REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

Département d'informatique

N° d'ordre : GLSD/M2/2020

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Génie logiciel et systèmes distribués**

Implémentation de la structure du diagnostiqueur à base de réseaux de pétri étiquetés

Par:

DJABOU TAKI EDDINE MOHAMED OUSSAMA

Soutenu le 11/11/2019, devant le jury composé de :

Nom et prénom	Grade	Président
Djaber Khaled	Grade	Rapporteur
Nom et prénom	Grade	Examinateur

Remerciement

En préambule à ce mémoire nous remerciant ALLAH qui nous aide et nous donne la patience et le courage durant ces langues années d'étude. Nous souhaitant adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.
Ces remerciements vont tout d'abord au corps professoral et administratif de la Faculté des Sciences, Juridiques et sociales, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenant à remercier sincèrement notre encadreur Monsieur, (DJABER KHALED), se sont toujours montrés à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'ils ont bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nous remercions vivement les membres de jury d'avoir accepté d'examiner ce mémoire
On n'oublie pas nos parents pour leur contribution, leur soutien et leur patience.
Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont --toujours encouragées au cours de la réalisation de ce mémoire.
Merci à tous et à toutes.

Dédicace

A peine nous venons de terminer la rédaction du mémoire de fin de cycle de Master, je voudrais très vite le dédier avec une immense joie, un grand honneur et un cœur chaleureux :

- *A ma très chère mère (Fadhila), affable, honorable, aimable : tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager et de prier pour moi.*
- *A mon cher père (Mohamed), aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eus pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.*
- *A ma sueur (Maroua) et mes frères (Aymen, Zaki et Ishak) Qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de générosité, qui me support dans ma vie, qui m'a appris, m'a supporté et ma dirigé.*
- *A tous mes amis précisément (Khaled) et mes chers collègues et à tous les membres de famille Djabou.*
- *A tous mes enseignants et encadreurs pour leur collaboration très précieuse à la réalisation de ce mémoire je veux dire merci.*

Table des matières

Introduction générale

Aujourd'hui les systèmes dynamiques et technologiques apparaissent de façon omniprésente dans l'industrie et dans notre vie quotidienne. Ils sont disponibles en diverses technologies dans les appareilles critiques d'où la fiabilité est la sûreté de fonctionnement des systèmes gérant ces dispositif devient une nécessité.

Pour cette raison la sûreté de fonctionnement concerne le diagnostic des défaillances pour valider le système. On utilise un processus spécifique pour examiner l'état actuel du système et déterminer l'ensemble des composants fautifs menant le système. A partir de l'ensemble des diagnostics trouvés la réparation du système sera faite et le processus de la validation sera répété jusqu'arrivé à un système valide.

Méthodologie récemment proposée pour le diagnostic des pannes des systèmes à événements discrets dans cette mémoire par la suite étendu à plusieurs ouvrages et a été utilisée avec succès dans une variété de domaines d'application, y compris le chauffage, la climatisation unités, et précisément dans l'intelligence artificielles.

La principale caractéristique de cette approche est l'utilisation d'un processus spécial à événements discrets. Dans le contexte des SED, le modèle du système est donné souvent sous la forme d'un Rdp étiqueté. Le projet concerne le diagnostic des SED modélisés par le Rdp étiqueté. Le principe consiste à générer de manière off ligne une structure dite le diagnostiqueur. L'objectif est de réaliser un outil logiciel (prototype) implémentant une telle technique qui résoudre les problèmes de pannes des systèmes à événement discret.

Le diagnostiqueur est construit à partir du modèle de système à évènement discret et est utilisé pour (i) effectuer une surveillance off ligne du système à des fins l'estimation l'état actuel du système et identifier les types des pannes diagnostic. Les états du diagnostiqueur contiennent des informations d'observation les traces visibles et les possibles types de fautes, selon le modèle du système.

Cette mémoire concerne les systèmes à évènement discrets modulaires (seule unité) modélisés par les réseaux de pétri comme un modèle du système. Ils ont été utilisés pour résoudre des problèmes de l'observabilité, l'explosion combinatoire, la surveillance du système, et le diagnostic des pannes dans plusieurs œuvres, y compris Systèmes possédant des structures modulaires.

Dans cette mémoire, nous définissons le formalisme de modélisation par événements discrets adopté dans cet article est celui des réseaux de pétri avec des transitions étiquetées, où certaines des transitions sont étiquetées par différents types d'événements de fautes non

observables. Dans l'approche de diagnostic, un diagnostiqueur d'un réseau de pétri étiqueté (cas particulier d'un Rdp), est construit à partir (i) du modèle réseau de pétri du système à événements discrets, (ii) de l'ensemble des événements observables (iii) de l'ensemble des événements non observables (iv) et de l'ensemble des fautes événements (types de fautes). Les états du diagnostiqueur.

Enfin, nous examinons l'état de diagnostic du système on utilise un algorithme de diagnostic pour objectif d'identifier les occurrences d'un événement non observable (les fautes) dans un système et son état actuel, à partir des chaînes d'événements observables (traces visibles).

Le contenu du mémoire se résume dans les points suivants :

- Dans le premier chapitre concept préliminaire qui est divisée en trois sections, le 1er définie un système à événement discret ensuite le diagnostic à base des automates enfin dans la 3^{éme} section on définit un réseau de pétri et ces propriétés.
- Le ^{2eme} chapitre, on donne une présentation détaillé de l'outil diagnostiqueur Rdp étiqueté suivie par une introduction de la technique de diagnostic à base de cet outil.
- Dans le 3eme décrire les différentes étapes de développement de notre outil de diagnostic à base des Rdps étiqueté.
- Dans le quatrième chapitre on donne la réalisation et l'implémentation de cet outil.

Le mémoire finira par une conclusion générale et des perspectives possibles.

Première partie

Etat de l'art

Chapitre 1

Concepts préliminaires

1.1. Introduction

Dans ce premier chapitre, nous allons d'abord parler des modèles systèmes à événement discret et du diagnostic à base de ce genre de modèles où nous présenterons les notions de base liées, ensuite nous allons rappeler quelques définitions relatives aux réseaux de Pétri.

1.2. Les systèmes à événements discrets :

Un système à événements discrets selon [Cassandra and S. Lafortune.1999] est un système dynamique pouvant être modélisé par des variables prenant des valeurs dans un domaine, discret et évoluant par l'occurrence, d'événements discrets et instantanés.

Formellement, un système G est défini par le quadruplet : $G = (X, \Sigma, \delta, x_0)$ tel que :
X est un ensemble d'états, Σ est un ensemble d'événements,$\delta \subseteq X \times \Sigma \times X$ est un ensemble fini de transitions et x_0 est l'état initial. Le langage généré par G est noté L(G) ou tout simplement L. L est un sous-ensemble de Σ* qui est la fermeture de Kleene de Σ. Il correspond à l'ensemble des traces (c.-à-d.) les mots) qui peuvent être exécutées dans G à partir de l'état initial. L(G) est donc préfixé i.e., $L(G) = pr(L(G)), \text{où } pr(L(G)) = \{u \,/\, \exists v \in \Sigma *, uv \in L(G)\}$ est l'ensemble des préfixes des traces dans L(G).

L'ensemble d'événements Σ est répartis en deux sous-ensembles Σ_0 et Σ_{u0}:
$\sum = \sum_0 \cup \sum_{u0}$. Σ_0 Contient les événements observables i.e., les événements dont l'occurrence peut être observée et qui peuvent être les commandes issues du contrôleur ou les valeurs obtenues des capteurs lors de l'exécution du système. Σ_{u0} Contient les événements non observables i.e., les événements dont l'occurrence ne peut pas être observée et qui contiennent les événements de fautes et tout autre événement qui peut causer un changement dans le système mais dont les capteurs ne détectent pas la présence. L'ensemble des fautes du système est noté Σ_f et représente un sous-ensemble des non observables: $\sum_f \subseteq \sum_{u0}$. On partitionne l'ensemble de fautes Σ_f en m ensembles disjoints qui correspondent aux différents types de fautes dont le traitement est assuré par les mêmes actions consécutives. Nous avons ainsi : $\sum_f = \sum_{f1} \cup \ldots \sum_{fm}$. $\prod_f \{1, \ldots m\}$ est l'ensemble des indices i correspondant aux types de fautes $\sum_{fi}$.

Exemple 1.1 : model d'un système à événement discret informatique fille d'attente.

E= {a, d, r1, r2, d1, d2}

Ou :

-a est une arrivée de l'extérieur dans le système).

-d est un départ de l'U.C. (état en exécution) vers l'extérieur (état terminé).

-r1, r2 sont des départs de l'U.C.vers les disques 1et2 respectivement.

-d1, d2 sont des départs de l'U.C.vers les disques 1et2 respectivement, qui retournent toujours vers la fille d'attente.

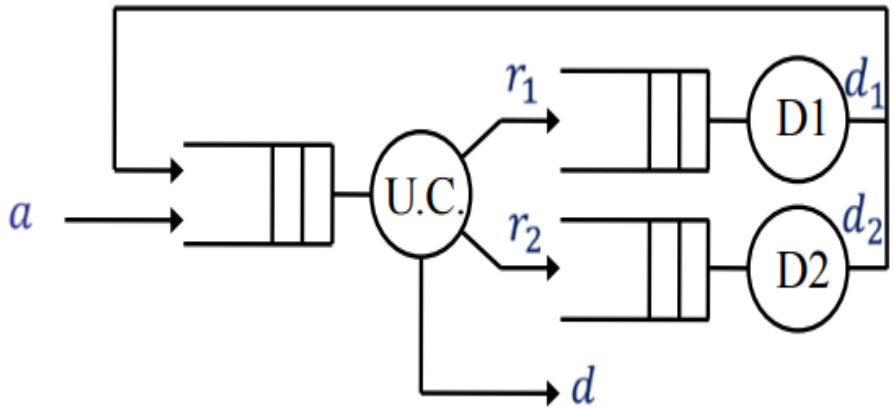


Figure 1.1.Model d'un système à événement discret informatique fille d'attente.

Définition 1.2 (Modèle) :

Le [TLFi] définit le modèle comme un système physique, mathématique, ou logique représentant les structures essentielles d'une réalité et capable à son niveau, d'en expliquer ou d'en reproduire dynamiquement, le fonctionnement Nous nous inspirons de cette définition pour donner notre propre définition dans notre contexte. Le modèle est une représentation mathématique, permettant de simuler le comportement d'un système.

1.3. Diagnostic des systèmes à événements discrets

Pour étudier le comportement des SEDs, on utilise l'outil des automates et les Rdp.

Définition 1.3.1. Les automates :

Sont un outil mathématique et graphique pour étudier le comportement des systèmes à évènement discret (décrit comme un langage) pour un avantage de simplicité.

Un automate est un ensemble d'états et de transitions menant d'un état à un autre état. Les transitions sont étiquetées par les événements menant d'un état à un autre état. Il existe de nombreuses définitions équivalentes d'automates. Nous avons choisi la définition $A = (Q, E, T, I, F)$ présentée en [Annexe] A. Dans cette définition, Q est l'ensemble des états

possibles du système et E est l'ensemble des événements qui peuvent avoir lieu sur le système. T Est l'ensemble des transitions. I L'ensemble des états initiaux. F L'ensemble des états finaux.

Définition 1.3.2. Automates non déterministe :

Dans les automates déterministes On a les suppositions suivantes :

1. Un seul état initial.
2. Les transitions sont définies via une fonction.
3. Tous les événements sont observables.

Il suffit que l'une de ces suppositions devienne non valide, c'est un automate non déterministe.

1.3.3. Observations :

Dans [PC05] Le système comporte un ensemble d'événements observables notéE_{obs}. L'occurrence d'un événement observable sur le système génère une observation c-à-dire l'information sur le comportement interne du système, résultant de l'occurrence d'un événement observable. Le comportement observé est alors défini comme l'ensemble partiellement ordonné des observations émises.

1.3.4. Observateur des SEDs observables :

Un SED $G = (Q, f, E, q_0)$ tel que :

Q : Ensemble fini d'états.

f : est la relation des transitions tel que : $f \subseteq (Q \times E \times Q)$.

E : Ensembles des évènements.

q_0 : Ensembles des états initiaux$(q_0 \subset Q)$.

Dans le cas général l'ensemble des évènements est partition en 2 sous-ensembles suivants :

$E = E_{obs} \cup E_{nonobs}$

E_{obs}= événements observable.

E_{nonobs}= $E_\tau \cup E_F$ « évènements non observable »

C'est-à-dire, on a une projection P : $P: E \to E_0$

$P_{ext}: E^* \to E_0^*$.

1.3.4.1. Construction d'un observateur

D'âpres [BEN19] l'observateur construit comme suite:

Entrée :$G = (X, E, f, x_0, X_m / E = E_0 \cup E_{N0})$

Sorties :$OBS(G) = (X_{obs}, E_0, f_{obs}, x_{0,obs}, X_{m,obs})$

1. Etape : $x_{0,obs} = UR(x_1) = Unobservel\ Reach.$

$$x_{0,obs} \leftarrow \{x_0, obs\}$$

2. Etape2 : pour chaque $B \in X_{obs}$, pour chaque $e \in E_0$.

$$f_{obs}(B, e) = \varepsilon\_R(y \in X / \exists x_e \in B : y \in f(x_e, e))$$

Ajouter $f_{obs}(B, e) = B'$ à X_{obs}

3. Etape3 : répéter l'etape2 jusqu'à ce que la partie entièrement accessible soit construite.
4. Etape4 : $X_{m,obs} \leftarrow \{B \epsilon X_{obs} / B \cap X_m \neq 0\}$

Exemple1

E= {a, b} ; X= {0, 1, 2,3} $X_0 = 0$; $X_m = \{0\}$

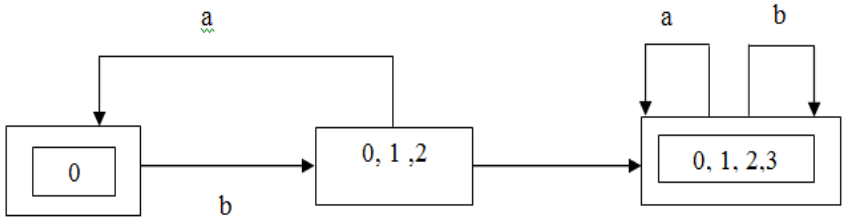


Figure 1.2.Automate observateur.

Dans l'ensemble des événements non observables des événements de fautes.

1.3.5 Diagnostique des événements de fautes :

Le diagnostic est un domaine de l'intelligence artificielle consistant à détecter, localiser et si possible identifier précisément tout dysfonctionnement au sein d'un système.

Définition 1.3.5.1 (Panne).

Une panne définit en [TLFi] par est un ensemble, d'états du système correspondant à un dysfonctionnement. Dans le cadre du diagnostic, on peut également être intéressé par la cause de la panne, c'est-à-dire l'événement qui a causé la panne (évènement de faute).

Définition 1.3.5.2 (Événement de faute).

Un événement de faute est l'événement conduisant le système, dans un état de panne. Un événement de faute est aussi appelé une panne c'est un évènement non observable dans le système et il a plusieurs types de fautes.

Parmi les événements non observable E_u, il y quelques événement qui sont considérés comme des fautes :

$$E_u = E_N \cup E_F$$

E_N : Événement normal.

E_F : Événement de faute.

Pour déterminer en plus de l'état actuel du système après la réception d'une séquence d'événement observables si le système à exécuté des événements de fautes, on doit modifier l'automate observateur pour obtenir une autre automate dit le diagnostiqueur [BEN19].

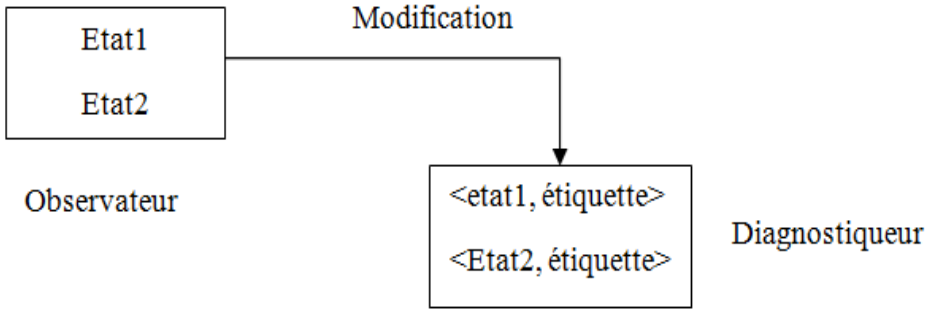


Figure 1.3.Diagnostiqueur à base observateur.

La modification par un couple (état /étiquette) tel que : étiquette peut être normal ou faute.

L'état initial toujours normal.

Propagation franchir l'étiquette après chaque transition.

1.3.6 Automate diagnostiqueur :

Diag (G) est similaire à Obs. (G) avec une légère modification où des étiquettes sont attachées aux états de G dans les états de Diag(G).

Les modifications à apporter à la procédure de construction de l'Obs (G) pour obtenir Diag (G) sont :

Lors de la construction de $UR(x_0)$de G, attacher l'étiquette N aux êtas qui peuvent être atteints depuis x_0 par séquences non observable dans E_U/E_F[BEN19].

Exemple 1.1 :

Le modèle initial *G* est défini par :

- L'ensemble des états est$X = \{x_0, x_1, x_2, x_3, x_4\}$.
- L'ensemble des événements est :$\Sigma = \{a, b, c, u, f_1\}$, les sous ensemble des événements observables, non observables et de fautes sont respectivement :

$$\sum_0 = \{a, b, c\};\ \sum_{u0} = \{u, f_1\};\ \sum_f = \{f_1\}$$

- L'ensemble de transition est :

$$\delta = \{(x_0, a, x_0)(x_0, f_1, x_0)(x_0, u, x_0)(x_0, b, x_0)(x_0, c, x_0)(x_0, b, x_0)(x_0, b, x_0)\}$$

- L'état initial est : x_0

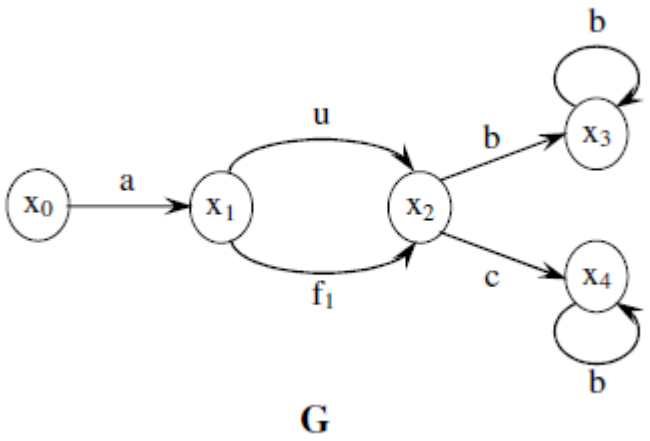


Figure 1.4.Exemple d'un système à évènement discret.

La figure 1.5 montre le diagnostiqueur correspondant au système *G* présenté dans l'exemple

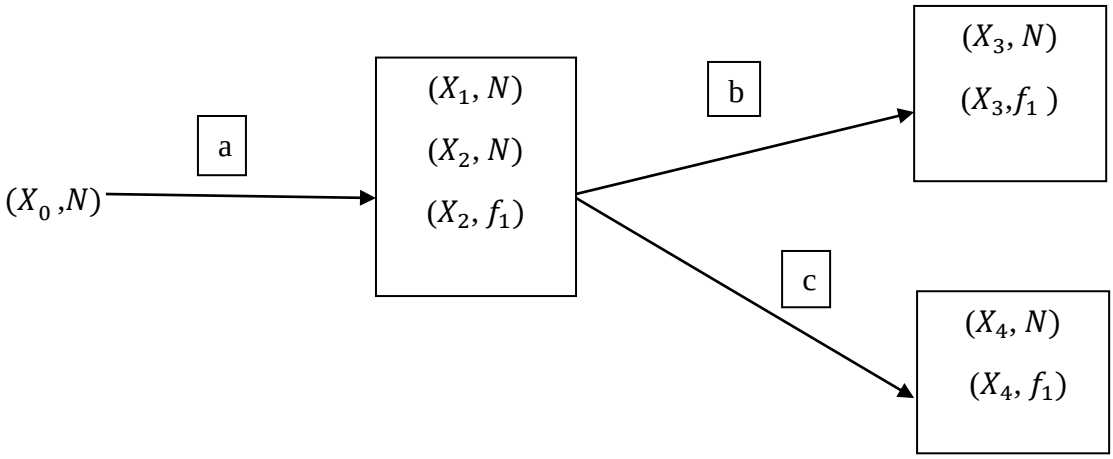


Figure 1.5.Diagnostiqueur du système présenté dans l'exemple 1

Généralement les événements de fautes sont des éventements non observables car le diagnostic des évènements des fautes observables est trivial.

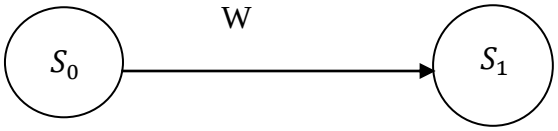
On a plusieurs événements de fautes possibles comme suite.

$$E_F = E_{F1} \cup E_{F2} \cup \ldots \cup E_{Fn}$$

E_{fi} Est l'ensemble des événements de fautes de type fi.

Une faute (erreur ou panne) est le franchissement de certaine transition de faute.

Lors la construction du diagnostiqueur, on utilise à la place de l'étiquette « y » l'étiquette « fi » tel que **:** Y= faute dans le cas générale ; Fi= faute dans le cas spécifique, est une type de faute ; N= évènement normale [BEN19].



$< S_n, N > si \nexists\, e \in E_F / e \in w.$
$< S_n, F_i > si \exists\, e \in E_{Fi} / e \in w.$
$< S_n, F_i, F_j > si \exists\, e_1 \in E_{Fi}; e_2 \in E_{Fj} / e_1, e_2 \in w.$

La Diagnostiquabilité signifie qu'une fois événement de faute a été exécuté par le système. Le diagnostiqueur doit être en mesure de confirmer après le fini qu'une faute du même type a été exécutée.

Automate diagnostiqueur Diag(g) est utilisé :

- Le diagnostic on-line d'un système en cours d'exécution.
- La vérification de la diagnosticabilité par exemple les systèmes de climatisation

1.3.7 Les problèmes des automates :

Un problème associé aux automates est la représentation des comportements concurrents. Deux ensembles d'événements l_1 et l_2 sont concurrents lorsqu'ils interviennent sur deux parties disjointes du système. Dès lors, lorsque les deux comportements sont possibles, ils peuvent avoir lieu dans n'importe quel ordre. La figure 1.5 montre ce qui se passe avec les deux ensembles $l_1$et l_2.

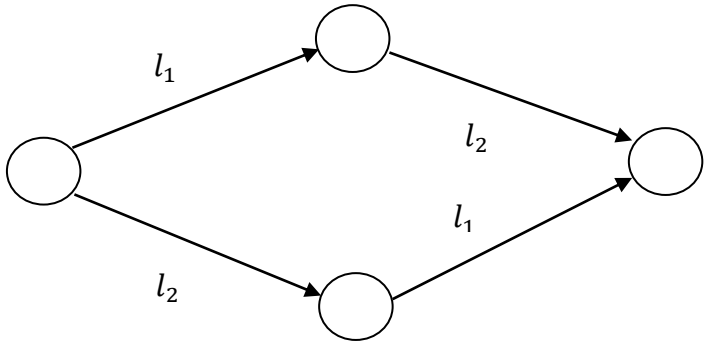


Figure 1.6.Représente problème de l'automate.

- ✓ Explosion combinatoire (explosion l'espace d'états qui est le résultat des nombreux comportements de composants combinés dans un modèle composé).
- ✓ N'excepte pas le parallélisme.

1.4. Les Réseaux de Pétri (RdPs) :

Les réseaux de pétri ont été utilisés pour représenter le modèle structurelle et comportementale du système sont exploitées comme mécanisme de raisonnement pour éviter le problème d'explosion combinatoire de l'espèce d'états des systèmes.

1.4.1. Définition formelle

Un réseau de Pétri (Rdp), appelé aussi réseau de Pétri ordinaire ou encore réseau de Pétri place/transition est introduit par [Pet62].Il est d´défini par un quadruplet (P, T, Pré, Post) où :

- P est un ensemble fini de places.
- T est un ensemble fini de transitions $T \cap P = \emptyset$.
- Pré est l'application (matrice) d'incidence avant : $Pre \subseteq (P \times T)$.
- Post est l'application (matrice) d'incidence arrière : $Post \subseteq (T \times P)$.

1.4.2. Définition informelle « graphique »

Un Réseau de pétri est un graphe bipartie orienté et pondéré, il se compose de deux types de nœuds :

Les places et les transitions, qui sont reliées par des arcs.

1. **les places** : notées graphiquement par des cercles. Chaque place contient un nombre entier (positif ou nul) de marques (ou jetons). Ces derniers sont représentés par des points noirs.
2. **les transitions** : notées graphiquement par un rectangle ou une barre. Une transition qui n'a pas de place en entrée est appelée transition source et une transition qui n'a pas de place en sortie est appelée transition puits.

Les places et les transitions sont reliées par des arcs orientés où :

3. **Un arc relie** : soit une place à une transition, soit une transition à une place mais jamais une place à une place ou une transition à une transition. Chaque arc est étiqueté par une valeur (ou un poids), qui est un nombre entier positif. L'arc ayant k poids peut être interprété comme un ensemble de k arcs parallèles. Un arc qui n'a pas d'étiquette est un arc dont le poids est égal à 1

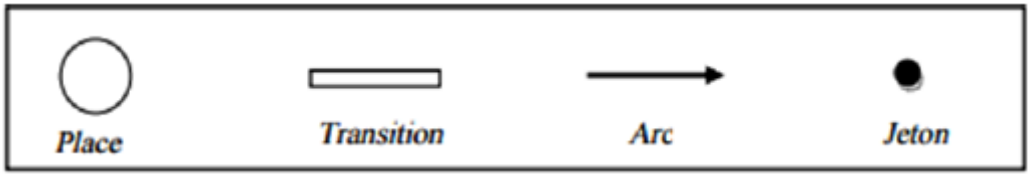


Figure1.7Représentation graphique des éléments de Rdp.

Figure 1.8.Format d'arc.

1.4.3. Matrice d'incidence :

La matrice d'incidence W qui va être décrite, fait la synthèse de tous les liens entre places et transitions du Rdp. Cette matrice est en général rectangulaire et possède un nombre de colonnes égal au nombre de transitions du réseau, ainsi qu'un nombre de lignes égal au nombre de places du réseau. Chaque élément de la matrice témoigne de la présence ou de l'absence de lien entre chaque place et chaque transition, ainsi que du poids attaché à l'arc en question. La direction de cet arc est transcrite par le signe de l'élément en question.

$$W_{ij} = W_{ij}^{+} - W_{ij}^{-}$$

Matrice d'incidence avant (pré) :

$$W^{-} = \left[W_{ij}^{-}\right] \text{ où } W_{ij}^{-} = pré(P_i, T_i)$$

Matrice d'incidence arrière (post) :

$$W^{+} = \left[W_{ij}^{+}\right] \text{ où } W_{ij}^{+} = post(P_i, T_i)$$

Où :

– Pré est une application de $P \times T \rightarrow \mathbb{N}$ dite d'incidence avant.

– Post est une application de $P \times T \rightarrow \mathbb{N}$ dite d'incidence arrière.

– Pré (Pi, Tj) est appelé poids de l'arc reliant Pi et Tj.

– Post (Pi, Tj) est appelé poids de l'arc reliant Tj et Pi.

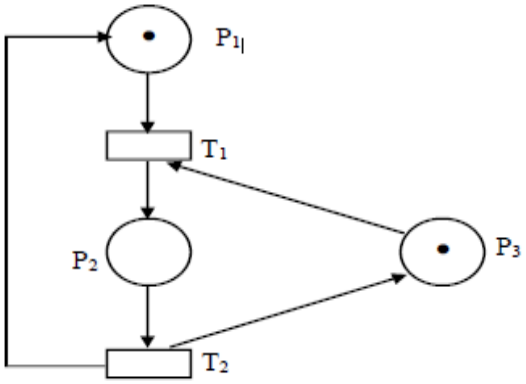


Figure1.9.Exemple d'un réseau de pétri.

$$W_a^- = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$W_a^+ = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

La matrice d'incidence de ce réseau est :

$$w_a = W_a^+ - W_a^- = \begin{bmatrix} -1 & +1 \\ +1 & -1 \\ -1 & +1 \end{bmatrix}$$

1.4.3. Dynamique :

Un réseau de pétri évolue lorsqu'on exécute une transition : des jetons sont retirés dans les places en entrée de cette transition et des jetons sont déposes dans les places en sortie de cette transition.

1.4.4. Marquage :

Chaque place contient un nombre entier (positif ou nul) de marques ou jetons. Le nombre de marque contenue dans une place P_i sera note soit $M(P_I)$ soit mi. Le marquage du réseau a l'instant$i$, M_i est defini par le vecteur de ces marquages mi c-à-dire $M_i = (m_i, m_2, \ldots, m_n)$. Le marquage dit initial décrit l'état initial du système modélise (M_0).Donc:

— Les places sont marquées par des jetons ou marques (points noirs), un nombre entiers positifs ou nul.

— Les jetons circulent dans les places selon certaines règles. Cette circulation symbolise l'évolution dynamique du système. Le marquage initial donne la position initiale des jetons [site1].

Exemple : Le marquage initial est M_0 = (1, 1, 0,0)

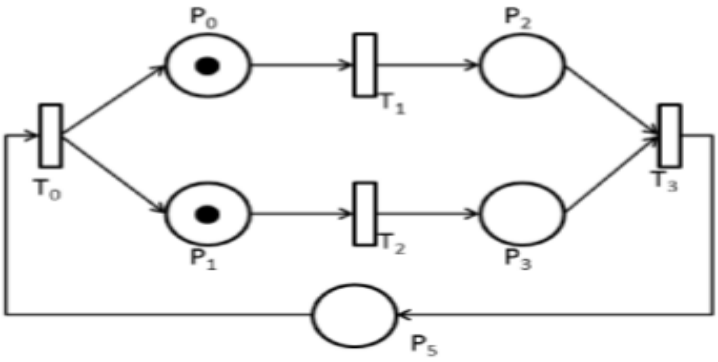


Figure1.10.Le marquage initial d'un Rdp.

Remarques Le marquage a une instante donne définit l'état du Rdp, ou plus précisément l'état du système décrit par le Rdp. L'évolution de cet état correspond donc a l'évolution du marquage (par franchissement de transitions). Par abus de langage, nous appellerons les Rdp marques : Rdp et les non marques : Rdp non marques. A tout marquage accessible à partir du marquage initial par franchissement d'une séquence de transitions correspond un état du système [site1].

Le marquage des places d'un Rdp représente l'état du système modélise a un instant donne. Ce marquage peut être modifie au cours du temps par le franchissement de transitions. Pour pouvoir être franchie, une transition doit être validée. Informellement, une transition est valide lorsque le nombre de jetons dans chacune des places d'entrée est supérieur ou égal au poids de l'arc qui la relie a la transition [site1].

1.4.5 Transition sensibilisée :

Une transition t est dite sensibilisée si chaque place d'entrée de la transition t contient au moins un nombre de jetons égal au poids de l'arc de liaison dirigée de pi à t.

Définition Formellement une transition t est sensibilisée pour un marquage M si :

$$\forall p_i \in {:}\bullet \text{ t: } M(p_i) > pré(p_i, t)$$

$\bullet t$: est l'ensemble des places d'entrée de la transition t. On note M le fait que t soit sensibilisée pour M

Lorsque cette condition est satisfaite, le franchissement consiste à enlever de chacune des places d'entrée, un nombre de jetons égal au poids de l'arc qui relie la place à la transition et a déposer, dans les places de sortie, un nombre de jetons égal au poids de l'arc qui relie la transition aux places de sortie [sitre1].

1.4.6 Franchissement d'une transition

Un Rdp marque $t_j \in T$ une transition franchissable à partir de $M_k(p_i)$. Le franchissement d'une transition sensibilisee t nous conduit d'un marquage courant M aà un nouveau marquage M' c'est à dire la création d'un nouveau marquage ou bien, en mettant à jour le marquage des places d'entrée/sortie de *t* définis dans M. Le franchissement d'une transition ou le tir d'une transition, consiste àa enlever un jeton dans chacune des places d'entrée de la transition et à ajouter un jeton dans chacune des places de sortie de la même transition.la règles de franchissement et de circulation des jetons sont :

- Le franchissement d'une transition ne peut s'effectuer que si chacune des places en amont de cette transition contient au moins une marque(Jeton).

- Une transition franchissable n'est pas forcément une transition franchie (par exemple, deux transitions disposant en amont d'une seule place munie d'un seul jeton).
- Il n'y a qu'un seul franchissement à la fois.
- Un franchissement est de durée nulle.

Définition Le franchissement d'une transition t d'un marquage M à M' est formalise par :

$$(\forall p)\ M'(p) = M(p) - pré(p,t) + post(p,t)$$

On note $M \xrightarrow{t} M'$

On dit que M' est accessible depuis M.

L'exemple suivant explique le franchissement d'une transition et les cas possibles représentent graphiquement le franchissement d'une transition dans un Rdp.

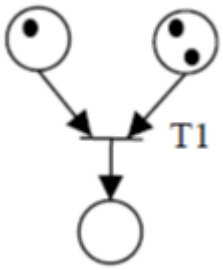


Figure 1.11.Franchissement d'une transition valide.

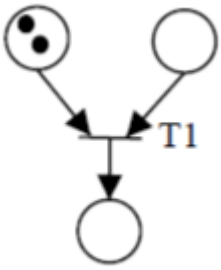


Figure 1.12.Franchissement d'une transition non valide

1.5. Le principe de choix réseau de pétri étiqueté :

Le choix de réseau de pétri étiquète pour diagnostic un système parce que :

Dans le diagnostic des systèmes événements discrets, les fautes sont modélisées via des événements non observables, si on cherche à utiliser les réseaux de pétri comme model, le model doit être un réseau de pétri étiqueté sur les transitions par les événements et l'observation est une séquence d'événement observables.

Afin d'appliquer ceci dans un contexte distribué, on doit définir des réseaux de pétri étiquetés et modulaires soit via des places de frontière sois via des transitions communes.

1.6. Conclusion

Après cette brève présentation du diagnostic des systèmes à événement discret et du formalisme de réseaux de pétri étiqueté, nous allons essayer dans le prochain chapitre de donner une présentation détaillée de technique plus efficace pour estimer l'état du système qui s'appelle le diagnostic off-ligne pour un diagnostiqueur de réseau de pétri étiquète.

Chapitre 2

Le réseau de pétri étiqueté outil pour Le diagnostic des SED

2.1. Introduction

Dans ce chapitre, nous décrivons la notion de diagnostiqueur de réseaux de pétri étiqueté, qui est utilisé comme un outil pour diagnostic dans les systèmes à évènements discrets.

Le système à diagnostiquer est modélisé par un réseau de pétri étiqueté. Les événements à diagnostiquer, ci-après appelés "fautes", sont modélisés comme des événements non observables dans le système. L'objectif est de diagnostiquer l'occurrence des événements de fautes (pannes) sur la base de la séquence des événements observés et sur la structure des modules de réseau de pétri. On cherche à obtenir un diagnostiqueur de réseaux de pétri étiqueté.

2.2. Un graphe de réseau de Pétri :

Est défini comme $\mathcal{N} = (P, T, A, w)$, où P et T sont des ensembles finis de lieux et de transitions, respectivement, A est l'ensemble des arcs des lieux aux transitions set des transitions aux lieux, et $w: A \to \mathbb{Z}^+$ est la fonction de poids sur les arcs.

On note par $W(P,t)$, la vectrice ligne de taille égale au nombre de places dans P et dont la i IIème colonnes est égale à $w(t,p_i) - w(p_i,t)$ où $p_i \in Pet\ t \in T$.

2.3. Un réseau de Pétri étiqueté :

Est défini comme $(\mathcal{N}, \Sigma, l, x_0)$, où Σ est l'ensemble des événements, $l : T \to \Sigma$ est la fonction d'étiquetage de transition, et x_0 est l'état initial. Une transition $t \in T$ peut tirer de $x \in X$, où X est l'espace d'état du réseau de pétri étiqueté, si et seulement si t est faisable (activé) à partir de x.Une transition t est activée à partir de x ssi $x + W(t) \geq 0$. Lorsque t tire de l'état x , la fonction de transition d'état $f: X \times T \to X$ donne l'état résultant selon le réseau de Pétri habituel dynamique, i. e., $f(x;\ t) = x + W(t)$.

Certains des événements de Σ sont observables, à savoir leur occurrence peut être observée (détectée par des capteurs), tandis que les autres événements sont non observables ; donc $\Sigma = \Sigma_0 \cup \Sigma_{\mu 0}$.

L'ensemble des événements de faute Σ_f est un sous-ensemble de $\Sigma_{\mu 0}$. Nous partageons l'ensemble de fautes en ensembles disjoints où chaque ensemble correspond à un type de faute différent. C'est parce qu'il ne pourrait pas être nécessaire pour détecter et isoler de manière unique chaque événement de faute, mais seulement l'occurrence d'un parmi un sous-ensemble (type) des événements de faute. On note ΣF_K l'ensemble des événements de faute correspondant à une faute de type k.Figure2.1Exemple d'un réseau de pétri étiqueté.

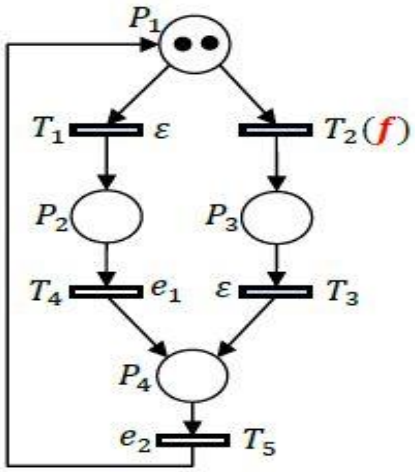


Figure 13.Exemple d'un réseau de pétri.

2.4. Diagnostiqueur réseau de pétri étiqueté :

Nous présentons maintenant le diagnostiqueur selon [Genc and S.Lafortune2003] Le diagnostiqueur est un réseau de pétri étiqueté construit à partir du modèle du système $(N, \sum, \mathrm{l}, \mathrm{x}_0)$.Ce réseau de Pétri étiqueté effectue des diagnostics tout en observant en ligne le comportement de$(N, \sum, \mathrm{l}, \mathrm{x}_0)$.

Le diagnostiqueur pour $(N, \sum, \mathrm{l}, \mathrm{x}_0)$ est $N_d = (N, \sum, \mathrm{l}, \mathrm{x}_{\mathrm{d0}}, \Delta_{\mathrm{f}})$.

Voici le schéma fonctionnel du système et de son diagnostiqueur :

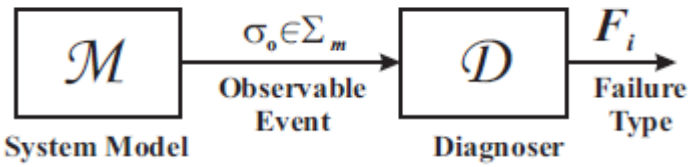


Figure 14.Diagnostiqueur du système.

Où $N, \sum, l$ sont définis comme précédemment, x_{d0} est l'état initial du diagnostiqueur et $\Delta_f = \{F_1, F_2, \dots, F_K\}$ est l'ensemble fini des types de fautes. Le réseau de pétri de diagnostic N_d conserve la structure graphique du modèle système sous-jacent. Jusqu'à présent, N_d n'est pas différent d'un réseau de pétri étiqueté. Cependant, sa dynamique est différente de celle d'un réseau de pétri étiqueté puisque sa fonction de transition d'état n'est définie que pour les événements observables. Le diagnostiqueur donne l'estimation de l'état actuel du système après l'occurrence d'un événement observable. Par la suite, lorsque nous disons «état», nous entendons l'état du modèle du système et lorsque nous disons «état du diagnostiqueur», nous entendons l'état du diagnostiqueur. L'état du diagnostiqueur est une liste de l'ensemble des états dans lesquels le modèle de système peut se trouver après l'observation d'un événement dans $\sum_0$ avec les informations de faute. Les informations de fautes dans un état de diagnostic sont codées par des étiquettes de fautes. Chaque état dans un état de diagnostic a une étiquette de faute. Une étiquette de faute est un vecteur de longueur k (le nombre de types de fautes) qui a des entrées de "0" ou "1".Si nous notons l'étiquette de faute par l_f puis$l_f \in \Delta = \{0,1\}^k$. Ainsi, le nombre d'étiquettes de fautes possibles est $|\Delta| = 2^k$. Lorsque l'étiquette de faute est le vecteur 0, nous disons que l'étiquette de faute est "normale". L'état initial a par définition l'étiquette de faute «normal». Nous définissons maintenant la fonction de propagation d'étiquette de faute $LP: X \times \Delta \times T^* \to \Delta$. LP propage les étiquettes de fautes cohérentes avec les traces d'événements. Soit $x \in X, l_f \in \Delta, s \in T^*$. Alors $LP(x, l_f, s)$ est défini comme :

$$LP(x, l_f, s) = l_f + \sum_{i=1}^{k} b_i^8$$

Où $b_i^8 \in \Delta$ et

$$b_i^a = \begin{cases} [0, \ \cdots, 0, \underset{\uparrow i^{th} \ coloumn}{1}, 0, \ \cdots, \ 0], \text{ if } l(s) \text{ contains an event from } \Sigma_{Fi}, \\ [0 \ , \cdots, 0, 0, 0, \ \cdots, \ 0] \text{ , otherwise.} \end{cases}$$

Avant de définir l'état de diagnostic et la fonction de transition d'état de diagnostic, nous avons besoin de la notion de portée non observable « unobservable reach » d'un état de diagnostic. Pour définir la portée non observable d'un état de diagnostic, nous définissons d'abord la portée non observable d'un état.

Soit $x_{d,i} = x_i l_i$ un état avec l'étiquette de faute l_i dans l'état de diagnostic x_d. La portée non observable de $x_{d,i}$ est désignée par $UR(x_{d,i})$ et définie comme suit :

$$UR(x_{d,i}) := \{x_i l_i\} \cup \{y l_y : \exists y \in R(\mathcal{N}, x_i),\ \exists s \in T^*,\ l(s) \in \Sigma_{uo}^* \\ (f(x_i, s) = y) \text{ and } (l_y = LP(x_i, l_i, s))\}.$$

Nous pouvons maintenant définir l'état initial du diagnostiqueur x_{d0} de N_d. x_{d0} Est la liste de tous les éléments distincts de $UR(x_0 l_0)$, où x_0 est l'état initial du réseau de pétri étiqueté sous-jacent $(N, \sum, \mathrm{l}, \mathrm{x}_0)$.et l_0 est l'étiquette faute de x_0 qui est "normal" par définition. Nous trouvons les états du diagnostiqueur accessibles à partir de l'état initial du diagnostiqueur en utilisant la fonction de transition d'état du diagnostiqueur. Afin de définir la fonction de transition d'état du diagnostiqueur, nous définissons d'abord les transitions réalisables puis les états atteints en déclenchant les transitions réalisables.

Nous notons $B(x_{d,i}, a)$ les transitions possibles de $x_{d,i}$= $x_i l_i$ où $x_i \in X$est un état dans l'état de diagnostic x_d, l_i est l'étiquette de faute dex_i, et a est un événement dans Σ_0. Formellement, $B(x_{d,i}, a)$ est défini comme :

$$B(x_{d,i}, a) = \{t \in T :\ l(t) = a \text{ and for all } p \in I(t)\ (x_i(p) \geq w(p,t))\}.$$

On définit $B(x_d, a)$ comme l'ensemble résultant de l'union de $B(x_{d,i}, a)$ pour tout i, où $1 \leq i \leq r$, c'est-à-dire,

$$B(x_d, a) = U_i \leq i \leq r B(x_{d,i}, a).$$

Où r est le nombre de lignes de x_d.Nous désignons par $S(x_d, a)$ l'ensemble de tous les états atteignables distincts, ainsi que leurs étiquettes de fautes, lorsque toutes les transitions de $B(x_{d,i}, a)$ sont déclenchées à partir de chaque $x_{d,i}$ de x_d. À savoir, $S(x_d, a)$ est défini comme :

$$S(x_d, a) := \cup_{1 \leq i \leq r} \cup_{t \in B(x_{d,i}, a)} \{x_i' l_i' :\ x_i' = f(x_i, t),\ l_i' = l_i\}$$

Où x_i est un état dans x_d, l_i est l'étiquette de faute de x_i et r est le nombre d'états dansx_d. Puisque les événements de fautes ne sont pas observables, la fonction de propagation d'étiquette ne change pas les étiquettes des fautes des états dans le processus de construction de $S(x_d, a)$.

L'état diagnostiqueur x_d du module N_d.est une matrice de la forme :

$$\begin{pmatrix} - & | & - \\ x_s(i) & | & x_f(i) \\ - & | & - \end{pmatrix}$$

Où $x_s(i)$désigne l'état dans la rangée i de l'état du diagnostiqueur x_d, $x_f(i)$ désigne l'étiquette de faute. La partie d'état $x_s(i)$de chaque ligne i correspond à un possible état de M suivant l'occurrence de la séquence d'événements observée.

La fonction de transition d'état du diagnostiqueur de N_d est de la forme $f_d: X_d \times \Sigma_0 \rightarrow \mathrm{X_d}$ où X_d est l'espace d'états du diagnostiqueur N_d. Étant donné l'état du diagnostiqueur $x_d \in X_d$ et l'événement observable $a \in \Sigma_0$, alors $f_d(x_d, a)$ est défini que Si $B(x_{d,i}, a) \neq 0$ pour certains $x_{d,i}$ dans x_d, Si $f_d(x_d, a)$ est définit, alors $x'_d = f_d(x_d, a)$ et x'_d est la liste des éléments de l'ensemble

$$U_s \in S(x_d, a)UR(s).$$

Les informations de diagnostic fournies par l'état du diagnostiqueur sont obtenues en examinant les k dernières colonnes de cet état: (i) si une colonne ne contient que des 0's, alors nous savons qu'aucun événement de faute du type correspondant n'aurait pu se produire; (ii)s'il s'agit d'une colonne contient seulement 1's, alors nous sommes certains qu'au moins un événement de ce type a eu lieu; (iii) sinon, si une colonne contient des 0's et des 1's , nous ne sommes pas certains l'occurrence d'une faute de ce type. Si le diagnostiqueur est certain qu'une faute de si le type i s'est produit, alors il génère les sorties "Fn" comme indiqué dans la figure 2.1. Ce diagnostic les informations sont équivalentes à celles obtenues à partir des automates de diagnostic du programme de diagnostic Approche de [54]

2.5 Etude de cas (exemple) :

Considérons le graphe du réseau de pétri *N* donné sur Fig. 2. L'ensemble des places de *N* est $P = \{p_1, p2,..., p10\}$. L'ensemble des transitions de *N* est $T = \{t_1, t2,..., t7\}$. Tous les poids de l'arc sont égaux à 1. Le marquage initial x_0 est :

$$x_0 = (1110000000000)$$

L'ensemble des événements est $S = \{a, e, \sigma_{u0}, f_1, f_2\}$. Nous n'écrivons pas explicitement la fonction d'étiquetage d'événement l, mais l'étiquette d'événement de chaque transition $t \in T$ est représentée sur la figure 2.3 L'ensemble des événements non observables est :

$\sum_{u0} = \{\sigma_{u0}, f_1, f_2\}$ et tous les événements restants dans l'ensemble d'événements $\sum$ sont observables. Il existe deux types de défauts et les ensembles correspondants sont $\sum_{f1} = \{f_1\}$ and $\sum_{f2} = \{f_2\}$.Soit

$N_d = (N, \sum, l, x_0, \Delta_f)$ Le diagnostiqueur. L'état initial du diagnostiqueur x_0 est la liste des éléments de l'ensemble $UR(x_0 l_0)$ où $l_0$est "normal". Ensuite, x_0 est trouvé comme

Ces quatre états dans l'état initial du diagnostiqueur sont représentés dans N_d sur la figure 2.4

Nous montrons les états du diagnostiqueur qui sont atteints si la séquence des événements observables est "ae". Ces états ne sont pas répertoriés ici en raison de contraintes d'espace. Nous notons que nous avons construire une matrice de diagnostic pour générer l'espace d'états des diagnostiqueurs de réseaux de pétri.

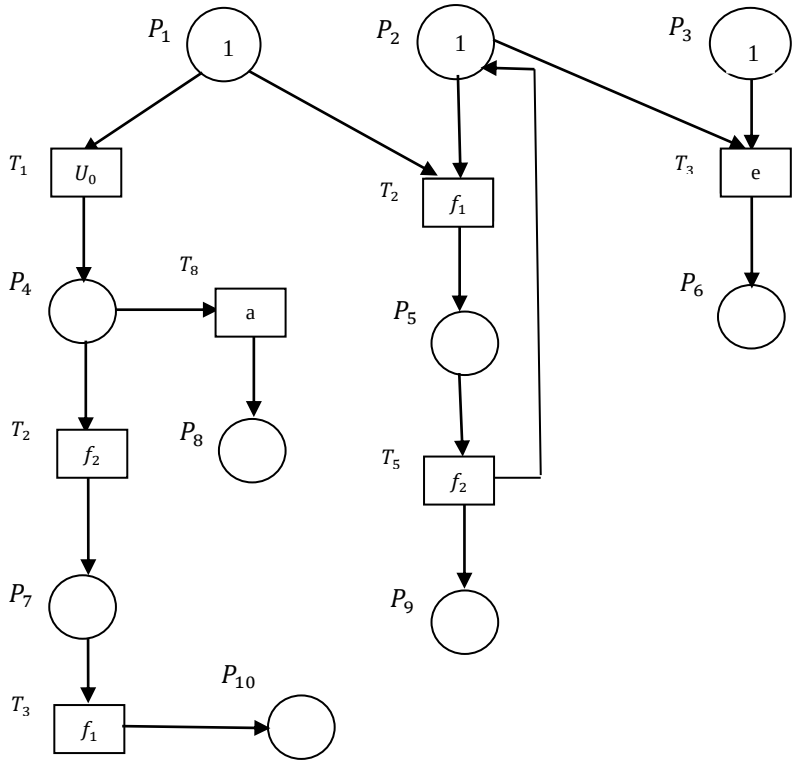


Figure 15.Le système a diagnostiqué.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	f_1	f_2
$x_{d,0}$	1	1	1	0	0	0	0	0	0	0	0	0
$x_{d,1}(u_0)$	0	1	1	1	0	0	0	0	0	0	0	0
$x_{d,2}(f_2)$	0	1	1	0	0	0	1	0	0	0	0	1
$x_{d,3}(f_1)$	0	1	1	0	0	0	0	0	0	1	1	1
$x_{d,4}(f_1)$	0	0	1	0	1	0	0	0	0	0	1	0
$x_{d,5}(f_2)$	0	0	1	0	0	0	0	0	1	0	1	1
$x_{d,6}(f_2)$	0	1	1	0	0	0	0	0	0	0	1	1
$x_{d,7}$(a)	0	0	0	0	0	1	0	0	0	0	0	0

Figure 16.Représente l'état de diagnostic.

2.6. Conclusion

Dans ce chapitre on a défini une classe particulière des RdPs nommé réseau de pétri étiqueté pour représenter et diagnostiquer les systèmes à évènement discret. Cette classe associé à chaque transition des déférentes évènements décrivant le comportement du système à diagnostiqué et les informations possibles pour connaissais l'Etat du système et les types des pannes s'il a trouvé. Le diagnostic à base de cet outil consiste à construire le diagnostiqueur Rdp étiqueté.

La conception et l'implémentation de cette technique font l'objet du prochain chapitre.

Chapitre3

Développement d'un outil de diagnostic à base de Rdp étiqueté

3.1Introduction

On a présenté dans le chapitre précédent une technique de diagnostic basée sur les Rdps étiquetés Notre travail consiste à développer un outil de diagnostic à base de cette technique.

Afin d'arriver à la fiabilité et la sûreté de fonctionnement de notre outil, on va suivre le cycle de vie classique de logiciels où la partie technique de développement de ceci peut être vue comme l'établissement d'une suite de descriptions de plus en plus précises et de plus en plus proches d'un programme exécutable. On commence par une définition des besoins, suivie par une conception architecturale bien décrite dans une conception détaillée et finalement l'implémentation.

3.2. Définition de besoins

C'est l'activité essentielle au début du processus de développement du logiciel. Elle a pour but d'éviter de développer un logiciel non adéquat et avez des fautes.

Comme nous avons mentionné auparavant, notre objectif est la réalisation d'un outil de diagnostic basé sur les Rdps étiqueté assurant :

- Une interface graphique simple en termes d'utilisation permettant la saisie du modèle sous forme d'un diagnostiqueur Rdps étiqueté.
- Offrir de manière simple la saisie de l'ensemble des événements pour que la tâche du diagnostic s'effectué et la fonction transitons étiquetés.
- La tâche de diagnostic doit être passée par :

➢ contient des informations de diagnostic, état initial du système puis l'état prochaines avec les listes de pannes possible c-à-dire une étiquette de faute, qui fournissent informations sur les types de pannes éventuellement survenues.

- une solution à un problème de diagnostic est l'estimation de l'état actuel du système avec trouver les type de faute.

3.3 Conception

3.3.1Conception générale

Par cette étape, le logiciel est décomposé en composants plus simples où l'interface et les fonctions de chacun sont précisées. A partir de l'architecture générale présentée dans la Figure 3.1, l'outil à réaliser prend en entrée le modèle sous forme d'un Rdp étiqueté représentant le comportement du système à diagnostiquer et l'ensemble des observations sous la forme d'un ensemble des événements partiellement ordonnées. En se basant sur ces entrées, Le Rdp et l'ensemble des évènements pour construire un modèle de système a diagnostiqué puis l'observation qui fournit des traces visible vérifier par le diagnostiqueur enfin il a trouvé ensemble de diagnostics expliquant le mal fonctionnement du système.

L'outil est composé ces modules :

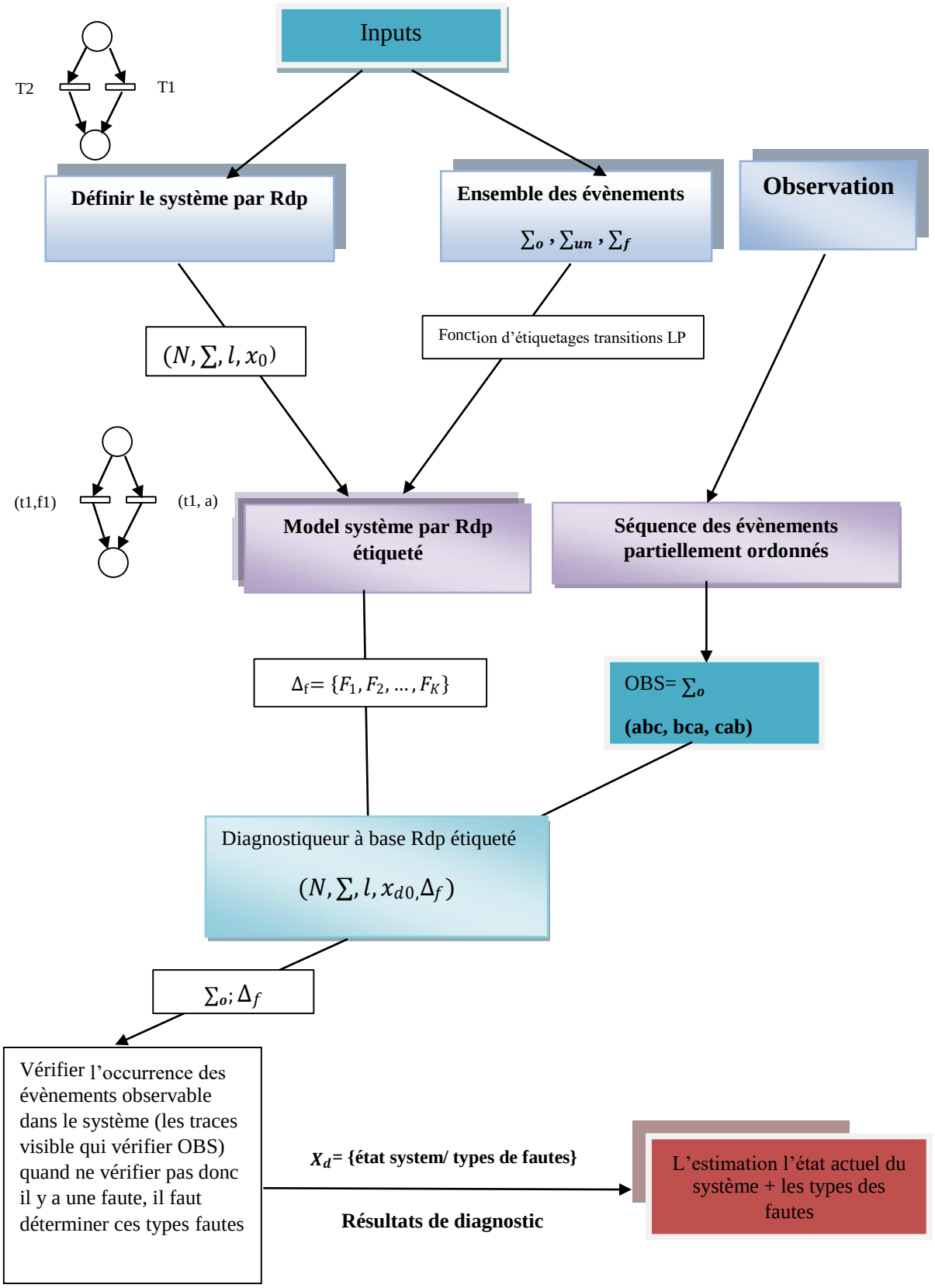


Figure 3.1.Architecture globale du système.

- ***Définir le système*** : prend en entrée le modèle du système sous forme d'un Rdp représentant le comportement du système à diagnostiquer (des places, des transitions et fonctions d'étiquetage pour étiqueté les transitions de ce Rdp) et produit en sortie une représentation d'un Rdp étiqueté. Les étiquettes dans ces transitions de Rdp représentés par 3 choix des éventements : observable (normale), non observable, et des éventements (des types de fautes).

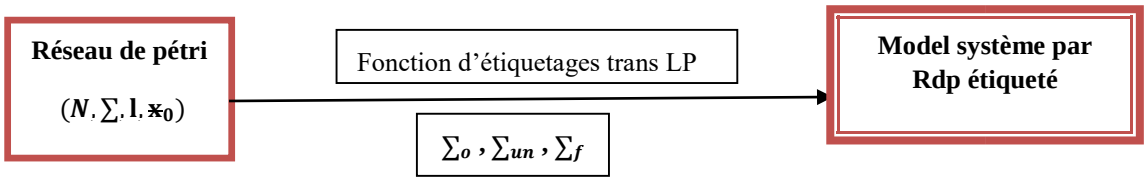


Figure 3.2.Module définir un système.

- ***Observation*** *:* c'est un ensemble des éventements partiellement ordonnées a observé réellement ou bien des évènements normaux, elles sont donnés les informations des observe pour le module diagnostic qui, elle représente le comportement du système (fonctionnement correct), qui a fournée (des traces visibles) vérifier par le diagnostic du système.

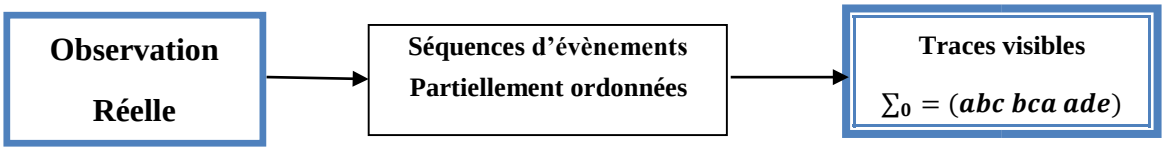


Figure 3.3.Module d'observation.

- ***Diagnostic :*** C'est la tâche principale de notre projet.Le module de diagnostic a pour but d'estimer l'état actuelle de systèmes estimer l'état actuel et quels sont les fautes qu'ont été exécutées par le système d'arriver à ce dysfonctionnement (pannes), En prend en entrée l'ensemble des événements partiellement ordonnées obtenu du module observation par la vérification des traces visibles ou bien diagnostiqué tous les chemins du système c à dire chaque itération franchirai des transitions (mouvement de jeton) détecte et identifier la faute et trouvera en sortie une résultat de diagnostic représente sous forme des états de diagnostic par un couple {nouveau marquage* type de fautes}.

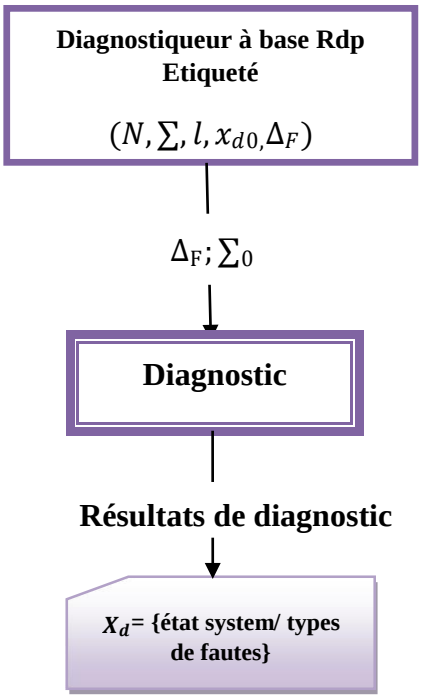


Figure 3.4.Module Diagnostic.

3.3.2. Conception détaillée

La conception détaillée consiste à déterminer les algorithmes pour définir chaque fonction logicielle.

3.3.2.1 Structure de données

Comme nous avons montré dans la conception générale, les structures de données utilisées est qui doivent représenter le diagnostiqueur Rdp étiqueté, les observations et le et l'état de diagnostic sont comme suivant :

Un réseau de pétri étiqueté est représenté par des classe :

- Places c'est l'ensemble des places.
- Transitions c'est l'ensemble des transitions.
- fonction d'étiquetage de transition. - Ensemble évènements de fautes.

Sachant que chaque place est représentée par :

Classe place :

- Id place : représente l'identificateur de la place.
- Nom place : représente le nom de la place.
- Jeton : représente le marquage actuel de la place où Pour les transitions, on a :
- Id_trans c'est l'identificateur de la transition.

- Nom_trans représente le nom de la transition.
- la fonction d'étiquetage transition.
- Ensembles des événements normaux et inobservables et des évènements de fautes.

➢ Les observations sont représentées par un ensemble des événements partiellement ordonnées.

3.3.2.2 Les principaux algorithmes

Dans cette section, on va fournir les algorithmes que nous jugeons importants. Dans le module définir model système on commence par l'algorithme 1 qui sert à franchir les jetons des les places grâce à transitions et l'algorithme 2 de Diagnostiqueur Rdp étiqueté program.

Algorithm_franchissement_jeton.

```
P← {} ;
T← {} ;
Debut
    Foreach (Var element in list place) Faire
        Si (element.P1.Name == "P" + list_transition [etat].From_Place) Alors
element.N_jeton1← element.N_jeton1 - 1;
                element.P1.Text ← element.P1.Name + " " + element.N_jeton1;
        Finsi
        Si (element.P1.Name == "P" + list_transition [etat].To_Place) Alors
element.N_jeton1 ← element.N_jeton1 + 1 ;
                element.P1.Text ← element.P1.Name + " " + element.N_jeton1;
        Finsi
                _ligne [element.P1.Name] ←element.N_jeton1 ;
 Print("okaay"+ element.N_jeton1);
 Fin.
```

```
Algorithme état de diagnostic.
      Var: dtC: DataTable();   Boolean dtc_t = false;  All_rowse:entiers;
         List<String> obs = new List<String>();
         List<String> nonObs = new List<String>();
         List<String> faut = new List<String>();
         List<TransitionClass> list_transition = new List<TransitionClass>();
      List<PlaceClass> list_place = new List<PlaceClass>();
Debut         f_n = 0:entier;   etat = 0:entier;   private void
button7_Click(object sender, EventArgs e)

      DataTable dtC= new DataTable();
            All_rows ←0;
               Si (dtc_t == false)Alors
foreach (var element in list_place)faire
dtC.Columns.ajouter(element.P1.Name);

            }
```

```
  dtc_t = true;
            Si (etat == list_transition.Count) Alors
transition.Enabled ←false;
            Si (f_n > 0) Alors
               Si (f_n == 1) Alors
                  Ecrire ("Le Système a une faute de type f" + 1);
Sinon
                   ecrire("Le Systéme a des fautes de type f" + 1 + ",f" + 2);
           Finsi
```

3.4. Conclusion

Le contenu de ce chapitre représente le développement d'un outil assurant le diagnostic des systèmes à base Rdp étiqueté. En suivant l'esprit du génie logiciel, on a commencé par une définition des besoins suivie par la conception globale de notre système. La section conception détaillée fournit les structures de données avec les principaux algorithmes utilisés.

Chapitre 4

L'implémentation et Réalisation

4.1. Introduction

Après les étapes d'analyse et de conception, nous démarrons la réalisation de notre outil. Cette étape nous a permis de mettre au point tout ce que nous avons étudié dans le chapitre précédent. Ce chapitre est composé de deux parties : (i) une première partie qui représente les différents outils informatiques et langages de développement que nous avons choisis pour entamer ce travail, (ii) une deuxième partie qui représente explicitement l'outil réalisé par montrer son interface, citer les différentes fonctionnalités qu'il offre à l'utilisateur et finir par ajouter un test de notre outil sur un exemple explicatif.

4.2. Outils et langages de développement

Durant l'étape de réalisation de ce projet, il a été nécessaire de recourir à certains domaines de la programmation et nous avons eu l'occasion de nous familiariser avec divers techniques et outils logiciels de développement qui seront présentés ci-dessous.

1.2.1. C_Sharp

Le C est un langage de programmation de bas niveau très populaire, crée dans les années 1970 par D.Ritchie et B.W.Kernighan. Ce langage est un langage orienté objet qui doit permettre aux développeurs de créer facilement des applications pour la plateforme .Net de Microsoft. La comparaison semble très complète, à lire. Il est portable, libre, faiblement typé (peu de types de variables différents : son fonctionnement est proche de l'ordinateur (gain en rapidité), mais un brin plus difficile à manipuler pour le programmeur). Le C n'est sans doute pas le langage le plus facile à apprendre (notamment à cause de l'adoption du concept parfois un peu obscure des pointeurs), ni le plus récent, mais ses qualités font de lui un langage incontournable en matière de programmation. [site3]

Figure 4.1.Langage Sharp.

Les avantages du langage « C » sont

- langage tout objet (à l'exception des types primitifs).
- grande robustesse (langage fortement typé, gestion automatique de la mémoire, gestion des erreurs, etc.).
- gestion de la sécurité, portable, libre.
- forte capacité d'intégration aux environnements web.
- facilité d'écriture d'interfaces graphiques professionnelles. Les inconvénients du langage C sont :
- un brin plus difficile à manipuler pour le programmeur. [site5]

4.2.2. L'outil de développement Visual Studio

The Visual Studio est un EDI (Environnement de Développement) qui permet la création d'application grâce à différent langage de programmation C++, C#, Visual Basic

JavaScript, XML, HTML, CSS. L'idée est de vous aider à écrire votre code plus rapidement, et plus proprement. En effet, vous allez pouvoir apprendre de ces suggestions, et écrire du meilleur code. Il permet de développer des applications indépendante (integrated development environment est une rampe de lancement de création que vous pouvez utiliser pour modifier, déboguer et créer du code, puis publier une application. Un environnement de développement intégré (IDE) est un programme riche en fonctionnalités qui peut être utilisé pour de nombreux aspects du développement logiciel. Au-delà de l'éditeur et du débogueur standard fournis par la plupart des IDE, Visual Studio comprend des compilateurs, des outils de complétion de code, des concepteurs graphiques et de nombreuses autres fonctionnalités pour faciliter le processus de développement logiciel. [site6].

Figure 4.2.Visual studio2017 C.

Principal avantage de Visual Studio

Qui offre en une seule application tout ce qu'il faut pour gérer de bout en bout l'application : design des pages HTML, écriture du code, tests, mise en production sur serveur distant [site4]. Une autre fonctionnalité importante d'un IDE est le débogueur intégré. Le débogueur permet de vérifier s'il y a des erreurs dans le code, et d'aider à identifier la ou les causes.

- un IDE complet est un éditeur unique dans lequel vous pouvez écrire, compiler, lancer, déboguer et publier votre application ;
- grâce à IntelliSense, vous pouvez écrire du code de meilleure qualité plus rapidement, à l'aide des suggestions ;
- avec le débogueur intégré de votre IDE, vous pouvez repérer rapidement les erreurs dans votre application, et recevoir une aide pour les résoudre. [site2].

C'est pour toutes ces raisons que j'ai choisi Visual Studio comme outil de développement, de plus, il est relativement simple d'utilisation.]

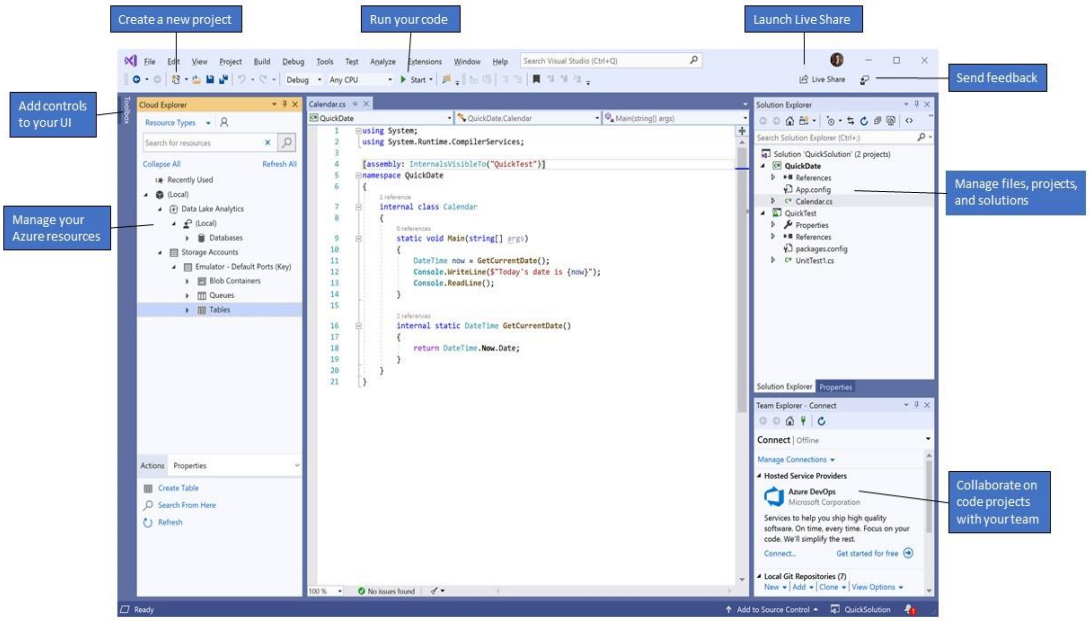


Figure 4.3.Avantage de Visual studio.

4.3 Implémentation

4.3.1 Environnement Matériel

Notre système était développé sur un ordinateur "Condor" qui possède les caractéristiques suivantes :

- Un processeur Intel® Pentium® Core I5 CPU 2410@ 2.30 GHz.
- Une mémoire vive (RAM) de 4 Go sous Windows 10.
- Un disque dur 500 Go.

4.3.2 Les captures d'écran

Dans cette partie on présenter quel que capture d'écrans pour voir le résultat de notre travaille on va donner une petit description sur interface graphique et leurs fonctionnalité.

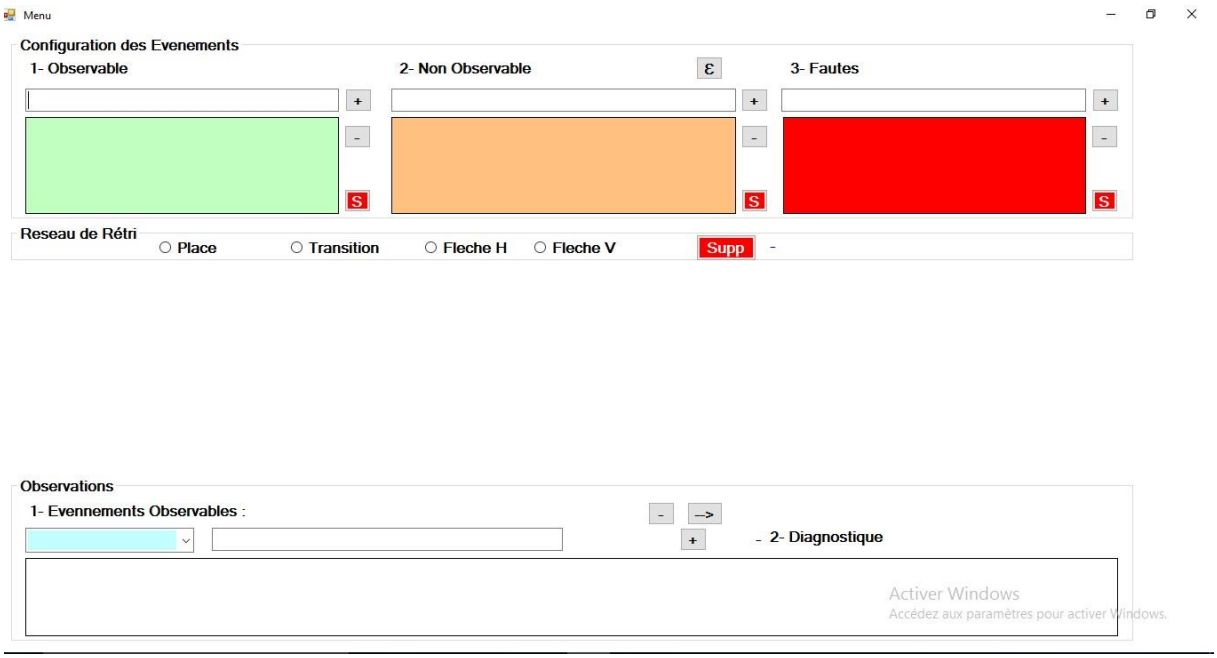


Figure 4.4.Interface graphique d'application.

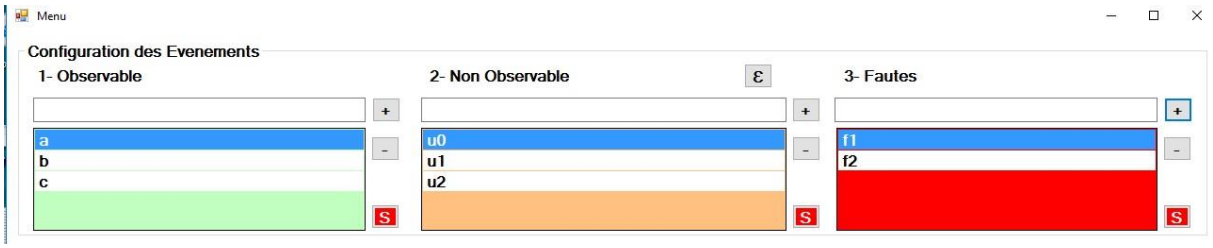


Figure 4.5.Les déférents évènements du système.

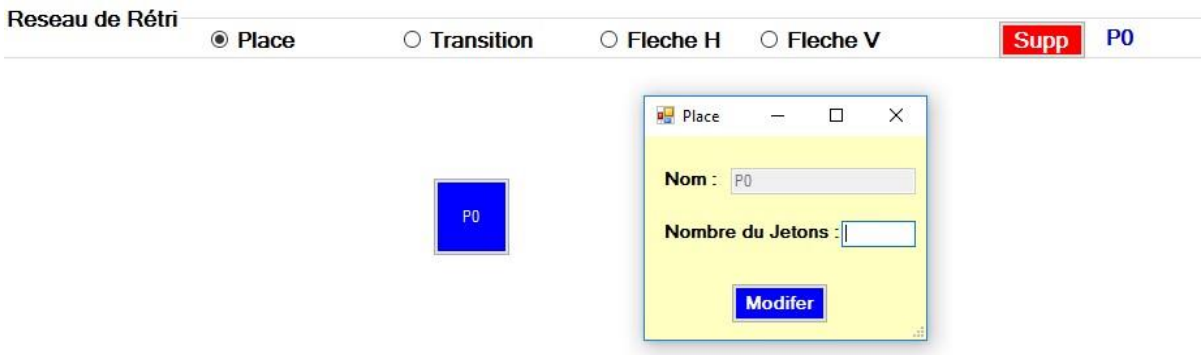


Figure 4.6.Représente comment crée une Place avec le nombre de jeton.

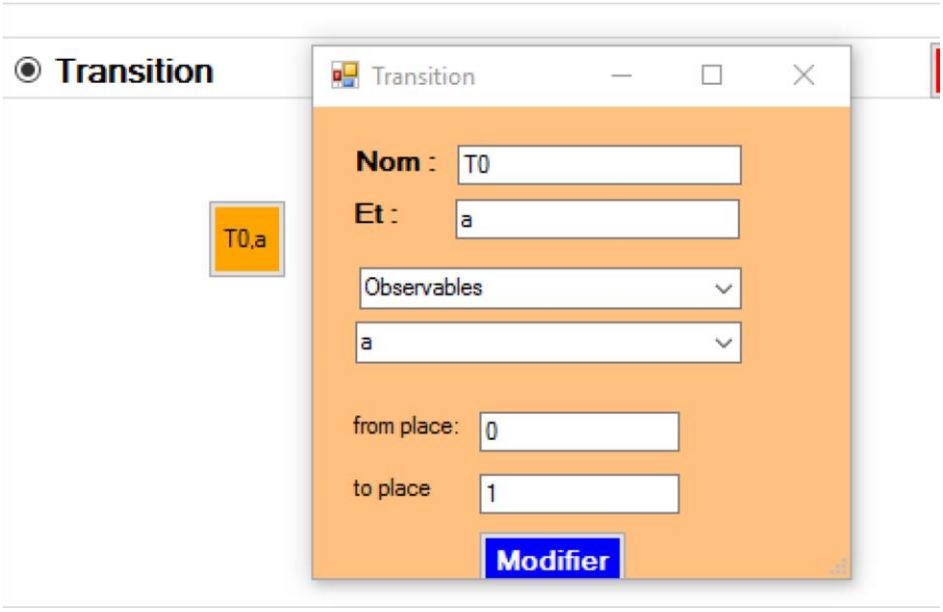


Figure 4.7..Représente comment crée une transition étiqueté.

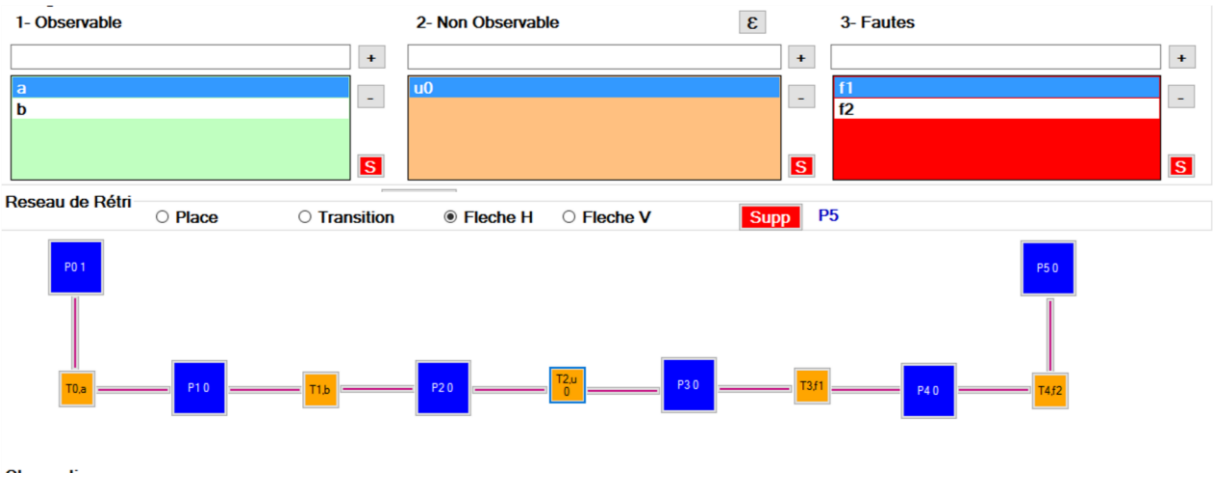


Figure 4.8.Représentation graphique d'un Rdp étiqueté.

Figure 4.9.Représentation de l'observation (les traces visible).

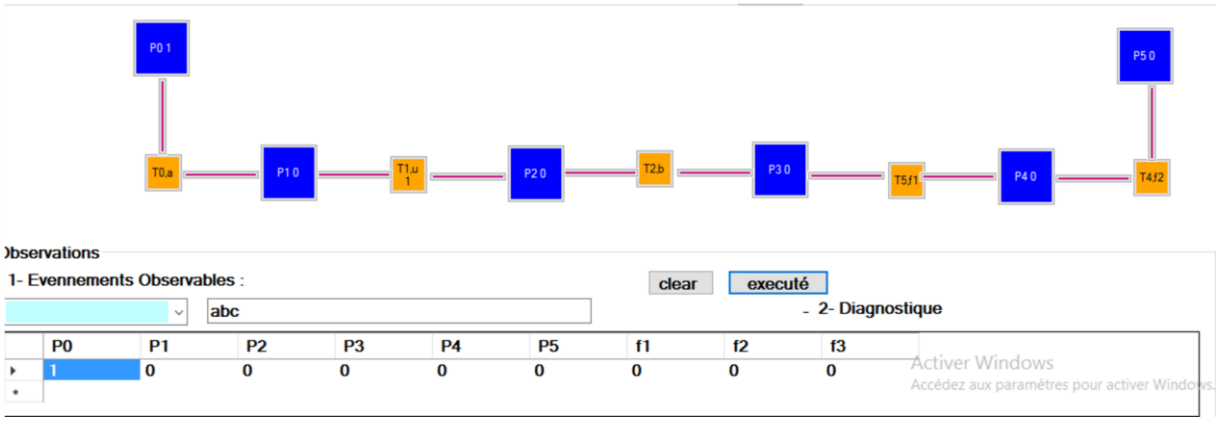


Figure 4.10. Représente l'état de diagnostic.

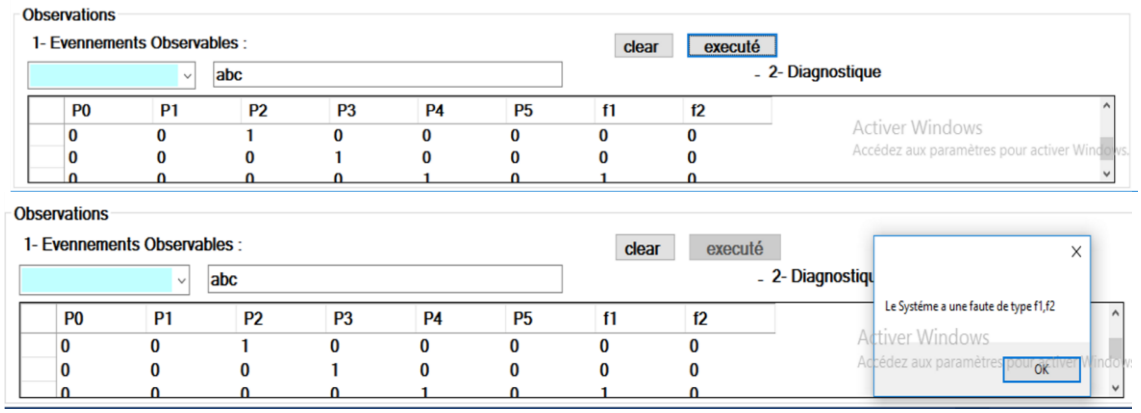


Figure 4.11.Représente l'état du système et les types des fautes.

4.4. Conclusion

Dans ce chapitre, nous avons présenté l'étape de réalisation d'un outil pour la spécification, la modélisation d'un diagnostiqueur, Cet outil est conçu pour être simple, avec une IDE qui permet à l'utilisateur de faire l'édition graphique Rdp, Chaque module est un éditeur qui dispose de ses propres fonctionnalités et de son propre format. Ce travail peut être amélioré par la réalisation d'une interface qui permet l'utilisateur plus libre en ce qui concerne la possibilité de changer les couleurs des places, transitions,arcs.

Conclusion Générale

La fiabilité et la complexité des systèmes à événement discrets et les contraintes de sécurité, ont mobilisé durant ces dernières années une large communauté de chercheurs pour améliorer la surveillance et le diagnostic des systèmes, ainsi la complexité de la tâche de diagnostic a motivé la recherche de son automatisation. De manière générale, le diagnostic est l'estimation de l'état actuel du système menant par acquérir les observations des événements des systèmes soit normal ou la fautif par l'observation réelle puis détecter le comportement fautif du système si il a trouvé ensuite identifier la panne à partir des événements observables et des fautes déterminent préalablement.

L'objectif du diagnostic consiste à estimer l'état actuel du système et identifier les fautes affectant le système étudié :

- Estimation : est définit les états qui peuvent être atteints dans le système si certains événements non observables se produisent.
- Identification : est définie comme un événement de faute ou bien un type de faute

Dans ce mémoire nous avons présenté une approche pour représenter et diagnostiquer les systèmes donnés au moyen de modèles à événement discret. Cette approche est basée sur les Rdps étiqueté. Les Rdps étiqueté sont une classe particulière de Rdp. Dans cette classe, chaque transition a une fonction d'étiquetage et espace d'état et un état initial.

Pour résoudre un problème de diagnostic donné au moyen de Rdps étiqueté ; il faut construire un diagnostiqueur Rdp étiqueté par un algorithme pétri fault program pour résoudre les problèmes de système et identifier le type de panne pour objectif de fiabilité et sureté du système.

Bibliographies

[Pet62]. Carl Adam Pétri. “Communication with automata. PhD thesis, Darmstadt Institut fur Instrumentelle Mathematik, Bonn (Germany), 1962.

[S. Genc and S. Lafortune]. Distributed diagnosis of discrete-event systems using Petri nets. In Application and Theory of Petri Nets, 2003 (Series Lecture Notes in Computer *Science)*, volume 2679, pages 316{336. Springer-Verlag, June 2003.

[C. G. Cassandras and S. Lafortune.] C. G. Cassandras and S. Lafortune. Introduction to Discrete EventSystemsKluwer Academic Publishers, 1999.

Chapitre IV-[Annexe] - CEG4566/CSI4541 – RNM – SITE – uOttawa – Hiver 2013.

[TLFi]. Le trésor de la langue française informatisé, développé par l'unité mixte de recherche ATILF (Analyse et Traitement Informatique de la Langue Française) du CNRS, http://atilf.atilf.fr/tlf.htm.

[PC05] Y. Pencolé et M.-O. Cordier. A formai framework for the deccntralised diagnosis of large scale discrète event systems and its application to télécommunication networks. Artificial Intelligence. Journal, 164(1-2):121 170, 2005.

[BEN19] Hammadi BENNOUI. model-based diagnosis an approach by interacting petri nets. 2019

Webographie

[site1]:https://fr.wikipedia.org/wiki/Réseau de_Petri (toujours en ligne).

[site2] : https://openclassrooms.com/fr/courses/5641796-adoptez-visual-studio-comme-environnement-de-developpement/6140251-decouvrez-les-avantages-dun-ide-complet.

[site3] :https://www.webopedia.com/TERM/C/C_sharp.html.

[site4] :https://forum.hardware.fr/hfr/Programmation/sujet-relatif-104786.htm.

[site5] :https://www.developpez.net/forums/d384238/dotnet/langages/csharp/avantages-inconvenient-csharp-net/.

[site6]:https://www.developpez.net/forums/d519593/general-developpement/programmation-systeme/windows/quoi-sert-visual-studio/

Résumé

Dans cette mémoire, nous traitons le problème de fiabilité et sureté dans les systèmes dynamiques modélisés comme des systèmes à événements discrets ensuite nous présentons une approche de diagnostic du SED modélisée par des réseaux de pétri étiquetés.

Le diagnostic des systèmes à événements discrets (SED) est un domaine de recherche qui a reçu beaucoup d'attention ces dernières années. Les défauts peuvent correspondre à tout événement discret. Ensuite, le diagnostic de fautes de SED, c-à-dire le processus d'identification des causes d'une panne basé sur l'observation des évènements partiellement ordonnées du système. Nous nous intéressons au diagnostic de SED à l'aide de diagnostiqueur basées sur modèle de système un réseau de pétri, le modèle de faute soit disponible. Le système est alors modélisé par un réseau de pétri étiqueté, pour lequel les transitions sont observables si elles sont associées à un événement de sortie, ou non observables ; tout comportement fautif est modélisé par une transition non observable ou bien évènement de fautes. Nous considérons une approche efficace basée sur la notion de diagnostic et leurs propriétés de base qui calcule un état de diagnostic, qui caractérise différents états du système tels que "normal", "fautif». Dans cette mémoire on cherche à implémenter une approche de diagnostic à base de modèle Rdp étiqueté. La technique de résolution estime pour résoudre les problèmes de fiabilité dans les systèmes dynamiques.

Abstract

In this thesis, we treat the problem of reliability and safety in dynamic systems modeled as discrete event systems then we present a diagnostic approach of DES modeled by labeled petri nets.

The diagnosis of discrete event systems (SED) is an area of research that has received a lot of attention in recent years. Faults can be any discrete event. Then, the diagnosis of SED faults, that is to say the process of identifying the causes of a fault based on the observation of partially ordered events of the system. We are interested in the diagnosis of SED using a petri net system model-based diagnostician, the fault model being available. The system is then modelled by a labeled petri network, for which the transitions are observable if they are associated with an exit event, or not observable; any faulty behaviour is modeled by an unobservable transition or a fault event. We consider an approach based on the notion of diagnosis and their basic properties which calculates a diagnostic state, which characterizes various states of the system such as “normal”, “faulty.” In this memory we seek to implement

a diagnostic approach Labeled Rdp-based model the estimation technique for solving reliability problems in dynamical systems.