6220S

50376
1992
8

N°d'ordre : 860

50376
1992
8

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE

en

PRODUCTIQUE,
AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

par

Agnès HEBRARD
Maître en INFORMATIQUE

BIBLIOTHEQUE INTERUNIVERSITAIRE
SECTION DE SCIENCES
LILLE

ETUDE ET REALISATION D'UN MODULE DE PILOTAGE ET D'UN OUTIL DE REPRESENTATION GRAPHIQUE APPLIQUES A LA CONCEPTION D'UN SYSTEME DE PRODUCTION FLEXIBLE

Soutenue le 7 février 1992 devant la Commission d'Examen

Membres du jury :

R. VALETTE	**Rapporteur**
C. VASSEUR	**Rapporteur**
J.P. BOUREY	**Examinateur**
E. CRAYE	**Examinateur**
P. DUMORTIER	**Examinateur**
J.C. GENTINA	**Examinateur, Directeur de Thèse**

Directeurs de travail : J.P. BOUREY, J.C. GENTINA

Avant - Propos

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique et d'Informatique Industrielle de l'Ecole Centrale de Lille sous la direction scientifique de Monsieur GENTINA, Directeur de l'EC Lille. Je tiens à le remercier vivement de m'avoir accueillie dans son équipe de recherche.

Je suis très reconnaissante à Monsieur Valette, Directeur de Recherche au LAAS-CNRS, et à Monsieur Vasseur, Professeur à l'Université des sciences et techniques de Lille I, Directeur de l'ENSAIT, de l'honneur qu'ils me font en acceptant de juger ce travail et d'être les rapporteurs de cette thèse.

Je tiens également à remercier Monsieur Bourey, Maître de Conférences à l'EC Lille, et Monsieur Craye, Maître de Conférences à l'EC Lille, pour l'honneur qu'ils me font en participant à mon jury de thèse.

Je suis en outre très heureuse de la présence à ce jury de Monsieur Dumortier, Ingénieur de Recherche à la Division Recherches et Développements de la Télémécanique.

Je voudrais ici remercier tous les membres du Laboratoire d'Automatique et d'Informatique Industrielle de l'EC Lille, et en particulier Messieurs Slimane AMAR, Jean-Pierre BOUREY, Emmanuel CASTELAIN, Etienne CRAYE, Christian AUSFELDER, Samir ELKHATTABI, Armand TOGUYENI, Pascal YIM, pour le soutien et l'aide précisieuse qu'ils m'ont apportés sur le plan scientifique et sur le plan humain.

Je remercie aussi très sincèrement les personnes qui ont assuré la reprographie de ce mémoire : Messieurs VANGREVENINGE et GERMON.

Enfin, je tiens à remercier très vivement mon mari, pour la grande patience dont il a fait preuve pendant la période de rédaction de ce mémoire.

Sommaire

Sommaire

Introduction générale. 25
Présentation de CASPAIM2. 33
Module de pilotage. 81
Interface graphique. 117
Conclusion générale. 185
Bibliographie. 191
Glossaire. 209
Annexe. 219

Chapitre I : Présentation de CASPAIM2.

Introduction. 35

I. Approche globale. 37
I.1. Introduction. 37
I.2. Les notions. 37
I.3. Conclusion. 41

II. CASPAIM1. 43
II.1. Introduction. 43
II.2. Les modèles. 43
II.2.1. Introduction. 43
II.2.2. La Partie Commande. 44
II.2.3. Le Niveau Hiérarchique. 47
II.2.4. Le Procédé. 48
II.2.5. Conclusion. 49
II.3. La démarche de conception. 50
II.3.1. Introduction. 50
II.3.2. Exemple. 51
II.3.3. La phase de spécification. 52
II.3.4. La phase de modélisation. 53
II.3.5. La phase de simulation. 54
II.3.6. La phase d'implantation. 55
II.3.7. Conclusion. 55
II.4. Les limites de la démarche. 56
II.5. Conclusion 59

III. CASPAIM2. 61
III.1. Introduction. 61
III.2. Exemple. 62
III.3. Méthode générale. 63
III.4. Démarche. 65
III.4.1. Introduction 65
III.4.2. La phase de spécification. 66
III.4.3. La phase de modélisation de la commande. 71
III.4.4. La modélisation de la commande effective et du comportement des ressources. 74
III.4.5. La phase de simulation. 75
III.5. Conclusion. 76

Conclusion. 79

Chapitre II : Module de pilotage.

Introduction. **83**

I. Généralités. **85**
I.1. Introduction **85**
I.2. La gestion de production **85**
I.2.1. Introduction. 85
I.2.2. La planification 86
I.2.3. L'ordonnancement. 87
I.2.4. La gestion de stocks 88
I.2.5. Le pilotage 88
I.2.6. Conclusion 89
I.3. La gestion de production dans CASPAIM2 ..... **91**
I.4. Conclusion **93**

II. Le module de pilotage. **95**
II.1. Introduction. **95**
II.2. La classification des indéterminismes. **96**
II.3. La résolution des indéterminismes. **102**
II.3.1. Introduction. 102
II.3.2. Les décisions dues à la flexibilité produit. 104
II.3.3. Les décisions face aux conflits d'accès. 105
II.3.4. Les décisions dues à la flexibilité du système de transport. 106
II.4. Conclusion. **113**

Conclusion. **115**

Chapitre III : Interface graphique.

Introduction. **119**

I. Présentation générale. **121**
I.1. Introduction. **121**
I.2. Intérêt d'une interface graphique. **121**
I.3. Apports d'une interface graphique. **122**
I.4. Structure générale d'une interface graphique. **123**
I.5. Conclusion. **127**

II. Modèles de l'interface graphique. **129**
II.1. Introduction. **129**
II.2. Formalisme de description des actions du prégraphe. **129**
II.2.1. Les actions de transformation. 130
II.2.2. Les actions positionnelles. 130
II.2.3. Les actions informationnelles. 131
II.3. Structures de données. **132**
II.3.1. Présentation générale. 132
II.3.2. Structure de données de l'application. 133
II.3.3. Structure de données de description 134
II.3.4. Fichier graphique. 138
II.4. Implantation de l'interface graphique. **141**
II.4.1. Présentation générale. 141
II.4.2. Choix d'implantation. 145
II.5. Conclusion. **146**

III. Méthode de représentation graphique. 147
III.1. Introduction. 147
III.2. Exemple explicatif. 147
III.3. Organisation du logiciel. 149
III.4. Edition et rappel d'un graphe. 153
III.5. Méthode de positionnement automatique. 155
III.5.1. Type de graphes. 155
III.5.2. Règles de dessin. 157
III.5.3. Termes de vocabulaire. 159
III.5.4. Analyse du graphe. 160
III.5.5. Structure de base. 161
III.5.6. Méthode. 164
III.5.7. Maille hexagonale. 170
III.6. Conclusion. 173

IV. Développements de l'interface graphique. 175
IV.1. Les utilisations de l'interface graphique. 175
IV.2. Le tracé des gammes logiques et opératoires. 177
IV.3. Conclusion. 182

Conclusion. 183

Annexe

I.1. Graphes de processus. 221
I.2. Graphes de liaisons. 222
I.3. Modèles du groupement moyens de transport..... 223
I.4. Modèle du robot R2..... 225
I.5. Modèles du convoyeur..... 226
I.6. Modèle comportemental du convoyeur..... 228
I.7. Graphe récapitulatif des différentes connexions entre modèles..... 229

II.1. Listing du programme de calcul du coût dynamique... 231
II.2. Modèle comportemental général..... 239

III.1. Fichier décrivant un prégraphe pour l'exemple de la cellule de l'Ecole Centrale de Lille. 241
III.2. Arborescence des objets prédéfinis d'Aïda. 244
III.3. Exemple de l'environnement d'Aïda..... 245
III.4. Listing du programme d'analyse des graphes. 246
III.5. Algorithme de la méthode de positionnement. automatique. 250
III.6. Premier exemple..... 252
III.7. Deuxième exemple..... 258
III.8. Tableau de calcul du voisinage des 25 places d'un circuit pour la maille hexagonale. 263

Introduction générale

Introduction générale.

Partout où cela est possible, on tend à éliminer la grande usine, rigide, conçue comme si elle devait durer toujours. En effet, depuis quelques années, les entreprises ont à faire face à un nouvel environnement :

- concurrence internationale accrue,
- exigences de la clientèle
  - sur la qualité du produit qui doit suivre des normes sévères et ne doit comporter aucun défaut,
  - sur la variété des produits, ce qui influe sur la durée de vie de plus en plus courte,
- augmentation des coûts des matières premières et de l'énergie, qui est la conséquence de l'épuisement des gisements facilement accessibles, de l'augmentation des frais de transport et des frais de stockage.

Elles doivent ainsi, modifier leur stratégie en conséquence pour répondre aux désirs des clients, pour abaisser les coûts de fabrication et pour améliorer les conditions de travail.

Face à ces contraintes, cette nouvelle stratégie vise d'une part les produits eux-mêmes, ayant pour but, d'augmenter leur différenciation et d'améliorer leurs performances, leur qualité et leur fiabilité, et d'autre part, les moyens de les obtenir, c'est-à-dire la production. Les mesures prises, au niveau de la production, pour répondre à ces contraintes, visent à :

- l'amélioration de la productivité,
- la réduction du cycle de fabrication,
- l'accroissement du taux d'engagement des machines,
- la réduction des rebuts,
- l'accroissement de la souplesse de fabrication,
- la réduction des stocks,
- l'optimisation de l'emploi des matières premières,
- le respect des délais.

En parallèle à l'étude de nouvelles stratégies de production, la technologie a connu des développements spectaculaires, pendant la dernière décennie, liés pour beaucoup aux progrès de l'informatique et des automatismes. C'est ainsi, que se sont multipliés les machines à commande numérique, les robots, ...

L'exploitation de ces nouvelles possibilités dans l'atelier de fabrication, jusqu'à présent, ne comprend que la **juxtaposition** des équipements divers de production, de manutention, et de stockage, avec peu d'interface entre eux.

Pour exploiter pleinement ces nouvelles possibilités, il s'agit maintenant, de réaliser des ateliers très intégrés où sont intimement liés :

- les équipements de production et de contrôle,
- les moyens de manutention,
- les matériels et logiciels de pilotage et d'automatisation,

et de réussir le mariage de deux objectifs a priori contradictoires : l'automatisation et la flexibilité.

Ce mariage doit se retrouver :

- au niveau des machines, par la diversité des opérations réalisées automatiquement sur chaque poste de travail dans une classe d'opération donnée,
- au niveau de l'enchaînement des différents postes par un convoyage non restrictif autorisant toutes les liaisons possibles,
- au niveau du système informatique de conduite, par sa capacité à régler les situations conflictuelles dans l'atelier en respectant les divers objectifs fixés.

La conception d'un atelier flexible de production est une tâche complexe, où les choix sont nombreux et doivent être faits dans des délais courts, par des intervenants ayant des préoccupations complémentaires et parfois contradictoires. Trop souvent, l'empirisme et l'improvisation président à la conception d'un atelier.

L'objectif du Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL), de l'Ecole Centrale de Lille, est de contribuer à l'élaboration et au développement d'une méthodologie globale, d'assistance dans la démarche de conception de systèmes de production flexibles depuis la spécification jusqu'à l'implantation finale sur site. Cette démarche constitue le projet CASPAIM (Conception Assistée de Systèmes de Production Automatisés en Industrie Manufacturière).

Le but ultime du projet est de permettre à l'utilisateur de définir progressivement les spécifications au bon niveau, dans un langage clair et accessible, pendant l'élaboration la plus transparente possible d'une solution.

Le projet CASPAIM s'inscrit dans le cadre ambitieux de gestion complète du cycle de vie d'un automatisme : conception - validation - exploitation - maintenance.

Il est le résultat du travail d'une équipe, dont plusieurs composantes ont déjà fait l'objet de plusieurs mémoires de thèses :

- élaboration d'un premier modèle [KAP 88], qui intègre les résultats de travaux effectués en amont de la démarche, notamment des méthodes d'élaboration de gammes d'assemblage [BOU 84], [BOU 87], [CAM 89], avec de plus la prise en compte des moyens de production,
- à partir de ce modèle, la deuxième étape aboutit à la création de la commande [BOU 88],
- phase d'évaluation par simulation de la commande [CAS 87],
- phase d'implantation [CRA 89].

L'application de cette méthodologie de conception, à des exemples industriels complexes, a révélé certaines limites. Leur analyse a conduit le laboratoire à étendre le cadre de la méthodologie et ainsi à développer une version étendue, appelée CASPAIM2, de la démarche de conception.

Le travail présenté dans ce mémoire est une nouvelle contribution au projet. Il s'intéresse d'une part, au pilotage d'un atelier flexible, et d'autre part, à la conception d'une interface graphique permettant la visualisation des différents modèles élaborés en phase de conception, ainsi qu'en phase de simulation.

Ce mémoire est composé de trois chapitres :

- **Le chapitre I** présente le projet CASPAIM, et ainsi le contexte dans lequel nous travaillons, en insistant particulièrement sur les différents modèles utilisés pour l'élaboration de la commande d'un atelier flexible de production.

- **Le chapitre II** aborde les problèmes de pilotage, et notamment le traitement des indéterminismes induits d'une part, par la flexibilité des processus étudiés, et d'autre part, par l'utilisation des ressources nécessairement limitées.

- **Le chapitre III** est consacré à la nécessité, aussi bien en phase de conception qu'en phase d'implantation, de disposer d'une interface graphique permettant la visualisation de tous les modèles utilisés dans le projet CASPAIM. Il y est montré comment les différentes représentations d'un certain type de graphes sont obtenues automatiquement, puis comment la méthode est étendue à une classe plus importante de graphes.

- **Un glossaire** permet de répertorier les définitions des termes apparaissant en *italique.*

- **Une annexe** apporte un complément d'information à tous les points qui n'auront pu être développés au sein du mémoire.

Chapitre I

Introduction.

Le travail, objet de ce mémoire, fait partie du projet CASPAIM, dans l'objectif d'élaborer une méthodologie de conception d'*ateliers flexibles* de production. La dimension du projet a conduit à une structuration des travaux de recherche sur différentes thématiques complémentaires.

Afin d'avoir un point de vue d'ensemble sur le projet, permettant de situer notre contribution, nous proposons dans ce chapitre, en premier lieu, d'établir un bilan général sur le concept d'ateliers flexibles de production.

Puis, nous nous intéresserons à l'élaboration de la méthodologie initiale CASPAIM1, et de ses outils permettant la prise en charge, depuis une phase de spécifications jusqu'à une phase d'implantation, des étapes de conception, de validation, d'exploitation et de maintenance de *cellules* de production flexibles.

Enfin, après analyse des limitations de cette méthodologie face à des exemples plus complexes, nous présenterons la version étendue, appelée CASPAIM2.

I. Approche globale.

I.1. Introduction.

La conception d'un atelier est une tâche complexe, où les choix sont nombreux et difficiles. Bien souvent, des erreurs de conception sont commises du fait d'une mauvaise communication, de la négligence de certains éléments essentiels, d'une approche peu rationnelle des problèmes, ...

Afin de mieux comprendre l'apport d'une conception globale d'un atelier flexible de production, nous nous proposons de faire un bref bilan. Pour cela, nous nous appuyons sur deux livres : "La conception des usines de demain" [BEN 85], et "Les ateliers flexibles de production" [BON 85].

I.2. Les notions.

Pour satisfaire l'objectif final d'une usine manufacturière qui est de transformer des matières premières en produits finis pour un coût minimum, il s'agit de faire communiquer de manière cohérente quatre systèmes de nature différente mais interdépendants qui la compose :

- **le système physique** : sous-système production - manutention - stockage, incluant les automatismes et l'informatique industrielle nécessaires à son fonctionnement,
- **le système humain** : englobe la totalité de l'effectif humain intervenant sur le site,
- **le système nerveux** : l'ensemble de l'organisation de l'information et des circuits de décision,
- **le système flux de produits** : flux physique de tous les composants entrant dans l'usine.

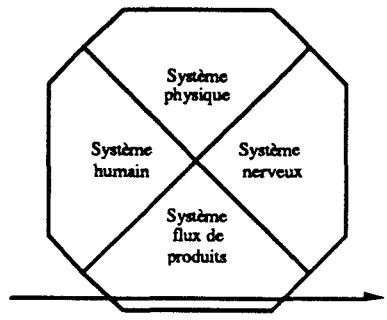


figure I.1

Nous faisons la différence entre la notion d'usine qui réunit les quatre systèmes et la notion d'atelier qui est le cœur de l'usine. Nous appelerons **atelier** un système qui est composé des système physique, système nerveux et système flux de produits, en excluant le système humain, bien qu'il ait une certaine influence sur le fonctionnement de l'atelier.

La première caractéristique d'un atelier automatisé **flexible** est une capacité à produire des pièces différentes, par opposition aux machines des pièces spéciales ou transferts qui ne peuvent exécuter que des produits identiques dans des limites fixées au préalable.

Si nous nous arrêtions à cette définition, les exemples seraient légions. Il existe en effet, une multitude d'ateliers de petites séries qui fabriquent des produits très divers. Même en grandes séries un certain niveau de flexibilité a toujours été recherché puisque généralement les matériels utilisés sont capables de produire des variantes d'un même ensemble ou d'un même produit.

L'atelier flexible peut revêtir des formes et des destinations très variées depuis la production en forte cadence jusqu'aux systèmes réalisant des pièces prototypes. Le problème de fabrication est excessivement vaste et la solution n'est pas unique.

Il n'existe donc pas **une** notion unique d'atelier flexible mais un **concept** de flexibilité.

Pour essayer de donner une définition plus précise, essayons de caractériser les différents niveaux de flexibilité.

De notre point de vue, nous pouvons différencier quatre niveaux intervenant dans la définition du terme flexibilité :

1- les machines, caractérisées par la diversité des opérations réalisées automatiquement sur chaque *poste* de travail dans une classe d'opération donnée,
2- les objets, par la possibilité éventuelle de réaliser des opérations différentes dans un ordre indifférent,
3- le système de transport, qui gère l'enchaînement des différents postes par un convoyage non restrictif autorisant toutes les liaisons possibles,
4- le système de conduite, capable de régler les situations conflictuelles de l'atelier en respectant les divers objectifs fixés.

Avant de décrire les objectifs à atteindre pour la conception d'un atelier flexible de production, établissons un bilan des différents éléments qui le constituent :

Les flux de matières, de produits finis ou semi-finis, d'informations

- ils étaient, en général, prédéterminés et peu variables dans le temps,	- ils deviennent éminemment variables, c'est-à-dire que l'outil de fabrication doit être **flexible**,
- ils étaient linéaires et faciles à maîtriser,	- ils sont maillés, et donc plus complexes,

Les machines de fabrication

- elles étaient, le plus souvent spécifiques à un type de fabrication ou à un produit,	- elles deviennent flexibles, capables de traiter des produits différents, elles intègrent le plus souvent plusieurs opérations,
- les temps de réglage et de changement d'outillage étaient long,	- ils sont réduits au minimum, (temps masqué),
- les contrôles se faisaient, en général, hors ligne de production,	- ils sont maintenant en ligne ou intégrés dans les machines elles-mêmes.

Les moyens de stockage

- ils étaient le plus souvent autonomes,	- ils sont de plus en plus intimement liés aux moyens de production.

Les interfaces

- il y avait peu d'interface entre les différents moyens,	- elles deviennent nombreuses en raison de l'automatisation. Elles sont gérées par des systèmes de pilotage informatique en temps réel, eux-même interfacés avec la gestion de production de l'usine.

L'informatique

- elle était principalement axée sur la gestion et le calcul scientifique,
- elle devient omniprésente aussi bien au niveau de l'atelier, pour piloter en temps réel les procédés et gérer l'ordonnancement à court terme, qu'au niveau général, pour la conception, la gestion, ...

Ajoutons à ce comparatif, la définition de l'univers de fabrication vis à vis des produits [BOU 90a].

La réalisation de tout produit manufacturé requiert trois types d'ingrédients :
- l'ensemble des composants élémentaires du produit,
- l'ensemble des outils de manipulation (préhenseurs, posages) et de *procédé* physique (usinage, peinture, ...),
- l'ensemble des opérateurs : machines, robots, opérateurs humains.

Parmis les *opérations* applicables aux produits, nous distinguons :
- les fonctions transformationnelles, qui modifient l'état physique du produit :
  - les usinages : tournage, fraisage, lavage, graissage, peinture ...
  - les assemblages, qui conduisent à la mise en liaison :
    - soit de deux composants élémentaires,
    - soit d'un composant élémentaire et d'un ensemble de composants déjà liés entre eux,
    - soit de deux ensembles de composants déjà liés entre eux,
- les fonctions informationnelles, qui permettent de garantir la conformité des états physiques obtenus après des interventions extérieures, opérations de métrologie, ...

La conception d'un atelier flexible de production doit donc être **globale**. Jusqu'à présent l'atelier de fabrication comprenait des équipements divers de production, de manutention, et de stockage, en général juxtaposés, avec peu d'interface entre eux. La recherche d'efficacité se traduit surtout actuellement par l'échange, le partage et la mise à jour automatiques des informations entre les machines et entre les différents services de l'entreprise. Les différentes entités du cycle de production ne doivent plus avoir un fonctionnement autonome mais **coordonné.**

Ceci définit la notion d'***usine intégrée*** (ou CIM, Computer Integrated Manufacturing).

Cette notion s'accompagne d'une augmentation conséquente de la taille et de la complexité des systèmes de pilotage dont la maîtrise fait intervenir les notions de **structuration** et de **modularité.**

Le problème soulevé par la volonté de structuration est un problème actuel et crucial. Il a fait l'objet de discussions au sein d'un groupe de travail "Systèmes logiques" de l'AFCET. En effet,

- les méthodes de construction modulaire peuvent être descendantes (décomposition) ou ascendantes (intégration),
- le choix des limites et des interfaces de chaque module est complexe,
- la démarche de structuration doit intervenir, à la fois, pour la spécification du problème et pour la spécification de la solution. Ces deux démarches doivent être intimement liées.

I.3. Conclusion.

Les problèmes posés sont relatifs à la représentation des systèmes de commande intégrant des exigences très diverses et à la façon de construire proprement cette représentation.

L'élaboration de cellules et d'ateliers flexibles de production reste très problématique, notamment :

- par l'hétérogénéité et l'absence de standardisation des systèmes de commande qui conduisent à une résolution empirique des problèmes de coordination intermachines,
- par l'absence de méthodes de conception, d'évaluation, et d'implantation,
- et par l'absence d'une méthodologie de construction constituant un standard de communication.

II. CASPAIM1.

II.1. Introduction.

C'est dans le contexte présenté dans la partie précédente, que divers projets sont nés de la concertation entre le monde industriel et la recherche universitaire, PIASTRE du LAM à Montpellier [PRU 87], SECOIA du LAAS à Toulouse [ATA 87], [SAH 87], [VAL 88], et CASPAIM du LAIL, afin de concevoir des méthodes de conception rigoureuses et systématiques de systèmes de commande d'ateliers flexibles de production.

Le projet CASPAIM s'intéresse à l'élaboration d'une méthodologie et des outils prenant en charge la conception, la validation, l'exploitation et la maintenance de cellules de production flexibles depuis la spécification du cahier des charges jusqu'à l'implantation finale sur site.

Ce projet ambitieux qui est le fruit du travail d'une équipe, a surtout permis de mettre au point une méthodologie originale qui s'étend jusqu'à l'implantation, et qui a permis de mettre en place une chaîne complète de logiciels cohérents.

II.2. Les modèles.

II.2.1. Introduction.

Depuis 1979, le choix de l'élaboration des modèles de représentation du système de pilotage et de commande des unités de production flexibles s'est établi progressivement au LAIL afin de prendre en compte, la flexibilité des systèmes et le caractère polyvalent des différentes *ressources* de l'atelier de production.

La démarche que nous avons adoptée pour établir l'analyse et la synthèse de commande d'ateliers flexibles, consiste d'une part à mettre à plat les différents niveaux qui constituent la partie procédurale du système de commande et d'autre part, à considérer un *niveau hiérarchique* qui paramètre la partie procédurale en intégrant les objectifs de production ainsi que les fonctions de supervision.

Le modèle général des systèmes de production flexibles utilisés par le projet CASPAIM1 est présenté figure I.2.

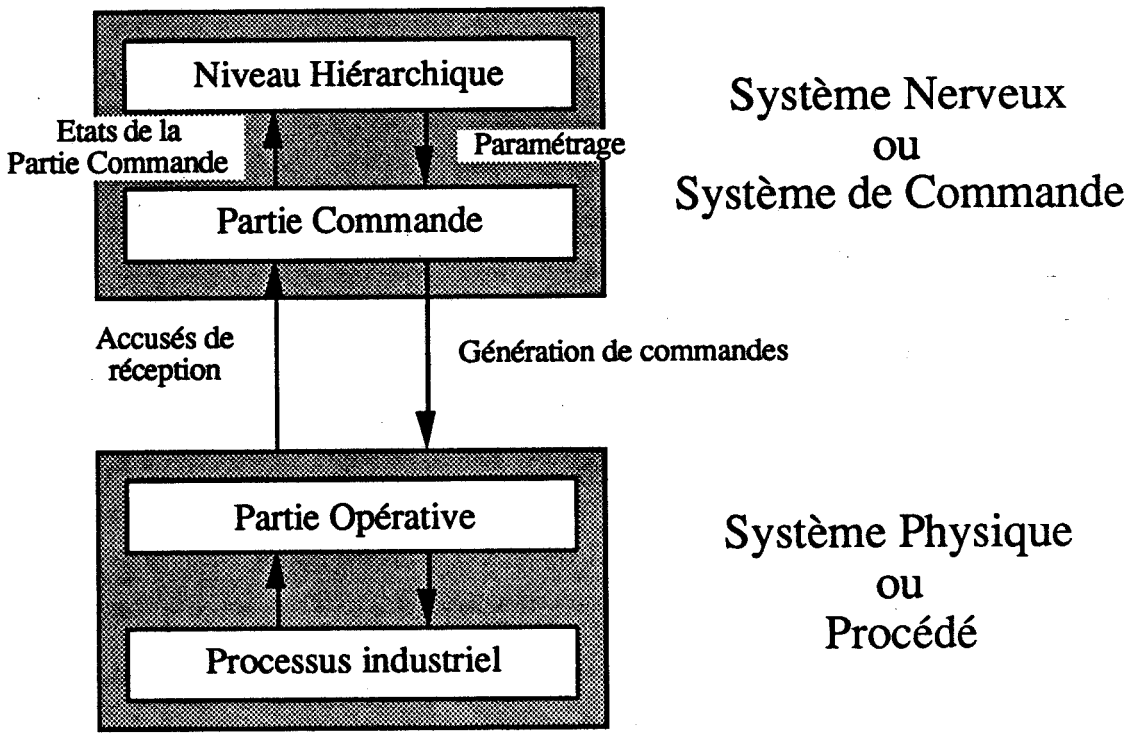


figure I.2

Les modèles utilisés par le projet CASPAIM1 pour la description de ces niveaux sont :

- les réseaux de Petri structurés adaptatifs colorés pour la *partie commande*,
- les règles de production pour le niveau hiérarchique,
- la temporisation du graphe de commande et un langage objet pour le procédé.

II.2.2. La Partie Commande.

Le rôle de la Partie Commande est d'assurer le séquencement des opérations sur les postes de transformation et la synchronisation entre les postes et les moyens de transport.

Les modèles utilisés par la Partie Commande sont les réseaux de Petri structurés adaptatifs et colorés.

Les réseaux de Petri sont sur le plan formel les modèles les plus avancés pour la représentation et la description logique des structures de contrôle des processus discrets. Ils permettent notamment d'exprimer au mieux les problèmes de synchronisation de la coopération et de la coordination des matériels, de représenter le comportement dynamique d'un système de commande discret, et constituent un modèle représentable graphiquement.

De plus, le but ultime de CASPAIM est l'implantation des modèles sur systèmes

cibles. L'utilisation de ces modèles est d'autant plus intéressante que, dans le cas d'*automates programmables*, l'outil de programmation est généralement le Grafcet, modèle de type graphe à événements discrets issu des réseaux de Petri. Malgré, la puissance des réseaux de Petri structurés adaptatifs et colorés, utilisés pour la modélisation au sein de la méthodologie CASPAIM, par rapport au Grafcet, la transposition est envisageable. Cette possibilité a fait l'objet d'un mémoire de thèse [CRA 89].

Donnons à présent quelques notions des modèles utilisés dans CASPAIM1, qui ont été développées en détail dans [BOU 88a] et [CAS 87].

- les places sont associées au concept d'état du système. Dans le cas présent, le système correspond à la Partie Commande, où la plupart des états sont attachés à un ou plusieurs ordres que la Partie Commande envoie à la *Partie Opérative.*
- Les transitions sont associées à la notion d'événement dont l'occurrence modifie l'état du système.
- Le marquage de certaines places et l'évolution de ce marquage dans le graphe permet d'identifier un état du système et ainsi les ordres que la Partie Commande doit activer.

Nous ne rappelerons pas ici les définitions concernant les règles de validation et de tir des transitions, [BRA 83], [CAS 87], [BOU 88a].

Le caractère **structuré** a été introduit afin de donner aux réseaux de Petri les caractéristiques d'un langage de programmation structurée et de faciliter une implantation répartie du modèle [BOU 87a], [GEN 87], [BOU 88b].

Nous distinguons deux types de graphe qui constituent la décomposition modulaire du graphe de commande :

- les graphes de processus,
- les graphes de liaison.

Les **graphes de processus**, apparentés à la notion de procédure informatique, décrivent un enchaînement de tâches séquentielles, et peuvent être représentés par un réseau de Petri structuré grâce à la combinaison de trois structures élémentaires : l'action, l'alternative, la répétitive, (cf. Annexe I.1.). Ainsi, chaque graphe de processus est associé à chaque lieu de transformation des produits.

Les **graphes de liaisons** assurent la coordination et l'interaction entre processus. Afin d'assurer ces interactions, trois types de liaisons ont été définies (cf Annexe I.2.) :

- l'exclusion mutuelle,
- la synchronisation avec accusé de réception,
- le producteur/consommateur.

Le caractère **adaptatif** permet de paramétrer la Partie Commande par un niveau décisionnel et rend ainsi les modèles capables de modifier leur propre condition de fonctionnement en intégrant un comportement flexible. Ce caractère permet de modéliser simplement, par exemple, un changement de mode de marche nécessité par l'état du procédé, (cf. figure I.3).

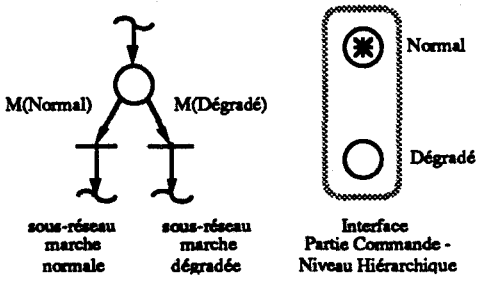


figure I.3

Si la place Normal (respectivement Dégradé) est marquée, l'arc adaptatif pondéré par la place Dégradé (resp. Normal) inhibe l'évolution de la partie droite (resp. gauche) du graphe.

La **coloration** a pour objectif de simplifier la structure d'un réseau, et permet de faire évoluer au sein d'un même graphe un ensemble fini de marques différentes. Par l'interprétation de leur couleur, elles permettent ainsi d'identifier l'état d'avancement d'un objet dans sa gamme opératoire. De plus, la coloration permet de réaliser des regroupements fonctionnels pour des opérations de même nature sur des produits différents s'effectuant sur une même machine, (cf. figure I.4).

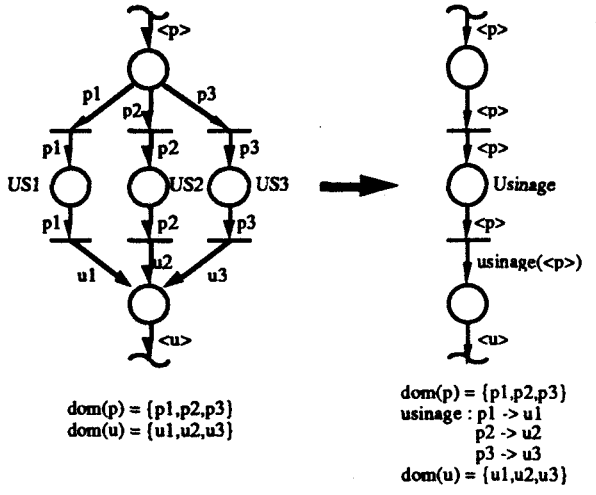


figure I.4

Trois pièces brutes p1 p2 p3 différentes sont à usiner sur une machine. Le résultat après les usinages spécifiques US1 US2 US3, donnera les produits u1 u2 u3.

La modélisation de la partie commande utilise donc les réseaux de Petri structurés adaptatifs et colorés qui permettent ainsi :

- une description fonctionnelle du système de production : caractère structuré,
- un interfaçage avec le niveau hiérarchique : caractère adaptatif,
- une localisation et une différenciation des différentes familles de pièces : caractère coloré.

II.2.3. Le Niveau Hiérarchique.

Le rôle essentiel du niveau hiérarchique est de construire les décisions et les choix destinés à résoudre les indéterminismes de la partie commande.

En effet, la flexibilité induite par les différentes possibilités de routage et de permutations des opérations pour un même produit, et les caractéristiques liées à la sûreté de fonctionnement, accroissent considérablement les possibilités de comportement de la partie commande et donc les choix à prendre pour assurer le déterminisme de circulation des marques du modèle.

C'est pourquoi, nous avons regroupé les mécanismes de paramétrage du graphe de commande résultant des différentes prises de décision au sein d'un niveau hiérarchique **distinct** de la partie commande. Ceci permet ainsi :

- de garder le caractère flexible de la partie commande,
- de pouvoir appliquer des stratégies différentes sans devoir reconfigurer le graphe de commande,
- et, d'avoir une vision globale du système pour intégrer la gestion de production et de supervision, (*ordonnancement*, *planification*, réaction aux perturbations, ...).

Différentes études ont été effectuées pour la modélisation du niveau hiérarchique :

- utilisation des réseaux de Petri, homogène avec le modèle de la partie commande,
- utilisation de règles de production.

La modélisation du niveau hiérarchique retenue est basée sur l'utilisation de règles de production interprétées par chaînage mixte, car le traitement des incidents et des diagnostics est facilité par une approche de type intelligence artificielle d'une part, et d'autre part, l'approche par règles de production rend les modifications hors ligne ou en ligne du contrôle

des stratégies plus aisées.

La structure des règles de production est :

- un identificateur de règle,
- un ensemble de variables libres,
- un énoncé de type **si** <*condition*> **alors** <*action*>

où - la partie <*condition*> est constituée d'une combinaison logique de prédicats testant l'état du graphe,

- la partie <*action*> est constituée d'une séquence d'opérateurs permettant d'ajouter ou d'enlever n marques d'une couleur donnée dans la place d'interface considérée.

La règle de production exprimant le passage en mode dégradé pour l'exemple de la figure I.3 serait :

règle : passage en fonctionnement dégradé
si <*condition*>
alors (enlever(1, *, Normal) **et**
ajouter(1, *, Dégradé))

Notons que les places d'interface entre ces règles et le graphe de commande sont des places qui ne sont pas connectées au graphe et dont le marquage est modifié par la partie <*action*> des règles.

La modélisation du niveau hiérarchique basée sur l'utilisation de règles de production permet ainsi :

- une grande facilité de traitement des incidents,
- l'établissement de diagnostic,
- une grande maîtrise de conception de la stratégie.

II.2.4. Le Procédé.

Le procédé recouvre l'ensemble des objets matériels auxquels s'adressent les ordres de la partie commande et émet en retour à celle-ci un certain nombre d'informations. Il est constitué :

- de la partie opérative, composée d'actionneurs, capteurs et boîtiers de commande numérique, qui permettent la communication entre le procédé et la partie commande,

- des machines et des moyens de l'unité de production, dont la durée des opérations est synthétisée par des temporisations dans la partie commande,
- des outils et des produits manufacturés, dont l'image est reconstituée sur la partie commande par la coloration.

Il s'avère que pour :

- l'élaboration des statistiques de fonctionnement,
- la vérification de la bonne conception du système de commande vis à vis du comportement fin du procédé,
- la simulation des défaillances et des anomalies,

cette représentation approximative est insuffisante, il est alors nécessaire de disposer d'un modèle spécifique du procédé en parallèle avec l'image qu'en a la partie commande.

La modélisation par langage objet a fait l'objet d'une étude [CAS 87], où le procédé était représenté par des objets organisés en arborescence générique, et permettait d'effectuer une évaluation qualitative des performances. Cette piste a été abandonnée pour des raisons que nous rappelerons dans la partie II.4. de ce chapitre.

L'évolution de la partie commande dépend donc de l'état instantané du procédé. C'est pourquoi, durant la phase de conception et d'évaluation d'un système, il est nécessaire de construire un modèle provisoire du procédé qui sera remplacé par le procédé réel au moment de l'implantation.

II.2.5. Conclusion.

Les modèles que nous venons de présenter, permettent de concevoir le système de commande d'un processus de fabrication, d'en faire une première évaluation par simulation et s'adressent essentiellement à la réalisation et à l'implantation du système de commande de l'unité de production.

Le modèle réseau de Petri structuré adaptatif et coloré permet de préciser l'architecture parallèle d'ensemble de la partie commande et de fournir les éléments nécessaires pour caractériser les modes de marche sans régler les conflits.

L'ensemble des règles de production du niveau hiérarchique permet d'assurer le comportement déterministe et le paramétrage de la partie commande.

L'utilisation d'un langage objet pour la modélisation du procédé et/ou la temporisation dans la partie commande permettent l'évaluation des performances du système.

II.3. La démarche de conception.

II.3.1. Introduction.

L'optimisation et la maîtrise de l'outil de conception nécessitent la mise en œuvre d'une méthodologie d'approche globale pour la conception des ateliers.

Le but du projet CASPAIM est de construire une méthodologie et des outils constitués d'un ensemble de logiciels d'aide à la conception et à l'implantation du système de commande d'unités flexibles. La méthode de conception itérative employée est de type modélisation-simulation.

Présentons brièvement la démarche de conception du projet CASPAIM1.

Sur le cycle de vie d'une application développée avec CASPAIM1 (cf. figure I.5), nous distinguons quatre phases :

- phase I : spécification du cahier des charges,
- phase II : modélisation,
- phase III : simulation,
- phase IV : implantation, c'est-à-dire synthèse du système de commande sur support informatique.

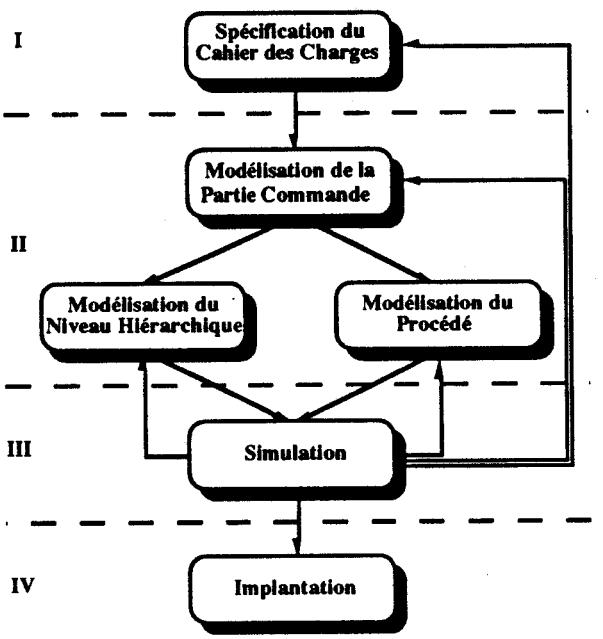


figure I.5

II.3.2. Exemple.

Le but n'étant pas ici de faire une description exhaustive de la démarche de conception, nous avons choisi de l'illustrer sur une petite cellule flexible de production (cf. figure I.6). Elle est constituée :

- d'un tampon d'entrée E,
- de deux machines M1 et M2 équipées de deux tables permettant d'effectuer les entrées et les sorties de pièces,
- d'un robot R qui assure le transfert des pièces entre les différents constituants du système,
- d'un tampon de sortie S.

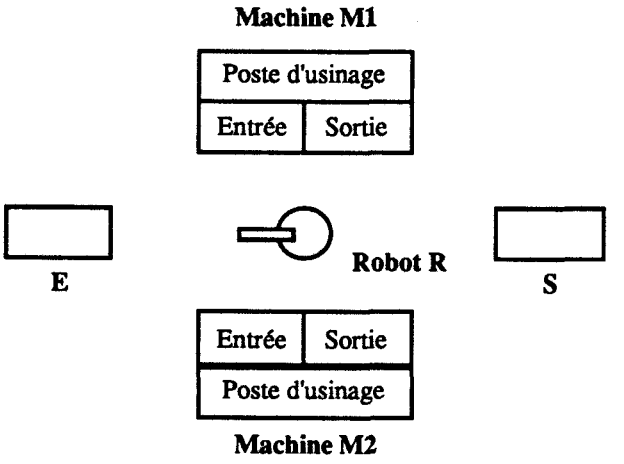


figure I.6

Quatre types de pièces circulent dans cette cellule. Ces pièces sont usinées :

- soit sur la machine M1,
- soit sur la machine M2,
- soit sur la machine M1 et M2 dans un ordre indifférent.

Pour simplifier les notations, nous appelons :

- m1 les pièces à usiner sur la machine M1,
- m2 les pièces à usiner sur la machine M2,
- u1 les pièces usinées sur la machine M1,
- u2 les pièces usinées sur la machine M2.

Nous considérons de plus, qu'une pièce ne peut accéder à la machine M1 (resp. M2), que si la machine M1 (resp. M2) est libre (présence d'aucune pièce sur le poste d'usinage et/ou sur la table de sortie de la machine considérée).

II.3.3. La phase de spécification.

Cette phase vise à obtenir à partir du cahier des charges, une description formelle des *gammes opératoires*, [KAP 87], [KAP 88]. Chaque gamme opératoire est décrite par un ensemble de règles de production de type

prémisses ➡ conséquents

où chaque prémisse et conséquent est constitué d'un prédicat ou conjonction de prédicats de la forme : (présence objet lieu)

La description des gammes opératoires pour chaque type de pièces s'effectue à l'aide d'un logiciel qui permet d'effectuer une analyse de cohérence et de complétude de cette description afin de s'assurer que la formalisation respecte bien les impératifs spécifiés au niveau du cahier des charges.

Deux des règles de production de la cellule présentée dans la partie II.3.2 de ce chapitre seront par exemple :

usinage : (présence m1m2 M1) ➡ (présence u1m2 M1)
transport : (présence u1m2 M1) ➡ (présence u1m2 M2)

A partir des règles de production, un modèle intermédiaire, que nous appelons ***prégraphe***, est généré [KAP 88]. Ce modèle est une agrégation des règles sous forme de réseau de Petri coloré où chaque place correspond à un lieu, une marque colorée à un objet et une transition à une règle de production. Ce graphe représente ainsi l'ossature initiale du modèle procédural structuré qui conduit à l'élaboration de la partie commande du système.

Le prégraphe de la cellule de la figure I.6 serait :

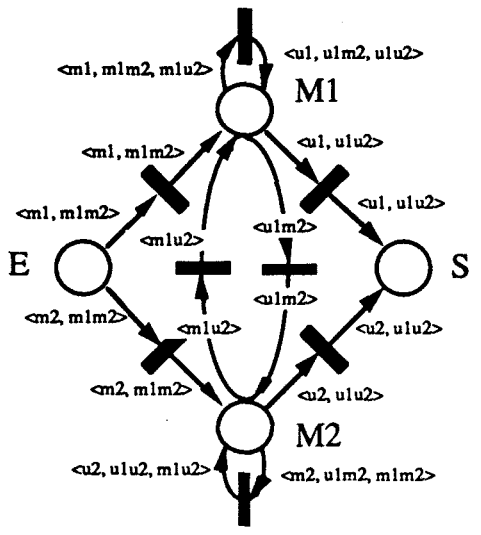


figure I.7

II.3.4. La phase de modélisation.

Cette phase a pour but d'assister la construction des trois modèles du système qui permettent l'analyse qualitative et quantitative du comportement dynamique du procédé, de sa commande et du niveau hiérarchique, c'est-à-dire du système complet.

- **Le graphe de commande** modélisant la partie commande est obtenu en procédant au développement du prégraphe et à la structuration (cf. partie II.2.2 de ce chapitre).
  Le développement de la place M1 (cf. figure I.8) du prégraphe (cf. figure I.7) modélisant la machine M1, consiste à associer à cette place, un graphe de commande qui assure l'alimentation de la machine en pièces, le séquencement des opérations à effectuer sur cette machine, et enfin, à l'évacuation des pièces usinées.

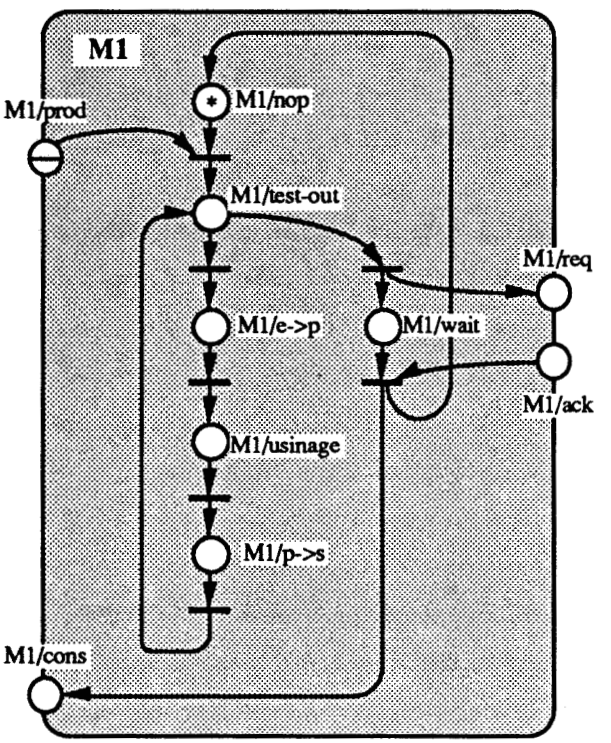


figure I.8

Le graphe de commande, développé sous forme de réseau de Petri structuré adaptatif et coloré de la partie commande, est à parallélisme maximal et tient compte de la nature et de la capacité des zones opératoires.

Cette phase de structuration est suivie d'une phase de réduction du parallélisme, par introduction des exclusions mutuelles et synchronisations traduisant le partage des ressources de transport et de stockage ainsi que la synchronisation des fonctions de manutention.

- **Le modèle du niveau hiérarchique**, sous forme d'un ensemble de règles de production (cf. partie II.2.3), permet ainsi de régler les conflits, les indéterminismes structurels, les blocages possibles, ...
  L'écriture des règles de gestion des ressources partagées, s'effectue après une phase de détection automatique des conflits et indéterminismes potentiels.

- **Le modèle du procédé**, sous forme de temporisation du graphe de commande (cf. II.2.4), permet d'effectuer le dimensionnement des zones de stockage et le choix des séquences d'entrées.

II.3.5. La phase de simulation.

La *simulation* est à la fois un outil de validation et d'aide à la conception du niveau hiérarchique, et constitue ainsi le seul moyen possible à l'évaluation des choix stratégiques.

La simulation d'un atelier décrit l'enchaînement de ces différents états dans le temps. Elle ne permet pas de résoudre un problème d'implantation ou d'ordonnancement dans le sens où elle trouverait automatiquement la solution optimale, mais elle permet de tester des solutions et de choisir la meilleure.

C'est dans ce sens qu'un simulateur qui prend en compte les trois modèles décrits précédemment, a été développé [CAS 87].

La simulation du système complet permet de mettre en évidence d'éventuelles anomalies de comportement. Les bouclages sur les phases précédentes permettent, en jouant sur différents éléments (modification de la base de règles du niveau hiérarchique influant sur la stratégie de pilotage de l'installation, doublement d'un site afin d'accroître la flexibilité, ...), de répondre progressivement à l'ensemble des critères imposés et de trouver ainsi une ou des solutions admissibles.

Cette phase terminée, il est donc possible d'envisager la phase d'*implantation*.

II.3.6. La phase d'implantation.

Lors de cette phase, nous procédons à une traduction systématique des modèles obtenus sous forme de modèles implantables [CRA 89].

La synthèse du système de commande (niveau hiérarchique - partie commande) sur support informatique est réalisée à partir du graphe de commande et de l'ensemble des règles et du moteur d'inférence formant le niveau hiérarchique.

Le support informatique est formé le plus généralement :

- d'un organe de pilotage (micro-ordinateur),
- d'un ou plusieurs réseaux locaux,
- d'un ensemble d'automates programmables industriels (API) et/ou d'armoires de commande numérique (*CNC*).

L'architecture matérielle générale d'un tel système de commande est donnée figure I.9 où les liaisons fonctionnelles sont assurées par le ou les réseaux locaux.

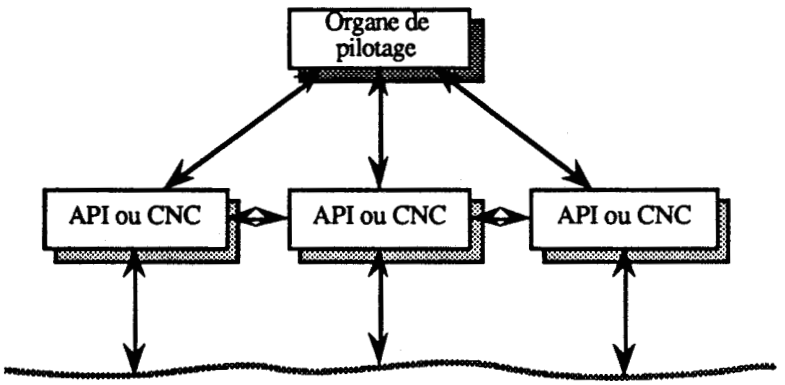


figure I.9

II.3.7. Conclusion.

Nous venons de présenter rapidement le projet CASPAIM1. Les outils qui accompagnent cette démarche permettent une assistance informatique et de tester différentes configurations de l'unité de production grâce à une conception modulaire et itérative.

Cette démarche s'est avérée suffisante dans le contexte d'une flexibilité de production limitée, souvent mise en œuvre ces dernières années. Dans un cadre plus prospectif, nous

avons pu en définir les limites et montrer dans quelles conditions de flexibilité plus complexe il s'avère intéressant de développer l'approche initiale proposée dans CASPAIM1.

Nous nous proposons d'exposer dans la partie suivante les limites de la démarches CASPAIM1.

II.4. Les limites de la démarche.

Les limites de la démarche ont quatre principales origines :

Le prégraphe est un modèle trop agrégé.

En effet, rappelons que :

- une place du prégraphe modélise un emplacement physique de transit d'objet ou un lieu sur lequel est effectuée une opération,
- une transition du prégraphe représente une transformation physique et/ou un transfert que subit l'objet,
- une marque colorée représente un objet et son état d'avancement dans sa gamme.

Il s'avère que ces hypothèses sont trop restrictives et conduisent à un modèle trop agrégé qui ne permet pas de lever un certain nombre d'ambiguïtés. Notamment, c'est le cas des assemblages sans contrainte d'antériorité (cf. figure I.10).

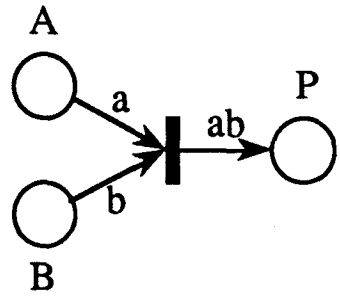


figure I.10

Opération d'assemblage de la pièce a provenant du poste A et de la pièce b provenant du poste B, sans imposer de séquencement dans l'ordre d'arrivée.

En effet, cet exemple amène deux interprétations différentes, lors de la phase de structuration qui consiste à parcourir le prégraphe et à en déduire la nature du traitement à réaliser sur les objets :

- soit l'assemblage est représenté par la transition du prégraphe, et cette

représentation amène un indéterminisme sur le lieu sur lequel s'effectue l'assemblage, avec ensuite un transfert de la pièce ab sur la poste P,
- soit l'assemblage sera réalisé sur la poste P, et la transition sera interprétée comme deux transferts distincts d'alimentation du poste d'assemblage et donc en exclusion mutuelle.

Prise en compte insuffisante du procédé.

En effet, la spécification des moyens de production est réalisée dans plusieurs phases de la démarche CASPAIM1 et de manière non homogène :

- par la spécification des lieux de transformation lors de la phase de définition des règles de production permettant d'élaborer le prégraphe,
- par la spécification des robots manipulateurs assimilés à des ressources critiques lors de la modélisation de la partie commande en phase de réduction du parallélisme,
  - cette spécification des moyens de transport n'est pas homogène dans le sens où un convoyeur et un robot par exemple ne sont pas modélisés au même niveau en tant qu'outils de transport,
- par la spécification des capacités des files d'attente et de la temporisation des actions effectuée lors de la phase de modélisation du procédé.

Il n'existe donc pas de modèle spécifique et autonome du procédé, ni de phase de modélisation associée, qui permettrait :

- d'exprimer la nature réelle du système physique,
- d'appréhender de façon **cohérente** l'emploi des différentes ressources de l'atelier,
- la prise en compte des aspects surveillance.

De plus, une modification même mineure de l'architecture de transport remet totalement en question les modèles des gammes opératoires et de la commande.

Il semble donc utile de proposer de dissocier les moyens utilisés pour disposer d'un modèle cohérent et autonome du procédé conduisant à un meilleur découplage des fonctions de commande et de surveillance.

Prise en compte des moyens plus complexes difficile

La modélisation sous forme de réseaux de Petri colorés d'un stockeur rotatif par exemple, qui est un moyen de transport mais qui est aussi un moyen de stockage, amène une explosion du nombre de places et de transitions. Pour un stockeur rotatif composé de six étagères disposant de deux emplacements chacune, le modèle prégraphe va comporter une place par emplacement et une transition par transfert.

Outre le problème de la combinatoire en nombre de places, la modélisation d'un système complexe va comporter des transitions qui ne correspondent à aucune commande effective, ce qui se trouve être en contradiction avec la définition initiale du prégraphe, (à chaque transition correspond une commande effective).

De plus, lors de la phase de structuration, vont se poser des problèmes de synchronisation.

Le problème de la prise en compte de systèmes complexes s'avère être lié aux deux problèmes déjà énoncés.

La volonté de disposer d'un modèle agrégé pour en déduire une commande qui pourra être implantée est trop contraignante ou impuissante lorsqu'il s'agit de prendre en compte des systèmes complexes.

La prise en compte éparpillée du procédé ne permet pas une gestion efficace de tels systèmes.

Au même titre que les problèmes liés à la prise en compte de systèmes complexes, la gestion de lots de pièces brutes qui après usinage doivent être reconstitués, est quasiment impossible.

Peu de relation possible avec le niveau ordonnancement.

Dans ce cadre, notons que l'emploi de la coloration ne permet pas d'avoir les informations suffisantes pour réaliser l'ordonnancement, notamment l'association à chaque objet d'une date prévisible de sortie du système ou d'une priorité, ni de caractériser la nature des produits, leur identification, ou leur éventuelle transformation.

II.5. Conclusion.

Ces limites ont conduit le laboratoire à l'élaboration d'une approche plus large et plus systématique.

Ainsi, gardant l'esprit de la méthodologie basée sur l'évaluation par simulation, nous avons contribué à l'élaboration d'une version étendue de la démarche de conception, baptisée CASPAIM2, afin d'obtenir :

- une description fonctionnelle **indépendante** des moyens opérationnels tant au niveau des moyens de transport qu'au niveau des moyens de transformation,
- un modèle de la partie commande comportant les informations nécessaires à l'ordonnancement et à la planification, tout en restant suffisamment agrégé,
- la prise en compte de cas complexes tels que, le traitement des lots et des moyens de transport complexes.

Notons que, pour l'étude de cas simples, la méthodologie CASPAIM1 demeure intéressante. De plus, le modèle prégraphe peut servir sous une forme adaptée à l'élaboration de l'architecture d'un atelier et au positionnement relatif des différents moyens de production et de transport (cf. chapitre III).

III. CASPAIM2.

III.1. Introduction.

Les objectifs de la méthodologie CASPAIM visent :

- à partir de la spécification des produits bruts et des produits finis, à définir les moyens nécessaires à l'analyse, la conception, et l'évaluation du système de commande implantable d'ateliers flexibles,
- à partir de moyens potentiels ou existants, à proposer une architecture matérielle de l'atelier.

Cette méthodologie doit donc permettre l'étude comparative de différentes solutions pour la conception d'un atelier flexible.

La démarche proposée se veut progressive, modulaire et à terme entièrement assistée. Elle vise à proposer :

- un outil méthodologique de conception de la commande qui intègre la sûreté de fonctionnement du système de commande,
- un outil d'analyse de l'architecture matérielle existante ou future,
- un outil d'évaluation de solutions complètes,
- un outil d'implantation d'une solution ainsi validée,
- un outil de surveillance en ligne,
- un outil de planification et de pilotage de l'atelier.

Pour atteindre de tels objectifs et notamment obtenir une commande implantable gardant un degré de flexibilité maximal, la méthode proposée est basée sur un découplage maximal entre ce qui concerne les pièces produites d'une part, et les moyens de production d'autre part. L'une des principales limitations de la méthodologie CASPAIM1 concernant l'absence de modélisation cohérente du procédé, conduit de ce fait à une modélisation incomplète et le plus souvent à une dégradation de la flexibilité potentielle du procédé.

Nous nous proposons d'exposer la nouvelle démarche afin de situer le contexte des travaux présentés dans ce mémoire.

III.2. Exemple.

A titre d'illustration, considérons l'exemple de la cellule suivante.

Cette dernière est une version modifiée de la cellule implantée dans le Département de Génie Mécanique de Centrale Lille qui est présentée dans la partie III.2 du chapitre III.

Elle est composée de :

- un tampon d'entrée E,
- un tour, noté T,
- un centre d'usinage, noté F, muni de deux tables équivalentes permettant d'effectuer les entrées/sorties de pièces,
- un convoyeur à chaîne, sur lequel sept butées définissent trois postes de chargement/déchargement de pièces (P5 P6 P7), et quatre files d'attente (P1 P2 P3 P4),
- un robot manipulateur, noté R1, chargé de l'alimentation du tour T,
- un robot manipulateur, noté R2, assurant les entrées/sorties de pièces ainsi que le chargement/déchargement du centre d'usinage F,
- un tampon de sortie S.

Le schéma simplifié de la cellule est le suivant :

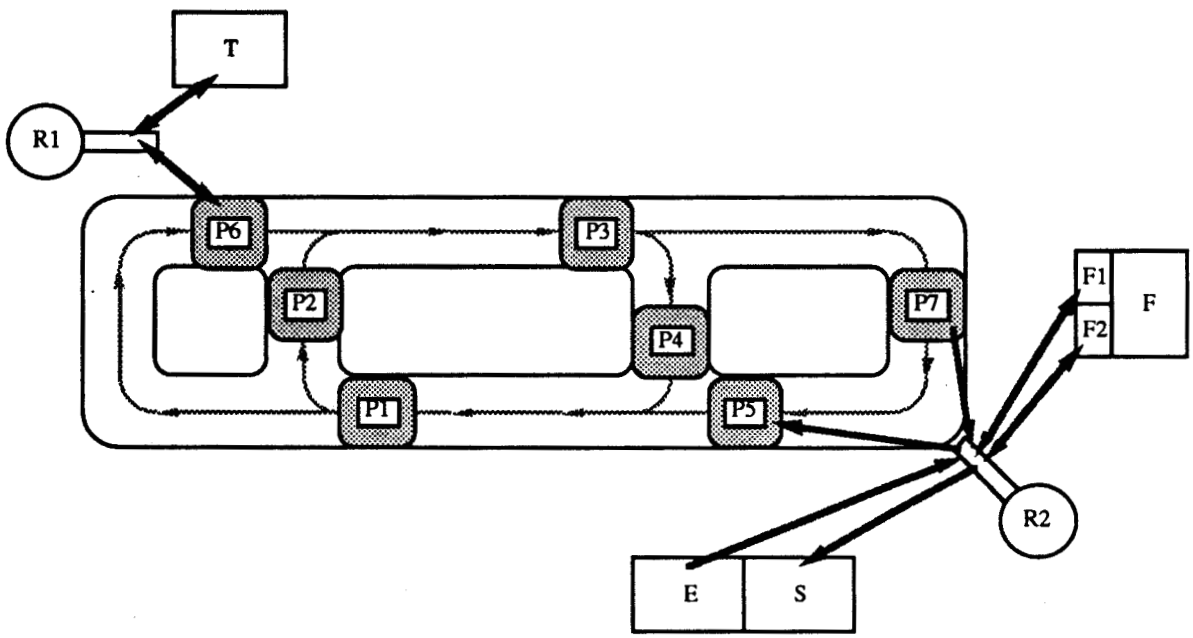


figure I.11

Les sept zones définies sur le convoyeur grâce aux butées, ont les fonctions suivantes :

P1
P2 sont les zones de files d'attente,
P3
P4

P5 est la zone de palettisation des pièces provenant de l'entrée ou du centre d'usinage,
P6 est la zone de palettisation/dépalettisation des pièces destinées au tournage,
P7 est la zone de dépalettisation des pièces destinée au centre d'usinage ou à la sortie.

Nous supposons que dans cette cellule circulent plusieurs types de pièces dont les pièces qui doivent subir un tournage puis un fraisage ou un fraisage puis un tournage.

III.3. Méthode générale.

La complexité des systèmes à modéliser nous a amené à développer une méthodologie qui rejoint les objectifs du *Génie Logiciel* appliqués aux automatismes.

Pour D.T. Ross, J.B. Goodenough et C.A. Irvine [ROS 75], le but principal du Génie Logiciel est d'atteindre un meilleur niveau de qualité des produits en terme de modifiabilité, d'efficacité, de fiabilité et d'intelligibilité.

Rappelons les définitions de ces termes [BOO 88] :

- un système est **modifiable** s'il est possible d'effectuer des changements sans augmenter la complexité du système original,
- un système est **efficace** s'il peut fonctionner en utilisant l'ensemble des ressources disponibles de façon optimale,
- un système est **fiable** s'il peut à la fois éviter les défauts de conception et de construction et permettre le traitement des pannes et des défauts de performance,
- un système est **intelligible** si il est compréhensible et est un modèle exact de notre compréhension du monde réel.

Pour atteindre ces objectifs, le Génie Logiciel introduit les concepts suivants :

- abstraction et dissimulation d'information : l'objectif est d'établir une échelle d'abstraction où chaque niveau est construit sur la base des niveaux inférieurs, en dissimulant les décisions de plus bas niveau afin d'éviter que les stratégies de plus haut niveau prennent en compte un niveau de détail trop fastidieux,

- modularité et localisation : l'objectif est de concevoir un ensemble de modules à faible couplage (limitation des échanges inter-modules), à forte cohésion (liaison étroite des éléments d'un même module) et réalistes (au sens d'un modèle effectif du monde réel),

- uniformité, intégralité et validabilité : l'objectif est d'aboutir à des systèmes cohérents et corrects par l'emploi d'une notation cohérente pour la définition des modules; ils doivent être complets en regard de leur spécification et validables.

Lors de la phase de redéfinition de la démarche de conception CASPAIM2, nous nous sommes attachés à respecter ces principes de conception.

D'un point de vue global, l'approche proposée rejoint l'approche systémique, [LEM 90]. En effet, l'absence d'étude de solution globale du système conduit à une mosaïque de solutions qui peuvent, dans certains cas, atteindre un niveau de complexité sans commune mesure avec ce que serait la solution par rapport au problème pris dans sa globalité, [TAR 86]. L'approche systémique avance l'idée que le tout est plus que la somme de ses parties et justifient ainsi l'intérêt d'une approche globale du système.

Par exemple, un système de gestion sera décomposé en trois sous-systèmes :

- le sous-système opérant, qui assure les fonctions et traite le flux physique d'événements,

- le sous-système d'information, dont l'objectif est de collecter, mémoriser, traiter et distribuer l'information,

- le sous-système de pilotage, qui décide des actions à conduire sur le sous-système opérant en fonction des objectifs et des stratégies.

Le modèle général d'un système, abordé par la nouvelle démarche de conception, est présenté figure I.12.

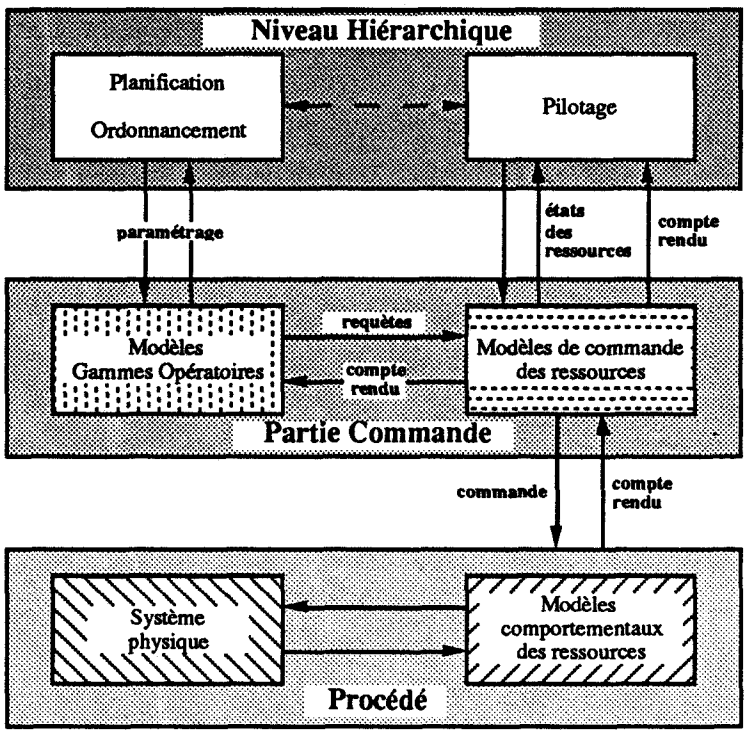


figure I.12

Les décisions relèvent du niveau hiérarchique pour atteindre les objectifs de production en intégrant un degré de flexibilité maximal. Les natures de ces décisions sont les suivantes :

- du point de vue ordonnancement/planification, il s'agit de définir les séquences d'entrée des pièces et de paramétrer les gammes opératoires,
- du point de vue pilotage, il convient de sélectionner les machines ou les moyens de transport sur lequel aura lieu l'opération demandée en choissant l'ordre d'acceptation des requêtes lors d'arrivées simultanées de requêtes,
- du point de vue surveillance, il sera nécessaire de détecter les pannes et de contrôler la gestion des modes dégradés.

Le rôle de la partie commande est ainsi de construire et d'émettre les commandes vers le procédé en accord avec le niveau hiérarchique.

Le procédé réalise quant à lui, les commandes émises par la partie commande.

III.4. Démarche.

III.4.1. Introduction.

Précisons d'avantage les objectifs de la démarche :

- A partir de la spécification des produits et des moyens existants, il s'agit	- de concevoir la commande implantable de l'atelier, intégrant la surveillance, - d'établir la répartition de la commande sur support informatique, (structure informatique),
- A partir de la spécification des produits et des moyens potentiels, il s'agit de plus,	- de fournir l'architecture physique de l'atelier,

Pour atteindre ce but, la méthode que nous présentons est basée essentiellement sur l'analyse du système physique effectuée indépendamment de la description fonctionnelle.

La démarche de conception CASPAIM2 répond à ces critères avec de plus, la prise en compte au bon niveau des différents degrés de flexibilité.

Pour la présentation de la démarche, nous considérons que nous disposons des moyens de production. En phase de conception, dans ce cas, le but est de générer la commande qui doit assurer le fonctionnement optimal de l'unité de production, en respectant la flexibilité maximale.

Examinons maintenant les différentes étapes de la démarche.

III.4.2. La phase de spécification.

Cette phase de spécification permet de fournir, par affinements successifs, les éléments nécessaires à l'élaboration de la commande en vue de son évaluation puis de son implantation.

Spécification des moyens de production.

Au cours de cette étape, nous cherchons à définir :

- la liste exhaustive des moyens de l'unité de production avec leur(s) fonction(s) associée(s),
- les relations d'accessibilité définissant tous les trajets possibles,

afin d'aboutir à une décomposition de l'unité de production en cellules élémentaires prenant en compte les degrés de flexibilité.

Pour l'exemple présenté dans la partie III.2, le résultat de cette étape de spécification fonctionnelle des moyens conduira à élaborer, dans un premier temps, la liste des moyens de l'unité de production avec leur(s) fonction(s) associée(s) :

Eléments	***Fonctions***
Tour T	Tournage
Centre d'usinage F	Fraisage - Transport - Stockage
Robot R1	Transport
Robot R2	Transport
Convoyeur	Transport - Stockage
Tampon d'entrée E	Stockage
Tampon de sortie S	Stockage

figure I.13

Puis dans un deuxième temps, à élaborer la spécification des relations d'accessibilité nécessaire pour la spécification de la décomposition en unités élémentaires de l'atelier de production.

Les relations d'accessibilité sont regroupées au sein d'un schéma (cf. figure I.14) de type entités-relations où chaque entité représente un moyen de l'unité de production, et où chaque liaison exprime la possibilité de transférer un objet quelconque d'un lieu vers un autre. Ce schéma représente donc tous les cheminements possibles en mode de marche normale d'un point de vue global.

En mode de marche dégradée, nous considérons que toutes les liaisons sont bidirectionnelles afin de permettre les recouvrements d'erreurs : une pièce provenant du tampon d'entrée pourra être remise sur celui-ci en cas de panne momentanée d'une partie du convoyeur par exemple.

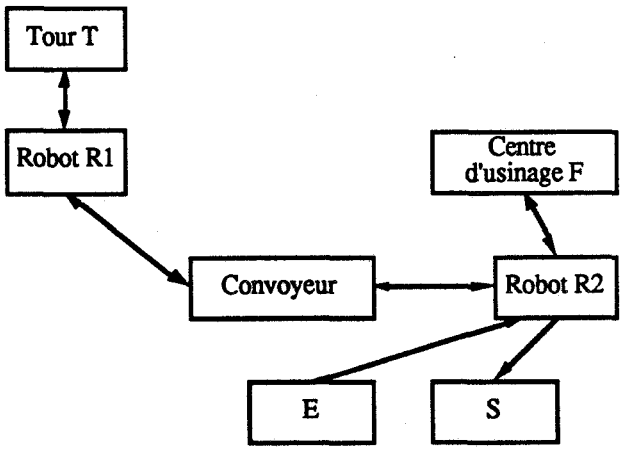


figure I.14

L'analyse de cette spécification fonctionnelle et des relations d'accessibilité des moyens de production aboutit à une première décomposition de l'unité en cellules élémentaires autonomes.

Au cours de cette décomposition n'apparaissent que les moyens dont la gestion doit être assurée explicitement par le graphe de commande. Ceci conduit à l'élimination des moyens n'assurant qu'une fonction élémentaire de stockage. Seront regroupés au sein d'une cellule élémentaire autonome, une ressource de fabrication ou plusieurs de même type, et éventuellement un moyen de transport si il n'effectue que l'approvisionnement des ressources contenues dans ce regroupement. Le tour T et le robot R1 sont donc regroupés au sein d'une même cellule, (le robot R1 est dédié exclusivement à l'approvisionnement du tour T), le centre d'usinage CU forme seul un groupement car le robot R2 effectuant son alimentation ne lui est pas dédié exclusivement.

Parallèlement à la définition des cellules élémentaires, apparaît un groupement de moyens de transport qui participent au routage globale de l'unité. Cette décomposition permet ainsi de gérer la flexibilité au bon niveau.

La décomposition pour l'exemple de la partie III.2 est représentée sur la figure I.15.

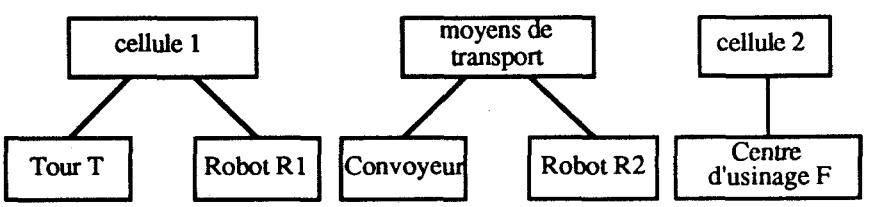


figure I.15

Spécification du processus de fabrication.

Cette phase permet de décrire la séquence des fonctions qui sont à appliquer à chacun des produits. Ces modèles ainsi générés sont appelés *gammes logiques.*

L'élaboration de ces gammes notamment des gammes d'assemblage a été largement étudiée par l'équipe d'Alain Bourjault du LAB de Besançon [BOU 84], [BOU 87b], [BOU 90b], [CAM 89], nous ne présenterons donc pas le moyen de les obtenir.

La gamme logique d'un produit fini décrit la séquence et la synchronisation des fonctions qui sont appliquées à chaque produit brut qui entre dans la composition du produit fini.

Cette description est purement fonctionnelle, c'est-à-dire qu'elle fait abstraction de l'aspect opérationnel du système.

Le modèle utilisé est de type réseau de Petri où une place modélise l'état d'avancement d'un produit, et une transition modélise une fonction qui modifie l'état d'avancement d'un produit ou crée de nouveaux produits à partir de produits existants, [CRU 91].

Les réseaux de Petri permettent d'exprimer les relations de précédence et le parallélisme d'exécution des fonctions appliquées aux produits. L'utilisation de réseaux de Petri ordinaires ne permet pas de modéliser la flexibilité des gammes de manière concise, ni d'exprimer l'assemblage de deux pièces bien précises par exemple.

Pour pallier cet inconvénient, nous utilisons une extension : les réseaux de Petri à Objets [SIB 85]. Donnons en brièvement la définition [CAR 91], [SIB 88] :

Le passage des réseaux de Petri aux réseaux de Petri à objets se fait en définissant le marquage d'un réseau non pas à l'aide de jetons indifférenciés mais avec des *objets* qui modélisent les entités manipulées par le système.

Un **objet** est défini comme une *instance* de sa *classe*, muni d'un ensemble d'*attributs* et d'un ensemble d'**opérations** (ou *méthodes*) permettant de manipuler les valeurs de ses attributs. Cette définition est enrichie, par rapport aux langages à objets, de la notion de **comportement** des objets d'une classe décrit par le réseau de Petri à objets. Ainsi, un objet, considéré comme une entité qui possède un état, n'est pas seulement

défini par la valeur de ses propriétés mais aussi par le marquage de son réseau de comportement. Le formalisme réseau de Petri permet donc à la fois de décrire la structure interne à chaque objet et celle résultant de la coopération entre ces objets.

Les places d'un réseau de Petri à objets peuvent donc contenir des valeurs qui sont soit des **constantes** (dont la valeur est indissociable du nom de la constante), soit des **objets** (dont la valeur qui est l'ensemble des valeurs de ses attributs, est modifiable). De plus, les valeurs se trouvant dans les places peuvent être liées dynamiquement les unes avec les autres; pour cela chaque place a une certaine **arité**.

Les transitions des réseaux de Petri à objets se comportent comme des fonctions dont chaque exécution prend un objet au moins dans chacune des places amont et produit en résultat un objet au moins dans chacune des places aval. Le corps d'une transition est constitué des éléments suivants :

- une **précondition** qui est une expression booléenne permettant de sélectionner les objets dans les places amont en fonction de leur valeur,
- une **action** qui permet de modifier la valeur de ces objets,
- une ou plusieurs **règles d'émission** qui déterminent en fonction de la valeur des objets dans quelles places aval ils doivent être déposés.

La gamme logique d'un produit, par définition, ne décrit que la séquence et les alternatives éventuelles des fonctions de traitement qui doivent lui être appliquées à l'exclusion de toute fonction de transport.

Dans une situation donnée, une **seule** fonction est autorisée. Ainsi, il est possible de modéliser l'état d'avancement d'un produit par un graphe d'états. La modélisation d'un état consiste à associer une place de réseau de Petri à chaque état atteint par le produit, et une transition à chaque fonction.

Considérons les pièces qui doivent subir un tournage et un fraisage dans un ordre indifférent. La gamme logique correspondante est représentée figure I.16, où apparaissent les quatre états atteints par les pièces, et les deux fonctions de tournage et de fraisage.

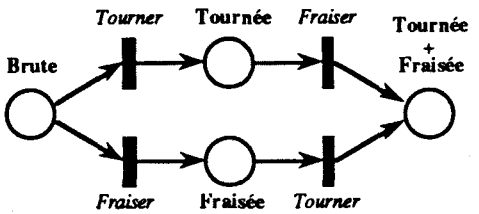


figure I.16

Cette gamme logique fait apparaître un indéterminisme induit par la flexibilité due aux permutations d'opérations dont le séquencement est indifférent. La résolution de cet indéterminisme sera effectuée en phase d'exploitation.

La phase de spécification permet donc d'effectuer en parallèle :

- la description fonctionnelle, sous forme de réseau de Petri à objets, de toutes les fonctions à appliquer sur les produits,
  ➡ obtention d'un modèle orienté produit
- la description opérationnelle, sous forme arborescente, des moyens de production,
  ➡ obtention d'un modèle orienté moyen

III.4.3. La phase de modélisation de la commande.

L'ossature de la partie commande est basée sur la description fonctionnelle des traitements appliqués à chaque objet.

La démarche de conception de la commande consiste dans un premier temps à intégrer progressivement l'aspect opérationnel par enrichissement de la gamme logique.

Cette étape permet de réaliser la synthèse entre la gamme logique décrivant les fonctions à appliquer sur chaque produit et la décomposition de l'atelier en cellules.

Le modèle que nous obtenons est un graphe d'état pour chaque pièce traitée dans lequel une pièce change de manière exclusive de lieu ou d'état. Il s'agit d'un réseau de Petri pour lequel :

- une transition modélise
  - soit un transfert d'un lieu à un autre,
  - soit une opération qui modifie l'état d'avancement du produit,
- une place marquée modélise le fait que le produit se trouve dans un lieu et un état d'avancement précis.

Le résultat de cette première étape appliquée à l'exemple de la figure I.11, est représenté figure I.17.

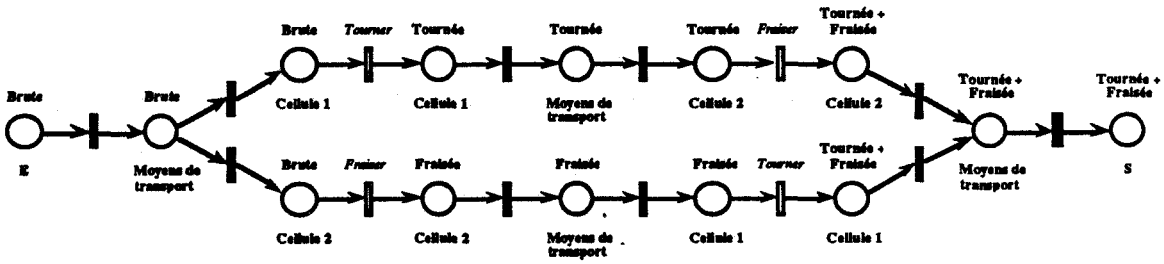


figure I.17

Cette nouvelle gamme constitue un niveau de modélisation intermédiaire entre un niveau de planification et d'ordonnancement et un niveau dédié aux commandes des cellules.

Pour cela, ce modèle doit être suffisamment concis, sans information superflue, afin que le niveau décisionnel puisse y retrouver l'information dont il a besoin le plus vite possible.

Ce modèle doit aussi permettre la coordination entre les différents processus qui composent l'atelier.

Pour réaliser l'interfaçage avec le niveau inférieur, ce modèle nécessite une étape d'enrichissement afin de solliciter séquentiellement toutes les cellules nécessaires à l'élaboration des produits.

Cette étape fait apparaître (cf. figure I.18) les communications avec les différentes cellules sollicitées, par ajout de places requêtes (demandes) et accusés de réception (fin de réalisation des demandes).

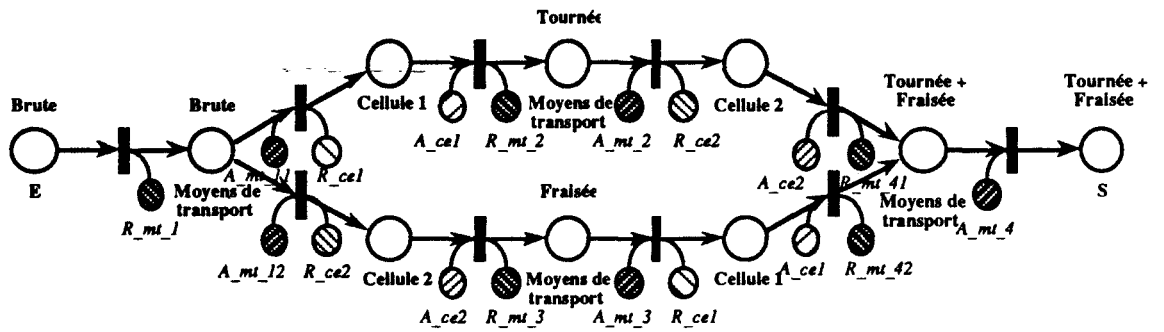


figure I.18

Notons que ce modèle, appelé *gamme opératoire*, orienté produit, ne fait apparaître

logiquement que la flexibilité sur l'ordre de réalisation des produits. De plus, un des caractères de flexibilité des cellules, possibilité de concevoir plusieurs produits dans le même atelier, est exprimé aussi à ce niveau. En effet, les différentes étapes que nous venons de décrire brièvement, permettent l'élaboration d'autant de gammes opératoires qu'il y a de produits ou de produits logiquement reliés entre eux, (lots indissociables).

L'étape suivante consiste à élaborer les commandes des différentes cellules.

Les modèles générés sont de type réseau de Petri à objets pour lesquels :

- les informations transitent par les objets,
- les liaisons avec les gammes opératoires sont réalisées par la correspondance entre les places requêtes/accusés de réception et les places "entrées"/"sorties" des modèles de la commande des cellules.

Ainsi, au travers de l'exemple, les places R_ce2 et A_ce2 de la figure I.18 apparaissent aussi au niveau du graphe de commande de la cellule 2 (cf. figure I.19).

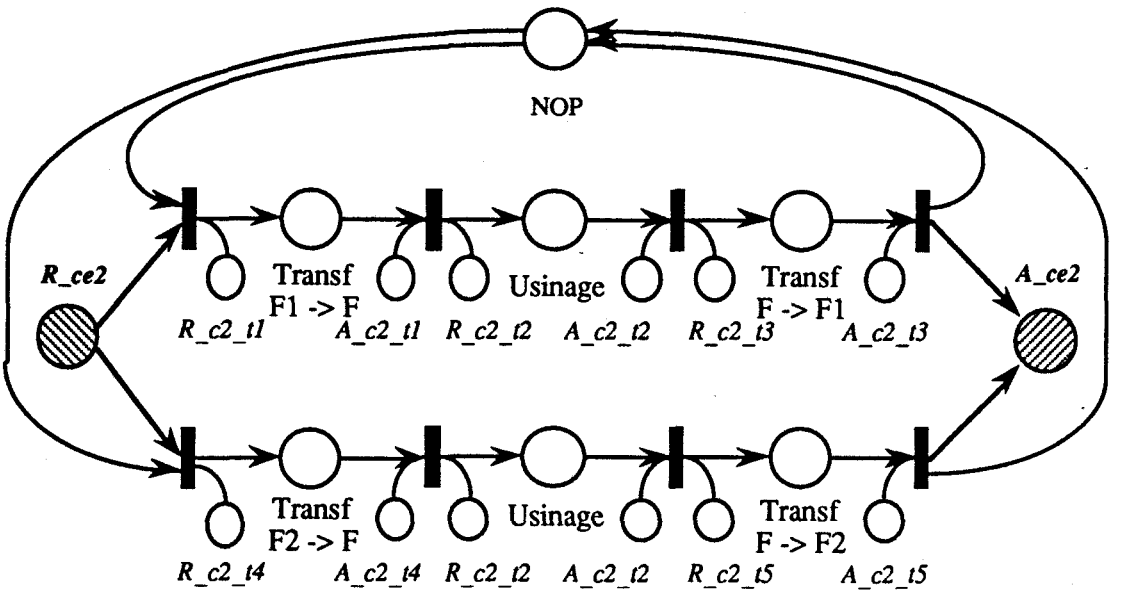


figure I.19

Ces modèles permettent ainsi d'exprimer, de manière plus précise, le séquencement des fonctions à réaliser pour chaque cellule ou groupement de moyens de transport; ils expriment également la coordination entre les différents moyens au sein d'une même cellule de façon similaire à la coordination par les gammes opératoires des différentes cellules (les modèles du niveau moyen de transport sont présentés dans l'Annexe I.3).

Sur ces modèles apparaissent progressivement les choix potentiels de routage des objets en faisant intervenir la flexibilité du système de transport, au niveau du groupement des moyens de transport, et au niveau des cellules elles-même. En effet, sur le graphe de la commande de la cellule 2 (cf. figure I.19), nous voyons apparaître la possibilité d'alimenter le poste d'usinage, soit par la table d'entrée/sortie F1, soit par la table F2, (la flexibilité d'alimentation du centre d'usinage se trouve quant à elle, visualisée sur le modèle du robot R2, cf. Annexe I.4).

Enfin, sur ce modèle de la commande de la cellule 2, nous voyons apparaître un interfaçage avec le graphe de la commande du centre d'usinage au moyen de nouvelles places requêtes/accusés de réception.

III.4.4. La modélisation de la commande effective et du comportement des ressources.

Jusqu'à présent, les modèles présentés, gammes opératoires, modèles de la commande des cellules, ne prenaient pas en compte la dynamique d'évolution des ressources, ni la commande effective, ainsi que son séquencement permettant de faire évoluer ces ressources de production.

Les différents modèles de la commande effective des ressources (cf. figure I.20 : modèle de la commande du centre d'usinage. Les modèles de la commande du convoyeur sont donnés en Annexe I.5) vont ainsi permettre de modéliser les tâches élémentaires à réaliser séquentiellement sur les ressources pour les faire changer d'état.

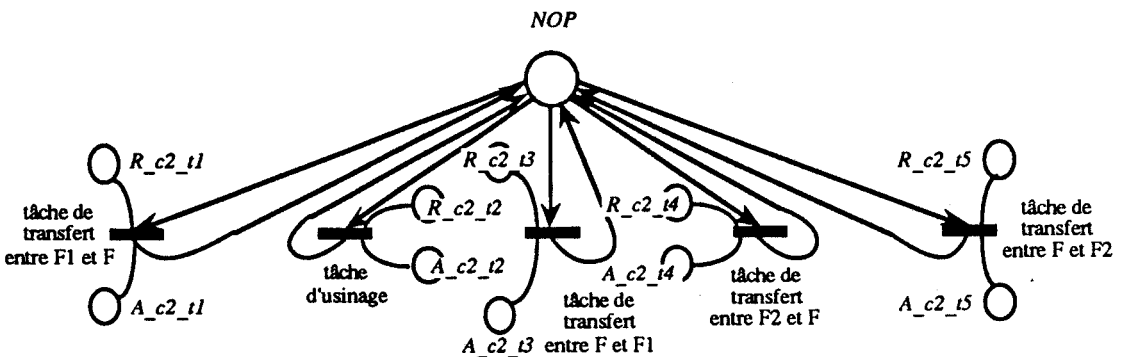


figure I.20

L'outil utilisé pour la modélisation des graphes de commande reste homogène avec les modèles des couches supérieures, car nous proposons ici encore d'utiliser les réseaux de Petri.

Les réseaux de Petri que nous proposons à ce niveau, sont les réseaux de Petri structurés présentés dans la partie II.2.2.

Dans de sens, sur le graphe de la commande du centre d'usinage, nous voyons apparaître la place NOP qui caractérise l'état initial de repos du graphe de commande de la ressource représentée. Dans ce cas très élémentaire, un seul processus (de type graphe d'état) suffit pour caractériser le comportement de la commande.

Ajoutons encore une spécificité à l'exemple de la figure I.20 : la commande effective n'a pas besoin de connaître l'état du procédé pour fonctionner. Pour des cas plus complexes, convoyeur, stockeur rotatif, ..., la commande effective aura besoin au contraire de connaître l'état du procédé et disposer des informations nécessaires exprimant son fonctionnement.

C'est pourquoi, un dernier niveau de représentation, formé soit d'un modèle comportemental (convoyeur, modèle donnée en Annexe I.6), ou d'une structure de données (stockeur rotatif), permettra en phase de simulation notamment de traduire la dynamique d'évolution des ressources complexes. Il permettra également d'assurer la prédiction de comportements futurs et de servir de modèle de référence pour les tâches de surveillance.

Nous aborderons plus précisément ce point, lors de l'examen des différents indéterminismes possibles au chapitre suivant.

III.4.5. La phase de simulation.

La dernière étape avant la phase d'implantation consiste en une étape de simulation.

La simulation peut jouer un triple rôle :

- Aider la conception :

  En effet, jusqu'à présent nous avons considéré que la structure matérielle de l'atelier était définie. La phase de simulation dans ce cas va simplement permettre d'aider au dimensionnement des différents composants internes : dimensionnement des files d'attente sur le convoyeur, ...

  L'objectif de la méthodologie CASPAIM est aussi de permettre la conception des

architectures de production. La phase de simulation est alors primordiale pour tester les différentes opportunités de choix.

De plus, en phase de conception, la simulation permet, outre le dimensionnement du procédé, de tester les différentes stratégies mises en œuvre pour la gestion par le niveau décisionnel de l'atelier.

- Valider le comportement du système complet :

En second lieu, la phase de simulation va permettre la validation du comportement dynamique du système afin d'effectuer une étude comparative des différentes solutions retenues, mais aussi afin d'étudier le comportement du système vis à vis des défaillances. Elle devra alors simuler des pannes afin d'étudier les propagations d'erreurs, ...

- Aider à la répartition de la commande :

Enfin, la phase de simulation devra permettre d'après les résultats apportés, la répartition de la commande effective sur les différents organes de commande.

III.5. Conclusion.

La synthèse des différents modèles de la phase de modélisation peut être représentée sur la figure I.21.

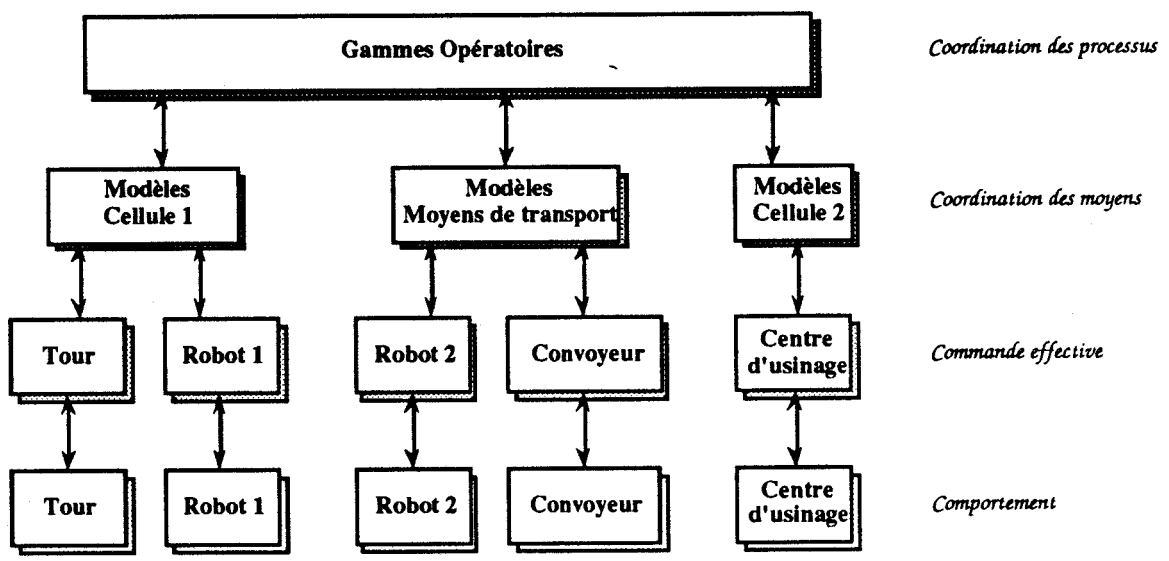


figure I.21

Nous distinguons quatre niveaux d'abstraction :

- les gammes opératoires
- les commandes des cellules et groupement des moyens de transport,
- la commande effective des ressources
- les modèles comportementaux des ressources

La démarche CASPAIM2 permet en résumé de tenir compte des différents niveaux d'abstraction en distinguant les opérations à réaliser, les moyens de production à mettre en œuvre, la commande effective des différents moyens, et leur comportement.

Elle permet également de prendre les décisions au bon niveau.

Enfin, elle constitue un outil de modélisation cohérent et validable du fait de la modularité et de la structuration générale de la démarche.

Les principes du Génie Logiciel que nous retrouvons ici, induisent une homogénéité de la méthodologie CASPAIM notamment :

- par un réel découplage entre les produits d'une part et les moyens de production d'autre part,
- par une prise en compte des différents degrés de flexibilité au bon niveau,
- par la libre résolution des indéterminismes dus aux degrés de flexibilité aussi bien localement que globalement selon les souhaits de l'exploitant.

De plus, de par la structuration utilisée dans cette approche, les différents modèles sont directement implantables sur des organes informatiques, les connexions entre les modèles sont ainsi toutes spécifiées (places requêtes/accusés), et la répartition est systématique.

Ajoutons que, par la modularité du graphe de commande, l'ajout ou le retrait d'un élément (produit ou moyen) n'entrainent pas sa redéfinition complète, mais localement l'ajout ou le retrait d'une tâche dans le module correspondant.

Enfin, la gestion des outils peut être modélisée de la même manière que les produits eux-mêmes, en établissant des tâches supplémentaires de transfert, de stockage, et de changement des outils.

Conclusion.

La méthodologie CASPAIM que nous proposons, permet donc :

l'aboutissement de la démarche de structuration qui intervient de manière cohérente pour la spécification du problème et pour la spécification de la solution,

la construction propre du système de commande intégrant des exigences très diverses,

la diminution des coûts de fabrication par l'intégration d'une flexibilité maximale,

l'apport d'une solution méthodologique intégrant la conception - l'évaluation - l'implantation.

Chapitre II

Introduction.

Dans ce chapitre, nous traiterons des indéterminismes induits d'une part, par la flexibilité du processus, et d'autre part, par l'utilisation des ressources nécessairement limitées.

Afin de préciser le cadre de notre travail, nous présentons, dans la première partie, de manière générale les différents objectifs du niveau hiérarchique, par rapport à la gestion de production.

La deuxième partie mettra en évidence de manière plus précise, les différents indéterminismes apparaissant sur les modèles de la commande en phase de simulation, et une première solution qui permet la résolution d'un type d'indéterminisme.

I. Généralités.

I.1. Introduction.

Lors de la présentation des différents modèles de la méthodologie CASPAIM2, nous avons remarqué qu'il subsistait certains indéterminismes dus notamment aux niveaux de flexibilité introduits lors de l'élaboration des graphes de commande.

D'un point de vue fonctionnel, la levée de ces indéterminismes et plus généralement le paramétrage, relèvent d'une stratégie de gestion de l'atelier liée directement avec les prises de décision réalisées dans un module décisionnel appelé niveau hiérarchique.

Avant d'établir la liste des différents indéterminismes, examinons succinctement les différents modules du niveau hiérarchique présentés sur la figure I.12, par rapport à la *gestion de production* et plus précisément à la *méthode MRP*.

En effet, toutes les activités de production sont planifiées suivant des décisions prises en temps voulu, et selon des échelles de temps appropriées. Malgré la diversité de ces décisions, essayons de les hiérarchiser afin de bien localiser l'objet de notre étude. Nous distinguons trois catégories de décisions, les décisions qui relèvent :

- de la planification,
- de l'ordonnancement,
- du pilotage.

I.2. La gestion de production.

I.2.1. Introduction.

La gestion de production a pour objet la recherche d'une organisation **efficace** de la production. En d'autres termes, le rôle de la gestion de production consiste à élaborer l'ensemble des ordres nécessaires afin que les objectifs globaux de l'entreprise soient atteints : produire une quantité donnée, dans des délais prévus, pour répondre à une demande précise, [DOU 83].

Face à la complexité de la gestion de production, les décisions de gestion sont habituellement classées en trois catégories afin de diminuer la taille du problème à optimiser, [GIA 88] :

- Les décisions stratégiques, se traduisent par la formulation de la politique à long terme de l'entreprise, (vision à plus de deux ans en général),

- Les décisions tactiques, correspondent à un ensemble de décisions à moyen terme, parmi lesquelles les décisions qui relèvent de la planification de la production, établie sur une période dont l'horizon peut varier de six à dix huit mois,

- Les décisions opérationnelles, assurent la flexibilité quotidienne nécessaire pour faire face à la demande, dans le respect des décisions tactiques. Parmi les décisions opérationnelles concernant la gestion de production, il y a celles notamment qui assurent la gestion des équipements (ordonnancement, plan de transport), et celles qui assurent la gestion de stocks.

Ces trois catégories de décisions ne diffèrent pas seulement par l'horizon qui les caractérise de prime abord. Deux éléments supplémentaires doivent être pris en compte : le niveau de compétence hiérarchique et celui d'agrégation de la décision, c'est-à-dire le niveau de détail des décisions prises quant aux productions à effectuer et aux moyens à mettre en œuvre.

Le niveau d'agrégation sera d'autant plus élevé que l'horizon de la décision sera éloigné. Ceci implique que pour des décisions tactiques ou stratégiques, le travail s'effectuera, par exemple, par famille de produits et par centaines ou milliers d'heures de travail, alors que pour les décisions opérationnelles, les décisions, portant sur des produits élémentaires et sur l'utilisation des différents postes de travail, s'effectueront dizaine de minutes par dizaine de minutes.

Précisons que, dans cette structure hiérarchisée, une décision prise à un niveau i fixe les objectifs du niveau inférieur i+1. Ces objectifs, qui ne prennent pas en compte dans le détail les caractéristiques du problème au niveau i+1, peuvent être considérés comme des contraintes pour ce niveau.

Donnons à présent, des définitions plus précises quant aux termes employés en gestion de production.

I.2.2. La planification.

Comme nous l'avons dit précédemment, la planification de la production est une

décision tactique qui répond à un souci de régulation à moyen terme de la production et constitue un lien entre les décisions opérationnelles du court terme et les décisions stratégiques du long terme.

En d'autres termes, la planification devra :

- répondre aux choix d'investissement globaux de l'entreprise, les décisions stratégiques touchant directement à la production concernent principalement le portefeuille de produits et la définition des ressources permanentes,
- fournir au niveau ordonnancement le plan directeur de production.

En fait, la planification de la production vise pour un horizon de planification, en général de quelque mois, à optimiser l'utilisation des facteurs productifs disponibles pour la production d'un ou de plusieurs produits répondant à des caractéristiques précises. Il s'agit d'un processus de traitement d'information aboutissant à une programmation prévisionnelle s'appuyant sur une démarche d'optimisation.

Le but de la planification est donc de rassembler et d'analyser périodiquement différents types d'informations, par exemple les données physiques, les données comptables, ... , afin de définir un plan de production sur un horizon de planification, [BIT 77], [KUS 83].

Un plan de production pourra contenir :

- les quantités à produire pour chaque produit,
- l'utilisation des facteurs productifs,

...

I.2.3. L'ordonnancement.

Le rôle de l'ordonnancement est donc de transformer des ordres généraux (décisions tactiques) en instructions d'exécution détaillées (décisions opérationnelles).

La définition de l'ordonnancement peut être exprimée de manière générale :

"C'est la détermination de l'ordre des différentes opérations à effectuer sur les produits en tenant compte des ressources disponibles."

Les problèmes d'ordonnancement en production se posent sur le très court terme (moins d'une semaine), et concernent de manière plus précise et donc plus réactive la

planification de l'utilisation des ressources disponibles pour répondre aux besoins définis par les décisions tactiques et stratégiques.

Les techniques d'ordonnancement relèvent de la problématique de la régulation à court terme de la production et plus particulièrement de celle de la gestion de l'utilisation des équipements de production, en complétant les techniques de gestion de stocks qui assurent la disponibilité des produits.

Résoudre un problème d'ordonnancement, [LAM 87], [BEL 88], c'est définir avec quelles ressources et à quel moment précis un certain nombre de tâches doivent être réalisées.

I.2.4. La gestion de stocks.

La gestion des stocks et l'ordonnancement représentent deux tâches distinctes mais complémentaires pour répondre aux décisions opérationnelles.

Les stocks sont par nature inévitables dans la plupart des processus industriels. Ils posent de nombreux problèmes, par exemple : tenue d'inventaire, valorisation des stocks, ... Le problème principal lié à la gestion de stocks, dans le cadre qui nous intéresse, est d'obtenir une disponibilité satisfaisante dans le temps et l'espace des différents produits.

En effet, au niveau de la fabrication, la fonction essentielle des stocks est de régulariser la production pour permettre une utilisation satisfaisante des diverses ressources de production. En effet, des ruptures de charges, peuvent être provoquées par des pannes, des ruptures d'approvisionnement, ou par l'arrivée d'un objet à un poste de travail postérieure au moment prévu. Ces perturbations sont amorties par les stocks. Ils permettent ainsi, de découpler les problèmes en limitant et retardant la propagation en cascade d'incidents. En fait, le dimensionnement des stocks est fortement lié au dimensionnement des différentes machines.

En définitive, il est clair que très souvent une action sur les stocks doit s'accompagner d'actions sur les autres ressources pour qu'il n'y ait pas de dégradation de fonctionnement du système.

I.2.5. Le pilotage.

Par pilotage, nous entendons, la conduite de l'exécution des différentes tâches élémentaires à appliquer aux divers produits en assignant les éléments physiques.

Selon la stratégie de gestion de production utilisée, deux types de conduite de l'atelier peuvent être envisagés :

Si les décisions opérationnelles ne sont pas toutes préétablies, le niveau pilotage va conduire le déroulement réel de la production en prenant des décisions fines qui dépendent toutefois de la stratégie de conduite et du niveau de flexibilité de l'atelier, [SHI 89]. Il va donc réaliser :

- le contrôle du transport entre groupement,
- le contrôle du déroulement des transformations.

Si les décisions opérationnelles sont déjà prises par le niveau ordonnancement à court terme, le superviseur (le pilote) ne contrôle que l'exécution des tâches suivant ces décisions.

I.2.6. Conclusion.

La synthèse de ce que nous venons de présenter globalement peut être représentée par la figure II.1. [CHA 83]. Nous y retrouvons les différentes notions de la gestion de production et de la conduite de production.

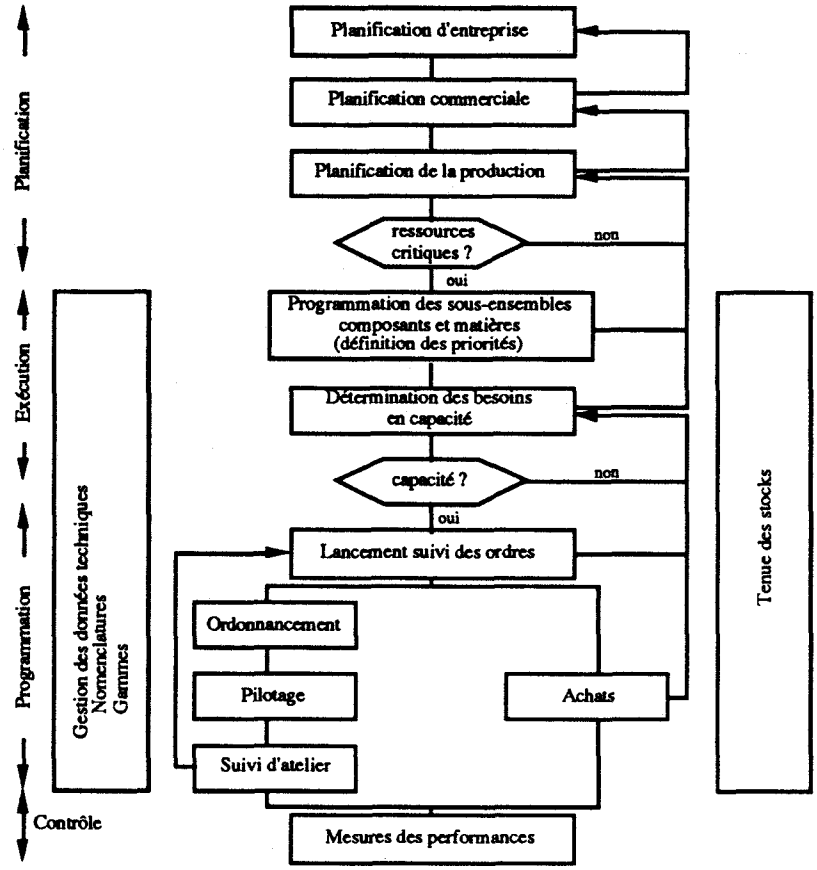


figure II.1.

Il est à noter que plus on descend dans le schéma, plus l'horizon d'intervention de la fonction considérée est court. L'échelle de temps utilisée, horizon et période d'étude, dépend de la production envisagée, en particulier du cycle de fabrication des produits.

Précisons de plus, les termes introduits sur le schéma qui n'ont pas encore été définis :

- La validation du plan de production par rapport aux ressources critiques.

Une fois que le plan directeur est établi, il convient de s'assurer à l'avance que globalement l'entreprise (ou l'atelier) disposera des ressources nécessaires pour en assurer la réalisation. Pour cela, il s'agit de déterminer, sur l'horizon du plan directeur, le profil de charge induit par le plan pour chacune des ressources critiques, et de comparer ce profil à la capacité disponible. Une ressource critique de l'entreprise représente en général sa capacité de fabrication, exprimée globalement atelier par atelier en heures productives, ou simplement la capacité de quelques goulots d'étranglement.

- La programmation des sous-ensembles composants et matières, ou le calcul des besoins.

Les objectifs de cette étape concernent respectivement :
- la définition des sous-ensembles composants et matières premières, leur quantité, et la date à laquelle ils doivent être fabriqués ou approvisionnés, pour respecter le plan de production,
- la proposition des ordres planifiés de fabrication ou d'approvisionnement correspondants, et l'indication de leur date de lancement au plus tard,
- la validation de la cohérence entre la charge de travail correspondant à ces ordres et la capacité disponible.

- La validation du plan de production par rapport aux capacités.

Cette phase va permettre de valider la phase de programmation. Cette validation s'effectue de la même manière que pour la validation du plan de production par rapport aux ressources critiques, la seule différence concerne l'horizon.

- Le suivi d'atelier.

Il consiste à enregistrer la fin des opérations et des ordres, les quantités produites et celles rebutées, les temps de fabrication et les opérations hors gamme.

- L'achat.

Il consiste à transformer les ordres d'approvisionnement issus de la programmation, en commandes aux fournisseurs.

- La tenue des stocks.

Elle représente une fonction de service des fonctions de programmation et d'exécution. Son rôle principal est d'identifier pour chaque produit, les quantités disponibles, indisponibles, réservées, en cours de fabrication ou de commande. Pour cela, elle enregistre pour chaque référence, tout mouvement d'entrée/sortie de stock.

I.3. La gestion de production dans CASPAIM2.

Nous venons de présenter quelques termes de la gestion de production, voyons à présent comment nous l'envisageons au LAIL.

La démarche du LAIL concernant le niveau hiérarchique, rejoint la gestion de production en terme général, cependant les méthodes employées ne sont pas adaptées au paramétrage et au pilotage que nous voulons réaliser dans le projet CASPAIM. Nous adaptons cependant les termes de vocabulaire définis précédemment ainsi que la hiérarchisation qui a été établie, en modifiant en outre les horizons sur lesquels les décisions seront prises. Nous parlerons donc, pour CASPAIM, de gestion de production temps réel.

Reprenons à présent la classification donnée dans le paragraphe I.2.

- les décisions stratégiques :
  * Concernent principalement le dimensionnement de l'atelier : choix des machines, moyens de transport, et leur nombre suivant leur capacité, ...
  * La remise en cause de ces décisions pourra se faire en moyenne tous les ans.

- les décisions tactiques :
  * Concernent les produits quant aux quantités à produire et les dates auxquelles la fabrication doit commencer.
  * La planification de la production va s'établir sur une période de l'ordre de la demi-heure, sur un horizon d'une journée.

- les décisions opérationnelles :
  * Dans le respect des décisions tactiques, les décisions opérationnelles vont permettre d'établir l'organisation des différentes opérations dans le temps.
  * L'ordonnancement s'effectuera sur un horizon de l'ordre de la demi-heure, affinant ainsi la période d'une demi-heure évoquée pour un horizon de planification d'une journée.

Ces décisions peuvent donc être associées à des niveaux. Les recherches engagées au LAIL, concernent pour l'instant les niveaux de planification de la production et d'ordonnancement, [HAM 91].

Le niveau hiérarchique comporte ainsi trois principaux modules :

Le module de planification : qui définit en termes de quantité à produire et de contraintes à respecter (dates limites, qualité de la fabrication, ...), les objectifs de production à réaliser.

Le module d'ordonnancement : qui détermine l'ordre dans lequel les opérations nécessaires à la réalisation de la production vont s'effectuer sur les moyens de production de façon à satisfaire les contraintes pour :
- un ensemble de moyens de production et de contraintes de fonctionnement,
- l'objectif de production déterminé par la planification.

Le module de pilotage : qui réalise l'ordonnancement choisi en suivant le prévisionnel, et résout les conflits en temps réel. Ce module est complété par la surveillance globale qui supervise l'exécution des décisions de pilotage, [TOG 91], et qui assure le diagnostic des anomalies dans l'atelier, [TOG 90].

Précisons qu'avant de réaliser l'ordonnancement, la planification est validée. La validation de l'ordonnancement se fera quant à elle au moyen du suivi de production.

I.4. Conclusion.

Cette présentation générale du niveau hiérarchique placée dans un contexte de gestion de production, permet de montrer le rôle de chaque module représenté sur la figure I.12 et leur connexion hiérarchisée.

Il est à noter que pour l'instant, la gestion des stocks n'est pas prise en compte, hormis les décisions prises sur la modulation des flux d'entrée.

D'un point de vue général, l'amélioration d'un système industriel vise deux objectifs principaux : d'une part, rapprocher le mieux possible les produits des besoins de la clientèle, c'est-à-dire améliorer la qualité au sens large, et d'autre part, améliorer la productivité. Si l'amélioration de la qualité sort des limites de la gestion de production, en revanche elle est fortement impliquée dans la recherche de l'amélioration de la production. Du fait de la difficulté de recherche de solution optimale, due en grande partie à une combinatoire importante des problèmes à résoudre, les travaux effectués au laboratoire se limitent à la recherche de solutions satisfaisantes.

Le module sur lequel porte notre travail est le module de pilotage. Dans un premier temps, nous proposons d'établir un bilan des différents indéterminismes trouvés lors de l'analyse des graphes de commande pour ensuite définir à quel niveau ils seront levés. Ensuite, nous présenterons une solution pour la conduite liée à la levée d'un certain type d'indéterminisme. Enfin, dans le cadre de l'implantation, nous exposerons une méthode de génération automatique du module de pilotage de l'atelier.

II. Le module de pilotage.

II.1. Introduction.

Rappelons qu'au niveau de CASPAIM, nous considérons que :

- le niveau ordonnancement détermine l'ordre dans lequel les opérations nécessaires à la réalisation de la production vont s'effectuer sur les moyens de production. Notons de plus, que l'ordonnancement est effectué de manière prévisionnelle. Concrètement, le module ordonnancement :
  - donne la priorité à certaines pièces et détermine la séquence d'entrée des pièces,
  - définit les chemins des différentes pièces, en utilisant la flexibilité des produits,

  en assignant à chaque instance des objets, les champs correspondants.

- le niveau pilotage réalise l'ordonnancement choisi en suivant le prévisionnel, et résout les conflits en temps réel.

Face aux indéterminismes, résultants de la prise en compte de la flexibilité, du parallélisme de flux, de l'accès à une ressource partagée, ..., nous avons attribué :

- au niveau ordonnancement :
  - le règlement des indéterminismes apparaissant sur les gammes opératoires par filtrage des objets, (cf. figure II.2),

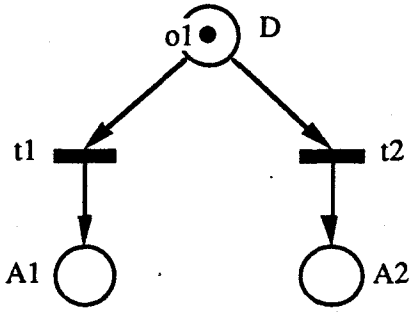


figure II.2

Après ordonnancement, les champs de l'objet o sont référencés avec entre autre sa destination (A1 ou A2). Dans le cas où la destination de o est A1, la transition t2 sera bloquée, l'objet o ne pourra franchir que la transition t1.

  - du fait de la donnée de priorité aux différents objets, l'ordonnancement peut bloquer une transition en prévision du passage d'un objet prioritaire, (cf. figure II.3),

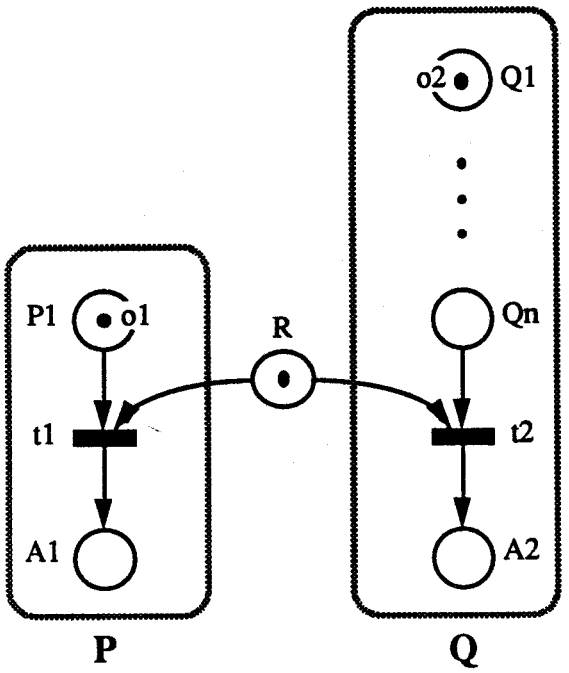


figure II.3

Considérons deux processus P et Q se partageant une même ressource, un robot R par exemple. La ressource R est libre, un objet o1 se trouve dans la place P1, un objet o2 se trouve dans la place Q1, avec o2 plus prioritaire que o1. L'ordonnancement peut alors bloquer monentanémant la transition t1 à l'objet o1 pour laisser l'objet o2 utiliser la ressource R. (Ce cas se présente essentiellement sur des ressources de type convoyeur, quand deux objets se présentent face à une jonction par exemple).

- au niveau pilotage :
  - le règlement des indéterminismes temps réel, notamment accès à une ressource partagée, (cf. figure II.4),

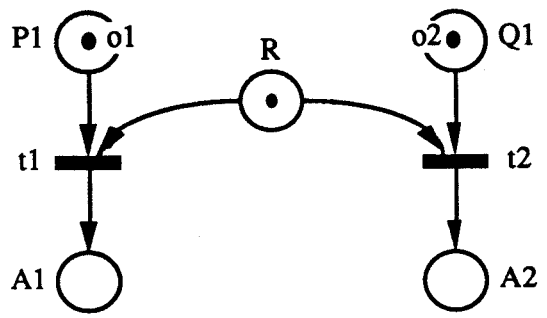


figure II.4

Les objets o1 et o2 se trouvent respectivement dans les places P1 et Q1, et requièrent simultanément une ressource R. L'indéterminisme doit donc être levé.

  - le règlement des conflits non résolus par le niveau ordonnancement.

Avant d'examiner plus en détail le niveau pilotage, effectuons un bilan des différents indéterminismes.

II.2. La classification des indéterminismes.

Etablissons donc la liste des indéterminismes relevés sur les graphes de commande, en suivant la décomposition représentée sur la figure I.21.

Niveau de coordination des processus.

Sur l'exemple de la gamme opératoire présentée figure I.18, un indéterminisme de direction apparaît (cf. figure II.5), dû à l'introduction de la flexibilité des produits (ordre des opérations indifférent pour un produit).

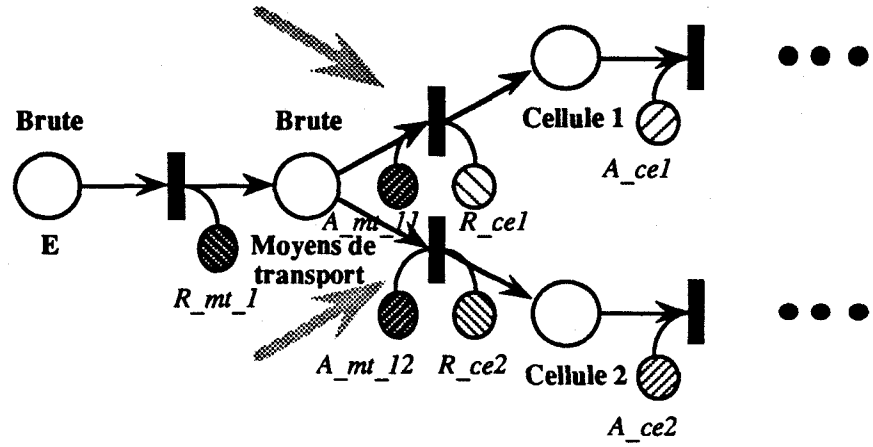


figure II.5.

Ce type d'indéterminisme est levé par le niveau ordonnancement dont la fonction est de déterminer l'ordre d'exécution des opérations et les ressources nécessaires à leur exécution pour chaque produit.

Nous considérons que lors du fonctionnement réel ou simulé, les objets qui transitent dans les gammes opératoires possèdent les données nécessaires à leur bon cheminement. La transition correspondant au choix, sera donc franchissable par cet objet, les autres transitions lui étant interdites.

Niveau de coordination des moyens.

- Considérons dans un premier temps, le modèle de la cellule 2, (cf. figure I.19).

Sur cet exemple, l'indéterminisme qui apparaît (cf. figure II.6), est dû à l'utilisation d'une ressource partagée, le poste d'usinage. En effet, le centre d'usinage de notre exemple possède deux tables d'entrée/sortie F1 et F2, sur lesquelles, à un moment donné, peuvent se trouver respectivement les pièces pℓ et pk. La place R_ce2 contient donc à cet instant, deux objets, obj ℓ et obj k, qui requièrent simultanément un usinage.

Cet usinage ne pouvant s'effectuer en parallèle, une décision doit être prise concernant l'ordre dans lequel les requêtes vont être exécutées, obj ℓ puis obj k, ou obj k puis obj ℓ

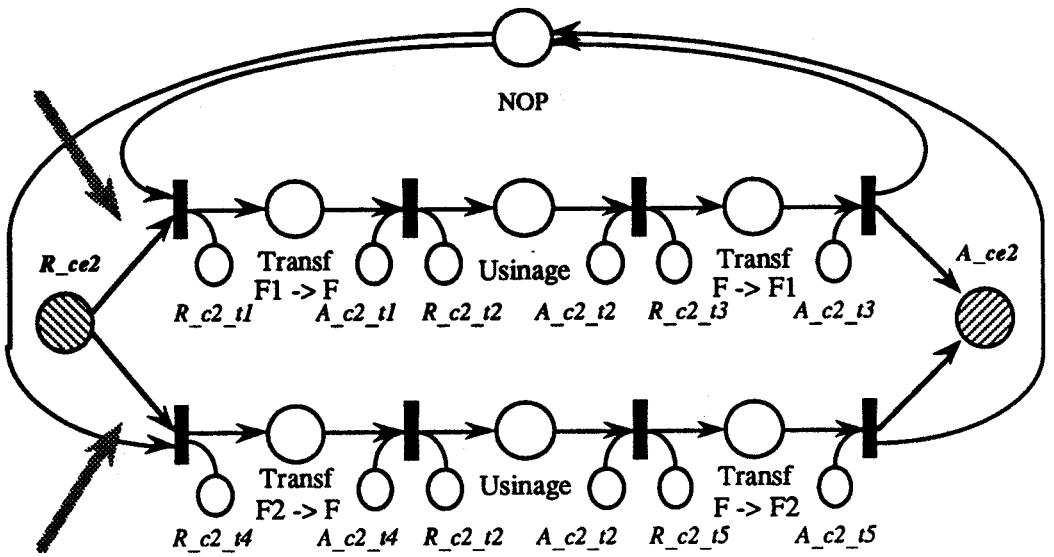


figure II.6.

Nous avons attribué la levée de ce type d'indéterminismes au niveau pilotage. Son rôle est alors de sélectionner la requête à satisfaire. Nous exposerons les moyens d'effectuer cette sélection dans le paragraphe II.3.3.

- Considérons à présent un des modèles du groupement des moyens de transport, (cf. figure II.7).

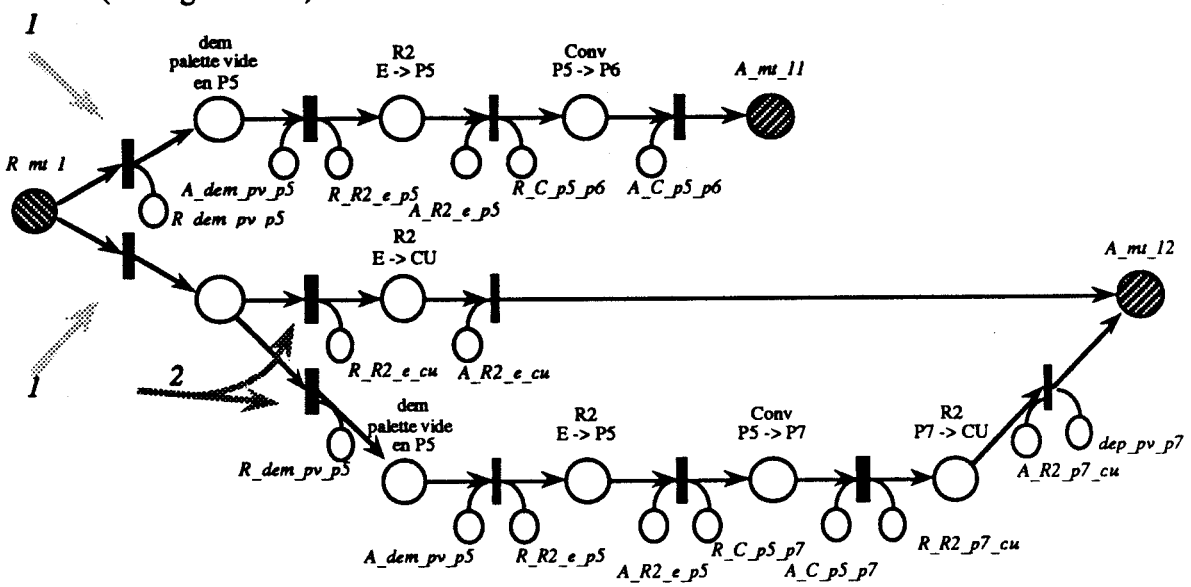


figure II.7.

BU LILLE

Ce modèle correspond à l'acheminement des pièces provenant de l'entrée E vers le tour T ou le centre d'usinage CU. Au même niveau, c'est-à-dire au niveau du groupement des moyens de transport, nous trouvons de la même manière, un modèle qui permet d'exprimer le cheminement des pièces du tour T vers la sortie S, et un modèle correspondant au transfert des pièces du centre d'usinage CU vers la sortie S, (cf. Annexe I.3).

L'indéterminisme 1, relevé sur le graphe de la commande de la figure II.7, correspond en fait à l'indéterminisme concernant la destination des pièces, qui est déjà apparu sur la gamme opératoire. Cet indéterminisme est donc levé de la même manière que précédemment par filtrage des objets au niveau des transitions.

L'indéterminisme 2, quant à lui, concerne le moyen d'atteindre le centre d'usinage CU à partir de l'entrée E, et reflète donc la flexibilité du système de transport. Nous verrons de manière plus précise comment résoudre ce type d'indéterminismes dans le paragraphe II.3.4.

Niveau de la commande effective.

A ce niveau, les indéterminismes n'apparaissent que sur les graphes de commande des moyens de transport, pour l'exemple considéré. En effet, pour les graphes de commande des ressources de transformation, l'unicité de l'exécution des tâches, par exemple les tâches d'usinage et de transfert de F1 vers F (cf. figure I.20), est garantie par le graphe de coordination des moyens. Les indéterminismes qui interviennent donc au niveau de la commande effective, proviennent :

- soit de la flexibilité du système de transport, prise à un niveau plus fin, (cas d'aiguillages, ou cas de dépose, par le Robot 2, d'une pièce sur la table F1 ou F2 du centre d'usinage),
- soit dans le cas de requêtes simultanées à une ressource partagée, (cas de jonctions), lié au flux de pièces.

♦ Cas d'une jonction.

Le graphe de la commande de la jonction Aig2 du convoyeur est donné figure II.8.

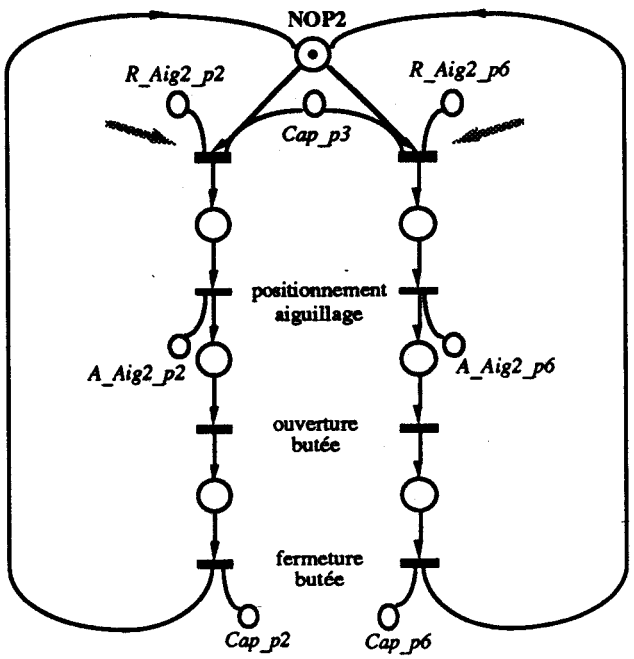


figure II.8.

Cet aiguillage étant pris à talon, il convient de distinguer deux types de fonctionnement :

- Une seule palette demande à être transférée vers la zone P3 depuis la tête de la zone P2 (resp. P6). Il y a alors émission d'une seule requête R_Aig2_p2 (resp. R_Aig2_p6) vers le modèle de la commande (figure II.8).
  Dans ce cas, si la fifo P3 n'est pas pleine, le processus de gestion de la butée Aig2 prend en compte la requête, puis émet l'accusé de réception A_Aig2_p2 (resp. A_Aig2_p6) vers le modèle comportemental du convoyeur (cf. Annexe I.6). Le processus, après avoir commandé la butée, reprend son état initial, et émet une requête Cap_p2 (resp. Cap_p6).

- Deux palettes demandent simultanément à être transférées depuis les zones P2 et P6 vers la zone P3. Il y a alors émission de deux requêtes vers le modèle de la commande.
  Dans ce second cas, il y a indéterminisme sur le choix de la palette à transférer sur la zone P3. Cet indéterminisme sera levé de la même manière que celui relevé au niveau du graphe de la commande de la cellule 2.

♦ Cas d'un aiguillage.

Le graphe de la commande de l'aiguillage Aig1 du convoyeur est donné figure II.9.

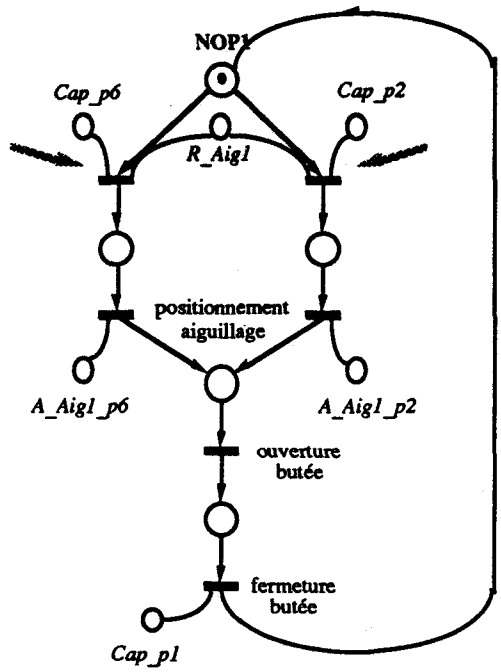


figure II.9.

Cet aiguillage permet de transférer des palettes de la zone P1 à la zone P2 ou à la zone P6. Lorsqu'une palette quitte la fifo P1 pour se placer en tête de fifo (place TP1), il y a émission d'une requête R_Aig1 vers le modèle de la commande. Cette requête est alors prise en charge par le processus de gestion de la butée de l'aiguillage Aig1 du modèle de la commande du convoyeur (cf. figure II.9).

Un indéterminisme directionnel se pose alors. La levée de cet indéterminisme (cf. figure II.9), procède du même raisonnement que pour la levée de l'indéterminisme 2 de la figure II.7, et concerne de la même manière la prise en compte de la flexibilité du système de transport.

Après levée de cet indéterminisme, et si la fifo de la zone P2 (resp. P6) n'est pas pleine, le processus de gestion de la butée Aig1 prend en compte la requête, puis émet un accusé de réception A_Aig1_p2 (resp. A_Aig1_p6) vers le modèle comportemental du convoyeur (cf. Annexe I.6). Le processus, après avoir commander la butée, reprend son état initial, et émet une requête Cap_p1.

Reprenons de façon synthétique et par niveau, les différents indéterminismes qui se présentent :

- Les indéterminismes des gammes opératoires caractéristiques de la flexibilité dans l'ordre de réalisation des transformations.

- L'allocation simultanée de la ressource de production à plusieurs gammes opératoires ou à une même gamme supportant un parallélisme de flux.
  Ce dernier correspond par exemple :
  - à une double requête simultanée du robot R2 : une pièce de type t-f+ souhaite quitter le centre d'usinage CU tandis que symétriquement une pièce t- se trouvant sur le tampon d'entrée E sollicite son transport,
  - à une double requête simultanée du poste d'usinage F : ces deux pièces se trouvant respectivement sur le tampon F1 et sur le tampon F2 du centre d'usinage CU. Cet indéterminisme sera traduit par la présence de deux marques dans la place R_ce2.

- La gestion des indéterminismes locaux, et caractéristiques d'une ressource de production complexe mais flexible.
  Ces indéterminismes résultent de la flexibilité des organes de production. Le convoyeur par exemple en possède de nombreux : flexibilité et donc choix dans le chemin à suivre face à un aiguillage, mais également choix pour déterminer la priorité quand deux palettes se présentent simultanément en regard d'une jonction (conflit d'accès).

En conclusion, trois types de décisions locales doivent être prises pour résoudre ces indéterminismes :

- des décisions résultant de la flexibilité des produits,
- des décisions quant aux choix à prendre face aux conflits d'accès (ressource partagée),
- des décisions de routage résultant de la flexibilité du système de transport.

II.3. La résolution des indéterminismes, [HEB 91].

II.3.1. Introduction.

D'un point de vue fonctionnel, le niveau pilotage comporte trois types de tâches :

- le suivi de l'ordonnancement dynamique, qui permet :
  - d'enregistrer les informations dues à l'évolution du procédé,
  - d'évaluer l'évolution du procédé par rapport aux prévisions,
  - de calculer localement des ordonnancements partiels, [FRE 88],
- la prise de décision, qui permet de régler localement les différents indéterminismes :
  - résolution des conflits d'accès aux ressources,
  - résolution des indéterminismes de "routage",
  - gestion des blocages,
- la réaction aux perturbations, va permettre notamment le passage du mode de marche normal à un mode de marche dégradée du système, [BOI 91].

Représentons d'un point de vue fonctionnel, les différents modules du pilotage (cf. figure II.10), en omettant le module qui permet d'assurer la fonction surveillance.

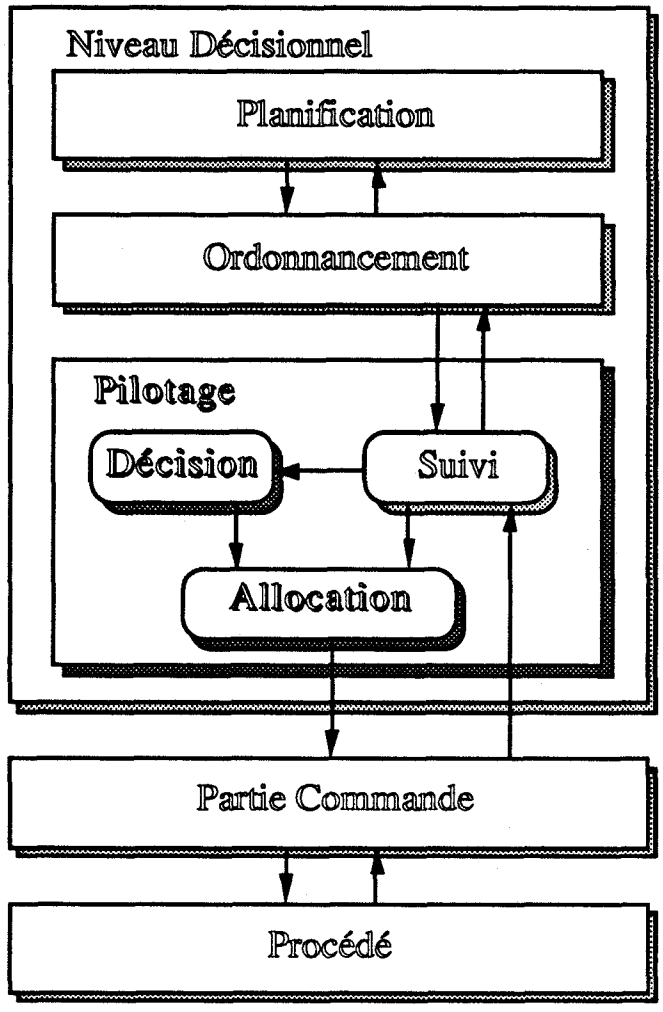


figure II.10

Les modules, objet de notre étude, sont les modules de décision et d'allocation. Nous présentons dans cette partie les moyens mis en œuvre pour effectuer la levée des différents indéterminismes examinés précédemment, puis dans la partie II.4, une première approche de mise en œuvre. Lors de cette présentation, nous ferons apparaître progressivement à quel module est attribuée la prise en compte des indéterminismes, et ainsi nous préciserons ce que nous entendons par suivi, décision, et allocation.

II.3.2. Les décisions dues à la flexibilité produit.

Les décisions à prendre face à un problème de routage de pièce sur les gammes opératoires notamment (cf. figure II.5), peuvent être considérées suivant deux points de vue.

Le premier consiste à considérer que le niveau ordonnancement a déjà défini de manière explicite et fine sur quel moyen de production les différentes opérations doivent être exécutées pour chaque pièces circulant dans le système.

Dans ce cas, la prise de décision se borne à l'exécution des prévisions établies par le niveau d'ordonnancement, par filtrage des objets. Ce filtrage consiste à tester dans un premier temps les champs des objets, puis à bloquer ponctuellement les transitions non franchissables par cet objet.

Le deuxième point de vue consiste à considérer que le niveau d'ordonnancement définit de manière globale sur quel moyen de production les opérations doivent être effectuées, et ne considère que le premier niveau de la décomposition en groupements, (cf. figure I.15).

La résolution de cet ordonnancement partiel est attribuée au module appelé suivi de l'ordonnancement; cet ordonnancement peut être effectué a priori ou en temps réel. La décision se borne là encore à l'exécution des prévisions.

Nous considérons donc que pour des indéterminismes provenant de la flexibilité produit :

- le module "suivi", réalise l'ordonnancement partiel suivant l'optique choisie,
- le module "décision", n'a à prendre aucune décision effective,
- le module "allocation", réalise le choix préétabli par l'ordonnancement, par une allocation en temps réel des ressources sélectionnées pour chaque objet.

II.3.3. Les décisions face aux conflits d'accès.

Ce type de décision est à prendre lors de l'arrivée de requêtes simultanées demandant l'accès à une même ressource, le terme ressource étant considéré de manière générale. Ce terme peut recouvrir, soit une ressource de production (un robot par exemple), soit une partie d'une ressource de production complexe, (le poste d'usinage du centre d'usinage, une jonction du convoyeur, ...). Rappelons de plus que ces conflits sont dus principalement, soit au parallélisme de flux, soit à la flexibilité du système de transport.

La décision relève donc du choix à effectuer entre ces requêtes simultanées, tout en respectant la prévision qui a été établie par l'ordonnancement provenant du niveau d'ordonnancement, ou du module de suivi. La décision semble donc être prise implicitement par l'ordonnancement, et équivaudrait en fait au test du champ correspondant à la priorité des pièces (date due, par exemple) demandant la ressource, le critère de choix étant a priori de privilégier la pièce de priorité la plus élevée. Mais ce critère n'est évidemment pas unique, et doit correspondre à la stratégie de pilotage de l'atelier.

Du fait de cette non unicité de critère, la décision à prendre doit donc pouvoir être explicitée de façon indépendante et peut être même interactive en phase d'exploitation. Le niveau de pilotage doit donc permettre la réalisation de ce type de conflit. Nous attribuons cette résolution au module de décision.

Pour effectuer sa prise de décision, nous considérons que :

- ce module possède des fonctions de résolution permettant de répondre aux différents critères,
- la sélection de la fonction à appliquer est effectuée par le module de suivi,
- une fois la décision prise, le module de décision transmet le choix au module d'allocation qui va attribuer effectivement la ressource à l'objet sélectionné.

La procédure de sélection que nous avons implantée, correspond pour l'instant à une sélection sur les requêtes véhiculant des jetons prioritaires. Indiquons toutefois, que l'on envisage de pouvoir sélectionner différentes fonctions de sélection, et ainsi permettre leur modification dynamique.

Remarquons de plus, que ces cas de conflits sont très peu probables, notamment en ce qui concerne les conflits d'accès à la ressource poste d'usinage.

II.3.4. Les décisions dues à la flexibilité du système de transport.

II.3.4.1. Introduction.

Nous considérons que la résolution de ce type de problèmes est assimilée à la résolution des stratégies de routage des pièces, palettes, ..., dans l'atelier.

Les problèmes liés aux stratégies de routage sont relatifs aux choix statique ou dynamique des chemins, à l'exploitation de ces chemins, et au contrôle de flux, [RAY 85].

De la même manière que précédemment, on pourrait penser que l'ordonnancement établit implicitement le routage des différents objets circulant dans le système lors de son évaluation. Cependant, face à la complexité et surtout à la combinatoire importante de ces problèmes, l'ordonnancement ne peut établir finement le routage de tous les éléments.

En effet, les modèles utilisés pour élaborer l'ordonnancement sont d'un faible niveau de détail. Ils prennent parfois en compte l'allocation des ressources, bien que ce soit rarement le cas dans le contexte d'une réelle flexibilité de production. Ils ne prennent jamais en compte l'état réel et le comportement des ressources. Il apparaît à l'évidence de réelles différences entre le comportement réel et le comportement souhaité vu au niveau du modèle d'ordonnancement prédictif.

Devant un indéterminisme directionnel, il faudra donc pouvoir prendre localement et en temps réel la décision d'un chemin optimal (au sens de suivi de la stratégie) à suivre par exemple par une palette qui se trouve face à un aiguillage, pour atteindre sa destination.

Nous attribuons la prise en charge de cette décision au module de décision. Une fois la décision prise, ce module en informe le module d'allocation.

Voyons à présent la proposition que nous formulons face à ce type de problème.

II.3.4.2. Le routage dans les réseaux informatiques.

Effectuons tout d'abord, une rapide digression dans le domaine de la téléinformatique en présentant le routage d'informations dans les circuits maillés à

commutation par paquets, problème proche du problème de routage des palettes sur un convoyeur.

Un réseau à commutation par paquets (cf. figure II.11), est constitué d'un ensemble de nœuds de commutation ou commutateurs (C1, C2, ...), reliés entre eux par des liaisons de données, formant ainsi un réseau généralement maillé [MAC 83], [PUJ 84].

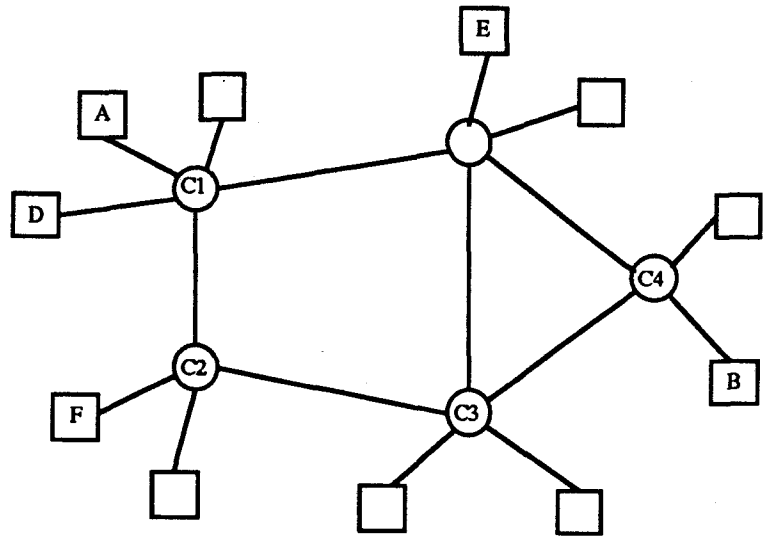


figure II.11

Ce réseau assure pour ces usagers (A, B, ...), le transport des blocs d'informations, par l'intermédiaire des commutateurs, qui déterminent progressivement le chemin que doit emprunter le paquet d'informations. Cette opération s'appelle l'acheminement ou le routage du paquet.

Cette activité d'acheminement est généralement décomposée en deux activités superposées : la commutation et l'adaptation des chemins.

La commutation : Au moment de décider de la liaison suivante que doit emprunter le paquet, chaque commutateur consulte une table dite d'acheminement, dans laquelle pour chaque destination il trouve l'identification de la liaison suivante. Ces mécanismes de commutation sont assemblés en cascade. L'ensemble des tables d'acheminement déterminent un chemin pour le transfert des informations entre chaque point du réseau.

L'adaptation des chemins : Cette activité superposée à la commutation est responsable de la mise à jour des tables d'acheminement. Elle consiste à choisir pour

chaque paire de points du réseau, un chemin parmi tous les chemins possibles. Le premier critère est le bon fonctionnement des éléments du chemin, l'adaptation permettant de contourner les commutateurs ou liaisons en pannes. Parmi les chemins restant, différents critères sont utilisés et éventuellement combinés, chemin le plus court en nombre de liaisons, chemin le plus rapide par rapport à la vitesse de liaison ou de la charge, ...

II.3.4.3. Méthode de résolution des indéterminismes directionnels.

Par analogie à ce mécanisme de transport d'informations dans les réseaux maillés, nous attribuons les fonctions de commutation au module de suivi, d'adaptation, au module de décision.

En effet, nous considérons que l'ordonnancement, effectué par le niveau ordonnancement ou plus finement par le module de suivi, permet d'établir un fléchage quant aux chemins à suivre par les différents éléments circulant dans le système. Cependant, ce fléchage, (par exemple : considérons que la pièce t+f-, venant d'être palettisée sur le poste de palettisation/dépalettisation du tour, doit être usinée sur le centre d'usinage) si il est suffisant dans le cas d'un système de transport simple (cf. figure I.11), peut s'avérer insuffisant dans le cas d'un système de transport complexe (cf. figure II.12).

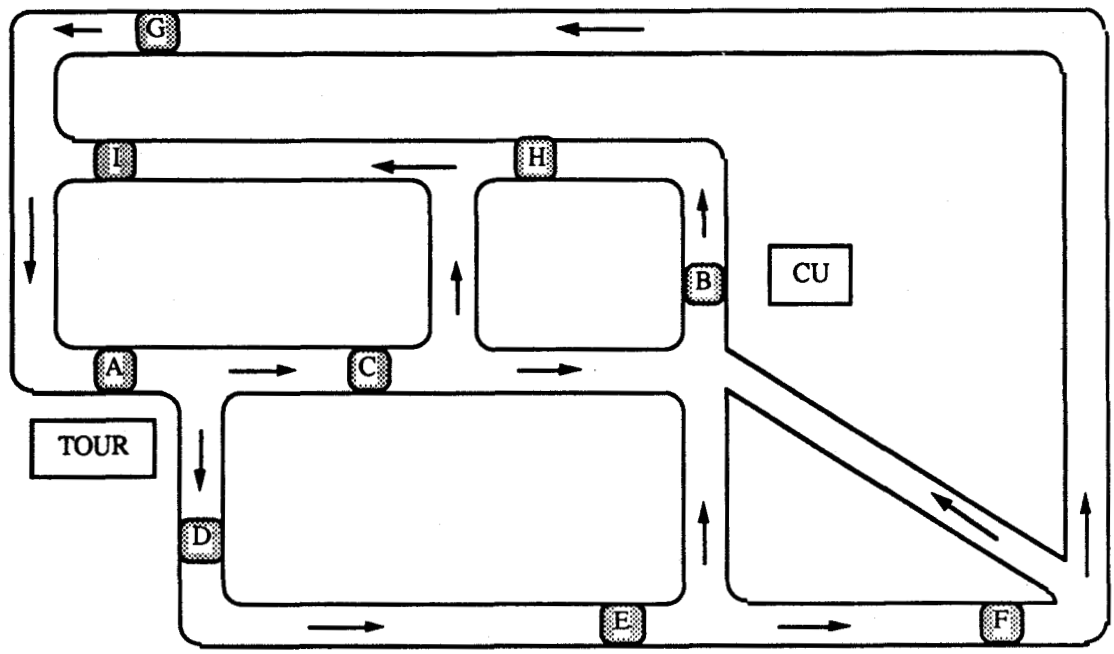


figure II.12

II.3.4.4. La fonction de coût semi-statique.

Cette fonction calcule pour tous les chemins possibles menant à la destination désirée, le taux d'occupation en nombre de palettes des postes se trouvant sur les chemins, au moment où la décision est prise, ainsi que la longueur des chemins en terme de capacité.

Pour tous les chemins, le calcul est donc le suivant :

- calcul de la longueur du chemin : ℓ_c
- calcul de son taux d'occupation : o_c

Prenons l'exemple de la configuration suivante (cf. figure II.14), nous considérons que la palette à aiguiller se trouve en tête du poste P1 et a comme destination la tête du poste P7.

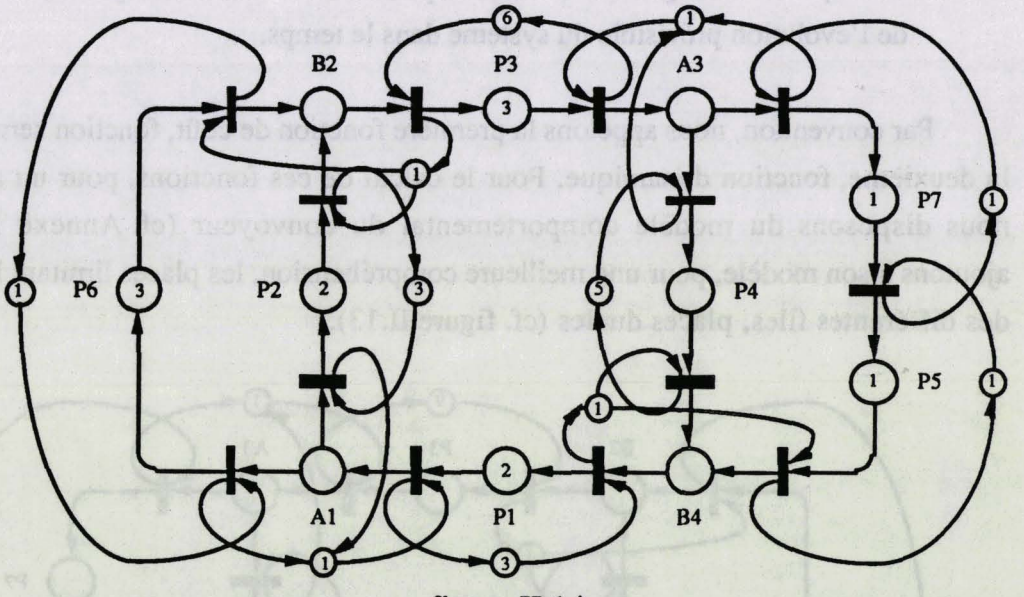


figure II.14

En considérant que les bouclages (ou circuits) sont interdits dans la prise de décision, les deux chemins possibles pour amener cette palette de P1 vers P7 sont respectivement, c1 = P1 A1 P6 B2 P3 A3 P7 et c2 = P1 A1 P2 B2 P3 A3 P7. Les résultats de la fonction de coût semi-statique pour les chemins c1 et c2, sont respectivement : $\ell_{c1} = 18$, $o_{c1} = 7$, $\ell_{c2} = 19$, $o_{c2} = 6$, (nous ne prenons pas en compte, pour le calcul de cette fonction, le premier poste du chemin, du fait de la présence en tête de ce poste de la palette à diriger).

C'est pourquoi, le rôle attribué au module de décision, va être de router les différentes palettes du convoyeur notamment, et pas seulement d'adapter le routage en cas de panne. Les décisions prises dans ce cas, vont permettre de lever les indéterminismes provenant de la flexibilité de transport (cf. figure II.7) et notamment ceux provenant des aiguillages (cf. figure II.9).

Examinons à présent, comment s'effectue la prise de décision pour une palette, connaissant sa destination.

Le choix d'aiguillage s'effectue à partir d'une fonction de coût appliquée aux différents chemins possibles. Cette fonction peut être plus ou moins fine. Par exemple , elle peut tenir compte :

- du taux d'occupation des différents postes au moment où la décision est prise, ainsi que de la longueur des chemins possibles en terme de capacité,
- de l'évolution prévisible du système dans le temps.

Par convention, nous appelons la première fonction de coût, fonction semi-statique, la deuxième, fonction dynamique. Pour le calcul de ces fonctions, pour un aiguillage, nous disposons du modèle comportemental du convoyeur (cf. Annexe I.6), nous ajoutons à son modèle, pour une meilleure compréhension, les places limitant la capacité des différentes files, places duales (cf. figure II.13).

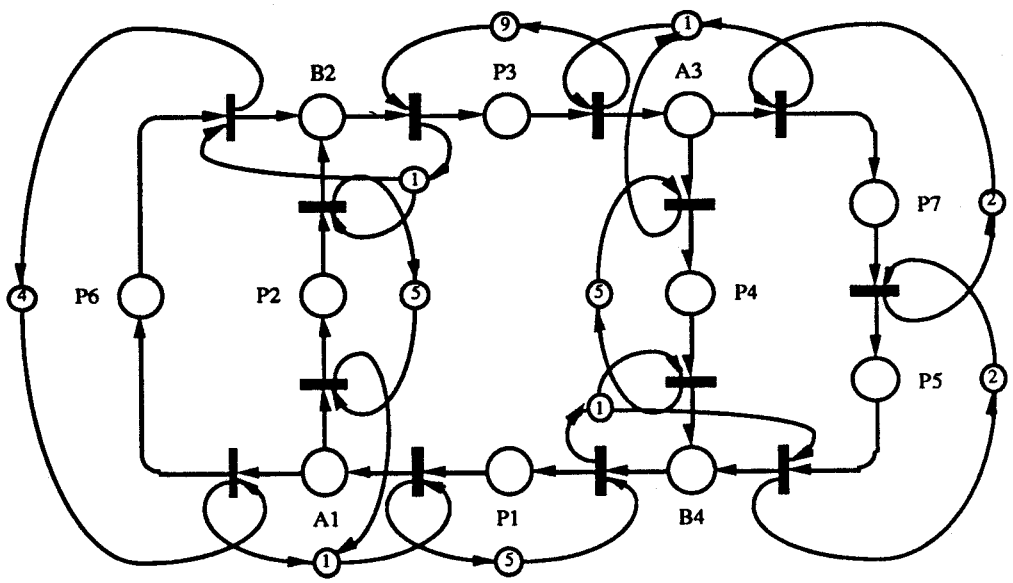


figure II.13

Le chemin emprunté par la palette sera le chemin le plus court en terme de capacité, possédant le plus petit taux d'occupation, afin d'éviter a priori les files saturées. Le chemin sélectionné pour cet exemple, sera le chemin c2.

Il est à noter que ce choix est effectué à un instant donné, et que le chemin n'est pas mémorisé, c'est-à-dire, que la décision ne porte d'une part que sur l'aiguillage concerné (la décision donne en effet, aller en P2, et non pas suivre le chemin c2), et d'autre part, qu'il peut être modifié lors d'une autre prise de décision sur un autre aiguillage (sur l'aiguillage A3 par exemple).

Cette sélection simple et locale est donc basée sur le taux d'occupation des différents postes d'un convoyeur, au moment de la prise de décision. Elle ne tient donc pas compte de l'évolution du système, et notamment des blocages temporaires de certaines palettes devant un poste d'usinage (cas de palettes qui attendent devant le poste d'usinage, que la pièce après usinage soit repalettisée).

II.3.4.5. La fonction de coût dynamique.

Au vue de la remarque précédente, si la décision doit être plus fine, et tenir compte notamment du temps d'attente devant un poste, nous sommes alors conduit à prédire l'évolution du système, notamment par simulation.

Pour pallier ce type de problème, la fonction de coût dynamique permet d'évaluer par simulation, pour chaque chemin possible, le temps mis par la palette pour atteindre sa destination, en tenant compte :

- du taux d'occupation du chemin,
- des blocages à éviter,
- de la longueur du chemin,
- du temps que vont mettre les palettes pour sortir de la zone.

Pour la réalisation du calcul de cette fonction, nous avons réalisé une maquette, que nous proposons de décrire brièvement, (le listing de ce programme est fourni en Annexe II.1). Les différentes contraintes que nous nous sommes efforcées de respecter sont donc :

- tenir compte des capacité de chaque poste, ce qui permet de déterminer la longueur du chemin,
- tenir compte des ressources critiques : jonctions,

- tenir compte des circuits, afin d'éviter les blocages,
- interdire à une palette de repasser par un poste, dans une optique du plus court chemin,
- tenir compte des palettes qui se trouvent déjà sur le chemin emprunté par la palette pour laquelle nous voulons prendre une décision, c'est-à-dire, de prendre en charge l'évolution de toutes les palettes du système,
- tenir compte des blocages temporaires d'une palette devant un poste d'usinage.

Les caractéristiques du réseau sont données sous la forme de deux fonctions :

- la description du réseau est donnée par une fonction **successeur** :
  successeur (Poste , Poste_successeur)
- la description des différentes capacités des postes est donnée par la fonction **capacité** : capacité (Poste , Capa)

La description de l'état du système au moment de la prise de décision est donnée par l'**état_initial** présenté sous la forme Backus-Naur :

Etat_initial ::= Liste_de_couples_1
Liste_de_couples_1 ::= (Poste , Liste_de_couples_2)
Liste_de_couples_2 ::= (Palette_dans_le_poste , Numéro_place)

La fonction de calcul, appelée **parcours**, permet :

- de trouver tous les chemins possibles pour que la palette pour laquelle on veut prendre une décision aille d'un poste de départ à un poste d'arrivée (en tête du poste d'arrivée),
- pour chaque chemin, de calculer au fur et à mesure le coût, tout en faisant évoluer toutes les palettes du système.

La construction du parcours s'établit pas à pas, et correspond à un avancement incrémental de toutes les palettes. Nous appelons **évolution** cette fonction. Pour effectuer l'évolution, nous prenons en considération les contraintes énoncées précédemment. Par exemple, si une palette a pu avancer, nous effectuons un test pour savoir si elle n'est pas déjà passée par ce poste, si oui, elle recule; une palette dont la destination est un poste d'usinage ne pourra évoluer qu'au bout d'un temps supplémentaire prédéfini; ...

A la fin de la simulation, c'est-à-dire à la fin du parcours de l'arborescence de résolution, la fonction parcours élabore le calcul du coût pour tous les chemins possibles.

Il est à noter qu'il est possible que la fonction de coût procure plusieurs résultats différents pour un même chemin. Ceci découle du fait que, pour un même chemin emprunté par la palette à diriger, l'évolution des autres palettes du système peut être différente.

Une fois la fonction de coût calculée pour tous les chemins possibles, la décision d'aiguillage sera prise pour le chemin qui réalise :

- pour les chemins identiques, l'optimisation des résultats de la fonction de calcul,
- pour les chemins différents, prendre la minimisation des maximum de chacun de ces chemins.

II.3.4.6. Conclusion.

Face à ces deux résultats, le choix de la fonction de coût dépend bien évidemment du niveau d'optimisation que l'on veut obtenir. Il est bien sûr envisageable d'établir de nouvelles fonctions de coût répondant à d'autres critères. Cependant, ces fonctions doivent être construites dans une optique d'optimisation, et de résolution locale. Elles permettent notamment :

- de fournir une décision vis à vis de la stratégie à appliquer,
- d'effectuer les calculs le plus rapidement possible,
- d'être facilement implantables.

Remarquons de plus, que pour répondre à l'indéterminisme directionnel de la figure II.7, deux optiques différentes peuvent être envisagées.

La première consiste à considérer que le choix, de la même manière que pour la flexibilité produit, est déjà établi par l'ordonnancement.

La deuxième consiste, quant à elle, à considérer qu'il faut effectivement résoudre cet indéterminisme. Pour la résolution de ce type d'indéterminismes, nous avons envisagé d'utiliser la même méthode que précédemment (calculer une fonction de coût pour tous les chemins envisageables), mais en utilisant un modèle spécifique plus général. Ce modèle est un modèle comportemental de plus haut niveau, intégrant les différentes ressources de transport participant à la flexibilité de transport, (cf. Annexe II.2).

II.4. Conclusion.

Un grand nombre de méthodes sont développées pour résoudre les problèmes de pilotage et de suivi temps réel de la production : méthodes analytiques [FRE 88], méthodes posant le

problème de routage en termes de commande optimale [CHA 91], ... Cependant, toutes ces méthodes ne s'intéressent en général qu'au pilotage global d'une cellule, ou d'un atelier, et résolvent les différents problèmes à ce niveau.

La décomposition du niveau de pilotage, proposée ici en trois modules suivi - décision - allocation, permet en revanche de traiter et de résoudre les différents problèmes d'indéterminismes à un niveau proche du modèle de commande, permettant ainsi de limiter notamment les temps de calcul, en facilitant une approche temps réel de la prise de décision.

Il est à noter que la hiérarchisation des problèmes de pilotage est facilitée par l'emploi de la décomposition des différents modèles de la commande. Notons à ce titre, qu'une décision émanant d'un niveau haut peut préempter une décision d'un niveau plus bas (cas figure II.3).

Conclusion.

Au vu de la décomposition du niveau hiérarchique, les tâches affectées aux différents niveaux ont été définies :

L'ordonnancement fournit l'ordre des opérations nécessaires à la réalisation de la production, et l'affectation correspondantes des différents moyens de production.

Le pilotage tend à approcher au mieux l'ordonnancement précalculé, et résoud les indéterminismes et conflits en temps réel.

En phase de simulation ou d'exploitation temps réel du pilotage, il est donc important de disposer d'une vue synthétique, sous forme de synoptique, permettant de suivre l'évolution et les réactions du système.

C'est pourquoi, nous aborderons dans le chapitre III, le problème de la réalisation d'une interface graphique permettant la visualisation de l'évolution du système.

Chapitre III

Introduction.

Nous avons maintenant défini le projet CASPAIM et les modèles qui sont utilisés à chaque étape de la méthodologie. Le but du chapitre III est de montrer le rôle important d'une interface graphique dans la chaîne de conception du projet CASPAIM ainsi que les développements qui sont en cours autour de cet outil.

La première partie mettra en évidence l'utilité de la création d'une interface graphique, et nous donnerons les aspects généraux de ce type d'interface.

Dans une deuxième partie, nous nous intéresserons aux modèles que nous avons définis pour la création de cette interface graphique, qui permet de représenter les différents graphes du projet CASPAIM1.

Dans la troisième partie, nous traiterons d'un problème particulier rencontré lors de la représentation de graphes, le positionnement automatique des différentes places d'un réseau de Petri ou des différents états d'un automate à états finis. Avant la description détaillée de cette méthode, nous décrirons, le fonctionnement général de l'interface graphique réalisée.

Enfin, nous présenterons les extensions en cours de réalisation et les différentes possibilités d'ouverture autour de ce noyau central.

I. Présentation générale.

I.1. Introduction.

Nous avons montré dans les deux chapitres précédents comment étaient obtenus les différents modèles réseaux de Petri de la méthodologie CASPAIM. Que ce soit au niveau spécification, modélisation, simulation ou implémentation, ces différents modèles sont donnés sous une forme littérale. Ces modèles étant des modèles graphiques par excellence, nous avons ressenti le besoin de concevoir une interface graphique permettant de visualiser ces graphes à tous les niveaux du projet CASPAIM.

Avant de rentrer dans la présentation détaillée de l'interface graphique, exposons d'une manière générale l'intérêt de cet outil.

I.2. Intérêt d'une interface graphique.

Ce qui nous a conduit à réaliser cette interface graphique est que l'utilisation de cet outil est en liaison directe avec la simulation du routage que nous venons de décrire dans le chapitre II.

D'une part, il y avait un manque énorme au niveau de l'exploitation graphique, de manière générale, des modèles de CASPAIM1, notamment lors de la phase de simulation du système conçu, et en particulier, lors de la simulation du routage. En effet, nous trouvons qu'il est frustrant d'utiliser dans la théorie, des modèles ayant une telle aptitude à la modélisation d'objets aussi complexes que divers, et que, dans la réalité de la simulation ou de l'implantation réelle sur site, ils ne soient exprimés que sous une forme littérale.

L'intérêt principal d'une représentation graphique par rapport à une description littérale, est qu'un graphe apporte une vue globale de ce que le concepteur ou l'utilisateur veulent décrire. En effet, un des inconvénients le plus sérieux réside dans le principe même du maniement des données sous forme de mots, de phrases. Ceci rend long et fastidieux l'examen d'un grand nombre de données défilant à la suite les unes des autres. Il faut de plus, pour les manipuler, faire l'apprentissage du formalisme de représentation des connaissances choisi et du langage adapté à l'implantation de ces structures. Enfin, ces systèmes ne mettent pas à profit l'extraordinaire faculté que possède l'homme d'effectuer visuellement un repérage rapide, dans un espace à agencement complexe, d'un objet ou d'un ensemble d'objets qui lui semble particulièrement intéressant [BOL 83]. Par exemple, dans le contexte qui nous intéresse, s'il est difficile de localiser un indéterminisme, lors de la phase de simulation par exemple, dans

une cellule flexible, si le message est "indéterminisme : transitions concernées : PRO/2 - CONS/2", à moins de connaître parfaitement tous les libellés, connexions et routages potentiels des pièces, et surtout l'état dans lequel se trouve le système afin de faire un choix entre telle ou telle transition, il semble difficile à un opérateur de voir où se trouve effectivement le problème sans un synoptique décrivant l'évolution du modèle concerné.

D'autre part, les réflexions sur les problèmes de routage et de prise de décision, que nous avons rencontrés, nous ont conduit à développer une méthode de positionnement automatique des places d'un réseau de Petri. Ce problème apparemment purement graphique, se trouve être un problème réel pour l'élaboration d'un atelier flexible de production. En effet, pourquoi ne pas essayer de minimiser certains chemins qu'emprunteront les différentes pièces, dès l'étape de conception, en essayant de positionner de manière optimale les éléments qui composent cet atelier, ou dans le cadre de la modélisation, pourquoi ne pas essayer d'obtenir un graphe clair et lisible permettant l'utilisation des modèles avec un maximum d'efficacité?

C'est pourquoi, nous avons développé une interface graphique et une méthode de positionnement automatique des places d'un réseau de Petri, utilisées notamment, à l'heure actuelle, en phase de conception, pour la simulation de la méthodologie CASPAIM1.

Avant de présenter de manière générale, ce qu'est une interface graphique, examinons ses différents apports.

I.3. Apports d'une interface graphique.

Un des aspects essentiels d'une interface graphique, est la **convivialité** qui en découle. L'image est d'abord un moyen de représenter de manière concise, efficace, analysable d'un coup d'œil, des résultats numériques multiples ou complexes. Il n'y a d'ailleurs là rien de nouveau : depuis longtemps, les scientifiques, les ingénieurs, les gestionnaires présentent leurs résultats sous forme de tableaux, de courbes, de diagrammes, d'histogrammes [COU 83]. Pourtant depuis peu, l'innovation provient de la manière dont les représentations graphiques sont obtenues : les images peuvent être synthétisées directement par un système informatique, ... Maintenant, quelles sont les équipes de recherche et de développement dans les usines de construction d'automobiles par exemple, qui ne disposent pas d'outil de CAO? L'imagerie, l'infographie sont d'ailleurs des termes nouveaux qui appartiennent déjà au vocabulaire courant. De manière plus proche du domaine de l'informatique industrielle, le Grafcet, [BLA 79], est un outil graphique employé depuis une quinzaine d'années par les industriels. Cet outil permet de décrire ainsi, de manière simple et générale, le séquencement de

l'application à concevoir, et ensuite de décrire les actions à l'aide d'un langage de bas niveau. Les applications peuvent donc être directement programmées en Grafcet sur certains automates. Par ailleurs, il existe, maintenant, des logiciels permettant de développer les applications en Grafcet sur un matériel informatique, puis de les transférer sur différents types d'automates programmables cibles.

Un autre aspect des interfaces graphiques, est l'**interactivité** à laquelle elles contribuent. Nous n'affirmons pas par cette assertion que les logiciels non graphiques n'offrent pas aussi une certaine interactivité, mais nous trouvons qu'il est plus aisé de sélectionner une commande par le déroulement d'un menu ou par un "clic" sur une icône à l'aide d'une souris, plutôt que de taper une suite de commandes sur le clavier. C'est Alan Kay, qui en 1972, décide d'incorporer un certain nombre de dispositifs annexes, tels que la souris, les écrans bit-map haute résolution, ..., qui se sont démocratisés ensuite par l'intermédiaire de la micro. L'utilisation de ces dispositifs a permis ainsi d'obtenir une réelle interactivité, aussi bien au niveau du système d'exploitation, qu'au niveau de la programmation.

C'est pourquoi, il nous semble important d'introduire pour toutes les étapes de la méthodologie CASPAIM, ce type d'outil graphique convivial et interactif, c'est-à-dire, de permettre d'accéder facilement au système informatique, et d'utiliser un mode conversationnel.

I.4. Structure générale d'une interface graphique.

Pour développer cette partie sur la structure générale d'une interface graphique, nous nous appuyons sur un livre "la réalisation des logiciels graphiques interactifs", [LUC 82], dans lequel sont présentés les principes généraux de l'infographie interactive.

Les images engendrées par ordinateur appartiennent à différentes catégories :

- des images non figuratives utilisées pour des recherches plastiques,
- des images figuratives, moyen d'expression privilégié des techniques de conception assistée par ordinateur, en architecture, génie civil, construction automobile, navale ou aéronautique,
- des graphiques, qui permettent de représenter rapidement les relations liant des informations plus ou moins diverses,
- des images animées, soit dans le cadre du contrôle de processus en temps réel, soit pour l'aide à la conception de dessins animés.

Au vue de cette classification, précisons que dans le cadre de ce mémoire, nous nous

intéressons aux graphiques qui deviennent par la suite des images animées pour la simulation temps réel de la commande d'un atelier flexible de production.

Le problème de la représentation graphique consiste à effectuer une correspondance entre un ensemble d'informations et les différents éléments d'un langage graphique, ce langage possédant un vocabulaire et une grammaire propres. Composer une image revient à choisir un langage graphique, à analyser l'information à transcrire pour trouver les correspondances avec les éléments du langage, et enfin, à organiser les éléments en image.

L'infographie interactive se caractérise par le fait que les images présentées sont des images vivantes susceptibles de modifications. L'image présentée sur l'écran est le reflet d'une situation d'un programme à un instant donné, situation modifiable au gré de l'utilisateur. Il parait évident que si la situation change, l'image qui la représente doit changer également, ce qui implique que l'image produite est destinée à être étudiée, en général, pendant un temps bref, qu'elle doit être produite rapidement afin d'éviter une attente trop longue de l'utilisateur, et enfin, que l'image présentée sur l'écran n'est souvent qu'une ébauche permettant de vérifier différentes caractéristiques, la représentation finale étant obtenue en différé.

Pour permettre de préciser les modifications de situations souhaitées, l'opérateur devra disposer d'outils pour entamer un véritable dialogue avec la machine. Nous l'avons dit dans la partie précédente, l'interactivité n'est pas l'apanage des interfaces graphiques, mais le dialogue par l'intermédiaire de fonctions graphiques semble plus simple, et permet de garder une certaine cohérence quant à l'utilisation de l'environnement de travail.

Les fonctions considérées comme élément de base pour la construction du dialogue homme/machine sont :

- le menu, qui permet la sélection d'une action à effectuer dans une liste de commandes,
- l'introduction de valeurs alphanumériques, permettant l'affectation d'une valeur à un paramètre,
- l'identification des différents composants à représenter, permettant d'indiquer le traitement à appliquer, (structure de données),
- et enfin, l'acquisition d'un élément d'image, permettant la construction des représentation destinées à l'ordinateur, (structure de données graphiques).

Il est bien entendu possible, de combiner ces quatre fonctions de base pour définir des actions plus complexes.

Un autre point essentiel de la communication graphique, est la structuration de la représentation. Le dessin ou l'image doivent pouvoir être décomposés en éléments manipulables séparément.

Les paramètres de définition des éléments de base de la transcription graphique sont :

- la forme des éléments, définie par son contour (suite de points),
- l'emplacement sur la surface de dessin repéré, par les coordonnées d'un point particulier appartenant à l'élément dans un repère absolu ou relatif,
- la taille de la surface occupée sur le plan du dessin,
- l'orientation,
- le grain, c'est-à-dire le nombre d'éléments séparables pour une entité de surface donnée,
- l'intensité de gris,
- et la couleur, réelle ou correspondant à un choix arbitraire.

Remarquons toutefois, que tous ces paramètres ne sont pas forcément utilisés pour la définition des éléments graphiques, la couleur ou le grain notamment, dépendent essentiellement du type de machines et d'écrans sur lequel sera implanté le logiciel graphique.

Ces éléments de base sont pour les objets graphiques (cercle, rectangle, ligne, ...), qui eux-même constituent un dessin ou une image, ce que sont les lettres et les chiffres pour les mots d'un texte.

Maintenant que nous avons présenté brièvement les éléments qui composent une interface graphique, donnons le schéma général de la chaîne de programmes nécessaires à l'établissement d'un tel outil.

En général, la description au plus bas niveau d'un dessin, est composée d'une liste de points. La vocation de la plupart des programmes d'application n'est pas de traiter de tels éléments. Les données manipulées sont adaptées au type de calcul qu'ils ont à effectuer, mais pas sous forme de liste de points a priori. Il faudra donc, qu'à un moment donné, une transcription des données en éléments graphiques ait lieu. Pour cela, nous pouvons dire qu'une séquence de programmes sera nécessaire. Les termes employés dans [LED 78], sont les suivants :

- un **logiciel d'application**, qui traite des données qui lui sont propres,
- un **logiciel de description**, dont le rôle est d'extraire de la structure de données de l'application, les éléments nécessaires à la codification des éléments qui permettront d'obtenir une représentation graphique, qui sera obtenue plus tard,
- un **logiciel de préparation** à la visualisation, dont le rôle est la création d'une

ébauche du dessin à partir de la scène codée, passage de la scène au fichier graphique,

- un **logiciel élémentaire**, dont le rôle est d'assurer la gestion de l'écran et qui permet d'obtenir le dessin final.

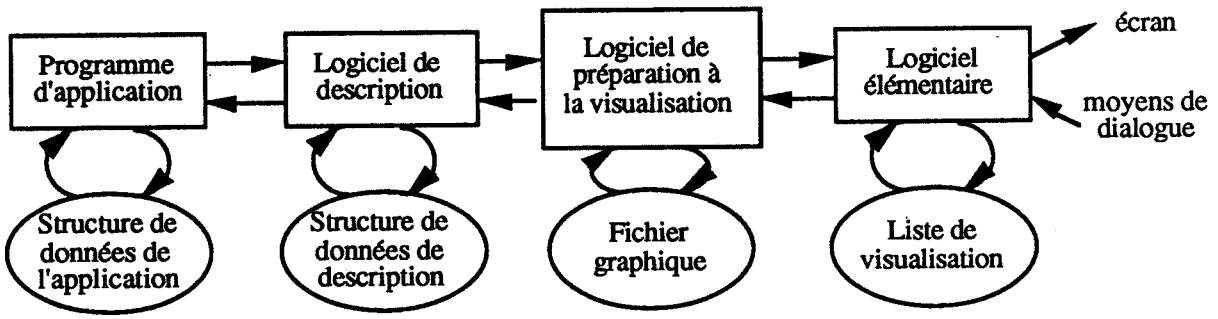


figure III.1

Nous remarquons que cette chaîne de programmes, peut être divisée en fait en deux parties : une partie description et décodage fortement dépendants de l'application, et une partie mise en page totalement indépendante.

Cette remarque nous permet de présenter de manière claire la séparation qui existe entre ce qui dépend de l'application et ce qui dépend du matériel utilisé. Nous distinguons donc :

le **logiciel de mise en forme**, qui permet de passer des informations de la structure de données aux informations graphiques,

et, le **logiciel graphique de base**, qui permet l'affichage sur l'écran de la représentation des informations qui lui sont transmises et de recueillir les données graphiques destinées au programme d'application via le logiciel de mise en forme.

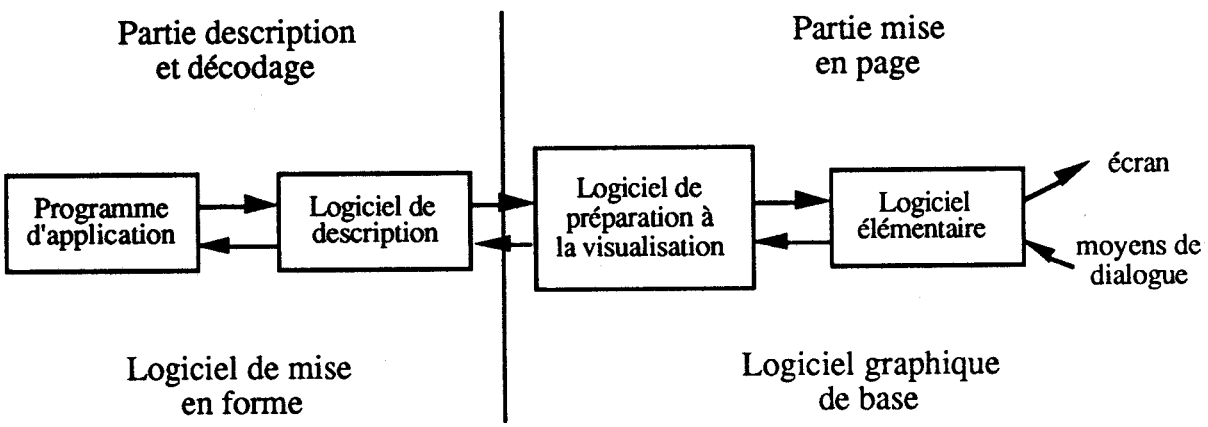


figure III.2

Le logiciel graphique de base a donc, la responsabilité d'assurer l'interface entre un

programme d'application et le matériel utilisé [GAR 90], le terme interface étant pris dans le sens commun de réalisation de la communication entre deux acteurs.

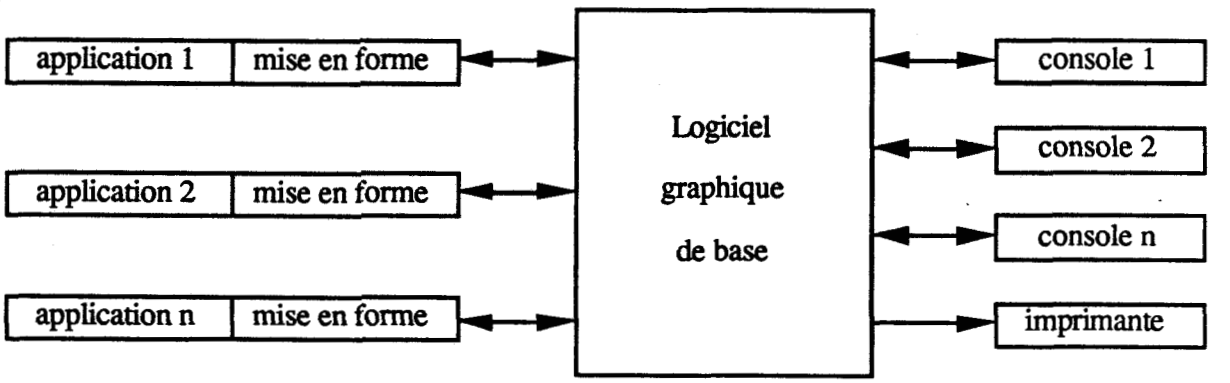


figure III.3

I.5. Conclusion.

La présentation générale que nous venons de faire, permet d'éclairer le lecteur sur les différents intérêts qu'apportent une interface graphique et comment sa construction peut être élaborée.

En suivant le schéma que nous venons de présenter, nous étudions les différents éléments graphiques ainsi que la structure de données, utilisés par le logiciel que nous avons réalisé, dans la partie II. Ensuite, afin de montrer comment il s'articule, nous donnons une description du logiciel graphique de base. Enfin, les extensions et les améliorations possibles autour de cet outil sont présentées dans la dernière partie.

II. Modèle de l'interface graphique.

II.1. Introduction.

L'interface graphique que nous avons réalisée permet de représenter des graphes de type réseaux de Petri.

Nous avons donné dans le premier chapitre les définitions des Réseaux de Petri utilisés pour la modélisation de la méthodologie CASPAIM1.

Rappelons que :

- les places d'un réseau de Petri sont associées au concept d'état du système,
- les transitions sont associées à la notion d'événements dont l'occurrence modifie l'état du système,
- un réseau de Pétri est défini par les places, par les transitions, et par un marquage initial,
- et que l'identification d'un état du système se fait par le marquage de certaines places et par le déplacement et l'évolution de ce marquage dans le graphe.

De plus, les prégraphes que nous voulons représenter graphiquement sont des réseaux de Petri colorés où la coloration permet de faire évoluer au sein d'un même graphe un ensemble fini de marques différentes. Le cheminement des marques d'une couleur s'effectue en suivant les arcs qui autorisent le passage de cette couleur.

Ce bref rappel sur les réseaux de Petri étant fait, nous allons présenter en détail les modèles utilisés par le logiciel réalisé pour l'interface graphique. Pour cela, nous donnons dans un premier temps les types d'informations portées par les réseaux de Petri de type prégraphe, qui correspondent aux types d'actions rencontrées lors de la description des gammes opératoires. Ensuite, nous présentons la hiérarchie des modèles utilisés, ainsi, qu'une analyse des graphes à représenter afin de justifier les choix qui ont été faits pour la modélisation de l'interface graphique. Enfin, nous donnons les raisons qui nous ont poussé à choisir le langage de programmation du logiciel de l'interface graphique.

II.2. Formalisme de description des actions du prégraphes.

Afin de définir les modèles utilisés par l'interface graphique, donnons dans un premier temps, les différents types de réseaux de Petri que nous voulons représenter.

Par convention, une place du prégraphe (de type réseau de Petri coloré), représente un

lieu physique qui ne sera représenté qu'une et une seule fois. Une marque représente un objet dans un état d'avancement donné dans sa gamme opératoire. Le tir d'une transition représente physiquement le transfert ou la transformation simultanée d'objets. Le degré d'avancement de la transformation des pièces sera repéré grâce aux symboles - et +, (une pièce non encore usinée sera représentée u-, une pièce usinée u+).

L'observation des systèmes réels nous a conduit à définir trois types d'actions rencontrées lors de la description des gammes opératoires, qui permettent de décrire complètement et sans ambiguïté les fonctionnalités du système de production pour chaque type de pièces.

II.2.1. Les actions de transformation.

Ces actions telles que les transformations physiques (usinage : tournage, fraisage, alésage, ..., conditionnement : traitements thermiques ou chimiques, lavage, mise en peinture, ...), et les assemblages (soudage, vissage, ...), pour lesquelles la pièce ne quitte pas le lieu considéré, modifient l'état matériel du produit.

Exemple 1.

Par exemple, l'opération de tournage d'une pièce t- sur un tour T, sera représentée sous la forme suivante :

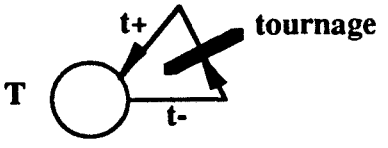


figure III.4

II.2.2. Les actions positionnelles.

Ces actions telles que les transferts entre postes (convoyeurs, navettes, chariots, ...), les repositionnements sur un poste (retournement, ...), et les positionnements avant assemblage et après désassemblage, pour lesquelles la pièce quitte le lieu considéré, excepté pour le retournement, ne servent qu'à situer les objets dans l'environnement de production.

Exemple 2.

Le transfert d'une pièce t- d'un poste P1 vers un poste P2 sera représenté sous la forme suivante :

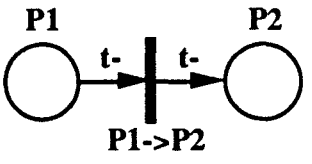


figure III.5

Exemple 3.

Le retournement d'une pièce t+r-t- effectué sur le tour T sera représenté sous la forme suivante :

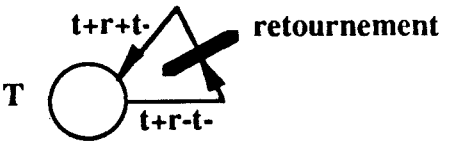


figure III.6

Exemple 4.

La représentation d'un positionnement avant assemblage d'une pièce a venant du poste A et d'une pièce b venant d'un poste B avec mise en correspondance des pièces a et b sur un poste C sera la suivante :

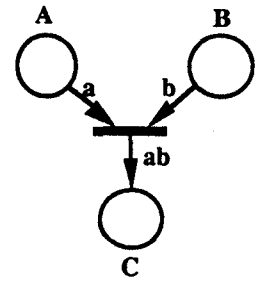


figure III.7

Exemple 5.

La représentation d'une opération de palettisation d'une pièce t+ provenant d'un poste T de tournage sur une palette p en attente au poste P sera la suivante :

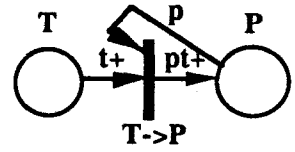


figure III.8

Exemple 6.

De même, l'opération de dépalettisation sera la suivante :

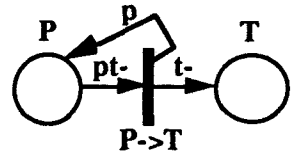


figure III.9

Exemple 7.

L'action de dépalettisation d'une pièce u- en P, puis de transfert entre ce poste P et un poste U, et d'assemblage de la pièce u- sur une platine d'usinage pa en U sera représentée de la façon suivante :

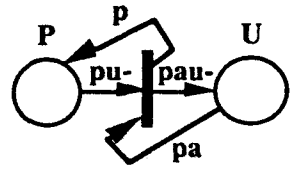


figure III.10

II.2.3. Les actions informationnelles.

Ces actions telles que des opérations de métrologie, dont dépend l'état de la pièce considérée, permettent de garantir la conformité des états matériels obtenus effectivement après un usinage par exemple.

Exemple 8.

Une pièce t+ ayant subi une opération de tournage sera à la sortie du poste M de métrologie classée dans deux intervalles t1+ et t2+ en fonction des résultats de mesure. Cette opération sera représentée sous la forme suivante :

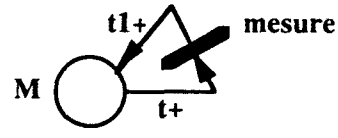


figure III.11

II.3. Structures de données.

II.3.1. Présentation générale.

Si nous nous référons à la séquence de programmes nécessaires à la réalisation d'un logiciel graphique définie dans la partie précédente (cf. figure III.2), l'interface graphique que nous avons réalisée est surtout un logiciel de mise en forme. Elle inclut cependant, une partie du logiciel de préparation à la visualisation, afin de permettre l'interfaçage entre le logiciel de mise en forme et le logiciel graphique de base. Les raisons de ce choix seront exposées dans la partie II.4 de ce chapitre.

Pour la conception de l'interface graphique, du point de vue de la structure de données, nous distinguons :

- la **structure de données de l'application**, en l'occurrence les éléments du fichier donnant la description du prégraphe,
- la **structure de données de description**, qui est propre à l'interface graphique,
- le **fichier graphique**, qui est une bibliothèque d'éléments graphiques de haut niveau par rapport aux primitives graphiques de base telles que le tracé d'un caractère, d'un point ou d'un segment de droite.

Avant de présenter de manière plus détaillée ces structures de données, ajoutons que l'interface graphique utilise la notion d'objet, [FER 90]. Rappelons pour cela, que :

Un **objet** est une entité informatique qui comprend à la fois

- une partie statique qui représente son état et les liens qui l'unissent à d'autres objets,
- et une partie dynamique qui décrit son comportement, c'est-à-dire l'ensemble des opérations qui lui sont applicables ainsi que ses réactions aux événements du milieu dans lequel il se trouve.

Plusieurs objets peuvent posséder une structure et des comportements en commun et peuvent être regroupés à l'aide d'un modèle général : une **classe**.

Les objets générés à partir d'une classe sont appelés les **instances** de la classe.

La définition d'une classe est constituée par un ensemble de ***champs***, appelés aussi attributs, qui décrivent la structure de ses instances, et d'opérations, appelées **méthodes**, qui leur sont applicables.

L'utilisation de la notion d'objet se fait à tous les niveaux de notre logiciel, aussi bien en gestion de structure interne, qu'au niveau de la gestion de l'environnement graphique.

Pour ce qui est de la gestion des objets graphiques, nous utilisons un logiciel graphique de base, où une arborescence d'objets graphiques est déjà définie. L'utilisation de la Programmation Orienté Objet rend le langage de description d'objet extrêmement facile à étendre tant au niveau des méthodes qu'au niveau des objets élémentaires.

II.3.2. Structure de données de l'application.

Le point d'entrée de l'interface graphique est un fichier décrivant sous une forme littérale le prégraphe à représenter. Ce fichier est un fichier Le_lisp provenant de la phase de spécification de CASPAIM1. Il comprend la description du réseau de Pétri sous forme de p-listes associées aux places et aux transitions. Le fichier du prégraphe de la figure III.24 est donné dans l'Annexe III.1. Pour facilité la lecture de ce fichier, nous l'exprimons sous une forme compréhensible par un non spécialiste des p-listes. De plus, nous ne donnons que les informations nécessaires à l'interface graphique.

Ce fichier comporte les informations suivantes :

- la liste des places,
- la liste des transitions.

Nous avons donc choisi, d'exprimer les différentes liaisons du graphe par les différents champs de deux classes d'objets, les objets **places** et les objets **transitions**.

Chaque instance de la classe **places** comporte les champs suivants :

- le nom de la place,
- les transitions amont,
- les transitions aval.

Chaque instance de la classe **transition** comporte les champs suivants :

- le nom de la transition,
- la nature de la transition (correspond aux types définis dans le paragraphe II.3.4.),
- le domaine de couleur,
- les arcs prédécesseurs
  - le poids de l'arc,
  - la couleur,
  - la place amont,
- les arcs successeurs
  - le poids de l'arc,
  - la couleur,
  - la place aval.

II.3.3. Structure de données de description.

Nous venons de décrire brièvement la structure de données de l'application. Ces données sont traduites et enrichies pour former la structure de données de description, sous forme d'objets, de l'interface graphique.

Cette structure de données de description permet la préparation à la représentation de prégraphes, et comporte trois classes d'objets qui permettent d'exprimer les différentes liaisons du graphes, les classes **place, transit**, et **arc**, et une classe **pregraphe** qui permet la gestion des fichiers des graphes édités.

La classe place.

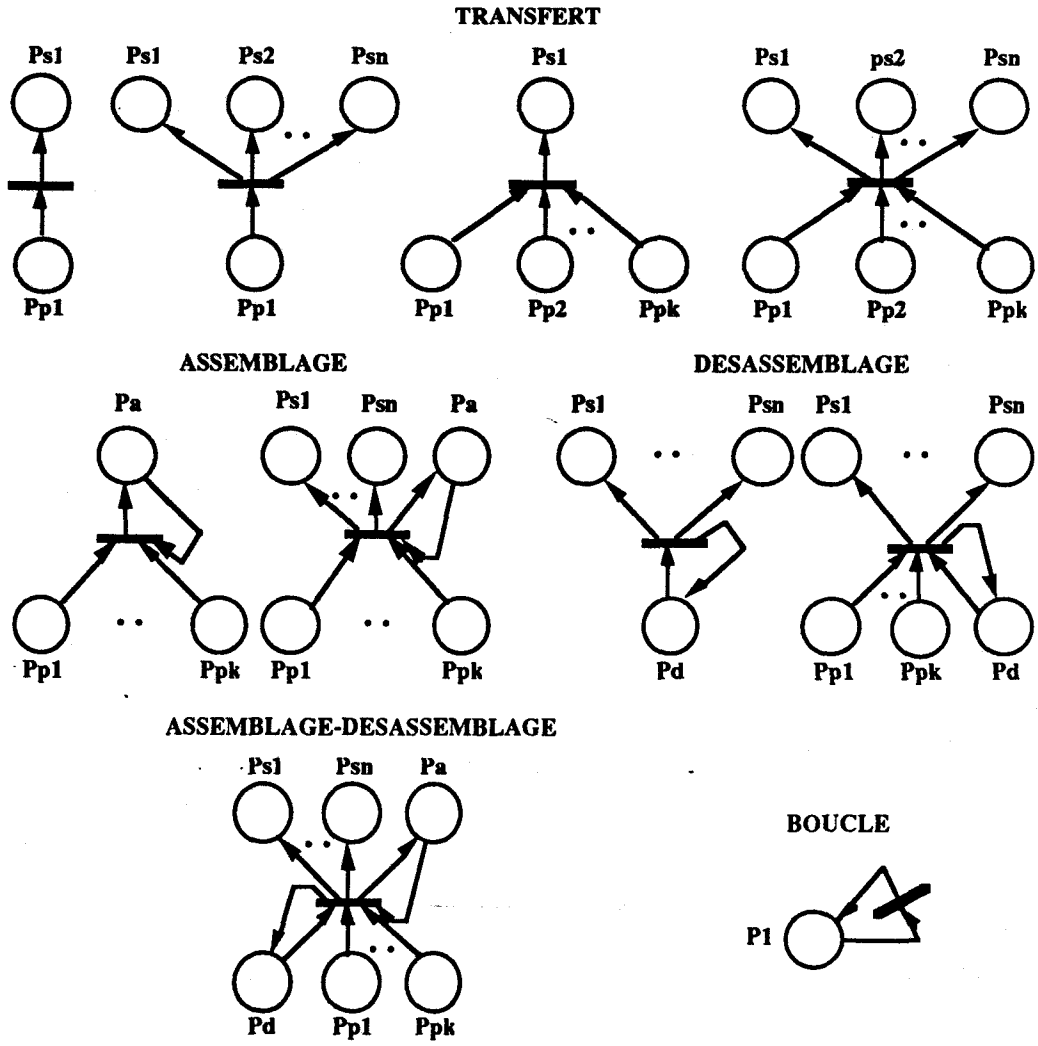
Chaque instance de la classe **place** décrit une place du prégraphe. Ces objets comportent les champs suivants :

- nom-pl : chaîne de caractères indiquant le nom de la place.
- coord-nomp : paire pointée d'entiers indiquant les coordonnées graphiques du nom de la place.
- coord-pl : paire pointée d'entiers indiquant les coordonnées graphiques du centre de l'objet cercle représentant la place.
- succ-pl : liste d'identificateurs d'objets de la classe arc, arcs successeurs de la place.
- pred-pl : liste d'identificateurs d'objets de la classe arc, arcs prédécesseurs de la place.
- des-pl : liste d'identificateurs d'images représentant le nom et l'objet de la classe cercle, représentation graphique de la place.

La classe transit.

Chaque instance de la classe **transit** décrit une transition du prégraphe.

D'après l'analyse graphique du prégraphe présentée dans la partie II.2. de ce chapitre, nous avons scindé en cinq groupes les liaisons qui composent un réseau de Petri. Précisons bien que cette classification est purement graphique, et permet ainsi de connaître de manière simple quelle est la représentation graphique que le logiciel doit prendre en compte. Ces cinq groupes sont représentés sur la figure III.12.



figures III.12

Nous tenons à préciser que les termes employés pour nommer ces groupes peuvent rappeler certaines notions que nous avons présentées dans la partie II.2. de ce chapitre, mais ne doivent en aucun cas se confondre avec celles-ci.

En parallèle à cette classification, l'analyse des graphes nous a permis de mettre en évidence le fait que deux places peuvent être mises en liaison par deux ou plusieurs transitions, comme le montre la figure III.13.

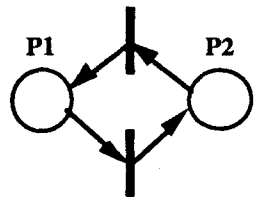


figure III.13

Afin d'éviter le chevauchement de ces transitions, nous devons connaître si deux places sont en liaison par plus d'une transition. Dans le cas où deux transitions ou plus relient deux même places, nous appelons ce type de relation : **liaison multiple.**

Considérons deux liaisons l1 et l2.

Soient P1 la place amont de l1, P2 sa place aval, et P3 la place amont de l2, P4 sa place aval.

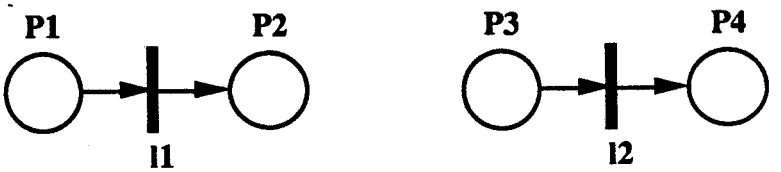


figure III.14

Nous considérons que l1 et l2 composent une liaison multiple uniquement si elles n'ont qu'une et une seule place amont et une et une seule place aval, et que de plus, soit que P1 = P3 et P2 = P4, soit que P1 = P4 et P2 = P3.

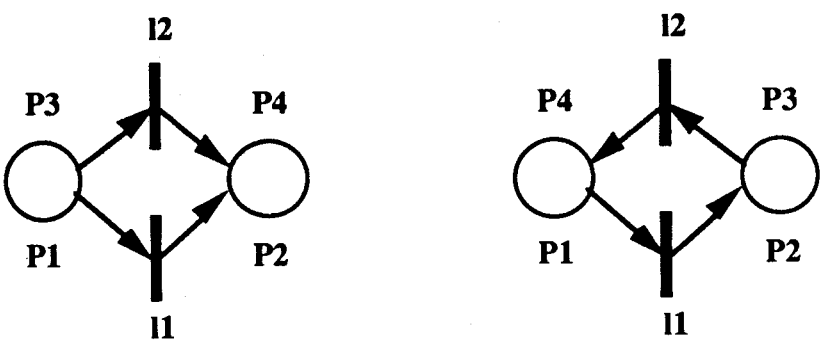


figure III.15

Les liaisons qui composent une liaison multiple peuvent appartenir à un des quatre cas suivants :

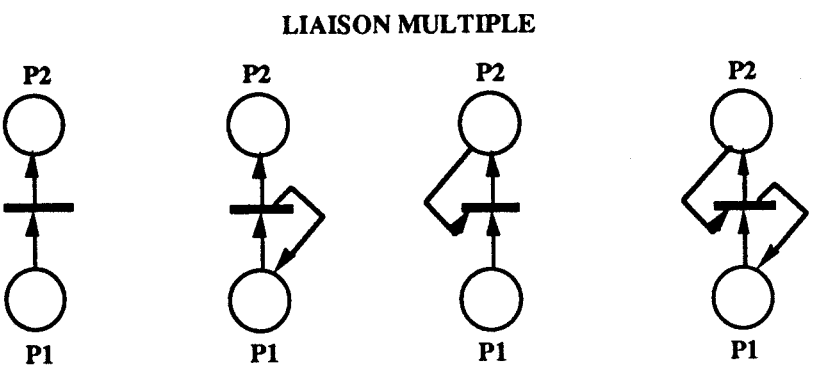


figure III.16

Les objets de la classe **transit** comportent donc les champs suivants :

- nom-trans : chaîne de caractères indiquant le nom de la transition.
- coord-nomt : paire pointée d'entiers indiquant les coordonnées graphiques du nom de la transition.
- coord-trans : paire pointée d'entiers indiquant les coordonnées graphiques du centre de l'objet de la classe transd représentant la transition (cf II.3.4. du chapitre III).
- angle-trans : réel indiquant l'angle que fait l'objet de la classe transd avec l'axe des abscisses.
- type-trans : chaîne de caractères indiquant le type de la transition, type-trans = { transfert, boucle, assemblage, désassemblage, assemblage-désassemblage}
- lia-multi : liste d'identificateurs d'objets de la classe transit composant la liaison multiple.
- dom-var : liste des variables couleurs des arcs successeurs et prédécesseurs de la transition et leur domaine associé.
- succ-trans : liste d'identificateurs d'objets de la classe arc, arcs successeurs de la transition.
- pred-trans : liste d'identificateurs d'objets de la classe arc, arcs prédécesseurs de la transition.
- des-trans : liste d'identificateurs d'images représentant le nom et l'objet de la classe transd, représentation graphique de la transition.

La classe arc.

Chaque instance de la classe **arc** décrit un arc du prégraphe. Ces objets comportent les champs suivants :

- coul-arc : chaîne de caractères indiquant la couleur portée par l'arc.
- coord-nomc : paire pointée d'entiers indiquant les coordonnées graphiques de la couleur de l'arc.
- type-arc : chaîne de caractères indiquant le type de l'arc, type-arc = {boucle, retour, simple}.
- sens-arc : chaîne de caractères indiquant le sens de l'arc, sens-arc = {tp, pt}.
- pl-arc : identificateur de l'objet de la classe place indiquant la place reliée par l'arc.

- trans-arc : identificateur de l'objet de la classe transit indiquant la transition reliée par l'arc.
- plc-arc : paire pointée d'entiers indiquant les coordonnées graphiques de l'intersection de l'arc avec la place.

Ces trois classes permettent de gérer les relations qui existent entre les différents éléments d'un graphe, et les relations qui existent avec les objets graphiques permettant l'affichage du prégraphe. Mais, afin de faciliter la gestion des diverses versions graphiques d'un graphe, ou des divers graphes que l'utilisateur veut éditer, nous avons créé une classe **prégraphe** qui assure la sélection des fichiers.

La classe prégraphe.

Chaque instance de la classe **prégraphe** décrit un fichier prégraphe à construire, à éditer, ou à simuler. Ces objets comportent les champs suivants :

- nom-pr : chaîne de caractères indiquant le nom du fichier,
- edit-pr : booléen permettant de savoir si le graphe a été édité graphiquement une première fois,
- sauv-pr : booléen indiquant si la version courante du fichier a été sauvée ou non,
- coul-pr : booléen indiquant si la version courante comporte l'affichage de la couleur des arcs ou non,
- vers-pr : entier indiquant le numéro de version du fichier en cours de saisie, de modification, ou de simulation.

II.3.4. Fichier graphique.

Notre but n'est pas de faire une présentation exhaustive in extenso des objets graphiques utilisés dans cette interface. C'est pourquoi, nous ne présenterons pas les objets concernant la gestion de l'environnement. Toutefois, donnons quelques termes de vocabulaire et quelques exemples des objets implantés dans le logiciel graphique de base, afin d'éclairer notre travail sur ce point.

Une **image** est un objet que l'on peut afficher sur un écran haute résolution.

Une **application** est une combinaison d'images actives et inactives. Les **images inactives** sont simplement affichées à l'écran, les **images actives** déclenchent des programmes en fonction des actions de l'utilisateur (boutons poussoirs, barres de

défilement, images sélectionnables et déplaçables, ...).

Le langage d'image du logiciel graphique de base permet de construire des **images élémentaires** (chaînes de caractères, rectangles, icônes, ...), et des **images composées** en combinant les images en colonnes, en rangées, ...

Nous utilisons cet environnement d'objets prédéfinis du logiciel graphique de base pour tout ce qui concerne la gestion des fenêtres, boutons poussoirs, barres de défilement, ..., (l'arborescence de ces objets prédéfinis se trouve en Annexe III.2).

Pour une meilleure compréhension de la méthode présentée dans la troisième partie de ce chapitre, il nous semble important de spécifier les choix, et les objets graphiques, définis pour représenter un prégraphe.

Pour la représentation de ces différents éléments, nous avons défini cinq classes d'objets faisant partie de l'arborescence de l'environnement du logiciel graphique de base. Pour que ces objets héritent des méthodes de leur classe d'appartenance, nous avons redéfini les méthodes d'affichage de ces objets, de paramétrage de leur boîte enveloppante, ainsi que les méthodes permettant de déplacer ces différents objets. Par ailleurs, pour une gestion simplifiée des relations inter-objets, nous employons des constantes globales, (par exemple, le rayon du cercle représentant une place, la longueur et la largeur d'une transition, ...).

Les éléments qui composent un réseau de Petri sont les places, les transitions, et les arcs. Les différentes classes que nous avons définies pour représenter ces éléments sont pour :

Les places.

La représentation d'une place est un objet de la classe **cercle** définie par les coordonnées du centre C.

cercle

- xc : entier correspondant à l'abscisse de C
- yc : entier correspondant à l'ordonnée de C

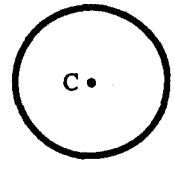


figure III.17

Les transitions.

La représentation d'une transition est un objet de la classe **transd** définie par les coordonnées du milieu de la transition M et par l'angle ang-tra que fait la transition avec l'axe des abscisses.

transd

- xm-tra : entier correspondant à l'abscisse de M
- ym-tra : entier correspondant à l'ordonnée de M
- ang-tra : réel correspondant à l'angle que fait la transition avec l'axe des abscisses

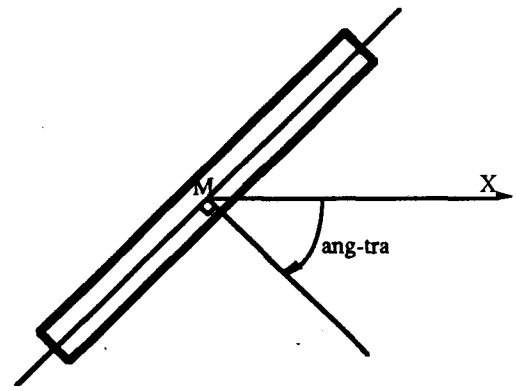


figure III.18

Les arcs.

Pour la représentation des arcs, nous avons défini deux classes d'objets. La première, la classe **liaison**, permet de représenter les arcs rectilignes. Les objets sont définis par les coordonnées du point de départ D de l'arc, et par les coordonnées du point d'arrivée A de l'arc. La deuxième, la classe **retour**, permet de représenter les arcs exprimant une palettisation, par exemple. Les objets sont définis par les coordonnées de D (point d'intersection de la place et de l'arc), par les coordonnées de A (point d'intersection de la transition et de l'arc), et par le sens de l'arc (sens place->transition ou sens transition->place).

liaison

- xd-lia : entier correspondant à l'abscisse de D
- yd-lia : entier correspondant à l'ordonnée de D
- xa-lia : entier correspondant à l'abscisse de A
- ya-lia : entier correspondant à l'ordonnée de A

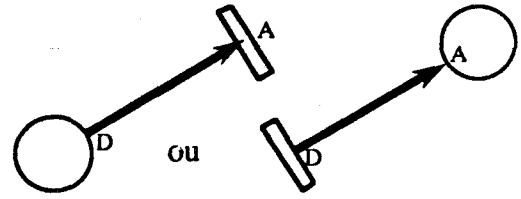


figure III.19

retour

- xd-ret : entier correspondant à l'abscisse de D
- yd-ret : entier correspondant à l'ordonnée de D
- xa-ret : entier correspondant à l'abscisse de A
- ya-ret : entier correspondant à l'ordonnée de A
- ty-ret : booléen correspondant au sens de l'arc

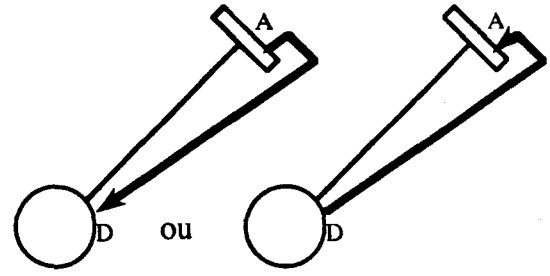


figure III.20

Enfin, nous avons introduit une cinquième classe, la classe **boucle**, qui permet l'affichage directe de la transition, de l'arc aval, et de l'arc amont, afin de représenter une fonction d'usinage, par exemple, de manière simple. Les objets sont définis par les

coordonnées du centre de la place, et par l'angle que fait l'axe de la transition avec l'axe des abscisses.

boucle

- xa-bou : entier correspondant à l'abscisse de A
- ya-bou : entier correspondant à l'ordonnée de A
- a-bou : réel correspondant à l'angle que fait l'axe de la transition avec l'axe des abscisses

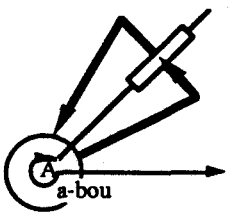


figure III.21

Ces cinq classes nous permettent de représenter tous les réseaux de Petri. Rappelons qu'il est possible d'étendre à volonté le nombre de classes, ainsi nous offrons la possibilité de représenter des graphes de type réseau de Petri ou autres avec d'autres formalismes de représentation graphique, sans changer les fonctionnalités du logiciel d'interface graphique.

II.4. Implantation de l'interface graphique.

II.4.1. Présentation générale.

De manière générale, le choix du langage et de la machine pour l'implantation d'un logiciel, provient du contexte d'utilisation.

Une "bonne" interface graphique doit laisser l'initiative à l'utilisateur. Il doit pouvoir décider librement d'activer tel ou tel type d'outil, parmi l'ensemble des outils proposés par l'application. Le passage de l'un des outils à l'autre doit être immédiat, plusieurs outils doivent pouvoir être activés simultanément à l'écran, dans des fenêtres indépendantes. L'utilisateur peut alors, dans une certaine mesure, organiser son travail et arranger son écran de façon à trouver la manière la plus confortable et la plus efficace d'utiliser le logiciel. Il peut ainsi développer une culture d'utilisation du programme, adaptée à la manière de comprendre les problèmes qui lui est propre et qu'il maîtrise complètement. L'écriture de tels programmes est cependant loin d'être une tâche facile. Les problèmes liés à la construction d'interfaces sont en fait de plusieurs ordres :

- pilotage par l'utilisateur : les programmes doivent être contrôlés par les événements. Guettant constamment les différentes actions de l'utilisateur, ils doivent être capables de passer immédiatement d'une application à une autre,

- structuration de l'écran : il faut d'abord pouvoir découper l'écran en zones indépendantes, contenant différents composants d'interface. Il faut ensuite pouvoir modifier ce découpage par programme, par exemple pour ajouter dynamiquement des composants dans un tableau de bord. Il faut donc un formalisme permettant de décrire et de manipuler la structure des objets affichés à l'écran,

- gestion de la cohérence : des vues multiples sur des mêmes structures, peuvent induire des incohérences dans le contenu des fenêtres lors de modifications des structures visualisées. Il faut mémoriser les vues sur les objets pour savoir comment conserver l'écran à jour,

- portabilité des applications : les bibliothèques graphiques sont encore trop liées au matériel, et il est toujours très difficile de concevoir une application qui fonctionnera indifféremment sur un grand nombre de machines différentes.

Les boîtes à outils résolvent généralement le premier problème. Une boîte à outils permet de construire des composants d'interface élémentaires affichables ensuite à l'écran. Une boucle d'interaction, fournie avec la boîte à outils, s'occupe de surveiller la souris et de déclencher les réactions des différents composants. L'application qui a construit les composants reprendra la main par le biais de procédures attachées aux composants. Le pilotage par les événements est ainsi permis par la boîte à outils, et l'application l'utilise pour décrire ses différents points d'entrée et y associer des composants d'interface.

La structuration de l'écran peut être obtenue par l'utilisation d'un langage d'image, permettant, à l'aide de constructeurs d'images hiérarchiques, de bâtir un modèle structuré du contenu des fenêtres. L'application commence alors par construire une représentation graphique structurée des objets à afficher à l'écran, et les transmet à la boîte à outils qui s'occupe de les afficher. Toute modification de la structure de l'image est répercutée par la boîte à outils sur l'écran, il suffit donc de faire des modifications sur la structure des images pour modifier le contenu des fenêtres, faire des animations, ... Les langages graphiques GKS et PHIGS répondent dans une certaines mesure à ces critères.

Très peu d'outils permettent actuellement de gérer la cohérence globale d'une application. L'utilisation d'un modèle d'interaction sain et complet, comme le modèle MVC (Modèle - Vue - Contrôleur) de SMALLTALK (développé par XEROX corp.), peut aider à assurer la cohérence entre les différentes vues d'un même objet. Les techniques de

programmation orientées objets peuvent être ici d'un grand secours.

L'expérience montre que ce n'est qu'un mythe de prétendre qu'il peut exister une indépendance totale entre le logiciel et la machine sur lequel il sera implanter, et il vaut mieux chercher à établir des règles assurant une certaine indépendance, quitte à effectuer des ajustements au contexte d'accueil, les plus minimes possibles. Les quatre composants du contexte d'utilisation d'un logiciel, graphique en l'occurrence, sont les suivants :

Indépendance par rapport au matériel.

Deux contraintes interfèrent sur ce type d'indépendance. Elles proviennent du calculateur exécutant le logiciel graphique, et de la console de visualisation.

Pour ce qui concerne le calculateur, l'influence directe sur le logiciel graphique est relativement faible car la plupart des ressources de la machine ne sont accessibles qu'à travers le système d'exploitation.

En ce qui concerne la console de visualisation, les problèmes proviennent de la diversité des technologies employées et de leurs caractéristiques parfois incompatibles. Les trois composants d'une console de visualisation étant l'écran, le processeur graphique, et les dispositifs de communication.

Indépendance par rapport à l'application.

Le logiciel graphique étant au service de diverses applications, il doit permettre la mise en œuvre de celles-ci sans accueillir dans sa définition des caractéristiques trop spécifiques aux applications considérées. Cette indépendance réside dans l'interface avec le logiciel de mise en forme. Le rôle du logiciel graphique de base sera limité à l'interprétation des données graphiques en vue de la représentation du dessin dans le plan, et à fournir des informations de nature graphique à partir des dispositifs de communication.

Indépendance par rapport au langage hôte.

Pour que l'utilisation d'un logiciel graphique soit correcte, il faut qu'il soit bien compris par les développeurs. En conséquence, le logiciel graphique, destiné à compléter les instructions d'un langage de programmation, doit être défini à un niveau équivalent à celui du langage hôte.

Indépendance par rapport au système d'exploitation.

Il est évident que la liberté de manœuvre est presque nulle, car le concepteur de l'interface graphique doit se plier au système d'exploitation.

Un système informatique est un ensemble de matériels et de logiciels destinés à réaliser des tâches qui mettent en jeu le traitement automatique de l'information. Un tel système est relié au monde extérieur par des organes d'accès, qui lui permettent de communiquer avec des usagers ou d'interagir avec des dispositifs physiques qu'il est chargé de commander ou de surveiller. L'ensemble des services fournis par un système informatique définit pour l'utilisateur de ce système une nouvelle machine, la machine abstraite. L'architecture d'un tel système peut être divisé en couches. La figure III.22 décrit cette organisation par un schéma hiérarchique où chaque couche utilise les ressources fournies par la couche inférieure et présente à la couche supérieure une interface décrivant les ressources qu'elle fournit. La couche basse représente le système d'exploitation fourni par le constructeur, est peut être elle-même décomposée en couches. La couche supérieure offre des services de plus en plus spécifiques à l'application.

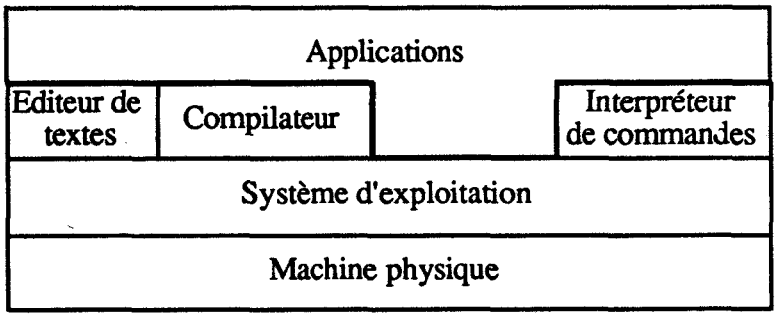


figure III.22

En fait, la portabilité des applications ne peut finalement être obtenue que par l'utilisation d'un langage portable, existant sur de nombreuses machines et muni d'une bibliothèque graphique portable.

De bons outils, répondant aux critères exposés ci-dessus, sont nécessaires pour pouvoir développer rapidement et sûrement des interfaces graphiques sophistiquées. Mais les outils ne suffisent pas et une méthodologie de construction d'interface est indispensable. Le principe clef, que nous avons exposé dans la première partie de ce chapitre, est la séparation du programme et de l'interface. Les langages fonctionnels et les mécanismes orientés objets peuvent être particulièrement bien adaptés à la réalisation de cette séparation. Dans un langage fonctionnel les procédures sont des objets qui peuvent être passés en arguments ou reçus en résultats.

II.4.2. Choix d'implantation.

L'interface graphique que nous présentons, est réalisée en Le_Lisp de l'INRIA, en interprété sur VAX, dont le système d'exploitation est VMS et tourne sur des VAX-stations 3100. Le choix du langage Le_Lisp a été fait pour plusieurs raisons.

La raison principale qui nous a poussé à utiliser Le_Lisp, est qu'il intègre une couche objet, et permet donc, d'utiliser le concept d'objets graphiques pour réaliser la représentation d'un réseau de Petri. Nous utilisons pour la réalisation de notre logiciel, une surcouche de Le_Lisp, Aïda (développé par la société Ilog), orientée graphique, qui permet une conception aisée d'interfaces graphiques. Nous avons adopté cette solution, par rapport à une solution GKS (première norme graphique adopté par l'ISO), afin de développer rapidement et de manière hiérarchisée notre interface graphique, car GKS ne dispose pas de structure hiérarchique pour décrire une image. Aïda est un système de construction d'interface général, portable et extensible, basé sur les principes que nous venons d'exposer dans la partie II.4.1 de ce chapitre. Un langage d'image permet de construire une représentation structurée de l'écran (l'arborescence des objets prédéfinis d'Aïda, est donnée en Annexe III.2), une boucle d'interaction s'occupe de suivre les mouvements de la souris, des mécanismes objets permettent de construire un modèle propre des objets visualisés à l'écran. L'utilisation d'un langage fonctionnel, Le_Lisp, facilite le paramétrage des programmes, encourageant la séparation programme/interface. Grâce à la grande portabilité du langage Le_Lisp et de sa bibliothèque graphique (le Bitmap Virtuel), Aïda est immédiatement portable sur un grand nombre de machines. Il en va de même pour les applications écrites en Aïda. Un exemple de l'environnement dont nous disposons avec Aïda, est présenté en Annexe III.3.

L'architecture en couche de l'application est présentée sur la figure III.23.

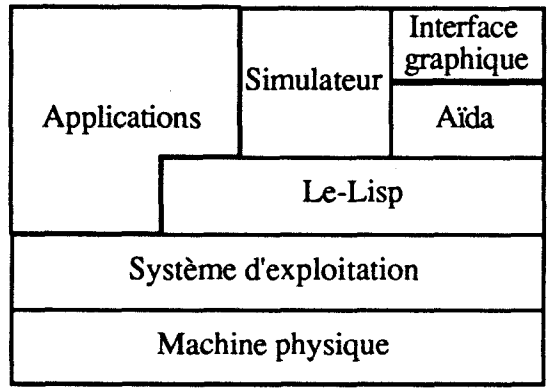


figure III.23.

II.5. Conclusion.

Nous venons de présenter les choix de modèles que nous avons pris pour réaliser l'interface graphique. Le choix de l'utilisation d'un langage orienté objets nous paraît être la méthode la plus appropriée pour permettre une évolution constante et facilement réalisable du logiciel. En effet, cette forme de structure permet d'avoir une grande flexibilité pour la réalisation du logiciel. Ainsi, il est facile de créer un environnement simple puis de plus en plus complexe au fil de l'élaboration du logiciel, par ajout ou retrait de variables d'instance, par création de nouvelles classes d'objets, ou par ajout de méthodes associées aux différents objets. Ceci permet donc, d'adopter une démarche incrémentale de la conception du logiciel. Par contre, la définition de certains objets peut paraître lourde, mais conduit à une réalisation adaptée d'une maquette d'interface graphique, prouvant l'utilité des méthodes que nous allons exposer dans la partie suivante. Enfin, la décomposition en modules de l'interface graphique (cf. figure III.1), permet de réutiliser le logiciel graphique de base pour des applications autres que le simulateur.

III. Méthode de représentation graphique.

III.1. Introduction.

L'interface graphique que nous avons réalisée, permet de visualiser des réseaux de Petri issus de la méthodologie CASPAIM1.

Les modèles retenus pour l'élaboration du logiciel d'interface graphique sont construits de manière automatique en suivant une méthode de décomposition orientée objets. Les différentes définitions de ces modèles venant d'être présentées, nous allons à présent commenter la méthode utilisée pour représenter les graphes.

Dans une première partie, nous présentons un exemple sur lequel sont bâties toutes les explications concernant la méthode. Dans un second temps, nous présentons de manière générale l'organisation du logiciel. Puis, nous examinons comment s'effectue l'édition d'un graphe. Enfin, nous développons la méthode de positionnement automatique des places d'un graphe, que nous avons élaborée.

III.2. Exemple explicatif.

Afin de préciser le comportement de l'interface graphique, considérons la cellule flexible implantée dans le Département de Génie Mécanique de Centrale Lille, qui va servir de support pour les explications de cette partie.

Le système est composé de :

- un tour (noté T),
- un centre d'usinage (noté F), qui dispose de deux tables équivalentes permettant d'effectuer les entrées/sorties de pièces,
- un convoyeur à chaîne, sur lequel six butées définissent trois postes de chargement/déchargement de pièces (P2, P3, P5) et trois files d'attente (P1, P4, P6). Le convoyeur est chargé du transport des pièces palettisées,
- un robot manipulateur (noté R1), chargé de l'alimentation du tour T,
- un robot industriel (noté R2), assurant les entrées/sorties de pièces, ainsi que le chargement/déchargement du centre d'usinage F,
- un tampon d'entrée E, et un tampon de sortie S.

Le schéma simplifié de la cellule est le suivant :

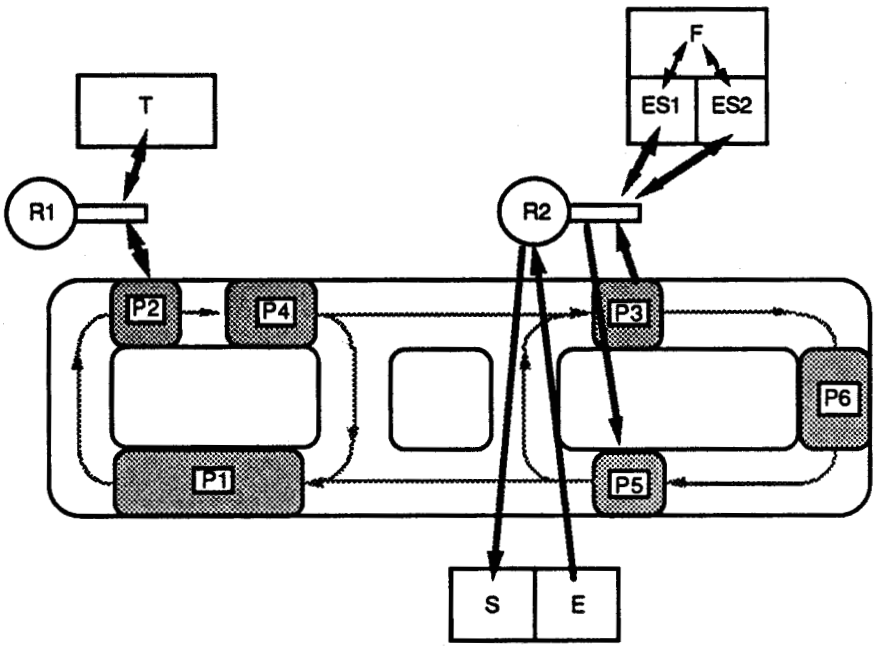


figure III.24

Les six zones définies sur le convoyeur grâce à des butées, ont les fonctions suivantes :

P1 est une zone tampon de pièces destinées au tournage,

P2 est la zone de palettisation/dépalettisation des pièces destinées au tournage,

P3 est le zone de dépalettisation des pièces destinées au centre d'usinage ou à la sortie,

P4 est une zone tampon de pièces provenant du tour,

P5 est la zone de palettisation des pièces provenant de l'entrée ou du centre d'usinage,

P6 est la zone de stockage des palettes vides.

Dans cette cellule circulent cinq types de pièces. Ces pièces sont usinées :

- soit sur le tour T, une(1) ou deux(2) fois,

(1), (2) - transfert et palettisation en P5 par le robot R2,

- transfert vers P1 puis vers P2,
- dépalettisation et transfert sur le tour par le robot R1, (la palette vide attend la fin d'usinage),
- usinage,

(2) - retournement de la pièce par le robot R1,

- deuxième usinage,

(1), (2) - transfert et palettisation de la pièce usinée en P2 par le robot R1,

- transfert vers P4 puis vers P3,
- dépalettisation et transfert vers la sortie par le robot R2,

- soit sur le centre d'usinage F,

(3) - transfert et palettisation par le robot R2 sur P5,
- transfert vers P3,
- dépalettisation et transfert par le robot R2 sur l'un des deux tampons d'entrée/sortie du centre d'usinage,
- usinage,
- transfert de la pièce usinée sur la sortie par le robot R2,

- soit sur le centre d'usinage F, puis sur le tour T,
  - même début que pour (3) jusqu'à la fin d'usinage,
  - transfert et palettisation de la pièce en P5 par le robot R2,
  - puis idem que pour (1),

- soit sur le tour T, puis sur le centre d'usinage F
  - même début que pour (1) jusqu'au transfert vers P3,
  - dépalettisation et transfert par le robot R2 sur l'un des deux tampons d'entrée/sortie du centre d'usinage,
  - usinage,
  - transfert de la pièce usinée sur la sortie par le robot R2.

III.3. Organisation du logiciel.

Notre première approche fut de réaliser un logiciel de représentation graphique du prégraphe de CASPAIM1.

Ce logiciel est scindé en plusieurs modules permettant la construction, l'édition, la correction et l'archivage du prégraphe du système de production. Son utilisation est régie par un ensemble de fenêtres et de menus dont l'organisation de base est la suivante :

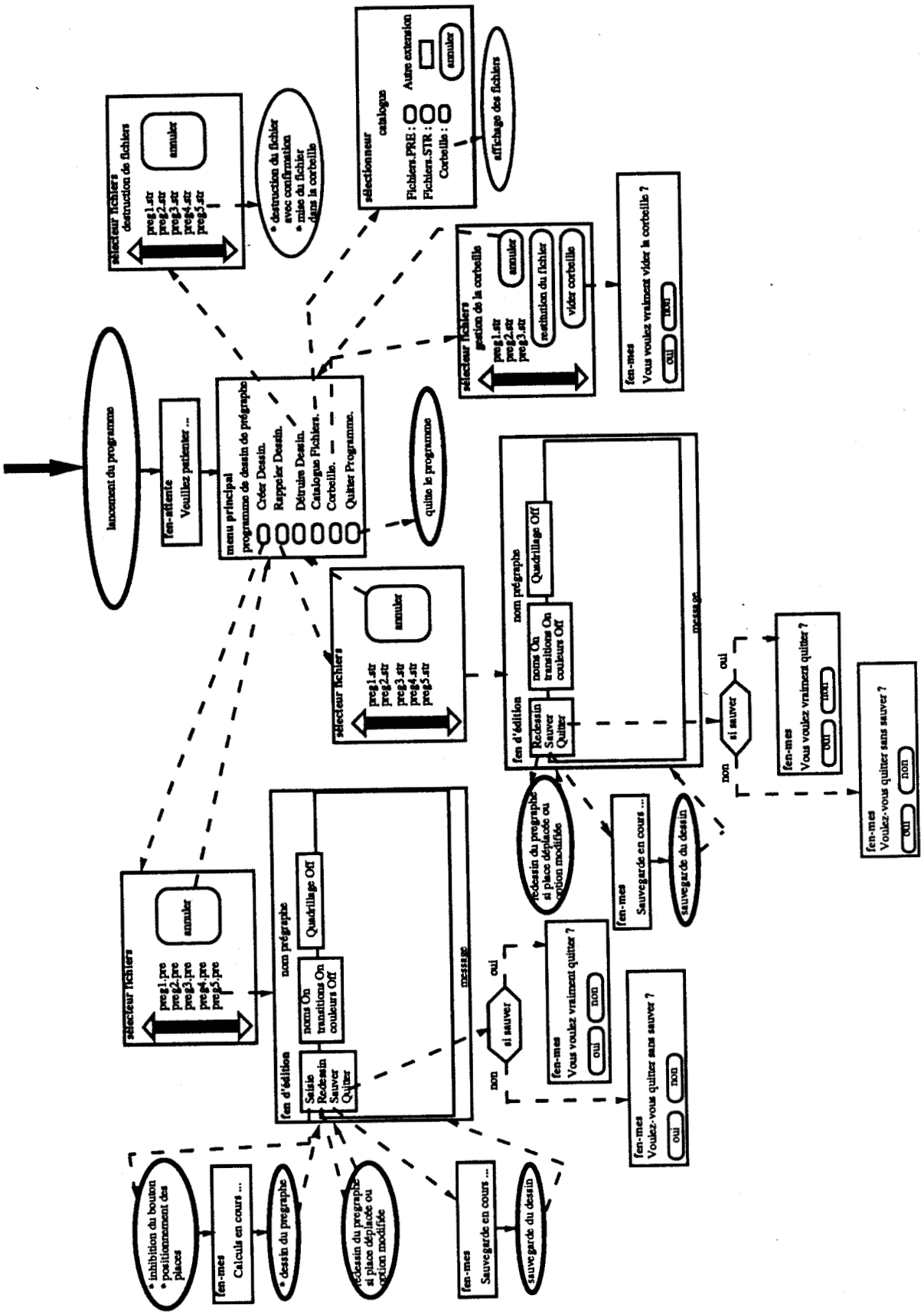


figure III.25

Une des caractéristiques principales de notre interface est la modularité du logiciel. En effet, nous considérons que chaque application Aïda est une classe d'objets (de fenêtres, de boutons, de sélectionneurs, ..., ainsi que de leur(s) méthode(s) associée(s)), et ainsi, nous pouvons étendre l'interface avec autant de fonctions supplémentaires nécessaires à une bonne gestion.

Lors de notre première approche, nous avons volontairement choisi de représenter les prégraphes, d'une part parce que le prégraphe constitue une description agrégée du rôle fonctionnel de production et constitue le plan de l'assemblage ultérieur du modèle de la partie commande, d'autre part, car la taille de ce réseau de Petri n'est pas trop importante et est facilement affichable à l'écran. La représentation manuelle du prégraphe de notre exemple est la suivante :

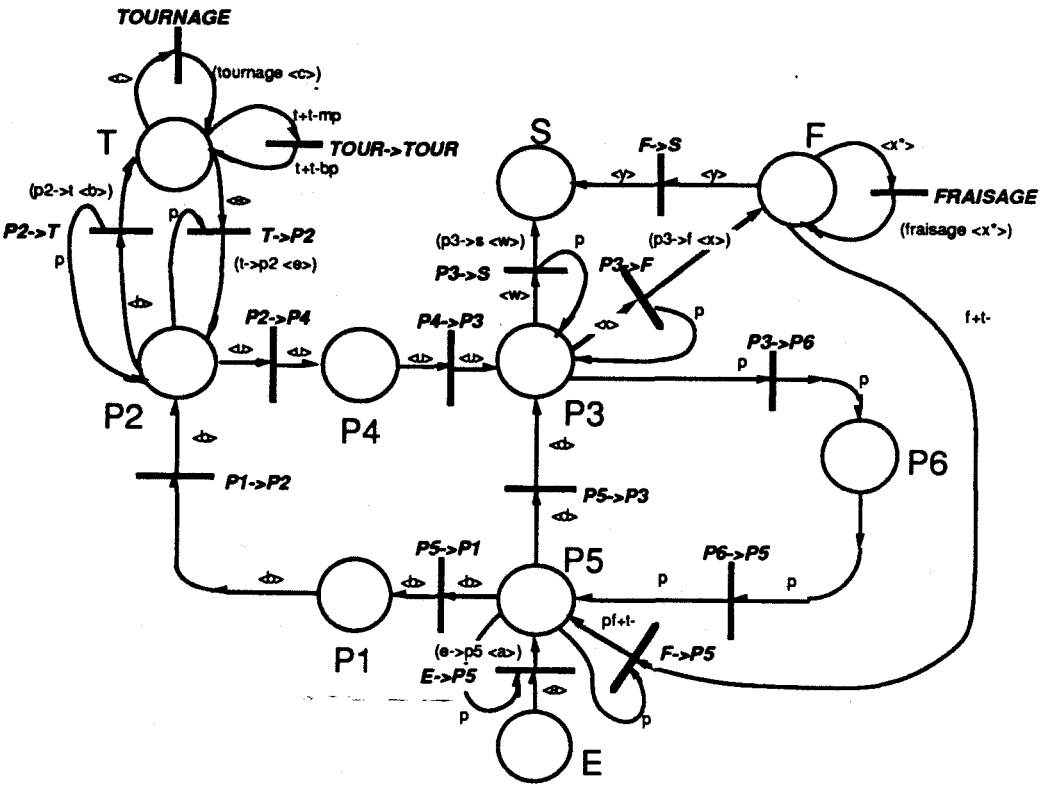


figure III.26

Cette représentation peut notamment traduire la visualisation temps réel de la simulation. Le prégraphe devient alors le synoptique de l'atelier. Lors d'un blocage ou d'une intervention de l'opérateur, le logiciel permet en outre d'effectuer des zooms sur une ou plusieurs places du prégraphe, ce qui fait apparaître une fenêtre contenant le réseau de Petri développé de la phase de structuration. Le dessin de ce dernier est réalisé de manière automatique. Le réseau de Petri développé est de trop grande taille pour permettre son affichage complet à l'écran. Pour le

prégraphe de l'exemple, composé de 10 places et de 17 transitions, le réseau de Petri développé est composé de 215 places et 126 transitions. La possibilité d'effectuer des zooms sur une places du prégraphe en phase de simulation notamment, donne ainsi une idée plus précise du lieu et de la raison d'un blocage par exemple. Le zoom sur la place T du prégraphe obtenu à partir du logiciel, est représenté sur la figure suivante.

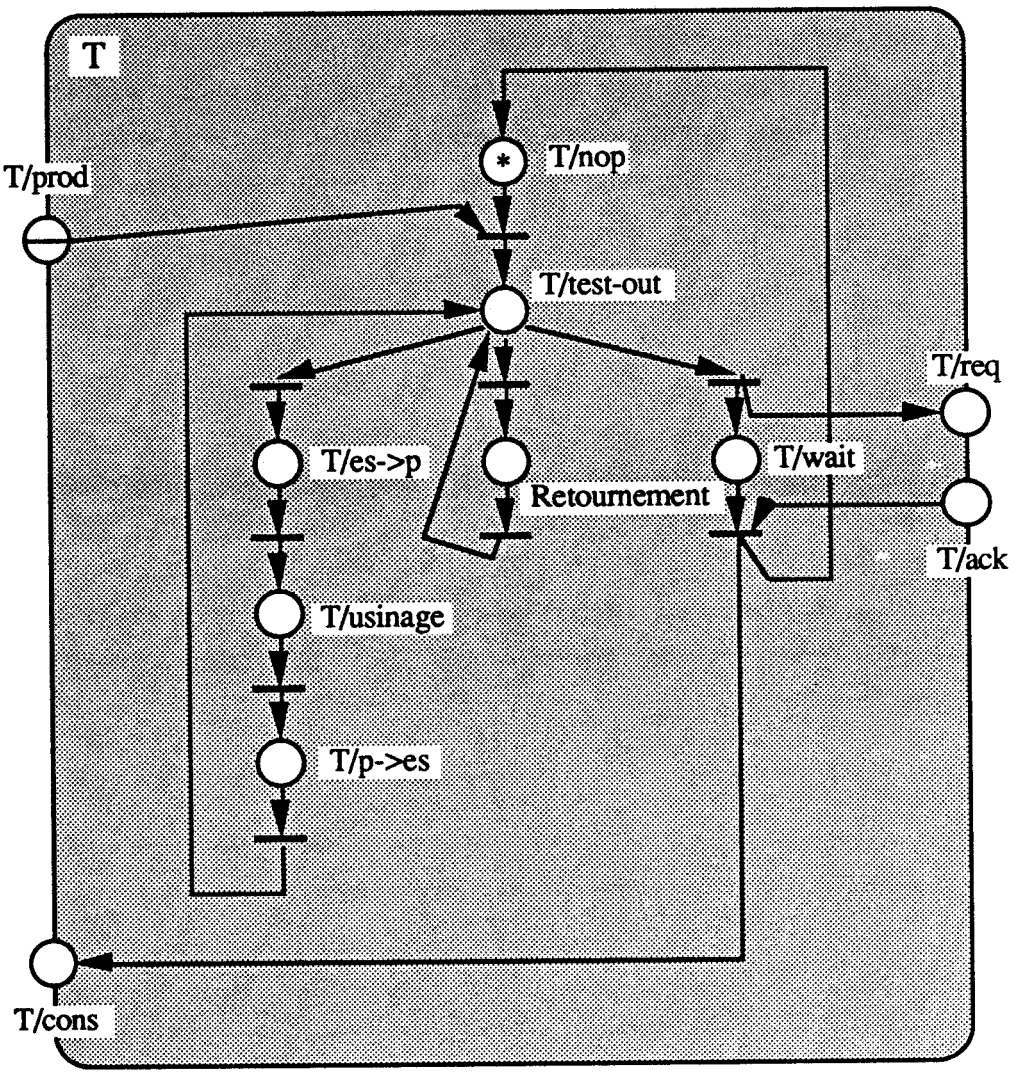


figure III.27

Dans un deuxième temps, en suivant l'évolution du projet CASPAIM, nous avons étendu l'interface graphique réalisée pour la représentation du prégraphe, à la représentation des gammes logiques et des gammes opératoires. Nous présentons cette extension dans la partie IV.

III.4. Edition et rappel d'un graphe.

A partir de la description sous forme littérale du prégraphe, la méthode est la suivante pour l'édition d'un prégraphe sélectionné :

- Lecture du fichier.

- Etablissement de la structure de données interne au logiciel sous forme d'objets présentées dans la partie II.3.3. de ce chapitre.

- Pour chaque place, positionnement par l'utilisateur de la place courante à l'écran. En effet, dans un premier temps, nous demandons, de manière interactive, le positionnement des places à l'utilisateur. Dans une seconde phase, nous avons développé une méthode de positionnement automatique, que nous présenterons dans un paragraphe suivant.

- Phase de calcul des différentes coordonnées des objets transitions et arcs.
  Le calcul des coordonnées des transitions est réalisé de la manière suivante. Les coordonnées du centre d'une transition sont les coordonnées du barycentre B du barycentre BP des places amont à la transition et du barycentre BS des places aval à la transition. L'angle que fait la transition avec l'axe des abscisses est l'angle que fait la droite passant par BP et BS avec l'axe des abscisses (cf. figure III.28).

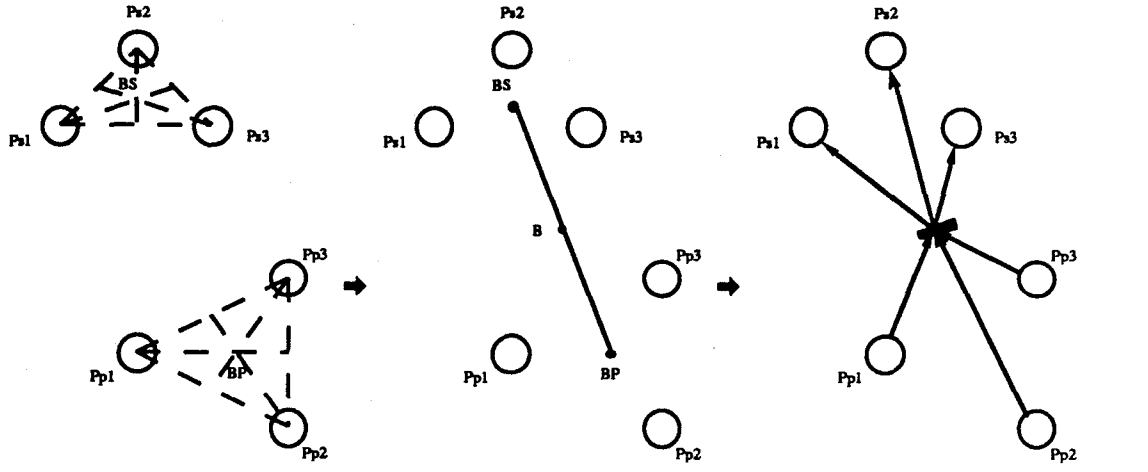


figure III.28

- Pour les liaisons, qui forment une liaison multiple, afin d'éviter le chevauchement des transitions, nous utilisons un autre mode de calcul des coordonnées. Le calcul des

coordonnées du centre des trois transitions t1, t2, et t3 reliant les places P1 et P2, s'effectue en prenant les coordonnées des points d'intersection entre la droite D perpendiculaire en B (milieu de P1 et P2) à la droite Δ passant par les deux places P1 et P2, et les droites Δ1 et Δ2 passant par P1 faisant un angle α constant avec la droite Δ passant par P1 et P2

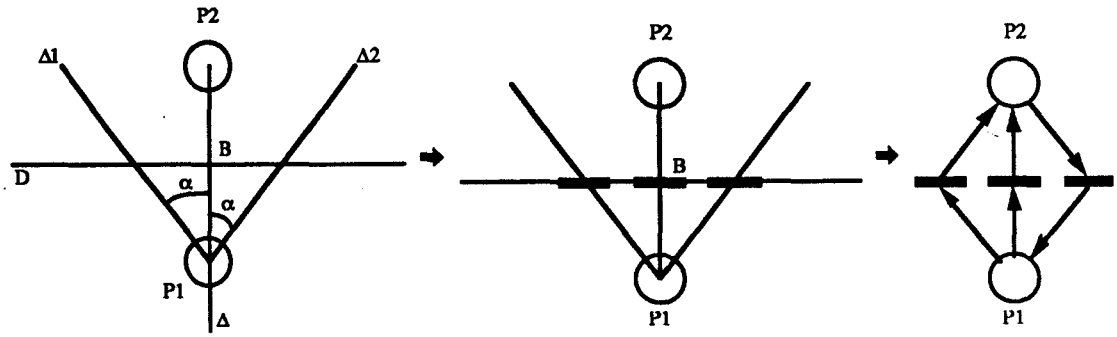


figure III.29

- Affichage automatique des arcs et transitions, ainsi que le nom des places, des transitions, et éventuellement des couleurs portées par les arcs. Nous avons offert la possibilité d'un affichage par options par sélection dans le menu de saisie. En effet, pour construire la première représentation d'un graphe, il n'est pas utile d'afficher les différents paramètres portés par celui-ci. L'utilisateur dispose des options : affichage des places uniquement, affichage des places, transitions, et arcs sans coloration, ou affichage complet du graphe.

- Une fois la saisie des places effectuée ainsi que l'affichage d'une première représentation du graphe, la possibilité de modeler le dessin est offerte à l'utilisateur. Le modelage du dessin se fait par le déplacement des places uniquement, avec mise à jour du graphe automatique. En effet, les places sont les nœuds du graphes, et nous avons pensé que pour obtenir un graphe lisible et clair, seul leur déplacement était intéressant. Cette possibilité de déplacement peut être étendue de la même manière aux transitions.

- Possibilité de sauvegarde du dessin, avec mise en mémoire de la structure de données graphique.

Après cette phase de saisie de graphe, l'utilisateur peut rappeler un graphe déjà entré. Le déroulement pour obtenir le graphe qu'il veut visualiser pour une simulation éventuelle, ou pour modifier l'aspect du graphe, se fait de manière identique à la saisie d'un graphe.

Signalons, que l'utilisateur n'a pas besoin de connaître le langage de programmation utilisé, puisqu'il se trouve dans un environnement graphique identique à ceux qu'utilisent les ordinateurs Mac Intosh, ou à ceux développés maintenant sous X-windows, ..., pour d'autres machines.

III.5. Méthode de positionnement automatique, [HEB 92].

Dans le contexte dans lequel nous travaillons qui s'inspire des ateliers de génie logiciel appliqué aux automatismes, nous avons développé un outil permettant la visualisation graphique des prégraphes, qui peut être utilisé pour toutes les étapes de la méthodologie CASPAIM. Le prégraphe constitue une description agrégée sous forme de réseau de Petri coloré, du rôle fonctionnel de l'unité de production. Il récapitule l'ensemble des cheminements élémentaires possibles et prévus pour les objets circulant dans l'atelier. C'est pourquoi, il donne une image du positionnement des différents organes réels de production. Chaque place du prégraphe représente un lieu physique opérationnel. La méthode de représentation automatique des places du prégraphe, que nous allons présenter maintenant, constitue un outil d'aide à la conception d'ateliers flexibles de production. En effet, outre le fait de représenter le prégraphe pour la phase de simulation de manière totalement automatique, la méthode va permettre d'aider le concepteur au positionnement des différentes machines de l'atelier. De plus, sur cette image de l'atelier, nous pouvons mettre en évidence, dès la première étape de la méthodologie CASPAIM1, les différents problèmes que pose le croisement des moyens de transport . Cette méthode permet aussi, dès l'étape de conception, de minimiser les trajets des pièces dans l'atelier.

III.5.1. Type de graphes.

Le but de notre travail est de représenter, de manière automatique, les différents modèles obtenus par la méthodologie CASPAIM. Ces représentations permettent de visualiser les réseaux de Petri déduits de la phase de modélisation, ainsi que la visualisation de l'évolution du marquage en phase de simulation. Dans la première version du logiciel que nous avons réalisé, le positionnement des places était demandé à l'utilisateur. Les arcs, les transitions, et toutes les informations nécessaires s'affichaient de manière automatique. Ensuite l'utilisateur pouvait modeler le graphe à volonté. Pour des représentations de modèles simples et petits en nombre de places (prégraphes composés de quatre à une quinzaine de places), cela donne des résultats satisfaisants. Par contre, pour des graphes de

plus grandes dimensions, la phase de modelage est souvent complexe. L'automatisation du positionnement des places nous est apparue alors nécessaire. Différents problèmes se sont alors posés. D'une part, il fallait que le graphe obtenu soit lisible, qu'il y ait peu de mouvements ultérieurs, et d'autre part, qu'il tienne compte le plus possible de la réalité physique du procédé. Rappelons que chaque place du prégraphe représente un lieu physique opérationnel. Ajoutons par ailleurs, qu'il n'apparaît pas toujours d'ordre partiel sur les places de ces graphes. En effet, des circuits apparaissent dans les prégraphes, dus pour l'essentiel à la modélisation des moyens de transport.

C'est pourquoi, nous considérons que les graphes étudiés font partis d'un ensemble "CE" de graphes qui peut être scindé en deux classes distinctes. La première classe "CC", regroupe des graphes comportants au moins un circuit, la deuxième "CL", des graphes que nous appellerons linéaires.

Les prégraphes appartiennent de manière générale à la première classe "CC". En effet, rappelons que la flexibilité d'un atelier se caractérise par différents critères :

- un même poste de transformation peut réaliser plusieurs opérations distinctes,
- un même objet manufacturé peut être réalisé selon des séquences d'opérations différentes,
- plusieurs objets indépendants entre eux peuvent cohabiter au sein du même système de production,
- et enfin, un même objet peut suivre différents trajets pour atteindre les postes de transformation au sein du système de production.

Donc, la flexibilité d'une cellule est obtenue par la flexibilité des opérations à effectuer sur les différents produits d'une part, et par l'utilisation de moyens de transport permettant d'effectuer des retours arrières lors de pannes, un routage des pièces différents dû à la saturation momentané d'une machine, ..., d'autre part. Sur l'exemple que nous avons pris, nous voyons apparaître trois de ces critères. L'objet de type t-f- peut être tourné puis fraisé, ou fraisé puis tourné. Les cinq gammes, indépendantes entre elles, cohabitent au sein de la même cellule de production. Enfin, les objets peuvent suivre des trajets différents sur le convoyeur. C'est pourquoi, les prégraphes appartiennent à la classe "CC", puisqu'ils sont le reflet d'ateliers **flexibles** de production.

Les gammes logiques et les gammes opératoires de CASPAIM2, appartiennent à la deuxième classe "CL" de graphes. En effet, les gammes logiques décrivent la séquence et la synchronisation des fonctions à appliquer à chaque produit. Les gammes opératoires, quant à elles, décrivent la succession des états d'avancement d'une pièce, ainsi que la succession

des lieux opératoires par lesquels elles transitent. La séquentialité de ces deux modèles apparait nettement, ce qui donne l'aspect linéaire de leur représentation (cf. figure III.30). L'automatisation de la représentation utilise alors, l'ordre partiel qui existe entre les différentes places de ces graphes. C'est pourquoi, nous considérons que la représentation automatique des graphes de la classe "CL" est un cas particulier de celle de la classe "CC".

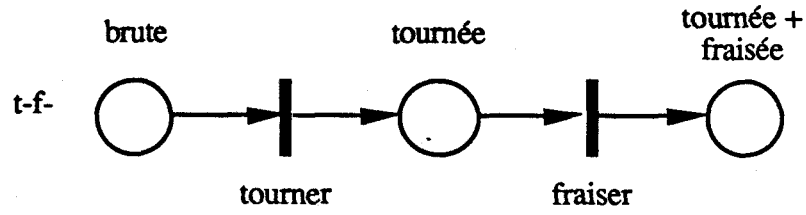


figure III.30

III.5.2. Règles de dessin.

La première idée a été de s'inspirer des méthodes de la théorie des graphes et de la recherche opérationnelle, [DRO 87], [KAU 68], [ROS 86], avec l'utilisation de matrices comme modèles de représentation interne. En effet, nous avons essayé d'appliquer pour la résolution du positionnement automatique, les méthodes d'obtention de circuit basées sur une représentation matricielle des graphes, l'algorithme de Ford pour déterminer les chemins de longueur minimale, et l'algorithme de Kruskal qui permet de déterminer un arbre recouvrant minimal, et en ajoutant certaines règles propres au domaine de notre recherche. Si les résultats étaient satisfaisants, la résolution du déplacement des nœuds d'un graphe était fondée sur une méthode empirique. Ce qui nous a poussé à rechercher une méthode plus pragmatique.

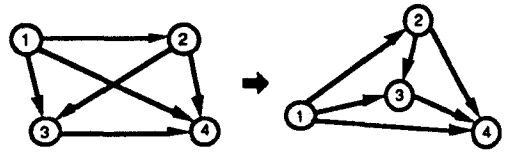
La deuxième idée a été de s'inspirer des logiciels d'aide à la fabrication de circuits imprimés, par exemple. Mais, ces derniers n'utilisent pas la même approche et n'ont pas le même objet. Pour ce type d'outil, il est essentiel que les liaisons entre les différents composants d'un circuit imprimé ne se coupent pas, [GOT 86], [ROS 86], [TRI 87], [ZOB 89]; et de par la même, l'établissement des liaisons est prioritaire sur le positionnement relatif des composants. Il n'en reste pas moins, que si le choix de la position de chaque composant est important, c'est en général le concepteur du circuit imprimé qui établit le premier positionnement. Les liaisons sont établies ensuite automatiquement, puis, si la solution n'est pas convenable, le logiciel apporte une aide au concepteur afin d'arriver à un résultat que valide celui-ci. C'est le cas pour les logiciels CAE de Metadesign et OrCAD/SDT de Systems Corporation, notamment.

Nous avons utilisé cette approche pour représenter les prégraphes, mais comme nous l'avons dit précédemment, quand le nombre de places devient important, la phase de modelage devient lourde. De plus, le cahier des charges, entrée de la méthodologie CASPAIM2, ne comporte pas forcément la description des différents organes de production, et a fortiori, nous ne disposons pas de leur positionnement relatif, or, le prégraphe est le reflet de l'atelier.

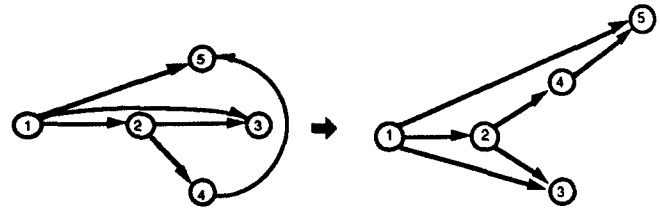
La méthode que nous proposons, va permettre, outre le fait de représenter le prégraphe de manière totalement automatique, d'aider le concepteur au positionnement des différentes machines de l'atelier. Le fait qu'il n'existe pas, à notre connaissance, de logiciel de représentation automatique de graphes, nous a conduit à développer une solution originale. Les graphes que nous considérons ici, pour l'exploitation de notre méthode, appartiennent tous à la classe "CC".

Il est souvent difficile de dessiner un graphe qui soit clair et facile à étudier lorsque le nombre de places et de transitions devient important. Avec de l'expérience, on apprend à "avoir l'œil" pour construire les graphes, et les règles suivantes, dans la pratique, ont montré leur efficacité, [BAT 67].

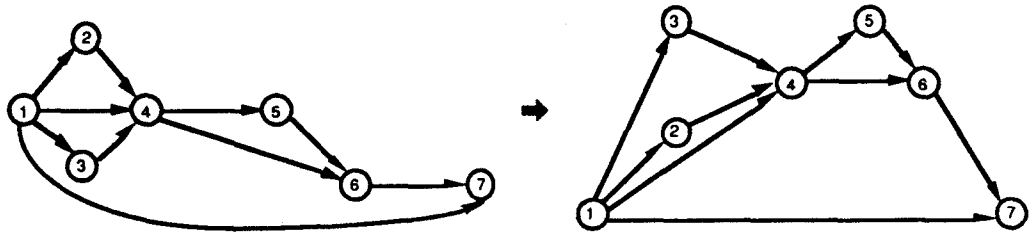
1. Eviter de tracer des arcs se coupant.



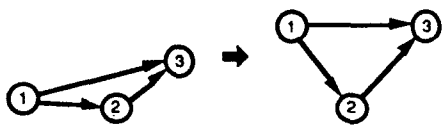
2. Essayer de tracer des arcs rectilignes.



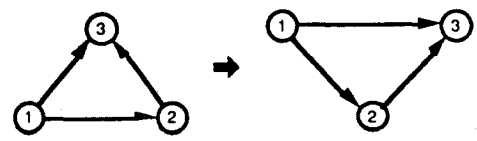
3. Eviter des variations trop grandes dans les longueurs des arcs.



4. Faire en sorte que les arcs fassent des angles aussi grands que possible.



5. Essayer de toujours tracer un arc de la gauche vers la droite.



figures III.31

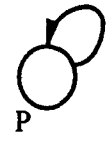
Examinons la mise en œuvre de ces règles. La règle 5 est difficilement applicable pour des graphes appartenant à la classe "CC". A l'inverse, pour des graphes de la classe "CL", cette règle est la première à appliquer. C'est pourquoi, afin de respecter la règle 5, nous effectuons, dans un premier temps, l'analyse du fichier initial décrivant le graphe, afin de différencier la classe à laquelle le graphe à représenter appartient. La deuxième étape va permettre de prendre en compte les règles 2, 3 et 4. Pour cela, nous définissons une structure rectangulaire, variant avec le nombre de places du premier circuit à positionner. Elle va permettre d'éviter de trop grandes variations dans les longueurs des arcs, de faire en sorte que les arcs fassent des angles aussi grand que possible, et de tracer des arcs rectilignes. La règle 1 va être appliquée après la première étape de positionnement du circuit le plus grand, par ajout des autres circuits et places n'appartenant à aucun circuit, en effectuant éventuellement un déplacement des places déjà positionnées.

III.5.3. Termes de vocabulaire.

Avant de détailler les différentes étapes de la méthode, énonçons quelques termes du vocabulaire que nous utilisons.

* Un **circuit** est un chemin qui se referme sur lui-même.

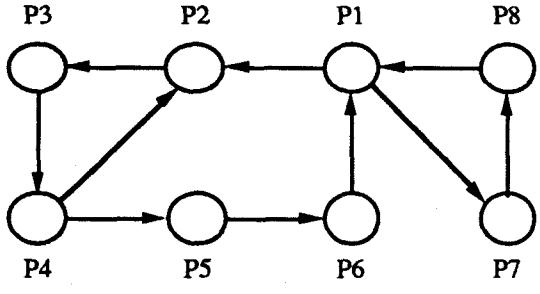
* Une **boucle** est un circuit de longueur 1.



* Un **aller-retour** est un circuit composé de deux places.

* Un **circuit pur** est un circuit composé d'au moins trois places.

* Un circuit pur est **indépendant** si et seulement si il n'est pas sous-circuit d'un autre circuit pur.



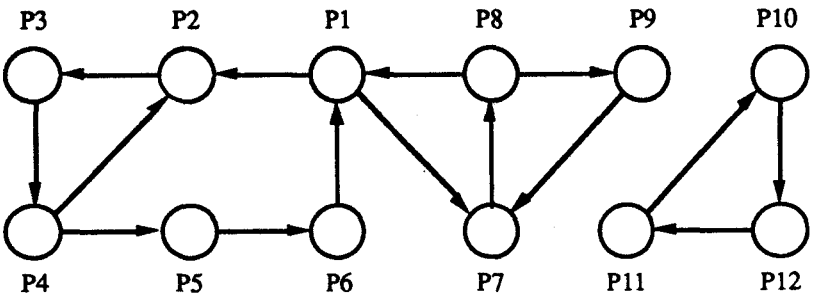
Les circuits (P1 P2 P3 P4 P5 P6) et (P1 P7 P8) sont indépendants.

Le circuit (P2 P3 P4) est "dépendant".

* Deux circuits purs indépendants appartiennent à une même **famille** si ils ont au moins une place en commun.

Le graphe suivant est composé de trois familles :

- la Fam-1 = {(P1 P2 P3 P4 P5 P6), (P1 P7 P8)}
- la Fam-2 = {(P1 P7 P8), (P7 P8 P9)}
- la Fam-3 = {(P10 P11 P12)}



* Un aller-retour est de **type 0** si aucune des places qui le compose n'appartient à un même circuit pur.

* Un aller-retour est de **type 1** si une et une seule de ses places appartient à un circuit pur.

* Un aller-retour est de **type 2** si les deux places qui le compose appartiennent à un même circuit pur.

III.5.4. Analyse du graphe.

L'exemple que nous prenons pour illustrer la méthode, est le prégraphe de la cellule présentée au début de cette partie.

Comme nous l'avons dit précédemment, le type de graphes que nous considérons appartiennent à la classe "CC". Pour vérifier à quelle classe appartient le graphe que nous voulons représenter, nous effectuons l'analyse du fichier contenant la description littérale du graphe. Nous présentons les liaisons formant le prégraphe de l'exemple dans le tableau récapitulatif ci-dessous (cf. figure III.32). Le fichier de ce graphe, présenté sous forme d'objets, se trouve en Annexe III.1.

Places	E	P5	P6	P1	P2	P4	T	P3	F	S
Prédécesseur		E P6 F	P3	P5	P1 T	P2	T T P2	P4 P5	F P3	F P3
Successeur	P5	P3 P1	P5	P2	P4 T	P3	T T P2	P6 F S	F P5 S	

figure III.32

Nous avons réalisé, pour l'analyse du prégraphe, un programme qui nous permet d'établir la liste de tous les circuits de manière automatique, puis, à partir de cette liste, d'extraire diverses propriétés de ces circuits. Notamment, il permet de créer des ensembles de circuits : l'ensembles des aller-retours, l'ensembles des circuits purs, l'ensemble des familles, ... Le fichier résultat de cette analyse pour l'exemple est présenté ci-dessous et le listing du programme en Annexe III.4.

Listes de places.

Liste des places du prégraphe
places = (E P5 P1 P2 T P4 P3 F S P6)

Liste des places ayant une boucle
l-pl-boucle = (F T)

Liste des places n'appartenant à aucun circuit
l-pl-cir-bar = (E S)

Liste des places appartenant à au moins un circuit
l-pl-cir = (P5 P1 P2 T P4 P3 F P6)

Liste des places appartenant à un aller-retour mais à aucun circuit pur
l-pl-cp-ar-bar = (T)

Listes de circuits.

Liste des circuits
l-cir = (C-1 C-2 C-3 C-4 C-5)
C-1 = (P2 T)
C-2 = (P5 P1 P2 P4 P3 F)
C-3 = (P5 P1 P2 P4 P3 P6)
C-4 = (P5 P3 F)
C-5 = (P5 P3 P6)

Liste des aller-retours
l-all-ret = (C-1)

Liste des aller-retours dont aucune des places n'appartient à un circuit pur
l-all-ret-ty0 = ()

Liste des aller-retours dont une place appartient à un circuit pur
l-all-ret-ty1 = (C-1)

Liste des aller-retours dont les deux places appartiennent à un circuit pur
l-all-ret-ty2 = ()

Liste des circuits purs
l-cir-pur = (C-2 C-3 C-4 C-5)

Liste des circuits purs indépendants
l-cir-pur-ind = (C-2 C-3)

Liste des circuits purs non indépendants
l-cir-pur-ind-bar = (C-4 C-5)

Liste des familles
l-famille = (fam-1)
fam-1 = (C-2 C-3)

III.5.5. Structure de base.

L'idée directrice de notre méthode est de positionner les places du plus grand circuit, puis les autres places.

L'analyse de la topologie d'une cellule montre, qu'autour d'un système de transport, de type convoyeur, gravitent toutes les machines et robots de chargement/déchargement. D'autre part, l'analyse d'un prégraphe permet de montrer que, de manière générale, les

circuits représentent les différents tronçons d'un convoyeur. C'est pourquoi, il est bon de positionner en premier, le plus grand circuit.

Afin de respecter les règles 2, 3 et 4, nous avons choisi d'utiliser comme ossature, pour le positionnement du circuit principal, la forme rectangulaire. Cela permet, avant le positionnement à l'écran, l'utilisation d'une structure de données interne de type tableau. De plus, les écrans sont rectangulaires ce qui permet d'obtenir des graphes mieux centrés. Le choix d'une telle structure nous permet aussi d'établir une règle récurrente "simple" qui assure la résolution **automatique** de notre problème. Ce choix, d'après les propriétés géométriques des rectangles, nous satisfait, sachant que nous aurions pu prendre un autre type de figure géométrique qui satisfasse encore mieux la règle 4. Cette possibilité est envisagée, et nous l'étudirons dans la partie III.7. Enfin, cette structure rectangulaire permet, connaissant le nombre de lignes et de colonnes utilisées pour le positionnement des places et les dimensions de l'écran, de déterminer les coordonnées du centre des places afin de permettre leur affichage réel, avant d'établir automatiquement les liaisons selon la manière décrite dans la partie III.4 de ce chapitre. Représentons sur la figure III.28, la corrélation qui existe entre la structure interne de type tableau et l'affichage du résultat sur l'écran.

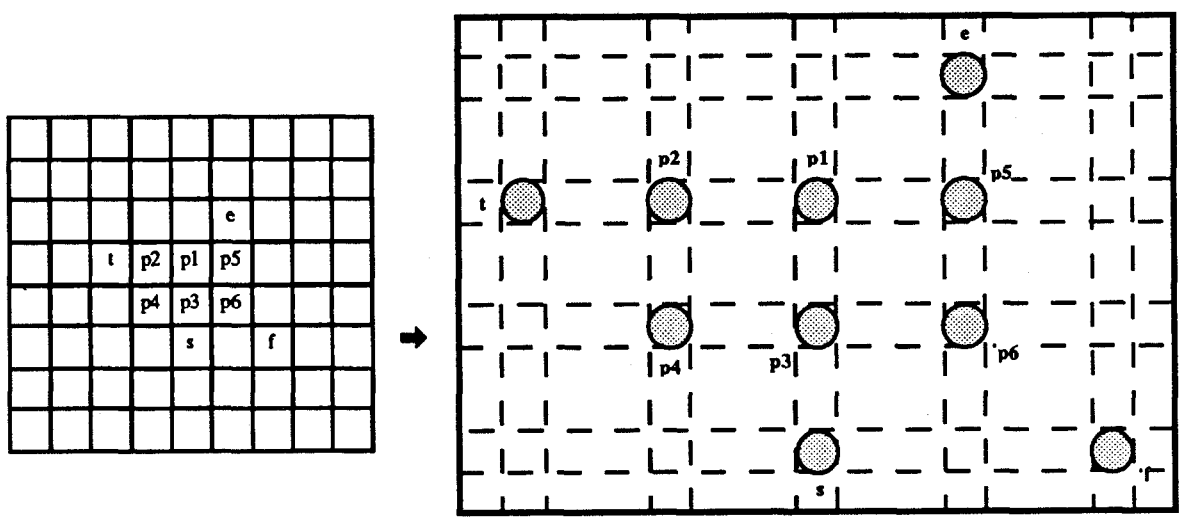


figure III.33

Pour déterminer, selon le nombre n de places composant le circuit, le nombre de lignes ℓ et de colonnes c dont nous avons besoin, de manière automatique, nous adoptons une règle récurrente qui privilégie le nombre de colonnes par rapport au nombre de lignes, en conservant des proportions équivalentes à celles de l'écran. C'est pourquoi, le nombre de colonnes c a été fixé à $[\ell, 2\ell[$.

Pour deux places à positionner, l'hypothèse de départ (l'étape 1), est que le nombre de colonnes c soit égal à 2 et le nombre de lignes ℓ à 1. Le passage de l'étape p-1 à l'étape p s'effectue par l'ajout d'une colonne si $c_p \in [\ell, 2\ell[$, sinon par l'ajout d'une ligne.

La règle récurrente s'énonce de la manière suivante :

Soit n le nombre de places, p l'étape, avec

si $n = 2p$ ➡ étape = p,

si $n = 2p + 1$ ➡ étape = $p + 1$

Pour l'étape 1 :

$\ell_1 = 1,\ c_1 = 2$

Cas général :

si $\dfrac{(c_{p-1} + 1)}{2} < \ell_{p-1}$

alors $\ell_p = \ell_{p-1},\ c_p = c_{p-1} + 1$

si $\dfrac{(c_{p-1} + 1)}{2} \geq \ell_{p-1}$

alors $\ell_p = \ell_{p-1} + 1,\ c_p = c_{p-1}$

Le tableau récapitulatif de la figure III.34 montre l'application de la règle récurrente jusqu'à la sixième étape.

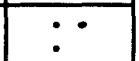
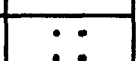
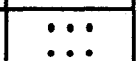
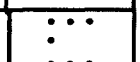
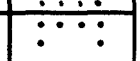
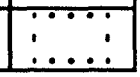
étapes	$\frac{c_{i-1}+1}{2}$	ℓ_{i-1}	c_i	ℓ_i	n	dessin
1			2	1	2	
2	1,5	1	2	2	3	
					4	
3	1,5	2	3	2	5	
					6	
4	2	2	3	3	7	
					8	
5	2	3	4	3	9	
					10	
6	2,5	3	5	3	11	
					12	

figure III.34

Connaissant le nombre de lignes et de colonnes nécessaires au positionnement du premier circuit, le "remplissage" du tableau s'effectue dans le sens trigonométrique, en balayant la liste des places du circuit.

III.5.6. Méthode.

Le but de la méthode est de permettre le positionnement des places d'un réseau de Petri appartenant à la classe "CC", de manière automatique, afin d'obtenir un graphe clair et proche d'une solution finale convenable. Pour cela, nous avons analysé les règles de représentation présentées figure III.31, et en avons déduit la méthode suivante, (l'algorithme complet est présenté en Annexe III.5).

1. Positionner le ou les circuits purs indépendants.
2. Positionner les aller-retours de type 2.
3. Positionner les circuits purs non indépendants.
4. Positionner les aller-retours de type 1.
5. Positionner les places qui n'appartiennent à aucun circuit.
6. Positionner les aller-retours de type 0.

1. Etape 1 : positionnement du ou des circuits purs indépendants.

Ce positionnement est réalisé suivant plusieurs critères.

Le premier critère pris en compte est le nombre de familles obtenues après la phase d'analyse du graphe considéré. Dans le cas où il y a plusieurs familles, il convient de regarder si elles ont des circuits purs indépendants en commun, (cf. § III.5.3, où la fam-1 et la fam-2 ont (P1 P7 P8) en commun). Nous ne détaillerons pas le cas où le nombre de familles est supérieur à un. Ce cas ne présente aucune difficulté supplémentaire, et se traite de manière similaire au cas que nous présentons. Pour l'exemple de la cellule de Centrale Lille, le résultat de la phase d'analyse ne comporte qu'une famille, fam-1 = (C-2 C-3), avec C-2 = (p5 p1 p2 p4 p3 f) et C-3 = (p5 p1 p2 p4 p3 p6).

Le deuxième critère, s'il n'existe qu'une seule famille, est le nombre de places qu'ont en commun les circuits pris deux à deux. Suivant le nombre de places qu'ils ont en commun, le premier positionnement s'effectuera pour un ou pour tous les circuits de la famille.

Si deux circuits c1 = (p1 p2 p3 p4 p5 p6) et c2 = (p6 p7 p8) ont une seule place en commun, le premier positionnement s'effectuera pour ces deux circuits. L'application de la loi récurrente pour le circuit c1 donne pour le nombre de colonnes $c_1 = 3$, et pour le nombre de lignes $l_1 = 2$, respectivement pour le circuit c2, $c_2 = 2$, et $l_2 = 2$. La place p6 qu'ils ont en commun sera positionnée sur la première colonne et la première ligne, dans le sens courant de lecture, du premier circuit rencontré. Le deuxième circuit pourra subir une rotation. Suivant l'ordre des deux circuits dans la liste, pour cet exemple, deux solutions peuvent être trouvées pour le positionnement de ces circuits, (cf. figure III.35).

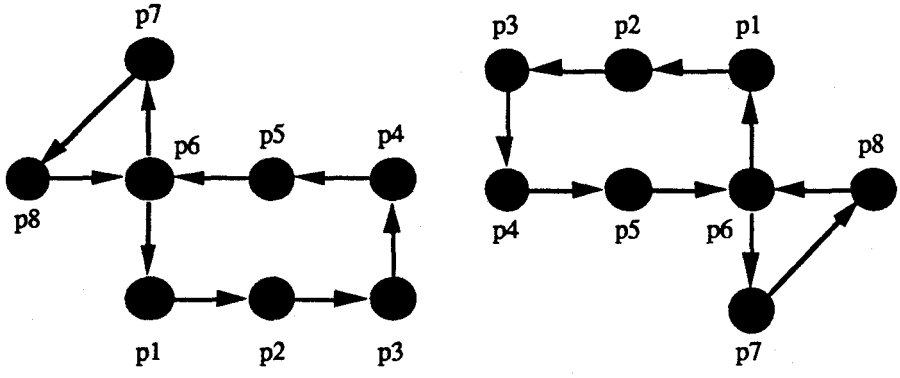


figure III.35

Sinon, dans le cas où les circuits de la famille ont plus d'une place en commun, c'est le cas de l'exemple de la cellule de Centrale Lille, les circuits C-2 et C-3 ont cinq places en commun, le positionnement s'effectuera pour un circuit. Le critère de choix de ce circuit à positionner, est de prendre le circuit qui est composé du plus grand nombre de places, le premier dans la liste, en cas d'égalité. Pour l'exemple, le premier circuit à positionner est C-2, car C-2 et C-3 sont composés du même nombre de places, et que C-2 est le premier dans la liste. Le nombre n de places de C-2 est 6, le résultat de l'application de la règle récurrente, pour définir le nombre de lignes et de colonnes nécessaires à l'affichage de C-2, nous donne $l = 2$, $c = 3$.

Les autres circuits purs indépendants de la famille sont ensuite positionnés. Pour cela, nous avons établi une règle pour expanser le graphe déjà représenté. Nous avons défini, autour de la structure rectangulaire initiale, un contour enveloppe, avec des positions bien précises. Le calcul des positions du contour enveloppe est réalisé de la manière suivante :

Soient ℓ l'indice ligne des positions de la structure initiale avec $\ell \in [1\ \ell_{max}]$,
et c l'indice colonne des positions de la structure initiale avec $c \in [1\ c_{max}]$,

soient ℓ_c et c_c les indices ligne et colonne de la position du contour enveloppe correspondante,

si $\ell = 1 \Rightarrow \ell_c = \ell - 1$ si $c = 1 \Rightarrow c_c = c + 1$
$\ell = \ell_{max} \Rightarrow \ell_c = \ell + 1$ $c = c_{max} \Rightarrow c_c = c - 1$
$1 < \ell < \ell_{max} \Rightarrow \ell_c = \ell$ $1 < c < c_{max} \Rightarrow c_c = c$

Le sens de parcours du contour enveloppe se fait dans le sens inverse au sens trigonométrique. Le résultat du calcul des positions du contour enveloppe pour des structures initiales composées 3 à 9 places est présenté sur la figure III.36.

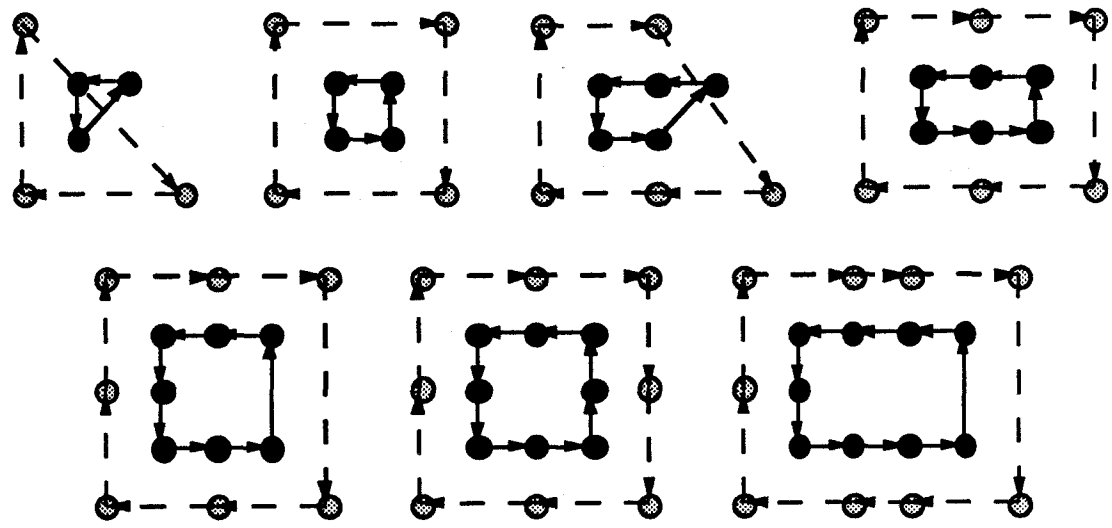


figure III.36

Le but est de représenter alors les circuits purs indépendants de la famille non encore positionnés. Pour chaque circuit, nous regardons toutes les places qui le composent. Si la place, non encore positionnée, est en liaison avec deux places déjà positionnées, la ou les places déjà positionnées entre les deux autres, sont bougées vers l'extérieur, sur la position du contour enveloppe correspondante, ou la position contigüe si la position correspondante est déjà occupée. Pour l'exemple, C-2 = (p5 p1 p2 p4 p3 f) est positionné. Le circuit pur indépendant de la famille non encore positionné est C-3 = (p5 p1 p2 p4 p3 p6). La seule place non encore placée est p6. P6 est en liaison avec p5 et p3, comme f. Pour placer p6, f est déplacée vers une position du contour enveloppe. Pour l'exemple, les étapes du positionnement des circuits

purs indépendants de la famille sont représentées sur la figure III.37.

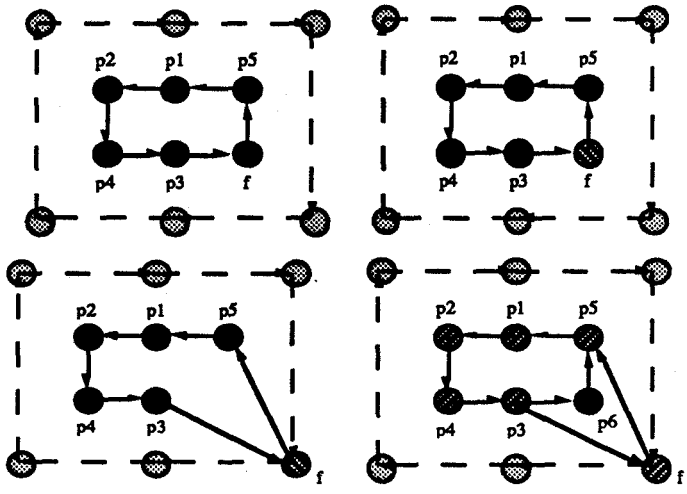


figure III.37

2. Etape 2 : positionnement des aller-retours de type 2.

L'étape 2, positionnement des aller-retours de type 2, consiste, non pas à positionner réellement les deux places p_i et p_j, puisqu'elles le sont déjà, mais, à vérifier si elles sont ou ne sont pas adjacentes. En effet, si une ou plusieurs places se trouvent entre p_i et p_j, nous vérifions que $c_i \neq c_j$ et $\ell_i \neq \ell_j$. Si cette condition n'est pas vérifiée, les places intermédiaires sont bougées vers l'extérieur. Dans l'exemple suivant, considérons qu'il existe deux aller-retours de type 2, all-ret1 = (p1 p3) et all-ret2 = (p3 p5). Pour le positionnement de all-ret2, nous pouvons effectuer la liaison car $c_3 \neq c_5$ et $\ell_3 \neq \ell_5$. Par contre, ce n'est pas le cas pour all-ret1, car $\ell_1 = \ell_3$. En effet, p2 se trouve entre p1 et p3. Nous effectuons donc, le déplacement de p2 vers une position du contour enveloppe.

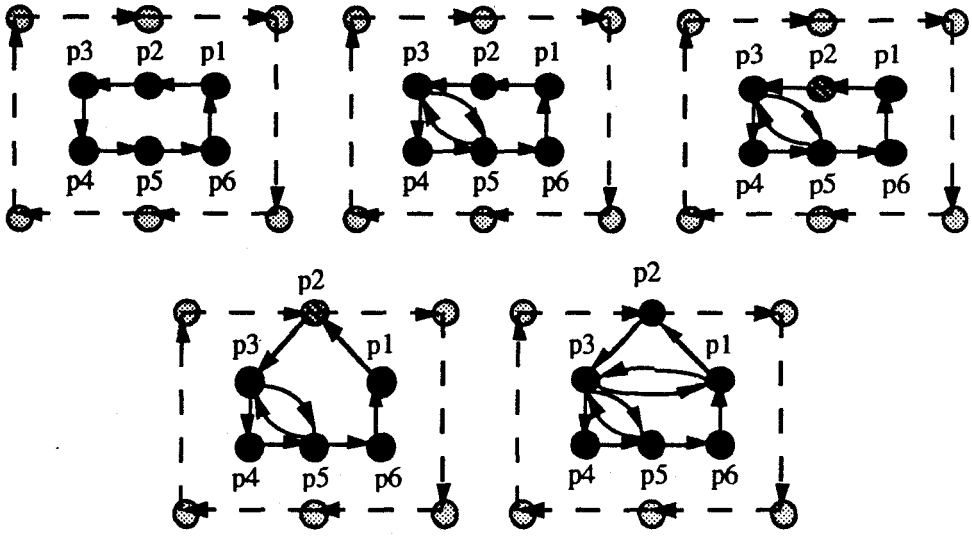


figure III.38

3. Etape 3 : positionnement des circuits purs non indépendants.

Le positionnement des circuits purs non indépendants permet, de même qu'à l'étape 2, de vérifier les indices ligne/colonne des places liées par des liaisons n'appartenant pas aux circuits purs indépendants. Les circuits C-4 = (p5 p3 f) et C-5 = (p5 p3 p6) sont "positionnés" comme suit, après vérification des indices ligne/colonne de p5 et p3, puisque les liaisons p3-p6, p6-p5, p3-f, et f-p5 sont déjà placées.

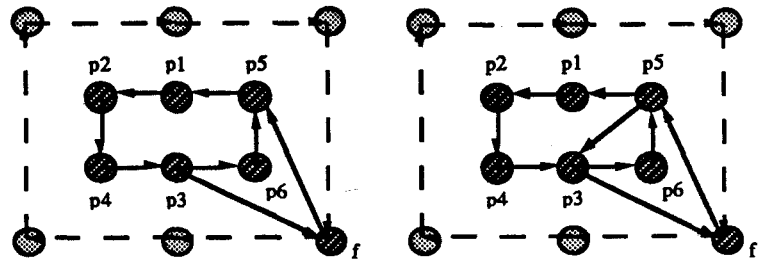


figure III.39

4. Etape 4 et 5 : positionnement des places n'appartenant à aucun circuit pur.

L'étape 4 permet le positionnement de la place appartenant à un aller-retour de type 1 qui n'appartient pas aux circuits purs déjà positionnés. De même, l'étape 5 permet de positionner les places qui n'appartiennent à aucun circuit mais qui sont en relation avec des places déjà positionnées. Le positionnement de ces places va s'effectuer sur le contour enveloppe, positionnement de t ∈ C-2 = (p2 t) pour l'étape 4, et de e et s pour l'étape 5. Précisons que le contour enveloppe, pour ces deux étapes, est enrichi de positions supplémentaires.

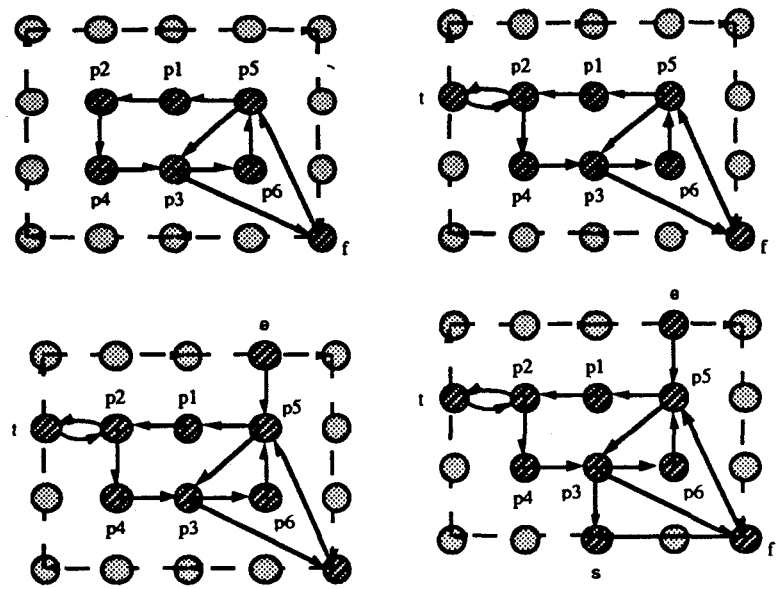


figure III.40

5. Etape 6 : positionnement des aller-retours de type 0.

Cette étape va permettre de placer les dernières places non encore positionnées. Nous regardons, en premier lieu, si les deux places qui composent l'aller-retour sont déjà placées. Si c'est le cas, nous effectuons une vérification identique à celle de l'étape 2. Si une des deux places est positionnée, le placement de la place à rajouter se fera comme pour l'étape 4, avec en plus, le placement des places n'appartenant à aucun circuit, qui ne sont pas encore positionnées et qui sont en relation avec des places positionnées. Enfin, si aucune des deux places n'est positionnée, nous effectuons le positionnement de ce circuit "simple" de la même manière que pour le positionnement de l'étape 1. Les dernières places sont ensuite positionnées.

III.5.6. Finitions.

Toutes les places sont positionnées. Les liaisons que nous avons fait apparaître, pour une meilleure compréhension de la méthode dans les différents exemples, ne sont en fait pas représentées. C'est après le "positionnement" des places dans la structure interne, que s'effectue d'une part l'affichage effectif des places à l'écran, puis l'affichage des arcs et transitions est ensuite réalisé d'autre part. La représentation "finale" du prégraphe pris pour l'exemple de la cellule présenté figure III.26 sera le suivant.

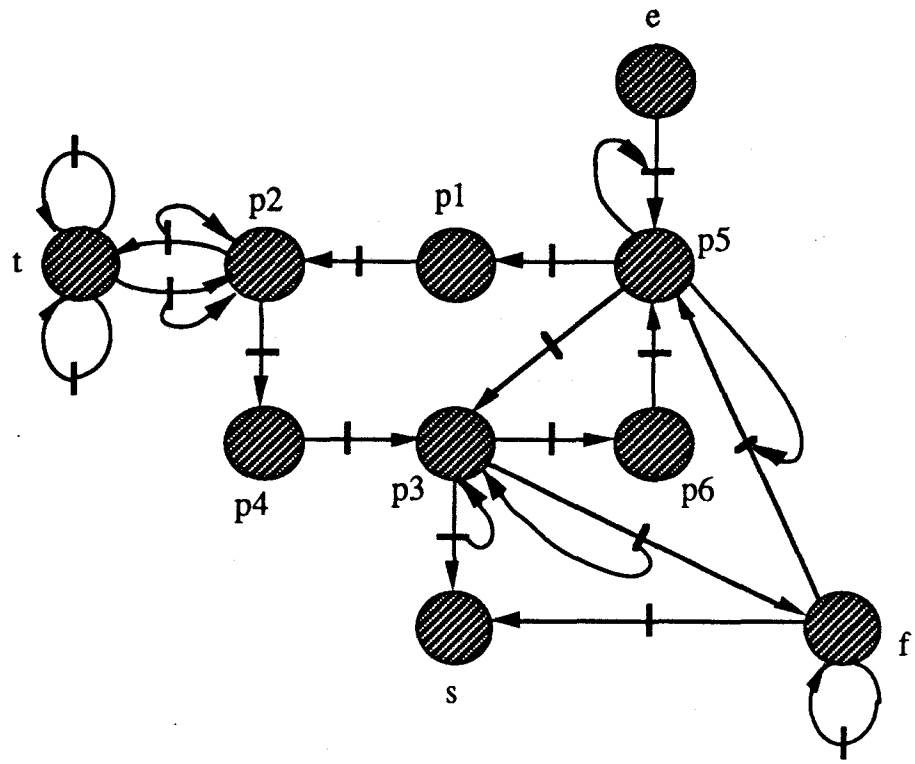


figure III.41

La méthode de positionnement automatique, nous permet d'obtenir des graphes satisfaisants et clairs. Toutefois, si l'utilisateur veut apporter des modifications au graphe, il en a la possibilité. Ces modifications peuvent être effectuées soit manuellement, soit automatiquement par symétrie et rotation. De plus, l'existence de méthodes d'optimisation du positionnement à partir d'une solution par la méthode du *recuit simulé* [HEC 90], par exemple, nous fait penser qu'il est possible de proposer d'autres versions du graphe de manière automatique, et nous conforte dans l'idée qu'il est préférable, pour l'application de ces méthodes, de partir d'une solution proche de "l'idéal", ce qu'apporte notre approche.

III.5.7. Maille hexagonale.

La méthode de positionnement automatique utilise comme structure de base la forme rectangulaire, or dans le cadre de la recherche d'une solution optimale, il est possible que cette structure ne soit pas la meilleure. C'est pourquoi, nous avons pensé utiliser une autre structure, et nous nous proposons de présenter le positionnement du premier circuit dans une maille hexagonale.

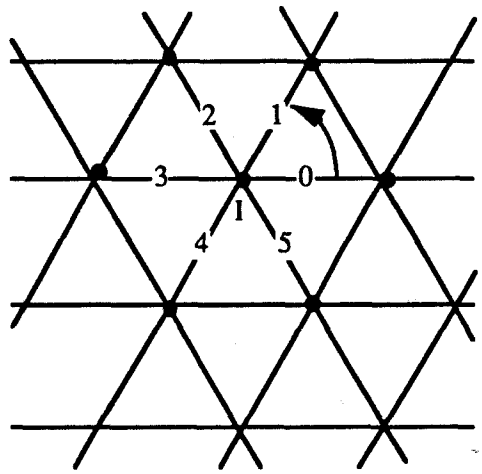


figure III.42

Toutes places i peut avoir six places voisines, indicées de 0 à 5. Afin de connaître l'emplacement des n places du circuit de manière automatique, nous utilisons une structure de données intermédiaire de type tableau, ce tableau T_{ij} ($i = 1$ à n, $j = 0$ à 5), indiquant les relations de voisinage. La réciprocité de voisinage entre la place i et une place m, se traduit par $T[i;j] = m$ et $T[m;k] = i$ avec $k = (j + 3) \bmod 6$. Une initialisation à 0 du tableau est effectuée, hormis pour les cases T[1;0] et T[2;3] dont les valeurs sont respectivement 2 et 1. Ensuite, le remplissage du tableau s'effectue ligne par ligne. La méthode de remplissage

d'une ligne i se fait de la façon suivante :

Pour inclure un élément k voisin de i et de m, eux-même voisins avec $i < m$, dans la ligne i, nous commençons par la recherche de la case où intercaler le voisin commun k. Pour cela, nous cherchons l'indice j du plus petit élément m strictement positif de la ligne, ayant une case voisine à 0. Ensuite, nous effectuons les affectations suivantes :

Case voisine de gauche à 0

- inscrire k dans T[$i;t$], avec $t = (j + 5) \bmod 6$
- inscrire k dans T[$m;l$]
  avec m = T[$i;j$] et $l = (3 + j + 1) \bmod 6$
- inscrire i dans T[$k;p$], avec $p = (3 + t) \bmod 6$
- inscrire m dans T[$k;q$], avec $q = (3 + l) \bmod 6$

Case voisine de droite à 0

- inscrire k dans T[$i;t$], avec $t = (j + 1) \bmod 6$
- inscrire k dans T[$m;l$]
  avec m = T[$i;j$] et $l = (3 + j - 1) \bmod 6$
- inscrire i dans T[$k;p$], avec $p = (3 + t) \bmod 6$
- inscrire m dans T[$k;q$], avec $q = (3 + l) \bmod 6$

Dans le cas du remplissage de la dernière case de la ligne i, il est nécessaire, en plus des affectations précédentes, d'indiquer que l'élément k est voisin de l'autre élément s de la case contigüe, mais qui n'est pas en relation avec celui-ci. Les affectations supplémentaires sont les suivantes :

s à droite de k

- inscrire -k dans T[|s| ;u],
  avec s = T[$i;v$] et $u = (3 + v + 1) \bmod 6$
- inscrire -|s| dans T[$k;w$]
  avec $w = (3 + u) \bmod 6$

s à gauche de k

- inscrire -k dans T[|s| ;u],
  avec s = T[$i;v$] et $u = (3 + v - 1) \bmod 6$
- inscrire -|s| dans T[$k;w$]
  avec $w = (3 + u) \bmod 6$

Le tableau T, pour un circuit composé de 25 places, est donné en Annexe III.8.

Une fois que le tableau est établi, le positionnement des places peut s'effectuer.

Soit i la place qui vient juste d'être positionnée, (au départ i = 1). Le positionnement de la place k suivante se fait en plaçant sur la branche de l'hexagone centré en i, référencée par l'indice de la colonne dans laquelle se trouve la place k. Le choix de la place k se fait en prenant la place non encore positionnée d'indice le plus grand de la ligne i.

La représentation graphique du positionnement des places d'un circuit, leur nombre variant de 1 à 10, pour la maille hexagonale, est présentée sur la figure III.43.

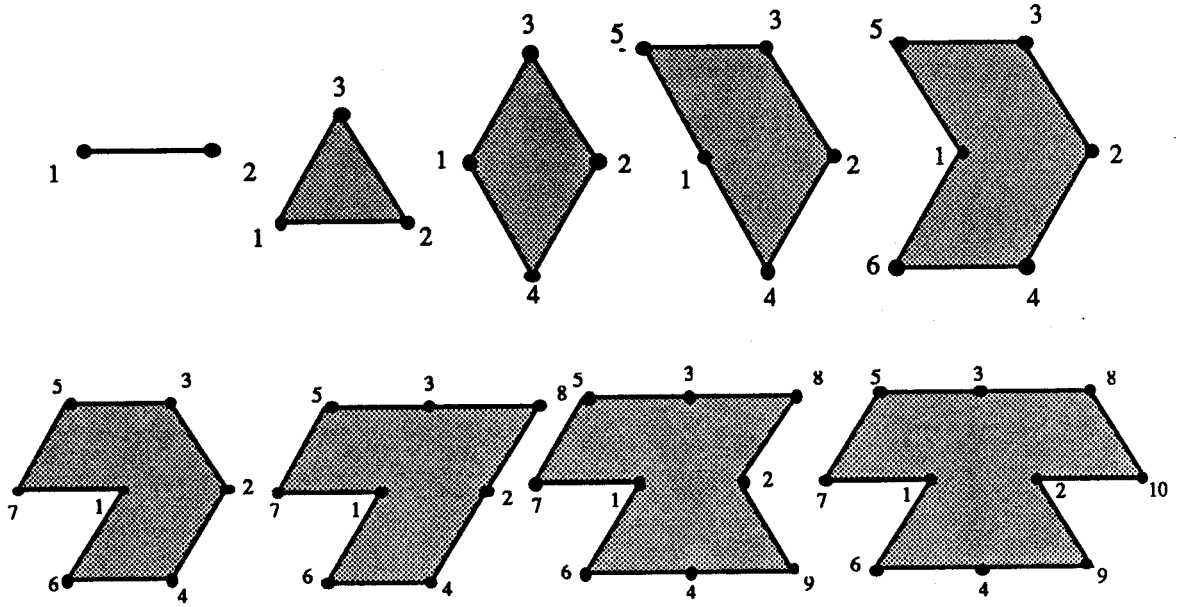


figure III.43

La représentation pour un circuit composé de 15, 20, et 25 places figure sur le schéma suivant :

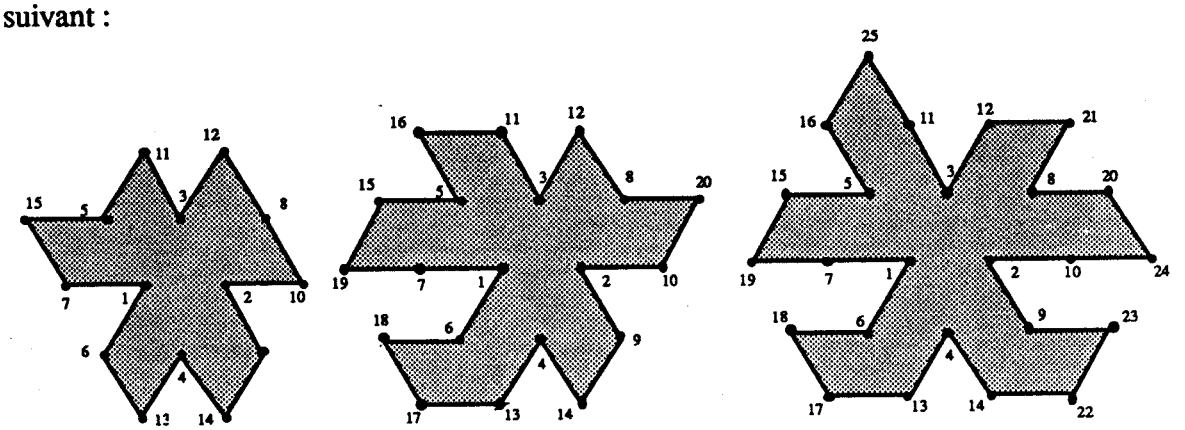


figure III.44

Une remarque que nous pouvons faire avant d'examiner les avantages et les inconvénients de cette méthode, est qu'il existe d'autres variantes pour le positionnement des places d'un circuit dans la maille hexagonale. En effet, il existe plusieurs manières pour construire le tableau, et ensuite, plusieurs façons de le balayer. Ce que nous venons de présenter, n'est qu'une illustration d'une de ces possibilités.

Le but de notre démarche est de chercher à optimiser le positionnement des places d'un graphe. Pour cela, il faut essayer d'appliquer "au plus près" les règles de la figure III.31.

Nous avons pensé alors, que l'utilisation de la maille hexagonale, un sommet ayant six voisins au lieu de quatre pour la structure rectangulaire, permettrait de mieux appliquer la règle 4.

Le seul avantage, à notre avis, de l'utilisation de la maille hexagonale, est de posséder un plus grand nombre de sommets formant des angles plus importants que pour la structure rectangulaire.

Par contre, nous retrouvons, pour les nombreuses variantes de l'automatisation du positionnement dans la maille hexagonale examinées, avec des circuits composés de plus de six places, une concavité dans leur représentation. Cette concavité engendre deux inconvénients. D'une part, tous les sommets ne pourront pas avoir un sommet équivalent sur le contour enveloppe pour positionner les autres places du graphe. D'autre part, cette concavité amène une possibilité plus grande pour les liaisons de se couper. Pour éviter la concavité dans les représentations, nous avons pensé introduire une règle supplémentaire, avant le positionnement des autres places, effectué de manière similaire à celui de la partie III.6 de ce chapitre. Cette règle consiste à effectuer un miroir des contour dès la rencontre d'une concavité, en s'arrêtant dès qu'il n'y en a plus. En fait cela revient à positionner les places du circuit sur le diamètre d'un cercle.

En conclusion, l'utilisation de la maille hexagonale pour le positionnement des places du premier circuit, s'avère être plus complexe et n'apporte pas vraiment une solution meilleure aux représentations effectuées avec la structure rectangulaire. Toutefois, d'après le résultat obtenu en appliquant cette méthode, nous pourrions choisir de positionner directement les places sur le diamètre d'un cercle. Mais, pour garder un espace raisonnable entre les places, pour un nombre important de places, le diamètre du cercle devient trop grand, ce qui amène, entre autre, une perte de place au centre de la figure.

III.6. Conclusion.

La technique de représentation adoptée pour traduire de façon automatique l'image sur écran d'un réseau de Petri associé à la modélisation d'un process de fabrication de type Systèmes à Evénements Discrets, conduit en général à un tracé à la fois clair et satisfaisant. De nombreux exemples de taille plus significative, ont permis de confirmer cette affirmation, (deux exemples de représentation de graphe de taille plus importante, sont présentés en Annexe III.6 et III.7).

L'approche que nous avons présentée, a l'avantage d'être à la fois rapide et performante, en regard de techniques basées sur d'éventuelles méthodes d'optimisation et conduit à partir de règles simples, à un résultat quasiment idéal puisque proche d'une visualisation réaliste toujours possible manuellement.

De plus, la possibilité d'une représentation automatique en conception de systèmes de contrôle - commande de processus manufacturiers, dégage le concepteur de la charge importante que constitue la gestion d'un graphe de type réseau de Petri.

IV. Développements de l'interface graphique.

La notion d'interface graphique est maintenant définie. Le but de cette partie concerne l'illustration du rôle de cette interface dans la chaîne de conception du projet CASPAIM, ainsi que les développements en cours.

IV.1. Les utilisations de l'interface graphique.

L'investissement financier mis en jeu pour l'élaboration d'un atelier flexible de production est important, il est donc souhaitable de minimiser les risques d'erreurs. C'est pourquoi, une phase de simulation est indispensable pour valider la conception avant mise en œuvre. La simulation doit permettre d'obtenir des résultats tant qualitatif que quantitatif, pour justifier et confirmer la réalité du projet conçu.

L'évaluation qualitative du système consiste à pouvoir détecter, analyser et résoudre les blocages et indéterminismes induits par une mauvaise conception ou par une spécification insuffisante du graphe de commande.

L'évaluation quantitative consiste, quant à elle, à permettre l'intégration de l'aspect temporel aux divers modèles utilisés afin d'analyser les performances du système pour ensuite les optimiser en dimensionnant éventuellement certains paramètres du système ou en modifiant la stratégie adoptée.

Le logiciel de simulation réalisé par E. Castelain, [CAS 87] puis étendu par T. Heinzerling, [HEI 88], pour la première version de CASPAIM, puis repris par C. Ausfelder, [AUS 91], pour CASPAIM2, concerne plus particulièrement l'évaluation qualitative et quantitative de la dynamique du système.

Ce logiciel permet l'édition, la simulation, la correction, et l'archivage du modèle de production flexible. Afin de permettre une définition progressive du système à concevoir, ces différentes parties ne doivent pas être indépendantes. L'utilisateur doit pouvoir simuler un modèle partiel et corriger son modèle après ou en cours de simulation. Le logiciel doit donc être essentiellement interactif.

Cette interactivité "logicielle", déjà présente dans le simulateur, a été accrue au niveau de l'utilisateur par une interface graphique appropriée. L'utilisation du simulateur s'est trouvé simplifiée par l'usage du multifenêtrage, par l'emploi des objets de sélection, tels que les

boutons poussoirs, les barres de défilement et les menus, ce qui rend son exploitation interactive et conviviale. D'autre part, l'apport de la visualisation graphique de cette interface, permet de suivre l'évolution de la simulation sur le modèle réseau de Petri du graphe de commande, par le déplacement des marques dans ce réseau, et ainsi de se rendre compte facilement des différents états par lesquels passe le système. La visualisation de la simulation, s'effectue directement sur le graphe, et permet d'avoir une vision synthétique de la commande. De plus, la convivialité de l'interface graphique permet d'agir plus rapidement sur les blocages et indéterminismes détectés par le simulateur, la visualisation du graphe de commande permet en effet une interprétation et une compréhension plus globale.

L'interface graphique répond donc à notre souci de rendre convivial l'exploitation de la simulation. Dans ce sens, nous allons ci-après énumérer les principaux apports de cette interface :

- elle permet de visualiser les différents chemins qu'empruntent les pièces, et procure ainsi l'avantage de vérifier le routage des pièces dans l'atelier,

- elle permet de vérifier si les stratégies choisies par le niveau hiérarchique et appliquées par le module interface, correspondent bien au cahier des charges,

- la méthode de positionnement automatique des places du graphe, utilisée par l'interface graphique en phase de saisie, offre la possibilité d'éliminer la phase manuelle de construction des graphes,

- les fonctions utilisées par la phase d'analyse afin d'extraire les informations nécessaires à l'application de la méthode de positionnement automatique, notamment la recherche des circuits, pourront être réutilisées pour définir le routage des pièces dans le module correspondant de l'interface,

- cette méthode peut être appliquée pour aider le concepteur au positionnement des différentes machines et systèmes de transport qui composent l'atelier de production, et offrir plusieurs versions du plan d'implantation de l'atelier, afin d'essayer d'optimiser les différents transferts, et de permettre une bonne gestion de l'environnement de travail,

- autre avantage : tous les graphes peuvent être sortis sur support papier, permettant une documentation complète et utile pour l'implantation sur site.

IV.2. Le tracé des gammes logiques et opératoires.

Dans le premier chapitre, il a été proposé de scinder la phase de spécification en deux étapes distinctes :

- d'une part la spécification fonctionnelle, qui concerne la traduction des séquences d'opérations sur chaque objet indépendamment des lieux d'opération, avec validation afin d'aboutir à l'analyse des besoins que doit satisfaire le système suivant des objectifs de production,
- et d'autre part, la spécification opérationnelle qui décrit les caractéristiques et les contraintes des différents moyens de production [CRU 90].

Il paraît évident que pour permettre au concepteur de saisir ces deux types de spécification, il est plus facile de le faire à l'aide d'un outil graphique, plus convivial qu'un outil de type éditeur de texte.

Pour ce qui est de la spécification opérationnelle des moyens de production, un éditeur graphique est réalisé par S. Amar, permettant de visualiser la composition de l'atelier donnée sous forme arborescente. Cet outil permet aussi de réaliser les liaisons moyens-fonctions pour permettre la spécification structuro-fonctionnelle. Ultérieurement, il est prévu, pour la construction de l'arborescence, de concevoir une base de données comportant toutes les caractéristiques des différentes machines et moyens de transport.

A partir de cette spécification, il est prévu d'étendre le logiciel que nous avons réalisé, en appliquant la méthode de positionnement automatique, pour permettre d'établir automatiquement la représentation graphique de l'atelier, au fur et à mesure de la spécification des relations d'accessibilité entre moyens de production, avec validation à chaque étape. De plus, en modifiant légèrement l'interface graphique, nous pouvons obtenir la représentation graphique des automates d'états finis, modèles générés à partir de la spécification complète du procédé.

De la même manière, pour la spécification fonctionnelle, l'utilisation d'une interface graphique doit assurer une meilleure convivialité. Rappelons que la phase de spécification est le point d'entrée de CASPAIM2, notamment pour élaborer les différents modèles réseaux de Petri et automates à états finis de la phase de conception de la commande. Le fait de définir dès le départ la spécification opérationnelle de manière graphique, permet de garder une meilleure cohérence d'ensemble du projet CASPAIM2. La description des différentes opérations à effectuer sur les objets, se fait alors pour chaque gamme de manière séquentielle et interactive.

L'interface graphique présentée dans ce chapitre, permet de représenter de manière automatique les gammes opératoires ainsi que les gammes logiques, mais pour cela un fichier décrivant le modèle sous forme littérale doit être fourni en entrée. La génération automatique des gammes opératoires, avec support graphique, est en cours d'étude.

La méthode employée pour afficher les graphes appartenant à la classe "CL", dont font partis les gammes opératoires et les gammes logiques, se déroule ainsi. Rappelons que nous considérons que la représentation graphique de ces graphes est un cas particulier de la méthode présentée dans la partie III de ce chapitre, et que l'algorithme se trouve en Annexe III.5.

La méthode consiste à appliquer les étapes suivantes pour chaque gamme. Comme pour la représentation des prégraphes, nous utilisons une structure de données interne de type tableau, et c'est une fois que ce tableau est "rempli" que la représentation effective du graphe a lieu.

1ère étape : Phase d'analyse.

Cette étape permet d'établir :

- les listes **l-des** des places successeurs des transitions de désassemblage,
- les listes **l-ass** des places prédécesseurs des transitions d'assemblage,
- la liste **l-ent** des places n'ayant aucun prédécesseurs,
- la fonction de Gründy associée au graphe, qui permet de l'ordonnancer par niveaux.

Définition de la fonction de Gründy, [ALJ 86] :

Soient un graphe simple, fini, $G = (E,\Gamma)$, avec E l'ensemble des sommets du graphes, Γ l'application multivoque de E dans lui-même et une application g de E dans IN définie partout. C'est une fonction de Gründy si, pour tout $x \in E$, $g(x)$ est le plus petit entier naturel qui n'appartient pas à l'ensemble des valeurs retenues pour ses successeurs immédiats.

Les graphes que nous considérons sont sans circuit, et peuvent donc être mis en niveaux. Pour trouver la fonction de Gründy d'un graphe, nous considérons que les "sorties" du graphe sont sur le niveau 0. Puis, le calcul se poursuit récursivement, en supprimant les sorties du niveau i et les arcs qui y sont incident intérieurement. Les sorties du sous-graphe obtenues formeront alors le niveau i+1.

La recherche de la fonction de Gründy d'un graphe sans circuit, fournit le résultat

représenté à titre d'exemple sur la figure III.45.

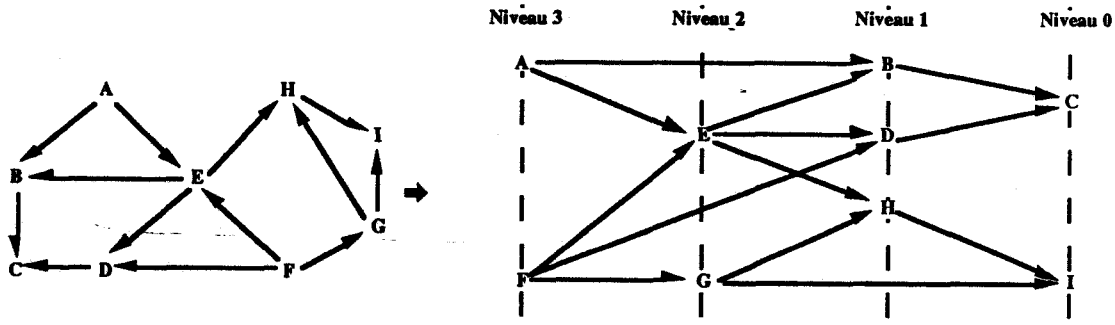


figure III.45

2ème étape : Positionnement des places "entrées".

Le positionnement de ces places s'effectue dans la première colonne du tableau. Pour connaître l'indice ligne l_i des places entrées, deux cas se présentent.

Soit il n'existe qu'une seule place, alors dans ce cas l'indice ligne l_1 vaut 1.

Soit il y en a plusieurs, ce qui veut dire qu'il y aura un assemblage ultérieur. Pour connaître l'espacement entre ces places, nous effectuons d'une part, la recherche du niveau n_i le plus petit, des places qui participent à l'assemblage, et d'autre part, la recherche du nombre maximum s_i des successeurs pour les places des niveaux compris entre n_i et N, N étant le niveau le plus grand, avant assemblage. L'indice ligne l_i de la place p_i, est alors égal à la somme de l'indice ligne l_{i-1} et de s_{i-1} de la place entrée p_{i-1}.

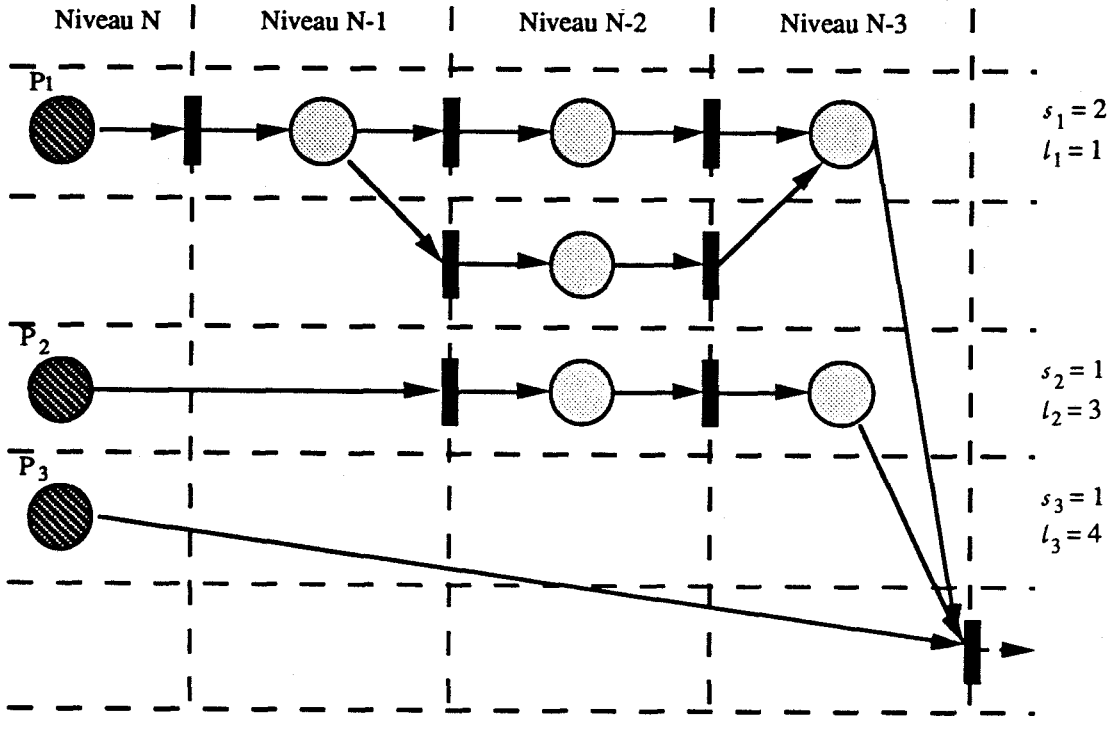


figure III.46

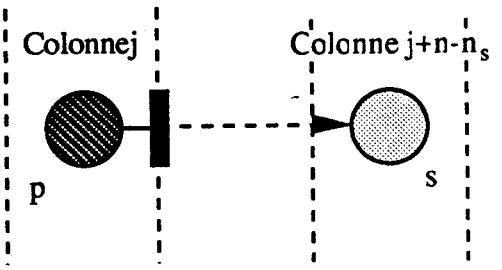
3ème étape : Positionnement des autres places.

Le positionnement des places, autres que les places "entrées", s'effectue en donnant la priorité au balayage vertical du tableau, et ainsi son remplissage se fait colonne par colonne.

Soient l'indice colonne $j \in [2,N]$, i l'indice ligne, et n le niveau de la place p dont nous cherchons à positionner les successeurs.

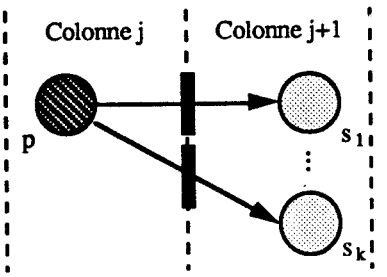
Le calcul des indices ligne ℓ_s et des indices colonne c_s, est réalisé en différenciant les cas suivants :

1. p n'a qu'un successeur s et p n'a pas de transition en amont de type assemblage ou désassemblage,
$\ell_s = i$, $c_s = j + n - n_s$,
avec n_s niveau de s



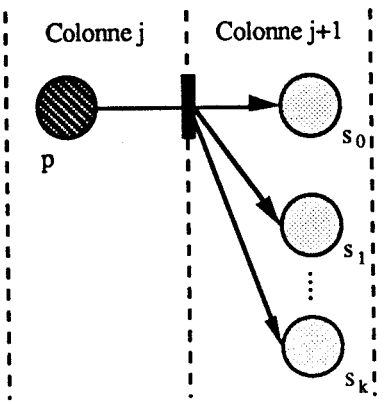
Pour tous les autres cas, l'indice colonne des successeurs est égal à j + 1.

2. p a plusieurs successeurs s_k et n'a pas de transition en amont de type assemblage ou désassemblage,
$\ell_{s1} = i$, $\ell_{sk+1} = \ell_{sk} + 1$



3. p a une de transition en amont de type désassemblage,

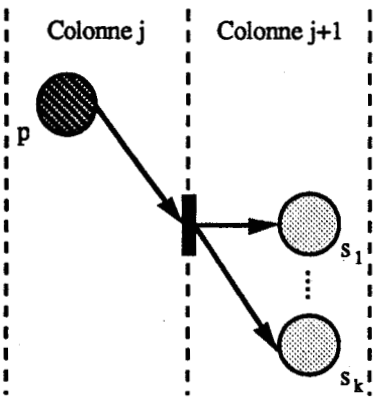
3.1 dans le cas où cette transition correspond à un désassemblage d'un lot qui sera reconstitué, une des places successeurs appartient à une l-ass, soit s_0 cette place,
$\ell_{s0} = i$,



3.2 soient sk les autres successeurs de cette transition,

$\ell_{s1} = i + 1$,

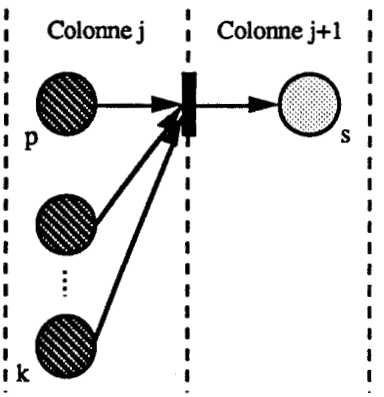
les intervalles entre les autres places successeurs est calculé de façon analogue à ceux de l'étape 2



4. p a une de transition en amont de type assemblage,

4.1 dans le cas où cette transition correspond au réassemblage d'un lot, la place p appartient à une l-des, soit s la place successeur de p,

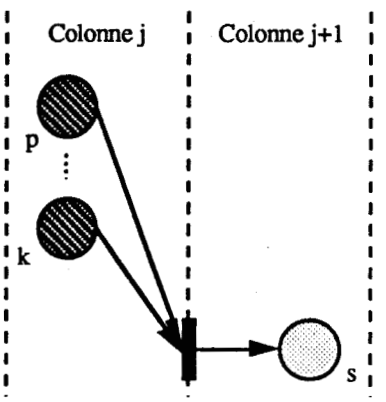
$\ell_s = i$,



4.2 sinon,

$\ell_s = i_k + 1$,

avec i_k l'indice le plus grand des places aval de la transition,



5. la place successeur a déjà été positionnée.

Il est préférable d'utiliser cette méthode de représentation des graphes de la classe "CL", leur description étant fournie sous une forme littérale, pour la représentation des gammes opératoires. En effet, ces graphes ne sont pas de taille trop importante, et ainsi, l'application de cette méthode n'est pas pénalisante. Par contre, pour les gammes logiques, il est préférable de les représenter au moment de leur génération. L'étude de génération des gammes logiques, à partir des gammes opératoires et de la spécification du procédé, combinée à une interface graphique, est en cours.

IV.3. Conclusion.

A l'issue de cette présentation, il est clair que les interfaces graphiques deviennent indispensables dans la chaîne de conception des cellules de production, et en particulier, des cellules flexibles.

Le rôle de l'interface graphique est encore destiné à s'accroître avec l'évolution du projet CASPAIM.

Conclusion.

L'interface graphique réalisée permet donc de visualiser les différents modèles réseaux de Petri utilisés lors de la conception et de la simulation de la commande du projet CASPAIM.

Cet outil nous semble important pour :

- visualiser les différents graphes lors de l'élaboration de la commande,
- disposer d'un synoptique, en phase de simulation,
- permettre la visualisation du suivi temps réel en phase d'exploitation, ce qui permet notamment de réagir plus rapidement face aux éventuelles défaillances du système,
- fournir un dossier de conception complet,
- permettre l'agencement des différents moyens de production d'un nouvel atelier, (méthode de positionnement automatique).

Conclusion générale

Conclusion générale.

Notre contribution à la démarche de Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière, a essentiellement consisté à proposer une approche de l'analyse et de la résolution des indéterminismes et conflits induits par la flexibilité des processus étudiés.

Un caractère original de la nouvelle méthodologie consiste notamment à intégrer de manière réactive l'ordonnancement en temps réel permettant la résolution a priori de tels conflits. Il est ainsi permis au gré du concepteur de les résoudre soit globalement par interfaçage avec le niveau ordonnancement, soit localement au niveau du pilotage.

D'un autre point de vue, concernant une meilleure interprétation des comportements (en phase d'exploitation) du process et de sa commande, ou éventuellement d'un modèle (en phase de validation par simulation), il nous a semblé essentiel de fournir un outil graphique adapté aux modèles les plus couramment utilisés, les réseaux de Petri, sous leurs différentes formes. Dans ce sens, nous avons abouti au développement d'une approche d'aide à la disposition des places des réseaux de Petri, basée sur la recherche des différents chemins et circuits apparaissant sur ces graphes.

Il existe à l'évidence une relation qui resterait à mieux préciser entre l'organisation réelle du système de production et de son transport, et la structure du graphe produit par construction selon les algorithmes proposés. Il conviendrait dans l'avenir, de voir dans quelle mesure l'optimisation des routages conduirait à discerner le et les meilleurs choix de structure.

L'approche objet associée au modèle réseau de Petri est à l'évidence nécessaire dans le contexte d'une forte flexibilité. La gestion de marques pouvant caractériser aussi bien des

instances que des classes d'objet est un problème délicat, notamment du point de vue de l'interprétation graphique des résultats. La stabilisation de ce type de modèle est actuellement en cours, il conviendra à court terme d'assurer la prise en compte de tels objets "dynamiques" par une extension du modèle graphique proposé.

Bibliographie

Introduction générale.

[BOU 84] A. BOURJAULT,
"Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires", Thèse d'état de l'université de Franche-Conté, Besançon, 1984.

[BOU 87] A. BOURJAULT, D. CHAPPE, J.M. HENRIOUD,
"Elaboration automatique des gammes d'assemblage à l'aide de réseaux de Petri", APII, vol 21, n° 4, 1987.

[BOU 88] J.P. BOUREY,
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 23 mars 1988.

[CAM 89] J.P. CAMPAGNE, G. CAPLAT,
"Elaboration automatique de gammes d'assemblage", APII, vol 23, n° 1, 1989.

[CAS 87] E. CASTELAIN,
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 26 février 1987.

[CRA 89] E. CRAYE,
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, janvier 1989.

[KAP 88] M. KAPUSTA,
"Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour des cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, décembre 1988.

Chapitre I.

[ATA 87] H. ATABAKHCHE,
"Utilisation conjointe de l'intelligence artificielle et des réseaux de Petri : application au contrôle d'exécution d'un plan de fabrication", Thèse de Docteur de l'université Paul Sabatier, Toulouse, 18 décembre 1987.

[BEN 85] G. BENCHIMOL,
"La conception des usines de demain", Hermès, 1985.

[BON 85] R. BONETTO,
"Les ateliers flexibles de production", Hermès Publishing, 1985.

[BOO 88] G. BOOCH,
"Ingénierie du logiciel avec ADA", InterEditions, IIA, 1988.

[BOU 84] A. BOURJAULT,
"Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires", Thèse d'état de l'université de Franche-Conté, Besançon, 1984.

[BOU 87a] J.P. BOUREY, D. CORBEEL, E. CRAYE, J.C. GENTINA,
"Utilisation des réseaux de Petri structurés adaptatifs colorés dans l'analyse et la synthèse du contrôle hiérarchisé de processus discontinus", APII, vol 21, n° 4, 1987, Partie A : les modèles de description.

[BOU 87b] A. BOURJAULT, D. CHAPPE, J.M. HENRIOUD,
"Elaboration automatique des gammes d'assemblage à l'aide de réseaux de Petri", APII, vol 21, n° 4, 1987.

[BOU 88a] J.P. BOUREY,
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 23 mars 1988.

[BOU 88b] J.P BOUREY, J.C. GENTINA,
"Structuration de la partie procédurale du système de commande de cellules flexibles de production", Congrès AFCET Automatique, Octobre 88, Grenoble, p 233.

[BOU 90a] J.P. BOURRIERES, F. LHOTE,
"Modélisation hiérarchisée des tâches d'assemblage", CIM'90, Bordeaux, 12-14 juin 90.

[BOU 90b] J.P. BOURRIERES,
"Contribution à la modélisation intégrée des systèmes robotisés d'assemblage", Thèse d'état de l'université de Franche-Conté, Besançon, 1990.

[BRA 83] G.W. BRAMS,
"Réseaux de Petri : Théorie et pratique", Masson, PARIS 1983, Tome 1 : Théoric et analyse, Tome 2 : Modélisation et applications.

[CAM 89] J.P. CAMPAGNE, G. CAPLAT,
"Elaboration automatique de gammes d'assemblage", APII, vol 23, n° 1, 1989.

[CAR 91] J. CARDOSO,
"Sur les réseaux de petri à marquage flou", Thèse de Docteur de l'université, Toulouse, 1991.

[CAS 87] E. CASTELAIN,
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 26 février 1987.

[CRA 89] E. CRAYE,
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, janvier 1989.

[CRU 91] D. CRUETTE,
"Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 8 février 1991.

[GEN 87] J.C. GENTINA, D. CORBEEL,
"Coloured adaptative structured Petri-net : a tool for automatic synthesis of hierarchical control of flexible manufacturing systems", IEEE, International Conference on Robotics and Automation, March,31/April,3, 1987, Raleigh, North Carolina.

[KAP 87] M. KAPUSTA, J.C. GENTINA,
"Introduction to a first step of aided design of the control system of flexible manufacturing cells", IEEE Montech'87-compint'87, pp 258-262, Canada, Novembre 1987.

[KAP 88] M. KAPUSTA,
"Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour des cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, décembre 1988.

[LEM 90] J.L. LE MOIGNE,
"La modélisation des systèmes complexes", Afcet Systèmes, Dunod, 1990.

[PRU 87] F. PRUNET, J.L. STURLESE, E. CAZALOT, E. GINESTET, D. PANAGET, G. DECHENAUX, P. LLORCA,
"Méthodologie et implantation automatique de commande d'automatisme à l'aide de la chaîne PIASTRE", revue RAIRO APII, vol. 21, n° 4, p 299-321, 1987.

[ROS 75] D.T. ROSS, J.B. GOODENOUGH, C.A. IRVINE,
"Software Engineering : Process, Principles and Goals", Computer, mai 1975.

[SAH 87] A. SAHRAOUI,
"Contribution à la surveillance et à la commande d'atelier", Thèse de Doctorat de l'université Paul Sabatier, Toulouse, 21 octobre 1987.

[SIB 85] C. SIBERTIN-BLANC,
"High level Petri nets with data structure", 6th European Workshop on Applications and Theory of Petri Nets, Helsinki, Finland, 1985.

[SIB 88] C. SIBERTIN-BLANC,
"Le prototypage des applications interactives à l'aide de réseaux de Petri", Séminaire d'informatique de l'IIE, Réseaux de Petri, 26-27 octobre 1988.

[TAR 86] H. TARDIEU, A. ROCHFELD, R. COLLETTI,
"La méthode Merise : Tome 1, principes et outils", Les éditions d'organisation, 1986.

[VAL 88] R. VALETTE, J. CARDOSO, H. ATABAKHCHE, M. COURVOISIER, T. LEMAIRE,
"Petri nets and production rules for decision levels in FMS control", IMACS 88, 12ème Congrés Mondial IMACS, P 522-524, vol 3, Paris, 18-22 juillet 1988.

Chapitre II.

BU LILLE

[BEL 88] G. BEL, E. BENSANA, D. DUBOIS,
"Construction d'ordonnancements prévisionnels : un compromis entre approches classiques et systèmes experts", APII, vol 22, pp 509-536, 1988.

[BIT 77] G.R. BITRAN, A.C. HAX,
"On the design of hierarchical production planning systems", Decision Sciences, vol 8, 1977, pp 29-55.

[BOI 91] S. BOIS,
"Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production",Thèse de Docteur de l'université, LILLE, 28 novembre 1991.

[CHA 83] G. CHASSANG, H. TRON,
"Gérer la production avec l'ordinateur", Dunod, 1983.

[CHA 91] C. CHARBONNIER, J.L. FARGES, J.J. HENRY,
"Le routage dynamique des véhicules", ECC 91, Grenoble, 2-5 juillet 91, pp 2391-2396.

[DOU 83] G. DOUMEINGTS, D. BREUIL, L. PUN,,
"La gestion de production assistée par ordinateur", Edition paris, Hermes, 1983.

[FRE 88] Y. FREIN, Y. DALLERY, J.J. PIERRAT, R. DAVID,
"Optimisation du routage des pièces dans un atelier flexible par des méthodes analytiques", APII, vol. 22, pp 489-508, 1988.

[GIA 88] V. GIARD,
"Gestion de production", Collection Gestion, Economica, 1988.

[HAM 91] S. HAMMADI,
"Une méthode d'ordonnancement minimisant les temps d'attente et de transit dans les systèmes de production flexibles de type job-shop", Thèse de Docteur de l'université, LILLE, 18 décembre 1991.

[HEB 91] A. HEBRARD, J.P. BOUREY, J.C. GENTINA,
"Concept d'interface intelligent et structuration de la commande de systèmes de production flexibles de type manufacturier", ECC'91, Grenoble, 2-5 juillet 91, pp 2123-2127.

[KUS 83] A. KUSIAK,
"Loading models in flexible manufacturing systems", Proceedings of the 7th international conference on production research, 1983, pp 119-132.

[LAM 87] P. LAMY,
"Ordonnancement et gestion de production", manuel pratique, Hermes, 1987.

[MAC 83] C. MACCHI, J.F. GUILBERT,
"Télématique : transport et traitement de l'information dans les réseaux et systèmes téléinformatiques", Dunod informatique, 1983.

[PUJ 84] G. PUJOLLE,
"La télématique : réseaux et applications", Eyrolles, 1984.

[RAY 85] M. RAYNAL,
"Algorithmes distribués et protocoles", Eyrolles, 1985.

[SHI 89] O.K. SHIN,
"Contribution au pilotage d'ilots flexibles d'assemblage", Diplome de Docteur de l'université de Franche Comté, 9 novembre 1989.

[TOG 90] A. TOGUYENI, E. CRAYE, J.C. GENTINA,
"A method of temporal analysis to perform on-line diagnosis in the context of flexible manufacturing systems", IECON'90, vol. 1, pp 445-450, Pacific Grove (Californie), novembre 1990.

[TOG 91] A. TOGUYENI, E. CRAYE, E. CASTELAIN,
"From the treatment of failures to the management of working modes",
IMACS / IFAC Symposium on "Modelling and Control of Technological Systems", Lille, vol. 1, pp 595-601, mai 1991.

Chapitre III.

[ALJ 86] A. ALJ, R. FAURE,
"Guide de la recherche opérationnelle", Masson, 1986, Tome 1 : les fondements.

[AUS 91] C. AUSFELDER,
"Conception et réalisation d'un simulateur d'ateliers flexibles modélisés par réseaux de Petri objets", Diplomarbeit, Technische Universität München / Institut Industriel du Nord Lille, 6 juin 1991.

[BAT 67] A. BATTERSBY,
"Méthodes modernes d'ordonnancement", Dunod, 1967, pp 17-18

[BLA 79] M. BLANCHARD,
"Comprendre, maîtriser et appliquer le GRAFCET", cepadues éditions, 1979.

[BOL 83] R. BOLT,
"Les images interactives", La Recherche n° 144, mai 1983, pp 678-686.

[CAS 87] E. CASTELAIN,
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 26 Février 1987.

[COU 83] F. COUPIGNY,
"La synthèse des images", La Recherche n° 144, mai 1983, pp 735-745.

[CRU 90] D. CRUETTE, J.P. BOUREY, J.C. GENTINA,
"Method of structural and fonctionel analysis of complex processes, and aided design of command under constraints in the Manufacturing", MIM-S '90, Brussels, 3-7 septembre 1990, pp IV-C-1-1/6.

[DRO 87] F. DROESBEKE, M. HALLIN, Cl. LEFEVRE,
"Les graphes par l'exemple", Ellipses, 1987.

[FER 90] J. FERBER,
"Conception et programmation par objets", série Technologies de pointe informatique, Hermès, 1990.

[GAR 90] Y. GARDAN, M. LUCAS,
"Langages de programmation graphique", Encyclopédie permanente, Techniques de l'ingénieur, Tome H2, article n° 420.

[GOT 86] S. GOTO,
"Design methodologies", North-Holland, Amsterdam, Netherlands, 1986.

[HEB 92] A. HEBRARD, J.P. BOUREY, J.C. GENTINA,
"A tool of petri net graphs representation in manufacturing context", INCOM'92, 25-28 mai 92, Toronto, Canada.

[HEC 90] H. HECKENROTH, P. SIARRY,
"Opimisation par la méthode du recuit simulé du dessin d'un modèle conceptuel des données", AFCET/INTERFACES, n° 98, pp 11-23, Décembre 1990.

[HEI 88] T. HEINZERLING,
"Adaptation of the Petri net of the pregraph defined in the CASPAIM project in order to make the prototypage and a preliminary evaluation of the parallel architecture of the control in the field of CIM", Diplomarbeit, Technischen Universität München / Institut Industriel du Nord Lille, 7 novembre 1988.

[KAU 68] A. KAUFMANN,
"Méthodes et modèles de la recherche opérationnelle", Dunod, Tome 2, 1968.

[LED 78] A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ,
"Conception et réalisation d'un logiciel graphique interactif indépendant du contexte d'utilisation : le logiciel de base GRIGRI", RAIRO Informatique/Computer Science, vol 12, n° 2, 1978, pp 147-173.

[LUC 82] M. LUCAS,
"Réalisation des logiciels graphiques intéractifs", Travaux dirigés de l'Ecole d'été d'informatique tenue au Bréau-sans-Nappe du 7 au 27 juillet 1979, Editions Eyrolles, 1982.

[ROS 86] ROSEAUX,
"Exercices et problèmes résolus de recherche opérationnelle", Masson, 1986, Tome 1 "Graphes : leurs usages, leurs algorithmes", pp 27.

[TRI 87] S. M. TRIMBERGER,
"Introduction to CAD for VLSI", Kluwer Academic Publishers - Boston, USA, 1987,

[ZOB 89] G. W. ZOBRIST,
"Progress in computer-aided VLSI design", Ablex Publishing - Norwood, USA, 1989, vol 2 Techniques.

Glossaire.

[CER 88] A. CERNAULT,
"La simulation des système de production", Cepadues Editions, 1988.

[GIA 88] V. GIARD,
"Gestion de production", Collection Gestion, Economica, 1988.

[GER 83] D. GERWIN,
"A framework for analysing the flexibility of manufacturing processes", School of Business and Administration, University of Wisconsin, Milwaukee USA, 1983.

[KAP 88] M. KAPUSTA,
"Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour des cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, décembre 1988.

[MEY 90] B. MEYER,
"Conception orientée objet", InterEdition, 1990.

Références bibliographiques du LAIL.

[AMA 90] S. AMAR, E. CASTELAIN, J.C. GENTINA,
"Modélisation des moyens de production par langage orienté objet en vue de la conception de la commande d'un système de production flexible", CIM'90, Colloque international "Productique et Intégrations", Bordeaux, juin 1990, pp 307-314.

[AUS 91] C. AUSFELDER,
"Conception et réalisation d'un simulateur d'ateliers flexibles modélisés par réseaux de Petri objets", Diplomarbeit, Technische Universität München / Institut Industriel du Nord Lille, 6 juin 1991.

[BOI 91] S. BOIS,
"Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production",Thèse de Docteur de l'université, LILLE, 28 novembre 1991.

[BOU 87] J.P. BOUREY, D. CORBEEL, E. CRAYE, J.C. GENTINA,
"Utilisation des réseaux de Petri structurés adaptatifs colorés dans l'analyse et la synthèse du contrôle hiérarchisé de processus discontinus", APII, vol 21, n° 4, 1987, Partie A : les modèles de description.

[BOU 88a] J.P. BOUREY,
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 23 mars 1988.

[BOU 88b] J.P BOUREY, J.C. GENTINA,
"Structuration de la partie procédurale du système de commande de cellules flexibles de production", Congrès AFCET Automatique, Octobre 88, Grenoble, p 233.

[CAS 87] E. CASTELAIN,
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 26 février 1987.

[CRA 89] E. CRAYE,
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, janvier 1989.

[CRU 90] D. CRUETTE, J.P. BOUREY, J.C. GENTINA,
"Method of structural and fonctionel analysis of complex processes, and aided design of command under constraints in the Manufacturing", MIM-S '90, Brussels, 3-7 septembre 1990, pp IV-C-1-1/6.

[CRU 91] D. CRUETTE,
"Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, 8 février 1991.

[ELK 92] S. ELKHATTABI, D. CORBEEL, J.C. GENTINA,
"Intégration des aspects de sureté de fonctionnement dans la conception des systèmes de commandes d'ateliers flexibles", INCOM'92, 25-28 mai 92, Toronto, Canada.

[GEN 87] J.C. GENTINA, D. CORBEEL,
"Coloured adaptative structured Petri-net : a tool for automatic synthesis of hierarchical control of flexible manufacturing systems", IEEE, International Conference on Robotics and Automation, March,31/April,3, 1987, Raleigh, North Carolina.

[HAM 91a] S. HAMMADI,
"Une méthode d'ordonnancement minimisant les temps d'attente et de transit dans les systèmes de production flexibles de type job-shop", Thèse de Docteur de l'université, LILLE, 18 décembre 1991.

[HAM 91b] S. HAMMADI, E. CASTELAIN, P. BORNE,
"Efficient and feasible job-shop scheduling in flexible manufacturing systems", IMACS/IFAC Symposium, Mars 91, Toulouse.

[HAM 92] S. HAMMADI, E. CASTELAIN, P. BORNE,
"A job-shop scheduling method minimizing the waiting and transit times in Flexible Manufacturing Systems", INCOM'92, 25-28 mai 92, Toronto, Canada.

[HEB 91] A. HEBRARD, J.P. BOUREY, J.C. GENTINA,
"Concept d'interface intelligent et structuration de la commande de systèmes de production flexibles de type manufacturier", ECC'91, Grenoble, 2-5 juillet 91, pp 2123-2127.

[HEB 92] A. HEBRARD, J.P. BOUREY, J.C. GENTINA,
"A tool of petri net graphs representation in manufacturing context", INCOM'92, 25-28 mai 92, Toronto, Canada.

[HEI 88] T. HEINZERLING,
"Adaptation of the Petri net of the pregraph defined in the CASPAIM project in order to make the prototypage and a preliminary evaluation of the parallel architecture of the control in the field of CIM", Diplomarbeit, Technischen Universität München / Institut Industriel du Nord Lille, 7 novembre 1988.

[HUV 92] B. HUVENOIT, E. CRAYE, J.C. GENTINA,
"Elaboration de la commande de cellules de production flexibles dans l'industrie manufacturière", Conférence et exposition sur l'automatisation industrielle, 1-3 juin 1992, Montréal, Canada.

[KAP 87] M. KAPUSTA, J.C. GENTINA,
"Introduction to a first step of aided design of the control system of flexible manufacturing cells", IEEE Montech'87-compint'87, pp 258-262, Canada, Novembre 1987.

[KAP 88] M. KAPUSTA,
"Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour des cellules de production flexibles dans l'industrie manufacturière", Thèse de Docteur de l'université, LILLE, décembre 1988.

[TOG 90] A. TOGUYENI, E. CRAYE, J.C. GENTINA,
"A method of temporal analysis to perform on-line diagnosis in the context of flexible manufacturing systems", IECON'90, vol. 1, pp 445-450, Pacific Grove (Californie), novembre 1990.

[TOG 91] A. TOGUYENI, E. CRAYE, E. CASTELAIN,
"From the treatment of failures to the management of working modes", IMACS / IFAC Symposium on "Modelling and Control of Technological Systems", Lille, vol. 1, pp 595-601, mai 1991.

Glossaire

Glossaire.

Atelier flexible :

où une machine donnée doit usiner un nombre important de pièces différentes. Nom donné pour les distinguer des ateliers de fabrication en grande série, qui produisent un flux continu de matériels homogènes.

Attribut (objet) :

cf. définition de *champs*.

Automate programmable (API) :

Un automate programmable est un appareil électrique qui comporte une mémoire, programmable par un utilisateur à l'aide d'un langage adapté au stockage interne des instructions composant les fonctions d'automatisme :

- logique séquentielle et combinatoire,
- temporisation,
- comptage, décomptage, comparaison,
- calcul arithmétique,
- réglage, asservissement, régulation, ...

pour commander, mesurer et contrôler au moyen de modules d'entrées et de sorties (logiques, numériques ou analogiques) différents types de machines ou des processus, en environnement industriel. (NF C63-850)

Cellule :

Une cellule est une unité de production :

- capable de fonctionner de manière autonome et satisfisante pendant une durée limité,
- non sécable en unités de production plus petites.

L'autonomie de fonctionnement à court terme qui caractérise la notion de cellule, est assurée par les stocks tampons amont et aval.

Une cellule est composée d'un ou plusieurs *postes* reliés par un système de transport inter-postes.

Champ (objet) :

Les champs ou *attributs* décrivent la structure des *instances* d'une *classe* d'*objets*.

Classe (objet) :

Une classe est un modèle général, qui permet de regrouper des *objets* qui possèdent une structure et un comportement en commun. La définition d'une classe est constituée par un un ensemble de *champs* qui décrivent la structure de ces *instances*, et de *méthodes* qui leur sont applicables.

CNC :

Système de commande d'une machine à commande numérique, (Commande Numérique par Calculateur).

Flexible :

La définition opposée : un système n'est pas flexible si par exemple, une machine est dédiée à la fabrication d'un produit spécifique, ou si il est impossible de reconfigurer ou d'adapter le système en cas de changement de production.

La flexibilité du système globale repose sur la flexibilité de ses sous-systèmes physiques et informationnels. Plusieurs types de flexibilité peuvent être distingués [GER 83] :

Flexibilité des opérateurs (Machine flexibility) : c'est la capacité des opérateurs d'effectuer plusieurs tâches élémentaires (éventuellement en changeant les outils) et de permettre la reprogrammation des manipulations.

Flexibilité de routage (Routing flexibility) : un système de production possède cette flexibilité, lorsqu'une pièce (ou un lot de pièces) lors de sa fabrication, peut être transférée à sa destination par plus d'un trajet.

Flexibilité dans le séquencement des tâches (Operation flexibility) : c'est la capacité de changer l'ordre des tâches à réaliser sur un produit en cours de production.

Flexibilité de produit (Product flexibility) : c'est la possibilité de modification du système pour produire un nouveau produit (ou une nouvelle famille de produits) d'une façon économique et rapide.

Flexibilité d'expansion (Expansion flexibility) : c'est la capacité de construire, d'étendre ou de réduire le système de production d'une façon rapide et peu coûteuse. Cette flexibilité dépend surtout de la modularité du système.

Flexibilité de mélange (Mix flexibility) : c'est la possibilité de produire simultanément sur une même machine, des produits de même nature (ayant des caractères de base communs) mais de type différent.

Flexibilité de volume (Volume flexibility) : c'est la capacité du système à réagir face à des variations de la demande correspondant à des fluctuations importantes sur les quantités à produire.

Flexibilité de production (Production flexibility) : c'est l'expression de l'étendue de l'univers des produits que le système est capable de produire.

Gamme logique (CASPAIM2) :

C'est un modèle purement fonctionnel du processus de fabrication des pièces à traiter.
La gamme logique d'un produit traduit la séquence des fonctions qui sont appliquées à ce produit. La gamme logique fait abstraction de l'aspect opérationnel du système de production effectif, c'est-à-dire ne fait pas référence à l'équipement.
Nous traduisons les gammes logiques par un réseau de Petri dans lequel les jetons transportent des informations en nombre a priori quelconque. Les *champs* de ces jetons servent à restreindre les possibilités de tir des *transitions*. Leur déclenchement est simplement conditionné par des prédicat portant sur les valeurs des champs.

Gamme opératoire (CASPAIM1) :

Description sous forme d'un ensemble de règles de production, des *opérations* que les objets (produits et moyens de transport : palettes) subissent.
Dans ce modèle, il est possible de voir apparaître simultanément changement de lieu et d'état.

Gamme opératoire (CASPAIM2) :

Modèle réseau de Petri objets qui correspond à un graphe dans lequel un objet change de manière exclusive de lieu ou d'état.
La démarche de conception de la *partie commande* consiste dans un premier temps à enrichir le modèle réseau de Petri de la *gamme logique* en précisant de manière exclusive et complète sur quels lieux physiques ont lieu les *opérations*. On voit apparaître sur cette gamme :

- les lieux de transit des pièces,
- la description fonctionnelle des opérations appliquées au produit.

Dans un deuxième temps, les gammes ainsi obtenues sont enrichies par l'ajout de *places* requête et acccusé, qui permettent de connecter cette gamme aux différents modèles des groupements cellules et moyens de transport.

Génie logiciel [MEY 90] :

Le Génie logiciel est l'application de méthodes scientifiques au développement de théories, modèles, techniques, langages, et outils favorisant la production (développement, maintenance, et exploitation) de logiciels de qualité.

Gestion de production [GIA 88] :

La gestion de production a pour objet la recherche d'une organisation efficace de la production de biens et de services, (la production est une transformation de ressources appartenant à un système productif et conduisant à la création de biens ou de services). Elle s'appuie sur un ensemble d'outils d'analyse et de résolution des problèmes qui visent à limiter les ressources nécessaires à l'obtention d'une production dont les caractéristiques technico-commerciales sont connues. La détermination de ces caractéristiques (définition du produit, des procédés de fabrication, de la demande à satisfaire) est très largement externe à la gestion de production qui n'intervient que pour en limiter, pour un horizon donné, le champs des possibilités.

Implantation :

Synthèse du système de commande sur un support informatique.

Après une phase de traduction des modèles du système de commande grâce à des règles de transposition, une implantation effective et distribuée peut être envisagée sur des matériels existant disposant d'outils de programmation normalisés.

Instance (objet) :

Les *objets* générés à partir d'une *classe* sont appelés les instances de la classe.

Machine à commande numérique :

On dit qu'une machine est à commande numérique lorsque tout ou partie des instructions nécessaires à son fonctionnement lui sont transmises sous forme alphanumérique.

Les instructions concernent :

- les données techniques telles que : vitesse de rotation de broche, vitesse d'avance, mise en route ou arrêt de la lubrification, ...
- les données géométriques : les coordonnées caractérisant les positions relatives de la pièce ou des outils, les trajectoires, ...
- les données séquentielles désignant : les outils à employés, l'ordre des cycles de fabrication, la mise en position initiale, ...

C'est en fait une machine où l'opérateur n'intervient plus directement sur le processus de fabrication, mais par l'intermédiaire d'un programme décrivant le travail à réaliser.

Méthode (objet) :

Une méthode est une opération (fonction) qui est applicable à une *classe* d'*objets*.

Méthode MRP [GIA 88] :

Plusieurs logiciels de nature bien différente coexistent sous l'appellation MRP. On peut en fait distinguer trois familles d'approche de la méthode et des logiciels qui s'en réclament :

- La première méthode, cherche à établir une programmation de la production sans se poser le problème des capacités de production effectivement disponibles. Ne pas se préoccuper des capacités effectivement disponibles, revient à considérer que les propositions faites sont réalisables, tout se passant comme si les capacités étaient infinies. (MRP = Material Requirement Planning)
- La seconde méthode va plus loin que la précédente en cherchant à ajuster la charge souhaitée et la charge disponible pour chaque centre de production. (MRP = Manufactoring Ressource Planning)
- La troisième, la plus ambitieuse, vise, en fait, à contrôler toutes les ressources de l'entreprise qui sont mobilisées directement ou indirectement par la production.

Méthode du recuit simulé :

Cette méthode d'optimisation s'appuie sur une idée intuitive : la recherche progressive d'une solution optimale à un problème np-complet de grande taille est semblable à l'évolution d'un système physique d'un état désordonné vers un état ordonné. Cette analogie permet de traiter un problème d'ingénierie en empruntant, à la physique statistique, la technique du recuit, qui conduit à un matériau parfaitement ordonné.

Niveau hiérarchique :

Le niveau hiérarchique est un niveau décisionnel qui permet d'effectuer le paramétrage des modèles de la *partie commande*, pour tenir compte des stratégies de production. Il est constitué :

- d'un niveau de *planification*,
- d'un niveau d'*ordonnancement*,
- et d'un système de pilotage.

Son rôle est de construire les décisions et les choix destinés à résoudre les indéterminismes de la partie commande.

Objet :

Un objet est une entité informatique qui comprend à la fois :

- une partie statique qui représente son état et les liens qui l'unissent à d'autres objets, (*champs*, ou *attributs*),
- et une partie dynamique qui décrit son comportement, c'est-à-dire l'ensemble des opérations qui lui sont applicables, ainsi que ses réactions aux événements du milieu dans lequel il se trouve, (*méthodes*).

Opération :

Parmis les opérations applicables aux produits, nous distinguons :

- les fonction transformationnelles, qui modifient l'état physique du produit :
  - les usinages : tournage, fraisage, lavage, graissage, peinture, ...
  - les assemblages, qui conduisent à la mise en liaison :
    - soit de deux composants élémentaires,
    - soit d'un composant élémentaire et d'un ensemble de composants déjà liés entre eux,
    - soit deux ensembles de composants déjà liés entre eux,
- les fonctions informationnelles, qui permettent de garantir la conformité des états physiques obtenus après des interventions extérieures, opérations de métrologies, ...

Ordonnancement [GIA 88] :

Les problèmes d'ordonnancement relèvent de la détermination de l'ordre de passage d'un certains nombre de travaux à exécuter. Les techniques d'ordonnancement relèvent de la problématique de la régulation à court terme de l'entreprise, et plus particulièrement de celle du contrôle de l'utilisation de la main d'œuvre et des équipements productifs.

Partie commande :

Son rôle est d'assurer la coordination et le séquencement des commandes applicables à la *partie opérative.*

Partie opérative :

Interface entre la partie commande est le procédé. Sont inclus dans la partie opérative, les capteurs, actionneurs et boîtiers de commande numérique.

Poste :

Un poste est une unité de production réalisant pendant une durée, appelée temps de cycle, une ou plusieurs "fonctions" sur un même exemplaire du produit.

Planification de la production [GIA 88] :

La planification de la production vise, pour un horizon de planification, à optimiser l'utilisation des facteurs productifs disponibles pour la production d'un ou de plusieurs produits répondant à des caractéristiques précises. Il s'agit d'un processus de traitement d'informations aboutissant à une programmation prévisionnelle s'appuyant sur une démarche d'optimisation.

Prégraphe [KAP 88] :

Ce modèle est une agrégation des règles opératoires sous forme de réseau de Petri coloré où chaque *place* correspond à un lieu, une marque colorée à un objet et une *transition* à une règle de production.

Il constitue l'ossature initiale du modèle procédural structuré qui conduit à l'élaboration de la *partie commande* du système.

Procédé :

Ce terme recouvre l'ensemble des objets matériels auxquels s'adressent les ordres de la partie commande, et émet en retour à celle-ci un certain nombre d'informations. Il est constitué de trois parties :

- une interface : la partie opérative, qui permet la communication avec la partie commande,
- un ensemble d'éléments physiques : formé des machines et moyens de l'unité de production,
- un ensemble d'éléments consommables : formé des outils et produits manufacturés.

Ressource :

Ce terme est utilisé pour définir les éléments de l'unité de production, tour, fraiseuse, stockeur rotatif, convoyeur, robot, ...

Simulation [CER 88]:

La simulation d'un système complexe consiste à le décrire en un grand nombre d'éléments simples, à décrire le comportement de chaque élément et ses interactions avec les autres, pendant l'horizon de simulation.

La simulation est l'utilisation d'un outil, généralement informatique, qui s'applique à un système donné, ici un système de production formé d'un système physique (procédé), d'un système d'information (partie commande), d'un système de conduite ou de gestion (niveau hiérarchique).
Il permet d'évaluer les performances du système en terme de circulation du flux de produits : historique des déplacements, évolution de l'état, taux d'engagement des moyens, longueurs des files d'attente, ...
Il utilise des techniques de modélisation qui permettent de décrire le fonctionnement du système avec le degré de détails nécessaire à la résolution du problème posé.

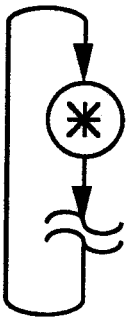
Usine intégrée :

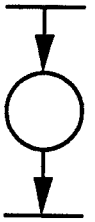
Usine où l'on retrouve intimement liés dans le système de production :

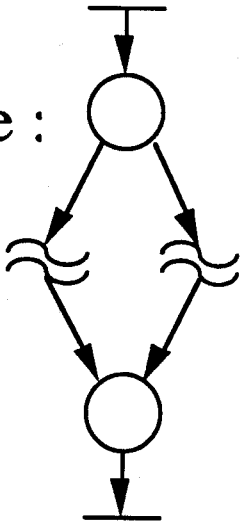
- des équipements de production et de contrôle,
- des moyens de transport,
- des matériels et logiciels de pilotage et d'automatisation.

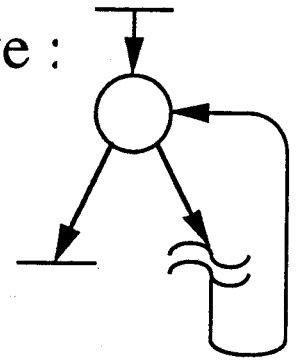
Annexe

I.1. Graphes de processus.

Processus : 

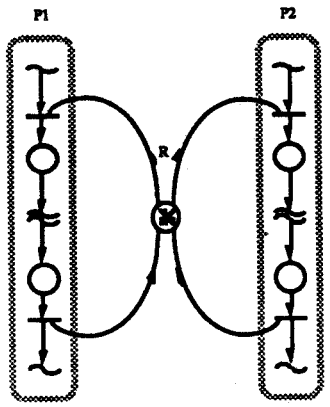
action : 

alternative : 

répétitive : 

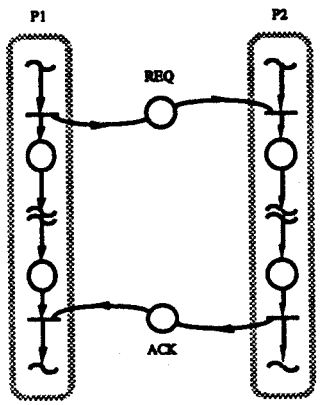
I.2. Graphes de liaisons.

L'exclusion mutuelle :



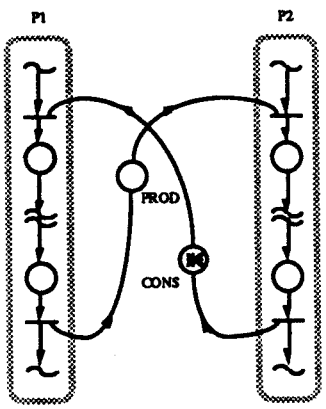
P1 et P2 sont deux processus exclusifs qui partages une même ressource critique R. C'est le cas, par exemple, de l'utilisation d'un robot par deux machines.

La synchronisation avec accusé de réception :



Le processus P2 est synchronisé par le processus P1. P2 attend une marque de P1 pour évoluer. P1 continue son évolution propre, puis attend un accusé de réception de P2 pour achever son cycle. Ce type de liaison permet de représenter des protocoles de communication de type émission - réception.

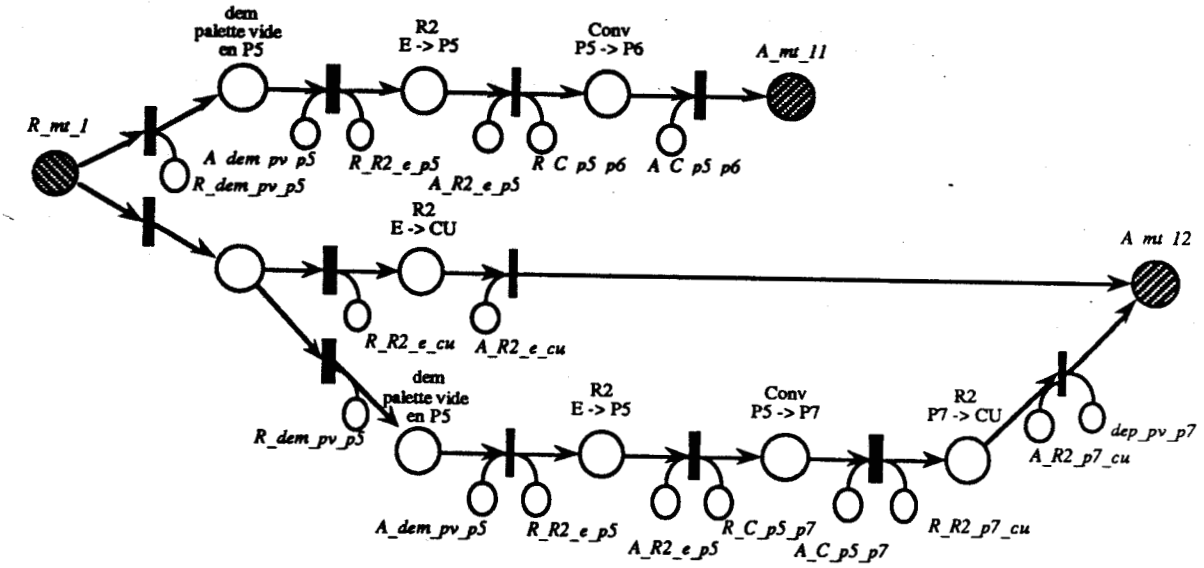
Le producteur/consommateur :

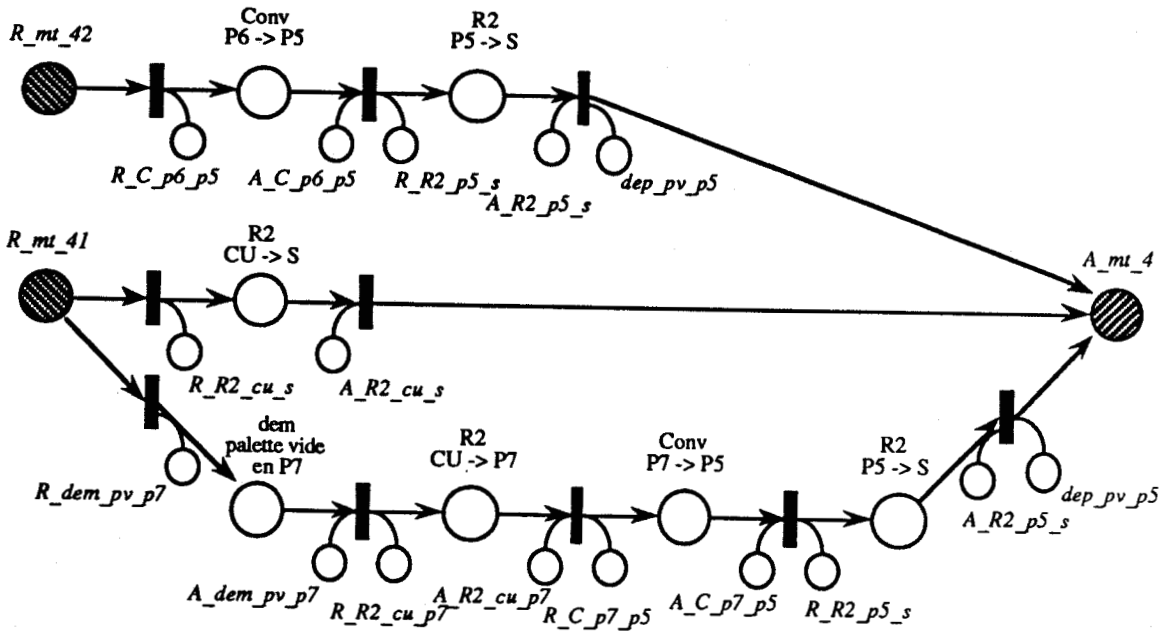


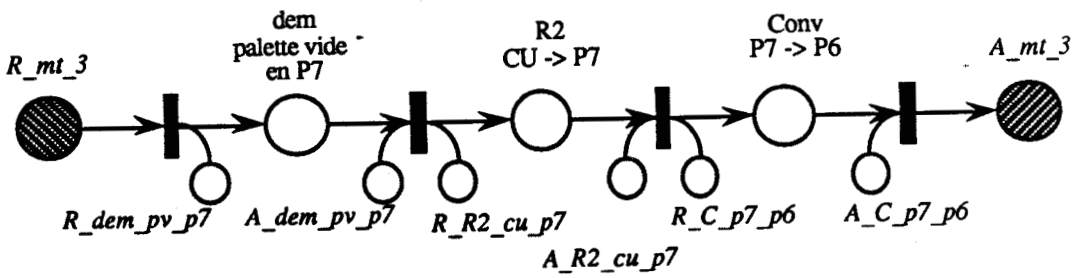
La place PROD permet au processus P2 de consommer des marques produites par le producteur P1.

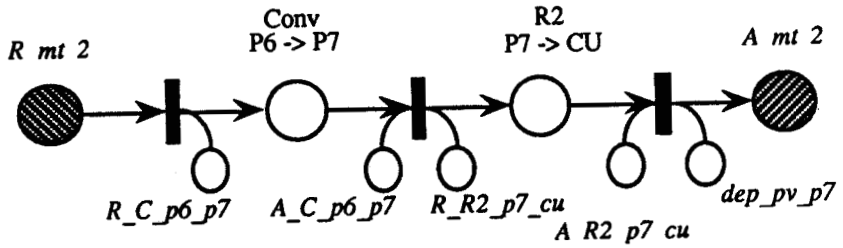
La place CONS permet au processus P2 de relancer la production d'une marque par P1.

I.3. Modèles du groupement moyens de transport.

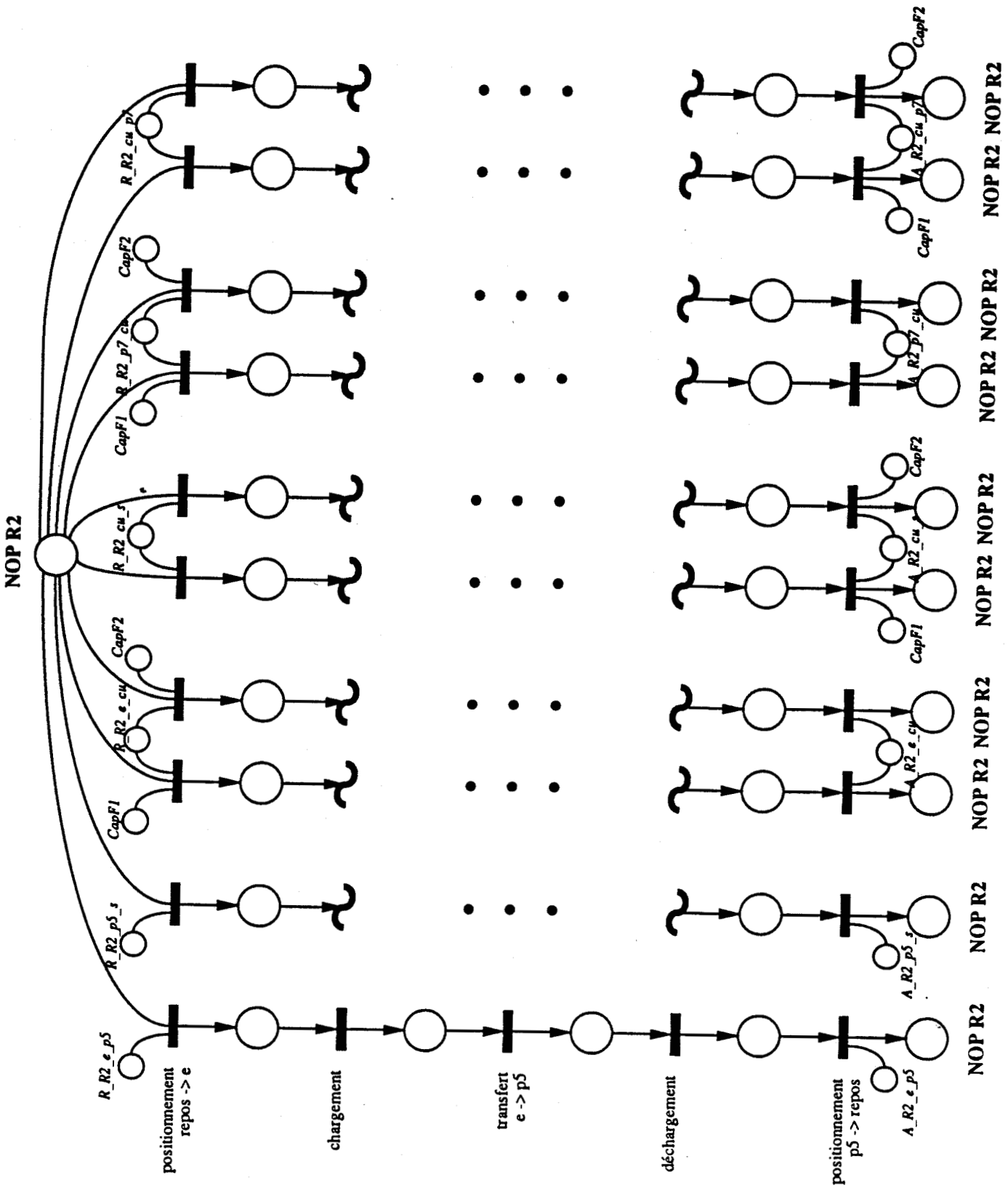





R_mt_3
dem
palette vide
en P7
R2
CU -> P7
Conv
P7 -> P6
A_mt_3
R_dem_pv_p7
A_dem_pv_p7
R_R2_cu_p7
A_R2_cu_p7
R_C_p7_p6
A_C_p7_p6

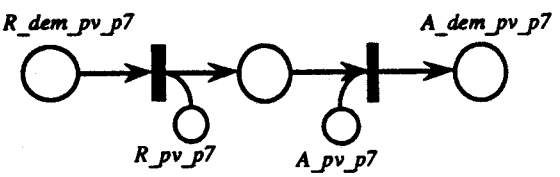

R mt 2
Conv
P6 -> P7
R2
P7 -> CU
A mt 2
R_C_p6_p7
A_C_p6_p7
R_R2_p7_cu
A R2 p7 cu
dep_pv_p7

I.4. Modèle du robot R2.

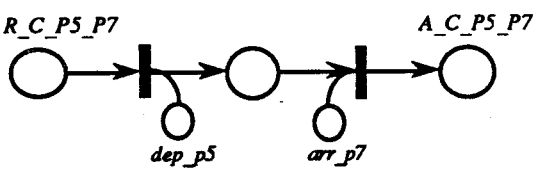


I.5. Modèles du convoyeur.

Exemples de graphes de liaisons entre les modèles du groupement de moyens de transport et les modèles de commande du convoyeur.

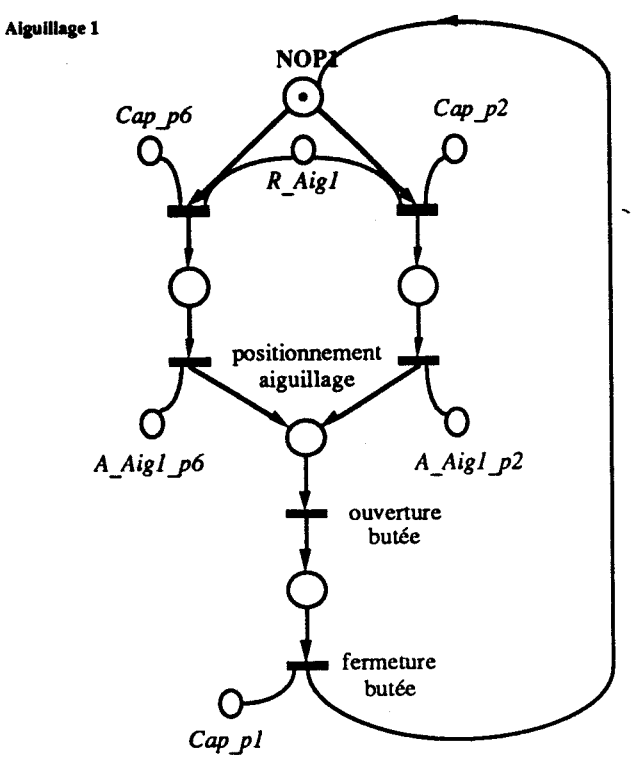


demande palette vide en P7

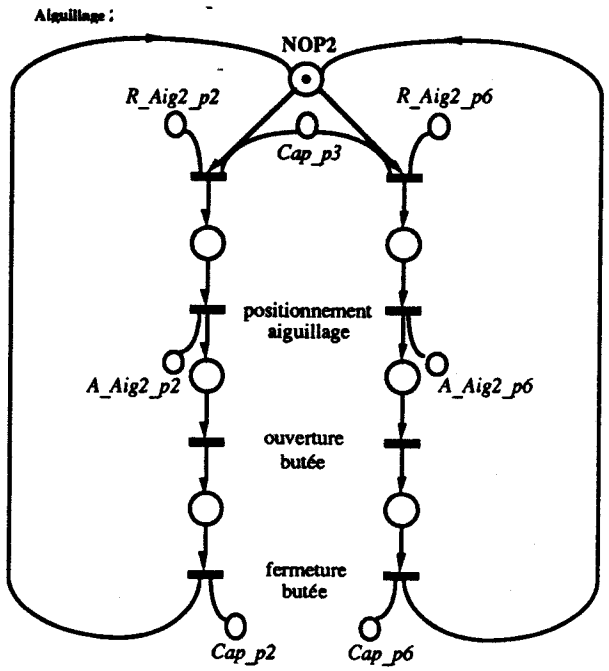


transfert palette pleine P5 -> P7

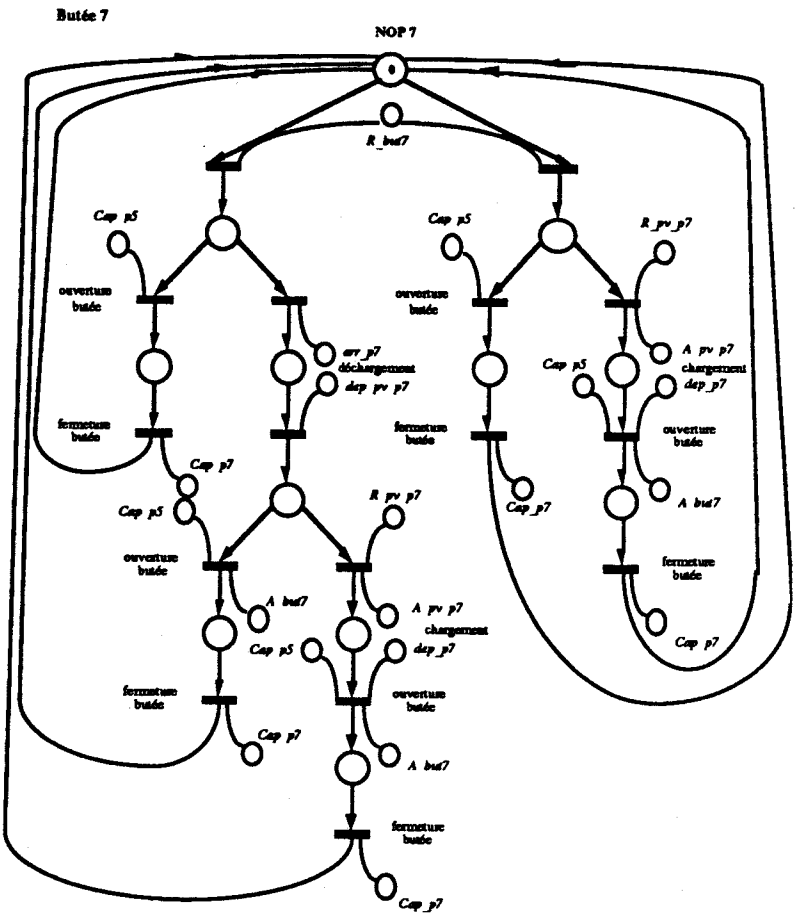
Exemple de graphe de la commande d'un aiguillage.



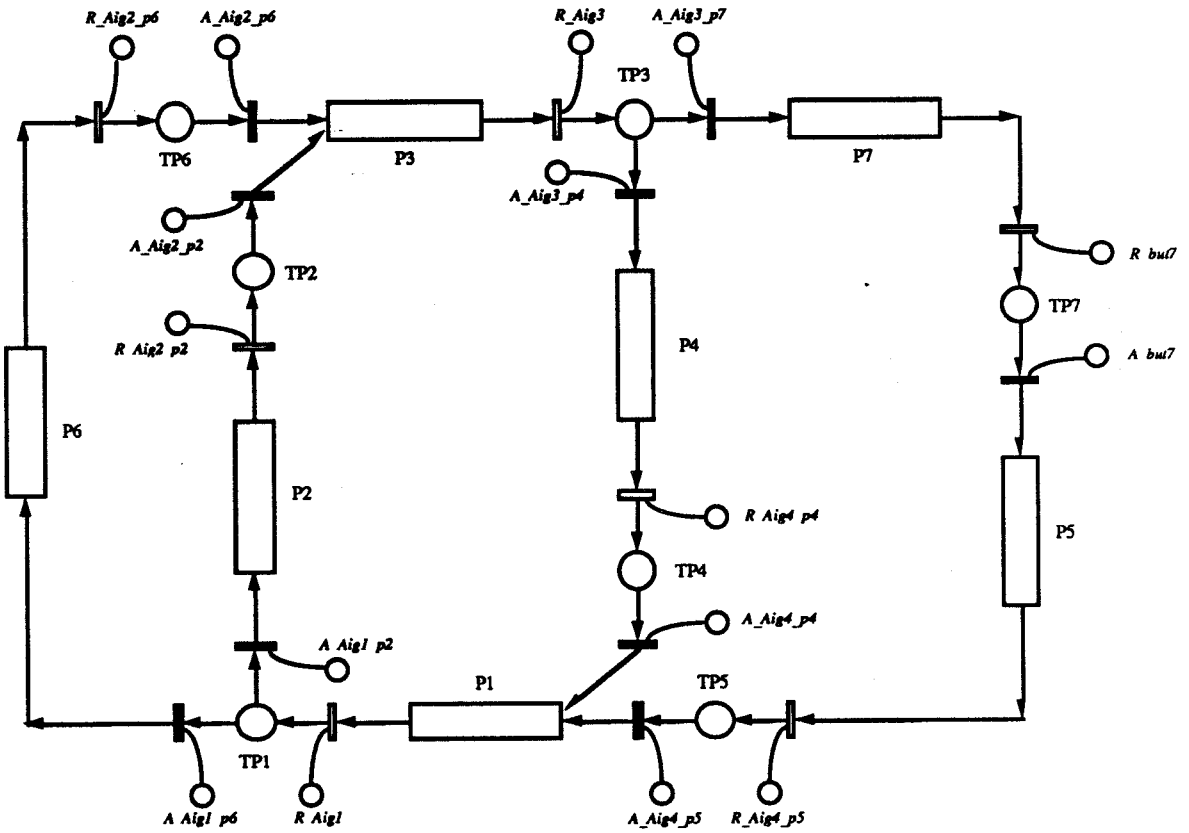
Exemple de graphe de la commande d'une jonction.



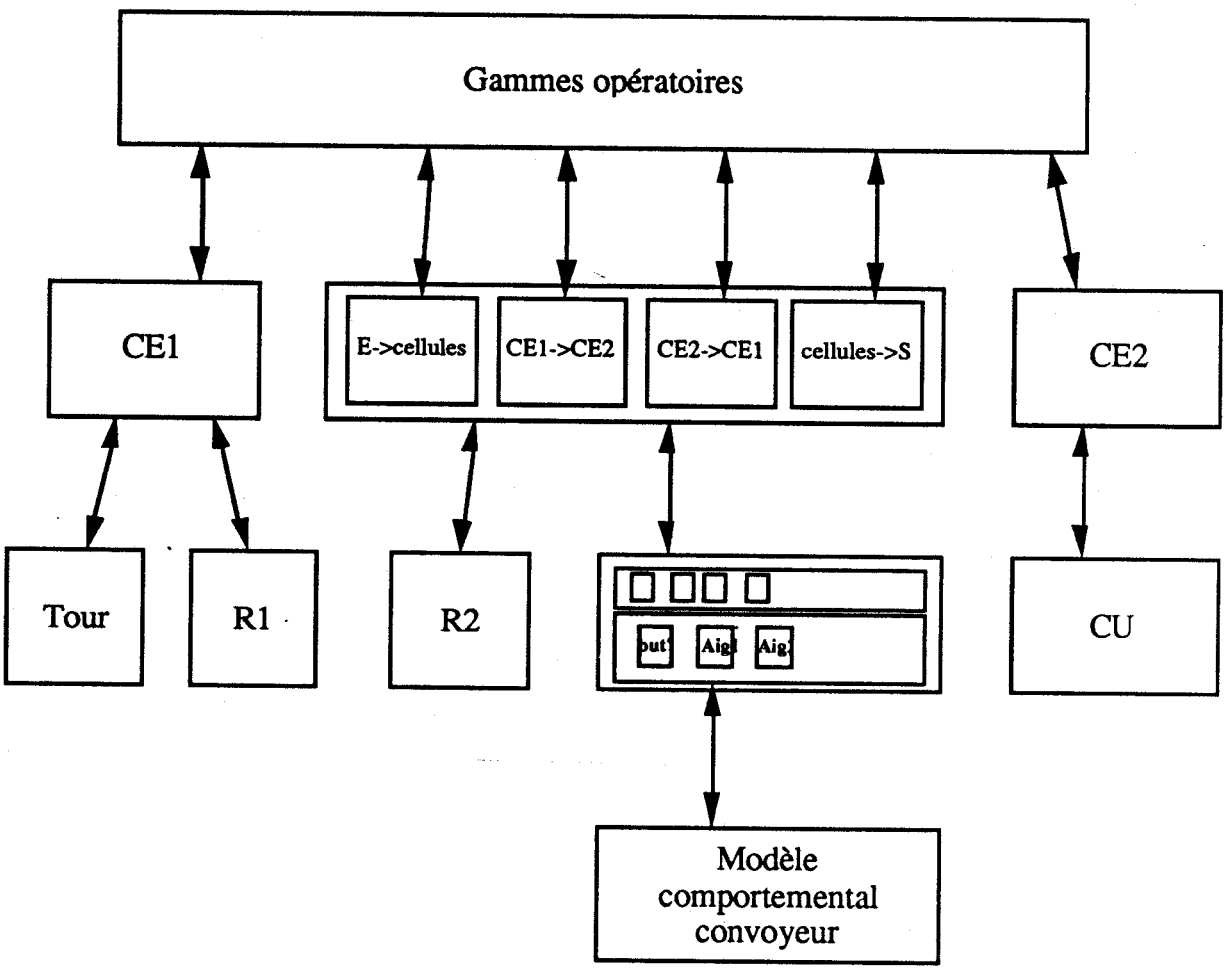
Exemple de graphe de la commande d'une butée.



I.6. Modèle comportemental du convoyeur.



I.7. Graphe récapitulatif des différentes connexions entre modèles.



II.1. Listing du programme de calcul du coût dynamique.

```
/* DESCRIPTION DU RESEAU */

/* Description des liaisons */
successeur(Po1,Po2) ->;
successeur(Po2,Po4) ->;
successeur(Po4,Ai1) ->;
successeur(Ai1,Re2) ->;
successeur(Ai1,Re3) ->;
successeur(Re3,Bi2) ->;
successeur(Bi2,Po1) ->;
successeur(Re2,Bi1) ->;
successeur(Re1,Bi2) ->;
successeur(Bi1,Po3) ->;
successeur(Re4,Bi1) ->;
successeur(Ai2,Re1) ->;
successeur(Ai2,Re4) ->;
successeur(Po3,Ai3) ->;
successeur(Po5,Ai2) ->;
successeur(Ai3,Po5) ->;
successeur(Ai3,Po7) ->;
successeur(Po6,Ai3) ->;
successeur(Po7,Bi3) ->;
successeur(Po13,Po6) ->;
successeur(Po11,Po13) ->;
successeur(Bi3,Po8) ->;
successeur(Po8,Po9) ->;
successeur(Po9,Po10) ->;
successeur(Po10,Ai4) ->;
successeur(Ai4,Po12) ->;
successeur(Ai4,Po11) ->;
successeur(Po12,Bi3) ->;

/* Description des capacités des postes */
capacite(Po1,4) ->;
capacite(Po2,1) ->;
capacite(Po4,4) ->;
capacite(Ai1,1) ->;
capacite(Re3,3) ->;
capacite(Bi2,1) ->;
capacite(Re2,4) ->;
capacite(Re1,4) ->;
capacite(Bi1,1) ->;
capacite(Re4,3) ->;
capacite(Ai2,1) ->;
capacite(Po3,1) ->;
capacite(Ai3,1) ->;
capacite(Po5,1) ->;
capacite(Po7,4) ->;
capacite(Po6,10) ->;
capacite(Po13,8) ->;
capacite(Po11,1) ->;
```

```
capacite(Po12,6) ->;
capacite(Bi3,1) ->;
capacite(Po8,1) ->;
capacite(Ai4,1) ->;
capacite(Po9,1) ->;
capacite(Po10,1) ->;

attente_poste(Po2,10) -> ;
attente_poste(Po3,13) -> ;
attente_poste(Po5,7) -> ;

destination_palette(Pa1,Po5) -> ;
destination_palette(Pa2,Po4) -> ;
destination_palette(Pa3,Po2) -> ;
destination_palette(Pa4,Po2) -> ;
destination_palette(Pa5,Po3) -> ;
destination_palette(Pa6,Po1) -> ;

/*
occupation(x, 0) ->;

chemin(x,y,l) -> chemin'(x,y,l,[]) ;

chemin'(x, y, [x,y], _) ->
        successeur(x, y);
chemin'(x, y, [x|l], l') ->
        successeur(x, z)
        not(membre(z,l'))
        chemin'(z, y, l, [x|l']);

calcul(x,y,l,d) -> calcul'(x,y,l,d,[]) ;

calcul'(x, y, [x,y], d, _) ->
        successeur(x, y)
        evaluation(y, d);
calcul'(x, y, [x|l], d, l') ->
        successeur(x, z)
        not(membre(z,l'))
        evaluation(z, e)
        calcul'(z, y, l, d', [x|l'])
        ,{d = e + d'};

evaluation(x, e) ->
        capacite(x, c)
        occupation(x, o)
        ,{e = c - o};

membre(x,[x|_]) ->;
membre(x, [_|l]) ->
        membre(x,l);
*/
```

```
/* ETAT INITIAL, POSITION DES PALETTES SUR LES DIFFERENTS POSTES */

etat_initial0(
<
<Po1, <>>,
<Po2, <>>,
<Po4, <>>,
<Ai1, <<Pa6,0>> >,
<Re3, <<Pa4,0>> >,
<Bi2, <<Pa5,0>> >,
<Re2, <<Pa1,0>, <Pa2,2>> >,
<Re1, <>>,
<Bi1, <>>,
<Re4, <>>,
<Ai2, <>>,
<Po3, <<Pa3,0>> >,
<Ai3, <>>,
<Po5, <>>,
<Po7, <>>,
<Po6, <>>,
<Po13, <>>,
<Po11, <>>,
<Po12, <>>,
<Bi3, <>>,
<Po8, <>>,
<Ai4, <>>,
<Po9, <>>,
<Po10, <>>
>) -> ;

etat_initial(
<
<Po1, <>>,
<Po2, <>>,
<Po4, <>>,
<Ai1, <<Pa6,0>> >,
<Re3, <<Pa4,0>> >,
<Bi2, <<Pa5,0>> >,
<Re2, <<Pa1,0>, <Pa2,2>> >,
<Re1, <>>,
<Bi1, <>>,
<Re4, <>>,
<Ai2, <>>,
<Po3, <<Pa3,0>> >,
<Ai3, <>>,
<Po5, <>>
>) -> ;

etat_initial1(
<
<Po1, <>>,
<Po2, <<Pa3,0>>>,
<Po4, <>>,
<Ai1, <<Pa6,0>> >,
<Re3, <<Pa4,0>> >,
```

```
<Bi2, <<Pa5,0>> >,
<Re2, <<Pa1,0>, <Pa2,2>> >,
<Re1, <>>,
<Bi1, <>>,
<Re4, <>>,
<Ai2, <>>,
<Po3, <>>,
<Ai3, <>>,
<Po5, <>>
>) -> ;

etat_initial2(
<
<Po1, <<Pa5,0>>>,
<Po2, <<Pa3,0>>>,
<Po4, <>>,
<Ai1, <<Pa6,0>> >,
<Re3, <<Pa4,0>> >,
<Bi2, <> >,
<Re2, <<Pa1,0>, <Pa2,2>> >,
<Re1, <>>,
<Bi1, <>>,
<Re4, <>>,
<Ai2, <>>,
<Po3, <>>,
<Ai3, <>>,
<Po5, <>>
>) -> ;


/* UN PAS D'EVOLUTION DU SYSTEME */

test1 ->
        etat_initial(e)
        parcours(<Pa6,_>, Ai1, Po4, e, e', c, e, n, 0)
        outl(c)
        line
        outl(n)
        line
        fail;

test2 ->
        etat_initial2(e)
        evolution(e,e', e, e2)
        afficher_etat(e')
        line
        fail ;

test3 ->
        etat_initial2(e)
        evolution1(e, e, e', <>, _pv, a, e, e2)
        afficher_etat(e')
        line
        fail ;
```

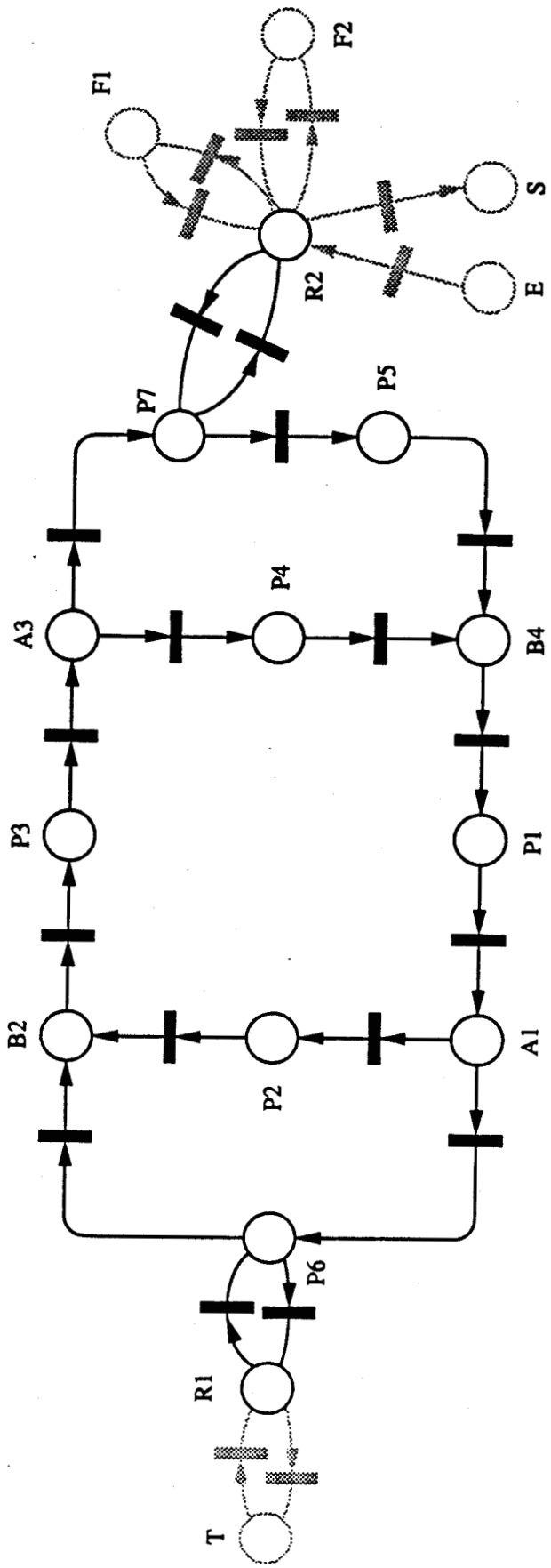
```
test4 ->
        etat_initial1(e)
        bouger_palette(Po2, <Pa3,_>, e, e', <>, e, e2, b1, b2)
        afficher_etat(e')
        line
        fail ;

test5 ->
        etat_initial(e)
        enlever_palette(Ai1, <Pa6,_>, e, e')
        afficher_etat(e')
        line
        fail ;

test6 ->
        etat_initial(e)
        ajouter_palette(Ai1, <Pa6,_>, e, e', e, e2, b1, b2)
        afficher_etat(e')
        line
        fail ;


evolution(_etat1, _etat2, _ev1, _ev2) ->
        evolution1(_etat1, _etat1, _etat', <>, _pv, p_a, _ev1, _ev')
        evolution2(_etat', _etat2, _pv, p_a, _ev', _ev2) ;

/* Une étape d'évolution */
evolution1(<>, _etat, _etat, _pv, _pv, <>, _ev, _ev) -> ;
evolution1(<<p, <> >>.l, _etat1, _etat2, _pv, _pv', p_a, _ev1, _ev2) ->
        evolution1(l, _etat1, _etat2, _pv, _pv', p_a, _ev1, _ev2) ;
evolution1(<< p, <s>.l >>.l', _etat1, _etat2, _pv, _pv', p_a, _ev1, _ev2) ->
        bouger_palette(p, s, _etat1, _etat', _pv, _ev1, _ev', b1, b2)
        evolution11(<< p, <s>.l >>.l', _etat', _etat2, _pv, _pv', p_a, _ev1, _ev2, b1, b2) ;

/* Regarde si une palette est deja passée par un poste */
evolution11(<< p, <s>.l >>.l', _etat1, _etat2, _pv, _pv', p_a, _ev1, _ev2, 1', 1') ->
        evolution1(l', _etat1, _etat2, <s>._pv, _pv', p_a, _ev1, _ev2) ;
evolution11(<< p, <s>.l >>.l', _etat1, _etat2, _pv, _pv', << p, <s>.l >>.p_a, _ev1, _ev2, 0',
1') ->
        evolution1(l', _etat1, _etat2, _pv, _pv', p_a, _ev1, _ev2) ;

/* Mise en attente */

evolution2(_etat, _etat, _pv, <>, _ev, _ev) -> /;
evolution2(_etat1, _etat2, _pv, p_a, _ev1, _ev2) ->
        not(eq(p_a,<>))
        evolution1(p_a, _etat1, _etat', _pv, _pv', p_a', _ev1, _ev')
        not(eq(_pa,_pa')) /
        evolution2(_etat', _etat2, _pv', p_a', _ev', _ev2) ;
evolution2(_etat, _etat, _pv, p_a, _ev, _ev) -> ;
```

```
/* Action de bouger une palette */
bouger_palette(p, s, _etat, _etat, _pv, _ev, _ev, 0', 0') ->
        palette_dans(s, _pv) ;
bouger_palette(p, s, _etat1, _etat2, _pv, _ev1, _ev2, b1, b2) ->
        not(palette_dans(s, _pv))
        bouger_palette1(p, s, _etat1, _etat2, _pv, _ev1, _ev2, b1, b2) ;


bouger_palette1(p, <_pa,_>, _etat1, _etat2, _pv, _ev, _ev, 1', 1') ->
        destination_palette(_pa,p)
        attente_poste(p,n)
        ajouter_attente(n,p, _pa, _etat1, _etat2) /;
bouger_palette1(p, <_pa,0>, _etat1, _etat2, _pv, _ev1, _ev2, 1', 1') ->
        ajouter_palette(p, <_pa,_>, _etat1, _etat', _ev1, _ev2, 1', 1')
        enlever_palette(p, <_pa,_>, _etat', _etat2, _pv) ;
bouger_palette1(p, <_pa,0>, _etat1, _etat2, _pv, _ev1, _ev2, b1, b2) ->
        ajouter_palette(p, <_pa,0>, _etat1, _etat2, _ev1, _ev2, b1, b2) ,
        {b1&b2 = 0'};
bouger_palette1(p, <_pa,n>, _etat1, _etat2, _pv, _ev, _ev, 1', 1') ->
        not(eq(n,0))
        enlever_palette(p, <_pa,n>, _etat1, _etat2, _pv) ;


ajouter_attente(n,p, _pa, << p, <<_pa,0>>.l >>.l', << p, <<_pa,n>>.l >>.l') ->
        /
        retract(destination_palette(_pa,p),_) ;
ajouter_attente(n,p, _pa, << p', 1 >>.l', << p', 1 >>.l") ->
        ajouter_attente(n,p, _pa, l', l") ;


/* Action d'enlever une palette d'un poste */

enlever_palette(p, <_pa,_>, << p, <<_pa, 0>>.l >>.l', << p, l1 >>.l', _pv) ->
        / decrementer_palettes(l, l1, _pv) ;
enlever_palette(p, <_pa,_>, << p, <<_pa, n>>.l >>.l', << p, <<_pa,n-1>>.l1 >>.l', _pv) ->
        / decrementer_palettes(l, l1, _pv) ;
enlever_palette(p, s, << p', l_s>>.l, << p', l_s >>.l', _pv) ->
        enlever_palette(p, s, l, l', _pv) ;

decrementer_palettes(<>, <>, _) -> / ;
decrementer_palettes(<<_pa,n>>.l, <<_pa,n>>.l1, _pv) ->
        palette_dans(<_pa,_>, _pv) /
        decrementer_palettes(l, l1, _pv) ;
decrementer_palettes(<<_pa,n>>.l, <<_pa,n-1>>.l1, _pv) ->
        decrementer_palettes(l, l1, _pv) ;


/* Action d'ajouter une palette sur un poste */
ajouter_palette(p, <_pa,_>, << p', 1 >>.l', << p', l.<<_pa,n-1>> >>.l', << p', _lv >>._ev,
<< p', _lv.<<_pa,_>> >>._ev, 1', 1') ->
        successeur(p, p')
        pas_plein(p', l, 1')
        pas_deja_passe(<_pa,_>, _lv, 1')
        capacite(p',n) ;
```

```
ajouter_palette(p, s, << p', l >>.l', << p', l >>.l', << p', _lv >>._ev, << p', _lv >>._ev, b1,
b2) ->
        successeur(p, p')
        pas_plein(p', l, b1)
        pas_deja_passe(s, _lv, b2) ,
        { b1&b2 = 0' } ;
ajouter_palette(p, s, << p', l >>.l', << p', l >>.l", << p', _lv >>._ev1, << p', _lv >>._ev2,
b1, b2) ->
        ajouter_palette(p, s, l', l", _ev1, _ev2, b1, b2) ;


/* Test si un poste n'est pas plein */
pas_plein(p, l, b) ->
        capacite(p, n)
        bound_size(l,n')
        val(inf(n',n),1) /
        eq(b, 1') ;
pas_plein(_, _, 0') -> ;

/* Test si une palette est déjà passée par un poste */
pas_deja_passe(s, l, b) ->
        not(palette_dans(s, l)) /
        eq(b,1') ;
pas_deja_passe(s, l, 0') -> ;


/* PARCOURS D'UNE PALETTE AVEC EVOLUTION DU SYSTEME */

/*
parcours(s, p1, p2, _etat1, _etat2, <p1,p2>, _ev1, 1) ->
        evolution(_etat1, _etat2, _ev1, _ev2)
        premiere_palette_dans_poste(p2, s, _etat2) / ;
parcours(s, p1, p2, _etat1, _etat2, <p1>.l_c, _ev1, n+1) ->
        evolution(_etat1, _etat', _ev1, _ev2)
        palette_dans_poste(p, s, _etat')
outm(" > poste : ") outl(p)
        parcours(s, p, p2, _etat', _etat2, l_c, _ev2, n) ;
*/

longueur_max(9999) -> ;

parcours(s, p1, p2, _etat1, _etat2, <p1>.l_c, _ev1, n, m) ->
        evolution(_etat1, _etat', _ev1, _ev2)
        parcours1(s, p1, p2, _etat', _etat2, l_c, _ev2, n, m+1) ;

parcours1(s, p1, p2, _etat, _etat, <p2>, _, m, m) ->
        premiere_palette_dans_poste(p2, s, _etat)
        longueur_max(q)
        val(inf(m,q),1) /
        retract(longueur_max(_),[])
        assert(longueur_max(m),[]) ;
```

```
parcours1(s, p1, p2, _etat1, _etat2, l_c, _ev, n, m) ->
        palette_dans_poste(p, s, _etat1)
        longueur_max(q)
        val(inf(m,q),1) /
outm(" > poste : ") outl(p)
        parcours(s, p, p2, _etat1, _etat2, l_c, _ev, n, m) ;

/* Teste si une palette est en premiere position dans un poste */
premiere_palette_dans_poste(p,<_pa,_>,<< p, <<_pa,0>>._ >>._) -> / ;
premiere_palette_dans_poste(p,s,<< p', _ >>.l) ->
        premiere_palette_dans_poste(p,s,l) ;

/* Test de présence d'une palette dans un poste */
palette_dans_poste(p,s,<< p,l>>._) ->
        palette_dans(s,l) ;
palette_dans_poste(p,s,<< p', _ >>.l) ->
        palette_dans_poste(p,s,l) ;


/* UTILITAIRES */

/* Affichage d'un état du système */
afficher_etat(<>) ->
        outml("---------------") ;
afficher_etat(<< p, l >>.l') ->
        outm(" poste : ") out(p) outm(" / ")
        outm(" palettes : ") outl(l)
        afficher_etat(l') ;

/* Test de présence d'un élément dans un t-uple */
palette_dans(<_pa,_>, <<_pa,_>>._) -> ;
palette_dans(s, <_>.l) ->
        palette_dans(s, l) ;



trace1 ->
        set_config("trace",1);

trace2 ->
        set_config("trace",2);


;;
```

II.2. Modèle comportemental général.



III.1. Fichier décrivant un prégraphe pour l'exemple de la cellule de l'Ecole Centrale de Lille.

Liste des places

(e p5 p1 p2 t p4 p3 f s p6)

Liste des transitions

(e->p5 p5->p1 p5->p3 p1->p2 p2->p4 p2->t t->p2 p4->p1 p4->p3 p3->p6
p3->f p3->s f->s f->p5 p6->p5 t->t tournage fraisage)

Les places

nom	**e**
predecesseur	
successeur	e->p5
nom	**p5**
predecesseur	p6->p5, f->p5, e->p5
successeur	f->p5, p5->p1, p5->p3, e->p5
nom	**p6**
predecesseur	p3->p6
successeur	p6->p5
nom	**p1**
predecesseur	p5->p1,
successeur	p1->p2
nom	**p2**
predecesseur	p1->p2, p2->t, t->p2
successeur	p2->t, t->p2, p2->p4
nom	**p4**
predecesseur	p2->p4
successeur	p4->p3
nom	**t**
predecesseur	p2->t, tournage, t->t
successeur	t->t, t->p2, tournage
nom	**p3**
predecesseur	p5->p3, p4->p3, p3->s, p3->f
successeur	p3->s, p3->f, p3->p6
nom	**s**
predecesseur	p3->s, f->s
successeur	
nom	**f**
predecesseur	p3->f, fraisage
successeur	f->p5, f->s, fraisage

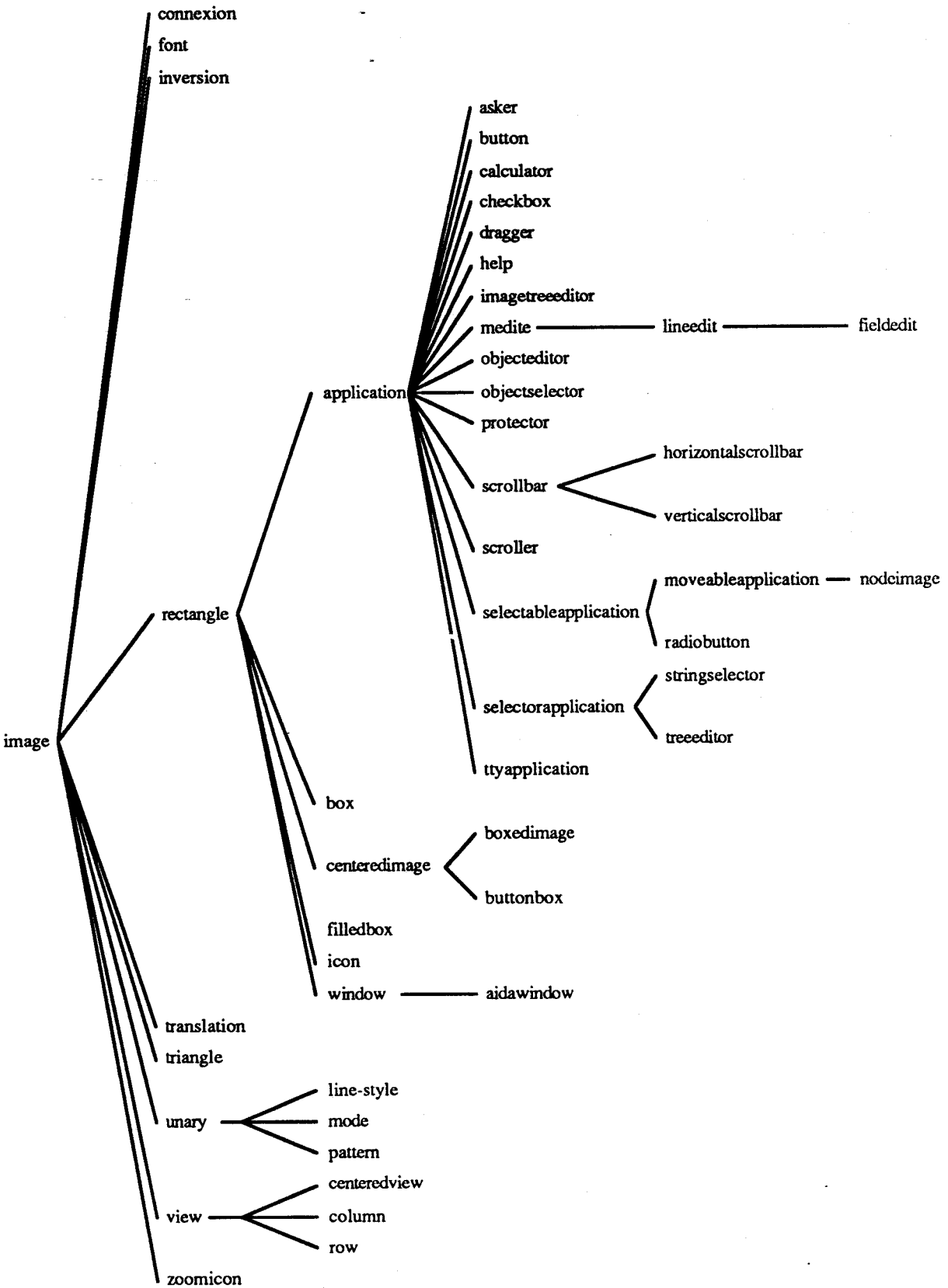
Les transitions

nom	**e->p5**
nature	assemblage
assemblage	p5
var-libre	a
dom-a	t- f- t-t- f-t- t-f-
arc-pre	(1 a e) (1 p p5)
arc-post	(1 (e->p5 a) p5)
nom	**p6->p5**
nature	transfert
arc-pre	(1 p p6)
arc-post	(1 p p5)

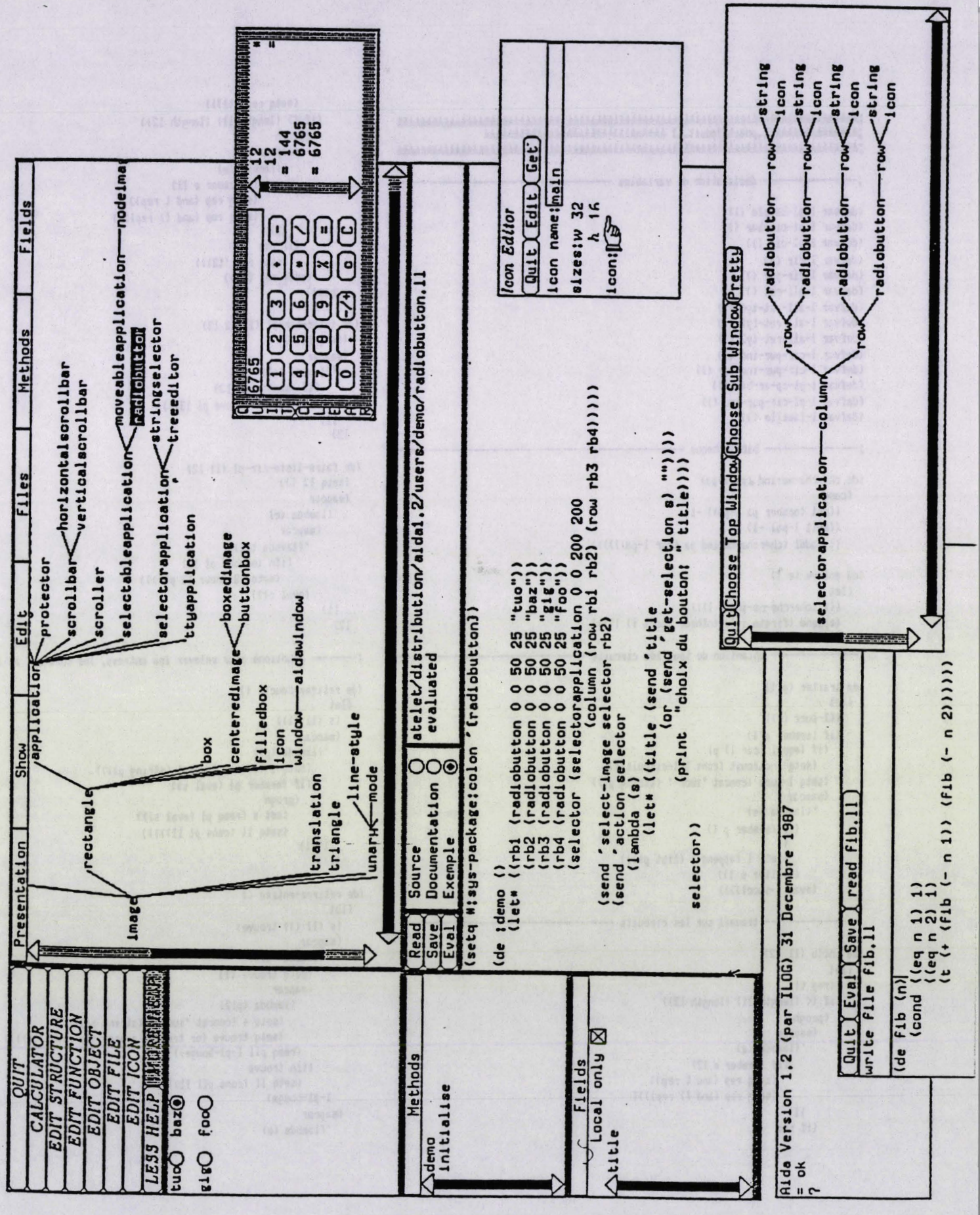
nom	**f->p5**
nature	assemblage
assemblage	p5
arc-pre	(1 f+t- f) (1 p p5)
arc-post	(1 f+t- p5)
nom	**p5->p1**
nature	transfert
var-libre	c
dom-c	pt- pt-t- pt-f- pf+t-
arc-pre	(1 c p5)
arc-post	(1 c p1)
nom	**p1->p2**
nature	transfert
var-libre	e
dom-e	pt- pt-t- pt-f- pf+t- pt+t-bp
arc-pre	(1 e p1)
arc-post	(1 e p2)
nom	**p2->t**
nature	desassemblage
dessassemblage	p2
var-libre	e
dom-e	pt- pt-t- pt-f- pt+f- pt+t-bp
arc-pre	(1 e p2)
arc-post	(1 (p2->t e) t) (1 p p2)
nom	**tournage**
nature	usinage
var-libre	e
dom-e	t- t-t- t-f- f+t- t+t-bp
arc-pre	(1 e t)
arc-post	(1 (tournage e) t)
nom	**t->t**
nature	positionnement
arc-pre	(1 t+t-mp t)
arc-post	(1 t+t-bp t)
nom	**t->p2**
nature	asssemblage
assemblage	p2
var-libre	e*
dom-e*	t+ t+f- f+t+ t+t+
arc-pre	(1 p p2) (1 e* t)
arc-post	(1 (t->p2 e*) p2)
nom	**p2->p4**
nature	transfert
var-libre	e*
dom-e*	pt+ pt+f- pf+t+ pt+t+
arc-pre	(1 e* p2)
arc-post	(1 e* p4)
nom	**p4->p3**
nature	transfert
var-libre	u
dom-u	pt+ pt+f- pf+t+ pt+t+
arc-pre	(1 u p4)
arc-post	(1 u p3)

nom	**p5->p3**
nature	transfert
var-libre	d
dom-d	pf- pf-t-
arc-pre	(1 d p5)
arc-post	(1 d p3)
nom	**p3->s**
nature	dessassemblage
dessassemblage	p3
var-libre	w
dom-w	pt+ pf+t+ pt+t+
arc-pre	(1 w p3)
arc-post	(1 p p3) (1 (p3->s w) s)
nom	**p3->f**
nature	dessassemblage
dessassemblage	p3
var-libre	x
dom-x	pt+f- pf- pf-t-
arc-pre	(1 x p3)
arc-post	(1 p p3) (1 (p3->f x) f)
nom	**p3->p6**
nature	transfert
arc-pre	(1 p p3)
arc-post	(1 p p6)
nom	**fraisage**
nature	usinage
var-libre	x
dom-x	t+f- f- f-t-
arc-pre	(1 x f)
arc-post	(1 (fraisage x) f)
nom	**f->s**
nature	transfert
var-libre	y
dom-y	t+f+ f+
arc-pre	(1 y f)
arc-post	(1 y s)

III.2. Arborescence des objets prédéfinis d'Aïda.



III.3. Exemple de l'environnement d'Aïda.



III.4. Listing du programme d'analyse des graphes.

```
;*****************************************************************************
;*************** [.posit3]posit.ll ***********************************
;*****************************************************************************

;-------------- declaration de variables ------------------------------------

(defvar l-pl-boucle ())
(defvar l-pl-cir-bar ())
(defvar l-pl-cir ())
(defvar l-cir ())
(defvar l-cir-pur ())
(defvar l-all-ret ())
(defvar l-all-ret-ty0 ())
(defvar l-all-ret-ty1 ())
(defvar l-all-ret-ty2 ())
(defvar l-cir-pur-ind ())
(defvar l-cir-pur-ind-bar ())
(defvar l-pl-cp-ar-bar ())
(defvar l-pl-cir-pur-ind ())
(defvar l-famille ())

;-------------- bibliotheque ------------------------------------------------

(de cherche-no-ind (pa l-pa)
  (cond
    ((not (member pa l-pa)) -1)
    ((null l-pa) -1)
    (t (addl (cherche-no-ind pa (cdr l-pa))))))

(de enleve (e l)
  (let
    ((i (cherche-no-ind e l)))
    (append (firstn i l) (nthcdr (addl i) l))))

;-------------- obtention de tous les circuits ------------------------------

(de traiter (p l)
  (let
    ((l-succ ()))
    (if (member p l)
      (if (equal (car l) p)
        (setq l-circuit (cons l l-circuit)))
      (setq l-succ (concat 'succ- (string p)))
      (mapcar
        '(lambda (s)
           (if (member p l)
             ()
             (setq l (append l (list p))))
           (traiter s l))
        (eval l-succ)))))

;-------------- travail sur les circuits ------------------------------------

(de inclu (l1 l2)
  (let
    ((rep t))
    (if (< (length l1) (length l2))
      (progn
        (mapcar
          '(lambda (e)
             (if (member e l2)
               (setq rep (and t rep))
               (setq rep (and () rep))))
          l1)
        (if rep
          (setq rep 't1)))
      (if () (length l1) (length l2))
        (progn
          (mapcar
            '(lambda (e)
               (if (member e l1)
                 (setq rep (and t rep))
                 (setq rep (and () rep))))
            l2)
          (if rep
            (setq rep 't2)))
        (setq rep ())))
    rep))

(de inter-bar-pl (l1 l2 l3)
  (setq l3 l2)
  (mapcar
    '(lambda (pl)
       (if (member pl l3)
         (setq l3 (enleve pl l3))))
    l1)
  l3)

(de faire-liste-cir-pl (l1 l2)
  (setq l2 ())
  (mapcar
    '(lambda (c)
       (mapcar
         '(lambda (pl)
            (ifn (member pl l2)
              (setq l2 (newr l2 pl))))
         (eval c)))
    l1)
  l2)

;-------- fonctions pour enlever les entrees, les sorties, et les bo

(de retirer-boucle ()
  (let
    (s (l1 ()))
    (mapcar
      '(lambda (pl)
         (setq s (concat 'succ- (string pl)))
         (if (member pl (eval s))
           (progn
             (set s (remq pl (eval s)))
             (setq l1 (cons pl l1)))))
      l-pl)
    l1))

(de retirer-entree ()
  (let
    (s (l1 ()) trouve)
    (mapcar
      '(lambda (pl1)
         (setq trouve ())
         (mapcar
           '(lambda (pl2)
              (setq s (concat 'succ- (string pl2)))
              (setq trouve (or trouve (member pl1 (eval s)))))
           (remq pl1 l-pl-bouge))
         (ifn trouve
           (setq l1 (cons pl1 l1))))
      l-pl-bouge)
    (mapcar
      '(lambda (e)
```

```
        (setq l-pl-bouge (remq e l-pl-bouge)))
      ll)
    ll))

(de retirer-sortie ()
  (let
    ((ll ()) s)
    (mapcar
      '(lambda (pl)
         (setq s (concat 'succ-' (string pl)))
         (if (null (eval s))
           (setq ll (cons pl ll))))
      l-pl-bouge)
    (mapcar
      '(lambda (e)
         (setq l-pl-bouge (remq e l-pl-bouge))
         (mapcar
           '(lambda (pl)
              (setq s (concat 'succ-' (string pl)))
              (if (member e (eval s))
                (set s (remq e (eval s)))))
           l-pl-bouge))
      ll)
    ll))

(de retirer-entree-sortie ()
  (let
    ((bouge t) (ll ()))
    (while bouge
      (setq l-pl-bouge l-pl)
      (setq ll (append ll (retirer-entree)))
      (setq ll (append ll (retirer-sortie)))
      (setq bouge (not (equal l-pl l-pl-bouge)))
      (setq l-pl l-pl-bouge))
    ll))

;---------- programme principal ----------------------------------------

(de letop ()
  (let
    (nom s l-pl-bouge l-pl2 i l-circuit l-cir-loc l-cir-vu rep
     num nom-sauve l-pl l-pl-cp l-pl-ar nb-max
     l-pl-max-cir-pur-ind fa nb-connexion l-pl-nb-connexion)

;-- chargement du fichier a traiter
    (print 'Quel fichier voules vous charger ? : ')
    (setq nom (string (read)))
    (loadfile (catenate (string nom) '.pre') ())
    (setq l-circuit ())

;-- initialisation avant recherche circuit

    (setq l-pl places)
    (setq l-pl-boucle ())
    (mapcar
      '(lambda (pl)
         (setq s (concat 'succ-' (string pl)))
         (set s (getprop pl 'successeur)))
      places)
    (setq l-pl-boucle (retirer-boucle))

    (setq l-pl-cir-bar (retirer-entree-sortie))
    (setq l-pl-cir l-pl)

;-- application de l'algo de recherche de circuit
    (setq l-pl2 l-pl)
    (setq l-circuit ())
    (mapcar
      '(lambda (pl)
         (traiter pl ())
         (mapcar
           '(lambda (cir)
              (let
                (pl p2 s)
                (setq pl (car cir))
                (setq p2 (cadr cir))
                (setq s (concat 'succ-' (string pl)))
                (set s (remq p2 (eval s)))))
           l-circuit)
         (retirer-entree-sortie))
      l-pl2)

;-- creation des circuits et de la liste correspondante -> c-i, l-cir
    (setq l-cir ())
    (setq i 0)
    (mapcar
      '(lambda (c)
         (let
           (cir)
           (setq i (addl i))
           (setq cir (concat 'c-' (string i)))
           (set cir c)
           (setq l-cir (newr l-cir cir))))
      l-circuit)

;-- creation des listes aller-retour et circuits purs -> l-all-ret, l-cir-p
    (setq l-all-ret ())
    (setq l-cir-pur ())
    (mapcar
      '(lambda (c)
         (if (equal (length (eval c)) 2)
           (setq l-all-ret (newr l-all-ret c))
           (setq l-cir-pur (newr l-cir-pur c))))
      l-cir)

;-- creation de la liste des circuits purs independants -> l-cir-pur-ind
    (setq l-cir-pur-ind ())
    (setq l-cir-loc l-cir-pur)
    (setq l-cir-vu ())
    (if (or (equal (length l-cir-pur) 0) (equal (length l-cir-pur) 1))
      (setq l-cir-pur-ind l-cir-pur)
      (mapcar
        '(lambda (c1)
           (setq l-cir-loc (cdr l-cir-loc))
           (mapcar
             '(lambda (c2)
                (setq rep (inclu (eval c1) (eval c2)))
                (cond
                  ((equal rep ())
                   (ifn (member c1 l-cir-vu)
                     (ifn (member c1 l-cir-pur-ind)
                       (setq l-cir-pur-ind (newr l-cir-pur-ind c1))))
                   (ifn (member c2 l-cir-vu)
                     (ifn (member c2 l-cir-pur-ind)
                       (setq l-cir-pur-ind (newr l-cir-pur-ind c2)))))
                  ((equal rep 't1)
                   (ifn (member c1 l-cir-vu)
                     (setq l-cir-vu (newr l-cir-vu c1)))
                   (if (member c1 l-cir-pur-ind)
                     (setq l-cir-pur-ind (enleve c1 l-cir-pur-ind)))
                   (ifn (member c2 l-cir-vu)
                     (ifn (member c2 l-cir-pur-ind)
```

```
                (setq l-cir-pur-ind (newr l-cir-pur-ind c2)))))
              ((equal rep 't2)
               (ifn (member c2 l-cir-vu)
                 (setq l-cir-vu (newr l-cir-vu c2)))
               (if (member c2 l-cir-pur-ind)
                 (setq l-cir-pur-ind (enleve c2 l-cir-pur-ind)))
               (ifn (member c1 l-cir-vu)
                 (ifn (member c1 l-cir-pur-ind)
                   (setq l-cir-pur-ind (newr l-cir-pur-ind c1)))))
              (t ())))
          l-cir-loc))
      l-cir-pur))

;-- creation de la liste des circuits purs dependants -> l-cir-pur-ind-bar
    (setq l-cir-pur-ind-bar ())
    (mapcar
      '(lambda (cir)
         (ifn (member cir l-cir-pur-ind)
           (setq l-cir-pur-ind-bar (newr l-cir-pur-ind-bar cir))))
      l-cir-pur)

;-- creation de la liste des places appartenant a un aller-retour mais
;-- pas a un circuit pur -> l-pl-cp-ar-bar
   (setq l-pl-cp (faire-liste-cir-pl l-cir-pur l-pl-cp))
   (setq l-pl-ar (faire-liste-cir-pl l-all-ret l-pl-ar))
   (setq l-pl-cp-ar-bar (inter-bar-pl l-pl-cp l-pl-ar l-pl-cp-ar-bar))

;-- creation de la liste des places appartenant aux circuits purs
;-- -> l-pl-cir-pur
   (setq l-pl-cir-pur-ind ())
   (mapcar
     '(lambda (cir)
        (mapcar
          '(lambda (pl)
             (if (member pl l-pl-cir-pur-ind)
               (set pl (add1 (eval pl)))
               (setq l-pl-cir-pur-ind (newr l-pl-cir-pur-ind pl))
               (set pl 1)))
          (eval cir)))
     l-cir-pur-ind)

   (setq l-all-ret-ty0 ())
   (setq l-all-ret-ty1 ())
   (setq l-all-ret-ty2 ())
   (mapcar
     '(lambda (cir)
        (cond
          ((and (member (car (eval cir)) l-pl-cir-pur-ind)
                (member (cadr (eval cir)) l-pl-cir-pur-ind))
           (setq l-all-ret-ty2 (newr l-all-ret-ty2 cir)))
          ((or (member (car (eval cir)) l-pl-cir-pur-ind)
               (member (cadr (eval cir)) l-pl-cir-pur-ind))
           (setq l-all-ret-ty1 (newr l-all-ret-ty1 cir)))
          (t (setq l-all-ret-ty0 (newr l-all-ret-ty0 cir)))))
     l-all-ret)

   (setq nb-max 0)
   (setq l-pl-max-cir-pur-ind ())
   (mapcar
     '(lambda (pl)
        (cond
          ((equal nb-max (eval pl))
           (setq l-pl-max-cir-pur-ind (cons pl l-pl-max-cir-pur-ind)))
          ((< nb-max (eval pl))
           (setq l-pl-max-cir-pur-ind (list pl))
           (setq nb-max (eval pl)))
          (t ())))
     l-pl-cir-pur-ind)
(print l-pl-max-cir-pur-ind)

   (setq l-famille ())
   (cond
     ((or (equal (length l-cir-pur-ind) 0) (equal (length l-cir-pur-in
      ())
     ((equal (length l-cir-pur-ind) (eval (car l-pl-max-cir-pur-ind)))
      (setq fa (concat 'fam-1'))
      (set fa l-cir-pur-ind)
      (setq l-famille (newr l-famille fa)))
     (t
(print 'nb-fam = ' (add1 (sub (length l-cir-pur-ind)
                              (eval (car l-pl-max-cir-pur-ind)))))
      (setq i 0)
      (setq l-pl-nb-connexion l-pl-max-cir-pur-ind)
      (setq nb-connexion (eval (car l-pl-max-cir-pur-ind)))
      (while (>= nb-connexion 2)
        (mapcar
          '(lambda (pl)
             (setq i (add1 i))
             (setq fa (concat 'fam-' (string i)))
             (set fa ())
             (setq l-famille (newr l-famille fa))
             (mapcar
               '(lambda (cir)
                  (if (and (member pl (eval cir)) (not (member cir (eva
                    (set fa (reverse (cons cir (eval fa))))))
                l-cir-pur-ind)
             (setq f (reverse l-famille))
             (setq e (eval (car f)))
             (setq f (reverse (cdr f)))
             (setq appart1 ())
             (mapcar
               '(lambda (ef)
                  (setq appart2 t)
                  (mapcar
                    '(lambda (cir)
                       (if (member cir (eval ef))
                         (setq appart2 (and appart2 t))
                         (setq appart2 (and appart2 ()))))
                    e)
                  (setq appart1 (or appart1 appart2)))
               f)
             (if appart1
               (progn
                 (setq l-famille f)
                 (setq i (sub1 i)))))
          l-pl-nb-connexion)
        (setq nb-connexion (sub1 nb-connexion))
        (setq l-pl-nb-connexion ())
        (mapcar
          '(lambda (pl)
             (if (equal (eval pl) nb-connexion)
               (setq l-pl-nb-connexion (reverse (cons pl l-pl-nb-connex
          l-pl-cir-pur-ind))))


 ;-- sauvegarde des donnees creees
    (setq nom-sauve (catenate (string nom) '.cir'))
    (setq num (openo nom-sauve))
    (outchan num)
    (print (list 'setq 'nomfichier (kwote nom-sauve)))
    (print)
    (print)
```

```
    (print (list 'setq 'places (kwote places)))
    (print)
    (print '  ' (list 'setq 'l-pl-boucle (kwote l-pl-boucle)))
    (print)
    (print '  ' (list 'setq 'l-pl-cir-bar (kwote l-pl-cir-bar)))
    (print)
    (print '  ' (list 'setq 'l-pl-cir (kwote l-pl-cir)))
    (print)
    (print '    ' (list 'setq 'l-pl-cp-ar-bar (kwote l-pl-cp-ar-bar)))
    (print)
    (print)
    (print (list 'setq 'l-cir (kwote l-cir)))
    (print)
    (mapcar
      '(lambda (c)
         (print (list 'setq c (kwote (eval c)))))
      l-cir)
    (print)
    (print '  ' (list 'setq 'l-all-ret (kwote l-all-ret)))
    (print)
    (print '    ' (list 'setq 'l-all-ret-ty0 (kwote l-all-ret-ty0)))
    (print)
    (print '    ' (list 'setq 'l-all-ret-ty1 (kwote l-all-ret-ty1)))
    (print)
    (print '    ' (list 'setq 'l-all-ret-ty2 (kwote l-all-ret-ty2)))
    (print)
    (print '  ' (list 'setq 'l-cir-pur (kwote l-cir-pur)))
    (print)
    (print '    ' (list 'setq 'l-cir-pur-ind (kwote l-cir-pur-ind)))
    (print)
    (print '      ' (list 'setq 'l-pl-cir-pur-ind (kwote l-pl-cir-pur-ind)))
    (print)
    (print '      ' (list 'setq 'l-famille (kwote l-famille)))
    (print)
    (mapcar
      '(lambda (f)
         (print '        ' (list 'setq f (kwote (eval f)))))
      l-famille)
    (print)
    (print '    ' (list 'setq 'l-cir-pur-ind-bar (kwote l-cir-pur-ind-bar)))
    (print)
    (outchan ())
    (close num)
)
  t)
```

III.5. Algorithme de la méthode de positionnement automatique.

Traitement principal

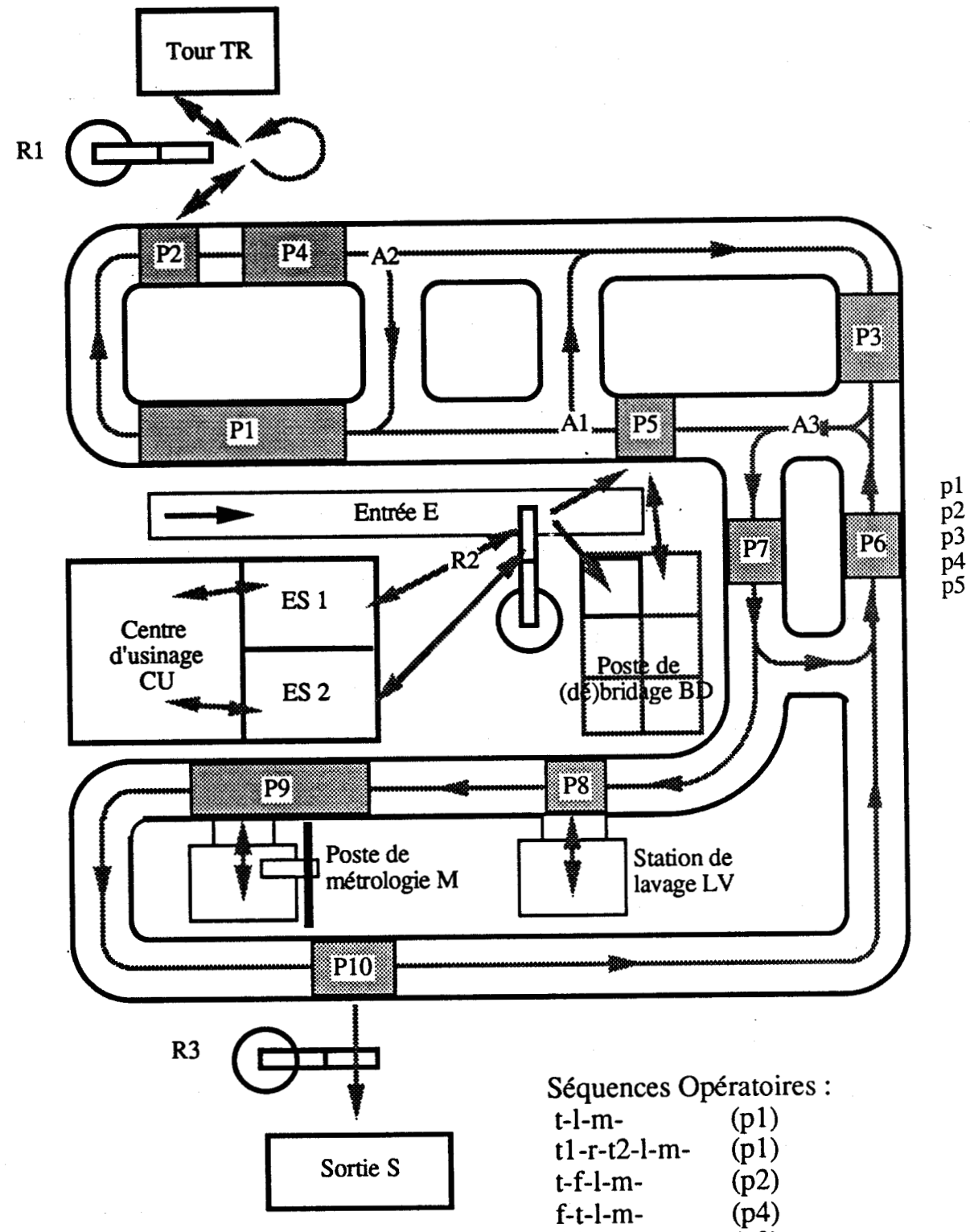
```
cas nombre de circuits purs indépendants faire
        0 :  si il n'y a pas d'aller-retour alors
                analyse du fichier
                positionnement des places "entrées"
                positionnement des autres places par balayage vertical du tableau
             sinon
                cas-aller-retour {même traitement que celui des circuits purs indépendants}
             fsi
        1 :  positionner le circuit pur indépendant {loi de positionnement}
             traitement
     autre :  si une seule famille alors
                si deux circuits ont plus d'une place en commun alors
                   positionner le circuit pur indépendant qui a le plus grand nombre de
                   places
                   pour tous les autres circuits purs indépendants faire
                      les "positionner" avec "bouge" éventuel
                   fait
                   traitement
                sinon
                   positionner les circuits purs indépendants
                   traitement
                fsi
             sinon
                si deux familles ont un circuit en commun alors
                   positionnement des circuits des deux familles
                   traitement
                sinon
                   positionnement des circuits de la famille composée du + grand nombre
                   traitement
                fsi
             fsi
  fait
fin
```

Traitement

pour tous les aller-retours de type 2 faire
si niv(p) = niv(q) alors
bouger les places entre p et q sur le contour enveloppe
fsi
fait
pour tous les circuits purs non indépendants faire
"positionner" avec "bouge" éventuel
fait
pour tous les aller-retours de type 1 faire
positionner la place n'appartenant à aucun circuit pur
fait
pour toutes les places n'appartenant à aucun circuit et en liaison avec une place positionnée faire
positionner la place
fait
pour tous les aller-retours de type 0 dont les deux places sont positionnées faire
si niv(p) = niv(q) alors
bouger les places entre p et q sur le contour enveloppe
fsi
pour tous les aller-retours de type 0 dont une des places est positionnée faire
positionner la place non encore positionnée
fait
pour toutes les places n'appartenant à aucun circuit et en liaison avec une place positionnée faire
positionner la place
fait
pour les aller-retours de type 0 restants faire
positionner l'aller-retour
fait
pour toutes les places n'appartenant à aucun circuit et en liaison avec une place positionnée faire
positionner la place
fait
fin

III.6. Premier exemple.

Plan de la cellule :



p1
p2
p3
p4
p5

Séquences Opératoires :

t-l-m-	(p1)
t1-r-t2-l-m-	(p1)
t-f-l-m-	(p2)
f-t-l-m-	(p4)
f-l-m-	(p3)
t1-f-t2-l-m-	(p5)

Fichier d'entrée :

Liste des places

(s2 s1 s4 lv s8 m s9 s e cu s3 s7 cr s10 s6 tr bd s5)

Liste des transitions

(e->s5 s5->s2 s2->s1 s1->tr tournage tr->s1 s1->s4 s4->s3 s7->s8 s3->cr cr->s7 s8->lv lavage lv->s8 s8->s9 s9->m mesures m->s9 s9->s10 s10->s e->bd bd->cu fraisage cu->bd bd->s5 s5->s3 s7->s6 s6->s7 s6->cr cr->s5 s10->s6 retournement s5->bd s4->s2 s3->s5)

Les places

nom **s2**
predecesseur s5->s2, s4->s2
successeur s2->s1

nom **s1**
predecesseur tr->s1, s2->s1
successeur s1->s4, s1->tr

nom **s4**
predecesseur s1->s4
successeur s4->s3, s4->s2

nom **lv**
predecesseur s8->lv, lavage
successeur lv->s8, lavage

nom **s8**
predecesseur lv->s8, s7->s8
successeur s8->s9, s8->lv

nom **m**
predecesseur s9->m, mesures
successeur m->s9, mesures

nom **s9**
predecesseur m->s9, s8->s9
successeur s9->s10, s9->m

nom **s**
predecesseur s10->s
successeur

nom **e**
predecesseur
successeur e->s5, e->bd

nom **cu**
predecesseur bd->cu, fraisage
successeur cu->bd, fraisage

nom **s3**
predecesseur s5->s3, s4->s3
successeur s3->cr, s3->s5

nom **s7**
predecesseur s6->s7, cr->s7
successeur s7->s6, s7->s8

nom **cr**
predecesseur s6->cr, s3->cr
successeur cr->s7, cr->s5

nom **s10**
predecesseur s9->s10
successeur s10->s6, s10->s

nom **s6**
predecesseur s10->s6, s7->s6
successeur s6->s7, s6->cr
nom **tr**
predecesseur s1->tr, tournage, retournement
successeur retournement, tournage, tr->s1
nom **bd**
predecesseur cu->bd, e->bd, s5->bd
successeur bd->cu, bd->s5
nom **s5**
predecesseur e->s5, bd->s5, cr->s5, s3->s5
successeur s5->bd, s5->s3, s5->s2

Fichier résultat de la phase d'analyse :

Listes de places.

Liste des places du prégraphe
places = (S2 S1 S4 LV S8 M S9 S E CU S3 S7 CR S10 S6 TR BD S5)

Liste des places ayant une boucle
l-pl-boucle = (TR CU M LV)

Liste des places n'appartenant à aucun circuit
l-pl-cir-bar = (E S)

Liste des places appartenant à au moins un circuit
l-pl-cir = (S2 S1 S4 LV S8 M S9 CU S3 S7 CR S10 S6 TR BD S5)

Liste des places appartenant à un aller-retour mais à aucun circuit pur
l-pl-cp-ar-bar = (BD CU M LV TR)

Liste des places composant les circuits purs indépendants
l-pl-cir-pur-ind = (S8 S9 S10 S6 CR S7 S2 S1 S4 S3 S5)

Listes de circuits.

Liste des circuits
l-cir = (C-1 C-2 C-3 C-4 C-5 C-6 C-7 C-8 C-9 C-10 C-11 C-12 C-13 C-14)
C-1 = (BD S5)
C-2 = (S7 S6)
C-3 = (S7 S6 CR)
C-4 = (S3 S5)
C-5 = (S3 CR S5)
C-6 = (CU BD)
C-7 = (M S9)
C-8 = (S8 S9 S10 S6 S7)
C-9 = (S8 S9 S10 S6 CR S7)
C-10 = (LV S8)
C-11 = (S1 TR)
C-12 = (S2 S1 S4)
C-13 = (S2 S1 S4 S3 S5)
C-14 = (S2 S1 S4 S3 CR S5)

Liste des allers-retours
l-all-ret = (C-1 C-2 C-4 C-6 C-7 C-10 C-11)

Liste des allers-retours dont aucune des places appartient à un circuit pur
l-all-ret-ty0 = (C-6)

Liste des allers-retours dont une place appartient à un circuit pur
l-all-ret-ty1 = (C-1 C-7 C-10 C-11)

Liste des allers-retours dont les deux places appartiennent à un circuit pur
l-all-ret-ty2 = (C-2 C-4)

Liste des circuits purs
l-cir-pur = (C-3 C-5 C-8 C-9 C-12 C-13 C-14)

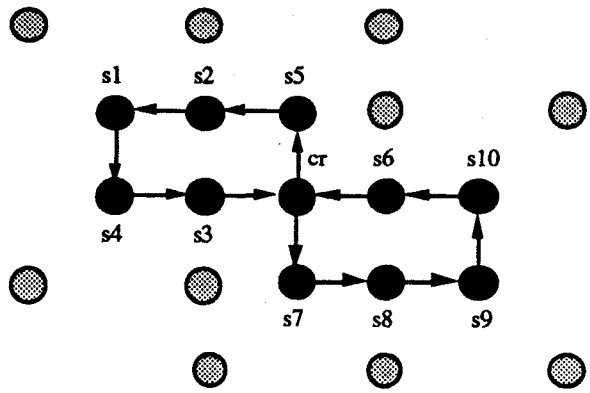
Liste des circuits purs indépendants
l-cir-pur-ind = (C-9 C-14)

Liste des circuits purs non indépendants
l-cir-pur-ind-bar = (C-3 C-5 C-8 C-12 C-13)

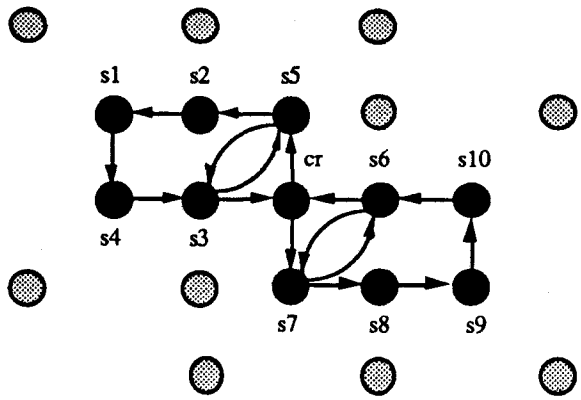
Liste des familles
l-famille = (fam-1)
fam-1 = (C-9 C-14)

Etapes de positionnement automatique :

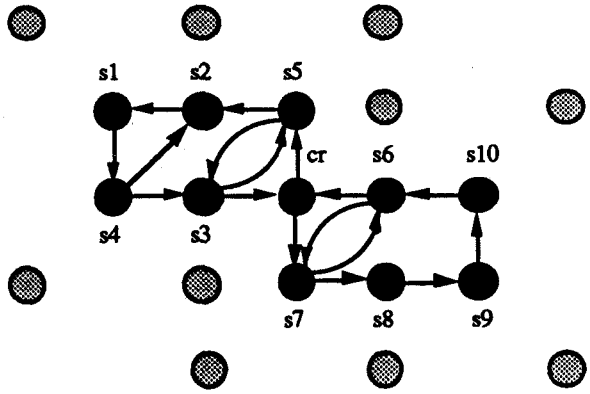
- Positionnement des circuits purs indépendants.



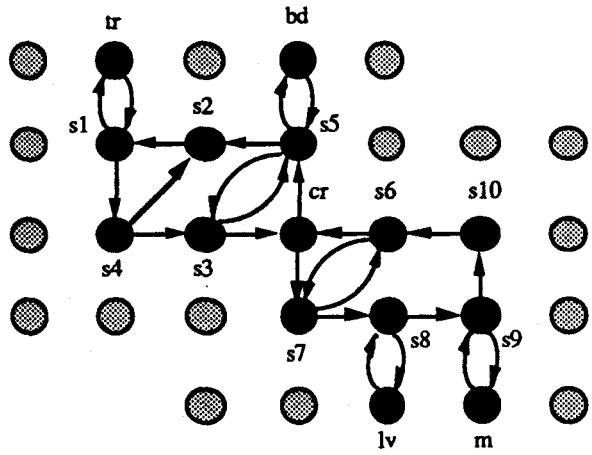
- Positionnement des aller-retours de type 2.



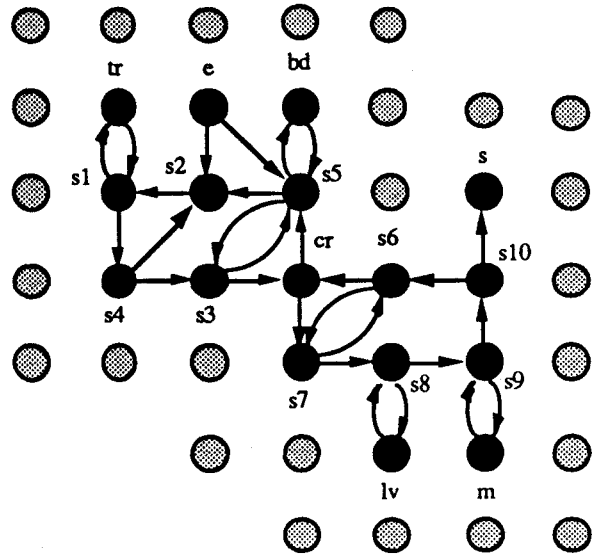
- Positionnement des circuits purs non indépendants.



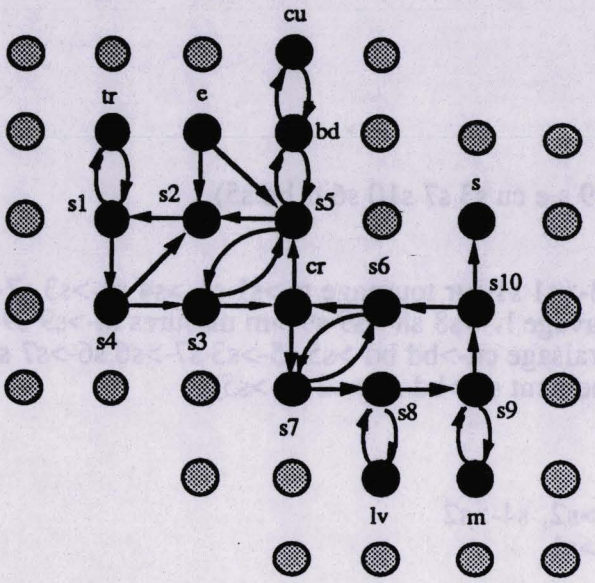
- Positionnement des aller-retours de type 1.



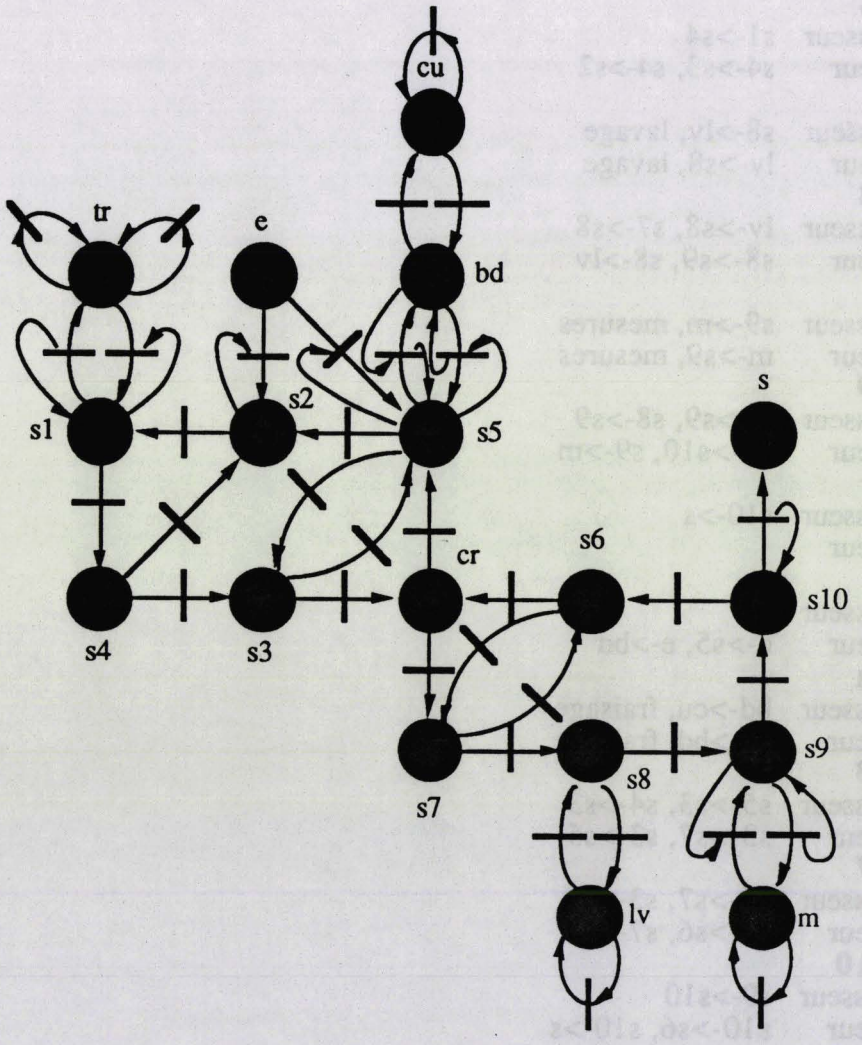
- Positionnement des places n'appartenant à aucun circuit.



- Positionnement des aller-retours de type 0.



Représentation finale du prégraphe :



III.7. Deuxième exemple.

Fichier d'entrée :

Liste des places
(s2 s1 s4 lv s8 m s9 s e cu s3 s7 s10 s6 tr bd s5)

Liste des transitions
(e->s5 s5->s2 s2->s1 s1->tr tournage tr->s1 s1->s4 s4->s3 s7->s8 s3->s7 s3->s7 s8->lv lavage lv->s8 s8->s9 s9->m mesures m->s9 s9->s10 s10->s e->bd bd->cu fraisage cu->bd bd->s5 s5->s3 s7->s6 s6->s7 s6->s5 s6->s5 s10->s6 retournement s5->bd s4->s2 s3->s5)

Les places
nom **s2**
predecesseur s5->s2, s4->s2
successeur s2->s1
nom **s1**
predecesseur tr->s1, s2->s1
successeur s1->s4, s1->tr
nom **s4**
predecesseur s1->s4
successeur s4->s3, s4->s2
nom **lv**
predecesseur s8->lv, lavage
successeur lv->s8, lavage
nom **s8**
predecesseur lv->s8, s7->s8
successeur s8->s9, s8->lv
nom **m**
predecesseur s9->m, mesures
successeur m->s9, mesures
nom **s9**
predecesseur m->s9, s8->s9
successeur s9->s10, s9->m
nom **s**
predecesseur s10->s
successeur
nom **e**
predecesseur
successeur e->s5, e->bd
nom **cu**
predecesseur bd->cu, fraisage
successeur cu->bd, fraisage
nom **s3**
predecesseur s5->s3, s4->s3
successeur s3->s7, s3->s5
nom **s7**
predecesseur s6->s7, s3->s7
successeur s7->s6, s7->s8
nom **s10**
predecesseur s9->s10
successeur s10->s6, s10->s

nom **s6**
predecesseur s10->s6, s7->s6
successeur s6->s7, s6->s5
nom **tr**
predecesseur s1->tr, tournage, retournement
successeur retournement, tournage, tr->s1
nom **bd**
predecesseur cu->bd, e->bd, s5->bd
successeur bd->cu, bd->s5
nom **s5**
predecesseur e->s5, bd->s5, s6->s5, s3->s5
successeur s5->bd, s5->s3, s5->s2

Fichier résultat de la phase d'analyse :

Listes de places.

Liste des places du prégraphe
places = (S2 S1 S4 LV S8 M S9 S E CU S3 S7 S10 S6 TR BD S5)

Liste des places ayant une boucle
l-pl-boucle = (TR CU M LV)

Liste des places n'appartenant à aucun circuit
l-pl-cir-bar = (E S)

Liste des places appartenant à au moins un circuit
l-pl-cir = (S2 S1 S4 LV S8 M S9 CU S3 S7 S10 S6 TR BD S5)

Liste des places appartenant à un aller-retour mais à aucun circuit pur
l-pl-cp-ar-bar = (BD CU M LV TR)

Liste des places composant les circuits purs indépendants
l-pl-cir-pur-ind = (S2 S1 S4 S3 S7 S8 S9 S10 S6 S5)

Listes de circuits.

Liste des circuits
l-cir = (C-1 C-2 C-3 C-4 C-5 C-6 C-7 C-8 C-9 C-10 C-11 C-12 C-13 C-14)
C-1 = (BD S5)
C-2 = (S7 S6)
C-3 = (S3 S5)
C-4 = (S3 S7 S6 S5)
C-5 = (CU BD)
C-6 = (M S9)
C-7 = (S8 S9 S10 S6 S7)
C-8 = (S8 S9 S10 S6 S5 S3 S7)
C-9 = (LV S8)
C-10 = (S1 TR)
C-11 = (S2 S1 S4)
C-12 = (S2 S1 S4 S3 S5)
C-13 = (S2 S1 S4 S3 S7 S8 S9 S10 S6 S5)
C-14 = (S2 S1 S4 S3 S7 S6 S5)

Liste des allers-retours
l-all-ret = (C-1 C-2 C-3 C-5 C-6 C-9 C-10)

Liste des allers-retours dont aucune des places appartient à un circuit pur
l-all-ret-ty0 = (C-5)

Liste des allers-retours dont une place appartient à un circuit pur
l-all-ret-ty1 = (C-1 C-6 C-9 C-10)

Liste des allers-retours dont les deux places appartiennent à un circuit pur
l-all-ret-ty2 = (C-2 C-3)

Liste des circuits purs
l-cir-pur = (C-4 C-7 C-8 C-11 C-12 C-13 C-14)

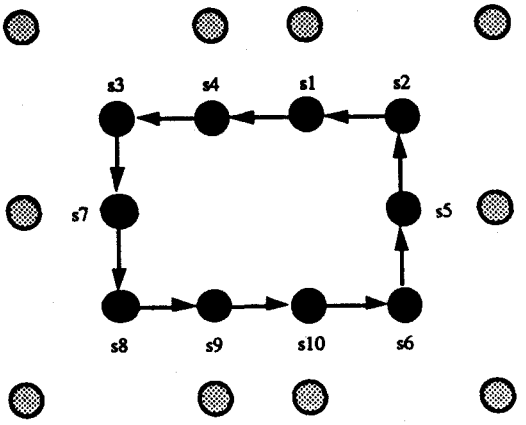
Liste des circuits purs indépendants
l-cir-pur-ind = (C-13)

Liste des circuits purs non indépendants
l-cir-pur-ind-bar = (C-3 C-5 C-8 C-12 C-13)

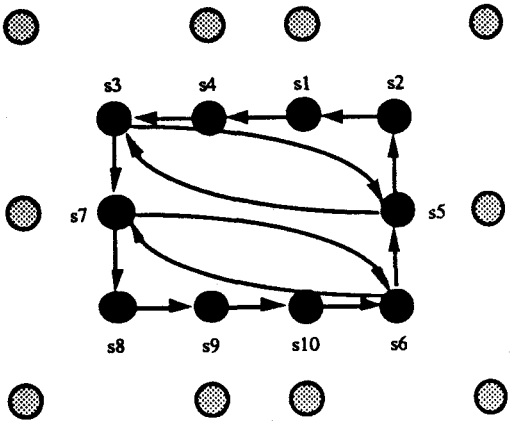
Liste des familles
l-famille = ()

Etapes de positionnement automatique :

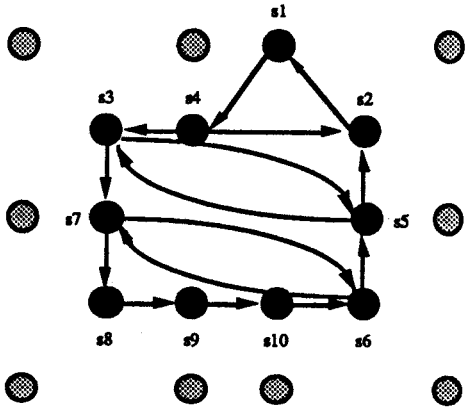
- Positionnement du circuit pur indépendant.



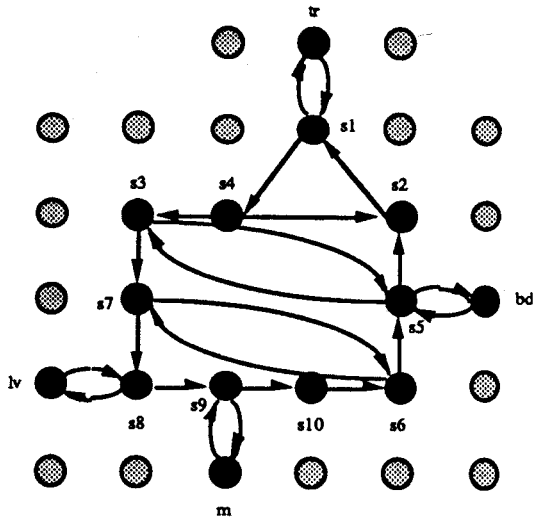
- Positionnement des aller-retours de type 2.



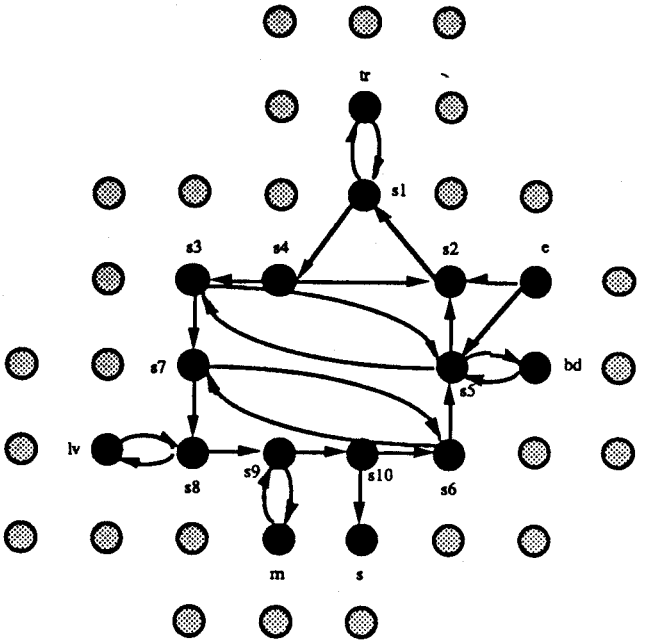
- Positionnement des circuits purs non indépendants.



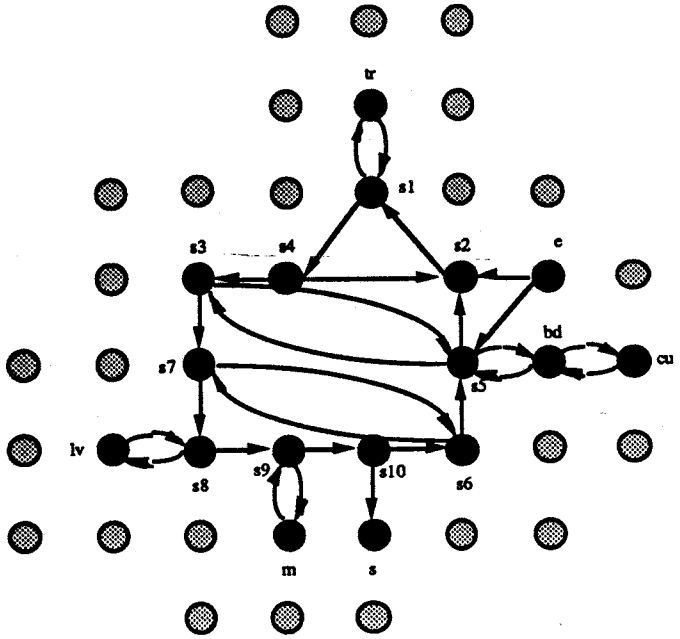
- Positionnement des aller-retours de type 1.



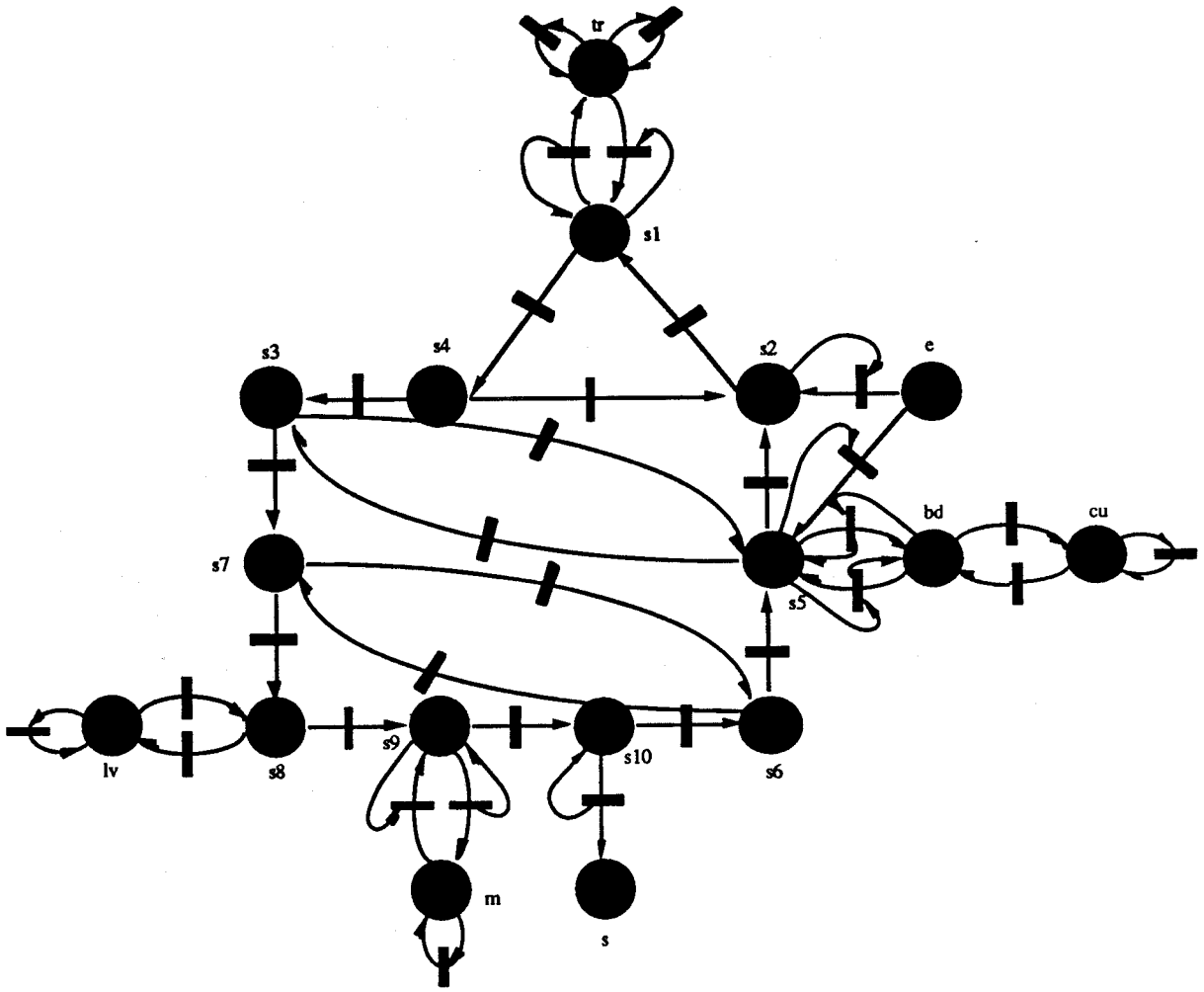
- Positionnement des places n'appartenant à aucun circuit.



- Positionnement des aller-retours de type 0.



Représentation finale du prégraphe :



III.8. Tableau du calcul du voisinage des 25 places d'un circuit pour la maille hexagonale.

	0	1	2	3	4	5
1	2	3	5	7	6	4
2	10	8	3	1	4	9
3	8	12	11	5	1	2
4	9	2	1	6	13	14
5	3	11	16	15	7	1
6	4	1	-7	18	17	13
7	1	5	15	19	-18	-6
8	20	21	12	3	2	10
9	23	-10	2	4	14	22
10	24	20	8	2	-9	-23
11	-12		25	16	5	3
12	21			-11	3	8
13	-14	4	6	17		
14	22	9	4	-13		
15	5	-16			19	7
16	11	25			-15	5
17	13	6	18			
18	6	-7	-19			17
19	7	15				-18
20			-21	8	10	24
21				12	8	-20
22		23	9	14		
23		-24	-10	9	22	
24			20	10	-23	
25					16	11

BIBLIOTHEQUE INTERUNIVERSITAIRE SECTION DE SCIENCES LILLE

Résumé.

La conception d'une cellule de production flexible discontinue consiste à parallèle, la conception d'un modèle réseau de Petri orienté produit, et d'un modèle orienté moyens de production qui permet ensuite de mettre en évidence les conflits concernant l'utilisation des ressources.

La stratégie que nous proposons dans ce mémoire pour résoudre les conflits concernant l'utilisation des ressources, consiste à les résoudre en temps réel, soit au niveau ordonnancement prévisionnel (cas des machines), soit au niveau pilotage pour les conflits concernant des opérations très courtes (cas des moyens de transport). Pour cela, nous avons développé un algorithme de calcul dynamique des coûts afin d'optimiser les différents itinéraires.

Afin de permettre la visualisation de ces différents chemins, et d'apporter une aide pour les phases de description, d'analyse, et de simulation du projet CASPAIM, nous avons développé une interface graphique permettant de représenter les différents modèles réseaux de Petri utilisés. Pour pouvoir représenter automatiquement les différents modèles, nous proposons une méthode de positionnement automatique des places d'un réseau de Petri, fondée sur la notion de circuits.

Un exemple de dimension industrielle illustre notre approche.

BIBLIOTHEQUE INTERUNIVERSITAIRE
SECTION DE SCIENCES
LILLE

Mots clés.

Méthodologie de conception
Systèmes de production flexibles
Réseaux de Petri
Résolution d'indéterminismes
Routage
Représentation graphique
Positionnement automatique de places