N° d'ordre : 1388

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

BERTRAND HUVENOIT
Maître EEA, Mastère IDN

DE LA CONCEPTION A L'IMPLANTATION DE LA COMMANDE MODULAIRE ET HIERARCHISEE DE SYSTEMES FLEXIBLES DE PRODUCTION MANUFACTURIERE

Soutenue le 25 octobre 1994 devant la Commission d'Examen :

M.	STAROSWIECKI M.	Examinateur, Président
M.	BOCQUET J.C.	Rapporteur
M.	LESAGE J.J.	Rapporteur
M.	PALUDETTO M.	Examinateur
M.	BIGAND M.	Examinateur
M..	GENTINA J.C	Examinateur, Directeur de Thèse
M.	BOUREY J.P.	Examinateur, Directeur de travail
M.	CRAYE E.	Examinateur, Directeur de travail

Cette thèse a été préparée au Laboratoire d'Automatique et d'Informatique Industrielle de Lille, URA - CNRS D 1440, Ecole Centrale de Lille.

REMERCIEMENTS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL) à l'Ecole Centrale de Lille sous la direction scientifique de Monsieur le Professeur J.C. GENTINA, Directeur de l'Ecole Centrale de Lille. Je tiens à le remercier vivement pour son accueil ainsi que pour la confiance et les précieux conseils qu'il ma apporté tout au long de ce travail.

Je suis très reconnaissant à Monsieur J.J. LESAGE, Professeur à l'Ecole Normale Supérieure de Cachan et membre du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA) et à Monsieur J.C. BOCQUET, Professeur à l'Ecole Centrale de Paris et membre du Laboratoire de Logistique et de Productique pour l'honneur qu'ils me font en acceptant d'examiner ce travail et d'être les rapporteurs de cette thèse.

Je suis flatté de la présence de Monsieur M. STAROSWIECKI, Professeur à l'Université des Sciences et Technologie de Lille (USTL), de Monsieur M. PALUDETTO, Maître de Conférence et membre du Laboratoire d'Automatique et d'Analyse des Système de Toulouse (LAAS) et de Monsieur M. BIGAND, Professeur Agrégé à l'Ecole Centrale de Lille et les remercie vivement d'avoir accepté d'être examinateurs de mes travaux.

Qu'il me soit permis de remercier tous les membres du LAIL. Je voudrais adresser une pensée toute particulière à mes anciens et nouveaux collègues de bureau, Messieurs S. AMAR, S. BOIS, A. FARAH, S. HAMMADI, L. KERMAD, J.P. MAIK, R. TAWEGOUM pour leur bonne humeur, leurs conseils, leur esprit d'équipe et qui ont contribué directement ou indirectement à l'aboutissement de ces travaux.

De même, je tiens à remercier tout particulièrement Messieurs J.P. BOUREY et E. CRAYE, respectivement Professeur et Maître de Conférences à l'Ecole Centrale de Lille, de m'avoir fait profiter de leur expérience. Je leurs suis très reconnaissant pour les remarques pertinentes et les conseils qu'ils ont su m'apporter.

Enfin, je remercie très sincèrement Monsieur M. VANGREVENINGE pour la reprographie de ce mémoire.

SOMMAIRE

SOMMAIRE

Pages

INTRODUCTION

Pour faire face à la concurrence qui est désormais mondiale, pour être créatrice d'emplois et de richesses, les entreprises doivent s'adapter aux technologies actuelles et aux besoins du marché où le consommateur est de plus en plus exigeant. Ce dernier recherche des produits qui sont à la fois de qualité, novateurs, personnalisés et à bas prix.

Pour être réactive à ces évolutions, une entreprise fabricant des produits en petite, moyenne ou grande série se doit de disposer d'un outil de production suffisamment automatisé.

Mais l'automatisation d'un système de production est un objectif ambitieux et onéreux étant donné la multitude de paramètres intervenant. En effet, un système de production est un système complexe où circule un ensemble de flux de natures diverses (Figure 1).

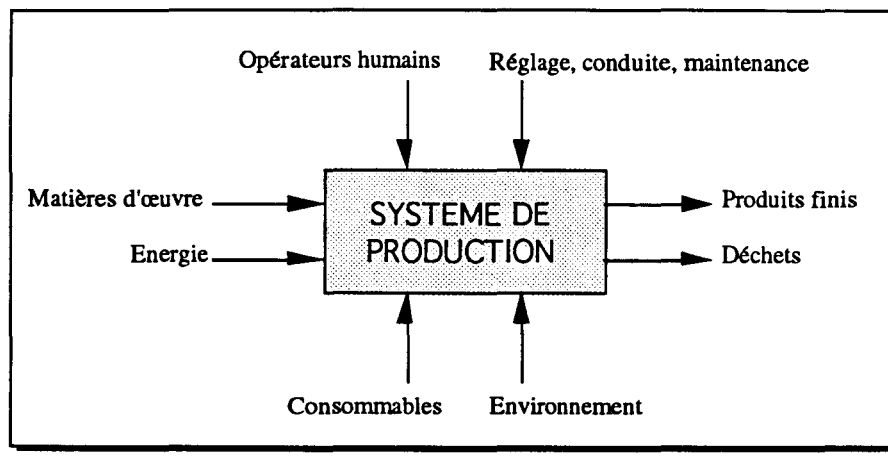


Figure 1 : Flux entrant et sortant dans un système de production

La nature du système de production est fortement liée aux caractéristiques des produits fabriqués [Rakotoson 1993]. Ainsi, nous pouvons distinguer :

- les Systèmes de production à Etats Continus (SEC),
- les Systèmes de production à Evénement Discrets (SED),
- les Systèmes de production Mixtes (Hybrides),
- les Systèmes de production à traitement par lots (Batch).

Dans les Systèmes de production à Etats Continus, le produit est un flux continu de matière (câblerie, chimie,...) ou d'énergie (production d'électricité).

Dans les Systèmes de production à Événements Discrets, les produits sont composés de matières solides (mis à part les lubrifiants, les colles,...) et ont une succession finie d'états. De fait, chaque produit est une entité individuelle que l'on peut facilement référencer et

facilement isoler. Nous trouvons principalement ces systèmes de production dans les industries mécaniques, électriques, électroniques,...

Lorsque le processus est partiellement continu et donc partiellement discontinu (processus dis-continus), la nature du produit étant bien définie pour chacune des phases du processus de fabrication, nous parlons de Système de production Mixte ou encore de Systèmes de production Hybride.

Enfin, lorsque la gestion du processus peut relever d'outils destinés au domaine continu ou discret suivant le niveau d'abstraction et le point de vue qui est choisi, nous sommes en présence d'un Système de production à Traitement par Lots, plus couramment désignés sous l'appellation de systèmes "Batch" (Agro-alimentaire, chimie,...).

Pour notre part, nous nous intéresserons uniquement aux **Systèmes de production à Evénements Discrets**.

Parmi l'ensemble de ces systèmes, il est possible de distinguer plusieurs styles d'organisation : les **machines spécialisées**, les **lignes transferts**, les **ateliers flexibles**,... La mise en place d'une solution ou d'une autre ne se fait pas sans un **compromis entre flexibilité et productivité**. En effet, une ligne transfert est beaucoup plus productive mais nettement moins flexible qu'un atelier flexible (Figure 2).

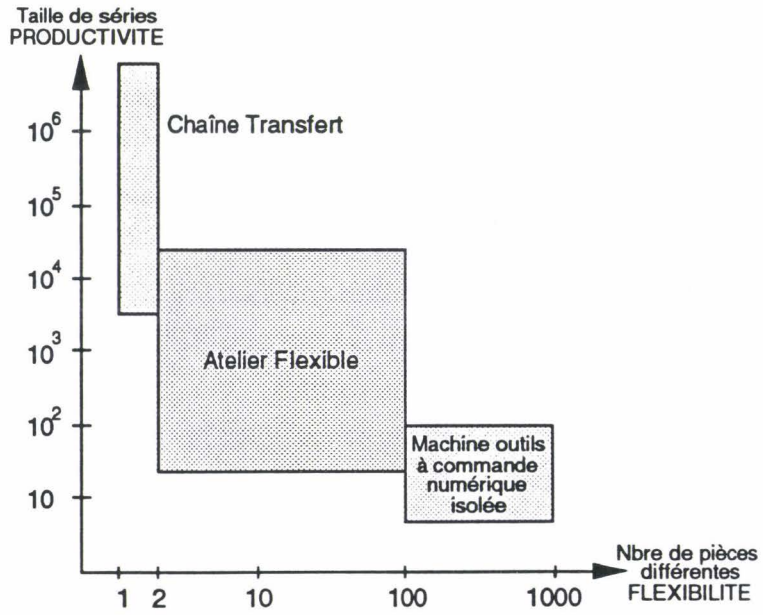


Figure 2 : Compromis entre productivité et flexibilité [Barbier et al 1992]

Le choix d'une organisation dépend également du type de production devant être mise en œuvre [Carlier et al 1988]. Ainsi, si les opérations élémentaires sur les produits sont indépendantes, nous avons une production de type "*Open-Shop*". Par contre, si les opérations élémentaires des produits sont liées par le même ordre total, nous avons une production de type "***Flow-Shop***". Enfin, si les opérations élémentaires sont liées par un ordre total mais pouvant différer d'un produit à un autre, nous avons une production de type "*Job-Shop*".

Pour notre part, nous nous intéresserons plus particulièrement aux productions de type "Job-Shop". Ce type de production est couramment réalisé à l'aide d'usines, d'ateliers ou de cellules flexibles. Ces systèmes sont désignés sous le vocable "**Systèmes Flexibles de Production Manufacturière** (SFPM)".

De tels systèmes ont une dynamique, un degré de parallélisme et une combinatoire élevés. Leur modélisation est basée sur des modèles à états discrets/transitions tels que les réseaux de Petri, le Grafcet, les automates à états finis, les réseaux à files d'attente... complétés par des modèles organisationnels tels les règles de production, les diagrammes SADT (Structured Analysis and Design Techniques), les diagrammes de JACKSON,... ou plus simplement par des algorithmes de décision [Calvez 1990].

Les Systèmes Flexibles de Production Manufacturière assurent la réalisation simultanée de plusieurs types de produits et peuvent être dissociés en trois composantes principales [Bonetto 1987] :

- une composante pour la fabrication,
- une composante pour la manutention,
- une composante pour le montage.

Les possibilités du SFPM et donc son degré de flexibilité dépendent directement de la complémentarité et de la coopération de ces diverses composantes. Par exemple, la présence de ressources de production complémentaires, la présence de plusieurs ressources de production de même type, la possibilité d'emprunter des chemins différents pour aller d'un point à un autre,... permettent d'augmenter la flexibilité mais aussi d'améliorer la réponse aux défaillances.

Pour l'industriel, un système flexible est un atout mais il engendre un lourd investissement lié à une complexité forte. Pour étaler celui-ci, une approche modulaire est nécessaire.

Une fois le SFPM bien spécifié, il faut penser à son automatisation. Encore de nos jour,

celle-ci est menée à l'aide de méthodes heuristiques, non universelles qui coûtent cher à l'entreprise d'autant plus que cette automatisation est souvent d'une maintenabilité réduite. C'est pourquoi, au sein de la communauté scientifique internationale, plusieurs équipes travaillent sur des méthodologies de conception des SFPM automatisés afin de remplacer l'empirisme technologique par un savoir faire formel à même de transmettre l'expérience acquise [Morel 1992].

Dans ce domaine, il existe trois axes de recherche :

- celui visant à développer une méthodologie de conception des systèmes d'information. C'est dans ce cadre que le projet PTA (Poste de Travail de l'Automaticien) a été mené par un ensemble de laboratoires, d'industriels et d'organismes français [Morel 1992].

- celui visant à développer une méthodologie de conception liée aux systèmes décisionnels. Dans ce sens, le laboratoire GRAI de Bordeaux, a développé une méthodologie pour l'analyse et la conception des systèmes de gestion de production [Doumeingts 1984].

- celui visant à développer une méthodologie de conception des systèmes opérationnels. Des laboratoires français comme le LAAS de Toulouse, le LAG de Grenoble, le LACN de Nancy, le LURPA de Cachan, le Lab de Besançon, le LAMM de Montpellier,... travaillent dans cet axe.

C'est dans ce dernier cadre de recherche que s'inscrit le projet **CASPAIM** (Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière) développé au LAIL (Laboratoire d'Automatique et d'Informatique Industrielle de Lille). L'objectif premier de ce projet est de trouver une organisation, une modélisation et une implantation "universelle".

Les travaux faisant l'objet de ce mémoire s'inscrivent dans le cadre du projet CASPAIM. Ils se situent plus particulièrement en aval du projet et sont axés sur la définition d'une commande opérationnelle facilement implantable. Dans ce sens, nous avons fait évoluer la modélisation existante [Cruette 1991]. Nous avons hiérarchisé et rendu modulaire la commande de coordination des produits et nous avons développé une méthode de construction de la commande des ressources de production. Dans les deux cas, nous avons tenté de dégager le maximum de généricité et de modularité afin de favoriser la réutilisabilité. Pour accueillir ces deux aspects de la commande d'un SFPM, nous avons développé une nouvelle structure de contrôle où les divers aspects décisionnels sont bien dissociés. Ensuite, nous avons tenté de développer une méthode permettant, à terme, d'automatiser en grande partie la génération du logiciel de simulation permettant de valider la dynamique de la commande et du logiciel de contrôle/commande directement exploitable sur site.

Ainsi, ce mémoire se décompose en trois chapitres.

Dans le **premier chapitre**, nous présenterons les tenants et les aboutissants de l'automatisation des systèmes flexibles de production manufacturière. Ensuite, nous présenterons les projets développés au LAIL liés à ce type d'automatisation : CASPAIM I et CASPAIM II.

Dans le **second chapitre**, après avoir bien spécifié les hypothèses de travail, nous aborderons la modélisation de la commande d'un SFPM en illustrant notre démarche sur un exemple de cellule flexible. Nous commencerons par une analyse du procédé, nous nous intéresserons ensuite à la commande de coordination des produits puis à la commande des ressources de production. Enfin nous aurons un aperçu sur les aspects décisionnels présents à plusieurs niveaux de la commande. Les modélisations entreprises feront largement appel aux Réseaux de Petri et aux concepts objets.

Enfin, dans le **troisième chapitre**, après avoir défini l'environnement informatique et après quelques rappels sur le langage d'implantation utilisé (ADA), nous décrirons une méthode de génération systématique et rigoureuse du logiciel de contrôle/commande totalement basée sur les modèles définis au chapitre II. Nous montrerons combien il est facile d'obtenir à la fois une version de ce logiciel pour une simulation permettant la validation du comportement dynamique de la commande et une version pour l'implantation répartie sur les organes informatiques du site de production.

CHAPITRE I

Contexte et problématique de l'étude

SOMMAIRE DU CHAPITRE I

I.A. Automatisation des SFPM

I.A.1. Définition et limitations

Automatiser un système flexible de production manufacturière consiste à mettre en place un **système informatique** - au sens large - **distribué et/ou réparti** permettant de le contrôler. Le système de production est alors constitué d'une partie commande qui contrôle la partie procédé par l'intermédiaire de commandes envoyées aux actionneurs et connaît l'état de celui-ci grâce aux compte-rendus retournés par les capteurs. Cet ensemble forme alors un système complexe ou apparaissent trois composantes : le système décisionnel, le système informationnel et le système opérationnel (modèle O.I.D. [Le Moigne 1990]) (Figure I.1).

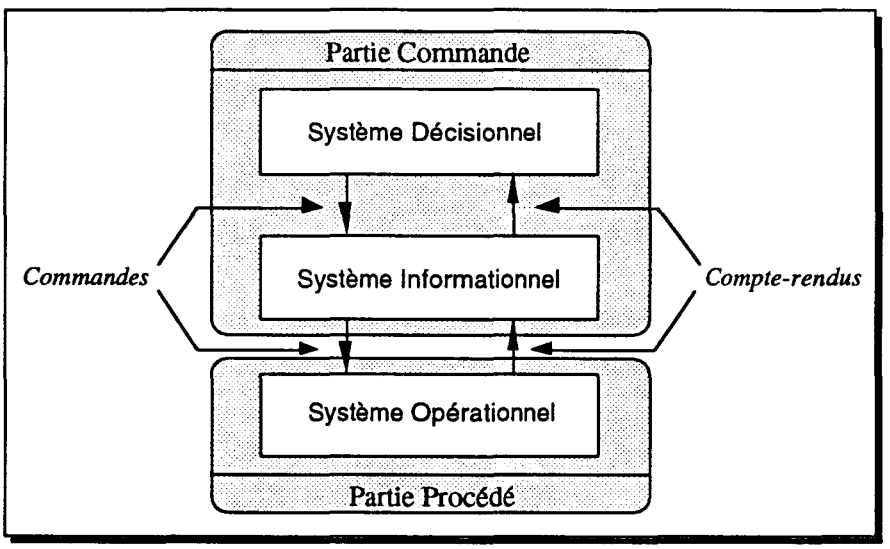


Figure I.1 : Décomposition systémique d'un SFPM automatisé

Une automatisation bien pensée et évolutive permet de renforcer la flexibilité du système opérant et permet d'accepter les évolutions futures du système de production. Elle doit aussi être garante d'une gestion plus rationnelle des ressources de production (répartition des charges), d'une amélioration de la qualité et d'une gestion plus rigoureuse des stocks (diminution des en-cours).

La démarche d'automatisation d'un système de production est difficile à mener. En effet, elle doit tenir compte de nombreux paramètres économiques, sociaux, technologiques,... Le plus souvent, une automatisation totale est coûteuse, ambitieuse et peut se révéler inadaptée. Pour certaines opérations délicates, la main d'œuvre humaine est et restera pour longtemps irremplaçable.

I.A.2. Objectif d'un SFPM automatisé

La course à la productivité, les variations et évolutions sur les produits, la recherche d'une meilleure qualité et d'une meilleure gestion entraînent une automatisation de plus en plus poussée des systèmes de production. Or pour être efficace et donc **compétitif**, tout Système Flexible de Production Manufacturière doit [Engineers 1988] [Rancky 1990 a] :

- répondre à un objectif de production,
- répondre à un objectif de qualité,
- répondre à un objectif de coût de production,
- être facilement maintenable,
- être rentable à moyen terme,
- répondre à la dynamique d'évolution du marché,
- être intégré à l'entreprise.

Les trois premiers points sont couverts par la **gestion de production** qui fournit à l'industriel un ensemble de méthodes et d'outils permettant de cerner et d'atteindre ses objectifs [Bénassy 1987].

Un **objectif de production** correspond à la définition des quantités de produit à fabriquer en un temps donné. Pour y répondre, il est nécessaire de mettre en œuvre un ensemble organisationnel composé de trois niveaux dépendant de l'horizon temporel d'étude [Lamy 1987] :

- la **planification** à moyen terme qui définit un plan de production sur la base des commandes clients et dont l'objectif consiste à assurer la répartition des dates de début de lancement en fabrication des produits,

- l'**ordonnancement** à court terme qui affecte les produits aux machines de production tout en définissant les marges temporelles de chaque opérations (Date de début au plus tôt, Date de fin au plus tard,...) afin de satisfaire le plan de production,

- le **pilotage** qui contrôle directement le processus de fabrication et qui tente d'assurer en temps réel (temps compatible avec les temps de cycles de la production) la réalisation du plan de production sur la base des contraintes calculées par l'ordonnancement.

Cet ensemble donne une hiérarchie décisionnelle qui doit former une boucle de régulation afin de répartir les charges de production mais aussi afin de tenir compte des imprévus ou des défaillances pouvant survenir (Figure I.2).

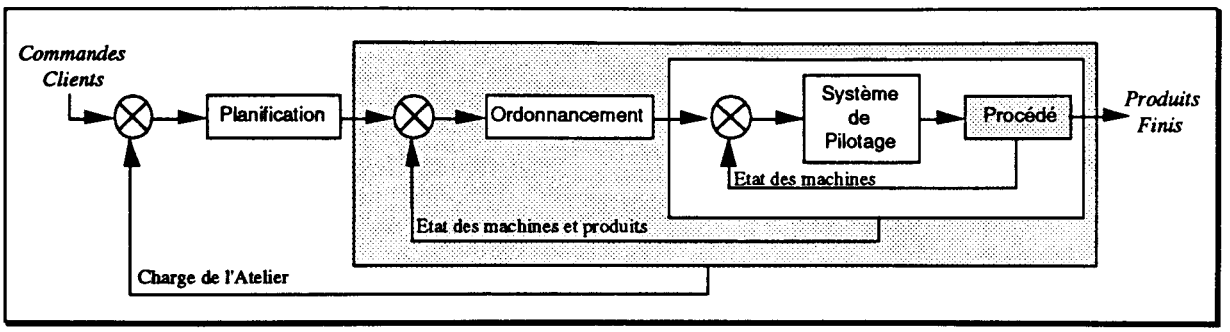


Figure I.2 : Organisation des différents niveaux décisionnels d'un système de production

D'une part, l'utilisateur final des produits manufacturés étant de plus en plus exigeant et d'autre part pour faire face à la concurrence, tout industriel se doit de rechercher une qualité maximale et donc de définir des objectifs de qualité pour sa production. Pour cela, les contrôles qualité doivent êtres intégrés au système de production (bancs de mesures, systèmes de vision,...). Mais la qualité ne se cantonne pas uniquement à la production, aujourd'hui c'est un concept d'entreprise qui s'applique aussi en amont de celle-ci. C'est ainsi que la définition d'une méthodologie de conception rigoureuse intervient directement dans la qualité globale de la production.

La structure du système de production influe fortement, à long terme, sur le coût de production des produits manufacturés. En effet, passer d'une production à une autre ou ajouter une nouvelle production peut s'avérer fort coûteux si le système de production est peu évolutif ou peu flexible. Ou encore, disposer de stocks ou d'encours trop importants est coûteux car ils représentent de la matière et donc des fonds immobilisés.

Enfin, un système de production doit être facilement maintenable dans le sens ou les défaillances doivent être rapidement détectées, localisées et réparées. De plus, la maintenance doit pouvoir se faire sans arrêter la totalité de la production.

I.A.3. Nécessité d'une automatisation intégrée et évolutive

Les entreprises ne peuvent être efficaces que si le passage de la conception des produits à la production de ceux-ci s'effectue sans déboires et sans perte de temps afin de réduire les coûts et les temps de réponse de l'entreprise par rapport à l'évolution du marché. Le processus de fabrication doit donc être pris dans sa globalité et intégrer l'ensemble des sous systèmes qui le composent : CAO, CFAO, GPAO, Systèmes Automatisés de Production,... L'intégration de ces différentes composantes permet alors de résoudre les problèmes liés au partage des données, à

la validité et à la cohérence des informations, d'améliorer la dynamique de l'entreprise,...

Dès sa conception, l'automatisation du système de production doit tenir compte de cette intégration. De plus, elle se doit d'être évolutive afin de faciliter le passage d'un produit à un autre et aussi afin de s'adapter aux modifications des méthodes de production ou à l'extension de l'outil de production.

I.A.4. Le concept CIM

Pour favoriser l'intégration des fonctions de l'entreprise, une nouvelle approche a été développée à partir du début des années 1980 : le concept **CIM (Computer Integrated Manufacturing)** [Rancky 1990 a]. La traduction donnée par l'ADEPA (Agence Nationale pour le Développement de la Productique Appliquée à l'Industrie) est **productique intégrée**.

Le CIM vise à intégrer, le plus possible, les divers sous systèmes et acteurs de l'entreprise par l'intermédiaire de l'informatique et par l'amélioration de la communicabilité grâce aux réseaux locaux. La pyramide CIM (Figure I.3) indique que cette intégration se propage des niveaux les plus bas (niveaux initialement intégrés) aux niveau les plus élevés de l'entreprise.

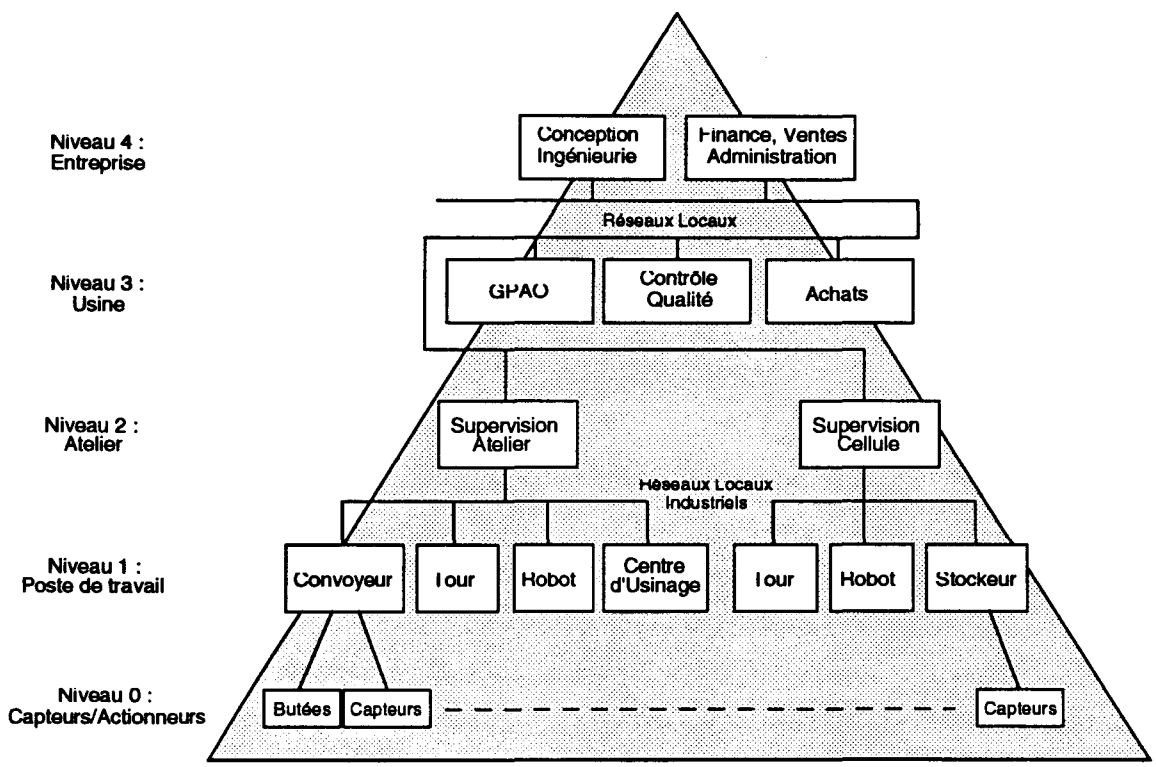


Figure I.3 : Pyramide CIM

Dans le CIM, l'entreprise est vue comme un ensemble de fonctions, d'informations et de composants (matériels et hommes) [Barbier et al 1992]. La finalité d'une méthodologie orientée CIM est d'établir un ensemble de dialogues efficaces entre toutes ces entités afin d'aboutir à une automatisation globale de l'entreprise. Pour atteindre cet objectif, le concept CIM préconise la mise en œuvre de réseaux locaux, de réseaux locaux industriels et de bases de données accessibles par l'ensemble des services de l'entreprise. De fait, le CIM permet d'améliorer les conditions de travail en valorisant l'opérateur humain.

La prise en compte du concept CIM par les entreprises n'en est qu'à ses débuts. De plus, l'objectif à atteindre n'est pas facile. En effet, actuellement, un grand nombre de domaines et de technologie hétérogènes entrent dans la mise en œuvre du concept.

I.A.5. Vers le Génie Automatique

Si l'on veut répondre aux objectifs de coût, d'intégration, de qualité et d'évolution de la commande des SFPM, leur conception doit être systématique, rigoureuse et donc être basée sur une méthodologie. Celle-ci doit fournir au producticien un ensemble de modèles, de méthodes, d'outils et de logiciels lui permettant de mener à bien et avec qualité l'automatisation de l'outil de production tout au long de son cycle de vie (cf. cycle en V de l'annexe I).

Cet ensemble d'outils-méthodes est intégré dans une discipline; le **Génie Automatique** dont la définition est la suivante :

"***On appelle génie automatique l'application de méthodes scientifiques au développement de théories, méthodes, techniques, langages et outils favorisant la production de systèmes automatisés de qualité.***" [Morel et al 1988]

Le génie automatique s'inspire fortement du génie logiciel [Jacobson et al 1993]. Ainsi, l'un de ses grands intérêts est qu'il favorise la capitalisation de l'expérience par la construction de composantes logicielles réutilisables. Ceci demande, dès le départ, un effort de spécification et de conceptualisation important mais permet, à terme, de produire efficacement et rapidement un système automatisé modulaire et de qualité.

Comme tout atelier, un atelier de génie automatique se doit d'assurer les fonctions de gestion de projet, de génération de code, d'assistance, d'évaluation, de gestion de bibliothèques de composantes réutilisables,...

I.B. Le projet CASPAIM I

I.B.1. Historique

Le projet **CASPAIM I (Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière)** est né de l'initiative du professeur J.C. Gentina au début des années 1980 au Laboratoire d'Automatique et d'Informatique industrielle de Lille (LAIL). L'objectif de ce projet était de prendre en charge la totalité des étapes du cycle de développement du système de commande d'un SFPM depuis la définition du cahier des charges jusqu'à l'implantation finale sur site industriel, en utilisant une approche homogène. L'équipe de recherche s'est alors orienté vers une large utilisation des Réseaux de Petri et de leurs extensions [Brams 1983] [Murata 1989].

Il est apparu que les techniques classiques pour l'analyse des propriétés des Réseaux de Petri et surtout que les méthodes analytiques pour l'étude des performances du système et de sa commande ne pouvaient être utilisées qu'au travers de simplifications des modèles qui ne permettent pas l'étude fine de certains comportements. A cette époque, la simulation a donc été l'unique méthode retenue pour la validation du système : analyse des performances, étude des blocages et indéterminismes, détermination des régimes transitoires et des modes de marche, ... Un simulateur de R.d.P. a donc été développé [Castelain 1987].

Lors de la phase de spécifications du simulateur, est apparue la nécessité de disposer d'une méthode et d'un outil permettant d'assister voire d'automatiser la construction du modèle complet de la commande à partir d'un graphe fonctionnel appelé **Prégraphe**. Une démarche de développement d'un modèle structuré simulable de la partie commande a alors été proposée [Bourey 1988]. Cette démarche a été complétée d'une part en amont, par une méthode et un outil permettant une génération assistée du prégraphe [Kapusta 1988] et d'autre part en aval par une méthode permettant l'implantation du modèle de la commande sur un réseau d'automates programmables industriels (A.P.I.) [Craye 1989]. Ces quatre méthodes et outils constituent la base du projet CASPAIM I présenté dans le paragraphe suivant [Bourey 1993].

I.B.2. Les modèles

Le projet CASPAIM I repose sur la décomposition d'un Système flexible de production en trois parties communicantes : le procédé, la partie commande et le niveau hiérarchique (figure I.4).

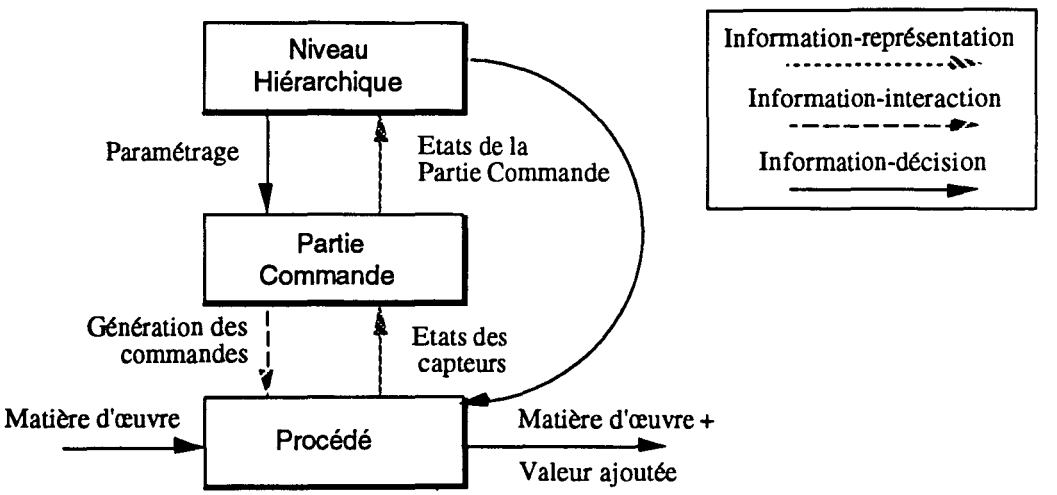


Figure I.4: Décomposition d'un S.F.P.M. dans CASPAIM I

- La **Partie Commande** (P.C.) a pour fonction de recevoir les signaux des capteurs et d'envoyer les commandes aux actionneurs en assurant la coordination et le séquencement des commandes applicables au procédé. Elle ne résout cependant pas les conflits et indéterminismes liés à la flexibilité du système de production.

- Le **Procédé** est constitué de la Partie Opérative (actionneurs et capteurs) et de l'ensemble des équipements et matériels intervenants dans le processus de fabrication.

- Le **Niveau Hiérarchique** (N.H.) est chargé de construire les décisions et choix destinés à résoudre les indéterminismes de la Partie Commande.

Cette description peut être comparée au modèle canonique "Opérant, Information, Décision" (O.I.D.) d'un Système Général Complexe dans le cadre d'une approche systémique [Tardieu et al 1983], [Le Moigne 1990] en considérant la correspondance suivante :

Niveau Hiérarchique ⟺ Système de Décision
Partie Commande ⟺ Système d'Information
Procédé ⟺ Système Opérant

Dans la suite de ce document, nous appellerons **Système de Commande** l'ensemble formé par le Niveau Hiérarchique et la Partie Commande.

Les différents modèles choisis à chaque niveau sont les suivants :

- Pour la **Partie Commande**, il s'agit de **R.d.P. Structurés Adaptatifs et Colorés** pour lesquels l'aspect coloré permet la représentation des différents types de pièces et l'aspect adaptatif permet l'interfaçage avec le niveau hiérarchique,

- Pour le **Niveau Hiérarchique**, il s'agit d'un modèle déclaratif de type **système expert** constitué d'un moteur d'inférences et de règles décisionnelles,

- Enfin pour le **Procédé**, deux niveaux de modélisation sont utilisés. Le premier repose sur l'utilisation de **temporisations intégrées** directement au niveau du graphe de commande. Ce type de modèle permet d'effectuer une première étude quantitative du comportement du système. Le second niveau de modélisation est basé sur une modélisation spécifique du procédé par une approche de type **langage objet**. Ce type de modèle devient indispensable dès que l'on désire faire une étude fine du comportement d'un système ou une simulation des défaillances et anomalies.

I.B.3. La démarche

Le projet CASPAIM I, qui est basé sur une démarche de type modélisation/évaluation se décompose en quatre phases principales (figure I.5):

1- Spécification du cahier des charges,
2- Modélisation des trois constituants du système,
3- Validation par simulation,
4- Implantation du code sur les systèmes de contrôle/commande.

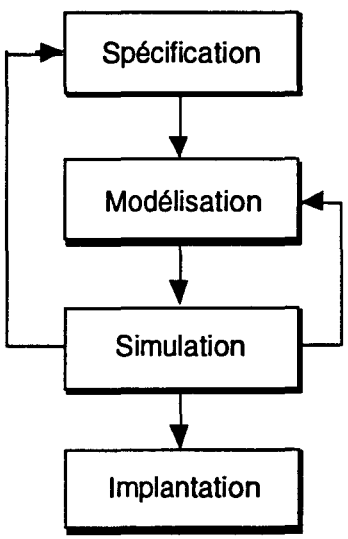


Figure I.5 : Démarche de conception CASPAIM I

L'objectif n'étant pas ici de présenter en détail l'ensemble de la méthodologie, nous limiterons la présentation aux grandes lignes de chaque étape sur l'exemple suivant.

Considérons une installation de production constituée :

✔ d'une zone d'entrée des pièces appelée **fifo1**,

✔ d'une machine **M** possédant un tampon d'entrée/sortie qui ne peut contenir qu'un seul objet et qui réalise les usinages de la cellule,

✔ d'une table d'assemblage **TA** où est assuré l'assemblage des pièces provenant de la machine ou directement de l'entrée (selon le type de pièce) avec des pièces provenant d'un stock appelé **fifo2**,

✔ d'une zone de sortie appelée **fifo-out**,

✔ d'un robot assurant l'assemblage et l'évacuation des pièces, appelé **RA**,

✔ d'un robot assurant les autres transferts, appelé **RT**.

Dans cette cellule, les opérations réalisées sont les suivantes :

✔ les pièces **p1** sont usinées pour donner les pièces **u**,

✔ sur les pièces **p** issues de **fifo2** sont assemblées indifféremment des pièces **u** ou des pièces **p2** provenant directement de l'entrée; l'assemblage des pièces **p** avec les pièces **u** (resp. **p2**) donne les pièces **s1** (resp. **s2**).

Sur la figure I.6 est représenté le schéma d'implantation de la cellule avec les différents flux de pièces.

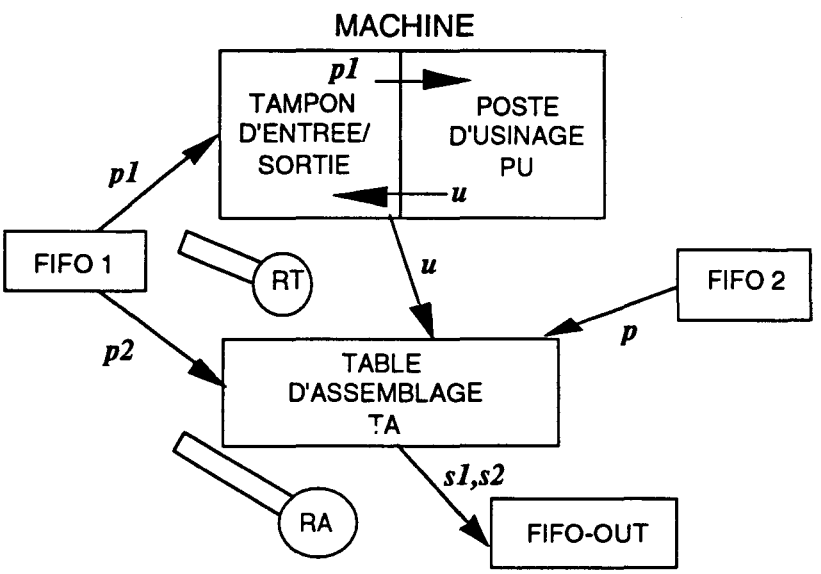


Figure I.6 : Schéma d'implantation de l'exemple;

➪ *Spécification du cahier des charges [Kapusta 1988]*

Cette phase vise à modéliser les gammes opératoires sous la forme d'un ensemble de règles de production traduisant les opérations élémentaires. Ces règles sont construites suivant le schéma suivant :

Prémisses => Conséquents

où Prémisses et Conséquents sont une conjonction de prédicats de la forme :

présence (objet, lieu).

Dans le cas de notre exemple, l'assemblage d'une pièce **p2** issue de **fifo1** sur une palette **p** sur **TA** se traduit par :

présence (**p2**, **fifo1**) et présence (**p**, **fifo2**) => présence (**s2**, **TA**).

Sur l'ensemble des règles est ensuite effectuée une analyse de complétude et de cohérence puis une génération du prégraphe pour lequel :

-une place modélise un emplacement physique de transit d'objets ou une zone sur laquelle est effectuée une opération,

-une transition représente une évolution que subit un objet physique. Cette évolution peut être une transformation physique et/ou un changement de lieu physique,

-une marque colorée représente un objet et son état d'avancement dans sa gamme.

Compte tenu des différentes opérations manufacturières de notre exemple, nous pouvons construire le prégraphe de l'architecture de commande de la cellule (figure I.7). Sur ce prégraphe, nous ne tenons pas encore compte du fait que les robots assurent plusieurs tâches exclusives ou que les capacités des différents lieux sont limitées.

En fait, le prégraphe permet de décrire le fonctionnement en mode pièce-à-pièce de l'installation. Il conviendra donc ultérieurement d'adjoindre la prise en compte des contraintes de coordination, de synchronisation et d'exclusion nécessaires à un parallélisme résultant de la relative indépendance des gammes.

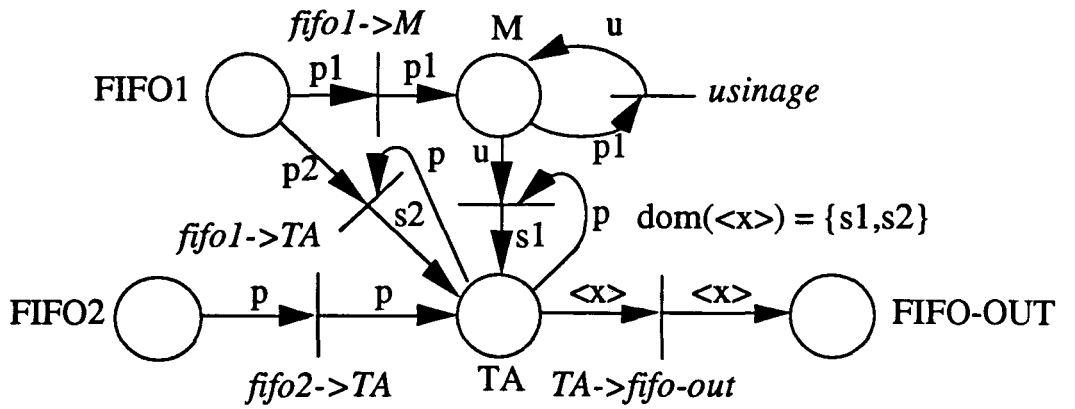


Figure I.7 : Prégraphe de la cellule de l'exemple

➪ ***Modélisation des trois constituants du système [Bourey 1988] :***

Cette phase a pour but d'assister la construction des trois modèles permettant l'analyse qualitative et quantitative du comportement dynamique du système.

Le graphe de commande est obtenu automatiquement en procédant à un ***développement*** et à la ***structuration*** du prégraphe à l'aide d'un certain nombre de primitives. Tel un système de compilation avec enrichissement, le développement s'effectue en structurant chaque constituant du prégraphe en un processus R.d.P.S.A.C. et en introduisant les liaisons nécessaires à la communication des processus.

L'originalité de la démarche se situe à trois niveaux :

(i) Le modèle global de la commande est obtenu à partir de la composition de primitives génériques et, en conséquence, il met en évidence les propriétés de ***modularité*** , d'***intelligibilité*** et d'***encapsulation*** qui sont essentielles dans toute méthode de conception visant la ***réutilisabilité*** et la ***maintenabilité*** des codes produits.

(ii) La méthode de développement utilisée pour chaque primitive génère un processus R.d.P qui possède de bonnes ***propriétés comportementales*** (vivacité, finitude et réinitialisabilité). Seuls certains types de liaisons entre les différents modules combinés à des séquences et fréquences particulières d'entrée de pièces peuvent engendrer des blocages. Il s'agit notamment des systèmes de transports bouclés à ressources limitées (convoyage par palettes, par exemple) pour lesquels il peut exister des cas de famine par manque de ressources ou des comportements limites de deadlock par saturation (verrous généralisés).

(iii) La démarche est ***systématique et complètement automatisable***. Ainsi, une maquette d'outil qui génère un modèle directement simulable a été réalisée.

Dans le cadre de notre exemple, nous déduisons directement l'ossature modulaire de la commande représentée par le schéma bloc de la figure I.8. sur lequel nous retrouvons cinq blocs de transfert (**FIFO1->M, M->TA, FIFO1->TA, FIFO2->TA, TA->FIFO-OUT**), trois blocs gérant des zones tampon (**FIFO1, FIFO2, FIFO-OUT**), un bloc gérant la machine **M** et un bloc gérant l'assemblage sur la station **TA**. Sur ce schéma qui montre bien la modularité du modèle développé, seuls les blocs de transfert et les zones tampons sont détaillés. Les blocs **M** et **TA** sont représentés sur les figures I.9 et I.10.

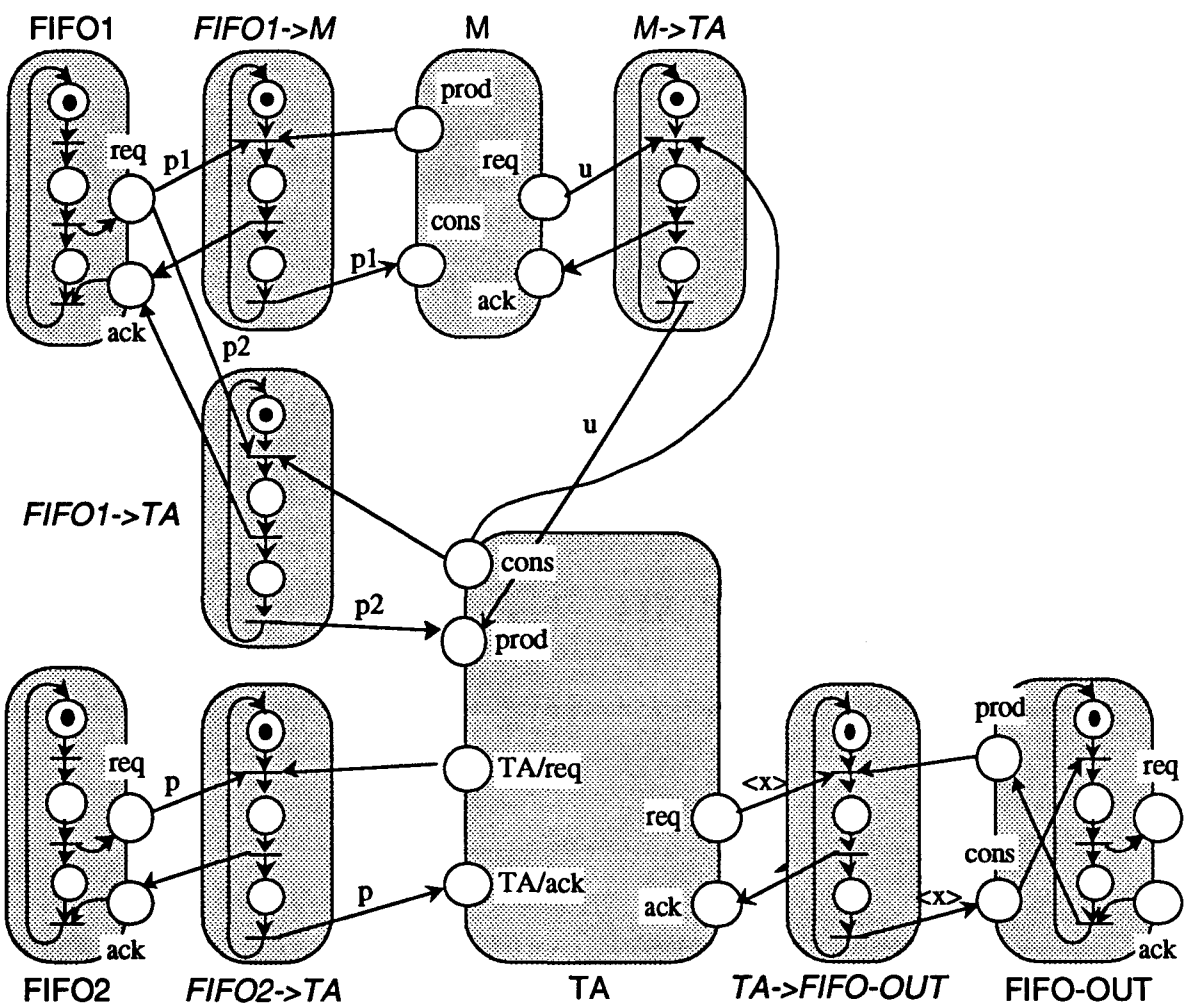


Figure I.8 : Schéma bloc de la commande de la cellule de l'exemple

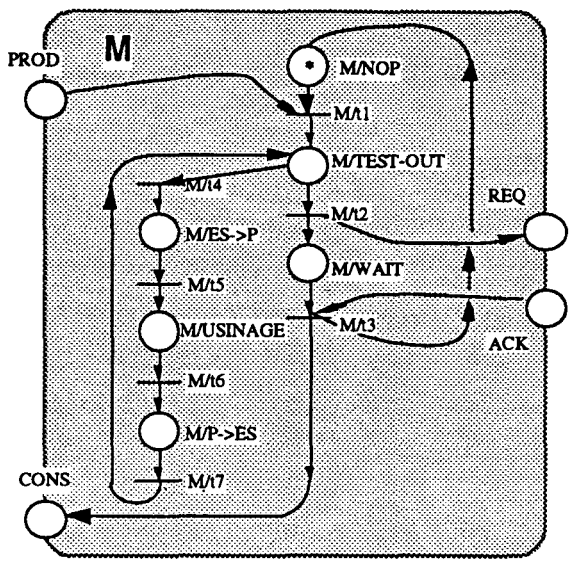


Figure I.9 : R.d.P.S.C. de la commande de la machine M

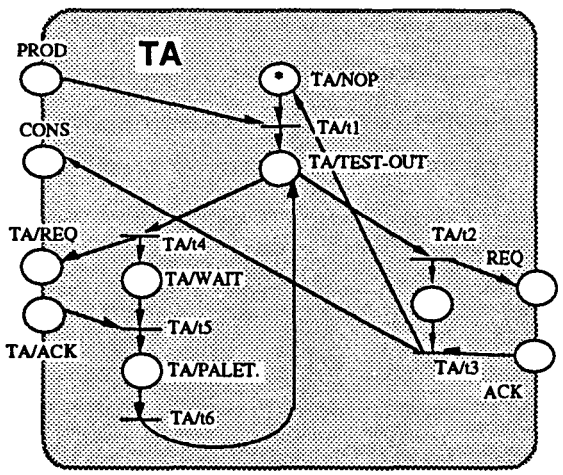


Figure I.10 : R.d.P.S.C. de la commande de l'assemblage; TA

Ces graphes de commande correspondent à un fonctionnement de l'installation avec un parallélisme maximal sur lequel il faut bien évidemment prendre en compte le fait que certaines opérations ne peuvent être réalisées simultanément. Ceci se traduit par l'ajout de liaisons de type exclusion mutuelle entre ces opérations comme, par exemple sur les transferts de **fifo1** vers la machine, de **fifo1** et **fifo2** vers la station d'assemblage et de la machine vers le poste d'assemblage (tous ces transferts sont assurés par le même robot **RT**).

Le modèle du niveau hiérarchique qui est constitué d'un ensemble de règles permettant de lever les indéterminismes du graphe de commande, est élaboré après une phase de détection automatique de tous les conflits structurels. Les cas de blocage seront résolus ultérieurement après détection lors des premières simulations.

Le modèle du procédé est ensuite construit et le dimensionnement des zones tampons est effectué ainsi que le choix des fréquences et séquences d'entrées.

➪ *Validation par simulation [Castelain 1987]*

Lors de la simulation, certains comportements peuvent amener le concepteur à effectuer des retours arrières lui permettant, par exemple, de redimensionner certaines zones, de modifier des priorités pour éviter des blocages, de synchroniser des opérations... afin de répondre progressivement aux contraintes et objectifs imposés au niveau du cahier des charges.

➪ *Implantation [Craye 1989]*

Lors de cette phase, il s'agit d'implanter les modèles obtenus (P.C. et N.H.) sur un ensemble d'organes de contrôle/commande et plus particulièrement réseaux d'automates et d'ordinateurs de supervision. Outre la traduction du R.d.P.S.A.C. de la commande en un langage supporté par les automates cibles, le problème est alors de définir des critères de décomposition judicieux permettant de tirer le meilleur parti d'une architecture de contrôle/commande existante ou de proposer une architecture optimale dans le cas d'une nouvelle installation.

I.B.4. Les limitations

La démarche proposée dans le cadre du projet CASPAIM I fut une des premières approches intégrées permettant de structurer et de maîtriser les critères de flexibilité. Elle s'est révélée satisfaisante pour des systèmes de production plutôt orientés usinage, peu complexes et peu sujets à évolution ultérieure.

Lors de plusieurs études [Cruette 1991] [Gloannec 1990], il s'est avéré que les processus de fabrication pour lesquels l'une au moins des conditions suivantes est vérifiée, posaient des difficultés :

☞ La flexibilité des moyens de transport est importante : dans ce cas, la méthode entraîne une explosion combinatoire du nombre de processus de transfert élémentaire générés par la méthodologie précédente : c'est le cas notamment des systèmes de transport à base de chariots filoguidés, de convoyeurs, de robots portiques, ...

☞ La zone opératoire d'un lieu physique est partagée par plusieurs ressources de

production : il s'agit ici de systèmes tels que les robots d'assemblage et de soudage,

☞ Certains constituants du procédé possèdent simultanément des fonctionnalités de stockage et de transferts : parmi ces systèmes citons les stockeurs rotatifs, les convoyeurs synchronisés de type carrousel, ...

☞ Les opérations d'assemblage sont flexibles du point de vue de l'ordre d'arrivée des constituants à assembler, CASPAIM I ne prenant en compte que des assemblages avec antériorité fixe de type palettisation,

☞ L'architecture fine du procédé n'est pas complètement connue : par exemple, pour construire le prégraphe relatif à un convoyeur à chaîne débrayable, il est nécessaire de connaître toutes les zones de stockage intermédiaire définies par les butées.

Ces limitations qui relèvent de la modélisation ont trois principales origines :

- ***La non dissociation de l'aspect opérationnel et de l'aspect fonctionnel du système.***
- ***La prise en compte non structurée et non systématique du Procédé.***
- ***Le manque de puissance de description du modèle R.d.P.C.***

Aux six limitations dont l'origine réside essentiellement dans la technique de modélisation utilisée viennent s'ajouter trois autres limitations :

- les deux premières sont d'ordre strictement méthodologique :

☞ Les performances et des situations bloquantes ne peuvent être étudiées qu'en phase de simulation après avoir construit un modèle *complet* (P.C, N.H, Procédé) du système,

☞ Les aspects relevant de la surveillance, des modes de marche, de l'ordonnancement, ... n'ont été que très partiellement abordés,

-la troisième relève de la méthodologie et de la modélisation :

☞ Si l'architecture du procédé évolue peu ou prou au cours ou après le développement du système, il faut alors réétudier complètement le système : il s'agit là d'un problème de non continuité modulaire au sens de [Meyer 1991] dans la mesure où une petite modification de l'architecture du procédé peut avoir un impact important sur l'architecture du système de commande.

I.B.5. Conclusion

La méthodologie CASPAIM I a été élaborée dans le but d'assister la démarche de conception du Système de Commande de S.F.P.M.. Cette démarche a cependant montré de nombreuses limitations. Avec un certain recul sur les travaux effectués dont nous disposons aujourd'hui, il apparaît que l'approche analytique choisie a été, d'une part, trop centrée sur la partie commande en négligeant les deux autres composantes des S.F.P.M. (N.H. et procédé) et que, d'autre part, elle n'a pas été suffisamment hiérarchisée en différents niveaux d'abstraction ce qui a entraîné une certaine confusion entre les points de vue opérationnels et fonctionnels du système. Notons néanmoins que l'approche proposée reste pertinente pour des systèmes de faible complexité.

Pour repousser ces limitations et donc étendre le domaine des systèmes étudiés, la chaîne de conception a été repensée pour donner la nouvelle méthodologie CASPAIM II [Cruette 1991], qui, tout en conservant l'objectif initial de CASPAIM I, permet, d'une part, de dissocier l'aspect opérationnel de l'aspect fonctionnel du système de production et, d'autre part, d'intégrer les aspects ordonnancement, planification et surveillance à travers une approche systémique. Cette nouvelle méthodologie, devrait également assurer une couverture plus large du cycle de vie d'un S.A.P. en intégrant la possibilité d'étudier un système de production non complètement défini.

I.C. Le projet CASPAIM II

I.C.1. Evolutions

Les limitations du projet CASPAIM I ont mené l'équipe SED du LAIL à redéfinir complètement la méthodologie CASPAIM et notamment par :

- la dissociation de l'aspect fonctionnel de l'aspect opérationnel. L'aspect fonctionnel décrit les opérations que subissent les pièces indépendamment de la mise en œuvre. Cette description s'effectue à l'aide de gammes de fabrication. L'aspect opérationnel représente l'ensemble des moyens de production et de manutention du SFPM,

- la définition de nouveaux concepts de modélisation facilitant la modularité et la généricité, intégrant les aspects objets,...

- l'utilisation de formalismes de modélisation de haut niveau, tels les Réseaux de Petri à

structures de données ou à objets [Sibertin-Blanc 1985], les langages orienté objets, les actigrammes SADT [Ross 1985], les langages synchrones [André et al 1993],...

CASPAIM II tente aussi de mieux intégrer les aspects relevant de la gestion de production, de la surveillance, de la gestion des modes de marche...

I.C.2. Système de commande d'un SFPM engendré par CASPAIM II

Dans CASPAIM II, nous décomposons un Système Flexible de Production Manufacturière Automatisé en quatre modules principaux communicants (Figure I.11) :

- le **Niveau Décisionnel Supérieur** (NDS) chargé de définir les stratégies de production et de résoudre les indéterminismes présents dans la Partie Commande Produits.

- la **Partie Commande Produits** chargée d'assurer le contrôle et le séquencement des opérations au niveau de chaque gamme de fabrication considérée de façon isolée, sans prendre en compte la résolution des problèmes posés par les évolutions simultanées de gammes concurrentes au sens de la prise en charge des contraintes d'utilisation des ressources de production.

- le **Gestionnaire de Ressources** chargé de satisfaire les différentes commandes (demandes de services) émanant de la Partie Commande Produit. Il contrôle et optimise le fonctionnement des ressources de production selon une stratégie de fonctionnement imposée par le Niveau Décisionnel Supérieur.

- le **Procédé** composé de l'ensemble des ressources de production.

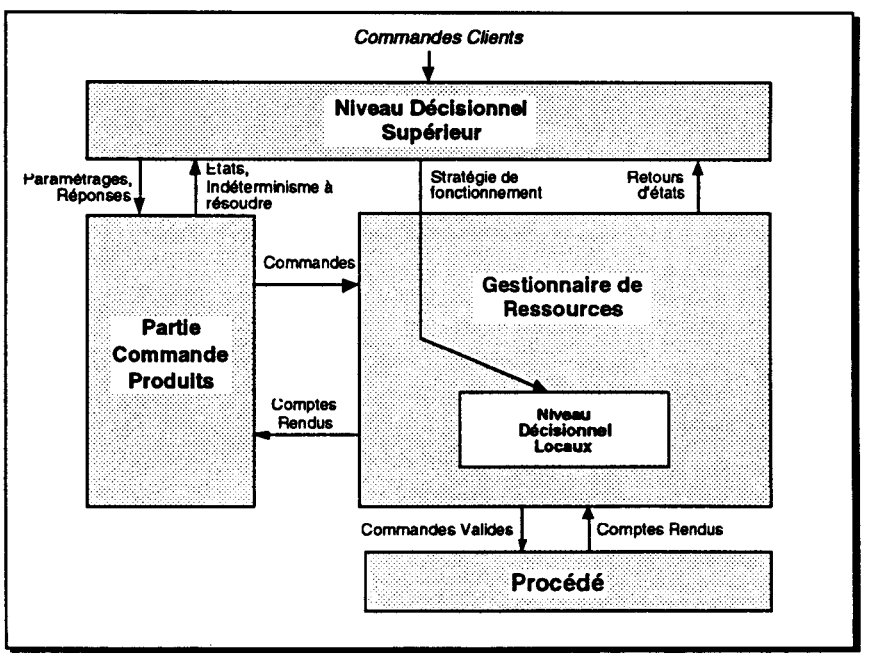


Figure I.11 : Décomposition d'un SFPM Automatisé selon CASPAIM II

Pour formaliser et systématiser cette automatisation, la méthodologie CASPAIM II a été définie par étapes successives. Nous allons aborder celles-ci plus précisément.

I.C.3. Description de la méthodologie CASPAIM II

La méthodologie CASPAIM II a été développée dans le souci de son intégration dans un atelier de génie automatique. Elle se décompose en quatre phases : la spécification, la conception, l'intégration, et enfin l'implantation en vue de l'exploitation. Ces différentes phases sont elles même composées de sous phases interdépendantes (Figure I.12).

Il est à noter que cette description n'est ni exhaustive, ni limitative; en effet, chacune de ces phases a fait ou fait actuellement l'objet d'études au L.A.I.L.

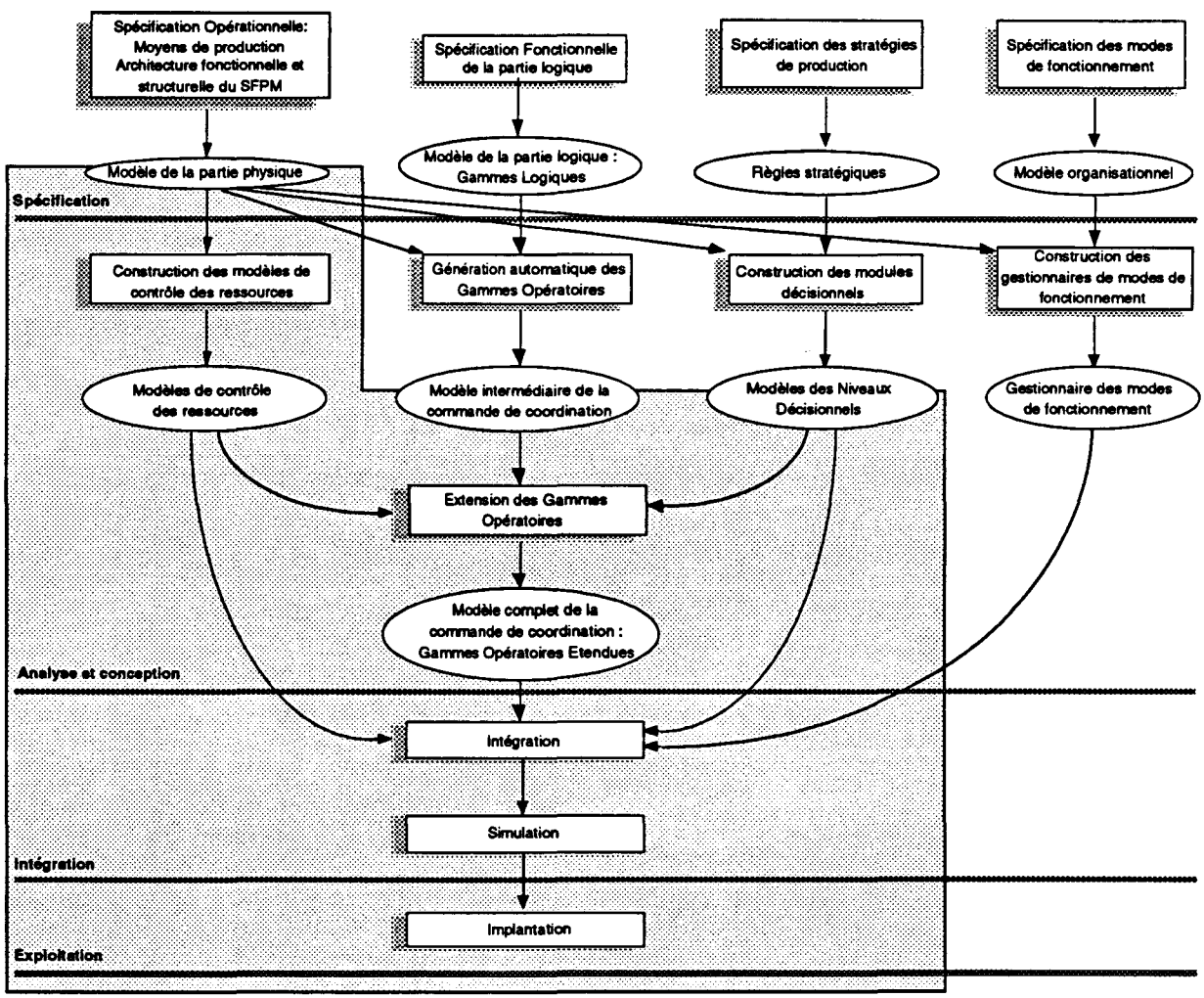


Figure I.12 : CASPAIM II : démarche de conception et situation de nos travaux (partie grisée)

Au niveau de la spécification et de la conception, l'étude de la partie physique, de la

partie logique, des niveaux décisionnels et des gestionnaires de modes de fonctionnement sont menées conjointement.

La **partie physique** considère les ressources de production et correspond à la réalisation du Gestionnaire de Ressources. La conception de cet ensemble s'effectue en trois phases successives où le modèle est peu à peu enrichi.

La **partie logique** considère les pièces à manufacturer et correspond à la réalisation de la Partie Commande Produit. La conception de cet ensemble s'effectue en deux étapes pour donner un modèle de contrôle des ressources connectable au modèle de contrôle des produits.

La conception des **niveaux décisionnels** revient à définir les différents niveaux de décisions (planification, ordonnancement, pilotage temps réel) ainsi que leurs connexions avec les autres modèles.

Enfin, la conception des **gestionnaires des modes de fonctionnement** correspond à définir les organisations relatives aux différentes gammes et aux différentes ressources de production et correspondant aux différentes reconfigurations à effectuer pour passer d'un mode de fonctionnement à un autre [Kermad et al 1993].

La finalité est d'obtenir un système hiérarchisé organisé en quatre couches (Figure I.13), celles-ci correspondant aux niveaux 0 à 3 de l'architecture **CIM**.

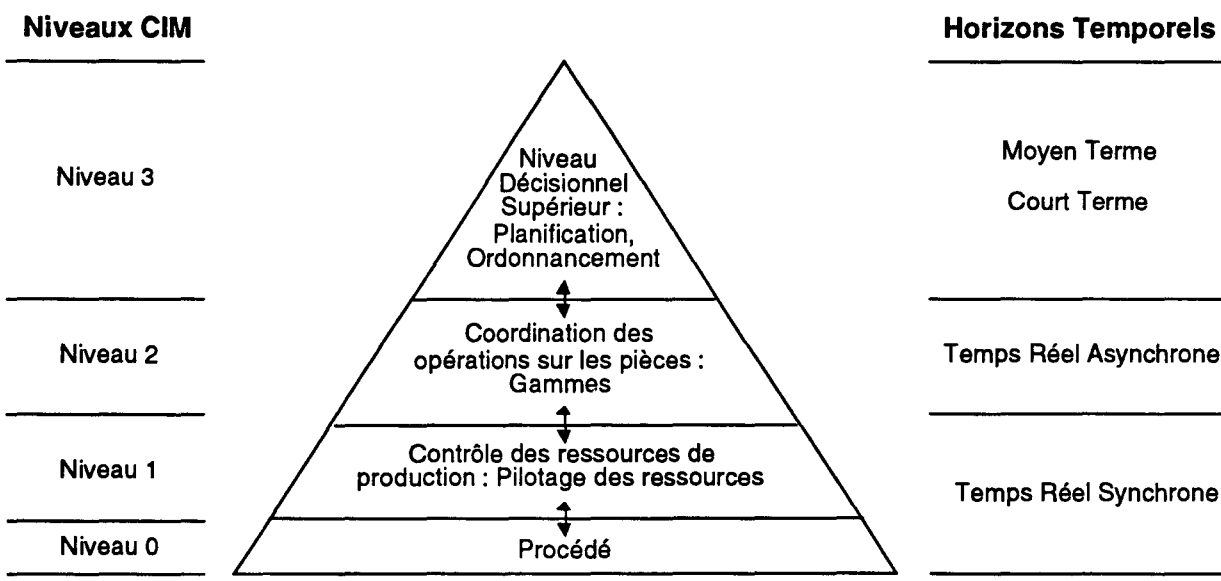


Figure II.13 : Hiérarchisation de la commande

Dans cette organisation, nous retrouvons le **Niveau Décisionnel Supérieur** (NDS); il est

composé des modules de planification, d'ordonnancement et du module décisionnel global du pilotage.

Nous retrouvons également bien séparées la gestion de la **partie logique** (Gammes) et la gestion de la **partie physique** (Pilotage des ressources). Ces deux couches peuvent intégrer des **Niveaux Décisionnels Locaux** (NDL) logiquement paramètrés par le Niveau Décisionnel Supérieur (NDS).

Cette organisation a l'avantage de bien séparer les différents types d'indéterminismes et de contraintes pouvant survenir dans la gestion d'une cellule flexible.

I.C.4. Situation de l'étude

Pour notre part, nous allons nous intéresser à l'élaboration des modèles de contrôle des ressources et à l'élaboration d'un modèle complet de la commande de coordination dans le but de les intégrer et de fournir un logiciel de contrôle/commande opérationnel, facilement implantable sur une architecture informatique répartie.

Dans ce sens, la phase de modélisation présentée dans ce mémoire a été enrichie par rapport aux travaux initiaux [Cruette 1991] dans le souci d'une mise en oeuvre effective d'un logiciel de contrôle/commande sur une architecture informatique répartie d'un site industriel et avec finalité l'automatisation de son développement afin de réaliser un atelier de génie automatique.

Ainsi, la nécessité d'une implantation répartie, la nécessité de la maintenabilité du logiciel, celui ci devant évoluer au grès des modifications et changements de produit à manufacturer et des évolutions du système de production, nous ont amené à étudier la faisabilité d'une démarche structurée, fortement modulaire et hiérarchisée.

Pour atteindre ce but, nous nous inspirerons du **génie logiciel** et plus particulièrement des **techniques orientées objets** : Analyse Orientée Objets [Coad et al 1992], Conception Orientée Objets [Booch 1992], Programmation Orientée Objets [Meyer 1988].

Nous laisserons de côté les aspects liés à la gestion des modes de fonctionnement et pour les aspects décisionnels, nous utiliserons des stratégies relativement simples. La prise en compte plus poussée de ces deux aspects ne posera pas de problèmes car, comme nous le verrons, l'approche utilisée est très modulaire et reste très ouverte.

CHAPITRE II

Modélisation de la commande d'un système flexible de production manufacturière en vue de l'implantation

SOMMAIRE DU CHAPITRE II

II.A. Introduction

Nous avons vu dans le chapitre précédent les objectifs à atteindre par une méthodologie telle que CASPAIM. La finalité est que les outils émanant de cette méthodologie soient intégrés dans un atelier de génie productique complet permettant de concevoir la commande d'un SFPM depuis la spécification jusqu'à l'implantation. Ces objectifs ne peuvent être atteints que par une approche hiérarchisée et fortement modulaire favorisant la réutilisabilité. La base fondamentale de cette orientation et sur laquelle repose toute la démarche CASPAIM II, est la dissociation des aspects opérationnel et fonctionnel du SFPM.

Dans ce sens, après avoir précisé les hypothèses de travail et après une analyse approfondie du procédé, nous aborderons la modélisation de la partie logique, celle-ci correspondant à l'aspect fonctionnel, puis la modélisation de la partie opérative, correspondant à l'aspect opérationnel. Nous aborderons ensuite la modélisation des différents niveaux décisionnels, ces derniers permettant de rendre déterministe la commande précédemment engendrée. Enfin, nous proposerons une architecture de contrôle complète d'un Système Flexible de Production Manufacturière.

II.B. Hypothèses de travail

La finalité de notre démarche est d'aboutir à la génération d'une commande opérationnelle, évolutive et facilement implantable sur site industriel. L'approche étudiée ici considère la mise en oeuvre de systèmes flexibles de production manufacturière dont l'architecture matérielle est existante ou tout au moins totalement spécifiée.

Les fonctionnalités ainsi que les procédures de mise en oeuvre de la partie opérative sont donc bien connues. Ainsi, l'ensemble des contraintes existant au niveau de chaque ressource sont référencées. Ce seront, par exemple, des contraintes d'exploitation pour les machines outils , des contraintes de trajectoire pour les robots (pour éviter les collisions...), des contraintes de coopération entre ressources (synchronisations,...), les capacités maximales des zones de stockage ou des zones tampon, etc....

Les types de langage de programmation de chaque ressource sont eux aussi censés être parfaitement connus. En effet, nous devons différencier, par exemple, les robots à apprentissage, très peu flexibles au niveau de leur programmation, des robots totalement programmables qui permettent le passage de paramètres de trajectoire.

En ce qui concerne les produits, nous considérons ceux-ci comme ayant été conçus à l'aide de logiciels de CFAO tels que EUCLID-IS [Matra 1992] ou CATIA [Dassault 1992]. La conception à l'aide de ces outils permet d'obtenir des fichiers de données CAO normalisés. Ces fichiers peuvent ensuite servir de base à l'élaboration des programmes d'usinage élémentaires. C'est ce que réalisent des logiciels tels que KRONOS [Pruvot 1993]. Ce logiciel intègre un module de simulation graphique temps réel et permet, à partir des données de CAO 3D des pièces, des caractéristiques des machines cibles (modèle géométrique, possibilités des outils de coupe, positionnement du brut d'usinage,...) et des tolérances acceptées, de définir les différentes passes d'usinage en accord avec l'opérateur. Un fois l'usinage validé, un post-processeur génère du code constitué de fonctions ISO compréhensible par les commandes numériques des machines outils [Magnin et al 1991]. Pour notre part, nous supposerons ces étapes réalisées et disposer de l'ensemble des programmes d'usinage.

L'approche abordée ici considère l'étude d'une cellule flexible mais cette approche pourra sans problème être étendue à l'étude de systèmes industriels plus larges tels que les ateliers flexibles.

II.C. Exemple illustratif

Pour illustrer notre démarche, nous avons imaginé une cellule flexible de production mécanique aux possibilités étendues intégrant les principaux moyens de production utilisés dans cette branche de l'industrie. L'architecture matérielle de celle-ci est décrite par la figure II.1.

La structure de cette cellule permet de mettre en évidence un grand nombre de problèmes pouvant apparaître dans tout SFPM :

- les ressources partagées,
- le routage des pièces,
- le partage de zones de stockage,
- les choix multiples de machines d'usinage,...

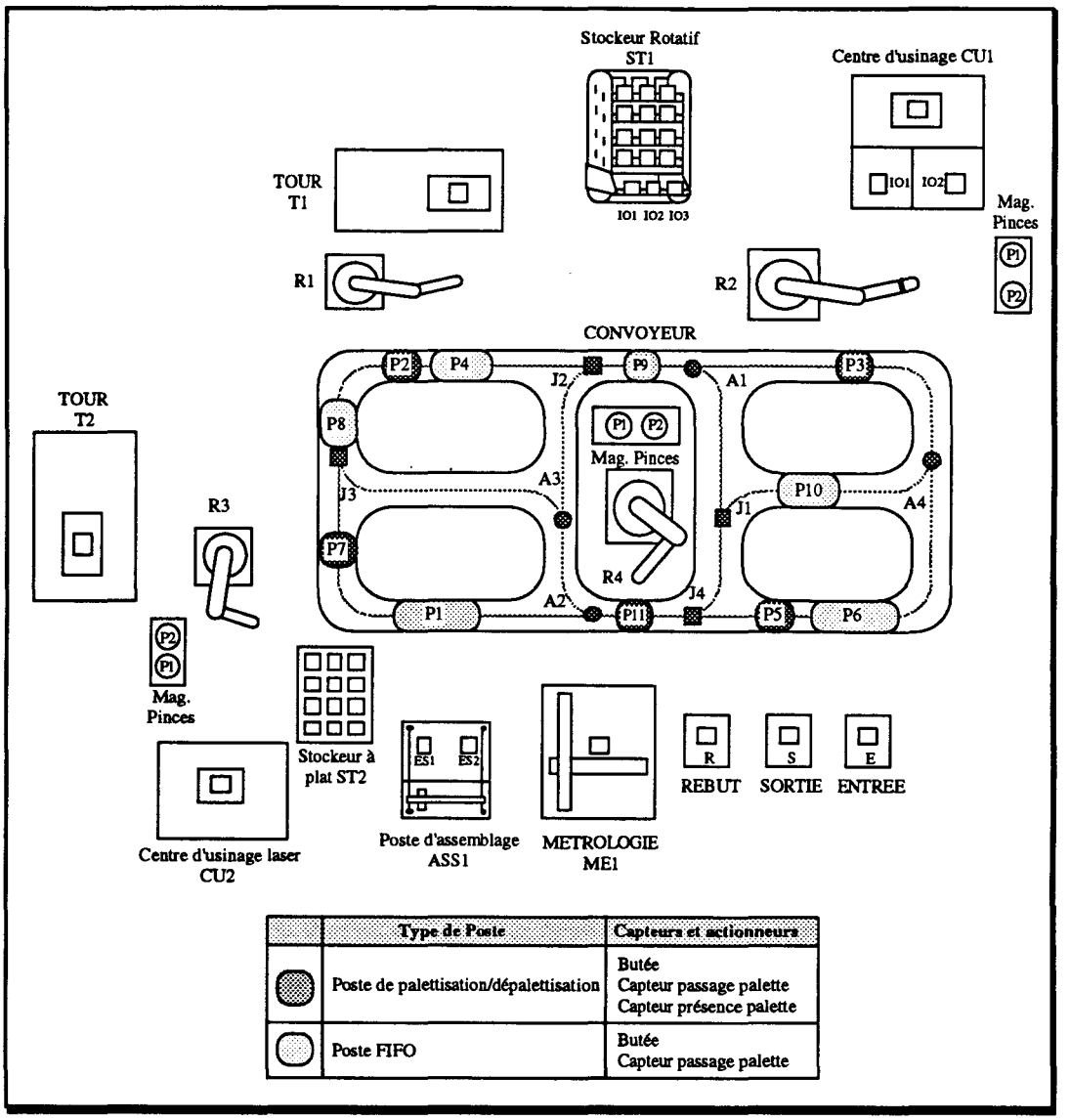


Figure II.1 : Architecture matérielle de la cellule flexible

Cette cellule est composée des éléments suivants :

➪ *pour la manutention des pièces :*

✔un convoyeur à chaîne pour le transfert des pièces palettisées entre les différents éléments de la cellule. Il est composé de quatre aiguillages (A1, A2, A3 et A4), de cinq postes de palettisation/dépalettisation (P2, P3, P5,P7 et P11) et de six postes de type FIFO (P1, P4, P6, P8, P9 et P10). L'ensemble des postes comportent à leur tête une butée et une série de capteurs.

✔deux robot de manutention universels (6 axes) R1 et R3, avec système de changement d'effecteurs pour R3, programmables par langage spécialisé (LM, NUM...),

✔deux robot de manutention universels (6 axes) R2 et R4, avec système de changement d'effecteurs dont les déplacements sont enregistrés par apprentissage.

Il est à noter que l'architecture du convoyeur a l'avantage de permettre le contournement des zones de chargement/déchargement afin de répartir les flux de palettes et donc d'éviter, en partie, les engorgements.

➪ *pour la transformation des pièces :*

✔deux tours identiques à commande numérique T1 et T2 sans tampon d'entrée/sortie,

✔un centre d'usinage classique à commande numérique CU1 à deux tampons d'entrée/sortie mono pièce,

✔un centre d'usinage laser à commande numérique CU2 sans tampon d'entrée/sortie.

La présence de plusieurs unités fonctionnelles du même type permet de prévenir les indisponibilités temporaires des fonctions clés de la cellule dues à des défaillances ou à la maintenance de l'une des unités.

➪ *pour le contrôle qualité :*

✔un centre de métrologie tridimensionnel ME1.

➪ *pour l'assemblage de pièces :*

✔un poste d'assemblage autonome ASS1.

Le centre d'assemblage est autonome afin de ne pas occuper inutilement une autre ressource telle que le robot R4.

➪ *pour le stockage des pièces :*

✔un stockeur rotatif ST1,

✔un stockeur à plat ST2.

Ces deux stockeurs permettent de libérer le convoyeur et donc d'améliorer l'écoulement des palettes et donc des pièces sur ce dernier. De plus, ils permettent de disposer en permanence de pièces pour les machines afin d'optimiser leur charge.

➪ *pour l'entrée et la sortie des pièces :*

✔un magasin d'entée/sortie de pièces constitué d'une FIFO d'entrée de pièces brutes (E), d'une FIFO de sortie de pièces finies (S) et enfin d'une FIFO de rebut de pièces défectueuses (R).

II.D. Analyse du procédé

II.D.1. Introduction

Une phase d'analyse du procédé est nécessaire afin de concevoir les modèles de commande des ressources mais aussi pour mener à bien la conception de la commande de coordination [Amar 1994] [Ausfelder 1994], des modules décisionnels [Tawegoum et al 1993] et des gestionnaires de mode de fonctionnement [Maik et al 1994].

Après une caractérisation des ressources basée sur une approche particulière de leur complexité, nous allons réaliser une analyse descendante basée sur une décomposition structurelle afin de disposer d'éléments du procédé observables (accès à l'information des capteurs) et commandables (accès aux actionneurs). Ensuite, nous aborderons une méthode d'analyse ascendante basée sur une agrégation structurelle du procédé.

La première analyse nous permettra, par la suite, d'élaborer les modèles de commande des ressources tandis que la seconde nous permettra de hiérarchiser la commande de coordination.

II.D.2. Les deux familles de ressources

Une phase d'analyse approfondie a permis de distinguer deux types majeurs de ressources : les **ressources simples** ou **élémentaires** et les **ressources complexes**. Afin de spécifier les particularités de ces deux types de ressource, un ensemble de concepts ont été établis dans [Amar et al 1992]. Quelques uns de ces concepts concernent respectivement les relations d'accessibilité et les lieux caractéristiques :

La **notion d'accessibilité** définit les liens existants entre différents moyens de production concernant les échanges de pièces. Un moyen est en accessibilité directe ou interne avec un autre si tout produit peut transiter de l'un à l'autre sans passer par un moyen physique intermédiaire. Les relations d'accessibilité indirectes ou externes se déduisent automatiquement par transitivité.

Un **lieu caractéristique** est un lieu physique, mobile ou non, pouvant recevoir un produit. Ce lieu peut être soit un lieu de transformation fonctionnelle, soit un lieu en relation d'accessibilité externe.

Il est à noter que cette dernière définition s'applique à des produits uniques mais peu également s'appliquer pour des lots groupés de produits.

En s'appuyant sur ces définitions, nous pouvons en déduire la définition caractérisant le type de la ressource considérée :

Une ressource est dite **complexe** si elle comporte soit plusieurs lieux caractéristiques ou soit, au moins, un lieu non caractéristique. Elle est dite **élémentaire** ou **simple** dans les autres cas [Amar et al 1992].

De fait, les ressources simples délivrent un seul service à la fois. C'est le cas pour, par exemple, des robots de manutention à simple préhenseur où le service est le transport d'une pièce unique ou d'un lot groupé de pièces d'un lieu vers un autre. C'est aussi le cas pour les machines outils à commande numérique mono emplacement ou à stockage simple en ligne (FIFO) où le service est un usinage pré défini. Ainsi, dans l'exemple précité, nous avons huit ressources simples: R1, R2, R3, R4, T1, T2, CU2, ME1.

En ce qui concerne les ressources complexes, elles peuvent selon les cas être :

- multi service,
- avoir un parallélisme de flux,
- avoir un parallélisme de stockage,
- avoir tout ou partie de ces caractéristiques.

Dans l'exemple, c'est le cas des ressources CONVOYEUR, ST1, ST2, CU1, ASS1.

Par souci de cohérence, ces particularités seront prises en compte tout au long des phases de conception et de développement.

II.D.3. Analyse multi niveaux du procédé

II.D.3.a. Niveau 0 de l'analyse

Une première **décomposition structurelle,** que nous nommerons **de niveau 0**, est obtenue en ayant une vision orientée **ressources de production élémentaires** indépendamment

de leurs types (simple ou complexe). Associé à cette décomposition, nous trouvons le graphe des accès où nous retenons les relations d'accessibilités entre les différentes ressources ainsi que les capacités d'accueil de ces dernières. La figure II.2 résume l'ensemble de ces données.

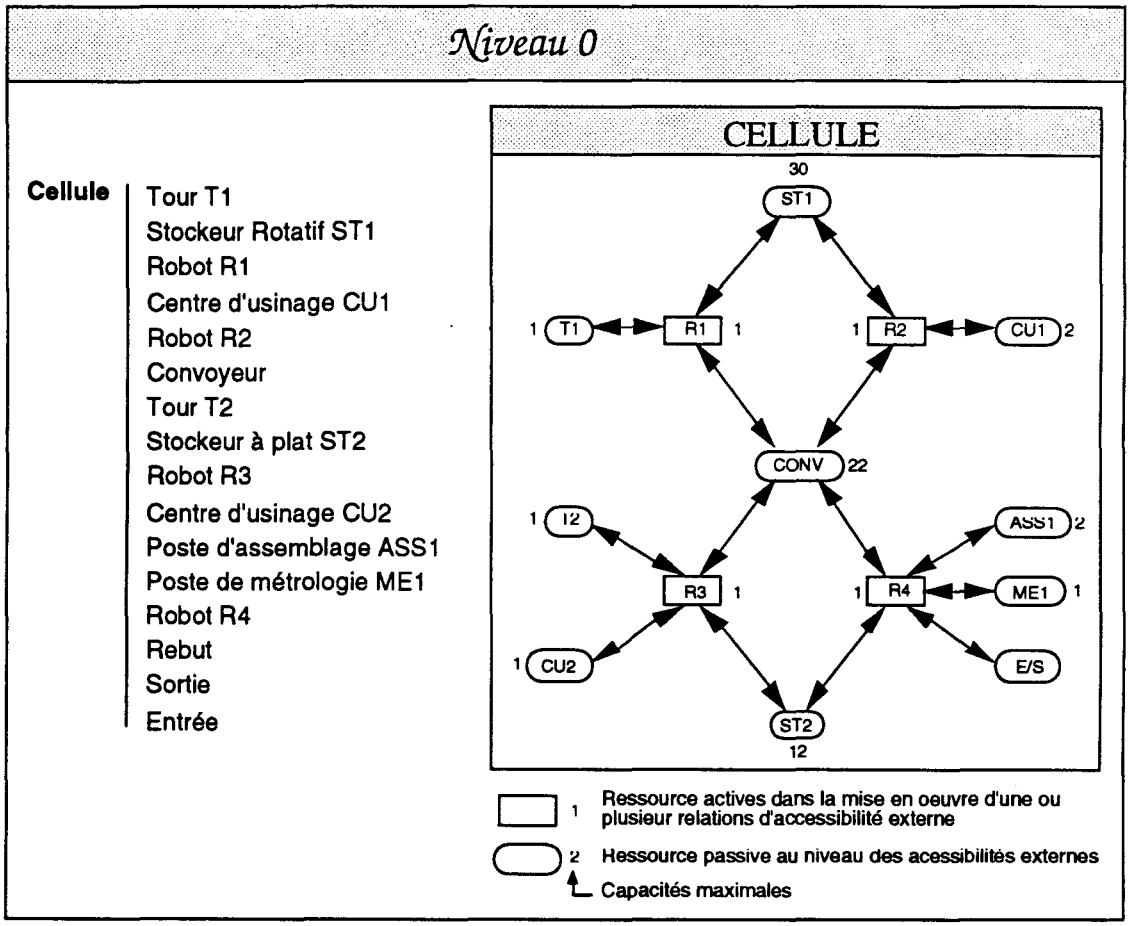


Figure II.2 : Décomposition structurelle de niveau 0 et graphe des accès associé

Dans la mise en oeuvre des relations d'accessibilités externes, nous pouvons distinguer les ressources actives des ressources passives. Les ressources actives sont celles qui transfèrent physiquement les pièces d'une ressource à une autre. Typiquement, ce seront des ressources de chargement/déchargement telles que les robots de manutention ou les systèmes de convoyage à accès direct. Les autres ressources qui sont donc desservies par ces ressources actives sont les ressources passives.

II.D.3.b. Analyse descendante

Dans notre optique qui est la conception et la mise en oeuvre de la commande de coordination, nous ne retiendrons pas systématiquement dans leur totalité l'analyse structurelle arborescente proposée par D. CRUETTE dans [Cruette 1991] ou l'analyse structuro

fonctionnelle proposée par S. AMAR dans [Amar et al 1990]. En effet, étant donnée la finesse de ces analyses, celles-ci sont plutôt destinées à la conception de la commande fine et/ou à la conception de la surveillance des machines [Toguyeni 1992].

Pour notre part, nous nous focaliserons sur la conception du logiciel de contrôle/commande, nous aurons donc une analyse fortement orientée vers la **commande de coordination**. Pour les machines outils et les robots, nous supposerons que ce sont des machines qui nous ont été livrées 'clés en mains'. Pour ce type de ressource, seuls les programmes d'usinage, de transfert,... sont à concevoir à l'aide d'outils de CFAO et il n'est alors pas nécessaire de pratiquer une décomposition structurelle.

Par contre, une analyse approfondie peut s'avérer nécessaire pour certaines ressources conçues spécifiquement ou construites à façon. Ce sera principalement le cas pour des ressources complexes telles que les convoyeurs. En effet, contrairement aux robots ou aux machines outils, la commande de bas niveau de ce type de ressource est généralement spécifique et il est nécessaire de la concevoir dans sa globalité.

Ainsi, si nous considérons le convoyeur de la cellule prise en exemple, pour pouvoir contrôler les palettes circulant sur celui ci, il faut le décomposer en éléments indépendants (Figure II.3); ceux-ci pouvant être de deux natures :

- les **éléments commandables** telles que les postes et les aiguillages que l'on pourra contrôler individuellement,

- les **éléments non commandables** correspondant à des sections critiques telles les jonctions et dont il faudra gérer les accès pour éviter les collisions.

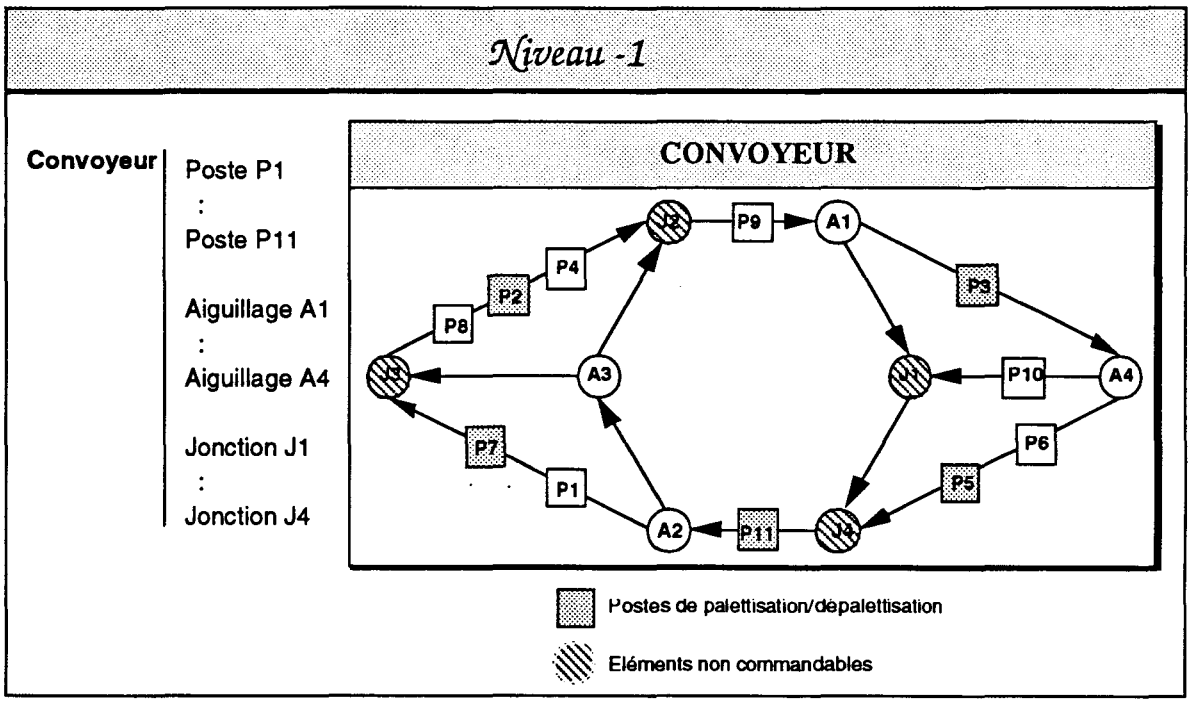


Figure II.3 : Décomposition structurelle du convoyeur

Cette décomposition structurelle sera désignée comme étant de **niveau -1.** Le processus de décomposition structurelle doit être mené tant que le procédé n'est pas observable (accès à l'information des capteurs) ou non commandable (accès aux actionneurs). Nous aurons alors une décomposition de plus en plus fine tendant vers une vision microscopique (niveaux -2, -3,...). Pour le convoyeur, cette décomposition s'arrête au niveau -2 (Figure II.4).

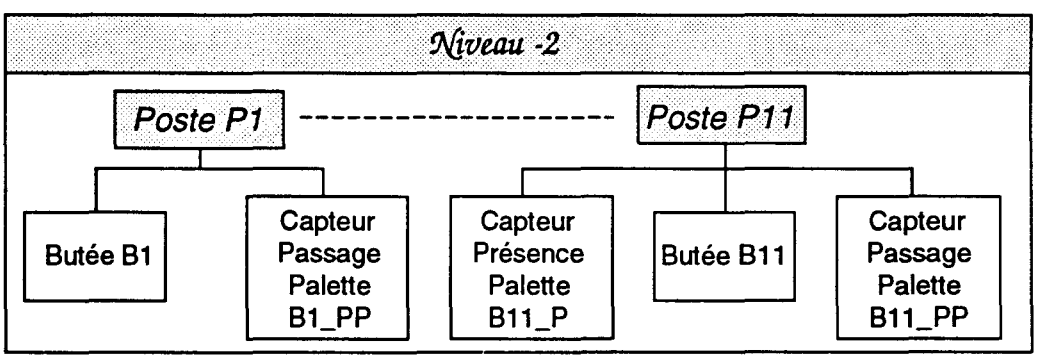


Figure II.4 : Décomposition structurelle du convoyeur de niveau -2

Au niveau le plus bas, ce qui importe le plus pour la commande, ce sont les informations recueillies par les capteurs, le rôle des actionneurs ainsi que les modes opératoires.

Ainsi, pour pouvoir gérer les divers éléments commandables, il est nécessaire de décrire le comportement des objets entrant dans leur composition, à savoir les objets commandables [Elkhattabi 1993] et les objets capteurs . Cette description s'effectuera à l'aide de graphes du type états/transitions tels les automates à états finis, les Statecharts ou les réseaux de Petri [Calvez 1990].

Pour le convoyeur, nous devons décrire le comportement des objets commandables que sont les butées et les aiguillages (figure II.5).

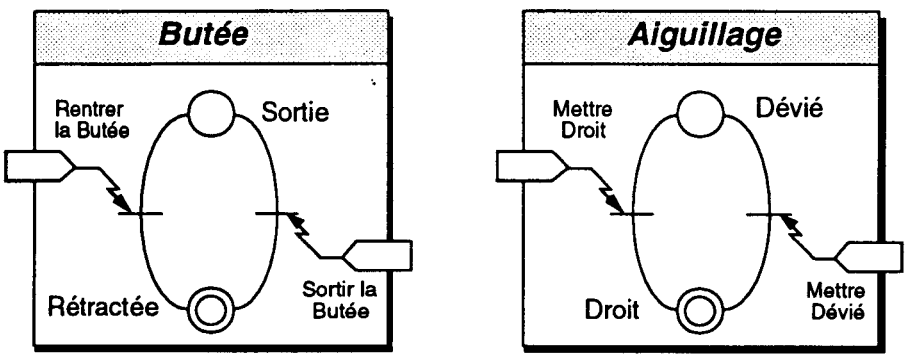


Figure II.5: Automates d'états des butées et des aiguillages

Ensuite, pour pouvoir mettre en place une stratégie de commande afin de contrôler les palettes, il est nécessaire de décrire les modes opératoires liant les objets commandables et les objets capteurs. A titre d'exemple, la figure II.6 présente le fonctionnement du système de butée dont nous disposons sur le convoyeur de la cellule flexible de l'Ecole Centrale de Lille [Mayet 1989].

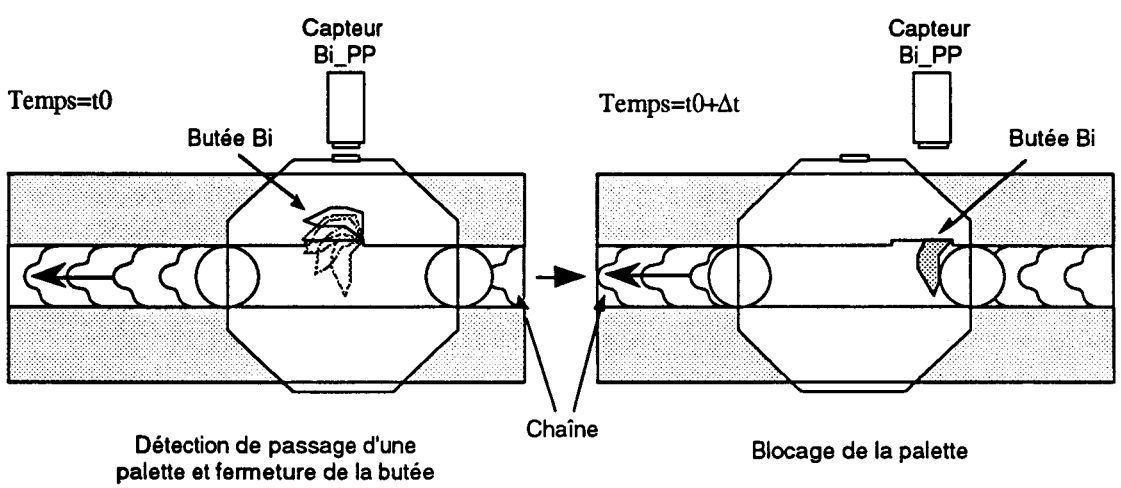


Figure II.6: Exemple de fonctionnement de butées de convoyeur

En sachant que le convoyeur est du type à chaîne, le fonctionnement de ce système de butée doit être le suivant : lorsqu'une palette est détectée grâce au capteur Bi_PP, la butée qui était précédemment rétractée doit sortir de son logement pour se trouver au centre de la palette. Cette dernière, continuant sa progression, doit venir se bloquer sur la butée et glisser sur la chaîne.

A partir de cette description littérale, nous pouvons en déduire les graphes fonctionnels décrivant le blocage et la libération des palettes (figure II.7).

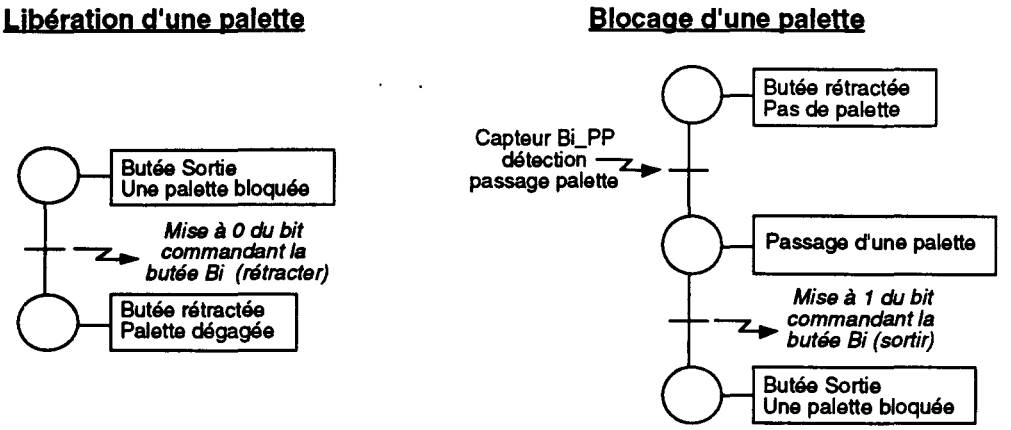


Figure II.7 : Graphes fonctionnels pour la gestion des butées

Parallèlement à cette analyse, il est possible d'obtenir des décompositions structurelles de niveau +1,+2,... par une démarche ascendante d'agrégation de ressources.

II.D.3.c. Analyse ascendante

Dans le but de hiérarchiser la commande de coordination, nous allons mener une agrégation multi niveaux de l'ensemble des ressources de production composant notre SFPM afin d'obtenir une vision hiérarchisée du procédé.

Pour cela, nous nous baserons sur les travaux de S. BOIS [Bois 1991], appliqués à l'origine à la gestion des modes de marche.

Afin de modéliser les liens et dépendances entre ressources de production, S. BOIS a recensé une grande partie des contraintes existant entre celles ci. Ce sont, entre autre :

✔ *des contraintes sur états telles que :*

- la contrainte de coopération CC,
- la contrainte de coopération partagée CCP,

- la contrainte d'exclusion CE,
- la contrainte structurelle série CSS,
- la contrainte structurelle parallèle CSP,

✔ *des contraintes sur changements d'états telles que :*

- la contrainte procédurale CPRO,
- la contrainte d'observabilité CO,...

La **contrainte de coopération** est présente lorsque deux ou plusieurs ressources coopèrent de manière étroite et sont fortement dépendantes les unes des autres. C'est le cas, par exemple, d'une machine de transformation et de son robot de chargement/déchargement.

La **contrainte de coopération partagée** est présente lorsque deux ou plusieurs machines partagent une même ressource. Ces machines peuvent fonctionner de manière parallèle et indépendante et n'avoir aucun lien particulier à l'exclusion de demande d'utilisation de la ressource partagée.

La **contrainte d'exclusion** se rencontre lorsque deux ou plusieurs ressources doivent être dans des états totalement exclusifs. Ce type de contrainte est spécifique aux états de marche.

La **contrainte procédurale** correspond à la prise en compte de protocole de marche à respecter entre machines tandis que la contrainte d'observabilité correspond à la possibilité ou non d'accéder aux informations nécessaires à la commande de la machine (aspect réseau, aspect informatique).

Enfin, afin d'obtenir des modèles suffisamment structurés, les **contraintes structurelles** ont été introduites pour palier aux cas ou aucune des contraintes fonctionnelles vues précédemment ne pourraient être dégagée.

Du point de vue des ressources de production, les contraintes le plus souvent rencontrées dans les systèmes de production de type JOB SHOP sont la contrainte de coopération CC et la contrainte de coopération partagée CCP.

De l'ensemble de ces contraintes, S.BOIS dégage l'existence de deux sous ensembles. Ce sont l'ensemble des contraintes de type ET et l'ensemble des contraintes de type OU. Les contraintes de type ET, telle que la contrainte de coopération CC, restreignent la notion de flexibilité tandis que les contraintes de type OU, telle que la contrainte de coopération partagée, engendrent de la flexibilité.

Pour représenter ces contraintes existant entre ressources, S.BOIS utilise une structure basée sur des arborescence de type ET/OU où chaque association de ressource par l'intermédiaire d'une même contrainte génère une entité dénommée machine ou ressource virtuelle (Figure II.8).

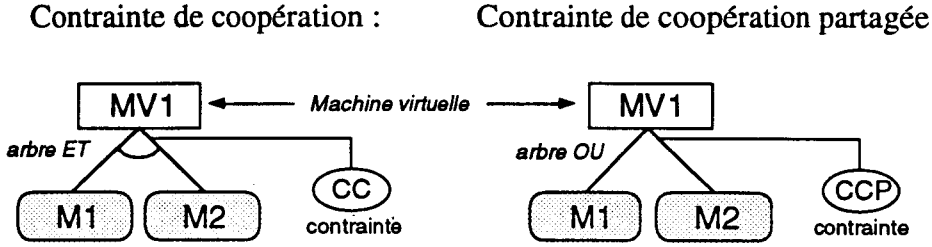


Figure II.8 : Représentation graphique de contraintes

En nous basant sur ces définitions et par étapes successives, nous allons pouvoir agréger en ressources virtuelles l'ensemble de nos ressources de production. La première étape consiste à répertorier dans une table l'ensemble des contraintes existant entre les ressources de production élémentaires constituant le SFPM étudié (Niveau 0 de la décomposition structurelle).

Pour notre part, dans toute la démarche, nous nous restreindrons aux seules contraintes de type ET. En effet, au niveau de la commande de coordination, la flexibilité engendrée par les contraintes de type OU doit être conservée.

Après avoir recensé l'ensemble des contraintes existant entre les différentes ressources du SFPM étudié, nous les répertorions dans une table des contraintes. Pour la cellule prise en exemple, nous avons alors la table des contraintes décrite par la figure II.9.

Sur cette table des contraintes, nous pouvons pratiquer des agrégation successives en nous basant sur la méthode définie par S.BOIS. Ainsi, la règle générale permettant de passer d'une agrégation structurelle de niveau n vers une agrégation structurelle de niveau n+1 consiste à rassembler en une ressource virtuelle chaque ensemble de ressources liées par la même contrainte Ci.

En tenant compte de ces remarques, nous pouvons effectuer la première agrégation permettant de passer du niveau 0 au niveau 1 de l'agrégation structurelle. Dans tous les cas, nous utiliserons la notation suivante :

RVNi.j : ressource virtuelle numéro j du niveau d'agrégation i.

	T1	R1	CU1	R2	ST1	CV	ST2	R3	T2	CU2	R4	ASS1	ME1	E/S
T1		CC1												
R1	CC1				CC3	CC4								
CU1				CC2										
R2			CC2		CC3	CC4								
ST1		CC3		CC3										
CONV		CC4		CC4				CC4			CC4			
ST2								CC6			CC6			
R3						CC4	CC6		CC5	CC5				
T2								CC5						
CU2								CC5						
R4						CC4	CC6					CC7	CC7	CC7
ASS1											CC7			
ME1											CC7			
E/S											CC7			

Figure II.9 : Table des contraintes initiale

Si nous considérons, par exemple, le tour T1 et le robot R1, il existe un contrainte de coopération CC1 entre ces deux ressources, nous pouvons donc les agréger en une ressource virtuelle de niveau 1 que nous appellerons RVN1.1. Si nous considérons l'ensemble de notre SFPM, nous obtenons la nouvelle table des contraintes décrite par la figure II.10.

Niveau 0

	T1	R1	CU1	R2	ST1	CV	ST2	R3	T2	CU2	R4	ASS1	ME1	E/S
T1		CC1												
R1	CC1				CC3	CC4								
CU1				CC2										
R2			CC2		CC3	CC4								
ST1		CC3		CC3										
CONV		CC4		CC4				CC4			CC4			
ST2								CC6			CC6			
R3						CC4	CC6		CC5	CC5				
T2								CC5						
CU2								CC5						
R4						CC4	CC6					CC7	CC7	CC7
ASS1											CC7			
ME1											CC7			
E/S											CC7			

☞

Niveau 1

	RVN1.1	RVN1.2	ST1	CONV	ST2	RVN1.3	RVN1.4
RVN1.1			CC3	CC4			
RVN1.2			CC3	CC4		CC4	
ST1	CC3	CC3					
CONV	CC4	CC4				CC4	CC4
ST2						CC6	CC6
RVN1.3				CC4	CC6		
RVN1.4				CC4	CC6		

Figure II.10 : Première phase d'agrégation :
réduction de la table des contraintes

Si l'on se restreint à la règle précédemment énoncée, il n'y a pas forcément unicité de l'agrégation. Ainsi, dans cet exemple, nous pourrions agréger en une seule ressource virtuelle le convoyeur et les robots R1, R2, R3 et R4, ceux-ci étant liés par la contrainte CC4.

Le choix d'une solution plutôt qu'une autre n'influe pas sur les méthodes de conception qui vont suivre. Le mieux est de choisir l'agrégation qui s'intégrera le plus facilement au sein de l'informatique répartie du site de production (cf. § III).

Une fois le choix de l'agrégation arrêté, nous pouvons représenter une vision structurelle simplifiée du SFPM où chaque ressource virtuelle est considérée comme une ressource de production autonome (figure II.11).

Dans cette représentation, nous avons indiqué à quelle ressource virtuelle sont dédiés les échanges de pièces avec l'extérieur (E/S) ainsi que les ressources permettant les échanges entre ressources virtuelles ou entre ressources virtuelles et ressources réelles. L'ensemble de ces données sera utilisé lors de la conception de la commande de coordination afin de hiérarchiser celle-ci.

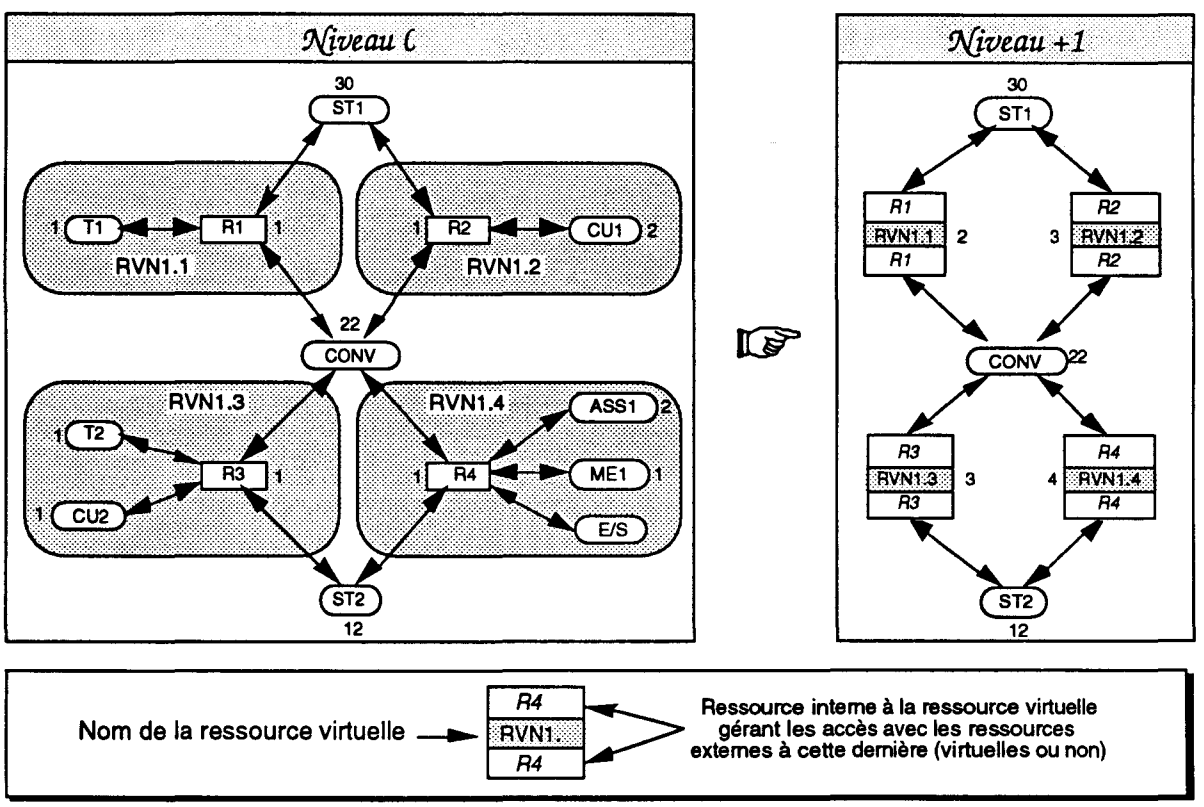


Figure II.11 : Agrégation structurelle de niveau 1

Parallèlement à cette agrégation structurelle, nous pouvons associer un point de vue fonctionnel et opérationnel. En effet, la ressource virtuelle RVN1.1 hérite à la fois des fonctionnalités du tour T1 et des fonctionnalités du robot R1, à savoir la possibilité de tournage et la possibilité de chargement/déchargement.

Nous pouvons donc caractériser chacune de ces ressources virtuelles par leurs entrées, leurs sorties, leur capacité maximale et enfin l'ensemble des opérations qu'elles peuvent réaliser.

C'est ainsi que la ressource virtuelle RVN1.3 , composée du tour T2, du centre d'usinage CU2 et du robot R3 , a pour entrées ou sorties soit le poste P7 du convoyeur (CONV.P7) soit une des zones de stockage du stockeur ST2, R3 étant l'administrateur de ces entrées/sorties. De plus, cette ressource virtuelle a une capacité maximale de 3 pièces et peut à la fois tourner et fraiser. Ces données peuvent être rassemblées dans un tableau nommé **'table caractéristique de la ressource '**(Figure II.12).

RVN1.1

		Administrateur
Entrées	CONV.P2 ST1	R1 R1
Sorties	CONV.P2 ST1	R1 R1
Opérations réalisables	Tourner	
Capacité maximale	2	

RVN1.3

		Administrateur
Entrées	CONV.P7 ST2	R3 R3
Sorties	CONV.P7 ST2	R3 R3
Opérations réalisables	Tourner, Fraiser	
Capacité maximale	3	

Figure II.12 : Tables caractéristiques des ressources virtuelles RVN1.1 et RVN1.3

En nous basant sur les mêmes règles, nous pouvons poursuivre l'agrégation par la réduction de la table des contraintes de niveau 1 pour obtenir une décomposition structurelle de niveau 2. (Figure II.13).

Il est à noter que les ressources virtuelles précédemment générées ne sont pas différenciées des autres ressources.

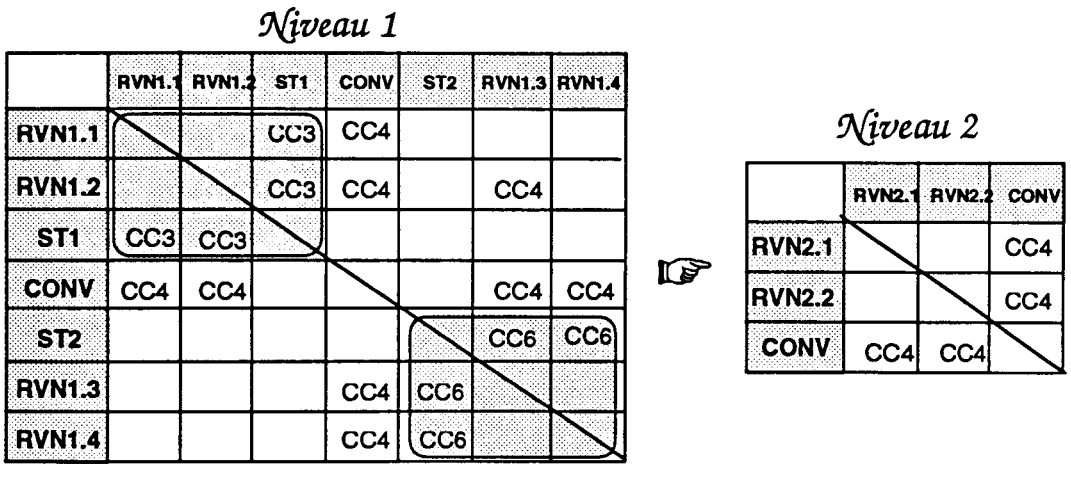


Figure II.13 : Réduction de la table des contraintes : seconde phase d'agrégation

Cette agrégation nous donne de nouvelles ressources virtuelles (de niveau 2) englobant encore plus de fonctionnalités (Figure II.14).

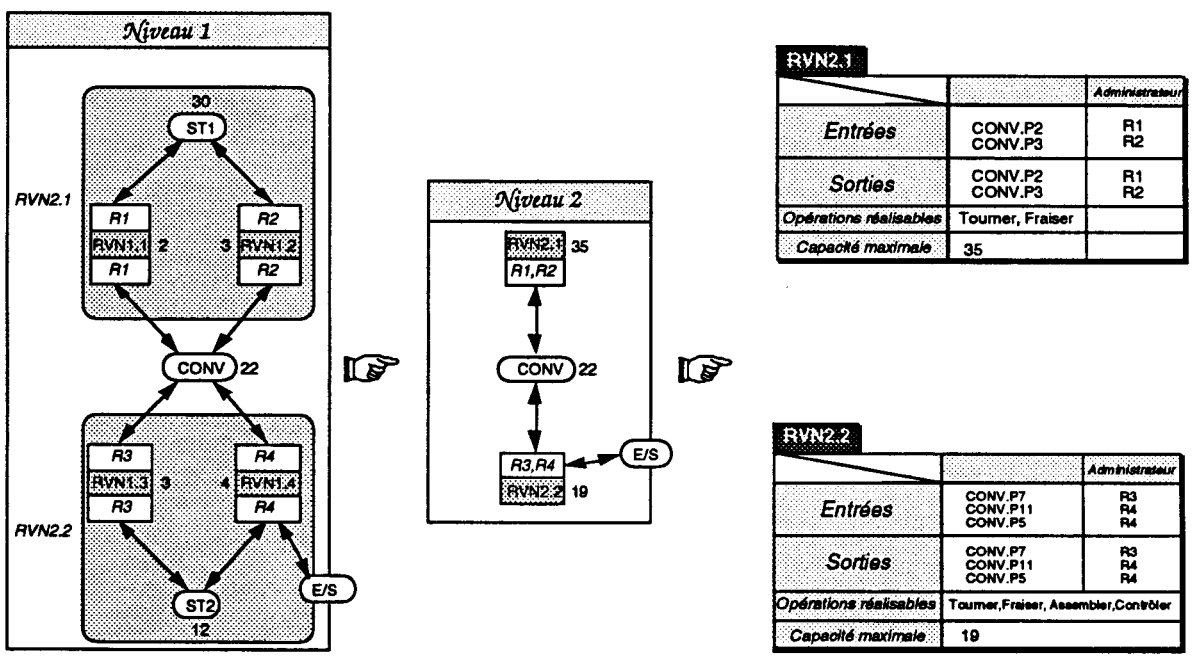


Figure II.14 : Seconde agrégation donnant une décomposition structurelle de niveau 2

L'agrégation s'arrête lorsque il n'y a plus de possibilité d'agrégation et généralement, la dernière ressource virtuelle obtenue correspond au système de production pris dans sa totalité (Figure II.15).

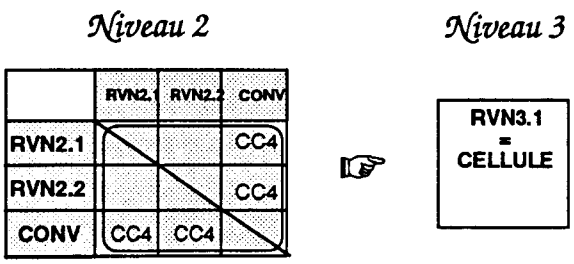
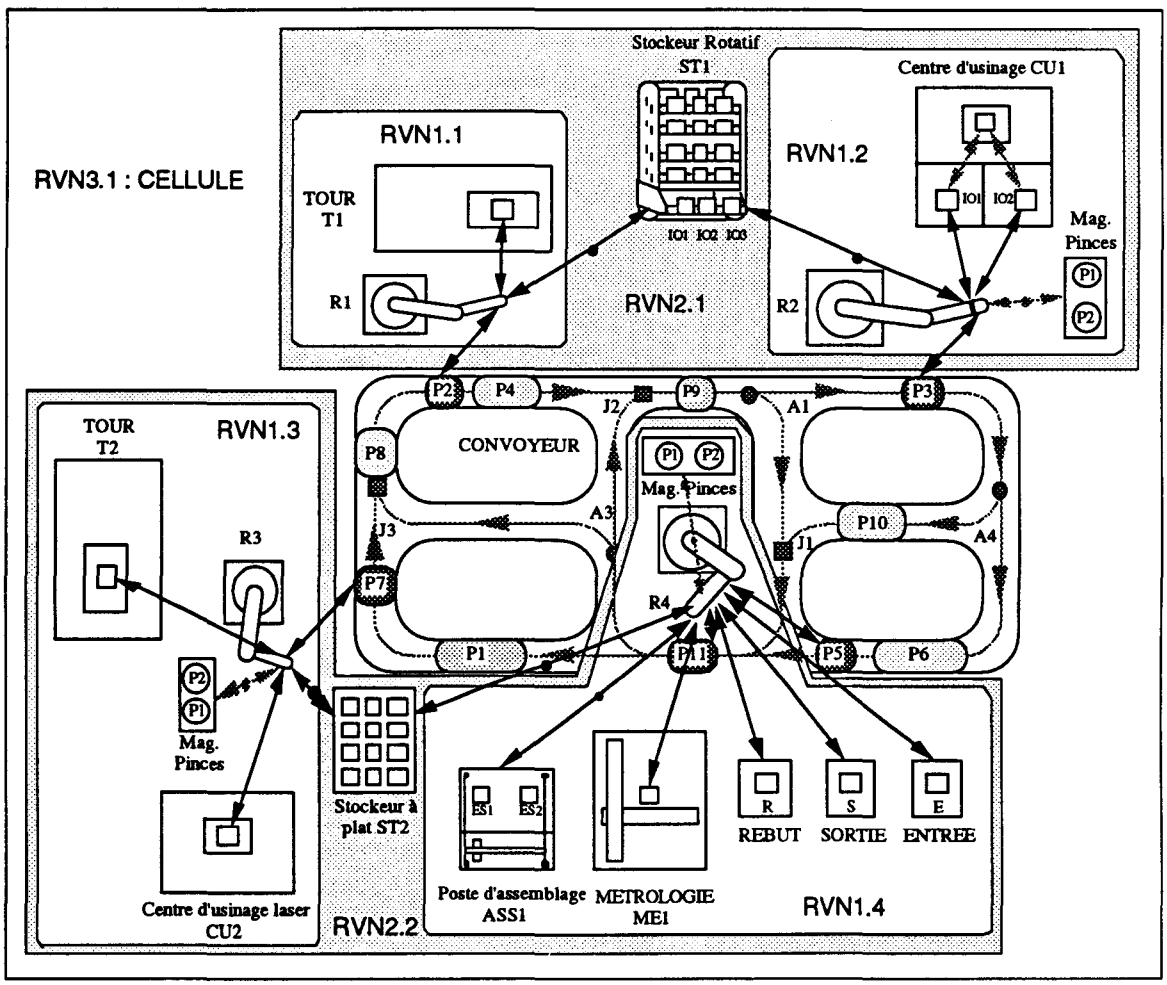


Figure II.15 : Réduction de la table des contraintes : dernier niveau d'agrégation

Si nous considérons l'ensemble de la cellule, nous pouvons reporter sur l'architecture physique de celle-ci, les relations d'accessibilité ainsi que les différentes agrégations (figure II.16).



Relation d'acessibilité interne
Relation d'acessibilité externe
Relation d'acessibilité externe multiple

	Type de Poste	Capteurs et actionneurs
	Poste de palettisation/dépalettisation	Butée Capteur passage palette Capteur présence palette
	Poste FIFO	Butée Capteur passage palette

Figure II.16 : Résumé des agrégations structurelles

L'ensemble des décompositions structurelles et des agrégations structurelles aura une répercussion sur le pilotage de la cellule en lui apportant une dimension hiérarchique. Mais il ne faut pas oublier pour autant l'aspect fonctionnel. Il est intéressant de regrouper les deux aspects sous forme d'une décomposition structuro-fonctionnelle plus ou moins complexe suivant l'utilisation qui en sera faite [Amar 1994]. La figure II.17 résume l'ensemble des données nous intéressant plus particulièrement.

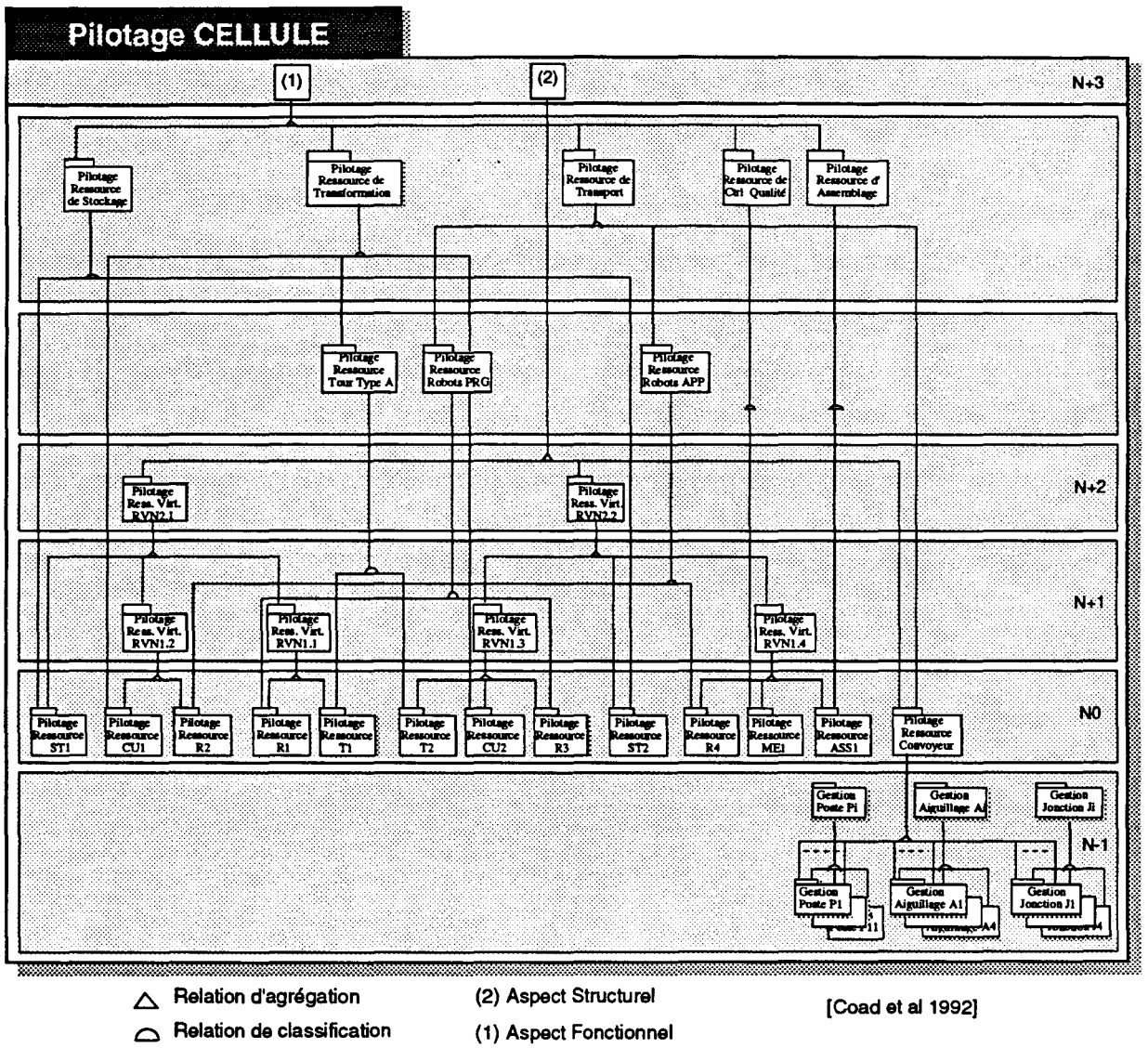


Figure II.17 : Organisation hiérarchique du pilotage de la cellule

II.E. La Partie Logique

II.E.1. Introduction

Dans cette partie, nous allons montrer comment construire, par étapes successives, la commande de coordination hiérarchisée et modulaire de notre SFPM.

II.E.2. Exemple de pièces à produire

Nous allons considérer la mise en fabrication de 5 types de pièces:

✔ les pièces de type P1 qui subissent un tournage de type 1 → P1=t1,

✔ les pièces de type P2 qui subissent un tournage de type 2 et un fraisage de type 2 dans un ordre indifférent → P2=(t2f2/f2t2),

✔ les pièces de type P3 qui subissent un tournage de type 3 puis un fraisage de type 3 et enfin un contrôle de type 2 → P3=t3f3c2,

✔ les pièces de type P4 qui subissent un fraisage de type 1 puis un contrôle de type 1 → P4=f1c1,

✔ les pièces de type P5 qui correspond à l'assemblage des pièces de type P3 et des pièces de type P4 → P5=P3+P4.

Les pièces de type P3 et P4 servent uniquement à la production des pièces de type P5, de ce fait, elles ne seront pas produites à titre individuel. Ainsi, à toute pièce P3 produite correspondra une pièce P4 et réciproquement.

II.E.3. Les Gammes Logiques

Nous définissons tout d'abord les **Gammes Logiques (GL)**; elles permettent de prendre en compte le séquencement des opérations sur chaque type de pièce tout en tenant compte des possibilités de flexibilité dans leur ordre d'exécution [Cruette 1991].

Les gammes logiques sont décrites à l'aide de Réseaux de Petri ordinaires [Brams 1983] où les opérations sur les produits sont représentées par les transitions et les différents états de ceux-ci sont représentés par les places. Chaque produit est alors logiquement représenté par un jeton.

Dans le cas de pièces simples, les Gammes Logiques peuvent se déduire naturellement. Mais cela peut nécessiter une approche méthodologique rigoureuse dans le cas de pièces complexes et notamment dans le cas d'assemblages mettant en oeuvre de multiples composants [Bourjault et al 1987] [Bourjault et al 1992].

Nous avons ici cinq gammes logiques élémentaires (une pour chaque type de pièce) qui se réduisent en trois gammes logiques principales du fait que les productions des pièces de type P3 et P4 soient uniquement dédiées à la production des pièces de type P5 par leur assemblage (Figure II.18).

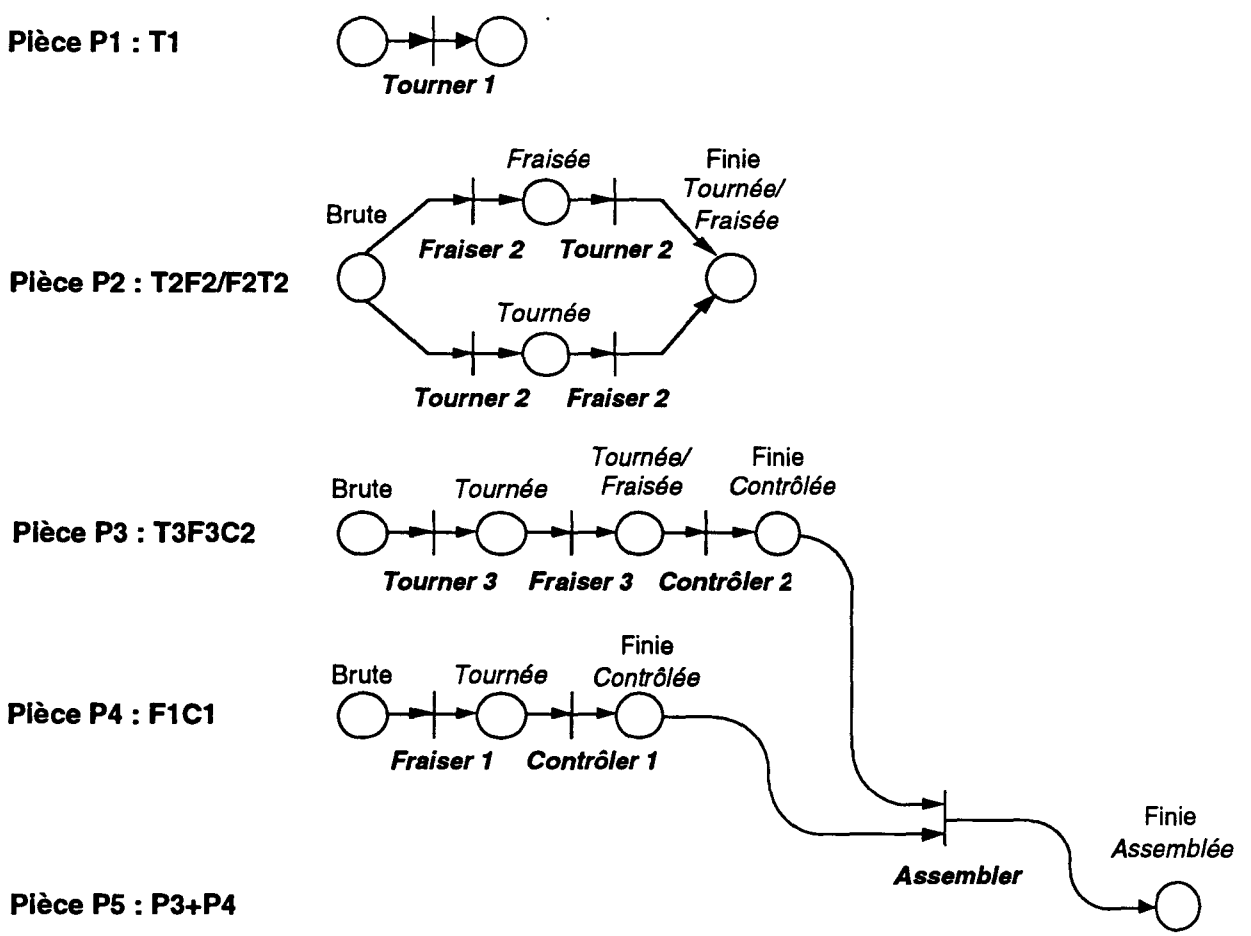


Figure II.18 : Ensemble des Gammes Logiques

II.E.4. Compatibilité entre opérations et machines

Au niveau des opérations à réaliser, il faut définir les compatibilités avec les machines présentes. En effet, même si deux machines ont la même fonction principale, il se peut qu'elles

ne puissent traiter les mêmes types de pièces. Elles peuvent différer, par exemple, par l'ensemble des outils de coupe présents dans leurs magasins d'outils, par la précision d'usinage, ou encore par le système de préhension des pièces, etc.

Ainsi, sur la cellule prise en exemple, les tournages de type t1, t2 et t3 peuvent être effectués indifféremment sur les tours T1 et T2, ceux-ci étant identiques. Le fraisage de type f1 peut être effectué indifféremment sur les centres d'usinage CU1 et CU2. Par contre, le fraisage de type f2 ne peut se faire que sur le centre d'usinage CU1 et le fraisage de type f3 ne peut se faire que sur le centre d'usinage CU2. Ces contraintes sont résumées dans le tableau suivant (Figure II.19) :

Opérations / *Machines*	Tourner 1	Tourner 2	Tourner 3	Fraiser 1	Fraiser 2	Fraiser 3	Ensemble des opérations
Tour 1	✓	✓	✓				t1+t2+t3
Tour 2	✓	✓	✓				t1+t2+t3
Centre d'usinage 1				✓	✓		f1+f2
Centre d'usinage 2				✓		✓	f1+f3
	T1⊕T2	T1⊕T2	T1⊕T2	CU1⊕CU2	CU1	CU2	

Figure II.19 : Compatibilités des Opérations vis à vis des Ressources Elémentaires

Pour ce qui est des contrôles c1 et c2, ils se déroulent uniquement sur le centre de métrologie ME1 et l'assemblage des pièces P3 et P4 s'effectue uniquement sur le poste d'assemblage ASS1.

A partir de ces données et afin de compléter plus précisément le champ 'Opérations Réalisables' des tables caractéristiques de chaque ressource virtuelle, nous pouvons en déduire les compatibilités de ces dernières vis à vis des opérations pour les différents niveaux d'agrégation (Figure II.20).

Au niveau de la cellule (agrégation de niveau N+3), nous retrouvons bien la totalité des fonctions de production disponibles.

	N+3		N+2		N+1		N0	
RESSOURCES / OPERATIONS	CELLULE	(t1 + t2 + t3). (f1 + f2). (f1 + f3). (Ass . (c1 + c2))	RVN2.1	(t1 + t2 + t3) .(f1 + f2)	RVN1.1	t1 + t2 + t3	T1	t1 + t2 + t3
							R1	
					RVN1.2	f1 + f2	CU1	f1 + f2
							R2	
			CONV					
			RVN2.2	(t1 + t2 + t3). (f1 + f3). (Ass . (c1 + c2))	RVN1.3	(t1 + t2 + t3). (f1 + f3)	T2	t1 + t2 + t3
							R3	
							CU2	f1 + f3
					RVN1.4	Ass . (c1 + c2)	ASS1	Ass
							R4	
							ME1	c1 + c2

Figure II.20 : Déduction des Compatibilités Opérations / Ressources Virtuelles

II.E.5. Les Gammes Opératoires

Les Gammes Logiques sont automatiquement converties en **Gammes Opératoires** grâce à une prise en compte de plus en plus fine des moyens de production puis des moyens de transport [Amar et al 1992].

Les gammes opératoires sont elles aussi basées sur les Réseaux de Petri où chaque état et chaque lieu sont représentés par les places et chaque opération ou changement de lieu sont représentés par les transitions [Cruette 1991].

Pour les systèmes manufacturiers de petite taille, la mise à plat des gammes opératoires est une bonne approche mais celle ci peut s'avérer très vite complexe pour des systèmes de plus grande taille ou du moins très flexibles [Toguyeni 1992][Ausfelder et al 1993] [Farah 1993].

Ainsi, pour notre cellule prise en exemple, il est préférable d'instaurer une hiérarchie au niveau des gammes opératoires. Pour notre part, nous avons introduit une hiérarchie basée sur les ressources virtuelles décrites précédemment.

Si l'on considère la démarche de génération automatique des gammes opératoires proposée par S.AMAR , la première étape consiste en l'intégration de l'aspect fonctionnel. Pour notre part, nous pouvons effectuer cette opération pour chacun des niveaux d'agrégation vus

précédemment (figures II.11 et II.14) en considérant les ressources virtuelles au même titre que les ressources réelles. En effet, au niveau fonctionnel, les ressources virtuelles disposent de l'ensemble des possibilités des ressources réelles ou virtuelles les composant. Pour intégrer cet aspect, nous nous référerons à la table des compatibilités Opérations/Ressources vue dans la partie précédente (figure II.20). Par exemple, pour les pièces de type P1, nous obtenons, l'ensemble décrit par la figure II.21.

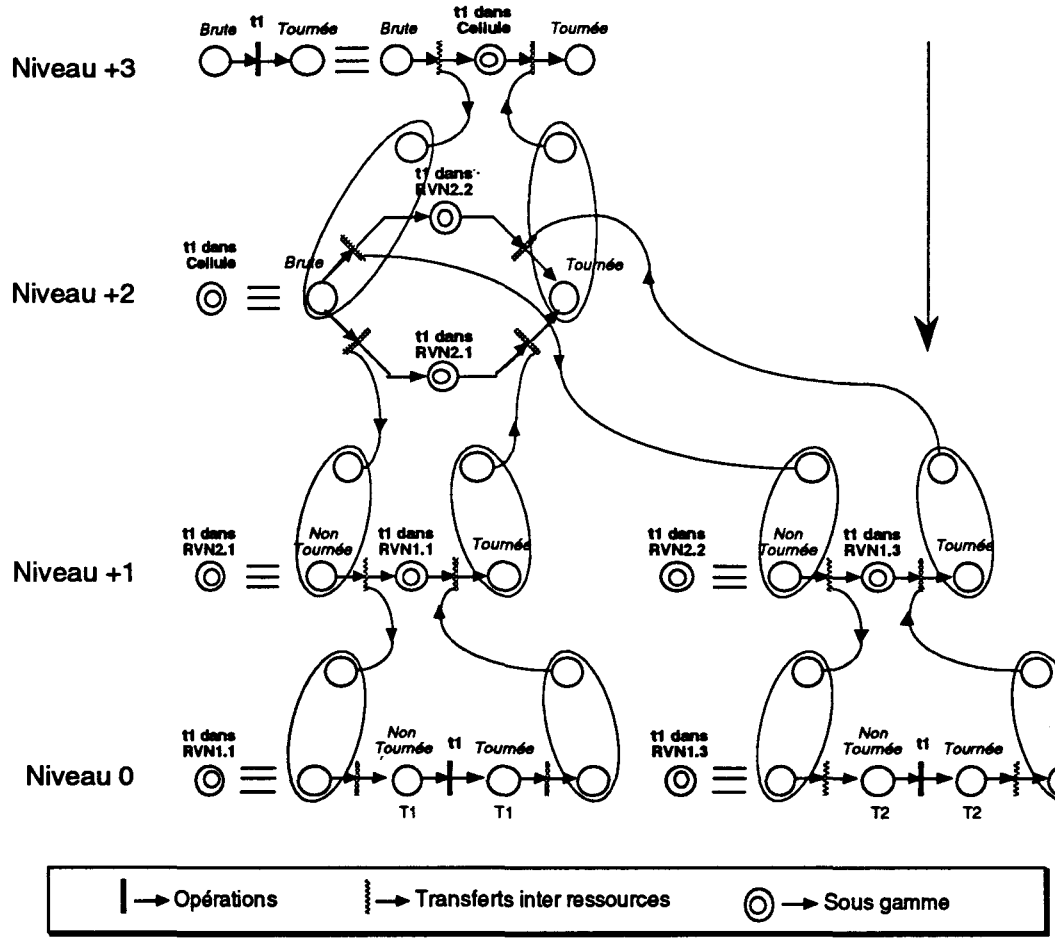


Figure II.21 : Première phase dans la génération des Gammes Opératoires pour les pièces de type P1

La seconde étape est l'intégration de l'aspect transitique. Dans un premier temps, pour les transferts au niveau des ressources virtuelles, nous nous contentons de spécifier les entrées et les sorties en nous référant aux tables caractéristiques des ressources virtuelles (Figure II.12).

Dans ces deux premières étapes, nous aurons une démarche descendante; nous partirons du niveau d'agrégation le plus élevé correspondant au niveau cellule pour terminer au niveau des ressources réelles, c'est à dire au niveau 0.

En respectant ces deux étapes, nous obtenons, pour les pièces de type P1, l'ensemble des Gammes Opératoires suivantes (Figure II.22):

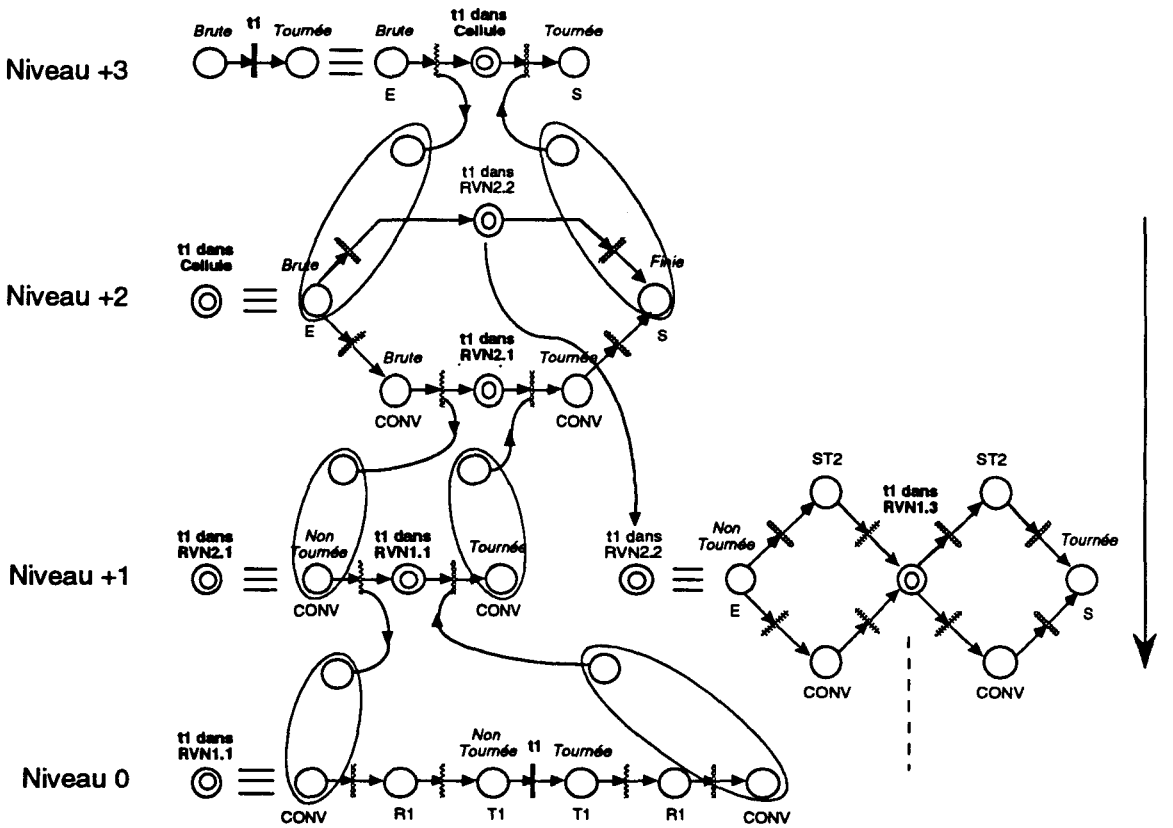


Figure II.22 : Seconde phase dans la génération des Gammes opératoires pour les pièces de type P1

Lors de cette génération, il est possible de restreindre le nombre de chemins empruntés par les pièces. C'est ce qui a été fait ici pour l'opération 't1 dans RVN1.1'. En effet, si l'on considère la table caractéristique de RVN1.1 (figure II.12), nous avons éliminé les chemin correspondant à l'entrée/sortie que constitue le stockeur ST2. Ce choix est à l'initiative du concepteur mais il peut aussi être déduit par une analyse des performances.

Nous remarquons qu'au niveau 0, nous retrouvons des gammes opératoires classiques correspondant à une approche non hiérarchisée [Huvenoit et al 1992].

Ensuite, il faut décrire l'ensemble des transferts restant en tenant compte des choix possibles. Par exemple, pour les pièces de type P1, le transfert de l'entrée E vers le convoyeur CONV peut s'effectuer suivant deux chemins différents internes à RVN2.2. Soit l'on reste dans RVN1.4, le transfert est alors direct par R4, soit l'on passe par RVN1.3 et il faut alors que la pièce soit échangée entre R4 et R3 par l'intermédiaire du stockeur ST2 (Figure II.23).

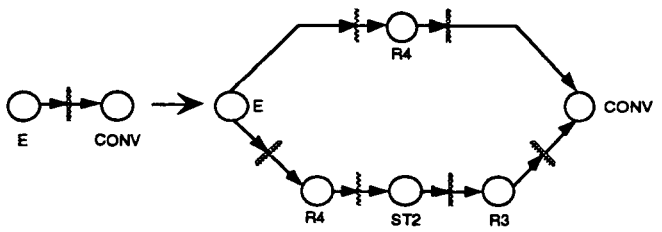


Figure II.23 : Choix entre deux chemins de transfert

La troisième et dernière étape correspond à la description des transferts internes aux ressources où l'on spécifie principalement les lieux de départ et les lieux d'arrivé.

Dans le but d'accroître la modularité et la réutilisabilité, nous pouvons considérer les transferts inter ressources vus précédemment comme étant des **macro gammes opératoires** (Figure II.24). Lorsque ces transferts seront présents dans d'autres gammes, il ne sera plus nécessaire de les détailler, il suffira d'y insérer la macro gamme correspondante.

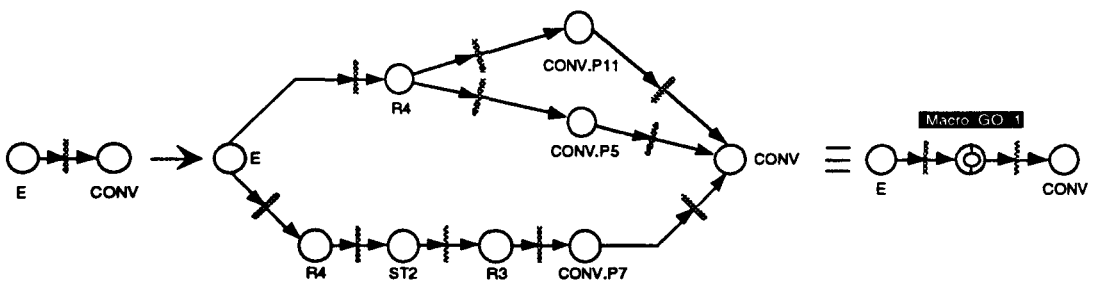


Figure II.24 : Macro Gamme Opératoire pour le transfert de l'entrée vers le convoyeur

Il est à remarquer qu'il n'est pas nécessaire de conserver tous les chemins possibles, certains chemins pouvant s'avérer superflus voir trop complexes ou dégradants vis à vis des performances. Pour la macro gamme opératoire ci dessus, il n'est pas primordial de conserver les deux chemins R4 vers CONV.P11 et R4 vers CONV.P5. On conservera ce dernier chemin, en effet, le poste P11 représente un goulot d'étranglement. Il faut donc éviter d'y palettiser/dépalettiser des pièces.

En tenant compte de toutes ces règles et remarques, nous obtenons finalement, pour les pièces de type P1, le résultat décrit par la figure II.25 (La sous gamme "t1 dans RVN2.2" n'a pas été développée par soucis de visibilité).

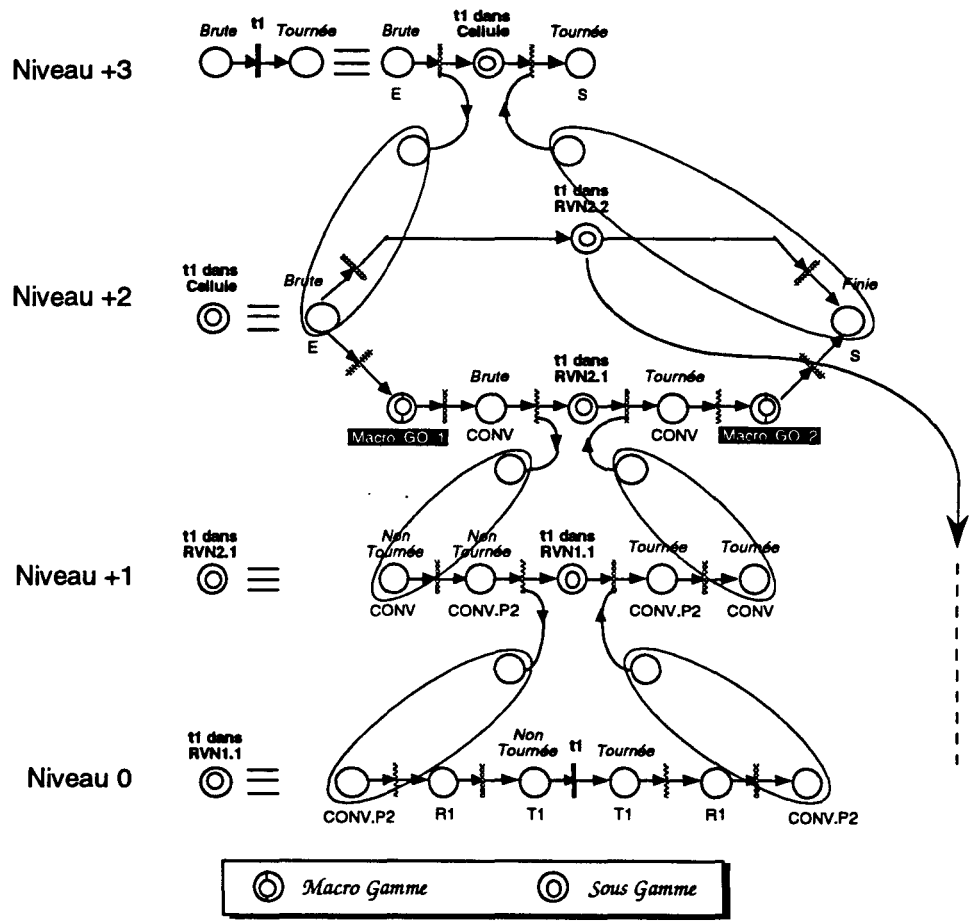


Figure II. 25 : Dernière phase dans la génération des Gammes Opératoires pour les pièces de type P1

Dans les Gammes Opératoires finales, les chemins internes aux ressources complexes telles que le convoyeur ne sont pas spécifiés, seuls sont spécifiés les lieux caractéristiques ou s'effectuent les palettisations/dépalettisations. Le routage des pièces à l'intérieur de ces ressources complexes s'effectue en temps réel et est réalisé par les objets de contrôle des ressources que nous aborderons ultérieurement et que nous appellerons filtres comportementaux (Partie II.F.3).

II.E.6. Les Gammes Opératoires Etendues

Les gammes opératoires décrites précédemment décrivent complètement le cheminement des pièces, mais elles ne sont pas directement exploitables sous cette forme. Comme nous avons pu le voir dans le § I, elles se situent à un niveau intermédiaire entre le Niveau Décisionnel Supérieur et le Pilotage des ressources de production (Figure I.13). Les gammes

opératoires vont donc devoir être enrichies pour pouvoir communiquer avec ces deux entités. Dans ce sens, nous avons introduit les **Gammes Opératoires Etendues** (GOE) basées sur les réseaux de Petri à structures de données[Huvenoit et al 1992][Huvenoit et al 1993].

Pour ces échanges inter niveaux et dans le but de conserver la structuration hiérarchisante, nous allons adjoindre aux gammes opératoires des protocoles de communication qui se feront dans tous les cas soit à l'aide du couple **Requête/Accusé de réception** (Req/Ack) simple ou multiple soit à l'aide du trio **Requête/Accusé de réception positif/Accusé de réception négatif** (Req/Ack/Nack) (Figure II.26). Dans l'optique d'une implantation effective, cette modélisation est garante d'une modularité nécessaire qui permettra, à terme, la distribution du code.

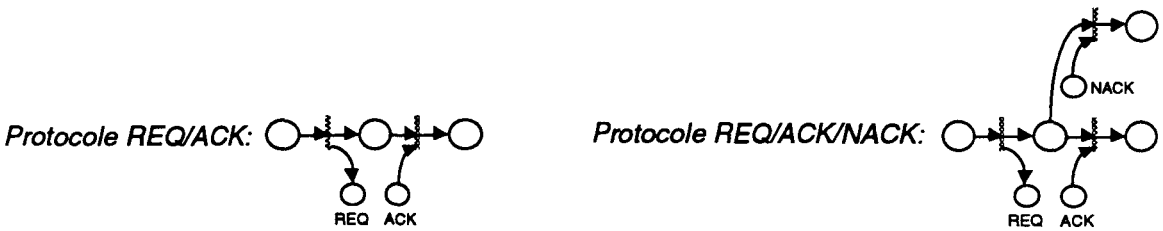


Figure II.26 : Protocoles de communication intégrés dans les Gammes Opératoires Etendues

Pour ce qui est des échanges avec le niveau décisionnel supérieur, il faut remarquer qu'ils sont liés à la présence de choix au niveau des gammes opératoires. Dans ce cas, une requête (REQ) est émise vers le module ou objet hiérarchique correspondant pour demander une résolution d'indéterminisme. Une fois le résultat obtenu, le module ou objet hiérarchique envoie un accusé de réception (ACK) vers la branche effectivement choisie.

Les appels des gammes de niveau inférieur se feront également à l'aide du couple Requête/Accusé de réception. Une requête émise par une gamme de niveau N permet d'activer une gamme de niveau N-1. Lorsque cette dernière est totalement terminée, elle renvoie un accusé de réception vers sa gamme appelante afin que celle ci puisse continuer son traitement.

Enfin, il faut ajouter des requêtes d'allocation permettant de contrôler l'accès aux ressources partagées. En effet, toutes les ressources réelles ou virtuelles possèdent une capacité d'accueil limitée et le nombre de pièces présentes dans une ressource additionné des pièces en attente de celle-ci ne doit pas dépasser cette capacité. Nous verrons plus en détail ce problème d'allocation dans la partie II.G.3.

Les requêtes d'allocation sont aussi basées sur les protocoles vus ci dessus. De plus, à chaque requête d'allocation correspondra une requête de désallocation qui sera émise une fois la pièce définitivement libérée par la ressource.

Suivant la position de la requête d'allocation, il sera possible de prendre en compte des opérations en temps masqué ou, au contraire, de ne débuter une opération que si toutes les ressources requises ont été allouées.

Ainsi, pour la macro gamme opératoire 1, nous obtenons la macro gamme opératoire étendue suivante (Figure II.27) (Dans cette figure, nous avons omis les flèches des protocoles de communication afin d'améliorer la lisibilité) :

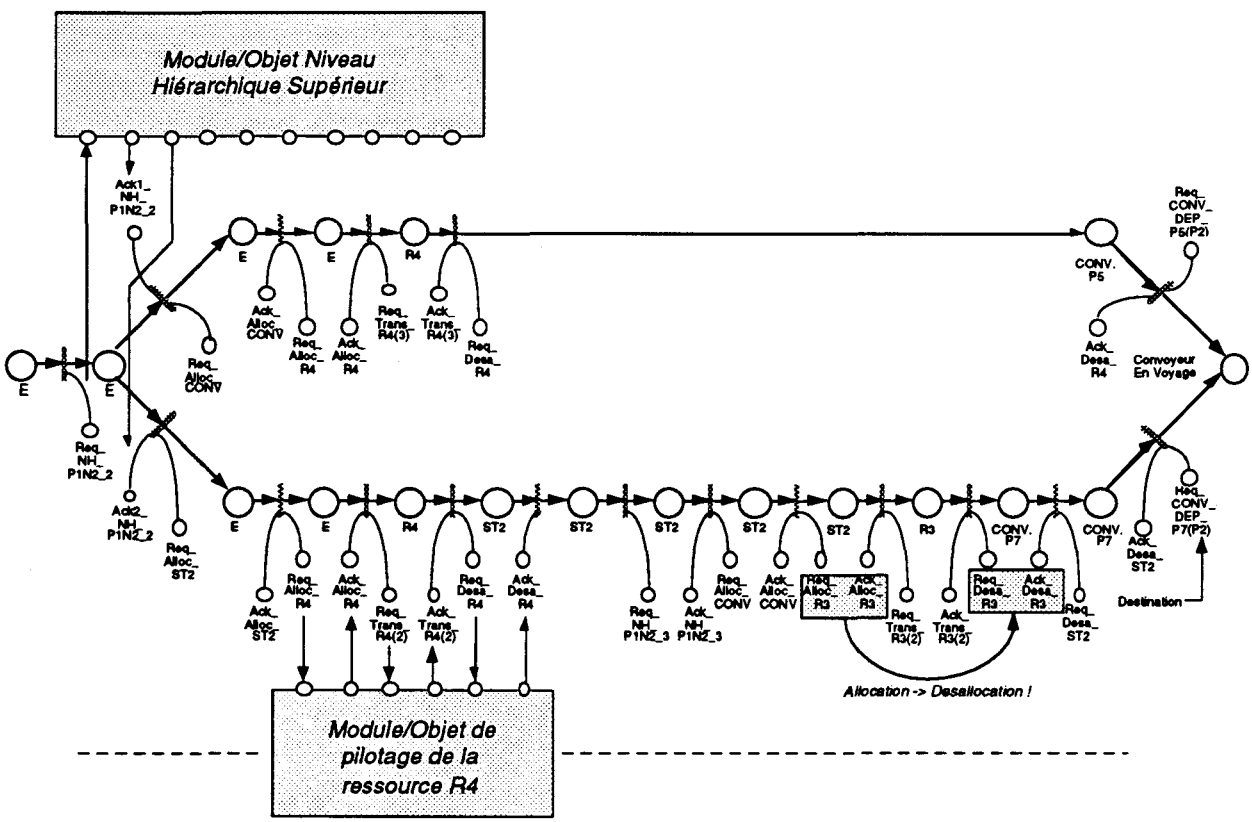


Figure II.27 : Macro Gamme Opératoire Etendue n°1 : Macro_GOE_1

Nous trouvons dans cette macro gamme deux appels au niveau hiérarchique supérieur. Le premier (P1N2_2) permet de décider si la pièce va prendre le chemin direct par l'intermédiaire du robot R4 ou si elle va transiter par le stockeur ST2. Ce dernier choix sera viable si la pièce n'est pas prioritaire ou si il y a saturation des demandes d'allocation du robot

R4. Le second appel permet de décider, en fonction de l'ordonnancement prévu, à quel moment la pièce sortira du stockeur ST2.

Il est à noter que pour la désallocation, l'ACK s'avère le plus souvent inutile mais il est nécessaire pour améliorer l'homogénéité dans la structure des gammes.

Ainsi, les gammes opératoires étendues sont composées, au plus, de trois types de structures de base qui sont **la séquence, l'alternative et le retour arrière** (Figure II.28).

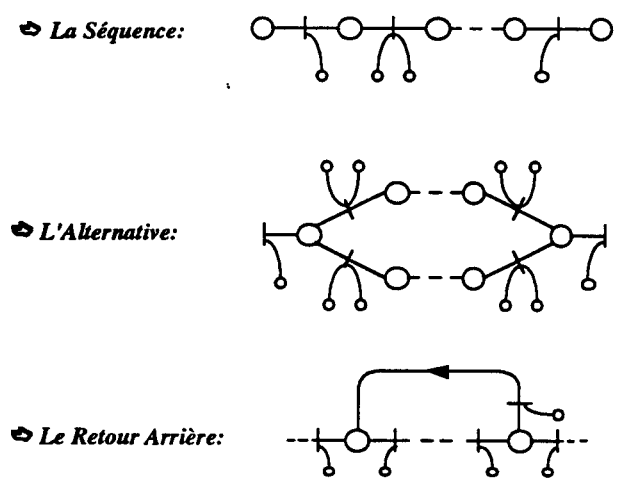


Figure II.28 : Structures de base présentes dans les gammes opératoires étendues.

Finalement, nous pouvons remarquer que la **séquence** est la structure minimale des GOE, on la trouve seule ou intégrée à d'autres structures.

L'**alternative**, qui peut être multiple, correspond soit à un Indéterminisme dû à la Flexibilité des opérations au sein des Produits, que nous dénommerons **IFP**, soit à un Indéterminisme dû au Choix entre Ressources de fonctionnalités identiques, que nous dénommerons **ICR**, soit à un Indéterminisme dû au Choix entre différents Chemins, que nous dénommerons **ICC**.

Le **retour arrière** est nécessaire dans les cas d'une stratégie de contrôle particulière ou de rejets d'allocations (Nack). Ces derniers peuvent apparaître, par exemple, à la suite d'une indisponibilité de ressource suite à une défaillance.

II.E.7. Gammes Opératoires Etendues et généricité

Nous avons déjà vu la notion de macro gammes et de sous gammes mais nous pouvons aller encore plus loin dans la modularité et la généricité. En effet, il y a peu de différence entre une gamme correspondant à l'opération "t1 dans RVN2.1" et une gamme correspondant à l'opération "t2 dans RVN2.1". La seule différence se situe au niveau du traitement (ici un usinage). On pourrait donc avoir une gamme générique pour effectuer l'opération "ti dans RVN2.1" avec ti $\in \{t1, t2, t3\}$ (Figure II.29). Cette gamme serait instanciée avec comme paramètre le traitement à effectuer.

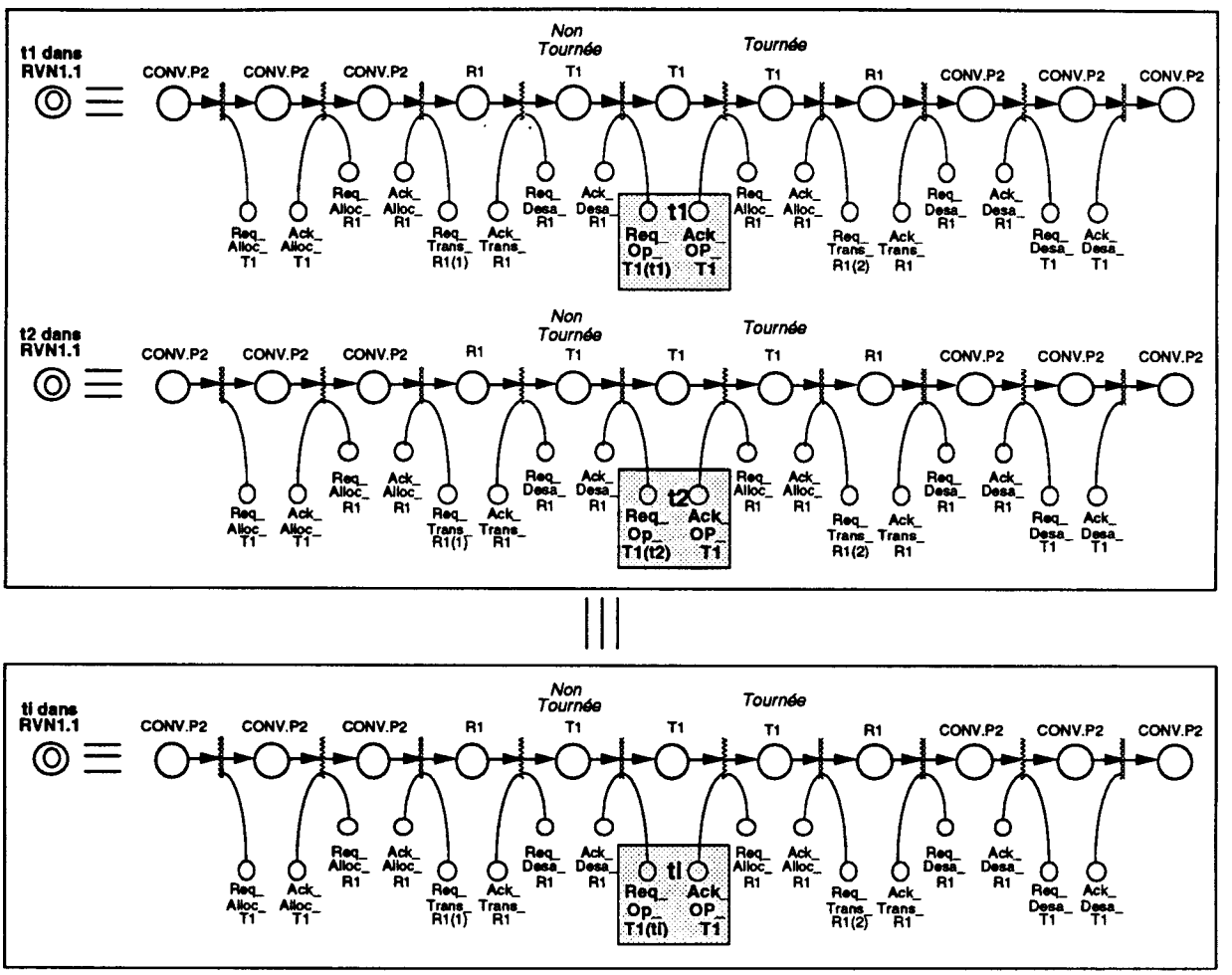


Figure II.29 : Agrégation de Gammes Opératoires Etendues

Il se peut que dans certains cas, le traitement ne soit pas la seule différence entre deux gammes qui semblent a priori similaires. En effet, les caractéristiques physiques des pièces peuvent imposer que telle ou telle pièce ne puisse être prise par tel ou tel robot car il ne dispose pas des préhenseurs requis pour leur manipulation. Dans ce cas, pour obtenir des gammes génériques, il est nécessaire d'agréger les différentes gammes en une seule qui en intègre toutes les possibilités.

La prise en compte effective de ces hypothèses de généricité permettra d'optimiser l'implantation (Cf chapitre III et annexe II).

II.F. Modélisation de la Partie Opérative

II.F.1. Introduction

Nous allons aborder dans cette partie la modélisation de la commande des ressources de production d'un SFPM. Pour ce qui est de notre niveau de détail, comme nous l'avons déjà vu, nous supposerons que les ressources achetées 'clé en main' sont opérationnelles et donc que les commandes fines ainsi que les fonctions standards ne sont plus à concevoir.

Comme nous avons pu le voir dans la partie II.D, nous pouvons distinguer deux types de ressources : les ressources dites **simples** et les ressources dites **complexes**.

Dans un premier temps, nous aborderons la modélisation des principaux types de ressources simples que l'on peut rencontrer dans un système de production manufacturier. Dans un second temps, nous nous attacherons à modéliser quelques exemples de ressources complexes. Dans les deux cas, nous insisterons sur les aspects relevant d'une approche orientée objets, à savoir la modularité et la généricité, intrinsèquement présents dans notre démarche.

II.F.2. Les Ressources Simples

A partir des définitions données dans la partie II.D, nous savons que les ressources simples sont mono service; pour les commander, il suffit de leur envoyer des ordres de commande élémentaire selon une séquence prédéterminée. Chaque ressource simple est alors considérée comme une entité fonctionnelle indépendante. De fait, ce type de ressource ne comporte pas d'indéterminisme au niveau de sa commande. Seul un éventuel indéterminisme découlant d'un choix de séquence d'opérations (choix du programme d'usinage pour les MOCN, choix du type de transfert pour les robots de manutention, choix de la séquence de mesure pour les centres de métrologie,...) subsiste.

Après une analyse de l'ensemble de ces ressources, nous pouvons dégager l'existance de deux types de coopération entre ressources. Ce sont d'une part la coopération sans synchronisation lors d'échange de pièces et d'autre part la coopération avec synchronisation lors d'échange de pièces. Ces comportements dépendent fortement de l'environnement immédiat et du contexte opérationnel des ressources considérées.

La coopération avec synchronisation apparait lorsqu'il y a une coopération physique forte, par l'intermédiaire de la (ou des) pièce(s), entre plusieurs ressources et plus particulièrement entre une ressource et ses ressources de chargement/déchargement. C'est le cas par exemple d'un tour et de son robot de chargement/déchargement où la pièce est maintenue dans le tour à l'aide de mors. En effet, lorsque le robot charge une pièce, il positionne celle ci entre les mors du tour puis doit attendre la fermeture de ces derniers avant de lâcher prise et de se dégager.

Le robot étant en attente de l'accusé de fermeture des mors, il y a bien synchronisation entre les deux ressources. Le tour est dans ce cas présent à l'initiative de la synchronisation. De même, la synchronisation duale se produit lors des déchargements et c'est alors le robot qui prend l'initiative.

En général, les caractéristiques fonctionnelles de l'ensemble des ressources sont bien spécifiées [Amar et al 1993]. De fait, pour une famille de ressources donnée ayant la même fonctionnalité principale, il est facile de dégager des ensembles de sous familles fonctionnelles à chacune desquelles correspondra un graphe de commande générique que nous avons appelé **Graphe de Commande Partiel** (GCP) [Huvenoit et al 1994][Huvenoit et al 1995].

Ainsi, l'analyse de la totalité des diverses familles de ressources simples composant les SFPM étudiés va nous permettre de construire une bibliothèque de graphes de commande partiels génériques (réutilisables).

Ensuite, nous pourrons accélérer et optimiser les phases de conception et d'implantation des **Graphes de Commande Complets** (GCC), ceux ci intégrant la totalité de la commande des ressources considérées [Huvenoit et al 1994][Huvenoit et al 1995]. Ces graphes de commande complets seront obtenus par agrégation de graphes de commande partiels découlant de l'instanciation des modèles génériques.

Bien sûr, à chaque étude de nouveaux sites industriels, nous étendrons notre connaissance par l'étude de nouveaux types de ressources. Ce qui nous permettra d'enrichir la **bibliothèque de Graphes de Commande Partiels.**

II.F.2.a. Les ressources simples de transport

Ces ressources seront généralement du type robot de manutention dont la fonction principale sera le plus souvent le chargement/déchargement de pièces. Ainsi, dans la cellule

prise en exemple, il y a quatre ressources simples de transport : les robots de manutention R1, R2, R3 et R4.

Pour cette **famille de ressource**, nous pouvons dégager une **classification comportementale** des divers types de transfert pouvant être réalisés suivant leurs caractéristiques essentielles (Figure II.30).

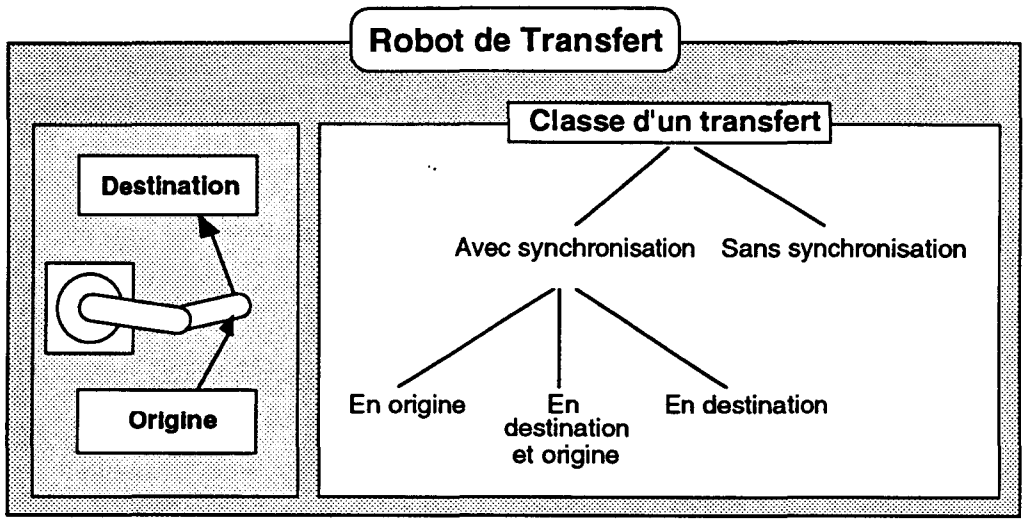


Figure II.30 : Classification comportementale des types de transfert pouvant être effectués par les robots de manutention

Les synchronisations peuvent se faire en origine lors du déchargement ou en destination lors du chargement des pièces. Il est également possible d'avoir une synchronisation entre deux robots de manutention.

A partir de cette classification , nous pouvons dégager l'existence de classes de graphes de commande partiels correspondant chacun à des comportements élémentaires.

Nous pouvons représenter chacune de ces classes en décrivant, d'une part, la partie comportement et d'autre part la partie attributs.

La partie comportement des graphes de commande partiels sera un graphe construit à l'aide de Réseaux de Petri. Celui-ci sera toujours du type **graphe d'état** car il représente un comportement mono fonctionnel.

Ainsi, nous obtenons les classes de graphes de commande partiels correspondant aux différents types de transferts élémentaires caractérisés par le type de synchronisation qu'ils mettent en oeuvre décrits par la figure II.31.

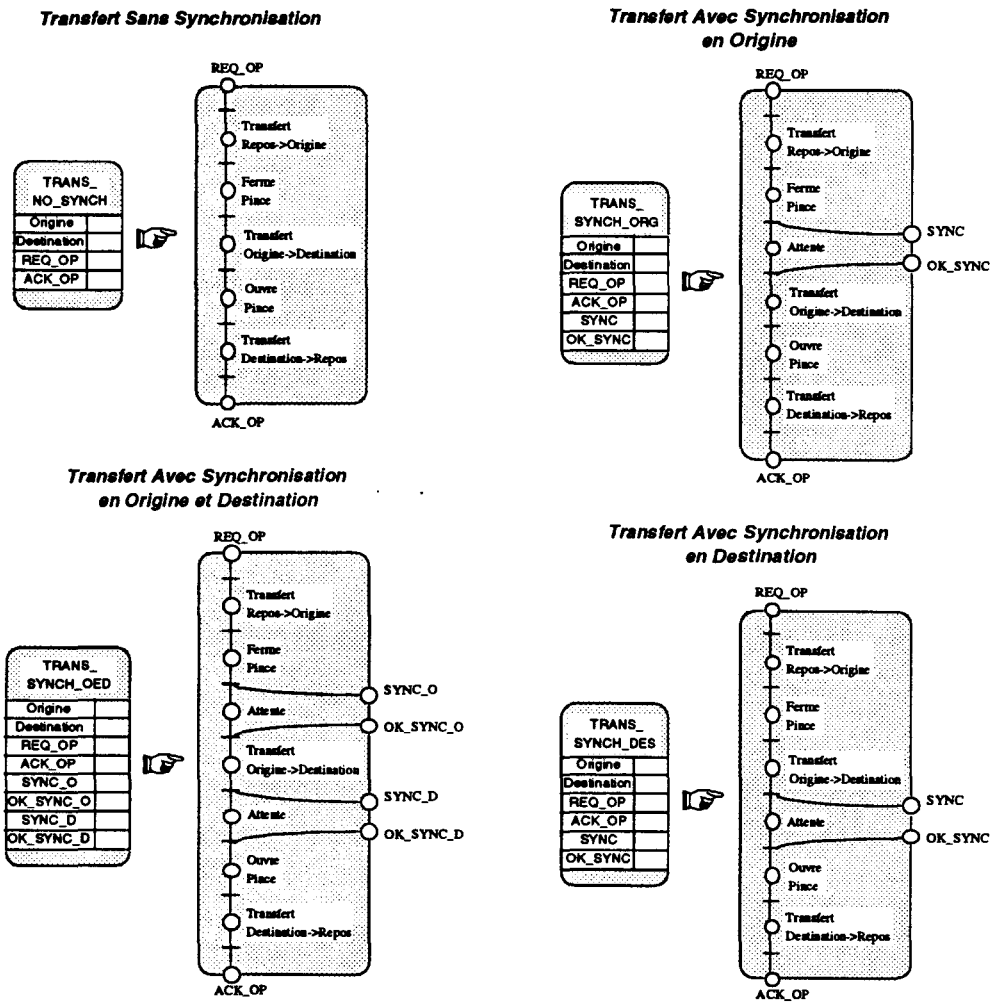


Figure II.31 : Classes de graphes de commande partiels pour les transfert d'un robot de manutention

Les synchronisations sont similaires à la synchronisation avec accusé de réception (SYNC/ACK) que l'on pouvait trouver dans les modèles de CASPAIM I [Bourey 1988] [Craye 1989].

Il est à noter que la position de repos correspond à une position quelconque dans le volume où le robot n'a aucune interaction possible avec son environnement.

Il faut ensuite tenir compte des contraintes annexes permettant le bon déroulement de ces graphes de commande partiels. Pour les robots de manutention, il faut voir si ceux-ci intègrent le changement de préhenseur. Dans le cas positif et avant toute manipulation, il faut sélectionner comme préhenseur courant, celui correspondant à la pièce à transférer. Pour assurer ce rôle, il est nécessaire de définir une classe de graphe de commande partiel correspondant au changement de préhenseur. Celle-ci est décrite par la figure II.32.

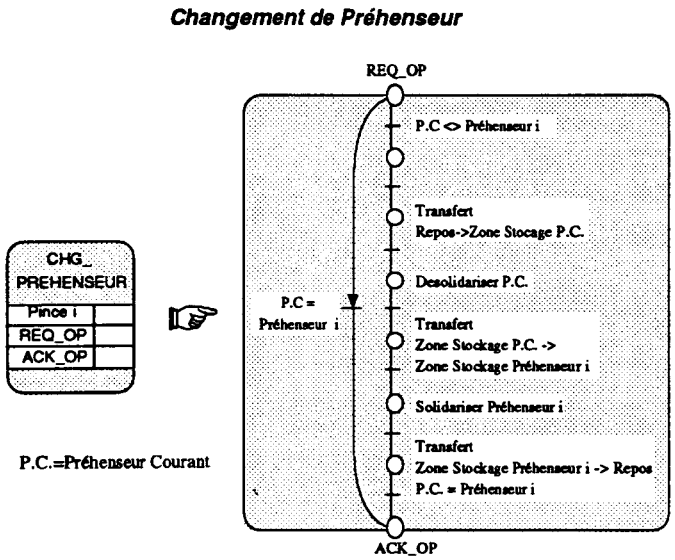


Figure II.32 : Classe de graphe de commande partiel pour un changement de préhenseur

Si le Préhenseur Courant (P.C.) est différent du préhenseur requis, le robot dépose celui-ci dans sa zone de stockage (magasin de préhenseurs) puis s'arme du nouveau préhenseur.

A la bibliothèque de base des graphes de commande partiels, il faut ajouter un ensemble d'éléments associatifs et/ou sélecteurs (Figure II.33) permettant de construire les graphes de commande complets. Ces éléments sont conçus dans l'esprit d'une programmation structurée ou l'objectif est que chaque graphe possède de bonnes propriétés de vivacité et de non blocage.

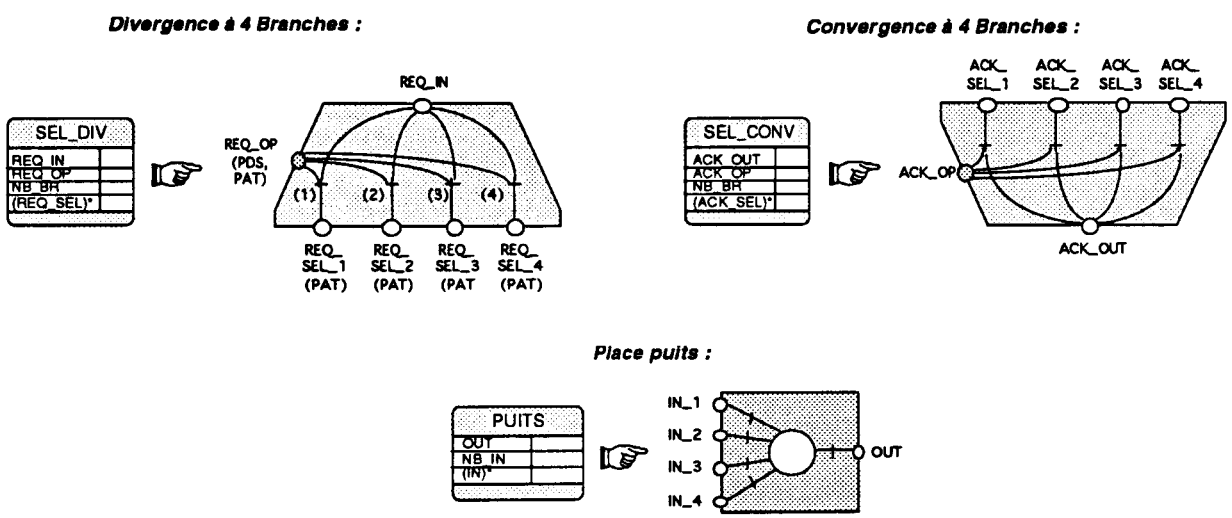


Figure II.33 : Eléments constructeurs

Le module "divergence" permet ici la sélection du transfert à effectuer mais il pourra aussi être utilisé lors de la construction de graphes de commande complets de ressources différentes des robots de manutention. La sélection s'effectue à partir du numéro d'identification transféré sous forme d'une couleur (Paramètre De Sélection=PDS=1..nbre_branches) lors d'une requête de service à effectuer par la ressource considérée. Accessoirement, d'autres paramètres peuvent bien sûr être transmis tels le type de pince (Paramètres Auxiliaires à Transférer=PAT).

Le module "convergence" est dual du module "divergence" et si cela s'avère nécessaire, il est possible de retourner des informations par l'intermédiaire de l'accusé de réception ACK_OP.

La place puits est une restriction de la convergence, elle permet de rassembler des places disjointes ayant la même fonction.

Nous disposons alors de tous les éléments pour construire le graphe de commande complet d'un quelconque robot de manutention par instanciation de ces classes puis par composition des graphes de commande partiels obtenus.

Appliquons la démarche au robot de manutention R1. Si nous considérons les relations d'accessibilité liées à celui-ci, nous pouvons en déduire, par transitivité, l'ensemble des transferts qu'il sera amené à effectuer (Figure II.34). Les valeurs Delta1, Delta 2 et Delta 3 correspondent à trois déplacements latéraux face au stockeur ST1. En effet, ce dernier possède trois emplacements de stockage par étagère (IO1, IO2, IO3) auxquels le robot R1 doit pouvoir accéder (cf figure II.16).

Etant donné qu'il s'agit d'un robot programmable par langage spécialisé et non d'un robot programmable par apprentissage, le nombre des types de transferts élémentaires à définir, ou **transferts de base minimaux**, se réduit à 6. En effet, les valeurs Delta1, Delta2 et Delta3 se réduisent à une unique variable Delta pouvant être prises en compte comme paramètre dans le programme robot. Les transferts restants ne peuvent être réduits car étant fondamentalement différents.

Transferts à effectuer	Synchro. en Origine	Synchro. en Destination	Sans Synchro.	Transferts de base minimaux
P2->ST1+Delta1			✓	P2->ST1+Delta
P2->ST1+Delta2			✓	
P2->ST1+Delta3			✓	
ST1+Delta1->P2			✓	ST1+Delta->P2
ST1+Delta2->P2			✓	
ST1+Delta3->P2			✓	
T1->P2	✓			T1->P2
P2->T1		✓		P2->T1
T1->ST1+Delta1	✓			T1->ST1+Delta
T1->ST1+Delta2	✓			
T1->ST1+Delta3	✓			
ST1+Delta1->T1		✓		ST1+Delta->T1
ST1+Delta2->T1		✓		
ST1+Delta3->T1		✓		

Figure II.34 : Ensemble des services à fournir par le robot R1

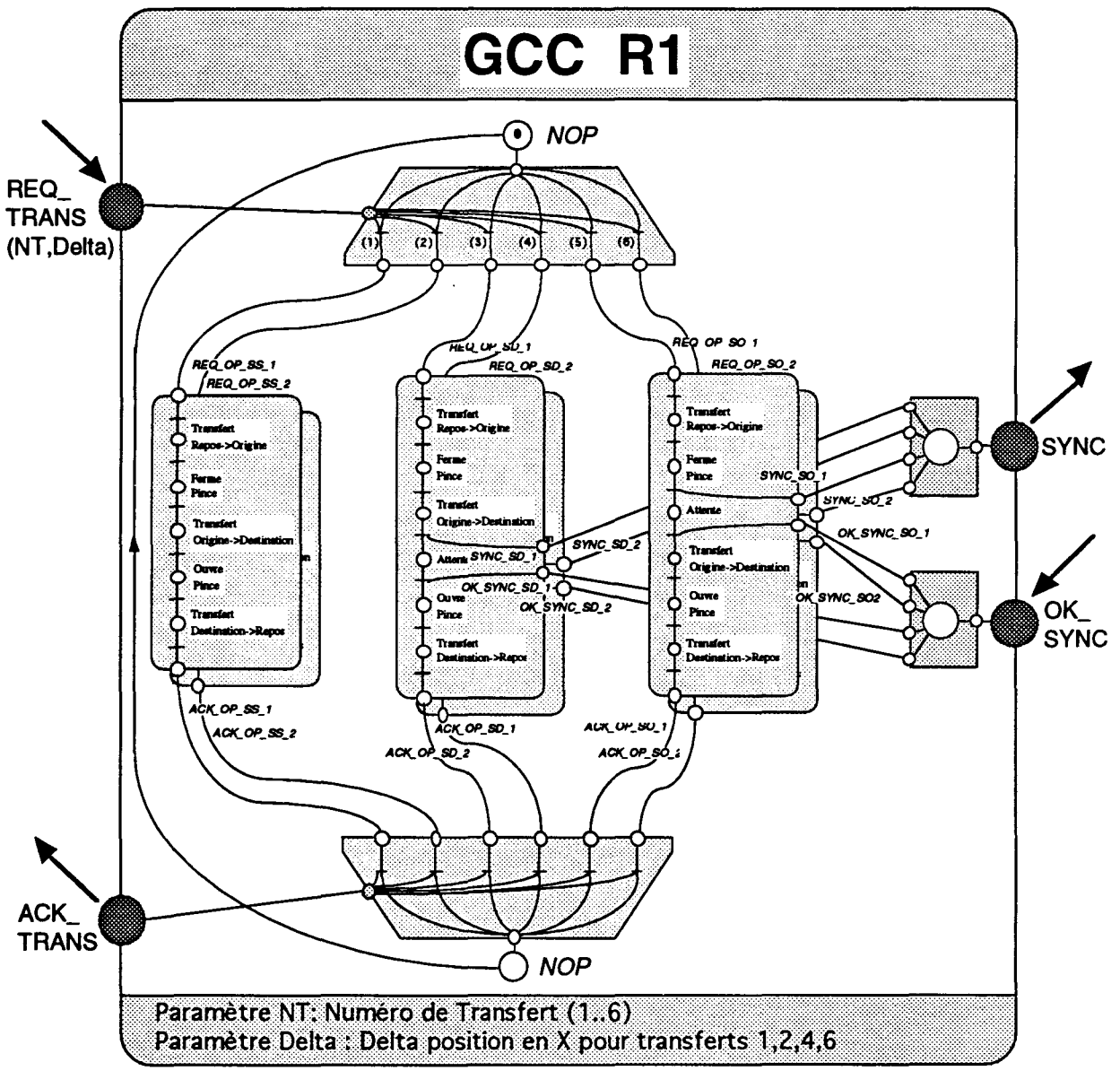
Pour ce robot, nous pouvons dégager l'existence de deux transferts sans synchronisation, de deux transferts avec synchronisation en origine et enfin de deux transferts avec synchronisation en destination. Les synchronisations avec le tour T1 sont nécessaires car il y a coopération entre la pince du robot et les mors du tour pour le chargement de toute pièce à tourner (synchronisation en destination pour R1) ou pour le déchargement de toute pièce tournée (synchronisation en origine pour R1).

L'association des graphes de commande partiels correspondant et des éléments constructeurs nous donne finalement le graphe de commande complet de la figure II.35.

Par soucis de visibilité, les places d'entrée et de sortie des GCP sont dédoublées (Les liens inter GCP sont virtuels). En réalité, une place de sortie d'un GCP correspondant à une place d'entrée d'un autre GCP sont une unique et même place.

Lors d'une requête de service à effectuer par le robot R1, nous transmettons le minimum d'informations, à savoir le numéro caractérisant le transfert à réaliser (NT) ainsi que le paramètre de décalage (Delta) nécessaire pour les transferts 1, 2, 4 et 6. Bien sûr, pour les transferts sans décalage, ce dernier paramètre est non significatif mais sa signature est

conservée pour assurer une cohérence dans la communication et donc un faible couplage entre les différents objets [Meyer 1988]. De plus, la minimisation de la quantité d'information transmise sera un point positif pour une implantation répartie.



Place Puits
Equivalent à:

Transfert N°	Origine	Destination
(1)	P2	ST1+Delta
(2)	ST1+Delta	P2
(3)	P2	T1
(4)	ST1+Delta	T1
(5)	T1	P2
(6)	T1	ST1+Delta

Figure II.35 : Graphe de commande complet du robot R1

Dans le même but, l'ensemble des places de synchronisation sortantes (SYNC) sont regroupées en une place puits de même que l'ensemble des synchronisations rentrantes (OK_SYNC). Il n'est pas nécessaire de les séparer car le robot est mono service (ressource simple).

Parallèlement, nous pouvons associer à la description comportementale du graphe de commande complet une description objet (Figure II.36) ou nous trouvons instanciées toutes les classes de graphes de commande partiels entrant dans la composition de celui-ci.

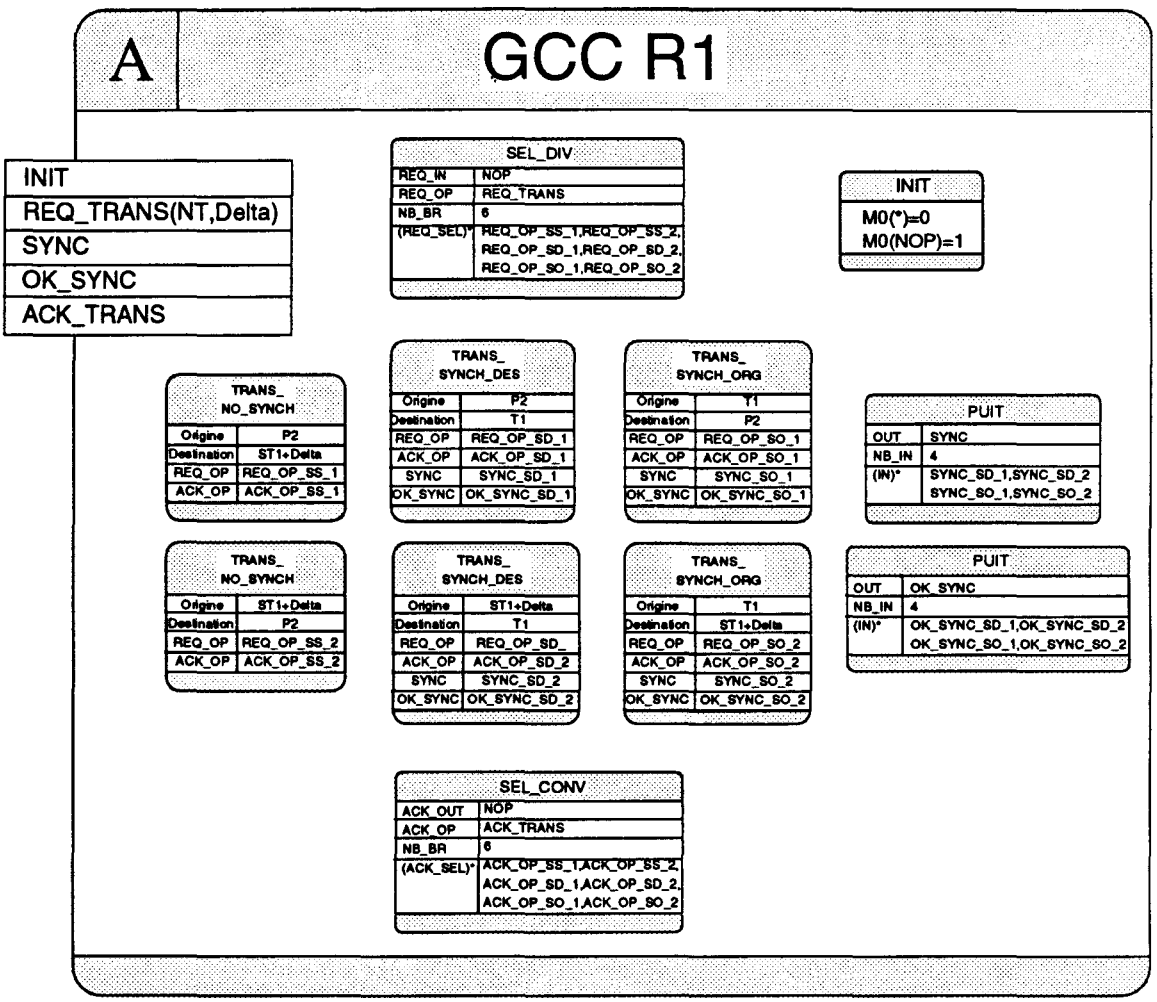


Figure II.36 : Représentation objet du GCC du Robot R1

Pour cette représentation graphique, nous avons utilisé le formalisme de la méthode HOOD (Hiérarchical Object Oriented Design) qui permet de mette en évidence les relations de composition ainsi que les interfaces fournies [HOOD 1992] [Paludetto 1992]. Par soucis de lisibilité, nous n'avons pas représenté les relations de type "Implemented by". Le module INIT permet l'initialisation de la structure obtenue.

Nous pouvons appliquer la même démarche aux robot R2, R3 et R4. Pour les robots R2 et R4, il faut tenir compte du fait que ce sont des robots programmables par apprentissage, il sera donc nécessaire de décrire l'ensemble des transferts à effectuer sans agrégation possible (impossibilité d'avoir des paramètres de décalage variables). De plus, pour les robots R2 et R3, il ne faut pas oublier d'adjoindre le graphe de commande partiel correspondant au changement de pince.

II.F.2.b. Les ressources simples de transformation

Nous mettons dans la catégorie des ressources simples dites de transformation, en supposant bien entendu qu'elles soient effectivement de type simple,

- ✔ celles qui usinent les produits pour leur enlever de la matière tels
  - les tours,
  - les centres d'usinages,
  - les rectifieuses,
  - les fraiseuses,
  - les perceuses,
  - ...

- ✔ celles qui modifient l' état surfacique des produits tels
  - les machines à laver,
  - les machines à sabler,
  - ...

- ✔ celles qui ajoutent aux produits des composants externes, ceux ci étant disponibles en quantité supposée illimitée dans la ressource elle même tels
  - les robots graisseurs,
  - les robots de soudage,
  - les ajouts de vis, de caches,...
  - les étiquetages,
  - ...

Pour ces ressources simples de transformation, le graphe de commande partiel de la phase opératoire se résume à celui de la figure II.37.

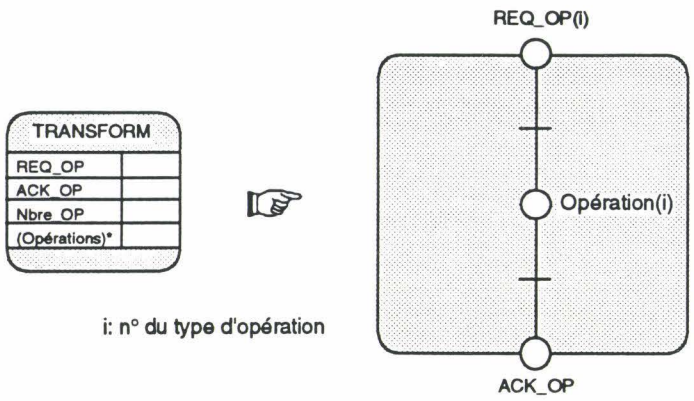


Figure II.37 : Graphe de commande partiel pour une transformation simple

Le type d'opération à effectuer correspond typiquement à un programme d'usinage qu'il faut charger (si il n'est pas dans la mémoire programme), sélectionner puis lancer. Cette information est transmise par l'intermédiaire de la couleur (i).

Pour le centre d'usinage CU2, nous obtenons, après composition, le graphe de commande complet est décrit par la figure II.38.

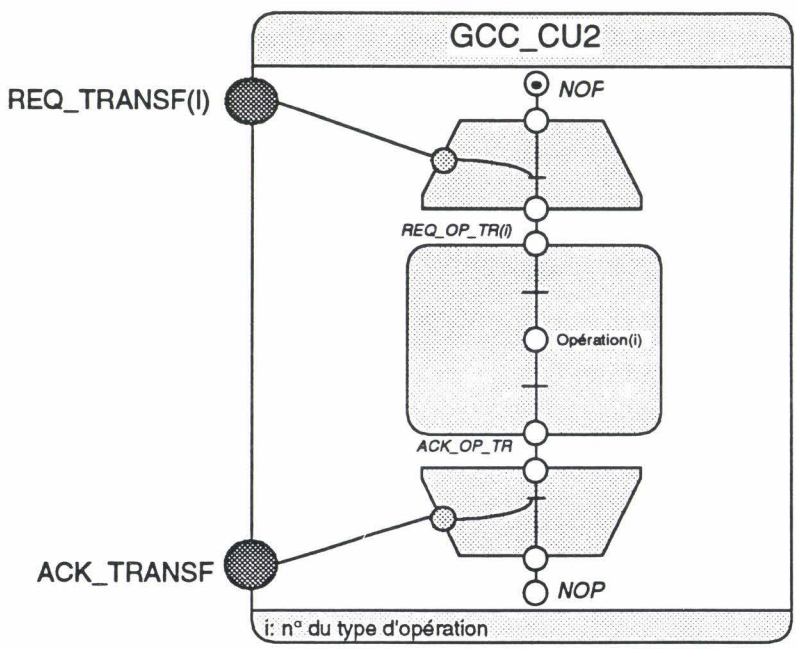


Figure II.38 : Graphe de commande complet du centre d'usinage CU2

Comme nous avons pu le voir, lors du chargement/déchargement des ressources de transformation, il est souvent nécessaire d'établir une synchronisation en chargement et/ou en déchargement avec la ressource dédiée à ces opérations. Nous avons donc tout naturellement un graphe de commande partiel (Figure II.39) associé à cette fonction.

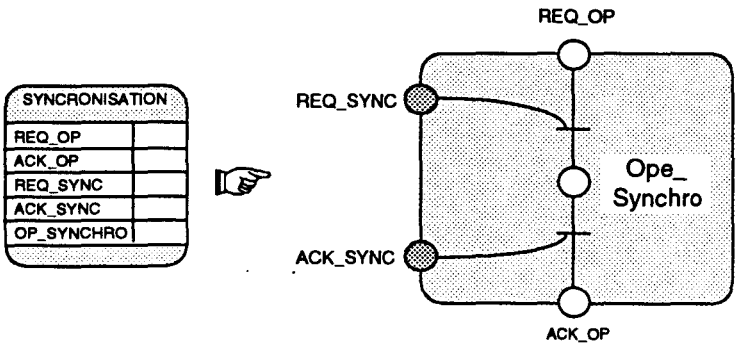


Figure II.39 : Graphe de commande partiel pour une synchronisation entrante

Par rapport à la synchronisation mise en oeuvre pour les robots de manutention où nous avions une synchronisation sortante, nous avons ici une synchronisation entrante. Les synchronisations sortantes sont utilisées pour les ressources actives lors des relations d'accessibilités tandis que les synchronisations entrantes sont utilisées pour les ressources passives

Pour les tours T1 et T2, ceux-ci étant identiques (Tours de type A), nous aurons les deux mêmes graphes de commande complets; nous pouvons donc les réunir en une seule classe qui pourra être instanciée selon les besoins.

Dans ce type de ressource, la pièce est maintenue à l'aide de mors, il doit donc y avoir à la fois une synchronisation en chargement et une synchronisation en déchargement. Pour intégrer cette caractéristique, nous ajoutons au graphe de commande de base, qui est dans ce cas identique à celui du centre d'usinage CU2, des graphes de commande partiels de synchronisation en amont et en aval. Finalement, nous obtenons la classe de graphe de commande complet de la figure II.40.

Parallèlement à cette description sous forme de réseaux de Petri, nous pouvons représenter cette classe d'objets sous forme d'objet HOOD . Celui-ci est décrit par la figure II.41 dans laquelle nous avons précisé les flux d'informations.

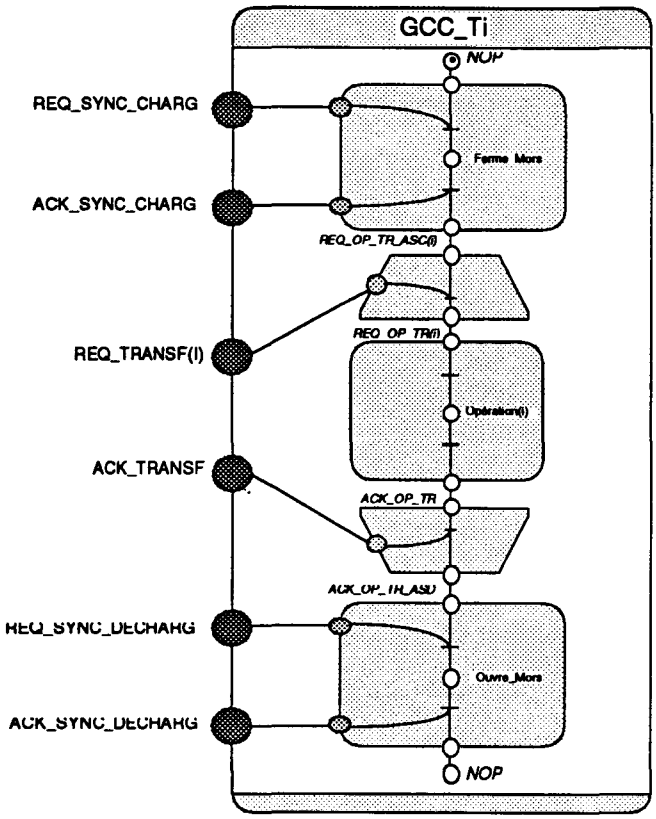


Figure II.40 : Graphe de commande complet pour les Tours de type A (T1 et T2)

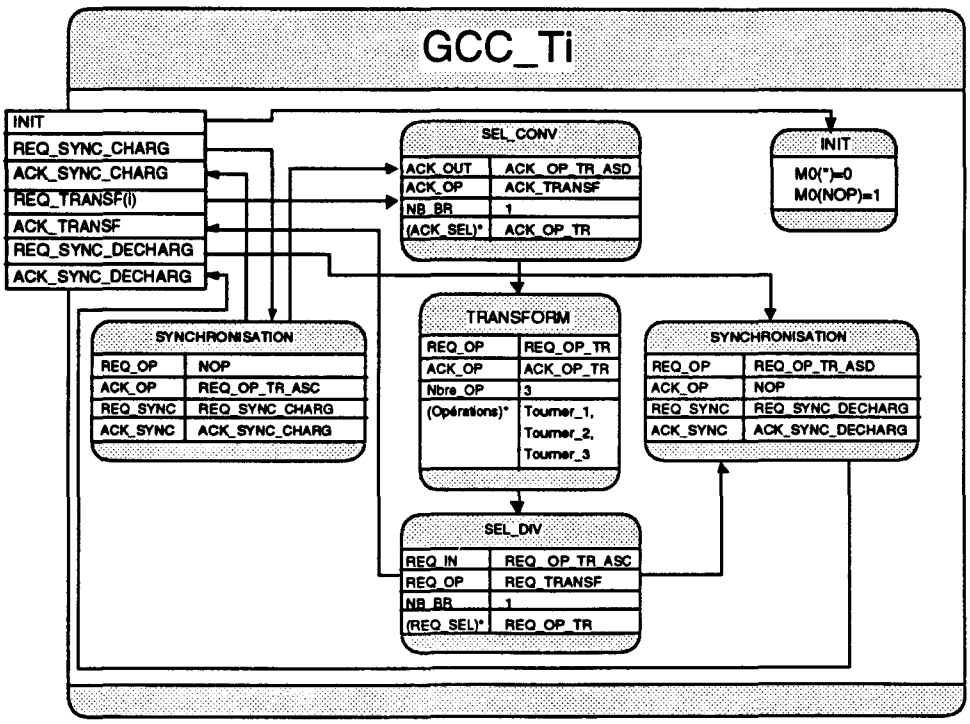


Figure II.41 : Représentation objet du graphe de commande complet des tours de type A (T1 et T2)

Dans la plupart des cas, les graphes de commande partiels présentés ici seront suffisant à l'élaboration de la commande des ressources simples de transformation que l'on trouve habituellement dans les ateliers de production manufacturière.

II.F.2.c. Les ressources simples de métrologie

Nous mettons dans cette catégorie de ressource toutes les ressources de contrôle des produits manufacturés tels que les centres de métrologie tridimensionnels, les systèmes de vision,... La fonction de ces ressources est de contrôler un produit fini ou partiellement fini pour décider si les critères de qualité ont été respectés lors de son élaboration.

Lors de la mise en oeuvre de ces ressources, il y aura généralement deux enchaînements possibles : le premier correspondant à la validation du produit et le second correspondant au rejet ou au recyclage du produit. Le critère de sélection sera une **information obligatoirement retournée** par la ressource de métrologie. Nous obtenons alors le graphe de commande partiel de la figure II.42.

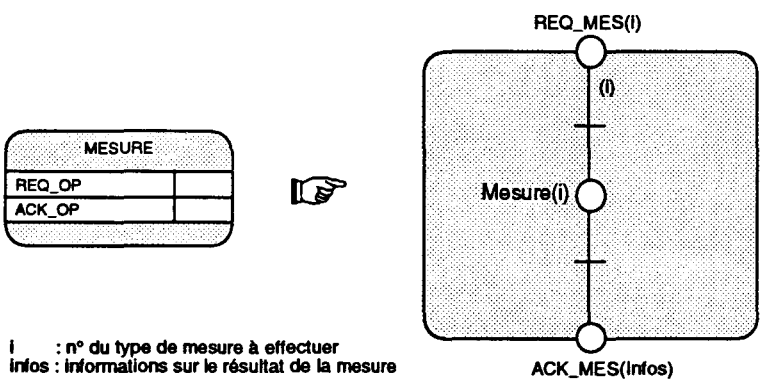


Figure II.42 : Graphe de commande partiel pour une opération de métrologie

Naturellement, nous retrouvons un graphe de commande partiel similaire à celui correspondant à une transformation simple mais avec un retour d'informations. Ces dernières peuvent, par exemple, décrire les écarts constatés sur tout ou partie du produit par rapport au modèle de mesure téléchargé ou présent dans la machine. En plus de leur fonction primaire qui est de permettre le contrôle qualité, ces informations peuvent également servir au niveau de la surveillance de l'usure des outils de coupe ou pour détecter les pannes ou dégradations parmi les ressources ayant participées au traitement du produit [Toguyeni 1993].

II.F.3. Les Ressources Complexes

Au vue des définitions de la partie II.D.2, nous pouvons déduire que dans la cellule prise comme exemple, les ressources complexes sont au nombre de cinq : le convoyeur central, les deux stockeurs ST1 et ST2, le centre d'usinage CU1 et enfin le poste d'assemblage ASS1.

Pour gérer ce type de ressource, nous avons besoin de **graphes de commande** pour contrôler les diverses **composantes commandables**, celle-ci ayant été mises en évidence lors de l'analyse descendante du procédé. Ce sont, par exemple, les aiguillages et les butées dans le cas du convoyeurs de palettes ou le poste opératoire dans le cas du centre d'usinage CU1. Mais nous avons également besoin de **filtres comportementaux** pour que les diverses sollicitations de ces graphes de commande restent cohérentes avec l'état global de la ressource [Morel et al 1990].

Chaque filtre comportemental intègre une description du comportement logique de la ressource et reflète donc la dynamique d'évolution de celle-ci. Dans la plupart des cas, ces filtres comportementaux intégrerons un ou plusieurs résolveurs d'***indéterminismes purement locaux*** (type IPL) afin de résoudre les choix résultant de la flexibilité interne aux ressources. Un convoyeur, par exemple, en possède de nombreux : au niveau de chaque aiguillage, il convient de décider de la direction à prendre pour chaque palette.

II.F.3.a. Les ressources de transport complexes

Nous mettons dans cette famille de ressource celles qui permettent le transfert simultané de plusieurs produits. Ce sont, par exemple, les convoyeurs de palettes utilisés principalement dans le cas des transports intra-cellules, les réseaux de chariots filo-guidés principalement utilisés dans les grands ateliers pour les transports inter-cellules.

Pour les ressources de transport complexes, a priori, seul deux types de graphes de commande sont nécessaires vis a vis des éléments transporteurs :

- celui permettant la commande d'avancement/arrêt,
- celui permettant la commande de direction.

Pour un système de convoyage à palettes, cela correspond respectivement à la commande des butées et à la commande des aiguillages. Pour un système de convoyage à chariots filo-guidés, cela correspond à la commande du chariot automoteur lui même.

Ces graphes de commande vont dépendre fortement du type de matériel mis en oeuvre. Pour les construire, nous devons nous référer aux automates d'états décrivant le comportement des objets commandables [Elkhattabi 1993] ainsi qu'aux graphes fonctionnels décrivant leur interaction avec les objets capteurs.

Si nous considérons les butées et les aiguillages du convoyeur, les automates d'états et les graphes fonctionnels ont été décrit lors de l'analyse du procédé dans la partie II.D.3.b (figures II.5 et II.7). Les graphes de commande correspondant sont alors ceux décrits par la figure II.43.

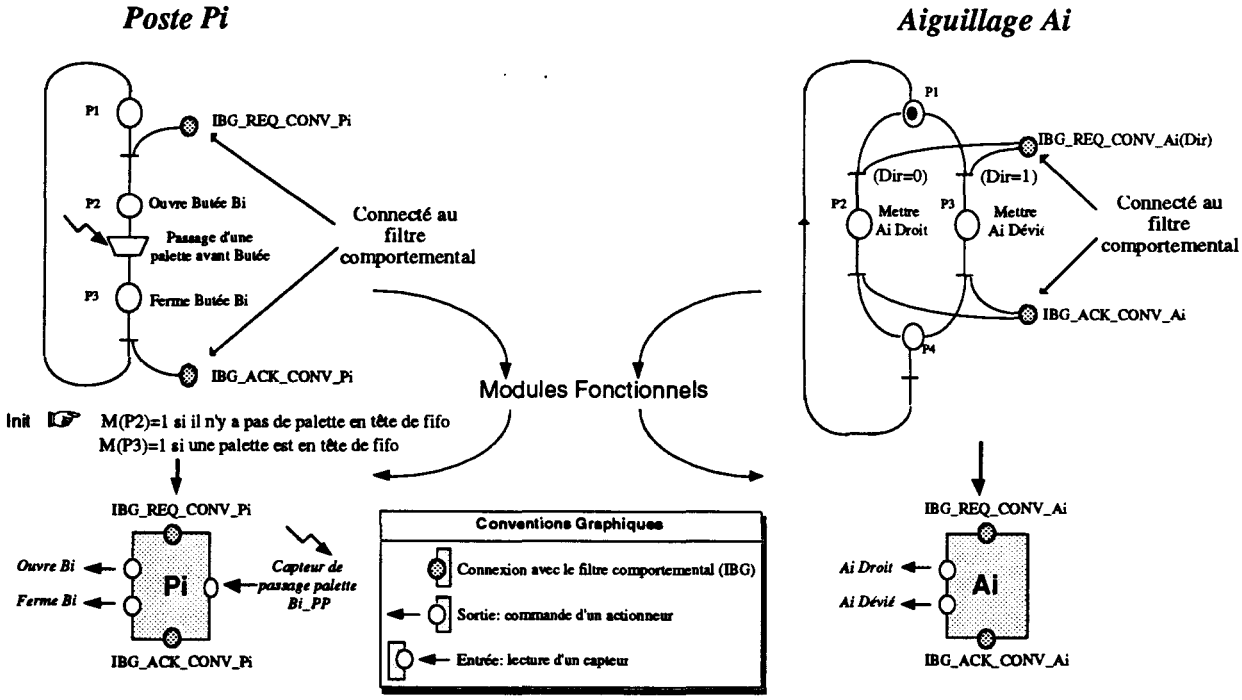


Figure II.43 : Graphes de commande génériques du convoyeur

Ces graphes de commande sont génériques dans le sens ou ils seront instanciés pour chaque poste (i=1..11) et chaque aiguillage (i=1..4) du convoyeur.

Le préfixe utilisé dans la notation des places connectées au filtre comportemental (***IBG : Interface entre Base logicielle commune et Graphes de commande***) trouvera son explication dans le § III.

Les composantes commandables peuvent être assimilées à des ressources simples, en effet, elles répondent bien à la définition de ces dernières. Les graphes de commande génériques de ces entités sont donc équivalents aux graphes de commande complets et peuvent, de la même manière, correspondre à l'association de graphes de commande partiels (figure II.44).

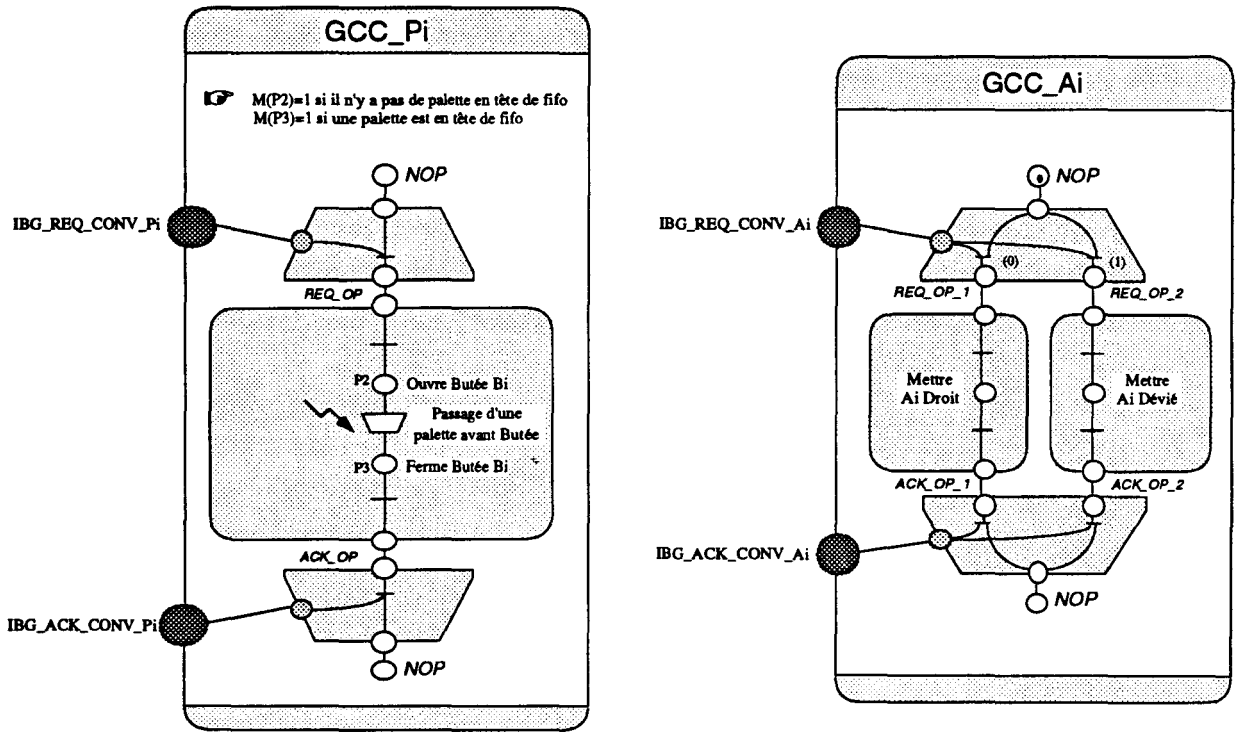


Figure II.44 : Décomposition des graphes de commande génériques des butées et aiguillages en graphes de commande partiels

Nous retrouvons dans ces graphes de commande complets les éléments constructeurs ainsi que des graphes de commande partiels correspondant au contrôle physique des éléments commandables.

Pour chaque poste et chaque aiguillage du convoyeur, nous devons instancier ces graphes de commande génériques pour obtenir l'ensemble des graphes de commande particularisés : GCC_P1, .. , GCC_P11 et GCC_A1, .., GCC_A4.

Il faut ensuite associer à l'ensemble de ces graphes de commande un filtre comportemental. Nous construisons celui ci par un assemblage d'entités élémentaires en utilisant les règles suivantes [Huvenoit et al 1992][Huvenoit et al 1994][Huvenoit et al 1995]:

- associer un **Module de Gestion de Poste** pour chaque poste Pi : les MGP Pi,
- associer une structure de donnée de type FIFO par **zone de stockages de palettes**,
- représenter les **chemins inter-postes** par des arcs sur lesquels nous indiquons l'ordre de séquencement des positionnements d'aiguillages et des réservations/libérations des jonctions.

Ainsi, nous proposons pour le convoyeur le filtre comportemental décrit par la figure II.45 .

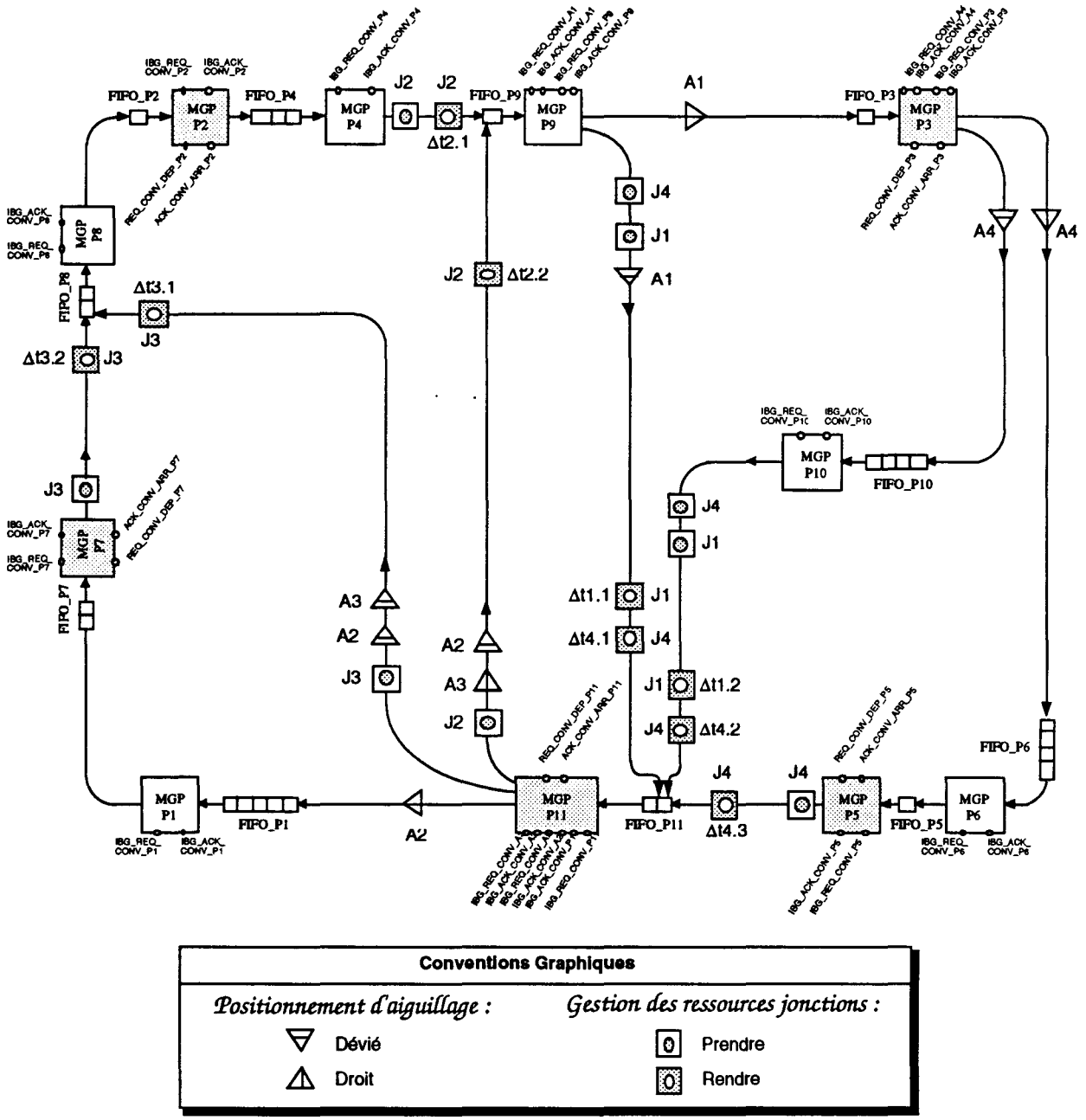


Figure II.45 : Modèle du filtre comportemental du CONVOYEUR

Dans ce modèle, il faut bien distinguer les **postes de destination finale** que sont P2, P3, P5, P7 et P11 (MGP en grisé sur la figure) des **postes intermédiaires**. Les premiers correspondent à des postes de palettisation/dépalettisation ou à des postes de traitement sur convoyeur (ex : étiquetage...) et leurs MGP sont dans tous les cas munis de protocoles de communication avec les gammes opératoires étendues (REQ_CONV_* et ACK_CONV_*). Les seconds correspondent uniquement à des postes de stockage (FIFO) et/ou de routage et ne sont pas en communication avec les gammes opératoires étendues.

De plus, pour chaque MGP, nous devons référencer les places de dialogue avec les graphes de commandes complets des composantes commandables qu'il sera amené à activer (IBG_REQ_CONV_* et IBG_ACK_CONV_*). En ce qui concerne la structure interne de ces modules de contrôle, nous la détaillerons ultérieurement.

Le formalisme sous-jacent à ce modèle est celui des réseaux de Petri. L'information qui circule correspond à des jetons à structure de données représentant les palettes. Les données référencées sont, entre autres, le type de palette, l'état d'allocation de la palette, la pièce présente sur la palette et le poste destination de la palette.

La gestion des jonctions est ici temporelle. Ainsi, les constantes de temps Δti.j correspondent au temps maximum mis par une palette pour franchir la zone critique. Bien sûr, cette gestion pourrait se faire à l'aide de capteur placés en fin de section critique.

Enfin, nous devons ajouter à ce modèle un ensemble de contrôles d'accès afin d'éviter les blocages par bouclage. Pour cela, il suffit de mettre en place les protocoles d'accès aux **circuits indépendants** [Cruette 1991]. Pour rechercher ces derniers, on considère le graphe biparti G où:

- les sommets sont les aiguillages ou jonctions du convoyeur,
- les arcs orientés sont les chemins entre ces aiguillages et/ou jonctions,

ce qui nous donne le graphe de la figure II.46.

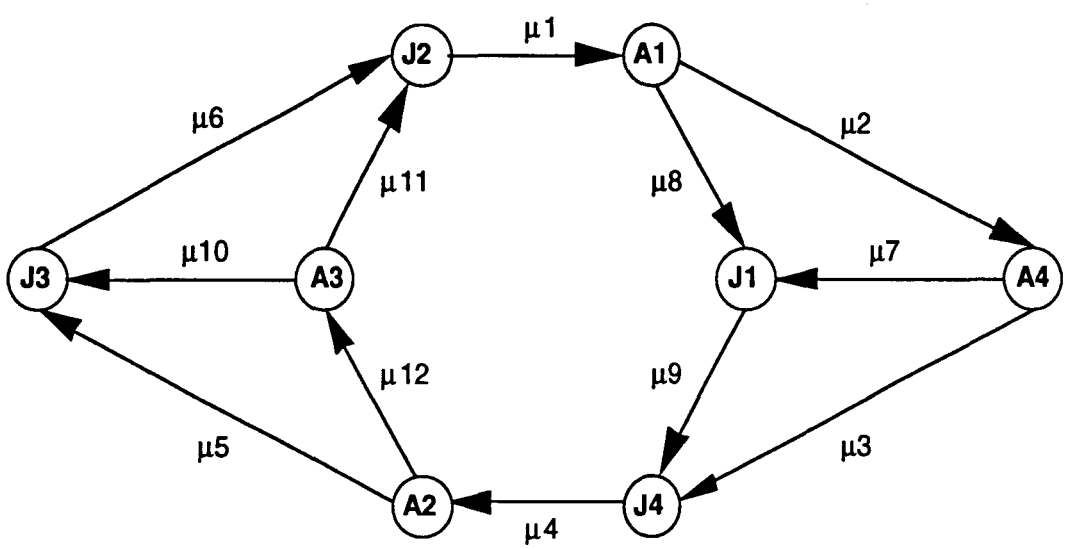


Figure II.46 : Graphe biparti G

A partir de ce graphe, nous pouvons en déduire l'ensemble des circuits le parcourant (Figure II.47):

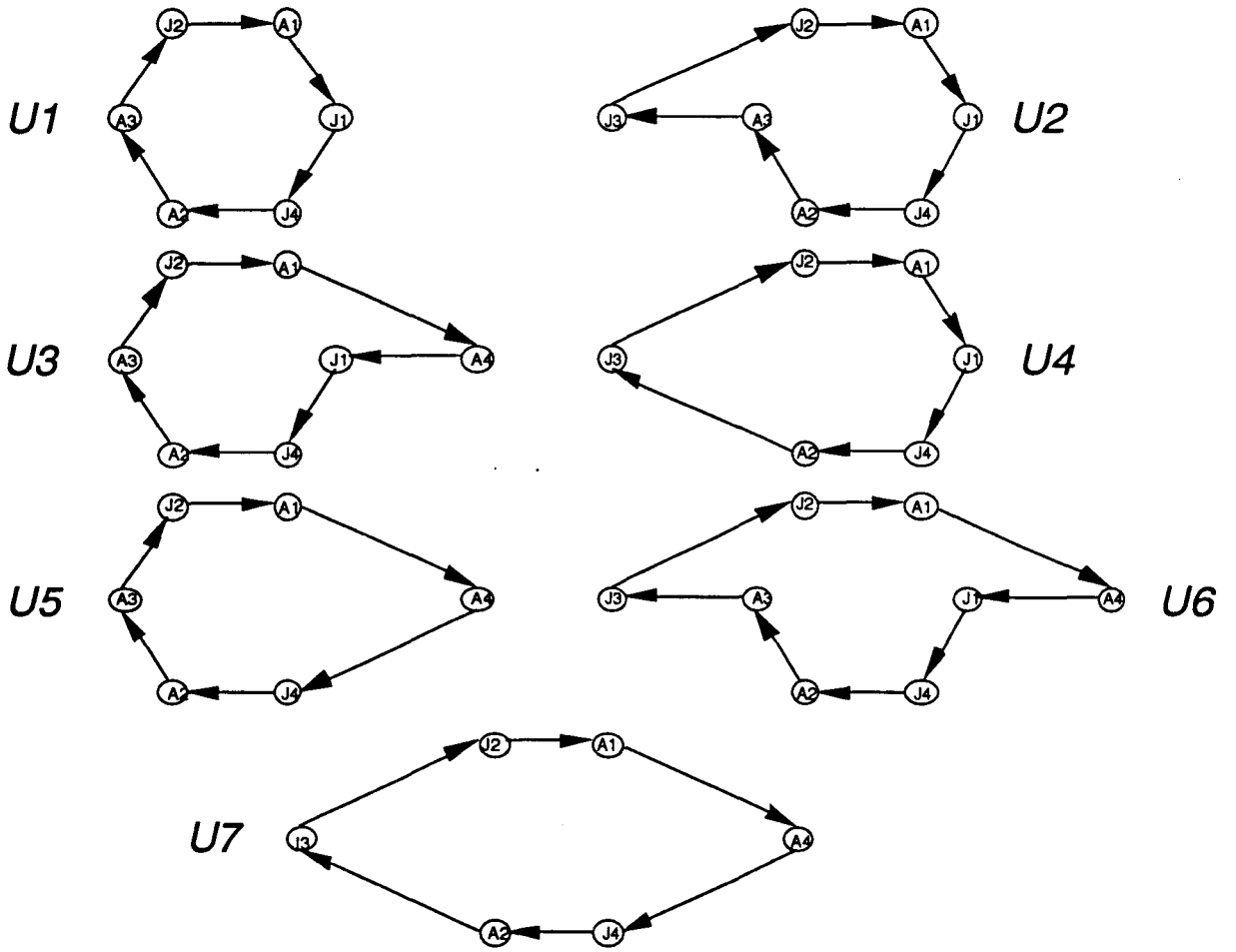


Figure II.47 : Circuits possibles dans G

Ces circuits au nombre de 7 ne sont pas indépendants. En effet, si nous considérons la base de cycles du graphe G, celle-ci est par définition un ensemble minimal de cycles indépendants. Le nombre de ces cycles indépendants est donc égal à la dimension de cette base que l'on appellera nombre cyclomatique $\upsilon(G)$. Or pour un graphe à N sommets, M arcs et p composantes connexes, nous avons [Gondram et al 1979] :

$$\upsilon(G)=M-N+p$$

Ici, nous avons 8 sommets (N=8), 12 arcs (M=12) et une seule composante connexe (p=1), nous avons donc un nombre cyclomatique égal à 5. Cela nous donne donc 5 circuits indépendants qui sont U1,U2,U3,U4 et U5, U6 et U7 étant des combinaisons linéaires de ces circuits.

Pour éviter les blocages du filtre comportemental du convoyeur, nous devons ajouter à celui-ci les protocoles d'accès à ces 5 circuits. Pour cela, nous définissons les sémaphores Ress_U1, Ress_U2, Ress_U3, Ress_U4, Ress_U5 dont la capacité est égale au nombre maximal de places dans chaque circuit correspondant moins une [Ausfelder 1994]. Par exemple, nous avons :

Max_Pal(U1)=M0(Ress_U1)=Long(Fifo_P9)+Long(Fifo_P11)-1

A l'initialisation, l'on soustrait de ces sémaphores autant de jetons qu'il y a de palettes présentes dans les circuits correspondant. Ensuite l'on prend un jeton dans le sémaphore Ress_Ui lorsque l'on entre dans le circuit Ui et que l'on le rend lorsque l'on en sort. Ainsi nous avons pour, par exemple, le passage d'une palette du poste P11 vers le poste P7 les mises à jour de sémaphores suivantes (Figure II.48) :

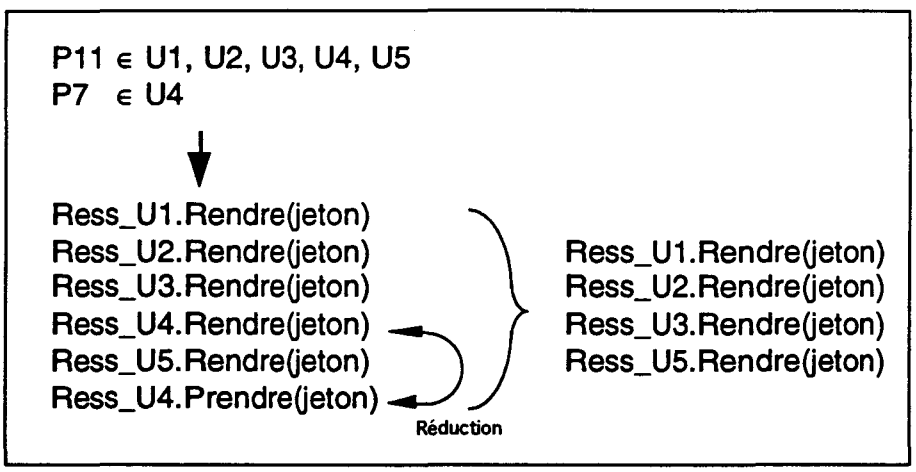


Figure II.48 : Modification des sémaphores de circuit pour un transfert de palette de P11 vers P7

A l'aide de tous ces éléments, nous pouvons alors construire la **gestion interne de chaque poste** (Les MGP Pi) [Huvenoit et al 1994][Huvenoit et al 1995]. Nous obtenons, par exemple, pour le poste P11, la modélisation suivante décrite par la figure II.49.

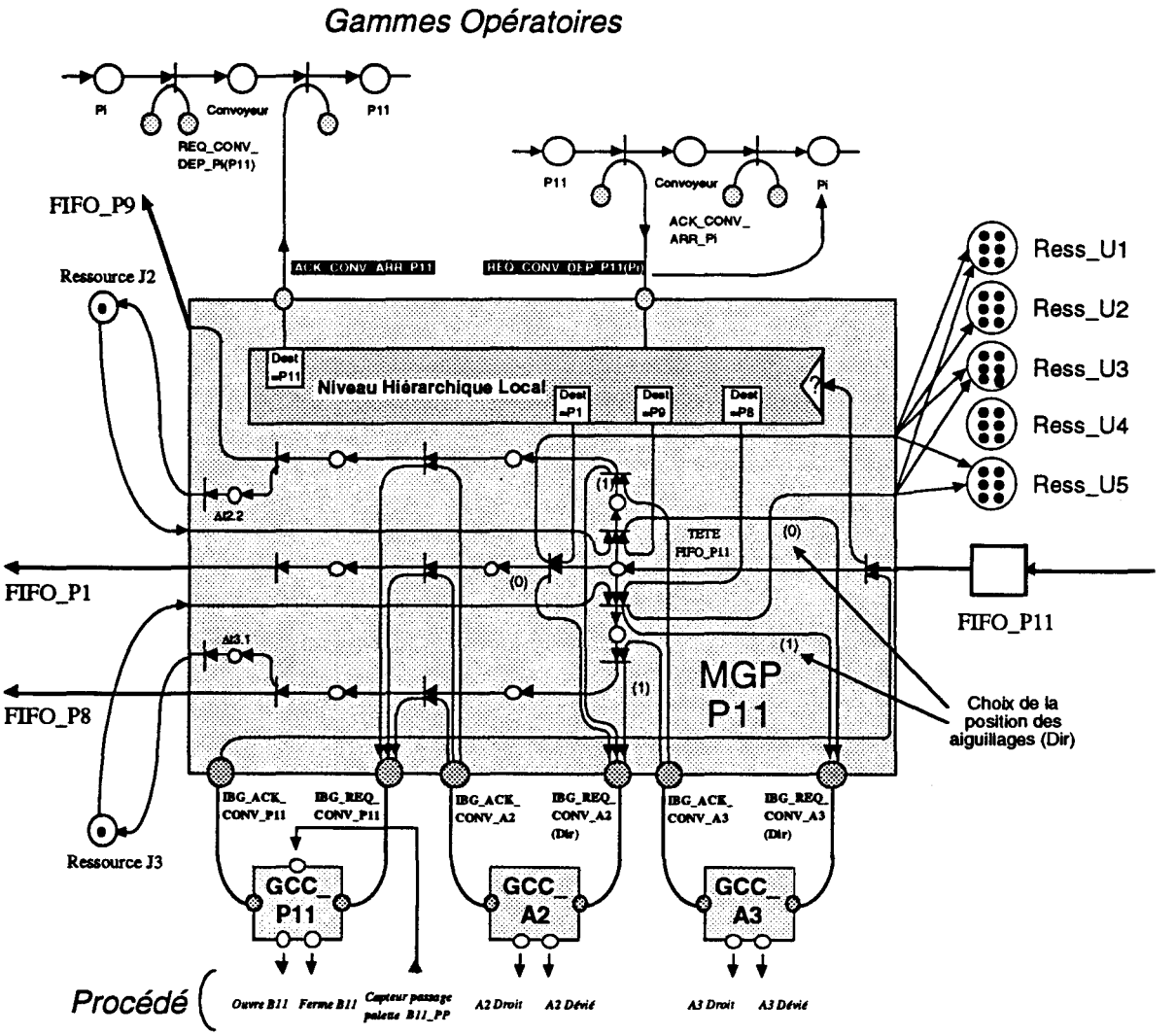


Figure II.49 : Modélisation de la gestion du poste MGP_P11

Décrivons le fonctionnement de cet ensemble : une palette arrivant en tête du fifo P11 est détectée par l'intermédiaire du graphe de commande GCC_P11, celui-ci retourne l'information sous forme de l'accusé IBG_ACK_CONV_P11. Le **Niveau Hiérarchique Local** (NHL) est alors consulté pour décider de l'utilisation de la palette.

Si la destination finale de cette palette, fixée lors d'une requête de gamme opératoire étendue, est le poste P11 [une REQ_CONV_DEP_Pi(P11)], un accusé de réception est renvoyé vers la GOE ayant fait cette demande de routage [un ACK_CON_ARR_P11] (Figure II.50).

Si la destination finale de la palette n'est pas le poste P11, c'est que cette dernière est en chemin vers un autre poste de destination finale. Dans ce cas, le NHL du poste P11 recherche le

chemin approprié pour ensuite aiguiller la palette dans la bonne direction (Postes P1, P8 ou P9). Il n'y a alors aucune communication avec les gammes opératoires étendues.

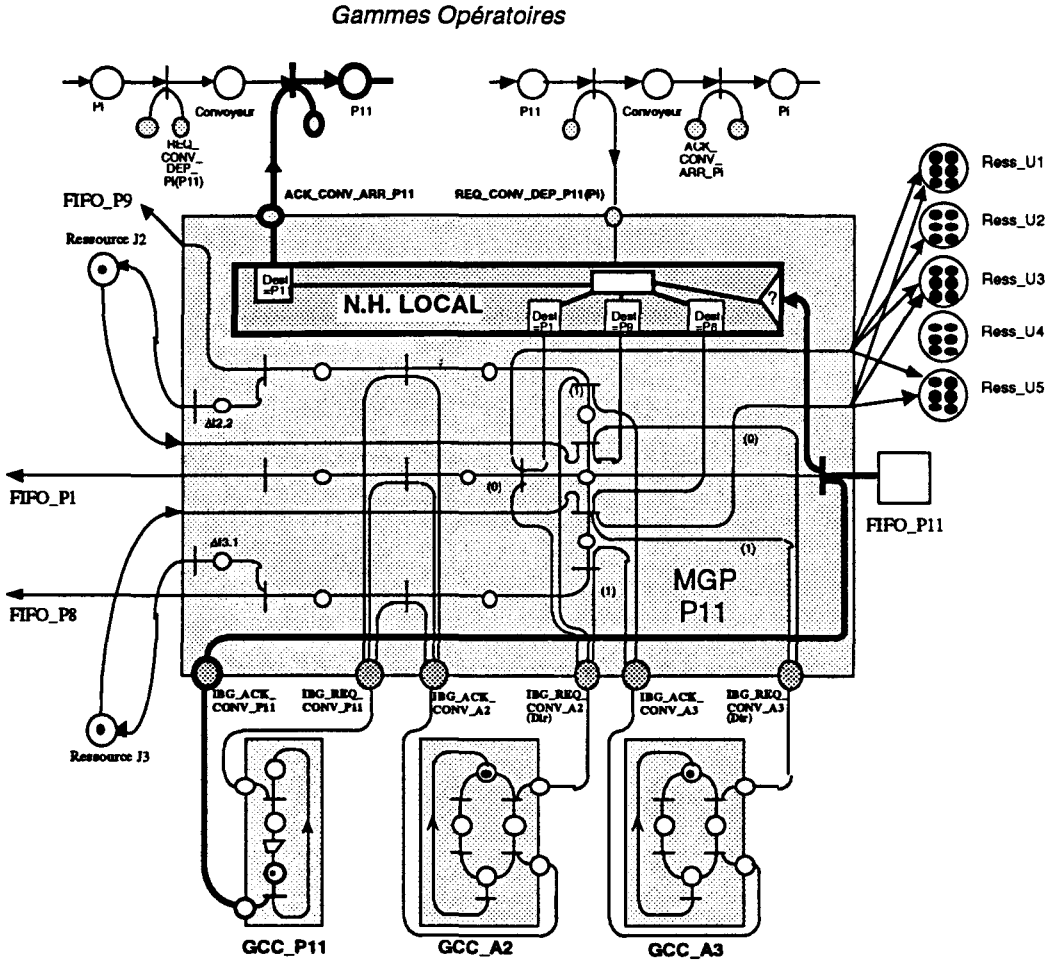


Figure II.50 : Fonctionnement du MGP du poste P11 : détection d'une palette

Après réalisation des opérations sur la pièce présente sur la palette, celles-ci étant définies par sa GOE (dépalettisation, usinage,...), si la palette n'est pas désallouée, la GOE demandera le départ de la palette vers une autre destination finale Pi qui peut être soit P2, soit P3, soit P5, soit P7, soit P11. Cette demande s'effectue par l'intermédiaire d'une requête REQ_CONV_DEP_P11(Pi). Le NHL est alors consulté et décide du chemin à prendre pour que la palette aboutisse vers cette destination. En conséquence, il va aiguiller la palette vers un des postes qui suit immédiatement P11 (P1, P8 ou P9).

L'opération d'aiguillage et de libération de la palette s'effectue par la commande séquentielle des graphes de commande complets des aiguillages (GCC_A2, GCC_A3) puis de la butée (GCC_P11) (Figure II.51).

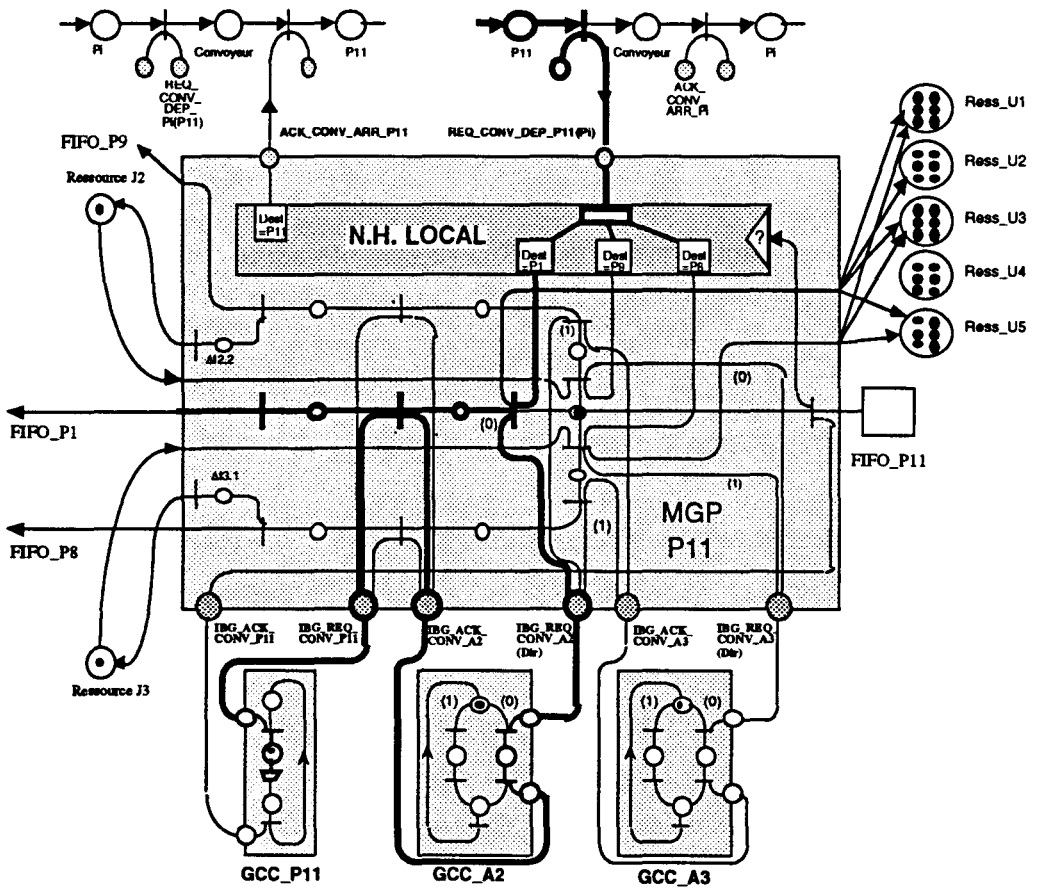


Figure II.51 : Fonctionnement du MGP du poste P11 : Aiguillage vers le poste P1

Nous remarquerons que ce poste P11 est un poste de destination finale. Cette particularité est soulignée par le fait que le Niveau Hiérarchique Local est directement en communication avec les gammes opératoires étendues, en effet, celui ci doit tenir compte des requêtes de service à effectuer que ces dernières lui envoient. Si le poste était un poste intermédiaire, cette communication serait inexistante.

Comme nous avons pu le voir pour le poste P11, les postes P3 et P9, ont un NHL qui intègre un **résolveur d'indéterminisme local** (type IPL) car en aval de ces postes, il y au moins deux chemins pouvant être empruntés par les palettes. En fonction de l'état de ces divers chemins et des caractéristiques de la pièce présente sur la palette, elle même présente en tête de fifo, le résolveur d'indéterminisme va décider de la direction effectivement prise par cette dernière.

Dans le modèle présenté, nous trouvons associé au NHL un Réseau de Petri qui contrôle l'ensemble des sémaphores de circuit ainsi que les sémaphores gérant l'accès aux composantes non commandables, que sont les jonctions, dans le but d'activer, suivant la bonne séquence (définie par les arcs de la figure II.51), les graphes de commande des aiguillages et butées.

Dans tous les cas, le NHL tient compte de l'état de tout ou partie du convoyeur ainsi que de l'ensemble des contraintes temporelles que doit respecter chaque pièce (ordonnancement). Nous verrons cela plus en détail dans la partie II.G.

II.F.3.b. Les ressources de transformation complexes

Dans notre exemple, seul le centre d'usinage CU1 est une ressource de transformation complexe. Cet aspect est dû au fait que celui ci dispose de deux tampons d'entrée/sortie et d'une seule zone opératoire.

Au maximum de ses possibilités, ce centre d'usinage peut recevoir simultanément deux demandes de transfert vers sa zone opératoire. Il faut donc mettre en place un filtre comportemental pour administrer ce conflit. Le Niveau Hiérarchique Local intégré dans celui ci aura pour fonction la sélection prioritaire de l'une des requêtes.

En considérant que les diverses commandes de transfert vers la zone opératoire (IO1→ZO, IO2→ZO, ZO→IO1, ZO→IO2) sont disponibles, nous obtenons le graphe de commande complet et le filtre comportemental décrits par la figure II.52.

Au niveau du graphe de commande, il ne subsiste plus d'indéterminisme, la sélection du chargement de la zone opératoire étant effectuée au niveau du filtre comportemental. Ce dernier peut être sollicité par l'intermédiaire des requêtes REQ_TRANSF_IO1 et REQ_TRANS_IO2 provenant de gammes opératoires étendues de deux pièces différentes.

Il est à noter que la sélection de IO1 ou de IO2 pour recevoir telle ou telle pièce s'effectue auparavant par l'intermédiaire de l'**allocateur de ressource** dont nous allons présenter la raison d'être ainsi que le fonctionnement dans la partie II.G..

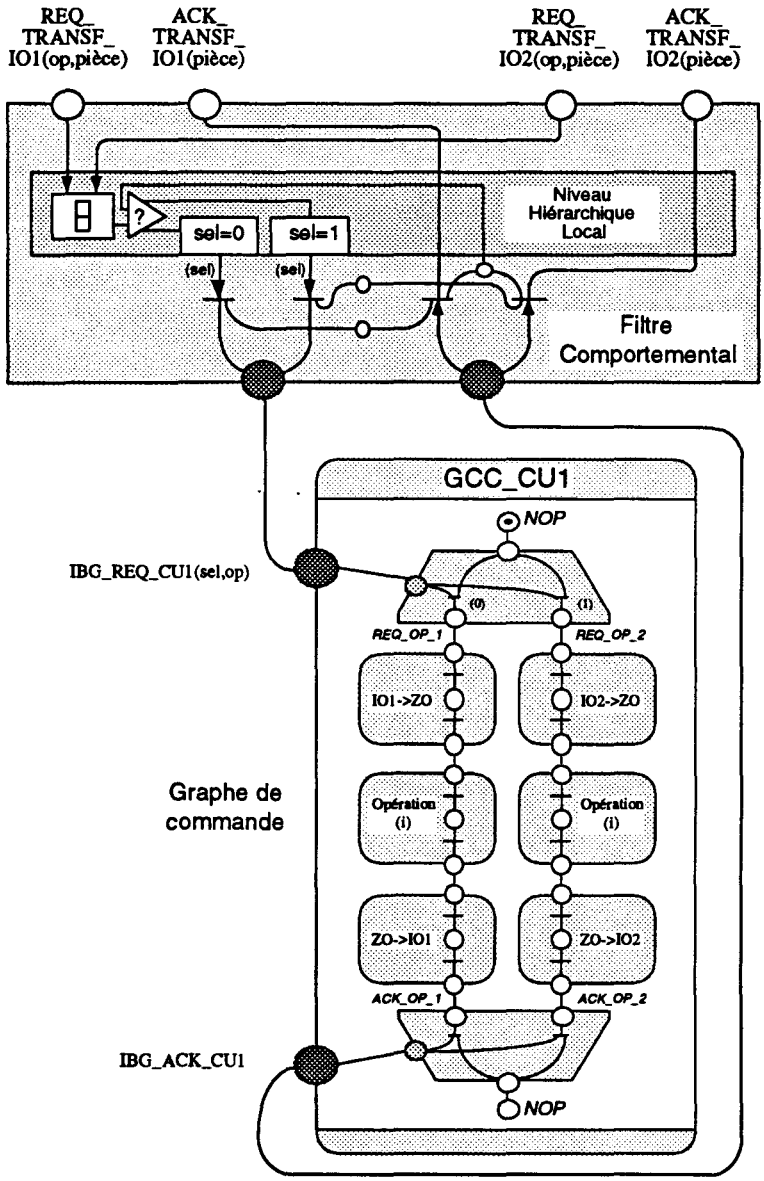


Figure II.52 : Filtre comportemental et graphe de commande du centre d'usinage CU1

II.F.3.c. Les autres ressources complexes

Les autres ressources complexes telles les stockeurs ST1 et ST2 ou le poste d'assemblage ASS1 seront traitées de la même manière. Dans tous les cas, nous retrouverons un filtre comportemental.

Pour les stockeurs, nous trouverons principalement dans le filtre comportemental une structure de données permettant de contrôler efficacement le stockage et le déstockage qui

représentent les deux services à rendre vis a vis des gammes opératoires étendues. Pour le stockeur à plat, il y aura absence de graphe de commande, en effet, celui ci ne requière aucune commande, c'est une ressource passive ne disposant pas d'actionneurs.

En ce qui concerne le poste d'assemblage, le filtre comportemental a pour rôle la synchronisation entre les différentes gammes des pièces à assembler. De fait, il ne comporte pas de niveau hiérarchique local (figure II.53).

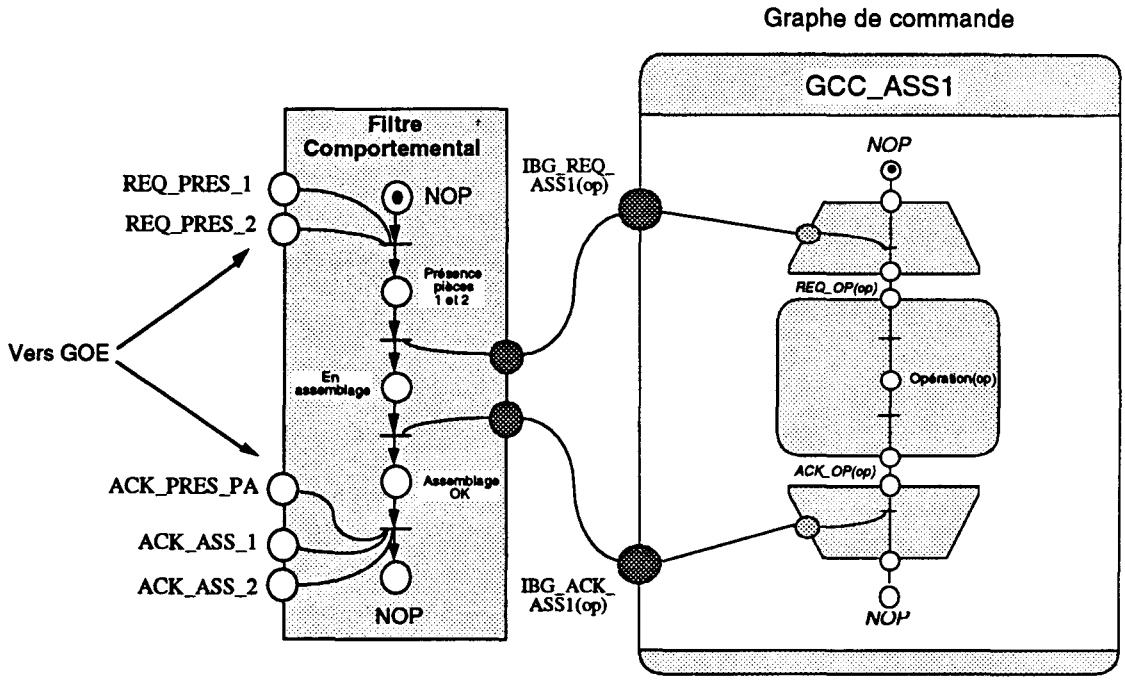


Figure II.53 : Filtre comportemental et graphe de commande du poste d'assemblage ASS1

La gamme opératoire étendue de la première pièce à assembler signale la présence de cette dernière sur le poste d'assemblage par l'intermédiaire de la requête REQ_PRES_1. La GOE de la seconde pièce agit de même en envoyant la requête REQ_PRES_2. En présence de ces deux requêtes, donc des deux pièces à assembler, le filtre comportemental du poste d'assemblage peut solliciter le graphe de commande afin de lancer l'opération d'assemblage (IBG_REQ_ASS1). Une fois celle-ci effectuée, le filtre comportemental envoie des accusés de réception, d'une part, vers les deux GOE appelantes (ACK_ASS_1 et ACK_ASS_2) et d'autre part, vers la GOE de la pièce correspondant à l'assemblage afin de signifier la 'naissance' de cette dernière (ACK_PRES_PA).

En cas d'assemblages de plusieurs types, l'allocateur de ressource aurait la charge de contrôler l'accès des bonnes séquences de pièces.

II.G. Modélisation des Niveaux décisionnels

Dans les divers modèles présentés lors des précédentes parties modélisant, d'une part, la partie logique et d'autre part, la partie physique, nous avons soulevé l'existence de plusieurs types d'indéterminismes. Ce sont :

➪ *dans les gammes opératoires étendues:*

- ✔ indéterminismes dus à la flexibilité des produits (IFP),
- ✔ indéterminismes dus au choix entre ressources de fonctionnalité identique (ICR),
- ✔ indéterminismes dus au choix entre différents chemins (ICC),

➪ *dans les filtres comportementaux des ressources complexes:*

- ✔ indéterminismes locaux dus à des choix internes aux ressources (IPL),

➪ *dans les allocateurs de ressource:*

- ✔ indéterminismes dus à des accès simultanés envers la même ressource (IAS).

Nous allons aborder la résolution de ces indéterminismes successivement en supposant qu'il existe une **priorité** entre les diverses pièces qui circulent sur la cellule. Cette priorité peut être statique ou dynamique et dépendre d'un ordonnancement prévisionnel ou d'un ordonnancement dynamique [Hammadi 1991].

II.G.1. Niveaux décisionnels liés aux gammes opératoires étendues

Normalement, l'ensemble des indéterminismes liés aux gammes opératoires étendues sont indirectement résolus pour chaque pièce par le module d'ordonnancement. Ce dernier calcule pour chaque pièce une série de contraintes temporelles que le Niveau Hiérarchique Supérieur (NHS) va tenter de faire respecter en tenant compte des perturbations pouvant apparaître dans le système [Tawegoum et al 1992].

II.G.2. Niveaux décisionnels liés aux ressources complexes

Les indéterminismes liés aux ressources complexes sont très dépendants de la nature physique de ces dernières.

Par exemple, pour le convoyeur de la cellule prise en exemple, nous avons un résolveur d'indéterminisme local intégré dans la gestion des postes situés en amont d'aiguillages (P3, P9,

P11). Si nous considérons le poste P11, nous avons vu dans la partie II.F.3.a que les palettes peuvent être dirigées vers une nouvelle destination finale pouvant être soit P2, soit P3, soit P5, soit P7, soit même P11. Il faut donc décider du chemin à emprunter pour atteindre cette destination en tenant compte de la priorité des pièces et de l'état des postes en aval. Pour cela, nous proposons l'algorithme suivant :

```
si POSTE_DESTINATION_FINALE=P7 alors
  boucle_1
    si FIFO_P1 non PLEIN
      alors TRANSFERT vers P1
            exit boucle_1
      sinon si PRIORITE_PIECE<SEUIL
                alors TRANSFERT vers P9
                      exit boucle_1
                sinon ATTENDRE(PERIODE_EVL)
  fin boucle_1

si POSTE_DESTINATION_FINALE=P2 alors
  boucle_2
    si FIFO_P8 non PLEIN
      alors TRANSFERT vers P8
            exit boucle_2
    sinon si FIFO_P1 non PLEIN
      alors TRANSFERT vers P1
            exit boucle_2
     sinon si PRIORITE_PIECE<SEUIL
                alors TRANSFERT vers P9
                      exit boucle_2
                sinon ATTENDRE(PERIODE_EVL)
  fin boucle_2

si POSTE_DESTINATION_FINALE=P3 alors
  boucle_3
    :
    :
```

La variable PRIORITE_PIECE peut être fixée a priori pour chaque type de pièce ou peut être calculée en fonction de l'ordonnancement. On a alors PRIORITE_PIECE=f(Marge_restante, Temps_au_plus_tard,...). La constante SEUIL est à calculer en fonction du temps moyen mis pour emprunter le chemin dévié. Le temps PERIODE_EVL est le temps minimal d'évolution de l'état du convoyeur et dépend directement de la célérité de ce dernier.

Afin d'optimiser les flux de palettes, un contrôle plus complexe prenant en compte dans sa globalité l'état du convoyeur peut être mis en oeuvre [Tawegoum et al 94].

II.G.3. Les allocateurs de ressources

Pour régler les indéterminismes résultant d'accès multiples envers les ressources partagées, nous avons vu qu'il faut adjoindre à notre structure de contrôle un module d'allocation assigné à chaque ressource.

Pour les ressources simples, nous pouvons mettre en place un module d'allocation générique dont la structure est la suivante (figure II.54) :

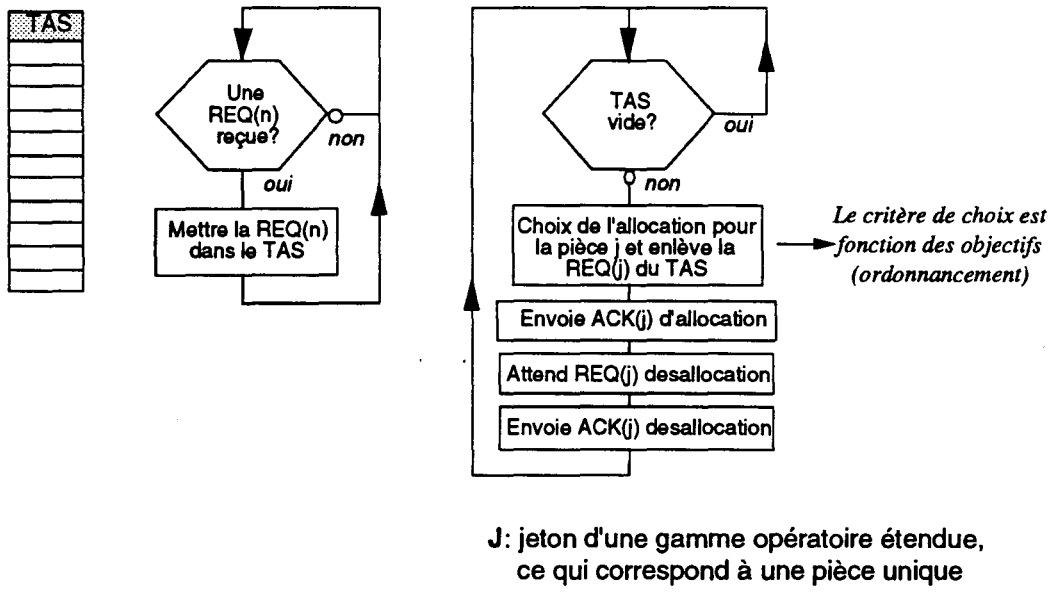


Figure II.54 : Algorithme générique d'allocation simple

Notons tout d'abord que les deux parties de cet algorithme fonctionnent en parallèle. Par le séquencement des opérations, nous retrouvons bien le fait qu'à toute allocation correspond une désallocation.

L'instanciation de ce module générique se fera en passant comme paramètre la fonction de choix. Celle ci va être utilisée pour choisir parmi les pièces en attente, dont les requêtes sont dans le TAS, celle qui va être servie. La fonction de choix prendra en compte la priorité de chaque pièce et en toute logique, la pièce ayant la priorité la plus élevée sera celle que l'allocateur retiendra.

Cet algorithme ne peut pas être généralisé aux ressources complexes. En effet, chaque ressource est alors un cas particulier dont l'organisation physique influe fortement sur la fonction d'allocation.

Par exemple, pour le convoyeur, nous devons mettre en place un allocateur simple par type de palette présente sur le convoyeur. De plus, dans certains cas, l'allocateur doit avoir connaissance de l'état de la ressource considérée pour pouvoir prendre une décision. Il est alors nécessaire d'instaurer une communication bidirectionnelle entre le module d'allocation et le filtre comportemental.

II.H. Vue globale sur la structure de contrôle d'un SFPM

L'assemblage de l'ensemble des modèles décrits dans ce chapitre permet d'obtenir la structure complète de la commande d'un système flexible de production manufacturière. A ces modèles, il faut adjoindre des modules annexes complémentaires, concernant notamment la surveillance. Finalement, nous proposons l'architecture fonctionnelle de contrôle décrite par la figure II.55 [Huvenoit et al 1992][Huvenoit et al 1994][Huvenoit et al 1995].

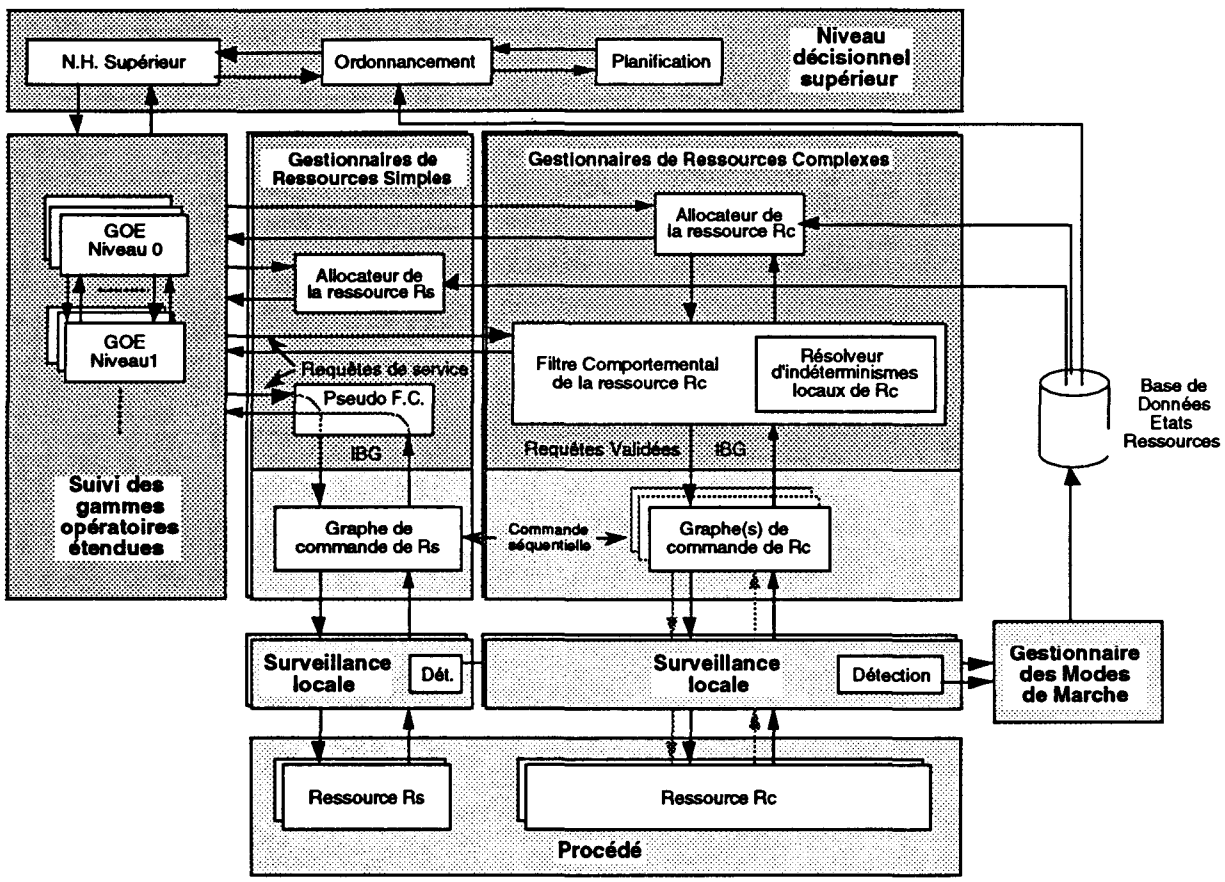


Figure II.55 : Architecture fonctionnelle de contrôle d'un SFPM

Dans cette architecture, nous retrouvons, au niveau le plus haut, le Niveau Décisionnel Supérieur qui intègre la planification et l'ordonnancement. Celui-ci contrôle les diverses gammes opératoires étendues qui vont solliciter de façon transparente les gestionnaires de ressources. A ce niveau, nous aurons donc une communication du type client/serveur.

Suivant le type de ressource commandée, le gestionnaire de ressource intégre (ressources complexes) ou n'intègre pas (ressources simples) de filtre comportemental. Mais pour ce dernier cas, nous avons introduit des **pseudo filtres comportementaux.** Ceux ci sont

totalement **transparents du point de vue fonctionnel**. Ils permettront de contrôler les flux d'informations et d'homogénéiser l'implantation.

Entre les graphes de commande et les ressources, nous insérons des modules de surveillance locale [Toguyeni 1992] [Elkhattabi 1993]. En cas de détection de défaillance, le gestionnaire des modes de marche met à jour la base de données représentant l'état des ressources [Bois 1991]. Cette dernière peut ensuite être consultée par les allocateurs de ressources afin de rejeter ou de mettre en attente les demandes d'allocation en cas d'indisponibilité de celles ci pour cause de panne.

Le module d'ordonnancement peut également consulter cette base de données afin d'apporter des corrections dynamiques dans le but de résorber les perturbations causées par la défaillance de une ou plusieurs ressources [Hammadi 1991].

CHAPITRE III

Une méthodologie d'implantation

SOMMAIRE DU CHAPITRE III

III.A. Introduction

Dans le chapitre précédent, nous avons vu comment modéliser l'ensemble de la commande de notre SFPM. Dans ce chapitre, après un aperçu sur les prérogatives nécessaires à la mise en œuvre de notre démarche, nous aborderons la phase d'implantation de l'ensemble des modèles de conception constituant la commande en montrant bien que celle ci est suffisamment automatisable pour être intégrée dans un atelier de génie automatique. Enfin, nous montrerons comment il est possible et peu contraignant de passer par une phase de validation par simulation avant l'implantation effective sur le site industriel.

III.B. Définition de l'environnement informatique

III.B.1. Contraintes inhérentes aux systèmes étudiés

La mise en œuvre des systèmes que sont les SFPM conduit à établir une coopération efficace entre une **partie matérielle** et une **partie logicielle**. La partie matérielle est composée de ressources de production, d'organes de commande informatiques (Commandes Numériques, Automates Programmables, PC,...), de divers réseaux informatiques,... tandis que la partie logicielle correspond à l'informatisation de la commande modélisée dans le chapitre II.

Les fonctionnalités globales de notre outil de production reposeront sur une bonne adéquation de ces deux parties. De plus, pour orienter notre développement, il est primordial de tenir compte des exigences imposées par la partie matérielle et du devenir de l'ensemble des logiciels.

Les systèmes que nous sommes amenés à mettre en œuvre s'inscrivent dans le cadre des applications **temps réel**, en effet :

> Peut être qualifiée de **temps réel** toute application mettant en œuvre un système informatique dont le fonctionnement est assujetti à l'évolution dynamique de l'état d'un environnement (appelé ici procédé) qui lui est connecté et dont il contrôle le comportement [CNRS 1988].

La coordination des ressources de production est une application du type **temps réel lâche** [Sielski 1993] car ce qui importe, ce n'est pas de respecter exactement des délais

d'exécutions fixes mais de respecter, aux mieux, des contraintes temporelles relativement flexibles (ordonnancement). Par contre, toutes les applications qui relèvent de la surveillance peuvent être considérées comme étant du type **temps réel critique**. En effet, en cas de défaillance, il est primordial de réagir le plus rapidement possible si ce n'est instantanément pour éviter d'aboutir à des états dangereux pour le procédé ou pour le personnel.

Certaines exigences de la **partie matérielle** ont été prises en compte dès la conception de la commande de coordination. Par exemple, les liens entre les graphes de commande complets et les graphes de coordination (Gammes Opératoires Etendues) ont été réduits au strict minimum afin d'obtenir un **faible couplage**. En effet, si l'on retient une **implantation répartie**, il est primordial de ne pas saturer les réseaux informatiques par des communications trop fréquentes et des flux de données trop importants.

Pour ce qui est du **logiciel**, celui ci doit être **maintenable** et **évolutif** afin de s'adapter rapidement aux modifications de la partie matérielle et aux changements d'orientations dans la production. Il doit être **performant** dans le sens où il doit avoir un temps de réponse conforme avec les objectifs de production et **robuste** pour rester opérationnel en cas d'incidents dans la partie matérielle.

III.B.2. Choix du langage d'implémentation

Lors d'une mise en œuvre logicielle, il faut définir le ou les langages utilisés pour mener à bien celle-ci. Or les systèmes auxquels nous nous consacrons sont du type temps réel avec un grand degré de parallélisme, ils ont donc un ensemble de comportements dynamiques plus ou moins corrélés qu'il faut contrôler simultanément. Pour la mise en œuvre de tels systèmes, plusieurs approches sont possibles [Wallas et al 1993] : l'approche **asynchrone**, l'approche **synchrone** [Peraldi 1993] et l'approche **mixte** [DRED 1993] [Farah et al 1994] . Pour notre part, nous nous cantonnerons à l'approche asynchrone.

Habituellement, dans l'industrie manufacturière, les techniques temps réel asynchrones sont mises en œuvre par l'intermédiaire d'exécutifs multitâches temps réels ou de systèmes d'exploitations temps réel tels iRMX, VRTX32, OS/9 ou POSIX [iRMX 1980] [Arcos et al 1994].

Ces solutions, souvent onéreuse, non normalisées et faiblement portables, ont l'inconvénient de nécessiter soit un matériel spécifique soit une double compétence soit un environnement logiciel spécifique permettant rarement de faire de la programmation de haut niveau intégrant des concepts tels que les objets.

Or, la modélisation du système de contrôle/commande que nous avons abordé dans le chapitre II favorise les aspects objets, nous avons donc recherché un langage à la fois de haut niveau et temps réel dont le parallélisme est géré indépendamment du système d'exploitation. Le langage **ADA** répond assez bien à l'ensemble de ces exigences et nous le retiendrons pour l'implantation effective.

A l'origine, ADA a été développé pour le compte du Département de la Défense des Etats Unis (DOD) afin de répondre aux exigences des développements concernant les applications embarquées (Embedded Systems). Il a été conçu dans le souci de l'efficacité et de la sécurité par un rassemblement de nombreux concepts [TSI 1985]. C'est ainsi l'un des rares langages à intégrer directement l'ensemble des notions que sont le **parallélisme explicite, la généricité, la séparation du code en paquetages** et le **traitement d'exceptions** [Le Verrand 1985].

La première version d'ADA, faisant l'objet d'une norme, date de 1983 [DOD 1983]. C'est avec cette version que nous avons effectué nos développements. Actuellement, une nouvelle version (projet ADA 9x) est à l'étude. En plus de nouvelles fonctionnalités objets et autres, cette nouvelle mouture du langage sera beaucoup mieux adapté au temps réel critique [Richard Foy 1994].

Le multi-tâches est inhérent à ADA, il est indépendant de l'architecture physique (Microcontrôleur, Monoprocesseur, Multiprocesseur,...) et du système d'exploitation (VMS, UNIX, MSDOS, OS/2,...) des calculateurs sur lesquels le compilateur est implanté. L'environnement multi-tâches d'ADA intègre des primitives de base permettant de mettre en oeuvre toutes les structures d'échanges synchrones ou asynchrones nécessaires au développement de logiciels temps réel.

La notion de paquetages permet **la compilation séparée, la modularité** et **l'encapsulation de l'information**. En effet, dans la définition des paquetages, ADA permet, entre autres, de bien séparer la spécification de l'implémentation effective et ce quelque soit l'entité (procédures, fonctions, tâches,...). Il permet aussi de déclarer des variables publiques ou privées; c'est à dire visibles ou non de l'extérieur du paquetage.

Le langage ADA intègre les principales caractéristiques du concept objets bien qu'il ne soit pas considéré comme un langage totalement orienté objet. Il est plutôt défini comme étant un **langage basé objet** (Object Based Language) [Booch 1992] car, comme le résume le tableau de la figure III.1, certaines notions telles l'héritage sont absentes du langage.

En ce qui nous concerne, cette imperfection est minime car dans notre approche nous

favorisons surtout la composition et les aspects hiérarchiques peuvent êtres compensés par la généricité.

L'aspect très modulaire d'ADA permet d'engendrer un **code réutilisable**; cela va nous permettre de construire une **bibliothèque de composants de base** correspondant, par exemple, au codage des graphes de commandes partiels modélisés dans le chapitre II.

Entité	Variables d'instances Méthodes d'instances Variables de classes Méthodes de classes	Oui Oui Non Non
Encapsulation	Des variables Des méthodes	Publique, privée Publique, privée
Modularité	Genres de modules	Empaquetage (spécification/corps)
Hiérarchie	Héritage Unités génériques Métaclasses	Non Oui Non
Typage	Fortement typé Polymorphisme	Oui Non
Simultanéité	Multi-tâches	Oui (définie par le langage)
Permanence	Objets permanents	Non

Figure III.1 : Caractéristiques principales du langage ADA [Booch 1992]

L'intérêt du langage ADA tient aussi du fait que c'est l'un des rares langages à disposer d'un **environnement de programmation standard** : l'**APSE** (Ada Programming Support Environment). Cet environnement apporte un ensemble d'outils logiciels intégrés permettant de soutenir les applications développées en ADA. Ces outils vont de l'éditeur à l'analyseur de code et permettent de maintenir la cohérence depuis la spécification jusqu'à l'implantation finale [Booch 1988].

Enfin, ADA est un langage d'une relecture facile, sans ambiguïtés dont la portabilité n'est plus à démontrer.

III.B.3. Architecture matérielle préconisée pour notre approche

L'**architecture logicielle** va découler de l'organisation établie lors de la conception , de fait elle sera **modulaire** et **hiérarchisée** . Elle se décomposera en trois niveaux principaux :

- le niveau décisionnel supérieur,
- le niveau suivi des gammes,
- le niveau pilotage des ressources.

Pour construire et/ou valider l'**architecture informatique** du site de production devant accueillir cette architecture logicielle, nous devons tenir compte de diverses exigences. Ainsi, pour une parfaite adéquation avec les modèles de commande, il est préférable de disposer ou de mettre en place une architecture informatique plus ou moins hiérarchisée. De plus, en raison du choix d'ADA comme langage d'implantation, il est nécessaire de disposer d'au moins un calculateur supportant ce langage.

Au niveau le plus haut de cette architecture informatique, nous trouvons un calculateur du type mini-ordinateur sur lequel est installé un ensemble de logiciels de CAO, CFAO, CFAO robotique, CCAO (Conception de Commande Assisté par Ordinateur),...Ces logiciels ont pour dessein la conception des pièces, la définition des gammes logiques, des gammes d'assemblage,... mais aussi la génération des programmes d'usinage, des programmes robots [Rancky 1990] [Engineers 1988].

Nous devons ensuite répartir l'architecture fonctionnelle de contrôle du SFPM (figure II.55) sur une **architecture opérationnelle répartie** et constituée de divers organes informatiques interconnectés par l'intermédiaire de réseaux.

La procédure à suivre pour implanter les commandes locales des ressources dépend fortement du type de ces dernières.

Si la ressource est de type simple, le filtre comportemental n'a pas de raison d'être. Une commande numérique (CNC), un automate programmable industriel (API) ou un micro ordinateur industriel muni de cartes d'entrées/sorties industrielles suffit pour implanter les graphes de commande complets quelque soit le langage métier utilisé (Grafcet, LM, ...). Dans le cas des machines livrées "clé en main", ce calculateur est généralement présent et intégré à la machine.

Si la ressource est de type complexe, elle peut nécessiter la gestion d'un modèle plus ou moins complexe : le filtre comportemental. Ce modèle caractérise le plus souvent des comportements dynamiques; il nécessite alors la mise en place d'un contrôle temps réel. Dans le cas d'un comportement statique, il peut être modélisé par une structure de données; l'implantation sous forme d'une base de données est alors suffisante.

Dans le cas d'un comportement dynamique et aux vues de la complexité du modèle, il est nécessaire de disposer d'un organe informatique pouvant supporter son implantation. Pour les modèles complexes qui disposent, en particulier, d'un niveau décisionnel local, certains calculateurs industriels programmables à l'aide d'un langage métier trop primitif sont à éviter car l'implantation serait trop hasardeuse et économiquement non viable.

Pour faciliter le passage de la conception à l'implantation en conservant la cohérence des structures de décision et en maximisant l'utilisation des bibliothèques de composants réutilisables, une solution intéressante et élégante consiste à utiliser, ici aussi, ADA comme langage d'implantation. Nous proposons alors l'utilisation de l'une des solutions (mais non exhaustives) décrites par la figure III.2.

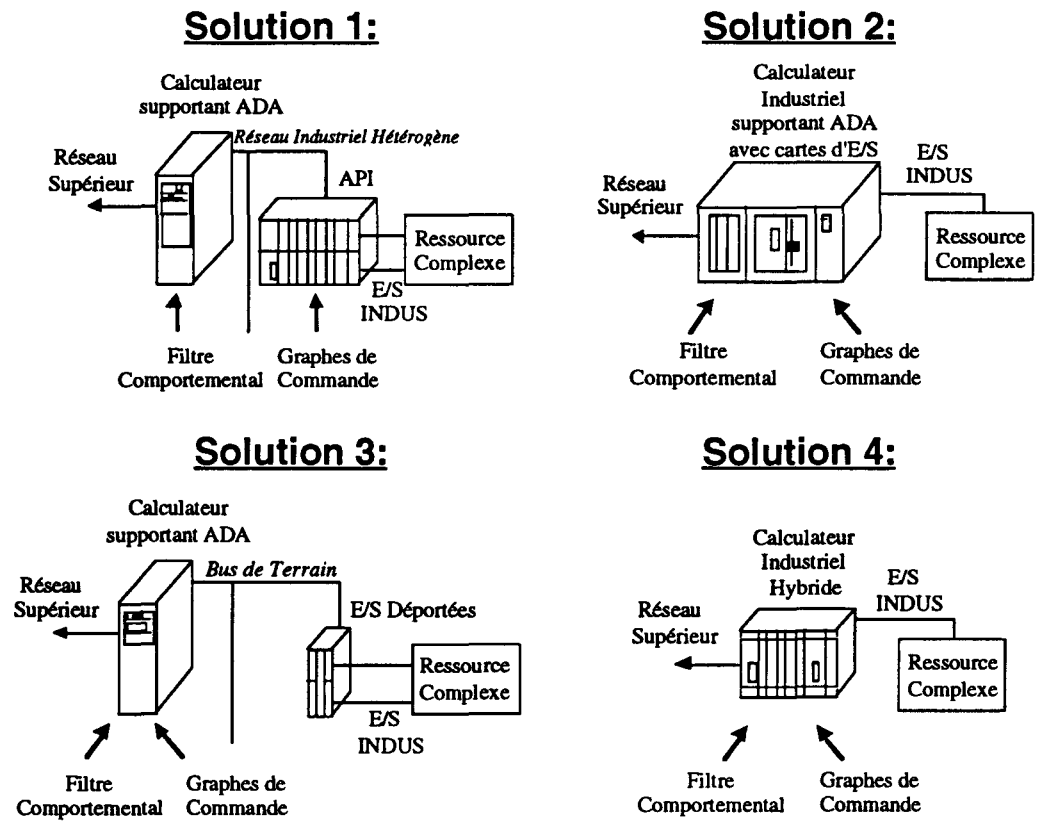


Figure III.2 : Exemples de ressources informatiques nécessaires à la mise en œuvre d'une ressource complexe

La première solution est basée sur un calculateur supportant ADA (mini, micro, station de travail...) communiquant avec un automate industriel par l'intermédiaire d'une ligne série ou d'un réseau local industriel (MAP, LAC, FACTOR, UNITELWAY,...) [Thomesse 1985].

La seconde solution est basée sur un micro ordinateur industriel supportant ADA (PC,

PS/2, VME 680x0,...) disposant, de base, de cartes d'entrées/sorties industrielles. Cette solution a l'avantage d'un coût modeste.

La troisième solution met en œuvre des entrées/sorties déportées connectées, par l'intermédiaire d'un réseau de terrain (PROFIBUS,...), à un calculateur (industriel ou non) supportant ADA. L'intérêt des entrées/sorties déportées est qu'elles permettent de simplifier le câblage et de valider l'installation sans programmation [Pradenc 1991]. De plus, si ces entrées/sorties déportées sont pourvues d'une intelligence, elles permettent d'intégrer différentes fonctions de filtrage numérique, de détection, de recouvrement défaillance,... au plus près des capteurs et actionneurs [Robert et al 1992].

Enfin la dernière solution proposée consiste à mettre en œuvre un calculateur-automate. Ce calculateur hybride est composé de deux unités de calcul coopérant par l'intermédiaire d'un bus interne haut débit. L'une de ces unités de calcul est du type automate avec ses entrées/sorties industrielles et l'autre est du type mini ou micro ordinateur. Un des premiers exemples de ce type de calculateur est le Pyramid Integrator d'Allan-Bradley qui comporte des modules automates et un module mini-ordinateur VAX [JR 1988].

Il est à noter que pour nous, un "calculateur supportant ADA" ne correspond pas obligatoirement à un calculateur sur lequel est implémenté un compilateur ADA, ce peut être un calculateur intégrant un run-time ADA et dont le développement logiciel s'effectue à l'aide d'un compilateur croisé sur une autre machine.

Pour les ressources de type complexe, les calculateurs intégrant la gestion du filtre comportemental sont connectés à un réseau informatique (cf. sur figure III.2 : réseau supérieur) par lequel vont transiter les requêtes de service ainsi que les accusés de réception.

Pour les ressources de type simple, les requêtes de services sont des requêtes valides. De fait, le réseau par lequel vont transiter les requêtes de service sera directement connecté au calculateur gérant les graphes de commande complets.

Ces réseaux sur lesquels vont circuler les requêtes de services et accusés de réception correspondant seront de préférence haut débit et du type hétérogène pour des raisons d'ouverture (Ethernet, Mapway, LAC,...) car, en plus de ces communications, ils doivent permettre le téléchargement de programmes d'usinage, la reconfiguration des machines, le rapatriement de données, ...

Pour ce qui est du Suivi des Gammes Opératoires, nous pouvons implanter les divers

niveaux des gammes opératoires étendues ainsi que les macro gammes opératoires étendues sur des calculateurs dédiés aux ressources ou ensemble de ressources virtuelles.

Pour implanter le Niveau Décisionnel Supérieur (Ordonnancement et Planification) nous devons décider d'une solution répartie ou d'une solution centralisée. Le choix d'une solution répartie se justifie dans le cas où l'approche globale est trop complexe pour suivre les aléas du système [Hillion et al 1988]. Dans ce cas, chaque ressource virtuelle ou ensemble de ressources virtuelles peut intégrer un ordonnancement local [Erschler et al 1988]. Dans le cas centralisé, on considère la cellule dans sa totalité [Bieselaar et al 1988] [Hammadi 1991].

A titre indicatif, la figure III.3 montre un exemple d'architecture informatique répartie pour la commande de la cellule développée dans le chapitre II.

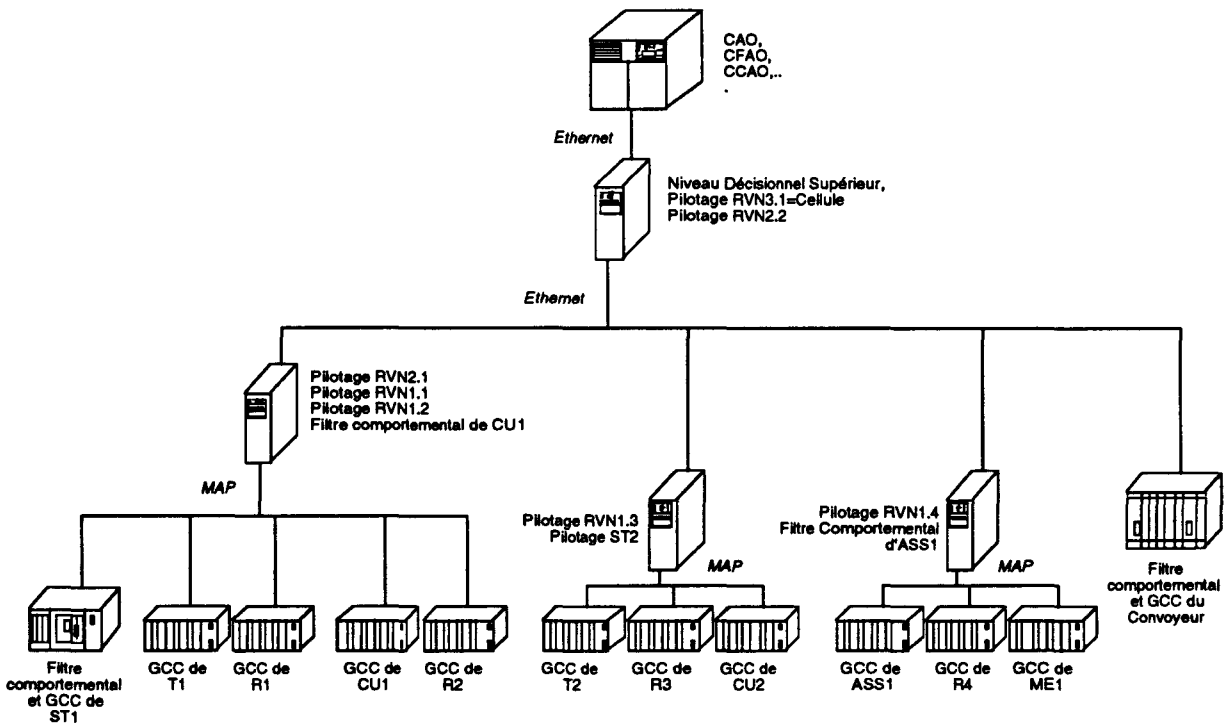


Figure III.3 : Exemple d'architecture informatique répartie et hiérarchisée pour la commande de la cellule décrite au chapitre II

III.B.4. Environnement logiciel

Pour les échanges de données entre les différents équipements informatiques et d'automatisation, il est intéressant de mettre en place une plate-forme d'intégration CIM. Celle-ci se présente sous la forme d'une boîte à outils logiciels et permet, entre autre, d'intégrer facilement des équipements hétérogènes et de rendre les applicatifs indépendants des

spécificités des réseaux et des composantes d'automatisation installées dans l'atelier. Actuellement, plusieurs constructeurs proposent ce type d'outil : Base Star de Digital, Cimview de Cimtech, PFS-DAE et Plant Works d'IBM, IPT de Helwet Packard,... Ces logiciels permettent une meilleure intégration des matériels et progiciels. Ils facilitent les acquisitions de données, la mise à jour de bases de données, l'animation de synoptiques,... dans l'optique d'effectuer le contrôle qualité, le suivi de production, la maintenance,...

Pour notre part, nous avons travaillé sous environnement VMS (Digital), nous disposons de l'environnement de programmation d'ADA (APSE) facilitant le développement : éditeur syntaxique LSE (Langage Sensitive Editor), analyseur statique de code SCA (Source Code Analyser), analyseur dynamique de code PCA (Performance Coverage Analyser),... Nous disposons aussi d'un ensemble de paquetages tel le paquetage X nous permettant de construire nos interfaces graphiques compatibles avec la norme X11 [Stocker 1991]. Enfin, comme plate-forme d'intégration, nous avons utilisé Base Star de Digital [Stephen et al 1992].

III.C. Notions avancées sur ADA

III.C.1. Introduction

Le langage ADA possède, comme tout langage généraliste de haut niveau, des structures de données, des structures de contrôle, etc.... Par contre, les particularités propres à ADA en font son intérêt; celles-ci sont principalement : la gestion du multitâche, le concept des paquetages, le concept d'unités génériques et le traitement des exceptions. Nous allons donc rappeler succintement l'ensemble de ces notion et concepts du langage ADA.

III.C.2. ADA et le temps-réel

Intrinsèquement, le langage ADA est un langage qui s'intègre dans le contexte temps réel. Il permet d'en implanter, plus ou moins facilement, les paradigmes correspondants : gestion de processus parallèles indépendants, communications inter processus, gestions d'interruptions...

Mais avant de mettre en œuvre ces paradigmes, il est intéressant de comprendre comment le langage ADA gère le parallélisme.

III.C.2.a. Philosophie du parallélisme ADA

En ADA, la programmation des processus parallèles s'effectue explicitement sous formes

d'unités de programme appelées *tâches*. Celles ci se décomposent en deux parties : la spécification déclarée par la primitive *task* pour les tâches individuelles ou *task type* pour les types tâches et le corps déclaré dans les deux cas par la primitive *task body*. Pour paralléliser l'ensemble de celles-ci, le noyau d'ADA intègre un mécanisme de gestion de tâches basé sur le concept des **processus séquentiels communicants de Hoare** [Hoare 1978]. Ainsi, à chaque tâche est associé un processus logique indépendant. Ces processus, gérés par un ordonnanceur, peuvent prendre quatre états : inexistants, éligibles, élus ou en attente (Figure III.4) [Briand 1991].

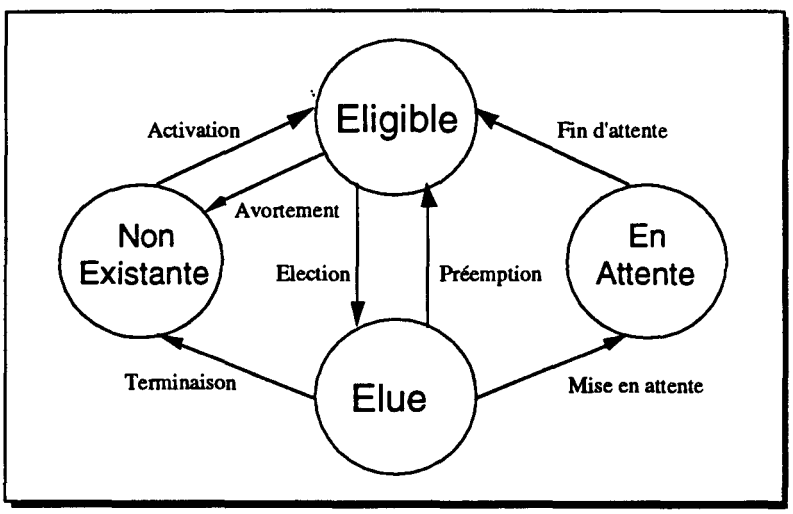


Figure III.4 : Etats d'une tâche ADA

Pour qu'il y ait effectivement commutation et donc simultanéité (indépendamment du système d'exploitation), l'ordonnancement des tâches ne peut avoir lieu que si ces dernières effectuent des échanges avec d'autres tâches ou avec l'environnement extérieur. Ces échanges assimilables à des entrées/sorties, peuvent être :

- des rendez -vous (communication avec d'autres tâches),
- des interruptions,
- des temporisations,...

III.C.2.b. Mécanismes de base pour les communications inter-tâches

Le **rendez-vous** est le seul mécanisme de base d'ADA pour permettre aux tâches de communiquer entre elles. Les rendez-vous doivent êtres déclarés dans la partie spécification des tâche à l'aide de l'instruction *entry* et utilisés dans les corps des tâches à l'aide de l'instruction *accept* (Figure III.5).

La communication entre tâches s'effectue lorsque une tâche appelante établit un rendez-vous (appel pointé) avec une tâche appelée. Plusieurs tâches peuvent bien sûr effectuer les

mêmes rendez-vous entre elles, les entrées sont alors mise dans une fifo. Lors de ces rendez-vous, il peut y avoir un échange unidirectionnel ou bidirectionnel de paramètres entre tâches appelantes et tâches appelées (paramètres formels déclarés en *in* et/ou *out*).

En se basant sur les mécanismes de base du rendez-vous, il est possible de mettre en œuvre différents types de communications inter-tâches. Ces différents types de communication se caractérisent par leur degré de synchronicité.

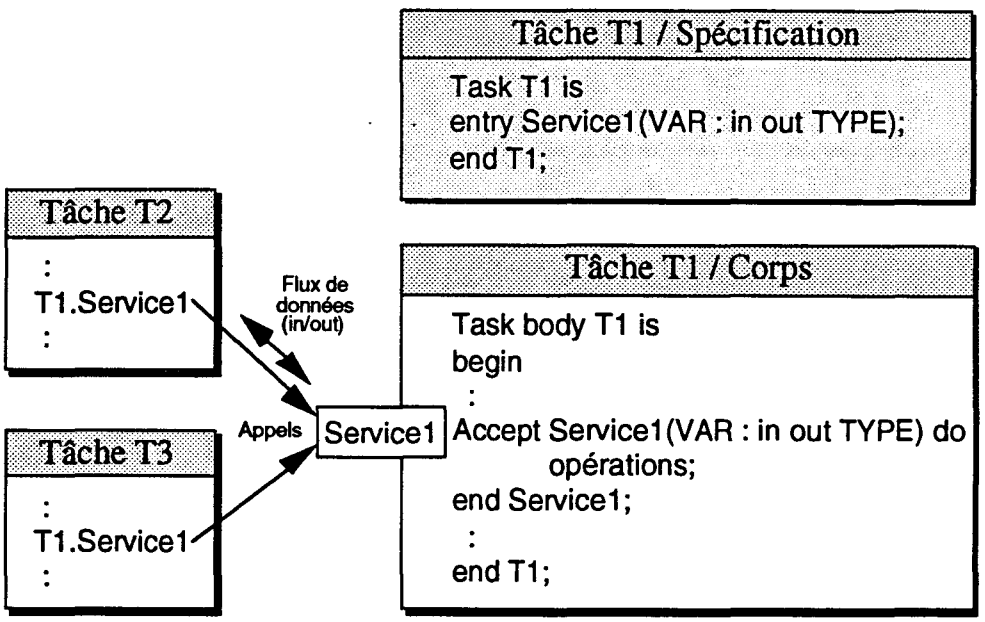


Figure III.5 : Les rendez-vous ADA

Si nous considérons uniquement deux tâches (client et serveur), il n'est possible d'effectuer que des communications à caractère synchrone, le rendez-vous ADA étant fondamentalement synchrone. Ces communications pourront être soit fortement synchrones soit faiblement synchrones.

Dans la communication fortement synchrone, dont la figure III.6 décrit le mécanisme sous forme de Réseaux de Petri, la tâche appelante est bloquée tant que le service offert par la tâche appelée (ensemble d'opérations) n'est pas terminé. Dans le formalisme HOOD, ce type de communication est dénommé HSER (Highly Synchronous Execution Request) [HOOD 1991] [Delatte et al 1993].

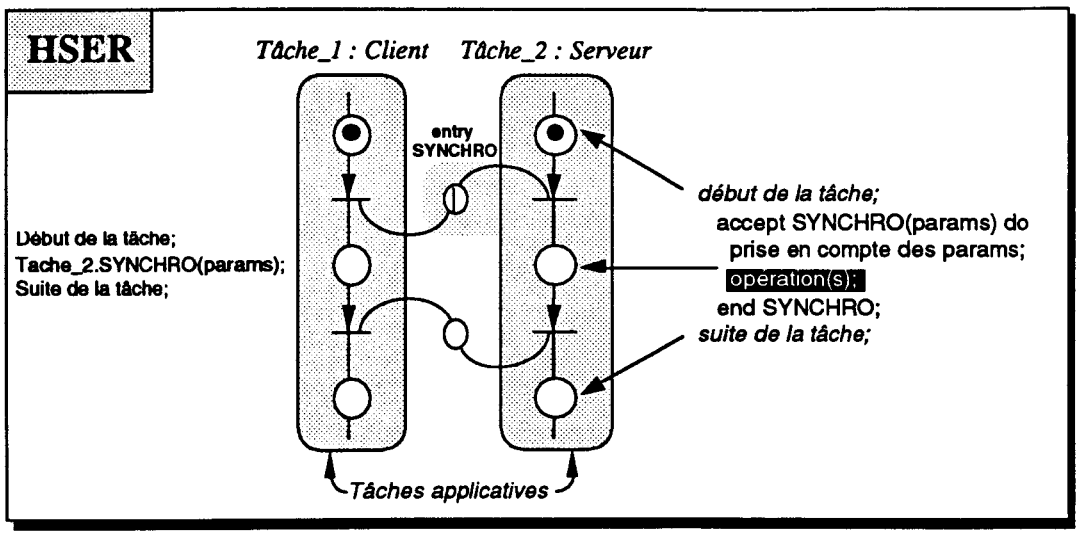


Figure III.6 : Communication inter tâches fortement synchrone

Dans la communication faiblement synchrone, dont la figure III.7 décrit le mécanisme, la tâche appelante est suspendue uniquement pour permettre l'échange de données avec la tâche appelée. Cette dernière exécute le service en dehors de la communication. Dans le formalisme HOOD, ce type de communication est dénommé LSER (Losely Synchronous Execution Request).

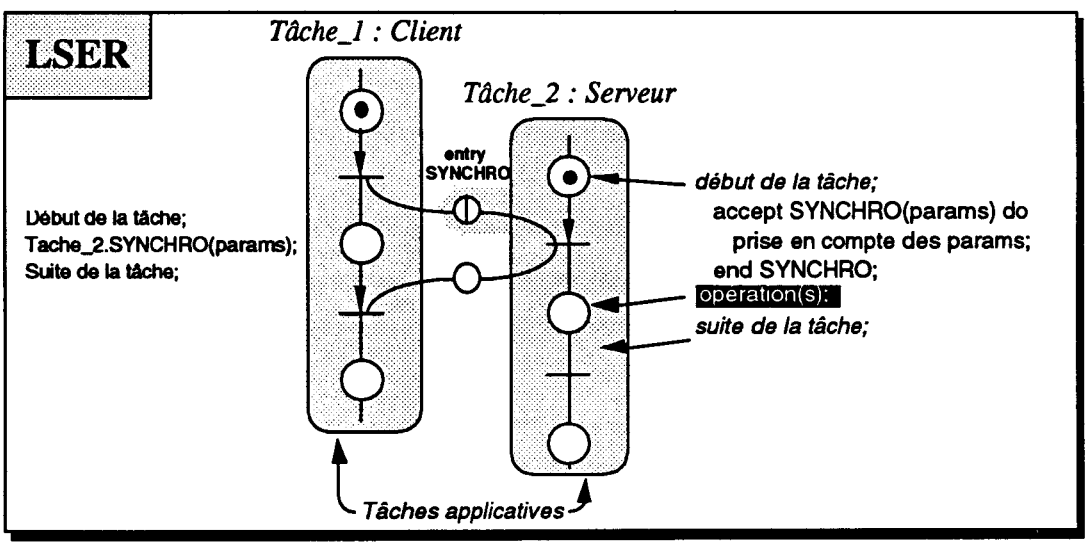


Figure III.7 : Communication inter tâches faiblement synchrone

Dans tous les cas, le couplage entre tâches reste fort, en effet, la tâche appelée reste bloquée sur l'*accept* tant qu'elle n'est pas sollicitée par une tâche appelante. De plus, la tâche client se doit de connaître exactement l'identité de la tâche serveur. Dans la partie qui suit, nous allons voir comment contourner ces restrictions.

III.C.2.c. Extension des communications inter-tâches

Nous pouvons étendre les possibilités de communication inter-tâches en ajoutant au moins une tâche intermédiaire dénommée *agent* [Barnes 1988] dans le but de construire ce que l'on appelle des *channels* dans la méthodologie DARTS (Design Approach for Real-Time Systems) [Calvez 1990].

Construire un channel entre deux tâches permet de désynchroniser plus ou moins la tâche émettrice de la tâche réceptrice. Cela nous apporte la possibilité d'accéder à des communications inter-tâches asynchrones. De plus, un des grands intérêts de cette notion de channel est qu'elle fait abstraction de l'identité de la ou des tâches réceptrices.

Ces channels peuvent être construits à l'aide de quatre types de tâches : la tâche serveur, la tâche avare, la tâche prodigue et la tâche tiroir [Briand 1991].

Pour notre part, nous nous intéresserons uniquement aux channels créés à l'aide de tâches intermédiaires de type *tiroir*. Celles-ci sont basées sur le mécanisme de la **Boîte Aux Lettres** (BAL) (En anglais: MBX : Mail BoX). Dans ce mécanisme, la tâche émettrice **(producteur)** écrit ses données à transmettre dans une boîte aux lettres dans laquelle la tâche réceptrice **(consommateur)** lit ces données (Figure III.8).

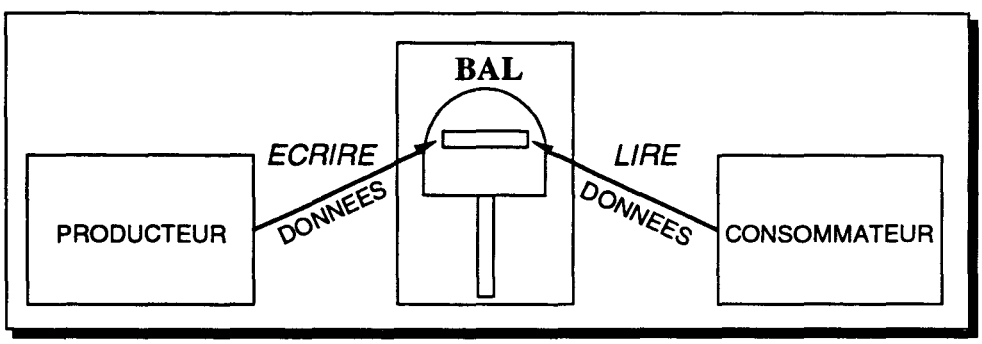


Figure III.8 : Mécanisme de la Boîte Aux Lettres

Nous proposons de mettre en œuvre deux types de boîtes aux lettres, leurs différences se situant principalement au niveau du degrés de désynchronisation entre les tâches applicatives. Le premier type de boîtes aux lettres équivaut à une communication faiblement synchrone. La figure III.9 décrit le mécanisme interne de celles-ci sous forme de Réseau de Petri. Avec ce type de boîte aux lettres, le consommateur est bloqué momentanément si il y a une absence de données.

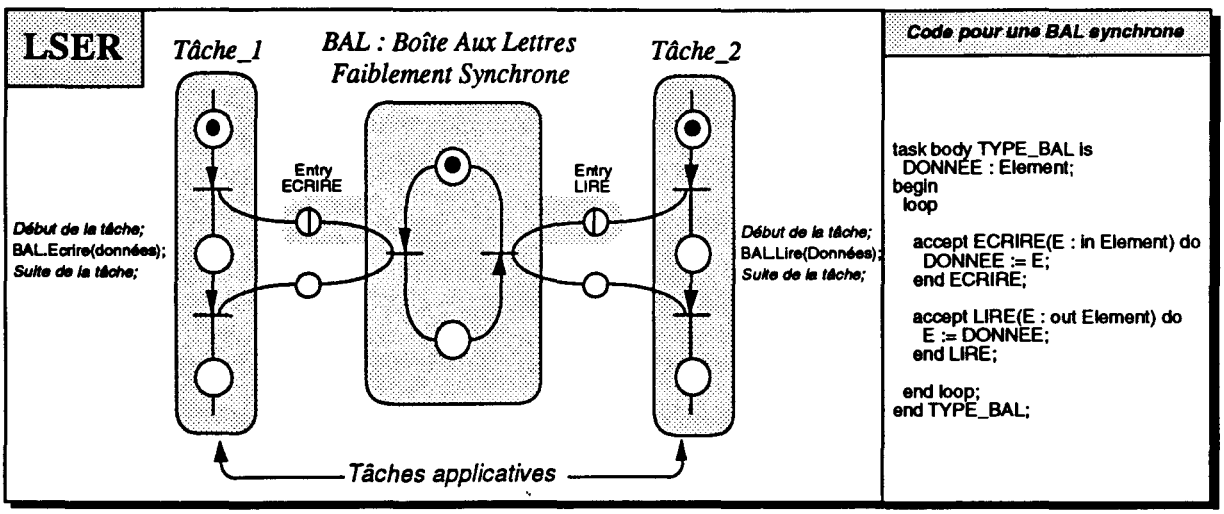


Figure III.9 : Communication par Boîte Aux Lettres Faiblement Synchrone

Dans certains cas, ce dernier point peut être gênant; un second type de boîte aux lettres peut alors être mis en œuvre. La figure III.10 décrit le fonctionnement de celle ci à l'aide de Réseaux de Petri à arcs inhibiteurs. Nous pouvons comparer cette communication à une communication asynchrone. Dans le formalisme HOOD, ce type de communication est dénommé ASER (ASynchronous Execution Request). Dans ce cas, l'utilisation d'un tampon de type FIFO permet au producteur d'émettre indépendamment du consommateur. Quant à ce dernier, il effectue des lectures dans la boîte aux lettres quand bon lui semble. De plus, il n'est plus bloqué par une absence de données grâce à une détection de cet état.

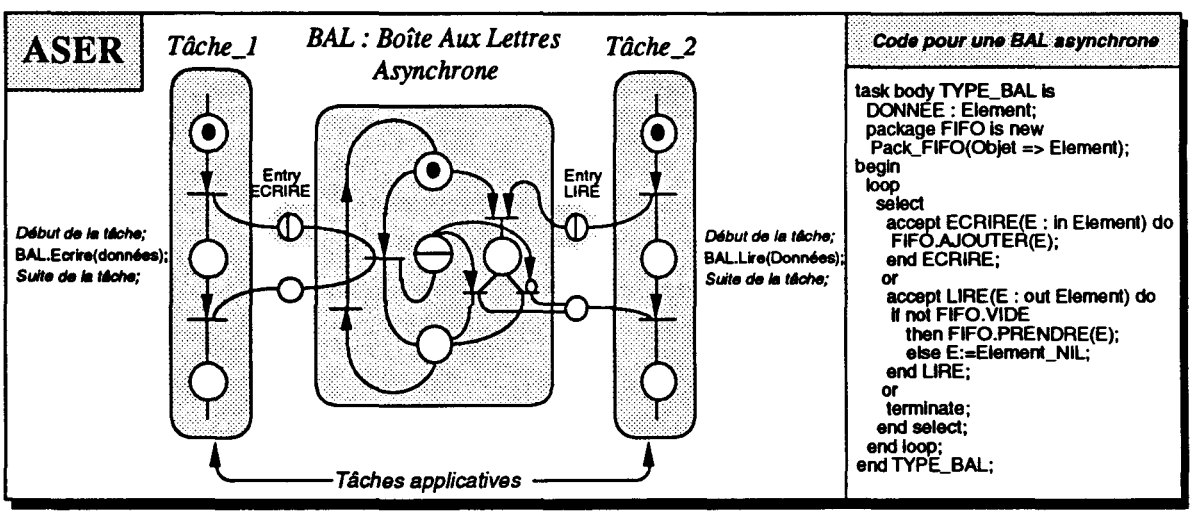


Figure III.10 : Communication par Boîte Aux Lettres Asynchrone

Contrairement aux rendez-vous directs entre tâches applicatives, l'utilisation de boîtes aux lettres permet de symétriser la communication. En effet, il n'y a plus de différence de structure entre le producteur et le consommateur. De plus, dans le contexte qui est le notre où nous recherchons un faible couplage entre les entités et une réutilisabilité maximale, l'intérêt de ces boîtes aux lettres est qu'elles permettent de masquer l'identité du consommateur vis à vis du producteur et inversement, la boîte aux lettres étant la seule entité visible à la fois par l'un et par l'autre.

III.C.3. Les paquetages

Les paquetages font partie des fondements d'ADA. Ce sont des unités de programme, pouvant être compilées séparément, qui encapsulent un ensemble logique de données et/ou de services. Les paquetages se décomposent en deux parties : la spécification définie par la clause *package* et le corps défini par la clause *package body*. La spécification représente l'interface avec le monde extérieur des services que le paquetage sera amené à délivrer tandis que le corps décrit les détails de l'implantation de ces services.

Ces deux parties, spécification et corps, peuvent être compilées séparément. Ainsi, la spécification peut être définie alors que le corps est encore inexistant. Ce point est intéressant en phase conception des applications de taille et de durée de vie importantes car il facilite le développement de prototypes incrémentaux. De plus, l'Ada Programming Support Environment (APSE) fournit tous les outils permettant de gérer facilement les bibliothèques de paquetages.

L'utilisation d'un paquetage s'effectue par l'intermédiaire de la clause *with* et l'utilisateur ne voit que la partie visible de celui ci, c'est à dire l'interface, l'implémentation reste cachée.

Cette séparation entre spécification et corps permet au concepteur, une fois la spécification arrêtée, d'améliorer l'implémentation d'un paquetage sans que l'utilisateur n'ait à modifier quoi que ce soit.

A titre d'illustration, nous allons décrire un paquetage permettant la construction d'une pile d'entiers dont la taille maximale est fixée à 30 éléments.

La spécification est la suivante :

```
Package GEST_PILE is
   procedure PUSH(Elem : in integer);
   procedure POP(Elem : out integer);
   PILE_PLEINE, PILE_VIDE : exception;
end GEST_PILE;
```

Le corps est le suivant :

```
package body GEST_PILE is

   TAILLE : constant := 30;
   INDEX : integer range 0..TAILLE := 0;
   ESPACE : array (1..TAILLE) of integer;

   procedure PUSH(Elem : in integer) is
      begin
         if INDEX=TAILLE then
            raise PILE_PLEINE;
         else
            INDEX:=INDEX+1;
            ESPACE(INDEX):=Elem;
         end if;
   end PUSH;

   procedure POP(Elem : out integer) is
      begin
         if INDEX=0 then
            raise PILE_VIDE;
         else
            Elem:=ESPACE(INDEX);
            INDEX:=INDEX-1;
         end if;
   end PUSH;
end GEST_PILE
```

Ensuite, une fois compilé et mis en bibliothèque, ce paquetage pourra être utilisé de la façon suivante :

```
with GEST_PILE;                 --pour rendre visible l'interface du paquetage
:
GEST_PILE.PUSH(ALPHA);          --pour empiler un entier référencé par ALPHA
:
GEST_PILE.POP(BETA);            --pour dépiler un entier référencé par BETA
:
```

L'utilisation des paquetages permet donc d'assurer la modularité, le masquage de l'information et l'abstraction des données. Ainsi, du point de vue de l'approche orientée objets, nous pouvons considérer qu'**un paquetage est en quelque sorte un objet**.

III.C.4. La généricité

Pour favoriser la construction de composants logiciels réutilisables, ADA inclut un mécanisme de déclaration d'unités génériques. Ces unités peuvent être soit des sous programmes soit des paquetages précédés de la clause *generic* et d'un ensemble de paramètres formels génériques.

Les unités génériques sont des modèles qu'il faut instancier en substituant aux paramètres génériques formels des paramètres effectifs correspondant à l'utilisation souhaitée. Ces paramètres peuvent être des structures de données mais aussi des types ou des sous

programmes. A chaque instanciation, il y a création d'un nouveau paquetage ou d'un nouveau sous programme qui peut être normalement utilisé.

A titre d'illustration, nous allons voir comment rendre générique le paquetage de gestion de piles d'entier afin qu'il permette de construire des piles de taille quelconque gérant tous type d'éléments.

La spécification est la suivante :

```
generic
   TAILLE : natural;                    --paramètre formel générique pour définir la taille de la pile
   type ELEMENT is limited private;     --paramètre formel générique pour définir les objets gérés

package GEST_PILE is
   procedure PUSH(Elem : in ELEMENT);
   procedure POP(Elem : out ELEMENT);
   PILE_PLEINE, PILE_VIDE : exception;
end GEST_PILE;
```

Le corps est alors le suivant :

```
package body GEST_PILE is

   INDEX : integer range 0..TAILLE := 0;
   ESPACE : array (1..TAILLE) of ELEMENT;

   procedure PUSH(Elem : in ELEMENT) is
      begin
         if INDEX=TAILLE then
            raise PILE_PLEINE;
         else
            INDEX:=INDEX+1;
            ESPACE(INDEX):=Elem;
         end if;
   end PUSH;

   procedure POP(Elem : out ELEMENT) is
      begin
         if INDEX=0 then
            raise PILE_VIDE;
         else
            Elem:=ESPACE(INDEX);
            INDEX:=INDEX-1;
         end if;
   end PUSH;

end GEST_PILE
```

Ensuite, une fois compilé et mis en bibliothèque, ce paquetage générique pourra être utilisé de la façon suivante :

```
with GEST_PILE;                         --pour rendre visible l'interface du paquetage générique
:
-- nous allons définir l'objet MA_PILE permettant d'empiler jusqu'à 20 éléments de type
   TYPE_JETON
```

```
package MA_PILE is new GEST_PILE(TAILLE      => 20,
                                 ELEMENT     => TYPE_JETON);
:
MA_PILE.PUSH(UN_JETON);        --pour empiler un élément de type TYPE_JETON
:
MA_PILE.POP(UN_JETON);         --pour dépiler un élément de type TYPE_JETON
:
```

Nous voyons que les paquetages génériques permettent la réutilisation de composants logiciels standards, ce qui nous permet de considérer **un paquetage générique comme une classe d'objets.**

Il est également possible de simuler des métaclasses. En effet, ADA autorise la déclarations de paquetages génériques dans la spécification d'un paquetage générique [Bailly et al 1987].

III.C.5. Les Exceptions

En ADA, une exception désigne un événement anormal qui provoque la suspension de l'exécution normale du programme. Celles ci peuvent être de deux natures. En premier lieu, ce sont des exceptions prédéfinies faisant partie intégrante du langage et correspondant à des erreurs numériques (*NUMERIC_ERROR*), à des erreurs de tâches (*TASKING_ERROR*), ... En second lieu, ce sont des exceptions définies par le concepteur du logiciel afin de faire face à des situations critiques tels le débordement de piles, les défaillances matérielles,...

De fait, la levée des exceptions permet de corriger des situations dégradées afin d'obtenir des systèmes fiables.

III.C.6. Les Interruptions

En ADA, l'interruption matérielle est traitée comme un appel d'entrée sur la tâche gérant l'interruption. Pour obtenir une réponse efficace, le rendez-vous déclenché est prioritaire par rapport aux autres rendez-vous pouvant exister entre les autres tâches.

L'appel d'entrée doit spécifier l'adresse matérielle de l'interruption. Par conséquent, l'utilisation des interruptions dépend fortement du système sur lequel ADA est implanté. De plus, le contrôle des interruptions n'est pas toujours disponible. Par exemple, en VAX ADA, celui-ci n'est pas implémenté, il est donc nécessaire de passer par les routines systèmes asynchrones (AST : Asynchronous System Trap).

III.D. Réalisation effective du logiciel

III.D.1. Introduction

Notre démarche d'implantation s'inscrit naturellement dans le cadre du cycle de vie global d'un Système Automatisé de Production (Cycle de vie en V d'un SAP) [Annexe I]. Dans ce sens et pour mener à bien la réalisation effective du système de conduite de notre SAP, nous devons passer par différentes phases telles que le codage, le test unitaire, l'intégration, la validation globale, l'exploitation,...

Si l'on considère l'architecture fonctionnelle de contrôle d'un SFPM définie au chapitre II (Figure II.57), nous nous intéresserons uniquement à l'implantation des gestionnaires de ressources, du suivi des gammes opératoires étendues et d'une version simplifiée du niveau décisionnel supérieur. La conception des autres parties fait actuellement l'objet d'études annexes et vue la modularité de notre approche, il sera facile de greffer l'implantation de ces partie sur l'ensemble obtenu.

Le choix du langage ADA comme principal langage d'implantation nous permet de travailler dans le contexte de la programmation basée objet [Booch 1988] [Booch 1992]. Ainsi, tous les aspects objets inhérents à la modélisation de la commande de notre SFPM, abordés dans le chapitre II, pourront être conservés : modularité, réutilisabilité, généricité, maintenabilité,... Dans ce sens, nous montrerons comment les modèles développés dans cette phase de conception (Suivi des Gammes Opératoires Etendues (SGOE), ALlocateurs de Ressources (ALR), Filtres Comportementaux (FC),...) peuvent être traduits de manière systématique en un code ADA directement utilisable et bien sûr réutilisable.

A terme, l'intégration de l'ensemble du code produit ainsi que sa répartition sur l'architecture informatique répartie du site de production doit permettre la mise en production immédiate. Mais avant l'implantation effective, il est nécessaire, afin de valider le comportement dynamique de l'ensemble de la commande de coordination, de passer par une phase de validation hors ligne. Nous avons choisi de réaliser cette phase de validation à l'aide d'une simulation événementielle, ce type de simulation convenant particulièrement bien aux systèmes à événements discrets que sont les Systèmes Flexibles de Production Manufacturière. De plus, dans un souci d'adaptation, d'efficacité, de modularité et de rapidité nous avons minimisé les parties de code différenciant le logiciel de simulation du logiciel d'exploitation.

Ainsi, seul le code correspondant aux graphes de commande complets des ressources simples et des ressources complexes est absent de la partie commune au logiciel de simulation

et au logiciel d'exploitation. Nous nommons cette partie commune dont la figure III.11 décrit la composition : **Base Logicielle Commune (BLC)** [Huvenoit et al 1995].

Afin de conserver la structure élaborée en phase de conception, la Base Logicielle Commune sera composée d'un ensemble d'objets, les **objets racines,** plus ou moins complexes car pouvant comporter plusieurs sous niveaux d'objets enfants. Ces objets racines peuvent être de trois types :

-des objets effectuant le Suivi des Gammes Opératoires Etendues,

-des objets correspondant aux gestionnaires de ressources simples ou complexes amputés des graphes de commande complets,

- des objets correspondant au niveau décisionnel supérieur.

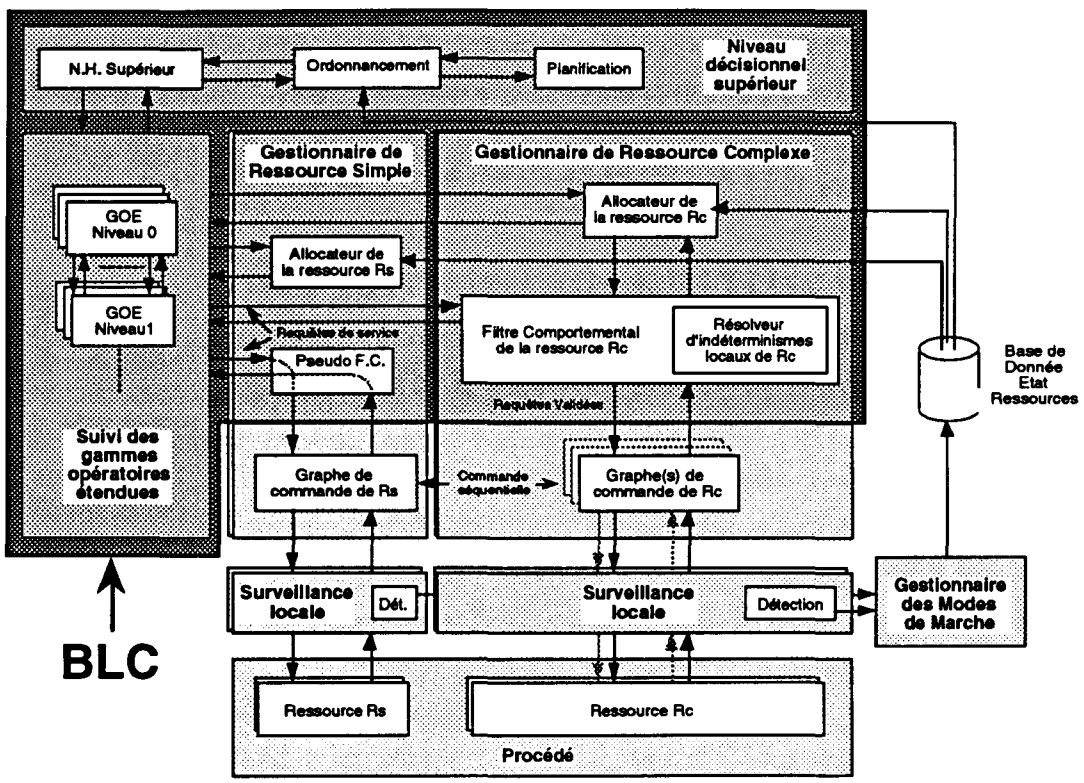


Figure III.11 : Composition de la Base Logicielle Commune

Notre objectif final étant de concevoir un atelier de génie automatique basé sur la méthodologie CASPAIM, nous devons développer un ensemble de méthodes systématiques permettant d'automatiser ou tout au moins d'assister le passage de la conception à l'implantation de ces objets. Ces méthodes seront intégrées dans divers outils logiciels qui, réunis, constitueront cet atelier de génie automatique. La figure III.12 propose une synthèse de l'ensemble des outils logiciels qu'il serait nécessaire de développer.

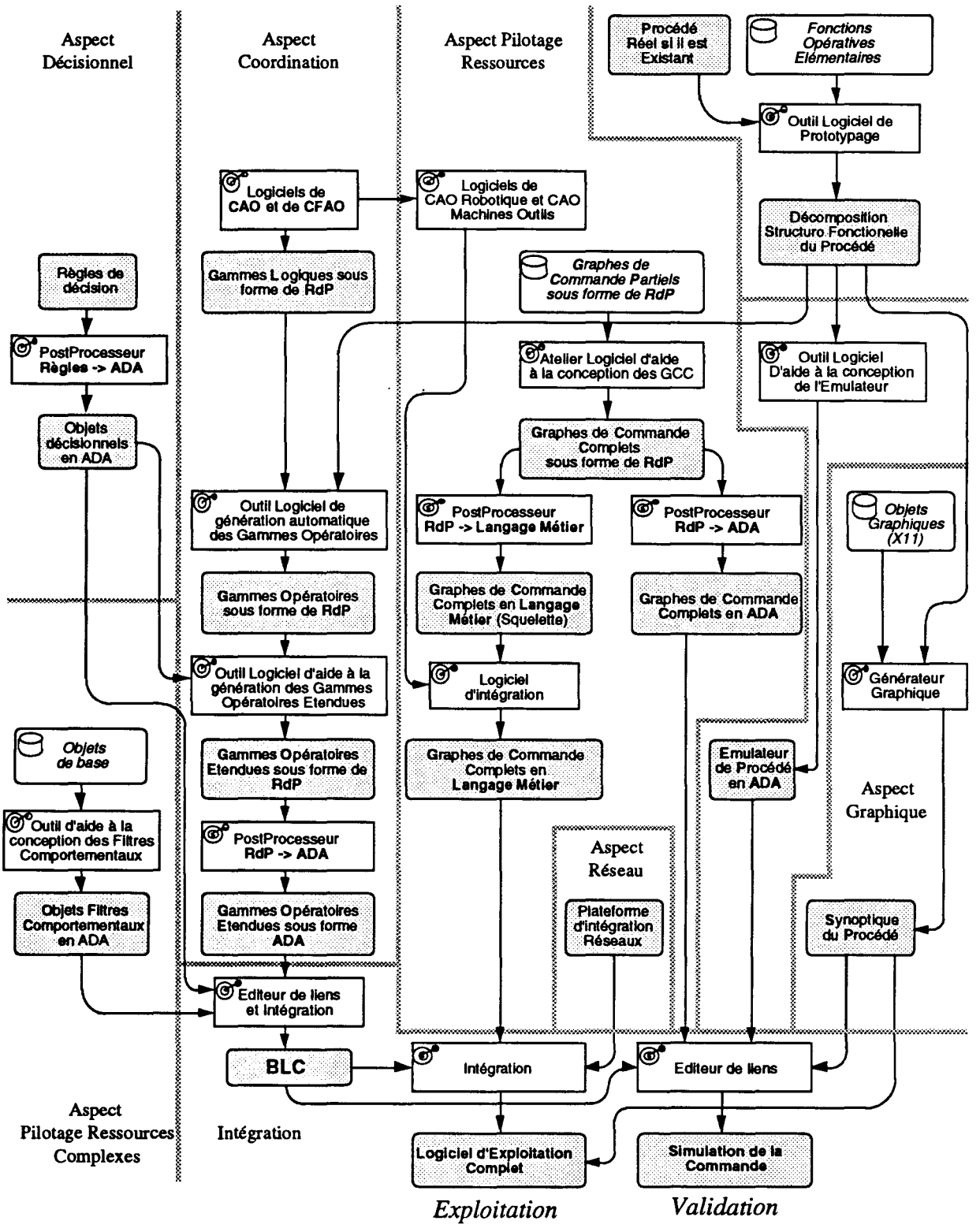


Figure III. 12 : Synthèse des outils logiciels nécessaires à la conception des logiciels de validation et d'exploitation

Nous pouvons voir que pour obtenir le logiciel de validation par simulation, nous associons à la BLC les graphes de commande complets des ressources traduits en ADA à l'aide

d'un postprocesseur ainsi qu'un émulateur de procédé écris lui aussi en ADA. Cet émulateur est généré en se basant sur la décomposition structuro-fonctionelle du procédé [Amar 1994].

Par contre, pour obtenir l'ensemble des logiciels d'exploitation, les graphes de commande sont traduits en langage métier (Grafcet, langage automate,...) à l'aide d'un postprocesseur dédié. Cette traduction donne en fait un squelette qu'il faut compléter avec les sous programmes correspondant aux opérations à effectuer (programmes d'usinage, programmes de transfert,...). Ces graphes de commande complets seront implantés sur les organes informatiques contrôlant directement les ressources de production (Automates Programmables Industriels, Commandes Numériques...).

Nous notons que très peu d'outils logiciels sont spécifiques à la phase de validation ou à la phase d'exploitation. Même le synoptique du procédé généré à l'aide d'un générateur graphique peut être utilisé en simulation ou en exploitation pour décrire l'état du système.

Dans les parties qui vont suivre, nous allons détailler les principes de génération de la Base Logicielle Commune ainsi que les extensions conduisant à l'obtention d'une part du logiciel de simulation et d'autre part des logiciels d'exploitation.

III.D.2. La Base Logicielle Commune

III.D.2.a Généralités et conventions

Afin de permettre une totale modularité, de réduire la visibilité de chaque objet, de faciliter la compilation séparée et enfin de permettre, a posteriori, tout type d'implantation répartie, toutes les communications inter objets racines dans la Base Logicielle Commune (REQ, ACK, NACK) se feront par l'intermédiaire de boîtes aux lettres (BAL) faiblement synchrones ou asynchrones dont les mécanismes ont été précédemment décrits.

Pour ne pas réduire les performances, ni augmenter la complexité, les communications intra-objets, lorsqu'elles sont nécessaires, ne se feront pas par l'intermédiaire de boîtes aux lettres mais à l'aide des primitives standards (ACCEPT direct).

Les conventions graphiques que nous allons utiliser pour représenter la mises en œuvre de ces différents types de communications sont décrites par la figure III.13.

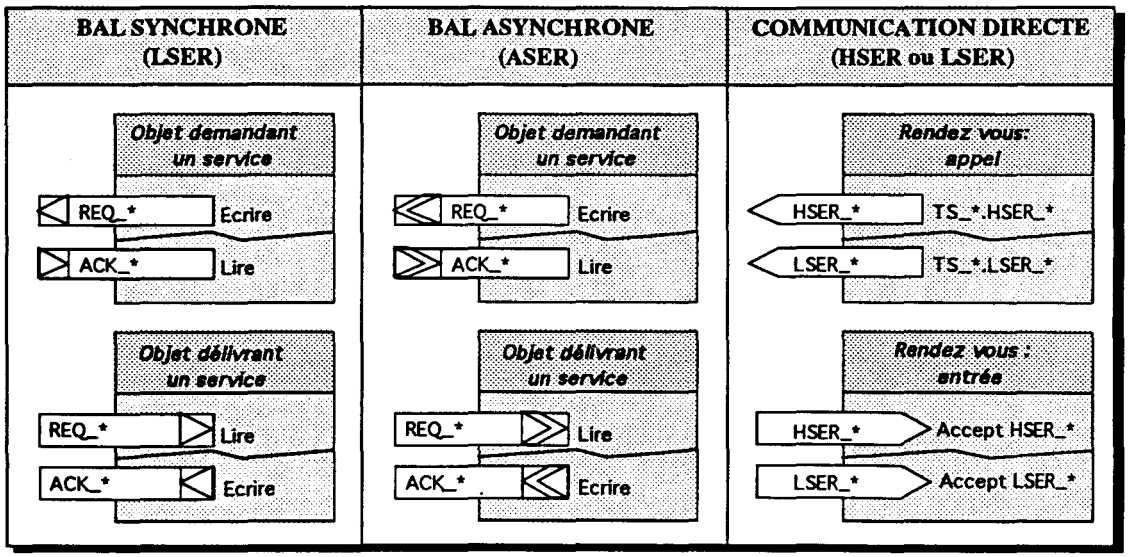


Figure III.13 : Conventions graphiques utilisées lors de la mise en place de communications inter-tâches

Le plus souvent, les requêtes et accusés de réception seront du type synchrone. Dans certains cas, comme par exemple pour les requêtes d'allocation de ressources, elles seront de type asynchrone.

La BLC et les graphes de commande complets, que ce soit en simulation ou en exploitation, échangent leurs données par l'intermédiaire d'une interface normalisée. Cette dernière est composée de boîtes aux lettres synchrones dénommées IBG_REQ_* pour les requêtes et IBG_ACK_* pour les accusés de réception (IBG : Interface entre la Base logicielle commune et les Graphes de commande complets).

Pour la représentation des objets, nous utiliserons un formalisme graphique inspiré de la méthode HOOD [HOOD 1992].

III.D.2.b. Le suivi des Gammes Opératoires Etendues:

i. Gestion dynamique des GOE

Une gamme opératoire étendue décrit précisément la suite des actions à réaliser pour un type de produit. Un jeton dans une GOE va caractériser un produit en cours de production.

Pour réaliser le suivi des gammes opératoires étendues, la solution mettant en œuvre un joueur de réseaux de Petri s'est avérée inappropriée car trop lente. Nous nous sommes donc orientés vers une solution traduisant les réseaux de Petri non pas en une structure de données

mais en une structure de programme très efficace et directement utilisable, celle ci étant, par essence flexible et très orientée implantation.

Ainsi, dans notre approche, nous associons un objet racine par gamme opératoire étendue ou sous gamme opératoire étendue. Ces objets sont implantés sous forme de paquetages principalement composés :

-d'un type tâche décrivant la G.O.E.,

-d'une tâche administrant la gestion dynamique de ce type tâche.

La mise en œuvre des gammes opératoires étendues possède un **double parallélisme**, en effet, nous avons :

- un parallélisme dans les types de produit simultanément opérationnels et
- un parallélisme dans un même type où il peut y avoir plusieurs produits.

La gestion dynamique du type tâche a été mis en place afin de garantir la **réentrance** du modèle initial découlant du second type de parallélisme. La solution retenue ne correspond pas à une réentrance directe du code ADA mais à la création dynamique d'objets gammes. La figure III.14 décrit plus précisément le mécanisme mis en œuvre.

Lorsqu'une pièce brute est disponible sur l'un des lieux d'entrée acceptés par sa gamme opératoire étendue et que l'on désire lancer le déroulement de celle ci (REQ_GOE_3), la tâche de gestion duplique le type tâche afin d'engendrer une tâche active. Cette création est effectuée en affectant un numéro de Gamme Active (GA n°i) puisé dans une pile ou sont stockés les numéros disponibles. Une fois la GOE de la pièce terminée, la tâche correspondante est détruite, le numéro de gamme active est remis dans la pile et un accusé de réception est retourné (ACK_GOE_3).

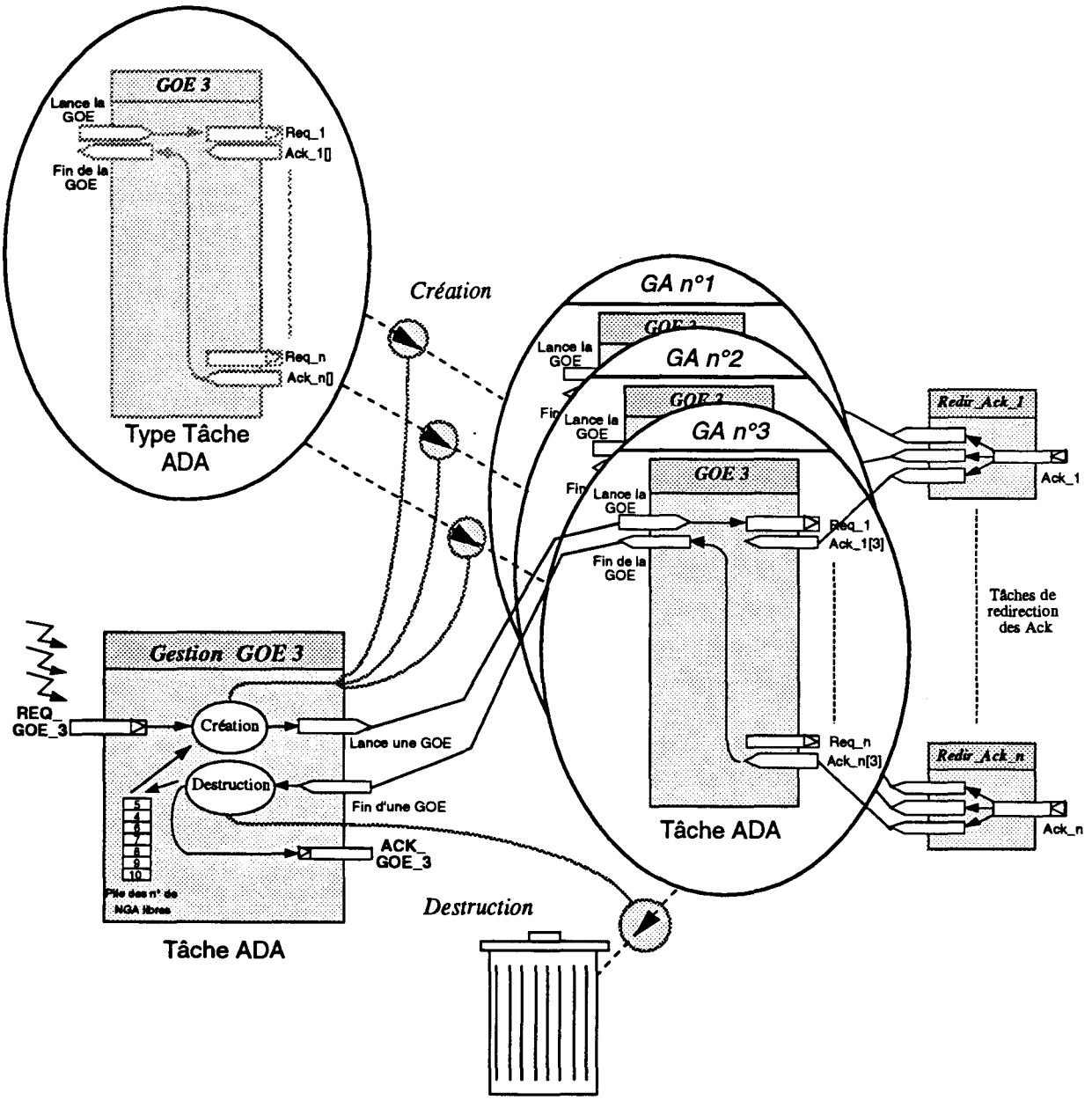


Figure III.14 : Mécanisme de gestion dynamique des GOE

Dans ces objets racines, il est nécessaire d'inclure des **tâches de redirection des accusés de réception** pouvant provenir des autres objets racines de la BLC (Objets Ressources, Objets Décisionnels, Objets Gammes de niveau inférieur). Ces tâches de redirection aiguillent les ACK vers les gammes correspondant aux jetons présent dans les messages (identifiées par le numéro de Gamme Active).

Finalement, les boîtes aux lettres correspondant aux comptes rendus (ACK) restent indépendantes vis à vis des gammes opératoires étendues et permettent l'ajout ou le retrait à volonté de ces dernières en fonction des évolutions de la production.

Ainsi, vu de l'extérieur, le suivi des gammes opératoires étendues se réduit à un ensemble d'objets rendant des services (REQ_GOE_* et ACK_GOE_*) et faisant appel aux services d'objets ressources et d'objets décisionnels (REQ_1, ACK_1, ..., REQ_n, ACK_n).

En plus de sa fonction première, la tâche administrant la gestion dynamique du type tâche pourrait faire une administration étendue avec des suivis statistiques et différents contrôles dans l'optique d'une supervision efficace.

Les **sous gammes opératoires** étendues sont traitées de la même manière que les gammes opératoires étendues. En fait nous parlèrons plutôt de **gammes de niveau supérieur** ou de **gamme de niveau inférieur**. Le lancement d'une gamme de niveau inférieur se fera aussi par l'intermédiaire d'une requête de service.

Les **macro gammes opératoires étendues** sont représentées par des objets implantés sous forme de paquetages génériques intégrant la description du fragment de gamme ainsi que les tâches de redirection des ACK.

Pour utiliser une macro gamme il faut, en premier lieu, instancier le package générique correspondant avec le jeton représentant la pièce et l'identificateur du type de gamme où celle-ci est intégrée. L'utilisateur dispose alors de trois services :

- MACRO_GOE_*.INIT pour initialiser la macro gamme,
- MACRO_GOE_*.LANCER pour lancer la macro gamme,
- MACRO_GOE_*.TUER pour détruire les structures de contrôle de la macro gamme.

Une macro gamme opératoire étendue pourra être utilisée par n'importe quelle gamme opératoire étendue. De fait, pour accélérer le développement, pour réduire les erreurs et pour augmenter l'efficacité, il est intéressant de dégager, dès la conception, tous les fragments de gammes similaires pouvant devenir des macro gammes. Nous pouvons même mettre en place une bibliothèque de macro gammes réutilisables à volonté.

Finalement, nous obtenons un ensemble d'objets gammes, indépendants les uns des autres, dont les liens avec les objets extérieurs s'effectuent par l'intermédiaire de boîtes aux lettres (Figure III.15).

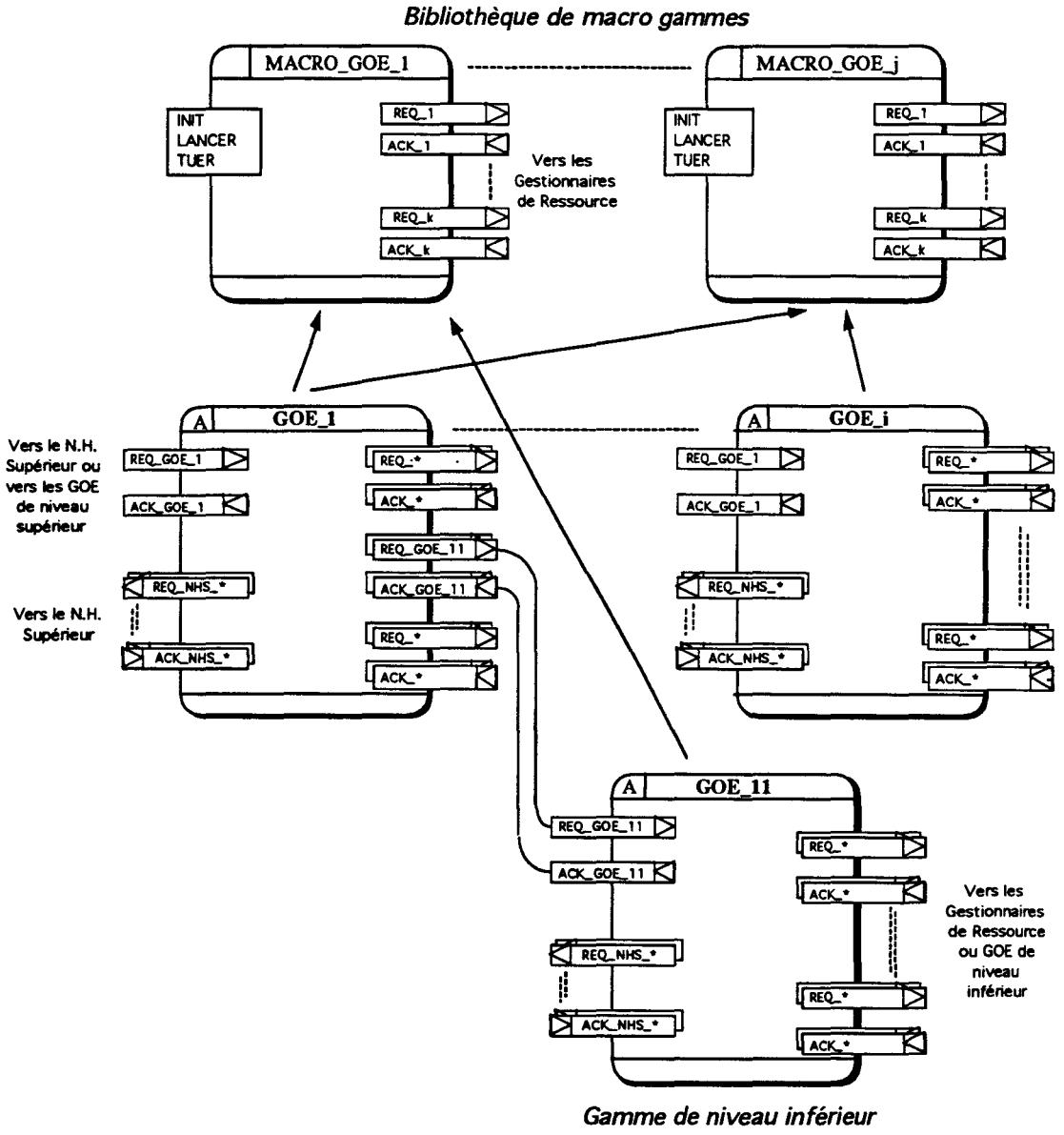


Figure III.15 : Ensemble des objets gammes

Dans tous les cas, que ce soit pour les gammes opératoires étendues, les sous gammes opératoires étendues ou les macro gammes opératoires étendues, la description ADA du séquencement de la gamme s'effectue à l'aide d'une méthode systématique et rigoureuse que nous allons détailler.

ii. Traduction des modèles des GOE

Dans la description du séquencement des gammes, nous retrouvons la traduction du modèle réseaux de Petri en code ADA et ce pour chacune des structures élémentaires dégagées dans le chapitre II : la séquence, l'alternative et le retour arrière.

Les jetons structurés qui circulent dans les réseaux de Petri décrivant les gammes opératoires étendues et qui correspondent aux pièces manufacturées sont représentés par un type de structure de données pouvant contenir plus ou moins d'informations suivant la complexité du problème à traiter. La figure III.16 décrit un exemple de structures de données minimales pouvant être mise en place. Ces structures de données sont implantées par des types *articles* en ADA.

TYPE_JETON	
TYPE	TYPE_IDENTIF_GOE
PIECE	TYPE_PIECE
GA	TYPE_GA
PARAM	TYPE_PARAM

TYPE_PIECE	
CATEGORIE	TYPE_CATEGORIE_PIECE
DU_LOT	TYPE_LOT
PRIORITE	TYPE_PRIORITE

TYPE_LOT	
CATEGORIE	TYPE_CATEGORIE_LOT
PRIORITE	TYPE_PRIORITE

Figure III.16 : Exemple de structures de données minimales

En plus des données caractérisant chaque pièce à manufacturer (TYPE_PIECE), le type TYPE_JETON intègre un champ contenant l'identificateur du type de gamme (TYPE_IDENTIF_GOE) ainsi qu'un champ correspondant au numéro de gamme active qui sera associée à la pièce (TYPE_GA). Le champ PARAM permet de passer divers paramètres lors de requêtes comme par exemple le numéro de programme d'usinage lors d'une demande de service envers une machine outils, le poste d'arrivée désiré lors d'une demande de service envers un convoyeur,...

Les données supplémentaires pouvant êtres ajoutées sont celles concernant l'ordonnancement [Tawegoum 1993], les identifications pour les assemblages [Cruette 1991], les numéros de série,...

Les requêtes se font par écriture du jeton dans les boîtes aux lettres correspondantes (Req_*_MBX.ECRIRE(JETON)) tandis que les accusés de réception se font par rendez-vous faiblement synchrones (accept ACK_*(J : TYPE_JETON) do JETON:=J; end;); l'appel des boîtes aux lettres correspondantes est externe à la description de la GOE car il est géré par les tâches de redirection des ACK.

La structure la plus simple qui est la séquence peut être assimilée à une succession de requêtes et d'accusés de réception. La figure III.17 montre l'aspect systématique de la traduction du réseaux de Petri en code ADA.

Réseau de Petri	Code ADA

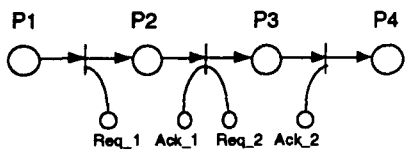
	**Req_1**_MBX.ECRIRE(JETON);
	accept **Ack_1**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_2_MBX.ECRIRE(JETON);
	accept **Ack_2**(J : TYPE_JETON) *do* JETON:=J; end;

Figure III.17 : Traduction de la séquence

L'alternative va s'effectuer en ADA par l'intermédiaire de la primitive d'attente sélective *select* qui supporte plusieurs appels d'entrées de façon non déterministe [Gehani 1988] [Booch 1988]. La figure III.18 montre un exemple de traduction pour une alternative à deux branches.

Réseau de Petri	Code ADA

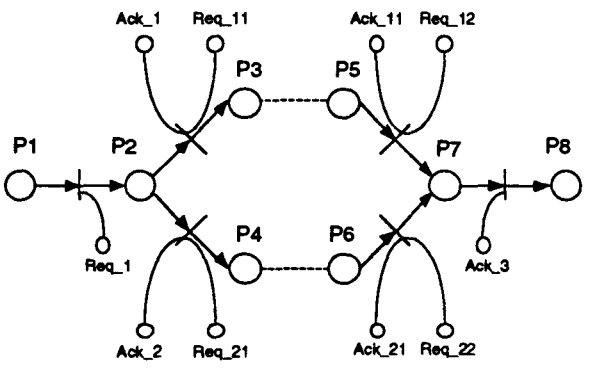
	**Req_1**_MBX.ECRIRE(Jeton);
	select
	accept **Ack_1**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_11_MBX.ECRIRE(JETON);
	accept **Ack_11**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_12_MBX.ECRIRE(JETON);
	or
	accept **Ack_2**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_21_MBX.ECRIRE(JETON)
	accept **Ack_21**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_22_MBX.ECRIRE(JETON);
	end select;
	accept **Ack_3**(J : TYPE_JETON) *do* JETON:=J; end;

Figure III.18 : Traduction de l'alternative

Cette traduction est admissible car, dans notre cas, il ne subsiste pas d'indéterminisme, celui-ci est résolu par l'envoie d'une requête vers le niveau décisionnel (ici Req_1).

Une alternative comportant plus de deux branches ne pose pas de problème particulier, il suffit d'ajouter autant de blocs de sélection (délimités par la primitive *or*) qu'il y a de branches supplémentaires.

Le retour arrière est équivalent à une boucle dont la condition de sortie dépend d'une alternative. La figure III.19 décrit la traduction ADA de ce type de structure.

Réseau de Petri	Code ADA

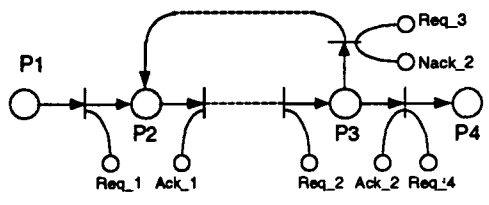
	**Req_1**_MBX.ECRIRE(Jeton);
	loop
	accept **Ack_1**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_2_MBX.ECRIRE(JETON);
	select
	accept **Ack_2**(J : TYPE_JETON) *do* JETON:=J; end;
	exit;
	or
	accept **Nack_2**(J : TYPE_JETON) *do* JETON:=J; end;
	Req_3_MBX.ECRIRE(JETON);
	end select;
	end loop;
	Req_4_MBX.ECRIRE(Jeton);

Figure III.19 : Traduction du retour arrière

La traduction en ADA de l'ensemble d'une gamme opératoire étendue est intégrée dans un type tâche et se fera en suivant scrupuleusement les mécanismes décrits. A terme, cette traduction sera assistée et pour une partie automatisée dans le cadre de la conception de l'atelier de génie automatique CASPAIM.

III.D.2.c. Les Ressources Simples

Parmi les autres composantes de la BLC, nous trouvons les objets racines correspondant aux gestionnaires de ressources simples amputés des graphes de commande complets. Cette amputation provient du fait que les GCC sont implantés sur les directeurs de commande des ressources et sont codés dans un langage métier spécialisé. Chacun de ces objets racines est donc composé de deux sous objets qui sont d'une part, un objet pseudo filtre comportemental et d'autre part, un objet allocateur. Ces objets sont implantés sous forme de paquetages génériques afin de définir des familles de ressources simples pouvant, par la suite, être facilement réutilisées.

i. Les filtres comportementaux

Comme nous avons pu le voir lors de la phase de conception, pour les ressources simples et si l'on considère uniquement l'aspect commande, les filtres comportementaux n'ont pas de raison d'être.

Nous avons pourtant introduit la notion de **pseudo filtre comportemental** pour ce type de ressource pour :

- harmoniser le format des requêtes dans les GOE pour qu'il soit indépendant du type de ressource,
- homogénéiser l'implantation,
- faciliter l'implantation répartie du logiciel de contrôle/commande.

Sur le site d'exploitation, les calculateurs gérant les graphes de commande complets peuvent être séparés du ou des calculateurs où est implanté la BLC par un ou plusieurs réseaux industriels. De fait, dans un souci d'efficacité, il est nécessaire de minimiser la quantité d'informations transitant par ces réseaux. Pour cela, les pseudo filtres comportementaux vont avoir pour fonction la rétention d'une partie des informations envoyées par les gammes opératoires étendues (REQ_*) vers les graphes de commande complets (IBG_REQ_*) (Figure II.20).

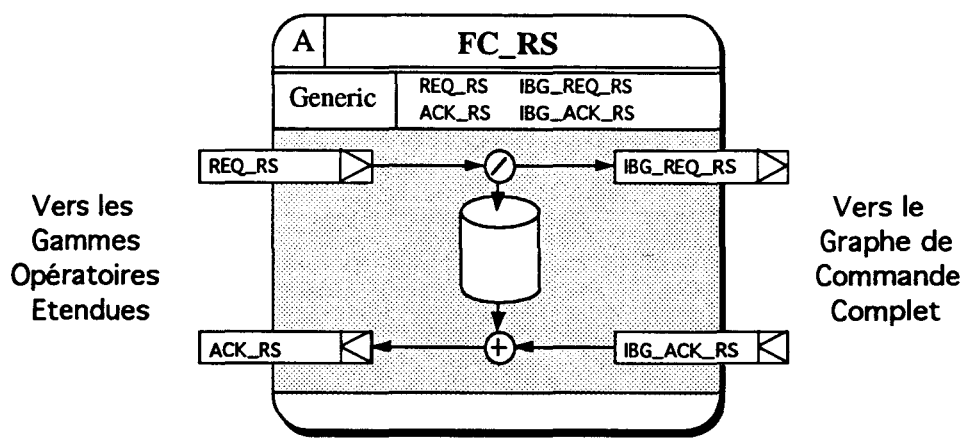


Figure III.20 : Filtre comportemental générique pour les ressources simples

Les informations effectivement transmises sont celles qui se révèlent être indispensables aux graphes de commande complets. Ce sont les paramètres de fonctionnement tels les numéros de programme d'usinage pour les MOCN, les numéros de transfert pour les robots,... Les informations non transmises sont stockées dans une structure de données. Elles sont ensuite ajoutées aux données de l'accusé de réception qui sera retourné vers les gammes opératoires étendues.

Les pseudo filtres comportementaux peuvent aussi servir à adapter les requêtes aux spécificités des ressources. Par exemple, pour un robot avec système de changement d'effecteur, la correspondance entre la pièce à saisir et l'effecteur à sélectionner est à charge du

pseudo filtre comportemental.

En raison de l'aspect générique peu complexe de ces filtres comportementaux, nous pouvons les rassembler en une même classe. Nous avons donc implanté ceux-ci sous forme d'une tâche incluse dans un paquetage générique.

ii. Les allocateurs

Comme nous avons pu le voir dans le chapitre II, les allocateurs ont pour fonction la résolution d'indéterminismes résultant d'accès simultanés (venant de diverses GOE) à une ressource partagée. Or les allocateurs de ressources simples ont tous une structure identique, seule diffère la fonction de choix. Nous avons donc créé un objet allocateur générique correspondant au codage ADA de l'algorithme d'allocation défini au chapitre II. Cet objet est décrit par la figure III.21.

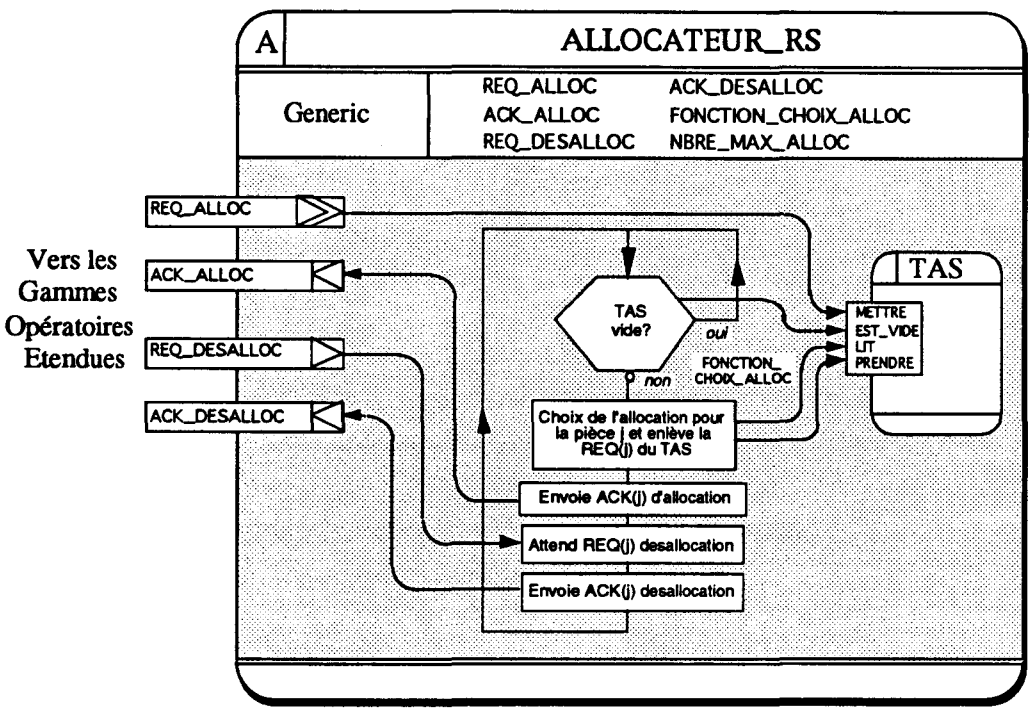


Figure III. 21 : Allocateur générique pour les ressources simples

Pour obtenir un objet allocateur dédié à une ressource simple, il suffit d'instancier cet allocateur générique avec la fonction de choix relative à celle ci ainsi que l'ensemble des boites aux lettres permettant la communication.

Nous remarquons que la boîte aux lettres correspondant à la requête d'allocation (REQ_ALLOC) est de type asynchrone. En effet, plusieurs de ces requêtes peuvent être

envoyées par l'ensemble des gammes opératoires étendues demandant accès à la ressource simultanément partagée. Il ne faut pas que les messages (jetons) restent bloqués dans une fifo d'entrée (synchronisme) mais il faut continuellement les recueillir afin de les déposer dans le TAS. Or cela n'est possible que si les requêtes des gammes sont consommées, d'où l'asynchronisme de la boîte aux lettres REQ_ALLOC.

III.D.2.d. Les Ressources Complexes

Comme pour les ressources simples, à chaque ressource complexe correspond un objet racine dans la BLC. Ces objets sont composés eux aussi d'un filtre comportemental et d'un allocateur.

i. Les filtres comportementaux

Dans le cas des ressources complexes, le filtre comportemental n'est pas passif, il intègre soit un aspect décisionnel (résolveur d'Indéterminismes Purement Locaux), soit un aspect informationnel (stockeurs,...), soit un aspect du type synchronisation (assemblages,...).

Dans la plupart des cas, le filtre comportemental sera en communication avec l'allocateur de ressource car ce dernier peu avoir besoin de connaître précisément l'état de la ressource complexe afin de prendre une décision d'allocation. La figure III.22 décrit synthétiquement la structure de ces objets filtres comportementaux.

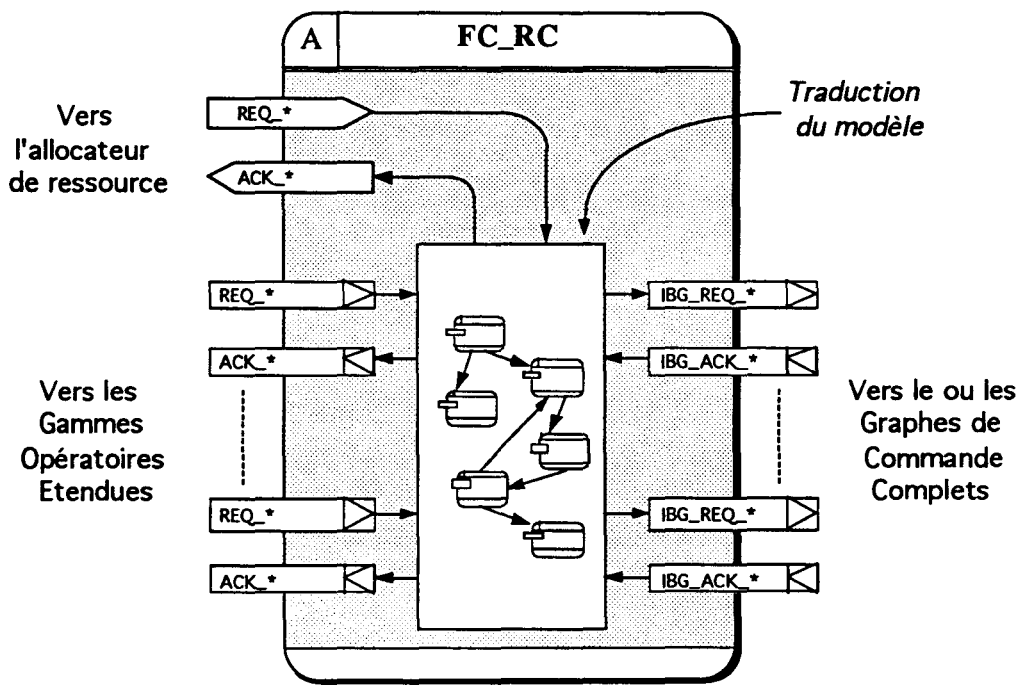


Figure III.22 : Implantation type d'un filtre comportemental de ressource complexe

L'implantation effective de ces filtres comportementaux est déduite de leurs modèles définis en conception; on y retrouve un ensemble d'objets, tels que des objets fifo, des objets sémaphores, des objets décisionnels,... [Lignelet 1990].

Par exemple, si l'on reprend le modèle du filtre comportemental du convoyeur analysé dans le chapitre II (Figures II.45 et II.49), nous implanterions celui-ci sous forme d'une tâche incluant :

- des objets fifo pour chaque fifo de stockage inter postes,
- des objets tâches correspondant aux Modules de Gestion de Poste (MGP Pi),
- des objets sémaphores pour les ressources critiques que sont les jonctions et les circuits bloquants.

ii. Les allocateurs

Les allocateurs des ressources complexes sont fortement liés au fonctionnement interne de celles-ci. Pour cela, il est impossible de définir un allocateur générique.

Par exemple, pour un convoyeur, nous aurons un allocateur simple par type de palette circulant sur celui-ci. Pour une machine d'assemblage, nous aurons un allocateur contrôlant l'ensemble des zones d'assemblage ainsi que la compatibilité des pièces à assembler (N° de série, N° de lot,...).

Ces allocateurs sont implantés sous forme de tâches incluses dans un paquetage communicant d'une part avec les gammes opératoires étendues et d'autre part avec les filtres comportementaux.

III.D.2.e. Intégration

L'objectif étant d'obtenir une Base Logicielle Commune exploitable, l'intégration de tous les paquetages représentant les objets racines (Objets gammes, objets ressources, objets décisionnels), ne peut se faire que grâce à leurs association avec un ensemble de paquetages annexes.

Cet ensemble est composé :

✔ du paquetage PACK_TYPES_GLOBAUX incluant les définitions des structures de données utilisées par tous les objets (TYPE_PIECE, TYPE_JETON,...) mais aussi les conditions initiales et limites (Nbre de pièces maximum par gamme,...),

✔ du paquetage PACK_IBG contenant les boites aux lettres REQ_IBG_* et ACK_IBG_* permettant la connexion entre la BLC et les graphes de commande complets,

✔ du paquetage PACK_BAL contenant la définition des boites aux lettres inter objets racines,

✔ des paquetages d'utilité générale tels le paquetage de gestion de FIFO générique, le paquetage de gestion de TAS générique,...

Finalement, si nous faisons abstraction de ces derniers paquetages, nous obtenons l'architecture basée objet décrit par la figure III.23.

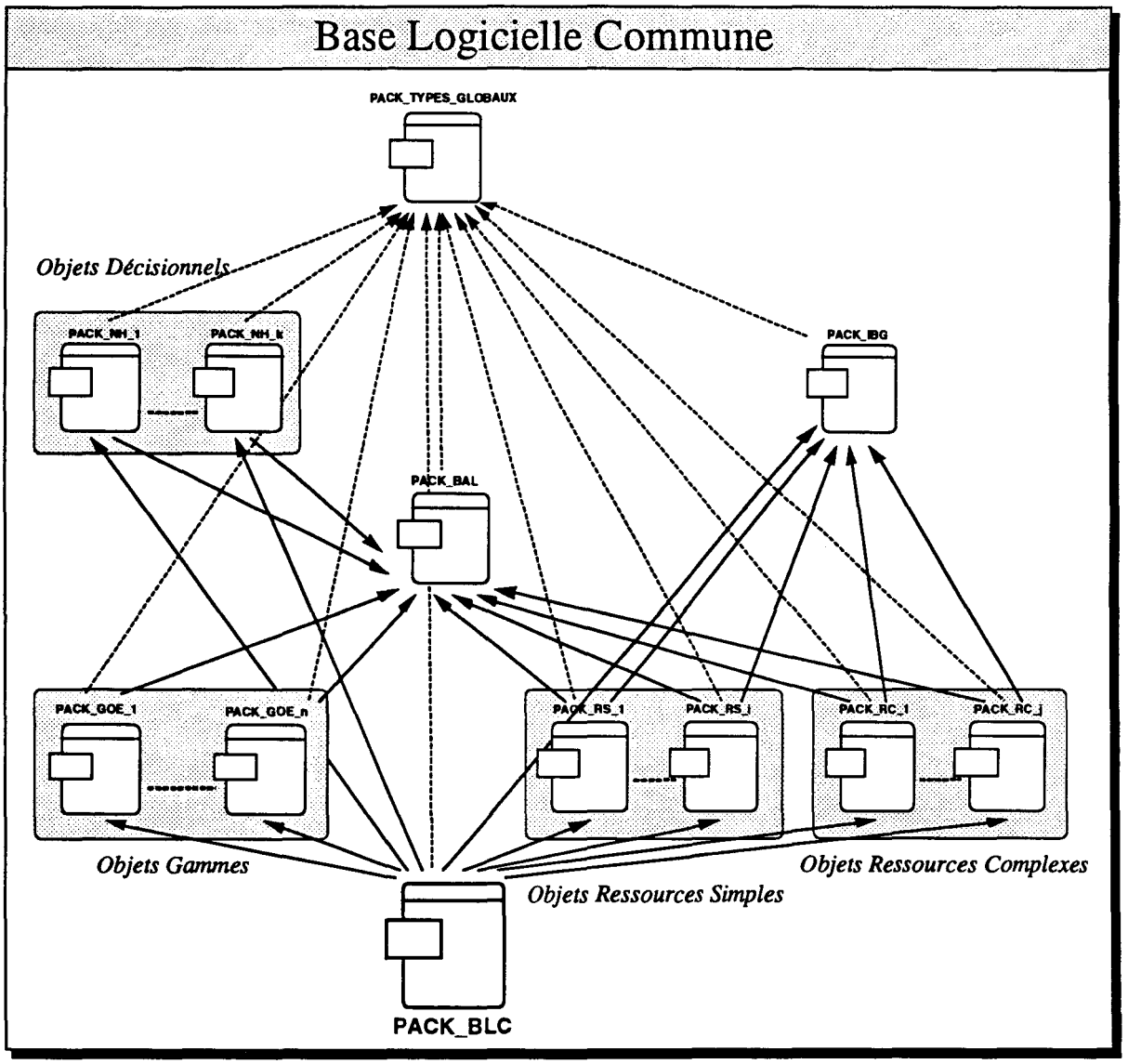


Figure III.23 : Intégration pour obtenir la BLC

La base logicielle commune étant élaborée, nous devons lui adjoindre des extensions pour permettre, d'une part, la validation par simulation et d'autre part, l'exploitation sur le site de production.

III.D.3. Extensions pour la validation par simulation:

III.D.3.a. Introduction

Pour valider le comportement dynamique de la commande implantable, il est nécessaire de passer par une phase de validation par simulation que nous qualifierons d'orientée programme car elle se base sur du code ADA directement exécutable. Il est à noter qu'un simulateur orienté structure de données simulant directement les réseaux de Petri objets a été développé au LAIL [Ausfelder et al 1992]. Ce dernier est surtout utilisé lors des phases de conception.

III.D.3.b. Mécanisme de simulation

Pour obtenir le logiciel de simulation, il faut ajouter au code ADA de la BLC un module de simulation développé lui aussi ADA. Ce dernier se compose des graphes de commandes complets traduits en ADA, d'un émulateur de procédé et d'un gestionnaire de simulation.

Les deux parties communiquent par l'intermédiaire de l'interface normalisée que constituent les boîtes aux lettres IBG_REQ_* et IBG_ACK_*. La figure III.24 résume cet interfaçage ainsi que l'architecture interne du module de simulation. Une compilation puis une édition de liens de cet ensemble nous donne alors un logiciel de simulation directement exécutable.

Le mécanisme de simulation est de type événementiel [Cavaille et al 1992] [Castelain 1987], l'évolution du système est régie par l'occurrence d'événements changeant l'état du procédé. Les événements créés sont classés dans un échéancier (Objet ECHEANCIER) par ordre temporel d'occurrence. La date d'occurrence est obtenue en additionnant à la date courante la durée d'activité de l'événement. La boucle de simulation (Objet GEST_EVOL) consiste alors à prendre, dans l'échéancier, l'événement ayant la date d'occurrence minimale, à le traiter et à recaler l'horloge (Objet HORLOGE) sur cette date. Traiter les événements consiste à faire évoluer les graphes de commande complets par une simulation de retour d'information et à faire évoluer la base de donnée représentant l'état du procédé.

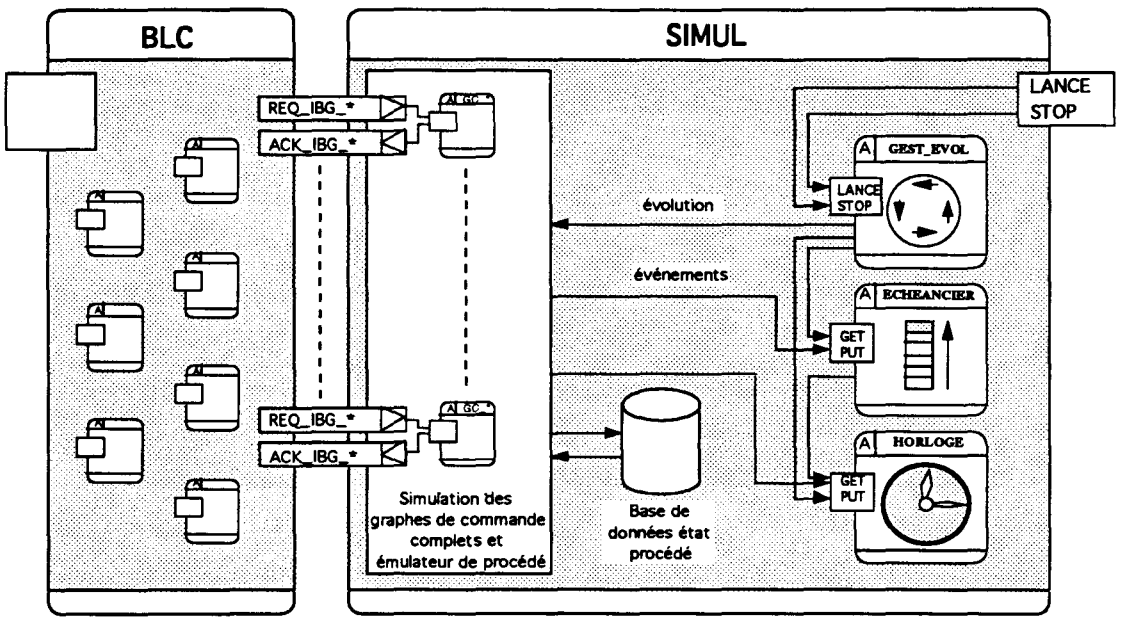


Figure III.24 : Extension pour la simulation

La durée d'activité des événements est obtenue soit par mesure statistique sur le système réel si celui-ci est opérationnel, soit à partir des données constructeur, soit à partir d'une simulation en CAO robotique (pour les robots), en CAO d'usinage (pour les MOCN),...

Le simulateur peut effectuer une sortie texte mais il est préférable que celui ci anime un synoptique 2D ou 3D représentant l'unité de production. Ce synoptique peut être développé à l'aide d'un générateur graphique qui se réfère à une bibliothèque d'objets graphiques (robots, MOCN, morceaux de convoyeurs,...). Ce synoptique pourra aussi être utilisé en exploitation afin de connaître l'état instantané du procédé ou servir de base à une interface homme-machine pour la maintenance, la surveillance...

Une étape intermédiaire avant l'implantation finale consiste à effectuer une simulation répartie sur l'ensemble des organes informatiques du site de production afin de valider la répartition qui sera utilisée en exploitation.

III.D.3.c. Traduction des graphes de commande complets

Les graphes de commande complets sont implantés sous forme de tâches. Ils font appel à une série de procédures correspondant aux actions élémentaires réalisables par le procédé et qui ont été utilisées lors de la construction des graphes de commande partiels telles :

✔ FERME_PINCE, OUVRE_PINCE, TRANSFERT(origine,destination),... pour un robot de manutention,

✔ OUVRE_BUTEE_P1, FERME_BUTEE_P1, ... pour les butées d'un convoyeur,

✔ USINE(num_programme),... pour une machine outils.

Ces procédures font évoluer l'émulateur de procédé qui contrôle une base de données représentant l'état courant du procédé. Pour les ressources simples, l'émulation du procédé consiste en une simple mise à jour de la base de données. Par contre, pour les ressources complexes, l'émulation permet de gérer les interactions internes aux ressources.

Par exemple, si nous considérons le convoyeur, lorsqu'un graphe de commande simule l'ouverture d'une butée, il y a transfert d'une palette d'un poste vers un autre avec une position intermédiaire qui est le voyage. L'émulateur du convoyeur met à jour la base de données comme suit : il enlève un objet palette de l'objet fifo représentant le poste origine et ajoute un objet palette à l'objet fifo représentant la position de voyage. A l'occurrence de l'événement correspondant à la terminaison du voyage, l'émulateur enlève l'objet palette de l'objet fifo représentant la position de voyage et l'ajoute à l'objet fifo représentant le poste de destination.

III.D.4. Extensions pour la mise en production

III.D.4.a. Introduction

Après avoir validé le comportement dynamique de la commande de notre SFPM, nous pouvons passer à la phase d'implantation effective sur site industriel, celui-ci étant composé du procédé réel et de l' architecture informatique de contrôle/commande.

III.D.4.b. Méthode utilisée

Pour réaliser l'implantation sur le site industriel, il faut adjoindre à la BLC un objet gérant les communication qui lui permettra, par l'intermédiaire du ou des réseaux industriels, de commander les graphes de commande complets répartis sur les différents calculateurs dédiés aux ressources de production. Nous obtenons alors l'architecture répartie de la figure III.25.

Pour la répartition de la BLC sur une architecture informatique répartie, nous tenons compte des considérations abordées dans la partie III.B.3 . Cette répartition est grandement facilitée grâce à notre approche très modulaire. Ainsi, les boîtes aux lettres logicielles utilisées tout au long de la démarche peuvent être converties en boîtes aux lettres réseaux. A ce niveau, une plate forme d'intégration CIM apporte beaucoup de commodités et facilite les modifications futures, la maintenance,...

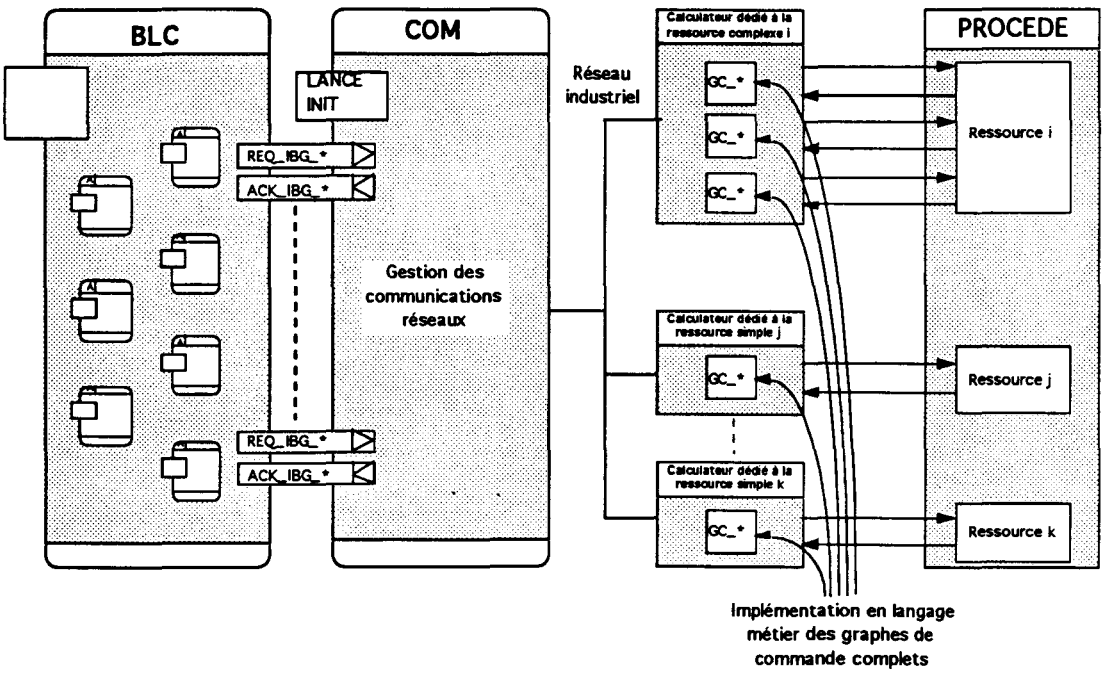


Figure III.25 : Extensions pour l'implantation sur site

III.D.4.c. Traduction des graphes de commande complets

Les graphes de commande complets sont implantés sur les calculateurs dédiés aux ressources de production. Il doivent donc être écris en langage métier (Grafcet, Langage CN,...) compréhensible par ces calculateurs. Les modèles de conception de ces GCC est généralement simple, ce qui permet de générer le squelette de leur code implantable par un simple transcodage. Cette traduction peut se faire à l'aide d'un postprocesseur dont les données initiales sont la description en Réseaux de Petri des graphes de commande complets. Ces squelette sont ensuite complété avec les programmes de transformation (MOCN) ou de transfert (Robots) développés en CAO d'usinage ou en CAO robotique.

III.E. Conclusion

Nous avons pu voir comment CASPAIM II peut être interprété selon une méthodologie de conception rigoureuse qui fournit des modèles de commande directement utilisables en implantation par l'intermédiaire d'une transposition systématique en langage ADA. De plus, la totale modularité de l'ensemble de l'approche et la banalisation des communications inter modules et inter objets permet d'aborder sans problèmes la répartition logicielle sur une architecture informatique répartie et hiérarchisée.

Un autre intérêt de l'approche proposée est qu'elle permet d'obtenir un logiciel de simulation efficace qui utilise la plupart des développements qui seront utilisés lors de l'implantation effective.

CONCLUSION GENERALE

Dans ce mémoire, nous avons abordé la modélisation de la commande d'un système flexible de production manufacturière et montré comment implanter celle-ci de façon systématique et rigoureuse à l'aide du langage ADA.

La phase de modélisation a largement fait appel au formalisme des réseaux de Petri et au formalisme objet. De fait, les modèles engendrés sont très modulaires, hiérarchisés et conservent toute la flexibilité potentielle du système de production. De plus, les différents aspects décisionnels sont bien dissociés afin de bien cibler et afin de réduire la complexité des décisions.

L'intérêt de l'approche présentée est qu'elle permet la construction de modèles complets à partir d'une bibliothèque de modèles de base représentant la capitalisation de l'expérience acquise.

Après avoir montré l'intérêt du langage ADA comme langage d'implantation principal, nous avons montré comment les différents modèles de la phase de conception de la commande pouvaient être transcrit de façon rigoureuse et systématique en langage ADA et ce dans le but de la construction de l'atelier de génie automatique CASPAIM. De cette transcription, il ressort que le codage en ADA permet de conserver intégralement la notion de processus parallèles mis en évidence en conception; chaque gamme opératoire, chaque ressource, chaque allocateur continuent d'exister sous formes d'objets logiciels.

Ainsi, les modèles produits en conception constituent un ensemble d'objets à faible couplage qui encapsulent les différentes fonctionnalités du système et il convient de retrouver ces propriétés au niveau du logiciel produit. Pour atteindre ce but, les aspects liés à la modularité, à la réutilisabilité, à la généricité, à la portabilité et aux caractéristiques temps-réel du langage ADA ont été largement utilisés. Un effort particulier a été porté sur la modularité du logiciel produit par la mise en place de mécanismes de communication inter-objets basés sur des boîtes aux lettres génériques.

L'ensemble des particularités de cette phase de codage se répercute sur la phase d'intégration qui est dès lors simplifiée. Nous avons pu voir que celle-ci permet de produire, dans un premier temps, un logiciel de simulation de la commande permettant de valider son comportement dynamique et dans un deuxième temps, le logiciel d'exploitation facilement implantable sur une architecture informatique répartie.

La phase de simulation correspond à une réelle mise en situation de la plupart des objets logiciels de la commande. En effet, hormis les graphes de commandes complets, le logiciel de

simulation utilise directement le code ADA exécutable qui sera utilisé lors de l'implantation effective (Base Logicielle Commune). Grâce à cette simulation, les propriétés déterminées en phase de conception analytiquement ou par une simulation directe des réseaux de Petri (simulation orientée structure de données) sont une ultime fois vérifiées avant l'implantation sur site. L'intérêt réside dans la validation temps réel de l'architecture retenue afin de vérifier si la réactivité du système est compatible aussi bien avec les impératifs de commande des ressources de production qu'avec les exigences en terme de productivité.

Nous avons pu voir que pour l'implantation sur le site industriel, il suffit de substituer au module de simulation les graphes de commande complets des ressources traduits en langage métier et implantés sur les organes informatiques réels tels les automates industriels, les commandes numériques... De plus, la modularité du logiciel est telle que plusieurs configurations de répartition de celui-ci peuvent êtres envisagées sans problèmes.

En perspective, l'utilisation de la nouvelle norme d'ADA (9x) permettra d'améliorer les aspects relevant de l'héritage et du temps réel critique. Les bases méthodologiques étant bien définies, le développement des divers modules logiciels de l'atelier de génie automatique ne pose, a priori, aucun problème particulier.

Une démarche d'intégration plus poussée est à envisager. En effet, il faudrait tenir compte de l'ensemble des travaux effectués sur l'ordonnancement, la surveillance et sur la gestion des modes de marches. L'architecture logicielle proposée étant ouverte, l'ajout des fonctions correspondantes se ferrait par l'adjonction de nouveaux objets aux objets existants sans nécessiter une refonte globale du logiciel.

Enfin, notons que des études sont actuellement menées au LAIL sur une méthode d'aide à la conception et à l'implantation répartie du logiciel de contrôle/commande d'un atelier flexible de production. La démarche de conception logicielle est basée sur les principes de la méthode HOOD et sur les modèles Réseaux de Petri et StateCharts appliqués à la commande et à la supervision. Cette approche est de type mixte asynchrone/synchrone. Pour l'implantation répartie, un modèle d'architecture hiérarchisé en couche de services a été développé. Ce modèle fait référence au modèle OSI pour la connexion des réseaux informatiques hétérogènes [Farah 1994].

BIBLIOGRAPHIE

[Amar et al 1990] S. Amar, E. Castelain, J.C. Gentina
"Modélisation des moyens de production par langages orientés objet en vue de la conception de la commande d'un système de production flexible".
Colloque international CIM 90, 12-14 juin 1990, Bordeaux, France.

[Amar et al 1992] S. Amar, E. Craye, J.C. Gentina
"Une méthode hiérarchique de spécification et de prototypage des systèmes de production flexible".
APII 1992 - volume 26 - numéro 5.

[Amar 1994] S. Amar
"Méthode de conception préliminaire du système de coordination des systèmes de production flexibles par prototypage orienté objet de la partie procédé".
Thèse de Doctorat d'Université de Lille I - Avril 1994.

[André et al 1993] C. André, M.A. Peraldi
"Synchronous programming : introduction and applications to industrial process control"
Congrès IEEE CompEuro'1993 - Paris Evry - 1993.

[Arcos et al 1994] P.J. Arcos, F. Riche
"Le temps réel embarqué et POSIX : l'état de l'art."
Conférence RTS'94 - Janvier 199'4 - Paris
Real-Time Systems - Teknea.

[Ausfelder et al 1992] C. Ausfelder, E. Castelain, J.C. Gentina
"An Object Oriented Simulation Tool to Validate the Dynamic Behaviour of FMS"
European Simulation Multiconference,
York,United Kingdom - Juin 1992

[Ausfelder et al 1993] C. Ausfelder, E. Castelain, J.C. Gentina
"A method for hierarchical modelling of the command of flexible manufacturing systems"
IEEE Transaction on Systems Man and Cybernetics 1993.

[Ausfelder 1994] C. Ausfelder
"Contribution à la conception d'un système de conduite pour les systèmes flexibles de production manufacturière : modélisation et validation de la commande".
Thèse de Doctorat d'Université de Lille I - Mars 1994.

[Bailly et al 1987] C. Bailly, J.F. Challine, P.Y. Gloess, H.C. Ferri, B. Marchesin
Les Langages Orientés Objets :
Concepts, Langages et Applications.
Cepadues Editions 1987.

[Barnes 1988] J. Barnes
Programmer en ADA.
InterEdition 1988.

[Barbier et al 1992] F. Barbier, P. Jaulent
Techniques orientées objet et CIM.
Eyrolles - 1992.

[Bénassy 1987] J. Bénassy
La gestion de production.
Hermès - 1987.

[Bérard 1992] C. Bérard
"La place de l'Ordonnancement dans le Pilotage d'Atelier."
Compte rendu GR Automatique du CNRS : Pôle SED - GT3
N°1 - Avril 1992

[Bieselaar et al 1988] J. Bieselaar, A. Le Gall, F. Roubellat, G. Simonnet
"Ordonnancement en temps réel d'ateliers : une méthode d'aide à la décision et à sa mise en œuvre."
Congrès Automatique 1988 - Octobre 1988, Grenoble.
AFCET

[Bois 1991] S. Bois
"Integration de la gestion des modes de marche dans le pilotage d'un système automatisé de production".
Thèse de Doctorat d'Université de Lille I. Novembre 1991.

[Bonetto 1987] R. Bonetto
Les ateliers flexibles de production.
Hermès - 1987.

[Booch 1988] G. Booch
Ingénierie du logiciel avec ADA.
De la conception à la réalisation.
InterEdition 1988.

[Booch 1992] G. Booch
Conception orientée objets et applications.
Addison-Wesley, 1992.

[Bourey 1988] J.P. Bourey
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturières"
Thèse de Doctorat de l'Université Lille I, Mars 1988.

[Bourjault et al 1987] A. Bourjault, D. Chappe, J.M. Henrioud
"Elaboration automatique des gammes d'assemblage à l'aide de Réseaux de Petri".
APII 1987, volume 21, numéro 4.

[Bourey 1988] J.P. Bourey
"Structuration de la partie procédurale du système de commande de cellules flexibles dans l'industrie manufacturière".
Thèse de Doctorat d'Université de Lille I - Mars 1988.

[Bourjault et al 1992] A. Bourjault, J.M. Henrioud
"La problématique de l'assemblage robotisé"
Revue d'automatique et de productique appliquées.
Vol 5, n°2, Hermes 1992.

[Brams 1983] G. W. Brams
Réseaux de Petri : théorie et pratique.
Tome 1 et Tome 2.
Editions Masson - 1983.

[Briand 1991] L. Briand
Systèmes temps réel en ADA.
Masson 1991.

[Calvez 1990] J.P. Calvez
Spécification et conception des systèmes : une méthodologie.
Masson - 1990.

[Carlier et al 1988] J. Carlier, P. Chrétienne
Problèmes d'ordonnancement:
modélisation/complexité/algorithmes
Masson - 1988.

[Castelain 1987] E. Castelain
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doctorat d'Université de Lille I - Février 1987.

[Cavaille et al 1992] J.B. Cavaille, J.M. Proth
SIPRODIS
Pratique de la simulation en production discontinue.
Collection Novotique - EC2 - 1992

[Coad et al 1992] P. Coad, E. Yourdon
Analyse Orientée Objets.
Masson / Prentice Hall, 1992.

[CNRS 1988] Le temps réel.
Rapport établi par le Groupe de Réflexion Temps Réel du CNRS
Technique et Science Informatiques Vol 7 n°5, 1988.
AFCET - Bordas.

[Craye 1989] E. Craye
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière".
Thèse de Doctorat de l'Université Lille I, janvier 1989.

[Cruette 1991] D. Cruette
"Méthodologie de conception des systèmes complexes à évènements discrets: application à la conception et à la validation de la commande de cellules flexibles de production dans l'industrie manufacturière".
Thèse de Doctorat d'Université de Lille I. Février 1991.

[Dassault 1992] Documentation CATIA
Dassault Systèmes
24-28, avenue du Général de Gaulle - BP 310 - 92150 Suresnes.

[Delatte et al 1993] B. Delatte, M. Heitz, J.F. Muller
HOOD reference manual 3.1
Masson / Prentice Hall - 1993

[DOD 1983] Reference Manual for the Ada Programming Language
Ansi/MIL-STD-1815A-1983

[Doumeingts 1984] G. Doumeingts
"Méthode GRAI : Méthode de conception des systèmes de productique"
Thèse d'état en Automatique- Université de Bordeaux I - 1984.

[DRED 1993] Rapport final DRED.
"Approches asynchrones/synchrones des systèmes flexibles de production."
GR Automatique-Pole systèmes à événements discrets.
Septembre 1992.

[Engineers 1988] I. Engineers
L'usine intégrée.
Hermès 1988.

[Elkhattabi 1993] S. Elkhattabi
"Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets: application aus systèmes de production flexibles."
Thèse de Doctorat d'Université de Lille I -Septembre 1993.

[Erschler et al 1988] J. Erschler, G. Fontan, C. Merce
"Agrégation et désagrégation en planification hiérarchisée."
Congrès Automatique 1988 - Octobre 1988, Grenoble.
AFCET

[Farah 1993] A. Farah
"Une méthodologie d'implantation répartie du système de commande d'un SAP"
Rapport interne LAIL URA CNRS D 1440 - 30 Mars 1993.

[Farah et al 1994] A. Farah, J.P. Bourey, E. Craye
"Une méthode hiérarchique orienté objets pour la conception du logiciel de contrôle/commande d'un SFPM.
Une approche synchrone/asynchrone."
APII - à paraître.

[Gehani 1988] N. Gehani
ADA Introduction avancée.
Eyrolles 1988.

[Gloannec 1990] J.M. Gloannec
"Conception d'un atelier d'assemblage flexible à l'aide de la méthodologie CASPAIM."
Rapport de DEA de productique de l'Université de Lille I - 1990.

[Gondran et al 1979] M. Gondran, M. Minoux
Graphes et algorithmes.
Editions Eyrolles 1979.

[Hammadi 1991] S. Hammadi
"Une méthode d'ordonnancement minimisant les temps d'attente et de transit dans les systèmes de production flexible de type Job-Shop".
Thèse de Doctorat d'Université de Lille I. Décembre 1991.

[Harel 1987] D. Harel
"STATECHARTS : a visual formalism for complex systems".
Science of computer programming - n°8 - 1987.

[Hillion et al 1988] H.P. Hillion, J.M. Proth
"Système de gestion de production hiérarchisée : conception et utilisation."
Congrès Automatique 1988 - Octobre 1988, Grenoble.
AFCET

[Hoare 1978] C. A. R. Hoare
Communicating sequential processes.
Revue ACM - n°8 Vol 21 - Août 1978.

[HOOD 1991] Tutorial HOOD
Cisi Ingenierie.
Conférence LIANA, Septembre 1991.

[Huvenoit et al 1992] B. Huvenoit, E. Craye, J.C. Gentina
"Elaboration de la commande de cellules flexibles dans l'industrie manufacturière".
Conférence Canadienne sur l'Automatisation Industrielle.
1-3 juin 1992, Montréal, Canada.

[Huvenoit et al 1993] B. Huvenoit, E. Craye, J.P. Bourey
"Implantation oriented methodology in design control of flexible manufacturing systems".
Conférence COMP EURO 93, 24-27 may 1993, Paris-Evry, France.

[Huvenoit et al 1995] B. Huvenoit, E. Craye, J.P. Bourey
"Design and Implantation methodology in based on Petri nets formalism of flexible manufacturing systems control".
Revue Production Planing & Control, à paraître dans le vol 6 n°1 ou n°2., Taylor & Francis.

[iRMX 1980] iRMX86 Workshop.
Student Study Guide.
Rev. 3.0, December 1980, Intel.

[Jacobson et al 1993] I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard
Le génie logiciel orienté objet.
Addison-Wesley - 1993.

[JR 1988]. *"L'automatique et l'informatique font un enfant : Le Pyramid Intégrator."*
Revue LeJournal de la Robotique - n°47 - novembre 1988

[Kapusta 1988] M. Kapusta
"Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellule de production flexibles dans l'industrie manufacturière."
Thèse de Doctorat d'Université de Lille I. Décembre 1988.

[Kermad et al 1993] L. Kermad, C. Ausfelder, J.P. Bourey, E. Castelain
"Integrative approach for a functional specification of FMS control"
Revue CIMS - Vol. 6 n°4 - 1993.

[Maik et al 1994] J.P. Maik, L. Kermad, J.C. Gentina, D. Delfieu, R. Moisand, A.E.K Sahraoui, C. Ausfelder.
"Integration of operating modes in the control of flexible manufacturing systems combining synchronous and asynchronous approaches"
IEEE International Conference on Systems, Man and Cybernetics, Le Touquet - France - October1993.

[Lamy 1987] P. Lamy
Ordonnancement et gestion de production.
Hermès - 1987.

[Leguy 1989] B. Leguy
ADA : Guide d'utilisation
Eyrolles - 1989.

[Le Moigne 1990] J.L. Le Moigne
La modélisation des systèmes complexes.
Afcet systèmes - Dunod - 1990.

[Le Verrand 1985] D. Le Verrand - AFCET
Le langage ADA : Manuel d'évaluation.
Bordas - Dunod informatique 1985.

[Lignelet 1990] P. Lignelet
Structures de données avec ADA.
Conception orientée objets.
Masson 1990

[Magnin et al 1991] R. Magnin, J.P. Urso
Mémotech commande numérique : programmation.
Editions Casteilla - 1991.

[Matra 1992] Documentation EUCLID-IS
Matra Datavision
31, avenue de la Baltique - 91944 Les Ulis cedex.

[Mayet 1989] J. Mayet
"*Sur la conception d'un atelier flexible de fabrication manufacturière*"
Mémoire Ingénieur CNAM, Lille, Juin 1987.

[Meyer 1988] B. Meyer
Object - Oriented Software Construction.
Prentice Hall - 1988.

[Meyer 1991] B. Meyer
Conception et programmation par objets pour un logiciel de qualité.
InterEditions - 1991.

[Morel et al 1990] G. Morel, F. Munerato, R. Vogrig, G. Ris
"Programmation orienté produit d'un ilot flexible de production de pièces mécaniques".
Laboratoire CRAN de Nancy.
Compte rendu réunion SED du 12 janviern 1990.

[Morel et al 1988] G. Morel, M. Roesh, M.Veron
"Génie Productique, génie X"
Congrès Automatique 1988 - TEC 88 - Grenoble
Afcet.

[Morel 1992] G. Morel
"Contribution à l'automatisation et à l'ingénierie des systèmes intégrés de production".
Habilitation à diriger des recherches - Janvier 1992
CRAN deNancy

[Murata 1989] T. Murata
Petri Nets : Properties, Analysis and Applications.
Petri Nets - Proceedings of the IEEE. - april 1989.

[Padiou et al 1990] G. Padiou, A. Sayah
Techniques de synchronisation pour les applications parallèles.
Cepadues - Editions - 1990.

[Paludetto 1992] M. Paludetto
"Sur la commande de procédés industriels: une méthodologie basée objets et réseaux de Petri".
Thèse de Doctorat de l'Université Paul Sabatier - Toulouse - 1992.

[Peraldi 1993] M.A. Peraldi
"Conception et Réalisation de Systàmes Temps-Réel par une Approche Synchrone."
Thèse de Doctorat de l'Université de Nice-Sophia Antipolis.
Juillet 1993.

[Pradenc 1991] H. Pradenc
"Entrées-sorties déportées : de l'intelligence et des communications."
Revue Axes Robotique - Janvier 1991

[Pruvost 1983] J.C. Pruvost
Point en ROBOTIQUE
TEC & DOC lavoisier, 1983.

[Pruvot 1993] F.C. Pruvot
"After CAD: Automatic Design; Dream or Reality"
1993 IEEE International Conference on Computers in Design, Manufacturing, and Production: Comp Euro 93, 24-27 Mai 1993, Paris-Evry, France.

[Rakotoson 1993] M.P. Rakotoson
"Synthèse des caractéristiques et techniques de développement de la commande des systèmes dis-continus : application aux systèmes de production flexible Batchs"
Thèse de Doctorat d'Université de Lille I. Juillet 1993.

[Ranky 1990 a] P. G. Ranky
Total Quality Control and JIT Management in CIM.
CIMware Limited , 1990.

[Rancky 1990 b] P.G. Rancky.
Manufacturing Database Management and Knowledge Based Expert Systems.
CIMware Limited -1990.

[Richard Foy 1994] M. Richard Foy
"ADA 9X : un langage adapté à la programmation temps réel critique".
La Lettre ADA. - Numéro 73, mars 1994.
EC2.

[Robert et Al. 1992]. M. Robert, J.L. Noizette, J.M. Rivière, J.P. Jouannet
"Les capteurs intelligents dans les systèmes automatisés de production à intelligence distribuée."
Conférence Canadienne sur l'Automatisation Industrielle.
Montréal, 1-3 juin 1992

[Ross 1985] D.T. Ross
"Applications and extensions of SADT"
IEEE Computer - April 1985.

[Sibertin-Blanc 1985] C. Sibertin-Blanc
"High Level Petri-Nets with Data Structure"
6th European Workshop on Applications and Theory of Petri-Nets
Helsinki - Finland - 1985.

[Sielski 1993] K. L. Sielski
"La mise en œuvre du mécanisme de tâches ADA dans un environnement Unix multiprocesseur à chaînage multiple".
La Lettre ADA. - Numéro 67-68, août-septembre 1993.
EC2.

[Stephen et al 1992] M. Stephen, F. Clouther
"BASEstar : The Integration Platform for a CIM Environment"
IEEE International Workshop on Emerging Technologies and Facory Automation.
Melbourne - Australia - 1992.

[Stoeckel 1991] R. Stoeckel
X Window programmation.
Armand Colin - 1991.

[Tardieu et al 1983] H. Tardieu, A. Rochefeld, R. Coletti
"La méthode MERISE"
Tomes 1 et 2 - Les éditions d'organisation -1983.

[Tawegoum et al1992] R. Tawegoum, E. Castelain, J.C. Gentina
"Real-time piloting of flexible manufacturing systems".
Third International Workshop on Project Management and Sceduling, 1992, Como, Italy.

[Tawegoum et al 1994] R. Tawegoum, E. Castelain, J.C. Gentina
"Dynamic operation control in flexible manufacturing systems (FMS)"
Congrès Intelligent Manufacturing Systems - june1994 - (IMS'94)
IFAC - Edited by Peter Kopacek

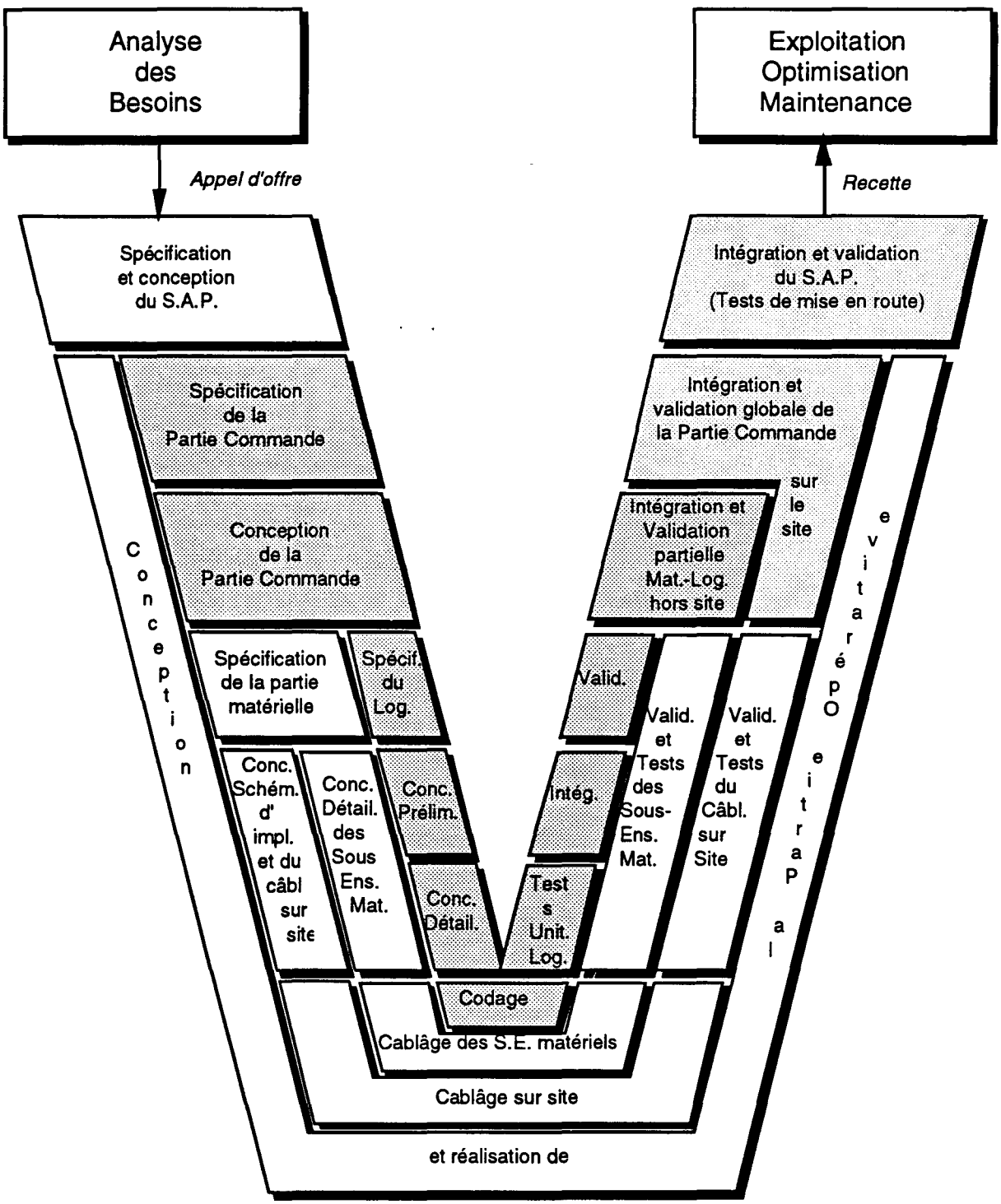
[Thomesse 1985] J.P. Thomesse
Les réseaux locaux industriels.
ETA - 1985.

[TSI 1985] Technique et Science Informatique
Spécial ADA - Volume 4 n°2 - 1985
Dunod - Afcet informatique

[Toguyeni 1992] A.K.A. Toguyeni
"Surveillance et diagnostic en ligne dans les ateliers flexibles de l'industrie manufacturière".
Thèse de Doctorat d'Université de Lille I - Novembre 1992.

[Walas et al 1993] M. Walas, A.M. Hugues
"Synthèse et classification des langages parallèles".
Génie Logiciel et Systèmes Experts N°32 Septembre 1993

ANNEXE I



Cycle en V pour la conception d'unSAP

ANNEXE II

SOMMAIRE DE L'ANNEXE II

A. Introduction

Dans cet annexe, nous allons présenter comment mener à bien l'automatisation complète d'un SFPM en se basant sur la méthodologie décrite dans ce mémoire. Pour cela, nous allons considérer la mise en œuvre d'une partie de la cellule flexible de fabrication mécanique de l'Ecole Centrale de Lille.

B. Description de la cellule prise en exemple

B.1. Architecture physique

Nous allons donc étudier une partie, que nous considérerons comme autonome, de la cellule flexible de production mécanique de l'Ecole centrale de Lille. L'architecture physique de cette ensemble est la suivante :

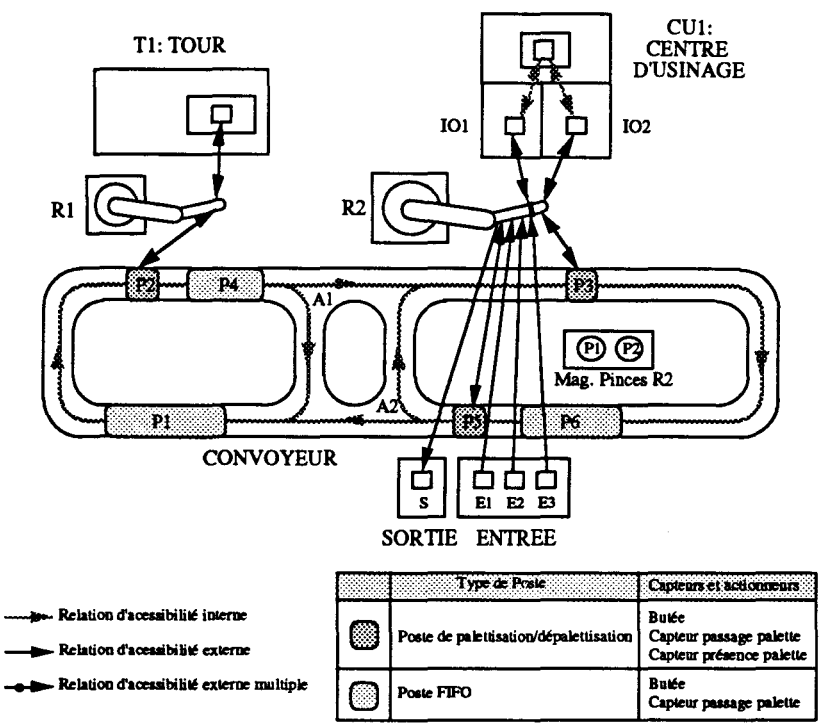


Figure AII.1 : Architecture physique de la cellule

Cette cellule est composée des éléments suivants :

✔un convoyeur à chaîne pour le transfert des pièces palettisées entre les différents éléments de la cellule. Il est composé de deux aiguillages (A1 et A2), de trois postes de palettisation/dépalettisation (P2, P3 et P5) et de trois postes de type FIFO (P1, P4 et P6).

L'ensemble des postes comportent à leur tête une butée et une série de capteurs.

✔un robot de manutention universel R1. Celui-ci est un modèle AFMA de type R3. Il possède 6 axes de liberté. Le calculateur dédié est une ROBONUM 800 et peut être programmé par langage spécialisé tel le L.M..

✔un robot de manutention universel R2. Ce robot est un modèle T3-776 de marque Cincinati. Il possède 6 axes de liberté et dispose d'un système de changement d'effecteurs. Les déplacements sont enregistrés par apprentissage.

✔un tour T1 à commande numérique intégrée. Ce tour est un modèle HES 400 de chez HErnault Somua (aujourd'hui HES Toyoda Automation). Il possède deux axes et une tourelle pouvant comporter 12 outils. Le directeur de commande est une CNC NUM 760 T spécialement développée pour ce type d'application.

✔un centre d'usinage CU1 à commande numérique. Il s'agit d'un modèle CU 60 LRM de la société Graffenstaden. Celui-ci est commandable sur quatre axes et est équipé d'un système de palettisation linéaire alternatif, composé de deux table-palettes (deux tampons d'entrée/sortie mono pièce). Un changeur automatique d'outils avec un magasin disques est intégré à la machine. Le directeur de commande est également une NUM 760 adaptée aux techniques de fraisage.

✔des fifo d'entrée/sortie des pièces.

B.2. Architecture informatique

Au niveau le plus bas de l'architecture informatique, nous retrouvons les calculateurs intégrés aux ressources de production T1, CU1, R1, R2 ainsi qu'un automate programmable industriel TSX 47/30 de marque Télémécanique dédié au convoyeur. Excepté le calculateur du robot R2 relié a l'API par une interface tout ou rien, l'ensemble de ces organes de commande sont reliés par l'intermédiaire d'un réseau local industriel UniTelway. Ce réseau de type maître/esclave utilise le mode de transmission différentiel de la norme RS485 et supporte le protocole UNI-TE de la société Télémécanique. Ce protocole permet, entre autre, l'accès aux diverses variables des équipements, le téléchargement de programmes d'usinage, le lancement de programmes, l'émission de données non formatées pouvant faire office d'interruptions,...

D'autre part, ce réseau est connecté à une station de travail faisant office de superviseur de cellule. Elle est elle-même connecté à un réseau Ethernet sur lequel est aussi connecté, entre autre, le calculateur central de l'Ecole Centrale de Lille. L'architecture globale est décrite par la figure suivante (figure AII.2) :

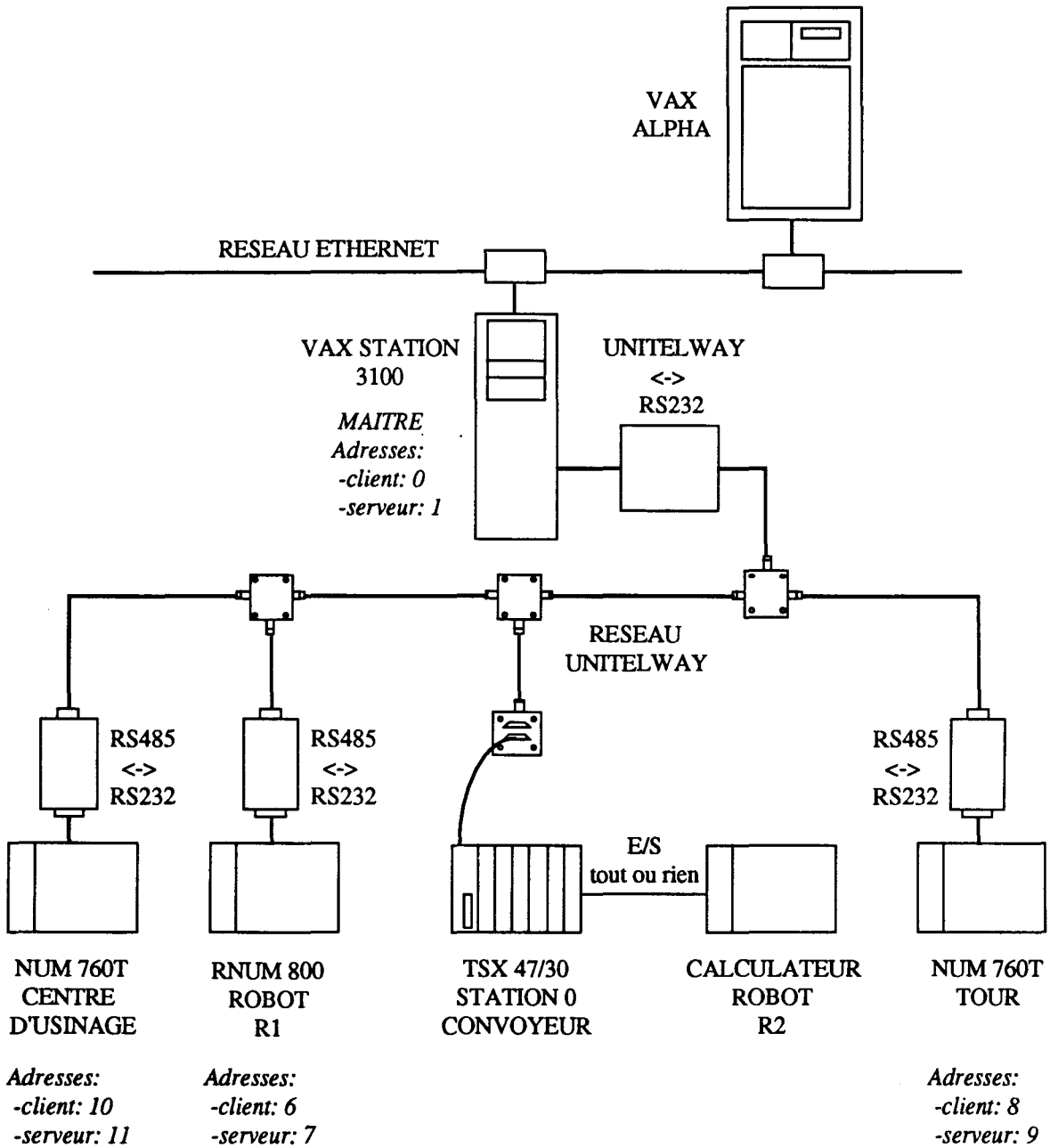


Figure AII.2 : Architecture informatique de la cellule

C. Analyse du procédé

En faisant référence aux critères énoncés dans la partie D du chapitre II, nous pouvons classifier les ressources présentes dans ce SFPM. Ainsi,

- les ressources simples sont : R1, R2, T1,
- les ressources complexes sont : CONV, CU1.

C.1. Niveau 0 de l'analyse

En premier lieu, il est nécessaire de décrire la décomposition structurelle de niveau 0. Celle-ci considère les ressources de production élémentaires, simples ou complexes, associées par l'intermédiaire de leurs relations d'accessibilités externes (Figure AII.3).

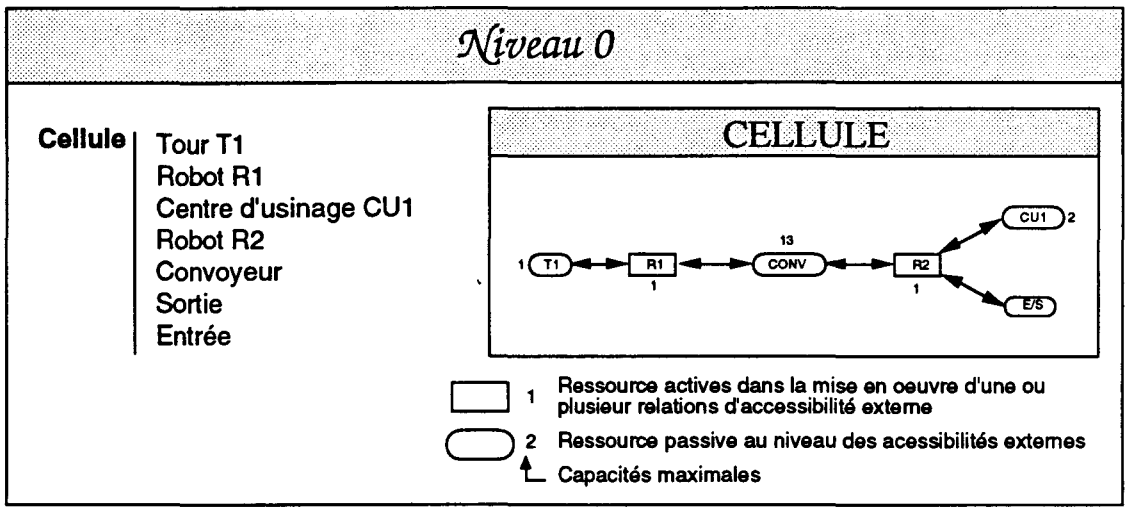


Figure AII. 3 : Décomposition structurelle de niveau 0

C.2. Analyse descendante

Pour le convoyeur, la commande de bas niveau étant inexistante, nous allons devoir concevoir celle-ci. Pour cela, une analyse descendante est nécessaire. Le premier niveau de décomposition, le niveau -1, décompose le convoyeur en éléments commandables et éléments non commandables (Figure AII.4).

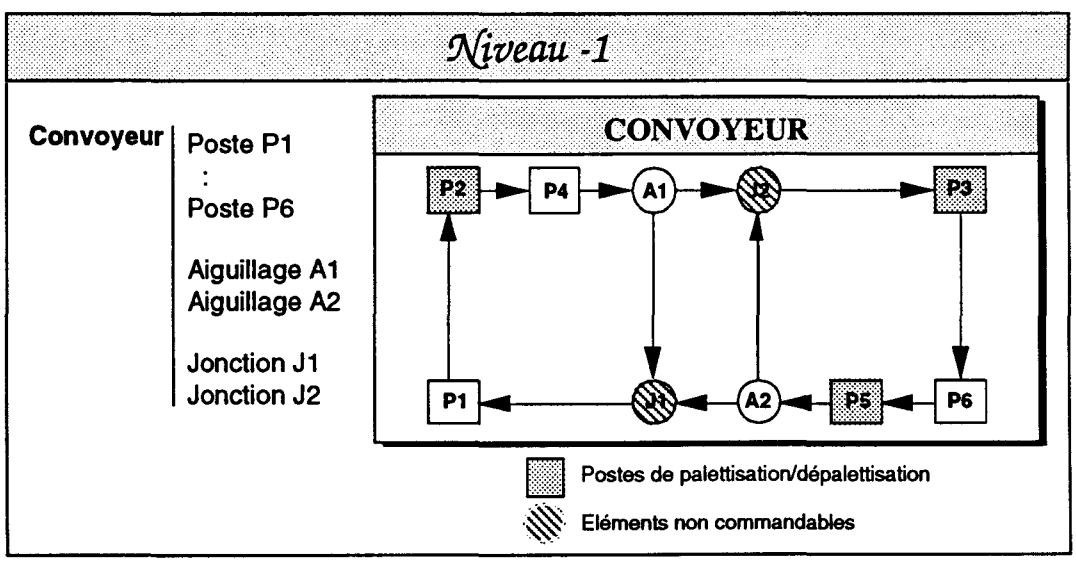


Figure AII. 4 : Décomposition de niveau -1

Pour la suite de l'analyse descendante, nous nous référerons à la partie II.D.3.b du chapitre II, celle-ci étant identique.

C.3. Analyse ascendante

En vue de hiérarchiser la commande de coordination, une analyse ascendante du procédé est nécessaire. Lors de celle-ci, nous effectuons des agrégations successives par réduction de la table des contraintes (Figure AII.5 et AII.6).

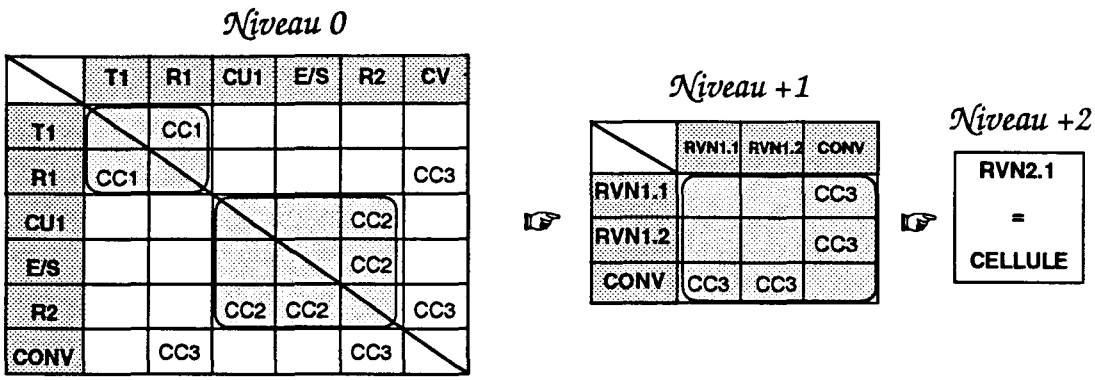
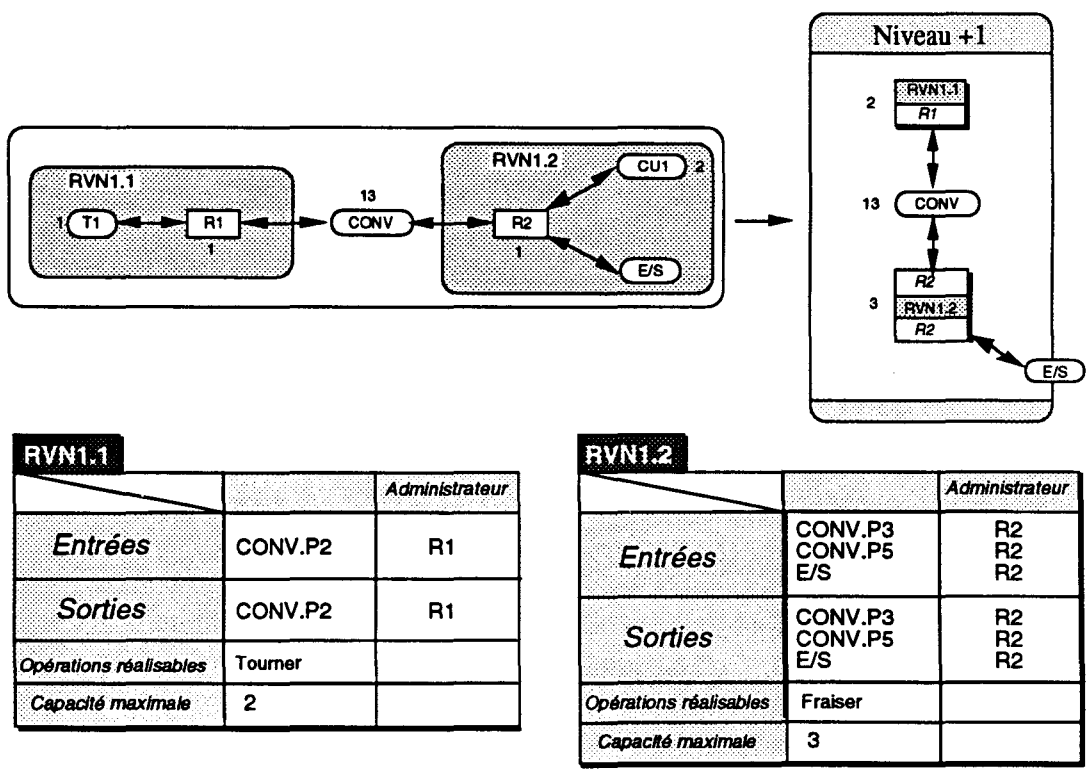


Figure AII. 5 : Réduction de la table des contraintes



RVN1.1

		Administrateur
Entrées	CONV.P2	R1
Sorties	CONV.P2	R1
Opérations réalisables	Tourner	
Capacité maximale	2	

RVN1.2

		Administrateur
Entrées	CONV.P3 CONV.P5 E/S	R2 R2 R2
Sorties	CONV.P3 CONV.P5 E/S	R2 R2 R2
Opérations réalisables	Fraiser	
Capacité maximale	3	

Figure AII. 6 : Agrégation de niveau +1

Pour RVN1.2, nous éliminerons, par la suite, la possibilité d'entrée et de sortie par le poste P5 du convoyeur, celle-ci étant sans intérêt.

C.4. Résumé de l'analyse

L'ensemble des agrégations et des décompositions structurelles va apporter une dimension hiérarchique au pilotage de la cellule (Figure AII.7).

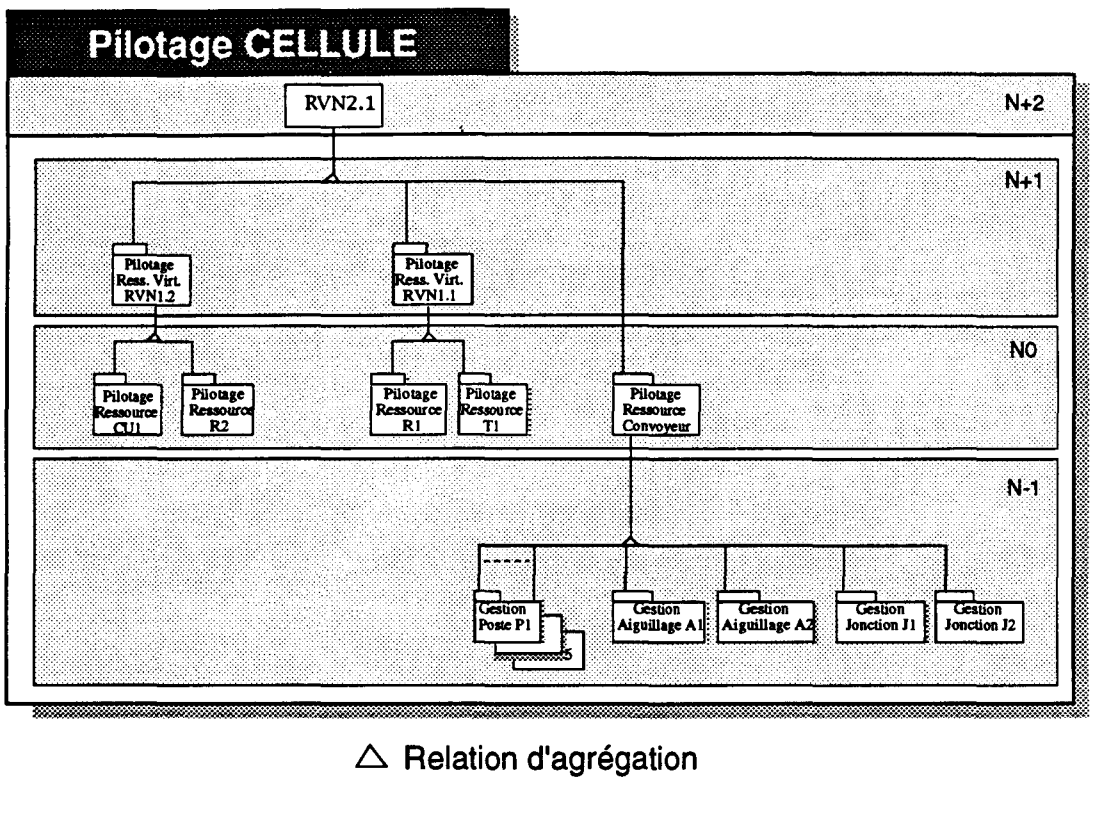


Figure AII.7 : Organisation hiérarchique du pilotage de la cellule

D. La partie logique

D.1. Ensemble des pièces à manufacturer

Nous allons considérer la mise en fabrication de trois types de pièces (Figure AII.8) :

✔ des éprouvettes de fatigue oligocycliques, référencées comme étant des pièces de type P1. Pour obtenir ces pièces, les bruts d'usinage subissent un tournage de type 1 → T1=t1,

✔ des pots de porte crayon, référencées comme pièces de type P2. Pour obtenir ces pièces, les bruts d'usinage subissent un fraisage de type 1 → P2=f1,

✔ des socles de porte crayon, référencés comme pièces de type P3. Pour obtenir ces

pièces, les bruts d'usinage subissent un tournage de type 2 puis un fraisage de type 2 → P3=t2f2,

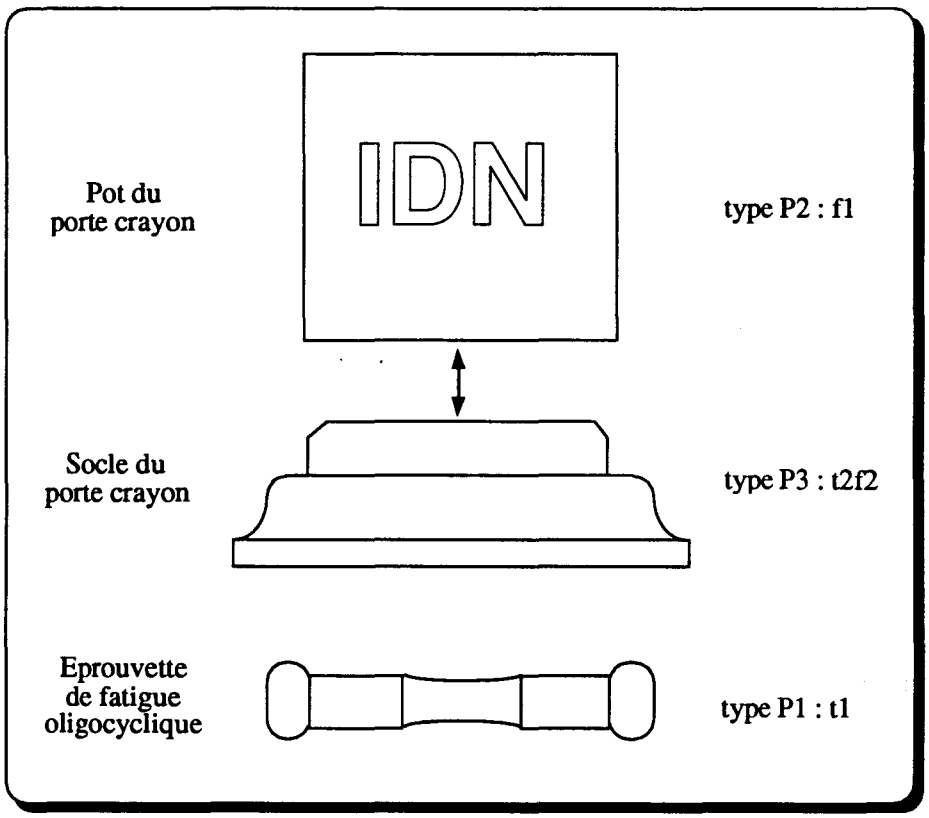


Figure AII.8 : Ensemble des pièces à produire

D.2. Les gammes logiques

Les gammes logiques correspondant aux pièces manufacturées sont les suivantes (Figure AII.9) :

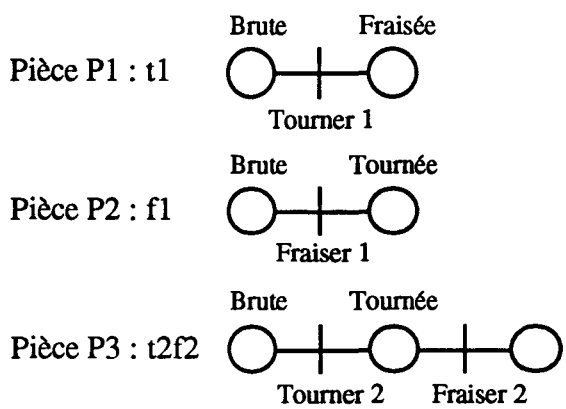


Figure AII.9 : Ensemble des gammes logiques

Les bruts d'usinage correspondant aux pièces de type P1, P2 et P3 sont respectivement disponibles aux entrées E1, E2 et E3.

D.3. Les gammes opératoires

Si nous appliquons la démarche décrite dans le chapitre II, nous pouvons dégager l'existence de deux macro gammes et de deux sous gammes génériques correspondant à l'agrégation de niveau 0 (Figure AII.10).

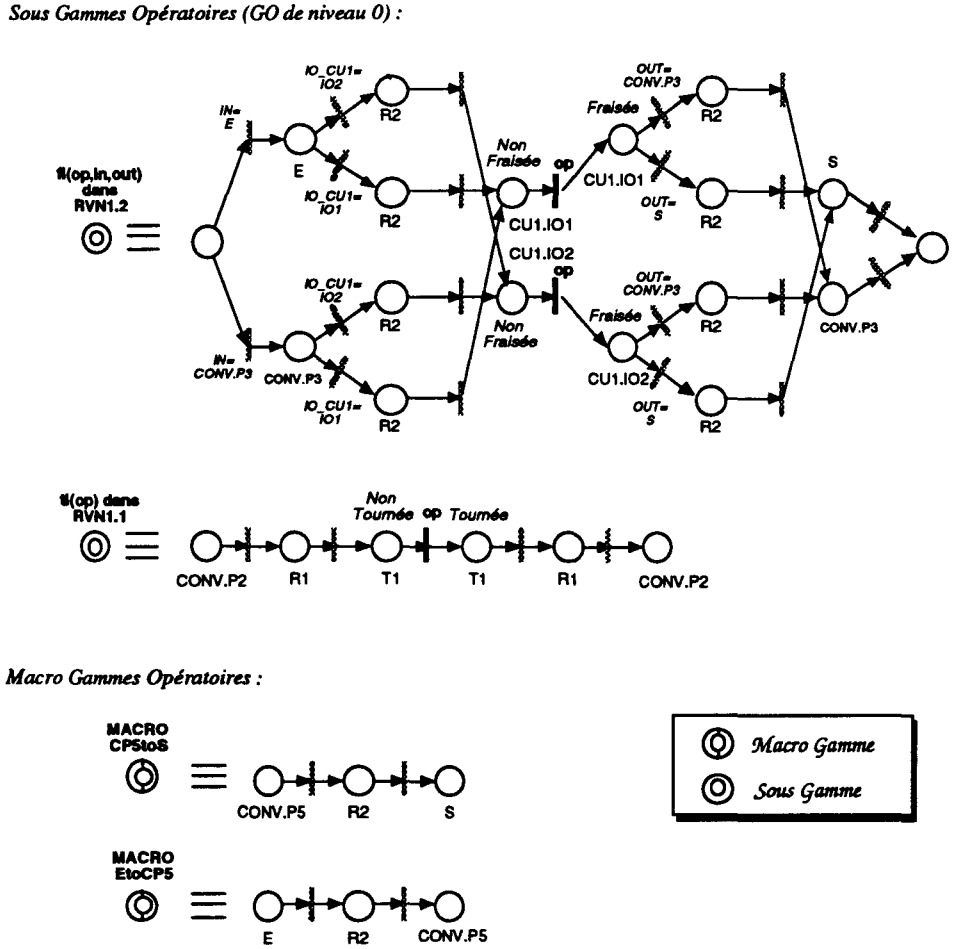


Figure AII.10 : Ensemble des sous gammes génériques et des macro gammes

A partir de ces sous gammes opératoires génériques et macro gammes opératoires, nous pouvons définir l'ensemble des gammes opératoires correspondant à la production des trois types de pièces (Figure AII.11).

A partir de ces sous gammes opératoires génériques et macro gammes opératoires, nous pouvons définir l'ensemble des gammes opératoires correspondant à la production des trois types de pièces (Figure AII.11).

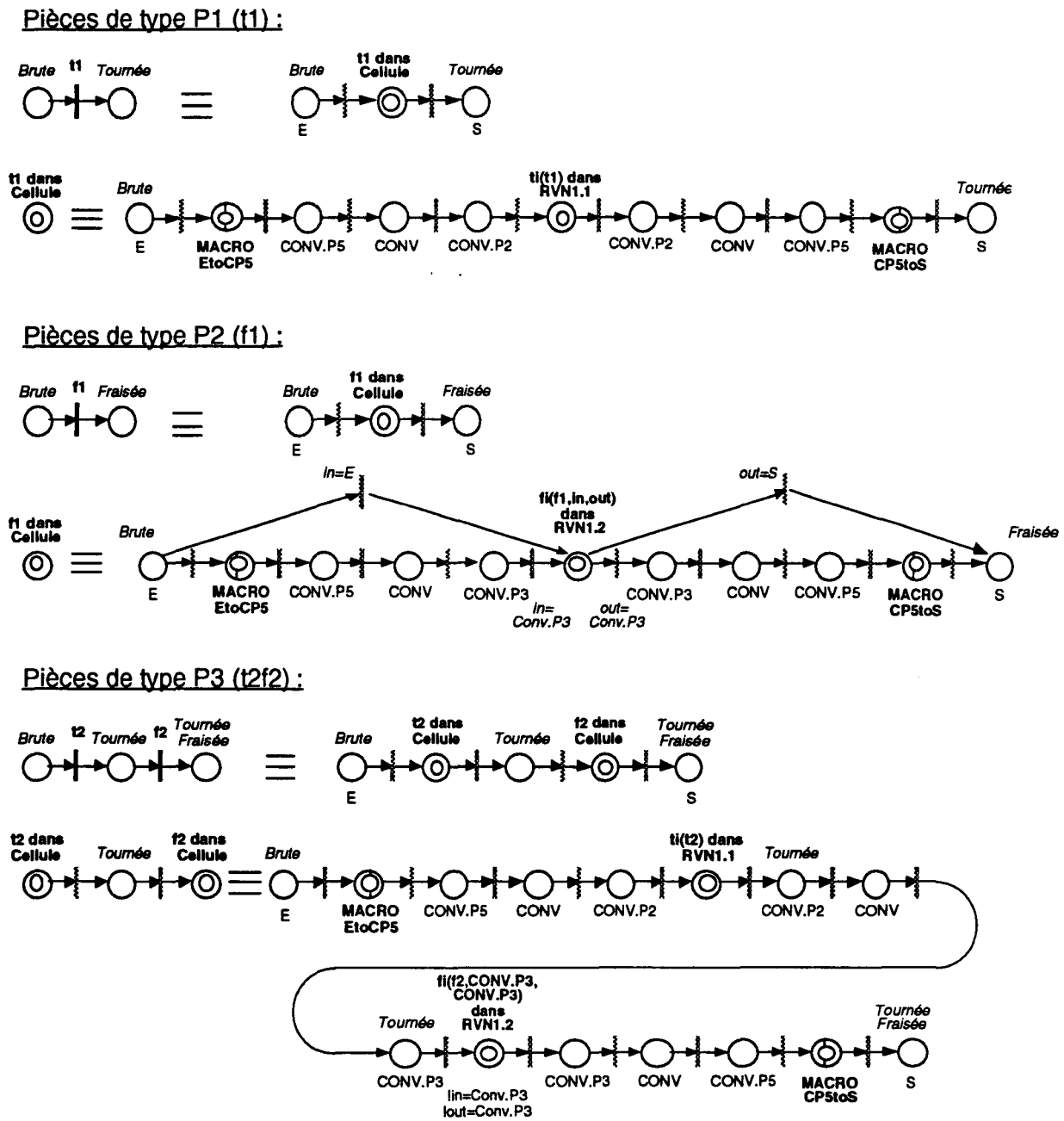


Figure AII. 11 : Ensemble des gammes opératoires

D.4. Les gammes opératoires étendues

Celles-ci sont une extension des gammes opératoires, elles intègrent tous les protocoles de communication avec les ressources, les allocateurs, les niveaux décisionnels et les autres

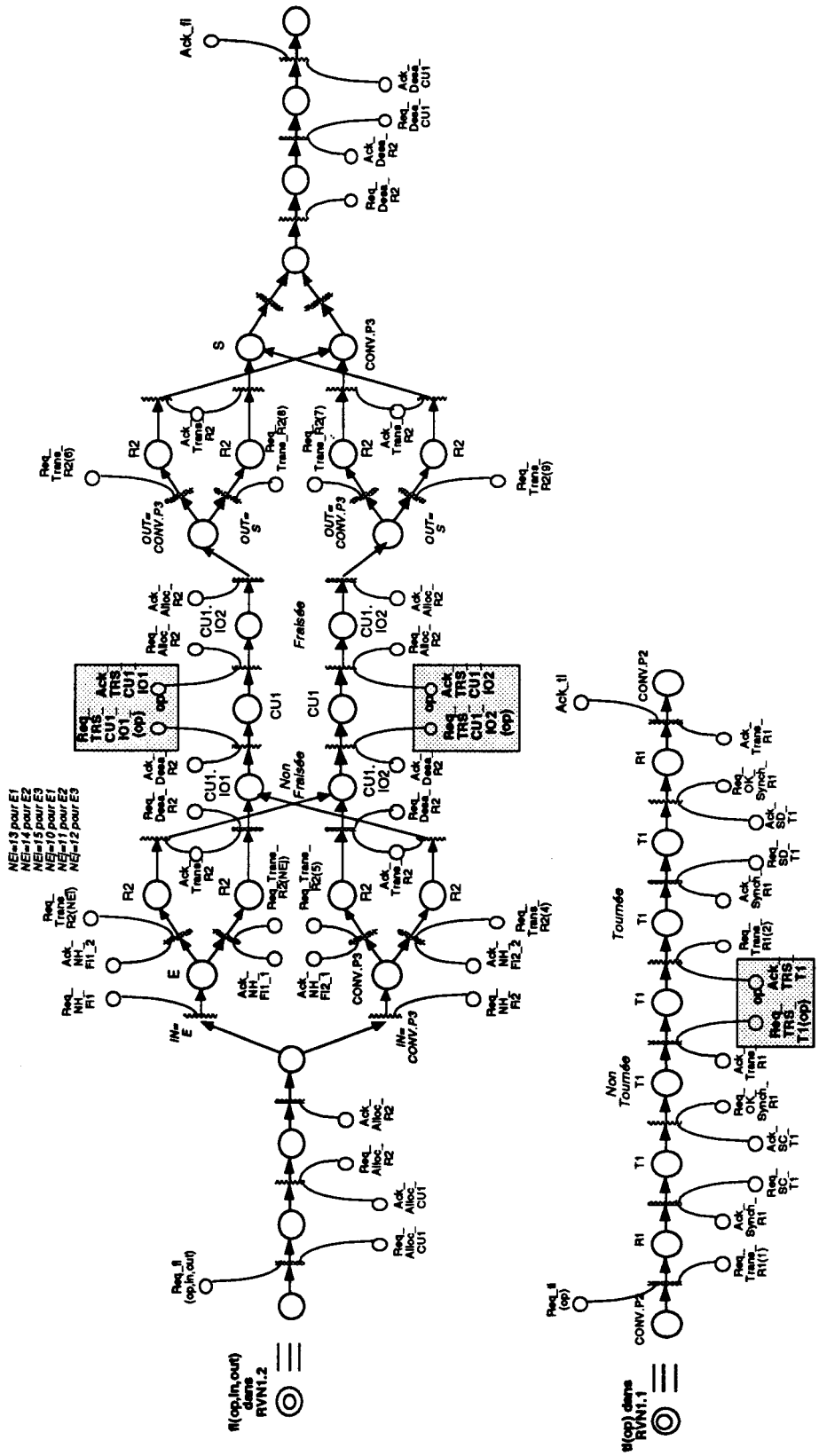


Figure AII.12 : Sous Gammes Opératoires Etendues de niveau 0

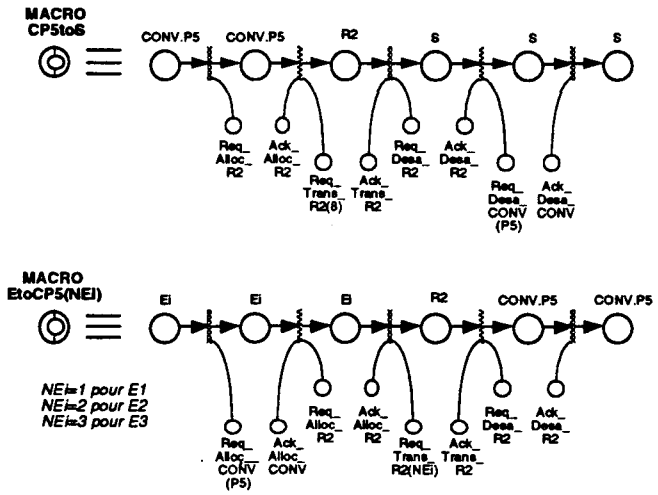
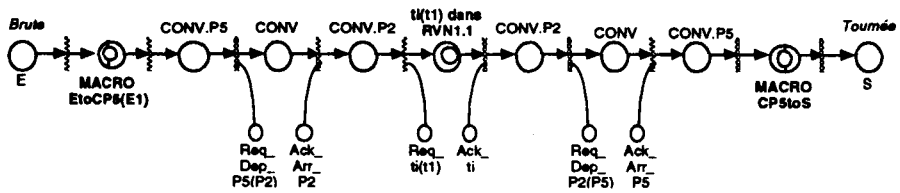
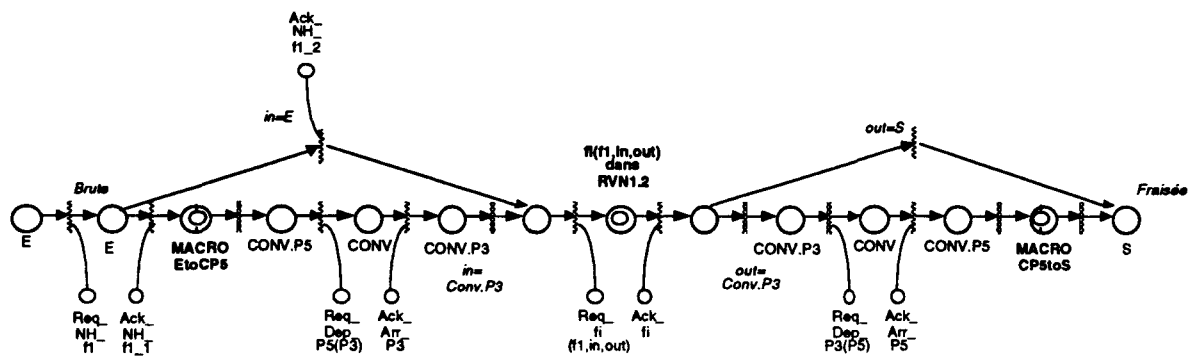


Figure AII.13 : Macro Gammes Opératoires Etendues

Pièces de type P1 (t1) :



Pièces de type P2 (f1) :



Pièces de type P3 (t2f2) :

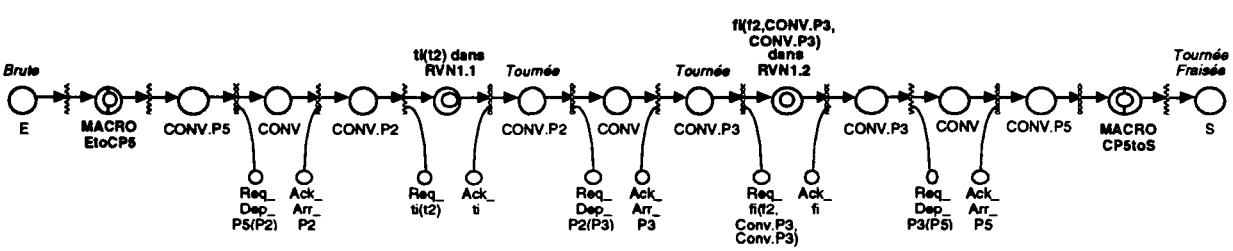


Figure AII.14 : Gammes Opératoires Etendues de niveau 1

D.5. Codage pour la simulation ou l'implantation

Les gammes opératoires étendues doivent être traduites en objets ADA afin de constituer la BLC (Base Logicielle Commune). Au total, nous aurons sept objets gammes : cinq objets actifs (deux gammes de niveau 0 et trois gammes de niveau 1) et deux objets passifs (macro gammes) (Figure AII.15).

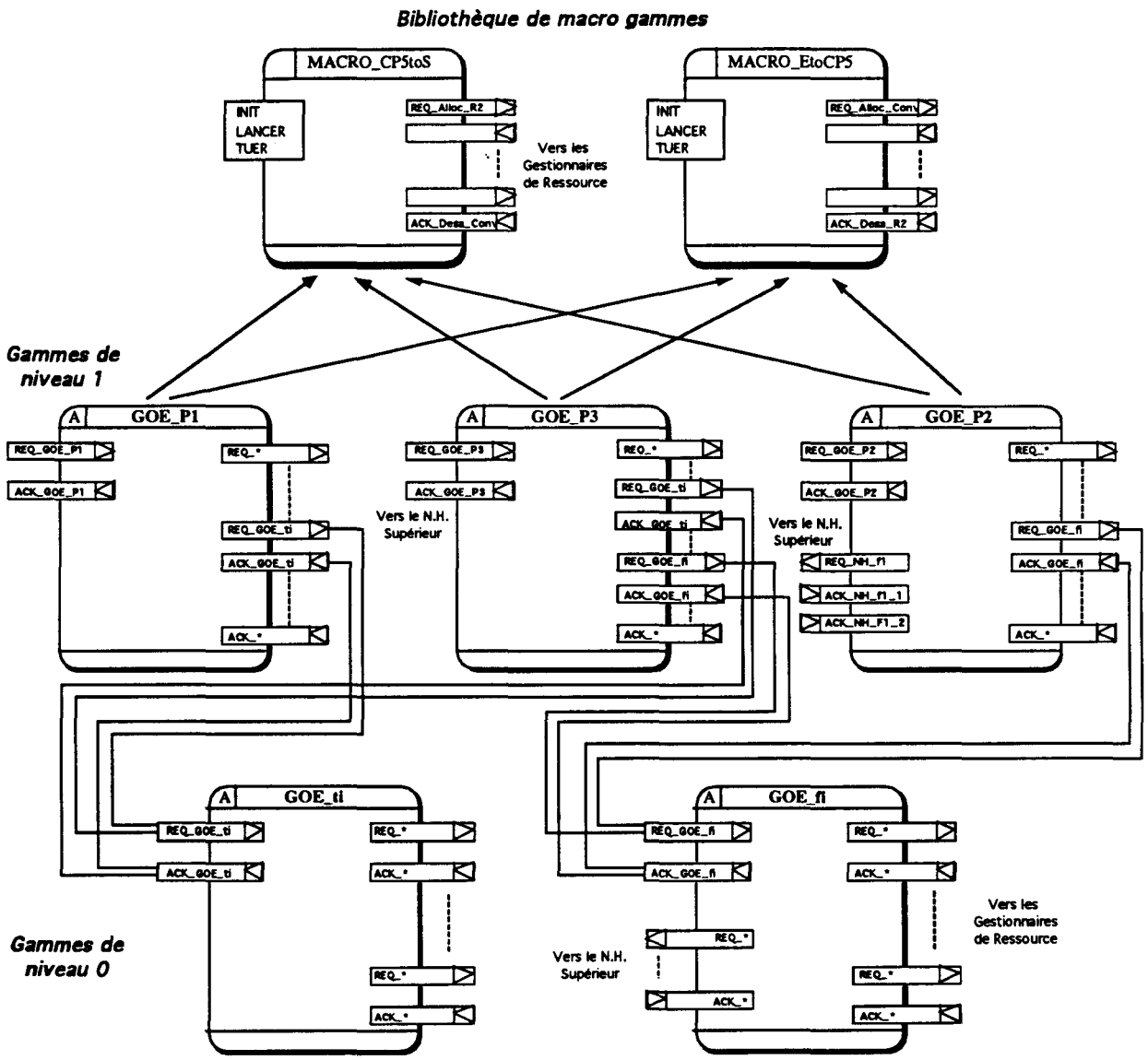


Figure AII.15 : Ensemble des objets gammes

La figure suivante décrit un exemple d'objet gamme, la GOE de niveau 1 correspondant aux pièces de type T1.

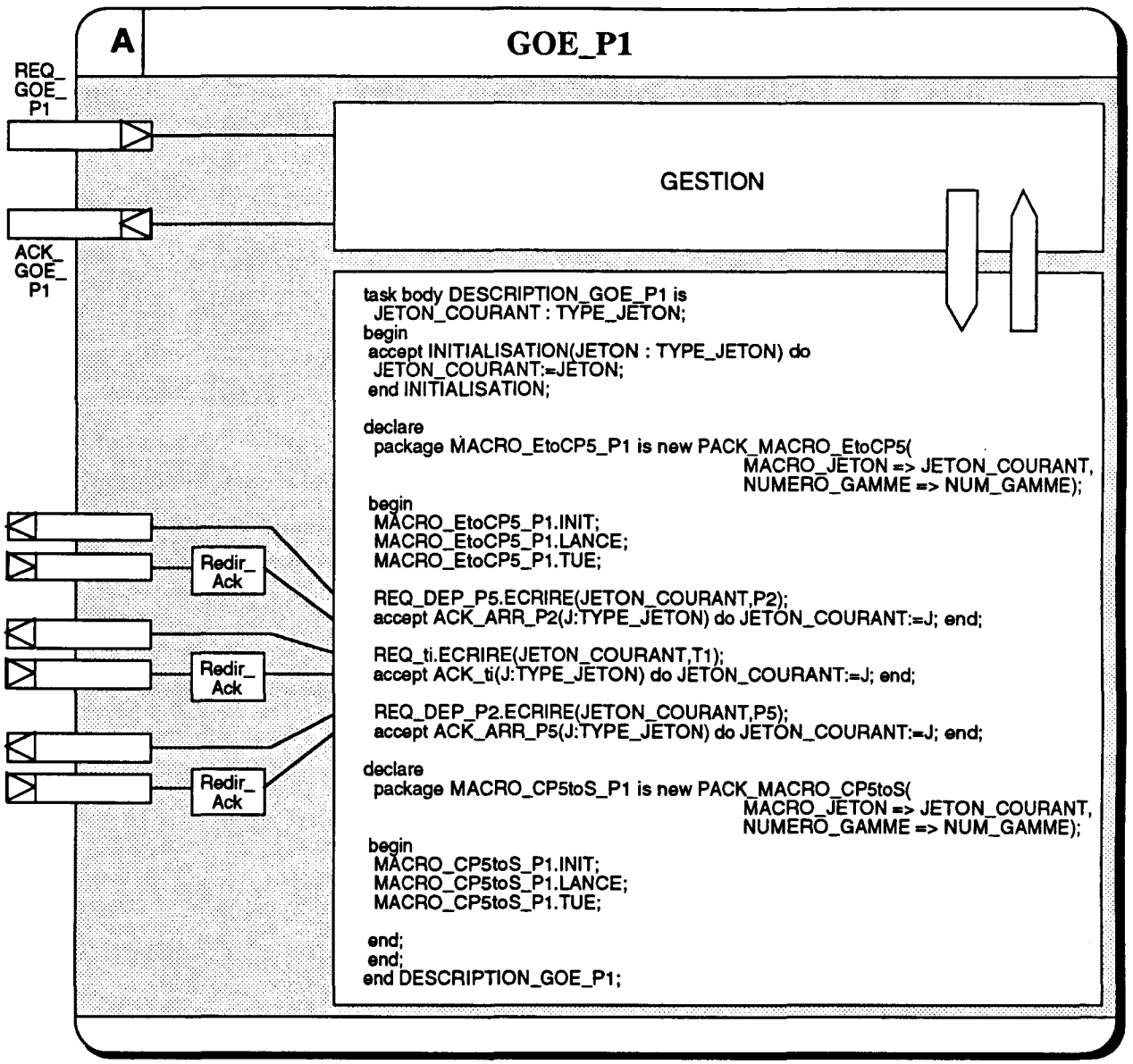


Figure AII.16 : exemple de transcription pour la gamme opératoire étendue P1 de niveau 1

Les objets gammes actifs comportent le type tâche décrivant la gamme opératoire étendue, la tâche de gestion ainsi que les tâches de redirection des accusés de réception. Les objets gammes non actifs (macro gammes) sont uniquement composés d'une description de la gammes opératoire étendue.

Les structures de données utilisées dans ces gammes sont identiques à celles décrites dans la partie D.2.b. du chapitre III. Les priorités des pièces sont statiques et ont été fixées a priori.

C. La partie opérative

C.1 Les ressources simples

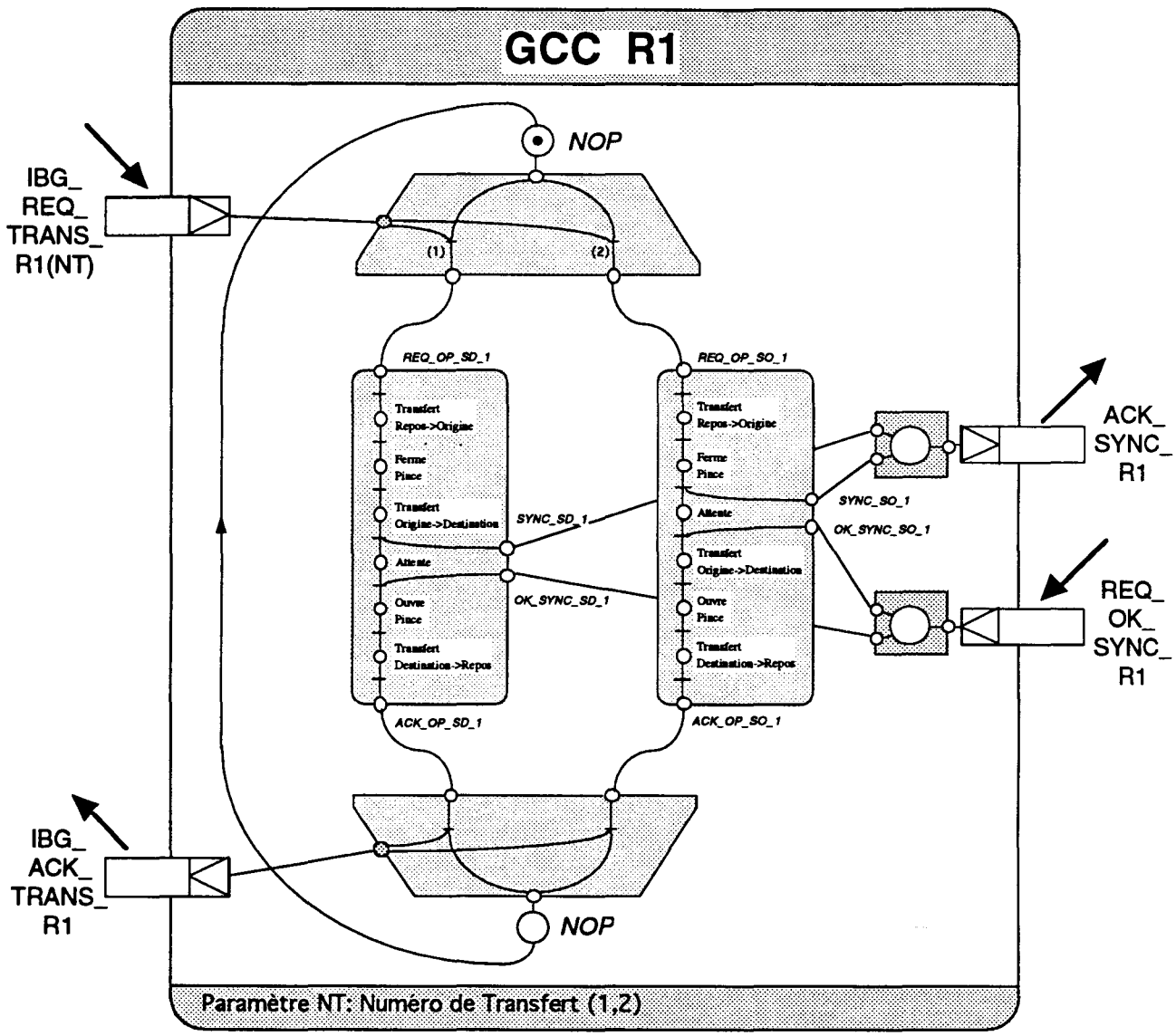
Les ressources simples sont au nombre de trois : les robots R1 et R2 et le tour T1.

C.1.a. Le robot R1

Le robot R1 a pour fonction le transfert de pièces du poste P2 du convoyeur vers le tour T1 et réciproquement. Au niveau du tour, les pièces sont maintenues à l'aide d'un mors, il doit donc y avoir synchronisation entre celui-ci et le robot lors du chargement/déchargement des pièces. Ce robot est programmable par langage spécialisé et dispose d'un seul effecteur. Celui-ci est suffisant car sa forme lui permet de saisir l'ensemble des pièces destinées au tour T1. Au vue de ces indications, nous obtenons le graphe de commande complet du robot R1 décrit par la figure AII.17.

Le filtre comportemental associé au robot R1 sera un pseudo filtre comportemental. En effet, mis à part l'adaptation du format des données, celui-ci n'a pas d'influence sur l'aspect fonctionnel du robot R1, il sera donc passif. Pour l'obtenir, il suffit d'instancier le filtre comportemental générique décrit dans la partie III.D.2.c.

Nous n'associons pas d'allocateur au robot R1. En effet, celui ci est uniquement dédié au chargement/déchargement du tour T1 à partir du poste P2 du convoyeur. Il n'est donc pas partagé par plusieurs ressources.



Transfert N°	Origine	Destination
(1)	P2	T1
(2)	T1	P2

Figure AII.17 : Graphe de commande complet du robot R1

C.1.b. Le robot R2

Le robot R2 doit pouvoir effectuer les transferts liants le convoyeur, le centre d'usinage CU1 et les tampons d'entrée/sortie. Au niveau du convoyeur, il y a deux possibilités de zone d'échange : les postes P3 et P5. Au niveau du centre d'usinage CU1, il y a aussi deux possibilités de zone d'échange : les tampons IO1 et IO2. Enfin, au niveau des tampons d'entrée/sortie, il y a quatre possibilités : S, E1, E2 et E3. Au total, les combinaison viables

représente un total de seize transferts.

Ce robot disposant d'un système de changement d'effecteurs, nous devons associer au pseudo filtre comportemental la fonction de sélection du bon effecteur en fonction du type et de l'état d'avancement de la pièce transférée (Figure AII.18).

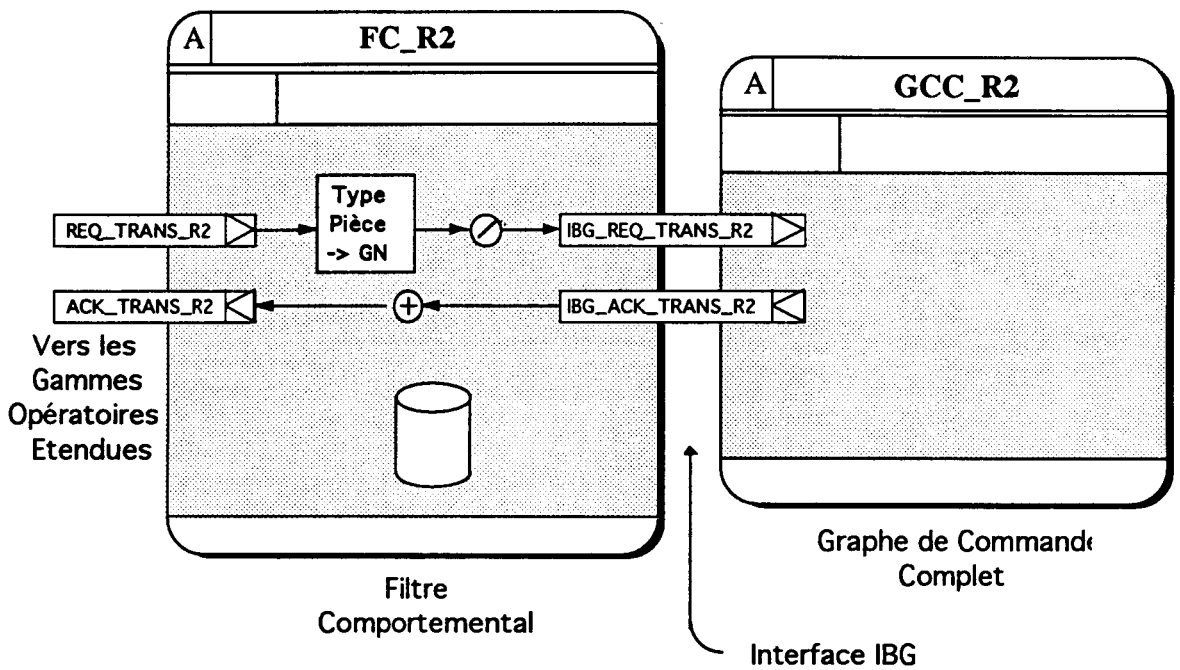
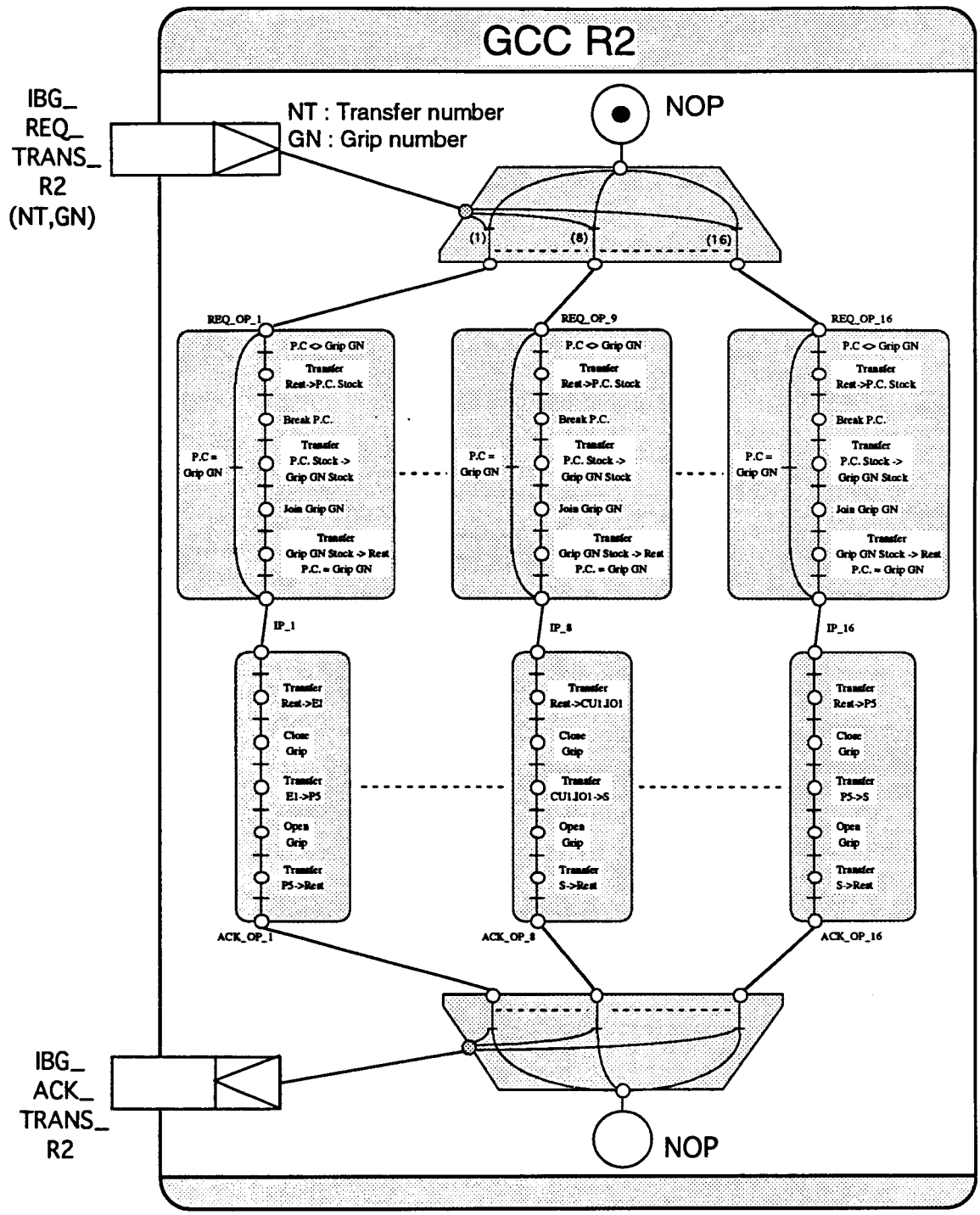


Figure AII.18 : Filtre comportemental du robot R2 et connexion avec le graphe de commande complet

Enfin, si nous considérons que le robot R2 est programmable par apprentissage, nous obtenons le graphe de commande complet décrit par la figure AII.19.

A ce robot, nous associons un allocateur. Celui-ci est une instanciation du modèle générique décrit dans la partie D.2.c du chapitre III. Nous avons choisi comme critère de sélection, celui qui tient compte de la priorité des pièces, celle-ci étant fixé a priori. Un autre critère favorisant, par exemple, la priorité au chargement/déchargement du tour pourrait très facilement être mis en œuvre.



NT	Transfert	NT	Transfert	NT	Transfert
1	E1->P5	7	CU1.IO2->P3	13	E1->CU1.IO2
2	E2->P5	8	CU1.IO1->S	14	E2->CU1.IO2
3	E3->P5	9	CU1.IO2->S	15	E3->CU1.IO2
4	P3->CU1.IO1	10	E1->CU1.IO1	16	P5->S
5	P3->CU1.IO2	11	E2->CU1.IO1		
6	CU1.IO1->P3	12	E3->CU1.IO1		

Figure AII.19 : Graphe de commande complet du robot R2

C.1.c. Le tour T1

Le tour T1 est une ressource simple de transformation nécessitant une synchronisation avec la ressource effectuant son chargement/déchargement (R1). Mis a part le rôle d'adaptation, le pseudo filtre comportemental n'a pas de fonction particulière à réaliser, il est donc passif. Finalement, nous obtenons donc le graphe de commande complet ainsi que le filtre comportemental décrits par la figure AII.20.

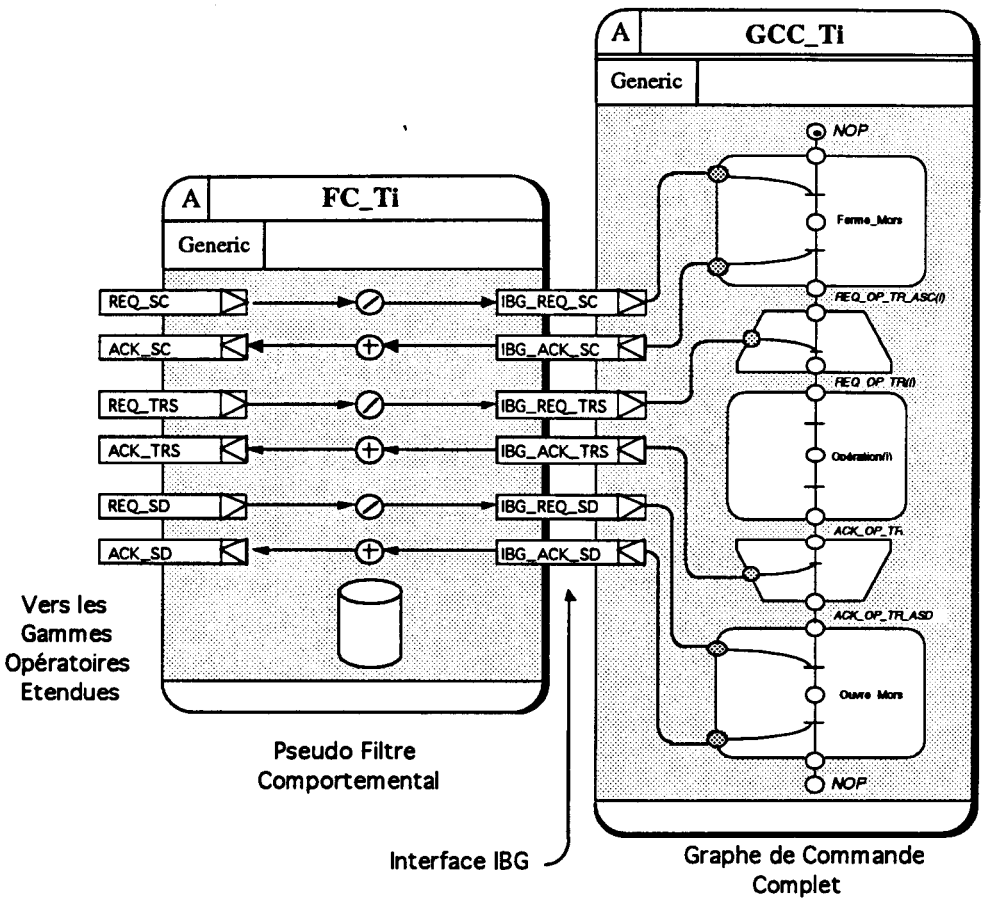


Figure AII. 20 : Pseudo filtre comportemental et graphe de commande complet génériques pour tout tour fonctionnellement identique à T1

Ici non plus, nous n'associons pas d'allocateur à cette ressource, celle-ci n'étant pas partagée.

C.2 Les ressources complexes

Les ressources complexes sont au nombre de deux : le centre d'usinage CU1 et le convoyeur.

C.2.a. Le centre d'usinage CU1

Ce centre d'usinage est une ressource complexe car il dispose de deux emplacements de chargement/déchargement pouvant être transférés vers une unique zone d'usinage.

Un filtre comportemental est donc nécessaire pour gérer le transfert de ces emplacements de stockage vers la zone opératoire. La figure suivante décrit ce filtre comportemental ainsi que le graphe de commande complet.

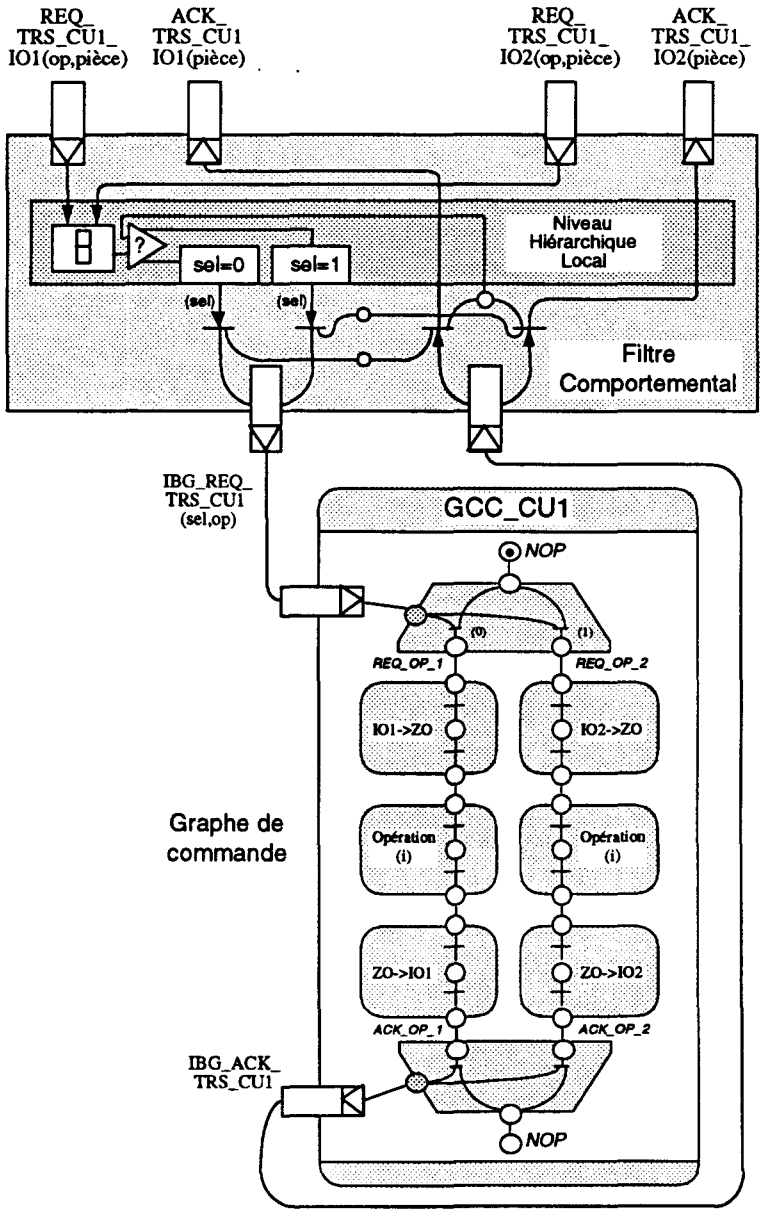


Figure AII.21 : Graphe de commande complet et filtre comportemental du centre d'usinage CU1

C.2.b. Le convoyeur

Pour commander cette ressource, deux types de graphes de commande complets génériques sont nécessaire : celui gérant les butées et celui gérant les aiguillages. La figure suivante décrit ceux-ci et la figure AII.24 décrit un exemple de traduction en Grafcet, pour l'implantation sur site (Automate TSX 47/30), et en ADA, pour la simulation.

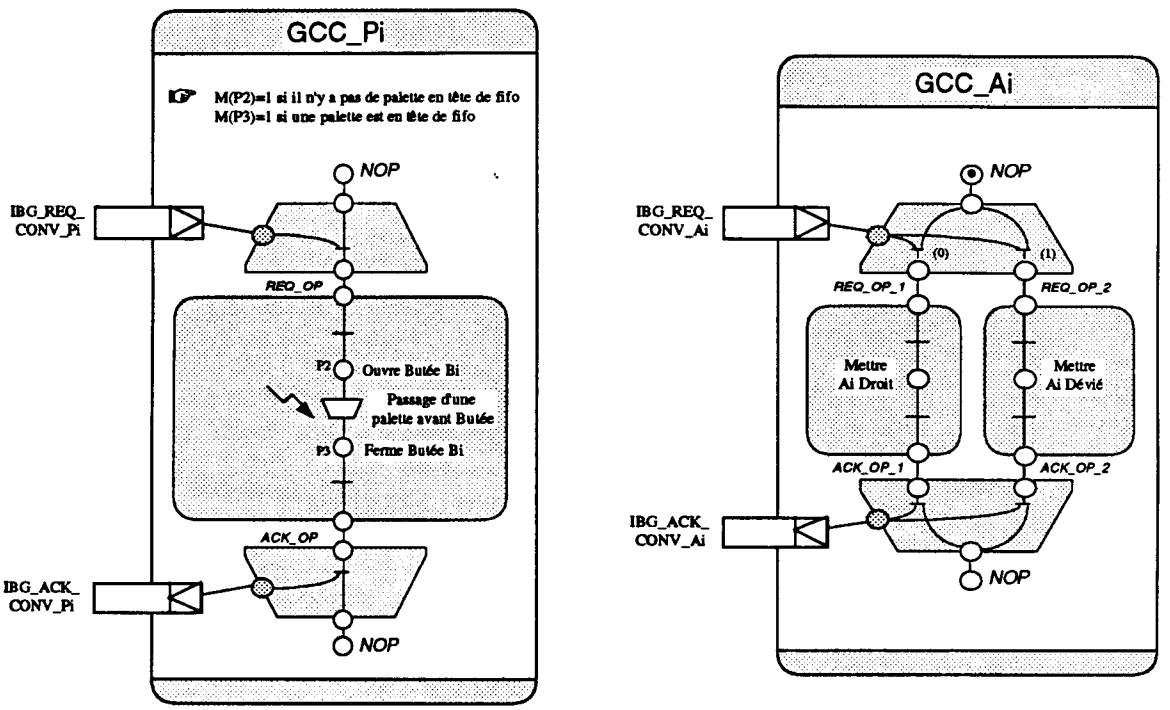


Figure AII. 22: Graphes de commande complets génériques pour les postes et les aiguillages du convoyeur

Pour obtenir l'ensemble des graphes de commande complets, il faut instancier les modèles génériques. Au total, huit sont requis (Figure AII.23)

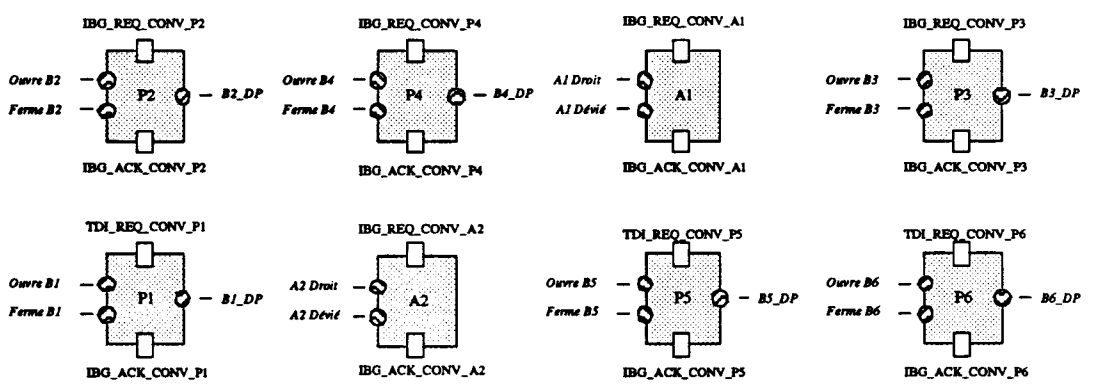


Figure AII.23 : Ensemble des graphes de commande complets du convoyeur

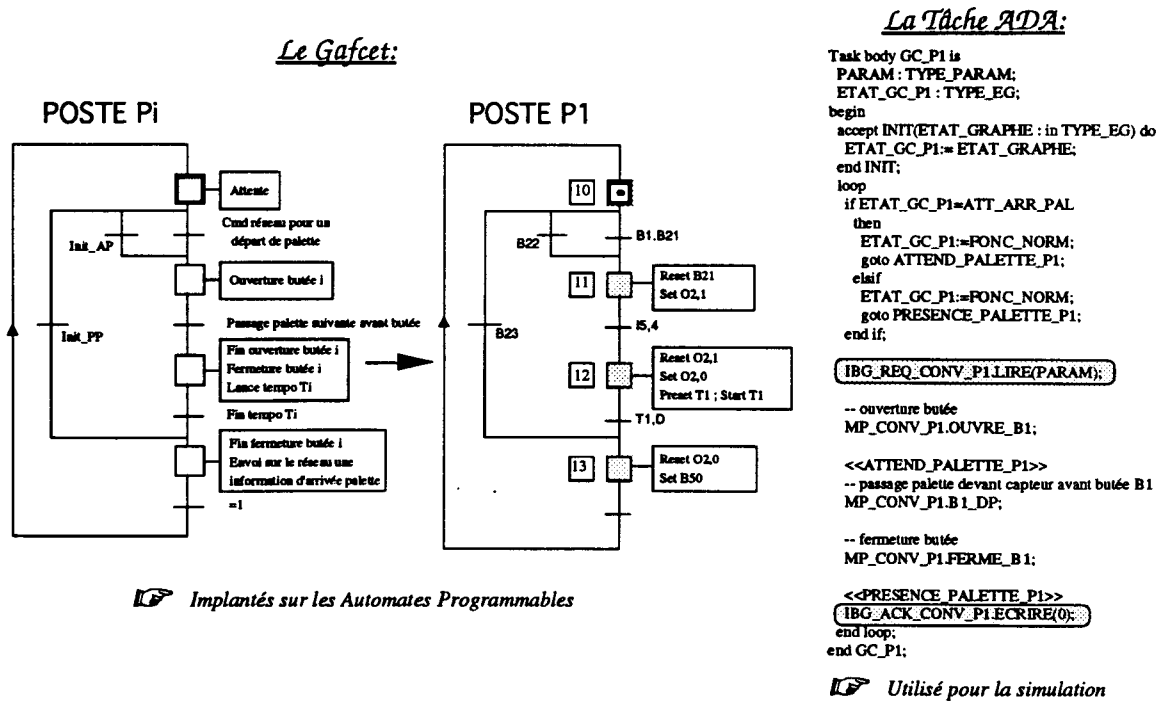


Figure AII. 24 : Traduction en Grafcet et en ADA du graphe de commande associé aux butées

Le filtre comportemental est ici primordial, il doit permettre de gérer les requêtes de services en fonction de l'état courant du convoyeur en évitant les conflits et en optimisant les transferts. Celui-ci est décrit par la figure suivante :

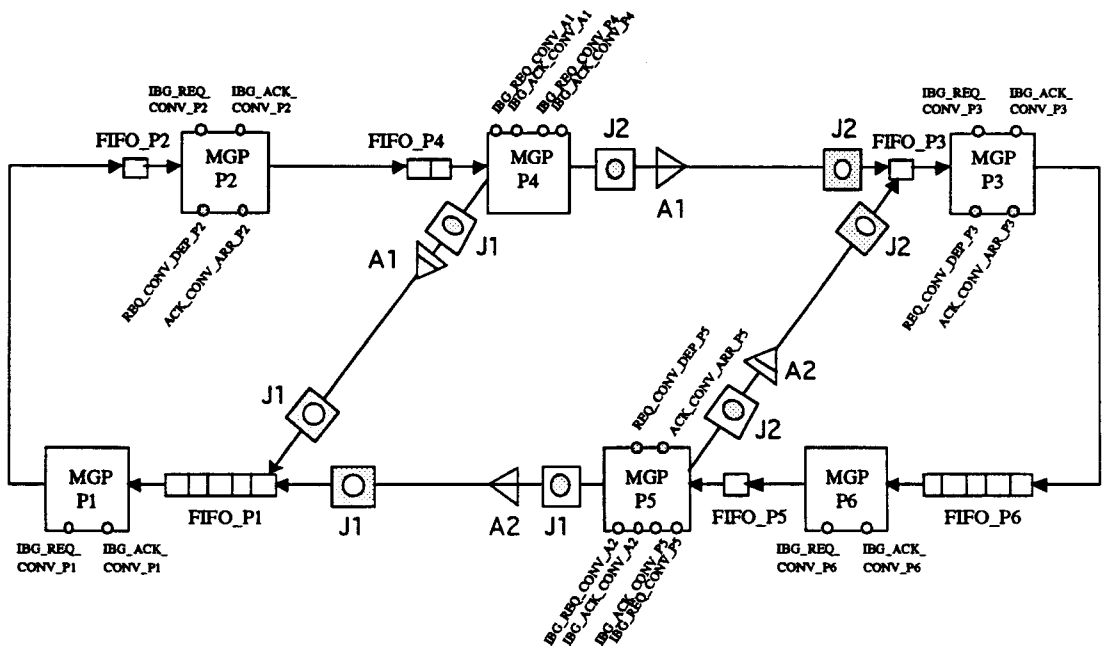


Figure AII.25 : Filtre comportemental du convoyeur

Les circuits bloquants de ce convoyeur sont au nombre de deux :

- le circuit C2 correspondant au chemin P1->P2->P4->P1,
- le circuit C3 correspondant au chemin P3->P6->P5->P3.

Pour éviter leur saturation, nous mettons en œuvre les sémaphores correspondants Ress_C2 et Ress_C3.

Nous pouvons ensuite déterminer les modules de gestion des postes (MGP Pi) suivant la fonctionnalité qu'ils devront remplir. La figure AII.26 décrit le MGP du poste P5 tandis que la figure AII.27 décrit le MGP du poste P2.

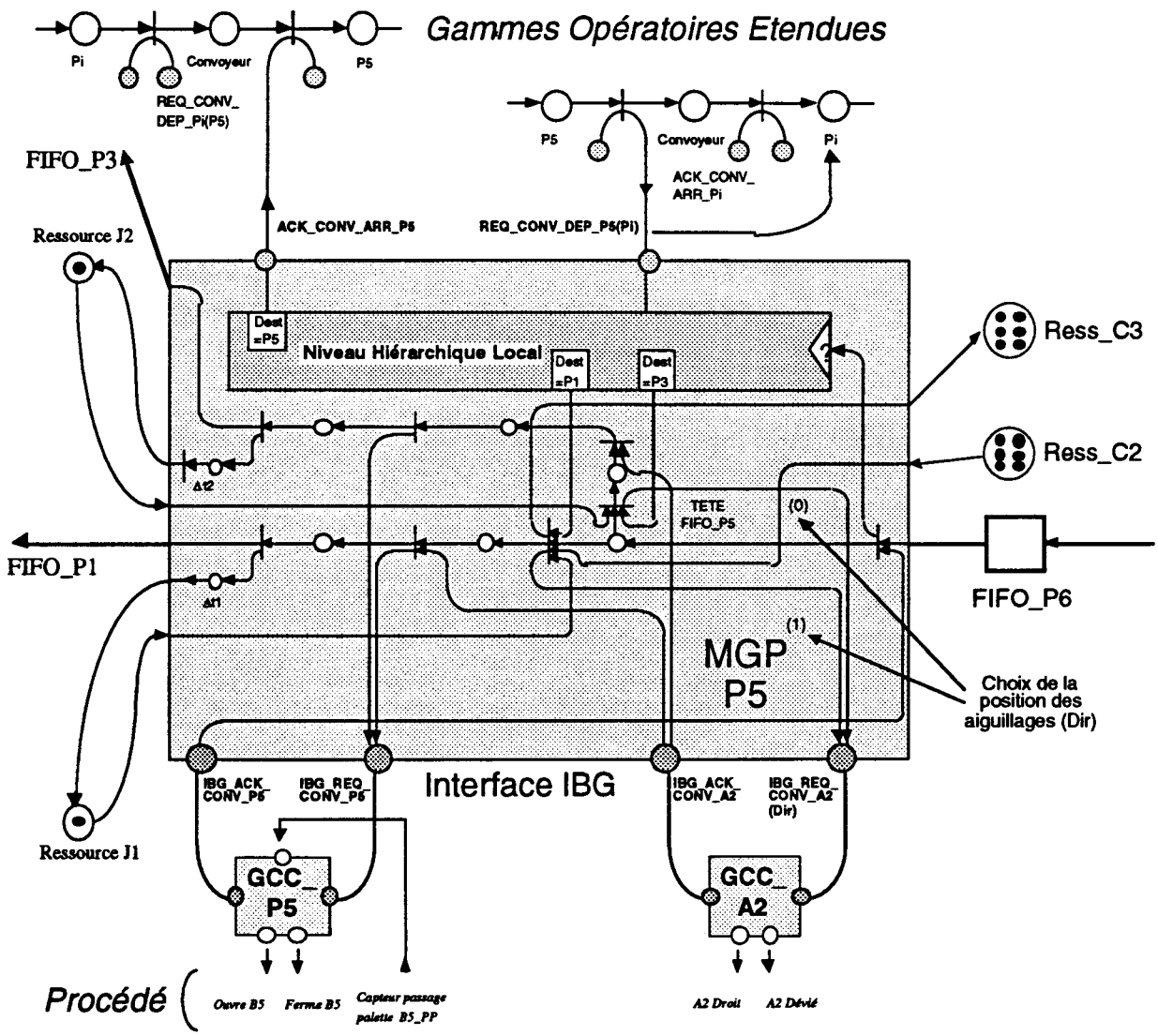


Figure AII.26 : MGP du poste P5

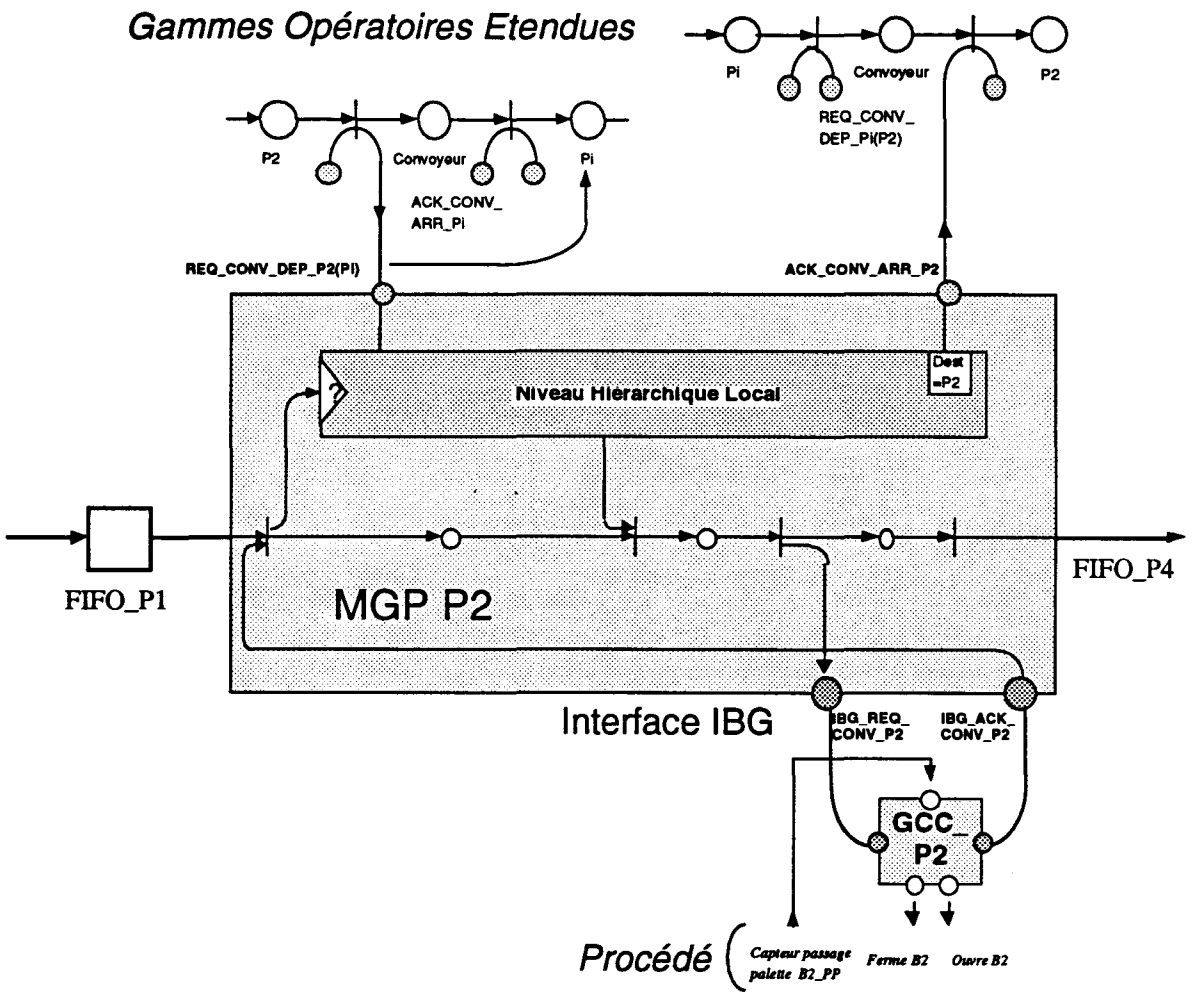


Figure AII.27 : MGP du poste P2

A titre d'exemple, nous pouvons détailler la traduction ADA du MGP du poste P2 :

```
task body MGP_P2 is
PARAM          : TYPE_PARAM;
LE_JETON       : TYPE_JETON;
LA_PALETTE     : TYPE_PALETTE;
DESTINATION    : TYPE_DESTINATION;
begin
 accept LANCER;
 loop
  -- recoit un arrive palette du GCC du poste P2
  IBG_ACK_CONV_P2.LIRE(PARAM);
  FIFO_P2.LIRE(LA_PALETTE);
  if LA_PALETTE.POSTE_DESTINATION = P2 then
   --envoyer ACK vers la gamme opératoire etendue correspondante
   NUM_GOE := LA_PALETTE.JETON.NUM_GAMME;
   ACK_CONV_ARR_P2(NUM_GOE).ECRIRE(LA_PALETTE.JETON);
   --attendre REQ de depart poste P2
```

```
    REQ_CONV_DEP_P2.LIRE(LE_JETON,DESTINATION);
    --mise a jour des donnees de la palette
    LA_PALETTE.JETON := LE_JETON;
    LA_PALETTE.POSTE_DESTINATION := DESTINATION;
    FIFO_P2.ECRIRE(LA_PALETTE);
  end if;

  -- laisser partir la palette vers P4
  loop
  if not FIFO_P4.PLEIN
   then
    FIFO_P2.PRENDRE(LA_PALETTE);
    FIFO_P2.AJOUTER(LA_PALETTE);
    --envoie un depart palette au GCC du poste P2;
    PARAM := NUL_PARAM;
    IBG_REQ_CONV_P2.ECRIRE(PARAM);
    exit;
   else
    delay
  end if;
  end loop;

end loop;
end MGP_P2;
```

Les tâches correspondant à l'ensemble des MGP ainsi que les objets FIFO et les objets sémaphores sont inclus dans un seul package pour donner l'objet filtre comportemental du convoyeur (Figure AII.28).

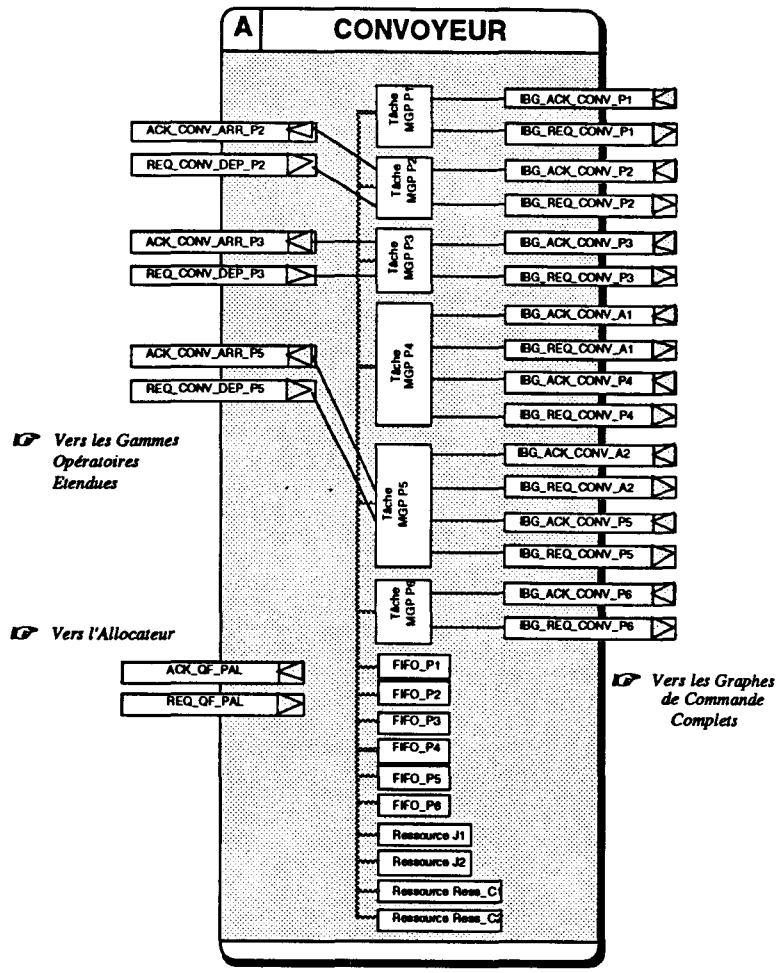


Figure AII.28 : Composition de l'objet filtre comportemental du convoyeur

D. Intégration

L'ensemble des objets décrits codés en ADA sont rassemblés pour donner la Base Logicielle Commune du logiciel de contrôle/commande de la cellule.

A cette BLC, nous avons ajouté les paquetages permettant la simulation ainsi qu'un paquetage gérant un synoptique animé écris avec la norme graphique X11. La figure AII.29 montre un exemple de sortie écran lors d'une simulation.

E. Implantation sur site

L'ensemble des graphes de commande complets sont implantés en langages métiers sur les automates ou commandes numériques dédiés aux ressources. Les filtres comportementaux, le suivi des Gammes Opératoires Etendues, les allocateurs de ressources ainsi que le N.H.

supérieur sont implantés en ADA sur la station de travail faisant office de superviseur de la cellule (Figure AII.30).

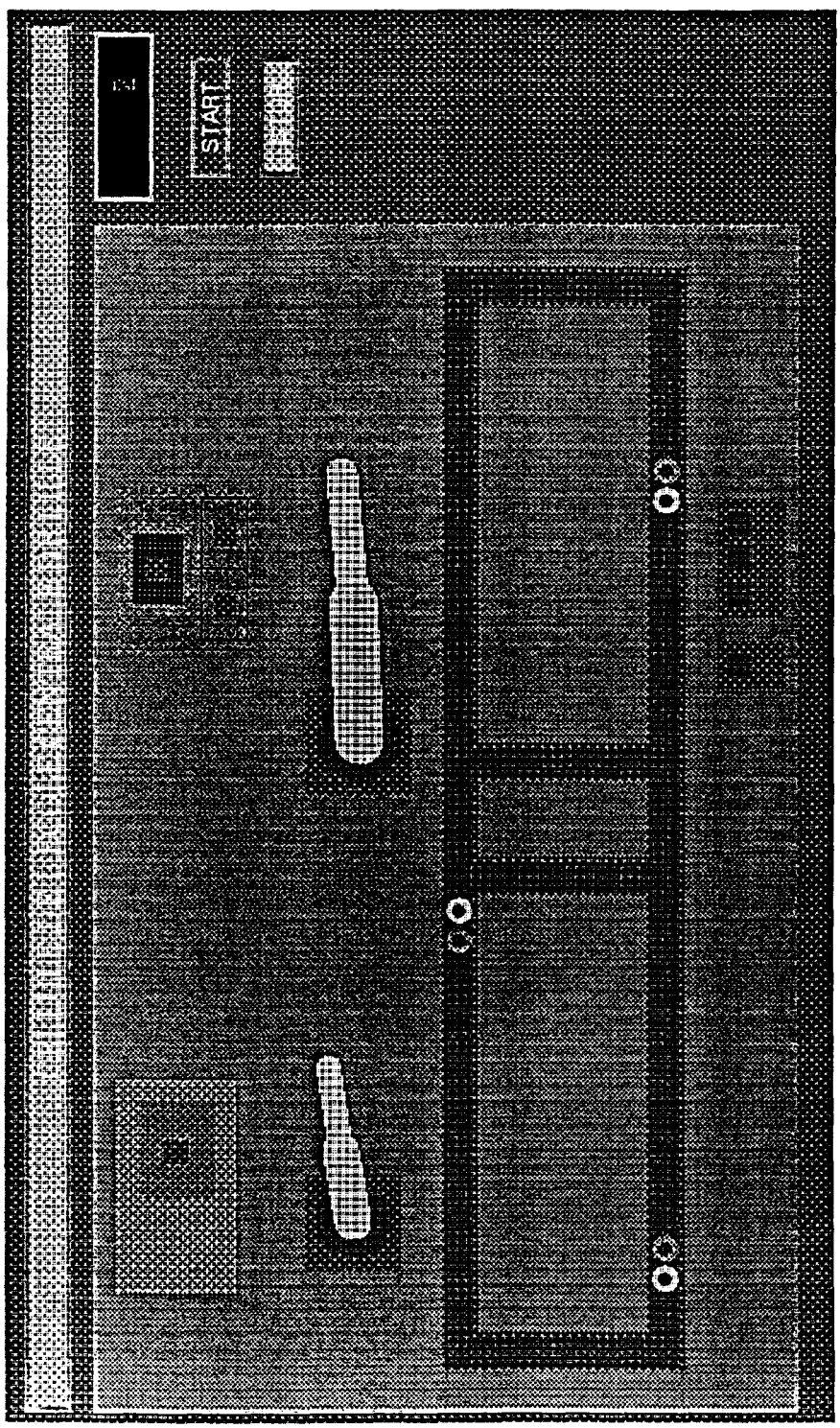


Figure AII.29 : Exemple de sortie écran lors d'une simulation

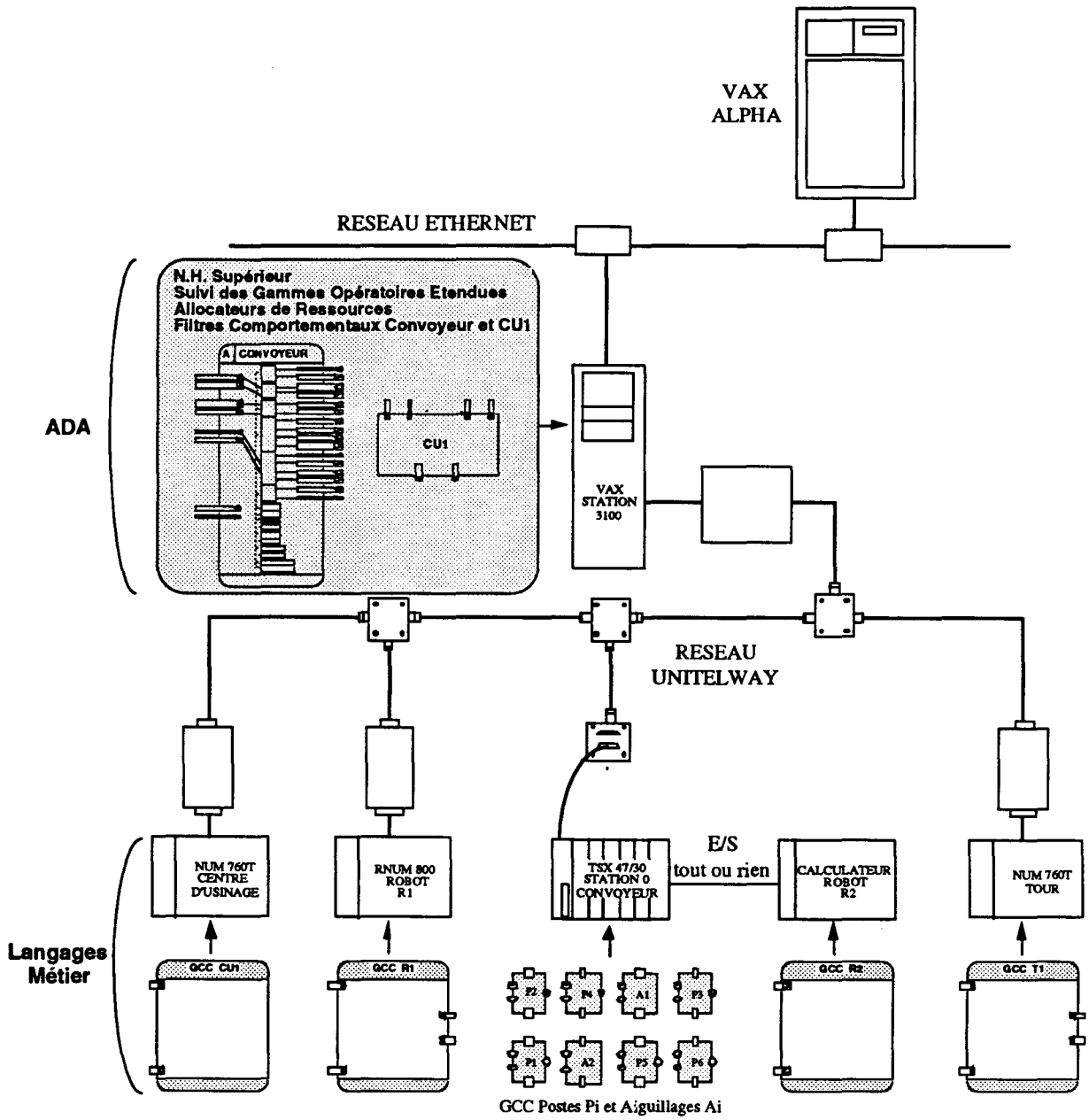


Figure AII.30 : Répartition du logiciel

SOMMAIRE DETAILLE

SOMMAIRE DETAILLE

Pages