ETAT DE L'ART : Modélisation et vérification en vue de la supervision des systèmes de production

Les systèmes de production, à l'initiative de l'homme, sont caractérisés par une forte complexité et flexibilité ; ils sont considérés comme des systèmes à événements discrets. L'un des objectifs des systèmes de production est d'avoir une productivité optimale. Optimiser la productivité signifie, avoir une sûreté de fonctionnement. Le but est donc d'élaborer des procédures de supervision au dessus de la couche de commande du système de production, en vue de garantir ses objectifs. Ces procédures de supervision sont basées sur la structure du système. Donc La modélisation du système de production est une étape initiative pour la supervision ainsi que la vérification de ses propriétés.

Ce chapitre présente un état de l'art sur la modélisation et la vérification des systèmes de production en vue de leur supervision. Il se décompose en quatre parties. Dans la première partie, le concept de systèmes dynamiques à événements discrets (DES), sera présenté. Une description des systèmes de production fera l'objet de la deuxième partie. La commande par supervision, qui repose sur la théorie de Ramadge et Wonham, sera étudiée dans la troisième partie. Les potentialités de modélisation et de vérification des systèmes de production seront abordées dans la quatrième partie. Dans la conclusion, il sera tenté de dégager quelques éléments de réflexion sur cet état de l'art.

1.1 Systèmes à événements discrets

1.1.1 Définition

Les Systèmes Dynamiques à Evénements Discrets (DES) sont une classe de systèmes dynamiques, et complexes.

Par opposition aux systèmes dynamiques continus, ils satisfont généralement les propriétés suivantes :

§ L'espace d'état est décrit par un ensemble discret. c'est à dire le changement d'états est causé par l'occurrence des évènements à des instants non prédictibles. Dans ce cas, chaque

évènement peut être dépendant d'autres évènements et la fin d'une opération déclenche d'autres opérations. [Ferrier, 2004][Uzam, 1998].

§ Les transitions d'état apparaissent seulement à des instants discrets du temps ; ces transitions d'état sont associées à des événements [Ferrier, 2004].

§ Le parallélisme, c'est à dire plusieurs opérations peuvent être exécutées en même temps.

§ Les opérations sont asynchrones, à l'opposé des autres systèmes où chaque changement est synchronisé par une horloge globale, dans les DES les évènements arrivent de manière asynchrone.

§ Le non déterminisme, l'occurrence des évènements est non déterminée à priori, c'est à dire les différentes évolutions peuvent être possibles d'un état donné. [Uzam, 1998].

1.1.2 Exemple de Systèmes a événements discrets

L'exemple présenté concerne une souris qui se déplace de manière spontanée (elle agit sans intervention extérieure) à l'intérieur d'un labyrinthe.

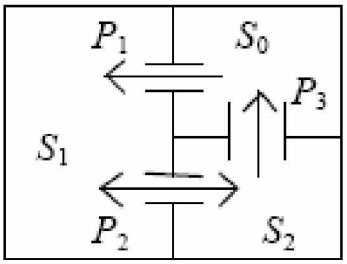

Figure 1.1 Exemple d'un Systèmes a événements discrets [Ferrier, 2004].

Les salles **Si** communiquent par des portes unidirectionnelles $\mathbf{P_1}$ et $\mathbf{P_3}$ et bidirectionnelle $\mathbf{P_2}$. On note par **"pi"** l'événement : "la souris passe par la porte **Pi**".

Soit $\sum$ l'ensemble des événements : $\sum = \mathbf{\{p_1, p_2, p_3\}}$.

On conçoit que les situations (ou comportements) possibles sont :

§ La souris est dans la salle $\boldsymbol{S_0}$.

§ La souris est dans la salle $\boldsymbol{S_1}$.

§ La souris est dans la salle $\boldsymbol{S_2}$.

Ce qui définit trois états différents, relatifs aux possibilités d'occupation des salles.

Le passage de l'état "souris en $\mathbf{S_0}$" (état **A**) à "souris en $\mathbf{S_1}$" (état **B**) a lieu sur l'occurrence de l'événement $\mathbf{p_1}$. On peut alors construire le diagramme des états (Figure 1.2). L'espace d'états s'écrit **X = {A, B, C}.**

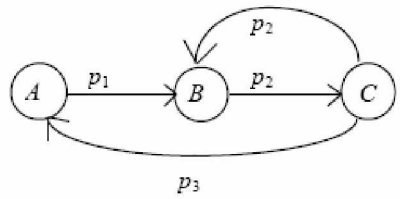

Figure 1.2 Diagramme des états pour l'exemple de la Figure 1.1

La théorie des DES peut être divisée actuellement en deux grandes approches :

- **Approche logique** qui ne s'intéresse qu'à l'occurrence des événements ou l'impossibilité de cette occurrence (*"impasse"* ou *"blocage"*, en Anglais *"deadlock"*) et à la succession de ces événements, mais pas à la date précise de ces occurrences, autrement dit pas aux aspects de performances ;cette approche est initiée par W.M. Wonham et P. Ramadge, et après eux de nombreux auteurs, ont étendu à la théorie des Automates et des Langages Formels la problématique de la commande, qui agit dans ce cas sur l'inhibition de certaines transitions d'état pour éviter les comportements non désirés .
- **Approche quantitative** qui s'adresse à l'aspect "évaluation de performance" (mesurée par le nombre d'événements survenant dans un laps de temps donné), voire à l'optimisation des performances.

1.1.3 Systèmes continus, systèmes à événements discrets

Des grandeurs telles que la position, la vitesse, l'accélération, le niveau, la pression, la température, le débit, la tension, le courant,...etc, sont des variables continues, dans le sens qu'elles peuvent prendre n'importe quelle valeur dans R lorsque le temps, lui-même est "continu", évolue.

Les grandeurs telles que le nombre de produits dans un stock, le nombre de processeurs en activité, ...etc sont discrètes. L'évolution est conditionnée par l'occurrence d'événements, à "certains instants", tels que la fin d'exécution d'une tâche, le franchissement d'un seuil devant entraîner une action, l'arrivée d'un produit, d'un client, la défaillance d'un dispositif, ...etc. [Ferrier, 2004].

Donc, on peut dire qu'un Système à Evénements Discrets est un système à espace d'état discret dont les transitions entre états sont associées à l'occurrence d'événements discrets asynchrones. Par contre, le comportement d'un système continu à l'instant **t** peut être résumé par son état qui représente la mémoire minimale du passé nécessaire à la détermination du futur.

1.1.4 Différents modèles pour les DES

Un modèle est une approximation, une vue partielle plus ou moins abstraite de la réalité afin de l'appréhender plus simplement, selon un point de vue et qu'il est établi pour un objectif donné. Il peut être exprimé par des mathématiques, des symboles, des mots, des graphes,...etc.

Il existe trois formalismes de modélisation et d'analyse des DES, les réseaux de Petri, l'algèbre des dioïdes, les langages et automates [Ferrier, 2004].

1.1.4.1 Réseaux de Petri

Les réseaux de Petri (RdP), sont un outil graphique à support mathématique permettant de modéliser, visualiser et analyser des évolutions de ces systèmes [Ferrier, 2004].

Ils sont utilisés pour modéliser, analyser et concevoir des systèmes à évènements discrets, incluant les systèmes de production, les systèmes informatiques, les systèmes de communication…etc. Les réseaux de Petri présentent également deux caractéristiques principales : Premièrement, ils permettent la modélisation comportementale comprenant le parallélisme, la synchronisation et le partage de ressources. Deuxièmement, Le modèle réseau de Petri peut être utilisé pour implémenter les systèmes de contrôle en temps réel pour ces systèmes [Uzam, 1998].

L'état du système modélisé par un RdP est représenté par un vecteur de marquage définissant le nombre de jetons que contient chaque place. L'évolution de l'état (dynamique du système) correspond donc à une évolution du marquage. L'évolution du marquage se produit par le franchissement de transitions: à l'occurrence d'un événement correspond le franchissement d'une transition. L'espace des marquages accessibles tient lieu d'espace d'états dans le système.

1.1.4.2 Algèbre des dioïdes

La structure algébrique de dioïde permet de modéliser et d'évaluer les performances de certains systèmes à événements discrets. C'est une structure algébrique munie d'une addition associative, commutative, avec élément neutre ("zéro"), et d'une multiplication associative, avec élément neutre ("un"), la multiplication étant distributive par rapport à l'addition, le zéro étant absorbant pour la multiplication (zéro fois *a* égale zéro pour tout *a*) ; et l'addition est idempotente, c'est-à-dire que *a* plus *a* égale *a* pour tout *a*. Elle est basée essentiellement sur les graphes d'événements temporisés qui sont une sous-classe des réseaux de Petri, avec des équations d'état linéaires [Ferrier, 2004].

A la différence du modèle RdP, l'état considéré ici pour aboutir à cette représentation linéaire est associé non plus aux places mais à leurs transitions. Le temps, associé aux événements, est intégré de manière naturelle à cette classe de modèles. L'état du système est modélise dans ce cas par un ensemble d'équations linéaires plus un graphe d'événements [Uzam, 1998].

1.1.4.3 Langages et Automates

Les langages et les automates permettent de traiter mathématiquement les problèmes relatifs aux DES, essentiellement d'un point de vue logique. La théorie des automates à états finis a été principalement développée avec la théorie des langages. Ces modèles reviennent à spécifier des ensembles d'états et des transitions entre ces états.

Chaque DES a un ensemble d'événements qui lui est associé, ces événements font l'évoluer dans le temps. Cet ensemble peut être vu comme un alphabet d'un langage et les séquences d'événements sont des mots (aussi appelés chaînes) de ce langage. Un automate est alors un dispositif qui engendre un langage en manipulant l'alphabet (les événements). [Ferrier, 2004].

1.1.5 Théorie de supervision pour la commande des DES

Les systèmes à événement discrets sont des systèmes qui sont caractérisés par les séquences d'évènements qu'ils peuvent accepter ou exécuter [Cantarelli, 2006].

L'objectif d'une commande est d'imposer au procédé un comportement spécifié, tout en respectant un ensemble de contraintes [Ferrier, 2004].

Le contrôle de ces systèmes est initié par Ramadge et Wonham, qui ont proposé une théorie de supervision (SCT). Le travail est développé autour des langages réguliers formels générés par des automates d'états finis [Pinzon, 1997].

1.1.5.1 Définitions

La théorie de supervision pour un Système à événements Discrets consiste à proposer des algorithmes permettant de définir automatiquement un modèle de contrôle et de commande, à partir de modèles formels de la dynamique d'un procède à contrôler [Wonham, 2008].

Dans cette théorie, le système à événements discrets à contrôler est modélisé par un système de transitions étiquetées **P** (plant model), appelé le générateur. Les étiquettes indiquent le phénomène physique qui a causé la transition [Pinzon, 1997].

Le comportement du DES est modélisé par un langage formel **L** généré par le système de transitions. L'alphabet du langage $\sum$ est un ensemble d'évènements. Ces évènements sont

identifiés à l'aide d'une spécification $\mathbf{S_p}$. La spécification est une description du comportement voulu du DES, elle est décrite sous forme d'un langage formel, généré par un système de transitions **H**.

La caractéristique du contrôle réside dans le fait que certains événements (transitions) peuvent être empêchés par un contrôleur externe. Ce sont des évènements contrôlables et les autres sont des évènements non contrôlables

Un contrôleur **C** est construit à partir d'un système de reconnaissance **G** et le système de transitions **H** [Wonham, 2008].

1.1.5.2 Démarche de conception d'un superviseur de contrôle

Soit les générateurs **G** et **H** définie chacun comme un 5-tuplet: $\mathbf{(Q,\sum,\delta,q_0,q_m);}$

§ **Q:** est un ensemble d'états ;

§ $\sum$ est un ensemble fini d'alphabet interprété comme l'ensemble des évènements dans le DES ;

§ $\mathbf{\delta: \sum \times Q \rightarrow Q}$ est une fonction de transition **;**

§ $\mathbf{q0 \in Q}$ **:** est un état initial ;

§ $\mathbf{q_m \in Q}$**:** est un état final.

Le DES sera modélisé via un ensemble de sous générateurs **G1, G2…….Gn**, présentant chacun un sous système du DES. Les actions de ces générateurs sont asynchrones et indépendantes, donc leurs alphabets sont disjoints et toute interaction entre ces sous générateurs est exprimée dans la spécification $\mathbf{S_p}$ [Pinzon, 1997].

La démarche de conception d'un superviseur de contrôle pour un DES consiste en six étapes :

1- Modéliser les sous générateurs pour l'ensemble des sous systèmes du DES.

2- Construire le générateur **G = G1 ||G2 … ||Gm**. G par une concaténation de l'ensemble des sous générateurs **Gi.**

3- Construire les sous générateurs de spécifications **Hj**.

4- Augmenter les spécifications **Hj**, par des boucles étiquetées par les évènements présents dans **G** mais ne sont pas dans **Hj**.

5- Construire la spécification globale $\mathbf{H = H1 \cap H2 \ldots \cap Hm}$. par une intersection des systèmes de transitions **Hj** de l'ensemble des sous générateurs **Gi. H** va forcer le comportement de **G.**

6- Construire l'intersection **E = H ∩ G**. **E** représente le comportement globale du système qui vérifie toutes les contraintes de la spécification.

Le générateur obtenu **E** ne présente pas un superviseur parfait, due au fait qu'il ne vérifie pas les deux propriétés suivantes :

§ Le Non blocage, c'est à dire le générateur **E** peut contenir des états de blocage à partir des quelles l'état final ne peut pas être atteindre.

§ Contrôlabilité, c'est à dire le langage **L(E)** peut être non contrôlable par apport à **L (G)**. Cela signifie que lorsque **G** et **E** s'exécutent en parallèle, on arrive a un état à partir du quel un évènement non contrôlable est permis dans **G** mais pas dans **E** [Giua, 1992]**.**

Nous devons donc minimiser le comportement de **E**, pour obtenir un générateur contrôlable et non bloquant **S**. Le système **S** obtenu par cette procédure nous donne la structure de transition d'un superviseur de contrôle.

1.1.5.3 Exemple de conception d'un superviseur de contrôle

Nous donnerons ici un exemple de conception utilisant un modèle d'automate d'états finis.

Considérant le système **G1** et **G2** de la Figure 1.3 et la spécification **H.** Nous pouvons penser à **G1** comme un robot qui prend une partie (l'événement **a**) et porte la partie sur une machine (l'événement **b**). **G2** est une machine qui peut commencer à travailler (l'événement **c**) et peut transporter la partie produite à un transporteur **A** ou **B** (respectivement les événements **d** et **e**) avant de retourner à l'état disponible. La spécification que nous considérons, est représentée par le générateur **H**, spécifie que la machine peut commencer à travailler seulement après qu'elle a été chargé (les événements **b** et **c** doivent arriver alternativement). Et que le robot peut charger une nouvelle partie sur la machine seulement après que la partie précédemment chargée a été transportée au transporteur (des événements **d** et **b** doivent arriver alternativement). Nous supposerons que le seul événement incontrôlable est l'événement **b**.

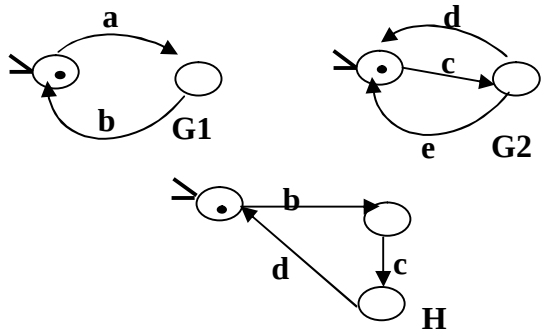

Figure 1.3 Les systèmes et la spécification pour l'exemple [Giua, 1992]**.**

La deuxième étape de conception exige la construction de **G = G1 || G2**, comme il est illustré dans la Figure 1.4.

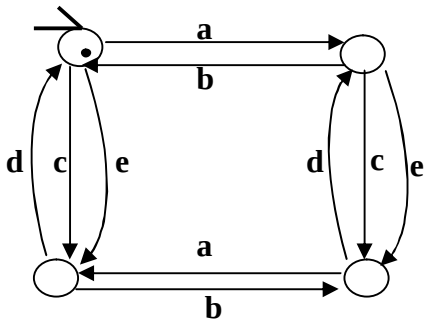

Figure 1.4 Concaténation de G1 et G2 [Giua, 1992]**.**

Dans la troisième étape nous construisons la spécification **H'** augmentée (Figure 1.5), par l'ajout des boucles à **H** avec les événements rencontrés **a** et **e**. La cinquième étape n'est pas nécessaire puisqu'il y a une seule spécification dans cet exemple.

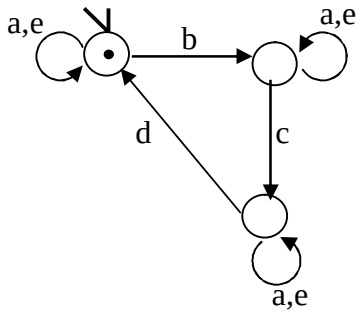

Figure 1.5 Générateur de la spécification H' [Giua, 1992]**.**

Dans la sixième étape, nous construisons le système **E = G ∩ H'** (Figure 1.6). Le système **E** n'est pas un superviseur parfait, car, il contient des états bloquants (depuis les quelles l'état final ne peut pas être accessible), ainsi que des états non contrôlables .Par exemple pour la chaîne d'événements **w = aba**, les évènements **b** et **c** peuvent apparaître dans **G**, mais le modèle de contrôle engendré par **E**, contient seulement l'évènement **c** .Bien que l'évènement **b** est non contrôlable, donc le langage généré par **E** est non contrôlable.

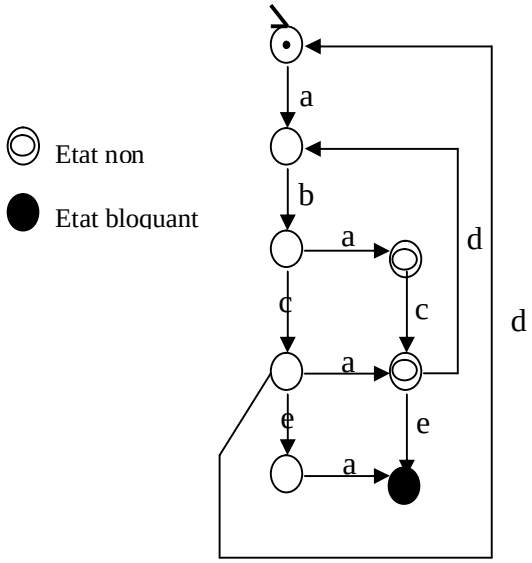

Figure 1.6 Le système E = G ∩H' [Giua, 1992]**.**

Si nous éliminons les états non contrôlables et les états bloquants, nous obtenons un générateur **S** non bloqué et contrôlable comme il est montré dans la Figure 1.7, qui est un superviseur parfait pour cet exemple.

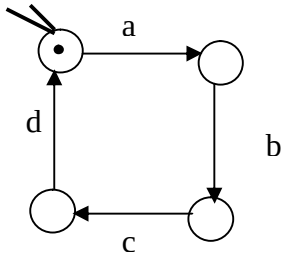

Figure 1.7 Le superviseur de contrôle S [Giua, 1992]**.**

1.2 Systèmes de productions

1.2.1 Définitions

Un système de production est un système qui vise à produire des éléments. C'est un ensemble de processus, chaque processus de production est constitué d'un ensemble de phases de production. Une phase de production est une activité ou un ensemble d'activités [Toshic, 2006], de natures différentes qui transforment les éléments dans le processus de production.

D'une part, il y a les activités de nature physique telles que le stockage, le transfert, la transformation, le contrôle ...etc qui peuvent être groupées dans un processus de production. D'autre part, il y a les activités de nature logique (décisionnelle) telles que l'ordonnancement, la définition de règles de priorité, la réaction à une perturbation, la programmation d'une action préventive ou corrective ..etc, qui peuvent être groupées dans un processus de contrôle. [Habchi, 2001].

Une autre définition est donnée par [Holmstrom, 2006], qui considère qu'un système de production est un ensemble d'objets et d'évènements qui sont connectés et contrôlés dans le temps et dans l'espace qu'ils occupent , dans le but d'atteindre des fonctions précises.

1.2.2 Composants d'un système de production

Un système de production est constitué de trois sous-systèmes qui coopèrent :

§ Le sous-système physique représentant le système opérant [Habchi, 2001]. Il agit directement sur les produits en effectuant des opérations de transformation, de contrôle, de manutention et de stockage [Gaillard, 2002].

§ Le sous-système d'information permettant l'acquisition, le traitement et la gestion des données du système et de son environnement.

§ Le sous-système de décision**,** ce système, est appelé aussi système de conduite ou de pilotage. Il a pour rôle de modifier l'évolution du système physique [Gaillard, 2002].

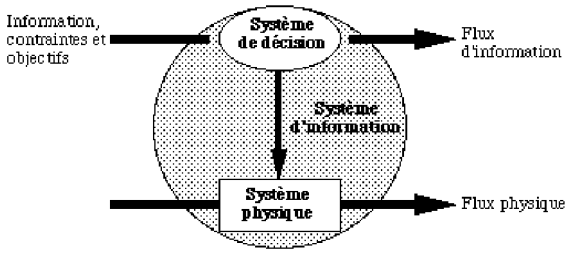

Figure 1.8 Composants d'un système de production [Gaillard, 2002].

Si cette décomposition est valable pour le système entrepris, et permet son analyse, elle est moins adaptée pour un système de production et sa modélisation. Dans ce dernier, les sous systèmes d'information et de décision n'ont pas d'existence propre, l'un sans l'autre. Ils constituent ensemble ce que nous appelons le système de pilotage ou le système d'information et de décision (SID) ou encore le système directeur [Habchi, 2001].

1.2.2.1 Système de fabrication (système physique)

Le système de fabrication est composé de ressources et d'entités. Les ressources comprennent les machines, les stocks, les opérateurs, les moyens de transfert...etc, et les entités comprennent les produits, les matières premières, les pièces, les lots...etc [Habchi, 2001]. Donc un système de fabrication est un ensemble de ressources qui effectuent des opérations de transformation sur les entités.

Les ressources peuvent être identifiées par deux types :

§ Ressources principales (machines, robots, moyens de transfert, stocks,…) qui sont actives et qui ont une certaine autonomie vis-à-vis du reste du système. Elles peuvent évoluer dans certaines limites, indépendamment du reste du système.

§ Ressources auxiliaires (outils, palettes,…) qui sont passives et qui permettent à des ressources principales d'accomplir une opération. Ce type de ressource est caractérisé par son exclusivité ou sa partageabilité [Habchi, 2001].

1.2.2.2 Système de pilotage

Piloter un système, c'est choisir un objectif par rapport auquel il faut définir la meilleure trajectoire. Le pilotage a pour but d'assurer la cohérence des décisions entre des ordres issus de la gestion prévisionnelle et les actions exécutées au niveau du système de production. Un système de pilotage peut être composé de:

§ Des points de capture (les capteurs) pour la récupération de l'information.

§ Un processus d*e* supervision pour l'analyse, le traitement de l'information, l'évaluation et la génération de décisions.

§ Des points d'action qui constituent les points de passage des ordres ou des actions vers le système physique [Habchi, 2001].

La réactivité du système de production dépend en premier lieu de la capacité de réaction de son système de pilotage. Cette capacité dépend de la qualité et de la quantité des points de capture et des points d'action, d'une part, et de l'efficacité du processus de supervision, d'autre part.

1.2.3 Caractéristiques des systèmes de production

Les systèmes de production sont basés sur des caractéristiques, telles que la flexibilité, la réactivité, la proactivité et la robustesse.

1.2.3.1 Flexibilité

L'évolution croissante des besoins d'une entreprise fait que la conception du système de production est de plus en plus orientée vers des familles de produits et non vers un seul type de produit. Les systèmes correspondants à une telle exigence doivent se révéler flexibles.

La flexibilité d'un système de production se caractérise par sa capacité d'adaptation à la production de nouveaux produits pour lesquels le système n'a pas été étudié [Habchi, 2001].

Plusieurs types de flexibilité ont été mis en évidence :

- Flexibilité de produits: offre la possibilité d'une reconFigureuration du système pour la prise en compte d'un nouveau produit ou famille de produits permettant ainsi un gain de productivité.
- Flexibilité de mélange : c'est la possibilité de produire simultanément un ensemble de produits ayant des caractéristiques de base communes ; cette flexibilité peut être mesurée par le nombre de produits différents qui peuvent être fabriqués simultanément.
- Flexibilité de quantité : il s'agit de la capacité du système à faire face aux fluctuations de la quantité des produits à fabriquer en modifiant les rythmes, ainsi que les temps de passage et d'engagement des outils.
- Flexibilité d'ordre des opérations : permet de changer l'ordre des opérations en cours de production et choisir la destination suivante après chaque opération.
- Flexibilité d'expansion : autorise une extension et une modification de l'architecture du système et elle exige une phase de modélisation.
- Flexibilité des ressources : c'est la capacité des ressources à effectuer plusieurs tâches élémentaires et de permettre leur reprogrammation [Habchi, 2001].

1.2.3.2 Réactivité

Un système de production réactif, est un système qui est capable de répondre rapidement et économiquement à un changement ou à un aléa. Ces aléas peuvent provenir soit du système de production (défauts de réalisations d'une tâche, pannes des machines, …etc) soit de son environnement (approvisionnements des matières premières).

La réactivité d'un système de production est définie comme l'aptitude à répondre (réagir) dans un temps requis aux changements de son environnement interne ou externe [Habchi, 2001].

1.2.3.3 Proactivité

La réactivité est nécessaire, mais elle n'est pas suffisante et les systèmes de production doivent présenter une nouvelle propriété qui est la proactivité.

La proactivité d'un système de production se caractérise par ses capacités d'anticipation (prévoir et/ou provoquer) les changements d'état, d'apprentissage et d'enrichissement des connaissances (pour améliorer sa réactivité) [Habchi, 2001]. Donc, un système de production proactif est avant tout un système réactif.

1.2.3.4 Robustesse

La robustesse d'un système de production se définit par son aptitude à produire conformément aux résultats attendus. Cela suppose la garantie de l'obtention des performances souhaitées en présence d'incertitudes dans le système [Habchi, 2001].

1.2.4 Processus de production

Un processus de production est un élément actif dans un système de production, son but est de produire des produits [Holmstrom, 2006]. Un processus de production est un ensemble d'activités et d'évènements et des objets qui sont en relation.

Une activité est un évènement dont lequel un objet acte pour changer son état ou l'état d'un autre objet, hors un évènement est une occurrence qui cause un changement d'état d'un objet [Holmstrom, 2006].

Un objet dans un processus de production est une ressource principale ou auxiliaire .Son état est une condition ou une situation durant son cycle de vie . Cette situation peut être la réalisation d'une activité ou l'attente d'un événement ou la satisfaction d'une condition [Holmstrom, 2006].

1.2.5 Complexité des systèmes de production

Durant ces dernières années, les systèmes de production sont devenus plus complexes. Cette complexité due à plusieurs facteurs, qui sont :

- § Les produits sont devenus et continuent à devenir plus complexes .Cela implique que la complexité de l'information nécessaire pour produire ces produits et le système qui produit sont ainsi complexes.
- § La technologie de production est devenue elle même complexe.
- § La taille des systèmes de production est grandis partiellement due à la complexité des produits et le nombre des variants de produits.
- § Les besoins du marché sont augmentés et les clients demandent toujours de la marque et de la qualité des produits .Cela a conduit à une évolution dans les activités de production [Petit, 1999].

1.2.6 Classes des systèmes de production

Les systèmes de production sont classifiés en 2 classes principales :

1.2.6.1 Systèmes de production distribués

Un système de production distribué est un système composé de plusieurs entités, qui sont en coopération, reliées par un réseau de communication, chacune représente une partie opérationnelle du système de production.

Un système de production distribué est caractérisé par un ensemble de propriétés, parmi ces propriétés :

- § L'autonomie, qui est la possibilité de créer et de contrôler l'exécution du plan de production.
- § La distribution, qui permet a toutes les entités du système d'opérer dans le système.
- § La décentralisation, indique qu'une opération peut être exécutée par plusieurs entités.
- § La dynamique, qui signifie le changement de la structure et du comportement du système durant l'exécution d'une opération [Sousa, 2008].

1.2.6.2 Systèmes de production flexibles

Un système de production flexible FMS est un ensemble de machines multi-fonctionnaires et d'équipements qui peuvent être réorganisés périodiquement. Il permet de lancer un nouveau programme de production à chaque réorganisation [Andrea, 2006].

L'objectif des FMS est de réaliser :

§ Une adaptation rapide aux types multiples de produits et aux programmes multiples de production.

§ Un taux maximum d'utilisation (100% si possible) des machines et d'équipements.

Un FMS est composé de deux systèmes, un système physique et un système de contrôle. Chacun est divisé en des sous systèmes et chaque sous système en des activités. Deux types d'activités sont considérés, des activités de décision et des activités d'exécution [Bérard, 1998].

1.2.7 Supervision des systèmes de production

1.2.7.1 Réflexions sur les terminologies Commande, Surveillance et Supervision

a. Commande

La commande a pour rôle de faire exécuter un ensemble d'opérations en fixant des consignes de fonctionnement en réponse à des ordres d'exécution. Il peut s'agir de réaliser :

- § Une séquence d'opérations constituant une gamme de fabrication dans le but de fabriquer un produit en réponse à une demande d'un client,

§ Une séquence d'actions corrective destinée à rendre au système de production toute ou partie des fonctionnalités requises pour assurer sa mission. La perte d'une partie des fonctionnalités initialement disponibles faisant suite à l'occurrence d'une défaillance,

§ Des actions prioritaires et souvent prédéfinies sur le procédé dans le but d'assurer la sécurité de l'installation et du personnel.

§ Des opérations de test, de réglage, de nettoyage permettant de garantir que le système de production pourra continuer d'assurer sa fonction.

b. Surveillance

La surveillance est limitée aux fonctions qui collectent des informations, les archivent, font des inférences, etc. sans agir réellement ni sur le procédé ni sur la commande. La surveillance a donc un rôle passif vis-à-vis du système de commande et du procédé.

c. Supervision

La supervision permet de contrôler et surveiller l'exécution d'une opération ou d'un travail effectué par d'autres sans rentrer dans les détails de cette exécution. La supervision recouvre l'aspect fonctionnement normal et anormal.

§ En fonctionnement normal, son rôle est surtout de prendre en temps réel des décisions. Pour cela elle est amenée à faire de l'ordonnancement en temps réel, à modifier la commande et à gérer le passage d'un algorithme de surveillance à l'autre.

§ En présence de défaillance, la supervision va prendre toutes les décisions nécessaires pour le retour vers un fonctionnement normal. Après avoir déterminé un nouveau fonctionnement, il peut s'agir de choisir une solution curative, d'effectuer des réordonnancements locaux, de prendre en compte la stratégie de supervision de l'entreprise, de déclencher des procédures d'urgence, etc. [Combacau, 2000]

Toutes ces terminologies s’intègrent dans le pilotage ou la conduite des systèmes de production.

1.2.7.2 Superviseur pour les systèmes de production

L’objectif principal de la supervision de la production est la bonne exécution du programme prévisionnel de production par le système physique, en garantissant sa sûreté de fonctionnement.

La supervision d’un système inclut des fonctions de collecte et de visualisation d’informations, de surveillance, de diagnostic et d'aide à la prise de décision pour

l'accommodation, la reconfiguration ou la maintenance. Un superviseur peut être décrit comme le montre la Figure 1.9.

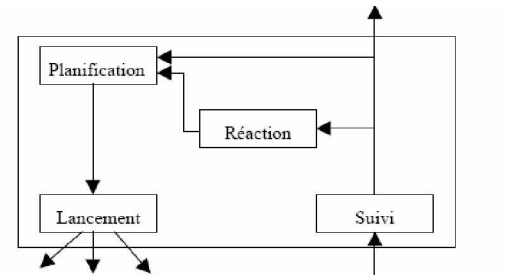

Figure 1.9 Modèle d'un superviseur [draghupb, 1998].

- **La planification**, qui consiste à mettre en oeuvre des techniques d'ordonnancement. Intégrées dans le processus global de supervision, la planification propose une affectation pour les différentes opérations, dans le temps et l'espace. C'est un moyen unique et incontournable pour s'assurer du respect des objectifs fixés.
- **Le lancement,** qui répartit et transmet les ordres au système physique en tenant compte de l'état des entités de production.
- **Le suivi**, qui recueille l'ensemble des événements survenant dans le système et qui met à jour une image interne du système opérant en autorisant ou inhibant certaines opérations toutes en assurant le bon fonctionnement du système.
- **La réaction**, qui corrige les déviations induites par les aléas de production. Cette activité est très liée à celle de la planification, car il s'agit de prendre des mesures correctives tout en s'assurant du respect des objectifs de production [draghupb, 1998]

Dans la littérature des systèmes de contrôle, un superviseur est un contrôleur qui utilise les données disponibles pour caractériser le comportement actuel du système.

La supervision d'un système de production consiste à utiliser un système de décision pour faire exécuter par le système physique l'ensemble des opérations de fabrication qui lui sont affectées :

- En respectant au mieux les objectifs de production fixés, tout en satisfaisant les contraintes spatiales, temporelles et de coûts.
- En s'assurant que chaque ordre transmit est cohérent vis-à-vis l'environnement dans lequel évolue le système.
- En utilisant un système d'information cohérent réalisant une interface robuste entre le système physique et le système de décision.

1.2.7.3 Structures de supervision d'un système de production

On peut distinguer essentiellement cinq structures différentes de supervision :

- § **Structure centralisée**, caractérisée par un superviseur localisé au sein d'une ressource unique qui supervise la production.
- § **Structure hiérarchisée**, dont chaque niveau coordonne les unités de supervision du niveau inférieur, et ce jusqu'au niveau le plus bas.
- § **Structure coordonnée**, correspond à un ensemble de structures hiérarchisées où une coopération est possible au sein d'un même niveau de supervision.
- § **Structure distribuée**, fondée sur une distribution totale des capacités de supervision.
- § **Structure distribuée supervisée**, caractérisée par un ensemble d'entités coopérantes sous le contrôle d'une entité superviseur dont le rôle est d'imposer, de conseiller ou de modifier une décision afin de respecter un objectif plus global.

Le choix de la structure de supervision dépend essentiellement d'un compromis entre le degré d'autonomie et la cohérence de décision dans le système de production.

1.2.8 Modélisation des systèmes de production en vue de leur supervision

La modélisation des systèmes de production est indispensable pour la compréhension et l'analyse des phénomènes mis en jeu dans ces systèmes. Donc La supervision de tels systèmes repose également sur l'utilisation de modèles. Ces modèles doivent rendre compte de la structure et du comportement du système et permettre l'analyse de ses propriétés structurelles et comportementales.

Ainsi, la mise en œuvre des différentes fonctions de supervision s'appuie sur des modèles de natures différentes: structurels ou fonctionnels, comportementaux, Ils font intervenir des informations de nature également différentes: numérique, logique, symbolique ou textuelle, ...etc [Draghupb, 1998].

La partie supervision d'un système de production doit disposer d'un modèle de la partie opérative (ou physique) du système, afin de réaliser son objectif. Dans ce but, plusieurs approches de modélisation des systèmes de production ont été envisagées. Nous considérons deux types de modélisation :

1.2.8.1 Modélisation structurelle

La modélisation structurelle est relative à l'expression des besoins du système et à sa structure. Parmi les approches de modélisation existantes :

a. Approche orientée données : Entité / Association

C'est une méthode conceptuelle Apparue vers le milieu des années 60, elle définit une structure générale de données, indépendante de programmes qui les manipulent. Elle s'intéresse aux aspects structurels et informationnels. Néanmoins cette approche n'intègre pas les aspects dynamiques, son modèle étant par définition statique ; ainsi les conditions de déclenchement, l'ordonnancement dans le temps sont difficilement représentables.

b. Approche orientée objets

L'approche objet est adaptée à la problématique de l'étude des systèmes de production. En effet, les mécanismes objets tels que l'abstraction, l'encapsulation, la modularité, la classification, et l'héritage permettent de définir une méthodologie s'appuyant sur des variations mineures d'entités préétablis et de les adapter aux besoins du système étudié.

La démarche de modélisation par objets a pour objectif de définir les objets du système de production identifiés lors du processus de conception. Une première étape correspond à la définition des entités génériques ou objets abstraits, ensuite l'exploitation de ces entités génériques. L'utilisateur de l'approche pourra adapter les modèles proposés à son besoin, en définissant de cette manière l'architecture de supervision du système de production considéré. Parmi les formalismes utilisés, le langage UML que nous allons l'étudier dans la suite.

c. Approche multi-agents

Les systèmes multi-agents ont été développés dans le cadre de l'intelligence artificielle distribuée. L'intérêt qu'ils suscitent est lié à leur capacité d'aborder les problèmes complexes d'une manière distribuée et de proposer des solutions réactives et robustes.

Dans la supervision des systèmes de production, la coopération entre les agents et la communication par passage de messages est indispensable pour résoudre des conflits pouvant apparaître, pour l'allocation des ressources limitées, pour réconcilier des préférences différentes et rechercher des solutions dans un espace globale à partir des informations locales.

Les systèmes multi-agents ajoutent à la localité de comportements, présentée dans l'approche objet, l'autonomie et la répartition de prise de décisions. En effet, l'agent, à l'encontre de l'objet, peut effectuer un certain travail, le refus pouvant s'expliquer par son manque de compétence (il ne possède pas le savoir-faire nécessaire) ou par sa trop grande occupation à une autre tâche ou par toute autre raison [draghupb ,1998].

1.2.8.2 Modélisation comportementale

La modélisation comportementale est relative à la spécification du comportement dynamique de l'ensemble des objets du système de production et sa stratégie de production.

Plusieurs types de modèles sont adaptés pour ce type de modélisation, parmi ces modèles, nous pouvons citer :

a. Les modèles semi-formels, comme les diagrammes dynamiques d'UML, ou REMORA, qui permettent de représenter différentes activités, leurs flux d'entrées et de sorties, informationnels et matériels, à différents niveaux de détail.

b. Les modèles formels, comme les automates d'états finis , les réseaux de Petri, les state-charts, les processus communicants de type CSP basés sur la logique temporelle etc , qui permettent de modéliser la dynamique des systèmes et de les analyser de manière formelle .

1.2.8.3 Langage de modélisation UML

a. Présentation

En 1994, Rumbaugh et Booch unissent leurs efforts pour mettre au point la méthode unifiée(Unified Method 0.8), La méthode unifiée à partir de la version 1.0 devient UML (Unified Modeling Language) [Frédéric, 2001]. UML est le standard de l'OMG (Object Groupe Management) pour la modélisation orientée objet des systèmes logiciels. Il propose un ensemble de notations, sous forme de diagrammes.

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation), c'est un langage graphique qui permet de représenter, de communiquer les divers aspects d'un système [Frédéric, 2001], il a été pensé pour permettre la modélisation des activités de l'entreprise, pas pour les régir.

UML est un langage pseudo-formel, il est fondé sur un méta-modèle, qui est une description très formelle de tous les concepts d'un langage,

UML 2.0 définit treize types de diagrammes qui peuvent être divisés en deux catégories. Six types de diagrammes pour la modélisation de la structure statique du système, sept sont pour les différents aspects de la modélisation dynamique du système.

Catégorie	Diagrammes
Diagrammes structurels ou	§ Diagramme de classes (Class diagram)

diagrammes statiques	§ Diagramme d'objets (Object diagram) § Diagramme de composants (Component diagram) § Diagramme de déploiement (Deployment diagram) § Diagramme de paquetages (Package diagram) § Diagramme de structures composites (Composite structure diagram)
Diagrammes comportementaux ou diagrammes dynamiques	§ Diagramme de cas d'utilisation (Use case diagram) § Diagramme d'activités (Activity diagram) § Diagramme d'états-transitions (State machine diagram)
	• **Diagrammes d'interaction (Interaction diagram)** § Diagramme de séquence (Sequence diagram) § Diagramme de communication (Communication diagram) § Diagramme global d'interaction (Interaction overview diagram) § Diagramme de temps (Timing diagram)

Tab 1.1 Les différents diagrammes d'UML [Audibert, 2006].

Ces diagrammes, sont d'une utilité variable selon les cas, ils ne sont pas nécessairement tous produits à l'occasion d'une modélisation. Les plus utiles sont les diagrammes d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions.

b. Sémantique à base de méta-modélisation

Depuis ses premières versions, le standard UML est caractérisé par sa sémantique définie par une approche de méta-modélisation. Un méta-modèle est la définition des constructions et des règles de création des modèles. Le méta-modèle d'UML définit donc la structure que doit respecter tout modèle UML [UML, 2007].

L'approche de méta-modélisation adoptée par l'OMG est connue comme une hiérarchie à quatre niveaux :

§ Niveau méta-méta-modèle (M3), il définit le langage de spécification du méta-modèle. Le MOF (Meta Object Facility) est un exemple d'un méta-méta-modèle.

§ Niveau méta-modèle (M2),le méta-modèle d'UML se situe à ce niveau et il est spécifié en utilisant le MOF, c.à.d. les concepts du méta-modèle d'UML sont des instances des concepts de MOF.

§ Niveau modèle (M1), qui correspond au niveau des modèles UML des utilisateurs.

§ Niveau objets (M0), qui correspond au niveau des objets à l'exécution [Ziadi, 2004].

Le méta-modèle d'UML est décrit en utilisant une partie de la notation d'UML lui même. Les concepts suivants sont utilisés :

§ Les classes d'UML, pour décrire les méta-classes.

§ Les attributs, pour décrire les propriétés attachées à une méta-classe.

§ Les associations, pour décrire des liens entre les méta-classes.

§ Les paquetages (packages), pour regrouper les méta-classses par domaine.

c. Diagrammes statiques d'UML

UML utilise des représentations par des diagrammes, pour modéliser les aspects statiques (structurels). Nous présentons ici deux diagrammes qui nous intéressent dans notre travail, qui sont le diagramme de cas d'utilisation et le diagramme de classes.

c.1 Diagramme de cas d'utilisation (Use Cases Diagram**)**

Un diagramme de cas d'utilisation est un diagramme qui est utilisé pour donner une vision globale du comportement fonctionnel d'un système. Il permet d'identifier les possibilités d'interaction entre le système et les acteurs (extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.

C'est le premier diagramme du modèle UML, celui où s'assure la relation entre l'utilisateur et les objets que le système met en œuvre. C'est donc une vue du système dans son environnement extérieur [Federic, 2001] [Audibert, 2006] [UML, 2007].

Ø Eléments des diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation est constitué de deux éléments, comme le montre la Figure suivante :

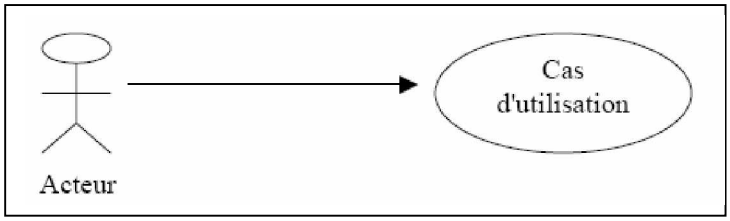

Figure 1.10 Les éléments constituants le diagramme de cas d'utilisation [Federic, 2001].

§ **Acteur**

Un acteur est une entité externe qui agit sur le système ; Le terme acteur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes. [Federic, 2001]. Un acteur se représente par un petit bonhomme (Figure 1.10) avec son nom (i.e. son rôle) inscrit dessous, il est également possible de le représenter sous la forme d'un classeur(rectangle) stéréotypé (étiqueté) par le mot « actor » et son rôle est inscrit dessous (Figure 1.11)

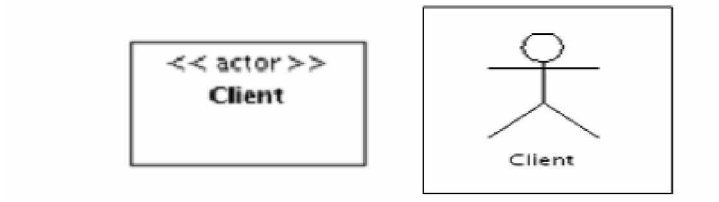

Figure 1.11 Représentation d'un acteur [Audibert, 2006].

Il existe deux catégories d'acteurs :

1. Les acteurs principaux : Cette catégorie regroupe les personnes qui utilisent les fonctions principales du système. Par exemple dans un système distributeur de billets, il s'agit des clients [UML, 2007].

2. Les acteurs secondaires. Cette catégorie regroupe les personnes qui effectuent des tâches administratives ou de maintenance. Dans le même exemple du distributeur de billets, il s'agit de la personne qui recharge la caisse du distributeur [UML, 2007].

§ **Cas d'utilisation**

Un cas d'utilisation est un ensemble d'actions réalisées par le système en réponse à une action d'un acteur [Federic, 2001]. Il décrit les objectifs du système et modélise un service rendu par le système, sans imposer le mode de réalisation de ce service. Le service est visible de l'extérieur.

Un cas d'utilisation se représente par une ellipse (Figure 1.12) contenant le nom du cas (un verbe à l'infinitif), et optionnellement, au-dessus du nom, un stéréotype (Une étiquette). Dans le cas où l'on désire présenter les attributs ou les opérations du cas d'utilisation, il est préférable de le représenter sous la forme d'un classeur« use case.

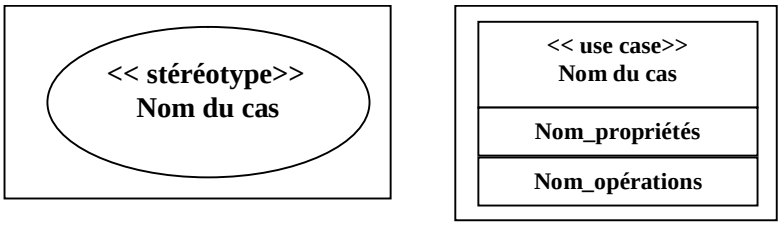

Figure 1.12 Représentations d'un cas d'utilisation [Audibert, 2006].

Ø Représentation d'un diagramme de cas d'utilisation

Comme le montre l'exemple de la Figure (1.13), la frontière du système est représentée par un cadre. Le nom du système Figure à l'intérieur du cadre, en haut. Les acteurs sont à l'extérieur et les cas d'utilisation sont à l'intérieur.

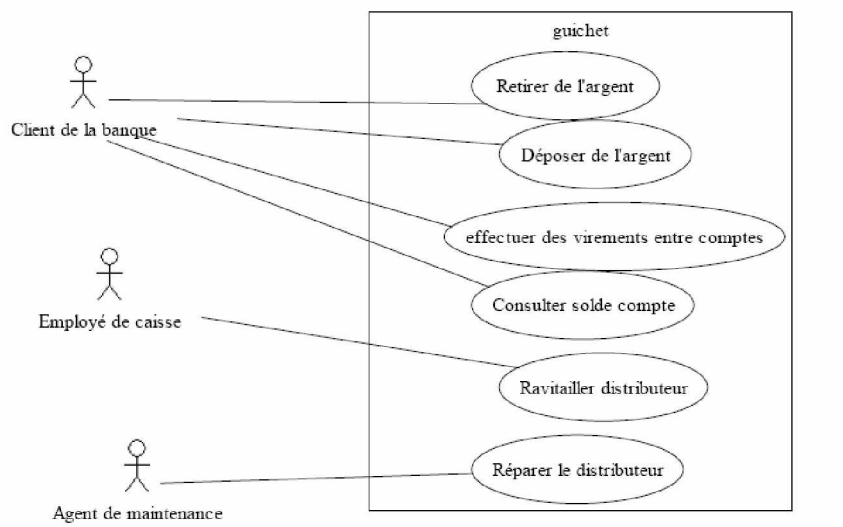

Figure 1.13 Exemple d'un diagramme de cas d'utilisation d'un guichet automatique bancaire [Federic, 2001].

Ø Description textuelle des cas d'utilisation

Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs, mais n'expose pas de façon détaillée le dialogue entre les acteurs et les cas d'utilisation. Bien que de nombreux diagrammes d'UML permettent de décrire un cas, il est recommandé de rédiger une description textuelle car c'est une forme souple qui convient bien dans des situations.

Une description textuelle couramment utilisée se compose de trois parties :

1. La première partie permet d'identifier les cas, elle doit contenir les informations suivantes :

- § **Nom :** Utiliser une tournure à l'infinitif (exp : Réceptionner un colis).
- § **Objectif :** Une description résumée permettant de comprendre l'intention principale du cas d'utilisation.
- § **Acteurs principaux :** Ceux qui vont réaliser le cas d'utilisation (la relation avec le cas d'utilisation est illustrée par le trait liant le cas d'utilisation et l'acteur dans un diagramme de cas d'utilisation).

§ **Acteurs secondaires :** Ceux qui ne font que recevoir des informations à l'issue de la réalisation du cas d'utilisation.

§ **Dates :** Les dates de création et de mise à jour de la description courante.

§ **Responsable :** Le nom des responsables.

§ **Version :** Le numéro de version.

2. La deuxième partie contient la description du fonctionnement du cas sous la forme d'une séquence de messages échangés entre les acteurs et le système. Elle contient toujours une séquence nominale qui décrit le déroulement normal du cas. A la séquence nominale s'ajoutent fréquemment des séquences alternatives (des embranchements dans la séquence nominale) et des séquences d'exceptions (qui interviennent quand une erreur se produit).

§ **Préconditions :** elles décrivent dans quel état doit être le système avant que ce cas d'utilisation puisse être déclenché.

§ **Scénarios :** Ces scénarii sont décrits sous la forme d'échanges d'évènements entre l'acteur et le système. On distingue le scénario normal, qui se déroule quand il n'y a pas d'erreur, et les scénarios d'exception qui décrivent les cas d'erreurs.

§ **Des postconditions :** Elles décrivent l'état du système à l'issue des différents scénarios.

3. La troisième partie de la description d'un cas d'utilisation est une rubrique optionnelle, elle contient généralement des spécifications non fonctionnelles (spécifications techniques,...). Elle peut éventuellement contenir une description des besoins en termes d'interface graphique [Audibert, 2006].

Ø **Modélisation avec le diagramme de cas d'utilisation**

Un bon diagramme de cas d'utilisation doit être simple avec un nombre d'acteurs limité, la démarche de construction d'un diagramme de cas d'utilisation doit contenir les points suivants :

1. Lister les acteurs.
2. Déterminer le rôle de chaque acteur.
3. Déterminer les cas d'utilisation.
4. Identifier le flot d'évènements auxquels le système doit réagir.
5. Structurer les cas d'utilisation.
6. Finaliser un ou plusieurs diagrammes par package [UML, 2007].

Ø Identification des acteurs dans un diagramme de cas d'utilisation

Les acteurs d'un système sont les entités externes à ce système qui interagissent (saisie de données, réception d'information, ...) avec lui. Chaque acteur doit être nommé. Ce nom doit refléter sont rôle car un acteur représente un ensemble cohérent de rôles joués vis-à-vis du système.

Pour trouver les acteurs d'un système, il faut identifier quels sont les différents rôles que vont devoir jouer ses utilisateurs (ex: responsable clientèle, responsable d'agence, administrateur, approbateur, ...).

Pour faciliter la recherche des acteurs, nous pouvons imaginer les frontières du système. Tout ce qui est à l'extérieur et qui interagit avec le système est un acteur, tout ce qui est à l'intérieur est une fonctionnalité à réaliser. Par exemple, l'hôtesse de caisse d'un magasin de grande distribution est un acteur pour la caisse enregistreuse, par contre, les clients du magasin ne correspondent pas à un acteur car ils n'interagissent pas directement avec la caisse [Audibert, 2006].

Ø Identification des cas d'utilisation

L'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction métier du système, selon le point de vue d'un de ses acteurs. Aussi, pour identifier les cas d'utilisation, il faut se placer du point de vue de chaque acteur et déterminer comment et surtout pourquoi il se sert du système. Par exemple, un distributeur de billets aura probablement un cas d'utilisation Retirer de l'argent et non pas Distribuer de l'argent [Audibert, 2006].

c.2 Diagramme de classes

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs relations [Audibert, 2006].

Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation [UML, 2007].

Ø **Eléments du diagramme de classes**

Un diagramme de classes est constitué d'un ensemble de classes qui permettent de décrire un ensemble d'objets (attributs et comportement), et de relations entre ces classes.

§ **Classes**

Une classe est une description d'un ensemble d'objets partageant la même sémantique, ainsi que les mêmes attributs, opérations et relations [Federic, 2001].

Une classe est un concept abstrait représentant des éléments variés comme :

- Des éléments concrets (ex : des avions),
- Des éléments abstraits (ex : des commandes),
- Des composants d'une application (ex : les boutons des boîtes de dialogue),
- Des structures informatiques (ex : des tables de hachage),
-Des éléments comportementaux (ex : des tâches), etc [Audibert, 2006].

Une classe est représentée graphiquement par un rectangle divisé en trois compartiments. Le premier indique le nom de la classe, le deuxième ses attributs et le troisième ses opérations.

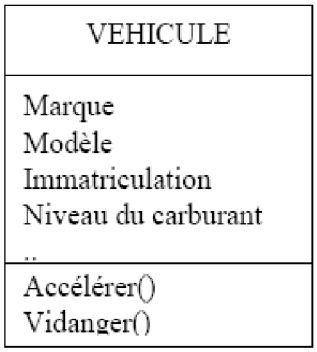

Figure 1.14 Exemple d'une classe [Federic, 2001].

1. **Nom d'une classe :** Le nom de la classe doit évoquer le concept décrit par la classe. nous pouvons ajouter des informations subsidiaires comme le nom de l'auteur de la modélisation, la date, etc [Federic, 2001].
2. **Attributs :** Les attributs définissent des informations qu'une classe ou un objet doit connaître. Ils représentent les données encapsulées dans les objets de cette classe. Chacune de ces informations est définie par un nom, un type de données, une visibilité et peut être initialisé. Le nom de l'attribut doit être unique dans la classe [Audibert, 2006].

3. **Opérations :** La définition d'une classe est complétée par l'ensemble des opérations qu'elle peut exécuter. Une opération est une fonctionnalité assurée par la classe. [Federic, 2001]. Dans une classe, une opération a un nom et des paramètres, elle doit être unique.

UML définit trois niveaux de visibilité pour les attributs et les opérations d'une classe :

1. **Public** qui rend l'élément visible à tous les clients de la classe,
2. **Protégé** qui rend l'élément visible aux sous classes de la classe,
3. **Privé** qui rend l'élément visible à la classe seule [Federic, 2001].

§ **Association entre classes**

Une association représente une relation structurelle entre classes d'objets. La plupart des associations sont binaires, c'est à dire qu'elles connectent deux classes. Nous présentons une association en traçant une ligne entre les classes associées. Les associations peuvent être nommées afin de faciliter la compréhension des modèles. Il est d'usage de nommer les associations par une forme verbale, comme le montre la Figure suivante :

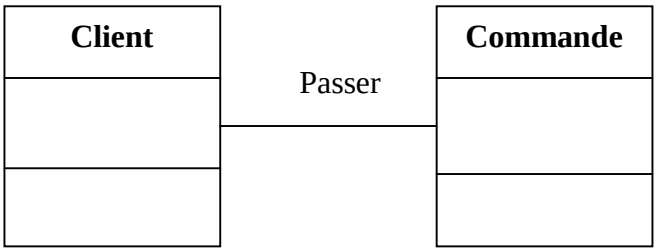

Figure 1.15 Association entre deux classes [Federic, 2001].

1. Arité des associations

On appelle arité d'une association le nombre de classes qui participent à l'association. Il y a deux types d'arités selon le nombre de classes associés à l'association. Il existe deux types d'association :

§ **Association binaire :** Une association binaire est matérialisée par un trait plein entre les classes associées Elle peut être ornée d'un nom, avec éventuellement une précision du sens de lecture .Quand les deux extrémités de l'association pointent vers la même classe, l'association est dite réflexive.

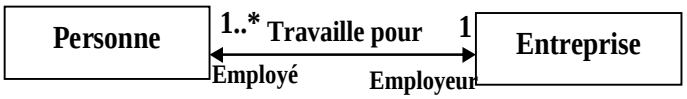

Figure 1.16 Une association binaire entre deux classes [Audibert, 2006].

§ **Association n-aire :** Une association n-aire lie plus de deux classes. Elle est représentée par un grand losange avec un chemin partant vers chaque classe participante. Le nom de l'association, le cas échéant, apparaît à proximité du losange.

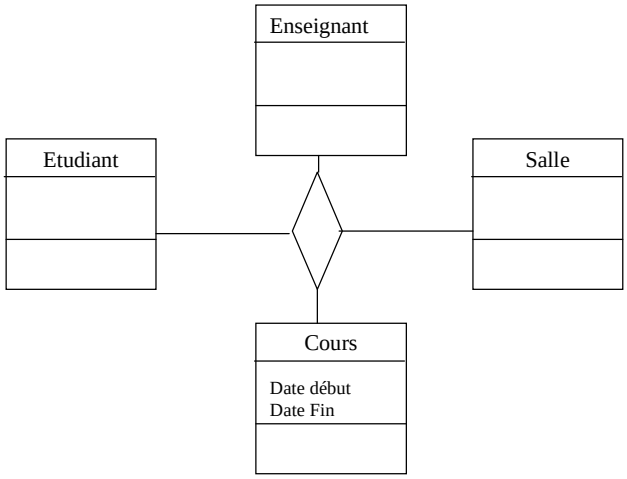

Figure 1.17 Une association d'arité égale à 4 [Federic, 2001].

2. Rôle d'une association

Les extrémités d'une association sont appelées rôles et peuvent porter un nom. Le rôle décrit comment une classe voit une autre classe au travers une association. Un rôle est nommé au moyen d'une forme nominale.

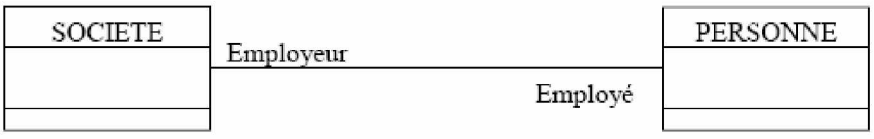

Figure 1.18 Identification d'un rôle d'association [Federic, 2001].

3. Multiplicité des associations

Chaque rôle peut porter une multiplicité montrant combien d'objets de la classe considérée (celle qui joue ce rôle) peuvent être liés à une instance de l'autre classe par l'association. La multiplicité est représentée sous la forme d'un couple de cardinalités [Federic, 2001]. Voici quelques exemples de multiplicité :

1..1 noté 1	Un et un seul
0..1	Zéro ou un
0..* noté *	De Zéro à n
1..*	De un à n
n..m	De n à m

Tab 1.2 Différents multiplicités d'une association [Audibert, 2006].

4. Types d'associations

§ **Classes associations** : il peut arriver que l'on ait besoin de garder des informations (attributs ou opérations) propres à une association. Une classe de ce type est appelée classe association. [Federic, 2001].Une classe association possède les propriétés des associations et des classes, elle se connecte à deux ou plusieurs classes et possède également des attributs et des opérations.

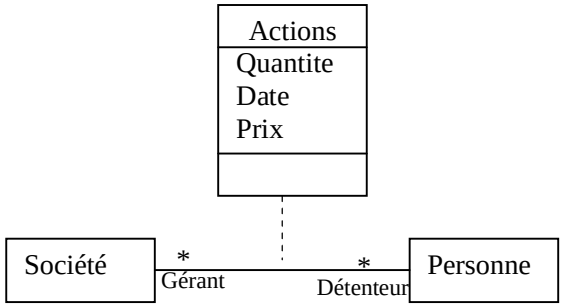

Figure 1.19 Exemple d'une classe association [Audibert, 2006].

§ **Agrégation :** Une agrégation est une association qui représente une relation d'inclusion structurelle ou comportementale d'un élément dans un ensemble. Graphiquement, on ajoute un losange vide du côté de l'agrégat [Audibert, 2006].

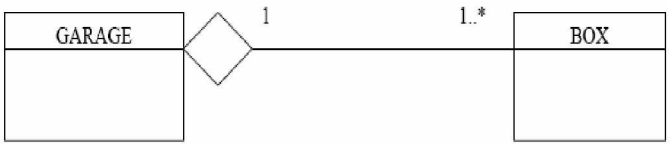

Figure 1.20 Exemple d'une classe association d'agrégation [Federic, 2001].

§ **Composition :** La composition, également appelée agrégation composite, décrit une contenance structurelle entre instances. Ainsi, la destruction de l'objet composite implique la destruction de ses composants. Une instance de la partie appartient toujours à au plus une instance de l'élément composite.

Graphiquement, on ajoute un losange plein du côté de l'agrégat [Audibert, 2006].

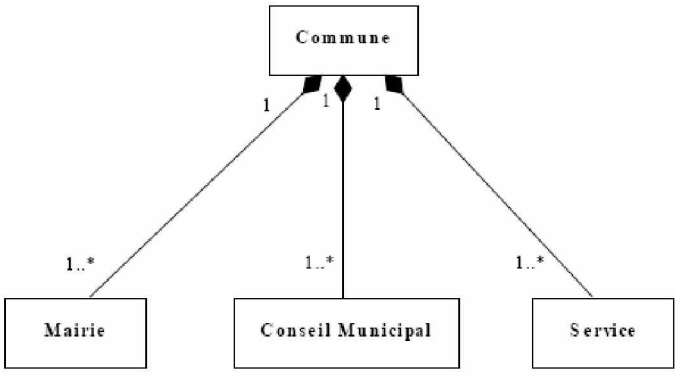

Figure 1.21 Exemple d'une classe association de composition [Federic, 2001].

§ **Généralisation** : L'association de généralisation est une relation de classification entre un élément plus général et un élément plus spécifique. La relation de généralisation signifie «est un» ou « est une sorte de ».

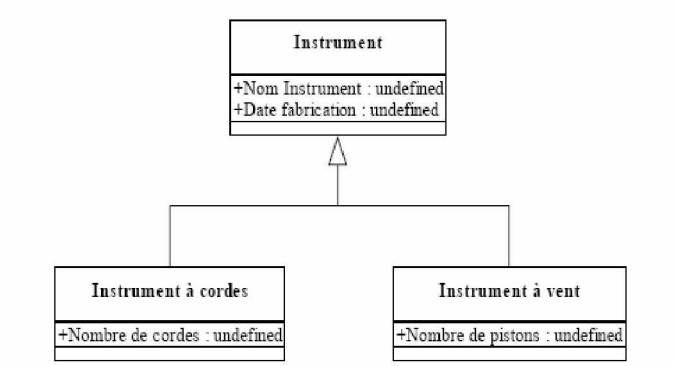

Figure 1.22 Exemple d'une classe association de composition [Federic, 2001].

Ø **Elaboration d'un diagramme de classes**

Pour élaborer un diagramme de classes, il y a au moins trois points de vue qui guident la modélisation:

§ Le point de vue spécification, qui met l'accent sur les interfaces des classes plutôt que sur leurs contenus.

§ Le point de vue conceptuel, qui capture les concepts du domaine et les liens qui les lient. Il s'intéresse à la manière éventuelle d'implémenter ces concepts et leurs relations et aux langages d'implémentation.

§ Le point de vue implémentation, qui est le plus courant, il détaille le contenu et l'implémentation de chaque classe.

Une démarche couramment utilisée pour bâtir un diagramme de classes consiste à :

1. Trouver les classes du domaine étudié**, c**ette étape empirique se fait généralement en collaboration avec un expert du domaine. Les classes correspondent généralement à des concepts ou des substantifs du domaine.

2. Trouver les associations entre classes**,** les associations correspondent souvent à des verbes, ou des constructions verbales, mettant en relation plusieurs classes, comme «est composé de», «pilote», «travaille pour».

3. Trouver les attributs des classes**.** les attributs correspondent souvent à des substantifs, ou des groupes nominaux, tels que «la masse d'une voiture» ou «le montant d'une transaction ». Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs.

4. Organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant la d'héritage [Federic, 2001].

1.2.9 Vérification des systèmes de production en vue de leur supervision

La vérification d'un système de production en vue de sa supervision est couverte par la connaissance d'un ensemble d'informations concernant les différentes fonctions du système ainsi que ses propriétés.

La vérification des systèmes de production repose sur l'existence d'un modèle qui modélise le système à vérifier. L'objectif est de vérifier ses propriétés. Les contraintes de sûreté de fonctionnement imposées aux systèmes de production conduisent à préconiser la mise en place de méthodes formelles pour la spécification, et la vérification des propriétés tels que: l'absence de blocage, vivacité, etc.

Le choix du formalisme de vérification est un compromis entre pouvoir d'expression et moyens de vérification existants.

Les formalismes les plus utilisés pour la vérification des systèmes de production sont : les modeles de checking, les automates d'états finis, et les réseaux de Petri, que nous les présentons dans la suite.

1.2.9.1 Modèle de Checking

Le modèle de checking est une technique de vérification formelle complètement automatique. Un modèle checker est un algorithme qui permet de vérifier si un modèle spécifié par un système d'états-transitions satisfait une propriété exprimée dans une logique temporelle.

Les systèmes de transitions désignent un ensemble de notations permettant de décrire des comportements dits « orientés contrôle ».Ces notations sont des états et des transitions entre les

états décrivant le comportement du système. Un exemple du modèlede checking est (la structures de Kripke, CTL) [Hammani, 2006].

Soit **P** un ensemble fini de propositions booléennes. Une structure de Kripke sur **P** est un quadruple **M = (S; T; I; L)** où :

§ **S** est l'ensemble des états .

§ **T** $\subseteq$ **S × S** est la relation de transition ; $\forall$ **s** $\in$ **S,** $\exists$ **s'** $\in$ **S; (s; s') $\in$ T**.

§ **I** $\subseteq$ **S** est l'ensemble des états initiaux.

§ **L : S** $\rightarrow 2^{P}$ est la fonction d'étiquetage [Deharbe, 2002].

Un exemple du modèle de Checking illustrant une partie d'une ligne de production est présenté dans la Figure 1.24 :

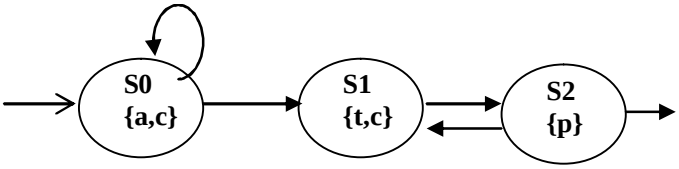

Figure 1.23 Exemple d'un modèle de Checking.

§ **P = {a, c, t, p}**, où: **a** : amener une pièce, **t** : travailler sur une pièce, **p** : passer une pièce a une autre machine, **c** : contrôler une pièce.

§ **S = {s0,s1,s2}**, **T = {(s_0, s_0); …….}**, **I = {s_0) et L : s_0 $\rightarrow$ {a}**….[Deharbe, 2002].

Les propriétés à vérifier sont exprimées dans la logique temporelle CTL.

Basant sur ce principe, le modèle de Checking permet de vérifier un ensemble de propriétés pour un système de production, suivant les étapes suivantes :

§ Modéliser le système de production par un modèle d'états-transitions.

§ Ecrire une spécification du système sous forme d'une logique temporelle.

§ Faire appel à un algorithme du modèle de Checker pour vérifier les propriétés du système et pour générer des scénarios qui mènent aux problèmes de fonctionnement ou les scénarios qui évitent ces problèmes [Garmhausen, 1998].

Cependant, cette méthode se heurte à la taille, souvent gigantesque, des graphes générés, et pose le problème d'explosion d'états dans le cas d'un système de production d'une grande taille. Ainsi ce formalisme ne permet pas de prendre en compte l'aspect temporel du fonctionnement du système.

1.2.9.2 Automates à états finis

Les automates à états finis ou machines à états finis, ont été employés pour spécifier le comportement dynamique de certains systèmes.

Les automates à états finis sont représentés graphiquement par des diagrammes d'états-transitions, des graphes orientés dont les noeuds sont des états et les arcs des transitions (Figure 1.24). Un état est un ensemble de valeurs qui caractérise le système à un moment donné dans le temps. Une transition d'état est une relation entre deux états indiquant un changement d'état possible, et qui peut être annotée pour indiquer les conditions et les sources de déclenchement (événements) et les opérations qui en résultent (sorties). On appelle automate fini le quintuplet **A = < Σ,Q,δ,I,F >** , où :

- § **Σ** est un alphabet.
- § **Q** est un ensemble d'états stables.
- § **I** est une partie de **Q** appelée ensemble des états initiaux.
- § **F** est une partie de **Q** appelée ensemble des états finaux.
- § **δ** est une partie de **Q×Σ ×Q** appelée ensemble des transitions. C'est une fonction. de transition qui à un état du système et un élément de l'alphabet associe le passage à un autre état [Pinzon, 1997].

Un exemple d'un automate à états finis illustrant une partie d'un système de production de pièces est présenté dans la Figure suivante. Il consiste simplement en deux marteaux (Pushers), qui ont deux degrés de prolongation demi-prolongé, et pleinement-prolongé. Quand le **Pusher 1** est entièrement prolongé, la pièce en position 1 s'est déplacée à la position 2. S'il y a un objet en position 2 et le **Pusher 2** se prolonge entièrement, alors la pièce se déplace à la position 3.

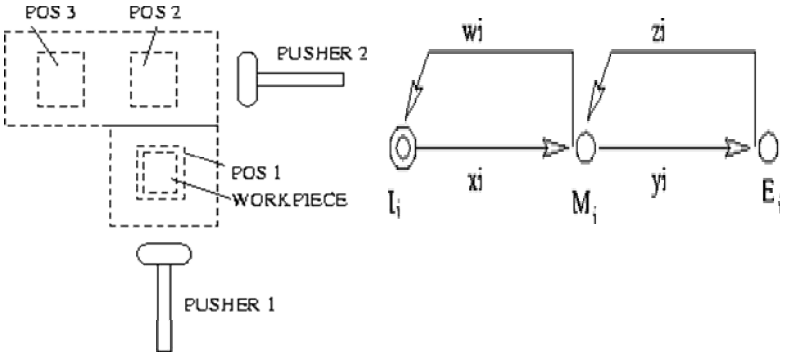

Figure 1.24 Exemple d'un automate a états finis [Pinzon, 1997].

Les états **Ii**, **Mi**, et **Ei**, indiquent les positions du Pusher : initial, demi-prolongé, et pleinement-prolongé respectivement, l 'état **Ii** est un état initial et finale au même temps.

L'ensemble des événements est **Σ= {xi,yi,wi : i=1,2}.**

Comme le modèle de Checking, les automates à états finis mènent rapidement à une explosion du nombre d'états et de transitions suivant la taille du système de production.

Ainsi, la complexité de vérification des systèmes de production par ce formalisme vient de:

§ La coopération : les automates décrivent des processus ayant un but commun.

§ La compétition : les automates partagent des ressources

§ Le pseudo parallélisme ou l'entrelacement (interleaving) : les événements sont tous ordonnés par un ordre total [Valette, 1999].

1.2.9.3 Réseaux de Petri

Les réseaux de Pétri sont un outil efficace pour modéliser, et vérifier les systèmes de production. Ils peuvent manipuler les problèmes qui ne peuvent pas être modélisé par les autres formalismes.

Contrairement aux formalismes précédents, Les réseaux de Pétri sont bien adaptés pour modéliser des systèmes de production parce qu'ils capturent les relations et les interactions entre les événements. En outre, une base mathématique solide existe pour l'analyse des propriétés de système telles que le blocage, le parallélisme et les conflits…etc.

a. Définition

Un réseau de Petri (RdPs) est un graphe biparti orienté qui a deux types de nœuds : les places (notées graphiquement par des cercles) et les transitions (notées graphiquement par un rectangle ou une barre) ; reliées par des arcs (qui sont des flèches).Un arc est un lien soit d'une place à une transition, soit d'une transition à une place.

Chaque arc est étiqueté par une valeur (ou un poids), qui est un nombre entier positif. L'arc ayant **k** poids peut être interprété comme un ensemble de **k** arcs parallèles. L'étiquette du poids égale à **1** est ignorée.

Formellement, un réseau de Petri est un graphe orienté biparti valué **< P, T, Pré, Post >** avec :

1. **P** est l'ensemble des places.
2. **T** est l'ensemble des transitions.
3. **Pré** = **P** x **T**→**N**, est une application d'incidence avant où : **Pré (p, t)** contient la valeur entière « **n** » associée à l'arc allant de « **p** » à « **t** ».

4. **Post** = **P** x **T** → **N**, une application d'incidence arrière où : **Post (p, t)** contient la valeur entière « **n** » associée à l'arc allant de « **t** » à « **p** ».

Remarques :

1- « **p** » est une place d'entrée de la transition « **t** » si **Pré (p, t) > 0.**

2- « **p** » est une place de sortie de la transition « **t** » si **Post (p, t) > 0** [Valette ,2002].

b. Modélisation par les RdPs

Un réseau de Petri est un outil pour modéliser une classe spécifique de problèmes qui sont les systèmes contenant des évènements concurrents ou parallèles. Il modélise en particulier deux aspects dans un système, les conditions et les événements et la relation entre les deux.

Dans ce cadre, dans un système à n'importe quel moment, certaines pré-conditions (une description logique d'un état du système nécessaires au déclenchement d'un événement) devenues vrai, cela va causer le déclenchement de certains événements qui vont changer l'état du système. Lorsqu'un événement se produit, certaines de ses pré-conditions peuvent cesser d'être vraies alors que d'autres post-conditions de l'événement deviennent vraies. [Peterson, 1999].

L'exemple de la Figure 1.25 présente un lecteur de cartes

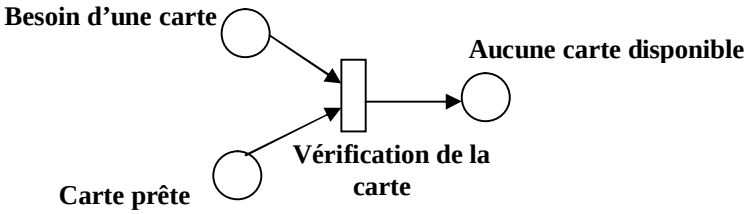

Figure 1.25 Un modèle simple de trois conditions et un événement [Peterson, 1999].

Où il y a les pré-conditions suivantes :

§ Besoin d'une carte pour la lecture

§ Carte Prête.

La pré-condition **''**Carte Prête'' va permettre l'apparition de l'événement '' Vérification d'une carte''. L'occurrence de cet évènement va mettre les deux pré-conditions précédentes à faux et rendre la post-condition '' Aucune carte disponible'' vrai.

Comme il est montré dans la Figure 1.25, les places sont utilisées pour représenter les conditions et les transitions pour représenter les événements [Peterson, 1999].

c. Marquage d'un RdP

Un réseau de Petri marqué est un couple :

N=<R, M_o > Où : **R** est un RdP, **M_o** est le marquage initial du réseau R.

Le marquage d'un RdP est une application **M : P → N** donnant pour chaque place le nombre de jetons qu'elle contient [Valette ,2002].

d. Evolution d'un RdP

L'évolution d'un RdP correspond à l'évolution de son marquage au cours du temps (évolution de l'état du système) : il se traduit par un déplacement des jetons pour une transition **t** de l'ensemble des places d'entrée vers l'ensemble des places de sortie de cette transition. Ce déplacement s'effectue par le franchissement de la transition **t** [Scorletti, 2006].

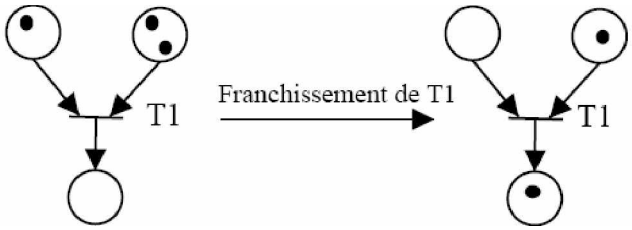

Figure 1.26 Franchissement de la transition T1 [Scorletti, 2006].

Pour un marquage **M**, une transition **t** est dite tirable (ou franchissable ou sensibilisée) si et seulement si:

$\forall$ pi$\in$ P , on a M(pi) >= Pré (pi, t)

C'est-à-dire, pour toutes les places d'entrées **pi** de **t**, le nombre de jetons dans **pi**, **M(pi)**, est supérieur ou égal au poids de l'arc allant de **pi** à **t**.

Le franchissement de **t** depuis **M** donnant le nouveau marquage **M'**, se notera : **M[t>M'** [Scorletti , 2006].

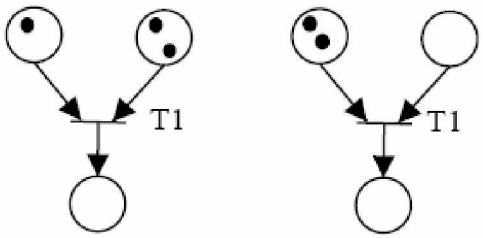

Figure 1.27 La Transition T1 (gauche) est franchissable, T1 (droite) est non franchissable [Scorletti , 2006].

L'évolution d'un RdP se fait par le franchissement d'une seule transition à la fois. Quand plusieurs transitions sont simultanément franchissables, on ne peut pas savoir dans quel ordre elles seront effectivement franchies. L'évolution n'est donc pas unique [Scorletti , 2006].

e. Structures fondamentales pour la modélisation des systèmes

Les RdPs permettent de modéliser un certain nombre de comportements importants dans les systèmes tels que le parallélisme, la synchronisation, le partage de ressources, la mémorisation et la lecture d'information, la limitation d'une capacité de stockage. [Scorletti , 2006].

1. Parallélisme

Le parallélisme représente la possibilité que plusieurs processus évoluent simultanément au sein du même système. On peut provoquer le départ simultané de l'évolution de deux processus à l'aide d'une transition ayant plusieurs places de sortie.

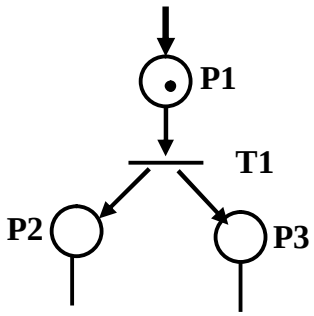

Figure 1.28 Structure du parallélisme [Scorletti, 2006].

Le franchissement de la transition **T1** met une marque dans la place **P2** (ce qui marque le déclenchement du processus 1) et une marque dans la place **P3** (ce qui marque le déclenchement du processus 2) [Scorletti , 2006].

2. Synchronisation

La synchronisation est modélisée sous cinq formes :

§ **Synchronisation mutuelle (Par rendez vous)**

La synchronisation mutuelle ou par rendez-vous permet de synchroniser les opérations de deux processus. La Figure 1.29 montre un exemple de deux processus, le franchissement de la transition **T7** ne peut se faire que si la place **P12** du **processus 1** et la place **P6** du **processus 2** contiennent chacune au moins un jeton. Si ce n'est pas le cas, par exemple la place **P12** ne contient pas de jetons, le **processus 2** est bloqué sur la place **P6** : il attend que l'évolution du **processus 1** soit telle qu'au moins un jeton apparaisse dans la place **P12**.

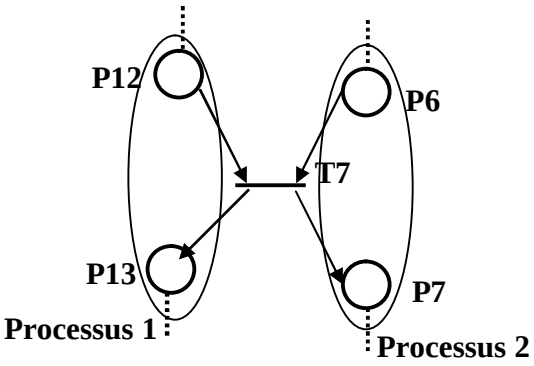

Figure 1.29 Synchronisation mutuelle [Scorletti, 2006].

§ **Synchronisation par signal (sémaphore)**

Dans la synchronisation par signal Les opérations du **processus 2** ne peuvent se poursuivre que si le **processus 1** a atteint un certain niveau dans la suite de ses opérations. Par contre, L'avancement des opérations du **processus 1** ne dépend pas de l'avancement des opérations du **processus 2**.

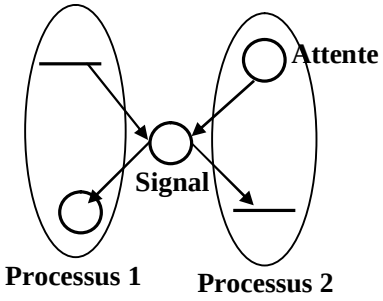

Figure 1.30 Synchronisation par sémaphore [Scorletti , 2006].

Si la place **Signal** est marquée et la place **Attente** ne l'est pas, cela signifie que le processus **P1** a envoyé le signal mais le processus **P2** ne l'a pas encore reçu. Si, par contre, la place **Signal** n'est pas marquée et que la place **Attente** est marquée, cela signifie que le processus **P2** est en attente du signal [Scorletti , 2006].

§ **Synchronisation par Partage de ressources**

Cette structure va modéliser le fait qu'au sein du même système plusieurs processus partagent une même ressource [Scorletti, 2006], en utilisant le principe de l'exclusion mutuelle.

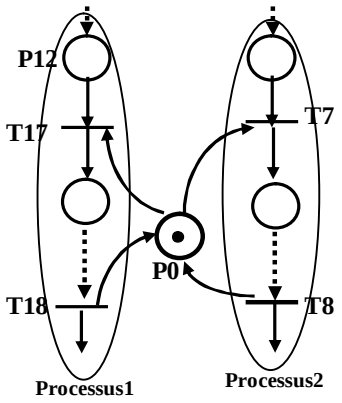

Figure 1.31 Synchronisation par Partage de ressources [Scorletti, 2006],

La marque dans la place **P0** présente une ressource mise en commun entre le **processus 1** et le **processus 2**. Le franchissement de la transition **T17** lors de l'évolution du **processus 1** entraîne la consommation du jeton présenté dans la place **P0.** La ressource que constitue cette marque n'est alors plus disponible pour l´évolution du **processus 2.** Lorsque la transition **T18** est franchie, un jeton est alors est placé dans la place **P0** : la ressource devient alors disponible pour l'évolution des deux processus.

§ **Synchronisation par Mémorisation**

Dans tous les modèles, on peut compter le nombre de tirs d'une transition en utilisant une place sans sortie [Scorletti , 2006].

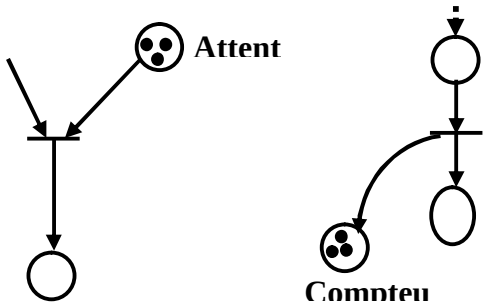

Figure 1.32 Synchronisation par Mémorisation [Scorletti, 2006].

Dans La Figure 1.32 les places **attente** et **Compteur** peuvent servir à indiquer combien d'instances d'un processus sont en attente.

§ **Capacité limitée**

Dans la Figure 1.33, pour que la transition **T3** soit franchissable, il est nécessaire que la place **P5** contienne des jetons. Le marquage de **P5** ne permet que deux franchissements successifs de **T3**. La transition **T3** sera à nouveau franchissable si le franchissement de la

transition **T4** permet de mettre des jetons dans la place **P5.** Au total, la place **P4** ne pourra pas contenir plus de 3 jetons [Scorletti , 2006].

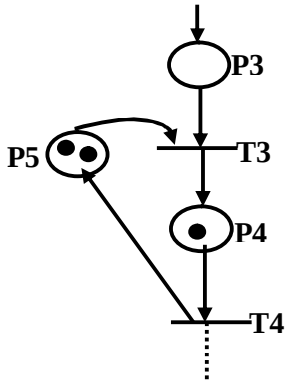

Figure 1.33 Synchronisation par Capacité limité [Scorletti , 2006].

f. Propriétés qualitatives des RdP

Après la modélisation d'un système, la question qui se pose qu'est qu'on peut faire avec ce modèle ? Les réseaux de Petri ont une grande capacité d'analyse des propriétés et des problèmes associés avec les systèmes concurrents .Deux types de propriétés peuvent être étudiées :

- § Les propriétés qui dépendent du marquage initial qui sont les propriétés comportementales.
- § Les propriétés qui dépendent de la structure du réseau qui sont les propriétés structurelles [Murata ,1989].

Nous nous intéressons ici aux propriétés comportementales

1. Propriétés Comportementales d'un RdP

§ **Accessibilité**

L'accessibilité est une propriété fondamentale pour étudier les propriétés dynamiques du système. Le franchissement d'une transition va changer la distribution des jetons sur le réseau, partant d'un marquage **M** et aboutissant à un autre marquage **M'**.

Un marquage $\mathbf{M_1}$ est dit accessible à partir d'un marquage $\mathbf{M_0}$ s'il existe une séquence de franchissement qui transforme $\mathbf{M_0}$ vers $\mathbf{M_1}$.

L'ensemble des marquages accessibles depuis $\mathbf{M_0}$ par une séquence de franchissement est noté par **A (N, $\mathbf{M_0}$)** ou **A ($\mathbf{M_0}$).**

L'ensemble de tous les marquages possibles accessibles depuis $\mathbf{M_0}$ est noté par **L (N,**

M_0) ou **L (M_0)** [Murata ,1989].

Le problème concernant l'accessibilité pour le réseau de Petri est de trouver si **M_n** $\in$ **A(M_0)** en donnant **M_n** et un réseau **(N , M_0)** [Vinh Duc,2005].

§ **Bornitude**

Un RdP **(N, M_0)** est K-borné ou simplement borné si le nombre de jetons dans chaque place ne dépasse pas un nombre fini **K** pour tout marquage accessible de puis **M_0** [Murata ,1989].

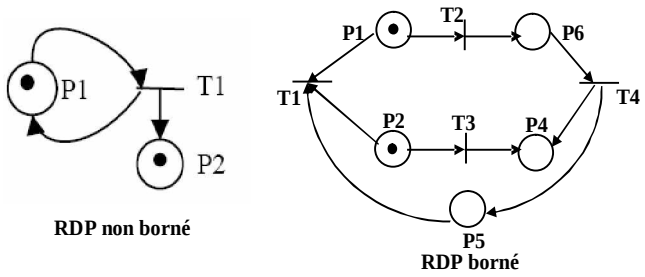

Figure 1.34 Exemple d'un RdP non borné et borné [Scorletti , 2006][Murata ,1989].

Pour le RdP non borné, la transition **T1** admet la place **P1** comme l'unique place d'entrée. La place **P1** a un jeton, la transition **T1** est franchissable. A chaque franchissement de **T1** s'ajoute un jeton dans la place **P2**, donc le marquage de celle-ci peut donc tendre vers l'infini [Scorletti , 2006].

§ **Vivacité et blocage**

La notion de vivacité est liée à l'absence d'interblocage dans un système opérationnel. Une transition **T_j** est vivante pour un marquage initial **M_0** si pour tout marquage accessible **M_k**, il existe une séquence de franchissements à partir de **M_k** contenant **T_j**.

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

Un réseau **(N, M0)** est vivant si toutes ses transitions sont vivantes. [Scorletti, 2006]. La vivacité exprime le fait qu'à tout instant de l'évolution du système, le franchissement à terme d'une transition vivante **T** n'est jamais définitivement impossible [Vinh Duc,2005].

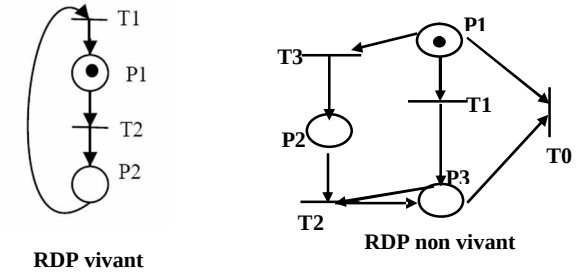

Figure 1.35 Exemple d'un RdP vivant et non vivant [Scorletti , 2006][Murata ,1989].

Un blocage (ou état puits) est un marquage pour lequel aucune transition n'est validée. Un RdP marqué est dit sans blocage pour un marquage initial $\mathbf{M_0}$ si aucun marquage accessible n'est un blocage. Le RdP marqué de la Figure 1.36 a pour blocage le marquage M3= [1, 0, 0,4].

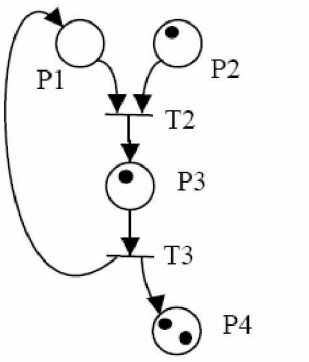

Figure 1.36 Exemple d'un RdP avec un blocage [Scorletti , 2006].

§ **Réseau de Petri Réinitialisable et Home d'etat**

Un réseau de Petri est réinitialisable si pour chaque marquage **M** dans $\mathbf{R(M_0)}$, $\mathbf{M_0}$ est accessible à partir de **M**.

Un marquage $\mathbf{M_0}$ est Home d'état si pour chaque marquage **M** dans $\mathbf{A\ (M_0)}$, **M** est accessible à partir de $\mathbf{M_0}$ [Vinh Duc, 2005].

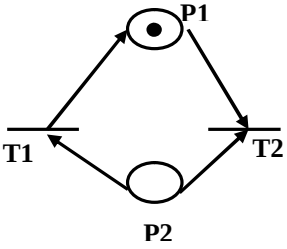

Figure 1.37 Exemple d'un RdP réinitialisable [Murata ,1989].

Un marquage $\mathbf{M_0}$ est Home d'état si pour chaque marquage **M** dans $\mathbf{A(M_0)}$, **M** est à partir de $\mathbf{M_0}$.

§ **Couverture**

Un marquage **M** dans un RdP $\mathbf{(N,M_0)}$ est dit couvré s'il existe un marquage **M'** dans $\mathbf{R(M_0)}$ telque $\mathbf{M'(p) \geq M(p)}$ pour chaque place du réseau [Murata ,1989].

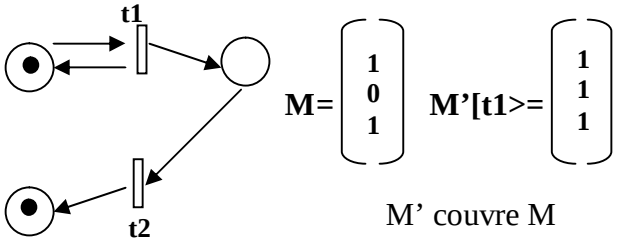

Figure 1.38 Exemple d'un RdP couvré [Murata ,1989].

§ **Persistance**

Un RdP **(N, M_0)** est dit persistant pour n'importe quel deux transitions, si le franchissement d'une transition ne doit pas inhiber les autres. Une transition dans un RdP persistant une fois elle est validée, elle doit rester validée jusqu'à son franchissement [Murata ,1989].

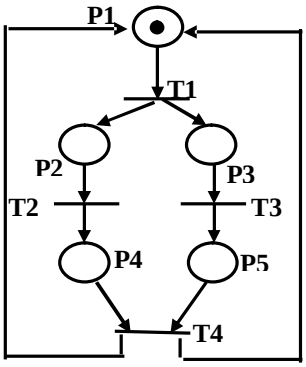

Figure 1.39 Exemple d'un RdP persistant [Murata ,1989].

g. Méthodes d'analyse

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

§ Méthode d'analyse par graphe de marquage ou de l'arborescence de couverture. Une arborescence est un graphe particulier composé d'arcs orientés qui divergent progressivement à partir d'un noeud appelé racine de l'arborescence.

§ Méthode d'analyse par algèbre linéaire ou équation de matrices. L'algèbre linéaire rend possible l'étude de propriétés structurelles, en particulier par la recherche des invariants linéaires de places (composantes conservatives) et des invariants linéaires de transitions (composantes répétitives) et donne aussi des résultats dépendant du marquage. On peut en déduire si le réseau est borné ou sauf, sans blocage et s'il est réinitialisable ou vivant.

§ Méthode d'analyse par Réduction de RdP en appliquant certaines règles de réduction particulières.

Nous nous intéressons ici à la méthode d'analyse par arbre de marquage, qui consiste à construire le graphe de tous les marquages du réseau et déduire les propriétés grâce aux techniques de la théorie de graphes.

1.Graphe de marquages

Le graphe des marquages accessibles **G(N, Mo)** est défini comme le graphe dont les nœuds sont les marquages accessibles de **A(N, Mo)** et dont les arcs sont les noms des transitions impliquées dans les franchissement menant d'un marquage à un autre [Vinh Duc,2005].

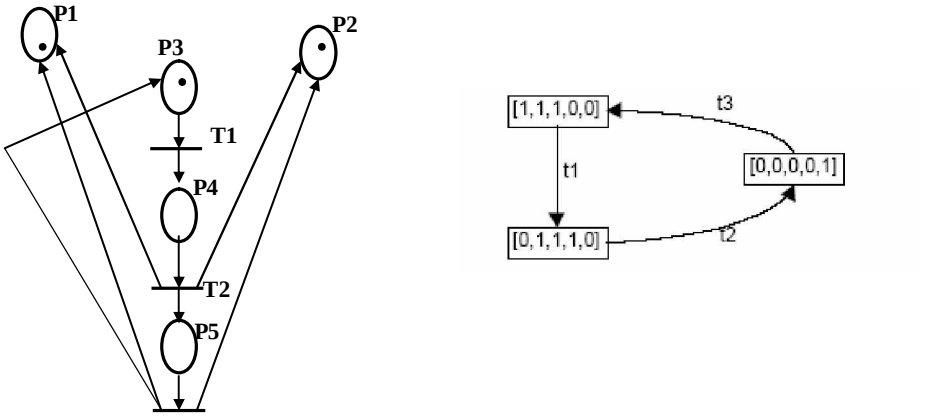

Figure 1.40 Exemple d'un graphe de marquage [Vinh Duc, 2005].

Le graphe des marquages permet d'explorer tous les marquages possibles dans un RdP. Si le graphe de marquages accessibles est fini, c'est la situation la plus favorable car toutes les propriétés peuvent être déduites simplement. L'alternative, si le réseau n'est pas borné, le graphe de marquages sera infini et on aura le problème d'explosion d'espace d'états.

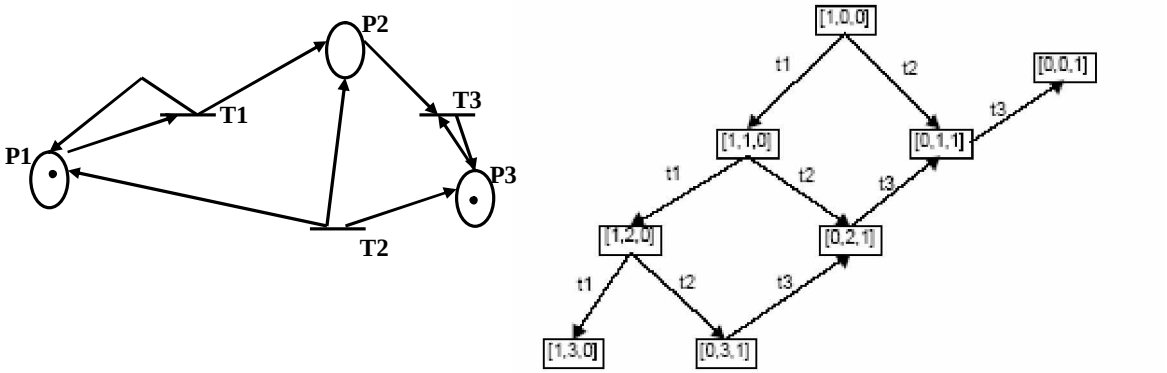

Figure 1.41 Exemple d'un graphe de marquage infini [Vinh Duc, 2005].

La solution pour ce problème consiste à construire le graphe de couverture. On doit accepter la perte des informations. Pour l'arbre infini, le nombre de jetons dans les places pouvant avoir une quantité arbitrairement grande de jetons, elle va être considérée comme une valeur particulière, notée par **ω**. Elle a les propriétés suivantes pour chaque entier **n.**

ω+ n = ω

ω - n= ω

n <= ω

ω <= ω

L'algorithme pour construire le graphe de couverture consiste à :

1. Assigner la racine à $\mathbf{M_0}$ et mettre **tag($\mathbf{M_0}$)** = **"nouveau"** ;

2. Tant qu'il existe un marquage ayant **tag = "nouveau",** nous faisons les étapes suivantes :

2.1 Choisir un **"nouveau"** marquage **M** ;

2.2 Si **M** est égal à un autre noeud dans le chemin de noeud à **M**, donc **tag(M)** = **"old"** et on continue (l'étape 2) avec un autre nouveau marquage ;

2.3 S'il n'y a pas de transition franchissable dans **M**, donc on assigne **tag(M)** = **"terminal"** ;

2.4 Tant qu'il existe une transition franchissable **t** dans **M**, on fait les étapes suivantes :

2.4.1 Calculer **M'** tel que **M[t > M'** ;

2.4.2 S'il existe un noeud **M'** dans le chemin de la racine à **M'** tel que :

$$\forall p \in P : M'(p) \geqslant M\ (p) \text{ et } M \neq M'$$

Càd **M < M'**, donc pour chaque **p** tel que **M'(p) > M (p)** on replace **M'(p)** par **ω.**

2.4.3 Ajouter le noeud **M'** à l'arbre et dessiner un arc de **M** à **M'** avec l'étiquette t, et on assigne **tag(M')** = **"nouveau"**.

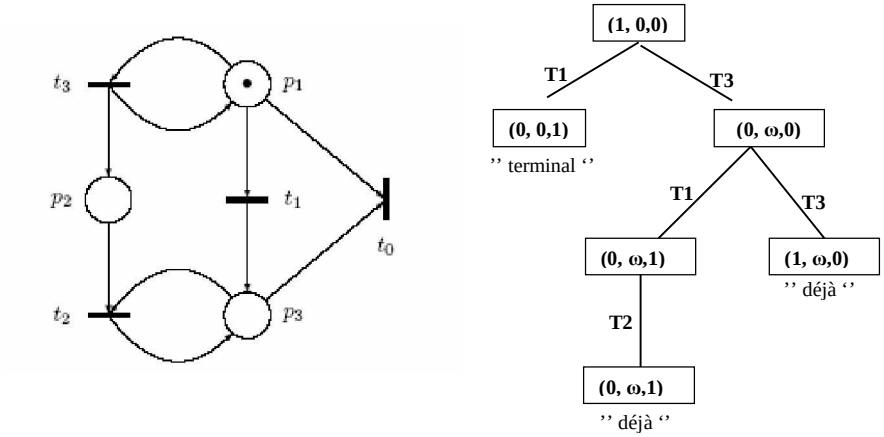

Figure 1.42 Exemple d'un graphe de couverture [Vinh Duc, 2005].

1.4 Conclusion

Dans ce chapitre nous avons présenté les concepts et les principes fondamentaux des systèmes à événements discrets et plus particulièrement les systèmes de production.

La supervision de ces systèmes, repose sur l'existence d'un modèle qui modélise la partie opérationnelle du système. Pour cela nous avons abordé les différentes méthodes de modélisation et nous avons mis l'accent sur UML.

Ainsi la vérification de certaines propriétés des systèmes de production fait une partie importante pour leur supervision, pour cette raison, nous avons présenté les différents formalismes apportés pour ce but et les réseaux de Petri sont le formalisme le plus adapté pour ce type de systèmes.

Dans le prochain chapitre notre intérêt sera sur la proposition d'une approche de modélisation d'un processus de production, utilisant UML avec un passage vers les réseaux de Petri, pour un but de vérification. Toute la procédure est en vue de concevoir un superviseur de contrôle pour ce processus.