République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider Biskra
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'Informatique

N° d'ordre : GLSD7/M2/1018

Mémoire

Présenté en vue de l'obtention du diplôme de master académique en

Informatique

Option : **Génie logiciel et système distribués**

Par

Un protocole de coopération entre réseaux de petri coloré avec places communes pour le diagnostic basé-modèles causaux des systèmes distribués

---

Présenté Par :

FRADJ HAMZA

Soutenu le : 25 / 06 /2018

Devant le jury composé de :

Dr. Kahloul Laid	MCA	Président
Dr. Bennaoui Hammadi	MCA	Rapporteur
Mme. Mohammedi Amira	MAA	Examinateur

Juin 2018

Dédicace

*J*e dédie ce modeste travail à : A mes parents .Aucun hommage ne pourrait être à la hauteur de l'amour Dont ils ne cessent de me combler. Que dieu leur procure bonne santé et longue vie.

A mes frères Abdou,Nizar et mon petit frère Thamer,ma sœur Mayada et aussi Moni sans oublié ma Grand-mère .

A l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que dieu te garde dans son vaste paradis, à toi mon père.

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur ; maman que j'adore.

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, et qui m'ont accompagnaient durant mon chemin d'études supérieures, mes aimables amis, collègues d'étude, et frères de cœur, toi Mahdi,Abd Allatiff et Ala –Edinne,Hossem,Saleh Edinne j'ai sans oublié Geddab-Amine.

Et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci.

Remerciement

Nous tenons avant tout à remercier Dieu tout puissant de nous avoir donné la force et la volonté pour achever ce modeste travail. Nous remerciement les plus distingués au

Mr.Hammadi Bennoui qui nous a honorés son encadrement, par sa présence toujours avec nous, pour sa direction son orientation, sa modeste, ces conseils et toutes ces remarque constructives pour le bon déroulement de notre projet.

Nous remercions beaucoup l'ensemble des enseignant du département d'informatique pour la formation qu'il nous ont assurée tout le long de notre cursus universitaire, ainsi que

le membre de jury qui prit la peine d'évaluer notre travail.

On n'oublie pas mes parents pour leur contribution, leur soutien et leur patience.

Finalement, nous adressons nos plus sincères remerciements à tous nos amis proches, Qui nous ont toujours réconforté et encouragé au cours de la réalisation de ce travail.

Résumé

Dans le cadre de ce projet, nous avons présenté une extension du formalisme *CCPN* qu'à été proposé comme outil de représentation de modèles causax de comportement. Une extension pour permettre de prendre en compte l'aspect modularité ; c'est-à-dire, le modèle du système en question sera donné sous forme de plusieurs modèles CCPNs en interaction. Cette dernière est modélisée via l'usage de places communes entre modèles CCPNs. En fait, les CCPNs ont été définis pour permettre d'accomplir le diagnostic centralisé à base de modèles causaux. La technique de diagnostic exploite une analyse en arrière du graphe d'accessibilté à partir d'un marquage final correspondant à l'observation faite sur le système sous diagnostic. Notre apport consiste donc à définir un protocole de coopération entre CCPNs (entre diagnostiqueurs locaux) pour la vérification de la cohérence globale entre les diagnostics obltenus localement. Ceci est accompli via une analyse en avant des graphes d'accessilité construits par chaque diagnostiqueur depuis chaque diagnostic calculté localement. Le développement d'un prototype logiciel implémentant un tel protocol est décrit.

Mots clés : CCPNs, modèles causaux, places de frontières, diagnostic basé-modèle.

Abstract

In the context of this master project, we presented an extension of the CCPN formalism which has been proposed as a tool to represent behavioral causal models. An extension to handle modularity aspect ; i.e., the whole model of the considered system will be viewed as a set of interacting CCPNs models. The interaction is modelled by using common places between CCPN models. In fact, CCPNs are defined to accomplish centralized diagnosis on causal models. diagnostic technique exploits a backward analysis of reachability graph constructed from a final marking corresponding to the made observation. Thus, our contribution consists on defining a cooperation protocol among CCPNs (between local diagnosers) to verify global consistency between local diagnoses. Such a consistency is accomplished by a forward analysis of reachability graphs constructed by each diagnoser. The development of a software prototype implementing such a protocol is described.

Keywords : CCPNs, causal models, Bordered places, model-based diagnosis.

Table des matières

Table des figures

Introduction

Les systèmes automatisés sont devenus de plus en plus importants dans le monde, à cause de différents services et avantages que ces systèmes nous offrent dans notre vie quotidienne. Cette importance oblige les développeurs à augmenter la productivité de ces systèmes pour répondre aux exigences du marché, mais en même temps elle a augmenté la complexité des moyens de production.

A partir des années quatre-vingt, le problème de diagnostic des pannes dans les systèmes artificiels a reçu beaucoup d'attention. Ceci est dû à son importance en termes de sécurité et d'efficacité de fonctionnement.Différentes approches complémentaires ont été développées par la communauté scientifique, on peut classifier ces approches en deux types. Les premières sont les approches à base d'expériences, elles consistent à utiliser un système expert conçu autour d'une base de connaissance. Ce genre d'approches est applicable dans les domaines où il existe une expérience suffisante. En plus, il est difficile de garantir que la base de connaissance est cohérente.

A cause aux limitations des approches à base d'expériences, de nouvelles approches ont été proposées. Ces approches sont appelées *«Diagnostic à base de modèle»*. Ces techniques *(basée-modèle)* utilisent des modèles de fonctionnement du système à diagnostiquer. On peut distinguer deux types d'approches basée-modèle, celles qui reposent sur des modèles de comportement normal du système *(le bon comportement)* et celles qui s'appuient sur des modèles de comportement anormal *(les modèles fautifs)* du système à diagnostiquer.

Donc, un système de diagnostic reçoit en entrée le modèle du système à diagnostiquer et une observation sur son comportement. Si l'observation indique un mal fonctionnement, il génère un ensemble de diagnostics expliquant l'observation. Par conséquence, un système traditionnel de diagnostic peut être vu comme un système centralisé ayant un modèle global

de tout le système et recevant toutes les signalisations d'observation. Cependant, il y a plusieurs raisons pour lesquelles une telle approche centralisée est inappropriée. Par conséquent, une approche distribuée d'un ensemble de diagnostiqueurs peut offrir une solution.

Dans ce mémoire, nous nous focalisons sur le diagnostic basé sur des modèles causaux. On considère le système à diagnostiquer comme une collection de sous-systèmes en interaction où lorsqu'une panne apparait dans un sous- système, elle peut générer des messages d'erreur et peut propager aux sous-systèmes voisins. Le système de diagnostic est vu comme un ensemble de diagnostiqueurs multiples dont chacun est responsable de diagnostiquer un sous-système du système global et de communiquer avec les diagnostiqueurs voisins pour garantir que ses diagnostics calculés localement sont globalement cohérents.

En particulier, chaque diagnostiqueur a un modèle local du sous-système correspondant et peut recevoir des observations produites par les éléments de ce sous-système. Le modèle local décrit le comportement causal du sous-système ainsi que ses interactions avec ses voisins. Lorsque les diagnostiqueurs observent un comportement anormal, chacun est responsable d'expliquer l'observation locale sur la base de son modèle local. Comme résultat, chaque diagnostiqueur calcule un ensemble de diagnostics locaux. Dans les modèles causaux, les diagnostics vont être donnés en termes d'états initiaux expliquant l'ensemble des symptômes observés en utilisant les relations de cause-effet décrites dans le modèle. Ces états initiaux représentent les perturbations initiales menant le système de se comporter de façon anormale.

D'après l'explication précédente, on peut observer qu'il est possible que les diagnostics locaux des différents diagnostiqueurs sont incohérents dans l'ensemble. Afin d'assurer une telle cohérence et de garantir que ces diagnostics locaux recouvrent complètement ceux globaux qui sont obtenus par un seul diagnostiqueur ayant une vision globale autour du système à diagnostiquer, les différents diagnostiqueurs doivent coopérer pour écarter les diagnostics incohérents.

On utilise dans ce modeste travail une nouvelle technique basée sur les ***CCPNs*** *(Causal Colored Petri Nets)* qui ont été proposés dans[1] pour représenter le comportement causal du système à diagnostiquer. Les *CCPNs* sont une classe particulière de ***CPNs*** *(Colored Petri Nets)* ou chaque transition a une matrice décrivant les jetons consommés et produits lors du franchissement de cette transition. Ils sont utilisés pour représenter le comportement causal

du système examiné. Le diagnostic à base de cet outil génère un ensemble de diagnostics possibles qui expliquent le dysfonctionnement oservé, via une analyse en arrière du graphe de marquages accessibles.

Notre objectif dans ce travail consiste à les étendre pour prendre en considération l'aspect modularité. C'est à-dire, permettre d'avoir un modèle sous forme d'un ensemble de CCPNs en interaction. Ceci peut être accompli en utilisant le concept de transitions de synchronisation entre CCPNs ou via l'usage de places communes (ou de frontières) entre CCPNs. Par conséquent, un protocole de coopération entre les différents CCPNs (en fait, entre diagnostiqueurs) doit être défini pour objectif de garantir la cohérence globale entre les diagnostics calculés localement.

Organisation du mémoire Le but de ce projet consiste à faire une coopération entre deux systèmes basée sur les *CCPNs* qui exploite le graphe de marquages en arrière et avant. Le contenu du mémoire se résume dans les points suivants :

Dans le premier chapitre, organisé lui même en trois sections, on expose un état de l'art. La première section définit le concepts du diagnostic à base de modèles causaux. La deuxième donne les définitions de base des ***PNs*** *et CPNs*. La dernière section discute quelques travaux de la littérature relatifs à l'usage des *PNs* dans le diagnostic.

Le deuxième chapitre donne une présentation détaillée de l'approche de diagnostic à base de *CCPN*, ainsi que Diagnostic des systèmes distribués par CCPNs avec places communes comme outil de représentation et de raisonnement. Enfin, une description d'un protocole de coopération entre modèles *CCPNs* est donnée.

Le dernier chapitre décrit les différentes étapes de développement de notre outil de diagnostic à base des *CCPNs*.

Finalement, ce document est conclu par une brève conclusion et des perspectives possibles.

Chapitre 1

Concepts préliminaires

Il est clair qu'un système est jugé réussi s'il peut fonctionner avec une absence d'anomalies. Ceci nous exige à garantir une surveillance permanente pour que ce système nous offre une qualité supérieure de services quel que soit sa complexité. Par conséquent, les principaux éléments d'un système de surveillance sont la détection, la localisation *(le diagnostic)* et la réparation. [3]

Dans notre étude, nous nous focalisons sur le diagnostic tel-que s'il y a un dysfonctionnement détecté dans le système, on doit identifier les vraies causes d'un tel dysfonctionnement. Le problème de diagnostic dans les systèmes artificiels a reçu beaucoup d'attentions. Nous nous intéressons à l'approche basée modèles causaux où le modèle décrit généralement le comportement causal du système à diagnostiquer en termes d'états et de relations de causalité entre ces états.

Pour cela, nous présentons dans ce chapitre les notions fondamentales du diagnostic basé modèles causaux, puis nous présenterons des définitions sur les réseau de Petri *(RdPs ou PNs)* et les réseaux de Petri colorés *(RdPCs ou CPNs)*. Enfin, une description de comment les *RdP* et les *RdPCs* ont été utilisés pour faire du diagnostic est donnée.

1.1 Diagnostic centralisé

Avant de montrer comment modéliser et résoudre un problème de diagnostic, on va discuter le concept d'un problème de diagnostic et sa solution selon l'approche basée modèle.

1.1.1 Définition (Modèle du système)

Un modèle est une abstraction d'un système, notée SD, écrite sous forme d'un ensemble de formules logiques du 1er ordre spécifiant à la fois :[4]

— BM : représente le modèle comportemental du système où il existe des formules décrivant la structure du système et/ou d'autres formules décrivant le comportement de chaque composant.

— $COMPS$: est un ensemble de constants dénotant les composants du système.

$$SD = (BM, COMPS)$$

Le comportement du système est défini par les comportements de ses composants, le prédicat $AB(\bullet)$ est utilisé pour les bien décrire *(dont l'argument est un composant du système).*

1.1.2 Définition (Diagnostic)

Le diagnostic permet de déterminer les causes d'une défaillance, qui ont entrainé la dégradation du système [5]. En effet, le diagnostic établit un lien de cause à effet entre un symptôme observé et la défaillance constatée.

1.1.3 Définition (Symptôme)

Un symptôme correspond à une ou plusieurs observations qui révèlent d'un dysfonctionnement. Il s'agit d'un effet qui est la conséquence d'un comportement anormal.

1.1.4 Définition (Conflit)

Un conflit est un ensemble de composants qui ne peuvent pas fonctionner collectivement de manière correcte.

1.1.5 Définition (Observation)

Une observation (OBS) est une information obtenue à partir du comportement ou du fonctionnement réel du système. La méthode de diagnostic à base de modèles est basée sur

une comparaison du comportement du système *(observation)* avec le comportement attendu du modèle *(prédiction)*.

S'il y a une différence, alors un processus de diagnostic est invoqué comme indiqué dans le schéma de la figure 1-1

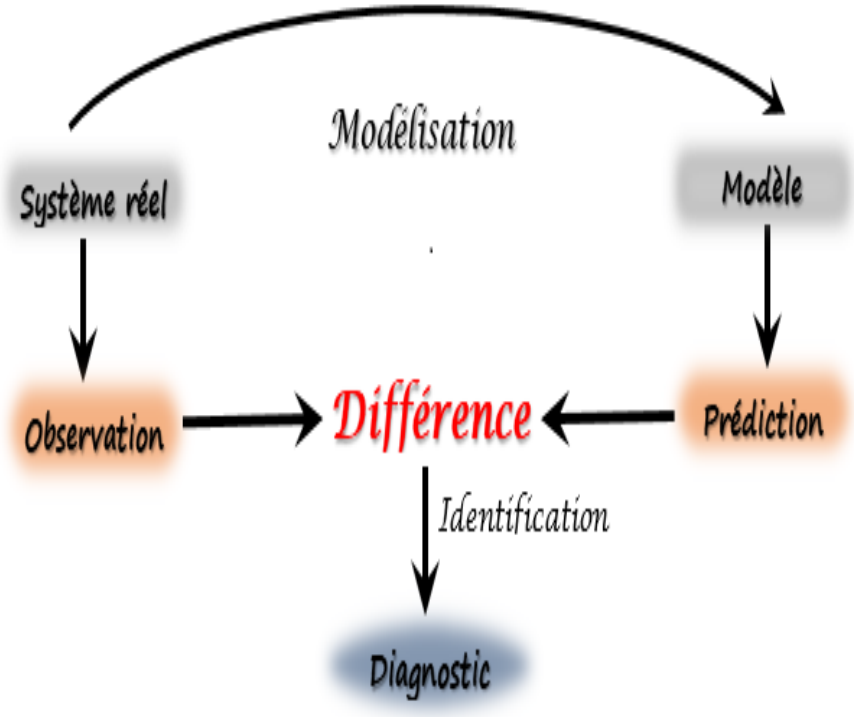

FIGURE 1-1 – *Principe du diagnostic basé modèle.*

Supposant qu'on a un système $S = (BM, COMPS)$ qui est défaillant, ce système doit être réparé, donc il passe par une étape de diagnostic pour expliquer le mal fonctionnement observé. Cette étape consiste à trouver un ensemble Δ des composants ($\Delta \subseteq COMPS$) que lorsqu'ils sont supposés en pannes, vont expliquer pourquoi le système est en dysfonctionnement.

Selon [6], la solution d'un problème de diagnostic selon l'approche basée modèle est de projeter le comportement anormal observé sur le modèle afin de trouver une explication possible pour un tel comportement.

Dans la littérature, deux notions d'explication ont étés proposées.

1. ***Diagnostic basé cohérence*** : Cette approche a été proposée pour diagnostiquer les systèmes en se basant sur leurs modes de comportement normal.
2. ***Diagnostic basé abduction*** : cette approche utilise le raisonnement abductif pour déterminer les diagnostics possibles.

Remarque 1 *Le diagnostic abductif a pour but d'expliquer pourquoi le système réagit comme il est observé, dans ce cas un composant est anormal s'il se comporte comme il est décrit dans le modèle du système.*

1.1.6 Définition (unificatrice)

Un cadre unifié à été proposé en 1992 dans lequel tout problème de diagnostic est transformé en un problème d'abduction avec des contraintes de cohérence [6].

$$AP = (< BM, COMPS >, Ctx, < \Psi^+, \Psi^- >)$$

Où :

— $< BM, COMPS >$: est la description du système, tel que :

— BM : représente le modèle comportemental du système donné sous forme d'un ensemble de formules décrivant la structure et/ou le comportement du système à diagnostiquer.

— $COMPS$: est un ensemble de constants dénotant les composants du système.

— Ctx : sont les données contextuelles. Elles forment un ensemble d'éléments représentant dans le cas d'un modèle causal toutes les hypothèses de fautes possibles du système.

— $< \Psi^+, \Psi^- >$: correspond à l'observation faite sur le système tel que :

— Ψ^+ : est l'ensemble d'observations qui doivent être prédites par une solution *(observations positives).*

— Ψ^- : est l'ensemble des valeurs de paramètres utilisées pour garantir la cohérence *(observations négatives).* D'un point de vue logique, un diagnostic Δ peut être vu comme un ensemble de suppositions de fautes $\Delta \subseteq Ctx$, où :

$$\forall m \in \Psi^+ : BM \cup \Delta \cup Ctx \vdash m$$
$$\forall n \in \Psi^- : BM \cup \Delta \cup Ctx \nvdash n$$

Remarque 2 *Dans cette définition, il y a toute une plage de définitions d'un problème de diagnostic depuis le cas où* $\Psi^+ = \emptyset$. *(purement à base de cohérence) jusqu'au cas où* $\Psi^+ = OBS$ *(Purement abductif).*

1.2 Diagnostic à base des réseaux de Petri

1.2.1 Réseaux de Petri

Historiquement, le concept de réseau de Petri a été développé pour la première fois par Carl Adam Petri, un mathématicien Allemand qui à défini un outil graphique et mathématique permettant de décrire les relations existantes entre des conditions et des évènements. Ils permettent de modéliser le comportement de systèmes à évènements discrets et de capturer divers phénomènes qui les caractérisent à savoir le parallélisme, la synchronisation, le partage de ressources, la concurrence, ...etc [7, 2].

L'avantage des réseaux de Petri est d'une part de se baser sur des fondations mathématiques fortes, et d'autre part de prendre en compte la concurrence. De plus, un grand nombre de logiciels permettent de simuler et d'analyser les réseaux de Petri [8].

Définition

Un *RdP* est un triplet $N = (P, T, F)$où :

1. $P \cap T = \emptyset$.
2. $P \cup T \neq \emptyset$.
3. $F \subseteq ((P \times T) \bigcup (T \times P))$.

P est un ensemble fini non vide de places *(représentées graphiquement par des cercles)*,T est un ensemble fini non vide de transitions *(représentées graphiquement par des rectangles ou des barres)* et F représente l'ensemble des arcs *(représentées graphiquement par des flèches)* reliant les places aux transitions et vice-versa [6].

Dans un *RdP*, la fonction de poids W, est définie par $W: F \longrightarrow N$, si $\forall f \in F: W(f) = 1$, le réseau est dit ordinaire. Pour tout $x \in P \cup T$, l'ensemble d'entrées de x est définit par : ${}^\bullet x = \{y \in P \cup T | (y, x) \in F\}$et l'ensemble de sorties

$x^\bullet = \{y \in P \cup T | (x, y) \in F\}$et pour que x ne soit pas isolé $|{}^\bullet x| + |x^\bullet| \geq 1$. si :${}^\bullet x \neq \emptyset$, x est dit source alors que si $x^\bullet = \emptyset$, x est dit puits.

Le marquage d'un réseau de Petri est une fonction $\mu: P \to N$ définissant pour chaque place le nombre de jetons existant dans celle-ci.

La dynamique du réseau est décrite par le déplacement de jetons dans les places selon la définition suivante de règles de sensibilisation et de franchissement :

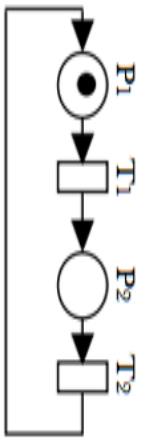

FIGURE 1-2 – *Exemple d'un Réseau de Petri.*

Définition

Soit un *RdP* ordinaire marqué (P, T, F, μ), Une transition $t \in T$ est dite franchissable depuis μ si et seulement si :

$$\forall p \in {}^\bullet p, \mu(p) \geq W(p, t)$$

Si t est sensibilisée depuis μ, alors t peut être franchie produisant un autre marquage

$\mu^{'}$tel que pour tout $p \in P$ on a :

$$\mu'(p) = \mu(p) - W(p,t) + W(t,p)$$

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[9].

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

Matrice d'incidence avant (pré) : $W^{-} = \lfloor W_{ij}^{-} \rfloor o\grave{u}\, W_{ij}^{-} = pr\acute{e}(P_i, T_i)$

Matrice d'incidence arrière ($post$) : $W^{+} = \lfloor W_{ij}^{+} \rfloor o\grave{u}\, {W_{ij}}^{+} = post(P_i, T_i)$

Où : [10]

— $Pr\acute{e}$ est une application de$P \times T \to N$ dite d'incidence avant.

— $Post$ est une application de $P \times T \to N$ dite d'incidence arrière.

— $Pr\acute{e}(P_i, T_j)$ est appelé poids de l'arc reliant P_i et T_j.

— $Post(P_i, T_j)$ est appelé poids de l'arc reliant T_j etP_i.

Exemple 1.1.

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

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

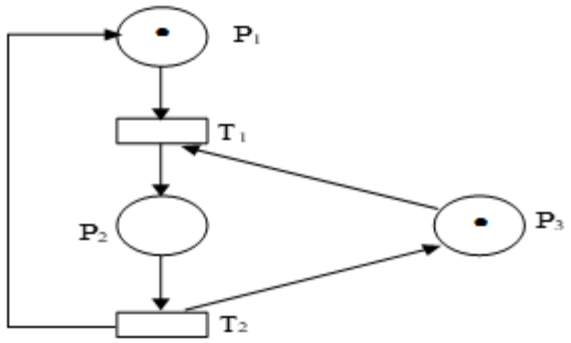

FIGURE *1-3 – Exemple d'un RdP.*

La matrice d'incidence de ce réseau est :

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

Graphe de marquages accessibles

L'évolution temporelle d'un *RdP* peut être décrite par un graphe de marquages représentant l'ensemble des marquages accessibles et d'arcs correspondant aux franchissements des transitions faisant passer d'un marquage à l'autre depuis un marquage initial M_0.

Le graphe des marquages d'un réseau (R, M_0) est un graphe orienté dont les nœuds sont les marquages de $A(R, M_0)$, et chaque arc relie un marquage à un autre qui est immédiatement accessible par une transition si $M_0 \xrightarrow{t} M_1$. Un arc est tracé de M_0 à M_1 et il marqué avec t.

Algorithme 1.1 Construction du graphe de marquages ;

entrée : marquage M ;

début

var $s \leftarrow M_0$;

pour toute transition t **faire**

si $M \xrightarrow{t} M_1$ **alors**

si $M_1 \notin S$ **alors**

$S \leftarrow S \cup M_1$;

créer le sommet M_1 ;

fin si

créer l'arc (M, M_1) ;

appeler l'algorithme 1.1 avec M_1 ;

fin si

fin pour

fin

Ce type de graphe donne une vue simple de l'évolution temporelle d'un réseau, néanmoins le graphe des marquages n'est pertinent que pour les réseaux bornés ; un réseau non borné a une infinité de marquages et ne pourrait être représenté.

L'algorithme de construction du graphe est définit récursivement, partant de l'état initial, on détermine de proche en proche les marquages accessibles. [11]

Soient M_i des sommets, si et seulement si : $M_1 \xrightarrow{t} M_2$.

1. $M_1, M_2 \in A(R, M)$.
2. $t \in T_i$.
3. M_2 est accessible par t à partir de M_1.

Si on applique l'algorithme sur le réseau de la Figure 1-3, on obtient le graphe suivant :

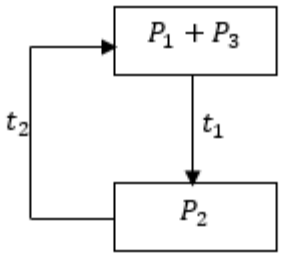

FIGURE 1-4 – *Graphe de marquages accessibles.*

Pour les problèmes où le modèle est grand ou composé de pièces identiques, il est bien connu que les réseaux de Petri colorés *(CPN)* sont bien adaptés à l'utilisation par rapport aux réseaux de Petri classiques. Au moyen de primitives de type de données, il est possible d'obtenir un modèle réduit sur le comportement du système examiné.

1.2.2 Les réseaux de Petri colorés (RdPCs)

Les *RdPCs* sont des outils de modélisations permettant de représenter l'aspect statique et l'aspect dynamique du système dans la même structure. C'est un graphe biparti dont les nœuds sont des places et des transitions reliées entre eux par des arcs orientés.

La notion de couleur dans un *RdP* apporte une sémantique de différentiation des jetons. Dans un réseau de Petri coloré, les jetons possèdent une valeur typée. Ils ne sont plus anonymes comme dans les *RdPs* ordinaires. Cette sémantique amène une grande expressivité et compacité au modèle *RdPCs*. Les jetons peuvent changer de valeur lors de tirs de transitions grâce aux expressions associées aux arcs. Les tirs de transitions peuvent être conditionnés par les valeurs de jetons grâce aux gardes.

Définition informelle

Chaque place p est caractérisée par un domaine de couleurs $C(p)$. Un jeton d'une place p est un élément de $C(p)$. Chaque transition t est caractérisée par un domaine de couleurs $C(t)$. Le domaine de couleurs d'une transition caractérise la signature de cette transition. Les fonctions de couleurs sur les arcs déterminent les instances de jetons nécessaires, consommées et produites lors du franchissement d'une transition.

Définition

Selon [12]un *RdP* coloré est un 9-uplet $RDPC = (S, P, T, A, N, C, G, E, I)$, où :

S : Un ensemble non vide de types, dit ensemble de couleurs.

P : Un ensemble fini de places.

T : Un ensemble fini de transitions.

A : Un ensemble fini d'arcs tel que : $P \cap T = P \cap A = T \cap A = \emptyset$.

N : Une fonction Nœud : définie de $A \subseteq P \cup T * T \cup P$

C : Une fonction couleur, définie de $P \Rightarrow S$.

G : Une fonction « guard », définie de T dans Expressions tel que :

Pour tout $t \in T[Type(G(t)) =$ Bool et Type $(Var(G(t))) \in S]$.

E : Une fonction *"arc expression"*, définie de A dans expressions, tel que :

Pour tout $a \in A$: $[Type(E(a)) = C(p)MS$ et $Type(Var(E(a))) \in S]$, où p est la place dans $N(a)$.

I : Une fonction d'initialisation, définie de P dans l'ensemble des expressions clos :

Pour tout $p \in P$: $[Type(I(p)) = C(p)MS]$.

En revenant à [12], les concepts de base ont été définis comme suit :

Définition (Instanciation)

Une instanciation *(binding)* b pour une transition t, est une fonction définie comme suit :

— Pour toute variable v de $Var(t)$, $b(v) \in type(v)$.

— L'instanciation b satisfait $G(t)$, et on note ceci par : $G(t) < b >$

Remarque 3 *Le nombre d'instanciation peut être infini, et l'ensemble de toutes les instanciations possibles pour la transition t est noté : $B(t)$.*

Définition (Un élément jeton)

Un élément jeton *(token element)* est un couple (p, c) où p est une place et $c \in C(p)$. Notons par Te l'ensemble de tous les éléments jetons

Définition

Un élément instancié *(binding element)* est un couple (t, b) où t est une transition est $b \in B(t)$. Notons par Be l'ensemble de toutes les instanciations d'éléments.

Définition

— Un marquage : est un multi-ensemble sur Te, tel que $M : Te \longrightarrow N$.

Exemple

$$(p1, 2) + +(p2, 6) + +(p2, 9) + +3'(p3, "a")$$

— Une étape : un multi-ensemble fini non vide sur Be ou $Y : Be \longrightarrow N$.

Exemple

$$(t_1, < x = 2, y = 5 >) + 2'(t_2, < x = 5, y = 9, a = "d" >)$$

Remarque 4 *Le marquage initial M_0 est donc la fonction (ou le multi-ensemble) qui doit satisfaire :*

$$\forall (p, c) \in Te : M_0(p, c) = (I(p))(c).$$

Définition (condition de franchissement d'une étape)

Une étape Y est franchissable dans un marquage M_1 si seulement si :

$$\forall p \in P : \sum_{(p,t) \in Y)} E(p, t) < b > \leq M_1(p).$$

Dans ce cas, Y peut être tirée depuis M_1 et son tirage mène vers un nouveau marquage M_2 défini par :

$$\forall p \in P : M_2(p) = (M_1 - \sum_{(p,t) \in Y)} E(p, t) < b > + \sum_{(p,t) \in Y)} E(p, t) < b >$$

Ceci est noté par : $M_1[Y > M_2$.

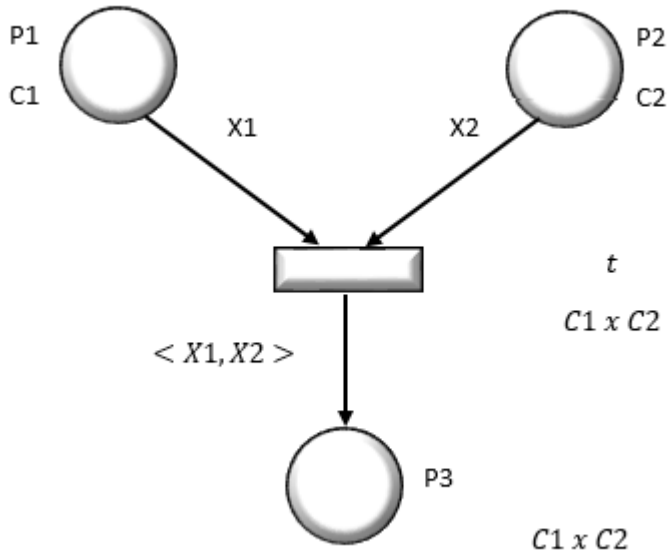

FIGURE 1-5 – *Exemple de franchissement.*

Soit $x1 \in C1$,$x2 \in C2$:

— t est franchissable pour $(x1, x2)$ si et seulement si :

1. $P1$contient au moins un jeton de couleur$x1$ $(X1(x1, x2) = x1)$.
2. $P2$contient au moins un jeton de couleur $x2$.

— Si on franchit t pour $(x1, x2)$ alors :

1. Un jeton de couleur $x1$ est retiré de $P1$.
2. Un jeton de couleur $x2$ est retiré de $P2$.
3. Un jeton de couleur $< x1, x2 >$ est produit dans$P3 :< X1, X2 > (x1, x2) =< x1, x2 >$.

1.2.3 Définition d'un modèle BPN

Une classe particulière de réseaux de Petri, appelée réseaux de Petri Comportemental (BPN), a été proposée pour modéliser le comportement causal du système à diagnostiquer. De plus, des notions formelles concernant les problèmes de diagnostic et leurs solutions ont été reformulées dans le cadre des BPN. L'objet de cette section est de montrer comment un BPN est utilisé pour représenter un modèle de comportement causal d'un système et comment le raisonnement du diagnostic basé sur un tel modèle[13].

Modèle Causaux

Dans les modèles causaux, le comportement d'un système peut être décrit par un ensemble d'états et de relations entre ces états. Chaque état modélise en fait un composant de l'état global du système, les relations décrivent les relations de cause-effet entre ces états [6].

Pour le diagnostic, il est nécessaire de distinguer entre au moins trois types d'entités :

— États initiaux : sont des états qui n'ont aucune cause dans le modèle. Ils représentent, dans le cas de modèle de comportement anormal, les perturbations initiales menant le système à suivre un comportement fautif.

— États internes : sont les conséquences des états initiaux. Ils sont associés aux éléments du système qui ne sont pas susceptibles de faire partie de la solution, parce qu'ils peuvent être expliqués par d'autres éléments d'états initiaux.

— Manifestations : sont des états observables. Dans le cas où le modèle représente un comportement anormal, ils représentent les symptômes attendus. Une solution à un problème de diagnostic à base de ce type de modèles consiste à expliquer un ensemble de manifestations en termes d'états initiaux, en utilisant les relations de cause-effet décrites dans le modèle. Un modèle BPN représentant le comportement causal d'un système donné a été défini dans [6] comme suit :

Définition

Un réseau de Petri comportemental (BPN) est un 4-uplets$N = (P, T_N, T_{OR}, F)$ tel que $(P, T_N \bigcup T_{OR}, F)$ est un réseau de Petri ordinaire acyclique qui satisfait les axiomes suivants :

1. $\forall p \in P(|{}^\bullet p| \leq 1 \wedge |p^\bullet| \leq 1$.
2. $\forall p1, p2 \in P(({}^\bullet p1 = {}^\bullet p2) \wedge (p1^\bullet = p2^\bullet) \longrightarrow p1 = p2)$.
3. $\forall t \in T_N(|{}^\bullet t| = 1 \wedge |t^\bullet| > 0) \vee (|{}^\bullet t| > 0 \wedge |t^\bullet| = 1)$.
4. $\forall t \in T_{OR}(|{}^\bullet t| \geq 2 \wedge |t^\bullet| = 1)$.

Un modèle de ce type se caractérise par les caractéristiques suivantes :

— Chaque place correspond à une instance parmi les états et transitions du modèle causal décrire les relations de cause à effet entre les instances étatiques correspondantes .

— Une place source (c'est-à-dire$p$: ${}^\bullet p = \emptyset$)correspond nécessairement à une instance d'état initiale.

— Une place puits représente soit une instance d'état de manifestation soit une instance d'état interne qui n'a pas de conséquences.

— Un modèle *BPN* est sûr, c'est-à-dire que n'importe quel place peut être marquée avec au plus un jeton.

— L'ensemble des transitions est divisé en deux sous-ensembles T_N et T_{OR}.

— Les transitions en T_N (et les transitions) sont prévues de la manière habituelle ; tandis que les transitions en T_{OR} (Or-transitions) sont destinées à représenter les transitions logiques conjonctives OR et elles représentent des macro-transitions dont la sémantique peut être donnée en termes de réseau Petri avec des arcs inhibiteurs (voir Figure1-6)

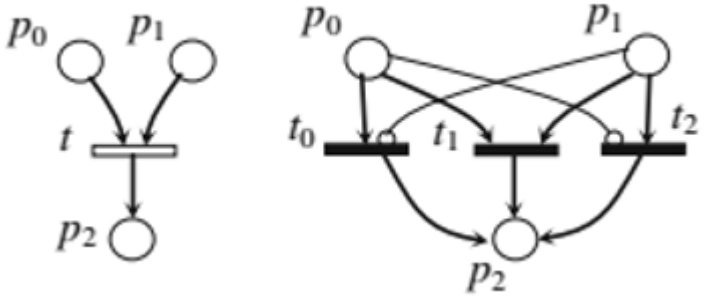

FIGURE 1-6 – *La sémantique de la transition* T_{OR}.

In-formellement, une transition $t \in T_{OR}$ *(graphiquement représentée comme une barre épaisse vide)* à une concession dans un marquage si et seulement si au moins une de ses places d'entrée est marquée.

1.3 Diagnostic par RdPCs

Il est clair qu'un système est jugé réussi s'il peut fonctionner avec une absence d'anomalies. Ceci nous exige à garantir une surveillance permanente pour que ce système nous offre une qualité supérieure de services quel que soit sa complexité. Par conséquent, l'un des principaux éléments d'un système de surveillance est le diagnostic. Il s'agit de déterminer les causes d'une défaillance ou d'une panne du système surveillé en se basant sur les observations qui sont des informations sur son comportement réel.

Comme nous avons signalé au niveau de l'introduction générale, notre objectif dans ce travail est de présenter un diagnostic d'un système distribué défaillant. Le système en question est vu comme un ensemble de sous-systèmes (ou composants) en interaction. Le comportement de chaque sous système est donné via un modèle *RdPC* et les interactions entre sous-systèmes est modéliser par le passage de jetons entre modèles *RdPCs* adjacents. En fait, le diagnostic local par analyse de *RdPCs* à été proposé et définit dans [1] et notre tâche consiste à les utiliser de manière modulaire. Ceci nécessite donc de définir un protocole de coopération entre modèles *RdPCs* pour garantir que les solutions obtenues de manière locales sont globalement cohérentes. On a préféré de laisser la présentation de comment ces *RdPCs* sont exploités pour implémenter le diagnostic au chapitre suivant.

1.4 Conclusion

Après cette définition du diagnostic à base de modèles causaux et de réseaux de Petri colorés, nous essayerons dans le chapitre suivant de donner une présentation détaillée d'une classe particulière des réseaux de Petri colorés *CCPNs* proposée dans *[1]* pour le diagnostic à base de modèles causaux. Pour le cas des systèmes distribués, un protocole de coopération entre diagnostiqueurs associés aux différents sous-systèmes est nécessaire.

Chapitre 2

Les CCPNs comme outil pour le diagnostic à base de modèles causaux

Comme nous avons indiqué au niveau de l'introduction générale, notre objectif est définir et implémenter un protocole de coopération entre réseaux de Petri colorés causaux. Donc, une extension des ***CCPNs*** proposées dans [1] est nécessaire pour modéliser les systèmes distribués.

2.1 Causal Colored Petri Nets (CCPNs)

L'analyse en arrière d'un ***CPN*** semble comme une inversion des arcs du modèle et donc de réaliser l'analyse en avant du modèle inversé. Le problème principal qui se pose ici est le franchissement en arrière d'une transition. En inversant les arcs, leurs expressions doivent être inversées aussi, et de même pour la garde de la transition. Une telle inversion dépend des expressions d'entrée et de sortie de la transition. Elle peut conduire à une explosion combinatoire car l'inversion d'une fonction ne donne pas généralement une fonction mais plutôt une relation.

Afin de faire face à ce problème, une extension particulière appelée *CCPNs* (*Causal Colored Petri Nets*) à été proposée dans [1] où à chaque transition est associée une matrice décrivant les combinaisons possibles de couleurs des jetons d'entrée et de sortie. Par conséquent, les expressions des arcs d'entrée d'une transition sont des variables typées et celles de

sortie sont des expressions simples qui extraient à partir d'une matrice donnée et en fonction des jetons en entrée ceux de sortie possibles. Dans ce qui suit, nous introduisons cette classe de *CPN*. Les *CCPNs* sont introduits pour objectif de représenter de façon plus compacte le comportement causal du système en question. En revenant à [1], les concepts de base ont été définis comme suit :

2.1.1 Définition (Un CCPN)

Un *CCPN* est un6-uplet $N = (\Sigma, P, T, A, C, FW)$ où :

— (Σ, P, T, A, C)est un *CPN*.

— $P = I_c \bigcup I_s \bigcup M_n$:

— $I_c = p \mid p \in P : {}^\bullet p = \emptyset$ [1] ; $M_n \subseteq p \mid p \in P : p^\bullet = \emptyset$ tandis que $I_s = P \backslash \{I_c \cup M_n\}$.

— A^+, dénote la fermeture transitive de A, est non réflexive.

— $FW : T \to MAT_{n,m}$tel que [2] :

— n est le nombre de transitions pour lesquelles une transition t peut être franchie dans un *PN* classique (*c'est à dire, c'est le nombre de façons dont t peut être tirée*).

— $m = |{}^\bullet t| + |t^\bullet|$ définit le nombre de places liées à t, en entrées et en sorties.

2.1.2 Définition (Un CCPN marqué)

Un *CCPN* marqué est une paire (N, μ) où N est un *CCPN* et μ est un marquage tel que : $\forall p \in P : |\mu(p)| \leq 1$. Soit μ_0 le marquage initial de N tel que :

$$\forall p \in P : \mu_0(p) \neq \emptyset \to p \in I_c$$

2.1.3 Définition (Un CCPN sûr)

Un *CCPN* marqué (N, μ) est dit sûr si : $\forall p \in P, \forall \mu \in R(N, \mu_0) : |\mu(p)| \leq 1$. Où $R(N, \mu_0)$ est l'ensemble de marquages atteignables à partir de μ_0.

---

1. ${}^\bullet x \{y \mid (y, x) \in A\}$ *et* $x^\bullet = \{y \mid (y, x) \in A\}$ dénote les ensembles d'entrée et de sortie de x respectivement où $x \in P \cup T$

2. $MAT_{n,m}(\bigcup_{w \in \sum^w})$ définit l'espace des matrices de n lignes et m colonnes.

Notation

— $A(t)$ désigne l'ensemble des arcs d'entrée de t.

— $Var(t)$ désigne l'ensemble des variables qui apparaissent dans les expressions d'arc d'entrée de t et dans sa garde associée$G(t)$. (*il est également utilisé pour les expressions* $Var(expr)$*)*. Dans un modèle $CCPN$: $Var(G(t)) = Var(t)$.

— $E(x_1, x_2)$ dénote l'expression attachée à un arc (x_1, x_2).

2.1.4 Définition (Instanciation)

Une instanciation (*binding)* d'une transition t est une fonction b définie sur $Var(t)$ tel que:$\forall v \in Var(t)$: $b(v) \in Type(v)$. Soit$Var(t) = v_1, \ldots\ldots\ldots v_n : [b(v_1)\ b(v_2) \ldots b(v_n)]$ est la $i^{éme}$ ligne donnée sous forme de sous-vecteur de $FW(t)$.

$Expr < b >$ désigne l'évaluation de l'expression $Expr$ dans l'instanciation b.

Comme une particularité des *CCPNs*, en comparaison avec les *CPNs*, est la non réflexivité de A^+ ; par conséquent, le modèle *CCPN* est acyclique. Ainsi, il est possible d'introduire un ordre partiel, noté $\prec$ entre ses transitions. Un tel ordre est inspiré de ce qu'est défini pour les *BPNs* dans [1]comme suit :

$$t_1, t_2 \in T : t_1 \prec t_2 \Longleftrightarrow t_1 A^+ t_2.$$

2.1.5 Définition (transition franchissable)

Soit μ un marquage. Une transition t est franchissable (enabled) à μ si seulement si :
$\forall p \in^\bullet t : E(p,t) < b > \leq \mu(p)$ et $\nexists t' < t$ tel que t' est franchissable *(enabled)*.

Définition

Soit t une transition franchissable dans un marquage μ, le tirage de t mène vers un nouveau marquage μ' défini par :

$$\forall p \in P : \mu'(p) = \mu'(p) - E(p,t) < b > + E(t,p) < b > .$$

2.1.6 Définition

Étant doné un *CCPN* marqué (N, μ), nous désignons par une étape l'ensemble de transitions activées et concurrentes à μ.

2.1.7 Définition

Étant donné un *CCPN* marqué (N, μ) et une étape $s = t_1, \ldots, t_n$, le tirage de s à μ atteint le nouveau marquage μ' tel que :

$$\mu' = \bigcup_{i=1}^{n} \mu_i,\ \mu[t_i > \mu_i'.$$

Dans un modèle *CCPN*, les places sont utilisées pour représenter les entités du modèle causal du système à diagnostiquer. Il y'a 3 classes d'entités :

— Causes initiales, notée I_c, représentées par des places sources .

— États internes, notée I_s, représentés par des places qui ont des transitions d'entrée .

— Manifestations, notée M_n, représentées par des places puits.

Les transitions représentent les relations de cause-effet entre les places correspondantes.

À chaque transition est attachée une matrice donnée par $FW(t)$. Les lignes de $FW(t)$ représentent les différentes façons qu'une telle transition t peut être tirée alors que chacune de ses colonnes est associée à une place p qui entoure t ; et donc, il détermine les valeurs possibles qui seront *consommées/ produites* à partir *de/dans* p par le tir de t. Chaque matrice de transitions $FW(t)$ peut être divisée en deux sous-matrices $FW_{in}(t)$ et $FW_{out}(t)$. FW_{in} correspond aux combinaisons possibles de couleurs de jetons en entrée de t alors que FW_{out} est la sous-matrice de sortie de t. En outre, chaque transition t peut être classée comme transition *joint* ou $fork$. Remarquons qu ' une transition linéaire est une transition particulière de type *joint* ou $fork$. Une transition *joint* est utilisée, comme d'habitude, pour représenter une conjonction de places, mais aussi, elle est utilisée, dans un *CCPN*, pour traiter les deux cas où il y a plusieurs possibilités concurrentes d'atteindre une place ou lorsqu'il existe un ou exclusif entre au moins deux chemins d'exécution pour atteindre cette place. Comme le marquage d'une place et l'évaluation d'une expression d'arc sont un multi-ensemble, le concept d'ensembles vides est utilisé comme composant d'une matrice de transition joint pour mieux

représenter les cas mentionnés ci-dessus.

Dans le cas où il y a plusieurs évolutions simultanées à partir d'une même valeur dans une place, une transition de type $fork$ sera utilisée pour dupliquer la place requis par le marquage spécifié pour chaque chemin. Pour chaque transition t , les expressions d'arc d'entrée sont des variables typées tandis que l'expression d'arc de sortie est une fonction qui contient comme entrée un vecteur d'instanciation et la matrice de transition et retourne la couleur de jeton de sortie qui correspond à une telle liaison.

Le modèle est sûr, c'est-à-dire n'importe quel place doit être marquée par au plus, un seul jeton coloré[1].

2.1.8 Définition

Soit (N, μ) un $CCPN$ marqué. μ peut conduire à une incohérence si et seulement si :

$$\exists t_i, t_j \in T\,, \exists \mu_i, \mu_j \in R(N, \mu) : \mu[t_i > \mu_i \wedge \mu[t_j > \mu_j \wedge \exists p \in P \mid \mu(p) \neq \mu_j(p).$$

Exemple

Comme exemple d'un modèle $CCPN$, nous considérons un modèle simple, adapté depuis un exemple utilisé pour représenter un modèle partiel fautif d'un moteur de voiture. Cependant, c'est assez représentatif pour introduire les constructions de base de $CCPNs$. La figure2-1 donne la représentation graphique du modèle considéré. Ce modèle est caractérisé par c_1, c_2 et c_3 comme des causes initiales et m_1 et m_2 comme des manifestations. Les places restantes représentent les états internes. Chaque place à un type attaché à elle, qui détermine l'ensemble des couleurs de jetons que le jeton de la place est autorisé à avoir. Comme exemple, les jetons de la place c_1 aura a ou b comme leur couleur de jeton.

Les transitions sont utilisées pour modéliser les relations de cause-effet entre les entités correspondantes du modèle causal. Chaque transition est étiquetée par une matrice qui définit ses modes de tir. t_1 est une transition $fork$ qui sert à dupliquer la place d'entrée c_1 par le marquage a en c_{11} et c_{12} de la même couleur de jeton. Dans $FW(t_1)$, la première colonne correspond à la place d'entrée c_1, tandis que, les deux dernières correspondent aux places de

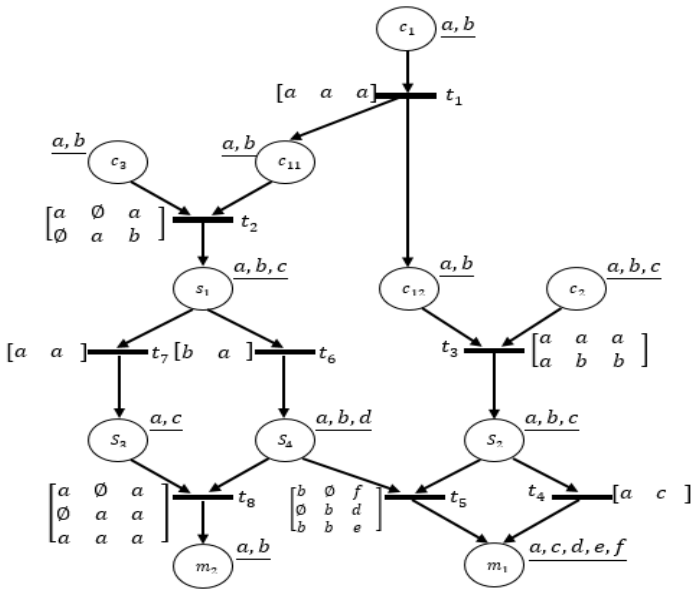

FIGURE 2-1 – *CCPN* exemple [1].

sortie c_{11}et c_{12}. t_2 sera activée seulement si une des places d'entrée est marquée par la couleur a et par conséquent, elle produit la couleur a ou b dans s_1. La transitiont_3 est comme le $et-$ logique tandis que t_5 et t_8 sont comme le $ou-$ logique. La transitiont_8 peut être activée dans trois cas (*selon le nombre de lignes dans sa matrice*), les deux premiers représentent le cas où s_3 ou $s_4$est marquée par une couleur a, tandis que le dernier est que lorsque les deux places sont marquées par la couleur a . La transition t_5 a la même interprétation quet_8, mais seulement, avec t_8 la couleur produite est la même pour toutes les cas (*façons de tir*), et donc, elle est utilisée pour garder la sûreté du modèle (*chaque place est marquée au plus par un jeton à un moment donné*), tandis que pour t_5 la couleur produite est définie selon la présence de couleurs dans s_2 et s_4, et ici, t_5 garde la *cohérence* du modèle.

2.2 Le diagnostic avec CCPN

Les *CCPNs*, comme des *CPNs* particuliers pour représenter le comportement causal d'un système, sont introduits principalement pour traiter le diagnostic de panne. Dans cette section, nous montrons comment le problème de diagnostic peut être formalisé sur la base d'un modèle *CCPN*. Un problème de diagnostic est signalé lorsqu'il apparaît une contradiction entre le comportement requis du système examiné et son comportement réel. Dans le diagnostic basé sur un modèle causal, le comportement observé est donné comme un ensemble de manifestations, noté OBS. Dans les *CCPN*s, il s'agit d'un marquage final μ_{OBS} où les places marquées sont celles qui appartenant à M_n.[1]

2.2.1 Définition

Étant donné un *CCPN* comme un modèle du système S, une observation est un marquage μ^{OBS} tel que :

$$\forall p \in P : \mu^{OBS} \neq \emptyset \rightarrow p \in Mn$$

2.2.2 Définition

Un problème de diagnostic est définie en termes de modèle *CCPN* par le triple : $DP = (N, INIT, < M^+, M^- >)$, où :

- N est le modèle *CCPN*.
- $INIT = (p, c) \mid p \in Ic, c \in C(p)$.
- $M^+ = (p, c) \mid p \in Mnc \in C(p), \mu^{OBS}(p) = c$.
- $M^- = (p, c) \mid p \in Mn, c \in C(p), \mu^{OBS}(p) \neq c$.

Dans cette définition, N représente le modèle comportemental causal du système à diagnostiquer. $INIT$ est un ensemble de couples (p, c) en termes desquels les solutions vont être données. Il se compose d'un ensemble de marques possibles c de chaque place p qui représente une cause initiale dans le modèle causal.$< M^+, M^- >$ représente l'observation faite. M^+ consiste en un ensemble de couples (p, c) représentant les manifestations qui doivent être impliqué par une solution à DP, tandis que M^-est l'ensemble des couples (p', c') qui contredisent l'observation effectuée *(c'est à dire, il est utilisée pour assurer la cohérence*

requise).

2.2.3 Définition

Soit N un *CCPN*,$p \in P$,$c \in C(p)$ et μ_0 un marquage initial :

$$(N, \mu_0) \vdash (p, c) \longleftrightarrow \nexists \mu \in R(N, \mu_0) \mid \mu(p) = c.$$

La définition peut être généralisée pour un ensemble de couples

$$Q = (p, c) \mid p \in P, c \in C(p)$$

comme suit :

$$(N, \mu_0) \vdash Q \longleftrightarrow \exists \mu \in R(N, \mu_0) \mid \forall (p, c) \in Q : (N, \mu_0) \vdash (p, c).$$

La version inversée est donnée par :

2.2.4 Définition

Soit N un *CCPN*, $p \in P$, $c \in C(p)$ et μ_0 un marquage initial :

$$(N, \mu_0) \nvdash (p, c) \leftrightarrow \nexists \mu \in R(N, \mu_0) \mid \mu(p) = c.$$

Pour un ensemble de couples $Q = (p, c) \mid p \in P, c \in C(p)$, il est donné par :

$$(N, \mu_0) \nvdash Q \longleftrightarrow \nexists \mu \in R(N, \mu_0) \mid \forall (p, c) \in Q : (N, \mu_0) \nvdash (p, c).$$

La notion de solution peut être maintenant capturée par la proposition suivante :

Proposition 1 *Etant donné un problème de diagnostic* $DP = (N, INIT, < M+, M- >)$, *un marquage initial* μ_0 *est une solution àDP si et seulement si :*

$(N, \mu_0) \vdash M^+$ *et* $(N, \mu_0) \nvdash M^-$*Cela signifie que* μ_0 *doit tenir compte de toutes les observations de*M^+, *alors qu'aucune dans* M^- *ne doit être atteinte à partir de* μ_0.

2.3 Résolution de problèmes de diagnostic avec CCPNs

Afin de résoudre un problème de diagnostic donné par un *CCPN*, nous sommes intéressés par une technique d'analyse en arrière du graphe d'accessibilité proposée dans [1]. Dans cette section, on tente de rappeler le principe de cette technique d'analyse d'un modèle *CCPN* afin de générer un ensemble de diagnostics possibles qui expliquent un dysfonctionnement donné. En générale, une analyse en arrière du graphe d'accessibilité permet de déterminer les marquages à partir desquels un marquage donné est accessible. Elle peut être faite par une simple inversion de la direction des arcs dans les *PNs* classiques, tandis que, la règle de tir d'une transition reste la même que pour le franchissement en avant. Dans les *CPNs*, l'inversion doit être appliquée sur les expressions des arcs et les gardes de transitions qui sont généralement des fonctions. Pour les transitions linéaires, le processus d'inversion est trivial. Dans le cas de transitions de type $fork$ ou $joint$, le processus d'inversion devient compliqué. Ainsi, il est possible de tomber dans un problème d'explosion combinatoire. Dans un *CCPN* en plus de l'inversion de la direction des arcs, le processus d'inversion peut être réalisé simplement par une réorganisation des matrices de transition FW. De cette façon, la sous-matrice d'entrée FW_{in} devient la sortie, notée b_FW_{out} ; alors que la sous-matrice de sortie FW_{out} devient d'entrée, notée b_FW_{in}. En conséquence, FW devient b_FW. Ainsi, les expressions d'arcs doivent être changées. Comme d'habitude, chaque expression d'arc d'entrée d'une transition t est une variable typée ; tandis que, chacune de sortie égale à l'expression suivante$f([v_{qi}..v_{qm}], b_FW(t))(p)$ tel que, p est ajoutée pour spécifier à quelle place est (c*'est à dire,* $f([v_{qi}..v_{qm}], b\_FW(t))$ *est un vecteur, pour qui, chacun de ses composants correspond, en arrière , à une place de sortie*).

2.3.1 Définition

Soit M un marquage, une transition t est sensibilisée en arrière dans M si et seulement si : $\forall p \in t^{\bullet} : E < b > \leqslant M(p)$ *et* $\nexists t' \succ t$ tel que t' est sensibilisée en arrière et l'instanciation b est définie dans $b\_FW(t)$. Supposant qu'une transition t est sensibilisée en arrière, le tirage de t en arrière rend le processus d'exécution du modèle en arrière, où il enlève les jetons colorés $c'_{i1\leq j\leq m}$ }à partir des places de sortie de t et ajoute d'autres $c'_{j1\leq j\leq m}$ chacun d'eux à

son propre place qui appartient aux entrées de t. Cet ensemble de jetons colorés peut être facilement défini par la matrice de transition en arrière de t . De la même manière que celle en avant, on peut calculer le nouveau marquage μ' après avoir tiré la transition t de μ.

En outre, μ peut conduire à un marquage incohérent *(définition* 2.1.8). Un marquage μ est incohérent pour le cas où il existe une transition $fork$ t dans laquelle ses places de sortie ont des marquages différents autre que l'ensemble vide.

2.3.2 Définition

Étant donné une $CCPN$ marqué (N, μ), soit t une transition $fork$ tel que ${}^\bullet t = p$ et $t^\bullet = p_1, \ldots, p_m$, t est forcé en arrière à μ si et seulement si :

— t n'est pas sensibilisée en arrière à μ.

— $\exists p_i (1 \leq i \leq m) \mid \mu(p_i) \neq \emptyset$.

— μ n'est pas incohérent.

— $\nexists t' \succ t$ Où t' est sensibilisée en arrière ou forcée à μ.

Si t est forcée à μ, alors $\forall p$, $p' \in t^\bullet$ tel que $\mu(p) = \emptyset$ et $\mu(p') \neq \emptyset \longrightarrow \mu(p) = \mu(p')$.

Résoudre un problème de diagnostic donné par DP consiste à construire un graphe d'accessibilité en arrière. Nous commençons une telle construction à partir d'un sous-marquage μde μ^{OBS} tel que $\forall (p, c) \in M^+ : \mu(p) = c$, tandis que les autres sont vides. Les nœuds terminaux du graphe peuvent être des marquages initiaux dans un ensemble μ^{ini} des marquages incohérents ou des marquages conduisant à une incohérence. Les arcs du graphe sont étiqueté

exemple

Afin de montrer comment les diagnostics sont calculés, nous considérons l'exemple représenté dans la figure 2-1 avec une observation donnée par le marquage μ^{OBS}où $\mu^{OBS}(m_1) = c$ et $\mu^{OBS}(m_2) = a$ (*Pour plus de simplicité, nous utilisons la notation des ensembles dans laquelle les éléments sont des couples des places avec ses marques et donc* $\mu^{OBS} = (m_1, c), (m_2, a)$). Une classification de manifestations observées peut être dans la quelle $M^+ = \mu^{OBS}$. La figure 2-2 montre le graphe d'accessibilité en arrière correspondant au cas où $\mu = \mu^{OBS}$. Les Arcs du graphe sont étiquetés par des étapes *(l'ensemble des transitions sensibilisées en arrière*). Les transitions encadrées représentent les transitions forcées. Notez

que t_1 est forcée lorsque la place c_1 devient marquée avec une couleur à (*qui est présente dans* $b\_FW(t_1)$). De plus, il existe un chemin dans le graphe d'accessibilité en arrière dont le nœud terminal est un marquage incohérent où deux marquages différents sont dans la place s_1 en même temps.

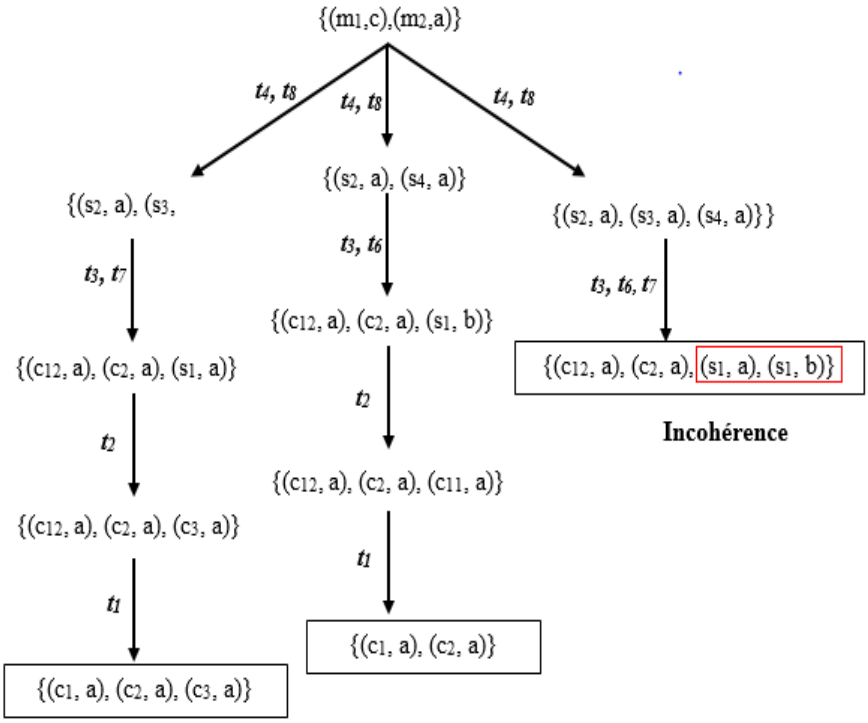

FIGURE 2-2 – *Exemple du graphe d'accessibilité en arrière.*

Les solutions que nous obtenons sont $\mu_1 = (c_1, a), (c_2, a)$ et $\mu_2 = (c_1, a), (c_2, a), (c_3, a)$.

On remarque que$\mu_1 \subset \mu_2$, donc $\mu^{ini} = \mu_1$ (*une solution à un problème de diagnostic doit être minimale*). Il est important de garder à l'esprit que μ^{ini} est un ensemble de solutions candidates à DP.

Une solution de ce problème est un marquage initial obtenu par l'analyse d'accessibilité en arrière ne conduisant à aucune combinaison dans M^-. Afin de définir à partir d'un ensemble de marquages initiaux les solutions de DP, nous suivons le principe« générer puis tester ». Pour le faire, pour chaque marquage μ de μ^{ini}, nous construisons le graphe d'accessibilité en avant. Les marquages initiaux qui ne conduisent pas à aucun élément de M^- sont choisis comme solutions. Pour le cas que nous avons μ_1 est considéré comme une solution au problème de diagnostic donné.

2.4 Définition logique

Selon la définition logique d'un problème diagnostic et d'une solution à un tel problème donnée dans le cas centralisé, on peut étendre ces définitions à un système constitué d'un système composé d'ensemble de n sous-systèmes. Dans cette vue, un DP initial est divisé en un ensemble de n sous-problèmes :

$$DP = \bigcup_{i=1}^{n} DP_i.$$

Chacun de ces DP_i correspond à un sous-système particulier et peut être vu comme le problème initial de diagnostic. Néanmoins, chaque sous-système n'est pas entièrement indépendant des autres et il interagit avec d'autres parties du système par les places communes. Par conséquent, chaque sous-problème devrait inclure dans sa définition les places communes reliant le sous-système correspondant avec ses voisins. $DP_i = (BM_i, Ctx_i, In_i, Out_i, < \Psi_i^+, \Psi_i^- >)$, Où :

- BM_i : représente le modèle comportemental du sous-système à diagnostiquer S_i.
- Ctx_i : est l'ensemble de causes de fautes possibles provenant depuis les composants de S_i.
- In_i et Out_icorrespondent aux places communes qui sont classifiées respectivement en places d'entrées deS_i qui sont déterminées à partir d'autres sous-systèmes S_j et places de sorties de S_i vers S_j.
- $< \Psi_i^+, \Psi_i^- >$: représentent l'observation locale définie avec$S_i$et leurs significations sont les mêmes que pour le cas centralisé [6].

Selon cette vue, le système de diagnostic peut être considéré comme un ensemble de diagnostiqueurs multiples dont chacun est responsable de résoudre un problème de diagnostic local DP_i et devrait communiquer avec les diagnostiqueurs voisins pour éliminer les solutions locales qui ne sont pas conformes aux résultats obtenus par ses voisins. En conséquence, In_i peuvent être vus comme des places initiales quand BM_i modélise le comportement causal de S_i, puisque leurs causes appartiennent à d'autres sous-systèmes, et donc elles ne sont pas expliquées localement. De la même façon, les éléments de Out_i peuvent être considérés comme des manifestations à expliquer même s'ils sont inobservables, parce qu'ils correspondent aux

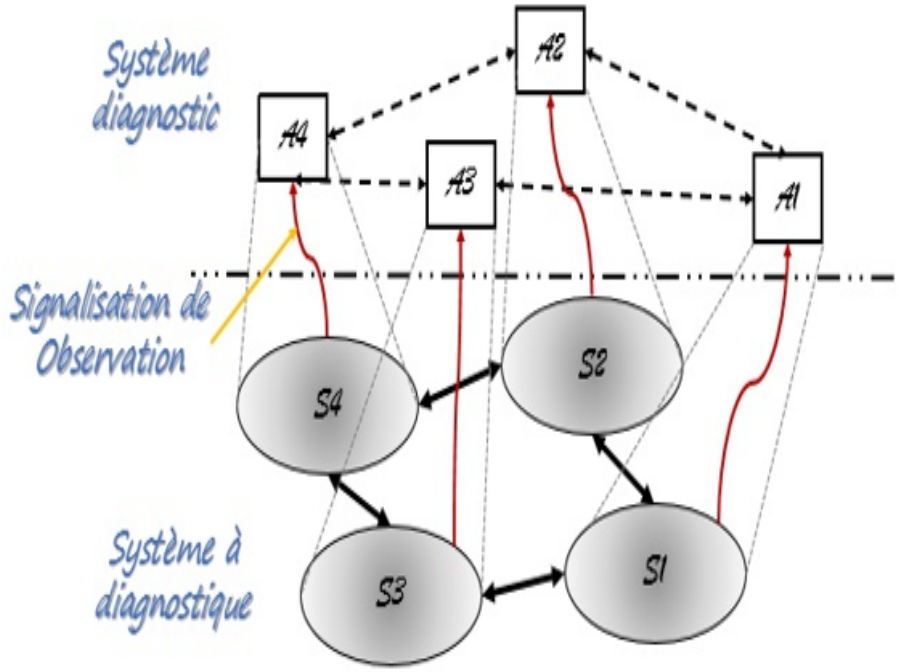

FIGURE 2-3 – *Architecture de diagnostic des systèmes distribués.*

places qui représentent des causes d'autres effets qui ne sont pas modélisés dans BM_i. En effet, les conséquences des éléments de Out_i appartiennent aux modèles voisins [3].

Selon cette vue, chaque diagnostiqueur A_i dans le système de diagnostic doit faire un diagnostic pour DP_i. Ceci permet de le voir comme un système de diagnostic centralisé si les valeurs de In_i et Out_i du sous-système $S_i$sont connues. Ainsi, A_i commence par calculer le diagnostic local $\Delta_i \subseteq CTX_i \cup In_i$ de l'observation locale $OBS_i =< \Psi_i^+, \Psi_i^- >$, et communique avec les diagnostiqueurs voisins selon les valeurs obtenues des éléments de In_i. En effet, les diagnostics vont être donnés en terme des CTX_i et les parties relatives aux In_i permettent de faire une telle communication. Cela est dû au fait que les éléments de In_ireprésentent les entrées de S_i qui correspondent aux sorties des sous-systèmes voisinsS_j.

En termes logiques, un diagnostic local de DP_i peut être considéré comme un ensemble de suppositions de fautes locales $\Delta_i \subseteq CTX_i$ tel que :

$$\forall m \in \Psi_i^+ : BM_i \cup In_i \cup \Delta_i \vdash m$$

$$\forall n \in \Psi_i^- : BM_i \cup In_i \cup \Delta_i \nvdash n$$

Une fois les Δ_i sont obtenus, ils seront utilisés pour déduire les instances des places correspondant aux sorties deS_i. De telles instances doivent être comparées afin de vérifier la cohérence avec les valeurs demandées par les diagnostiqueurs voisins comme leurs valeurs d'entrées. En d'autres termes, puisque Out_i peuvent être vus similaires aux manifestations observées, ils sont classifiés également en deux sous-ensembles Out_i^+ et Out_i^-, Out_i^+ correspondent aux valeurs de sortie qui sont modélisées dans BM_i et sont déduites du Δ_i ; alors que Out_i^- correspondent aux valeurs modélisées qui sont en contradiction avec les valeurs déduites [3].

$$\forall m \in Out_i^+ : BM_i \cup In_i \cup \Delta_i \vdash a$$

$$\forall n \in Out_i^- : BM_i \cup In_i \cup \Delta_i \nvdash b$$

Par conséquent, Δ_i est considéré comme un diagnostic local qui est cohérent avec les diagnostics locaux des autres diagnostiqueurs si et seulement si :

$$\forall m \in \Psi_i^+ \cup Out_i^+ : BM_i \cup In_i \cup \Delta_i \vdash m$$

$$\forall n \in \Psi_i^- \cup Out_i^- : BM_i \cup In_i \cup \Delta_i \nvdash n$$

2.5 Diagnostic des systèmes distribués par CCPNs

Afin d'utiliser les *CCPNs* pour faire un diagnostic d'un système distribués composé de n sous-systèmes, une approche naturelle consiste à représenter le comportement de chaque sous-système comme un *CCPN* local et à représenter les interconnections entre sous-systèmes par soit l'utilisation de places communes entre modèles *CCPNs* voisins, soit par l'utilisation de transitions de synchronisation *(partagées)* entre *CCPNs* voisins. En ce qui concerne le raisonnement proprement dit, une phase d'échange de messages entre modèles *CCPNs (en fait, entre diagnostiqueurs CCPNs)* est ajoutée au diagnostic local pour vérifier si les solutions obtenues localement ne sont pas contradictoires avec les solutions des voisins. Dans notre travail, on s'intéresse à l'usage des places communes entre *CCPNs*.

2.5.1 Places communes entre CCPNs

Comme dans [13] où chaque DP_i est représenté un *BPN* local avec des places communes, nous pouvons les exploitées de la même manière. La seule différence réside dans les jetons échangés entre modèles voisins. En effet, l'idée de représenter les interactions entre les différents sous-systèmes par le passage de jetons via les places communes dans les modèles *CCPN* . Selon la définition logique donnée dans la section précédente, les éléments de connexion entre les sous-systèmes ne sont pas nécessairement observables. Ainsi, quand un tel élément est observable, on veut dire que l'interaction correspondante peut être observée par le diagnostiqueur. En conséquence, le problème de diagnostic à base de *CCPN* avec places communes peut être défini comme :

$$CCPNDP = \bigcup_{i=1}^{n} CCPNDP_i$$

Chaque $CCPNDP_i$correspond à DP_iet est défini comme : 2.2.2

Remarque 1 *Puisque chaque CCPN local est par définition acyclique, nous supposerons que les interactions entre les modèles locaux sont également acycliques.*

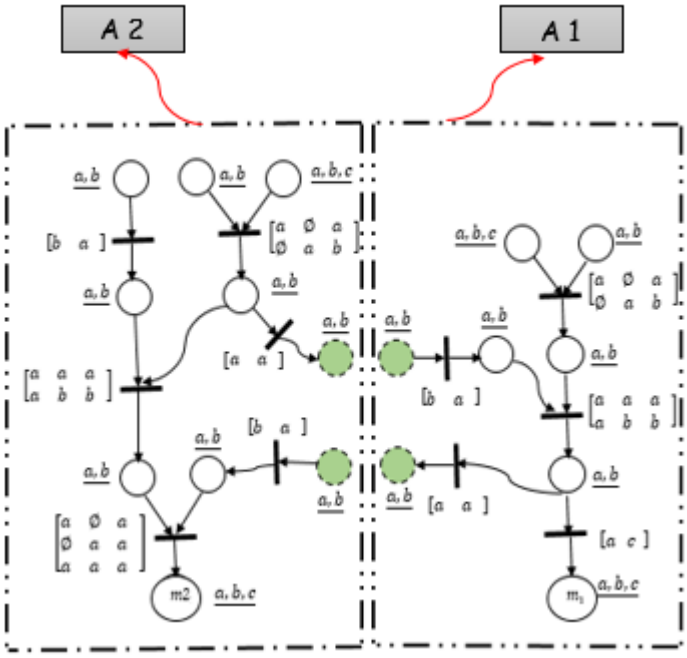

FIGURE 2-4 – *Exemple d'un CCPN distribué.*

2.5.2 Protocole de l'analyse CCPNs distribué

Afin d'assurer que les diagnostics locaux obtenus couvrent complètement les diagnostics globaux qui doivent être générés par un diagnostiqueur centralisé ayant le modèle global du système et reçoit toutes les signalisations de manifestation, chaque diagnostiqueur doit garantir que les couleur de jetons demandés par ses places d'entrée sont soutenus par les couleurs jetons offerts par les diagnostiqueurs voisins. Sinon, il y a une incohérence entre les diagnostics de ces diagnostiqueurs. Pour le faire, chaque diagnostiqueur A_i essaye de prédire pour chaque μ_i^{ini} obtenu *(pour chaque diagnostic local)*,la couleur de jetons des places communes dans le but de les comparer avec n'importe quel message reçu. si il existe une incohérents avec le voisin seront simplement envoyés un Msg d'un contradiction , parce qu'ils ne sont pas supportés par aucun diagnostic local des autres diagnostiqueurs et en conséquent, ils n'appartiennent à aucun diagnostic global. Comme résultat final, un diagnostiqueur A_i compare chaque message reçu Msg_j de A_j avec les marquages obtenus des places de communication entre ces voisins . Si n'existe pas une incohérente pour la couleur de marquage dans Msg_jet la couleur de marquage dans places de communication des voisins , dénote les places communes entre les modèles *CCPNs* de A_i et A_j .alors A_i répond au

message envoyé par A_j avec une réponse positive, ce qui signifie que le diagnostic local de A_j depuis lequel Msg_j est généré est cohérent avec les diagnostics de A_i. à l'inverse, si pour tous μ_i^{ini} il n'existe aucun marquage qui supporte Msg_j, alors A_i répondra à A_j par une réponse négative. Par conséquent, quand le diagnostiqueur A_j reçoit une réponse négative au message Msg_j d'un diagnostiqueur voisin, il éliminera son diagnostic local Δ_j depuis lequel le message Msg_j à été généré, parce qu'il n'est pas cohérent avec les diagnostics des voisins même s'il explique localement le comportement observé. En plus, il se peut que le diagnostic éliminé soit utilisé pour valider la cohérence entre les diagnostics de A_j et ceux d'un autre diagnostiqueur voisin $A_k(k \neq i)$. Comme résultat, quelques diagnostics de A_k doivent être éliminés lorsqu'ils deviennent incohérents avec les diagnostics de A_j ; et ainsi la communication entre les diagnostiqueurs sera lancée une autre fois. En conséquence, la vérification de la cohérence terminera après quelques cycles de communication quand une condition de stabilité en termes de diagnostics locaux de tous les diagnostiqueurs est satisfaite.

2.6 Conclusion

Dans ce chapitre, nous avons présenté de façon détaillée une extension des *CCPNs* pour prendre en compte les interaction entre composants d'un système distribué. Ainsi, un protocole de coopération est défini pour l'assurance globale de la cohérence entre diagnostics calculés localement. Dans le chapitre suivant, nous essayerons d'expliquer les différentes phases conduisant à implémenter un outil pour le diagnostic des systèmes distribués à base de *CCPNs* avec places communes.

Chapitre 3

Développement d'un outil de diagnostic distribué

On a présenté dans le chapitre précédent une technique de diagnostic basée sur les *CCPNs*. Notre travail consiste à développer un protocole coopératif de diagnostic des systèmes distribués à base de cette technique. Afin d'arriver à la fiabilité et la sureté 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.

3.1 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. Comme nous avons mentionné auparavant, notre objectif est la réalisation d'un outil de diagnostic des systèmes distribués basé sur les *CCPNs* assurant :

1. Une interface graphique simple en termes d'utilisation permettant la saisie du modèle sous forme d'un *CCPN*.
2. Offrir de manière simple la saisie de l'ensemble des manifestations pour que la tâche

du diagnostic s'effectue.

3. La tâche de diagnostic doit être passée par :
   - Construction d'un graphe de marquages en arrière pour déterminer les marquages initiaux μ_{ini}.
   - Une solution à un problème de diagnostic doit être minimale et qui ne conduit pas à aucun élément de M^- .
   - Échange de messages entre *CCPNs (diagnostiqueurs)* pour garantir une cohérence globale entre solutions obtenues localement.
   - Construction du graphe d'accessibilité en avant pour chaque marquage μ de μ_{ini}

3.2 Conception du système

La conception est un processus itératif à plusieurs niveaux, elle commence par une conception globale qui peut être raffinée jusqu'à aboutir à un niveau moins abstrait à partir duquel il est simple de générer le code source.

3.2.1 Conception globale

Notre système composé par un ensemble des diagnostiqueurs attachés à un ensemble de sous-systèmes qui sont connecté par un réseau de communication. On peut schématiser notre système par la figure suivante Figure 3-1 :

Dans notre mémoire, on s'intéresse à étudier le travail de chaque diagnostiqueur tel que pour chaque sous-système un diagnostiqueur est associé, le diagnostiqueur ne connaît que le modèle local du sous-système, reçoit l'observation locale et peut échanger des informations limitées avec les diagnostiqueurs voisins pour l'élimination des solutions contradictoires [3].

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 3-2, chaque sous système à réaliser prend en entrée le modèle sous forme d'un *CCPN* représentant le comportement causal du système à diagnostiquer et l'ensemble des observations sous la forme d'un marquage final

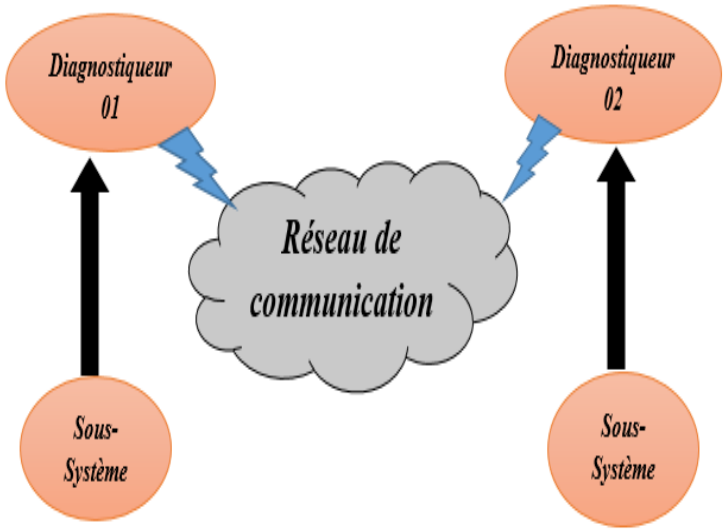

FIGURE 3-1 – *L'architecture générale du système[2]*

pour lequel le système est considéré défaillant. En se basant sur ces entrées, chaque diagnostiquer local génère un *graphe de marquages en arrière* puis calcul l'ensemble de diagnostics expliquant le mal fonctionnement du sous-système et ensuite *communique* avec ses voisins pour la vérification de la cohérence globale. [10]

On peut schématiser le fonctionnement d'un diagnostiqueur local par la figure suivante figure 3-3.

3.2.2 Conception détaillée de l'outil

La conception détaillée fournit pour chaque composant une description de la manière dont les fonctions de notre système de diagnostic sont implémentées en termes d'algorithmes et de structures de données. [10]

— ***Edition*** : le module d'édition prend en entrée le modèle du système sous forme d'un graphe *CCPN* représentant le comportement causal du système à diagnostiquer, et produit en sortie une représentation interne sous forme matricielle du modèle.

— ***Calcul*** : le module calcul construit le graphe de marquages en arrière sur lequel le diagnostic est accompli. Pour cette raison, le module calcul prend en entrée la matrice généré par le module édition et l'ensemble des observations sous la forme

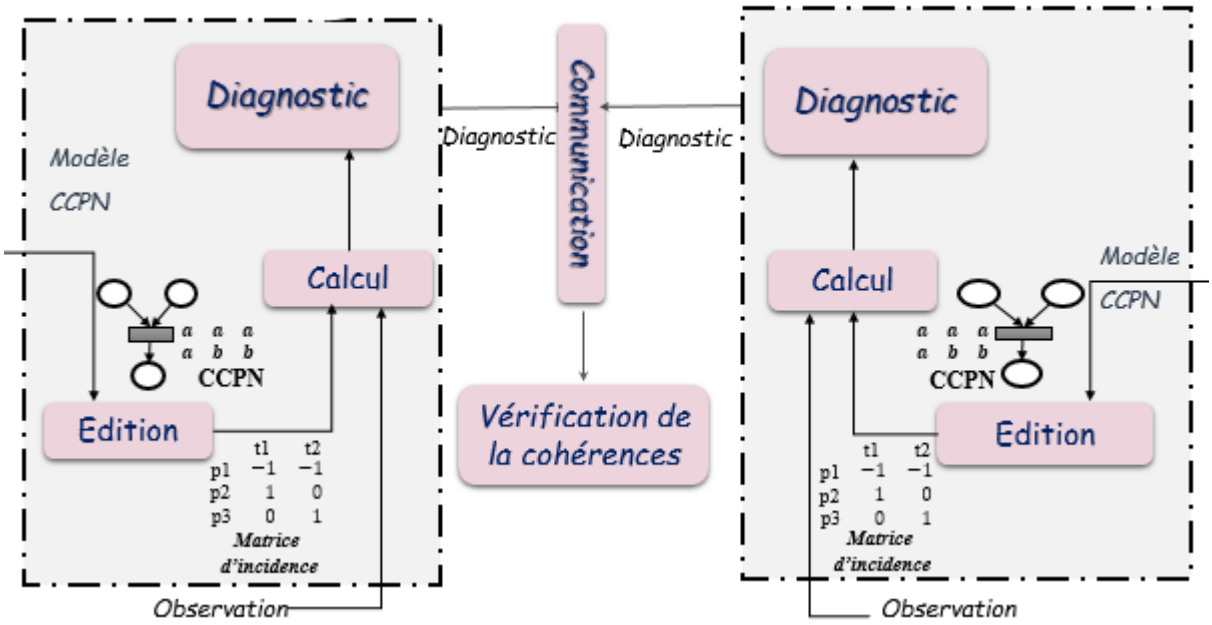

FIGURE 3-2 – L'architecture générale du système de diagnostic.

d'un marquage final pour lequel le sous-système est considéré défaillant et donne comme résultat le graphe de marquages en arrière de ce modèle.

— ***Diagnostic*** : Le module de diagnostic a pour but de déterminer l'ensemble des états initiaux menant le sous-système d'arriver à ce dysfonctionnement, en prend en entrée le graphe des marquages obtenu du module calcul. Une première étape consiste à générer l'ensemble de marquages initiaux. L'étape suivante sert à assurer la cohérence d'une solution proposée par laquelle nous construisons un graphe d'accessibilité en avant pour chaque marquage de μ^{ini}. Finalement nous sélectionnons comme solutions les marquages qui n'atteignent aucun élément de M^{-}.

— ***Coopération*** : Dans la phase de coopération, les diagnostiqueurs échangent des messages Msg contenant le marquage des places communes, D_1 Envoie des messages à D_2, ces messages contiennent le marquage de la place coopérative x, D_2 envoie aussi des messages à D_1 concernant le marquage des places coopératives y. D_1 lance un franchissement en avant et il trouve comme résultat que :$\Delta_2 = \mu_2^1, \mu_2^2, \mu_2^3$. D_2 lance un franchissement en avant et il trouve comme résultat que : $\Delta_1^{=}\mu_1^8, \mu_1^9, \mu_1^{10}, \mu_1^{15}, \mu_1^{16}$.

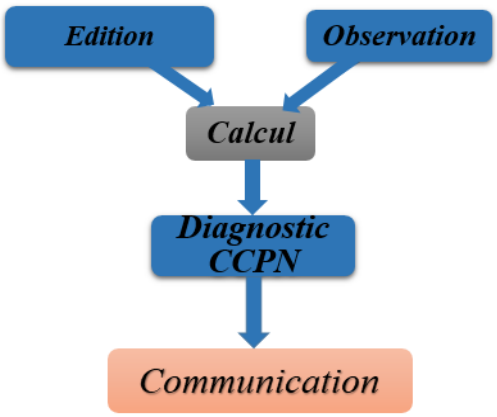

FIGURE 3-3 – *Le fonctionnement d'un diagnostiqueur* .

— **Vérification de la cohérence globale** suite à l'arrivée d'un message, le module de vérification compare le marquage des places communes demandé avec celui calculé localement. S'ils sont équax, il les considèrent cohérentes et répond positivement ; sinon il doit répondre par un message négatif.

3.2.3 Protocole de coopération

Initialement, on nommait protocole ce qui est utilisé pour communiquer sur une même couche d'abstraction entre deux machines différentes. Par extension de langage, on utilise parfois ce mot aussi aujourd'hui pour désigner les règles de communication entre deux couches sur une même machine.

Les sockets

Formellement, un socket est un point de communication bidirectionnel par lequel un processus pourra émettre ou recevoir des informations. Moins formellement, au lieu de parler de la communication entre processus, parlons de la communication entre personnes par téléphone ou par courrier. Cette communication nécessite que les personnes disposent, soit d'un poste de téléphone, soit d'une boite aux lettres. Le poste de téléphone et la boite

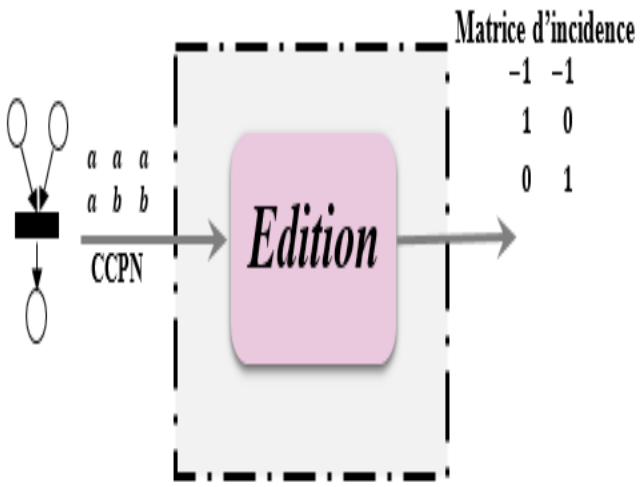

FIGURE 3-4 – *module Edition*

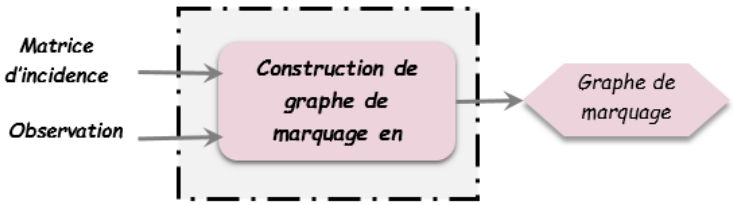

FIGURE 3-5 – *module Calcul*

aux lettres sont des « *points de communicatio*n » ayant des adresses connues du monde extérieur. Les sockets sont l'analogue de ces "*points de communications*". Les processus sont l'analogue des individus. Lorsque deux processus veulent se communiquer des données, il faut que chacun d'eux dispose d'au moins un socket. Les sockets ne sont pas le seul moyen de communication entre deux processus mais un grand nombre d'applications sont fondées sur les sockets [14].

Les sockets sont aussi associées à un protocole. Dans notre cas nous utiliserons le protocole *TCP/IP*. Les sockets servent à établir une transmission de flux de données *(octets)* entre deux machines ou applications. Une socket est un identifiant unique représentant une adresse sur le réseau. Des processus peuvent s'y connecter pour y envoyer des données ou pour en recevoir. Les processus devront adopter un protocole de communication afin d'assurer un échange de données cohérent. L'adresse de la socket est spécifiée par le nom de l'hôte sur lequel on la

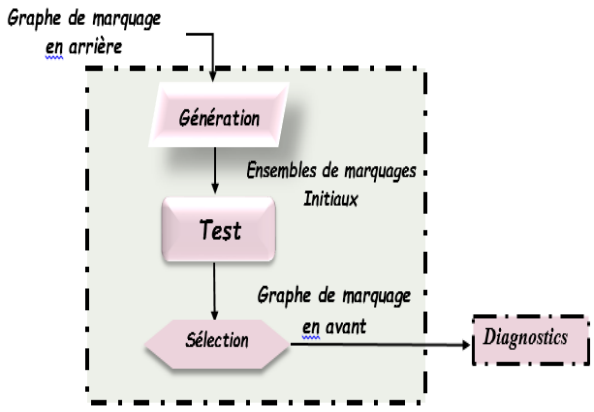

FIGURE 3-6 – Module diagnostic.

crée et le numéro de port. grâce aux sockets, vous pouvez faire communiquer des langages totalement différents pour peu qu'ils adoptent le même protocole de communication.

3.3 Implémentation

Comme nous avons montré dans la conception générale, les structures de données utilisées est qui doivent représenter les *CCPNs*, les observations et le graphe de marquages sont comme suivant :

Un Causal Coloured Petri net *(CCPNs)* est représenté par une matrice :

Classe matrice :

— Places : c'est l'ensemble des places.

— Transitions : c'est l'ensemble des transitions.

— Vale : c'est la valeur de la matrice d'incidence.

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 chaque jeton est une chaine de

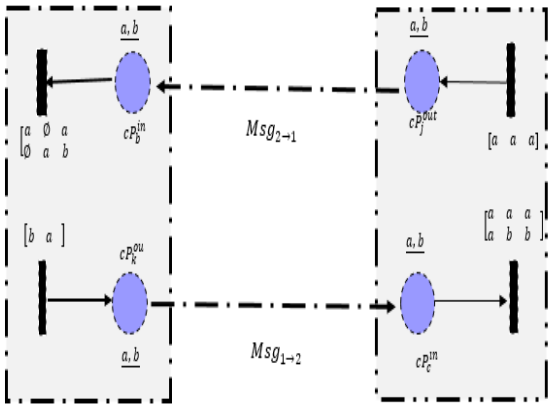

FIGURE 3-7 – *Module de coopération.*

caractères.

Pour les transitions, on a :

— Classe transition : Id_trans c'est l'identificateur de la transition.

— Nom_trans : représente le nom de la transition.

— Mat : c'est une matrice $N \times M$ décrivant les combinaisons possibles de couleurs de jetons d'entrée et de sortie où n est le nombre de façons dont t peut tirer et m définit le nombre de places liées à t.

Chaque nœud (marquage) d'un graphe de marquages contient une ou plusieurs places :

Classe marquage :

Id_marquage c'est l'identificateur de marquage.

Nom_place représente la place marquée.

Jeton représente le marquage de la place .

Les observations sont représentées par un marquage final μ^{OBS} où les places marquées sont celles appartenant aux manifestations.

Les principaux algorithmes

Dans cette section,on va fournir les algorithmes que nous jugeons importants Dans le module calcul on commence par l'algorithme 1qui sert à calculer les places post de chaque transition.

Algorithm 3.1 Calcul la post place

entrée : matrice incidence, transition t.

sortie : vecteur des places.

début

var $p \leftarrow \{\}$;

pour chaque j allant de 1 à $m-1$ **faire**

si $incidence[0][j].t = t$ **alors**

pour chaque i allant de 1 à $n-1$ **faire**

si $incidence[0][j].Vale = 1$ **alors**

$p \leftarrow incidence[i][j].t$;

fin si

fin pour

fin si

fin pour

return p ;

fin

Algorithm 3.2. test si toutes les places post de la transition sont marquées

entrée : $EnsP$, liste de marquage L.

sortie : vecteur des places.

début

var $k \leftarrow 0$;

pour $p \in EnsP$ **faire**

pour $M \in L$ **faire**

si $p.no = m.nom$ **alors**

$p \leftarrow incidence[i][j].t$;

$k \leftarrow k + 1$;

si ($k = taille$ de L) **alors**

return $vrai$;

fin si

sinon

return $vrai$;

fin si

fin pour

fin pour

fin

où algorithme 3.2 (test) assure que toutes les places post de la transition sont marquées.

Algorithm 3.2. enabled transitions

entrée : matrice $incidence[n][m]$, liste de marquages L ;

sortie : vecteur des transitions sensibilisées ;

début

$t \leftarrow \emptyset$; $p \leftarrow \emptyset$;

pour $l \in L$ **faire**

pour i allant de 1 à $n-1$ **faire**

si $l.nom = incidence[i][0].nomplace$ **alors**

pour j allant de 1 à $m-1$ **faire**

si $incidence[i][j].vale = 1$ **alors**

$p \leftarrow calculpostplace(incidence, incidence[i][j].t)$;

$b \leftarrow faux$;

si $b = vrai$ **alors**

pour k allant de 1 à $m-1$ **faire**

si $l.jeton == incidence[i][j].t.mat[k][0]$ **alors**

$t \leftarrow incidence[i][j].t$;

fin si

fin pour

fin si

fin si

fin pour

fin si

fin pour

return t ;

fin pour

fin

Cet algorithme return les transitions franchissables (enabled)

3.3.1 Environnement et langage de programmation utilisé

Après avoir finalisé l'étape de conception, nous allons présenter dans la phase qui suit, l'implémentation de notre outil de diagnostic *CPPNs* des systèmes distribués .

Cette activité consiste à réaliser le résultat de la conception détaillée et le transformer en un programme. Concernant notre travail, on a choisi comme langage de programmation le langage java et deux Environnement différente sont *Eclipse mars* et *IntelliJ IDEA*.

Java

Java est un langage de programmation orienté objet créé en 1991 par « SUN Microsystem». L'objectif initial était la programmation de petits appareils comme des télécommandes, la programmation d'applications Web. Il est devenu actuellement l'un des langages de programmation généraliste les plus utilisés. La caractéristique importante de Java est qu'il est un langage purement orienté objet. En effet, l'entité de base de tout code Java est la classe : c'est-à-dire qu'en Java, tout se trouve dans une classe, il ne peut y avoir de déclarations ou de code en dehors du corps d'une classe. La syntaxe du langage Java est très proche du langage C++ (et par conséquent du langage C) [15].

Qu'est-ce qu'Eclipse ?

Eclipse est un IDE, Integrated Development Environment (*EDI environnement de développement intégré en français*), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation. Au fur et à mesure que vous programmez, eclipse compile automatiquement le code que vous écrivez, en soulignant en rouge ou jaune les problème qu'il décèle. Il souligne en rouge les parties du programme qui ne compilent pas, et en jaune les parties qui compilent mais peuvent éventuellement poser problème (*on dit qu'eclipse lève un avertissement, ou warning en anglais*). Pendant l'écriture du code, cela peut sembler un peu déroutant au début, puisque tant que la ligne de code n'est pas terminée (*en gros jusqu'au point-virgule*), eclipse indique une erreur dans le code. Il est déconseillé de continuer d'écrire le programme quand il

contient des erreurs, car eclipse est dans ce cas moins performant pour vous aider à écrire le programme.[16]

Eclipse est un EDI qui permet le développement d'applications dans plusieurs langues. En fait, il peut être appelé un environnement de développement logiciel complet composé de l'IDE et du système de plug-in. C'est gratuit, et le logiciel open source publié sous licence publique Eclipse. Cependant, avec l'utilisation de plug-ins appropriés, il peut être utilisé pour développer des applications dans de nombreux autres langages tels que C, C ++, Perl, PHP, Python, Ruby, etc. L'IDE Eclipse s'appelle Eclipse ADT, Eclipse CDT, Eclipse JDT et Eclipse PDT, lorsqu'ils sont utilisés avec Ada, C / C ++, Java et PHP, respectivement. Il s'agit d'un IDE multi-plateforme.

Parmi ses concurrents, on trouve NetBeans`http://www.netbeans.org`, gratuit et développé par Sun, IDEA de JetBrain `http://www.jetbrains.com` qui est payant Eclipse est disponible sur le site `http://www.eclipse.org` pour linux, BSD, Windows ou MacOS. Normalement, la page `http://www.eclipse.org/downloads` choisit automatiquement la bonne version.

FIGURE 3-8 – Eclipse Mars

Pourquoi IntelliJ IDEA

Intellij IDEA est un IDE Java développé par JetBrains. La première version d'IntelliJ est sortie en 2001. A l'époque, c'était le seul IDE supportant la navigation et le refactoring de code avancés. C'est un produit commercial, où un essai gratuit de 30 jours (avec toutes les fonctionnalités) est disponible pour toutes les plateformes. Plus récemment, une édition open source a été mise à disposition.Il offre un support pour dessiner des diagrammes de classes UML, la modélisation visuelle dans Hibernate, Spring , l'analyse des dépendances et Maven. Les applications dans de nombreuses langues telles que Java, JavaScript, HTML, Python, Ruby, PHP et bien d'autres peuvent être développées en utilisant IntelliJ. IntelliJ prend en charge une large gamme de frameworks et de technologies tels que JSP, JSF, EJB, Ajax, GWT, Struts, Spring, Hibernate et OSGi. En outre, divers serveurs d'applications tels que GlassFish, JBoss, Tomcat et WebSphere sont pris en charge par IntelliJ. L'intégration facile avec CVS, Subversion, Ant, Maven et JUnit est rendue possible par IntelliJ. [17]

IntelliJ IDEA est disponible sur le site `http://www.jetbrains.com` pour linux, BSD, Windows ou MacOS. Normalement, la page `https://www.jetbrains.com/idea/download/` choisit automatiquement la bonne version.

FIGURE 3-9 – *IntelliJ IDEA*

Différence entre Intellij et Eclipse

Le marché IDE *(environnement de développement intégré)* est l'un des plus intensément concurrencés dans le domaine des outils de programmation. IntelliJ IDEA et Eclipse sont deux des quatre principaux concurrents dans ce domaine *(NetBeans et Oracle JDeveloper sont les deux autres)*. Eclipse est un logiciel gratuit et open source, alors que IntelliJ est un produit commercial.[17]

FIGURE 3-10 – *Différence entre IntelliJ et Eclipse*

— **Quelle est la différence entre Intellij et Eclipse ?**

Bien que IntelliJ et Eclipse soient deux des IDE Java les plus populaires à l'heure actuelle, ils ont leurs différences. Tout d'abord, Eclipse est gratuit et entièrement open source, alors qu'IntelliJ est un produit commercial. Le support pour Maven est meilleur dans IntelliJ. IntelliJ IDEA est livré avec un générateur d'interface graphique intégré pour Swing, mais vous devez utiliser un plug-in distinct dans Eclipse pour le même objectif. En fait, la communauté Java considère le générateur d'interface graphique d'IntelliJ comme le meilleur concepteur d'interface graphique pour le moment. En termes de support XML, IntelliJ offre la meilleure option.Il a un éditeur XML intégré avec des fonctionnalités sophistiquées telles que l'achèvement et la validation de code (qui n'est pas présent dans Eclipse). Cependant, le système de plug-in et la grande quantité de plug-ins extensibles disponibles chez de nombreuses personnes font d'Eclipse un produit très populaire dans l'industrie. Malgré les différences de fonctionnalités, les opinions générales au sein de la communauté Java sur la performance de ces deux IDE sont assez similaires.

3.4 Conclusion

L'objectif visé par ce chapitre est de spécifier les différents composants de notre système. Nous avons présenté le développement d'un outil assurant le diagnostic des systèmes distribués à base de *CCPNs*, en suivant l'esprit du génie logiciel, on a commencé par une définition des besoins, suivie par la phase de la conception globale où nous avons défini les différents composants de notre outil, suivie par la conception détaillée. Finalement nous avons présenté les structures de données et les algorithmes utilisés dans notre implémentation et l'environnement Java.

Conclusion générale

Les systèmes artificiels d'aujourd'hui deviennent de plus en plus complexes. En conséquence, leurs analyses deviennent de plus en plus difficiles, particulièrement pour ceux où la société attend une fiabilité très élevée. Généralement, un système peut être vu comme un ensemble de composants (sous-système) qui interagissent l'un avec l'autre pour atteindre les buts pour lesquels il a été désigné. Donc, la performance du système entier dépend de la performance de chacun de ses composants ainsi que de la qualité de l'interaction entre eux.

L'essentiel du travail réalisé dans ce mémoire est d'étendre un outil à base de réseaux de Petri colorés appélé CCPN qu'à été proposé dans [1] pour le diagnostic centralisé à base de modèles causaux. Une extension au cas distribué tel que le modèle du système en question est vu comme un ensemble de modèle CCPNs locaux avec des places communes utilisées pour capturer les interactions possibles entre sous-systèmes. Le diagnostic local est réalisé à travers une analyse en arrière exploitant le graphe de marquages du modèle CCPNs associé à partir d'un marquage final correspondant à l'observation reçue localement.

Une fois les diagnostics locaux ont été calculés, les diagnostiqueurs exécutent un protocole de coopération pour écarter les diagnostics qui sont en contradiction les uns avec les autres. Ceci est accompli via un échange de messages codifiant le marquage des places communes. En particulier, lorsu'un diagnostiqueur reçoit un message depuis l'un de ses voisins, il construira un graphe d'accessibilité en avant depuis chacun de ses diagnoistics locaux. Deux réponses sont possibles : positive, signifiant que le diagnostic à partir duquel le message à été généré est cohérent avec ceux du diagnostiqueur en question ; ou une réponse négative signifiant l'inverse. Par conséquence, lors de la réception d'une réponse négative, le diagnostic en question doit être écarté.

Enfin, le développement d'un outil implémentant notre protocole est décrit.

Bibliographie

[1] S. Mancer and H. Bennoui, "Coloured petri nets based diagnosis on causal models," in *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'17), co-located with the38th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2017 and the 17th International Conference on Application of Concurrency to System Design ACSD 2017, Zaragoza, Spain, June 25-30, 2017.*, 2017, pp. 123–136. [Online]. Available : http://ceur-ws.org/Vol-1846/paper8.pdf

[2] C. S. Housine, "Diagnostic probabiliste des systèmes distribués à base de ppn ouverts," Master's thesis, université mohamed khider biskra, 2017.

[3] B. Safaa, "Analyse de BPNs en interaction pour le diagnostic basé-modèle des systèmes distribués," Master's thesis, Université Mohamed Khider Biskra, 2012.

[4] J. de Kleer and J. Kurien, "Fundamentals of model-based diagnosis," *IFAC Proceedings Volumes*, vol. 36, no. 5, pp. 25 – 36, 2003, 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 2003, Washington DC, 9-11 June 1997. [Online]. Available : http://www.sciencedirect.com/science/article/pii/S1474667017364674

[5] M. Combacau,P. Berrut,F. Charbonnaud,and A. Khatab, "Refexions sur la terminologie : Surveillance -supervision," *Groupement pour la recherche en Produtique*, vol. -1, 2000, to appear.

[6] B. Hammdi, "Distributed causal model-based diagnosis an approach by interacting petri nets," Master's thesis, université mohammed khider, 2012. [Online]. Available : http://dspace.univ-biskra.dz:8080/jspui/handle/123456789/4655

[7] K. Labadi, "Contribution to modelling and performances analysis of logistic systems by using a new stochastic Petri nets model," Theses, Université de Technologie de Troyes, Nov. 2005, diplôme de doctorat délivré conjoitement avec l'Université de Technologie de Compiègne (UTC). [Online]. Available : https://tel.archives-ouvertes.fr/tel-00389432

[8] J. Comet, "Les réseaux de petri pour la simulation de systèmes biologiques," Ph.D. dissertation, Université de Nice-Sophia-Antipolis Ecole Polytech, april 25 ,2014. [Online]. Available : http://www.i3s.unice.fr/~comet/SUPPORTS/Nice-EPU-GB4-TechSim/coursRdPMetabolique.pdf

[9] J. Peterson, *Petri net theory and the modeling of systems*, ser. Foundations of Philosophy Series. Prentice-Hall, 1981. [Online]. Available : https://books.google.dz/books?id=xNtQAAAAMAAJ

[10] M. Raouf, "Une approche réseaux de petri colorés pour le diagnostic à base de modèles causaux," Master's thesis, université mohammed khider, 2017.

[11] Wikipédia, "Réseau de petri — wikipédia, l'encyclopédie libre," 2018, [En ligne ; Page disponible le 7-Avril-2018]. [Online]. Available : http://fr.wikipedia.org/w/index.php?title=R%C3%A9seau_de_Petri&oldid=146202353

[12] K. Jensen, *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use - Volume 1*, ser. EATCS Monographs on Theoretical Computer Science. Springer, 1992. [Online]. Available : https://doi.org/10.1007/978-3-662-06289-0

[13] H. Bennoui, " Interacting behavioral Petri nets analysis for distributed causal model-based diagnosis," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 2, pp. 155–181, Mar 2014. [Online]. Available : https://doi.org/10.1007/s10458-013-9221-5

[14] A. Habibi, "Travaux Pratiques Réseaux "Les Socket"," 2018. [Online]. Available : https://www.google.dz/search?q=A.+Habibi+Travaux+Pratiques+Reseaux+%2C+%22Les+sockets&oq=A.+Habibi+Travaux+Pratiques+Reseaux+%2C+%22Les+sockets&aqs=chrome..69i57.3671j0j7&sourceid=chrome&ie=UTF-8

[15] C. Delannoy, *Programmer en Java.* Eyrolles, 2007. [Online]. Available : https://books.google.dz/books?id=P-t0HwAACAAJ

[16] Julien.Cervelle, "Eclipce," 2018, [En ligne ; Page disponible le 30-Avril-2018]. [Online]. Available : http://www.enseignement.polytechnique.fr/informatique/profs/Julien.Cervelle/eclipse

[17] esdifferent, "Difference between intellij and eclipse," 2018, [En ligne ; Page disponible le 7-Mai-2018]. [Online]. Available : https://fr.esdifferent.com/difference-between-intellij-and-eclipse