République Algérienne Démocratique Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université d'Ibn Khaldoun – Tiaret

Faculté des Mathématiques et de l'Informatique

Département Informatique

Thème

Conception et réalisation d'un éditeur graphique de réseau de pétri

Pour l'obtention du diplôme de Master

Spécialité : Génie Informatique

Option : système d'information et technologie de web

Rédigé par : BENAMAR Khaled

AYACHI Chemseddine

Dirigé par : Mr ALEM Abdelkader

Année universitaire: 2016-2017

Dédicaces

Je dédie ce modeste travail, A mes très chers parents, mes modèles et guides dans la vie, qui m'ont fourni les moyens, le support et la sagesse dans tous les moments. Que dieu les protège.

A mes deux chères enfants Abdellah, Farouk et ma femme merci pour la joie que vous apportez à ma vie.

A mes deux chers frères Mohamed et Ahmed, Je leur souhaite beaucoup de réussite.

A tous mes amis, particulièrement :

Mustapha, kheireddine et ziane.

Khaled

Dédicaces

Je dédie ce modeste travail, A mes très chers parents, mes modèles et guides dans la vie, qui m'ont fourni les moyens, le support et la sagesse dans tous les moments. Que dieu les protège.

A mon frère et ma sœur merci pour la joie que vous apportez à ma vie.

A tous mes amis

Chemseddine

Remerciements

Avant tout, je tiens à remercier ALLAH le tout Puissant et Miséricordieux qui m'a donné la force et la patience d'accomplir ce mémoire.

Aussi je tiens à remercier mon encadreur Monsieur Alem Abdelkader , pour ses prises en charge et ses bons conseils, qui ont abouti à l'élaboration de ce travail.

Je tiens à remercier les membres du jury pour leurs attentions et intérêts portés envers mon travail. Merci de m'avoir honorée de votre présence.

Enfin, mes remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail particulièrement. Qu'ils trouvent tous ici l'expression de ma gratitude et parfaite considération.

Abstract

La modélisation et la simulation des systèmes jouent un rôle crucial dans divers domaine comme la médecine, génie mécanique, informatique etc. Divers formalisme dédiée pour effectuer une telle simulation, nous pouvons citer langage B, langage Z, UML et réseau de pétri.

Derrière chaque formalisme les industries et les organismes académique développe des outils de simulation graphique pour facilite la modélisation graphique, l'analyse, la vérification et la génération de code. Ces outils de simulation sont souvent présenté comme une entité obscure avec sans détail d'implémentation et mise en œuvre comme TINA, Project, Canva etc...,

Dans ce cas il est difficile d'éditer, étudier et modifier le code source de ces outils commerciaux.

Dans ce travail nous somme intéressées de l'outil de modélisation RdP qui' est puisant dans le domaine de la modélisation et de la vérification des systèmes

Notre objectif dans ce mémoire est de proposer un outil open source et académique pour le l'aide de modélisation graphique RdP. Cet outil sera utiliser et enrichie par les autre étudiants a chaque année.

Pour atteindre cet objectif nous avons analysé le domaine, modéliser le domaine et proposer un outil prototype.

Mot clés : *modélisation simulation ; Langage modélisation, RdP, éditer graphique*

Lexique

RdP : Réseau de pétri

UML : Unified Modeling Language

Java : langage de programmation informatique orienté objet

XML : Extensible Markup Language

XMI: XML Metadata Interchange

JDK : Java Development Kit

IHM : Interface homme-machine

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

Contexte et Motivation :

Depuis longtemps, les développeurs des applications ont utilisé les éditeurs de code pour les aider dans leur tâche de programmation, puisque ces derniers offrent des fonctionnalités comme la coloration syntaxique, la complétion automatique, la détection d'erreur etc. La demande de plus en plus exigeante des programmeurs_en éditeurs adaptés à leurs besoins a permis des progrès incessants dans ce domaine.

Aujourd'hui on trouve des éditeurs de plus en plus spécialisés dans des tâches précises comme la création des interfaces graphiques, des sites Web, des documents structurés, etc.

Ces éditeurs ont rendu les programmeurs et/ou développeurs dépendants d'eux, étant Donnés tous les avantages qu'ils offrent. Alors que d'un autre point de vue, la généralisation de l'utilisation des éditeurs peut aussi être vue comme un handicap, surtout lorsqu' un développeur commence à travailler avec un nouveau langage qui n'est pas encore supporté par aucun éditeur.

On peut également citer une simulation des fonctionnements des réseaux de pétri avec ces éditeurs graphiques

D'un point de vue organisationnel, pour bien mener notre travail, nous avons suivi un plan compose de trois chapitres :
Le premier chapitre permet de voir des notions générales sur les réseaux pétri et éditeur graphique.
Dans le deuxième chapitre nous présentons notre conception d'outil prototype.

Dans le troisième chapitre nous montrons nos choix des outils et langages utilisées en développement ainsi que notre implémentation pour réaliser l'application

Problématique :

Dans notre projet, nous avons identifié les problèmes suivant : **(i)** Difficulté d'établir un RdP, (ii) Absence d'un outil communication des graphes RDP ente étudiant/ enseignant.

- Problème de vérification et debugger des erreurs de conception

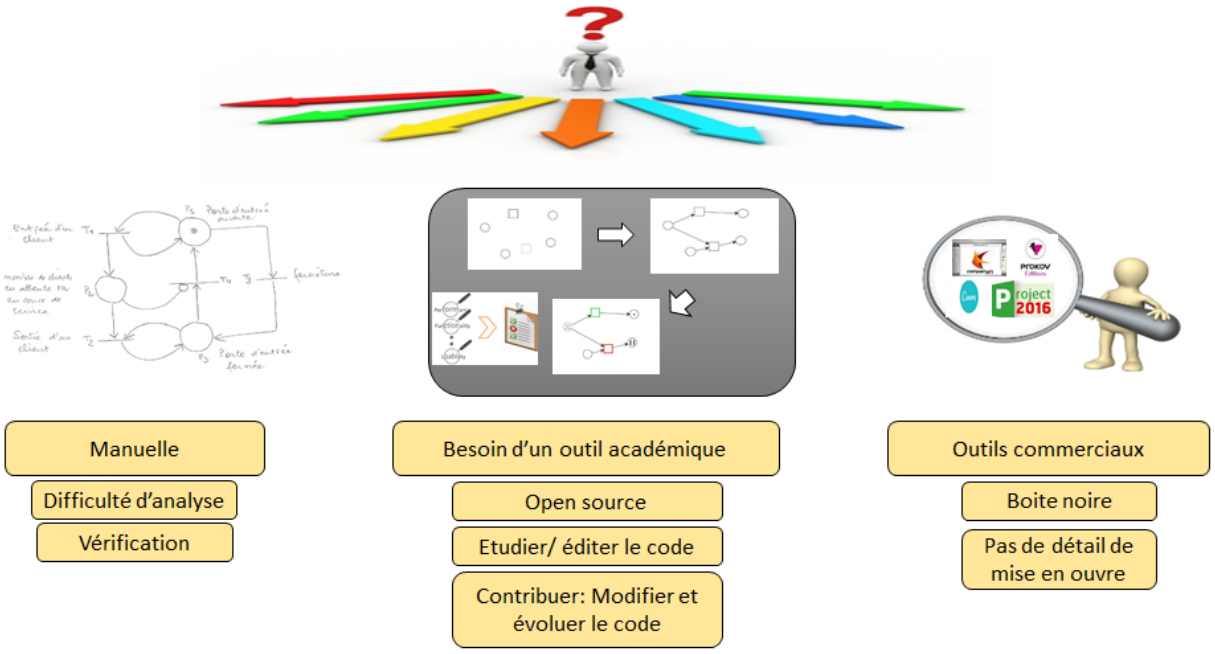

Figure 1: Problematique

Objectif :

- Concevoir un système d'éditeur graphique pour les RDP
- Implémenter une solution logicielle pour notre approche proposée
- Valider notre approche prenant on considération des *feedback* des étudiants de master de l'université UIK.

Organisation de notre mémoire :

Pour mieux aider les lecteurs de notre mémoire de fin d'étude à le lire et à le comprendre facilement, nous avons jugé nécessaire de le présenter sous forme de quatre chapitres résumés comme suit :

Chapitre 1 : Synthèse bibliographique

Il traite tout d'abord l'éditeur graphique et les réseaux de pétri de façon générale en évoquant les définitions et les concepts de bases relatifs à ce sujet .

Chapitre 2 : Analyse et conception

Dans ce chapitre nous allons nous baser sur l'aspect fonctionnel de notre système puis nous allons concevoir un modèle sur le quel notre système sera construit dans la phase de l'implémentation.

Chapitre 3 : Réalisation

A propos de ce dernier chapitre, il va décrire les différentes techniques et les différents outils utilisés lors de la conception de notre solution. Il décrira également la manière dont fonctionne le nouveau système. Enfin, une conclusion vient récapituler le travail effectué et évaluer sa convenance.

Quant à l'évolution et l'amélioration d'un tel projet, d'intéressantes perspectives seront suggérées à la fin du document. En outre, une annexe vient compléter les notions présentées dans ce rapport.

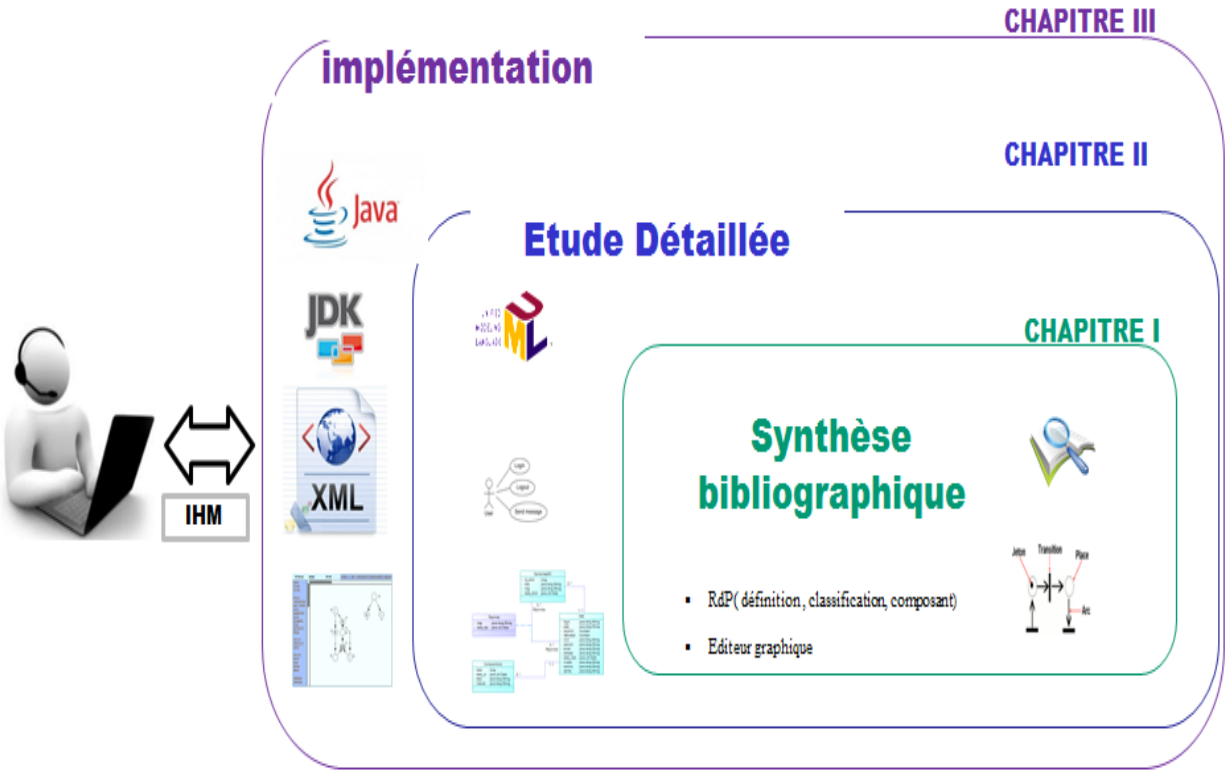

Figure 2 : Organisation de notre mémoire

Chapitre I

Généralité sur les réseaux de pétri et les logiciels graphique

Introduction

Les Réseau de Pétri représentent un outil mathématique puisant dans le domaine de la modélisation et de la vérification des systèmes. En plus de leur force d'analyse, ils offrent une représentation graphique simple qui aide à la modélisation des systèmes complexes.

Donc Le modèle des réseaux de Pétri est un outil graphique de modélisation et d'analyse des systèmes parfaitement adapté à l'étude des structures de contrôle. Il permet notamment de maîtriser et d'assurer la sûreté de fonctionnement de logiciels complexes (aéronautique, transports, industrie...).

Le formalisme formel des Réseau de Pétri (RdP), adapté à la prise en compte des problèmes de concurrence, de synchronisme et de parallélisme, constitue un excellent outil de spécification fonctionnelle d'un problème et de mise en évidence des contraintes. Les propriétés mathématiques qui découlent de l'analyse des RdPs permettent une étude comportementale et structurelle essentielle à la validation d'une spécification. Les possibilités de simulation offertes par les outils informatiques supportant le formalisme contribuent également à cette validation.

En général, les méthodes de l'étude de systèmes par réseau de pétri se composent de trois étapes : premièrement on écrit le système en termes de réseau, pour obtenir un modèle en réseau ; deuxièmement on analyse le modèle obtenu, pour en déduire des propriétés comme l'absence de blocage, existence d'une solution, etc. Finalement, on fait la révision des propriétés obtenues pour montrer si le système est bon. Le résultat de cette méthode nous indique une analyse qualitative du système. Elle constitue une approche très importante pour avoir une bonne évaluation des systèmes.

I. Réseau de Pétri

1. Qu'est-ce que les réseaux de Pétri

Les réseaux de Pétri sont définis comme étant un formalisme qui permet la description et l'analyse du comportement des systèmes concurrents, introduit par *Carl Adam Petri* en 1962.**[1]**

Les définitions concernant les **réseaux de Pétri** portaient sur deux aspects :

- **Un aspect structurel**

  Quelles sont les actions, quels sont les sites, quelles sont les conditions pour qu'une action

  soit possible et quelles sont les conséquences d'une action?

- **Un aspect comportemental**

  Comment représenter le fonctionnement d'un réseau de Pétri? c.-à-d. ce qui se passe quand une action ou plusieurs actions sont exécutées.

1.1. L'aspect structurel

Définition d'un réseau de pétri

Un réseau de pétri (***R***) est un triple ***R*= (*P, T, W***) où ***P*** est l'ensemble des places (les places représentent les sites) et ***T*** l'ensemble des transitions (les transitions représentent les actions) tel que $P \cap T = \emptyset$ et ***W*** est la fonction définissant le poids porté par les arcs tel que $W : ((P \times T) \cup (T \times P)) \rightarrow \mathbf{N} = \{0, 1, 2, \ldots\}$.

1.1.1. Représentation d'un réseau de Pétri

a. Représentation graphique:

L'un des aspects les plus agréables des réseaux de Pétri est qu'il est extrêmement aisé de les visualiser; c.-à-d., donner une interprétation graphique à sa structure qui peut être représentée à travers un **graphe** bipartie fait de deux types de sommets: les **places** et les **transitions** reliées alternativement par des **arcs orientés** qui portent des poids entiers positifs, si un poids n'est pas porté alors il est égal à **1** (**RdP ordinaire**). Généralement, les places sont représentées par des cercles et les transitions par des rectangles, le marquage d'un **RdP** est représenté par la distribution de jetons dans l'ensemble de ses places telle que chaque place peut contenir un ou plusieurs jetons représentés par des points dans le cercle représentant la place. [**2]**

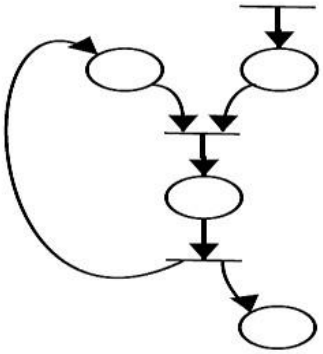

Figure 3 : Réseau de Pétri simple

b. Représentation matricielle :

Une représentation matricielle d'un RdP est offerte afin de simplifier les Tâches d'analyse et de vérification effectuée sur un modèle RdP. Agir sur une représentation graphique d'un modèle RdP est une Tâche délicate en comparant avec une représentation matricielle.

Il est possible de représenter la fonction ***W*** (fonction de poids) par des matrices.**[2]**

Définition

Soit Un réseau de pétri ***R*= (*P, T, W*)** avec ***P*= {p1, p2**, …, **pm} et *T*= {t1, t2**, …,**tn}**, on appelle matrice des pré conditions *pré* la matrice m × n à coefficients dans **N** tel que ***pré* (i,j)= *W*(pi, tj)**, elle indique le nombre de marques que doit contenir la place **pi** pour que la transition **tj** devienne franchissable, de la même manière on définit la matrice des post conditions ***post*** la matrice **n × m** tel que ***post* (i,j)= *W*(tj, pi)** contient le nombre de marques déposées dans **pi** lors du franchissement de la transition **tj**. La matrice ***C*= *post* – *pré*** est appelée matrice d'incidence du réseau (**m** représente le nombre de places d'un **réseau de Pétri** et **n** le nombre de transitions.).

La représentation Graphique d'un marquage dans un RdP marqué est présentée par des marques dans la place appelées jetons.

Le marquage d'un réseau de Petri est représenté par un vecteur de dimension **m** à coefficients dans **N**. La règle de franchissement d'un réseau de Petri est définie par

$$M'(p) = M(p) + C(p, t).$$

c. Représentation d'un RdP marqué :

Un réseau de Pétri marqué est le couple $N = <R, M>$ où :

- *R* est un réseau de Pétri
- *M* est une application de marquage
- M : P→ N

M(p) est le nombre de marques (jetons) contenus dans la place p. Le marquage d'un réseau de Pétri est une opération qui consiste à assigner des jetons dans les places.

On appelle marquage M d'un Réseau de Pétri le vecteur du nombre de marques dans chaque place : la I^{ème} composante correspond au nombre de marques dans la I^{ème} place. Il indique à un instant donné l´état du RdP**.[2]**

Exemple :

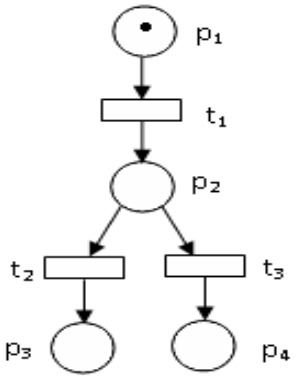

Figure 4: Réseau de Pétri marqué

Pour le réseau de la *Figure 4*

$P = \{p_1, p_2, p_3, p_4\}$ $T = \{t_1, t_2, t_3\}$

$$Pré = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad Post = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice d'incidence est :

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

Le vecteur de marquage M est :

$$M = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Le marquage du RdP présenté à la *Figure 3* est donné par :

$$M = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

On appelle marquage initial, noté M_0, le marquage à l' instant initial ($t = 0$).

1.2. L'aspect comportemental

Le comportement d'un réseau de Pétri est déterminé par sa structure et par son état. Pour exprimer l'état d'un réseau de Pétri, les places peuvent contenir des jetons qui ne sont que de simples marqueurs. **.[1]**

1.2.1. L'état dans un réseau de Pétri

Dans la théorie des réseaux de Pétri, l'état d'un réseau est souvent appelé *marquage* du réseau qui est définit par la distribution des jetons sur les places. Le marquage d'un réseau de Pétri $R = (P, T, W)$ est défini par la fonction de marquage $M : P \rightarrow \mathbf{N}$.

Un réseau de Pétri **marqué** est dénoté par $\sum = (P, T, W, M_0)$ où M_0 est le marquage initial. Le comportement d'un **réseau de Pétri** marqué est déterminé par ce qu'on appelle **règle de franchissement.**

1.2.2. Franchissement d'une transition

Une règle de franchissement est une simple relation de transition qui définit le changement d'état dans un réseau marqué lors de l'exécution d'une action. Afin de définir une règle de franchissement, il est nécessaire de formaliser quand le réseau peut exécuter une action: on dit qu'une transition $t \in T$ peut être **franchie** à partir d'un marquage M (qui représente l'état du système à un instant donné) si et seulement si chaque place d'entrée $p \in {}^*t$ de la transition **t** contient au moins un nombre de jetons qui est supérieur ou égal au poids de l'arc reliant cette place d'entrée **p** avec la transition **t** tel que: $M(p) \geq W(p, t) \quad \forall \quad p \in P$.

Une règle de franchissement est définie par $M'(p) = M(p) - W(p, t) + W(t, p)$ pour tout $p \in P$, ce qui veut dire que lorsque la transition **t** est franchie à partir d'un marquage M, il faut saisir $W(p, t)$ jetons à partir de chaque place d'entrée à la transition **t** et déposer $W(t, p)$ jetons dans chaque place de sortie de la transition **t** ce qui permet de produire un nouveau marquage M'.

Le franchissement d'une transition **t** dénoté par $M[t > M'$ est dite l'**occurrence** de **t**. On dit que deux transitions t_1, t_2 (pas nécessairement distinctes) sont franchies en concurrence par un marquage M si et seulement si $M(p) \geq W(p, t_1) + W(p, t_2)$ pour toute $p \in P$. **.[3]**

Cette vision de l'exécution concurrente de deux transitions dans un RdP est contradictoire avec celle qui impose que deux occurrences de transition sont parallèles si et seulement si : elles sont causalement indépendantes et n'ont pas une relation de conflit entre eux. Deux occurrences sont en conflit si l'un des deux peut avoir lieu mais pas toutes les deux.

1.2.3. L'exécution d'un réseau de Pétri

a. Exécution séquentielle :

- **Séquence de franchissement**

Une séquence de franchissement « s » est une suite de transitions ($t_1, t_2, \ldots, t_n$) qui permet, à partir d'un marquage « M », de passer au marquage « M' » par le franchissement successif des transitions définissant la séquence.

- **Marquage accessible**

Le marquage d'un Réseau de Pétri à un instant donné est une vectrice colonne dont la valeur de la Ième composante est le nombre de marques dans la place P_i à cet instant.

Le franchissement d'une transition conduit à un changement du marquage. Le passage du marquage M_k au marquage M_l par franchissement de la transition T_j est noté : $M_k\ (T_j > M_l$. Le nombre de marques dans la place P_i pour le marquage M_k est noté $M_k(P_i)$. A partir d'un Même marquage, il peut être possible de franchir plusieurs transitions, menant ainsi à des marquages différents. L'ensemble des marquages accessibles à partir du marquage M_0 est l'ensemble des marquages obtenus à partir de M_0 par franchissements successifs d'une ou plusieurs transition(s). Cet ensemble est noté $A(R\ ;\ M_0)$. **[3]**

- **Graphe de marquage**

On peut représenter l'ensemble des marquages accessibles par un graphe si ce dernier et fini. Le graphe de marquage a comme sommet l'ensemble des marquages accessibles

$A(R,M_0)$. Un arc orienté relie deux sommet $\mathbf{M_i}$ et $\mathbf{M_j}$ s'il existe une transition **t** franchissable permettant de passer d'un marquage à un autre $\mathbf{M_i}\ [t>\mathbf{M_j}$. Les arcs du graphe sont étiquetés par les transitions correspondantes. **[3]**

Exemple :

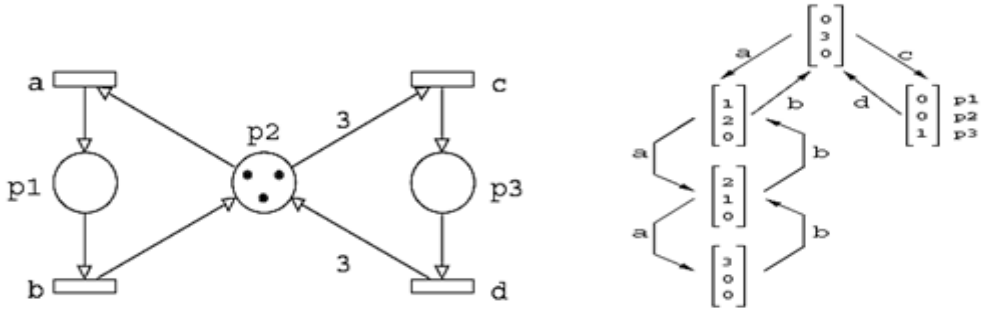

Figure 5 : Graphe de Marquage

- **L'exécution séquentielle d'un réseau de Pétri**

L'exécution séquentielle d'un réseau de Pétri est définie en termes d'un ensemble de séquences d'occurrences. Une séquence d'occurrences est une séquence de transitions franchissables dénotée par $\sigma = M_0\ t_1\ M_1\ t_2 \ldots$tel que $M_{i-1}[t_i > M_i$. Une séquence $t_1\ t_2 \ldots$ est une séquence de transitions (commencée par le marquage M) si et seulement si il existe une séquence d'occurrences $M_0 t_1 M_1 \ldots$ avec $M = M_0$. Si la séquence finie $t_1\ t_2 \ldots t_n$ conduit à un nouveau marquage M' à partir du marquage M, on écrit $M[t_1\ t_2 \ldots t_n > M'$ ou simplement $M[t_1 t_2 \ldots t_n >$ si on ne veut pas spécifier le marquage résultat. **[3]**

b. Exécution concurrente

Une exécution concurrente d'un réseau de Pétri est une exécution dans laquelle plusieurs transitions peuvent se franchir en même temps, elle est souvent déterminée par la notion de processus. Ceci permet de donner une interprétation de la concurrence dans un réseau de Petri selon la sémantique basée sur la vraie concurrence (sémantique d'ordre partiel) qui est interprétée dans la théorie des réseaux de Pétri par un type spécial de réseaux appelés réseaux

d'occurrences. **[3]**

II. Réseaux particuliers

Le graphe associé à un réseau de Petri peut être très complexe. Un certain nombre de situations présente un intérêt particulier :

Graphe d'états

Dans ce cas chaque transition ne dispose que d'une place en entrée et une place en sortie.

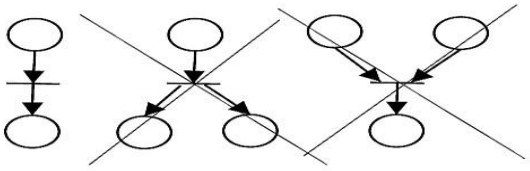

Figure 6 : Graphe d'états ou pas

2.1. Les réseaux sans conflits

Un réseau de Pétri est dit sans conflit si et seulement si toute place a *au plus une* transition de sortie (voir *Figure 7*). Un conflit (structurel) correspond à l'existence d'une place P_i qui a *au moins deux* transitions de sortie T_j, T_k, etc... Notation $< Pi, \{T_j, T_k, \ldots\}>$. Sur le RdP de droite de la *Figure 5*, on a le conflit $< P1, \{T2, T3\}>$. Quand la place *P*1contient une marque, les transitions *T*1 et *T*2 sont franchissables. Seule une des deux transitions peut être franchie : il est nécessaire de prendre une décision pour savoir laquelle des deux le sera effectivement. L'absence ou la présence d'un conflit est une propriété importante d'un réseau de Pétri. **[4]**

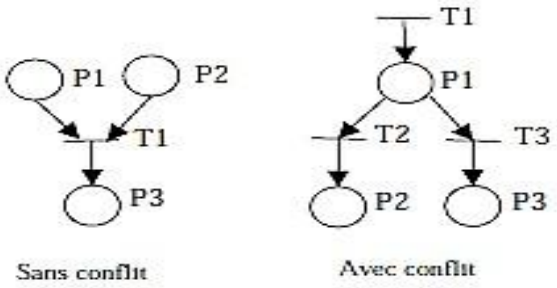

Figure 7: Les réseaux sans conflits

2.2. Les réseaux purs

Un réseau de Pétri est dit pur si et seulement si il n'existe pas de transition ayant une place d'entrée qui est aussi place de sortie. Le RdP représenté *Figure 8* n'est pas pur car la place *P*3 est place d'entrée et place de sortie de la transition *T*1. On parle alors de RdP impur.

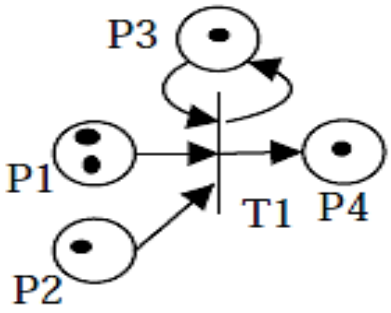

Figure 8 : Les réseaux n'est pas purs

3. Propriétés des RdP

3.1. Réseau K-borné

Une place Pi est bornée pour un marquage initial M0 si pour tout marquage accessible à partir de M0, le nombre de marques dans Pi reste borné. Elle est dite **k-bornée** si le nombre de marques dans Pi est toujours inférieur ou égal à k. Un RdP marqué est (k) borné si toutes ses places sont (k) bornées. **[5**]

Un RdP marqué peut ne pas être borné : sur l'exemple représente *Figure 9*, la transition *T*1 admet la place *P*1 comme unique place d'entrée. La place *P*1 a une marque : la transition *T*1 est franchissable. Comme *P*1 est aussi place de sortie de *T*1, le franchissement de *T*1 ne change pas le marquage de *P*1. La transition *T*1 est donc franchissable en permanence et peut donc être franchie un nombre de fois infini. Chaque franchissement de *T*1 ajoutant une marque dans la place *P*2, le marquage de celle-ci peut donc tendre vers l'infini. **[5]**

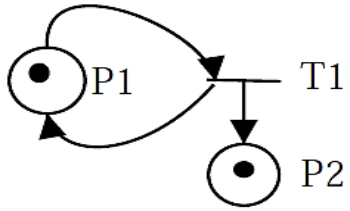

Figure 9 : RdP non borné

Définition : Un RdP marqué est sauf ou binaire pour un marquage initial M_0 s'il est 1-borné.

3.2. Réseau vivant

L'évolution du marquage d'un RdP se fait par franchissement de transitions. Lorsqu'au cours de son évolution, certaines transitions ne sont jamais franchies, cela indique que l'évènement associé à la transition ne se produit pas et que le marquage d'une partie du RdP n''évolue pas. Cela indique que le sous-système modélisé par cette partie-là ne fonctionnera pas. Il y a donc un problème au niveau de la conception du système. L'idée est d'être capable de détecter systématiquement ce phénomène par l'analyse de propriétés du modèle RdP du système afin de disposer d'un outil d'aide à la conception des systèmes. **[5]**

Définition : Une transition T_j est vivante pour un marquage initial M_0 si pour tout marquage accessible M_k, il existe une séquence de franchissements à partir de M_k contenant T_j :

$$\forall M_k \in {}^*M_0, \quad \exists S, \quad M_k \mid S > \; et\; S = \ldots T_j \ldots .$$

Si une transition T_j est vivante alors, à tout instant, on sait que T_j peut être franchie dans le futur. Dans le cas d'un RdP modélisant un système fonctionnant en permanence, si une transition n'est pas vivante et si une fonction du système est associée au franchissement de cette transition, cela veut dire qu'à partir d'un certain instant, cette fonction ne sera plus disponible dans le futur, ce qui peut traduire une erreur ou une panne.

Exemples : Les transitions *T*1 et *T*2 du RdP marqué *Figure9* sont vivantes

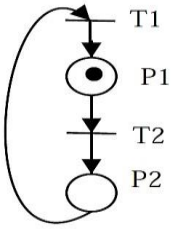

Figure 10 : RdP vivant

3.3. Réseau réinitialisable

Soit un réseau de pétri marqué < *R*;*M*$_0$ > et soit A(*R*;*M*$_0$) l'ensemble de ses marquages accessibles. Ce réseau marqué est réinitialisable si et seulement si :

$$\forall M \in \mathcal{A}(\mathcal{R}; M_0)\ , M \neq M_0\ ,\ \ \exists s \ \text{ tel que } \ M \xrightarrow{s} M_0$$

On dit également que ***M*$_0$** est un état d'accueil (en anglais "*home state*") pour le réseau de Pétri marqué.

Soit un réseau de Pétri marqué <***R***;***M*$_0$**>, s'il est réinitialisable et que toute ses transitions sont **quasi-vivantes** ou **vivantes**, alors il est vivant. **[5]**

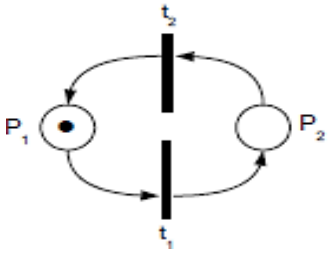

Figure 11: Réseau réinitialisable

4. Les méthodes d'analyse des RdP

Les méthodes d'analyse des RdP peuvent être classées en trois groupes:

- Méthode d'arbre de couverture.
- Approche d'équations matricielles.
- Technique de réduction et de décomposition.

4.1. Méthode d'arbre de couverture

L'idée la plus naturelle pour étudier les propriétés d'un RdP est de construire le graphe de tous les marquages accessibles. Le graphe des marquages accessibles est un graphe dont chaque sommet correspond à un marquage accessible et dont chaque arc correspond au franchissement d'une transition permettant de passer d'un marquage à l'autre. **[6]**

Pour un RdP marqué (**R**, **M0**), à partir d'un marquage initial M_0 on peut obtenir autant de nouveau marquage qu'il existe de transition franchissable. A partir de chaque nouveau marquage on peut accéder à d'autres nouveaux marquages.

Le résultat de ce processus est un arbre dont chaque nœud est un marquage accessible et chaque arc est une transition franchissable qui transforme un marquage à un autre.

L'arbre de couverture est une méthode alternative. Le mécanisme de construction est le même que pour le graphe des marquages accessibles. pour chaque nouveau marquage (nœud du graphe) ajouté, on vérifie s'il n'est pas supérieur à un marquage déjà présent sur au moins une séquence entre M0 et le nouveau marquage. Si tel est le cas tous les marquages de place supérieurs sont remplacés par ω. Ce symbole matérialise le fait que la
place en question peut contenir autant de jetons que souhaité (elle est donc non bornée).

Dans la suite, les places non bornées le demeurent naturellement et ceci quelles que soient les transitions franchies, ainsi le symbole ω ne disparaît jamais. Cette méthode produit le graphe de couverture, un graphe fini dans tous les cas.

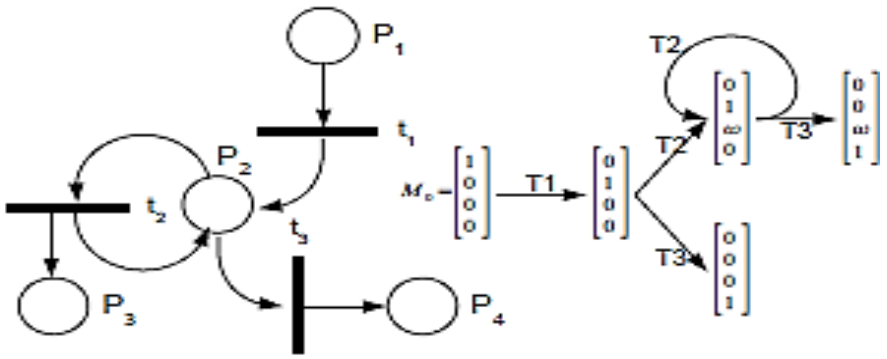

Figure 12: RDP et son arbre de couverture

4.2. Approche d'équations matricielles

L'analyse par équations matricielles (algèbre linéaire) permet d'étudier des propriétés d'un réseau (caractère borné, vivacité) indépendamment d'un marquage initial. De ce fait, on parlera de propriétés structurelles du réseau. Par exemple, on pourra dire qu'un réseau est structurellement borné s'il est borné pour tout marquage initial fini. De la même façon, si pour tout marquage initial, le réseau est vivant, on dira que le réseau est structurellement vivant. **[6]**

4.3. Technique de réduction et de décomposition

L'analyse des RdPs par réduction permet d'obtenir à partir d'un RdP marqué, un RdP marqué plus simple, c'est-à-dire avec un nombre réduit de places et un nombre réduit de transitions, ceci, en appliquant un ensemble de règles dites règles de réduction. **[6]**

5. Réseaux de Pétri de haut niveau

Dans la pratique, on est amené à modéliser toute sorte de système, à savoir les protocoles de communication, les systèmes de production, les systèmes réactifs, les systèmes temps réel, etc. Cette variété de systèmes a poussé les chercheurs à étendre la théorie des RdPs en introduisant beaucoup de concepts relatifs à chaque domaine d'application. Ces efforts ont donné naissance aux RDPs de haut niveau tels que les RDPs colorés, hiérarchiques, temporisés et autres. **[6]**

5.1. Les réseaux de Pétri coloré

a. Définitions formelles pour les réseaux de Pétri colorés

La modélisation d'un système réel peut mener à des réseaux de Pétri de taille trop importante rendant leur manipulation et/ou leur analyse difficile. La question est alors de modifier (étendre) la modélisation par RdP de façon à obtenir des modèles RdP de plus petite taille : Les RdPs colorés. Ils sont importants pour la modélisation de systèmes de production (au sens large). **[6]**

b. Association de couleur

Dans le but de différencier les jetons, on leur associe des couleurs ou des entiers ou encore un ensemble d'étiquette. On associe à chaque place l'ensemble des couleurs des jetons qui peuvent y séjourner et à chaque transition l'ensemble de couleurs pour laquelle elle est franchissable**. [6]**

c. Association de fonctions aux arcs

Dans un RdP coloré les arcs ne seront plus étiquetés par des entiers mais par des fonctions, qui désigneront les actions à effectuer à chaque franchissement. A chaque arc correspond une fonction qui à chaque couleur de transition associe un vecteur ligne d'entiers positifs qui décrit le nombre de jetons de chaque couleur qui sont retirés ou produits à chaque franchissement **[6]**

- **Définition informelle**

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

- **Définition formelle**

D'abord on définit des multi-ensembles.

- **Définition :**

Un multi-ensemble **m**, sur un ensemble non vide **S**, est une fonction, représentée comme somme formelle :

$$\sum_{s \in S} m(s)`s$$

On dénote **SMS** l'ensemble de tous les multi-ensembles sur S. Les nombres entiers non négatifs **{m(s) : s S}** sont les coefficients du multi-ensemble.

- Les éléments d'un type, *T*. L'ensemble de tous les éléments dans *T*est dénoté par le nom *T* lui-même de type ;
- Le type d'une variable, *v* - dénoté par *Type(v)* ;
- Le type d'une expression, *expr* - dénoté par *Type(expr)* ;
- L'ensemble des variables dans une expression, *expr* - dénoté par *Var(expr)* ;
- Une association d'un ensemble de variables, V - assignant chaque variable *v* V à un élément *b(v) Type(v)* ;
- La valeur obtenue en évaluant une expression, *expr*, dans une association, *b* - dénoté par *expr < b >.Var(expr)* est appliqué d'abord pour obtenir un sous-ensemble des variables de *b*, puis l'évaluation est exécutée en remplaçant chaque variable *v Var(expr)* à la valeur *b(v) Type(v)* qui est déterminée par la association.
- Une expression sans variables s'appelle une expression fermée. Elle peut être évaluée dans toutes les associations, et toutes ces évaluations donnent la même valeur.

- **Définition :**

Un réseau coloré est un tuple CPN = (, P, T, A, N, C, G, E, I) où :

- est un ensemble fini de types non-vides, appelé des ensembles de couleur ;
- **P** est un ensemble fini de places ;
- **T** est un ensemble fini de transitions ;
- **A** est un ensemble fini d'arcs tel que :

$$P \cap T = P \cap A = T \cap A = \emptyset$$

- **N** est une fonction de nœud. Il est défini de A dans P x T T x P ;
- **C** est une fonction de couleur. Il est défini de P dans ;
- **G** est une fonction de garde. Il est défini de T dans les expressions tel que :

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

- **E** est une fonction d'expression d'arc. Il est défini de A dans les expressions tel que :

$$\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$$

Où *p(a)* est la place de *N(a)* ;

- **I** est une fonction d'initialisation. Elle est définie de P dans les expressions fermées tels que :

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

- **Définition :**

Un pas est un multi-ensemble des éléments de l'association. Un pas Y est accessible dans un marquage M si et seulement si la propriété suivante est satisfait :

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

Quand un pas Y est accessible dans le marquage M1, s'il se produit, M1change à un marquage M2, défini par :

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

La première somme s'appelle le jeton supprimé tandis que la seconde s'appelle le jeton ajouté. D'ailleurs on dit que le M2 est directement accessible de M1 par l'occurrence du pas Y, qu'on dénote également : M1[Y > M2. Une séquence d'occurrence est une séquence des marquages et des pas :

$$M_1[Y_1 > M_2[Y_2 > M_3 \cdots M_n[Y_n > M_{n+1}$$

Tel que $\forall i \in \{1, \cdots, n\} : M_i[Y_i > M_{i+1}$.On utilise [M > pour dénoter l'ensemble de tous les marquages qui sont accessibles de M.

Exemple :

- Soit x1 C1, x2 C2
- t est franchissable pour (x1, x2) ssi P1 contient au moins un jeton de couleur x1 (X1(x1, x2) = x1). P2 contient au moins un jeton de couleur x2
- Si on franchit t pour (x1, x2) alors un jeton de couleur x1est retiré de P1, un jeton de couleur x2 est retiré de P2, un jeton de couleur < x1, x2> est produit dans P3: <X1, X2> (x1, x2) = < x1, x2>

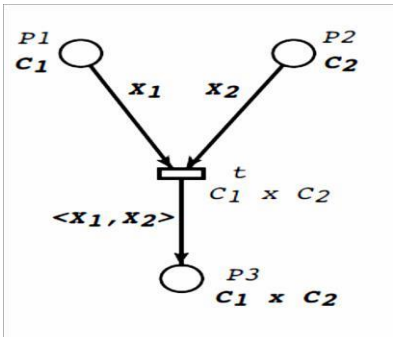

Figure 13: Dynamique d'un RdPC

5.2. Les réseaux de Pétri à Objet

Les réseaux de Pétri objet (OPN) étendent le formalisme des réseaux de Pétri colorés avec une intégration complète des propriétés orientées objet y compris l'héritage le polymorphisme et la liaison dynamique, l'orientation objet fournit des primitives de structuration puissant permettant la modélisation des systèmes complexes **[6]**

5.3. Les réseaux de Pétri temporels

Les réseaux de Pétri temporels sont obtenus depuis des réseaux de Pétri en associant des dates min et max aux transitions.

Supposons qu'une transition t est devenue sensibilisée pour la dernière fois à l'instant x, alors elle ne peut l'être encore qu'après l'instant x + min(t) et avant l'instant x + max(t), sauf si le tir d'une autre transition a désensibilisé t avant que celle-ci ne soit tirée. Le tir des transitions est de durée nulle. Les réseaux de Pétri temporels expriment nativement des spécifications en «délai». Ils peuvent aussi exprimer des spécifications en «durées». Leur domaine d'application est donc large. **[6]**

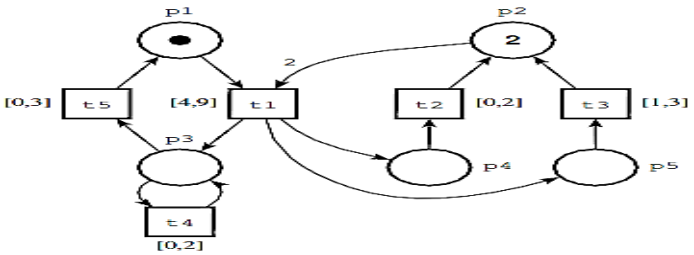

Figure 14 : Exemple de réseau de Pétri temporel

III. Editeur graphique

1. Logiciel graphique

Dans le domaine de l'infographie, un logiciel graphique (ou logiciels d'édition d'image, logiciel de graphisme ou logiciel de retouche d'image) est un programme informatique ou une collection de programmes permettant de manipuler et de traiter des formes, dessins, vecteurs ou images sur un ordinateur, éventuellement via une tablette graphique mais aussi un écran tactile comme le logiciel.

Ces logiciels, plus ou moins spécialisés sont utilisés par les artistes plasticiens, ou pour la retouche photo ou encore par des designers, architectes ou ingénieurs pour modéliser et présenter leurs projets, ou dans le cinéma et partout où l'image peut avoir une importance. **[7]**

2. Graphisme

Le graphisme est une discipline intellectuelle et manuelle qui consiste à choisir et à utiliser des d'éléments graphiques (dessins, caractères typographiques, photos couleurs...) pour élaborer un objet de communication. Chacun des éléments est symbolique et signifiant dans la conception du projet, selon les axes définis éventuellement avec d'autres intervenants du domaine de la communication.

Selon ses domaines d'interventions (communication d'entreprise, presse, édition, packaging, publicité, web, etc.), il fait partie de la chaîne graphique liée à l'imprimerie ou à d'autres média. **[8]**

2.1. Son Origine

L'origine du graphisme en tant que discipline et la reconnaissance du métier de graphiste est floue. On peut, en revanche, reconnaître des facteurs qui ont favorisés l'apparition et la reconnaissance du terme et non la discipline, celle-ci étant aussi ancienne que l'art graphique et les peintures préhistoriques.

La discipline s'émancipe à la fin du XIXe siècle avec les typographes créant leurs propres bureaux de conception, lorsque ceux-ci n'étaient alors qu'ouvriers-artisans de la chaîne

de production en imprimerie, à une époque ou il existe un vide entre l'artisanat d'art et l'art (graphique) entre l'art de faire, le savoir-faire et l'art communication de la pensée.

En effet, si l'artiste et l'artisan se confondent dans l'Histoire de l'art ce n'est que tardivement à travers les courants artistiques tel que le romantisme, le symbolisme et l'expressionnisme, puis par l'invention de nouveaux médiums (photographie, cinéma, informatique) que l'art est reconnu comme un instrument de la pensée et ne se reconnaît plus dans l'artisanat considéré alors comme l'unique expression d'un savoir-faire et/ou d'une recherche esthétique.

Au XXe siècle l'industrialisation, la société de consommation, l'émergence de nouveaux médias, du marketing et de la publicité, mais aussi l'apparition de disciplines connexes (design et architecture) favorisent l'émergence d'un nouveau type d'emplois spécialisé dans la création graphique pour valoriser les outils de communication. Le graphiste devient alors celui qui formalise et clarifie un message de communication, puis qui le met en page graphiquement. Mettant tout à tour son intellect puis sa créativité graphique au service d'une commande, "le graphiste" est alors moins considéré comme un artisan, ni reconnu en tant qu'artiste, et c'est dans ce contexte contemporain que la discipline et le terme graphisme se popularise pour désigner tout et n'importe quoi.

Largement galvaudé, le terme de graphiste correspond pourtant à un cursus précis dispensé dans des écoles de graphisme et qui préparent encore aujourd'hui, à toutes les techniques du prépresse (imprimerie). Connaissance de la typographie, de l'art de la mise en page, des signes et du code typographique en vigueur, le graphiste doit d'abord conceptualiser un message clair avant de mettre sa créativité au service de l'illustration graphique d'un projet. **[8]**

Conclusion

Dans ce chapitre nous avons donné les définitions de base d'un réseau de Pétri. Bien que simples, ces définitions se compliquent lorsque l'on va au bout des choses.

On a représenté brièvement 2 types de réseau : réseau de Place/Transition et réseau coloré. Ils sont les représentations mathématiques permettant la modélisation d'un système. Et l'analyse d'un réseau de Pétri peut révéler des caractéristiques importantes du système concernant à sa structure et son comportement dynamique.

Les réseaux de Pétri Place/Transition ont une théorie basée sur une base mathématique solide, les résultats sont profonds. Les systèmes étant modélisé par les réseaux de Pétri Place/Transition peuvent être évalué facilement grâce à ces résultats. Mais avec les applications industrielles, c'est très difficile à être modélisé par ces réseaux à cause des règles de franchissement trop simple, donc c'est trop compliqué, on doit peut-être utiliser des millier de places et transitions pour une petite application. Dans ce cas, les RdP Colorés sont plus raisonnables.

Les réseaux colorés ont une définition très compliquée, c'est très difficile pour les praticiens de maitriser la syntaxe du réseau. Mais heureusement, il existe des outils pour éditer les réseaux colorés, vérifier sa syntaxe, vérifier automatiquement des propriétés comportementales du réseau, etc. Donc, les praticiens peuvent utiliser le réseau coloré mais ils ne doivent pas apprendre par cœur leurs syntaxes. Il est similaire dans le cas du langage, on peut écrire des programmes mais on ne doit pas connaitre toutes les grammaires du langage.

La relation entre le réseau Place/Transition et le réseau coloré peut être considéré comme relation entre le langage à bas niveau (le langage de machine, le langage Assembly etc.) et le langage à haut niveau (Java, C, Pascal, etc.). Ils ont même capacité de la programmation mais pour les grandes applications, qui conviennent plus aux langages de haut niveau que ceux de bas niveau.
Nous avons ensuite, présenté la définition de logiciel graphique , la définition de graphisme ,et son origine

Nous expliquerons dans le prochain chapitre Conception de notre outil prototype.

Chapitre II

Conception de notre outil prototype

I. Introduction

Le « **cycle de vie d'un logiciel** » (en anglais *software lifecycle*), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la **validation** du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la **vérification** du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre. **[9]**

Le cycle de vie du logiciel comprend généralement au minimum les activités suivantes:

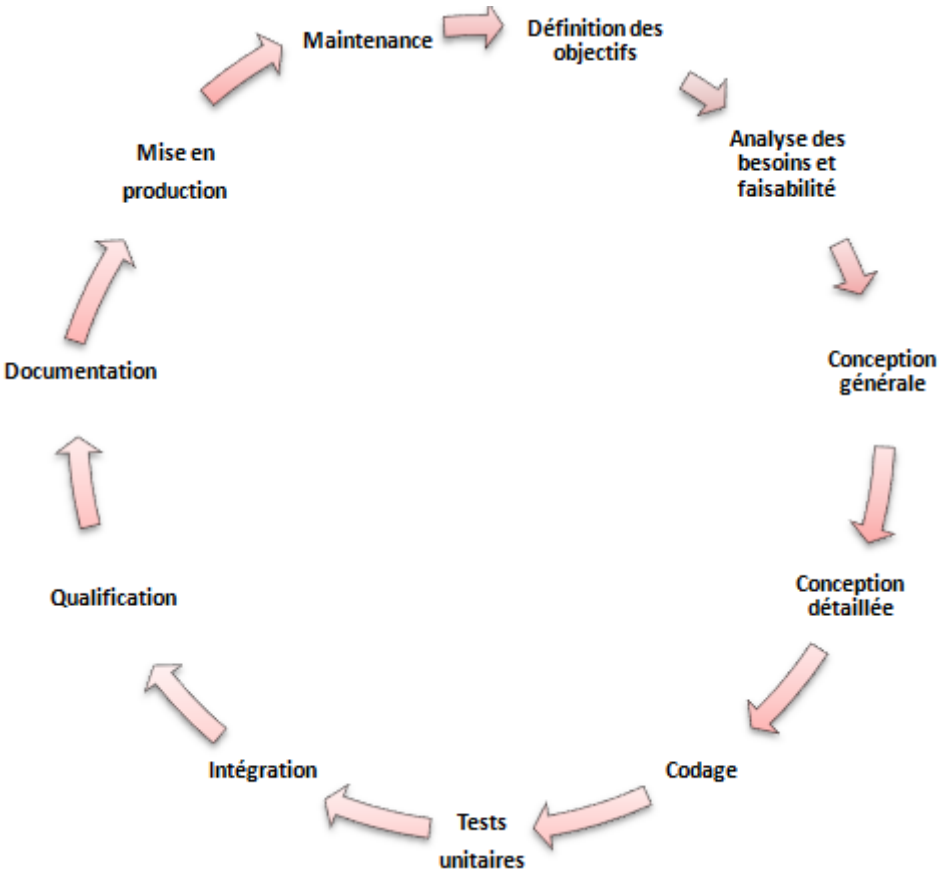

Figure 15 : cycle de vie d'un logiciel

✓ **Définition des objectifs**

Consistant à définir la finalité du projet et son inscription dans une stratégie globale.

✓ **Analyse des besoins et faisabilité**

C’est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes.

- ✓ **Conception générale**

  Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.

- ✓ **Conception détaillée**

  Consistant à définir précisément chaque sous-ensemble du logiciel.

- ✓ C**odage** (Implémentation ou programmation)

  Soit la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.

- ✓ **Tests unitaires**

  Permettant de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.

- ✓ **Intégration**

  Dont l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de *tests* d’intégrations consignées dans un document.

- ✓ **Qualification** (ou *recette*)

  C’est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.

- ✓ **Documentation**

  Visant à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.

- ✓ **Mise en production**

  C’est le déploiement sur site du logiciel .

- ✓ **Maintenance**

  Comprenant toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

1. Future système

La figue 15 suivante illustre future système. Le système projeté se déroule en **07** étapes :

- **Définition des composantes** : définir les briques de base de réseaux RDP ;
- **Etablir les dépendances:** définir les liaisons entre ces composantes ;
- **Marquage et vérification :** *pour chaque place, spécifier le nombre de jeton ;*
- **Etablir le RDP** *:* Etablir le schéma global du réseau de pétri *(RdP)*
- **Validation** : valider le schéma de l'utilisateur ;
- **Génération des XML(s)** *:* le résultat peut exporter sous format standard d'échange (XML) ou bien avoir une possibilité de charger un projet existant sous forme XML
- **Résultat et feedback** *:* l'utilisateur final peut analyse le résultat de simulation et introduire des modifications pour lancer un nouveau test.

Pour mener la réalisation de ce système avec ces fonctionnalités présentées précédemment, nous allons présenter dans la section suivante les étapes de notre démarche adoptée**.**

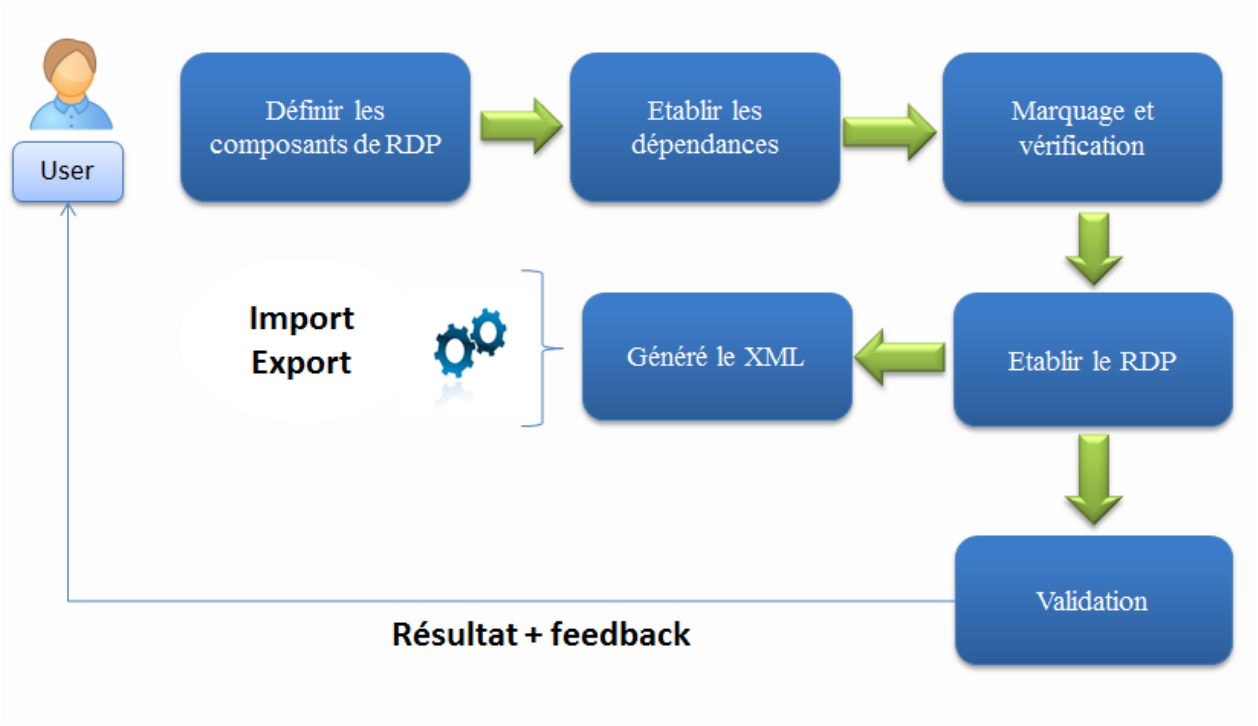

Figure 16: Future Système

2. Vue globale de notre approche

Afin d'atteindre notre objectif, nous avons adopté une démarche constitué de trois étapes fondamentale : (i) analyse de domaine de RDP, (ii) modélisation de notre système en utilisant le paradigme UML et (iii) et l'implémentation d'un outil prototype destinée pour deux types des utilisateurs : *étudiant* et *enseignant* dans le domaine d'éducation. La figue 16 suivante illustre Vue globale de notre démarche.

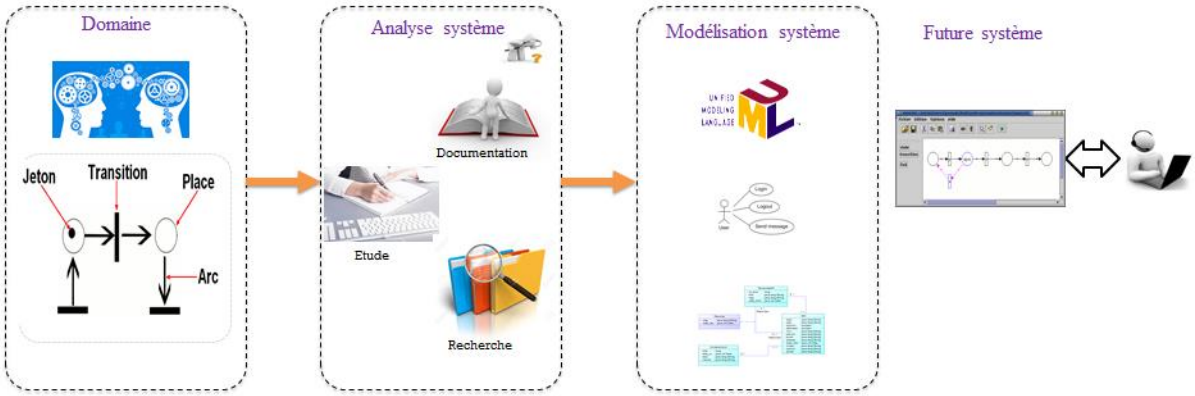

Figure 17 : Vue globale de notre démarche

Analyses de domaine :

Dans notre analyse de domaine des RdPs nous avons fait une recherche dans la littérature analyser les articles et les outils commerciaux et académique des réseaux de pétri. Cette analyse sa nous permet de dégager les concepts de domaine, les dépendances entres ces concepts ainsi les fonctionnalités de ces outils. Les outils commerciaux sont caractérisés par:

- Outils propriétaires ;
- Pas de détail sur l'implémentation.
- Boite noire (*black box*) ;
- Difficulté pour *étudier* et *réutiliser* les codes sources de ces outils.

La figue 17 suivante illustre l'analyse de notre domaine.

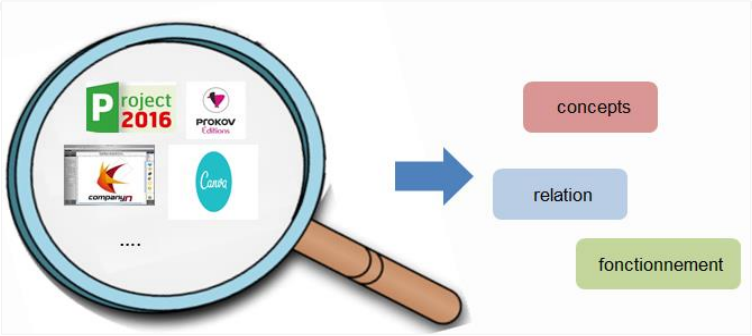

Figure 18: Analyse de domaine

II. Modélisation UML

1. Méthodologies de conception

Dans cette section, nous allons présenter les étapes de modélisation de notre système.

1.1.Modélisation avec UML

UML (Unified Modeling Language ou « langage de modélisation unifié »).

Un modèle est une représentation simplifiée d'un problème.UML permet d'exprimer les modèles objets à travers un ensemble de diagrammes. Ces derniers sont des moyens de description des objets ainsi que des liens qui les relient. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, qu'il peut être appliqué à toutes sortes de systèmes et ne se limitant pas au domaine informatique**.[10]**

1.2. Spécification du système

Parmi les diagrammes UML largement connus on cite :

- ✓ ***Diagramme de cas d'utilisation*** **:** il permet de recueillir, d'analyser et d'organiser les besoins. Avec lui débute l'étape d'analyse de notre système.
- ✓ ***Diagramme de séquence*** **:** il permet de représenter des interactions entre objets et acteurs, selon un point de vue temporel avec une chronologie des envois de messages, c'est un type de diagramme d'interaction.
- ✓ ***Diagramme de classe*** **:** il exprime la structure statique du système en termes de classes, ainsi qu'aux relations entre ces classes.

Dans notre travail, nous allons essayer de travailler avec deux vues (quatre diagrammes)

- **Vue statique :**

-Diagramme des classes : définit la structure statique.

- **Vue dynamique :**

- Diagramme de cas d'utilisation : décrit les besoins utilisateurs.

- Diagramme de séquence **:** scénarios et flots de messages-

- Diagramme d'activité : décrit le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités.

1.2.1 Identification des cas d'utilisation :

Les cas d'utilisation constituent une technique qui permet de déterminer les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système. En d'autres termes, ils décrivent le comportement d'un système du point de vue de ses utilisateurs. Ils décrivent les interactions entre les utilisateurs d'un système et le système lui-même. **[11]**

Dans La figure 18 nous présentons le diagramme de cas d'utilisation pour la compréhension du fonctionnement du système.

- **Diagramme de cas d'utilisation :**

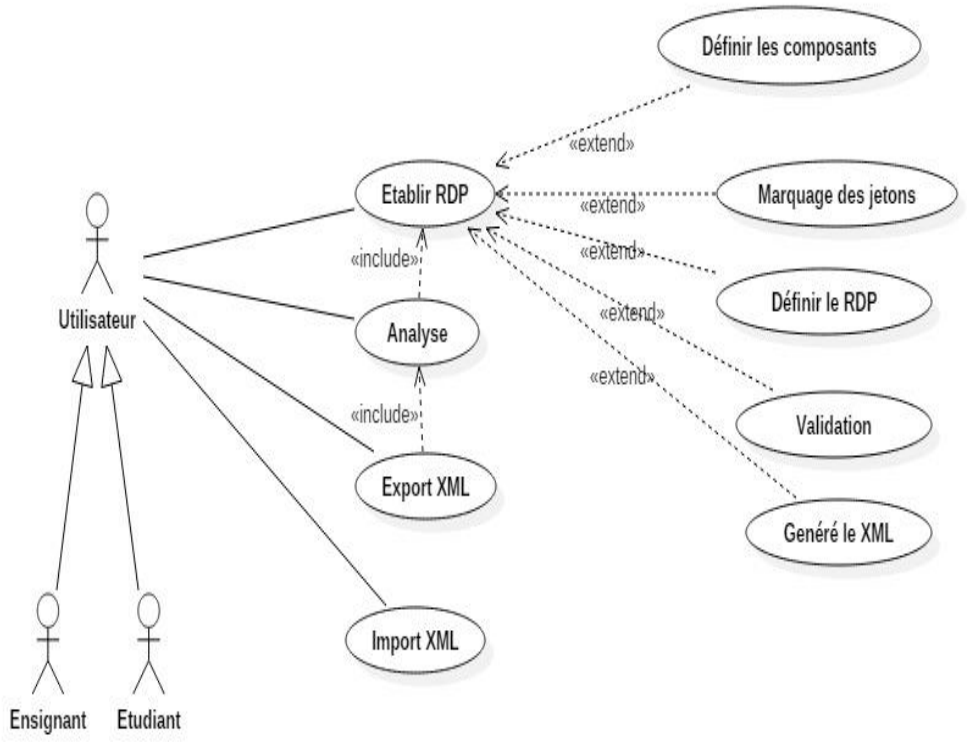

Figure 19 : Diagramme des cas d'utilisation

La liste des cas d'utilisation déterminés avec les acteurs correspondants est présentée dans le tableau suivant :

	Cas d'utilisation	Acteur
1	Etablir le RdP	Utilisateur (Etudiant, Enseignant)
2	Analyse	Utilisateur (Etudiant, Enseignant)
3	Export XML	Utilisateur (Etudiant, Enseignant)
4	Import XML	Utilisateur (Etudiant, Enseignant)

Tableau 1 : table description de cas d'utilisation.

- **Description détaillée des cas d'utilisation :**

Nous proposons dans ce qui suit une description textuelle des principaux cas d'utilisation.

Cas d'utilisation : « Etablir le RdP»

Description sommaire :
Acteurs : Utilisateur (Etudiant, Enseignant)
Objectif : Produit le RDP
Description des enchainements :
Pré conditions : L'utilisateur nécessite un problème de modélisation.
Post conditions : modèle de RDP.

Tableau 2 : table description de cas d'utilisation « Etablir le RdP».

Cas d'utilisation : «Analyse**»**

Description sommaire :
Acteurs : Utilisateur (Etudiant, Enseignant)
Objectif : exécuter le RdP
Description des enchaînements :
Pré conditions : établir le RdP
Post conditions : RdP exécuté

Tableau 3 : table description de cas d'utilisation «Analyse».

Cas d'utilisation : «Export XML**»**
Description sommaire :
Acteurs : Utilisateur (Etudiant, Enseignant)
Objectif : Export le RdP en XML
Description des enchaînements :
Pré conditions : Un modèle de RdP
Post conditions: Fichier XML.

Tableau 4 : table description de cas d'utilisation «Export XML».

Cas d'utilisation : «Import XML**»**
Description sommaire :
Acteurs : Utilisateur (Etudiant, Enseignant)
Objectif : Import le XML en RdP
Description des enchaînements :
Pré conditions: Fichier XML.
Post conditions : Modèle RdP

Tableau 5 : table description de cas d'utilisation «Import XML».

- **Diagrammes de séquence :**

Les diagrammes de séquences permettent de représenter les interactions entre objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages. **[12]**

Dans cette partie, nous allons présenter un diagramme de séquence comme suit :

La figure illustre l'action/réaction entre l'utilisateur et le système pour établir un RdP :

- L'utilisateur demande la création d'un RDP ;
- Le système affiche la palette des composants ;
- L'utilisateur définir les composants, liens, et marquages ;
- L'utilisateur demande l'analyse de RdP ;
- Le système affiche le résultat.

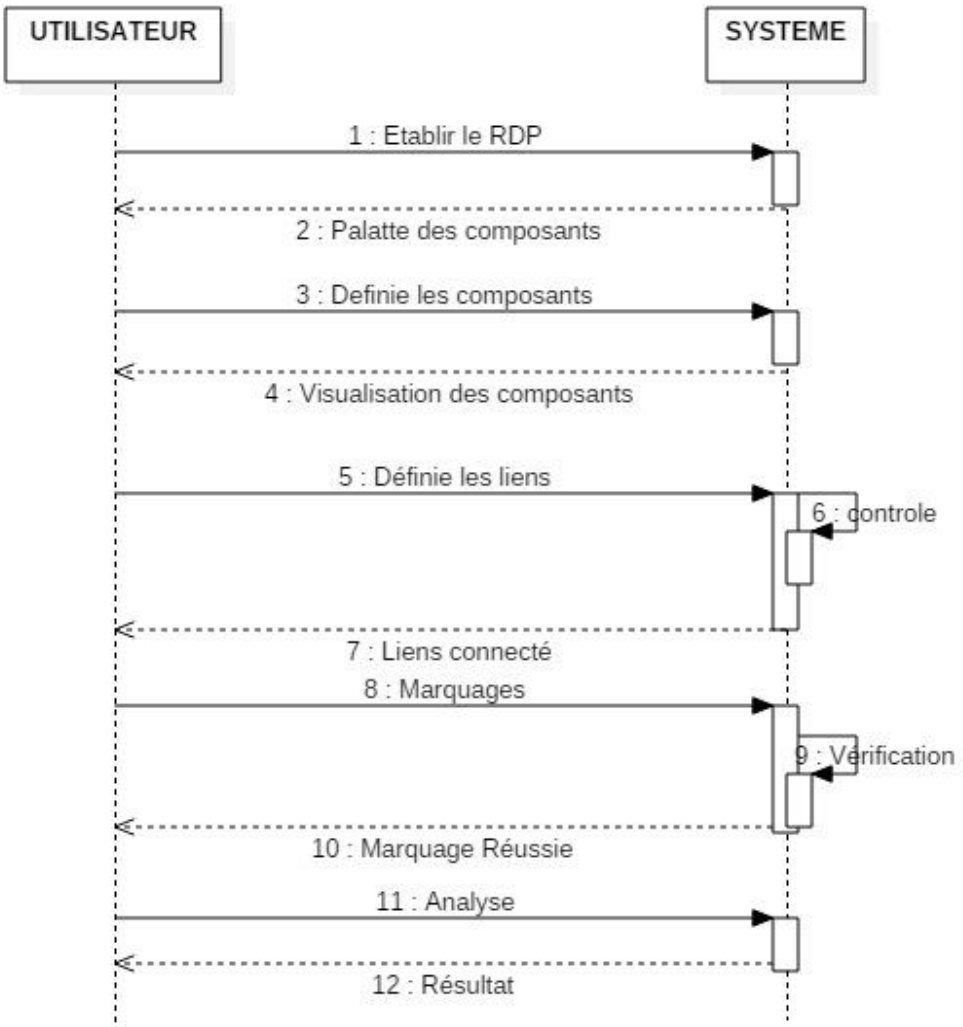

Figure 20: Diagrammes de séquence

- ***Diagramme* d'activité :**

Les diagrammes d'activités décrivent le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités. Une activité désigne une suite d'actions. **[13]**

Le passage d'une action vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début immédiat d'une autre (elles sont automatiques).

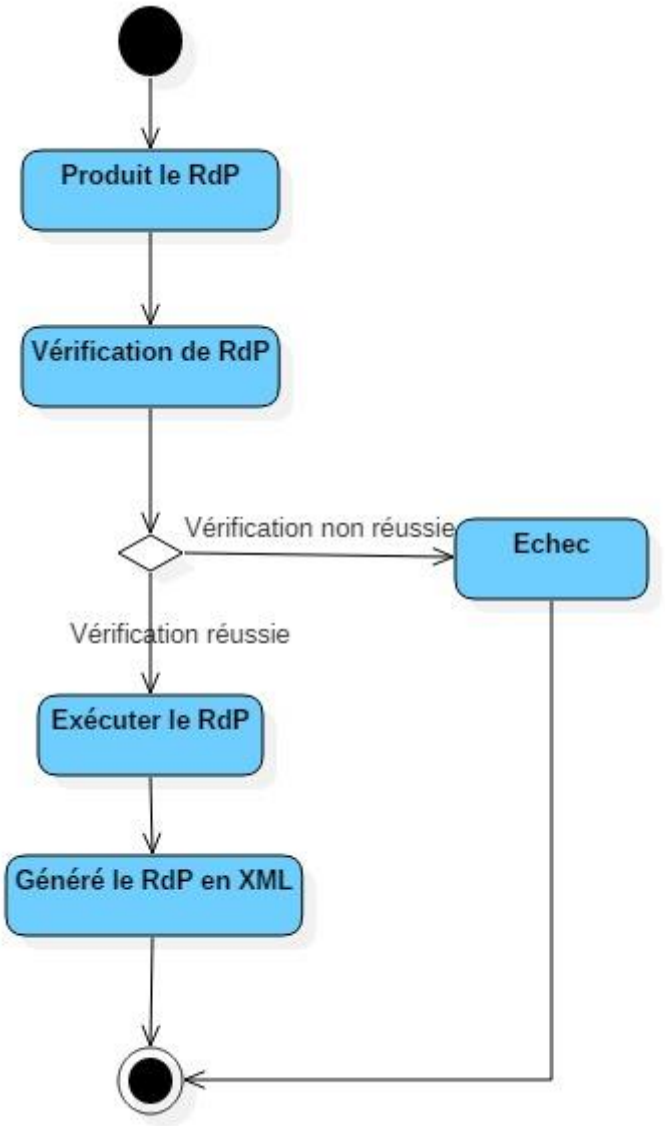

Figure 21: Diagramme d'activité

- **Diagramme de Classe *:***

Les diagrammes de classes servent à comprendre la structure de classe des projets, on les utilise pour personnaliser, partager et présenter avec d'autres classes les informations relatives au projet. **[14]**

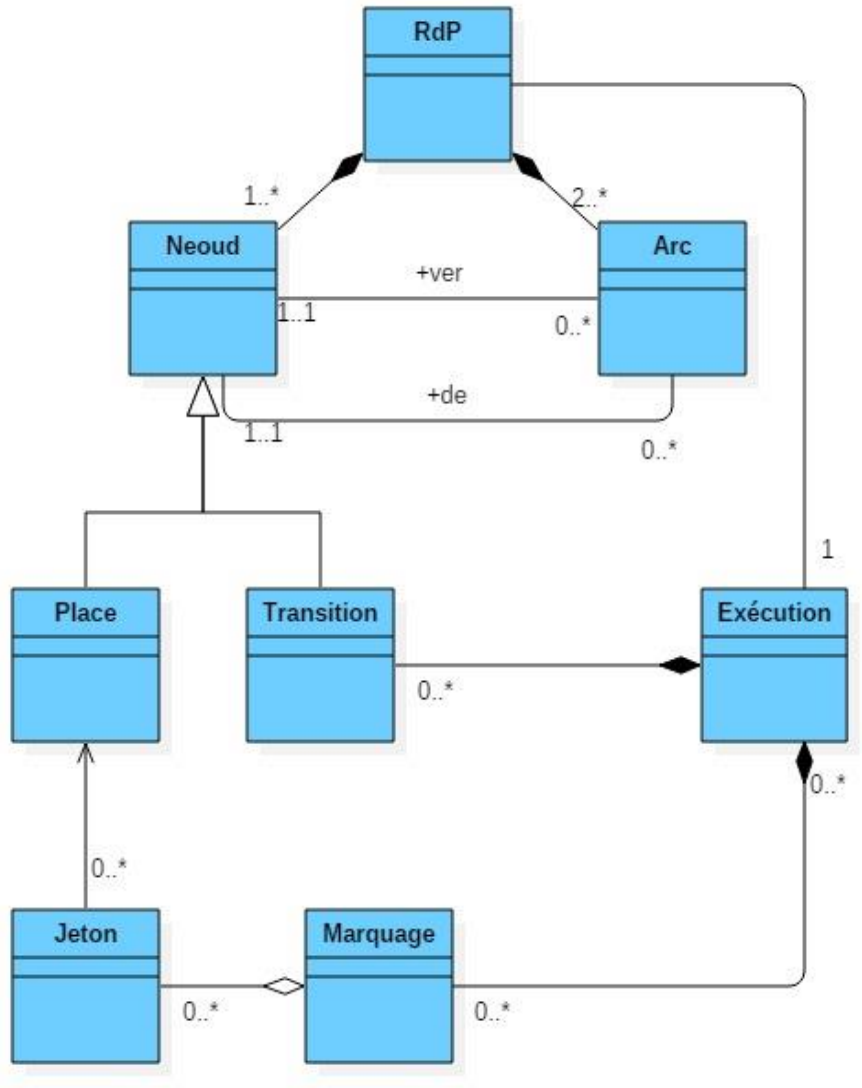

Figure 22: Diagramme de Classe

Les éléments impliqués dans ce diagramme sont :

a) *Arc*

Un Arc relie deux Nœuds entre eux, mais les contraintes inhérentes au modèle des réseaux de Pétri sont vérifiées dans l'interface. Un Arc connaît son Nœud de départ et son Nœud d'arrivée ainsi que la liste d'étiquettes qui lui sont associées.

b) *Nœud*

Cette classe réunit les caractéristiques des Nœuds du réseau, à savoir les Places et les Transitions. Elle possède une liste des Arcs entrants et une des Arcs sortants.

c) *Place*

Une Place est reliée à une ou plusieurs Transitions via des Arcs. La Place est l'entité qui contient les Jetons, qui ne peuvent y être placés que si la Place les admet. C'est une sous classe de Nœud.

d) *Transition*

Une Transition est reliée à une ou plusieurs Places via des Arcs. La Transition est l'entité qui conditionne le passage des Jetons (tir de la Transition) et qui peut agir sur ceux-ci.

C'est une sous-classe de Nœud.

e) *RdP*

Un RdP Simple est l'entité qui contient les Nœuds, les Arcs, les Classes d'un réseau de Pétri.

f) Marquage

est le nombre de marques (jetons) contenus dans la place p. Le marquage d'un réseau de pétri est une opération qui consiste a assigner des jetons dans les places.

g) Jetons

La représentation graphique d'un marquage dans un RdP marque est représenter par des marques dans la place appelées jetons.

h) Exécution :

avoir le résultat de ce travail.

Conclusion :

Dans ce chapitre nous avons présenté une étape fondamentale de notre système en utilisant le paradigme UML. En a utilisé quatre diagramme qui définie comme suit :

Le diagramme de cas d'utilisation qui décrive les interactions entre les utilisateurs d'un système et le système lui-même.

Diagramme de séquence : permettre de représenter les interactions entre objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages.

Diagramme d'activité décrive le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités. Une activité désigne une suite d'actions

Diagrammes de classes aidé à comprendre la structure de classe des projets, on les utilise pour personnaliser, partager et présenter avec d'autres classes les informations relatives au projet.

Chapitre III

La Réalisation

Introduction

Après les études théoriques et préliminaires menées précédemment nous allons entamer le volet pratique de notre projet. Nous définirons les étapes de réalisation, et la mise en œuvre de notre éditeur graphique. Nous commencerons par une brève présentation des outils de développement que nous avons utilisé ; et nous terminerons par l'implémentation de notre éditeur.

I. Outils de développement

1. NetBeans

Est un projet open source fondé par Sun Microsystems. L'IDE NetBeans est un environnement de développement permettant d'écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'IDE NetBeans. L'IDE NetBeans est un produit gratuit, sans aucune restriction quant à son usage. **[15]**

Figure 23: NetBeans

2. Java

Java est un langage Objet permettant le développement d'applications complètes s'appuyant sur les structures de données classiques (tableaux, fichier) et utilisant abondamment l'allocation dynamique de mémoire pour créer des objets en mémoire. La notion de structure, ensemble de données décrivant une entité (un objet en Java), est remplacée par la notion de classe au sens de la programmation Objet. Le langage Java permet également la définition d'interfaces graphiques (GUI : Graphical User Interface) facilitant le développement d'application interactives et permettant à l'utilisateur de piloter son programme dans un ordre non imposé par le logiciel. **[16]**

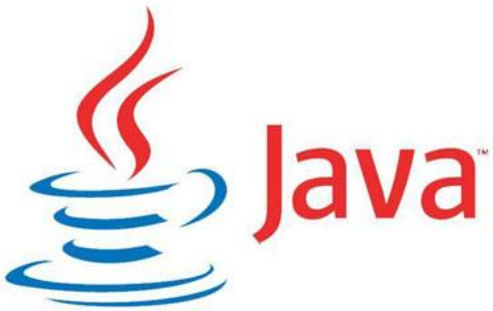

Figure 24:Java

3. JDK :

Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java. **[17]**

Il existe plusieurs éditions de JDK, selon la plate-forme Java[1] considérée (et bien évidemment la version de Java ciblée) :

- JSE pour la Java 2 Standard Edition également désignée J2SE ;
- JEE, sigle de Java Enterprise Edition également désignée J2EE ;
- JME 'Micro Edition', destinée au marché mobiles ;
- etc.

À chacune de ces plateformes correspond une base commune de Development Kits, plus des bibliothèques additionnelles spécifiques selon la plate-forme Java que le JDK cible, mais le terme de JDK est appliqué indistinctement à n'importe laquelle de ces plates-formes.

Figure 25:JDK

4. XML :

Hétérogènes (interopérabilité). Avec ses outils et langages associés, une L'Extensible Markup Language (XML, « langage de balisage extensible » en français) est un métalangage informatique de balisage générique qui dérive du SGML. Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG… Elle est reconnaissable par son usage des chevrons (<, >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche…) entre systèmes d'informations application XML respecte généralement certains principes :

- la structure d'un document XML est définie et validable par un schéma ;
- un document XML est entièrement transformable dans un autre document XML. **[18]**

Figure 26:XML

5. **StarUML :**

est un logiciel de modélisation UML, cédé comme open source par son éditeur, à la fin de son exploitation commerciale, sous une licence modifiée de GNU GPL.

L'objectif [Par qui ?] de la reprise de ce projet était de se substituer à des solutions commerciales comme IBM Rational Rose ou Borland Together.

StarUML gère la plupart des diagrammes spécifiés dans la norme UML 2.0.

StarUML est écrit en Delphi1, et dépend de composants Delphi propriétaires (non open-source), ce qui explique peut-être pourquoi il n'est plus mis à jour. **[19]**

Figure 27:StarUML

II. Les composantes applicatives réalisées :

1. L'interface de l'éditeur :

Concernant le design de l'interface, deux possibilités s'offraient à nous. Une première, Classique, avec une fenêtre principale contenant toutes fenêtres des réseaux, une autre avec un ensemble de fenêtres indépendantes. Cette dernière possibilité a été retenue car la gestion des sous-fenêtres est faite directement par Java alors, que les fenêtres principales sont gérées par le système.

Voici un schéma de l'interface :

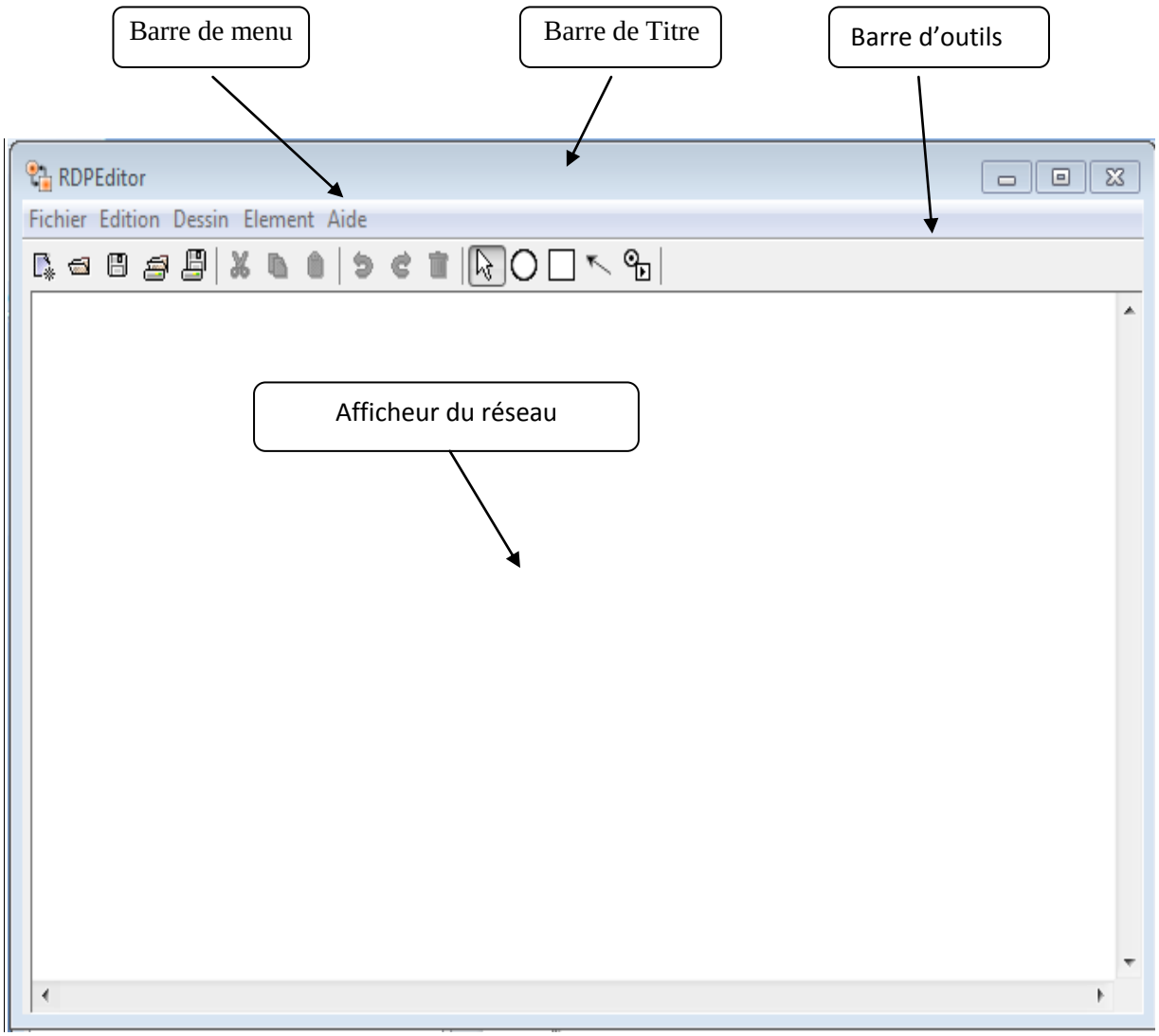

Figure 28 : Fenêtre principale

Liste des fenêtres :

a) Editeur

Cette fenêtre est la principale de l'interface elle centralise la plupart des informations et gère la liste des réseaux ouverts. Elle est composée d'une barre de menu, et d'une barre d'outils et barre graphique.

l'éditeur a été développé en Java, il s'appuie donc sur une approche objet afin que cet éditeur soit modulaire et facile à faire évoluer. De plus le Langage Java fournit toute une série de bibliothèques qui permettent de créer facilement des interfaces.

Figure 29 : Editeur

Barre de menu

Figure 30 : Barre de Menu

Cette barre de menu comporte des menus classiques comme le menu Fichier, Edition, Elément , Aide et d'autres menus comme dessin , en voici une description :

• Fichier

Ce menu possède les actions habituelles comme Nouveau, Ouvrir, Enregistrer, Enregistrer sous , importé , Exporte , Quitter . Ce menu a en plus des actions spécifiques aux réseaux de Pétri, comme Importer/Exporter un modèle générique, Générer une image XML du réseau, Compiler un réseau en Java (cette fonctionnalité est appelée à disparaître dans le futur) et gérer les librairies de Classes.

• **Edition**

Cette partie permet de Couper/Copier/Coller/Supprimer un objet

Sélectionné. Elle permet aussi d'Annuler où de Rétablir une action.

• **Dessin**

Ce menu Contient les élément de dessin Place, Transition , Arc , sélection permet de dessiner le graphe .

- **Elément**

Ce menu Contient le libellé des places et transition et les place de début et de la fin

• **Aide**

Classiquement, ce menu donne accès à l'aide du logiciel ainsi qu'a des informations le concernant.

Barre d'outils :

Figure 31: Barre d'outils

Nouveau :Permet d'ouvrir une nouvelle fenêtre avec un réseau vide.

Ouvrir : Permet d'ouvrir un réseau de Pétri précédemment enregistré.

Enregistrer : Enregistre le réseau courant dans un fichier.

Exporté : exporté le réseau ver un fichier XML

Importé : importé des fichiers XML

Couper : Coupe la sélection courante.

Copier : Copie la sélection courante.

Coller : Colle le contenu de notre presse-papier en haut à gauche du réseau.

Annuler : Annule la dernière action effectuée sur le réseau.

Rétablir : Rétablit une action annulée auparavant.

Supprimer : Supprime la sélection.

b) **Afficheur** *:*

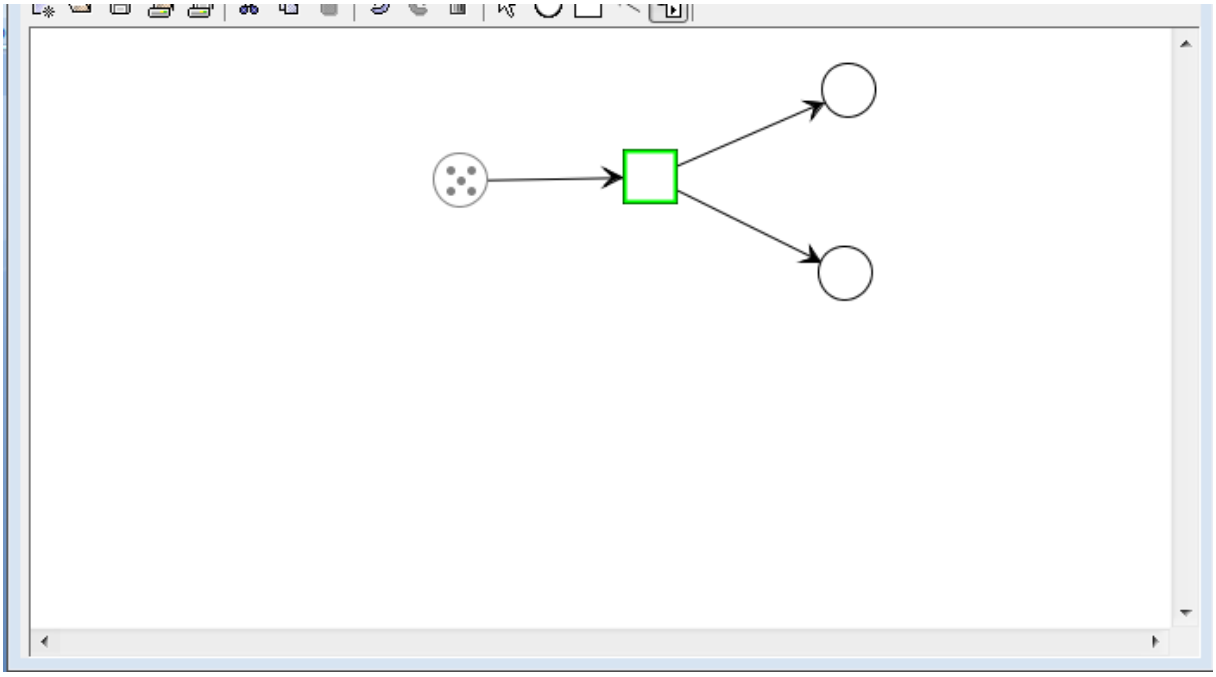

Figure 32 : Afficheur

Cette fenêtre permet, comme son nom l'indique, d'afficher un réseau via une Page.

C'est cette fenêtre qui gère les évènements claviers et souris ainsi que les menus contextuels. Pour des raisons de commodités, la Page est contenue dans une entité qui permet de faire défiler le réseau, dans le cas où celui-ci serait plus grand que l'Afficheur.

C'est la partie la plus importante d'un Afficheur, elle contient le réseau, crée les éléments de celui-ci, les listes gérant l'annulation/rétablissement d'une action. En fait elle gère quasiment tout ce qui est relatif au réseau lui-même sauf les propriétés des éléments. En résumé elle gère :

• La création des éléments graphiques du réseau.

• Le dessin du réseau.

• Les actions comme l'impression, le couper/copier/coller/supprimer, l'annulation

ou le rétablissement d'une action, la sélection d'un objet.

Barre graphique :

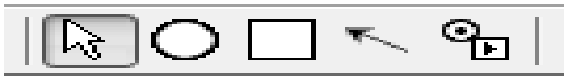

Figure 33 : schéma de la palette d'outils

Ce barre est l'un des plus utiles pour utilisateur. En effet il lui permet de créer

les éléments graphiques du réseau. On peut donc créer ainsi une Place, une Transition, un

Arc, un Sous-réseau et aussi choisir l'outil de sélection.

Conclusion

Dans cette dernière partie, nous avons présenté les différentes parties nécessaires à la réalisation de notre éditeur graphique, présentation des différents outils utilisés pour réaliser notre solution, illustration des différentes interfaces de notre application, Et enfin nous avons conclue par une comparaison entre les résultats théoriques et expérimentaux.

Conclusion Générale

Notre projet de fin d'étude consiste à concevoir et réaliser une application (éditeur graphique) pour simuler les réseaux de pétri

Pour réaliser ce travail, nous avons passé par plusieurs phases. Au début, nous avons fait une recherche sur le domaine d'étude, cette recherche s'est portée sur les généralités des réseaux de pétri et les éditeurs graphique (définition d'application graphique, son rôle, ses avantages, ses types,….), l'étude réalisée nous a permet d'avoir plus de connaissances en ce qui concerne le développement des applications graphique surtout les éditeurs graphiques.

La deuxième phase de notre projet concerne l'analyse, cette dernière a porté les méthodes de collecte des données, les anomalies devant notre système avec des suggestions.

Pour la phase relative à la conception de notre solution, nous avons fait appel au modèle UML, qui nous a permis de modéliser notre problème, par l'utilisation des différents diagrammes comme le diagramme de cas d'utilisation, et le diagramme de séquence, et le diagramme d' activité , et le diagramme de classe.

La dernière étape décrit l'implémentation de notre application, l'environnement de et les outils utilisés pour la réalisation de notre travail tout en présentant quelques captures d'écran de certaines interfaces.

Références bibliographie

[1]. (G. Scorletti et G. Binet). *« Réseaux de Petri ».* Note de cours à l'université Caen,: 4 juillet 2005.

[2]. (Robert Valette,). *« Les Réseaux de Petri»,.* notes de cours LAAS CNRS Toulouse,: Septembre 2000.

[3]. (Hocine HAMDI). *« Notions sur les réseaux de Petri»,.* , les éditions de l'université Mentouri-Constantine: 2002-2003.

[4]. (Thomas BOURDEAUD'HUY,). *«Techniques d'abstraction pour l'analyse et la synthèse de réseaux de Petri »,* . thèse pour obtenir le grade de Docteur de l'Ecole Centrale de Lille, : 13 décembre 2004.

[5]. (Nicolas RIVIÈRE). *«Modélisation et analyse temporelle par réseaux de Petri et logique linéaire»,.* thèse pour l'obtention du Doctorat de l'Institut National des Sciences Appliquées de Toulouse: , 26 Novembre 2003.

[6]. (Mohamed BOUALI,). *«Contributions à l'analyse formelle et au diagnostic à partir de réseaux de Petri colorés avec l'accessibilité arrière», .* thèse pour l'obtention du Doctorat de l'Université de Technologie de Compiègne,: 21 décembre 2009.

[7]. (s.d.). *http://www.logitheque.com/logiciels/windows/graphisme/.*

[8]. (s.d.). http://magikunicorn.over-blog.com/article-2647031.html.

[9]. (Audibert). *L. UML2. Interfaces, 3, 10.* (2007).

[10]. (Leray). *D (D'une famille de composants dialogiques à une méthodologie de synthèse de modèles d'assistance pour un Agent Conversationnel Assistant.* (Doctoral dissertation, Université Paris Sud-Paris XI): 2009).

[11]. (Roques, P., & Vallée, F.). *UML 2 en action: de l'analyse des besoins à la conception. .* Editions Eyrolles.: (2011).

[12]. (Glassey, O., & Chappelet, J. L.). *Comparaison de trois techniques de modélisation de processus: ADONIS, OSSAD et UML. IDHEAP.* Institut de hautes études en administration publique: (2002).

[13]. (Ermine, J. L., Chaillot, M., Bigeon, P., Charreton, B., & Malavieille, D. M.). *Méthode pour la gestion des connaissances. Ingénierie des systèmes d'information, AFCET-Hermès,* . 4(4), 541-575.: (1996).

[14]. (GHAFFOUR, I. S., & BENLEDGHEM, R.). *Réalisation d'une application client/serveur (gestion d'une banque) avec JDBC et MySQL selon le modèle à trois couches sous Netbeans (Doctoral dissertation).* (2014).

[15]. (Bouanani, H.). *Suivi et gestion répartie des clients d'une banque Cas de la Banque BADR (Doctoral dissertation).* (2014).

[16]. (Mahi, A). *Conception et réalisation d'un systéme de gestion de demande de formation pour une entreprise GDFE (Doctoral dissertation).* (2014).

[17]. (MAMMAR, O., & BENAOUDA, S. A. A.). *Développement d'une application Client/serveur avec Java RMI (Doctoral dissertation).* (2014).

[18]. (BENFEDEL, A., & KAZI AOUL, A.). *Développement Odoo pour la gestion de pièces de rechange, pour un client de SOGESI (Doctoral dissertation).* (2016).

[19]. (ZRAN, B.). *BULK SWS (Doctoral dissertation, Université Virtuelle de Tunis).* (2011).