REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : RTIC16/M2/2018

Mémoire

présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : RTIC

Une modélisation formelle des services composites

Par :

CHEMMAR HOUSSEM EDDINE

Soutenu le 25/06/2018 , devant le jury composé de :

Mme	Sahli Siham	M A A	Encadreur
Mr	Meklid Abdessalem	M A A	Président
Mme	Chami Djazia	M A A	Examinateur

Remerciements

Tout d'abord, je remercie Dieu le Tout Puissant de m'avoir donné le courage et la force pour mener à bien ce travail.

Je tiens avant tout à exprimer ma profonde gratitude à Madame « Sahli Siham », Enseignant l'université Biskra, Département Informatique, qui m'a fait l'immense honneur d'être mon promoteur, pour m'avoir dirigé dans l'élaboration de cette thèse. Son encadrement, ses critique constructives, ses précieux conseils, ses relectures pointilleuses m'ont été d'une aide précieuse. Pour tout cela ainsi que pour sa confiance, ses encouragements ininterrompus, le climat agréable de travail, et sa disponibilité du début à la fin de l'élaboration de cette thèse, je le remercie vivement et qu'il trouve ici l'expression de ma considération profonde.

Je remercie également l'honorable jury, Je tiens à remercier aussi tous ceux qui m'ont soutenu, Mes chers parents, Ma chère mère, Mes frères et sœurs, et qui ont contribué de près ou de loin à l'élaboration de ce travail.

Dédicace

Je dédie ce modeste travail

A

Mes chers parents : Saddek , Amrai

A

Mes frères

A

Mes soeurs

A

Fils et filles de mes frères et soeurs

A

Toute ma famille

A

Sara,Khalil,Tarik

A

Tous mes amis

Résumé

De nos jours, les services web sont devenus très utilisés notamment par les entreprises pour rendre accessible leurs métiers et leurs données via le web. La composition des services web est un sujet qui suscite l'intérêt des chercheurs, elle offre la possibilité de traitement de problèmes complexes même avec des services simples existants tout en coopérant entre eux. Toutefois, cette tâche reste très complexe et nécessite pour son accomplissement des techniques formelles.

L'objectif principal de notre travail est de fournir une modélisation formelle de services Web composite de type orchestration « BPEL ». Nous avons également transformé le fichier BPEL en réseau pétri coloré, avec la possibilité de vérifier ce RdPC afin de vérifier le service composite.

Mots-clés : service web, composition de services web, modélisation formelle de services Web composite, orchestration, BPEL, CPN Tools.

« Glossaire »

C

CPN: colored petri nets

B

B2B: Business-to- Business

B2C: Business-to- Consumer

BPMI: Business Process Management Initiative

BPML: Business Process Modeling Language

BPEL4WS: Business Process Execution Language for web Services

C

COM: Component Object Model

COBRA: Common Object Request Broker Architecture

D

DAML: DARPA Agent Markup Language

DCOM: Distributed Component Object Model

F

FTP: file transfer protocol

H

HTML: Hyper Text Markup Language

HTN: Hierarchical Task Network

HTTP: Hyper Text Transfer Protocol

O

OWL: Ontology Web Language

OWL-S: Ontology Web Language for Services

R

RPC : Remote procedure call

RDPC : Réseaux de Pétri colorés

RdP: réseaux de Pétri

S

SMTP: Simple Mail Transfer Protocol

SOA: Service Oriented Architecture

SOAP: simple Object Access protocol

U

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

UDDI: Universal Description Discovery and Integration

W

W3C : *World Wide Web Consortium*

WSCI: web services Choreography Interface

WSDL: Web Services Description Language

WSDL-S: Web Services Description Language-Semantic

WSFL: Web Services Flow Language

WS-I ! Web Service Interoperability

X

XML: Extensible Markup Language

XLANG : XML LANGuage

XSLT : Extensible Stylesheet Language Transformations

XSD : XML Schema

XPath : XML Path Language

«Table des Matières »

Chapitre 2 : « La modélisation formelle des services composite »

Chapitre 3 : « La conception de system »

Chapitre 4 : « Implémentation »

« Conclusion générale »

« Table des Figures »

« Table des Figures »

« Listes des tableaux »

« Introduction générale »

Introduction générale

De nos jours l'internet est quasi indispensable, et cela n'est en aucun cas un hasard, car simplement internet est avant tout le moyen le plus utile pour se communiquer, partager des données et des informations, se divertir d'une manière simple facile, rapide et accessible au public. Elle permet une prise de décision dans un délai meilleur et une actualisation rapide des informations. Cela a conduit à l'émergence d'un type d'application Web qui fournit des services électroniques entre une application et une autre application ou un système et un autre système, ces applications s'appellent des services web.

Au fil du temps, les services Web ont été assemblés pour réaliser des services spécifiques afin de répondre aux besoins des consommateurs, ces services s'appellent services web composites. La composition des services web permet à plusieurs services de se regrouper et d'interconnecter pour former un seul service du point de vue utilisateur qui peut fournir une nouvelle fonctionnalité à partir des services existants plus simples.

L'objectif de notre projet est la modélisation formelle des services composites afin de fournir des formalismes spécifiant le fonctionnement du service Web composite, comment ses services interagissent et communiquent. Et pour atteindre cet objectif nous allons organiser notre mémoire en quatre chapitres :

Le premier chapitre présente une aperçue sur les services web tel que l'architectures orientées services (SOA), les principes de base des services web et la composition des services web

Le deuxième chapitre est consacré à la modélisation formelle des services composites tels que les outils et les formalismes utilisés pour la modélisation. Ce chapitre aussi contient une synthèse approfondi de ces formalismes afin de justifier notre choix de Réseaux de Petri Coloré pour la modélisation des services composites.

Le troisième chapitre présente la conception de notre system sous formes des sections comme l'analyse du langage BPEL, transformation de BPEL en RDPC et architecture du système ... etc.

Finalement dans le quatrième chapitre nous détaillerons l'implémentation de notre système, l'environnement de développement et les outils utilisés. Pour bien illustrer notre

implémentation nous allons expliquer les différentes interfaces du système réalisé et nous allons discuter les résultats obtenus.

Chapitre 1

« Les services web et la composition des services web »

1. Introduction

Le web qu'a était à l'origine un simple moyen de partage d'information, a eu un développement considérable qui lui a permis de devenir un moyen universel pour externaliser les applications des entreprises, donnant ainsi naissance au concept d'intégration d'applications. En effet, ce concept permet aux entreprises de collaborer, coopérer et d'interagir entre elles, en produisant ainsi des processus a valeur ajoutée, sans aucune influence sur leur autonomie. Cependant, assurer une collaboration transparente des applications autonomes reste un défi majeur.

L'architecture orienté service est apparue dans les dernières années pour pallier aux problèmes liés au développement des applications distribuées complexes à partir des entités logicielles appelées services. L'utilisation de cette architecture permet la communication et l'intégration de plusieurs applications qui sont développées dans des environnements hétérogènes. A cet égard, nous allons présenter dans le présent chapitre l'émergence l'architecture orientée service (SOA) et les différents acteurs intervenants dans cette architecture. Ensuite, nous allons exposer la technologie des services web. Enfin, nous allons définir le processus de composition des services, les différents types de la composition et les besoins qui en découlent.

2. Les services web

2.1. L'Architecture Orientées Services (SOA)

Une Architecture Orientée Service est un paradigme fondée sur la description des services et sur la description de leurs interactions [4]. Les caractéristiques principales d'une architecture orientée service sont le couplage faible entre les services, l'indépendance par rapport aux aspects technologiques et la mise à l'échelle. La propriété de couplage faible implique qu'un service n'appelle pas directement un autre service. En effet, les interactions sont gérées par une fonction d'orchestration [5]. La réutilisation d'un service est alors plus facile, du fait qu'il n'est pas directement lié aux autres services de l'architecture dans laquelle il évolue. L'indépendance par rapport aux aspects technologiques est quant à elle, obtenue grâce aux contrats d'utilisation associés à chaque service. En effet, ces contrats sont indépendants de la plate-forme technique utilisée par le fournisseur du service. Enfin, la mise à l'échelle est rendue possible grâce à la découverte et à l'invocation de nouveaux services lors de l'exécution.

D'un point de vue métier [6] « L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d'un environnement métier agile pour un intérêt concurrentiel ».

Selon [7] une autre définition technique plus large de SOA, « L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoir faires distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs faires pour produire le résultat désiré avec des pré-conditions et des buts mesurables».

Une autre Définition est présentée par [8], « L'architecture SOA permet l'intégration d'applications et de ressources de manière flexible, en représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant à un service d'échanger des informations structurées (messages, documents, objets métier), coordonnant et en organisant les services, afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement. ».

2.1.1. Service

Le service est la brique de base de la SOA. Il représente une entité logicielle fonctionnelle déployée et invocable à distance. Il est difficile de trouver une définition exacte du terme « Service » parce que plusieurs définitions existent. Dans [2] un service est défini comme une entité qui représente certaines fonctionnalités (application fonctionnelle, transaction commerciale, un service du système de base, etc.) exposée en tant que composant d'un processus métier. Un service est défini à l'aide d'une interface qui expose les fonctionnalités et masque les détails de mise en œuvre sous- jacents. Le service doit être sans état pour assurer qu'il ne dépend pas de l'état ou du contexte d'autres services. Les services sont appelés via des protocoles de communication bien définis qui mettent l'accent sur l'interopérabilité et la transparence de localisation.

2.1.2. Environnement d'intégration des services

L'architecture orientée services offre un environnement d'intégration et d'exécution des services dont les éléments peuvent être divisés en deux catégories [10**]** :

- Les éléments traitant les aspects fonctionnels
- Les éléments traitant les aspects qualité de service

Comme illustré dans la figure 1, les aspects (ou éléments) fonctionnels incluent :

- ✓ **Le transport** : est le mécanisme utilisé pour déplacer les requêtes de service du consommateur de services au fournisseur de service, et les réponses des services du fournisseur de service au consommateur de service.
- ✓ **Le protocole de communication** : est un mécanisme utilisé par le fournisseur et le consommateur de service pour communiquer le contenu des requêtes et des réponses.
- ✓ **La description de service** : est un schéma spécifique pour décrire les fonctionnalités du service, la manière dont il doit être invoqué et les données requises pour invoquer le service correctement.
- ✓ **La composition de services** : est une collection de services, invoqués dans une séquence particulière, pour répondre à un besoin particulier. Il est à noter qu'une composition de services est considérée comme étant un service en elle-même.
- ✓ **Le registre (ou annuaire) de services** : est un répertoire des descriptions de service et des données. Il peut être utilisé par les fournisseurs de services afin de publier leurs services, comme il peut être utilisé par les consommateurs de services afin de découvrir les services disponibles.

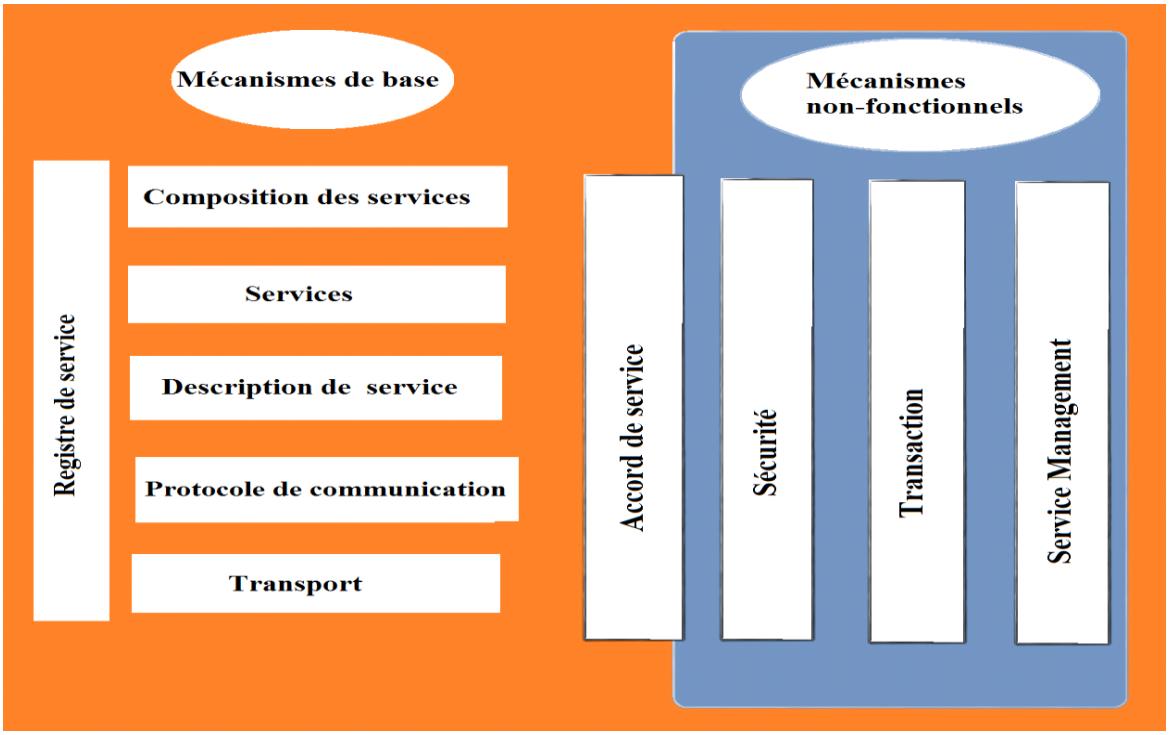


Figure 1.1 : Environnement d'intégration de services [10]

Les aspects (ou éléments) concernant la qualité de services quant à eux incluent :

- ✓ **L'accord de service (ou contrat de service)** : est un ensemble de règles ou de conditions sous lesquelles un fournisseur de services rend le service disponible aux consommateurs. Il y a des aspects du contrat qui sont fonctionnels, et des aspects qui concernent la qualité de service comme par exemple, un niveau de performances (temps de réponse, fiabilité, etc.).

- ✓ **La sécurité** : est l'ensemble des règles qui peuvent être appliquées lors de l'identification, de l'autorisation, et du contrôle d'accès des consommateurs qui invoquent les services. Le consommateur peut aussi demander l'utilisation de règles de sécurité. Par exemple, dans le cas où il veut garantir l'intégrité des données transmises.

- ✓ **Transaction** : un environnement d'intégration de services peut contenir un élément chargé de garantir la cohérence des données utilisées et des résultats retournés par une collaboration (ou composition) de services.

- ✓ **Management de service** : est le processus chargé de surveiller et de contrôler de manière proactive le comportement d'un service ou d'un ensemble de services. Le management de service s'opère sous des contraintes qui dépendent du contexte commercial.

2.1.3. Modèle fonctionnel du SOA

Le modèle SOA est un modèle (abstrait) qui définit un système par un ensemble de composants logiciels distribués qui collaborent afin de réaliser une fonctionnalité globale préalablement établie. Ce modèle permet de répondre aux problèmes d'intégration dans les architectures logicielles en facilitant l'ajout et la suppression d'un module sans remettre en cause toute l'architecture [11].

L'architecture SOA vise trois objectifs importants [12] :

- Identification des composants fonctionnels,
- Définition des relations entre ces composants
- Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

Cette architecture s'articule autour d'un « médiateur » central qui garantit la publication des services proposés par des fournisseurs au profit des futurs clients. Les trois opérations de base de cette architecture sont la *publication* de la description, la découverte du service suite au déclenchement de la *recherche* et l'interaction entre le client et le fournisseur par *invocation*.

La Figure 1.2 montre le modèle fonctionnel de l'architecture SOA.

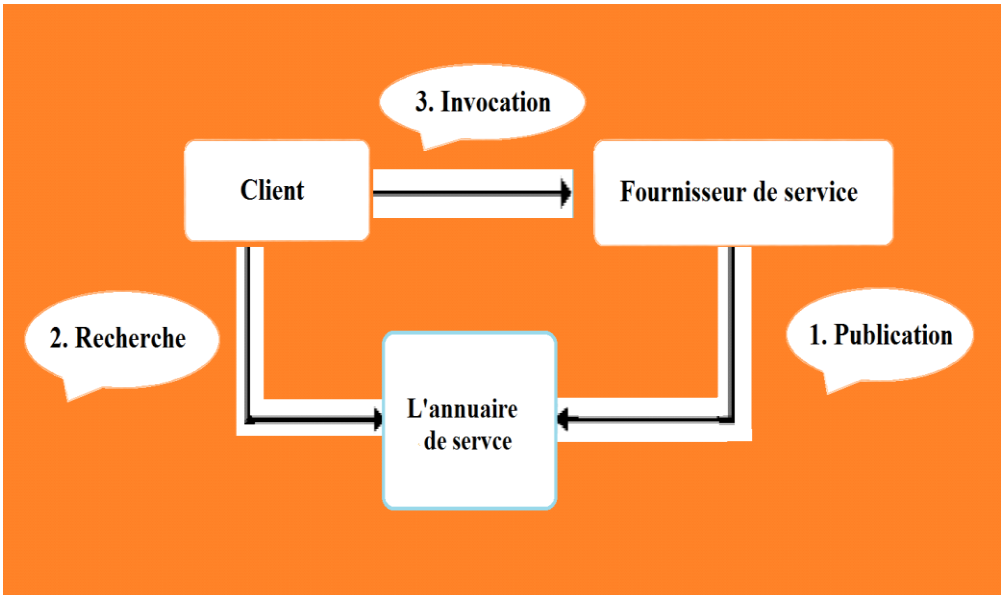


Figure 1.2 : Modèle Fonctionnel de l'architecture SOA. [12]

Ces interactions sont : Publication, découverte et invocation.

- ✓ ***Publication de service*** : permet aux fournisseurs de service d'enregistrer les descriptions de services qu'ils fournissent afin qu'elles puissent être découvertes et invoquées par un ou plusieurs consommateurs de service. Cette interaction s'effectue entre les fournisseurs et l'annuaire de service.

- ✓ ***Recherche et découverte d'un service*** : Selon leurs besoins, cette action permet aux consommateurs de service de rechercher et découvrir les fournisseurs de service qu'ils requièrent. Cette interaction a lieu entre le consommateur et l'annuaire de service. l'opération de découverte peut-être réalisée à deux phases différentes du cycle de vie de l'application : à la phase de conception pour récupérer la description de l'interface du service afin de développer l'application et à l'exécution pour récupérer l'emplacement du service afin de faire la liaison.

- ✓ **Invocation** : Après avoir consulté la description du service découvert et fait la liaison avec le fournisseur du service choisi, le consommateur de service procède à l'invocation de ce service en fonction des informations présentes dans la description de service. Cette interaction s'effectue entre le consommateur et le fournisseur de service au moment de l'exécution de l'application.

L'architecture SOA comprend trois acteurs : [12]

- ✓ **Le fournisseur du service** : il désigne l'entité propriétaire du service. D'un point de vue technique, un fournisseur est constitué par la plate-forme d'accueil du service.

- ✓ **Le client ou demandeur du service** : C'est le consommateur de service. D'un point de vue technique, le service client est constitué de l'application qui va rechercher et invoquer un service.

- ✓ **L'annuaire de service** : Correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ; et des facilités de recherche de services à l'intention des clients. En d'autres termes, l'annuaire joue le rôle d'intermédiaire entre les clients et les fournisseurs de services.

2.2. Service web

2.2.1. Définition W3C

Le W3C est un sigle utilisé pour définir le *World Wide Web Consortium* qui est une organisation non lucrative permettant de définir des standards pour les technologies liées aux web. Il est important de noter que du point de vue européen, les standards fournit par cet organisme ne sont que des recommandations et non des normes standardisés. Cela permet de guider les technologies du web dans une même direction sur le long terme et ainsi améliorer leur compatibilité. Par la même occasion cela permet de mettre en place des directives sur des sujets variées (exemple: l'accessibilité). [3]

2.2.2. Définition services web

Il existe de nombreuses définitions d'un service web. Citons la définition de l'organisme de normalisation du World Wide Web Consortium (W3C):

"Un services web est un système logiciel conçu pour permettre l'interopérabilité entre les machines sur un réseau. Il possède une interface qui décrit, dans un format normalisé, le moyen de communiquer avec la machine (par exemple : WSDL). D'autres systèmes interagissent avec les services web, conformément à l'interface, en utilisant les messages SOAP envoyés par le protocole HTTP et écrits en XML, en liaison avec d'autres normes standards du Web."

Une autre définition est présentée par [13], "Un services web est une application informatique possédant une URI (Uniform Resource Identifier), hébergée par un serveur d'applications, qui est composée de procédures dont l'exécution représente un service proposé à un autre programme informatique (nommé client) et qui est accessible sur internet par l'utilisation de protocoles standards (HTML, XML)".

Les services web créent une architecture de type client/serveur dans laquelle des clients (ordinateur de bureau, ordinateur portable, téléphone portable…) utilisent, via internet, des procédures qui sont stockées sur un serveur d'applications (Figure 1.3).

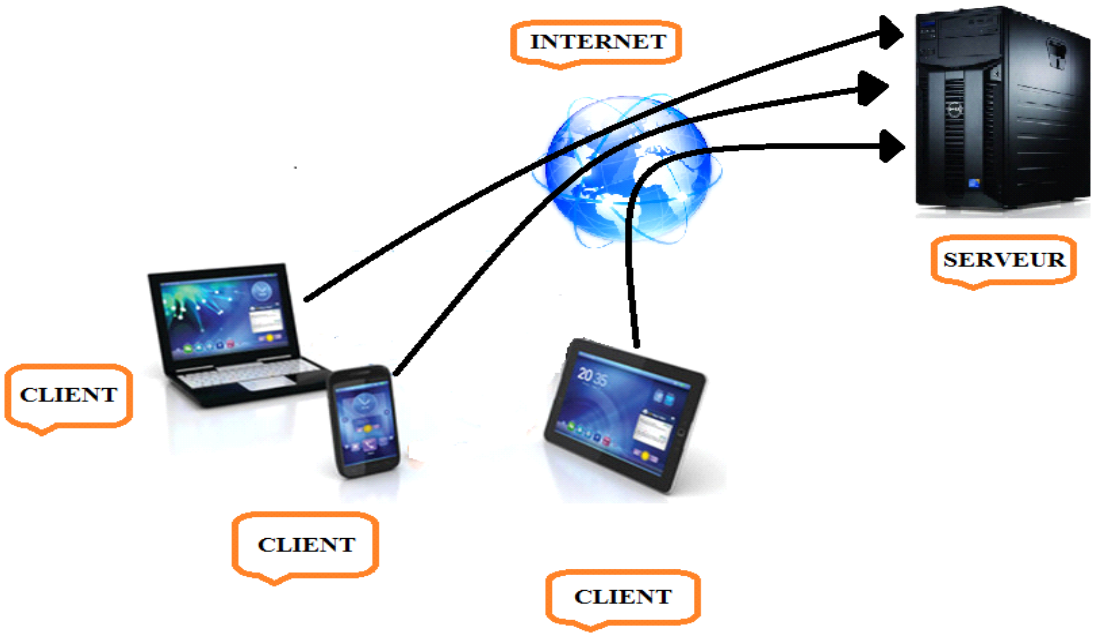


Figure 1.3. Architecture client/serveur [13]

2.2.3. Les composants

De manière schématique, trois composants (Figure 1.4) sont nécessaires dans un service web [13] :

- Un protocole pour décrire le service (idéalement il doit lister les méthodes disponibles et leurs arguments…) ;
- Un protocole décrivant la composition des messages ;
- Un protocole de transport pour faire circuler les informations sur Internet.

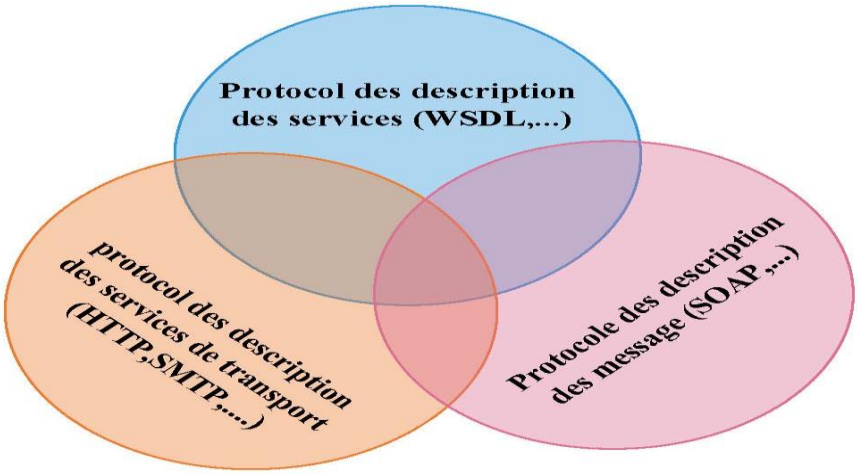


Figure 1.4 : Les différents composants d'un service web [13]

2.2.4. Caractéristiques des services Web

Plusieurs acteurs définissent les services Web par des caractéristiques technologiques distinctives, qui sont [15].

- **Un service Web est une application logicielle qui est reconnue par un URI**

URI est la façon d'identifier un point de contenu sur le Web, que ce soit une page de texte, une vidéo, une image ou un programme. La forme la plus commune des URI est appelé Uniform Resource Locator (URL).

Le service web est donc accessible en spécifiant son URI, Par exemple cette URI:http://www.w3.org/Icons/WWW/w3c_main.gifidentifie un fichier qui peut être consulté à l'aide de l'application du protocole Web, ("http: //") qui est logé sur un ordinateur nommé www.w3.org, Le fichier se trouve dans le chemin d'accès "/Icons/WWW/w3c_main.gif."

- **Capacité des interfaces et liaisons d'être publiées, localisées et invoquées via le langage XML**

Les principales tâches d'un service Web sont : La publication dans un registre, la localisation en interrogeant le registre qui l'héberge et l'invocation par un ou plusieurs services Web après sa localisation. Ces tâches sont réalisées via l'utilisation d'XML

- **Capacité d'interagir avec les composantes des logiciels via des éléments XML avec l'utilisation des protocoles Internet standards**

Un service Web est créé pour être utilisé et interagir avec d'autres logiciels contrairement à une page Web, ou à une autre application qui n'utilise pas les services Web. C'est l'interopérabilité basée sur l'utilisation de l'XML et les protocoles Internet standards, par exemple, HTTP, SMTP, FTP, . . . etc.

- **Composante logicielle légèrement couplée à interaction dynamique**

Un service Web ayant un programme qui permet de l'invoquer est appelé consommateur de service Web. Le service Web et son consommateur sont indépendants l'un de l'autre. Si une modification est à faire sur le consommateur, on n'a pas besoin de connaitre la machine, le langage de programmation, le système d'exploitation ou autre paramètre, afin d'établir à nouveau une communication entre le service Web et son consommateur. Le consommateur possède une fonctionnalité qui correspond à faire une localisation et une invocation sur le

service Web, au moment de l'exécution du programme de service Web de manière automatique.

2.2.5. Principe de fonctionnement des services Web

Le principe de fonctionnement des services Web est représenté dans la Figure 1.5. il est fondé sur le model fonctionnel du SOA et sur des standards basés sur XML.

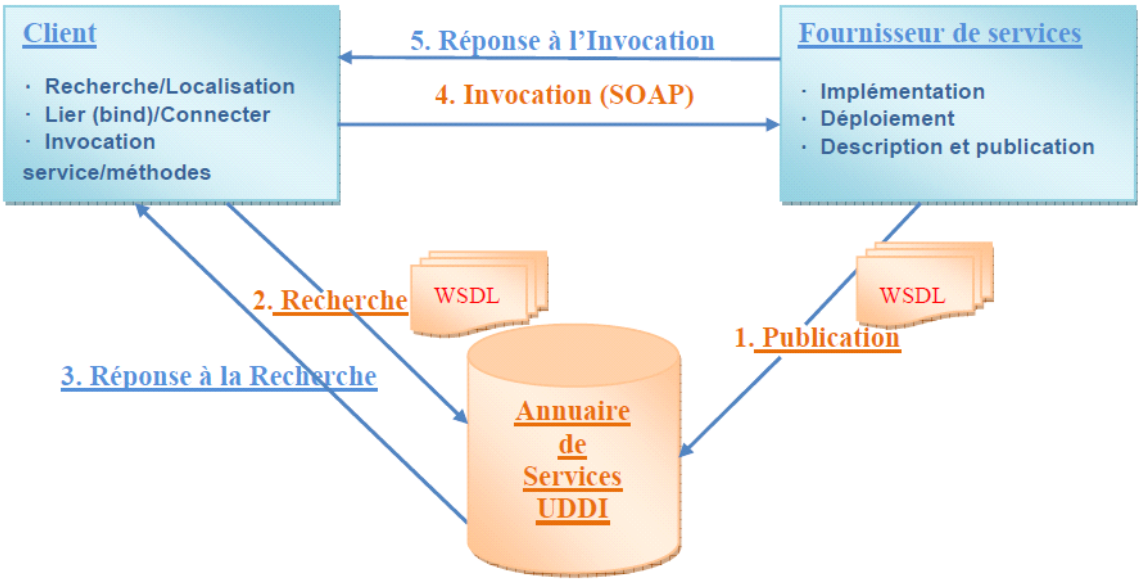


Figure 1.5 : Principe de Fonctionnement des Services Web [16]

Les interactions dans lesquelles les trois acteurs (fournisseur, client, annuaire) peuvent s'engager sont décrites par les cinq opérations suivantes [16] :

- ✓ **Publication**: le fournisseur de services déploie le SW sur un serveur et génère une description du service (WSDL : Web Service Description Language). Il doit la publier dans l'annuaire de services (UDDI : Universal Description, Discovery and Integration) afin de permettre aux futurs utilisateurs de prendre connaissance des caractéristiques du service. Cette opération consiste en la fourniture des informations concernant le fournisseur (Information commerciales), relatives au service lui-même (les fonctions assurées) ainsi que les informations techniques (méthode d'invocation, adresse…etc.).

- ✓ **Recherche du service** : le client du service Web lance une recherche auprès de l'annuaire (UDDI) qui peut être public ou privé par l'envoi d'une requête encapsulée dans une enveloppe SOAP (Simple Object Access Protocol) afin de rechercher le(s) service (s) répondant à ses besoins. Dans le cas de plusieurs réponses, des techniques de sélection de

services sont offertes (premier trouvé, aléatoirement) ou sur la base de paramètres techniques telle que la Qualité de service (QdS) (Contraintes non fonctionnelles).

- ✓ **Réponse à la recherche**: Le résultat de la recherche dans l'annuaire est un fichier WSDL qui sera transmis, dans un message SOAP, au client demandeur. Ce fichier contient la description du service et les informations techniques nécessaires pour son invocation.

- ✓ **Invocation**: Le client utilise la description du service qui existe dans le fichier WSDL, et interagit directement avec le fournisseur du service. Il envoie un message SOAP, contenant en plus du (des) nom(s) de(s) l'opération(s) à exécuter et de ses paramètres, l'adresse (URI) et le type de protocole de communication (HTTP). Techniquement, cette invocation est basée sur un appel de procédure à distance (RPC).

- ✓ **Réponse à l'invocation**: Le fournisseur réagit à la réception du message d'invocation de la part du client en envoyant au client la réponse résultant de l'exécution de la procédure. Cette réponse est transmise sous la forme d'un document XML via SOAP.

2.2.6. Les technologies des Web Services :

XML, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description, Discovery and Integration) sont les technologies dominantes des Web Services. Fondamentalement, SOAP, WSDL et UDDI sont des technologies issues de l'intérêt parmi des membres de la communauté Internet à développer un mécanisme pour échanger des documents XML sur le Web entre systèmes d'information.

2.2.6.1. Le langage XML (eXtensible Markup Language)

XML est une famille de technologies développées au sein du W3C (World Wide Web Consortium). XML est née de la tentative de mettre SGML (*Standard Generalized Markup Language*) sur le Web. La première spécification de XML est apparue en février 1998 et se concentre sur les données, contrairement à HTML (par exemple) qui focalise sur la présentation. XML permet donc de transformer Internet d'un univers d'information et de présentation de sites Web statiques à un univers Web programmable et dynamique, centré sur les données. [17]

La technologie des services web a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages

entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des services web 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). [18]

Parmi les spécifications XML, nous soulignons :

- ✓ XSD (XML Schema) : c'est un langage qui sert à décrire formellement un vocabulaire [19].
- ✓ XSLT (Extensible Stylesheet Language Transformations) : est utilisé pour transformer un document XML basé sur un certain schéma en un autre document XML qui peut être un document lui-même basé sur un autre schéma [20].
- ✓ XPath (XML Path Language) : fournit une syntaxe d'expressions utilisées pour créer des chemins de localisation [21].

```
<personne>
    <nom>
        <prenom>Houssem</prenom>
        <nomdefamille>Chemmar</nomdefamille>
    </nom>
    <adresse>
        <numero>28</numero>
        <rue> rue des Lakhdir</rue>
        <ville>Biskra</ville>
        <pays>Algérie</pays>
    </adresse>
</personne>
```

Figure 1.6 : Exemple de document XML.

2.2.6.2. La couche de Transport

Cette couche s'intéresse aux protocoles de transport de bas niveau, ces derniers vont transporter les requêtes et les réponses échangées entre services. Le protocole le plus utilisé et recommandé par le consortium WS-I (Web Service Interoperability) est l'HTTP, mais

d'autres implémentations peuvent utiliser un autre ensemble de protocoles tels que : FTP, JMS (java messagerie services), SMTP, etc.

2.2.6.3. La couche de Communication (SOAP)

Cette couche spécifie les protocoles d'échanges de documents XML entre le service web et ses clients, elle caractérise aussi le mode d'échange (s'il est bloquant ou non). Le protocole SOAP est adopté comme un standard pour la messagerie entre les services web.

SOAP (Simple Object Access Protocol), est un protocole de dialogue par appels de procédures à distance entre objets logiciels. Sa syntaxe d'utilisation est fondée sur XML et ses commandes sont envoyées sur Internet par l'intermédiaire du protocole HTTP mais aussi SMTP et POP sous forme de texte structuré. [22]. Le protocole SOAP est une note du Consortium W3C dont Microsoft fait partie, mais qui n'est pas spécifique à Microsoft et Windows. IBM a également participé à l'élaboration de ce protocole. Bien qu'il soit utilisable avec d'autres protocoles de transport, HTTP est le plus couramment utilisé. Le deuxième standard, XML, utilisé pour la structuration des données sous forme de messages est quant à lui le seul utilisé. SOAP est constitué de deux parties : une enveloppe XML, et un entête d'un protocole de transport. La spécification du protocole SOAP ne donne aucune indication sur le mécanisme de transport du message.

L'enveloppe XML contient à son tour trois sous éléments :

- ✓ L'entête « Header »: est optionnel, il peut contenir des informations de sécurité (telles que les signatures électroniques), des informations transactionnelles, des informations de traçabilités, etc.
- ✓ L'élément « Body »: est obligatoire, il contient les éléments suivants :
  - Soit le nom de méthode, avec les données correspondantes, ou un simple document XML, pour le cas d'une requête.
  - Soit les valeurs de retour pour le cas d'une réponse.
- ✓ L'élément « Fault »: est optionnel, il fournit des informations sur d'éventuelles erreurs survenues lors de l'analyse du message.

Extrait de code définition type d'envelope SOAP :

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
      <soap:Header>
          <!-- en-tête -->
      </soap:Header>
      <soap:Body>
          <!-- corps -->
      </soap:Body>
</soap:Envelope>
```

Figure 1.7 : représente le format standard d'un message SOAP. [63]

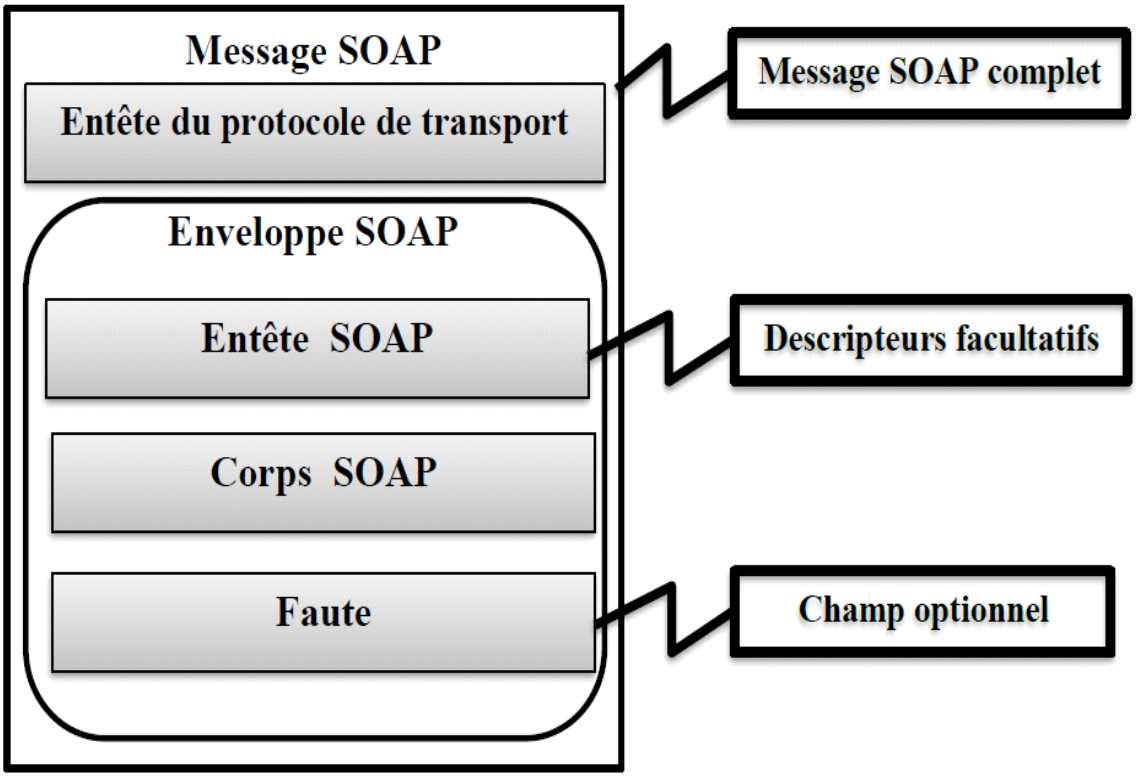


Figure 1.8 : Schéma d'un message SOAP [63]

2.2.6.4. La couche de Description (WSDL)

Les premiers travaux du W3C concernant un langage de description universelle de services Web a vu le jour en 2001 lors de l'émergence de cette technologie. Il a fallu trouver un langage de description utilisable et compréhensible par le plus grand nombre afin que les

services Web conçus soient interopérables. Le W3C a rapidement proposé le langage WSDL (Web Service Description Language) [23]. La description d'un service doit inclure la définition des composants nécessaires au protocole de communication (SOAP pour les services Web) et à l'interaction avec un client ou un autre service Web. Les problématiques de réutilisation et d'interaction ont guidé le W3C afin de définir les catégories d'information à prendre en compte dans la description d'un service Web.

Les différents éléments décrits dans WSDL sont les suivants :

- Les opérations proposées par le service Web ;
- Les données et messages échangés lors de l'appel d'une opération ;
- Le protocole de communication ;
- Les ports d'accès au service.

Dans WSDL, il existe une séparation entre deux niveaux indépendants, respectivement nommés abstrait et concret. Le niveau abstrait regroupe les informations pouvant être réutilisées (non spécifique à un service), tandis que le niveau concret est constitué de la description des protocoles d'accès au service Web (information particulière à un service). Le niveau abstrait est utilisé principalement lors du processus de sélection, tandis que le niveau concret est seulement utilisé lors de l'invocation des méthodes du service Web. [41]

❖ **Le niveau abstrait :** décrit les informations propres aux méthodes proposées par le service, ainsi que les informations traitant des messages et des données échangés lors de l'invocation du service. Si deux services proposent les mêmes méthodes, le niveau abstrait de description WSDL peut être réutilisé. Ce niveau est composé des informations suivantes :

✓ **Les types de données :** Le document WSDL permet de décrire les types de données échangées. WSDL supporte les types élémentaires (tels que les entiers, les chaînes de caractères) et les types complexes. Si les données échangées possèdent une structure particulière (i.e. un type complexe), il est possible de les décrire par l'intermédiaire d'un schéma XML [24].

✓ **Les messages :** Un message correspond aux données qui sont véhiculées selon les méthodes invoquées. Chaque opération du service possède deux définitions de message : la première correspond à la requête et la seconde correspond à la réponse. La description

d'un message contient le nom de l'élément en paramètre – d'entrée ou de sortie selon le type du message et son type.

- ✓ **Les opérations :** Une opération représente une unité de travail, c'est-à-dire une méthode proposée par le service Web décrit. Chaque opération est identifiée par son nom.

- ❖ **Le niveau concret :** décrit la manière dont le client accède à un service en particulier, et, est de ce fait, non réutilisable (propre à un service unique). Les informations décrites dans le niveau concret sont les suivantes :

  - ✓ **Le protocole de communication :** La description des protocoles de communication permet de définir le protocole à utiliser pour l'appel des méthodes du service. Si nécessaire, le document WSDL peut contenir autant de descriptions de protocole que d'opérations, étant donné que le protocole de communication peut être différent pour chaque opération du service décrit.
    - ✓ **Les ports d'accès au service :** Dans un document WSDL, l'accès au service est défini par une collection de ports d'accès. Chaque port représente la localisation du service (i.e. son URL). Un même service peut être accessible sur plusieurs ports différents.

Les informations contenues dans WSDL constituent la description du profil fonctionnel du service. Avec WSDL, le client peut invoquer le service par le biais de sa description abstraite (méthodes disponibles, paramètres d'entrée et sortie) et concrète (description des protocoles de communication et des points d'accès du service).

- ❖ **Structure d'un document WSDL**

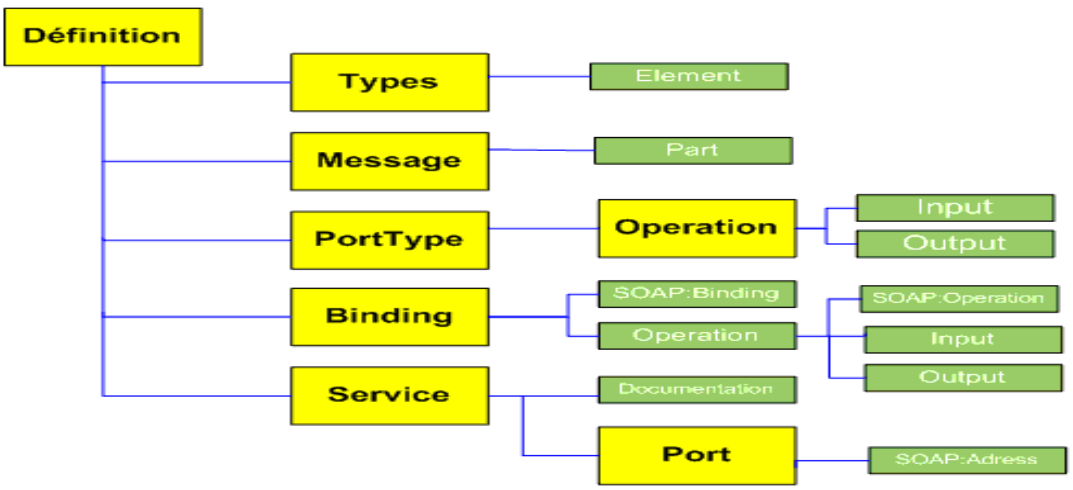


Figure 1.9 : Structure d'un document WSDL. [14]

Un fichier WSDL contient donc sept éléments. [25]

- **<Types> :** fournit la définition de types de données utilisés pour décrire les messages échangés.
- **<Messages> :** représente une définition abstraire (noms et types) des données en cours de transmission.
- **<PortTypes> :** décrit un ensemble d'opérations. Chaque opération à zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
- **<Binding> :** spécifie une liaison entre un <portType> et un protocole concret (SOAP, HTTP...).
- **<Service> :** indique les adresses de port de chaque liaison.
- **<Port> :** représente un point d'accès de services défini par une adresse réseau et une liaison.
- **<Opération> :** c'est la description d'une action exposée dans le port.

La spécification WSDL décrit quatre types d'opérations [25] :

- ✓ ***Opération à sens unique (One-way) :*** Le service reçoit un message. L'opération a donc un seul élément <*input*>.
- ✓ ***Opération de type requête – réponse (Request-response):*** Le service reçoit un message et envoie une réponse. L'opération a donc un élément <*input*>, suivi d'un élément <*output*>. Pour encapsuler des erreurs, un élément optionnel <*fault*> peut également être spécifié.
- ✓ ***Opération de type réponse sollicitée (Solicit-response):*** Le service envoie un message et reçoit une réponse. L'opération a donc un élément <*output*>, suivi d'un élément <*input*>. Pour encapsuler des erreurs, un élément optionnel <*fault*> peut également être spécifié.
- ✓ ***Opération de type notification (Notification):*** Le service envoie un message.

L'opération a donc un seul élément <*output*>.

La spécification WSDL définit également les éléments facultatifs suivants :

- **<*documentation*> :** L'élément « documentation » est utilisé pour fournir des informations lisibles par l'utilisateur (par exemple : description textuelle du service en

utilisant le langage naturel). Il peut être inclus à l'intérieur de n'importe quel autre élément WSDL.

- ***<import>*** **:** L'élément « import » est utilisé pour importer d'autres documents WSDL ou des schémas XML. Par exemple, deux documents WSDL peuvent importer les mêmes éléments de base et encore inclure leurs propres éléments pour assurer le même service sur deux adresses physiques différentes. Il est à noter que cette fonctionnalité n'est pas supportée par tous les outils WSDL pour l'instant.

Figure 1.10 : fichier WSDL. [25]

2.2.6.5. La couche de Découverte et de Publication (UDDI)

UDDI (Universal Description, Discovery and Integration) **définit** les mécanismes permettant de répertorier des Web Services. Ce standard régit donc l'information relative à la publication, la découverte et l'utilisation d'un Web Service. En d'autres mots, UDDI détermine comment nous devons organiser l'information concernant une entreprise et les Web Services qu'elle offre à la communauté afin de permettre à cette communauté d'y avoir accès. En fait, UDDI définit un registre des Web Services sous un format XML. Ce registre peut être public, privé ou partagé [26].

2.2.6.5.1. Publication de services Web avec UDDI

Les différents composants de la publication faite par UDDI sont des documents XML manipulant de l'information à propos du fournisseur (Business Entity), le service lui même (Business Service), les accès au service (Binding Template), le type de service (tModel) et des relations entre deux parties (Publisher Assertion) [26].

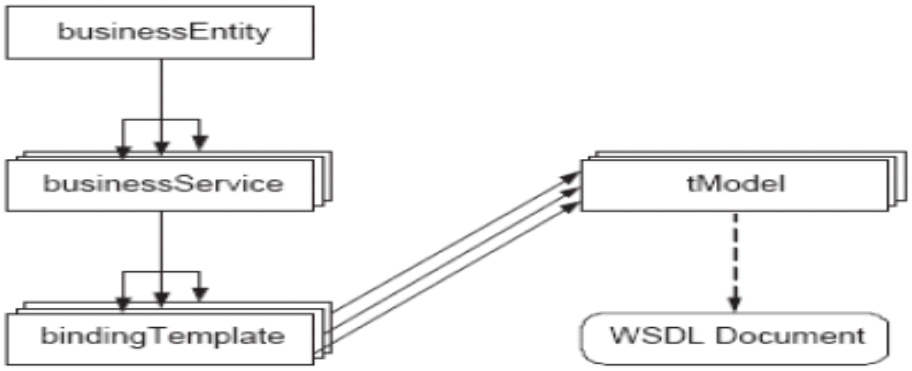


Figure 1.11 Entités composant un annuaire UDDI. [62]

- ✓ **Le fournisseur (Business Entity)** : Les informations concernant l'organisation hébergeant le service, le fournisseur et celles nécessaires à l'identification de l'entreprise sont répertoriées dans ce document XML. Ce document contient de l'information descriptive sur l'entreprise ou le fournisseur et sur les services proposés

- ✓ **Le service (Business Service) :** Ce composant représente les services proposés par l'organisation. La description des services contenue dans l'entité Business Service . Les informations à propos du nom du service et de son objectif sont représentées dans ce composant. Le fournisseur peut rassembler dans cette entité un ensemble de services répondant aux mêmes objectifs dans une même catégorie.

- ✓ **Les accès au service (Binding Template) :** Ce module décrit les points d'accès aux services Web (URL) et le moyen d'y accéder (les différents protocoles à utiliser) afin d'invoquer les services.

- ✓ **Le type de service (tModel) :** Le tModel permet d'associer un service à sa description WSDL. Le client potentiel peut ainsi avoir connaissance des conventions d'utilisation du service. La liaison entre les entités Binding Template et tModel est nécessaire pour l'invocation du service

2.2.6.5.2. Recherche de services Web avec UDDI

UDDI organise l'ensemble des informations qu'il contient en trois parties, spécifiées en XML. Chacune d'elles peut être utilisée pour faire une recherche via UDDI. Ces parties sont les suivantes [27] :

- ***Les pages blanches :*** qui décrivent les informations de contacts sur les entreprises.
- ***Les pages jaunes :*** qui décrivent des informations de classification de services.
- ***Les pages vertes :*** qui donnent des informations techniques des services.

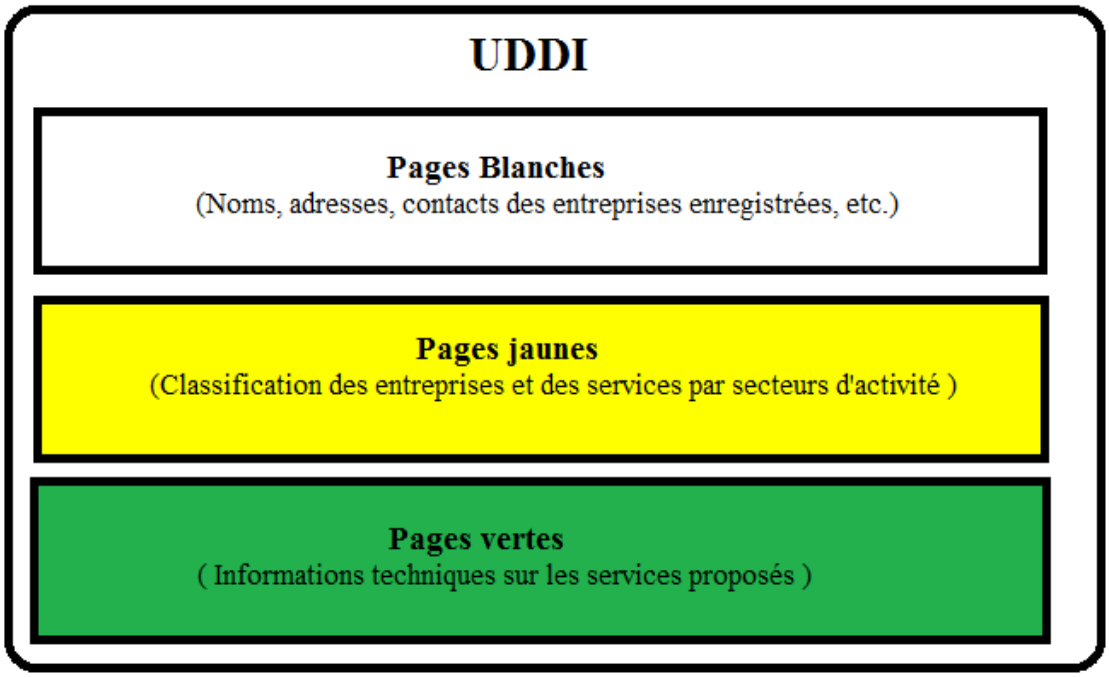


Figure 1.12 : Le contenu de l'annuaire UDDI [63]

2.2.7. Avantages et inconvénients des services web

Selon [28], la technologie des services web présente plusieurs avantages mais a aussi des inconvénients.

2.2.7.1. Avantages

La technologie des services web est populaire et couramment utilisée car elle offre des avantages intéressants pour les utilisateurs des systèmes distribués :

- Les services web réduisent le temps de mise en marché des services offerts par les diverses entreprises.

- Les services web permettent à des programmes écrits en des langages différents et sur des plateformes différentes de communiquer entre eux par le biais de certaines normes. En d'autres termes, les services web permettent une meilleure interopérabilité entre les logiciels.
- Les services web utilisent des normes et protocoles ouverts.
- Grâce au protocole HTTP, les services web peuvent fonctionner malgré les pare-feu sans pour autant nécessiter des changements sur les critères de filtrage.
- Les protocoles et les formats de données sont offerts, le plus possible, en format texte pour que la compréhension du fonctionnement des échanges soit plus intuitive.
- Grâce aux services web, les coûts sont réduits par l'automatisation interne et externe des processus commerciaux.

2.2.7.2. Inconvénients

La technologie des services web comporte plusieurs inconvénients dont :

- ✓ Problèmes de sécurité: Il est facile de contourner les mesures de sécurité mises en place par les pare-feu -l'utilisation du protocole HTTP (tel que mentionné ci-haut) n'a pas que des avantages -car les normes de sécurité des services web laissent encore à désirer. CORBA, par exemple, qui est une technologie plus mûre, est plus sécuritaire.
- ✓ Problèmes de performance: Les services web sont encore relativement faibles par rapport à d'autres approches de l'informatique répartie telles que CORBA ou RMI.
- ✓ Confiance: Les relations de confiance entre différentes composantes d'un service web sont difficiles à bâtir, puisque parfois ces mêmes composantes ne se connaissent même pas.
- ✓ Syntaxe et sémantique: On se concentre beaucoup sur comment invoquer des services (syntaxe) et pas assez sur ce que les services web offrent (sémantique).
- ✓ Fiabilité: Il est difficile de s'assurer de la fiabilité d'un service car on ne peut garantir que ses fournisseurs ainsi que les personnes qui l'invoquent travaillent d'une façon fiable.
- ✓ Disponibilité: Les services web peuvent bien satisfaire un ou plusieurs besoins du client. Seront-ils pour autant toujours disponibles et utilisables? Ça reste un défi pour les services web.

3. Composition de service web :

3.1. Définition :

Un des avantages de SOA est l'intégration de services ou de systèmes distribués pour l'approvisionnement de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour des applications, d'autres services ou plus communément pour des utilisateurs humains. En effet, si une application ou un client requièrent des fonctionnalités, et qu'aucun service n'est seul apte à les fournir, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client [29] [30]. C'est ce que l'on appelle la *composition de services* web.

La composition de services vise à faire Coopération, interagir et coordonner plusieurs services pour la réalisation d'un but [31].

Une composition de services Web est constituée de plusieurs services qui interagissent les uns avec les autres, afin d'offrir de nouvelles fonctionnalités qu'un seul service ne pourrait pas les offrir [32]. La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités. Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers.

La composition de services est à l'origine d'un nombre considérable de travaux, aussi bien dans la recherche académique que dans l'industrie. En dépit de tous les efforts déployés, la composition de services reste un problème très complexe. Cette complexité tient au fait que les solutions de composition de services doivent tenir compte du nombre croissant de services déployés sur le web, de leur mise à jour continuelle et de leur hétérogénéité [33]. En conséquence, elles nécessitent de traiter les problématiques de la coordination des services, leurs transactions, leur exécution et leur modèles de conversation (ou d'interaction) [34].

3.2. Cycle de vie d'une composition des services Web

Selon [35], le cycle de vie d'une composition de services Web reposant à partir de six activités :

1. **L'encapsulation de services natifs (*Wrapping services*) :** Cette première activité permet de s'assurer que tout service peut être appelé lors d'une composition, indépendamment de son modèle de données, de son format de message, et de son protocole d'interaction.

2. **L'établissement d'accord d'externalisation (*Setting outsourcing agreements*) :** Cette seconde activité consiste à négocier, établir, et appliquer des obligations contractuelles entre les services.

3. **L'assemblage de services composants (*Assembling composite services*) :** Cette activité permet de spécifier, à un haut niveau d'abstraction, l'ensemble des services à composer afin d'atteindre l'objectif attendu. Cet assemblage comporte une phase d'identification des services et de spécification de leurs interactions conformément aux descriptions et aux accords entre services.

4. **L'exécution de services composants (*Executing services*) :** Cette activité consiste en l'exécution des spécifications de la composition précédemment définies.

5. **Le contrôle de l'exécution de services composites (*Monitoring services*) :** La phase de contrôle permet de superviser l'exécution de la composition en vérifiant, par exemple, l'accès aux services, les changements de statut, les échanges de messages. Ce contrôle permet de détecter des violations de contrats, de mesurer les performances des services appelés et de prédire des exceptions.

6. **L'évolutivité des services (*Evolving services*) :** Cette dernière phase permet de faire évoluer la composition en modifiant les altérations de l'organisation de services, en utilisant de nouveaux services, ou en prenant en compte les retours de la phase de contrôle.

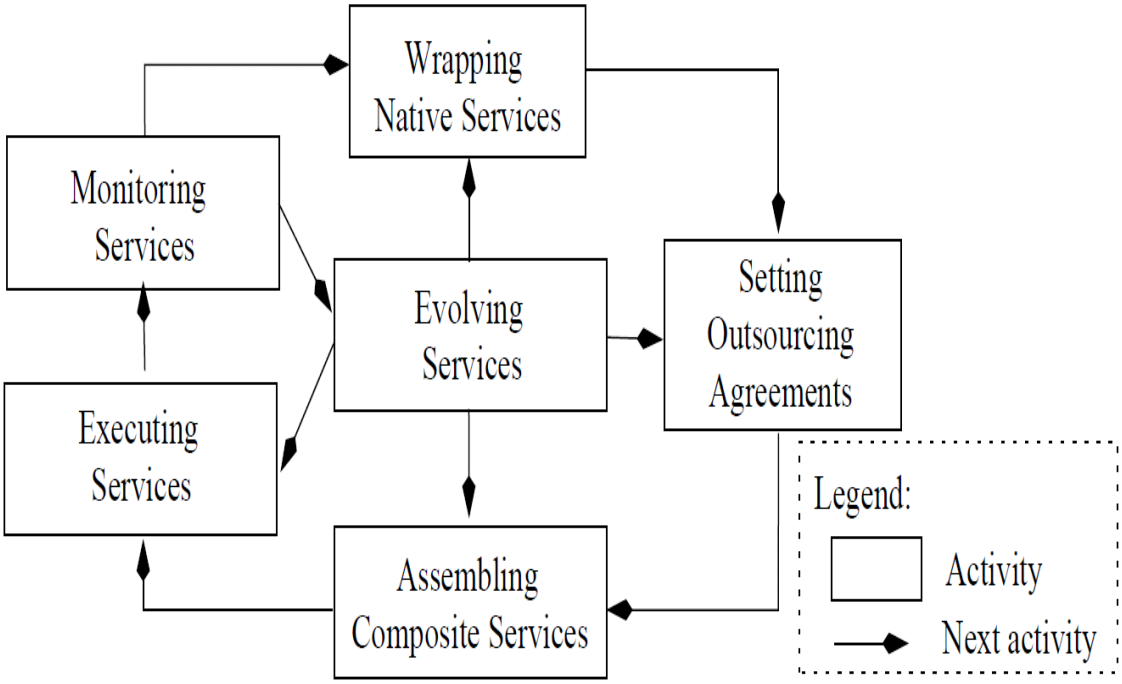


Figure 1.13 : Illustration du cycle de vie de d'une composition de services Web. [35]

Il existe deux domaines d'application de la composition de services Web (le e-business et le Web sémantique) [35]. Cependant, chacun d'eux utilise la composition à des fins qui lui sont propres :

- ✓ **L'e-business :** Les organisations utilisent des méthodes afin de représenter les flots de données inter et intra entreprises. Les concepteurs des applications d'entreprise (ou les experts métiers) définissent au préalable les processus. La définition du processus repose sur la spécification du besoin, des flots de contrôle entre les activités, et des contraintes. La composition de services Web permet d'attribuer à chaque activité du processus un service Web.

- ✓ **Le Web sémantique :** Les travaux dans le domaine du Web sémantique rendent le Web exploitable par les machines elles-mêmes, sans intervention humaine. Certains travaux de ce domaine étendent ce même objectif (l'exploitation par les machines elles-mêmes) aux services Web. On parle alors de services Web sémantiques. Dans le contexte de la composition de services Web, le Web sémantique intègre l'automatisation aux processus de découverte, d'invocation, et de surveillance des services Web [36].

3.3. Exemple de composition de service

❖ *Exemple 1:*

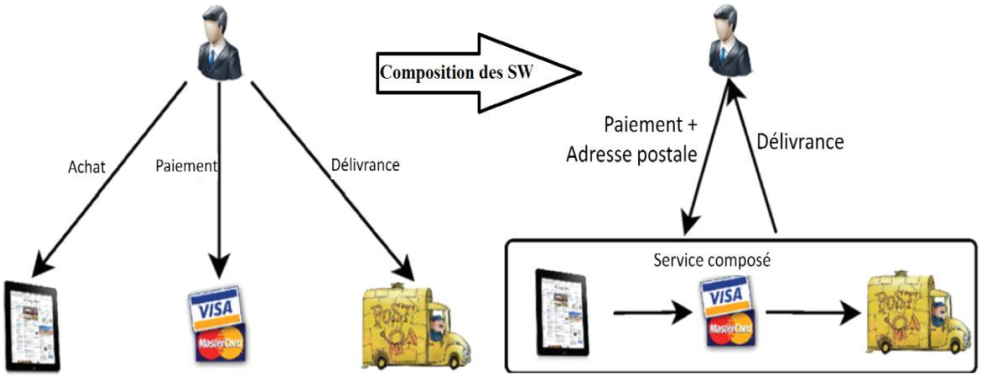


Figure 1.14 : Deux scenarios possibles pour un achat en ligne en utilisant les services Web. [36]

Dans le scénario illustré dans la Figure 1.14, un client a besoin de faire plusieurs appels à des services Web élémentaires pour compléter un achat en ligne. En outre, les appels doivent être faits dans un ordre spécifique dans lequel une ou plusieurs références de l'appel précédent sont utilisées dans l'invocation suivante. En plus d'être un processus lourd ce scénario augmente le risque d'erreurs. D'où l'appel à une composition des services Web est indispensable.

❖ *Exemple 2 :*

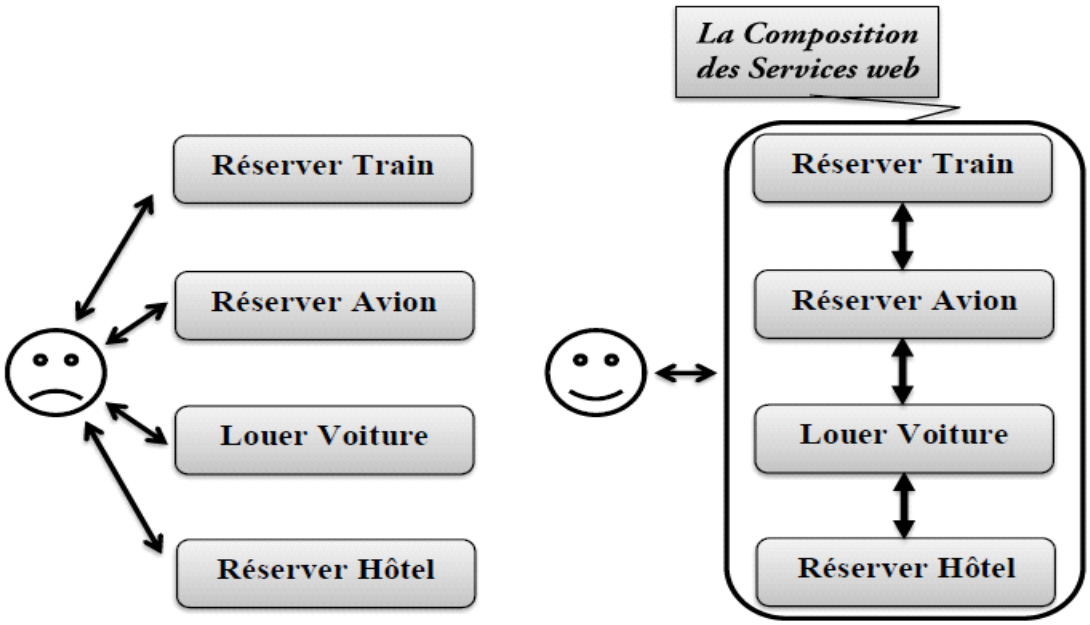


Figure 1.15 : Exemple de planification d'un voyage [63]

Le service de planification de voyage d'une agence de voyages est le résultat d'une composition de plusieurs services web atomiques (basiques) appartenant à des organisations différentes : réservation de vol, réservation d'hôtel et paiement en ligne…. etc.

3.4. Défis de composition des services Web

Le problème de la composition des services Web est en plein essort. Nous soulignons ici quelques sources de ses complexités [38] :

- ✓ la complexité liée à la conception de services Web : un services web n'est pas adaptable. Il est passif jusqu'à ce qu'il soit invoqué. Il a des connaissances de lui-même mais pas de ses applications ou de ses utilisateurs clients.
- ✓ le nombre des Web services disponibles est très grand, il est déjà au-delà des capacités humaines pour les analyser manuellement.
- ✓ puisque le nombre de Web services augmente jour âpres jour, il devient plus difficile de trouver artificiellement les services web qui peut effectuer la tâche à accomplir et encore plus difficile de composer un ensemble de services avec des approches classiques.
- ✓ les Web services peuvent être crées et réactualises a la volée. Par conséquent, le système de composition doit détecter la mise `à jour lors de l'exécution. Ainsi, le schéma de la composition devrait s'adapter en fonction des nouvelles informations.
- ✓ les Web services sont généralement établis par des organisations différentes qui utilisent différents modèles conceptuels pour la présentation des caractéristiques des services. Cela exige l'utilisation d'informations pertinentes pour faire correspondre (matching) et composer les services Web.

3.5. Orchestration et Chorégraphie

3.5.1. Orchestration

L'orchestration décrit, du point de vue d'un service, les interactions de celui-ci ainsi que les étapes internes (ex. transformations de données, invocations à des modules internes) entre ses interactions [39]. L'orchestration de services Web résulte un nouveau service Web dit service Web composé, qui peut être défini comme l'agrégation de plusieurs autres services Web atomiques ou composés. Ce service composé contrôle la collaboration entre les services Web engagés dans la composition, tel qu'un chef d'orchestre [40].

L'orchestration de services Web exige de définir l'enchaînement des services Web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce dernier est une entité logicielle qui joue le rôle d'intermédiaire entre les services, en les appelants suivant le script d'orchestration [41].

L'orchestration de services Web consiste en la programmation d'un moteur qui appelle un ensemble de services Web selon un processus prédéfini. Ce moteur définit le processus dans son ensemble et appelle les services Web (tant internes qu'externes à l'organisation) selon l'ordre des tâches d'exécution. La Figure 1.16 illustre l'exécution du moteur (lui-même un service Web – Service Web Moteur) permise par l'enchaînement de l'exécution de deux autres services Web (le Service Web 1 puis le Service Web 2). Cet enchaînement est possible via un opérateur d'ordonnancement (représenté par le losange dans la figure). L'exécution de la composition repose sur l'appel du Service Web 1, puis sur l'appel du Service Web 2, réalisés tous deux par le Service Web Moteur [42].

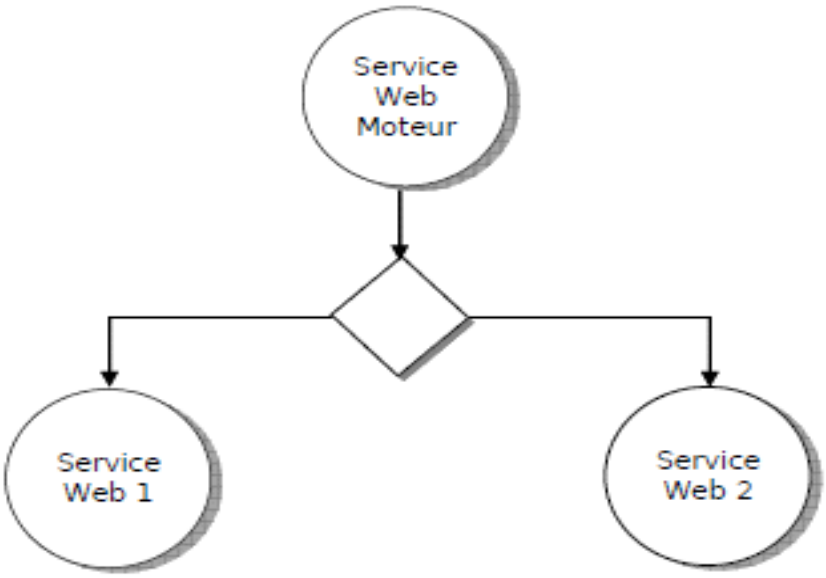


Figure 1.16 : Illustration de l'orchestration. [42]

La Figure 1.17 illustre l'orchestration. La requête du client (logiciel ou humain) est transmise au moteur d'exécution (Moteur). Ce dernier, d'après le processus préalablement défini, appelle les services Web (ici, SW1, SW2, SW3 et SW4) selon l'ordre d'exécution [41].

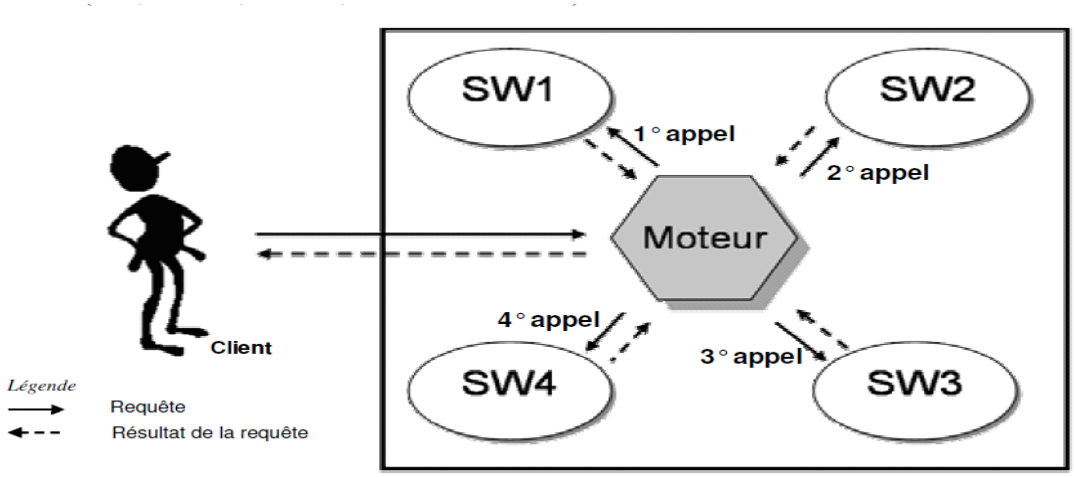


Figure 1.17 : Vue générale de l'orchestration. [41]

3.5.2. Chorégraphie

La chorégraphie décrit la collaboration entre une collection de services dont le but est d'atteindre un objectif donné. L'accomplissement de ce but commun se fait alors par des échanges ordonnés de messages [43]. La chorégraphie de services Web est une généralisation de l'orchestration qui consiste à concevoir une coordination décentralisée des services Web. Dans une chorégraphie, les interactions de type pair-à-pair (P2P) sont décrites dans un langage de description de chorégraphie (CDL). Les services suivent alors le scénario global de composition sans point de contrôle central [40].

La chorégraphie est aussi appelée composition dynamique. En effet, l'exécution n'est pas régie de manière statique comme dans une composition de type orchestration. Dans une chorégraphie à chaque pas de l'exécution, le service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance [41].

La description de chaque service Web intervenant dans la chorégraphie inclut la description de sa participation dans le processus. De ce fait, ces services peuvent collaborer à l'aide de messages échangés afin de savoir si tel ou tel service peut aider dans l'exécution de la requête. Chaque service Web peut communiquer avec un autre service Web par l'intermédiaire d'échange de messages. La Figure 1.18 représente un protocole d'initiation de collaboration entre deux services dans le cadre d'une chorégraphie. Dans cet exemple, le Service Web 1 demande l'exécution d'une méthode du Service Web 2 par l'intermédiaire d'un envoi de message (Requête de service). Cette requête est acceptée par le service Web 2.

Ce dernier envoi un message d'acceptation au Service Web 1 (Acceptation). Le Service Web 1 accepte le service proposé par le Service Web 2 en lui envoyant un message (Service admis) accordé (Acceptation) par ce second service. Une fois ces messages échangés le Service Web 1 peut invoquer les Service Web 2 dans le cadre de la composition [42].

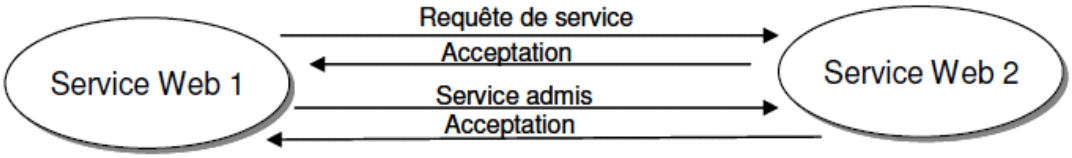


Figure 1.18 : L'illustration de la chorégraphie. [42]

La Figure 1.19 permet d'illustrer une vue générale d'une composition de services Web de type chorégraphie. Le client (logiciel ou humain) établit une requête qui est satisfaite par l'exécution automatique de quatre services Web (SW1, SW2, SW3 et SW4). La requête de l'utilisateur est transmise au premier service Web (SW1) qui est exécuté. Le SW1 découvre ensuite le service Web lui succédant. Le processus de découverte repose, selon les cas, soit sur une recherche dans un registre local ou public, soit sur une découverte globale sur le Web à l'aide d'ontologies. Une fois le service découvert, les deux services (SW1 et SW2) échangent des messages (comme illustré par la Figure 1.18) afin de vérifier si leur communication est viable dans le cadre de la requête. Si les échanges de messages sont concluants, le résultat de l'action du SW1 est transmis au SW2 qui l'utilise comme paramètre d'entrée. Le processus d'implémentation de la composition est identique pour chaque étape (SW2 et SW3). Le SW4 termine le processus et le résultat de son action est transmis au client. [42].

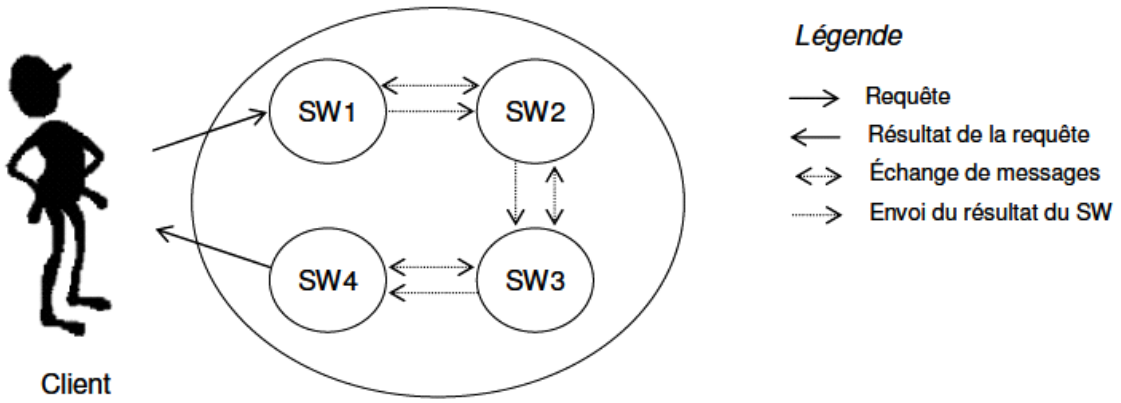


Figure 1.19 : vue générale de la chorégraphie**.** [42]

Le W3C compte depuis 2002 parmi ses groupes de travail, le groupe de travail sur la chorégraphie de services Web (Web Services Choreography Working Group). Pour ce dernier, la chorégraphie des services Web concerne les interactions observables des services avec leurs utilisateurs (appelés aussi clients) [44]. Ces utilisateurs, automatisés ou non, peuvent être d'autres services Web, des applications, ou des concepteurs d'applications. Cet ensemble spécifique d'interactions peut être comparé à une collaboration entre un ensemble de services Web et leurs clients. La description d'une chorégraphie est un contrat multi-parti qui décrit, à partir d'un point de vue global, le comportement observable externe entre plusieurs clients (généralement des services Web). Chaque comportement externe observable est défini comme la présence ou l'absence de messages échangés entre un service Web et ses clients.

L'orchestration et la chorégraphie sont toutes deux des manières de composition de service web agrégé. La différence principale entre ces deux manières, est que dans la chorégraphie, chaque service connaît les autres services qui interagissent, alors que dans le cas de l'orchestration, le service qui joue le rôle de chef d'orchestre est le seul qui connaît les services en interaction.

3.5.3. Inconvénient d'orchestration et chorégraphie

Elles présentent plusieurs inconvénients [45] :

- Ces approches exigent de l'utilisateur d'avoir des connaissances bas-niveau du système. Dans le cas de WS-BPEL (Web Services Business Process Execution Language), BPWS4J et WS-CDL (The Web Services Choreography Description Language) par exemple, l'utilisateur est censé construire un workflow au niveau XML. Ces systèmes ne s'adressent donc qu'à des usagers spécialistes en ingénierie de l'information.

- Les langages de chorégraphie permettent de décrire une collaboration entre services pour l'accomplissement d'un but. Toutefois, n'étant pas exécutables, ils ne suffisent pas seuls à l'implémentation d'une chorégraphie automatique de services.

- Dans ces travaux, les services à composer et le flot entre services sont préalablement définis. En conséquence, si l'un des services à composer n'est pas disponible, l'exécution échoue.

- Des inconsistances peuvent survenir à cause de l'approvisionnement de nouveaux services ou du remplacement d'anciens services par d'autres. Dans ces cas, il est inévitable de changer d'architecture logicielle et de la relier à d'autres services ou même, dans le pire des cas, changer la définition du processus et reconcevoir le système.

3.6. Les types de composition de services web

Après avoir présenté les deux techniques de coordination des services web (Orchestration et Chorégraphie), nous allons présenter dans cette partie les différents types de la composition qui peuvent être classifié selon deux axes [46] :

- **Premier axe** : En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition (Composition manuelle, semi-automatique ou automatique).

- **Deuxième axe** : Selon la nature du processus de la sélection des services web (sélection à priori ou non des services composants). Dans ce cas, nous distinguons deux catégories (Composition statique ou dynamique).

Dans ce qui suit, nous allons présenter ces différents types de composition.

3.6.1. En fonction du degré de participation de l'utilisateur

- **Composition manuelle** : Le seul avantage majeur de ce type de composition c'est qu'elle permet à l'utilisateur de générer la composition manuellement selon ses préférences sans l'aide des outils dédiés [47]. Mais, elle nécessite une grande connaissance au niveau de la programmation et elle requiert beaucoup de temps à l'implémentation. Par conséquent, elle devienne inefficace de nos jours qui se caractérisent par un développement dans le nombre des services Web.

- **Composition semi-automatique :** La composition semi-automatique permet à l'utilisateur de maintiendra un contrôle sur le processus de composition sans besoin d'avoir des connaissances de programmation. En d'autres termes, elle fait des suggestions pour aider l'utilisateur à construire sa composition en utilisant des outils de modélisation graphique [47]. Nous pouvons conclure que ce type de composition est un mécanisme d'assistance pour générer progressivement la composition.

- **Composition automatique :** La composition totalement automatisée est proposée pour remédier au défi de composition engendré par l'augmentation du nombre de services

Web [48]. Ce type de composition prend en charge tout le processus de composition [47]. Elle permet de découvrir et composer automatiquement les services Web afin de répondre à une requête sans aucune intervention de l'utilisateur [48].

Nous pouvons citer différentes sources de complexité :

- La difficulté de la composition en fonction de l'expressivité du modèle de Services et l'objectif de la composition.
- Le grand nombre de services web sur le Web.
- La diversité des modèles de conception de services en raison des besoins de modélisation et/ou de la vision des développeurs.

3.6.2. Composition statique et composition dynamique

- **Composition statique de services Web *(off-line)* :** dans ce type de composition la construction d'un modèle abstrait des tâches qui doit être réalisé au cours de l'exécution de la composition se fait à la conception et/ou la compilation avant que la planification de la composition commence. Ce modèle abstrait est rien qu'une représentation d'un ensemble des tâches et des dépendances entre eux, ou chaque tâche correspond à un service Web [49]. Les services Web participants à la composition sont choisis, liés ensemble, compilés puis déployés.
- **Composition dynamique de services Web *(on-line)* :** contrairement à la composition statique La sélection des services composants et la réalisation des liaisons sont retardées jusqu'au moment de l'exécution. Les étapes de planification et de construction sont réalisées au moment de l'exécution en fonction des fournisseurs de services Web disponibles. [50]

La technologie de la composition dynamique est généralement contestée par :

- ✓ Le grand nombre de services qui deviennent disponibles chaque jour.
- ✓ La nature volatile de services web (par exemple : ils peuvent disparaître, être modifiés ou être temporairement indisponibles).
- ✓ Le nombre sans cesse croissant de fournisseurs de services.

Dans le tableau ci-dessus Tab 1.1 on donne les principales différences entre la composition statique et dynamique de services Web.

Critère	statique	dynamique
Génération de modèle de processus	A la conception /compilation	A l'exécution
Découverte /sélection des services	A la conception/compilation	A l'exécution
Liaison des services	A la conception (i e : chaque instanciation du service composé sera des mêmes services composants)	A l'exécution
Conception de la composition	A la conception	A l'exécution
Personnalisation a l'exécution	impossible	possible
Possibilité d'extension l'exécution	impossible	possible
Nombre de services	Limité	Illimité
Cout	Constant	Varie selon les services sélectionnés
Fiabilité et tolérance aux fautes	Faible	Elevée (spécifiquement dans le cas où un service devient indisponible après un certain temps qui peut être substitué par l'invocation du service fonctionnellement équivalent)
Adaptabilité aux changements d'environnements	Faible	Elevée

Tab 1.1 : Comparaison entre la composition statique et dynamique. [51]

3.7. Modèles des services composites

La structure d'un service composite peut prendre plusieurs modèles [52] qui présentent les plans d'exécution des services composites. Les quatre modèles fondamentaux sont graphiquement représentés dans la Figure 1.20.

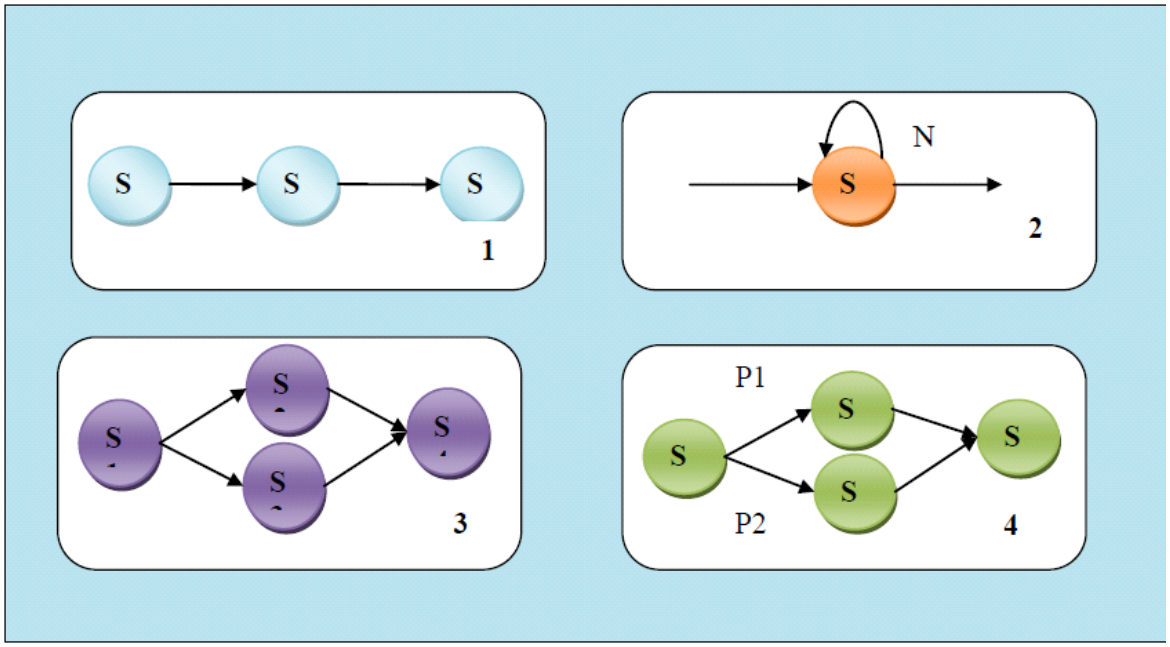


Figure 1.20 : les modèles de composition [52]

- **Le premier modèle :** c'est le modèle séquentiel qui permet d'exécuter une séquence de services.
- **Le deuxième modèle :** c'est le modèle Loop (boucle) qui permet d'exécuter un service n fois.
- **Le troisième modèle :** c'est le modèle parallèle qui permet d'exécuter deux services simultanément.
- **Le dernier modèle :** c'est le modèle conditionnel qui exécute une branche de services selon une probabilité.

Il existe plusieurs autres modèles de composition, tel que le modèle AND-JOIN, AND-SPLIT, etc.

3.8. Langages de composition de services Web

De nombreux industriels tels qu'IBM ou Microsoft et consortium tel que le W3C travaillent afin de mettre en œuvre un langage de composition de Services Web standard. Dans cette section, nous étudions les langages qui sont soit largement utilisés (BPEL4WS [53]), soit en cours de standardisation (WS-CDL [54] et OWL-S [55]). Les bases théoriques sur lesquelles le langage est construit et une illustration de la description d'une composition de services.

3.8.1. BPEL

BEA, IBM, SAP, Siebel Systems et Microsoft ont uni leurs efforts afin de produire un langage de composition de services Web, conçu pour supporter les processus métier à travers les services Web. Ce langage, BPEL4WS (Business Process Execution Language for Web Services), est issu de la fusion de deux langages : WSFL – Web Service Flow Language d'IBM et XLANG de Microsoft. BPEL4WS combine les caractéristiques d'un langage de processus structuré par bloc (XLANG) avec ceux d'un langage de processus basé sur les processus métier (WSFL). BPEL ou BPEL4WS est basé sur XML et sur les workflows. Ce langage distingue les processus abstraits des processus exécutables. [53]

- **Le processus abstrait :** Ce processus spécifie les messages échangés entre les différents parties (services, Web, composants) sans indiquer le comportement de chacune d'elles. parle de Business Protocol, c'est-à-dire la spécification du comportement des partenaires par rapport aux messages échangés, sans rendre public le comportement interne. Les services Web communiquent alors à l'aide d'échanges de messages.

- **Le processus exécutable :** Ce processus permet de spécifier l'ordre d'exécution des activités, le partenaire concerné, les messages échangés entre ces partenaires, et les mécanismes des erreurs et des exceptions. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les partenaires.

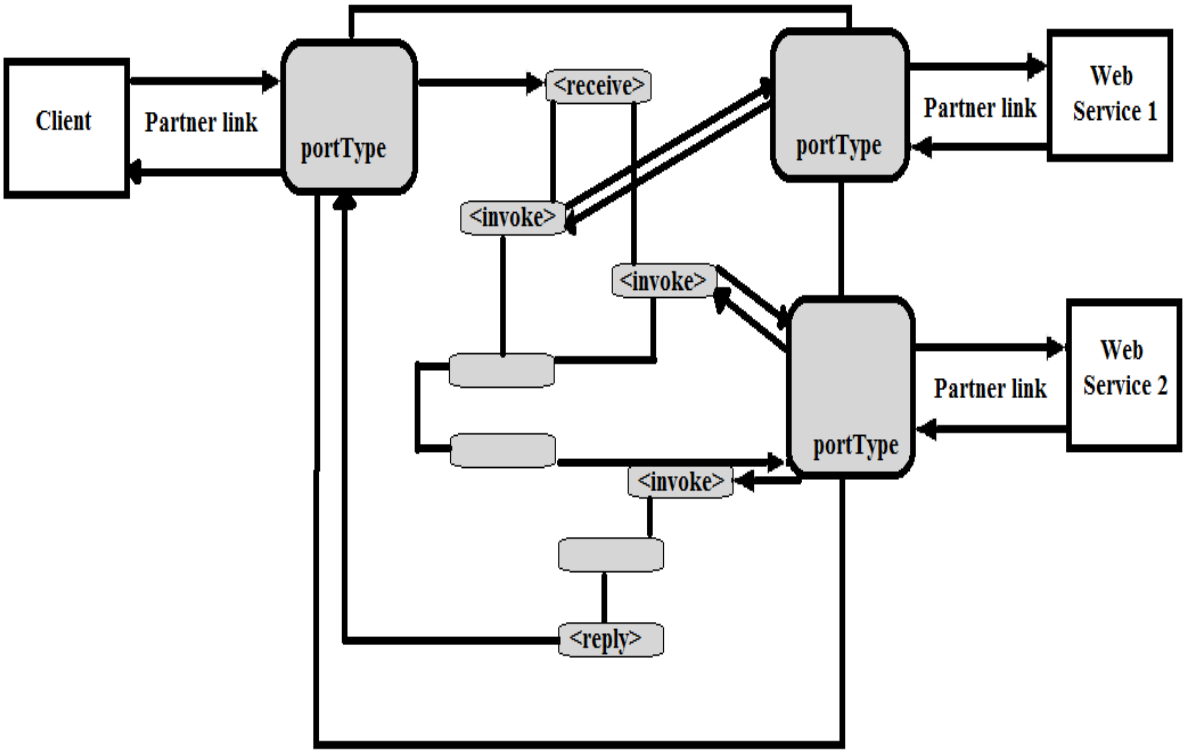


Figure 1.21 : un exemple de processus BPEL [64]

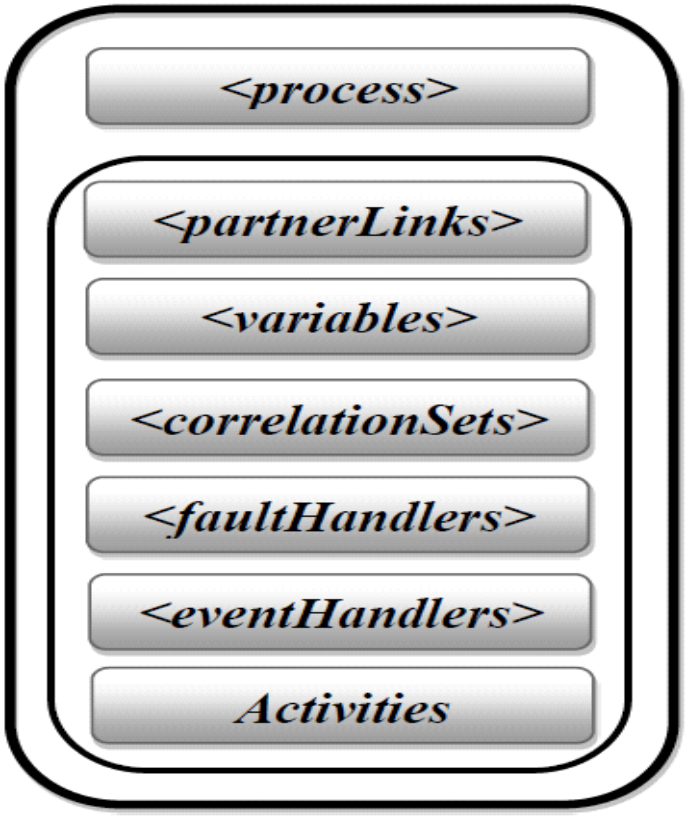


Figure 1.22 : Structure d'un fichier BPEL [63]

Comme montré dans la figure 1.23, la spécification BPEL offre plusieurs éléments décrits comme suit :

- **<process> :** L'élément « *process* » représente la racine du fichier BPEL. Il contient l'attribut « *name* » qui permet de définir le nom du processus.

- **<partnerLinks> :** Il permet de spécifier les différents partenaires participant dans la composition. Il est composé d'un ou plusieurs éléments <*PartnerLink*>. Ce dernier contient les attributs suivants :

  - ✓ **name :** Le nom donné au *partnerLink*.
  - ✓ **myRole :** Spécifie le rôle du processus BPEL.
  - ✓ **partnerRole :** Spécifie le rôle du partenaire ou du client.
  - ✓ **partnerLinkType :** Représente le type de *partnerLink* défini dans la description WSDL.

Si l'attribut « *myRole* » est uniquement utilisé (sans « *partnerRole* »), cela signifie que seules les interactions vers le processus sont autorisées. Dans le cas opposé (si l'attribut «*partnerRole* » est uniquement utilisé sans « *myRole* »), seules les interactions vers les partenaires et les clients sont autorisées. Il est à noter que les deux attributs peuvent être utilisés en même temps.

- **<variables> :** Il permet de définir les différentes variables utilisées dans le processus BPEL. Ces dernières servent à stocker des données ou des messages échangés afin de les réutiliser ultérieurement.

- **<correlationSets> :** Une orchestration peut être exécutée plusieurs fois. Chaque exécution est nommée instance. Pour acheminer les données à une instance particulière, nous utilisons l'élément <*correlationSets*>.

- **<faultHandlers> *:*** Il permet de gérer les exceptions au niveau du <*catch*>.

- **<eventHandlers> :** Il représente les événements pouvant survenir au cours de l'exécution. Il permet aussi d'associer un traitement à chacun de ces évènements.
- **Les activités:**
  - ✓ <***receive***> *:* C'est une attente bloquante d'un message entrant.
  - ✓ <***reply***> *:* Elle retourne un message suite à une réception d'un autre message (à l'aide de la primitive <*receive*>).

La combinaison *receive-reply* utilise une opération *request-response* d'un portType du processus.

- ✓ <**invoke**> *:* Elle permet l'appel d'un partenaire (service) en se basant sur une opération de type *one way* ou *request-response*.
- ✓ <**Assign**> *:* Cette activité permet la mise à jour des variables.
- ✓ <**Throw**> *:* Elle permet d'indiquer les erreurs et les exceptions survenues lors de l'exécution du processus et de les envoyer au *catch* de *fault handler*.
- ✓ <**Wait**> *:* Elle permet de bloquer l'exécution du processus pour une période donnée.
- ✓ <***Empty***> *:* C'est l'opération rien-faire, elle est utile pour la synchronisation.

Ces activités de base peuvent être combinées pour définir un algorithme complexe spécifiant les étapes par lesquelles passe le processus en utilisant les activités structurées telles que :

- ✓ <**Sequence**> *:* Elle définit une suite d'activités exécutées par ordre d'apparition dans la séquence.
- ✓ <**flow**> *:* Elle définit des activités exécutées en parallèle.
- ✓ <**switch**> *:* Elle définit un branchement conditionnel.
- ✓ <**if**> *:* Elle représente un autre cas de branchement conditionnel.
- ✓ <**while**>*:* Elle modélise une boucle conditionnelle.
- ✓ <**Pick**> *:* Elle bloque une activité jusqu'à la réception d'un message ou l'expiration d'une période de temps.

De façon générale, un processus BPEL commence par une activité <*receive*> et se termine par une activité <*reply*>. Il est interprété par un moteur d'orchestration.

3.8.2. WSCI

WSCI (Web service Chorégraphie Interface) [57] est un langage de haut niveau basé sur UML qui décrit la session de la couche métier. C'est un langage reposant sur l'XML aussi. Il propose de se focaliser sur la représentation des WSs en tant qu'interfaces décrivant le flux des messages échangés (la chorégraphie des messages) illustré dans la figure1. 25.

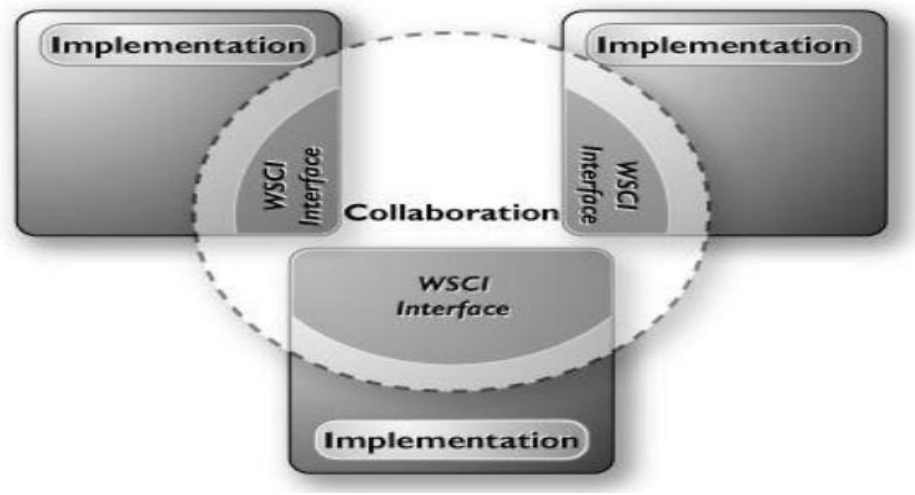


Figure 1.23 : Architecture de WSCI défini par W3C

Il propose ainsi de décrire le comportement externe observable du service. Pour ce faire, WSCI propose d'exprimer les dépendances logiques et temporelles entre les messages échangés à l'aide des contrôles des séquences, la corrélation, la gestion des fautes et les transactions. On remarque que le WSDL et ses définitions abstraites sont réutilisées afin de pouvoir également décrire par la suite les modalités de concrétisation des éléments manipulées pour modéliser un service. Cependant, il n'offre pas la possibilité de la description d'un WS considéré isolément.

3.8.3. WS-CDL

WS-CDL (Web Service Choreography Description Language) est un langage issu des efforts de standardisation du groupe de travail du W3C portant sur la chorégraphie de services Web (Web Services Choreography Working Group). L'objectif de ce langage est de décrire les relations entre les services Web lors d'une composition de type chorégraphie. [54]

WS-CDL est un langage de composition de type chorégraphie. Les interactions entre les services reposent sur les fondements du π-calcul [58].

3.8.4. BPML :

BPML, Business Process Modeling Language est un langage de la modélisation des processus métiers. Il permet de définir un modèle abstrait d'interaction entre les collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires. Les processus métiers sont représentés par un flux de données, un flux d'événements sur lesquels on peut influer en définissant des règles métier, des règles de sécurité, des règles de transaction. On peut ensuite lancer l'exécution du modèle et vérifier le fonctionnement théorique des processus différents. [59]

3.8.5. WSCL :

WSCL, Web Service Conversation Language propose de décrire à l'aide de document XML et les services Web en mettant l'accent sur les conversations de ceux-ci. En outre, les messages à échanger sont pris en compte. WSCL a été pensé pour s'employer conjointement avec WSDL. Les définitions de WSDL peuvent être manipulées par WSCL pour décrire les opérations possibles ainsi que leur chorégraphie. En retour, le WSDL fournit les concrétisations vers des définitions de messages et des détails techniques pour les éléments manipulés par WSDL. [60]

3.8.6. OWL-S :

OWL-S : (*Ontology Web Language for Services*) est un langage de mise en œuvre de services Web sémantiques qui permet la description, la découverte, l'invocation et la composition de type chorégraphie des services Web. [55]

OWL-S décrit les interactions entre les services Web sémantiques dans une composition de type chorégraphie à l'aide d'ontologies et de logiques de description.

Ses objectifs sont **:**

- ✓ la description de services Web sémantiques.
- ✓ la découverte automatique de ces services
- ✓ l'invocation automatique de ces services,
- ✓ la composition automatique de services (description et invocation) et la surveillance de l'exécution de la composition.

4. Conclusion :

La technologie des services web a pour objectif d'uniformiser la présentation des services offerts par une entreprise et d'en rendre l'accès transparent pour tout type de plateforme, au travers d'un certain nombre de standards d'interopérabilité. Dans ce chapitre, nous avons présenté de manière générale quelques concepts liés à la technologie des services web. Nous avons d'abord présenté les définitions de base de ce concept. Nous avons également décrit l'architecture fondamentale des services web ainsi que les langages et les protocoles de base permettant leur déploiement. Après, nous avons abordé le sujet de la composition de services web. Dans le chapitre suivant nous allons présenter les différents concepts de modélisation formelle des services composites.

Chapitre 2

« La modélisation formelle des services composite »

1. Introduction

La composition de services Web est la plus importante fonctionnalité assurée par une architecture SOA. Celle-ci offre un environnement homogène pour la composition dans la mesure où toutes les parties de la composition sont des services décrits de la même façon et communiquant par les mêmes standards d'échange de messages. Nous avons vue dans le chapitre précédent que la composition permet de combiner des services pour former un nouveau service dit composé ou composite.

L'exécution d'un service composite implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalités, afin de vérifier le bon fonctionnement et exécution de ce service composite les langages de composition comme BPEL doivent être munis avec des formalismes permettant de vérifier formellement ces services et c'est ce qui est absent dans ces langages. Pour surmonter ce manque il faut modéliser ces services composites en utilisant des modèles et techniques permettant la vérification formelles de ces services.

Ce chapitre fournit une présentation des différents formalismes utilisés pour la modélisation des services web composites, avec une synthèse permettant de justifier notre choix des RdPC que nous allons utiliser dans notre travail pour modéliser ces services composites.

2. La modélisation

2.1. Pourquoi modéliser ?

Plusieurs langages de composition de services web ont été proposés au cours de ces dernières années. Il s'agit malheureusement de langages textuels exécutables conçus pour satisfaire à la phase d'implémentation d'un service web composé et qui négligent de fait l'étape de spécification qui est importante, car elle facilite la compréhension globale du système ainsi que la tâche de développement. De plus, l'analyse formelle des langages proposés n'est pas possible à cause de leur manque de formalisme. Donc, il est indispensable d'utiliser des techniques et des modèles permettant la spécification formelle de la composition de services web, dans le but de s'assurer du bon fonctionnement des services avant de les implémenter.

De plus le processus de modélisation vise à mieux cerner les limites du système réalisé où les modèles sont raffinés de plus en plus pour aboutir au code sachant dans notre cas le

système est le service composite. Les modèles garantissent aussi une meilleure indépendance du système par rapport aux fonctions demandées et une meilleure capacité d'adaptation et d'évolution du modèle lorsque des fonctionnalités sont modifiées ou ajoutées.

2.2. Classification des Modélisation

La composition de services permet de définir et faciliter le processus d'intégration et de collaboration entre plusieurs services intra-entreprise ou inter entreprise. Cette technologie émergente a eu beaucoup d'intérêt autant dans le domaine de la recherche que dans le domaine de l'industrie. Comme cela a été présenté précédemment, de nombreux langages ont vu le jour au cours de ces dernières années tels que XLANG, WSFL, BPEL ou encore WSCI. Ces approches reposent ainsi sur des concepts de programmation et négligent de ce fait l'étape de spécification qui doit prendre place au début du cycle de développement. De plus, les langages proposés manquent de formalisme permettant la vérification et la validation de la composition de services. Il n'est donc pas possible d'appliquer directement les méthodes mathématiques sur ces langages, rendant ainsi la vérification difficile. En effet, les processus de validation et de vérification visent à assurer la conformité des systèmes aux exigences requises et le respect des caractéristiques de conception attendues. Ces processus apportent tout leur intérêt pour conclure sur la bonne description du modèle, et pour s'assurer du bon fonctionnement du système étudié avant sa réalisation et son implémentation. Par conséquent, il est indispensable de prendre en compte cette étape en se basant sur d'autres approches et modèles pour la spécification formelle de la composition de services. [88]

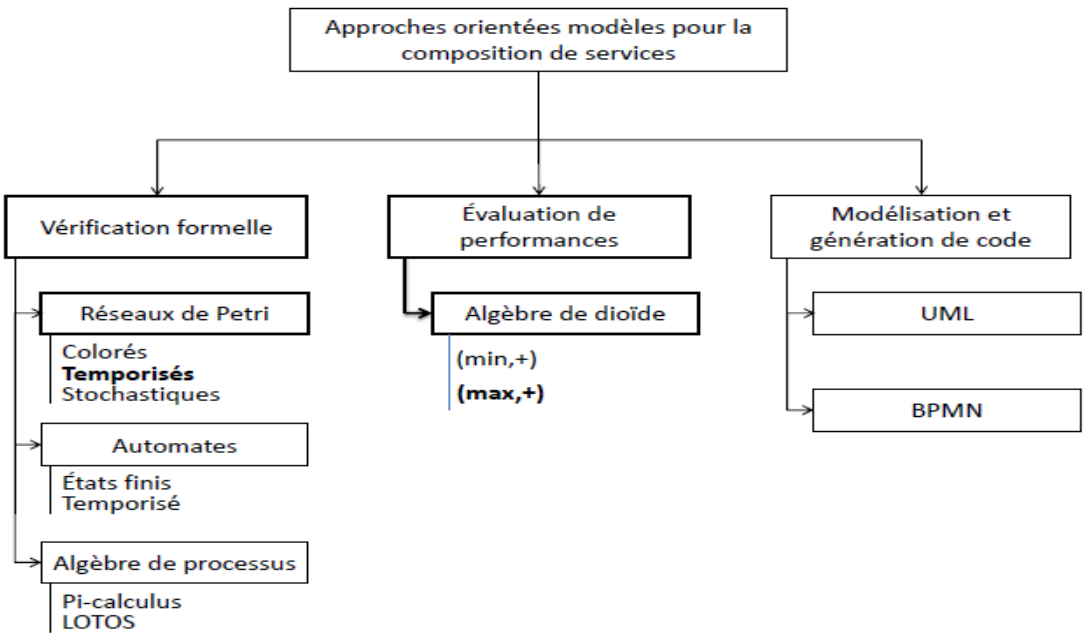


Figure 2.1 : Une classification des approches orientées modèles pour la composition de service [88]

Les approches orientées modèles pour la composition de services peuvent être classées en trois types selon leurs fonctions. La Figure 2.1 présente la classification de ces approches qui peuvent être classés en trois grandes familles, les approches orientées vérification formelle, les approches orientées modélisation et génération de code, et les approches orientées évaluation de performances.

2.3. Outils et méthodologies pour la modélisation

Dans cette section, nous allons d'abord discuter brièvement d'autres outils et méthodologies, ainsi que de la façon dont ils peuvent être utilisés pour modéliser la composition des services Web. En fournissant leurs comparaisons, nous discuterons de notre sélection sur la technologie des réseaux de Pétri avec des raisons.

Actuellement, il existe généralement trois techniques alternatives pour vérifier formellement la composition des services Web : (1) les réseaux de Pétri, (2) les algèbres de processus et (3) les automates à états finis, Il y a aussi d'autres techniques telles que : (4) UML, (5) BPMN (6) Graphes.

Pour illustrer l'utilisation de chaque approche de modélisation, nous allons considérer l'exemple de composition présenté dans la figure 2.2., nous donnerons la modélisation de cet exemple dans chacun des langages considéré afin de visualiser leur utilisation pour la composition de services. Dans ce scénario, le service S1 est invoqué en premier. Un choix exclusif (XOR) est ensuite réalisé afin de déterminer la branche d'exécution à emprunter. Le système invoquera alors soit le service S2 seul, soit les services S3 et S4 en parallèle. Enfin, le système fera appel au service S5 avant de mettre fin à son exécution. Cela signifie que deux chemins d'exécution sont possibles, en fonction du résultat du choix exclusif [89] :

- ✓ **Premier chemin : S1 → S2 → S5**
- ✓ **Second chemin : S1 → (S3 || S4) → S5**

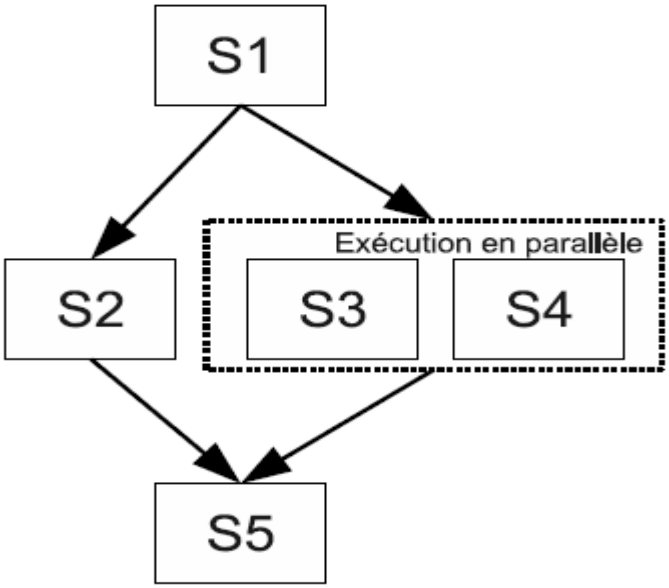


Figure 2.2 : Exemple de scenario de composition [89].

2.3.1. Réseaux de Pétri

Les réseaux de Pétri constituent une approche bien établie pour la modélisation de processus. Un réseau de Pétri est un graphe dirigé, connecté et biparti, c'est à dire composé de deux types de nœuds. Les nœuds représentent soit des places, soit des transitions. Les places peuvent éventuellement contenir des jetons, qui correspondent généralement aux ressources disponibles. Lorsqu'un jeton est présent dans chacune des places en entrée d'une transition, la transition est dite activée et peut alors s'exécuter. L'exécution d'une transition consomme un jeton dans chaque place en entrée et place un jeton dans chaque place en sortie. [90].

Il est représenté par un graphe biparti orienté reliant des places (représentant l'aspect local des états globaux) et des transitions (représentant des actions). Les réseaux de Pétri ont été largement utilisés pour modéliser les processus BPEL [91, 92]. Ainsi, chaque processus BPEL est transformé en un réseau de Pétri dont la sémantique a été définie formellement. Les outils et les techniques développés pour les réseaux de Pétri peuvent être utilisés et exploités dans le contexte de la composition de services Web. Une algèbre de services Web a été proposée dans [93] afin de définir des opérateurs de composition de services Web. . ont utilisé les réseaux de Pétri comme sémantique formelle de cette algèbre.

Ensuite, pour composer des services web, il suffit d'appliquer des opérateurs de composition aux réseaux de Pétri représentant les dits services web (chaque service web est représenté par un réseau de Pétri). Un réseau de Pétri avec une place d'entrée (input place)

pour recevoir des informations et une place de sortie (output place) pour émettre des informations, facilite la définition des opérateurs de composition, ainsi que l'analyse et la vérification de certaines propriétés [90].

Il est possible de modéliser la composition de services à l'aide des réseaux de Pétri en assignant une transition à chaque appel de service et une place à chaque état entre les appels. Cette approche a été proposée dans [94] c'est celle qui a été choisie pour la modélisation du scénario dans la figure 2.3. Chaque service composé est ainsi modélisé par un unique réseau de Pétri contenant une place d'entrée sur laquelle aucun arc ne pointe ainsi qu'une place de sortie d'où aucun arc ne part. Ces places d'entrée et de sortie permettent de modéliser respectivement l'absorption et l'émission d'informations par le service composé

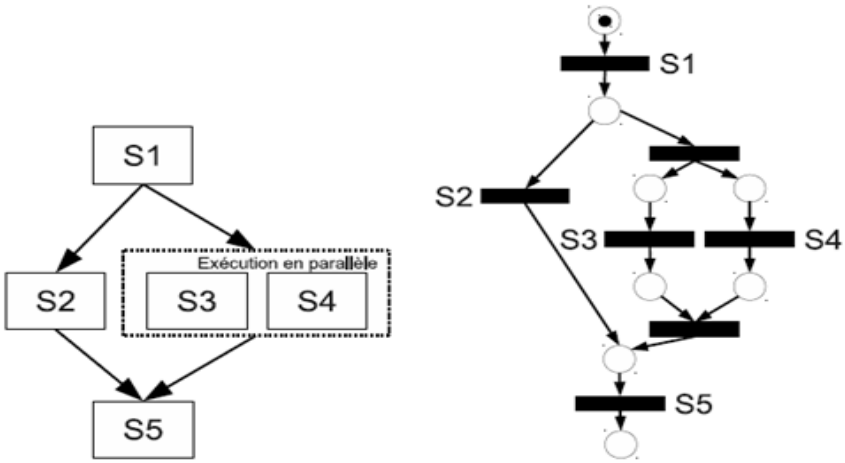


Figure 2.3 : Exemple de modélisation par réseau de Pétri d'un service composé [89].

2.3.2. les algèbres de processus

Les algèbres de processus sont des familles de langages formels permettant de modéliser les systèmes logiciels, en particulier les systèmes distribués et concurrentiels. Elles constituent un outil pour une description du plus haut niveau d'interaction, de communication et de synchronisation entre les processus. Elles définissent des règles algébriques permettant d'une part, l'analyse et la manipulation des descriptions des processus, et d'autre part, le raisonnement formel des équivalences entre les processus. Un tel raisonnement est très utile pour déterminer par exemple si un service peut remplacer un autre dans la composition, ou de vérifier simplement la redondance des services. Le fondement des algèbres des processus est basé sur des systèmes de transition étiquetés, à savoir les automates. Algèbres sont des

formalismes bien étudiés permettant la vérification automatique de certaines propriétés des comportements des systèmes [95].

Pour illustrer l'utilisation des algèbres de processus pour la spécification de services composés, nous fournissons un exemple basé sur π-calcul dans la figure 2.4. Cette représentation adopte l'approche événement-condition-action (ECA), Chaque activité du workflow est ainsi représentée par un processus indépendant. Les processus utilisent des événements, sous la forme d'échange de messages, pour coordonner le comportement du workflow. L'échange de messages s'effectue au niveau de canaux de communication dont le nom a été précisé sur le modèle dans la partie gauche de la figure, afin de faciliter la compréhension. En π-calcul, le parallélisme est exprimé par | et le choix exclusif par +. Nous définissons i comme la tâche réalisée en interne par le service i. c .X indique l'émission d'un message sur le canal "c" (ex: s1_to_s2 dans la figure) avant de continuer en tant que processus X. c. X (sans ligne supérieure) indique au contraire l'attente en réception d'un message sur le canal "c". Le processus "0" est un processus inerte, indiquant la fin de l'exécution du service. [95].

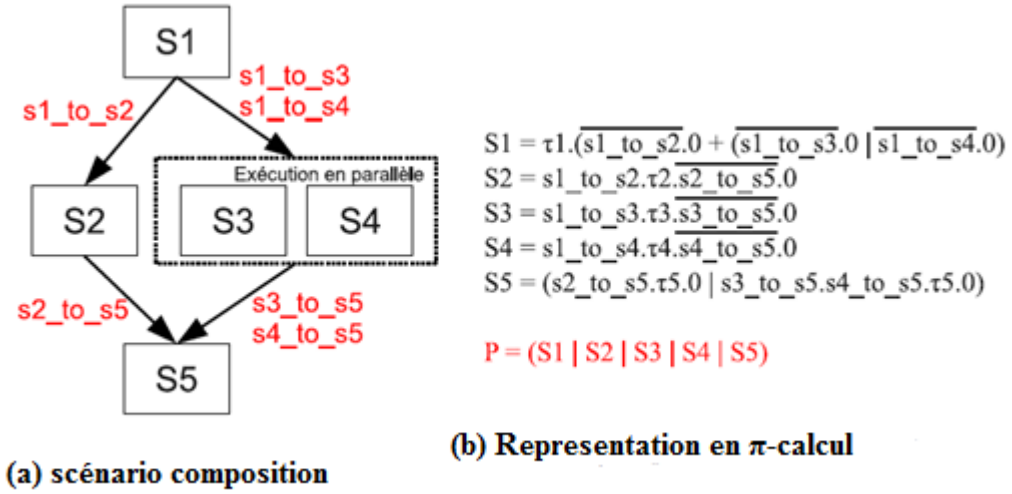


Figure 2.4 : Exemple de représentation en π-calcul d'un service composé [89].

2.3.3. les automates à états finis :

Une machine à états finis ou automate fini est un modèle bien établi de comportements composés d'états, de transitions et d'actions. Un état stocke des informations; une transition indique un changement d'état et est protégée par une condition qui doit être remplie; une action est une description d'une activité à effectuer à un moment donné. Généralement, quatre

types d'actions peuvent être identifiés: action d'entrée, action de sortie, action d'entrée et action de transition. [96].

Les machines à états finis peuvent être adoptées pour représenter les aspects d'un processus de composition global. En détail, la spécification de composition de services Web peut être décrite en utilisant une logique temporelle; le modèle FSM peut ensuite être traversé et vérifié pour vérifier si la spécification de flux de travail est valide. Spécialement, cette approche peut être utilisée pour vérifier la cohérence des données, et la satisfaction de l'entreprise. Les efforts de recherche actuels dans ce sens peuvent être classés en deux groupes: la spécification de la conversation et la composition automatique des services. La recherche sur le premier se concentre sur l'utilisation de machines Mealy pour modéliser les messages asynchrones entre les composants du service, vérifiant ainsi la réalisabilité d'une spécification de composition de services. La recherche sur le dernier se concentre sur la modélisation des comportements de service en tant qu'arbre d'exécution, puis de le traduire en un FSM. Le FSM généré peut ensuite être vérifié pour vérifier si une composition possible existe, c'est-à-dire s'il peut être terminé en nombre fini d'étapes.

Un exemple d'une telle modélisation est fourni dans la figure 2.5. Plus précisément, chaque branche parallèle d'exécution est modélisée par un automate indépendant et des événements de synchronisation (syn_start et syn_end dans la figure) sont utilisés pour les connecter. Ceci nécessite malheureusement la duplication de l'état d'entrée et de celui de sortie dans chaque branche parallèle. [97]

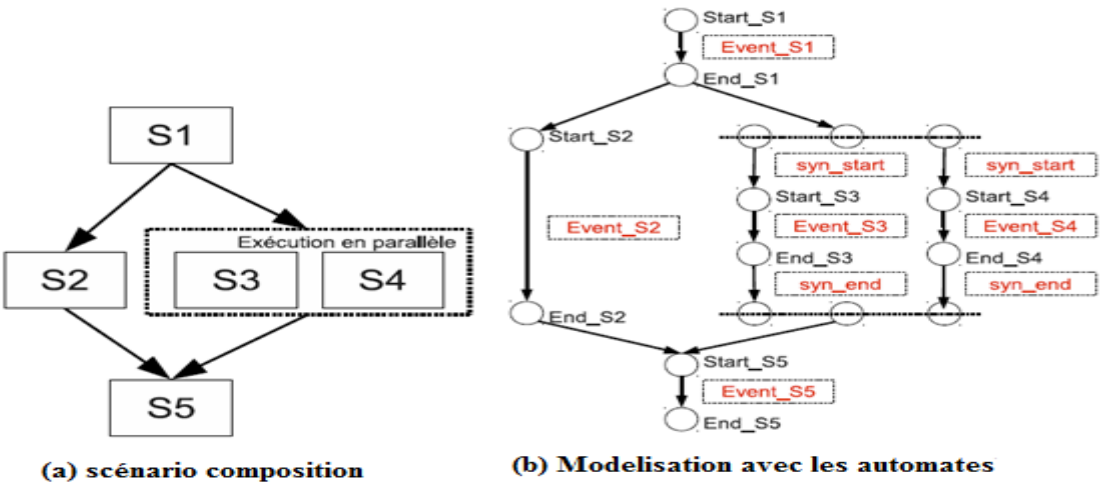


Figure 2.5 : Exemple de modélisation de service composé avec les automates [89].

2.3.4. Langage de modélisation unifié (UML) :

UML est un langage standardisé par l'OMG permettant de modéliser un système selon différents points de vue, statique et dynamique. La structure statique permet de modéliser un système en utilisant des objets, attributs, opérations et relations. La structure dynamique permet de modéliser le comportement dynamique d'un système en montrant les interactions entre les objets ou les changements d'états au sein d'un objet. Les diagrammes d'états et d'activités font partie de cette catégorie. [98].

UML, en particulier le diagramme d'activités, a été adapté pour la modélisation de la composition de services. Par exemple, dans [99], un profil UML a été introduit pour la modélisation de processus métier à l'aide des diagrammes d'activités. Un nouveau langage basé sur le diagramme d'activité UML 2.0 pour la modélisation de la composition de services a été présenté dans [100]. En utilisant ce langage, le code BPEL peut être généré à partir des modèles de la composition. Dans [101], UML-S (UML pour les Services) a été proposée pour modéliser la composition de services. Les auteurs ont également introduit des règles de transformation des diagrammes UML-S en BPEL.

L'exécution d'un service est généralement représentée par une action, c'est à dire une étape de l'activité. La transition vers cette activité modélise alors l'appel au service et inversement celle en sortie indique la fin de l'exécution du service. Ce choix de modélisation a été utilisé pour la représentation de l'exemple de composition dans la figure 2.6.

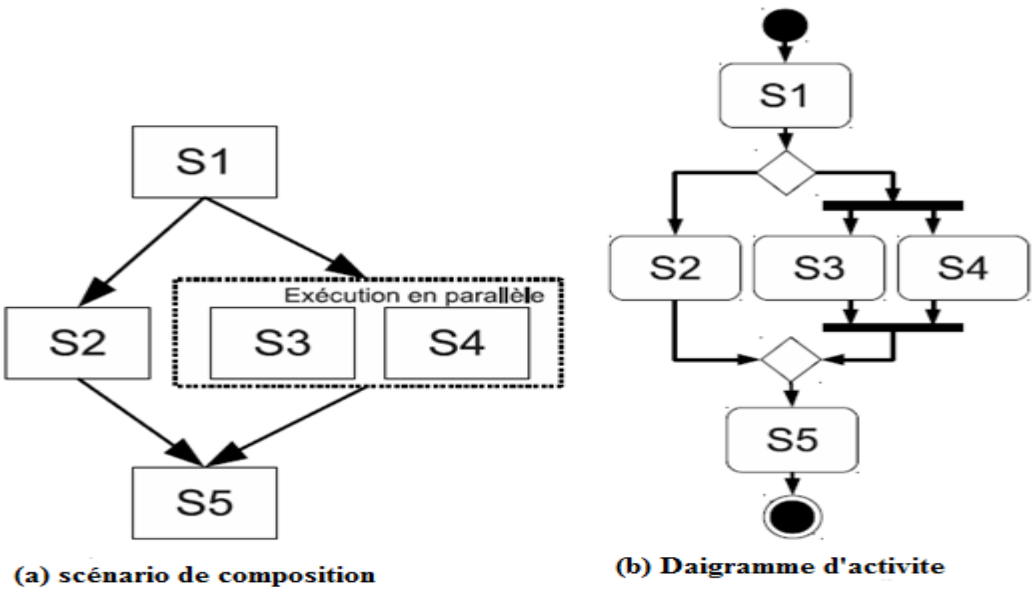


Figure 2.6 : Exemple de modélisation UML pour la composition de service [89].

2.3.5. BPMN :

La Business Process Management Initiative (BPMI) a développé une notation standard nommée BPMN pour la modélisation de processus métiers [102]. BPMN est désormais maintenu par l'OMG depuis son regroupement avec BPMI en 2005.

BPMN est une représentation graphique pour le BPM avec une accentuation sur les structures de contrôle. Elle définit un diagramme de processus métier (BPD) qui peut être considéré comme un organigramme de programmation intégrant des structures de contrôle adaptées au BPM telles que le choix exclusif (XOR split), la jonction simple (XOR-join), le branchement multiple (AND-split) ou la synchronisation (AND-join). [89]

Un exemple de modélisation pour la composition de services en BPMN est fourni dans la figure 2.7. Le modèle représente ainsi les interactions entre les services sous forme d'un processus métier en utilisant les éléments BPMN.

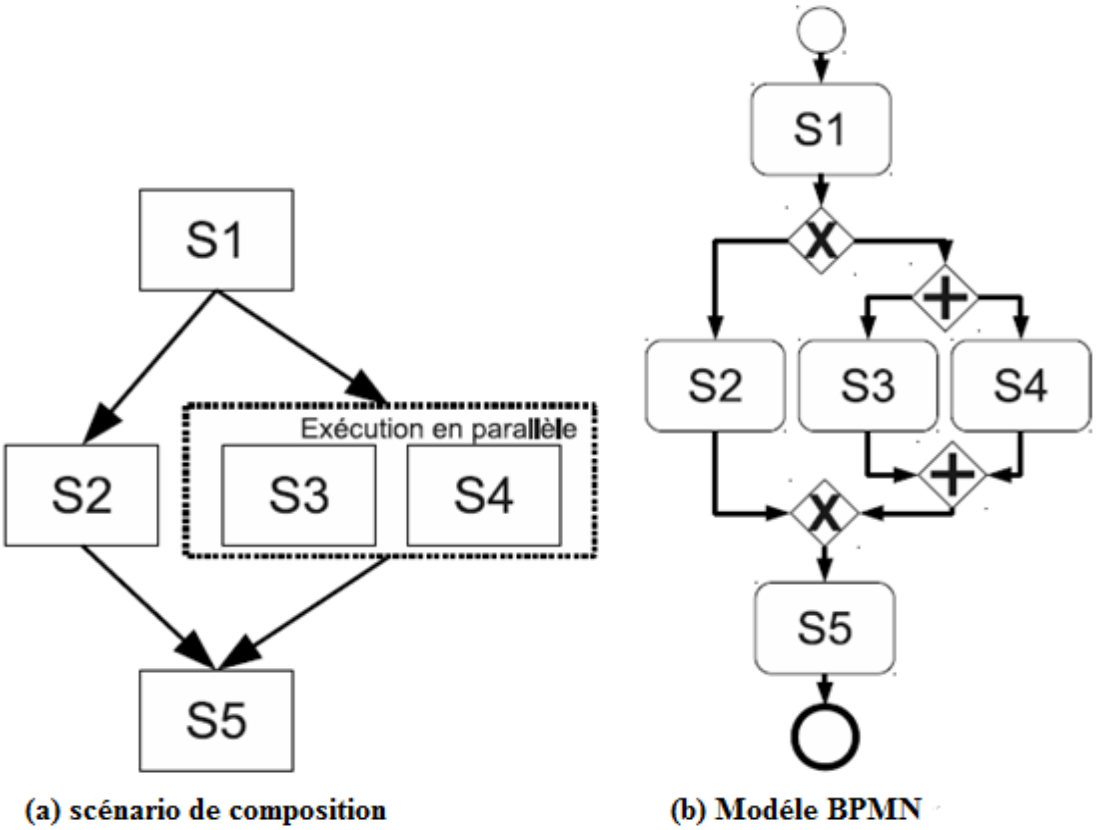


Figure 2.7 : Exemple de modélisation en BPMN pour la composition de services [89].

2.3.6. Graphes :

Les graphiques sont une structure générale et puissante de données pour la représentation des objets et des concepts. Dans une représentation graphique, les nœuds représentent généralement des objets ou parties d'objets, tandis que les arrêtes décrivent des relations entre des objets ou parties d'objets. Les graphes ont des propriétés d'invariances intéressantes. Si un graphe est établit sur papier au sens mathématique (s'il est traduit, pivoté, et transformé en son image miroir) il toujours le même graphe. Ainsi que, le fait que les graphes sont bien sui table pour la modélisation d'objets en termes de portions et de leurs relations. Ces propriétés d'invariance et de modélisation les rendre très intéressant pour diverses applications [103].

Dans [104], les auteurs ont introduit une transformation de BPEL à EPCs (Event-Driven Process Chains). Un schéma EPC plat est définit comme un graphe orienté et cohérente avec les contraintes type et cardinalité. Construit sur un mapping conceptuelle, cette approche présente un programme de transformation qui est capable de générer automatiquement des modèles EPC sous forme des fichiers EPML (format d'échange à base XML pour EPCs) à partir des définitions de processus BPEL. Une telle transformation permet de communiquer des processus BPEL aux analystes d'affaires qui sont souvent impliqués dans l'approbation du logique métier. La visualisation EPC est basée sur le comportement dynamique du modèle BPEL.

2.4. Synthèse des Outils pour la modélisation

Nous allons maintenant comparer certains modèles utilisés dans la modélisation de services Web composites Dans Tab 2.1, en se basant sur [40] [88] [96] [119] [120] :

	Réseaux de Pétri	**Algèbres de Processus**	**Automates à états finis**
Formel	+	+	+
Capacité d'analyse	+	+	+
Gérer la complexité	+	-	-
Simulation	+	-	+
Outils	+	-	-
Synchronisation	+	-	-
Modèles de propriétés statiques	+	+	+
Modèles de propriétés dynamiques	+	-	-
Détection d'inter-blocage	+	-	-
Concurrence	+	-	-

Tab 2.1 : comparaison certains modèles utilisés dans la modélisation.

En se basant sur cette table nous trouvons que les réseaux de Pétri sont plus forts que les autres modèles :

1. sémantique computationnelle riche,
2. capacité de modéliser formellement des systèmes caractérisés comme étant simultanés, asynchrones, distribués, parallèles, non déterministes et stochastiques,
3. capacité à effectuer une analyse de performance quantitative.
4. disponibilité d'outils de simulation graphique. Les réseaux de Pétri ont également une représentation naturelle des changements et de la simultanéité, qui peuvent être utilisés pour établir une sémantique opérationnelle distribuée et exécutable des services Web.

En outre, les réseaux de Pétri peuvent traiter des tâches d'analyse hors ligne telles que la composition statique des services Web, ainsi que des tâches d'exécution en ligne telles que la détermination des inter-blocages et la satisfaction des ressources. De plus, les réseaux de Pétri possèdent un moyen naturel d'aborder le partage des ressources et le transport, ce qui est impératif pour le paradigme des services Web. De plus, les réseaux de Pétri disposent d'un ensemble d'outils de simulation graphique disponibles, représentés par Design / CPN (Meta Software Corporation 1993).

Les RdPs constituent un Framework pour la modélisation de la composition de services. La facilité de la compréhension, l'utilisation de ses notations graphiques, ainsi que sa puissance de vérification et d'analyse formelles à travers un ensemble de bases et techniques mathématiques, justifient le choix de l'utilisation de cet outil pour des problématiques de modélisation et de vérification de la composition de services. De plus, différents logiciels (CPN Tools 3, TINA 4, etc.) pour la vérification du comportement des modèles RdP sont disponibles, permettant ainsi de détecter les erreurs liées à la modélisation avant l'implémentation du système.

3. Approche de réseaux de pétri

Dans cette partie nous allons présenter les réseaux de Pétri qui permettent de modéliser la structure et le comportement des services composites. D'abord, nous allons présenter le formalisme du réseau de Pétri ordinaires [105-115][118], Ensuite nous allons faire une comparaison entre les différents types des réseaux de petri afin de choisir le meilleur type. Et finalement nous allons bien détailler notre choix qui est les réseaux de Pétri colorés [106] [109] [110] [116] [117] [118].

3.1. Réseaux de pétri

3.1.1. Définition informelle

Un RdP est un graphe orienté comprenant deux sortes de nœuds : **des *places*** et **des *transitions***. Ce graphe est constitué de telle sorte que ***les arcs*** du graphe ne peuvent relier que des places aux transitions ou des transitions aux places.

Un Rdp Contient de :

- **Places** (variable d'état du système)
- **Transitions** (événement ou action)
- **Jetons** (valeur de la variable)
- **Arcs** (À chaque arc est associé un nombre entier strictement positif appelé poids de l'arc)

Un arc relie une place à une transition ou une transition à une place, mais jamais une place à une place ou une transition à une transition Cas particuliers

On représente graphiquement les places par des cercles et les transitions par des barres (Figure 2.8). Les places servent à représenter les états du système modélisé, tandis que les transitions représentent les changements d'état ou les événements.

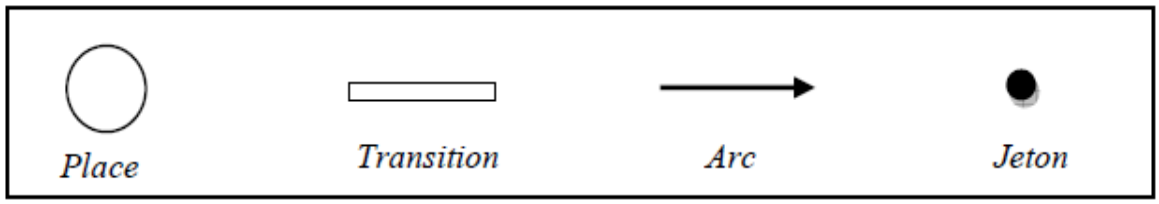


Figure 2. 8 : Représentation graphique des éléments de RdP.

Quelques des transitions et leurs places d'entrées et de sorties sont présentées dans Le tableau (Tabl 2.2).

Places d'entrées	transition	Places de sortie
Pré-conditions	Evènement	Post-conditions
Données d'entrée	Traitement	Données de sortie
Signaux d'entrée	processeur	Signaux de sorties
Ressources demandées	Tâche	Ressources libérées
Conditions	Clauses en logique	Conclusions
Buffers	Processeur	Buffers

Tab 2.2 : Quelques interprétations typiques de transitions et de places [118].

Exemple informel :

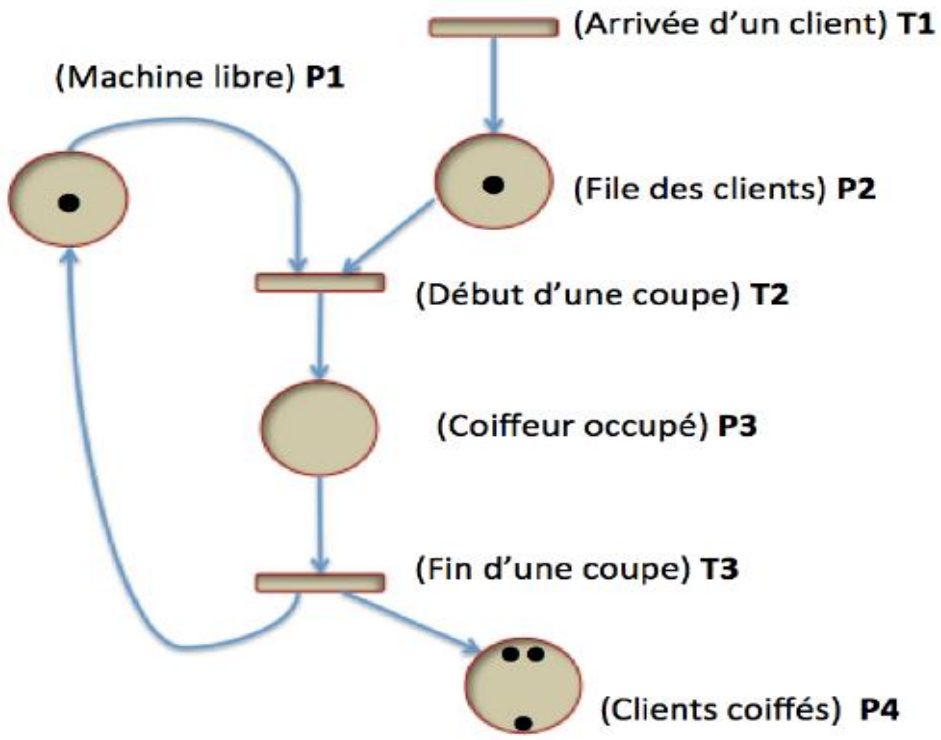


Figure 2.9 : exemple informel [115]

3.1.2. Définition formelle :

Un réseau de Pétri est un 5-uplet (P, T, A ,W, M_0) où :

– P est un ensemble fini de places

– T est un ensemble fini de transitions

– $A \subseteq (P \times T) \cup (T \times P)$ est un ensemble fini d'arcs

– W : A → N est la fonction poids associée aux arcs où on assigne à chacun un entier positif qui indique le nombre de jetons nécessaires pour franchir une transition ou le nombre de jetons produits après le franchissement d'une transition.

– M_0 : P → N est le marquage initial du modèle RdP qui correspond au nombre de jetons dans les places à l'état initial.

Exemple formelle :

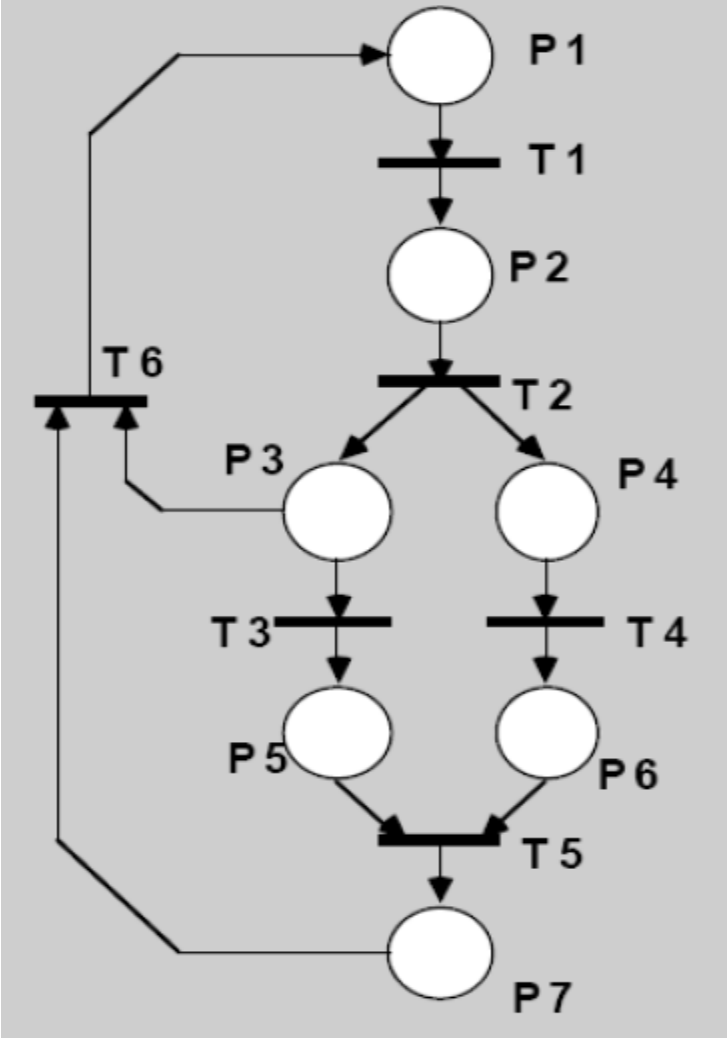


Figure 2.10 : exemple formel [105].

On dit que la place P2 est en amont ou est une entrée de la transition T2

On dira que la place P7 est en aval ou est une sortie de la transition T5

3.1.3. Marquage des places :

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

Les jetons circulent dans les places selon certaines règles (définies ci-dessous). Cette circulation symbolise l'évolution dynamique du système. Le marquage initial (celui indiqué sur le dessin) donne la position initiale des jetons.

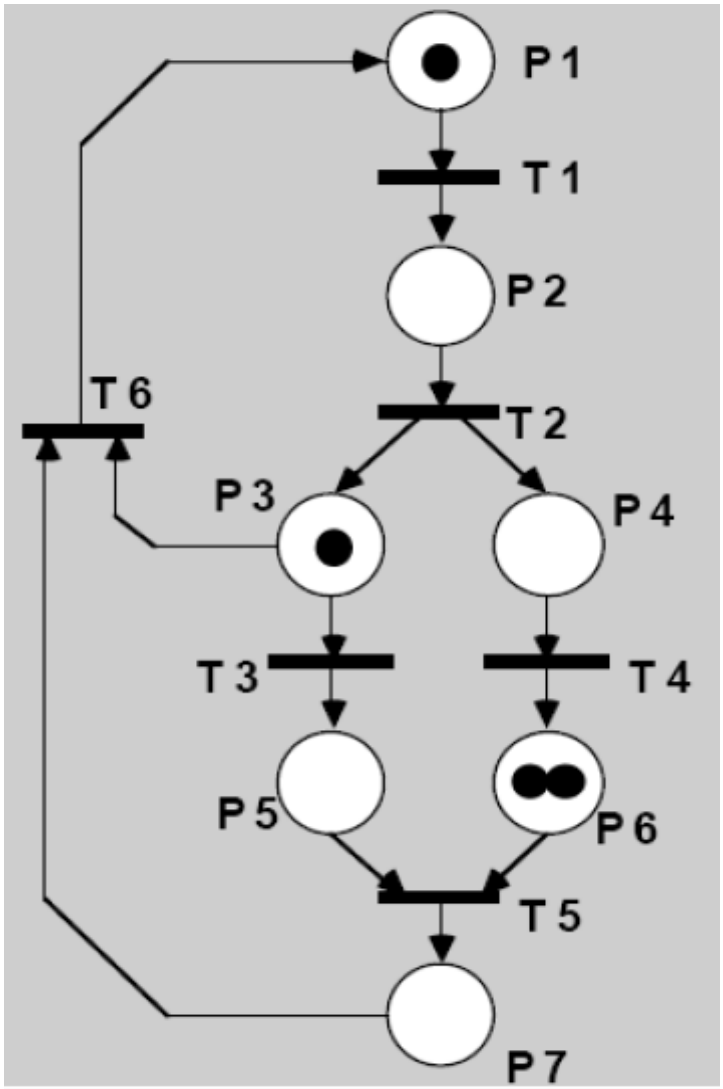


Figure 2.11 : exemple de marquage et marquage M= (1,0,1,0,0,2,0) [105].

3.1.4. Franchissement de Transition

Les règles de franchissement et de circulation des jetons sont :

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

b. Une transition sans place amont est toujours validée, on dit que c'est une transition source.

c. Le franchissement (le tir) d'une transition Tj consiste à retirer une marque (jeton) dans chacune des places en amont de la transition Tj et à ajouter une marque dans toutes les places en aval de Tj.

d. Lorsqu'une transition est validée cela n'implique pas qu'elle sera franchie immédiatement.

e. Il y a un seul franchissement à la fois.

f. Le franchissement d'une transition est indivisible.

g. Le franchissement d'une transition à une durée nulle (sauf dans les RdP temporisés qu'on va voir plus tard).

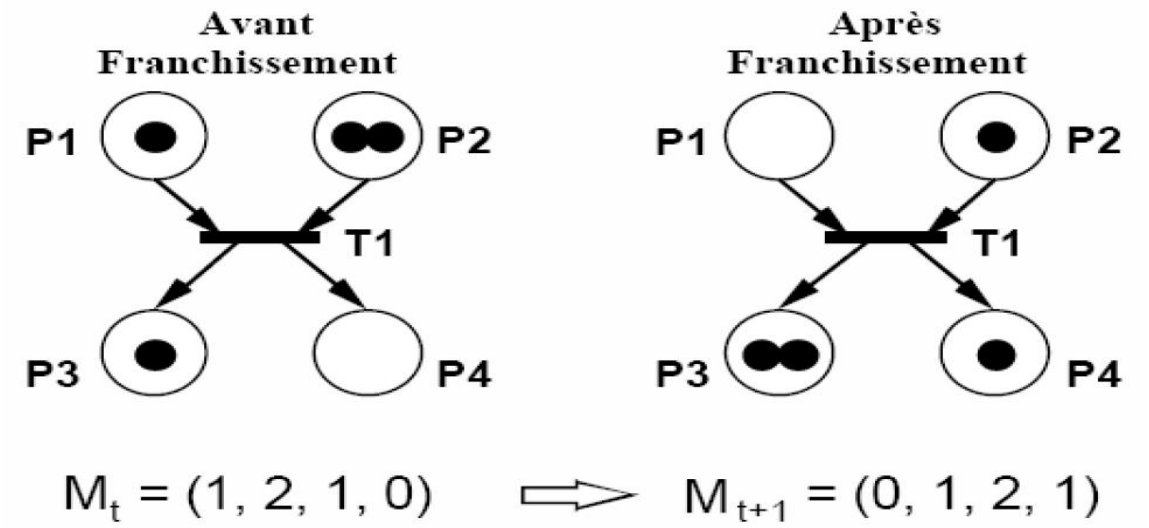


Figure 2.12 : exemple de franchissement [105].

Remarque : si une transition est validée, cela n'implique pas qu'elle sera immédiatement franchie.

3.1.5. Graphes de marquage :

L'évolution temporelle d'un RdP peut être décrite par un graphe de marquage représentant l'ensemble des marquages accessibles et d'arcs correspondant aux franchissements des transitions faisant passer d'un marquage à l'autre pour un marquage initial M0.

Exemples :

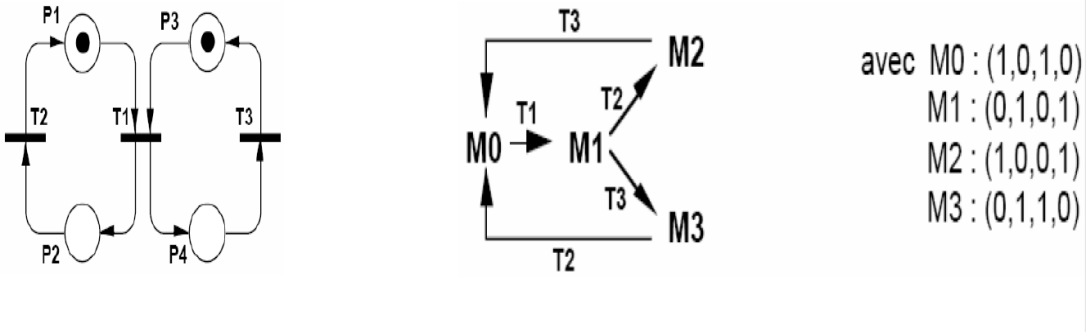


Figure 2.13 : exemple de Graphes de marquage [105].

Remarque : cette représentation permet de déterminer certaines propriétés d'un graphe. Par exemple si le graphe présente une zone non bouclé, cette partie du marquage une fois atteinte constitue un arrêt de l'évolution du RdP et celui-ci sera déclaré avec blocage.

3.1.6. Matrice d'incidence :

a- Matrice Post-incidence :

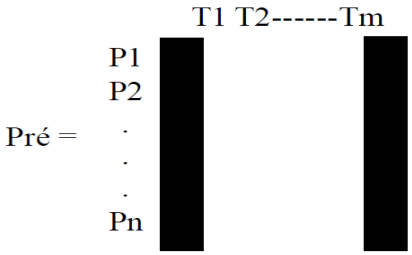
C'est une matrice à n ligne et m colonnes avec n le nombre de places et m le nombre de transitions dans le RdP.

$$Post = \begin{array}{c} \\ P1 \\ P2 \\ \cdot \\ \cdot \\ \cdot \\ Pn \end{array} \begin{array}{c} T1\ T2 \text{------} Tm \\ \begin{bmatrix} & \\ & \\ & \\ & \\ & \\ & \end{bmatrix} \end{array}$$

Chaque élément de cette matrice Post (Pi, Tj) correspond au nombre de jetons à rajouter dans Pi en franchissant Tj.

b- Matrice Pré-incidence :

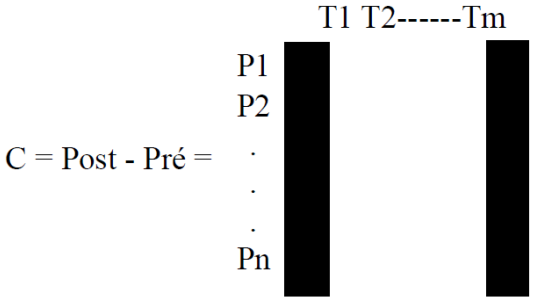
C'est une matrice à n ligne et m colonnes avec n le nombre de places et m le nombre de transitions dans le RdP.



Chaque élément de cette matrice Post (Pi, Tj) correspond au nombre de jetons à enlever dans Pi en franchissant Tj.

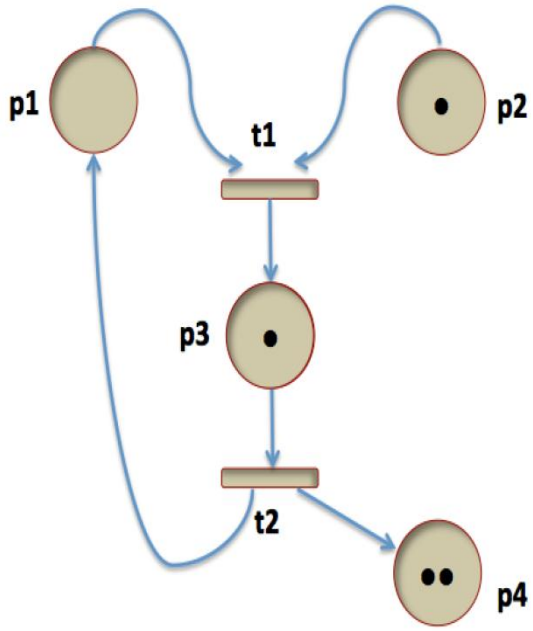
c- Matrice d'incidence :

C'est une matrice à n ligne et m colonnes avec n le nombre de places et m le nombre de transitions dans le RdP. Elle peut être calculée comme suit :



Chaque élément de cette matrice C(Pi, Tj) correspond au nombre de jetons à rajouter moins celui à enlever dans Pi en franchissant Tj.

Exemple :



$$M_0 = \begin{bmatrix}0\\1\\1\\2\end{bmatrix} \quad Pre = \begin{bmatrix}1 & 0\\1 & 0\\0 & 1\\0 & 0\end{bmatrix} \quad Post = \begin{bmatrix}0 & 1\\0 & 0\\1 & 0\\0 & 1\end{bmatrix} \quad C = Post - Pre = \begin{pmatrix}-1 & 1\\-1 & 0\\1 & -1\\0 & 1\end{pmatrix}$$

Figure 2.14 : exemple de Matrice d'incidence [115].

3.1.7. Classification des réseaux de Pétri :

3.1.7.1 . RdP autonomes

Un RdP autonome décrit le fonctionnement d'un système qui évolue de façon autonome, c'est à dire dont les instants de franchissement ne sont pas connus, ou pas indiqués. (Exemple : Le cycle des saisons).

a. RdP ordinaire

Un réseau de Pétri ordinaire est un réseau dont les poids des arcs sont tous égaux a 1.

b. RdP généralisé

Un RdP dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs. L'arc Pi → Tj a un poids p. La transition Tj ne sera validée que si Pi contient au moins p jetons.

Lors du franchissement de cette transition, *p* jetons seront retirés de la place Pi.

Lorsqu'un arc Tj → Pi a un poids *p* cela signifie que lors du franchissement de Tj, *p* jetons seront ajoutés à la place Pi.

c. RdP à capacité

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

d. RdP coloré

Un RdP coloré comporte des jetons auxquelles on attribue des couleurs (possibilité de représenter des processus parallèles)

e. RdP à arcs inhibiteurs

Un arc inhibiteur est un arc orienté qui part d'une place pour aboutir à une transition (et non l'inverse). Son extrémité est marquée par un petit cercle. La présence d'un arc inhibiteur entre une place Pi et une transition Tj signifie que la transition Tj n'est validée que si la place Pi ne contient aucun jeton. Le franchissement de la transition Tj consiste à retirer un jeton dans chaque place située en amont de la transition à l'exception de la place Pi, et à ajouter un jeton dans chaque place située en aval de la transition.

f. RdP à priorité

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

g. RDP à prédicats

Un réseau de Pétri à prédicats comporte des marques aux quelles sont associes des paramètres. En utilise ce type de réseaux lorsque les marques d'une même place ne sont pas du même type.

3.1.7.2 RdP non autonomes

Un RdP non autonome décrit le fonctionnement d'un système dont l'évolution est conditionnée par des événements externes ou par le temps (Exemple : Démarrage d'un moteur).

a. RdP Synchronisé

Un réseau de Pétri Synchronisé(RdPS) est un réseau de Pétri auquel on a associe un ensemble d'événements externes aces transitions.

b. RdP Temporisé

Un réseau de Pétri Temporise (RdPT) permet de décrire un système dont le fonctionnement dépend du temps.

c. RdP continu

Leur particularité est que le marquage d'une place est un nombre réel (positif) et non plus un nombre entier.

Remarque :

Les RdP généralisés, à capacité, coloré sont des abréviations des RdP ordinaires. Toutes les propriétés que nous allons voir dans ce qui suit peuvent donc être adaptées à ces modèles.

Les RdP à arcs inhibiteurs, les RdP à priorités, les RdP non autonomes et les RdP continus sont des extensions des RdP ordinaires. Certaines propriétés des RdP ordinaires leur sont applicables mais pas toutes.

- ✓ **Abréviations** : Des représentations simplifiées utiles pour alléger le graphisme mais auxquelles on peut toujours faire correspondre un RdP ordinaire (c'est-à-dire. un RdP autonome marqué fonctionnant selon les règles prédéfinies).
- ✓ **Extensions** : Des modèles auxquels des règles de fonctionnement ont été ajoutées afin d'enrichir le modèle initial pour aborder un plus grand nombre d'applications

3.1.8. Structures RdP :

a. Graphe d'états

Un RdP est un graphe d'états si est seulement si toute transition a exactement une place d'entrée et une place de sortie.

b. Graphe d'événements

Un RdP est un graphe d'événement si et seulement si toute place a exactement une transition d'entrée et une transition de sortie. Un Graphe d'événement est donc dual d'un graphe d'états.

c. RdP sans conflit

Un RdP dans lequel toute place a au plus une transition de sortie. Un conflit (ou conflit structurel) correspond à l'existence d'une place P1 qui a au moins deux transitions de sortie T1, T2,…On notera <P1, {T1, T2,….}>

d. RdP à choix libre

Un RdP à choix libre est un RdP dans lequel pour tout conflit <P1, {T1, T2,….}> aucune des transitions T1, T2,…ne possède une autre place d'entrée que P1.

e. RdP simple

C'est un RdP dans lequel chaque transition ne peut être concernée que par un conflit au plus. S'il existe une transition T1 et deux conflits <P1, {T1, T2,…}> et <P2, {T1, T3,…}>, alors le RdP n'est pas simple.

f. RdP pur

C'est un RdP dans lequel il n'existe pas de transition ayant une place d'entrée qui soit également place de sortie de cette transition.

g. RdP sans boucle

Un RdP sans boucle est tel qu'il existe une transition Tj et une place Pi qui est à la fois place d'entrée et place de sortie de Tj, alors Tj à au moins une autre place d'entrée.

3.1.9. Propriétés des réseaux de Petri :

a. RdP borné :

Une place Pi est dite bornée pour un marquage initial M0 si pour tout marquage accessible à partir à partir de M0 le nombre de jetons dans Pi est fini.

Un RdP est borné pour un marquage initial M0 si toutes les places sont bornées pour M0.

Si pour tout marquage M appartement à l'ensemble des marquages accessibles à partir de M0 (noté *M0), on a M(Pi) $\leq$ K où K est un nombre entier naturel, on dit que Pi est K-borné. Si la propriété est vraie pour toute place on dit que ce RdP est K-borné.

b. RdP sauf :

C'est un RdP 1-borné (ou binaire).

c. Vivacité :

Une transition Tj est vivante pour un marquage initial M0 si pour tout marquage accessible Mi $\in$ *M0, il existe une séquence de franchissement S qui contient Tj à partir de Mi (c'est à dire il subsistera toujours une possibilité de franchir Tj).

Un RdP est vivant pour un marquage initial M0 si toutes ses transitions sont vivantes pour M0 (c-à-d aucune transition ne sera jamais définitivement infranchissable)

d. Blocage :

Un blocage (ou état puits) est un marquage tel qu'aucune transition n'est validée.

Un RdP est dit sans blocage pour un marquage initial M0 si $\forall$ marquage accessible Mi $\in$ *M0, il est sans blocage.

3.1.10. Graphe des marquages et arbre de couverture

Pour pouvoir trouver si tel RdP présente telle ou telle propriété, il existe principalement 3 classes de méthodes :

- ✓ Établissement du graphe de marquage ou de l'arbre de couverture
- ✓ Utilisation des méthodes basées sur l'algèbre linéaire : résultats puissants.
- ✓ Les méthodes de réduction des RdP.

➢ **Graphe des marquages** :

Il est composé de nœuds qui correspondent aux marquages accessibles et D'arcs correspondant aux franchissements de transition faisant passer d'un marquage à l'autre.

➢ **Arbre de couverture :**

Quand on ne peut pas construire le graphe des marquages (RdP non borné), on construit un arbre de couverture qui possède un nombre fini de nœuds. Un arbre est un graphe particulier dans lequel il n'y a pas de boucle ni de circuit.

3.1.11. Synthèse des types des réseaux de pétri

Type d'extension du réseau de Pétri	Objectif de modélisation
Coloré	Analyse qualitative des propriétés du système
Temporisés	Décrit un système dont le fonctionnement dépend du temps
Stochastique	Simulation d'un system à faible concentration
Hybride	Simulation d'un système

Tab 2.3: comparaison des types des réseaux de pétri.

Dans notre travail, nous allons nous concentrer sur l'utilisation des réseaux de Pétri colorés. Les réseaux de Pétri colorés permettent de modéliser l'information dans un système de production. Contrairement aux réseaux de Pétri places-transition il est possible d'avoir dans une même place des jetons de nature différente. Si une place représente un stock, avec les réseaux de Pétri colorés il est possible de mettre dans cette place deux jetons de différentes natures représentant des produits différents alors qu'avec les réseaux de pétri simples cela n'est pas possible car il sera impossible de distinguer les deux jetons.

Par exemple :

L'exemple proposé à présent va permettre une première approche de la démarche de coloration. Il s'agit d'étudier la modélisation de l'état de trois machines similaires traitant tour à tour deux types de pièces C1 et C2. La première modélisation (figure 2.15) se base sur

l'utilisation de réseaux de Pétri ordinaires. Ce RdP se présente sous la forme de trois boucles similaires et totalement indépendantes. Le marquage initial indique que les machines 1 et 2 sont entrain de traiter une pièce de type 1, alors que la machine 3 traite une pièce de type 2.

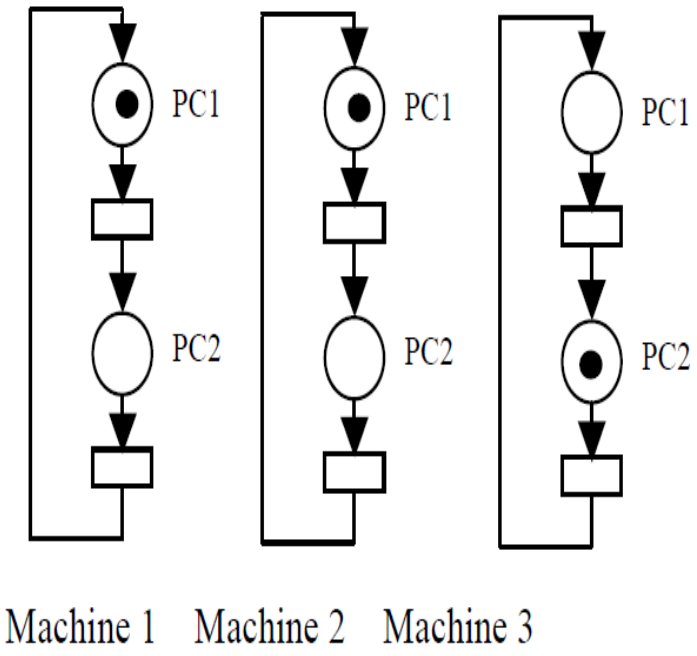


Figure 2.15 : Réseau de Pétri pour trois machines.

Il apparaît que le graphisme du modèle va se dupliquer et qu'il serait souhaitable que l'information soit alors plus condensée parce que il représenter la situation de n machines identiques.

La figure 2.16 modélise le même problème par un réseau de Pétri coloré. Le principe des RdPC va être de traduire l'idée suivante: En fait, dans le RdP initial, le même modèle est utilisé, dupliqué 3 fois. Il est alors intéressant de concentrer toutes les informations dans une seule boucle

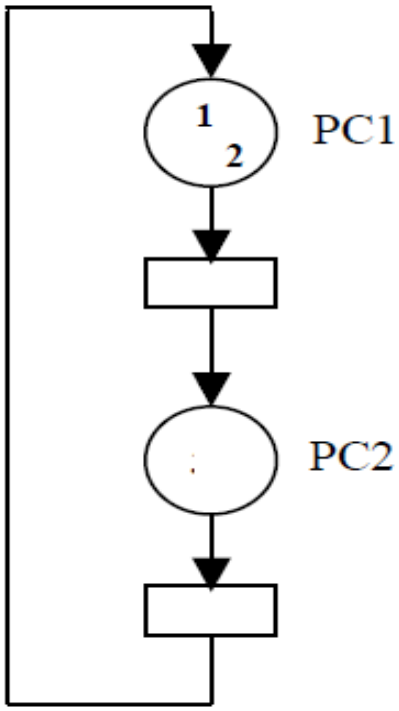


Figure 2.16 : Réseau de Pétri Coloré

Chaque jeton de ce modèle représente à présent une machine spécifique. Les deux premières machines, symbolisées par les couleurs <1> et <2>, sont entrain de traiter des pièces de type C1. La machine 3 traite une pièce de type C2. Les jetons <1>, <2> et <3> sont alors respectivement situés dans les places PC1, PC1 et PC2.

- Dans un système de production flexible, les produits peuvent être de nature diverse c'est pour cela que l'utilisation des réseaux de pétri colorés s'avère nécessaire. De plus les couleurs d'un réseau de Pétri peuvent représenter n'importe quel type de données (réels, entiers, chaîne de caractères, structure etc.).

3.2. Réseaux de Pétri colorés

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

- **Association de couleur**

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

- **Association de fonctions aux arcs**

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

3.2.1. Définition informelle

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

3.2.2. Définition formelle

Des jetons colorés : on les compte par couleur dans un multi-ensemble.

- **D'abord on définit des multi-ensembles :**

Un multi-ensemble sur C est une fonction de C vers N.

Un multi-ensemble est un ensemble pouvant contenir plusieurs occurrences d'un même élément. Chaque multi-ensemble a sur C peut être représenté par la somme formelle $a = \sum_{c \in C} a(c).c$ où l'entier positif ou nul a(c) désigne le nombre d'occurrences de l'élément y dans le multi-ensemble a. Par exemple, la somme m = 2.α + 3.β désigne le multi-ensemble contenant 2 fois la valeur α et 3 fois la valeur β sur un ensemble contenant les éléments α et β. Nous notons Bag(C) l'ensemble des multi-ensembles sur C.

Pour deux multi-ensembles $\mu, \mu' \in$ Bag(C). $\mu \leq \mu'$ si $\forall c \in C, \mu(c) \leq \mu'(c)$.

On les note parfois comme une somme d'éléments de C.

- **Définition formelle**

Un réseau de Pétri coloré est un neuf-uplet (P, T, A, C, $\sum$, W^-, W^+, G , M_0) où :

– P est un ensemble de places. $P \cap T = \emptyset$,$P \cap T = \emptyset$

– T est un ensemble de transitions.

– A est un ensemble fini d'arcs tel que : $A \subseteq (P \times T) \cup (T \times P)$

– C définit pour chaque place et chaque transition son domaine de couleur $P \oplus T$

– $\sum$ est un ensemble fini non vide d'ensembles de couleurs.

– W^- (= Pré) (resp. W^+ = Post), indexée sur P x T, est la matrice d'incidence arrière (resp. avant) du réseau.

– W^- (p, t) et W^+ (p, t) sont des fonctions linéaires de couleurs définies de Bag(C(t)) dans Bag(C(p)).

– G la garde, telle que : $\forall t \in T, G(t) \subset C(t)$.

G : est une fonction de garde qui assigne une garde à chaque transition *t* de manière à ce que Où Bool est une variable booléenne. *Type[G(t)]* = Bool

– M_0 est le marquage initial du réseau ; c'est un vecteur indexé par P et M_0 (p) est un élément de Bag(C(p)) :

p ∈ P, m(p) ∈ Bag(C(p)).

3.2.3. La dynamique d'un RDPC :

Un marquage associe à chaque place p un multi-ensemble sur C(p).

Un marquage M est un vecteur indexé par P, avec M(p) ∈ Bag(C(p))

Une transition t est franchissable pour une instance c_t ∈ C(t) et un marquage M si et seulement si :

– Soit *t* est non gardée, soit la garde vaut 1 (= vrai) pour c_t

$\forall$ p ∈ P, M(p) $\geq W^-$ (p, t) (c_t)

Le marquage M´ atteint après le franchissement de *t* pour une instance c_t à partir du marquage **M** est défini par :

$\forall$ p ∈ P, M' (p) = M(p) - W^- (p, t) (c_t) + W^+ (p, t) (c_t)

On note : M [(t, ct)> M' et $M \xrightarrow{(t,c_t)} M'$

Exemple de franchissement :

Soit x1 ∈ C1, x2 ∈ C2

• t est franchissable pour (x1, x2) si

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

2) P2 contient au moins un jeton de couleur x2

• Si on franchit t pour (x1, x2) alors

1) Un jeton de couleur x1 est retiré de P1

2) Un jeton de couleur x2 est retiré de P2

3) Un jeton de couleur <x1, x2> est produit dans P3 : <X1, X2> (x1, x2) = <x1, x2>

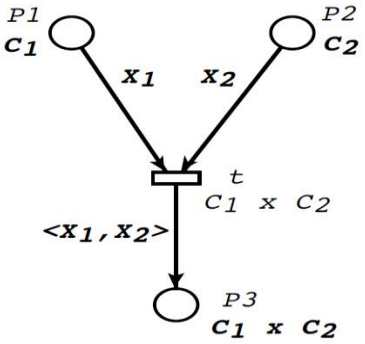


Figure 2.17 : dynamique d'un RDPC **[**116]

3.2.4. Couleurs et fonctions classiques de pré/post-condition

Un domaine de couleur construit par produit cartésien des classes de couleurs, dans lequel la classe C_i apparaît e_i fois.

$$\mathbf{C} = \prod_{i=1}^{n} \prod_{j=1}^{e_i} C_i$$

Dans la pratique, C'est le domaine d'une transition.

$$\mathbf{C} = <\boldsymbol{c_1^1}, \boldsymbol{c_1^2}, \ldots\ldots\ldots\ldots\ \boldsymbol{c_1^{e1}}, \ldots\ldots\ldots, \boldsymbol{c_n^1}, \boldsymbol{c_n^2}, \ldots\ldots\ldots, \boldsymbol{c_n^{e1}}> \in \mathbf{C}$$

• Identité/Projection :

Notée par une variable : X, Y, ou X_1, ou X_1^1 **,** ou p, q, …

$$X_i^j(\mathbf{c}) = c_i^j$$

Y (<x, y>) = y

q (<p, q, r>) = q

Un domaine de couleur *ordonné circulairement* de taille k est un ensemble isomorphe à ({0,…., k – 1}). On écrit alors x_{++} au lieu de $x_{+1[k]}$.

- Successeur (sur C_i ordonnée circulairement) : Notée $X_1{+}{+}$ ou ($X_i \oplus 1$) ou $X_i!$

La relation successeur est définie par l'ordre d'énumération des éléments de la classe C_i est :

$$X_i^j{+}{+}(\mathbf{c}) = \mathbf{successeur}\ (c_i^j)$$

- Diffusion / Synchronisation (sur C_i)

  Notée C_i.All ou S_{C_i} $\quad C_i.All(c) = \sum_{x \in C_i} x$

  La valeur est indépendante de l'élément choisi dans C_i

3.2.5. Fonctions arcs aval, fonctions arcs amont :

Il est maintenant nécessaire de définir plus précisément une fonction sur un arc aval à une transition. C'est la fonction qui transforme une couleur après franchissement de cette transition.

Il représente quelques exemples de franchissement d'une transition par plusieurs couleurs. Les arcs amont et aval à la transition portent des fonctions.

Dans ce réseau de Pétri à quatre places et une transition circulent des jetons de trois couleurs différentes <a>, <b>, <c>. Les arcs sont affectés de quatre types de fonctions f, g, h et Id (fonction Identité).

Il convient de définir ces fonctions, par leur action sur chacune des couleurs du réseau. Ces fonctions sont ainsi définies dans ce l'exemple :

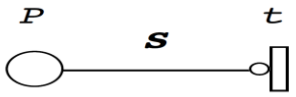
f < a > = < b >	g < a > = < c >	h < a > = < a >	Id < a > = < a >
f < b > = < a > + < b >	g < b > = < b >	h < b > = < a > + < b >	Id < b > = < b >
f < c > = < b > + < c >	g < c > = < a >	h < c > = < c >	Id < c > = < c >

3.2.6. Conditions de franchissable de la transition :

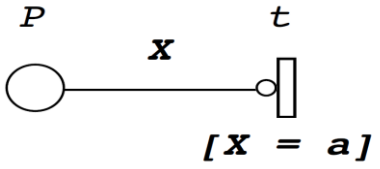
Pour qu'une transition T soit franchissable pour une couleur < c >, il faut trouver dans chaque place Pi amont à T, un ensemble de couleurs C_i tel que f_i< c > = C_i ; où f_i est la fonction portée par l'arc amont reliant la place Pi à la transition T. Dans l'exemple traité, les fonctions f, g et h sont définies sur les trois couleurs <a>, <b>, <c>.

3.2.7. Arc inhibiteur coloré

✓ Pour tester qu'une place est vide



✓ Pour tester qu'une place ne contient pas de jeton de couleur a



Exemple d'un Réseaux de Pétri colorés :

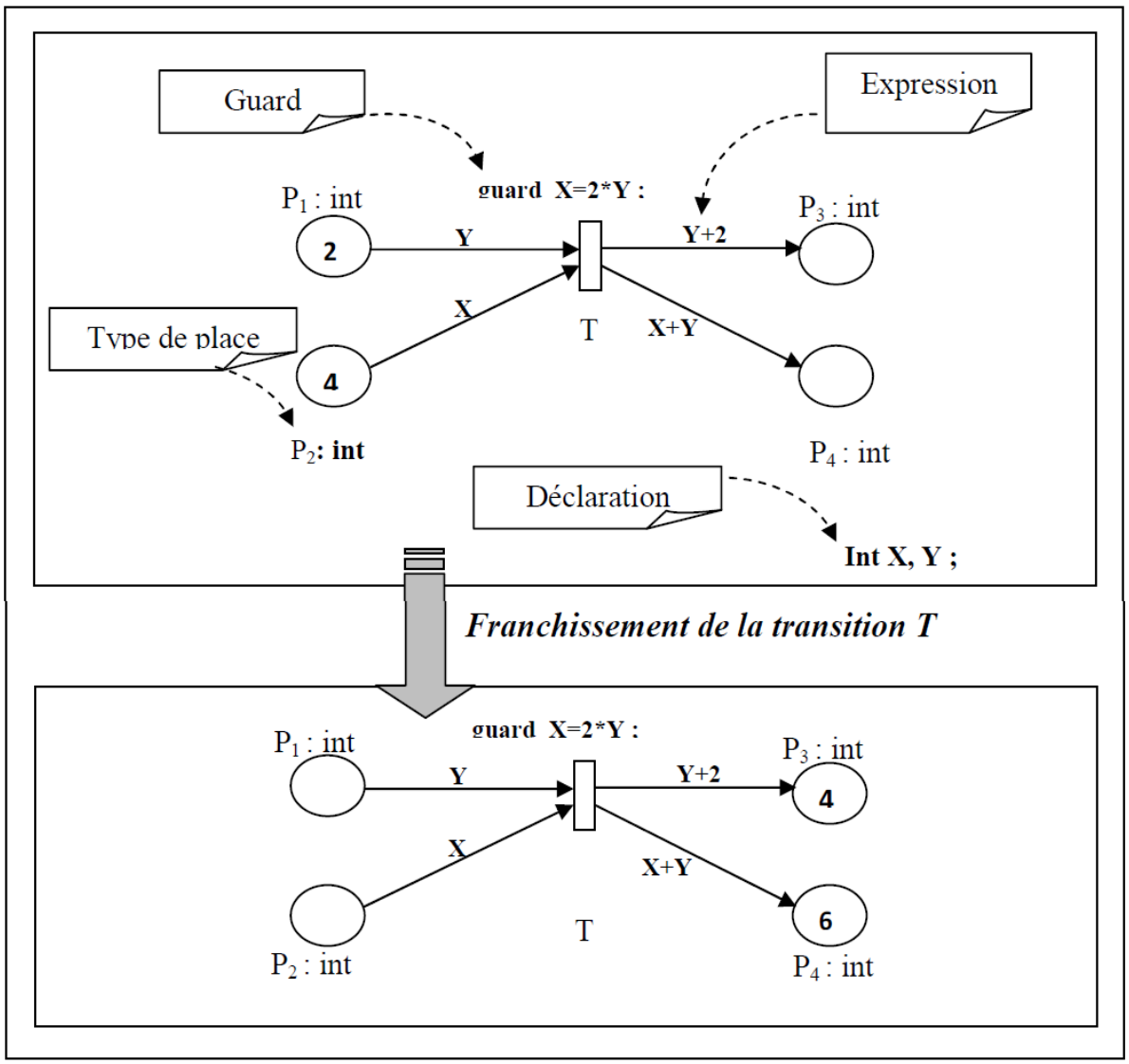


Figure 2.18 : Réseau de Pétri coloré [118]

4. Conclusion :

La modélisation, vérification et validation des services composites constituent aujourd'hui un enjeu crucial. Nous avons concentré ce chapitre sur les réseaux de Pétri, qui présentent un formalisme de modélisation très puissant car ils fournissent un cadre formel de spécification de ces services Web composites. Dans le chapitre suivant nous allons détailler une architecture qui permet la modélisation des services composite en utilisant les RdPC.

Chapitre 3

« La conception de system »

1. Introduction:

BPEL est comme tous les langages de composition proposés manque de formalisme permettant la vérification et la validation de la composition de services. Il n'est donc pas possible d'appliquer directement les méthodes mathématiques sur ce langage, rendant ainsi la vérification difficile. En effet, les processus de validation et de vérification visent à assurer la conformité des systèmes aux exigences requises et le respect des caractéristiques de conception attendues. Ces processus apportent tout leur intérêt pour conclure sur la bonne description du modèle, et pour s'assurer du bon fonctionnement du système étudié avant sa réalisation et son implémentation. Par conséquent, il est indispensable de prendre en compte cette étape en se basant sur d'autres approches et modèles pour la spécification formelle de la composition de services. Nous avons prouvé dans le chapitre précédent que les Réseaux de Colorés CP-nets présentent un formalisme efficace et capable de bien spécifier un service composite.

Dans ce chapitre nous allons présenter la conception de notre système de modélisation formelle des services composites résultants d'une composition avec BPEL. D'abord nous allons présenter l'analyse du langage BPEL, ensuite comment transformer un fichier BPEL en CP-nets et enfin une description détaillée de l'architecture générale de notre système sous forme de modules.

2. Analyse du langage BPEL

2.1. Spécification du langage BPEL

Une composition des services Web spécifié en langage BPEL est composée de quatre parties qui sont [65]:

1. *Les attributs supérieurs :* Cette première partie permet la déclaration des informations concernant le processus métier comme son nom, les espaces de nom supportés, les documents WSDL ou schémas XML importés par le processus métier.
2. *Lien de partenaire :* Cette deuxième partie permet la déclaration des Partnerlinks : ces derniers définissent une liaison entre les ensembles d'opérations des services invoqués par le service Web composite décrit en BPEL, et un nom de liaison bien précis du service Web engagé dans la composition. Ces liaisons sont appelées Partner permettent ainsi

d'identifier facilement les différents services Web utilisés dans cette composition (processus métier).

3. *Les variables :* Cette troisième partie permet de déclarer des variables, utilisant les types importés de descriptions WSDL associées à la spécification BPEL. Ces variables seront utilisées dans la partie description comportementale du processus métier, pour maintenir un état au niveau du service Web, et ainsi assurer des liaisons de données entre deux opérations.
4. *Les activités :* Les activités décrivent le comportement du processus métier lui-même en utilisant différents opérateurs. Elles peuvent être basiques ou complexes. Les activités complexes sont composées d'autres activités basiques ou complexes.

Les deux tableaux, Tab (3.1) et Tab (3.2) illustrent les différentes activités basiques et complexes du langage BPEL.

Les activités basiques	Rôles
<*invoke*>	Invoque une opération dans un service web.
<*receive*>	Attend un message d'une source externe.
<*reply*>	Réponde à une source externe.
<*assign*>	Attend un certain moment.
<*wait*>	Copie les données d'une place à l'autre.
<*terminate*>	Termine l'instance de service en entier.
<*empty*>	Ne fait rien.
<*throw*>	Lance une erreur d'exécution.

Tab 3.1: Activités basiques du langage BPEL. [65]

Les activités complexes	Rôles
<*sequence*>	Définit un ordre d'exécution.
<*switch*>	Exprime l'acheminement conditionnel des tâches.
<*while*>	Exprime la boucle.
<*pick*>	Attend l'arrivée d'un évènement.
<*flow*>	Exprime l'acheminement parallèle des tâches.

Tab 3.2: Activités complexes du langage BPEL. [65]

2.2. Les activités de BPEL

Tout processus BPEL dispose d'une seule activité principale. Cette activité pourrait correspondre à l'une des activités structurées, des activités base(ou basiques) et des activités de communication qui sont offertes par le langage BPEL. [66].

2.2.1. Les activités de base

Les activités de base sont des activités simples sont : **<empty>**, **<throw>**, **<exit>**, **<assign>, <wait>** et **<rethrow>.**

empty : modélise une absence d'activité ou une activité vide.

```
<empty standard-attributes>
      standard-elements
</empty>
```

throw : Permet de signaler une erreur interne d'une façon explicite.

```
<throw faultName="qname" faultVariable="ncname" ?
                    standard-attributes>
                     standard-elements
 </throw>
```

exit : Permet de mettre fin à l'exécution d'une instance d'un processus, d'une façon immédiate.

assign : Permet de copier des données d'une variables à une autre, de même que de construire de nouvelles données à base d'expressions, e.g., expressions XPath.

wait : Indique soit une durée soit une date limite pendant/avant laquelle l'exécution du processus est bloquée.

rethrow : Permet de lever (à une autre portée) une erreur ayant été initialement capturée par le **<faultHandler>** englobant.

```
<sequence>
        <wait until=’TravelApproval’/>
        <invoke operation="employeeTravelStatus"/>
        <assign>
            <copy>
                <from variable="TravelRequest" part="flightData"/>
                <to variable="FlightDetails" part="flightData"/>
             </copy>
        </assign>
</sequence>
```

2.2.2. Les activités de communication

Parmi les activités les plus importantes, on trouve celles qui sont liées à la réception et à l'envoi des messages, respectivement, de la part et à destination des partenaires. Ces activités sont [68]:

receive : permet aux partenaires de faire appel à une des opérations exposées par le processus avec lequel ils interagissent. Elle bloque l'exécution séquentielle jusqu'à recevoir un message dont les propriétés concordent avec les valeurs des attributs ***portType*** et ***operation***. Une fois reçu, ce message est sauvegardé par la variable qu'elle spécifie. Il faut aussi noter que cette activité joue un rôle clé dans le cycle de vie d'un processus puisqu'elle permet son instanciation si l'attribut ***createInstance*** est à **yes**.

```
<receive  name = "Receive1"
          createInstance= "yes"
          PartnerLink= "PartnerLink1"
          operation= "exampleOperation"
          portType= "examplePortType"
          variable= "var1In" />
```

reply : Est utilisée pour l'envoi d'un message en réponse à un premier message ayant été reçu à travers une activité <***receive***> précédemment exécutée sur le même ***partnerLink***, ***portType*** et ***operation***. Le résultat d'un <***reply***> peut prendre deux formes: la première forme correspond au cas où le <***reply***> résulte en une réponse normale qui sera portée par la variable qu'il spécifie, tandis que la deuxième correspond au cas où ce dernier résulte en une erreur dont le nom sera spécifié par l'attribut ***faultName***.

```
<reply partnerLink="lncname"
        portType="lqname"
       operation="ncname"
       variable="ncname"?
       faultName="lqname "?
</reply>
```

invoke : Permet l'invocation d'une opération exposée par un service Web. Cette invocation peut être synchrone (sous forme d'une requête/réponse) ou asynchrone (sous forme d'une requête à sens unique). La déclaration d'une invocation synchrone requiert la présence et d'une ***inputVariable*** spécifiant le message à envoyer et d'une ***outputVariable*** pour la sauvegarde du message à recevoir. Contrairement à cela, la déclaration d'une invocation asynchrone n'exige la présence que d'une ***inputVariable***. Une invocation synchrone peut aussi résulter en un message reportant une erreur.

```
<invoke partnerLink="lncname"
          portType="lqname"
         operation="ncname"
         inputVariable="ncname"?
        outputVariable="lncname"?
</invoke>
```

2.2.3. Les activités structurées

Les activités structurées dotent le langage BPEL de cette capacité à exprimer des compositions de services selon une approche structurée, et les activités structurées décrivent l'ordre d'exécution de leurs sous-activités. [67]

Les activités **<sequence>**, **<if>**, **<while>**, **<repeatUntil>,** <**forEach**> et <**scope**> permettent un contrôle séquentiel.

L'activité **<flow>** permet quant à elle une exécution parallèle de ses activités ainsi que leur synchronisation.
Enfin, l'activité **<pick>** définit un choix contrôlé par l'arrivée d'un événement.

Sequence : Contient (une à) plusieurs activités, structurées ou de base, qui devront être exécutées en séquence, suivant l'ordre selon lequel elles sont listées.

```
<sequence  name= "Main" >
        [Actions]
</sequence>
```

while : Permet une exécution répétitive de l'activité qu'elle enveloppe, et cela tant que la condition qu'elle définit est maintenue à vrai.

```
<while >
    <condition> bool-expr </condition>
        activity
</while>
```

repeat Until : Permet une exécution répétitive de l'activité qu'elle enveloppe, ct cela tant que la condition qu'elle définit est maintenue à faux.

```
<repeatUntil standard-attributes>
    activity
        <condition >
            bool-expr
        </condition>
</repeatUntil>
```

if : Permet de sélectionner, à partir d'une liste ordonnée de choix, une et une seule activité à exécuter.

pick : attend qu'un message particulier (un événement extérieur) arrive ou qu'une alarme soit déclenchée. Quand l'un des deux événements se produit, l'activité associée est ainsi exécutée.

```
<pick createInstance="yes|no"? standard-attributes>
        standard-elements
          <onMessage partnerLink="ncname" portType="qname"
              operation="ncname" variable="ncname" ?>+
            <correlations>?
                <correlation set="ncname" initiate="yes|no" ?>+
            </correlations>
                    activity
          </onMessage>
</pick>
```

flow : permet une exécution parallèle d'un ensemble d'activités et de spécifier un ordre partiel d'exécution. Une ou plusieurs de ses sous-activités seront donc concurrentes. Les liens définis dans cette activité permettent de synchroniser et/ou de spécifier des dépendances temporelles entre les sous-activités de <***flow***>.

```
<flow standard-attributes>
        standard-elements
          <links> ?
            <link name="ncname">+
          </links>
            activity+
</flow>
```

forEach : Permet d'exécuter, d'une façon parallèle ou séquentielle, un certain nombre d'instances de l'activité <***scope***> qu'elle enveloppe.

scope : Définit un contexte d'exécution complet pour l'activité qu'elle enveloppe. Ce contexte a une structure similaire à celle d'un processus vu dans sa globalité. Un <***scope***> pourrait aussi définir des constructeurs <***compensationHandlers***> et <***terminationHandlers***>. Les premiers permettent de défaire le travail exécuté par le <scope> auxquels ils sont attachés, Alors que les deuxièmes permettent à ce dernier de contrôler, à un certain niveau, la sémantique de sa terminaison forcée.

3. Transformation de BPEL en CP-nets

La composition des services Web est définie comme des workflows composés d'activités dans WS-BPEL. Pour transformer un service composite spécifié par WS-BPEL à un modèle CP-nets, les activités sont représentées par une transition et les relations de flux de contrôle entre les activités sont capturées avec les règles de tir de jeton CP-nets. Il existe deux types d'activités: l'activité de base et l'activité structurée. Les activités structurées sont combinées par des activités de base ou d'autres activités structurées utilisant certaines constructions.

3.1. Transformation des activités de base

- ✓ **L'activité receive :** L'activité <receive> spécifie le lien Partner dont il s'attend à recevoir, et portType et operation le Partner. La transformation se réfère à la Figure 3.1.

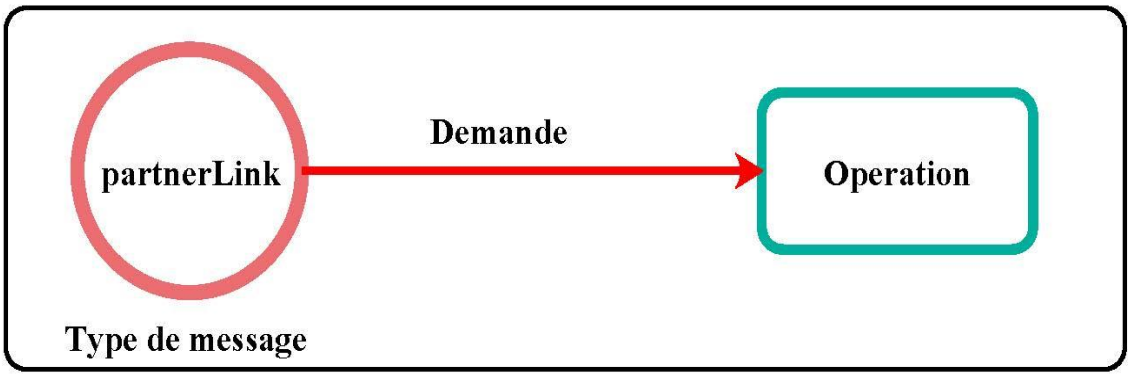


Figure 3.1 : Transformation l'activité recevoir en CP-net.

- ✓ **L'activité reply :** est utilisée pour envoyer une réponse à une request précédemment acceptée via une activité receive. De telles réponses ne sont significatives que pour les interactions synchrones. Une réponse asynchrone est toujours envoyée en invoquant l'opération one-way correspondante sur le lien partnerlink. Une activité reply peut spécifier une variable qui contient les données de message à envoyer en reply. La transformation se réfère à la Figure 3.2.

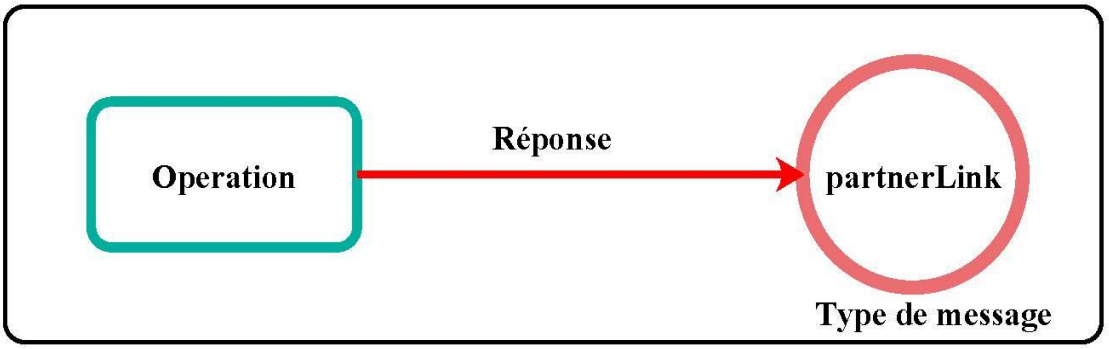


Figure 3.2 : Transformation l'activité reply en CP-net.

- ✓ **L'activité invoke :** Invoquer d'une opération peut être une request/response synchrone ou une opération one-way asynchrone. BPEL utilise la même syntaxe de base pour les deux avec des options supplémentaires pour le cas synchrone. l'invocation asynchrone requiert uniquement la variable input de l'opération. Une invocation synchrone requiert nécessite à la fois une variable input et une variable output. La transformation se réfère à la Figure 3.3

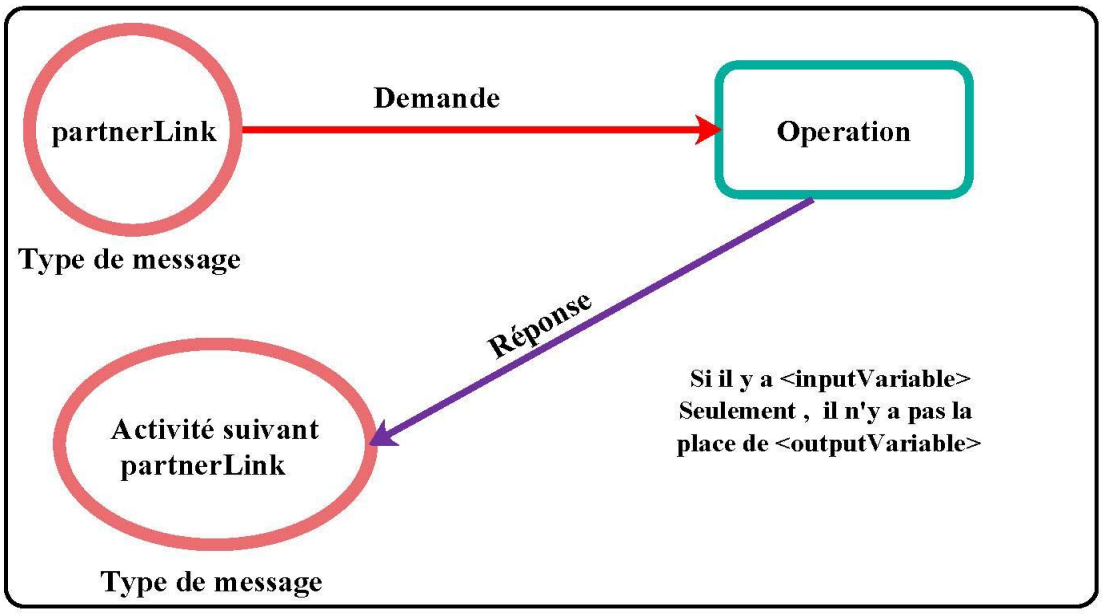


Figure 3.3 : Transformation l'activité invoke en CP-net.

- ✓ **L'activité empty :** La construction <empty> vous permet pour insérer une instruction non opération dans un business Processus. La transformation se réfère à la Figure 3.4.

Figure 3.4 : Transformation l'activité empty en CP-net.

3.2. Transformation des activités structurées

Pour permettre la présentation de structures complexes dans BPEL, les activités structurées suivantes sont définies: ***sequence***, pour définir un ordre d'exécution, ***switch***, pour le routage conditionnel, ***while***, pour la boucle, ***pick***, pour les conditions de course basées sur le temps ou les déclencheurs externes, et flow, pour le routage parallèle.

- ✓ **L'activité sequence :** La construction <sequence> de BPEL contient une ou plusieurs activités qui sont exécutées séquentiellement, dans l'ordre dans lequel elles sont listées dans l'élément <sequence>. La transformation se réfère à la Figure 3.5.

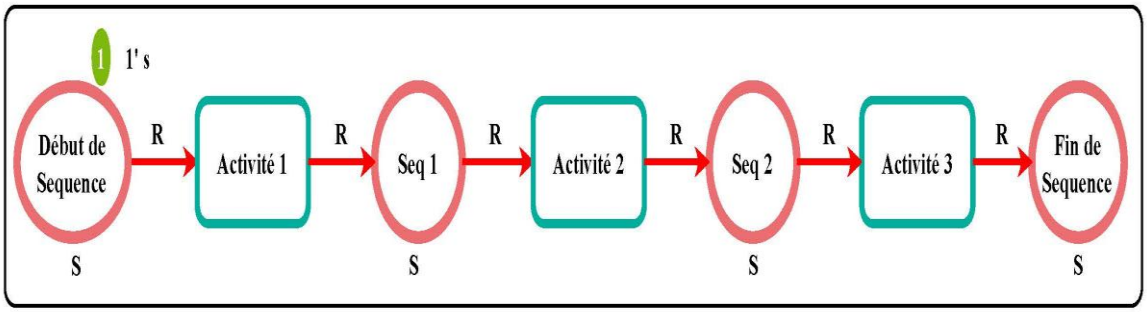


Figure 3.5 : Transformation l'activité sequence en CP-net.

- ✓ **L'activité switch :** La construction <switch> de BPEL supporte le comportement conditionnel. L'activité consiste en une liste ordonnée d'une ou de plusieurs branches conditionnelles. Les branches du switch sont considérées dans l'ordre dans lequel elles apparaissent. La première branche dont la condition est vraie est prise et fournit l'activité effectuée pour le switch. Si aucune branche avec une condition n'est prise, Alors l'activité du switch est terminée lorsque l'activité de la branche sélectionnée est terminée. La transformation se réfère à la Figure 3.6.

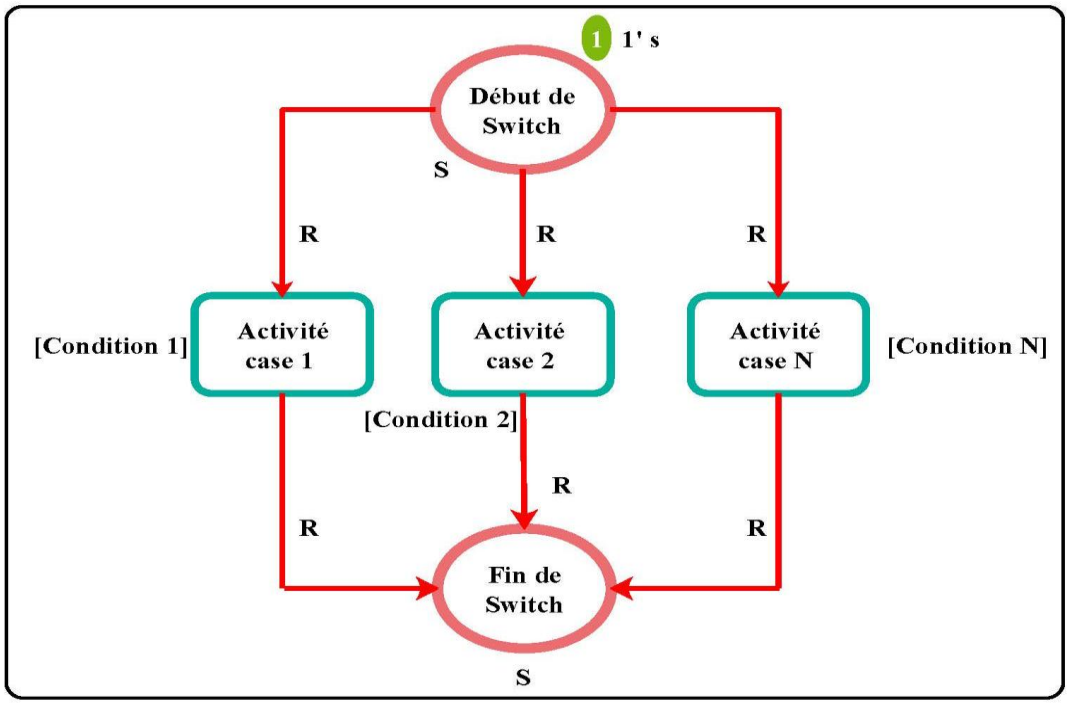


Figure 3.6 : Transformation l'activité switch en CP-net.

✓ **L'activité while :** La construction <while> de BPEL prend en charge les performances répétées d'une activité itérative spécifiée jusqu'à ce que la condition booléenne donnée n'ait plus la valeur true. La transformation se réfère à la Figure 3.7.

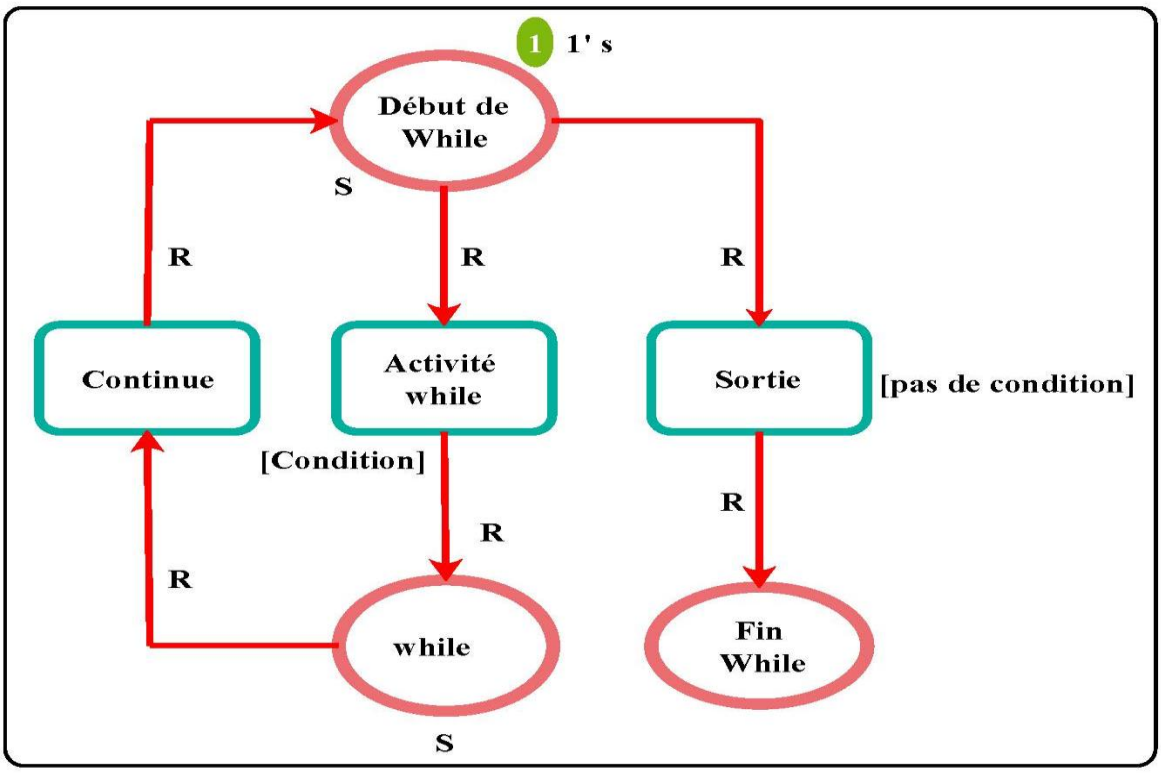


Figure 3.7 : Transformation l'activité while en CP-net.

✓ **L'activité pick:** L'activité <pick> attend l'occurrence d'un ensemble d'événements, puis exécute l'activité associée à l'événement qui s'est produit. L'occurrence des événements est souvent mutuellement exclusive. L'activité pick se termine lorsque l'une des branches est déclenchée par l'occurrence de son événement associé et que l'activité correspondante est terminée. La transformation se réfère à la Figure 3.8.

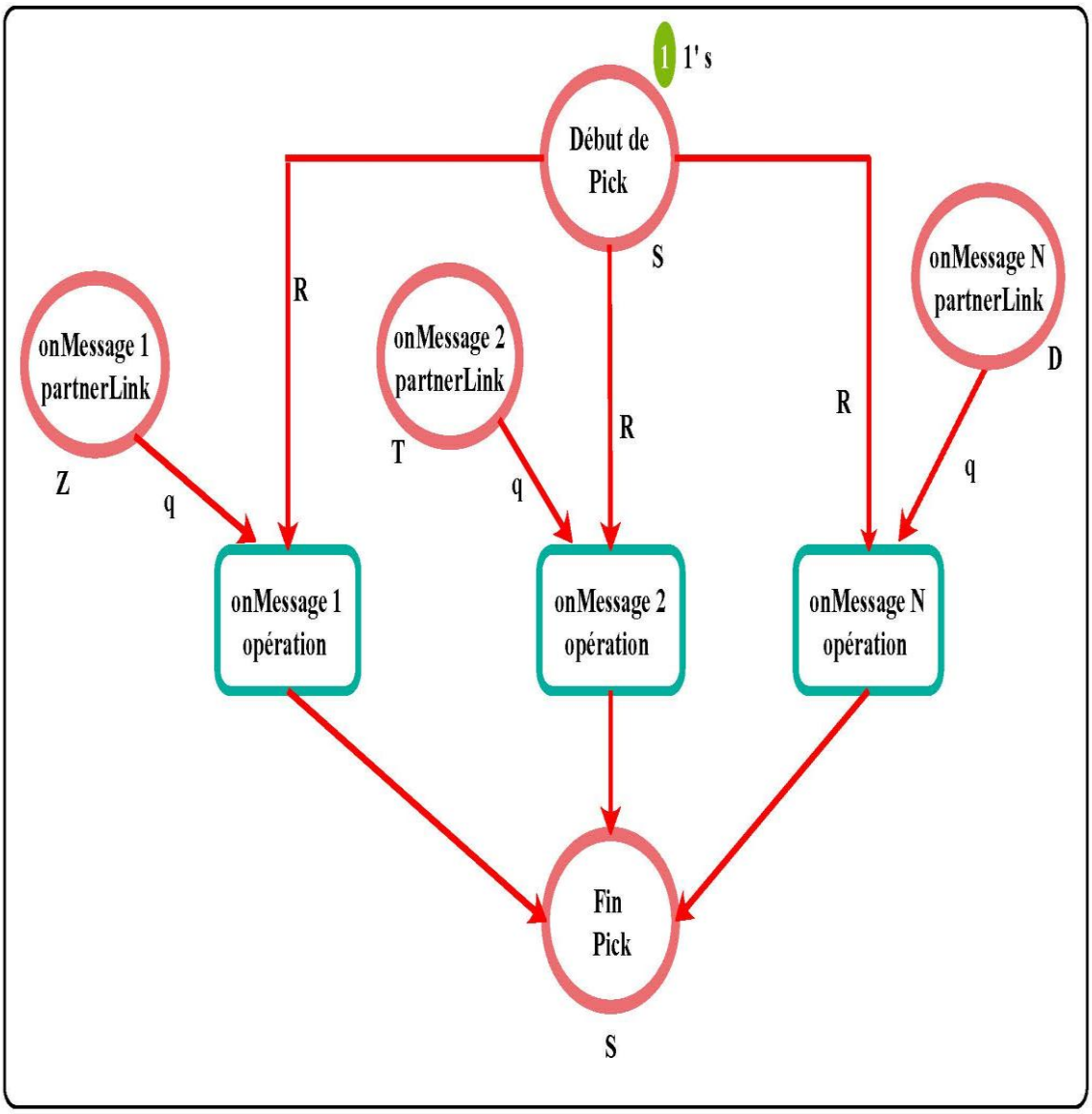


Figure 3.8 : Transformation l'activité pick en CP-net.

✓ **L'activité flow :** La construction <flow> de BPEL vous permet de spécifier une ou plusieurs activités à effectuer simultanément. Un flow se termine lorsque toutes les activités du flux sont terminées. La transformation se réfère à la Figure 3.9.

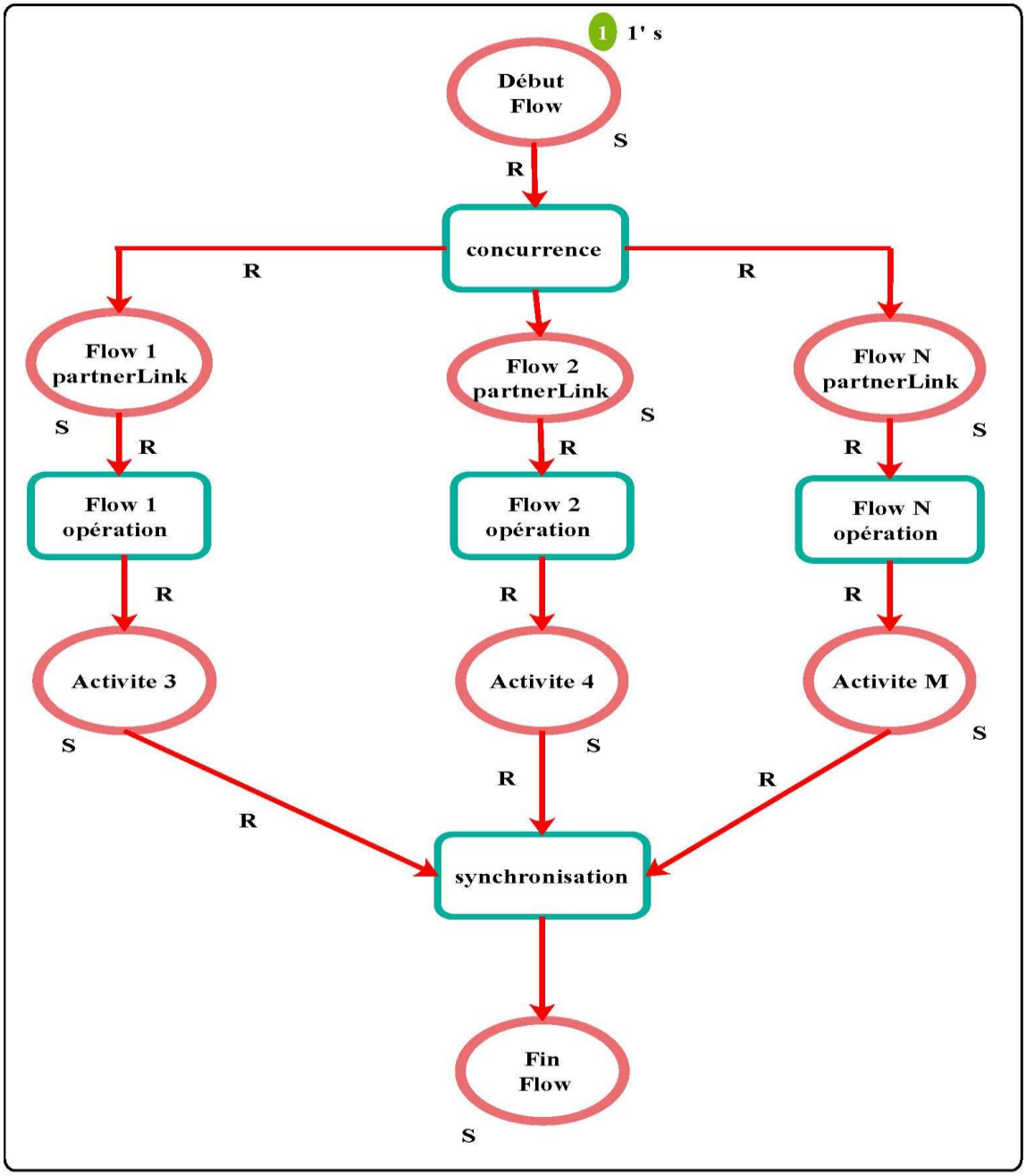


Figure 3.9 : Transformation l'activité flow en CP-net.

4. Architecture du système

L'objectif de notre projet est de réaliser un outil pour la modélisation formelle des services composites. Nous allons présenter tout d'abord la conception fonctionnelle globale du système (montrant ses fonctions majeures), ensuite nous passons à présenter le raffinement des différents modules du système.

4.1. Architecture globale du système

Notre système peut être vu comme un ensemble de modules reliés entre eux. La figure 3.10 présente l'architecture fonctionnelle du système.

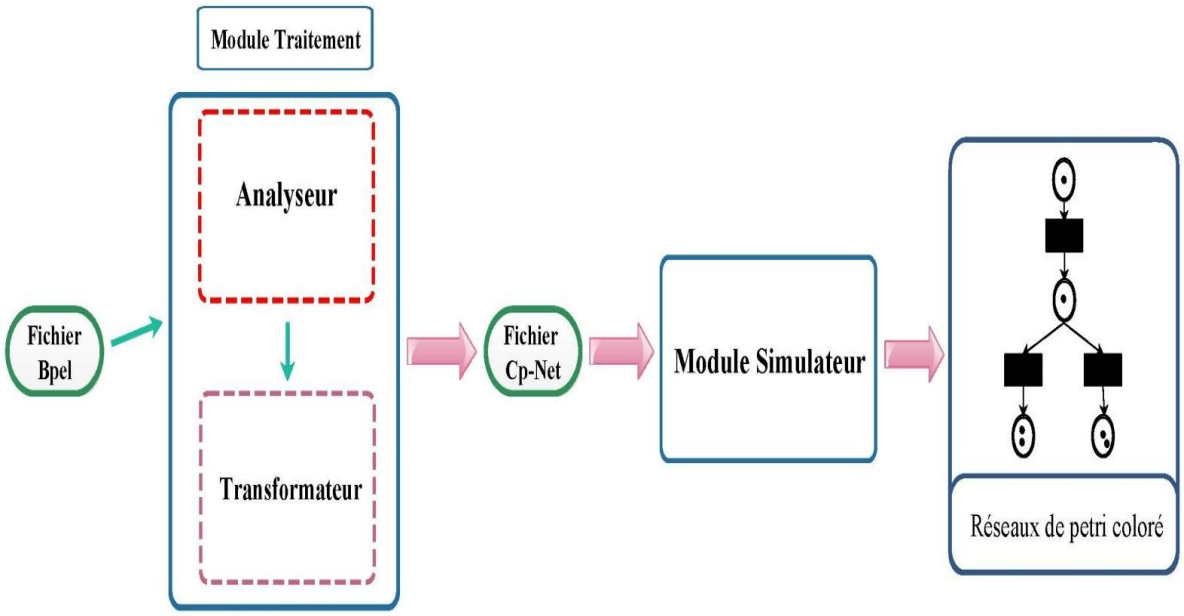


Figure 3.10 : Modèle fonctionnel du système.

Le système est composé principalement de Deux modules :

4.1.1. Module Traitement

Ce module reçoit un fichier BPEL qui décrit la composition des services Web et le convertit en réseaux de Pétri Colorés (CP-Net), qui sera stocké dans un autre fichier, cette transformation sera effectuée au niveau de deux sous modules au sein de module traitement :

4.1.1.1. Sous Module Analyseur

Ce module reçoit en entrée un fichier « bpel » qui est en réalité un fichier « XML » il contient un ensemble d'arcs triangulaires <> et une structure similaire à un arbre appelé une (Treelike Structure) et les balise du fichier XML sont appelées éléments XML. Un élément :

- ✓ Peut transporter des données textuelles.
- ✓ ou peut transporter d'autres éléments.
- ✓ ou peut combiner cela avec cela.
- ✓ ou peut être un élément vide.

Ce module analyse un fichier Bpel car il cherche, sépare et distingue les activités d'un fichier Bpel en fonction de leurs catégories en cherchant toutes les balises qui composent ce fichier Bpel: sachant que les balises qui constituent un fichier Bpel (input) sont: <receive>, <reply>, <invoke>, <empty>, <sequence>, <while>, <pick>, <switch>. Ces balises présentent les éléments de base d'un fichier bpel, mais il contient aussi d'autres éléments qui doivent être cherchés et leur extraction car ils présentent les places et les transitions et même les jetons et les types des messages échangés entre les places en plus des conditions de la transition et ces balises sont : <variable>, <case>, <onMessage> et le résultat de ce module (output) présente un ensemble des activités avec leurs relations.

❖ **Algorithme qui analyse fichier BPEL**

```
Analyse fichier BPEL () ;
Variable M, S, N, attributes: String;

Début
    Pour i ← 0 à dernier-ligne-de-fichier-BPEL
        Si (M =  = sequence) Alors
            N ← M ;
        Sinon
            Si (M = = receive) Alors
                Pour   j ← 0 à 6
                    N ← attributes.getName(j) + attributes.getValue(j) ;
                    J ←  Suivant ;
                FinPour
            Sinon
                Si (M = = invoke) Alors
                    Pour   j ← 0 à 6
                        N ← attributes.getName(j) + attributes.getValue(j) ;
                        j ← Suivant ;
                    FinPour
                Sinon
                    Si (M = = reply) Alors
                        Pour   j ← 0 à 6
                            N ← attributes.getName(j) + attributes.getValue(j) ;
                            j← Suivant ;
                        FinPour
```

```
Sinon
Si (M = = flow) Alors
    Pour   j ← 0 à 6
        N ← attributes.getName(j) + attributes.getValue(j) ;
        j← Suivant ;
    FinPour
Sinon
Si (M = = switch) Alors
    Pour   j ← 0 à 6
        N ← attributes.getName(j) + attributes.getValue(j) ;
        j ← Suivant ;
    FinPour
Sinon
Si (M = = case) Alors
    Pour   j ← 0 à 6
        N ← attributes.getName(j) + attributes.getValue(j) ;
        j ← Suivant ;
    FinPour
Sinon
Si (M = = pick) Alors
    Pour   j ← 0 à 6
        N ← attributes.getName(j) + attributes.getValue(j) ;
        j ← Suivant ;
    FinPour
Sinon
```

```
        Si (M = = onMessage) Alors
            Pour   j ← 0 à 6
                N ← attributes.getName(j) + attributes.getValue(j) ;
                j ← Suivant ;
            FinPour
        Sinon
        Si (M = = variable) Alors
            Pour   j ← 0 à 6
                N ← attributes.getName(j) + attributes.getValue(j) ;
                j← Suivant ;
            FinPour
        Finsi Finsi Finsi Finsi Finsi Finsi Finsi Finsi Finsi Finsi
    i ← Suivant ;
FinPour
FIN
```

4.1.1.2. Sous Module Transformateur

Ce module reçoit en entrée la sortie de *Module Analyseur* sous forme des activités qui présentent les propriétés générales de fichier Bpel et les convertit aux réseaux de Pétris colorés (CP-Net), en utilisant les règles suivantes :

- Toutes les balises : <receive>, <reply>, et <onMessage> contenues dans balise <pick>, présentent les attributs suivant : « partnerLink », « operation », « variable », mais ils varient en valeur des attribut en ce qui concerne « partnerLink » et « operation », et quant à « variable » ils varient en type de message « messageType », le tableau suivant présente comment convertir ces activités en réseaux Pétri colorés :

<receive> **<reply>** **<onMessage>**	« partnerLink »	Présente des places
	« operation »	présente les transitions
	« variable »	présente les types des jetons dans les places. et le jeton entrant dans la place.
	« messageType »	présente le jeton

Tab 3.3 : Conversion d'activité en réseaux Pétri colorés.

- La balise <invoke> composée des attributs suivant : « partnerLink » « operation », « inputVariable », «outputVariable », avec des valeurs variées des attributs « partnerLink » et « operation », quant à « inputVariable » et «outputVariable » ils varient en type de message « messageType », le tableau suivant présente comment convertir cette activité aux réseaux Pétri colorés.

<invoke>	« partnerLink »	présente des places
	« operation »	présente la transition.
	« inputVariable »	présente le type de jeton dans les places. et le jeton entrant dans la place.
	«outputVariable »	présente le type de jeton dans les places. et le jeton sortant dans la place suivante.
	« messageType »	présente le jeton

Tab 3.4 : la conversion d'activité <invoke> au réseau Pétri colorés.

- La balise <while> composée d'un autre type de balise : « condition » et ils varient dans la syntaxe « textuel » de ces attributs, ils présentent la condition de la transition appelée « Guard ».

<while>	« condition »	présente la condition de la transition « Guard ».

- La balise <switch> composé d'un autre type de balise : « case » et chaque balise contient un attribut « condition » mais ils varient en valeur d'attribut « case », ils présentent la condition de la transition appelée « Guard ».

< switch >	« case »	« condition »	présente la condition de la transition « Guard ».

- Après avoir appliqué ces règles au fichier Bpel en respectant l'ordre des activités, nous obtiendrons toutes les caractéristiques des réseaux Pétri colorés (CP-Net) : « place », « Transition », « jeton », « Guard » …. ect.

Ces résultats sont stockés dans un fichier appelé « CP-NET » que nous pouvons donc passer au logiciel « CPN Tools » pour tracer le RdPC.

❖ **Algorithme Transformateur du fichier bpel au réseaux pétri coloré**

```
Transformer fichierBPELtoCPN () ;
   Variable M, N, attributes: String;
Début
   Pour i ← 0 à dernier-ligne-de-fichier-BPEL

      Si (M = = partnerLink) Alors
            N ← M.getValue ;
            Ecrire (" Place ", N);
      Sinon
      Si (M = = messageType) Alors
            N ←  M.getValue ;
            Ecrire (" Jeton ", N) ;
      Sinon
      Si (M = = operation) Alors
            N ←  M.getValue ;
            Ecrire (" Transition ", N) ;
      Sinon
      Si (M = = partnerLink) Alors
            N ←  M.getValue ;
            Ecrire (" Guard ", N) ;
      Finsi Finsi Finsi
    i ←  Suivant ;
   FinPour
FIN
```

4.1.2. Module Simulateur

Dans ce module, nous allons connecter la plateforme « éclipse » avec le logiciel « CPN Tools » via Java afin de convertir un fichier de construction en « CP-NET » où les entrées présentent le fichier « CP-NET » que nous avons parlé précédemment et qui présente la sortie de *Module Traitement*.

Au début nous créons une opération de liaison qui se présente par un lien entre la plateforme de travail « éclipses » avec le logiciel « CPN Tools », et puis nous détaillerons comment lire le fichier « CP-NET » de la part de logiciel « CPN Tools ». Le fichier « CP-NET » présente un formulaire qui se compose de « Place », « Transition », « des Arcs » …. etc., Le résultat est un réseau de Pétri colorés (CP-Net), la figure 3.11 présente le *Module Simulateur*.

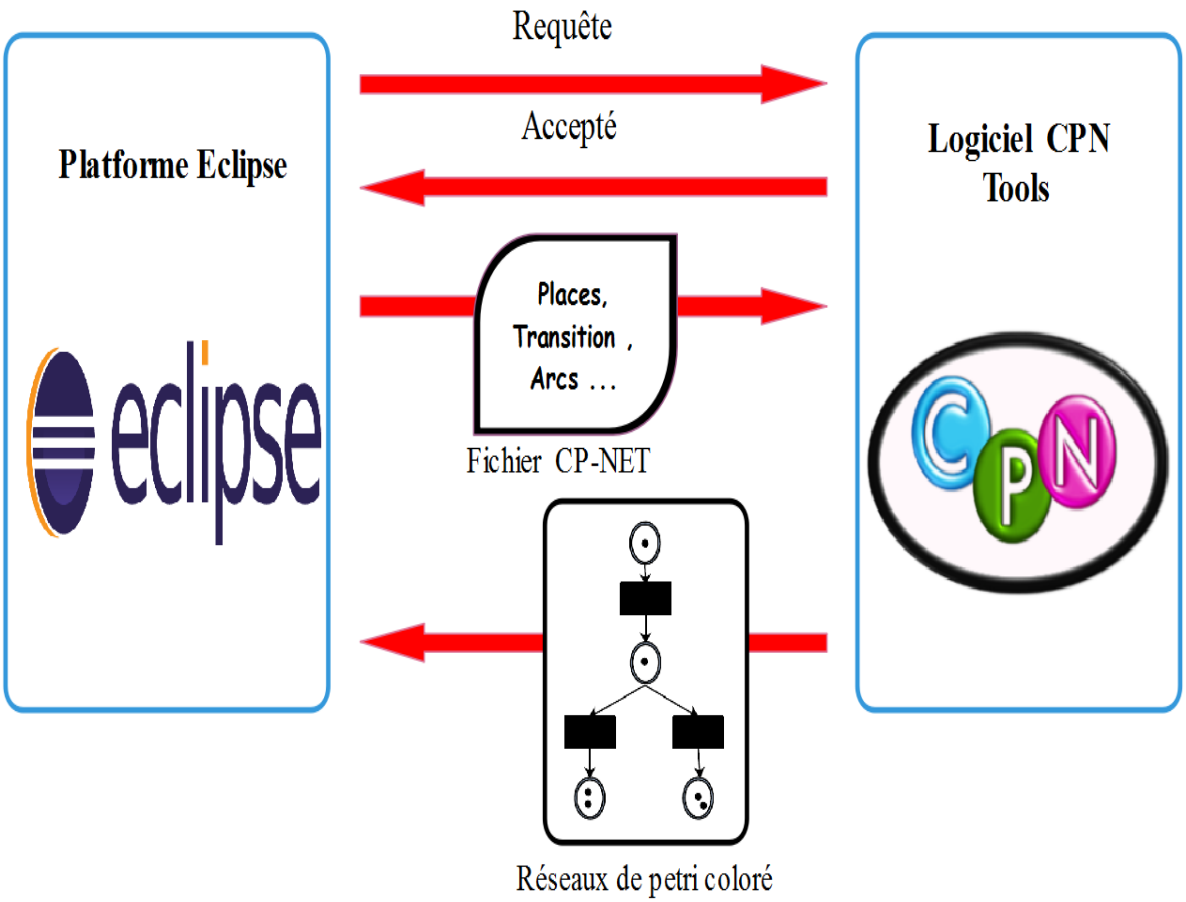


Figure 3.11 : la liaison entre CP-Net Tools et Eclipse.

Le résultat est une interface dans le logiciel « CPN Tools » qui présente un réseau de Pétri colorés (CP-Net), et il présente également le fonctionnement d'un service Web composite.

Après la création de réseau de Pétri colorés, nous pouvons vérifier le service web composite, comment il fonctionne, sa connexion à ses services, et toutes ses interactions avec ses services élémentaire à travers la vérification de RdPC qui présente l'output de logiciel « CPN Tools ». La figure 3.12 montre en détail le logiciel CP-net Tools :

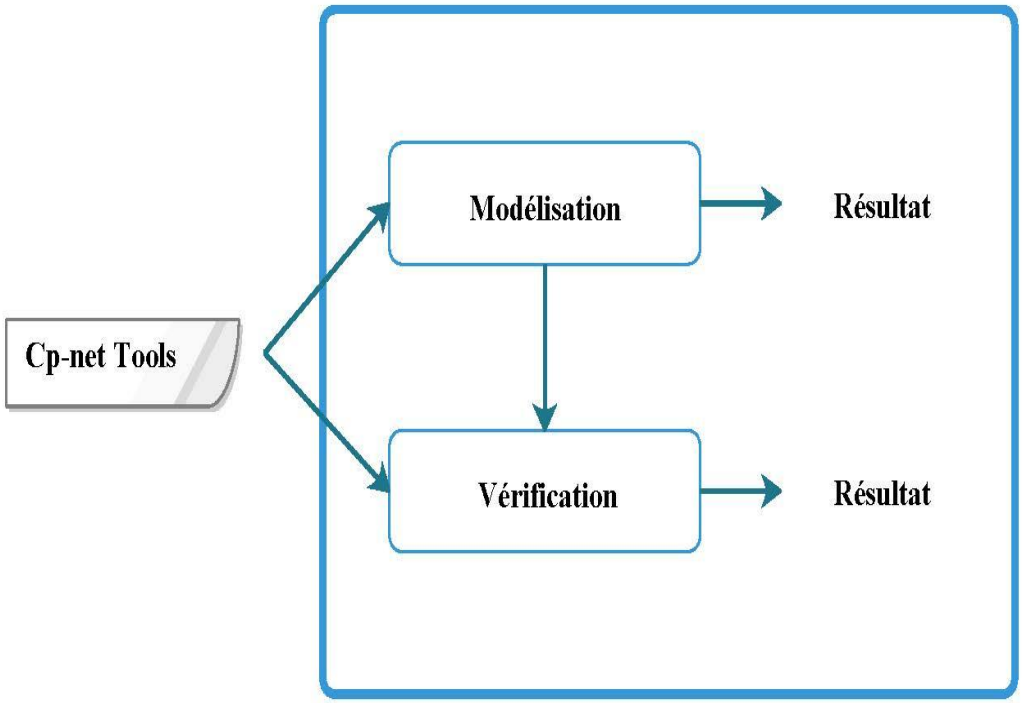


Figure 3.12 : Modèle fonctionnel du logiciel CP-net Tools.

4.2. Architecture globale de l'outil détaille

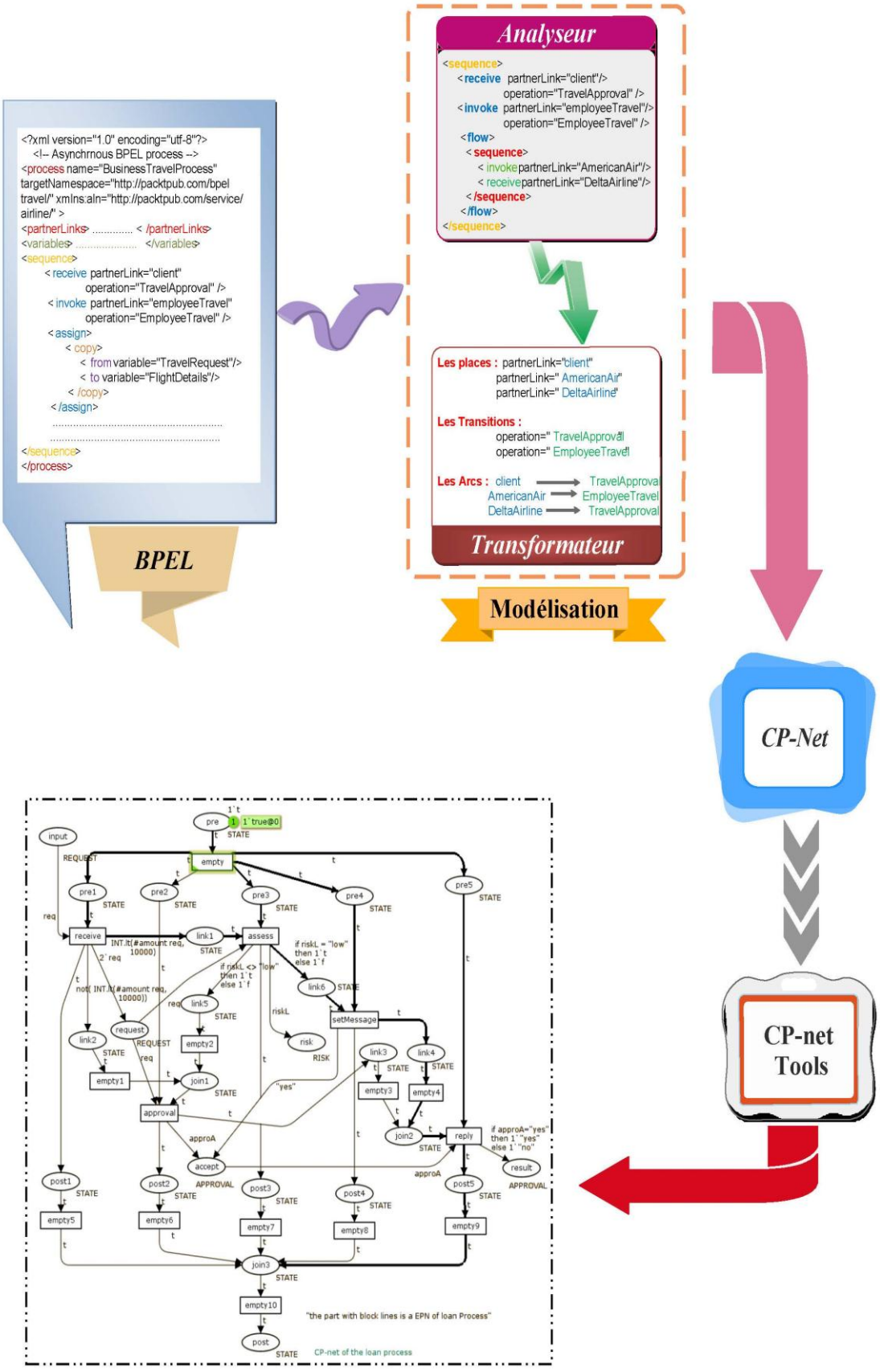


Figure 3.13 : Modèle fonctionnel du système détaillé.

5. Conclusion :

Nous avons essayé au long de ce chapitre de présenter l'architecture générale du système de modélisation formelle d'un service composite spécifié par BPEL avec un réseau de Petri coloré sans oublier de détailler les différents modules constituants cette architecture. Dans le chapitre suivant nous allons détailler l'implémentation, puis quelques résultats expérimentaux.

Chapitre 4

« Implémentation »

1. Introduction

L'objectif de ce chapitre d'implémentation est la réalisation de notre conception qui est bien détaillée dans le chapitre précédent en commençant par la description de l'environnement de développement, le langage de programmation et les outils de développement. La tâche principale de notre système est de transformer un fichier BPEL qui présente un Service Web composite en un réseau de Pétri coloré, sachant que nous allons présenter la simulation graphique du RdPC en utilisant le logiciel « CPN Tools ».

Pour bien illustrer l'implémentation de notre projet nous allons décrire les différentes interfaces de notre système et présenter des tests variés et exhaustifs décrits par des fichiers BPEL et leurs RdPC.

2. Environnement de développement utilisé

Pour réaliser notre système, nous avons utilisé le langage de programmation java, et quelques outils de développement. Nous allons décrire brièvement notre environnement de développement dans les sous sections suivantes :

2.1. Langage de programmation Java

Le langage Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java [80].

Dans notre implémentation nous allons utiliser *Eclipse* qui présente un environnement intégré de développement (IDE) pour le langage Java (et d'autres langages), il peut être téléchargé gratuitement sur le site www.eclipse.org. Il peut être amélioré et modifié en utilisant de nombreux plug-ins [81]. La licence fournie avec Eclipse est une licence open-source un peu particulière appelée EPL (Eclipse Public License).

2.2. CPN Tools : La modélisation et la validation des modèles des réseaux de Pétri colorés sont supportées par CPN Tools [82]:

- ✓ Modification et vérification de la syntaxe
- ✓ Simulation interactive et automatique.
- ✓ Exploration et vérification de l'espace d'état.
- ✓ Analyse de performance.
- ✓ Visualisation comportementale à l'aide de graphiques de domaine d'application.

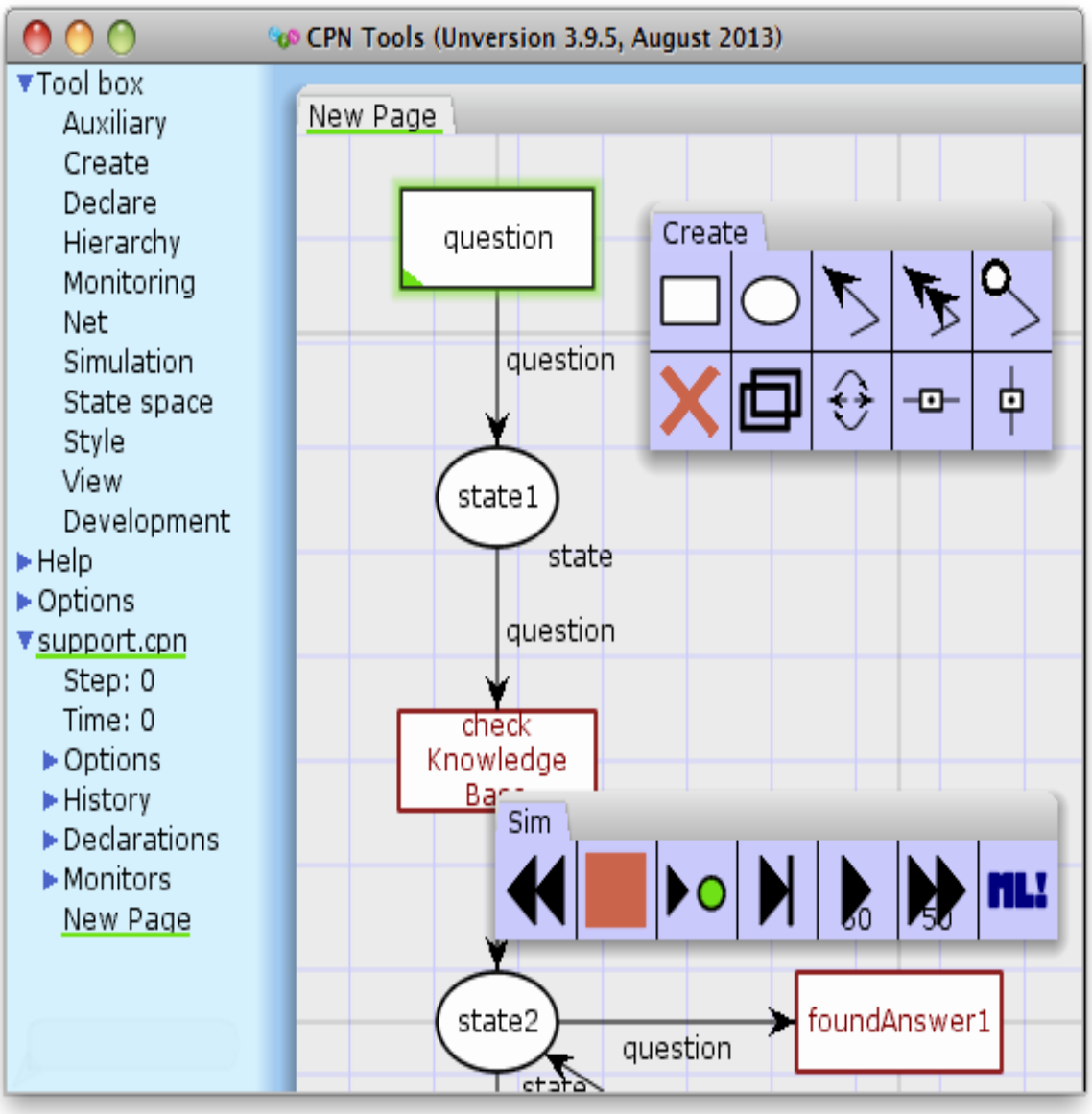


Figure 4.1 : l'interface CPN Tools.

J'ai aussi utilisé un programme de dessin qui m'a aidé à dessiner toutes les Figures de ma note « Edraw Max ».

2.3. EdrawMax : est un logiciel polyvalent de conception de diagrammes, avec des caractéristiques qui le rendent parfait non seulement pour éditer des diagrammes de flux dans un style très professionnel, des organigrammes, des diagrammes et graphiques des ventes, mais aussi pour réaliser des diagrammes réseaux, des plans de construction, des cartes heuristiques (mind maps), des flux de données, des diagrammes de conceptions, des diagrammes UML, des diagrammes d'ingénierie en électricité, des illustrations scientifiques. [83]

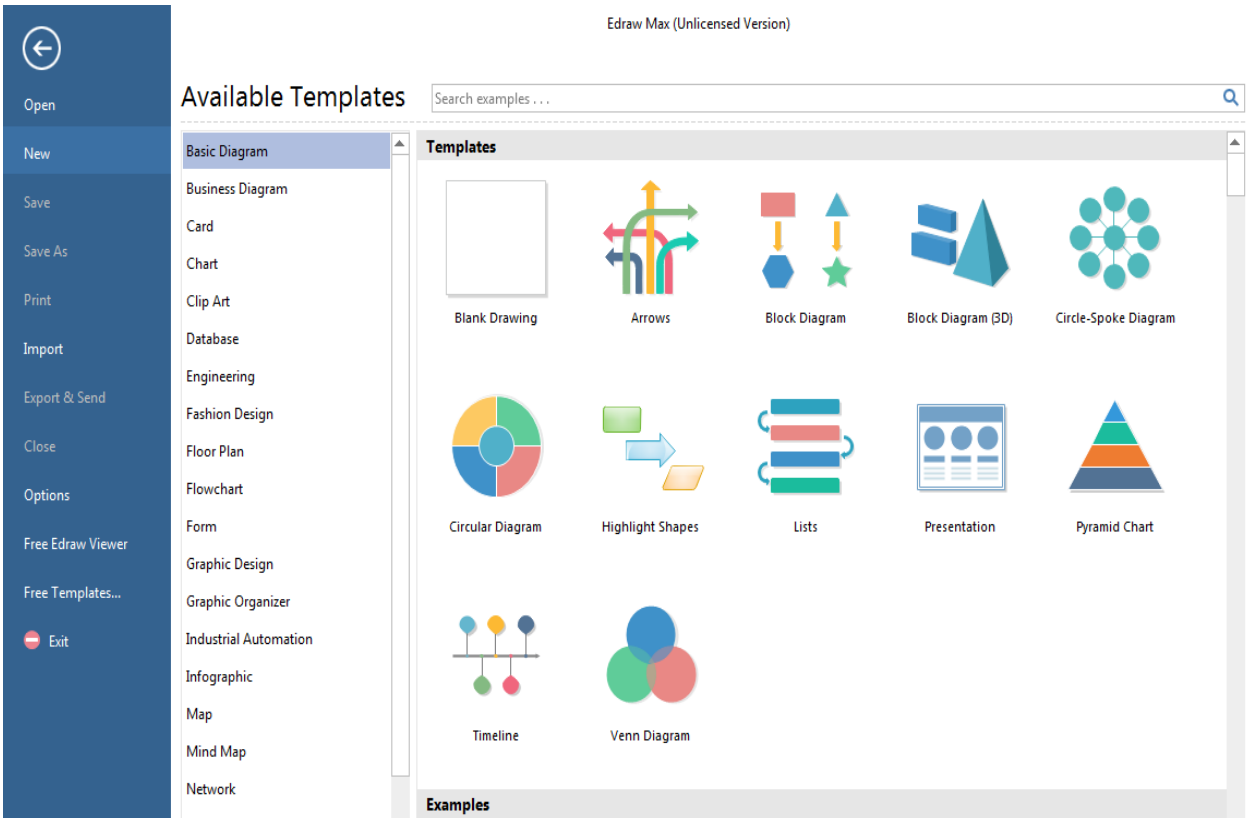


Figure 4.2 : l'interface EdrawMax.

3. Réalisation du système

Comme nous avons mentionné précédemment, l'objectif de notre système est de modéliser formellement un service composite spécifié par un fichier BPEL en utilisant les RdPC . L'implémentation de notre système sera fait par le langage java à travers Eclipse et les RdPC vont être simuler et tracer en utilisant l'outil « CPN Tools ».

3.1. Les interfaces du système

- ***Interface principale du système***

Dans l'interface principale, il y a un ensemble de boutons. Nous commençons avec le bouton « Ouvrir », il permet d'ouvrir un fichier. Un bouton « Save » pour enregistrer le résultat et le bouton « Cp-Net» fonctionne dans un autre fichier. Le bouton « Cp-Net Tools » connecte la plate-forme de travail à « éclipse » et au programme « Cp-Net Tools » pour échanger le fichier précédemment enregistré. Le bouton « Clear » permet de formater toutes les étapes mentionnées ci-dessus afin de travailler sur un nouveau fichier et le bouton « Close » ferme le système.

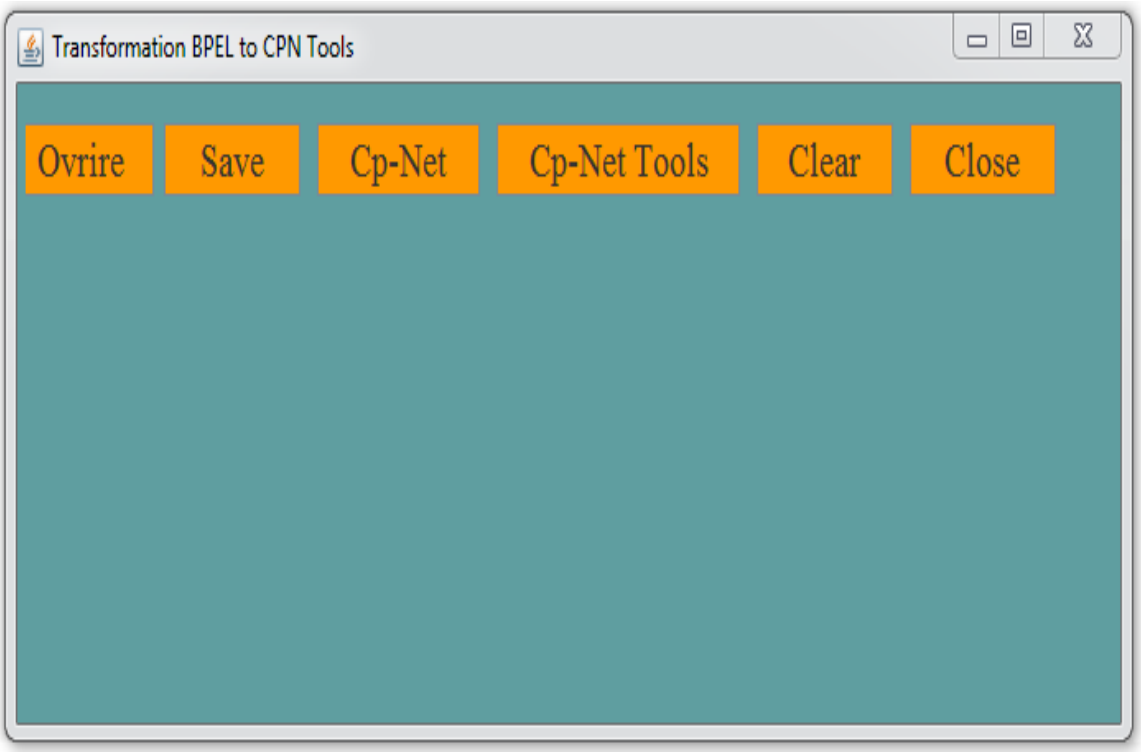


Figure 4.3 : Interface général de system.

- ***Description détaillée de l'interface principale***

Comme nous l'avons dit, l'interface de base consiste en un ensemble de boutons et chaque bouton a sa propre fonction :

- ✓ *Bouton « Ouvrir » :* L'utilisateur clique sur le bouton « Ouvrir » où il cherche un fichier BPEL avec un format « .bpel »
- ✓ *Bouton « Save » :* L'utilisateur enregistre le résultat du bouton « Cp-Net » dans le fichier « CPN » avec un format « .cpn »
- ✓ *Bouton « Cp-Net » :*
  1. Avant de sélectionner le fichier via le bouton « Ouvrir », l'utilisateur doit appuyer sur le bouton « Cp-Net », nous constatons que tous les champs sont vides, ce qui représente les place et les transitions et les arcs ….. etc. (figure 4.4).

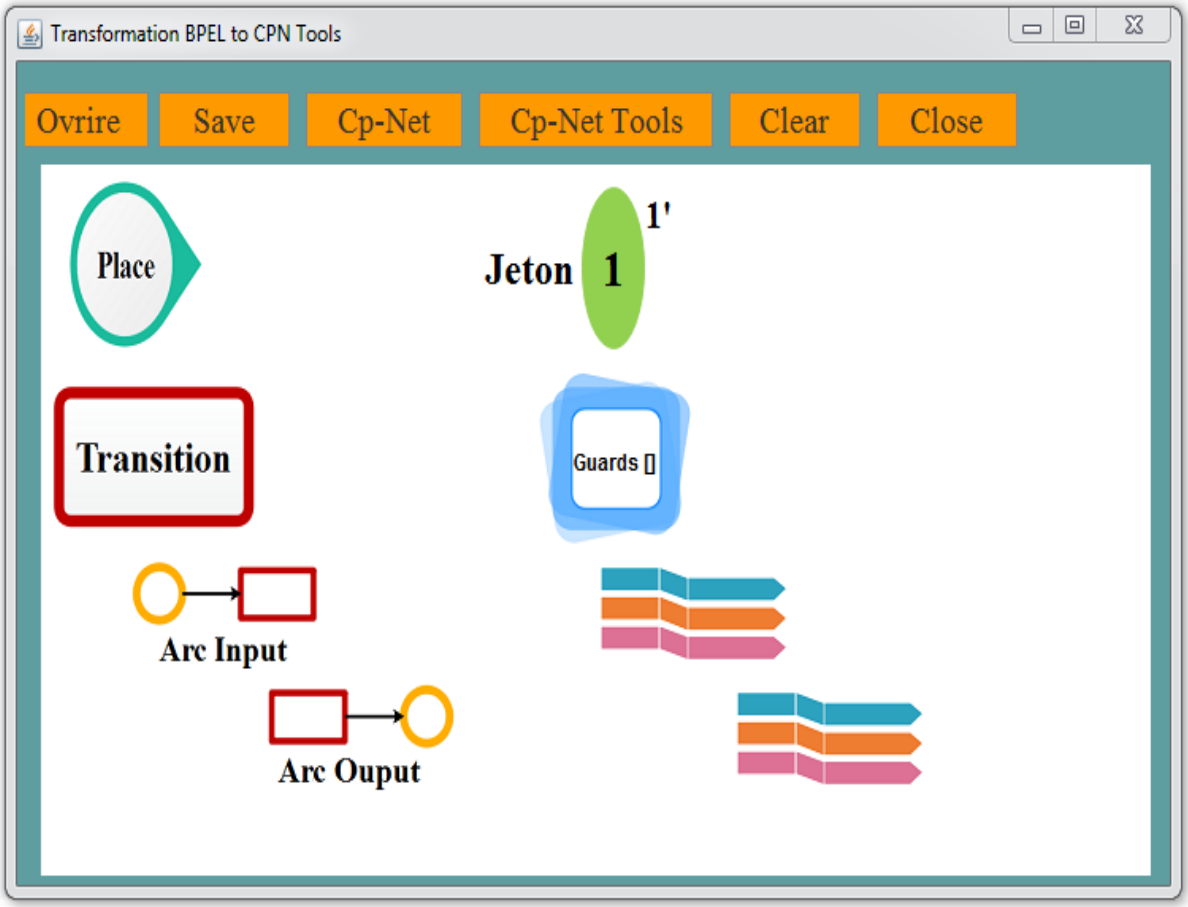


Figure 4.4 : Interface de résultat de Bouton « Cp-Net » vide.

2. Lorsque l'utilisateur sélectionne le fichier et ré-appuye sur le bouton « Cp-Net », tous les champs vont être remplis (figure 4.5).

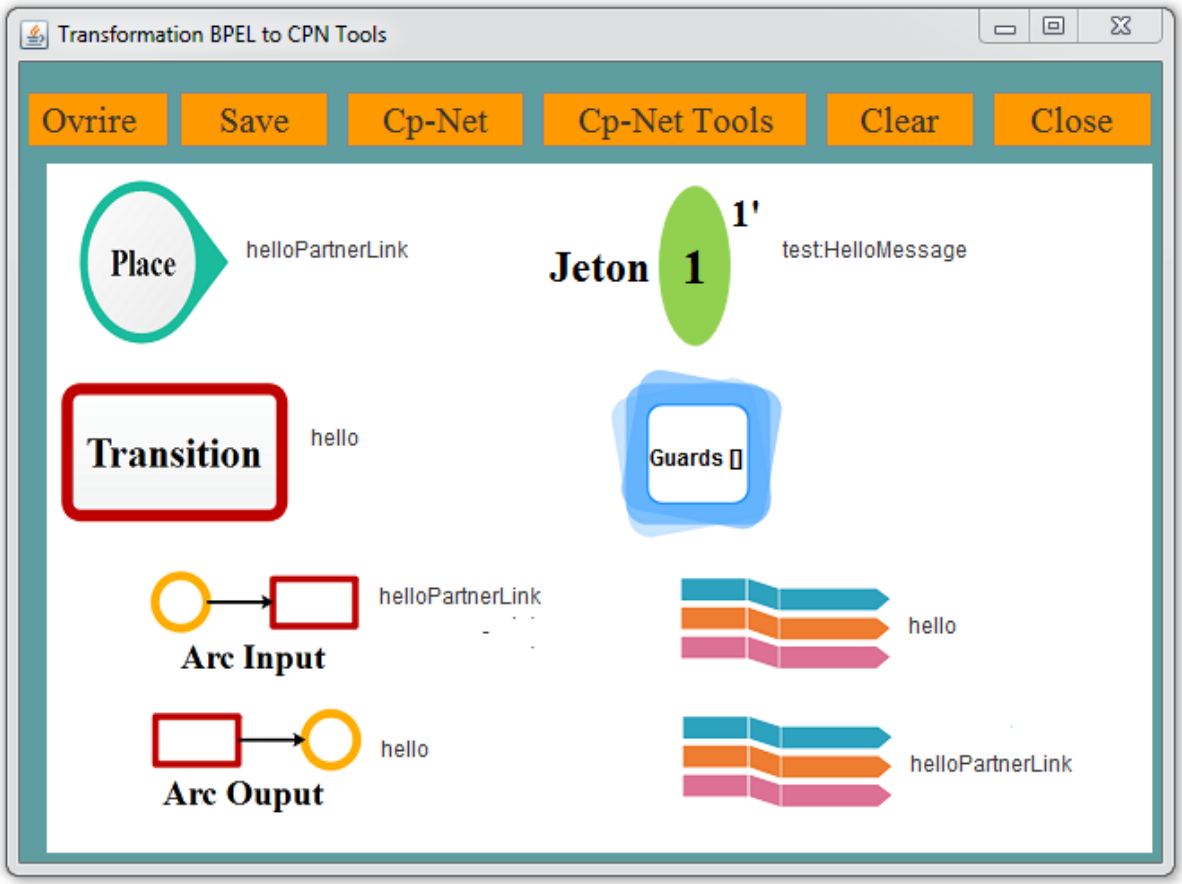


Figure 4.5 : Interface de résultat de Bouton « Cp-Net » mais remplir.

✓ *Bouton « Cp-Net Tools » :* Ce bouton connecte la plate-forme « Eclipse » et la plateforme de modélisation « Cp-Net Tools » selon les étapes suivantes :

1. L'exécution du serveur et la recherche des extensions dans « Eclipse », telles que la classe et l'interface et les enregistrez dans un serveur Comme illustré dans Figure 4.6.

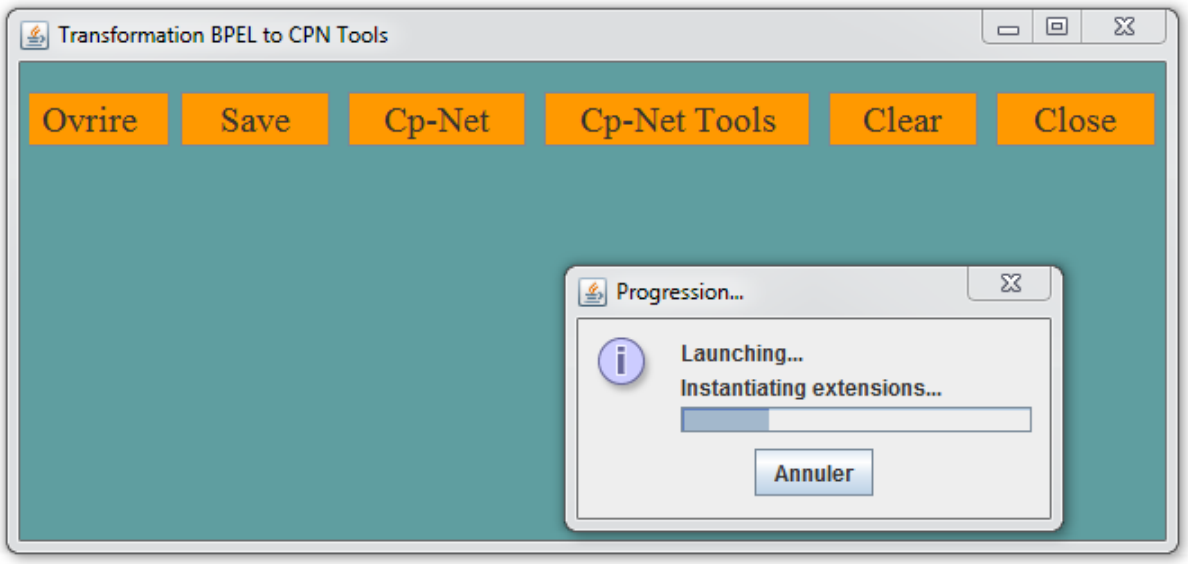


Figure 4.6 : Interface de résultat de Bouton « Cp-Net Tools » qui lancer l'extension.

2. Après l'exécution du serveur et l'installation des extensions, il résulte l'interface suivante qui représente le serveur et ses extensions :

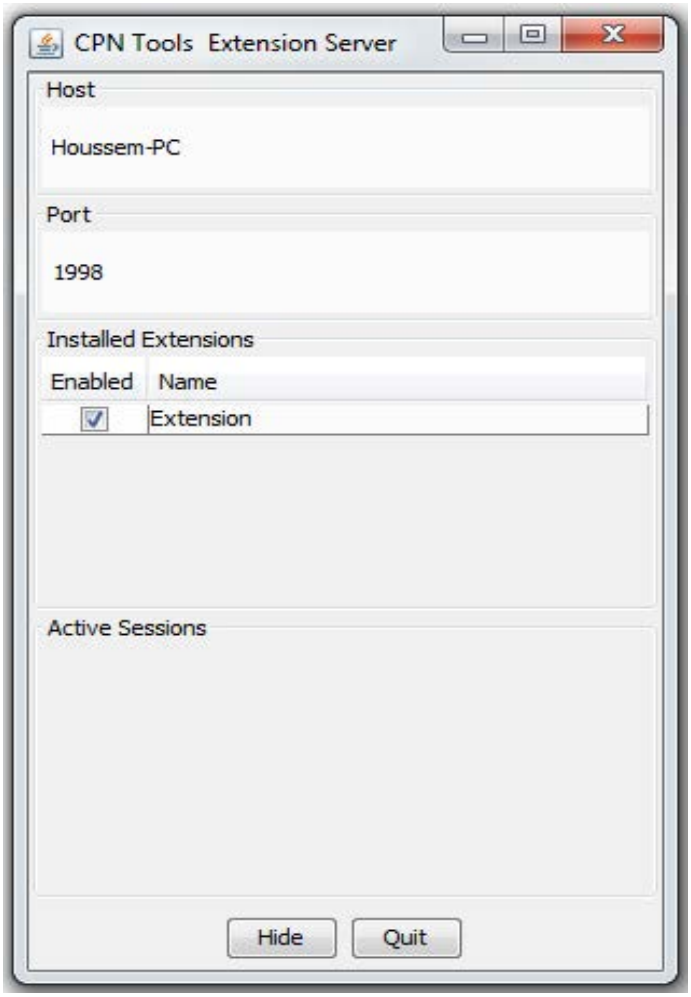


Figure 4.7 : Interface de résultat de Bouton « Cp-Net Tools » qui marche l'serveur.

3. Ici, le serveur envoie la demande de connexion provenant d'Eclipse au programme « CPN Tools » pour l'échange d'informations (Figure 4.8).

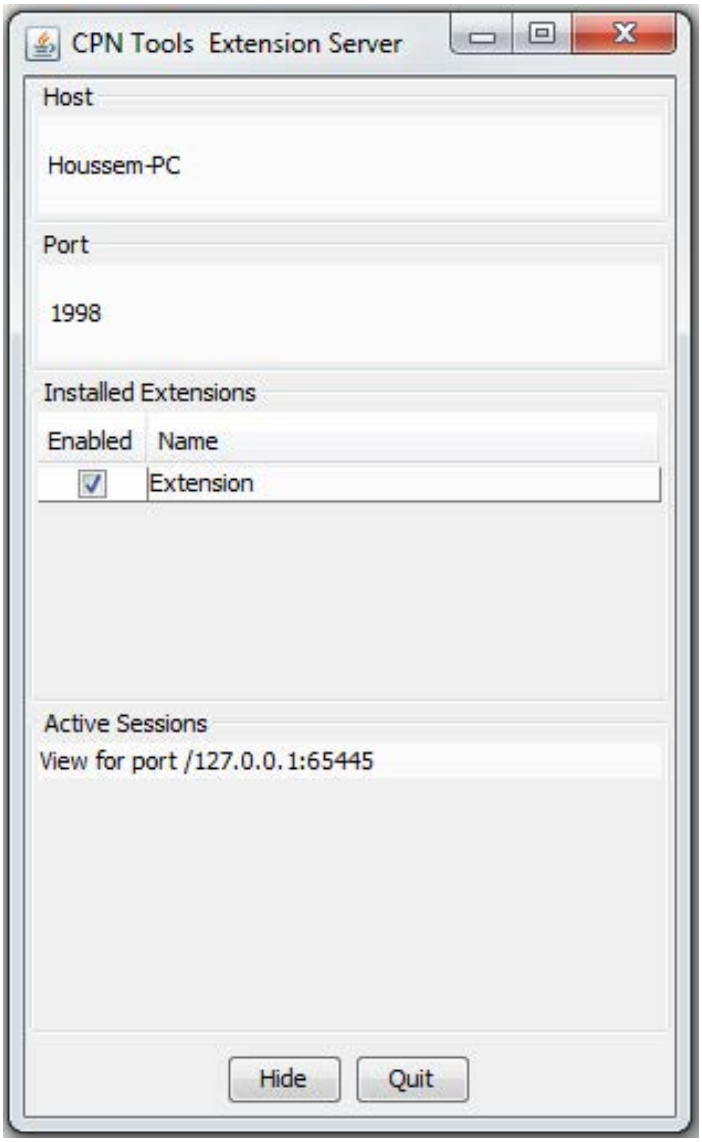


Figure 4.8 : Interface qui accepté la connexion entre « Eclipse » et « Cp-net Tools ».

Et lorsque nous ouvrons l'interface de programme « CPN Tools », Nous sortons un outil « Create » situé dans les outils « toolBox » Situé dans le programme « CPN Tools », un nouvel outil apparaîtra dans les outils « Create », ce nouvel outil s'appelle « CPN ». (Figure 4.9)

Figure 4.9 : Interface de outil « Create » dans les outils « toolBox ».

4. Ensuite nous cliquons sur cet outil et on le fait glisser vers le modèle obtenu dans le programme « CPN Tools », nous verrons une autre interface Java qui contient deux interface, le premier interface qui définit qui a un bouton « Ouvrir » qui permet d'ouvrir le fichier précédemment enregistré, qui est le fichier « CPN » avec un format « .cpn » (Figure 4.10) , et le deuxième interface qui définit des places et des transitions, Et aussi des entrées et sorties pour chacune des places et des transitions et propriétés des transitions et des types de jeton (Figure 4.11).

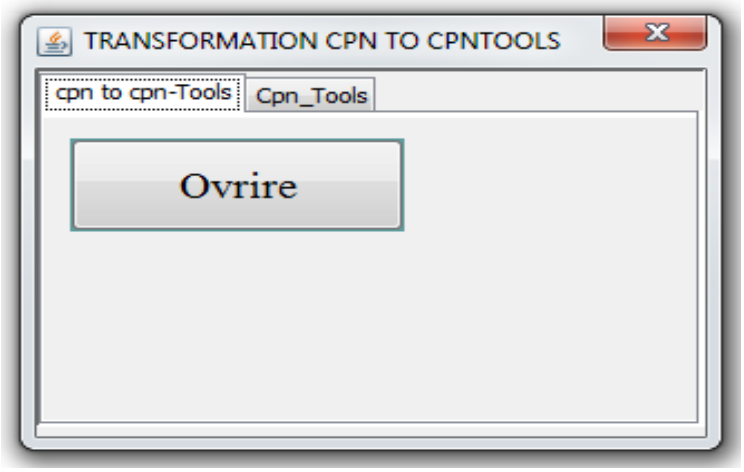


Figure 4.10 : Interface de bouton « Ouvrir »

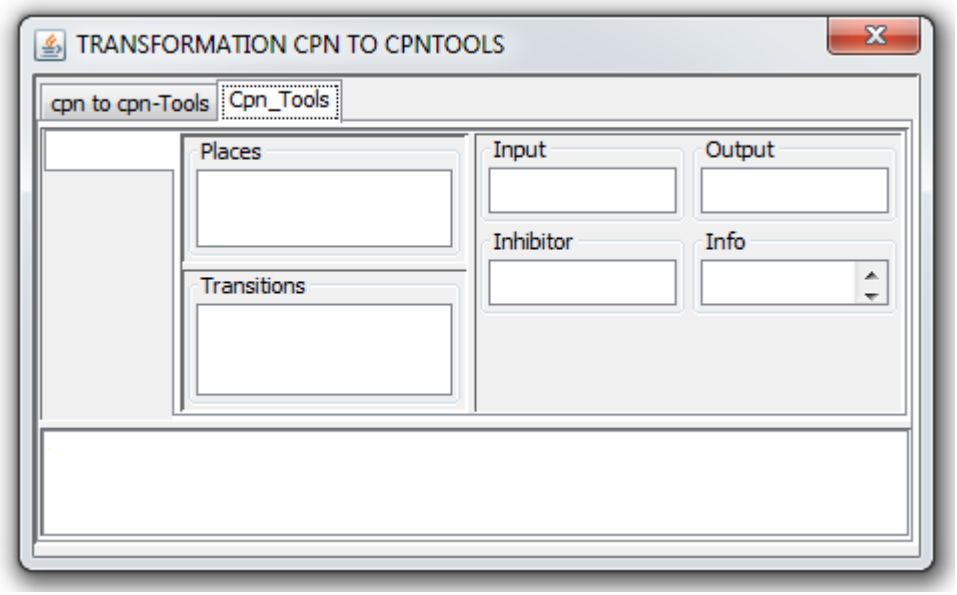


Figure 4.11 : Interface de Java qui définir des places et des transitions dans programme « CPN Tools ».

5. Après avoir sélectionné le fichier à partir du bouton « Ouvrir ». Une 'interface sera affichée dans le programme « CPN Tools » contenant des dessins graphiques représentent les places et les transitions et les arcs…etc., résultant du fichier « CPN » Comme dans la figure suivante :

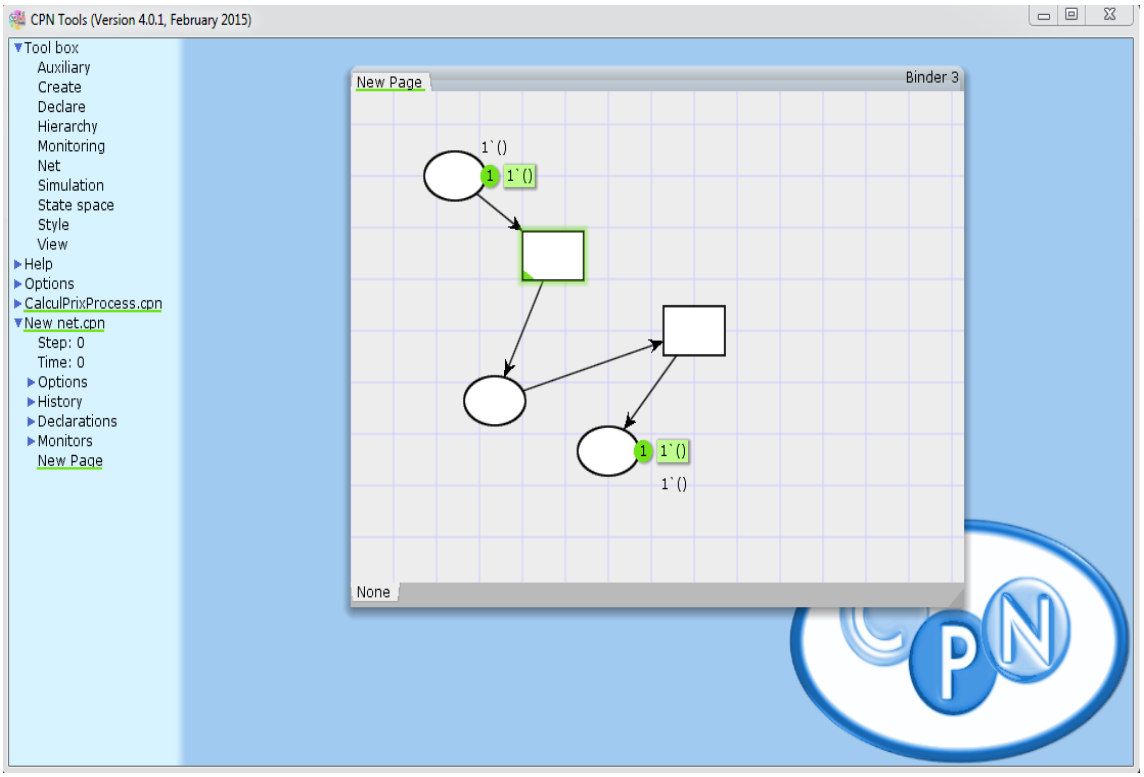


Figure 4.12 : Interface dans programme « CPN Tools ».

4. Test des résultats

4.1. Le service composite Travel

Le premier exemple de processus BPEL que nous allons modéliser est un processus permettant l'achat, en ligne, de billets d'avion pour des voyages d'affaire [84].

4.1.1. Description de scénario

Un client cherche à acheter un billet d'avion, il envoie un message à un employé donné. Le message reçu inclut le nom de l'employé concerné, la destination, la date de départ ainsi que la date d'arrivée. Après cette réception, le processus cherche avant tout à vérifier le statut de l'employé. Pour ce faire, le processus fait appel au service Web « Employee Travet Status ». Le processus procède à la consultation du prix du billet d'avion auprès des deux compagnies aériennes: *American Airlines* et *Delta Airlines*. Cette consultation se fait à travers deux services Web dont chacun est offert par une des deux compagnies, à savoir le « American Airlines web service» et le « Delta Airlines web service ». L'appel à ces deux services web se fait d'une façon concurrente. En dernier lieu, le processus choisit le prix le moins cher entre ceux fournis par les deux compagnies pour finalement retourner le plan de voyage au client.

Une représentation graphique du processus Travel est fournie. Cette représentation correspond à un diagramme d'activité du processus pour indiquer les opérations BPEL définies par ledit processus.

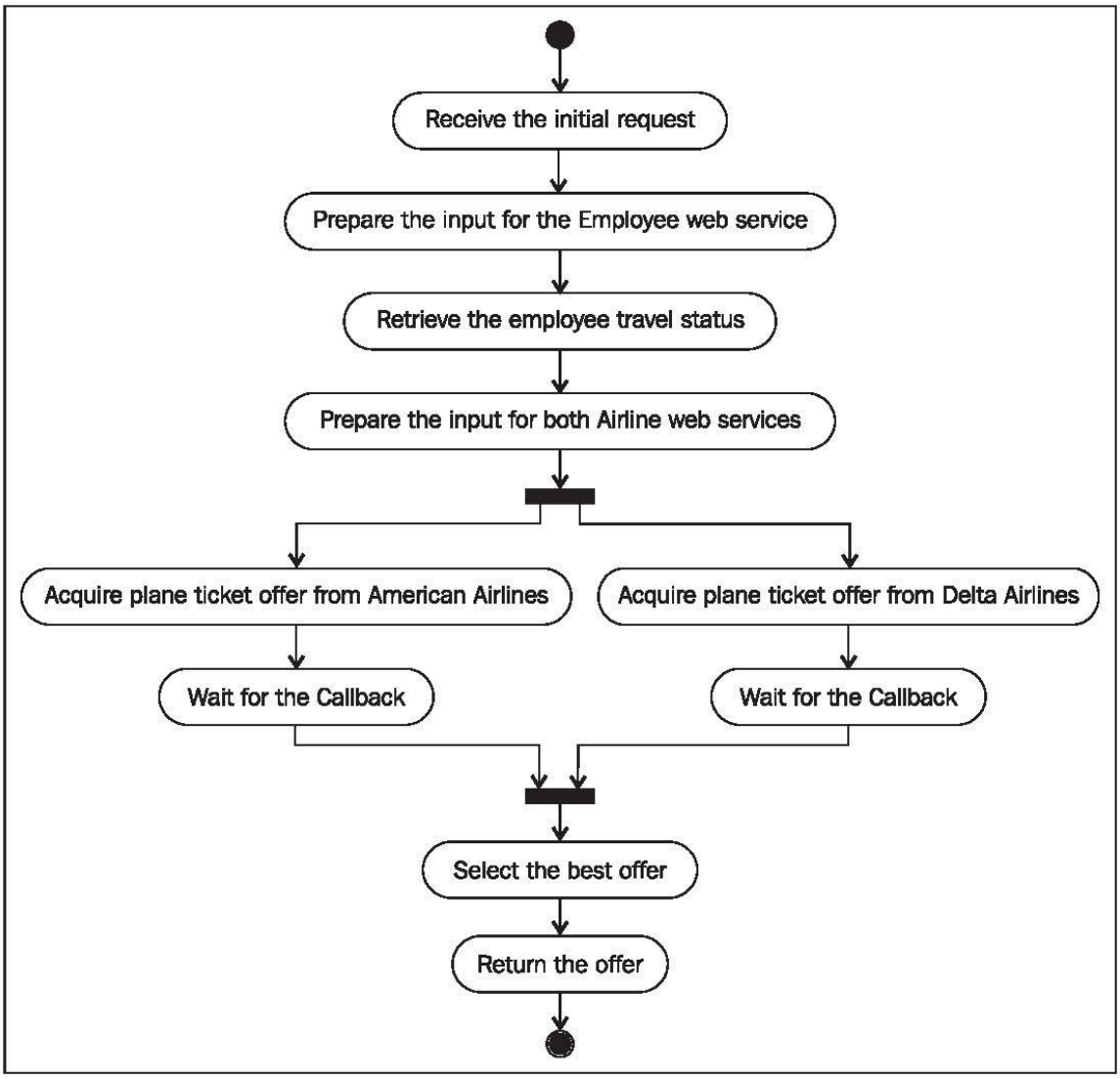


Figure 4.13 : La représentation graphique du processus Travel [84].

4.1.2. La spécification BPEL de processus TRAVEL

```
<process name="BusinessTravelProcess"
         targetNamespace="http://packtpub.com/bpel/travel/"
         xmlns="http://schemas.xmlsoap.org/ws/2003/03/business
         process/"
         xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/
         business- process/"
         xmlns:trv="http://packtpub.com/bpel/travel/"
         xmlns:emp="http://packtpub.com/service/employee/"
         xmlns:aln="http://packtpub.com/service/airline/" >

   <partnerLinks>
      <partnerLink name="client"
                   partnerLinkType="trv:travelLT"
                   myRole="travelService"
                   partnerRole="travelServiceCustomer"/>

      <partnerLink name="employeeTravelStatus"
                   partnerLinkType="emp:employeeLT"
                   partnerRole="employeeTravelStatusService"/>

      <partnerLink name="AmericanAirlines"
                   partnerLinkType="aln:flightLT"
                   myRole="airlineCustomer"
                   partnerRole="airlineService"/>

      <partnerLink name="DeltaAirlines"
                   partnerLinkType="aln:flightLT"
                   myRole="airlineCustomer"
                   partnerRole="airlineService"/>

   </partnerLinks>

   <variables>
      <variable name="TravelRequest"
                messageType="trv:TravelRequestMessage"/>
      <variable name="EmployeeTravelStatusRequest"

messageType="emp:EmployeeTravelStatusRequestMessage"/>
      <variable name="EmployeeTravelStatusResponse"

messageType="emp:EmployeeTravelStatusResponseMessage"/>
      <variable name="FlightDetails"
                messageType="aln:FlightTicketRequestMessage"/>
      <variable name="FlightResponseAA"
                messageType="aln:TravelResponseMessage"/>
      <variable name="FlightResponseDA"
                messageType="aln:TravelResponseMessage"/>
      <variable name="TravelResponse"
                messageType="aln:TravelResponseMessage"/>
   </variables>
```

```
 <sequence>

            <receive partnerLink="client"
               portType="trv:TravelApprovalPT"
               operation="TravelApproval"
               variable="TravelRequest"
               createInstance="yes" />

            <assign>
        <copy>
          <from variable="TravelRequest" part="employee"/>
          <to variable="EmployeeTravelStatusRequest"
part="employee"/>
        </copy>
      </assign>

      <invoke partnerLink="employeeTravelStatus"
              portType="emp:EmployeeTravelStatusPT"
              operation="EmployeeTravelStatus"
              inputVariable="EmployeeTravelStatusRequest"
              outputVariable="EmployeeTravelStatusResponse" />

      <assign>
        <copy>
          <from variable="TravelRequest" part="flightData"/>
          <to variable="FlightDetails" part="flightData"/>
        </copy>
        <copy>
          <from variable="EmployeeTravelStatusResponse"
part="travelClass"/>
          <to variable="FlightDetails" part="travelClass"/>
        </copy>
      </assign>

      <flow>

        <sequence>

          <invoke partnerLink="AmericanAirlines"
                  portType="aln:FlightAvailabilityPT"
                  operation="FlightAvailability"
                  inputVariable="FlightDetails" />

          <receive partnerLink="AmericanAirlines"
                   portType="aln:FlightCallbackPT"
                   operation="FlightTicketCallback"
                   variable="FlightResponseAA" />

        </sequence>
```

```
  <sequence>

          <invoke partnerLink="DeltaAirlines"
                  portType="aln:FlightAvailabilityPT"
                  operation="FlightAvailability"
                  inputVariable="FlightDetails" />

          <receive partnerLink="DeltaAirlines"
                   portType="aln:FlightCallbackPT"
                   operation="FlightTicketCallback"
                   variable="FlightResponseDA" />

   </sequence>

      </flow>

      <switch>

        <case
condition="bpws:getVariableData('FlightResponseAA','confirmati
onData','/confirmationData/aln:Price')
                        &lt;=
bpws:getVariableData('FlightResponseDA','confirmationData','/c
onfirmationData/aln:Price')">

           <assign>
             <copy>
               <from variable="FlightResponseAA" />
               <to variable="TravelResponse" />
             </copy>
           </assign>
        </case>

        <otherwise>
           <assign>
             <copy>
               <from variable="FlightResponseDA" />
               <to variable="TravelResponse" />
             </copy>
           </assign>
        </otherwise>
      </switch>

      <invoke partnerLink="client"
              portType="trv:ClientCallbackPT"
              operation="ClientCallback"
              inputVariable="TravelResponse" />
   </sequence>

</process>
```

4.1.3. Transformation au réseaux de pétri colore

Nous allons transformer un fichier BPEL «Travel » en réseaux de Pétri colorés en suivant les étapes suivantes :

1. Appuyez sur le bouton « Ouvrir » et recherchez un fichier « Travel.bpel » Puis en appuyant sur le bouton « Cp-Net » Où la recherche des noms des places et des transitions et comment se connecter avec certains (les arcs) et le type de jeton … etc., et nous voyons le résultat suivant dans ce figure :

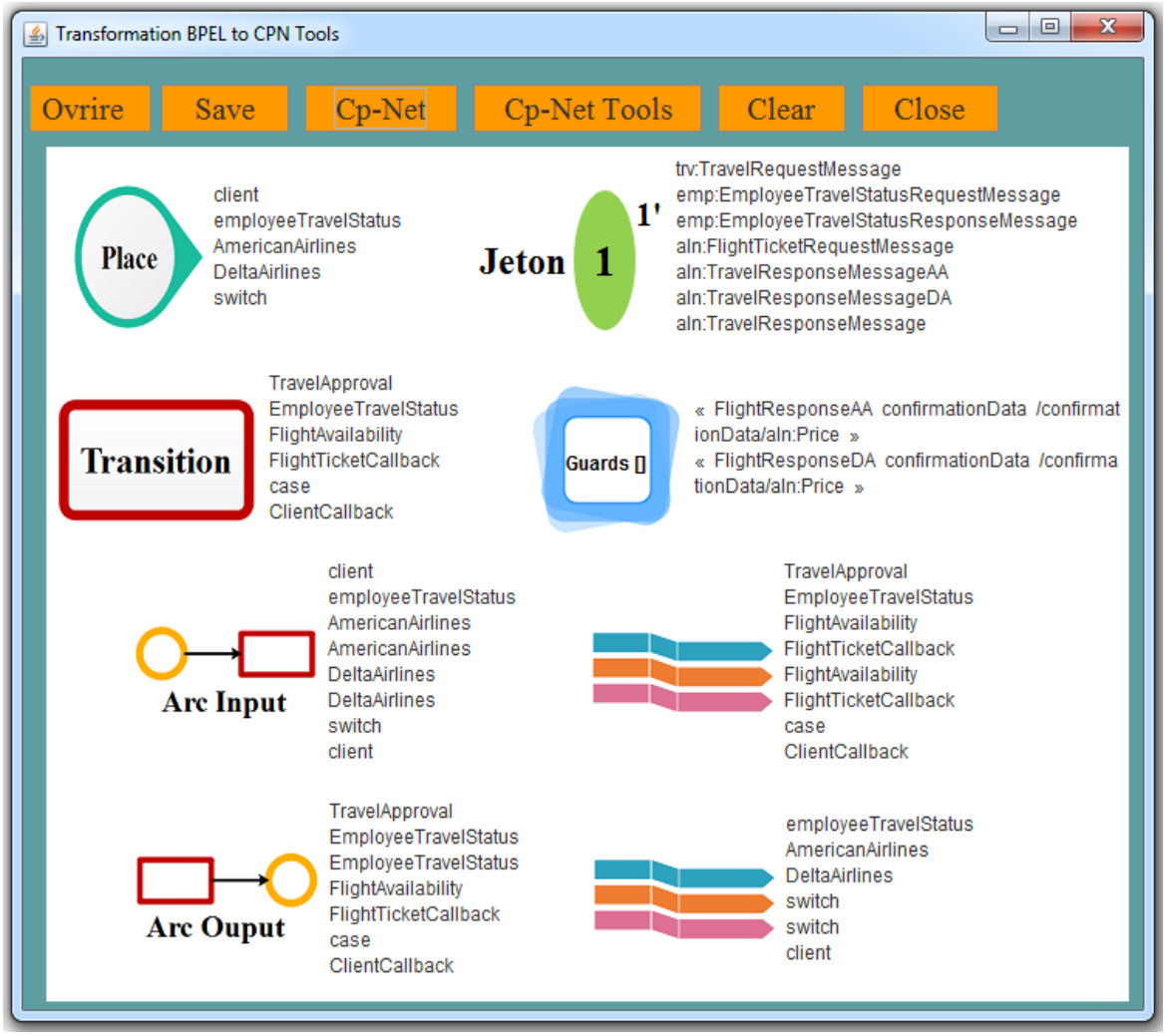
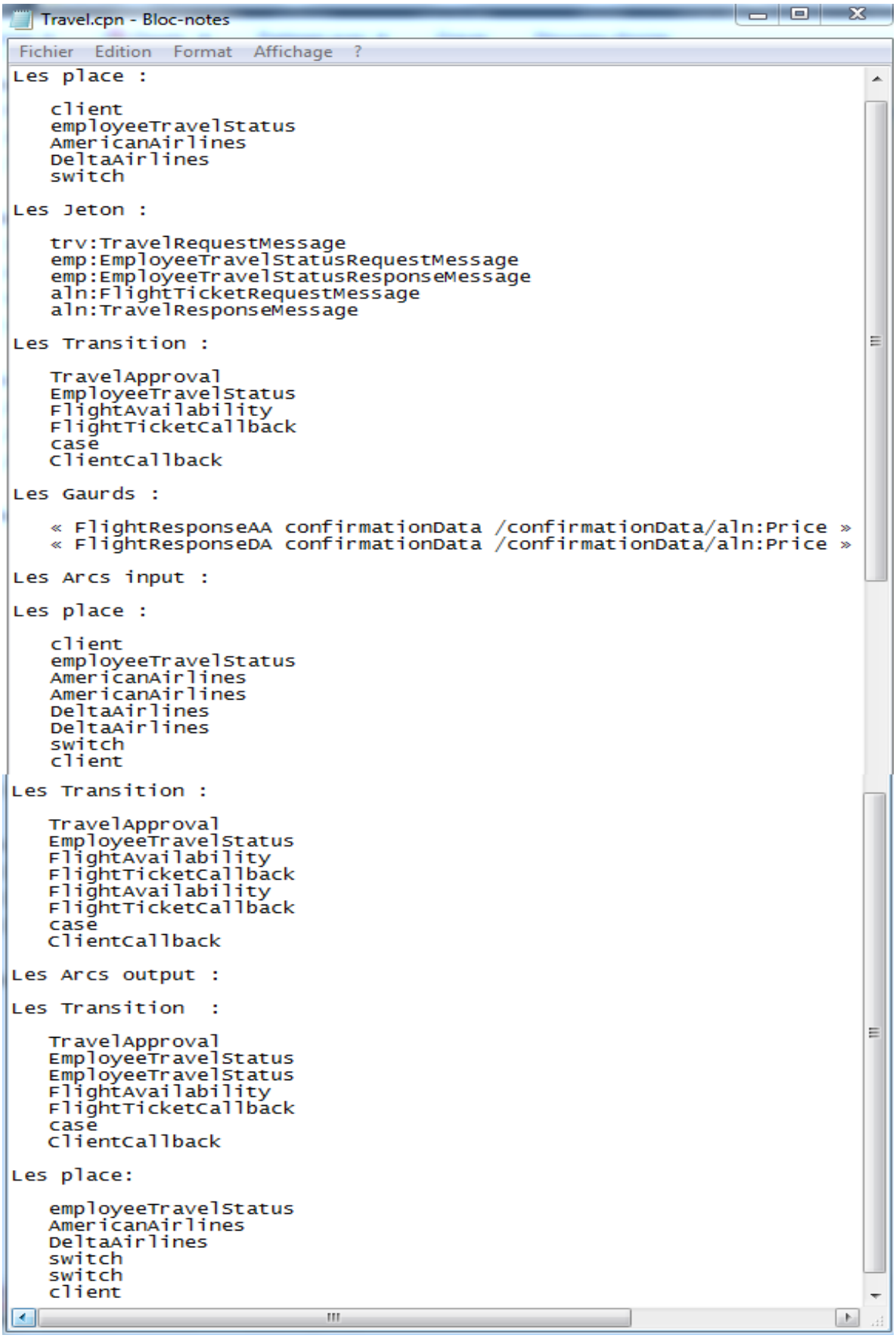


Figure 4.14 : Interface de résultat de Bouton « Cp-Net » fichier BPEL «Travel ».

2. Nous appuyons ensuite sur le bouton « Save » pour enregistrer les résultats obtenus dans le fichier « CPN » avec un format « .cpn » où le contenu du fichier est détaillé dans la figure 4.15 :



Travel.cpn - Bloc-notes

Fichier Edition Format Affichage ?

```
Les place :

   client
   employeeTravelStatus
   AmericanAirlines
   DeltaAirlines
   switch

Les Jeton :

   trv:TravelRequestMessage
   emp:EmployeeTravelStatusRequestMessage
   emp:EmployeeTravelStatusResponseMessage
   aln:FlightTicketRequestMessage
   aln:TravelResponseMessage

Les Transition :

   TravelApproval
   EmployeeTravelStatus
   FlightAvailability
   FlightTicketCallback
   case
   ClientCallback

Les Gaurds :

   « FlightResponseAA confirmationData /confirmationData/aln:Price »
   « FlightResponseDA confirmationData /confirmationData/aln:Price »

Les Arcs input :

Les place :

   client
   employeeTravelStatus
   AmericanAirlines
   AmericanAirlines
   DeltaAirlines
   DeltaAirlines
   switch
   client

Les Transition :

   TravelApproval
   EmployeeTravelStatus
   FlightAvailability
   FlightTicketCallback
   FlightAvailability
   FlightTicketCallback
   case
   ClientCallback

Les Arcs output :

Les Transition  :

   TravelApproval
   EmployeeTravelStatus
   EmployeeTravelStatus
   FlightAvailability
   FlightTicketCallback
   case
   ClientCallback

Les place:

   employeeTravelStatus
   AmericanAirlines
   DeltaAirlines
   switch
   switch
   client
```

Figure 4.15 : Interface de fichier enregistrer « Travel.cpn ».

✓ Nous appuyons sur le bouton « Cp-Net Tools » La plate-forme de travail « Eclipse » est connectée au programme de modélisation « CPN Tools » comme nous l'avons déjà dit , Après avoir connecté le programme « CPN Tools » avec « Eclipse » une autre interface Java apparaîtra Contenant le bouton « Ouvrir » ,Ensuite, nous allons sélectionner le fichier « CPN » précédemment enregistré, Nous obtenons une interface dans le programme « CPN Tools » qui contient une modélisation formelle du fichier BPEL « Travel » présenté par un réseaux de pétri colorés (figure 4.16) :

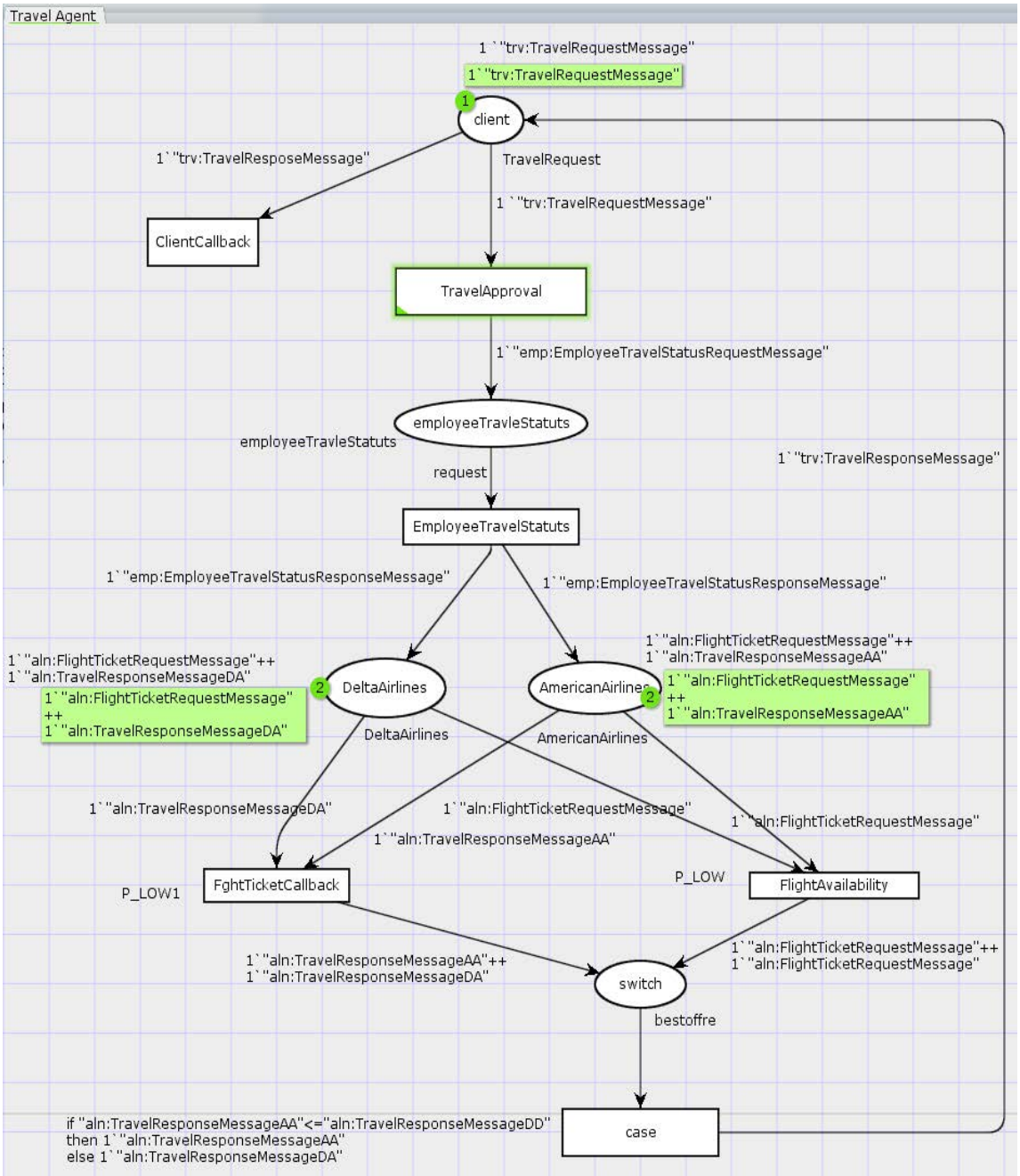


Figure 4.16 : Interface de modélisation formelle du fichier BPEL « Travel » dans programme « CPN Tools ».

3. nous verrons une autre interface Java qui définit des places et des transitions … etc. situé dans nouvel outil s'appellent « CPN » (Figure 4.17).

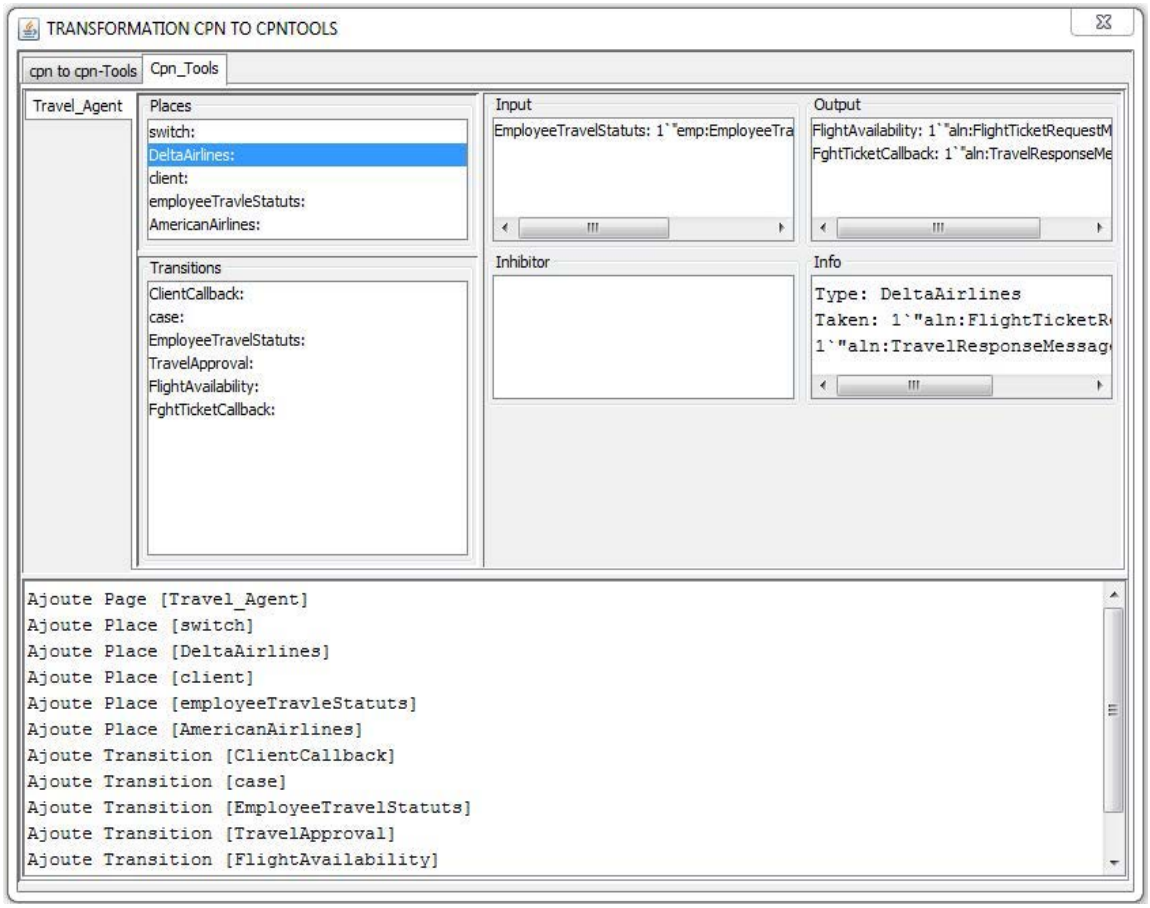


Figure 4.17 : Interface Java qui définit des places et des transitions du fichier BPEL « Travel » dans « CPN Tools ».

4.2. Le service composite Travel Agency

L'agence de voyage est un exemple académique classique pour expliquer les principes SOA. Cet exemple est une implémentation d'un service qu'une agence de voyages peut fournir à ses clients. Le processus BPEL permet de réserver un vol, une voiture et un hôtel pour un client. Ces réservations sont faites simultanément aux trois partenaires de l'agence de voyages [85].

4.2.1. Description de scénario

Le processus Travel Agency correspond à un processus d'orchestration de 3 partenaires. Le client doit indiquer son nom et son adresse, ses dates et lieux de départ et d'arrivée. Le processus renvoie au client un message indiquant que la réservation a été effectuée avec succès et que ses identifiants de réservation ont été perdus, soit la réservation de l'un des partenaires a échoué.

Ces partenaires sont un service de réservation de billets d'avion et un service de location de voitures mis en œuvre par des Services Web synchrones in-out, et un service de réservation d'hôtels mis en œuvre par un service asynchrone. (Figure 4.18) :

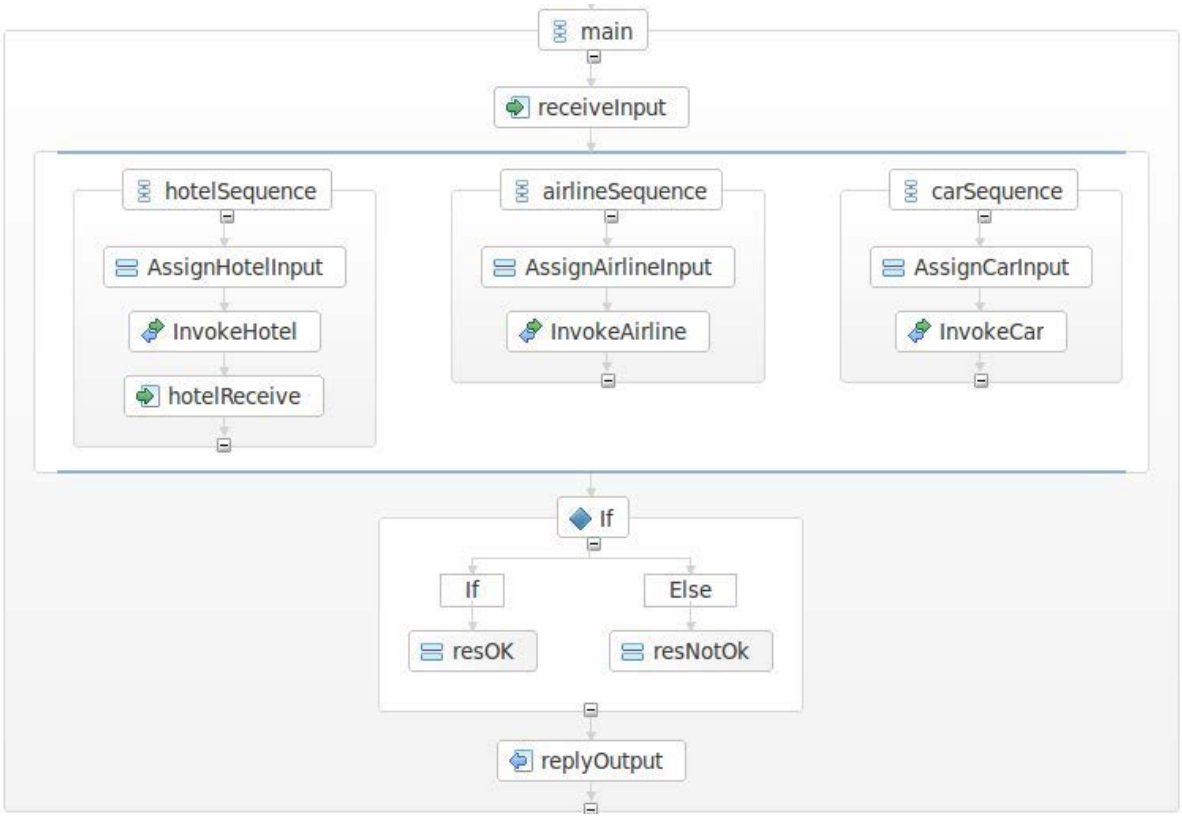


Figure 4.18 : La représentation graphique du processus Travel Agency. [85]

4.2.2. La spécification BPEL de processus Travel Agency

```
<bpel:process name="travelagency"
targetNamespace="http://org.ow2.petals/demo/travelagency/agency/"
     xmlns:hotc="http://www.example.org/processHotelConfirmation/
"
     suppressJoinFailure="yes"
     xmlns:tns="http://org.ow2.petals/demo/travelagency/agency/"
     xmlns="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
     xmlns:bpel="http://docs.oasis-
open.org/wsbpel/2.0/process/executable"
     xmlns:ns0="http://org.ow2.petals/demo/travelagency/hotel/"
     xmlns:car="http://org.ow2.petals/demo/travelagency/car/"
     xmlns:ns2="http://www.w3.org/2001/XMLSchema">
<bpel:partnerLinks>
     <bpel: partnerLink name="client"
                       partnerLinkType="tns:travelAgency"
                       myRole="travelAgencyProvider" />
     <bpel:partnerLink name="airlinepartner"
                       partnerLinkType="ns1:airlineLT"
                       partnerRole="airlineRole">
     </bpel:partnerLink>
     <bpel:partnerLink name="hotelpartner"
                       partnerLinkType="ns1:hotelLT"
            partnerRole="hotelRole">
     </bpel:partnerLink>
     <bpel:partnerLink name="carpartner"
                       partnerLinkType="ns1:carLT"
                       partnerRole="carRole">
     </bpel:partnerLink>
</bpel:partnerLinks>

<bpel:variables>
     <bpel:variable name="input"
                    messageType="tns:travelAgencyRequestMessage"
/>
     <bpel:variable name="output"
                    messageType="tns:travelAgencyResponseMessage"
/>
     <bpel:variable name="airlinepartnerResponse"
                    messageType="ns:bookFlightResponse" />
     <bpel:variable name="airlinepartnerRequest"
                    messageType="ns:bookFlightRequest" />
     <bpel:variable name="hotelpartnerRequest"
                    messageType="ns0:bookRoomRequest" />
     <bpel:variable name="hotelpartnerResponse"
                    messageType="ns0:bookRoomResponse" />
     <bpel:variable name="carpartnerResponse"
                    messageType="car:bookCarResponse" />
     <bpel:variable name="carpartnerRequest"
                    messageType="car:bookCarRequest" />
     </bpel:variables>
```

```
<bpel:sequence name="main">
     <bpel:receive  name="receiveInput"
                    partnerLink="client"
                    portType="tns:travelAgency"
                    operation="bookTravel" variable="input"
                    createInstance="yes" />

<bpel:flow name="Flow">
        <bpel:sequence name='hotelSequence'>
         <bpel:assign validate="no" name="AssignHotelInput">
          <bpel:copy>
<bpel:from><![CDAT[$input.payload/tns:lastname]]>
           </bpel:from> ……..        </bpel:copy>
          </bpel:assign>

<bpel:invoke  name="InvokeHotel"
              partnerLink="hotelpartner"
               operation="bookRoom"
              inputVariable="hotelpartnerRequest"
              outputVariable="hotelpartnerResponse"
              portType="ns0:hotel">
</bpel:invoke>
</bpel:sequence>
<bpel:sequence name="carSequence">
           <bpel:assign validate="no" name="AssignCarInput">
           <bpel:copy>
           <bpel:from> <![CDATA[$input.payload/tns:lastname]]>
           </bpel:from>…….</bpel:copy>
</bpel:assign>

            <bpel:invoke name="InvokeCar"
                         partnerLink="carpartner"
                         operation="bookCar"
                         portType="car:car"
                         inputVariable="carpartnerRequest"
                         outputVariable="carpartnerResponse">
          </bpel:invoke>
</bpel:sequence>
<bpel:sequence name="airlineSequence">
        <bpel:assign validate="no" name="AssignAirlineInput">
           <bpel:copy>
           <bpel:from><![CDATA[$input.payload/tns:lastname]]>
           </bpel:from>…….</bpel:copy>
</bpel:assign>

          <bpel:invoke name="InvokeAirline"
                         partnerLink="airlinepartner"
                         operation="bookFlight"
                         portType="ns:AirLineBook"
                         inputVariable="airlinepartnerRequest"

outputVariable="airlinepartnerResponse">
          </bpel:invoke>
</bpel:sequence>
</bpel:flow>
```

```
<bpel:if name="If">

     <bpel:condition><![CDATA[$airlinepartnerResponse.paramet
ers/ns:bookFlightReturn/ns:booked='true' and
$carpartnerResponse.parameters/car:out=true() and
$hotelpartnerResponse.parameters/ns0:out=true()]]></bpel:condi
tion>
<bpel:assign validate="no" name="resOK">
          <bpel:copy>……..</bpel:copy>
          <bpel:copy>
          <bpel:from></bpel:to></bpel:copy><bpel:copy>

     <bpel:to><![[$output.payload/tns:airLineReservationId]]>
               </bpel:to>
               </bpel:copy>
               </bpel:copy>
</bpel:assign>
          </bpel:else>
          </bpel:if>


<bpel:reply       name="replyOutput"
                  partnerLink="client"
                  portType="tns:travelAgency"
                  operation="bookTravel"
                  variable="output" />
</bpel:sequence>
</bpel:process>
```

4.2.3. Transformation au réseau de pétri coloré

Nous suivrons les mêmes étapes d'exemple précédent, donc il n'est pas nécessaire de tout mentionner :

✓ bouton « Cp-Net » :

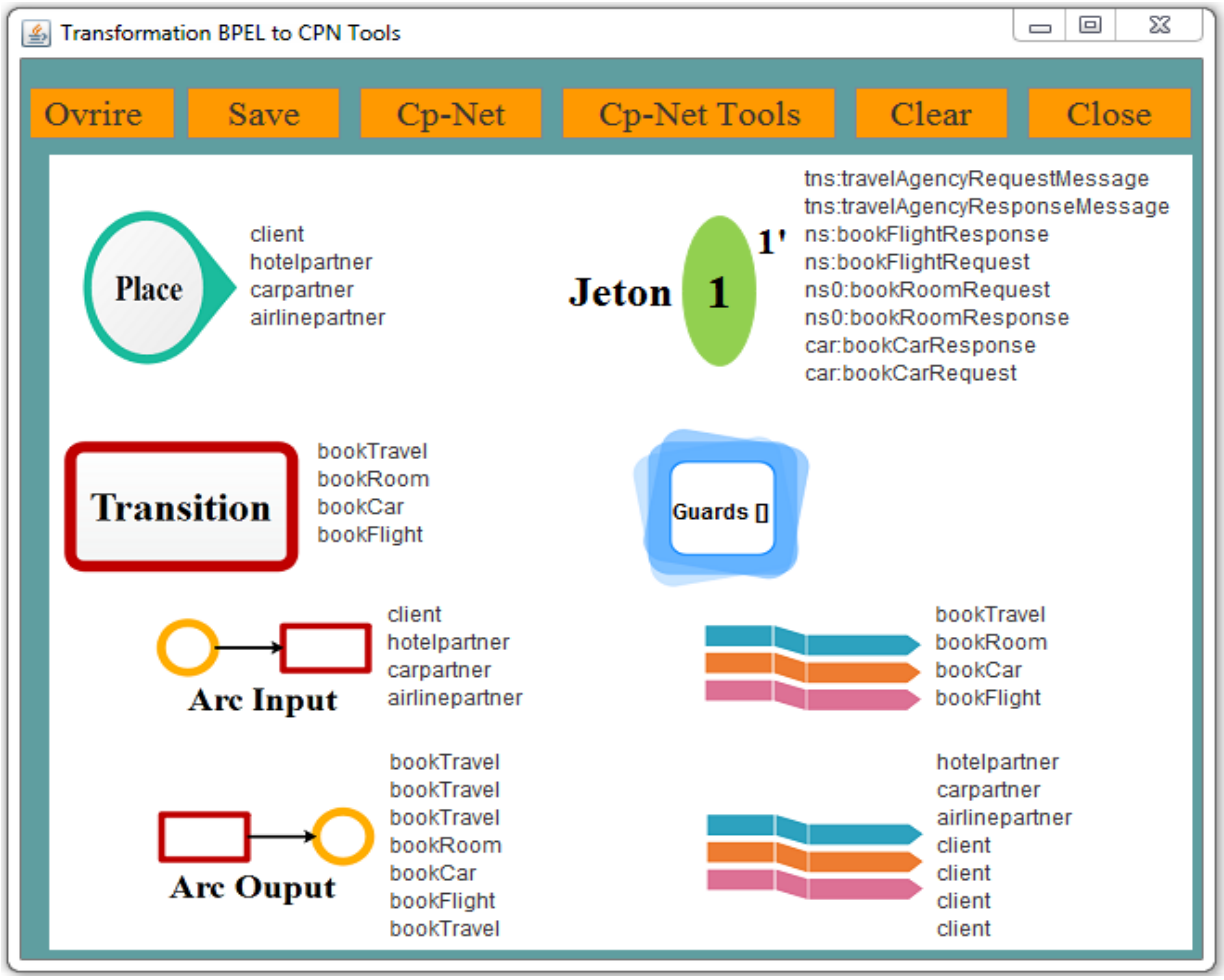


Figure 4.19 : Interface de résultat de Bouton « Cp-Net » fichier BPEL «Travel Agency ».

✓ bouton « Save », enregistrer les résultats obtenus dans le fichier « CPN », figure 4.19 :

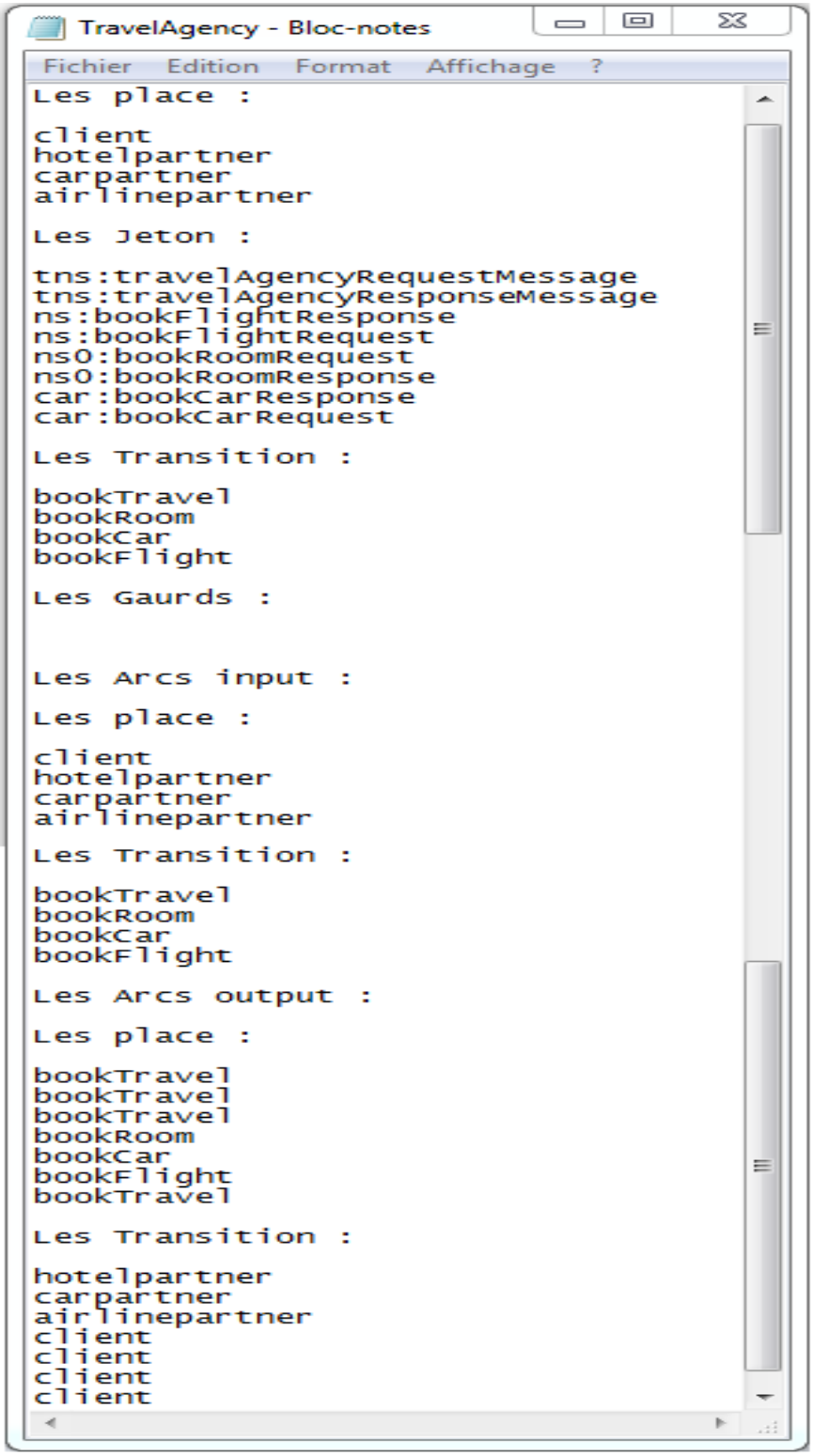


Figure 4.20 : Fichier « TravelAgency.cpn ».

✓ Nous obtenons une interface dans le programme « CPN Tools » qui contient une modélisation formelle du fichier BPEL « Travel Agency » en utilisant les réseaux de pétri colorés comme dans la figure suivante :

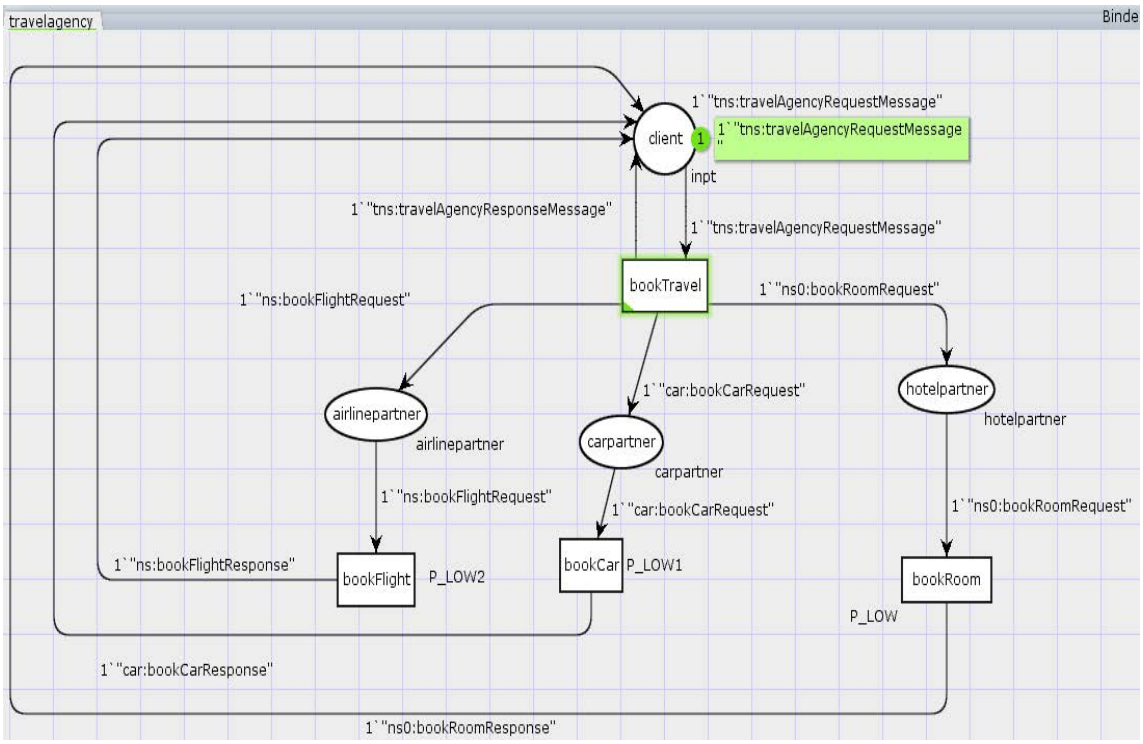


Figure 4.21 : Interface de modélisation formelle du fichier BPEL « Travel Agency » dans programme « CPN Tools ».

✓ L'outil « CPN » à dans les outils « Create », figure 4.22 :

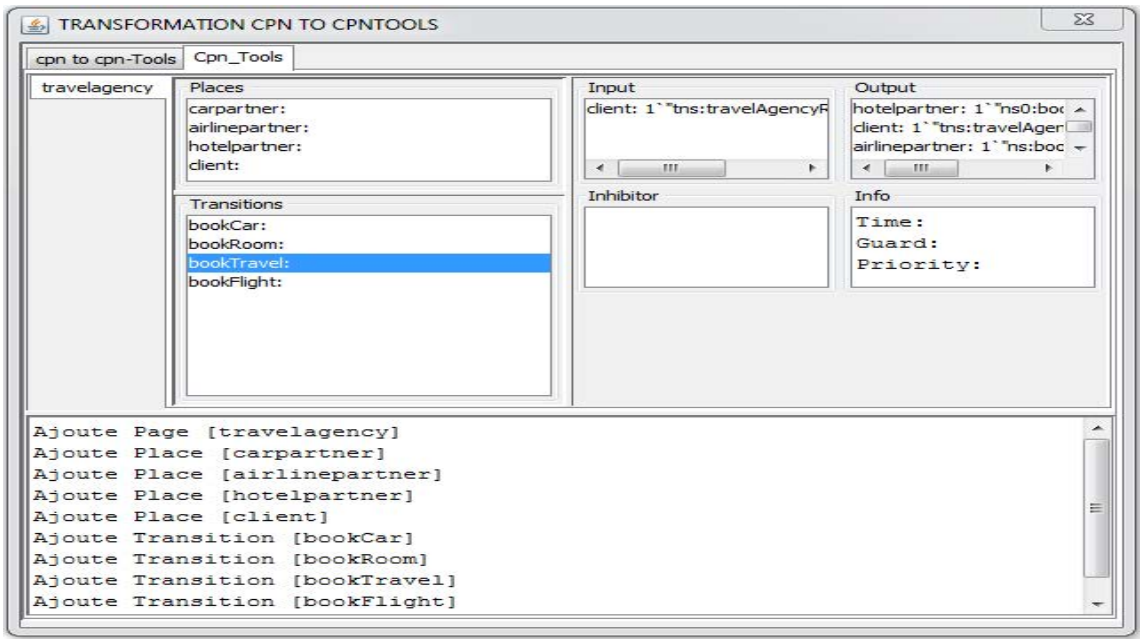


Figure 4.22 : Interface Java qui définit les places et les transitions du fichier BPEL « Travel Agency » dans programme « CPN Tools ».

4.3. Le servie composite gestion de crédit

Cet exemple présente un processus métier d'évaluation de demande de crédit sur un achat d'une voiture [86].

4.3.1. Description du scénario

La gestion de crédit est un processus métier d'évaluation de demande de crédit sur un achat d'une voiture et il est réalisé en plusieurs étapes :

1. La réception de la demande du client qui inclut :
   - son numéro de sécurité sociale (SSN)
   - le montant du crédit demandé (CR)
   - son salaire brut (SAL)
2. L'évaluation de la solvabilité du demandeur en effectuant une requête auprès du service dédié (en se basant sur le SSN du demandeur) et retournant sa qualification (solvable ou non)
3. En cas de solvabilité, la demande est envoyée à la banque (CR et SAL)
4. La proposition de la banque est renvoyée au demandeur, et inclut les informations suivantes : taux du crédit, nombre des années de remboursement (figure 4.23) :

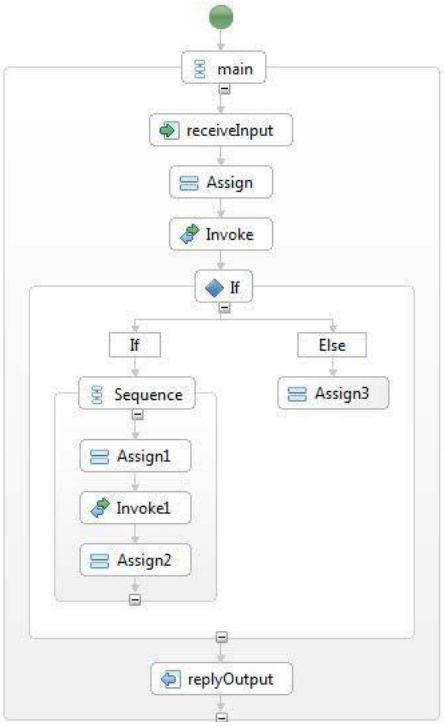


Figure 4.23 : La représentation graphique du processus de gestion de crédit. [86]

4.3.2. La spécification BPEL de processus de gestion de crédit

```
<bpel:process
name="GestionCreditProcess"suppressJoinFailure="yes"
              xmlns:bpel="http://docs.oasisopen.org/wsbpel/2.0/
              process/executable"
              xmlns:ns="http://ws.banque.tn/" xmlns:ns0="http://
              ws.credit.tn/">

    <bpel:partnerLinks>
        <bpel:partnerLink

name="client"partnerLinkType="tns:GestionCreditProcess"
              myRole="GestionCreditProcessProvider"/>
        <bpel:partnerLink name="solvabilitePL"
                          partnerLinkType="tns:solvabilitePLT"
                          partnerRole="solvabiliteConsumer">
        </bpel:partnerLink>
        <bpel:partnerLink name="creditPL"
                          partnerLinkType="tns:creditPLT"
                           partnerRole="creditConsumer">
        </bpel:partnerLink>
    </bpel:partnerLinks>

    <bpel:variables>
            <bpel:variable name="input"
         messageType="tns:GestionCreditProcessRequestMessage"/>
             <bpel:variable name="output"
     messageType="tns:GestionCreditProcessResponseMessage"/>
              <bpel:variable name="solvabilitePLResponse"

messageType="ns:evaluerSolvabiliteResponse">
              </bpel:variable>
              <bpel:variable name="solvabilitePLRequest"

messageType="ns:evaluerSolvabilite">
              </bpel:variable>
              <bpel:variable name="creditPLResponse"

messageType="ns0:traiterCreditResponse">
              </bpel:variable>
              <bpel:variable name="creditPLRequest"
                             messageType="ns0:traiterCredit">
              </bpel:variable>
    </bpel:variables>
```

```
    <bpel:sequence name="main">

        <bpel:receive name="receiveInput"
                      partnerLink="client"
                      portType="tns:GestionCreditProcess"
                      operation="process" variable="input"
                      createInstance="yes"/>

        <bpel:assign validate="no" name="Assign">
         <bpel:copy><bpel:from>
           <bpel:assign validate="no" name="Assign">
                         <bpel:copy>
                         <bpel:from>
                        <bpel:literal>
   <tns:evaluerSolvabilite xmlns:tns="http://ws.banque.tn/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                   instance"><ssn>ssn</ssn>

                            </tns:evaluerSolvabilite>
                     </bpel:literal>
                </bpel:from>
                <bpel:to variable="solvabilitePLRequest"
part="parameters"></bpel:to>
            </bpel:copy>
            <bpel:copy>
             <bpel:from part="payload" variable="input">
                            <bpel:query
 queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0
                                  ">
               <![CDATA[tns:ssn]]></bpel:query>
                </bpel:from></bpel:copy></bpel:assign>

<bpel:invoke name="Invoke"
             partnerLink="solvabilitePL"
             operation="evaluerSolvabilite"
             portType="ns:SolvabiliteWS"
             inputVariable="solvabilitePLRequest"
             outputVariable="solvabilitePLResponse">
</bpel:invoke>
        <bpel:if name="If">
            <bpel:condition>
      <![CDATA[string($solvabilitePLResponse.parameters/
                    estSolvable)="true"]]>
                       </bpel:condition>
```

```
            <bpel:sequence>
                <bpel:assign validate="no" name="Assign1">
                    <bpel:copy>
                        <bpel:from>
                            <bpel:literal>
      <tns:traiterCredit xmlns:tns="http://ws.credit.tn/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                         <montant>0.0</montant>
                         <salaire>0.0</salaire>

                                    </tns:traiterCredit>
                                </bpel:literal>
                            </bpel:from>
                        </bpel:copy>

                    </bpel:assign>

<bpel:invoke name="Invoke1"
              partnerLink="creditPL"
              operation="traiterCredit"
              portType="ns0:CreditWS"
              inputVariable="creditPLRequest"
              outputVariable="creditPLResponse">
</bpel:invoke>

                    <bpel:assign validate="no" name="Assign2">
                        <bpel:copy>……</bpel:assign>
            </bpel:sequence>
            <bpel:else>
                <bpel:assign validate="no" name="Assign3">
                    <bpel:copy>
                    </bpel:copy>
                </bpel:assign>
            </bpel:else>
        </bpel:if>

<bpel:reply name="replyOutput"
            partnerLink="client"
            portType="tns:GestionCreditProcess"
            operation="process"
            variable="output"/>
    </bpel:sequence>
</bpel:process>
```

4.3.3. Transformation au réseau de pétri coloré

✓ bouton « Cp-Net » :

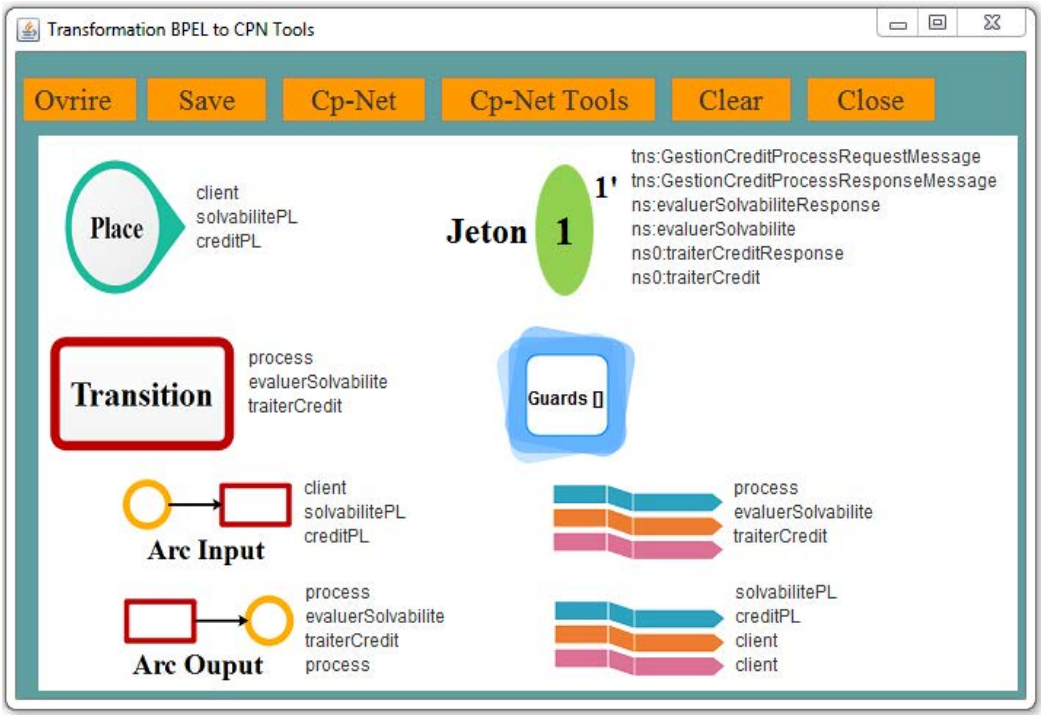


Figure 4.24 : Interface de résultat de Bouton « Cp-Net » fichier BPEL « gestion de crédit ».

✓ bouton « Save » enregistrer les résultats obtenus dans le fichier « CPN », il représente dans cette figure :

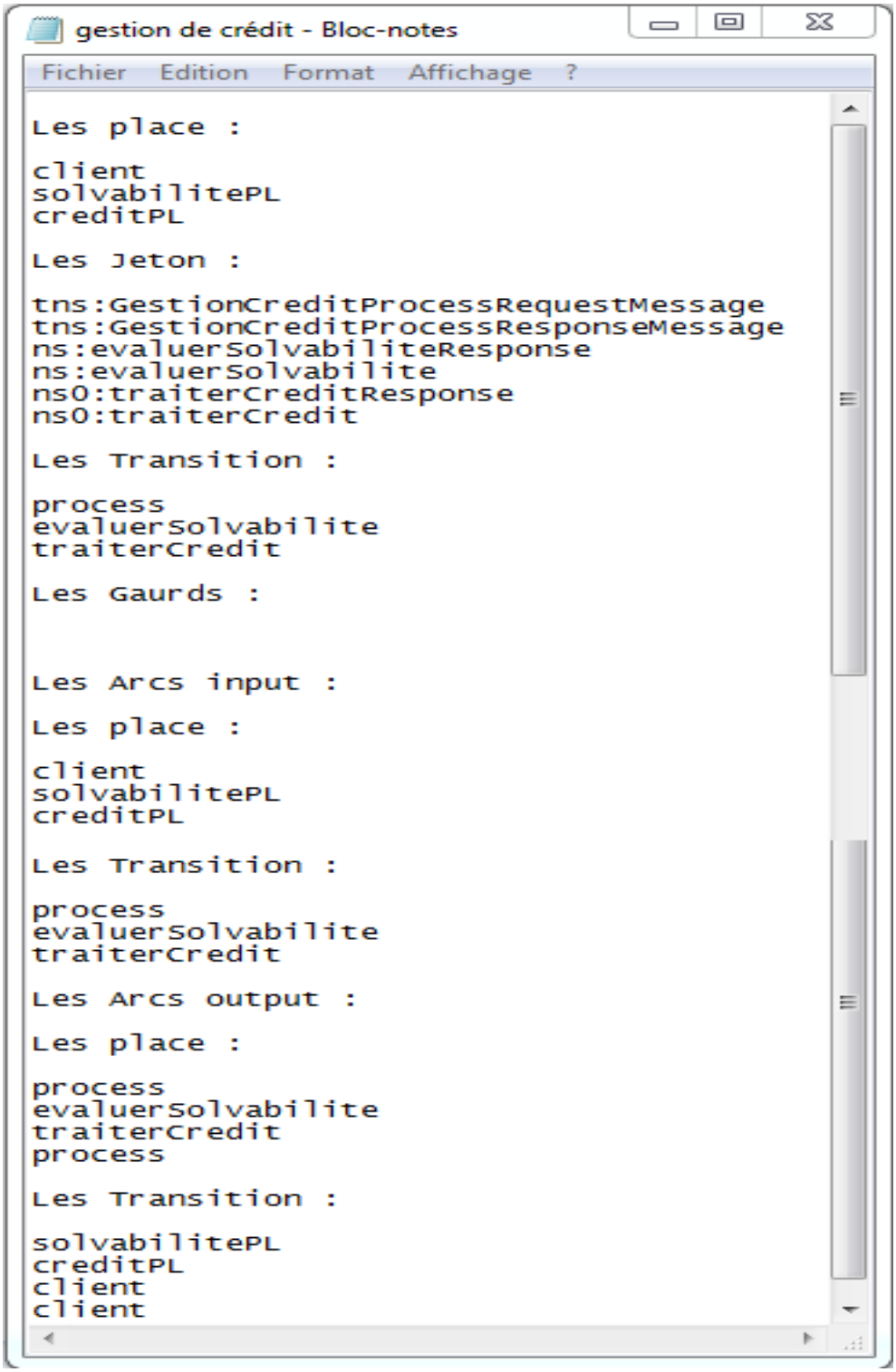


Figure 4.25 : Interface de fichier enregistrer « gestion de crédit.cpn ».

✓ Nous obtenons une interface dans le programme « CPN Tools » qui contient une modélisation formelle du fichier BPEL « gestion de crédit » utilisation de réseaux de pétri colorés comme dans la figure suivante :

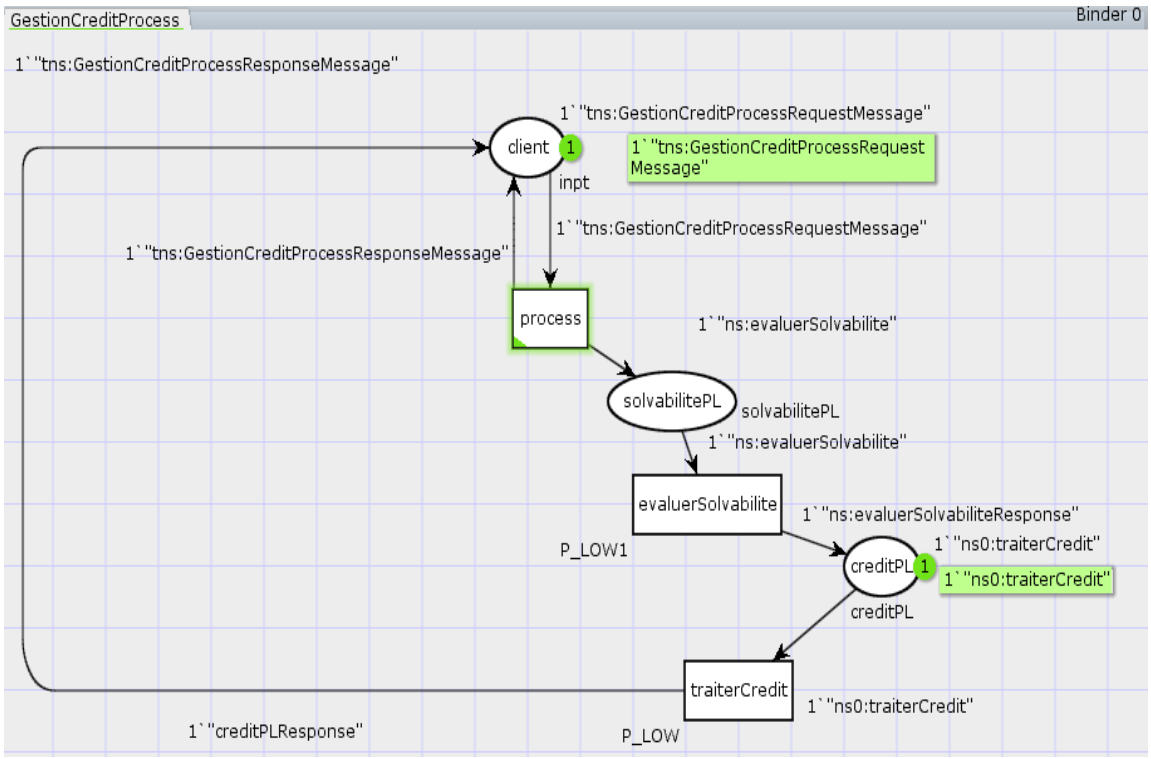


Figure 4.26 : Interface de modélisation formelle du fichier BPEL « gestion de crédit » dans programme « CPN Tools ».

✓ le outil « CPN » à dans les outils « Creat », il représente dans cette figure :

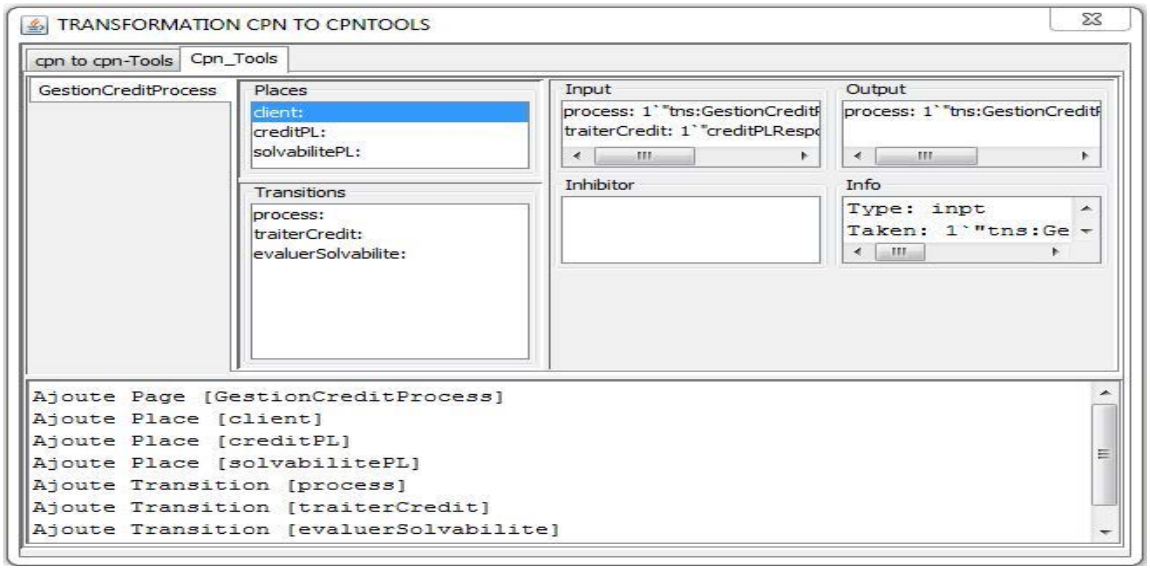


Figure 4.27 : Interface de Java qui définir des places et des transitions du fichier BPEL « gestion de crédit » dans programme « CPN Tools ».

4.4. Le service composite de calcul de prix TTC

Cet exemple calcule le prix TTC (toutes taxes comprises) d'un produit [86]. Ce processus se base sur les deux services web suivants :

1. TVAService : un service Web qui détermine le taux de la TVA (Taxe sur la Valeur Ajoutée) d'un produit en fonction de son code.
2. TTCService : un service Web qui calcule le prix TTC en fonction du prix HT (Prix Hors Taxes) et du taux de la TVA.

4.4.1. Description du scénario

Le processus BPEL (calcul de prix TTC) orchestre les deux services TVAService et TTCService :

- Le processus prend en entrée le code d'un produit (entier) et son prix HT.
- (double). Le prix HT doit être strictement supérieur à zéro sinon le processus retourne le message suivant « Erreur de saisie du prix HT ».
- Le processus invoque le service TVA qui retourne le taux TVA d'un produit en fonction de son code.
- Une fois le taux TVA est obtenu, le processus fait appel au service TTC qui retourne le prix TTC du produit en question en fonction de sin taux TVA ainsi que son prix HT.
- Si le calcul du prix TTC s'est bien effectué, le message suivant est affiché « Le prix TTC du produit [code] est [prixTTC] ». Il est présenté dans la figure suivante:

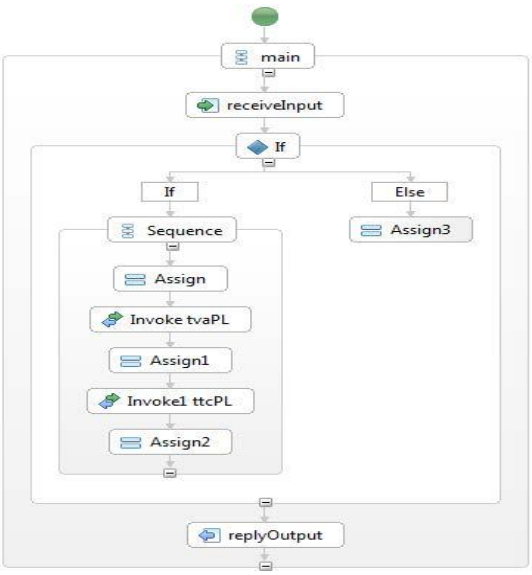


Figure 4.28 : La représentation graphique du processus calcul de prix TTC. [86]

4.4.2. La spécification BPEL du processus de calcul de prix TTC

```
<bpel:process name="CalculPrixProcess"
              xmlns:bpel="http://docs.oasis
              open.org/wsbpel/2.0/process/executable"
              xmlns:ns="http://ws.tva.esprit.tn/"
              xmlns:ns0="http://ws.ttc.esprit.tn/">


    <bpel:partnerLinks>
        <bpel:partnerLink name="client"

partnerLinkType="tns:CalculPrixProcess"
                          myRole="CalculPrixProcessProvider"/>
        <bpel:partnerLink name="tvaPL"
                          partnerLinkType="tns:tvaPLT"

partnerRole="tvaConsumer"></bpel:partnerLink>
        <bpel:partnerLink name="ttcPL"
                          partnerLinkType="tns:ttcPLT"

partnerRole="ttcConsumer"></bpel:partnerLink>
    </bpel:partnerLinks>
    <bpel:variables>

        <bpel:variable name="input"

messageType="tns:CalculPrixProcessRequestMessage"/>

        <bpel:variable name="output"

messageType="tns:CalculPrixProcessResponseMessage"/>
        <bpel:variable name="tvaPLResponse"
                       messageType="ns:calculerTVAResponse">
        </bpel:variable>
        <bpel:variable name="tvaPLRequest"
                       messageType="ns:calculerTVA">
        </bpel:variable>
        <bpel:variable name="ttcPLResponse"
                       messageType="ns0:calculerTTCResponse">
        </bpel:variable>
        <bpel:variable name="ttcPLRequest"
                       messageType="ns0:calculerTTC">
        </bpel:variable>
    </bpel:variables>
```

```
<bpel:sequence name="main">
        <bpel:receive name="receiveInput"
                      partnerLink="client"
                      portType="tns:CalculPrixProcess"
                      operation="process" variable="input"
                      createInstance="yes"/>


        <bpel:if name="If">

<bpel:condition><![CDATA[$input.payload/tns:prixHT>0]]>
</bpel:condition>
            <bpel:sequence>
                <bpel:assign validate="no" name="Assign">
                    <bpel:copy>
                        <bpel:from>
                            <bpel:literal>
     <tns:calculerTVA xmlns:tns="http://ws.tva.esprit.tn/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                         <code>0</code>

                               </tns:calculerTVA>
                            </bpel:literal>
                        </bpel:from></bpel:copy>

                </bpel:assign>



<bpel:invoke name="Invoke"
             partnerLink="tvaPL"
             operation="calculerTVA"
             portType="ns:TvaWS"
             inputVariable="tvaPLRequest"
             outputVariable="tvaPLResponse">
</bpel:invoke>
                <bpel:assign validate="no" name="Assign1">
                    <bpel:copy>

                    </bpel:copy>
                </bpel:assign>

<bpel:invoke name="Invoke1"
             partnerLink="ttcPL"
             operation="calculerTTC"
             portType="ns0:TtcWS"
             inputVariable="ttcPLRequest"
             outputVariable="ttcPLResponse">
</bpel:invoke>
```

```
<bpel:assign validate="no" name="Assign2">
                    <bpel:copy>
                        <bpel:from>
                            <bpel:literal>
                            </bpel:literal>
                        </bpel:from>
                    </bpel:copy>
                </bpel:assign>
            </bpel:sequence>
            <bpel:else>
                <bpel:assign validate="no" name="Assign3">
                    <bpel:copy>
                        <bpel:from>

                            </bpel:literal>
                        </bpel:from>
                    </bpel:copy>
                </bpel:assign>
            </bpel:else>
        </bpel:if>


<bpel:reply name="replyOutput"
            partnerLink="client"
            portType="tns:CalculPrixProcess"
            operation="process"
            variable="output"/>

</bpel:sequence>
</bpel:process>
```

4.4.3. Transformation au réseau de pétri coloré

✓ bouton « Cp-Net » :

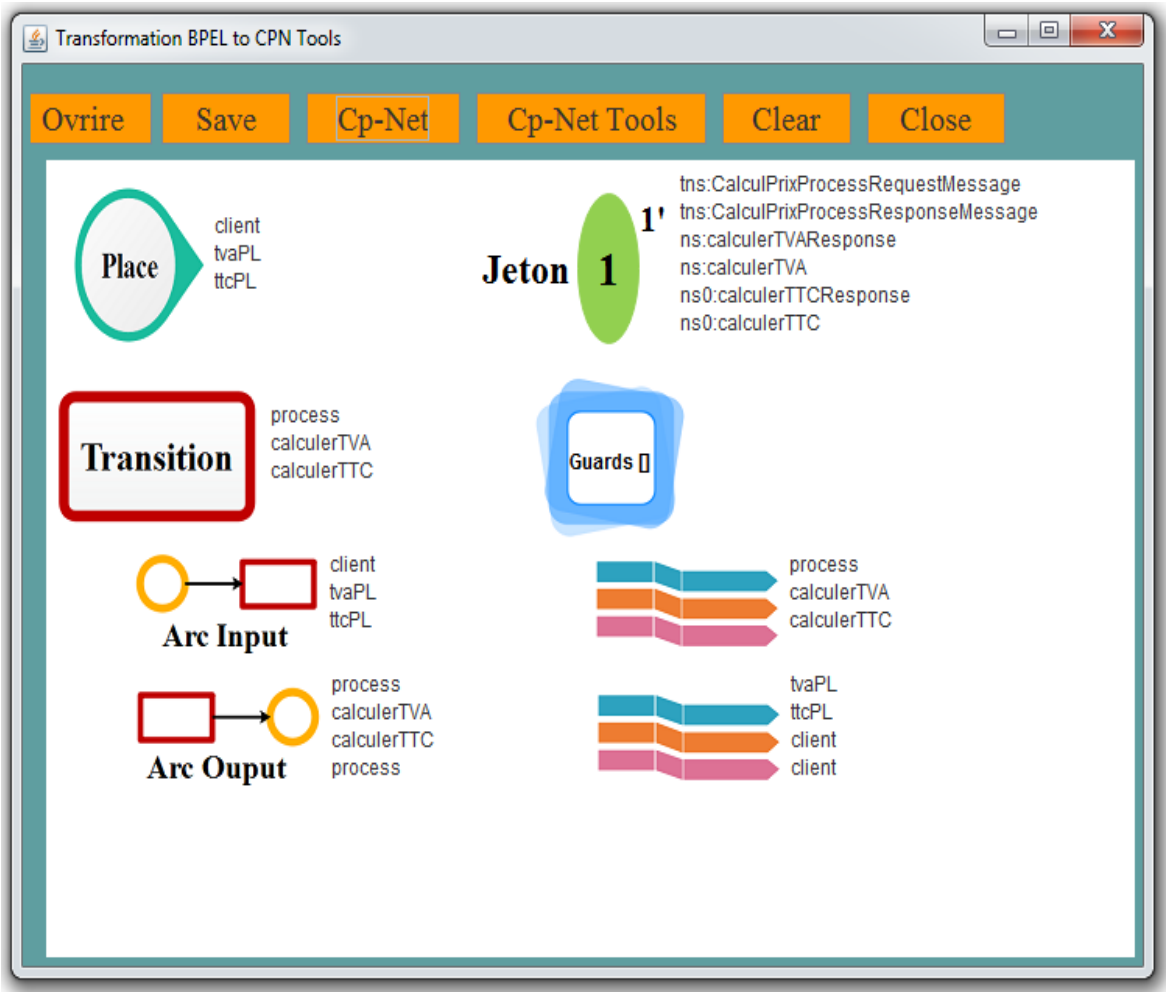


Figure 4.29 : Interface de résultat de Bouton « Cp-Net » du fichier BPEL « prix TTC ».

✓ bouton « Save » pour enregistrer les résultats obtenus dans le fichier « CPN », il est présenté dans la figure :

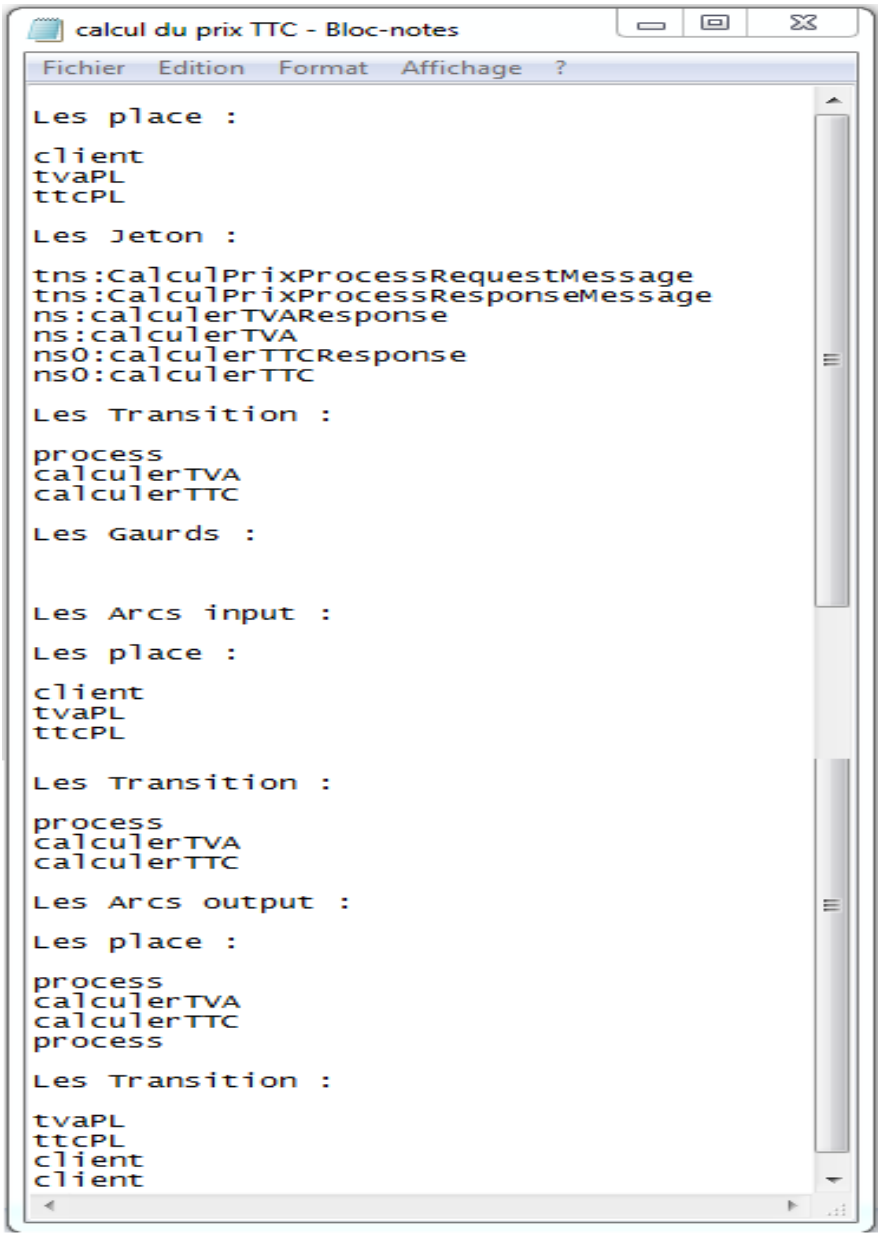


Figure 4.30 : Interface de fichier « calcul du prix TTC.cpn ».

✓ Nous obtenons une interface dans le programme « CPN Tools » qui contient une modélisation formelle du fichier BPEL « prix TTC » en utilisant les réseaux de pétri colorés comme montré dans la figure suivante :

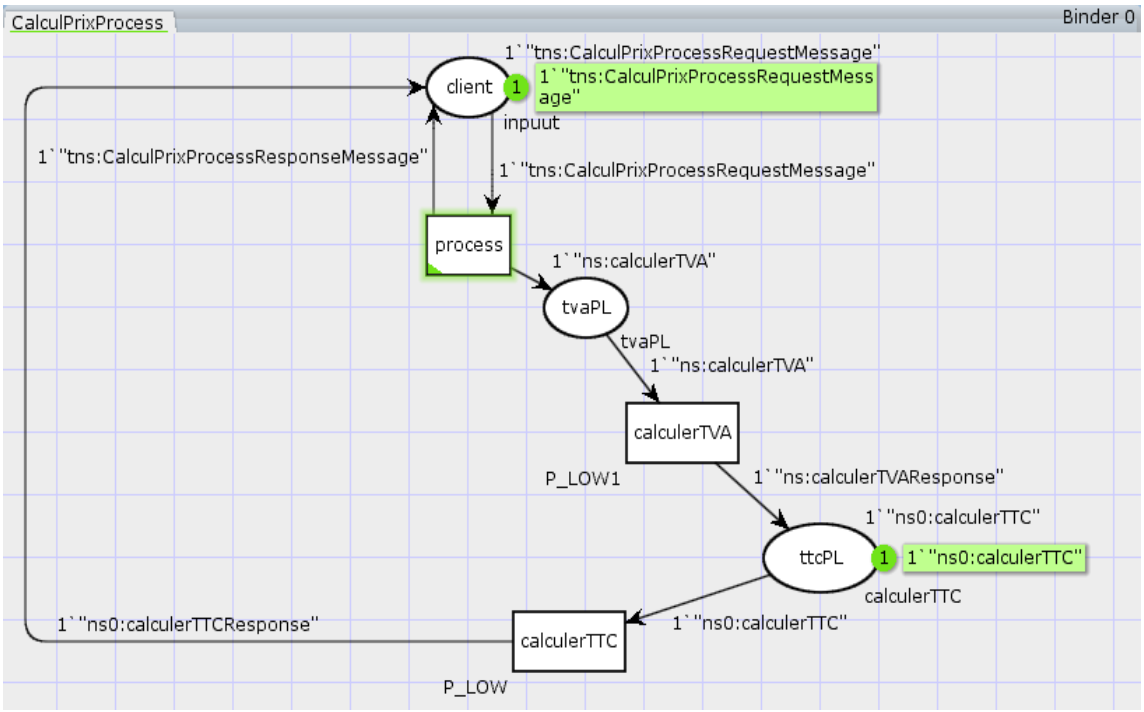


Figure 4.31 : Interface de modélisation formelle du fichier BPEL « calcul du prix TTC » dans programme « CPN Tools ».

✓ L'outil « CPN » dans « Creat », présenté dans cette figure :

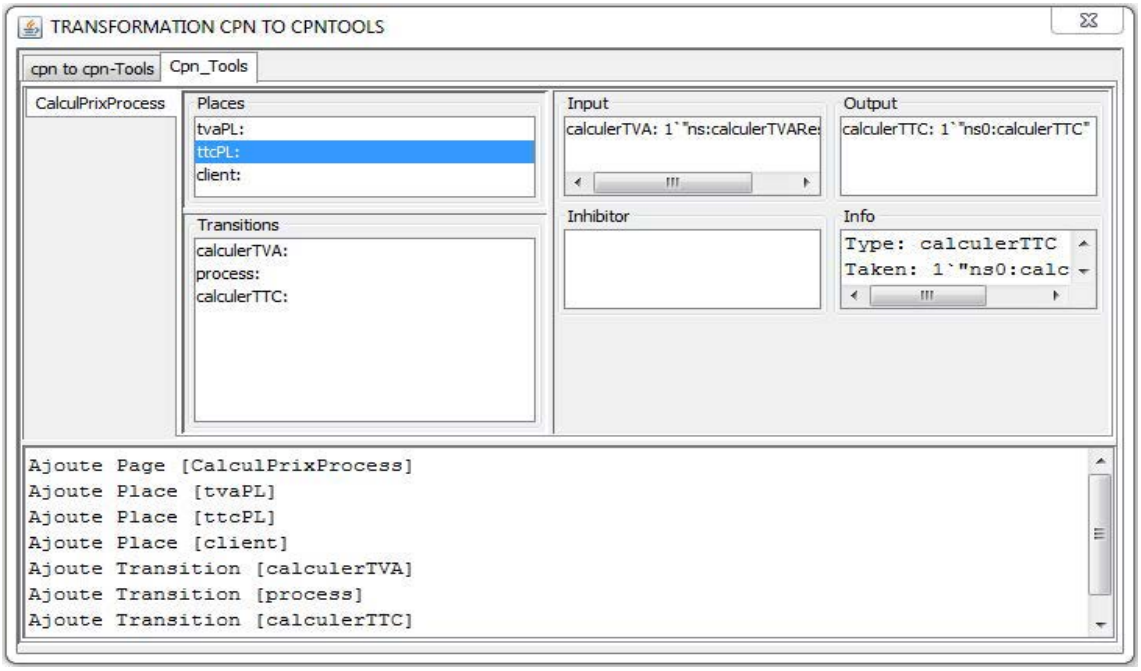


Figure 4.32 : Interface Java qui définit les places et les transitions du fichier BPEL « calcul du prix TTC » dans programme « CPN Tools ».

5. Discussion

Pour bien critiquer notre travaille nous allons le comparer avec d'autres travaux similaires en se basant sur des critères que nous allons présenter. Les critères de comparaisons sont extraits des caractéristiques des services composites et des RdPC:

- ✓ Test réel : l'utilisation des exemples des tests réels pour la validation de notre travail comme nous avons vu dans ce chapitre : « Travel », « Travel Agency », « gestion de crédit », « calcul de prix TTC », contrairement aux autres travaux.
- ✓ Les étapes de modélisation : Ce que signifie comment convertir un fichier BPEL en un réseau de Pétri coloré, et quelles sont les caractéristiques du fichier BPEL …. Etc. La majorité des travaux connexes n'ont pas fourni d'explication détaillée sur la façon de lire un fichier BPEL et de définir ses caractéristiques (les activités), seulement ils ont donné un indice qui varie d'un chercheur à l'autre. Par contre dans notre travail nous avons bien détaillé la transformation et la modélisation avec les RdPC à partir du fichier BPEL.
- ✓ Implémentation réelle: Nous avons essayé d'implémenté notre système par une application qui permet d'ouvrir et de lire un fichier « BPEL », rechercher ses propriétés (fichier BPEL) qui présentent les activités, et transformer ces propriétés à la forme des places, des transitions, des jetons et des arcs. Après nous avons connecté la plate-forme « Eclipse » à la plateforme de modélisation « CPN Tools » pour dessiner des réseaux de Pétri colorés. Alors que tous les autres chercheurs des travaux connexes que nous avons cité dans notre mémoire n'ont pas parlé de ce point.
- ✓ L'outil de simulation et vérification: nous avons utilisé le « CPN Tools » qui présente un outil très puissant pour modéliser graphiquement les RdPC, grâce à une propriété de simulation automatique sachant que cet outil permet de vérifier les RdPC ce qui va nous faciliter la tâche de simulation graphique et de vérification.
- ✓ Vérification : nous avons vérifié les services composites via la vérification des RdPC selon les caractéristiques suivantes telle que: « synchronisation », « concurrence », « Détection d'inter-blocage », ….. etc. En réalité ces caractéristiques présentent des pannes dans les services composites et c'est l'objectif de la modélisation formelle qui facilite énormément la tâche de vérification.

Le tableau suivant présente une comparaison de notre travail avec les travaux connexes :

	Test réel	Les étapes de modélisation	Implémentation réelle	L'outil de simulation et vérification	Vérification
[69]	+	-	-	-	-
[70]	-	+	-	-	-
[71]	-	+	-	+	+
[72]	-	-	-	-	+
[73]	-	-	-	+	+
[74]	-	+	-	-	-
[75]	-	+	-	-	-
[76]	+	-	-	+	-
[77]	-	+	-	-	-
[78]	-	-	-	+	-
Notre travail	++	++	++	+	++

Tab 4.1 : comparaison de notre travail avec les travaux connexes

Les fichiers bpel présentent des services composites difficiles à vérifier à cause de leur manque de formalisme. Pour remédier à ce problème, il est nécessaire d'avoir une méthode qui permet d'analyser les services web composites avant leurs mises en œuvre, dans le but de fournir et de créer des services web composites corrects. Dans ce cas, les méthodes formelles apparaissent comme la solution la plus adéquate. Par exemple, si on veut vérifier un fichier BPEL il faut comprendre le fonctionnement de ce fichier : comment les messages sont envoyés aux services, les types des messages, qui est le premier émetteur, dernier récepteur, comment connecter des services avec d'autres services, assurer les entrées et les sorties de service, y a-t-il des services arrêtés, quels services nous manquent …..ect. C'est pourquoi nous avons modélisé les services web composites en utilisant des réseaux pétri colorés. Ainsi, après avoir transformé

le fichier bpel et obtenir un fichier contenant les places et les transitions et les arcs, nous avons utilisé le logiciel « CPN Tools » qui permet de tracer un RdPC.

A ce niveau nous avons essayé de réaliser le passage automatique de la spécification formelle à la présentation graphique des RDPC via le CPN tools et c'est ce que nous avons trouvé impossible car il nécessite d'avoir le code source du logiciel CPN tools pour le modifier. Mais nous avons ajouté des extensions dans le serveur où nous somme arrivé à automatiser l'interface « Binder » dans CPN tools, et à ce niveau l'intervention de l'utilisateur sera obligatoire pour entrer la spécification de l'RDPC

Et comme la vérification est l'objectif de modélisation formelle, notre systèmepermet également de vérifier certaines propriétés telles que: « synchronisation », « concurrence », « Détection d'inter-blocage », ….. ect des RdPC. Cette vérification permet de vérifier la conformité de ces services web composites pour s'assurer de leurs bons fonctionnements et ainsi garantir l'interopérabilité technique.

6. Conclusion

Le chapitre d'implémentation a pour objectif de présenter les différents aspects pratiques de de notre système. Malgré les difficultés que nous avons rencontrées pour connecter l'environnement « Eclipse » et l'outil « CPN Tools » et qui n'était pas réalisé officiellement par Eclipse, mais nous avons fourni des solutions appropriées pour le processus de conversion du fichier bpel aux réseaux de Petri colorés.

« Conclusion générale »

Conclusion générale

Les services Web composites sont devenus très importants dans notre vie parce qu'ils couvrent les besoins variés et multiples des utilisateurs. Le langage BPEL est un langage de composition des services web très utilisé qui fournit des services composites difficiles à vérifier à cause de leur manque de formalisme. Pour remédier à ce problème, il est nécessaire d'avoir une méthode qui permet d'analyser les services web composites avant leurs mises en œuvre, dans le but de fournir et de créer des services web composites corrects. Dans ce cas, les méthodes formelles apparaissent comme la solution la plus adéquate

Afin de vérifier le bon fonctionnement du service web composite présentés par des fichiers BPEL, nous avons fourni une architecture de modélisation de ces services en utilisant un formalisme très puissant qui est les réseaux de Pétri colorés à raison de leur pouvoir de modélisation.

Nous avons également discuté les caractéristiques du fichier BPEL et comment le transformé aux réseaux pétris colorés avec des exemples pour tester la puissance de notre système. Et le point le plus fort de notre système c'est qu'après la modélisation des services Web composite, nous pouvons vérifier certaines propriétés telles que : synchronisation, concurrence, détection d'inter-blocage,….. etc, pour assurer le bon fonctionnement de ces services Web avant de les fournir aux l'utilisateurs.

Nous pouvons citer quelques perspectives pour les chercheurs qui veulent utiliser notre travail où ils peuvent améliorer notre architecture par l'implémentation d'un nouvel outil à la place de CPN tools pour permettre le passage automatique de la spécification formelle à la présentation graphique des RDPC car le passage direct via le CPN tools nécessite d'avoir le code source du logiciel CPN tools pour le modifier et c'est ce qui est impossible. Sachant que nous avons essayé de réaliser ce passage en ajoutant des extensions dans le serveur où nous somme arrivé à automatiser l'interface « Binder » dans CPN tools, mais à ce niveau l'intervention de l'utilisateur sera obligatoire pour entrer la spécification de l'RDPC.

Une autre perspective pour passer à un domaine très récent et très intéressant, c'est l'orchestration des services objets en utilisant BPEL dans l'internet des objets. Notre architecture peut être utilisée comme une base pour modéliser formellement les

services composites dans l'internet des objets présentés par des fichiers BPEL en utilisant les RDPC. Mais dans ce point il faut noter que les chercheurs dans ce domaine ont ajouté des extensions à BPEL pour couvrir les caractéristiques de l'internet des objets.

« Bibliographie »

« Bibliographie »

[1] Jacobson, « Object-oriented software engineering.acm ». 1991.

[2] M. Dodani, « From objects to services: A journey in search of component reuse nirvana », Journal of Object Technology, 2004.

[3] S. Hashimi, « Service-oriented architecture explained », 2003.

[4] R.Schulte , V.Natis. « Service oriented architectures », 1996.

[5] R.FAROS, «Etat de l'art sur la contractualisation et la composition ». Réseau National en Technologies Logicielles (RNTL),2006.

[6] B.Margolisand, J.Sharpe, « SOA for the Business Developer : Concepts,BPEL, and SCA », édition MC Press, 2007.

[7] N.Josuttis, « SOA in Practice », 2007.

[8] S.Hack, M.Lindemann, « Enterprise SOA , Einführen, Bonn, Allemagne, Galileo Press », 2007.

[9] F.Jennings, R.Loganathan, et all, « Approach to Integration ; XML, Web services, ESB, and BPEL in real-world SOA projects », édition Packt Publishing Ltd, 2007.

[10] C. Marin, « Une approche orientée domaine pour la composition de services ». Thèse de doctorat, Université Joseph Fourier, 2008.

[11] W.Schreiner, S.tdar, « A survey on Web services composition, Int. J. Web and Grid Services », 2005

[12] P.Kellert , F.Toumani. « Les web services sémantiques". In Web sémantique », Action spéci_que 32 CNRS/STIC, 2003.

[13] F.Jonathan, L.Philippe, et all. « "Les web services".INGÉNIERIE SYSTÈMES - Les web services - Concevoir et utiliser des applications 2.0 - C#, Java, PHP, API JavScript, Androïd SDK, iOS SDK … (niveau C) ».2013.

[14] P.Mohamed Youssfi .« Web Services SOAP et RESTful ». ENSET, Université Hassan II Mohammedia.2012

[15] E.Cerami, « Web Services Essentials », édition O'Reilly, 2002.

[16] K.Radhia. « Vérification de la Compatibilité des Services Web pour une Composition », . Thèse de Magister, Université Constantine2.2014

[17] Inc.SoftQuad, « The SGML Primer. SoftQuad's Quick Reference Guide to the Essentials of the Standard: The SGML Needed for Reading a DTD and Marked-up Documents and Discussing them Reasonably ». Version 3.0, 1991.

[18] T.Melliti, «Interopérabilité des services Web complexes, Application aux systèmes multi-agents(in french) », PhD thesis, Department of Computer Science, University of Paris IX Dauphine, 2004.

[19] D.C.Fallside , P.Walmsley, « XML Schema Part0: Primer Second Edition », W3C,2004.

[20] M.Kay, « XSL Transformations (XSLT) », version 2.0, W3C Recommendation, 23 Jan 2007.

[21] J.Robie, D.Chamberlin, et all, « XML Path Language (XPath) », version 3.0, W3C Recommendation, 08 Apr 2014,

[22] C.Lablanche, F.Seine, et all. « Les Web Services ».2005.

[23] E.Christensen, F.Curbera, et all. « Web Services Description Language (WSDL) ». W3C ,2001.

[24] H.S.Thompson, D.Beech, et all. « XML Schema Part 1: Structures Second Edition ». W3C ,2004.

[25] E.Christensen, F.Curbera, et all, « Web services description language (wsdl) », 2001

[26] F.Curbera, M.Duftler, et all, « Unraveling the web services web an introduction to soap, wsdl, and uddi », IEEE INTERNET COMPUTING,2002.

[27] M.Dovey, I.Kostadinov, et all « Engineering Tasks Force Evaluation of UDDI for UK e-Science ». UK Technical Report ,2005.

[28] Amina bekkouche. « Composition des Services Web Sémantiques À base d'Algorithmes Génétiques », Thèse de Magister, Université Constantine2, Abou-bekr Belkaid Tlemcen 2012.

[29] M.Matskin, P.Küngas, et all. « Enabling Web Services Composition with Softward Agents ». In IMSA , 2005.

[30] C.Preist, A.Byde, et all. « Towards Agent-based Service Composition through Negotiation in Multiple Auctions ». Technical Report hpl-2001-71, Hewlett Packard, 2001.

[31] A.Gustavo, F.Casati, et all. « Web Services. Concepts, Architectures and Applications ». Springer-Verlag Berlin Heidelberg, 2004.

[32] A.Aît-Bachir, A.Med, « un canevas pour la détection et la résolution des incompatibilités des conversations entre services web », thèse de doctorat, Université Joseph Fourier-Grenoble 1, 2008.

[33] J.Rao , X.Su, et all. « A survey of automated web service composition methods. In Proc of Semantic Web Services and Web Process Composition », First International Workshop,San Diego, CA, 2004.

[34] B.Benatallah, M.Dumas, et all. « Towards Patterns of Web Services Composition »,2003

[35] B.Benatallah, Dumas, et all, « Overview of Some Patterns for Architecting and Managing Composite Web Services ». ACM SIGecom Exchanges, 2002.

[36] T.R.Payne , O.Lassila . « Semantic Web Services ». IEEE Intelligent Systems. 2004.

[37] Z.Tari, P.Bertok, et all, « Verification of communication protocols in web services: model-checking service compositions » , Wiley series on parallel and distributed computing.2013

[38] F.Mohamad, « Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multiagents », thèse de doctorat, université de Caen, 2010.

[39] C.Peltz. « Web Services Orchestration and Choreography ». Computer, 26(10), 2003.

[40] C.Dumez, « Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en oeuvre de services Web composés », thèse de doctorat Université de Technologie de Belfort-Montbéliard, 2010.

[41] C.Lopez-velasco, « Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation », thèse de doctorat, université JOSEPH FOUR IER, 2008

[42] C.Peltz, . «Web Services Orchestration and Choreography ». IEEE Computer, 2003.

[43] D.Austin, A.Barbir, et all. « Web Services Choreography Requirements W3C Working Draft » . Technical report, World Wide Web Consortium (W3C), 2004.

[44] D.Austin, A.Barbir, et all. « Web Services Choreography Requirements ». W3C Working Draft, 2004.

[45] H.Ouassila. «Composition sémantique des services web dans un contexte d'ebXML»,thèse Doctorat. 2011

[46] Y. Charif , N. Sabouret, « Coordination in Introspective MultiAgent Systems », in Proc. the International Conference on Intelligent Agent Technology (IAT'07), Silicon Valley, California, USA, 2007.

[47] W.Sellami, « Une approche formelle pour la vérification des propriétés non fonctionnelles d'orchestration des services Web » , Mémoire de Magister de recherche: Université de Sfax, 2011.

[48] W. Jiang, S. Hu, et all, « Continuous Query for QoS-Aware Automatic Service Composition », in IEEE 19th International Conference on Web Services (ICWS'12), Honolulu HI.USA, 2012.

[49] J.Rao , X.Su. « A survey of automated Web service composition methods », Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, California, USA, 2004.

[50] M.Fluegge, G.Santos, et all, « Challenges and techniques on the road to dynamically compose Web services », Proceedings of the 6th international conference on Web engineering, (New York, USA), ACM, 2006.

[51] A.Mohammed Omer, « A framework for Automatic Web Service Composition based on service dependency analysis », thèse de doctorat, Technical University of Dresden Germany, 2011.

[52] T. Yu, K.-J. Lin , Y. Zhang, « Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints », ACM Transactions on The Web Journal (TWEB'07), 2007.

[53] T.Andrews, F.Curbera, et all. « Business Process Execution Language for Web Services », Version 1.1. IBM Specification ,2003.

[54] Kavantzas, N.Burdett, et all. « Web Services Choreography Description Language », W3C ,2005.

[55] D.Martin, D., M.Burstein, et all , « OWL-S: Semantic Markup for Web Services ». W3C, 2004.

[56] F.Leymann , « Web Service Flow Language ». IBM, 2001.

[57] A.arkin, S.askary, et all ,« Web service choreography interface (WSCI) ». Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems ,2002.

[58] R.Milner, « Communicating and Mobile Systems: The _-Calculus ». Cambridge University Press, Cambridge, UK, 1999.

[59] A.ARKIN, « Business process modeling language » . BPMI. org ,2002.

[60] A.banerji, C.bartolini, et all, « Web services conversation language (wscl) » . W3C ,2002.

[61] P.Châtel, « Une approche qualitative pour la prise de décision sous contraintes Non-fonctionnelles dans le cadre d'une composition agile de services », THÈSE Doctorat de l'université Pierre et Marie Curie – Paris 6 (spécialité informatique) ,2010

[62] M.Keidl,, A.Kemper, « Towards Context-Aware Adaptable Web Services ». NewYork, USA, 2004.

[63] S.Chemaa, « Une approche de composition de services Web à l'aide des Réseaux de Petri orientés objet »,Thèse de Doctorat en informatique,université abdelhamid mehri, constantine 2,2014.

[64] T.Zhang, « une médiation de composition dynamique de Services Web dans des environnements ubiquitaires »,Thèse Docteur de l'Université Paris 13 Nord,2015.

[65] D.Hanane, L.Siham, « Une approche formelle pour la composition des services web », thèse master academique université echahid hamma lakhdar - el-oued,2015.

[66] OASIS Standard. « WS-BPEL », 2007.

[67] M.alaoui-selsouli, « test unitaire de processus bpel : génération orientée chemins de cas de test »,université du québec à montréal,2010.

[68] Dr. Djamel Benmerzoug, « Composition de Services Web »,Département TLSI Faculté des NTIC ,Université Constantine 2 – Abdelhamid Mehri,2016

[69] C.Dechsupa, W.Vatanawood, et all, « Formal Verification of Web Service Orchestration Using Colored Petri Net » ,Proceedings of the International MultiConference of Engineers and Computer Scientists,Hong Kong,2016

[70] Baojun Tian,Yanlin Gu , « Formal Modeling and Verification for Web Service Composition » ,journal of software, 2013.

[71] YanPing Yang, QingPing Tan, et all, « Transformation BPEL to CP-Nets for Verifying Web Services Composition »,Computer College of National University of Defense Technology Changsha, Hunan, P.R.China,2005.

[72] Y.Yang, Q.Tan, et all, « Verifying Web Services Composition: A Transformation-Based Approach » ,Computer College of National University of Defense Technology,2005.

[73] Y.Yang,Q.Tan, et all, « Verifying Web Services Composition Based on Hierarchical Colored Petri Nets » ,Bremen, Germany.,2005

[74] HKang, X.Yang, S.Yuan, « Modeling and Verification of Web Services Composition based on CPN » ,International Conference on Network and Parallel Computing - Workshops,2007.

[75] Z.Zhao-li , H.Fan , et all, « A colored Petri net-based model for web service composition » , Shanghai Univ (Engl Ed),2008

[76] X.Yi , Krys ,et all, « A CP-nets-based Design and Verification Framework for Web Services Composition » ,Proceedings of the IEEE International Conference on Web Services,2004.

[77] C.Yong-shang, W.Zhi-jian, et all, « Modeling and verifying composite semantic Web service based on colored Petri nets » ,Sixth International Conference on Advanced Language Processing and Web Information Technology, 2007.

[78] N.Yang, Y.Wang, « Test Case Generation of Web Service Composition based on CP-nets » ,JOURNAL OF SOFTWARE, 2014.

[79] W.Maung-Maung, H.Aung-Aung, « Colored Petri-Nets (CPN) based Model for Web Services Composition » ,International Journal of Computer & Communication Engineering Research (IJCCER) ,2014.

[80] E.Duris, « Les bases de la programmation orientée objet avec Java »,2016.

[81] « Débuter avec Eclipse »,universite Lille Nord de France,CP2 – JAVA ,2017.

[82] M.Kristensen, « Coloured Petri Nets »,Department of Computer Engineering Bergen University College, NORWAY,2010.

[83] https://fr.edrawsoft.com/.

[84] M.Juric, B.Mathew, and P.Sarang. « Business Process Execution Language for Web Services 2nd Edition ». Packt Publishing, 2006.

[85] https://research.linagora.com/display/easiestdemo/Travel+Agency/, fichier BPEL dans ce site,2018.

[86] A.U, « BPEL, Module SOA », Ecole supérieur privée d'ingénierie et de Technologies,esprit,2016.

[87] www.w3.org , http://www.w3.org/TR/ws-arch/.

[88] W.A.c.BIHI,« Approche orientée modèles pour la vérification et l'évaluation des performances de l'interopérabilité et l'interaction des services »,these de doctorat,l'universite de technologie de belfort-montbeliard,2012 .

[89] C.Dumez. « Approche dirigée par les modèles pour la spécification, la v´erification formelle et la mise en oeuvre de services Web composes ».thèse Doctorat de l'Université de Technologie de Belfort-Montbéliard.2010

[90] M.Lifa ,« Thème Une approche formelle pour la composition des services web », université echahid hamma lakhdar - el-oued ,2014.

[91] B.Franck , K.Maria. « Models and verification of BPEL », 2006.

[92] N.Lohmann, R.Dijkman, et all. « Petri Net Transformations for Business Processes - A Survey. Transactions on Petri Nets and Other Models of Concurrency », 2008.

[94] R.Hamadi , B.Benatallah. « A petri net-based model for web service composition », Darlinghurst, Australia, Australia, 2003.

[95] F.Puhlmann , M.Weske. « Using the pi-calculus for formalizing workflow patterns ». Lecture Notes in Computer Science , 2005.

[96] J.Zhang, K.Chang,et all. « A Petri Net-Based Specification Model towards Verifiable Service Computing ».2012

[97] Y.Yan , P.Dague. « Modeling and diagnosing orchestrated web service processes ». IEEE International Conference on Web Services, 2007.

[98] Object Management Group OMG. « Unified modeling language 2.2 specification ».2009.

[99] J.Amsden, T.Gardner, et all. « Draft UML 1.4 profile for automated business processes with a mapping to BPEL », 1.0. Technical report, IBM, 2004.

[100] R.Bendraou, M.Gervais, et all. « Uml4spm : a uml2.0-based metamodel for software process modelling », Berlin, Heidelberg, 2005.

[101] C.Dumez, J.Gaber, et all. « Model-driven engineering of composite web services using uml-s »., New York, NY, USA, 2008.

[102] S.White. « Process modeling notations and workflow patterns ». Workflow Handbook, , 2004.

[103] Bunke. « Graph matching : Theoretical foundations, algorithms, and applications », 2000.

[104] J. Mendling ,J.Ziemann. « Transformation of bpel pro- cesses to epcs" In Proc ». of the 4th GI Workshop on Event-Driven Process Chains (EPK2005), 2005.

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

[106] P.Peyre, « Introduction à la vérification structurelle des réseaux de Petri et des réseaux de haut-niveau Master SAR », Module VFSR,2011.

[107] J.Comet, « Les réseaux de Petri pour la simulation de systèmes biologiques » , Département Génie Biologique ,2014.

[108] K.Labadi, « Contribution à la modélisation et à l'évaluation de performances des systèmes logistiques à l'aide d'un nouveau modèle de réseaux de Petri stochastiques »,thése de doctorat de l'UTT.2009

[109] L.Lahouaoui, « Les automates programmables industriels »,Cours et TD Master 2 Filière Electronique Université Mohamed Boudiaf de Msila,2017

[110] J.Eloundou, « Modélisation Multi-contraintes d'un Système de Production Flexible »,thése de doctorat Spécialité Informatique Préparée au sein de l'INSA de Rouen,2016

[111] D.Tissilia, « Un Cadre Formel pour La Modélisation et L'analyse Des Agents Mobiles » ,Thèse de Doctorat en Sciences en Informatique Université Mentouri Constantine,2013

[112] « Chapitre 4 LES RESEAUX DE PETRI »,Academie pro,2015

[113] J-C.Geffroy , « Chapitre 10 Introduction aux Réseaux de Petri », cyranac.free,2014

[114] M.Benmoussa, « Une approche basé sur la transformation du graphe sous ATom3 pour l'intégration des deux outils de réseau de petri »,thése Magister université echahide hamma lakhdar- el-oued,2015

[115] C.Agon ,R.Demangeon, « Cours 1 - Introduction à la concurrence Modélisation avec des résaux de Petri,Paradigmes de Programmation Concurrente » ,UPMC Paris Universitas,2017

[116] « Master SAR – Modélisation de systèmes répartis (NI 405) » ,Master d'Informatique spécialité « SAR », 2012.

[117] M.Bourcerie, « la modelisation des systemes de production par reseaux de petri », Magister recherche Systèmes Dynamiques et Signaux,2011

[118] H.Haiouni, « Approche mixte de modélisation par Réseaux de Petri et SMA » ,thése de Magister en informatique universite mentouri de constantine,2010

[119] W.Maher , « Using Colored Petri Nets for Modeling E-Commerce Services» ,Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series,2014

[120] L.Mounir,« Modelisation et Test Fonctionnelde L'Orchestration de Service Web »,these de doctorat ,l'Universite d'evry-val d'essonne,2012