ETUDE DE CAS :Exemples d'application de la méthode de conception d'un superviseur de contrôle pour un processus de production distribué

Dans le chapitre précédent, une méthode de modélisation et de vérification des processus de production en vue de concevoir un superviseur de contrôle a été présentée. Pour montrer comment cette méthode est utilisée, nous proposons dans ce chapitre une application de cette méthode sur deux exemples de processus de production .Le premier est une chaîne d'emballage des produits et le deuxième est une chaîne d'embouteillage. La structure physique de chaque chaîne, ainsi que les caractéristiques des éléments physiques, sont présentées. Ensuite, leur modélisation et vérification sont réalisées à l'aide des outils nécessaires présentés dans le chapitre précédent.

Pour chaque exemple, nous essayons de donner deux variantes pour comparer les résultats obtenus.

3.1 Exemple1 : Chaîne d'emballage de produits

3.1.1 Présentation

La Figure suivante (Figure 3.1) illustre une ligne d'emballage de produits simple, composée de quatre composants : une ceinture (belt), un dévidoir de feuilles d'emballage (Film), un appareil de collage de feuilles (welder) et un coupoir (cutter). Un produit à emballer (ou un JOB) et la feuille d'emballage qui a une étiquette imprimée (ou TAG) sont identifiés par leurs supports c'est-à-dire la ceinture et le dévidoir de feuilles, respectivement. Le produit (JOB) et l'étiquette (TAG) sont disposés suivant la ceinture et le dévidoir Chacun de ces modules peut être contrôlé séparément. Par exemple, la ceinture est mise en marche par un moteur [Bordbar, 2000].

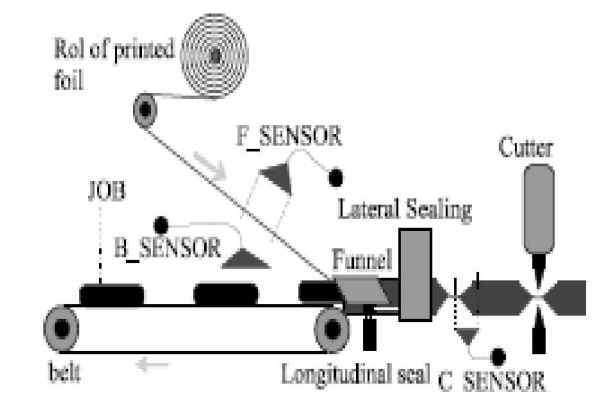

Figure 3.1 Exemple d'une chaîne d'emballage [Bordbar, 2000].

3.1.2 Modélisation avec UML

3.1.2.1 Diagramme de cas d'utilisation

Le fonctionnement de la chaîne d'emballage de la Figure 3.1 est décrit comme suit :

Quand un produit arrive (new JOB) sur son support (belt), l'état de l'étiquette (TAG) est évalué. Si l'étiquette est sur son support (film), l'opération d'emballage peut avoir lieu. Cependant, si l'étiquette est hors la zone d'emballage, le JOB s'arrêtera et prend l'état d'attente, en attendant l'étiquette de parvenir à son support. Quand l'étiquette arrive (new TAG), le JOB se repris pour être dans la zone d'emballage. Quand le produit et l'étiquette sont tout les deux dans la zone d'emballage, la feuille d'emballage prend la forme d'un tube, via un entonnoir et un appareil de soudure colle les deux bords de la feuille. Un fer à souder latéral soude le tube entre les différents paquets et le produit emballé sort de la zone d'emballage. Les produits scellés sont ensuite séparés par le coupoir pour produire des produits individuellement emballés [Bordbar, 2000]. Et tout le cycle recommence.

A partir de ce texte qui présente le scénario de fonctionnement de cette ligne d'emballage, nous pouvons distinguer les acteurs de ce diagramme qui sont : la ceinture (belt), le dévidoir de film d'emballage (Film), l'appareil de soudure (welder) et le coupoir (cutter).

La synchronisation entre ces acteurs s'effectue deux à deux, entre la ceinture et le dévidoir de film d'emballage, entre le dévidoir de film d'emballage et l'appareil de soudure et entre le dévidoir de film d'emballage et le coupoir.

3.1.2.1 Diagramme de classes

Dés que le diagramme de cas d'utilisation pour la ligne d'emballage a quatre acteurs, alors nous avons quatre classes dans le diagramme de classes comme le montre la Figure 3.2.

Les relations entre ces classes sont des associations entre chaque deux classes, présentant

la synchronisation entre chaque deux acteurs du diagramme de cas d'utilisation.

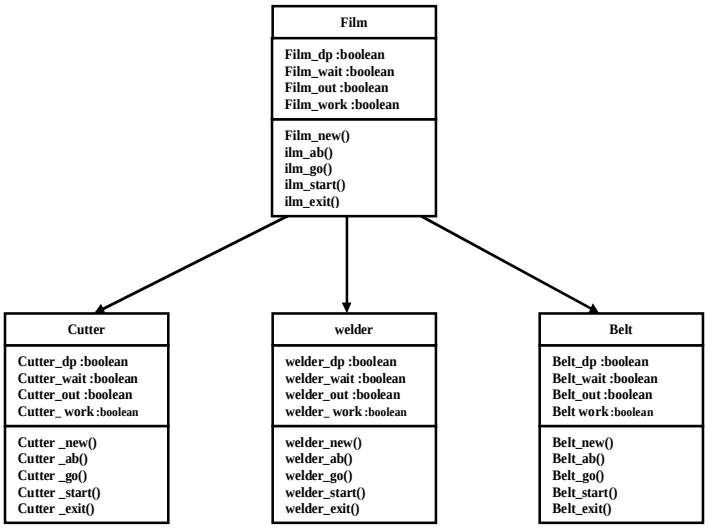

Figure 3.2 Diagramme de classes pour la chaîne d'emballage.

3.1.2.3 Modélisation automatique de l'exemple

Suivant le méta-modèle présenté dans la Figure 2.11 du chapitre précédent et avec l'outil de modélisation généré par ATOM^3, le diagramme de classes de la Figure 3.2 est modélisé dans cet outil comme suit :

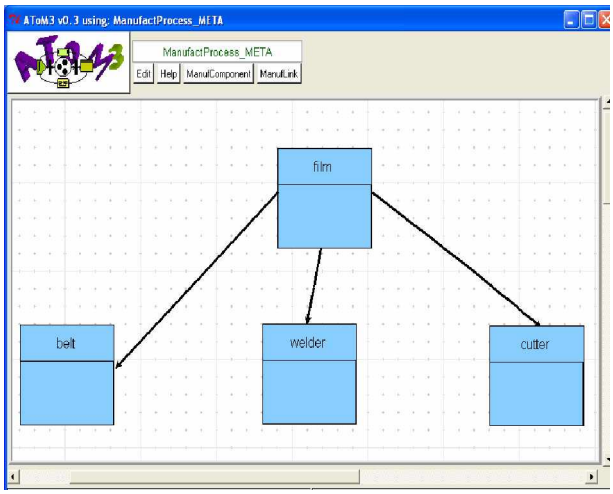

Figure 3.3 Diagramme de classes pour la chaîne d'emballage dans ATOM^3.

Pour chaque classe, nous pouvons saisir les valeurs de ses attributs, chaque attribut coché indique qu'elle est égale à vrai, par conséquent dans le réseau de Petri équivalant, la place correspondante à cet attribut est marquée par un jeton.

Les méthodes des classes sont sans code puisque ce qui nous intéresse c'est la synchronisation entre leurs exécutions et pas comment elles sont implémentées.

Pour cet exemple, nous allons étudier deux variantes. Dans la première, nous laissons l'exemple tel qu il est, et dans le deuxième nous modifions l'état initial de la ligne d'emballage.

a. Variante n°1

Dans cette variante nous cochons l'attribut **out** pour toutes les classes pour indiquer que dans l'état initial, toutes les machines sont hors la zone d'emballage.

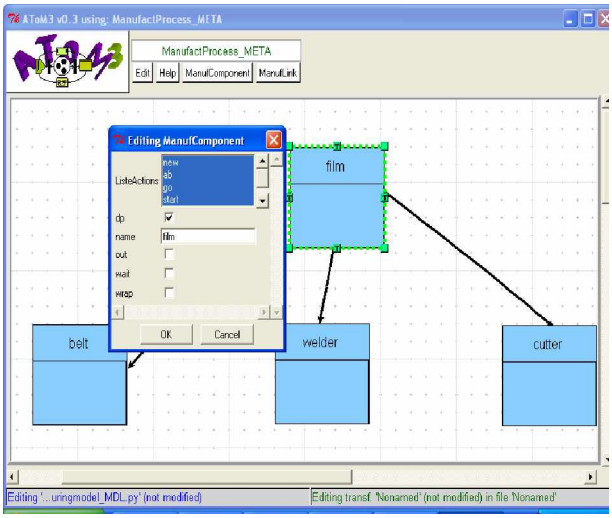

Figure 3.4 Saisie des valeurs des attributs des classes pour la variante 1.

b. Variante n°2

Dans cette variante nous cochons un attribut différent pour chaque classe, par exemple l'attribut **dp** pour la classe Film et **out** pour la classe Belt, pour indiquer que dans l'état initial, chaque machine est dans un état différent de l'autre.

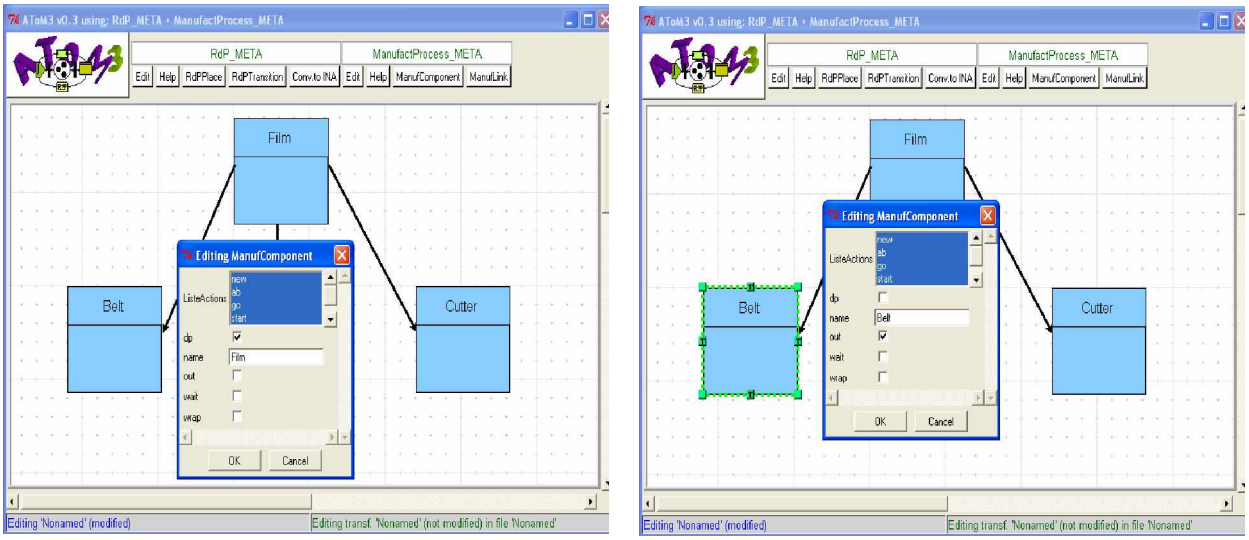

Figure 3.5 Saisie des valeurs des attributs des classes pour la variante 2.

3.1.3 Transformation du diagramme de classes vers les réseaux de Petri

Pour obtenir Un modèle réseau de Petri équivalent au diagramme de classes de la Figure 3.3, nous avons appliqué notre grammaire qui est composée de dix règles .L'application de la grammaire s'effectue en trois étapes :

§ Chargement des deux méta-modèles pour les réseaux de Petri et le diagramme de classes.

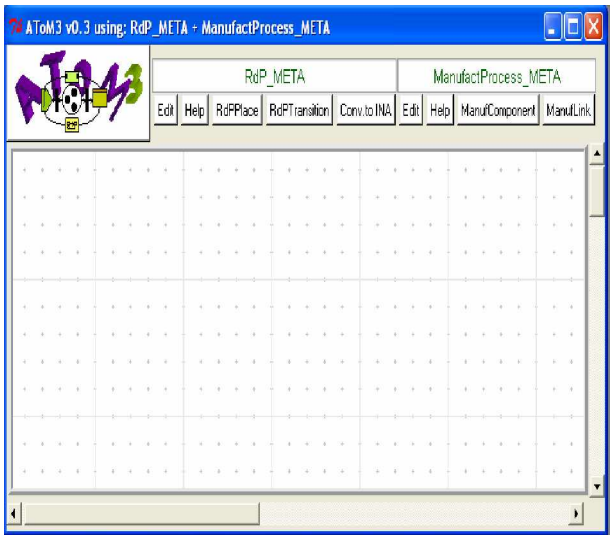

Figure 3.6 Chargement des méta-modèles.

§ Ouverture du modèle de diagramme de classes de la figure 3.3 qui a été sauvegardé.

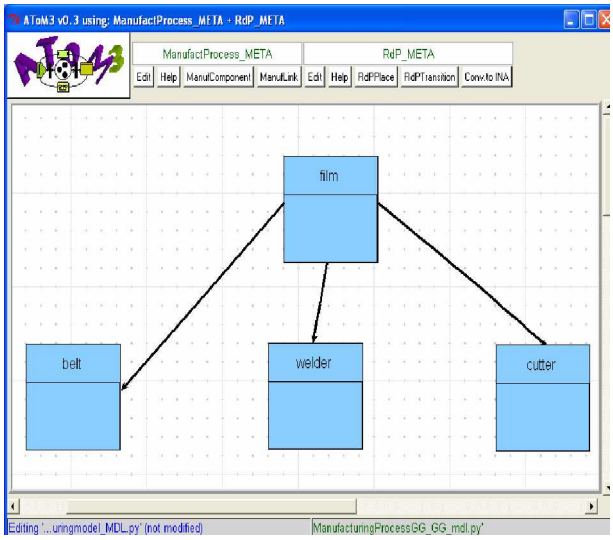

Figure 3.7 Ouverture du diagramme de classes.

§ Chargement de la grammaire de transformation.

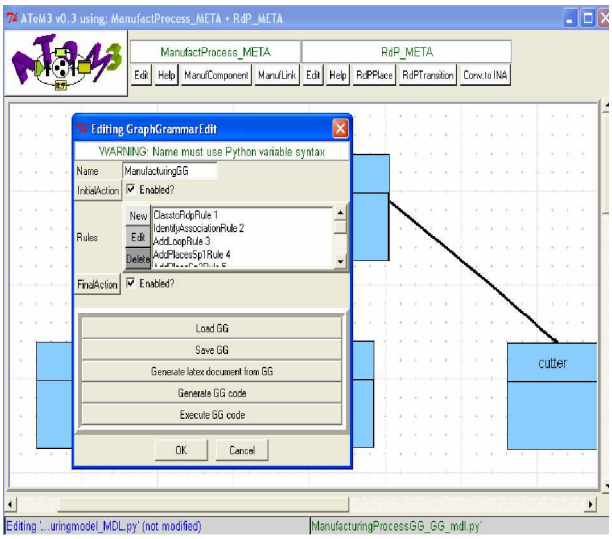

Figure 3.8 Chargement de la grammaire de transformation.

§ Exécution de la grammaire en lançant son fichier exécutable.

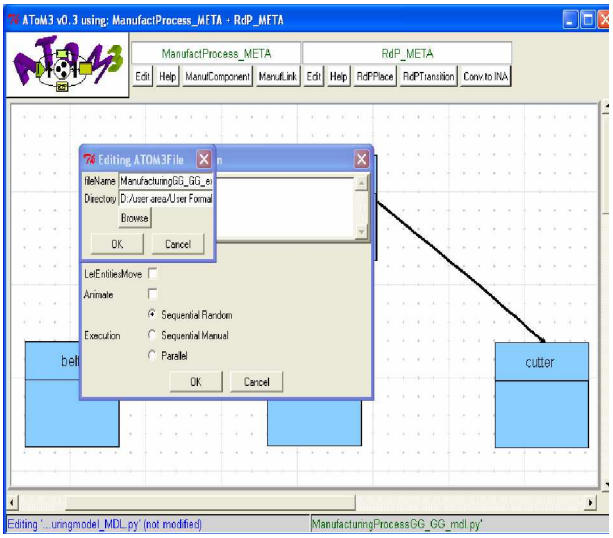

Figure 3.9 Ouverture du fichier exécutable de la grammaire.

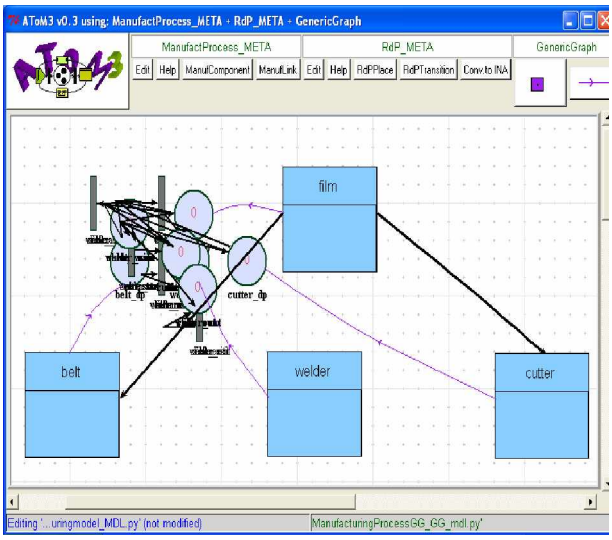

Figure 3.10 Lancement de la grammaire.

Pour une raison de la taille du modèle réseau de Petri généré, nous présentons ici le réseau de Petri qui concerne la synchronisation entre le Belt et le Film pour l'exemple du processus d'emballage. Les résultats de la transformation sont illustrés dans les Figures suivantes :

a. Variante n°1

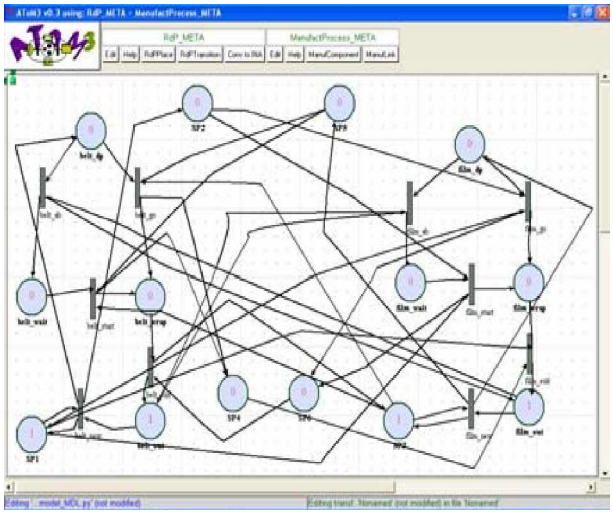

Figure 3.11 Réseau de Petri généré pour la variante n°2.

b. Variante n°2

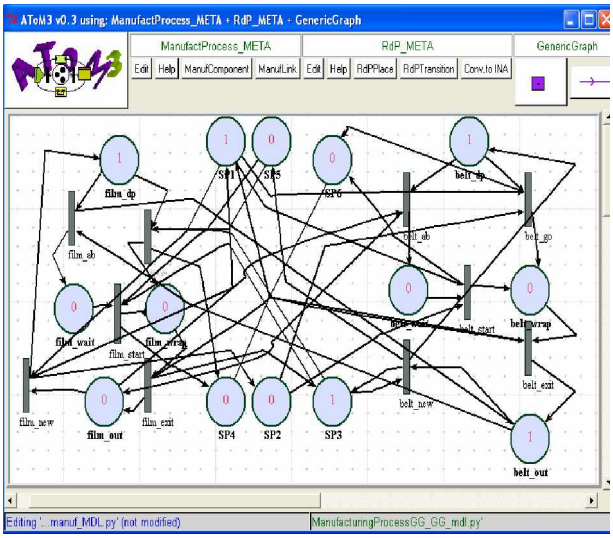

Figure 3.12 Réseau de Petri généré pour la variante n°2.

3.1.4 Vérification des propriétés du réseau de Petri généré

Comme nous avons mentionné dans le chapitre précèdent, que la vérification des propriétés comportementales du processus de production est réalisée en utilisant l'outil de vérification INA, adapté pour les réseaux de Petri.

Nous avons intégré l'outil INA dans ATOM^3 de telle sorte que nous pouvons obtenir un fichier texte qui est une description du modèle réseau de Petri généré. Les étapes permettant la

réalisation de cette procédure sont :

§ Génération du fichier texte d'extension **.pnt** décrivant le réseau de Petri généré a partir de ATOM[3], en cliquant sur le bouton **Conv to INA**, comme le montre la Figure suivante :

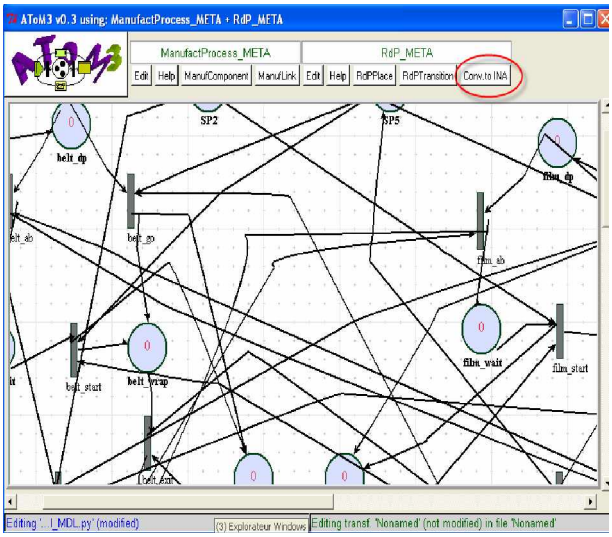

Figure 3.13 Obtention du fichier d'entrée pour l'outil INA.

Après, le fichier texte correspondant à ce modèle est généré, sous la forme présentée dans le chapitre précédent. Les Figures suivantes présentent le contenu de ce fichier :

a. Variante n°1

```
P  M  PRE,POST  NETZ 1:
0  0  0,  12
1  0  2,  3
2  0  13, 4
3  1  45, 05
4  0  6,  75
5  0  5,  8
6  0  78, 9
7  1  92, 62
8  1  90, 087
9  1  46, 631
10 0  87, 4
11 0  13, 9
12 0  0,  78
13 0  6,  13
@
place nr. Name     capacity  time
     0:  belt_dp      00        0
     1:  belt_wait    00        0
     2:  belt_wrap    00        0
     3:  belt_out     00        0
     4:  film_dp      00        0
     5:  film_wait    00        0
     6:  film_wrap    00        0
     7:  film_out     00        0
     8:  SP1          00        0
     9:  SP3          00        0
    10:  SP6          00        0
    11:  SP4          00        0
    12:  SP2          00        0
    13:  SP5          00        0
@
trans nr.  name          priority  time
     0:  belt_new          0        0
     1:  belt_go           0        0
     2:  belt_ab           0        0
     3:  belt_start        0        0
     4:  belt_exit         0        0
     5:  film_ab           0        0
     6:  film_new          0        0
     7:  film_go           0        0
     8:  film_start        0        0
     9:  film_exit         0        0
@
```

Figure 3.14 Fichier d'entrée pour l'outil INA pour la variante n°1.

b. Variante n°2

```
 P  M  PRE ,POST  NETZ 1:
 0  1   0,   12
 1  0   2,   3
 2  0   13,  4
 3  0   45,  05
 4  1   6,   75
 5  0   5,   8
 6  0   78,  9
 7  1   92,  62
 8  1   90,  087
 9  1   46,  631
10  0   87,  4
11  0   13,  9
12  0   0,   78
13  0   6,   13
@
place nr.    Name        capacity time
     0:      film_dp       00     0
     1:      film_wait     00     0
     2:      film_wrap     00     0
     3:      film_out      00     0
     4:      belt_dp       00     0
     5:      belt_wait     00     0
     6:      belt_wrap     00     0
     7:      belt_out      00     0
     8:      SP1           00     0
     9:      SP3           00     0
    10:      SP6           00     0
    11:      SP4           00     0
    12:      SP2           00     0
    13:      SP5           00     0
@
trans nr.   name        priority  time
     0:     film_new       0     0
     1:     film_go        0     0
     2:     film_ab        0     0
     3:     film_start     0     0
     4:     film_exit      0     0
     5:     belt_ab        0     0
     6:     belt_new       0     0
     7:     belt_go        0     0
     8:     belt_start     0     0
     9:     belt_exit      0     0
@
```

Figure 3.15 Fichier d'entrée pour l'outil INA pour la variante n°2.

§ Lancement de l'outil INA et l'opération de vérification, en tapant le caractère **A** et en spécifiant le nom du fichier d'entrée :

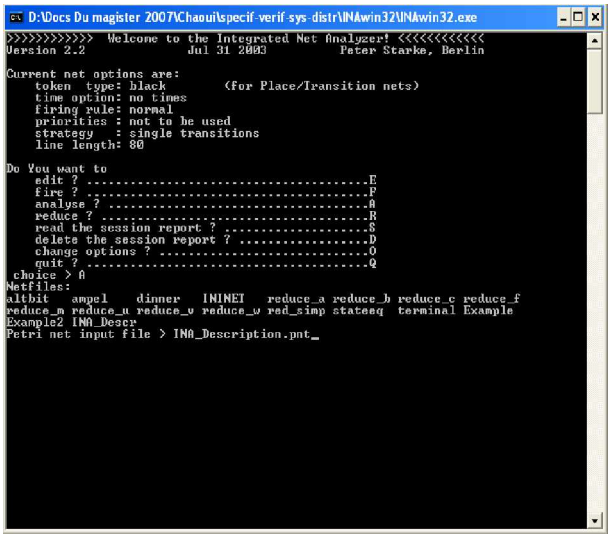

Figure 3.16 Lancement de l'outil INA.

Un menu d'analyse est affiché ensuite, nous devons spécifier le type d'analyse que nous voulons faire et comme, nous nous sommes intéressés aux propriétés comportementales du modèle, nous avons choisis les caractères **B**,**R** ensuite **L**.

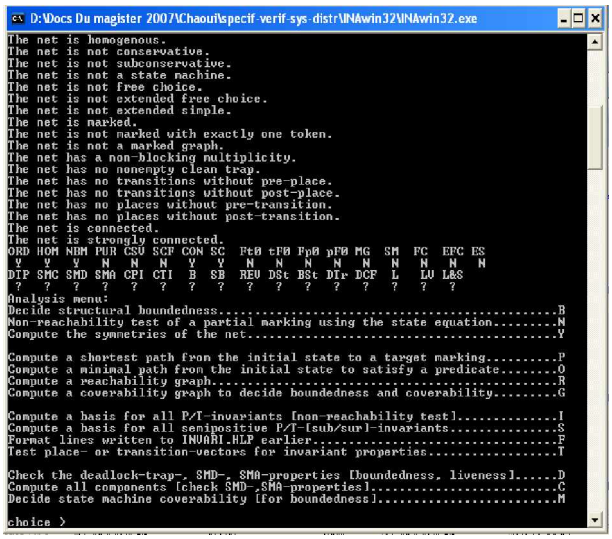

Figure 3.17 Choix du type d'analyse.

a. Variante n°1

Les résultats obtenus pour la variante 1 sont illustrés dans les Figures suivantes

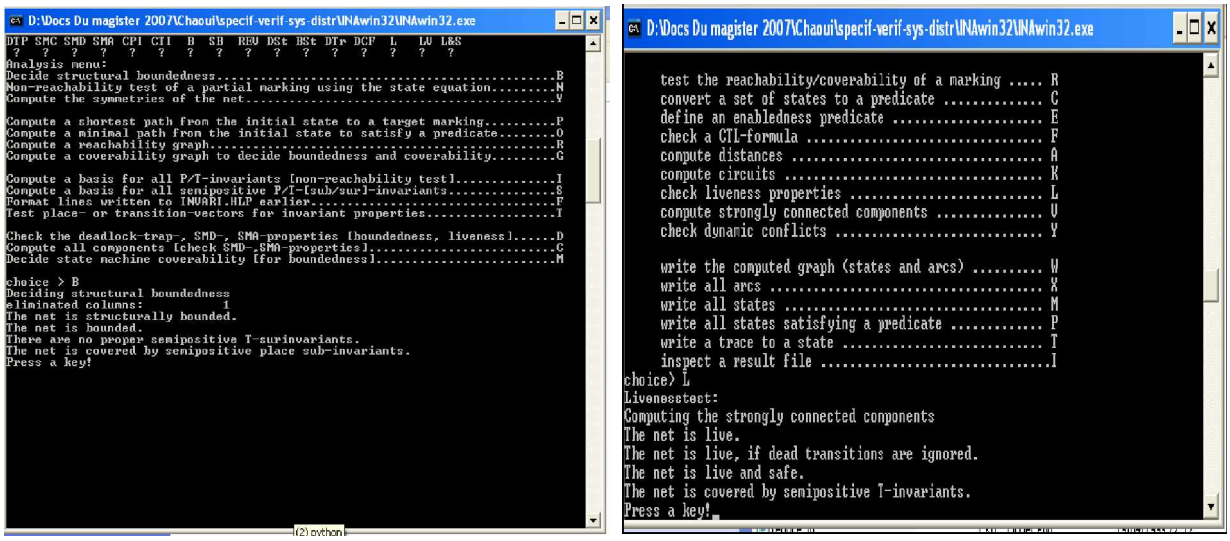

Figure 3.18 Analyse comportementale du modèle pour la variante n°1.

D'après le résultat d'analyse nous observons que le modèle est vivant et borné et n'a aucune transition bloquante.

b. Variante n°2

Les résultats obtenus pour la variante 2 sont les suivants :

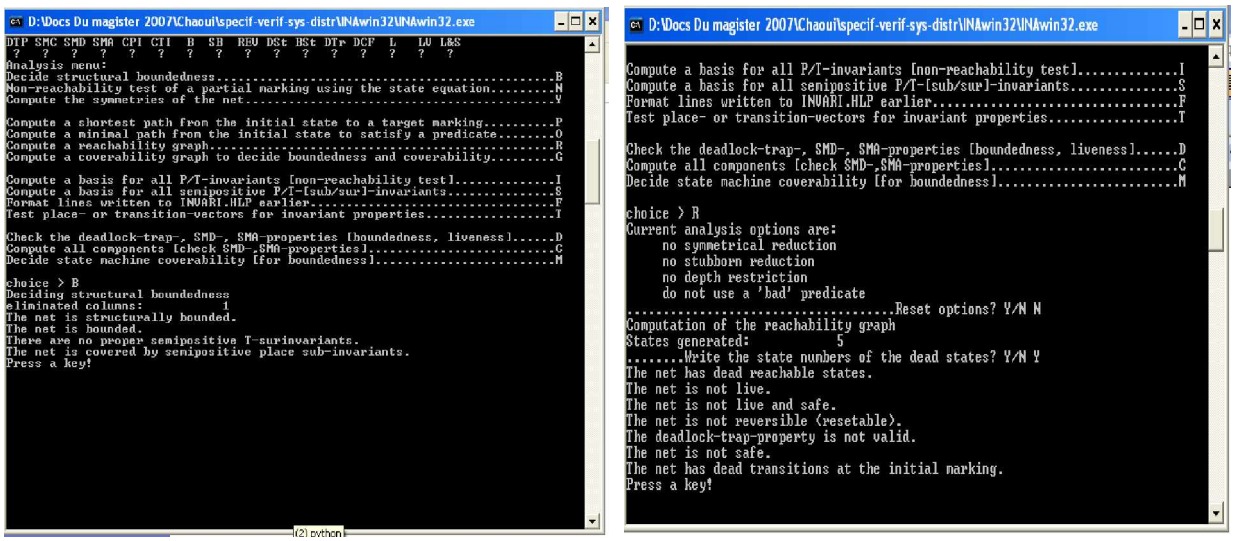

Figure 3.19 Analyse comportementale du modèle pour la variante n°2.

D'après le résultat d'analyse nous observons que le modèle est borné mais non vivant et a des transitions bloquantes.

§ Génération du graphe des états désirables. Comme nous avons mentionné dans le chapitre précédent que ce graphe est équivalent au graphe de marquages, ce graphe permet de produire un enchaînement des activités du processus de production en évitant les cas du fonctionnement anormal du processus.

L'outil INA nous a permis de générer ce graphe a partir du fichier texte d'entrés. Le résultat est un autre fichier texte d'extension ‘‘**.gra’’.** La procédure est réalisée en tapant dans INA le caractère **R** ensuite **W.** Après, le fichier texte correspondant à ce graphe est généré.

a. Variante n°1

Le GDS pour la variante 1 est calculé par l'outil INA car d'après l'analyse nous avons trouvé que le modèle réseau de Petri de cette variante est borné et vivant. Les résultats sont illustrés dans les Figures 3.20 et 3.21

```
State nr.   1
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  0  1  0  0  0  1  1  1  0  0  0  0
==t0=> s2
==t6=> s12
State nr.   2
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  0  0  0  1  1  1  0  0  1  0
==t2=> s3
==t6=> s10
State nr.   3
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  1  0  0  0  0  0  1  1  1  0  0  1  0
==t6=> s4
State nr.   4
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  1  0  0  1  0  0  0  1  1  0  0  1  1
==t3=> s5
==t7=> s9
State nr.   5
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  1  0  1  0  0  0  1  0  0  1  1  0
==t7=> s6
State nr.   6
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  1  0  0  0  1  0  0  0  1  1  0  0
==t4=> s7
==t9=> s8
State nr.   7
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  0  1  0  0  1  0  0  1  0  1  0  0
==t9=> s1
State nr.   8
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  1  0  0  0  0  1  1  0  1  0  0  0
==t4=> s1
State nr.   9
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  1  0  0  0  0  1  0  0  1  1  0  0  1
==t3=> s6
State nr.   10
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  1  0  0  0  1  1  0  0  1  1
==t1=> s5
==t7=> s11
State nr.   11
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  0  0  1  0  0  1  1  0  0  1
==t1=> s6
State nr.   12
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  0  1  1  0  0  0  1  1  0  0  0  1
==t0=> s10
==t5=> s13
State nr.   13
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  0  1  0  1  0  0  1  1  0  0  0  1
==t0=> s14
State nr.   14
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  0  1  0  0  1  1  0  0  1  1
==t1=> s15
==t8=> s11
State nr.   15
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  1  0  0  1  0  0  1  0  0  1  1  0
==t8=> s6
```

Figure 3.20 Fichier du graphe de marquage généré par l'outil INA pour la variante n°1.

Pour mieux comprendre le GDS généré, nous avons illustré le contenu du fichier par un graphe de marquages, comme le montre la Figure suivante :

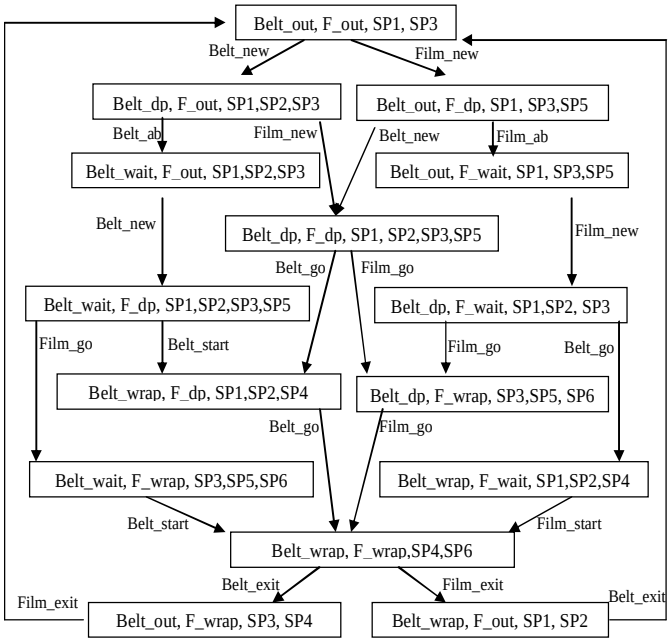

Figure 3.21 GDS généré par l'outil INA pour la variante n°1.

b. Variante n°2

Le GDS pour la variante 2 est non calculé par l'outil INA car le réseau de Petri généré pour cette variante est borné mais n'est pas vivant. Le résultat obtenu est illustré dans la Figure suivante

```
State nr.    1
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  1  0  0  1  1  1  0  0  0  0
==t2=> s2
==t6=> s5
State nr.    2
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  1  0  0  1  0  0  1  1  1  0  0  0  0
==t6=> s3
State nr.    3
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  1  0  0  2  0  0  0  1  1  0  0  0  1
==t3=> s4
State nr.    4
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  0  0  1  0  2  0  0  0  1  0  0  1  0  0
dead state
State nr.    5
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13
toks:  1  0  0  0  2  0  0  0  1  1  0  0  0  1
==t1=> s4
```

Figure 3.22 Fichier du graphe de marquage généré par l'outil INA pour la variante n°2.

3.2 Exemple2 : Chaîne d'embouteillage

3.2.1 Présentation

Notre deuxième exemple concerne une chaîne d'embouteillage d'eau. Une chaîne d'embouteillage est constituée de machines en série (Figure 3.23). Ces machines sont : un dépalettiseur, un convoyeur, une rinceuse, et un remplisseur.

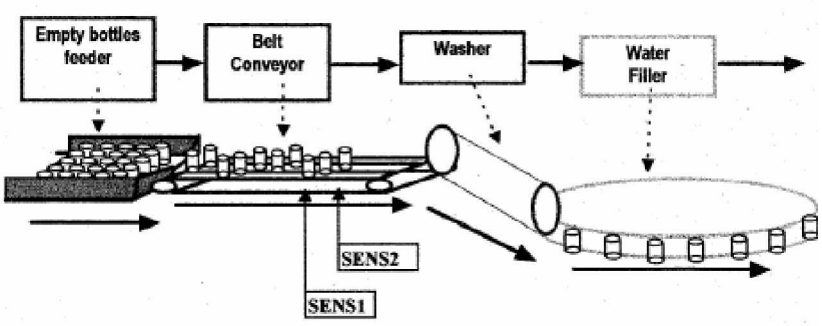

Figure 3.23 Exemple d'une ligne d'embouteillage [Adry, 1995].

- **Le dépalettiseur (Feeder) : s**a fonction est de déverser des couches de 360 bouteilles vides sur le convoyeur à accumulation situé en aval. Il a un fonctionnement asynchrone (il dépose une couche de 360 bouteilles uniquement s'il y a la place suffisante sur le convoyeur en aval).
- **Le convoyeur (conveyor):** son rôle est de transporter les bouteilles vides.
- **La rinceuse (washer) :** Pour assurer une parfaite hygiène avant le remplissage, les bouteilles doivent être nettoyées. C'est la fonction de la rinceuse, constituée d'un tunnel incliné et fermé à la poussière extérieure. Sa capacité est de 80 boîtes.
- **Le remplisseur (Filler):** La fonction du remplisseur est à la fois le remplissage d'eau et le sertissage (pose des couvercles). Sa cadence maximale est de 48000 boîtes par heure [Adry, 1995].

3.2.2 Modélisation avec UML

3.2.2.1 Diagramme de cas d'utilisation

Quand le dépalettiseur commence à fournir les bouteilles vides, si le convoyeur a des places vides, les bouteilles sont évacuées au convoyeur et le transport des bouteilles peut avoir lieu aux vitesses de 1 ou 2 mètres/seconde. La quantité de bouteilles sur le convoyeur reste entre les limites détectées par des capteurs d'accumulation. Cependant, si le convoyeur est inactif, le dépalettiseur s'arrêtera, attendant le convoyeur soit actif. Quand le convoyeur reprend son fonctionnement, le dépalettiseur est remis en marche. Quand les deux machines sont dans l'état

de fonctionnement, les bouteilles peuvent être accumulées vers la sortie du convoyeur. Si la sortie est fermée, la rinceuse commence à laver. Si le remplisseur est prêt alors il met une quantité constante de l'eau dans chaque bouteille. Pour faire cela, les bouteilles doivent être toujours remplies à la même vitesse. Ainsi, quand le remplisseur est de s'arrêter, les bouteilles qui sont intérieur doivent être évacuées et ne peuvent pas s'accumuler. Ceci est fait par la fermeture de l'entrée du remplisseur. Les bouteilles remplies sortent de la zone d'embouteillage. Ensuite elles seront stockées. Et tout le cycle d'embouteillage recommence.

Depuis le texte, les acteurs de ce diagramme sont : le dépalettiseur (Feeder), le convoyeur (conveyor), la rinceur (washer) et le remplisseur (filler). La synchronisation entre ces acteurs s'effectue deux à deux.

3.2.2.2 Diagramme de classes

Le diagramme de classes pour cet exemple est composé de quatre classes comme le montre la Figure 3.24. Les relations entre ces classes sont des associations entre chaque deux classes, présentant la synchronisation entre chaque deux acteurs du diagramme de cas d'utilisation.

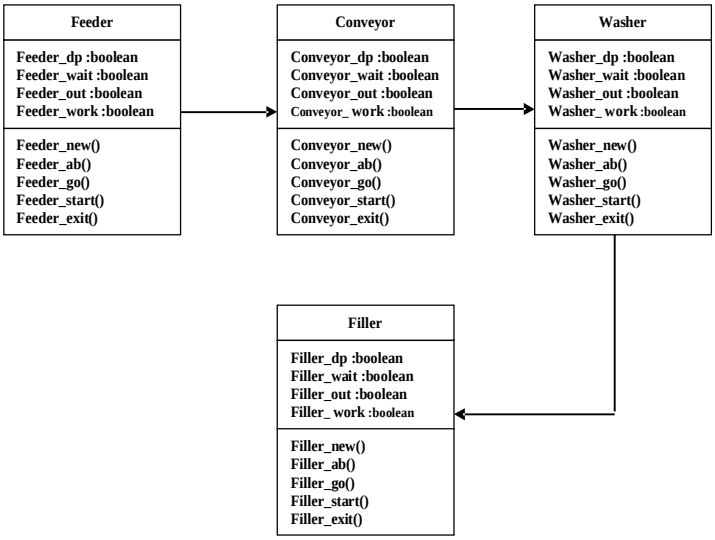

Figure 3.24 Diagramme de classes pour la chaîne d'embouteillage.

3.2.2.3 Modélisation automatique de l'exemple

a. Variante n° 1

Dans cette variante de l'exemple, le diagramme de classes est modélisé de telle sorte que le washer prend en charge aussi le rôle du Filler. Donc il y aura une synchronisation entre le Feeder et le conveyor et entre ce dernier et le washer.

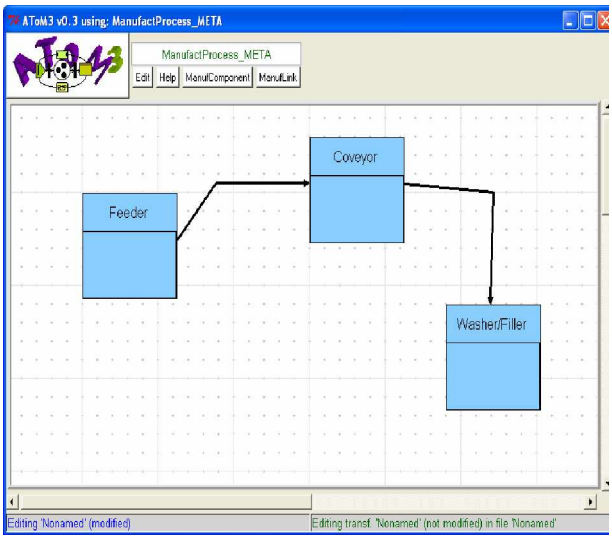

Figure 3.25 Diagramme de classes pour la variante n°1.

b. Variante n° 2

Dans La variante 2, le diagramme de classes modélise la chaîne d'embouteillage telle qu'elle est en mettant quatre classes comme le montre la Figure 3.26

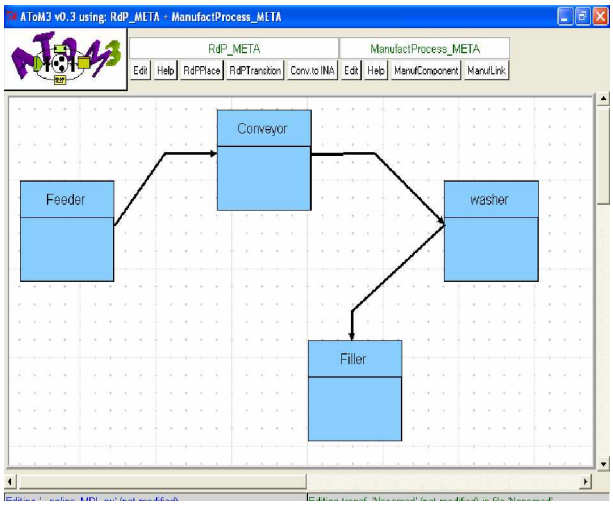

Figure 3.26 Diagramme de classes pour la variante n°2.

3.2.3 Transformation du diagramme de classes vers les réseaux de Petri

a. Variante n° 1

Le réseau de Petri équivalent au diagramme de classes de la Figure 3.25, après l'application de notre grammaire de transformation est illustré dans la Figure 3.27. Ainsi, nous présentons le réseau de Petri qui concerne la synchronisation entre le conveyor et le Feeder et entre le conveyor et le washer.

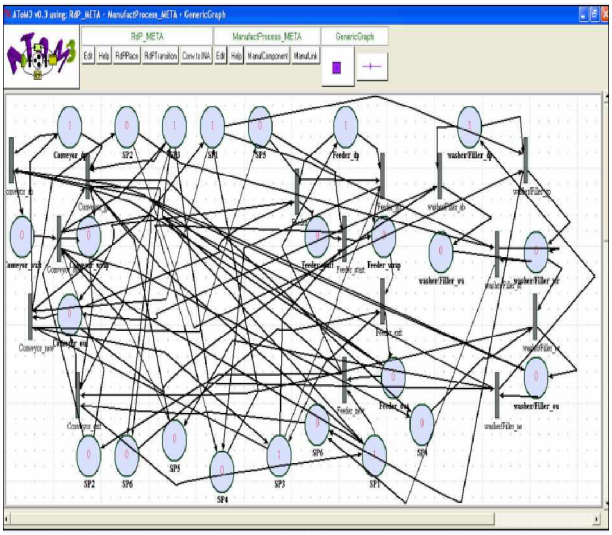

Figure 3.27 Réseau de Petri généré pour la variante n°1.

b. Variante n° 2

Pour la variante 2 le réseau de Petri équivalent au diagramme de classe de la Figure 3.26 est présenté dans la Figure suivante, et toujours à cause de la taille de ce réseau de Petri généré, nous présentons une partie qui concerne la synchronisation entre le conveyor et le feeder :

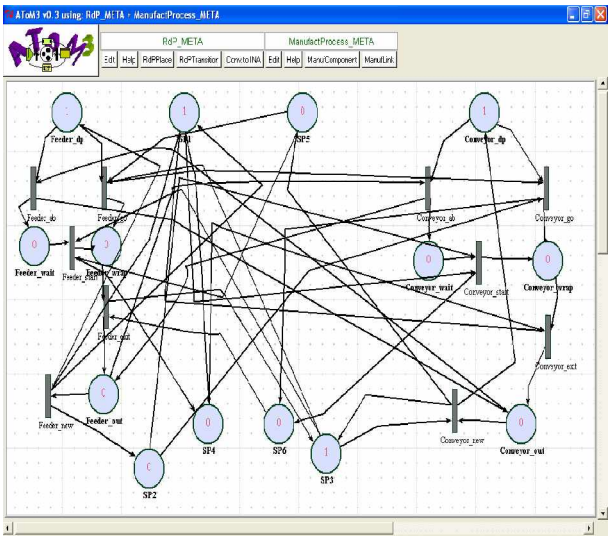

Figure 3.28 Réseau de Petri généré pour la variante n°2.

3.2.4 Vérification des propriétés du réseau de Petri généré

a. Variante n°1

Le fichier d'entrée pour l'outil INA est généré après l'application de la grammaire, comme le montre la Figure suivante :

```
 P  M  PRE ,     POST NETZ 1:
 0  1  0 ,       1 2
 1  0  2 ,       3
 2  0  1 3 ,     4
 3  0  4 5 ,     0 5
 4  1  6 ,       7 5
 5  0  5 ,       8
 6  0  7 8 ,     9
 7  0  9 2 10 ,  6 2 10
 8  1  11 ,      12 10
 9  0  10 ,      13
10  0  12 13 ,   14
11  0  14 5 ,    11 5
12  1  9 0 ,     0 8 7
13  1  4 6 ,     6 3 1
14  0  8 7 ,     4
15  0  1 3 ,     9
16  0  0 ,       7 8
17  0  6 ,       1 3
18  1  14 6 ,    6 13 12
19  1  9 11 ,    11 8 7
20  0  13 12 ,   9
21  0  7 8 ,     14
22  0  6 ,       12 13
23  0  11 ,      7 8
@
place nr.      name          capacity  time
     0:        Feeder_dp        00    0
     1:        Feeder_wait      00    0
     2:        Feeder_wrap      00    0
     3:        Feeder_out       00    0
     4:        Conveyor_dp      00    0
     5:        Conveyor_wait    00    0
     6:        Conveyor_wrap    00    0
     7:        Conveyor_out     00    0
     8:        washer/Filler_dp 00    0
     9:        washer/Filler_wa 00    0
    10:        washer/Filler_wr 00    0
    11:        washer/Filler_ou 00    0
    12:        SP1              00    0
    13:        SP3              00    0
    14:        SP6              00    0
    15:        SP4              00    0
    16:        SP2              00    0
    17:        SP5              00    0
    18:        SP1              00    0
    19:        SP3              00    0
    20:        SP6              00    0
    21:        SP4              00    0
    22:        SP2              00    0
    23:        SP5              00    0
 @
trans nr.      name          priority time
     0:        Feeder_new       0     0
     1:        Feeder_go        0     0
     2:        Feeder_ab        0     0
     3:        Feeder_start     0     0
     4:        Feeder_exit      0     0
     5:        Conveyor_ab      0     0
     6:        Conveyor_new     0     0
     7:        Conveyor_go      0     0
     8:        Conveyor_start   0     0
     9:        Conveyor_exit    0     0
    10:        washer/Filler_ab 0     0
    11:        washer/Filler_ne 0     0
    12:        washer/Filler_go 0     0
    13:        washer/Filler_st 0     0
    14:        washer/Filler_ex 0     0
@
```

Figure 3.29 Fichier d'entrée de l'outil INA généré pour la variante n°1.

L'analyse avec l'outil a donné les résultats suivants :

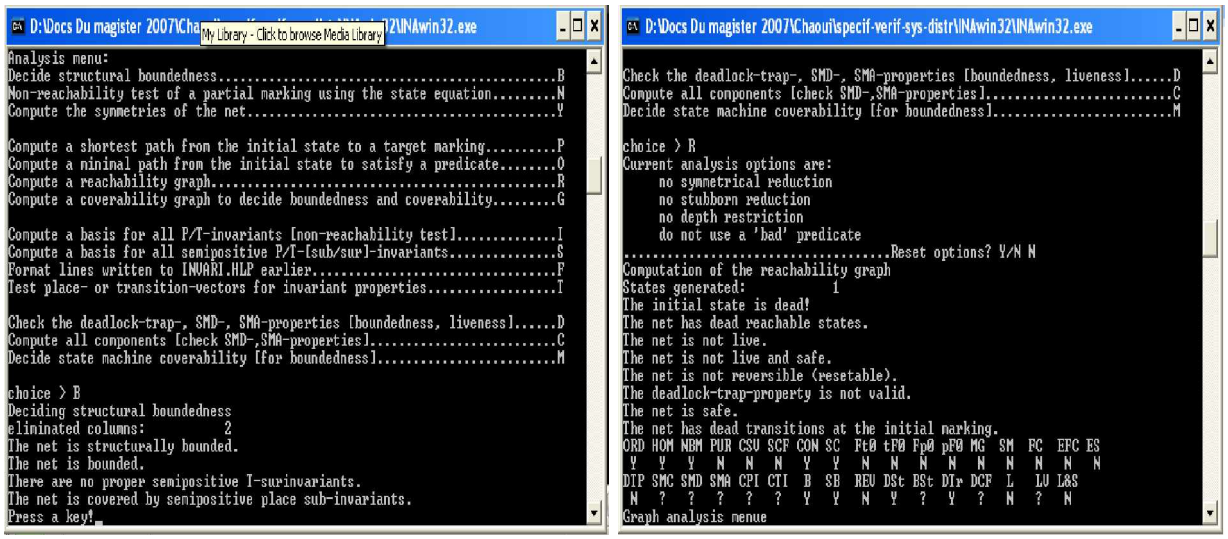

Figure 3.30 Analyse comportementale du modèle pour la variante n°1.

D'après les deux écrans d'INA, nous observons que le modèle est non vivant et borné et Il a des transitions mortes dés le marquage initial et le graphe des états désirables ne peut pas être calculé. Le fichier d'extension ''**.gra**'' est illustré dans la Figure suivante :

```
State nr.   1
P.nr:  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
   : 23
toks:  1  0  0  0  1  0  0  0  1  0  0  0  1  1  0  0  0  0  1  1  0  0  0
   : 0
dead state
```

Figure 3.31 Fichier du graphe de marquage généré par l'outil INA pour la variante n°1.

b. Variante n°2

Le fichier d'entrée pour l'outil INA est généré, comme le montre la Figure suivante

```
P   M    PRE,POST   NETZ 1:
  0 1      0 , 1 2
  1 0      2 , 3
  2 0      1 3 , 4
  3 0      4 5 , 0 5
  4 1      6 , 7 5
  5 0      5 , 8
  6 0      7 8 , 9
  7 0      9 2 , 6 2
  8 1      9 0 , 0 8 7
  9 1      4 6 , 6 3 1
 10 0      8 7 , 4
 11 0      1 3 , 9
 12 0      0 , 7 8
 13 0      6 , 1 3
@
place nr.                name capacity time
        0: Feeder_dp                00     0
        1: Feeder_wait              00     0
        2: Feeder_wrap              00     0
        3: Feeder_out               00     0
        4: Conveyor_dp              00     0
        5: Conveyor_wait            00     0
        6: Conveyor_wrap            00     0
        7: Conveyor_out             00     0
        8: SP1                      00     0
        9: SP3                      00     0
       10: SP6                      00     0
       11: SP4                      00     0
       12: SP2                      00     0
       13: SP5                      00     0
@
trans nr.                name priority time
        0: Feeder_new                0     0
        1: Feeder_go                 0     0
        2: Feeder_ab                 0     0
        3: Feeder_start              0     0
        4: Feeder_exit               0     0
        5: Conveyor_ab               0     0
        6: Conveyor_new              0     0
        7: Conveyor_go               0     0
        8: Conveyor_start            0     0
        9: Conveyor_exit             0     0
@
```

Figure 3.32 Fichier d'entrée de l'outil INA généré pour la variante n°2.

L'analyse avec l'outil a donné les résultats suivants :

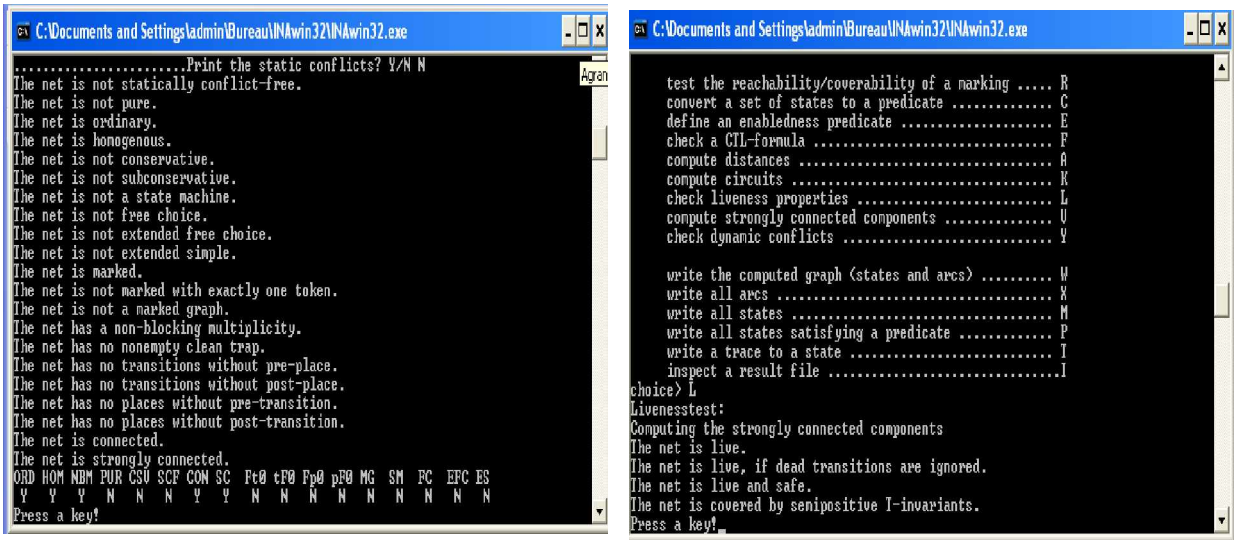

Figure 3.33 Analyse comportementale du modèle pour la variante n°2.

D'après les résultats obtenus, nous observons que le modèle est vivant et borné, donc le graphe des états désirables peut être calculé.

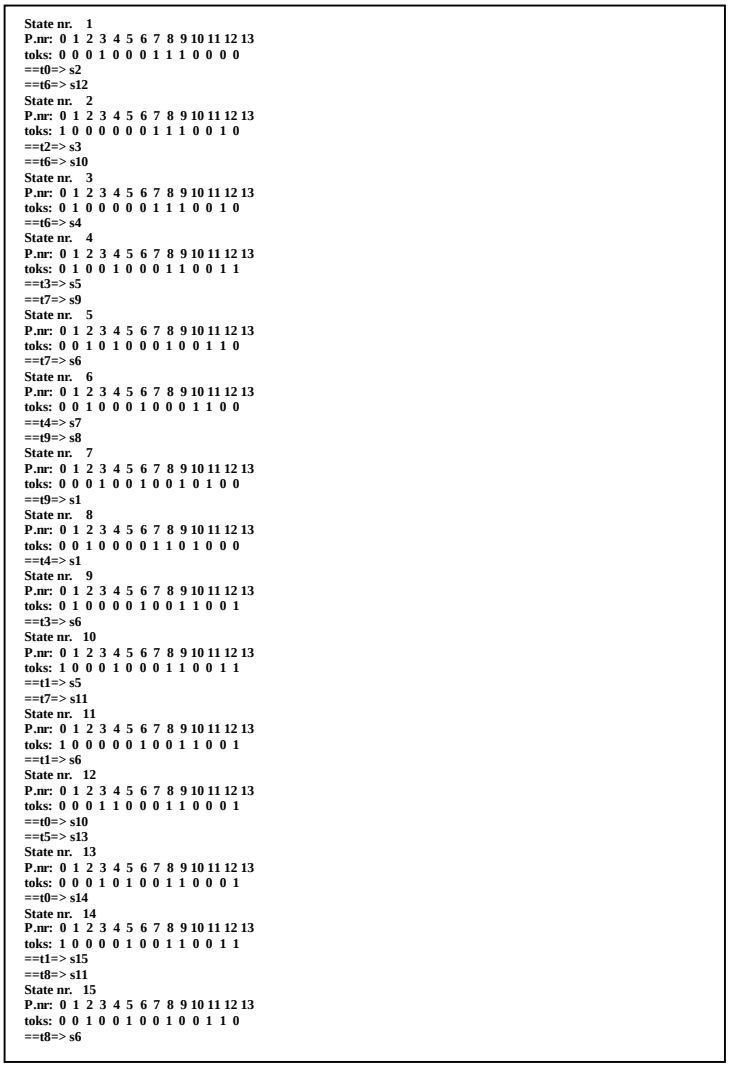

Figure 3.34 Fichier du graphe de marquage généré par l'outil INA pour la variante n°2.

Le format graphique correspondante à ce fichier texte est la suivante :

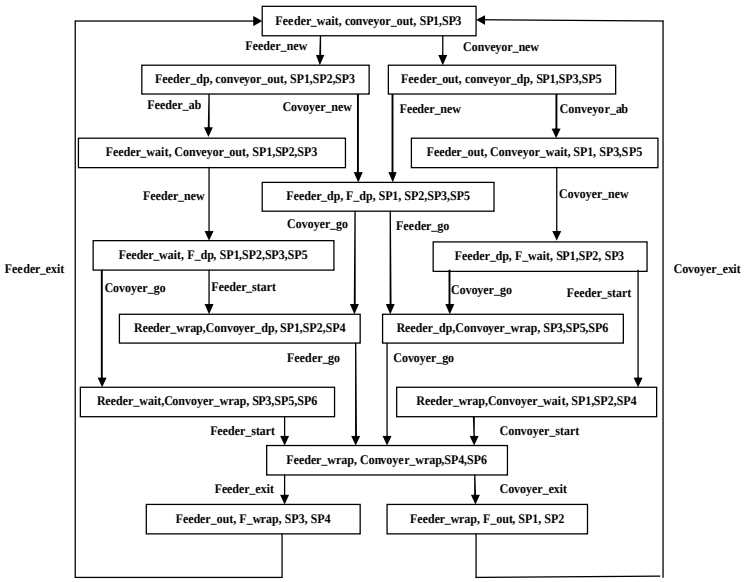

Figure 3.35 GDS généré par l'outil INA.

3.3 Conclusion

A partir d'une chaîne d'embouteillage et une autre d'emballage de produits, deux exemples ont été proposés pour la raison qu'ils sont traités de manière manuelle dans un travail précédent [Bordbar, 2000], donnant la possibilité de valider nos résultats. Ces deux exemples nous ont permis d'appliquer notre méthode de conception d'un superviseur de contrôle. Les deux exemples ont suivi les différentes étapes menant à la modélisation d'un processus de production distribué, et la vérification de ses propriétés en vue de sa supervision.

Nous avons étudié deux variantes pour chaque exemple. Dans le premier exemple concernant la chaîne d'emballage nous avons modélisé le processus tel qu il est dans la première variante, ensuite nous avons modifié l'état initial des machines composant la chaîne et nous avons trouvé que le processus risque d'être dans un état de blocage. .

Dans le deuxième exemple, Nous avons essayé de modifier la synchronisation entre les machines composant la chaîne d'embouteillage, après la vérification des propriétés, nous avons obtenu que le modèle est non vivant et le graphe des états désirables ne peut pas être calculé, indiquant qu' il y a un problème de blocage dans le processus d'embouteillage.