République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed KHIDER - BISKRA
Faculté des Sciences et de Sciences de l'ingénieur
Département d'Informatique

N° d'ordre : ……...
Série : .....

Mémoire

En vue d'obtention du diplôme de Magister en informatique
Option: Systèmes d'Informations Avancés et Intelligence Artificielle

Réseaux de Petri Avancés pour la modélisation des Agents Mobiles pour la Synchronisation des Workflows

Réalisé par :
Mr. Messaoud MEZATI

Membres de jury :

Président	: Mr. Noureddine DJEDI	Professeur	Université de Biskra.
Rapporteur	: Mr. Allaoua CHAOUI	Maître de conférences	Université de Constantine
Examinateur	: Mr. Mohamed Chaouki BABAHENINI	Maître de conférences	Université de Biskra.
Examinateur	: Mr. Cherif FOUDIL	Maître de conférences	Université de Biskra.

.

Année Universitaire: 2007 / 2008

Remerciements

Je tiens à exprimer mes sincères et chaleureux remerciements à mon encadreur Monsieur le professeur **Allaoua CHAOUI** qui m'a suivi et dirigé.

Je remercie tous les membres du jury qui m'ont fait l'honneur d'évaluer ce travail.

Je présente tous mes remerciements à l'ensemble des personnes de l'institut d'informatique.

En particulier, je dois citer monsieur **M^ed C. Babahenini** le chef du département d'informatique et tous les enseignants.

Je remercie les éléments de ma promotion de magistère.

J'exprime ma profonde reconnaissance à tous ceux qui m'ont aidé à l'élaboration de ce mémoire de prés ou de loin.

ملخـــــص

تمثل التجارة الالكترونية واحدا من موضوع ما يعرف بالاقتصاد الرقمي (Economie) Numérique حيث يقوم الاقتصاد الرقمي على حقيقتين :- التجارة الإلكترونية و تقنية المعلومات (Technologie de l'information). فتقنية المعلومات أو صناعة المعلومات في عصر الحوسبة والاتصال هي التي خلقت الوجود الواقعي والحقيقي للتجارة الإلكترونية باعتبارها تعتمد على الحوسبة والاتصال ومختلف الوسائل التقنية للتنفيذ وإدارة النشاط التجاري.

وقد كان اهتمامنا في هذا العمل منصبا على كيفية استغلال العميل الجوال (Agent mobile) لينوب عن العميل الفعلي في التجارة الالكترونية من أجل تحقيق أفضل توافق بين الممونين (Fournisseurs) و المستهلكين (Consommateurs)، حيث أن هذا الأخير يقوم بإعداد تقرير شامل لمهمته، على غرار النموذج الكلاسيكي الذي يعتمد على تصفح المواقع والبحث المجهد وسط العديد من نتائج البحث الغير مجدية غالبا.

Résumé

Le commerce électronique fait partie de ce qu'on appelle (Economie numérique), car ce dernier repose sur deux axes fondamentaux : le commerce électronique et la technologie de l'information.

La technologie de l'information a du très grand pacte sur le développement du commerce électronique parce qu'elle permet d'utiliser les derniers techniques du traitement des informations relatives aux différentes activités dont les activités commerciales fait partie.

Dans notre travail nous avons exploité les caractéristiques des agents mobiles pour remplacer les agents réels qui fissent du commerce afin d'aboutir à un niveau élevé de correspondance entre les fournisseurs et les consommateurs tout en discutant l'apport de l'utilisation des agents mobiles par rapport à la navigation classique qui est généralement inutile.

Introduction générale

L'objectif du workflow est d'automatiser des processus dans des organisations. Un processus est un enchaînement coordonné de tâches suivant un certain schéma et aboutissant à un résultat bien déterminé (tel que le suivi de dossier médical, le remboursement des frais ou bien les procédures de recrutement). Durant l'exécution d'un processus, des informations, des formulaires ou des documents sont partagés ou sont transmis d'un poste de travail à un autre pour y être traités.

Ce travail vise à faire une intégration de la théorie de réseaux de Petri avec les approches de la conception du système. La motivation de cette intégration est de faire un pont entre le traitement formel de réseaux de Petri et l'approche orienté objet. L'approche orienté objet offre la spécification et le prototypage d'un système complexe.

Le mémoire organisé en quatre chapitres faisant un tour sur le domaine du workflow interorganisationnel et les agents mobiles. Après une présentation du modèle G-net avec toutes ses caractéristiques (l'héritage etc). Ainsi, nous décrivons un modèle qui offre la modélisation des agents mobiles par G-net dans les workflows interorganisationnels et nous présentons les résultats de son implémentation sur un exemple.

Le premier chapitre expose les concepts de base du workflow, agent mobile, les réseaux de Petri et précise ensuite progressivement notre problématique : la modélisation des Workflows Interrganisationnels.

Le deuxième chapitre introduit les notions relatives au modèle G-net et montre comment elles peuvent être utilisées pour modéliser les systèmes d'information distribués, le Workflow Interorganisationnel en particulier.

Le troisième et quatrième chapitre constituent le cœur de notre contribution. Dans le chapitre 3, nous expliquons la mise à jour appliquée sur le modèle G-net pour offrir une plate-forme permettant de supporter la modélisation des agents mobiles. Cette mise à jour est assurée par la notion d'héritage des classes. Nous présenterons dans le chapitre 4 les résultats expérimentaux de l'implémentation du modèle G-net orienté agent sur un exemple de vote.

Nous terminons par une conclusion dans laquelle nous dressons le bilan de cette intégration et présentons les perspectives futures de notre travail de recherche.

1 Introduction

Le workflow interorganisationnel (WIO) est une problématique de recherche actuelle qui envisage la coopération de plusieurs processus issus d'organisations réparties, autonomes et hétérogènes. Dans ce chapitre, nous offrons la définition des workflow et de l'algèbre d'itinéraire d'agents qui est utilisé pour la spécification d'itinérant d'agent du workflow interorganisationnel (WIO). En plus, nous présentons les caractéristiques du réseau de Petri, du workflow net et WIO net**.**

2 Définition d'algèbre d'itinéraire

On offre une description brève de l'algèbre d'itinéraire avec un modèle orienté objet pour les agents, ou un agent comme une instance de classe défini comme suivant [1] :

Agent mobile = états + action + mobilité.

Aussi, un agent possède la possibilité de cloner lui-même, c'est-à-dire de créer des copies de mêmes états et de mêmes codes de lui-même. Un agent peut faire une communication pour la synchronisation des mouvements, et le code d'agent est exécutable dans chaque place qu'il visite. On A, Q et P définis comme des ensembles d'agents, d'actions et de places et respectivement.

L'itinéraire noté par *I* représente l'activité nulle, atomique, parallèle, séquentielle, non déterministe, conditionnelle, comportement non déterministe et à syntaxe suivant :

$I=0 \mid A_p^a \mid (I \| \oplus I) \mid (I.\ I) \mid (I \mid I) \mid (I:\ \sqcap\ I))$

Où, A∈ **A**. a∈ **O**. p∈ **P**. ⊕ c'est l'opérateur. Après qu'une opération parallèle causant clonage ; on doit faire le clonage. Par la suite, on doit faire la combinaison ou l'élimination des agents clonés pour avoir un seul agent. Et Π c'est l'opérateur qui retourne une valeur logique pour le modèle conditionnel. On assume que tous les agents itinéraires ont une place débutante (ce qu'on appelle la maison des agents). On décrit le sens des opérations informelles attribuées à un itinéraire *I*.

2.1 Le mouvement d'agent (A_p^a)

(A_p^a **)** veut dire,'' l'agent *A* se déplace vers *p* et accomplit l'action e ''. Cette expression implique un seul agent, un seul déplacement et une seule action dans la destination.

2.2 Une composition parallèle (''||'')

Les deux expressions composées par ''||'' sont exécutées en parallèle. Par exemple, ($A_p^a \parallel B_q^b$) veut dire que l'agent *A* et *B* sont exécutés simultanément. Le parallélisme implique le clonage des agents. Par exemple, pour exécuter l'expression ($A_p^a \parallel A_q^b$). Où $p \neq q$, il faut appliquer le clonage puisque l'agent *A* exécute les actions dans *p* et *q* de façon parallèle.

2.3 Une composition séquentielle (''.'')

Les deux expressions composées par un opérateur ''.'' sont exécutées séquentiellement. Par exemple ($A_p^a . A_q^b$) veut dire le déplacement de l'agent *A* à la place *p* pour accomplir l'action *a* et puis il se dirige vers la place *q* pour accomplir l'action *b*.

2.4 Le non déterminisme indépendant (''|'')

Un itinéraire de forme *(I | J)* est utilisé pour exprimer le choix non déterminé'' | '' quelque soit le choix, il exécute *I* ou *J*.

Si l'agent $I \cap$ agent $J \neq \phi$, aucun clonage n'est assumé, c'est-à-dire, *I* et *J* sont traités indépendamment.

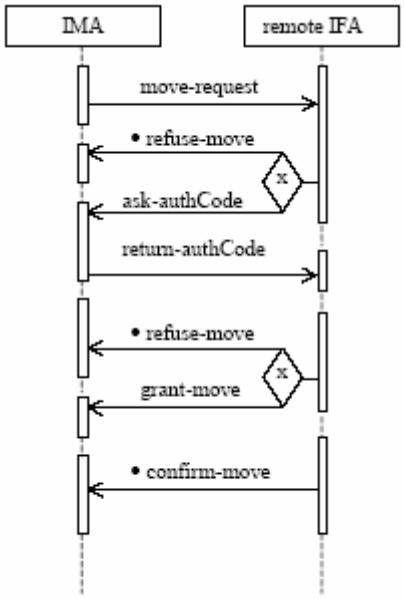

Figure 1: le protocole de communication entre AMI et AFI [13]

On présente dans la figure 1 la communication entre deux agents (Agent mobile et agent facilitateur). La première étape est l'envoi par l'agent mobile intelligent (AMI) d'une demande de migration vers l'Agent Facilitateur Intelligent (AFI). L'AFI peut alors choisir de répondre à l'AMI en refusant sa migration ou en demandant à l'AMI le code d'autorisation. Si l'AFI se refuse à distance de migration fondée sur la limitation des ressources ou d'autres

raisons. Le protocole prend fin, sinon l'agent AFI attend que l'AMI du code d'autorisation soit fourni. Si l'AMI à l'autorisation de code correctement fourni, l'agent AFI peut accorder à l'AMI pour la migration si elle est personnalisable ou de refuser la migration autrement. Encore une fois, si l'agent AFI refuse la migration de AMI, le protocole se termine sinon, un message de confirmation sera fourni ultérieurement [13]

3 Le Workflow traditionnel

Cette section se base sur les travaux de la **W**ork**F**low **M**anagement **C**oalition (WFMC) et ceux de la **W**orkflow **P**atterns **I**nitiative (WPI). La WFMC est une association mondiale de normalisation dans le domaine du workflow, elle est à l'origine de la terminologie associée au workflow et d'une architecture logicielle de référence. La WPI, créé sur l'initiative de van der Aalst et Arthur ter Hofstede, s'est concentrée sur des aspects plus sémantiques, notamment sur la définition des exigences en terme de pouvoir d'expression des langages de description de workflow. Ces exigences, exprimées à l'aide des motifs appelés «patterns» décrivant des schémas récurrents.

3.1 Le Workflow

Le workflow se définit comme l'automatisation de tout ou d'une partie d'un processus d'organisation. Il définit un ensemble de tâches composantes, leur coordination, l'information et les acteurs impliqués dans chaque tâche. L'exemple de processus qui va servir de fil conducteur à cet article est le processus de *relecture des articles dans une conférence* (sélection des membres du comité de programme, répartition des articles, lecture et évaluation en parallèle avec éventuellement sous-traitance du travail…). Ce processus fait circuler des articles, des grilles d'évaluation et implique les membres de comité du programme, notamment son président.

L'objectif du workflow est d'automatiser des processus dans des organisations [7].

Un *processus* est un enchaînement coordonné de tâches suivant un certain schéma aboutissant à un résultat bien déterminé (tel que le suivi d'un dossier médical, le remboursement de frais ou les procédures de recrutement). Durant l'exécution d'un processus, des informations, des formulaires ou des documents sont partagés ou transmis d'une poste de travail à un autre pour être traités.

Trois modèles sont généralement utilisés pour décrire la structure du processus, les ressources et les informations nécessaires à sa mise en œuvre (Figure 1).

Le *modèle organisationnel* structure les ressources en rôles et leur attribue des autorisations pour réaliser certaines tâches composant le processus. Le *modèle informationnel* décrit la structure des formulaires, des documents, et des données qui sont utilisées et produits par le processus. Le *modèle de processus* définit les tâches composantes, leur coordination, l'information et les ressources impliquées dans chaque tâche.

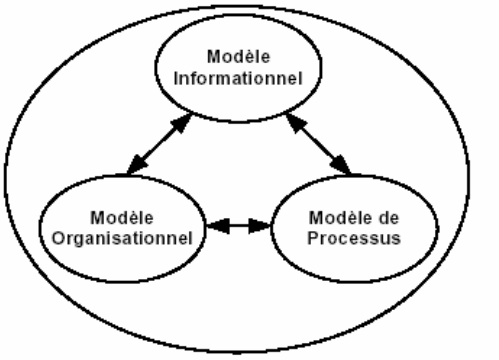

Figure 2 : Les trois modèles conceptuels d'un workflow [7]

3.2 Cycle de vie d'un workflow

Le cycle de vie d'un workflow (Figure 2) est composé essentiellement de deux étapes [9].

1. La première étape est consacrée à la modélisation (ou l'édition) d'un processus, c'est-à-dire à la conception d'un schéma de processus. On la retrouve également sous le vocable anglais "Build time". Dans la plupart des Systèmes de gestion Workflow (SGWf), un environnement graphique est proposé pour cette étape.

2. La deuxième étape est consacrée à l'instanciation d'un schéma de processus appelé "cas" et à l'exécution de celui-ci. L'exécution peut faire appel à l'utilisateur et/ou à des applications externes. Cette étape est connue sous le vocable anglais de "Run time".

Le composant logiciel chargé de cette étape s'appelle un moteur de workflow.

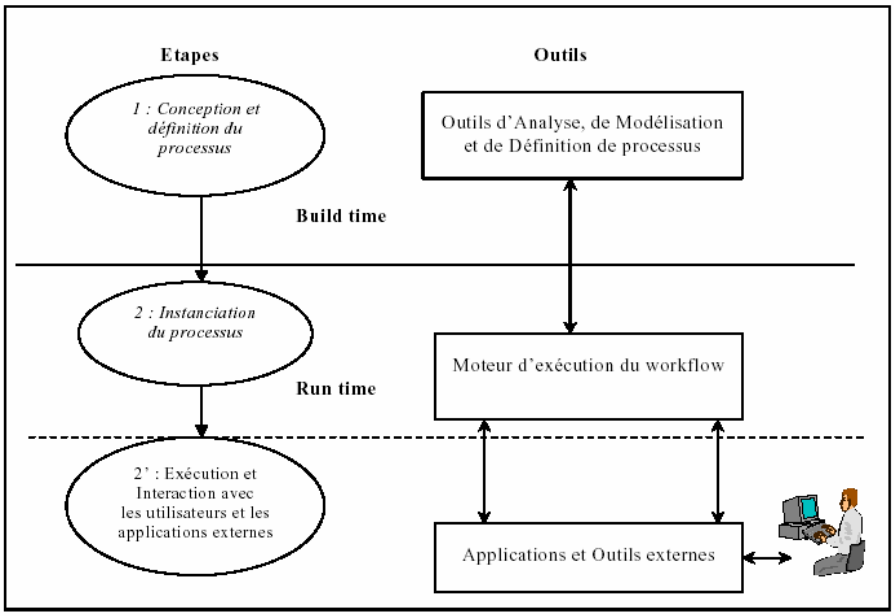

Figure 3 : Cycle de vie d'un Workflow [7]

Un système de gestion de Workflow (SGWf) se définit comme un système qui définit, implémente et gère l'exécution du processus workflow à l'aide d'un environnement logiciel fonctionnant avec un ou plusieurs moteurs de workflow. Il est capable d'interpréter la définition d'un processus, de gérer la coordination des participants et d'appeler des applications externes.

Pour permettre l'émergence de standards dans le monde des systèmes des workflows, des éditeurs de logiciels, des laboratoires de recherches et des utilisateurs de ces systèmes ont créé le consortium Workflow Management Coalition (WfMC). L'objectif de cette association est la promotion et le développement des systèmes de workflows. Pour cela, ils ont réalisé un glossaire afin d'unifier la terminologie, ils ont défini un modèle de référence tel qu'il est noté dans la figure3) centré autour du moteur d'exécution. Ce modèle présente 5 interfaces de standardisations [10]:

Interface 1 : Elle correspond à l'échange des modèles entre moteurs de workflows et les différents outils de modélisation de processus.

Interface 2 : Elle permet à des applications clientes de communiquer avec le moteur de workflows.

Interface 3 : Elle permet au système de workflows d'appeler des applications clientes.

Interface 4 : Elle permet l'interopérabilité entre moteurs de workflows.

Interface 5 : Elle correspond à l'interaction entre les applications d'administration et le moteur de workflows.

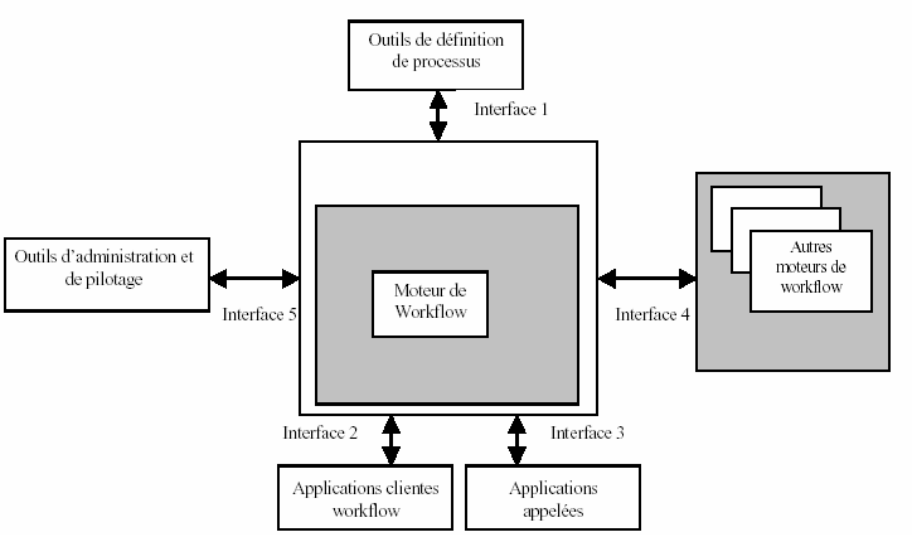

Figure 4: Architecture de référence d'un SGWF [10]

3.3 Types de Workflow

Dans la littérature, on distingue entre plusieurs types de Workflow:

- Les workflows de production ;
- Les workflows administratifs ;
- Les workflows *ad hoc* ou collaboratifs.

Cette distinction peut bien sûr être remise en cause et souvent, les workflows *ad hoc* sont distingués des workflows collaboratifs. Par ailleurs, il n'est pas toujours possible de distinguer les workflows administratifs des workflows collaboratifs.

3.3.1 Les workflows de production

Ce type de workflow est également appelé «workflow procédural ». La gestion de workflows de production a pour but d'optimiser des processus répétitifs et bien maîtrisés afin d'augmenter la productivité et la qualité des articles produits. Le niveau d'automatisation des tâches du processus est en général très élevé et le rôle du système de workflow est le plus souvent limité à la gestion des exceptions. Les tâches et leurs contraintes sont définies de manière rigoureuse, ce qui permet une planification déterministe des séquences de tâches, effectuée hors ligne, avant le démarrage de la production.

Ce workflow se doit surtout d'être rapide, fiable et sécurisé, car les chaînes de production tournent souvent sans arrêt, 24 heures sur 24. Sa flexibilité se limite à l'application de scénarios prédéfinis.

3.3.2 Les workflows administratifs

Comparés aux workflows de production, les workflows administratifs possèdent une définition moins précise des tâches à effectuer. Une tâche administrative n'est définie que dans ses grandes lignes. Ainsi le nombre de tâches par heure est souvent de dix à cent fois inférieur par rapport aux workflows de production.

La gestion de ce type de workflow doit être souple afin de prendre en compte les contraintes de flexibilité imposées par les procédures administratives.

3.3.3 Les workflows *ad hoc* ou collaboratifs

Les workflows *ad hoc* ne sont pas connus *à priori*, au niveau organisationnel. Ils sont crées individuellement à la demande de chaque utilisateur. Dans ce type de workflow, l'optimisation de l'exécution d'une instance de workflow est moins importante que la capacité de réagir de manière flexible aux modifications imprévues. Les processus sont définis d'une manière informelle. Les acteurs humains peuvent intervenir à tout instant dans la décision du cheminement du processus.

L'exécution d'une tâche dans le cas d'un workflow *ad hoc* se décompose en quatre étapes:

1. La négociation (Y a-t-il une ressource disponible ?)
2. L'accord (Oui, la ressource est disponible)
3. L'allocation (La ressource est utilisée pour l'exécution de la tâche)
4. La vérification (La tâche a-t-elle été correctement exécutée ?)

4 Les notions de base du Workflow Interorganisationnel (WIO)

On définit *Workflow Interorganisationnel* (WIO) comme une extension du workflow traditionnel, dont il se distingue par trois aspects ceci est possible puisqu'il y a une similarité entre les deux paradigmes objet et agent [7] :

- **la distribution**, qui concerne les processus composants, les informations et les ressources;
- **l'autonomie des organisations**, qu'il faut préserver : chaque organisation participante à un WIO doit pouvoir décider par elle-même les conditions de coopération c'est-à-dire : quand, comment et avec qui coopérer. De plus, elle doit pouvoir préserver le contrôle sur les tâches locales et maintenir leur confidentialité ;
- **l'hétérogénéité** qui, hormis les problèmes d'interconnexions concerne les différences de structure, syntaxe et sémantique des modèles décrivant les processus, les

informations et les ressources nécessaires à la mise en oeuvre de ces derniers (processus).

Le WIO peut se caractériser simplement de la manière suivante : N organisations partenaires mettent en commun leur workflow local, l'équation à résoudre est alors la suivante:

WIO = n Workflow locaux + 1 modèle de coordination

4.1 Typologie du WIO

La coordination dans le WIO ne peut pas être étudiée précisément si par ailleurs on ne fixe pas le type d'application visé. A identifié deux types d'applications conduisant à des exigences assez différentes en terme de coordination. Cette typologie permet de préciser d'avantage le contexte d'application du WIO et fournit en même temps un cadre d'étude de la coordination dans le WIO.

1. le ***WIO Serré*** correspond à une coopération structurelle entre des organisations. Cette coopération est basée sur une infrastructure bien établie entre les associés qui sont connus et ne changent pas. Ce genre de workflow interorganisationnel est nécessaire quand les organisations impliquées sont engagées dans une coopération à long terme et quand une forte interdépendance entre leur workflow existe. Cette interdépendance se manifeste par des interactions fréquentes entre les instances de workflow des différentes organisations.

2. le ***WIO Lâche*** correspond à une coopération occasionnelle avec des contraintes de coopérations structurelles libres où les organisations impliquées et leur nombre ne sont pas prédéfinis (l'univers est dynamique).

4.2 Concepts d'interopérabilité pour les workflows inter organisationnels

Le contrôle et la gestion d'un workflow impliquant plusieurs participants peuvent être distribués de différentes manières [9]:

4.2.1 Le partage des capacités

Les tâches sont allouées aux participants selon des règles définies, par une instance de gestion centrale. Cette méthode nécessite un serveur unique mais l'exécution des tâches peut s'effectuer de manière distribuée.

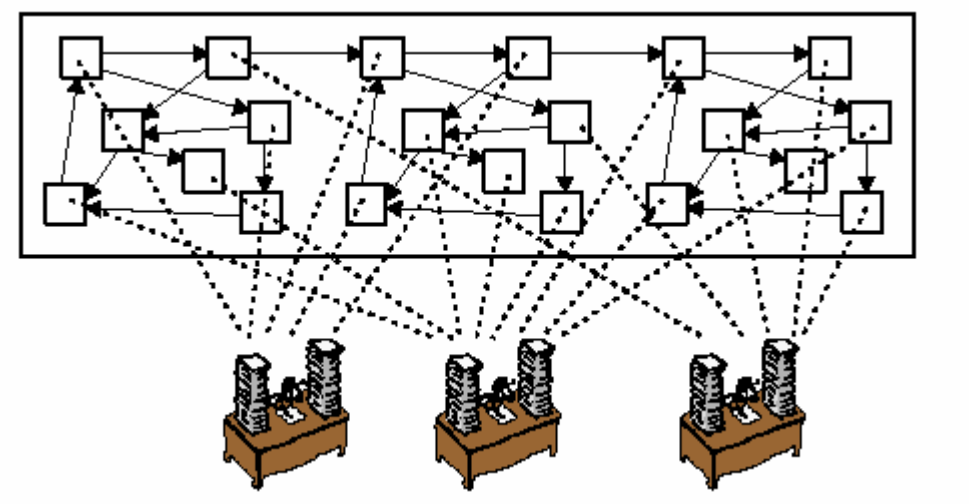

Figure 5 : Le partage de capacités [9]

4.2.2 L'exécution en chaîne

Le workflow est découpé en plusieurs parties qui sont exécutées séquentiellement par les différents participants. Dès que l'un d'eux finit sa partie, il cèdera le contrôle au participant suivant. Contrairement aux systèmes utilisant le partage de capacités, l'exécution *et* le contrôle sont ainsi distribués.

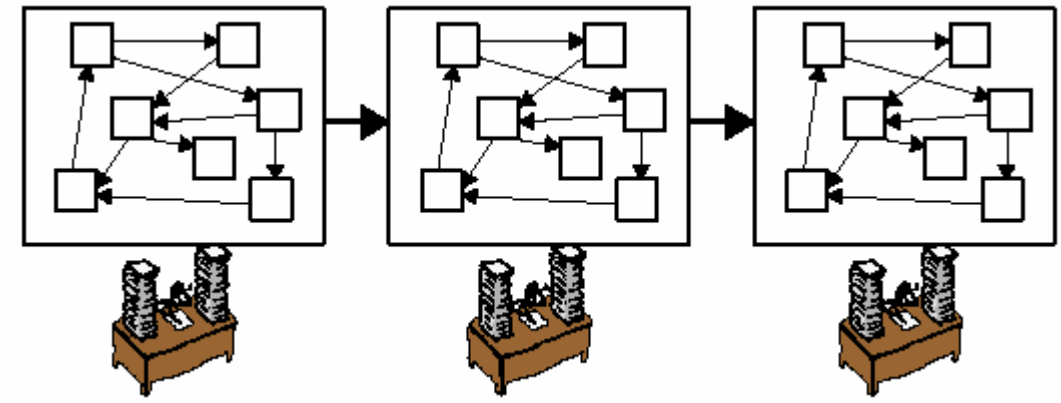

Figure 6 : L'exécution en chaîne [9]

4.2.3 La sous-traitance

La gestion du workflow est distribuée de façon hiérarchique entre les participants. Le niveau supérieur cède le contrôle d'une partie de son workflow au sous-traitant. Celui-ci exécute les tâches et rend le contrôle au niveau supérieur après avoir fini l'exécution.

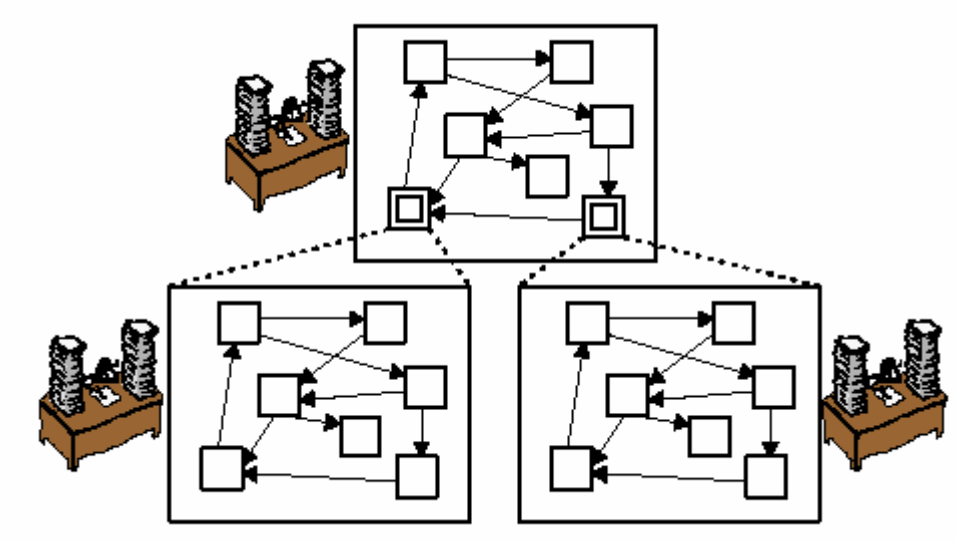

Figure 7 : La sous-traitance [9]

4.2.4 Le transfert d'instances de workflow (en anglais « case transfer ») – Chacun des participants possède une copie du schéma complet du workflow. Seules les instances du workflow («workflow cases ») sont transférées entre les participants afin d'équilibrer le partage de charges ou parce qu'une tâche doit être exécutée par un participant précis.

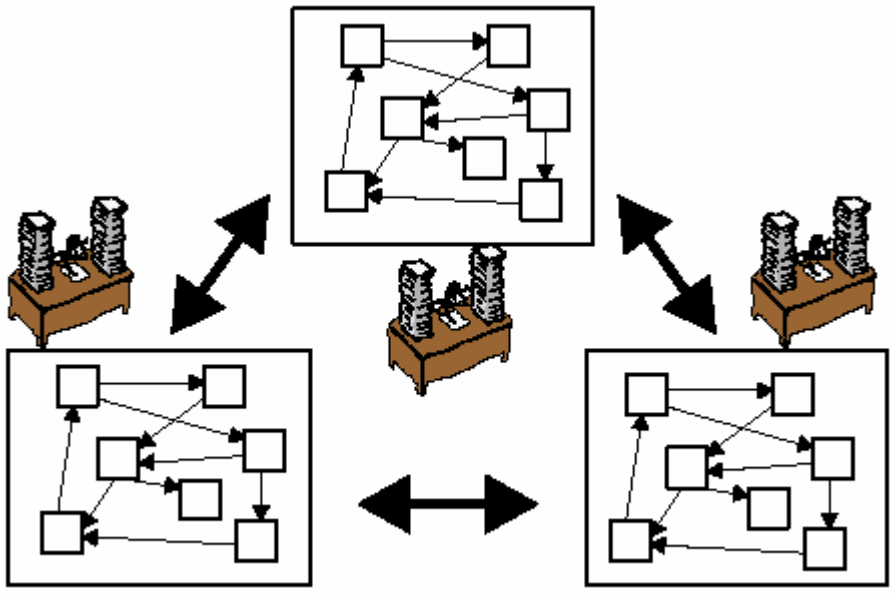

Figure 8 : Transfert d'instances de workflow [9]

4.2.5 Le couplage souple (en anglais « loosely coupled ») – Des parties d'un schéma workflow sont distribuées entre les participants et peuvent être actives en parallèle. Les participants ne connaissent pas toujours le schéma et l'état des instances de workflow de leurs partenaires. La communication entre les instances se fait selon un protocole commun.

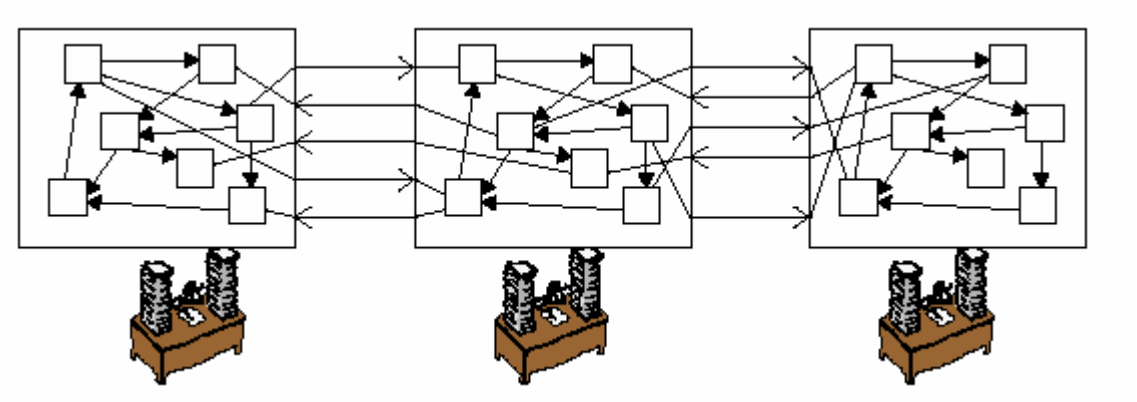

Figure 9 : Couplage souple [9]

4.2.6 L'approche « Public vers Privé »

Un schéma global est disponible pour tous les participants du workflow distribué. Chaque participant crée son propre schéma à partir d'un schéma public en respectant les contraintes qui y sont imposées. Le schéma privé est donc une sous-classe d'une partie du schéma public.

En ce qui concerne l'utilité des différentes approches pour le contexte spécifique de l'‘entreprise virtuelle, il est évident que le contrôle centralisé nécessaire aux systèmes de partage de capacités, pose un problème organisationnel. L'exécution en chaîne est uniquement adaptée à des processus ordonnés séquentiellement, ce qui rend cette approche très contraignante. Dans la réalité des entreprises virtuelles il n'est souvent pas possible de transmettre le contrôle entier d'une partie de l'instance d'un workflow à un participant.

Les approches *à priori* adaptées à la réalisation d'un système de gestion informatique de workflow sont donc la sous-traitance, le transfert d'instances, le couplage souple et l'approche « Public vers Privé ». Cette dernière est la seule qui a introduit un formalisme de protection des connaissances confidentielles.

5 Les réseaux de Petri

Les Réseaux de Petri ont été inventés par Carl maria Petri au début des années soixante. Des travaux ultérieurs ont permis de développer les Réseaux de Petri comme un outil de modélisation pour les systèmes à variables logiques puisque ceux-ci sont un cas particulier des systèmes à variables discrètes.

Un caractère très intéressant est que les modèles Réseaux de Petri sont sous la forme d'une représentation mathématique *graphique.* Ce point est important car le fait d'écrire sous forme graphique un modèle plutôt que sous forme d'équations pouvant permettre de le rendre lisible par des personnes dont la formation scientifique n'est pas forcement poussée.

5.1 Notations et définitions

Un Réseau de Petri (RdP) est un graphe orienté comprenant deux types de sommet [8]:

– les places

– les transitions

Celles-ci sont reliées par des arcs orientés. Un arc relie soit une place à une transition soit une transition à une place mais jamais une place à une place ou une transition à une transition. Tout arc doit avoir à son extrémité un sommet (place ou transition). Un exemple de Réseaux de Petri est représenté au Figure 10. A chaque place et transition, un nom peut être associé par exemple sur Le RdP de la Figure 10 ; les places sont nommées P_1, P_2, P_3 et P_4 et les transitions T_1, T_2 et T_3. T_1 est reliée à P_1 par un arc orienté T_1 vers P_1 : on dit que P_1est en sortie ou en aval de T_1. P_1 est reliée à T_2 par un arc orienté de P1vers T_2 : on dit que T_2 est en sortie de P_1. De même on peut dire que P_1 est en entrée ou en amont de T_2 .La place P_3 est en entrée de T_1.

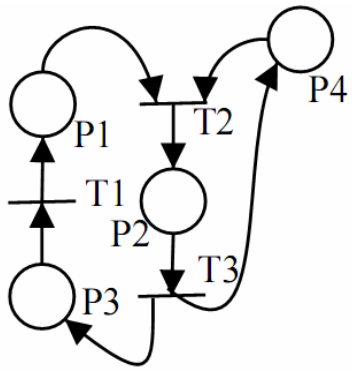

Figure 10 : Exemple d'un réseau de Petri

Cas particuliers

Transition source pas de place en entrée de la transition.

Transition puits pas de place en sortie de la transition.

Une place correspond à une variable d'état du système qui va être modélisé et une transition à un événement et/ou une action qui va entraîner l'évolution des variables d'état du système. A un moment donné, une place contient un certains nombres de marques ou jetons qui va évoluer en fonction du temps : ils indiquent la valeur de variable d'état à cet instant.

Quand un arc relie une place à une transition, cela indique que la valeur de la variable d'état associée à la place influence l'occurrence de l'événement joint à la transition.

Quand un arc relie une transition à une place, cela veut dire que l'occurrence de l'événement associé à la transition influence la valeur de la variable d'état associée à la place.

5.1.1 Marquage

Le marquage du réseau de Petri à un instant donné est un vecteur colonne dont la valeur de la $I^{éme}$ composant est le nombre de marquage q dans la place Pi à ce moment.

Le franchissement d'une transition conduit à un changement du marquage. Le passage du marquage M_k à celui marquage M_j par franchissement de la transition T_j est noté : $M_k/T_j > M_j$. Le nombre de marquages dans la place P_i pour le marquage M_k est noté $M_k(P_i)$.

A partir d'un même marquage, il peut être possible de franchir plusieurs transitions, menant ainsi à des marquages différents. L'ensemble des marquages accessibles à partir du marquage M_0 est l'ensemble des marquages obtenus à partir de M_0 par franchissements successifs d'une ou plusieurs transition(s). Cet ensemble est noté $*M_0$.

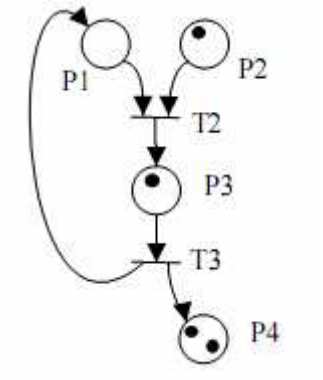

Figure 11 : Atelier de coupe simplifié

Le RdP représenté en Figure 11 une présente version simplifiée de l'atelier de coupe.
L'ensemble des marquages accessibles est donc défini par : $*M_0=\{M_1,M_2,M_3\}$.

5.1.2 Séquence de franchissements

Une séquence de franchissement est une suite de transitions qui sont successivement franchies pour passer d'un marquage à un autre. Dans l'exemple de l'atelier précédent:

$$M_0 / S > M_3 \ avec \ S = T_3 T_2 T_3.$$

Les séquences de franchissement sont décrites à l'aide des notations présentées dans le tableau 3.1. Par exemple, le franchissement de la transition T_1 suivi du franchissement de la transition T_2 ou du franchissement de la transition T_3 se note:

$$T_1(T_2 + T_3) = T_1T_2 + T_1T_3$$

Le franchissement de la transition T_1 suivi du franchissement de la transition T_3 ou le franchissement de la transition T_2 suivi du franchissement de la transition T_3 se note:

$$(T_1 + T_2)T_3 = T_1T_3 + T_2T_3$$

5.2 Propriétés

Dans cette partie ; on va présenter les propriétés du formalisme de réseau de Petri.

5.2.1 Réseaux de Petri borné et Réseaux sauf

Une place P_i est *bornée pour un marquage initial* M_0 si pour tout marquage accessible à partir de M_0, le nombre de marques dans P_i reste borné/ Elle est dite k-bornée si le nombre de marques dans P_i est toujours inférieur ou égal à k. Un RdP marqué est (k) borné si toutes ses places sont (k) bornées.

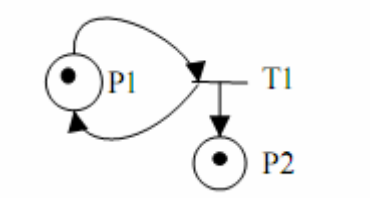

Figure 12 : RdP non borné

Un RdP marqué peut ne pas être borné : Dans l'exemple représenté figure 11, la transition T_1 admet la place P_1 comme unique place d'entrée. La place P_1 a une marque : transition T_1 est franchissable. Comme P_1 est aussi place de sortie de T_1, le franchissement de T_1 a change le marquage de P_1. La transition T_1 est donc franchissable en permanence et peut donc être franchie un nombre de fois infinies. Chaque franchissement de T_1 ajoutant une marque dans la place P_2, le marquage de celle-ci peut donc tendre vers l'infini.

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

Dans le cas où un RdP marqué modélise un système logique, chaque place du RdP correspond à un élément du vecteur d'état du système. Celui-ci ne peut prendre comme valeur

que 0 ou 1, soit zéro ou une marque dans la place correspondante. Tester si le système logique est cohérent revient alors à vérifier que son modèle RdP est sauf.

Il est à noter qu'il faut bien préciser que les propriétés précédentes dépendent du marquage initial. Par exemple, pour le RdP représenté Figure 11, avec *M_0=[0 N], le RdP marqué serait borné.

Propriété Si un RdP marqué n'est pas borné pour le marquage initial M_0 alors il n'est pas borné pour le marquage initial $M'_0 >= M_0$.

5.2.2 Vivacité et blocage

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

Une transition T_j est vivante pour une marque initiale M_0 si pour tout marquage accessible M_k, il existe alors une séquence de franchissement à partir de M_k contenant T_j:

$$\forall M_k \in {}^*M_0, \exists S, M_k / S > et S = ...T_j....$$

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

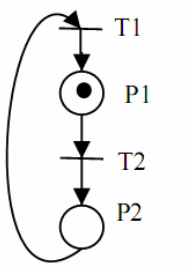

Figure 13 : RdP vivant

Les transitions T_2 et T_3 du RdP marqué Figure 10 ne sont pas vivantes. Les transitions T_1 et T_2 du RdP marqué Figure 13 sont vivantes : pour ce RdP.

*Un **RdP** marqué est **vivant pour un marquage initial** M_0 si toutes ces transitions sont vivantes pour ce marquage initial. Un RdP est dit **conforme** s'il est sauf, vivant.*

Les RdP marqué Figure 13 est vivant et conforme.

*Une **transition T_j est quasi vivante pour un marquage initial** M_0 s'il existe une séquence de franchissement à partir de M_0 contenant T_j:*

$$\exists S,\ M_0 / S > \ et\ S = \ldots T_j \ldots$$

*Un RdP marqué est **quasi vivant pour un marquage initial** M_0 si toutes ces transitions sont quasi vivantes pour ce marquage initial.*

Le RdP marqué figure 10 est quasi vivant.

6- Les réseaux Workflow (WF net)

Avant d'entrer dans les détails des applications des réseaux de Petri au workflow interorganisationnel, nous considérons que la modélisation et l'analyse des workflows a un seul organisme.

Les workflows sont considérés comme une base d'état de chaque partie du travail qui est exécutée pour un état bien déterminé. On peut citer quelques exemples d'états : une hypothèse, une demande d'assurance, déclaration d'une taxe, une commande ou une demande de renseignements etc. Les états sont souvent engendrés par un client externe. Cependant, il est aussi possible qu'un état puisse être généré par un autre département au sein d'un même organisme (client interne).

Le but de la gestion du workflow est de maintenir les états aussi efficaces et effectives que possible. Le processus du workflow est conçu pour tenir d'autres états partiels.

Les états sont tenus en exécutant les tâches dans un ordre bien déterminé. La définition du processus du workflow spécifie l'ordre d'exécution des tâches.

Dans le processus workflow, les blocs de construction comme ET-partager, ET-joindre; OU-partager, OU-joindre (AND-split, AND-join, OR-split and OR-join) ; sont utilisés pour modéliser un modèle séquentiel, conditionnel, parallèle et itératif respectivement.

Il est clair que le réseau de Petri peut être utilisé pour déterminer le chemin des états. Les tâches sont modélisées par les transitions alors que les dépendances causales sont modélisées par des places. En fait, une place correspond à une condition qui peut être utilisée comme pré et/ou post-condition pour des tâches. Un ET-partager ''AND-split '' correspond à une transition avec deux ou plusieurs places de sortie alors que un ET-joindre '' AND-join '' correspond à une transition avec deux sortants/entrants.

6.1 Réseau WF

Un réseau de Petri *PN= (P, T, F)* est considéré comme un réseau WF si et seulement si [6]:

1- *PN* possède deux places spéciales : i et o.
   a. La place i est une place d'entrée; $\bullet i = \Phi$.
   b. La place o est une place de sortie.
2- Si on ajoute une transition t* au PN qui connecte la place o avec i, (c à d : $\bullet t^* = \{o\}$ et $t^* \bullet = \{i\}$), alors le réseau de Petri obtenu est fortement connecté.

Un réseau WF possède une place d'entrée (i) et une place de sortie (o) parce que n'importe quel état manipulé par la procédure représentée de la part du réseau WF est créé que si elle accède au système de gestion du Workflow. Cet état est supprimé une fois qu'il est complètement manipulé par le système de gestion du Workflow c'est à dire : le réseau WF détermine le cycle de vie d'un état.

La deuxième contrainte dans cette définition déclare l'exigence d'avoir un chemin entre la place ''i'' vers ''o'' à travers t pour chaque transition.

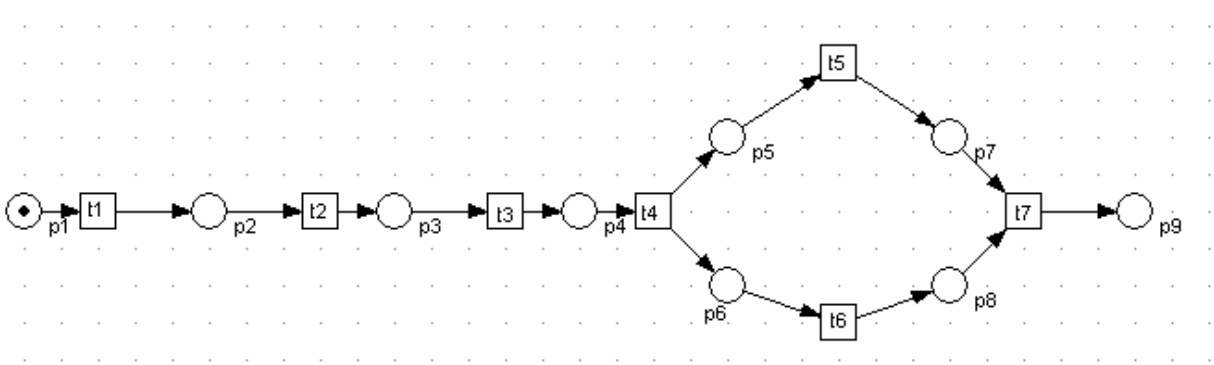

Figure 14 : Un exemple de WF-net

La figure 14 illustre un réseau WF. Ce réseau WF modélise un processus Workflow composé de sept tâches : t_1,.., t_7. L'exécution des tâches t_1, t_2, t_3, t_4, séquentielle et l'exécution de la tâche t_4 active deux flux parallèles t_5 et t_6. Les deux flux parallèles sont synchronisés par t_7. On peut remarquer que l'ensemble d'aiguillage séquentiel, conditionnel et parallèle est présent dans cet exemple.

6.2 Vérification des réseaux WF

Les deux exigences présentées dans la définition précédente peuvent être vérifiées de façon statique, c'est-à-dire qu'ils ont uniquement un rapport avec la structure du réseau de Petri. Cependant, une autre exigence doit être satisfaite.

En aucun cas la procédure ne sera éventuellement terminée qu'au moment où la procédure est finie : il y a un signe dans la place ''o'' et que toutes les autres places sont vides. De plus, il ne faut pas avoir de tâches stagnantes, c'est-à-dire qu'il sera possible d'exécuter une tâche arbitraire en suivant le chemin approprié à travers le réseau WF.

Ces deux contraintes supplémentaires correspondent à la soi-disant propriété « Bon état ».

Une procédure modélisée par un réseau WF PN= (P, T, F) est résonnée (sound) si et seulement si :

1- *pour tout état « M » accessible de l'état « i » il existe alors une séquence franchie conduisant de l'état « M » à l'état « o ».*

$\forall M\,(i[*)M \Rightarrow (M[*)o)$

2- *L'état « o » est le seul état accessible de l'état « i » avec au moins un signe dans la place « o ».*

$\forall M\,(i[*)M \ \wedge M \geq o \Rightarrow (M = o)$

3- *Ils n'ont pas de transitions stagnantes dans (PN, I).*

$\forall t \in T\ \exists M, M'i[*)M[t)M'$

Il faut noter que la propriété Bon état '' optimal'' «soundness » est relative aux dynamiques du réseau WF. La première exigence dans la définition précédente déclare que lorsqu'on commence de l'état initial (état i), il est toujours possible d'atteindre l'état avec un seul signe en place 'o' (état o). Si nous assumons de l'idée de franchise, dans ce cas la première exigence implique que le 'o' sera éventuellement achevé.

Le fait d'assumer qu'une franchise est raisonnable dans le contexte de gestion du workflow, tous les choix sont alors faits implicitement et explicitement par des applications (êtres humains ou acteurs externes). Il est clair qu'ils ne doivent pas introduire une boucle infinie. La deuxième exigence déclare qu'au moment où le signe est placé à l'endroit 'o', toutes les autres places doivent être vides. La dernière exigence déclare qu'il n'existe plus de transitions stagnantes dans l'état initial 'i'.

Théorème 1 [12] *Workflow net PN =(P,T,F) est résonné (sound) si et seulement si (PN',i), tel que PN'=(P',T',F'), P'=P,* $T' = T \cup \{t^*\}$, *et* $F' = F \cup \{(o, t^*), (t^*, i)\}$, *est vivante et bornée.*

7 Workflow interorganisationnel (WFIO)

Dans les deux sections précédentes, nous avons appliqué les réseaux de Petri à la modélisation et l'analyse des workflows au sein d'un seul organisme (organisation). C'est le temps maintenant de considérer les workflows interorganisationnel.

Le WIO est considéré comme une extension du workflow traditionnel puisqu'il envisage la coopération de plusieurs processus issus de différentes organisations réparties**,** autonomes et hétérogènes. Il se distingue du workflow traditionnel par trois aspects essentiels [7]:

- La répartition des processus d'organisations.
- L'autonomie des organisations : chaque organisation prise individuellement décide par elle-même les conditions de coopération : c'est à dire quand, comment et avec qui elle doit coopérer.
- L'hétérogénéité des organisations à faire coopérer : cela concerne les différences en termes de modèles et de systèmes.

Le WIO prend aujourd'hui une importance considérable car il a des applications immédiates et utiles telles que les procédures administratives, interministérielles et les services fédérés proposés par des entreprises virtuelles etc.

Un workflow intergénérationnel est essentiellement un ensemble de processus couplés sans précision, d'un workflow typique ; il n'y a pas de partenaires d'affaires qui sont impliqués dans un processus workflow « Global ». Chacun de ces partenaires possède son propre processus workflow « local ».

Chaque processus workflow local est considéré comme particulier, c'est-à-dire que le partenaire correspondant peut être complet. Ces processus workflow locaux ont besoin de se communiquer, parce que chacun d'eux dépend des autres pour la bonne exécution des états.

Le processus workflow global se compose de processus workflow locaux et une structure d'action réciproque. Il existe deux façons d'avoir une action réciproque : la communication asynchrone et la communication synchrone. La communication asynchrone correspond à l'échange de messages entre les processus workflow locaux.

La communication synchrone oblige les processus workflow locaux d'exécuter en même temps des tâches bien déterminées.

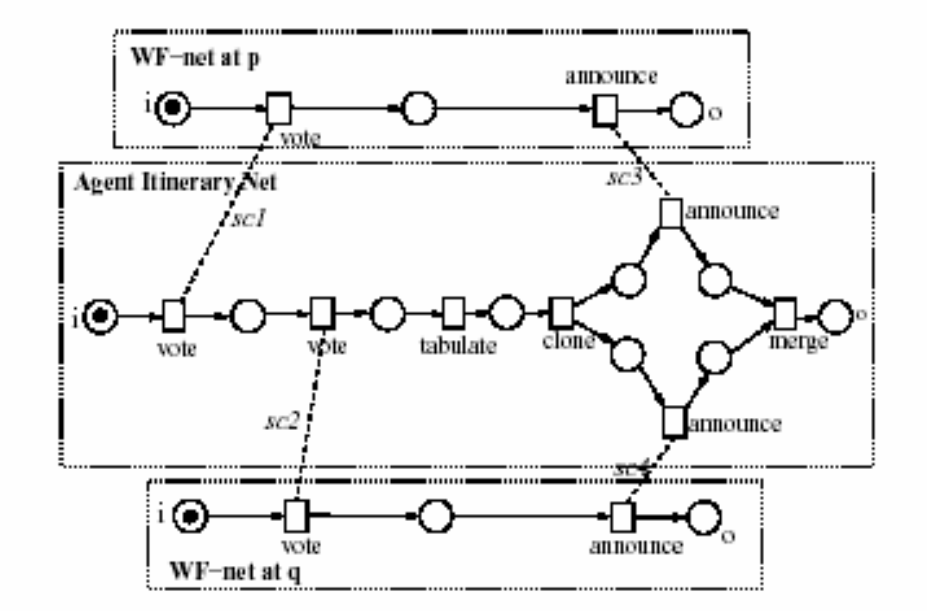

Figure 15 : IOWF-net pour le vote [1]

Un agent V, à partir de sa place d'origine porte une liste de candidats de l'hôte à l'hôte visitant chaque partie de vote. Une fois que chaque partie a voté, l'agent va à la place original tabuler des résultats (supposant que la place d'origine fournit les ressources et les détails au sujet de la façon tabulée). Il annonce alors les résultats à tous les votes en parallèle (et le clonage pour exécution parallèle). Assumant deux votes (à place *p* et place *q*), *vote* est une action acceptant une vote. La *tabulation* est une action de tabuler des résultats. *Annonce est* une action d'affichage des résultats.

Le comportement de mobilité est : $V_p^{vote}.V_q^{vote}.V^{tabutation}.(V_p^{annonce} \| V_q^{annonce})$.

La figure 15 montre au l'IOWF-net correspondant, ce qui se compose de deux workflow locaux aux places *p* et *q* respectivement et un AI-net pour l'agent V. Il y a quatre points synchrones de communication : sc1, sc2, sc3 et sc4.

Les points de synchronisation offrent en même temps à workflow local une exécution des tâches. En d'autres termes, l'exécution des tâches dépend non seulement des conditions locales mais également de l'environnement externe.

7.1 Workflow interorganisationnel

Soit workflow interorganisationnel (WFIO) c'est le triple suivant :

WFIO = *(PN1, ..., PNn , P_{AC} AC,T_{SC} , SC*), où :

I. *n∈ N c'est le nombre du workflow nets local.*

II. *Pour chaque k∈ [1..n] : PN_k est le WF-net avec place départ i_k et place finale o_k.*

III. *Pour tout k, I ∈ [1..n] : si k≠ I, alors* $((P_k \cup T_k) \cap (P_I \cup T_I) = \phi$.

IV. $T^{\square} = \bigcup_{k\in[1..n]} T_k, P^{\square} = \bigcup_{k\in[1..n]} P_k, F^{\square} = \bigcup_{k\in[1..n]} F_k,$

V. P_{AC} *c'est l'ensemble des éléments de communication synchrone (les places de communication).*

VI. T_{SC} *c'est l'ensemble des éléments de communication synchrone (les ensembles fusion).*

VII. $P_{AC} \cap T_{SC} = \Phi, (P_{AC} \cup T_{SC}) \cap (P^{\square} \cup T^{\square}) = \Phi$

VIII. $AC \subseteq P_{AC} \times P(T^{\square}) \times P(T^{\square})$ *sont les relations de communication asynchrone.*

IX. $SC \subseteq T_{SC} \times P(T^{\square})$ *c'est la relation de communication synchrone.*

X. Pour tout $p \in P_{AC}\{(p', x, y) \in AC \setminus p' = p\}$

XI. Pour tout $t \in T_{SC}\{(t', x, y) \in SC \setminus t' = t\}$

XII. *Pour tout* $(t_1, x_1), (t_2, x_2) \in SC$: *si* $t_1 \neq t_2$ *Alors* $x_1 \cap x_2 = \phi$

Chaque élément de communication asynchrone correspond à un nom de place dans P_{AC}. La relation *AC* spécifie un ensemble de transition d'entrée et de sortie pour chaque élément de communication asynchrone. La condition 'exigence' *(x)* indique que pour chaque place de communication il y a un seul élément dans *AC.*

Chaque élément de communication synchrone est représenté par un nom de transition dans T_{SC} qui correspond à un ensemble de transitions fusionnées. La relation *SC* spécifie pour quel élément dans T_{SC}, l'ensemble correspondant des transitions fusionnées. La condition (XI) indique qu'il existe un seul élément dans *SC* pour chaque ensemble de fusions. La condition *(XII)* déclare que ces ensembles de fusions doivent être séparés. Il est à noter que la définition précédente, autorise la communication des éléments qui sont connectés au même Workflow.

Il est aussi à noter que chaque réseau Petri local a une place d'entrée I_k et une place de sortie O_k, Parfois on n'aura pas besoin de ces places, par exemple, si une organisation est un sous-traitant d'une autre organisation : alors le workflow du sous-traitant peut être initié par un message (c'est-à-dire un élément de communication asynchrone). Tandis que pour des

raisons sémantiques on ajoute une place d'entrée I_k et une place de sortie O_k. Si l'on prend l'exemple de la figure 16, on peut déplacer i_1 et i_2 sans changer le comportement actuel.

7.2 Vérification des Workflows Interorganisationnels

Dans la section 5, on a présenté une technique pour vérifier l'exactitude d'une définition du processus workflow dans l'isolation, sachant qu'on peut utiliser cette technique pour montrer que les deux workflows locaux (figure16) sont corrects c'est-à-dire, LWF1 et LWF2 sont résonnants. Cependant, un workflow interorganisationnel qui est composé d'un nombre de workflows locaux résonnants peut être sujet d'erreurs de synchronisation. Considérons par exemple le workflow interorganisationnel illustré dans figure16. Il est possible que LWF1 exécute t4 et LWF2 exécute t12. Dans ce cas, le message dans ac2 est mal tenu. Il est aussi possible que les impasses soient introduites par les éléments de communication.

On définit le déploiement d'un workflow interorganisationnel dans un réseau WF.

Soit *IOWF= (PN1, ..., PNn, P_{AC} , AC,T_{SC} , SC*), est interorganisationnel Workflow. $U(IOWF) = (P^U, T^U, F^U)$ est unfoloding de IOWF est défini comme suivant:

I. $P^U = P^{\square} \cup P_{AC} \cup \{i, o\}$,

II. $T^U = \mathrm{r}(\mathrm{T}^{\square}) \cup T_{AC} \cup \{t_i, t_o\}$,

III. $\{i, o, t_i, t_o\} \cap (P^{\square} \cup \mathrm{T}^{\square} \cup P_{AC} \cup T_{SC}) = \phi$.

IV. *R c'est une fonction de nomination : r(x)=t_{SC} si il y a $t_{SC} \in T_{SC}$ et a y $\subseteq$ $\mathrm{T}^{\square}$ tel que* $(t_{SC}, y) \in SC$ *et* $x \in y$ *, sinon r(x)=x.*

V. $F' = F^{\square} \cup \{(t,p) \in T^{\square} \times P_{AC}\ \{(p,x,y) \in AC \wedge t \in x\} \cup$

$\{(p,t) \in P_{AC} \times T^{\square}\}\{(p, x, y) \in AC \wedge t \in y\} \cup$

$\{(i, t_i), (t_o, o)\} \cup \{(t_i, i_k), k \in \{1,...,n\}\} \cup \{(o_k, t_o), k \in \{1,...,n\}\}$.

VI. $F^U = \{(r(x), r(y)), (x, y) \in F\}$.

Dans un réseau déployé, tous les réseaux locaux WF sont connectés entre eux avec une transition de départ ''T_i'' et une transition finale t*. En plus, une place source globale ''i'' et une place but globale ''O'' ont été ajoutées. Les éléments de communication asynchrone sont arrangés dans des places ordinaires (P_{ac}). Les transitions fusionnées ensemble par les éléments

de communication asynchrone sont remplacées par de nouvelles transitions (T_{sc}). Le résultat du déploiement est un nouveau réseau WF.

7.3 (IO-Soundness)

Du IOWF est IO-sound (IO-optimal) si et seulement si localement et globalement optimal. Un IOWF est localement optimal si et seulement si tous les Workflow net locaux sont optimaux (sound). Un IOWF est globalement optimal si et seulement si U (IOWF) est optimal.

Le workflow interorganisationnel dans la figure 16 est un exemple d'un workflow qui est résonnant localement mais pas globalement. Le réseau déployé n'est pas résonnant. Dans le cas si t_4 et t_{12} fonctionnent, un signe sera fixé dans la place ac_2. L'erreur peut être corrigée en remplaçant l'élément de communication asynchrone ac_2 par un élément de communication synchrone.

La figure 16 montre un exemple d'un IOWF globalement optimal mais non localement; le WF-net LWF2 n'est pas optimal puisqu'un nombre arbitraire de jetons peuvent avoir piégé en place p14. Tandis que l'élément de communication asynchrone ac_2 l'empêche.

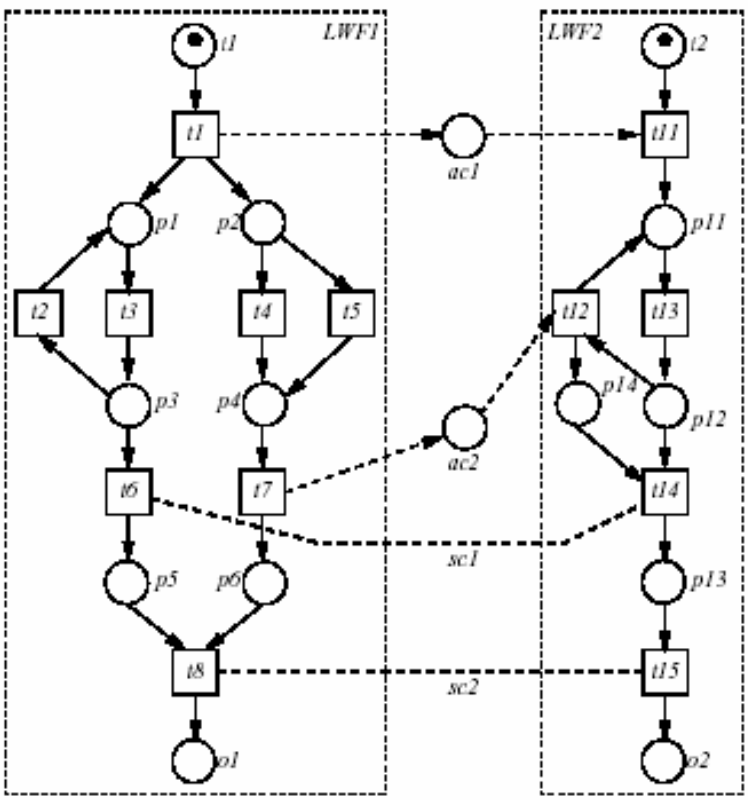

Figure 16 : Le IOWF qui est globalement optimal mais n'est pas localement optimal [6]

Ces exemples montrent qu'il est possible d'avoir un workflow interorganisationnel localement optimal mais non globalement et vice-versa pour le rendre optimal (IO-Optimal). Le workflow interorganisationnel doit être optimal dans les deux côtés local/global. Dans la figure 17 l'exemple de workflow interorganisationnel optimal est représenté.

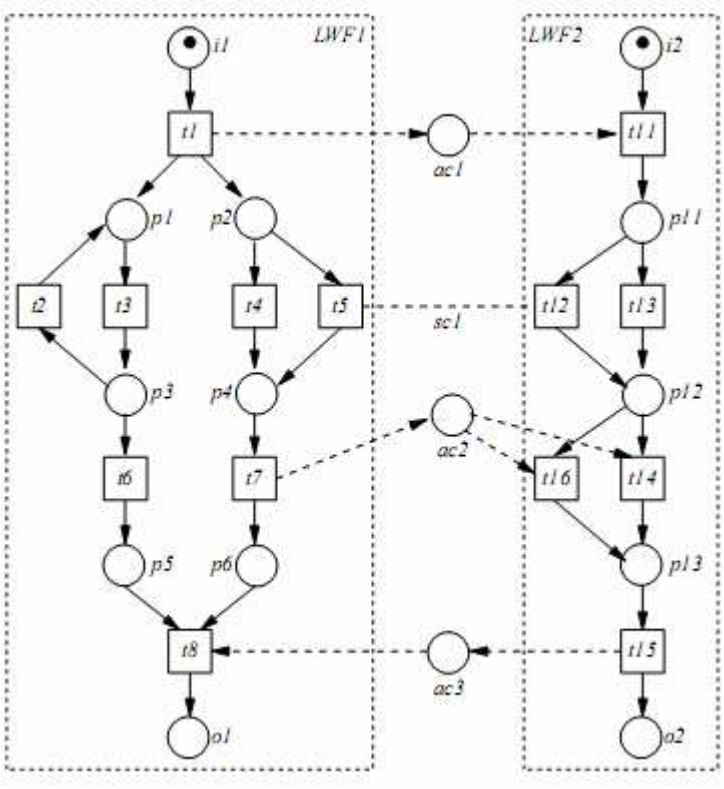

Figure 17 : workflow interorganisationnel est optimal [6]

L'optimisation d'un workflow interorganisationnel *IOWF= (PN1, ..., PNn, PAC AC, TSC, SC correspond à l'optimisation de* $n+1$ WF-nets: PN_1, ..., PN_n *et* U *(IOWF).* Par conséquent on peut utiliser le théorème (1) pour vérifier l'exactitude du workflow interorganisationnel. Intuitivement, la sémantique des réseaux IOWF rassemble le comportement des agents mobiles (s) de l'interaction avec plusieurs organisations à différents endroits. Un agent se déplace avec un itinéraire bien précis entre des processus de déroulement pour activer et synchroniser les tâches locales des organisations. Effectivement, l'agent itinéraire peut être représenté par un seul réseau WF, dont les transitions sont les tâches à différents endroits spécifiés dans l'itinéraire. Nous appelons un tel réseau WF un agent itinéraire (AI) de réseau.

Nous définissons une notion d'agent vivant [1]:

Soit A est AI-net qui a une interaction avec n nombre de WF-net (N_1 ... N_n), dans un

IOWF-net, IOWF = (N_1; ... N_n; A; $S_{A\ C}$; A C; T_{SC}; SC), avec un état initial M0 = i_{N1} +… +i_{Nn}+ i_A.

Un agent α, dont le comportement est spécifié par A, est un vivant si et seulement il répond aux conditions suivantes:

1. A est optimal (sound) ;

2. Pour chaque séquence de la transition σ généré en A, de telle sorte que $i_A \xrightarrow{\sigma} o_A$, il y a une séquence correspondante générée en IOWF tels que:

(A) $M_0 \xrightarrow{\sigma'} M_f$, où M_f est l'état final et $o_A \in M_f$.

(B) σ est une subséquence de σ' .

La condition 1 est une condition locale. Elle précise que le AI-net doit être localement optimal (sound) et doit accomplir les tâches assignées à l'itinéraire pour atteindre son état final o_A.

La condition 2 insiste sur le fait que dans le cadre interorganisations, de coopération et de l'environnement mondial, l'agent doit aussi être en mesure d'atteindre son état final o_A.

Soundness implique Agent Liveness:

Compte tenu d'une IOWF-net contenant une AI-net d'un agent α, si le IOWF net est optimal, alors α est a-live (a est vivant).

8 Conclusion

Le problème difficile de la gestion des workflows interorganisationnels qui se pose est le suivant : comment définir et gérer les processus participants par l'organisation autonome, le contrôle centralisé et usuellement conduisant à un coût très élevé de communication ainsi que la réduction l'autonomie des participants.

Dans le chapitre suivant, on présentera l'intégration de la notion orienté objet dans le réseau de Petri. Cette intégration a donné un nouveau modèle appelé G-net.

1 Introduction

On présente un environnement de réseaux de Petri appelé G-net pour la conception modulaire et la spécification de système d'information distribuée. L'environnement est une intégration de la théorie de réseaux de Petri avec l'approche de génie logiciel pour la conception du système.

La motivation de cette intégration est de faire un pont entre le traitement formel de réseaux de Petri et l'approche orienté objet. Celle-ci offre la spécification et le prototypage d'un système complexe. Dans ce chapitre, on a utilisé un modèle basé sur le réseau de Petri appelé le G-net où nous allons présenter les concepts de base du G-net et le système G-net.

2 G-net et système G-net

La conception des logiciels est basée sur un principe " largement acceptable" dont le système doit être composé d'un ensemble de modules indépendants où chaque module cache les détails internes des activités de traitement. Les modules se communiquent entre eux à travers des interfaces bien définies [2].

Le modèle G-net fournit un support très puissant pour ce principe les G-nets présentent une extension d'un objet de base des réseaux de Petri : nous notons que les réseaux de Petri incluent trois entités de base : les places (représentées graphiquement par des cercles), les transitions (représentées graphiquement par les barres pleines), ainsi, les arcs dirigés peuvent relier des endroits aux transitions ou aux places. En outre, les places peuvent contenir les jetons, appelés marque. Un jeton peut se déplacer entre les places par le « franchissement » des transitions associées. L'état d'un réseau de Petri se rapporte à la distribution des jetons dans les places à n'importe quel point particulier de temps (un tel arc n'est pas réellement une partie de la structure statique des modèles de G-net).

Un système G-net est composé d'un ensemble des G-net, chacun d'eux représente un module ou un objet autonome. Un G-net est composé de deux parties [3] :

Une place spéciale appelée « Generic Switch Place » (GSP) et une structure interne « Internal Structure» (IS). Le GSP fournit l'abstraction du module et peut servir comme la seule interface entre le G-net et les autres modules.

Le IS qui est un réseau de Petri modifié, représente une conception détaillée du module.

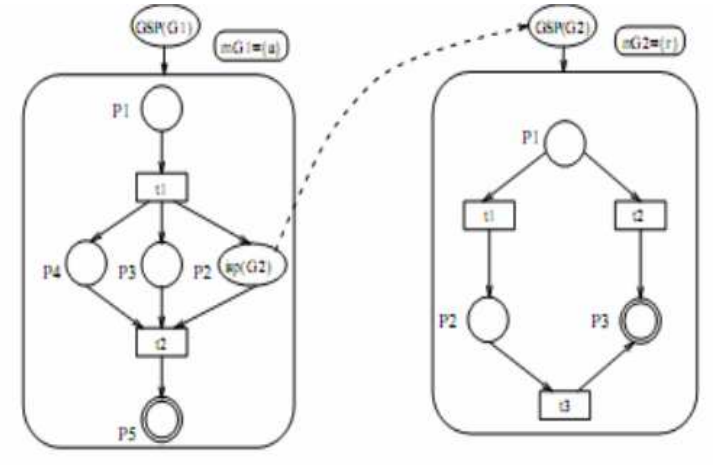

Figure 18 : Deux G-nets connectés par isp [3]

La figure 18 montre un exemple du G-net. Ici les modules G-net représentent deux objets : l'acheteur et le vendeur. Les GSP sont indiqués par GSP (acheteur) et GSP (Vendeur) entourés par des ellipses.

3 Les notions de base

Nous donnons maintenant quelques définitions clé qui nous montrent la structure formelle de notre modèle G-net.

3.1 Le système G-net

Un système G-net est un Triple GNS= (TS, GS, AS) où :

- TS est une collection de jetons qui sont générés automatiquement durant l'exécution du système.
- GS est un ensemble de G-net
- AS est un ensemble d'agents qui sont décentralisés et qui exécutent le système G-nets.

3.2 G-net

Un G-net est un triple G (GSP, IS) où :

a. GSP est une place générique (GSP) qui fournit une abstraction pour le G-net.
b. IS est la structure interne tant qu'elle est une transition/prédicat modifiée du réseau.

3.3 Place générique (Generic Switching Place)

GSP est défini par (NID, MS, AS) où :

a. NID est un identificateur unique du G-net G.

b. MS est un ensemble de méthodes ; et

$\forall\ mtd_i \in MS,\ mtd_i = (m\_name_i, m\_\arg uments_i, m\_initiator_i)$, où

m-name : est un nom pour le MTDi.

m_argument :est un triple de variables spécifiant le jeton d'entrée pour le MTDi.

m-initiator : est une forme similaire m-argument, le marquage initial du réseau à la structure interne associée à la méthode MTDi.

c. AS est un ensemble d'attributs et $\forall as_i \in AS, as_i \in N$. de la définition précédente où qu'un GSP est identifié uniquement par un nom dénoté par NID qui est l'abstraction d'un ensemble de méthodes définissant tous les ensembles possibles du marquage initial, qui résultent dans différents comportements de la structure interne. Dans la définition 2.3.3 m-initiator est défini comme des informations sur m-arguments. Celles-ci sont transformées à des jetons du type correspondant. Ces jetons sont déposés dans le cas où plus d'un argument est défini dans la place initiale pour la méthode.

3.4 La structure interne

La structure interne d'un G-net est une structure de réseau qui est un graphe biparti orienté et défini par :

G.IS= (Σ , P, T, I, O), où

1- Σ est une structure constituée d'un certains types de prédicats : ensemble avec un ensemble de relations et d'opérations définies sur ces prédicats.

2- P est un ensemble fini est non vide de places notées par des cercles.

3- T est un ensemble fini de transitions dénotées par des rectangles.

4- $I : T \longrightarrow P^{\infty}$ est appelée la fonction d'entrée, qui assigne des inscriptions sur les transitions d'entrée des arcs.

5-O : $T \longrightarrow P^{\infty}$ est appelée la fonction de sortie, qui assigne des inscriptions sur les transitions de sortie des arcs.

3.5 Ensemble de places

On a trois différents types de places dans la structure interne du G-net définis comme suit [5] :

G.IS est la structure interne du G-net. L'ensemble de places $P \in G.IS$ est défini par : P= (*ISP, NP, GP*) où :

1- *ISP* est un sous-ensemble de places instables.

2- *NP* est un sous-ensemble de places normales.

3- *GP* est un sous-ensemble de places cibles.

La définition ci-dessus pour l'ensemble de places p, G est nécessaire puisque chaque sous-ensemble de places a différents sens. Maintenant, on va détailler le sens de chacun de ces sous ensembles de places.

Le isp$\in$ *ISP* fournit le mécanisme utilisé dans les systèmes G-nets pour implémenter l'interconnexion G-net. Un isp définit dans un réseau G et fait appel au réseau G', dénoté par isp(*G'm*) et est défini comme un quadruple :

Isp(G'm)=(NID, mtd, action-Avant, action-Après)

NID est l'unique identification de *G'.mtd* $\in$ G'.GSP.MS, *action-Après* et *action-Avant*, qui sont des actions primitives. Ce qui est appelé la primitive propagation des jetons de l'action.

Plus spécifiquement, une méthode de *mtd* $\in G.MS$ définit les paramètres d'entrées, le marquage initial du RDP interne correspondant (l'état initial de l'exécution).

L'ensemble des places cibles représentent les états finaux de l'exécution de chaque méthode. Ainsi les résultats seront obtenus. La collection des méthodes et les attributs fournissent l'abstraction ou la vue externe du module. On impose la restriction. C'est que l'ensemble des places cibles ne doit pas être vide.

3.6 Les jetons

Soit G un triple et tkn est un jeton, alors

1. tkn est un triple d'items de la forme : *tkn= (seq, scolor, d1,…, d2),* où :

tkn.seq est la séquence de propagation du jeton tkn.scolor $\in$ (Avant, Après) est l'état de couleur du jeton et tkn.d_i , i $\in$ א est la partie du message du jeton.

L'item *tkn.seq* est une séquence de (NID, isp, PID) où NID est l'identification du G-net, isp est le nom d'un ISP et PID est une identification unique d'un processus.

Le jeton est offert uniquement pour autoriser le franchissement des transitions scolor-Après.

2. Si tkn=w, il peut se comparer avec n'importe quelle séquence.

La séquence de propagation d'un jeton est changée seulement lorsqu'on accède un ISP en un GSP. Lorsqu'un G-net G est invoqué d'un ISP isp dans un autre G-net G' (lorsqu'un isp reçoit un jeton), un triple (G', isp, $Pid_{G'}$), où $Pid_{G'}$ est l'identificateur d'un processus qui

exécute G' inséré dans la séquence de propagation du jeton avant qu'il ne soit envoyé vers G-net G'. Cette triple indique que lorsque l'exécution de G dépasse le jeton, le résultat doit retourner vers l'identification de la place par isp dans le G-net G'. L'identification du processus est nécessaire pour distinguer à quelle instance d'exécution du G', le jeton retourné appartient.

Lorsque le jeton d'entrée est reçu dans le GSP G, un triple (0, 0, Pid_G) est inséré à la fin de sa séquence de propagation. Ceci indique que l'agent responsable à exécuter l'invocation est identifié par Pid_G. Parce que Pid_G est unique, la séquence de propagation est aussi unique. La structure du jeton dans la plate-forme du G-net ne garantit pas seulement que tous les jetons qui appartiennent à une instance d'une exécution d'un G-net ont la même et l'unique séquence de propagation mais contiennent aussi l'historique complet de la propagation des jetons. Ceux-ci génèrent les interactions entre les processus qui exécutent la spécification du G-net.

Dû au champ de séquence associée à un jeton, plus d'un appel au G-net peut être exécuté simultanément. Puisque les différentes exécutions (les appels) peuvent être uniquement identifiés par la séquence de propagation.

Comme il a été indiqué précédemment, l'état couleur d'un jeton à 2 valeurs possibles soit: Avant ou Après. Un jeton est invariable si scolor=Après. Sinon il est variable.

Lorsqu'un nouveau jeton est reçu dans une place, son état est affecté par Avant. Après la primitive action dans la place est prise, l'état couleur du jeton est affecté alors par Après indiquant que le jeton est prêt à être utilisé dans un franchissement de transition séquentielle. Le champ du message d'un jeton est une liste de valeurs d'une application spécifique.

Une place normale $\boldsymbol{np} \in \boldsymbol{NP}$ aux règles pour manipuler le jeton lorsqu'un jeton a les paramètres d'entrées spécifiés par le champ du message.

Le résultat de l'action est associé au champ du message du jeton et un champ appelé branche couleur du jeton est défini. La branche couleur d'un jeton sert à définir à quelle transition de sortie le jeton est offert. L'action associée à une place normale manipule la même action comme des expressions et les inscriptions de transitions d'un réseau PrT. Finalement le champ scolor du jeton est mis à jour pour scolor $\longleftarrow$ Après, et le jeton est Après pour autoriser le franchissement des transitions.

Une place cible GPi GP doit appartenir à un marquage final de G.IS. Pour chaque méthode mtd, les places cibles doivent être accessibles et uniques pour chaque méthode. L'information résultat de l'exécution peut être retournée au G-net appelée :

Le mécanisme de franchissement des transitions du G-net est défini par les règles suivantes de franchissement des transitions.

3.7 Les règles de franchissement transition

Dans le G-net, la transition franchie par les règles suivantes :

1. Une transition est dite franchissable si chaque place $P \in I(t)$ contient au minimum un jeton et son contenu satisfait la condition définie par I (p, t),
    a- Tous les jetons restants ont la même séquence de propagation.
    b- Tous les jetons restants ont leur *scolor=Après*.
    c- Le nombre de jetons dans $O(t)$ est inférieur de la capacité des places.
2. Une transition est franchissable dans les cas suivants :
    a. un jeton qui satisfait $I(p, t)$ est supprimé de chaque $p \cup I(t)$.
    b. un jeton qui a sa séquence de propagation est de la même somme que celle où les jetons supprimés de $I(t)$, qui ont scolor=Avant, et qui a la partie du message est déterminée par $O(t, p)$ est déposée à chaque place $p \in O(t)$.

L'appel d'un G-net définit le mécanisme pour commencer l'exécution des structures internes basées sur le message associé au jeton.

3.8 L'appel d'un G-net

Soit un G-net G, l'appel d'un réseau G qui se base sur une méthode $mtd \in$ G. MS est rejeté de la manière suivante :

1. Détermine le marquage initial basé sur la définition d'un *mtd* (le contenu de jetons dépend du contenu du jeton d'entrée).
2. Une transition autorisée et franchissable.
3. Appel des primitives autorisées.
4. Répéter (2)-(3) jusqu'à ce qu'une place cible est accessible.
5. Envoyer le résultat d'exécution (si défini par *mtd*) à l'appelant.

4 Exemple de modèle G-net pour les objets de acheteur /vendeur [11]

Un exemple de G-nets est illustré dans la figure 19. Ici le modèle G-net représente deux objets : un commerçant et un acheteur.

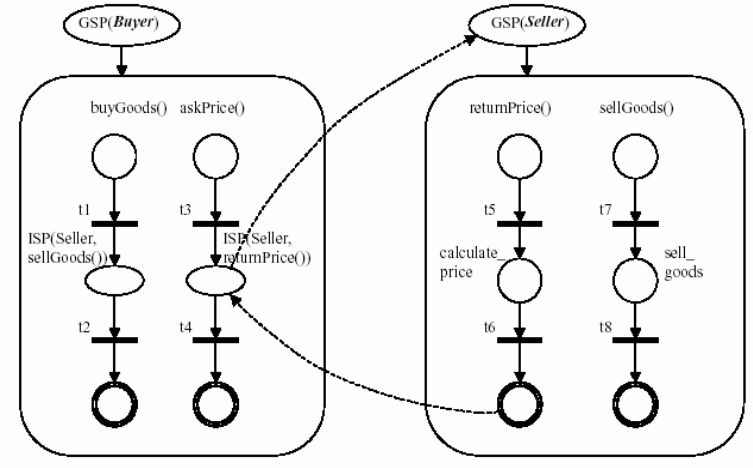

Figure 19 : Modèle G-net pour les objets de : acheteur /vendeur [11]

Les places génériques sont représentées par *GSP* (*acheteur*) entouré par une ellipse et les structures internes de ces modèles sont représentées par des rectangles qui contiennent quatre méthodes : *buygoods*(), *askprice*(), *returprice*() définies comme suit :

Baygoog appelle la méthode *sellgoog*() définie dans G-net seller pour acheter certaines marchandises, *askprice*() appelle la méthode *returnprice*() définie dans G-net seller pour obtenir le prix de certaines marchandises, *returnprice* () est définie dans G-net seller pour calculer le prix minimum pour certaines marchandises et *sellgoog*() est défini dans G-net seller pour attendre le payement, acheter les marchandises et générer.

Un GSP d'un G-net G contient un ensemble de méthodes *G.Ms* spécifiant les services ou les interfaces fournies par les modules et un ensemble d'attributs G-AS qui définissent les variables d'états. Dans G.IS la structure interne d'un G-net, les places du réseau de Petri représentent des primitives. Les transitions et les arcs ensembles représentent des connexions ou des relations entre ces primitives. Les primitives peuvent définir des actions locales ou des appels des méthodes. Les appels de méthodes sont représentés par les places spéciales appeléés *ISP* (*Instantiated switsh place*). Une primitive devient valide si elle reçoit un jeton alors une primitive valide peut être exécutée.

Un G-net donné, un ISP d'un G est un 2 triples (*G'.Nid.mtd*). Où G' peut être le même G de *G.m*, où un autre G-net, Nid est un identificateur unique du G.net G' et *mtd G'.MS*.

Chaque *ISP (G'Nid, mtd)* dénote un appel de méthode *mtd*() à G-net G'. Un exemple ISP (dénoté comme une ellipse dans la figure 18) est illustré dans la méthode *askprice*() définie dans G-net *Buyer*, où la méthode *askprice*() fait un appel de méthode à *retunprice*() de G-net *selle* pour connaître le prix de certaines marchandises notées. On a représenté cet appel dans la figure 18 par une flèche discontinue mais ce type d'arc n'est pas actuellement une

partie de la structure statique d'un modèle G-net. De plus, on a enlevé tous les paramètres de fonctions et les déclarations de variables pour la simplicité.

On appelle un modèle G-net qui supporte les différents types de classes du modèle un modèle G-net standard. Il est à noter que l'exemple de la figure 3 soit le prototype client-serveur dans lequel l'objet acheteur a le rôle d'un client. Ce modèle est le résultat de l'utilisation de la notion d'héritage qui offre la spécification des classes et qui utilise dans la conception tous les avantages liés à l'héritage; (encapsulation, réutilisation, etc.). Par conséquent le modèle G-net standard est efficace dans la conception à base objet ce qui n'est pas suffisant dans le cas de la conception à base d'agent pour les raisons suivantes :

Premièrement. Les agents dans des systèmes multi agents sont toujours développés par différents vendeurs indépendamment, ces agents vont être distribués sur des réseaux à grande échelle comme l'Internet. Pour rendre possible le contact entre ces agents il faut avoir un langage commun entre eux et suivre des protocoles communs. Cependant, le modèle G-net standard ne supporte pas directement un langage basé sur des protocoles.

Deuxièmement. Le modèle de communication principale des agents est usuellement asynchrone et n'importe quel agent peut décider s'il veut exécuter des actions demandées par d'autres agents. Le modèle G-net standard ne supporte pas directement des applications de méthodes synchrones sous forme de places ISP.

Troisièmement. Les agents sont généralement conçus pour définir leurs comportements basés sur des objectifs individuels et leur propre connaissance. Ils peuvent initier d'une façon autonome et spontanée leur comportement interne ou externe à n'importe quel instant. Les modèles G-net standard peuvent supporter uniquement un flux de contrôle prédéfini.

5. Conclusion

On peut remarquer que le modèle G-net représente essentiellement un module ou un objet plus qu'une abstraction d'un ensemble d'objets similaires pour supporter une modélisation logicielle orienté objet. Dans le chapitre suivant, on induira un nouveau concept de système multi agent (*SMA*) et par une incorporation des modules additionnels dans le standard G-net. On dérive un modèle G-net à base d'agents qui supporte la modélisation d'agents.

1 Introduction

Notons que dans l'exemple précédent (Figure 11, chapitre2), celui-ci suit le paradigme client/serveur. Dans le cas où l'objet vendeur en tant que serveur et l'objet acheteur est un client quoique le modèle standard G-net fonctionne bien dans la conception basée sur un objet. Ce n'est pas suffisant comme un modèle basé sur un agent pour les raisons suivantes :

1- Agent dont la forme d'un système multi agent doit être développé par des vendeurs indépendants. Ces agents doivent être largement distribués sur des réseaux à grande échelle comme l'Internet. Pour rendre possible la communication entre ces agents, il est souhaitable pour eux d'avoir un langage de communication commune et suivre un protocole identique. Toutefois, le model standard G-net ne va pas supporter directement un langage basé sur le protocole de communication entre les agents.

2- Le modèle de communication agent introduit usuellement un asynchrone et l'agent doit décider de faire des requêtes par d'autres agents. Le modèle standard G-net ne va pas supporter directement un message asynchrone ainsi que la prise de décision mais seulement supporter un appel synchrone des méthodes sous la forme des places *ISP*.

3- Les agents sont conçus pour déterminer leur comportement, basé sur des buts individuels, leurs connaissances et l'environnement. Ils doivent d'une façon autonome et spontanée initier un comportement interne ou externe. Le modèle standard G-net peut seulement supporter un flux de contrôle prédéfini.

Dans ce chapitre, on va proposer un modèle G-net basé sur l'agent et étudier toutes les caractéristiques de ce modèle et enfin l'appliquer dans le cas de problème de vote.

2 Modèle G-net à base agent

Visant le problème de l'indépendance et l'héritage des modèles G-net pour les agents, on ajoute une place but, une place base de connaissance, une place environnement, le module planification, les fonctions détaillées et le processus de message pour le G-net traditionnel [4].

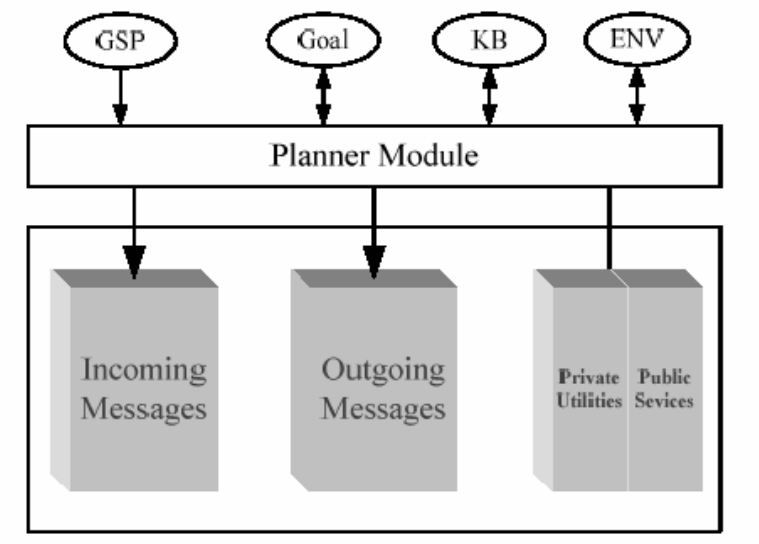

Figure 20 : Modèle de G-net pour l'agent [4]

Le G-net orientée agent est un 6 triple *AG= (GSP, IS, GP, KB, EP, Planner),* ou *GSP*, (*Generic Switch Place*) similaire de GSP dans G-net, est favorisé dans la conception d'agent seulement.

-*IS*, (*internal structure*) similaire de IS dans G-net, moins amélioré en ce qui concerne les mécanismes de synchronies et d'asynchrones de commande.

- *GP*, (*goal place*) place de but.
- *KB*, (*Knawledge base place*) place de base de connaissance.
- *EP*, (*Environment Place*) place d'environnement.
- *PLANNER*, module de planificateur

La place but représente les instructions ainsi que les arrangements de l'agent qui inclut la perception, jugement et perspective des agents pour une vision future. La Place de base de connaissance est l'abstraction sur les états internes. Les termes qu'un agent peut tenir : la connaissance inclue, connaissance de réalité, connaissance de règles, connaissance du contrôle et connaissance primitive définie. La place environnement définit l'abstraction de l'environnement. Les agents ont perçu l'environnement par des transducteurs qui ont un impact sur l'environnement à l'aide des appareils efficaces après un traitement de l'information d'environnement qui sont récupérés par le système.

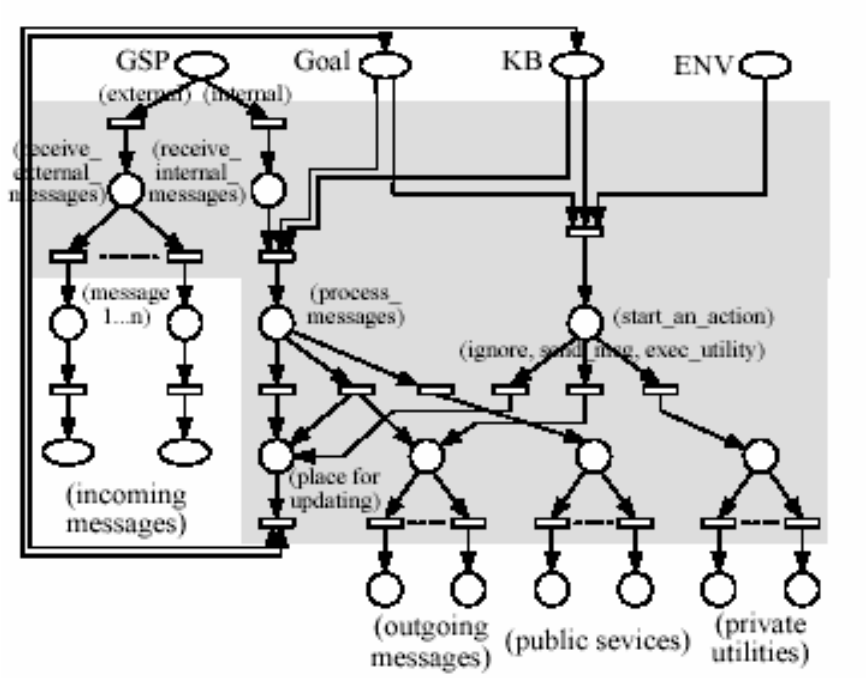

Figure 21 : Structure interne du module planificateur dans le modèle G-net Orienté Agent [4]

Le module planificateur représente les opérations des agents. Il relie l'état présent et l'environnement avec les buts, la connaissance pour accomplir la classification et la génération des informations et aussi le rafraîchissement des buts et connaissances etc. La structure interne du module planificateur est montrée sur la figure 21.

D'après le module G-Net orienté agent, la structure de l'IS est ajustée, ce qui est composé de trois parties : section du message entrant, section du message sortant et section de méthode. La section message entrant / sortant est utilisée pour traiter les messages entrants / sortants.

La section méthode contient des services publics et des utilités privés où les services publics définissant les méthodes dont les autres G-Nets auront besoin. Les utilités privées sont constituées de fonctions qui peuvent être introduites par le G-Net et peuvent aussi être rappelées par lui-même.

La plupart des communications entres agents sont asynchrones. Il est utile pour un système d'introduire un nouveau mécanisme appelé « place de commutation de messages passagers » "Message-passing Switch Palce" (MSP) afin de supporter les messages asynchrones passants directement. Quand un signe arrive à la zone MSP le signe est déplacé vers la zone GSP de l'agent appelant.

Autrement que le mécanisme ISP, l'agent appelant n'attend pas le signe de retourner avant qu'il ne puisse continuer à exécuter l'étape suivante.

Comme il est montré sur Figure 21. Le module but des messages entrants est un MSP, ça veut dire qu'il peut appeler lui-même avec une fonction vide. Dès qu'un MSP appelle lui-même, l'agent lui-même peut continuer son chemin en d'autres transactions.

Ils existent plusieurs aspects similaires entre les entités des agents conçus par le G-Net orienté agent et les objets traditionnels. Par exemple, ils ont également quelques entités évaluées qui résument les états et les fonctions et utilisent le mécanisme de message quand les entités ont besoin de se négocier entre elles.

Ils existent d'autres différences entres elles [4]:

1- Leurs mécanismes de prendre la décision sont différents. Les décisions sont exécutées par des objets appelant dans l'orientée objet.

2- Leurs comportements autonomes (autonomie, réactivité et socialité) sont différents. Les modèles d'objets standard n'ont pas ce qui est mentionné entre parenthèses par exemple, ils ne sont pas relatifs avec ces propriétés.

3 Les notions de base

Nous donnons maintenant quelque définitions-clé qui nous montrons la structure formelle de notre agent basé sur les modèles G-net.

3.1 Agent basé G-net

Un agent basé G-net est un 7 triple AG= (GSP, GL, PL, KB, EN, PN, PS), ou GSP est une place de choix générique qui produit une abstraction pour l'agent basé G-net, GL est un module but, PL un module Plan, KB un module de base de connaissance, EN un module d'environnement, PN un module planificateur et IS une structure Interne de AG.

3.2 Module Planificateur

Le module planificateur d'un agent basé G-net AG est un sous réseau colorié défini comme un 7 triple (*IGS, IGO, IPL, IKB, IEN, IIS, DMU*), où *IGS, IGO, IPL, IKB, IEN* et *IIS* sont respectivement en interface avec GSP, module but, module plan, module de base de connaissance, module d'environnement et une structure interne de AG.

DMU est un ensemble de l'unité de prise de décision, elle contient trois transitions abstraites; prise de décision et mise à jour.

3.3 Structure Interne

La structure interne d'un agent basé G-net AG est un triple *(IM, OM, PU),* où *IM/OM* sont les sections messages entrants et messages sortants qui définissent un ensemble d'unités de traitement et *PU* est la section de méthode utilitaire qui définit un ensemble de méthodes.

3.4 Unité de Traitement messages (MPU)

Une unité de traitement de message est un triple (*P,T, A*) ; où P est un ensemble de places, composé de trois places spéciales : place entrée *(EP),* place de choix instancier *(ISP)* et place de choix message *(MSP).* Chaque *MUP* a une seule *EP* et une seule *MSP* mais peut contenir de multiples *ISP*.

T est un ensemble de transitions, chaque transition peut être associée avec un ensemble de gardes.

A est un ensemble d'arcs définis comme suit :

$$((P-(MSP\})\times T).(T\times(P-\{EP\}).$$

3.5 Méthode utilitaire

Une méthode Utilitaire (U-Methode) est un triple *(P, T, A),* où *P* est un ensemble de place consistant trois places spéciales :

- Place entrée *(EP),* place de choix instancier *(ISP)* et place return *(RP).* Chaque *méthode* à une seule *EP* et une seule *RP*, mais peut contenir de multiples *ISP*.
- *T* est un ensemble de transitions, et chaque transition peut être associée avec un ensemble de gardes.

  A est un ensemble d'arcs définis comme suit : $((P-(RP\})\times T).(T\times(P-\{EP\}).$

4 Problème de VOTE modélisé par G-net

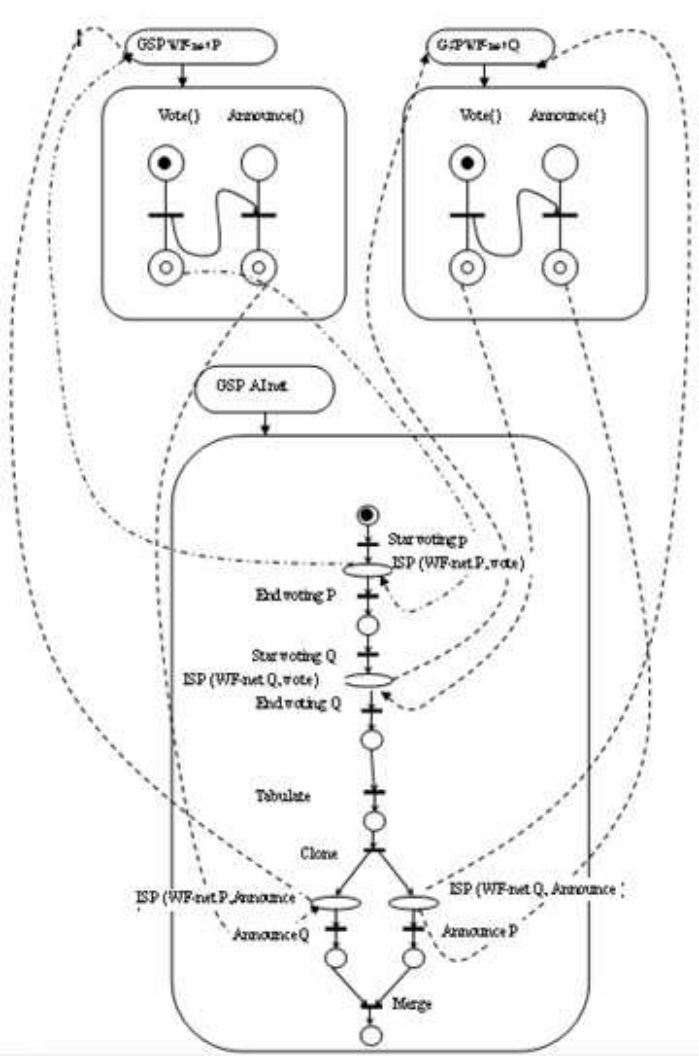

Figure 22 : G-net pour IOWF-net de Vote

Dans la figure 22 on distingue que: celle-ci contient deux WF-net. Chacun possède deux méthodes Vote () et Annonce (). Respectivement la première gère l'opération vote, ensuite la méthode annonce et affiche les résultats. La deuxième partie d'AI-net, est celle qui assure l'interorganisationnel entre les WF-nets.

Le processus d'exécution commence par une transition qui indique le démarrage de l'opération de vote de *p*, on fait un appel alors à la méthode Vote () dans GSPWF-net par ISP (WF-netp, Vote ()). Après son exécution, cette méthode dans GSPWF-net*p*, retourne les résultats pour que la transition EndVoting*p* puisse être franchissable. Après le vote de p, il répète avec la même procédure avec GSPWF-net*q*. A la suite de cette opération, on passe à l'étape de tabulation des résultats. Ensuite, on effectue une transition *clone* qui fournit l'exécution parallèle de l'opération Annonce comme suivant : Après le clonage, faire un appel simultanément aux Annonce () dans GSPWF-net *p* et GSPWF-net *q* par ISP (WF-netp, Annonce( )) et ISP(WF-netq,Annonce( )).

Le retour des différents Annonces des ISP (WF-netp, q) par les clones du même agent, une procédure de migration de ces clones est effectuée enfin d'exécution.

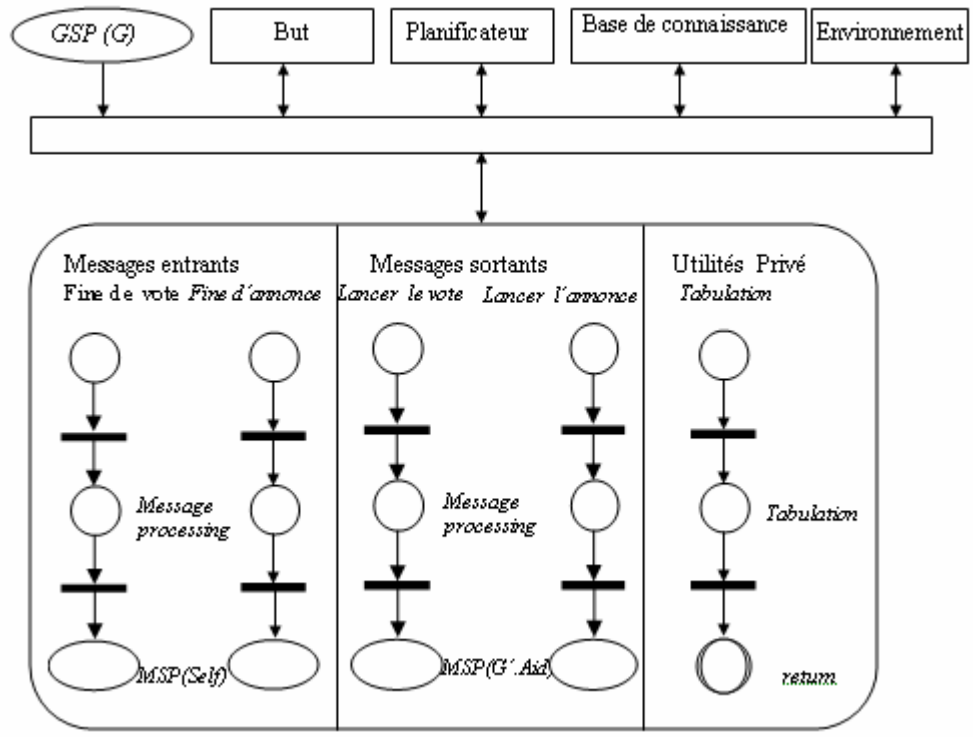

Figure 23 : G-net orientée agent pour IOWF-net de Vote [13]

Comme on a déjà mentionné précédemment le G-net orienté agent est formé de six composants: GSP est l'interface de communication entre G-net. GP : la place qui définit le but. KB : la place qui définit la base de connaissances EP : la place qui définit l'environnement d'agent. Enfin PLANNER : la place qui définit le planificateur. Les cinq derniers composants devraient effectuer une opération de mise à jour après chaque exécution d'une tâche.

On note dans le composant planificateur trois notions de base: -les messages entrants et les messages sortants. Ils sont considérés comme deux ports qui offrent la possibilité d'envois et la réception des messages. Les troisièmes notions 'utilités-privées' qui contiennent toutes les actions privées de l'agent. Dans la figure 23 l'agent commence par l'envoi d'un message qui invoque la méthode *vote ()* dans le GSPWF-net*p*. Ce message « jeton » est passé à la place associée aux messages sortants pour être envoyé à GSPWF-net*p*. Après la réception du jeton au franchissement du transition, *Vote ()* celui-ci fait automatiquement. Ce qui en résulte deux arcs. Le premier vers la transition Annonce pour attendre l'appel de la méthode ISP (WF-net*p*, *annonce* ()), le deuxième vers GSP V qui est réceptionné par la place associée aux messages entrants. Cette place envoie un jeton pour franchir la transition

EndVoting*p*. Ensuite, cette même procédure s'effectue avec WF-net*q*. À la fin du franchissement de la transition EndVoting*q*, on passe à l'exécution de la transition *Tabutat.* Après cela, l'opération du clonage exécute simultanément la phase d'annonce.

Enfin, s'effectue le franchissement de la transition M*erge*. Après la processus de migration, un message « jeton » est alors envoyé à la transition *UPDATE* pour faire la mise à jour de BUT, PLAN, LA BASE DE CONNAISSENCE, L' ENVIRONEMENT.

On applique l'héritage pour le modèle G-net orienté agent, on obtient les résultats suivants [3] :

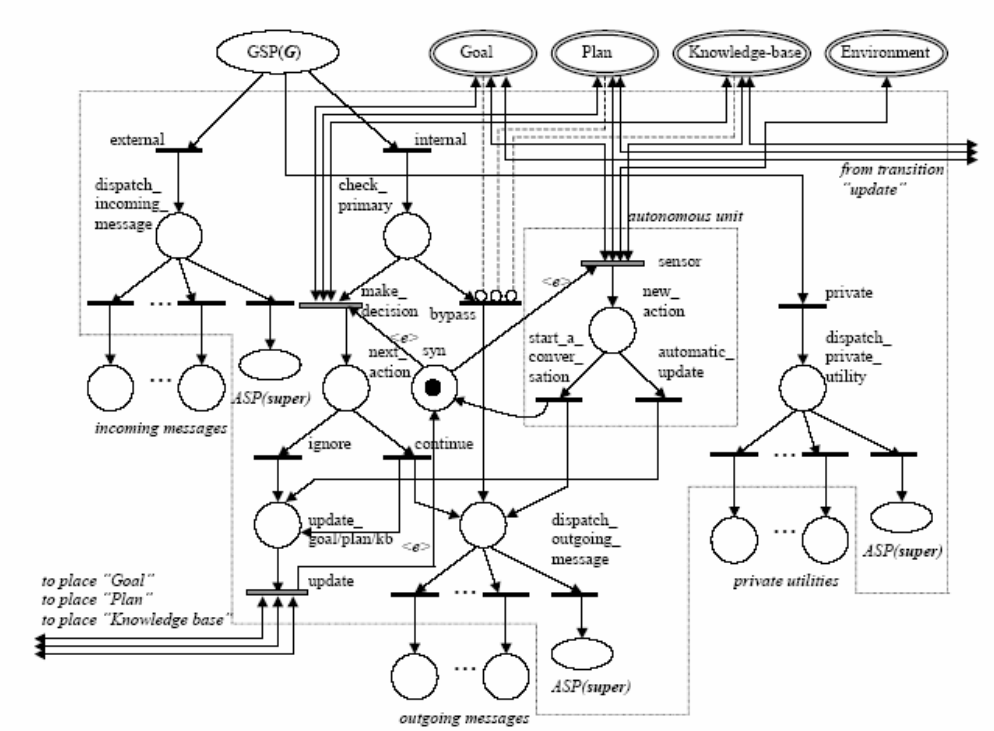

Figure 24 : le module de planificateur de subclasse[3]

On remarque qu'il existe des ***ASP (super)*** qui signifient ce module de planificateur de subclasse qui est capable d'hériter toutes les méthodes/MPU de la classe supérieure pour expliquer comment faire l'interconnexion entre deux classes superclasse et subcalsse. On présente la conception pour agent mobile intelligent dans le problème de vote.

5 Le model G-net orientée agent pour la classe agent mobile intelligent

On considère que l'on a un objet de classe agent mobile (OAM).Cet objet reçoit un message de ask-authcode. Ce message est alors envoyé par un objet de classe agent facilitateur (OAF).

Le jeton mTkn avec étiquette **externe** est déposé dans le GSP de la première subclasse de OAM (c'est à dire le GSP correspond à la classe d'agent mobile intelligent (AMI)). Ensuite la transition externe de module planificateur dans AM est franchie. Ce franchissement doit déposer le jeton vers la place de « message entrant » dans cette partie. Le MPU pour ask-authcode est défini dans la structure interne du AM, le jeton sera déposé dans la place d'entrée du MPU après le traitement de ce message,

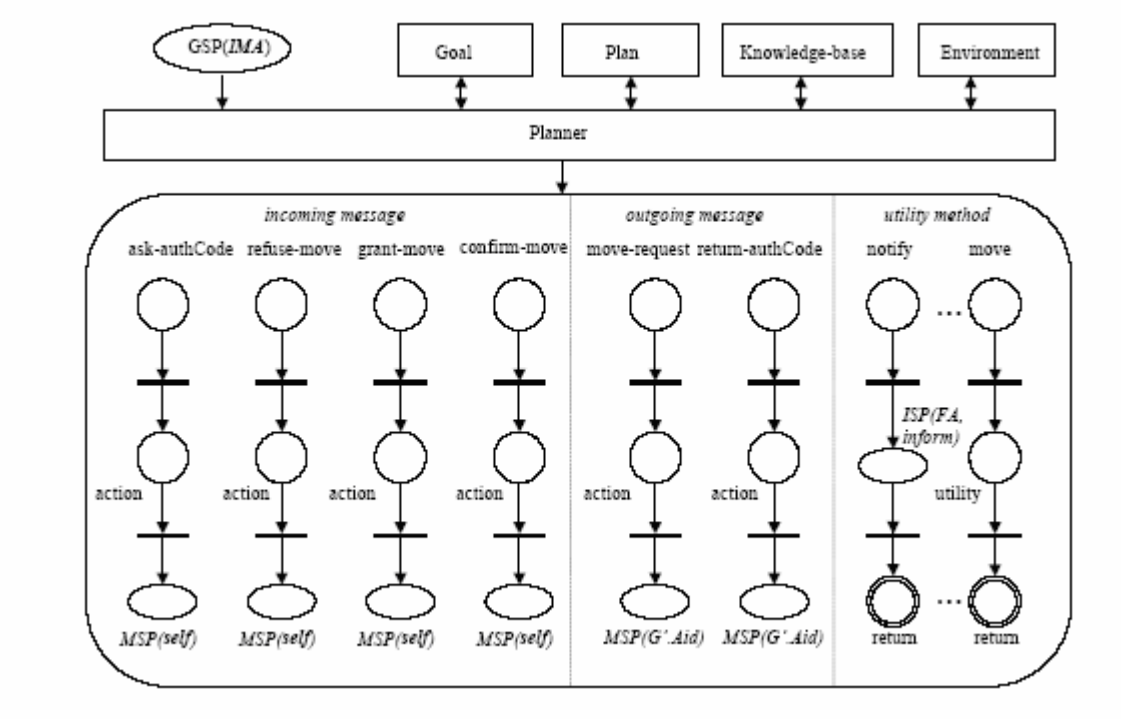

Figure 25 : le modèle G-net orienté-agent pour classe Agent Mobile Intelligent (AMI)[13]

Le MSp (self) doit changer l'étiquette de mTkn de sa forme externe vers la forme interne, et par la suite le jeton sera envoyé vers GSP de AMI. Dans ce cas, la transition interne dans le module planificateur du AM est franchie. Le jeton sera alors déplacé vers la place check-primary. A ce point les places spéciales but, plan, la base de connaissances et environnement donnent la nature de l'action suivante par le franchissement de la transition mak-decision qui offre deux possibilités (annulation, continuation) de l'action. Si le choix est annulation, les places but, plan, base de connaissances font la mise à jour. Après cela, un nouveau jeton sera construit mTkn avec étiquette interne et sera déposé dans la place « duspath-message-sortant ». Le nouveau jeton prend un message appelé return-authcode suivant le protocole défini dans la figure 1. A ce moment là le jeton est déposé dans la place d'entrée de MPU (MPU correspondant à return-authcode et qui sont définies dans AMI). Après le traitement de ce message, le mécanisme MSP (G'.Aid) change l'étiquette du jeton

mTkn de sa forme interne vers la forme externe et transfère le jeton vers les GSP de l'agent récepteur (l'objet agent facilitateur (OAF))

6 La conception pour agent mobile intelligent dans le problème de vote

Dans la figure 26 on distingue les agents avec la classification hiérarchique. La classe d'agent mobile intelligent (AMI) est définie comme la superclasse capable de connecter avec la classe d'agent facilitateur intelligent (AFI). La fonctionnalité de la classe (AMI) peut être héritée par l'agent subclasse comme la classe d'agent mobile IA-net et classe mobile WF-net (vote). Avec l'héritage, la classe d'agent mobile IA-net comme la subclasse de la classe d'agent mobile (AM) réutilise MPU/méthodes définies dans la structure interne d'AM de la même façon. La classe WF-net (vote) hérite toutes les MPU/méthodes du AM.

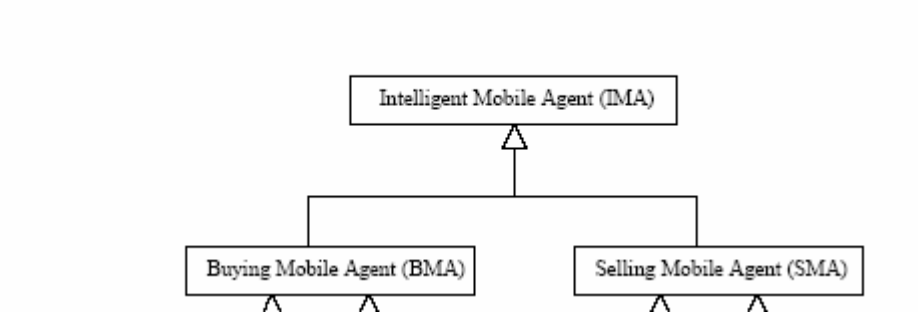

Figure 26 : les deux classes (AM IA-net, AM WF-net (vote)) sont des subclasse et AMI est superclasse [13]

Maintenant on présente un exemple d'une question à poser : comment réutiliser les MPU/méthodes de travail. On a un objet d'agent IA-net qui reçoit un message de forme ask-authcode envoyé par l'objet d'agent facilitateur (OAF). Le jeton mTkn sera déposé dans le GSP de la classe agent mobile IA-net, On commencera par le franchissement de la transition externe dans du module planificateur du (AMIA-net) et le jeton doit être déplacé vers la place « dispath – message-entrant ». Dans ce cas le MPU de ask-authcode n'est pas défini dans la structure interne de (AMIA-net). Pour cela le mTkn sera déplacé vers la place ASP (Super). Dans cet exemple la superclasse est unique « la classe d'agent mobile ».

A ce moment là, le jeton doit entrer dans le MPU du ask-authcode, après le traitement de ce message. MSP (self) procède au changement de l'étiquette du jeton de forme externe vers interne, et l'envoie vers le GSP du (AM IA-net). Depuis l'arrivée du message de jeton, la transition interne dans le module planificateur du AM IA-net sera franchie, le jeton mTkn sera déplacé vers la place check-primay. A ce point les places spéciales but, plan, base de connaissances et environnement donnent la nature de l'action suivante par le franchissement de la transition mak-decision qui offre

deux possibilités (Annulation/continuation). Si l'on choisit la continuation, les places spéciales le but, plan et base de connaissances font la mise à jour. Cette mise à jour finit par construire un nouveau jeton avec étiquette interne qui sera déposé vers la place 'disptch-message-sortant'. Ce jeton prend un message appelé 'return-authcode' suivant le protocole défini dans la figure1.

On ne trouve pas le MPU de return-authcode défini dans AM IA-net. Donc, le nouveau jeton mTkn sera envoyé vers les GSP de AM. Après l'arrivée de ce jeton à le GSP de AM, le franchissement de la transition interne dans le module planificateur de AM se fait automatiquement. A ce moment-là le jeton est déposé dans la place d'entrée de MPU qui correspond au message return-authcode au niveau d'AM. Après l'achèvement du traitement de ce message, le mécanisme MSP (G'.Aid) effectue un changement sur l'étiquette de jeton de la forme interne vers la forme externe et transfère le jeton vers le GSP d'agent récepteur (OAF).

7 La vérification du modèle G-net basé agent

L'un des avantages de la construction d'un modèle formel pour des agents dans la conception basée sur les agents est d'assurer : une conception correcte qui rencontre quelques spécifications. Une conception correcte d'agents a au moins les propretés suivantes :

- *L3-live* : n'importe quel acte communicatif peut être exécuté autant de fois qu'on veut.
- *Concurrent* : un nombre de conversations supérieures de l'agent peuvent avoir lieu en même temps.
- *Effective* : un protocole de communication agent peut être tracé correctement dans le modèle agent.

Nous utilisons un outil de Petri, appelé INA (Integrated Net Analyzer) pour analyser et vérifier automatiquement nos modèles agent. Nous utilisons un exemple d'un modèle simplifié de réseau de Petri pour l'interaction entre un seul agent d'achat et deux agents de vente. L'outil INA est un outil interactif d'analyse qui intègre un grand nombre de méthodes puissantes d'analyse des réseaux Place/Transition (P/T). Ces méthodes comprennent l'analyse :

(1) des propriétés structurelles telles que les boundedness structurelles analyse invariante T et P ;

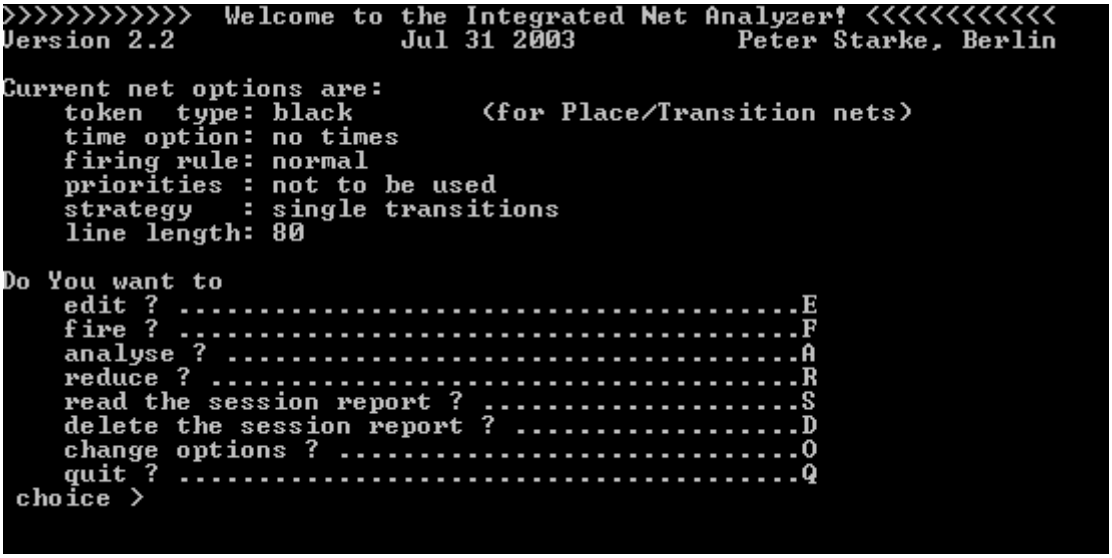

Figure 27 : Le menu de INA.

(2) des propriétés de comportement, telles que boundedness, sécurité, vivacité, impasse libre.

(3) un modèle de vérification.

Nous nous concentrons sur la vérification des propriétés comportementales et le modèle de vérification.

Pour vérifier que les modèles G-net basé agent sont corrects et respectant les propriétés ci- dessous. On réduit en premier lieu les modèles G-net base agent à un réseau de Petri ordinaire comme suit :

1- Simplifier le module but, module plan et le module base de connaissance en places ordinaires avec des jetons d'ordinaire.
2- Enlever les sections 'méthodes utilitaires'.
3- Simplifier les jetons mTKn en jetons ordinaires.
4- Utiliser une réduction réseau pour simplifier le réseau de Petri correspondant aux méthodes MPU en une seule place.
5- Vérifier notre modèle G-net base agent par INA.

8 Conclusion

L'extension de l'approche orientée objet à l'approche orientée agent est un processus naturel de tous les programmes. Ceci est possible puisqu'il y a une similarité entre les deux paradigmes objet et agent. Dans ce chapitre, nous proposons un modèle G-net basé agent qui supporte la modélisation agent. Nous construisons un G-net basé agent pour IOWF-net de Vote.

1 Introduction

L'un des avantages de la construction d'un modèle formel pour les agents dans une conception orienté agent est de contribuer à assurer une bonne conception qui répond à certaines spécifications. Une bonne conception de l'agent devrait satisfaire certaines exigences clés, telles que la vivacité et la concurrence. Aussi, certaines propriétés des mécanismes spéciaux tel que le mécanisme d'héritage, doivent être vérifiés afin d'assurer son bon fonctionnement. Les réseaux de Petri offrent une technique prometteuse appuyée par un outil pour vérifier la conception. Dans ce chapitre, nous utilisons deux outils de Petri, appelé WoPeD (**Wo**rkflow **Petri** net **D**esigner) et INA. Ensuite analyser et vérifier automatiquement nos modèles agents. Nous utilisons un exemple d'un modèle simplifié de réseau de Petri pour l'interaction entre un seul agent IA-net et deux agents de vote.

L'outil INA est un outil interactif d'analyse qui intègre un grand nombre de méthodes puissantes d'analyse des réseaux Place/Transition (P/T). Ces méthodes comprennent l'analyse : (1) des propriétés structurelles telles que les boundedness structurelles, analyse invariante T et P. (2) des propriétés de comportement, telles que boundedness et vivacité.

Au début, nous utilisons l'outil WoPeD pour vérifier les propriétés structurelles telles que les boundedness structurelles et des propriétés de comportement tels que boundedness et vivacité pour vérifier la soudness (optimal) de l'IOWF-net de vote. A partir de la définition IO-soudness dans le chapitre 1, il faut vérifier localement et globalement optimal un IOWF c'est-à-dire : 1- tous les Workflow nets locaux sont optimaux, 2- U (IOWF) est optimal.
Il y a trois Workflow nets locaux (IA-net et WF-net vote p, q).

2- Utilisation de WoPeD

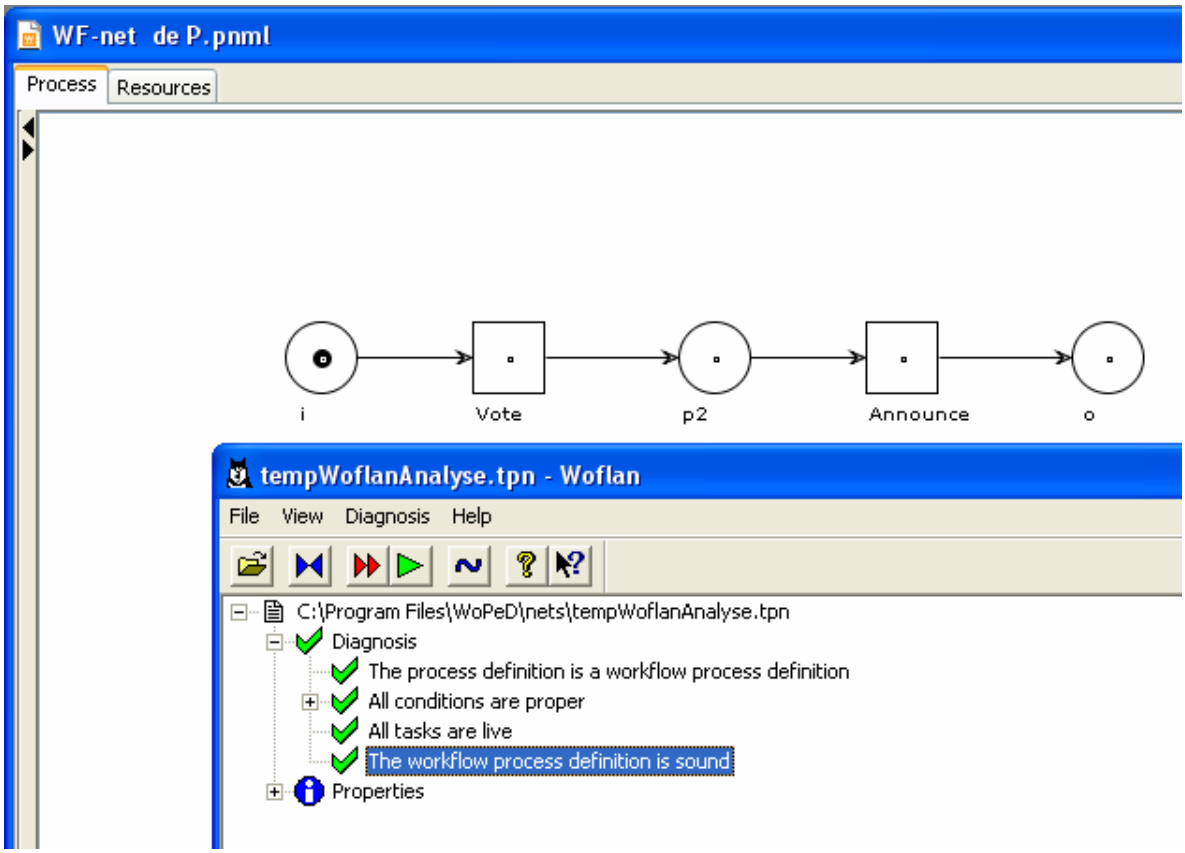

Figure28 : Résultat de vérification du WF-net de p,q

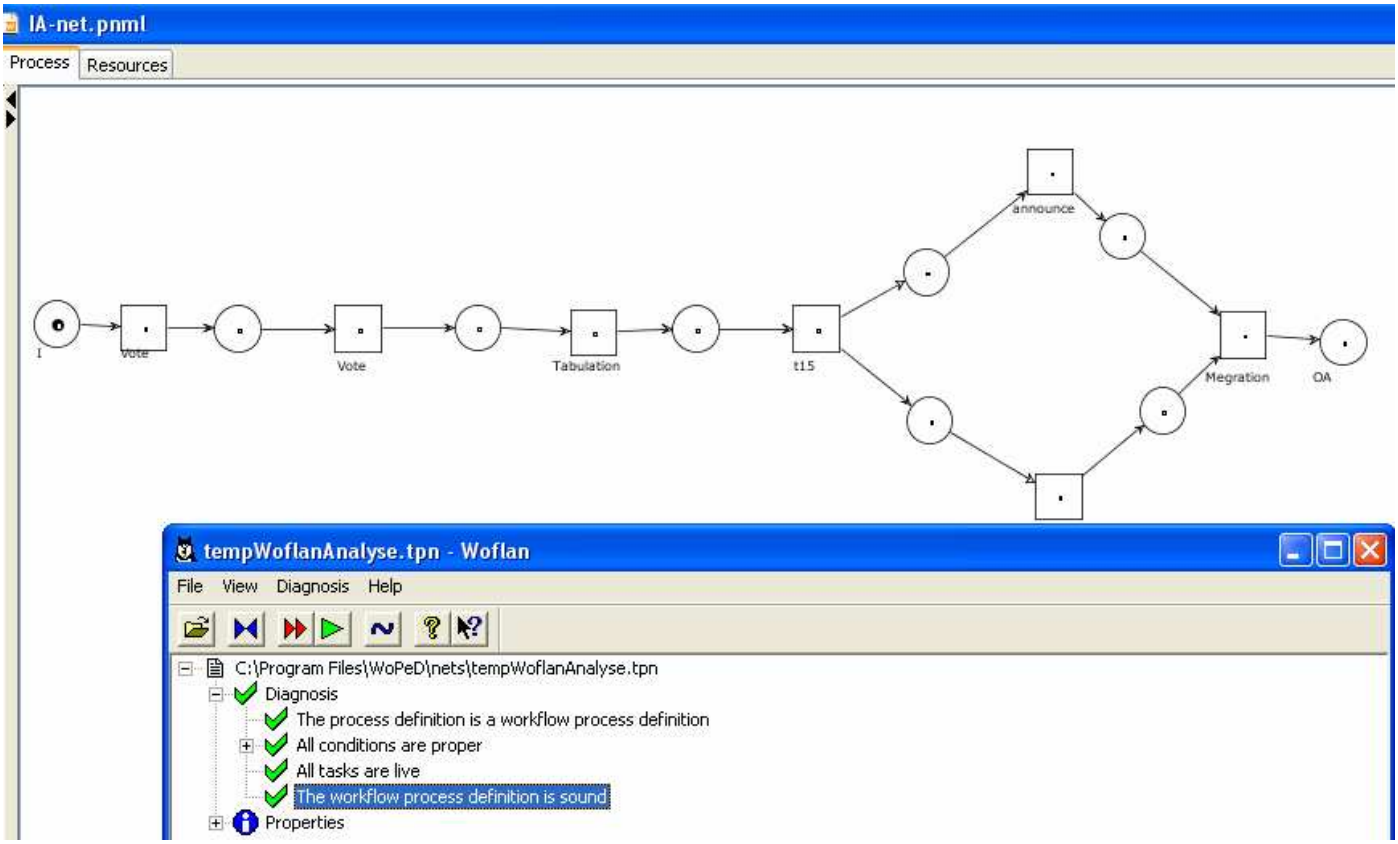

Figure 19 : les résultats de vérification du IA-net

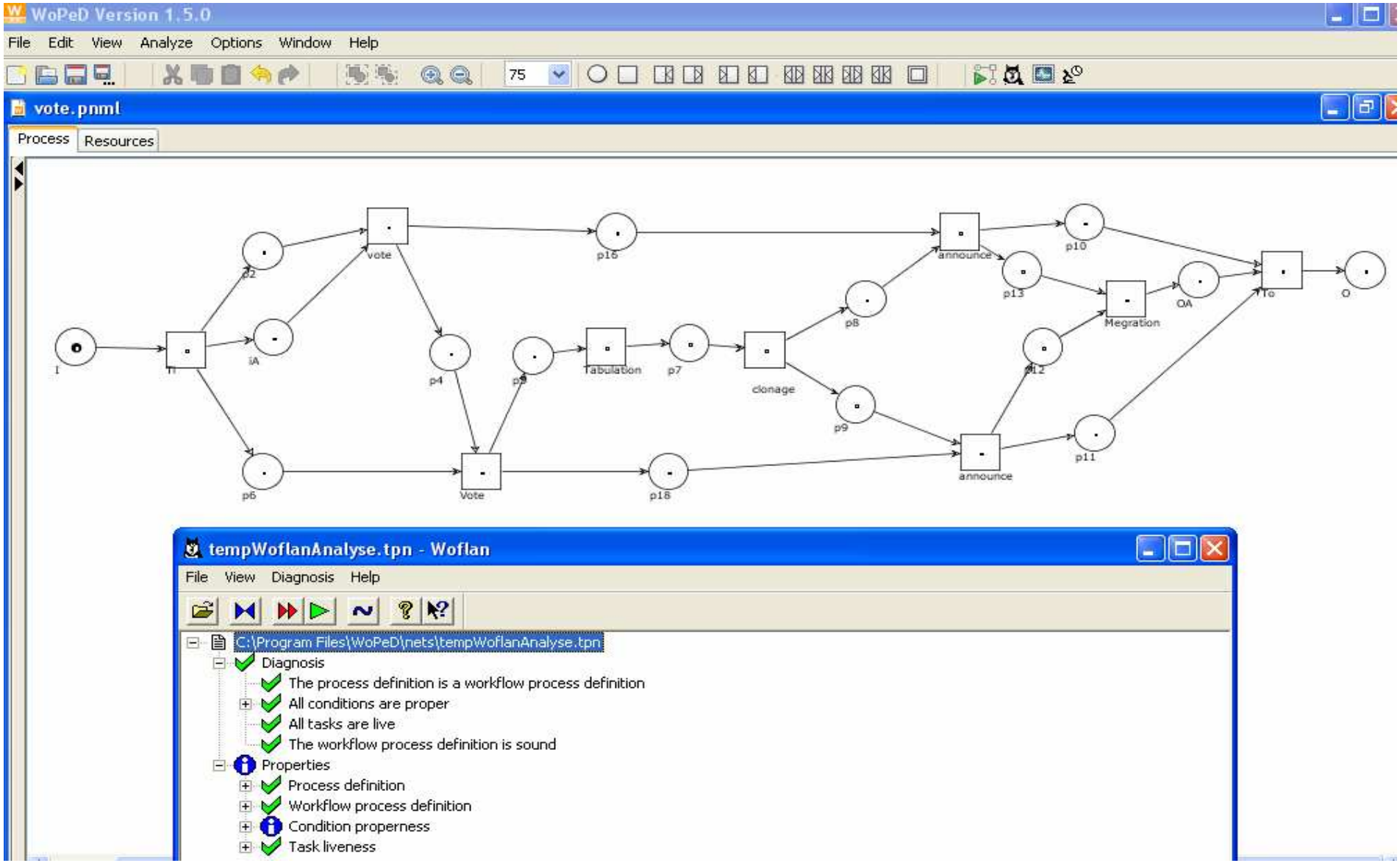

Figure 20 : les résultats de vérification du U(IOWF)

3 Utilisation d' INA

A partir du théorème 1 [12] dans le chapitre 1, on démontre si le WF-net est optimal (sound) ou non. Pour cela, il faut vérifier les deux propriétés suivantes : Liveness, Boundedness du *short-circuited WF-net.*

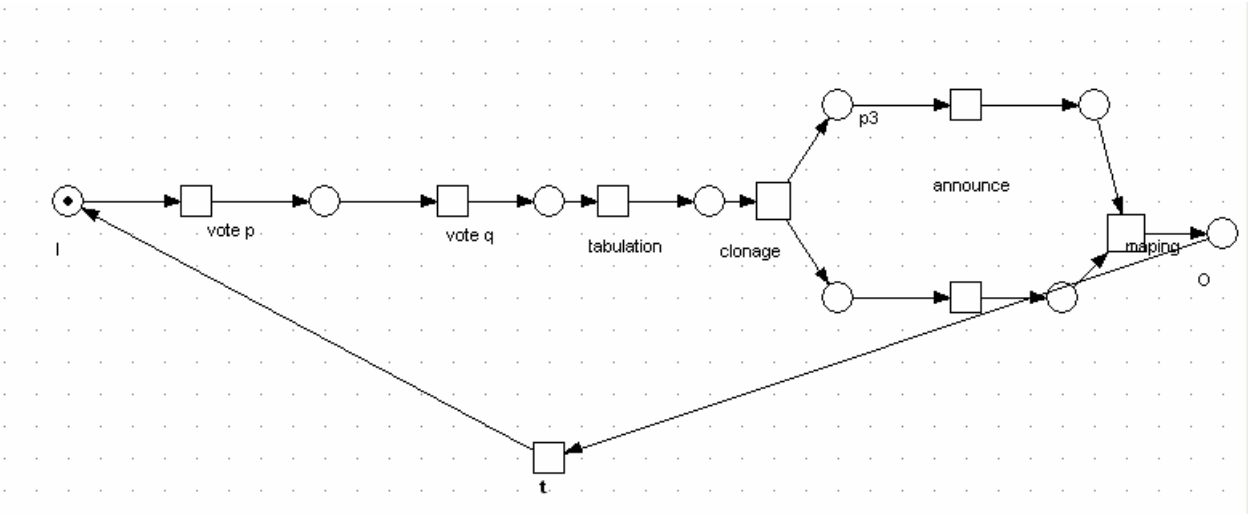

Figure 31 : sort-Circuited IA-net

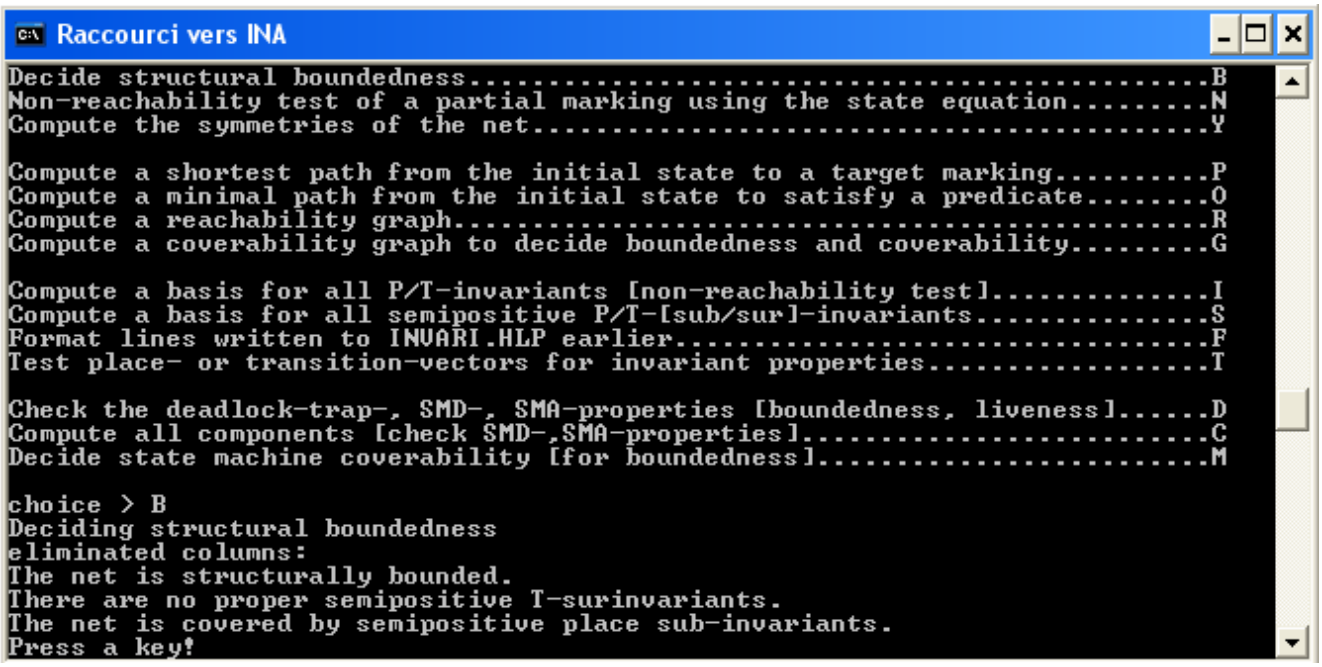

Figure 32 : Résultats de vérification de la propriété Boundedness de sort-Circuited IA-net

```
Check the deadlock-trap-, SMD-, SMA-properties [boundedness, liveness].....D
Compute all components [check SMD-,SMA-properties].........................C
Decide state machine coverability [for boundedness].........................M

choice > D
Current name options are:
     transition names not to be written
     place names not to be written
.....................................Reset options? Y/N N
We check the deadlock-trap-property.

processed candidates:          21
The deadlock-trap-property is valid.
The net has no dead reachable states.
The net is live.
The net has no dead transitions at the initial marking.
The net is live, if dead transitions are ignored.
The net is covered by semipositive T-invariants.
The net is state machine decomposable (SMD).
The net is coverable by state machines (SMC).
The net is covered by semipositive P-invariants.
The net is reversible (resetable).
Executing SMA-test..
The net is state machine allocatable (SMA).
The net is covered by 1-token SCSM's.
The net is live and safe.
The net is safe.
Press a key!_
```

Figure 33 : Résultats de la vérification de la propriété Liveness sort-Circuited IA-net

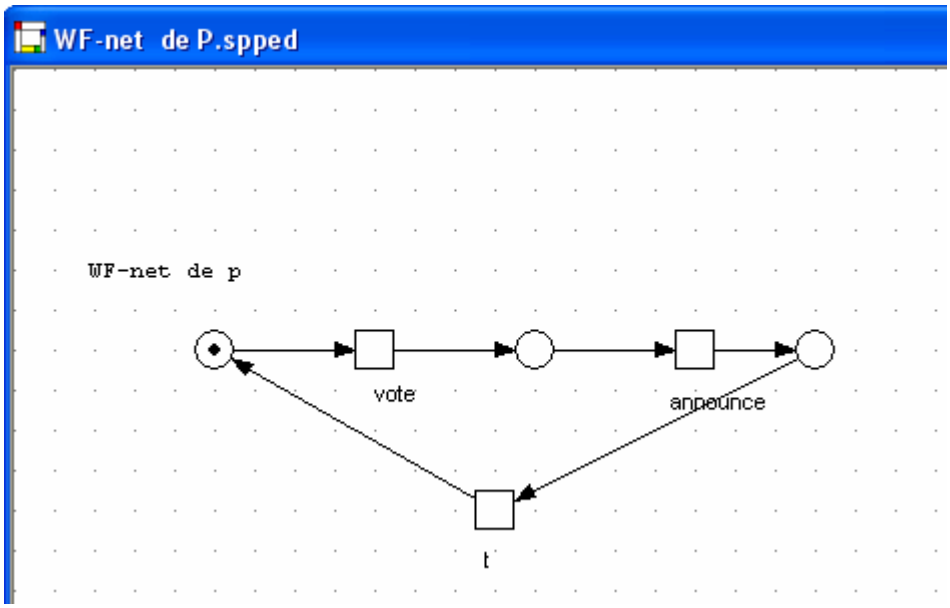

Figure 34 : sort-Circuited WF-net de p/q

```
The net is structurally bounded.
The net is bounded.
There are no proper semipositive T-surinvariants.
The net is subconservative.
The net is a state machine.
The net is extended free choice.
The net is extended simple.
The net is free choice.
The net is marked.
The net is marked with exactly one token.
The net is a marked graph.
The net has a non-blocking multiplicity.
The net has no nonempty clean trap.
The net has no transitions without pre-place.
The net has no transitions without post-place.
The net has no places without pre-transition.
The net has no places without post-transition.
The net is connected.
The net is strongly connected.
The net is state machine allocatable (SMA).
The net is state machine decomposable (SMD).
The net is coverable by state machines (SMC).
The net is covered by semipositive P-invariants.
The net is live.
```

Figure 35 : Résultats de la vérification des propriétés Liveness et boundedness de Sort-circuited WF-net de p/q

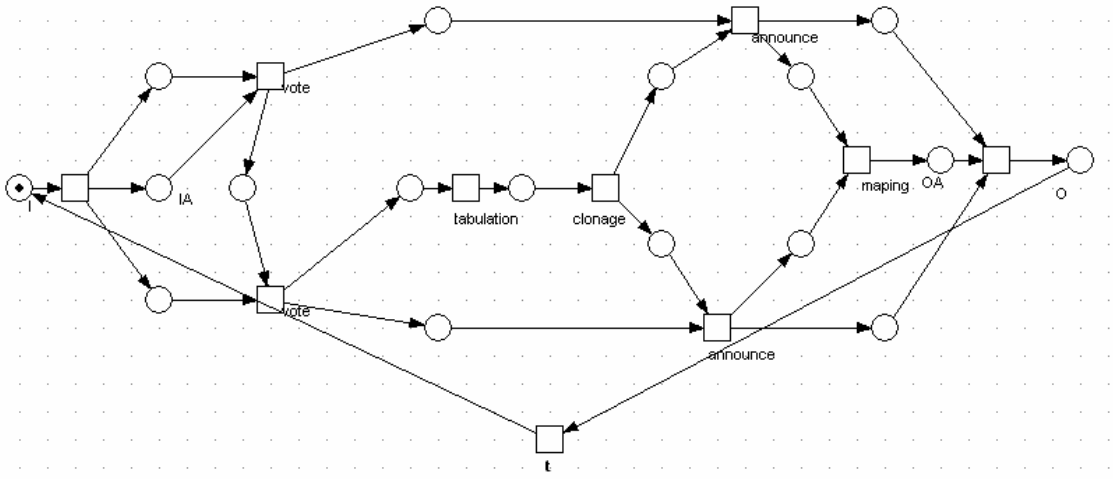

Figure 36 : Sort-Circuited UIOWF-net

```
The net is structurally bounded.
The net is bounded.
There are no proper semipositive T-surinvariants.
The net is subconservative.
The net is a state machine.
The net is extended free choice.
The net is extended simple.
The net is free choice.
The net has places without pre-transition.
The net is not state machine decomposable (SMD).
The net is not state machine allocatable (SMA).
The net is not strongly connected.
The net is not covered by semipositive T-invariants.
The net is not live.
The net is not live and safe.
```

Figure 37 : Résultats de la vérification des propriétés Liveness et boundedness de Sort-Circuited UIOWF-net

4 Un modèle simplifié de réseau de Petri d'un agent IA-net et agents de WF-net (Vote)

L'interaction entre un agent IA-net et deux agents WF-net (vote) peut être modélisée comme un réseau présenté dans la Figure 38. Le tableau V et VI fournissent une explication de chaque place et transition dans la figure 38. Pour obtenir ce modèle de réseau nous utilisons une place SGP pour représenter chaque agent de WF-net (vote).

Cela est possible parce qu'un modèle G-net orienté agent peut être abstrait comme une seule place SGP, les modèles agents ne peuvent pas interagir les uns avec les autres à travers les places SGP. Nous simplifions cet agent comme suit:

1. Puisque les places spéciales de but *(Goal),* de plan (*Plan)* et de base de connaissances (*Knowledge-base)* ont les mêmes interfaces avec le module planificateur dans la classe agent, nous les fusionnons en une seule place *Goal* /Plan/kb. En outre, nous avons simplifié cette place fusionnée but/ plan/ kb et la place de l'environnement (*environment)* des lieux ordinaires avec des jetons ordinaires.

2. Enlever les sections méthodes utilitaires.

3. Nous simplifions les jetons mTkn comme des jetons ordinaires. Bien que cette simplification entraînera un graphe d'accessibilité de nos réseaux de Petri transformés pour devenir plus grands. Ceci simplifie les jetons de message : ce qui nous permet d'ignorer les détails de messages.

4. Nous utilisons une réduction de réseaux de Petri, afin de simplifier les réseaux de Petri correspondant à un MPU/Méthode comme une seule place.

5. On utilise l'accession du monde fermé et envisageons un système qui ne contient que trois agents: un agent IA-net et deux agents de WF-net (vote). Si un système contient plus de trois agents peuvent être vérifié de la même manière.

Il est à noter que le modèle réseaux de Petri simplifié préserve la plupart des propriétés comportementales d'un modèle G-net orienté agent. On utilise WoPeD comme un outil existant de réseaux de Petri, comme éditeur et vérificateur de ce modèle.

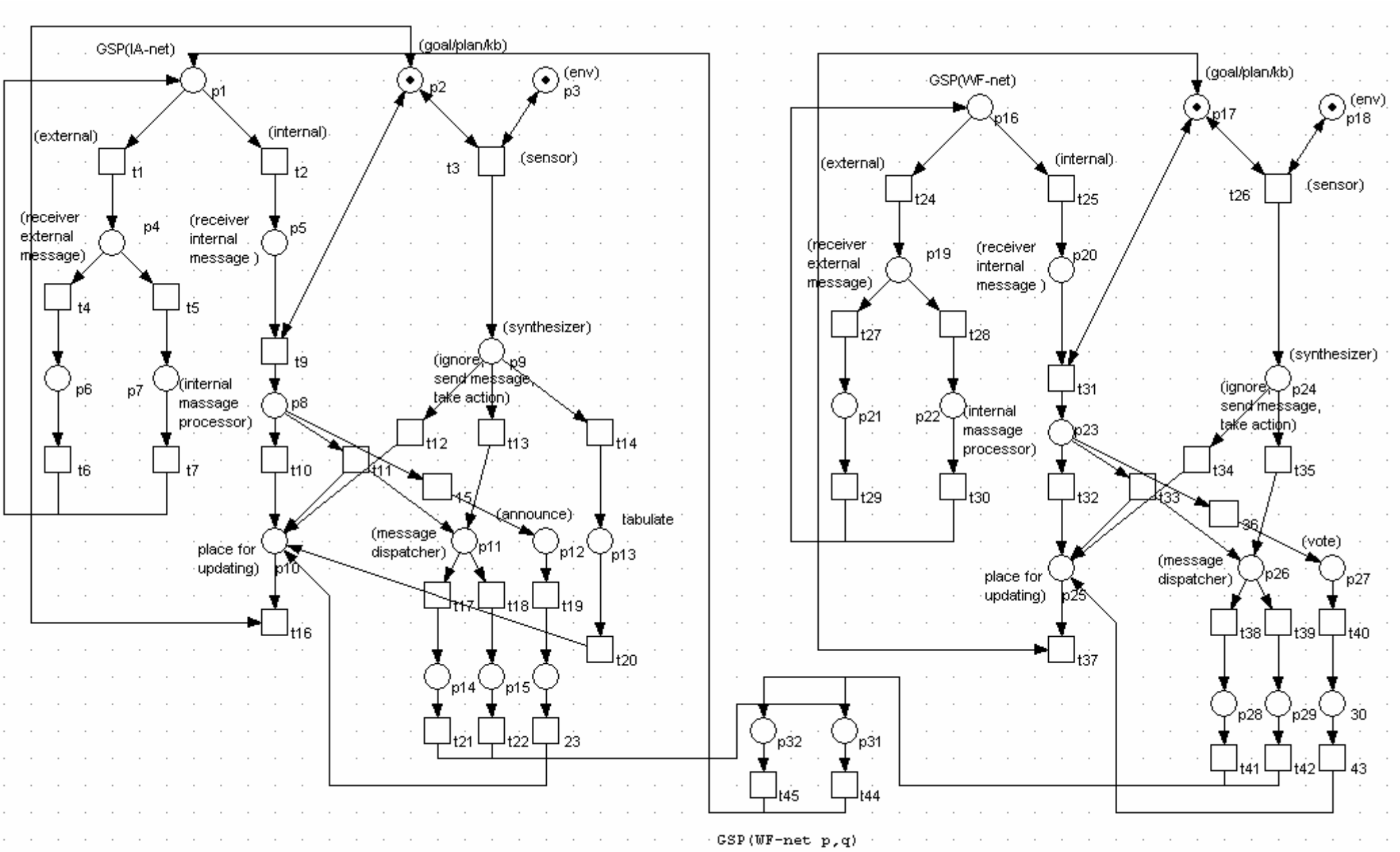

Figure 38 : Un nouveau modèle de l'IA-net et deux agents WF-net (vote)

Place	description
p1/p16 La place SGP de -agent IA-net/ agent de WF-net(vote). p2 / p17 La place fusionnée du module but, plan et de base de connaissance de agent IA-net/ agent de WF-net(vote). p3 / p18 La place pour le module d'environnement d'agent IA-net/ agent de WF-net(vote). p4/p19 La place d'envoi des messages entrants. p6, p7 /p21, p22 les places des messages récepteurs de L'annonce /vote. p9 / p24 La place de synchronisation de prise de décisions p11/p26 La place d'envoi des messages sortants. p10 / p25 La place de mise à jour de l'état mental de l'agent. p14, p15/ p28, p29 Les places des messages envoyeurs de L'annonce /vote. P12 La place de Les utilités services de l'agent IA-net. P13/27,30 La place des utilités privées de l'agent IA-net/agent WF-net (vote).	
Transition	**description**
t1/t24 La transition externe qui se déclenche lorsque le jeton de la SPG a une étiquette de l'extérieur. t2/t25 La transition interne qui se déclenche lorsque le jeton de la SPG, a une balise de contrôle interne. t3/t26 La transition de scénario d'exaction. t4, t5 La transition de message externe t6, t7/ t29, t30 Les transitions des messages extérieurs (concerne les messages reçus MPU_S avec le t4/t5 et le t27/t28) t14, t27 La transition des utilités privées t20, t43 La transition des utilités privées (concerne l'utilité privée MPU_S avec le t14 et le t27). t9, t31 La transition de synchronisation. t16, t37 La transition de mise à jour t17, t18/ t38, t39 La transition d'envoyer un message T21, t22/t41, t42 Les transitions d'envoyer des messages (concerne les messages envoyés MPUS avec le t17/t18 et le t38/t39)	

Tableau 1 : DESCRIPTION DES PLACES et TRANSITIONS

Pour vérifier le bien-fondé de notre modèle d'agent, nous utilisons des définitions de certaines principales propriétés de comportement de réseaux de Petri telles que la boundedness et la vivacité (Liveness).

5 Définition Boundedness

Un réseau de Petri (N, M0) est dit k-limité ou simplement borné si, le nombre de jetons dans chaque lieu ne dépasse pas un nombre fini k pour tout marquage accessible à

partir de M0.

6 Définition Liveness

Un réseau de Petri (N, M0) est dit à être vivant si, pour tout marquage M est accessible à partir de M0. Il est alors possible de franchir finalement toutes les transitions du réseau par la progression de quelques séquences de tirs.

Avec notre modèle de réseaux de Petri dans la figure 38 à titre de contribution, l'outil WoPeD produit les résultats suivants:

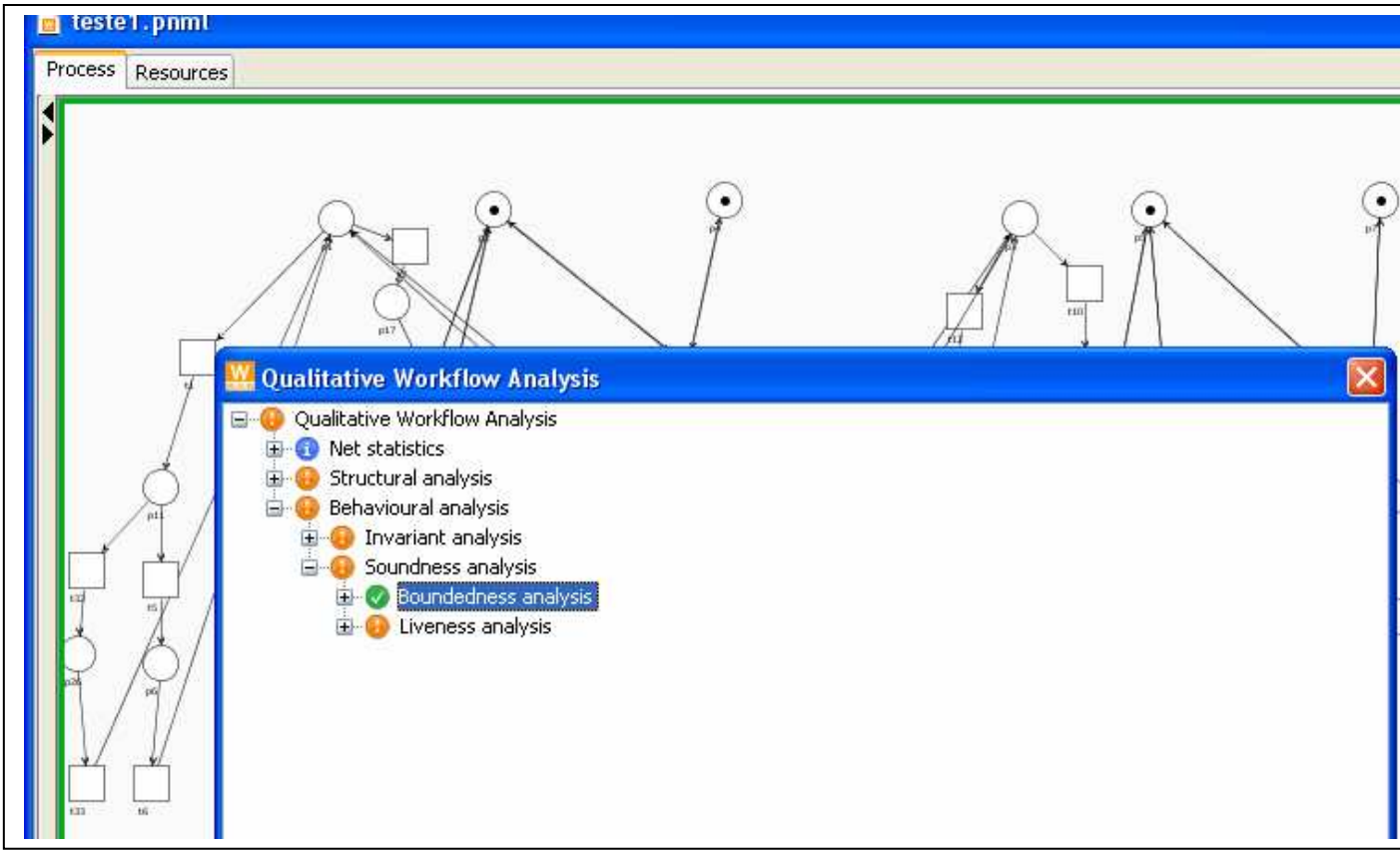

Figure 39 : Un nouveau modèle de l'IA-net et agents WF-net (vote)

Avec notre modèle de réseaux de Petri dans la figure 38 à titre de contribution, l'outil INA produit les résultats suivants:

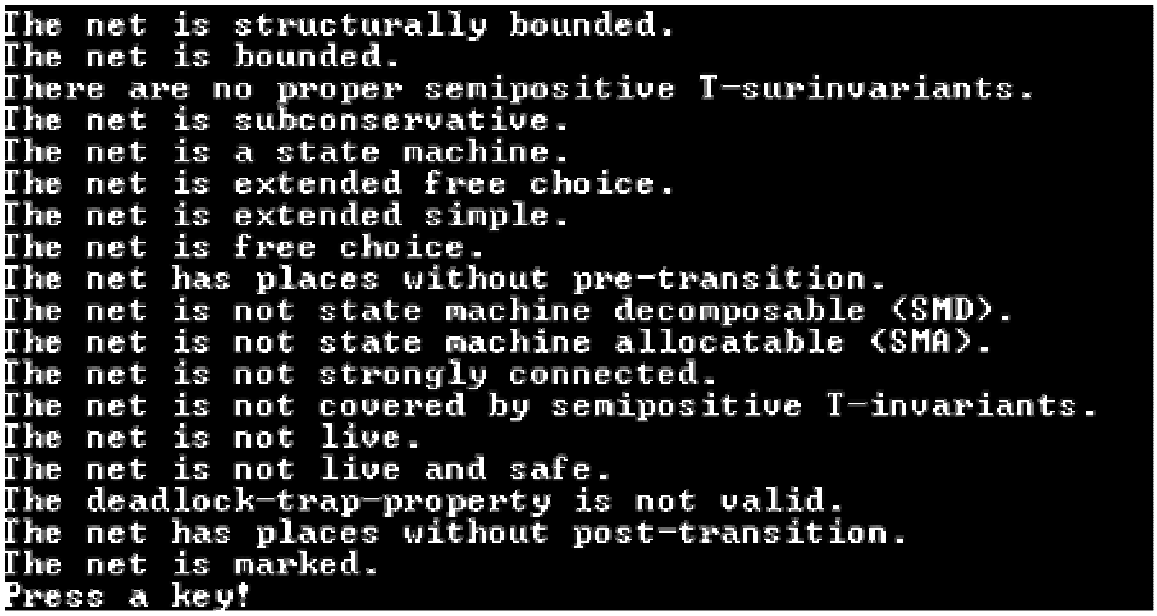

Figure 40 : Un nouveau modèle de l'IA-net et agents WF-net (vote) par INA

7 Conclusion

Nous avons effectué, dans ce chapitre, une étude expérimentale de notre approche. Le problème de VOTE est choisi, ce dernier est modélisé par les réseaux de Petri standards dans le travail de ''Sea Ling , Seng Wai Loke'' [1], cette modélisation ne donne pas de bonne représentation des agents dans notre système (Workflow interorganisationnel), pour cela on dérive vers les réseaux de Petri orienté objet et nous présentons le passage des réseaux de Petri ordinaires vers les réseaux de Petri orienté objet comme solution, avec l'absence des outils de vérification pour les réseaux de Petri orienté objet.

Conclusion générale

Dans ce mémoire nous avons décrit un modèle de G-net orienté agent pour modéliser les agents mobiles dans workflow interorganisationnel. Nous avons vu que ce modèle utilise des notions très liées. Celles-ci sont présentées comme suit :

1- Worklfow Inteorganisational est exposé au chapitre 1. On présente la définition de Workflow ensuite Workflow net qui est composé de plusieurs Workflow. Pour cela, nous donnons d'abord les définitions de base des réseaux de Petri ainsi que les primitives essentielles. Puis, on étudie les agents mobiles lorsque nous expliquons en détail la propriété d'itinéraire qui offre la possibilité de trouver et contrôler le chemin de déplacement d'agent entre les Workflow net. Et enfin, on donne les algorithmes de vérification des Workflow nets et workflow interorganisationnel.

2- Le modèle G-net est exposé au chapitre 2. On présente la définition de G-net et tous les composants structurels. Dans cette partie on conclue à une proposition qui suggère de trouver un model G-net à base d'agent qui supporte la modélisation d'agent.

3- Le modèle G-net orienté agent est exposé au chapitre 3. On présente l'intégration d'approche orienté agent avec G-net. Cette intégration produit un modèle capable de modéliser les agents mobiles qui forment l'interorganisationnel entre les workflow nets.

4- On applique le modèle qui produit dans la phase précédente sur un exemple de vote et on donne les résultats de la vérification par les outils de vérification (INA et WoPeD). Cette vérification se fait après une étape faite manuellement (le passage du modèle G-net orienté agent vers un modèle réseaux de Petri standard).

Les perspectives futures pour notre travail seraient d'automatiser l'algorithme de passage du modèle G-net orienté agent vers un modèle réseaux de Petri ordinaire et ainsi traiter toutes les complexités de ce passage.

Table des matières

Liste des figures

Liste des tableaux

BIBLIOGRAPHIE

[1] Sea Ling, Seng Wai Loke, “Advanced Petri Nets for Modelling Mobile Agent Enabled InterorganizationalWorkfows”. Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS.02).

[2] Yi Deng, S.K.Chang, Jorge C.A. de Figueired et Angelo Perkusich, “Integrating Software Engineering Methods and Petri Nets for the Specification and Prototyping of Complex Information Systems”. Proc. 14th Int’l Conf. Application and Theory of Petri Nets, pp. 206-223, June 1993.

[3] Haiping Xu, et Sol M. Shatz, ”A Framework for Model-Based Design of Agent-Oriented Software”. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 1, JANUARY 2003.

[4] Yangdong Ye, Zundong Zhang, Limin Jia et Honghua Dai, ” Research on Train Group Operation Model in RITS”. Autonomous Decentralized Systems, 2005. ISADS 2005.Proceedings.

[5]A. Perkusich and J. de Figueiredo, ”G-nets: A Petri Net Based Approach for Logical and Timing Analysis of Complex Software Systems, ” *Journal of Systems and Software*, 1997.

[6] W. v. Aalst. *”Interorganizational workflows - an approach based on message sequence charts and Petri nets”*. Department of Mathematics and Computing Science, Eindhoven University of Technology, 1999.

[7] Lotfi BOUZGUENDA.’’*Coordination Multi-Agents pour le Workflow Inter-organisationnel Lâche’’*. Doctorat de L'Université Toulouse I, Spécialité : INFORMATIQUE.2005.

[8] G. Scorletti et G. Binet, GREYC AUTO. *’’Réseaux de Petri’’*. UNIVERSIT CAEN/BASSE–NORMANDIE. 20 juin 2006.

[9] W.M.P. VAN DER AALST. *’’PROCESS-ORIENTED ARCHITECTURES FOR ELECTRONIC COMMERCE AND INTERORGANIZATIONAL WORKFLOW’’. Department of Mathematics and Computing Science, Eindhoven University of Technology.1999.*

[10] Thomas Vantroys, Yvan Peter. *’’Un système de workflows flexible pour la formation ouverte et à distance’’*. Université des Sciences et Technologies de Lille-Laboratoire TRIGONE – Equipe NOCE.2002.

[11] Haiping Xu and Sol M. Shatz.''*An Agent-based Petri Net Model with Application to Seller/Buyer Design in Electronic Commerce''*. Department of Electrical Engineering and Computer Science, The University of Illinois at Chicago. 2001.

[12] Mathias Weske, ''Business Process Management Concepts, Languages, Architecture"s. Hasso Plattner Institut an der Universität Potsdam. Springer-Verlag Berlin Heidelberg 2007.p 277.

[13] Haiping Xu. *''A MODEL-BASED APPROACH FOR DEVELOPMENT OF MULTI-AGENT SOFTWARE SYSTEMS"*. Thesis Doctor of Philosophy in Computer Science in the Graduate College of the University of Illinois at Chicago, 2003