension, on définira par Σ^* l'ensemble des mots d'une longueur *n* quelconque que l'on peut construire sur Σ. Ainsi Σ^* peut être défini par :

$$\sum^* = \bigcup_{i \geq 0} \sum^i \tag{1.2}$$

Où ∪ représente le symbole d'union. Etant donné l'alphabet Σ={*d*, *f*}, nous obtenons alors :

$$\Sigma^* = \{\varepsilon, d, f, dd, df, fd, ff, ddd, \dots\} \tag{1.3}$$

Définition 1.5. On appelle un langage défini sur un alphabet Σ, tout sous-ensemble de Σ^*.

Par exemple l'ensemble des séquences telles que « *d* » apparaît en premier et que « *d* » et « *f* » apparaissent alternativement est décrit par le langage : L = {ε, d, df, dfd, dfdf,… }. On notera par Φ, le langage vide.

Nous pouvons à présent définir le *préfixe-clôture* d'un langage *L*, comme le langage contenant tous les préfixes des chaînes de *L*. Nous noterons par $\overline{L}$, le préfixe-clôture du langage *L*.

$$\overline{L} = \{s_1 \in \Sigma^* \mid \exists\, s_2 \in \Sigma^*, s_1 s_2 \in L\} \tag{1.4}$$

Plus d'informations sur les langages pourra être retrouvé en annexe 2.

1.6.3.2 Généralités sur les automates

Un automate à états fini déterministe *A*, peut être défini de la façon suivante [Shallit, 2008]:

Définition 1.6. Un automate fini *A* est un 5-tuplet $A = (Q, \Sigma, \delta, q_0, Q_m)$ où :

- Q est l'ensemble fini des états qui constitue l'espace discret des états,
- Σ est un alphabet fini décrivant l'ensemble des événements,
- δ est la fonction de transition d'état $\delta : Q \times \Sigma \rightarrow Q$,
- q_0 est l'état initial représenté par une flèche,
- $Q_m \subseteq Q$ est l'ensemble des états marqués qui définissent les états finaux.

Un automate peut être représenté par son *graphe de transitions d'états*. Le graphe de transition d'états d'un automate à états fini *A* est donné dans la figure 1.8.

Exemple 1. 1. Considérons un exemple d'un client entrant dans un magasin. On peut modéliser le comportement de ce client par un SED et le représenter via un automate à états finis (Figure 1.8). Le client peut entrer dans le magasin, choisir quelque chose à acheter, payer en espèces ou par carte de crédit. Le client peut quitter le magasin à tout moment sans faire un achat.

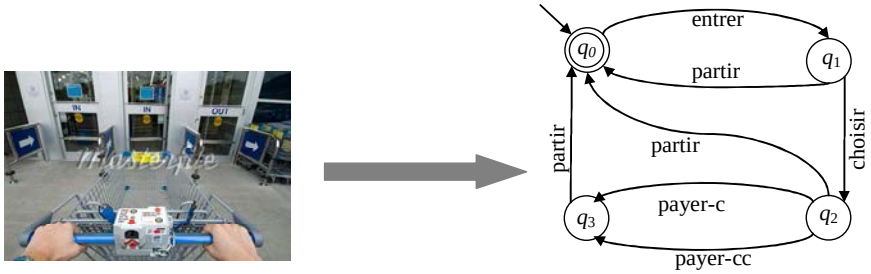


Figure 1.8. Graphe de transition d'états d'un automate fini A représentant le comportement d'un client dans un magasin.

Dans la figure 1. 8 les états de *A* sont représentés par des cercles, nous avons : $Q = \{q_0, q_1, q_2, q_3\}$. L'état initial q_0 est indiqué par une flèche entrante. Les états finaux sont représentés par des doubles cercles, ainsi : $Q_m = \{q_0\}$. La fonction δ de transition d'états est représentée par des arcs associés à des événements de l'alphabet Σ. Dans notre exemple, l'alphabet Σ correspond à l'ensemble {*entrer*, *choisir*, *payer-c*, *payer-cc*, *partir*}. Il existe une transition d'états associée à l'événement *entrer* liant q_0 et q_1. Cela signifie : $\delta(q_0, entrer) = q_1$.

Définition 1.7. L'automate *A* est dit *déterministe* dans le sens où depuis tout état, il n'existe pas deux transitions de sortie qui soient associées à un même événement et qui conduisent à deux états différents.

Définition 1.8. Un état $q \in Q$ est dit accessible s'il existe une chaîne $s \in \Sigma^*$ telle que $q = \delta(q_0, s)$, c'est-à-dire que l'automate peut y accéder depuis l'état initial. Par extension, l'automate *A* est accessible si tout état $q \in Q$ est accessible.

Les automates à états sont basés sur des notions de langages d'événements, représentés ci-dessus, et permettant la manipulation d'algorithmes mathématiques.

Définition 1.9. Le langage généré par un automate fini *A* est définie comme étant l'ensemble de toutes les séquences possibles d'événements dans le système. Il est donné formellement par:

$$L(A) = \{s \in \Sigma^* | \delta(q_0, s) \in Q\} \quad (1.5)$$

Définition 1.10. Le langage marqué est définie comme étant l'ensemble de toutes les séquences d'événements qui conduisent à un état final. Le langage est dit marqué par *A* et est noté $L_m(A)$. Un langage marqué est défini formellement comme suivant :

$$L_m(A) = \{s \in L(A) \mid \delta(q_0, s) \in Q_m\} \quad (1.6)$$

Le langage L(*A*) peut être considéré comme le comportement sans restriction d'un SED et $L_m(A)$ comme les séquences d'événements qui accomplissent une tâche, aussi appelées chaînes marquées.

Dans l'automate fini de la figure 1.8, on peut prendre, comme exemple, les séquences d'événements "entrer, partir" ou "entrer, choisir, payer-cc". En effet, La seconde séquence n'est pas complète alors elle n'appartient pas à $L_m(A)$. Cependant, elle appartient à $\overline{L_m}(A)$, car il est un préfixe de la séquence "entrer, choisir, payer-cc, partir", qui se trouve dans $L_m(A)$.

Définition 1.11. Une chaîne *t* est appelé un préfixe de la chaîne *s*, donnée par $t \leq s$, si $\exists u \in \Sigma^*$ et $s = tu$. La chaîne vide ε est un préfixe de toutes les chaînes.

On appelle un automate à états finis *non-bloquant* si tous les chaîne préfixes qui peut être éventuellement générer conduisent à un état final (marqué). Cela veut dire que L(*A*) =

$\overline{L_m}(A)$. La propriété de *non-blocage* est importante parce que, quand elle n'est pas satisfaite, le SED peut avoir un "bug" lors de son fonctionnement, c'est à dire atteindre un état à partir duquel un état final (marqué) n'est pas accessible.

1.6.3.3 Notions de contrôlabilité et observabilité

Dans l'étude des systèmes à événements discrets, l'une des questions les plus intéressantes à se poser est de savoir si un système est observable ou non et comment le contrôler/superviser. Parmi les premiers travaux qui ont été fait afin de répondre à cette question ceux de Ramadge et Wonham [Ramadge et Wonham, 1989] ont consistés à modéliser le système par des automates à états finis et proposer différentes notions de contrôlabilité et observabilité.

Définition 1.12. L'observabilité d'un système, modélisé par un automate à états finis en tant que SED, est basée principalement sur le fait de partitionner, a priori, l'ensemble de ses événements Σ en deux sous-ensembles: celui des événements observables Σ_o, ce sont, généralement, les lectures faites par des capteurs du système, et celui des événements non observables Σ_{uo} qui englobent justement les événements externes au système (bruit, défaut …). Ainsi, ce qu'on observe d'un système, modélisé en tant SED, sont des séquences d'événements observables.

Définition 1.13. La notion de contrôlabilité consiste à deviser l'ensemble des événements observables Σ_o du système en deux sous ensembles disjoints $\Sigma_c \subseteq \Sigma_o$ et $\Sigma_{uc} \subseteq \Sigma_o$, représentants respectivement l'ensemble des événements contrôlables, et l'ensemble des événements non contrôlables. Les événements contrôlables (Σ_c) correspondent aux événements générés par la partie commande vers le système. Par contre, l'ensemble des événements non contrôlables (Σ_{uc}) correspond aux événements envoyés par les capteurs du système vers le contrôleur. De ce fait, le contrôleur ne peut exercer aucune influence directe sur cet ensemble d'événements.

L'ensemble des événements Σ dans un système à événements discrets peut donc être illustré par la figure 1.9 où $\Sigma = \Sigma_o \cup \Sigma_{uo}$ et $\Sigma_o = \Sigma_c \cup \Sigma_{uc}$.

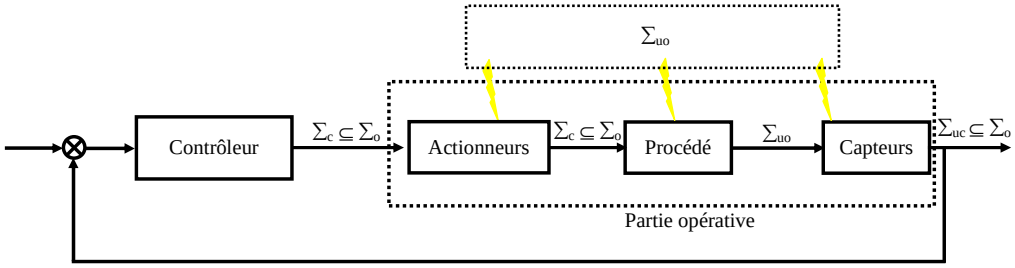


Figure 1.9. L'ensemble d'événements $\Sigma = \Sigma_o \cup \Sigma_{uo}$ dans un SED.

Des spécifications peuvent être imposées dans un cahier de charge afin de pouvoir obtenir un comportement désiré d'un système à événements discrets modélisé par un automate à états finis *A*. Ces spécifications sont définies comme étant un langage $K \subseteq L(A)$. La restriction d'un système vers un comportement désiré peut se faire en désactivant les événements contrôlables comme par exemple dans les travaux autour de la *Supervisory Control Theory* définit par Ramadge et Wonham dans [Ramadge et Wonham, 1987]. Ceci peut être formalisé par la construction d'un automate superviseur $S = (Q_s, \Sigma_s, \delta_s, q_{0s}, Q_{ms})$, sachant que $K = L(S)$. Par conséquent, le comportement contrôlé du système est désigné par $L(S/A)$ et peut être obtenu par l'intersection des deux langages $L(A)$ et $L(S)$ tel que: $L(S/A) = L(S) \cap L(A)$.

1.6.3.4 Opérations sur les automates

Dans de nombreux cas, un système à événements discrets est souvent composé de modules indépendants qui interagissent entre eux. En conséquence, pour construire un modèle global représentant le système complet, il est nécessaire de composer les modèles représentant les différents modules élémentaires de ce système. De la même façon, il est parfois nécessaire d'effectuer l'opération inverse afin de définir une projection d'un comportement à travers un ensemble d'événements que l'on souhaite observer. Nous allons dans cette section présenter succinctement quelques opérations. Des exemples seront disponibles au lecteur en annexe 2.

- ***Composition synchrone***

Cette composition est effectuée quand les alphabets des langages associés aux automates considérés ont au moins un événement en commun. Plus formellement :

Soit deux automates A_1 et A_2 définis respectivement par : $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01})$, $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02})$.

La composition synchrone entre A_1 et A_2 est défini comme étant l'automate $A_1 \parallel A_2 = (Q, \Sigma, \delta, q_0)$ tel que :

- $Q = Q_1 \times Q_2$ est l'ensemble des états.
- $\Sigma = \Sigma_1 \cup \Sigma_2$ représente l'alphabet.
- $q_0 = (q_{01}, q_{02})$ est l'état initial.

Pour tout $q = (q_1, q_2) \in Q$ et pour tout $\sigma \in \Sigma$ la fonction de transition δ est définie de la façon suivante :

- Pour $\sigma \notin \Sigma_1 \cap \Sigma_2$:

$$\delta((q_1, q_2), \sigma) = (q'_1, q_2) \text{ si } \delta_1(q_1, \sigma) = q'_1 \text{ et } \sigma \in \Sigma_1 \quad (1.7)$$

$$\delta((q_1, q_2), \sigma) = (q_1, q'_2) \text{ si } \delta_2(q_2, \sigma) = q'_2 \text{ et } \sigma \in \Sigma_2 \quad (1.8)$$

- Pour $\sigma \in \Sigma_1 \cap \Sigma_2$:

$$\delta((q_1, q_2), \sigma) = (q'_1, q'_2) \text{ si } \delta_1(q_1, \sigma) = q'_1 \text{ et } \delta_2(q_2, \sigma) = q'_2 \quad (1.9)$$

Si un événement σ, appartient à $\Sigma_1 \cap \Sigma_2$, il doit se produire de manière synchrone dans les deux automates, par contre, s'il n'appartient qu'à un ensemble, il se produit de manière asynchrone.

Remarque : Il existe également une composition synchrone dite « totale » ou « complète » où l'évolution ne peut se faire uniquement sur les événements communs. Cette opération, généralement notée "$A_1 \times A_2$", débouche sur des situations de blocage lorsque l'on ne peut plus évoluer sur des événements communs à partir de chaque état des procédés.

☛ ***Composition asynchrone***

Cette composition est réalisée quand les alphabets des langages associés aux automates considérés n'ont aucun événement en commun.

Soit deux automates A_1 et A_2 définis respectivement par : $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01})$, $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02})$. La composition asynchrone entre A_1 et A_2 est défini comme étant l'automate $A_1 \parallel A_2 = (Q, \Sigma, \delta, q_0)$ tel que :

- $Q = Q_1 \times Q_2$ est l'ensemble des états.
- $\Sigma = \Sigma_1 \cup \Sigma_2$ représente l'alphabet.
- $q_0 = (q_{01}, q_{02})$ est l'état initial.

Pour tout $q = (q_1, q_2) \in Q$ et pour tout $\sigma \in \sum$ la fonction de transition δ est définie de la façon suivante :

$$\delta((q_1, q_2), \sigma) = (q_1', q_2) \text{ si } \delta_1(q_1, \sigma) = q_1' \text{ pour } \sigma \in \Sigma_1 \quad (1.10)$$

$$\delta((q_1, q_2), \sigma) = (q_1, q_2') \text{ si } \delta_2(q_2, \sigma) = q_2' \text{ pour } \sigma \in \Sigma_2 \quad (1.11)$$

La composition asynchrone ne représente qu'un cas particulier de la composition synchrone. On pourra également remarquer que si l'on définit *m* le nombre d'états associé à l'automate A_1 et *n* le nombre d'états associés à l'automate A_2, alors la composition asynchrone de $A_1||A_2$ possède au final $m \times n$ états.

- ***Projection naturelle***

Apres avoir introduire la notion d'événements observables dans un système à événements discrets, nous allons présenter une des opérations les plus importante sur les automates à états finis modélisant un SED et qui se base particulièrement sur cette notion. L'opération est appelée projection naturelle (ou *mask*).

Afin de définir formellement cette opération, nous introduisons dans un premier temps une opération utile sur les langages : la projection dite naturelle sur un sous-ensemble d'événements observables. Soient Σ' et Σ deux alphabets tels que $\Sigma' \subseteq \Sigma$, alors $P_{\Sigma'} : \Sigma^* \rightarrow \Sigma'^*$ la projection naturelle de Σ' sur Σ est définie par :

- $P_{\Sigma'}(\varepsilon)=\varepsilon$,
- $P_{\Sigma'}(\sigma)=\sigma$ si $\sigma \in \Sigma'$,
- $P_{\Sigma'}(\sigma)=\varepsilon$ si $\sigma \in \Sigma \backslash \Sigma'$,
- $P_{\Sigma'}(s\sigma)= P_{\Sigma'}(s)\, P_{\Sigma'}(\sigma)$ pour $s \in \Sigma^*$ et $\sigma \in \Sigma^*$,

La notion de projection sur Σ' peut être étendue à un langage L tel que :

$$P_{\Sigma'}(L)= \{s \in \Sigma'^* | \exists\, s' \in L, P_{\Sigma'}(s')=s\} \quad (1.12)$$

L'opération $P_{\Sigma'}$ consiste donc à enlever d'une séquence donnée d'un langage, tous les événements n'appartenant pas à Σ'. De manière duale, étant donnes deux alphabets Σ' et Σ, qu'un langage $L \subseteq \Sigma'^* \subseteq \Sigma^*$, la *projection inverse* de Σ' dans Σ, notée $P_{\Sigma'}^{-1}$ est définie par :

$$P_{\Sigma'}^{-1}(L) = \{s \in \Sigma^* | \exists\, P_{\Sigma'}(s) \in L\} \quad (1.13)$$

Il existe de nombreuses variantes et extensions de définition des automates à états finis (*Finite State Machine*). Qu'ils soient de Moore, de Mealy, déterministes ou non, …ils représentent des outils de modélisation puissants permettant de représenter des systèmes à événements discrets sans tenir compte de l'écoulement du temps entre les événements. Dans ces automates, on ne peut pas spécifier le temps d'exécution d'une action ou le temps séparant l'arrivée d'un événement et la fin de l'action qui réagit à cet événement. Pourtant dans certains systèmes l'utilité des actions dépend de l'écoulement du temps. En conséquence, différentes propositions ont été faites afin de prendre en compte des contraintes de temps pour représenter les systèmes à événements discrets d'une meilleure façon. Les automates temporisés que nous allons maintenant décrire sont parmi les outils qu'ont été proposés pour satisfaire ses besoins.

1.6.4 Automates temporisés

Ces automates ont été introduits au début des années 90 par Rajeev Alur et David Dill [Alur, 1991] [Dill, 1989]. Ces machines représentent des automates classiques munis d'un ensemble des variables particulières appelées horloges qui évoluent d'une manière continue avec le temps et qui mesurent ainsi les délais séparant les différentes actions du système modélisé. Une garde (sur la valeur des horloges) est associe à chaque transition. Cette garde indique l'instant d'exécution de la transition dont laquelle est associe. Lors de franchissement d'une transition, un ensemble d'horloges est remise à zéro. Dans chaque état de l'automate se trouve une contrainte sur les horloges, appelée *invariant*, qui définie le temps de séjour dans cet état.

1.6.4.1 Notions de base

On considère un ensemble d'horloges X. Une valuation de X est une fonction « $v : X \rightarrow \Re_+$ » qui associe à chaque horloge x sa valeur $v(x)$. L'ensemble des valuations est noté $\Re_+^X$. On note par « $v+t$ » la valuation qui associe à tout horloge x la valeur $v(x) + t$, tels que $v \in \Re_+^X$ est un valuation et $t \in \Re_+$.

Soit $Y \subseteq X$, $v[Y \rightarrow 0]$ représente la valuation v' définie par : $v'(x)=0$ pour les horloges de Y et $v'(x)=v(x)$ aux autres horloges $x \in X \backslash Y$.

L'ensemble des contraintes d'horloges sur X, noté X(X), l'ensemble des combinaisons booléennes de contraintes atomiques de la forme $x<>c$ avec $x \in X$, $<> \in \{=, <, \leq, >, \geq\}$ et $c \in \mathbb{N}$ ($\mathbb{N}$: domaine de temps discret). Les contraintes d'horloges s'interprètent de manière naturelle sur les valuations d'horloges : une contrainte atomique $x<>c$ est satisfaite par une valuation v lorsque $v(x)<> c$. La satisfaction d'une contrainte g par une valuation v est noté $v \models g$.

Apres avoir représenté quelques notations qui nous seront utile par la suite dans cette section, on va, maintenant, introduire formellement la définition d'un automate d'états temporisé.

Définition 1.14. Un automate temporisé A est un 6-uplet (Q, q_0, X, Σ, E, I) avec :

- Q est l'ensemble fini des états ou de localité,
- $q_0 \in$ Q est l'état (localité) initial,
- X est un ensemble fini d'horloges,
- Σ est un alphabet fini décrivant l'ensemble des événements ou d'actions,
- $E \subseteq Q \times X(X) \times \Sigma \times 2^X \times Q$ est un ensemble fini de transitions, une transition de q vers q' est de la forme e =<q, g, a, r, q' > ∈ E avec g est la garde de la transition, a est l'événement et r est l'ensemble des horloges remises à zéro.
- I : $E \rightarrow X(X)$ associe un invariant à chaque état ou localité.

Exemple 1.4. Un exemple d'automate temporisé est présenté à la figure 1.10. Cet automate possède une seule horloge, notée x, deux états : q_0 et q_1. L'état q_0 est l'état initial de l'automate. Il possède deux transitions : une de l'état q_0 à l'état q_1 qui permet de réaliser l'action a si la valeur de l'horloge x est strictement supérieure à 0.8 et remet cette même horloge à 0, l'autre transition mène de l'état q_1 à l'état q_0, permet d'effectuer l'action b si x vaut plus que 0.8 et remet l'horloge x à zéro.

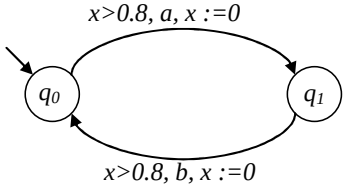


Figure 1.10. Automate temporisé.

Il est important de remarquer que les comportements futurs de l'automate ne dépendent pas uniquement de l'état dans lequel le système se trouve, mais aussi de la valeur de l'horloge *x*. En effet, on ne pourra pas faire la même action selon la valeur de *x* : si x = 0.5, on ne peut pas changer d'état, alors lorsque *x* > 0.8, il est possible de passer dans l'état q_1.

1.6.4.2 Composition des automates temporisés

Au même titre que les automates à états finis, il est possible de définir des opérations sur les automates temporisés. Nous dérivons ici le produit de deux automates, sachant que l'opération Produit est associative. Il existe donc également deux sortes de composition qui peut se faire pour obtenir un automate temporisé global.

☛ ***Composition synchrone***

Comme nous avons vu dans les automates à états finis classiques, cette composition est réalisée dans le cas où les automates, qui modélisent des sous-systèmes, ont d'événements en commun entre eux.

Soit deux automates temporisés A_1 et A_2 définis respectivement par : $A_1 = (Q_1, q_{01}, X_1, \Sigma_1, E_1, I_1)$, $A_2 = (Q_2, q_{02}, X_2, \Sigma_2, E_2, I_2)$.

La composition asynchrone entre A_1 et A_2 est défini comme étant l'automate $A=A_1 \| A_2 = (Q, q_0, X, \Sigma, E, I)$ tel que :

- $Q= Q_1 x\ Q_2$ est l'ensemble d'états,
- $q_0= (q_{01}, q_{02}) \in Q$ est l'état (localité) initial,
- $X= X_1 \cup X_2$ est l'ensemble d'horloges,
- $\Sigma= \Sigma_1 \cup \Sigma_2$ est l'ensemble d'événements ou d'actions,
- $E = E_1 \cup E_2 \subseteq Q \times X(X) \times \Sigma \times 2^X \times Q$ est un ensemble fini de transitions, une transition de q vers q' est de la forme $e = <q, g, a, r, q'> \in E$ est définie par :
  - $<(q_1,q_2), g_1 \wedge g_2, a, r_1 \cup r_2, (q'_1, q'_2)> \in E$ si $<q_1,, g_1, a, r_1, q'_1> \in E_1$ et $<q_2, g_2, a, r_2, q'_2> \in E_2$ et $a \in \Sigma_1 \cap \Sigma_2$.
  - $<(q_1,q_2), g_1, a, r_1 \cup r_2, (q'_1, q_2)> \in E$ si $<q_1, g_1, a, r_1, q'_1> \in E_1$ et $a \in \Sigma_1 \setminus \Sigma_2$.
  - $<(q_1,q_2), g_2, a, r_1 \cup r_2, (q_1, q'_2)> \in E$ si $<q_2, g_2, a, r_2, q'_2> \in E_2$ et $a \in \Sigma_2 \setminus \Sigma_1$.
- $I = I_1 \wedge I_1$ est l'ensemble des invariants.

Un exemple illustratif est présent en annexe 3 pour le lecteur.

☛ ***Composition asynchrone***

Au même titre que dans les automates à états finis classiques, cette composition est réalisée dans le cas où les automates, qui modélisent des sous-systèmes, n'ont pas d'événements en commun entre eux. L'automate global est alors le produit cartésien des automates des sous-systèmes ; un état global est en fait un vecteur des différents états des sous-systèmes (états locaux).

Soit deux automates temporisés A_1 et A_2 définis respectivement par : $A_1 = (Q_1, q_{01}, X_1, \Sigma_1, E_1, I_1)$, $A_2 = (Q_2, q_{02}, X_2, \Sigma_2, E_2, I_2)$.

La composition asynchrone entre A_1 et A_2 est défini comme étant l'automate $A=A_1 \parallel A_2= (Q, q_0, X, \Sigma, E, I)$ tel que :

- $Q= Q_1 x\ Q_2$ est l'ensemble d'états,
- $q_0= (q_{01}, q_{02}) \in Q$ est l'état (localité) initial,
- $X= X_1 \cup X_2$ est l'ensemble d'horloges,
- $\Sigma= \Sigma_1 \cup \Sigma_2$ est l'ensemble d'événements ou d'actions,
- $E = E_1 \cup E_2 \subseteq Q \times X(X) \times \Sigma \times 2^X \times Q$ est un ensemble fini de transitions, une transition de q vers q' est de la forme $e =<q, g, a, r, q'> \in E$ est définie par :
  - $<(q_1,q_2), g_1, a, r_1 \cup r_2,(q'_1, q_2)> \in E$ si $<q_1, g_1, a, r_1,q'_1> \in E_1$ et $a \in \Sigma_1 \setminus \Sigma_2$.
  - $<(q_1,q_2), g_2, a, r_1 \cup r_2,(q_1, q'_2)> \in E$ si $<q_2, g_2, a, r_2,q'_2> \in E_2$ et $a \in \Sigma_2 \setminus \Sigma_1$.
- $I = I_1 \wedge I_1$ est l'ensemble des invariants.

1.7 Conclusion

Dans ce premier chapitre de thèse, nous avons essayé de faire le point sur la terminologie utilisée dans le cadre de diagnostic de défauts. Cette terminologie choisie dans ce chapitre va nous permettre, par la suite, de commenter certains travaux de recherche qui ont été développés dans le domaine de diagnostic au cours des 10-15 dernières années. Ensuite, une classification de défauts a été présentée. Cette classification a été faite selon plusieurs critères et paramètres où nous avons montré la particularité de chaque catégorie de défauts ainsi que les différentes parties du système où un défaut peut avoir lieu. Nous avons

également présenté les différentes classes des systèmes dynamiques ainsi que les particularités de chaque classe. Le diagnostic de défauts dans les systèmes dynamique a été présenté dans son cadre générale.

Nous nous sommes intéressés par la suite aux systèmes à événements discrets (SEDs). Ces systèmes ont été représentés d'une manière générale avant de mettre en avant certains outils de représentation et modélisation. En effet, le choix des outils de modélisation qui ont été représentés dans ce chapitre (RdPs, Grafcet, automates à états finis et automates temporisés) est basé sur le fait que leur utilisation est importante en pratique. En revanche, l'utilisation de l'un de ces outils est arbitraire et peut dépendre du système lui-même, du niveau de complexité, tâche à réaliser, ainsi que de la maitrise de l'utilisateur. Ensuite, un intérêt personnel a été porté aux deux outils que sont les automates à états finis et automates temporisés. En effet, ces derniers nous aiderons dans le chapitre 3 lors de la présentation de notre travail. En conséquence, une étude plus détaillé et approfondie a été consacrée pour ces deux outils dans le reste de ce chapitre.

Dans le deuxième chapitre, nous allons approfondir les méthodes de diagnostic des SEDs retrouvées dans la littérature. En fait, la présentation de ces méthodes sera planifiée à l'aide d'une classification qui sera faite selon plusieurs critères (utilisation d'un modèle de diagnostic ou non, choix de l'outil de modélisation, structure de l'approche, …).

Chapitre 2

Diagnostic des Systèmes à Evénements Discrets

2.1 INTRODUCTION 39
2.2 APPROCHES SANS ET AVEC MODELE(S) 40
2.2.1 Approches sans modèle 40
2.2.1.1 Arbre de défaillances 41
2.2.1.2 Système expert 42
2.2.1.3 Approches à base de classification de données 42
2.2.1.4 Approches à base des Réseaux Bayésiens 43
2.2.2 Approches à base de modèle(s) 44
2.3 REPRESENTATION DES DEFAUTS DANS LE MODELE 45
2.3.1 Approche avec représentation des défauts dans le modèle 45
2.3.2 Approche sans représentation de défauts dans le modèle 46
2.4 OUTILS DE REPRESENTATION DES APPROCHES DE DIAGNOSTIC 47
2.4.1 Modèles à base d'automates à états finis 47
2.4.1.1 Approches à base d'événements 48
2.4.1.2 Approches à base d'états 49
2.4.2 Approches à base de Réseaux de Petri (RdP) 52
2.4.3 Approches à base de Templates 54
2.4.4 Approches à base de Chroniques 55
2.5 STRUCTURE DU DIAGNOSTIC 57
2.5.1 Structure centralisée 57
2.5.2 Structure décentralisée 59
2.5.3 Structure distribuée 62
2.5.4 Choix d'une structure 64
2.6 NOTION DE DIAGNOSTICABILITE 66
2.7 CONCLUSION 67

2.1 Introduction

Le premier chapitre introductif a permis de présenter quelques définitions concernant la terminologie utilisée dans le domaine de diagnostic de défauts. Cette terminologie nous sera nécessaire que ce soit dans le présent chapitre ou dans les chapitres suivants. Afin d'assurer un fonctionnement correct et sûr des systèmes dynamiques, le diagnostic de défauts dans les Systèmes à événements discrets a reçu une attention considérable ces dernières années. L'intérêt porté pour ce domaine a commencé au travers des travaux réalisés par Lin et Wonham il y a environ 30 ans [Lin et Wonham, 1988(a)] [Lin et Wonham, 1988(b)] [Lin et Wonham, 1990]. Dans leurs premiers travaux, ils se sont intéressés à la supervision et la synthèse de commande des SEDs. Ce deuxième chapitre est consacré à la présentation d'un état de l'art d'une façon succincte sur les principales méthodes de diagnostic des SEDs rencontrées dans la littérature. Cet état de l'art nous permettra de positionner notre travail de recherche par rapport aux approches présentées.

Le diagnostic précoce des défauts dans un système peut aider à éviter la progression de ces derniers. Une propagation d'un défaut a souvent de graves conséquences et ainsi peut conduire le système dans un état de défaillance. Le domaine de diagnostic est un aspect important dans l'ingénierie des systèmes. Cette importance n'est pas seulement du point de vue de la sûreté de fonctionnement mais aussi pour atteindre les objectifs de la maintenance afin de garantir un bon rendement et une bonne qualité concernant le fonctionnement du système. L'intérêt porté au diagnostic, que ce soit de la part des industriels ou de la part des chercheurs académiques, est dû à l'impact nocif de l'occurrence de défauts dans le système [Nimmo, 1995]. Il a donnée naissance à une grande variété de techniques et approches visant le diagnostic de défauts dans les systèmes dynamiques et plus particulièrement les SEDs.

L'objectif fondamental des sections suivantes est de présenter succinctement les différentes approches de diagnostic. Une classification de ces méthodes peut être faite selon plusieurs paramètres ou critères :

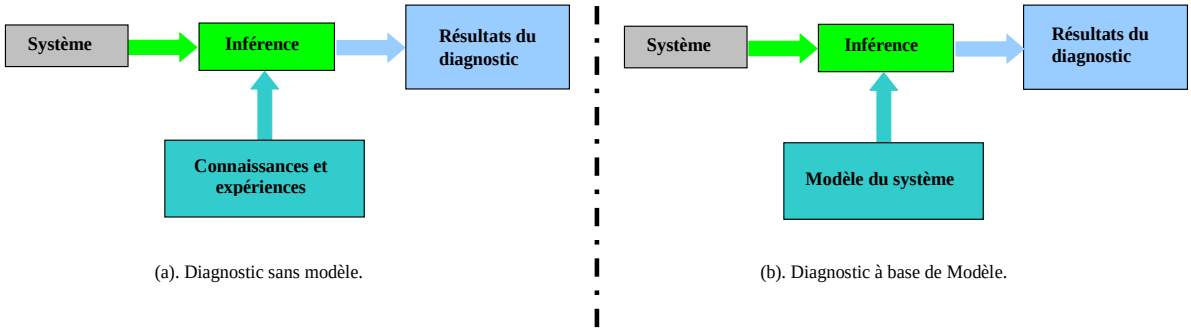
- Approches sans et avec modèle(s),
- Selon représentation des défauts dans le modèle,
- Outils de représentation des approches de diagnostic,
- Selon la structure du diagnostic.

2.2 Approches sans et avec modèle(s)

Les approches de diagnostic des SEDs, qu'on peut rencontrer dans la littérature, peuvent être classifiées selon un critère important qu'est le type de connaissances utilisées concernant le système à diagnostiquer. Les connaissances a priori de base qui sont nécessaire pour la construction du module de diagnostic sont les suivantes :

- L'ensemble de défauts qui peuvent avoir lieu dans le système,
- L'ensemble des observations et le type de relations qui les relient avec l'ensemble de défauts.

Les connaissances à priori peuvent être récoltées à partir de l'expérience humaine en utilisant le système. Ce type de connaissance est considéré comme peu profond. Les approches basées sur un tel type de connaissances sont appelés « *méthodes sans modèle* » ou bien aussi « *méthodes à base de données* » (Figure 2.1(a)). D'autre part, elles peuvent être développées à partir d'une compréhension fondamentale de la dynamique du système. Les approches qui se basent sur une telle connaissance sont appelés « *méthodes à base de modèle* » (Figure 2.1(b)).



(a). Diagnostic sans modèle.

(b). Diagnostic à base de Modèle.

Figure 2.1. Principes de diagnostic sans et à base de modèle.

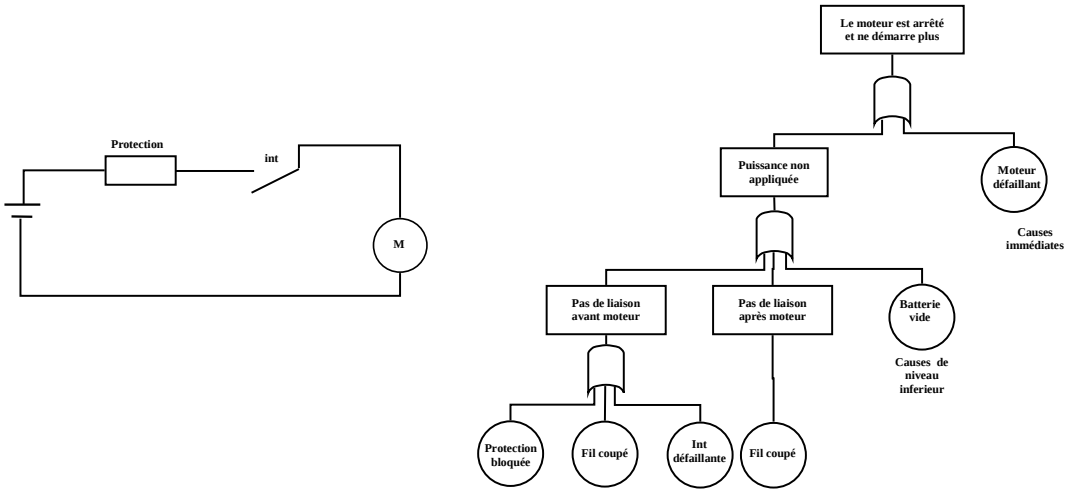
2.2.1 Approches sans modèle

Les méthodes à base de données ou bien sans modèle font appel à la disponibilité de grandes quantités de données issues des enregistrements fait le long du fonctionnement du système. Dans cette partie, nous décrivons brièvement quelques approches qui ont en point commun l'association explicite des défauts connus à priori à un comportement observé. Les méthodes utilisant ces approches sont souvent appelés *méthodes associatives*.

2.2.1.1 Arbre de défaillances

L'arbre de défaillance est une technique largement utilisée dans l'industrie pour l'analyse de la fiabilité et le diagnostic des défauts dans des systèmes modélisés en tant que SEDs afin de garantir leur bon fonctionnement [Lapp et Powers, 1977] [Lee et al., 1985] [Viswanadham et Johnson, 1988] [Vries, 1990]. Cette méthode a été proposée par H.A. Watson, en 1962 dans les laboratoires de la "Bell Telephone Company" et dans le cadre du projet des missiles "minuteman ICBM" commandé par US Air Force. Un arbre de défaillance est un graphe orienté qui représente des relations de cause à effet entre les défauts dans le système. Généralement, l'arbre est présenté de haut en bas. L'idée de base de cette approche est de suivre une logique déductive qui consiste à rechercher les causes immédiates d'un événement indésirable (événement redouté) correspondant à une défaillance du système. Les relations entre les différents événements sont établies en utilisant des portes logiques traditionnelles de l'algèbre de Boole (And, Or, Not,…). Le processus est répété jusqu'à ce que la construction de l'arbre soit complète. Dans cette approche, une probabilité d'un défaut peut être trouvée en attribuant des probabilités aux événements de base. L'Arbre de Défaillances a l'avantage d'être facilement compréhensible par tout utilisateur.

Exemple 2.1. Considérons un simple circuit (Figure 2.2(a)) représentant un moteur alimenté par une batterie. La mise en marche/arrêt du moteur *M* se fait via un interrupteur « *int* ». Un arbre de défaillance de ce circuit est illustré dans la figure 2.2(b). Cet arbre de défaillance affiche les causes lorsque le système est dans état où le moteur ne démarre plus.



(a). Circuit d'alimentation d'un moteur.

(b). L'arbre de défaillance du circuit du moteur.

Figure 2.2. Exemple d'un arbre de défaillance.

2.2.1.2 Système expert

C'est une méthode populaire pour le diagnostic et la supervision des systèmes complexes, elle est souvent utilisée en conjonction avec les structures de l'arbre de défaillances [Tzafestas et al., 1990]. Les systèmes experts sont particulièrement bien adaptés pour les systèmes qui sont difficiles à modéliser, et qui comportent des interactions complexes entre les composants. Dans les systèmes experts traditionnels, cette connaissance est représentée à base des règles et utilisé en conjonction avec un moteur d'inférence.

Cette approche heuristique a plusieurs inconvénients. L'acquisition de connaissances auprès d'experts est difficile et prend du temps, et pour de nouveaux systèmes une quantité considérable de temps pourrait s'écouler avant que suffisamment de connaissances soient accumulées pour faire un diagnostic fiable possible.

2.2.1.3 Approches à base de classification de données

Lorsque le modèle explicite dynamique n'est pas disponible, la connaissance du système se résume à des mesures en temps réel, éventuellement complétée par l'historique du processus. Avec ces données, deux stratégies principales pourraient être adoptées. Dans un sens, toutes les deux visent à interpoler le nouveau point mesuré en se basant sur les données disponibles. La première stratégie est la classification [Kramer et Mah, 1993] [Boudaoud, 1997] [Friedman et Kandel, 1999]. Elle implique la construction de classes à partir de la base de données soit d'une manière supervisée (c'est à dire avec l'aide d'un expert) ou d'une manière non supervisée (par la collecte des éléments de la base de données qui sont proches les uns des autres). Un classifieur est ensuite construit à l'égard de ces classes pour effectuer la classification des variables nouvellement mesurées en tant que représentation d'un comportement normal ou défaillant. La deuxième stratégie est la régression. Elle construit un modèle statistique qui utilise la redondance de l'historique du processus afin de prédire les valeurs des nouvelles variables et de générer des résidus en comparant les prédictions aux valeurs mesurées.

La figure 2.3 montre un exemple issu d'une méthode de classification statistique. Les données sont représentées selon deux paramètres « Amplitude » et « Magnitude ». Comme est indiqué sur la figure, ces données sont présentées pour être regroupées en trois conditions de

défaut {défaut1, défaut2, défaut3}. Les points qui n'appartiennent à aucune classe représentent une erreur de classification. Par contre, l'appartenance des points à plusieurs classes en même temps signifie une ambigüité.

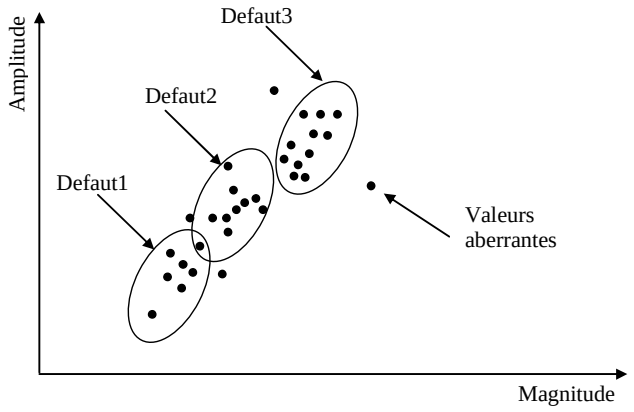


Figure 2.3. Exemple d'une méthode de Classification des données statistiques.

2.2.1.4 Approches à base des Réseaux Bayésiens

Les réseaux Bayésiens sont largement utilisés dans les cas de la modélisation de connaissances incertaines et offrent un moyen de décrire les distributions de probabilité d'une façon compacte [Breese et al., 1988]. Formellement, ce sont des graphes acycliques orientés dont les nœuds sont des variables aléatoires telles que les causes, les effets, etc. Les arcs du réseau représentent les relations de dépendance entre ces variables. Lors de la conception d'un réseau Bayésien, on spécifie généralement la dépendance des variables, les probabilités a priori des nœuds racines, et les probabilités conditionnelles des nœuds non-racines en se basant sur leurs prédécesseurs directs. La figure 2.4 illustre un réseau Bayésien simple ainsi que l'avantage de calcul acquis en utilisant cette méthode. Le réseau montre que « *A* » provoque l'occurrence de « *B* » et « *C* » et que « *C* » et « *D* » sont des conséquences de l'occurrence de « *E* ». Les quantités requises sont les probabilités *P*(*A*) et *P*(*D*) et les probabilités conditionnelles *P*(*B**A*), *P*(*C**A*) et *P*(*E**D*, *C*). Par rapport à la représentation directe *P*(*A, B, C, D, E*), si *A, B, C, D* et *E* sont considérés comme des variables binaires (de type vrai/faux), la représentation directe a besoin de 2^5-1 = 31 distributions, alors que le réseau Bayésien n'a besoin que d'un modèle à 5 nœuds.

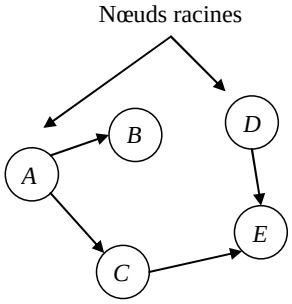


Figure 2.4. Exemple d'un réseau Bayésien.

Les avantages de ces réseaux sont les suivants: (1) Si les distributions données sont cohérentes, alors toutes les probabilités, calculées en utilisant ces distributions, sont également cohérentes. (2) Le réseau définit toujours de façon unique une distribution; ce qui veut dire que la distribution conjointe pour un réseau Bayésien est définie de façon unique pour chaque variable aléatoire. L'évaluation d'un réseau Bayésien est une tâche ardue, car il s'agit généralement d'un problème NP-difficile. Des solutions exactes ne sont disponibles que pour les réseaux qui sont simplement connectés.

Les réseaux Bayésiens peuvent être utilisés pour étudier le problème de diagnostic [Alonso-Gonzalez et al., 2010] [Ricks et Mengshoel, 2010]. Des probabilités sont calculées pour l'occurrence de défauts en utilisant les informations probabilistes collectées à partir du fonctionnement du système.

Les approches se basant sur l'utilisation des réseaux Bayésiens peuvent se confronter à quelques inconvénients. Les informations probabilistes a priori qui sont nécessaires pour effectuer le diagnostic ne sont pas toujours disponibles. De plus, la dépendance des événements n'est pas toujours satisfaite. Ainsi la complexité de calcul est presque insoluble [Freyermuth, 1991].

2.2.2 Approches à base de modèle(s)

Le principe général de ces méthodes est de détecter une divergence entre le comportement attendu représenté par un modèle du système et le comportement observé représenté par les informations (mesures réelles) fournies par les capteurs. La modélisation est une tâche difficile, et la qualité du modèle influence d'une façon très importante la qualité des résultats retournés par le module du diagnostic. En conséquence, si un modèle approprié peut être développé alors une approche à base de modèle fournit une meilleure solution pour le problème de diagnostic. En fait, la construction d'un modèle qui contient des informations bien profondes à propos du fonctionnement du système est mis à profit d'un diagnostic plus

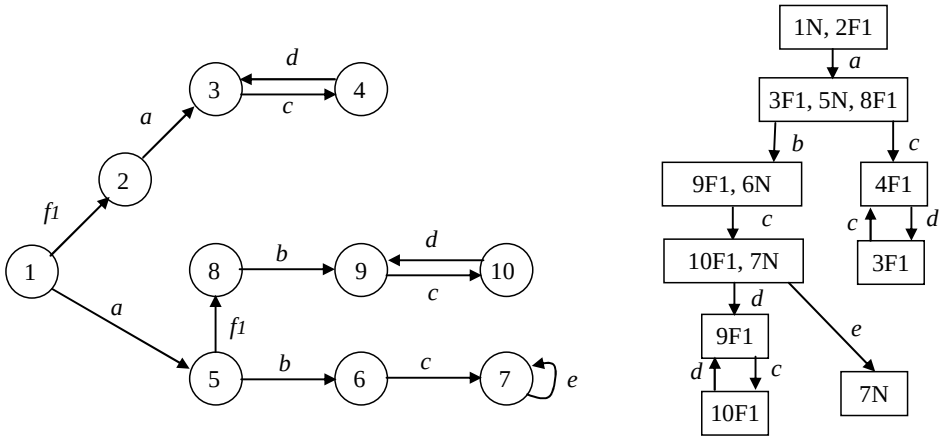
précis [Reiter, 1987]. En outre, le modèle des solutions peut offrir des méthodes d'analyse pour vérifier les propriétés importantes, telles que l'établissement des diagnostics. De plus, les approches à base de modèle peuvent offrir des solutions afin de pouvoir vérifier des propriétés importantes telles que la diagnosticabilité du système.

Les approches de diagnostic à base de modèles diffèrent principalement par la façon dont laquelle le système est modélisé (normal ou anormal) ainsi que l'outil de modélisation utilisé pour représenter le fonctionnement de ce dernier. La manière dont le système est modélisé influence largement les techniques de diagnostic de défauts, et la précision de diagnostic qu'ils peuvent obtenir.

2.3 Représentation des défauts dans le modèle

2.3.1 Approche avec représentation des défauts dans le modèle

Dans ces approches, durant la modélisation d'un système, le comportement normal et le comportement défaillant, représentant tous les défauts qui peuvent avoir lieu dans un composant, doivent être explicitement inclus dans chaque modèle des composants élémentaires qui constituent le système. Par conséquent, le modèle global du système issu par exemple en utilisant la composition synchrone des différents modèles élémentaires englobera l'ensemble de tous les défauts possibles dans le système. Dans ces approches, l'objectif principal de cette modélisation est la construction d'un diagnostiqueur qui pourra prendre une décision concernant l'état du système à un instant donnée. Ensuite, un compte rendu concernant l'occurrence des défauts dans le système est envoyé par ce diagnostiqueur sous forme d'étiquettes [Sampath et al., 1995] [Debouk, 2000] [Genc et Lafortune, 2003].



a) Modèle du procédé b) Diagnostiqueur

Figure 2.5. Modèle du système et son diagnostiqueur.

Exemple 2.2. La figure 2.5(a) présente la modélisation d'un procédé à travers un automate à états pour la détection d'un événement de défaut f_1.

Pour ce modèle de procédé, le diagnostiqueur, ou modèle avec représentation de défauts, obtenu est présenté sur la figure 2.5(b). Chaque numéro associé à une lettre, ou étiquette, permet d'associer à un état du modèle, un état du comportement du système. Ainsi, 1N correspond à l'état 1 en fonctionnement normal alors que 2F1 correspond à l'état 2 du procédé en fonctionnement anormal associé au défaut f_1. Un état du diagnostiqueur avec une seule étiquette permet de confirmer avec certitude l'état du système, alors qu'un état avec plusieurs étiquettes ne permet pas d'isoler le défaut et de s'assurer du bon fonctionnement du procédé. Pour l'exemple de la figure 2.5, c'est après l'occurrence de l'événement *c* que l'événement de défaut f_1 est détecté par l'étiquette F1. C'est donc l'occurrence de l'événement *c* qui permet d'isoler le défaut.

2.3.2 Approche sans représentation de défauts dans le modèle

Les approches qui utilisent cette représentation se basent sur la construction d'un modèle du système sans intégrer l'ensemble des défauts possibles. Parmi les premiers travaux qui ont été réalisés dans ce sens celui dans [Reiter, 1987]. Dans ce travail, une théorie générale pour le diagnostic avec une description du système sans utiliser un modèle de défauts a été proposée. Cette approche appartient au domaine de l'intelligence artificielle et se base sur une description abstraite du système qui ne peut être réalisée en utilisant différentes

représentations de la logique. En effet, le système est modélisé par une paire (*DS*, *COMPS*), où *DS* (anglais : Description System) représente une description du système en utilisant un ensemble de formules de la logique du premier ordre sous forme d'égalités, tandis que *COMPS* (anglais : Component) regroupe un ensemble fini de composants appartenant au système (actionneurs, capteurs…etc.). Les observations dans le système sont représentées via l'ensemble *OBS* à travers des formules utilisant également la logique du premier ordre. Par conséquent, le système observé à diagnostiquer est représenté par un triplet (*DS*, *COMPS*, *OBS*) où (*DS*, *COMPS*) est le modèle du système tandis que *OBS* est l'ensemble d'observations. Le fonctionnement d'un composant *c* appartenant à *COMPS* est exprimé via un prédicat de l'ensemble *DS* noté *AB*. Le fonctionnement anormal du composant *c* est alors donné par $AB(c)$ alors que son fonctionnement normal est donné par $\neg AB(c)$. Le diagnostic de défauts consiste alors en une vérification entre les observations actuelles *OBS* du système et sa description *DS*. Cette procédure est donc similaire à celle retrouvée dans les approches de diagnostic à base de modèles.

Une autre approche de diagnostic sans modèle de défauts a été développée par Pandalai et Holloway dans [Pandalai et Holloway, 2000]. Cette approche est proposée pour le diagnostic des systèmes manufacturiers à travers la construction de « Templates » avec capteurs et actionneurs discrets. Ces travaux seront développés plus précisément par la suite dans ce chapitre.

2.4 Outils de représentation des approches de diagnostic

Comme cela a été évoqué dans le premier chapitre, les systèmes à événements discrets peuvent être modélisés de plusieurs façons, par exemple, avec un automate à états finis ou un réseau de Petri, etc. En conséquence, la grande variété de travaux de recherche qu'on peut rencontrer dans la littérature, et qui portent sur le diagnostic de défauts pour les systèmes à événements discrets, peuvent être classé selon l'outil de modélisation utilisée afin de construire un modèle de système ainsi que le module de diagnostic.

2.4.1 Modèles à base d'automates à états finis

Dans le cadre du diagnostic des systèmes à événements discrets, plusieurs approches originales ont été proposées en utilisant les automates à états finis comme un outil de

modélisation. En conséquence, on peut classifier ces approches dans deux catégories principales : celles à base d'événements et celles à base d'états.

2.4.1.1 Approches à base d'événements

Cette approche a été initialement proposée par Sampath dans [Sampath et al., 1995] [Sampath et al., 1996(a)] et est appelée *diagnostiqueur*. Elle comporte deux étapes principales. Dans la première étape un modèle à événements discrets représentant le comportement du système est construit. Dans cette phase, l'idée de base consiste à modéliser chaque composante élémentaire du système à l'aide des automates à états finis. Chaque automate représente les deux comportements normal et défaillant pour le composant correspondant. Ensuite, le modèle global du système peut être construit automatiquement via la composition synchrone des modèles représentant ses composantes élémentaires. Dans la deuxième étape, le modèle global issu de la synchronisation des modèles élémentaires du système est exploité pour la compilation (hors-ligne) du diagnostiqueur correspondant. Ce diagnostiqueur est un automate à états finis. A chaque nœud de cet automate est associée une paire d'éléments, le premier élément de cette paire représente l'état du système, le deuxième élément est une liste de défauts qui peuvent avoir lieu dans son état représenté par le premier élément de cette paire. Les arcs reliant les nœuds du diagnostiqueur définissent comment le système peut évoluer au fil du temps. Ces arcs sont étiquetés avec des événements observables du système à diagnostiquer.

Une fois que le diagnostiqueur est compilé, la tâche de diagnostic devient simple. En fait, à chaque fois qu'un événement e est observé le diagnostic consiste à suivre un arc étiqueté avec e de telle sorte que le nœud actuel du diagnostiqueur conduit à un nouveau nœud. Ce nouveau nœud comporte des informations sur l'état du système et la liste des défauts qui sont possible à avoir lieu.

Si nous reprenons l'exemple de la figure 2.5, le modèle du procédé dispose seulement d'informations provenant des événements. Ces états sont, par conséquent, difficilement interprétables. En effet, à partir de l'état initial, il est impossible de savoir si le système se trouve dans l'état 1 en fonctionnement normal ou dans l'état 2 indiquant l'occurrence du défaut f_1. De même, à partir de l'occurrence de l'événement a, il est difficile de savoir si le

procédé se trouve dans l'état 5 indiquant un fonctionnement normal du procédé ou dans l'état 8 indiquant un défaut de type F1. Il faut faire appel au diagnostiqueur de la figure 2.5(b) ayant la connaissance de l'accessibilité des états pour en savoir plus sur l'état de fonctionnement du système. Ce type de diagnostiqueur correspond à un automate réduit aux événements observables avec mémoire des événements de défauts représentés par des étiquettes indiquant le fonctionnement normal ou défaillant.

Cependant, le diagnostiqueur doit être initialisé en même temps que le procédé afin qu'il puisse suivre l'évolution du procédé. Cette initialisation est difficile à obtenir pour les systèmes manufacturiers. En effet, lorsqu'un défaut survient et demande un arrêt complet du système, la procédure de remise en route consiste très souvent à réinitialiser la commande, la partie opérative et également donc les diagnostiqueurs. Dès lors, il n'est pas possible de connaître la situation de ces derniers puisqu'ils sont modélisés à travers leurs événements et non pas leurs états.

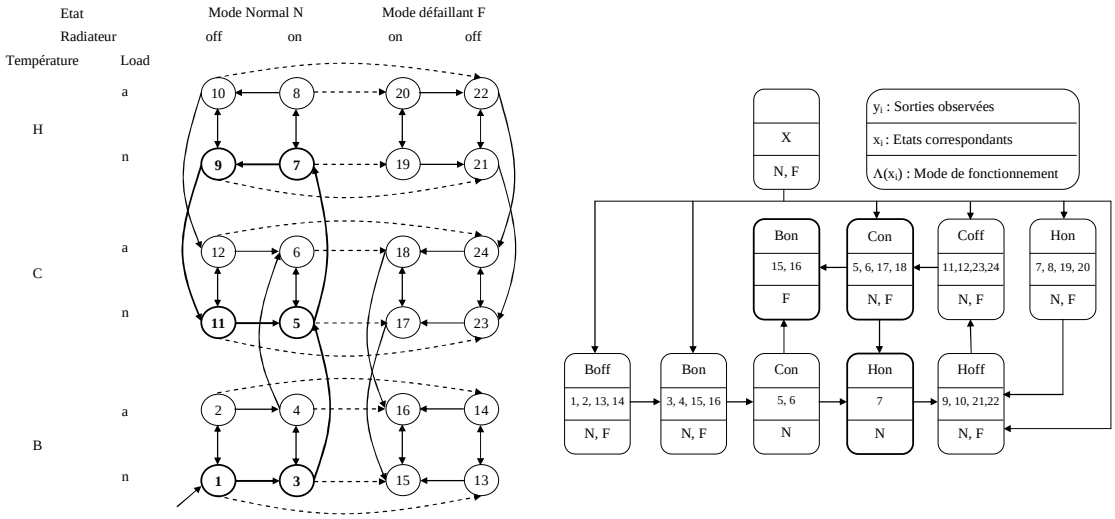
2.4.1.2 Approches à base d'états

Parmi les travaux de recherche qui ont été destiné pour le diagnostic des systèmes à événements discrets, une approche à base d'état a été proposée par [Zad et al., 2003]. Dans ce type d'approche, l'accent est mis sur la détermination de l'état défaillant du système (occurrence des défauts) sans avoir besoin aux événements. C'est parce que, dans les cas pratiques, le modèle global du système est obtenu en composant plusieurs sous-modèles représentant les composants élémentaires. Généralement, chacun de ces sous-modèles dispose d'un seul mode de fonctionnement normale et un nombre limité des modes de fonctionnement défaillant. Ainsi, il existe une relation directe entre l'état du système et son mode de fonctionnement défaillant. L'approche dans [Zad et al., 2003], suppose que l'ensemble d'états du système peut être partitionné en fonction du comportement du système. Ces partitions sont définies par un seul mode de fonctionnement normal et plusieurs modes de fonctionnement. Afin d'effectuer un diagnostic à base d'état pour les systèmes à événements discrets, le modèle est augmentée pour inclure l'ensemble des sorties du système. Dans ce cas, l'état de fonctionnement du système est déduit de la séquence des sorties. Un diagnostiqueur à base d'états est illustré dans [Zad et al., 2003]. Chacun des états de ce diagnostiqueur contient une sortie du système, ses états possibles liés à cette sortie, et les modes de fonctionnements

possibles du système associés à ces états. Dans cette approche, la compilation du diagnostiqueur ainsi que la définition de la diagnosticabilité se font de la même manière que dans l'approche à base d'événements.

Exemple 2.3. Nous illustrons dans la figure 2.6, un exemple inspiré de [Zad et al., 2003], qui présente un modèle SED et son diagnostiqueur.

Le modèle du procédé global est représenté en figure 2.6(a) où les arcs en pointillés représentent un événement de défaut d'un radiateur. Le label « Load » modélise les effets de la perturbation sur le capteur de température dus à la température ambiante ou aux transferts de chaleur des salles voisines. Les perturbations sont alors représentées par deux états : *n* pour normal et *a* pour perturbé. Elles sont considérées comme des perturbations compensables par le régulateur. L'hypothèse faite est de ne pas les prendre en compte et qu'en présence de perturbation, la température garde sa consigne.



a) Modèle du procédé à base d'états

b) Diagnostiqueur à base d'états

Figure 2.6. Modèle du système et son diagnostiqueur à base d'états .

Les six situations du procédé correspondent aux sorties observables suivantes :

- Boff : température basse (B), radiateur éteint (off)
- Bon : température basse (B), radiateur allumé (on)
- Coff : température sous la consigne (C), radiateur éteint (off)
- Con : température sous la consigne (C), radiateur allumé (on)

- Hoff : température au-dessus de la consigne (H), radiateur éteint (off)
- Hon : température au-dessus de la consigne (H), radiateur allumé (on)

Le modèle du procédé est alors composé de l'ensemble des états X = {1, 2, …, 24}, de l'ensemble des sorties observables des états Y = {Boff, Bon, Coff, Con, Hoff, Hon} et de l'ensemble des labels, ou étiquettes, de fonctionnement normal et défaillant Λ = {N, F} où $l(x_i)$ = N pour les états de $1 \leq i \leq 12$ et $l(x_i)$ = F pour les états de $13 \leq i \leq 24$. Ainsi, à partir de l'état 1 correspondant à la situation du radiateur éteint avec une température basse sans perturbation, il est possible d'évoluer vers l'état 3 où le radiateur est allumé. La température se trouve sous la consigne dans l'état 5. Lorsque la température dépasse la consigne alors le procédé est dans l'état 7 puis dans l'état 9 afin d'arrêter le radiateur et de redescendre en dessous de la consigne en état 11. La régulation de la température est alors représentée par le retour à l'état 5. Le modèle indique bien qu'à partir de chaque état, il est possible d'avoir un défaut et de retrouver le procédé dans un mode défaillant F qui correspond à une défaillance de la résistance de chauffe du radiateur.

A partir du modèle du procédé à base d'états, l'auteur de [Zad et al., 2003] construit un diagnostiqueur à base d'états (Figure 2.6(b)). Le diagnostiqueur est composé d'états enrichis indiquant les sorties observées y_i, les états correspondant x_i et le label du mode de fonctionnement du procédé $l(x_i) \in \Lambda$. Ainsi, pour la sortie Con, les états accessibles sont {5, 6, 17, 18} et le mode est soit normal N ou défaillant F. A partir de ces états, il est possible d'évoluer soit vers l'état 7 représenté par la sortie Hon, soit vers les états 15 et 16 associés à la sortie Bon (états en gras sur la figure 2.6(b)). Dès lors, les états 15 et 16 représentant un défaut, l(15) = l(16) = F, sont isolés de l'état 7 représentant un fonctionnement normal, l(7) = N.

Cet exemple ne traite pas de l'ensemble des défauts possibles et suppose notamment d'avoir des capteurs robustes qui ne sont pas considérés comme susceptibles d'être défaillants. Ce type de modélisation a l'avantage de ne pas avoir besoin d'initialiser le diagnostiqueur en même temps que le procédé. Par contre, l'inconvénient de la modélisation à base d'états est qu'elle ne peut détecter les pannes intermittentes qui sont des événements brefs amenant à des états non stables, donc non représentatifs.

2.4.2 Approches à base de Réseaux de Petri (RdP)

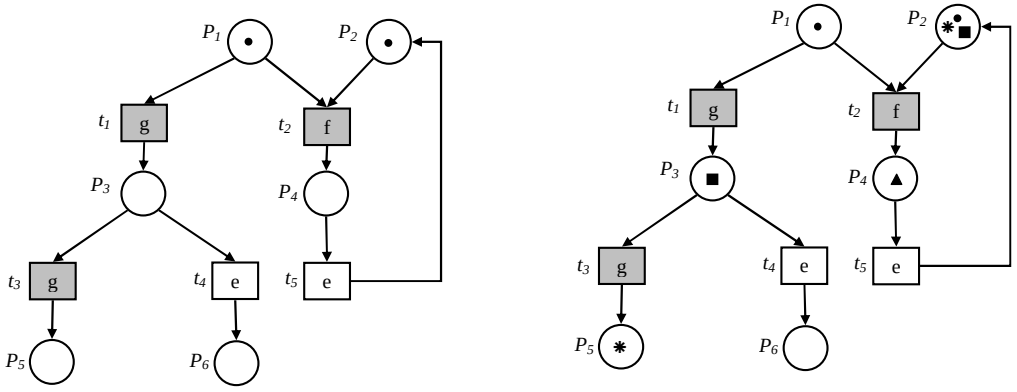
Parmi les premiers travaux qui ont été réalisés en utilisant les Réseaux de Petri, nous rappelons celui de Prock [Prock, 1991]. Dans son travail, l'auteur propose une technique de détection en ligne des défauts en se basant sur une surveillance du nombre de jetons dans les P-invariants: lorsque le nombre de jetons à l'intérieur des P-invariants changent, alors une erreur est survenue.

En général, dans les approches à base de réseaux de Petri, les modèles utilisés pour le diagnostic du système (diagnostiqueurs) ont la même structure du système à diagnostiquer. Dans le modèle RdP d'un système, certaines places peuvent être non bornées. Lorsqu'on dit « place non bornée » cela signifie que le nombre de jetons dans cette place n'est pas connu. D'autre part, les transitions peuvent être observables ou non observables. En générale, dans les systèmes modélisés par un RdP, le problème de diagnostic consiste à déterminer la distribution des jetons dans le réseau de Petri dans les places bornées et non bornées (ou la séquence de tir des transitions non observables) en se basant seulement sur les informations observables du système.

Dans l'approche proposée par [Genc et Lafortune, 2003], le système est modélisé par un RdP labellisé et borné. Un label, ou étiquette, est une indication sur le type d'événement présent sur les transitions du RdP. Ces étiquettes appartiennent à un ensemble d'événements et elles peuvent être observables ou non observables (défauts). Le problème de diagnostic consiste à déterminer les événements non observables qui peuvent avoir lieu dans le système en se basant sur l'analyse des événements observables. Dans cette approche, Le diagnostiqueur est construit en tant que réseau de Petri et il représente les deux comportements, normal et défaillant, du système. Pendant que la dynamique du système est définie par les événements observables et non observables, le diagnostiqueur évolue uniquement en fonction des événements observables. En exploitant les événements observables, le diagnostiqueur peut estimer l'état futur du système. Ensuite, il peut également indiquer les types des défauts qui peuvent avoir lieu dans le système.

Exemple 2.4. La figure 2.7(a) représente un modèle RdP d'un système ainsi que son diagnostiqueur. Les transitions qui sont labellisées par des événements non observables

apparaissent en gris. Dans le diagnostiqueur (figure 2.7(b)), chaque état représente plusieurs marquages possibles du système. Chaque marquage représente une configuration quelconque du système et illustrée par un jeton symbolique. Le marquage donné par le symbole « • » et qui se trouve dans les places (P_1 et P_2) représente l'état du système, comme on le voit sur la figure 2.7(a). A partir de cet état du système, une occurrence de l'événement non observable *g* est possible. Si cela est le cas, alors la transition t_1 sera franchie. Le franchissement de t_1 conduit à un changement du marquage des places (P_2 et P_3). Ce nouvel état est représenté par les symboles « ■ ». L'occurrence de l'événement *g* pour une deuxième fois conduit au franchissement de la transition t_3. Ce franchissement change le marquage des places (P_2 et P_5) qui apparait alors avec un jeton symbolique « ✱ » dans le diagnostiqueur. Dans cette approche de diagnostic, la matrice des marquages du diagnostiqueur (Figure 2.7(c)) permet de connaître le ou les comportements possibles du système.



(a). Modèle RdP du système.

(b). Diagnostiqueur à base de RdP.

$$X_{d0} = \left(\begin{array}{cccccc|cc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & g & f \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right) \begin{array}{c} \bullet \\ \blacksquare \\ \blacktriangle \\ \ast \end{array}$$

c) Matrice d'états du diagnostiqueur

Figure 2.7. Exemple du diagnostic à base de réseaux de Petri.

Des travaux ont été également développés par Wu et Hadjicostis afin de permettre la détection et l'identification de défauts en utilisant des techniques de décodage algébrique [Wu et Hadjicostis, 2005]. Dans cette approche, les auteurs considèrent deux types de défauts: (1) des défauts sur les places qui peuvent modifier le marquage du réseau Petri, et (2) des défauts

sur les transitions qui provoquent une mise à jour incorrecte du marquage après occurrence d'un événement. Bien que cette approche soit générale, le marquage du réseau de Petri doit être périodiquement observable. Dans le même sens, Lefebvre et Delherm ont cherché à déterminer un ensemble de places qui doit être observé pour une estimation exacte et immédiate de l'occurrence des défauts [Lefebvre et Delherm, 2007]. Par la même, dans les travaux de Genc et Lafortune [Genc et Lafortune, 2007], les auteurs proposent un diagnostiqueur modulaire qui assure le diagnostic des défauts dans chaque module, sous-système, du procédé. Un canal de communication relie les différents modules de diagnostic afin d'assurer mise à jour de leurs informations. Même si l'approche est exposée au problème d'explosion combinatoire en termes de places, une amélioration de cette approche est proposée par le couplage de places communes pour la communication des décisions locales [Benveniste et al., 2003] [Basile et al., 2009] [Dotoli et al., 2009].

2.4.3 Approches à base de Templates

Dans l'approche proposée dans [Pandalai et Holloway, 2000] et [Holloway et Chand, 1994], les auteurs utilisent un modèle du système à base d'un automate temporisé sans horloge mais dont l'intervalle de temps est associé à des états discrets. Dans cette approche, les auteurs proposent une technique de diagnostic à base des templates. Les templates représentent des contraintes temporelles entre un événement déclencheur initial et un ou plusieurs événements conséquences (Figure 2.8). Les templates sont déterminées par un ensemble de règles à partir d'un événement déclencheur initial. En conséquence, en se basant sur l'occurrence de cet événement, un ensemble d'instances est déterminé. Cet ensemble contient les événements possibles qui suivent l'occurrence de l'événement initial associés à des intervalles de temps appropriés. Une instance est décrite sous la forme (t, e, C, w) où t est le temps, e est un événement, C est une conséquence et w représente une étiquette (label de l'instance). La conséquence C est une paire (e', τ), où e' est un événement et τ est un intervalle de temps.

Exemple 2.5. Considérons un exemple de templates illustré par la figure 2.8 où deux instances sont représentés: (2, e_1, {(e_2, [2, 4]), (e_3, [6, 8])}, w_1) et (9, e_3, {(e_5, [3, 5])}, w_2).

Les deux premiers éléments de l'instance w_1 indiquent que l'événement e_1 a eu lieu à l'instant t = 2 unités de temps. L'instance est satisfaite si e_2 apparaît dans son intervalle valide

$4 \leq t \leq 6$ ou si e_3 apparaît dans $8 \leq t \leq 10$. Dans le cas d'occurrence de l'événement e_3, la prochaine instance w_2 est déclenchée. Dans cet exemple, l'occurrence de l'événement e_2 (flèche en pointillé) n'est pas prévue par l'attente w_1. Par conséquent, on en déduit que l'occurrence tardive de l'événement e_2 est dû à un défaut.

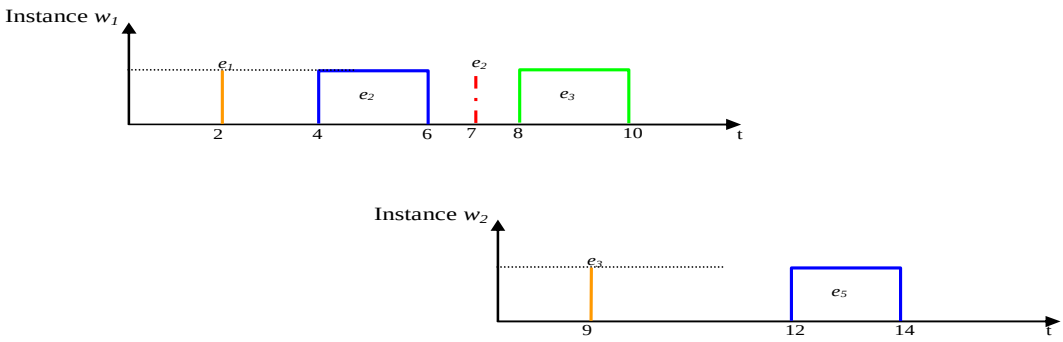


Figure 2.8. Exemple d'une *Template*.

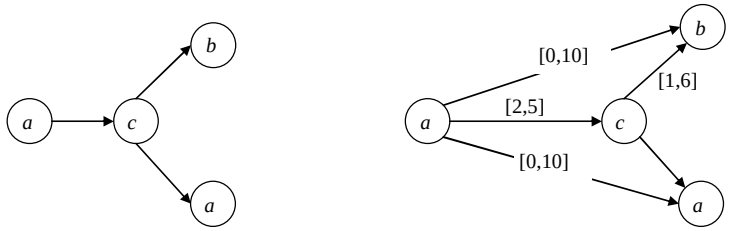
Dans [Pandalai et Holloway, 2000], les travaux sont poursuivis sur le même formalisme afin de modéliser les systèmes à événements discrets. Ce formalisme permet la modélisation des processus dans lesquels les comportements à instance unique et les comportements à instances multiples sont exposées simultanément. Un comportement à instance unique correspond à une trace d'un seul processus alors qu'un comportement à instances multiples correspond aux entrelacements temporisés d'un nombre non spécifié de processus identiques qui s'exécutent simultanément. Le formalisme des templates permet de modéliser le fonctionnement correct des systèmes comprenant une combinaison des comportements concurrents qui sont à la fois à instances uniques et instances multiples. Cette approche a été utilisée en ligne dans la cadre de diagnostic des défauts afin de confirmer le bon fonctionnement du système ou le contraire. En effet, les templates sont capables de représenter des langages qui ne pouvaient pas être représentés ou surveillés en utilisant seulement des automates temporisés.

2.4.4 Approches à base de Chroniques

C'est une approche de diagnostic des systèmes à événements discret qui exploite des informations temporelles concernant l'occurrence des événements dans le système [Milne et al., 1994] [Bousson et al., 1993]. Dans cette approche, une chronique est définie par un ensemble de motifs d'événements qui sont liés par des contraintes temporelles [Bousson et al.,

1993]. Les chroniques sont conçues pour représenter les évolutions possibles du comportement du système pour chaque type de défauts. En conséquence, cela implique d'avoir une connaissance à priori concernant tout type de défauts qui peuvent avoir lieu dans le système. Une approche de diagnostic similaire à l'utilisation de *Chroniques* est également retrouvées dans [Saddem et al., 2012] où les auteurs utilisent des modèles modulaires à base de Signatures Temporelles Causales (STC).

Exemple 2.6. Prenons un exemple tiré de [Dousson, 2002] représentant un comportement d'un système traduit par une occurrence ordonnée d'événements {*a*,*b*,*c*,*d*}. L'événement *a* précède l'occurrence de l'événement *c* avec de 2 à 5 unités de temps. Ensuite, l'événement *b* aura lieu après l'occurrence de deux événement *a* et *c*. l'événement *a* se produira pour une deuxième fois dans moins de 10 unités de temps après le début. Nous avons également une contrainte temporelle supplémentaire [1, 6] entre l'occurrence de l'événement *c* et *b*. la chronique représentant ce comportement du système est donnée dans la Figure 2.9.



(a). l'ordre partiel (induit) entre les événements de la chronique

(b). La chronique représentant un comportement du système.

Figure 2.9. Exemple d'une *Chronique*.

Dans les approches à base de chroniques, un système de reconnaissance de chroniques a été développé dans [Milne et al., 1994] (Figure.2.10). Ce système de reconnaissance prend en charge l'analyse le flux d'événements qui reçoit à son entrée et procède à l'identification, en temps réel, tout motif correspondant à une situation décrite par une chronique. Le système de reconnaissance est basé sur la prévision des dates possibles pour l'occurrence de chaque événement qui n'a pas encore eu lieu. Cet ensemble, appelée fenêtre temporelle, est réduite par la propagation des dates d'occurrences des événements observés à travers cette fenêtre temporelle. Lorsqu'un nouvel événement arrive dans le flux d'entrées, de nouvelles instances de chroniques sont générées et une mise à jour de l'ensemble des hypothèses est faite. Ensuite,

l'ensemble des hypothèses est organisé d'une façon arborescente. Les instances sont rejetées dès que possible lorsque les contraintes sont violées ou lorsque la fenêtre temporelle devient vide.

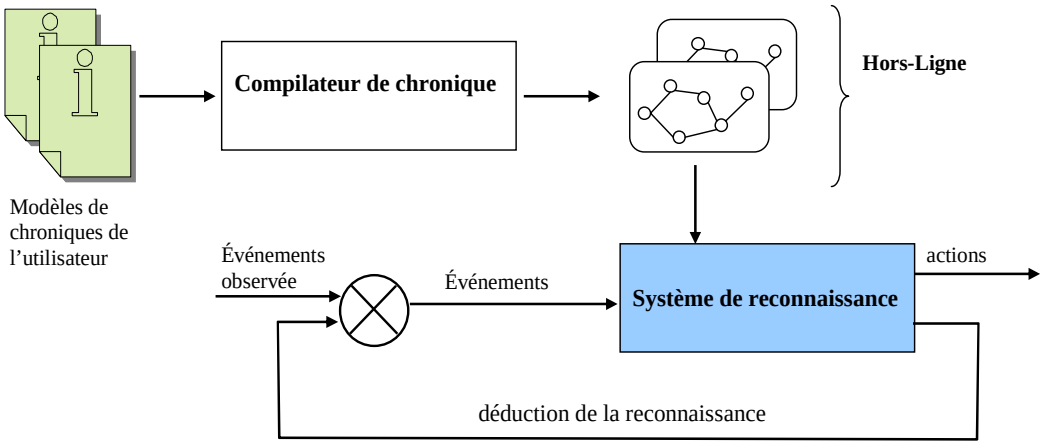


Figure 2.10. Principe géneral de reconnaissance de chroniques.

Exemple 2.7. Reprenons le comportement du système décrit dans l'exemple 2.6. Supposons que le superviseur reçoit la séquence d'événements suivante accompagnée des dates de leurs occurrences:

{(e_1 : a, t=4), (e_2 : d, t=5), (e_3 : a, t=6), (e_4 : c, t=8), (e_5 : b, t=10), (e_6 : e, t=11), (e_7 : a, t=12), (e_2 : b, t=14)}.

La chronique représentée dans la figure 2.9 a été reconnu à trois reprises: les cas appariés sont {e_1, e_4, e_5, e_7}, {e_3, e_4, e_5, e_7} et {e_3, e_4, e_7, e_8}.

2.5 Structure du diagnostic

2.5.1 Structure centralisée

Dans les approches qui utilisent ce type de structure (Figure 2.11), le système est représenté par un modèle global associé à un diagnostiqueur unique. Celui-ci collecte des observations et prend alors une décision finale au sujet des défauts qui apparaissent dans le système à diagnostiquer [Sampath et al., 1995] [Sampath et al., 1996(a)] [Zad et al., 1998]. Cependant, les approches se basant sur la construction du diagnostiqueur selon cette architecture souffrent de quelques inconvénients et notamment celui de la composition des

sous-modèles de composants. En effet, afin d'obtenir un modèle global du système, un produit cartésien des états représentant chaque modèle de composants est nécessaire. Par conséquent, cela peut nous conduire vers un problème d'explosion combinatoire en termes de taille de l'espace d'états du modèle global du système. Cette problématique limite considérablement l'applicabilité des approches utilisant cette architecture dans le cas des systèmes complexes à grande taille.

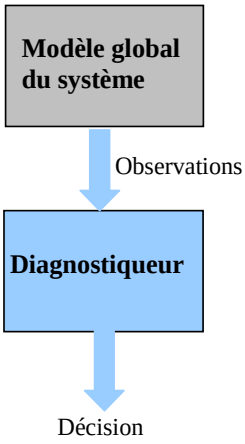
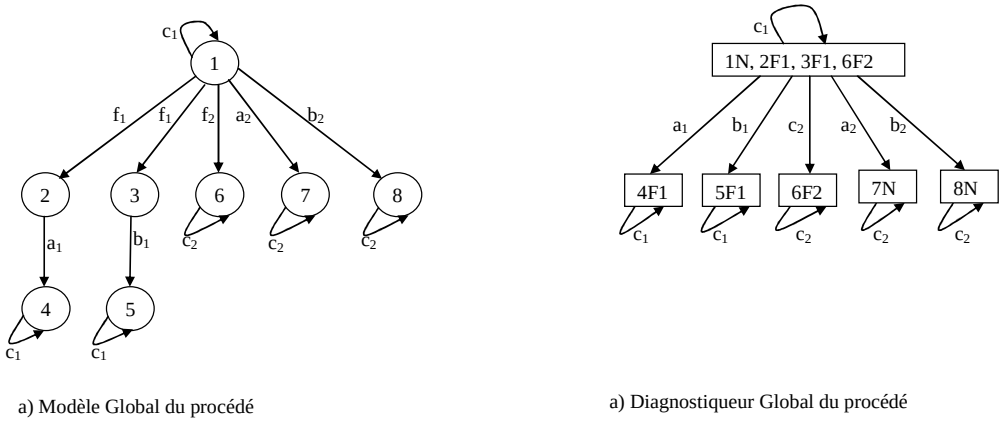


Figure 2.11. Structure de diagnostic centralisé.

Exemple 2.8. Considérons un exemple tiré de [Wang et al., 2005(a)] d'un procédé représenté par son modèle global *G* (Figure 2.12(a)). Afin de pouvoir construire le diagnostiqueur global G_d du procédé, il faut définir les partitions de défaut $\Pi_{F1} = \{f_1\}$ et $\Pi_{F2} = \{f_2\}$. Ce diagnostiqueur est représenté par la figure 2.12(b). A partir de l'état 1 du modèle du procédé, l'état initial (1N, 2F1, 3F1, 6F2) du diagnostiqueur est construit. En effet, dans cette état le diagnostiqueur peut indiquer que le procédé est soit dans l'état 1 en fonctionnement normale, soit dans l'état 2 ou 3 avec un défaut de type F1 ou également dans l'état 6 avec un défaut de type F2. De l'état initial, le procédé va dans les états du fonctionnement normal {7N}, {8N} avec l'observation des événements respectivement a_1 et b_1. Le diagnostiqueur global peut identifier et localiser les deux types de défauts F1 et F2 en observant les événements a_1 et b_1 pour le défaut F1 et l'événement c_2 pour le défaut F2. Dans le cas d'occurrence d'un défaut de type F1 peut amener le procédé dans l'état {7F1} ou {5F1}, tandis que l'occurrence d'un défaut de type F2 peut conduire le procédé dans l'état de défaut {6F2}.



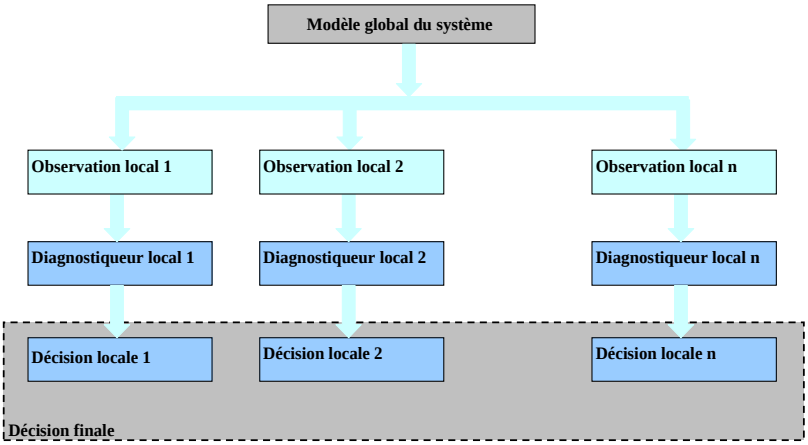
a) Modèle Global du procédé

a) Diagnostiqueur Global du procédé

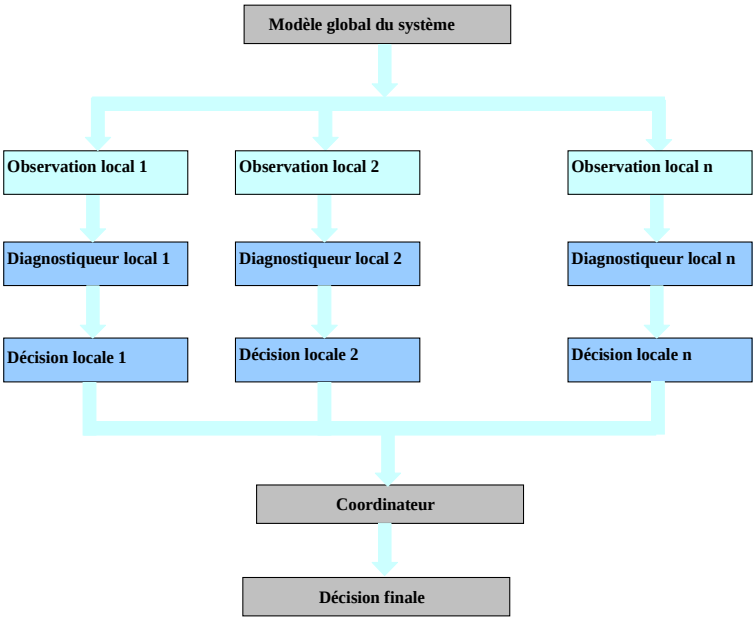
Figure 2.12. Modéle du système et son diagnostiqueur global.

2.5.2 Structure décentralisée

Le premier objectif de cette structure de diagnostic est de compenser l'explosion combinatoire issue d'une structure centralisée [Heiming et al., 1997] [Barrett et Lafortune, 2000] [Su et Wonham, 2000]. L'architecture de diagnostic décentralisée est cependant devenue nécessaire afin de pouvoir diagnostiquer des défauts dans les systèmes où l'information est décentralisée de nature. Dans cette architecture, le système est divisé en plusieurs parties locales appelées « sites ». Chaque site possède ses propres capteurs qui communiquent leurs données à un diagnostiqueur local dont il est associé. Les diagnostiqueurs locaux sont construits à partir du modèle global du système par une projection naturelle de ce dernier sur les événements observables concernant chaque site [Wang et al., 2005(a)]. Dans l'architecture décentralisée, les diagnostiqueurs locaux opèrent de manière indépendante, sans avoir des échanges entre eux. En effet, chaque diagnostiqueur fonde sa décision, concernant l'état du site qu'il observe, en se basant sur les observations locales envoyées par ses capteurs associés. Ensuite, une décision finale concernant l'état global du système peut être obtenue selon deux façons différentes : sans coordinateur (Figure 2.13(a)) ou avec coordinateur (Figure 2.13(b)).



(a).Diagnostic décentralisé sans coordinateur.



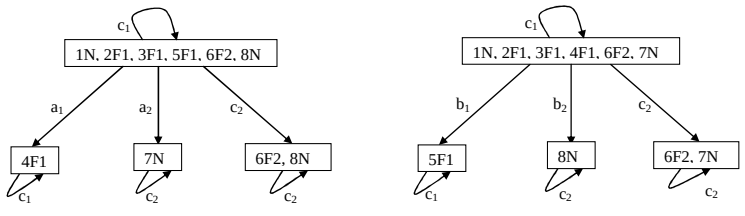
(b).Diagnostic décentralisé avec coordinateur.

Figure 2.13. Structure de diagnostic décentralisé.

Dans le cas d'une architecture décentralisée sans coordinateur, le résultat global du diagnostic est considéré comme l'ensemble des décisions de tous les diagnostiqueurs locaux. En conséquence, cette structure de diagnostic souffre souvent de problème d'ambigüité par rapport à l'occurrence de certain type de défauts. Ce problème est dû principalement au fait de non-communication en ligne entre les décisions des différents diagnostiqueurs locaux ainsi qu'à leur observabilité partielle. Afin de pouvoir diminuer le nombre d'état d'ambigüité, une approche de diagnostic décentralisée avec coordinateur a été proposée [Debouk, 2000]. Dans ce type d'approche, le résultat final du diagnostic est issu via un centre d'information, appelé

coordinateur, qui orchestre l'ensemble de diagnostiqueurs locaux. En général, un coordinateur reçoit une partie ou la totalité des informations de chaque diagnostiqueur et la traite selon plusieurs protocoles [Debouk, 2000].

Exemple 2.9. Reprenons le même procédé de l'exemple 2.8. Nous souhaitons construire un diagnostiqueur selon une approche décentralisée. Le modèle du procédé, comme dans l'approche décentralisée, caractérise à la fois le comportement normal N mais aussi le comportement anormal pour des défauts de type F1 et F2. A partir du modèle global du procédé (Figure 2.12(a)), il est possible d'établir deux diagnostiqueurs locaux G_{d1} et G_{d2}. Chacun de ces diagnostiqueurs possède ses propres observations : $\Sigma_{o1} = \{a_1, a_2, c_1, c_2\}$ pour G_{d1} et $\Sigma_{o2} = \{b_1, b_2, c_1, c_2\}$ pour le second diagnostiqueur local G_{d2}. Ces deux sous-ensemble des observations appartiennent à l'ensemble globale du procédé donné par $\Sigma_o = \{a_1, a_2, b_1, b_2, c_1, c_2\}$. Egalement, Le modèle G possède deux partitions de défaut $\Pi_{F1} = \{f_1\}$ et $\Pi_{F2} = \{f_2\}$. Les diagnostiqueurs locaux obtenus sont représentés en figure 2.14. La construction de G_{d1} et G_{d2} s'effectue sur le même principe. En effet, il suffit de décrire la manière selon laquelle G_{d1} a été obtenu qui sera la même pour avoir G_{d2}. Dans un premier temps, le procédé G est supposé normal dans son état initial 1, alors il est inclus dans l'état initial du G_{d1} avec l'étiquette N. Il en résulte 1N. L'état initial de G_{d1} comporte également tous les états du G atteignables par les événements non observables par G_{d1}. Ces états sont accompagnés par leurs étiquettes correspondantes. Par conséquent, nous aurons 2F1, 3F1, 5F1, 6F2, 8N dus à l'occurrence de, respectivement, f_1, f_1, b_1, f_2, b_2. Alors, l'état initial du G_{d1} est représenté par {1N, 2F1, 3F1, 5F1, 6F2, 8N}. Ensuite, à partir de cet état initial, nous prendrons en considération tous les états atteignables par l'occurrence d'un événement observable par G_{d1}. Ainsi, avec l'occurrence de a_1, G_{d1} atteint le seul état 4 du G avec l'étiquette F1. C'est un état certain où G_{d1} peut décider avec certitude de l'occurrence de f_1. De la même manière, on construit le reste des états de G_{d1}.



a) Diagnostiqueur Local G_{d1} b) Diagnostiqueur Local G_{d2}

Figure 2.14. Diagnostiqueurs locaux du procédé *G*.

Chaque diagnostiqueur prend une décision locale. Cependant, les diagnostiqueurs locaux doivent fournir au final les mêmes performances qu'un diagnostiqueur global. L'observation de l'événement c_2 sur les deux diagnostiqueurs engendre une indécision sur le procédé. En effet, après l'occurrence de c_2, G_{d1} retourne un état incertain {6F2, 8N} et G_{d2} un état {6F2, 7N} également incertain. Dès lors, il est impossible de prendre une décision sur le comportement du procédé. Si l'on regarde le diagnostiqueur global (Figure 2.12(b)) du procédé de la figure 2.12, il en ressort une identification de tous les défauts et aucun état incertain. En effet, l'événement c_2 génère un défaut de type F2 sur l'état 6 du procédé. C'est pourquoi, un coordinateur de décisions locales doit être ajouté à la structure décentralisée afin de résoudre le problème d'indécision de l'occurrence du défaut de type F2.

L'obtention d'un coordinateur s'effectue par l'étude du modèle global du procédé par un expert. Il émet des priorités sur les décisions des diagnostiqueurs afin de lever les cas d'indécision. Le Tableau 2.1 représente un coordinateur simple pour le cas de deux diagnostiqueurs locaux (Figure 2.14). Ce coordinateur est utilisé pour lever le problème d'indécision sur l'occurrence du défaut de type F2.

Règle :	Décision de G_{d1}	Décision de G_{d2}	Décision Globale
1	N, F1, F2	N, F1, F2	Indécision
2	F1	N, F1, F2	F1
3	N	N, F1, F2	N
4	N, F1, F2	F1	F1
5	N, F1, F2	N	N
6	F2, N	N	N
7	N	F2, N	N
8	F2, N	F2, N	F2

TABLEAU 2.1. TABLE DE DECISION DU COORDINATEUR DE L'EXEMPLE 2.8

2.5.3 Structure distribuée

Dans ce type de structure (Figure 2.15), le système est constitué de plusieurs composants locaux associés à plusieurs diagnostiqueurs locaux. Chacun de ces diagnostiqueurs reçoit des observations à partir d'un composant local spécifique dont il est responsable. Dans cette structure, contrairement aux deux structures précédentes, chaque composant est représenté par un modèle local. Par conséquent, les résultats locaux de diagnostic sont obtenus grâce à un module ou canal de communication qui permet l'échange

d'informations entre les différents diagnostiqueurs locaux. Dans la littérature, nous trouvons ceux qu'ont été présentés par [Pencolé et Cordier, 2005] [Pencolé et al., 2001]. Dans ces travaux, Pencolé et Cordier se sont confrontés au problème du diagnostic des réseaux de télécommunications. Tout d'abord, un réseau de télécommunication est spatialement distribuée et implique un grand nombre de composants, donc la construction d'un modèle global pour ce type de systèmes n'est pas pratique. D'autre part, une propagation des défauts d'un composant vers un autre apparaît comme une conséquence évidente qui doit être prise en compte. La propagation des défauts dans le système a deux conséquences majeures:

1. Un grand nombre d'observations sont envoyées vers le module de diagnostic et,
2. Une interférence entre l'occurrence de plusieurs défauts peut empêcher le diagnostiqueur de les détecter et ensuite de les localiser.

Une approche de diagnostic entièrement distribuée à base de diagnostiqueurs locaux a été proposée dans [Pencolé et Cordier, 2005]. Comme dans [Sampath et al., 1995] [Sampath et al., 1996(b)], des modèles sont construites pour chaque composant élémentaire du système à l'aide des automates à états finis. Cependant, au lieu de construire un modèle global pour tout le système, les auteurs ont décomposé le système en un ensemble de sous-systèmes. Ensuite, ils ont construit un modèle pour chaque sous-système (ce qui implique un nombre limité de composants). Le modèle d'un sous-système est issu d'une composition synchrone uniquement des modèles des composants de ce sous-système et il n'est pas nécessaire de construire un modèle global du système. Pour chaque sous-système, un diagnostiqueur est compilé hors-ligne afin de diagnostiquer uniquement le sous-système auquel il est associé. Toutefois, étant donné que les sous-systèmes peuvent échanger les événements en interne, les diagnostiqueurs locaux ne sont donc pas totalement indépendants les uns des autres. Dans ce cas, un module de communication entre les différents diagnostiqueurs locaux est intégré au module de diagnostic afin de fournir un diagnostic global du système.

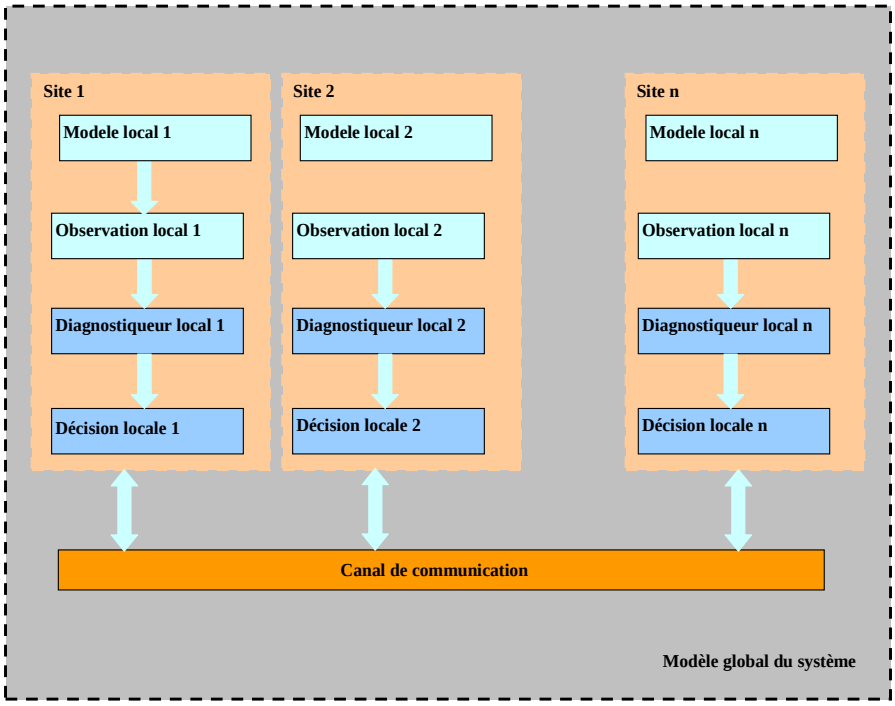


Figure 2.15. Structure de diagnostic distribué.

2.5.4 Choix d'une structure

Nous venons de voir les différentes structures de prise de la décision dans les approches de diagnostic issues de la littérature. Cette étude permet de justifier nos choix dans l'approche de diagnostic que nous allons développer.

Le Tableau 2.2 rappelle les différentes caractéristiques qu'ont été représentées pour chaque structure et fait ressortir, en grisé sur le tableau, tous les points qui nous intéressent.

	Centralisée	Décentralisée Sans Coordinateur	Décentralisée Avec Coordinateur	Distribuée
Modèle du procédé	Global	Global	Global	Local
Diagnostiqueur	Global	Local	Local	Local
Communication	Aucune	Aucune	Coordinateur	Entre modules
Décision	Globale	Locale	Locale	Locale
Incohérence	Aucune	Importante	Faible	Aucune
Complexité du Protocole	Aucune	Aucune	Faible	Importante

TABLEAU 2.2. ETUDE COMPARATIVE DES DIFFERENTES STRUCTURES

Le choix d'une structure de décision pour le diagnostic s'effectue essentiellement selon le procédé et l'information disponible. Dans le cadre de notre étude, nous souhaitons nous focaliser sur les systèmes dont ses différents composants fournissent une information décentralisée.

Concernant la structure centralisée pour le diagnostic des systèmes à événement discrets, malgré qu'elle soit efficace au niveau de la prise de décision (absence d'incohérence), cette structure est soumise au problème de l'explosion combinatoire. Pour les approches utilisant une structure décentralisée, leur inconvénient est le fait qu'elles se basent sur un modèle global du procédé afin de construire les diagnostiqueurs locaux. Cela implique également un risque d'explosion combinatoire dès la modélisation du système. En outre, lorsque cette structure est utilisée sans coordinateur, elle peut souffrir d'un problème d'incohérence lors de la prise de décision. Cependant, un coordinateur peut être utilisé afin de diminuer cette incohérence. Ce coordinateur peut être enrichi afin d'exprimer uniquement les spécificités globales du procédé, non exprimées par les diagnostiqueurs locaux, sans pour autant décrire l'ensemble de son comportement. Les approches de diagnostic qui adoptent pour une structure distribuée paraissent les plus souhaitables à utiliser. Cependant, ils impliquent dans ce cas que le système soit distribué par nature. Un inconvénient évident est donc l'établissement d'un protocole de communication. Celui-ci peut alors s'avérer complexe à mettre en place et surtout à implémenter. En effet, des délais de communication peuvent avoir lieu suite à cette complexité.

L'approche décentralisée que nous allons proposer dans le chapitre suivant consiste à construire différents diagnostiqueurs locaux pour des sous-systèmes indépendants. Cette construction sera faite de manière totalement modulaire. En fait, chaque diagnostiqueur local sera renforcé par des conditions d'autorisations ainsi que des fonctions booléennes, caractérisant l'occurrence des événements dans chaque sous-système correspondant, afin de l'aider à prendre une décision avec un minimum d'incohérence. C'est dans cette optique que nous souhaitons développer notre approche afin d'adapter la structure décentralisée et la rendre moins sensible au problème d'explosion combinatoire au niveau du modèle du procédé.

2.6 Notion de diagnosticabilité

L'utilisation d'approches pour le diagnostic est indispensable pour les systèmes plus ou moins complexes. Cependant, une question reste en suspens : le système est-il diagnosticable ? En effet, avant d'appliquer une méthode sur un système, il faut pouvoir vérifier si ce dernier dispose d'informations en quantité suffisante pour pouvoir effectuer le diagnostic. Par conséquent, la notion de « diagnosticabilité » va permettre de déterminer l'ensemble des pannes pouvant être diagnostiqué selon. Un SED est dit diagnosticable pour un ensemble de partitions de défauts et pour un ensemble d'événements observables s'il est possible de détecter l'occurrence de n'importe quel défaut appartenant à une des partitions de défaut dans un délai fini [Contant et al., 2004].

Cependant, cette notion est très dépendante de l'approche utilisée. En effet, selon la structure (centralisée, décentralisée, distribuée) et l'information disponible, des extensions de la notion de diagnosticabilité ont été définies dans la littérature pour les SEDs. Elle s'applique autant pour les méthodes à base d'états que celles à base d'événements. Il convient alors de supposer que pour un même système, une approche de diagnostic permet de détecter et localiser un défaut alors que d'autres non. Cette notion devient alors un indicateur de performance intéressant.

Par exemple, [Sampath, 1995] définie une notion de diagnosticabilité issue d'une approche de diagnostic centralisée à base d'événements. L'auteur y détermine deux conditions afin de satisfaire cette notion sur le fait (i) qu'il existe au moins un état du diagnostiqueur pour lequel le diagnostiqueur décide avec certitude l'occurrence d'un défaut appartenant à une partition et (ii) qu'il ne doit pas y avoir de cycles dits " indéterminés" pour lesquels le diagnostiqueur est incapable de décider avec certitude l'occurrence d'un défaut appartenant à une partition.

Dans [Wang et al., 2005(b)], une notion de co-diagnosticabilité est établie pour les structures décentralisées. Elle permet d'assurer le fait que toute défaillance doit être diagnostiquée dans un délai borné par au moins un diagnostiqueur local en utilisant ses propres observations. Concernant les structures distribuées, une diagnosticabilité collaborative (Joint-Diagnosability) est retrouvée dans [Qiu, 2005]. C'est une extension de la co-

diagnosticabilité puisqu'elle se base sur les informations locales de chaque diagnostiqueur mais également sur les informations des diagnostiqueurs voisins.

2.7 Conclusion

Dans ce deuxième chapitre de thèse, nous avons présenté une brève étude concernant les approches de diagnostic destinées aux systèmes à événements discrets. Dans cette représentation, une classification a été faite selon certains critères et paramètres spécifiques (Figure 2.16). En premier temps, le critère qui a été choisi pour répertorier les différentes méthodes de diagnostic est le type de connaissance à exploiter pour construire le module de diagnostic. Selon ce critère les approches ont été classifiées en deux catégories principales, approches sans modèle et approches à base de modèle(s). Dans la première catégorie, on retrouve les approches dont le diagnostic se fait sans utilisation d'un modèle représentant le fonctionnement du système. Dans ces approches, L'absence d'un modèle du système est due soit à la complexité du système, soit à l'indisponibilité d'informations nécessaires pour le construire. En conséquence, ces approches se basent sur l'utilisation des donnée issues de l'historique de fonctionnement du système ou fournies par un expert. Dans la deuxième catégorie selon le critère de modélisation, on trouve les approches à base de modèle(s) représentant à travers un outil de représentation plus ou moins abstrait le comportement du système.

La classification a été faite ensuite en prenant en considération l'outil utilisé pour modéliser le système ainsi que la construction du diagnostiqueur. Nous avons choisi de développer notamment les approches à base d'automates à états finis, de réseaux de Petri, de templates et finalement de chroniques. Les structures de diagnostic ont été ensuite représentées en partant d'une structure de diagnostic centralisée en arrivant à celle distribuée en exprimant finalement les avantages et inconvénients de chacun. Ce chapitre se termine par l'expression d'une notion de diagnosticabilité permettant de garantir ou non l'applicabilité d'une approche.

Dans le chapitre suivant, nous allons nous appuyer sur ce qui vient d'être décrit auparavant afin de proposer notre contribution à une approche décentralisée de diagnostic pour les Systèmes à Evénements Discrets.

Figure 2.16. Critères de classification des méthodes de diagnostic des SEDs.

Chapitre 3
Diagnostic des SEDs par distributions de probabilité

3.1 INTRODUCTION 71
3.2 CONTEXTE ET OBJECTIFS 71
3.3 DESCRIPTION GENERALE DE L'APPROCHE 72
3.4 CONSTRUCTION DU MODULE DE DIAGNOSTIC (HORS LIGNE) 76
3.4.1 Décomposition du système et Définition des contraintes 76
3.4.1.1 Principe de décomposition du système 76
3.4.1.2 Apprentissage des séquences d'événements 78
3.4.2 Construction des Templates et Chroniques 79
3.4.3 Construction des modules de diagnostic 84
3.4.3.1 Caractérisation de l'occurrence d'événements par distribution de probabilité 84
3.4.3.2 Construction des conditions d'autorisation et des fonctions de non-occurrence 86
3.5 DEPLOIEMENT DU DIAGNOSTIC (EN LIGNE) 90
3.5.1 Procédures de détection des défauts 94
3.5.2 Procédure de localisation des défauts 97
3.6 CONCLUSION 99

Chapitre 3 : Diagnostic des SEDs par distributions de probabilité

3.1 Introduction

Dans le chapitre précédent, un état de l'art concernant les principales méthodes de diagnostic des systèmes à événements discrets a été présenté. Dans cet état de l'art, nous avons mis l'accent sur certaines méthodes à base de modèles. Dans ce contexte, nous nous sommes intéressés aux méthodes de diagnostic qui utilise les automates à états finis comme outil de représentation. Un intérêt plus particulier a été porté aux deux méthodes sur lesquelles est basée notre approche de diagnostic à proposer : Les « approches à base des templates » et les « approches à base des chroniques ».

Dans le présent chapitre, une approche de diagnostic des défauts dans les systèmes à événements discrets (SEDs) va être illustrée. Ce travail est basé sur la construction d'un modèle temporel représentant les comportements normal et défaillant du système à diagnostiquer. Ces modèles sont construits hors ligne en exploitant les différents signaux d'entrées/sorties du système. Afin d'enrichir ces modèles et les rendre plus efficaces, des informations de type temporelle sont intégrées dans ces dernier sous forme de contraintes. Ces contraintes temporelles expriment les dates d'occurrence des événements dans le système à travers un intervalle de temps. Par la suite, ces contraintes sont associées à des *distributions de probabilité* (*DP*) afin de caractériser la probabilité d'occurrence de chaque événement dans l'intervalle de temps à qu'il appartient. Ainsi, les distributions de probabilités vont être utilisées afin de confirmer ou infirmer le fonctionnement normal ou défaillant du système.

3.2 Contexte et objectifs

Notre travail de thèse vise à concevoir un module de diagnostic et l'intégrer dans un système modélisé en tant que SED. Ce module est construit d'une manière décentralisée après avoir décomposé le système global en sous-systèmes indépendants. Cette décentralisation du module de diagnostic est dû au fait que beaucoup de systèmes sont décentralisés par nature, notamment la majorité des systèmes complexes (réseaux de communication, de puissance et les systèmes manufacturiers, etc.). En effet, le choix d'une architecture décentralisée a été fait pour affronter la complexité des systèmes et leur distribution que ce soit sur le plan géographique (physique) ou sur le plan informationnelle. Pour ces raisons, il devient nécessaire de diviser le diagnostiqueur en plusieurs sites.

Dans le présent manuscrit, le diagnostiqueur consiste à déterminer s'il y a un défaut dans le système ou non. Dans le cas d'une présence de défaut, celui-ci doit être détecté, puis localisé. En fait, ce diagnostiqueur est basées sur l'utilisation d'un modèle du fonctionnement nominal (désiré) G_N du système et/ou un modèle des fonctionnements défaillants G_{F_i} en réponse à un défaut spécifié $f \in \Pi_{F_i} \subseteq \Sigma_f = \{\Pi_{F_1}, ..., \Pi_{F_d}\}$, où Π_{F_i} est la partition de défauts associés à la faute du label F_i.

Le travail présenté dans ce chapitre est basé sur deux méthodes particulières pour réaliser le diagnostic : les *Chroniques* [Pencolé et Subias, 2009] [Cordier et al., 2007] [Guerraz et Dousson, 2004] et les *Templates* [Pandalai et Hollaoway, 2000]. L'intérêt porté à ces deux approches est dû à leur prise en compte des informations temporelles dans le modèle du système. Les *Templates* et les *Chroniques* sont construites par apprentissage et leur reconnaissance n'est pas une tâche triviale.

Une ***Template*** consiste, par définition, en un événement déclencheur suivi par un ensemble de conséquences. Elle représente un mode de fonctionnement normal.

Une ***Chronique*** est considérée comme un ensemble d'événements observables dont des contraintes temporelles leur sont associées. Une *Chronique* représente un mode de fonctionnement en cas de présence d'un défaut.

Dans la présente approche, le modèle logique du système est établi hors ligne. Ensuite, les *Templates* sont utilisés en ligne pour confirmer le bon fonctionnement du système. Les *Chroniques* quant à elles identifient un défaut appartenant à une partition de défaut par une reconnaissance de signature en ligne.

3.3 Description générale de l'approche

Cette partie est consacrée à la présentation du cadre général de notre approche de diagnostic. La mise en œuvre de cette approche comprend deux phases principales :

1. La première phase est la construction hors ligne du module de diagnostic (Figure 3.1). Cette phase s'effectue en trois étapes :

- *Décomposition du système en sous-systèmes indépendants.* Cette décomposition est basée sur l'exploitation des signaux, événements, d'entrées/sorties échangés entre le procédé et le contrôleur. Des sous-systèmes sont identifiés à partir d'une notion d'indépendance. Par la suite, un apprentissage, par enregistrement des occurrences des événements observables, est effectué afin de définir un modèle logique pour chaque sous-système.

- *Construction des Templates et Chroniques.* A partir des modèles logiques, un apprentissage de contraintes temporelles selon la date d'occurrence des événements est effectué tout d'abord en fonctionnement normal. Ceci permet de construire un modèle de *Template* G_N pour chaque sous-système. De là, une simulation du fonctionnement du système est réalisée pour chaque type de défauts que l'on souhaite diagnostiquer. Cette simulation de scénarii, réalisée sous *Matlab*, permet de définir un modèle de *Chronique* pour chaque type de défauts. Il convient donc de définir au préalable les partitions de défauts à diagnostiquer. Les modes de fonctionnement normal et défaillant du système à diagnostiquer sont donc représentés par des automates temporisés.

- *Construction des diagnostiqueurs*. Afin d'apporter plus d'informations aux modèles de l'étape précédente, les contraintes temporelles sont exprimées par des distributions de probabilité (*DPs*). Ces distributions sont construites par apprentissage. Par ailleurs, afin de diagnostiquer aussi bien les défauts pouvant avoir lieu dans les capteurs et actionneurs Tout ou Rien que certains défauts process, des conditions ainsi que des fonctions caractérisant respectivement l'occurrence et la non-occurrence des événements sont construites. Ces conditions d'autorisation d'occurrences sont définies pour chaque nœud (état) des *Templates* et *Chroniques* alors que les fonctions de non-occurrence d'événements sont présentes uniquement dans les *Templates*. Elles retournent une valeur booléenne qui aidera par la suite en la localisation en ligne du défaut. Il en résulte en ensemble de modules de diagnostic $D_i = \{D_{Ni}, D_{Fj}\}$ où j représentant le nombre de défaut à diagnostiquer pour chaque sous-système SS_i.

2. La deuxième phase de notre approche est destinée au déploiement en ligne du module de diagnostic également en 3 étapes (Figure 3.2) :

- *Implémentation*. Chaque module de diagnostic D_i de sous-système est implémenté pour évolution parallèle. L'évolution d'un nœud à un autre se faisant uniquement par l'occurrence des événements observables en respectant les contraintes temporelles entre les différents nœuds.

- *Détection*. La détection d'un défaut peut s'effectuer de différentes manières :
  - Le non-respect d'une condition d'autorisation conduisant à un flag de détection (dans le cas par exemple d'un événement non attendu).
  - La violation d'une contrainte de non-occurrence lorsqu'un événement est attendu.
  - La dégradation d'un composant par la probabilité faible d'un événement par rapport à la distribution de probabilité de fonctionnement normal.

- *Localisation*. A travers l'ensemble des informations issus de l'étape de détection, et selon le ou les modèles affectés, le défaut peut être localisé et/ou fournir une information sur la dérive de fonctionnement du système à l'opérateur. Pour cela, un indicateur de dégradation « I » représentant la dérive de la distribution de probabilité du fonctionnement normal lors de l'occurrence d'un événement est calculé.

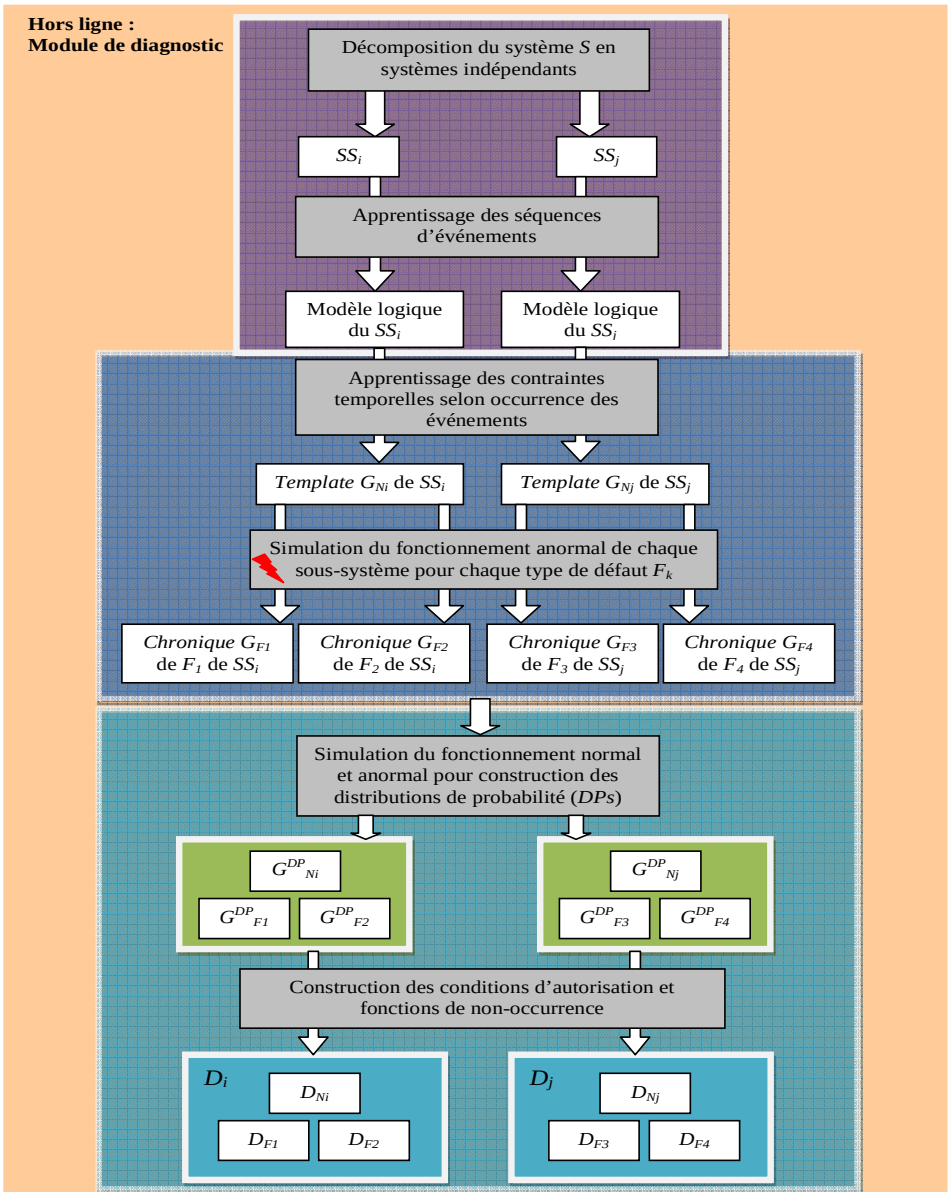


Figure 3.1. Construction du module de diagnostic

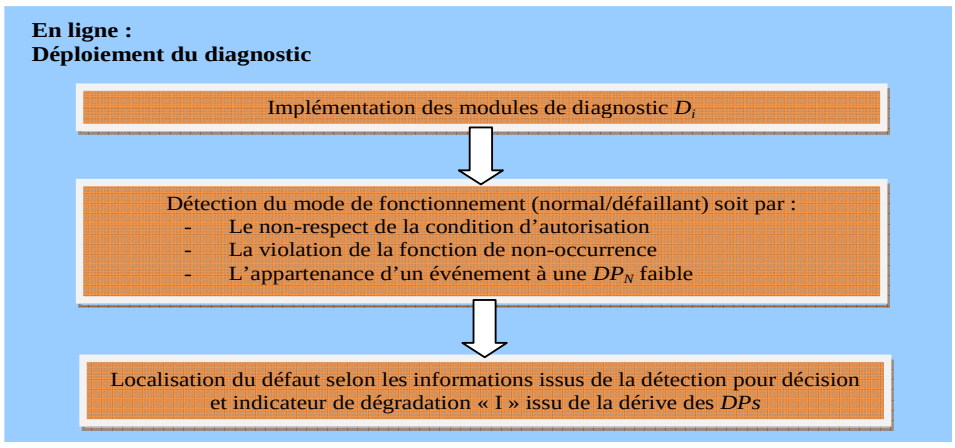


Figure 3.2. Construction du module de diagnostic

Dans la suite de ce paragraphe, chaque étape va être détaillée plus en profondeur.

3.4 Construction du module de diagnostic (hors ligne)

La première phase de l'approche proposée vise à construire hors ligne les diagnostiqueurs à travers un apprentissage des contraintes temporelles.

3.4.1 Décomposition du système et Définition des contraintes

Les systèmes complexes sont généralement très difficiles à manipuler lorsque nous les traitons en un seul bloc. Par conséquent, les tâches consistant à leur modélisation et leur diagnostic deviennent compliqués voire impossibles à réaliser. De ce fait, une décomposition du système global en sous-systèmes est nécessaire. Cette décomposition permet une analyse souvent plus simple du système. Cette section représente une description de la méthodologie de décomposition d'un système à événements discrets SEDs. La décomposition consiste à obtenir des sous-systèmes indépendants au niveau événementiel et géographique. L'interaction produit-procédée n'est pas prise en considération dans cette indépendance.

3.4.1.1 Principe de décomposition du système

Afin de pouvoir illustrer la méthodologie de la décomposition dans notre travail, nous considérons la structure d'un SEDs de la figure 3.3. Il se compose de deux parties : la partie opérative et son contrôleur. Le contrôleur représente la logique de fonctionnement du processus devant respecter des spécifications fixées par le cahier des charges. Il envoi des ordres vers la partie opérative et reçoit en retours les différentes informations des capteurs. La partie opérative traduit le comportement mécanique du système au travers d'actions qu'elle effectue selon les ordres reçus par le contrôleur. Elle accomplie les tâches qui lui sont assignées et envoie un compte rendu de ses actions effectuées via ses capteurs.

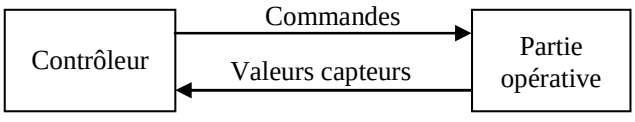


Figure 3.3. Structure d'un SED à decomposer

Dans un SED, l'ensemble des événements observables Σ_o d'un système est subdivisé en deux sous-ensembles $\Sigma_c \cup \Sigma_{uc}$ représentants respectivement l'ensemble des événements contrôlables (ordres envoyés par le contrôleur), et l'ensemble des événements non contrôlables (valeurs des capteurs).

Dans ce travail, nous avons choisi l'interprétation de S. Balemi [Balemi et al., 1993]. De ce fait, l'état d'une commande *x*, ou une valeur d'un capteur *y*, est exprimé par une variable booléenne. Un événement contrôlable correspond à l'activation, $\forall e \in \Sigma_c$: $e = \uparrow x$, ou à la désactivation, $\forall e \in \Sigma_c$: $e = \downarrow x$, d'une commande *x* envoyée par le contrôleur, tandis qu'un événement non contrôlable est associé soit à un front montant, $\forall e \in \Sigma_{uc}$: $e = \uparrow y$, soit à un front descendant, $\forall e \in \Sigma_{uc}$: $e = \downarrow y$, d'une entrée *y* du contrôleur. Par conséquent, les ensembles Σ_c et Σ_{uc} sont écrits comme suit : $\Sigma_c = \uparrow X \cup \downarrow X$ et $\Sigma_{uc} = \uparrow Y \cup \downarrow Y$, où $\uparrow X$ and $\downarrow X$ correspondent à l'activation et la désactivation de l'ensemble de toutes les commandes $X = \{x_1, x_2, \ldots, x_m\}$ du contrôleur, et $\uparrow Y$ and $\downarrow Y$ reflètent, respectivement, le front montant et le front descendant du changement d'état des capteurs $Y = \{y_1, y_2, \ldots, y_l\}$.

L'activation d'une commande peut produire un changement dans l'état du système caractérisé par le changement des valeurs de capteurs. Ainsi, après l'activation d'une commande, on peut s'attendre à l'occurrence d'événements corrélés dans certains intervalles de temps.

La décomposition d'un SED en *n* sous-systèmes indépendants $SS_i \in \{1,\ldots,n\}$ est obtenue en observant les signaux de commande envoyés vers les actionneurs qui constituent ce système et les valeurs capteurs corrélés à cette action. Une partie du système est considérée comme un sous-système SS_i lorsque les actionneurs qui la constituent réagissent à certains signaux de commande envoyés par le contrôleur provoquant une réaction sur un certain nombre de capteurs [Malki et al., 2009]. Par conséquence, chaque sous-système SS_i a ses propres sous-ensembles des événements contrôlables $\Sigma_{ic} \subseteq \Sigma_c$ et sous-ensemble des événements non contrôlables $\Sigma_{iuc} \subseteq \Sigma_{uc}$ où $\Sigma_c = \Sigma_{1c} \cup \Sigma_{2c} \cup \ldots \cup \Sigma_{nc}$ et $\Sigma_{uc} = \Sigma_{1uc} \cup \Sigma_{2uc} \cup \ldots \cup \Sigma_{nuc}$.

Définition 3.1. (Notion d'indépendance). Considérons deux sous-systèmes SS_i et SS_j issus d'une décomposition d'un système global *S* (figure 3.4). Les deux sous-systèmes sont dit indépendants si et seulement si la condition suivante est vérifiée :

$$\forall SS_i, SS_j, i, j \in \{1,\ldots,n\}, i \neq j : \begin{cases} \Sigma_{ic} \cap \Sigma_{jc} = \phi \\ \quad \wedge \\ \Sigma_{iuc} \cap \Sigma_{juc} = \phi \end{cases} \tag{3.1}$$

Dans notre approche, la propriété d'indépendance s'appuie sur les hypothèses suivantes :

- L'occurrence d'un défaut dans un sous-système ne peut pas être propagé aux autres sous-systèmes, ce qui signifie que les sous-systèmes sont complètement découplés les uns des autres.
- L'occurrence d'un défaut dans un sous-système peut être diagnostiquée (détectée et isolée) en se basant uniquement sur l'observation des événements liés à ce sous-système. En d'autres termes, le défaut est diagnostiqué en utilisant seulement le modèle de ce sous-système. Ainsi, le diagnostic d'un défaut ne nécessite pas de communication entre sous-systèmes ou de modèle global du système.

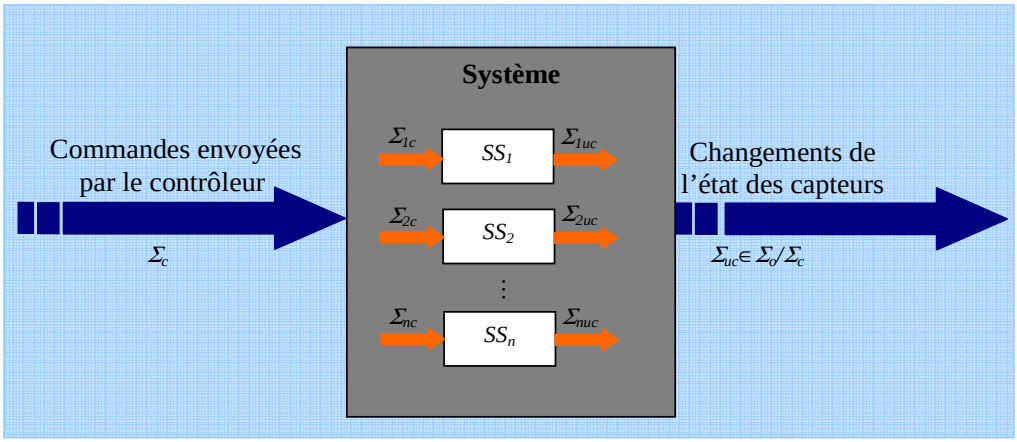


Figure 3.4. Décomposition du système

3.4.1.2 Apprentissage des séquences d'événements

Cette tâche consiste à considérer le fonctionnement du système comme normal durant une phase d'apprentissage du cycle de fonctionnement du système. Pour cela, un enregistrement des différentes variables du sous-système est effectué sous la forme d'un chronogramme. Celui-ci est ensuite traduit sous la forme d'un modèle à états où chaque changement de valeur d'une variable permet au modèle d'évoluer d'un état à un autre (figure 3.5). Une recherche de signature répétitive est effectuée dans ce chronogramme afin de ne pas oublier de séquences d'événements ou de ne pas créer d'états redondants. Le modèle obtenu est exclusivement logique.

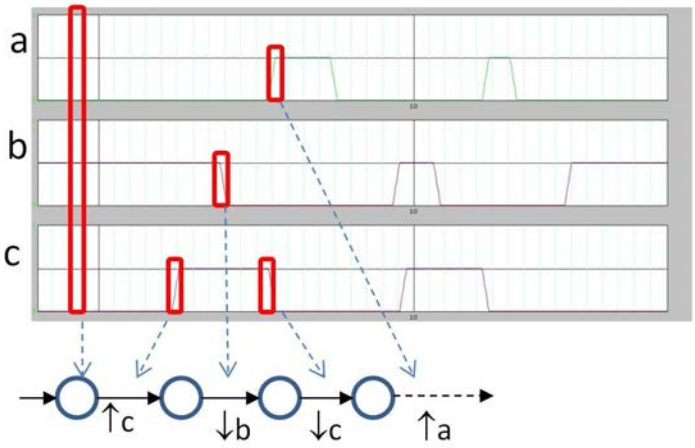


Figure 3.5. Construction du modèle logique

3.4.2 Construction des *Templates* et *Chroniques*

Une tâche intermédiaire avant la construction des diagnostiqueurs consiste en la modélisation des observateurs des sous-systèmes. Pour cela, nous avons choisi le formalisme des automates temporisés. Ce choix est justifié par la généralité qu'offrent les automates temporisés. En effet, leur capacité à représenter les contraintes temporelles caractérisant les dates d'occurrence des événements et leur pouvoir d'expression permettent une grande capacité d'analyse. Pour la construction de ceux-ci, le modèle logique des sous-systèmes est utilisé pour le séquencement des événements et transformé en *Templates* et *Chroniques* à l'aide de contraintes temporelles [Malki et Sayed-Mouchaweh, 2011(a)].

Un sous-système SS_i est représenté par un modèle temporel qui représente les séquences d'événements générées par le système à diagnostiquer ainsi que les différentes contraintes temporelles entre ces événements [Ghallab, 1996]. Les labels des événements sont associés aux nœuds du graphe, tandis que les contraintes temporelles sont associées aux arcs. Une contrainte temporelle est représentée par un intervalle du temps [a, b]. Un arc reliant un événement e_j à un événement e_i, à laquelle un intervalle [a, b] est associée, indique que l'événement e_j doit se produire dans l'intervalle temporel [a, b] après l'occurrence de l'événement e_i.

Definition 3.2. Un graphe temporel *G* est un 5-uplet (Q, q_0, *C*, Σ, Tr) dont :

- Q est l'ensemble fini des états (ou de localités) contenant des événements,
- $q_0 \in Q$ est l'état initial,

- C est un ensemble fini de contraintes,
- Σ est un alphabet fini décrivant l'ensemble des événements,
- $Tr \subseteq Q \times C \times \Sigma \times Q$ est un ensemble fini de transitions. Une transition de q vers q' est de la forme $tr=<q, c, q'> \in Tr$ où $c=[a, b] \in C$ est la contrainte temporelle de la transition, a est le temps minimum et b le temps maximum nécessaire à l'occurrence de l'événement labelé de l'état q' après l'occurrence de l'événement labelé de l'état q.

Exemple 3.1. La figure 3.6 montre un exemple d'un modèle temporel représenté par un graphe temporel. Ce modèle représente la séquence $e_1e_2e_3e_4e_5$ tandis que les contraintes temporelles entre les événements sont représentées par les intervalles de temps $c_1 = [a_1, b_1]$, $c_2 = [a_2, b_2]$, $c_3 = [a_3, b_3]$, $c_4 = [a_4, b_4]$ et $c_5 = [a_5, b_5]$. Ces contraintes définissent les dates d'occurrence pour chaque événement à partir de la date d'occurrence de l'événement qui le précède. Ainsi, l'événement e_2 est attendu dans l'intervalle c_1 après l'occurrence de l'événement e_1.

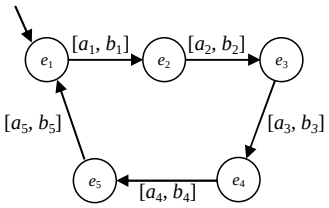


Figure 3.6. Exemple d'un modèle temporel

Le modèle temporel peut représenter le comportement désiré G_N ou défaillant G_F du système. Le comportement desiré est modelisé à travers une *Template* alors que le comportement défaillant est modélisé via des *Chroniques*.

Une *Template* est utilisée comme un graphe temporel correspondant à l'occurrence d'une séquence d'événements contrôlables et/ou non contrôlables comme conséquence à un événement généré par le contrôleur (activation ou désactivation d'une commande x).

Une *Chronique* est également représentée par un graphe temporel correspendant à une séquence spécifique qui permet l'identification d'un type de défaut qui a eu lieu dans le système. Une *Chronique* est donc considérée comme la signature d'un défaut F_i.

La construction de ces modèles est faite de la manière suivante :

- Détermination du nombre de nœuds. Dans un système cyclique, ce nombre correspond aux activations et désactivations de commandes ainsi que le nombre des fronts montants et descendants générés pendant un cycle de fonctionnement dans le système à diagnostiquer. Dans notre cas, il est issu des modèles logiques de l'étape précédente.
- Construction des arcs reliant les différents nœuds,
- Définition des contraintes temporelles representées par des intervalles de temps $[t_{min}, t_{max}]$ dans lequel les événements peuvent avoir lieu. Les dates d'occurrence des événements qui appartiennent à ces intervalles sont calculées à partir de l'instant d'activation de la commande qui a entraîné leurs occurrences. En outre, ces contraintes sont définies pour les événements observables et non contrôlables.

Cet apprentissage est à effectuer en deux temps. :

1. Un premier temps pour l'obtention du modèle de *Template* G_{Ni} dans le cas d'un fonctionnement normal pour chaque sous-système SS_i (en situation réelle sous hypothèse d'absence de défauts ou en simulation du système).
2. Un second temps pour la simulation de scénarii de défauts où pour chaque scénario, une signature est obtenu à travers des contraintes temporelles spécifiques au défaut. Il en résulte un modèle de *Chronique* G_{Fj} pour chaque scénario, donc pour chaque type de défaut F_j simulé.

Dans ces travaux, nous considérerons les types de défauts suivants :

- Information capteur TOR bloqué-on (le capteur voit l'information « 1 » alors qu'il ne devrait pas),
- Information capteur TOR bloqué-off (le capteur voit l'information « 0 » alors qu'il ne devrait pas),
- Actionneur TOR bloqué-on (l'actionneur continu d'être actif alors que la commande ne lui demande pas),
- Actionneur TOR bloqué-off (l'actionneur continu d'être inactif alors que la commande lui demande de l'être),

- Défaut sur le process (présence d'une fuite, canalisation bouchée, …) [Malki et Sayed-Mouchaweh, 2011(b)].

Exemple 3.2. Considerons un exemple explicatif d'une porte automatisée pour un garage (figure 3.7). La porte dispose de quatre capteurs discrets. Les deux premiers capteurs sont liés à la porte pour indiquer son ouverture, capteur « Po », et sa fermeture, capteur « Pf ». Les deux autres capteurs « Pv_1 » et « Pv_2 » indiquent la présence d'une voiture, respectivement, à l'extérieur, à l'intérieur du garage. La porte est équipée d'un moteur M piloté en bistable par l'ordre « O » pour l'ouverture et « F » pour la fermeture. Deux boutons poussoirs intérieurs « Bpi » et extérieur « Bpe » sont également présents pour les demandes d'ouverture. Fonctionnellement, la porte se ferme suite au passage d'un véhicule de l'intérieur vers l'extérieur par la désactivation du capteur « Pv_1 » ou par le passage de l'extérieur vers l'intérieur par la désactivation du capteur « Pv_2 ».

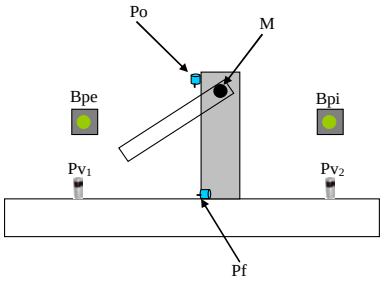


Figure 3.7. Porte de garage automatisée

Afin de pouvoir modéliser le comportement desiré de ce système sous forme de *Template*, il est necessaire de déterminer l'ensemble de commandes des actionneurs X={O, F} et l'ensemble des capteurs Y={Pf, Po, Pv_1, Pv_2}. L'activation et la désactivation des commandes envoyées par le contrôleur sont représentées respectivement par les deux ensembles $\uparrow X=\{\uparrow O, \uparrow F\}$ et $\downarrow X=\{\downarrow O, \downarrow F\}$. L'ensemble des événements contrôlables est donc $\Sigma_c=\uparrow X\cup\downarrow X=\{\uparrow O, \uparrow F, \downarrow O, \downarrow F\}$. Les changements des capteurs sont donnés par $\uparrow Y=\{\uparrow Pf, \uparrow Po, \uparrow Pv_1, \uparrow Pv_2\}$et $\downarrow Y=\{\downarrow Pf, \downarrow Po, \downarrow Pv_1, \downarrow Pv_2\}$. Ces deux ensembles forment un ensemble des événements non contrôlables $\Sigma_{uc}=\uparrow Y\cup\downarrow Y=\{\uparrow Pf, \downarrow Pf, \uparrow Po, \downarrow Po, \uparrow Pv_1, \downarrow Pv_1, \uparrow Pv_2, \downarrow Pv_2\}$. L'ensemble des événements observables s'écrit alors $\Sigma_o=\Sigma_c\cup\Sigma_{uc}=\{\uparrow O, \downarrow O, \uparrow F, \downarrow F, \uparrow Pf, \downarrow Pf, \uparrow Po, \downarrow Po, \uparrow Pv_1, \downarrow Pv_1, \uparrow Pv_2, \downarrow Pv_2\}$.

Ce système peut être décomposé en plusieurs sous-systèmes. On peut remarquer par exemple que l'ensemble {porte – capteurs de fin de course porte} est indépendant des boutons ou des capteurs de présence voiture.

Pour simplifier les explications, nous considérons pour cet exemple un fonctionnement simple consistant à demander l'ouverture de la porte jusque son fin de course, puis à fermer la porte jusque Pf.

De là, et à travers l'apprentissage des contraintes temporelles lors d'un fonctionnement normal (Tableau 3.1), il est possible d'obtenir la *Template* de la figure 3.8. Les dates d'occurrence sont calculées à partir de l'instant t_0 de l'activation (ou désactivation) d'une commande.

Commandes envoyées	Evénements conséquents	Contraintes temporelles en fonctionnement normal
↑O	↓Pf	c1 = $[t_{min}^{\downarrow Pf}, t_{max}^{\downarrow Pf}]$
	↑Po	c2 = $[t_{min}^{\uparrow Po}, t_{max}^{\uparrow Po}]$
	↓O	c3 = $[t_{min}^{\downarrow O}, t_{max}^{\downarrow O}]$
↓O	↑F	c4 = $[t_{min}^{\uparrow F}, t_{max}^{\uparrow F}]$
↑F	↓Po	c5 = $[t_{min}^{\downarrow Po}, t_{max}^{\downarrow Po}]$
	↑Pf	c6 = $[t_{min}^{\uparrow Pf}, t_{max}^{\uparrow Pf}]$
	↓F	c7 = $[t_{min}^{\downarrow F}, t_{max}^{\downarrow F}]$
↓F	↑O	ϕ - sur demande du conducteur

TABLEAU 3.1. LES CONTRAINTES TEMPORELLES DU COMPORTEMENT NORMAL POUR LA PORTE DE GARAGE

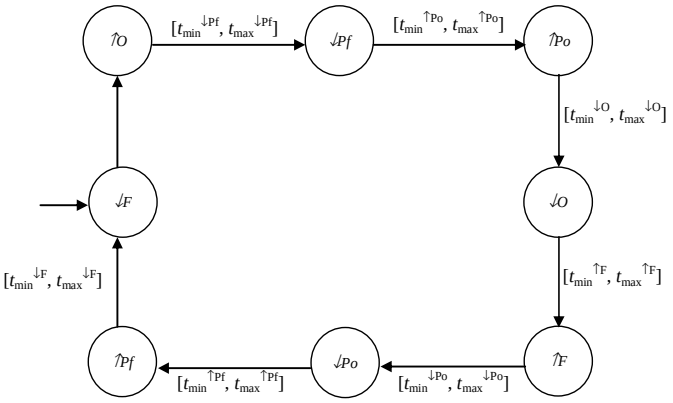


Figure 3.8. Exemple d'une *Template* pour la porte de garage

Le comportement défaillant du système est représenté par des *Chroniques* pouvant exprimer une violation des contraintes temporelles caractérisant les dates d'occurrence des événements. Ainsi, elles se manifestent comme des séquences particulières d'événements liés à des types bien définis de défauts. Par conséquent, leur représentation est similaire à la *Template* de fonctionnement normal mais avec des intervalles différents. Pour cet exemple, nous verrons un peu plus loin dans ce chapitre la représentation d'une *Chronique* pour un type de défaut complétée des informations de distributions de probabilité, de conditions d'autorisation et de fonctions de non-occurrence.

Remarque : Cet apprentissage peut également être effectué comme une analyse de classification à l'aide par exemple l'outil de classification SALSA (Situation Assessment using LAMDA claSsification Analysis) développé au sein du LAAS de Toulouse [Kempowsky et al., 2003]. Cet outil identifie les situations du système en s'appuyant sur la méthode de classification floue avec apprentissage LAMDA (Learning Algorithm for Multivariable Data Analysis).

3.4.3 Construction des modules de diagnostic

Dans cette section, l'objectif est de rendre les modèles de *Templates* et de *Chroniques* construits dans la tâche précédente plus représentatifs à travers l'attribution de distributions de probabilité (*DPs*). Le but de l'utilisation de ces distributions de probabilité est d'apporter plus d'informations aux modèles construits. Ensuite, des conditions d'autorisation ainsi que des fonctions booléennes de non-occurrence d'événements sont intégrées afin d'obtenir pour chaque sous-système un module de diagnostic enrichi.

3.4.3.1 Caractérisation de l'occurrence d'événements par distribution de probabilité

Après avoir enregistré les dates d'occurrence de différents événements et avoir construit les contraintes temporelles, une amélioration de ces données peut être faite en intégrant l'utilisation des distributions de probabilité. Ces distributions vont représenter chaque contrainte temporelle dans les modèles construits afin d'exprimer la possibilité d'occurrence d'un événement à une date précise.

Comme nous avons vu dans les sections précédentes, une contrainte temporelle pour un événement *e* est représentée par un intervalle du temps de type $[t_{min}^{e}, t_{max}^{e}]$. La construction d'une distribution de probabilité pour cette contrainte est réalisée en se basant sur l'utilisation

d'un histogramme des probabilités (Figure 3.9). Cet histogramme est limité par deux valeurs, minimale t_{min}^{e} et maximale t_{max}^{e}, qui correspondent, également, aux valeurs minimale et maximale de la contrainte temporelle. L'histogramme est devisé en « *h* » barres qui ont des largeurs égales. Le nombre *h* d'un histogramme est déterminé expérimentalement. La largeur Δ d'une barre d'histogramme est définie comme suit:

$$\Delta = \frac{(t^{e}_{\max} - t^{e}_{\min})}{h} \tag{3.2}$$

L'hauteur de chaque barre $b_k, k \in \{1,2,..,h\}$ correspond au nombre de fois n_b qu'un événement a eu lieu dans cette barre. La distribution de probabilité $\{ p(y_{b_k}), k \in \{1,2,..,h\} \}$ est obtenu en divisant la hauteur de chaque barre par le nombre total de fois *N* que cet événement a eu lieu dans l'intervalle du temps [t_{min}^{e}, t_{max}^{e}]. Ces probabilités sont affectées aux centres des barres $y_{b_k}, k \in \{1,2,..,h\}$. En effet, une distribution de probabilité est donnée comme suit:

$$p(y_{b_k}) = n_{b_k} / N \tag{3.3}$$

La fonction de distribution de probabilité (*DP*) est obtenue en reliant les centres des hauteurs de barres.

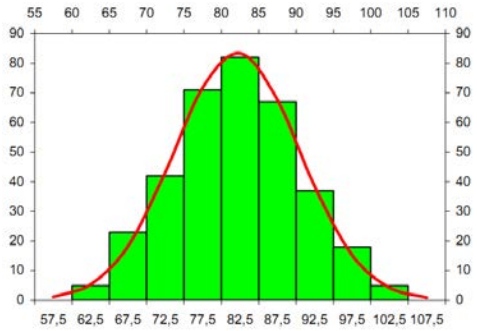
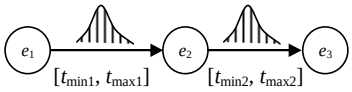
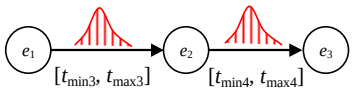


Figure 3.9. Exemple de distrubution de probabilité d'un événement *e*.

Les distributions de probabilité sont construites afin de caractériser la possibilité d'occurrence d'un événement e_j après l'occurrence de l'événement e_i. Elle est utilisée pour confirmer un fonctionnement normal ou identifier un fonctionnement défaillant avec un degré de certitude (valeur de probabilité). Il en résulte pour chaque *Template* et *Chronique*, un modèle enrichi $G^{DP}{}_{Ni}$, $G^{DP}{}_{Fj}$ (Figure 3.10).



(a) : G^{DP}_{Ni} représentant le fonctionnement normal



(b) : G^{DP}_{Fj} représentant le fonctionnement défaillant en présence d'un défaut de type F_j

Figure 3.10. Exemple d'une *Template* et *Chronique* associés à des *DPs*

3.4.3.2 Construction des conditions d'autorisation et des fonctions de non-occurrence

Les nœuds dans un modèle temporel représentent les séquences d'événements qui peuvent avoir lieu dans le sous-système. Dans un modèle temporel, l'activation (ou la désactivation) d'une commande $x_i \in$ X entraîne des changements dans les sorties du système. Ces changements sont représentés par les lectures effectuées par des capteurs appartenant à l'ensemble « Y ». En effet, un changement dans la lecture d'un capteur $y_p \in Y$ génère un événement de type $\{\uparrow y_p, \downarrow y_p\}$. Cependant, il est tout à fait possible qu'un événement non prévu se réalise au même titre que la non-occurrence d'un événement attendu ait lieu. Pour faire face à ces deux cas, le module de diagnostic peut être enrichi par deux outils qui consistent à caractériser l'occurrence d'événements dans le système à diagnostiquer [Malki et Sayed-Mouchaweh, 2011(a)] :

- les conditions d'autorisation d'occurrence d'événements et
- la fonction booléenne caractérisant la non-occurrence d'événements.

☞ ***Conditions d'autorisation***

Considérons un système *S* qui change son état suite à des commandes envoyées par le contrôleur vers ses actionneurs. Chacune de ces commandes est représentée par une variable booléenne $\{x_1, x_2, \ldots, x_m\} \in X$ décrivant l'état de la commande correspondante. Ainsi, les changements dans l'état du système sont mesurés via des capteurs. L'état de ces capteurs est représenté par des variables booléennes $\{y_1, \ldots, y_k\} \in Y$ décrivant leur état (on/off). L'activation (ou désactivation) d'une commande x_i influence un ensemble de capteurs. Le changement d'état d'un capteur y_p appartenant à cet ensemble produit l'événement $e_p \in \{\uparrow y_p, \downarrow y_p\} \subset \Sigma_{uc}$. Par conséquent, l'état actuel du système peut être représenté par un vecteur booléen défini par :

$$\left(BV_{x_i}^{y_1,\ldots,y_k}\right)_{present} = (y_1,\ldots,y_k) \tag{3.4}$$

L'occurrence d'un événement e_p dans le système conduit à un changement de son état actuel représenté par le vecteur $\left(BV_{x_i}^{y_1,\ldots,y_k}\right)$. Le changement dans ce vecteur peut être déduis en utilisant un vecteur de déplacement $DV_{x_i}^{e_p} = (dv_{y_1},\ldots,dv_{y_p},\ldots,dv_{y_k})$. Lorsque l'élément dv_{y_p} de ce vecteur est égal à 1, alors l'occurrence de l'événement e_p entraîne un changement dans l'état de la variable y_p. De là, le nouveau vecteur booléens $\left(BV_{x_i}^{y_1,\ldots,y_k}\right)_{nouveau}$ estimant l'état futur du système après l'occurrence de l'événement e_p est calculé comme suit:

$$\left(BV_{x_i}^{y_1,\ldots,y_k}\right)_{nouveau} = \left(BV_{x_i}^{y_1,\ldots,y_k}\right)_{present} \oplus DV_{x_i}^{e_p} \tag{3.5}$$

La construction des conditions d'autorisation pour chaque événement e_i se base essentiellement sur l'utilisation du vecteur $\left(BV_{x_i}^{y_1,\ldots,y_k}\right)_{present}$ représentant l'état actuel du système. Ces conditions sont définies en tant qu'une combinaison "AND" de l'état booléen actuel de l'ensemble de toutes les variables:

$$y_p, p \in \{1,2,..,k\}\ en_{x_i}^{e_i} = AND((BV_{x_i}^{y_1,\ldots,y_k})_{present}) \tag{3.6}$$

L'ensemble de ces conditions d'autorisation est présente dans les *Templates* afin de confirmer ou infirmer un comportement défaillant.

Exemple 3.3. Reprenons le système de la porte automatisée pour un garage.

Les conditions d'autorisation pour les différents événements sont données dans le Tableau 3.2. Par exemple, pour l'ouverture de la porte, lorsque l'ordre O est activé, l'événement non-contrôlable ↓Pf conséquent à cette ordre ne peut avoir lieu que si le capteur Pf est déjà à 1 et si le capteur Po est à 0.

Tâche à effectuer	Commande envoyée	Evénements conséquents	Condition d'autorisation
Ouverture de la porte	↑O	↓Pf	$en_1 = en\uparrow_O^{\downarrow Pf}$ = Pf . /Po
		↑Po	$en_2 = en\uparrow_O^{\uparrow Po}$ = /Pf . /Po
Fermeture de la porte	↑F	↓Po	$en_3 = en\uparrow_F^{\downarrow Po}$ = /Pf . Po
		↑Pf	$en_4 = en\uparrow_F^{\uparrow Pf}$ = /Pf . /Po

TABLEAU 3.2. CONDITIONS D'AUTORISATION POUR LES EVENEMENTS DANS LE SYSTEME DE LA PORTE

☞ ***Fonctions booléennes de non-occurrence***

La construction de cette fonction booléenne de non-occurrence est basée sur l'exploitation de la contrainte temporelle correspondant à un événement e_i attendu dans le fonctionnement normal. Elle doit donc être présente uniquement dans les *Templates*. En fait, la non-occurrence de cet événement va être caractérisée par une fonction $U_{x_i}^{e_i}$. Cette fonction prend la valeur « 1 » dans le cas de la non occurrence de l'événement e_i, sinon elle prend la valeur « 0 » (Figure 3.11). Formellement, cette fonction est définie comme suit:

$$\forall e_i \in \Sigma_{uc} : U_{x_i}^{e_i}(t) = \begin{cases} 1 & if\ t > t_{\max}^{e_i} + \Delta t^{e_i} \\ 0 & \text{sinon} \end{cases} \qquad (3.7)$$

Le terme $t_{\max}^{e_i}$ désigne le temps maximum prévu pour l'occurrence de e_i. Δt^{e_i} est une marge de sécurité, définie expérimentalement, pour laquelle nous pouvons confirmer la non-occurrence de l'événement e_i.

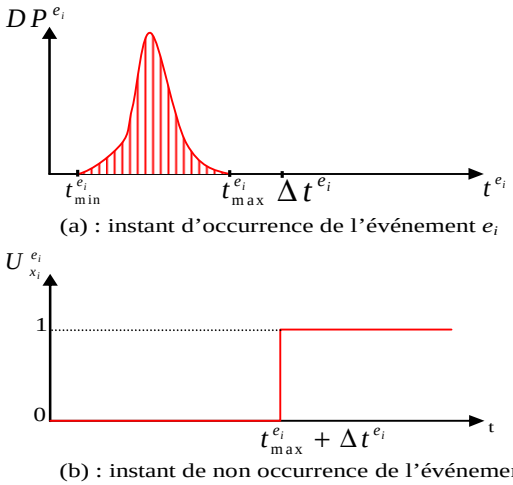


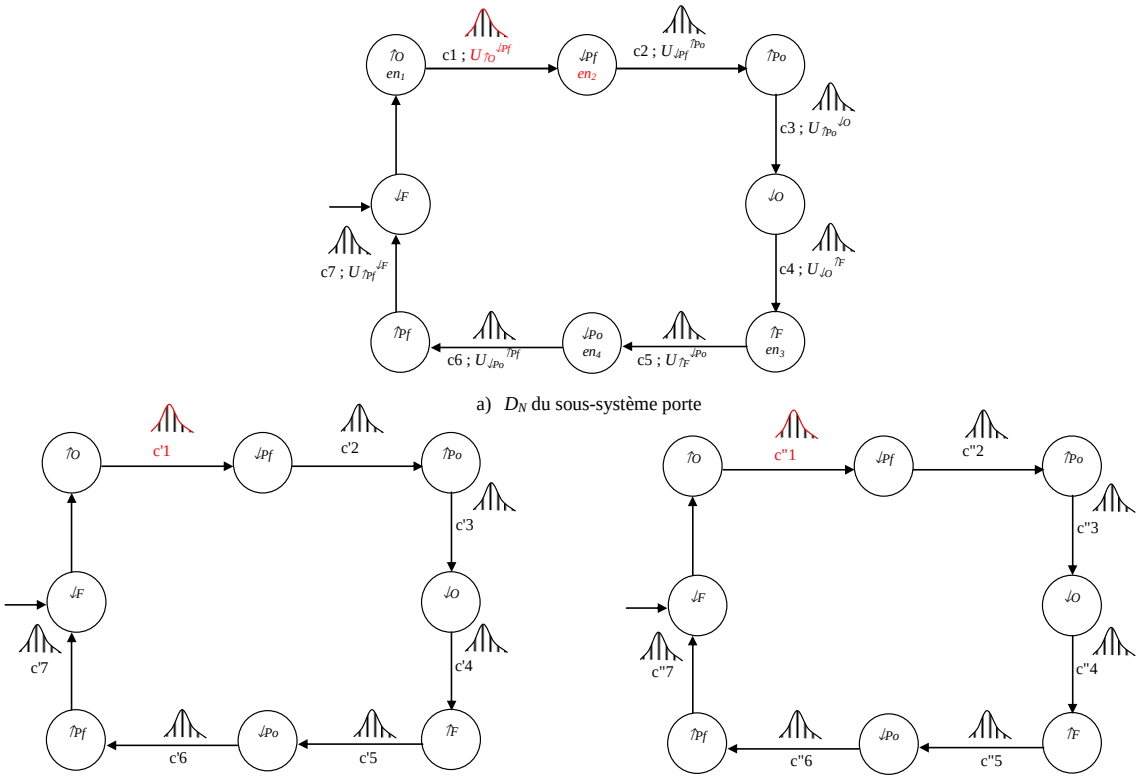
Figure 3.11. Fonction Booléene caracterisant la non-occurrence d'un événement e_i

Exemple 3.4. A travers les conditions d'autorisation et les fonctions de non-occurrence, reprenons l'exemple de la porte de garage. Trois types de défauts peuvent être considérés : (i) les défauts F_M sur le moteur M lorsque la porte est bloquée en position fermée ou ouverte, (ii) les défauts F_{Pf} sur le capteur Pf (bloqué-on ou bloqué-off) et (iii) les défauts F_{Po} sur le capteur Po (bloqué-on ou bloqué-off).

Dans le cas d'une séquence d'événement $\{\uparrow O \uparrow Po\}$ et que le capteur Pf est à 1 alors la condition $en_2 = en_{\uparrow O}^{\uparrow Po}$ = /Pf . /Po n'est pas respectée. Dans ce cas, le défaut F_{Po} montrant un événement non attendu sur Po devient un candidat à la défaillance. Il en est de même du défaut F_{Pf} sur le fait que $\downarrow Pf$ n'a pas eu lieu. Dans ce cas, c'est la fonction de non-occurrence $U_{\uparrow O}^{\downarrow Pf}$ qui est activée (Figure 3.12). La localisation précise du défaut se fera en ligne à l'aide des informations issues des contraintes temporelles (c'1 et c''1) des défauts concernés (Tableau 3.3), des distributions de probabilité et de l'indice de dégradation de la *Template* et des *Chroniques* de Pf et Po.

Commandes envoyées	**Evénements conséquents**	**Contraintes temporelles de D_N**	**Contraintes temporelles de D_{FPo}**	**Contraintes temporelles de D_{FPf}**
↑O	↓Pf	c1	c'1	c''1
	↑Po	c2	c'2	c''2
	↓O	c3	c'3	c''3
↓O	↑F	c4	c'4	c''4
↑F	↓Po	c5	c'5	c''5
	↑Pf	c6	c'6	c''6
	↓F	c7	c'7	c''7
↓F	↑O	ϕ	ϕ	ϕ

TABLEAU 3.3. CONTRAINTES TEMPORELLES POUR LA PORTE DE GARAGE



a) D_N du sous-système porte

b) D_{FPo} du sous-système porte pour le capteur Po

c) D_{FPf} du sous-système porte pour le capteur Pf

Figure 3.12. Module de diagnostic D_{Porte}

3.5 Déploiement du diagnostic (en ligne)

Ce paragraphe est à la phase en ligne de l'approche de diagnostic proposée. La mise en œuvre du module de diagnostic construit au préalable va permettre la détection et la localisation des défauts affectant le système.

La première étape du déploiement du module de diagnostic consiste à implémenter les modèles D_i issus des *Templates* et *Chroniques*. Cet ensemble récupère les informations échangées entre la PO et la PC afin de faire évoluer ses modèles de manière déterministe (Figure 3.13).

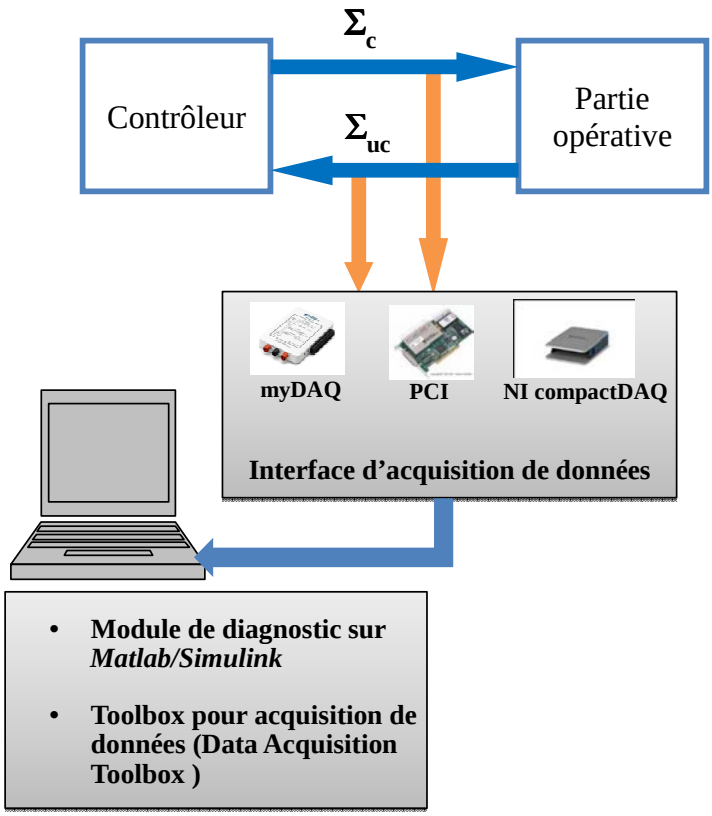


Figure 3.13. Implémentation du module de diagnostic

La deuxième étape qui consiste en la détection des défauts. Dans cette étape, le module de diagnostic surveille les conditions d'autorisation d'événements, les fonctions booléennes de non-occurrence, mais également d'un indice de dégradation exprimé en fonction de la date d'occurrence de l'événement par rapport à la distribution de probabilité calculée.

La troisième étape recherche certaines caractéristiques du défaut comme son instant d'apparition, son amplitude, sa gravité. Dans cette étape, la procédure de localisation fait appel aux conditions d'autorisation des événements ainsi qu'aux fonctions booléennes.

En retour du module de diagnostic, un état du système est communiqué à l'utilisateur afin que celui-ci prenne une décision concernant l'action à entreprendre sur le système. Il peut s'agir de maintenir le système dans le même mode opératoire, de corriger son fonctionnement ou encore de l'arrêter complètement selon la criticité du défaut comme par exemple celui défini dans [Vachtsevanos et al., 2006] (Figure 3. 14).

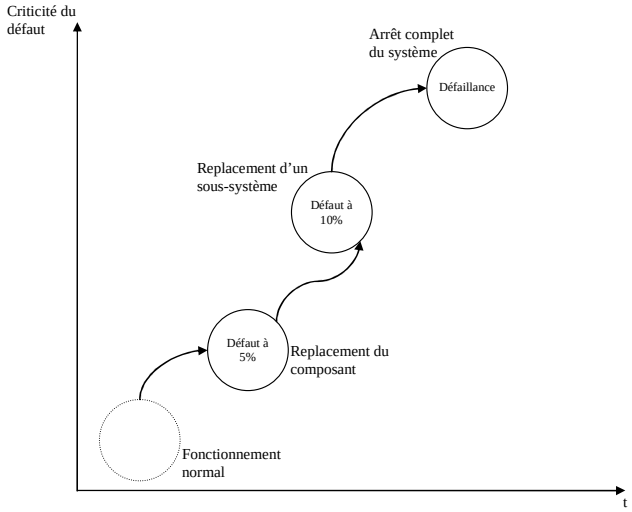


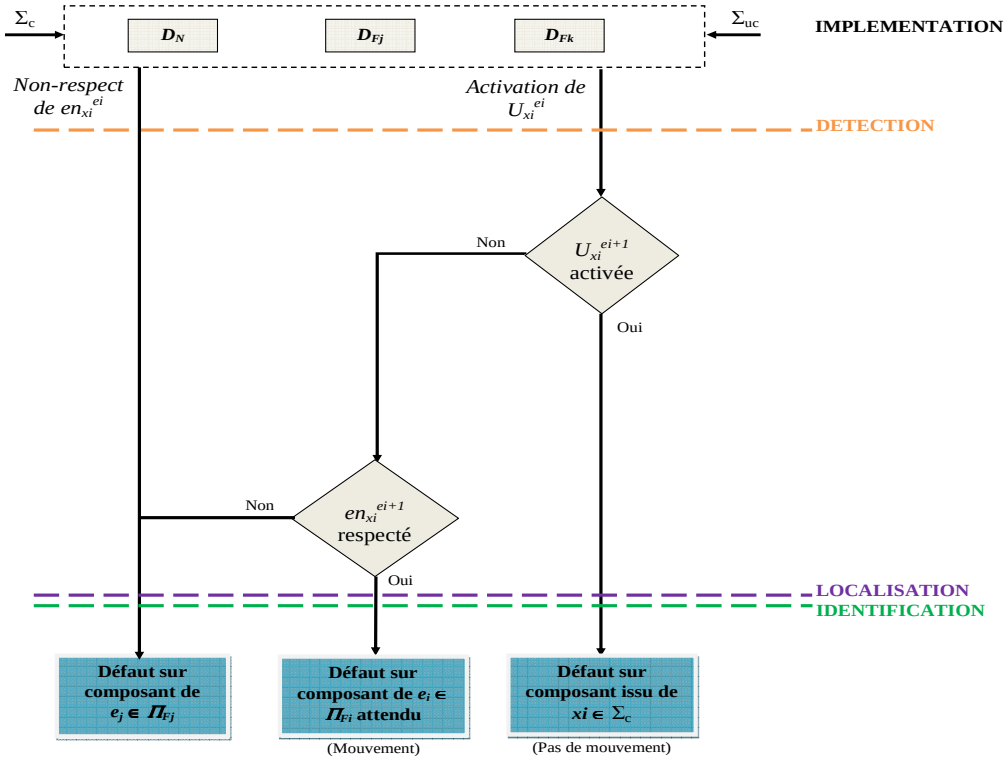
Figure 3.14. Actions de maintenance requise selon la criticité du defaut

La procédure de mise en œuvre en ligne du module de diagnostic est présentée en figure 3.15, elle se base sur l'hypothèse qu'un seul défaut sur chaque sous-système ne peut survenir. Pour chaque nouvel événement (ou non-événement) :

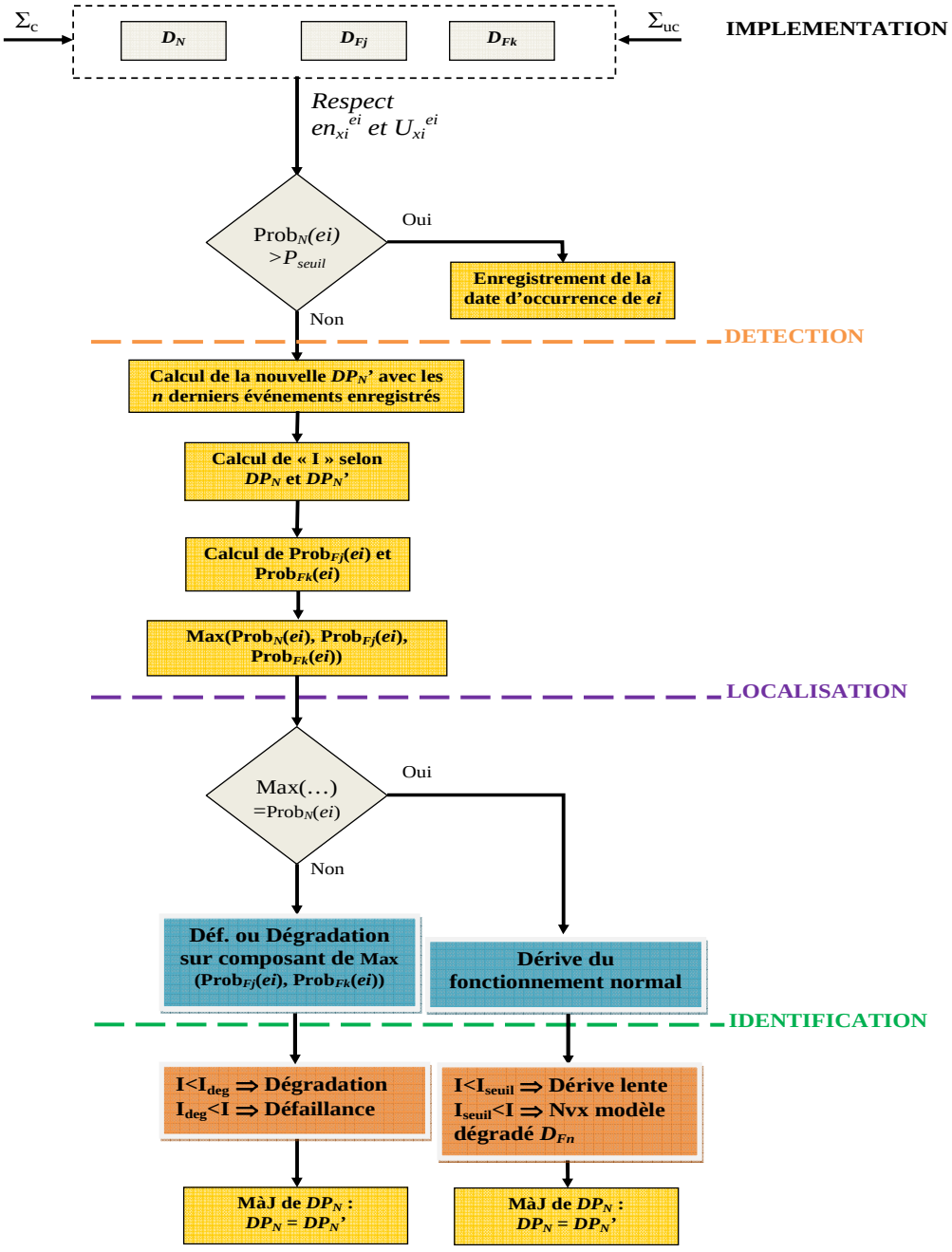
- Si un nouvel événement e_j non attendu survient, alors la condition d'autorisation du nœud actuel en_{xi}^{ei} n'est pas respectée et le défaut est aussitôt détecté et localisé sur le composant associé à cet événement.

- Si aucun événement n'est observé, alors la fonction de non-occurrence U_{xi}^{ei} est activée. Un défaut est donc détecté mais pas encore isolé. Il faut attendre l'observation ou non d'un second événement suite à l'ordre *xi* afin de localiser le défaut. Si la fonction de non-occurrence U_{xi}^{ei+1} est activée alors l'ordre *xi* envoyé n'a provoqué aucun mouvement et le défaut est donc localisé sur l'actionneur associé à l'ordre. Par contre, si la fonction n'est pas activée, cela signifie qu'un nouvel événement est bien survenu. Il faut donc vérifier la condition d'autorisation en_{xi}^{ei+1} de cet événement. Si la condition est respectée alors un mouvement suite à l'ordre *xi* a bien eu lieu et le défaut est alors localisé sur le composant associé à l'événement non apparu *ei*. Si la condition n'est pas respectée, alors le défaut est aussitôt détecté et localisé sur le composant associé à l'événement violant la condition.

- Enfin, lors d'un nouvel événement *ei*, si la condition d'autorisation est respectée et que la fonction de non-occurrence n'est pas activée, alors il faut regarder la date d'arriver de celui-ci par rapport à la distribution de probabilité en fonctionnement normal. Si la probabilité de cet événement $Prob_N(ei)$ en fonctionnement normal est au-dessus d'un seuil d'acceptation P_{seuil}, alors on considère que le sous-système est en fonctionnement normal et on enregistre la date d'occurrence de ce dernier. Par contre, si $Prob_N(ei)$ est en dessous, l'événement exprime une dérive et le sous-système peut se retrouver en mode dégradé ou défaillant. Il faut alors calculer la nouvelle DP_N' en considérant les *n* derniers événements enregistrés (*n* étant à définir pour limiter le temps de calcul en ligne). Cette DP_N' est ensuite comparée à la DP_N précédente afin d'obtenir un indicateur de dégradation « I ». La localisation du défaut est réalisé ensuite en regardant le max($Prob_N(ei)$, $Prob_{Fj}(ei)$, $Prob_{Fk}(ei)$, …). L'indicateur est ensuite repris afin de fournir une caractéristique d'identification. Il est comparé à un indicateur seuil I_{seuil} de bon fonctionnement et un indicateur de limite de dégradation avant défaillance I_{deg} dont les valeurs sont définies au cas par cas. Pour finir, il faut mettre à jour la distribution de probabilité DP_N.



a) Diagnostic de défauts dans un composant Tout ou Rien (ToR)



b) Diagnostic d'une dérive d'un composant analogique ou le process

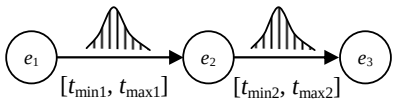
Figure 3.15. Déploiement en ligne du module de diagnostic

3.5.1 Procédures de détection des défauts

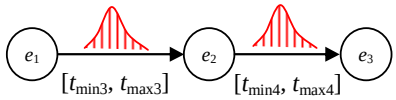
La procédure de détection et localisation dans le cas de la violation d'une fonction de non-occurrence et du non-respect d'une condition d'autorisation étant simple et très proche des travaux de [Philippot et al., 2006], nous ne présentons dans cette section que la partie de procédure utilisant les distributions de probabilité et l'indice de dégradation [Malki et Sayed-Mouchaweh, 2011(b)].

Le fonctionnement du système à un instant « t » est caractérisé par un vecteur v. Ce vecteur est composé de l'ensemble des probabilités d'occurrence de tous les événements $y_i \subset \Sigma_{uc}$. L'occurrence de ces événements est considéré comme une conséquence de l'envoie d'une commande x_i. En effet, ces événements sont corrélés par des contraintes temporelles. Une contrainte temporelle est représentée par un intervalle de temps comprenant les dates d'occurrence d'un événement y_i. Ces dates d'occurrence sont calculées à partir de l'instant de l'envoi de la commande x_i, considéré comme l'instant t_0.

Prenons l'exemple de la figure 3.16. Le modèle temporel apparaissant dans la figure 3.15(a) représente le fonctionnement normal d'un système, tandis que celui de la figure 3.15(b) représente son comportement défaillant en présence d'un défaut appartenant à une partition de défauts Π_{F_i} .



(a) : template représentant le fonctionnement normal



(b) : chronique représentant le fonctionnement défaillant en présence d'un défaut $f \in \Pi_{F_i}$

Figure 3.16. Exemple d'une *Template* et d'une *Chronique* d'un systéme

Le fonctionnement d'un système durant des fenêtres temporelles peut être représenté par des régions constituées par les distributions de probabilité. Ces régions, appelés aussi classes, sont représentées dans un espace caractéristique formé par les dates d'occurrence des événements qui sont liés à un comportement bien précis. Ces événements sont définis comme la conséquence directe de l'envoie d'une commande x_i. Dans l'exemple précèdent, les dates d'occurrence des deux événements e_2 et e_3 sont calculées à partir de l'instant t_0 de l'envoie de la commande e_1. Ces dates d'occurrences sont exprimées via des contraintes temporelles représentées par les intervalles du temps $[t_{min1}, t_{max1}]$, $[t_{min2}, t_{max2}]$ dans le cas du comportement normal du système et par $[t_{min3}, t_{max3}]$, $[t_{min4}, t_{max4}]$ dans le cas de son comportement défaillant. Chaque intervalle du temps est accompagné par une distribution de probabilité correspondant aux dates d'occurrence de l'événement associé. Les régions représentant les deux comportements, normal et défaillant, du système de la figure. 3.16 sont formées par les

contraintes temporelles de deux modèles temporels (Figure 3.17). La région en texture noire correspond au fonctionnement normal du système tandis que la région en texture rouge représente son fonctionnement défaillant.

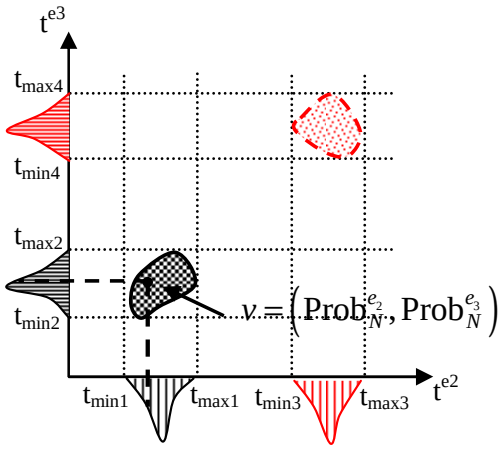


Figure 3.17. Régions corespondantes au fonctionnements normal et defaillants du système.

Afin de déterminer si le système est en mode de fonctionnement normal ou en défaut, un vecteur *v* représentant l'état du système à un instant « *t* » est utilisé. Ce vecteur est appelé *pattern* (Figure 3.17). Chaque *pattern* comporte les probabilités d'occurrence de tous les événements à cet instant « *t* ». Le pattern *v* appartenant à l'une des régions normale ou défaillante est représenté par un point dans l'espace caractéristique formé par les événements associés. Classifier un pattern *v* et déterminer son appartenance à l'une des régions ou classes normale, W_N, ou défaillante, W_{F_i}, est réalisée comme suit :

$$\begin{cases} v \in W_N & \text{if } \text{Prob}_N(v) > \text{Prob}_{F_i}(v) \\ v \in W_{F_i} & \text{sinon} \end{cases} \qquad (3.8)$$

Où $\text{Prob}_N(v)$ et $\text{Prob}_{F_i}(v)$ désignent respectivement la probabilité d'appartenance de *v* à W_N et W_{F_i}.

La procédure de détection consiste donc à regarder l'appartenance d'un événement à une classe normale, mais aussi de vérifier que cette appartenance représente un lien fort d'occurrence et non pas le début d'une dérive. Ainsi, la probabilité de l'événement *ei* en fonctionnement normal est comparée tout d'abord à un seuil d'acceptation P_{seuil}. La définition de ce seuil est très arbitraire en fonction du système considéré. Si $\text{Prob}_N(ei)$ est supérieure à ce

seuil, alors cet événement est considéré comme normal et est enregistré. Dans le cas contraire, une dérive de fonctionnement est donc détectée.

3.5.2 Procédure de localisation des défauts

A la détection de cette dérive, une nouvelle distribution de probabilité DP_N' est calculée à l'aide des *n* derniers enregistrements effectués en fonctionnement normal. Ce paramètre *n* est limité afin de ne pas provoquer un temps de calcul de cette DP_N' trop importante.

En fait, l'occurrence d'un défaut dans le système ou l'un de ses composants implique des changements dans leur fonctionnement. Ces changements peuvent être caractérisés par un déplacement de la distribution de probabilité représentant la dynamique du système dans le fonctionnement normal. Lorsque le système commence à mal fonctionner (dévier), les distributions de probabilité, caractérisant la chance d'occurrence des événements, se déplacent d'une région normale vers une autre représentant une défaillante ou un fonctionnement dégradé (Figure 3.18).

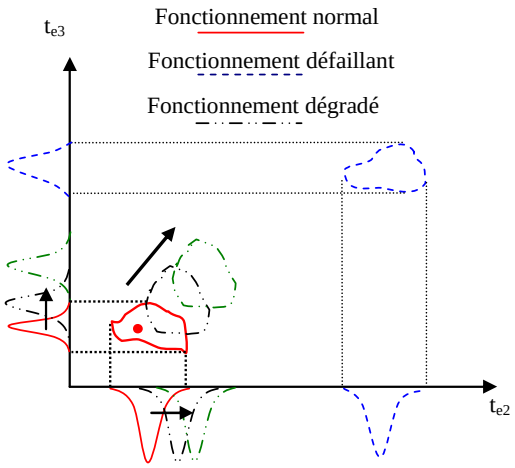


Figure 3.18. Caractérisation d'une déviation du système par des distributions de probabilité.

Un déplacement de la distribution de probabilité caractérisant la déviation du système de son mode fonctionnement normal vers un mode de fonctionnement en présence d'un défaut peut être identifié via un *indicateur de dégradation* « *I* ». En effet, considérons la distribution de probabilité DP_e construite par apprentissage pour l'occurrence d'un événement *e* lors d'un fonctionnement normal du système. La distribution de probabilité de cet événement construite en ligne est donnée par DP'_e. Dans la littérature, il existe plusieurs méthodes qui sont utilisés pour comparer deux distributions de probabilité [Dall'Aglio, 1991] [Cha et Srihari, 2002]. En général, elles mesurent la distance géométrique entre deux distributions de probabilité. Dans

notre travail, nous avons choisi celle de SØrensen [Sørensen, 1948] afin de détecter un déplacement d'une distribution de probabilité. L'indicateur de dégradation est calculé comme suit :

$$I = \max\left(\frac{\left|DP_e - DP_e^{'}\right|}{\left|DP_e + DP_e^{'}\right|}\right),\ 0 \le I \le 1 \qquad (3.9)$$

Lorsque la valeur de l'indicateur *I* est égale à zéro, cela indique qu'il n'y a pas de déviation du système de son mode de fonctionnement normal. Cependant, lorsque cet indicateur prend la valeur 1, alors un écart complet dans la distribution de probabilité est présent et par conséquent une déviation du système a eu lieu. Ce paramètre va permettre notamment de caractériser le défaut lorsque celui-ci sera localisé.

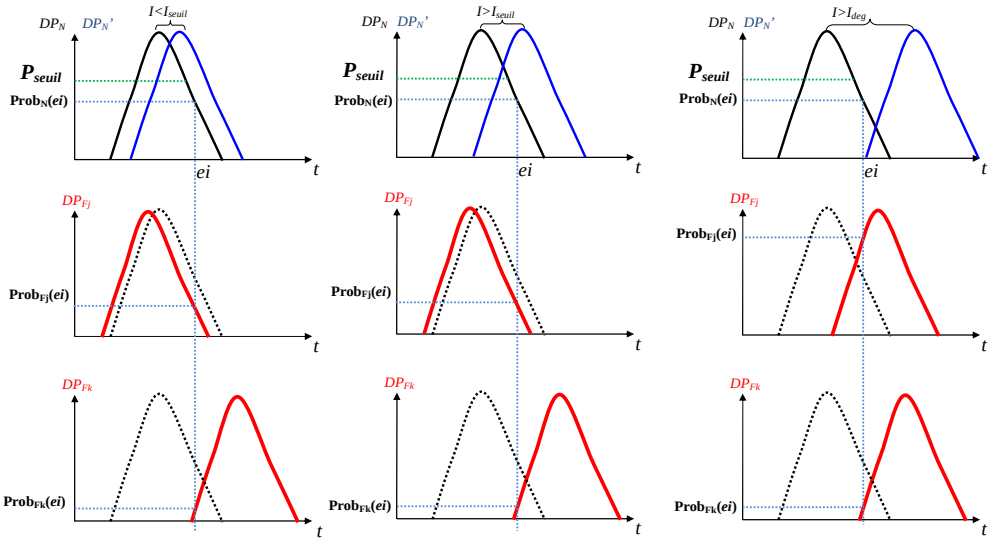
Pour cela, la probabilité de l'événement *ei* est calculé maintenant pour les distributions $\{\text{Prob}_{Fj}(ei), \text{Prob}_{Fk}(ei), \ldots\}$ de chaque modèle de défauts $\{DF_j, DF_k \ldots\}$. Ces probabilités sont comparées avec celle de fonctionnement normal afin de trouver par le maximum l'appartenance de l'événement.

$$\forall ei \in \Sigma_{uc} : \text{si } \text{Prob}(ei) = \max(\text{Prob}_{\text{X}}(ei)) \Rightarrow ei \in D_{\text{X}} \left| \text{X} \in \left\{\text{N}, \text{F}_{\text{j}}, \text{F}_{\text{k}}\right\}\right. \qquad (3.10)$$

Si la probabilité maximum de (3.10) est $\text{Prob}_N(ei)$, alors le fonctionnement n'est peut-être pas dégradé. Il faut alors regarder l'indicateur de dégradation *I*. Si l'indicateur *I* est inférieure à un seuil de tolérance du fonctionnement normal I_{seuil}, alors le comportement est considéré comme en dérive mais dans un fonctionnement normal. Sinon, cela signifie qu'il existe un mode de fonctionnement dégradé qui n'a pas été encore pris en compte. Il faut alors reconstruire un nouveau modèle D_{Fn} sur ce type de défaut.

Dans le cas où la probabilité maximum de (3.10) est $\text{Prob}_{Fj}(ei)$, alors le défaut est localisé sur le composant de modèle D_{Fj}. Par contre, ce défaut peut tout aussi bien représenter une dégradation en cours qu'une défaillance définitive. Pour identifier cette caractéristique, il faut à nouveau utiliser l'indicateur de dégradation *I* afin de la comparer cette fois-ci avec un indicateur de limite de dégradation avant défaillance I_{deg} (Figure 3.19). Cette limite est arbitraire mais il convient souvent de la définir à 0.5. En effet, la figure 3.20 montre plusieurs indicateurs *I* représentant la différence entre deux distributions de probabilité. La distribution

de probabilité en trait noir correspond au fonctionnement normal tandis que celle en trait rouge correspond à un mode de fonctionnement en présence d'un défaut dans le système. Il convient de remarquer qu'un seuil supérieur ou égal à une valeur 0,5 pour l'indicateur est souvent suffisant pour confirmer une dégradation forte du système.



c) Max(Prob$_X$(ei)) = Prob$_N$(ei) et I<I$_{seuil}$ b) Max(Prob$_X$(ei)) = Prob$_N$(ei) et I>I$_{seuil}$ a) Max(Prob$_X$(ei)) = Prob$_{Fj}$(ei) et I>I$_{deg}$

Figure 3.19. Détection et localisation selon probabilités

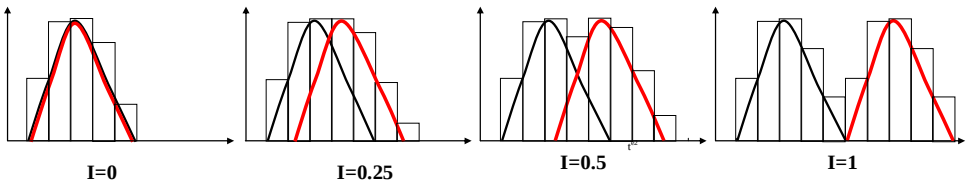


Figure 3.20. Distributions de probabilité pour différentes valeurs de l'indicateur *I*

Suite à cette identification d'appartenance, la distribution de probabilité du sous-système en fonctionnement normal concerné est mise à jour.

3.6 Conclusion

Ce chapitre constitue le cœur de l'approche de diagnostic proposée. Elle se décompose en deux phases : (i) une première phase hors ligne de construction du module de diagnostic et (ii) une deuxième phase en ligne de déploiement de ce module pour détection et localisation de défauts.

La phase de la construction consiste en la décomposition du système en sous-systèmes indépendants. Sur chacun d'eux, un apprentissage est effectué afin d'obtenir aussi bien un modèle du comportement désiré local (par *Template*), qu'un modèle de comportement

défaillant pour chaque type de défauts (par *Chroniques*). Chaque contrainte temporelle est modélisée par une distribution de probabilité (*DP*) caractérisant un fonctionnement normal, dégradé ou défaillant avec un certain degré de certitude. Cette identification du fonctionnement est représentée par la valeur d'un indicateur de dégradation *I*.

De plus, l'occurrence des évènements est caractérisé par des conditions d'autorisation ainsi que des fonctions booléennes de non-occurrence afin de détecter et localiser les défauts qui peuvent avoir lieu dans les équipements du système (capteur, actionneur). Ces deux outils sont utilisés afin d'expliquer la non-occurrence des évènements attendus, ou l'occurrence imprévue, des événements dans le système.

Par cette procédure, il est aussi bien possible de détecter des défauts d'équipements, comme réalisé dans [Philippot et al., 2007] et [Lunze et Schröder, 2000] par exemple, que des défauts et/ou dégradation du process. Il sera par conséquent possible de se tourner peu à peu vers le diagnostic de Systèmes à Dynamique Hybride (SDH).

Dans le chapitre suivant, l'approche proposée est illustrée et testée sous *Matlab/Simulink* en utilisant un benchmark de deux réservoirs équipés de composants discrets mais dont avec la dynamique apparaît comme continue. Ainsi, les deux côtés pourront être étudiés.

Chapitre 4
Illustration de l’approche et résultats de simulation

4.1 INTRODUCTION 102
4.2 DESCRIPTION DU SYSTEME 102
4.3 CONSTRUCTION DU DIAGNOSTIQUEUR (HORS LIGNE) 103
4.3.1 Décomposition du système 104
4.3.2 Modélisation du système 105
4.3.2.1 Construction des distributions de probabilité 110
4.3.2.2 Construction des conditions d'autorisation et des fonctions de non-occurrence 111
4.4 DEPLOIEMENT DU DIAGNOSTIQUEUR (EN LIGNE) 114
4.4.1 Procédure de détection, localisation et identification des défauts 114
4.4.1.1 Défauts abrupts dans des composants de type Tout ou Rien « ToR » 115
4.4.1.2 Défauts graduels dans des composants de type analogique ou dans le processus 117
4.4.2 Décision et dialogue Homme-Machine 122
4.5 CONCLUSION 124

Chapitre 4 : Illustration de l'approche et résultats de simulation

4.1 Introduction

Dans le chapitre précèdent, nous avons représenté notre approche pour le diagnostic des systèmes à évènements discrets. Dans cette présentation, nous avons vu que notre approche se décompose en deux phases principales : (i) élaboration hors ligne du module de diagnostic et (ii) exploitation en ligne. Ce présent chapitre est consacré à illustrer les divers concepts théoriques développés dans les chapitres précédents au travers d'un exemple reconnu par la communauté automaticienne.

Cet exemple permettra de développer à travers les différentes sections :

- le principe de la décomposition avec la notion d'indépendance,
- la modélisation des modules de diagnostic hors ligne,
- l'analyse des résultats d'implémentation.

La modélisation des sous-systèmes et les résultats d'expérimentation sur le benchmark sont réalisés en simulation à l'aide de l'outil logiciel *Matlab/Simulink*.

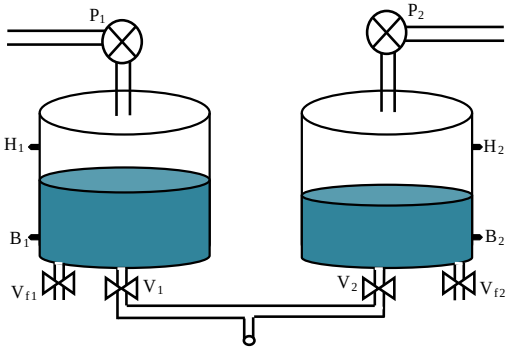
4.2 Description du système

Afin d'illustrer la tâche du diagnostic, le benchmark de deux réservoirs (Figure. 4.1(a)) a été considéré. Cet exemple, a priori plutôt continu, a été choisi afin de montrer que l'approche de cette thèse peut s'étendre à l'étude des Systèmes à Dynamique Hybride. En effet, nous allons considérer dans un premier temps ce benchmark comme un ensemble équipé d'actionneurs et de capteurs purement discrets. Puis par la suite, nous le considérerons comme un système hybride où certains actionneurs (ou capteurs) peuvent être analogiques. La cohabitation des informations Tout ou Rien et continues sera traitée afin de faire le diagnostic aussi bien de défauts sur l'équipement que sur le process [Malki et Sayed-Mouchaweh, 2011(a)] et [Malki et Sayed-Mouchaweh, 2011(b)].

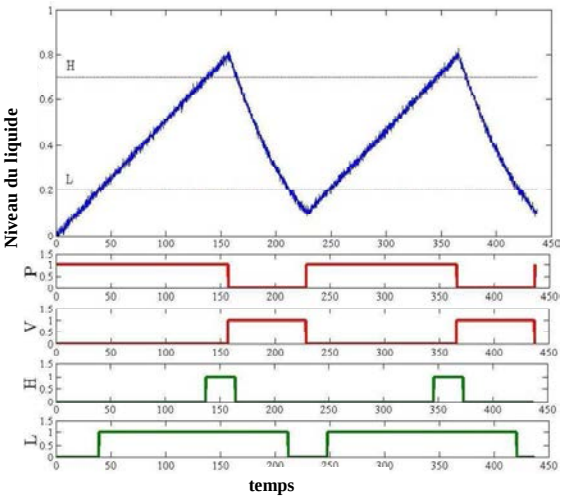
Le benchmark considéré dans la figure. 4.1(a) est un système hydraulique composé de deux réservoirs cylindriques R_1 et R_2 de section S_1 et S_2. Les deux réservoirs sont alimentés

par deux pompes identiques P_1 et P_2 avec des débits Q_{E1} et Q_{E2}. Les deux pompes sont commandées en ToR. La dynamique de ces pompes est rapide, alors le transitoire sera négligé. Les débits de sorties Q_{S1} et Q_{S2} des deux réservoirs sont mélangés via les vannes V_1 et V_2 qui sont commandées également en Tout ou Rien. Le benchmark est équipé de deux vannes supplémentaires V_{ft1} et V_{ft2} afin de simuler une fuite dans R_1 et R_2 pour notre étude. Chaque réservoir est équipé de deux capteurs discrets mesurant les niveaux bas (B_i=1) et haut (H_i=1), où i = {1, 2} indique le numéro du réservoir. La section de la vanne V_1 est donnée par A_1, tandis que la section de la vanne V_2 est donnée par A_2.

L'objectif principal du système de commande est de maintenir les niveaux des liquides dans chaque réservoir R_i entre les hauteurs B_i et H_i par action sur la vanne V_i et la pompe P_i. Le système est donc considéré comme cyclique. Initialement, les pompes sont considérées à l'arrêt et toutes les vannes sont fermées. Avant la mise en marche du système, les deux réservoirs sont vides. La figure. 4.1(b) représente le comportement désiré du système pour un réservoir avec ses événements contrôlables et non contrôlables correspondants.



(a) : Système (*S*) de deux réservoirs



(b) : Cycles de fonctionnement et les événements observables pour un réservoir

Figure 4.1. Sytème de deux réservoirs avec ses cycles de fonctionnement et ses événements observables

4.3 Construction du diagnostiqueur (hors ligne)

Cette partie traite de la mise en œuvre de l'approche développée dans le chapitre 3 autour du benchmark de deux réservoirs.

4.3.1 Décomposition du système

La procédure de décomposition du système développée dans le chapitre précédent (§ 3.4.1.1) est appliquée sur le système global (S) de deux réservoirs. Le système de deux réservoirs (S) est considéré en tant que système à événements discrets. L'ensemble des événements observables Σ_o est donc donné par le tableau 4.1. Cet ensemble est divisé en deux sous-ensembles distinctes : (i) un ensemble des événements observables et contrôlables représentés par les deux premières lignes du tableau 4.1 ; (ii) un ensemble des événements observables mais non contrôlables représentés par la troisième ligne du tableau 4.1.

Equipements	Evénements observables	Evénements contrôlables	Evénements non contrôlables
Pompes	$\uparrow P_1, \downarrow P_1, \uparrow P_2, \downarrow P_2$	✔	✘
Vannes	$\uparrow V_1, \downarrow V_1, \uparrow V_2, \downarrow V_2$	✔	✘
Capteurs de niveaux	$\uparrow B_1, \downarrow B_1, \uparrow B_2, \downarrow B_2, \uparrow H_1, \downarrow H_1, \uparrow H_2, \downarrow H_2$	✘	✔

TABLEAU.4. 1. ÉVENEMENTS OBSERVABLES (CONTROLABLE ET NON CONTROLABLE) DANS LE SYSTEME GLOBALE (S)

Afin d'illustrer la procédure de décomposition, le système de deux réservoirs est subdivisé d'un point de vue événementiel et géographique. De ce fait, nous pouvons considérer que le benchmark de deux réservoirs est composé de deux sous-systèmes SS_1 et SS_2 (Figure. 4.2(a)). Le sous-système SS_1 est composé du réservoir R_1, de la pompe P_1 et de la vanne V_1 tandis que le sous-système SS_2 est composé du réservoir R_2, de la pompe P_2 et de la vanne V_2. Les deux sous-systèmes SS_1 et SS_2 vérifient la notion d'indépendance donnée par l'équation (3.1) sur l'appartenance des événements. Par conséquent, chaque sous-système SS_i, $i \in \{1,2\}$, possède son propre ensemble d'événements observables et contrôlables $\Sigma_{ic}=\{\uparrow P_i, \downarrow P_i, \uparrow V_i, \downarrow V_i\}$ ainsi que son ensemble d'événements observables et non contrôlables $\Sigma_{iuc}=\{\uparrow B_i, \downarrow B_i, \uparrow H_i, \downarrow H_i\}$ (Tableau 4.2)(Figure 4.2(b)). La vérification de la notion d'indépendance par les deux sous-systèmes implique que l'évolution du comportement d'un sous-système n'a aucune relation et/ou influence sur l'évolution de celui du deuxième. Nous pouvons donc par la suite ne considérer qu'un seul des deux réservoirs.

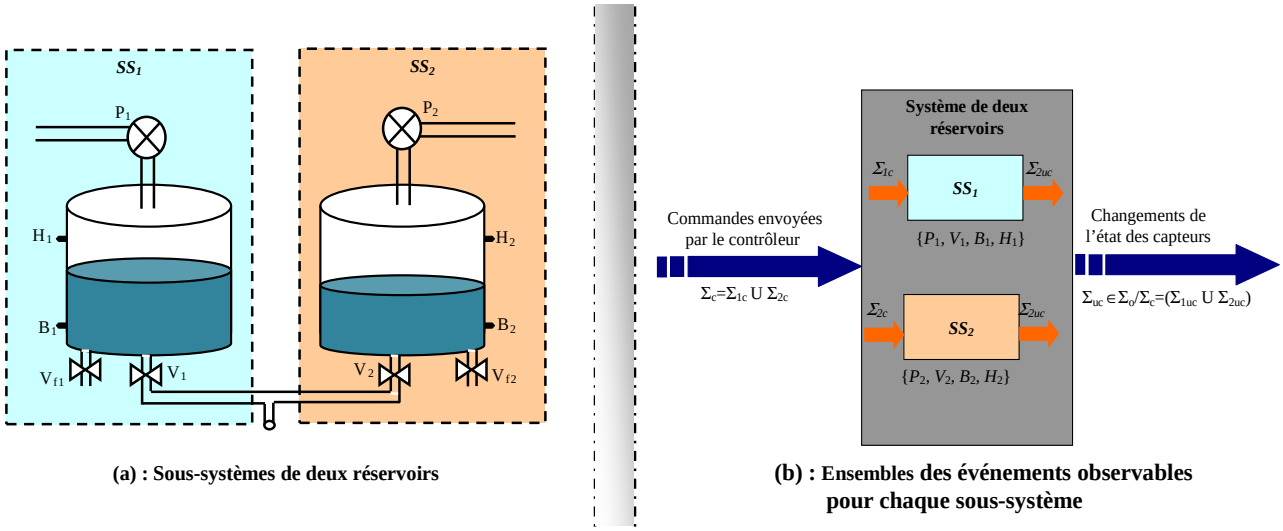


Figure 4.2. Décomposition du système de deux réservoirs en deux sous-systèmes

Evénements dans R $_{i\ /\ i=\{1,2\}}$	***signification***
$\uparrow P_i$	Événement front montant de la commande P_i indiquant le démarrage de la pompe.
$\downarrow P_i$	Événement front descendant de la commande P_i indiquant l'arrêt de la pompe.
$\uparrow V_i$	Événement front montant de la commande V_i indiquant l'ouverture de la vanne.
$\downarrow V_i$	Événement front descendant de la commande V_i indiquant la fermeture de la vanne.
$\uparrow B_i$	Événement front montant de B_i indiquant que le liquide dans le réservoir R_i passe par le niveau bas dans le sens de remplissage.
$\downarrow B_i$	Événement front descendant de B_i indiquant que le liquide dans le réservoir R_i passe par le niveau bas dans le sens de vidange.
$\uparrow H_i$	Événement front montant de H_i indiquant que le liquide dans le réservoir R_i passe par le niveau haut dans le sens de remplissage.
$\downarrow H_i$	Événement front descendant de H_i indiquant que le liquide dans le réservoir R_i passe par le niveau Haut dans le sens de vidange.

TABLEAU.4. 2. ÉVENEMENTS OBSERVABLES ET LEUR SIGNIFICATION DANS CHAQUE RESERVOIR R_i

4.3.2 Modélisation du système

Comme nous l'avons indiqué dans le chapitre précédent, chacun des deux sous-systèmes doit être représenté par un graphe temporel. En fait, le comportement désiré (normal) G_N d'un sous-système SS_i est représenté par une *Template* tandis que son comportement défaillant G_F est représenté par une *Chronique* pour chaque type de défaut. Pour cela, il nous a fallu modéliser le système sous l'environnement logiciel *Matlab/Simulink*.

La figure 4.3 montre la structure haut-niveau sous *Simulink* où l'on retrouve d'un côté le système (PO+PC) et le module diagnostiqueur de l'autre. Dans le modèle Simulink du système, on retrouve une représentation du système physique (réservoirs + actionneurs), une représentation des capteurs de niveaux, une représentation de la commande avec la possibilité de générer des défaillances ou dégradations, et enfin un module de visualisation des différents signaux (Figure 4.4).

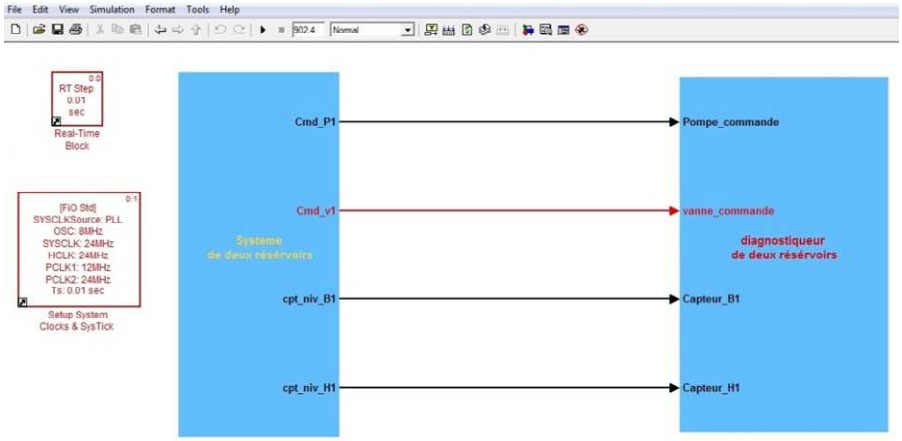


Figure 4.3. Structure haut-niveau des deux réservoirs

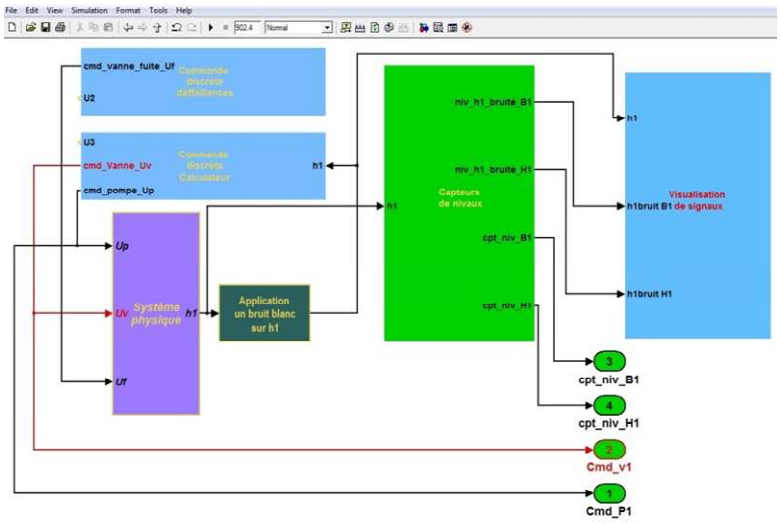


Figure 4.4. Modèles Simulink du système de deux réservoirs

Les différentes partitions de défauts à diagnostiquer sont représentées dans le tableau 4.3. Cette liste détaille les partitions de défauts dans chaque sous-système SS_i. La procédure de diagnostic est applicable de la même manière pour les deux sous-systèmes. Dans notre travail, la pompe va être considérer, dans un premier temps, comme un équipement de type

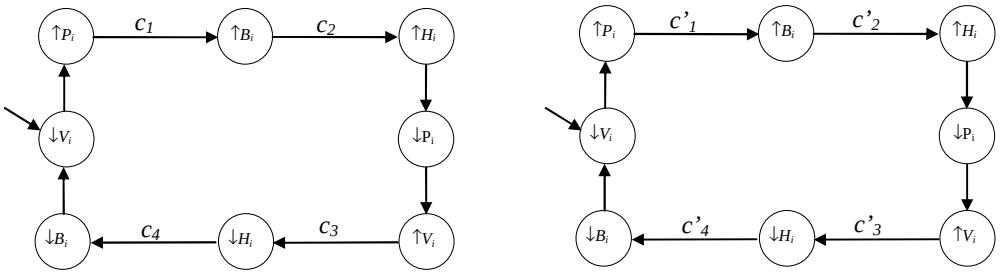
« ToR ». Ensuite, nous allons supposer que ce composant a un comportement analogique afin de diagnostiquer les défauts graduels avec un pourcentage sur l'ouverture de celle-ci. Cela veut dire que la pompe peut aller dans un mode de défaillance « bloquée-on ou bloquée-off » d'une manière progressive (bloquée-on à 20%, 40%, …).

Sous-système SS_i			
Composant du système	Nom du composant	Partition de défauts	Types de défauts
Actionneurs	Pompe (P_i)	$\Pi_{F_1} = \{F_1, F_2\}$	F_1: pompe P_i bloquée-on
			F_2: pompe P_i bloquée-off
	Vanne (V_i)	$\Pi_{F_2} = \{F_3, F_4\}$	F_3: vanne V_i bloquée-on
			F_4: vanne V_i bloquée-off
Capteurs	Capteur niveau bas "B_i"	$\Pi_{F_3} = \{F_5, F_6\}$	F_5: capteur B_i bloqué-on
			F_6: capteur B_i bloqué-off
	Capteur niveau haut "H_i"	$\Pi_{F_4} = \{F_7, F_8\}$	F_7: capteur H_i bloqué-on
			F_8: capteur H_i bloqué-off
Processus	Réservoir "R_i"	$\Pi_{F_5} = \{F_9\}$	F_9: fuite dans le réservoir R_i

TABLEAU.4. 3. PARTITIONS DE DEFAUT DANS LE RESERVOIR R_i

Chaque contrainte temporelle c_i sur les dates d'occurrence d'un événement est calculée à partir de l'instant t_0 de l'envoie de la commande qui l'a déclenché. Dans le cas des sous-systèmes SS_i, l'ensemble des contraintes temporelles associées pour chaque occurrence d'événements est donnée par le tableau 4.4. Dans ce tableau, « UT » signifie unité du temps, elle dépend du processeur utilisé pour la simulation. Ces contraintes temporelles sont établies en exploitant l'historique du fonctionnement du système (fonctionnement normal et défaillant) pendant 100 cycles sous *Matlab/Simulink*. Pour cela, il nous a fallu modéliser la *Template* et les *Chroniques* de la figure 4.5 sous cet environnement logiciel (Figure 4.6). Cependant, pour

représenter sous cet environnement le fonctionnement d'une *Template* ou *Chronique*, nous avons dû effectuer une équivalence de nos modèles sous *StateFlow* (Figure 4.7).



(a) : Template du fonctionnement normal (G_N) **(b) : Chronique du fonctionnement défaillant « pompe bloquée-on » (G_{Fi})**

Figure 4.5. Modéles temporels des fonctionnements normal et defaillant « pompe bloquée-on » du sous-système SS_1

Activation (désactivation) de la commande X à l'instant t_0	Les événements conséquences à l'activation (désactivation) de la commande X	Les contraintes temporelles pour chaque événement à partir de l'instant t_0	la date minimale t_{min}^{e} pour l'occurrence d'un évènement *e* (UT)	la date maximale t_{max}^{e} pour l'occurrence d'un évènement *e* (UT)
$\uparrow P_i$ (remplissage du réservoir "i")	$\uparrow B_i$	$c_1=[t_{min}^{\uparrow Bi}, t_{max}^{\uparrow Bi}]$	58,8	87,2
	$\uparrow H_i$	$c_2=[t_{min}^{\uparrow Hi}, t_{max}^{\uparrow Hi}]$	119,3	158,3
$\uparrow V_i$ (vidange du réservoir "i")	$\downarrow H_i$	$c_3=[t_{min}^{\downarrow Hi}, t_{max}^{\downarrow Hi}]$	3,1	6,2
	$\downarrow B_i$	$c_4=[t_{min}^{\downarrow Bi}, t_{max}^{\downarrow Bi}]$	26,8	31,1

TABLEAU.4. 4. LES CONTRAINTES TEMPORELLES DANS LE CAS DU FONCTIONNEMENT NORMAL POUR UN SOUS-SYSTEME SS_1

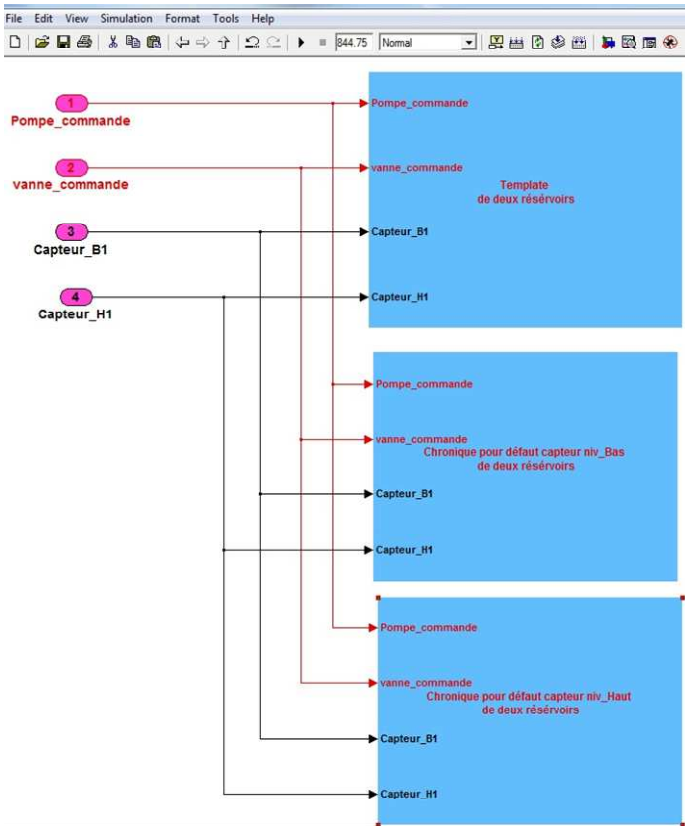


Figure 4.6. Modèles Simulink de la Template et de deux Chroniques

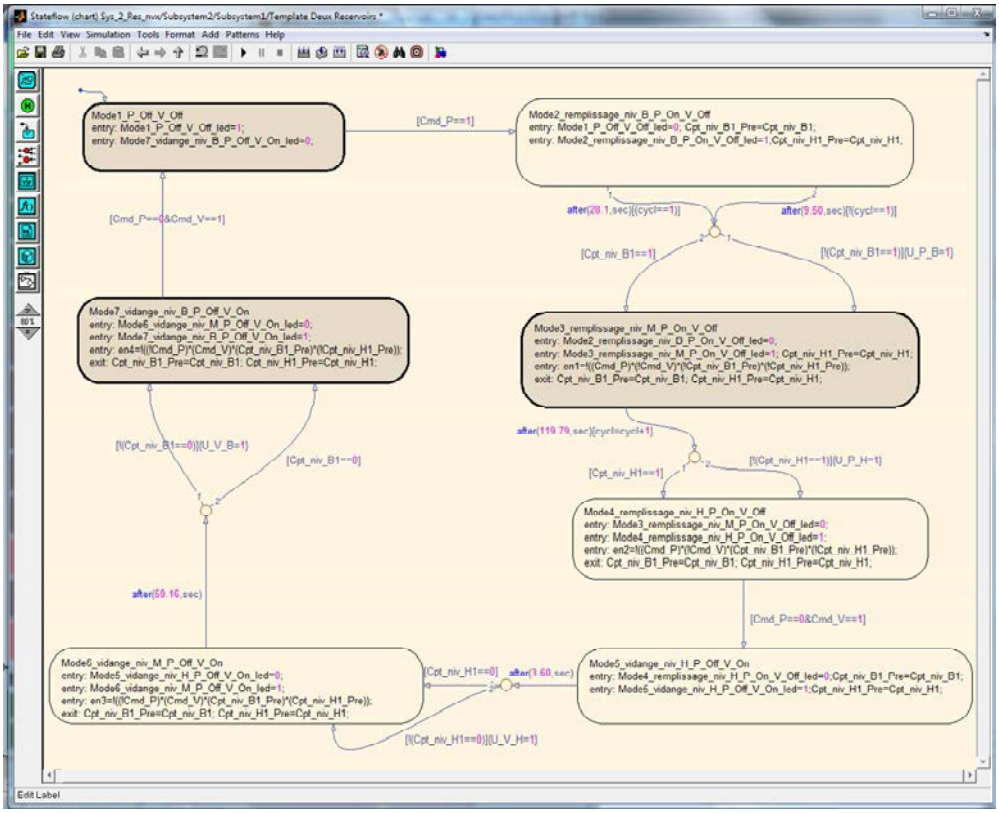
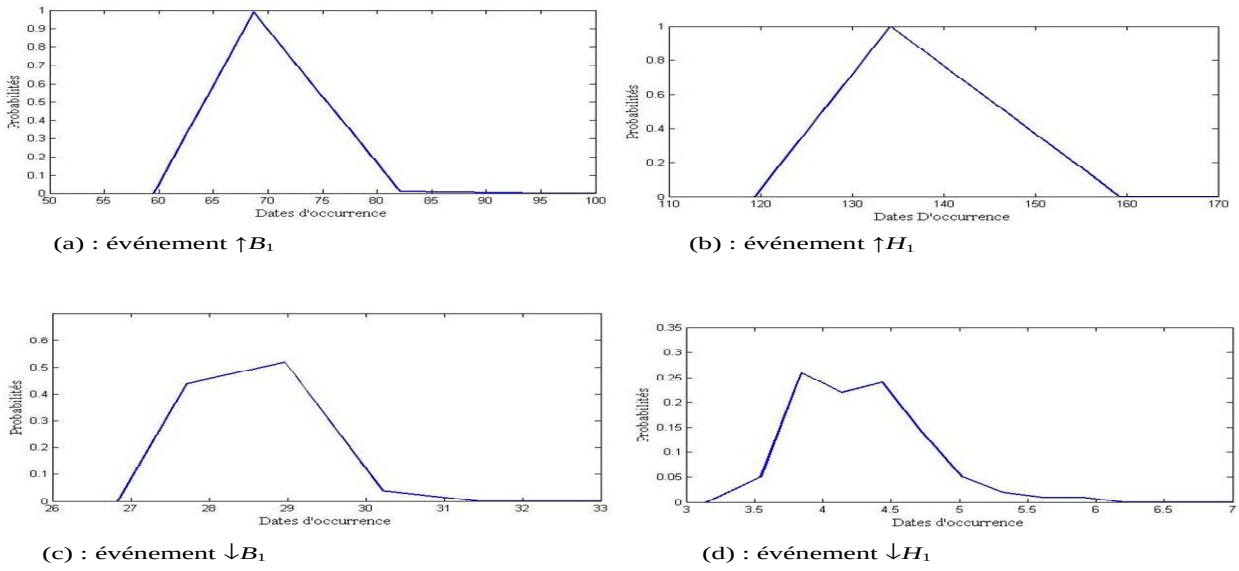


Figure 4.7. Modèle StateFlow de la Template du sous-système SS_1

De même, les contraintes dans le cas d'occurrence d'un défaut dans le système (Pompe bloquée-on ou off, capteur bloqué, fuite…) sont également obtenues par la simulation. Les dates d'occurrence des événements observables non-contrôlables sont calculées à partir de l'instant t_0 de l'activation de leur commande correspondante.

4.3.2.1 Construction des distributions de probabilité

La construction de la distribution de probabilité est réalisée en se basant sur l'utilisation de l'histogramme des probabilités. Cet histogramme est limité aux deux valeurs, minimale t_{min}^{e} et maximale t_{max}^{e}, de la contrainte temporelle correspondante à l'événement *e* (tableau 4.3). L'histogramme est divisé en « *h=20* » barres d'une largeur Δ. Cette largeur est calculée en utilisant la relation (3.2) indiquée dans le chapitre 3. La hauteur des barres dans les différentes contraintes est représentée par la probabilité correspondant pour chaque date d'occurrence. En effet, les distributions de probabilité concernant l'occurrence des événements ($\uparrow B_i$, $\uparrow H_i$), respectivement ($\downarrow B_i$, $\downarrow H_i$), dans le cas du fonctionnement normal sont calculées en se basant sur les valeurs obtenues par simulation (Figure.4.8).

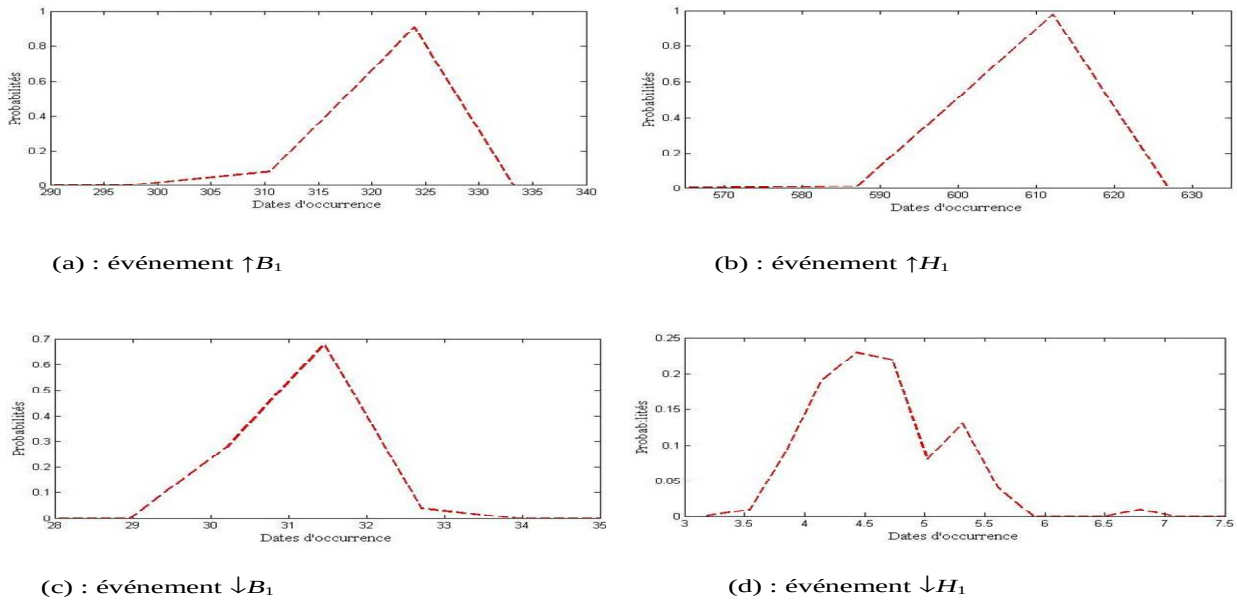


(a) : événement $\uparrow B_1$ (b) : événement $\uparrow H_1$

(c) : événement $\downarrow B_1$ (d) : événement $\downarrow H_1$

Figure 4.8. Distributions de Probabilité des événements $\uparrow B_1$, $\uparrow H_1$, $\downarrow B_1$ et $\downarrow H_1$ dans le cas du fonctionnement normal .

Dans le cas d'un défaut sur la pompe bloqué à 20% de son ouverture, les distributions de probabilité fournissent une signature différente caractéristique de ce défaut (Figure 4.9).



(a) : événement $\uparrow B_1$ (b) : événement $\uparrow H_1$

(c) : événement $\downarrow B_1$ (d) : événement $\downarrow H_1$

Figure 4.9. Distributions de Probabilité des évènements $\uparrow B_1$, $\uparrow H_1$, $\downarrow B_1$ et $\downarrow H_1$ dans le cas de fonctionnement avec un défaut d'une pompe bloquée-on à 20% de son débit nominal.

4.3.2.2 Construction des conditions d'autorisation et des fonctions de non-occurrence

☞ ***Conditions d'autorisation***

Dans chacun des sous-systèmes SS_i, les conditions d'autorisation pour chaque événement e_i sont obtenues en se basant essentiellement sur l'utilisation du vecteur $\left(BV_{x_i}^{B_i H_i}\right)_{present}$ décrivant l'état actuel du système. Ces conditions sont considérées comme la combinaison "AND" de tous les variables représentant l'état booléen actuel du système :

$$\forall i \in \{1,2\} : en_{x_i}^{e_i} = AND((BV_{x_i}^{B_i,H_i})_{present}) \tag{4.1}$$

L'évolution de chaque sous-système est décrite par le vecteur $(BV_{P_1}^{B_1,H_1})_{nouveau}$ représentant son état futur et la matrice de déplacement $DV_{x_i}^{e_i} = (dv_{B_i}, dv_{H_i})$ indiquant les changements qui ont eu lieu dans le sous-système en question. De ce fait, les matrices de déplacement, DV_{P_1} et DV_{V_1}, pour les différents événements, dans le sous-système SS_i, sont données comme suit:

$$DV_{P_1} = \begin{array}{c} \\ \uparrow B_1 \\ \uparrow H_1 \end{array} \begin{array}{c} \begin{array}{cc} B_1 & H_1 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \tag{4.2}$$

$$DV_{V_1} = \begin{array}{c} \\ \downarrow B_1 \\ \downarrow H_1 \end{array} \begin{array}{c} \begin{array}{cc} B_1 & H_1 \end{array} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \qquad (4.3)$$

Nous supposons que le système dans son état initial est dans un mode de fonctionnement normal et que les deux réservoirs sont vides. De ce fait, le vecteur booléen $(BV_{P_1}^{B_1,H_1})_{présent}$ est égal à $(\overline{B_1},\overline{H_1})$. A partir de cet état initial, l'événement attendu est le passage par le niveau bas ($\uparrow B_i$) après démarrage de la pompe P_1. Dans ce cas, la condition d'autorisation est donnée par:

$$en_{P_1}^{\uparrow B_1} = AND((BV_{P_1}^{B_1,H_1})_{présent}) = \overline{B_1}.\overline{H_1} = \overline{0}.\overline{0} = 1.$$

L'occurrence de l'événement $\uparrow B_1$ est donc autorisée. Par conséquent, après l'occurrence de l'événement $\uparrow B_1$, le vecteur booléen devient :

$$(BV_{P_1}^{B_1,H_1})_{nouveau}^{\uparrow B_1} = (BV_{P_1}^{B_1,H_1})_{présent} \oplus DV_{P_1}^{\uparrow B_1} = (0,0) \oplus (1,0) = (B_1,\overline{H_1}).$$

Ensuite, l'occurrence de l'événement $\uparrow H_1$ est autorisée par la condition :

$$en_{P_1}^{\uparrow H_1} = AND((BV_{P_1}^{B_1,H_1})_{présent}) = B_1.\overline{H_1}.$$

L'occurrence de cet événement change le vecteur $(BV_{P_1}^{B_1,H_1})_{présent}$ de $(B_1,\overline{H_1})$ à (B_1,H_1).

Le tableau 4.5 représente les conditions d'autorisation pour l'occurrence de chacun des événements à partir de l'activation de sa commande correspondante *P_1* ou *V_1*.

Tâche à effectuer	Commande envoyée	Evènements conséquences	Condition d'autorisation
Remplissage du réservoir R_i	$\uparrow P_i$	$\uparrow B_i$	$en_{\uparrow P_i}^{\uparrow B_i} = P_i.\overline{V_i}.\overline{B_i}.\overline{H_i}$
		$\uparrow H_i$	$en_{\uparrow P_i}^{\uparrow H_i} = P_i.\overline{V_i}.B_i.\overline{H_i}$
Vidange du réservoir R_i	$\uparrow V_i$	$\downarrow H_i$	$en_{\uparrow V_i}^{\downarrow H_i} = \overline{P_i}.V_i.B_i.H_i$
		$\downarrow Bi$	$en_{\uparrow V_i}^{\downarrow B_i} = \overline{P_i}.V_i.B_i.\overline{H_i}$

TABLEAU.4. 5. CONDITIONS D'AUTORISATION POUR LES EVENEMENT DU SYSTEME DE DEUX RESERVOIRS

☞ ***Fonctions booléennes de non-occurrence***

Les fonctions booléennes caractérisant la non-occurrence des événements dans le système de deux réservoirs sont mentionnées dans le tableau 4.6. À partir de ce tableau, nous pouvons constater que chacune des fonctions vaut la valeur « 0 » dans le cas d'événement attendu, tandis qu'elle vaut la valeur « 1 » dans le cas de non occurrence de l'événement correspondant. En fait, une fonction booléenne $U_{x_i}^{e_i}$ passe à l'état active « 1 » dans un temps Δt^{e_i}, représentant une marge de sécurité, après le franchissement de la limite maximale d'une contrainte temporelle (Figure 4.10).

Tâche à effectuer	Commande envoyée	Evènements conséquences	Fonction booléenne
Remplissage du réservoir R_i	$\uparrow P_i$	$\uparrow B_i$	$U_{\uparrow P_i}^{\uparrow B_i}$
		$\uparrow H_i$	$U_{\uparrow P_i}^{\uparrow H_i}$
Vidange du réservoir R_i	$\uparrow V_i$	$\downarrow H_i$	$U_{\uparrow V_i}^{\downarrow H_i}$
		$\downarrow Bi$	$U_{\uparrow V_i}^{\downarrow B_i}$

TABLEAU.4. 6. FONCTIONS BOOLEENES POUR LES EVENEMENT DU SYSTEME DE DEUX RESERVOIRS.

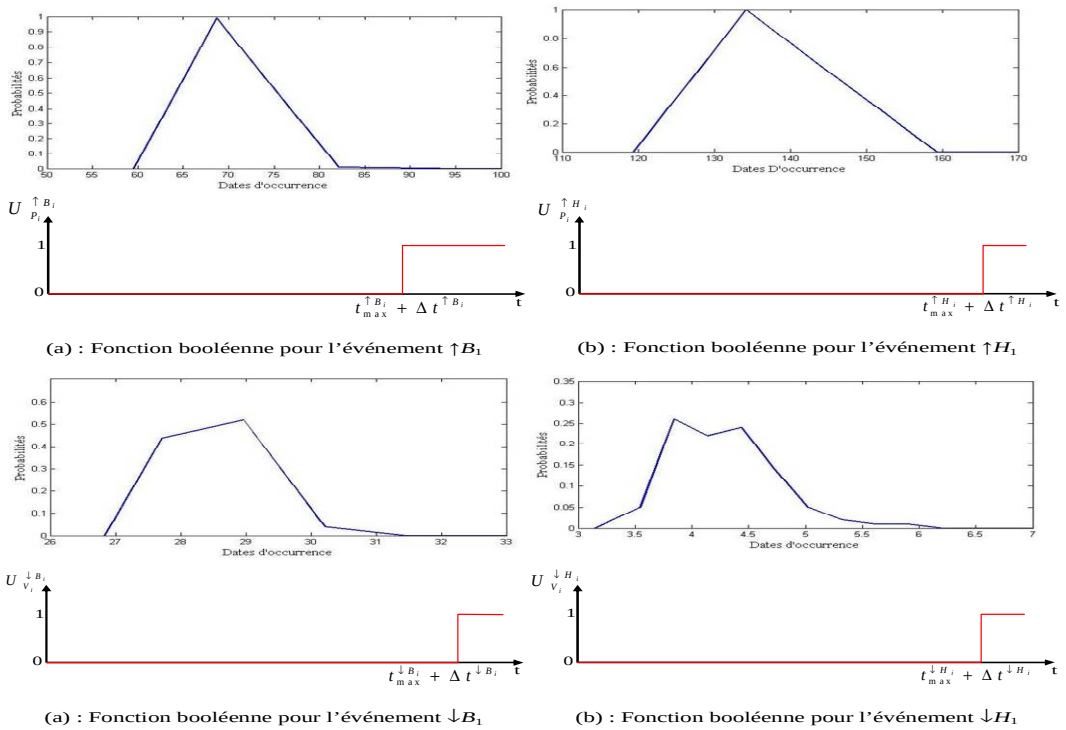


Figure 4.10. Fonctions de non-occurrence d'événement

4.4 Déploiement du diagnostiqueur (en ligne)

Dans cette partie nous allons exploiter le diagnostiqueur construit hors ligne pour détecter, localiser et identifier les défauts dans le système de deux réservoirs. Dans le cas de ce benchmark, nous nous sommes intéressé aux deux catégories de défauts : (i) défauts abrupts permanents qui peuvent avoir lieu dans des équipements de type « ToR » (actionneurs et capteurs) ; (ii) défauts graduels permanant qui peuvent avoir lieu soit dans des équipements de type analogique (actionneurs et capteurs analogiques) ou soit dans le processus lui-même (fuite,…). Afin d'illustrer notre méthode, les défauts graduels dans les équipements vont être générés seulement au niveau de l'actionneur Pompe P_i. Ce type de défauts peut être généré de la même manière pour les autres équipements. Les défauts graduels au niveau processus vont être illustrés en générant une fuite dans l'un des réservoirs R_i.

4.4.1 Procédure de détection, localisation et identification des défauts

La détection de défauts dans un sous-système SS_i est réalisée en exploitant les distributions de probabilité (DP_N) exprimant les contraintes temporelles qui caractérisent l'occurrence des événements dans le sous-système. La procédure de localisation et

identification de défauts est illustrée en utilisant le système de deux réservoirs. Cette procédure vient après la détection de l'occurrence d'un défaut dans l'un des sous-systèmes. Dans cette section, nous nous sommes intéressés à deux classes de défauts :

- Classe n°1 : Cette classe comporte des défauts selon le composant responsable de leur occurrence. Dans cette classe les défauts sont divisés en trois catégories :
  - ✓ Défauts sur capteurs de niveaux de type « ToR » (Π_{F3}, Π_{F4}),
  - ✓ Défauts sur les actionneurs (pompes et vannes) (Π_{F1}, Π_{F2}). Dans ce travail, les pompes sont au préalable considérées « ToR » avant d'être remplacées par des pompes analogiques.
  - ✓ Défauts sur le processus lui-même.
- Classe n°2 : Cette classe comporte des défauts selon leur dynamique d'occurrence. Dans cette classe les défauts sont divisés en deux catégories :
  - ✓ Défauts abrupts sur les capteurs de niveaux et les actionneurs (pompes et vannes) de type « ToR »,
  - ✓ Défauts graduels sur les actionneurs (pompes et vannes) de type analogique ainsi que dans le processus lui-même (Fuite dans l'un des réservoirs R_i).

4.4.1.1 Défauts abrupts dans des composants de type Tout ou Rien « ToR »

Pour diagnostiquer ces partitions de défauts, les distributions de probabilité, fonctions Booléennes et les conditions d'autorisation Booléens seront utilisés (Tableau 4.7).

Sous-système SS_i				
Composant du système	Nom du composant	Types de défauts	Conséquence	Outil de diagnostic
Actionneurs	Pompe (P_i)	$F_1 \in \Pi_{F_1}$	l'occurrence tardive de $\downarrow B_i$ et $\downarrow H_i$	$v \in W_{F_1}$
		$F_2 \in \Pi_{F_1}$	Non-occurrence de $\uparrow B_1$ et $\uparrow H_1$	$U_{P_i}^{\uparrow B_i} = 1 \wedge U_{P_i}^{\uparrow H_i} = 1$
	Vanne (V_i)	$F_3 \in \Pi_{F_2}$	l'occurrence tardive de $\uparrow B_i$ et $\uparrow H_i$	$v \in W_{F_3}$
		$F_4 \in \Pi_{F_2}$	Non-occurrence de $\downarrow B_i$ et $\downarrow H_i$	$U_{V_i}^{\downarrow B_i} = 1 \wedge U_{V_i}^{\downarrow H_i} = 1$
Capteurs	Capteur niveau bas (B_i)	$F_5 \in \Pi_{F_3}$	Non occurrence de $\downarrow B_i$ et l'occurrence prévue de $\downarrow H_i$	$U_{V_i}^{\downarrow B_i} = 1 \wedge en_{V_i}^{\downarrow B_i} = 1 \wedge U_{V_i}^{\downarrow H_i} = 0$
		$F_6 \in \Pi_{F_3}$	Non occurrence de $\uparrow B_i$ et l'occurrence imprévue de $\uparrow H_i$	$U_{P_i}^{\uparrow B_i} = 1 \wedge en_{P_i}^{\uparrow H_i} = 0 \wedge U_{V_i}^{\downarrow B_i} = 1 \wedge en_{V_i}^{\downarrow H_i} = 0$
	Capteur niveau haut (H_i)	$F_7 \in \Pi_{F_4}$	Non occurrence de $\downarrow H_i$ et l'occurrence imprévue de $\downarrow B_i$	$U_{V_i}^{\downarrow H_i} = 1 \wedge en_{V_i}^{\downarrow H_i} = 1 \wedge U_{V_i}^{\uparrow B_i} = 1 \wedge en_{V_i}^{\downarrow B_i} = 0$
		$F_8 \in \Pi_{F_4}$	Non occurrence de $\uparrow H_1$ et l'occurrence prévue de $\uparrow B_1$ ou Non occurrence de $\downarrow H_i$ et l'occurrence prévue de $\downarrow B_i$	$U_{P_i}^{\uparrow H_i} = 1 \wedge en_{P_i}^{\uparrow H_i} = 1 \wedge U_{P_i}^{\uparrow B_i} = 0 \vee U_{V_i}^{\downarrow H_i} = 1 \wedge en_{V_i}^{\downarrow B_i} = 1$

TABLEAU.4. 7. DIAGNOSTIC EN LIGNE DE DEFAUTS DANS LE SOUS-SYSTEME SS_1

A partir de ce tableau, nous pouvons constater que l'occurrence d'un défaut entraine des conséquences dans le système. Ces conséquences peuvent avoir des effets, directs ou indirects, sur un ou plusieurs événements observables. L'ensemble des conséquences peuvent être exprimées en utilisant les conditions d'autorisation ainsi que les fonctions booléennes caractérisant les événements influencés par l'occurrence d'un défaut. Par exemple, l'occurrence d'un défaut de type « F_2 » où « F_6 » conduit à la non-occurrence de l'événement $\uparrow B_i$.

Afin de localiser et identifier ces deux défauts, nous allons exploiter les conditions d'autorisation et les fonctions booléennes construites pour chaque événement. De ce fait, nous pouvons apercevoir que lorsque l'un des défauts a lieu dans le système, la fonction booléenne $U_{\uparrow P_i}^{\uparrow B_i}$ prendra la valeur « 1 » dans les deux cas. En effet, afin de connaitre, avec certitude, le type de défaut qui a eu lieu dans le système, nous avons besoin d'une autre information qui nous aidera à éliminer les candidats non responsables de son occurrence. Cette information complémentaire consiste en l'attente de l'évènement $\uparrow H_i$, qui vient après l'occurrence de l'événement $\uparrow B_i$ dans le cas du fonctionnement normal du système. En fait,

l'occurrence du défaut « F_2 » est confirmé par la non-occurrence de l'événement $\uparrow H_i$ dans un intervalle de temps prédéfini, représenté par la contrainte temporelle « c_2 », après l'activation de P_i. La non-occurrence de l'événement $\uparrow H_i$ déclenche la fonction booléenne $U_{\uparrow P_i}^{\uparrow H_i}$.

En revanche, l'apparition du défaut « F_6 » dans le système est confirmé par l'occurrence inattendue de l'événement $\uparrow H_i$ après l'activation de P_i, sa condition d'autorisation étant $en_{\uparrow P_i}^{\uparrow H_i} = 0$. La procédure de localisation et d'identification de toutes les partitions de défauts décrites dans le tableau 4.3 peut se faire en suivant le même raisonnement ci-dessus. Cette procédure est réalisée en se basant sur les outils décrits dans le tableau 4.7.

4.4.1.2 Défauts graduels dans des composants de type analogique ou dans le processus

Le diagnostic d'une dérive d'un équipement de type analogique de son fonctionnement normal vers le comportement défaillant est réalisé en observant le déplacement des distributions de probabilité construites dans le cas de fonctionnement normal à l'aide de l'indicateur de dégradation « I ».

☞ ***Défaut sur l'actionneur analogique « Pompe »***

Un déplacement de la nouvelle distribution de probabilité (DP_N') par rapport à la distribution de probabilité normale (DP_N) indique qu'un défaut a eu lieu dans le système (Figure 4.11), (Figure 4.12).

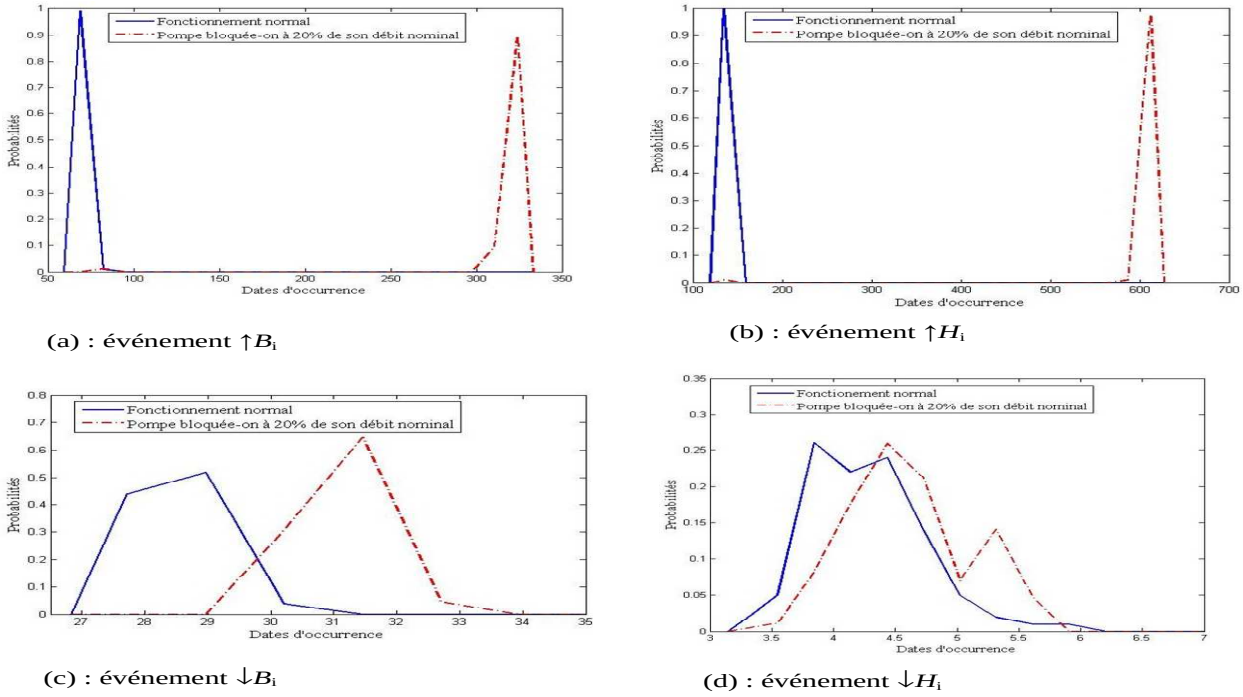


Figure 4.11. Distributions de probabilité des événements $\downarrow B_i$ et $\downarrow H_i$ dans le cas de fonctionnement normal et dégradé (pompe P_i bloquée-on à 20%)

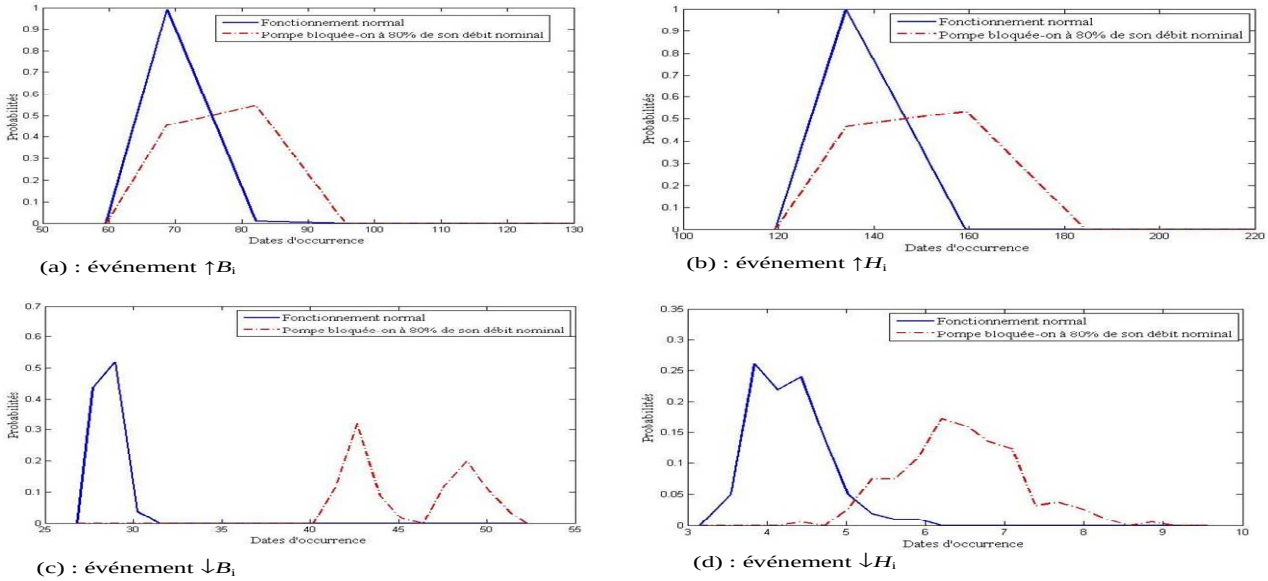


Figure 4.12. Distributions de probabilité des événements $\downarrow B_1$ et $\downarrow H_i$ dans le cas de fonctionnement normal et dégradé (pompe P_i bloquée-on à 80%)

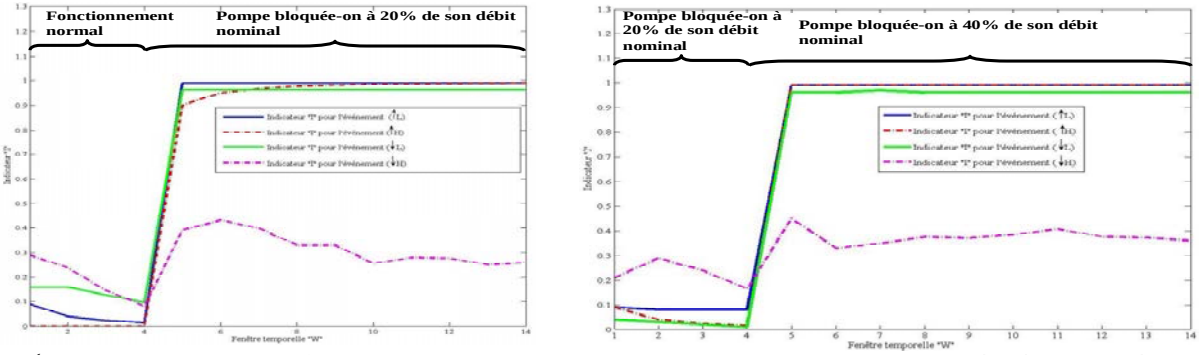
La figure 4.11 montre le déplacement des distributions de probabilité (DP') caractérisant les différents événements (remplissage et vidange des réservoirs) dans le cas où la pompe est bloquée-on à 20% de son débit nominal. Ce déplacement est considéré par rapport aux distributions de probabilité construites dans le cas de fonctionnement normal du système. De même, la figure 4.12 montre le déplacement des distributions de probabilité dans

le cas où la pompe est bloquée-on à 80% de son débit nominal par rapport aux distributions de probabilité dans le cas du fonctionnement normal du système. A partir de ces deux figures, nous pouvons constater que lorsque la pompe est bloquée-on à un faible pourcentage de son débit nominal, le déplacement des distributions de probabilité représentant les événements dans le cas de remplissage des réservoirs est plus important que le déplacement des distributions de probabilité représentant les événements dans le cas de vidange des réservoirs. En revanche, pour un blocage de la pompe en ouverture à un pourcentage élevé de son débit nominal, le déplacement des distributions de probabilité en remplissage est moins important par rapport au déplacement des distributions de probabilité lorsque les réservoirs se vident. Par conséquent, le comportement du système lorsque la pompe est bloquée-on à un pourcentage élevé est plus proche de son comportement normal en remplissage plutôt qu'en vidange. A contrario, le comportement avec la pompe bloquée-on à un faible pourcentage est plus proche du comportement normal en vidange plutôt qu'en remplissage.

La figure 4.13(a) montre l'indicateur « I » représentant la différence entre la distribution de probabilité associée au système dans son mode de fonctionnement normal et celle associée à son mode de fonctionnement dans la présence d'un défaut où la pompe P_1 est *bloquée-on* à 20% de son débit nominal. Cet indicateur est calculé pour une fenêtre temporelle « *W* » d'une taille de 10 patterns, un pattern représentant une date d'occurrence d'un événement obtenu durant un cycle de fonctionnement.

Afin d'illustrer nos propos, nous avons utilisé 140 patterns. Les 40 premiers patterns correspondent au fonctionnement normal du système, tandis que les 100 patterns qui suivent représentent le fonctionnement du système dans le cas où la pompe P_1 est bloquée-on à 20% de son débit. Nous remarquons que l'indicateur a une faible valeur pour les quatre premières fenêtres temporelle. Ce résultat est logique puisque ces patterns ont les mêmes caractéristiques que ceux de la distribution de probabilité (*DP*) correspondant au fonctionnement normal. En revanche, à la fin de la cinquième fenêtre temporelle, on peut observer une croissance importante de la valeur de cet indicateur. Cette croissance indique que les nouveaux patterns ont des caractéristiques différentes des patterns obtenus lors du fonctionnement normal du système. Les autres fenêtres temporelles ne comportent aucun changement dans les caractéristiques de la distribution de probabilité pour les patterns de la cinquième fenêtre temporelle. Afin de confirmer nos résultats, nous avons utilisé 140

nouveaux patterns (Figure 4.13(b)). Mais cette fois-ci, les 40 premiers patterns représentent un défaut sur la pompe P_1, où elle est *bloquée-on* à 20% de son débit, tandis que les 100 patterns suivant ont des nouvelles caractéristiques liées à un défaut sur la pompe où elle est *bloquée-on* à 40% de son débit. La figure. 4.13(b) montre bien qu'il est possible de différencier les deux défauts.

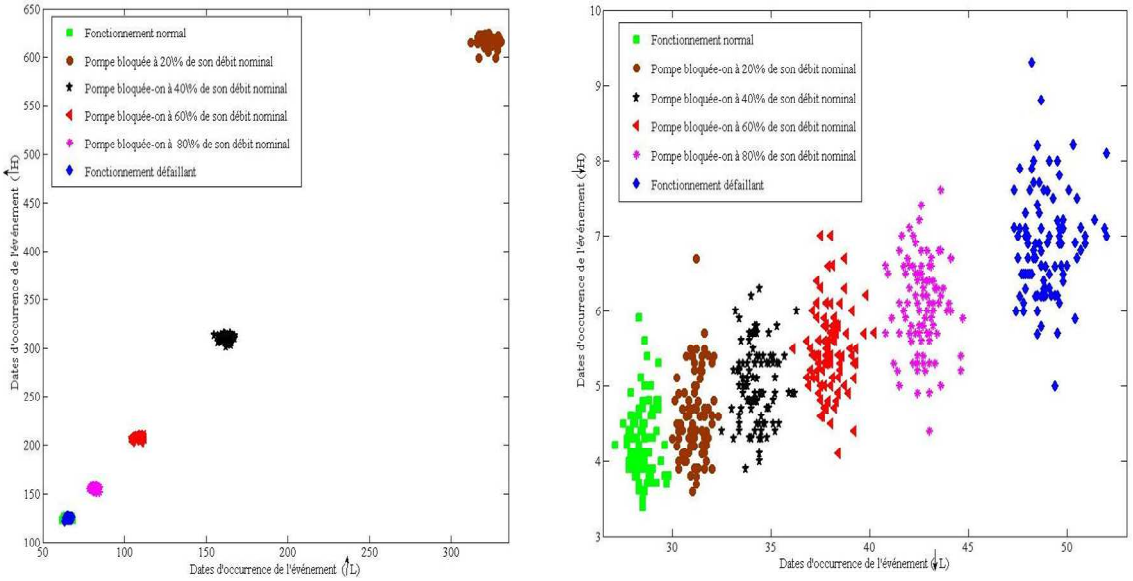


(a) : Indicateurs « *I* » dans le cas de fonctionnement normal et le cas de pompe bloquée-on à 20% de son débit nominal.

(b) : Indicateurs « *I* » dans le cas de pompe bloquée-on à 20% de son débit nominal et le cas de pompe bloquée-on à 40%.

Figure 4.13. Indicateur "I" représentant la différence entre les distributions de probabilité pour les événements ($\uparrow B_1$, $\uparrow H_1$, $\downarrow B_1$, $\downarrow H_1$) dans le cas où la pompe est bloquée-on à 20% et 40% de son débit.

La figure 4.14 représente la dérive de la classe correspondant au mode de fonctionnement normal vers la classe correspondant au mode de fonctionnement défaillant du système (pompe P_1 complètement bloquée-on). En fait, l'espace de représentation des différentes classes est formé par des événements qui sont la conséquence de l'envoie de la même commande. Dans notre cas, nous avons deux espaces de représentation, le premier espace, qui représente le remplissage du réservoir R_1, est formé par les deux événements $\uparrow B_1$ et $\uparrow H_1$ que leur occurrence est lié à l'envoie de la commande $\uparrow P_1$, tandis que le deuxième espace, qui représente la vidange du réservoir, est formé par les deux événements $\downarrow B_1$ et $\downarrow H_1$ qui sont la conséquence de la commande $\uparrow V_1$. A partir de la figure 4.14, nous remarquons que durant cette dérive le système passe par plusieurs régions représentants ses modes de fonctionnement dégradés avant qu'il n'arrive au mode de fonctionnement défaillant où il ne pourra plus remplir sa fonction.



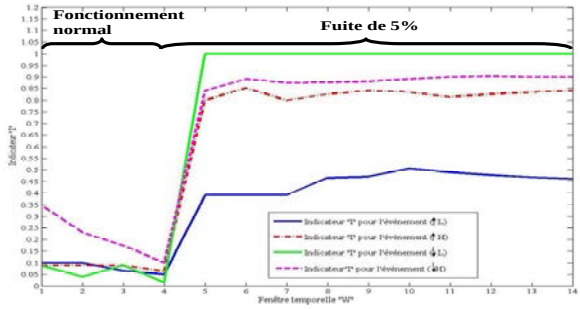
(a) Régions de fonctionnement de la pompe par les événements ($\uparrow B_1$, $\uparrow H_1$) (b) Régions de fonctionnement de la pompe par les événements ($\downarrow B_1$, $\downarrow H_1$)

Figure 4.14. Suivi de dérive dans le système de deux reservoirs

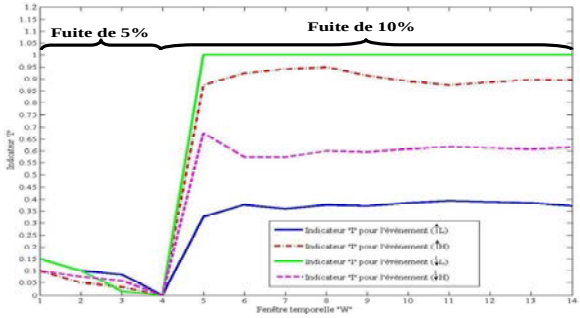
☞ ***Défaut sur le process « fuite de réservoir »***

Le diagnostic d'une dérive peut être effectué également dans le cas de défauts dans le processus lui-même. Pour cela, nous avons appliqué notre approche sur le système de deux réservoirs en simulant un défaut de fuite de faible débit dans le réservoir R_1. Le suivi de la dérive du fonctionnement du système vers le mode de fonctionnement défaillant en présence de ce défaut est réalisé de la même manière que pour les équipements de type analogique. Les résultats obtenus lors de la simulation sont donnés par les figures 4.15 et 4.16.

La première figure montre l'indicateur de dégradation pour deux valeurs différentes de fuite. En premier temps, l'indicateur « I » est calculé pour une fuite d'un débit de 5% par rapport au débit de la pompe. Cet indicateur est obtenu également à l'aide d'une fenêtre temporelle « W » d'une taille de 10 patterns. Un changement dans la valeur de l'indicateur indique une dérive du système vers un autre mode de fonctionnement dégradé. Ensuite, l'indicateur « I » est calculé en utilisant les patterns obtenus lors d'un fonctionnement du système en présence d'une fuite d'un débit de 10%. Ces nouveaux patterns sont comparés par ceux obtenus lors d'une fuite de 5%. La figure 4.14 montre les espaces de représentation dans le cas de remplissage, formé par les événements $\uparrow B_1$ et $\uparrow H_1$, et dans le cas de vidange, formé par les événements $\downarrow B_1$ et $\downarrow H_1$.

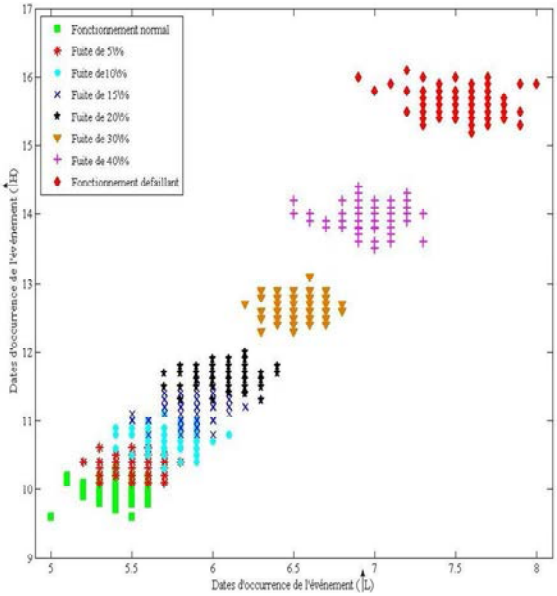


(a) : Indicateur « *I* » dans le cas du fonctionnement normal et d'une fuite de 5%

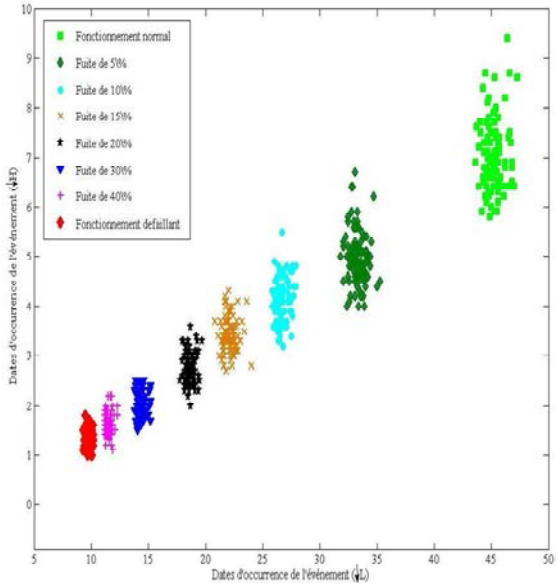


(b) : Indicateur « *I* » dans le cas d'une fuite de 5% et d'une fuite de 10%

Figure 4.15. Indicateur de dégradation "I" pour les événements (↑L1 , ↑H1, ↓L1 , ↓H1) dans le cas d'une fuite de 5% et de 10%.



(a) Régions de fonctionnement formées par les événements ($\uparrow L_1, \uparrow H_1$)



(b) Régions de fonctionnement formées par les événements ($\downarrow L_1, \downarrow H_1$)

Figure 4.16. Régions pour le diagnostic de fuites dans le système de deux réservoirs

4.4.2 Décision et dialogue Homme-Machine

Un module de diagnostic ne doit pas simplement se déployer en interne d'un système. Il doit également rendre des comptes à un utilisateur afin d'informer celui-ci de l'état de fonctionnement du système et l'aider dans sa prise de décision (maintenance, reconfiguration, fonctionnement en mode dégradé …).

Pour cela, suite à l'ensemble de la procédure que nous venons de présenter, une interface graphique simple a été mis en place (Figure 4.17). Cette interface permet de suivre graphiquement la *Template* de fonctionnement normale du système afin de voir son état courant. Une zone d'affichage dynamique permet également de suivre pour notre simulation le remplissage et la vidange du réservoir. Du côté diagnostic, les états des *Templates* sont

représentés ainsi que les conditions d'autorisation et fonctions de non-occurrence. Il en est de même pour les *Chroniques* (ici pour les capteurs B et H) afin de vérifier s'ils restent en fonctionnement normal (en vert) ou si un défaut est apparu (en rouge). L'état des capteurs de niveau est également donné.

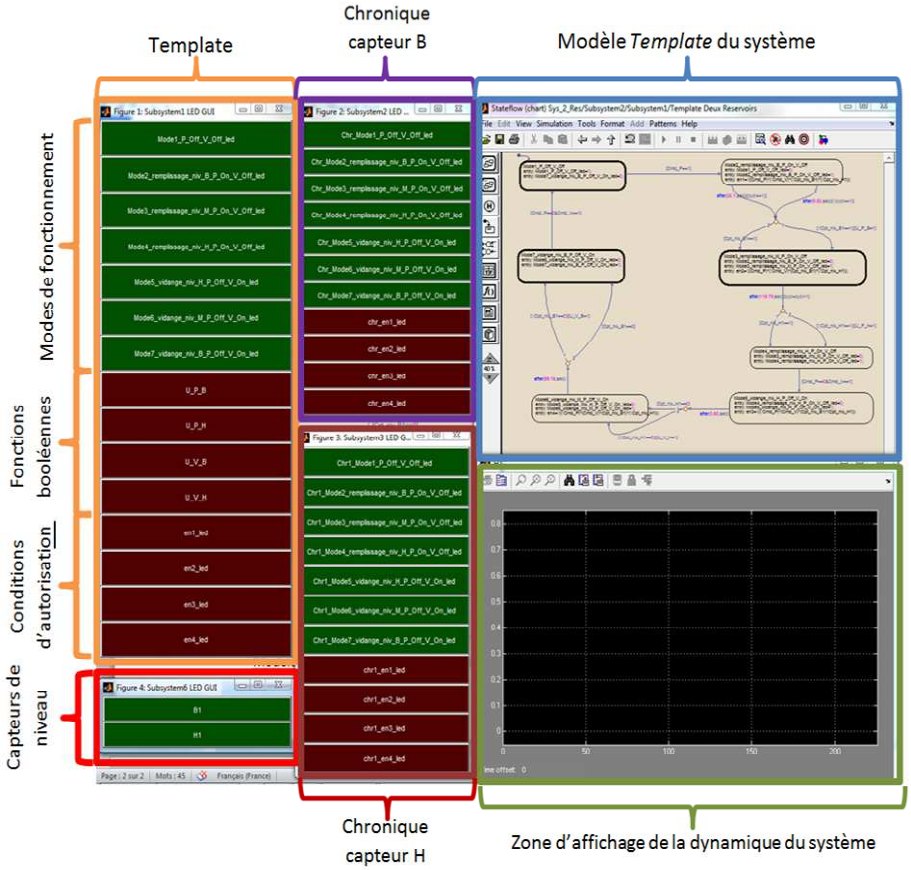


Figure 4.17. Interface du module de diagnostic

Par exemple, si l'on regarde le résultat de la figure 4.18, nous pouvons remarquer un état en rouge de non-occurrence d'un événement attendu sur l'activation du capteur H_1 alors que le réservoir est en cours de remplissage. Dans ce cas, il faut indiquer à l'opérateur qu'il doit arrêter la pompe P_1 rapidement pour éviter un débordement du réservoir.

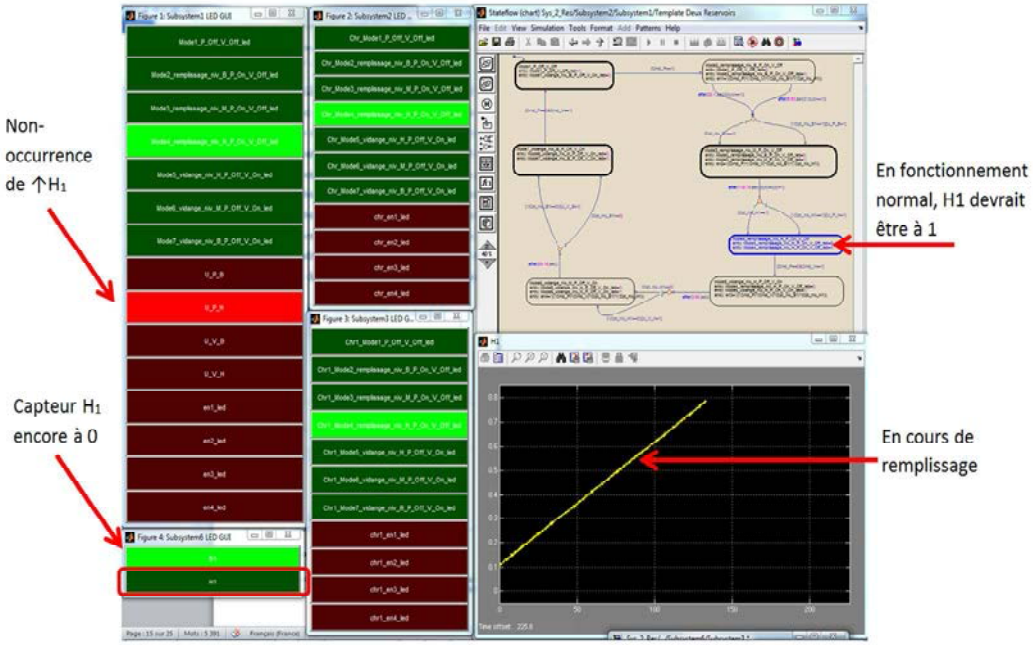


Figure 4.18. Détection d'un défaut « H_1 » bloqué-off dans un réservoir

4.5 Conclusion

Ce chapitre illustre l'approche proposée dans le chapitre 3 au travers d'un benchmark. Ce benchmark a été traité en simulation sous *Matlab/Simulink* afin de permettre l'apprentissage du modèle de fonctionnement normal et ceux anormaux. Cet exemple a été choisi car il permet de traiter le système comme totalement ou partiellement discret et par conséquent de traiter différents types de défauts (équipements ToR ou analogiques, processus).

Lors de la mise en œuvre de ces résultats, une attention a été mise en place concernant le retour d'informations à fournir à l'utilisateur. Une interface homme-machine a été créée sommairement sous l'environnement *Simulink* dans le cadre de nos simulations. Cette interface mérite bien entendu d'être plus développée par la suite au travers d'outils dédiés au suivi et à la surveillance de processus réel (SCADA - Supervisory Control And Data Acquisition). Le contrôle sera alors réalisé par des instruments automatique de mesure et commande dits RTU (Remote Terminal Units) ou par des Automates Programmables Industriels (API ou PLC en anglais pour Programmable Logic Controller).

Dans le cadre de cet exemple de deux réservoirs, une étude a également été effectuée à l'aide de l'outil de classification SALSA (Situation Assessment using LAMDA claSsification

Analysis) développé au sein du LAAS de Toulouse [Kempowsky et al., 2003]. Cet outil d'aide à la détection et au diagnostic de défaillances se base sur les mesures hors ou en ligne issues du procédé afin d'identifier les situations du système. SALSA s'appuie sur la méthode LAMDA (Learning Algorithm for Multivariable Data Analysis) qui est une méthode de classification floue avec apprentissage. Une phase hors ligne permet notamment la construction d'un modèle de référence du comportement du système surveillé sous la forme de classes (phase d'apprentissage). Cette étude, après paramétrage des caractéristiques du logiciel, a montré une cohérence avec nos modèles de *Template* et *Chroniques*. Elle a été réalisée à travers une simulation d'un réservoir modélisé sous ProcesSim (www.processim.hecfh.be) et commandée par le simulateur d'API de Unity Pro (www.schneider-electric.fr). Nous avons choisi de ne pas présenter cette étude dans ce mémoire.

Conclusion Générale et Perspectives

Conclusion Générale

Les travaux présentés dans ce mémoire de thèse représentent une contribution au problème de diagnostic des Systèmes à Evénements Discrets (SEDs). L'objectif de notre travail a été dans un premier temps une proposition d'une démarche de diagnostic en exploitant l'aspect temporel caractérisant l'occurrence des événements. Pour cela, le système a été modélisé par des graphes temporels appartenant au formalisme des automates temporisés. L'approche est conçue selon une architecture décentralisée afin d'éviter toute explosion combinatoire dans la construction des modèles. Elle a permis la détection et localisation des défauts abruptes survenant sur les équipements notamment en combinant des conditions d'autorisation d'événements et des fonctions de non-occurrence d'événements.

Dans un second temps, nous avons voulu considérer les défauts graduels issus du process lui-même. Pour cela, les contraintes temporelles exprimant les dates d'occurrence des événements dans les *Templates* et les *Chroniques* sont modélisées par des distributions de probabilités (*DPs*). Celles-ci sont utilisées afin de caractériser un fonctionnement normal, dégradé ou défaillant de chaque sous-système avec un certain degré de certitude. Cette identification du fonctionnement est représentée par la valeur d'un indicateur de dégradation *I*.

L'ensemble a été illustré dans le chapitre 4 autour d'un benchmark simulé sous *Matlab/Simulink.*

Perspectives

Au terme du travail présenté dans ce manuscrit, plusieurs perspectives peuvent être discernées à court terme ou à long terme. Elles permettront une prolongation de l'étude menée pendant ces années de thèse. Ces perspectives sont représentées de la manière suivante :

Dans un premier temps, il est envisagé d'implémenter la proposée sur un site réel. Pour cela, la station « poste de mélange » de l'atelier flexible *CellFlex* de l'Université de Reims Champagne-Ardenne est à l'étude (Figure 1) (www.univ-reims.fr/site/laboratoires/meserp/accueil,9485,17409.html).

Cette station a notamment l'avantage d'être très proche du benchmark étudié en chapitre 4. A savoir, c'est un système composé de 3 cuves de matière première avec 3 électrovannes ToR en sortie, une pompe amenant les différents liquides dans un réservoir tampon. Chaque cuve est instrumentée de capteurs de niveau haut et bas ToR et d'un capteur flotteur anti-débordement. Un débitmètre est également présent en sortie de pompe. Une IHM via un pupitre tactile est également disponible. Ce travail mettra en avant d'autres problématiques comme la mise en œuvre dynamique des calculs en ligne de distributions de probabilité et d'indicateurs de dégradation.

Figure 1. Poste de mélange de la platforme Cellflex.

Nous avons pu constater dans le chapitre 2 que les performances des approches de diagnostic ont besoin d'être évaluées et que de nombreuses notions autour de la diagnosticabilité existaient. Il convient donc de vérifier la capacité de diagnostic de l'approche proposée à travers également le formalisme d'une de ces notions. Il faut par conséquent adapter et définir des indicateurs de performance pertinents. Pour cela, les approches de vérification par Model-Checking pourront être envisagées afin de vérifier l'atteignabilité de situations défaillantes ou dégradées dans un système. Par ailleurs, il est possible de retourner un indicateur concernant la couverture de diagnostic, ou Diagnotic Coverage (Figure 2), d'une approche par rapport à un système traduisant le rapport de probabilité de défaillances dangereuses détectées (dd) à la probabilité de toutes les défaillances dangereuses (dd + du).

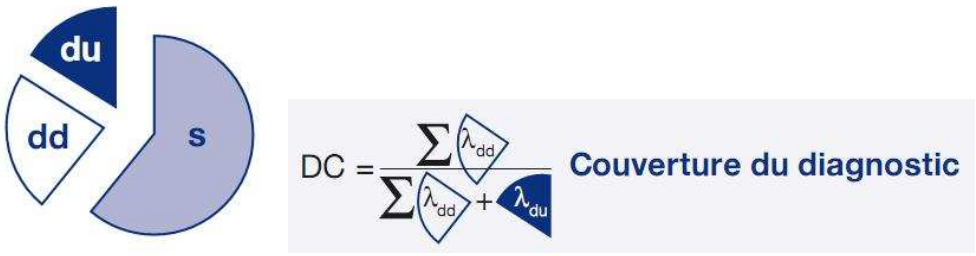


Figure 2. Représentation de l'indicateur de couverture de diagnostic

Après ce travail à court terme, il nous semble intéressant d'entamer une étude comparative de l'approche proposée avec d'autres de la littérature se basant également sur l'utilisation de modèles temporisé. L'approche de diagnostic modulaire à base de Signatures Temporelles Causales (STC) développé dans [Saddem et al., 2012] semble très proche de notre étude. Cette comparaison va nous permettre de dévoiler les points forts et faibles de notre approche. L'étude comparative peut être également le moyen d'explorer les différentes méthodes d'apprentissage pour la construction des distributions de probabilité afin d'apporter plus de précision à notre indicateur de dégradation. En fait, une étude dans ce sens a été déjà engagée par rapport à la méthode *SALSA* où des débuts de résultats ont été obtenus.

La notion d'indépendance développée au sein du chapitre 3 dispose de deux hypothèses fortes autour de la non-propagation de défauts entre sous-systèmes. Cependant, bien que des équipements peuvent être indépendants physiquement/géographiquement, ils peuvent tout de même être corrélés non pas par leurs événements mais par le produit lui-même. Il conviendrait donc de construire un modèle de produit afin de prendre en compte celui-ci dans une nouvelle définition de la notion d'indépendance. Par ailleurs, si le système ne peut se décomposer en sous-systèmes indépendants, il serait intéressant de définir des notions de dépendance « faible » et « forte » entre sous-systèmes. Ceci en vue de ne pas avoir à repasser par une structure centralisée de diagnostic mais d'établir des sous-systèmes pouvant communiquer entre eux. La structure deviendrait proche d'une structure distribuée ou hiérarchique. Un protocole de communication des décisions serait alors nécessaire.

Enfin, dans le cadre de la politique de recherche du groupe AUTO du laboratoire CReSTIC, nous envisageons d'étendre l'approche de diagnostic aux Systèmes à Dynamiques Hybrides (SDHs). Ce travail permettra un rapprochement des équipes travaillant sur les SEDs et les Systèmes Continus. L'objectif est d'élaborer un module de diagnostic prenant en

considération la dynamique continue du système en sus de celle discrète. Cela nécessite évidement une construction de modèles appropriés en se basant par exemple sur le formalisme des automates temporisés hybrides.

Bibliographie

A

[Alligood et al., 1997] Alligood KT, Sauer TD, York JA, « Chaos: an Introduction to Dynamical Systems ». Springer, 1997.

[Alonso-Gonzalez et al., 2010] Alonso-Gonzalez C, Moya N, Biswas G, « Factoring dynamic bayes networks using possible conflicts », In Proceeding of the 21th International Workshop on Principles of Diagnosis (DX10), 2010.

[Alur, 1991] Alur R, « Techniques for automatic verification of real-time systems ». Ph.D. thesis, Stanford University, USA, 1991.

[Antsaklis et al., 1999] Antsaklis PJ, Kohn Wolf, Lemmon MD, Nerode A, Sastry S, « Hybrid Systems V ». Lecture Notes in Computer Science 1567, Springer 1999, ISBN 3-540-65643-X.

B

[Balemi et al., 1993] Balemi S, Hoffmann G.J, Gyugyi P, Wong-Toi H, Franklin G.F, « Supervisory control of a rapid thermal multiprocessor », IEEE Transactions on Automatic Control, vol. 38, n°7, p1040-1059, 1993.

[Barrett et Lafortune, 2000] Barrett G, Lafortune S, « Decentralised supervisory control with communicating controllers », IEEE Transactions on Automatic Control, 45, 1620:1638, 2000.

[Basile et al., 2009] Basile F, Chiacchio P, De Tommasi G. « An efficient approach for online diagnosis of discrete event systems ». IEEE Transactions on Automatic Control, Vol.54, N°4, 2009.

[Benveniste et al., 2003] Benveniste A, Fabre E, Haar E, Jard C, « Diagnosis of asynchronous discrete event systems: A net unfolding approach », IEEE Transactions on Automatic Control, 48(5):714–727, 2003.

[Blanchard, 1979] Blanchard M, « Comprendre, maîtriser et appliquer le GRAFCET », CEPADUES Editions, Collection Automatisation et Production, 1979,

[Blanke et al., 2003] Blanke O, Landis T, Mermoud C, Spinelli L, Safran AB, «Direction-selective motion blindness after unilateral posterior brain damage ». Eur J Neurosci; 18: 709-22, 2003.

[Boudaoud, 1997] Boudaoud N., « Conception d'un système de diagnostic adaptatif en ligne pour la surveillance des systèmes évolutifs », Thèse de Doctorat, Université de Technologie de Compiègne, France, 1997.

[Bousson et al., 1993] Bousson C, Gaborit P, Ghallab M, « Situation recognition: Representation and algorithms », In Proceedings of the 13th IJCAI, 1993.

[Branicky , 1994] Branicky MS, « Analysis of continuous switching systems: Theory and examples », Proceedings of the American Control Conference, Baltimore, MD, June, pp: 3110-3114, 1994.

[Brauer et Nohel, 1969] Brauer F, Nohel JS, « The qualitative theory of ordinary differential equations ». W. A. Benjamin, Inc., New York, 1969.

[Breese et al., 1988] Breese JS, Horwitz EJ, Henrion M, « Decision theory in expert systems and artificial intelligence », International Journal of Approximate Reasoning, 1988.

C

[Cassandras et Lafortune, 2008] Cassandras CG; Lafortune S, « Introduction to Discrete Event Systems ». 2eme edition. Boston (MA): Kluwer Academic Publishers. 2008.

[CEI/IEC 1131-3, 1993] Automates programmables-Partie 3 : Langages de programmation, Norme Française et Norme Européenne NF EN 61131-3, CENELEC ; Version française de la norme CEI/IEC 1131-3, Juillet 1993, 202 pages.

[CEI/IEC 848, 1988] Établissement des diagrammes fonctionnels pour systèmes de commande / Preparation of function charts for control systems. Norme Internationale / International Standard, CEI/IEC 848, décembre 1988, CEI-Genève-Suisse.

[Cha et Srihari, 2002] Cha S, Srihari S. N, « On Measuring the Distance between Histograms », in Pattern Recognition, Vol 35(6), pp: 1355-1370, 2002.

[Chow et Fang, 1994] Chow TWS, Fang Y, « An iterative learning control method for continuous-time systems based on 2-D system theory », IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications, vol.45, n°.6, pp:683 -689, 1998.

[Cohen et Coon, 1953] Cohen GH, Coon GA, « Theoretical consideration of retarded control ». Trans. A.S.M.E., Vol. 75(No. 1): pp. 827–834, 1953.

[Contant et al., 2004] Contant O, Lafortune S, Teneketzis D, « Diagnosis of modular discrete event systems », 7th International Workshop on Discrete Event Systems WODES'04, pp.337-342, 2004.

[Cordier et al., 2007] Cordier M.O, Le Guillou X, Robin S, Rozé L, Vidal T, « Distributed chronicles for on-line diagnosis of web services », 18th International Workshop on the Principles of Diagnosis (DX'07), pp: 37-44, Nashville (USA), 2007.

D

[Dall'Aglio, 1991] Dall'Aglio G, « Fréchet Classes: The Beginnings », Advances in Probability Distributions with Given Marginals, Mathematics and Its Applications, Kluwer, 1991.

[Debouk, 2000] Debouk R, Lafortune S, Teneketzis D, « Coordinated decentralized protocols for failure diagnosis of discrete event systems ». Discrete Event Dynamic Systems, 10 (1-2), 33-86, 2000.

[Dill, 1989] Dill D, « Timing Assumptions and Verification of Finite-State Concurrent Systems ». InProc. of the Workshop on Automatic Verification Methods for Finite State Systems, vol. 407 of Lecture Notes in Computer Science, pp. 197-212. Springer-Verlag, 1989.

[Dotoli et al., 2009] Dotoli D, Fanti MP, Mangini AM. « Fault detection of discrete event systems using Petri nets and integer linear programming ». In Proc. of 17th IFAC World Congress, Seoul, Korea, 2008.

[Dousson, 2002] Dousson C, « Extending and unifying chronicle representation with event counters », Proceedings of the 15th International Conference on Artificial Intelligence, 257-261, 2002.

F

[Freyermuth, 1991] Freyermuth B, « Knowledge based incipienct fault diagnosis of industrial robots », In Proceedings of SAFEPROCESS' 91 International Conference on Fault Detection, Supervision and Safety for Technical Processes, volume 2, Baden-Baden, Germany, 1991.

[Friedman et Kandel, 1999] Friedman M, Kandel A, « Introduction to pattern recognition – statistical, structural, neural and fuzzy logic approaches ». Imperial College Press, London 1999.

G

[Genc et Lafortune, 2003] Genc S, Lafortune S, « Distributed diagnosis of discrete-event systems using Petri nets », Lecture Notes in Computer Science, 2679, 316-336, 2003.

[Genc et Lafortune, 2007] Genc S, Lafortune S, « Distributed diagnosis of place-bordered Petri nets », IEEE Transactions on Automatic Science and Engineering, 4(2):206–219, 2007.

[Gertler, 1998] Gertler J, « Fault detection and diagnosis in Engineering Systems », Marcel Dekker, New York, 1998.

[Ghallab, 1996] Ghallab M, « On chronicles: representation, on-line recognition and learning », Conference on Principles of Knowledge Representation and Reasoning (KR'96), pp: 597-606, Cambridge (USA), 1996.

[Guerraz et Dousson, 2004] Guerraz B, Dousson C, « Chronicles construction starting from the fault model of the system to diagnose », 15th International Workshop on the Principles of Diagnosis (DX'04), pp: 51–56, Carcassonne, France, 2004.

H

[Heiming et al., 1997] Heiming B, Lunze J, Teneketzis D, « Parallel Knowledge-based process diagnosis applied to a local power station plant », In proceeding safeprocess, pp. 1114-1120, 1997.

[Hogg et Iverson, 1991] Hogg J, Iverson R, « Representing concurrent communication systems », In proceeding of the Workshop on Object-Based Concurrent Programming, pages: 37-39, 1991.

[Holloway et Chand, 1994] Holloway LE, Chand S, « Time templates for discrete event fault monitoring in manufacturing systems », In Proceedings of the American Control Conference, pages 701-706, 1994.

I

[Isermann et Ballé, 1997] Isermann R, Ballé P, « Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes ». Control Engineering Practice, 5(5), pp. 709-719, 1997.

[Isermann, 2005] Isermann R, « Model-Based Fault Detection And Diagnosis-Status And Applications » Annual Reviews in Control 29, 71–85 – Elsevier, 2005.

K

[Kempowsky et al., 2003] Kempowsky T, Aguilar-Martin J, Subias A, Le Lann M.V,. « Classification tool based on interactivity between expertise and self learning techniques », In Proceedings of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'2003), Washington DC (USA), 2003.

[Kramer et Mah, 1993] Kramer MA., Mah RSH., « Model-Based Monitoring », Proceedings of the second International Conference on Foundations Of Computer-Aided Process Operations (FOCAPO), pp. 45-68,1993.

L

[Lapp et Powers, 1977] Lapp S, Powers G, « Computer aided synthesis of fault trees ». IEEE Transactions on Reliability, 26(1), 2–13, 1977.

[Lee et al., 1985] Lee WS, Grosh DL, Tillman FA, Lie C.H, « Fault tree analysis, methods, and applications -A review ». IEEE Transactions on Reliability, 34,194–203, 1985.

[Lefebvre et Delherm, 2007] Lefebvre D, Delherm C, « Diagnosis of DES with Petri net models », IEEE Transactions on Automatic Science and Engineering, 4(1):114–118, 2007.

[Lin et Wonham, 1988(a)] Lin F, Wonham WM, « Decentralized supervisory control of discrete-event systems », Information Sciences 44(3), 199-224, 1988.

[Lin et Wonham, 1988(b)] Lin F, Wonham WM, « On observability of discrete event systems », Information Sciences 44(3), 173-198, 1988.

[Lin et Wonham, 1990] Lin F, Wonham WM, « Decentralized control and coordination of discrete event systems with partial observation », IEEE Transactions on Automatic Control 35(12), 1330-1337, 1990.

[Lunze et Schröder, 2000] Lunze J, Schröder J, « Sensor and actuator fault diagnosis of systems with discrete inputs and outputs », Automatica, 2000.

[Ly et al., 2009] Ly F, Simeu-Abazi Z, Leger JB, « Terminologie Maintenance: bilan ». Groupe de travail « Maintenance » de l'association Diag21.

M

[Malki et al., 2009] Malki N., Philippot A., Sayed-Mouchaweh M., Carré-Ménétrier V., « Independent codiagnosability of discrete event systems using component based-approach: from modelling to diagnosis », 28th European Annual Conference on Human Decision-Making and Manual Control (EAM09), Reims, France, septembre 2009.

[Malki et Sayed-Mouchaweh, 2011(a)] Malki N. et Sayed-Mouchaweh M., « Boolean temporal model-based approach for the diagnosis of Discrete Event Systems », 7th IEEE Conference on Automation Science and Engineering (CASE 2011), Trieste, Italy, 2011.

[Malki et Sayed-Mouchaweh, 2011(b)] Malki N. et Sayed-Mouchaweh M., « Time-based modeling approach for drift monitoring in Discrete Event Systems », 19th Mediterranean Conference on Control and Automation (MED 2011), Corfu, Greece, 2011.

[Milne et al., 1994] Milne R, Nicol C, Ghallab M, Trave-massuyes L, Bousson K, Quevedo J, Dousson C, Aguilar J, Guasch A. « Tiger: real-time situation assessment of dynamic systems », Intelligent Systems Engineering. 1994.

N

[Nimmo, 1995] Nimmo I, « Adequately address abnormal situation operations », Chemical Engineering Progress, 91(9), 36-45, 1995.

P

[Pandalai et Holloway, 2000] Pandalai DN, Holloway LE, « Template languages for fault monitoring of timed discrete event processes ». IEEE transactions on automatic control, 45(5), 868-882, 2000.

[Pencolé et al., 2001] Pencolé Y, Cordier MO, Rozé L, « Incremental decentralized diagnosis approach for the supervision of a telecommunication network », In Proc International Workshop on Principles of Diagnosis (DX01), 2001.

[Pencolé et Cordier, 2005] Pencolé Y, Cordier MO, « A formal framework for the decentralised diagnosis of large scale discrete event systems and its application telecommunication networks », Artificial Intelligence, 164: 121-170, 2005.

[Pencolé et Subias, 2009] Pencolé Y, Subias A, « A chronicle-based diagnosability approach for discrete timed-event systems: Application to web-services », Journal of Universal Computer Science, vol.15, pp. 3246-3272, 2009.

[Petri, 1962] Petri CA, « Kommunication mit Automaten ». PhD thesis, Institut fur Instrumentelle Mathematik, Schriften des IIM, No. 3, Bonn, Germany, 1962.

[Philippot et al, 2005] Philippot A, Tajer A, Gellot F, Carré-Ménétrier V, « Méthodologie de modélisation dans le cadre de la synthèse formelle des SED », Revue électronique des sciences et technologies de l'automatique, e-STA, 2(2), 2005.

[Philippot et al., 2006] Philippot A, Sayed Mouchaweh M, Carré-Ménétrier V, Riera B, « Decentralized approach to diagnose manufacturing systems », IMACS Multiconference on Computational Engineering in Systems Applications (CESA06), vol. 1, pp: 912-918, Beijing, China, 2006.

[Philippot et al., 2007] Philippot A, Sayed Mouchaweh M, Carré-Ménétrier V, « Unconditional Decentralized Structure for the fault diagnosis of Discrete Event Systems », 1st IFAC Workshop on Dependable Control of Discrete-event Systems (DCDS'07), Cachan, France, 2007.

[Prock, 1991] Prock J, « A new technique for fault detection using Petri nets », IEEE Transactions on Automatic Control, 27(2), 239-245, 1991.

Q

[Qiu, 2005] Qiu W, « Decentralized / distributed failure diagnosis and supervisory control of discrete event systems », Thesis, Iowa State University, 2005.

R

[Ramadge et Wonham, 1987] Ramadge PJ, Wonham WM, « Supervisory control of a class of discrete event systems ». SIAM Journal on Control and Optimization, 25:206–230, 1987.

[Ramadge et Wonham, 1989] Ramadge PJ, Wonham WM, « The Control of Discrete Event Systems », Proceedings of the IEEE, Vol.77, No.1, Janvier 1989.

[Reiter, 1987] Reiter R, « A theory of diagnosis from first principles », Artificial Intelligence, 32(1), 57-95, 1987.

[René et Alla, 1992] René D, Alla, « Du Grafcet aux réseaux de Petri, », Paris, Hermès, 1992, 2eme édition, ISBN 2-86601-325-5.

[Ricks et Mengshoel, 2010] Ricks BW, Mengshoel OJ, « Diagnosing intermittent and persistent faults using static bayesian networks », In Proc of the 21st International Workshop on Principles of Diagnosis (DX-10), 2010.

S

[Saddem et al., 2012] Saddem R, Toguyeni A.K.A, Tagina M, « Algorithme d’interprétation d'une base de Signatures Temporelles Causales pour le diagnostic en ligne des systèmes à

évènements discrets », 9ème Conf. Inter. de Modélisation, Optimisation et SIMulation (MOSIM'12), France, 2012.

[Sampath et al., 1995] Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D, « Diagnosability of discrete-event systems », IEEE Transactions on Automatic Control40(9), 1555–1575, 1995.

[Sampath et al., 1996(a)] Sampath M, Lafortune S, Teneketzis D, « Active diagnosis of discrete event systems », IEEE Transactions on Control Systems Technology, 4(2), pp. 105-124, 1996.

[Sampath et al., 1996(b)] Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D, « Failure diagnosis using discrete-event models », IEEE Transactions on Control Systems Technology, 4(2), pp. 105-124, 1996.

[Sampath, 1995] Sampath M. « A discrete Event Systems Approach to Failure Diagnosis », Thesis, University of Michigan, 1995.

[Shallit, 2008] Shallit JO, « A Second Course in Formal Languages and Automata Theory », Cambridge University Press, ISBN 978-0-521-86572-2, 2008.

[Smith, 1958] Smith OJM, « Closer control of loops with dead time ». Chemical Engineering Progress, 53(5) :217–219, 1958.

[Sørensen, 1948] Sørensen T, « A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons », Académie royal danois des sciences, 5 (4): 1-34, 1948.

[Su et Wonham, 2000] Su R, Wonham WM, « Decentralised fault diagnosis for discrete event systems », In proceeding CISS, TP Transactions on Automatic Control, 45, 1620:1638.:1-6, 2000.

T

[Tzafestas et al., 1990] Tzafestas S, Watanabe K. « Modern approaches to system/sensor fault detectionand diagnosis », Journal A, 31(4), 1990.

U

[UTE C03-191, 1993] Établissement des diagrammes fonctionnels pour systèmes de commande-Diagramme fonctionnel grafcet. Extension des concepts de base, Documentation de référence (documentation et symboles graphiques), UTE C03-191, juin 1993, 35 p.

V

[Vachtsevanos et al., 2006] Vachtsevanos G, Lewis F, Roemer M, Hess A, Wu B, « Intelligent Fault Diagnosis and Prognosis for Engineering Systems », John Wiley & Sons, Inc, 2006.

[Viswanadham et Johnson, 1988] Viswanadham N, Johnson TL, « Fault detection and diagnosis of automated manufacturing systems ». In Proc. 27th IEEE Conf. on Decision and Control, Austin, Texas, pages: 2301–2306, 1988.

[Vries, 1990] Vries RD, « An automated methodology for generating a fault tree ». IEEE Transactions on Reliability,39(1),76–86, 1990.

W

[Wang et al., 2005(a)] Wang Y, Yoo T, and Lafortune S, « New Results on Decentralized Diagnosis of Discrete Event Systems », Annual Allerton Conference on Communication, Control and Computing, 2005.

[Wang et al., 2005(b)] Wang Y, Yoo T, Lafortune S, « Decentralized Diagnosis of Discrete Event Systems Using Unconditional and Conditional Decisions », In Conference on Decision and Control (CDC'05). Seville, Spain, 2005.

[Wu et Hadjicostis, 2005] Wu Y, Hadjicostis CN. « Algebraic approaches for fault identification in discrete-event systems », IEEE Trans. Robotics and Automation, 50(12):2048–2053, 2005.

Z

[Zad et al., 1998] Zad SH, Kwong RH, Wonham WM, « Fault diagnosis in discrete event systems », In Proc IEEE Conference on Decision and Control (CDC'98). Pages: 3769-3774, 1998.

[Zad et al., 1999] Zad SH, Kwong RH, Wonham WM, « Fault diagnosis in finite-state automata and timed discrete-event systems », dans Proceedings of the 38th IEEE Conference on Decision and Control (CDC'99), 1999.

[Zad et al., 2003] Zad SH, Kwong R, Wonham WM, « Fault diagnosis in discrete-event systems: framework and model reduction », IEEE Transactions on Automatic Control, 48(7), July, pp: 1199-1212, 2003.

[Zaytoon, 2001] Zaytoon J, « Systemes Dynamique Hybrides ». Hermès sciences publications, Paris, 2001.

ANNEXES

ANNEXE 1 : GRAFCET

Étape : L'étape représente un état dans lequel le système est invariant vis-à-vis de ses entrées/sorties. Elle peut être active ou inactive. L'état du Grafcet est défini, à un instant donné, par l'ensemble de ses étapes actives. L'étape correspondant à l'initialisation du système est appelée étape initiale. Elle est représentée par un double carré. Il peut y avoir plusieurs étapes initiales dans un même Grafcet.

Transition : La transition traduit la possibilité d'évolution d'une étape vers une autre. Cette évolution est la conséquence du franchissement de la transition. Une transition est validée si toutes ses étapes immédiatement en amont sont actives.

Liaison orientée : Une liaison orientée relie une étape à une transition et inversement. Elle indique les configurations atteignables à partir d'un état donné.

Action : Une action (ou ordre) est représentée par une forme littérale ou symbolique dans un rectangle associée à une étape.

Réceptivité : Une réceptivité (ou condition de transition) peut être décrite par un texte, une expression booléenne ou par des symboles graphiques normalisés. Elle est associée à une transition et correspondant à ces conditions de validation.

L'évolution d'un Grafcet s'appuie essentiellement sur 2 règles de syntaxe et 5 règles d'évolution :

Règles de syntaxe

L'alternance étape-transition et transition-étape doit toujours être respectée quelle que soit la séquence parcourue.
Deux étapes ou deux transitions ne doivent jamais être reliées par une liaison orientée. La liaison orientée relie obligatoirement une étape à une transition ou une transition à une étape.

Règles d'évolution

Situation initiale (Règle 1)

La situation initiale d'un grafcet caractérise le comportement initial de la partie commande vis à vis de la partie opérative, de l'opérateur et/ou des éléments extérieurs. Elle traduit généralement un comportement de repos.

Franchissement d'une transition (Règle 2)

Une transition est dite validée lorsque toutes les étapes immédiatement précédentes reliées à cette transition sont actives.
Le franchissement d'une transition se produit :
* lorsque la transition est VALIDEE,
* ET QUE la réceptivité associée à cette transition est VRAIE.

Lorsque ces deux conditions sont réunies, la transition devient franchissable et est alors **obligatoirement franchie**

Evolution des étapes actives (Règle 3)

Le franchissement d'une transition entraîne simultanément l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

Evolutions simultanées (Règle 4)

Plusieurs transitions simultanément franchissables sont simultanément franchies.

Cette règle de franchissement simultané permet notamment de décomposer un grafcet en plusieurs diagrammes, tout en assurant de façon rigoureuse leur synchronisation. Dans ce cas, il est indispensable de faire intervenir dans les réceptivités les états actifs des étapes.
La variable « Xi » représente l'état de l'étape i.
Xi=1, l'étape i est active
Xi=0, l'étape i est inactive

Activation et désactivation simultanées d'une étape (Règle 5)

Si, au cours du fonctionnement, la même étape est simultanément activée et désactivée elle res-te active.

ANNEXE 2 : AUTOMATES A ETATS FINIS

Notions de langages

- ***Langages réguliers***

Pour définir un langage régulier on définit d'abord l'expression régulière.

Une expression régulière sur un alphabet $\sum$ est une expression dont les opérandes sont des symboles de $\sum$, et dont les opérateurs sont pris dans l'ensemble {+, · , *}. Par exemple l'expression : $b+a\cdot b*$ est une expression régulière.

Une expression régulière définit un langage sur $\sum$. Chaque symbole « *a* » de $\sum$ correspond au mot *a*. Les opérateurs + et * sont interprétés respectivement comme les opérateurs d'union, de concaténation et de fermeture itérative sur des langages. Par exemple, l'expression régulière : $b + a \cdot b*$ représente le langage: $\{b\} \cup \{a\}.\{b\}^*$

On appelle *langage régulier*, tout langage qui peut être défini par une expression régulière. Par exemple, considérons le langage *L* sur l'alphabet $\sum = \{a,b\}$ qui comprend les chaînes telles que *a* et *b* apparaissent alternativement et telles que *a* apparaît en premier. Nous obtenons : L = {ε, *a, ab, aba, abab, ababa,* …}. Ce langage est régulier puisqu'il peut être décrit par l'expression régulière : $(a.b)^*(\varepsilon+a)$.

- ***Opérations sur les langages***

Soit L_1 et L_2 deux langages construits sur un même alphabet Σ. On peut définir les opérations élémentaires suivantes :

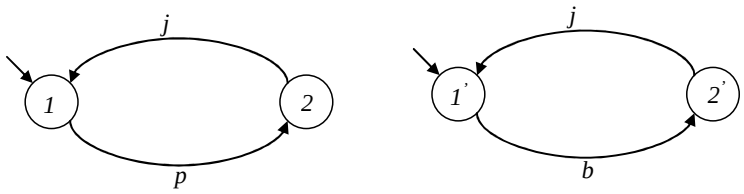
- L'*union* de L_1 et L_2, notée, $L_1 \cup L_2$, est le langage contenant toutes les chaînes qui sont soit contenues dans L_1, soit dans L_2.
- L'*intersection* de L_1 et L_2 notée, $L_1 \cap L_2$, est le langage contenant toutes les chaînes qui sont soit contenues à la fois dans L_1 et dans L_2.

- La *concaténation* de L_1 et L_2 notée, $L_1 \cdot L_2$, est le langage contenant toutes les chaînes formées d'une chaîne de L_1 suivie d'une chaîne L_2.
- La *fermeture itérative* d'un langage L_1, notée L_1^*, est l'ensemble des chaînes formées par une concaténation finie de chaînes de L_1.

Opérations sur les automates

- ***Composition synchrone***

Exemple 1. Prenons le fonctionnement d'un système décrit par les 2 modèles automates de la figure 1.



(a) : Automate A_1 (b) : Automate A_2

Figure 1. Modèles automates avant synchronisation .

L'automate A_1 de la figure 1(a) est construit sur un alphabet de 2 symboles $\Sigma_1=\{p, j\}$. Nous pouvons remarquer que $L_1= \{\varepsilon, p, pj, pjp,\ldots\}$. Supposons que le symbole « *p* » modélise l'événement « introduction d'une *p*ièce » dans une machine et que « *j* » modélise l'événement « libération d'un *j*eton ». Alors, dans la figure 1(b), A_1 est construit sur un alphabet également de 2 symboles $\Sigma_2=\{d, j\}$ et génère un langage $L_2= \{\varepsilon, b, bj, bjb,\ldots\}$. Le symbole « *b* » correspond à l'événement « bouton machine appuyé par l'utilisateur » et le symbole « *j* » a la même signification que dans A_1. Cet automate modélise la contrainte de fonctionnement que le jeton est délivré seulement si une pièce a préalablement été introduit dans la machine (nous supposons que les événements *p* et *j* apparaissent toujours alternativement). À partir des deux modèles automates A_1 et A_2, on peut déduire que $\Sigma_1 \cap \Sigma_2 = j$ et $\Sigma = \Sigma_1 \cup \Sigma_2 = \{b, j, p\}$.

Un modèle global *A* de notre machine issu de l'opération de composition synchrone des deux automates finis A_1 et A_2 est représenté dans la figure 2. Ce modèle permet d'exprimer la conjonction des deux contraintes de fonctionnement définies par A_1 et par A_2.

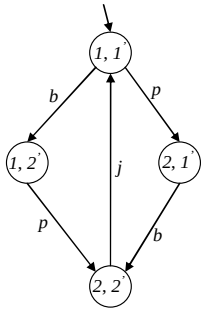
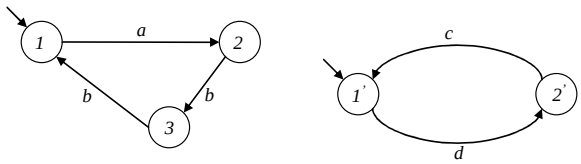


Figure 2. Modèle automate A pour la machine après synchronisation de A1 et A2.

Dans l'automate *A*, un état correspond à un couple (*q, q'*) où *q* est un état de A_1 et *q'* est un état de A_2. L'état initial (*1, 1'*) est l'état correspondant aux états initiaux de A_1 et de A_2. L'automate *A* est construit sur un alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$ qui est la réunion des alphabets de A_1 et A_2.

- ***Composition asynchrone***

Prenons l'exemple de la figure 3 où deux automates A_1 et A_2 sont représentés avec $\Sigma_1 = \{a, b\}$ et $\Sigma_2 = \{c, d\}$. On en déduit $L_1 = \{\varepsilon, a, ab, abb, \ldots\}$ et $L_2 = \{\varepsilon, c, cd, \ldots\}$ avec $\Sigma_1 \cap \Sigma_2 = 0$ et $\Sigma = \Sigma_1 \cup \Sigma_2 = \{a, b, c, d\}$.



(a) : Automate A_1 (b) : Automate A_2

Figure 3. Modèles automates avant composition asynchrone .

A partir de leur état initial, les deux automates peuvent évoluer soit par l'événement *a* avec la relation (1), soit par l'événement *c* par la relation (2). A partir de son état initial (1,1'), la composition asynchrone fait évoluer l'automate soit en état (2,1'), soit en état (1,2'). L'automate de composition asynchrone $A_1||A_2$ modélisant le comportement global est représenté en figure 4.

$$\delta((q_1, q_2), \boldsymbol{\sigma}) = (q_1', q_2) \text{ si } \delta_1(q_1, \boldsymbol{\sigma}) = q_1' \text{ pour } \boldsymbol{\sigma} \in \Sigma_1 \quad (1)$$

$$\delta((q_1, q_2), \boldsymbol{\sigma}) = (q_1, q_2') \text{ si } \delta_2(q_2, \boldsymbol{\sigma}) = q_2' \text{ pour } \boldsymbol{\sigma} \in \Sigma_2 \quad (2)$$

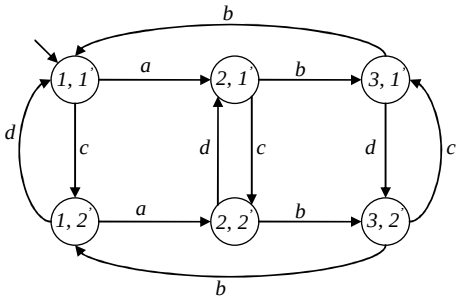


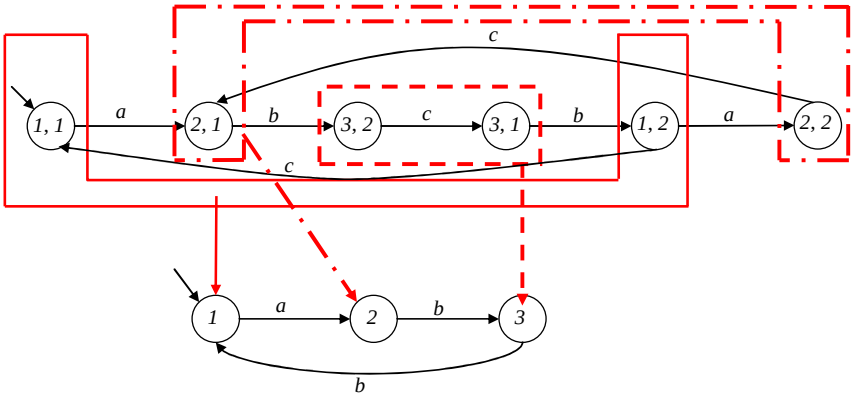
Figure 4. Modèle automate A pour après la composition asynchrone de A1 et A2.

- ***Projection naturelle***

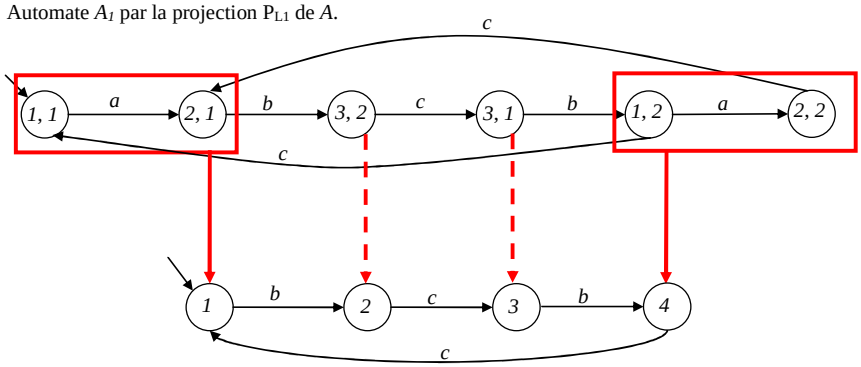
Exemple 2. Prenons l'exemple de la figure 5, un automate *A* de langage L, où $L_1 \subseteq L$ et $L_2 \subseteq L$, avec $\Sigma = \Sigma_1 \cup \Sigma_2$ l'ensemble des événements et $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$ est l'ensemble des événements observables sur l'automate *A*.

La projection naturelle sur le langage L_1, notée "$P_{L1} : \Sigma_1^* \rightarrow \Sigma_{o1}^*$", de l'automate *A* permet d'obtenir l'automate A_1. Elle consiste à ne prendre sur *A* que les événements de L_1 comme observables. Dès lors, tous les autres événements ne sont pas considérés. La figure 5(a) montre un exemple d'obtention d'un automate A_1 de projection naturelle P_{L1} à partir d'un automate *A* à 6 états. Pour l'ensemble des événements $\Sigma = \{a, b, c\}$ de *A*, la projection P_{L1} consiste à prendre uniquement les événements *a* et *b* comme observables. Il en ressort un automate A_1 à 3 états.

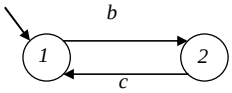
De la même manière, la projection naturelle sur L_2, notée "$P_{L2} : \Sigma_2^* \rightarrow \Sigma_{o2}^*$", de l'automate *A* est l'automate A_2 de langage L_2. Cette fois-ci, ce sont les événements de L_2 sur *A* qui sont considérés comme observables. La figure 5(b) montre l'obtention d'un automate A_2 de projection naturelle P_{L2} à partir d'un automate *A* à 6 états. Pour l'ensemble des événements $\Sigma = \{a, b, c\}$ de *G*, la projection P_{L2} consiste à prendre uniquement les événements *b* et *c* comme observables. Il en ressort un automate A_2 à 4 états qui peut être réduit à 2 états (Figure 5(c)).



(a) Automate A_1 par la projection P_{L1} de A.



(b) Automate A_2 par la projection P_{L2} de A.



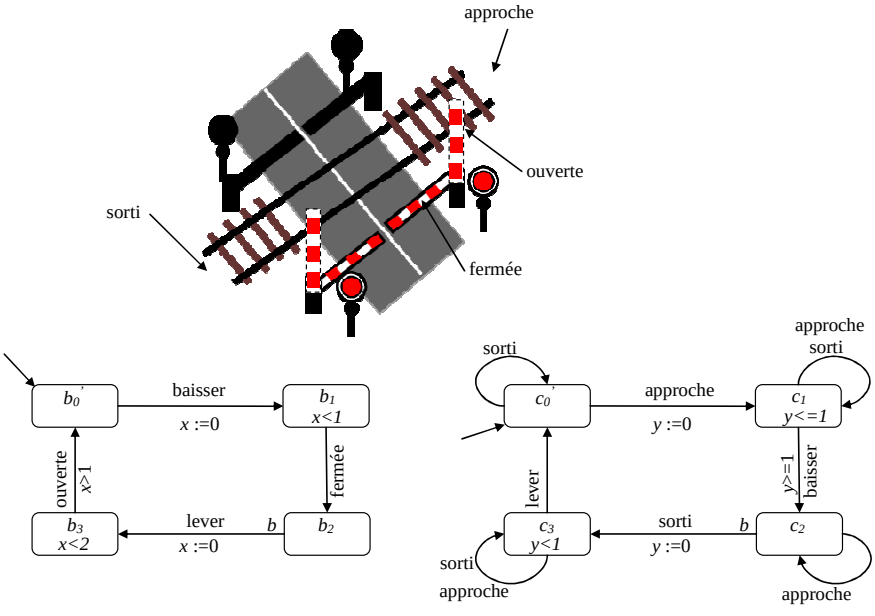
(c) Automate équivalent de A_2.

Figure 5. Exemple de projection naturelle.

ANNEXE 3 : AUTOMATES TEMPORISES

- ***Composition synchrone***

Exemple 3. Pour illustrer une composition synchrone, nous considérons un système de contrôle d'un passage à niveau figure 6. Le système considéré est composé de deux sous-systèmes: contrôleur, barrière. La barrière est modélisée par l'automate temporisé de la figure 6(a). L'ensemble d'événements est donnée par Σ_1= {*baisser*, *lever*, *fermée*, *ouverte*}. La barrière initialement est à l'état b_0 et elle est ouverte. Lorsque la barrière reçoit la commande « *baisser* » envoyé par le contrôleur, elle atteint la position basse en moins d'1 minute. Lorsque le contrôleur envoie la commande « *lever* » à la barrière, cette dernière met entre 1 et 2 minute pour atteindre la position haute. Dans le cas où la barrière ne reçoit aucune commande, elle peut rester indéfiniment en position haute ou basse. L'événement « *fermée* » indique la fin de l'opération de fermeture de la barrière tandis que l'événement « *ouverte* » marque la fin de l'opération d'ouverture. Les contraintes sont exprimées à l'aide d'une horloge *x*.



(a) Automates A_1 représentant la barrière.

(b) Automates A_2 représentant le controleur.

Figure 6. Modèles barrière et contrôleur avant synchronisation.

L'automate modélisant le contrôleur est donné par la figure 6(b). L'ensemble d'événements est défini par Σ_2= {*approche*, *sorti*, *baisser*, *lever*}. L'état de départ du contrôleur est c_0. Lorsqu'il reçoit l'événement *« approche »* indiquant l'arrivé d'un train à la section critique contrôlée, le contrôleur répond, en 1 minute exactement, en envoyant l'événement « *baisser* » à la barrière. Lorsque le train sort de la section critique, le contrôleur reçoit l'événement *« sorti »*. Ensuite, il envoie, en 1 minute au maximum, l'événement *« lever »* à la barrière. Les contraintes sont exprimées à l'aide d'une horloge *y*.

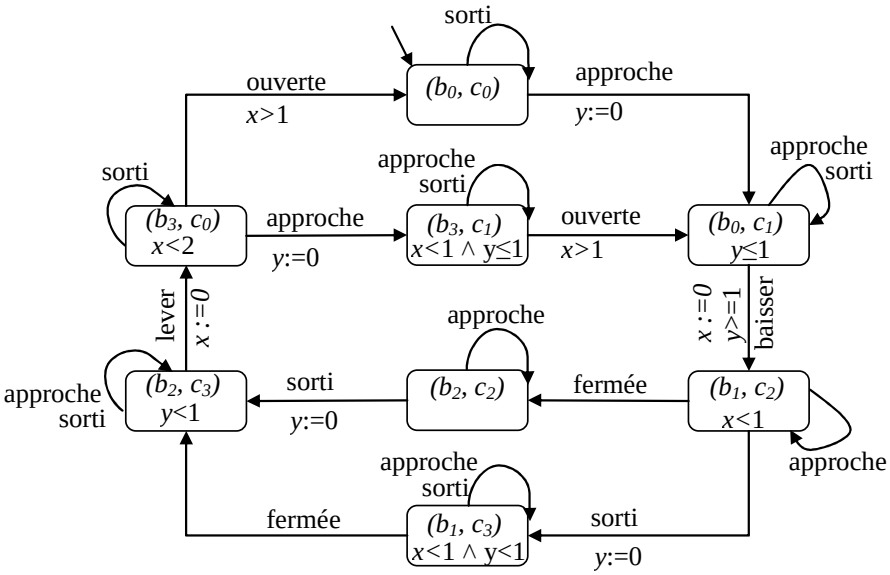


Figure 7. Modèle barrière et contrôleur après la synchronisation.

La figure 7 représente l'automate temporisé *A* qui correspond au système global contrôleur-barrière résultant de la composition synchrone des automates temporisé A_1 et A_2 de la figure 6. Chaque état dans l'automate *A* est représenté par un couple (q, q') où *q* est un état de A_1 et *q'* est un état de A_2. L'alphabet Σ est la réunion des alphabets qui correspondent aux automates A_1 et A_2.

Résumé:

Ce mémoire de thèse présente une contribution au problème de diagnostic des Systèmes à Evénements Discrets (SEDs). Une démarche de diagnostic en exploitant l'aspect temporel caractérisant l'occurrence des événements est proposée. Le système est modélisé par des graphes temporels appartenant au formalisme des automates temporisés. L'approche est conçue selon une architecture décentralisée afin d'éviter toute explosion combinatoire dans la construction des modèles. Elle a permis la détection et localisation des défauts abruptes survenant sur les équipements notamment en combinant des conditions d'autorisation d'événements et des fonctions de non-occurrence d'événements. Les défauts graduels issus du process sont également considérés. Pour cela, les contraintes temporelles exprimant les dates d'occurrence des événements dans les Templates et les Chroniques sont modélisées par des distributions de probabilités (*DPs*). Celles-ci sont utilisées afin de caractériser un fonctionnement normal, dégradé ou défaillant de chaque sous-système avec un certain degré de certitude. Cette identification du fonctionnement est représentée par la valeur d'un indicateur de dégradation.

Mots clés: Systèmes à Evènements Discrets, diagnostic, apprentissage, modèle temporel, distribution de probabilité, indicateur de dégradation.

Abstract:

The work presents a contribution to the problem of diagnosis in discrete event systems (DES). A diagnosis approach by exploiting the temporal aspect which characterizing the occurrence of events is proposed. The system is modeled by temporal graphs belonging to the timed automata formwork. The approach is designed according to the decentralized architecture to avoid any combinatorial explosion in the construction of the models. It has allowed the detection and isolation of abrupt faults occurring on equipment by combining the enablement conditions of events and the Boolean functions for the non-occurrence of events. Gradual faults coming from the process its self are considerate. For this, time constraints expressing the dates of occurrence of events in the *Templates* and *Chronicles* are modeled by probability distributions (PDs). These are used to characterize normal, degraded or failed functioning of each subsystem with a degree of certainty. Identification of this functioning mode is represented by the value of a degradation indicator.

Keywords: Discrete Event Systems, diagnosis, learning, temporal model, probability distribution, degradation indicator.