**Université des Sciences et Technologies de Lille**

**N° d'ordre : 2771**

# THESE

pour obtenir le grade de

## DOCTEUR DE L'UNIVERSITE LILLE 1

en

## Productique Automatique et Informatique Industrielle

Présentée et soutenue publiquement
par

**Djibril NDIAYE**
**Maître en Electronique Electrotechnique et Automatique**

**Le 18 septembre 2000**



# APPORT DES TECHNOLOGIES ORIENTEES OBJET DANS L'ETUDE ET LA MISE EN PLACE D'UN REFERENTIEL DE CONCEPTION POUR LES SYSTEMES AUTOMATISES DE PRODUCTION

## *JURY*

| | | |
|---|---|---|
| Mme. Mireille BAYART | ***Présidente*** | Professeur à l'Université de Lille1 |
| M. Jean BEZIVIN | ***Rapporteur*** | Professeur à l'Université de Nantes |
| M. François VERNADAT | ***Rapporteur*** | Professeur à l'Université de Metz |
| M. Jean-Pierre BOUREY | ***Directeur de thèse*** | Professeur à Ecole Centrale de Lille |
| M. Michel BIGAND | ***Codirecteur de thèse*** | MdC à l'Ecole Centrale de Lille |
| M. Didier CORBEEL | ***Examinateur*** | MdC à l'Ecole Centrale de Lille |
| M. Jean-Claude GENTINA | ***Examinateur*** | Professeur à l'Ecole Centrale de Lille |
| M. Gilles GONCALVES | ***Examinateur*** | Professeur à l'Université d'Artois |
| Mme. Monique PICAVET | ***Examinatrice*** | MdC HDR à l'Université de Lille 1 |

# *A l'école du Professeur Caméléon*

*Si j'ai un conseil à vous donner, je vous dirai : ouvrez votre cœur ! Et surtout allez à l'école du caméléon ! C'est un très grand professeur. Si vous l'observez vous verrez…Qu'est-ce que le caméléon.*

*D'abord quand il prend une direction, il ne détourne jamais sa tête. Donc, ayez un objectif précis dans votre vie, et que rien ne vous détourne de cet objectif.*

*Et que fait le caméléon ? Il ne tourne pas la tête, mais c'est son œil qu'il tourne. Le jour où verrez un caméléon regarder, vous verrez : c'est son œil qu'il tourne. Il regarde en haut, il regarde en bas. Cela veut dire : Informez vous !*

*Quand il arrive dans un endroit, le caméléon prend la couleur du lieu. Ce n'est pas de l'hypocrisie ; c'est d'abord la tolérance, et puis le savoir vivre…*

*Et que fait-il le caméléon ? Quand il lève le pied, il se balance, pour savoir si les deux pieds posés ne s'enfoncent pas. C'est après seulement qu'il va déposer les deux autres. Il balance encore …il lève…Cela s'appelle : la prudence dans la marche.*

*Et sa queue est préhensible. Il l'accroche. Il ne se déplace pas comme ça…Il l'accroche, afin que si le devant s'enfonce, il reste suspendu. Cela s'appelle assurer ses arrières…*

***Amadou Hampâté Bâ***
*Sur les traces d'Amkoulel l'enfant peul, ACTES SUD*

# Remerciements

*Le travail présenté dans ce mémoire a été effectué au LAIL (Laboratoire d'Automatique et d'Informatique Industrielle de Lille) au sein de l'équipe Production Flexible Manufacturière à l'Ecole Centrale de Lille sous la direction scientifique de Monsieur le Professeur Jean-Pierre BOUREY et de Monsieur Michel BIGAND Maître de Conférences à l'Ecole Centrale de Lille. Je tiens à les remercier très vivement pour l'accueil, l'encadrement, le soutien et les précieux conseils dont j'ai bénéficié tout au long de ce travail.*
*Ce travail a pu être mené à terme grâce aux précieuses contributions de Monsieur Didier CORBEEL, Maître de Conférences à l'Ecole Centrale de Lille.*

*Je suis très reconnaissant à M. Jean BEZIVIN, Professeur à l'Université de Nantes et à M. François VERNADAT, Professeur à l'Université de Metz, pour l'honneur qu'ils me font en acceptant d'examiner ce travail et d'être les rapporteurs de cette thèse. Leurs remarques et conseils m'ont permis d'améliorer sensiblement la qualité du mémoire.*

*Je remercie tous les membres du jury : Mme. Mireille BAYART, Professeur à l'Université de Lille1, M. Jean-Claude GENTINA, Professeur à l'Ecole Centrale de Lille, M. Gilles GONCALVES, Professeur à l'Université d'Artois, et Mme. Monique PICAVET, MdC HDR à l'Université de Lille 2, pour l'honneur qu'ils me font en examinant ce travail.*

*Je tiens également à adresser mes remerciements à tous les membres du LAIL qui, de par leur disponibilité, m'ont aidé tout au long de mes travaux ; j'associe également Monsieur Max VANGREVENINGE pour la reprographie de ce mémoire.*

*Une pensée toute particulière à ma Famille, la Famille Mbaye, la Famille Ndour et à tous ceux qui m'ont adressé un moment donné, un sourire ou un mot d'encouragement.*

# S O M M A I R E

# Introduction générale

## Contexte de l'étude

De nos jours, les entreprises évoluent dans un contexte concurrentiel acharné et doivent répondre à des normes précises en termes de qualité, de protection de l'environnement et d'une manière générale de satisfaction du client. Ces exigences obligent les entreprises à prendre en considération de nombreux paramètres internes et externes comme par exemple les coûts, les délais, la fiabilité logistique, la pénétration internationale, la productivité, la compatibilité biologique du conditionnement, le taux de recyclage constituent l'environnement d'élaboration du produit etc. [BOC98].

La maîtrise de tous ces paramètres demande l'intervention de plusieurs disciplines aussi variées que les sciences économiques, les sciences humaines et sociales, les sciences pour l'ingénieur. Les activités de l'entreprise sont alors tout aussi diversifiées et en permanente croissance. Chaque activité s'intéresse à un ou plusieurs aspects de l'entreprise et développe des méthodes, outils afin de créer les modèles pour spécifier la structure d'une ou plusieurs entité organisationnelle de l'entreprise, pour visualiser et contrôler son architecture, pour mieux comprendre son organisation et ainsi déceler les possibilités de simplification et de réutilisation et, enfin, pour gérer les risques encourus, [BRJ00].

Un défi important que les entreprises doivent surmonter consiste alors à assurer la coordination de tous ses moyens (méthodes, outils, modèles) et activités hétéroclites afin de former un ensemble d'éléments synergiques. Plusieurs travaux (CIMOSA, GRAI, PERA, etc.) ont alors été initiés dans ce sens, et ont fourni des cadres de référence utilisés comme mécanisme d'unification sémantique ou de partage de connaissances [VER96].

Nous nous intéressons plus particulièrement aux systèmes de production de l'entreprise qui ont rapidement évolué, et doivent concilier des objectifs contradictoires : le coût, la qualité et les délais. Le système de production est l'entité qui reflète le plus la maîtrise technologique de l'entreprise, l'intervention humaine est ainsi de plus en plus réduite, on parle alors de Systèmes Automatisés de Production (SAP) [FRA87]. La maîtrise de la complexité des SAP et la coordination de leurs activités passe nécessairement par le partage et l'échange d'informations cohérentes entre les activités de l'entreprise. Parmi les travaux les plus importants dans le domaine du Génie industriel, nous pouvons notamment cité la norme BASE-PTA qui fournit un modèle considéré comme cadre de référence destiné à servir de base pour l'échange de données entre outils XAO [AFN95].

## Démarche

Les résultats les plus importants favorisant l'intégration des activités de l'entreprise (ou une entité de l'entreprise) et abordant l'aspect informationnel sont basés sur des cadres de références. C'est une approche qui garantit la cohérence des informations échangées et partagées ; elle propose des modèles de références cohérents qui sont ensuite adaptés, particularisés à des cas précis.

Notre problématique au sein de l'équipe REFERENTIEL du LAIL est différente : nous ne préjugeons pas en effet de ce que devrait être le système, mais nous partons des moyens et activités préexistants que nous devons intégrer. Nous devons alors concilier deux critères fondamentaux du Génie industriel :

(1) l'indépendance des activités et des moyens ;

(2) la cohérence entre ces mêmes activités et moyens.

Nous proposons alors une démarche pour intégrer des moyens et gérer la totalité de l'information. Nous nous basons pour cela sur des outils et méthodes du Génie Logiciel :

(1) une approche orientée objet utilisant le langage UML

(2) la métamodélisation et la réutilisation de concepts de modélisation (patterns, frameworks).

## Réalisations

Après avoir défini le **cahier des charges** du **référentiel**, nous allons déterminer une **architecture** du référentiel sur trois couches pour la structuration de l'ensemble des modèles et informations que nous devons gérer. Ensuite nous allons formaliser une **démarche pragmatique de réutilisation** des **patterns** que nous avons créés. Ces patterns seront réutilisés pour faciliter la démarche de **métamodélisation** et **d'intégration** des modèles.

# Plan de lecture du mémoire

Ce mémoire est divisé en deux parties, la première comporte les chapitres 1 et 2 ; la deuxième partie les chapitres 3, 4 et 5.

**Partie A : Contexte de l'étude et définition du cahier des charges du référentiel**

- **Chapitre 1 : Entreprise et système de production : évolution des concepts**

Dans ce chapitre nous étudions les principaux paradigmes dans les entreprises manufacturières. Nous étudions ensuite les différentes architectures des SAP, puis nous présentons le projet CASPAIM. Enfin nous définissons le cahier des charges du référentiel.

- **Chapitre 2 : Problématique de la cohérence de l'information dans les SAP**

Après l'étude des cadres de références CIMOSA, et BASEPTA, notamment leur aspect informationnel, nous positionnons notre approche. Nous considérons l'indépendance des activités de conception et l'hétérogénéité de leurs moyens comme un état de fait ; la démarche que nous proposons vise d'une part à assurer le partage de l'information avec une interprétation globalement cohérente, d'autre part à préserver l'indépendance des activités.

**Partie B : Démarche d'intégration suivant une approche orientée objet par métamodélisation et réutilisation**

- **Chapitre 3 : Intégration suivant une approche orientée objet par métamodélisation**

Dans ce chapitre nous justifions le choix d'une approche orientée objet mise en œuvre à l'aide du langage UML. Puis nous proposons une démarche d'intégration des concepts de modèles préexistants par métamodélisation. Ensuite nous définissons une architecture du référentiel, pour la réalisation d'une démarche d'intégration.

- **Chapitre 4 : Réutilisation de concepts de modélisation**

La réutilisation est un atout principal d'une approche orientée objet. Après avoir montré l'apport de la réutilisation dans notre démarche d'intégration, nous formalisons une démarche de réutilisation. Nous créons des patterns de conception et des patterns métiers qui seront par la suite réutilisés dans la construction des différents métamodèles UML.

- **Chapitre 5 : Etude du référentiel CASPAIM**

A travers le dernier chapitre de ce mémoire, nous appliquons la totalité de notre démarche proposée dans les chapitres 3 et 4 au projet CASPAIM. Nous mettons alors en œuvre la réutilisation des patterns que nous avons créés au chapitre 4. Enfin nous présentons le logiciel CASPAIM SOFT que nous avons développé pour valider concrètement les métamodèles obtenus.

# Partie A

## Résumé

**Cette première partie du mémoire présente le contexte général de nos travaux.**

**A travers le premier chapitre, nous montrons les évolutions du monde industriel qui ont progressivement mené les systèmes d'information à occuper une position centrale dans les systèmes automatisés de production.**

**Dans le deuxième chapitre nous nous intéressons plus précisément aux problèmes de modélisation et d'intégration des données, et nous positionnons notre démarche par rapport aux approches les plus connues.**

# Introduction Partie A

*« Les variations du contexte économique que l'on constate actuellement forcent les entreprises à devenir plus flexibles et réactives pour pouvoir anticiper et s'adapter aux changements fréquents auxquels elles doivent faire face. Ces changements, de nature économique, technologique, sociologique ou relatifs à l'environnement, peuvent avoir des impacts profonds sur la structure de l'entreprise, allant de la refonte de certaines procédures opératoires ou de certains systèmes d'information jusqu'à la remise en cause de la structure organisationnelle, voire de la stratégie globale de l'entreprise ou de sa culture.* ***La maîtrise du changement*** *devient une des clés du succès pour nombre d'entreprises confrontées à une forte concurrence »,* [VER99].

Les entreprises sont ainsi amenées à revoir constamment leur organisation et par conséquent leurs schémas directeurs évoluent sans cesse.

La compétitivité se mesure en terme de trois objectifs QCD : produit de bonne **Qualité**, fabriqué avec un **Coût** faible, et livré dans des **Délais** courts. L'équilibre entre ces trois objectifs devient alors le principal enjeu, comme le montre [AFI94], [PRI94], [WES91].

La rigueur de la démarche d'adaptation du système de production constitue pour chaque entreprise un atout majeur. Les **systèmes de production** deviennent, de par leur niveau technologique et leur degré d'innovation, les éléments essentiels de compétitivité des entreprises industrielles.

D'une manière générale, on considère comme système tout élément doté d'une structure, qui réalise une activité, une fonction, évoluant dans le temps et dans un environnement pour finaliser un besoin initialement formulé. Un système de production est un espace constitué de trois familles d'éléments : les **produits**, les **moyens** de production et les **opérateurs**. Dans cet espace, se déroulent les opérations de production dans le but de faire évoluer les produits vers un état précis, [ROD89]. Les entreprises atteignent des tailles critiques et leurs systèmes de production deviennent de plus en complexes. **La maîtrise des flux d'information** devient alors un challenge important dans la vie de toute entreprise.

Dans le premier chapitre nous présentons les principales organisations des entreprises et les architectures des systèmes automatisés de production, puis nous définissons le cahier des charges d'un système d'information (appelé référentiel de conception) pour les systèmes automatisés de production.

Nous traitons ensuite dans le deuxième chapitre les problèmes liés à la modélisation et à l'intégration des activités de l'entreprise. Ensuite nous proposons une approche pour l'intégration des modèles après avoir étudié les travaux existant dans ce domaine.

# CHAPITRE 1.

# Entreprise et Système de production : évolution des concepts

## Résumé

**Ce chapitre présente les principaux paradigmes dans le monde industriel, en s'intéressant plus particulièrement aux entreprises manufacturières, depuis la taylorisation dans les années cinquante au Concurrent Engineering et le Computer Integrated Manufacturing. Nous étudions ensuite les différentes architectures des Systèmes Automatisés de Production qui se doivent d'être plus flexibles, réactifs et fiables. La conception de ces Systèmes Automatisés de Production fait appel à des méthodologies du Génie industriel. Dans ce cadre le LAIL a développé le projet CASPAIM. La gestion de l'information provenant de sources multiples est effectuée au sein d'un référentiel de conception.**

## 1.1. Evolution des entreprises industrielles

### 1.1.1. Evolution jusqu'aux années 80

L'organisation des entreprises industrielles a évolué avec la performance et la complexité croissante des produits. Cette complexité est due à une concurrence vive à l'échelle mondiale et une exigence des clients en terme de qualité, prix et délais de livraison. L'évolution des entreprises est ainsi fortement liée à l'environnement économique [JAG93].

Dans les années 50, les entreprises s'appuient sur le principe du taylorisme pour la reconstruction de l'après-guerre. Le système qualité est principalement basé sur le contrôle avec vérification de la conformité des pièces réputées bonnes ou mauvaises.

Les années 60 bénéficient d'un environnement économique favorable. Une production de masse engendre de grands stocks et l'automatisation se développe progressivement autour d'un produit unique.

Les premiers chocs pétroliers dans les années 70 associés à la saturation de nombreux marchés obligent les entreprises à revoir leur mode de production. La taylorisation et la rigidité des systèmes automatisés sont remises en cause par les évolutions du marché et des technologies. Ainsi la CAO (Conception Assistée par Ordinateur), la commande numérique et les automates programmables permettent une flexibilité croissante.

La reprise économique des années 80 permet aux entreprises de produire mieux, dans une démarche de réduction des coûts. L'objectif « zéro défaut » impose une rigueur accrue dans l'organisation du suivi de production et de la maintenance. Les outils informatiques connaissent une grande expansion, avec l'introduction des réseaux locaux, des Systèmes de gestion de Bases de Données relationnels (SGBD) et de puissantes stations de travail. Cette période voit la naissance du concept Computer Integrated Manufacturing (CIM).

### 1.1.2. Le concept CIM

Le concept CIM ([RAN86], [WAL92]) propose une démarche globale d'intégration des activités de l'entreprise. C'est un facteur clé pour faire face à toutes les mutations (intégration, automatisation, flexibilité) et améliorer la productivité de l'entreprise.

L'objectif initial du concept CIM est l'intégration des « îlots d'automatisation » résultant de l'automatisation intensive des années 70. L'intégration de la production est alors considérée comme une priorité. Les autres services liés à la production sont alors implicitement intégrés.

Le concept CIM propose aussi une intégration par niveau. L'intégration au sein d'une entreprise vise à développer des solutions et des outils informatiques pour faciliter la coordination du travail et la circulation de l'information entre les différents acteurs de l'entreprise. Au niveau humain, cette intégration facilite la prise de décision, réduit les délais et améliore la communication car elle décloisonne l'information ; au niveau de la production, elle offre un meilleur contrôle des équipements automatisés [HAN96].

D'une manière générale, le concept CIM regroupe un ensemble de méthodes conçues pour intégrer les données et les informations des différents secteurs d'activité d'une entreprise ; celle-ci n'est plus considérée comme une simple unité de production.

Aux Etats-Unis, ICAM (Integrated Computer Aided Manufacturing) est le plus important projet CIM développé. L'objectif est d'utiliser les moyens informatiques les plus avancés dans le domaine de la production. Le résultat fut l'élaboration d'un modèle de production pour l'industrie aérospatiale. Ce modèle consiste à construire une hiérarchie des activités réalisées par le système avec une finesse identique à chaque niveau. Chaque activité d'un niveau donné est explicitée par un ensemble d'activités d'un niveau inférieur.

Le modèle très répandu, proposé par le NBS (National Bureau of Standards) aujourd'hui appelé NIST (National Institute of Standards and Technology) représente cinq niveaux d'intégration sous forme de pyramide (figure 1.1-1).

**Niveau 5** : implantation multi-site

**Niveau 4** : entreprise entière

**Niveau 3** : optimisation de la production

**Niveau 2** : supervision d'un atelier

**Niveau 1** : commande des machines

**Niveau 0** : machines, robots

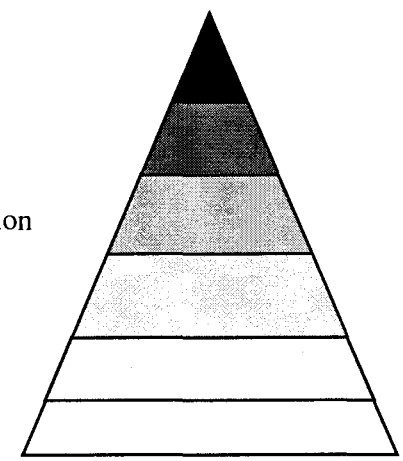


**Figure 1.1-1 : La pyramide CIM**

Ces travaux (ICAM et NIST) s'intéressent aux fonctions exécutives de l'entreprise et définissent des méthodes et des hiérarchies de représentation des fonctions de l'entreprise. Une autre approche met en œuvre une gestion globale basée sur la définition de fonctions stratégiques. C'est le cas notamment de la méthodologie :

- GRAI, développée au laboratoire GRAI de Bordeaux [DOU92] ;
- Purdue, développée à l'université de Purdue (USA) [WIL92] ;

- Et de CIMOSA initié par le consortium AMICE [AMI93] dans le cadre du programme européen ESPRIT (European Strategic Program for Research and Development in Information Technology). Le système CIM est considéré comme un système intégré par ordinateur dans lequel l'ensemble du processus de production est systématiquement informatisé. Dans ces systèmes, la conception, la production, les études, les essais, les réparations et l'assemblage seront assistés par ordinateur au travers d'une base de données commune.

On trouve selon la nature de l'activité et l'expérience des auteurs, des définitions légèrement différentes. Le consensus se fait autour du caractère multidisciplinaire du concept CIM.

Une extension du concept CIM inter-entreprise est traduit par la notion « d'Entreprise Etendue », [BRO95].

Les années 90 sont caractérisées par un environnement économique instable dû aux fluctuations des marchés. La compétition très vive dans ces marchés versatiles pousse les entreprises industrielles à définir de nouvelles stratégies. Les industries recherchent une innovation permanente, dans le but essentiel de réduire les délais de livraison grâce à une réactivité accrue. Ainsi une nouvelle approche voit le jour : l'ingénierie simultanée (Concurrent Engineering).

### 1.1.3. L'ingénierie simultanée

La pression du facteur temps voit la naissance des approches suivantes qui permettent à l'entreprise de se placer en premier (Time-To-Market) sur des marchés très versatiles :

- Le « Just-in-time » [DAM91], par une réduction la surproduction de stock,
- Le « Design-for-manufacturing » [JAG93], en simplifiant la structure des produits.
- L'ingénierie simultanée [KIM92] en réduisant le cycle de vie du produit depuis sa définition jusqu'à sa mise en production.

L'IDA (Institut for Defense Analysis, USA) définit l'ingénierie simultanée comme une *approche systémique qui intègre le développement simultané des produits et des processus associés, incluant la fabrication et la logistique. Cette approche prend en considération dès le démarrage, le cycle de vie du produit depuis sa conception jusqu'à son exploitation en incluant la qualité, les coûts, la planification et les besoins des utilisateurs. »* (IDA Report R-338).

L'organisation séquentielle et linéaire de l'entreprise des années 80 est remise en cause. L'ingénierie simultanée est caractérisée par une prise en compte de toutes les disciplines et un chevauchement des étapes du cycle de vie dès le démarrage du projet (voir figure 1.1-2) [FOU94].

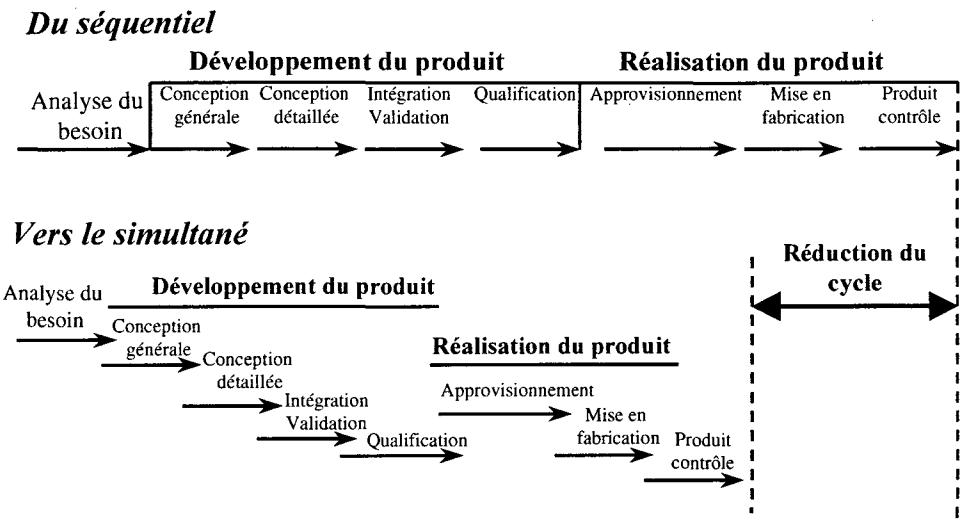



**Figure 1.1-2 : Du séquentiel à l'ingénierie simultanée**

Le succès de l'ingénierie simultanée est mesurable à travers l'impact du programme CALS (Computer aided Acquisition and Logistic Support) initié par le DoD (Departement of Defense). CALS est une stratégie globale d'échange et de gestion de données informatisées entre tous les intervenants d'un même programme militaire.

Les changements périodiques observés depuis les années 50 dans le choix de la stratégie des entreprises se poursuit jusqu'à nos jours. Ce choix est fortement influencé par la performance des outils technologiques disponibles. L'introduction d'une nouvelle stratégie bouscule les structures classiques des entreprises, provoque un changement de l'organisation et une mutation culturelle de l'entreprise.

Le début des années 2000 confirme la mise en place du commerce électronique qui va modifier l'organisation de la production des entreprises, grâce aux Nouvelles Technologies de l'Information et de Communication (NTIC). Nous pouvons citer le « Supply Chain Management », une technique qui permet de récupérer et d'analyser les informations afin d'optimiser en temps réel la production d'une entreprise en fonction de la demande. Le modèle est, bien sûr, le fabricant de micro-ordinateurs DELL, champion incontesté de la vente directe et de la fabrication rapide.

Quel que soit le paradigme adopté dans sa stratégie (Computer Integrated Manufacturing, Just-In-Time, Concurrent Engineering etc. ), une entreprise est toujours confrontée au choix des produits à fabriquer au bon moment, en quantités suffisantes, au moindre coût avec la satisfaction (Qualité, Coût, Délais) du client au final.

Cela passe par une maîtrise des changements fréquents du marché. Les entreprises ne peuvent plus compter sur une demande constante et importante du même produit sur plusieurs années. Une innovation constante est nécessaire pour la survie des produits. La conception de ces derniers est alors sans cesse modifiée.

Les systèmes de production prennent alors toute leur importance dans la stratégie des entreprises.

## 1.2. Architectures des systèmes de production

### 1.2.1. Place des systèmes de production dans l'entreprise

Dans l'environnement concurrentiel et fluctuant de l'entreprise, les critères qui caractérisent la performance d'un système de production sont [IDE94] :

- La productivité, à savoir la quantité de produits par unité de temps, par ressource utilisée ;
- La flexibilité qui caractérise la capacité du système de production à changer de configuration en fonction de la demande ;
- La qualité qui mesure l'aptitude du produit à satisfaire les besoins du client.
- De plus, le système de production doit s'intégrer aux autres systèmes de l'entreprise qui le sollicitent (figure 1.2-1), il doit continuellement s'adapter [COU97], [ROB88].

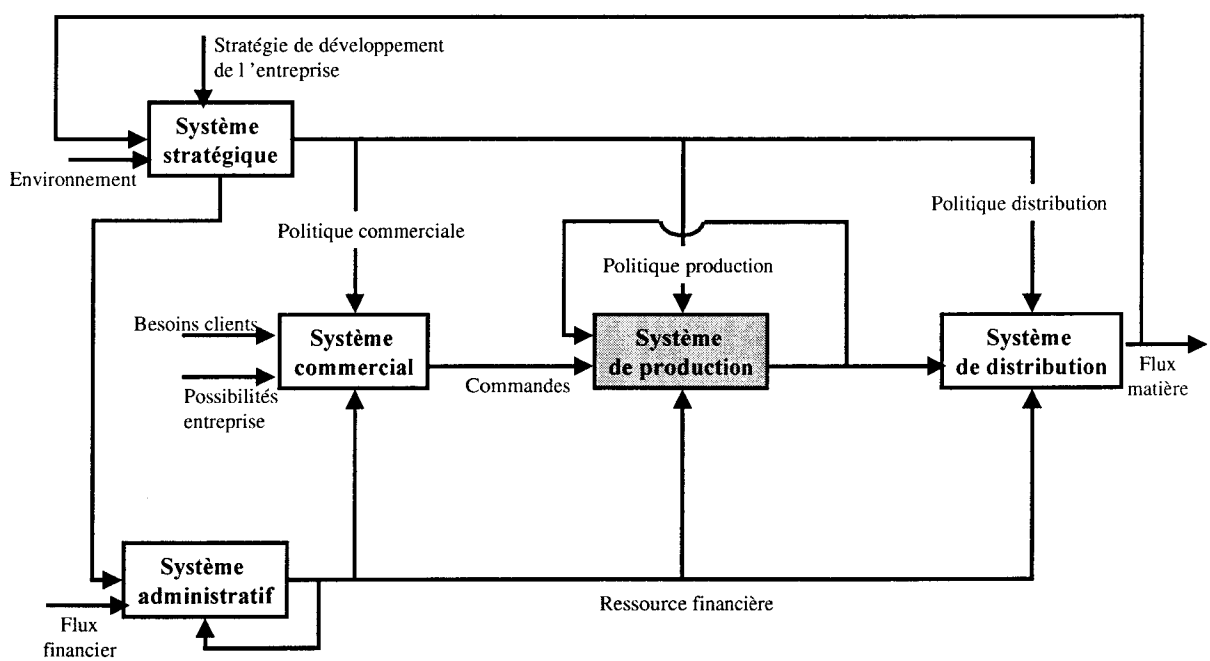



**Figure 1.2-1 : Place du système de production dans l'entreprise**

L'évolution des technologies de traitement de l'information ces vingt dernières années a largement favorisé l'évolution de l'architecture des systèmes de production. Nous allons étudier les architectures des systèmes de production les plus répandues.

Dans nos travaux, nous nous intéressons aux Systèmes Automatisés de Production (SAP). Dès lors que l'on cherche à réduire l'intervention humaine, le système de production est alors qualifié de SAP. C'est alors un système complexe qui utilise les technologies les plus récentes et nécessitent par conséquent la mise en place d'architectures bien définies pour maîtriser le cycle de vie. Le Génie industriel se fixe comme objectif de maîtriser la complexité des SAP [BAR95], [FRA87].

D'une manière générale, le développement d'un SAP suit le cycle de vie proposé par le Centre Coopératif de Génie Automatique [CCG91] sur la figure 1.2-2. Ce schéma décrit les différentes activités concernées par le SAP. Il n'impose aucune séquence d'activités par rapport au processus de conception.

- La partie descendante (branche de gauche) représente les phases de spécification et de conception du SAP,
- La partie basse correspond à la réalisation concrète des activités,
- la partie ascendante (branche de droite) représente une suite d'intégrations et de tests de sous-ensembles de plus en plus complexes,
- et la partie horizontale correspond aux activités de l'exploitation et de maintenance du SAP [COU97].

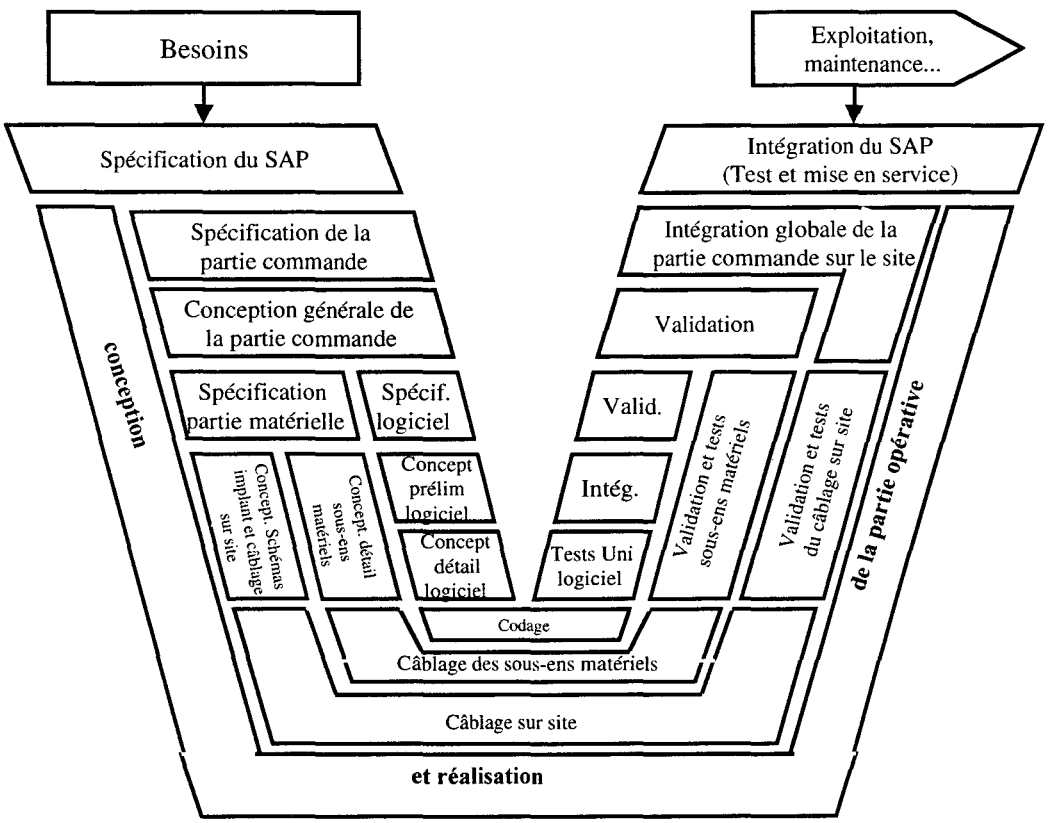



**Figure 1.2-2 : Cartographie des activités d'un SAP [CCG91]**

Sur une échelle de temps plus longue, le cycle de vie d'un SAP est en fait une succession de cycles en V séparés par des périodes de production (figure 1.2-3).

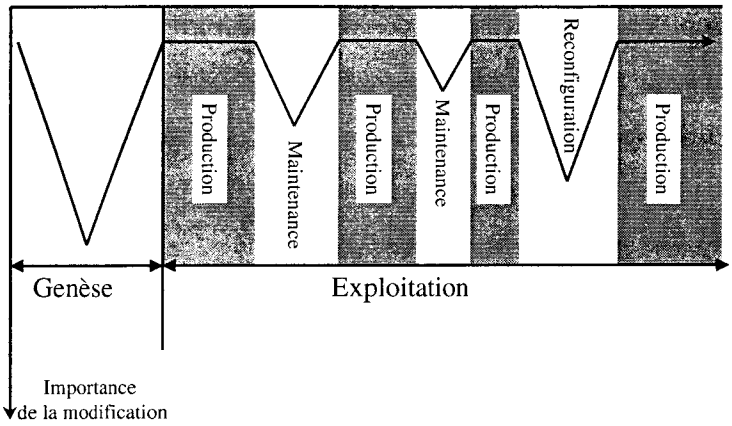



**Figure 1.2-3 : De la genèse à l'exploitation d'un SAP [VER91]**

## 1.2.2. Les systèmes hiérarchisés

L'application de la théorie des systèmes hiérarchisés aux systèmes de production, tant continus que discrets, a pour effet d'agencer leur structure en niveaux de décision successifs formant ainsi une architecture dans laquelle un niveau supérieur définit les contraintes et les objectifs à atteindre par le niveau suivant [LHO99]. Ainsi, le flux de décision est descendant tandis que le flux d'information est ascendant. Parmi les organisations les plus représentatives de ce type d'architecture pour les systèmes de production manufacturière, nous trouvons l'architecture AMRF (Automated Manufacturing Research Facility) développé par le NIST (figure 1.2-4) [SIM82].

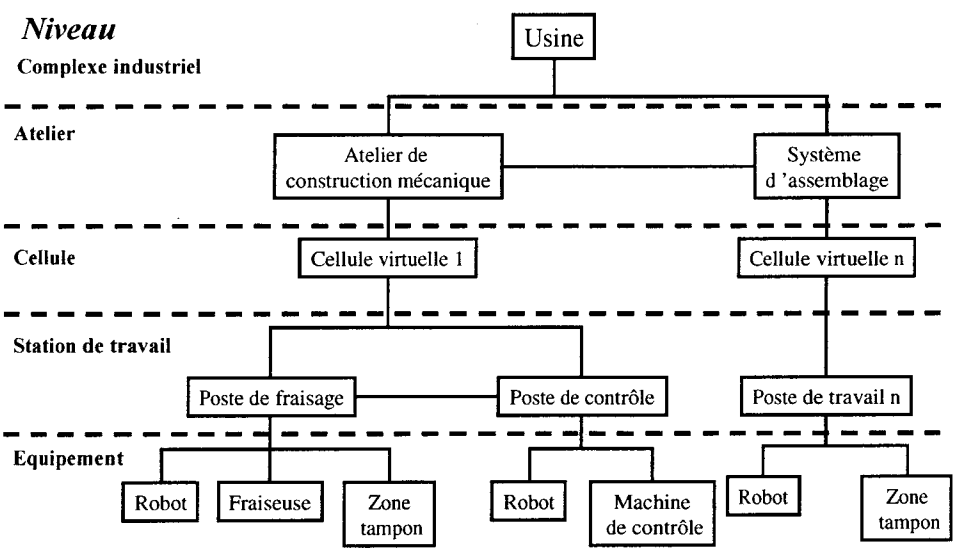



**Figure 1.2-4 : Modèle du NIST pour les ateliers de production manufacturière**

C'est une architecture arborescente sur cinq niveaux : (1) Usine, (2) Atelier, (3) Cellule, (4) Poste de travail, (5) Equipement.

Cette organisation rigide est la principale cause des limites des systèmes hiérarchisés. En effet ils peuvent difficilement évoluer car leur exploitation nécessite une structure fixe ainsi qu'un comportement déterministe des composants. Par exemple, le changement d'un équipement va nécessiter une mise à jour de toutes les données de niveau supérieur [WYN99].

### 1.2.3. Les systèmes intégrés de production

Le concept organisationnel CIM se traduit au niveau des systèmes de production par l'intégration de l'automatisation de production. Un système intégré de production est un ensemble de moyens matériels, logiciels et humains qui intègre, grâce à une architecture informatique globale, tous les éléments qui concourent à la production pour une plus grande synergie au niveau de l'entreprise.

La structure qui résulte de la coordination des fonctions autour d'un système d'information, constitue une pyramide hiérarchique dans laquelle les fonctions d'une entreprise sont coordonnées par des liens de subordination.

Chaque fonction rend ainsi visible son image informationnelle aux fonctions hiérarchiquement supérieures de la pyramide (figure 1.2-5).

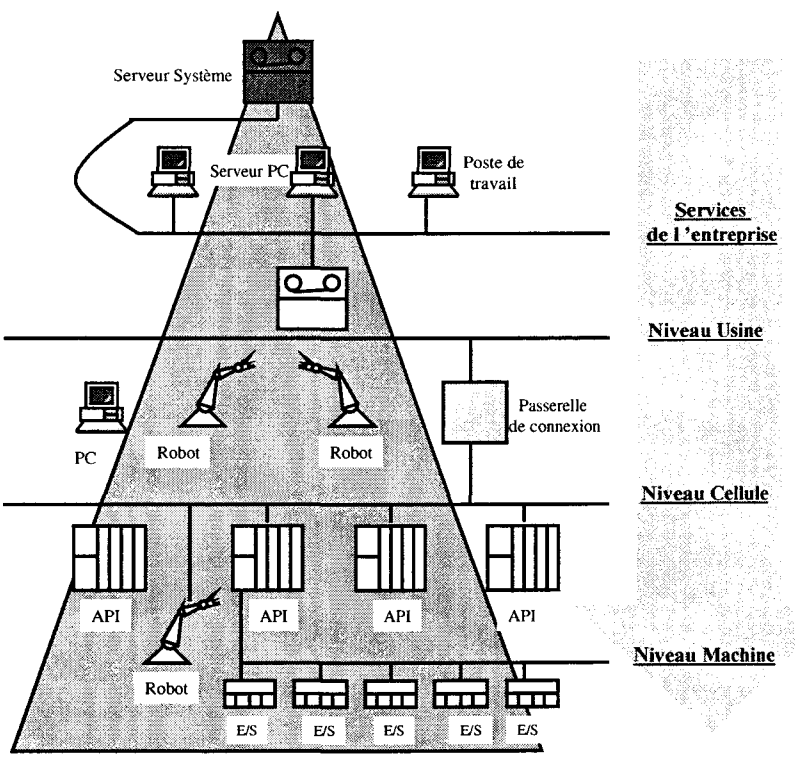



**Figure 1.2-5 : Structure CIM**

Cette évolution des systèmes intégrés de production, encore appelés CIMS (Computer Integrated Manufacturing System) [RAN86], [BAU91], utilise l'information comme vecteur d'intégration de l'ensemble des fonctions d'un système afin d'assurer la disponibilité de la bonne information au bon endroit. Les Systèmes Intégrés de Production permettent de faire communiquer et de coordonner les différentes fonctions des organisations de systèmes hiérarchiques de production, pour former un tout cohérent [MOR92].

### 1.2.4. Les systèmes distribués

Les systèmes distribués de production proposent une structure plate composée d'entités indépendantes appelées agents. Les Systèmes Distribués de Production ont pour objet d'agencer en îlots de production que l'on voudrait coopérants, des organisations de systèmes intégrés de production. Cette évolution, directement liée aux évolutions des technologies de l'information, remet en cause la structure purement pyramidale des systèmes hiérarchisés de production afin d'améliorer la disponibilité de l'information au bon moment et au bon endroit [LHO99]. La structure qui résulte de la distribution de ces îlots d'automatisation autour de multiples médias de communication, constitue une hétérarchie, par exemple, l'hétérarchie WorldFip [HAU97], figure 1.2-5.

La coopération de ces îlots d'automatisation dépend de leur interopérabilité, c'est à dire de la compatibilité de leurs services, de leurs protocoles, de leurs ressources et de leurs performances temporelles.

Cependant, l'indépendance des agents interdit toute gestion globale de l'information. Ainsi la performance globale du système est difficile à optimiser, étant donné que la prévision du comportement des différentes commandes est impossible.

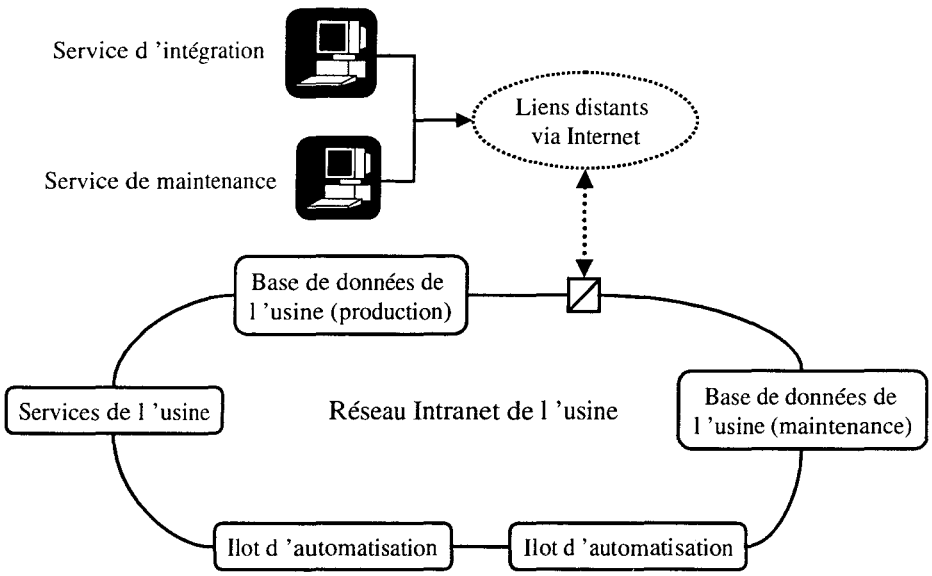



**Figure 1.2-6 : Hétérarchie Worldfip**

Sur l'initiative du programme IMS (Intelligent Manufacturing System), les Systèmes Holoniques de Production (Holonic Manufatcuring System, HMS) proposent un compromis entre une structure hiérarchique (stabilité) et une structure hétérarchique (flexibilité, dynamisme).

### 1.2.5. Les systèmes « intelligents »

A l'instar de tous les nouveaux concepts, les systèmes dits « intelligents » ont pour principal objectif d'avoir une bonne réactivité, afin de s'adapter rapidement aux changements fréquents de l'environnement. Ce concept est une interprétation [VAL94] récente du concept d'holon proposé par Arthur Koestler [KOE89] pour décrire une unité de base d'une organisation biologique ou sociale. Holon vient du grecque *holos* (ensemble), et le suffixe *on* désigne une particule ou partie.

Le consortium HMS définit le concept holon comme étant : *« un élément de base autonome et coopératif d'un système de production pour transformer, transporter, stocker et/ou valider l'information et les objets physiques. Un holon est composé d'une unité de traitement de l'information et souvent d'une unité de traitement physique. Un holon peut appartenir à un autre holon. »* Ces holons coopèrent au sein d'une *holarchie* qui détermine leurs rôles et leur autonomie. Un Système Holonique de Production (figure 1.2-7) est ainsi une holarchie qui prend en compte la totalité des activités de production dès la définition du cahier des charges, pour satisfaire les critères d'adaptabilité requis par la nouvelle génération de systèmes de production (Next Generation Manufacturing System) [KUR96].

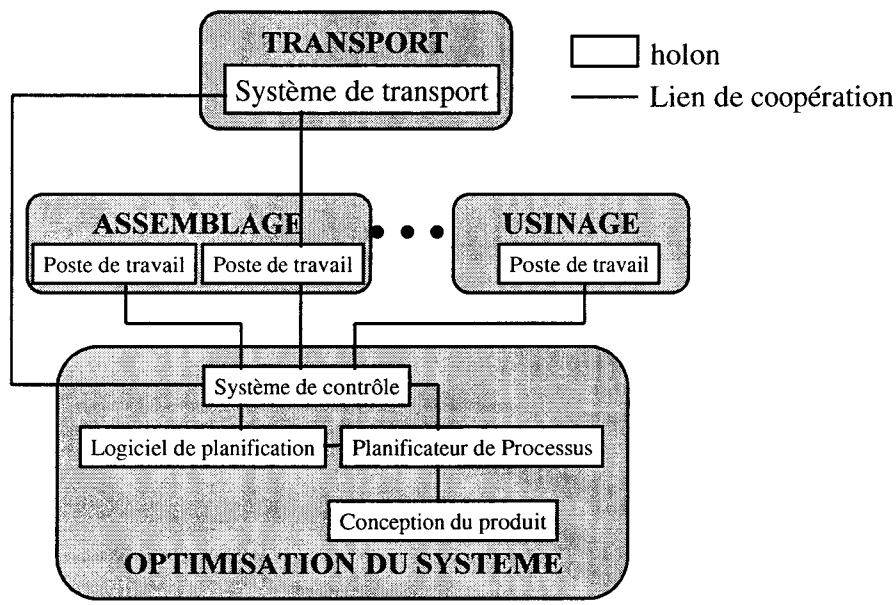



**Figure 1.2-7 : Système holonique d'assemblage de l'université de Louvain [VAL94]**

Les systèmes holoniques de production apparaissent finalement comme un bon compromis recherché entre des organisations intégrées, en tant qu'ensemble cohérent au sein d'un système, et des organisations distribuées, en tant que parties réactives à l'environnement de ce système. *« Ce type d'organisation appliqué à des systèmes intelligents de production n'est utilisé pour l'instant que dans des recherches et développements académiques »* [LHO99].

L'importance des études menées sur les différentes architectures (distribuées, hiérarchisées, holoniques, intégrées) des systèmes de production, montre la place capitale occupée par ces derniers dans la compétitivité des entreprises. La conception, l'exploitation et la maintenance des systèmes de production doivent prendre en compte : l'évolution du système lui-même, l'augmentation de la complexité des produits, la diversité et le caractère évolutif des produits, la gestion des services de plus en plus nombreux offerts par l'entreprise. Ainsi, d'après [BRO94], l'usine du futur devra avoir les particularités suivantes :

- Fonctionner 24h sur 24 pour amortir le coût de l'équipement ;
- Produire par lots de petite taille, pour offrir une grande flexibilité, répondre à l'évolution du produit et au besoin de la clientèle ;
- Avoir des délais d'exécution des différentes opérations du processus très courts ;
- Réduire sa main-d'œuvre sur le lieu de production ;
- Etre de petite taille, avec des machines hautement adaptables et intégrées.

Il devient alors impératif de disposer d'outils et de méthodes permettant de faire une structuration et une analyse cohérente des systèmes complexes étudiés. Dans ce cadre l'équipe PFM (Production Flexible Manufacturière) du LAIL (Laboratoire d'Automatique et d'Informatique Industrielle de Lille) a développé le projet CASPAIM.

## 1.3. Le projet CASPAIM

Depuis 1985, les activités de l'équipe PFM du LAIL sont organisées autour du projet CASPAIM : Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière. L'équipe PFM a développé une méthodologie qui permet d'appréhender les différentes facettes du contrôle/commande d'un Système Flexible de Production Manufacturière.

### 1.3.1. La Flexibilité des Systèmes de Production

Pour adapter rapidement les produits à leurs nouvelles caractéristiques, les systèmes de production doivent disposer de procédés de fabrication souples sans que cela entraîne des frais importants. De même, pour gérer le changement dynamique du cycle de vie des différents produits (figure 1.3-1 [RAN90]), les systèmes flexibles de production doivent être capables de fabriquer plusieurs familles de pièces, en même temps et à un coût moindre que s'ils les fabriquaient séparément, [GOL83].

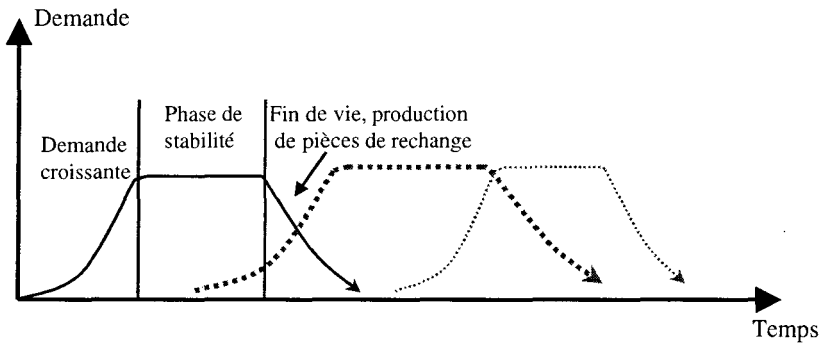



**Figure 1.3-1 : Gestion simultanée de plusieurs cycles de vie**

Il existe trois types de systèmes flexibles d'après [BON85]. (1) Les cellules flexibles qui sont caractérisées par des Machines-Outils à Commande Numérique (MOCN) avec changement automatique d'outils et de dispositifs automatiques pour la manutention et le transport. (2) Les ateliers flexibles se situent dans le prolongement des cellules flexibles (avec une infrastructure plus importante) et assurent la réalisation du processus complet de différents types de produit simultanément. (3) Et enfin, les lignes de transfert flexibles avec un maximum de flexibilité au niveau des lignes d'assemblage à l'instar de l'industrie automobile.

Ainsi les systèmes flexibles de production offrent plusieurs degrés de flexibilité au niveau de la gamme de produits, des machines, de la production, du routage des produits et de leur organisation [BRO85]. Ces flexibilités sont liées entre elles comme le montre la figure 1.3-2 [BRO94].

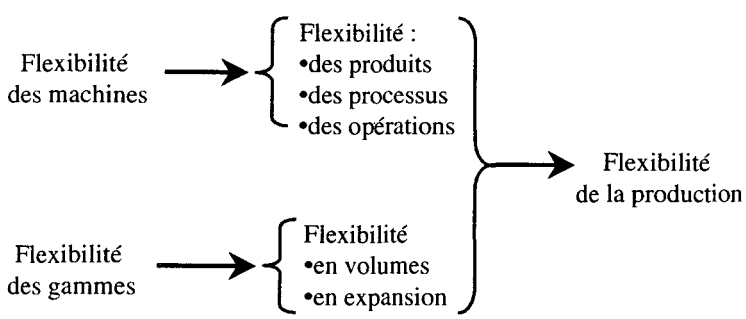



**Figure 1.3-2 : Relations entre les flexibilités**

### 1.3.2. Les systèmes de production selon CASPAIM I

L'objectif initial du projet CASPAIM était d'automatiser et d'assister le processus complet de conception d'un système de commande performant, depuis la définition du cahier des charges jusqu'à l'implantation finale sur site suivant une approche homogène.

Pour cela, tous les aspects relevant des systèmes automatisés de production, d'un point de vue général, ont été pris en considération et en particulier :

- la description des produits : types de pièces, gammes opératoires, ...
- les contraintes et les objectifs de production : délais, débits, temps de cycle, ...
- la définition des stratégies de contrôle et de pilotage du système de production,
- et la représentation fonctionnelle et matérielle des architectures retenues : organes opératifs (machines, robots, ... ), systèmes de transport véhiculant les flux de produits, ressources véhiculant les flux de données.

Pour mieux appréhender la complexité des SFPM (Système Flexible de Production Manufacturière), ils sont décomposés en deux parties (figure 1.3-3) :

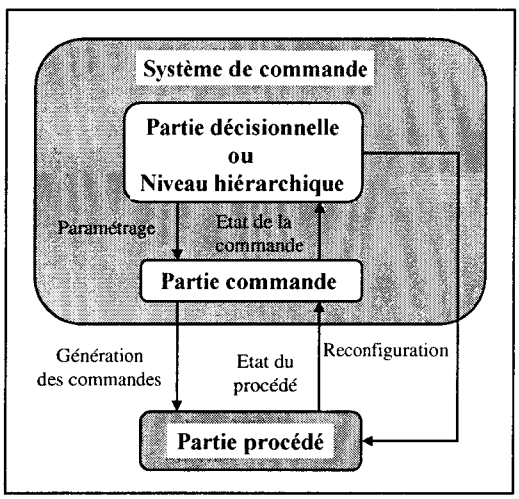



**Figure 1.3-3 : Décomposition d'un SFPM dans CASPAIM I**

- le **système de commande** qui regroupe deux parties :
  - La **partie commande** applique l'ensemble des commandes aux différents équipements de production, pour réaliser les différents produits en fonction de l'état courant du procédé.
  - La **partie décisionnelle** (Niveau hiérarchique) doit résoudre tous les conflits d'accès issus du parallélisme et les indéterminismes issus de la flexibilité.
- La **partie procédé**, constituée de l'ensemble des équipements de production ainsi que des produits en cours de fabrication.

### 1.3.3. Démarche de conception dans CASPAIM I

Le processus de conception comporte quatre étapes (figure 1.3-4) de validation afin de s'affranchir de sa complexité due à la flexibilité des SFPM, à l'existence de plusieurs procédés de fabrication et à la complexité des produits.

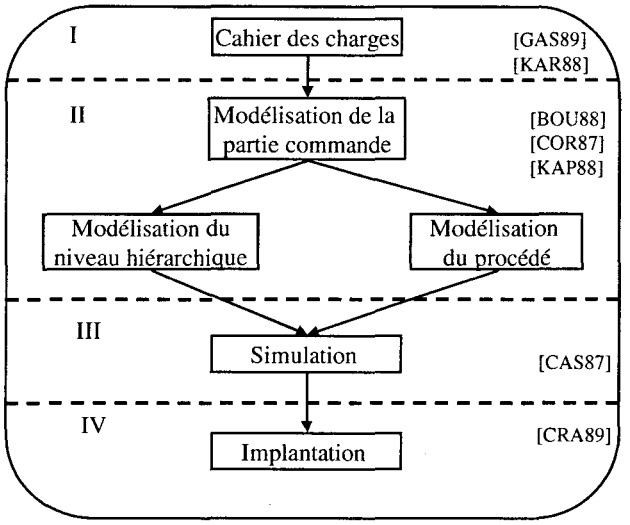



**Figure 1.3-4 : Processus de conception CASPAIM I**

- Cahier des charges [GAS89], [KAR88]

  Cette première phase de spécification permet de travailler tout au long du projet sur des données cohérentes en fournissant une base de règles pour spécifier les gammes opératoires en langage semi-formel. Ensuite une représentation formelle des gammes opératoires est effectuée à l'aide de règles de production.

- Modélisation de la partie commande [BOU88], [COR87], [KAP88]

  A partir des règles de production, un prégraphe (Réseaux de Petri, RdP) est généré. Il constitue l'architecture de base du système à partir duquel un modèle de commande sera développé.

- Validation par simulation

  La simulation (RdP, règles de production) valide le graphe commande obtenu dans la phase précédente, en évaluant le comportement du système.

- Implantation

  Enfin l'implantation transcrit les modèles de conception en modèle implantable (Grafcet) et définit les critères de décomposition permettant d'obtenir une architecture de contrôle commande optimale.

Au début des années 90, le projet CASPAIM a évolué pour faire face au développement des moyens de production de plus en plus complexes (Stockeurs Rotatifs, Robots Portiques ), et à l'introduction de techniques de gestion des produits de plus en plus sophistiquées (gestion par lots, notion de dates dues).

De plus, de nouveaux axes de recherche sont abordés : la surveillance et l'évaluation de performances. Ainsi, des concepts supplémentaires de structuration et une nouvelle approche intitulée CASPAIM II est donc définie pour appréhender des facteurs de complexité bien plus importants.

### 1.3.4. Organisation du contrôle/commande selon CASPAIM II

Cette nouvelle démarche CASPAIM II repose sur un nouveau modèle logique [BOU93] et va mettre en œuvre de nouveaux concepts de modélisation, utiliser de nouveaux formalismes et prendre en compte de nouveaux aspects.

Le contrôle/commande est divisé en trois fonctions essentielles : la planification / ordonnancement, la supervision et la commande (figure 1.3-5).

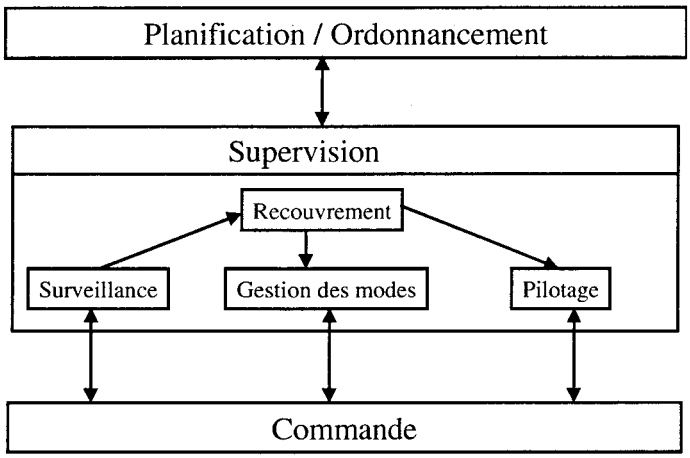



**Figure 1.3-5 : Organisation du contrôle / commande selon CASPAIM II**

- La planification / ordonnancement élabore le planning prévisionnel des commandes de fabrication et de maintenance, suivant la disponibilité des ressources du système de production. L'étape d'ordonnancement permet de fixer les séquences et le nombre d'opérations sur les machines ainsi que les dates de début de chaque tâche.
- La supervision garantit les objectifs de production sur la base des données de l'ordonnancement et des consignes des opérateurs, par ses fonctions de Pilotage, Gestion des modes, Surveillance et Recouvrement.
  - La surveillance de la commande [ELK93] qui permet une détection et un diagnostic immédiat à l'aide d'un langage synchrone.
  - La surveillance du procédé [TOG92] effectue une analyse temporelle du procédé pour la détection et fait appel à l'intelligence artificielle.

- La gestion des modes décrit les modes de fonctionnement du SFPM en spécifiant les contraintes de fonctionnement entre ressources [BOI91] ou entre fonctionnalité des ressources [KER96].

➢ La partie commande a pour fonction d'appliquer l'ensemble des commandes aux différents équipements du procédé en assurant leur séquencement et leur coopération. Elle correspond à la modélisation d'un système de coordination mettant en jeu les opérateurs de la partie procédé en vue d'une production spécifique.

### 1.3.5. Démarche de conception de la commande

Ce qui caractérise la démarche de CASPAIM II (figure 1.3-6), c'est la séparation au niveau de la commande de ce qui relève du produit et de ce qui relève des moyens de production.

➢ La **spécification** s'effectue en plusieurs phases :

- **Spécification de la partie logique** [CRU91] **:**

Le produit à fabriquer est décrit sans tenir compte de la nature des équipements à utiliser, à l'aide d'une gamme logique. Cette dernière représente le séquencement des différentes fonctions de transformation (usinage, assemblage) que doit subir le produit brut jusqu'à son état fini.

- **Spécification de la partie physique** [AMA94] **:**

Les différentes ressources de production sont décrites ainsi que les différentes contraintes (agencement, accessibilité...) qui les lient.

- **Spécification de la partie décisionnelle** [TAW95] **:**

Les stratégies de pilotage du système sont décrites sous forme de règles décisionnelles qui permettront de résoudre les conflits et indéterminismes d'affectation.

➢ **Phase d'analyse et de conception**

Les gammes opératoires sont générées automatiquement à partir des gammes logiques en tenant compte des ressources de transformation et de l'aspect transitique de la partie physique [AMA94]. Ces gammes constituent un modèle pour la partie coordination de la commande. Pour chaque ressource, un graphe de commande est élaboré. Ce graphe représente une séquence d'actions élémentaires de commande à partir d'une bibliothèque de graphes partiels [HUV94].

➢ **Phase de validation**

La validation repose sur deux étapes :

- une validation statique des gammes, basée sur l'analyse du modèle RdP. Elle recherche les blocages liés à une saturation du système de transport [AUS94] ou à une utilisation croisée des ressources [CRU91].

- une validation dynamique du système par simulation permet une analyse globale du modèle [OHL95].

➢ **Phase d'implantation** [HUV94], [FAR93] **:** L'implantation de la commande est faite sur une transposition des modèles en langage Ada.

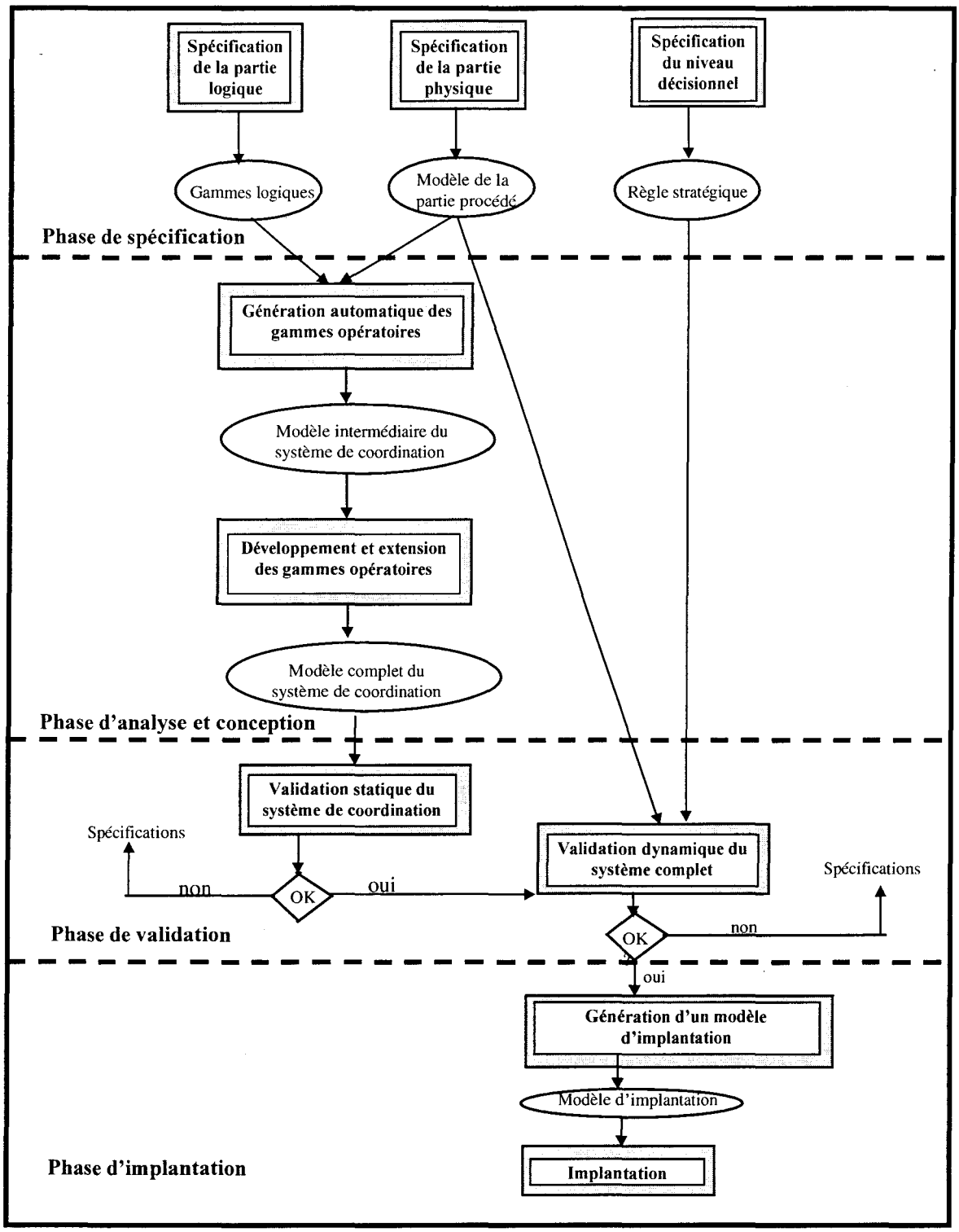



**Figure 1.3-6 :Démarche de conception dans CASPAIM II** [AMA94]

### 1.3.6. Etat actuel des travaux dans CASPAIM

De récents travaux (entre 97 et 99) ont permis de prendre en compte de nouvelles fonctions à savoir :

- la commande prévisionnelle [CAM97], [KOR98], cette fonction permet de déterminer, à partir des gammes opératoires, une commande cyclique déterministe à flux de production maximal.
- l'analyse de la tolérance et de reconfiguration [BER98], fait partie de la supervision et joue un rôle décisionnel en déterminant comment le système peut conserver un caractère opérationnel malgré la présence d'une panne.
- la surveillance prédictive indirecte [LY99], basée sur la surveillance du flux des produits fabriqués. Cette fonction propose aussi des procédures de correction et de maintenance connaissant l'impact réel d'une défaillance sur la production.
- Les travaux sur le gestion des modes de marche sont poursuivis par N. DANGOUMAU, suivant une approche de modélisation distribuée et hiérarchique [DANOO].

L'ensemble des fonctions, étudiées dans CASPAIM II, du contrôle / commande d'un SFPM, et les échanges entre ces fonctions, sont schématisées sur la figure 1.3-7, [LY99].

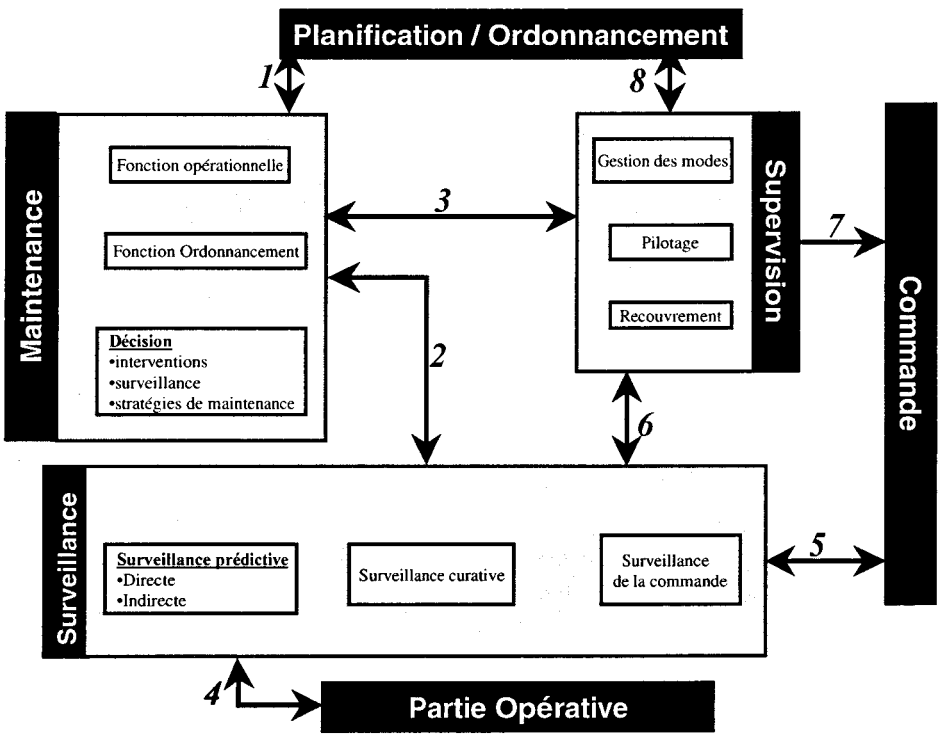



**Figure 1.3-7 : Les différentes fonctions du contrôle-commande d'un SFPM**

1) Les informations échangées entre les fonctions Maintenance et Planification/Ordonnancement concernent la disponibilité des ressources pour la réalisation du plan de production et le calcul des ratios.
2) Entre la surveillance et la maintenance, les défaillances sont localisées et l'état de dégradation des ressources surveillées est transmis afin d'établir une stratégie de maintenance.
3) Sur la base d'informations de la maintenance, sur le flux de produits et des défaillances, la supervision met en place des procédures de reconfiguration et de rattrapage des dérives de flux.
4) A partir de l'observation du procédé effectuée par la partie opérative, une commande saine filtrée est envoyée par la surveillance.
5) La commande reçoit (en retour d'une commande brute) les comptes-rendus filtrés.
6) Selon les résultats de la surveillance (prédictive, curative et de la commande), la supervision peut demander un gel de la commande.
7) La supervision informe la commande sur les changements de mode de fonctionnement.
8) En fonction des ressources utilisées et des ratios de production, la supervision peut demander un réordonnancement ou une autre production.

L'équipe PFM du LAIL, à travers le projet CASPAIM vise à modéliser, concevoir et exploiter la commande d'un système flexible de production manufacturière (SFPM). Ces travaux font intervenir plusieurs concepteurs qui abordent les différents aspects des SFPM avec des points de vue divers et des modèles variés. Les SFPM sont abordés de façon modulaire dans le but de s'affranchir d'une partie de leur complexité. Ces modules conçus de manière autonome doivent cependant interagir avec les autres. L'information devient alors le principal acteur de ces échanges.

Sans système d'information intégrant, le système étudié apparaît alors davantage comme une addition de solutions conceptuelles, technologiques et informatiques, définies indépendamment les unes des autres. Pour préserver la cohérence globale d'une telle organisation et garantir une performance optimale, nous avons décidé d'orienter nos travaux sur la définition et la réalisation d'une plate-forme d'intégration pour la conception assistée des systèmes automatisés de production que nous appellerons dans ce mémoire le référentiel.

## 1.4. Un référentiel pour gérer l'information

### 1.4.1. Système d'information d'un système de production

L'étude du système d'information se place dans la recherche d'une amélioration globale et durable des performances de l'entreprise. D'après [ROL86], *« un système d'information est un artefact, un objet artificiel, greffé sur un objet naturel qui peut être une organisation. Il est conçu pour mémoriser un ensemble d'images de l'objet réel à différents moments de sa vie ; ces images doivent être accessibles par les partenaires de l'organisation qui s'en servent pour décider des actions à entreprendre dans les meilleures conditions »* Le système d'information (S.I.) est le support de la performance de l'entreprise, traduction de sa stratégie et de son organisation.

Dans une entreprise, toute unité de production est considérée comme système à part entière [LES94], et possède par conséquent son propre S.I.

L'analyse systémique [LEM90] consiste à définir les limites du système à étudier, à identifier les éléments importants et les types d'interaction entre ces éléments, puis à déterminer les liaisons qui les intègrent en un tout organisé. En suivant cette approche, un système de production est composé de trois systèmes indiqués sur la figure 1.4-1.

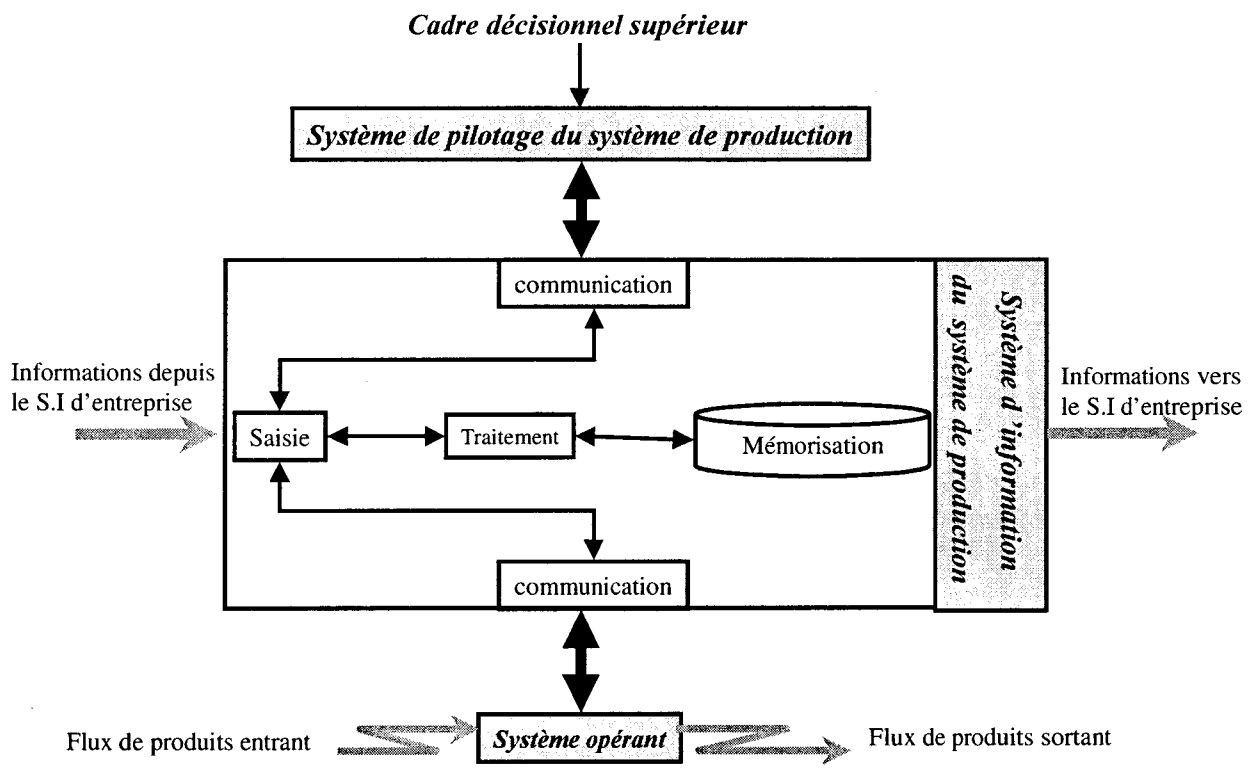



**Figure 1.4-1 : Place du S.I. d'un système de production en entreprise**

1) Le **système opérant** contient les fonctions de transformation des flux relatifs aux produits et services, spécifiques de l'activité de l'entreprise.
2) Le **système de pilotage,** où figurent les fonctions de décisions prises selon les objectifs assignés au système opérant de suivi des ressources ; ainsi que la définition des règles de transformation et d'adaptation à l'environnement ;
3) Le **système d'information** représente le système de couplage entre le système opérant et le système de pilotage.

Pour assurer ce rôle de couplage, le système de production dispose d'un S.I. composé de plusieurs fonctions de saisie, traitement, mémorisation, et communique avec les autres fonctions de l'entreprise [COU93].

- La saisie constitue la première étape de toute activité mettant en œuvre un transfert d'information. Cette fonction juge de l'utilité des données à saisir, décide de leur codification et teste leur fiabilité ainsi que leur validité.
- La mémorisation « range » les données à un endroit qui permet de les retrouver afin de les exploiter. Une modélisation et une structuration des données sont nécessaires pour que l'accès devienne pertinent et efficace.
- Le traitement permet à l'utilisateur d'accéder, de mettre en forme et de manipuler les données.
- La communication se situe tant au niveau de l'échange au sein du système d'information, que dans l'interface entre ce même système d'information et ses sources ou ses utilisateurs.

Ainsi, le système d'information fait office de partie intégrante du système de production, et constitue donc le support de l'amélioration globale et durable des performances du système de production.

### 1.4.2. Le flux d'information dans les systèmes de production

Le système de production d'une entreprise manufacturière est constitué d'un ensemble de processeurs de transformation, de transfert, et de stockage. Il est aussi doté d'un ensemble de processeurs de traitement et de communication d'information.

L'information est un flux qui circule dans un système de production au même titre que les matières (matières premières, produits intermédiaires et finis, moyens physiques et humains) [COU93]. C'est un flux dont le but est d'avoir une connaissance précise des autres flux en vue de prendre des décisions [BOU94].

Le flux d'information permet ainsi de réguler le flux de matières. Le groupe de travail BERMUDES [TCH97] recense les différents éléments de flux d'information dans un système de production sur la figure 1.4-2.

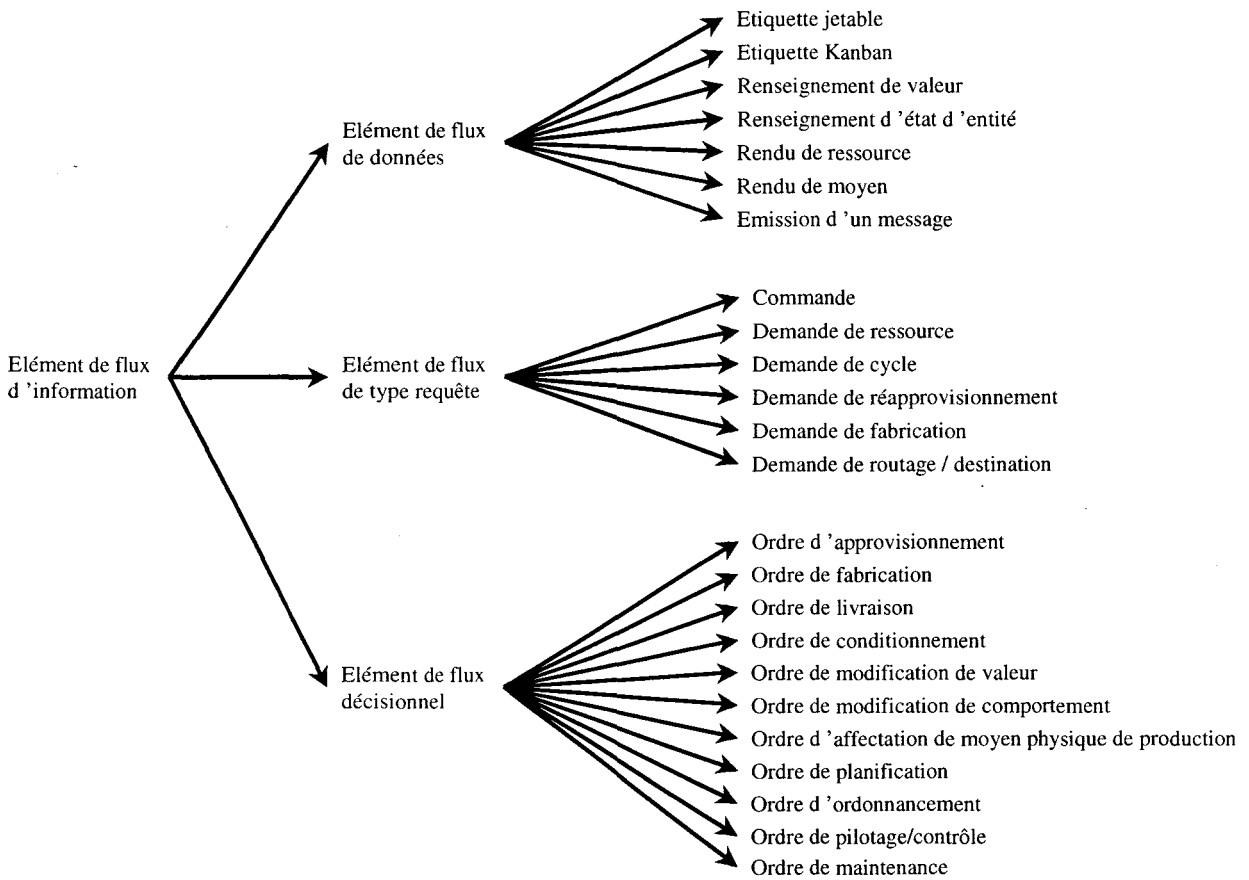



**Figure 1.4-2 : Eléments de flux d'information d'un système de production**

Assurer l'échange d'informations entre les différents éléments (Opérateurs, Moyens, Produits) d'un système de production est un challenge qui génère une densité du flux d'information et nécessite la mise en place d'un système d'information.

L'information est une ressource qui doit assurer une efficience maximum face à des volumes d'information élevés et des synchronisations impliquant des temps de réponse faibles. La capacité d'un système d'information à fournir les bonnes informations au bon moment sert ainsi de support au fonctionnement normal du système de production.

### 1.4.3. Les difficultés liées à la conception d'un projet en génie industriel

Un projet en génie industriel est caractérisé par la forte hétérogénéité des disciplines, outils, méthodes, modèles. Ajouter à cela le cycle de développement non linéaire et le caractère évolutif des SAP, la communication entre les experts du projet devient hautement difficile.

Une équipe pluridisciplinaire intervient, et chaque acteur-concepteur s'intéresse à un ou plusieurs aspects particuliers du SAP et utilise ses propres outils, méthodes, modèles dédiés. Chacun a alors une interprétation propre de chaque donnée échangée.

D'une manière générale, tout projet en génie industriel aborde des aspects bien définis du SAP et fait appel à des connaissances très variées. Nous montrons la diversité de l'ensemble des modèles et outils utilisés dans le projet CASPAIM, ainsi que les différentes fonctions abordées, sur la figure 1.4-3.

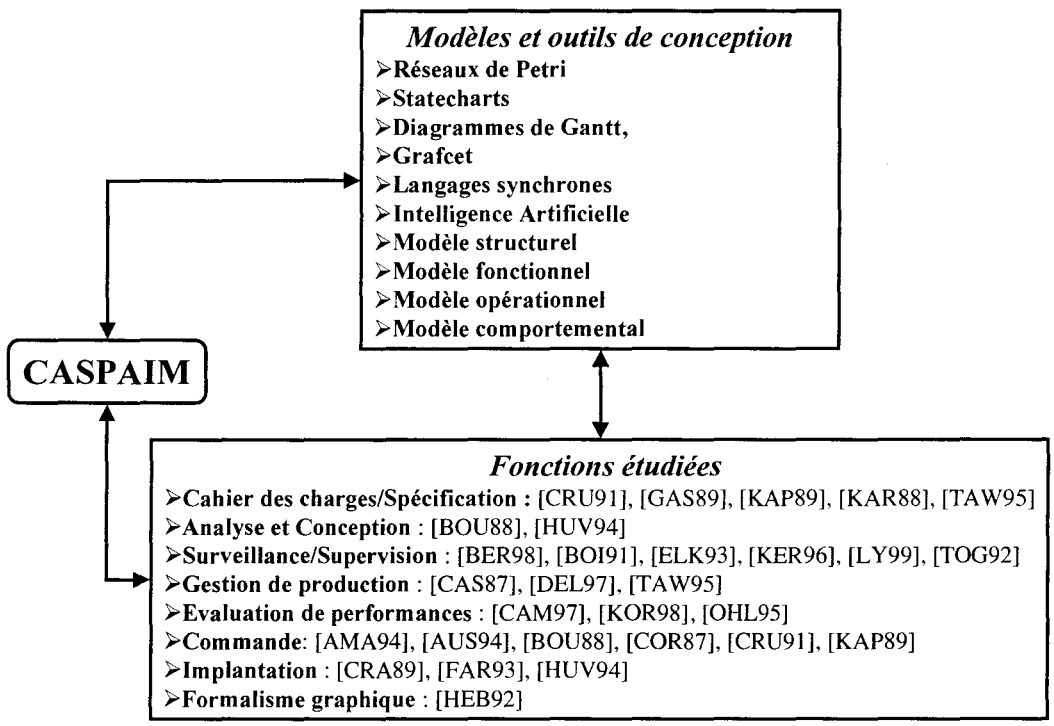



**Figure 1.4-3 : Hétérogénéité des composants du projet CASPAIM**

On définit généralement une tâche de conception ainsi : *« Etant donné un ensemble de composants, un ensemble de relations entre ces composants, une tâche de conception consiste à spécifier comment réaliser un objet à l'aide de ces composants de manière à satisfaire un ensemble d'exigences. Bien que dans certains types d'activités l'ensemble des composants ne soit pas toujours fini et que de nouveaux composants doivent être conçus, cela ne pose théoriquement pas de problème particulier car la conception peut être récursive* » [GUE92].

Un problème de conception est un problème ouvert qui n'admet ni formulation, ni solution définitive. En effet, on ne peut jamais, ni exprimer intégralement le besoin que l'objet à concevoir a pour objectif de satisfaire, ni établir une liste exhaustive des moyens qui peuvent servir à construire un modèle de cet objet. Ainsi, le processus de conception est un processus au cours duquel le problème de conception est formulé et résolu plusieurs fois.

La résolution d'un problème de conception dépend de la disponibilité de certaines connaissances en fonction desquelles on doit pouvoir choisir les méthodes de résolution. La solution d'un problème de conception peut être considérée comme étant la spécification complète d'un ensemble de composants et de leurs relations, le tout décrivant un objet satisfaisant des contraintes et remplissant des fonctions [CHA90], [HAR97].
Les travaux de [HAR97] ont permis d'énumérer six problèmes majeurs auxquels les concepteurs se trouvent le plus fréquemment confrontés :

**(1) L'incomplétude des données en conception**

Quand le processus de résolution commence, il est rare que le concepteur dispose de données précises et complètes. Celles-ci vont évoluer tout le long du processus de conception afin de réduire les incertitudes et rejoindre les spécifications initiales du problème de conception. Dans ce cas, la solution finale sera un compromis entre ce qui a été spécifié et ce qui a été obtenu après exploration de l'espace des solutions possibles.

**(2) La disponibilité du savoir-faire de l'expert**

S'il peut être facile de comprendre une solution élaborée par un concepteur, il reste difficile de savoir comment il y est arrivé. Il est donc nécessaire de conserver une trace de la conception pour recueillir les différents raisonnements et choix qui ont été effectués ainsi que l'ordre dans lequel ils ont été réalisés. De plus, chaque concepteur possède sa façon propre de raisonner influencée par sa formation, son environnement et sa propre sensibilité. Ainsi, pour un même cahier des charges et donc pour une même définition du produit, plusieurs solutions, sensiblement différentes, peuvent être proposées par différents concepteurs.

**(3) La réutilisation de 1'existant**

D'après [MIN85] : *« Pour résoudre un nouveau problème, il faut d'abord essayer des méthodes similaires à celles qui avaient bien fonctionné dans des problèmes similaires »*. Lorsqu'un concepteur élabore une nouvelle gamme de produits, il utilise en moyenne 80% de solutions déjà existantes. Une gamme de produits évolue régulièrement mais toujours sur une même base de départ. Il faut donc pouvoir gérer un volume important d'informations en évolution permanente. Néanmoins, la conception de produits entièrement nouveaux est une situation qui n'est pas rare.

**(4) La mise en relation de plusieurs expertises**

La conception d'un produit fait, dans la plupart des cas, appel à divers domaines d'expertise bien souvent détenus par divers experts. La cohabitation entre ces différentes expertises va complexifier la résolution du problème mais elle est pourtant importante pour garantir la réalisation d'un produit de qualité. Pour cela, il faut tout d'abord identifier ces expertises puis établir des lois de négociation et de coopération entre les experts, afin de permettre à ces derniers d'accéder à la bonne information au bon moment.

**(5) Les points de vue multiples**

Les composants d'un produit peuvent être conçus simultanément mais séparément par différents concepteurs dans plusieurs bureaux d'études : on parle alors de co-conception. Par ailleurs plusieurs concepteurs peuvent avoir des perceptions différentes d'un même composant ou d'un même produit [TIC95]. Pour arriver à obtenir un ensemble cohérent, il faut considérer tous les aspects provenant des concepteurs, chacun ayant son point de vue sur les contraintes à prendre en compte. Ceci entraîne des conflits qu'il faut pouvoir gérer.

**(6) La gestion des configurations**

Les aspects les plus importants concernant les données de CAO sont leur évolutivité, leur multi-représentations et leur structure complexe. En effet, le caractère itératif des activités de conception et les multiples tentatives entreprises par les concepteurs font en sorte que les environnements de CAO génèrent une multitude de données. Ces données doivent être maintenues afin de constituer un historique de la conception et leur organisation doit prendre en charge leurs différentes représentations sous forme de hiérarchies et d'agrégations.

Ainsi, un objet de conception est considéré comme une agrégation de données de conception traitée comme une unité cohérente par les concepteurs. A travers le temps, ces représentations sémantiquement significatives de l'objet ou du produit forment des versions [KAT90]. Chaque version peut être un descendant de versions existantes (si cette version n'est pas l'unique ou la première) comme elle peut servir d'ancêtre à de futures versions.

Toutes ces difficultés obligent à disposer dans tout projet de conception d'un outil de gestion rigoureuse de la totalité de l'information échangée entre les différents intervenants. C'est l'objectif que l'on s'est fixé à travers l'étude d'un référentiel de conception.

### 1.4.4. Hypothèses de travail

Nous avons identifié **quatre ensembles** d'éléments qui interviennent dans un projet de conception d'un SAP :

- L'objet de l'étude, à savoir le **SAP** (ou une partie du SAP) et ses éléments : ses composants physiques, nous entendons par-là tout objet ou groupe d'objets réels du SAP (robot, centre d'usinage, convoyeur, cellule etc.) ;
- Les **Activités** constituées par l'ensemble des fonctions du SAP à étudier. Il est possible de regrouper les fonctions qui concourent à réaliser le même objectif au sein de modules.
- L'ensemble des **Acteurs** constitué des différents experts (que nous appelons aussi intervenant ou acteur-concepteur) ;
- Enfin l'ensemble des **moyens** comprenant les méthodes, modèles, et outils XAO.

Les éléments de ces divers ensembles interagissent tout au long du processus de conception, nous avons ainsi défini les relations entre ces éléments sur la figure 1.4-4.

(1) **Relations entre les activités et les intervenants** :

Le choix des fonctions à étudier et de leur organisation dépend uniquement des concepteurs du projet. Un expert peut intervenir dans l'élaboration de plusieurs fonctions. De même, une fonction peut être conjointement élaborée par plusieurs experts.

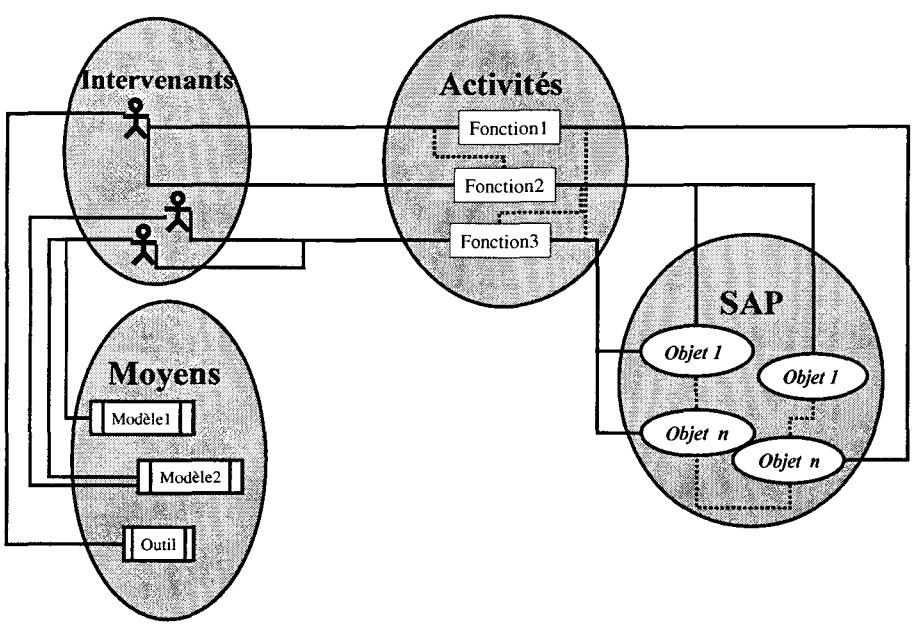



**Figure 1.4-4 : Les constituants d'un projet**

(2) **Relations entre les intervenants et les moyens** :

Il existe un certain nombre d'outils et de modèles standards (Réseaux de Petri, Grafcet etc.) que les intervenants peuvent utiliser pour expertiser leurs domaines d'études, cependant il arrive fréquemment qu'ils développent des extensions de ces outils ou tout simplement qu'ils créent leurs propres outils pour des cas très particuliers. Un modèle ou outil peut servir à plusieurs intervenants.

(3) **Relations entre les fonctions et les objets du SAP** :

Les fonctions définies pour l'étude d'aspects particuliers du SAP peuvent se regrouper en modules. Par exemple le module supervision est composé des fonctions pilotage, recouvrement et gestion des modes (voir figure 1.3-7, page 22). Chaque fonction ou module peut s'appliquer à plusieurs composants physiques (robot, machine, convoyeur, cellule etc.) du SAP. De même chaque composant du SAP peut être concerné par plusieurs fonctions ou modules. Les éléments physiques du SAP possèdent ainsi plusieurs facettes. Les informations, qu'ils fournissent, ne sont pas perçues de la même manière d'une fonction à une autre. C'est le cas par exemple d'un élément étudié sur le plan structurel, comportemental, fonctionnel ou autre.

*« Tout constituant d'un SAP peut être considéré comme un ensemble unifié de vues. Il est ainsi défini selon des points de vue différents. Ces points de vue sont alors dépendants de la perception des différents intervenants pour lequel l'objet a une utilité. Ainsi, l'objet physique peut apparaître comme une composition de plusieurs vues techniques issues de domaines technologiques différents, à l'image d'un objet observé sous plusieurs angles »,* [BOI98], [RIE90].

**Ce qui nous amène aux postulats suivants :**

- Chaque fonction possède ses propres moyens d'accès aux objets du SAP, à partir desquels elle extrait les informations qui lui sont pertinentes et constitue un seul Point de Vue.
- Un acteur-concepteur intervient sur un à plusieurs points de vue.
- Un point de vue donné peut faire intervenir un à plusieurs experts.

Partant de ces hypothèses, nous allons maintenant définir un cahier des charges pour le référentiel.

### 1.4.5. Cahier des charges du référentiel

La principale difficulté est d'obtenir une perception globale cohérente des données échangées du fait du recouvrement sémantique et de leurs liens syntaxiques. Cette difficulté provient de l'hétérogénéité des éléments (acteurs, moyens, activités, objets du SAP) de ce projet. La conséquence au niveau des données manipulées, est qu'elles sont perçues par les fonctions sous divers angles, donc différemment d'un point de vue à un autre.

Le cahier des charges établi dans [BIG96] met alors en évidence, la nécessité pour le référentiel de gérer les points de vue : il s'agit d'offrir la possibilité de décrire le même système selon des points de vue différents.

*« L'information »* est ici un objet conceptuel qui permet de définir chaque objet dans une syntaxe et une sémantique connue des différents points de vue.

La mise en place d'un référentiel ne doit pas modifier la structure du projet. Ainsi, il n'intervient pas d'une part sur le découpage, la définition et l'organisation des activités, d'autre part sur leur séquencement dans les différentes phases du projet.

Le référentiel propose un modèle collectif cohérent sur les divers composants (Objets du SAP, Intervenants, fonctions, Modèles), du projet. Les différentes fonctions d'un projet peuvent ainsi échanger et partager les informations pertinentes et cohérentes (données, traitements, documents, ressources, etc.) relatives à un produit (procédé, pièce, etc.) et au processus de conception durant son cycle de vie.

Chacun des intervenants doit pouvoir accéder aux informations qui le concernent, sans être noyé dans un ensemble de données parasites qui nuiraient à la lisibilité de son domaine. Des autorisations d'accès (en création, lecture, mise à jour et suppression) sont donc allouées à chaque point de vue. Ainsi, le référentiel se comporte comme un filtre entre l'ensemble des informations relatives au système et les intervenants, ces derniers pouvant disposer d'un ou plusieurs points de vue (figure 1.4-5).

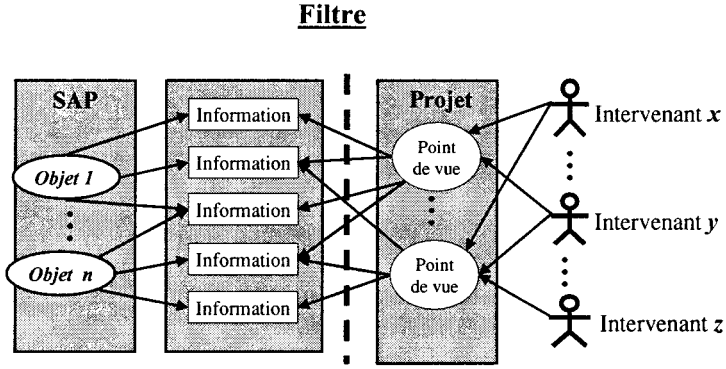



**Figure 1.4-5 : Gestion des points de vues**

Le référentiel doit gérer les différentes configurations, tout en gardant une trace de l'historique, [BIG97]. Il est ainsi possible d'effectuer un véritable traçage de l'information, pour répondre à des questions telles que « quel utilisateur a modifié telle information ? », ou « sur quelle machine-outil a été usiné tel alésage ? ».

**Le référentiel dispose ainsi de trois fonctions (figure 1.4-6) :**

- Une première définit un **modèle conceptuel du SAP** étudié. Ce modèle sert de base à l'intégration des points de vue.
- Une deuxième fonction pour **intégrer les moyens** (modèles et outils) de conception. Cette fonction étudie les concepts utilisés dans les divers points de vue afin de détecter leurs intersections.
- Et une dernière pour la **gestion des points de vues**. Cette fonction gère les droits d'accès et de modifications des divers points de vue sur chaque donnée.

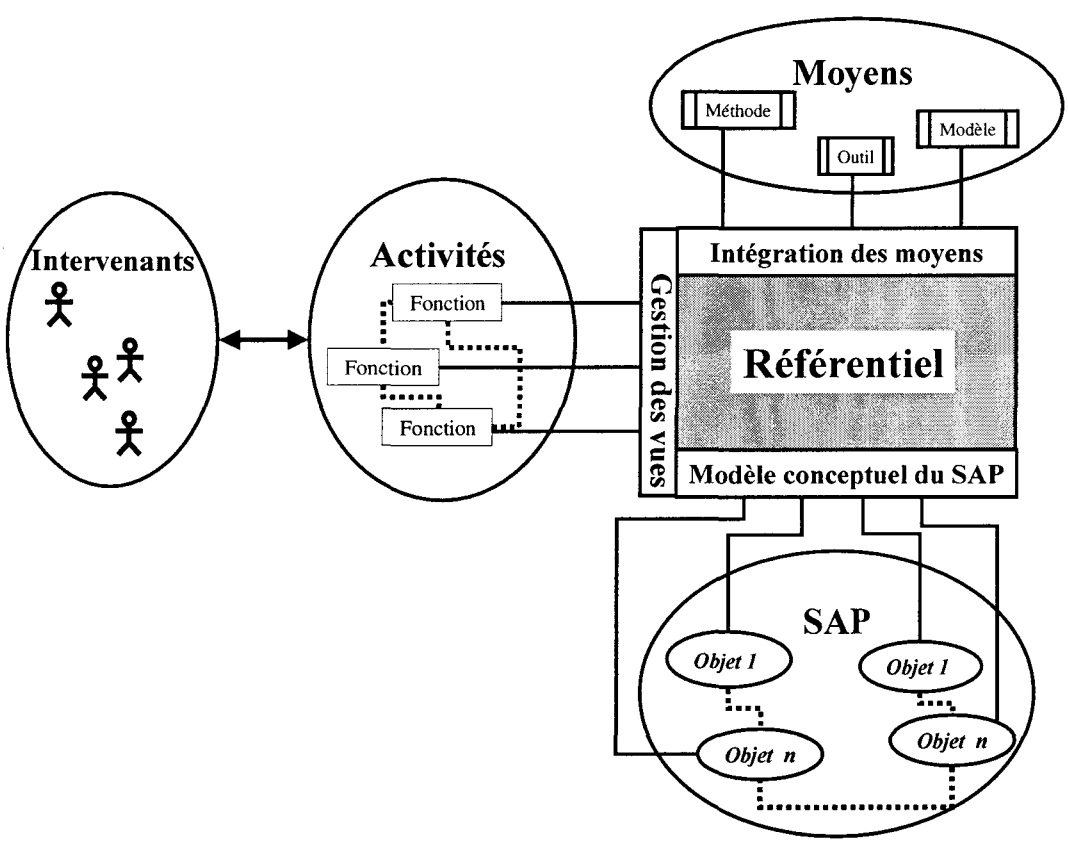



**Figure 1.4-6 : Les fonctions du référentiel**

Le travail de conception du référentiel dans ce mémoire, relève de la modélisation et de l'intégration de l'information des divers points de vue, il contribue par conséquent à l'intégration des activités de conception des SAP. Dans le deuxième chapitre, nous allons étudier les méthodes classiques afin de positionner notre approche.

# CHAPITRE 2.

# Problématique de la cohérence de 'information dans les SAP

## Résumé

**Pour garantir la cohérence de tout ou partie des activités de l'entreprise, des cadres de références ont été développés. Après avoir étudié deux des principaux cadres de référence CIMOSA et BASE-PTA, notamment leur aspect informationnel, nous définissons notre approche.**

**Considérant l'indépendance des activités de conception et l'hétérogénéité de leurs moyens comme un état de fait, la démarche que nous proposons tente d'une part de garantir le partage de l'information avec une interprétation globalement cohérente ; par ailleurs c'est une méthode qui n'a aucun impact sur l'organisation et le fonctionnement de ces activités.**

## 2.1. Problème de la cohérence des moyens

La conception des Systèmes Automatisés de Production (SAP) fait appel à un ensemble d'activités de plus en plus nombreuses et très diversifiées. Il n'existe pas de méthode générale ni de cycle de vie générique, et l'organisation pratique des activités du cycle de vie dépend souvent des moyens utilisés et de leur mise en relation [LHO94]. Les activités font usage de moyens (méthodes, outils, modèles ) très spécialisés afin de répondre à des problèmes localisés.

Dès lors, l'amélioration de la performance des SAP passe par l'optimisation de **fonctions locales** dans le processus de conception. Au moment de mettre en commun ces solutions optimisées localement en vue de répondre à une **solution globale**, le problème de cohésion va se poser.

De plus, les démarches de conception employées sont le plus souvent personnalisées par les acteurs de la conception eux-mêmes, en fonction de leurs besoins particuliers et surtout de leur savoir-faire [KIE96]. Cette hétérogénéité des moyens renforce l'indépendance des activités les unes par rapport aux autres et s'oppose par conséquent à la finalité visée par le processus d'ingénierie de couvrir l'ensemble des phases de conception de façon globale et continue [LES89].

Il y a alors risque de rupture dans la cohérence des moyens sur le cycle de vie du SAP, figure 2.1-1. Cette rupture est due à l'isolement des activités qui sont dotées de leurs propres moyens pour chaque phase [BOI98].

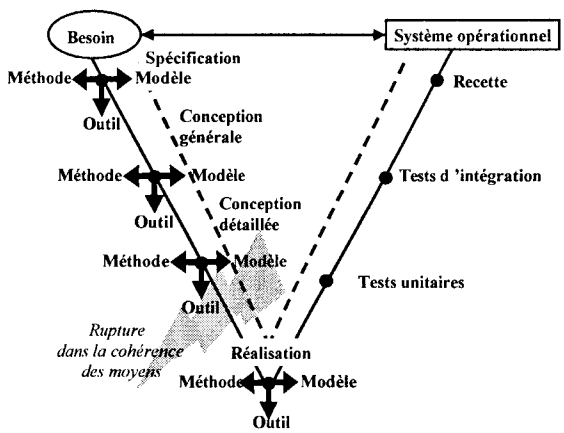



**Figure 2.1-1 : Rupture de la cohérence sur un cycle de vie de SAP**

Il faut donc s'atteler à trouver des moyens permettant d'assurer la continuité initialement visée et l'enchaînement cohérent des activités. Dès lors, la problématique de qualité de conception des SAP se concentre sur la recherche :

- d'une part, de **l'indépendance** des activités et des moyens en interaction ;
- d'autre part, de **cohérence** entre les activités d'ingénierie en terme de cycle de vie particulier.

Ces deux contraintes sont dichotomiques car il faut *« diviser pour réunir »* [MUL97].

D'une manière générale, l'intégration consiste à assembler des composants hétérogènes pour former un ensemble synergique. L'intégration des activités va plus loin qu'un simple échange de données, elle permet un réel partage de l'information, contribuant ainsi à améliorer la coordination des différentes activités.

La plupart des travaux sur la modélisation et l'intégration des activités d'entreprise incluant une **vue informationnelle** sont basés sur un **cadre de référence**. Dans ce cas, un « modèle de référence » d'entreprise est utilisé comme mécanisme d'unification sémantique ou de partage de connaissances, il est de plus implanté dans une infrastructure intégrante [VER96].

## 2.2. Intégration basée sur un cadre de référence

La complexité de l'entreprise ne permet pas de faire son analyse à partir d'un modèle unique qui représente l'ensemble de ses activités. Les principaux cadres de référence vont ainsi définir une démarche de modélisation et d'intégration afin de coordonner les différentes activités de l'entreprise. Parmi les travaux les plus représentatifs (en dehors de **CIMOSA** et **BASE-PTA** que nous étudierons plus en détail), nous trouvons :

La méthodologie **GRAI** (Graphe à Résultats et Activités Inter-reliés) devenu par la suite **GIM** (GRAI Integrated Methodology) propose un ensemble de méthodes qui permettent de modéliser l'Entreprise, d'évaluer ses performances et de proposer des améliorations. L'analyse est effectuée en décomposant les entreprises en quatre sous systèmes (1) informationnel, (2) décisionnel, (3) physique, (4) opérationnel [DOU92], [ROB88].

La méthodologie **PERA** (Purdue Enterprise Reference Architecture ) [WIL92] est caractérisée par son architecture en couches. Elle s'intéresse à l'ingénierie des systèmes industriels et couvre la totalité du cycle de vie de l'entreprise.

L'architecture **GERAM** (Generic Enterprise Reference Architecture and Methodology) s'est inspirée des résultats de CIMOSA, GIM et PERA. L'objectif poursuivi par GERAM est de proposer une référence pour toute la communauté travaillant sur l'intégration d'entreprise, en fournissant les définitions sur la terminologie, un environnement de modélisation cohérent, une méthodologie détaillée [BER94].

L'architecture **ARIS** [SCH94] (ARchitecture for Integrated Information System) a pour objet de réaliser un système d'information intégré pour les systèmes de pilotage d'ateliers. La structure globale du cadre ARIS présente beaucoup de similitudes avec celle de CIMOSA. Il comporte trois niveaux (physique, conception/spécification, définitions des besoins) d'abstraction et quatre vues (information, fonction, organisation, contrôle).

### 2.2.1. Cadre de modélisation en productique : CIMOSA

CIMOSA (CIM Open System Architecture) fournit une architecture « ouverte » (dans la mesure où les formalismes utilisés dans les différentes vues ainsi que les différents niveaux de l'architecture sont extensibles) et cohérente aussi bien pour la modélisation de l'entreprise que pour l'intégration d'entreprise en suivant le concept CIM. Il a été développé par le consortium AMICE (European Computer Integrated Manufacturing Architecture) dans le cadre du projet européen ESPRIT [AMI93]. CIMOSA propose essentiellement un cadre architectural, un cadre de modélisation ainsi qu'une infrastructure intégrante. Un cycle de vie CIM générique a été défini comme une succession de plusieurs phases à utiliser pour construire une architecture particulière, de la définition des besoins jusqu'à son installation et sa maintenance. La structure architecturale est constituée de trois composants : (1) le cadre de modélisation de l'entreprise, (2) une infrastructure intégrante, (3) et un cycle de vie d'un système CIM, figure 2.2-1 [VER96].

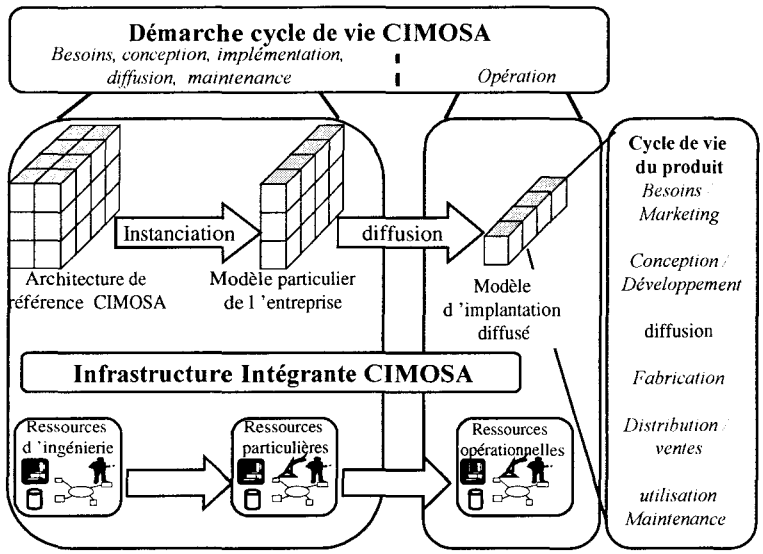



**Figure 2.2-1 : Structure architecturale CIMOSA**

Le cadre de modélisation CIMOSA, figure 2.2-2 (appelé aussi cube CIMOSA) permet l'unification sémantique des concepts partagés dans le système. Le cube CIMOSA est composé de deux parties : une architecture de référence et une architecture particulière.

Il est aussi basé sur trois principes orthogonaux qui permettent de créer un modèle d'architecture de système de production, depuis l'élaboration du cahier des charges jusqu'à son implémentation suivant l'axe **dérivation**. L'axe **génération** permet de modéliser l'entreprise suivant quatre points de vue (extensible) représentant les aspects fonction, information, organisation et ressources. Le principe d'**instanciation** est basé sur trois couches : générique, partielle et particulière.

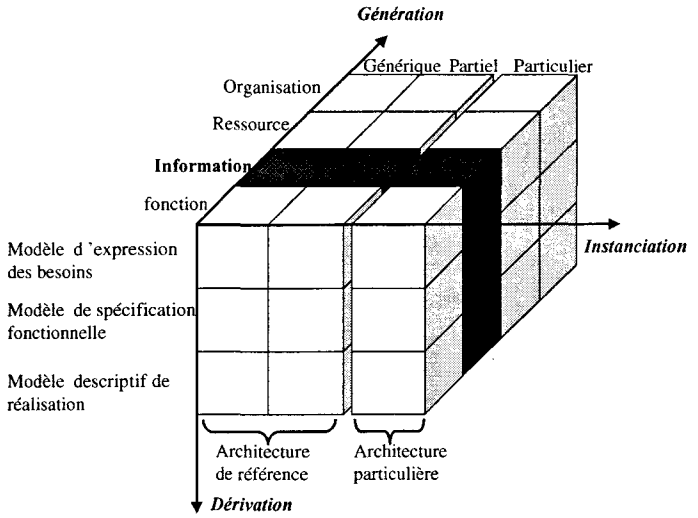



**Figure 2.2-2 : Le cadre de modélisation CIMOSA**

Les éléments de construction du modèle CIMOSA sont dénommés « constructs ». Le modèle particulier de l'entreprise est obtenu à partir de modèles partiels, ces derniers étant eux-mêmes conçus à l'aide de « constructs » génériques. Pour la vue information, les « constructs » nécessaires à la constitution du modèle d'expression des besoins et leurs liens, sont formalisés (modèle M*, [VER89]) sur la figure 2.2-3. Les activités communiquent au moyen d'un système d'information unique et commun pris en compte dans la vue information. Les entrées et sorties des activités de l'entreprise (**Enterprise activity**) sont des vues d'objet (**Object view**). Ces vues d'objet décrivent les manifestations externes d'un ou plusieurs objets d'entreprise (**Enterprise object**) de la façon dont elles sont perçues par un groupe d'utilisateurs ou employées par les activités d'entreprise. Les vues d'objet sont composées d'éléments d'information (**Information element**), c'est à dire les informations élémentaires perçues par les utilisateurs.

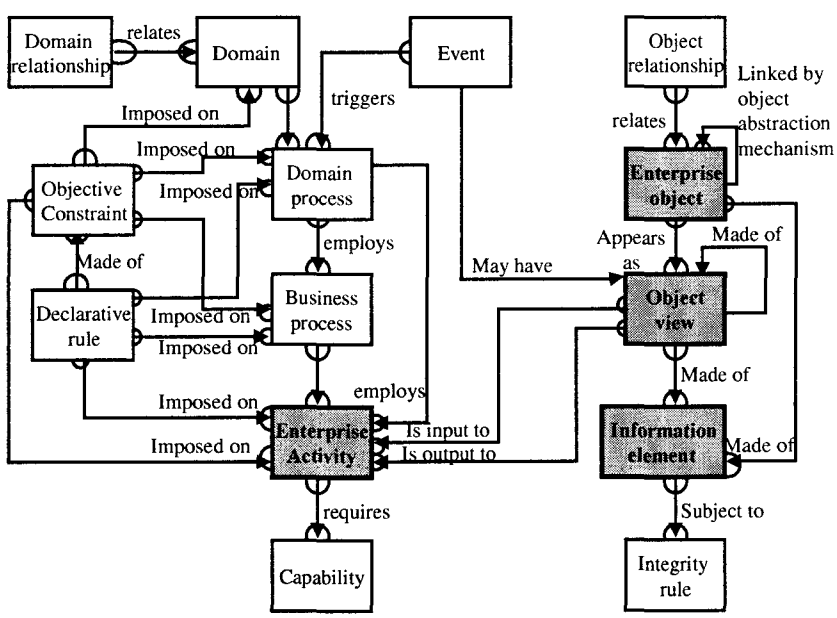



**Figure 2.2-3 : Les constructs de CIMOSA : vue information**

### 2.2.2. Cadre de modélisation en génie industriel : BASE-PTA

Cette présente norme est l'aboutissement des travaux du projet PTA (Poste de Travail de l'Automaticien), défini comme un atelier logiciel ouvert, multi-offreurs, multi-sites, constitué de modules intégrés par les données dans une structure d'accueil unique : l'Atelier Intégré pour le Génie industriel. L'objet principal de cette norme [AFN95] est de proposer un cadre général pour décrire les données manipulées par les divers outils XAO, sous la forme d'un **modèle conceptuel de données** : le modèle BASE-PTA. Ce modèle considéré comme cadre de référence est destiné à servir de base à l'élaboration :

- de normes d'échange de données entre les outils XAO ;
- de normes d'accompagnement définissant le vocabulaire de domaines ou métiers particuliers ;
- de nouveaux systèmes XAO.

BASE-PTA prend en compte des données manipulées dans toutes les activités du cycle de vie des SAP, de l'expression des besoins à l'exploitation et la maintenance.

- **Structure de BASE-PTA, figure 2.2-4**

Le modèle conceptuel proposé est structuré en quatre groupes d'entités décrivant des aspects différents de l'application.

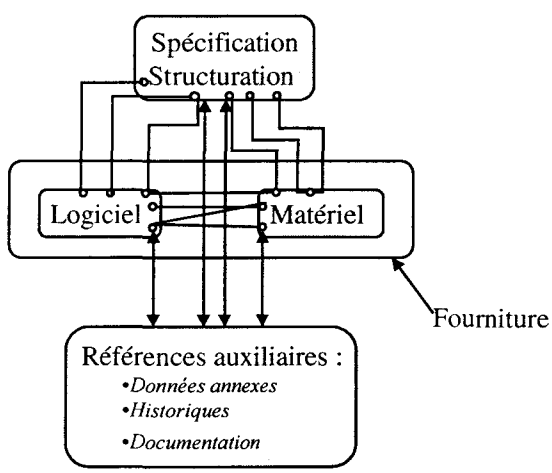



**Figure 2.2-4 : Structure générale du modèle de référence BASE-PTA**

L'aspect spécification/structuration permet de décrire une partie du résultat de la spécification du système de contrôle et de commande. A ce niveau sont décrits les objectifs, les contraintes, l'architecture du SAP ainsi que les comportements des éléments de cette architecture. Les solutions technologiques de ce qui est défini dans la spécification sont données dans l'aspect fourniture matérielle et logicielle. L'aspect références auxiliaires complète la description de l'application en effectuant la documentation, l'archivage etc.

➢ **Elaboration de fichiers d'échanges à partir de BASE-PTA, figure 2.2-5**

L'intégration des outils s'effectue par l'intermédiaire de fichiers d'échange neutres, construits à partir du modèle de référence BASE-PTA en trois étapes.

(1) L'obtention de ces fichiers d'échange repose sur l'extraction d'un Sous-Schéma Conceptuel (SSC). Le SSC est un ensemble connexe d'entités et d'associations inclus dans le modèle BASE-PTA et concernés par un échange.

(2) Ensuite un SSC Appliqué (SSCA) est établi par une succession de phases de spécialisations et restrictions du SSC [COU97]. Le SSCA permet de rendre compte des données propres à un métier particulier.

(3) Enfin les fichiers d'échanges sont produits par traduction des modèles conceptuels précédemment (SSCA) obtenus.

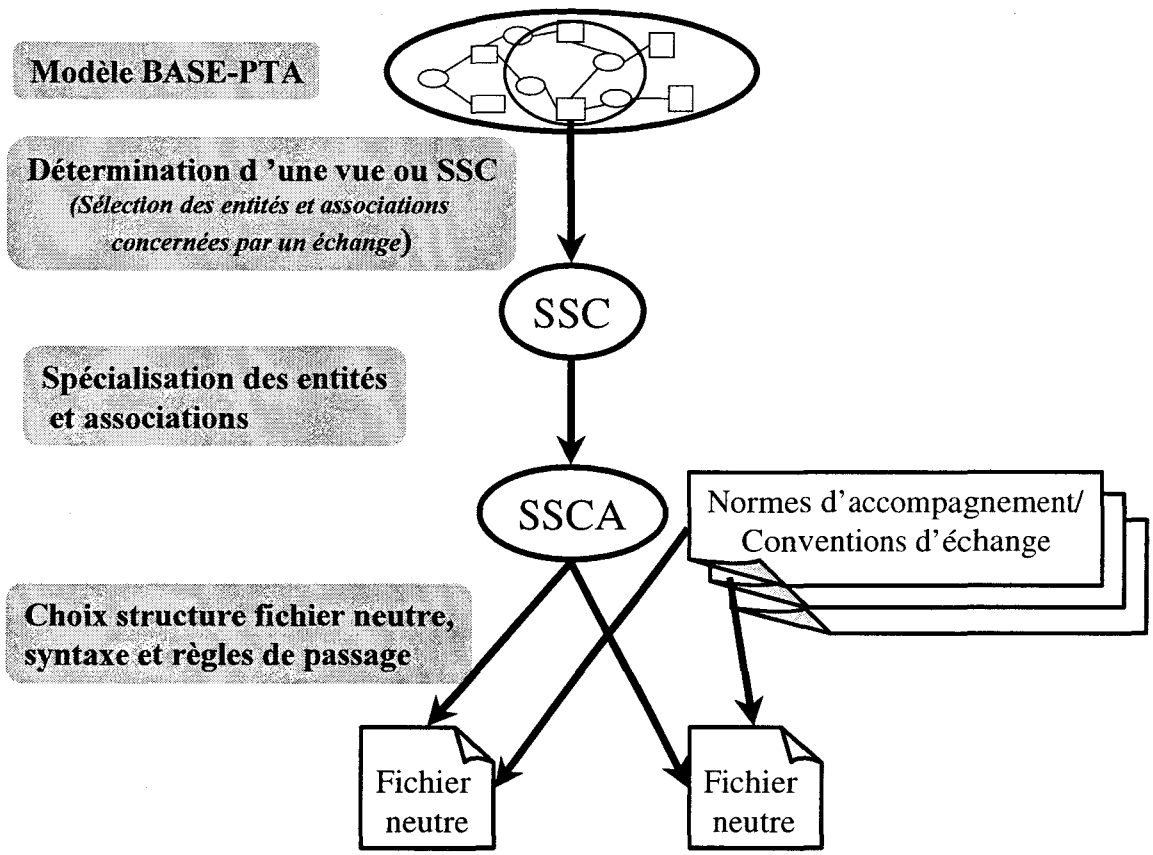



**Figure 2.2-5 : Définition de normes d'échange à partir du modèle BASE-PTA**

Cette démarche d'intégration a fait l'objet d'une application dans le cadre du projet ELEGA (Environnement Logiciel Expérimental pour le Génie Automatique) [KOE90], mettant en œuvre trois outils CAO hétérogènes.

## 2.2.3. Discussion sur les cadres de références

### 2.2.3.1. Cohérence des vues garantie

Les notions qui ressortent en premier dans l'utilisation d'un cadre de référence dont l'objectif est de construire le modèle d'un système de production particulier au moyen d'un modèle générique, sont les suivantes : les notions de dérivation, de particularisation ou d'instanciation.

CIMOSA fournit une architecture de référence entièrement cohérente, sur l'ensemble des vues de l'entreprise nécessaires à sa modélisation. Les architectures partielles et particulières qui en sont dérivées ne peuvent donc être que cohérentes entre elles. BASE-PTA idéalise la représentation des applications d'automatisme à l'aide d'un modèle conceptuel référençant l'ensemble de données impliquées dans les échanges entre activités.

Les cadres de références formalisent ainsi dans une « **image idéale** », le système à concevoir sous forme d'un modèle de référence. Le modèle de référence issu de ce système idéal permet de générer des modèles propres au système réel étudié, figure 2.2-6. Le moyen d'assurer la cohérence entre les modèles du système créés par les activités est alors d'utiliser cette image idéale comme « pivot sémantique » pour chacun des modèles produits [BON98]. Cela suppose une connaissance du domaine d'application, elle privilégie donc une **intégration sémantique**.

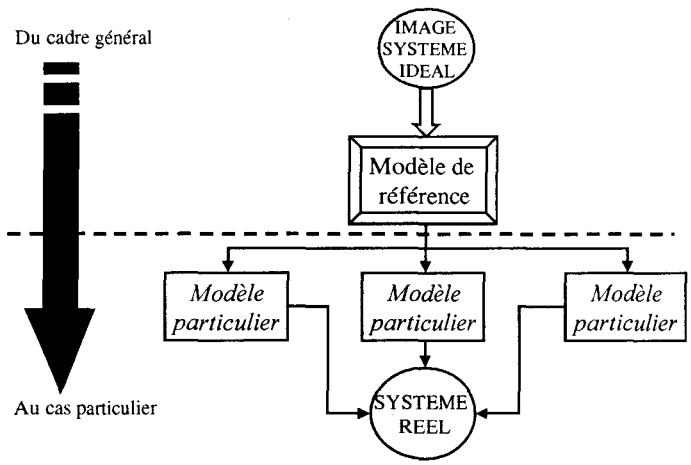



**Figure 2.2-6 : Intégration par modèle de référence**

Ainsi, l'intégration d'activités hétérogènes est mise en œuvre par élaboration d'un modèle particulier, résultat d'instanciations successives. Ce modèle s'applique alors à une infrastructure intégrante permettant, de relier physiquement ces activités hétérogènes et d'assurer les échanges de flux d'information. Cette approche est qualifiée de Principe de Matérialité de l'Ingénierie par [BOI98] : « *la cohérence des vues est postulée par l'existence d'un objet image multi-vues, formalisant la connaissance d'objets applications* ».

### 2.2.3.2. Difficulté d'adaptation des modèles préexistants

Dans la pratique cette modélisation par cadre de référence peut servir de base à l'intégration et à la coordination des entreprises, mais reste néanmoins une intégration mieux adaptée à la mise en place de nouveaux modèles : partant du modèle des besoins pour aller jusqu'aux modèles d'implémentation, et en préjugeant de l'existence des moyens permettant de supporter cette transformation.

La qualité de l'intégration dépend alors fortement de la qualité du modèle de référence et s'applique à un domaine particulier, comme c'est le cas pour BASE-PTA. L'extraction des informations du modèle générique (définition de SSCA) laisse beaucoup d'ambiguïtés quant au mécanisme d'identification des données partageables entre les vues à intégrer.

L'intégration de modèles préexistants à l'égard de ces modèles de référence reste difficile, car elle ne peut se faire que par identification et adaptation de leurs caractéristiques en rapport à celles proposées dans les modèles de référence. Or les objectifs visés par le Génie industriel (§ 2.1), sont d'une part, le développement de manière continue et cohérente des SAP et, d'autre part la prise en compte d'une certaine indépendance des activités de conception de manière à ne pas imposer une méthode de développement particulière.

Notre problématique est différente, car la mise en place d'un référentiel (§ 1.4.5) ne doit pas modifier la structure du projet. Ainsi, il n'intervient pas d'une part sur le découpage, la définition et l'organisation des activités, d'autre part sur leur séquencement dans les différentes phases du projet. De ce fait nous avons choisi une démarche d'intégration dont le point de départ est constitué des points de vues et modèles réellement produits par les différents intervenants, figure 2.2-7.

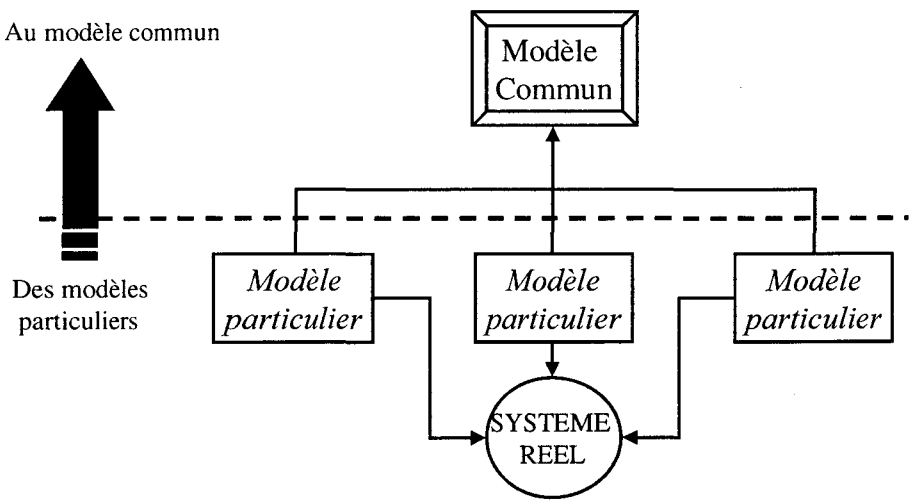



**Figure 2.2-7 : Intégration de modèles préexistants**

## 2.3. Intégration de modèles préexistants

L'étude de systèmes complexes passe nécessairement par l'établissement de plusieurs modèles. La modélisation est *« l'action d'élaboration et de construction intentionnelle, par composition de symboles, de modèles susceptibles de rendre intelligible un phénomène perçu complexe, et d'amplifier le raisonnement de l'acteur projetant une intervention délibérée au sein du phénomène ; raisonnement visant notamment à anticiper les conséquences de ces projets d'actions possibles »*, [LEM90]. Nous allons étudier dans cette partie les principes que nous mettons en œuvre pour assurer la cohérence des modèles à savoir : l'intégration syntaxique des concepts de modélisation, une intégration forte par les données. Tout d'abord, commençons par clarifier la notion de modèle.

### 2.3.1. Notion de Modèle : concepts et formalismes

#### 2.3.1.1. Des modèles pour quel usage ?

Un inventaire rapide des définitions fournies par la littérature permet d'établir trois classes essentielles de modèles [MOR95] : (1) le modèle comme objet d'imitation ; (2) le modèle comme figure d'objet idéal ; (3) et le modèle comme objet de raisonnement.

Le **modèle, objet d'imitation** sert de référence, il est reproductible à l'exemple des prototypes et maquettes.

Le **modèle comme objet idéal**, est le résultat d'un processus d'abstraction, il est alors considéré comme une idéalisation dont l'objectif est de représenter certains aspects relevant d'un système. Des modèles figuratifs peuvent alors être utilisés dans un but de faire communiquer plusieurs acteurs.
[RUM91] considère un modèle comme *« une abstraction de quelque chose de réel qui permet de comprendre avant de construire. Parce qu'il ne tient pas compte des éléments qui ne sont pas essentiels, le modèle est plus facile à manipuler que l'entité originale. L'abstraction est une capacité fondamentalement humaine qui nous permet de gérer la complexité. Pour construire des systèmes complexes, le développeur doit abstraire différentes vues du système, construire des modèles en utilisant une notation précise, vérifier que les modèles satisfont aux spécifications du système, et progressivement ajouter des détails pour transformer les modèles en une implémentation »*.

Toujours dans la définition de modèle comme objet d'abstraction, nous trouvons celle donnée par [VER96] : *« un modèle est une représentation utile d'un certain sujet. C'est l'abstraction (plus ou moins formelle) d'une réalité (ou univers du discours) exprimée à l'aide de formalisme (ou langage) défini par des concepts de modélisation selon les besoins de l'utilisateur ».*

Le **modèle comme objet de raisonnement** est défini dans [MOR95] comme *« une structure formalisée utilisée pour rendre compte d'un ensemble de phénomènes qui possèdent entre eux certaines relations (modèles mathématiques), représentation schématique d'un processus, d'une démarche raisonnée (modèle linguistique). Du point de vue de la méthode scientifique, les modèles ont le statut de formalismes pour permettre des raisonnements logiques ».*

#### 2.3.1.2. Concepts et formalismes de modélisation

Tout modèle (résultats d'un acte de modélisation) est décrit à l'aide d'un formalisme (graphique, mathématique etc.) de modélisation qui est *« une représentation de connaissance à communiquer sans ambiguïté. Il permet de construire des modèles selon des concepts associés »* [DOU90].

Les **concepts** de modélisation possèdent une sémantique clairement définie, ils sont nécessaires pour effectuer une **interprétation** des **formalismes** de modélisation, tandis que ces derniers ont pour objet de permettre une **description** de ces concepts, figure 2.3-2.

Comme le précise [MOR95], *« Description n'est pas concept : la carte routière représente mais n'exprime pas le trajet. Elle n'indique pas que l'autoroute comparée à la route départementale, peut constituer un trajet d'une durée plus courte bien qu'elle montre une distance plus longue entre deux points ».*

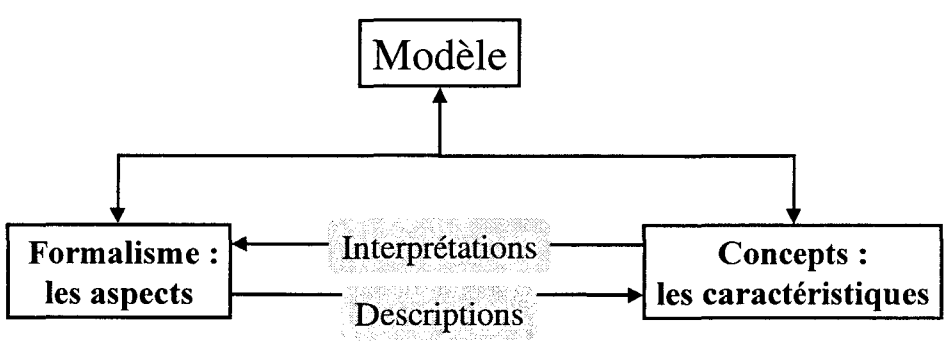



**Figure 2.3-1 : Liens entre Modèle, Concepts et Formalismes de modélisation**

L'un des objectifs du référentiel, est d'assurer la cohérence de l'ensemble des modèles utilisés. Nous pouvons garantir la cohésion entre les différents modèles à travers leurs concepts et/ou leurs formalismes de modélisation. Nous allons justifier dans ce qui suit le choix d'une intégration basée sur les concepts de modélisation.

## 2.3.2. Intégration syntaxique des concepts de modélisation

### 2.3.2.1. Intégration des concepts de modélisation

Compte tenu de nos hypothèses de travail (§1.1.4), la fonction « Intégration des moyens » du référentiel doit disposer de deux composants essentiels permettant d'une part l'intégration des **modèles**, d'autre part l'intégration des **outils** utilisés dans les divers points de vue, figure 2.3-1.

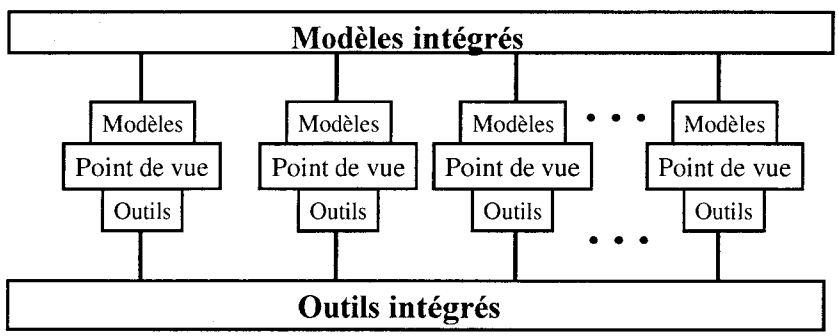



**Figure 2.3-2 : Les deux structures pour l'intégration des points de vue**

➢ **Intégration des outils**

La plupart des travaux sur l'intégration d'**outils** XAO se fait à travers une **plate-forme d'intégration** et son infrastructure intégrante, c'est à dire un support (logiciel et matériel) pour permettre l'échange et le partage d'informations entre les différents points de vue. Parmi les travaux de développement d'une plate-forme d'intégration nous pouvons citer :

(1) Le projet SHOOD [NGU93], qui propose une plate-forme pour le support des activités de Conception et de Fabrication Assistée par Ordinateur (CFAO).

(2) La norme PCTE [MIN91] (Portable Common Tool Environment) qui définit un modèle de référence ainsi que la spécification et la réalisation d'interfaces publiques pour l'intégration d'outils.

[BOI98] a également proposé une méthode d'intégration d'outils de génie industriel autour d'un système d'information unifié, approche basée sur une intégration syntaxique de ces outils.

➢ **Intégration des modèles**

Ainsi l'intégration des modèles peut s'effectuer à travers l'étude des formalismes et/ou concepts de modélisation. Il est plus intéressant de porter l'intégration au niveau des formalismes et/ou concepts de modélisation car ces notions sont totalement indépendantes du système étudié, contrairement aux modèles des systèmes.

[BON98] propose une intégration des modèles à travers leurs formalismes de modélisation. Il se trouve que ces derniers sont sujets à des évolutions assez fréquentes, de ce fait nous avons fait le choix d'intégrer les modèles à travers leurs concepts de modélisation qui présentent plus de stabilité, figure 2.3-3.

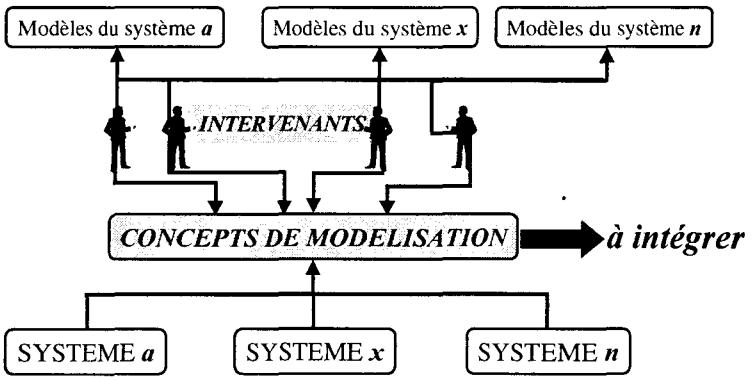



**Figure 2.3-3 : Intégration des concepts de modélisation**

A notre connaissance peu de travaux ont porté sur ce thème d'intégration de concepts de modélisation.

Pour alléger la lecture, nous utilisons dans la suite du mémoire l'expression « intégration de modèles » pour désigner « l'intégration de modèles par leurs concepts de modélisation ».

### 2.3.2.2. Intégration syntaxique

Notre approche vise à obtenir la cohérence des différents modèles utilisés dans les divers points de vue. En intégrant les modèles, on obtient un modèle commun et cohérent grâce à une étude des intersections (liens syntaxiques et recouvrements sémantiques) des différents **concepts** de modélisation (figure 2.3-4).

Si l'espace sémantique occupé par un concept de modélisation recouvre une partie de l'espace sémantique d'un autre concept, il faut l'identifier. Il est alors possible de définir des relations sémantiques entre les concepts de modélisation.

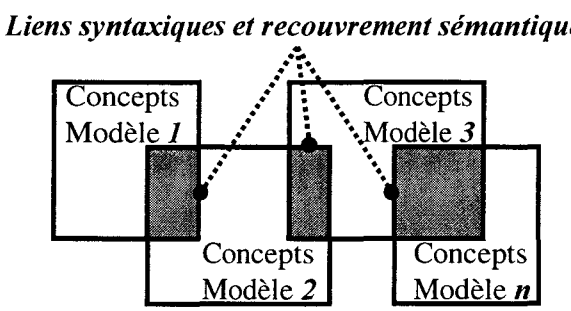



**Figure 2.3-4 : Intersections entre concepts des modèles**

Notre démarche privilégie une **intégration syntaxique**, car la connaissance du système étudié n'est pas un préalable. Etant donné que les concepts de modélisation sont indépendants du système étudié, le type d'application visée n'influe pas sur leur mise en cohérence.

C'est une approche basée sur le principe « d'ingénierie de la matérialité » stipulant que : *la cohérence entre vues peut être assurée par l'identification et la formalisation des intersections entre les points de vue* [BOI98]. Cette démarche semble satisfaire les deux objectifs initialement posés comme problématique du Génie industriel en assurant, d'une part la continuité du processus d'ingénierie et en favorisant, d'autre part l'indépendance des activités, laissant libre cours à toute organisation spécifique du cycle de conception.

Après avoir déterminé les intersections entre les concepts de modélisation, il nous faut maintenant déterminer comment les points de vue peuvent échanger et partager leurs concepts.

## 2.3.3. Intégration forte par les données

### 2.3.3.1. Intégration par les données

Les entrées/sorties des activités sont constituées d'un ensemble de données modélisées, figure 2.3-5. Nous nous intéressons aux données techniques définies par le NIST comme étant des données informatiques relatives à la conception et à l'ingénierie produit, à la fabrication, à la gestion de production, à la maintenance, l'usage, et au marketing.

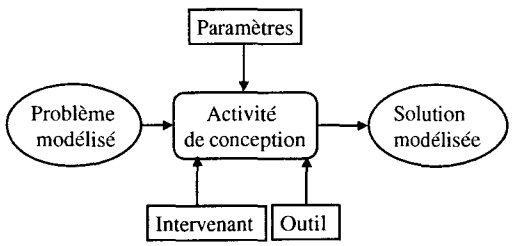



**Figure 2.3-5 : Représentation de l'activité de conception** [COU97]

Ces données techniques sont issues des différents objets du SAP, et chaque point de vue extrait les informations qui lui sont pertinentes à partir de ces objets. Les données techniques sont le plus souvent des objets volumineux, évolutifs, composites et pouvant être concernés par plusieurs points de vue. D'un point de vue à un autre, les informations modélisées et perçues différemment doivent contribuer à la résolution d'un même problème, elles peuvent ainsi provoquer des conflits lors de la communication intra-activité ou inter-activités.

Le rôle du système d'information est de réguler le partage et la distribution de l'information et de gérer les données utiles pour atteindre les objectifs [PIC97].

La principale difficulté consiste à obtenir une perception collectivement cohérente des informations échangées entre les différents points de vue.

Pour exploiter de manière optimale des données de nature différente, issues de plusieurs sources, nous procédons par une intégration des données parce que c'est dans la **structure des données** que se concentre la **stabilité du système d'information**.
Chaque point de vue possède une interprétation partielle et propre de chaque donnée échangée, de plus les données techniques sont caractérisées par une structure complexe et une sémantique très riche.

L'intégration par les données consiste à gérer et à distribuer les données communes aux activités, indépendamment des traitements et en respectant l'intégrité de ces données. Cette intégration nécessite que les données soient représentées par un modèle de données le plus indépendamment du niveau physique car il représente l'aboutissement du processus d'ingénierie des besoins ou d'analyse [MIL98].

Les points de vue disposent alors d'une interprétation commune des données qu'ils échangent ou partagent. Le terme Univers du discours [TAR83] est utilisé pour désigner les choses et les événements auxquels se réfèrent les différents points de vue. Le référentiel permet alors aux différents points de vue d'opérer sur un modèle commun dont ils ont une interprétation partielle mais collectivement cohérente. Ce modèle commun et complet est obtenu par intégration des différents modèles partiels. Par conséquent, ces modèles apparaissent alors comme des vues du modèle commun, qui permettent aux points de vue d'en avoir une interprétation cohérente.

C'est typiquement une **représentation multidisciplinaire** où chaque intervenant manipule une ou plusieurs vues d'un modèle commun de l'objet de la conception [TIC95]. Il est nécessaire de disposer d'un principe intégrateur apportant une intelligibilité sinon une compréhension commune de l'objet de la conception, de telle sorte que l'interaction entre les points de vue repose sur des représentations pertinentes, partagées et cohérentes.
Cette approche dite d'intégration par les données convient particulièrement, dès qu'il est nécessaire de faire travailler en parallèle une équipe interdisciplinaire, pour considérer tous les aspects d'un problème industriel complexe [COU97].
Plusieurs niveaux d'intégration sont alors envisageables selon que la communication se fait directement entre les points de vue ou au travers d'une base de données techniques.

### 2.3.3.2. Choix d'une Intégration forte

Il existe trois niveaux d'intégration de modèles, depuis leur simple connexion par le biais de leurs concepts de modélisation, jusqu'à leur intégration forte.

➢ **Connexion de concepts de modélisation**

A un premier niveau, il est possible d'établir des connexions entre modèles particuliers à travers leurs concepts de modélisation, afin qu'ils puissent échanger directement des données.

Cette étape, qualifiée par [ATZ93] de mécanisme le plus rudimentaire pour intégrer les modèles et les méthodes, correspond aux premiers « essais » d'intégration d'outils pour les applications informatiques par la réalisation de traducteurs réalisant des passerelles entre outils.

Cependant, elle ne réalise pas une réelle intégration, et ne permet pas d'obtenir une interopérabilité générale des modèles [BON98]. De plus, si l'accès direct aux différents modèles permet de pouvoir utiliser les ressources de manière optimale, l'inconvénient majeur réside dans le nombre d'interfaces de connexion à réaliser, de l'ordre de n(n-1) où n représente le nombre de modèles, figure 2.3-6. De ce fait, la prise en compte d'un nouveau modèle conduit à étudier ses interactions avec chacun des autres modèles.

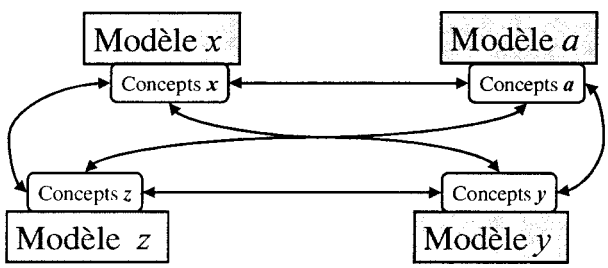



**Figure 2.3-6 : Connexion des modèles**

➢ **Intégration faible**

Contrairement à la simple connexion, les modèles ne communiquent plus directement entre eux mais, à travers un support neutre de représentation des données [BOU95], figure 2.3-7.

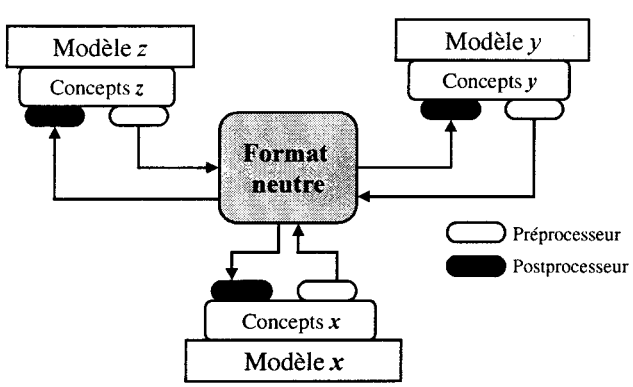



**Figure 2.3-7 : Intégration faible**

Les données utiles à la connexion des différents concepts des modèles sont formalisées dans un modèle neutre, assurant ainsi un standard dans les échanges. Les différents concepts des modèles sont alors interprétés par rapport à une référence commune [BON98], ce qui implique que la référence doit être aussi riche que tous les concepts concernés, tout en restant indépendante. Un ensemble d'interfaces (2*n, avec n le nombre de modèles) et de protocoles doit également être défini entre le format neutre et chaque concept de modélisation.

L'existence de ce support neutre peut garantir la persistance et l'intégrité des données échangées. Chaque modèle doit maintenant garantir la cohérence entre ses propres données et celles partagées.

Ce principe d'intégration faible a été adopté dans le projet BASE-PTA. De même, la norme STEP (ISO 10303) propose un format neutre de l'information indépendant de toute application particulière du système, permettant aux différents outils XAO d'un système CIM, d'échanger et de partager un modèle de produit neutre.

➢ **Intégration forte**

A ce niveau d'intégration, l'ensemble des concepts de modélisation est mis en commun. L'intégrité des données est alors assurée puisque les informations échangées font partie d'un même noyau, de même le problème de connexion ne se pose plus. Concrètement, ce dernier niveau nécessite de *« casser les structures logicielles classiques pour les intégrer sous la forme d'une base commune »* [LOM94], et conduit à la conception d'ateliers intégrés regroupant l'ensemble des concepts utilisés par les différentes activités.

**Nous avons fait le choix d'une approche basée sur l'intégration forte**, nous cherchons ainsi à formaliser les différents concepts de modélisation utilisés dans les différents points de vue afin de permettre leur mise en commun. La finalité étant d'obtenir des données partageables, persistantes et collectivement cohérentes. Il nous faut donc décloisonner (figure 2.3-8) les points de vue grâce à une intégration de leurs modèles.

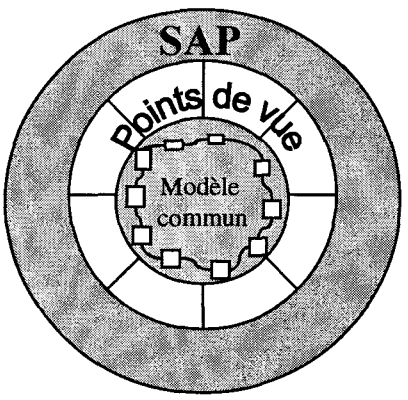



**Figure 2.3-8 : Intégration forte**

La mise en commun des informations, permet à l'ensemble des constituants du projet de communiquer à travers une base de données techniques.

Les principales difficultés à surmonter lors de l'intégration forte de données, sont les incohérences dues principalement à :

- la **polysémie** lorsqu'une donnée a des significations différentes dans plusieurs modèles,
- La **synonymie**, des termes différents employés pour désigner la même donnée
- L'**hypernimie** lorsque différentes modélisations, représentent un même concept dans différents modèles partiels et engendrent des conflits de structure [LAR97].

# Conclusion de la Partie A

La maîtrise de l'information est un préalable à tout fonctionnement harmonieux des entreprises.
La complexité croissante des SAP caractérisés par une hétérogénéité manifeste de leurs constituants se traduit par la diversité des activités de conception et par conséquent par une offre multiple en moyens (méthode, outils XAO et modèles).
Les informations circulant entre les différentes activités de conception doivent alors être exactes, sans ambiguïté, compréhensibles de tous, modifiables, traçables, etc. Malheureusement nous savons que cette situation idéale est loin de la réalité.
Le défi est donc d'assurer la cohérence des informations échangées et partagées.

Les principales démarches d'intégration étudiées reposent sur une intégration sémantique basée sur un modèle de référence applicable sur un domaine particulier. Cette approche permet de réutiliser par spécialisation un modèle de référence afin d'obtenir un modèle de données particulier. C'est une démarche qui convient parfaitement à la mise en place d'un nouveau système. Mais l'intégration des moyens préexistants suivant cette approche par modèle de référence se fait difficilement. Compte tenu des objectifs du Génie industriel, à savoir : la cohérence et l'indépendance des activités, c'est alors une contrainte supplémentaire à gérer.

De ce fait nous proposons une démarche qui, partant de moyens préexistants (les modèles utilisés dans les différents points de vue) étudie les intersections (intégration syntaxique) entre les concepts de modélisation afin d'obtenir un modèle commun par intégration forte des données.
Notre démarche contribue à l'intégration des activités de conception d'un SAP en cherchant la cohérence globale des données manipulées dans les différents modèles. Pour cela, nous avons mis en œuvre les deux principes suivants :

- Une intégration syntaxique des modèles à travers leurs concepts de modélisation, en identifiant et formalisant les intersections afin d'obtenir une interprétation commune des modèles, indépendamment du SAP étudié ;
- Une intégration forte par les données, garantissant ainsi l'intégrité des données manipulées qui supportent la stabilité des systèmes d'information.

L'établissement d'un modèle commun est l'objet de la deuxième partie de ce mémoire. Il mettra en œuvre deux principes essentiels : la méta-modélisation (par le biais du langage UML) et la réutilisation par la définition de patterns.

# Partie B

## Résumé

**Cette deuxième partie du mémoire présente les deux principes que nous mettons en œuvre pour l'intégration de modèles : la métamodélisation et la réutilisation. Au chapitre 3, nous procédons par métamodélisation pour traduire les différents modèles des intervenants dans un langage unique afin de permettre l'étude des recouvrements entre modèles. Ce processus de métamodélisation est facilité par des mécanismes de réutilisation grâce aux patterns que nous créons dans le chapitre 4.**

**Dans le chapitre 5, nous appliquons notre démarche au projet CASPAIM développé au LAIL.**

# Introduction Partie B

La gestion de l'information des systèmes complexes tels que les systèmes automatisés de production exige le développement d'un élément unificateur. Cette ressource doit s'adapter au cycle de vie du système et permettre aux différents concepteurs d'échanger, de partager des informations pertinentes. L'offre en modèles est multiple, de même les disciplines concernées par ces échanges sont variées. L'objectif que nous nous fixons à travers la mise en place du référentiel est de pouvoir détecter les points de convergence et les contradictions émanant de l'unification de modèles divers. Cette unification ne doit pour autant contraindre les différents concepteurs dans leur expertise ; pour cela nous proposons d'utiliser la métamodélisation, qui est un acte de modélisation appliqué à un modèle. Cette traduction dans un langage unique de tous les modèles va ainsi faciliter l'étude des intersections entre ces modèles, grâce à une unification sémantique. Plusieurs travaux (IRDS, SHOOD, CDIF etc.) ont montré l'intérêt de procéder par métamodélisation dans la conception d'un système d'information. Le chapitre 3 sera consacré à l'étude de l'intégration des modèles par métamodélisation suivant une approche orientée objet. Nous allons adapter des techniques éprouvées en génie logiciel à notre démarche de conception du référentiel.

Dans le chapitre 4, nous montrons l'intérêt de la réutilisation dans le processus de conception du référentiel. D'une manière générale, la réutilisation est un enjeu scientifique et économique important qui va contribuer à l'amélioration des facteurs QCD :

- La qualité sera majorée par la réutilisation de composants testés et validés ;
- Le coût sera réduit car le composant est réalisé une fois, puis réutilisé à plusieurs reprises ;
- Enfin les délais seront forcément réduits, car on gagne inéluctablement du temps de la phase de conception, à la maintenance.

Une réutilisation efficace suppose qu'il existe des invariants de modélisation. L'étude des modèles utilisés dans la conception des systèmes automatisés de production, montre une prédominance des modèles descriptifs dont les concepts de base sont fournis par la théorie des graphes. Nous avons alors orienté nos recherches de composants réutilisables dans la théorie des graphes. Ces composants sont capturés et documentés à l'aide de patterns.

# CHAPITRE 3.

# Intégration suivant une approche orientée objet par Métamodélisation

## Résumé

**Dans ce chapitre nous justifions le choix d'une approche orientée objet mise en œuvre à l'aide du langage UML. Puis nous proposons une démarche d'intégration des concepts de modèles préexistants par métamodélisation. Ensuite nous définissons une architecture du référentiel.**

## 3.1. Approche orientée objet

### 3.1.1. Choix du langage UML

La réalisation d'une intégration forte par les données (§2.3.3) nécessite un modèle commun. L'obtention de ce modèle commun doit permettre une meilleure compréhension des différents modèles et une définition sans équivoque de la structure des données, afin d'assurer l'échange et le partage d'informations cohérentes. Pour réaliser ce modèle commun, nous devons donc traduire les différents concepts de modélisation existants, à l'aide d'un **langage unique.**

Nous avons choisi une démarche qui s'appuie sur une analyse et une conception orientée objet. Cette approche est bien adaptée à la modélisation des systèmes complexes grâce à sa grande modularité. Par ailleurs l'approche objet permet un développement incrémental qui s'associe parfaitement à notre démarche progressive qui intègre au fur et à mesure différents points de vue. C'est aussi une approche qui convient à l'intégration d'éléments disparates, comme le souligne [MUL00] *« l'approche objet tire sa force de sa capacité à regrouper ce qui a été séparé, à construire le complexe à partir de l'élémentaire, et surtout à intégrer statiquement et dynamiquement les constituants d'un système ».*

Le langage que nous utilisons pour la métamodélisation doit posséder une sémantique assez riche pour traduire tous les concepts utilisés dans les divers modèles, il doit aussi être facilement compréhensible des autres intervenants. Il doit par conséquent être un modèle descriptif, basé sur une **représentation graphique** dont la syntaxe est à la fois simple, intuitive et expressive.

Notre choix s'est donc porté sur le langage UML (Unified Modeling Language). Pour endiguer la prolifération des méthodes objet dans la première moitié des années 90 la fusion des trois principales approches orientées objet : OMT (Object Modeling Technique) [RUM91], Booch [BOO91] et OOSE (Object Oriented Software Engineering) [JAC92] donnent naissance au langage de modélisation UML. Comme le souligne [BEZ95], *« le maître mot actuellement est l'unification. Après l'apparition de multiples propositions, la communauté des méthodologistes sentant venir le danger, semble vouloir se restreindre, se discipliner, s'organiser ».*

Le langage UML, l'architecture standardisée de méta-modélisation MOF (Meta Object Facility) et le format d'échange XMI (XML Model Interchange ) constituent les trois recommandations majeures de l'OMG (Object Management Group) qui ont fortement influencé la communauté objet et le paysage de la modélisation logicielle [BEZ98].

Le principal objectif de l'OMG est de faire émerger des standards industriels pour l'intégration d'applications distribuées hétérogènes et la réutilisation de composants logiciels, en utilisant des technologies objet. Face à la prolifération de produits matériel et logiciel, l'OMG propose l'architecture distribuée CORBA (Common Object Request Broker Architecture) pour répondre au besoin d'interopérabilité.

UML[1] est un langage évolutif, avec un champ d'application large, supporté par de nombreux outils, et normalisé pour l'industrie. Il permet de modéliser, visualiser, construire, et documenter les artefacts d'un système à forte composante logicielle. Cependant il s'applique aussi à différents types de systèmes (télécommunications, sciences, transports, services bancaires, aérospatiale etc.), domaines, méthodes et processus. UML n'est donc pas limité à la modélisation de logiciel, dans nos travaux nous utilisons UML d'une part pour la modélisation et l'intégration d'applications du Génie industriel, d'autre part comme schéma conceptuel pour la construction de bases de données.

### 3.1.2. Concepts de base du langage UML

Dans cette partie nous effectuons une présentation succincte des éléments de modélisation UML que nous utilisons dans ce mémoire, à partir des références suivantes : [BRJ00], [ERI98], [KET98], [KRU00], [MUL00], [ROQ00]. Le lecteur peu habitué au langage UML disposera ainsi de suffisamment d'information pour comprendre la suite. Nous invitons le lecteur familier avec le langage UML à passer directement au §3.2.

La terminologie UML inclut trois sortes de briques de base : des **éléments** (structurels, comportementaux, de regroupement et d'annotation), des **relations** et des **diagrammes** au nombre de neuf. Les éléments sont les abstractions essentielles à un modèle, reliés entre eux au sein de diagrammes.

A l'aide de ces briques de base, UML représentent les systèmes à travers les modèles suivants :

- Le modèle de classes qui capture la **structure statique**,
- Le modèle d'états qui exprime le **comportement dynamique** des objets,
- Le modèle de cas d'utilisation qui décrit les **besoins de l'utilisateur**,
- Le modèle d'interaction qui représente les **scénarios** et les flots de **messages**,
- Le modèle de réalisation qui montre les **unités de travail**,
- Le modèle de déploiement qui précise la **répartition des processus**.

---

[1] Les spécifications et modifications récentes sur les travaux de l'OMG et notamment le manuel de référence UML sont disponibles sur le site suivant: http://www.omg.org.

Etant donné que nous construisons un système d'information, nous nous intéressons plus aux **modèles structurels** pour montrer un ensemble structuré d'éléments. Ce modèle est complété par des contraintes que nous définissons grâce aux mécanismes d'extensibilité prévu par UML. Nous définissons également des contraintes OCL (Object Constraint Language) pour assurer la cohérence des modèles obtenus.

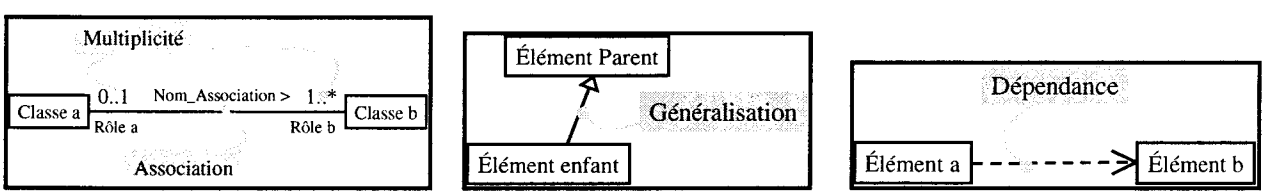
Nous allons successivement étudier les concepts de base UML suivant : les relations, les mécanismes d'extensibilité, les diagrammes de classe et les paquetages.

#### 3.1.2.1. Les relations

Les relations les plus courantes sont au nombre de trois : la dépendance, l'association et la généralisation. Chaque relation est une connexion sémantique entre éléments, et est représentée par un type de trait.

(1) Une association est une relation structurelle qui décrit un ensemble de liens entre les objets d'un élément donné avec ceux d'un autre élément. Une association est représentée par un trait plein, reliant une classe à d'autres classes ou à elle-même, figure 3.1-1(a). Une association comprend au niveau de ses extrémités des multiplicités et éventuellement des noms de rôles ou des qualifications. Par défaut, une association est navigable dans les deux sens sauf indication contraire. Il existe plusieurs types d'associations à savoir l'agrégation et la composition que nous présentons dans le paragraphe suivant.

(2) La généralisation est une relation entre un élément général (appelé super-classe ou parent) et un élément dérivé (appelé sous-classe ou enfant) de celui-ci. C'est une relation qualifiée de est_une_sorte_de (la sous-classe est une sorte de la super-classe); elle permet d'indiquer des relations parent/enfant. L'enfant hérite des propriétés du parent et peut les modifier ou en rajouter d'autres qui lui sont propres. Les instances de l'enfant peuvent être utilisées partout où les instances du parent s'appliquent. Une généralisation est représentée par une flèche dont la pointe creuse est dirigée vers le parent, figure 3.1-2 (b).




(a) (b) (c)

**Figure 3.1-1 : Les trois types de relations**

(3) La dépendance est une relation d'utilisation qui indique qu'un changement de spécification d'un élément (classe, paquetage, note) peut en affecter un autre qui l'utilise, mais l'inverse n'est pas nécessairement vrai. Une dépendance est schématisée par une flèche en pointillés dont l'origine est rattachée à l'élément dépendant, figure 3.1-2 (c).
Il existe une quatrième relation : la réalisation qui est une sorte de croisement entre la dépendance et la généralisation, elle sera présentée en annexe.

#### 3.1.2.2. Les mécanismes d'extensibilité

UML possède quatre **mécanismes généraux** qui s'appliquent de manière cohérente à l'ensemble du langage. Il s'agit des **spécifications** qui fournissent une base sémantique pour tous les modèles, des **décorations** qui sont les détails ajoutés à la notation graphique de base de chaque élément, des **distinctions communes** propres à la modélisation orientée objet et portant sur la dichotomie classe/objet et enfin des **mécanismes d'extensibilité**.
Le langage UML reste « ouvert » grâce aux mécanismes d'extensibilité qui sont au nombre de trois : les stéréotypes, les étiquettes (tagged value) et les contraintes. Ces mécanismes d'extensibilité permettent d'adapter et de compléter UML, pour répondre aux besoins d'un projet particulier.
(1) Lorsque la sémantique de base d'un élément n'est plus suffisant pour modéliser le système étudié, le vocabulaire UML peut alors être étendu grâce aux **stéréotypes.** Ces derniers permettent de créer de nouvelles sortes de briques de base qui dérivent de celles qui existent, en les adaptant à un problème donné. Un stéréotype est représenté par son nom entre guillemets.

(2) Une **étiquette** permet l'ajout de nouvelles propriétés à un élément de base UML, permettant ainsi la création de nouvelles informations dans la spécification de cet élément. Une étiquette est représentée par une chaîne de caractères entre accolades, placée sous le nom de l'élément auquel elle se rapporte.

(3) Une **contrainte** élargit la sémantique d'une brique de base UML, permettant ainsi l'introduction de nouvelles règles sémantiques ou de changer celles qui existent. Elles précisent les conditions nécessaires à l'obtention d'un modèle correct. Une contrainte est représentée par une chaîne de caractères entre accolades, placée à coté de l'élément auquel elle est associée. Il est aussi possible de décrire cette contrainte grâce au langage OCL (Object Constraint Language) que nous étudierons au § 3.1.3.

### 3.1.2.3. Les diagrammes de classes

Il existe quatre **diagrammes structurels** en UML qui permettent de visualiser, spécifier, construire et documenter les aspects statiques d'un système.

(1) Les diagrammes de déploiement illustrent la vue de déploiement statique d'une architecture : le déploiement des composants sur les dispositifs matériels.

(2) Les diagrammes de composants (physiques) représentent la vue d'implémentation statique d'un système.

(3) Les diagrammes d'objets illustrent les structures de données, les vues statiques d'instances des éléments que l'on trouve dans les diagrammes de classe.

(4) Les diagrammes de classe sont constitués d'un ensemble de classes, d'interfaces et de collaborations ainsi que leurs relations. Ils illustrent la vue de **conception statique d'un système**.

Notre démarche a pour objectif final, l'élaboration de diagrammes de classes permettant de rendre compte de la totalité des données manipulées dans les différents points de vue, puis d'effectuer l'implémentation de ces diagrammes de classe dans une base de donnée.

L'exemple sur la figure 3.1-2 montre un diagramme de classes avec les notions les plus usuelles.

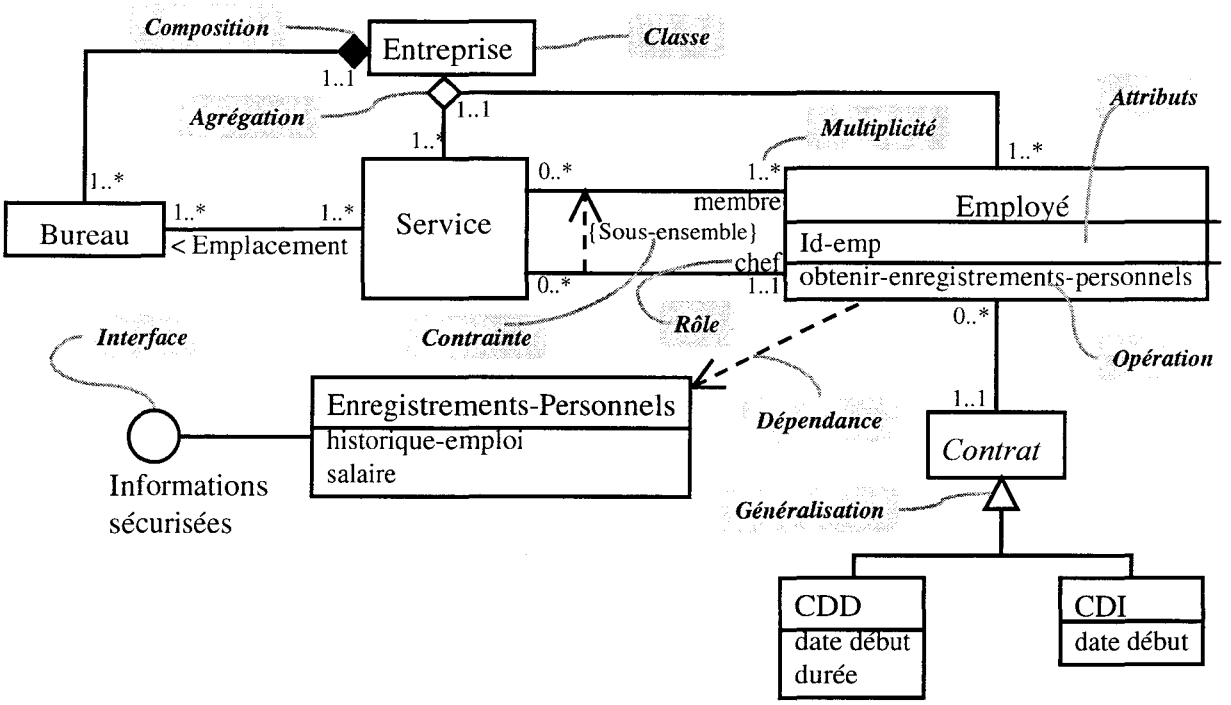



**Figure 3.1-2 : Exemple de diagramme de classes**

Les classes sont représentées par des rectangles comprenant un compartiment pour le nom de la classe et deux autres qui contiennent respectivement les attributs et les opérations. Les attributs et opérations peuvent être visibles par tous (public), par les enfants de la classe (protégé) ou uniquement par la classe qui les contient (privé).

Une agrégation (losange du coté de l'agrégat) représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant, c'est une relation de type de « fait-partie-de ». La composition (losange noir) est une forme particulière d'agrégation, l'agrégat contient alors physiquement ses éléments.

Une interface (petit cercle relié à la classe qui fournit les services) décrit le comportement visible d'une classe, elle fournit ainsi une vue totale ou partielle d'un ensemble de services offerts par un ou plusieurs éléments.

#### 3.1.2.4. Les paquetages

Un paquetage est un mécanisme UML d'ordre général qui permet d'organiser tous types d'éléments liés et cohésifs (classes, paquetages, diagrammes etc.) en groupes.

Chaque élément est la propriété directe d'un seul et unique paquetage, la visibilité de chaque élément est contrôlée de la même manière que celle des attributs et des opérations contenus dans des classes. Les paquetages sont définis dans UML comme des éléments de gestion de configuration, de stockage et de contrôle d'accès.

Les paquetages peuvent être reliés par les mécanismes suivants sur la figure 3.1-3.

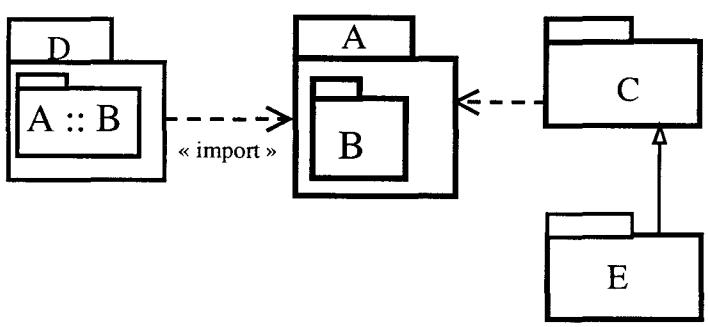



**Figure 3.1-3 : Relations entre paquetages**

- La contenance stricte : un paquetage peut en contenir un autre, dans notre exemple figure 3.1-4, A contient le paquetage B ;
- L'utilisation : si un élément d'un paquetage C est en relation avec un autre élément d'un paquetage A, alors C utilise A. L'utilisation résume toutes les relations qui existent entre les éléments des paquetages C et A, elle est unique et est représentée par une relation de dépendance ;
- L'importation permet aux éléments d'un paquetage d'accéder aux éléments d'un autre paquetage dans un seul sens. D importe le paquetage B appartenant au paquetage A ;
- La généralisation est employée pour décrire des familles de paquetages, elle est similaire à la généralisation entre classes.

### 3.1.3. OCL (Object Constraint Language)

Les diagrammes obtenus tel que le modèle de classes ne sont pas suffisamment précis du fait de la notation graphique. Cette notation peut induire des ambiguïtés dans l'interprétation et la spécification des objets. Il faut alors nécessairement définir des contraintes sur les objets. Cela peut être effectué par des langages formels (le langage z par exemple), toutefois ces langages sont basés sur des définitions mathématiques et sont difficilement exploitables par le commun des concepteurs. L'utilisation d'un langage naturel présente l'avantage d'être facilement compréhensible mais ne résout que partiellement les problèmes du fait de sa manque de précision.

Ainsi, IBM a développé OCL[2], un langage formel qui cherche à tirer avantage des langages formels et des langages naturels. Le langage OCL est utilisé pour exprimer de manière précise des contraintes qui ne peuvent être spécifiées dans les diagrammes UML. Nous présentons dans cette partie les concepts de base du langage OCL.

OCL est un langage formel d'expressions typées, il peut être utilisé pour :

(1) spécifier des invariants de classes ou de types sur les diagrammes de classes,

(2) spécifier des pré ou post conditions sur les opérations et méthodes,

(3) spécifier des contraintes sur les opérations,

(4) naviguer,

(5) décrire des conditions de garde.

Le mot clé **`context`** définit le contexte de l'expression OCL. Les mots clés **`inv`**, **`pre`** et **`post`** désignent respectivement les stéréotypes « invariant », « precondition » et « postcondition ». Chaque contrainte OCL est écrite dans le contexte d'une instance ou d'un type spécifique, le mot clé **`self`** est utilisé pour faire référence à l'instance contextuelle.

Prenons l'exemple du diagramme de classes, figure 3.1-4 adapté du manuel de référence OCL.

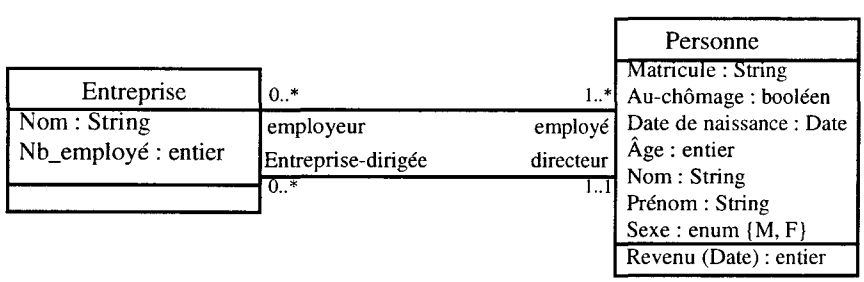



**Figure 3.1-4 : Exemple de diagramme de classes**

[2] Les informations sur le manuel de référence OCL sont disponibles sur le site de l'OMG, http://www.omg.org

➢ **Une expression OCL** peut se rapporter aux types, classes, interfaces, associations. Il en est de même pour tous les attributs, opérations, rôles et méthodes. La valeur d'une propriété (attribut, opération, rôle et méthode) d'un objet, définie dans un diagramme de classe est spécifiée par un point suivi du nom de la propriété.

```
context untype inv :
self.propriété
```

➢ **Invariant et Post (Pre)condition**

Invariant, le nombre d'employés doit être supérieur à 10.

```
Context Entreprise inv :
Self.Nb_employé > 10
```

Postcondition sur le revenu des employés, l'opération revenu prend comme paramètre la date.

```
Context Personne :: revenu (d : date) : entier
Post : result = 5000
```

Il est possible de récupérer des valeurs antérieures dans les pre et postconditions. Pour faire référence à une valeur avant le début de l'opération on utilise le symbole '@'. Par exemple lors de la mise à jour de l'âge d'une personne :

```
Context Personne :: jour-anniversaire()
Post : âge = âge@pre + 1
```

➢ **Navigation**

La navigation est effectuée grâce aux rôles. L'expression objet.nom_rôle retourne un ensemble d'objets.

**Exemple :** Assurer que le nombre d'employés n'est pas nul

```
Context Entreprise
Inv : self.employé-> notEmpty
```

On peut accéder à un ensemble d'objets satisfaisant un certain nombre de critères par les opérateurs **select, reject** et **collect**. Par exemple, sélectionner l'ensemble des employés ayant plus de 50 ans.

```
Context Entreprise inv :
Self.employé->select(âge>50)
```

Lorsqu'une contrainte s'applique à tous les éléments on utilise l'opération **forAll**.

```
Context Entreprise inv :
Self.Employé->forAll(Person e1,e2 | e1<> e2 implies e1.matricule
<>e2.matricule)
```

## 3.2. La métamodélisation comme moyen d'intégration

### 3.2.1. Le principe de métamodélisation

Le référentiel n'est pas construit en préjugeant de ce que devrait être le système étudié et les modèles développés par les concepteurs. Comme nous l'avons précisé dans le cahier des charges du référentiel, la mise en place de ce dernier ne doit pas jouer sur les points de vue à définir, ni sur le choix des concepteurs. Notre principal objectif est de représenter dans un modèle commun les différents modèles issus des points de vue afin d'assurer le partage et l'échange des informations qu'ils manipulent. La plupart des modèles utilisés dans les points de vue sont souvent stables et validés, mais leur spécialisation dans un domaine donné pose des problèmes de cohérence que nous avons évoqués dans le chapitre précédent.

A partir des modèles utilisés dans les différents points de vue, nous construisons un modèle commun en prenant en compte des intersections entre les différents concepts de modélisation.

D'après [BRA95], l'unification ou l'intégration de modèles diversifiés passe nécessairement par la métamodélisation. On parle de métamodélisation dès lors que l'acte de modélisation s'applique sur des modèles. Un acte de métamodélisation a les mêmes objectifs qu'un acte de modélisation avec pour seule différence l'objet de la modélisation, figure 3.2-1 [OUS97].

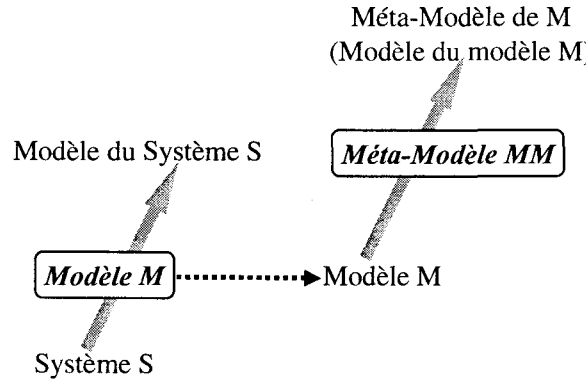



**Figure 3.2-1 : Modèle et métamodèle**

La métamodélisation concerne l'analyse systématique d'un modèle dans le but d'en spécifier clairement les concepts, d'après [OUS97] elle peut être utilisée comme :

- technique de dialogue, dans ce cas elle sert de moyen de comparaison et d'unification de modèles ;
- technique de réflexivité, elle permet alors à un modèle de s'auto-représenter ;
- technique d'ingénierie, pour formaliser des modèles semi-formels.

Nous effectuons une métamodélisation dans l'optique d'intégrer des modèles particuliers au niveau conceptuel. Nous utilisons donc la métamodélisation comme **technique de dialogue**, en décrivant les concepts des différents modèles, l'objectif étant de *« dépasser les problèmes terminologiques et les conventions graphiques afin d'identifier leurs points communs et ceux divergents »* [OUS97].

Pour cela, le métamodèle obtenu doit satisfaire les critères suivants [BRA95] :

- **Généricité**, il doit pouvoir s'appliquer à un ensemble d'applications le plus large possible ;
- **Unification sémantique**, il doit permettre la mise en correspondance de concepts équivalents dans différents modèles utilisant des représentations différentes ;
- **Indépendance de forme et contexte**, l'information du métamodèle ne doit pas être biaisée par le modèle de représentation et ne dépend pas des conditions d'implémentation.

L'étude des intersections entre modèles en vue de leur intégration est grandement facilitée par la métamodélisation. L'expression des concepts de modélisation dans un langage unique lève toute ambiguïté sur leur définition, ainsi les liens sémantiques et syntaxiques des différents modèles sont alors plus facilement mis à jour. De même l'intégration forte par les données qui sont exprimées dans un même langage de modélisation, devient possible. La figure 3.2-2 reprend les différentes étapes d'intégration des modèles par métamodélisation. L'identification d'éventuelles incohérences et des points communs est effectuée après métamodélisation.

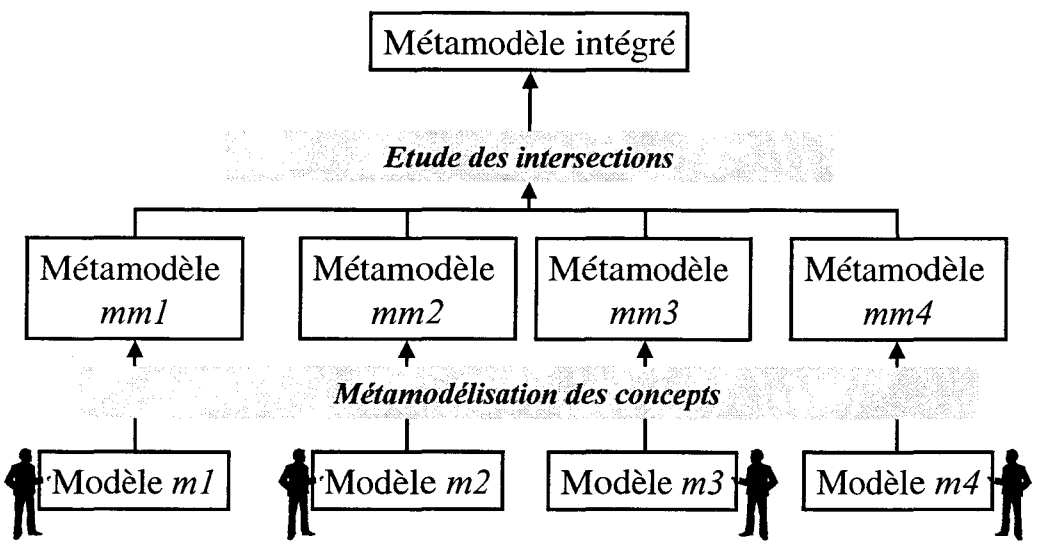



**Figure 3.2-2 : Processus de métamodélisation**

### 3.2.2. Etat de l'art sur la métamodélisation

Ces dix dernières années, plusieurs travaux sur l'intégration d'outils CASE (Computer Aided Software Engineering), d'AGL (Ateliers de Génie Logiciel) et de systèmes divers, sont basés sur le principe de métamodélisation. L'approche TSER (Two Stage Entity Relationship) propose des moyens pour intégrer les différentes bases de données de l'entreprise grâce à un métamodèle [HSU93]. Le projet MENINGE (MEta-modélisatioN pour les projets d'INGEnierie) du laboratoire Logiciels-Systèmes-Réseaux (LSR-IMAG, Institut d'Informatique et de Mathématiques Appliquées de Grenoble) propose un noyau de métamodélisation permettant une spécification et une implantation dans un formalisme unique de différents modèles [BOU96].
Deux approches ont fait l'objet d'une normalisation : le référentiel IRDS et les standards CDIF.

#### 3.2.2.1. La norme IRDS

L'IRD (Information Resource Dictionary) est un référentiel partageable pour la définition des ressources d'informations pertinentes de l'entreprise. La création, la manipulation et la mise à jour d'un référentiel IRD, s'effectue grâce aux moyens fournis par la norme IRDS[3] (IRD System), dont l'architecture est normalisée ISO/IEC 10027:1990. La norme IRDS permet l'intégration d'outils logiciels qui supportent l'analyse et la conception des processus. Elle est typiquement utilisée comme référentiel pour l'analyse et la conception de l'information.
Le référentiel IRDS repose sur une architecture à quatre niveaux hiérarchiques [MAR91]. Pour chaque paire de niveaux consécutifs, le niveau supérieur définit les types d'objets qui peuvent être contenus dans le niveau inférieur, figure 3.2-3.

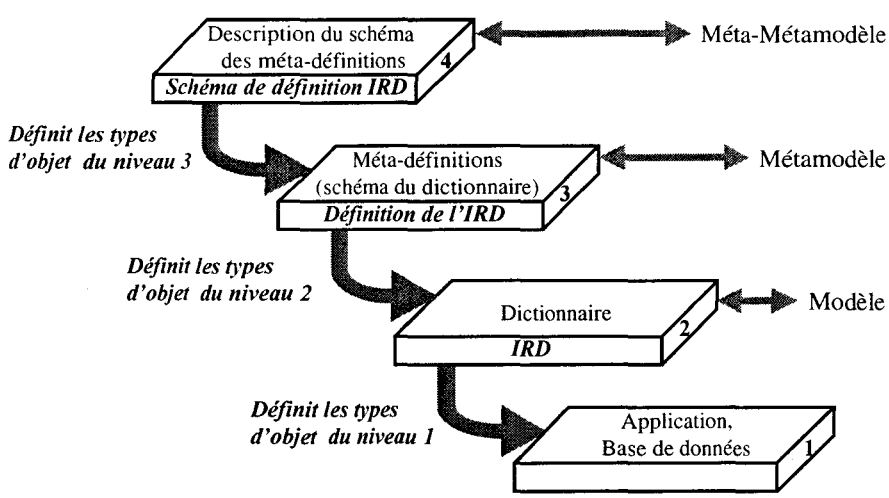



**Figure 3.2-3 : Les quatre couches du référentiel IRDS**

[3] les informations mises à jour sur les travaux de normalisation IRDS sont disponibles sur le site http://www.irds.org

Le niveau 1 (Application Level) contient les objets du monde réel, il est situé hors du référentiel. Le niveau 2 (IRD Level) est le dictionnaire des objets du niveau 1. Le référentiel offre à ce niveau des services de création d'objets, de maintenance des définitions, d'interrogations et d'accès par vues. Le niveau 3 (IRD Definition Level) est le schéma du dictionnaire (méta-dictionnaire). Le référentiel offre les mêmes services qu'au niveau 2, mais ils agissent sur des définitions qui structurent le dictionnaire du niveau 2. Le niveau 4 (IRD Definition Schema Level) définit les concepts utilisés pour la définition du dictionnaire.

Les concepts utilisés sont de plus en plus abstraits en allant du niveau 1 au niveau 4. En terme de modèles, le passage d'un niveau donné (1, 2 ou 3) à un niveau supérieur se fait par modélisation des concepts manipulés dans le niveau inférieur. Partant des données du monde réel (niveau 1), une première abstraction produit un ensemble de modèles au niveau 2, qui correspond donc au niveau modèle. En appliquant le même principe, les niveaux 3 et 4 vont correspondre respectivement au « metamodèle » et au « meta-metamodèle ».

#### 3.2.2.2. CDIF

Pour assurer l'échange d'informations entre des outils CASE, l'EIA[4] définit une famille de normes : CDIF (Case Data Interchange Format). CDIF fait l'objet de travaux de normalisation à travers le projet ISO/IEC JTC1/SC7/WG11, en abordant trois aspects, l'architecture (Framework for modeling and extensibility), le métamodèle intégré (Integrated Meta-model) et le format de transfert (Transfer Format), figure 3.2-4.

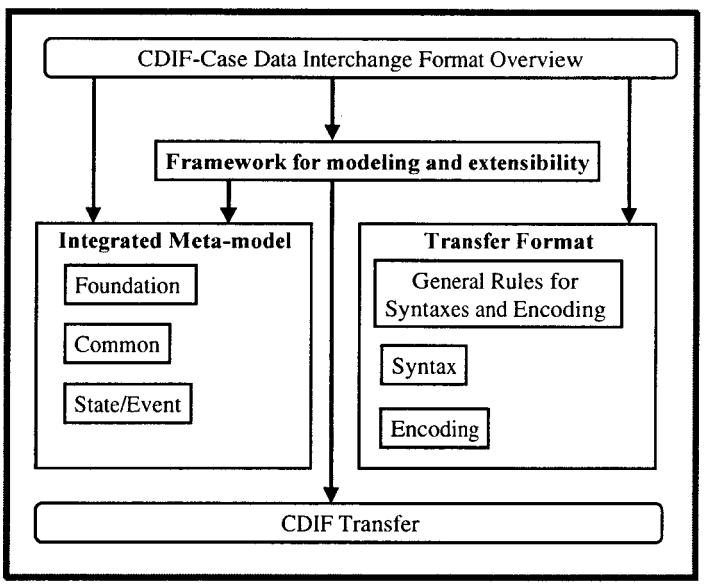



**Figure 3.2-4 : Cadre CDIF**

[4] Electronic Industries Association, http://www.eigroup.org/cdif/online.html

CDIF dispose d'une architecture en quatre couches (Donnée, Modèle, Métamodèle, Méta-métamodèle), à l'image de celle du référentiel IRDS. Cette architecture définit les relations entre l'information représentée dans un transfert CDIF (CDIF transfer), la sémantique de cette information et les règles d'extension du métamodèle intégré.

CDIF fournit un format de transfert indépendant de toute architecture informatique liée au transfert. Il est défini comme la combinaison d'une syntaxe (grammaire et structure du format d'échange) et d'un encodage (représentation informatique) qui respectent des règles générales.

Le métamodèle intégré définit les données qui peuvent être échangées selon CDIF et la sémantique de ces données. L'aspect fondamental de l'approche CDIF réside dans la séparation entre la sémantique de l'information transférée et sa représentation. Le métamodèle intégré de CDIF est divisé en domaines d'études (Subject Areas), chaque domaine d'étude (foundation, common, state/event) correspond à un métamodèle.

### 3.2.3. L'architecture du référentiel

Comme nous venons de le voir, la plupart des architectures de référentiel sont constituées de quatre niveaux. La finalité du référentiel est de pouvoir gérer un ensemble de données multi-facettes, multi-disciplines, dans une base de données commune. Nous avons pour cela défini une architecture suivant un axe horizontal et un axe vertical qui comporte trois niveaux [BIG00], [NDI00a] figure 3.2-5.

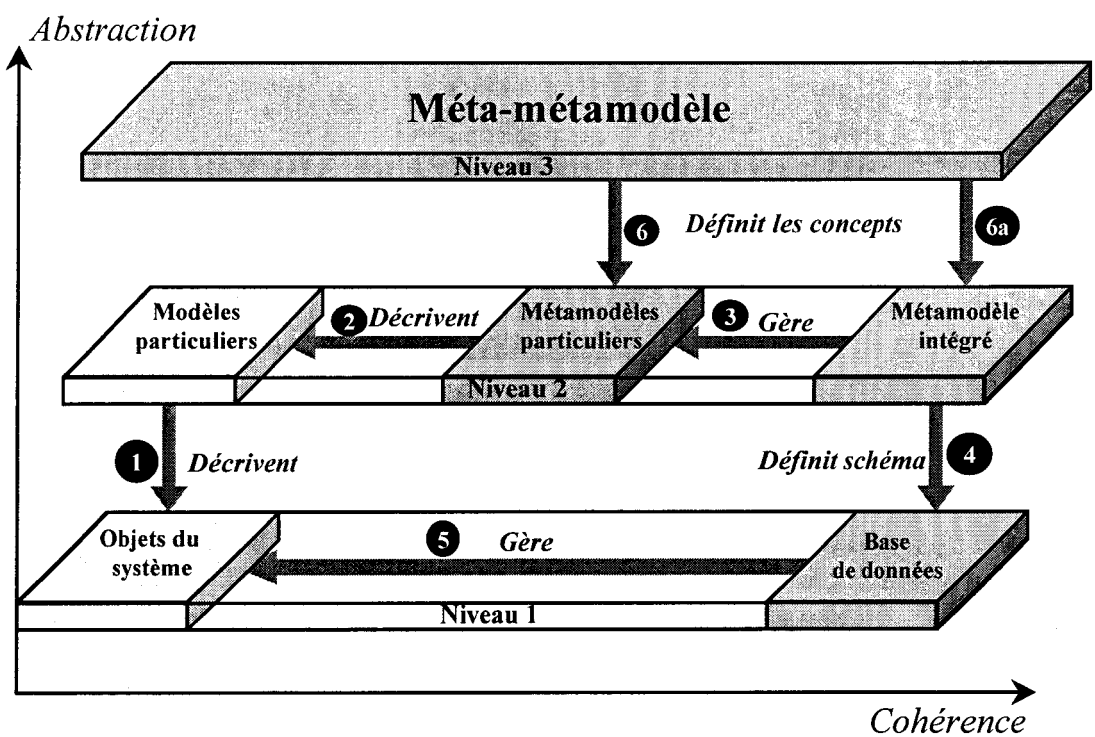



**Figure 3.2-5 : Architecture du référentiel**

L'axe vertical exprime les différents niveaux d'abstraction des concepts de modélisation. Il va du niveau 1 qui comprend les objets réels et l'implantation de ces objets dans une base de données, au niveau 3 qui est le plus abstrait et définit les concepts et formalismes du langage utilisé pour la métamodélisation au niveau 2.

L'axe horizontal exprime la structuration interne de chaque niveau, qui contient des éléments de même degré d'abstraction. Cet axe mesure la cohérence et l'intégration de ces éléments.

➢ Le **niveau 1** correspond aux données de base et aux objets du monde réel, par exemple les constituants physiques d'un système, les données d'une simulation etc.
Suivant l'axe horizontal nous distinguons deux degrés de cohérence : les objets réels qui fournissent diverses informations selon les points de vue, et la base de données unique chargée de gérer de manière cohérente la totalité de cette information (flèche 5).

➢ Au **niveau 2**, se trouvent les différents modèles qui permettent aux intervenants d'étudier le système, ces modèles particuliers décrivent séparément les objets réels du système (flèche 1). Ces modèles sont traduits dans un langage de modélisation unique afin de détecter leurs liens et les éventuelles incohérences (flèche 2), on obtient des métamodèles particuliers s'apprêtant mieux à une intégration. Pour avoir une perception globale et cohérente des métamodèles particuliers, ils sont intégrés au sein d'un métamodèle unique (flèche 3).
Les métamodèles particuliers et le métamodèle intégré doivent rester au même niveau que les modèles des intervenants car ils n'introduisent pas un degré d'abstraction supplémentaire mais proposent un langage unifié dans le but de faciliter l'intégration des modèles partiels. La traduction des différents concepts de modélisation dans un langage de modélisation unique permet de détecter les intersections entre les différents modèles et de préserver la cohérence de l'ensemble. Le métamodèle intégré permet de définir le schéma de la base de données (flèche 4).

➢ Le méta-métamodèle au **niveau 3** que nous avons défini dans [BIG99], permet la description et le contrôle du métamodèle intégré et des métamodèles particuliers. Il définit les concepts et formalismes du langage de modélisation utilisé pour traduire les modèles particuliers du niveau 2 (flèches 6 et 6a). C'est le niveau le plus abstrait, il forme un tout cohérent en définissant la sémantique de base de langage de métamodélisation.

Nous avons choisi le langage UML (§3.1.1) ; le niveau 3 est alors constitué d'un modèle UML réflexif dans la mesure où il définit d'autres concepts UML manipulés dans le métamodèle intégré. Le diagramme de classes sur la figure 3.2-6 présente un extrait simplifié du méta-métamodèle du référentiel.

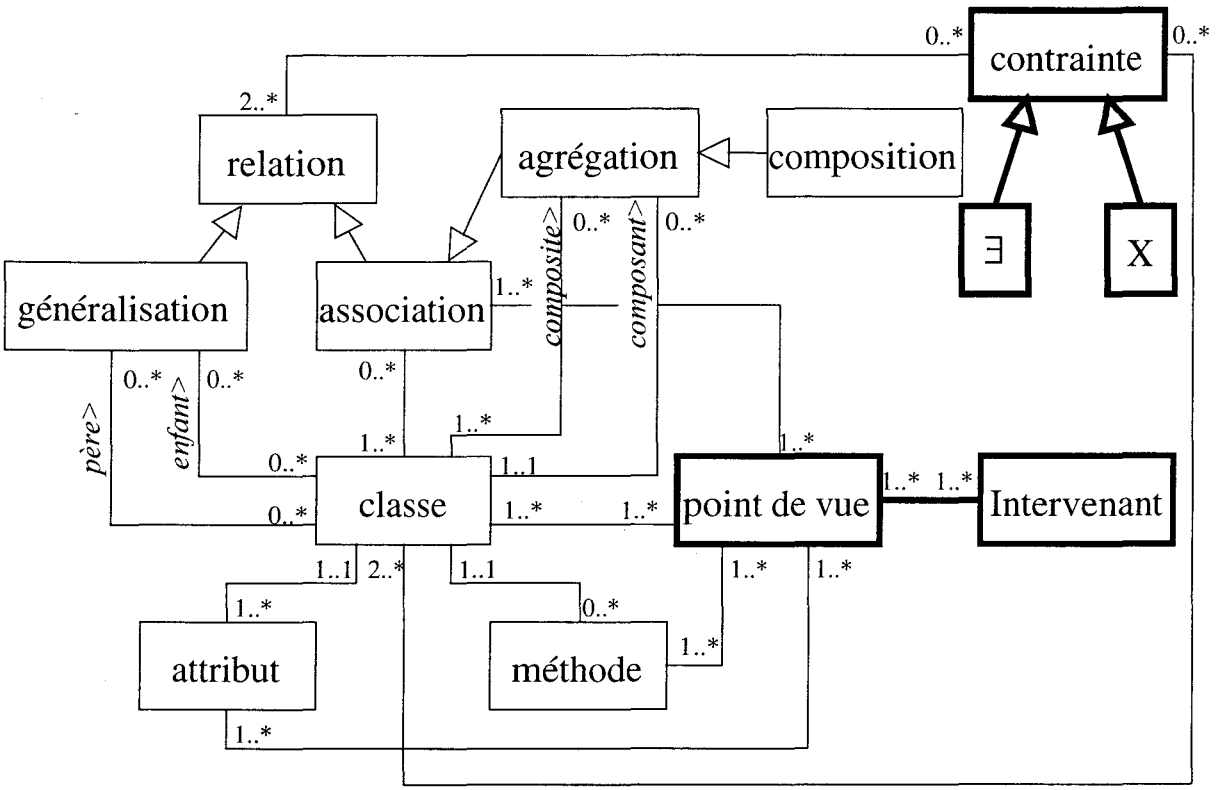



**Figure 3.2-6 : Extrait du Méta-métamodèle du référentiel**

Le niveau 3 prend en compte la gestion des points de vue qui peut concerner un à plusieurs intervenants. Chaque intervenant peut participer à l'élaboration d'un ou plusieurs points de vue. Chaque point de vue possède des autorisations d'accès et de modifications sur une donnée ; par exemple une classe attribut peut être créée ou mise à jour par un point de vue donné, lue par un autre et inutilisé par un troisième.

L'agrégation est une sorte d'association ; de même la composition est une forme particulière d'agrégation, d'où les relations d'héritage entre les classes association←agrégation←composition.

Nous avons proposé des extensions UML dans [BIG98] pour assurer une meilleure cohérence de la dynamique des liens entre objets : il s'agit des contraintes exclusive (X) et existentielle (∃).

### 3.2.4. Extension du langage UML : contraintes exclusive et existentielle

Les données que nous avons à gérer font fréquemment appel à une modélisation par objet composite : il s'agit d'un objet constitué par l'agrégation d'objets composants. Une étude détaillée sur la modélisation des objets composites est effectuée dans [OUS97].
Un objet composite est caractérisé par une sémantique riche, une structure souvent très complexe, et nécessite par conséquent une gestion particulière, notamment dans les opérations de mise à jour ou suppression de ses composants. C'est dans ce cadre que nous avons étendu UML par les contraintes exclusives et existentielles.

- **La contrainte exclusive :**

En UML, l'agrégation est une forme spéciale d'association qui spécifie une relation tout-partie entre l'agrégat (le tout) et les composants. La composition est une forme d'agrégation avec une forte possession et une durée de vie coïncidente entre le tout et les composants. Les composants peuvent être supprimés avant la mort du composite, et n'appartiennent au plus qu'à un seul composite (figure 3.2-7) ; la disparition du composite entraîne celle des composants.

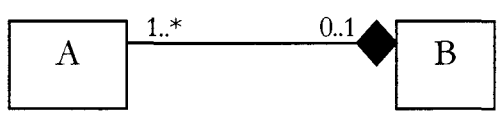



**Figure 3.2-7 : Relation de composition en UML.**

Dans le cadre du traçage d'information, il est important de gérer les relations dans le temps ; il sera en effet possible de savoir qu'une instance de la classe A a été composant d'une instance de la classe B entre deux dates, puis est devenue composant d'une autre instance de la classe B. Pour cela, nous ajoutons la classe association historique et utilisons l'agrégation (figure 3.2-8).

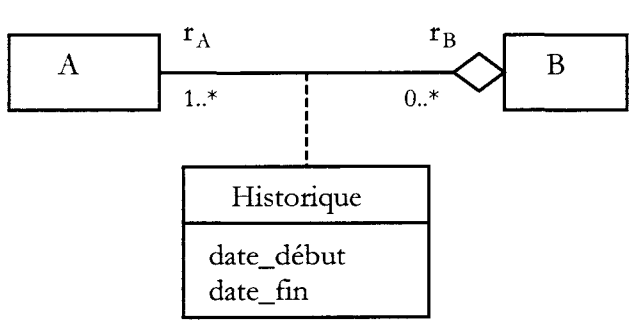



**Figure 3.2-8 : Gestion de l'historique en UML standard.**

Si nous voulons exprimer qu'une instance de la classe A n'est associée qu'à au plus une seule instance de la classe B à un instant donné, nous devons ajouter l'expression OCL ci-après, qui évite l'interférence de périodes dans l'historique.

```
Context Historique inv :
Self.date_fin > self.date_début
Self.historique->forAll (h1,h2:historique|h1<>h2 and h1.rA=h2.rA
                    implies h1.date_début>=h2.date_fin or h2.date_début>=
                              h1.date_fin)
```

L'extension que nous avons proposée permet d'exprimer qu'un composant n'appartient au plus qu'à un composite à un instant donné, mais qu'il peut appartenir à plusieurs composites pendant sa durée de vie. Cette extension est représentée figure 3.2-9 avec un stéréotype de dépendance (c'est-à dire qu'une modification de l'élément indépendant – ici A - va affecter l'élément dépendant – ici B. On notera la concision de la représentation, car la figure 3.2-9 équivaut à la figure 3.2-8 associée à l'expression OCL.

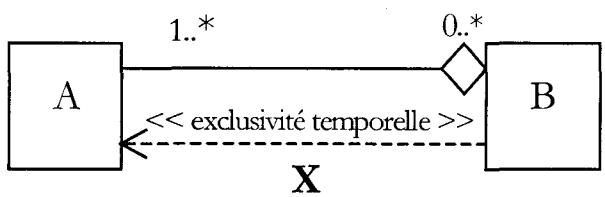



**Figure 3.2-9 : Contrainte exclusive.**

➢ **La contrainte existentielle :**

Avec UML, un composite n'est pas détruit lorsqu'on supprime un ou plusieurs de ses composants ; or nous avons parfois à exprimer une telle contrainte. Ceci a été développé dans [OUS97] par la notion de prédominance : l'existence d'un objet composite dépend de celle de ses composants.

C'est pourquoi nous avons proposé la contrainte existentielle comme extension à UML : la suppression d'une instance de la classe A entraîne celle d'une instance de la classe B. Ainsi la cohérence du référentiel est préservée lors d'une opération de suppression. La figure 3.2-10 représente la contrainte existentielle avec un stéréotype de dépendance.

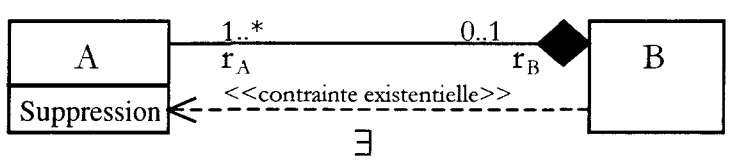



**Figure 3.2-10 : Contrainte existentielle.**

Ce formalisme évite d'avoir à préciser la contrainte, exprimée ci-après à l'aide d'OCL, qui signifie qu'après la suppression d'une instance de la classe A, l'intersection entre toute instance de la classe B qui lui était associée précédemment et les instances courantes de B est vide.

```
A:: suppression ()
Post: B.Allinstances -> intersection (self.rB@pre)->isempty
```

Face au nombre importants et variés de modèles utilisés dans la conception des systèmes complexes, une homogénéisation s'avère nécessaire pour le partage, l'échange d'informations. Nous avons vu que la métamodélisation est une technique qui convient particulièrement à cet effet, en effet elle permet de déceler plus facilement les liens entre objets. Par le biais de la métamodélisation nous parvenons à faire coopérer plus facilement différents modèles métiers, car l'expression dans un langage unique des informations échangées permet de ressortir plus facilement les points de convergence (concepts équivalents) et les contradictions entre ces modèles.

Le méta-métamodèle au niveau 3 du référentiel établit clairement les concepts utilisés pour la métamodélisation. Il est défini dans le manuel de référence sur la sémantique et la syntaxe UML.

Nous avons enrichi le métamodèle UML avec les classes contrainte (existentielle et exclusive) pour une meilleure expression de la dynamique des liens entre objets. Chaque point de vue possède un accès particulier sur les attributs, classes, associations et méthode. Nous offrons alors une meilleure flexibilité ainsi qu'une visibilité adéquate aux divers intervenants, leur permettant ainsi de manipuler les données qui leur sont pertinentes.

Au niveau 2 du référentiel, nous utilisons le langage UML pour traduire les différents modèles métiers préexistants. Cette traduction n'introduit aucune contrainte supplémentaire sur l'organisation des points de vue ni sur la démarche de conception des intervenants, préservant le principe d'indépendance évoqué au § 2.1. Les métamodèles particuliers seront en contrepartie étroitement liés aux différents modèles métiers utilisés car leur conception passe par une étude exhaustive de leurs concepts.

Pour faciliter et accélérer l'obtention des métamodèles particuliers (par conséquent celui du métamodèle intégré), nous proposons une démarche de réutilisation de solutions conceptuelles, en permettant une grande réutilisabilité des métamodèles obtenus. Pour ce, nous procédons par capitalisation de solutions conceptuelles à l'aide de patterns, et nous proposons une démarche de réutilisation dans le chapitre 4.

# CHAPITRE 4.

# Réutilisation de concepts de modélisation

## Résumé

**La réutilisation est un atout principal d'une approche orientée objet. Après avoir montré l'apport de la réutilisation dans notre démarche d'intégration, nous formalisons une démarche de réutilisation. Nous créons des patterns de conception et des patterns métiers qui seront par la suite réutilisés dans la construction des différents métamodèles UML.**

## 4.1. La réutilisation à l'aide de patterns

### 4.1.1. La réutilisation comme moyen d'aide à la conception

Plusieurs méthodologies dans le domaine de la conception reposent sur des mécanismes de réutilisation, par instanciation, adaptation etc. D'une manière générale, la réutilisation consiste à se servir d'un composant pour en créer un nouveau [MOR95a].

- Les cadres de références (§ 2.2, CIMOSA, BASE-PTA etc.) disposent de modèles de référence générique, qui sont réutilisés par instanciation et adaptation au système étudié. Mais comme nous l'avons souligné (§ 2.2.3.2) ce mécanisme de réutilisation prend difficilement en compte les modèles préexistants.
- La méthodologie KADS (Knowledge and Analysis Design Support, ESPRIT I N° 1098 ) connu maintenant sous le nom de CommonKADS, est destiné au développement de systèmes à base de connaissances, pour la réalisation de systèmes experts. Le modèle KADS permet la construction de modèles d'expertise génériques, qui servent de structure initiale lors de la construction de modèles spécifiques [MAR96].
- Le projet MENINGE travaille sur la spécification et le développement d'un environnement permettant aux organisations de spécifier, d'implanter puis d'utiliser des modèles et des méthodes. Il propose une conception assistée, en favorisant la réutilisation de modèles existants lors de l'élaboration de nouveaux modèles [BOU96].
- Le projet POSEIDON (Patrons d'ObjetS pour les Echanges Industriels de DONnées) initié en 1997 par le programme de recherche PROSPER du CNRS, porte sur l'étude d'un Système d'Information Produit (SIP) qui gère la base documentaire accompagnant le développement des produits, ainsi que la répartition des tâches du processus de développement entre acteurs, etc. La méthode utilisée dans POSEIDON permet une capitalisation et une réutilisation au maximum des concepts déjà rencontrés, afin d'accélérer la spécification du SIP [TOL00].

D'une manière générale toutes ces méthodologies proposent des systèmes génériques d'aide à la conception, par le biais de mécanismes de réutilisation. Cependant, aucune méthodologie ne fournit des modèles qui couvrent totalement les systèmes d'information du fait de leur complexité, au risque de fournir *« des modèles trop génériques pour être d'une part accessibles aux utilisateurs chargés de leur validation et d'autre part facilement adaptés à un domaine particulier »* [TOL00].

### 4.1.2. Une réutilisation basée sur des concepts objet

Dans une approche orientée objet, la réutilisation se manifeste à travers des mécanismes tels que l'héritage et la composition etc. C'est une propriété très utilisée dans la communauté objet, [MOR95a] pensant même qu'il serait *« anormal de ne pas réutiliser »*.

Nous rappelons que UML est un langage de modélisation, il faut par conséquent lui associer une méthode pour définir une démarche reproductible pour la création des métamodèles particuliers. Des méthodes de conception systémiques comme Merise permettent l'obtention d'un modèle entité/association normalisé qui offre l'avantage de faciliter la communication visuelle et de favoriser l'expression des règles de gestion. Néanmoins, la séparation entre les données et les traitements pose problème dans un domaine où la dynamique n'est pas dissociable des données [BIG99a]. De plus, ces méthodes présentent des faiblesses en conception logicielle [TOL00].

La réutilisabilité d'un objet ou concept est une propriété très intéressante, formalisée en génie logiciel sous la notion de Composant Logiciel Réutilisable (CLR). Un CLR est un modèle indépendant, élémentaire pouvant être réutilisé afin de réduire le coût de développement, améliorer la qualité et accélérer le développement. Dans [COU96] nous trouvons une étude comparative des différentes techniques de réutilisation (la duplication, le préprocessing, les bibliothèques, génération de code etc.) suivant plusieurs critères (la productivité, la maintenabilité, la fiabilité, l'évolutivité, la facilité d'utilisation, l'adaptabilité).

Cependant ces techniques concernent la réutilisation de composants implémentés, nous considérons ce niveau de réutilisation « bas » (niveau application, niveau 1 du référentiel), par ailleurs les composants réutilisés sont souvent élémentaires, en effet la réutilisation concerne directement les objets (classe, type, attribut etc.).

Notre objectif n'est pas de fournir des modèles génériques à partir desquels la totalité des modèles du système sont construits, mais de capitaliser des modèles, pouvant être réutilisé par le concepteur du référentiel principalement au niveau 2 de l'architecture.

Nous proposons de mettre en place une stratégie favorisant la capitalisation et la réutilisation de composants conceptuels, pour faciliter la construction de nouveaux métamodèles particuliers. Ces composants réutilisables doivent :

- d'une part, offrir une granularité plus importante, en proposant des blocs de réutilisation plus conséquents ;
- d'autre part, se situer dans un niveau d'abstraction plus élevé, la réutilisation s'effectue ainsi au niveau 2 du référentiel, en étudiant les concepts des différents modèles.

Pour réaliser la capitalisation et la réutilisation de composants conceptuels, nous pensons qu'il est judicieux d'utiliser le concept de pattern, développé dans le domaine du génie logiciel et considéré comme l'unité de réutilisation dans la conception objet.

### 4.1.3. Apport des patterns

Le dictionnaire Le Robert édition 1994 donne la définition suivante du mot Pattern :

> ***Pattern*** : mot anglais signifiant modèle schématique.
>
> Modèle simplifié de structure.
>
> Synonymes : patron, schéma, structure, type.

La notion de pattern issue du domaine de l'architecture dans les années 70, est largement répandue dans le cadre de l'ingénierie logicielle (dans les approches objet en particulier) et tend à se généraliser dans d'autres domaines tels que l'ingénierie des systèmes d'information. Un pattern (ou patron) est défini par [COA96] comme *« une forme entièrement réalisée, originale ou un modèle accepté ou proposé pour une imitation : quelque chose qui est vu comme un exemple normatif pouvant être copié, archétypé ou utilisé comme exemple »*. Nous trouvons dans la littérature [BUS96], [COA96], [COP95], [GAM95], [FOW96], plusieurs définitions sur les patterns[5] qui se recoupent. D'une manière générale, les patterns offrent des **solutions** éprouvées à des **problèmes** récurrents dans un même **domaine** ou dans des domaines différents. Ainsi, les patterns capturent le savoir que les experts appliquent pour résoudre des problèmes récurrents. Par analogie, *« un pattern est pour les objets, l'équivalent du sous-programme pour les instructions, ou encore des circuits intégrés pour les transistors »* [MUL00].

Dans nos travaux, les patterns sont des modèles (diagrammes de classes UML) participant à la construction d'autres modèles (métamodèles particuliers). *« Il s'agit ainsi de proposer des artefacts prédéfinis et adaptables à des problèmes similaires et à des technologies différentes d'implantation »* [TOL00]. Les principaux avantages des patterns fréquemment cités sont les suivants : (1) ils proviennent d'expériences sur de bons savoir-faire et n'ont pas été créés artificiellement ; (2) ils sont un moyen de documentation et d'archivage rigoureux et systématique, d'un ensemble de connaissances ; (3) ils facilitent la communication entre les intervenants d'un projet (dans le cas de développement en équipe), en proposant des modèles compréhensibles de tous ; (4) ils permettent de guider une activité d'ingénierie en organisant hiérarchiquement et fonctionnellement les problèmes et leurs solutions.

---

[5] Le site http://www.csioo.com/cetusfr/oo_patterns.html contient des informations très intéressantes sur les patterns.

Notre démarche de réutilisation est mise en oeuvre en supposant que dans tout système, il existe des **invariants**, c'est à dire des concepts récurrents qui ne bougeront pas ou peu dans le temps. A travers les patterns, nous proposons des solutions au problème de modélisation de ces invariants, solutions qui ne constituent en aucun cas une prescription. Les nouveaux modèles seront alors construits par réutilisation (extension, adaptation etc.) totale ou partielle des patterns créés. Les patterns sont des invariants obtenus par une étude des concepts de modélisation essentiels, ce sont des éléments qui rassemblent, car ils permettent de poser les bases sur lesquelles tous les points de vue pourront s'accorder, en vue de la construction des métamodèles particuliers, étape préalable à l'obtention du métamodèle intégré.

## 4.2. Mécanismes de réutilisation

Nous mettons en oeuvre une démarche de réutilisation de patterns pour assister le processus de métamodélisation des modèles particuliers du niveau 2 du référentiel [NDI00b], [NDI00c]. Le métamodèle intégré du référentiel sera construit autour d'invariants clairement identifiés. Les patterns contribuent ainsi à l'intégration de points de vue dans la mesure où ils interviennent dans une étape importante de notre démarche d'intégration : la métamodélisation des différents concepts de modélisation.

La réutilisation de patterns n'a pas pour seul intérêt d'accélérer le processus de métamodélisation, car en réutilisant des modèles testés et validés, on améliore considérablement la fiabilité, l'évolutivité et l'adaptabilité du métamodèle intégré.

La modélisation de systèmes complexes induit la manipulation d'un nombre élevé d'éléments de modélisation. Cela nous a amené à organiser les éléments du niveau 2 du référentiel dans des paquetages (§ 3.1.2.4) d'une part pour appréhender la complexité du SI, d'autre part pour offrir un bon niveau de granularité à la réutilisation.

En terme d'architecture, les paquetages offrent plusieurs avantages [KET98] :

- Tout d'abord, les paquetages constituent un excellent moyen de **structuration** des **modèles** du système. Leur organisation montre tout de suite la structure générale du système, et constitue une perspective de haut niveau nécessaire pour appréhender les systèmes complexes.
- Ensuite, ils permettent d'organiser les différents points de vues d'un système.
- Enfin ils sont définis dans UML, comme des éléments de gestion de configurations, de stockage et de contrôle d'accès. En effet, les paquetages ne sont pas de simples vues, mais il sont propriétaires d'éléments.

Nous définissons les trois paquetages suivants au niveau 2 du référentiel : « patterns », « points de vue » et « métamodèle intégré », figure 4.2-1 [NDI00c]. Chacun de ces paquetages contient des diagrammes de classes regroupés dans d'autres paquetages. Nous pouvons ainsi considérer deux niveaux d'imbrication des paquetages : la première constituée des trois paquetages cités précédemment, et la deuxième constituée des paquetages qu'ils contiennent. Les paquetages d'un niveau n'utiliseront que ceux du même niveau ou de niveau inférieur.

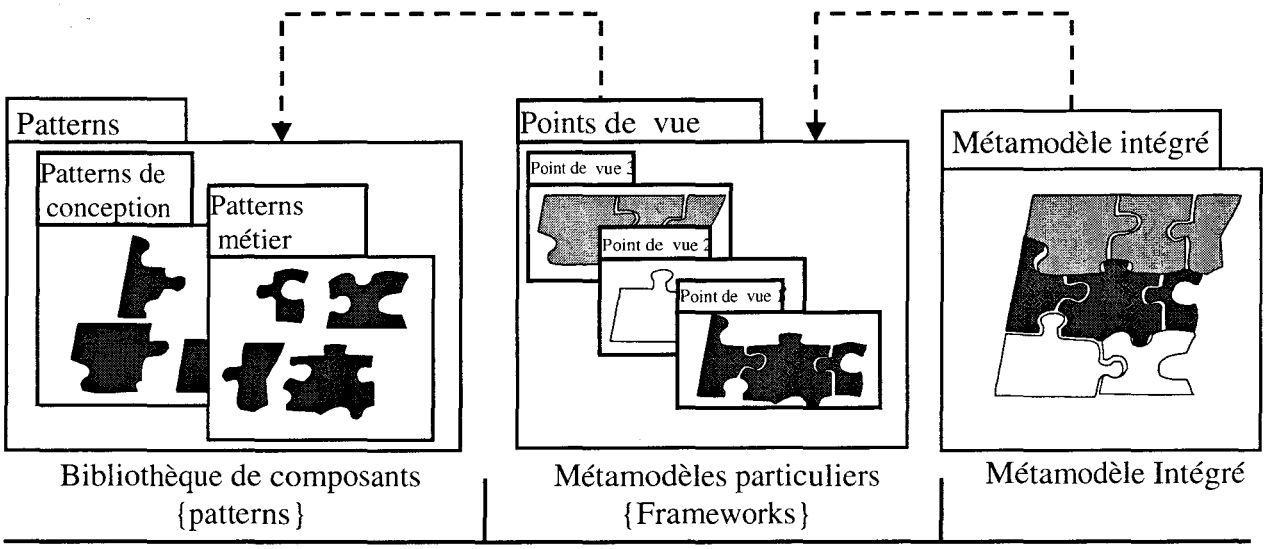



**Figure 4.2-1 : Les paquetages au niveau 2 du référentiel**

## 4.2.1. Le paquetage « Patterns »

Ce paquetage contient le premier niveau de réutilisation, il est constitué de deux paquetages : « Patterns de conception » et « Patterns métier ». Les éléments de ces paquetages peuvent être réutilisés lors de la construction d'éléments du paquetage « Points de vue ».

### 4.2.1.1. Les patterns de conception

L'analyse, la conception et l'implémentation constituent les trois principales phases de développement dans une approche orientée objet. Des patterns ont été proposés pour chaque phase. Pour la conception, les catalogues de pattern les plus connus, ont été publiés par [GAM95] et 'Siemens Group' [BUS96].

Les patterns de conception (Design Patterns) proposés par [GAM95] sont de loin les plus connus et les plus utilisés. Un pattern de conception donne un nom, identifie les principes fondamentaux d'une structure générale, pour en faire un moyen utile à l'élaboration d'une conception orientée objet réutilisable.

Les patterns de conception décrivent des objets coopératifs et des classes spécialisées pour résoudre un problème général de conception, dans un contexte particulier.

Le paquetage « Patterns » contient les patterns de conception que nous définissons après étude des concepts récurrents utilisés dans les différents modèles des intervenants. Les modèles des intervenants sont très souvent des modèles descriptifs utilisant des concepts basés sur la théorie des graphes, [GON95].

Nous avons ainsi proposé un ensemble de patterns de conception permettant la description des concepts utilisés dans la plupart des modèles graphiques rencontrés en génie industriel (Réseaux de Petri, Statecharts, Grafcet, modèles de comportement, fonctionnels, structurels etc.) [NDI00a], [NDI00b], [NDI00c]. L'étude de ces patterns est effectuée au § 4.3.

Les patterns de conception sont des diagrammes de classes génériques, indépendants de tout domaine d'application. Ils sont donc utilisables par instanciation, ensuite ils peuvent être adaptés et particularisés.

**Apport des patterns de conception dans la démarche d'intégration**

La contribution des patterns de conception est déterminante lors de la phase métamodélisation, qui est une étape essentielle dans notre démarche d'intégration.

#### 4.2.1.2. Les Patterns métier

L'étude de systèmes se rapportant à une discipline donnée ou un domaine particulier, fait appel à un ensemble de modèles type. Ces modèles définissent un domaine partageable et utilisable par les différents métiers. Par l'exemple, l'étude des SAP dans le cadre des systèmes à événements discrets, fait appel à des modèles de base qui s'intéressent souvent aux aspects structurels, fonctionnels, comportementaux, opérationnels, etc. des objets du SAP.

Dans le cadre du génie logiciel, pour décrire un domaine particulier et en donner une architecture adaptée, les patterns métier (Business patterns) également dénommés patterns de domaine ou patterns d'analyse ont été développés [FOW96]. Contrairement aux patterns de conception, les patterns métier sont liés à un domaine d'application particulier. Les patterns métier sont obtenus par analyse d'un domaine d'expertise, dont ils représentent les invariants pertinents.

Nous avons adopté cette de notion de patterns métier, pour capitaliser les concepts de modélisation récurrents liés au domaine d'application. Un pattern métier définit un modèle récurrent pouvant être utilisé par plusieurs points de vue, dans un secteur d'activité. Les patterns métiers sont construits en utilisant au mieux les patterns de conception.

**Apport des patterns métiers dans l'intégration des métamodèles :**

Les métamodèles des points de vue (ou des différentes d'un point de vue) sont établis séparément. Au moment de les intégrer dans un diagramme de classes unique, il faudrait étudier toutes les classes et associations afin de déceler les points de convergence et les incohérences.

Pour un modèle comportant peu de classes cette étude est possible, mais dans la plupart des cas le nombre de classes et associations est assez important pour qu'une telle étude soit réalisable dans des délais raisonnables.

Nous proposons une méthode qui permet d'accélérer et le processus d'intégration des différents métamodèles en utilisant les patterns métier et les **diagrammes de cas d'utilisation** UML. Ces diagrammes formalisés par [JAC92], décrivent les fonctionnalités d'un système en examinant les besoins fonctionnels de chaque acteur.
Les cas d'utilisation sont représentés par des ellipses à l'intérieur desquelles figure le nom du cas d'utilisation. La relation entre un acteur et un cas d'utilisation est une association représentée par une ligne. Nous avons le plus souvent une relation d'utilisation entre les cas d'utilisation lorsque la construction d'un modèle s'appuie sur la définition d'un autre modèle.
Les cas d'utilisation correspondent aux modèles que les acteurs doivent définir.

L'exemple sur la figure 4.2-2 montre trois acteurs qui doivent concevoir trois modèles. La conception du Modèle_x s'appuie sur le Modèle_y, d'où la relation d'utilisation. Certains de ces modèles correspondent à des patterns métier (Modèle_z et Modèle_y par exemple) qui rappelons le, définissent des modèles récurrents pouvant être utilisés par plusieurs intervenants.

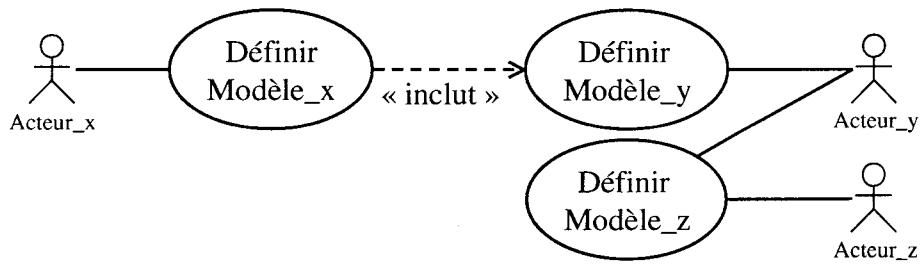



**Figure 4.2-2 : exemple de cas d'utilisation**

En exprimant de manière structurée le besoin de chaque intervenant en termes de modèles ainsi que leurs interactions, nous proposons une démarche qui assiste l'intégration des métamodèles, et nous réduisons considérablement la complexité du nombre de comparaison à effectuer, ce qui va nous éviter d'être noyé dans une masse d'informations au moment d'intégrer ces modèles.
Les patterns métiers contribuent de manière significative dans notre démarche d'intégration :

- d'une part en facilitant le processus de construction des métamodèles ;
- d'autre part en indiquant des « pistes » pour l'intégration des métamodèles.

### 4.2.2. Le paquetage « Points de vue »

La première étape de notre démarche d'intégration consiste à métamodéliser les modèles particuliers des différents point de vue en UML. L'établissement de ces métamodèles doit utiliser au mieux les patterns de conception et les patterns métiers.
Chaque point de vue correspond à un ensemble de métamodèles particuliers, et devient à son tour un modèle réutilisable, mais cette fois dans un domaine d'application particulier, le degré d'abstraction est moins élevé. Les points de vue sont représentés par un nombre de classes plus important. Pour faciliter la compréhension du métamodèle intégré (constitué d'un ensemble de points de vue), il nous faut alors introduire des concepts structurants plus larges.

En génie logiciel, la notion de **framework** modélise un ensemble de classes qui coopèrent et permettent des conceptions réutilisables dans des catégories spécifiques de logiciels. Un framework permet de réaliser plus rapidement une famille d'application dans un domaine donné, en fournissant une architecture générique [GAM95]. Pour pouvoir s'appliquer à plusieurs applications, les frameworks doivent être flexibles et facilement adaptables.

Les métamodèles des points de vue regroupent plusieurs métamodèles particuliers, donc un nombre plus importants de patterns. Nous adaptons alors du génie logiciel la notion de framework pour représenter le métamodèle de chaque point de vue. Un point de vue étant lié à un domaine d'expertise, le concept de framework semble satisfaire les besoins pour la modélisation et l'implantation de ces derniers.
Les frameworks, tout comme les patterns, sont des moyens de réutilisation mais il existe des différences essentielles [GAM95] :

- Les patterns sont plus abstraits, car les frameworks permettent de capitaliser la connaissance dans un domaine particulier ;
- les frameworks sont d'une granularité supérieure, ils sont composés de plusieurs patterns, l'inverse n'étant jamais vrai.

Dès lors que tous les métamodèles sont créés pour l'ensemble des points de vue, nous procédons à l'intégration des diagrammes de classes obtenus au sein du métamodèle intégré.

### 4.2.3. Le paquetage « Métamodèle intégré »

Le métamodèle intégré est un schéma conceptuel global complet, cohérent et non redondant. Il est obtenu par réutilisation des différents diagrammes de classes (paquetages) des points de vue. Les difficultés à surmonter lors de l'intégration de ces modèles, proviennent des recouvrements et des liens syntaxiques entre les diagrammes de classes. Parmi les problèmes les plus fréquents, nous pouvons citer : la synonymie, l'homonymie, la polysémie, les problèmes de cardinalité. La figure 4.2-3 propose une liste des conflits les plus courants [GAR99]. Le schéma conceptuel global sert ensuite de support pour la création du schéma logique de la base de données sous forme de tables puisque nous effectuons une implémentation dans un SGBD relationnel.

| |
|---|
| **Classe ↔Classe** |
| Noms différents pour des classe équivalentes |
| Noms identiques pour des classe différentes |
| Inclusion de l'une dans l'autre |
| Intersection non vide |
| Contraintes entre instances |
| **Attribut↔Attribut** |
| Noms différents pour des attributs équivalents |
| Noms identiques pour des attributs différents |
| Types différents |
| Compositions différentes |
| **Classe↔Attribut** |
| Significations identiques |
| Compositions similaires |
| **Association↔Association** |
| Noms différents pour des associations équivalentes |
| Noms identiques pour des associations différentes |
| Cardinalités différentes |

**Figure 4.2-3 : Les principaux conflits d'intégration**

Le métamodèle intégré est un diagramme de classes construit à l'aide de patterns de conception, de patterns métiers et de classes qui interagissent. Il devient alors un modèle réutilisable lié au domaine d'activité : c'est un framework qui constitue le dernier niveau de réutilisation avec un faible degré d'abstraction. Ce framework assure la cohérence globale de l'information et fournit le socle sur lequel toute une famille d'applications pourra être élaborée.

### 4.2.4. Formalisation d'une démarche pragmatique de capitalisation-réutilisation

D'une manière générale, la plupart des composants réutilisables sont fournis sans documentation ni guide dans la démarche de réutilisation. Pour pallier ce manque, nous avons défini des patterns qui sont tous documentés, et nous avons formalisé une démarche pragmatique d'enrichissement progressive du référentiel et de réutilisation des patterns et frameworks. Les données initiales de notre démarche d'intégration sont les suivantes :

- l'objet de l'étude (machine, cellule de production, système de production etc.) ;
- les différents points de vue et modèles définis par les intervenants du projet ;
- les patterns de domaine issus d'expériences antérieures, et les patterns de conception issus de l'étude des concepts de modélisation des différents modèles.

A partir de cette situation initiale, nous mettons en œuvre une démarche d'intégration incrémentale et récursive qui permet d'enrichir au fur et à mesure le noyau de composants réutilisables du référentiel, par capitalisation figure 4.2-4.

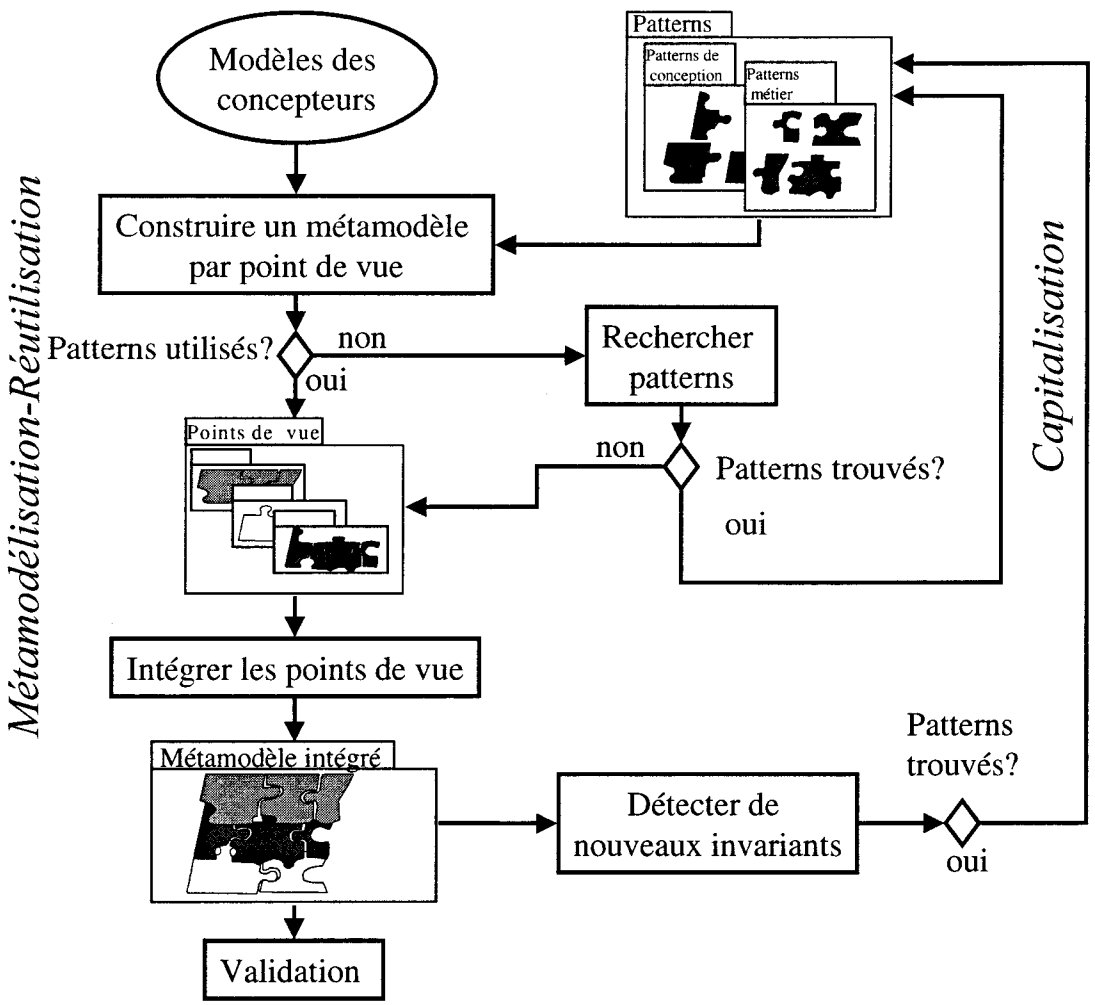



**Figure 4.2-4 : Démarche d'intégration**

C'est une démarche qui permet d'intégrer différents modèles par métamodélisation. Pour effectuer cette métamodélisation, nous disposons d'un noyau de composants réutilisables : les patterns de conception et les patterns métiers. Si toutefois les éléments de ce noyau n'étaient pas suffisamment réutilisés, nous lançons une recherche de nouveaux patterns.

La réutilisation n'est pas systématique, il est donc possible de créer le métamodèle pour un point de vue sans pour autant réutiliser un pattern.

Il y a ainsi trois niveaux de réutilisation, figure 4.2-5 :

- Le premier offert par le paquetage « patterns », qui contient une bibliothèque de composants réutilisables (patterns de conception et patterns métier) utilisés pour la construction des métamodèles de points de vue.
- Le deuxième constitué des éléments (frameworks) du paquetage « point de vue », qui à leur tour sont réutilisés pour former le métamodèle intégré.
- Le troisième et dernier correspond au métamodèle intégré qui est réutilisable pour une classe d'applications donnée.

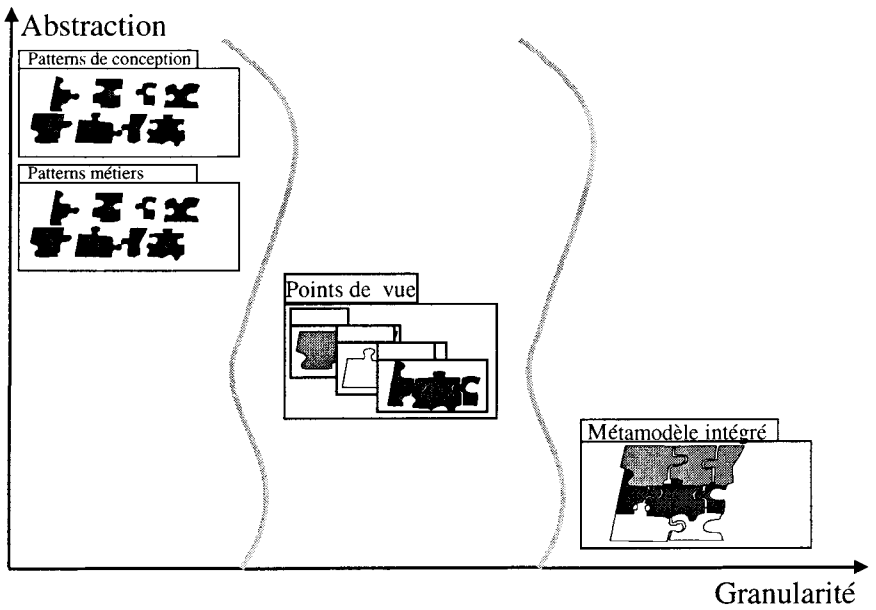



**Figure 4.2-5 : Niveaux de granularité et d'abstraction de la réutilisation**

Il se peut que certains invariants ne soient pas détectés au cours de la conception. Mais étant donné que le métamodèle intégré constitue un bloc important, il peut favoriser l'apparition de « motifs » et « invariants de modélisation » dans le domaine étudié. Dans ce cas nous enrichissons les paquetages des patterns en vue d'une prochaine réutilisation.

Le diagramme de classe final est alors traduit en modèle relationnel puis implanté dans une base de données ORACLE. Pour cela il faut d'abord traduire les diagrammes de classes en modèle relationnel.

## 4.3. Des patterns basés sur la théorie des graphes

Les patterns que nous allons proposés sont des patterns **structurels,** qui montrent comment un ensemble de classes et d'objets se rassemblent pour former des structures plus importantes, de la même façon que les objets du système étudié fournissent des informations qui sont agrégées par les différents concepteurs pour créer leurs propres modèles.

Nous nous intéressons aux modèles descriptifs, qui sont basés sur la théorie des graphes. *« Le langage des graphes permet de représenter simplement la structure d'un grand nombre de situations »* [GON95]. La théorie des graphes est initialement une discipline mathématique qui s'est par la suite répandue dans des sciences diverses telles que l'électronique (théorie des réseaux électriques), la chimie (modélisation de structures), l'automatique (schémas blocs, Bond-Graph), l'informatique (SADT : Structured Analysis and Design Technique, graphe des dépendances fonctionnelles…).

En productique, ils existe plusieurs outils basés sur la théorie des graphes, qui constitue l'un des instruments les plus efficaces du fait de sa capacité à cerner des systèmes complexes. Parmi les outils de modélisation les plus courants, nous pouvons notamment citer les réseaux de Petri, le Grafcet, les Statecharts (appelé aussi diagramme de HAREL), les diagrammes de Gantt, les Graphes d'état, les réseaux PERT etc.

Tous ces modèles sont construits selon des concepts et formalismes clairement définis. Les formalismes sont très différents d'un modèle à un autre, par contre les concepts de base sont tous établis à partir de la théorie des graphes. Un fait immuable est que tous les graphes sont constitués de **sommets** (appelés aussi nœuds) reliés par des **arcs (ou arêtes)**. Cela nous a amené à penser qu'il y a sûrement des invariants dans les concepts de modélisation, même si ces modèles sont souvent particularisés, adaptés ou étendus selon les besoins du concepteur.

Les patterns que nous allons proposer vont permettre de métamodéliser en UML les briques de base de ces différents modèles, briques qui pourront par la suite entrer dans la conception des autres métamodèles UML. Nous allons successivement étudier les patterns de conception suivants : Graphe orienté, Graphe orienté biparti, Arborescence. Puis nous établirons les patterns métiers : modèle structurel, modèle fonctionnel, et modèle de comportement.

Nous allons d'abord donner la définition de quelques notions sur la théorie des graphes, utilisées dans la suite du mémoire.

- **Adjacence :**

Deux sommets sont adjacents s'ils sont joints par un arc.

Deux arcs sont adjacents s'ils ont au moins une extrémité commune.

➢ **Graphe partiel engendré par un sous-ensemble d'arcs :**

Soient un graphe G=[X,U], X l'ensemble des sommets, U l'ensemble des arcs et V⊂U. Le graphe partiel engendré par V⊂U est le graphe ayant le même ensemble X de sommets que G, et dont les arcs sont les arcs de V (sont éliminés de G les arcs U-V). Si G est le graphe représentant les routes de France, celui qui représente les routes nationales de France est un graphe partiel.

➢ **Sous-graphe engendré par un sous-ensemble de sommets :**

Etant donné A⊂X, le sous-graphe engendré par A est le graphe $G_A$ dont les sommets sont les éléments de A et dont les arcs sont les arcs de G ayant leurs deux extrémités dans A. Si G est le graphe représentant les routes de France, celui qui représente les routes du Nord est un sous-graphe.

## 4.3.1. Les patterns de conception structurels

Pour chaque pattern, nous donnons tout d'abord **l'utilisation** qui peut en être fait, puis nous l'illustrons par un **exemple** qui aide à comprendre les applications qui en seront faites par la suite, ensuite nous définissons sa **structure** à l'aide d'un diagramme de classes UML, et enfin nous étudions les éventuelles **adaptations** et extensions possibles.

### 4.3.1.1. Le pattern Graphe Orienté

**Utilisation**

Ce pattern est destiné à la modélisation de tout modèle basé sur un graphe orienté, figure 4.3-1. Dans beaucoup d'applications, les relations entre éléments d'un ensemble sont orientés, c'est à dire qu'un élément i peut être en relation avec un autre j sans que j soit nécessairement en relation avec j.

Un graphe orienté G=[X,U] est déterminé par la donnée [GON95] :

➢ d'un ensemble X dont les éléments sont appelés sommets.

➢ d'un ensemble U dont les éléments u sont des couples ordonnés de sommets appelés arcs. Si u=(i,j) est un arc de G, i est l'extrémité initiale de u et j l'extrémité terminale.

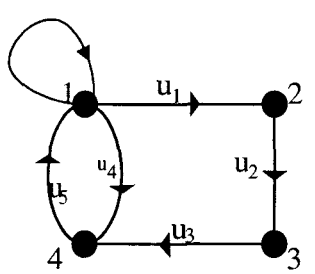



**Figure 4.3-1 : Exemple de graphe orienté**

**Exemple**

Les automates sont typiquement des modèles basés sur les concepts d'un graphe orienté. Les automates sont destinés à la description comportementale des systèmes complexes. Un automate est un graphe orienté dont les sommets représentent les états du système, et les arcs sont étiquetés par les événements qui déclenchent les transitions entre états, figure 4.3-2.

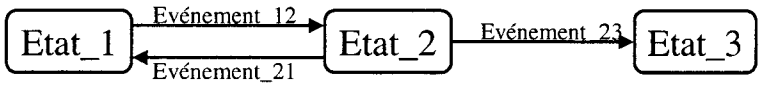



**Figure 4.3-2 : Exemple de statecharts**

**Structure, figure 4.3-3**

Un graphe orienté est un objet composite constitué de un à plusieurs (1..*) sommets qui sont reliés par des arcs. L'orientation d'un arc est définie par un sommet initial et un sommet terminal. Un arc possède une étiquette et peut être supprimé d'où l'utilisation de la classe association Arc.

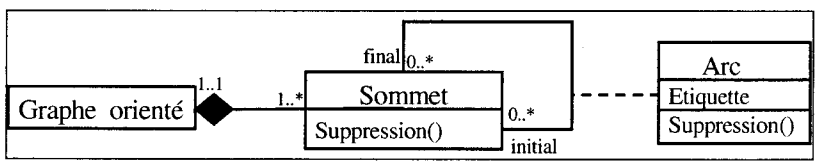





**Figure 4.3-3 : Structure du pattern graphe orienté**

Un même sommet peut être le sommet initial et/ou final de zéro à plusieurs (0..*) arcs. Il doit être relié à au moins un arc, ce qui n'est pas assuré par le diagramme de classes figure 4.3-3. Nous pouvons modifier les multiplicités de 0..* à 1..*, mais cela signifierait que chaque sommet est obligatoirement associé à un arc comme sommet initial et sommet final, ce qui bien entendu est faux. Pour assurer le fait qu'un sommet soit toujours lié à un arc au moins, en gardant les multiplicités 0..*, nous avons ajouté la contrainte OCL qui suit.

```
context Sommet inv :
(self.initial→union(self.final))→notEmpty
```

Cette contrainte signifie que l'ensemble des arcs reliés à un sommet ne peut être vide. Lors de la suppression d'un arc, cette contrainte garantit également l'absence de sommet sans relation avec un arc. Par contre après la suppression d'un sommet *s* tous les arcs liés à *s* doivent être détruits. Nous avons ainsi défini une contrainte OCL sur l'opération suppression(). Cette post-condition vérifie qu'après la suppression d'un sommet, l'intersection entre l'ensemble des arcs du graphe et l'ensemble des arcs liés au sommet avant la suppression du sommet est vide.

```
context Sommet :: suppression()
Post :(Arc.allInstances→intersection(self.arc@pre)→isEmpty
```

**Adaptations**

La structure du pattern graphe orienté sur le diagramme de classes figure 4.3-3, exprime qu'un sommet et un arc ne peuvent appartenir qu'à un seul et unique graphe. Cependant il est tout à fait envisageable qu'un sommet ou un arc puisse être partagé par plusieurs graphes, pour déduire par exemple un sous-graphe ou un graphe partiel. Dans ce cas il faut transformer la relation de composition en agrégation, de même que la multiplicité du coté de l'agrégat (1..*).

### 4.3.1.2. Le pattern graphe orienté biparti

**Utilisation**

Les graphes orientés bipartis sont des graphes colorés avec un nombre chromatique égal à deux. La coloration d'un graphe consiste en une affectation de couleurs à tous les sommets du graphe de telle sorte que deux sommets adjacents ne soient pas porteurs de la même couleur. Le nombre chromatique est défini comme le nombre minimum de couleurs distinctes nécessaires à la coloration des sommets de G.

Un graphe orienté G=[X,U] est dit biparti si l'ensemble des sommets X peut être partitionné en deux sous-ensembles X1 et X2 de telle sorte que, pour tout arc u(i, j) $\in$ U, figure 4.3-4 :

- $i \in X1 \Rightarrow j \in X2$
- $i \in X2 \Rightarrow j \in X1$

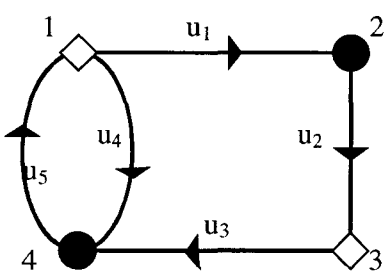



**Figure 4.3-4 : Graphe orienté biparti**

**Exemple**

Les réseaux de Petri (figure 4.3-5) constituent un outil suffisamment complet pour modéliser des phénomènes de natures très variées, et plus particulièrement les systèmes de contrôle/commande.

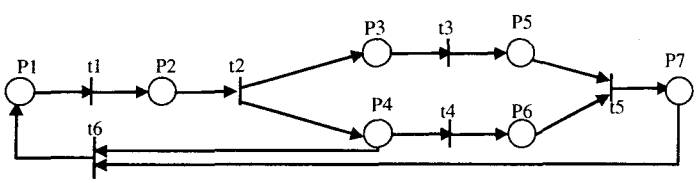



**Figure 4.3-5 : Réseau de Petri**

Ils permettent notamment de modéliser et de visualiser des comportements tels que le parallélisme, la synchronisation, et le partage de ressources. Les réseaux de Petri comportent deux ensembles de sommets distincts : les places (représentées par des cercles) et les transitions (représentées par des traits). Les arcs relient une transition à une place ou une place à une transition. Nous invitons le lecteur à se référer à [DAV92] pour plus de détails sur les réseaux de Petri et les grafcets qui sont aussi des graphes orienté bipartis avec deux types de sommets, les étapes et les transitions.

**Structure**

La structure du pattern « graphe orienté biparti » est décrite par le diagramme de classes figure 4.3-6. Ce diagramme comporte trois classes. La classe GOB (Graphe Orienté Biparti) qui est composée de un à plusieurs sommets appartenant à deux familles disjointes représentées par les deux classes distinctes Type_Sommet_1 et Type_Sommet_2. Ces sommets sont reliés par des arcs orientés, représentés par les relations précède et succède. L'absence d'une de ces associations signifierait que l'ensemble des instances d'une famille de sommets (Type_Sommet_1, Type_Sommet_2) seraient exclusivement extrémités initiales ou terminales.

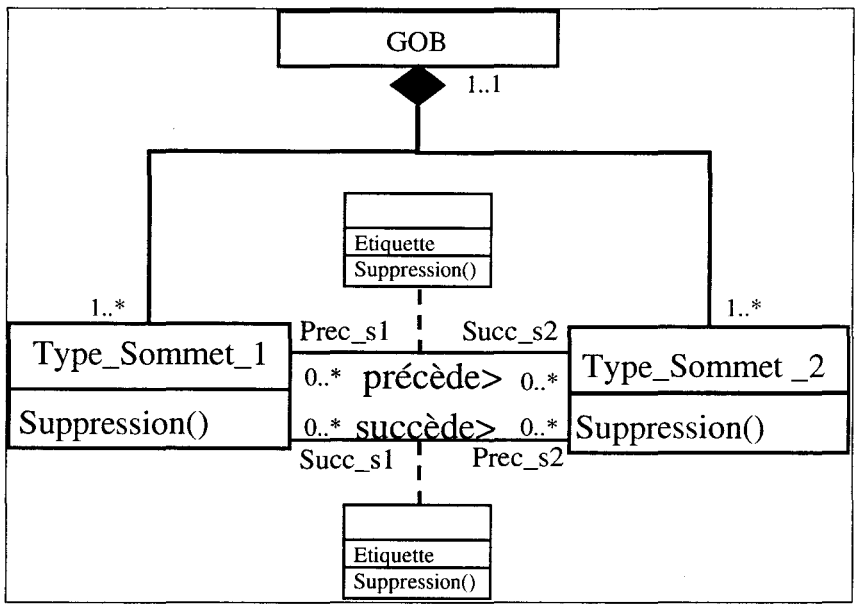



**Figure 4.3-6 : Structure du pattern graphe orienté biparti**

A cause des multiplicités minimales 0 sur les associations précède et succède, nous avons ajouté une contrainte OCL pour assurer que chaque instance d'un sommet (Type sommet_1 ou Type Sommet_2) est au moins l'extrémité d'un arc.

```
context Type_Sommet_1 inv :
      (self.Succ_s2→union(self.Prec_s2))→notEmpty
context Type Sommet_2 inv :
      (self.Succ_s1→union(self.Prec_s1))→notEmpty
```

Cette contrainte exprime qu'un sommet est au moins successeur ou prédécesseur d'un autre type de sommet. Lors de la suppression d'un arc (instance des associations précède ou succède), cette contrainte (un invariant) garantit aussi l'absence de sommet non relié à un arc.

Par contre, lors de la suppression d'un sommet s nous devons vérifier que tous les arcs liés à s sont détruits. Nous avons ainsi défini une contrainte OCL sur l'opération suppression().

```
context Type_Sommet_1 :: suppression()
      Post : (self.Succ_s2→union(self.Prec_s2)→isEmpty
context Type_Sommet_2 :: suppression()
      Post : (self.Succ_s1→union(self.Prec_s1)→isEmpty
```

**Adaptations**

L'augmentation du nombre chromatique peut nous amener à modifier la structure du pattern graphe orienté biparti pour le transformer facilement et l'adapter aux graphes tripartis etc. Pour cela nous utilisons une super-classe sommet que nous pouvons spécialiser pour ajouter le nombre de sommets nécessaires. La structure du pattern graphe orienté biparti est modifiée comme indiqué sur la figure 4.3-7.

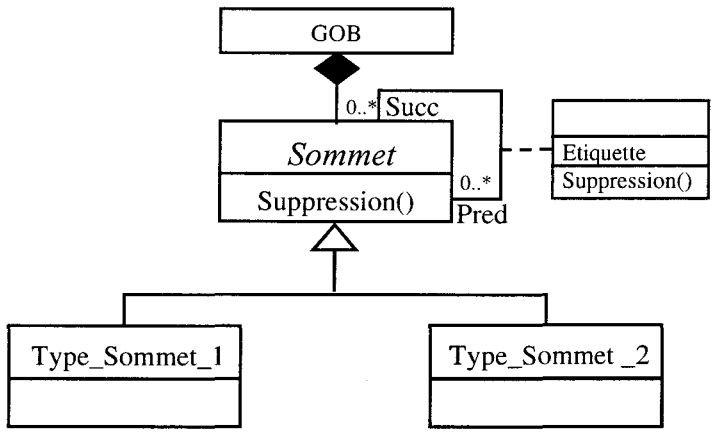



**Figure 4.3-7 : Pattern graphe orienté biparti avec héritage**

Les contraintes OCL vont alors porter sur la super-classe Sommet. Nous définissons une première contrainte OCL pour exprimer le fait qu'un sommet est extrémité d'un arc au moins. Etant donné que la classe Sommet est abstraite, elle n'est pas instanciable ; or le mot clé self fait référence à l'instance contextuelle, nous avons alors effectué une conversion de type grâce à la propriété prédéfinie oclAsType.

```
Context Sommet
inv: (self.oclAsType(Type_Sommet_1).succ→
      union(self.oclAsType(Type_Sommet_1).pred))→notEmpty
inv: (self.oclAsType(Type_Sommet_2).succ→
      union(self.oclAsType(Type_Sommet_2).pred))→notEmpty
```

Une deuxième contrainte garantit la cohérence du graphe en cas de suppression d'un sommet :

```
context Sommet :: suppression()
    Post : (self.oclAsType(Type_Sommet_1).succ→
    union(self.oclAsType(Type_Sommet_1).pred)→isEmpty and
    (self.oclAsType(Type_Sommet_2).succ→
    union(self.oclAsType(Type_Sommet_2).pred)→isEmpty
```

Il faut rajouter une troisième contrainte pour garantir que les deux sommets d'un arc ne peuvent appartenir à la même famille.

```
Context Sommet
inv :self.oclAsType(Type_Sommet_1).succ <> self.oclAsType(Type_Sommet_1).pred
inv :self.oclAsType(Type_Sommet_2).succ <> self.oclAsType(Type_Sommet_2).pred
```

La représentation du pattern graphe orienté biparti avec un héritage nous permet d'exprimer de manière dynamique les contraintes OCL, de plus il est facilement extensible aux graphes tripartis etc. Cependant un composant physique est souvent référencé par un lien de composition partagé. C'est à dire qu'il peut faire partie de plusieurs objets composites. Dans ce cas, l'agrégation est utilisée en UML pour exprimer cette possibilité de partage.

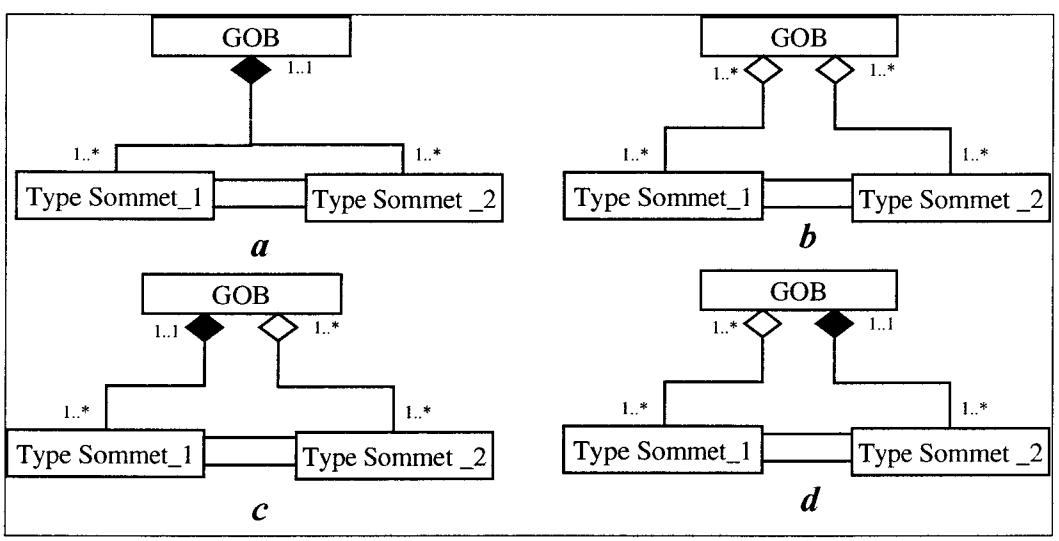



**Figure 4.3-8 : Différentes adaptions du pattern GOB**

Avec le diagramme de classes figure 4.3-6, on peut aisément adapter le pattern graphe orienté biparti aux configurations sur la figure 4.3-8. Par contre la solution avec héritage n'offre pas un bon degré de variabilité, et va difficilement s'adapter à un problème concret. Ainsi il n'est pas possible d'exprimer les situations *c* et *d* (un seul type de sommet partageable) avec le diagramme de classe sur la figure 4.3-7. Notre choix se porte donc le type de diagramme de classes sur la figure 4.3-6. Puis selon le type de partage, on choisira l'un des diagrammes (a, b, c, ou d) de la figure 4.3-8.

#### 4.3.1.3. Le Pattern arborescence

**Utilisation**

Ce pattern sert à la modélisation d'objets composites qui possèdent une forme arborescente, figure 4.3-9. Une arborescence est un graphe orienté où chaque sommet possède un seul précédent, sauf un qui n'en a pas : la racine. Quelque soit x ∈ X (ensemble des nœuds), il existe un chemin unique de la racine à x. Un nœud y quelconque sur le chemin unique de la racine à x est appelé ancêtre de x. Si le dernier arc sur le chemin de r vers x est (y, x), alors y est le père de x, y est alors un objet composite. Certains nœuds sont sans fils, ils sont appelés feuilles.

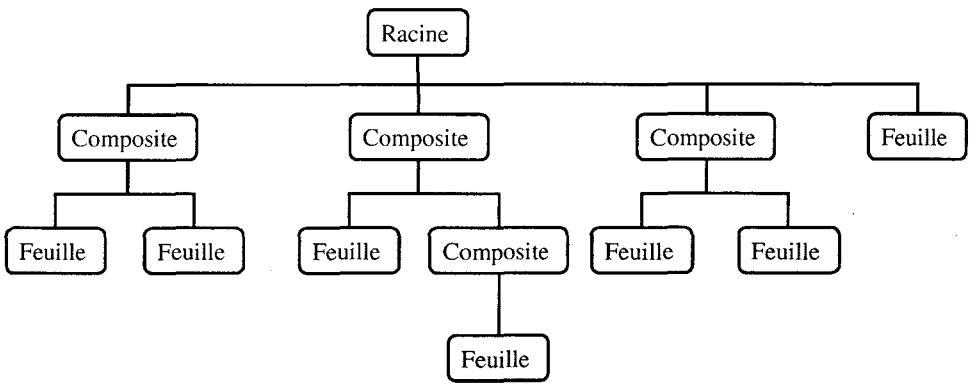



**Figure 4.3-9 : Objet composite avec une structure arborescente**

**Exemple, figure 4.3-10**

Une arborescence permet de traduire une hiérarchie de composition physique, structurelle. C'est une relation de type composant/ objet qui traduit le fait que les composants exhibent certaines formes de relations structurelles ou fonctionnelles entre eux et le composite auquel ils appartiennent [OUS97].

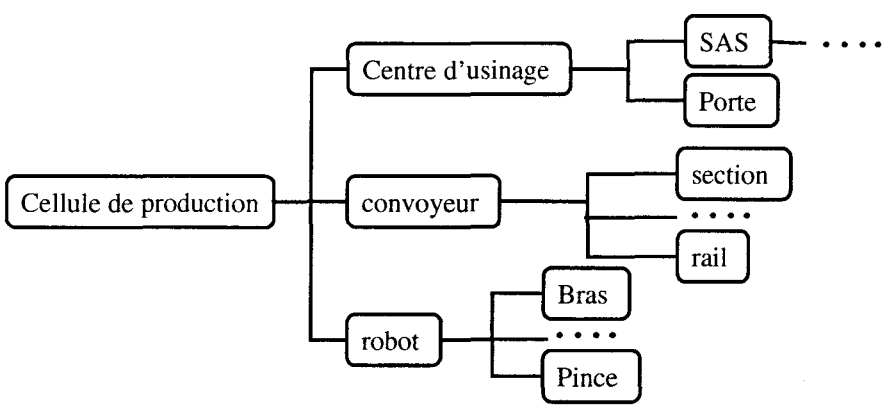



**Figure 4.3-10 : Description arborescente d'un système de production**

**Structure**

Le diagramme de classes sur la figure 4.3-11 représente le pattern arborescence. Il a été proposé par [GAM95]. La classe abstraite *composant* modélise l'ensemble des nœuds d'une arborescence. Un composant est soit une feuille (élément non décomposable), soit un composite dans ce cas, il regroupe un à plusieurs autres composants. Un composant ne peut appartenir qu'à un seul composite, sauf pour la racine qui n'appartient à aucun composant, d'où la multiplicité minimale 0 coté composite. A cause de cette multiplicité minimale, des feuilles peuvent être isolées. Nous définissons alors la contrainte OCL suivante , pour garantir l'existence d'un chemin unique qui mène à la racine à partir d'un composant quelconque.

```
context Composite inv :
    self.allInstances→forAll(c | c <> 'racine' implies c.père→size=1)
context Feuille inv :
    self.allInstances→forAll(f | f.père→size=1)
```

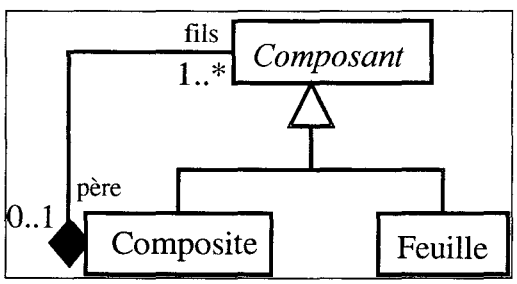



**Figure 4.3-11 : Structure du pattern arborescence**

**Adaptation**

Une telle classification en feuille et composite suppose que le type de chaque élément est connu a priori. Cependant il arrive que l'on ne puisse pas faire cette différentiation. Selon le niveau de granularité voulu, on peut poursuivre la décomposition. Ainsi un composant considéré comme feuille est susceptible de devenir composite. Pour cette raison, nous proposons une deuxième façon de représenter une structure arborescente sur la figure 4.3-12.

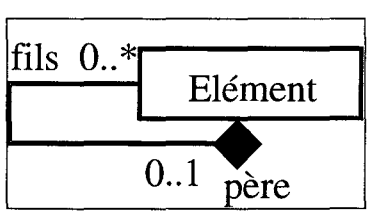



**Figure 4.3-12 : Adaptation du pattern arborescence**

Un élément peut se décomposer en zéro ou plusieurs autres éléments. Nous mettons une multiplicité minimale 0 du coté père car la racine n'en a pas. Les éléments de type feuille n'ont pas de fils, ce qui explique aussi la multiplicité minimale du coté fils. Un élément ne peut appartenir au plus qu'à un seul autre élément. On doit ajouter une contrainte OCL, pour assurer qu'il n'y a pas d'élément isolé : tous les éléments ont un seul et unique père, excepté la racine.

```
context Elément inv :
    self.allInstances→forAll(e | e<> 'racine' implies e.père→size=1)
```

### 4.3.2. Les patterns métiers

L'étude de systèmes dans un domaine particulier fait appel à un ensemble de modèles type. Ces modèles définissent un domaine partageable et utilisable par les différents métiers. L'analyse des systèmes à événements discrets, se fait fréquemment à partir de trois aspects fondamentaux : leur structure physique, les fonctions qu'ils assurent et leurs comportements.
Nous avons alors créé trois patterns métiers pour décrire chacun de ces aspects.

#### 4.3.2.1. Modèle structurel

**Utilisation**

Le premier aspect récurent à étudier dans un système complexe, est naturellement son aspect structurel à travers sa décomposition matérielle. Le système est ainsi appréhendé comme un objet composite, formé par l'agrégation d'un ensemble d'objets, appelés ses composants. Nous créons le pattern « modèle structurel » pour représenter la structure physique de tout système.

**Exemple**

Tout élément d'un système de production est constitué de composants, chaque composant est susceptible d'être décomposé à son tour. Cette structure se présente donc sous forme arborescente. La figure 4.3-13 montre un extrait de la décomposition d'une machine outil à commande numérique, le CU60 de Graffenstaden [TOG92].

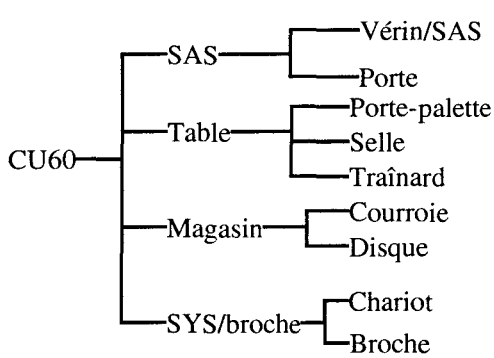



**Figure 4.3-13 : Modèle structurel d'un centre d'usinage**

**Structure**

Nous avons vu (§ 4.3.1.3) que tout objet se présentant sous la forme arborescente peut être représenté par le pattern arborescence, d'où la représentation du modèle structurel sur le diagramme de classes figure 4.3-14. Un système physique est constitué de plusieurs éléments qui à leur tour peuvent être décomposés. Nous avons réutilisé le pattern arborescence sur la figure 4.3-12, étant donné que nous ne connaissons pas a priori la finesse de la décomposition du système. En effet le concepteur peut se limiter à une décomposition au niveau des machines, des constituants de ces derniers ou aller jusqu'au niveau de composants tels que les capteurs etc.

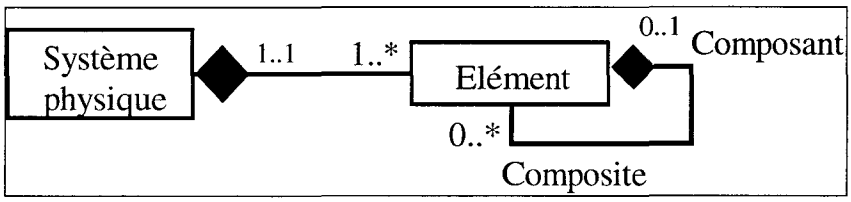



**Figure 4.3-14 : Modèle structurel**

### 4.3.2.2. Modèle fonctionnel

**Utilisation**

L'analyse d'un système physique permet de mettre en évidence non seulement sa structure matérielle, mais également les différentes fonctions du système. Une fonction est définie par [TOG92] comme la réponse d'un système ou d'un composant à un stimulus donné, lorsqu'il fonctionne normalement, indépendamment de l'environnement dans lequel il est sollicité. Le modèle fonctionnel spécifie alors les différentes fonctions d'un système à partir de la composition de fonctions élémentaires, nous proposons alors un pattern pour décrire tout modèle fonctionnel.

**Exemple**

La figure 4.3-15 montre un extrait du modèle fonctionnel du CU60, qui a une fonction principale d'usinage [TOG92]. La construction se fait de proche en proche. Le résultat final est un graphe orienté dont les nœuds modélisent les fonctions, et les arcs les liens entre ces fonctions.

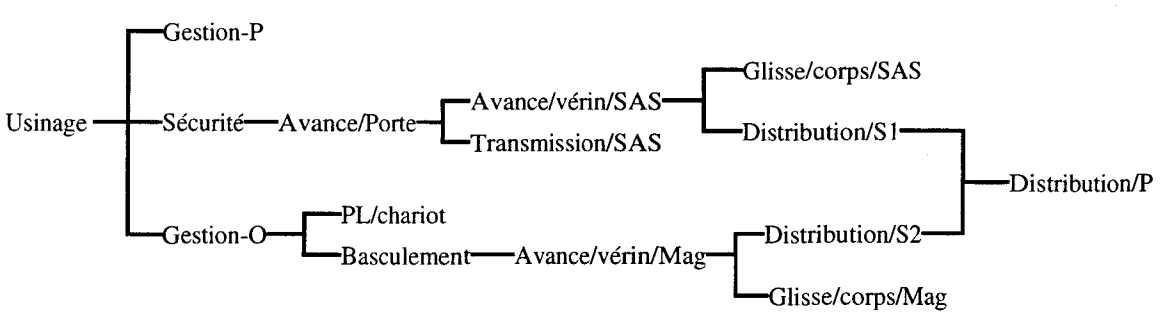



**Figure 4.3-15 : Exemple de graphe fonctionnel**

**Structure**

Le pattern « modèle fonctionnel » est ainsi modélisé en utilisant le pattern graphe orienté, comme indiqué sur la figure 4.3-16. L'élément modèle fonctionnel est composé de fonctions qui sont reliées par des liens fonctionnels. Pour chaque fonction, il faut déterminer les fonctions mères (les fonctions qui l'utilisent) et les fonctions filles (celles qui la produisent). Nous laissons les multiplicités minimales à 0, pour la fonction principale (pas de mère) et les fonctions terminales (pas de fille).

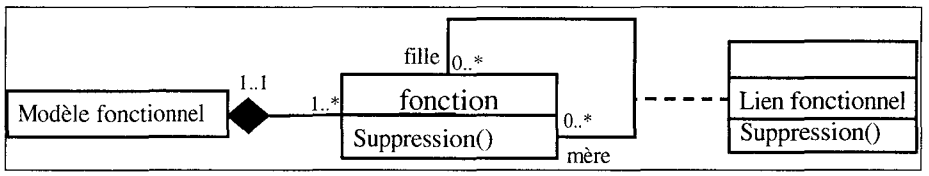



**Figure 4.3-16 : Modèle fonctionnel**

#### 4.3.2.3. Modèle de comportement

**Utilisation**

Un troisième axe important est souvent abordé concernant la modélisation d'un système : il s'agit de l'étude de son comportement. Il existe plusieurs modèles pour l'étude des comportements d'un système. Cependant qu'il s'agisse des machines à état, des statecharts, des diagrammes d'états-transitions UML, ou d'un quelconque outil permettant la représentation d'un comportement d'un système, ils sont très souvent basés sur des modèles graphiques dont les nœuds représentent l'état du système et les arcs les transitions entre les états. Nous avons créé le pattern « modèle de comportement » pour permettre d'une manière générale de représenter le comportement du système.

**Exemple**

Prenons l'exemple d'un téléviseur en considérant trois états : marche, arrêt et veille. Le modèle comportemental du téléviseur peut alors être représenté comme indiqué sur la figure 4.3-17. Les différents états (sommets du graphe) possibles atteints sont mis en valeur et les évolutions possibles entre états sont représentés par des arcs (actions).

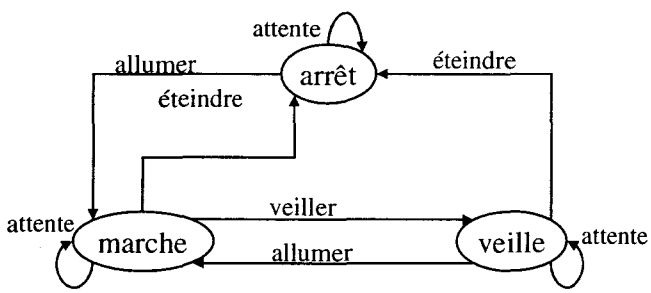



**Figure 4.3-17 : Exemple de graphe d'état**

**Structure**

Le diagramme de classes du modèle de comportement est réalisé grâce au pattern graphe orienté, figure 4.3-18. Un modèle de comportement est composé d'états reliés par des actions. Chaque action fait passer d'un état à un autre (qui n'est pas forcément différent). Nous laissons les multiplicités minimales à 0, pour l'état initial (n'a pas de précédent) et l'état final (n'a pas de successeur).

Du fait de l'explosion combinatoire, les graphes d'états peuvent devenir très rapidement difficiles à lire. Les concepteurs effectuent alors des regroupement d'états. Pour prendre en compte cet aspect, il suffit de rajouter une composition réflexive sur la classe Etat.

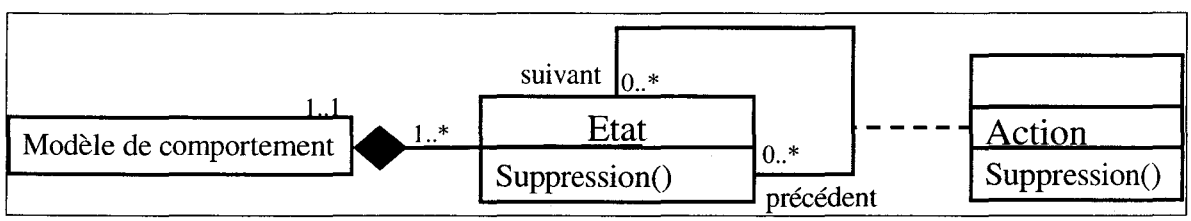



**Figure 4.3-18 : Modèle de comportement**

L'utilité de développer des composants réutilisables a été très largement démontrée, surtout en génie logiciel. Cependant, il ne suffit pas de disposer de composants réutilisables pour rendre la réutilisation spontanée. Nous avons ainsi proposé un ensemble de composants réutilisables et surtout nous avons formalisé une démarche pragmatique pour la capitalisation et la réutilisation de ces derniers.

Le référentiel est évolutif, il est enrichi par les retours d'expériences, par conséquent à long terme le taux de réutilisation sera alors très « satisfaisant ». Nous ne disposons pas de métrique permettant de mesurer un taux de satisfaction, mais nous pensons que n'importe quel niveau de réutilisation est préférable à aucune réutilisation, car cela représente une économie de ressources qui, sinon, seraient nécessaires pour traiter un problème déjà résolu.

Notre démarche s'appuie sur une approche orientée objet et offre une bonne adaptabilité et une bonne maintenabilité du référentiel qui devient ainsi très réactif. Ces caractéristiques sont essentiels pour tous les moyens intervenant dans la conception des systèmes automatisés de production, car ils permettent d'améliorer leur qualité et leur réactivité.

Nous allons présent appliquer la totalité de notre démarche dans le chapitre 5, en concevant un référentiel pour le projet CASPAIM.

# CHAPITRE 5.
# Etude du référentiel CASPAIM

## Résumé

**A travers le dernier chapitre de ce mémoire, nous appliquons la totalité de notre démarche proposée dans les chapitres 3 et 4 au projet CASPAIM (cf. § 1.3). Nous allons créer des métamodèles pour les points de vue (Commande, Planification/Ordonnancement, Supervision, Partie Procédé) abordés dans CASPAIM, puis nous procédons à leur intégration. Nous mettrons alors en œuvre la réutilisation des patterns que nous avons créés au chapitre 4. Enfin nous présentons le logiciel CASPAIM SOFT que nous avons développé pour valider concrètement les métamodèles obtenus.**

**Introduction**

Nous avons présenté au § 1.3 le projet CASPAIM, ainsi que les différents points de vue traités dans le cadre de ce projet. En suivant la démarche que nous avons proposée dans les chapitres 3 et 4, nous allons voir tout d'abord, comment passer des modèles métier développés par les concepteurs de CASPAIM aux métamodèles UML.

Puis nous procédons à l'intégration de ces métamodèles.

Le but n'est donc pas de décrire exhaustivement l'ensemble des points de vue, mais de montrer l'intérêt de notre approche. Pour plus de détails sur les modèles métiers, nous invitons le lecteur à consulter les références correspondantes.

## 5.1. Le point de vue partie procédé [NDI00c]

### 5.1.1. Architecture d'un système flexible de production

La partie procédé d'un système de production gère le flux de matières (produits) ; elle comprend l'ensemble des dispositifs matériels intervenant dans le processus de production ainsi que leur organisation. Les produits circulant dans le système de production subissent des transformations par le biais de la partie opérative (opérateurs, ressources etc.). Pour les besoins du projet CASPAIM, [AMA94] effectue une description de l'architecture physique basée sur trois concepts essentiels : les ressources, les lieux physiques et les relations d'accessibilité, figure 5.1-1.

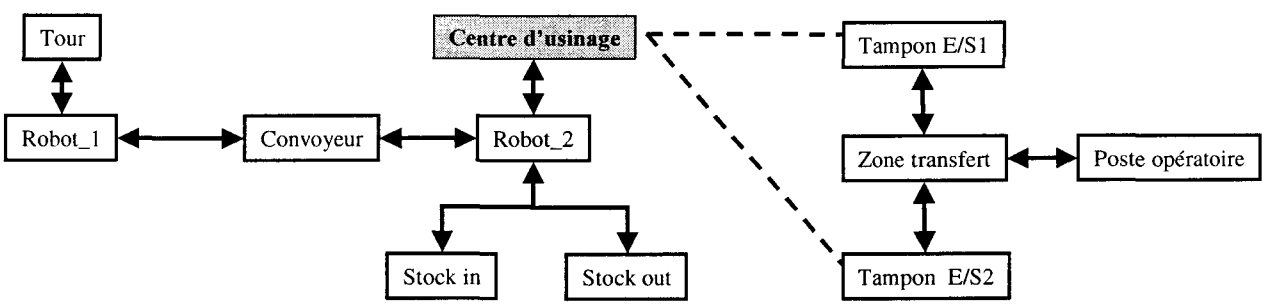



**Figure 5.1-1 : Relations d'accessibilité entre ressources du système**

Un système de production est composé de plusieurs ressources (machines, robots, convoyeurs etc.) qui peuvent comporter plusieurs lieux physiques. Sur la figure 5.1-1, le système est composé d'un tour, de deux robots etc. Le centre d'usinage contient plusieurs lieux physiques, à savoir deux zones tampon, une zone de transfert et un poste opératoire. Un lieu physique appartient à une ressource de production et représente soit une zone opératoire soit une zone d'accès à cette ressource.

La relation d'accessibilité définit les liens existant entre les différentes ressources (machines, robots etc.) de production concernant les échanges de pièces. Si un produit peut transiter d'une ressource à une autre sans intermédiaire, on parle d'accessibilité directe. Les accessibilités indirectes sont déduites par transitivité. Deux types d'accessibilité sont alors envisageables : les accessibilités externes entre des zones physiques appartenant à des ressources distinctes, et les accessibilités internes entre zones physiques d'une même ressource.

Par exemple sur la figure 5.1-1, le convoyeur est accessible directement à partir du robot_1 (degré d'accessibilité 1), tandis que pour amener un produit du robot_2 au tour il faut effectuer trois transitions en passant par le convoyeur puis par le robot_1 (degré d'accessibilité 3).

### 5.1.2. Métamodèle de l'architecture physique et du produit

**Métamodèle de l'architecture physique du système**

La figure 5.1-2, représente la cellule flexible de l'Ecole Centrale de Lille. Sa description structurelle peut varier d'un point de vue à un autre. Elle peut être perçue comme un seul système composé de machines, automates, robots, convoyeur etc. Ces éléments peuvent à leur tour être décomposés jusqu'à un niveau de détail voulu par le concepteur. Cette même cellule peut aussi être considérée comme un ensemble de trois systèmes : un système de transport, un système d'assemblage et un système d'usinage.

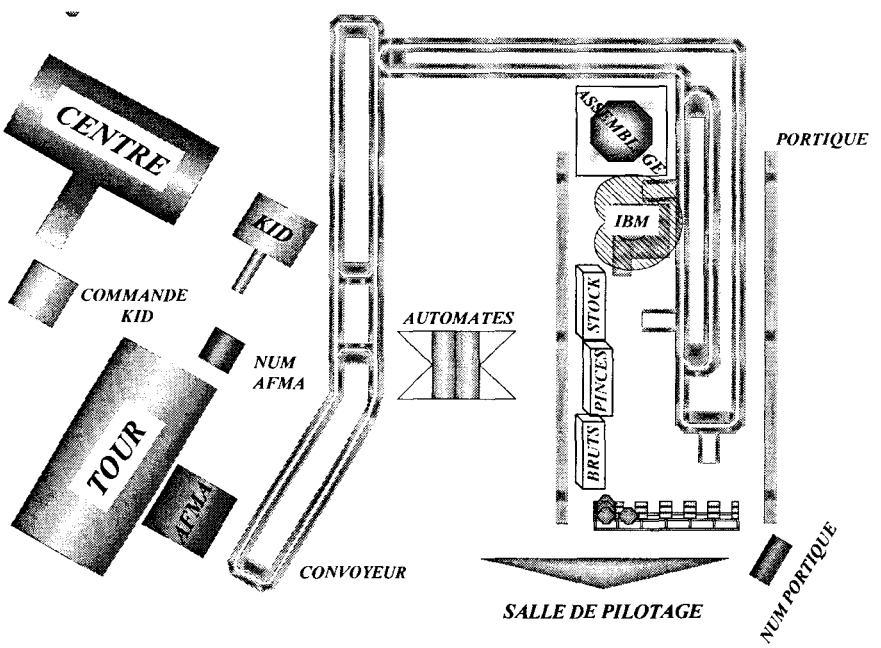



**Figure 5.1-2 : La cellule flexible de l'Ecole Centrale de Lille**

La description de la partie physique repose sur une analyse structurelle qui montre un ensemble d'éléments reliés. La métamodélisation de l'architecture physique des systèmes de production peut alors être réalisée en utilisant le pattern métier « modèle structurel » que nous avons adapté sur le diagramme de classes figure 5.1-3.

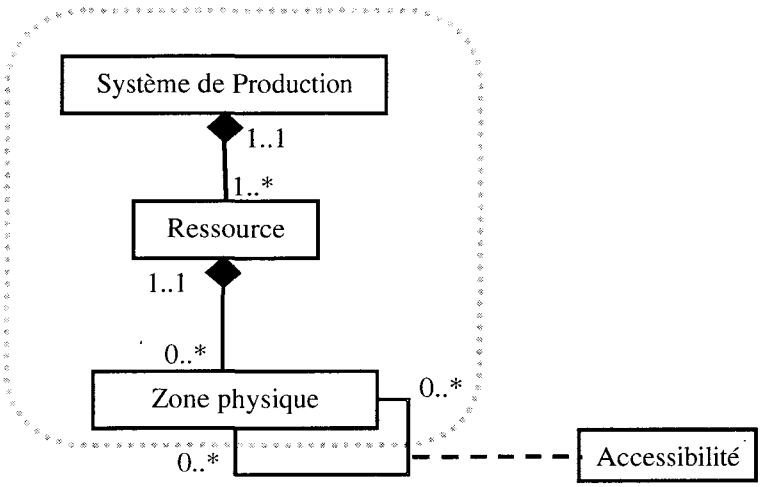



**Figure 5.1-3 : Métamodèle de l'architecture physique**

Un système de production est composé de plusieurs ressources, une ressource appartient à un seul système de production. Les éléments qui intéressent les intervenants sont de type zones physiques, c'est à dire des lieux pouvant intervenir dans le transfert d'un produit, ou servant de zone d'opération avec changement de l'état du produit. Nous décrivons les deux niveaux de décomposition : les ressources et les zones physiques qui ne sont pas décomposables. Nous créons alors deux classes distinctes : la classe composite ressource, et la classe zone physique.

Pour traduire les accessibilités directes (internes et externes), nous rajoutons une classe accessibilité. La relation d'accessibilité est mise directement sur les zones physiques, car tous les échanges passent par ces dernières. Etant donné qu'une zone physique appartient à une seule ressource, les accessibilités entre ressources s'en déduisent automatiquement. Une zone physique peut être accessible ou accéder à zéro ou plusieurs autres zones physiques.

Une zone physique devient lieu caractéristique dès lors qu'elle est accessible à d'autres zones externes. Grâce à ce métamodèle, nous pouvons traduire les trois concepts de la partie physique que sont les ressources et leur décomposition en lieux physiques ainsi que leurs relations d'accessibilité. Ce métamodèle permet toutes les flexibilités de décomposition. Pour poursuivre une analyse structurelle plus fine du système, il nous suffit de rajouter une agrégation réflexive sur la classe zone physique. Il est également possible que deux zones physiques définissent les liens d'accessibilité entre deux systèmes différents ; par exemple un aiguillage faisant le lien entre les différents systèmes de la cellule représentée sur la figure 5.1-2.

**Métamodèle de la description des produits**

Ainsi, la fabrication d'un produit passe souvent l'usinage et l'assemblage de plusieurs autres produits. Nous pouvons décrire un produit ainsi que ses différents constituants (d'une manière générale sa nomenclature) par le biais du pattern « composite » sans distinction entre feuilles et composants, figure 5.1-4.

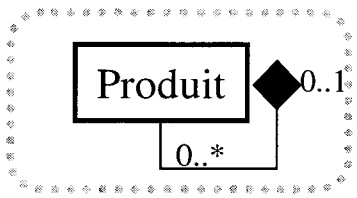



**Figure 5.1-4 : Métamodèle de la description du produit**

Le boulon est un exemple typique d'assemblage, il est constitué d'un vis et d'un écrou, figure 5.1-5.

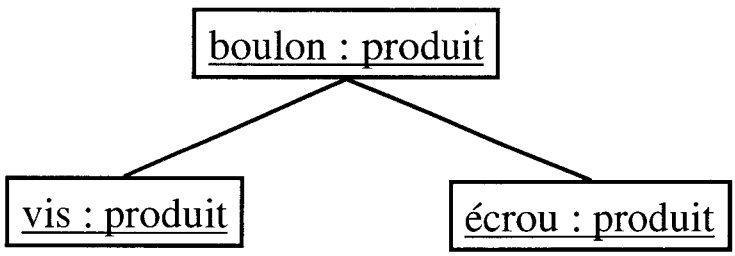



**Figure 5.1-5 : Diagramme d'objets, exemple du boulon**

**Remarque** : ce modèle de produit ne prend en compte que la partie nomenclature, il est utilisé pour définir les gammes logiques comme on le verra au § 5.2.1. Il pourra être complété par la suite selon les besoins des concepteurs, pour par exemple être mis en relation avec un système de gestion de données techniques.

## 5.2. Le point de vue partie commande [NDI00c]

La partie commande d'un système de production assure la coordination et le séquencement des commandes (flux informationnels) applicables à la partie procédé. Nous étudions les modèles développés par [AMA94] et [CRU91] pour la partie commande, ensuite nous proposons les métamodèles UML correspondants. Les modèles de la commande abordent deux aspects indépendamment l'un de l'autre :

- la partie logique concerne les objectifs de production ; elle utilise deux outils de modélisation graphique : les gammes logiques et les gammes opératoires ;
- et la partie physique caractérisant les moyens de production ainsi que leur organisation.

## 5.2.1. Gamme logique

### 5.2.1.1. Modèle RdP

La gamme logique d'un produit décrit le séquencement (contraintes d'ordre) des opérations élémentaires et caractéristiques définissant ainsi le processus de fabrication qui permet d'obtenir le produit fini à partir de son état brut.

Le produit à fabriquer est décrit à l'aide d'une gamme logique, sans tenir compte de la nature des équipements à utiliser. Une gamme logique correspond ainsi à une description purement fonctionnelle du processus de fabrication, sans aucune référence, d'une part au procédé technique utilisé pour réaliser une opération caractéristique, et d'une autre part, à l'aspect physique ou matériel du système de fabrication.

Le modèle utilisé est le RdP où une place modélise l'état d'un produit, et une transition modélise une fonction qui modifie l'état du produit ou crée de nouveaux produits à partir de produits existants, figure 5.2-1.

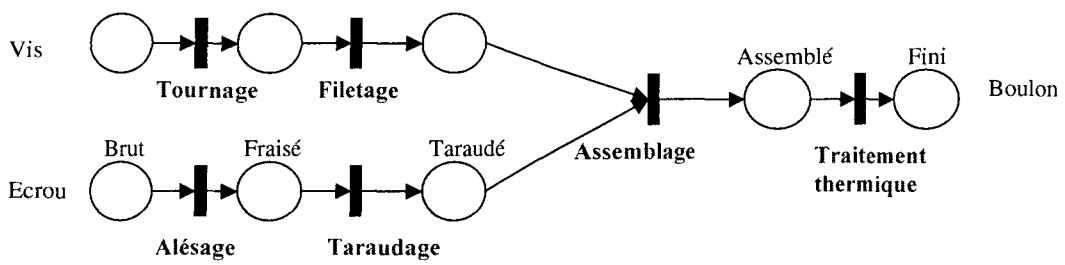



**Figure 5.2-1 : Modélisation de la gamme logique par réseau de Petri**

### 5.2.1.2. Métamodèle de la gamme logique

Les réseaux de Petri sont des graphes orientés bipartis ; nous pouvons alors construire le métamodèle UML des gammes logiques en réutilisant le pattern « graphe orienté biparti ». Une place peut être partageable, dans le cas par exemple d'un assemblage. Pour fabriquer un boulon il faut trois gammes logiques : une gamme pour la vis, l'écrou et le boulon, figure 5.2-2.

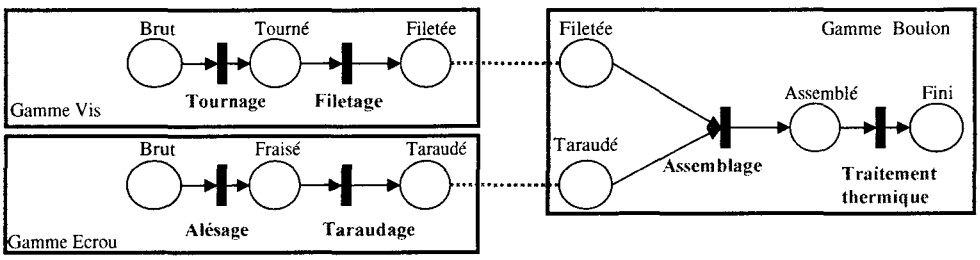



**Figure 5.2-2 : Places partagées entre gammes logiques**

Les places finales des gammes vis et écrou sont les places initiales de la gamme boulon. Pour cela nous utilisons le pattern « graphe orienté biparti » avec un type sommet partageable, figure 5.2-3.

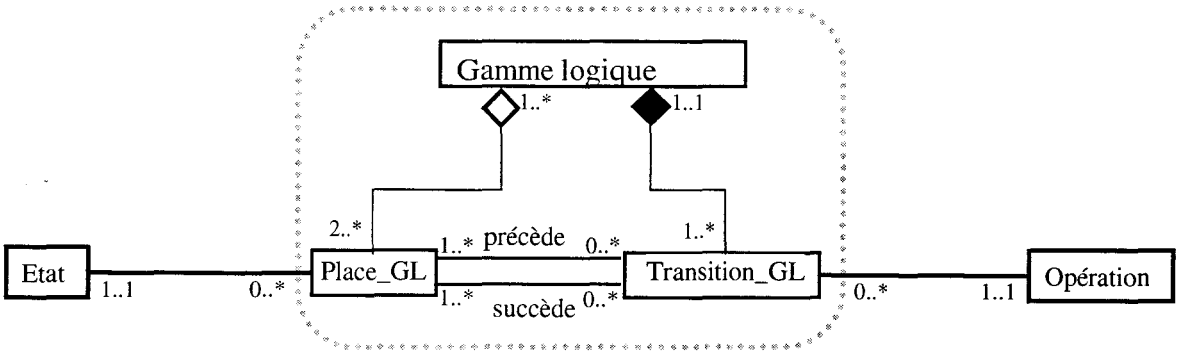



**Figure 5.2-3 : Métamodèle de la gamme logique**

Chaque place de la gamme correspond à un état d'avancement du produit, on peut retrouver cet état dans plusieurs places. Une transition correspond à une opération qui change l'état du produit, cette opération peut être répétée plusieurs fois dans la gamme.

Une place de la gamme logique suit ou précède zéro à plusieurs transitions ; les multiplicités minimales sont égales à zéro pour prendre en compte les places initiales et terminales. Par contre une transition est forcément suivie et précédée d'une place au moins, car une gamme logique débute et se termine toujours par une place. Une gamme logique est alors constituée d'au moins deux places et une transition.

## 5.2.2. Gamme opératoire

### 5.2.2.1. Modèle RdP

La gamme opératoire d'un produit décrit le séquencement des différentes transformations, fonctionnelles et positionnelles, faisant apparaître la succession des lieux physiques sur lesquels il transite. Une transformation fonctionnelle correspond à un traitement qui modifie l'état du produit, tandis qu'une opération positionnelle effectue le transfert du produit d'une zone opératoire à une autre.

Les gammes opératoires sont obtenues à partir des gammes logiques, en prenant en compte les informations relatives à l'architecture physique. Le système sur la figure 5.2-4 est composé de deux tours, d'une fraiseuse et d'un convoyeur ; les relations d'accessibilité sont schématisées par les flèches bidirectionnelles.

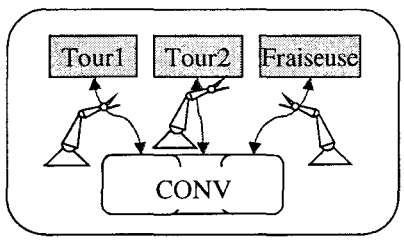



**Figure 5.2-4 : Exemple d'atelier de production**

Pour chacun des traitements figurant sur les gammes logiques, une liste des machines candidates, capables de réaliser le traitement considéré, est effectuée. Chaque traitement, ayant plusieurs machines possibles, générera une structure alternative traduite par une flexibilité de choix des machines et une flexibilité d'ordre des opérations. Une distinction entre les transferts et traitements est effectuée. Une gamme opératoire est ainsi représentée par un RdP dont les places représentent l'état d'avancement dans le processus de fabrication, et les transitions correspondent à des opérations de traitement ou de transfert du produit, figure 5.2-5.

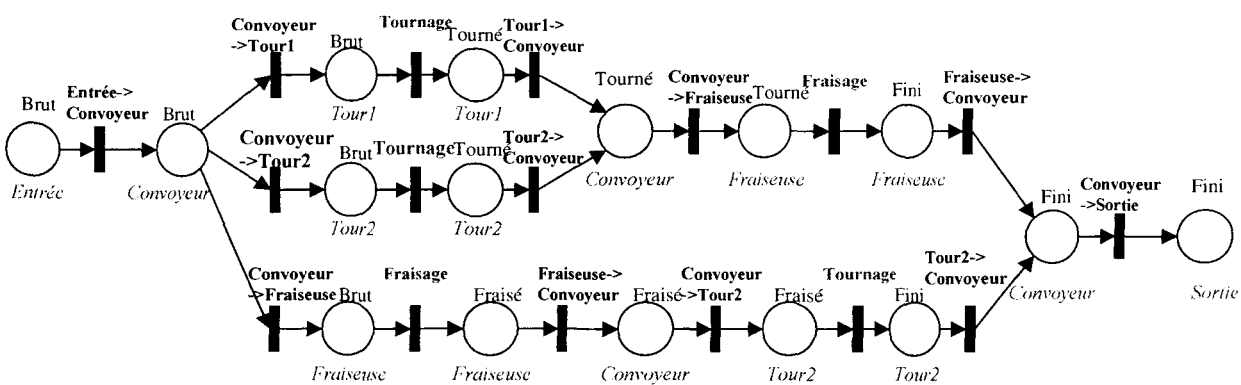



**Figure 5.2-5 : Génération de gamme opératoire**

### 5.2.2.2. Métamodèle de la gamme opératoire

Pour construire le métamodèle UML des gammes opératoires, nous utilisons le pattern « graphe orienté biparti » de la même façon que nous l'avons utilisé pour la gamme logique, figure 5.2-6. Une place de la gamme opératoire correspond toujours à un état du produit, tandis qu'une transition désigne soit une opération positionnelle (transfert uniquement), soit une opération fonctionnelle (usinage par exemple).

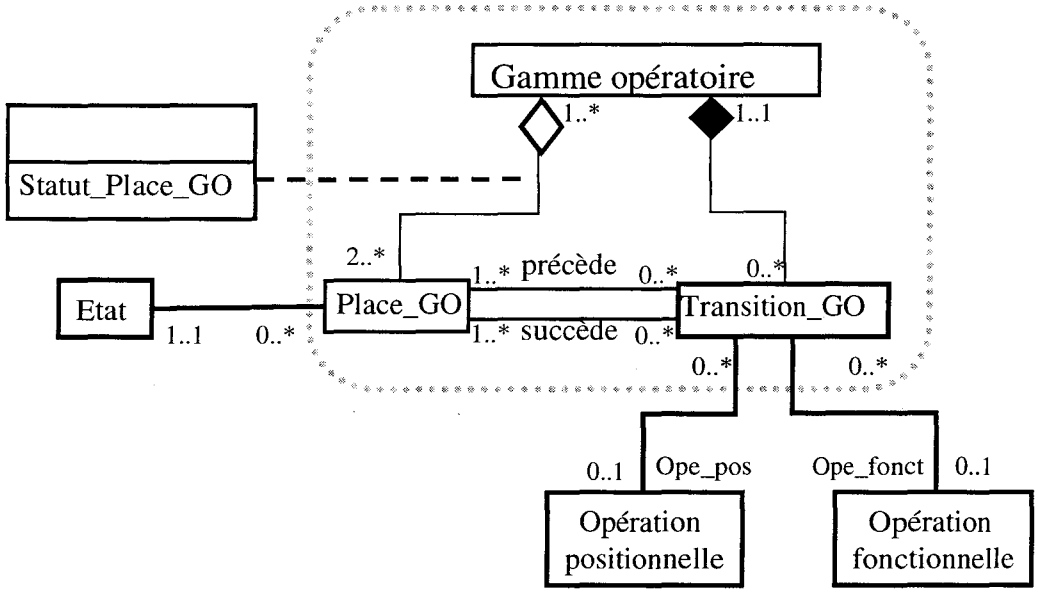



**Figure 5.2-6 : Métamodèle de la gamme opératoire**

Une transition ne peut représenter simultanément une opération fonctionnelle et positionnelle, et les multiplicités minimales (0) du côté des opérations peuvent faire qu'une transition soit sans opération affectée. Nous définissons alors la contrainte OCL suivante :

```
context Transition_GO inv :
If self.ope_pos → size=1 then self.ope_fonct →size=0
else self.ope_fonct → size=1
endif
```

L'attribut statut_place_GO permet de connaître la nature (initiale, finale ou normale) de la place dans une gamme opératoire donnée. Cette distinction est nécessaire car les places étant partageables, la position de chaque place est susceptible de varier d'une gamme à une autre. Par exemple, une place finale dans une gamme donnée peut être la place initiale d'une autre gamme et vice-versa. Cette situation est fréquente notamment pour les produits obtenus par assemblage.

Nous pouvons également représenter le métamodèle de la gamme opératoire en utilisant une classe abstraite *Opération*, dans ce cas la contrainte OCL définie ci-dessus n'est plus nécessaire. Mais étant donné que nous effectuons par la suite une implantation dans une base de données relationnelle, nous préférons une solution sans héritage.

Etant données les relations entre les points de vue Commande et Partie Procédé, il nous semble judicieux de commencer par une première intégration de ces deux points de vue avant de poursuivre l'étude des autres points de vue [NDI00b], [NDI00c].

## 5.3. Intégration Partie Commande/Partie Procédé

Deux acteurs [AMA94] et [AMA94]&[CRU91][6] interviennent dans la construction des points de vue partie commande et partie procédé. Les modèles à concevoir sont au nombre de quatre : le modèle de la description du produit, le modèle de l'architecture physique, la gamme logique et la gamme opératoire.

Nous définissons le diagramme des cas d'utilisation sur la figure 5.3-1 :

- Les gammes opératoires sont obtenues à partir des spécifications des besoins fonctionnels (Gamme logique) et de l'architecture physique du système étudié.
- La gamme logique est construite d'après les spécifications du produit à fabriquer.
- Le modèle du produit et la gamme logique sont spécifiés et conçus indépendamment de l'architecture physique du système de production.

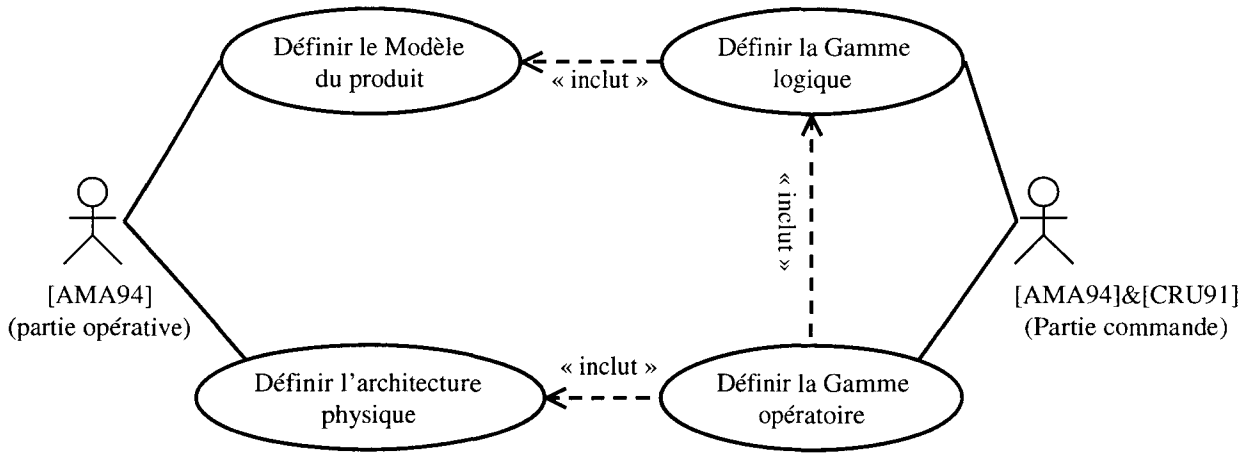



**Figure 5.3-1 : Cas d'utilisation des modèles de la partie commande et procédé**

Le diagramme de classes sur la figure 5.3-2 représente le métamodèle intégrant les modèles de la partie commande (les gammes logiques et opératoires), et de la partie procédé (architecture physique et modèle du produit).

---

[6] [AMA94] a poursuivi les travaux de [CRU91] sur les gammes logiques et les gammes opératoires ; de ce fait nous avons créé l'acteur [AMA94]&[CRU91]

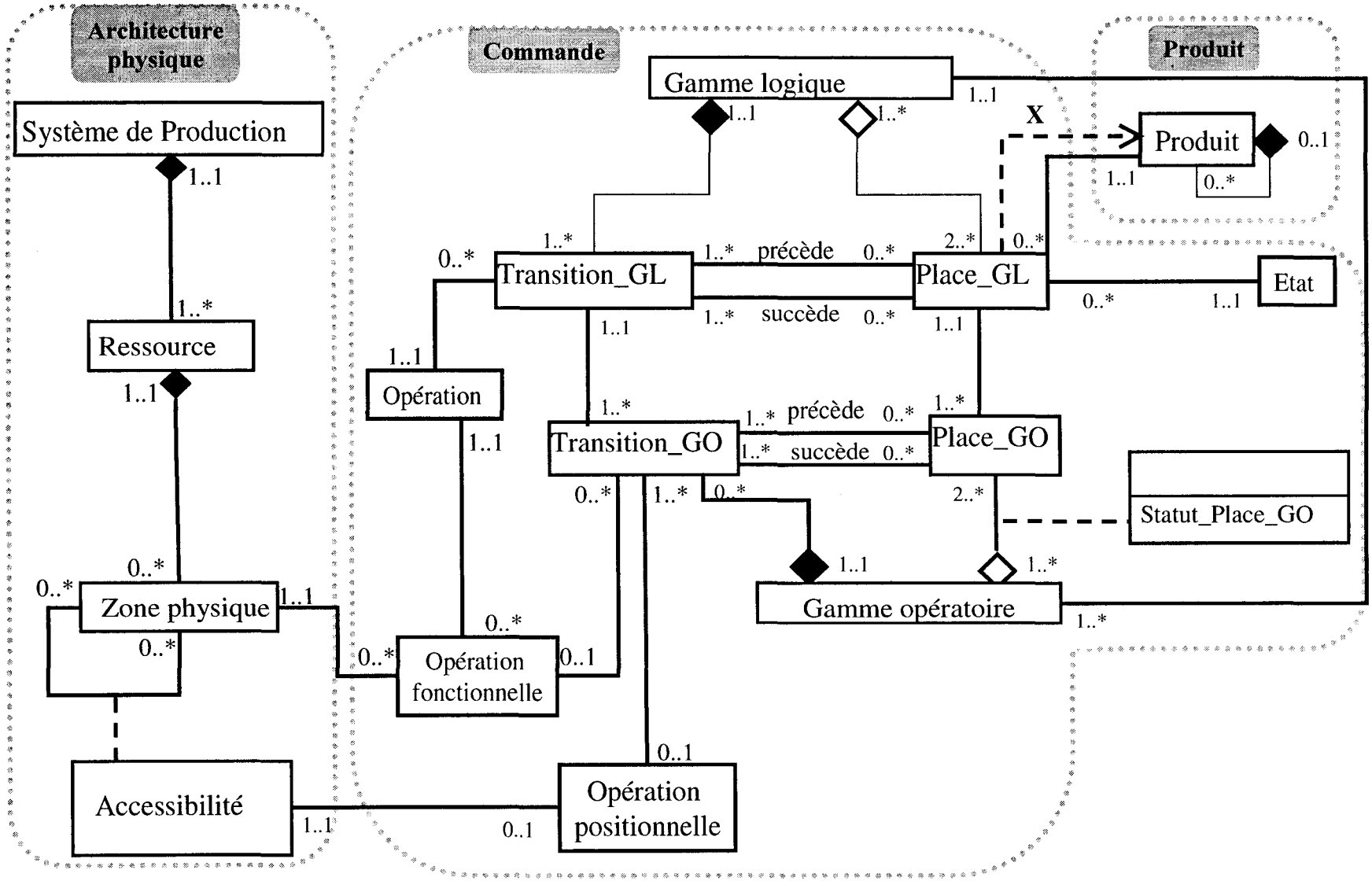



**Figure 5.3-2 : Métamodèle intégrant les point de vue Partie Commande/ Partie Procédé**

**Liens entre la gamme opératoire et la gamme logique**

Chaque gamme opératoire est générée à partir d'une seule gamme logique qui elle, peut générer plusieurs gammes opératoires. Une transition de la gamme logique peut générer une à plusieurs transitions de la gamme opératoire par dissociation des opérations fonctionnelles des transferts. Cette duplication des transitions entraîne forcément celle des places, ainsi une place de la gamme logique génère plusieurs places de la gamme opératoire.

La classe Etat existe dans les diagrammes de classes sur les figures 5.2-2 (gamme logique) et 5.2-4 (gamme opératoire), avec la même signification sémantique dans les deux cas. Nous gardons la classe Etat de la gamme logique, car une place de la gamme opératoire est générée à partir d'une seule place de la gamme logique. Par navigation, on peut alors retrouver l'état associé à chaque place de la gamme opératoire.

Une Opération de la gamme logique correspond à zéro ou plusieurs opérations fonctionnelles de la gamme opératoire. Une opération fonctionnelle est générée à partir d'une seule opération de la gamme logique.

**Liens entre la gamme opératoire et l'architecture physique**

La gamme opératoire s'appuie sur l'architecture physique du système de production, contrairement à la gamme logique. Une opération fonctionnelle est réalisée sur une et une seule zone physique, chaque zone physique pouvant servir à la réalisation de zéro à plusieurs opérations fonctionnelles.

Une opération positionnelle entre deux ressources (ou au sein d'une même ressource) n'est possible que s'il existe une relation d'accessibilité entre deux zones physiques de ces ressources. Elle est définie par une et une seule relation d'accessibilité (une zone de départ et une d'arrivée, sans changement d'état du produit). Une accessibilité entre deux zones physiques définit zéro ou une opération positionnelle.

**Liens entre le modèle du produit et la gamme logique**

Dans une place de la gamme logique, nous repérons l'état d'un produit. Un produit va se trouver dans plusieurs états et places dans la gamme logique, mais à un instant donné, il se trouve dans une seule place et un seul état. Pour respecter cette contrainte, nous rajoutons la contrainte d'exclusion temporelle (voir § 3.2.4).

## 5.4. Le point de vue supervision [NDI00a]

Dans la supervision, cinq fonctions sont principalement abordées dans CASPAIM :

- La gestion des modes de marche [BOI91],
- Le diagnostic [TOG92],
- La surveillance de la commande [ELK93],
- Le recouvrement [BER98],
- La surveillance prédictive indirecte [LY99].

Pour établir le métamodèle UML de la supervision, nous créons d'abord le métamodèle pour chacune de ces fonctions, ensuite nous procédons à leur intégration.

### 5.4.1. La gestion des modes de marche

#### 5.4.1.1. Modèle des machines virtuelles

Dans l'étude du processus de commande [BOI91] s'intéresse au pilotage des machines et au filtrage des ordres vers le procédé. Le but est d'assurer la commande effective des composants d'une unité de production, en tenant compte des contraintes de comportement existantes (machines à l'arrêt, en marche, en panne, dégradées etc.).

Pour décrire le comportement de chaque machine, un graphe d'état lui est associé. La prise en compte des contraintes entre machines est définie à l'aide de tables de contraintes. La notion de machine virtuelle est introduite : c'est un regroupement de machines (virtuelles ou réelles) dont les comportements sont liés par des contraintes de fonctionnement. Le modèle final est la représentation du système de production sous forme arborescente à partir des contraintes définies, figure 5.4-1. Les contraintes machines doivent empêcher, d'une part que les machines se trouvent dans un état mutuellement incompatible, d'autre part l'envoi d'une séquence d'opérations incohérentes.

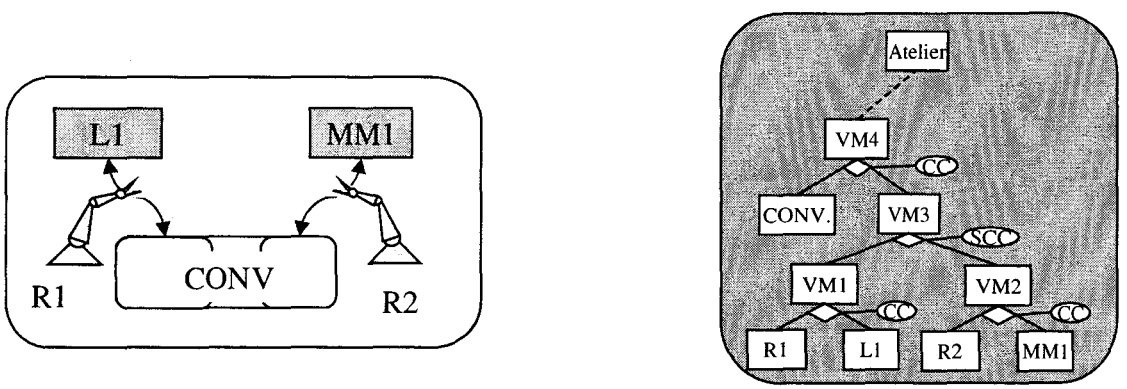



**Figure 5.4-1 : Atelier et machine virtuelle**

### 5.4.1.2. Métamodèle pour la fonction gestion des modes

Le modèle développé dans le cadre du pilotage d'un système de production est basé sur l'étude du comportement des machines. Par une démarche ascendante de regroupement, une décomposition de l'unité de production est effectuée, jusqu'à recouvrir le procédé par une seule machine virtuelle ; le résultat final est une arborescence. Le comportement des machines est décrit à l'aide d'un graphe d'état. Ainsi le métamodèle UML de la fonction gestion des modes de marche est obtenu en utilisant deux patterns : le pattern « composite » et le pattern métier « modèle de comportement », figure 5.4-2.

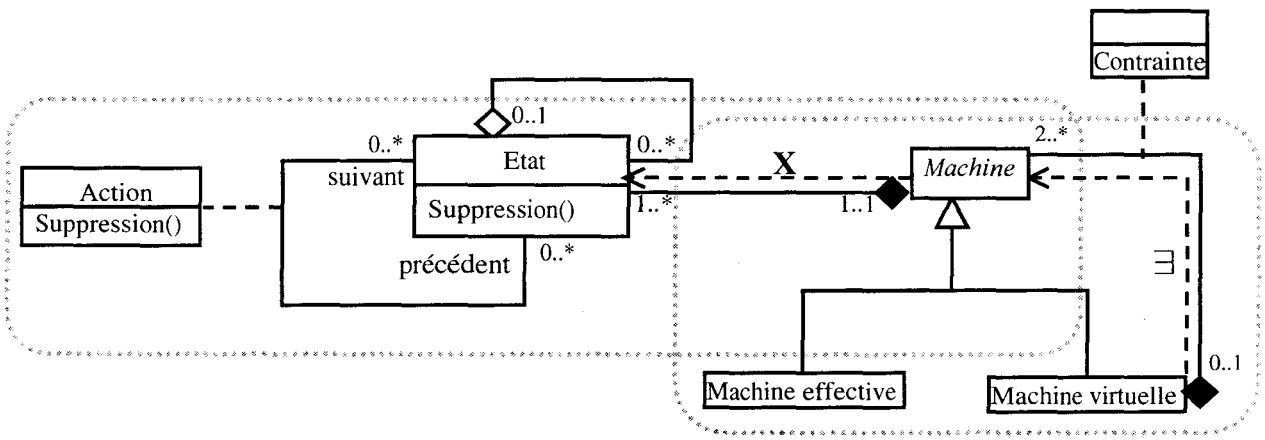



**Figure 5.4-2 : Métamodèle pour la gestion des modes**

Une machine est soit une machine effective soit une machine virtuelle. Une machine effective n'est pas décomposable d'après les spécifications de [BOI91]. Nous utilisons alors le pattern « composite » en distinguant les feuilles (machines effectives) des composites (machines virtuelles). Une machine virtuelle regroupe selon des contraintes de fonctionnement, des machines (virtuelles ou effectives).

La suppression d'une machine (en cas de retrait de la machine du système) entraîne nécessairement une recomposition du modèle ; toutes les machines virtuelles se trouvant sur le chemin entre la machine supprimée et la racine sont alors supprimées à leur tour, d'où la contrainte existentielle entre les classes *Machine* et Machine virtuelle.

Le comportement dynamique de chaque machine est étudié en lui associant un graphe d'état spécifique que nous pouvons représenter par utilisation du pattern métier « modèle de comportement ». Lorsqu'un ensemble d'états possède des relations identiques vis à vis des autres états du système, ils sont regroupés dans un macro-état. Nous rajoutons donc sur la classe Etat une agrégation réflexive puisqu'un état peut regrouper plusieurs autres états.

Une machine se trouve dans un certain nombre d'états au cours de son fonctionnement, cependant elle possède un état unique en un moment donné, d'où la contrainte exclusive entre les classes *Machine* et Etat.

## 5.4.2. Diagnostic

### 5.4.2.1. Modèle structuro-fonctionnel

L'objectif ici est d'obtenir l'automatisation du diagnostic (recherche des causes d'une défaillance). La méthode de conception est basée sur un modèle fonctionnel [TOG92]. Le modèle fonctionnel spécifie les différentes fonctions du système à partir de la composition de fonctions élémentaires. Cette composition est réalisée en définissant les liens fonctionnels (coopérations et échanges entre composants) entre fonctions.

La conception du modèle fonctionnel d'un système se fait à partir de son modèle structurel qui consiste à décomposer le système en composants, jusqu'à l'obtention de composants de base tels que les capteurs directs. Ensuite, une spécification fonctionnelle par composant est effectuée à partir d'une liste des fonctions utiles du système. Enfin, les liens de dépendance fonctionnelle entre les fonctions des composants sont définis afin d'obtenir le modèle fonctionnel final. Ce modèle montre pour chaque fonction, les fonctions mères (celles qui l'utilisent) et les fonctions filles (celles qui la produisent).

Le diagnostic s'appuie alors sur un modèle structuro-fonctionnel comme indiqué sur la figure 5.4-3 :

- concernant le modèle fonctionnel : la fonction production utilise les fonctions fraisage, tournage et taraudage ;
- dans le modèle structurel : la cellule est composée d'un robot des machines 1 et 2 et d'un convoyeur ;
- les composants du modèle structurel assurent des fonctions disponibles dans le modèle fonctionnel ; la cellule a une fonction de production, la machine-1 assure la fonction Usinage-1 etc.

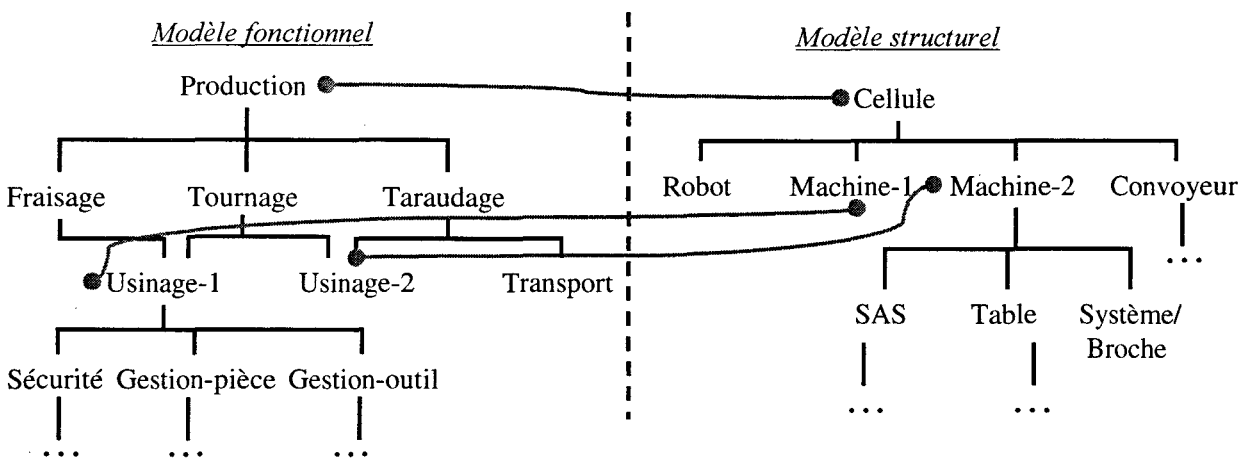



**Figure 5.4-3 : Exemple de modèles structurel et fonctionnel d'une cellule**

### 5.4.2.2. Métamodèle pour la fonction diagnostic

Le métamodèle UML de la fonction diagnostic est construit en utilisant les patterns métier « modèle fonctionnel » et « modèle structurel », figure 5.4-4.

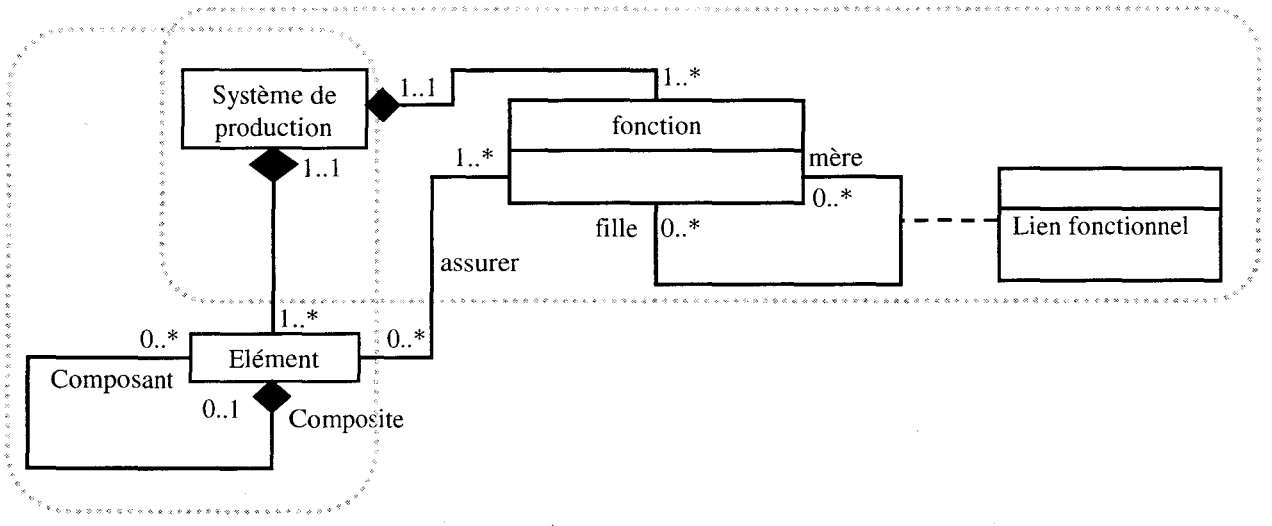



**Figure 5.4-4 : Métamodèle pour la fonction de diagnostic**

Un système de production est composé de plusieurs éléments, un élément appartient à un seul et unique système. Ces éléments à leur tour sont décomposables en plusieurs autres éléments sous une forme arborescente, la finesse de cette décomposition est laissée au choix du concepteur. L'ensemble de ces éléments et leurs liens structurels constituent un modèle structurel du système étudié.

Une deuxième analyse du système fait apparaître l'ensemble de ses fonctions et leurs liens de dépendances. Une fonction peut utiliser d'autres fonctions pour être réalisée, de même elle peut être utilisée dans plusieurs autres fonctions. Ces relations entre fonctions sont définies par des liens fonctionnels. Le pattern métier « modèle fonctionnel » est utilisé pour décrire les fonctions du système et leurs relations.

Le lien entre les modèles structurel et fonctionnel est établi entre les classes Elément et Fonction. Un élément peut assurer une à plusieurs fonctions disponibles dans le système ; la même fonction peut être assurée par zéro (en cas de non disponibilité par exemple) ou plusieurs éléments.

## 5.4.3. Surveillance de la commande

### 5.4.3.1. Modèle des objets commandables

Dans le cadre de la surveillance des systèmes à événements discrets, [ELK93] propose un module des filtres de commande. Ce module doit éviter la réalisation de commandes incompatibles avec l'état du procédé ; il est basé sur le concept d'Objet Commandable.

Les objets commandables sont considérés comme un modèle exprimant le comportement d'un composant physique. Un objet est dit commandable si à partir d'un état quelconque, nous accédons à tous les autres états en réalisant une suite d'actions, figure 5.4-5. L'objet doit avoir un comportement déterministe vis-à-vis de toutes les actions réalisables. L'état représente l'état physique stable de l'objet commandable à un instant donné, l'action permet de passer d'un état à un autre de façon instantanée, une action est une commande élémentaire.

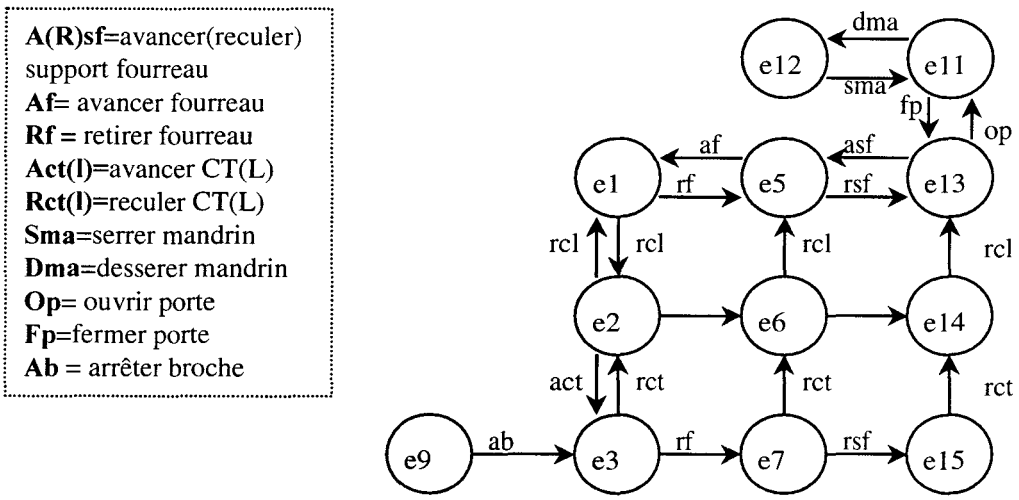



**Figure 5.4-5 : Extrait d'un modèle comportemental**

Les Objets Commandables Elémentaires (OCE, non décomposables) sont regroupés suivant des contraintes fonctionnelles par paires et conduisent à la création d'un nouveau composant : le Composant Fonctionnel Logique (CFL). Le CFL obtenu est considéré à son tour comme un objet commandable. Le regroupement se termine lorsqu'il n'y a plus de contraintes à exploiter. Les contraintes se formulent par l'interdiction de réaliser une action d'un composant quand un autre composant se trouve dans un état particulier.

### 5.4.3.2. Métamodèle pour la surveillance de la commande

Etant donné que les OCE et les CFL sont étudiés suivant le même modèle comportemental, la super-classe *Composant* est une abstraction qui nous évite de dupliquer le pattern métier « modèle de comportement » au niveau des classes OCE et CFL.

Un composant est considéré :

- d'une part comme un graphe orienté dont les arcs sont des actions reliant deux états ;
- d'autre part comme un composite dont les feuilles sont des OCE, les CFL étant les éléments composites.

Nous établissons alors le métamodèle pour le module des filtres de commande en utilisant le pattern « composite » et le pattern métier « modèle de comportement », figure 5.4-6.

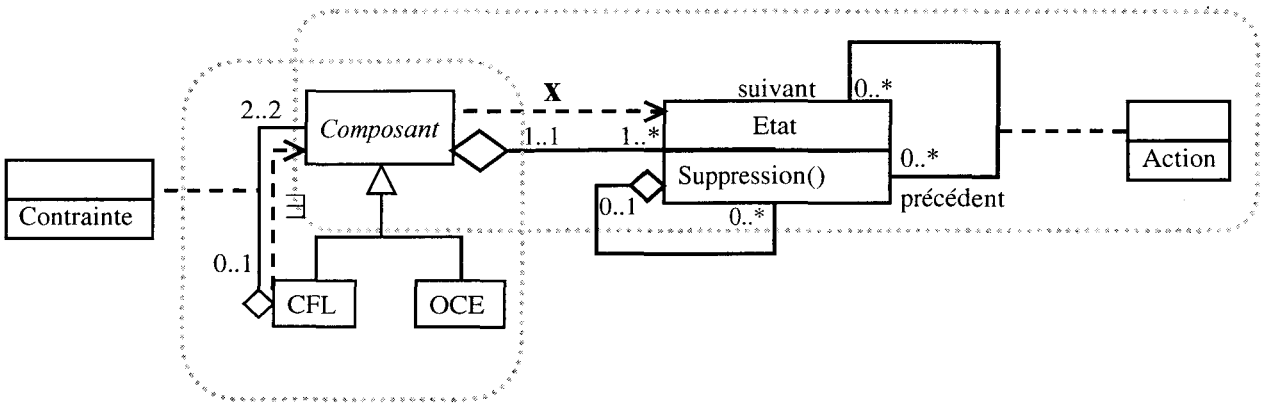



**Figure 5.4-6 : Métamodèle pour le module des filtres de commande**

Un CFL regroupe des composants (OCE ou CFL) par paires sous contrainte. Un composant peut participer à plusieurs regroupements ; prenons l'exemple d'un tour : (1) la broche doit être à l'arrêt lors de l'ouverture de la porte, (2) la broche doit être à l'arrêt lors du chargement/déchargement du mandrin. Ces deux contraintes broche-porte et broche-mandrin conduisent à la création de deux CFL. Un CFL est un objet composite, mais ne présente pas une structure arborescente car un nœud peut avoir plusieurs pères, nous avons ainsi adapté le pattern « composite » à cette situation en mettant des multiplicités 0..* du coté de la classe CFL. La suppression d'un composant entraîne une nouvelle recomposition et un nouveau CFL, d'où la contrainte existentielle entre les classes CFL et *Composant*.

Un composant va se trouver dans un certain nombre d'états selon les actions qu'il subit, mais il se trouve dans un seul et unique état physique stable à un moment donné. Pour cette raison, nous rajoutons une contrainte exclusive entre les classes *Composant* et Etat. Les états de plusieurs OC présentant un comportement similaire sont agrégés dans des macro-états, d'où l'agrégation réflexive sur la classe Etat.

## 5.4.4. Recouvrement

### 5.4.4.1. Le graphe d'accessibilité opérationnelle

Le recouvrement détermine un nouvel état du système de production, afin d'assurer la disponibilité du système de production malgré la présence d'une panne matérielle.

L'approche proposée par [BER98] vise à modéliser la partie opérative (description d'un système de production sous forme d'opérations) ; pour cela, il a construit un modèle appelé le Graphe d'Accessibilité Opérationnelle (GAO), figure 5.4-7. Ce modèle exprime la flexibilité de l'architecture d'un SFPM, en montrant l'ensemble des opérations réalisables et accessibles les unes après les autres.

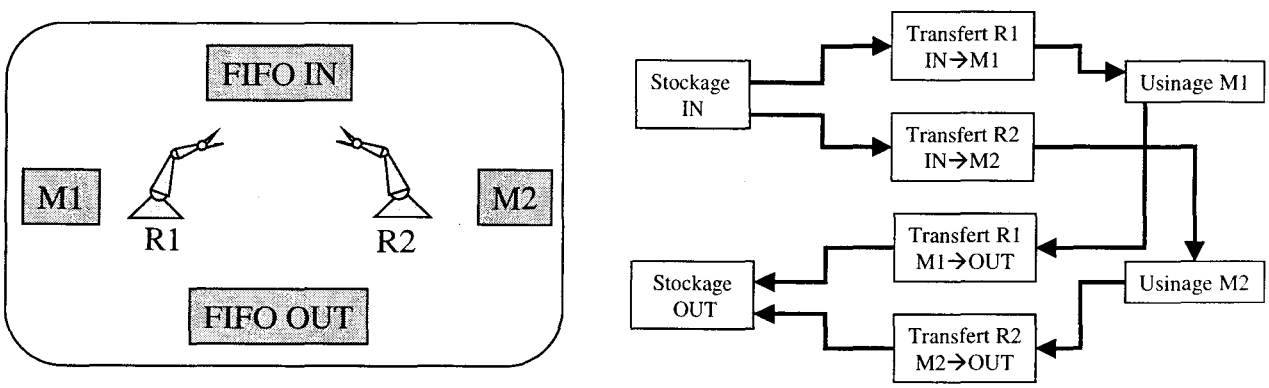



**Figure 5.4-7 : GAO d'un atelier flexible**

Les **propriétés** structurelles suivantes caractérisent le GAO :

(1) Le GAO est un graphe orienté dont les nœuds d'entrée et de sortie sont dissociés par construction.

(2) Il existe **deux types de nœuds** : des nœuds de transfert (IN→M1, IN→M2, M1→OUT, M2→OUT), des nœuds stationnaires (Stockage IN, Stockage OUT, Usinage M1, Usinage M2).

(3) Deux nœuds de même nature ne peuvent être directement reliés.

(4) Il ne possède pas d'arcs bidirectionnels ; le retour vers un nœud précédent devra se faire en passant obligatoirement par un autre.

(5) Seuls les nœuds stationnaires peuvent avoir plusieurs nœuds en amont ou plusieurs nœuds en aval.

(6) Un nœud de transfert n'aura qu'un seul nœud stationnaire en amont et qu'un seul stationnaire en aval.

(7) Un nœud sera composé d'une ou plusieurs opérations de type transfert ou type stationnaire.

### 5.4.4.2. Métamodèle du GAO

D'après les propriétés (2) et (3), le GAO est typiquement un graphe orienté biparti, comprenant deux types de nœuds, des opérations de transfert et des opérations stationnaires, ces nœuds sont reliés par des relations d'accessibilité. Le métamodèle du GAO sur la figure 5.4-8 est ainsi construit en utilisant le pattern « graphe orienté biparti ».

Conformément aux propriétés (5) et (6) citées précédemment, une Opération de transfert est précédée (suivie) par une seule Opération Stationnaire. A l'inverse, une Opération Stationnaire peut avoir plusieurs Opérations de transferts comme prédécesseurs ou successeurs.

Les opérations (de transfert ou stationnaires) peuvent être agrégées comme l'indique la propriété (7). Ce sera le cas des opérations réalisées par une ressource polyvalente. D'où l'adjonction de deux agrégations réflexives sur les classes Opération de transfert et Opération stationnaire.

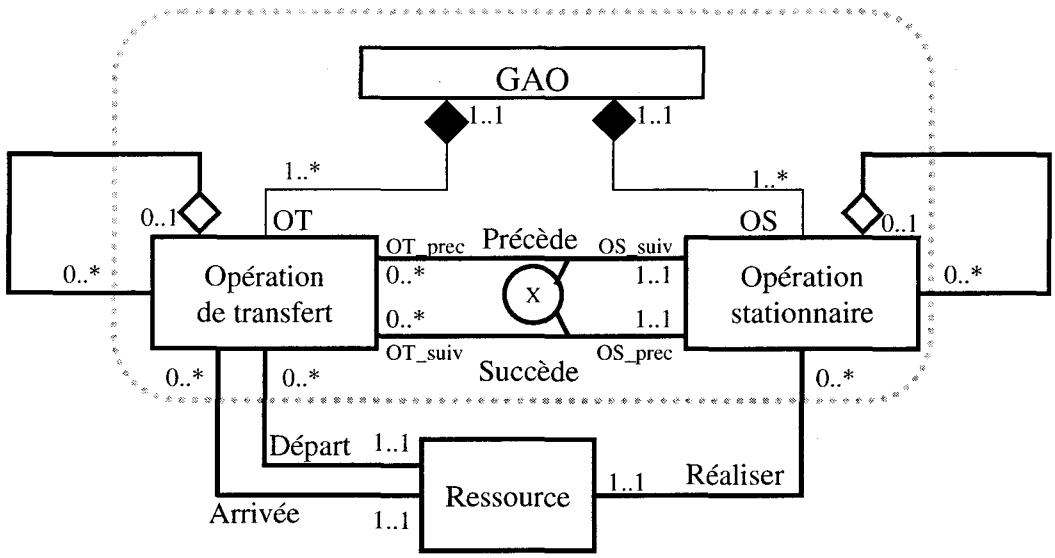



**Figure 5.4-8 : Métamodèle du GAO**

Pour respecter la propriété (4), à savoir qu'il n'y a pas de nœuds bidirectionnels (si l'arc n1→n2 existe alors l'arc n2→n1 n'existe pas et vice-versa), nous rajoutons la contrainte exclusive entre les deux associations Précède et Succède. Cette contrainte équivaut aux deux contraintes OCL suivantes.

```
Context Opération stationnaire inv :
self.OT_prec→intersection(self.OT_suiv)→isEmpty

Context Opération de transfert inv :
self.OS_prec <> self.OS_suiv
```

## 5.4.5. La maintenance

### 5.4.5.1. Graphe équivalent d'un système de production

Dans le cadre de la maintenance, [LY99] propose une stratégie de maintenance préventive basée sur le suivi indirect de l'état des ressources en surveillant le flux des produits fabriqués. Ce flux est mesuré à l'aide de capteurs physiques et de capteurs logiques qui délivrent une information à partir de la synthèse d'informations provenant des capteurs physiques.

L'analyse commence en déterminant la position des capteurs à placer sur le convoyeur, pour mesurer les flux induits par les ressources et les points de changement de direction (aiguillages et jonctions) du convoyeur. La méthode de placement des capteurs est basée sur la théorie des graphes ; le système de production est alors considéré comme un graphe orienté, figure 5.4-9. Les machines, robots, jonctions et aiguillages représentent les sommets du graphe. Les arcs orientés (selon le sens de circulation des pièces dans le système) correspondent aux tronçons de convoyeurs et aux opérations de transfert effectuées par les robots.

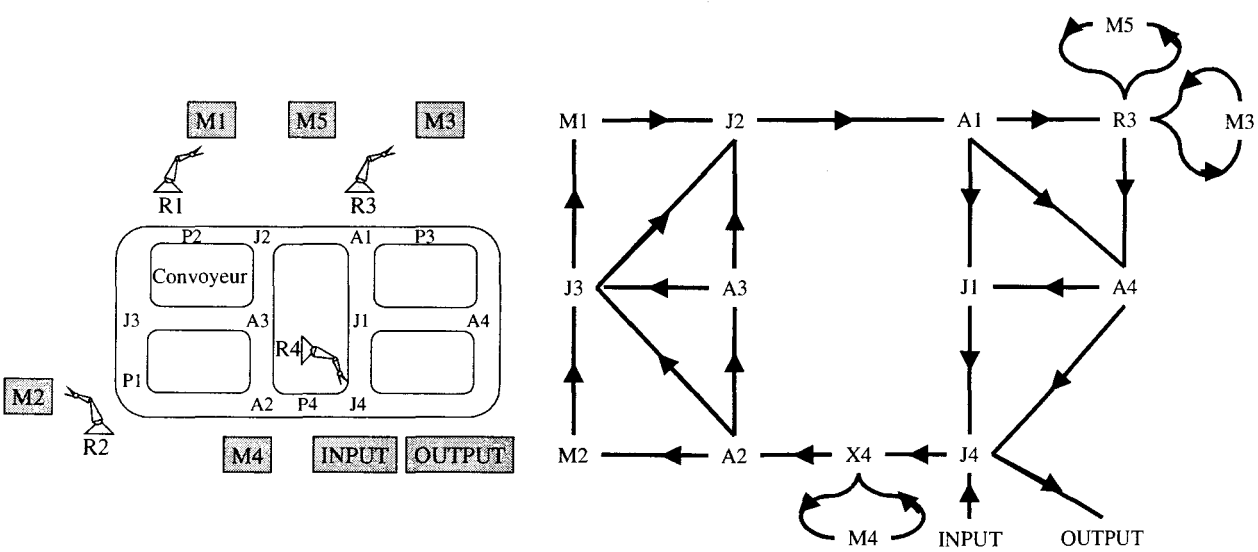



**Figure 5.4-9 : Atelier flexible et son graphe équivalent**

Parmi ces sommets, un sous-ensemble est choisi pour placer les capteurs. Par exemple sur le graphe ci-dessus, le concepteur a choisi (par la méthode dite de Maghout) les sommets suivants : {Output, J2, J3, R3, X4, J4}.

Cette phase de placement des capteurs conduit à la première étape de la maintenance prédictive : la détection qui est appliquée à chaque gamme opératoire. Ensuite l'ensemble des ressources potentiellement défaillantes est localisé grâce au diagnostic. Enfin un pronostic est effectué pour évaluer l'impact d'une dégradation sur le flux de production et le temps nécessaire avant qu'une intervention ne soit obligatoire.

### 5.4.5.2. Métamodèle pour la maintenance

Un système de production est composé de plusieurs éléments décomposables. Cette décomposition structurelle des systèmes est représentée à l'aide du pattern métier « modèle structurel ».

A chaque système, on associe un seul et unique Graphe équivalent. De même un Graphe équivalent décrit un seul Système de production. Ce graphe contient des sommets qui sont reliés par des arcs qui sont les chemins de transfert des produits. Ce graphe est obtenu en utilisant le pattern « graphe orienté ». Chaque sommet du graphe représente un à plusieurs éléments du Système de production ; par exemple les robots R1, R2, et R4 ont été respectivement agrégés avec les machines M1, M2, et M4 auxquelles ils sont dédiés. Un Elément est associé à zéro ou un Sommet.

Il existe deux sortes de capteurs : des capteurs physiques placés sur des points d'observation, et les capteurs logiques qui synthétisent l'information des capteurs physiques. La description de ces composites est faite en utilisant le pattern « composite » dont les feuilles sont des capteurs physiques. Les informations fournies par un capteur logique dépendent de celles fournies par chacun des capteurs le constituant ; nous rajoutons alors une contrainte existentielle pour exprimer le fait que la suppression d'un capteur entraîne celle du capteur qui l'utilise. Un sommet correspondant à une ressource de transformation n'est pas un point d'observation, donc sur chaque sommet on peut placer zéro ou un capteur.

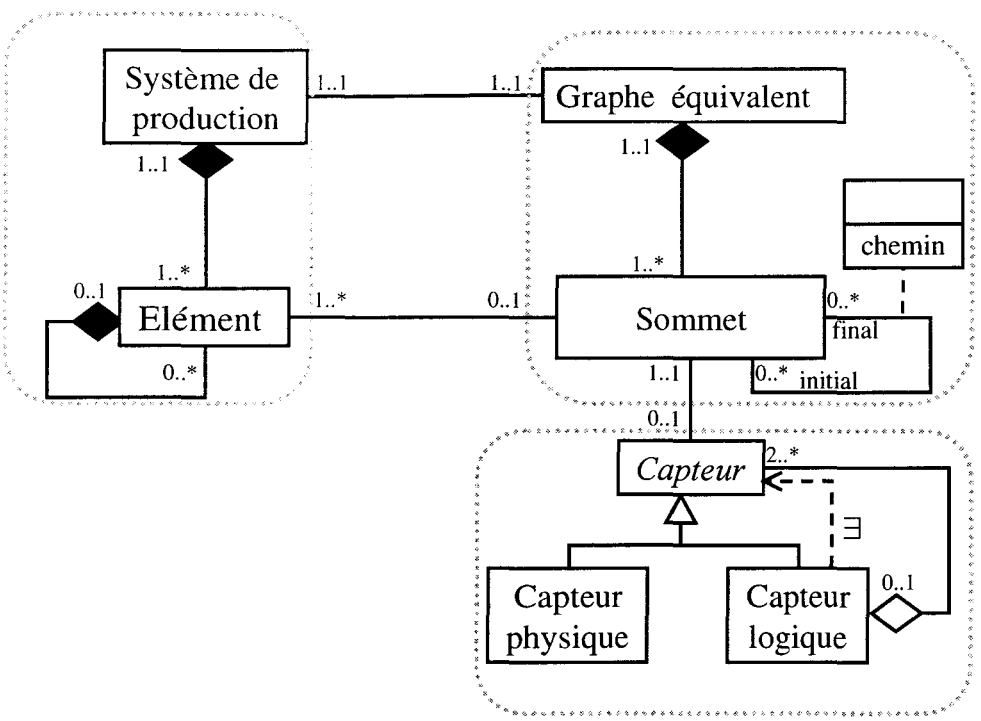



**Figure 5.4-10 : Métamodèle pour la maintenance**

### 5.4.6. Le métamodèle UML du point de vue Supervision

Nous avons établi les métamodèles UML des cinq fonctions qui interviennent du point de vue Supervision, nous allons à présent intégrer ces diagrammes de classes afin d'obtenir le métamodèle UML pour la supervision.

Quatre modèles (dont trois patterns métiers) sont utilisés par les intervenants, comme indiqué sur la figure 5.4-11. Le GAO est basé sur les relations d'accessibilité définies entre les ressources du système de production, par conséquent il utilise le modèle structurel. Le Modèle de comportement procède par regroupement de composants liés par des contraintes de fonctionnement ; il utilise pour cela aussi bien le modèle fonctionnel que le modèle structurel. Quant au Graphe équivalent, il est directement dérivé de l'architecture physique du système de production.

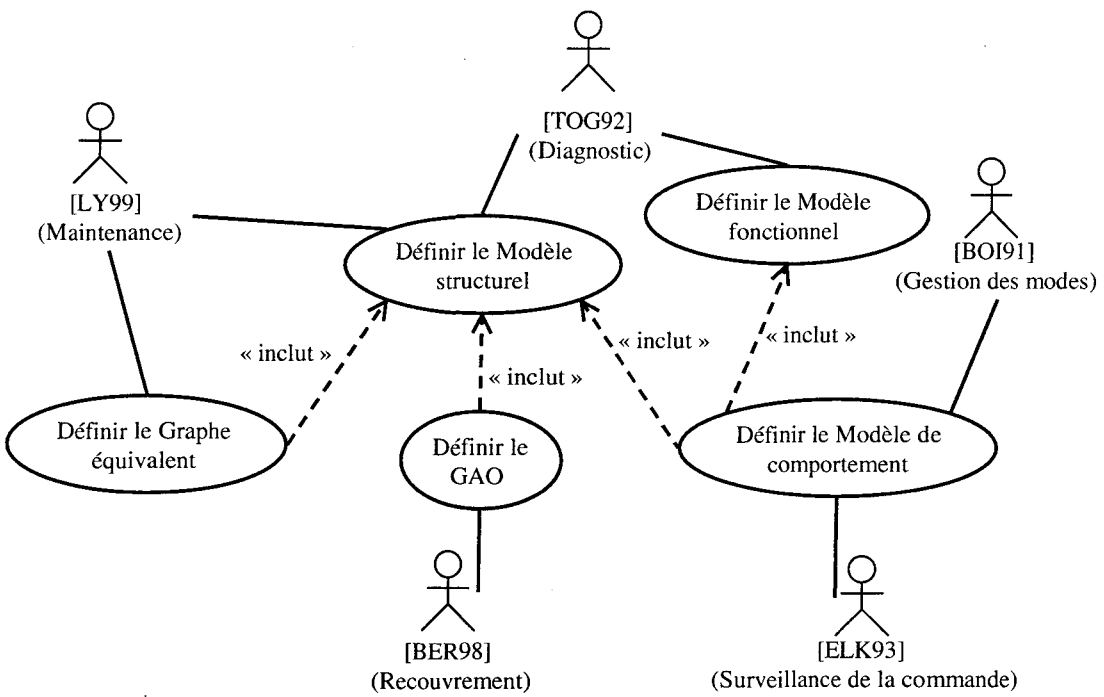



**Figure 5.4-11 : Cas d'utilisation des modèles de la supervision**

Ce diagramme de cas d'utilisation montre que le modèle structurel reste le modèle le plus utilisé. De ce fait nous allons d'abord étudier les liens entre les différents modèles avec celui-ci afin d'intégrer les différents diagrammes de classes correspondant au métamodèle de chacune de ces cinq fonctions. De même nous constatons que les acteurs [BOI91] et [ELK93 ] utilisent tous les deux le pattern métier « Modèle de comportement », il y a alors une forte probabilité de retrouver des points de convergence entre leurs modèles respectifs. Il en est de même pour [LY99] et [TOG92] concernant le modèle structurel. A partir de ces analyses nous établissons le diagramme de classes sur la figure 5.4-12.

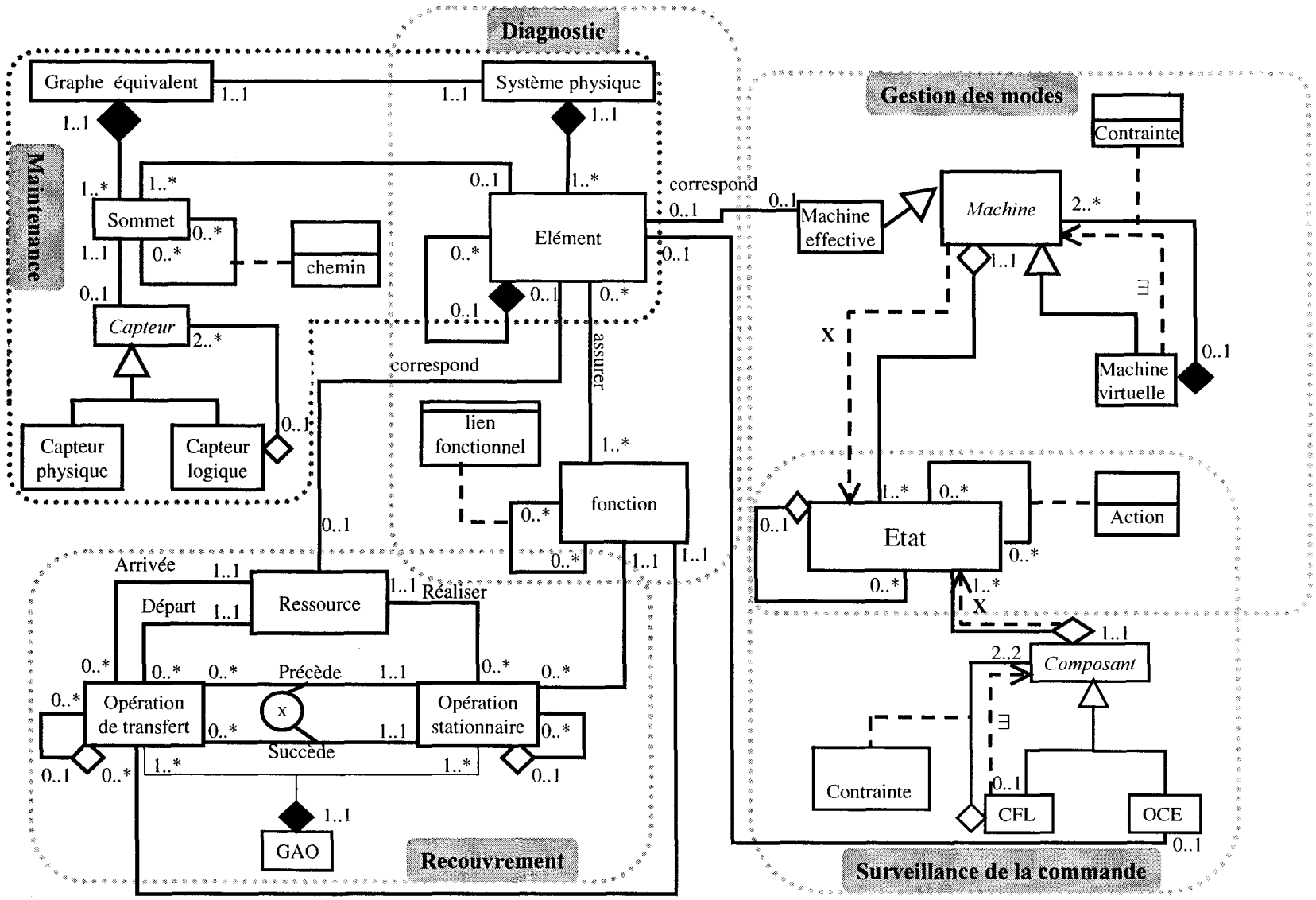



**Figure 5.4-12 : Métamodèle pour le point de vue supervision**

**Liens entre la maintenance et le diagnostic**

Entre les métamodèles de ces deux fonctions, il apparaît immédiatement un lien à travers le pattern métier « modèle structurel ». Les classes du modèle structurel se retrouvent de part et d'autre des deux métamodèles avec la même sémantique et les mêmes associations. Il nous suffit alors de conserver le modèle structurel tel quel et de garder les mêmes associations avec les autres classes. La relation d'utilisation entre le Graphe équivalent et le Modèle structurel a été expliqué dans le § 5.4.5.

**Liens entre la gestion des modes et la surveillance de la commande**

Les intersections entre les métamodèles des ces deux fonctions sont aussi assez évidentes, du fait qu'ils utilisent tous les deux le pattern métier « modèle comportement ». L'analyse que nous avons fait ci-dessus reste valable. Le comportement des composants et des machines est décrit à partir du même modèle comportemental.

**Liens entre le recouvrement et le diagnostic**

Le GAO est basé sur le concept de relations d'accessibilité qui sont déterminées d'après l'agencement des ressources du système de production étudié. Le GAO dépend donc du modèle structurel. Une Ressource correspond à zéro ou un Elément de la décomposition structurelle. La multiplicité minimale est égale à zéro car il se peut que la décomposition structurelle effectuée dans le point de vue diagnostic ne contienne pas la ressource étudiée. Car chaque point de vue effectue sa propre décomposition et regroupement des éléments d'un système. De même, un Elément correspond à zéro ou une ressource.

Une fonction disponible dans le système peut être utilisée plusieurs fois ; elle définit alors les Opérations de transfert et les Opérations stationnaires. Par exemple, une fonction fraisage, correspond aux opérations stationnaires fraisage_1 sur M1 et fraisage_2 sur M2.

**Liens entre la gestion des modes et le diagnostic**

Le métamodèle de la gestion des modes montre un ensemble de machines qui sont regroupées par des contraintes de fonctionnement. La classe Machine effective représente les machines réelles du système de production, donc elle correspond à un Elément du modèle structurel ; d'où l'association entre les classes Machine effective et Elément. Les multiplicités 0..1 sont choisies pour des raisons identiques à celles expliquées ci-dessus.

**Liens entre la surveillance de la commande et le diagnostic**

La surveillance de la commande repose sur l'étude du comportement de composants qui sont regroupés suivant des contraintes. Les CFL sont des composants virtuels, contrairement aux OCE. Un Elément du modèle structurel correspond alors à zéro ou un OCE ; de même un OCE correspond à zéro ou un Elément.

## 5.5. Le point de vue Planification/Ordonnancement

Ce point de vue s'intéresse à l'élaboration d'une commande prévisionnelle déterministe et cyclique des SFPM, à l'aide des réseaux de Petri [CAM97], [KOR98], [OHL95]. L'évaluation des performances est effectuée en phase de conception tandis que la planification/ordonnancement intervient en phase d'exploitation. Le problème consiste à utiliser toutes les flexibilités que recèlent les SFPM, afin d'optimiser la production à partir de la demande qui est à réaliser. Nous allons successivement étudier la modélisation du problème initial puis le modèle de la commande établi sous forme de graphe d'événement.

### 5.5.1. Modélisation du problème initial

#### 5.5.1.1. Gamme opératoire

Les gammes opératoires sont utilisées pour modéliser les séquences d'opérations pour la fabrication des produits. Trois gammes opératoires (GO_A, GO_B, GO_C) sont définies sur la figure 5.5-1, les ratios de production ($r_A$, $r_B$ et $r_C$) sont fixés pour chacune d'elles. Les ratios de routages sont ensuite déterminés pour les différentes branches de chaque gamme opératoire. Chacune des ressources ($R_1 \ldots R_6$) regroupe un ensemble de machines pouvant effectuer les mêmes opérations dont les durées opératoires sont exprimées par une temporisation des transitions. Deux familles de palettes sont utilisées pour le transport des pièces ; la première famille est dédiée exclusivement à la gamme GO_A, la deuxième aux gammes GO_B et GO_C.

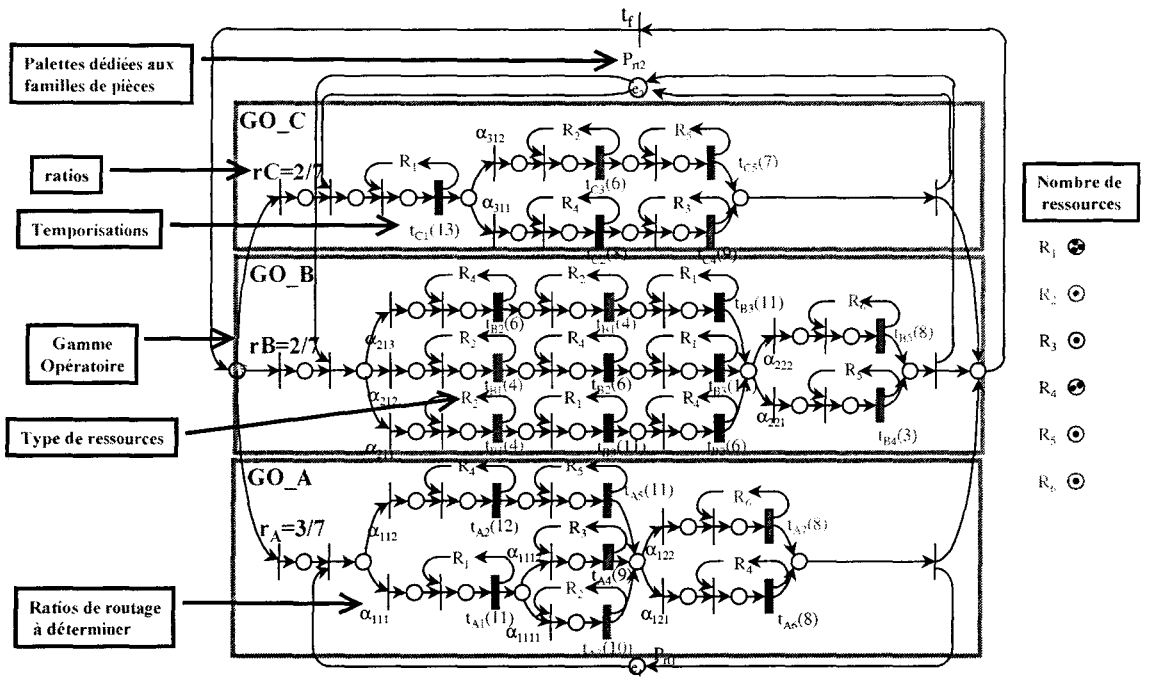



**Figure 5.5-1 : Réseau de Petri modélisant le problème initial**

#### 5.5.1.2. Métamodèle du RdP initial

La première étape de modélisation utilise les gammes opératoires (réutilisation du pattern « graphe orienté biparti », cf. § 5.2.2), figure 5.5-2. Cependant seule la succession des opérations intéresse les concepteurs, la durée des opérations est prise en compte par une temporisation des transitions. De même les opérations de transferts ne sont pas prises en compte par les concepteurs de ce point de vue. Une gamme opératoire fabrique un seul type de produit, de même un type de produit n'est fabriqué que par une seule gamme opératoire. Le transport d'un type de produit est assuré par une seule et unique famille de palettes.

Dans une production, plusieurs cycles de fonctionnement sont définis et plusieurs types de produits sont fabriqués, ces derniers pouvant bien entendu être concernés par plusieurs productions. Une opération est réalisable sur une seule ressource, cette dernière pouvant effectuer plusieurs opérations.

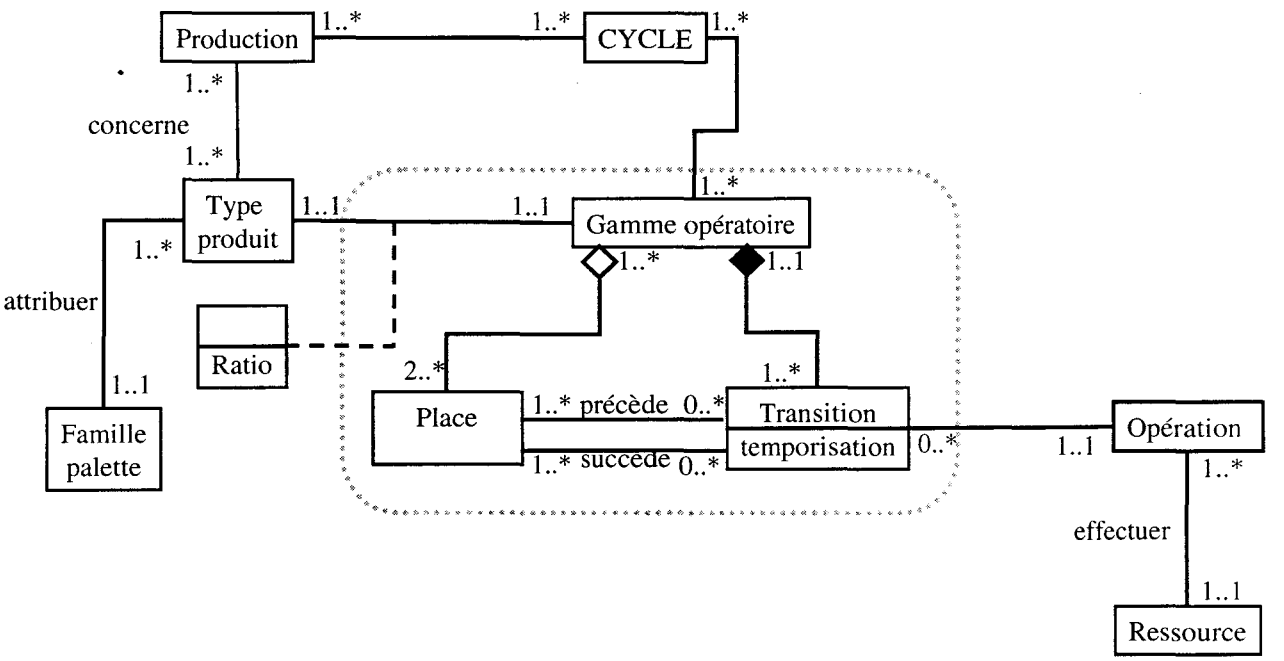



**Figure 5.5-2 : Métamodèle de la gamme initiale**

### 5.5.2. Modélisation de la commande

#### 5.5.2.1. Graphe d'événement

Le RdP modélisant le problème initial exprime toutes les flexibilités de fabrication des produits. A partir de ce modèle, une commande déterministe sous forme de graphe d'événement est réalisée en phase d'ordonnancement.

Tous les indéterminismes dus aux flexibilités d'affectation (la même opération réalisable par plusieurs ressources) de permutation (gammes partiellement ordonnées) et de procédé (substitution d'une suite d'opérations par une autre suite d'opérations) sont levés.

Les gammes opératoires sont linéarisées et regroupées de manière séquentielle dans des macro-gammes (MG1 et MG2). Une famille de palettes est associée à chaque macro-gamme.

Sur la figure 5.5-3 nous représentons le graphe d'événement final obtenu à partir du modèle initial de la figure 5.5-1.

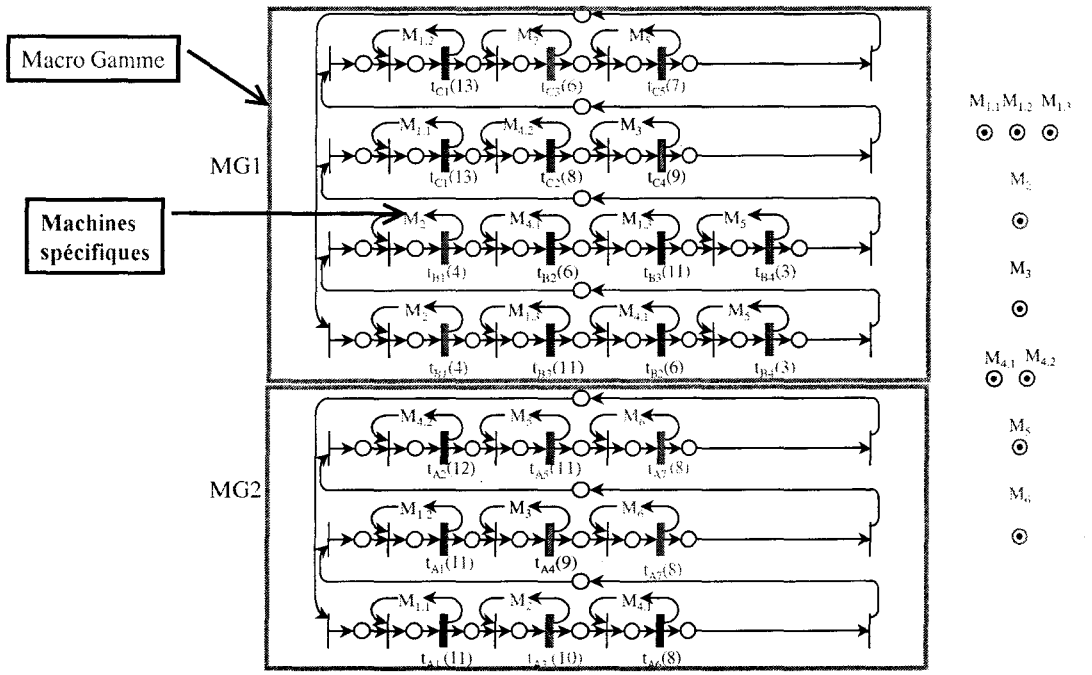



**Figure 5.5-3 : Commande par graphe d'événement**

### 5.5.2.2. Métamodèle du graphe de commande

Ce métamodèle est conçu par réutilisation du pattern « graphe orienté biparti », figure 5.5-4.

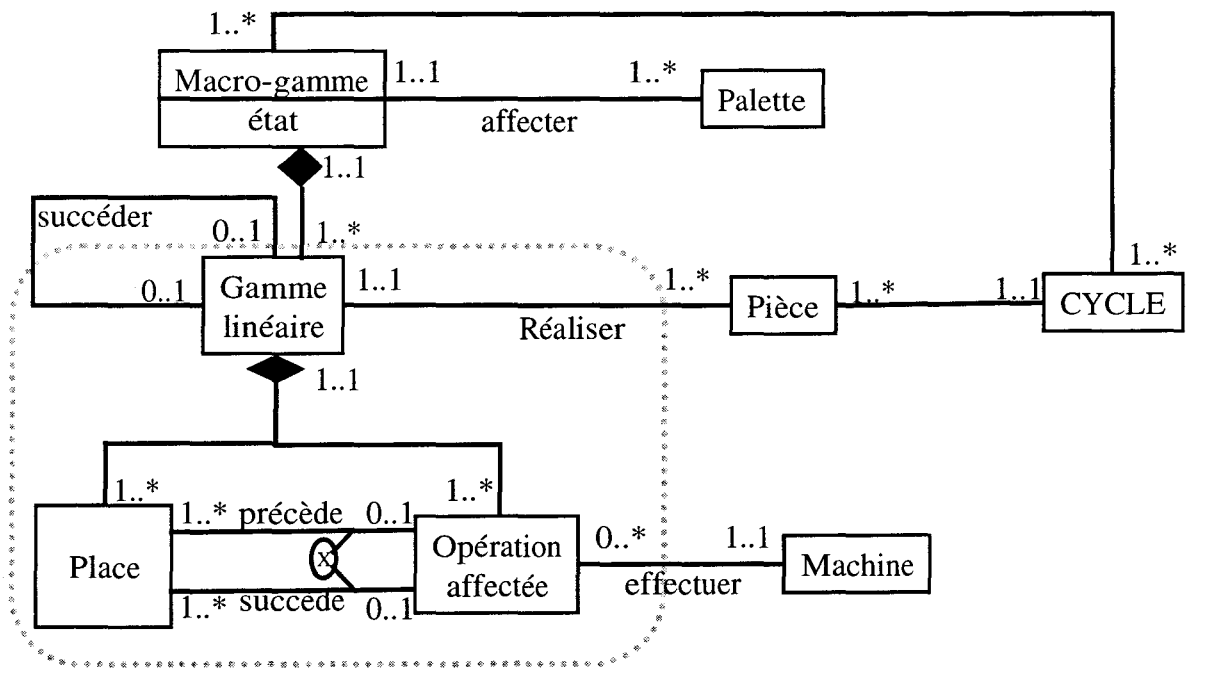



**Figure 5.5-4 : Métamodèle du graphe de commande**

Une gamme linéaire correspond à plusieurs pièces à réaliser durant un cycle, une pièce n'est réalisable que dans une seule et unique gamme linaire. Chaque opération de la gamme linaire est affectée à une machine déterminée, qui peut réaliser plusieurs opérations.

Dans une gamme linéaire, une place est suivie de zéro ou une opération, les nœuds bidirectionnels sont interdits, d'où la contrainte exclusive entre les associations précède et succède.

Une macro-gamme est un regroupement ordonné de gammes linéaires, de sorte qu'une place est toujours suivie d'une opération. Nous définissons alors une nouvelle contrainte OCL pour cela.

```
Context Place inv :
Self.ope_prec→union(self.ope_succ)→notEmpty
```

Une palette est affectée à une seule macro-gamme, et plusieurs palettes sont utilisées dans une macro-gamme. Pour un cycle de production, plusieurs pièces sont fabriquées et plusieurs macro-gammes sont déterminées. Une pièce est fabriquée lors d'un seul et unique cycle de production.

Nous allons à présent intégrer ces deux métamodèles pour obtenir le métamodèle complet du point de vue Planification/Ordonnancement, et faciliter par la même, la tache d'intégration (effectuée au § 5.6) de tous les métamodèles de la méthode CASPAIM.

### 5.5.3. Planification/Ordonnancement : Métamodèle final

Dans la conception de la commande, deux modèles interviennent : un modèle pour le problème initial et un autre pour la commande déterministe. La conception du modèle de commande nécessite la construction de gammes linaires. De même le modèle initial est obtenu à l'aide de gammes opératoires figure 5.5-5. Les gammes linéaires sont créées à partir des gammes opératoires.

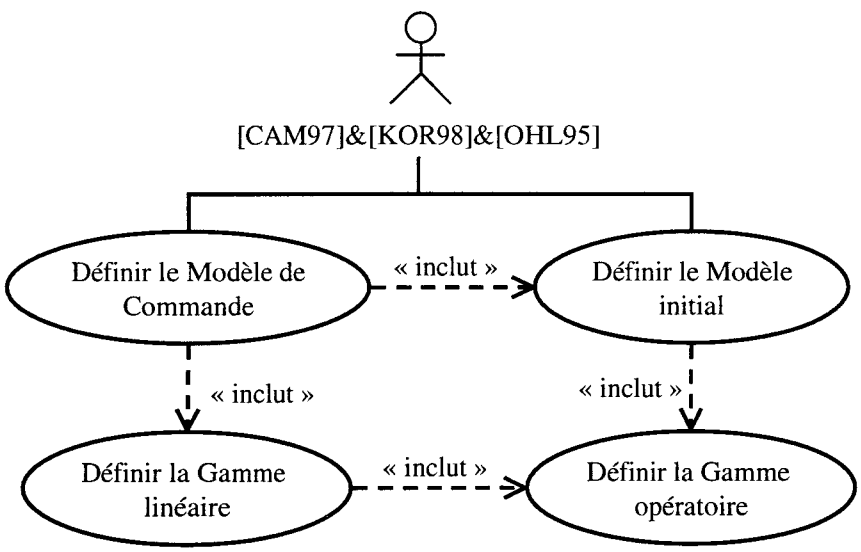



**Figure 5.5-5 : Diagramme de cas d'utilisation de la planification/ordonnancement**

Le diagramme de classes sur la figure 5.5-6 représente le métamodèle complet du point de vue Planification/Ordonnancement. Nous l'avons établi en étudiant les intersections entre les modèles précédemment cités.

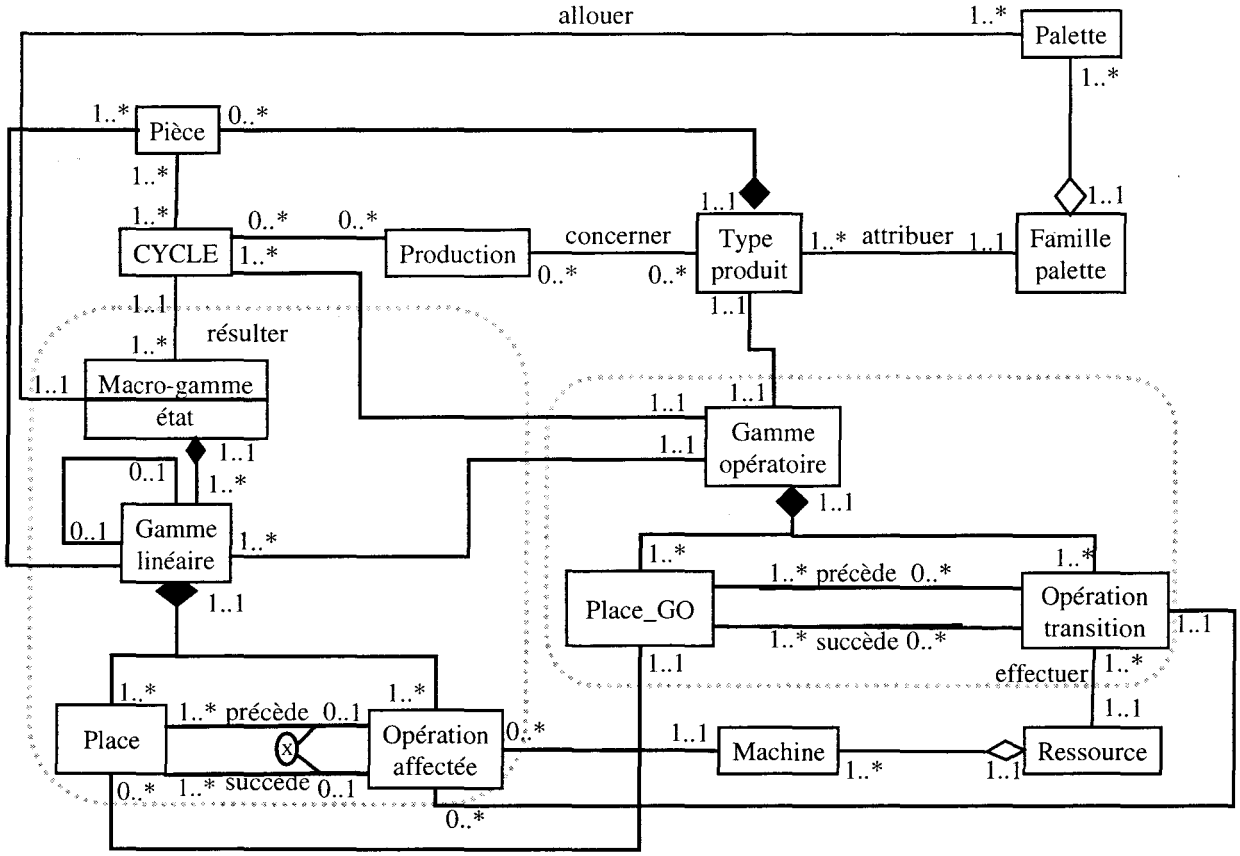



**Figure 5.5-6 : Métamodèle complet du point de vue Planification/Ordonnancement**

**Liens entre gamme linéaire et gamme opératoire**

Une gamme opératoire (opération transition, place_GO) peut générer une à plusieurs gammes linéaires (opération affectée, place) ; chaque gamme linéaire (opération affectée, place) est issue d'une seule gamme opératoire (opération transition, place_GO).

**Liens entre le modèle de commande et le modèle RdP initial**

Une famille de palettes du modèle initial regroupe les palettes associées à une macro-gamme.

Un type de produit associé à une gamme opératoire peut se décomposer en plusieurs pièces qui sont affectées à une gamme linéaire.

Chaque ressource est définie par regroupement de machines semblables.

Le cycle de production est déterminé une fois, il est associé à la gamme opératoire et à la macro-gamme.

## 5.6. Métamodèle intégré CASPAIM

Quatre points de vue sont abordés dans le projet CASPAIM, le référentiel doit fournir une base de données unique pour la gestion des informations partagées et échangées entre ces points de vue, figure 5.6-1.

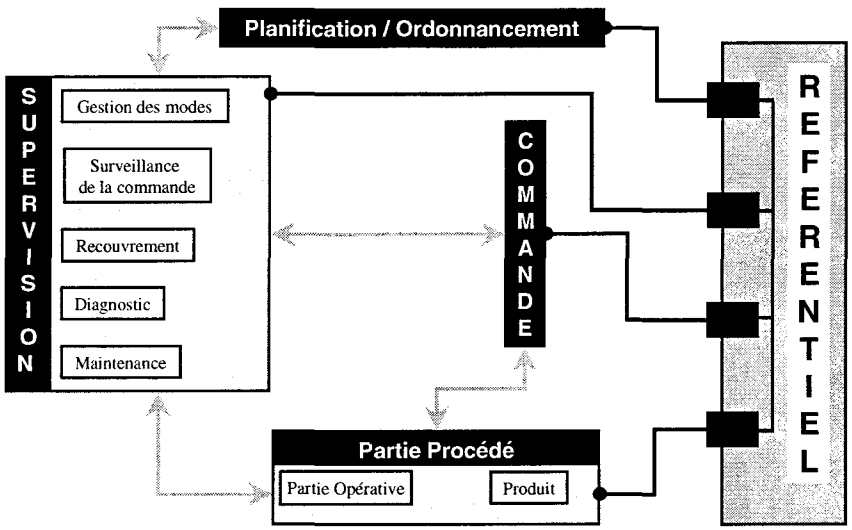



**Figure 5.6-1 : Différents points de vue dans CASPAIM**

**Phase de métamodélisation**

Nous avons déjà créé les métamodèles UML des différents points de vue. Nous avons par ailleurs intégré les points de vue Commande et Partie procédé (cf. § 5.3). De même, nous avons intégré les différents métamodèles des fonctions de la supervision (cf. § 5.4). Ensuite nous avons construit le métamodèle UML du point de vue planification/ordonnancement (cf. § 5.5).

Nous avons donc obtenu trois diagrammes de classes que nous devons maintenant intégrer à leur tour, pour obtenir le métamodèle intégré final du projet CASPAIM.

Nous allons procéder de la même façon que pour l'intégration des métamodèles des fonctions d'un point de vue à savoir :

- Etablir d'abord le diagramme des cas d'utilisation des modèles pour le projet CASPAIM
- Construire le diagramme de classes en étudiant les intersections des différents modèles, grâce aux renseignements fournis par le diagramme des cas d'utilisation.

Le diagramme des cas d'utilisation sur la figure 5.6-2, représente ainsi tous les intervenants du projet et les différents modèles qu'ils doivent concevoir.

**Liens entre Commande/Partie procédé ←→ Planification/Ordonnancement**

Les intersections entre les points de vue Planification/Ordonnancement et Commande/Partie procédé se trouvent au niveau de la gamme opératoire.

**Liens entre Commande/Partie procédé ←→ Supervision**

De même le modèle structurel servira de base d'étude des points de convergence entre les points de vue Commande/Partie procédé et Supervision.

**Liens entre Planification/Ordonnancement ←→ Supervision**

Enfin, la gamme opératoire servira à nouveau de point d'entrée pour l'intégration des points de vue Supervision et Planification/ordonnancement.

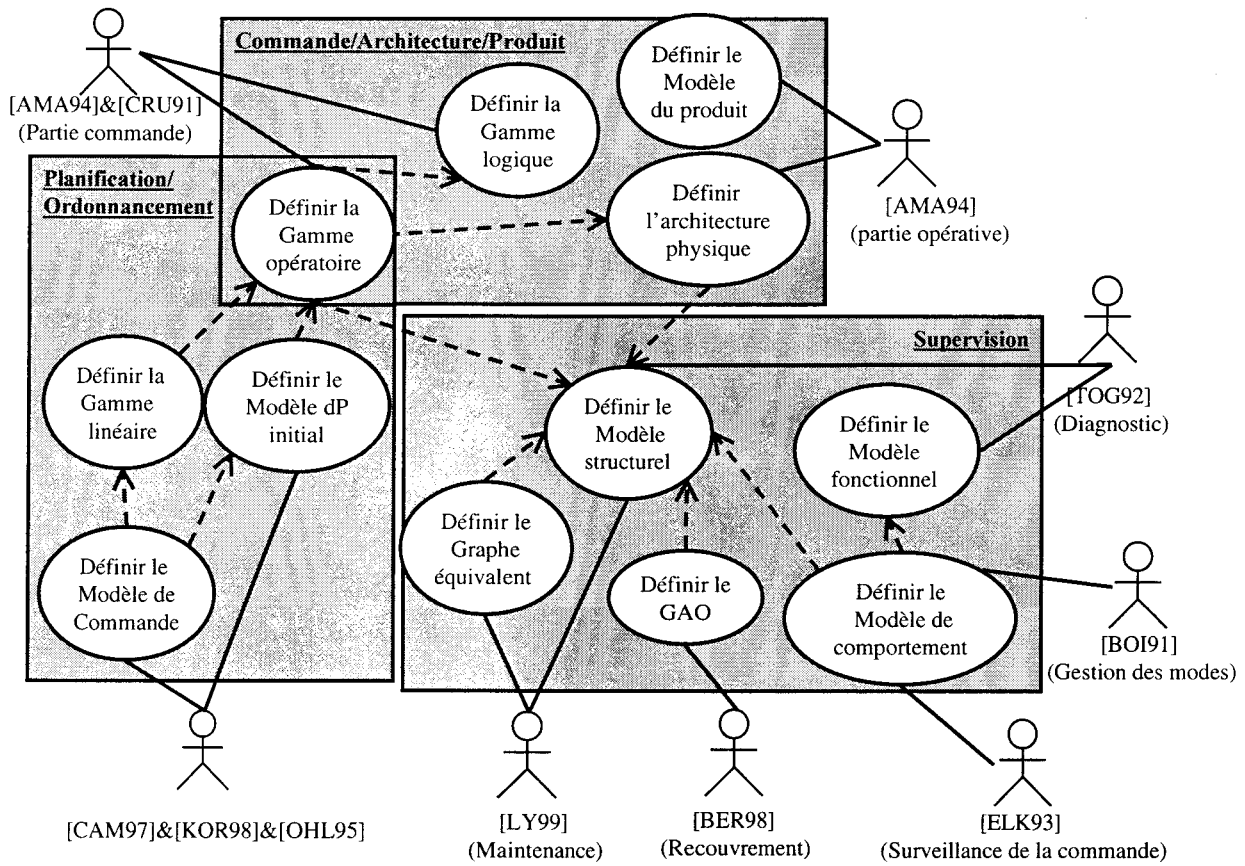



**Figure 5.6-2 : Diagramme des cas d'utilisation dans CASPAIM**

Le diagramme de classes final du métamodèle intégré pour le projet CASPAIM, contient plus de cinquante classes regroupées dans trois paquetages. Sa représentation est possible avec l'aide d'outils comme Rational Rose[7] ou Objecteering[8].

[7] http://www.rational.com

## 5.7. Réalisation pratique : CASPAIM Soft

Pour valider sur le plan pratique l'ensemble des métamodèles que nous avons proposés pour le projet CASPAIM, nous avons développé le logiciel[9] CASPAIM Soft. Nous allons présenter les caractéristiques essentielles du logiciel et puis nous donnerons plus de détails (schémas de tables, scripts, etc.) en annexe.

L'implantation concerne deux points de vue : les parties commande/procédé et la planification/ordonnancement. L'implantation du référentiel a été organisée autour du système de gestion de base de données relationnelle ORACLE. Le choix de ce SGBD a été dicté par les critères suivants :

- Persistance des données,
- Pérennité dans les développements,
- Ouverture vers de nombreux outils tiers,
- Disponibilité d'une plate-forme et de compétences au sein du laboratoire.

La démarche d'implantation a consisté dans un premier temps à traduire l'ensemble des diagrammes de classes en tables et l'ensemble des contraintes OCL en contraintes d'intégrité. Nous avons également réalisé un ensemble de formulaires et d'états permettant d'effectuer la saisie des éléments de modélisation et l'édition de rapports sur le contenu du référentiel. Ces formulaires et états ont été développés avec le produit ORACLE Developer 2000.

Les états concernant les modèles de type réseaux de Petri étant sous forme textuelle, nous avons réalisé en Ada des utilitaires de transformation du contenu des tables dans lesquelles sont stockés les RdP en fichiers lisibles par deux outils de simulation de RdP : Petri maker[10] et RENEW[11]. Le schéma de principe est donné ci-dessous sur la figure 5.7-1.

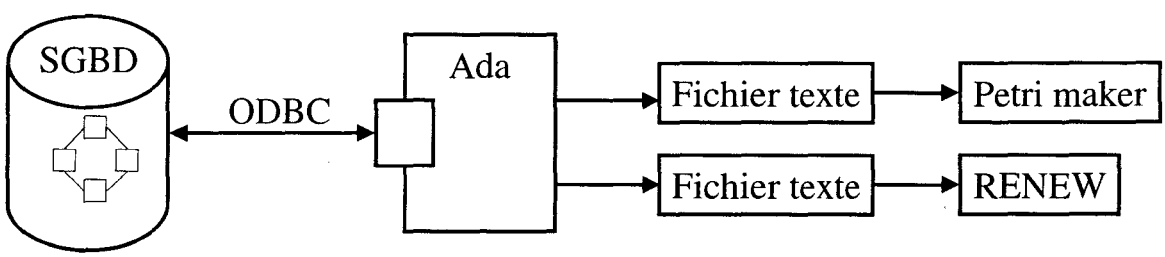



**Figure 5.7-1 : Sortie des données stockées sous forme de RdP**

---

[8] http://www.objecteering.com

[9] Les Sous-Lieutenants Jean Augier, Quentin Fayollat puis Emmanuel Duée de l'Ecole Spéciale Militaire de Saint-Cyr ont également contribué au développement de ce logiciel

[10] http://www.istia.univ-angers.fr/~pmaker/

[11] http://www.renew.de

L'implantation des patterns GOB se traduisant par une structure de tables standard et réutilisable dans la base de données, l'adaptation du programme d'interface pour d'autres outils RdP ne nécessite que très peu de temps à condition, bien sûr que le format d'entrée de l'outil RdP soit connu.

Après avoir implanté la structure interne du référentiel, nous avons reconçu et codé les algorithmes de construction des modèles. Nous avons notamment réalisé :

- Pour la commande, les passages :
  - relations d'accessibilité directe→relations d'accessibilité indirecte,
  - Gamme Logique→Gamme Opératoire,
- Pour l'évaluation de performances, les passages :
  - Gamme Logique→Gamme Opératoire,
  - Gamme Opératoire→Gamme Linéaire,
  - Gamme Linéaire→Macro-gamme, et le choix des macro-gammes.

Les données sont structurées de sorte qu'une même gamme logique peut être appliquée à plusieurs systèmes.

La réalisation de ces applications qui n'entrait pas directement dans le cadre des travaux sur le référentiel, a pu être menée à bien grâce à la bonne structuration des modèles. De plus, elle a pu démontrer non seulement la faisabilité du référentiel, mais aussi l'intérêt pour toute l'équipe PFM, d'une telle approche.

A travers ce chapitre nous avons appliqué la totalité de notre démarche pour intégrer les différents modèles préconçus par les différents experts intervenant dans le projet CASPAIM.

Nous constatons un bon niveau de réutilisation des patterns de conception et des patterns métier ; en effet sur la cinquantaine de classes que compte le métamodèle complet du référentiel CASPAIM une trentaine sont directement obtenues par réutilisation de patterns.

Tous les métamodèles ont été soumis et expliqués aux différents experts qui les ont **validés** sur le plan **théorique**, CASPAIM SOFT effectuant la **validation concrète** (le point de vue supervision est en cours de développement).

Les métamodèles que nous avons conçus ne prennent pas en compte la totalité des informations sur le produit, le processus etc. cependant toutes ces informations pourront être intégrées dans le référentiel sans grandes difficultés, du fait caractère évolutif du référentiel dont la maintenance est facilitée par l'utilisation de concepts clairement définis : les patterns et frameworks.

# Conclusion de la partie B

Nous avons développé une démarche d'intégration de modèles variés qui s'appuie sur la métamodélisation et la réutilisation, deux techniques très répandues en génie logiciel, particulièrement dans la communauté objet. A travers la métamodélisation, nous construisons une architecture pour le référentiel qui pourra jouer pleinement son rôle unificateur, en homogénéisant les modèles à intégrer. L'autre avantage de la métamodélisation, c'est l'indépendance préservée des modèles de concepteurs préexistants qui sont décrits par des métamodèles particuliers (figure 3.2-5). Le langage UML présente des intérêts du fait de son évolutivité, de sa standardisation et des avantages qu'il offre pour une approche orientée objet. Par ailleurs le langage OCL associé à UML nous permet de définir des modèles descriptifs avec une bonne cohérence.

Le principal intérêt de notre démarche réside dans le fait qu'elle propose d'intégrer différents modèles préexistants, et qu'à chaque expérience le référentiel est susceptible d'être enrichi par de nouveaux éléments réutilisables (patterns, frameworks). Les différents intervenants dans un projet ne subissent aucune contrainte, contrairement aux approches qui proposent un modèle de référence générique.

La plupart des objets manipulés sont des composites, ils sont caractérisés par une sémantique très riche, les contraintes existentielles et exclusives que nous avons proposées, apportent une meilleure dynamique des liens composants/composites.

Le métamodèle intégré du référentiel est construit autour d'invariants clairement identifiés. Les patterns contribuent ainsi à l'intégration de points de vue dans la mesure où ils interviennent dans une étape importante de notre démarche d'intégration : la phase de métamodélisation des différents concepts de modélisation ; de plus les patterns métiers contribuent à l'étape d'intégration proprement dite. La réutilisation de patterns a pour principal intérêt d'accélérer le processus de métamodélisation et de faciliter l'intégration des métamodèles mais aussi, en réutilisant des modèles testés et validés, on améliore considérablement la fiabilité, l'évolutivité et l'adaptabilité du métamodèle intégré. De même ils apportent un gain de rapidité en phase d'implantation. Les patterns permettent par ailleurs de différencier explicitement l'expression d'un problème de sa solution, ce qui permet une bonne modularité et un degré de maintenabilité élevé. Les patterns permettent ainsi d'exprimer des problèmes et des solutions à un niveau générique et abstrait, ce qui offre un bon degré de variabilité de la solution fournie par conséquent un bon niveau d'adaptation à un problème concret.

# Conclusion générale

## Contribution

Dans le cadre de l'intégration des activités de conception d'un SAP, l'apport de notre travail se situe au niveau de la gestion de l'information grâce au REFERENTIEL. La mise en place de ce système d'information permet d'obtenir une cohérence globale des données manipulées dans les différents points de vue.
Le choix d'une intégration syntaxique des concepts de modélisation permet l'obtention de métamodèles UML indépendants des formalismes de modélisation et du domaine étudié. La stabilité du référentiel se trouve dans l'intégrité et la persistance des données, qui sont garanties d'une part par le choix d'une intégration forte, d'autre part, par une implémentation dans une base de données. Par métamodélisation nous obtenons un langage de modélisation unique, et par intégration nous obtenons un modèle unique qui offre une interprétation globalement cohérente.

Nous avons créé des patterns qui sont réutilisables pour la plupart des modèles graphiques utilisés dans la conception des SAP. Notre démarche est explicite et peut être reconduite sur d'autres applications. Les patterns sont d'excellents moyens de documentation, et grâce à la démarche de réutilisation que nous avons formalisée, le procédé est partiellement automatisé.
Nous avons eu pour principal objectif de fournir une démarche qui reste indépendante à l'égard des choix d'implémentation, tout en évitant d'introduire une marge supplémentaire entre les métamodèles et les modèles utilisateurs. De ce fait les métamodèles et les modèles des intervenants se trouvent au même niveau d'abstraction dans l'architecture du référentiel.

Dans nos travaux, nous avons montré que les méthodes et outils du Génie Logiciel peuvent apporter une contribution efficace dans le domaine de la productique. Grâce aux apports d'une approche orientée objet, nous avons conçu un référentiel évolutif qui fonctionne suivant un processus incrémental, capable de prendre en compte de nouveaux points de vue. Sa maintenance est facilitée parce qu'il est construit autour de concepts modulables, clairement identifiés et validés.

Nous avons aussi sensibilisé les autres membres de l'équipe PFM au génie logiciel, notamment à l'approche orientée objet et aux systèmes d'information. En effet les concepteurs d'un SAP doivent faire face à un volume de données important ; ils disposent de modèles adéquats pour mener à bien leurs études théoriques mais disposent rarement d'outils adaptés à la gestion de leurs données.

## Perspectives

Les travaux que nous menons se trouvent à la croisée de plusieurs disciplines : le génie industriel, le génie logiciel, la modélisation d'entreprise et la gestion de bases de connaissances. Les apports potentiels de cette dernière discipline n'ont pas encore été mis en exergue. Il serait alors intéressant d'exploiter ce domaine, plus précisément les systèmes experts qui nous permettraient d'automatiser certaines tâches grâce notamment aux règles d'inférence.

Dans l'optique d'améliorer l'intégrité de la base de données du référentiel, il serait intéressant de définir des règles qui permettent de générer automatiquement les contraintes OCL sous forme de déclencheurs base de données, procédures et fonctions stockées.

Nous comptons effectuer des applications réelles sur la cellule de l'Ecole Centrale de Lille avec une mise à jour en temps réel de la base de données. Cela se fera suivant la norme CORBA (Common Object Request Broker Architecture) de l' OMG, née dans les années 1990 du besoin de faire communiquer ensemble des applications en environnement hétérogène (plusieurs systèmes et plusieurs langages).

Nous allons poursuivre la recherche d'invariants de modélisation en étudiant d'autres projets appliqués aux SAP. Cela nous permettra d'une part, d'améliorer la qualité des patterns que nous avons créés (par la définition de nouvelles contraintes, extensions et autres mécanismes d'adaptation), d'autre part le référentiel en sera d'autant plus enrichi par retour d'expériences (création de nouvelles variantes de patterns, à l'instar des composants électroniques).

L'objectif final du référentiel est de disposer de suffisamment de métamodèles pour pouvoir prendre en compte la totalité des informations concernant le produit et le processus de conception ; nous devons alors mener une étude comparative des différents modèles existants à cet effet afin de réaliser une synthèse la plus exhaustive possible. Cela nous amènera alors définir des outils et méthodes pour la capitalisation des connaissances.

# Bibliographie

# A


**[AFI94] AFITEP (Association Française des Ingénieurs Techniciens d'Estimation de Planification et de Projets)**
« Le management de projet : Principes et pratique »
AFNOR gestion, 1994

**[AFN95] Normalisation française**
« Représentation des Systèmes de Contrôle et de Commande des Systèmes Automatisés »
AFNOR Z 68-901, 1995

**[AMA94] S. AMAR**
« Systèmes automatisés et flexibles de production manufacturière : méthode de conception du système de coordination par prototypage orienté objet de la partie procédé »
Thèse de Doctorat de l'Université de Lille1, 1994.

**[AMI93] ESPRIT Consortium AMICE**
« Open System Architecture for CIM research »
2nd revised and extended edition, Springer Verlag, 1993

**[AUS94] 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, Université de Lille 1, 1994

**[ATZ93] P. ATZENI, R. TROLONE**
« A metamodel approach for the management of multiple models and the translation of schemes »
Revue Information Systems, vol 18 n°6, pp 349-362, 1993


# B


**[BAR95] P. BARACOS**
« le Génie Automatique en Amérique du Nord »
Deuxième conférence internationale sur l'automatisation industrielle, pp. LIII-LVIII, Nancy, juin 1995

**[BAU91] H. BAUMGARTNER, K. KNISCHEWSKI, H. WIEDING**
« CIM: Proposition pour une mise en œuvre de la Productique »
Monographie SIEMENS, TEKNEA, 1991

**[BER94] P. BERNUS, L. NEMES.**
« A framework to define a Generic Enterprise Reference Architecture and Methodology »
ICARCV'94 (3rd International Conference on Automation, Robotics and Computer Vision), pp. 88-92, Singapour, novembre 1994.

**[BER98] P. BERRUET**
« Contribution au recouvrement des systèmes flexibles de production manufacturière : analyse de la tolérance et reconfiguration »
Thèse de Doctorat de l'Université de Lille 1, 1998.

**[BEZ95] J. BEZIVIN**
« Objets et Ingénierie des Besoins »
Revue l'OBJET : logiciel, bases de données, réseaux. Vol 1 n°1 1995

**[BEZ98] J. BEZIVIN**
« On different Interoperability Modes in Software Engineering : the case of Modeling Activities at OMG »
Software Engineering'98, Paris 1998

**[BIG96] M.BIGAND, J.P. BOUREY, D. CORBEEL, J.P. MAIK**
« A generalized object approach for the design of flexible manufacturing systems »
CESA'96 IMACS Multiconference, pp352-357, Lille, 1996.

**[BIG97] M. BIGAND, D. CORBEEL, J.P. BOUREY**
« The design of Flexible Manufacturing Systems by using a knowledge based system »
15th IMACS World Congress, pp541-546, Berlin, 1997.

**[BIG98] M. BIGAND, D. CORBEEL, D. NDIAYE, J.P. BOUREY**
« Extensions of object formalism for representing the dynamics: application to the integration of viewpoints in the design of a production system »
Actes de 3rd International Conference on Integrated Design and Manufacturing in Mechanical Engineering IDMME'98, pp1169-1178, Compiègne, 1998.

**[BIG99] M. BIGAND, D. NDIAYE, D. CORBEEL, J.P. BOUREY**
« Contribution to viewpoint modelling using an object approach in an information system for production engineering »
Actes de 2nd World Manufacturing Congress ISMS'99, pp 246-251, Durham, septembre 1999

**[BIG99a] M. BIGAND, D. NDIAYE, D. CORBEEL, J.P. BOUREY**
« Un référentiel pour les systèmes de production manufacturière »
Revue Internationale d'Ingénierie des systèmes de production Mécanique, N°3 pp V.3-V.10, décembre 1999.

**[BOC98] J. C. BOCQUET**
« Conception de produits mécaniques », chapitre 1 « Ingénierie simultanée, conception intégrée »
Editions HERMES, 1998

**[BOI91] S. BOIS**
« Intégration de la gestion des modes de marche dans le pilotage d'un système automatisé de production »
Thèse de Doctorat de l'Université de Lille 1, 1991.

**[BOI98] L. BOITARD**
« Contribution à l'Intégration d'Outils de Génie Automatique autour d'un Système d'Information Unifié. Prototypage d'un Atelier Intégré de Génie Automatique »
Thèse de Doctorat de l'Université de Nancy1, 1998.

**[BON85] R. BONETTO**
« Les ateliers flexibles de production »
Editions HERMES, 1985

**[BON98] E. BON-BIEREL**
« Contribution à l'intégration des modèles de Systèmes de Production Manufacturière par Méta-Modélisation »
Thèse de Doctorat de l'Université de Nancy1, 1998.

**[BOO91] G. BOOCH,**
« Object Oriented Design with Applications »
The Benjamin/Cimmings publishing Company Inc, 1991.

**[BOU88] J. P. BOUREY**
« Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière »
Thèse de Doctorat, Université de Lille1, 1988

**[BOU93] J. P. BOUREY**
« Méthode de conception de la commande des systèmes flexibles de production manufacturière »
Habilitation à diriger des recherches de l'université de Lille 1, 1993

**[BOU94] P. BOURDICHON**
« L'ingénierie simultanée et la gestion des informations »
Editions HERMES, 1994

**[BOU95] M. BOUAZZA**
« la norme STEP »
Editions HERMES, 1995

**[BOU96] F. BOUNAAS, D. RIEU, P. MORAT, D. TAMZALIT**
« MENINGE : un environnement de MEta-modélisatioN pour les projets d'INGEnierie : le Modèle »
Rapport de recherche RT958-I, LSR-IMAG, juillet 1996.

**[BRA95] C. BRAESCH, A. HAURAT**
« la modélisation systémique en entreprise »
Editions HERMES, 1995

**[BRJ00] G. BOOCH, J. RUMBAUGH, I. JACOBSON**
« Le guide de l'utilisateur UML »
Editions Eyrolles, 2000

**[BRO85] J. BROWNE, K.E. STECKE**
« Variations in flexible manufacturing systems according to the relevant types of automated material handling »
Material Flow 2, pp. 179-185, 1985

**[BRO94] J. BROWNE, J. HARHEN, J. SHIVNAN**
« les systèmes de production dans un environnement CIM »
AFNOR, 1994

**[BRO95] J. BROWNE, D. O'SULLIVAN**
« Re-engineering the enterprise »
Chapman &Hall, London, 1995

**[BUB94] J. A. BUBENKO**
« Enterprise Modelling »
Revue Ingénierie des Systèmes d'Information, AFCET/HERMES, vol. 2 n°6 pp. 657-677, 1994

**[BUS96] F. BUSCHMANN, R. MEUNIER, H. ROHNERT, P. SOMMERLAD, M. STAL**
« Pattern Oriented Software Architecture : A System of Patterns »
John Wiley & Son Ltd, 1996

# C

**[CAM97] H. CAMUS**
« Conduite de systèmes flexibles de production manufacturière par composition de régimes permanents cycliques : modélisation et évaluation de performances à l'aide des réseaux de Petri »
Thèse de doctorat, Université de Lille 1, 1997.

**[CAS87] E. CASTELAIN**
« Modélisation et simulation interactive des cellules flexibles dans l'industrie manufacturière »
Thèse de Doctorat, Université de Lille1, 1987

**[CCG91] Centre Coopératif de Génie Automatique**
« Carte des activités de Développement d'un SAP »
ISMCM Saint-Ouen, 1991

**[CHA90] B. CHANDRASEKARAN**
« Design problem solving : a task analysis »
AI Magazine, Winter 1990

**[COA96] P. COAD, D. NORTH, M. MAYFIELD**
« Objects Models: Strategies, Patterns & Applications »
Yourdon Press Computing Series, 1996

[COP95] J. O. COPLIEN, D. C. SCHMIDT
« Pattern Language of Program Design »
Addison Wesley, 1995

[COR87] D. CORBEEL, J. C. GENTINA
« Coloured adaptative structured Petri Nets : a tool for the automatic synthesis of hierarchical control of FMS »
Congrés IEEE Robotics and Automation, Proc. pp 1166-1173, Raleigh-USA avril 1987

[COU93] J.-C. COURBON
« Systèmes d'information : structuration, modélisation et communication »
InterEditions, Paris, 1993

[COU96] B. COULANGE
« Réutilisation du logiciel »
Editions Masson, 1996

[COU97] F. COUFFIN
« Modèle de données de référence et processus de spécialisation pour l'intégration des activités de conception en Génie Automatique »
Thèse de Doctorat de l'E.N.S de Cachan, 1997.

[CRA89] E. CRAYE
« De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles en industrie manufacturière »
Thèse de Doctorat, Université de Lille1, 1989

[CRU91] D. CRUETTE
« méthodologie de conception des systèmes complexes à événements discrets : application à la conception et à la validation hiérarchisée de la commande de cellules de production flexibles dans l'industrie manufacturière ».
Thèse de doctorat, Université de Lille 1, 1991.

# D

[DAM91] Y.G. DAMODAR, CAROL L.S.
« The just in time philosophy : A literature review. »
International journal of production research, 29(4), 657-76, 1991

[DANOO] N. DANGOUMAU, S. EL KHATTABI, A.K.A. TOGUYENI, E. CRAYE
« Modes Management for automated production systems »
WAC 2000, Maui Hawaii, 11-16 juin 2000

[DAV92] R. DAVID, H. ALLA
« Du Grafcet aux réseaux de Petri »
Edition HERMES, 1992

[DEL97] M. DELAVAL
« Séquencement des lignes d'assemblage à modèles mélangés »
Thèse de Doctorat, Université de Lille1, 1997

[DOU90] G. DOUMEINGTS
« Méthodes pour concevoir et spécifier les systèmes de production »
actes du colloque international CIM'90, Bordeaux, pp 89-103, juin 1990

[DOU92] G. DOUMEINGTS, B. VALLESPIR, M. ZANETTIN et al.
« GIM, GRAI integrated methodology fir designing CIM systems »
Rapport LAP/GRAI, Université de Bordeaux I, 1992

# E


**[ELK93] S. EL KHATTABI**
« Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets »
Thèse de Doctorat, Université de Lille1, 1989

**[ERI98] H.-E. ERIKSSON, M. PENKER**
« UML Toolkit »
Wiley Computer Publishing, 1998


# F


**[FAR93] A. FARAH**
« Une méthodologie d'implantation répartie du système de commande d'un SAP »
rapport interne du LAIL, Ecole Centrale de Lille, 1993.

**[FOU94] C. FOULARD**
« la modélisation en entreprise CIM-OSA et ingénierie simultanée »
Edition HERMES, 1994

**[FOW96] M. FOWLER**
« Analysis Patterns: Reusable objects models »
Addison Wesley, 1996

**[FRA87] J. P. FRACHET**
« Une introduction au Génie Automatique : faisabilité d'une chaîne intégrée d'outils CAO pour la conception et l'exploitation des machines automatiques industrielles »
Thèse d'Etat, Université de Nancy 1, 1987.


# G


**[GAM95] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES**
« Design Patterns, Elements of Reusable Object-Oriented Software »
Addison-Wesley, 1995

**[GAR99] G. GARDARIN**
« Bases de Données objets et relationnel »
Editions Eyrolles, 1999

**[GAS89] B. GASNIER**
« Sur un outil interactif de spécification de gammes opératoires en production manufacturière »
Thèse de Doctorat, Université de Lille1, 1989

**[GOL83] J. GOLDHAR**
« Plan for economy of scope »
Harvard Business review, 61(6) , pp141-148, 1983

**[GON95] M. GONDRAN, M. MINOUX,**
« Graphes et algorithmes »
Editions EYROLLES, 1995

[GUE92] F. GUENA
« Réutilisation de solutions génériques pour résoudre des problèmes de conception »
Thèse en Informatique de l'Institut Blaise Pascal, Paris VI, 1992

# H

[HAN96] R. HANNAM
« Computer Integrated Manufacturing : from concepts to realisation »
Addison Wesley, 1996

[HAR97] Y. HARANI
« Une approche Multi-Modèles pour la Capitalisation des Connaissances dans le domaine de la Conception »
Thèse de Doctorat de l'Institut National Polytechnique de Grenoble, 1997.

[HAU97] J.P. HAUET
« Gateway to the Internet »,
WorldFip FieldBus, n° 16, décembre 1997

[HEB92] A. HEBRARD
« Etude et réalisation d'un module de pilotage et d'un outil de représentation graphique appliqués à la conception d'un système de production flexible »
Thèse de Doctorat, Université de Lille1, 1992

[HUV94] B. HUVENOIT
« Conception à l'implantation de la commande modulaire et hiérarchisée de systèmes flexibles de production manufacturière »
Thèse de Doctorat, Université de Lille1, 1994

[HSU93] C. HSU, Y-C TAO, M BOUZIANE ? G. BABIN
« Paradigm Translations in Integrating Manufacturing Information using a Meta-model: the TSER approach »
Revue Ingénierie des Systèmes d'Information, vol1, n°3 pp. 325-352, 1993

# I

[IDE94] Z. IDELMERFAA.
« Méthodologie de génération de la conduite d'un système intégré de production »
Thèse de doctorat en Production automatisée, Université de Nancy 1, 1994

# J

[JAC92] I. JACOBSON, M. CHRISTERSON, P. JONSSON,G. OVERGAARD
« Object Oriented Software Engineering »
Addison-Wesley, ACM Press, 1992

[JAG93] P. JAGOU
« Concurrent Engineering, la maîtrise des coûts, des délais et de la qualité »
Edition HERMES, 1993

**[KAP88] M. KAPUSTA**
« Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles »
Thèse de Doctorat, Université de Lille1, 1988

**[KRU00] P. KRUCHTEN**
« Introduction au Rational Unified Process »
Editions Eyrolles, 2000

**[KAR88] A. KARFIA**
« Présentation d'une base de connaissances adaptée à la modélisation par réseaux de Petri structurés du contrôle des processus de production discrétisés »
Thèse de Doctorat, Université de Lille1, 1988

**[KAT90] R.H. KATZ**
« Toward a Unified Framework for version Modeling in Engineering databases »
ACM Computing Surveys, vol 22, N° 4, pp 375-408, Décembre 1990

**[KET98] N. KETTANI, D. MIGNET, P. PARE, C. ROSENTHAL-SABROUX**
« De Merise à UML »
Editions Eyrolles, 1998

**[KIM92] F. KIMURA, T. KJELLBERG, F-L. KRAUSE, S.C.-Y LU, M. WOSNY**
« Interim report, Annals of the CIRP » 41/2, 743-48, The first CIRP international workshop on Concurrent Engineering for product realization, Tokyo, janvier 1992

**[KIE96] F. KIEFER**
« Contribution à l'ingénierie intégrée des systèmes de production : formalisation des mécanismes d'intégration entre modèles et applications sur site industriel »
Thèse de doctorat de l'Ecole Normale Supérieure de Cachan, janvier 1996

**[KER96] L. KERMAD**
« Contribution à la supervision et à la gestion des modes et des configurations des systèmes flexibles de production manufacturière »
Thèse de Doctorat, Université de Lille1, 1996

**[KOE89] A- KOESTLER**
«The GHOST in the MACHINE »
Editions ARKANA, Londres, 1989

**[KOE90] J.B. KOESCHLIN**
« Logiciel ELEGA : Première réalisation mettant en œuvre les travaux BASE-PTA »
Acte CIM'90 Productique et Intégration, pp. 333-343, Bordeaux Juin 1990

**[KOR98] O. KORBAA**
« Commande Cyclique des systèmes flexibles de production manufacturière à l'aide de réseaux de Petri : de la planification à l'ordonnancement des régimes transitoires »
Thèse de Doctorat, Université de Lille1, 1998

**[KUR96] T. KURIHARA, P. BUNCE, J. JORDAN**
«Next Generation Manufacturing Systems (NGMS) in IMS Program »
Actes de la 2ème Conférence Internationale DIISM (The Design of Information Infrastructure Systems for Manufacturing), Kaatsheuvel, 15-18 septembre 1996

**[LAR97] O. LARAB, A.N. BENHARKAT**
« Raffinement des correspondances dans l'intégration des schémas de bases de données »
Revue Ingénierie des Systèmes d'Information, vol. 5, n°3, pp. 307-337, 1997

**[LEM90] J.-L. LEMOIGNE**
« La modélisation des systèmes complexes »
AFCET systèmes, éditions DUNOD, 1990

**[LES94] J.-J. LESAGE**
« Contribution à la formalisation des modèles et méthodes de conception des systèmes de production »
Habilitation à diriger des recherches, université de Nancy1, 1994

**[LES89] J.-J. LESAGE**
« Conception de la commande des systèmes de production : Contribution à la Structuration, Application à la conception de la commande d'un Atelier Flexible »
Thèse de Doctorat Automatique Productique, LURPA Paris, 1989

**[LHO99] P. LHOSTE, P. PUJO**
« Rapport de synthèse du Groupe de Travail n°2 »
GT GRP-COMPIL, Mars 1999

**[LOM94] M. LOMBARD-GREGORI**
« Contribution au Génie Productique : prototypage d'une architecture d'ingénierie concourante des systèmes intégrés de fabrication manufacturière »
Thèse de Doctorat, Université de Nancy 1, 1994

**[LY99] F. LY**
« Contribution par la surveillance prédictive indirecte à l'optimisation de la maintenance dans les systèmes flexibles de production manufacturière »
Thèse de Doctorat, Université de Lille1, 1999



**[MAR91] D. MARTIN**
« La norme de référentiel IRDS : objectif et fonctions »
Revue Génie Logiciel & Systèmes Experts, N°22 pp 16-23, mars 1991

**[MAR96] P. MARET, L. POULLET, J.M. PINON**
« Des modèles conceptuels pour capitaliser la connaissance au sein d'une organisation »
Revue Ingénierie des Systèmes d'Information, Volume 4, n°4, pp. 491-540, 1996

**[MIL98] O. MILLION**
« De l'intégration des métiers par les données techniques vers la maîtrise de la modélisation conceptuelle : la méthode VIM (Viewpoint Information Modelling) »
Thèse de Doctorat, Université de Nancy 1, 1998

**[MIN85] M. MINSKY**
« The society of mind »
Simon & Schuster Ed, 1985

**[MIN91] R. MINOT, D. SOUFFLET**
« PCTE, Une plate-forme d'intégration normalisée »
Revue Génie Logiciel & Systèmes Experts, n°22, pp. 10-14, mars 1991

**[MOR92] G. MOREL**
« Contribution à l'automatisation et à l'ingénierie des systèmes intégrés de production »
Habilitation à diriger des recherches, Université de Nancy 1

**[MOR95] B. MORAND**
« Statut épistémologique des modèles dans la conception des systèmes 'information »
Revue Ingénierie des Systèmes d'Information, vol 3, n°5, pp. 665-700, 1995

**[MOR95a] R. MOREAU**
« L'approche objet : Concepts et Techniques »
Masson, 1995.

**[MUL97] P.A. MULLER**
« Modélisation objet avec UML »
Editions Eyrolles, Paris 1997

# N

**[NDI99] D. NDIAYE, M. BIGAND, D. CORBEEL, J.P. BOUREY**
« Information system for production engineering : contribution to maintaining consistency of composite data using object approach »
Actes de International Conference on Industrial Engineering and Production Management IEPM99, Book 2 pp 191-199, Glasgow juillet 1999

**[NDI00a] D. NDIAYE, M. BIGAND, D. CORBEEL, J.P. BOUREY**
« Démarche ascendante pour la conception des systèmes de production »
Actes de 3rd International Conference on Integrated Design and Manufacturing in Mechanical Engineering IDMME2000, Montréal mai 2000

**[NDI00b] D. NDIAYE, M. BIGAND, D. CORBEEL, J.P. BOUREY**
« Utilisation de patterns pour le modèle de commande des systèmes de production »
Actes de la Conférence Internationale francophone d'Automatique CIFA'2000 pp. 133-138, Lille juillet 2000

**[NDI00c] D. NDIAYE, M. BIGAND, D. CORBEEL, J.P. BOUREY**
« Reuse and patterns for viewpoints integration of Automated Production Systems »
Actes du 7th ISPE International Conference On Concurrent Engineering CE2000, Lyon, juillet 2000

**[NGU93] G.T NGUYEN**
« SHOOD, Plate-forme pour la conception assistée »
Revue Ingénierie des systèmes d'information. Vol. 1 - n°3 pp. 393-414, 1993

# O

**[OHL95] H. OHL**
« Fonctionnements réactifs des systèmes flexibles de production manufacturière : analyse et optimisation des performances à l'aide des réseaux de Petri »
Thèse de Doctorat, Université de Lille1, 1995

**[OUS97] C. OUSSALAH et al**
« Ingénierie Objet, Concepts et techniques »
InterEditions, 1997

# P

**[PIC97] M. PICAVET**
« La complexité dans la modélisation du système d'information de l'entreprise. Propositions de solutions : concepts, outils et démarches »
Thèse d'habilitation à diriger des recherches de l'Université de Lille 1, 1997

**[PRI94] J. PRINZT**
« Aspects économiques du génie logiciel »
AFCET, Groupe de travail Génie Logiciel, Etat actuel et perspectives pp. 33-36, journée d'étude 21 octobre 94

# R

**[RAN86] P.RANKY**
« Computer Integrated manufacturing »
Editions Prentice-Hall, Englewood Cliffs, 1986

**[RAN90] P. RANKY**
« Flexible Manufacturing Cells and Systems in CIM »
CIMware Ltd. , 1990

**[RIE90] D. RIEU, V. FAVIER, P. JEAN, B. DAVID, G.T. NGUTEN**
« SHERPA :un support d'intégration pour le processus CEDF »
CIM90 Productique et Intégration, P. 451-458, Bordeaux, 12-14 juin 1990

**[ROB88] M. ROBOAM**
« Modèles de référence et intégration des méthodes d'analyse pour la conception des systèmes de production »
Thèse de doctorat, Université de Bordeaux 1, 1988

**[ROD89] G. RODDE**
« les systèmes de production, modélisation et performances »
Edition HERMES, 1989

**[ROQ00] P. ROQUES, F. VALLEE**
« UML en action »
Editions Eyrolles, 2000

**[ROL86] C. ROLLAND**
« Introduction à la conception des systèmes d'information et panorama des méthodes disponibles »
Revue Génie logiciel n° 4, éditions EC2, avril 1986

**[RUM91] J. RUMBAUGH, M. BLAHA, W. PREMERLANI, F. EDDY, W. LORENSEN**
« Object oriented Modelling and design »
Prentice-Hall, 1991.

# S

**[SIM82] J.A. SIMPSON, R.J. HOCKEN, J.S. ALBUS**
«The Automated Manufacturing Research Facility of the National Bureau of Standards »
Journal of Manufacturing Systems, Vol. 1, n°1, 1982

[SCH94] W.-A. SCHEER, C. KRUSE
« ARIS Framework and Toolset : A comprehensive business process re-engineering methodology »
ICARCV'94 (3rd International Conference on Automation, Robotics and Computer Vision) , pp. 327-331, Singapour, novembre 1994.

## T

[TAR83] H. TARDIEU, A. ROCHFELD, R. COLLETTI
« La méthode Merise Tome 1 : principes et outils »
Editions d'Organisation, 1983

[TAW95] R. TAWEGOUM
« Contrôle temps réel du déroulement des opérations dans les systèmes de production flexibles »
Thèse de Doctorat, Université de Lille1, 1995

[TCH97] N. TCHERNEV
« Réseaux d'écoulement des flux de matières »
Groupe de travail BERMUDES, Bulletin de liaison n° 4, juin 1997, Lille

[TIC95] S. TICHKIEWITCH, E. CHAPA, P. BELLOY
« Un modèle produit multi-vues pour la conception intégrée »
Productivity in world without borders conference, Montréal, 1995

[TOG92] A.K.A. TOGUYENI
« Surveillance et diagnostic en ligne dans les systèmes flexibles de l'industrie manufacturière »
Thèse de doctorat, Université de Lille 1, 1992

[TOL00] M. TOLLENAERE
« Patrons d'ObjetS pour les Echanges Industriels de DONées »
Rapport d'activité 1999, PROSPER projet POSEIDON, mars 2000

## V

[VAL94] P. VALCKENAERS, H. VAN BRUSSEL, F. BONNEVILLE, L.. BONGAERTS, J. WYNS
« IMS Test Case 5 : Holonic Manufacturing Systems »
Preprints of IMS'94, IFAC Workshop, Vienne, 13-15 juin 1994

[VER91] L. VERDIN
« De la spécification à l'exploitation : le cycle de vie des SAP »
Actes du GAMI : Génie Automatique et Production Industrielle, tome 2,
Saint-Ouen, France, mars 1991

[VER89] F. VERNADAT, A. DI LEVA, P. GIOLITO
« Organization and information system design of manufacturing environments : the new M* approach »
Computer-Integrated manufacturing Systems, vol 2, n°2, pp. 69-81, 1989

[VER96] F. VERNADAT
« Enterprise Modeling and Integration : principles and applications »
CHAPMAN & HALL, 1996

[VER99] F. VERNADAT
« Techniques de Modélisation en Entreprise : Applications aux Processus Opérationnels »
Edition ECONOMICA, 1999

**[WAL92] J.-B.WALDNER**
« CIM : principles of Computer Integrated Manufacturing »
Editions John Wiley & Sons, New York, NY, 1992

**[WES91] R. E. WESTNEY**
« Gestion de petits projets : Techniques de planification, d'estimation et de contrôle »
AFNOR gestion, 1991

**[WIL92] T.J. WILLIAMS**
« The Purdue Enterprise Reference Architecture »
Instrument Society of America, Research Triangle Park, North Carolina, USA, 1992

**[WYN99] J. WYNS**
« Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration »
Thèse de Doctorat, Université Catholique de Louvain Belgique, 1999

# Liste des figures

## CHAPITRE 1



## CHAPITRE 2

## CHAPITRE 3



## CHAPITRE 4

## CHAPITRE 5

# Liste des abréviations et acronymes

| | |
|---|---|
| AGL | Atelier de Génie Logiciel |
| AIT | Advanced Information Technology in Design and Manufacturing Integration platform |
| AMRF | Automated Manufacturing Research Facility |
| ARIS | ARchitecture for Integrated Information System |
| CALS | Computer Aided Acquisition and Logistic Suppor |
| CAO | Conception Assistée par Ordinateur |
| CASE | Computer Aided Software Engineering |
| CASPAIM | Conception Assistée des Systèmes de Production Automatisés en Industrie Manufacturière |
| CDIF | Case Data Interchange Format |
| CFAO | Conception et de Fabrication Assistée par Ordinateur |
| CIM | Computer Integrated Manufacturing |
| CIMOSA | CIM Open System Architecture |
| CLR | Composant Logiciel Réutilisable |
| EIA | Electronic Industries Association |
| ELEGA | Environnement Logiciel Expérimental pour le Génie Automatique |
| ESPRIT | European Strategic Program for Research and Development in Information Technology |
| GAO | Graphe d'Accessibilité Opérationnelle |
| GERAM | Generic Enterprise Reference Architecture and Methodology |
| GRAI | Graphe à Résultats et Activités Inter-reliés |
| HMS | Holonic Manufatcuring System |
| ICAM | Integrated Computer Aided Manufacturing |
| IDA | Institut for Defense Analysis |
| IMS | Intelligent Manufacturing System |
| IRDS | Information Resource Dictionary System |
| KADS | Knowledge and Analysis Design Support |
| MENINGE | MEta-modélisatioN pour les projets d'INGEnierie |
| MOF | Meta Object Facility |
| NBS | National Bureau of Standards |
| NIST | National Institute of Standards and Technology |
| NTIC | Nouvelles Technologies de l'Information et de Communication |
| OCL | Object Constraint language |
| OMG | Object Management Group |
| OMT | Object Modeling Technique |
| OOSE | Object Oriented Software Engineering |
| PCTE | Portable Common Tool Environment |
| PERA | Purdue Enterprise Reference Architecture |
| PFM | Production Flexible Manufacturière |
| POSEIDON | Patrons d'ObjetS pour les Echanges Industriels de DONnées |
| PTA | Poste de Travail de l'Automaticien |
| RdP | Réseaux de Petri |
| SADT | Structured Analysis and Design Technique |
| SAP | Systèmes Automatisés de Production |
| SFPM | Système Flexible de Production Manufacturière |
| SGBD | Systèmes de gestion de Bases de Données |
| SI | Système d'Information |
| SSC | Sous-Schéma Conceptuel |
| TSER | Two Stage Entity Relationship |
| UML | Unified Modeling language |
| XMI | XML Model Interchange |
| XML | Extended Markup Language |

# Annexes

## Résumé

**Dans cette partie annexe, nous présentons le logiciel CASPAIM SOFT à travers deux exemples traités pour les points de vue Commande et Evaluation de performances. Cette partie contient des copies d'écran pour illustrer CASPAIM SOFT à travers les différentes étapes de saisies. Par ailleurs nous montrons quelques sorties impression ainsi que les RdP générés dans RENEW à partir des données stockées dans les tables.**

## Sommaire



## Liste des copies d'écran

# A. Données communes

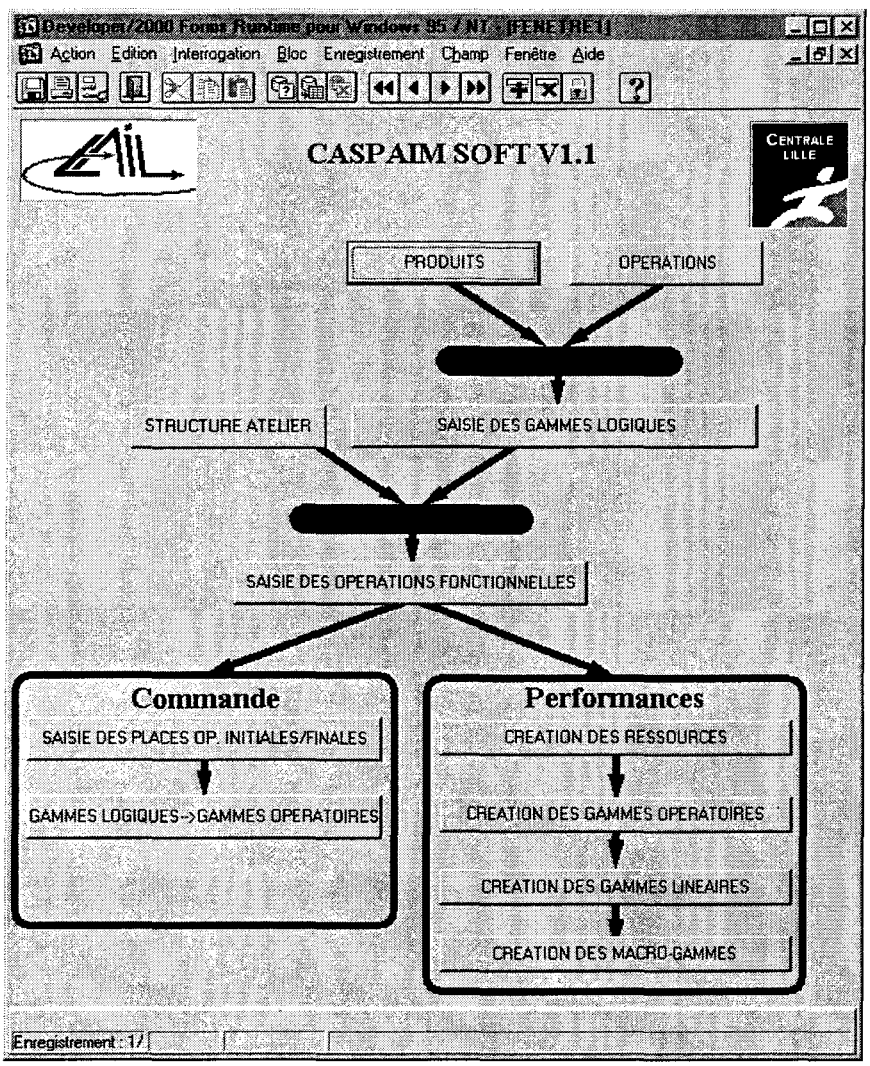
Au démarrage de CASPAIM SOFT, l'utilisateur a la possibilité de saisir les données de base qui pourront être utilisées dans les différents points de vue (ici la Commande et l'évaluation de performances).

Les informations à saisir dans un premier temps concernent les produits à fabriquer ainsi que les opérations pouvant être réalisées par le système de production.

La description du produit ainsi que la définition des opérations réalisables par le système sont nécessaires à la définition des gammes logiques.

Ensuite la structure physique de l'atelier (système de production) est décrite afin de montrer les relations d'accessibilité entre les différents éléments du système étudié.

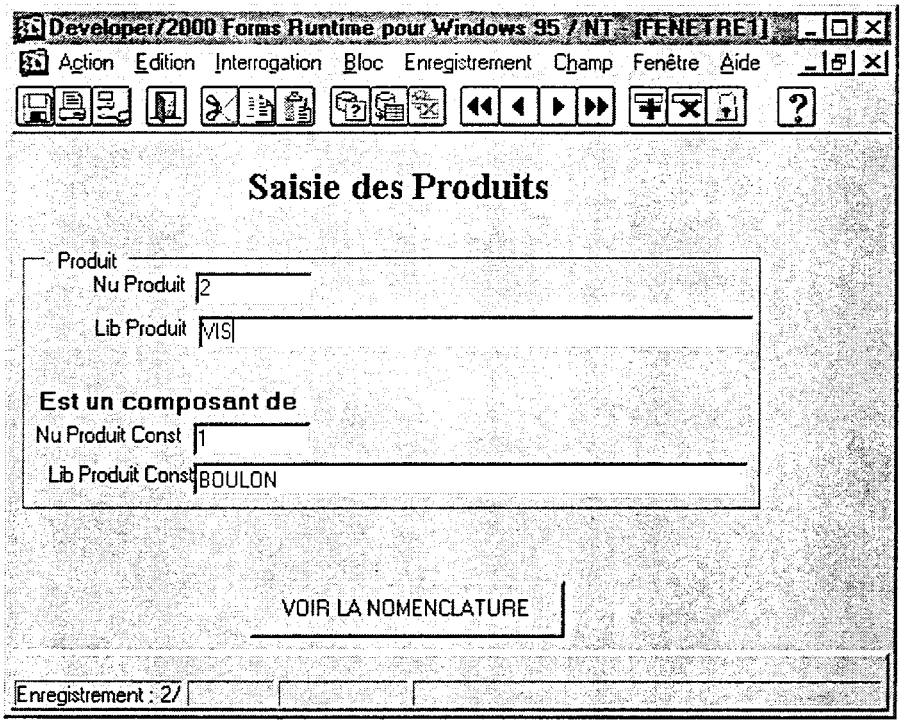
Enfin les opérations fonctionnelles sont définies sur les différentes zones opératoires des ressources.




**Copie d'écran A-1 : page d'accueil**

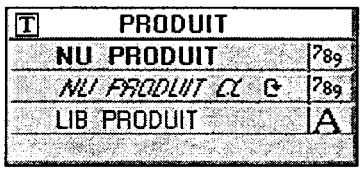
## A.1. Nomenclature des produits

Les données prises en compte pour le produit concernent sa nomenclature. Par exemple le produit boulon composé d'une vis et d'un écrou.




**Copie d'écran A-2 : fenêtre de saisie d'un produit et ses composants**

Ces données saisies par l'utilisateur sont stockées dans la table ci-dessous.




**Copie d'écran A-3 : table produit**

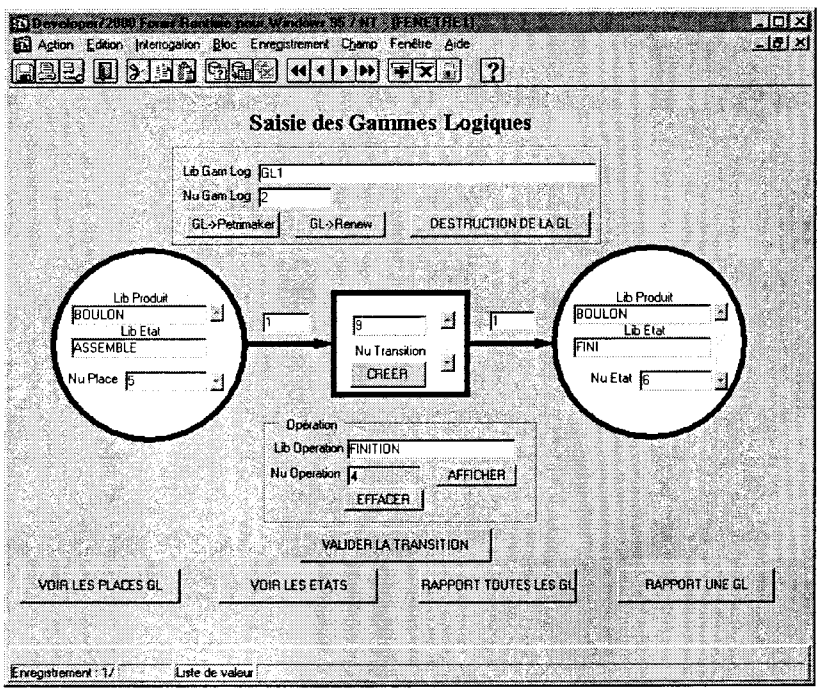
30-MAI-00 14:51:00

# Nomenclature Produit

| Nomenclature (n°:libellé) |
|---|
| 1:BOULON |
| --- 2:VIS |
| --- 3:ECROU |

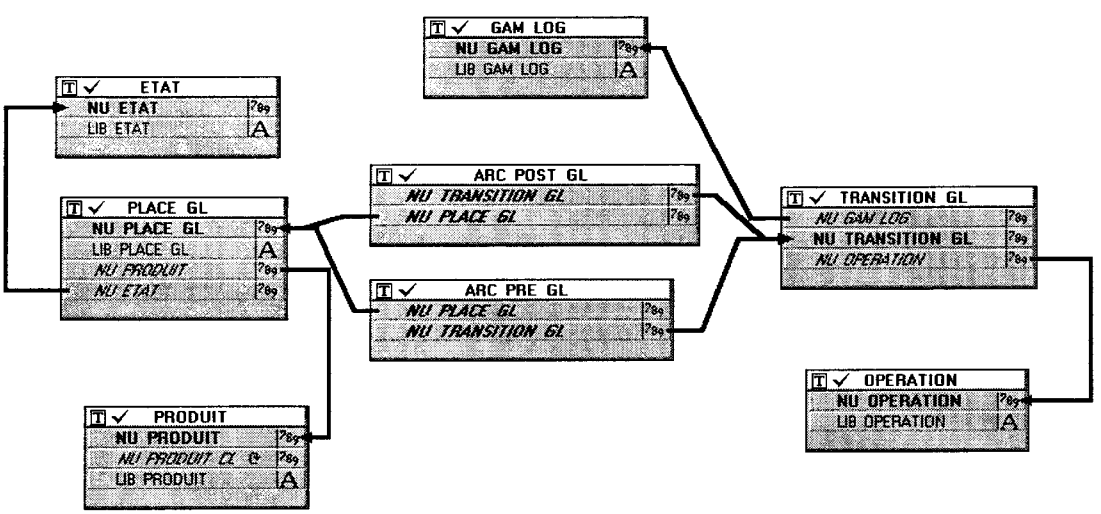
## A.2. Gamme logique

Le produit à fabriquer est décrit à l'aide d'une gamme logique, sans tenir compte de la nature des équipements à utiliser. Une gamme logique correspond ainsi à une description purement fonctionnelle du processus de fabrication, sans aucune référence, d'une part au procédé technique utilisé pour réaliser une opération caractéristique, et d'une autre part, à l'aspect physique ou matériel du système de fabrication.
Chaque produit est défini par son libellé. Il se trouve successivement dans plusieurs états, chaque état étant associé à une place. Chaque transition est associée à une seule et unique opération, cette dernière pouvant être affectée à plusieurs transitions.



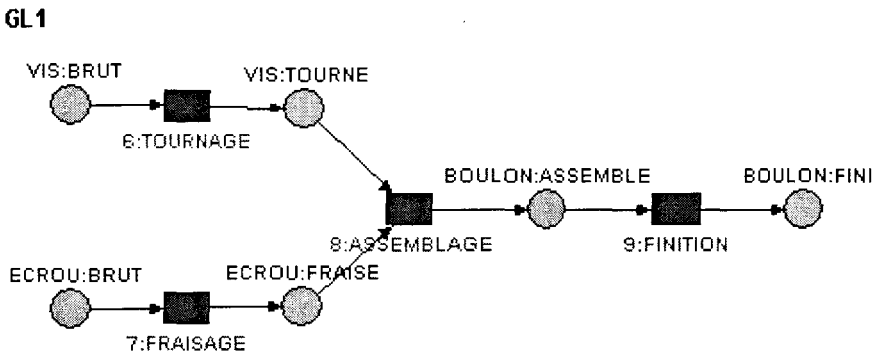

**Copie d'écran A-4 : saisie des gammes logiques**

Les tables ci-dessous stockent toutes les données sur les gammes logiques.




**Copie d'écran A-5 : schéma des tables pour la gamme logique**

A partir des données saisies pour la gamme logique, nous pouvons alors générer dans Petrimaker ou Renew le réseau de Petri correspondant.




**Copie d'écran A-6 : Génération de la gamme logique « boulon » sous RENEW**

30-MAI-00 15:14:0

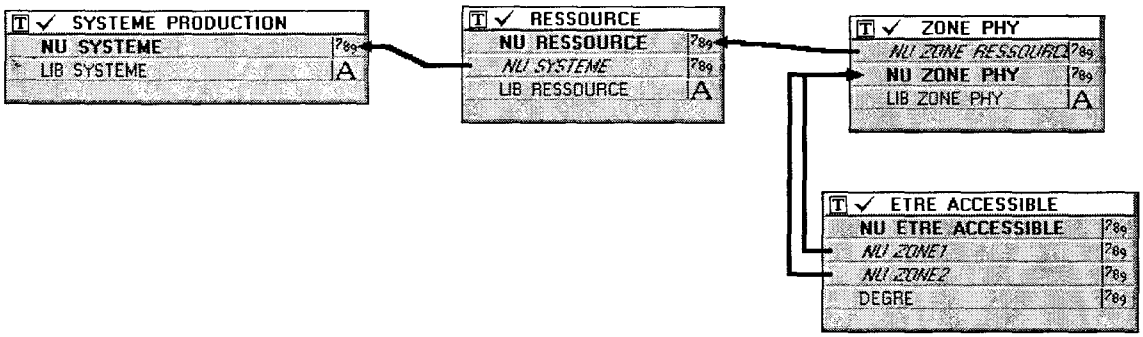
# Gammes Logiques

Lib Gam Log :**GL1**
Nu Gam Log :**2**

| n° | Places Amont | Pds | n°Trans | Lib Operation | Pds | n° | Places Aval |
|---|---|---|---|---|---|---|---|
| 5 | BOULON:ASSEMBLE | 1 | 9 | FINITION | 1 | 6 | BOULON:FINI |
| 3 | ECROU:BRUT | 1 | 7 | FRAISAGE | 1 | 4 | ECROU:FRAISE |
| 4 | ECROU:FRAISE | 1 | 8 | ASSEMBLAGE | 1 | 5 | BOULON:ASSEMBLE |
| 1 | VIS:BRUT | 1 | 6 | TOURNAGE | 1 | 2 | VIS:TOURNE |
| 2 | VIS:TOURNE | 1 | 8 | ASSEMBLAGE | 1 | 5 | BOULON:ASSEMBLE |

## A.3. Structure du système de production

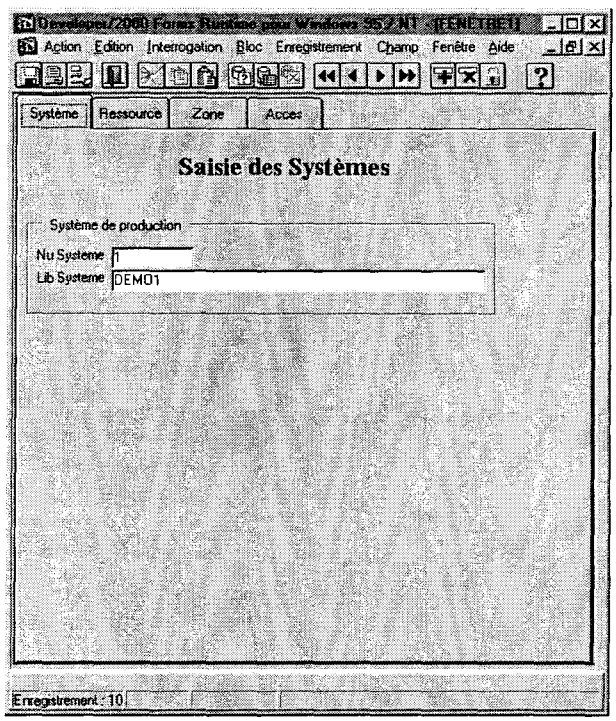
Pour chaque système étudié, nous devons définir l'ensemble de ses ressources, les zones physiques de chaque ressource, ainsi que les relations d'accessibilité entre les ressources.
Ces données sont stockées dans les tables ci-dessous, la saisie s'effectue dans quatre fenêtres (pages d'onglet).




**Copie d'écran A-7 : schéma de tables pour la structure de l'atelier**

### A.3.1. Création des systèmes

Cette fenêtre de saisie permet de créer un nouveau système de production.




**Copie d'écran A-8 : création d'un système**

### A.3.2. Ressources du système

Le système sur la figure A-1 sera utilisé à titre d'exemple. Il est composée des ressources Robot, M1, M2, IN et OUT.

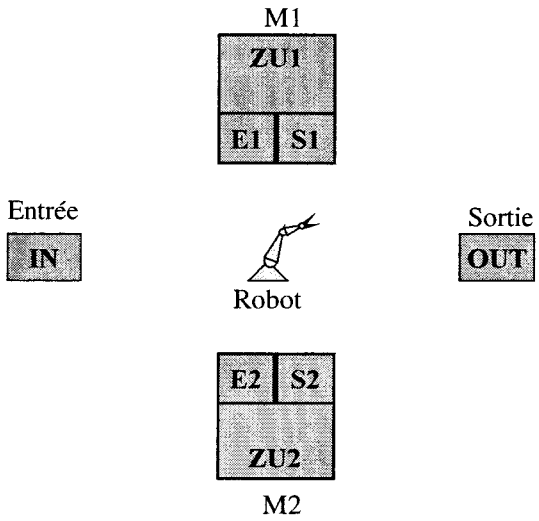
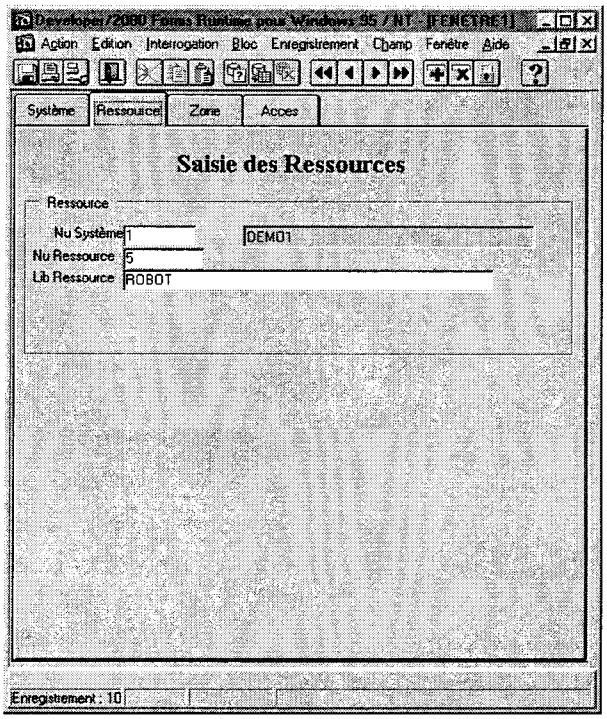



**Figure A-1 : exemple d'atelier de production**

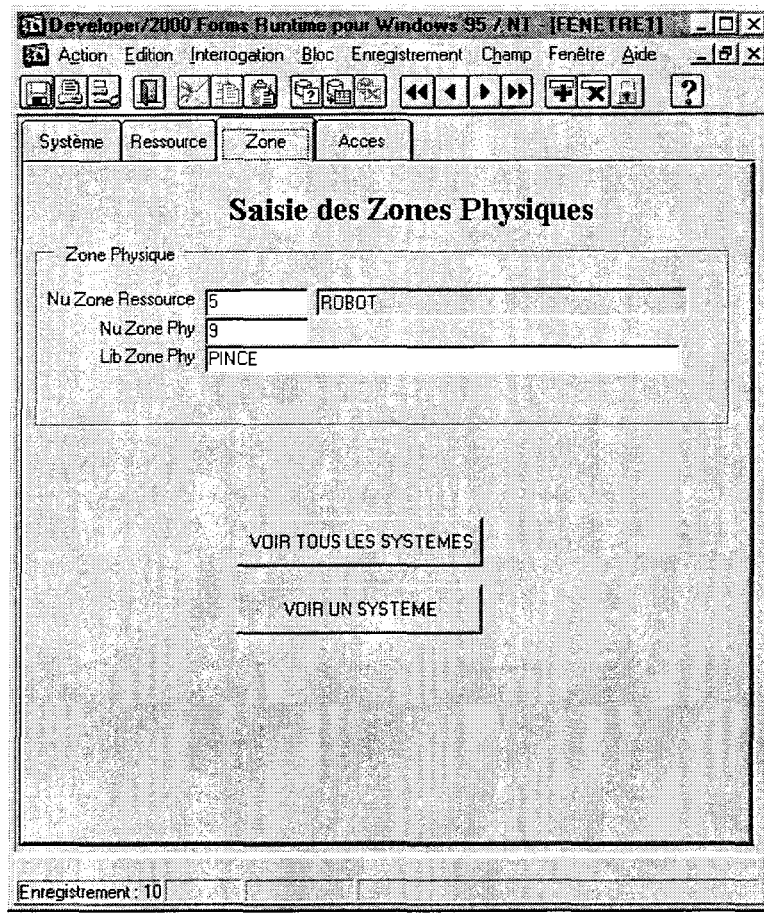
Chaque ressource possède un libellé et un numéro, et appartient à un système identifié par un numéro unique.




**Copie d'écran A-9 : saisie des ressources du système**

### A.3.3. Zones physiques des ressources

Les ressources du système possèdent un à plusieurs zones physiques par lesquelles transitent les produits. Une zone physique appartient à une ressource de production et représente soit une zone opératoire soit une zone d'accès à cette ressource. Par exemple sur la figure A-1 la machine M1 possède une zone d'entrée E1 et une zone de sortie E2.




**Copie d'écran A-10 : saisie des zones physiques**

# Structure des systèmes

Nu Systeme **1**
Lib Systeme : **DEMO1**

Nu Ressource :**1**
Lib Ressource : ***ENTREE***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 1 | IN |

**NB zones phy de la ressource : 1**

Nu Ressource :**2**
Lib Ressource : ***SORTIE***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 2 | OUT |

**NB zones phy de la ressource : 1**

Nu Ressource :**3**
Lib Ressource : ***M1***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 3 | E1 |
| 4 | ZU1 |
| 5 | S1 |

**NB zones phy de la ressource : 3**

Nu Ressource :**4**
Lib Ressource : ***M2***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 6 | E2 |
| 7 | ZU2 |
| 8 | S2 |

**NB zones phy de la ressource : 3**

Nu Ressource :**5**
Lib Ressource : ***ROBOT***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 9 | PINCE |

Nu Systeme **1**
Lib Systeme : **DEMO1**

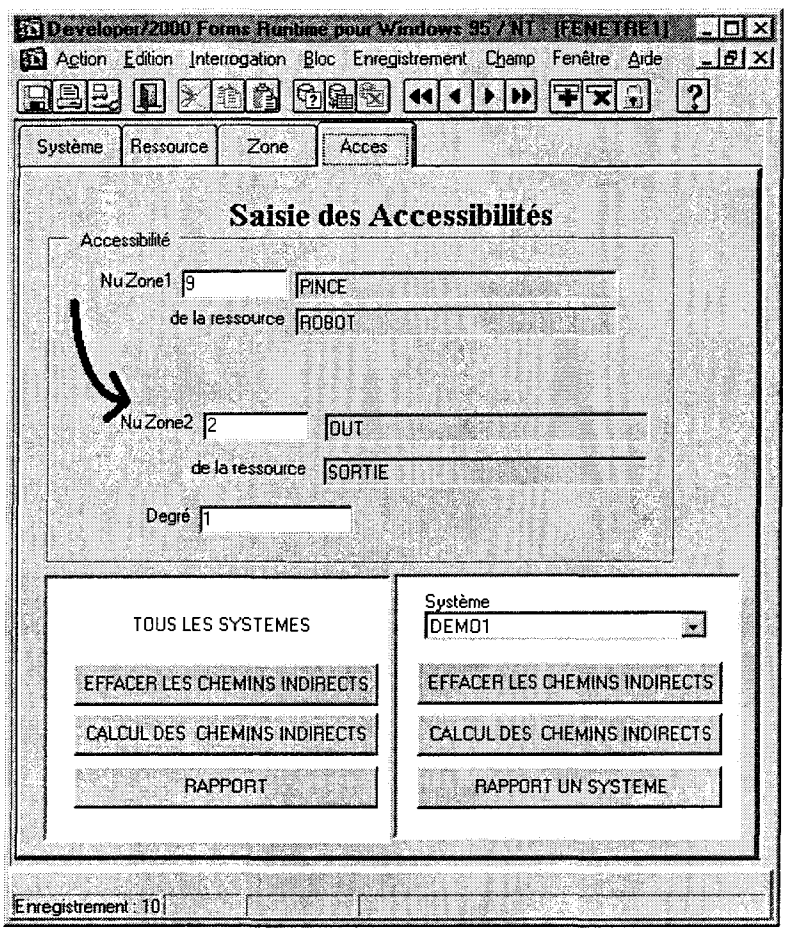
Nu Ressource :**5**
**NB zones phy de la ressource : 1**

**NBZones phy du systèm 9** **NB Ressources du système: 5**

### A.3.4. Définition des relations d'accessibilité

La relation d'accessibilité définit les liens existant entre les différentes ressources de production concernant les échanges de pièces. Si un produit peut transiter d'une ressource à une autre sans intermédiaire, on parle d'accessibilité directe. Les accessibilités indirectes sont déduites par transitivité.

Les relations d'accessibilité directes sont saisies par l'utilisateur, ensuite les accessibilités indirectes sont calculées.




**Copie d'écran A-11 : saisie des accessibilités**

Les rapports sur les relations d'accessibilité sont générés sous forme matricielle. Les lignes de la matrice correspondent aux zones de départ et les colonnes aux zones d'arrivée. La valeur de chaque élément de la matrice exprime le nombre de transitions à faire pour passer de la zone de départ à la zone d'arrivée.

30-MAI-00 15:19:31

# Matrice d'accessibilité

| Lieu Arrivee | M1:E1 | M1:S1 | M1:ZU1 | M2:E2 | M2:S2 | M2:ZU2 | ROBOT:PINCE | SORTIE:OUT |
|---|---|---|---|---|---|---|---|---|
| Lieu Depart | | | | | | | | |
| ENTREE:IN | | | | | | | 1 | |
| M1:E1 | | | 1 | | | | | |
| M1:S1 | | | | | | | 1 | |
| M1:ZU1 | | 1 | | | | | | |
| M2:E2 | | | | | | 1 | | |
| M2:S2 | | | | | | | 1 | |
| M2:ZU2 | | | | | 1 | | | |
| ROBOT:PINCE | 1 | | | 1 | | | | 1 |

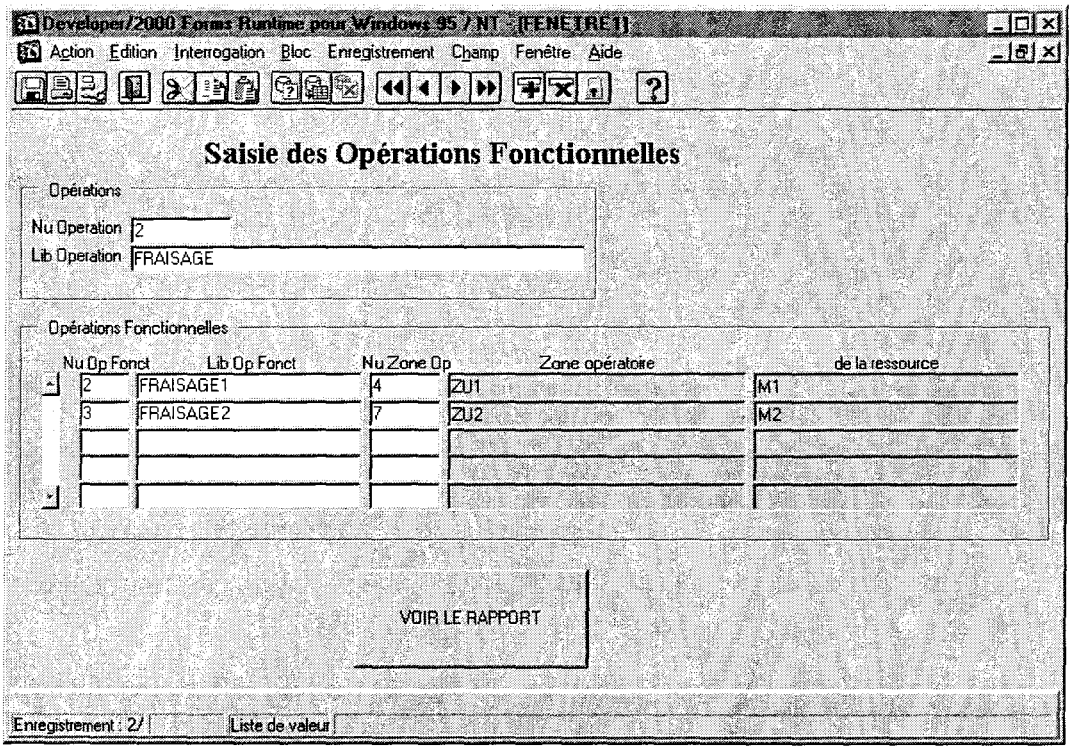
30-MAI-00 15:19:58

# Matrice d'accessibilité

| Lieu Arrivee | M1:E1 | M1:S1 | M1:ZU1 | M2:E2 | M2:S2 | M2:ZU2 | ROBOT:PINCE | SORTIE:OUT |
|---|---|---|---|---|---|---|---|---|
| Lieu Depart | | | | | | | | |
| ENTREE:IN | 2 | 4 | 3 | 2 | 4 | 3 | 1 | 2 |
| M1:E1 | | 2 | 1 | 4 | 6 | 5 | 3 | 4 |
| M1:S1 | 2 | | 3 | 2 | 4 | 3 | 1 | 2 |
| M1:ZU1 | 3 | 1 | | 3 | 5 | 4 | 2 | 3 |
| M2:E2 | 4 | 6 | 5 | | 2 | 1 | 3 | 4 |
| M2:S2 | 2 | 4 | 3 | 2 | | 3 | 1 | 2 |
| M2:ZU2 | 3 | 5 | 4 | 3 | 1 | | 2 | 3 |
| ROBOT:PINCE | 1 | 3 | 2 | 1 | 3 | 2 | | 1 |

## A.4. Opérations fonctionnelles

Après avoir défini les opérations réalisables par le système, il nous faut maintenant déterminer les ressources du système capables de les effectuer. Par exemple l'opération fraisage peut être effectuée sur la zone ZU1 de la ressource M1 et la zone ZU2 de la ressource M2.




**Copie d'écran A-12 : saisie des opérations fonctionnelles**

# Opérations fonctionnelles

**Lib Operation : *ASSEMBLAGE***
**Nu Operation *3***

**Lib Ressource : *M1***
**Nu Ressource : *3***

| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| ASSEMBLAGE1 | 4 | 4 | ZU1 |

**Lib Operation : *FINITION***
**Nu Operation *4***

**Lib Ressource : *M1***
**Nu Ressource : *3***

| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| FINITION1 | 5 | 4 | ZU1 |

**Lib Ressource : *M2***
**Nu Ressource : *4***

| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| FINITION2 | 6 | 7 | ZU2 |

**Lib Operation : *FRAISAGE***
**Nu Operation *2***

**Lib Ressource : *M1***
**Nu Ressource : *3***

| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| FRAISAGE1 | 2 | 4 | ZU1 |

**Lib Ressource : *M2***
**Nu Ressource : *4***

| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| FRAISAGE2 | 3 | 7 | ZU2 |

**Lib Operation : *TOURNAGE***
**Nu Operation *1***

**Lib Ressource : *M1***
**Nu Ressource : *3***

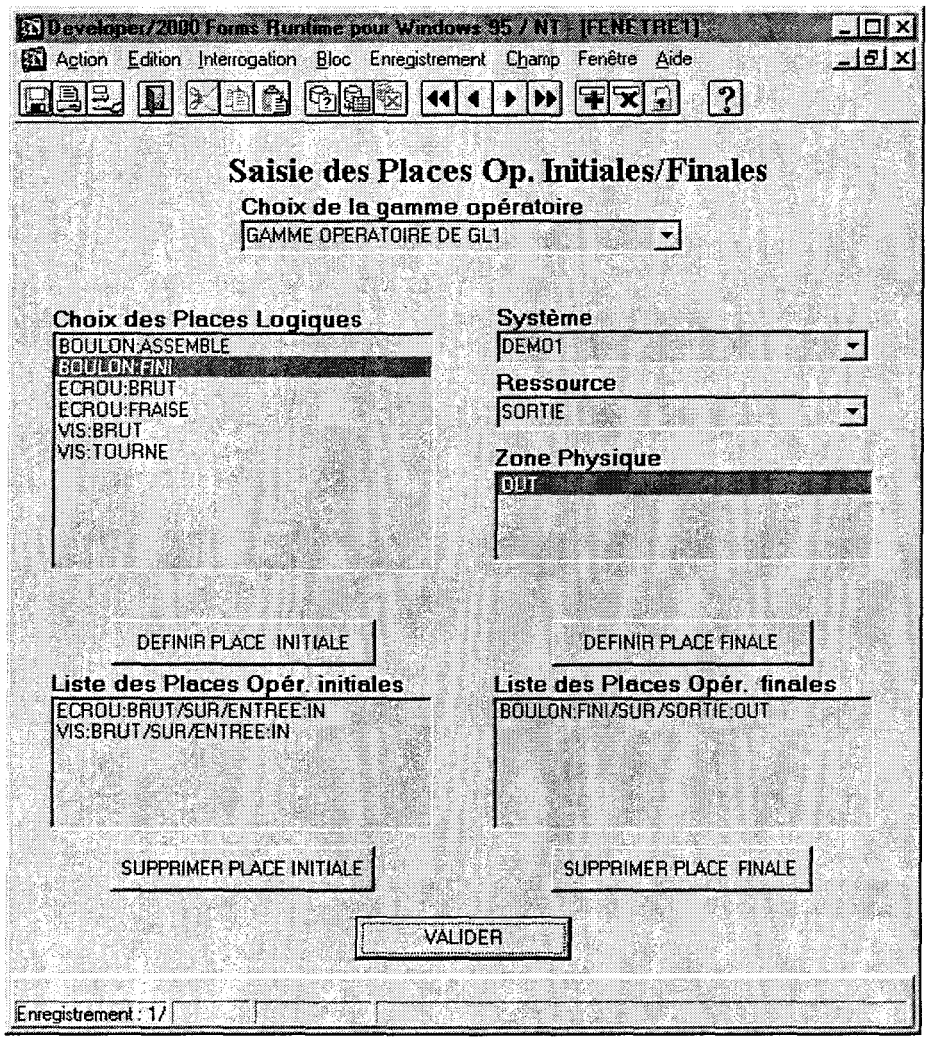
| Lib Op Fonct | Nu Op Fonct | Nu Zone Phy | Lib Zone Phy |
|---|---|---|---|
| TOURNAGE1 | 1 | 4 | ZU1 |

# B. Point de vue Commande

Les gammes opératoires sont obtenues à partir des gammes logiques, en prenant en compte les informations relatives à l'architecture physique.

## B.1. Attribution des places initiales et finales

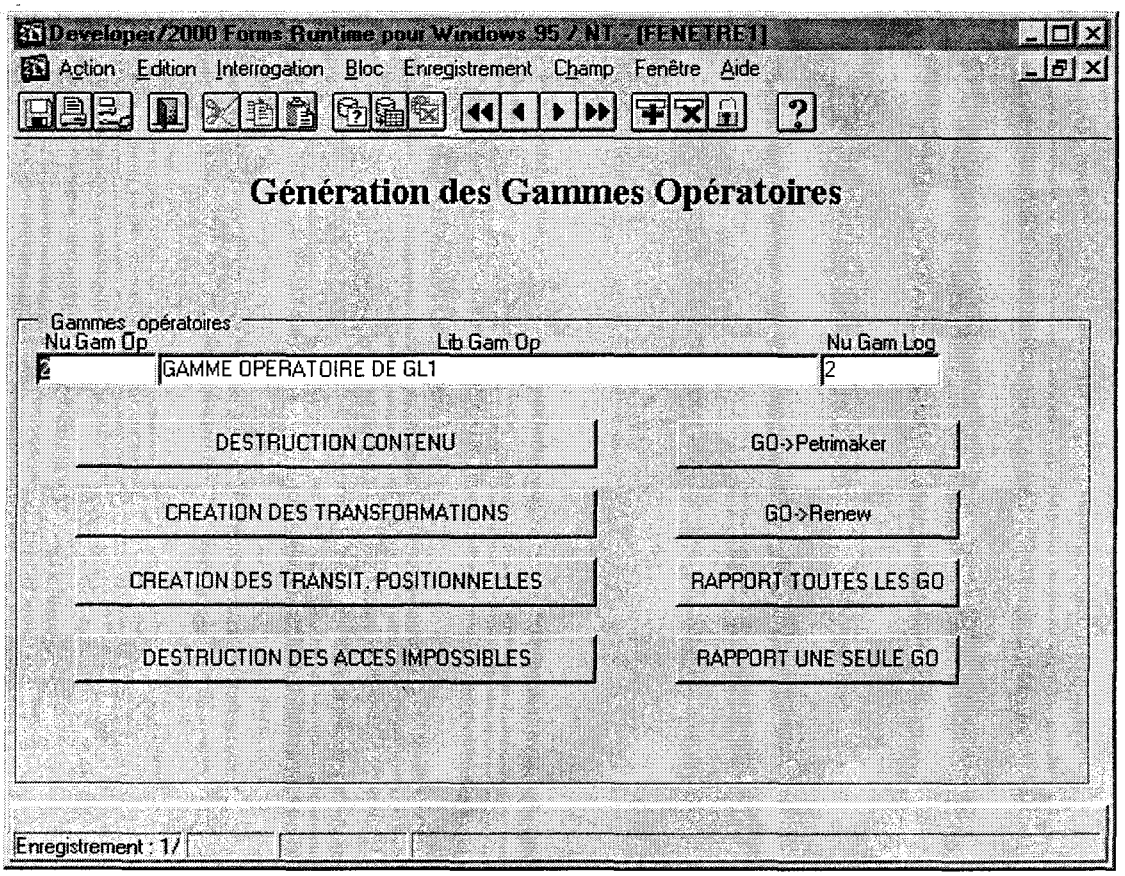
Pour créer une gamme opératoire à partir d'une gamme logique, l'utilisateur commence par définir les places initiales et finales de gamme opératoire. Ces places sont définies en associant une place de la gamme logique à une zone physique.




**Copie d'écran B-1 : saisie des places initiales et finales**

## B.2. Génération des gammes opératoires

Pour chacun des traitements figurant sur les gammes logiques, une liste des machines candidates, capables de réaliser le traitement considéré, est effectuée. Chaque traitement, ayant plusieurs machines possibles, générera une structure alternative traduite par une flexibilité de choix des machines et une flexibilité d'ordre des opérations.
Une distinction entre les transferts et traitements est effectuée, nous générons alors les opérations de transformation puis les opérations de transfert.

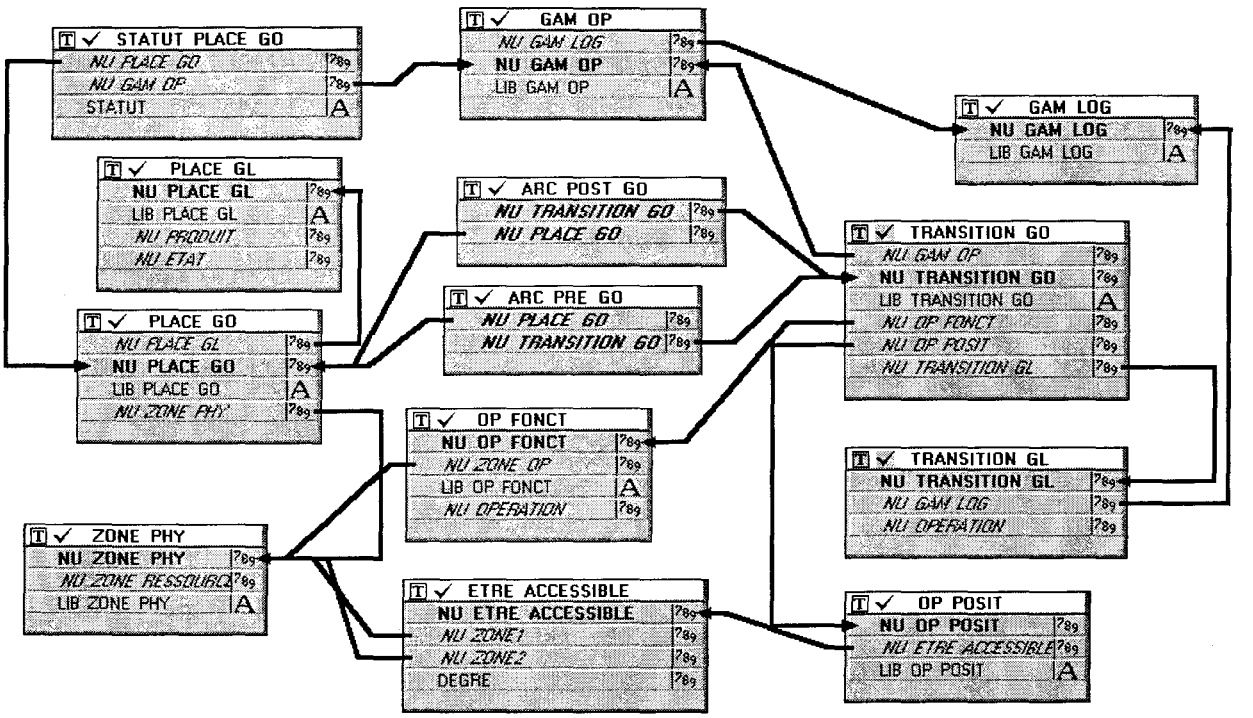



**Copie d'écran B-2 : génération des gammes opératoires**

Une gamme opératoire est représentée par un RdP dont les places représentent l'état d'avancement dans le processus de fabrication, et les transitions correspondent à des opérations de traitement ou de transfert du produit
La génération du RdP correspondant est effectuée sous Renew ou Petrimaker avec les conventions de notation suivantes :

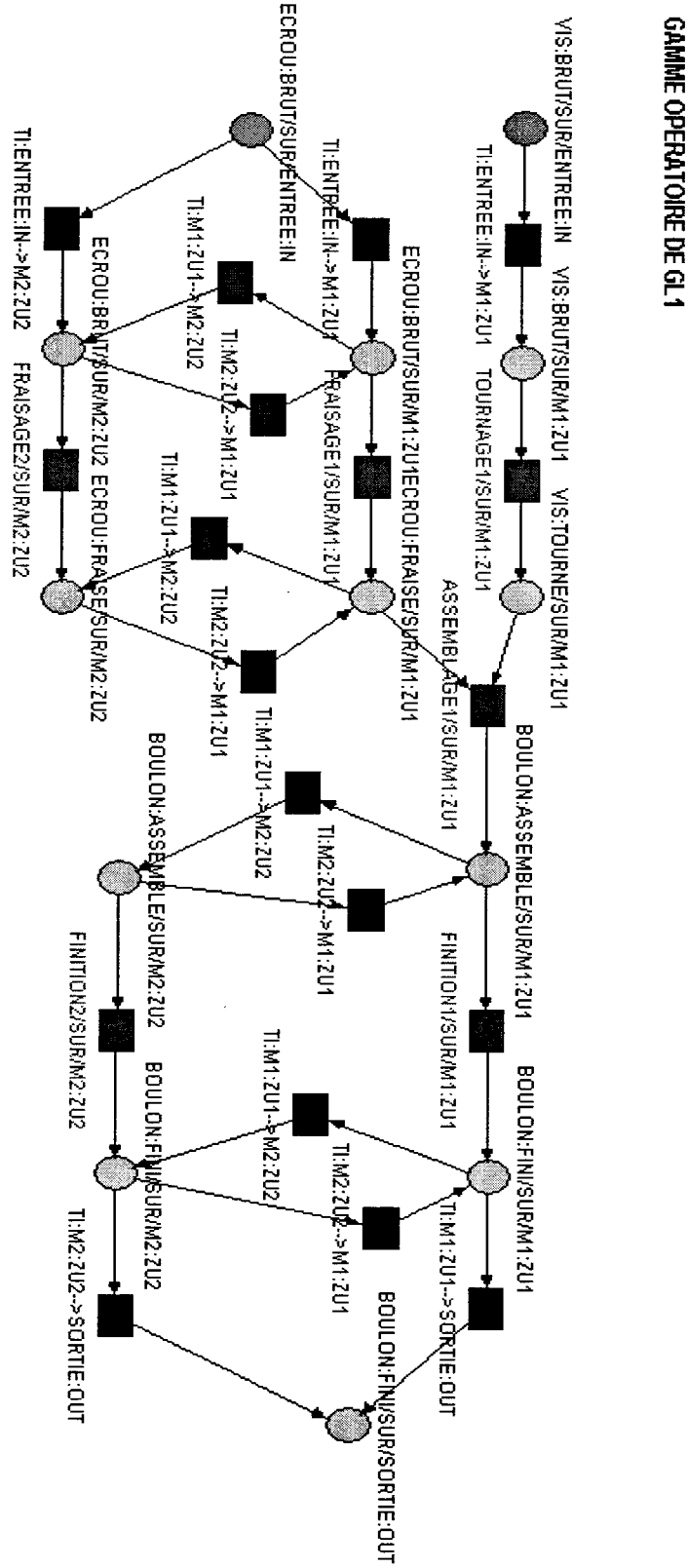
- pour chaque place : **Produit : Etat/ sur/ Ressource : zone physique** ;
- pour les transitions correspondant à une opération positionnelle :
  **TI : Ressource : zone physique de départ → Ressource : zone physique d'arrivée** ;
- pour les transitions correspondant à une opération de transformation :
  **Opération fonctionnelle/sur/ Ressource : zone physique**

Les données du point de vue commande sont stockées dans les tables ci-dessous.




**Copie d'écran B-3 : schéma de tables pour le point de vue commande**

Gamme opératoire générée à parti de la gamme logique « boulon » et de l'atelier de production sur la figure A-1.




**Copie d'écran B-4 : génération d'une gamme opératoire dans RENEW**

# C. Point de vue Evaluation de performances

## C.1. Création des ressources

Chacune des ressources (A1...A6) regroupe un ensemble de machines pouvant effectuer les mêmes opérations. Par exemple la ressource A1 est constituée des machines M11, M12 et M13 ; la ressource A3 regroupe les machines M21 et M31.

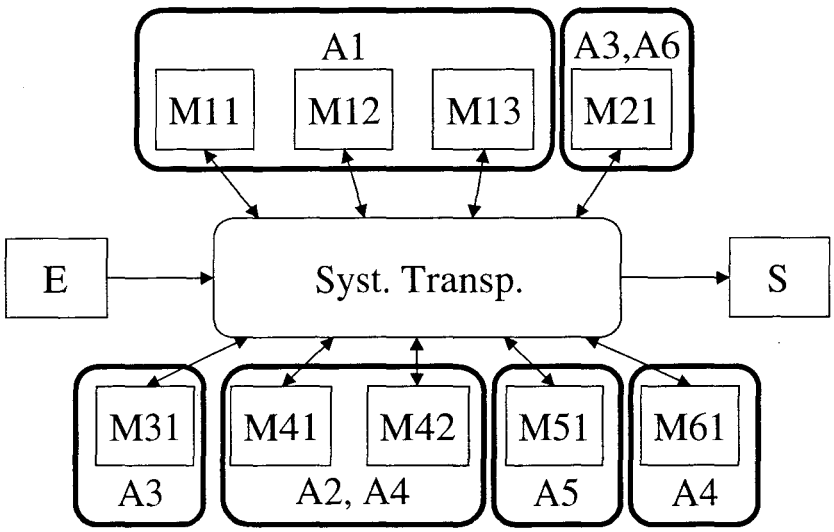
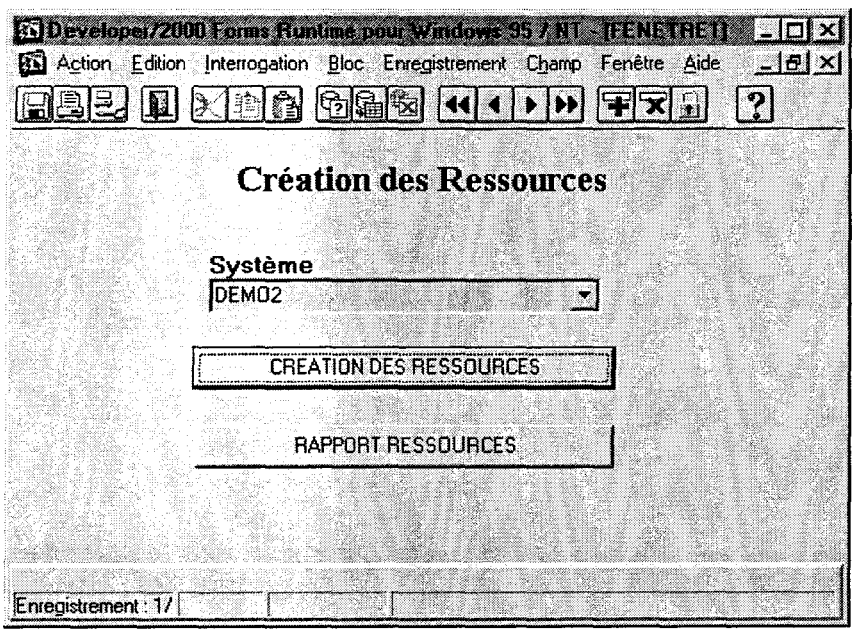
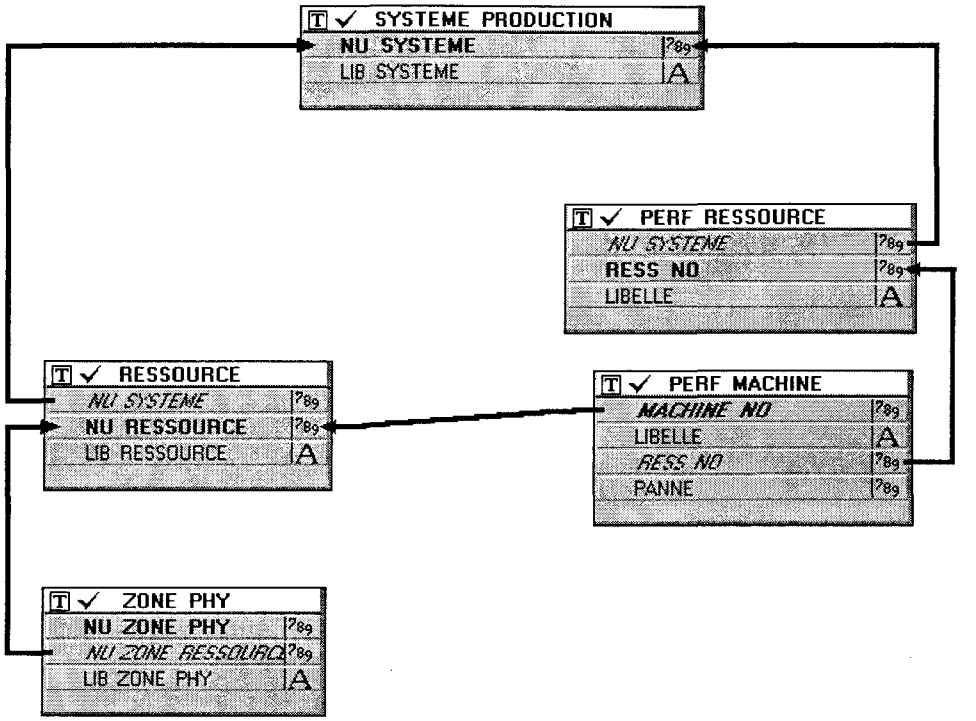



**Figure C-1 : exemple de regroupement en ressources**

La création des ressources est effectuée à partir des opérations fonctionnelles.




**Copie d'écran C-1 : création des ressources**




**Copie d'écran C-2 : schéma de tables avec les ressources**

# Structure des systèmes

Nu Systeme **2**
Lib Systeme : **DEMO2**

Nu Ressource :**6**
Lib Ressource : ***ENTREE2***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 10 | IN2 |

**NB zones phy de la ressource : 1**

Nu Ressource :**7**
Lib Ressource : ***SORTIE2***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 11 | OUT2 |

**NB zones phy de la ressource : 1**

Nu Ressource :**8**
Lib Ressource : ***M11***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 12 | ZU11 |

**NB zones phy de la ressource : 1**

Nu Ressource :**9**
Lib Ressource : ***M12***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 13 | ZU12 |

**NB zones phy de la ressource : 1**

Nu Ressource :**10**
Lib Ressource : ***M13***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 14 | ZU13 |

**NB zones phy de la ressource : 1**

Nu Ressource :**11**
Lib Ressource : ***M21***

# Structure des systèmes

Nu Systeme **2**
Lib Systeme : **DEMO2**

Nu Ressource :**11**
Lib Ressource : ***M21***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 15 | ZU21 |

**NB zones phy de la ressource : 1**

Nu Ressource :**12**
Lib Ressource : ***M31***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 16 | ZU31 |

**NB zones phy de la ressource : 1**

Nu Ressource :**13**
Lib Ressource : ***M41***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 17 | ZU41 |

**NB zones phy de la ressource : 1**

Nu Ressource :**14**
Lib Ressource : ***M42***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 18 | ZU42 |

**NB zones phy de la ressource : 1**

Nu Ressource :**15**
Lib Ressource : ***M51***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 19 | ZU51 |

**NB zones phy de la ressource : 1**

Nu Ressource :**16**
Lib Ressource : ***M61***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 20 | ZU61 |

**NB zones phy de la ressource : 1**

Nu Ressource :**17**
Lib Ressource : ***TRANSP***

| Nu Zone Phy | Lib Zone Phy |
|---|---|
| 21 | CONV |

**NB zones phy de la ressource : 1**

**NBZones phy du systèm 12** **NB Ressources du système: 12**

# Matrice d'accessibilit

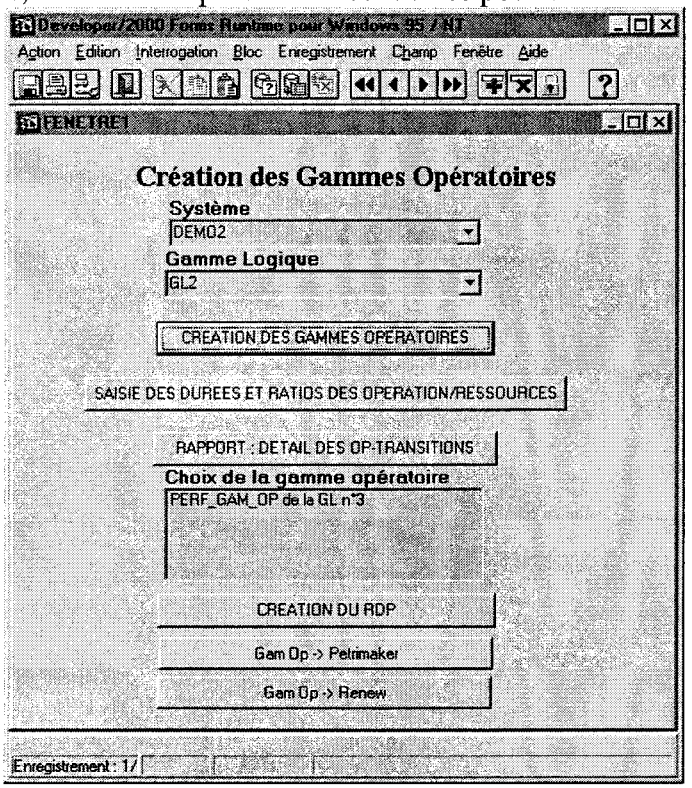
| Lieu Arrivee | M11:ZU11 | M12:ZU12 | M13:ZU13 | M21:ZU21 | M31:ZU31 | M41:ZU41 | M42:ZU42 | M51:ZU51 | M61:ZU61 | SORTIE2:OUT |
|---|---|---|---|---|---|---|---|---|---|---|
| Lieu Depart | | | | | | | | | | |
| ENTREE2:IN2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| M11:ZU11 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| M12:ZU12 | 2 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| M13:ZU13 | 2 | 2 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| M21:ZU21 | 2 | 2 | 2 | | 2 | 2 | 2 | 2 | 2 | 2 |
| M31:ZU31 | 2 | 2 | 2 | 2 | | 2 | 2 | 2 | 2 | 2 |
| M41:ZU41 | 2 | 2 | 2 | 2 | 2 | | 2 | 2 | 2 | 2 |
| M42:ZU42 | 2 | 2 | 2 | 2 | 2 | 2 | | 2 | 2 | 2 |
| M51:ZU51 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | | 2 | 2 |
| M61:ZU61 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | | 2 |
| TRANSP:CONV | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

30-MAI-00 15:58:09

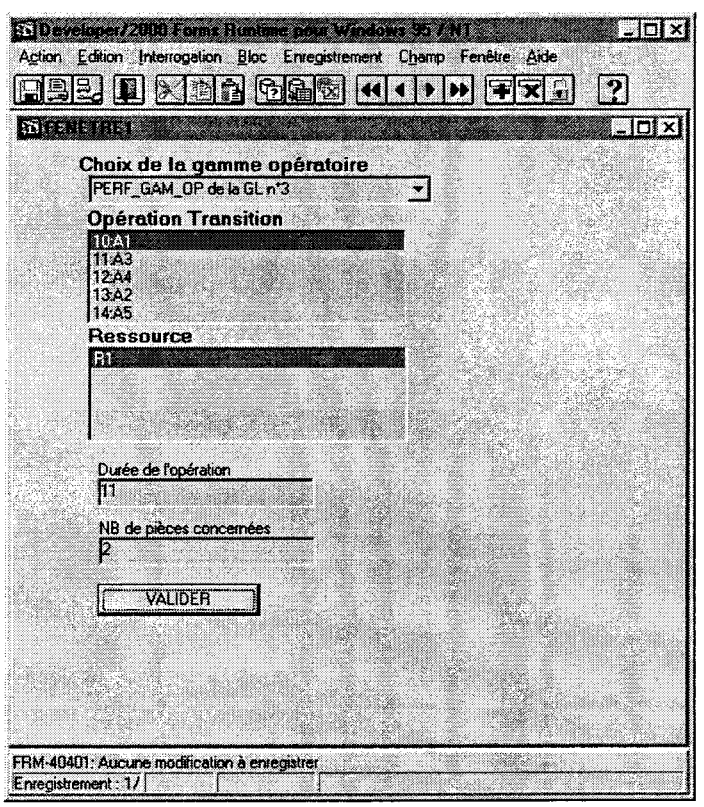
| Lieu Arrivee | TRANSP:CONV |
|---|---|
| Lieu Depart | |
| ENTREE2:IN2 | 1 |
| M11:ZU11 | 1 |
| M12:ZU12 | 1 |
| M13:ZU13 | 1 |
| M21:ZU21 | 1 |
| M31:ZU31 | 1 |
| M41:ZU41 | 1 |
| M42:ZU42 | 1 |
| M51:ZU51 | 1 |
| M61:ZU61 | 1 |
| TRANSP:CONV | |

## C.2. Gammes opératoires

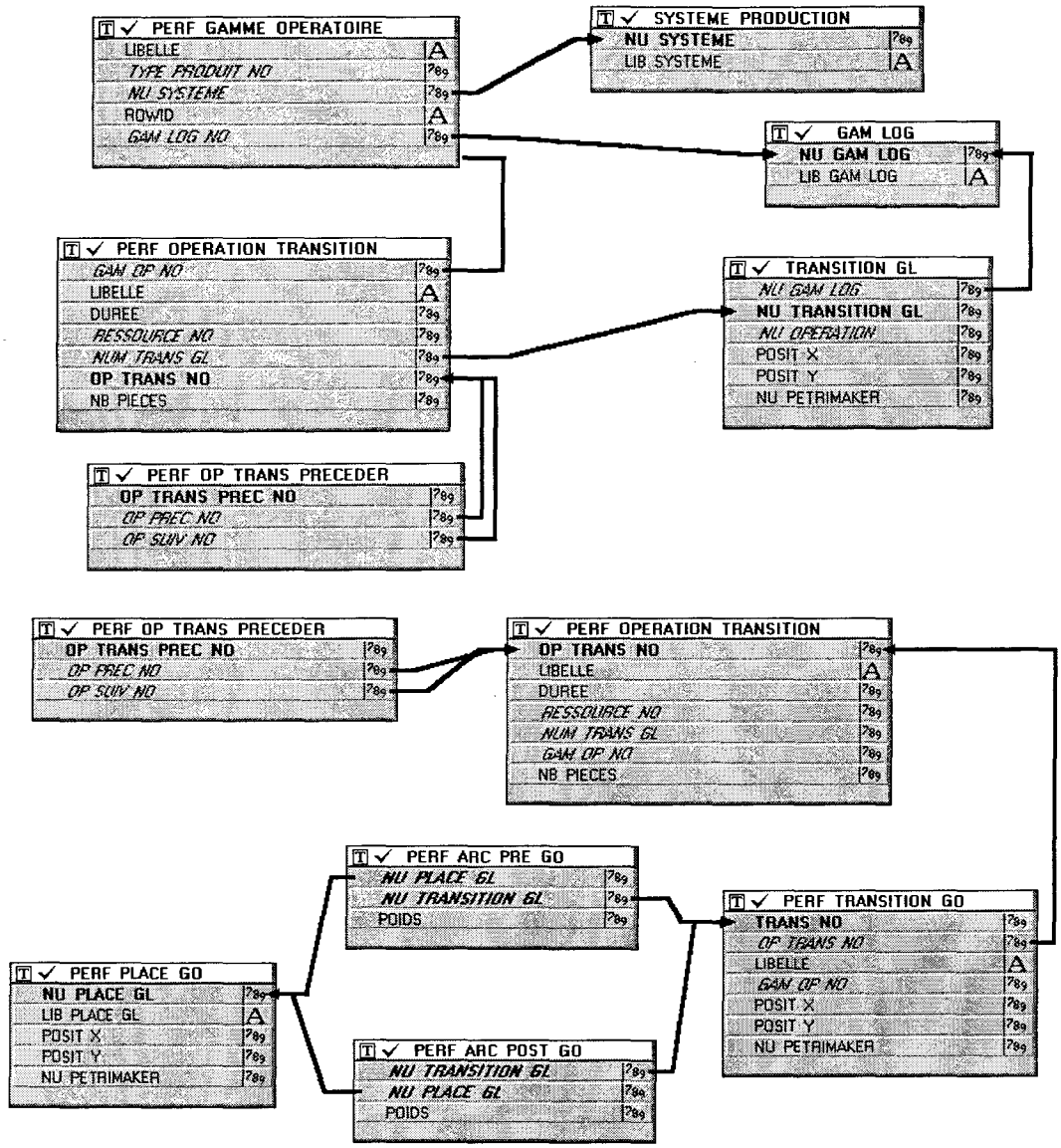
Les gammes opératoires sont utilisées pour modéliser les séquences d'opérations pour la fabrication des produits, les ratios de production sont fixés pour chacune d'elles.




**Copie d'écran C-3 : création des gammes opératoires**

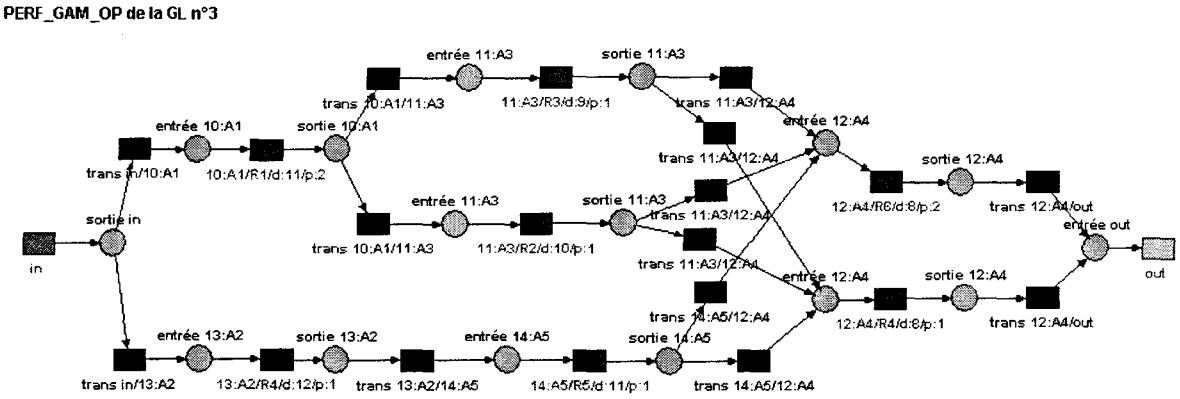



**Copie d'écran C-4 : durée des opérations et ratios**




**Copie d'écran C-5 : schéma de tables pour les gammes opératoires**

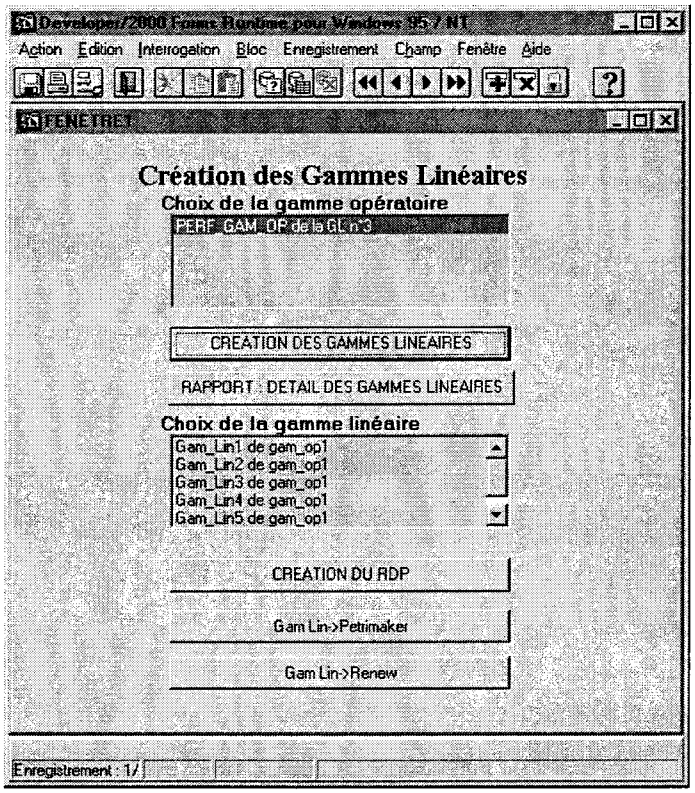
Pour chaque opération on définit la ressource sur laquelle elle est effectuée, sa durée et le nombre de pièces à usiner., d'où la convention de notation suivante : **Opération/Ressource/durée/nombre de pièces.**



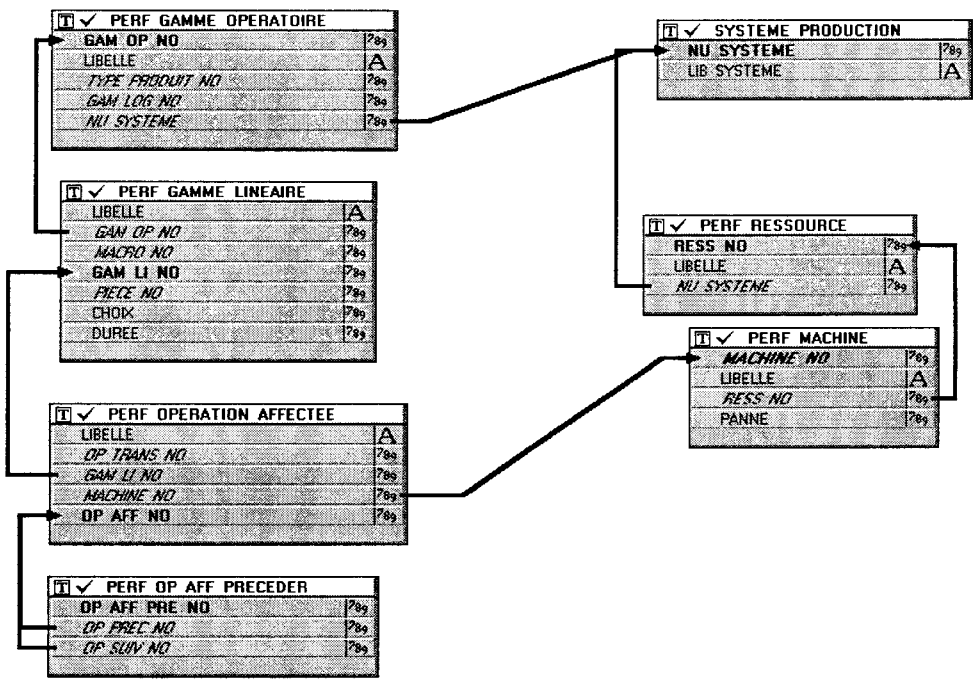

**Copie d'écran C-6 : gamme opératoire générée dans RENEW**

## C.3. Gammes linéaires

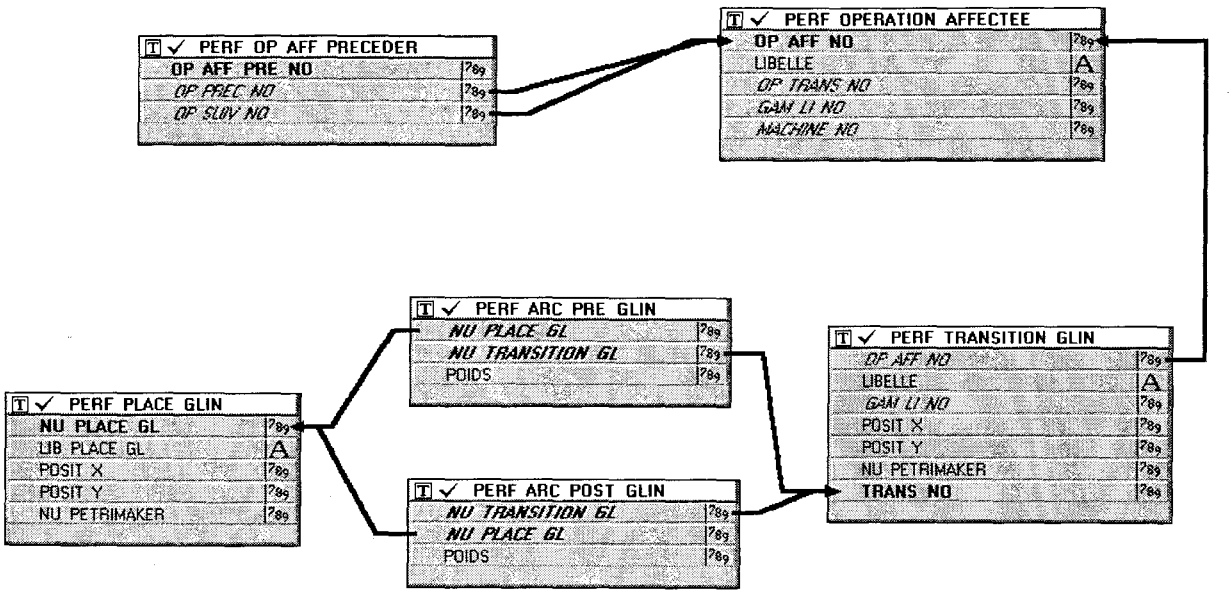
Tous les indéterminismes dus aux flexibilités d'affectation (la même opération réalisable par plusieurs ressources) de permutation (gammes partiellement ordonnées) et de procédé (substitution d'une suite d'opérations par une autre suite d'opérations) sont levés. On génère alors plusieurs gammes linéaires.



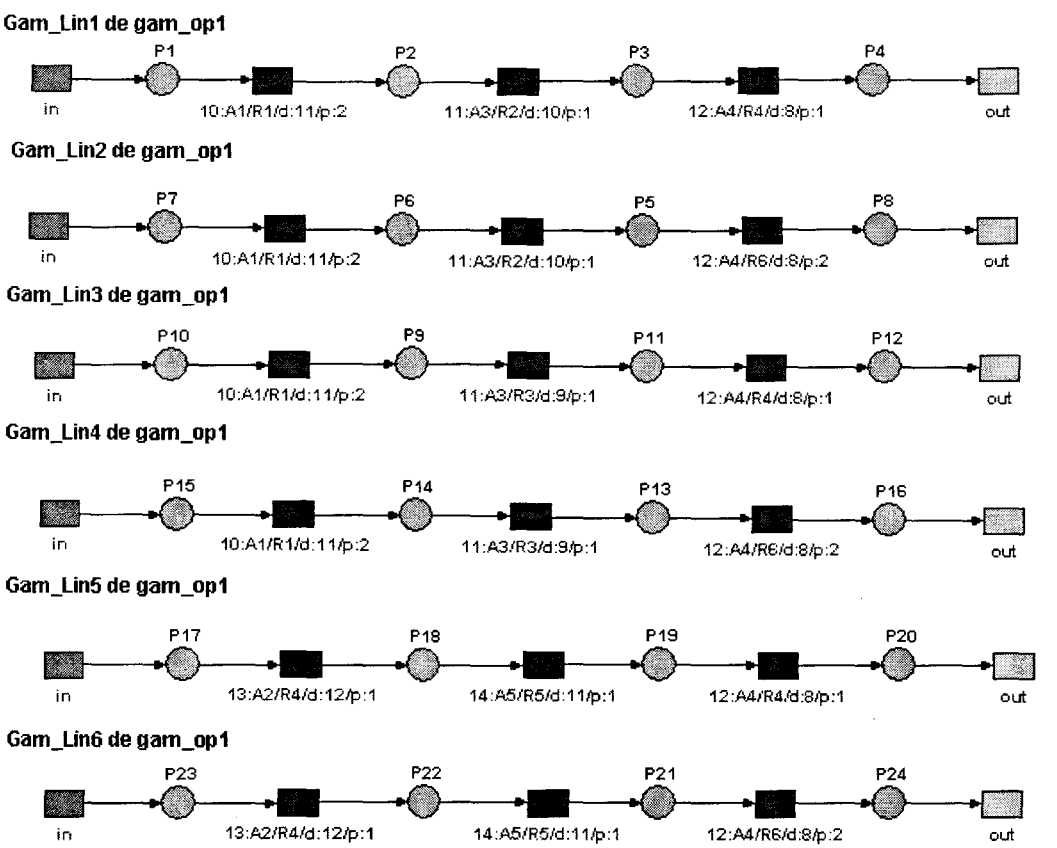

**Copie d'écran C-7 : création des gammes linéaires**




**Copie d'écran C-8 : schéma de tables gamme linéaire**




**Copie d'écran C-9 : suite schéma de tables gamme linéaire**




**Copie d'écran C-10 : gammes linéaires générées dans RENEW**

# Détail des gammes linéaires de PERF_GAM_OP de la GL n

**Gam Li No** 1
**Libelle** Gam_Lin1 de gam_op1

| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 3 | 10:A1 | 2 | R1 |
| 4 | 11:A3 | 1 | R2 |
| 6 | 12:A4 | 1 | R4 |

**Gam Li No** 2
**Libelle** Gam_Lin2 de gam_op1

| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 4 | 11:A3 | 1 | R2 |
| 3 | 10:A1 | 2 | R1 |
| 7 | 12:A4 | 2 | R6 |

**Gam Li No** 3
**Libelle** Gam_Lin3 de gam_op1

| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 3 | 10:A1 | 2 | R1 |
| 5 | 11:A3 | 1 | R3 |
| 6 | 12:A4 | 1 | R4 |

**Gam Li No** 4
**Libelle** Gam_Lin4 de gam_op1

| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 5 | 11:A3 | 1 | R3 |
| 3 | 10:A1 | 2 | R1 |
| 7 | 12:A4 | 2 | R6 |

**Gam Li No** 5
**Libelle** Gam_Lin5 de gam_op1

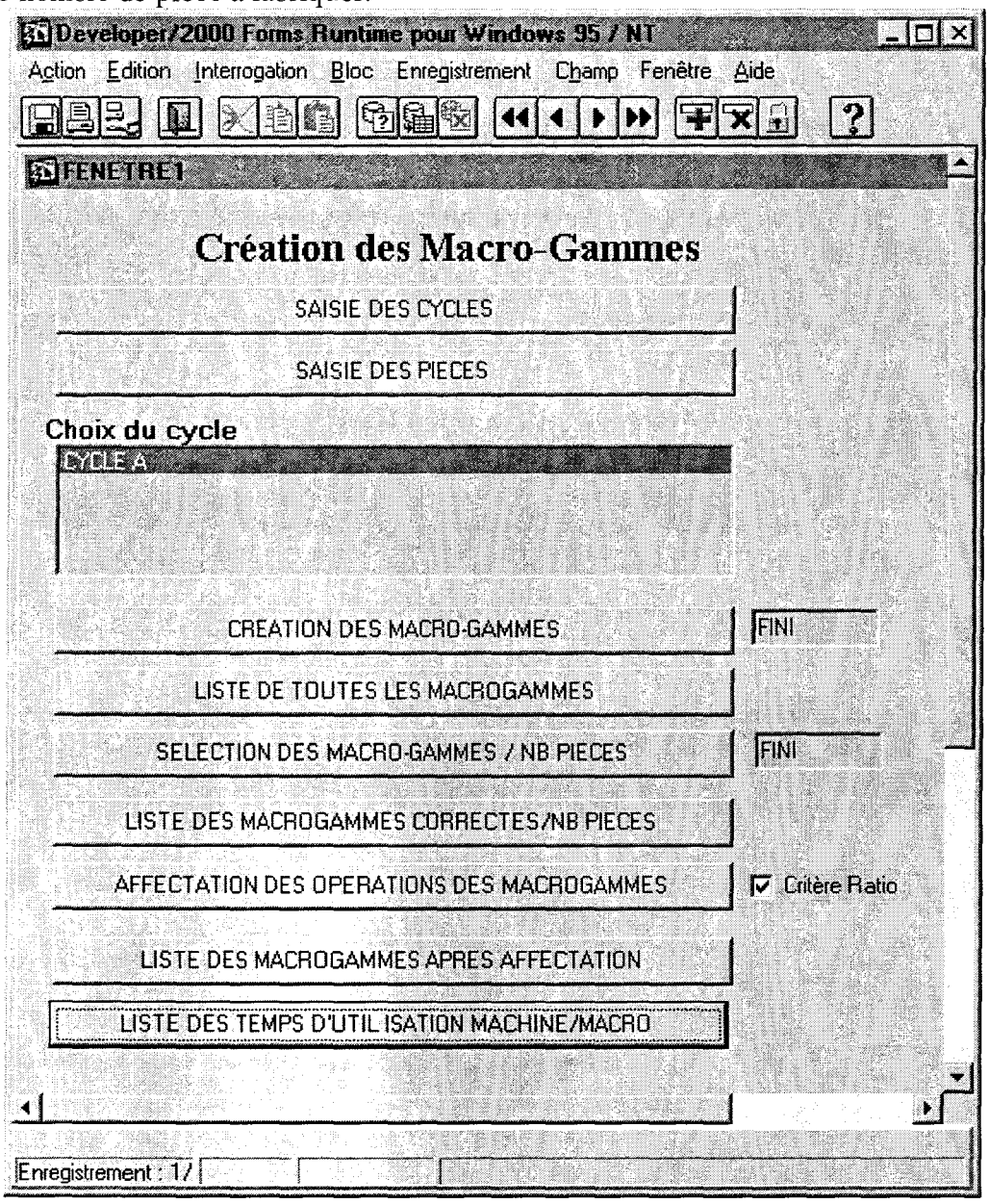
| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 8 | 13:A2 | 1 | R4 |
| 9 | 14:A5 | 1 | R5 |
| 6 | 12:A4 | 1 | R4 |

**Gam Li No** 6
**Libelle** Gam_Lin6 de gam_op1

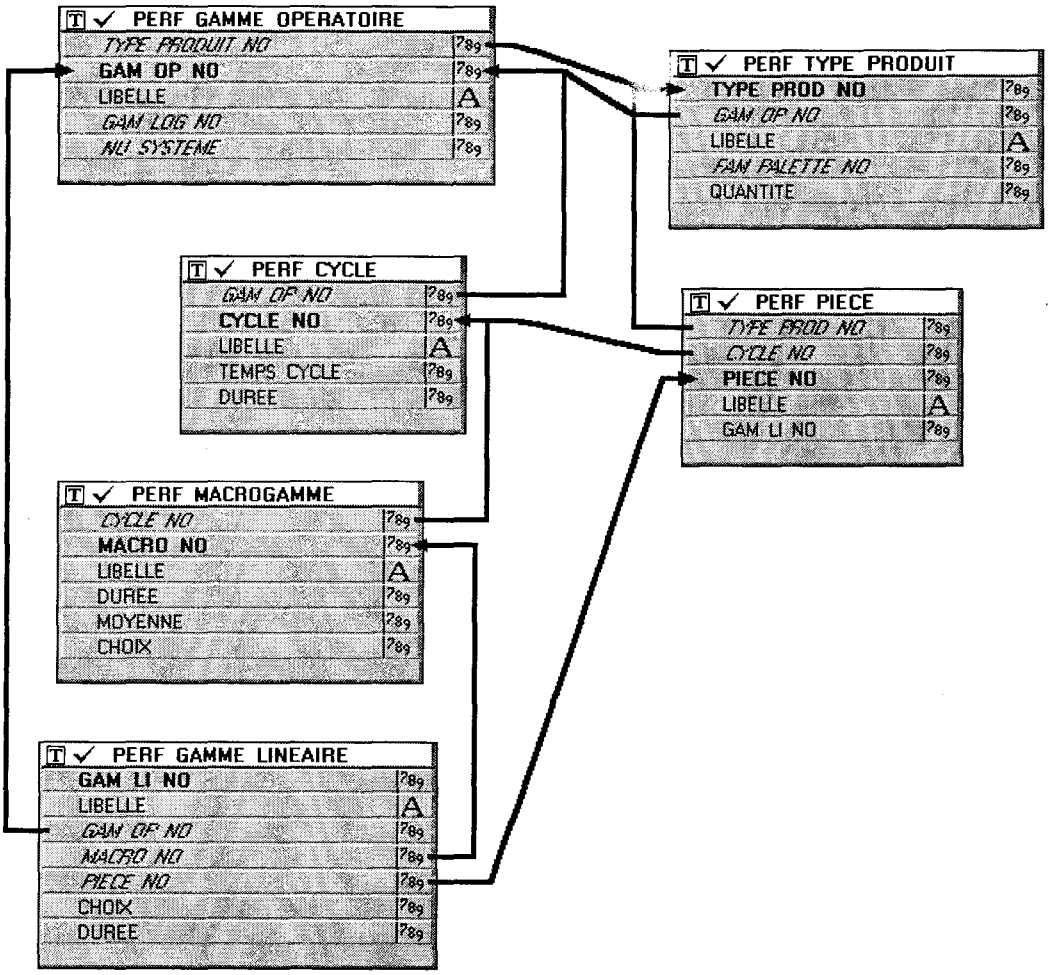
| Op Trans No | Opération | Nb Pieces | Ressource |
|---|---|---|---|
| 9 | 14:A5 | 1 | R5 |
| 8 | 13:A2 | 1 | R4 |
| 7 | 12:A4 | 2 | R6 |

## C.4. Macro-gammes

Les gammes opératoires sont linéarisées et regroupées de manière séquentielle dans des macro-gammes. Parmi la liste des macro-gammes générées, nous sélectionnons celles qui respectent le critère de nombre de pièce à fabriquer.




**Copie d'écran C-11 : création de macro-gammes**




**Copie d'écran C-12 : schémas de tables macro-gamme**

# Macro gammes du cycle : CYCLE A

**Macro No** 1
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 7 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 8 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 9 | Gam_Lin1 de gam_op1 | 3 | 1 | |

**Macro No** 2
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 10 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 11 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 12 | Gam_Lin2 de gam_op1 | 3 | 1 | |

**Macro No** 3
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 13 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 14 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 15 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 4
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 16 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 17 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 18 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 5
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 19 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 20 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 21 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 6
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 22 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 23 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 24 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 7
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 25 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 26 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 27 | Gam_Lin2 de gam_op1 | 3 | 1 | |

**Macro No** 8
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 28 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 29 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 30 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 9
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 31 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 32 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 33 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 10
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 34 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 35 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 36 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 11
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 37 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 38 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 39 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 12
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 40 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 41 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 42 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 13
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 43 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 44 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 45 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 14
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 46 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 47 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 48 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 15
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 49 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 50 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 51 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 16
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 52 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 53 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 54 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 17
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 55 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 56 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 57 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 18
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 58 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 59 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 60 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 19
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 61 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 62 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 63 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 20
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 64 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 65 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 66 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 21
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 67 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 68 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 69 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 22
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 70 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 71 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 72 | Gam_Lin2 de gam_op1 | 3 | 1 | |

**Macro No** 23
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 73 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 74 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 75 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 24
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 76 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 77 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 78 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 25
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 79 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 80 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 81 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 26
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 82 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 83 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 84 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 27
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 85 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 86 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 87 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 28
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 88 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 89 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 90 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 29
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 91 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 92 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 93 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 30
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 94 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 95 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 96 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 31
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 97 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 98 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 99 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 32
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 100 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 101 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 102 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 33
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 103 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 104 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 105 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 34
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 106 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 107 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 108 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 35
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 109 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 110 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 111 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 36
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 112 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 113 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 114 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 37
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 115 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 116 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 117 | Gam_Lin3 de gam_op1 | 3 | 1 | |

**Macro No** 38
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 118 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 119 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 120 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 39
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 121 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 122 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 123 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 40
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 124 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 125 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 126 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 41
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 127 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 128 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 129 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 42
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 130 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 131 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 132 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 43
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 133 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 134 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 135 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 44
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 136 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 137 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 138 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**Macro No** 45
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 139 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 140 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 141 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 46
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 142 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 143 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 144 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 47
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 145 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 146 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 147 | Gam_Lin4 de gam_op1 | 3 | 1 | |

**Macro No** 48
**Libelle1**
**Choix1** 0
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 148 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 149 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 150 | Gam_Lin5 de gam_op1 | 3 | 1 | |

| | | | | |
|---|---|---|---|---|
| **Macro No** | 55 | | | |
| **Libelle1** | | | | |
| **Choix1** | 0 | | | |
| **Cycle No** | 1 | | | |
| **Gam Li No** | **Libelle** | **Choix** | **Gam Op No** | **Piece No** |
| 169 | Gam_Lin5 de gam_op1 | 3 | 1 | |
| 170 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 171 | Gam_Lin6 de gam_op1 | 3 | 1 | |

| | | | | |
|---|---|---|---|---|
| **Macro No** | 56 | | | |
| **Libelle1** | | | | |
| **Choix1** | 0 | | | |
| **Cycle No** | 1 | | | |
| **Gam Li No** | **Libelle** | **Choix** | **Gam Op No** | **Piece No** |
| 172 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 173 | Gam_Lin6 de gam_op1 | 3 | 1 | |
| 174 | Gam_Lin6 de gam_op1 | 3 | 1 | |

30-MAI-00 16:34:47 **Macro gammes du cycle : CYCLE A**

**NB MACROGAMMES: 56**

30-MAI-00 16:35:43

# Macro gammes du cycle : CYCLE A respectant le critère de nombre de pièce

**Macro No** 18
**Libelle1**
**Choix1** 1
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 58 | Gam_Lin1 de gam_op1 | 3 | 1 | |
| 59 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 60 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 30
**Libelle1**
**Choix1** 1
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 94 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 95 | Gam_Lin3 de gam_op1 | 3 | 1 | |
| 96 | Gam_Lin6 de gam_op1 | 3 | 1 | |

**Macro No** 32
**Libelle1**
**Choix1** 1
**Cycle No** 1

| Gam Li No | Libelle | Choix | Gam Op No | Piece No |
|---|---|---|---|---|
| 100 | Gam_Lin2 de gam_op1 | 3 | 1 | |
| 101 | Gam_Lin4 de gam_op1 | 3 | 1 | |
| 102 | Gam_Lin5 de gam_op1 | 3 | 1 | |

**NB MACROGAMMES** 3

# qui respectent le critère de pièces



Num Macro 18
Lib Mac

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 11 | 9 | M12 | 1 | R1 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

Num Macro 30
Lib Mac

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 11 | 9 | M12 | 1 | R1 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

# Temps d'utilisation des machines pour les macrogamme du cycle CYCLE qui respectent le critère de pièces

**Num Macro** 32
**Lib Mac**

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 9 | M12 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

**qui respectent le critère de pièces**



Num Macro 18
Lib Mac

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 11 | 9 | M12 | 1 | R1 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

Num Macro 30
Lib Mac

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 11 | 9 | M12 | 1 | R1 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

# Temps d'utilisation minimaux des machines pour les macrogamme du cycle CYCLE qui respectent le critère de pièces

Num Macro 32
Lib Mac

| Temps | Num Mach | Lib Mach | Num Ress | Lib Ress |
|---|---|---|---|---|
| 16 | 16 | M61 | 6 | R6 |
| 12 | 14 | M42 | 4 | R4 |
| 11 | 8 | M11 | 1 | R1 |
| 11 | 9 | M12 | 1 | R1 |
| 11 | 15 | M51 | 5 | R5 |
| 10 | 11 | M21 | 2 | R2 |
| 9 | 12 | M31 | 3 | R3 |
| 8 | 13 | M41 | 4 | R4 |
| 0 | 10 | M13 | 1 | R1 |

# Table des matières