République Algérienne Démocratique et Populaire
Université Abderrahmane MIRA de Béjaia
Faculté des Sciences Exactes

**Département de Recherche Opérationnelle**

Mémoire Présenté pour L'obtention du Diplôme de Master
en Mathématiques Appliquées

**Spécialité : Optimisation et Fiabilité des Réseau de Communication**

# Évaluation de Performances d'un modèle de Web services avec les Réseaux de Petri Colorés

Présenté par :
**ASSAM Slimane**

**Sous la direction de : Dr Bernine Nassima**

Défendu le 29/06/2025, devant le jury composé de :

| | | | |
|---|---|---|---|
| M$^{r}$ S. Adjabi | Professeur | Président du jury | UAMB - Bejaia |
| M$^{me}$ N. Bernine | M.C. classe/ A | Promotrice | UAMB - Bejaia |
| M$^{me}$ S. Hakmi | M.C. classe/ B | Examinatrice | UAMB - Bejaia |

**Année Universitaire** 2024 – 2025

# Remerciements

***Louange A Dieu, le miséricordieux, sans Lui rien de tout cela n'aurait pu être.***

***J**e tiens tout d'abord à remercier **M$^{me}$ N. Bernine** , pour l'honneur qu'elle m'a fait en acceptant de m' encadrer. Ces conseils précieux ont permis une bonne orientation dans la réalisation de ce modeste travail.*

***J**e tiens également à remercier les membres du jury **M$^{r}$ S. Adjabi, M$^{me}$ S. Hakmi**, pour l'honneur qu'ils m'ont fait en acceptant de juger ce travail, et d'avoir consacrer leurs temps pour sa lecture.*

***J**e tiens à exprimer mon profonde gratitude à l'ensemble du corps enseignant qui a contribué à mon formation.*

***E**nfin je tiens à rendre hommage à toute ma famille et tous mes amis et spécialement ma chére amie Kenza pour le soutien qu'ils m'ont apportés durant toutes ces années d'études.*

***ASSAM Slimane***

# Dédicace

**A** œur veillant, rien d'impossible ;
**À** conscience tranquille, tout est accessible ;
**Q**uand il y a la soif d'apprendre.
**T**out vient à point à qui sait attendre.
**L**es études sont avant tout notre unique et seul atout.
**S**ouhaitant que le fruit de nos efforts fournis jour et nuit
**N**ous mènera vers le bonheur fleuri.

***J**e Dédie Ce Travail :*

***T**out d'abord à ma chère famille, sans qui, ma vie n'aurait pas etée aussi belle, riche et réussie. Je les remercie pour leur encouragement et leur présence à mes côtés tout au long de mon parcours : mes parents, ma soeur Nayes et mes proches spécialement ma chére amie Kenza.*

***À** tous mes amis que je ne peux malheureusement pas citer tellement la liste est longue, mais merci d'avoir fait partie de ma vie et d'avoir embellie ma vie avec une amitié si sincère qui fait mon bonheur.*

***ASSAM Slimane***

# Table des matières

# Table des figures

# Liste des tableaux

# Liste d'abréviations et notations

**RDPC** : Réseaux de Petri Colorés
**SOAP** : Simple Object Access Protocol
**WSDL** : Web Services Description Language
**UDDI** : Universal Description Discovery and Integration
**XML** : Extensible Markup Language
**URI** : Uniform Resource Identifier
**IBM** : International Business Machines
**HTTP** : Hypertext Transfer Protocol
**QoS** : Qualité de Services
**DTD** : Document Type Définition
**RPC** : Remote Procedure Call
**RdPS** : Les Réseaux de Petri Stochastiques
**TCPN** : Les Réseaux de Petri Colorés et Temporisés
**UML** : Diagramme de Séquence
**ATL** : Atlas Transformation Language
**GSPN** : Réseau de Petri Stochastique Temporel Généralisé
**MDE** : Model-Driven Engineering
**PNML** : Petri Net Markup Language
**ACF** : Analyse des Concepts Formels

# Introduction générale

Dans les environnements informatiques modernes, les applications sont souvent amenées à échanger des données à travers un réseau, notamment lorsque certaines opérations nécessitent une puissance de calcul plus importante disponible sur une autre machine. Ce type de communication repose généralement sur une architecture client-serveur, dans laquelle une application cliente envoie des requêtes à un service distant chargé d'exécuter les traitements demandés.

Parmi les technologies qui permettent de faciliter cette communication entre systèmes hétérogènes, les Web services occupent une place importante. Ils représentent une évolution naturelle du Web, conçue pour simplifier les échanges de données entre entreprises et permettre aux applications de fonctionner ensemble, indépendamment des plateformes ou des langages de programmation utilisés.

Les Web services s'appuient sur des protocoles et des standards ouverts comme SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description Discovery and Integration). Ces standards assurent une interopérabilité efficace et une intégration facilitée entre systèmes distribués.

Pour garantir le bon fonctionnement d'un système basé sur les Web services et évaluer ses performances, il est nécessaire de recourir à des méthodes spécifiques comme l'analyse mathématique, la simulation ou encore la création de prototypes. Ces méthodes permettent d'obtenir des mesures quantitatives importantes telles que le temps de réponse moyen, le débit ou encore le taux de requêtes traitées.

Dans ce contexte, les réseaux de Petri offrent un cadre formel puissant pour la modélisation et l'analyse de systèmes concurrents. Grâce à leur capacité à représenter le parallélisme et à leurs propriétés analytiques, ils permettent une compréhension rigoureuse du comportement des systèmes. Les réseaux de Petri colorés (RdPC), une extension enrichie de ces modèles, ajoutent des informations supplémentaires aux jetons, rendant possible la modélisation de systèmes complexes et de grande taille.

Notre travail s'inscrit dans cette perspective en proposant un modèle de simulation basé sur les réseaux de Petri colorés pour évaluer les performances d'un système de Web services. Le modèle comprend deux files d'attente : l'une représente les Web services disponibles, et l'autre les requêtes des clients. Les performances du système sont analysées à l'aide de l'outil de simulation CPN Tools, spécialement conçu pour les réseaux de Petri colorés.

# Problématique

Dans ce travail nous nous intéressons à la modélisation, l'évaluation de performances d'un système de Web services reposant sur deux entités principales :

1. Les clients recherchent les Web services correspondant à leur besoin.
2. Les Web services, quant à eux, sont responsables de fournir les services demandés

Dans notre modélisation, nous avons pris en compte à la fois l'arrivée des clients et la disponibilité des Web services. Nous avons également spécifié qu'un client peut être pris en charge soit par un Web service simple, soit par un Web service composite. Pour représenter ce fonctionnement et analyser les performances du système, nous avons proposé un modèle basé sur les réseaux de Petri colorés (RdPC). Ce modèle permet de satisfaire différents types de clients en leur associant les Web services adaptés, qu'ils soient simples ou composites.

# Plan de mémoire

Ce mémoire est organisé en 4 chapitres :

— Chapitre 1 : On présente l'état de l'art des Web services.

— Chapitre 2 : On présente l'état de l'art les Réseaux de Petri coloré.

— Chapitre 3 : On cite certains travaux de l'existant sur l'évaluation des performances des Web services.

— Chapitre 4 : Nous présentons dans ce travail une approche de modélisation d'un système de Web services basée sur les réseaux de Petri colorés.

# 1

# État de l'art des Web services

**Sommaire**



## Introduction

Les Web services doivent leur origine à l'informatique distribuée et au développement du Web. L'accès aux systèmes d'information s'appuie aujourd'hui de plus en plus sur des technologies internet. Les efforts de standardisation dans ce contexte ont accentué l'engouement des personnes et des organisations, pour l'utilisation de l'internet et ont permis l'émergence des Web services comme support de développement des applications accessibles par Internet. L'apparition des Web services a permis la création et l'utilisation des application qui permette de réaliser un système d'information rapide et efficace distribué sur Internet.

## 1.1 Définition des Web services

On retrouve plusieurs définitions des Web services, parmi ces définitions on cite [34] :
Les Web Services sont des applications qui relient des programmes, des objets, des bases de

données ou des processus d'affaires à l'aide de XML (Extensible Markup Language) et de protocoles Internet standards. Un web service est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet, par et pour des applications ou machines, sans intervention humaine, et en temps réel [34]. Un Web Service est une application qui permet d'échanger des données avec d'autres applications Web. Même si ces dernières sont construites dans des langages de programmation différents. Parmi les Web services les plus connus on peut citer Google Maps Web service, YouTube et Spotify Web [41].

**Citation W3C (Word Wild Web Consortium)[27]**

Un Web service est un composant logiciel identifié par une URI (Uniform Resource Identifier), dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les Web services peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles internet.

**Citation Dico du Net[35]**

Une technologie permettant à des applications de dialoguer à distance via Internet indépendamment des plates-formes et des langages sur lesquels elles reposent.

**Citation IBM (International Business Machine)[36]**

Un Web service est un ensemble de fonctions d'applications associées pouvant être appelées par programme via Internet. Les entreprises peuvent associer différents Web services de manière dynamique afin d'exécuter des transactions complexes demandant un minimum de programmation. Les Web services permettent aux acheteurs et aux fournisseurs du monde entier de se trouver, d'entrer en contact de manière dynamique, et d'exécuter des transactions en temps réel, le tout avec très peu d'interaction humaine.
Les Web services sont des applications modulaires autonomes et auto-descriptives qui peuvent être publiées, localisées et appelées à travers Internet.

## 1.2 Caractéristiques des Web services

La technologie des Web services repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un Web service possède les caractéristiques suivantes [16, 38] :

1. **Les Web services sont basés sur des standards ouverts :** Le fait que le Web est constitué de plates-formes totalement hétérogènes où les intérêts des différents acteurs du marché s'entremêlent ne l'a pas empêché de se développer et d'être universel, Ce succès est dû essentiellement à l'utilisation d'un ensemble de standards ouverts, dont le plus connu est le protocole HTTP. Les Web services suivent la même approche que le Web en se basant sur des standards ouverts. Ceci permet de réduire le coût d'intégration des applications qui est essentiellement dû, au fait que les modules interactifs ont des interfaces différentes, supportent différents protocoles de communication et différents formats de données et modèles d'interactions.

2. **Les Web services sont faiblement couplés :** Un consommateur d'un Web service n'est pas directement lié à ce Web service. L'interface du Web service peut changer au fil du temps sans compromettre la capacité du client à interagir avec le service. Un système étroitement couplé implique que la logique client et serveur sont étroitement liées l'une à l'autre, ce qui implique que, si une interface change, l'autre doit être mise à jour. L'adoption d'une architecture faiblement couplée tend à rendre les systèmes logiciels plus faciles à gérer et permet une intégration plus simple entre différents systèmes.

3. **Accessibilité universelle :** Les Web services peuvent être définis, décrits et découverts sur le Web, ce qui contribue à leur accessibilité. Non seulement les utilisateurs de Web services peuvent trouver des services appropriés, mais les Web services peuvent se décrire et se publier afin qu'ils puissent se lier et interagir les uns avec les autres.

4. **Auto-contenu et auto-descriptif :** Un Web service contient dans sa description toutes les informations nécessaires à l'utilisation de ses applications.

5. **Gros Grains :** : La technologie des Web services offre un moyen naturel de définir des services à granularité grossière qui accèdent à la bonne quantité de logique métier.

6. **Possibilité d'être synchrone ou asynchrone :** : La synchronicité fait référence à la liaison du client à l'exécution du service. Dans les appels synchrones, le client est bloqué et attend que le service termine son opération avant de continuer. Les opérations asynchrones permettent à un client d'appeler un service, puis d'exécuter d'autres fonctions.

## 1.3 Architecture des Web services

### 1.3.1 Architecture de référence

Pour faciliter l'interopérabilité et l'extensibilité du paradigme des Web services. une architecture de base(référence) est nécessaire afin de préserver les objectifs initiaux visés par les Web services lors des évolutions technologiques successives[39].

**Fournisseur de service (provider) :** Le fournisseur correspond au propriétaire du service, sa tâche consiste à implémenter les fonctions du Web services, décrire l'interface de ces fonctions puis publier l'interface dans un annuaire de service, pour permettre aux consommateurs de découvrir les services.

**Le Client (requester) :** Il correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service.

**L'annuaire des services (Service registry) :** Il correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de la recherche des services.
La (Figure I.1) montre l'architecture de référence de Web service :

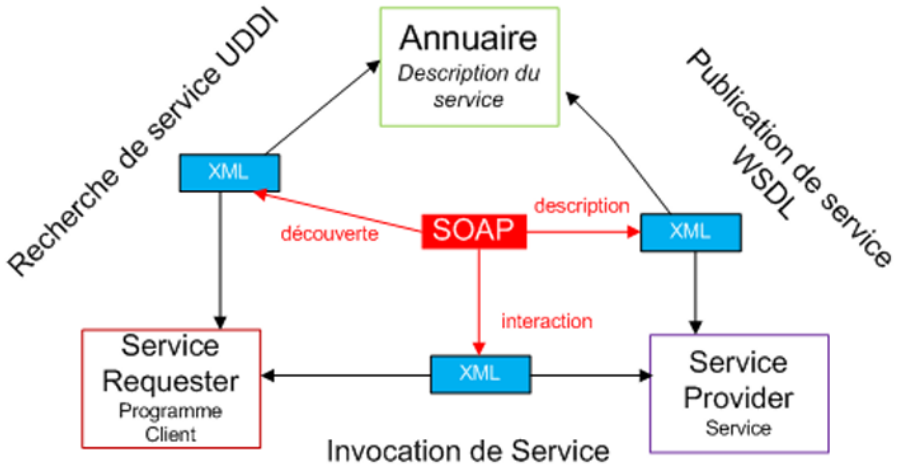



FIGURE 1.1 – Scénario d'utilisation des Web services par les acteurs client/fournisseur

### 1.3.2 Architecture étendue :

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de pile des Web services. Les fonctions de couches supérieurs reposent sur celles de couches inférieurs.
L'infrastructure de base des Web services seule, ne répond pas à tout les problèmes d'intégration. Elle est complétée par d'autres couches :
**La couche Business processus :** permet d'utiliser les Web services dans le domaine du e-business d'une manière effective. Elle représente un business processus comme un ensemble de Web services.
**Les couches transversales :** contiennent un ensemble de standards qui rendent fiable l'utilisation effective des Web services dans le monde industriel (sécurité, administration, transactions et qualité de services (QoS)).
**La couche Transport :** Assure la connectivité physique.
La (Figure 1.2) présente l'architecture étendue :

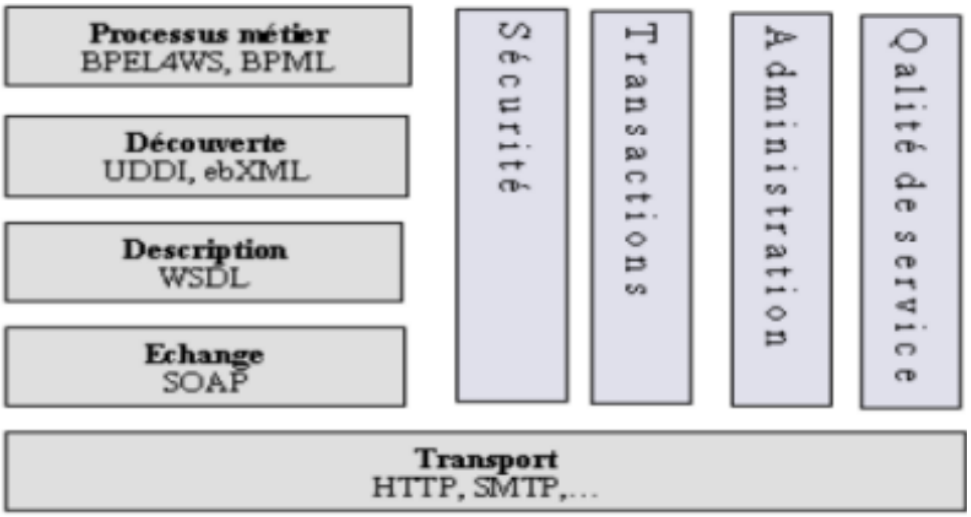



FIGURE 1.2 – Architecture étendu

## 1.4 Les principales technologies des Web services

Le terme technologies de Web services désigne un ensemble de technologies basées sur des standards ouverts (non propriétaires), essentielles pour le transport et la transformation des données d'un client vers un service d'une façon standard. Les plus courants sont : XML, SOAP, WSDL, UDDI.

### 1.4.1 XML (Extensible Markup Language)

XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet, la technologie des Web services est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type). L'objectif initial de XML est de faciliter l'échange automatisé de contenus complexes (arbres, texte enrichi, etc) entre systèmes d'informations hétérogènes (interopérabilité)[43].

La norme XML comporte deux parties : **XML à proprement parler** et **les DTD (Document Type Dénition)** qui définissent les balises qui sont utilisées dans une famille de documents XML.
XML a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes [42] :

— La lisibilité à la fois par les machines et par les utilisateurs.
— La définition sans ambiguïté du contenu d'un document.
— La définition sans ambiguïté de la structure d'un document.
— La séparation entre documents et relations entre documents.
— La séparation entre structure du document et présentation du document.

### 1.4.2 SOAP (Simple Object Access Protocl)

SOAP est un protocole de la famille XML servant à l'échange d'informations dans un environnement distribué et décentralisé. Il est considéré comme la technologie la plus importante des Web Services. Le standard SOAP a été proposé au W3C par Microsoft, IBM, Lotus, DevelopMentor et Userland, Ce protocole peut appeler des méthodes RPC (Remote Procedure Call) et envoyer des messages à des machines distantes via HTTP, à l'aide de SOAP les applications peuvent converser entre elles et échanger des données sur Internet, quelles que soient les plates-formes sur lesquelles elles s'exécutent[37, 44].
La structure d'un message SOAP :

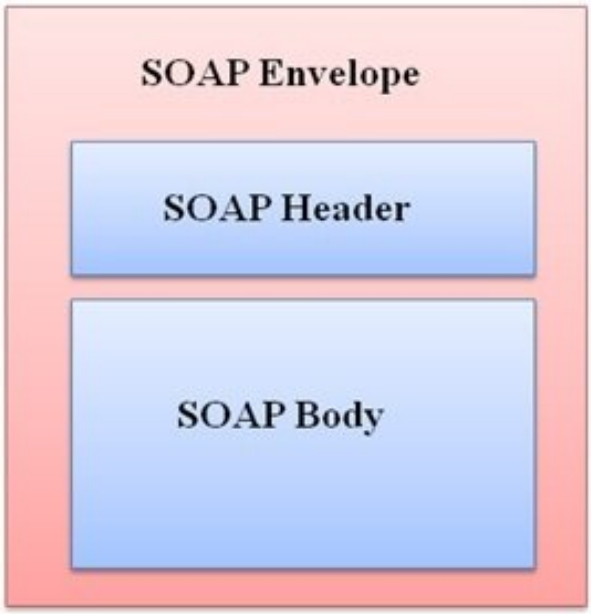



FIGURE 1.3 – Structure d'un message SOAP

**L'enveloppe SOAP (<Envelope>) :** est l'élément racine de chaque message SOAP, il contient deux éléments enfants, un élément facultatif <Header> et un élément obligatoire <Body>.

**L'en-tête SOAP (l'élément <Header>) :** est un sous-élément facultatif de l'enveloppe SOAP, il est utilisé pour transmettre les informations relatives à l'application qui sont traitées par les nœuds SOAP le long du flux de messages.

**Le corps SOAP (l'élément <Body>) :** est un sous-élément obligatoire de l'enveloppe SOAP, qui contient des informations destinées au destinataire final du message.

### 1.4.3 UDDI (Universal Description Discovery and Integration)

UDDI est une spécification définissant la manière de publier et de découvrir les Web services sur le réseau. L'idée de l'UDDI est de normaliser le format des entrées d'entreprises et de services dans un annuaire pour faciliter la découverte des Web services et encourager les

échanges d'affaires entre elles.

**Rôle de UDDI :**

Les informations dans l'UDDI sont divisés en trois catégories :

— **Pages jaunes :** Elles indiquent ce que fait le service.

— **Pages blanches :** Elles donnent des informations sur le fournisseur du service.

— **Pages vertes :** Elles donnent des détails sur comment utiliser techniquement le Web service.

Le registre UDDI utilise le protocole SOAP ainsi que le protocole HTTP, et suit un modèle d'échange de données fondé sur la notation XML.

**Langage de Description**

L'information stockée dans le registre UDDI correspond à la (Figure 1.4) et comprend les quatre types d'informations suivantes :

— **businessEntity :** Qui donne les informations sur l'entreprise offrant le service.

— **businessService :** Qui donne les informations sur le service offert.

— **bindingTemplate :** Qui donne l'information permettant d'utiliser un Web service particulier.

— **tModel :** (technical model) Qui représente toute sorte d'information. Cela peut être une interface de service, une classification, la sémantique d'une opération,... .

Le tModel, grâce à ses liens avec les autres éléments, est réellement le coeur des spécifications UDDI. Le contenu du tModel peut représenter tout type d'information et peut être écrit dans n'importe quel langage. Le tModel est référencé par sa clé unique.

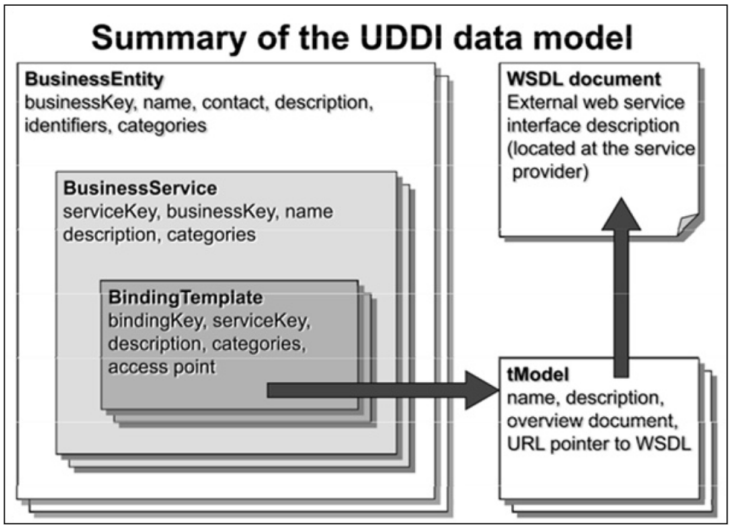



FIGURE 1.4 – Structure de données d'UDDI

### 1.4.4 WSDL (Web Service Definition Language)

WSDL est un langage de description de Web services sous format XML. Il permet de définir la structure des objets d'une manière abstraite et indépendante du langage de développement, et cela quels que soient le protocole ou le codage utilisé par des objets spécifiques à chaque système[45].
Un fichier WSDL comporte les éléments suivants :

- Définitions abstraites des données qu'on souhaite transmettre.
- Informations de type adresse afin de localiser le service à spécifier.
- Informations sur le type de données.
- Informations sur toutes les fonctions disponibles publiquement.
- Informations obligatoires sur le protocole de transport qu'il faut spécifiquement utiliser.

La puissance de WSDL découle de deux architectures principales : la capacité de décrire un ensemble d'opérations commerciales et la capacité de séparer la description en deux unités de base.
Ces unités sont la description abstraite et la description concrète.

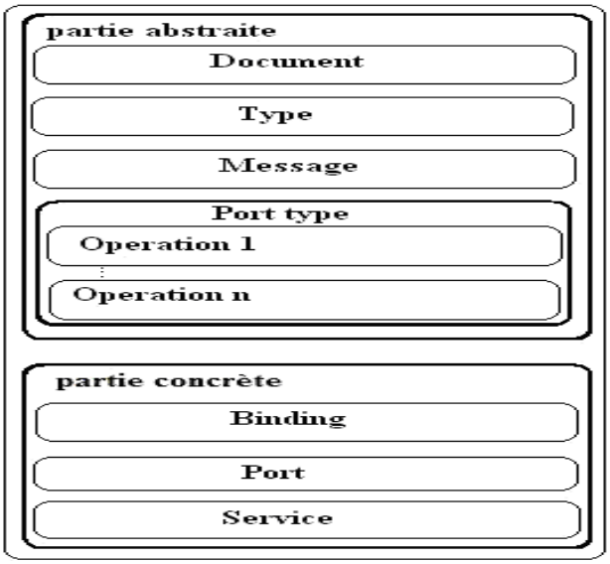



FIGURE 1.5 – Structure d'un document WSDL

**Description de la partie abstraite :**

Définir les éléments de l'interface de service tel que les types de données, les messages, les types de ports. Ces parties décrivent des informations abstraites indépendantes au contexte de mise en œuvre.
Elle est composée de six éléments :
Définition, Types, Message, Portype, Import et Document.

- **Définition :** Cet élément est considéré comme la racine du document WSDL. Il définit le nom du Web service et déclare les différents espaces de nom utilisés dans les documents.

— **Type :** Cet élément contient des types de données importants pour les messages échangés, pour un maximum d'interopérabilité et une neutralité de la plateforme.

— **Message :** Il représente une définition abstraite des données transmises. Il peut être un message d'entrée ou un message de sortie d'un Web service.

— **PortType :** Un Web service peut contenir plusieurs portTypes. Chaque portType définit un ensemble d'opérations abstraites offertes par le Web service. Un portType est identifié par un nom et contient une opération au moins. Chaque opération a un nom et contient au plus un message d'entrée qui est défini par l'élément "Input" et au plus un message de sortie qui est défini par l'élément "Output". En cas d'exception, l'élément 'Fault' est présent et définit un message d'erreur retourné.

— **Import :** On l'utilise pour importer d'autres documents WSDL, en associant un espace de nom à la location d'un document.

— **Document :** Cet élément est facultatif, le document WSDL : permet de décrire des commentaires.

**Description de la partie concrète :**

Elle comporte deux éléments : liaison et service.

— **Liaison :** Il définit le format d'un message et les protocoles de communication, pour des opérations et des messages définis par un portType. Un fichier WSDL peut contenir plusieurs éléments 'Binding', dont chacun est identifié par un nom et fait référence à un portType.

— **Service :** Regroupe un ensemble de ports reliés entre eux. Il a un nom et contient un ou plusieurs ports. Chaque élément "port" a un nom et définit un point d'accès au Web service.

## 1.5 Composition des Web services

Des différentes définitions ont été proposées pour le concept de composition, nous citons quelqu'unes :

La composition des Web services c'est la capacité d'offrir des services à valeur ajoutée en combinant des services existants probablement offerts par différentes organisations.

La composition des Web services peut être définie comme le processus de sélection, de combinaison et d'exécution de services en vue d'accomplir un objectif donné[6].

### 1.5.1 Types de composition

La composition peut être classifiée en cinq catégories [40] :

— **Composition manuelle :** C'est à l'utilisateur de programmer et d'implémenter la composition.

— **Composition semi-automatique :** Elle fournit des suggestions sémantiques pour aider à la sélection des Web services dans le processus de composition.

— **Composition automatique :** prend en charge tout le processus de composition sans aucune intervention de l'utilisateur ne soit requise. Cependant, la composition automatique a été toujours une tâche difficile à réaliser.

— **Composition statique (offline) :** Une telle composition est adaptée pour les environnements fermés où les composants n'évoluent pas souvent. Ce type de composition engendre des applications peu flexibles, parfois inappropriées avec les exigences des clients. Microsoft Biztalk est l'un des moteurs de composition statiques de Web services.

— **Composition dynamique (on-line) :** Elle permet de créer de manière autonome des services complexes en combinant des composants à la volée en fonction des demandes de l'utilisateur et du contexte[10]. Elle évolue dans des environnements flexibles et ouverts, où la sélection et la combinaison des composants sont effectuées à la demande.

## 1.6 Avantages et Inconvénients des Web services

### 1.6.1 Avantages des Web services

— Les Web services permettent des programmes écrits dans différentes langues et sur des plateformes différentes de communiquer entre elles-mêmes.

— Les Web services utilisent des normes et des protocoles ouverts tel que SOAP et HTTP.

— Les outils de développement, basés sur ces standards, permettent la création automatique de programmes utilisant les Web services existants.

— Les services sont conçus de façon à ce qu'ils puissent être réutilisés ultérieurement pour minimiser la redondance et rendre le service réutilisable par les différentes applications du système d'information.

### 1.6.2 Inconvénients des Web services

— Le manque de la sémantique.

— Le langage de la description (WSDL) est classique.

## Conclusion

Un Web service met à disposition un service via Internet. Il constitue ainsi une interface permettant à deux applications de communiquer. Pour y parvenir, la technologie doit disposer de deux propriétés essentielles être multiplate-forme et être partagée, dans ce chapitre nous avons établit une étude de l'état de l'art des Web services. Dans le chapitre suivant, nous aborderons un outil formel fondamental pour la modélisation et l'analyse des systèmes concurrents : Les Réseaux de Petri.

# 2

# Les Réseaux de Petri

**Sommaire**



## Introduction

La théorie des réseaux de Petri (RdP) a été développée à partir de la thèse de doctorat de Carl Adam Petri en 1962. Un réseau de Petri est un modèle abstrait et formel de flux d'informations. Les propriétés, les concepts et les techniques des RdP sont développés dans une recherche de ressources naturelles, des méthodes simples et puissantes pour décrire et analyser le flux d'informations et de contrôle dans les systèmes, en particulier les systèmes qui peuvent présenter des activités asynchrones et simultanées. La principale utilisation des réseaux de Petri a été la modélisation des systèmes d'événements dans lesquels il est possible que certains événements se produisent simultanément, c'est un outil de modélisation très puissant qui propose une modélisation statique et dynamique.

— Modélisation statique, qui décrit l'architecture du système.

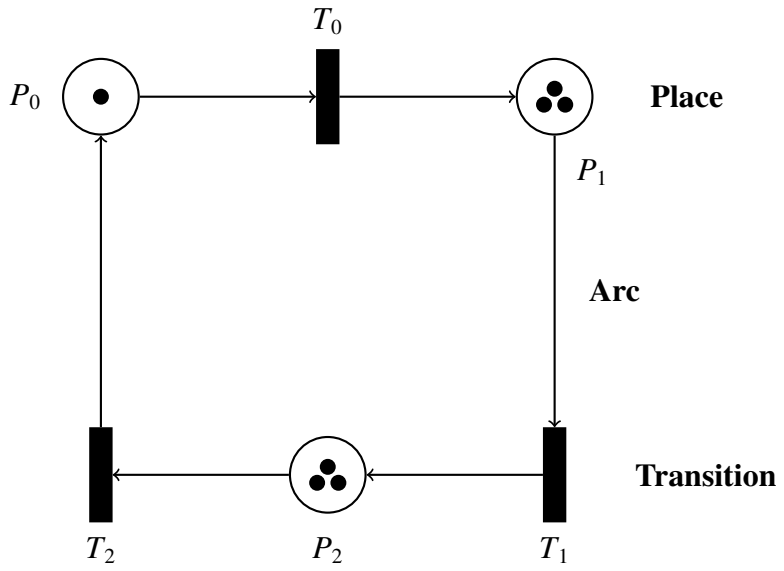
— Modélisation dynamique, qui décrit les comportements possibles du système.

## 2.1 Notion de base

**Définition 2.1.** *Un réseau de Petri est un graphe biparti, dont on particularise les deux familles de sommets : les places, les transitions qui sont reliés entre eux par les arcs. La façon dont ces transitions sont reliée définit le comportement d'un RdP.*

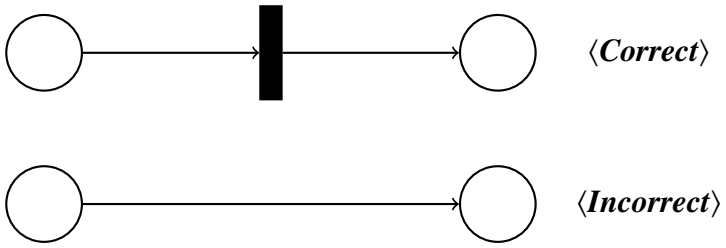
- *Des places représentées par des cercles et peuvent être marquées par une ou plusieurs marques appelées jetons.*
- *Des transitions représentées par des rectangles ou des traits, sous certaines conditions sur le marquage du réseau.*
- *Les arcs sont représentés par des flèches qui lient une place à une transition ou inversement une transition à une place.*

**Définition 2.2.** *D'une manière formelle un Rdp non marqué est un quadruplet R = (P,T,Pre,Post) tel que :*
$P = \{p_1, p_2, ..., p_n\}$ *: est un ensemble fini de places.*
$T = \{t_1, t_2, ..., t_m\}$ *: est un ensemble fini de transitions.*
$Pre : P * T \rightarrow N$ *: est l'application d'incidence avant (places précédents) ;*
$Post : P * T \rightarrow N$ *: est l'application d'incidence arrière (places suivantes) ;*
*N : est l'ensemble des entiers naturels.*
$P \cap T = \varnothing, \quad P \cup T \neq \varnothing$




**Exemple d'un réseau de Petri**

*Remarque* 1. *Comme dans tout graphe biparti, un arc ne relie jamais deux sommets de la même famille, comme le montre l'exemple en bas. C'est aussi pareil pour les transitions.*




*Si un arc n'a pas de poids associé (appelé aussi valuation), c'est que ce poids vaut 1, sinon l'arc est marqué et une valuation lui est a attribuée.*

### 2.1.1 Matrice d'incidence

La matrice d'un RdP est la matrice d'incidence entière C, elle traduit le coût global du franchissement d'une transition pour chaque place.

— La matrice d'incidence du réseau est :

$$\mathbf{C = Post - Pre}$$

— Matrice d'incidence avant : $Pre(p_i, t_j)$ indique le nombre de marquages qui doit contenir la place $p_i$, pour que la transition $t_j$ soit franchissable.

$$\boldsymbol{C^- = [C^-_{i,j}] \quad où \quad C^-_{i,j} = Pre(p_i, t_j)}$$

— Matrice d'incidence arrière : $Post(p_i, t_j)$ indique le nombre de marquages déposés dans la place $p_i$ aprés le franchissement de la transition $t_j$.

$$\boldsymbol{C^+ = [C^+_{i,j}] \quad où \quad C^+_{i,j} = Post(p_i, t_j)}$$

Ainsi la matrice $C^-$ fait le bilan des liaisons amont aux transitions, et la matrice $C^+$ fait le bilan des liaisons avant les transitions.

### 2.1.2 Notations matricielles

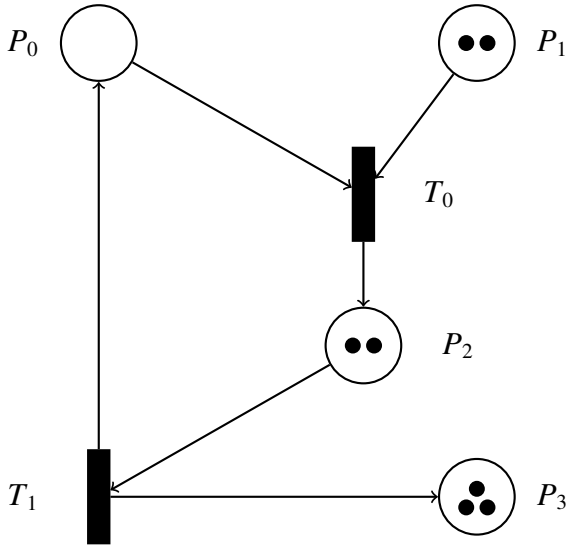
• Un arc relie une place p à uneè transition t, si et seulement si $Pre(p, t) \neq 0$. Un arc relie une transition t à une place P, si et seulement si $Post(p, t) \neq 0$. Les valeurs non nulles des matrices Pre et Post sont associées aux arcs comme étiquettes[17].

• Le réseau de Petri est initialement graphique, peu exploitable sous cet aspect malgré la convivialité qu'il apporte, il peut être transcrit algébriquement. "C" est une matrice d'incidence

qui va être décrite, fait la synthèse de tous les liens entre places et transitions du RdP.

• Cette matrice “C“ en générale possède un nombre de colonne 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.

**Exemple :** Soit le RdP suivante :
Où tous les arcs sont de poids 1, on a :




**Un exemple d'un réseau de Petri**

Matrice d'incidence arrière $C^+$ (Post), matrice d'incidence avant $C^-$ (Pré) et la matrice d'incidence C :

$$C^+ = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}; \qquad C^- = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}; \qquad C = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}$$

## 2.2 Graphe associé pour un RDP

Un RdP peut se présenter par un graphe biparti (c'est un graphe dont les sommets se répartissent en deux ensembles disjoints P et T).

**Définition 2.3.** ***Graphe associé****[3] :*

*Le graphe G =< P,T,L,V > biparti associé au RdP R =< P,T,Pre,Post > est définit comme suit :*

$$\begin{cases} L(p) = \{t \in T \mid Pre(p,t) > 0\} \\ L(t) = \{p \in P \mid Post(p,t) > 0\} \end{cases}$$

*Où L permet d'obtenir les successeurs d'une place ou d'une transition.*

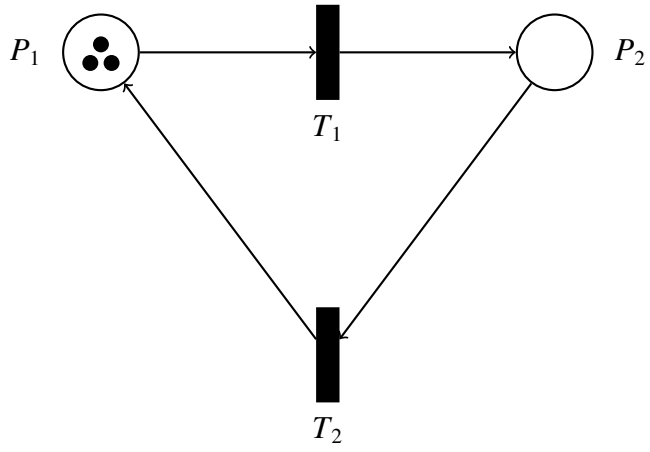
## 2.3 Rdp marqué

Le marquage d'un RdP est un vecteur de dimension n (n : nombre de places du réseau) à composantes entières positives ou nulles[22].
Un Réseau Marqué R est un couple (R,M) constitué d'un réseau de Petri R et d'une application de marquage définie sur P et à valeurs dans N (ie le marquage du réseau à un instant donné). Un RdP marqué est caractérisé par un couple A = (R,$M_0$) où :

- R : est un RdP.
- $M_0$ : est le marquage initial qui est une application définie par :

$$\begin{cases} M_0 : p \rightarrow N \\ P \rightarrow M_0(P) \end{cases}$$

**Exemple :**




**Un RdP marqué**

Le marquage initial : $\boldsymbol{M_0} = (\boldsymbol{M_0(p_1)}; \boldsymbol{M_0(p_2)}) = (\mathbf{3; 0})$

*Remarque* 2. *Un marquage peut être représenté sous forme d'un vecteur colonne*

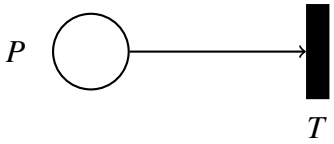
$$\boldsymbol{M_0} = (\boldsymbol{M_0(p_1)}; \boldsymbol{M_0(p_2)})^T = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

*Le marquage initiale $M_0$ d'un RdP correspond à la distribution initiale des jetons dans chacune de places du RdP qui précise l'état initial du système. Dans ce cas, on parle du RdP marqué.*

**Définition (Transition source) :**
Une transition source est une transition qui ne comporte aucune place d'entrée.


**Un RDP avec une Transition source**

**Définition 2.4.** ***Transition puits :***
*Une transition puits est une transition qui ne comporte aucune place de sortie.*




***Un RDP avec une Transition puits***

**Définition 2.5.** ***Tir de transition :***
*Une transition t est tirable (ou franchissable, ou validée) lorsque[29] :*

$$\forall \boldsymbol{p} \in \boldsymbol{P}, \boldsymbol{M(p)} \geq \boldsymbol{Pre(p, t)}$$

*Est une transition validée dans le marquage M.*

## 2.4 Évolution d'un RdP

L'évolution d'état du réseau de Petri correspond à une évolution du marquage, les jetons qui indiquent l'état du réseau à un instant donné, peuvent passer d'une place à l'autre par franchissement, ou par un tir d'une transition. Dans le cas des réseaux dites à arcs simples ou de poids égale à 1, le franchissement d'une transition consiste à retirer 1 jeton dans chacune des places d'entrée de la transition, et à ajouter un, dans chacune des places de sorties.
En générale l'évolution des états d'un réseau de Petri marqué simple obéit aux règles suivantes[13] :

— Une transition est franchissable ou sensibilisée ou encore tirable lorsque chacune des places d'entrées, possède au moins le nombre de jetons correspondant au poids de l'arc reliant à la transition.
— Le réseau ne peut évoluer que par franchissement d'une seule transition à la fois transition sélectionnée parmi toutes celles validées au moment du choix.
— Le franchissement d'une transition est indivisible et de durée nulle.

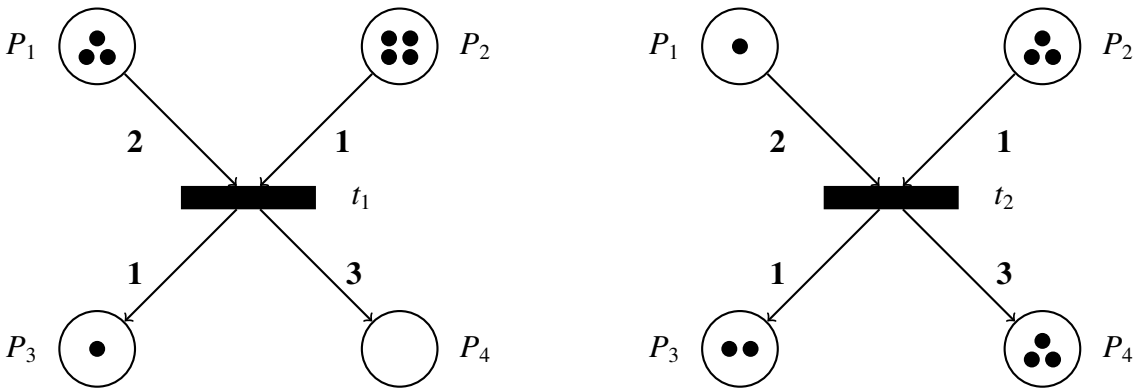
**Définition 2.6.** ***Évaluation de marquage [29] :***

*Soit $A = (R, M_0)$ un réseau de Petri marqué, de transitions T et de places P. Le franchissement d'une transition t de T validée dans le marquage $M_0$ conduit au marquage $M_1$ :*

— $\forall \boldsymbol{p} \in P, \forall \boldsymbol{t} \in T, \quad \boldsymbol{M_1(p) = M(p) + C(p,t)}$
— $\forall \boldsymbol{p} \in P, \forall \boldsymbol{t} \in T, \quad \boldsymbol{M_1(p) = M(p) + Post(p,t) - Pre(p,t)}$

*On note alors $M[t > M_1$.*

**Exemple :**




RDP Avant Franchissement RDP Aprés Franchissement

Les marquages sont les suivants :
Avant franchissement : $M(P_1) = 3$ ; $M(P_2) = 4$ ; $M(P_3) = 1$ ; $M(P_4) = 0$.
Après franchissement : $M(P_1) = 1$ ; $M(P_2) = 3$ ; $M(P_3) = 2$ ; $M(P_4) = 3$.

*Remarque* 3. *Le franchissement d'une transition ne garantit pas la conservation de la quantité des marques globales. Dans l'exemple précédent, on a globalement un jeton de plus après franchissement de la transition. Selon les poids attribués aux arcs liés à une transition donnée, les transitions sont :* ***"Consommatrice"****,* ***"Génératrice"*** *ou* ***"Conservatrice"*** *de marques.*

— ***Transition Consommatrice*** *: Une transition est dite consommatrice si, lorsqu'elle se déclenche, elle diminue le nombre total de jetons dans le réseau.*

— ***Transition Génératrice*** *: Une transition est génératrice si, lorsqu'elle se déclenche, elle augmente le nombre total de jetons dans le réseau.*

— ***Transition Conservatrice*** *: Une transition est conservatrice si le nombre de jetons reste inchangé après son franchissement.*

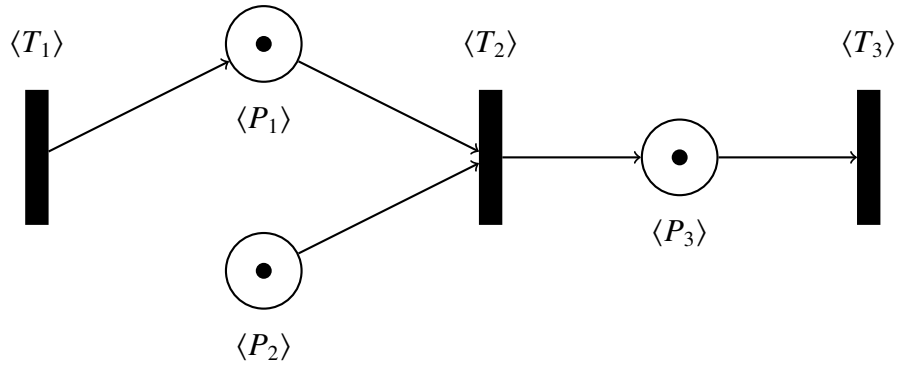
### 2.4.1 Franchissement d'une transition

Si t est franchissable pour le marquage M, le franchissement (tir, firing) de t donne le nouveau marquage M′ tel que [21] :

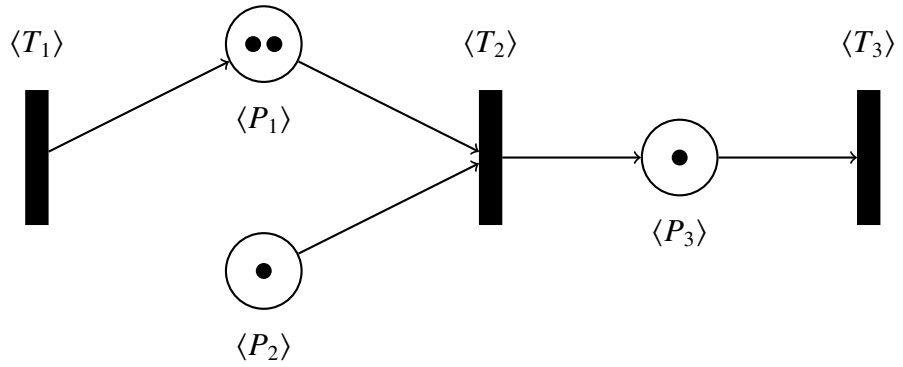
$$(\forall p \in P) M'(p) = M(p) - Pre(p,t) + Post(p,t)$$

**Définition 2.7.** ***Franchissement d'une transition source :***

*Une transition source est une transition qui ne comporte aucune place d'entrée, c'est une transition toujours franchissable et le franchissement a lieu lorsque l'événement associé se produit.*

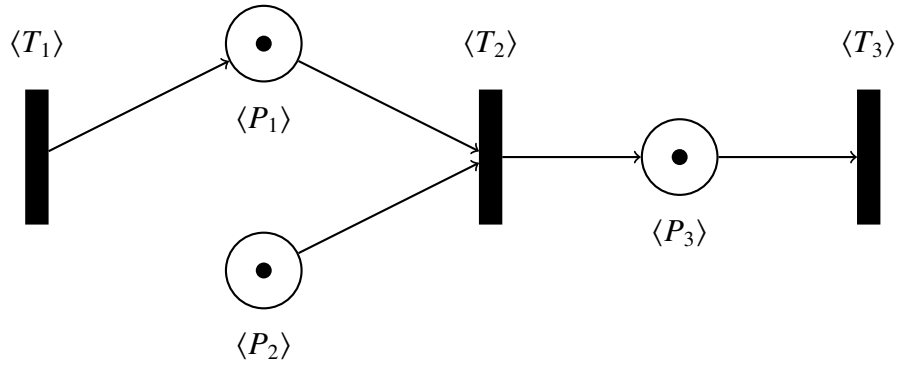



Aprés avoir tirer la transition source "T1", nous aurons ce résultat :

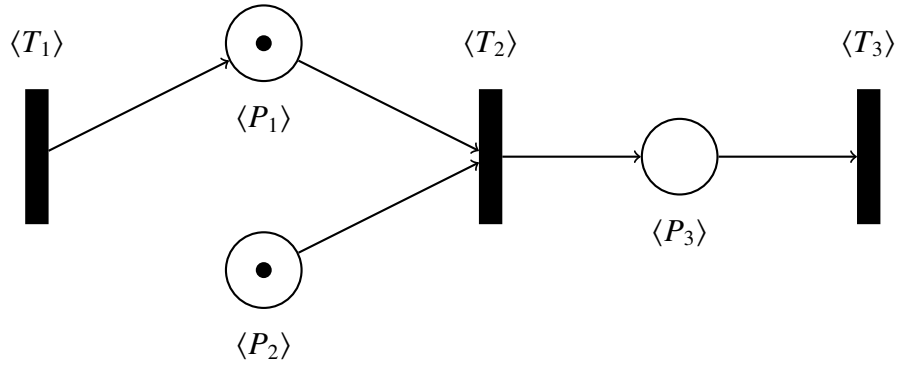



**Définition 2.8.** ***Franchissement d'une Transition puits :***

*Une transition puits est une transition qui ne comporte aucune place de sortie, le franchissement d'une transition puits enlève des jetons de toutes les places d'entrée de la transition.*




Aprés avoir tirer la transition source "T3", nous aurons ce résultat :




### 2.4.2 Conflit et parallélisme

L'avantage des RdP réside dans leur capacité de modéliser un grand nombre de comportements dans les systèmes complexes. Parmi ces comportements, nous trouvons le parallélisme,

la synchronisation, le partage de ressources, les conflits, etc.

**Définition 2.9.** ***Le parallélisme :***

*Le parallélisme est défini comme l'évolution simultanée de plusieurs processus dans un même système. Dans un RdP, le parallélisme est déclenché avec une transition ayant plusieurs places de sortie[22].*

1. **Parallélisme structurel :** Deux transitions $t_1$ et $t_2$ sont parallèles structurellement si :

$$\boldsymbol{Pre(., t_1)^T \times \ Pre(., t_2) = 0}$$

   Elles n'ont donc aucune place d'entrée commune (le produit scalaire de leurs vecteurs Pre soit nul).

2. **Parallélisme effectif :** Deux transition $t_1$ et $t_2$ sont parallèles pour un marquage donné M si et seulement si elles sont parallèles structurellement et [31] :

$$\boldsymbol{M(p) \geq Pre(p, t_1)}$$

$$\boldsymbol{M(p) \geq Pre(p, t_2)}$$

**Définition 2.10.** ***Conflit [31, 25] :***

*Le conflit est l'existence d'une place qui a au moins deux transitions de sortie. La notion de conflit modélise un choix ou une décision. Deux sortes de conflit existent.*

1. **Conflit structurel :** Une place $p_i \in P$ constitue un conflit structurel si elle est une place d'entrée pour au moins deux transitions.

$$\boldsymbol{\exists p \quad Pre(p, t_1) \cdot Pre(p, t_2) \neq 0}$$

2. **Conflit effectif :** On parle d'un conflit effectif entre deux transitions en conflit structurel s'il existe un marquage qui sensibilise les deux transitions et que le franchissement d'une transition empêche le franchissement de l'autre une seule transition sera franchie[31].

$$\boldsymbol{M(p) \geq Pre(p, t_1)}$$

$$\boldsymbol{M(p) \geq Pre(p, t_2)}$$

## 2.5 Séquence de franchissement

**Définition 2.11.** ***Définition informelle [29] :***

*Le franchissement successif de transitions (sensibilisées) dans un ordre donné, à partir d'un marquage donné constitue une séquence de franchissements.*

- *On s'intéresse à l'évolution du réseau, lors du tir successif de plusieurs transitions.*
- *Lorsque $M[t_1/t_2 > M_2$, on dit que la séquence de transitions $t_1t_2$ est franchissable depuis le marquage M.*
- *On note $M[t_1t_2 > M_2$.*

*Une séquence de franchissement est un mot construit sur l'alphabet T′ des transitions de T . On note "s" une séquence de franchissement.*

**Définition 2.12.** ***Définition formelle [2] :***

- *Soit (R,$M_0$) : un RdP marqué.*
- *$s = t_1, t_2, ..., t_n \in T'$ : une séquence de transitions.*
- *La séquence s est franchissable depuis M, si et seulement s'il existe des marquages $M_1, M_2, ..., M_n$, tels que*

$$\boldsymbol{M_1[t_1 > M_2[t_2 > ...M_{n-1}[t_{n-1} > M_n}$$

*Avec T′ est un sous ensemble de T constitué des transitions qui forment la séquence de franchissement. Dans ce cas le tir ou le franchissement conduit au marquage $M_n$, on note alors $M(s > M_n$ [2].*

**Définition 2.13.** ***Équation d'état :***

*Un RdP possède un état d'accueil $M_a$ pour un marquage initial $M_0$, si pour tout marquage accessible $M_i$, il existe une séquence de tirs s telle que $M_i[s > M_a$, il s'en suit qu'un RdP est réinitialisable pour un marquage initial $M_0$ si $M_0$ est un état d'accueil.*

*Remarque* 4. *Il s'agit d'une condition nécessaire mais pas suffisante : il se pourrait que ne soit pas franchissable depuis M.*

*Deux étapes pour calculer un marquage :*

1. *Démontrer que le marquage valide la séquence ;*
2. *Calculer le marquage résultat.*

**Définition 2.14.** ***Marquage accessible [21] :***

*Franchissement d'une transition validée dans un RdP apporte une modification au marquage initial $M_0$. Un marquage $M_n$ est dit accessible à partir du marquage initial $M_0$, s'il existe une séquence de franchissement $s = T1, T_2, ..., T_n$ qui transforme $M_0$ en $M_n$. Dans ce cas on écrit :*

$$\boldsymbol{M_0[s > M_n}$$

*qui signifie que $M_n$ est accessible à partir $M_0$ par s.*
• *Ensemble de toutes les marques accessible à partir de $M_0$ dans un RdP.*
• *Ensemble de toutes les séquences de franchissement possibles à partir de $M_0$ est noté $L(M_0)$.*

$$\boldsymbol{M_i[s > M_k}$$

**Définition 2.15.** ***Ensemble d'accessibilité [25] :***

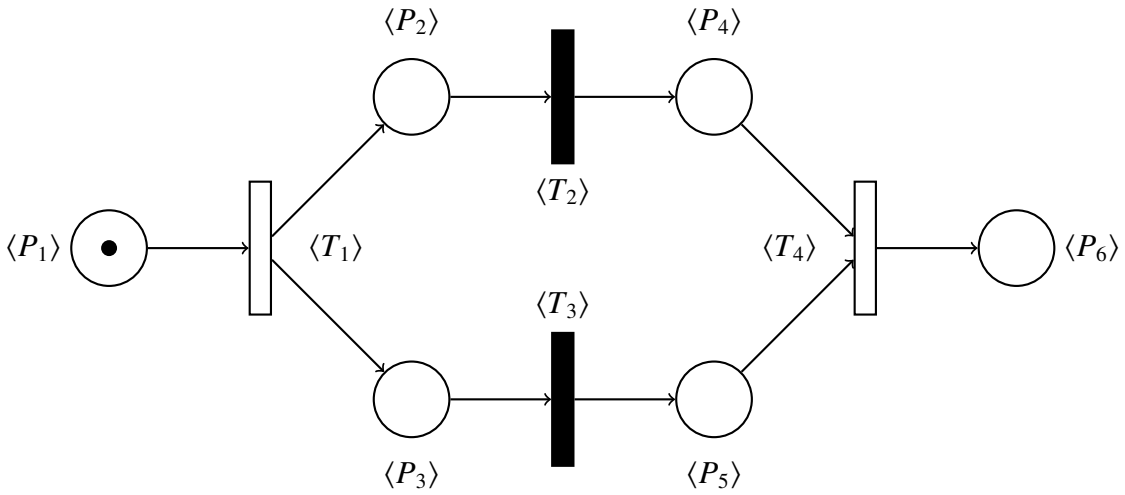
*Soit un RdP. L'ensemble des marquages accessibles ou ensemble d'accessibilité de ce réseau est noté A(R,$M_0$) ou A, est l'ensemble des marquages atteints par une séquence de franchissement.*

**Définition 2.16.** ***Graphe des marquages accessibles :***
*L'ensemble des marquages accessibles $A(R, M_o)$ est défini par [29] :*

$$\begin{cases} \boldsymbol{M_0 \in A(R, M_0)} \\ \textit{si } \boldsymbol{M \in A(R, M_0)} \textit{ et } \boldsymbol{M|t > M'} \textit{ alors } \boldsymbol{M' \in A(R, M_0).} \end{cases}$$

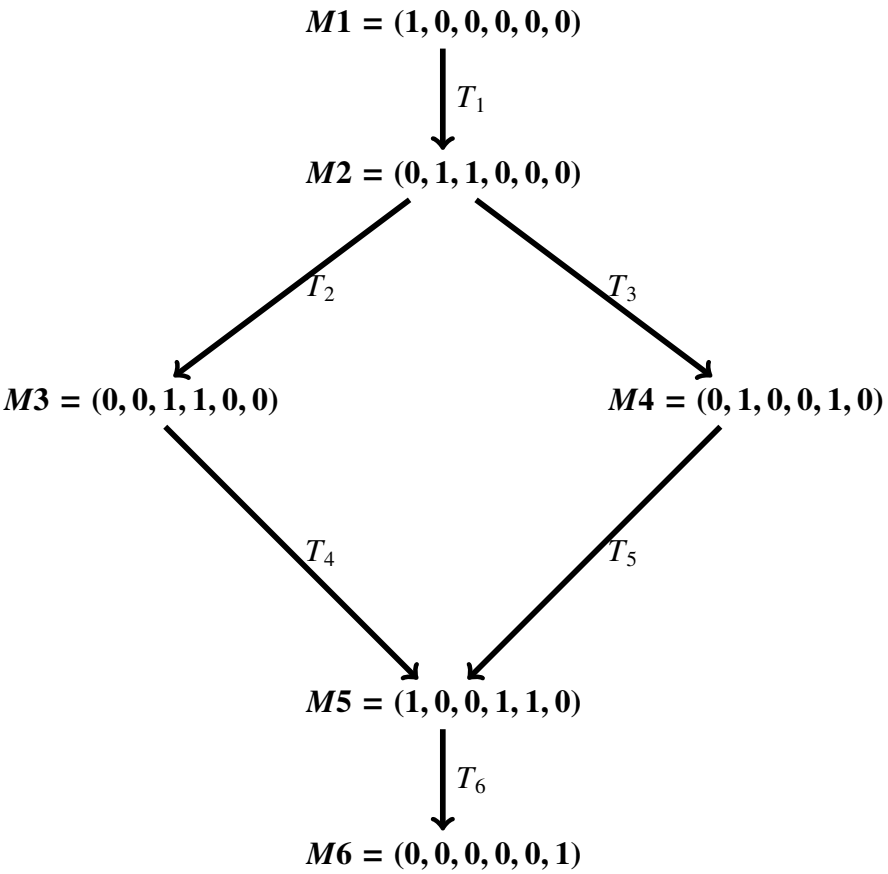
*Cette définition signifie que le marquage initial est un marquage accessible et tout marquage successeur d'un marquage accessible est un marquage accessible.*

**Exemple Graphe de marquage :**
Soit le réseau Réseau de petri marqué suivant :




Prenant le réseau de Petri au dessus, l'ensemble des marquages accessibles sont $\{M_2, M_3, M_4, M_5, M_6\}$. Le graphe de marquages est représenté par le graphe en bas.

**Graphe de marquage :**




## 2.6 Propriétés d'un réseau de Petri

Le modèle de RdP nous donne des techniques pour analyser les propriétés qui sont les caractéristiques qui permettent d'évaluer la qualité d'un système donné. Elle peuvent être classées en deux groupes.

### 2.6.1 Les propriétés dynamiques

Ces propriétés dépendent à la fois du marquage initial $M_0$ et de la structure du réseau[9].

1. **Réseau borné :**

   - La bornitude d'un RdP exprime le fait que le nombre d'états que peut prendre le système modélisé par ce RdP est fini, autrement dit, le nombre de marquages accessibles est fini.
   - Dans le cas contraire, où le RdP est non borné, le nombre d'états est infini et ceci est du, au fait que certains paramètres de ce système sont non bornés.

- Un RdP marqué est k- borné si toutes ses places sont k-bornées, c'est-à-dire, quel que soit le marquage accessible M à partir du marquage initial $M_0$, et quel que soit la place P, considérée le nombre de jetons contenus dans cette place est inférieur ou égale à une borne k.

**Définition 2.17.** *Soit un RdP R = (P,T,Pre,Post). une place $p \in P$ est dite K-bornée pour un marquage initial $M_0$ si et seulement si :*

$$\boldsymbol{A(R, M_0)} \textit{ est k-borné} \Leftrightarrow \boldsymbol{\forall M \in A(R, M_0), \forall p \in P, M(p) \leq M}.$$

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

2. **Vivacité [19] :**

Le réseau de Petri R est dit vivant si, quel que soit le marquage atteint à partir de $M_0$, il est possible de franchir toute transition du réseau en franchissant à travers une certaine séquence de franchissement . Un Rdp vivant est un réseau de Petri sans blocage. Un blocage est un marquage tel qu'aucune transition n'est validée.

3. **État d'accueil et réseau de Petri réinitialisable :**

Un réseau de Petri possède un état d'accueil $M_a$ pour un marquage initial $M_0$, si pour tout marquage accessible $M \in R(m_0)$, il existe une séquence de franchissement s tel que, M[s $> M_a$. Un réseau de Petri est réinitialisable pour un marquage initial $M_0$ si $M_0$ est un état d'accueil.

4. **Accessibilité :**

Le franchissement d'une transition validée dans un réseau de Petri apporte une modification au marquage initial du réseau. Un marquage $M_k$ est dit accessible à partir du marquage $M_0$, s'il existe une séquence de franchissement $s = T_1T_2...T_k$ qui donne $M_k$ à partir de $M_0$ . Dans ce cas, on écrit $M_0[s > M_k$ ce qui signifie que $M_k$ est accessible à partir de $M_0$ par s. L'ensemble de tous les marquages accessibles à partir de $M_0$ dans un réseau de Petri R est noté R($M_0$). L'ensemble de toutes les séquences de franchissement possibles à partir de $M_0$ dans un réseau de Petri est noté R($M_0$).

5. **Blocage :** Un blocage signifie qu'aucune transition n'est sensible (activable) à partir de ce marquage.

### 2.6.2 Les propriétés structurelles

Les propriétés structurelles dépendent uniquement de la topologie du réseau. Il s'agit de faire ressortir les propriétés statique du système étudié. Ces différentes propriétés sont indépendantes

du marquage [20, 23].

1. **P-invariant :** Les invariants de marquage appelés p-invariant ou encore p-semi flots, illustrent la conservation du nombre de jetons sans un sous ensemble de places du RdP. Un vecteur noté Y de dimension égale au nombre de places du RdP est un p-invariant, si et seulement s'il vérifié l'équation suivante :

$$Y^t * C = \vec{0}$$

   L'ensemble des places pour lesquelles la composante associée dans le p-invariante est non nulle est appelée la composante conservative du RdP, où C correspond à la matrice d'incidence du RdP

2. **T-invariant :** un vecteur non nul d'entiers X est un T-invariant, ou encore T-semi flots du RdP, si et seulement s'il vérifie l'équation suivante :

$$C * X = 0$$

   Un T-invariant correspondant à une séquence de franchissement réalisable appelé composante répétitive.

## 2.7 Réseaux de Petri particuliers

Quelques définitions des RdP particuliers [9, 26, 1] :

1. **Graphe d'état :** Un graphe d'état est un RdP, où chaque transition possède exactement une place d'entrée et une place de sortie.

2. **Graphe d'événement (GEV) :** Un graphe d'événement est un RdP, où chaque place possède exactement une transition d'entrée et une transition de sortie. Et parfois appelé graphe de transition.

3. **RdP sans conflit :** Un RdP est un réseau sans conflit, si et seulement si chaque place a au plus une transition de sortie. Un RdP avec conflit est un réseau qui possède donc une place avec au moins deux transitions de sorties.

4. **RdP Simple :** Un RdP simple est un RdP, ordinaire tels que chaque transition a au plus une place d'entrée qui peut être reliée à d'autres transitions.

5. **RdP autonome :** Un RdP autonome décrit le fonctionnement d'un système, dont les instants de franchissement ne sont pas connus ou indiqués.

6. **RdP non autonome :** Un RdP non autonome décrit le fonctionnement d'un système ; dont l'évolution est conditionnée par les événements externes ou par le temps. On peut dire qu'un RdP non autonome est synchronisé et/ou temporisé.

7. **RdP Pur :** Un RdP pur est un réseau dans lequel, il n'existe pas de transition ayant une place d'entrée qui soit à la fois place de sortie de cette transition.

8. **RdP Impure :** Un RdP impure est un réseau dans lequel, il existe une transition ayant une place qui soit à la fois place d'entrée et place de sortie.

9. **RdP généralisé :** Un RdP généralisé est un RdP dans lequel les poids sont associés aux arcs.

10. **RdP a capacité :** Un RdP a capacité est un RdP dans lequel des capacités (nombre entiers strictement positifs) sont associés aux places. Le franchissement d'une transition d'entrée d'une place $P_i$ dont la capacité et cap($p_i$) n'est possible que si le franchissement ne conduit pas à un nombre de jetons dans $P_i$ qui dépasse cette capacité.

11. **RdP priorité :** Dans un tel réseau, si on atteint un marquage tel que, plusieurs transitions sont franchissables on doit franchir la transition qui a la plus grande priorité.

## 2.8 Extensions des RdP

Le concept RdP classique a été largement développé par de nombreux auteurs dans les années 70, dans le monde entier, en intégrant particulièrement l'aspect temporel et stochastique dans le modèle initial. Ci-dessous, nous introduisons quelques extensions.

### 2.8.1 Réseaux de Petri à temps discret :

Les réseaux de Petri à temps discret peuvent être classés en deux familles :
• Les RdP temporisés qui permettent une modélisation du temps sur un espace entier.
• Les RdP stochastiques qui ont un comportement qui n'est pas déterministe, mais stochastique, ils permettent de modéliser les processus aléatoires.

#### Réseaux de Petri stochastiques (RdPS)

Les réseaux de Petri stochastiques (RdPS) occupent une place prépondérante en tant qu'approche d'évaluation des performances des systèmes informatiques ou indusriels [12]. Les problème d'évaluation liés à la sûreté faisant intervenir des phénomènes aléatoires, les transitions du réseau de Petri ont comporté des temps de franchissement aléatoires, distribués par une loi exponentielle. Cette distribution exponentielle permet d'exploiter les propriétés mathématiques

d'un processus de Markov.

**Définition 2.18.** *Un réseau de Petri stochastique est le couple (R ;∧) avec [25] :*
• *R = (P ;T ; A ;$M_0$)est un réseau de Petri ;*
• *∧ est une fonction qui à chaque transition t associe un taux de franchissement $\lambda_t = \wedge(t)$.*

Dans les RdPS la durée de sensibilisation est une variable aléatoire $\theta$, avec une distribution de probabilité, dans le cas de distribution exponentielle :

$$P_0(x) = P[\theta \leq x] = 1 - e^{-\lambda x}$$

La fonction $P_\theta(x)$ décrit la probabilité pour que le franchissement ait lieu avant x. Avec les RdPS, on pourra par exemple, calculer le temps de bon fonctionnement entre deux défaillances, le temps de réparation ou dans certains cas la durée opérationnelle d'une machine, les taux de production, l'évolution des stocks, etc.

#### Réseaux de Petri temporisés

L'introduction des temporisations dans un réseau de Petri permet de décrire un système dont le fonctionnement dépend du temps et permet ainsi d'envisager une analyse quantitative du système étudié. Dans un réseau de Petri temporisé, les temporisations sont associées aux places (RdP P-temporisé) ou aux transitions (RdP T-temporisé) [33]. Ces deux façons d'envisager les temporisations conduisent à des outils équivalents, même si la mise en oeuvre est différente.

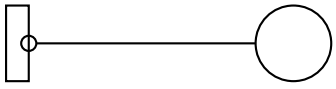
**Définition 2.19.** *Un RdP temporisé est défini par le couple (R,d) avec [25] :*

• *R est un réseaux de Petri R = (P,T,Pre,Post,$M_0$) ;*
• *$d : T \Longrightarrow Q^+$ est la fonction de temporisation.*

### 2.8.2 Les RdP étendus

#### Les réseaux de Petri à arcs inhibiteurs :

Un arc inhibiteur est un arc orienté qui part d'une place (P) pour aboutir à une transition [1]. La transition (T) n'est validée que si la place (P) ne contient aucune marque.
Le franchissement de (T) consiste à retirer une marque dans chaque place d'entrée de (T) à l'exception de (P), et à ajouter une marque dans chaque place de sortie de (T). On utilise aussi les expressions "test à zéro" et "RdP étendus".



Représentation d'un arc inhibiteur

**Définition 2.20.** *[4]*

*Un RdP à arcs inhibiteur est défini par un 5-uplet R = (P ;T ;Pre ;Post ;Inh), où :*
• *P : est un ensemble fini de places.*
• *T un ensemble fini de transitions.*
• *Pre : $P \times T$ et Post : $P \times T$ sont les fonctions d'incidence avant et d'incidence arrière respectivement.*
• *Inh : $P \times T \rightarrow N/(0)$ est la fonction d'inhibition.*

**Les réseaux de Petri à priorité :**

Un tel réseau est utilisé lorsque l'on veut imposer un choix entre plusieurs transitions validées [1].

**Définition 2.21.** *Un RdP étendu est un tuple RDPE = (R,Inh,>,$M_0$) , où :*

• *R est un RdP,*
• *Inh : $P \times T \rightarrow N^*$ est l'application d'inhibition qui associe à tout couple $(p_i, t_j)$ le poids de l'arc inhibiteur reliant la place $p_i$ à la transition $t_j$.*
• *> est une relation de priorité entre transitions ;*
• *$M_0$ le marquage initial.*

### 2.8.3 Réseaux de Petri hiérarchiques

Un réseau de Petri hiérarchique se compose de plusieurs modules 'pages', où chaque module représente un sous réseau de Petri. Le référencement d'un module se fait avec une transition, ou une place de substitution portant le nom du module en question. L'utilisation des RDP hiérarchiques est préconisée pour la modélisation des problèmes de grande taille [8], construire un modèle en combinant un certain nombre des réseaux en un seul réseau. L'objectif est de réduire la complexité d'un modèle en permettant la modélisation modulaire [18].

### 2.8.4 Réseau de Petri coloré

Les réseaux de Petri ordinaire sont généralement insuffisant et doivent être étendus pour qu'il serait applicable à des problèmes réels. Les RdP colorée (Jensen and Rozenberg) offrent une possibilité d'expression plus compacte que les réseaux Petri ordinaires. Parmi les caractéristique d'un RdP coloré, nous citons [32] :

• Chaque place contient des jetons typés (colorés). Nous associons à chaque place et transition des domaines de couleurs.

• Un arc liant une place et une transition est étiqueté par une application linéaire appelée fonction de couleur.

• Les transitions du réseau peuvent être munies d'une garde (une condition booléenne sur la couleur de la transition qui limite le tir de la transition aux couleurs pour lesquelles la garde est vraie).

Les RdP colorés ou dérivés des RdP ordinaires sont des RdP dans lesquels les jetons portent des couleurs et chaque couleur correspond à une information bien définie associée au jeton. Deux principales raisons justifie l'appel aux RdP colorés [11] :

• Possibilité de transporter une information structurée car les places ne contiennent pas que des jetons uniformes.

• Les RdPC s'utilise pour éviter le problème de taille importante des RdP ordinaires dans le cas ou des entités différentes présentent des comportements similaires ce qui condence le modèle.

**Définition 2.22.** *[11]*

*Un réseau de petri coloré non hiérarchique est un produit cartésien R = ( $\sum$,P,T,A,N,C,G) défini par :*

*•$\sum$ : l'ensemble fini de types non vides, appelés ensembles de couleurs, P : l'ensemble fini des places.*
*• T : l'ensemble fini des transitions.*
*• A : l'ensemble fini d'arcs tel que $P \cap T = P \cap A = T \cap A = \varnothing$.*
*• N : la fonction des nœuds. Elle est définie de A vers $P \times T \cup T \times P$.*
*• C : la fonction de couleurs de P vers $\sum$.*
*• G : la fonction des gardes.*

**Définition 2.23.** ***Marquage coloré [32] :***

*Soit R un RdP coloré. un marquage coloré désigne un état du réseau dans lequel :*
*Chaque place contient des jetons associés à des valeurs (appelées couleurs), pas simplement des jetons "anonymes" comme dans les réseaux de Petri classiques.*

## 2.9 Outils de simulations des réseaux de Petri

Avec l'évolution, plusieurs chercheurs ont développé des outils de simulation et de vérification des RdP tel que (CPNTools, Greatspn, CPNAMI, PROD, JARP, MARIA, LOLA, Petri Net Kernel,INA, OPMSE, Artifex, ExSpect, FLOWer,... . Ces outils présentent un environnement gra-

phique d'édition des RdP avec la possibilité de simuler le modèle et d'analyser des propriétés génériques des RdP.

## Conclusion

L'un des principaux atouts des réseaux de Petri réside dans leur solide fondement mathématique, ainsi que dans leur aptitude à modéliser la concurrence. Dans ce chapitre, nous avons introduit le formalisme des réseaux de Petri, en présentant certaines définitions et propriétés fondamentales, ainsi que quelques-unes de leurs extensions. Dans les chapitres suivants, nous proposons une étude de l'existant sur l'évaluation des performances des Web services.

# 3

# Étude de l'existant sur l'évaluation des performances des Web services

**Sommaire**



## Introduction

L'évaluation des performances des Web services est devenue un enjeu majeur dans les systèmes distribués modernes. De nombreux chercheurs ont proposé des approches variées pour modéliser, simuler et analyser le comportement de ces services en termes de temps de réponse, de disponibilité ou encore de débit. Ce chapitre présente une étude de l'existant en mettant en lumière les principales méthodes et modèles utilisés dans la littérature. Une comparaison critique entre ces travaux est ensuite proposée afin d'identifier les points forts, les limites et les perspectives d'amélioration, notamment en ce qui concerne l'utilisation des Réseaux de Petri dans ce contexte.

## 3.1 Travaux sur l'évaluation de performances des Web sevices

Dans [30], Ce travail est basé principalement sur les Réseaux de Petri Colorés (RdPC), les Réseaux de Petri Stochastiques (RdPS), ainsi que sur des modèles de files d'attente. Ces modèles ont été utilisés de manière complémentaire afin de représenter à la fois la structure logique, le comportement aléatoire et la dynamique d'arrivée des requêtes dans un système de

Web services. Les auteurs ont proposé des solutions pour la découverte, la sélection et la composition des Web services dans le but d'optimiser la recherche en termes de temps de réponse. L'évaluation des performances des solutions proposées est basée sur la simulation, ce qui permet d'analyser le système dans des scénarios réalistes et dynamiques. Deux axes sont abordés :
* La sélection de Web services fondée sur des contraintes de qualité de service (QdS),
* La sélection de Web services composites pour un groupe de clients.
Les indicateurs analysés incluent notamment le temps de réponse et le débit pour évaluer la qualité de service, ainsi que le temps de réponse et le coût global dans le cadre d'un ensemble de clients. Ce travail se distingue par une modélisation riche et détaillée, capable de couvrir les phases critiques de fonctionnement des Web services, bien qu'il reste limité à une validation par simulation, sans mise en œuvre concrète dans un environnement réel.

Dans [5], propose un modèle innovant de générateurs d'événements basé sur les réseaux de Petri colorés et temporisés (TCPN) pour simuler les systèmes Web. L'objectif principal est de concevoir un mécanisme flexible permettant de reproduire des charges de travail réalistes, essentielles pour l'analyse de performance des systèmes distribués comme les infrastructures web ou cloud.
Le modèle repose sur la construction de générateurs d'événements configurables, capables de simuler divers profils d'utilisateurs en générant des flux de requêtes selon des paramètres définis (déterministes ou stochastiques, temporisés ou non). Chaque type de générateur (déterministe/ stochastique et temporisé/non-temporisé) est formalisé à l'aide de TCPN et implémenté dans l'outil CPN Tools, permettant à la fois la modélisation graphique et la simulation des systèmes. Les générateurs produisent des événements en fonction de distributions statistiques précises (ex. loi exponentielle) et peuvent moduler la fréquence ou l'intensité selon le temps, simulant ainsi un trafic réaliste sur un système web. Plusieurs hypothèses ont été posées pour modéliser ce comportement :

— Les événements représentent des requêtes d'utilisateurs vers des ressources web.

— Le système est modélisé comme un ensemble de transitions entre états, où chaque événement peut être attribué à un utilisateur ou une ressource.

— Les événements peuvent suivre des distributions aléatoires ou être définis de manière déterministe.

— Le système serveur ne modélise pas les délais réseau physiques, et les événements sont supposés indépendants.

Les résultats obtenus à partir des simulations montrent que les générateurs proposés sont adaptables et efficaces pour produire des charges de travail variées. Ils permettent de :

— Analyser l'utilisation des ressources du système (CPU, file d'attente, etc.)

— Identifier les goulets d'étranglement dans des situations de surcharge

— Évaluer la performance de traitements sous différentes charges (nombre de requêtes, etc.)

Une étude dans l'article démontre l'application d'un générateur stochastique temporisé à un modèle simplifié de serveur web. Les résultats montrent que le nombre de processeurs disponibles affecte directement le temps d'attente dans la file d'entrée et la capacité du système à traiter les requêtes en temps utile. Le modèle met en évidence l'importance de la simulation

pour tester différents scénarios et optimiser l'architecture système dès la phase de conception.

Dans [7], Ils ont proposé une approche innovante pour l'évaluation des performances des Web services, fondée sur une modélisation en réseaux orientée vers la découverte, la recommandation et la composition des services. Contrairement aux approches classiques, elle ne considère pas les services comme des entités isolées, mais comme des nœuds d'un graphe interconnecté, mettant en évidence les relations fonctionnelles et sémantiques qu'ils entretiennent. Pour cela, elle conçoit deux types de réseaux : les réseaux d'interaction, qui modélisent les dépendances de composition entre services, et les réseaux de similitude, basés sur des critères de ressemblance fonctionnelle ou sémantique.

La méthodologie repose sur des outils issus des sciences des graphes et de la fouille de données. L'analyse des concepts formels (ACF) est utilisée pour organiser les services selon leurs propriétés communes, permettant une représentation hiérarchique de leurs caractéristiques. Un modèle probabiliste est ensuite introduit pour la sélection de services, tenant compte de la pertinence à la requête, de la diversité (pour éviter la redondance) et de la densité relationnelle du voisinage. En complément, un système de recommandation hybride est proposé, combinant le filtrage collaboratif et le filtrage basé sur le contenu, enrichi par l'extraction de motifs fréquents et l'identification de modèles thématiques. Ces derniers permettent de détecter des liens sémantiques implicites, améliorant ainsi la qualité des recommandations.

Les hypothèses de travail formulées dans cette étude reposent sur l'idée que la structuration des services sous forme de graphes permet une meilleure compréhension et exploitation de leurs relations. Elle suppose également que l'intégration de critères comme la diversité et la densité améliore la pertinence des résultats. Une dernière hypothèse suggère que les Web services ont tendance à se regrouper naturellement en communautés cohérentes, tant sur le plan topologique que sémantique.

Les résultats expérimentaux confirment la validité de ces hypothèses. L'approche probabiliste améliore la découverte de services variés tout en maintenant leur pertinence. Le système de recommandation hybride s'est avéré plus précis grâce à l'enrichissement sémantique, et l'analyse des communautés a révélé des regroupements cohérents de services. Ainsi, cette thèse apporte une contribution notable en combinant modélisation en graphes, techniques d'intelligence artificielle et extraction de connaissances pour répondre efficacement aux enjeux complexes de la gestion intelligente des Web services.

Dans [15], Ils proposent une approche de modélisation destinée à évaluer les performances des systèmes de Web services, qu'ils soient simples ou composites. Le contexte de ce travail repose sur la croissance exponentielle des Web services disponibles sur Internet et la nécessité d'assurer des réponses rapides et pertinentes aux requêtes des utilisateurs. Dans ce cadre, les auteurs s'intéressent tout particulièrement à la problématique de la découverte et de la composition des Web services, des processus devenus critiques face à la complexité croissante des demandes utilisateurs.

Pour analyser et simuler le fonctionnement d'un tel système, les auteurs ont développé un modèle fondé sur les réseaux de files d'attente, plus précisément un modèle markovien. Ce choix repose sur deux hypothèses fondamentales : d'une part, les temps entre deux arrivées de requêtes et les temps de service sont modélisés comme des variables aléatoires indépendantes suivant des

lois exponentielles ; d'autre part, le système présente la propriété de "sans mémoire", propre à la loi exponentielle, ce qui facilite grandement les calculs analytiques. Le modèle met en scène trois stations : la première représente l'accès Internet, la deuxième s'occupe de la découverte des services simples, et la troisième de la composition des services complexes lorsque les simples ne suffisent pas à satisfaire une requête. En cas d'échec même après la tentative de composition, la requête est renvoyée à la première station pour une nouvelle tentative, illustrant ainsi un mécanisme réaliste de réémission des requêtes.

La méthode de résolution employée s'appuie sur le calcul des probabilités stationnaires et des taux d'arrivée dans chaque station. À partir de ces éléments, les auteurs déterminent des indicateurs de performance clés tels que la durée moyenne de séjour dans le système. L'un des résultats majeurs de cette étude montre que cette durée moyenne augmente proportionnellement avec le taux d'arrivée des requêtes. Cela signifie que le système devient rapidement saturé si un trop grand nombre de requêtes échouent lors de la première tentative, soulignant l'importance de l'efficacité initiale du processus de découverte.

Dans [14], Ils proposent une approche analytique pour modéliser et évaluer les performances des systèmes de Web services. Ce travail s'inscrit dans le contexte de la croissance rapide des Web services et de la nécessité d'assurer des réponses efficaces aux requêtes des utilisateurs.

Le modèle proposé repose sur une structure de files d'attente, où les Web services sont représentés comme des serveurs traitant des requêtes arrivant selon un processus aléatoire. Les auteurs utilisent des chaînes de Markov pour modéliser le comportement du système, en supposant que les temps d'arrivée des requêtes et les temps de service suivent des distributions exponentielles. Cette modélisation permet de capturer les dynamiques du système et d'analyser des indicateurs de performance tels que le temps moyen de réponse et le taux d'utilisation des serveurs.

Les hypothèses principales incluent la description formelle des Web services, permettant une analyse automatique, et l'organisation des services en communautés basées sur des critères de similarité. Les auteurs supposent également que l'utilisation de mesures multicritères améliore la pertinence des services sélectionnés.

Les résultats obtenus à partir du modèle montrent que l'approche proposée permet d'évaluer efficacement les performances des systèmes de Web services. Les analyses révèlent que l'organisation en communautés et l'utilisation de structures d'indexation contribuent à améliorer la rapidité et la pertinence des processus de découverte et de composition des services. Cette modélisation offre ainsi un outil précieux pour la conception et l'optimisation des architectures de Web services dans des environnements distribués.

Dans [24], ils ont proposé une approche de modélisation et d'évaluation des performances d'un système de Web services en utilisant les Réseaux de Petri Colorés (RdPC). Le modèle développé repose sur la mise en place de deux files d'attente distinctes : la première représente les Web services disponibles (qu'ils soient simples ou composites), tandis que la seconde modélise les demandes formulées par les clients.

Chaque entité (client ou Web service) est représentée à l'aide de couleurs distinctes, ce qui permet une meilleure structuration et différenciation dans la simulation. La synchronisation entre les files d'attente permet d'imiter le mécanisme réel d'association entre une requête

client et un Web service adéquat. L'objectif de cette approche est de simuler le comportement dynamique du système afin d'en extraire des indicateurs de performance pertinents.

Les performances du système ont été évaluées à l'aide du logiciel CPN Tools, un outil spécialisé dans la simulation de réseaux de Petri colorés. Les métriques étudiées incluent notamment : Le nombre moyen de clients présents dans le système, le nombre moyen de Web services actifs ainsi que le nombre moyen de clients servis.

Ce travail met en évidence la capacité des RdPC à modéliser avec précision des systèmes complexes à comportements concurrents et à grande échelle, tout en tenant compte de la diversité des entités et des interactions. Il confirme également l'utilité des outils de simulation dans l'analyse quantitative des performances, notamment dans le contexte des architectures orientées services.

Dans [16], Ils ont proposé un modèle de réseau de petri coloré avec synchronisation de deux file d'attente, où la première présente les Web services et la deuxième présente les demandes des clients. ils ont utilisé la méthode analytique pour calculé nombre moyen de clients dans le système et le temps moyen de séjour dans le système.

## 3.2 Étude comparative des travaux

La TAB 4.1 montre une étude comparative pour les travaux présenté précédemment sur l'évaluation de la performance des Web services.

**Modèle :** Identifier les méthodes utilisé.

**Solution :** méthode analytique, simulation et un prototype.

**Évaluation de performance :** temps de réponse, coût, etc.

**Web service :** nous avons deux types, Web service simple, Web service composites.

| | TR1 [30] | TR2 [5] | TR3 [7] | TR4 [15] | TR5 [14] | TR6 [24] | TR7 [16] |
|---|---|---|---|---|---|---|---|
| **Modèle** | | | | | | | |
| RdPC | ✓ | ✓ | ✓ | | | ✓ | ✓ |
| RdPS | ✓ | | | | | | |
| File d'attente | ✓ | | ✓ | | ✓ | | |
| RdT | | ✓ | | ✓ | | | |
| **Solution** | | | | | | | |
| Analytique | | | | | | | ✓ |
| Simulation | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Prototype | | | | | | | |
| **Évaluation de performance** | | | | | | | |
| Disponibilité | | | | | | | |
| Temps de réponse | ✓ | | | | ✓ | | ✓ |
| Coût | ✓ | | | | | | |
| Nombre moyen | | | | | ✓ | ✓ | ✓ |
| Débit | ✓ | | ✓ | | ✓ | | |
| Requête client | | ✓ | | | | | |
| Temps d'arrivée | | | | ✓ | | | |
| TMDS | | ✓ | | | | | |
| Durée moyenne | | ✓ | ✓ | ✓ | | | |
| **Web service** | | | | | | | |
| Simple | | | | ✓ | ✓ | ✓ | |
| Composite | ✓ | ✓ | ✓ | | | ✓ | ✓ |

**Table 4.1 :** Étude comparative des travaux de l'existence de l'évaluation de performances des Web services

**TR1 [30] : Le travail de :** " W.S. Serrai, " Evaluation de performances de solution pour la découverte et la composition des Web services", Thèse, Université Paris-Est, 2020 ".
**TR2 [5] : Le travail de :** " Bozek A., Rak T., Rzonca D., "Timed Colored Petri Net-Based Event Generators for Web Systems Simulation", 2022 ".
**TR3 [7] : Le travail de :** " Hafida Naim, "Réseaux de Web service : construction, analyse et applications", Thèse, 2017 ".
**TR4 [15] : Le travail de :** " N. Bernine, H. Nacer, K. Adel, D. Aissani, "Simulation et analyse d'un Web service", Séminaire Mathématique de Bejaia (LaMos), vol.13, pp. 17–20, 2014 ".
**TR5 [14] : Le travail de :** " N. Bernine, D. Aissani, "Un Modèle pour l'évaluation des Performances d'un Système de Web Services", Séminaire Mathématique de Bejaia (LaMos), vol.14, pp. 3–7, 2015 ".
**TR7 [24] : Le travail de :** " Sahli Ramzi, " Evaluation de performances d'un système des Web services", Mémoire Master, Université de Béjaia, 2022. "
**TR6 [16] : Le travail de :** " N.Bernine Qualité de services dans la découverte et la composition des web services. Thèse de doctorat université Bejaia 2018 ."

### 3.2.1 L'analyse critique du tableau

À travers cette étude comparative, on constate que la majorité des approches privilégient la simulation à l'analyse formelle, souvent au détriment de la généricité et de la rigueur mathématique.

De plus, pas mal d'approches modélisent à la fois la diversité des Web services et le comportement dynamique des clients. C'est dans cette optique que notre travail se positionne, en proposant un modèle RdPC capable de représenter finement les interactions entre Web services simples et composites, tout en assurant une simulation fidèle via CPN Tools.

## Conclusion

L'étude de l'existant sur l'évaluation des performances des Web services met en lumière la diversité des approches, des outils et des indicateurs utilisés pour mesurer et analyser l'efficacité de ces services. Qu'il s'agisse de métriques comme le temps de réponse, le débit, la disponibilité ou encore la scalabilité, ces critères sont essentiels pour garantir un niveau de qualité de service (QoS) adapté aux besoins des utilisateurs et aux exigences des applications modernes. Les recherches actuelles montrent une évolution constante vers des solutions plus automatisées, intégrant des mécanismes d'analyse en temps réel, d'auto-adaptation et d'optimisation dynamique. Cette synthèse constitue ainsi une base solide pour orienter les travaux futurs, notamment dans le développement de nouvelles méthodes d'évaluation plus précises, contextualisées et orientées utilisateur.
C'est dans cette optique que notre travail se positionne, en proposant un modèle RdPC capable de représenter finement les interactions entre Web services simples et composites, tout en assurant une simulation fidèle via CPN Tools.
Dans le chapitre qui suit nous allons proposé un modèle basé sur les réseaux de petri colorés, nous prenons en considération les arrivées des clients et des Web services, à l'aide d'un simulateur. Nous allons évaluer les performances du modèle, par rapport à certains critères (Nombre des clients, durée moyen d'arrivé, durée moyen de séjour, ...).

# 4

# Évaluation de performances d'un modéle du Web services avec un réseau de Petri coloré

**Sommaire**



## Introduction

Dans ce chapitre nous allons modéliser et évaluer les performances d'un système de Web services. Nous avons construit un modèle avec les réseaux de petri coloré. Cette approche nous a permis d'avoir un modèle du système de Web services, qui prend en compte deux types différents d'arrivées : les clients et les Web services, sachant que les clients sont différents et les Web services sont différents. Les performances de ce modèle sont calculées à l'aide d'un outil de simulation appelé CPN Tools [24].

## 4.1 Modélisation des demandes des clients dans un système de Web services

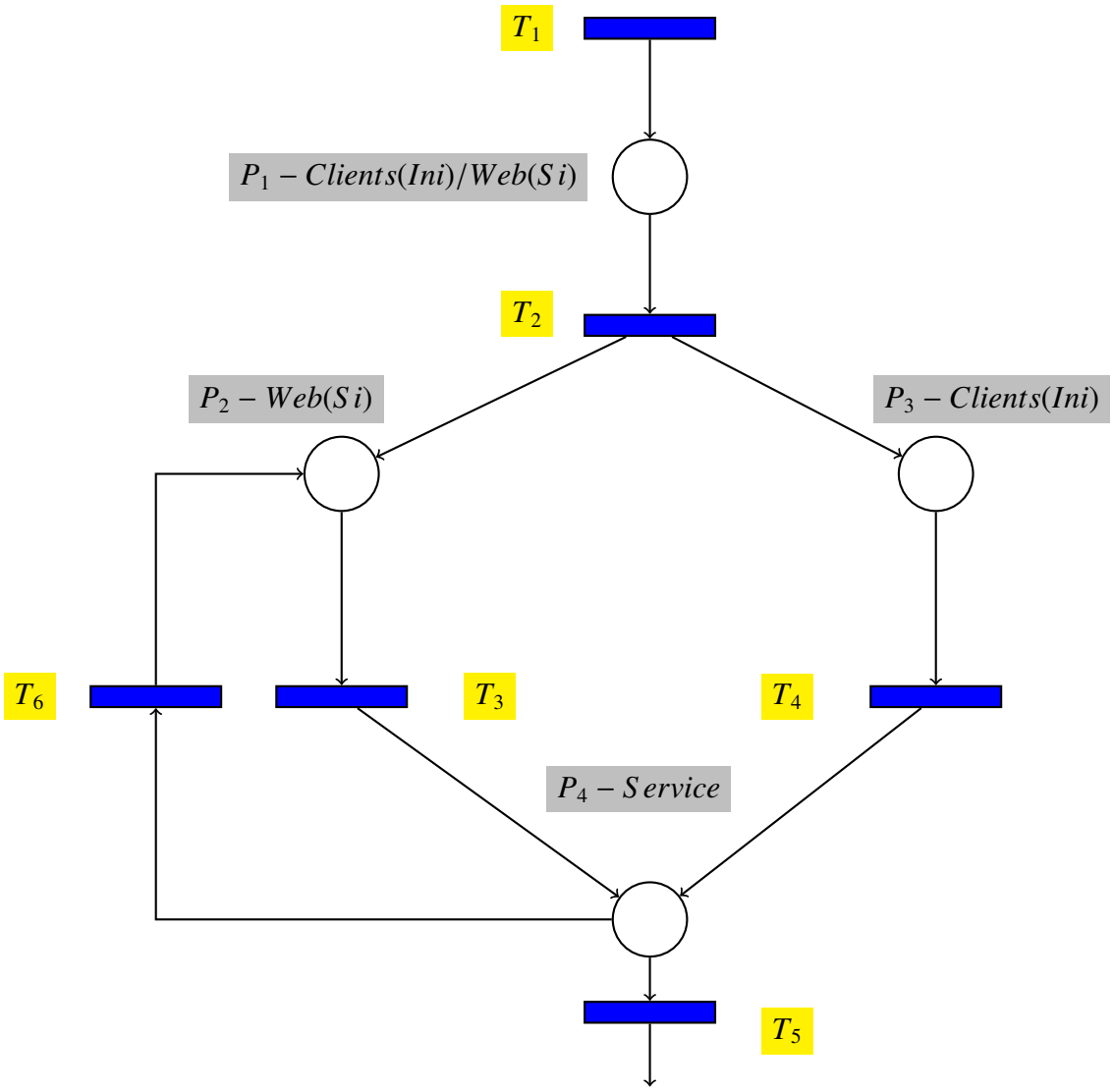
Le modèle que nous avons proposé décrit le système de Web services, où l'arrivée des demandes clients et des Web services sont prises en compte pour évaluer ses performances,

avec un serveur exponentiel. Dans ce modèle, nous avons deux demandes :

1. Plusieurs demandes des clients qui arrivent noté ($C_{i:n}$) tel que {$C_1$, ... ,$C_n$} et n= {1, ... ,10}.
2. Plusieurs demandes des web services noté ($Ws_{i:n}$) tel que {$Ws_1$, ... ,$Ws_n$} et n= {1, ... ,10}.

Notons que les arrivée sur le réseau suivent la distribution de Poisson.

Le modèle en bas présente un modèle d'un Web service avec les réseaux de petri colorés.




**Modèle de réseau de Petri coloré du système des Web services**

## 4.2 Description du modèle de réseau de Petri coloré pour un Web service

Le modèle proposé est un réseau de Petri coloré, a le but de modéliser le fonctionnement d'un système d'invocation de Web services par des clients. Il est constitué de cinq places principales et de six transitions, chacune jouant un rôle bien précis dans la gestion des requêtes client-service et dans la production des résultats. Voici les détails des composants :

— **Places :**

1. **$P_1$ (Clients et Web services) :** Représente les clients et les Web services. Les variables utilisées sont *Ini* et $Si$ respectivement.
2. **$P_2$ (Web services) :** Représente les Web services disponibles pour le traitement.
3. **$P_3$ (Clients) :** C'est une place qui représente les clients.
4. **$P_4$ (Service) :** Cette place représente le traitement et l'exécution de la demande de client avec un Web service. C'est l'affectation d'un Web service à un client.

— **Transitions :**

1. **$T_1$ (L'arrivée) :** Cette transition permet l'arrivée des Web services et des clients dans la place $P_1$.
2. **$T_2$ (Acheminement) :** Une transition conçue pour acheminer prioritairement avec une condition qui lui est déjà accordé les Web services vers $P_2$, puis rediriger ultérieurement les clients vers $P_3$.
3. **$T_3$,$T_4$ (Affectation1, Affectation2) :** Elles permettent de diriger les Web services et les clients vers $P_4$ où la demande du client est traitée et exécutée.
4. **$T_5$ (Extraction) :** Elle assure l'extraction de résultat de service depuis $P_4$ où le client est considéré comme servi et de le faire sortir dans le système.
5. **$T_6$ (Récupération) :** Elle vise à récupérer le Web service déjà utilisé pour le réaffecter à la place $P_2$ en vue d'une nouvelles utilisation.

— **Déroulement de modèle :**
Le modèle présenté illustre un système organisé de traitement et d'affectation des Web services à des clients.
Le processus débute par l'arrivée des Web services et des clients dans le système, représentée par la transition $T_1$. Celle-ci introduit ces entités dans la place $P_1$, qui joue le rôle de point d'entrée central. Les entités y sont différenciées par leur type : les Web services sont désignés par la variable $S_i$ tandis que les clients le sont par $In_i$.

Une fois introduits dans $P_1$, ces entités sont orientées vers leurs files respectives grâce à la transition $T_2$, qui agit comme un répartiteur intelligent. Cette étape de distribution suit une logique de priorité avec une condition qui lui est déjà accordé : les Web services sont dirigés en premier vers la place $P_2$, où ils sont mis à disposition pour traitement. Par la suite, les clients sont transférés vers la place $P_3$, en vue d'un traitement ultérieur.

Lorsque des clients et des Web services sont disponibles, les transitions $T_3$ et $T_4$ entrent ensuite dans le déroulement. Ces deux transitions fonctionnent de manière parallèle et permettent l'affectation d'un client à un Web service. Elles réalisent ainsi le couplage entre les entités issues respectivement de $P_2$ et $P_3$, les dirigeant ensemble vers la place

$P_4$. Cette dernière symbolise l'unité d'exécution où la demande du client est traitée par le Web service sélectionné.

Une fois le traitement achevé, la transition $T_5$ prend le relais. Elle extrait le client traité de $P_4$, qui représente les clients ayant été servis et de le faire sortir dans le système. Cela marque la fin du cycle pour la demande client, le résultat ayant été produit et enregistré.

Enfin, pour assurer une utilisation optimale des ressources, le modèle prévoit une étape de recyclage des Web services via la transition $T_6$. Celle-ci permet de récupérer les services ayant terminé leur traitement dans $P_4$ pour les renvoyer vers $P_2$, les rendant ainsi de nouveau disponibles pour de futures demandes.

Ce déroulement illustre un cycle fluide et efficace, assurant à la fois la satisfaction des demandes clients et une réutilisation intelligente des Web services. Il met en œuvre des principes fondamentaux tels que la séparation des flux, l'affectation contrôlée, la production synchrone et la réutilisation dynamique, répondant parfaitement aux exigences des systèmes distribués modernes.

## 4.3 Étude du modèle du réseau de Petri coloré

Nous avons implémenté le modèle sur le simulateur CPN Tools, la figure en bas montre comment le modèle apparaît sur l'interface du simulateur :

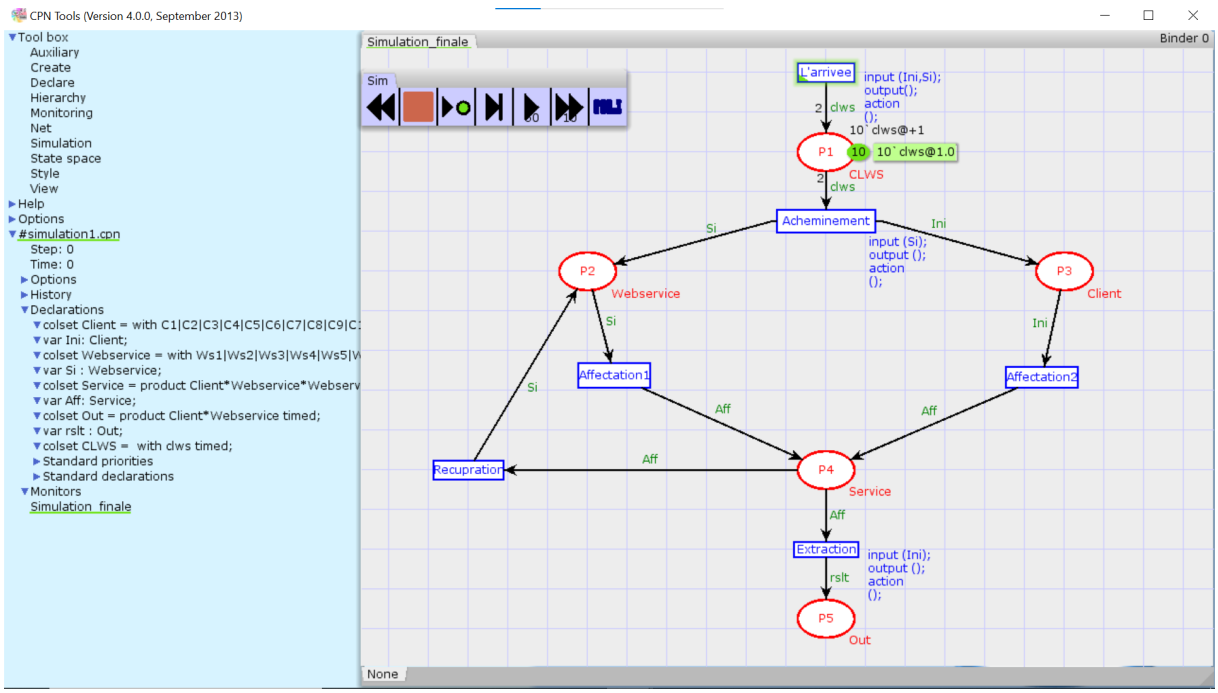



FIGURE 4.1 – Le modèle proposé sur l'interface de simulateur CPN Tools

## 4.4 Résultats de la simulation

Après avoir exécuter le modèle avec le simulateur, nous avons eu des différent résultats.

### 4.4.1 Résultats par rapport au Client

Nous avons effectué une simulation du notre modèle à l'aide du simulateur CPN Tools, en partant de 10 jusqu'à 100 000 steps. Les résultats obtenus sont résumés dans les tableaux ci-dessous :

**Nombre des clients dans le système**

| $N^{\dagger}$ simulation | Nombre de Client |
|---|---|
| 10 | 4 |
| 100 | 33 |
| 1000 | 335 |
| 10000 | 3392 |
| 100000 | 32866 |

TABLE 4.1 – Nombre des clients dans le système

La tab 4.1 nous montre le nombre des clients dans le système qui attendent pour être servis par les Web services dont il a besoin. Nous remarquons qu'à chaque fois que le nombre de simulation augment le nombre des client dans le système augment a cause des arrivées des clients de l'extérieur.

**Nombre des clients servis dans le système**

| $N^{\dagger}$ simulation | Nombre de Client servi |
|---|---|
| 10 | 4 |
| 100 | 16 |
| 1000 | 164 |
| 10000 | 1621 |
| 100000 | 16895 |

TABLE 4.2 – Nombre des clients servis dans le système

La tab 4.2 nous montre le nombre des client servis dans le système. Nous remarquons qu'à chaque fois que le nombre de simulation augment le nombre des client servis dans le système augment a cause des affectation qui déroule dans la place "Service".

A partir de la tab 4.1 et 4.2, nous remarquons que il y a encore un certain nombre de clients qui n'ont pas servis, cela signifie qu'ils attendent d'autres Web services dont ils ont besoin ou ne sont pas servis dans notre système. Cette différence nous montre que le déroulement peut avoir des limitations dans la gestion des affectations, en particulier lorsque les Web services nécessaires ne sont pas disponibles en temps voulu. Cela peut également refléter un déséquilibre entre l'arrivée des clients et la disponibilité des services, ou encore une inefficacité dans le mécanisme d'affectation utilisé.

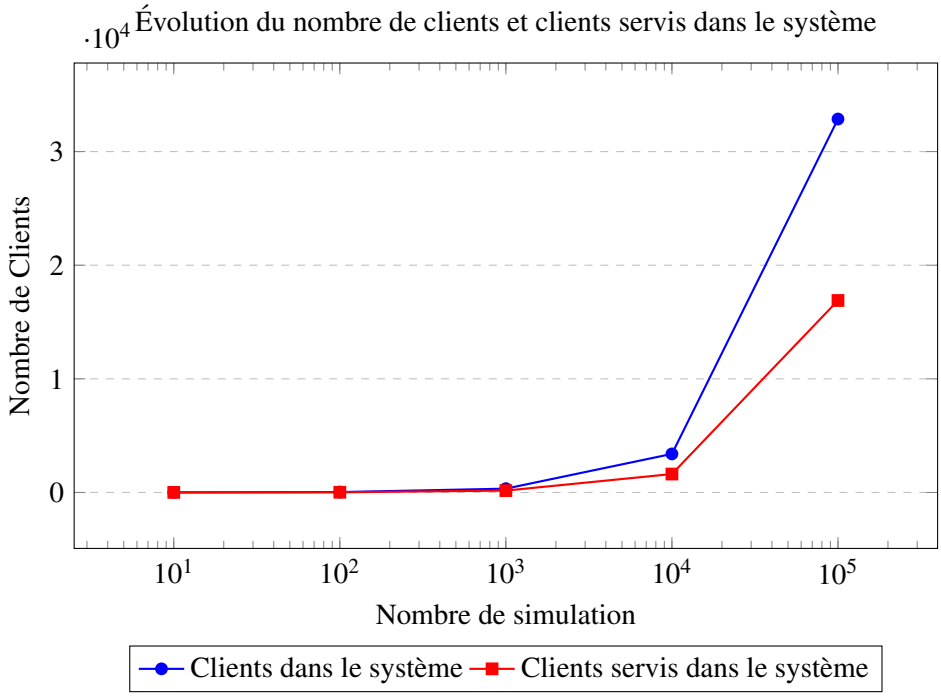



FIGURE 4.2 – Graphe pour nombre de clients et clients servis dans le système en fonction du nombre de simulations

A partir de deux tableaux en constate l'évolution des clients et clients servis dans le système, et avec le graphe, les deux courbes suivent une croissance linéaire, avec l'axe des abscisses en échelle logarithmique.

**Résultats relatifs à la répartition des Web services entre les clients**

Nous avons analysé les résultats obtenus en fonction de la stratégie de composition et d'affectation des Web services, mettant en évidence l'importance de la combinaison de Web services simples pour répondre à des requêtes complexes avec trois simulation (pour 10 steps, 100 steps et 1000 steps) .

Le premier résultat obtenue pour une simulation égale à 10 steps donne :

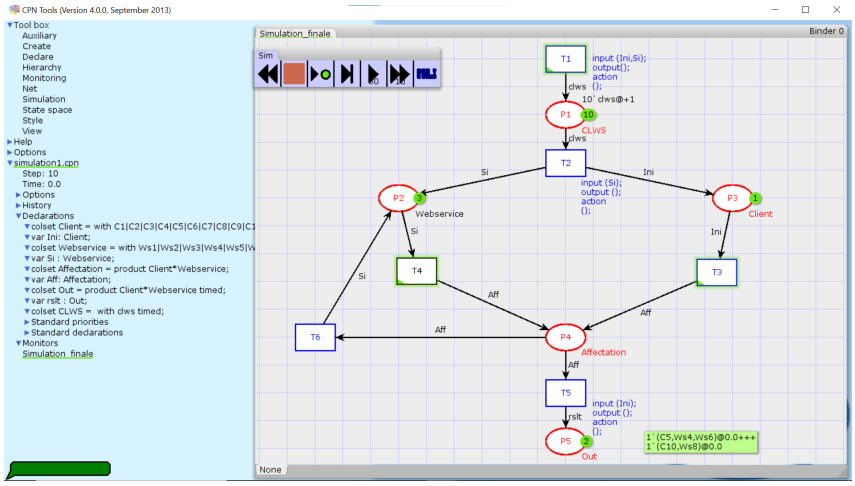



FIGURE 4.3 – Les résultats de la simulation pour 10 steps illustrent la diversité des affectations entre clients et Web services

La Figure 4.3 met en évidence la présence de deux types de clients : ceux bénéficiant d'une affectation simple ( ex : 1‘(C10,Ws8) ) et ceux traités via une affectation composite ( ex : 1‘(C5,Ws4,Ws6) ).

Le deuxième résultat obtenue pour une simulation égale à 100 steps donne :

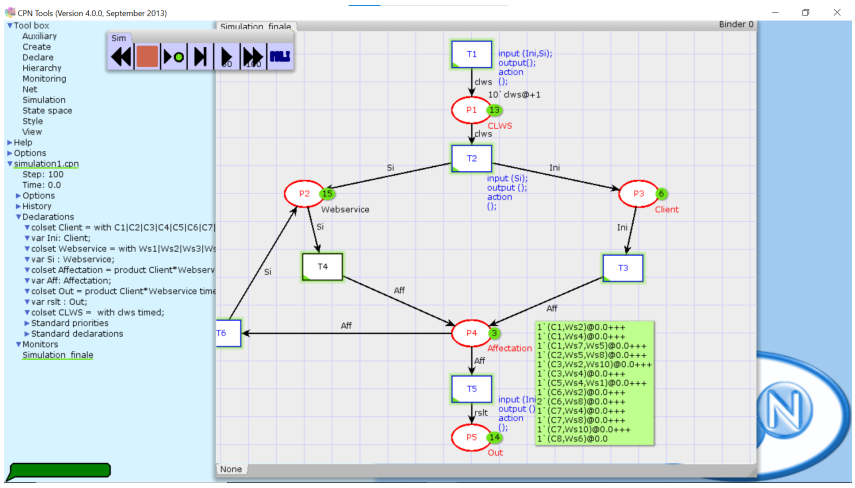



FIGURE 4.4 – Les résultats de la simulation pour 100 steps illustrent la diversité des affectations entre clients et Web services

La Figure 4.4 met en évidence la présence de plusieurs types de clients : ceux bénéficiant d'une affectation simple ( ex : 1‘(C1,Ws2) ) et ceux traités via une affectation composite ( ex : 1‘(C3,Ws2,Ws10) ).

Le troisième résultat obtenue pour une simulation égale à 1000 steps donne :

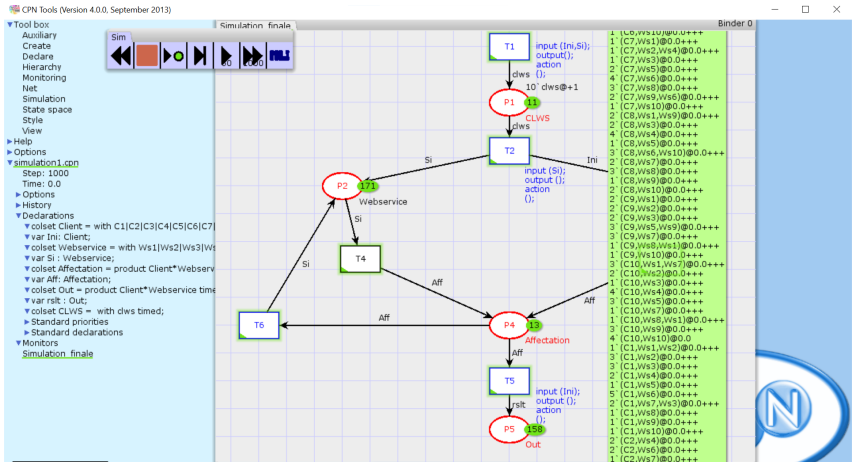



FIGURE 4.5 – Les résultats de la simulation pour 1000 steps illustrent la diversité des affectations entre clients et Web services

La Figure 4.5 met en évidence la présence de plusieurs types de clients : ceux bénéficiant d'une affectation simple ( ex : 2‘(C7,Ws5) ) et ceux traités via une affectation composite ( ex : 3‘(C8,Ws6,Ws10) ).

Ce qui résume que notre modèle et simple et composite au même temp. Il peut gérer des affectation simple ou multiple de Web service pour un client traité.

### 4.4.2 Résultats par rapport aux différents types de durée moyenne

Afin d'évaluer les performances temporelles du système modélisé à l'aide des réseaux de Petri colorés, une série de simulation a été réalisée. L'objectif de cette étape est de mesurer et d'analyser différents indicateurs clés, tels que la Temps moyen d'arrivée des clients et web services(P1), la durée moyen de séjour dans les places critiques (P2,P3) et la durée de sortie(P4).

TABLE 4.3 – Différents type de durée moyenne

| N° steps | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|
| Temps moyen d'arrivée P1 | 5,5 | 50,5 | 500,5 | 5000,5 | 50000,5 |
| Durée moyenne de séjour P2 | 2,0 | 2,0 | 2,0 | 2,0 | 2,0 |
| Durée moyenne de séjour P3 | 2,0 | 2,0 | 2,0 | 2,0 | 2,0 |
| Durée moyenne de sortie P4 | 1,0 | 1,0 | 1,0 | 1,0 | 1,0 |

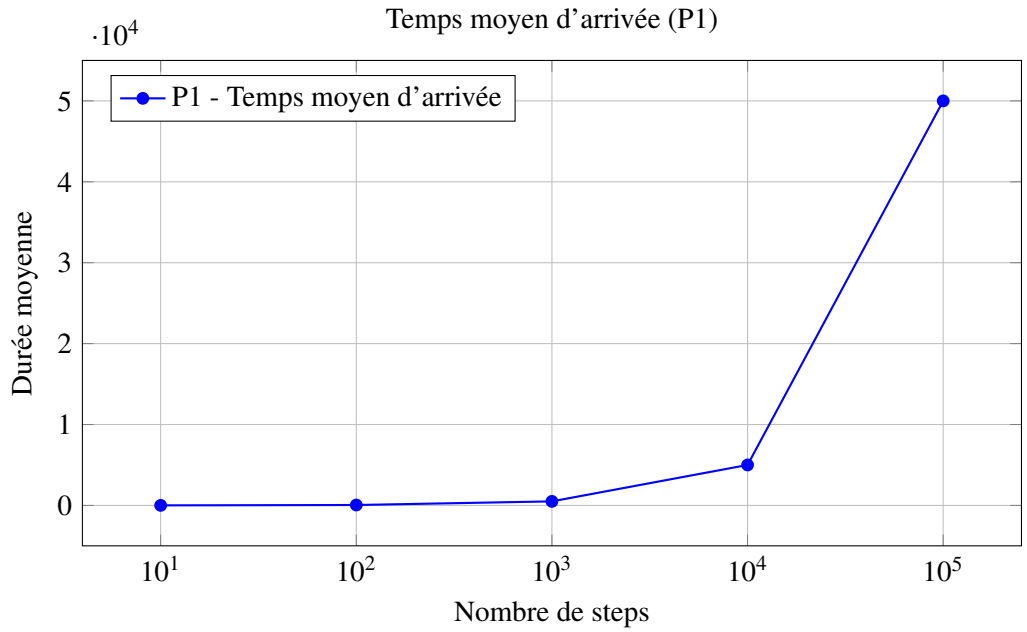



FIGURE 4.6 – Évolution du temps moyen d'arrivée (P1) selon le nombre de steps.

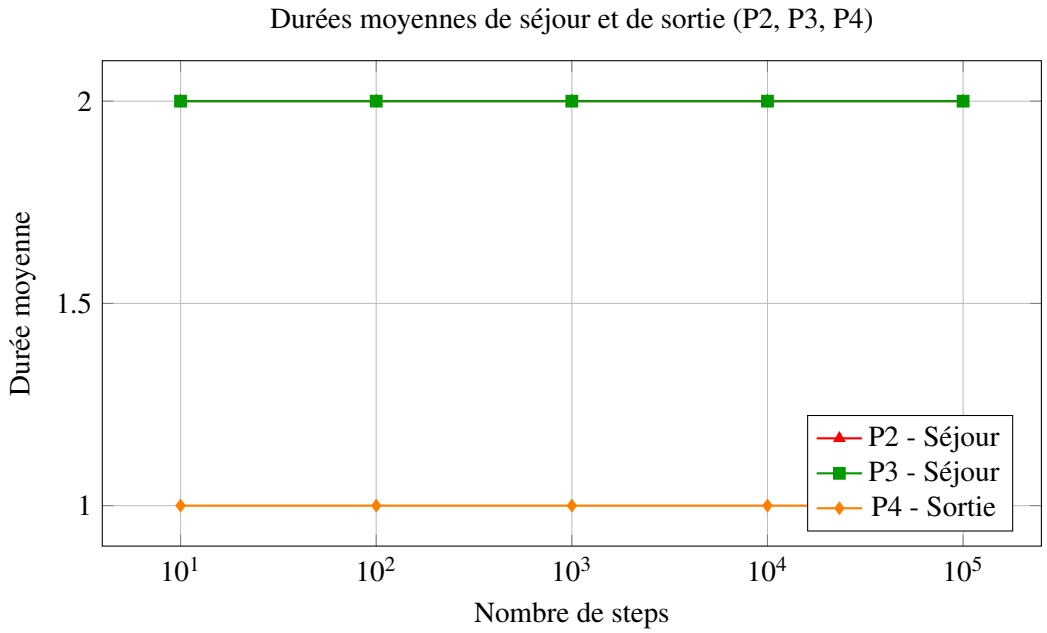



FIGURE 4.7 – Durées moyennes constantes pour P2, P3 (séjour) et P4 (sortie).

— **Temps moyen d'arrivée dans P1 :**
La Temps moyen d'arrivée augmente proportionnellemnt avec le nombre de steps simulès. Cette croissance n'est pas due à une lenteur du système, mais simplement à la maniére dont les jetons (clients et web services) sont injectés dans le modèle : un jeton est introduit à chaque unité de temps.

— **Durée moyen de séjour dans P2 :**
Ce temps est resté stable autour de 2.0 unités, quelle que soit la durée de la simulation. Cela s'explique par le fait que :
• Chaque web service, qu'il soit nouveau ou recyclé, passe le même temps dans la place P2.
• La transition associée (T3 par exemple) est programmée avec un délai fixe une unité.
• Même si un même web service revient plusieurs fois, chaque passage individuel est traité de la même façon.
Ainsi le recyclage des web services est correctement pris en compte en calculant la moyenne par passage, et non par identifiant unique. Cela justifie la stabilité de la valeur.

— **Durée moyen de séjour dans P3 :**
De la même manière que pour P2, les clients passent un temps constant une unité dans P3, en attente de traitement. L'absence de file d'attente ou de congestion dans le modèle garantit un temps de séjour stable et prévisible, indépendamment du nombre total de client simulés.

— **Durée moyen de sortie dans P4 :**
Le traitement d'un couple client-web service dans la place P4 est immédiat ou légèrement différé (la durée sur le simulateur : une unité), ce qui reflète un mécanisme d'affectation efficace et rapide, sans surcharge.

**Durée moyenne de séjour selon les différentes affectations des Web services**

TABLE 4.4 – Durée moyen de séjour (client/web service) avec 1000 steps

| Nombre de service des Web services | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Durée moyenne de séjour /Unité de temps | 2.0 | 2.0 | 2.3 | 2.6 | 3.0 | 3.3 | 3.8 | 4.1 | 4.6 | 5.0 |

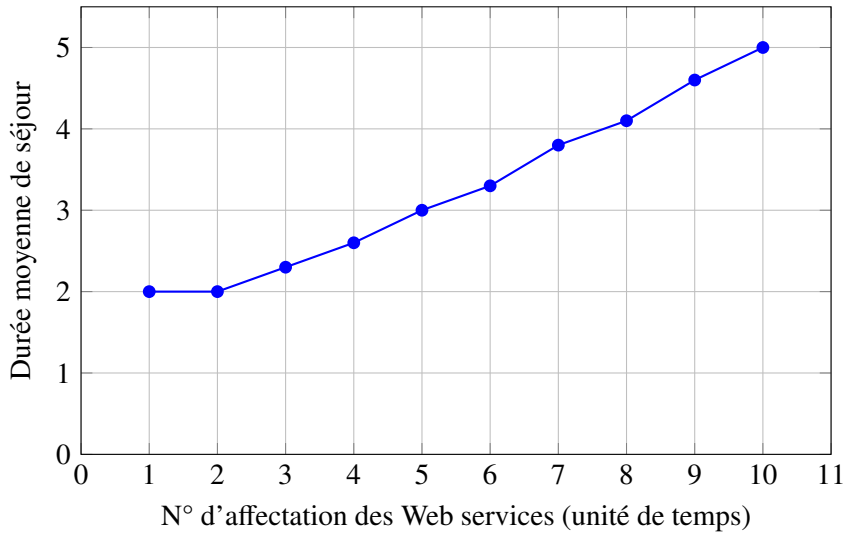



FIGURE 4.8 – Graphe de la durée moyenne de séjour selon l'affectation des Web services (1000 steps)

On remarque que la durée moyen de séjour augmente avec l'augmentation de nombre d'affectation des web services comme nous voyons sur le graphe.
On fait la simulation pour 10 clients avec 10 Web services. L'unité de temp est (@+1) pour chaque durée d'exécution d'un web service.
* Les clients sont générés et traités séquentiellement.
* À chaque nouveau groupe de clients, on augmente le nombre de Web services affectés.
* La simulation est définit pour 1000 steps.
* On suppose que les affectations sont bien enchaînées (aucun blocage ou saturation des WS).

Le tableau 4.5 nous donne des exemples de simulation (CPN Tools) de multiples affectation de Web services au clients.

TABLE 4.5 – Exemple d'affectation sur le simulateur CPN TOOLS

| N°d'affectation des Web services | Exemple d'affectation |
|---|---|
| 1 | 2‘(C1,**Ws2**) |
| 2 | 1‘(C3,**Ws1,Ws5**) |
| 3 | 1‘(C7,**Ws7,Ws4,Ws3**) |
| 4 | 3‘(C3,**Ws5,Ws4,Ws2,Ws9**) |
| 5 | 1‘(C9,**Ws1,Ws4,Ws8,Ws10,Ws6**) |
| 6 | 4‘(C2,**Ws10,Ws4,Ws7,Ws1,Ws2,Ws6**) |
| 7 | 2‘(C10,**Ws1,Ws2,Ws5,Ws9,Ws6,Ws8,Ws9**) |
| 8 | 3‘(C6,**Ws10,Ws2,Ws3,Ws7,Ws6,Ws3,Ws4,Ws8**) |
| 9 | 1‘(C5,**Ws1,Ws6,Ws3,Ws7,Ws8,Ws2,Ws4,Ws7,Ws5**) |
| 10 | 2‘(C8,**Ws1,Ws3,Ws4,Ws7,Ws6,Ws10,Ws2,Ws5,Ws9,Ws8**) |

## 4.5 Position de notre travail par rapport à l'existant

Notre travail porte sur la modélisation d'un système de Web services à l'aide d'un réseau de Petri coloré. Le modèle proposé prend en compte deux types distincts d'arrivées : celles des Web services et celles des clients, ainsi que les mécanismes de découverte et de composition des Web services. Nous avons évalué les performances du système à l'aide du simulateur CPN Tools. Les résultats obtenus tels que le nombre de clients, le nombre de clients servis et temp moyen de séjour dans le système montrent qu'un réseau de Petri coloré est adapté à la modélisation d'un tel système. Ils révèlent également que les clients sélectionnent les Web services correspondant à leurs besoins.

**TR1 [30] : Le travail de :** “ W.S. Serrai, “ Evaluation de performances de solution pour la découverte et la composition des Web services”, Thèse, Université Paris-Est, 2020 ”.
**TR2 [5] : Le travail de :** “ Bozek A., Rak T., Rzonca D., “Timed Colored Petri Net-Based Event Generators for Web Systems Simulation”, 2022 ”.
**TR3 [7] : Le travail de :** “ Hafida Naim, “Réseaux de Web service : construction, analyse et applications”, Thèse, 2017 ”.
**TR4 [15] : Le travail de :** “ N. Bernine, H. Nacer, K. Adel, D. Aissani, “Simulation et analyse

d'un Web service", Séminaire Mathématique de Bejaia (LaMos), vol.13, pp. 17–20, 2014 ".
**TR5 [14] : Le travail de :** " N. Bernine, D. Aissani, "Un Modèle pour l'évaluation des Performances d'un Système de Web Services", Séminaire Mathématique de Bejaia (LaMos), vol.14, pp. 3–7, 2015 ".
**TR7 [24] : Le travail de :** " Sahli Ramzi, " Evaluation de performances d'un système des Web services", Mémoire Master, Université de Béjaia, 2022. "
**TR6 [16] : Le travail de :** " N.Bernine Qualité de services dans la découverte et la composition des web services. Thèse de doctorat université Bejaia 2018 ."

| | TR1 [30] | TR2 [5] | TR3 [7] | TR4 [15] | TR5 [14] | TR6 [24] | TR7 [16] | **Notre modèle** |
|---|---|---|---|---|---|---|---|---|
| **Modèle** | | | | | | | | |
| RdPC | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ |
| RdPS | ✓ | | | | | | | |
| File d'attente | ✓ | | ✓ | | ✓ | | | |
| RdT | | ✓ | | ✓ | | | | |
| **Solution** | | | | | | | | |
| Analytique | | | | | | | ✓ | |
| Simulation | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Prototype | | | | | | | | |
| **Évaluation de performance** | | | | | | | | |
| Disponibilité | | | | | | | | |
| Temps de réponse | ✓ | | | | ✓ | | ✓ | |
| Coût | ✓ | | | | | | | |
| Nombre moyen | | | | | ✓ | ✓ | ✓ | ✓ |
| Débit | ✓ | | ✓ | | ✓ | | | |
| Requête client | | ✓ | | | | | | |
| Temps d'arrivée | | | | ✓ | | | | ✓ |
| TMDS | | ✓ | | | | | | ✓ |
| Durée moyenne | | ✓ | ✓ | ✓ | | | | ✓ |
| **Web service** | | | | | | | | |
| Simple | | | | ✓ | ✓ | ✓ | | ✓ |
| Composite | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ |

**Table 4.2 :** Étude comparative des travaux de l'existence de l'évaluation de performances des Web services avec notre modèle

## 4.6 L'interprétation du tableau

Plusieurs travaux ont été réalisés autour de l'évaluation des performances des Web services à l'aide des Réseaux de Petri, notamment les contributions de Sahli Ramzi (mémoire de Master, 2022) et celles de Dr Bernine Nassima (publications 2014-2015, thèse doctorale 2018). Bien que ces études aient permis de poser les bases de la modélisation de systèmes de Web services, notre travail se distingue sur plusieurs aspects majeurs, tant sur le plan de la modélisation que sur celui de la simulation et des résultats obtenus.

Premièrement, notre travail prend en compte deux types de services, en adaptant leur affectation en fonction des besoins spécifiques des clients. Cette approche permet de simuler des scénarios complexes plus proches des environnements réels. À ce niveau, nous allons plus loin que les travaux précédents qui se limitaient souvent à la gestion de services simples ou à des compositions.

Deuxièmement, en termes de méthodologie, nous avons opté pour une simulation complète et détaillée sous CPN Tools, avec une analyse fine des performances : temps moyen d'arrivée, durée moyenne de séjour, temps moyen de sortie des clients, ainsi que le nombre de clients servis. Les travaux antérieurs, bien qu'utilisant parfois les RdPC ou des extensions, restent généralement dans une perspective analytique ou se limitent à des indicateurs globaux (comme le nombre moyen de clients dans le système, requête client, ...).

Ainsi, notre contribution s'inscrit dans la continuité des recherches précédentes, tout en proposant un modèle plus riche, plus dynamique et mieux adapté à la réalité des systèmes de Web services distribués.

## Conclusion

Notre travail se distingue par une modélisation plus réaliste et une simulation approfondie des systèmes de Web services distribués. Contrairement aux approches précédentes qui traitaient un flux unique ou une structure statique, nous différencions explicitement les clients et les Web services. Cette distinction reflète mieux la dynamique réelle de l'offre et de la demande.

Nous prenons également en compte deux types de services simples et composites et adaptons leur affectation selon les besoins spécifiques des clients, ce qui permet de représenter des scénarios plus complexes que ceux abordés par les travaux antérieurs.

Enfin, notre méthodologie repose sur une simulation détaillée via CPN Tools, avec une analyse fine des performances (temps d'arrivée, de séjour, de sortie, et nombre de clients servis), là où les études précédentes se limitaient souvent à des indicateurs globaux ou à une analyse purement théorique.

# Conclusion générale

Dans ce travail, nous nous sommes intéressés à la modélisation et à l'évaluation des performances d'un système de Web services, en mettant l'accent sur la découverte et de la composition des services. Face à l'hétérogénéité croissante des systèmes distribués et à l'importance des échanges inter-applicatifs, il devient essentiel de disposer d'outils formels permettant de mesurer et d'optimiser le comportement de ces systèmes.

Nous avons d'abord présenté l'état de l'art des Web services, en détaillant les standards technologiques tels que SOAP, WSDL et UDDI, qui assurent l'interopérabilité des applications sur le réseau. Ensuite, nous avons abordé les méthodes d'évaluation des performances qui permettent d'analyser quantitativement un système.

Pour formaliser notre approche, nous avons introduit les réseaux de Petri, en particulier leur extension les réseaux de Petri colorés (RdPC) qui offrent une puissance de modélisation adaptée aux systèmes complexes et concurrents. Ces modèles permettent de représenter les différentes interactions entre clients et Web services tout en tenant compte de leur diversité et des règles de composition.

Nous avons ainsi proposé un modèle basé sur les RdPC, avec les requêtes des clients et les Web services disponibles. Les arrivées dans le système suivent une loi de Poisson et les temps moyen de service sont modélisés par une distribution exponentielle. À l'aide du simulateur CPN Tools, nous avons pu évaluer plusieurs indicateurs de performance tels que le nombre de clients dans le système et le nombre de clients servis, le temps moyen d'arrivée, la durée moyen de séjour et la durée moyen de sortie dans le système. Les résultats ont confirmé qu'un client peut être correctement pris en charge par un Web service simple ou composite en fonction de ses besoins.

Ce travail a permis de démontrer la pertinence des réseaux de Petri colorés comme outil de modélisation et de simulation dans l'étude de systèmes de Web services, en offrant une vision claire et mesurable de leurs performances.

**Perspectives**

Comme perspective nous envisageons un ensemble de travaux futurs :

— Évaluation des performance des Web services avec l'Intelligence Artificielle.

— Modélisation des Web services avec la théorie des jeux.

— Intégration d'algorithmes d'ordonnancement intelligents pour l'optimisation des Web services : Simulation par réseaux de Petri colorés.

# Bibliographie


[1] A. Boushaba, O. Mohammed, R. Benabbou, “Évaluation des performances des protocoles de routage Ad hoc”.

[2] A. C. Geniet, “Les réseaux de Petri : un outil de modélisation”, Springer-Verlag, 2e éd., 2006.

[3] A. CHOQUET-GENIET, “Les réseaux de Petri, un outil de modélisationv, Springer-Verlag, 2e édition, 2006.

[4] A. IDRISSI, “How to minimise the energy consomption in mobile ad hoc network”, Université Mohammed V, Maroc, 2012.

[5] Bozek A., Rak T., Rzonca D., “Timed Colored Petri Net-Based Event Generators for Web Systems Simulation”, [DOI :10.3390/app122312385](https://doi.org/10.3390/app122312385),2022.

[6] F. Casati, M.-C. Shan, “Event-Based Interaction Management for Composite Eservices ineFlow”, Information Systems Frontiers, 4(1), pp. 19–31, 2002.

[7] Hafida Naim, “Réseaux de Web service : construction, analyse et applications”, Thèse, 2017.

[8] H.Haïouni, N. Zaghib, “Approche mixte de modélisation”, Magister, École Doctorale Informatique de l’Est, Constantine, 2010.

[9] G. SCORLETTI, G. DINET, E. MAGAROTTO, “Réseaux de Petri“, Cours Master I, Université de Caen, 2006.

[10] K. Fujii, T. Suda, “Dynamic service composition using semantic information”, In : “Proc. 2nd International Conference on Service Oriented Computing“, USA, 2004.

[11] M. Bouali Contribution à l’analyse formelle et au diagnosyic à partir de réseaux de petri colorés avec l’accessibilité arriére 2010.

[12] M. Ioualalen, A. Aissani, “Les symétries dans les réseaux de Petri stochastiques (RdPS)”, RAIRO Recherche Opérationnelle, tome 34, n°2, 2000, pp. 237–249.

[13] M. R. Bahri, “Une approche intégrée mobile UML/Réseaux de Petri pour l’analyse des systèmes distribués à base d’agents mobiles”, Thèse doctorat, Université de Constantine.

[14] N. Bernine, D. Aissani, “Un Modèle pour l’évaluation des Performances d’un Système de Web Services”, Séminaire Mathématique de Bejaia (LaMos), vol.14, pp. 3–7, 2015.

[15] N. Bernine, H. Nacer, K. Adel, D. Aissani, “Simulation et analyse d’un Web service”, Séminaire Mathématique de Bejaia (LaMos), vol.13, pp. 17–20, 2014.

[16] N. Bernine, “Qualité de services dans la découverte et la composition des Web services”, Thèse de doctorat, Université de Béjaia, 2018.

[17] P. Hastir, B. Jodocy, “Les réseaux de Petri et leurs applications”, Mémoire Master, Université de Namur.

[18] P. Hubert, K. Jensen, R.M. Shapiro, “Hierarchies in Coloured Petri Nets”, Advances in Petri Nets, LNCS vol. 483, Springer, 1990.

[19] R. DAVID, H. ALLA, “Grafcet et Réseaux de Pétri”, Hermès, 2e édition, 1992.

[20] René David, Hassane Alla, “Du Grafcet aux réseaux de Petri : optimisation des temps d’attente des systèmes flexibles de production”, Hermès.

[21] R.Kara, ” Poroduction” cours Master II Académique, Université de Mouloud Mammeri, TiziOuzou 2011.

[22] Sadok Rezig, “Approches canoniques pour la synthèse des contrôleurs réseaux de Petri”, Thèse doctorat, Université de Lorraine, 2016.

[23] Saggadi Samira, “Optimisation des temps d’attente des systèmes flexibles de production basée sur les réseaux de Petri”, Magistère, Université de Boumerdès, 2007.

[24] Sahli Ramzi, “Évaluation de performances d’un système des Web services”, Mémoire Master, Université de Béjaïa, 2022.

[25] S. Hakmi, “Évaluation des Performances des Systèmes Prioritaires à l’aide des Réseaux de Petri Stochastique Généralisés”, Magistère, Université de Béjaia, 2011.

[26] S. Hamaci, “Étude des graphes d’évènements temporisés avec multiplicateurs en algèbre (min,+)”, Thèse de doctorat, Université d’Angers, 2005.

[27] Support de cours sur les Web services, OpenClassrooms, 2016.

[28] T. Shailesh, A. Nayak, D. Prasad, “UML-Based Performance Evaluation of Real-Time Systems Using Timed Petri Net”, Manipal Institute of Technology, 2020.

[29] Vincent Auguste, “Support de cours Réseaux de Petri”, École des Mines de Saint-Étienne, 2012.

[30] W.S. Serrai, “Évaluation de performances de solution pour la découverte et la composition des Web services”, Thèse, Université Paris-Est, 2020.

[31] Yann Morère, “Support de cours Réseaux de Petri”, 2002.

[32] Y. Mahjoub, “Étude des systèmes de transport public et réseaux logistiques par les réseaux de Petri colorés et l’algèbre (max,+)”, Thèse doctorat, 2019.

[33] Zhang and Z. Mengchu A Stochastic Petri Net Approach to Modeling and Analysis of Ad Hoc Network. University Heights.

[34] https://www.techno-science.net/glossaire-definition/Service-Web.html (consulté le 12/02/2025)

[35] http://www.dicodunet.com/definitions/normes/service-web.htm

[36] https://www.ibm.com/docs/fr/radfws/applications-web-services-overview

[37] https://www.ibm.com/docs/en/itcam-app-mgr/7.2.1 (consulté le 13/02/2022)

[38] https://www.tutorialspoint.com/webservices/web-services-characteristics.htm (consulté le 12/02/2022)

[39] https://www.clever-age.com/soap-vs-rest-choisir-la-bonne-architecture-web-ser

[40] https://arima.episciences.org/1928/pdf

[41] https://www.oracle.com/fr/cloud/definition-web-service/ (consulté le 12/02/2025)

[42] https://www.w3.org/standards/xml/core (consulté le 13/02/2022)

[43] https://www.w3.org/XML/ (consulté le 13/02/2022)

[44] https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (consulté le 13/02/2022)

[45] https://www.ibm.com/docs/en/sc-and-ds/stack-web-service-description-language

# Annexes

## Annexe 1

## CPN TOOLS

### Histoire

CPN Tools est destiné à remplacer Design/CPN qui est un progiciel répandu pour les CP-nets, Design /CPN a été publié pour la première fois en 1989 avec un support pour l'édition et la simulation de réseaux CP. CPN Tools est le résultat d'un projet de recherche, le projet CPN à été développé , à l'Université d'Aarhus de 2000 à 2010[24]. Les principaux architectes derrière l'outil sont Kurt Jensen, Søren Christensen, Lars M. Kristensen et Michael Westergaard. À partir de l'automne 2010, CPN Tools est transféré au groupe AIS, Université de technologie d'Eindhoven, Pays-Bas. L'objectif du projet CPN était de profiter de l'évolution de l'interaction homme-machine, et d'expérimenter ces techniques dans le cadre d'une refonte complète de l'IHM pour Design/CPN. La dernière version est apparue en 2015.

### Définition (CPN TOOLS)

CPN Tools est un outil d'édition, de simulation et d'analyse de réseaux de Petri colorés. L'outil propose une vérification incrémentielle de la syntaxe et la génération de code, qui ont lieu pendant la construction d'un réseau. Un simulateur rapide gère efficacement les filets non chronométrés et chronométrés[23]. Des espaces d'état complets et partiels peuvent être générés et analysés, et un rapport d'espace d'état standard contient des informations, telles que les propriétés de limite et les propriétés de vivacité.

### L'interface de CPN tools

La colonne de gauche s'appelle l'index et le reste de l'interface c'est l'espace travail. Comme le montre l'image ci-dessous.

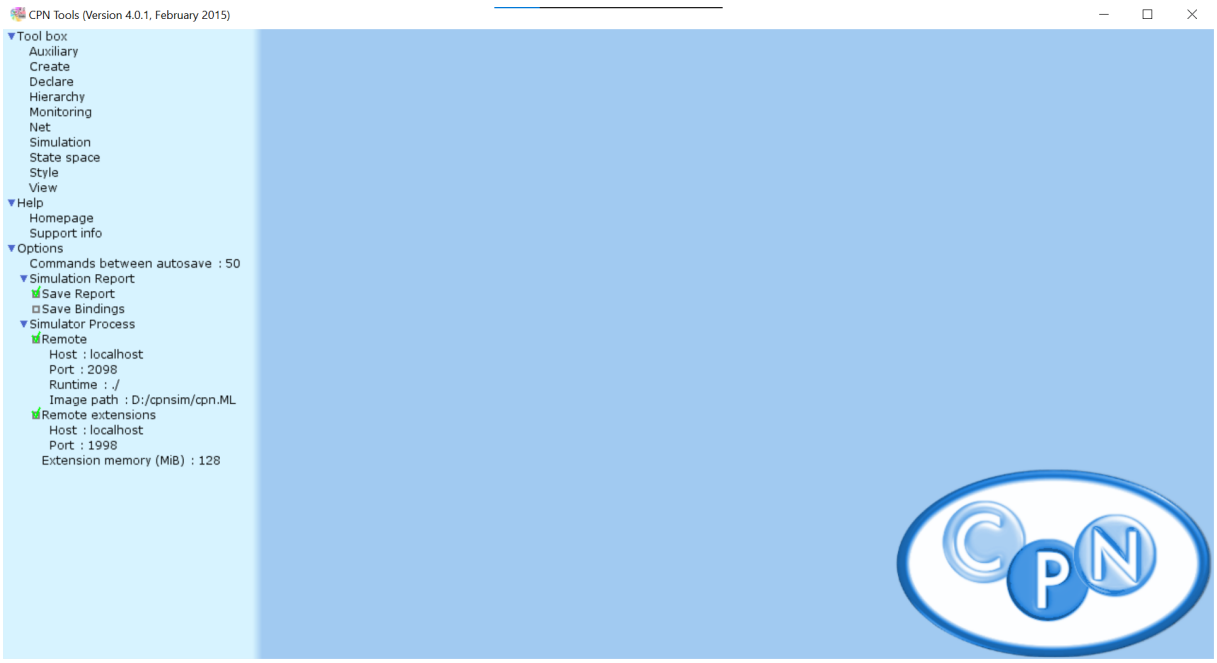



FIGURE 4.9 – Interface de CPN TOOLS

La colonne de gauche (index) contient une zone qui s'appelle outil .


FIGURE 4.10 – La boite à outils d'index

Elle même contient une liste des Palettes disponibles create(crée), simulation, style,... . Pour accéder à une palette d'outils dans l'index faire glisser une palette vers l'espace travail comme la montre la FIG 4.4.

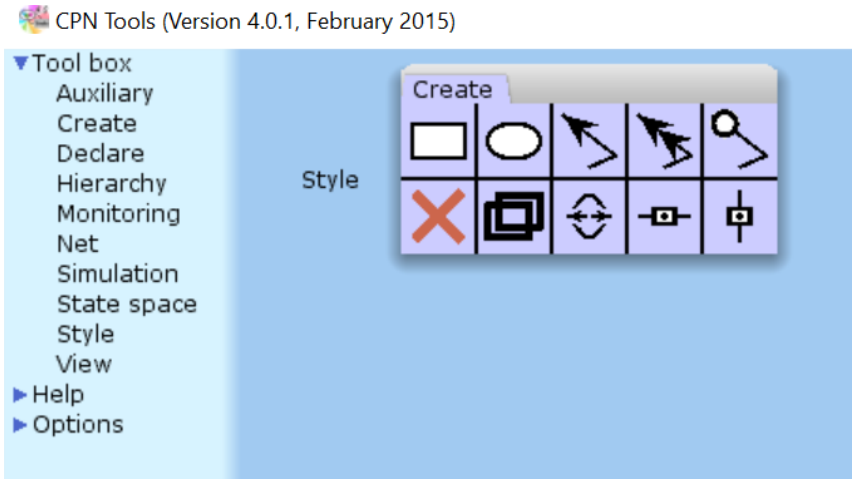



FIGURE 4.11 – Palettes d'outils dans l'espace travail

Dans l'image FIG 4.5 on vas voir touts les palette qui sont disponible dans Outil dans l'espace travail.

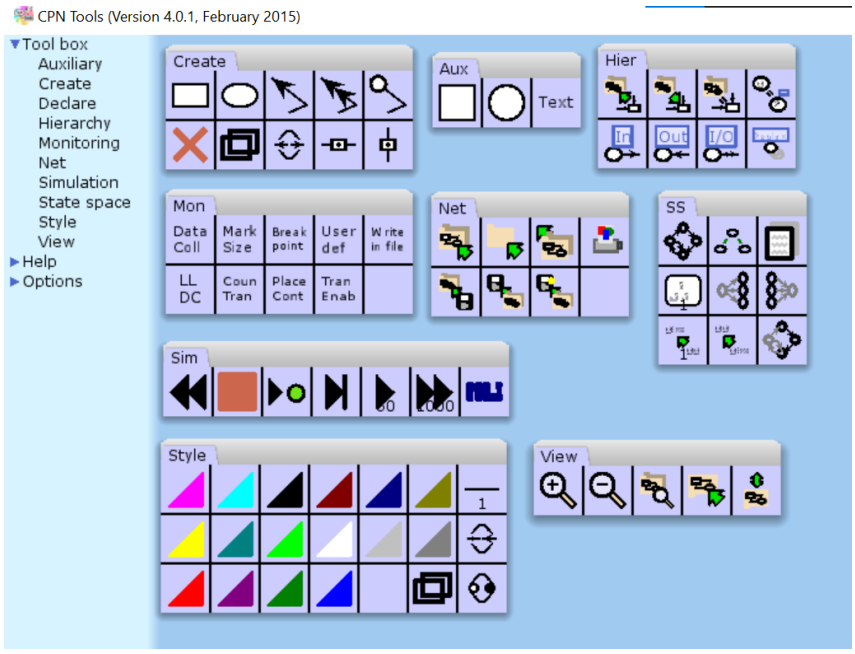



FIGURE 4.12 – Toutes les palettes dans l'espace travail

# Les palettes

• Outils auxiliaires : sont utilisés lors de la création d'éléments auxiliaires.


FIGURE 4.13 – Outils auxiliaires

• Outils de création : sont utilisés lors de la création de la structure de réseau.


FIGURE 4.14 – Outils de création

• Outils de hiérarchie : sont utilisés pour modifier la structure hiérarchique de réseau.

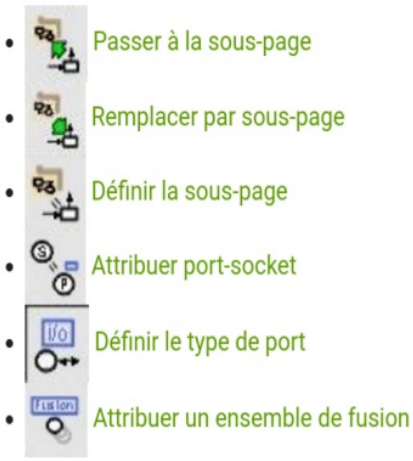



FIGURE 4.15 – Outils hiérarchie

• Outils net : sont utilisés pour charger et enregistrer des réseaux.


FIGURE 4.16 – Outils net

• Outils de simulation : sont utilisés pour simuler le réseau.


FIGURE 4.17 – Outils de simulation

• Outils d'espace d'état : sont utilisés par exemple pour calculer des espace d'état.


FIGURE 4.18 – Outils d'espace d'état

## Résumé

Dans les systèmes distribués, les Web services occupent une place centrale, rendant cruciale l'assurance de leur performance, fiabilité et disponibilité. Ce mémoire propose un modèle basé sur les réseaux de Petri colorés pour modéliser la découverte et la composition automatique de Web services en tenant compte des exigences de qualité de service (QoS).

La méthode repose sur la modélisation formelle avec CPN Tools, permettant d'analyser le comportement dynamique du système à travers des simulations et une exploration de l'espace d'états. Plusieurs scénarios ont été testés pour évaluer des indicateurs de performance tels que le temps d'attente et le taux de succès.

Les résultats montrent l'efficacité du modèle pour analyser les performances et anticiper les défaillances, bien que sa complexité augmente avec le nombre de services. Ce travail offre ainsi une base pour des améliorations futures, notamment via l'optimisation multi-critères ou l'intégration de l'intelligence artificielle.



**Mots-clés :** Web services, Réseaux de Petri colorés, QoS, Composition, Modélisation, Simulation, CPN Tools.



## Abstract

In distributed systems, web services occupy a central role, making ensuring their performance, reliability, and availability crucial. This thesis proposes a model based on colored Petri nets to model the automatic discovery and composition of web services while taking into account quality of service (QoS) requirements.

The method is based on formal modeling with CPN Tools, allowing for the analysis of the dynamic behavior of the system through simulations and state space exploration. Several scenarios were tested to evaluate performance indicators such as waiting time and success rate.

The results demonstrate the model's effectiveness in analyzing performance and anticipating failures, although its complexity increases with the number of services. This work thus provides a basis for future improvements, particularly through multi-criteria optimization or the integration of artificial intelligence.



**Keywords :** Web services, Colored Petri Nets, QoS, Composition, Modeling, Simulation, CPN Tools.