MASTER EN SCIENCES INFORMATIQUES

Spécification Formelle des Besoins pour un Système Productique
Quelques exemples d'application

Garcia, Joaquin; Gelenne, Alexandre

Award date:
1993

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grandgagnage, 21b
B-5000 NAMUR

Spécification Formelle des Besoins pour un Système Productique

Quelques exemples d'application

Joaquin Garcia
Alexandre Gelenne

Promoteur : Eric Dubois

Mémoire présenté en vue
de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1992 - 1993

Résumé

Ce mémoire traite de l'élaboration de spécifications des besoins pour les systèmes productiques. Les enjeux de l'Ingénierie des Besoins sont abordés et la nécessité de disposer d'une structure générique pour les systèmes productiques est mise en évidence. La modélisation d'une entreprise productique particulière est présentée. Cette modélisation comprend la hiérarchisation des différents composants. Ceux-ci sont organisés en trois niveaux. La structure d'état de chaque composant est donnée. Un agent est spécifié complètement, sa structure d'état ainsi que les contraintes régissant son comportement sont données. Une illustration du processus de raffinements successifs est également proposée.

Abstract

This thesis is related to the construction of requirements specifications for Computer Integrated Manufacturing (CIM) systems. The stakes of Requirements Engineering is studied and the need for a generic structure for the CIM systems is stressed. One modelization of a particular CIM plant is presented. This modelization is composed of a hierarchy of several components. Those are organised in 3 levels. For each component, a definition of its state structure is given. A specific agent is completely specified, its state structure as well as behavioural constraints are given. An illustration of the refinement process is also proposed.

Remerciements

Nous tenons à exprimer notre gratitude envers Monsieur le Professeur Eric Dubois, promoteur de ce mémoire, pour son soutien constant et ses remarques judicieuses nécessaires à l'amélioration de la qualité de ce travail.

Nous sommes également reconnaissants envers Michaël Petit, notre prédécesseur et ami, pour ses explications ainsi que pour ses lectures vigilantes.

Merci aussi à Monsieur Philippe Du Bois, pour les explications concernant le langage ALBERT.

Nous tenons également à remercier Monsieur Claude Wehenkel, sans qui ce stage n'aurait pas été possible, les membres du Centre de Recherche Public Henri Tudor de Luxembourg pour leur acceuil. En particulier, l'équipe du Centre de Ressources CIM, Yves Wantz, pour toutes les informations qu'il nous a fournies sur l'atelier flexible . Les membres du projet CIM-Orienté Objet, l'équipe de Génie Logiciel et Patrick Moes, Patrick Grivet, José Mendez et Fabrizio Sitzia pour leur compagnie quotidienne.

Un remerciement tout spécial est adressé à Monsieur Jean-Pol Michel, responsable de notre stage; il nous a permis de collaborer à un projet dans le domaine de la productique.

Merci à nos parents, sans qui nous n'aurions pu entreprendre ces années d'étude.

Merci également à nos "collègues" d'étude pour avoir su créer une ambiance agréable tout au long de ces années, autant dans les moments sérieux que dans les moments de détente.

Chapitre 3
Vers une structure générique des systèmes productique

Chapitre 4
Structuration de l'atelier flexible

Chapitre 5
Spécification complète d'un agent de transport : le système AGV

Introduction

L'informatique prend une part de plus en plus importante dans les entreprises manufacturières, non seulement dans les départements administratifs mais aussi dans les départements de production. En effet, n'est-il pas rare de voir une entreprise manufacturière qui n'opte pas pour une informatisation importante (voire même totale) de ses moyens de production. Pour les entreprises, cette automatisation est un moyen de rester dans la course à la productivité.

Cependant, l'automatisation des entreprises se fait malheureusement département par département. Alors que dans une entreprise, l'ensemble de ceux-ci collaborent à la construction d'un ou plusieurs produits ou services.

Une nouvelle dimension vient donc s'ajouter à la complexité, déjà importante, des systèmes d'information (isolés). Cette dimension est la coopération des différents systèmes d'information. Pour assurer une telle coopération ou plus exactement une intégration des (sous) systèmes, il convient de maîtriser l'ensemble de l'organisation ainsi que son fonctionnement.

Pour atteindre cette maîtrise, il faut définir avec précision les objectifs attendus des systèmes d'information. Ces objectifs sont exprimés par les dirigeants de l'organisation.

Le principal problème provient justement de la formulation de ces objectifs, ceux-ci sont trop souvent incomplets, imprécis, ambigus, voire même contradictoires. Afin de pallier à ce manque de précision et de complétude de l'expression des besoins des futurs utilisateurs, plusieurs "langages" ou "méthodes" ont été élaborées depuis plus d'une

dizaine d'années [Ros77], [Tar83], [Bod89]. Cependant, de tels langages possèdent une expressivité limitée puisqu'ils mettent l'accent sur les structures de données et sur les flux d'information. Ils ne disposent pas de moyens pour exprimer des contraintes temps réel et de performance qui font, cependant, partie intégrante de l'expression des objectifs du futur système d'information.

Pour "corriger" les défauts de tels langages, de nouveaux formalismes ont été introduits. Ceux-ci reposent sur bases plus formelles (algébriques, logiques, ...). Grâce à de tels langages, on peut garantir l'interprétation unique d'une expression des besoins. De plus, des règles sont introduites afin de vérifier la complétude, la cohérence des spécifications. On a également ajouter des mécanismes de structuration plus complets permettant d'organiser les informations.

Les développements informatiques sont rarement isolés. Nous voulons dire par là, qu'il est rare qu'un certain type d'informatisation ne se fasse qu'une seule fois. En effet, par exemple, si une entreprise automobile introduit un système informatique permettant de contrôler l'ensemble de la production, il est impensable (si ce système s'avère une réussite) que les autres entreprises du même secteur restent indifférentes et donc elles se lanceront également dans la course à l'informatisation. C'est dans ce sens, que plusieurs auteurs comme [Bar93], [Cas93] préconisent la construction de structures génériques adaptées à un domaine précis. Chaque système développé pour une entreprise de ce domaine, sera donc conçu au départ d'un système d'information générique de ce secteur.

Ce travail a pour principal objectif d'appliquer, ou plus exactement utiliser, une structure générique pour le domaine particulier qu'est la productique (*Computer Integrated Manufacturing* en anglais). En plus de l'instanciation de cette structure, une évaluation a été apportées.

Le présent travail s'organise autour de 5 chapitres. Le premier est consacré à la présentation des enjeux de l'Ingénierie des Besoins. Le second chapitre présente un langage pour l'expression des besoins qui est utilisé dans la suite du travail. Ensuite, le troisième chapitre présente la structure générique qui a été instanciée ainsi qu'une évaluation de celle-ci. Le quatrième chapitre correspond à l'instanciation proprement dite de la structure générique à une petite entreprise productique. Dans ce chapitre on trouvera une description détaillée des équipements, ainsi que la proposition d'organisation de l'entreprise. Enfin, un dernier chapitre est consacré à un équipement particulier de l'entreprise. Pour cet équipement, nous proposons une spécification complète.

Chapitre 1

Les enjeux de l' Ingénierie des Besoins

1.1. Introduction

Dans ce chapitre, nous précisons la notion de système d'information tel que nous l'utiliserons dans la suite de ce travail. Nous présentons les principales étapes du développement des systèmes d'information. Ensuite, nous nous attarderons sur l'étude des besoins des futurs utilisateurs, afin de montrer l'importance que revêt cette étape en présentant quelques types d'erreurs les plus couramment rencontrés. A la suite de cela, nous proposerons une série de recommandations qu'il serait bon de suivre lors de l'étape d'analyse des besoins. Enfin, nous terminerons ce chapitre en présentant les tendances actuelles en Ingénierie des Besoins essentiellement dans le domaine des systèmes composites et temps réel.

1.2. Système d'information

"*Les organisations conçoivent, réalisent et utilisent des systèmes d'information pour satisfaire les besoins en informations engendrés par leurs* ***comportements organisationnels***." [Bod89].

Cette première définition présente le système d'information comme étant une boîte noire, dont on ne connaît que les objectifs.

Le système d'information est un ensemble d'informations, de traitements de ces informations, de règles d'organisation, de ressources humaines et techniques. Le système d'information n'est donc pas réduit au seul système automatisé ou informatique mais il englobe des traitements automatisés, interactifs et même exclusivement manuels.

David Barstow présente le système d'information comme étant "*une collection de matériels et de programmes qui fonctionnent ensemble pour calculer certains résultats ou causer certains effets*" [Bar93].

Nous serions tentés d'intégrer ces deux définitions en faisant intervenir la notion d'acteur. La définition du système d'information pourrait être :

Un système d'information est une collection d'acteurs qui gèrent, traitent de l'information, de façon automatisée, interactive, manuelle ,selon des règles d'organisation. Les acteurs (machines, systèmes informatiques, personnes) collaborent pour satisfaire les besoins en informations de l'organisation.

Cette nouvelle définition a pour avantage de pouvoir s'adapter aux systèmes composites ainsi qu'aux systèmes productiques constituant, en fait, le cadre de ce travail.

1.3. Le cycle de vie d'un système

Le cycle de vie d'un système d'information débute à partir du moment où l'organisation envisage ou décide de le construire. Les grandes étapes de ce cycle de vie sont : l'étude des besoins (ou analyse d'opportunité chez certains auteurs), une étape de développement proprement dit du système, une période plus ou moins longue pendant laquelle le système est utilisé. Le cycle de vie se termine lorsque l'organisation décide que le système est devenu obsolète ou inadéquat.

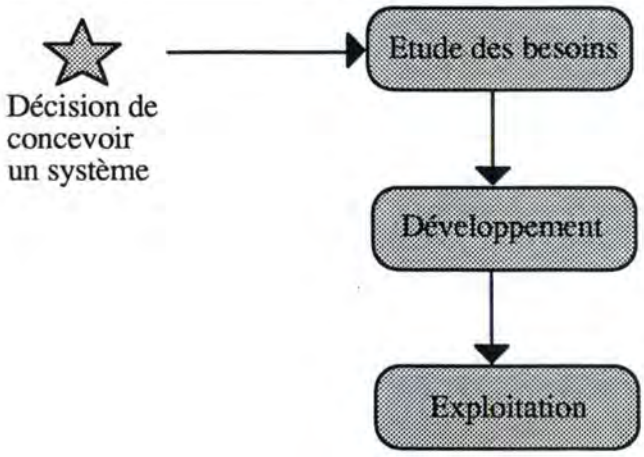


Figure -**1.1** - : Le cycle de vie d'un système d'information

Dans le cadre de ce travail, nous nous intéresserons plus particulièrement et exclusivement à la phase d'étude des besoins ainsi qu'à la première partie de l'étape de développement du système (*design*).

L'étape de *développement du système*, peut être décomposée en deux parties, la première consiste à fournir une solution "**logique**" ou abstraite, en ne tenant compte d'aucune décision d'implémentation. La solution issue de cette phase est exécutable sur une machine abstraite. La deuxième étape de développement consiste à fournir une solution "**effective**" qui, cette fois, sera exécutable sur une machine réelle. A l'issue de cette étape, on dispose alors d'une solution exploitable. On trouvera à la figure suivante, une représentation de cette démarche.

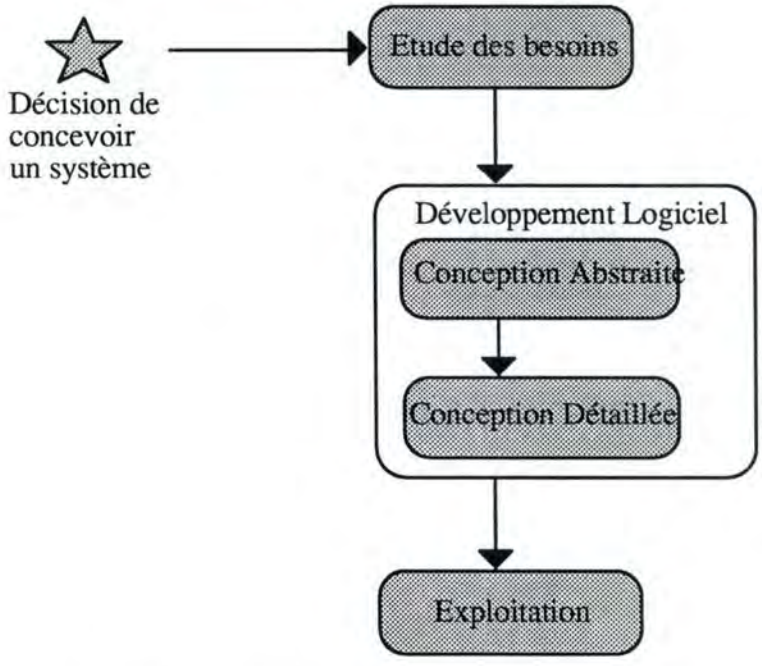


Figure -**1.2** - : Le cycle de vie d'un système d'information

Le cycle de vie d'un système d'information débute donc par la phase d'étude des besoins qui constitue la partie essentielle du développement. Une mauvaise étude de ces besoins risque d'avoir de graves conséquences sur le système. Cette phase doit se faire avec la plus grande rigueur, car les éléments mis en évidence lors de cette étape constituent les bases même de tout développement futur du système.

1.4. L'étude des besoins

1.4.1. Définition

Lors de cette étape, le concepteur du système doit s'attacher à cerner et comprendre au mieux les besoins des futurs utilisateurs ou, plus généralement, les clients.

De nombreuses études montrent que beaucoup de systèmes développés sont sous exploités voire même laissés à l'abandon faute de ne pas rencontrer les exigences des utilisateurs. Boehm signale que dans les grands systèmes, 95 % du code ont du être réécrits afin de répondre aux exigences changeantes des utilisateurs, et que 12 % des erreurs détectées après 3 ans d'utilisation, remontent en fait à la phase d'étude des besoins des utilisateurs.

L'activité d'Ingénierie des Besoins est loin d'être une simple retranscription des désirs des futurs utilisateurs. Ceux-ci souffrent d'imprécision, de manque de cohérence, de structuration, choses qui rendent le travail de l'analyste des besoins des plus délicats. *"Il convient donc de mettre en place des processus interactifs et répétitifs permettant au client d'exprimer ses volontés en vue ensuite de les valider, de les compléter,..."* [Pet92].

1.4.2. Les étapes de l'Etude des Besoins

L'étude des besoins peut être décomposée en quatre étapes qui sont, dans l'ordre, l'élicitation, la modélisation, l'analyse et enfin la validation. Ces quatre étapes sont regroupées en deux activités : l'**acquisition** et la **spécification**.[Dub91]

- La *phase d'élicitation* consiste en la collecte d'informations sur le problème posé par le client. Cette collecte peut prendre plusieurs formes : interviews, discussions, étude de l'existant, étude de la documentation disponible, ...

- La *modélisation* traite les informations collectées lors de la phase d'élicitation pour obtenir un modèle (c'est-à-dire une représentation de problème).

- La *phase d'analyse* tente de mettre à jour des problèmes (tels que des contradictions, des inconsistances, ...) lors de l'élaboration du modèle.

- La *phase de validation*. Lors de cette étape, la modélisation du problème est confrontée aux besoins des clients en vue de vérifier l'adéquation du futur système avec les attentes des utilisateurs.

On peut présenter ce processus par la figure suivante :

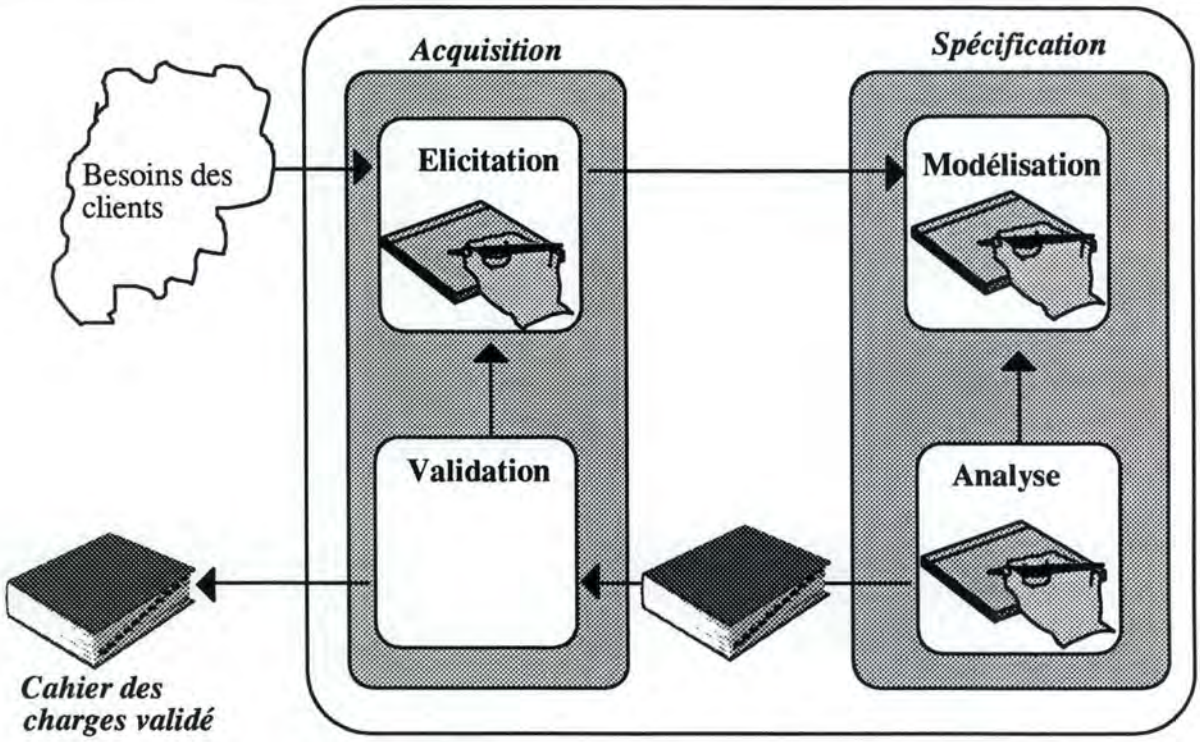


Figure **-1.3** - : L'Etude des Besoins

Le point de départ de l'étude des besoins est un ensemble de souhaits exprimés par le client. Ces souhaits ne peuvent être exploités directement car ils sont inconsistants, incomplets, ambigus, non documentés, ... [Rze93]. Le but de l'analyste des besoins est donc structurer ces besoins, de les analyser et produire ensuite un document spécifiant, avec précision, les attentes des utilisateurs. Ce document constitue la base du développement ultérieur, il peut être comparé à un contrat liant les développeurs avec les clients.

On peut trouver dans [Rze93] un modèle plus complet de la phase d'étude des besoins. On y trouvera notamment une vue détaillée de l'étape de validation.

1.4.3. Importance de l'Etude des Besoins

Une bonne analyse des besoins des futurs utilisateurs ne garantit pas le succès du nouveau système d'information. "*Cependant, aucun projet ne peut aboutir sans une description claire et précise des besoins des utilisateurs et donc des attentes de ceux-ci envers le système d'information.*"[Mil82]

On constate qu'actuellement, les prix d'un développement software deviennent plus onéreux que les développements hardware. Cette nouvelle tendance engendre un intérêt croissant pour les recherches dans le domaine du Génie Logiciel (*Software Engineering*).

Il est important de souligner que les erreurs trouvant leur origine dans la phase d'analyse des besoins sont les erreurs les plus difficiles et les plus coûteuses à corriger, car elles risquent de remettre en cause les fondements même du système.

Il subsiste cependant un gros problème lors de cette étape. La compréhension et l'interprétation des besoins des utilisateurs peuvent différer d'un individu à l'autre. Cette divergence peut être réduite si l'on dispose d'un "langage" de communication, d'expression des besoins, qui soit le plus clair possible, et qui réduise au maximum l'interprétation ambiguë des besoins.

La collecte des informations peut prendre plusieurs formes : interviews, discussions, étude de la documentation disponible, observation, ... Sur base des informations recueillies, on élabore un modèle du futur système.

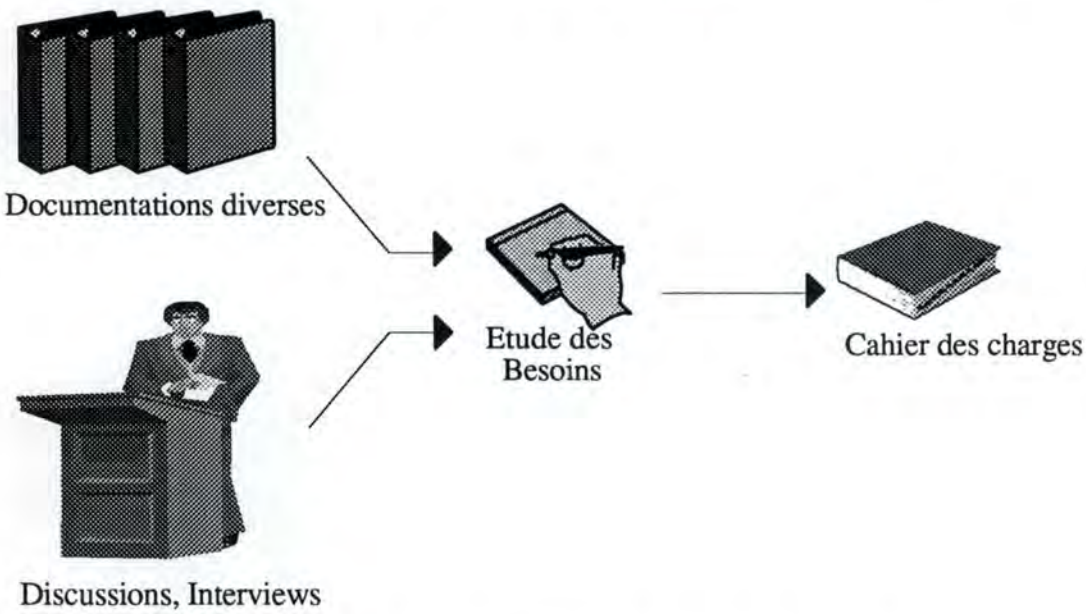


Figure -**3.4** - Les sources d'informations de l'étude des besoins

La plupart du temps, on utilise le langage naturel (Français, Anglais, ...) pour la communication entre développeurs et utilisateurs, son principal avantage est d'être communicable à tout un chacun. Mais le défaut majeur de tels langages est qu'un même texte peut être compris, interprété de manière différente selon les lecteurs.

Même si on dispose de langages pour exprimer les spécifications du futur système d'information (par exemple IDA, MERISE, SADT); la plupart de ceux-ci souffrent d'un manque de sémantique formelle. C'est donc dans ce but que d'autres langages ont été développés (RML, GIST, ERAE, TELOS, ALBERT). Ces derniers disposent d'une sémantique logique mathématique qui permet d'exprimer de façon consistante et non ambiguë les besoins des utilisateurs.

Ils permettent une meilleure structuration du cahier des charges. Dans ce sens, le paradigme "Orienté-Objet" semble voué à une utilisation plus intensive, non plus seulement dans le domaine de la programmation, mais aussi dans les phases préliminaires à ce développement telle que la phase d'étude des besoins.

Dans la suite de ce travail, nous nous intéressons plus particulièrement aux systèmes hétérogènes ou systèmes composites faisant intervenir des composants très divers et où la notion de temps réel est importante.

1.4.4. Les erreurs couramment relevées dans la conception de systèmes d'information

Les propos illustrés ci-dessous sont inspirés de [Lut93].

Une erreur software est définie comme étant une différence entre la valeur calculée, mesurée selon les cas, par le système et la valeur théorique ou valeur réelle.

Outre les erreurs de codage, la plupart des erreurs relevées dans un système critique sont issues d'une divergence entre les spécifications des besoins et les attentes des utilisateurs. Nous dénommons ces erreurs : *fautes de fonctionnalité*. Elles concernent des opérations qui n'étaient pas nécessaires, qui ont été oubliées, ou encore qui ne répondent pas aux attentes.

Les erreurs de fonctionnalité sont des erreurs couramment rencontrées, par exemple, dans le système de la sonde spatiale Gallilée, 79% des erreurs retrouvées étaient de ce type.

Des problèmes peuvent également se présenter lorsque le système à réaliser doit être interfacé avec un ou plusieurs autres systèmes .Les erreurs de ce type sont appelées *fautes d'interfaçage*. Elles concernent les interactions entre différents systèmes.

Les fautes d'interfaçage trouvent leur origine dans des problèmes de communication au sein d'une équipe ou entre plusieurs équipes (93% des erreurs du système de la sonde Voyager étaient des fautes d'interfaçage)

En résumé, les difficultés lors de l'analyse des besoins sont les causes principales des erreurs des systèmes. Ces erreurs persistent jusqu'à la phase d'intégration des différents sous-systèmes et la phase de test.

Les erreurs issues des phases de l'Etude des Besoins (*Requirements Engineering)* et de Conception (*Design)* sont plus difficiles à trouver et à corriger que celles insérées dans les phases ultérieures car elles impliquent des structures de programme plus complexes.

1.4.5. Le développement de l'Etude des Besoins

Au départ, l'étude de besoins se déroulait essentiellement par des interviews, des discussions entre les différents partenaires.

De plus en plus, les systèmes à réaliser englobent des activités de l'organisation aussi larges que complexes et faisant intervenir une multitude d'acteurs. Il s'est avéré nécessaire de favoriser des langages, ou plus généralement des outils supportant ou facilitant la communication entre les développeurs et les utilisateurs.

Etant donné l'ampleur sans cesse croissante des systèmes d'information, il n'est pas rare de voir plusieurs équipes de développeurs travaillant sur le même projet. Cette

nouvelle dimension ajoute une complexité supplémentaire dans la gestion du projet. Pour pallier à ce nouveau défi, on a développé des outils ou ateliers logiciels (*CASE Tools*) qui permettent de suivre le projet depuis le début, c'est-à-dire depuis l'étude des besoins jusqu'à la fourniture d'un produit fini. Cette intégration de plusieurs outils permet de garantir une cohérence entre les différentes équipes travaillant chacune sur une des étapes du cycle de vie. On donne ci-après quelques exemples de tels systèmes.

- *Controlled Requirements Expression* (CORE):

Cette méthode prend en considération la notion de besoins émanant de plusieurs points de vue. CORE peut hiérarchiser ces différents points de vue. [Rze93]

- *Rapid Prototyping System* (RPS) :

Il s'agit d'une collection d'outils qui supportent la construction, l'exécution et l'analyse des interfaces utilisateur. On dispose alors d'un prototype qui permet d'examiner les fonctions critiques de manière à améliorer la compréhension des besoins des utilisateurs. [Rze93]

- *Prototyping Environment* (Proto) :

Il s'agit d'un système de prototypage qui consiste en une méthode d'analyse des flux de données (*Data Flow*), un langage de prototypage de haut niveau, des outils intégrés permettant le développement de prototypes et des composants réutilisables. Proto dispose d'outils graphiques permettant à l'utilisateur de créer une représentation du processus du futur système. Pour cela on dispose de composants réutilisables.[Rze93]

1.4.6. Les directions futures

On tend à utiliser de plus en plus des outils basés sur des techniques d'Intelligence Artificielle.

- *Knowledge Based Requirements Assistant* : Dans ce type d'outil, on peut utiliser plusieurs méthodes, ce qui permet de ne pas être limité à une seule catégorie de systèmes. Le spécifieur peut choisir la méthode la plus adaptée au type de système sur lequel il travaille.

- *Specification Assistant (Ex ARIES)* : Dans ce type d'outil, on présume que l'on démarre d'une définition initiale de haut niveau d'un système. La spécification évolue d'une étape à l'autre par le procédé de raffinements (*refinement,* en anglais) successifs. Les spécifications peuvent aussi être formées à partir de l'assemblage de plus petits modules préalablement développés. On dispose d'assistants vérifiant la cohérence et la complétude des spécifications.

1.4.7. Quelques recommandations

Le langage utilisé dans la phase d'étude des besoins, devrait disposer des caractéristiques suivantes : [VDi89]

- **Expressivité** : c'est-à-dire qu'il doit supporter facilement le "*mapping*", et si possible un "*mapping*" naturel, entre toutes les choses devant être décrites dans un cahier des charges (dans le domaine d'application pour être plus général) et les différents concepts disponibles dans le langage.

la spécification d'un système ne se limite pas aux informations et aux types de traitements qu'il faut leur appliquer. Il est également nécessaire de faire apparaître des considérations temporelles et de capacité du système dans la spécification. Le langage devrait donc pouvoir offrir un moyen permettant d'exprimer, par exemple, qu'un traitement n'est acceptable que si il est exécuté dans un certain délai.

- **Mécanismes de structuration**: La définition des besoins des systèmes complexes implique la manipulation d'un masse considérable d'informations. Afin de pouvoir "digérer" ces informations, il convient de disposer de mécanismes de structuration pour organiser la spécification et la maintenir lisible. On peut donc disposer d'une spécification "découpée" en plusieurs morceaux, ceux-ci étant plus "simples", la lecture de la spécification globale, s'en voit simplifiée.

Il est fréquent, lors de l'analyse d'un problème, d'utiliser des spécifications ayant été développées pour d'autres systèmes. Disposer de mécanismes d'abstraction permet donc de réutiliser de parties plus ou moins importantes de la spécification d'une application à une autre.

La spécification complète résulterait de la combinaison de toutes ces (petites) unités de spécification.

Le paradigme "Orienté-Objet" ayant envahi le domaine de la programmation ainsi que celui des spécifications, semble poursuivre son extension à l'ensemble du cycle de vie de développement : il englobe dès lors également la phase d'élaboration du cahier des charges du futur système. L'utilisation d'une méthode dite "orientée-objet" permet d'introduire une structuration plus forte dans l'analyse.

- **Interprétation unique et non ambiguïté** : le but de l'étude des besoins est de fournir une représentation claire et précise du problème du client. Pour ce faire, le langage utilisé ne doit en aucun cas souffrir d'ambiguïté ou d'imprécision. Les langages naturels souffrent d'un manque évident de précision : il est fréquent qu'un même texte soit compris de façon différente selon le lecteur. On doit disposer de règles rigoureuses garantissant l'absence d'ambiguïté. On doit également avoir à sa disposition des règles de déduction permettant de dériver de nouveaux éléments à partir d'éléments donnés. Les langages formels, eux, disposent de règles d'interprétation qui permettent de garantir l'absence d'ambiguïté. Ils permettent souvent l'utilisation d'outils pour supporter la phase d'analyse des besoins, ils permettent de vérifier la complétude, la cohérence. Cette

vérification se base sur des dérivations utilisées pour engendrer des conséquences à partir de la définition des besoins.

- **Intégrité conceptuelle** : un formalisme de spécification des besoins doit s'intégrer dans une méthodologie globale de développement. Ainsi le passage d'une étape à l'autre du développement se ferait plus aisément.

- **Lisibilité et compréhension** : la spécification étant continuellement présentée au client: elle doit être la plus lisible possible. Il est évident que le client ne saurait marquer son accord ou son désaccord sur quelque chose qu' il ne maîtrise pas. Il convient donc de favoriser un langage qui soit aisément compréhensible par toutes les personnes impliquées dans le développement de l'application. Cette lisibilité et cette compréhensibilité seront d'autant plus accrues que le langage utilisé dans une phase du développement le sera aussi dans les autres étapes du cycle de vie. Il convient donc de favoriser l'utilisation d'un langage unique lors des spécifications. On risque de perdre de la sémantique lors des multiples traductions d'un langage vers un autre.

- **Maintenance des spécifications** : Le cycle de vie d'une application ne s'arrête pas à la phase d'étude des besoins: lorsque l'on découvre une erreur, plus tard dans le développement, il convient de corriger les spécifications afin de garantir la cohérence entre les spécifications et le produit final. Cette prise en compte des modifications est grandement facilitée par la modularisation.

- **Absence de surspécification** : une spécification des besoins ne doit pas imposer, ni même privilégier une implémentation plutôt qu'une autre. Elle doit permettre d'envisager une grande variété de solutions. La spécification se cantonne au "Quoi" sans aborder la question du "Comment".

- **Outils** : toutes les phases du cycle de vie de définition des besoins (élicitation, modélisation, analyse, validation) devraient être supportées par un ou plusieurs outil(s) intégré(s) permettant de soulager l'analyste dans des tâches telles que l'élaboration de la documentation et lui permettant de travailler aisément sur la spécification réalisée jusqu'alors.

On a besoin d'une méthodologie pour mener à bien cette étape d'étude des besoins d'un système. Cette méthode doit permettre au développeur de mieux cerner la complexité du système étudié. Celui-ci étant bien souvent composé d'une multitude de sous-systèmes dont il faut également tenir compte lors de la structuration des informations.

Outre le besoin d'une méthodologie à destination du développeur, il convient d'adopter une représentation aisément compréhensible par les futurs utilisateurs. Ceci facilitera l'étape de validation du modèle élaboré lors de cette étude des besoins.

Lors de la phase d'étude des besoins, il convient de réaliser une modélisation du système. Cette modélisation doit rester de haut niveau, c'est-à-dire dans laquelle les services attendus par l'utilisateur s'y retrouvent ainsi que leurs interrelations, sans pour autant aborder la façon dont ils seront implémentés. Cette modélisation doit rester

compréhensible par les utilisateurs, car ce sont eux, qui lors de la validation, avaliseront ou non le modèle proposé.

Le document résultant de l'étude des besoins ne doit pas se contenter de spécifier le système à réaliser en terme de système informatique, il doit également aborder des notions de responsabilités, des notions de performances, etc.De tels systèmes sont appelés *systèmes composites*, car ils font intervenir non seulement des composants informatiques mais également humains, organisationnels etc.

Dans le cahier des charges, chacun des acteurs se voit attribuer des objectifs (*Goals*). La réunion de tous ces objectifs contribue à réaliser l'objectif global, qu'est l'objectif assigné au système (vu comme un tout indissociable).

Zeh et Zave présentent le cahier des charges comme étant "*un ensemble de propriétés précisément établies ou contraintes que le système doit satisfaire*". Ce document n'aborde nullement la question de l'implémentation ou plus généralement la façon dont on va résoudre le problème (*How*) mais il se cantonne à la structuration du problème (*What*). Le cahier des charges doit présenter de façon claire et univoque ce que le futur système doit faire.

A l'issue de cette étape, on dispose d'un document dans lequel le problème à résoudre est défini avec précision, complètement et de façon cohérente. Ce document constitue le point de départ de la conception (*design*) proprement dite du système. Il est souvent comparé à un contrat liant les développeurs aux clients.

Yeh précise l'utilité d'un document définissant les besoins qui serait le plus complet et précis possible : " *Le document de requirements d'un système établit les frontières de l'espace de solutions du problème de développement d'un software utile*".

Une fois, la phase d'élicitation des besoins terminée, il convient de valider le document. Cette validation est nécessaire pour mettre en évidence les erreurs afin d'éviter que celles-ci ne se propagent tout au long du développement.

Il est donc important de suivre une certaine méthodologie pour réaliser ce document.

1.4.8. Evolution et tendances actuelles

La plupart des recherches récentes dans l'Ingénierie des Besoins ont fait glisser les spécifications de systèmes spécifiques vers la *spécification de domaines d'applications*.

Les systèmes informatiques sont rarement conçus de manière isolée. En fait, la plupart des systèmes représentent une nouvelle instance d'un produit dans une famille de systèmes reliés.

Chaque système partage typiquement, avec les autres applications de sa famille, une structure commune de comportement.

Un des objectifs principaux est de permettre une plus grande **réutilisation** des spécifications de systèmes déjà maîtrisés. Si l'on peut produire des spécifications de besoins pour un domaine en général, cela peut augmenter considérablement notre capacité à produire des spécifications de besoins pour une instance particulière.

David Barstow définit un domaine comme étant une "*collection de concepts, d'attributs, de relations qui organisent les entités en ensemble cohérent de phénomènes naturels ou informatisés.*" [Bar93].

Pour rester cohérent avec la définition que nous avons donnée d'un système d'information, il nous parait intéressant de compléter également cette définition du domaine. La définition que nous en donnerions serait : *le domaine est une collection de concepts (d'informations, de traitements, d'acteurs), de règles d'organisation régissant la collaboration des acteurs (machines, systèmes informatiques, personnes). Le domaine doit prendre en compte les comportements automatisés, interactifs ainsi que les traitements manuels.*

Disposer d'une spécification de domaine permet de vérifier que le système spécifique ne viole pas les règles de comportement du domaine. Cela permettrait également de guider le spécifieur dans la phase d'analyse des besoins.

Le problème se situe dans l'identification des frontières du domaine d'application. Il est clair que la spécification d'un domaine peut seulement être construite à partir d'une expérience acquise dans la spécification d'instances de systèmes similaires. C'est seulement avec la pratique dans le domaine, que l'on est en mesure de détecter, d'indentifier les structures abstraites significatives. Dans la cas contraire, on risque d'identifier de mauvaises structures abstraites.

Bien que l'étude des besoins soit inévitablement plus abstraite que la conception (*design*), cela rend la **généralisation** et la **réutilisation** plus facile et plus efficace. Le danger est d'arriver à des généralisations qui soient trop générales et donc ne pourraient pas être utilisées.

Nous pouvons exploiter ce fait, en spécifiant des domaines d'application plutôt que des systèmes spécifiques. Cela permettrait de

- disposer d'une uniformité dans les produits de la même famille (domaine).
- réduire les coûts en répartissant les efforts de développement sur l'ensemble des systèmes de la famille.
- réutiliser certains composants d'un système à l'autre.
- disposer de structures d'abstraction.
- comprendre plus facilement un nouveau système lorsqu'il utilise des concepts communs à l'ensemble des systèmes de la famille.
- d'éviter de recommencer maintes et maintes fois un développement similaire en reprenant chaque fois depuis le début.

Nous croyons que l'existence d'un domaine rend le développement, la personnalisation et l'évolution d'un système plus simple. Le domaine fournit des **bases**

naturelles pour l'organisation du système et la spécification de ses composants. La personnalisation et l'évolution d'un système sont directement basées sur le domaine.

Un modèle du domaine est donc un investissement qui fournit une plus grande efficacité dans les activités futures.

Chapitre 2

ALBERT, un langage de spécification des besoins

2.1. Introduction

Nous avons vu au chapitre précédent combien l'étape de l'étude des besoins constituait une phase importante, voire cruciale, dans le cycle de vie d'un système d'information.

Dans ce chapitre, nous présentons un langage de spécification des besoins qui tient compte des recommandations faites au chapitre précédent (l'expressivité, les mécanismes de structuration, l'interprétation unique et non ambiguë, l'intégrité conceptuelle, la lisibilité et la compréhension, la maintenance des spécifications, l'absence de sur spécification, l'existence d' outils). Ce langage est particulièrement bien adapté pour la spécification de systèmes composites temps-réel constituant le cadre de ce travail.

La présentation du langage sera basée sur un exemple simplifié à l'aide duquel nous illustrerons la démarche adoptée.

2.2. Description générale d'ALBERT

Le langage présenté dans les pages qui suivent se nomme ALBERT, ce qui signifie *Agent-oriented **L**anguage for **B**uilding and **E**licitating **R**equirements for real-**T**ime systems*. Ce langage a été développé par l'équipe de Mr Eric Dubois, à l'Institut d'Informatique des Facultés Universitaires Notre-Dame de la Paix (Namur, Belgique) dans le cadre d'un projet ESPRIT II nommé ICARUS entièrement dédié à la spécification des besoins.

Dans ce qui suit, nous allons présenter les différents éléments importants d'ALBERT. Comme beaucoup d'autres langages de spécification, ALBERT dispose d'outils graphiques supportant une partie de l'analyse des besoins.

Les spécificités de ce langage sont :

- La possibilité de structurer les besoins (*Requirements*) pour des systèmes composites en termes d'agents (tels que un robot, un système de contrôle, ...). Chaque agent a des responsabilités contractuelles concernant les informations qu'il gère, auxquelles il accède, et qu'il modifie dans l'état d'autres agents.

- L'existence de mécanismes de structuration (basés sur la paramétrisation et l'héritage) assurant une meilleure structuration du cahier des charges (en particulier en définissant un vocabulaire spécifique pour le domaine de l'application envisagée) et permettant la réutilisation de fragments existants.

2.3. Les éléments d'ALBERT

2.3.1. Hiérarchie d'éléments

On peut représenter les différents concepts d'ALBERT selon une hiérarchie. Le système à analyser est constitué d'un ou plusieurs agents. Chacun de ces agents est composé d'entités, d'associations, d'actions et de contraintes. De plus, on peut décomposer les entités et les associations en attributs.

Il est possible d'introduire une telle hiérarchie car cette décomposition se retrouve dans tous les systèmes analysés. On trouvera à la figure suivante une représentation de cette hiérarchie des objets d'ALBERT.

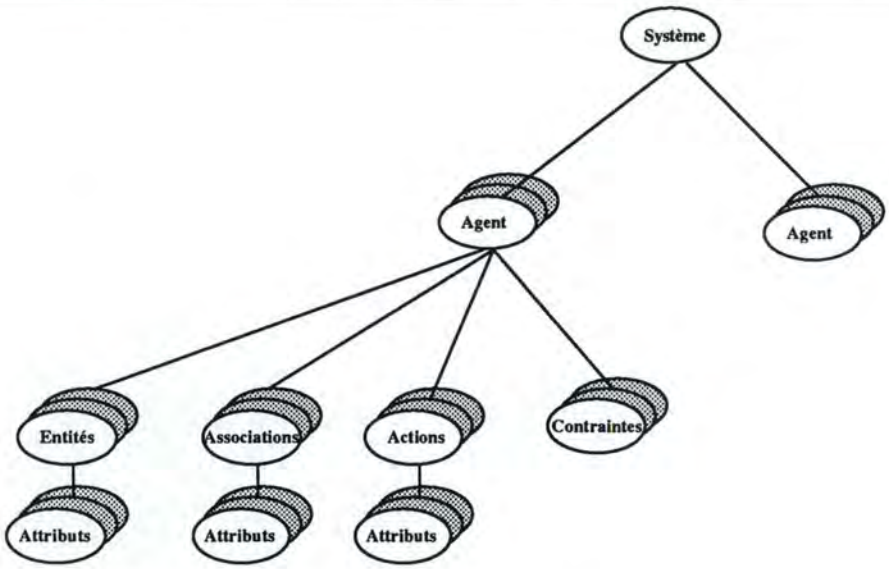


Figure - **2.1** - "Méta-modèle" d'ALBERT

2.3.2. Les types de données

Lors de l'utilisation d'ALBERT, nous pouvons utiliser des types de données. Parmi ceux-ci, nous pouvons distinguer :

- des *types de données pré définis* (tels que INTEGER, STRING, BOOLEAN,...) ainsi que les opérations habituelles définies sur ces types.
- des *types de données complexes* construits par l'analyste en combinant différents connecteurs logiques tels que le produit cartésien, les sets, les listes,... ainsi que les opérations habituellement définies dessus, tel par exemple atteindre le $x^{ième}$ élément d'une liste.
- les *types de données élémentaires* définis par l'utilisateur, par exemple NAME.

Ces types de données sont utilisés pour

- définir le type d'attributs.
- identifier une certaine occurrence parmi l'ensemble des agents du même type.

En fait, en plus de la définition propre de type de données, il y a une caractéristique intrinsèque aux types de données utilisés pour les différents agents. Cette caractéristique est en fait un mécanisme d'identification. Par exemple, si l'on considère que dans une cellule de production, on dispose de plusieurs robots de la même marque, de même capacité,... bref deux robots identiques dans leurs caractéristiques, alors le type de données **ROBOT** permet de différencier, donc d'identifier les deux robots.

2.3.3. L'agent

2.3.3.1. Définition

Ce concept peut être perçu comme étant une spécialisation de la notion d'*objet* dans le sens utilisé dans l'atelier logiciel OBLOG, c'est-à-dire est considéré comme objet tout **processus concurrent et communicant**. [Dub93d]

D'une part, l'agent peut être considéré comme un moyen de structuration de larges spécifications en terme de composants plus fins.

ALBERT permet de spécifier la **collaboration** de plusieurs agents à la réalisation d'un objectif global (celui du système d'information à réaliser). Chaque agent est responsable d'un certain nombre d'informations qu'il utilise, gère pour atteindre l'objectif qui lui a été assigné. Pour représenter cette collaboration, on **hiérarchise** les différents agents composant le système à réaliser. Par exemple, dans la figure qui suit, on peut voir que les agents n° 3 et 4 collaborent pour réaliser les objectifs assignés à l'agent n°2.

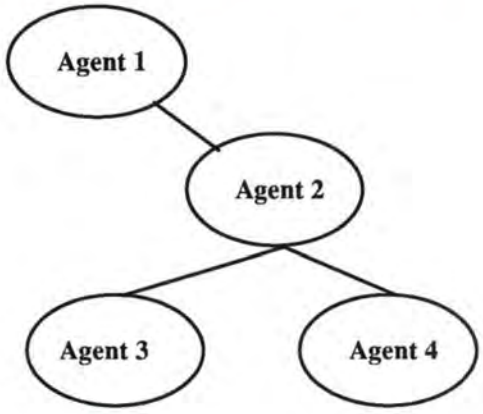


Figure - **2.2** - Représentation de la collaboration de plusieurs agents

D'autre part, les propriétés ainsi que les informations attachées à un agent fournissent des **lignes de conduite** pour l'analyste qui est en charge de concevoir le système composite. Le langage ALBERT rend possible le raisonnement sur les responsabilités de changement et de perception d'informations incombant à l'agent.

2.3.3.2. Représentation graphique de l'agent

L'agent est représenté par un parallélogramme : les informations qui se trouvent à l'intérieur sont sous sa responsabilité. Celles situées à l'extérieur de la forme proviennent d'autres agents. Le nom de l'agent est mentionné au dessus du parallélogramme.

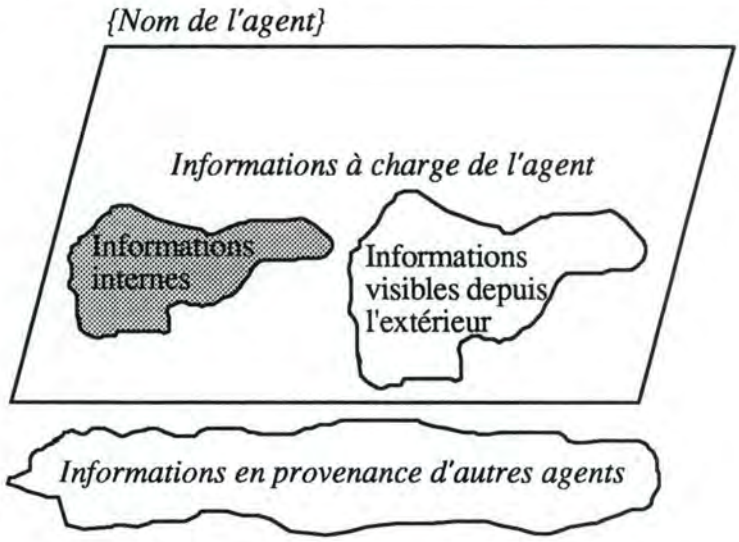


Figure - **2.3** - Représentation graphique d'un agent

Parmi les informations sous la responsabilité de l'agent, on distingue les informations visibles (représentées en clair) depuis l'extérieur, de celles à usage interne uniquement (représentées en grisé).

On peut également distinguer deux types d'agents : les agents simples ou indécomposables et les agents complexes :

- Les agents "*simples*" ou *indécomposables* : Ils correspondent aux feuilles de l'arbre. Ces agents sont des composants terminaux pour lesquels l'analyste doit garantir une implémentation valide par rapport à ses responsabilités.

- Les agents "*complexes*" : ils correspondent à des noeuds non terminaux de la hiérarchie. Ils sont composés d'autres agents plus simples. Ces agents sont des composants *virtuels* parce qu'ils n'ont pas d'existence en soi et ils sont uniquement des agrégats d'agents plus simples.

Par exemple, dans la figure suivante, les agents complexes sont représentés par des formes grisées alors que les agents indécomposables sont représentés par des formes claires. Dans la suite de ce travail, nous appliquerons cette représentation lorsque nous serons en présence d'agents complexes et indécomposables.

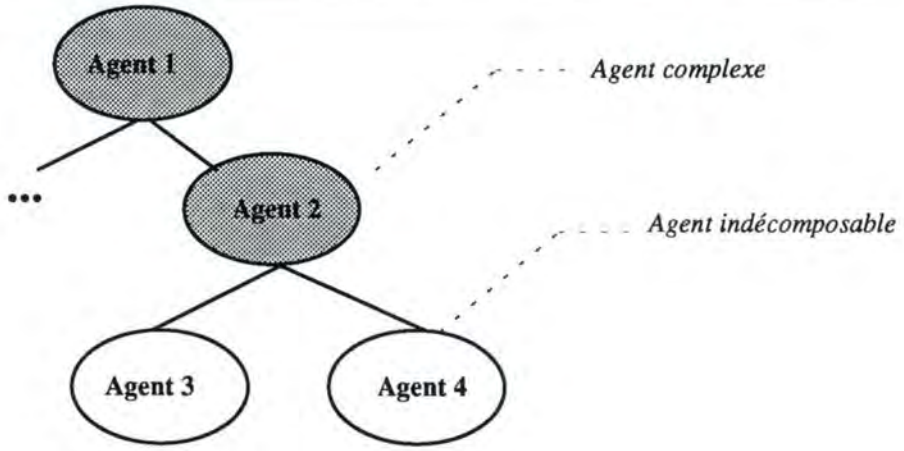


Figure - **2.4** - Agents indécomposables et agents complexes

2.3.4. Les entités

Les informations contenues dans l'agent sont représentées par des types d'entités au même sens que dans des méthodes telles qu' IDA [Bod89] , ERAE [Dub90]...

2.3.4.1. Définition

Une entité est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un agent qui la considère comme un tout, lui confère une existence autonome et la distingue d'autres entités et de son environnement. [Bod89]

2.3.4.2. Catégories d'entités

Nous pouvons distinguer deux grandes catégories d'entités : les *ensembles d'entités* et les *entités isolées*. La distinction porte sur le nombre d'occurrences que l'on peut trouver dans l'agent.

2.3.4.2.1. Ensembles d'entités

En plus de cette classification, il convient "d'éclater" la catégorie Ensemble d'entités en deux sous-groupes : on dispose, en effet d'*ensembles d'entités à population fixe* et d'autres à *population variable.*

- Ensemble d'entités à population variable :

On dira donc qu'un type d'entités sera un Ensemble d'entités à population variable si l'on peut identifier plusieurs valeurs possibles pour ses attributs. Par exemple, si l'on dispose d'un ensemble d'entités "*Batteries_possibles*" de type **BATTERIE** définie comme étant un produit cartésien de Référence (**STRING**) et de Status (**BOOLEAN**). Supposons qu'au sein de cet ensemble on identifie deux occurrences de "*Batteries_possibles*" définies comme étant

{< BATT1, on > , < BATT2, off >}

Si, au fil du temps, les valeurs des attributs (ou le nombre d'occurrences) changent, nous parlerons d'ensemble d'entités à population variable. Dans le cas contraire, nous parlerons d'ensemble d'entités à population fixe (voir description ci-dessous).

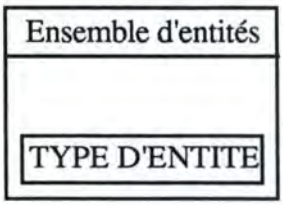


Figure - **2.5** - Un ensemble d'entités à population variable

- Ensemble d'entités à population fixe :

Un ensemble d'entités à population fixe, est un ensemble dont les valeurs des attributs (ainsi que la cardinalité de l'ensemble) n'évoluent pas dans le temps.

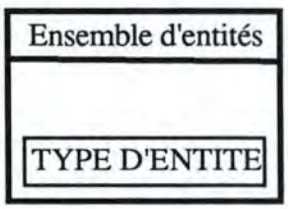


Figure - **2.6** - Un ensemble d'entités à population fixe

2.3.4.2.2. Entités isolées

Un type d'entité isolée correspond à un type d'entité dont on ne trouvera jamais qu'une et une seule occurrence dans l'agent.

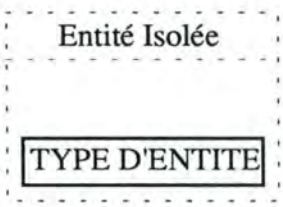


Figure - **2.7** - Une entité isolée

2.3.5. Les actions

2.3.5.1. Définition

Dans ALBERT, la notion d'action regroupe les notions suivantes :

- des événements ayant un effet sur l'état (également appelés actions dans d'autres langages de spécification)

• des événements n'ayant pas d'influence directe sur l'état de l'agent (appelés événements dans certains langages de spécification).

Donc, sont regroupés sous le terme action, les événements qui ont un effet direct ou indirect sur l'état.

Les actions peuvent avoir des arguments, ce qui permet de transférer de l'information d'un agent vers un autre.

2.3.5.2. Représentation graphique des actions

Précédemment nous avons différencié les informations internes à l'agent de celles qui sont visibles à l'extérieur de l'agent. C'est dans la même optique qu'il convient de faire une distinction entre les actions en se basant sur leur origine. Si une action trouve son origine au sein de l'agent, elle sera dénommée, par la suite, *action interne*. Inversement, lorsqu'une action trouve son origine à l'extérieur de l'agent envisagé, elle sera appelée *action externe*.

Une représentation graphique permet de distinguer les *actions internes* des *actions externes*.

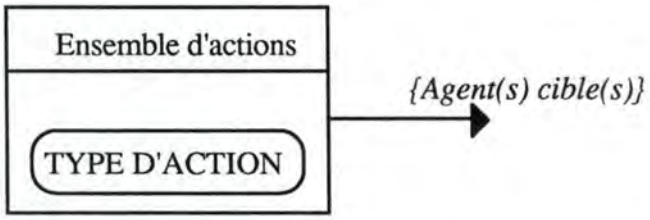


Figure - **2.8** - Un ensemble d'actions émises à l'extérieur de l'agent

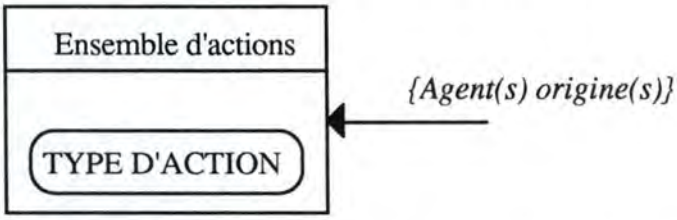


Figure - **2.9** - Un ensemble d'actions reçues de l'extérieur

Les occurrences d'actions ont lieu entre deux états successifs de l'agent. Cela signifie que le changement d'état d'un agent se fait à la suite d'une ou plusieurs actions (interne(s) ou externe(s)).

2.3.6. Les associations

2.3.6.1. Définition

Une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné. On veut enregistrer de l'information relative à cette correspondance. [Bod89]

2.3.6.2. Représentation graphique

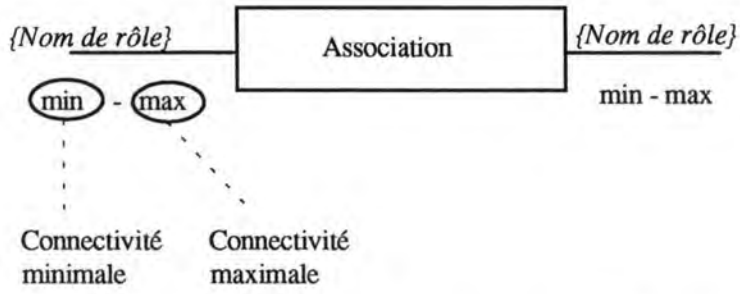


Figure - **2.10** - Une association reliant deux ensembles d'entités.

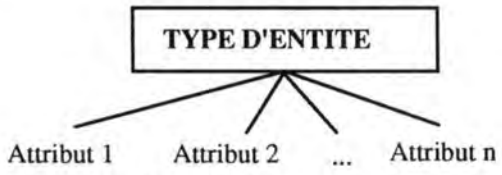
2.3.7. Les attributs

2.3.7.1. Définition

Un attribut est une caractéristique ou qualité d'une entité ou association.[Bod89]

On retrouve les attributs pour les entités ainsi que pour les actions. Dans le cas d'attributs d'action, on parlera alors d'**arguments**.

2.3.7.2. Représentation graphique



TYPE ATTRIBUT 1 TYPE ATTRIBUT 2 TYPE ATTRIBUT n

Figure - **2.11** - Les types d'attributs d'un type d'entité

Il convient cependant de faire une distinction entre les attributs de classe d'entités et les attributs d'une entité.

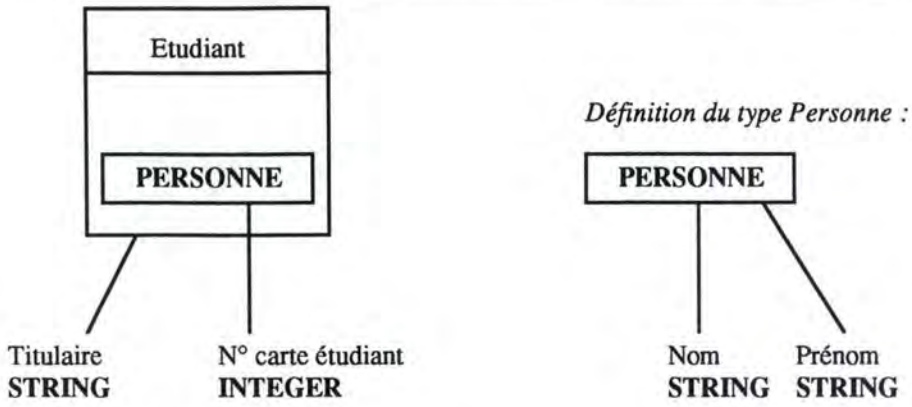


Figure - **2.12** - Les attributs de la classe "PERSONNE" et l'attribut propre aux Etudiants

Dans la figure précédente, on a défini un type de donnée **<PERSONNE>**, qui est composé des attributs *<Nom>* et *<Prénom>*. Il s'agit alors, des caractéristiques communes à toutes les personnes.

On définit ensuite une classe de personnes qui sont des étudiants. Ceux-ci sont des personnes (donc appartenance au type **<PERSONNE>**). Mais en plus des caractéristiques communes aux personnes, ils possèdent un attribut supplémentaire, propre aux personnes étudiantes : numéro de carte étudiant. Cet attribut supplémentaire est lié aux occurrences.

On trouve également un attribut *<Titulaire>*. Ce dernier est **attribut de la classe** Etudiant. Il n'y en a donc qu'un et un seul pour toute la classe.

2.3.8. Les contraintes

Les contraintes sont des énoncés logiques qui identifient les comportements acceptables du système et en excluent certains. L'expression de ces contraintes est purement textuelle.

Chaque agent est caractérisé d'une part par un ensemble d'états possibles définis par son schéma ainsi que par les actions définissant tous ses comportements possibles. La vie de l'agent est décrite comme étant une séquence (in)finie d'états et d'actions.

Les contraintes restreignent le comportement de l'agent à un sous-ensemble de séquences d'états et actions qui définissent les vies acceptables de celui-ci.

2.4. La notion de raffinement (*refinement*)

Lors de l'utilisation d'ALBERT nous allons mettre en pratique la notion de raffinements (*refinement*, en anglais) successifs des besoins sur laquelle se base l'analyse des besoins.

C'est-à-dire que dès que l'on a identifié un agent ainsi que ses responsabilités, on suppose qu'il dispose de tous les moyens nécessaires pour atteindre l'objectif qui lui a été assigné.

Toutefois, par la suite (toujours dans l'étape de l'analyse) on va mettre à jour, chez cet agent, certaines difficultés à réaliser son objectif : par exemple, il s'avère que pour réaliser un certain traitement, il ne dispose pas de l'information nécessaire. L'agent ainsi représenté est contractuellement non implémentable. Il va donc falloir affiner la représentation que l'on a de l'agent pour rendre cette information disponible ou du moins accessible.

Par la suite, des contraintes portant sur les performances vont venir remettre en question la représentation que l'on a de l'agent.

Tout le processus de l'analyse des besoins pour un agent est donc guidé par ce "mécanisme" de raffinements successifs. A chaque étape de l'analyse, on affine la représentation que l'on a de l'agent.

Au départ, on dispose donc d'un agent "*idéal*" censé pouvoir atteindre l'objectif assigné sans difficulté, or il s'avère que pour réaliser son objectif l'agent ne dispose pas de toutes les informations nécessaires, de plus on lui impose certaines performances.

A la fin de l'analyse, on sera en possession d'un agent réalisant les objectifs assignés en tenant compte des performances demandées, des informations dont il dispose, ou auxquelles il peut avoir accès. Lors de l'analyse des besoins on va donc "glisser" d'un modèle "*idéal*" vers une situation "*réelle*".

Une façon élégante de mettre en oeuvre le raffinement est d'adopter une convention graphique telle, par exemple, celle proposée par Coad et Yourdon dans leur outil OOA Tool [Coa92]. Le modèle proposé par Coad et Yourdon est un modèle en couches (*layers*) qui se superposent l'une sur l'autre pour obtenir, à la fin du processus, une spécification complète du système. Nous schématisons ci-dessous, une représentation similaire (*appliquée à un seul agent*) qui, nous semble-il, s'adapte bien à ALBERT.

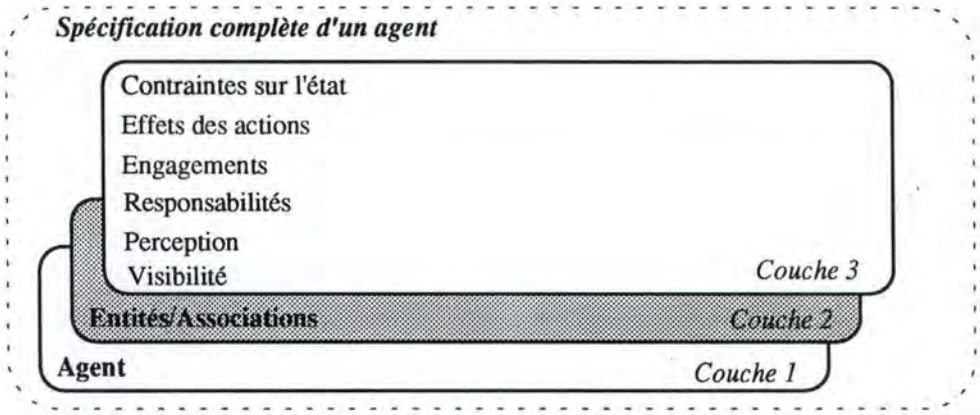


Figure - **2.13** - Représentation graphique de la mise en oeuvre du "*Refinement*" pour un agent

Actuellement, un outil graphique supportant cette phase de l'analyse des besoins est en cours de réalisation et met en oeuvre un tel mécanisme de visibilité.

Maintenant que nous venons de présenter les concepts véhiculés par ALBERT, nous allons proposer, dans la section suivante, la démarche à suivre lors de l'utilisation de ce langage de spécification.

2.5. L'utilisation d'ALBERT

L'utilisation du langage de spécification ALBERT se fait en deux grandes étapes qui sont la *structuration du domaine* et l'*expression des contraintes*.

La structuration du domaine consiste à introduire le vocabulaire qui sera utilisé dans la suite de l'analyse. Pour cette étape, nous disposons d'un support graphique.

L'expression des contraintes consiste à construire des énoncés logiques qui identifient les comportements possibles du système. Pour cette étape, on ne dispose pas (encore) de support graphique.

Dans ce qui suit nous allons présenter la démarche de spécification sur base d'un petit exemple issu de [Dub93a].

2.5.1. Enoncé de l'exemple

L'exemple est basé sur un cas réel, cependant pour les propos qui nous intéressent ici, nous avons simplifié quelque peu la situation. Nous renvoyons le lecteur au chapitre 4 de ce travail où il trouvera la spécification d'un atelier flexible complet.

On peut représenter la situation de l'atelier flexible comme suit :

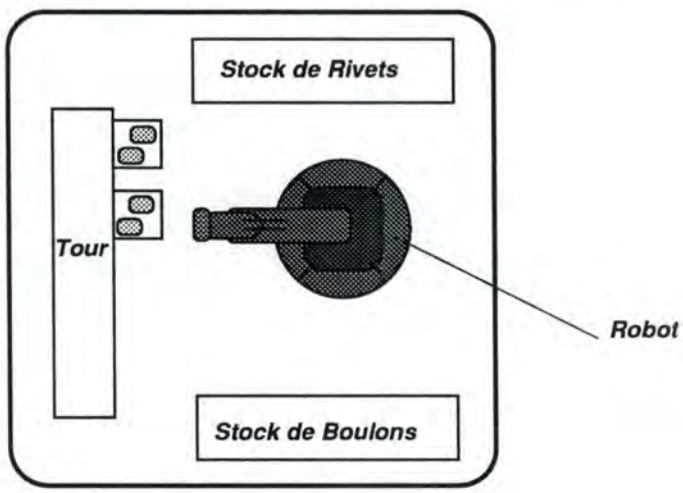


Figure - **2.14** - L'atelier flexible

La cellule est responsable de la production de boulons lorsqu'une commande arrive. Un boulon résulte de la transformation d'un rivet. Le boulon doit être obtenu au plus tard 10 minutes après la réception de la commande.

La cellule est composée de :

- Un *stock de rivets* : Les rivets sont produits par une autre cellule et sont placés dans ce stock en attendant d'être traités.
- Un *stock de boulons* : Les boulons sont le résultat de la transformation des rivets par la cellule et sont stockés à cet emplacement.
- Le *tour* : Cette machine transforme le rivet en boulon en produisant le filet du boulon.
- Le *robot équipé d'une main* . Son rôle est double ; d'une part, il transporte le rivet depuis son stock vers le tour, ensuite, il transporte le boulon depuis le tour (qui vient de le produire) vers le stock de boulons.

Maintenant que le cadre de cet exemple est fixé, nous allons identifier les différents agents composants ce système (la cellule de production).

2.5.2. La déclaration

Cette première phase consiste à identifier les différents agents composant le système ainsi que leur structure d'état.

2.5.2.1. Hiérarchie d'agents

La spécification du système global est faite de la spécification de plusieurs agents.

On représente à la figure suivante la hiérarchie des agents correspondant à notre exemple :

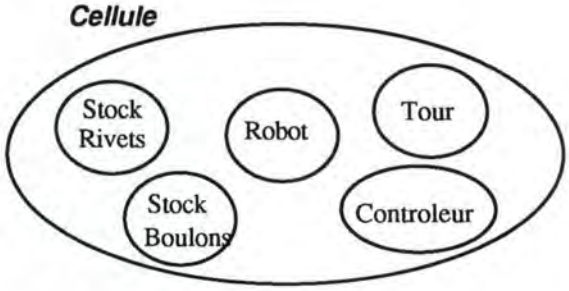


Figure - **2.15** - La hiérarchie des agents de la cellule de production

Dans la figure précédente, l'agent représenté en grisé constitue ce que l'on appelle un agent complexe, alors que les agents représentés par des zones claires correspondent aux agents simples ou indécomposables.

L'attrait d'une telle distinction entre agents "*simples*" et "*complexes*" permet d'introduire un niveau d'abstraction plus fort dans la spécification du système. En effet, au départ, on dispose d'un agent complexe (le système tout entier, ici la cellule), on est en possession d'informations concernant les responsabilités du système. Dans notre cas, nous sommes en mesure de savoir que la cellule doit être capable de produire des boulons à partir de rivets et ce sur demande.

A ce moment de la spécification, nous ne désirons pas (encore) connaître la façon dont la cellule transforme les rivets en boulons. Nous savons juste que la cellule produit des boulons à partir de rivets, comment le fait-elle ?, cela ne nous regarde pas pour l'instant. On suppose toutefois que la cellule met tout en oeuvre (en son sein) pour mener à bien la tâche qui lui incombe. En fait, on pourrait comparer (à ce moment) la cellule de production comme étant une *boîte noire*, dont on ne connaît que ces composants.

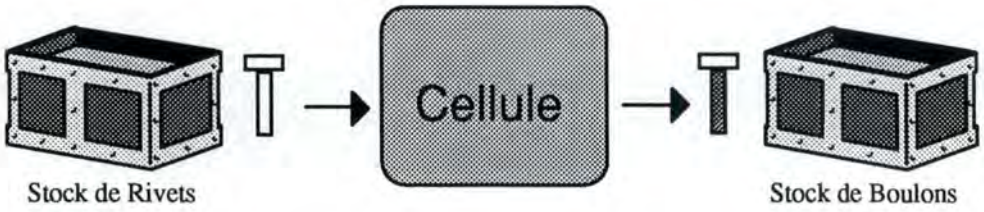


Figure - **2.16** - La cellule comme une boîte noire

Si maintenant on "ouvre" la boîte noire, on trouve les agents composant la cellule. Les agents *Stock de rivets*, *Stock de boulons*, *Tour*, *Robot* et *Contrôleur* sont des agents "simples" ou *terminaux* qui coopèrent pour fournir une solution au problème de la cellule (qui est de produire des boulons à partir de rivets).

On remarque cependant que dans le modèle de la boîte noire, nous présentons le stock de rivets et le stock de boulons bien que ceux-ci fassent partie intégrante de la cellule. C'est en effet, là, qu'ils sont localisés comme entrées/sorties de la cellule.

C'est spécialement le cas pour le *Contrôleur* qui est en charge d'assurer la synchronisation de tout le processus de production de la cellule. Les agents *Stock de rivets* et *Stock de boulons* ont un rôle plus passif qui consiste dans le maintien du stock respectivement de rivets et de boulons.

L'expression de la hiérarchie peut se faire soit de manière graphique, comme nous venons de le voir, mais également sous une forme textuelle. Cette forme textuelle est exprimée à l'aide de constructeurs qui sont le produit cartésien (CP)et les ensembles (SET).

Dans notre cas nous avons donc :

Cellule = CP [Tour ;
Stock de rivets;
Stock de boulons;
Robot;
Contrôleur]

2.5.2.2. Structure d'état et actions

Nous allons maintenant appliquer ce que nous venons de voir au cas nous concernant. On se limitera cependant ici à donner les spécifications de l'agent cellule

uniquement. Pour plus de détails nous renvoyons le lecteur à la partie concernant la spécification de l'atelier flexible complet.

2.5.2.2.1. Etat de l'agent Cellule

On a identifié l'agent *Cellule* (dans la définition de la hiérarchie des agents). Il convient donc de le représenter selon le formalisme présenté précédemment. A ce stade de la spécification, nous ne connaissons rien de cette cellule. La façon dont elle réalise son objectif nous est inconnu. Nous savons juste qu'elle est en mesure de produire des boulons à partir de rivets.

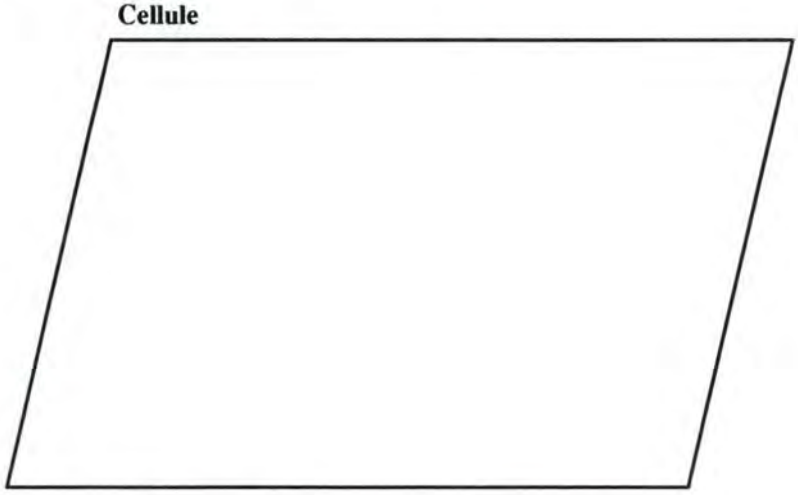


Figure - **2.17** - Etape n° 1 : L'agent Cellule

La deuxième étape consiste à identifier les informations contenues dans l'agent et dont il est responsable. Ces informations sont structurées comme nous l'avons vu en termes d'entités et d'associations. Dans notre cas (simple), il n'y a pas d'association.

La cellule dispose d'un stock de rivets (Stock d'entrée) ainsi que d'un stock de boulons (Stock de sortie).

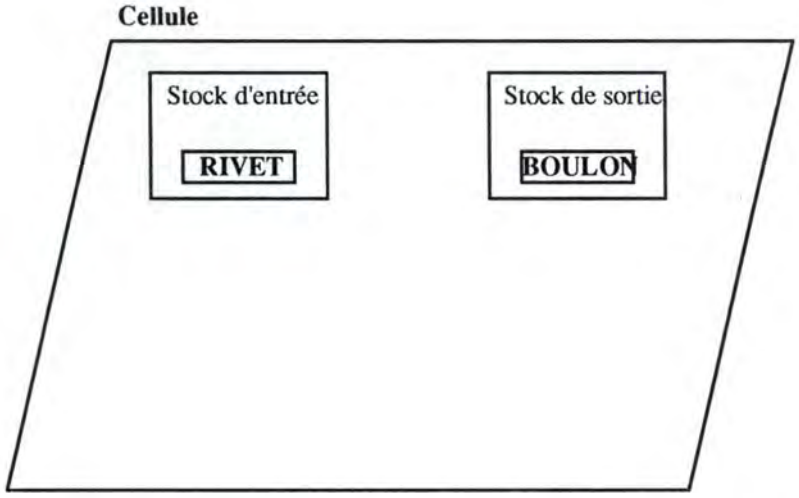


Figure - **2.18** - Etape n°2 : L'agent Cellule et les informations lui appartenant

Ensuite, il convient de déterminer les actions auxquelles l'agent doit être en mesure de "répondre". On identifie donc les actions internes et externes en mentionnant la provenance (pour les actions externes) et la destination éventuelle (pour les actions internes).

Dans la cellule, nous disposons d'une action "*Retrait_rivet*" permettant de signaler aux composants de la cellule que l'on effectue un retrait dans le stock de rivet. Cette action est uniquement destinée à un usage interne. On trouve également une action "*Stocke_boulon*" qui indique au contrôleur que l'on vient de stocker un nouveau boulon dans le stock correspondant. On trouve une action "*Stocke_Rivet*" qui avertit la cellule qu'un rivet vient d'entrer dans le stock de rivets. L'action "*Produire*" déclenche la début de la production au sein de la cellule. La cellule est avertie d'un retrait de boulon par l'action "*Retrait_boulon*".

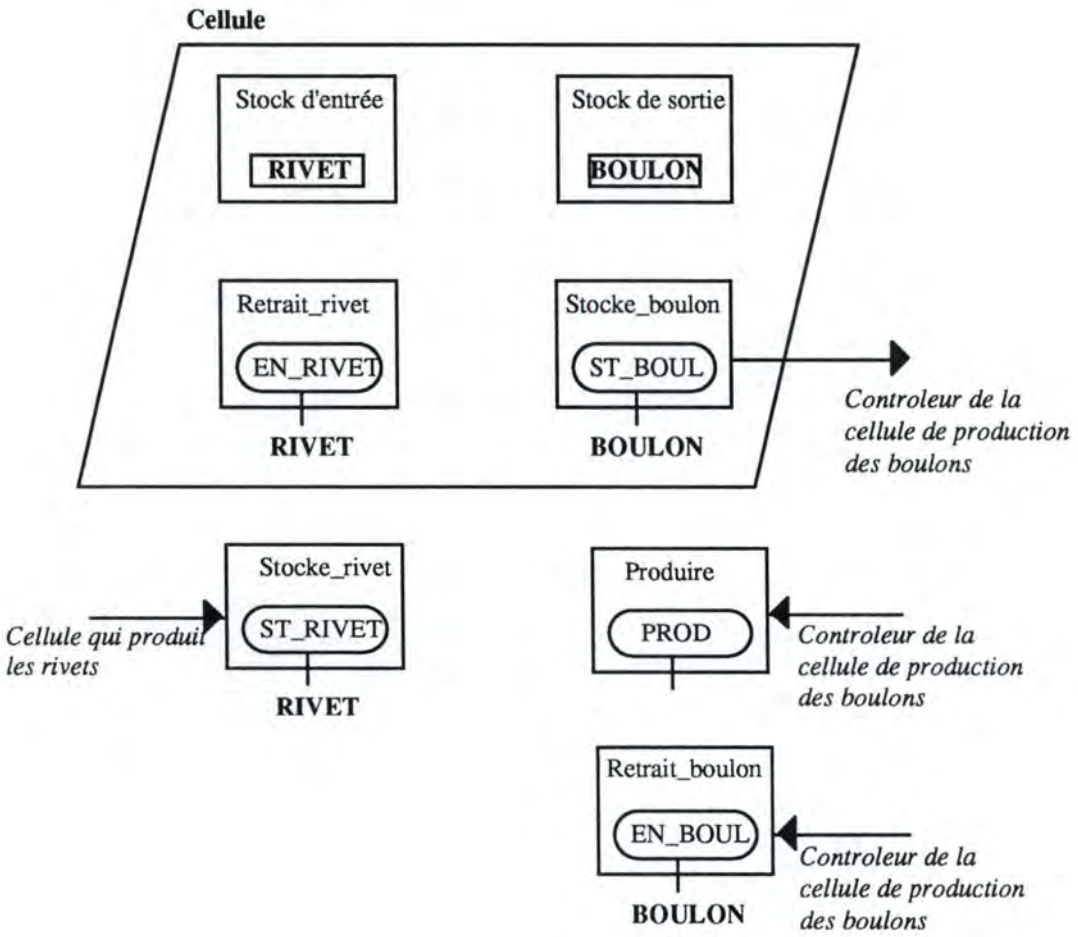


Figure - **2.19** - Etape n° 3: L'agent Cellule et les actions le concernant

Nous avons également vu que l'agent ne rendait visible que certaines de ses informations. Il est donc nécessaire de différencier les informations pouvant être perçues, par les autres agents, des informations purement internes à l'agent considéré (ici la cellule). Pour cela, on représente les informations internes en grisé alors que les

informations visibles depuis l'extérieur sont en clair. Il faut également mentionner le(s) agent(s) qui perçoit (perçoivent) cette information.

Le fait que la cellule produit les boulons à partir de rivets n'intéresse que la cellule. L'objectif assigné à la cellule est la production de boulons, elle doit donc mettre tout en oeuvre pour réaliser cet objectif mais l'extérieur n'est pas intéressé par la façon dont elle réalise cette production. Cela signifie que le Stock d'entrée n'est pas connu de l'extérieur, et donc est représenté en grisé.

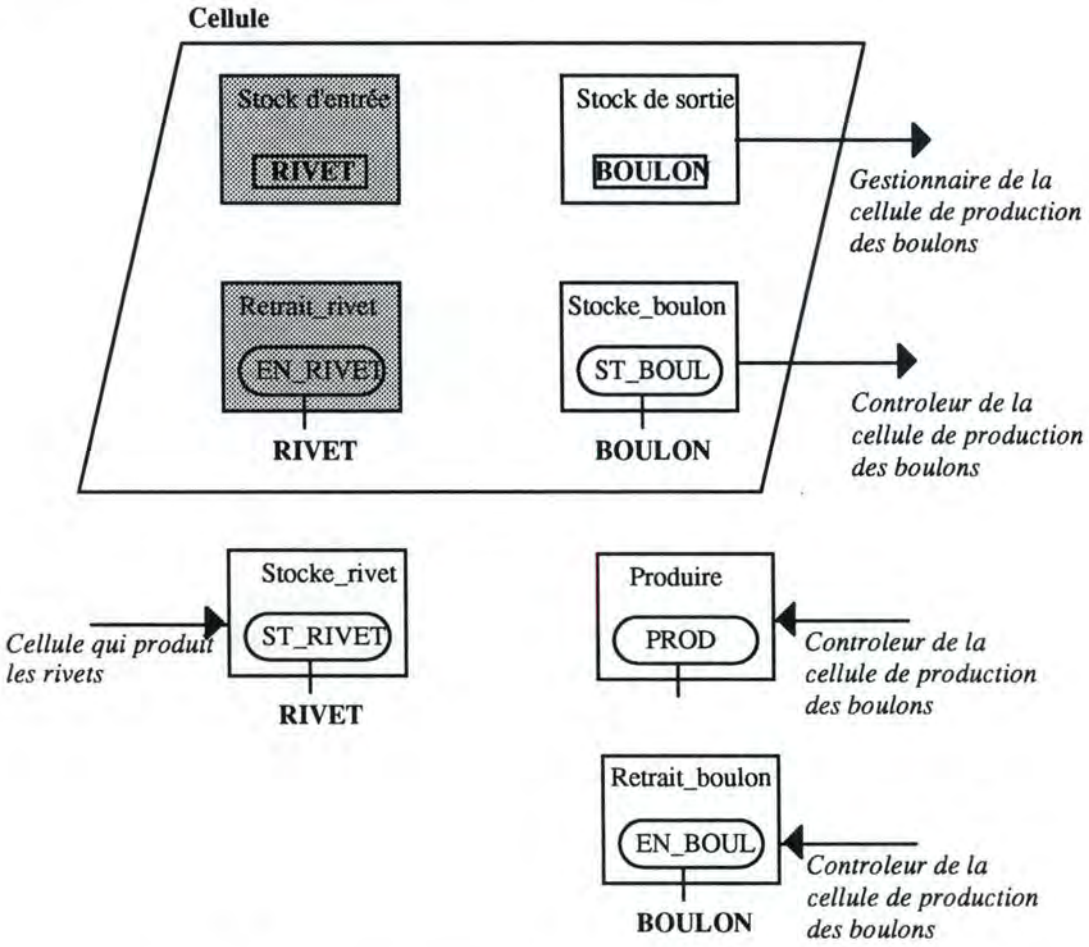


Figure - **2.20** - Etape n° 4 : L'agent Cellule et les informations internes et externes

2.5.3. La définition des contraintes

Chaque agent est défini par un ensemble d'états possibles par lesquels il peut passer lors de son existence. L'ensemble de ces changements (actions) permet de décrire, de définir son comportement. La vie d'un agent est une séquence (in)finie d'états et d'actions. Chaque état (structuré en termes d'entités) est "étiqueté" par une valeur temporelle qui croît tout au long de la vie de l'agent. Les actions apparaissent entre deux états successifs et peuvent être simultanées.

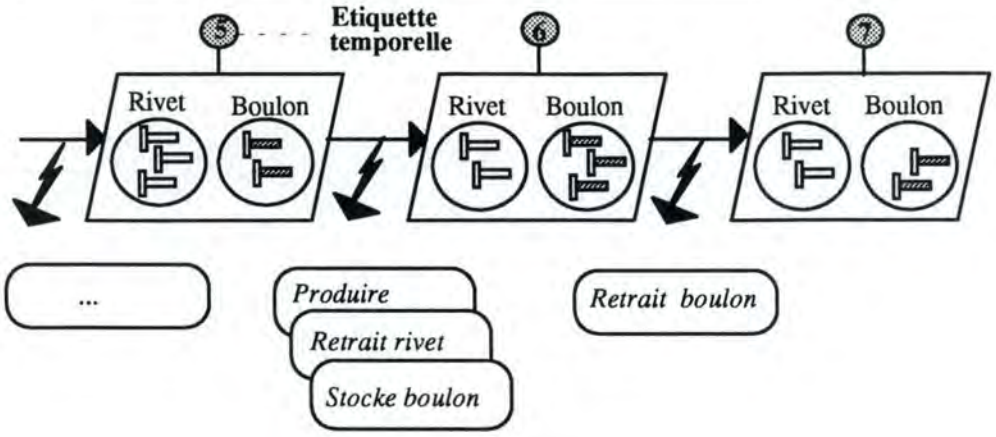


Figure - **2.21** - Les états possibles de l'agent Cellule

Supposons que l'on aboutisse à l'état numéroté 5 suite à une ou plusieurs actions quelconques. On passe de l'état "étiqueté" 5 à l'état numéro 6 suite à l'action *Retrait_rivet* (provenant d'un état précédent quelconque), suite également à l'action *Stocke_boulon* (issue également d'un état précédent). L'effet de l'action *Produire* sera constaté dans un état futur. Le passage de l'état numéroté 6 à l'état 7 est du à l'action *Retrait_boulon*, qui peut être vue comme la dernière étape d'une commande de production apparue lors d'un état précédent.

Les contraintes sont utilisées pour "élaguer" l'ensemble des vies possibles. L'expression formelle des contraintes est basée sur une logique des prédicats du premier ordre, faisant intervenir les concepts de variables, de prédicats, de fonctions. Pour pallier à la complexité de l'écriture de formules du premier ordre, les contraintes ont été regroupées sous différents titres et quelques connecteurs spécifiques ont été introduits pour les exprimer.

Contraintes sur l'état (*State behaviour*)	
Effets des actions (*Effects of action*)	
Engagements (*Commitments*)	
Responsabilités (*Responsabilities*)	
Perception (*Perception*)	
Publicité (*Publicity*)	

Tableau - **2.1** - Structure du document concernant les contraintes

Les configurations possibles des états sont restreintes par les contraintes. Ces contraintes sont rédigées en accord avec les règles usuelles de la logique des prédicats du premier ordre. En particulier, elles sont formées par des connecteurs logiques tels que $\neg$ (not), $\wedge$ (et), $\vee$ (ou), $\Rightarrow$ (implique), $\Leftrightarrow$ (si et seulement si), $\forall$ (pour tout), $\exists$ (il existe).

Les quantificateurs universels peuvent être omis : toute variable qui n'est pas expressément sous la portée d'un quantificateur est universellement quantifiée (implicitement) à l'extérieur de la formule.

En plus des contraintes qui sont vérifiées à tout moment de la vie de l'agent (ce que l'on appelle couramment les *invariants*), il y a des contraintes concernant les changements d'états. Ces dernières contraintes conditionnent les changements valides d'états. Elles sont élaborées en ajoutant des composantes temporelles au langage. Ces connecteurs sont inspirés de la logique temporelle. On trouvera dans la table suivante la signification de ces nouveaux connecteurs :

Expression	Signification
$\Diamond\ \phi$	ϕ est vrai dans l'état actuel ou sera vrai dans un état futur, quel qu'il soit
$\blacklozenge\phi$	ϕ était vrai dans l'état actuel ou était vrai dans un état passé (quel qu'il soit)
$\Box\phi$	ϕ est vrai dans l'état actuel et le restera dans tous les états futurs
$\blacksquare\ \phi$	ϕ était vrai dans tous les états passés y compris l'état actuel.
$\phi\ U\ \psi$	ϕ est vrai dans l'état actuel et sera vrai dans les états futurs jusqu'au (Until) prochain état lors duquel ψ sera vrai (pour toujours si ψ n'est jamais vrai)
$\phi\ S\ \psi$	ϕ est vrai dans l'état actuel et depuis (Since) le dernier état passé dans lequel ψ était vrai (depuis toujours si ψ n'a jamais été vrai)
$\bigcirc\ \phi$	ϕ sera vrai dans l'état futur
$\bullet\ \phi$	ϕ était vrai dans l'état précédent

Tableau - **2.2** - : Les connecteurs temporels

Ces contraintes s'avèrent nécessaires pour l'expression de propriétés dites "*temps-réel*". Elles permettent de décrire des délais, ou des temps d'attente en indiçant les connecteurs temporels par une période de temps. Cette période de temps peut être précisée en utilisant les fonctions pré définies sur base des unités temporelles habituelles : secondes, minutes, heures, jours, ... Par exemple on pourrait avoir :

$\Diamond$=3 min ϕ : ce qui signifie que ϕ sera vrai dans exactement 3 minutes.
$\Diamond$>5 sec ϕ : ce qui signifie que ϕ sera vrai dans un état futur éloigné d'au moins 5 secondes de l'état courant.

Il est également possible d'exprimer le fait qu'une ou plusieurs actions peuvent survenir au même instant avec certaines caractéristiques de succession, de parallélisme ou d'exclusion. Pour cela, trois notations sont introduites :

Expression	Signification
action_1 ; action_2	L'occurrence de l'action_1 est suivie par l'action_2
action_1 // action_2	L'action_1 et l'action_2 se succèdent dans un ordre quelconque
action_1 ⊕ action_2	L'action_1 apparaît ou bien l'action_2 apparaît (ou exclusif)

Tableau - **2.3** - : Le séquencement des actions.

On définit également une notation particulière permettant d'exprimer des contraintes dites de performances, telles par exemple, le fait que la cellule de production s'engage à produire un boulon endéans les 10 minutes qui suivent la réception de la commande.

$$\text{Envt.Produire} \rightarrow^{\Diamond \leq 10\,\text{min}} \text{Retrait\_rivet (r) ; Stocke\_boulon (b)}$$

Le symbole "→" exprime une relation d'implication différente de celle rencontrée dans la logique modale.

Pour exprimer des permissions, obligations ou interdictions de réactions à certaines actions nous utilisons une extension à la logique des prédicats du premier ordre et à la logique temporelle en introduisant des contraintes appelées "**déontiques**".

- Sous le vocable Obligation (*Obligation*) on exprime qu'une action (<action>) apparaît si et seulement si les conditions requises (<situation>) sont présentes.

O (<action> / <situation>)

- Sous le terme Interdiction (*Forbidden*) nous reprenons l'ensemble des actions (<action>) qui ne peuvent en aucun cas être déclenchées lorsqu'une situation donnée (<situation>) se présente.

F (<action> / <situation>)

- Enfin, les obligations exclusives (*Exclude*) regroupent l'ensemble des actions (<action>) qui ne peuvent être déclenchées lorsque l'on se trouve dans une situation donnée (<situation>) , alors que cette action est obligatoirement déclenchée si cette même situation ne se présente pas.

E (<action> / <situation>)

Nous allons maintenons exprimer les contraintes pour l'exemple de la cellule de production de boulons. Pour cela, nous progresserons par étapes : nous allons mettre en évidence les contraintes concernant l'état de l'agent, ensuite celles qui déterminent les effets des actions, puis les engagements de l'agent, et enfin les contraintes qui concernent

les responsabilités de l'agent, la perception que l'agent a de l'extérieur, la publicité que fait l'agent concernant ses informations.

2.5.3.1. Les contraintes sur l'état de l'agent

Contraintes sur l'état	• *Le stock d'entrée ne peut pas être vide* ¬ Empty(Stock_entrée) • *Si le rivet se trouve dans le stock d'entrée (actuellement) alors dans un état futur il n'y sera plus* In(Stock_entrée, rivet) ⇒ ◊ ¬ In(Stock_entrée, rivet)

Tableau - **2.4** - Les contraintes concernant le comportement de l'agent.

2.5.3.2. Effets des actions

Ces contraintes associent l'occurrence d'une action aux modifications apportées à l'état courant de l'agent. Seules les actions apportant un changement effectif sont décrites. Dans la description de l'effet d'une action, nous utilisons une règle implicite de construction : seule la partie de l'état changeant d'un agent est décrite.

L'effet d'une action est exprimé en terme de propriétés caractérisant l'état qui suit l'apparition de l'action.

Effets des actions	• *Retrait_rivet signifie que le rivet (r) est retiré du stock d'entrée.* Retrait_rivet(r) : Stock_entrée = Remove[1](Stock_entrée, r) • *Stocke_boulon signifie que le boulon (b) est ajouté dans le stock de sortie.* Stocke_boulon (b) : Stock_sortie = Add [2](Stock_sortie,b) • *Lorsque l'extérieur produit une action de retrait d'un boulon, on retire le boulon du stock de sortie* *Envt*[3].Retrait_boulon (b) : Stock_sortie = Remove(Stock_sortie, b) • *Lorsque la cellule reçoit un rivet, celui-ci est stocké dans le stock d'entrée* *Envt*.Stocke_rivet(r) : Stock_entrée = Add(Stock_entrée ,r)

Tableau - **2.5** - Les effets des actions de l'agent Cellule

2.5.3.3. Les engagements de l'agent

La partie droite d'un engagement (*commitment*) peut seulement faire apparaître des actions générées au sein même de l'agent considéré.

Engagements	• *La cellule s'engage à produire un boulon à partir d'un rivet. Le délai de production est d'au plus 10 minutes. L'action Produire est suivie uniquement de l'action Retrait_rivet qui elle même est suivie de l'action Stocke_boulon.* *Envt*.Produire : $\rightarrow^{\Diamond \leq 10\ min}$ Retrait_rivet(r) ; Stocke_boulon(b)

Tableau - **2.6** - Les engagements de l'agent.

[1] Remove est un mot réservé du langage ALBERT. L'instruction **Remove (SET,occ)** permet de supprimer une occurrence (occ) de l'ensemble (SET)

[2] Add est un mot réservé du langage ALBERT. L'instruction **Add(SET,occ)** permet d'ajouter une occurrence (occ) à l'ensemble (SET).

[3] Envt. est une abréviation pour Environnement. Toute action provenant de l'extérieur de l'agent doit être préfixée par le nom de l'agent qui à généré cette action.

2.5.3.4. Les responsabilités des agents

Sous cet en-tête, nous précisons le rôle que l'agent doit jouer suite à l'apparition des actions internes. Cependant, dans l'exemple qui nous occupe, il n'y a pas de contraintes concernant les responsabilités de l'agent.

2.5.3.5. Les contraintes de Perception

Perception	• *La cellule n'autorise pas un agent extérieur (Envt.) à venir prendre un boulon lorsque le Stock_sortie est vide* Forbidden(*Envt.*Retrait_boulon) / Empty (Stock_sortie) • *La cellule n'accepte pas d'ordre de production de boulon lorsque le Stock_entrée est vide. Cependant, elle est obligée de répondre à cet ordre de production lorsque le Stock_entrée n'est pas vide.* Forbidden(*Envt.*Produire) / Empty (Stock_entrée) and Obligation(*Envt.*Produire) / ¬ Empty (Stock_entrée)

Tableau - **2.7** - Les contraintes de perception de l'agent (version 1) .

2.5.3.6. Les contraintes de Publicité des informations de l'agent

Les contraintes de publicité permettent d'exprimer comment (et sous quelles conditions) une information (appartenant à l'agent en question) est rendue disponible à un ou plusieurs autres agents.

Publicité	• *L'environnement n'a pas accès (n'a pas connaissance) au Stock_sortie si ce dernier est vide, cependant l'environnement a accès à cette information lorsque le Stock_sortie n'est pas vide* Forbidden(Stock_sortie.*Envt*) / Empty (Stock_sortie) and Obligation(Stock_sortie.*Envt*) / ¬ Empty (Stock_sortie)

Tableau - **2.8** - Les contraintes concernant la publicité des informations de l'agent (version 1)

L'expression des contraintes de responsabilités, de perception et de publicité est parfois assez lourde à rédiger, par exemple, lors des obligations exclusives (par exemple, on doit répondre à l'ordre de production si le stock est plein, mais on ne peut pas répondre à cet ordre si le stock est vide). Afin de simplifier quelque peu la rédaction de telle contrainte, nous adoptons une "abréviation" de cette formule.

Une contrainte de la forme :

Forbidden(<*action*> / <*situation*>) **and Obligation** (<*action*> / <¬ *situation*>)

devient

Exlude (<*action*> / <*situation*>)

On trouvera dans les deux tableaux suivants les contraintes de perception et de publicité exprimées avec ce nouveau formalisme.

Perception	• *La cellule n'autorise pas un agent extérieur (Envt.) à venir prendre un boulon lorsque le Stock_sortie est vide* Forbidden(*Envt.*Retrait_boulon) / Empty (Stock_sortie) • *La cellule n'accepte pas d'ordre de production de boulon lorsque le Stock_entrée est vide.* *Cependant, elle est obligée de répondre à cet ordre de production lorsque le Stock_entrée n'est pas vide.* Exlude(*Envt.*Produire) / ¬ Empty (Stock_entrée)

Tableau - **2.9** - Les contraintes de perception de l'agent (version 2)

Publicité	• *L'environnement n'a pas accès (n'a pas connaissance) au Stock_sortie si ce dernier est vide.* *Cependant l'environnement a accès à cette information lorsque le Stock_sortie n'est pas vide* Exclude(Stock_sortie.*Envt* / ¬ Empty (Stock_sortie)

Tableau - **2.10** - Les contraintes de publicité des informations de l'agent (version 2)

2.5.4. Spécifications complètes de l'agent Cellule

Maintenant que nous avons précisé toutes les contraintes portant sur l'agent Cellule, nous disposons d'une spécification complète des besoins pour ce système. Pour rappel, les spécifications des besoins comportent deux volets : le premier met en évidence les composants du système et définit le vocabulaire employé, le second volet restreint le comportement de l'agent considéré en définissant des contraintes.

Chapitre 3

Vers une structure générique des systèmes productiques

3.1. Introduction

Dans le chapitre précédent, nous avons introduit le langage ALBERT et illustré son utilisation sur une petite partie d'une application CIM (*Computer Integrated Manufacturing*). Au delà des spécifications de petits systèmes, le langage peut aussi être utilisé pour décrire des applications CIM plus importantes.

Les entreprises productiques partagent un grand nombre de points communs, tels par exemple, la présence d'un département de conception, un département de planification ainsi que des moyens de production plus ou moins importants selon les cas. Il s'est rapidement avéré intéressant de disposer d'une structure d'entreprise productique qui puisse être très facilement adaptée pour répondre à une situation précise [Ami93], [Sys89], [Val91].

Dans ce chapitre, nous présentons les résultats du travail de Michaël Petit [Pet92] qui propose une structure générique pour les systèmes productiques. Dans le cadre de ce travail, nous ne présentons que la partie déclaration, c'est-à-dire l'identification des différents agents entrant en ligne de compte dans la structure de l'entreprise. Le lecteur intéressé par les structures d'état et les actions correspondant aux différents agents se reportera au travail de Michaël Petit.

3.2. Caractéristiques principales de cette structure générique

Les applications productiques sont reconnues comme étant très complexes car elles font intervenir d'énormes masses d'informations ainsi qu'un grand nombre de composants de nature très différente. La complexité de tels systèmes provient aussi du fait qu'il y a de nombreux niveaux d'abstraction qui peuvent être considérés pour une même information (par exemple, la position du robot est définie en millimètre au sein du système Robot alors que pour la cellule de production une position en centimètre suffit , ...)

Pour gérer une telle complexité, plusieurs auteurs, dont Scheer [Sch88] ont proposé une architecture générale pour les applications productiques. Le principal attrait de telles architectures est de pouvoir servir de "brouillon" ou plus exactement de **grille de lecture** pour l'analyste en charge de l'étude des besoins d'une application productique spécifique. Lors de l'étude de besoins de la nouvelle application, l'analyste doit être en mesure de repérer les composants réutilisables parmi ceux proposés dans la structure générique et ensuite les adapter pour qu'ils représentent au mieux la nouvelle application.

Les composants génériques peuvent constituer ce que l'on appelle couramment une bibliothèque (*library*) d'agents dans laquelle l'analyste peut "puiser" les composants nécessaires et les adapter au cas concret.

3.3. Structure générique des applications productiques

L'architecture productique comprend deux aspects, qui sont la décomposition de l'entreprise en niveaux et ensuite la répartition des fonctions sur ces niveaux.

3.3.1. Notion de niveau d'abstraction

Le problème avec les structures habituellement proposées dans la littérature est que l'entreprise productique fait l'objet d'une décomposition en un nombre fixé de niveaux. Les différentes architectures proposées dans la littérature[1] diffèrent l'une de l'autre par le nombre de ces niveaux ainsi que par leur appellation.

Par exemple, Scheer [Sch88] décrit toute entreprise productique comme étant composée d'un niveau "Entreprise" (*Enterprise*) , "Zone de produit" (*Product area*), "Usine" (*Factory*) , "Groupe d'équipement" (*Equipment group*) , "Equipement" (*Equipment*) et "Composant d'équipement" (*Equipment component*).

[1] Le lecteur intéressé par ces architectures pourra en trouver un aperçu dans [Dou90]

Le principal inconvénient de telles structures est le nombre imposé de niveaux. En effet, il est rare qu'une organisation concrète s'adapte entièrement à l'architecture proposée dans la littérature. Il conviendrait donc de disposer d'un niveau suffisamment général et abstrait que pour pouvoir être instancié un certain nombre de fois à l'organisation concernée suivant sa complexité.

La structure proposée par Michaël Petit [Pet92] définit un tel niveau générique qui peut être instancié un nombre quelconque de fois, selon l'organisation à modéliser.

3.3.2. Notion de fonction

Outre la structuration de l'entreprise en terme de niveaux, il convient de mettre en évidence un certain nombre de fonctions que doivent réaliser ces niveaux. La plupart des architectures proposées précédemment dans la littérature attribuent une ou l'autre fonction (souvent dans son entièreté) à un niveau. Or il s'avère que cette approche ne correspond pas réellement avec la situation des organisations. En effet, la fonction de planification, par exemple, est souvent réalisée sur plusieurs niveaux. Chacun de ces niveaux étant responsable d'une planification plus ou moins fine de ses activités.

La structure générique proposée tient compte de cette répartition des fonctions sur l'ensemble des niveaux de l'architecture. L'idée défendue est que "*...la structure fonctionnelle de l'entreprise (le nombre de niveaux et la répartition des fonctions sur ceux-ci) devrait être définie en fonction de la structure des produits qu'elle fabrique.*"

Certes, une telle architecture ne se rencontrera pas dans la réalité, mais elle constitue une "*base de négociation avec le client concernant l'expression de son problème productique*" [Pet92] . Elle constitue, donc, un point de départ à partir duquel peuvent être dérivées toute une série de structures d'implémentation. Ces dernières seront progressivement adaptées suivant les décisions du client.

Lors de l'implémentation, la structure adoptée comportera un nombre souvent inférieur de niveaux : les fonctions seront éventuellement redistribuées vers d'autres niveaux. Par exemple, il arrivera fréquemment que toutes les fonctions de conception (*design*) soient regroupées au sein d'un seul niveau.

3.4. L'approche fonctionnelle

Cette approche permet d'identifier différents agents ayant une fonction bien précise et se retrouvant à chaque niveau de décomposition.

3.4.1. Computer Aided Design (CAD)

Cette partie est responsable de la conception (*design*) d'un produit. Ce composant utilise de l'information concernant le nouveau produit. Cette information provient soit du client (lorsqu'il s'agit de conception de pièce à façon), soit du département de planification stratégique (pour le lancement futur d'un nouveau produit).

Lors de cette étape, des informations concernant les seuils de tolérances, les taux de résistances aux charges etc.. sont identifiées. Des analyses économiques et techniques sont également réalisées.

C'est lors de cette étape de conception (*design*) que l'on va identifier différents (sous) produits pouvant composer le nouveau produit. Ce dernier résultera de l'assemblage des sous-produits.

3.4.2.Computer Aided Process Planning (CAPP)

L'agent précédent a fourni un *design* de la nouvelle pièce. Maintenant il convient de s'intéresser à la façon dont on va la réaliser. Le but de ce nouvel agent est donc la conception de la recette de production.

L'élaboration de la recette de production est basée sur les informations reçues du *CAD* (pour les informations concernant le futur produit) et les informations reçues des différents équipements nécessaires à la réalisation du produit (informations concernant les caractéristiques des équipements).

3.4.3. Production Planning and Control (PPC)

Maintenant que nous savons ce que nous allons produire et comment nous allons le faire, il va falloir planifier le travail à effectuer.

Lors de cette étape, il faut tenir compte des capacités de chaque équipement de production ainsi que des commandes reçues, afin de réaliser un plan de production. Ce plan est établi en tenant compte des informations provenant du *CAPP* et des disponibilités des différents équipements. Lorsque la production a été planifiée, il est alors possible de lancer la production.

Or, lors de celle-ci, il peut survenir des incidents, d'autres commandes plus urgentes qui peuvent venir remettre en question le planning élaboré précédemment.

La planification se fait donc en deux étapes : la première se situe avant le début proprement dit de la production, c'est-à-dire que l'on prend en considération l'ensemble des commandes que l'on a déjà reçues à ce moment précis. La deuxième planification a lieu, elle, lors de la production, et a pour but de réorganiser la production pour tenir compte de nouveaux aléas, événements qui troublent le processus de fabrication (par exemple, le fait qu'une machine tombe en panne risque de provoquer des retards dans la livraison).

3.4.4. Computer Aided Manufacturing (CAM)

Cette partie prend en charge l'exécution des différentes étapes du processus prévu. L'exécution de la production est assurée par l'utilisation de différents équipements qui permettent, par exemple, l'assemblage, le transport, le façonnage, le stockage, etc.

3.5. L'approche Orienté-Agent

Les recherches récentes dans le domaine de la productique conduisent les auteurs, comme par exemple Scheer [Sch88], Dougmeingts [Dou90] à critiquer l'approche fonctionnelle et à encourager l'identification d'une architecture productique basée sur des critères de décomposition différents.

La tendance actuelle dans le Génie Logiciel est une utilisation sans cesse croissante du paradigme Orienté-Objet. Ce dernier permet une structuration plus forte des spécifications.

L'approche adoptée dans le cadre des recherches de l'équipe de Mr Eric Dubois est basée sur le concept d'agent. Ce concept peut être vu comme une spécialisation de ce que l'on nomme habituellement Objet dans les méthodes de modélisation conceptuelles.

3.6. Identification d'un composant générique

La vue "Orientée Agent" permet de construire une architecture CIM basée sur un nombre quelconque de niveaux d'abstraction, selon la complexité du processus de production de l'organisation considérée. Cependant il est intéressant de mettre en évidence des points communs entre les différents niveaux de l'architecture. Ces points communs permettent de mettre en lumière l'existence d'un niveau générique. A l'intérieur de ce niveau, on retrouve des composants ayant les fonctionnalités présentées précédemment.

A la figure suivante, nous représentons les interactions existant entre les composants d'un même niveau de production. Ce que l'on dénommera par après un niveau.

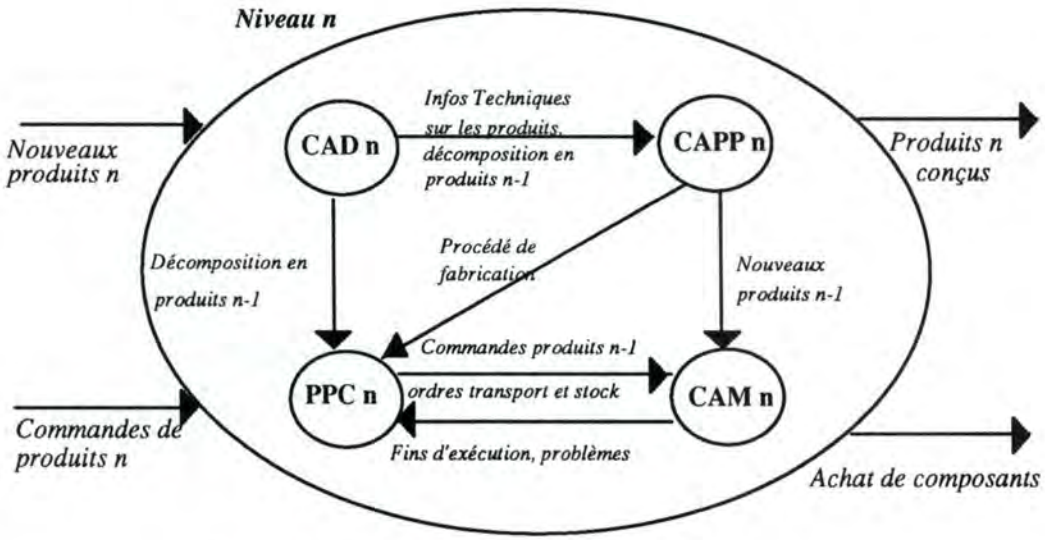


Figure **-3.1** - : Les interactions entre les composants d'un niveau

3.6.1. Computer Aided Design (CAD-n)

Le rôle du composant *CAD* du *Niveau-n* est d'identifier et de concevoir les différentes parties constituant le produit de Niveau-n. En plus de cette identification, il est nécessaire de mettre en évidence les tolérances, les charges et toutes autres caractéristiques que le produit doit posséder. Le CAD-n doit également spécifier comment les différents éléments (produits de niveau n-1) doivent être assemblés pour obtenir le produit final (produit du niveau-n).

3.6.2. Computer Aided Process Planning (CAPP-n)

Le rôle de ce composant est de préparer la recette de production du produit concerné. Cette recette de production fixe les différentes étapes de la fabrication qui doivent être effectuées par les moyens de production de ce niveau n. De plus, la séquence des ordres de fabrication doit aussi être fixée.

3.6.3. Production Planning and Control (PPC-n)

Ce composant est responsable de la planification (des allocations) du travail des différents équipements du niveau n. Le plan de fabrication résultant est constitué d'une suite de périodes pendant lesquelles les équipements sont occupés ou libres. Le *PPC* joue également un rôle de séquenceur, pour le lancement des différents ordres à destination des équipements du niveau n. Il assure aussi le contrôle et le suivi de l'exécution des ordres envoyés aux équipements.

3.6.4. Computer Aided Manufacturing générique (CAM-n)

Le rôle du composant *CAM-n* est de fabriquer le produit demandé à ce niveau. Pour ce faire, il est peut faire appel aux composants suivants : les *entrées/sorties* du niveau-n, les *unités de production du niveau inférieur*, le *stockage* des produits du niveau-n, les *moyens de transport* au sein du niveau-n.

Cependant, le composant *CAM-n* n'a pas de rôle effectif comme les autres agents. En effet, pour fabriquer les différents produits de l'organisation ou de l'unité de production, il faut disposer de machines. La production des biens est assurée par les machines. On conserve la notion d'agent *CAM-n* à des fins de regroupement syntaxique des moyens de production de l'organisation.

A la figure suivante, nous présentons la structure générique du composant *CAM n.*

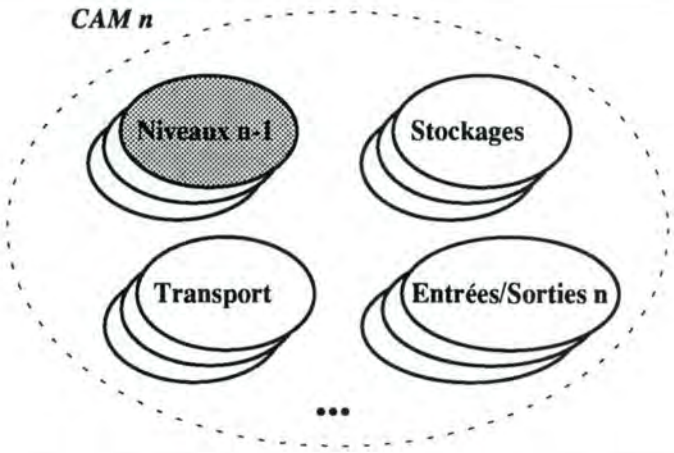


Figure -**3.2** - La structure générique du composant *CAM n*

Désormais, dans les figures, nous représentons les niveaux pouvant faire l'objet d'une décomposition plus fine en grisé.

Les composants *CAM-n* seront, quant à eux, représentés en pointillés afin de rappeler qu'ils ne constituent qu'un regroupement syntaxique d'équipements de production.

3.6.4.1. Les entrées/sorties (Niveau n)

Les *entrées-sorties* du niveau n sont des points d'accès au composant *CAM-n*. Il s'agit d'endroits où les produits entrant et sortant du niveau n peuvent être placés. Ce composant d'entrée-sortie est nécessaire car son absence signifierait que le niveau n est refermé sur lui même, c'est-à-dire qu'il n'aurait aucun contact avec l'extérieur, avec d'autres unités de production par exemple.

Michaël Petit présente le composant *Entrée/Sortie* comme étant **toujours obligatoire** : il y a toujours au moins un composant *Entrée/sortie* dans un composant *CAM n*. Si cela n'était pas le cas, le niveau n serait complètement fermé et donc rien ne pourrait y rentrer n'y en sortir.

☞ Suite à l'instanciation de la structure générique, nous avons apportés une modification aux Entrées/Sorties. Voir la section sur l'évaluation de la structure générique.

3.6.4.2. Les unités de production inférieures (Niveau n-1)

Les unités de production inférieures assurent la production d'une partie du produit fabriqué par le niveau n. Par exemple, pour un niveau n chargé de produire des voitures, on peut disposer, par exemple, d'une unité de production d'un niveau inférieur pour la fabrication des sièges, une autre pour l'assemblage du bloc moteur, etc. A ce moment de la spécification, on ne "connaît rien" (on ne dispose que d'une vue abstraite) de ces unités de production inférieures, on ne connaît que ses entrées/sorties c'est-à-dire ce qu'elles reçoivent et ce qu'elles fournissent (modèle de la boîte noire).

Le composant *Niveau n-1* est **facultatif** : comme nous l'avons déjà mentionné auparavant, le nombre de niveaux dépend de la structure de production de l'organisation concernée.

3.6.4.3. Le stockage (Niveau n)

Le composant de *stockage* est un lieu où les produits transitant à l'intérieur du niveau n sont stockés pendant un certain temps. Il a un comportement proche de l'entrée/sortie, cependant il est limité à un usage interne et n'a aucun contact avec l'extérieur.

Le composant *Stockage* n est **facultatif** : la remarque faite concernant les *niveaux n-1* est également valable pour le composant de *stockage*.

3.6.4.4. Le moyen de transport (Niveau n)

Le moyen de *transport* a pour but de transporter les produits du *niveau n* d'un composant interne vers un autre.

Ces deux composants peuvent être

- du même type, par exemple deux unités de production, deux emplacements d'entrée/sortie
- ou de type différent, par exemple une unité de production et une entrée/sortie, un stockage et une unité de production.

Le type de transporteur peut varier d'un niveau à l'autre, par exemple, pour le niveau de production des voitures, le moyen de transport utilisé pourrait être un tapis roulant alors que pour l'unité de production des sièges (sellerie), un AGV (*Auto Guided Vehicule*) peut être utilisé pour transporter les sièges d'un endroit vers un autre.

Le composant de *transport* est **obligatoire** : en effet, si cela n'était pas le cas, lorsqu'il n'y a pas de transport dans un niveau : aucun mouvement de produits ne serait possible entre par exemple les *Entrées/Sorties* de ce niveau et le composant de *Stockage*.

3.7. Structure d'un niveau générique

La figure suivante représente la structure d'un niveau générique. Le symbole "..." signifie que la structure que l'on donne n'est pas complète: le cas échéant on pourra ajouter un ou l'autre composant selon les cas. Par exemple, un composant de contrôle de qualité. Cette figure constitue donc un agrégat des figures présentées précédemment.

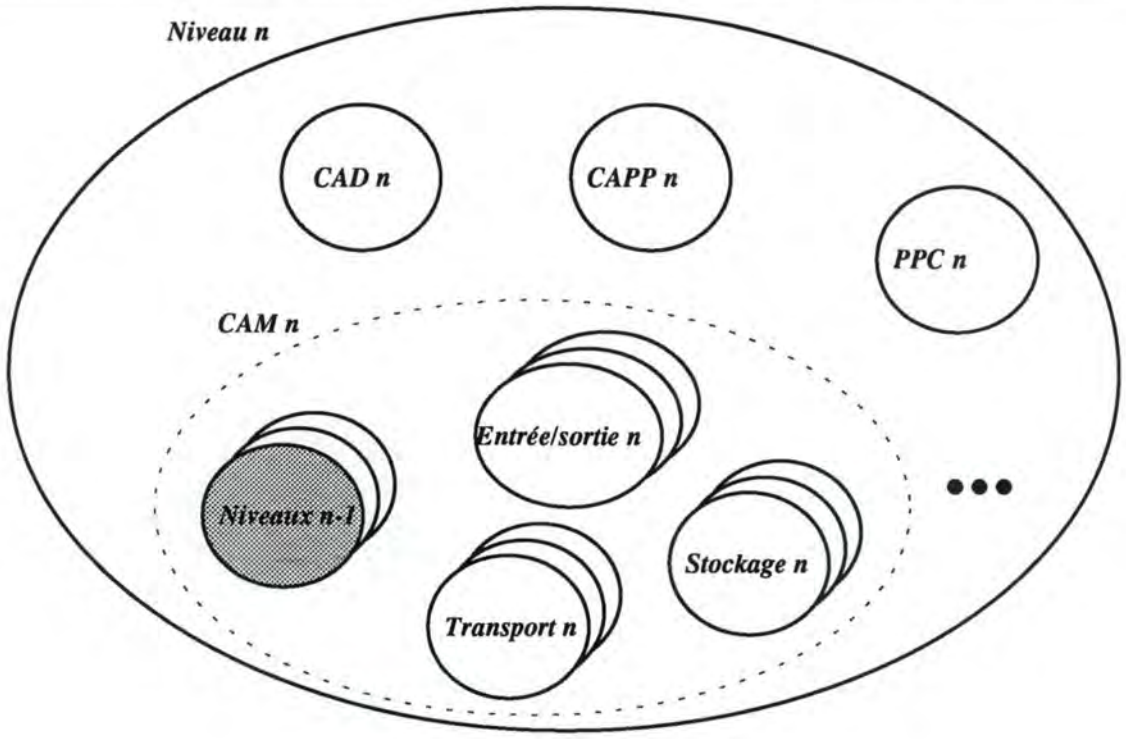


Figure -**3.3** - : La structure générique d'un niveau.

3.8. Notion de niveau terminal

Comme nous venons de le voir, la structure d'un niveau est constituée de niveaux n-1. Puisqu'un tel composant générique peut être instancié un nombre quelconque de fois, à différents niveaux, il convient de définir un niveau terminal, c'est-à-dire un niveau qui ne fasse plus l'objet d'une décomposition en niveaux inférieurs.

Un niveau terminal dispose d'une structure propre (simplifiée). L'interface (c'est-à-dire les informations reçues en entrée et celles fournies en sortie) d'un tel niveau respecte celle d'un niveau pour pouvoir être utilisée à n'importe quel niveau. Puisque le niveau terminal respecte l'interface d'un niveau, on retrouve les fonctions de CAD, CAPP, PPC que l'on qualifie ici de terminales.

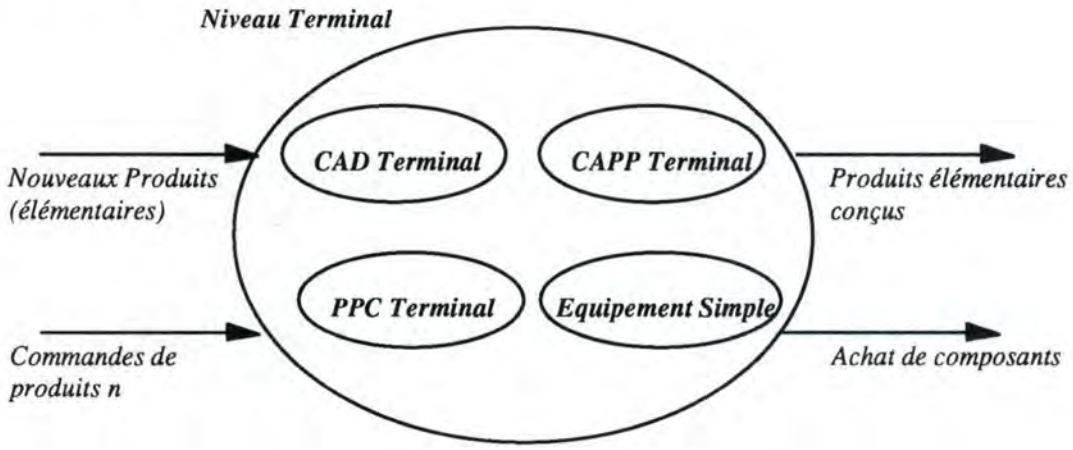


Figure -**3.4** - La structure d'un niveau terminal

3.8.1. CAD-terminal

Ce composant est en charge d'assurer la conception des produits terminaux, c'est-à-dire qui ne peuvent être décomposés en sous-produits.

3.8.2. CAPP-Terminal

Ce composant est chargé de la conception des programmes NC (c'est-à-dire les recettes de production) pour les produits du niveau terminal. Ces programmes commandent les différents équipements.

3.8.3. PPC-Terminal

Ce composant est responsable de la planification, du contrôle et du séquencement des ordres de productions de produits terminaux reçus du niveau supérieur. Il dirige donc l'exécution des programmes NC.

La fonction CAM terminale est prise en charge par un équipement simple. On n'y retrouve ni composant de transport, ni composant de stockage.

3.8.4. Equipement-simple

Michaël Petit [Pet92] définit ce composant comme étant uniquement capable d'exécuter des ordres de production et ne réalisant aucune fonctionnalité de type CAD, CAPP, PPC. Il est uniquement composé d'entrée/sortie.

☞ Suite à l'instanciation de la structure générique, nous avons modifiés la structure de l'Equipement simple. Voir la section sur l'évaluation de la structure générique.

3.9. Exemple d'application de la structure générique

Supposons que l'on doive structurer une entreprise automobile.

L'entreprise est structurée (par exemple) de la façon suivante :

- La sellerie : unité spécialisée dans la fabrication des sièges.
- La carrosserie : responsable de tous les traitements portant sur la carrosserie des voitures. Les activités de l'atelier s'étendent de la carrosserie à la peinture.
- La fonderie : est chargée de la production du bloc moteur.
- L'atelier de mécanique : responsable de la mise en place du moteur.

Si nous structurons notre entreprise automobile à l'aide du niveau générique proposé auparavant, nous obtenons la situation présentée à la figure suivante :

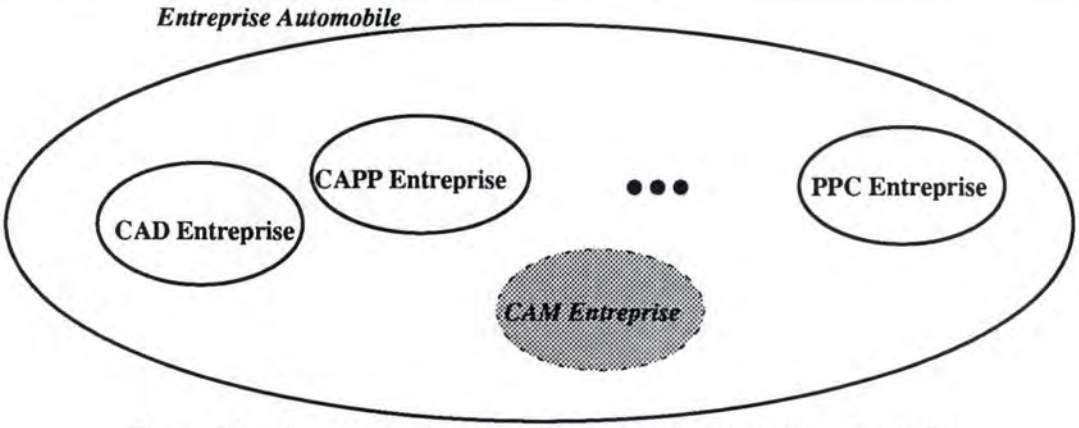


Figure -**3.5** - : Le premier niveau de structuration de l'entreprise automobile.

Notre premier niveau correspond à l'entreprise automobile proprement dite. Celle-ci dispose d'un département *CAD*, chargé de la conception de nouveaux modèles de voitures. La recette de production de ces modèles est fixée par le *CAPP Entreprise*. Les différentes commandes sont ordonnées et planifiées par le *PPC Entreprise*. Les commandes reçues sont réalisées par le composant *CAM Entreprise*.

Le composant *CAM* n'a pas d'existence en soi, il est considéré comme un regroupement syntaxique utilisé à des fins de lisibilité et de clarté uniquement. C'est dans ce composant que l'on retrouvera, en fait, les différents moyens de production de l'entreprise.

Pour réaliser les voitures, le niveau Entreprise fait appel aux moyens de production présents dans le composant *CAM Entreprise*. Il convient donc "d'ouvrir" ce composant et de le structurer en terme de niveau.

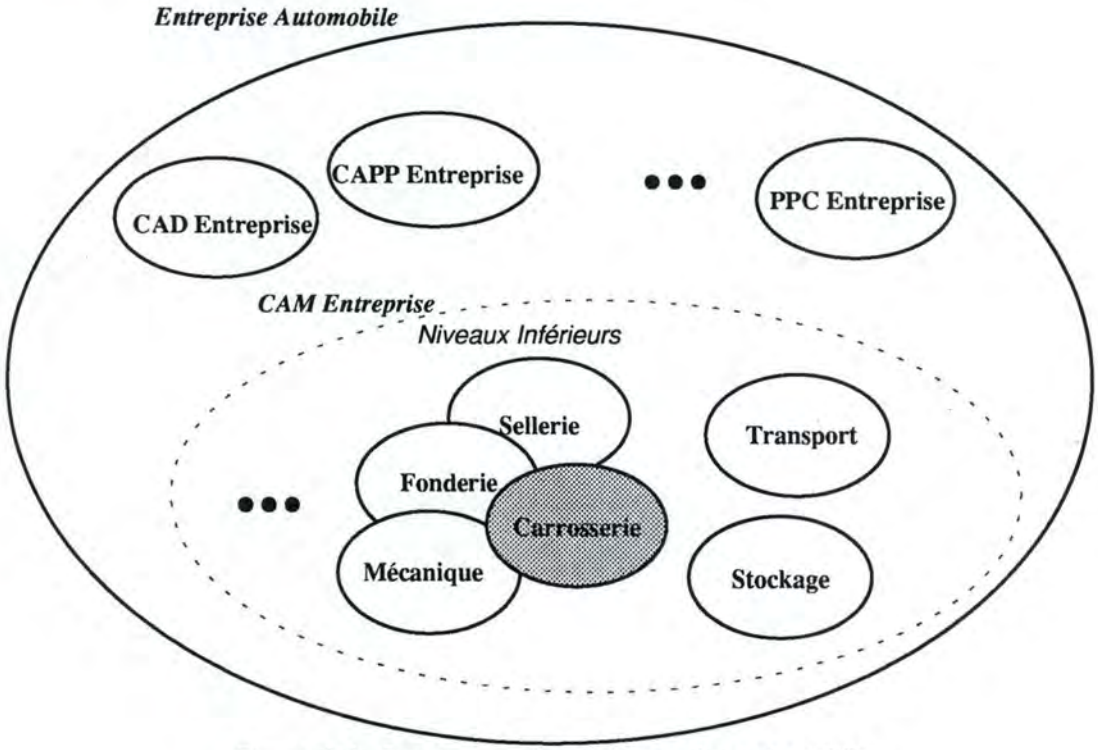


Figure -**3.6** - Structure du niveau *Entreprise automobile*.

Dans notre cas, nous allons considérer uniquement l'atelier de carrosserie comme unité de production et donc niveau inférieur et décomposable de la même manière qu'un niveau générique.

L'atelier de carrosserie est composé:

- d'un Robot qui est chargé de souder les différents éléments de carrosserie.
- d'une Presse destinée à la mise en forme des composants de la carrosserie.
- d'un four.
- d'une station de découpe.

On retrouve également un *CAPP Carrosserie*, pour l'élaboration des recettes de production, un *PPC Carrosserie* chargé de la planification du travail de l'atelier de carrosserie.

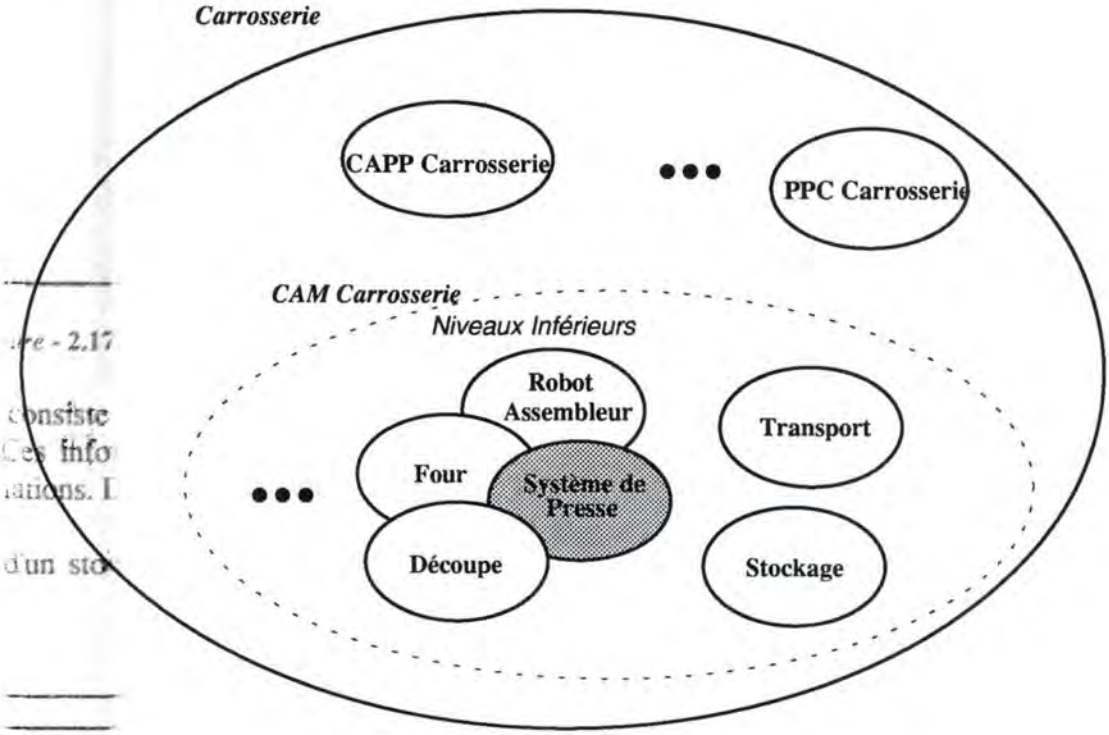


Figure -3.7 - Structure du niveau *Carrosserie*

Dans le composant *CAM Carrosserie* on peut trouver des niveaux supplémentaires de décomposition qui sont : le *système de Presse, un robot assembleur, un four et un poste de découpe*. Dans notre cas, nous n'allons détailler que le système de Presse que l'on considère comme un niveau terminal et donc nous allons instancier la structure de niveau correspondante.

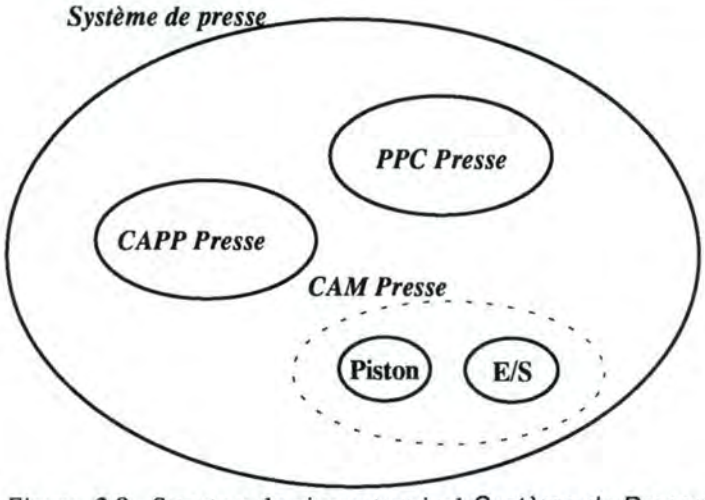


Figure -**3.8** - Structure du niveau terminal *Système de Presse*

On peut structurer l'agent *Presse Hydraulique* comme Equipement-simple, c'est-à-dire qu'il est composé d'entrées/sorties et de composants d'équipement.

3.10. Evaluation de la structure générique

3.10.1. Introduction générale

Suite à l'application de la structure générique proposée par Michaël Petit, au cas concret de l'atelier du Centre de Recherche Henri Tudor, nous avons cru bon d'affiner et même de modifier quelque peu la spécification de certains composants.

Les principales modifications apportées sont présentées ci-dessous. Le lecteur intéressé peut consulter le mémoire de Michaël Petit [Pet 92] pour les spécifications d'agents qui sont restées telles quelles.

Afin d'améliorer la clarté de ce mémoire, nous avons choisi d'utiliser les symboles suivants:

- Signifie que l'élément décrit est un type d'entité.
- apparaît quand la description concerne une action.

3.10.2. Remarque concernant les Entrées/Sorties

Le lecteur pourra constater, dans l'application de la structure générique donnée au chapitre suivant, que pour un niveau donné, nous ne disposons pas de composant d'Entrée/Sortie. Ces composants sont en fait présents dans un niveau inférieur.

Il convient donc de revoir quelque peu la structure générique. En effet, il semblerait plus opportun d'imposer que l'on truve un composant d'entrée/sortie au moins dans un des niveaux inférieurs.

3.10.3. Remarque concernant les Equipements-Simples

Nous proposons de revoir la structure de l'équipement simple proposée par Michaël Petit. En effet, Michaël Petit définit l'équipement simple comme étant un

ensemble d'entrées/sorties qui soit capable d'exécuter des ordres de productions (élémentaires) directement sur le produit. Or, les entrées/sorties ne sont pas douées de telles caractéristiques.

Il semblerait plus adéquat d'ajouter un élément dans la structure de l'équipement simple : à savoir les composants de cet équipement. En effet, tout équipement aussi simple qu'il soit est constitué d'un ou plusieurs composants qui interviennent directement sur les produits. Seuls ces composants sont indécomposables.

On trouvera à la figure suivante la structure modifiée de l'Equipement-Simple:

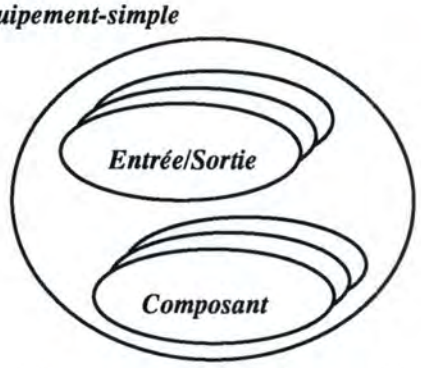


Figure **-3.9.** - Structure d'un Equipement-Simple

3.10.4. Le CAD de niveau n

3.10.4.1. Introduction

Pour rappel, le rôle du CAD de niveau n est de réaliser la conception de nouveaux produits que ce niveau devra, par la suite, réaliser.

3.10.4.2. Structure d'état de l'agent CAD n

Le schéma décrivant la structure de tous les états du composant CAD n est donné à la figure 3.10.

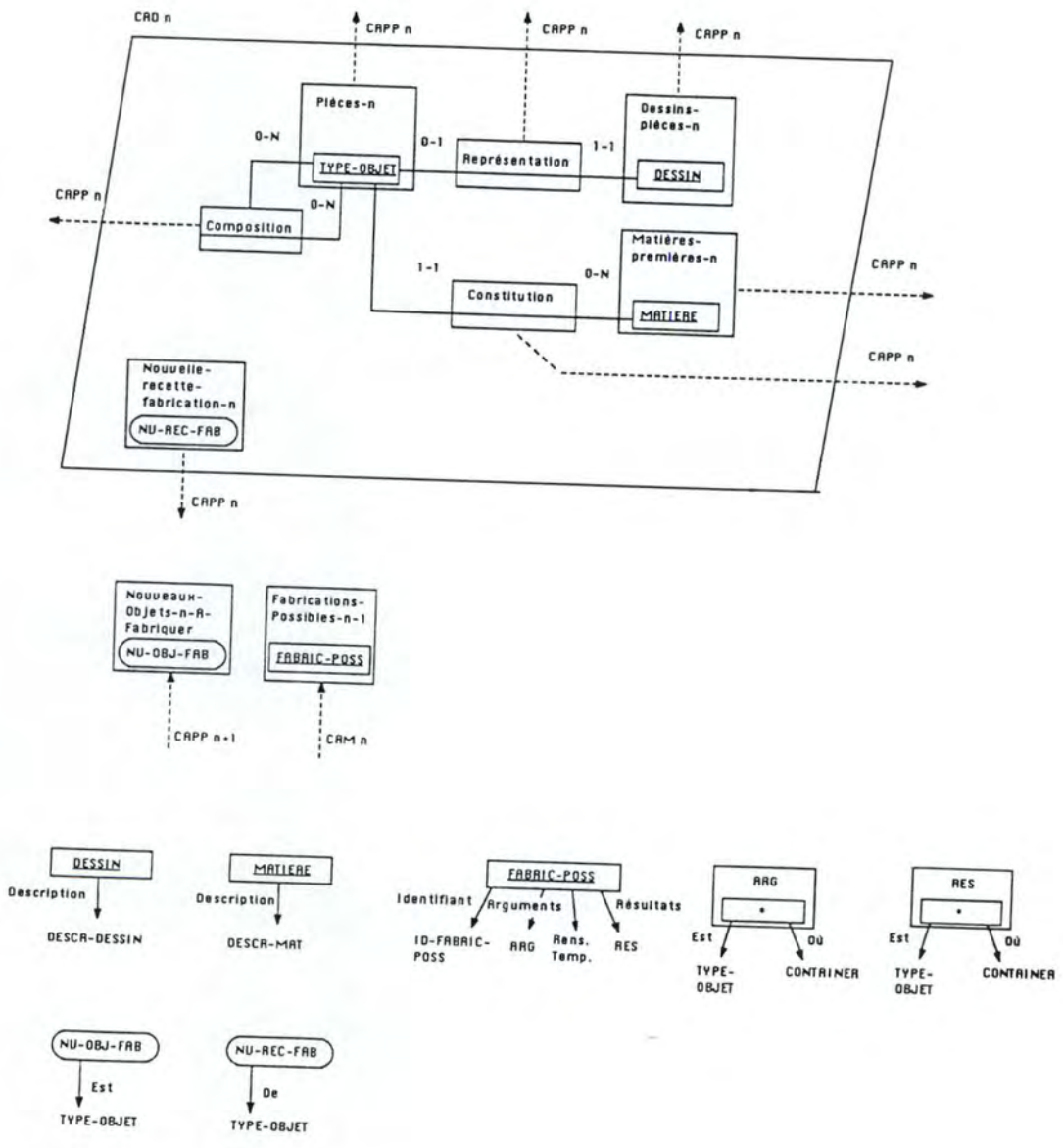


Figure **-3.10-** Schéma du composant CAD n.

3.10.4.3. Description de l'agent CAD n

Informations gérées par l'agent

- *Dessins_pièces_n*

  ■ Description : Contient un ensemble d'informations sur tous les dessins connus du CAD: identifiant, date de création, date de mise à jour,...
  ■ Type : **DESSIN**
  ■ Lien : • **Représentation**: Permet de désigner la pièce représentée par un dessin.
  ■ Visible par : *CAPP n*
  ■ Structure : ☞ Figure 3.10

- *Pièces_n*

  ■ Description : Ensembles des pièces que le niveau n est en mesure de fabriquer. Ces pièces sont décrites, entre autres, par une dimension idéale et un poids idéal.
  ■ Type : **TYPE_OBJET**
  ■ Liens : • **Représentation**: Voir ci-dessus.
  • **Composition**: Permet de connaître les pièces qui en composent d'autres.
  • **Constitution**: Définit la matière première dans laquelle la pièce est fabriquée.
  ■ Visible par : *CAPP n*
  ■ Structure : ☞ Figure 3.10

- *Matières_premières_n*

  ■ Description : Reprend une description informelle et quelques formules de calcul importantes pour chaque matière connue du CAD.
  ■ Type : **MATIERE**
  ■ Lien : • **Constitution**: Voir ci-dessus.
  ■ Visible par : *CAPP n*
  ■ Structure : ☞ Figure 3.10

- *Nouvelle_recette_fabrication_n*

  ■ Description : Signale que le CAD vient de terminer la conception d'une pièce et demande au CAPP de réaliser la recette qui va permettre de la fabriquer.
  ■ Type : **NV_REC_FAB**
  ■ Destination : *CAPP n*
  ■ Structure : ☞ Figure 3.10

Informations reçues de l'extérieur

- *Fabrications_possibles_n-1*

  ■ Description Ensembles des fabrications que peuvent réaliser les niveaux inférieurs.
  ■ Type **FABRIC_POSS**
  ■ Lien ∅
  ■ Provient de *CAM n*
  ■ Structure ☞ Figure 3.10

- *Nouveaux_objets_n_a_fabriquer*

  ■ Description Demande de conception d'un nouveau produit (qui provient du niveau supérieur).
  ■ Type **NV_OBJ_FAB**
  ■ Origine *CAPP n+1*
  ■ Structure ☞ Figure 3.10

3.10.5. Le CAPP de niveau n

3.10.5.1. Introduction

L'agent CAPP de niveau n a pour objectif de réaliser les recettes de fabrication des produits de ce niveau. Pour ce faire, il se base sur les caractéristiques générales des équipements du niveau et sur les informations que lui fournit le CAD. Il doit également rendre le niveau capable de réaliser les opérations de transport, retraits et stockages qui lui sont demandées.

3.10.5.2. Structure d'état de l'agent CAPP n

Le schéma relatif au composant CAPP n est représenté à la figure 3.11 et la structure des types est reprise à la figure 3.12.

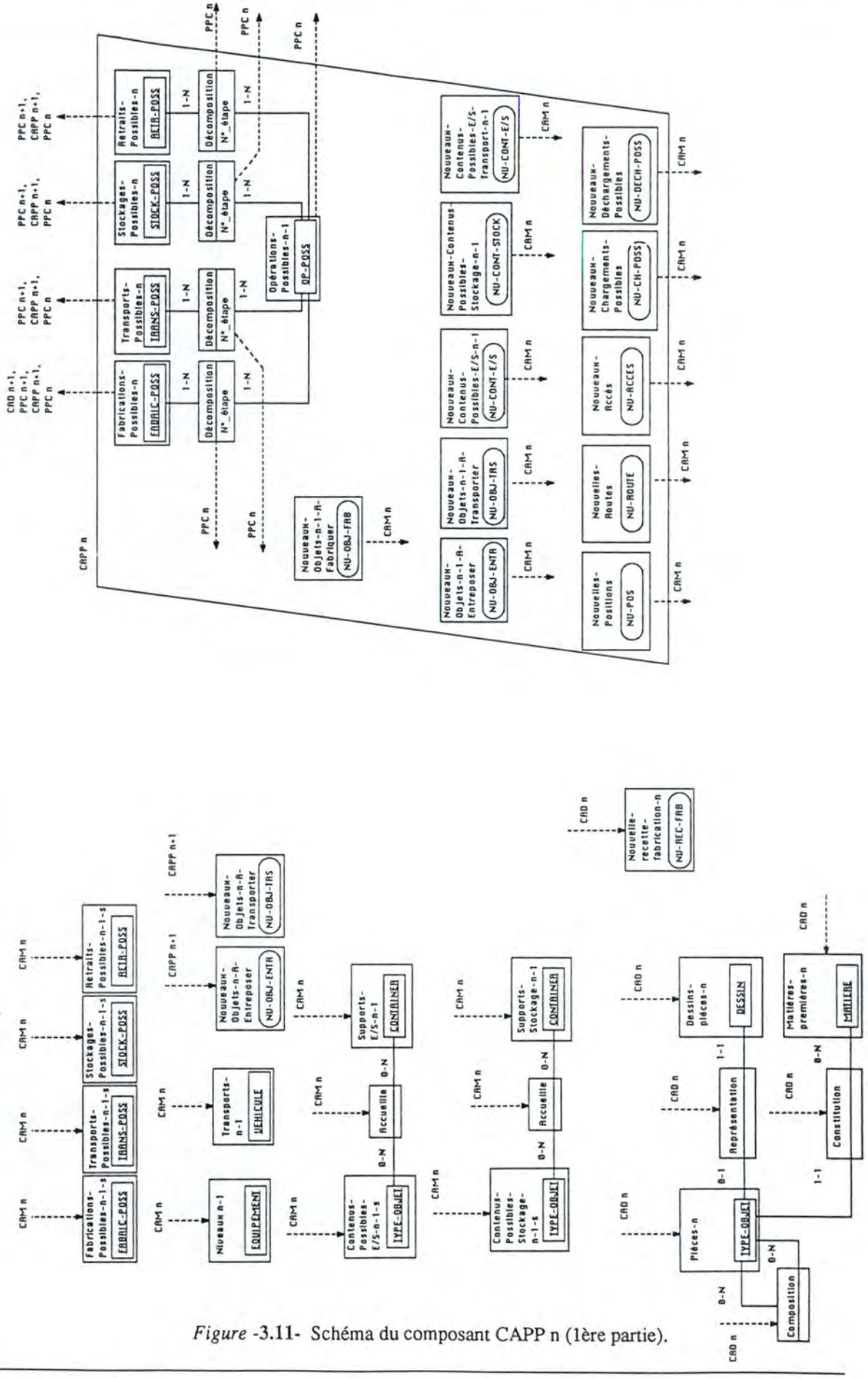


Figure -3.11- Schéma du composant CAPP n (1ère partie).

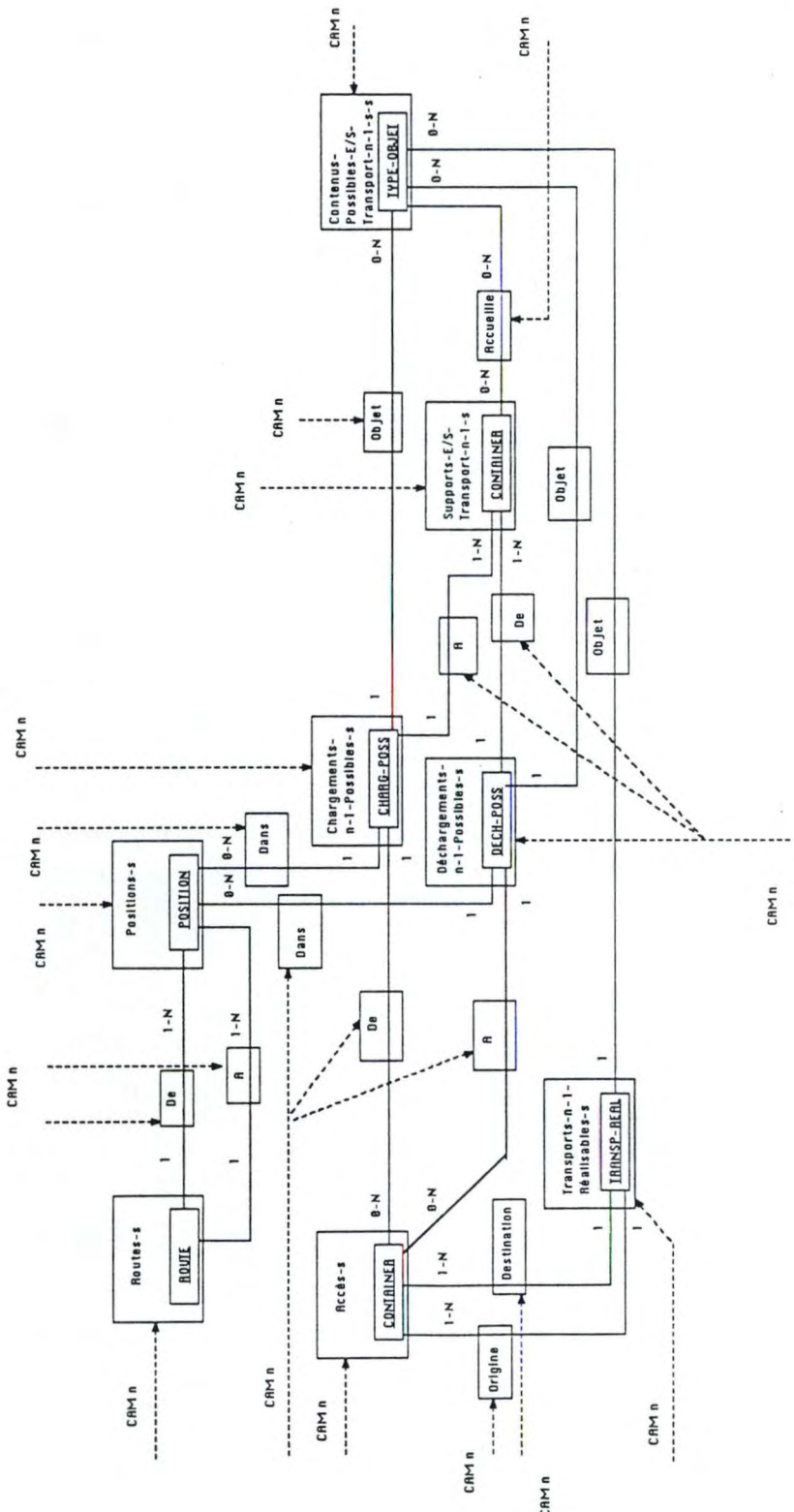


Figure **3.11 (suite)** Schéma du composant CAPP n (2ème partie).

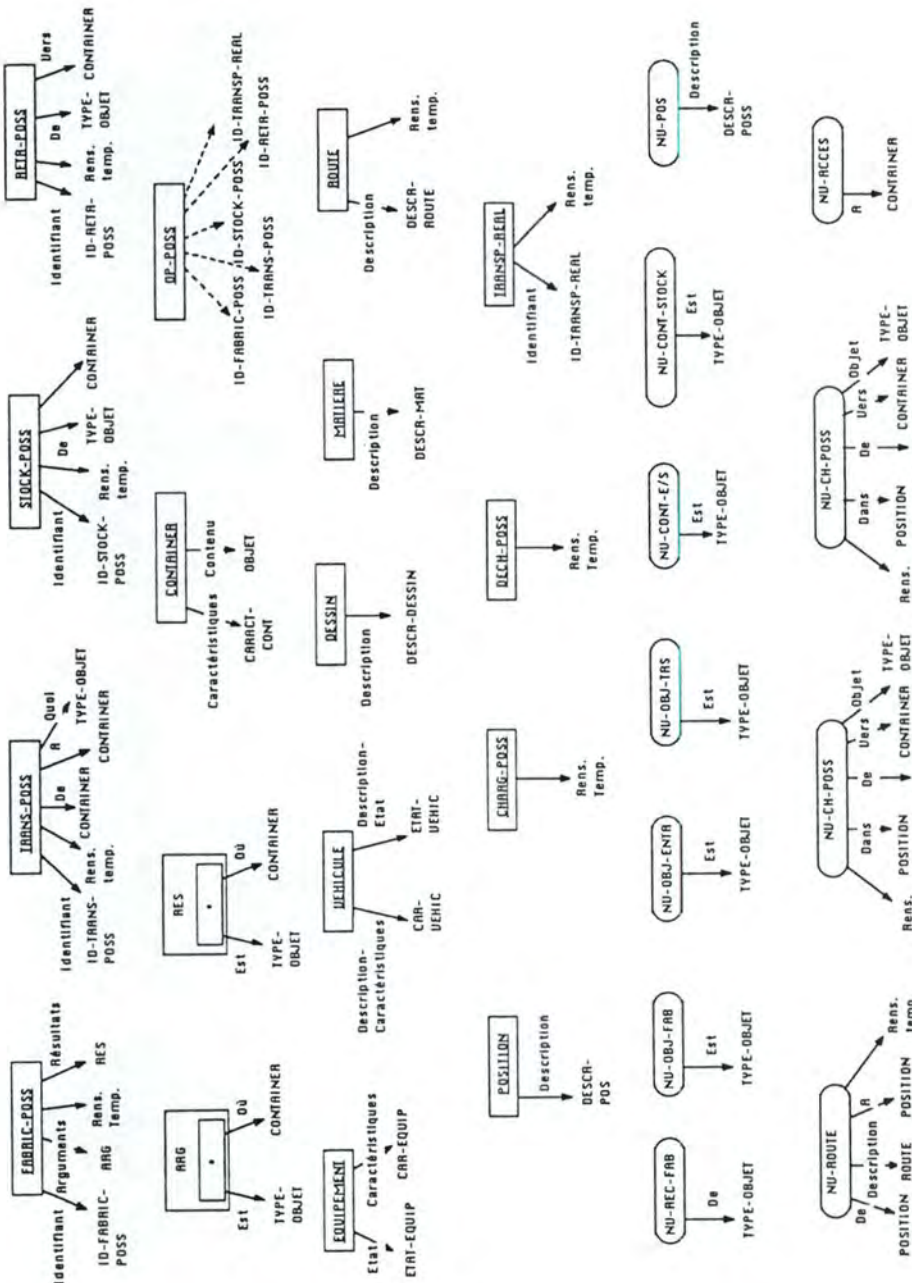


Figure **3.12** : Le composant CAPP n: structure des types.

3.10.5.3. Description de l'agent CAPP n

Informations gérées par l'agent

- *Opérations_possibles_n-1*

  ■ Description Références à des Fabrications_possibles, Transports_possibles, Stockages_possibles, Retraits_possibles et Transports_réalisables de niveaux n-1.
  ■ Type **OP_POSS**
  ■ Lien • **Décomposition**: Ce type d'association permet de se référer, pour une opération de niveau n (Fabrications_possibles_n, Transports_possibles_n, Stockages_possibles_n et Retraits_possibles_n), à un ensemble d'opérations de niveaux n-1 (Fabrications_possibles_n-1_s, Transports_possibles_n-1_s, Stockages_possibles_n-1_s, Retraits_possibles_n-1_s et Transports_n-1_réalisables_s) dont l'agencement (grâce au numéro d'étape) réalise l'opération de niveau n concernée.
  ■ Visible par *PPC n*
  ■ Structure ☞ figure 3.12

- *Fabrications_possibles_n*

  ■ Description Ensemble d'informations sur les différentes fabrications réalisables par le niveau.
  ■ Type **FABRIC_POSS**
  ■ Lien • **Décomposition**: Voir ci-dessus.
  ■ Visible par *PPC n, CAPP n+1, PPC n+1, CAD n+1*
  ■ Structure ☞ figure 3.12

- *Transports_possibles_n*

  ■ Description Possibilités de transport qu'offre le niveau.
  ■ Type **TRANS_POSS**
  ■ Liens • **Décomposition**: Voir ci-dessus.
  ■ Visible par *PPC n, CAPP n+1, PPC n+1*
  ■ Structure ☞ figure 3.12

- *Stockages_possibles_n*

  ■ Description Stockages que le niveau est en mesure de réaliser.
  ■ Type **STOCK_POSS**
  ■ Lien • **Décomposition**: Voir ci-dessus.
  ■ Visible par *PPC n, CAPP n+1, PPC n+1*
  ■ Structure ☞ figure 3.12

- *Retraits_possibles_n*

  ■ Description Ensemble des retraits que le niveau n+1 peut demander au niveau n.
  ■ Type **RETR_POSS**
  ■ Lien • **Décomposition**: Voir ci-dessus.
  ■ Visible par *PPC n, CAPP n+1, PPC n+1*
  ■ Structure ☞ figure 3.12

- *Nouveaux_objets_n-1_a_fabriquer*

  ■ Description Demande au CAD de l'un des niveaux n-1 (du composant CAM n) de concevoir une nouvelle pièce.
  ■ Type **NV_OBJ_FAB**
  ■ Destination *CAM n*
  ■ Structure ☞ figure 3.12

- *Nouveaux_objets_n-1_a_entreposer*

  ■ Description Informe les niveaux inférieurs qu'ils devront stocker de nouveaux objets.
  ■ Type **NV_OBJ_ENTR**
  ■ Destination *CAM n*
  ■ Structure ☞ figure 3.12

- *Nouveaux_objets_n-1_a_transporter*

  ■ Description Demande, à un niveau inférieur (n-1), la conception d'un nouveau transport.
  ■ Type **NV_OBJ_TRS**
  ■ Destination *CAM n*
  ■ Structure ☞ figure 3.12

- *Nouveaux_contenus_possibles_E/S_n-1*

  ■ Description Sur base des informations générales dont il a connaissance, le CAPP imagine des nouveaux contenus pour les entrées-sorties du composant CAM de niveau n.
  ■ Type **NV_CONT_E/S**
  ■ Destination *CAM n*
  ■ Structure ☞ figure 3.12

- *Nouveaux_contenus_possibles_stockage_n-1*

  - Description : Indique les nouveaux contenus que les composants de stockage (du CAM n) peuvent accepter.
  - Type : **NV_CONT_STOCK**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouveaux_contenus_possibles_E/S_transport_n-1*

  - Description : Signale au moyen de transport du composant CAM n qu'il peut accueillir un nouveau type d'objet à ses entrées-sorties.
  - Type : **NV_CONT_E/S**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouvelles_positions*

  - Description : Donne les caractéristiques de positions que le moyen de transport pourra occuper.
  - Type : **NV_POS**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouvelles_routes*

  - Description : Indique une nouvelle route à un des moyens de transport du composant CAM n.
  - Type : **NV_ROUTE**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouveaux_accès*

  - Description : Transmet les caractéristiques d'un nouveau container auquel le moyen de transport a accès.
  - Type : **NV_ACCES**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouveaux_chargements_possibles*

  - Description : Communique les caractéristiques d'un nouveau chargement possible que le CAPP a imaginé.
  - Type : **NV_CH_POSS**
  - Destination : *CAM n*
  - Structure : ☞ figure 3.12

- *Nouveaux_déchargements_possibles*

■ Description Le CAPP conçoit de nouveaux déchargements et fait part de leurs descriptions au moyens de transports du CAM n.
■ Type **NV_DECH_POSS**
■ Destination *CAM n*
■ Structure ☞ figure 3.12

Informations reçues de l'extérieur

- *Fabrications_possibles_n-1_s*

■ Description Fabrications que les niveaux inférieurs (n-1) sont en mesure de réaliser.
■ Type **FABRIC_POSS**
■ Lien ∅
■ Provient de *CAM n*
■ Structure ☞ figure 3.12

- *Transports_possibles_n-1_s*

■ Description Ensemble des transports que les niveaux n-1 peuvent effectuer.
■ Type **TRANS_POSS**
■ Liens ∅
■ Provient de *CAM n*
■ Structure ☞ figure 3.12

- *Stockages_possibles_n-1_s*

■ Description Caractéristiques des stockages possibles des niveaux n-1.
■ Type **STOCK_POSS**
■ Lien ∅
■ Provient de *CAM n*
■ Structure ☞ figure 3.12

- *Retraits_possibles_n-1_s*

■ Description Descriptif des retraits de produits que les niveaux n-1 peuvent réaliser.
■ Type **RETR_POSS**
■ Lien ∅
■ Provient de *CAM n*
■ Structure ☞ figure 3.12

- *Niveaux_n-1*

  ■ Description Caractéristiques générales des niveaux n-1.
  ■ Type **EQUIPEMENT**
  ■ Lien ∅
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Transports_n-1*

  ■ Description Descriptif des moyens de transport du composant CAM n
  ■ Type **VEHICULE**
  ■ Lien ∅
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Supports_E/S_n-1*

  ■ Description Ensemble des entrées-sorties du composant CAM n.
  ■ Type **CONTAINER**
  ■ Lien • **Décomposition**: Détermine le contenu que les containers peuvent accueillir.
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Contenus_possibles_E/S_n-1_s*

  ■ Description Types d'objets que les entrées-sorties n-1 acceptent.
  ■ Type **TYPE_OBJET**
  ■ Lien • **Décomposition**: Voir ci-dessus.
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Supports_stockage_n-1*

  ■ Description Informations relatives aux composants de stockage du CAM n
  ■ Type **CONTAINER**
  ■ Lien • **Décomposition**: Désigne les contenus acceptés par les stocks n.
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Contenus_possibles_stockage_n-1_s*

  - Description : Liste des contenus possibles des composants de stockage n.
  - Type : **TYPE_OBJET**
  - Lien : • **Décomposition**: Voir ci-dessus.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Dessins_pièces_n*

  - Description : Informations relatives aux dessins réalisés par le CAD de niveau n.
  - Type : **DESSIN**
  - Lien : • **Représentation**: Indique quelle est la pièce représentée par un dessin.
  - Provient de : *CAD n*
  - Structure : ☞ figure 3.12

- *Pièces_n*

  - Description : Ensemble des pièces que le niveau n est en mesure de fabriquer.
  - Type : **T_OBJET**
  - Liens : • **Représentation**: Voir ci-dessus.
    • **Composition**: Détermine les objets qui en composent d'autres.
    • **Constitution**: Désigne la matière première qui constitue un objet.
  - Provient de : *CAD n*
  - Structure : ☞ figure 3.12

- *Matières_premières_n*

  - Description : Liste des matières premières connues du niveau.
  - Type : **MATIERE**
  - Lien : • **Constitution**: Voir ci-dessus.
  - Provient de : *CAD n*
  - Structure : ☞ figure 3.12

- *Chargements_n-1_possibles_s*

  - Description : Ensemble des chargements réalisables par les moyens de transport du composant CAM n.
  - Type : **CHARG_POSS**
  - Liens :
    - **Objet**: Désigne le type d'objet concerné par le chargement.
    - **A**: Indique le container du moyen de transport dans lequel l'objet est chargé.
    - **De**: Indique le container accessible à partir duquel le chargement s'effectue.
    - **Dans**: Donne la position que le transport doit occuper lors du chargement.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Déchargements_n-1_possibles_s*

  - Description : Ensemble des déchargements réalisables
  - Type : **DECH_POSS**
  - Liens :
    - **Objet**: Indique le type d'objet concerné par le déchargement.
    - **De**: Détermine le container du moyen de transport qui contient l'objet à décharger
    - **A**: Désigne le container dans lequel l'objet va être déchargé.
    - **Dans**: Définit la position à occuper pour effectuer le déchargement.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Transports_n-1_réalisables_s*

  - Description : Ensemble des transports réalisables
  - Type : **TRANSP_REAL**
  - Liens :
    - **Objet**: Détermine le type d'objet transporté.
    - **Origine**: Désigne le container (accessible) à partir duquel le transport est réalisé.
    - **Destination**: Indique le container qui accueille l'objet après transport.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Supports_E/S_transport_n-1_s*

  - Description : Informations sur les containers des entrées-sorties du moyen de transport.
  - Type : **CONTAINER**
  - Liens :
    - **De**: Voir ci-dessus.
    - **A**: Voir ci-dessus.
    - **Accueille**: Indique les types d'objets qu'un container peut accueillir.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Contenus_possibles_E/S_transport_n-1_possibles_s_s*

  - Description : Informations sur les contenus possibles des containers.
  - Type : **TYPE_OBJET**
  - Liens :
    - **Objet**: Voir ci-dessus.
    - **Accueille**: Voir ci-dessus.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Accès_s*

  - Description : Descriptif de l'ensemble des containers auxquels les moyens de transport ont accès.
  - Type : **CONTAINER**
  - Liens :
    - **De**: Voir ci-dessus.
    - **A**: Voir ci-dessus.
    - **Origine**: Voir ci-dessus.
    - **Destination**: Voir ci-dessus.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Position_s*

  - Description : Ensemble des positions connues du transport.
  - Type : **POSITION**
  - Liens :
    - **Dans**: Voir ci-dessus.
    - **De**: Donne la position qui est à l'origine de la route.
    - **A**: Désigne la destination de la route.
  - Provient de : *CAM n*
  - Structure : ☞ figure 3.12

- *Route_s*

  ■ Description Ensembles des routes que le transport n-1 peut emprunter.
  ■ Type **ROUTE**
  ■ Liens • **De**: Voir ci-dessus.
  • **A**: Voir ci-dessus.
  ■ Provient de *CAM n*
  ■ Structure ☞ figure 3.12

- *Nouveaux_objets_n_a_entreposer*

  ■ Description Informe le CAPP n que le niveau doit pouvoir stocker d'autres types d'objets.
  ■ Type **NV_OBJ_ENTR**
  ■ Origine *CAPP n+1*
  ■ Structure ☞ figure 3.12

- *Nouveaux_objets_n_a_transporter*

  ■ Description Le CAPP de niveau n+1 demande de concevoir un nouveau transport au sein du niveau n.
  ■ Type **NV_OBJ_TRS**
  ■ Origine *CAPP n+1*
  ■ Structure ☞ figure 3.12

- *Nouvelle_recette_fabrication_n*

  ■ Description Le CAD de même niveau demande au CAPP de concevoir la recette de production associée au type d'objet désigné.
  ■ Type **NV_REC_FAB**
  ■ Origine *CAD n*
  ■ Structure ☞ figure 3.12

Chapitre 4

Structuration de l'atelier flexible

4.1. Introduction

Dans ce chapitre, nous allons instancier la structure générique, proposée au chapitre précédent, pour modéliser une (petite) entreprise productique. En l'occurrence, l'atelier flexible du Centre de Recherche Public Henri Tudor de Luxembourg.

Nous allons présenter brièvement l'organisation générale de l'atelier flexible. Ensuite, pour chaque agent de l'atelier, nous présentons ses différents composants ainsi que leur fonctionnement. Enfin, nous donnons la hiérarchie des différents composants ainsi que la structure d'état établie au moyen de l'architecture générique décrite au chapitre précédent.

Pour la structuration de l'atelier, nous procéderons selon une démarche "*Top-Down*".

L'atelier a été structuré en trois niveaux : le premier sera désigné par le **niveau Atelier** et correspond à ce que A.-W. Scheer [Sch88] appelle Fabrique (*Factory*). Le

second constitue le **niveau Palettage** et équivaut au Groupe d'équipement (*Equipment group*) et enfin le dernier est ce que Scheer appelle le niveau Equipement (*Equipment*) et sera dénommé **niveau Système de bridage**.

Ce chapitre correspond donc à la première étape de l'utilisation du langage ALBERT, c'est-à-dire la phase de déclaration, lors de laquelle nous définissons la hiérarchie des agents constituant l'atelier flexible.

4.2. Etude de l'existant et structuration de l'atelier

Pour structurer l'atelier, nous allons passer en revue les différents équipements de l'atelier : pour chacun d'eux nous donnerons une description ainsi que sa structure.

4.2.1. Configuration générale de l'atelier

Dans l'atelier, se trouvent d'une part un logiciel *CAD* et un logiciel *CAPP* (*Mecanics*) et d'autre part les équipements suivants : un magasin, deux machines outils à commandes numériques (*la fraiseuse* et le *tour*), un *robot,* un *système de bridage*, un *véhicule optoguidé* (AGV) ainsi que deux *palettiseurs* par machines outils. On trouvera à la figure suivante un plan (sommaire) de la configuration actuelle de l'atelier flexible.

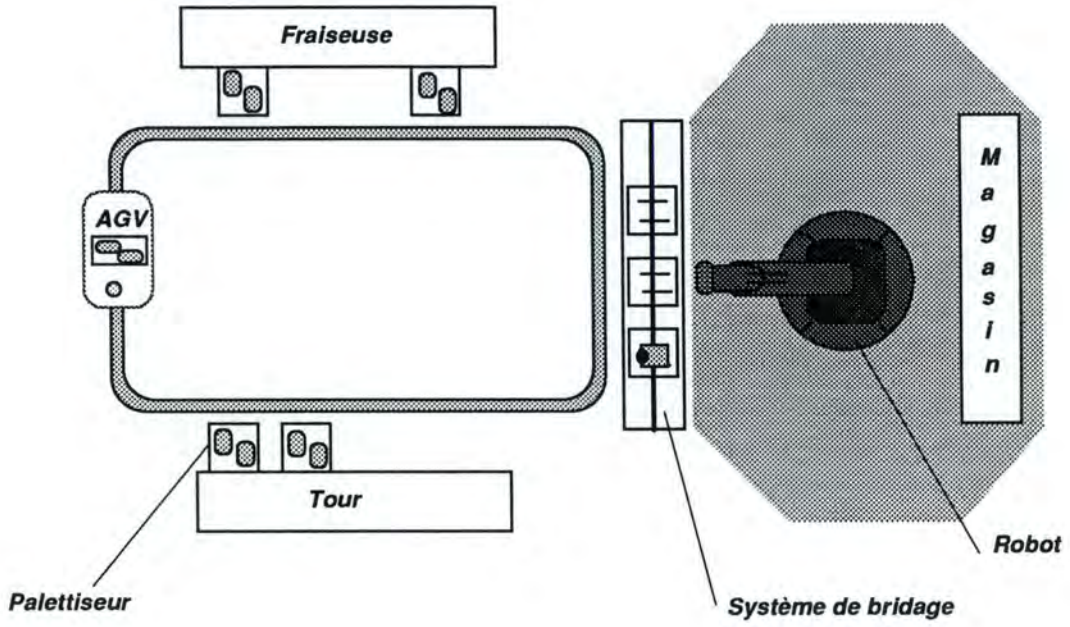


Figure -**4.1**- Plan de l'atelier flexible du CRP-HT de Luxembourg.

Pour pouvoir traiter les pièces, pour pouvoir les transporter, il faut obligatoirement placer ces pièces sur une palette. Ces palettes garantissent la libre circulation des pièces à travers l'atelier.

4.2.2. Les palettes

Signalons, enfin, que tout objet véhiculé par les palettiseurs ou l'AGV doit avoir comme support une palette. Il en existe deux classes :

- les *palettes pour outils* : dans cette classe, nous distinguons également les palettes pour outils destinés à la fraiseuse et celles pour outils destinés au tour car elles disposent d'un mécanisme de fixation différent.

Figure -**4. 2**- Les palettes outils pour la fraiseuse.

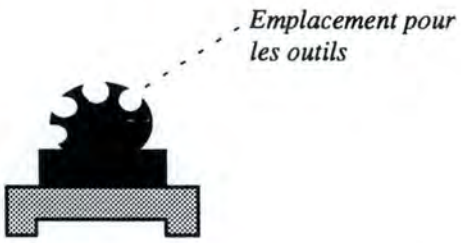


Figure -**4. 3**- Les palettes outils pour le tour.

- les *palettes pour pièces* : elles regroupent les palettes pour les pièces que doit traiter le tour et celles pour les pièces que doit traiter la fraiseuse. Les palettes comprennent en fait deux parties:

- le *socle:* composant commun à toute palette qui lui permet de s'engager dans les rails du palettiseur.

- le *support* proprement-dit: ensemble constitué des portes-outils pour fraiseuse et tour ainsi que des portes-pièces pour fraiseuse (qui sont en fait des étaux) et tour (qui sont de simples supports).

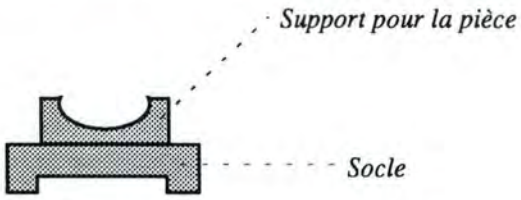


Figure -**4. 4**- Structure d'une palette pièce pour le tour

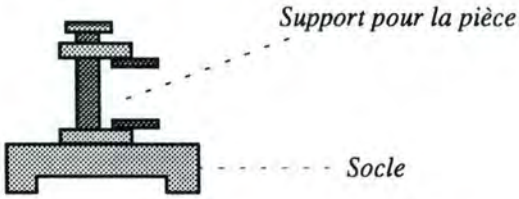


Figure -**4. 5**- Structure d'une palette pièce pour la fraiseuse.

Dans l'état actuel des choses, ces palettes sont préparées manuellement : c'est en fait l'opérateur qui se charge d'assembler socles et supports pour constituer toutes les palettes nécessaires à la production et à placer chacune d'elles aux endroits convenus avant le lancement de la production.

4.2.3. Les palettiseurs

Ils servent de transition entre certaines machines (le *tour* et la *fraiseuse*) et l'*AGV*. Sur les machines, on dispose d'un palettiseur pour les outils (*Palettiseur Outil*) et d'un autre pour les pièces (*Palettiseur Pièce*). Chacun d'eux peut contenir une ou deux palettes simultanément. Il existe également un palettiseur sur l'*AGV*. Toutefois, pour l'*AGV*, nous considérons que le véhicule et le palettiseur constituent un tout, alors que, dans les autres cas, nous considérons davantage les palettiseurs comme une unité de transport indépendante de la machine. En effet, les palettiseurs des machines occupent une position (en termes de coordonnées X,Y,Z) toujours fixe alors que le palettiseur de l'AGV voit sa position évoluer selon le mouvement de l'AGV.

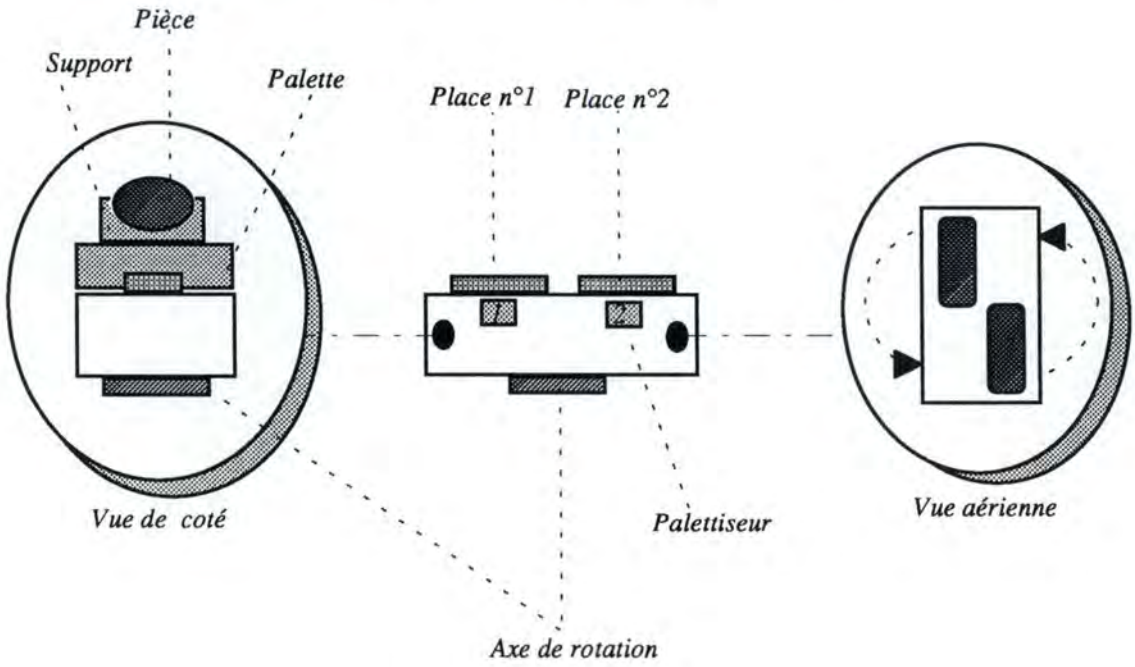


Figure **-4. 6-** Un palettiseur

Mis à part cette distinction, leur fonctionnement est identique, à savoir :

- Pour le chargement d'une pièce à fabriquer :

1. L'AGV se positionne devant le palettiseur et s'y fixe
2. Le palettiseur de l'AGV charge la palette sur le palettiseur de la machine.
3. Le palettiseur Pièce de la machine tourne sur lui-même pour présenter la pièce du coté machine. (sur l'entrée/sortie pièce si il s'agit d'un palette pièce, sur l'entrée/sortie outils dans le cas contraire)
4. L'entrée-sortie Pièce de la machine charge la palette sur la machine.

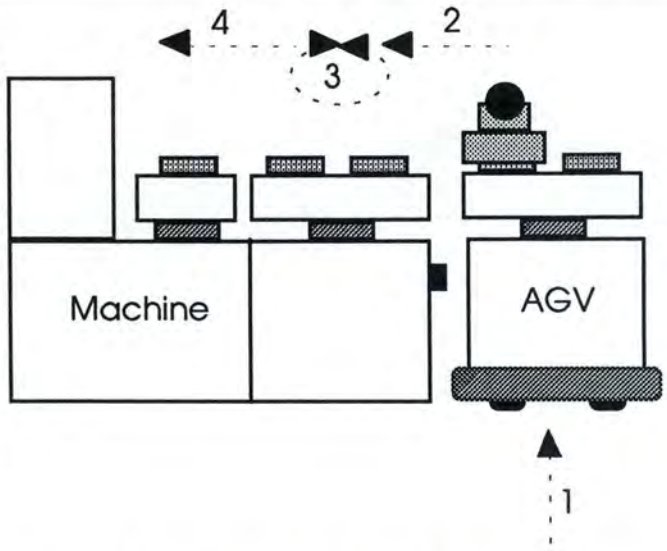


Figure **-4. 7-** Le chargement d'une palette sur une machine

- Pour le déchargement d'une pièce fabriquée :

1. Le palettiseur Pièce transfère la palette de la machine vers lui (pour autant que cette place soit libre).
2. Le palettiseur Pièce de la machine tourne sur lui-même pour présenter la palette du coté AGV.
3. L'AGV se positionne devant le palettiseur Pièce et s'y fixe.
4. Le palettiseur de l'AGV charge la palette sur lui.

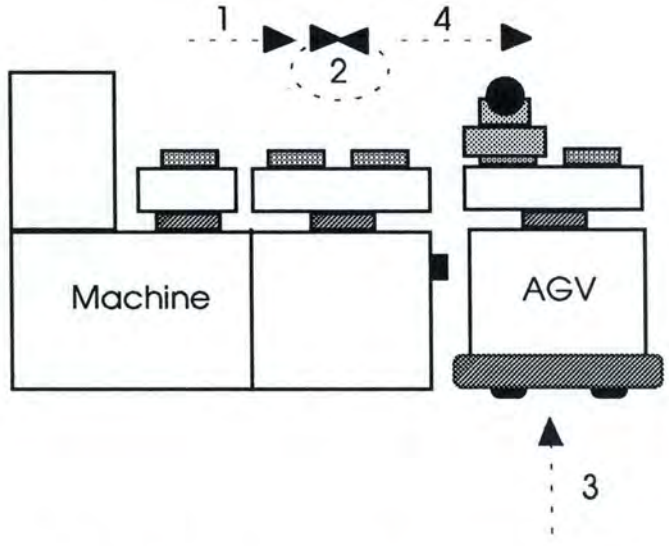


Figure **-4. 8-** Le chargement d'une palette sur l'AGV

4.2.4. Le véhicule optoguidé (Auto Guided Vehicule)

L'AGV suit une piste circulaire tracée sur le sol de l'atelier. Sur cette piste, se trouve une paire de marques par poste particulier (palettiseur outils de la fraiseuse et du tour, palettiseur pièce de la fraiseuse et du tour, emplacement n° 1,2,3 du système de bridage). Ces marques servent de point de repère à l'AGV afin qu'il puisse reconnaître les

endroits où il doit s'arrêter. L'AGV est en mesure de connaître les positions des différents postes existant dans l'atelier grâce à une description de ce dernier stockée sur une disquette introduite dans le véhicule. L'AGV peut transporter une seule ou deux palettes en même temps.

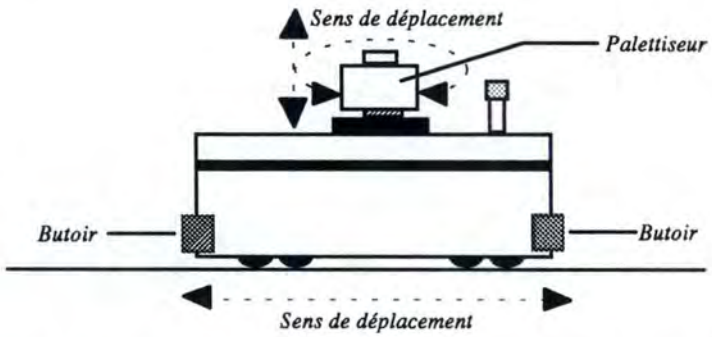


Figure **-4. 9-** Le véhicule opto-guidé *-Auto Guided Vehicule* (Vue de face)

Pour s'immobiliser avec précision devant un poste particulier, il dispose de deux crans qui sont introduits dans deux anneaux, eux-mêmes fixés à ce poste.

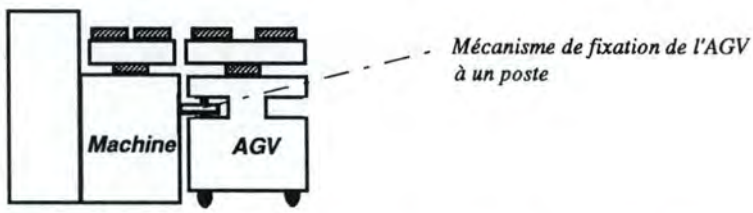


Figure **-4. 10-** Le mécanisme de fixation de l'AGV à un poste

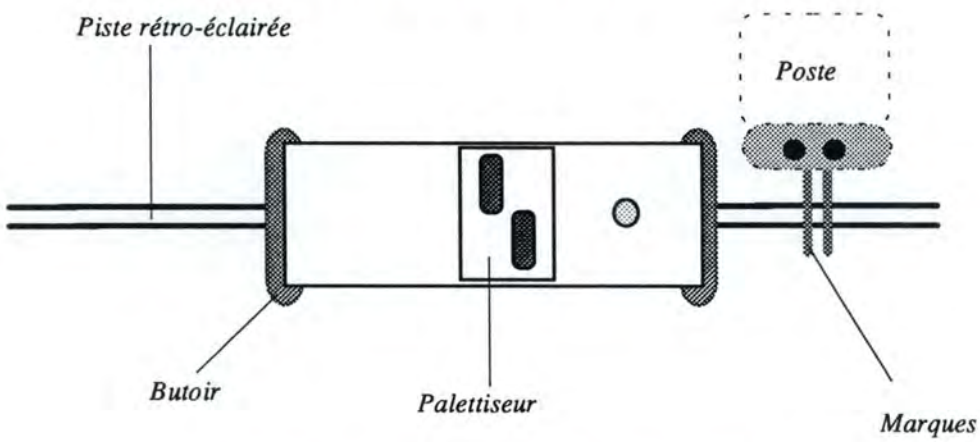


Figure **-4. 11-** Le véhicule opto-guidé (vue aérienne)

4.2.5. Déroulement de la production

Dans cette section, nous allons décrire la démarche actuellement suivie pour produire une nouvelle pièce à partir du moment où l'on reçoit ses spécifications.

4.2.5.1. Conception de la pièce

Avant toute production de pièce, il est nécessaire de disposer d'un plan (dessin) qui la décrit de façon précise. La pièce est dessinée sur un système de CAD (*Computer Aided Design*) qui est ici Me10 ou Me30 fonctionnant sur station de travail Helwett Packard 9000.

Le dessin est fait sur base des spécifications fournies par le client et tient compte des calculs de résistance effectués par l'opérateur. Le résultat de cette étape est un dessin contenant une description géométrique de la pièce.

4.2.5.2. Génération des programmes numériques

Lors de cette étape, l'opérateur détermine, sur base du dessin de pièce obtenu précédemment, les opérations que devront effectuer les machines numériques.

Mecanics permet, d'une part, de définir "manuellement" les étapes nécessaires à la fabrication de la pièce grâce à une série d'instructions et d'autre part de générer les programmes NC (programmes pour les commandes numériques) correspondants destinés aux machines à commandes numériques.

Pour cela, il dispose d'une série d'instructions pour lesquelles il devra définir différents paramètres (vitesse de rotation, outils à utiliser,...).

Par exemple, lors de la réalisation d'une pièce, le concepteur précise les différents endroits où l'on doit forer des trous d'un certain diamètre. Ensuite, sur base de la description de l'étape à réaliser, le système *Mecanics* génère des "micro"-instructions (en langage NC) réalisant cette étape.

Toujours pour l'exemple de l'opération "Forer Trou", on obtient un programme Nc dans lequel on trouve des instructions du style "Déplacer Mandrin", "Ouvrir Mandrin". Cependant, les instructions NC ne sont pas aussi "parlantes", en effet, elles correspondent plus à des instructions en assembleur qu'à des instructions écrites en langage de plus haut niveau.

4.2.5.3. Elaboration du programme pilote

Mecanics génère les programmes NC des machines à commandes numériques. Toutefois, dans la plupart des cas, une machine devra effectuer plusieurs opérations sur une même pièce. Nous disposerons donc de plusieurs programmes NC pour une même machine qu'il faudra charger et lancer à des moments précis. Ceci est réalisé par le programme pilote écrit par l'opérateur. Ce programme charge et lance les programmes NC quand il le faut, mais il commande aussi l'AGV, le système de bridage, le robot et les

palettiseurs. Il veille enfin à coordonner le tout: par exemple, attendre qu'une certaine machine aie bien terminé son travail, avant de lancer une opération sur celle-ci ou sur une autre.

Les différentes machines renvoient des informations concernant la terminaison des pièces mais, en ce qui concerne les erreurs, il n'y a pas à proprement parler d'envoi de codes d'erreur. Il est cependant possible, en consultant certains paramètres (consommation actuelle d'énergie, ...), de détecter une erreur. C'est donc à l'opérateur de se charger du diagnostic.

4.2.5.4. Préparation de la production

Avant de démarrer toute production, il est impératif que l'opérateur place l'AGV à un point 0 (position de départ). Il doit également placer toutes les autres machines dans un état convenu (l'état initial).

De plus, pour que les pièces puissent "circuler" dans l'atelier, elles doivent impérativement être disposées sur des palettes. Celles-ci devront donc être préalablement constituées par assemblage de socles et de supports.

Dans l'état actuel des choses, cette première tâche est à charge de l'opérateur. Il en est de même pour les outils: l'opérateur doit "construire" les différentes palettes mais, en plus, il devra y placer les outils nécessaires à la production de la pièce. Une fois que l'opérateur dispose de toutes les palettes, il les place aux endroits convenus (palettiseurs, AGV, postes du système de bridage) dans le système de pilotage.

Pour le moment, cette préparation se fait manuellement. Cependant, on envisage de créer un stock de palettes et un stock d'outils afin de disposer d'un point d'entrée dans l'atelier.

4.2.5.5. Lancement de la production

Cette étape correspond au lancement effectif du programme de pilotage par l'opérateur.

4.2.5.6. Production et suivi

Traitement entièrement automatisé et assuré par le système de pilotage.

4.3. Construction de la hiérarchie d'agents

4.3.1. Le niveau Atelier

L'atelier constitue ce que A.-W. Scheer [Sch88] appelle une fabrique (*factory*). En effet pour assurer la production des produits demandés par le niveau supérieur (l'entreprise - *enterprise*), l'atelier dispose d'un certain nombre d'équipements qui constituent un groupe d'équipements, chacun réalisant une partie de la production à charge de l'atelier.

Le niveau atelier respecte la structure générique d'un niveau, c'est-à-dire que l'on retrouve les zones fonctionnelles suivantes :

- *la fonction CAD* est jouée par le *CAD_Atelier* : ce composant est chargé de la conception (*design*) des pièces que l'atelier doit produire. Ce composant est, dans notre cas, le logiciel CAD (c'est-à-dire Me10 ou Me30) à l'aide duquel l'opérateur dessine la pièce. Le résultat fourni est une description géométrique de celle-ci.

- *la fonction CAPP* est jouée par le *CAPP_Atelier* : ce composant est chargé de déterminer les différentes étapes de la production d'une pièce (la recette de production). Dans notre cas, ce composant est élaboré grâce au logiciel *Mecanics* (voir description ci-dessus).

- *la fonction PPC* est jouée par le *PPC_Atelier* : ce composant est chargé de planifier la production, de revoir constamment la planification de celle-ci en vue de pouvoir prendre en compte une commande urgente. Ce composant n'existe pas, en tant que tel, dans le cas concret. On le représente toutefois afin de respecter la structure générique. Actuellement, ce composant est réalisé par un programme de pilotage, entièrement programmé par l'opérateur. Toute modification du processus de production, tout changement de produit entraîne une refonte totale du programme de pilotage.

- *la fonction de transport* est jouée par l'*AGV* et l'ensemble des *palettiseurs*. En effet l'AGV permet de transporter les pièces ou les outils entre les différents composants de niveaux N-1 c'est-à-dire entre les différents équipements de l'atelier. Les palettiseurs permettent de transférer des pièces ou des outils depuis l'AGV vers les entrées/Sorties des différents équipements.

- *la fonction d'entrée/sortie* est matérialisée par une partie du *magasin*. En effet, cette partie du magasin permet de stocker des pièces en cours de traitement entre deux étapes de production.

- L'autre partie du *magasin* constitue le *composant de stockage*. En effet, on y stocke les matières premières, les produits semi-finis, les produits finis.

Cependant, il n'est pas nécessaire de représenter le magasin à ce niveau car il fait partie intégrante du niveau palettage.

☞ Voir la remarque faite dans la section concernant l'évaluation de la structure générique dans le chapitre 3.

- *la fonction CAM* est jouée par le *CAM_Atelier* : ce composant est responsable de la fabrication des produits au niveau de l'atelier. Pour cela, il fait appel aux différents équipements dont dispose l'atelier, c'est-à-dire la fraiseuse, le tour, le système de palettage.

Nous représentons à la figure suivante, la structure de notre premier niveau (le *niveau atelier*) de décomposition de l'atelier.

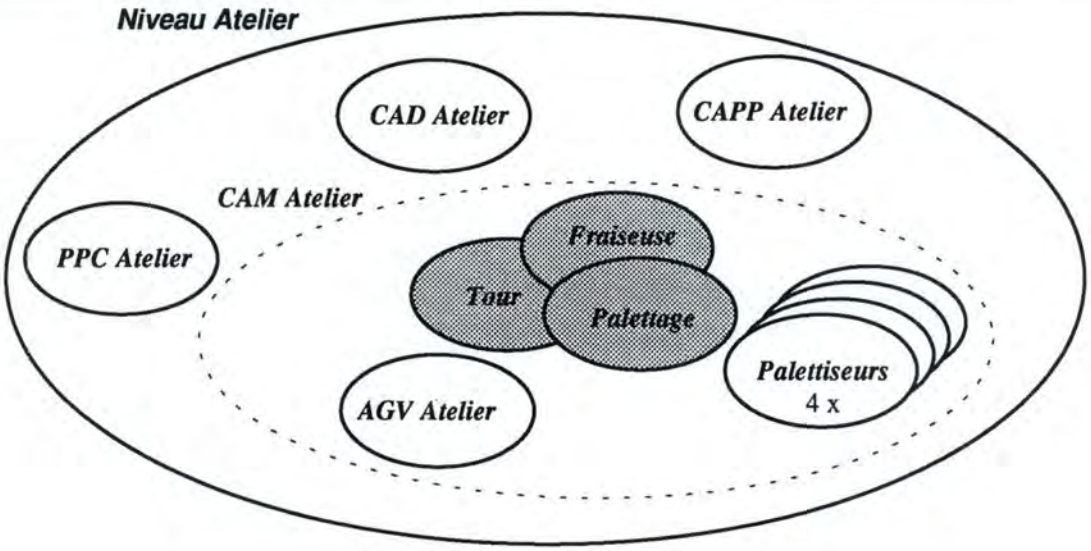


Figure **-4. 12-** La structure du *niveau Atelier*

Dans la figure précédente, on constatera que le composant *CAM Atelier* est représenté par un trait pointillé. Nous adoptons cette convention afin de mettre en évidence que cet agent n'a pas d'existence propre, c'est-à-dire qu'il constitue un regroupement syntaxique des moyens de production du niveau en question.

Les composants qui sont représentés en grisé constituent des niveaux inférieurs, et font donc l'objet d'une description dans les pages qui suivent.

4.3.2. Le niveau Fraiseuse

4.3.2.1. Description

Cette machine outil sert surtout à des travaux de fraisage. Toutefois, avant d'être usinées, les pièces doivent être bridées (attachées) par le système de bridage qui fera l'objet d'une description ci-dessous. Quant aux outils, ils se trouvent sur un support-outils spécial pouvant en contenir plusieurs. Ils seront disposés, par l'opérateur, sur le support-outils qui, lui, devra encore être fixé manuellement sur une palette (voir description ci-dessus) avant toute utilisation. On envisage toutefois de créer un stock d'outils grâce auquel il serait possible d'automatiser cette préparation des outils.

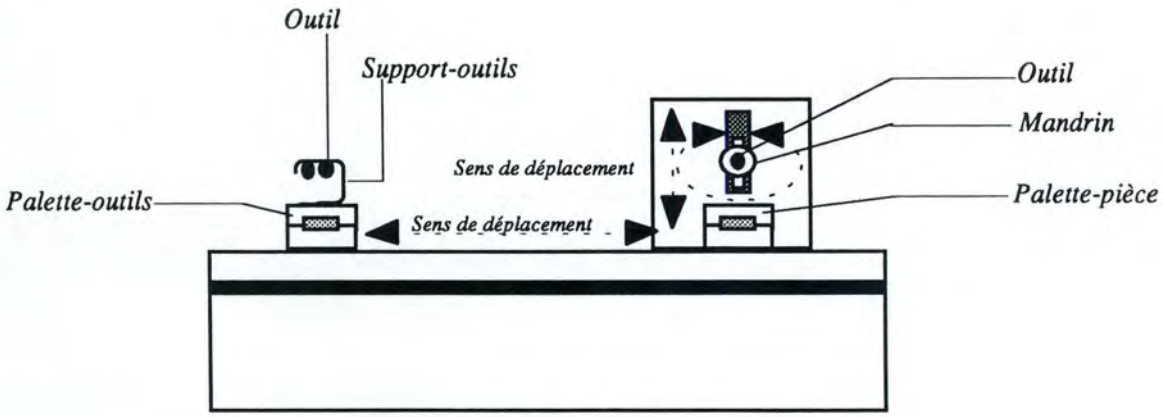


Figure **-4. 13-** La fraiseuse (vue de face)

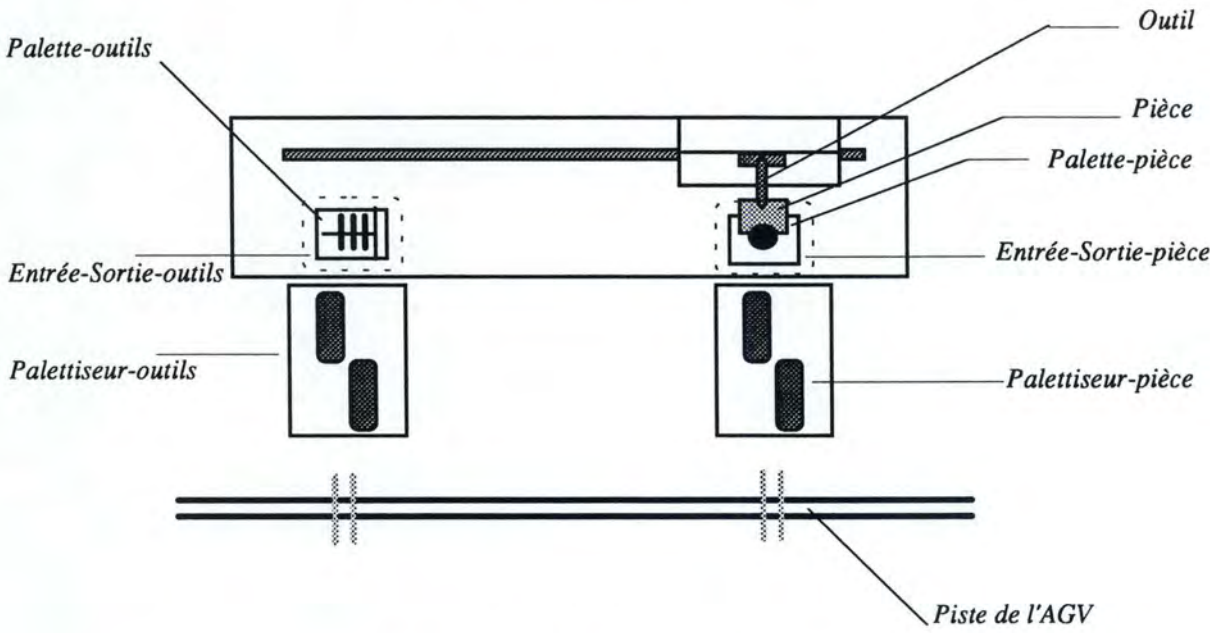


Figure **-4. 14-** La Fraiseuse (vue aérienne)

4.3.2.2. Structure

L'analyse de ce niveau révèle que la structure générique à appliquer correspond à la structure générique d'un **niveau terminal** car le système de fraiseuse est constitué d'un seul équipement, en l'occurrence une fraiseuse (**équipement simple**).

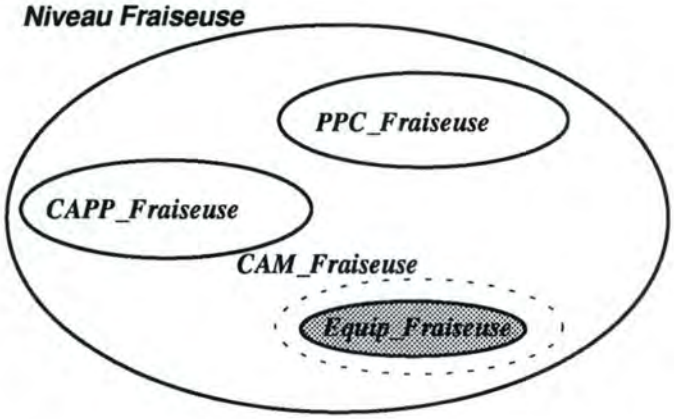


Figure **-4. 15-** Structure du *niveau Fraiseuse*

Le *niveau_Fraiseuse* respecte la structure d'un **niveau terminal** à la différence près qu'il n'y a pas de zone fonctionnelle *CAD* car le niveau_Fraiseuse n'est pas responsable de la conception de pièces. La fonction *CAD* du niveau_Fraiseuse est donc reportée au niveau supérieur, c'est-à-dire le *niveau_Atelier.*

La zone fonctionnelle *CAM_Fraiseuse* est assurée par l'équipement simple *Equip_Fraiseuse* qui est constitué de la *fraiseuse* proprement dite, ainsi que de deux *entrées/sorties* : une réservée aux pièces, l'autre pour les outils.

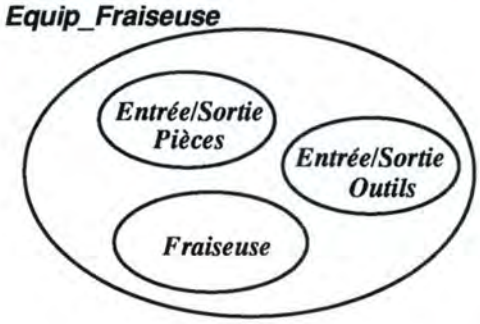


Figure **-4. 16-** Structure de l'équipement simple *Equip_Fraiseuse*

4.3.3. Le niveau Tour

4.3.3.1. Description

Cette machine sert à des travaux de tournage. Contrairement à la fraiseuse, les pièces arrivent, au tour, sans être bridées. Une fois arrivée sur la machine, la pièce est prise par un bras (appelé robot portique) qui place la pièce dans le mandrin pour ensuite être fixée. La préparation des outils est identique à celle de la fraiseuse.

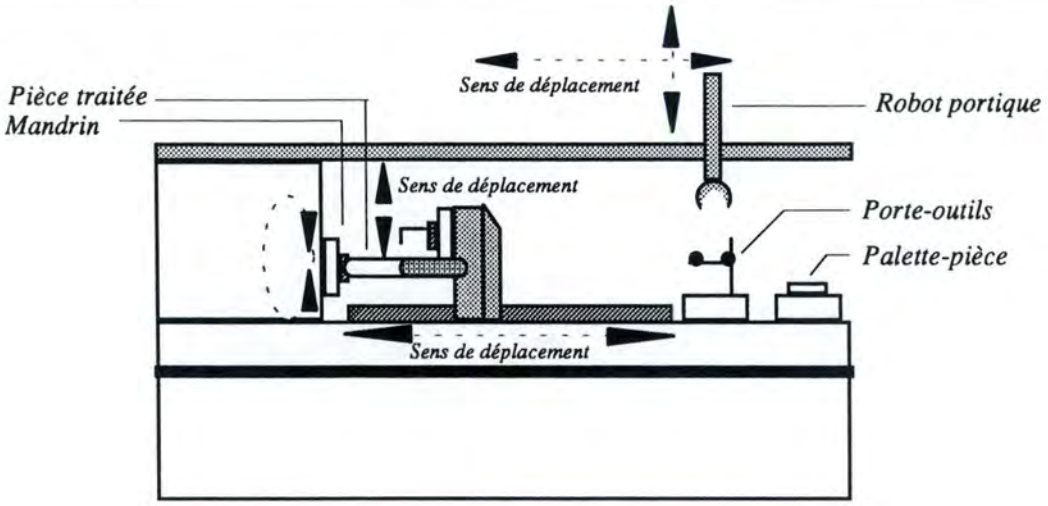


Figure **-4. 17-** Le tour (vue de face)

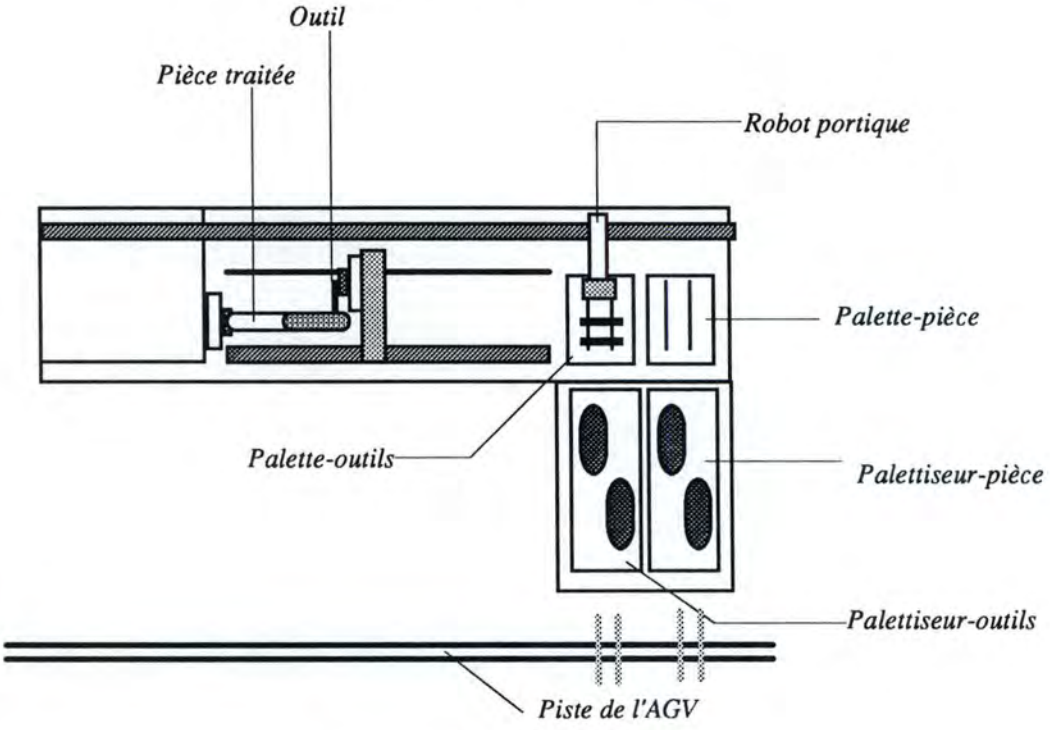


Figure **-4. 18-** Le tour (vue aérienne)

4.3.3.2. Structure du tour

L'analyse de ce niveau révèle que la structure générique à appliquer correspond à la structure générique d'un **niveau terminal** car le système de tour est constitué d'un seul équipement, en l'occurrence un tour (**équipement simple**).

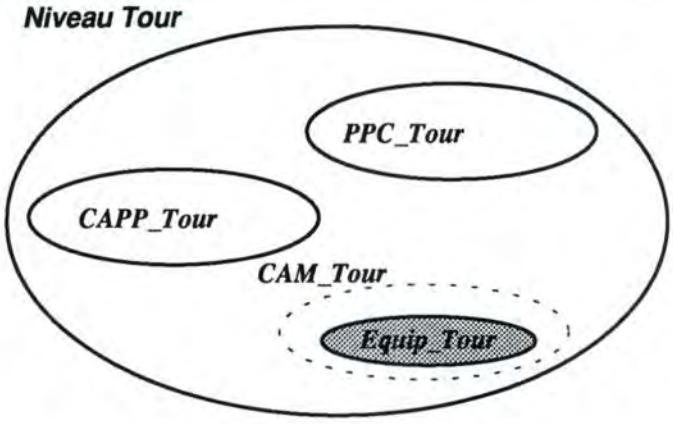


Figure **-4. 19-** Structure du *niveau Tour*

Le *niveau_Tour* respecte la structure d'un **niveau terminal** à la différence près qu'il n'y a pas de zone fonctionnelle *CAD* car le niveau_Tour n'est pas responsable de la conception de pièces. La fonction *CAD* du niveau_Tour est donc reportée au niveau supérieur, c'est-à-dire le *niveau_Atelier.*

La zone fonctionnelle *CAM_Tour* est jouée par l'équipement simple *Equip_Tour* qui, lui, est constitué du *tour* proprement dit, le *robot portique*, ainsi que de deux *entrées/sorties*, une réservée aux pièces, l'autre pour les outils.

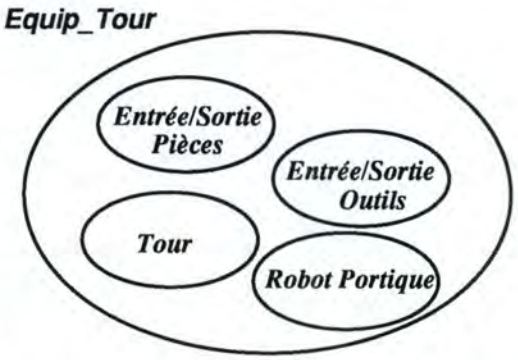


Figure **-4. 20-** Structure de l'équipement simple *Equip_Tour*

4.3.4. Le niveau Palettage

Nous avons déjà signalé que, pour être traitées dans l'atelier, il est nécessaire que les pièces soient bridées ou déposées sur des palettes, selon la machine chargée d'effectuer l'opération.

L'opération de mise en place de la pièce sur la palette porte le nom de palettage. On peut considérer que cette opération constitue un tout nécessaire à la réalisation d'une opération au sein de l'atelier. On conviendra donc de l'existence d'un niveau de palettage qui sera responsable envers le niveau atelier de la réalisation des opérations de bridage, de débridage ainsi que du stockage des pièces.

Le *niveau Palettage* respecte bien la structure d'un niveau générique car on y retrouve les zones fonctionnelles suivantes :

- *la fonction CAPP* est jouée par le *CAPP_Palettage*: ce composant est chargé de déterminer les différentes étapes nécessaires à l'opération de palettage d'une pièce donnée sur une palette (la recette du palettage).

- *la fonction PPC* est jouée par le *PPC_Palettage* : ce composant est chargé de planifier les opérations de palettage, de revoir constamment la planification de celles-ci en vue de pouvoir prendre en compte une commande urgente de palettage reçue du niveau supérieur, c'est-à-dire le *niveau atelier*.

- *la fonction de transport* est réalisée par le *robot*. En effet le robot permet de transporter les pièces depuis les emplacements de stockage vers les postes de bridage et inversement. Le robot est également utilisé pour retourner une pièce.

Le robot sert de transition entre le magasin et le système de bridage. Il prend les pièces (situées sur le support en bois) dans le magasin et les dépose sur une palette d'un des postes du système de bridage qui sera ensuite transférée sur l'AGV et inversement. Le robot est également utilisé pour un changement de position de la pièce dans l'étau (d'une palette), après que celle-ci ait été débridée. Il doit, pour prendre une pièce, disposer d'une "procédure" décrivant la méthode d'approche, la manière de la saisir, etc. Dans le futur on peut imaginer un robot supplémentaire, pour l'assemblage et/ou pour le contrôle de qualité (qui présenterait la pièce sous ses différentes facettes à un système de vision).

- *la fonction d'entrée/sortie* et *de stockage* est jouée par un seul agent qui est le magasin. Cependant, une partie du magasin (*Zone_E/S_Magasin*) permet de stocker des pièces en cours de traitement entre deux étapes de production (**entrée/sortie**). L'autre partie (*Zone_Stock_Magasin*) permet de stocker les matières premières, les produits semi-finis, les produits finis (**stockage**).

Le magasin est actuellement une simple étagère dans laquelle des emplacements pour les pièces ont été définis. Ces emplacements sont en fait des supports en bois qui contiennent des matières premières, des produits semi-finis et des produits finis. Signalons que la gestion de ce magasin est entièrement prise en charge par le système de pilotage (voir description ci-dessus), alors qu'idéalement elle serait à charge d'un système de gestion autonome. En effet, on peut trouver, de nos jours, des systèmes de stockage extrêmement complexes qui englobent le stockage physique des pièces mais également la gestion du stock.

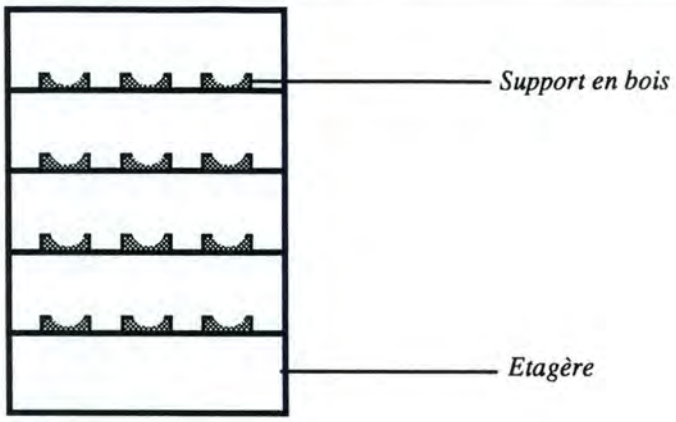


Figure -**4. 21**- Le magasin

• *la fonction CAM* est jouée par le *CAM_Palettage* : ce composant est responsable de la fabrication des produits au *niveau palettage*. Pour cela, il fait appel aux différents équipements dont dispose la cellule de palettage, c'est-à-dire le robot, le moyen de stockage et le système de bridage.

Nous représentons à la figure suivante, la structure du *niveau Palettage*.

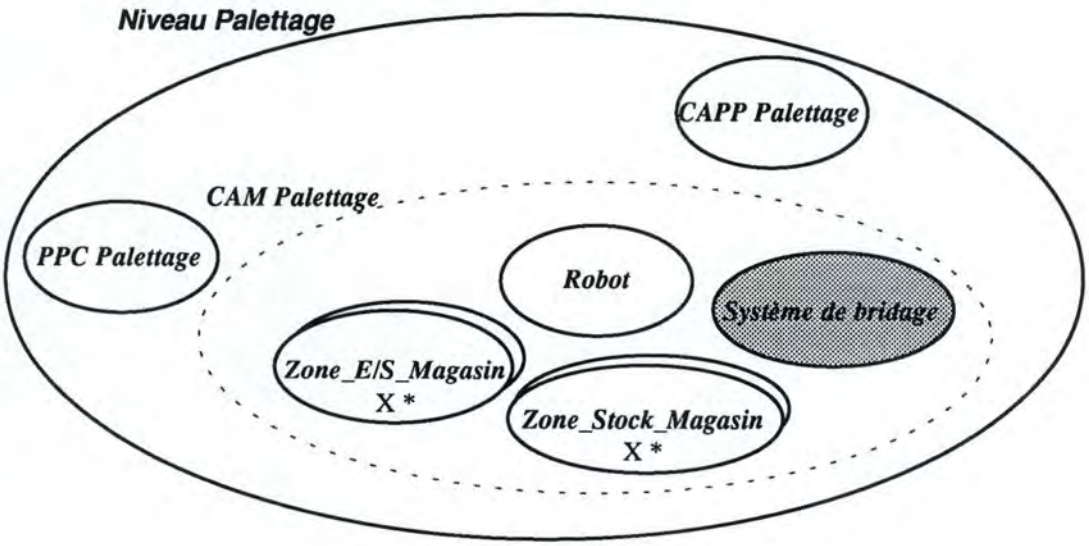


Figure -**4. 22**- La structure du *niveau Palettage*

On peut remarquer que le composant *CAD Palettage* n'est pas présent. En effet, le niveau Palettage n'est pas responsable de la conception de pièces, donc la fonction *CAD* qui devrait être réalisée à ce niveau est reportée au niveau supérieur, c'est-à-dire le *niveau Atelier*.

Le *robot* est l'agent chargé du transport au *niveau Palettage*. Dans le cas actuel, le *robot* n'effectue que des déplacements de pièces. Il joue donc bien un rôle de moyen de transport. Cependant, on pourrait imaginer un robot supplémentaire qui soit chargé de l'assemblage de pièces. Dans notre cas, comme on l'a exposé dans le chapitre 3 (portant

sur la structure générique), il est nécessaire d'avoir un moyen de transport au sein d'un niveau, sous peine de rendre le niveau fermé.

4.3.5. Le niveau Système-de-bridage

4.3.5.1. Description

Le bridage est une opération qui consiste à fixer une pièce dans un étau. Ce système est constitué de trois emplacements (postes de bridage) sur lesquels sont déposées des palettes. Parmi ces emplacements, un seul est, actuellement, utilisé pour le bridage proprement dit, tandis que les deux autres servent de lieu de transit entre le *robot* et l'*AGV*. Actuellement, le bridage d'une pièce ne se fait que sur un seul emplacement et toujours à un même endroit sur la palette. Il est toutefois prévu d'ajouter à ce système deux axes supplémentaires afin de pouvoir brider les pièces sur les trois emplacements et à des endroits différents sur la palette.

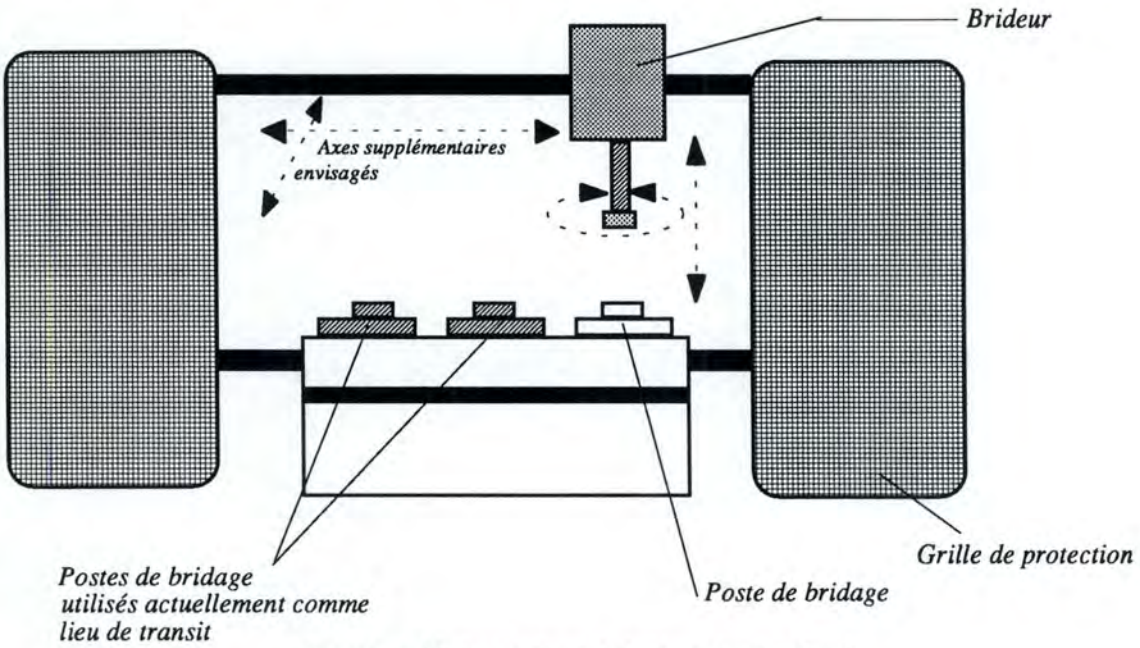


Figure **-4. 23-** Le système de bridage (vue de face)

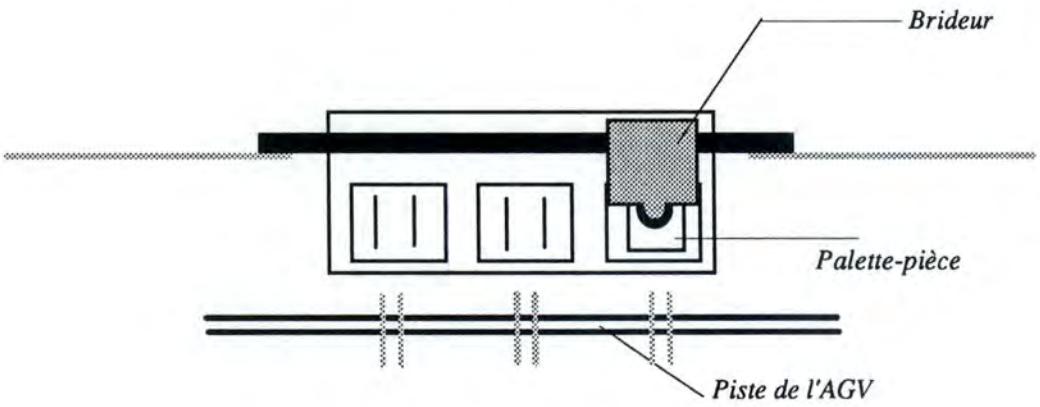


Figure **-4. 24-** Le système de bridage (vue aérienne)

4.3.5.2. Structure

L'analyse de ce niveau révèle que la structure générique à appliquer correspond à la structure générique d'un **niveau terminal** car le système de bridage est constitué d'un seul équipement, en l'occurrence un brideur (**équipement simple**).

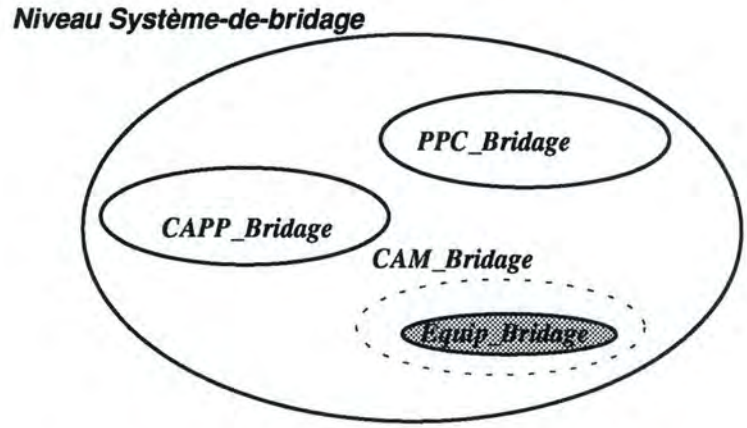


Figure -**4. 25**- Structure du *niveau Système-de-Bridage*

Le *niveau_Système-de-bridage* respecte la structure d'un **niveau terminal** à la différence près qu'il n'y a pas de zone fonctionnelle *CAD* car le niveau_Système-de-bridage n'est pas responsable de la conception de pièces. La fonction *CAD* du niveau_Système-de-bridage est donc reportée au niveau supérieur, c'est-à-dire le *niveau_Palettage.*

La zone fonctionnelle *CAM_Bridage* est jouée par l'équipement simple *Equip_Bridage* est constitué de trois *entrées/sorties* destinées à recevoir les pièces qui doivent être bridées (fixées) sur les palettes et du *brideur* proprement dit.

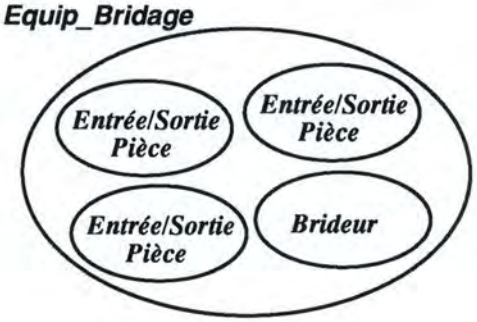


Figure -**4. 26**- Structure de l'équipement simple *Equip_Bridage*

4.3.6. Hiérarchie des agents de l'atelier

La hiérarchie que l'on propose pour la structuration de l'atelier constitue une architecture à mi-chemin entre la structure existante et la structure idéale. En effet, beaucoup de zones fonctionnelles seront, lors de la conception, reportées aux niveaux supérieurs.

Mais, actuellement, il nous semble intéressant de disposer d'une structure intermédiaire, et aussi proche que possible de la structure générique afin de mieux maîtriser les informations disponibles (savoir qui en est responsable, savoir à qui est destinée telle ou telle information).

Cependant, pour certains aspects, nous avons déjà franchi un pas supplémentaire. En effet, pour ce qui en est de la zone fonctionnelle *CAD*, on ne la retrouve qu'au niveau atelier, alors qu'idéalement on devrait la retrouver à chaque niveau. Etant donné que dans le cas concret, il n'y a aucune autre conception que celle qui a lieu au niveau atelier, il ne nous a pas semblé utile de surcharger la structure avec des composants dont on est déjà certain, au moment de l'étude des besoins, qu'ils ne sont pas utiles.

On trouvera à la figure suivante la hiérarchie que nous proposons pour l'atelier flexible du CRP-HT de Luxembourg.

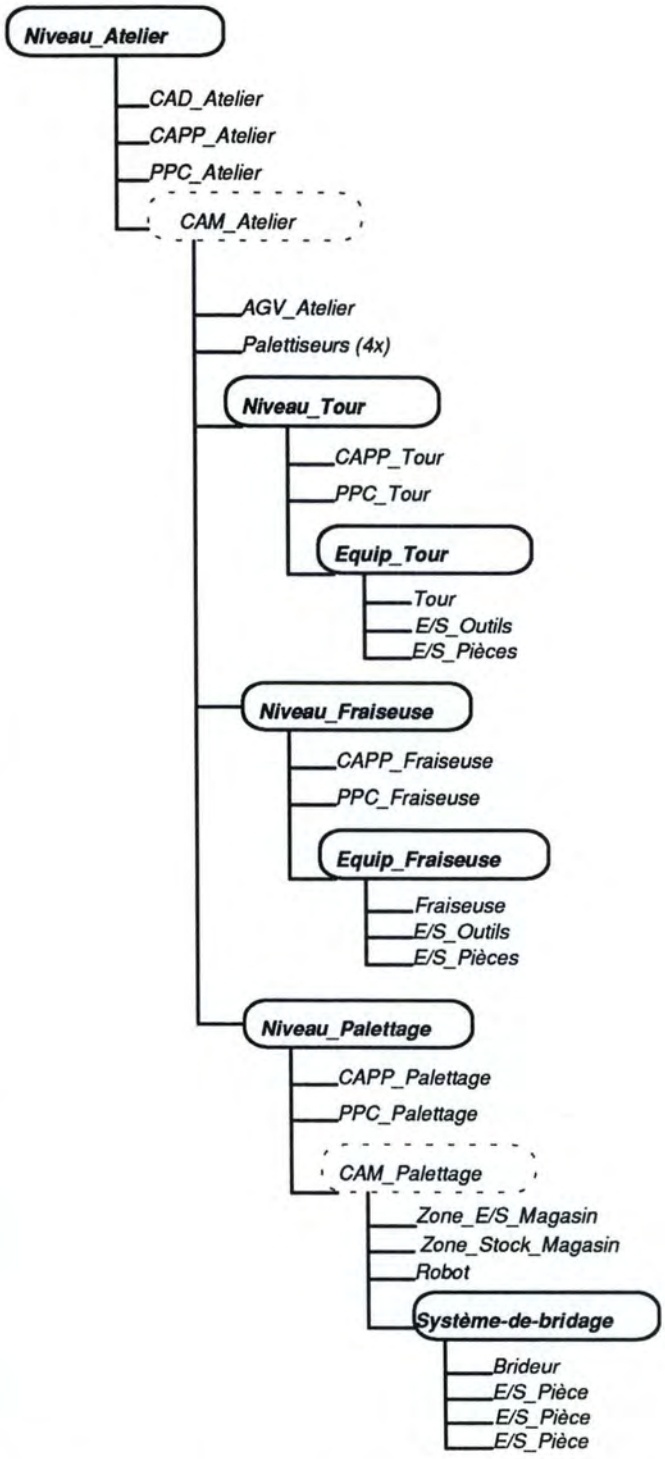


Figure -**4. 27**- Hiérarchie des agents de l'atelier flexible

4.4. Construction de la structure d'état des agents

Dans cette partie, nous nous proposons de construire les structures d'état des agents constituant l'atelier flexible. Cependant, par manque de place, nous ne donnerons dans ce travail que la structure d'un seul agent, en l'occurrence le composant *AGV* chargé du *transport* des pièces dans l'atelier. Pour être cohérent avec ce qui va suivre, nous désignerons l'*AGV* par *Système_AGV*. Le lecteur intéressé par la structure des autres agents pourra les trouver dans les annexes de ce travail (annexe A).

Pour construire la structure d'état de l'agent chargé du transport, nous allons mettre en oeuvre le mécanisme de "raffinements successifs" (*refinement*). Tout comme pour la construction de la hiérarchie, nous nous basons sur la structure des composants génériques proposés par Michaël Petit [Pet92].

4.4.1. Couche n° 1 : Identification de l'agent

Nous avons l'agent *Système_AGV* à représenter : pour cela nous utilisons le formalisme introduit dans le chapitre 2 (concernant le langage ALBERT). A l'issue de cette étape, nous disposons donc du parallélogramme destiné à représenter l'état du *Système_AGV*.

4.4.2. Couche n° 2 : Identification des Entités/Associations

- *Système_AGV*

  ■ Description : Contient un descriptif du Système_AGV. On a besoin de connaître les caractéristiques générales véhicule
  ■ Type : **VEHICULE**
  ■ Lien : ∅
  ■ Visible par : *PPC_ATELIER, CAPP_ATELIER*
  ■ Structure : ☞ Annexe C

- *Périodes_indisponibilité*

  ■ Description : Pour assurer une planification des travaux au sein de l'atelier, nous devons être en mesure de connaître les périodes pendant lesquelles le Système_AGV est indisponible pour des raisons telles qu'une panne, un entretien.
  ■ Type : **PERIODE**
  ■ Lien : ∅
  ■ Visible par : *PPC_ATELIER*
  ■ Structure : ☞ Annexe C

- *Postes_accessibles*

  ■ Description Le Système_AGV assure le transport de pièces entre les différents postes auxquels il a accès
  ■ Type **POSTE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER, CAPP_ATELIER*
  ■ Structure ☞ Annexe C

- *Batteries_possibles*

  ■ Description Contient les informations au sujet des batteries qu'accepte le Système_AGV.
  ■ Type **BATTERIE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER*
  ■ Structure ☞ Annexe C

- *Opérations_type*

  ■ Description Ces opérations possibles correspondent à la réalisation effective d'une opération type telle que "se déplacer". Contient les informations sur les opérations générales que le Système_AGV peut effectuer. Exemples: Chargement, déplacement,...
  ■ Type **OP_TYPE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER, CAPP_ATELIER*
  ■ Structure ☞ Annexe C

- *Places*

  ■ Description Contient une description des places associées au Système_AGV.
  ■ Type **PLACE**
  ■ Lien • **Acceptation** : Permet de déterminer les objets que la place peut accepter.
  • **Accueil** : Détermine le contenu effectif de chaque place à un moment donné.
  ■ Visible par *PPC_ATELIER, CAPP_ATELIER*
  ■ Structure ☞ Annexe C

↳ Ces places peuvent accepter un certain nombre de contenus possibles (*Contenus_possibles_place*). A un moment donné, la place de l'*AGV* accueille un objet (*Contenu_effectif_place*).

- *Contenus_possibles_place*

■ Description Contient un descriptif des types d'objets.
■ Type **T_OBJET**
■ Lien • **Acceptation** : Voir ci-dessus.
■ Visible par *PPC_ATELIER, CAPP_ATELIER*
■ Structure ☞ Annexe C

- *Contenu_effectif_place*

■ Description Contient un descriptif d'un objet situé sur une des places..
■ Type **OBJET**
■ Lien • **Accueil** : Voir ci-dessus.
■ Visible par *PPC_ATELIER*
■ Structure ☞ Annexe C

↳ On désire connaître l'opération en cours de réalisation (*Opération_en_cours*) ainsi que la position occupée (*Position_actuelle*) et la charge de la batterie effective (*Batterie_effective*).

- *Opération_en_cours*

■ Description Contient une série d'informations sur l'opération exécutée par le Système_AGV.
■ Type **OPERATION_EXEC**
■ Lien ∅
■ Visible par *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Position_actuelle*

■ Description Indique le poste qu'occupe le Système_AGV.
■ Type **POSITION**
■ Lien ∅
■ Visible par *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Batterie_effective*

■ Description Contient des informations sur les batteries utilisées par le Système_AGV.
■ Type **BAT_EFF**
■ Lien ∅
■ Visible par *PPC_ATELIER*
■ Structure ☞ Annexe C

4.4.3. Couche n°3 : Visibilité

Sous cette rubrique nous regroupons les actions qui doivent être visibles à l'extérieur de l'agent. Nous allons donc exprimer ce que l'agent **exporte** vers son environnement. Normalement, nous devrions également reprendre les entités et associations qui sont visibles par l'environnement. Cependant, par manque de place, nous avons regroupé ces informations dans la couche n°2, lorsque nous identifions les entités et associations.

- *Fin_ordre_opération*

■ Description Signale la fin de l'exécution d'une opération.
■ Type **FIN_ORDRE_OP**
■ Destination *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Erreurs_ordre_opération*

■ Description Signale une erreur dans l'exécution d'une opération.
■ Type **ERREUR_ORDRE_OP**
■ Destination *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Problèmes_batterie*

↳ Une fois que le taux de charge de la batterie a atteint un niveau critique défini, il convient de le signaler afin de procéder au remplacement de celle-ci (*Problèmes_batterie*).

■ Description Signale les problèmes de batterie.
■ Type **PBM_BAT**
■ Destination *PPC_ATELIER*
■ Structure ☞ Annexe C

4.4.4. Couche n° 4 : Perception

Lors de cette étape, nous cherchons à distinguer parmi l'ensemble des informations connues de l'*AGV*, celles qui proviennent de l'environnement de l'AGV. Ici, également, la perception des informations ne se limite pas aux seules actions, mais les agents peuvent également percevoir des informations sous forme d'entités et/ou associations.

- *Nv_périodes_indisponibles*

■ Description Signale une nouvelle période d'indisponibilité pour cet équipement.
■ Type **N_PERIODE**
■ Origine *GESTIONNAIRE*
■ Structure ☞ Annexe C

- *Ordre_opération*

■ Description Signale qu'un ordre d'opération est demandé au Système_AGV.
■ Type **ORDRE_OP**
■ Origine *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Arrêt_ordre_opération*

■ Description Signale que l'opération en cours doit être arrêtée.
■ Type **STOP_OP**
■ Origine *PPC_ATELIER*
■ Structure ☞ Annexe C

- *Nv_batterie_poss*

■ Description Signale au Système_AGV qu'il peut désormais accepter une nouvelle batterie.
■ Type **N_BATTERIE**
■ Origine *PPC_Atelier*
■ Structure ☞ Annexe C

- *Nv_batterie_effective*

■ Description Signale au Système_AGV que l'on vient de changer de batterie.
■ Type **N_BATTERIE_EFF**
■ Origine *PPC_Atelier*
■ Structure ☞ Annexe C

- *Nv_opération_type*

■ Description Signale à l'AGV qu'il peut désormais effectuer une nouvelle opération type.
■ Type **N_OP_TYPE**
■ Origine *CAPP_ATELIER*
■ Structure ☞ Annexe C

- *Nv_poste_acces*

■ Description Signale au Système_AGV qu'il y a un nouveau poste à prendre en considération
■ Type **N_POSTE**
■ Origine *CAPP_ATELIER, GESTIONNAIRE*
■ Structure ☞ Annexe C

- *Nv_contenu_possible*

  ■ Description Signale au Système_AGV que, pour une place donnée, elle peut accepter désormais un nouveau contenu possible.
  ■ Type **N_CONT_POSS**
  ■ Origine *CAPP_ATELIER*
  ■ Structure ☞ Annexe C

- *Nv_contenu_effectif*

  ■ Description Signale au Système_AGV que, pour une place donnée, elle accepte pour le moment un contenu effectif donné.
  ■ Type **N_CONT_EFF**
  ■ Origine *PPC_ATELIER*
  ■ Structure ☞ Annexe C

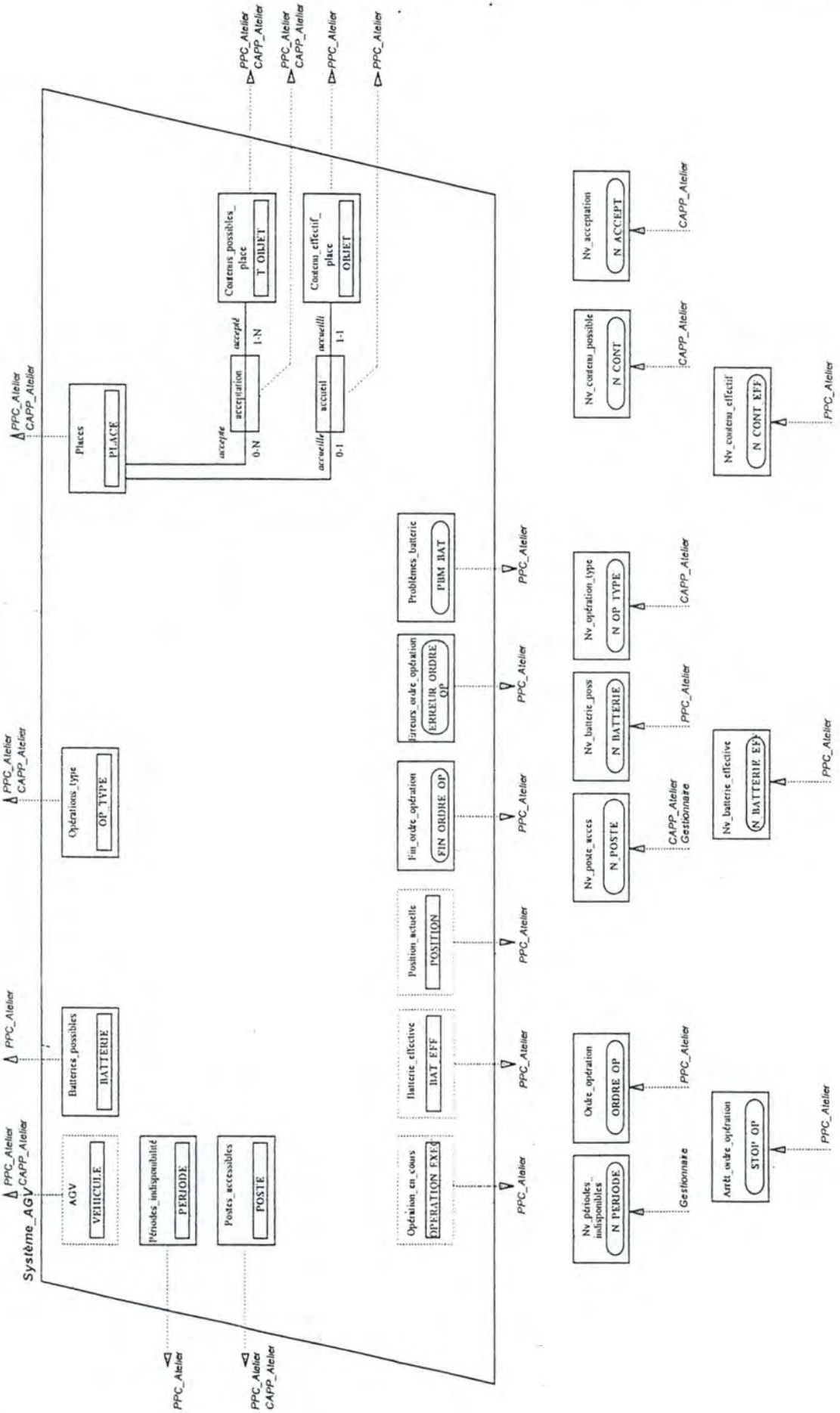
- *Nv_acceptation*

  ■ Description Permet d'enregistrer une nouvelle acceptation, pour une place, d'un objet
  ■ Type **N_ACCEPT**
  ■ Origine *CAPP_ATELIER*
  ■ Structure ☞ Annexe C

Nous donnons à la figure suivante, la structure d'état de l'agent *AGV*. La structure que l'on en donne est la structure idéale, c'est-à-dire celle qui est la plus proche possible de la structure générique des composants de transport donnée par Michaël Petit [Pet92]. A cette structure, s'ajoute également les différentes structures des types de données (par exemple, la structure du type de données Equipement). Le lecteur intéressé par ces structures de données pourra les trouver dans l'Annexe C de ce travail.

Nous verrons, dans le chapitre suivant, qu'il est nécessaire de "*modeler*" quelque peu la structure générique afin que celle-ci représente le plus fidèlement possible la réalité de l'entreprise productique modélisée.

On trouvera à la figure suivante la structure d'état de l'*AGV*.


Système_AGV
PPC_Atelier
CAPP_Atelier
AGV
VEHICULE
Batteries_possibles
BATTERIE
Périodes_indisponibilité
PERIODE
Postes_accessibles
POSTE
Opérations_type
OP_TYPE
Places
PLACE
Contenus_possibles_place
T_OBJET
Contenu_effectif_place
OBJET
acceptation
accueil
accepté
accueilli
0-N
1-N
0-1
1-1
Opération_en_cours
OPERATION_EXE
Batterie_effective
BAT_EFF
Position_actuelle
POSITION
Fin_ordre_opération
FIN ORDRE OP
Erreurs_ordre_opération
ERREUR_ORDRE_OP
Problèmes_batterie
PBM BAT
Nv_périodes_indisponibles
N PERIODE
Ordre_opération
ORDRE OP
Nv_poste_accès
N POSTE
Nv_batterie_poss
N BATTERIE
Nv_opération_type
N OP TYPE
Nv_batterie_effective
N BATTERIE EFF
Arrêt_ordre_opération
STOP OP
Nv_contenu_possible
N CONT
Nv_contenu_effectif
N CONT EFF
Nv_acceptation
N ACCEPT
Gestionnaire
CAPP_Atelier
Gestionnaire

Chapitre 5

Spécification complète d'un agent de transport : le système AGV

5.1. Introduction

Dans le chapitre précédent, nous avons présenté une déclaration en langage ALBERT de l'ensemble de l'atelier étudié au Centre Henri Tudor.

Nous allons passer maintenant à une spécification complète en ALBERT d'un sous-ensemble de cet atelier en l'occurence l'AGV ou plutôt ce que nous appelons le système AGV (cfr.: partie non grisée de la figure 5.1). En fait, nous préférons cette dernière appelation à celle d'AGV car, au cours de notre travail de spécification, nous avons constaté que d'autres composants s'ajoutaient au véhicule proprement-dit.

Dans ce chapitre, nous allons décrire informellement le système AGV. Nous présenterons, ensuite, la modélisation de ce système, qui comprend la hiérarchisation et la spécification des composants définis ainsi que l'expression des contraintes au sujet de ces composants.

Signalons au lecteur que cette partie du travail s'insère dans un projet du CRP-HT dont l'objectif principal est de concevoir un rapport au niveau conceptuel. Ce rapport doit refléter, au mieux, l'activité de l'atelier et constitue la première étape d'un projet global visant à l'élaboration d'un système de monitoring (but à court terme) et ensuite à une intégration complète de l'atelier (but à long terme). Par conséquent, dans cette modélisation, nous nous efforçons principalement à décrire la structure de chaque agent et ce, afin de réaliser au plus vite le système de monitoring. Nous avons bien entendu essayé d'être les plus exhaustifs, en gardant constamment à l'esprit que notre travail servirait à une intégration complète de l'atelier. Toutefois, nous sommes obligés de constater que certains éléments ont été (volontairement ou involontairement) omis, bien que nécessaires à la réalisation de l'objectif à long terme.

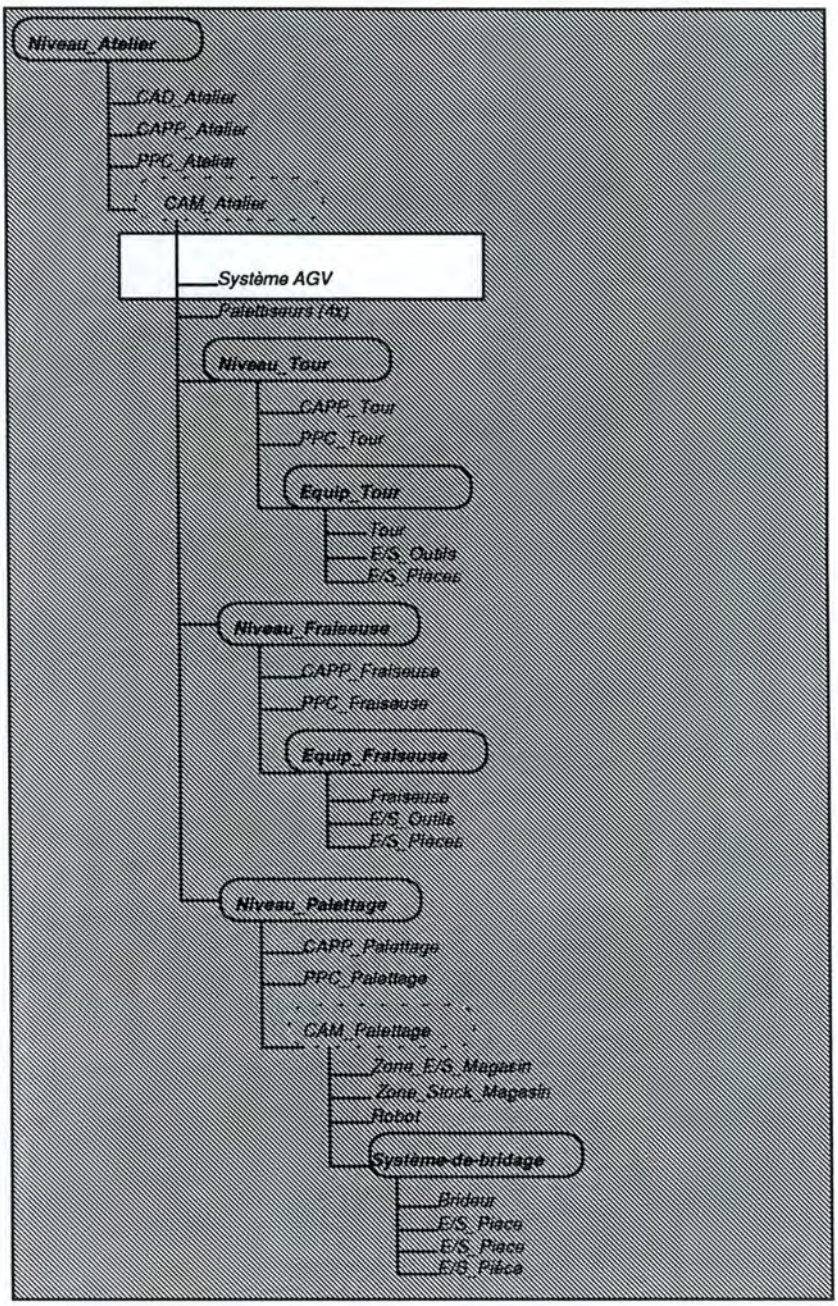


Figure **-5.1-** Objet de la modélisation.

5.2. Description informelle du système AGV

L'objet de cette partie du mémoire est de présenter le plus précisément possible l'ensemble des informations au sujet du moyen de transport qu'il nous semble bon que d'autres agents de l'atelier (CAPP, PPC et gestionnaire) puissent connaître, ainsi que les services qu'il devrait leur offrir afin de pouvoir réaliser les objectifs (à court et long terme) définis dans le cadre du projet du CRP-HT.

Le système AGV imaginé peut réaliser un certain nombre d'opérations types considérées comme de haut niveau. Ce sont les opérations de déplacement vers un poste (*Move*), de chargement sur une place du palettiseur à partir d'un poste (*Take*), de déchargement d'une place vers un poste (*Put*) et d'attente pendant un certain temps (*Wait*). Celles-ci sont connues du PPC ainsi que du CAPP de l'atelier et caractérisées par une description, une estimation de la durée théorique d'exécution et une série de paramètres.

Il a aussi la possibilité d'accéder à des postes dont le CAPP et le PPC connaissent l'identifiant, la position et le type, indiquant si le moyen de transport peut y effectuer un chargement/déchargement ou s'il s'agit d'un simple point d'arrêt.

Il comprend un palettiseur dont les places (surveillées par un détecteur) accueillent au plus un objet qui appartient impérativement à l'ensemble des types d'objets acceptés par cette place.

Le PPC dispose d'informations sur les batteries acceptées par le moyen de transport et éventuellement des informations sur :

- les périodes au cours desquelles il ne pourra fonctionner.
- l'opération de haut niveau en cours d'exécution.
- les batteries réellement branchées.
- la position occupée.

Le véhicule doit signaler au PPC la fin d'exécution d'une opération mais aussi les éventuelles erreurs survenues lors de son exécution et les problèmes rencontrés au niveau des batteries.

Le gestionnaire de l'atelier définit les batteries que le moyen de transport pourra utiliser, tandis que le composant responsable de la planification et du contrôle de production détermine les périodes d'indisponibilité ainsi que leur cause. Mais, bien que définies par d'autres agents, ces informations sont gérées par le système AGV lui-même.

De même, le PPC détermine la batterie à utiliser, le contenu d'une place du palettiseur et, si le système est en état de fonctionner, il lui demande d'exécuter certaines opérations. Il se réfère pour cela à une opération-type pour laquelle il définit les valeurs de paramètres. L'exécution pourra dès lors être lancée mais le PPC dispose toujours de la possibilité d'interrompre son déroulement normal.

Sur base de toutes les informations que le transport lui fournit ainsi que d'une description générale, le CAPP se charge d'imaginer les nouvelles opérations pouvant être réalisées et les nouveaux types d'objets que les places peuvent accueillir. Il communique enfin la description des nouveaux postes auxquels il est possible d'accéder.

En fait, nous venons d'exprimer les objectifs idéaux du moyen de transport que l'on aimerait le voir réaliser dans le cadre de l'atelier étudié. Toutefois, nous sommes forçés de constater que le "simple" véhicule existant dans l'atelier (appelé ci-dessous AGV) ne peut, à lui seul, réaliser tout ce qui lui est demandé. C'est pourquoi, si nous approfondissons notre analyse et essayons d' "implémenter" ces objectifs (cfr.: notion de refinement présentée au chapitre 2), nous devons ajouter deux autres composants au véhicule proprement-dit: le serveur et le gestionnaire. Ces trois éléments vont donc devoir communiquer entre-eux et interagir afin de réaliser les objectifs établis pour le système AGV.

L'AGV comporte deux batteries, un palettiseur pouvant transporter au plus deux palettes et un PC XT avec unité de disquette.

Il peut exécuter une série d'instructions de plus bas niveau qui permettent le déplacement vers un poste, l'arrêt de tout mouvement, de monter ou descendre l'ascenceur, de tourner le palettiseur, etc... Une fois qu'il reçoit un ordre, il se charge d'en vérifier la syntaxe, de contrôler si son exécution est possible, de gérer les mouvements des différents moteurs afin de le réaliser, et enfin de signaler la fin d'un ordre ou les éventuels problèmes rencontrés.

Pour mener à bien l'exécution d'une opération, il dispose d'un fichier de configuration de l'atelier décrivant la piste sur laquelle il se déplace (longueur, nombre total de codes détectés au sol, nombre de postes réels [1]) ainsi qu'une série d'informations sur les postes figurant sur cette piste (numéro de code à partir de l'origine, distance entre l'origine et le poste, hauteur du palettiseur pour le poste et éventuellement le nom du poste).

Le moyen de transport comprend également une carte d'entrée et une carte de sortie pour faciliter l'utilisation de la partie électronique.

La carte d'entrée indique:

- La présence d'une palette sur un des emplacements du palettiseur.
- L'état de charge des deux batteries.
- La perte de piste.
- Si le signal d'urgence est activé (dans ce cas, le gestionnaire devra activer manuellement la tension des moteurs pour que l'AGV puisse être utilisé à nouveau).
- Le code reçu de la télécommande qui est en cours d'exécution.
- Etc...

[1] Etant donné le mauvais état du sol sur lequel il se déplace, l'AGV détecte des codes qui, parfois, ne correspondent pas à un poste de travail.

D'autre part, la carte de sortie va permettre:

- D'utiliser un écran à cristaux liquides (écran LCD) qui offre la possibilité d'afficher l'ordre en cours d'exécution, de signaler la fin d'exécution d'une opération, d'indiquer les erreurs survenues lors de l'exécution d'une instruction (perte de piste, collision, arrêt d'urgence enfoncé,...).
- De commander les moteurs de direction du véhicule.
- De commander les moteurs du palettiseur.
- Etc ...

L'AGV peut recevoir des ordres du serveur (auquel il est relié par une liaison sérielle) ou du gestionnaire (via une commande infra-rouge).

La principale tâche du serveur (appelé aussi PC) consiste à gérer l'ensemble des informations nécessaires au niveau du système AGV, mais indisponibles au niveau de l'AGV. Il se charge également de réaliser les opérations dites de haut niveau. Pour ce faire, il dispose d'un programme pour chaque opération type. Ces programmes constituent en fait une suite de références à des opérations réalisables par l'AGV. Par conséquent, lorsque le PPC ordonne d'exécuter une opération, celui-ci (le serveur) va se référer à la séquence d'instructions correspondante qu'il fera exécuter à l'AGV au fur et à mesure.

En retour, le PC lui signale la fin d'exécution d'une opération ou les éventuels problèmes rencontrés. Pour mener à bien la réalisation de toutes ces opérations, il a accès aux informations contenues dans la carte d'entrée de l'AGV. Il connaît l'instruction en cours d'exécution et est immédiatement averti une fois qu'elle est terminée ou que des erreurs sont survenues.

Si le PPC a la faculté de commander le serveur, cette possibilité est également offerte au gestionnaire de l'atelier. Celui-ci peut ordonner au PC d'exécuter une instruction de bas niveau (dans ce cas, le serveur joue le simple rôle de relayeur) ou un programme qui correspond à une opération de plus haut niveau (dans ce cas, le serveur se comporte, à peu de choses près, comme si l'ordre provenait du PPC). En retour, le PC lui signale la fin de l'exécution demandée ou il lui communique les erreurs survenues.

L'agent humain détermine également les éventuelles périodes d'indisponibilité du transport regroupées au niveau du PC et conçoit les nouveaux programmes suite à une demande de création d'une nouvelle opération.

Il est aussi responsable de la mise à jour des caractéristiques sur les postes accessibles (au niveau du serveur) en même temps que celle du fichier de configuration situé sur l'AGV. Ces modifications ont d'ailleurs toujours lieu après une demande d'ajout d'un poste en provenance du CAPP.

De plus, le gestionnaire dispose d'une télécommande infra-rouge pour commander le transport.

Exceptionnellement, il éprouve parfois la nécessité d'interrompre le fonctionnement normal des moteurs, en appuyant sur le signal d'urgence. L'AGV pourra alors reprendre une activité normale après que son responsable ait activé la tension des moteurs.

5.3. Modélisation

Rappelons au lecteur que tout modèle, exprimé dans le langage ALBERT, comporte trois parties:
1. La hiérarchisation des agents.
2. La spécification de chacun des agents.
3. L'expression des contraintes pour chaque agent.

5.3.1 Hiérarchisation

A partir de la description informelle présentée ci-dessus, nous pouvons définir dans notre hiérarchie (représentée par la figure 5.2.) un premier agent, le système AGV, dont nous présentons la spécification au point 5.3.2 de ce chapitre. Si nous nous intéressons d'avantage au système AGV et passons à son implémentation (refinement), il faut définir en plus de l'AGV l'agent "Serveur" et l'agent "Gestionnaire". Ces trois derniers agents ont évidemment un rôle précis (voir Annexe B) et permettent par leurs interactions de réaliser les objectifs assignés au système AGV.

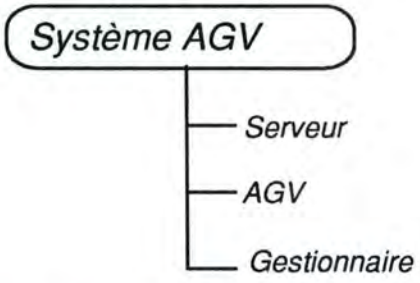


Figure **-5.2-** Le système AGV: Hiérarchisation des agents

5.3.2. Spécification du système AGV

5.3.2.1. Structure des états

La figure 5.3. reprend la structure de ce composant et la figure 5.4. celle des types utilisés.

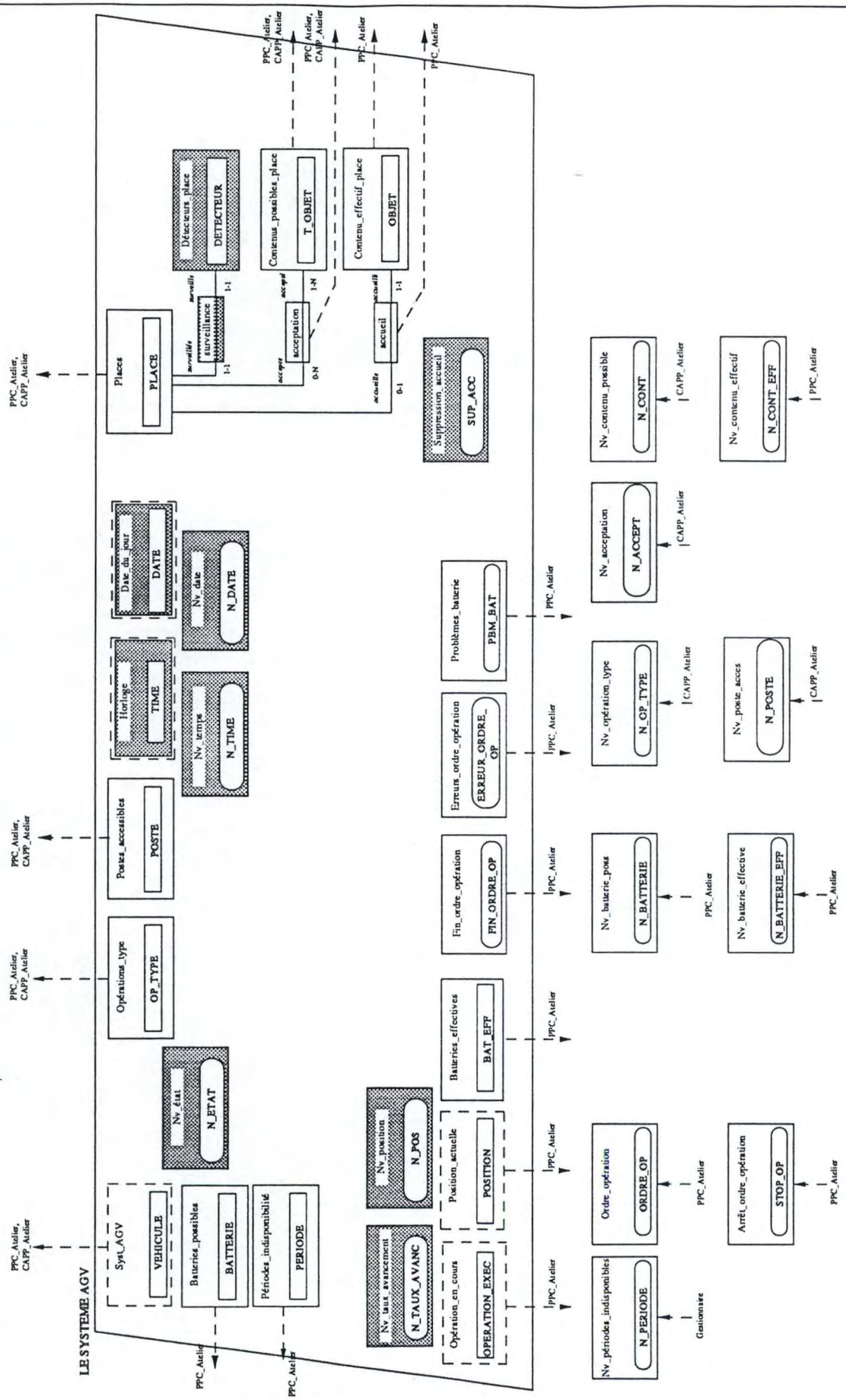


Figure -5.3- Structure des états du système AGV

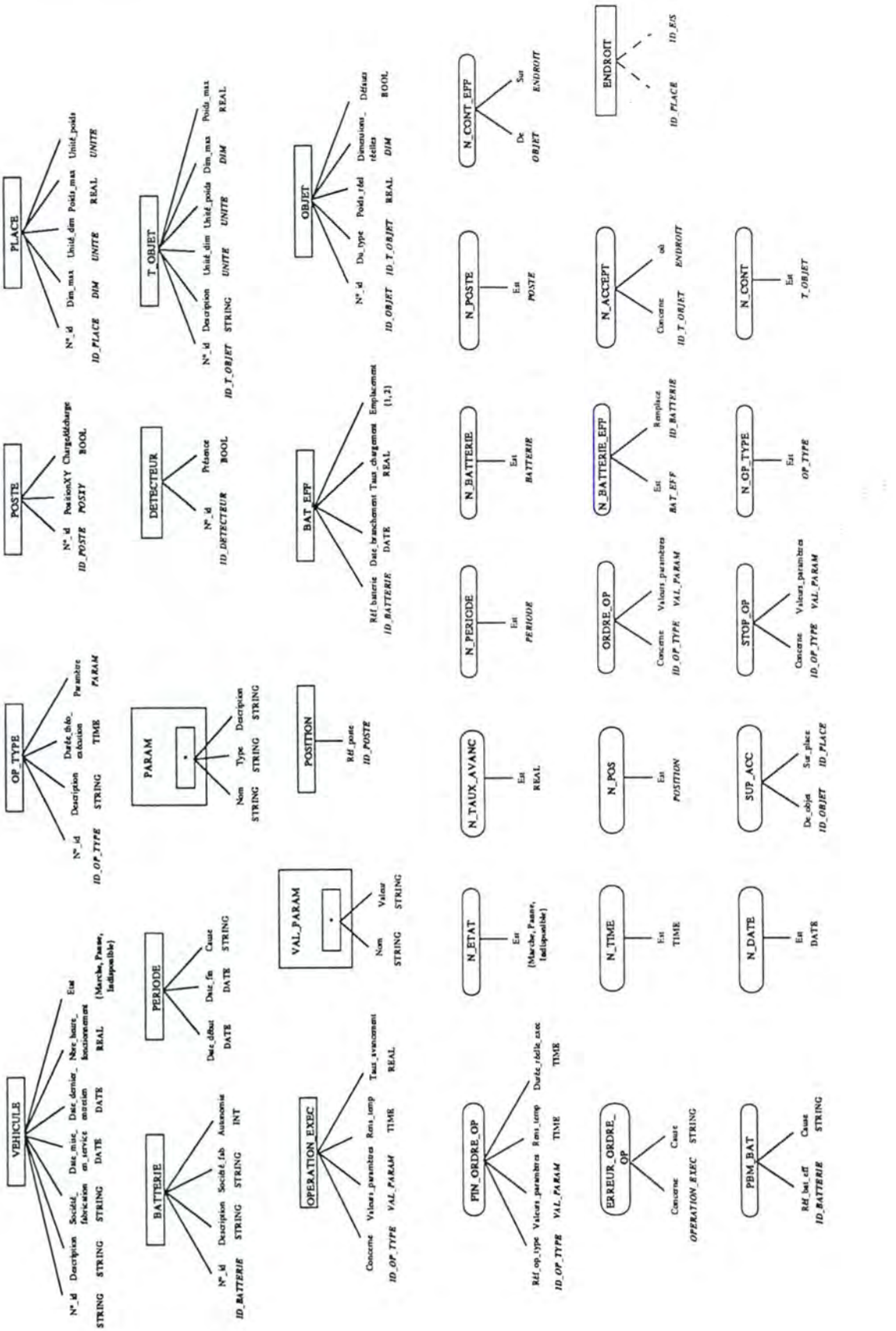


Figure **-5.4-** Le système AGV: structure des types

5.3.2.2. Commentaires

Informations gérées par l'agent

- Syst_AGV

  ■ Description Informations générales sur le système AGV telles que l'identifiant, la description informelle de ses caractéristiques, les dates de mise en service et du dernier entretien, l'état du système (indiquant s'il est en état de marche, en panne ou indisponible),...
  ■ Type **VEHICULE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER, CAPP_ATELIER*
  ■ Structure ☞ Figure 5.4

- Batteries_possibles

  ■ Description Liste des batteries acceptées. Celles-ci sont caractérisées par un identifiant, une société de fabrication, une autonomie, etc...
  ■ Type **BATTERIE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER*
  ■ Structure ☞ Figure 5.4

- Périodes_indisponibilité

  ■ Description Reprend un ensemble d' intervalles de dates correspondant à des périodes d'indisponibilité du moyen de transport ainsi que la cause de cette indisponibilité.
  ■ Type **PERIODE**
  ■ Lien ∅
  ■ Visible par *PPC_ATELIER*
  ■ Structure ☞ Figure 5.4

• Opérations_type

- ■ Description: Contient les informations sur les opérations de haut niveau que le système AGV peut effectuer. Ces opérations sont caractérisées par un nom identifiant (Move, Wait,...), une description informelle, une durée théorique d'exécution et par un ensemble de paramètres à définir pour son exécution.
- ■ Type: **OP_TYPE**
- ■ Lien: ∅
- ■ Visible par: *PPC_ATELIER, CAPP_ATELIER*
- ■ Structure: ☞ Figure 5.4

• Postes_accessibles

- ■ Description: Reprend une description des postes auxquels le système peut accéder. Chacun des postes est caractérisé par un identifiant, une position dans l'atelier et un booléen qui indique si on peut y réaliser un chargement ou un déchargement ou s'il s'agit d'un simple point d'arrêt.
- ■ Type: **POSTE**
- ■ Lien: ∅
- ■ Visible par: *PPC_ATELIER, CAPP_ATELIER*
- ■ Structure: ☞ Figure 5.4

• Places

- ■ Description: Différentes caractéristiques (telles que les dimensions et poids maximaux des palettes qu'elles peuvent accueillir) des deux places du palettiseur.
- ■ Type: **PLACE**
- ■ Lien:
  - **Surveillance**: Désigne le détecteur associé à une place.
  - **Acceptation**: Indique les types d'objets acceptés par la place.
  - **Accueil**: Détermine le contenu effectif d'une place à un moment donné
- ■ Visible par: *PPC_ATELIER, CAPP_ATELIER*
- ■ Structure: ☞ Figure 5.4

• Détecteurs_place

- Description: Reprend l'identifiant du détecteur ainsi qu' un booléen indiquant s'il détecte ou non une présence.
- Type: **DETECTEUR**
- Lien: **Surveillance**: Voir ci-dessus.
- Visible par: ∅
- Structure: ☞ Figure 5.4

• Contenus_possibles_place

- Description: Comprend entre autres un identifiant, une description, une dimension et un poids idéal pour chaque type d'objet connu du système AGV.
- Type: **T_OBJET**
- Lien: **Acceptation**: Voir ci-dessus
- Visible par: *PPC_ATELIER, CAPP_ATELIER*
- Structure: ☞ Figure 5.4

• Contenu_effectif_place

- Description: Caractéristiques sur les objets qui figurent réellement sur le moyen de transport. Ces caractéristiques donnent, en plus de l'identifiant de l'objet, le type d'objet auquel il appartient, les dimensions et le poids réels et indiquent si un défaut a été constaté.
- Type: **OBJET**
- Lien: **Accueil**: Voir ci-dessus.
- Visible par: *PPC_ATELIER*
- Structure: ☞ Figure 5.4

• Opération_en_cours

- Description: Donne les informations suivantes sur l'opération en cours d'exécution: la référence à l'opération type (indéterminée si le système ne fait rien), la valeur des paramètres, le moment précis où l'ordre a été pris en considération et le taux d'avancement de l'exécution.
- Type: **OPERATION_EXEC**
- Lien: ∅
- Visible par: *PPC_ATELIER*
- Structure: ☞ Figure 5.4

• Position_actuelle

- ■ Description Désigne la position courante du système. Celle-ci est indéterminée si le moyen de transport effectue un déplacement.
- ■ Type **POSITION**
- ■ Lien ∅
- ■ Visible par *PPC_ATELIER*
- ■ Structure ☞ Figure 5.4

• Batteries_effectives

- ■ Description Reprend les références des batteries effectivement branchées sur l'AGV ainsi que leur date de branchement, leur taux de charge réel et, enfin, l'emplacement sur lesquelles elles sont situées.
- ■ Type **BAT_EFF**
- ■ Lien ∅
- ■ Visible par *PPC_ATELIER*
- ■ Structure ☞ Figure 5.4

• Horloge

- ■ Description Indique l'heure courante.
- ■ Type **TIME**
- ■ Lien ∅
- ■ Visible par ∅

• Date_du_jour

- ■ Description Donne la date du jour.
- ■ Type **TIME**
- ■ Lien ∅
- ■ Visible par ∅

• Fin_ordre_opération

- ■ Description Indique qu'une certaine opération est terminée et donne le moment où son exécution a commencé et la durée réelle de son exécution.
- ■ Type **FIN_ORDRE_OP**
- ■ Destination *PPC_ATELIER*
- ■ Structure ☞ Figure 5.4

• Erreurs_ordre_opération

- Description: Informe le PPC de la cause qui a interrompu l'exécution normale d'une opération, du moment auquel cette exécution a débuté et du taux d'avancement de celle-ci.
- Type: **ERREUR_ORDRE_OP**
- Destination: *PPC_ATELIER*
- Structure: ☞ Figure 5.4

• Problèmes_batteries

- Description: Signale au PPC les éventuels problèmes détectés en ce qui concerne les batteries effectives.
- Type: **PBM_BAT**
- Destination: *PPC_ATELIER*
- Structure: ☞ Figure 5.4

• Nv_état

- Description: Met à jour l'état du système.
- Type: **N_ETAT**
- Destination: ∅
- Structure: ☞ Figure 5.4

• Nv_temps

- Description: Permet de mettre à jour l'horloge.
- Type: **N_TIME**
- Destination: ∅
- Structure: ☞ Figure 5.4

• Nv_date

- Description: Action qui vient modifier la date courante.
- Type: **N_DATE**
- Destination: ∅
- Structure: ☞ Figure 5.4

• Suppression_accueil

- Description: Action qui supprime l'association entre l'objet et la place désignés et qui enlève ce même objet de l'ensemble des objets effectivement situés sur les places du moyen de transport.
- Type: **SUP_ACC**
- Destination: ∅
- Structure: ☞ Figure 5.4

• Nv_taux_avancement

- Description: Donne le taux d'avancement de l'exécution en cours.
- Type: **N_TAUX_AVANC**
- Destination: ∅
- Structure: ☞ Figure 5.4

• Nv_position

- Description: Indique la nouvelle position qu'occupe le moyen de transport.
- Type: **N_POS**
- Destination: ∅
- Structure: ☞ Figure 5.4

B. Informations reçues de l'extérieur

• Nv_périodes_indisponibles

- Description: Détermine certaines périodes au cours desquelles le système sera indisponible (pour entretien, par exemple).
- Type: **N_PERIODE**
- Origine: *Gestionnaire*
- Structure: ☞ Figure 5.4

• Ordre_opération

- Description: Demande au système d'exécuter une certaine opération type pour les valeurs des paramètres définies.
- Type: **ORDRE_OP**
- Origine: *PPC_ATELIER*
- Structure: ☞ Figure 5.4

• Arrêt_ordre_opération

- Description : Interrompt l'opération en cours d'exécution.
- Type : **STOP_OP**
- Origine : *PPC_ATELIER*
- Structure : ☞ Figure 5.4

• Nv_batterie_poss

- Description : Donne la description de nouvelles batteries que l'AGV pourra par la suite utiliser.
- Type : **N_BATTERIE**
- Origine : *PPC_ATELIER*
- Structure : ☞ Figure 5.4

• Nv_batterie_effective

- Description : Désigne la batterie à remplacer et celle à brancher.
- Type : **N_BATTERIE_EFF**
- Origine : *PPC_ATELIER*
- Structure : ☞ Figure 5.4

• Nv_opération_type

- Description : Exige du système qu'il puisse réaliser l'opération de haut niveau dont le CAPP transmet les caractéristiques.
- Type : **N_OP_TYPE**
- Origine : *CAPP_ATELIER*
- Structure : ☞ Figure 5.4

• Nv_poste_acces

- Description : Donne la description d'un nouveau poste accessible.
- Type : **N_POSTE**
- Origine : *CAPP_ATELIER*
- Structure : ☞ Figure 5.4

• Nv_acceptation

- Description : Informe le système qu'une certaine place peut accueillir un certain type d'objet.
- Type : **N_ACCEPT**
- Origine : *CAPP_ATELIER*
- Structure : ☞ Figure 5.4

- Nv_contenu_possible

  - Description : Permet d'ajouter un élément à l'ensemble des types d'objets connus du système
  - Type : **N_CONT**
  - Origine : *CAPP_ATELIER*
  - Structure : ☞ Figure 5.4

- Nv_contenu_effectif

  - Description : Communique les informations sur le nouveau contenu d'une place
  - Type : **N_CONT_EFF**
  - Origine : *PPC_ATELIER*
  - Structure : ☞ Figure 5.4

5.3.2.3. Les contraintes

Contraintes sur l'état

- *Le système AGV ne peut pas être indéfiniment hors d'usage.*

  Etat (Syst_AGV) = "*Panne*" **or** Etat (Syst_AGV) = "*Indisponible*"
  $\Rightarrow \Diamond$ Etat (Syst_AGV) = "*Marche*"

- *Le système doit à un moment ou à un autre terminer l'exécution d'une opération.*

  Concerne (Opération_en_cours)$\neq$ *Indéterminé*
  $\Rightarrow \Diamond$ Concerne (Opération_en_cours) = *Indéterminé*

Effets des actions

- *L'action Nv_périodes_indisponibles a pour effet d'ajouter une occurence dans la classe Périodes_indisponibilité.*

  PPC.Nv_périodes_indisponibles (period) :
  Add (Périodes_indisponibilité,period)

• *Dès que le PPC demande l'exécution d'une opération, les éléments décrivant l'opération en cours sont mis à jour:*

- *L'opération exécutée est celle désignée par l'action. Il en est de même pour la valeur des paramètres.*
- *Le champs Rens_temp détermine le moment où l'ordre émis par le PPC est pris en considération.*
- *Au départ, le taux d'avancement pour l'exécution en cours est nul.*

PPC.Ordre_opération (ordre):
Concerne (Opération_en_cours) = Concerne (ordre) **and**
Valeurs_paramètres (Opération_en_cours) = Valeurs_paramètres (ordre) **and**
Rens_temp (Opération_en_cours) = Horloge **and**
Taux_avancement (Opération_en_cours) = 0

• *Suite à une action Nv_batterie_poss du PPC, l'agent met à jour la classe Batteries_possibles.*

PPC.Nv_batterie_poss (bat) :
Add (Batteries_possibles,bat)

• *L'action Nv_batterie_effective a pour effet de supprimer l'ancienne batterie de l'ensemble des batteries effectives et d'y ajouter les caractéristiques de la nouvelle batterie.*

PPC.Nv_batterie_effective (bat):
Remove (Batteries_effectives, Remplace(bat)) **and**
Add (Batteries_effectives, Est(bat))

• *L'agent met à jour les opérations types après l'action Nv_opération_type.*

CAPP.Nv_opération_type (op) :
Add (Opérations_type,op)

• *L'action Nv_poste_acces ajoute un nouveau poste accessible.*

CAPP.Nv_poste_acces (p) :
Add (Postes_accessibles,p)

• *L'action Nv_acceptation établit une association entre la place et le type d'objet désignés.*

CAPP.Nv_acceptation (ncont):
Add (Acceptation, < Concerne(ncont), Où(ncont) >)

• *La classe Contenus_possibles_place se voit ajouter un élément après une occurence de Nv_contenu_possible.*

CAPP.Nv_contenu_possible (cont):
Add (Contenus_possibles_place,cont)

• *Cette action met à jour les informations au sujet de l'objet figurant sur la place indiquée.*

PPC.Nv_contenu_effectif (cont_eff):
Add (Contenu_effectif_place, De(cont_eff)) **and**
Add (Accueil, <N°_id(De(cont_eff)), Sur(cont_eff)>)

• *Le système donne la valeur indéterminée au champs Concerne de l'opération en cours quand l'action Fin_ordre_opération survient.*

Fin_ordre_opération (fin).*PPC*:
Concerne (Opération_en_cours) = *Indéterminé*

• *Suite à une erreur dans l'exécution d'une opération, la référence à l'opération type (au niveau de l'opération en cours) est indéterminée.*

Erreurs_ordre_opération (erreur).*PPC*:
Concerne (Opération_en_cours) = *Indéterminé*

• *Cette action permet de changer l'état du système.*

Nv_état (nouv_état):
Etat(Syst_AGV)=nouv_état

• *Cette action permet de mettre à jour l'horloge.*

Nv_temps (nouv_temps):
Horloge=nouv_temps

• *L'action Nv_date met à jour la date du jour.*

Nv_date (nouv_date):
Date_du_jour=nouv_date

• *L'action Suppression_accueil a pour effet de supprimer l'association existante entre la place désignée et son contenu et d'enlever ce dernier de l'ensemble des objets effectivement situés sur les places.*

Suppression_accueil (s_cont):
Remove (Accueil, < De_objet(s_cont), Sur_place(s_cont) >) **and**
Remove (Contenu_effectif_place, De_objet(s_cont))

- *Cette action donne la nouvelle position qu'occupe le système AGV.*

Nv_position (nouv_pos):
Position_actuelle=nouv_pos

- *Nv_taux_avancement indique le taux d'avancement pour l'exécution en cours.*

Nv_taux_avancement (nouv_avanc):
Taux_avancement (Opération_en_cours)=nouv_avanc

Engagements

Signalons au lecteur que les contraintes temporelles exprimées ci-dessous le sont à titre indicatif. Il faudra donc éventuellement les revoir dans la phase ultérieure d'implémentation.

- *L'agent s'engage à répondre (dans les dix minutes) à une demande d'éxécution d'opération, par une action indiquant la fin d'exécution de cette opération ou par une action définissant la cause qui a interrompu la bonne exécution de l'opération.*

PPC.Ordre_opération (ordre)
→ ◊ (≤*10min*)
(Fin_ordre_opération (fin).*PPC*
With Concerne (fin)=Concerne (ordre) **and**
Valeurs_paramètres (fin)=Valeurs_paramètres (ordre) **and**
Rens_temp (fin)=Rens_temp (Opération_en_cours) **and**
Durée_réelle_exec (fin)
=Dif_temps (Horloge, Rens_temp(Opération_en_cours) ⊕
(Erreurs_ordre_opération (erreur).*PPC*
With Concerne (erreur)=Concerne (ordre) **and**
Valeurs_paramètres (erreur)
=Valeurs_paramètres (Opération_en_cours) **and**
Rens_temp (erreur)=Rens_temp (Opération_en_cours) **and**
Taux_avancement (erreur)
=Taux_avancement (Opération_en_cours))

{Dif_temps(t1,t2): fonction qui donne pour résultat la différence entre le temps t1 et le temps t2}

• *Le système AGV s'engage à générer une action Erreurs_ordre_opération après une demande d'interrruption de l'exécution en cours.*

PPC.Arrêt_ordre_opération (ar_ordre)
→ ◊ (≤*1min*)
Erreurs_ordre_opération (erreur).*PPC*
With Concerne (erreur)=Concerne (ar_ordre) **and**
Valeurs_paramètres (erreur)
=Valeurs_paramètres (Opération_en_cours) **and**
Rens_temp (erreur)=Rens_temp (Opération_en_cours) **and**
Taux_avancement (erreur)
=Taux_avancement (Opération_en_cours) **and**
Cause (erreur)="*Arrêt*"

Responsabilités

• *Le système ne tient pas compte d'un ordre s'il est en panne ou indisponible, s'il exécute déjà une autre opération ou si cet ordre se réfère à une opération inconnue.*

F[*PPC*.Ordre_opération (ordre)/
Etat (*Syst_AGV*)="*Panne*" **or**
Etat (*Syst_AGV*)="*Indisponible*" **or**
Concerne (Opération_en_cours)≠*Indéterminé* **or**
NOT In_Opérations_type (Concerne(ordre))]

• *L'agent ne tient pas compte de l'action d'arrêt, s'il ne fait rien, et refuse d'interrompre l'exécution d'une opération, si l'opération et la valeur des paramètres spécifiées ne correspondent pas à l'opération et à la valeur des paramètres de l'opération en cours.*

F[*PPC*.Arrêt_ordre_opération (ar_ordre)/
Concerne (Opération_en_cours)= *Indéterminé* **or**
Concerne (Opération_en_cours) ≠ Concerne (ar_ordre) **or**
Valeurs_paramètres (Opération_en_cours)
≠ Valeurs_paramètres (ar_ordre)]

• *Le système AGV ne prend pas en considération l'action Nv_acceptation quand la place ou l'objet désigné est inconnu.*

F[*CAPP*.Nv_acceptation (ncont)/
NOT In_Places (Concerne (ncont)) **or**
NOT In_Contenus_possibles_place (Où(ncont))]

- *L'action Nv_batterie_effective est omise si:*
  - *La batterie à brancher ne figure pas dans l'ensemble des batteries acceptées par le système AGV.*
  - *La nouvelle batterie est déjà utilisée.*
  - *La batterie remplacée ne correspond pas à celle qui est située à l'endroit désigné.*

F[*PPC*.Nv_batterie_effective (bat)/
NOT In_Batteries_possibles (Réf_batterie(Est(bat))) **or**
In_Batteries_effectives (Réf_batterie(Est(bat))) **or**
{Remplace(bat)≠Réf_batterie(bat_rempl) **and**
Emplacement(Est(bat))=Emplacement(bat_rempl) **and**
In_Batteries_effectives (Réf_batterie(bat_rempl))}]

- *L'action Nv_contenu_effectif n'a aucun effet si le détecteur associé à la place désignée n'indique aucune présence, si l'objet indiqué n'appartient pas à un type d'objet accepté par cette même place ou, enfin, si un objet occupe déjà cette place.*

F[*PPC*.Nv_contenu_effectif (cont_eff)/
(∃ d: In_Surveillance (d, Sur (cont_eff)) **and**
Présence (Détecteurs_place (d))="*N*") **or**
NOT In_Acceptation (Sur(cont_eff), Du_type(De(cont_eff))) **or**
∃ c: In_Accueil (c, Sur(cont_eff))]

- *L'agent ne prend pas en compte l'action Nv_position si elle ne désigne pas un poste accessible.*

F[Nv_position (nouv_pos)/
NOT In_Postes_accessibles (nouv_pos)]

- *Le système ne considère pas cette action si aucune opération n'est en cours ou si le nouveau taux est inférieur au taux d'avancement de l'exécution en cours.*

F[Nv_taux_avancement (nouv_avanc)/
Concerne (Opération_en_cours)=*Indéterminé* **or**
(Concerne (Opération_en_cours)≠*Indéterminé* **and**
Taux_avancement (Opération_en_cours)>nouv_avanc)]

- *L'action Nv_date n'a aucun effet si la date qu'elle indique est inférieure à la date courante.*

F[Nv_date (nouv_date)/
Supérieur (Date_du_jour,nouv_date)]

{Supérieur(d1,d2): cette fonction vérifie que la date d1 est supérieure à la date d2}

• *Le système doit être mis dans l'état "Indisponible" si l'on se trouve dans une période d'indisponibilité.*

O[Nv_état("*Indisponible*")/
Etat (*Syst_AGV*)="*Marche*" **and**
∃ d1, d2: In_Périodes_indisponibilité(d1,d2) **and**
Supérieur (Date_du_jour,d1) **and**
Supérieur (d2,Date_du_jour)]

• *L'agent signale un problème au sujet d'une batterie, quand le taux de chargement de cette dernière est inférieur à 20 pourcents.*

O[Problèmes_batteries(pbm).*PPC*/
∃ b: In_Batteries_effectives (b) **and**
b=Réf_bat_eff (pbm) **and**
Taux_chargement (Batteries_effectives(b))<*0.2*]

• *Si le détecteur associé à une place ne détecte aucune présence, mais qu'une association Accueil indique qu'un objet figure sur cette place, il faut supprimer cette association.*

O[Suppression_accueil (s_cont)/
∃ d: In_Surveillance (d, Sur_place(s_cont)) **and**
Présence (Détecteurs_place(d))="*N*" **and**
In_Accueil (De_objet(s_cont), Sur_place(s_cont))]

Nous venons donc de présenter les spécifications complètes du système AGV. Toutefois, si nous affinons ces objectifs et essayons de déterminer, au niveau de l'analyse des besoins, comment ils vont être réalisés (Refinement), on se rend très vite compte que l'existant , à savoir l'AGV , ne peut à lui seul atteindre l'ensemble de ces objectifs. Par conséquent, il faut introduire deux agents supplémentaires, le serveur et le gestionnaire, qui viennent en quelque sorte compléter l'AGV, et ce, afin de pouvoir "implémenter" les spécifications développées ci-dessus. Chacun de ces trois composants a bien entendu un rôle précis, qui est en partie défini par la structure d'états (Voir figures 5.5, 5.6 et 5.7).

Si nous parcourons superficiellement leurs structures d'états, on peut constater que le serveur est responsable de décomposer les ordres de haut niveau provenant du PPC en instructions que l'AGV est mesure d'exécuter. De plus, il gère une série d'informations nécessaires au niveau de l'atelier (PPC et CAPP) et que l'AGV est dans l'incapacité de fournir telles quelles. Ce dernier, par contre, se charge de commander les différents moteurs afin d'exécuter les ordres que lui envoie le serveur et il l'informe des fins d'exécution, des erreurs survenues,... Finalement, le gestionnaire effectue les mises à jour du fichier de configuration nécessaire au bon fonctionnement du véhicule et conçoit les programmes réalisant les opérations de haut niveau, suite aux actions du CAPP indiquant qu'un nouveau poste est accessible ou qui exigent du système AGV qu'il puisse exécuter une nouvelle opération.

A partir de ce bref aperçu des fonctions de ces composants, on devine rapidement que, par leurs échanges d'informations et interactions, ces agents constituent le système AGV et réalisent ainsi les objectifs qui lui sont assignés dans le cadre de l'atelier étudié.

Le lecteur intéressé peut consulter l'annexe B, s'il désire prendre connaissance des spécifications complètes de ces trois derniers agents.

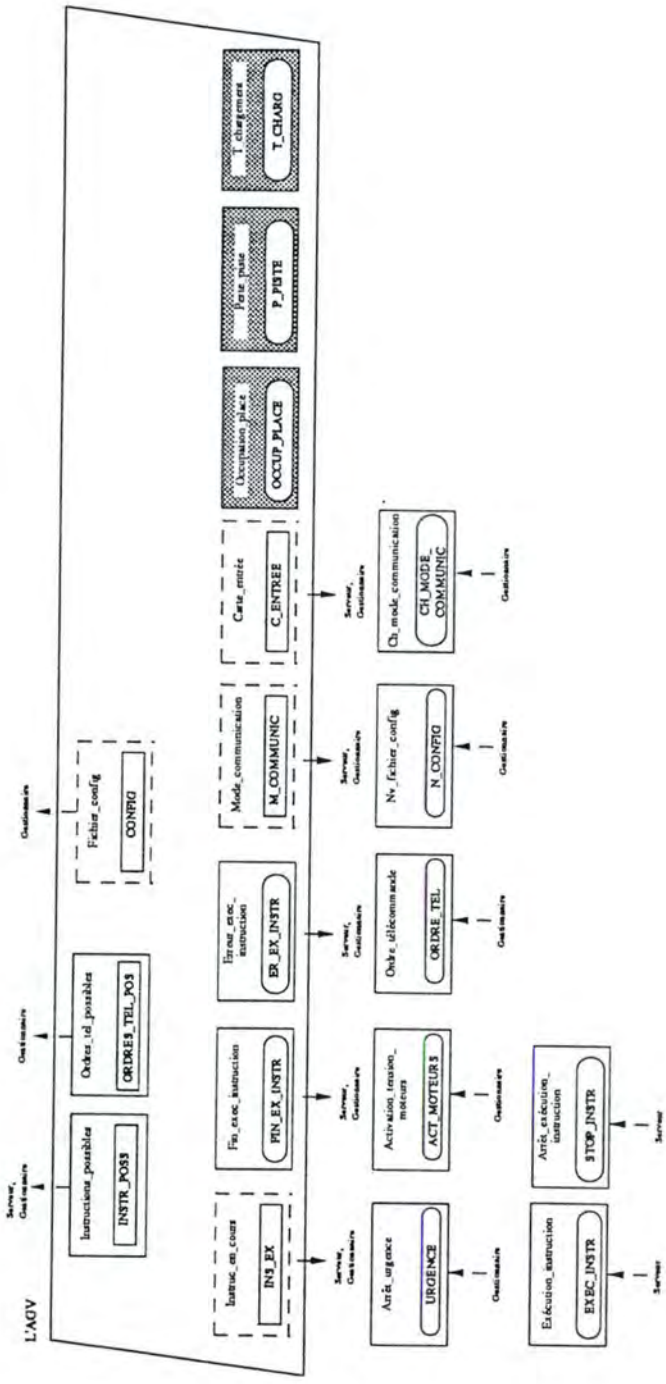


Figure **-5.5-** Structure des états de l'AGV

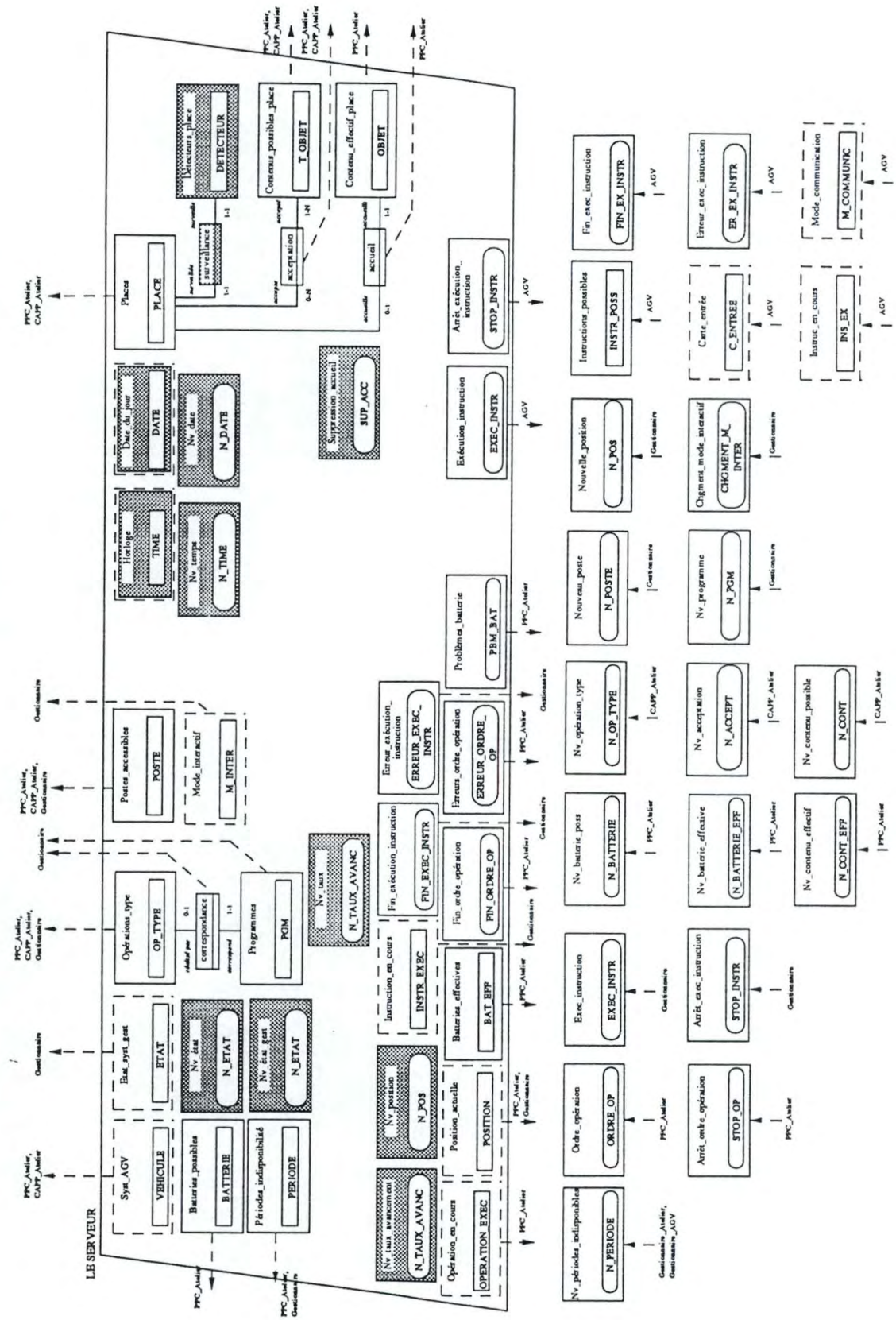


Figure **-5.6-** Structure des états du serveur

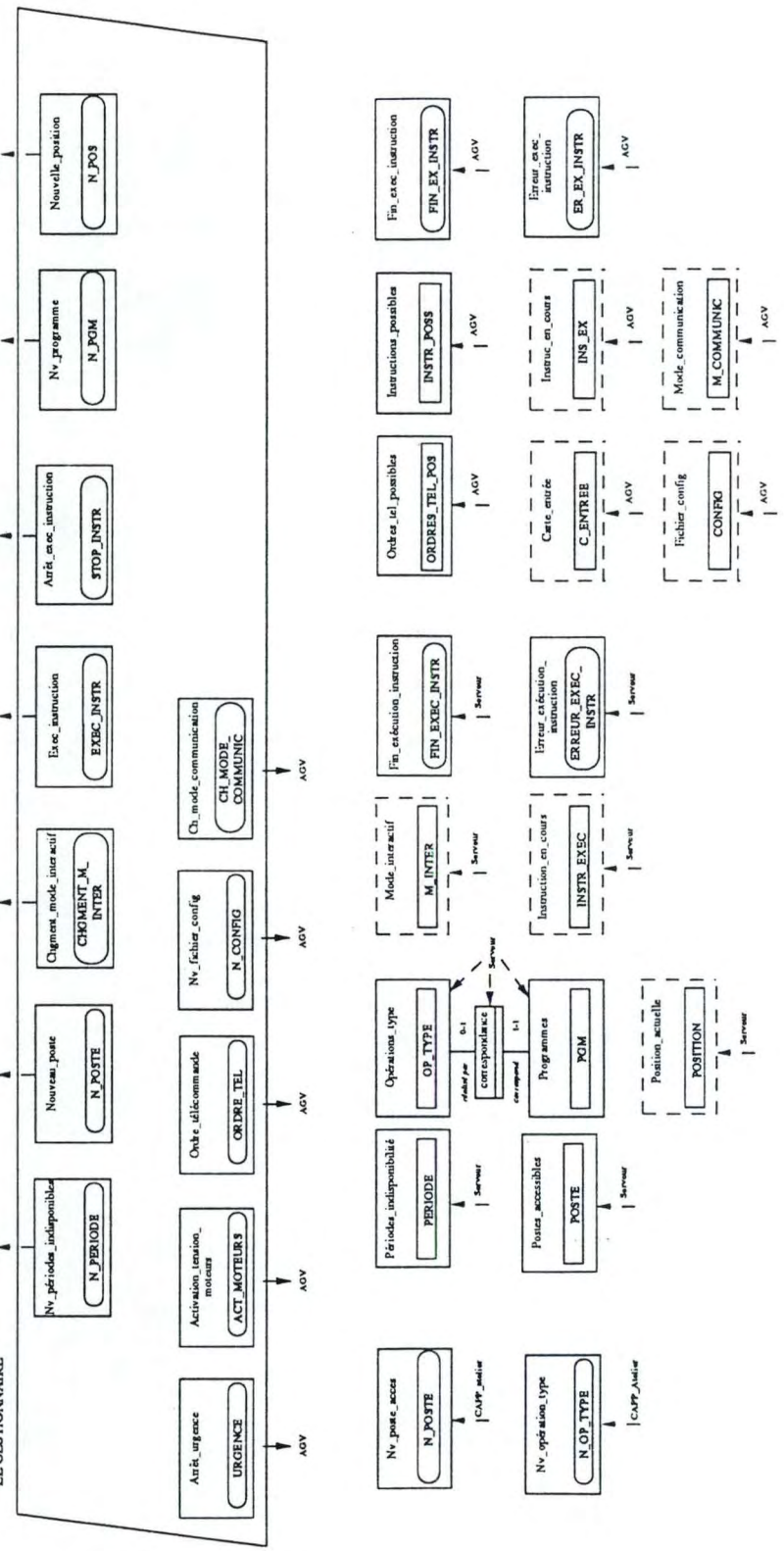


Figure **-5.7-** Structure des états du gestionnaire

Conclusion

Dans ce travail, nous avons commencé par présenter les enjeux importants que sont ceux de l'Ingénierie des Besoins ainsi que les nouvelles tendances, à savoir la définition de structures génériques pour des domaines d'application particuliers.

Pour l'élaboration de telles structures génériques et ensuite pour les structures particulières, nous avons présenté le langage ALBERT. Ce langage est basé sur la notion de systèmes composites dont les caractéristiques semblent bien adaptées à la formulation des besoins pour les systèmes productiques.

Nous avons ensuite présenté une structure générique pour le domaine de la productique. Cette structure générique a été instanciée pour modéliser une entreprise productique de petite taille. Lors de cette instanciation, nous avons pu mettre en évidence les avantages et les faiblesses de la structure générique. Cela nous a amené à proposer quelques modifications de la structure générique. A l'issue de cette instanciation, nous disposons d'une architecture particulière, adaptée à notre entreprise.

Cette architecture nous donne une image du système d'information "idéal", c'est-à-dire celui qui répond aux besoins formulés par les utilisateurs. Cependant, le plus souvent le système idéal doit être construit à partir d'un existant dont il faut tenir compte.

Les particularités et surtout les limites de cet existant, nous ont poussés à affiner la structure que nous proposions. Cette démarche de raffinement a été mise en oeuvre pour un seul composant, à savoir le composant de transport AGV.

Nous avons donc défini de nouveaux composants (de niveaux inférieurs) qui collaborent à la réalisation de l'objectif du système AGV "idéal". Pour chacun de ces composants, nous avons donné une spécification complète, c'est-à-dire leur structure d'état ainsi que les contraintes régissant leur comportement.

La démarche utilisée est du type *Top-Down*. Ce type de démarche semble bien adaptée pour l'étude des besoins, mais la plupart du temps le nouveau système (à créer) repose sur un existant (si petit soit-il) dont il faut tenir compte. A partir de ce moment, il convient d'adopter une démarche de type *Bottom-Up*.

Plusieurs extensions à ce travail peuvent être envisagées. Nous n'en proposons que quelques unes.

- Tout d'abord, ce travail consistait en l'expérimentation d'une structure générique afin d'en découvrir les avantages et les inconvénients. Par exemple, dans cette structure générique, nous n'avons pas trouvé de composant s'occupant du contrôle de qualité, alors que cela devient une préoccupation sans cesse croissante dans les entreprises.

- Ensuite, la spécification (globale) que nous avons proposée pour l'entreprise n'est que partielle. En effet, le choix des informations pertinentes a été guidé par l'objectif qu'était la réalisation d'un système de monitoring de la production ne constituant qu'une petite partie de l'objectif ultime de la modélisation, à savoir un système totalement intégré. A plusieurs reprises, nous nous référons à un agent dénommé Gestionnaire. Cependant, aucune définition de structure d'état n'en a été donnée.

- Enfin, nous ne proposons dans ce travail, la spécification complète que pour un seul agent. Il serait donc utile de poursuivre la spécification des autres composants tel que le CAD.

Bibliographie

[Ami93] Esprit Consortium AMICE, editor CIMOSA : Open System Architecture for CIM, Volume 1 of Research Reports Esprit, Project 688/5288 AMICE, Springer-Verlag, 2nd revised and extended edition, 1993.

[Bar93] D. Barstow, "*Should We Specify Systems or Domains ?*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Bod89] F. Bodart, Y. Pigneur, "*Conception Assistée des Systèmes d'Information : Méthode, Modèles, Outils*", 2ème Edition, Masson, 1989.

[Cas93] S. Castano, V. De Antonellis, "*Reuse of Conceptual Requirements Specifications*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Coa92] P. Coad, E. Yourdon, "*Object Oriented Analysis*",2ème édition, Yourdon Press Computing Series, 1992

[Coh86] B. Cohen, W.T. Harwood, M.I. Jackson, "*The Specification of Complex Systems*", Addison-Wesley Publishing Company, 1986.

[Dou90] G. Dougmeingts, "*Méthodes pour concevoir et spécifier les systèmes de production*", In "*Productique et Intégration (CIM : Integration Aspects)*",pages 89-103, Colloque International CIM 90 (Bordeaux), Teknea, Marseille Toulouse Barcelone, 1990

[Dub82] E.Dubois, JP. Finance, A. Van Lamsweerde, "*Towards a Deductive Approach to Information System Specification and Design*", in "*Requirements Engineering Environnements*", edited by Y. Ohno, North-Holland Publishing Company, 1982.

[Dub90] E. Dubois, J. Hagelstein, A. Rifaut, "*ERAE : A Fromal Language for expressing and structuring Real-Time Requirements*", Philips Research Laboratory, Louvain-La-Neuve, Juin 1990

[Dub91] E.Dubois, J.Hagelstein, A. Rifaut, "*A Formal Language for the Requirements Engineering of Computer Systems*", in "*From Natural Language Processing to Logic for Expert Systems*", Edited by A. Thayse, Wiley, 1991.

[Dub92] E.Dubois, Ph. Du Bois, A. Rifaut, "*Elaborating, Structuring and Expressing Formal Requirements of Composite Systems*", Presented at the 4th International Conference on Advanced Information Systems Engineering - CAISE 92, Manchester (Angleterre), Mai 1992.

[Dub93a] E. Dubois, Ph. Du Bois, M. Petit, "*Elicitating and Formalising Requirements for C.I.M. Information Systems*", Presented at the Fifht Conference on Advanced Information Systems Engineering - CAISE 93, Paris (France), Juin 1993.

[Dub93b] E.Dubois, Ph. Du Bois, M. Petit, "*O-O Requirements Analysis : an Agent Perspective*", Presented at the Seventh European Conference on Object-Oriented Programming - ECOOP'93, Kaiserlautern (Allemagne), Juillet 1993.

[Dub93c] E. Dubois, Ph. Du Bois, "*Reasoning on the Elaboration of Formal Requirements for Composite Systems*", Presented at the IEEE International Symposium on Requirements Engineering - RE 93, San Diego CA, USA, Janvier 1993.

[Dub93d] E. Dubois, J-M Zeippen, "*Object-Oriented Formal Development of Cooperative Information Systems*", Position Statements for the ECOOP'93 Workshop on the Application of Object Formal Method, 1993

[Eas93] S. Easterbrook, "*Domain Modelling with Hierarchies of Alternatives Viewpoints*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Gar93] D. Garlan, "*Domain Specifications Require First Class Connectors*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Gru93] R. Gruia-Catalin, "*A Taxonomy of Current Issues In Requirements Engineering*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Jak93] M. Jackson, P. Zave, "*Domaine Descriptions*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Lut93] R. R. Lutz, "*Analysing Software Requirements Errors in Safety-Critical Embedded Systemes*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA, Janvier 1993.

[Mil82] T.J. Miller, B. J. Taylor, "*A Requirements Methodology for Complex Real-Time Systems*", in "*Requirements Engineering Environments*", edited by Y. Ohno, North-Holland Publishing Company, 1982.

[Mit82] R. T. Mittermeir, P. Hsia, R. T. Yeh, "*Alternatives to Overcome the Communication Problem of Formal Requirements Analysis*", in "*Requirements Engineering Environments*", edited by Y. Ohno, North-Holland Publishing Company, 1982.

[Pet92] M. Petit, "*Construction et Formalisation de Spécifications Conceptuelles pour les Systèmes Productiques*", Mémoire de l'Institut d'Informatique des Facultés Universitaires Notre Dame de la Paix, Namur, 1992.

[Ros77] D. T. Ross, K. G. Schoman, "*Structured Analysis for Requirements Definition*", IEEE Trans. Soft. Eng., SE-3 (1), pages 1-65, 1977.

[Rze93] W. E. Rzepka, J. L. Sidoran, D. A. White, "*Requirements Engineering Technologies at Rome Laboratory*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA,Janvier 1993.

[Som85] I. Sommerville, "*Software Engineering*", second edition, Addison-Wesley Publishing Company, 1985.

[Sys89] "*Open CAM System, Dynamic Manufacturing Management and Control-Report on the results of Esprit Projet n° 418 Open CAM System*", Octobre 1989.

[Tar83] H. Tardieu, A. Rochfeld, R. Colletti, "*La méthode MERISE : principes et outils*", Les éditions d'Organisation, 1983.

[VDi89] N.W.P. Van Diepen, H. A. Partsch, "*Some Aspects of Formalising Informal Requirements*", University of Nijmegen, Department of Computer Science, Netherlands, 6 Septembre, 1989.

[Val91] B. Valespir, D. Chen, M. Zanettin, G. Dougmeingts, "*Definition of a CIM architecture within the Esprit 'IMPACS' project*" , in G. Dougmeingts, J. Browne and M. Tamljanovich, editors, "*Computer Applications in Production and Engineering, CAPE'91*", page 731- 738, Bordeaux, France, 10-12 Septembre 1991, IFIP, Elsevier Science Publishers B.V.

[Yu93] E.S.K. Yu, "*Modelling Organizations for Information Systems Requirements Engineering*", in RE'93, IEEE International Symposium On Requirements Engineering, San Diego, CA, USA Janvier 1993.

Facultés Universitaires Notre-Dame de la Paix
Institut d'Informatique
Rue Grandgagnage, 21b
B-5000 NAMUR

Spécification Formelle des Besoins pour un Système Productique

Annexes

Joaquin Garcia
Alexandre Gelenne

Promoteur : Eric Dubois

Mémoire présenté en vue
de l'obtention du grade de
Licencié et Maître en Informatique

Année académique 1992 - 1993

Annexe A

Spécification de l'atelier flexible

A.1. Introduction

Dans cette annexe, nous présentons l'ensemble des spécifications des agents composants l'atelier flexible du CRP-HT de Luxembourg. Par manque de place, nous ne donnerons pas les différents schémas reflétant les étapes de raffinements successifs, seuls les schémas complets seront donnés. Le lecteur intéressé par les structures de données pourra les trouver dans l'annexe C du présent document.

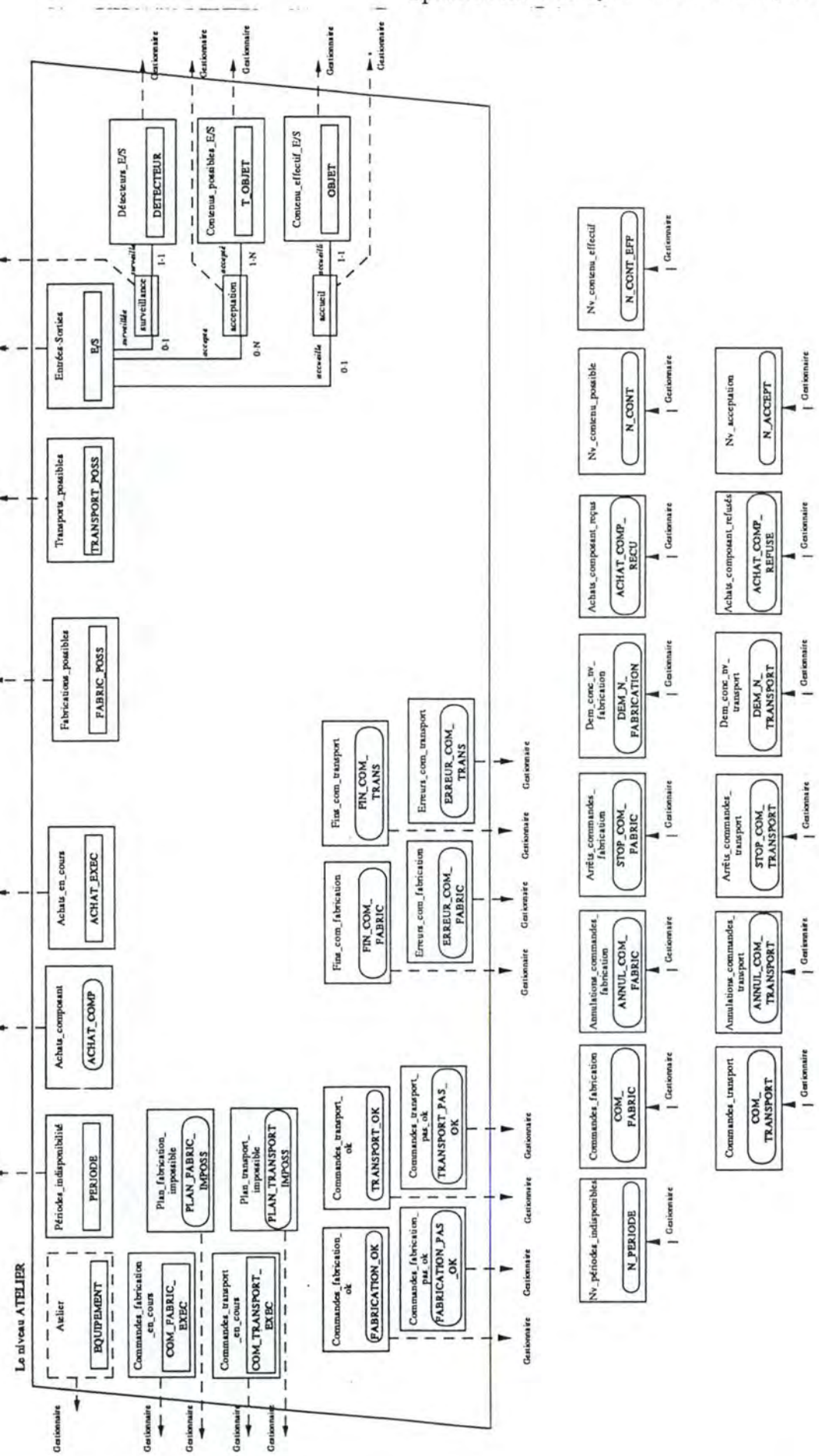
Afin de facilité autant que possible la lecture de cette annexe, nous avons adopté quelques conventions typographiques :

Normal	Ce texte est une description sommaire.
GRAS	Fait référence à un type d'entités, d'associations, type d'actions
ITALIC	Fait référence à un agent de la hiérarchie.
∅	La rubrique correspondante est vide.
☞	Renvoi le lecteur à l'endroit mentionné.

2. Le niveau Atelier

2.1. Structure d'état de l'agent

On trouvera à la figure suivante, la structure d'état de l'agent.


Le niveau ATELIER
Atelier
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Achats_composant
ACHAT_COMP
Achats_en_cours
ACHAT_EXEC
Fabrications_possibles
FABRIC_POSS
Transports_possibles
TRANSPORT_POSS
Entrées Sorties
E/S
Détecteurs_E/S
DETECTEUR
Contenu_possibles_E/S
T_OBJET
Contenu_effectif_E/S
OBJET
surveillance
acceptation
accueil
Commandes_fabrication_en_cours
COM_FABRIC_EXEC
Commandes_transport_en_cours
COM_TRANSPORT_EXEC
Plan_fabrication_impossible
PLAN_FABRIC_IMPOSS
Plan_transport_impossible
PLAN_TRANSPORT_IMPOSS
Commandes_fabrication_ok
FABRICATION_OK
Commandes_transport_ok
TRANSPORT_OK
Commandes_fabrication_pas_ok
FABRICATION_PAS_OK
Commandes_transport_pas_ok
TRANSPORT_PAS_OK
Fins_com_fabrication
FIN_COM_FABRIC
Fins_com_transport
FIN_COM_TRANS
Erreurs_com_fabrication
ERREUR_COM_FABRIC
Erreurs_com_transport
ERREUR_COM_TRANS
Nv_périodes_indisponibles
N_PERIODE
Commandes_fabrication
COM_FABRIC
Annulations_commandes_fabrication
ANNUL_COM_FABRIC
Arrêts_commandes_fabrication
STOP_COM_FABRIC
Dem_conc_nv_fabrication
DEM_N_FABRICATION
Achats_composant_reçus
ACHAT_COMP_RECU
Nv_contenu_possible
N_CONT
Nv_contenu_effectif
N_CONT_EFF
Commandes_transport
COM_TRANSPORT
Annulations_commandes_transport
ANNUL_COM_TRANSPORT
Arrêts_commandes_transport
STOP_COM_TRANSPORT
Dem_conc_nv_transport
DEM_N_TRANSPORT
Achats_composant_refusés
ACHAT_COMP_REFUSE
Nv_acceptation
N_ACCEPT
Gestionnaire

2.2. Description du schéma

2.2.1. Atelier

- ■ Description: Contient un descriptif du niveau atelier: localisation, description informelle, statut (marche, panne ou indisponible),...
- ■ Type: **EQUIPEMENT**
- ■ Lien: ∅
- ■ Visible par: *GESTIONNAIRE_ATELIER*
- ■ Structure: ☞ Annexe C

2.2.2. Périodes_indisponibilité

- ■ Description: Reprend des périodes durant lesquelles le niveau atelier est indisponible
- ■ Type: **PERIODE**
- ■ Lien: ∅
- ■ Visible par: *GESTIONNAIRE_ATELIER*
- ■ Structure: ☞ Annexe C

2.2.3. Achats_en_cours

- ■ Description: Contient une description des achats de composants en cours
- ■ Type: **ACHAT_EXEC**
- ■ Lien: ∅
- ■ Visible par: *GESTIONNAIRE_ATELIER*
- ■ Structure: ☞ Annexe C

2.2.4. Fabrications_possibles

- ■ Description: Reprend une description des fabrications possibles (que le niveau atelier peut effectuer) comprenant le type d'objet en input, le type d'objet en output, la durée théorique d'exécution,....
- ■ Type: **FABRIC_POSS**
- ■ Lien: ∅
- ■ Visible par: *GESTIONNAIRE_ATELIER*
- ■ Structure: ☞ Annexe C

2.2.5. Transports_possibles

- Description : Contient les informations sur l'ensemble des transports que l'atelier peut réaliser: origine, destination, type d'objet concerné,...
- Type : **TRANSPORT_POSS**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

2.2.6. Entrées-Sorties

- Description : Contient un descriptif des entrées-sorties du niveau atelier.
- Type : **E/S**
- Liens :
  - **Surveillance** : Relie les entrées-sorties aux détecteurs chargés de les surveiller.
  - **Acceptation** : Permet de donner les objets qu'un entrée-sortie accepte.
  - **Accueil** : donne le contenu effectif d'une entrée-sortie
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

2.2.7. Détecteurs_E/S

- Description : Reprend des informations sur les détecteurs chargés de surveiller les entrées-sorties du niveau atelier. Ces informations indiquent entre autres si une présence est détectée.
- Type : **DETECTEUR**
- Lien : • **Surveillance** : Voir ci-dessus
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

2.2.8. Contenus_possibles_E/S

- Description : Donne des informations sur les types d'objet qu'acceptent les entrées-sorties de l'atelier.
- Type : **T_OBJET**
- Lien : • **Acceptation** : Voir ci-dessus
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞Annexe C

2.2.9. Contenu_effectif_E/S

- Description : Contient des informations sur les objets qui se trouvent effectivement aux entrées-sorties du niveau atelier.
- Type : **OBJET**
- Lien : • **Accueil** : Voir ci-dessus
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞Annexe C

2.2.10. Commandes_fabrication_en_cours

- Description : Contient un descriptif des commandes de fabrication en cours d'exécution au sein du niveau atelier: type d'objet à fabriquer, quantité à fabriquer, moment où la réalisation de la commande a débuté, taux d'avancement, échéance,...
- Type : **COM_FABRIC_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞Annexe C

2.2.11. Commandes_transport_en_cours

- Description : Donne une série d'informations sur les commandes de transport en cours d'exécution dans l'atelier: entrée-sortie d'origine, entrée-sortie destination, objet à transporter, moment où l'on a lancé le transport, échéance,...
- Type : **COM_TRANSPORT_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞Annexe C

2.2.12. Achats_composant

- Description : Action qui permet de demander au gestionnaire l'achat de composants.
- Type : **ACHAT_COMP**
- Destination : *GESTIONNAIRE_ATELIER*
- Structure : ☞Annexe C

2.2.13. Plan_fabrication_impossible

- Description Signale au gestionnaire l'impossibilité de respecter le plan de fabrication initialement prévu.
- Type **PLAN_FABRIC_IMPOSS**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.14. Plan_transport_impossible

- Description Indique au gestionnaire qu'il est impossible de suivre le plan de transport
- Type **PLAN_TRANSPORT_IMPOSS**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.15. Commandes_fabrication_pas_ok

- Description Signale que la commande de fabrication indiquée est refusée.
- Type **FABRICATION_PAS_OK**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.16. Commandes_fabrication_ok

- Description Indique qu'une certaine commande est acceptée.
- Type **FABRICATION_OK**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.17. Commandes_transport_pas_ok

- Description Désigne les commandes de transport qu'il ne peut accepter
- Type **TRANSPORT_PAS_OK**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.18. Commandes_transport_ok

- Description Donne les commandes que l'atelier accepte.
- Type **TRANSPORT_OK**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.19. Fins_com_fabrication

- Description Permet de signaler que la commande de fabrication indiquée est terminée
- Type **FIN_COM_FABRIC**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.20. Fins_com_transport

- Description Indique qu'une certaine commande de transport a été exécutée.
- Type **FIN_COM_TRANS**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

2.2.21. Erreurs_com_fabrication

- Description Fait savoir au gestionnaire qu'une erreur a eu lieu et a empêché l'exécution de la commande de fabrication désignée.
- Type **ERREUR_COM_FABRIC**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

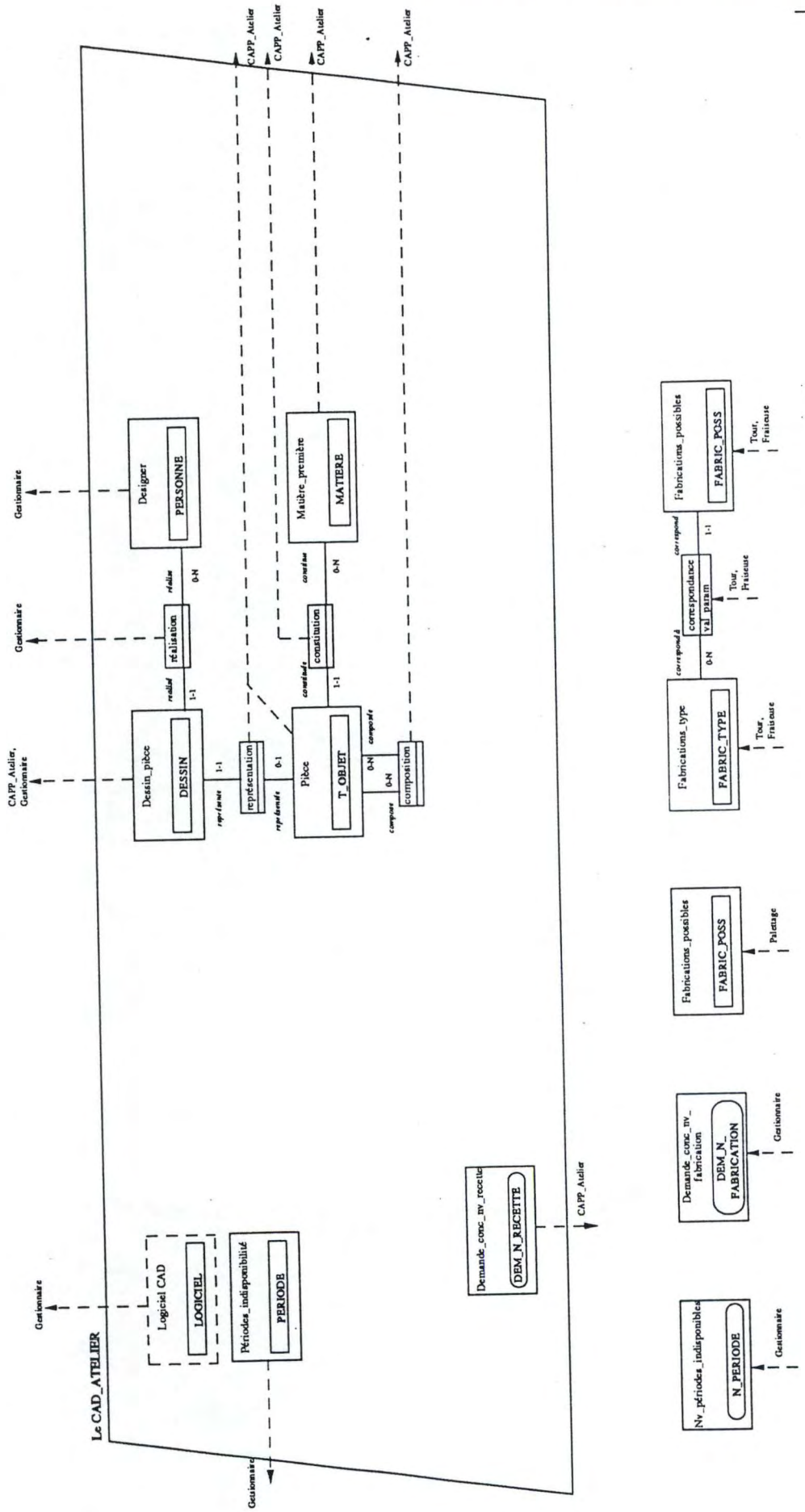
2.2.22. Erreurs_com_transport

- Description Signale qu'une erreur s'est produite lors de l'exécution de la commande de transport indiquée
- Type **ERREUR_COM_TRANS**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞Annexe C

3. Le CAD_ATELIER

3.1. Structure d'état de l'agent

On trouvera à la figure suivante, la structure d'état de l'agent CAD.


Le CAD_ATELIER
Logiciel CAD
LOGICIEL
Périodes_indisponibilité
PERIODE
Gestionnaire
CAPP_Atelier
CAPP_Atelier, Gestionnaire
Dessin_pièce
DESSIN
Designer
PERSONNE
réalisation
réalise
réalisé
0-N
1-1
représentation
représente
représenté
0-1
Pièce
T_OBJET
Matière_première
MATIERE
constitution
constitue
constitué
composition
composé
composer
Demande_conc_nv_recette
DEM_N_RECETTE
Demande_conc_nv_fabrication
DEM_N_FABRICATION
Nv_périodes_indisponibles
N_PERIODE
Fabrications_possibles
FABRIC_POSS
Palettage
Tour, Fraiseuse
Fabrications_type
FABRIC_TYPE
correspondance
val_param
correspond
0-N
1-1

3.2. Description du schéma

3.2.1. Périodes_indisponibilité

- Description : Donne les périodes durant lesquelles le CAD est indisponible.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

3.2.2. Logiciel CAD

- Description : Contient une description générale du logiciel qui aide le gestionnaire quand il assure les fonctions du CAD.
- Type : **LOGICIEL**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

3.2.3. Dessin_pièce

- Description : Donne un série d'informations sur les dessins existants (identifiant, dates de création et de mise à jour,...) .
- Type : **DESSIN**
- Liens :
  - **Réalisation** : Permet de désigner la personne qui a réalisé le dessin.
  - **Représentation** : Etablit le lien entre le dessin et la pièce qu'elle représente.
- Visible par : *CAPP_ATELIER, GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

3.2.4. Pièce

- Description : Donne le descriptif des pièces connues du niveau.
- Type : **T_OBJET**
- Liens :
  - **Représentation** : Voir ci-dessus.
  - **Composition** : désigne les pièces qui en composent d'autres.
  - **Constitution** : définit la matière première de la pièce.
- Visible par : *CAPP_ATELIER*
- Structure : ☞ Annexe C

3.2.5. Matière_première

- Description : Donne une série d'informations (dont une description informelle, quelques formules de calcul concernant, par exemple, la résistance de la matière,...) sur les matières premières connues.
- Type : **MATIERE**
- Lien : • **Constitution** : voir ci-dessus.
- Visible par : *CAPP_ATELIER*
- Structure : ☞ Annexe C

3.2.6. Designer

- Description : reprend un descriptif des personnes connues.
- Type : **PERSONNE**
- Lien : • **Réalisation** : Voir ci-dessus.
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

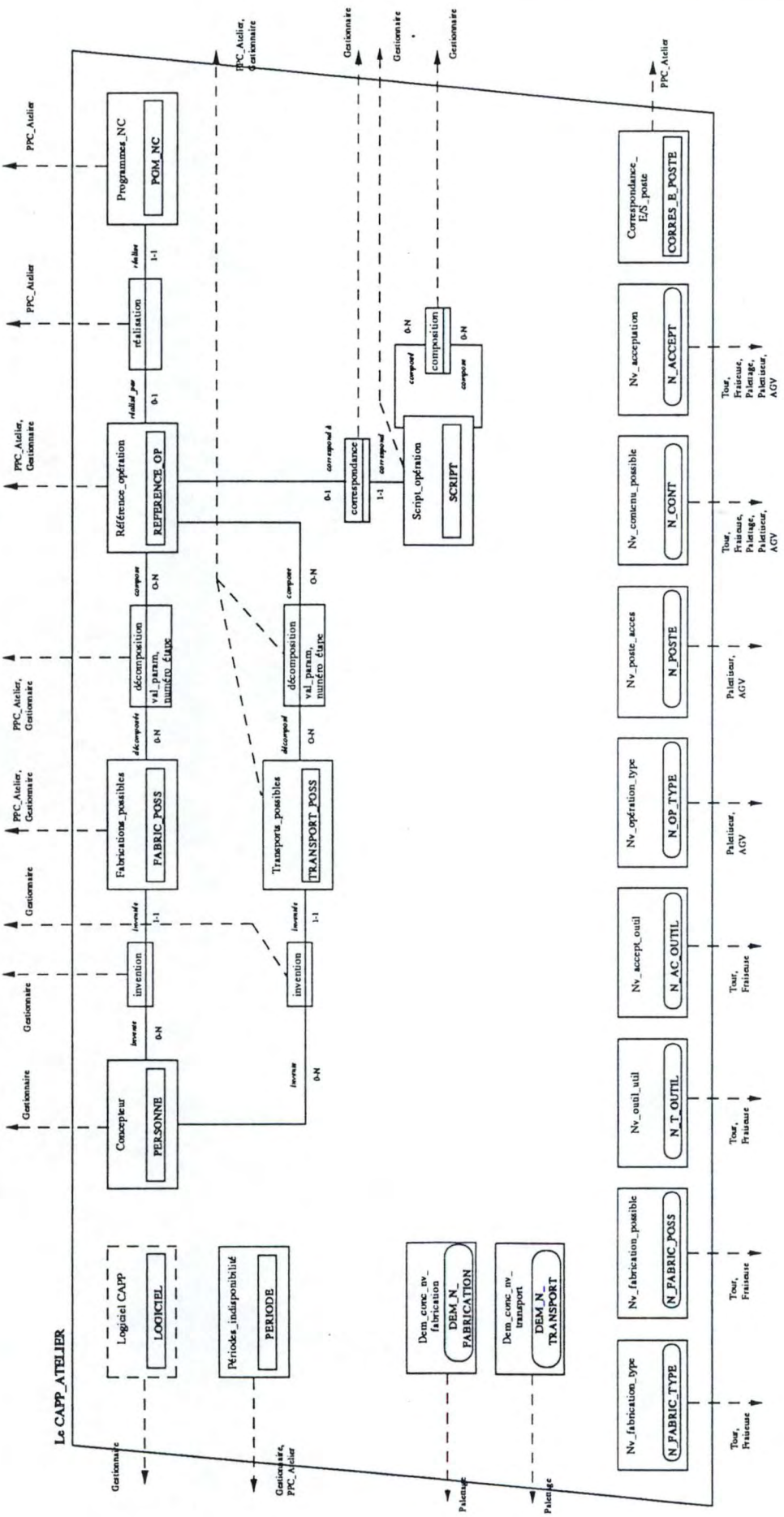
3.2.7. Demande_conc_nouv_recette

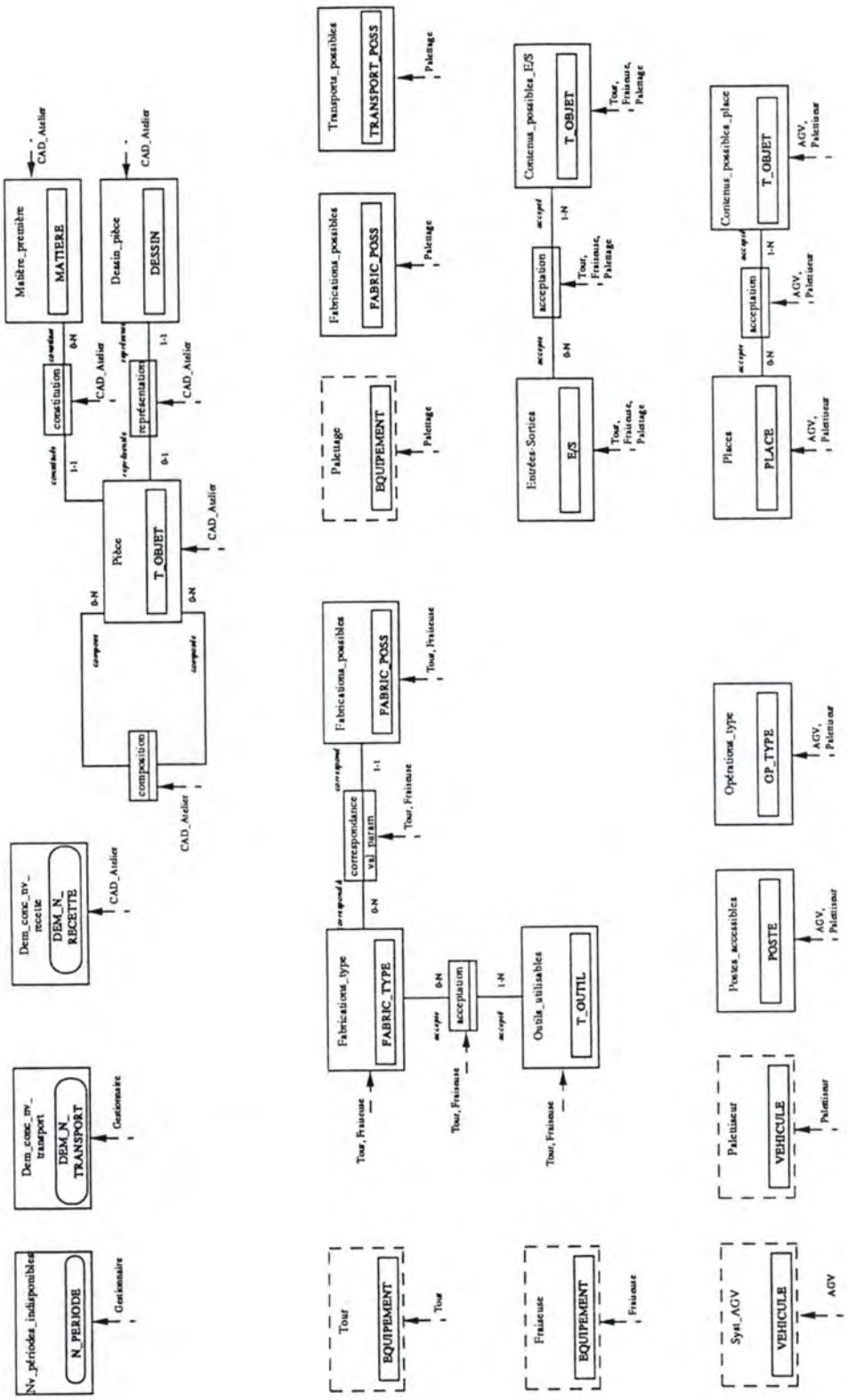
- Description : Demande au CAPP de concevoir la recette de fabrication pour une pièce dont on vient de terminer le design.
- Type : **DEM_N_RECETTE**
- Origine : *CAPP_ATELIER*
- Structure : ☞ Annexe C

4. Le CAPP_Atelier

4.1. Structure d'état de l'agent

On trouvera à la figure suivante, la structure d'état de l'agent CAPP_Atelier.


Le CAPP_ATELIER
Logiciel CAPP
LOGICIEL
Périodes_indisponibilité
PERIODE
Concepteur
PERSONNE
invention
Fabrications_possibles
FABRIC_POSS
Transports_possibles
TRANSPORT_POSS
décomposition
val_param, numéro_étape
Référence_opération
REFERENCE_OP
correspondance
Script_opération
SCRIPT
composition
réalisation
Programmes_NC
POM_NC
Correspondance_E/S_poste
CORRES_E_POSTE
Nv_acceptation
N_ACCEPT
Nv_contenu_possible
N_CONT
Nv_poste_accès
N_POSTE
Nv_opération_type
N_OP_TYPE
Nv_accept_outil
N_AC_OUTIL
Nv_outil_util
N_T_OUTIL
Nv_fabrication_possible
N_FABRIC_POSS
Nv_fabrication_type
N_FABRIC_TYPE
Dem_conc_nv_fabrication
DEM_N_FABRICATION
Dem_conc_nv_transport
DEM_N_TRANSPORT
Gestionnaire
PPC_Atelier
Tour, Fraiseuse
Tour, Fraiseuse, Perçage, Palettiseur, AGV
Palettiseur, AGV
Perçage


Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
Dem_conc_nv_transport
DEM_N_TRANSPORT
Gestionnaire
Dem_conc_nv_recette
DEM_N_RECETTE
CAD_Atelier
Matière_première
MATIERE
CAD_Atelier
Dessin_pièce
DESSIN
CAD_Atelier
constitution
représentation
Pièce
T_OBJET
CAD_Atelier
composition
Transports_possibles
TRANSPORT_POSS
Palettage
Fabrications_possibles
FABRIC_POSS
Palettage
Palettage
EQUIPEMENT
Palettage
Contenus_possibles_E/S
T_OBJET
Tour, Fraiseuse, Palettage
acception
Tour, Fraiseuse, Palettage
Entrées-Sorties
E/S
Tour, Fraiseuse, Palettage
Contenus_possibles_place
T_OBJET
AGV, Palettiseur
acception
AGV, Palettiseur
Places
PLACE
AGV, Palettiseur
Fabrications_possibles
FABRIC_POSS
Tour, Fraiseuse
correspondance
val_param
Tour, Fraiseuse
Fabrications_type
FABRIC_TYPE
acception
Tour, Fraiseuse
Outils_utilisables
T_OUTIL
Tour, Fraiseuse
Opérations_type
OP_TYPE
AGV, Palettiseur
Postes_accessibles
POSTE
AGV, Palettiseur
Palettiseur
VEHICULE
Palettiseur
Tour
EQUIPEMENT
Tour
Fraiseuse
EQUIPEMENT
Fraiseuse
Syst_AGV
VEHICULE
AGV

4.2. Description du schéma

4.2.1.. Logiciel CAPP

- Description : Contient une description générale du logiciel qui aide le gestionnaire quand il assure les fonctions du CAPP.
- Type : **LOGICIEL**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

4.2.2. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes durant lesquelles le CAPP_Atelier sera indisponible.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

4.2.3. Concepteur

- Description : Contient les informations sur les personnes qui ont conçu les recettes ou les transports du niveau atelier.
- Type : **PERSONNE**
- Lien :
  - **Invention** : permet de désigner la personne qui a conçu une fabrication possible ou un transport possible.
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

4.2.4. Fabrications_possibles

- Description : Contient un descriptif des fabrications possibles que le niveau atelier peut réaliser (identifiant, description informelle, durée théorique d'exécution, type d'objet en input et type d'objet en output).
- Type : **FABRIC_POSS**
- Liens :
  - **Décomposition** : Permet de décomposer une fabrication possible en une suite d'opérations de niveau inférieur. Ces opérations sont aussi bien des opérations de fabrication que de déplacement.
    Remarque: l'ensemble des valeurs de paramètres est vide, quand on se réfère à des fabrications réalisées par le tour ou la fraiseuse.
  - **Invention** : Voir ci-dessus.
- Visible par : *GESTIONNAIRE_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

4.2.5. Transports_possibles

- Description : Contient les informations (origine, destination, type d'objet transporté) concernant les transports qu'il est possible de réaliser au sein du niveau atelier
- Type : **TRANSPORT_POSS**
- Liens :
  - **Décomposition** : Permet de décomposer un transport possible en une suite d'opérations de niveau inférieur.
  - **Invention** : Voir ci-dessus.
- Visible par : *GESTIONNAIRE_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

4.2.6. Script_opération

- Description : Contient les informations sur les scripts existants.
- Type : **SCRIPT**
- Liens :
  - **Correspondance** : effectue le lien entre les opérations réalisables par les machines NC et les scripts correspondants. Un script est un texte rédigé dans un pseudo-code à partir duquel le logiciel CAPP génère des programmes numériques.
  - **Composition** : Permet de déterminer les scripts appelés dans d'autres.
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

4.2.7. Programmes_NC

- Description : Décrit les programmes NC existants.
- Type : **PGM_NC**
- Lien :
  - **Composition** : Etablit le lien entre les opérations réalisables par les machines NC et les programmes numériques correspondants.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

4.2.8. Référence_opération

- Description : Contient les références à des opérations que peuvent réaliser machines numériques, niveau palettage et moyens de transport.
- Type : **REFERENCE_OP**
- Liens :
  - **Décomposition** : Voir ci-dessus.
  - **Correspondance** : Voir ci-dessus.
  - **Réalisation** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

4.2.9. Dem_conc_nv_fabrication

- Description : Permet de demander au niveau palettage de concevoir une nouvelle fabrication possible.
- Type : **DEM_N_FABRICATION**
- Destination : *PALETTAGE*
- Structure : ☞ Annexe C

4.2.10. Dem_conc_nv_transport

- Description Permet de demander au niveau palettage de concevoir un nouveau transport possible.
- Type **DEM_N_TRANSPORT**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

4.2.11. Nv_fabrication_type

- Description Signale au tour ou à la fraiseuse qu'ils disposent d'une nouvelle fabrication type.
- Type **N_FABRIC_TYPE**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

4.2.12. Nv_fabrication_possible

- Description Signale au tour ou à la fraiseuse qu'ils peuvent exécuter une nouvelle fabrication posssible (càd une fabrication-type pour laquelle les valeurs de paramètres sont définies).
- Type **N_FABRIC_POSS**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

4.2.13. Nv_outil_util

- Description Signale aux machines numériques qu'elles disposent d' un nouve outil.
- Type **N_T_OUTIL**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

4.2.14. Nv_accept_outil

- Description Etablit les liens entre les fabrications type des machines NC et les outils qu'elles peuvent utiliser.
- Type **N_AC_OUTIL**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

4.2.15. Nv_opération_type

- Description Signale à l'AGV ou au palettiseur qu'ils disposent d'une nouvelle opération-type.
- Type **N_OP_TYPE**
- Destination *AGV, PALETTISEUR*
- Structure ☞ Annexe C

4.2.16. Nv_poste_acces

- Description Fait savoir à l'AGV ou au palettiseur qu'ils ont accès à de nouveaux postes.
- Type **N_POSTE**
- Destination *PALETTISEUR, AGV*
- Structure ☞ Annexe C

4.2.17. Nv_contenu_possible

- Description Indique au tour, fraiseuse,... qu'ils peuvent accueillir un nouveau contenu à leurs des entrées-sorties ou sur une certaine place.
- Type **N_CONT**
- Destination *TOUR, FRAISEUSE, PALETTAGE AGV, PALETTISEUR*
- Structure ☞ Annexe C

4.2.18. Nv_acceptation

- Description Fait savoir à l'AGV, à la fraiseuse,... que telle place ou telle entrée-sortie peut accueillir tel type d'objet.
- Type **N_ACCEPT**
- Destination *TOUR, FRAISEUSE, PALETTAGE AGV, PALETTISEUR*
- Structure ☞ Annexe C

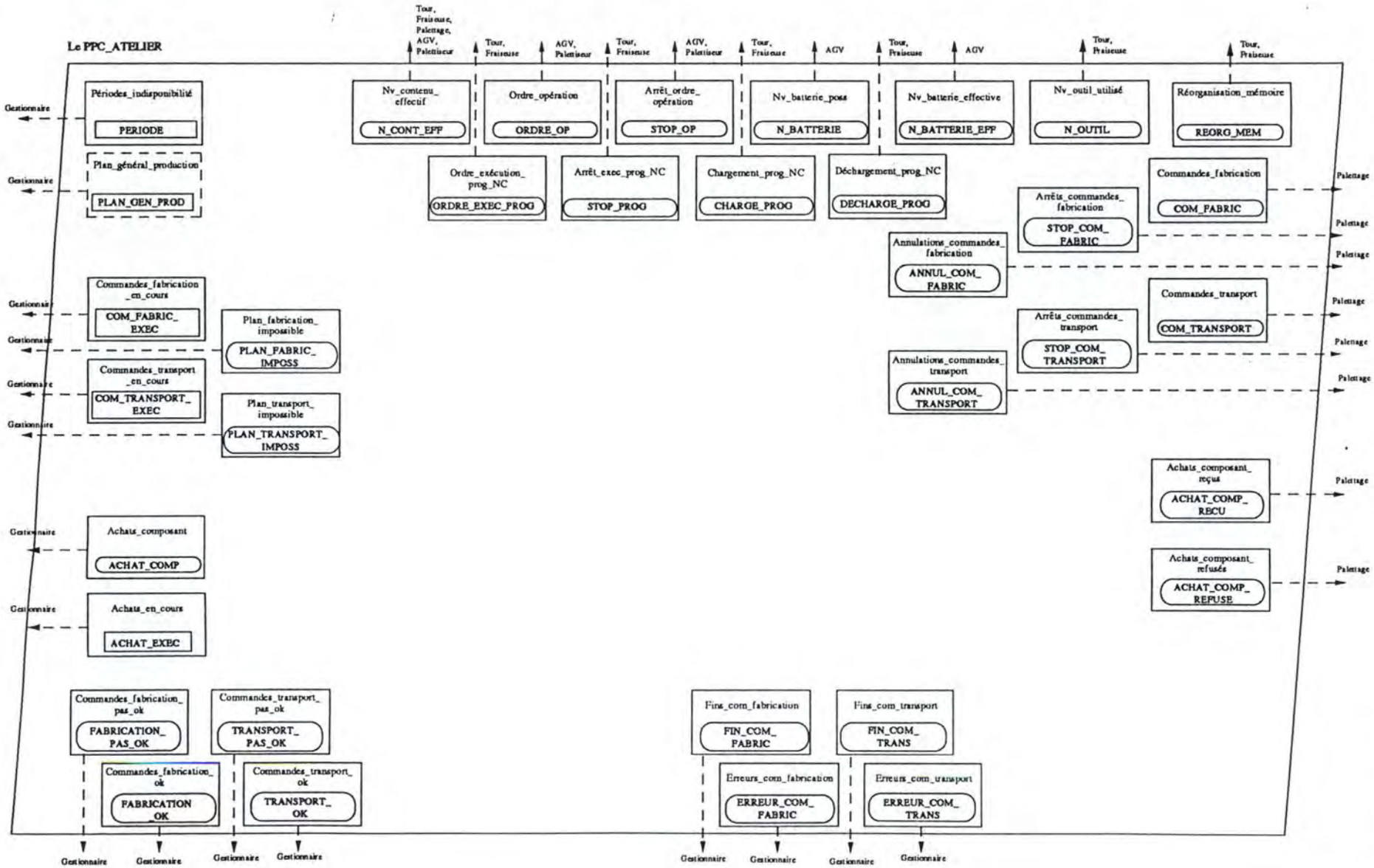
4.2.19. Correspondance_E/S_poste

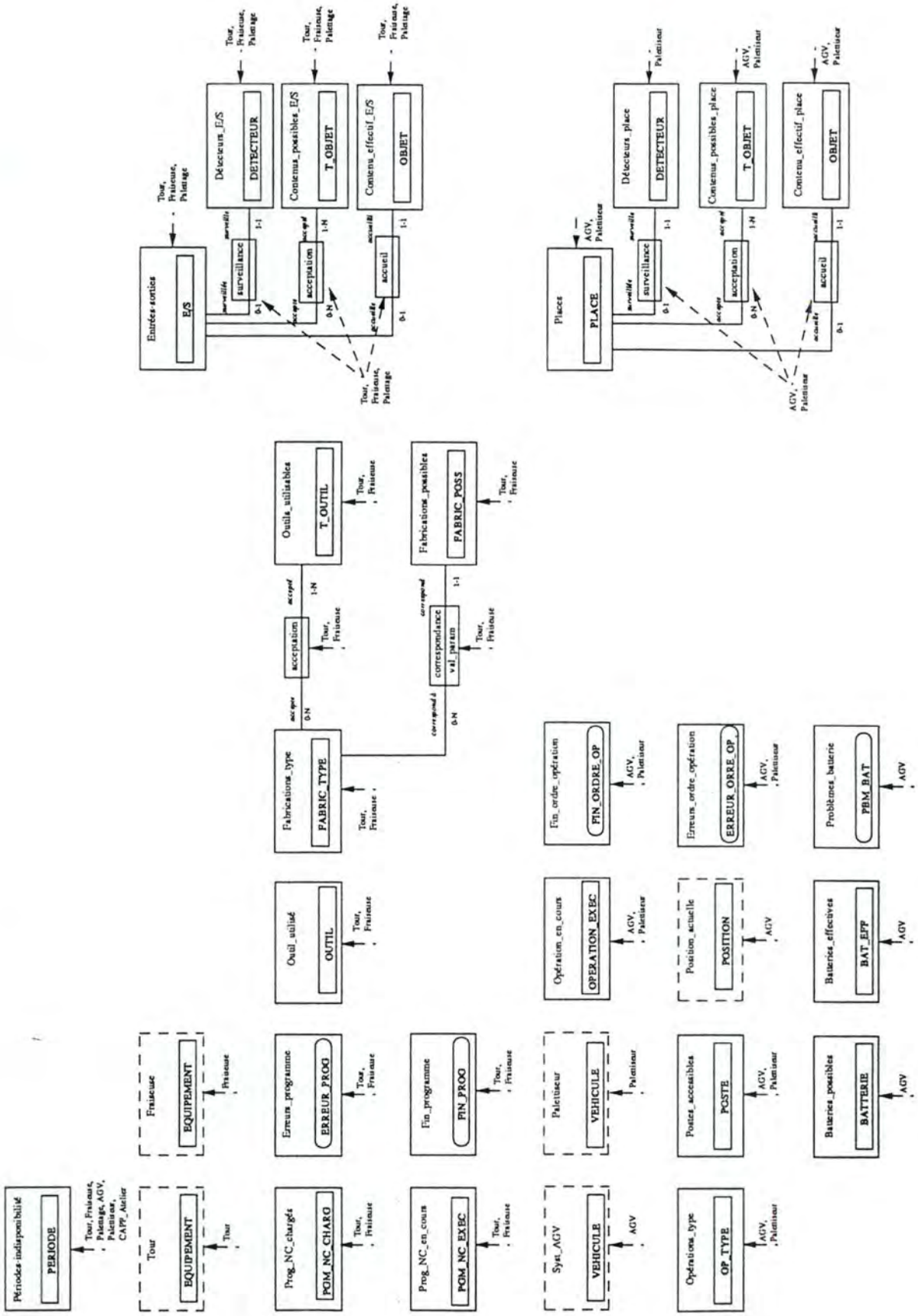
- Description Permet de désigner les postes à partir desquels on peut accéder à certaines entrées-sorties. Il s'agit en fait d'une liste constituée d'une référence à un poste existant et d'une référence à une entrée-sortie
- Type **CORRES_E_POSTE**
- Lien ∅
- Visible par *PPC_ATELIER*
- Structure ☞ Annexe C

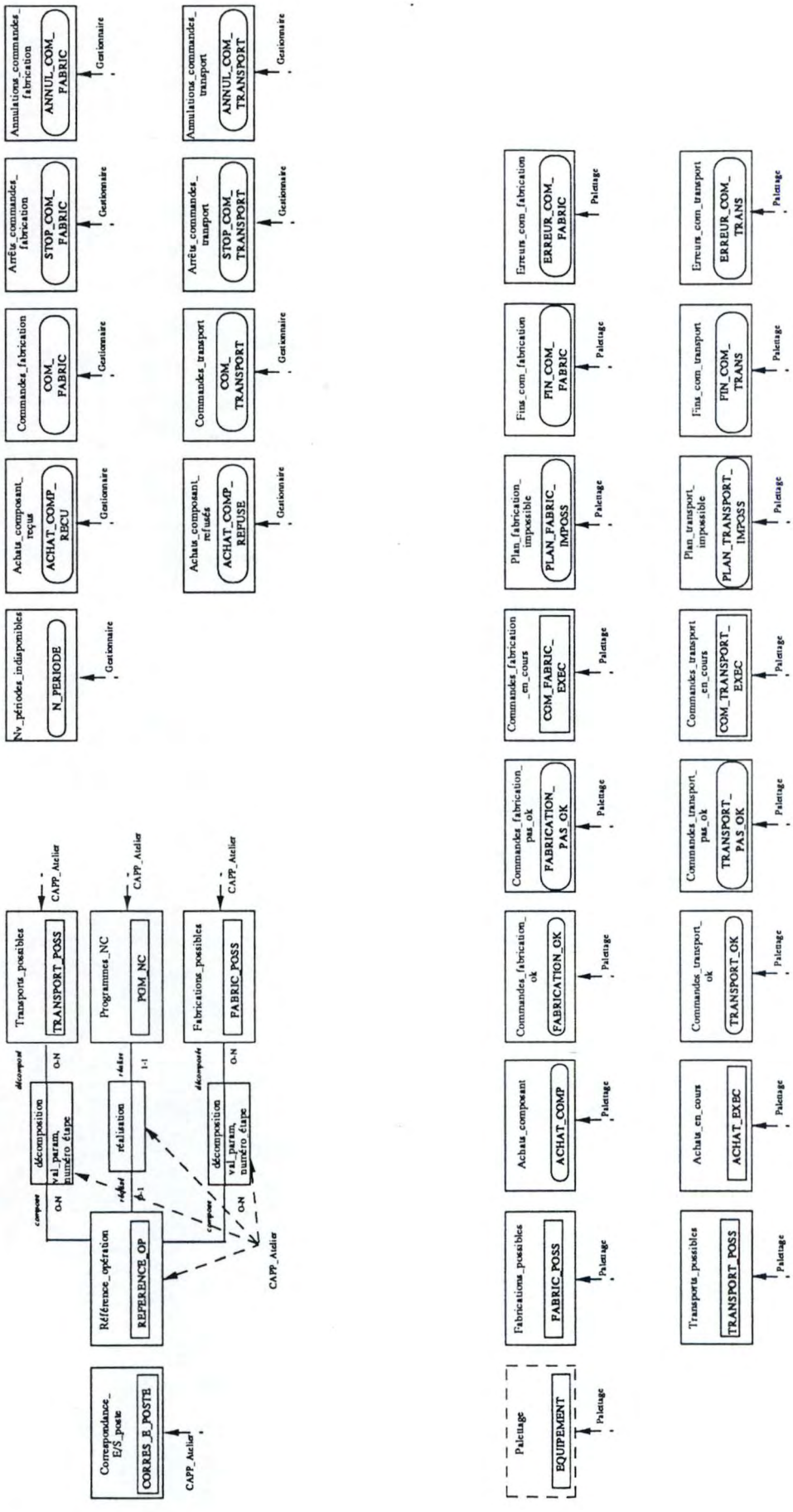
5. Le PPC_ATELIER :

5.1. Structure d'état de l'agent

On trouvera à la figure suivante, la structure d'état de l'agent PPC_Atelier.


Le PPC_ATELIER
Périodes_indisponibilité
PERIODE
Plan_général_production
PLAN_GEN_PROD
Commandes_fabrication_en_cours
COM_FABRIC_EXEC
Commandes_transport_en_cours
COM_TRANSPORT_EXEC
Achats_composant
ACHAT_COMP
Achats_en_cours
ACHAT_EXEC
Nv_contenu_effectif
N_CONT_EFF
Ordre_opération
ORDRE_OP
Arrêt_ordre_opération
STOP_OP
Nv_batterie_poss
N_BATTERIE
Nv_batterie_effective
N_BATTERIE_EFF
Nv_outil_utilisé
N_OUTIL
Réorganisation_mémoire
REORG_MEM
Ordre_exécution_prog_NC
ORDRE_EXEC_PROG
Arrêt_exec_prog_NC
STOP_PROG
Chargement_prog_NC
CHARGE_PROG
Déchargement_prog_NC
DECHARGE_PROG
Commandes_fabrication
COM_FABRIC
Arrêts_commandes_fabrication
STOP_COM_FABRIC
Annulations_commandes_fabrication
ANNUL_COM_FABRIC
Commandes_transport
COM_TRANSPORT
Arrêts_commandes_transport
STOP_COM_TRANSPORT
Annulations_commandes_transport
ANNUL_COM_TRANSPORT
Achats_composant_reçus
ACHAT_COMP_RECU
Achats_composant_refusés
ACHAT_COMP_REFUSE
Plan_fabrication_impossible
PLAN_FABRIC_IMPOSS
Plan_transport_impossible
PLAN_TRANSPORT_IMPOSS
Commandes_fabrication_pas_ok
FABRICATION_PAS_OK
Commandes_transport_pas_ok
TRANSPORT_PAS_OK
Commandes_fabrication_ok
FABRICATION_OK
Commandes_transport_ok
TRANSPORT_OK
Fins_com_fabrication
FIN_COM_FABRIC
Fins_com_transport
FIN_COM_TRANS
Erreurs_com_fabrication
ERREUR_COM_FABRIC
Erreurs_com_transport
ERREUR_COM_TRANS
Tour, Fraiseuse, Palettage, AGV, Palettiseur
Tour, Fraiseuse
AGV, Palettiseur
AGV
Palettage
Gestionnaire


Entrées-sorties
E/S
Détecteurs_E/S
DETECTEUR
Contenus_possibles_E/S
T_OBJET
Contenu_effectif_E/S
OBJET
surveillance
acceptation
accueil
Places
PLACE
Détecteurs_place
Contenus_possibles_place
Contenu_effectif_place
Outils_utilisables
T_OUTIL
Fabrications_possibles
FABRIC_POSS
Fabrications_type
FABRIC_TYPE
correspondance
val_param
Outil_utilisé
OUTIL
Fin_ordre_opération
FIN_ORDRE_OP
Opération_en_cours
OPERATION_EXEC
Erreurs_ordre_opération
ERREUR_ORRE_OP
Position_actuelle
POSITION
Problèmes_batterie
PBM_BAT
Batteries_effectives
BAT_EFP
Erreurs_programme
ERREUR_PROG
Fin_programme
FIN_PROG
Paletiseur
VEHICULE
Postes_accessibles
POSTE
Batteries_possibles
BATTERIE
Périodes_indisponibilité
PERIODE
Tour
EQUIPEMENT
Prog_NC_chargés
PGM_NC_CHARG
Prog_NC_en_cours
PGM_NC_EXEC
Syst_AGV
Opérations_type
OP_TYPE


Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
Achats_composant_reçus
ACHAT_COMP_RECU
Achats_composant_refusés
ACHAT_COMP_REFUSE
Commandes_fabrication
COM_FABRIC
Commandes_transport
COM_TRANSPORT
Arrêts_commandes_fabrication
STOP_COM_FABRIC
Arrêts_commandes_transport
STOP_COM_TRANSPORT
Annulations_commandes_fabrication
ANNUL_COM_FABRIC
Annulations_commandes_transport
ANNUL_COM_TRANSPORT
Transports_possibles
TRANSPORT_POSS
Programmes_NC
POM_NC
Fabrications_possibles
FABRIC_POSS
CAPP_Atelier
décomposition
val_param, numéro étape
réalisation
Référence_opération
REFERENCE_OP
Correspondance_E/S_poste
CORRES_E_POSTE
O-N
1-1
Paletage
Achats_composant
ACHAT_COMP
Achats_en_cours
ACHAT_EXEC
Commandes_fabrication_ok
FABRICATION_OK
Commandes_transport_ok
TRANSPORT_OK
Commandes_fabrication_pas_ok
FABRICATION_PAS_OK
Commandes_transport_pas_ok
TRANSPORT_PAS_OK
Commandes_fabrication_en_cours
COM_FABRIC_EXEC
Commandes_transport_en_cours
COM_TRANSPORT_EXEC
Plan_fabrication_impossible
PLAN_FABRIC_IMPOSS
Plan_transport_impossible
PLAN_TRANSPORT_IMPOSS
Fins_com_fabrication
FIN_COM_FABRIC
Fins_com_transport
FIN_COM_TRANS
Erreurs_com_fabrication
ERREUR_COM_FABRIC
Erreurs_com_transport
ERREUR_COM_TRANS
Equipement
EQUIPEMENT

5.2. Description du schéma de l'agent

5.2.1. Périodes_indisponibilité

- Description : Reprend des intervalles de dates durant lesquelles le *PPC_ATELIER* est indisponible.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

5.2.2. Plan_général_production

- Description : Contient les informations concernant le plan général de production du niveau palettage atelier, élaboré après réception des commandes.
- Type : **PLAN_GEN_PROD**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

5.2.3. Commandes_fabrication_en_cours

- Description : Reprend des descriptifs des commandes de fabrication acceptées par le niveau atelier.
- Type : **COM_FABRIC_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

5.2.4. Commandes_transport_en_cours

- Description : Contient les informations au sujet des commandes de transport acceptées.
- Type : **COM_TRANSPORT_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

5.2.5. Achats_en_cours

- Description : Contient une description des achats de composants en cours
- Type : **ACHAT_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE_ATELIER*
- Structure : ☞ Annexe C

5.2.6. Achats_composant

- Description Action qui permet de demander au gestionnaire l'achat de composants
- Type **ACHAT_COMP**
- Destination *GESTIONNAIRE_ATELIER*
- Structure ☞ Annexe C

5.2.7. Nv_contenu_effectif

- Description Indique au tour, fraiseuse,... le contenu effectif d'une certaine entrée-sortie..
- Type **N_CONT_EFF**
- Destination *TOUR, FRAISEUSE, PALETTAGE, PALETTISEUR, AGV*
- Structure ☞ Annexe C

5.2.8. Ordre_opération

- Description Demande à un moyen de transport d'exécuter une certaine opération.
- Type **ORDRE_OP**
- Destination *AGV, PALETTISEUR*
- Structure ☞ Annexe C

5.2.9. Arrêt_ordre_opération

- Description Demande à un moyen de transport de stopper l'opération en cours d'exécution.
- Type **STOP_OP**
- Destination *AGV, PALETTISEUR*
- Structure ☞ Annexe C

5.2.10. Nv_batterie_poss

- Description Signale à l'AGV qu'il peut utiliser une nouvelle batterie.
- Type **N_BATTERIE**
- Destination *AGV*
- Structure ☞ Annexe C

5.2.11. Nv_batterie_effective

- Description Indique à l'AGV quelle est la batterie branchée à l'endroit spécifié
- Type **N_BATTERIE_EFF**
- Destination *AGV*
- Structure ☞ Annexe C

5.2.12. Nv_outil_utilisé

- Description Indique au tour (ou à la fraiseuse) l'outil qu'il (elle) utilise effectivement.
- Type **N_OUTIL**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.13. Réorganisation_mémoire

- Description Demande à la machine NC de réorganiser le contenu de sa mémoire afin de d'éliminer toute perte de place.
- Type **REORG_MEM**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.14. Ordre_exécution_prog_NC

- Description Ordonne l'exécution d'un programme chargé dans la mémoire d'une des machines NC.
- Type **ORDRE_EXEC_PROG**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.15. Arrêt_exec_prog_NC

- Description Demande l'arrêt d'un programme en cours d'exécution
- Type **STOP_PROG**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.16. Chargement_prog_NC

- Description Effectue le chargement d'un programme numérique dans une des machines.
- Type **CHARGE_PROG**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.17. Déchargement_prog_NC

- Description Décharge certains programmes de la mémoire des machines NC.
- Type **DECHARGE_PROG**
- Destination *TOUR, FRAISEUSE*
- Structure ☞ Annexe C

5.2.18. Commandes_fabrication

- Description Envoie une commande de fabrication au niveau palettage
- Type **COM_FABRIC**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.19. Arrêts_commandes_fabrication

- Description Demande au niveau palettage d'arrêter la production de la commande désignée.
- Type **STOP_COM_FABRIC**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.20. Annulations_commandes_fabrication

- Description Demande au niveau palettage d'arrêter la production de la commande désignée et de revenir dans l'état qui précédait le début de son exécution.
- Type **ANNUL_COM_FABRIC**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.21. Commandes_transport

- Description Envoie une commande de transport au niveau palettage.
- Type **COM_TRANSPORT**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.22. Arrêts_commandes_transport

- Description Demande l' arrêt de l'exécution de la commande de transport précisée.
- Type **STOP_COM_TRANSPORT**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.23. Annulations_commandes_transport

- Description Demande l'annulation (càd un arrêt suivi d'un retour à l'état initial) de la commande désignée
- Type **ANNUL_COM_TRANS**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.24. Plan_fabrication_impossible

- Description Signale au gestionnaire l'impossibilité de respecter le plan de fabrication.
- Type **PLAN_FABRIC_IMPOSS**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.25. Plan_transport_impossible

- Description Indique au gestionnaire qu'il est impossible de suivre le plan de transport.
- Type **PLAN_TRANSPORT_IMPOSS**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.26. Commandes_fabrication_pas_ok

- Description Désigne les commandes qu'il ne peut accepter.
- Type **FABRICATION_PAS_OK**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.27. Commandes_transport_pas_ok

- Description Signale que la commande de transport désignée n'est pas acceptée.
- Type **TRANSPORT_PAS_OK**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.28. Commandes_fabrication_ok

- Description Désigne les commandes que l'atelier accepte de produire.
- Type **FABRICATION_OK**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.29. Commandes_transport_ok

- Description Indique les commandes de transport acceptées.
- Type **TRANSPORT_OK**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.30. Fins_com_fabrication

- Description Indique au gestionnaire les commandes qui sont terminées
- Type **FIN_COM_FABRIC**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.31. Fins_com_transport

- Description Signale au gestionnaire les commandes de transport terminées.
- Type **FIN_COM_TRANS**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.32. Erreurs_com_fabrication

- Description Communique au gestionnaire qu'une erreur s'est produite lors de l'exécution de la commandes indiquée.
- Type **ERREUR_COM_FABRIC** (Action Interne)
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.33. Erreurs_com_transport

- Description Fait savoir au gestionnaire qu'il y a eu une erreur lors de l'exécution d'une commande de transport.
- Type **ERREUR_COM_TRANS**
- Destination *GESTIONNAIRE*
- Structure ☞ Annexe C

5.2.34. Achats_composant_reçus

- Description Indique au palettage que les composants sont arrivés suite à une demande d'achat.
- Type **ACHAT_COMP_RECU**
- Destination *PALETTAGE*
- Structure ☞ Annexe C

5.2.35. Achats_composant_refusés

- Description Prévient le niveau palettage du refus de répondre à la demande d'achat indiquée
- Type **ACHAT_COMP_REFUS** (Action Interne)
- Destination *PALETTAGE*
- Structure ☞ Annexe C

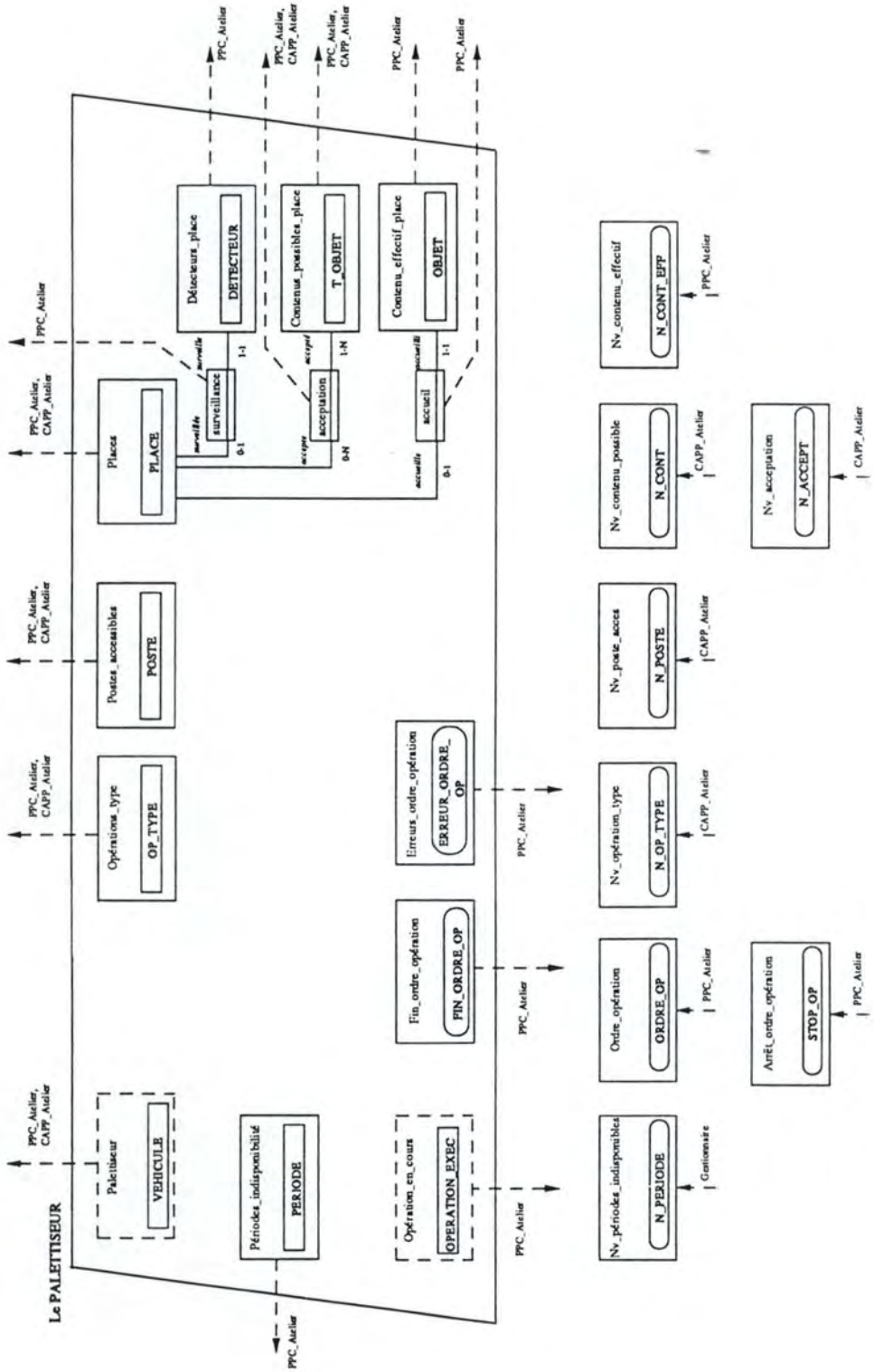
6. Système AGV

Nous renvoyons le lecteur au chapitre 5, où le système AGV a déjà fait l'objet d'une description.

7. Palettiseur

7.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent Palettiseur.


Le PALETTISEUR
PPC_Atelier
PPC_Atelier, CAPP_Atelier
Palettiseur
VEHICULE
Périodes_indisponibilité
PERIODE
Opération_en_cours
OPERATION_EXEC
Places
PLACE
Postes_accessibles
POSTE
Opérations_type
OP_TYPE
Détecteurs_place
DETECTEUR
surveillance
Contenus_possibles_place
T_OBJET
acceptation
Contenu_effectif_place
OBJET
accueil
Erreurs_ordre_opération
ERREUR_ORDRE_OP
Fin_ordre_opération
FIN_ORDRE_OP
Ordre_opération
ORDRE_OP
Arrêt_ordre_opération
STOP_OP
Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
Nv_opération_type
N_OP_TYPE
Nv_poste_acces
N_POSTE
Nv_contenu_possible
N_CONT
Nv_contenu_effectif
N_CONT_EFF
Nv_acceptation
N_ACCEPT

7.2. Description du schéma

7.2.1. Palettiseur

- Description : Contient un descriptif du palettiseur.
- Type : **VEHICULE**
- Lien : ∅
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

7.2.2. Périodes_indisponibilité

- Description : Contient les informations sur les périodes d'indisponibilité du palettiseur.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

7.2.3.Opérations_type

- Description : Contient les informations sur les opérations que propose le palettiseur.
  Exemples: Chargement, déchargement,...
- Type : **OP_TYPE**
- Lien : ∅
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

7.2.4.Postes_accessibles

- Description : Contient une description des postes auquels le palettiseur a accès
- Type : **POSTE**
- Lien : ∅
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

7.2.5. Places

- Description : Contient une description des places du palettiseur.
- Type : **PLACE**
- Liens :
  - **Surveillance** : Indique le détecteur associé à chacune des places.
  - **Acceptation** : Permet de déterminer les objets que la place peut accepter.
  - **Accueil** : Détermine le contenu effectif de chaque place à un moment donné.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

7.2.6. Détecteurs_place

- Description : Contient les informations sur chacun des détecteurs.
- Type : **DETECTEUR**
- Lien :
  - **Surveillance** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

7.2.7. Contenus_possibles_place

- Description : Contient un descriptif des types d'objets que les places acceptent.
- Type : **T_OBJET**
- Lien :
  - **Acceptation :** Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

7.2.8. Contenu_effectif_place

- Description : Donne une description d'un objet (type d'objet auquel l'objet appartient, dimension et poids réels...)situé sur une des places du palettiseur.
- Type : **OBJET**
- Lien :
  - **Accueil** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

7.2.9. Opération_en_cours

- Description : Contient les informations suivantes sur l'opération exécutée: référence à l'opération-type exécutée, valeur des paramètres, heure de lancement de l'exécution, taux d'avancement de celle-ci.
- Type : **OPERATION_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

7.2.10. Fin_ordre_opération

- Description : Signale la fin de l'exécution d'une opération, donne l'heure à laquelle elle a commencé et la durée réelle d'exécution.
- Type : **FIN_ORDRE_OP**
- Destination : *PPC_ATELIER*
- Structure : ☞ Annexe C

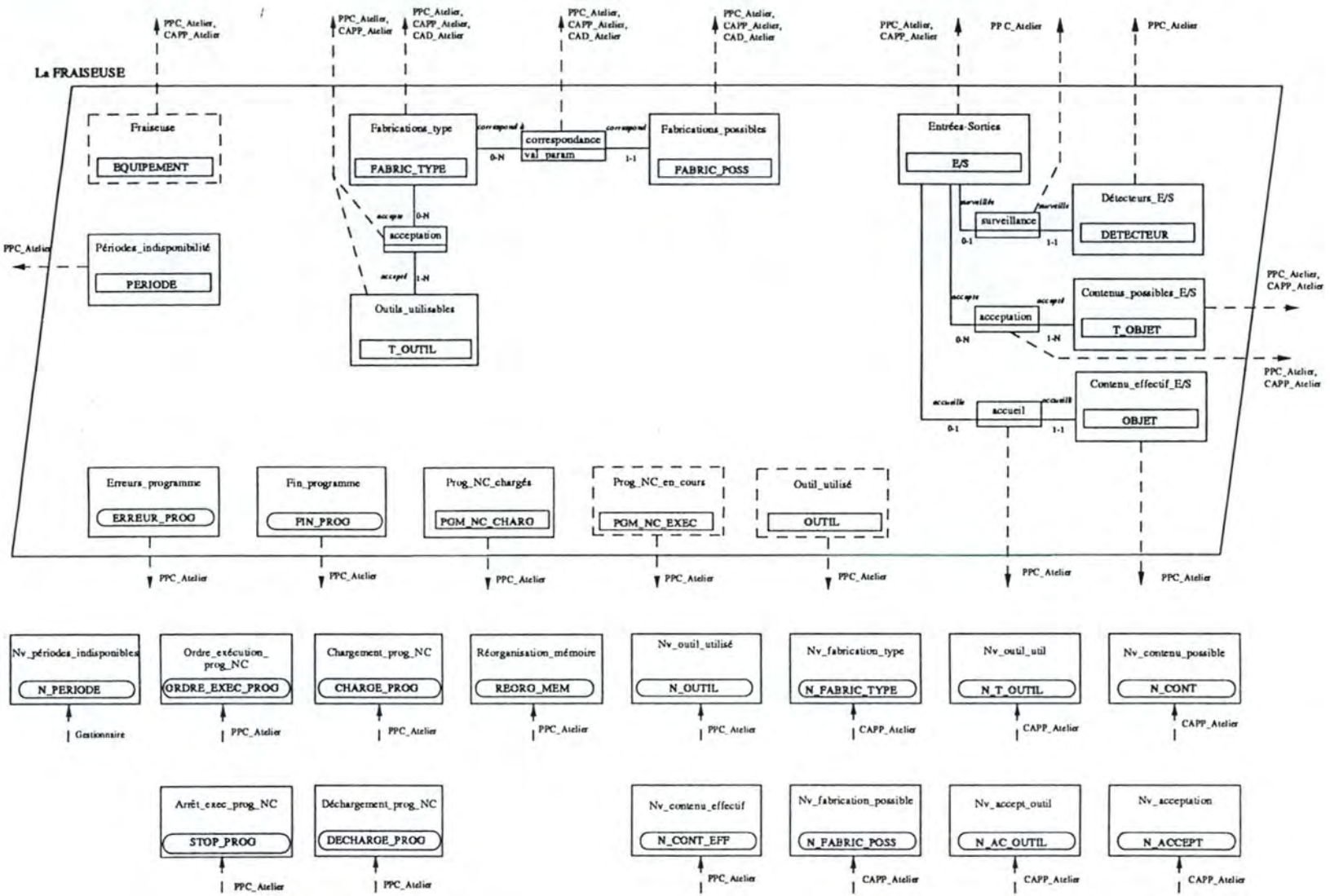
7.2.11. Erreurs_ordre_opération

- Description : Signale qu'une opération n'a pu être exécutée, elle en donne la cause et indique quel était le taux d'avancement lorsque l'erreur est survenue.
- Type : **ERREUR_ORDRE_OP**
- Destination : *PPC_ATELIER*
- Structure : ☞ Annexe C

8. La fraiseuse

8.1. Structure d'état de la fraiseuse

On trouvera à la figure suivante la structure d'état de l'agent Fraiseuse.


La FRAISEUSE
Fraiseuse
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Fabrications_type
FABRIC_TYPE
correspondance
val_param
Fabrications_possibles
FABRIC_POSS
acceptation
Outils_utilisables
T_OUTIL
Entrées-Sorties
E/S
surveillance
Détecteurs_E/S
DETECTEUR
acceptation
Contenus_possibles_E/S
T_OBJET
accueil
Contenu_effectif_E/S
OBJET
Erreurs_programme
ERREUR_PROG
Fin_programme
FIN_PROG
Prog_NC_chargés
PGM_NC_CHARG
Prog_NC_en_cours
PGM_NC_EXEC
Outil_utilisé
OUTIL
Nv_périodes_indisponibles
N_PERIODE
Ordre_exécution_prog_NC
ORDRE_EXEC_PROG
Chargement_prog_NC
CHARGE_PROG
Réorganisation_mémoire
REORG_MEM
Nv_outil_utilisé
N_OUTIL
Nv_fabrication_type
N_FABRIC_TYPE
Nv_outil_util
N_T_OUTIL
Nv_contenu_possible
N_CONT
Arrêt_exec_prog_NC
STOP_PROG
Déchargement_prog_NC
DECHARGE_PROG
Nv_contenu_effectif
N_CONT_EFF
Nv_fabrication_possible
N_FABRIC_POSS
Nv_accept_outil
N_AC_OUTIL
Nv_acceptation
N_ACCEPT

8.2. Description du schéma

8.2.1. Fraiseuse

- Description : Contient les informations décrivant la fraiseuse.
- Type : **EQUIPEMENT**
- Lien : ∅
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

8.2.2. Périodes_indisponibilité

- Description : Contient les informations sur les périodes d'indisponibilité de la fraiseuse ainsi que leur cause.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.3. Fabrications_type

- Description : Contient les informations concernant les fabrications types que la fraiseuse peut réaliser.
- Type : **FABRIC_TYPE**
- Liens :
  - **Correspondance** : Effectue le lien entre chaque Fabrications-type et les Fabrications-possibles qui lui correspondent (une fabrication possible étant une fabrication-type pour des valeurs de paramètres bien déterminées).
  - **Acceptation :** Détermine les outils que la machine accepte pour chaque opération type.
- Visible par : *PPC_ATELIER, CAPP_ATELIER, CAD_ATELIER*
- Structure : ☞ Annexe C

8.2.4. Fabrications_possibles

- Description : Contient les informations concernant les fabrications possibles (càd des opérations type pour lesquelles les valeurs de paramètres sont définies) que la fraiseuse est en mesure d'exécuter.
- Type : **FABRIC_POSS**
- Lien :
  - **Correspondance** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER, CAD_ATELIER*
- Structure : ☞ Annexe C

8.2.5. Outils_utilisables

- Description : Contient la description des types d'outils qu'accepte la fraiseuse.
- Type : **T_OUTIL**
- Lien : • **Acceptation** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

8.2.6. Entrées-Sorties

- Description : Contient les informations sur les entrées-sorties
- Type : **E/S**
- Lien : • **Surveillance** : Relie l'Entrée-Sortie au Détecteur chargé de la surveiller.
  • **Acceptation** : Indique les types d'objets que l'E/S peut accepter.
  • **Accueil** : Permet de désigner l'objet qui se trouve réellement sur une E/S
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

8.2.7. Détecteurs_E/S

- Description : Contient les informations décrivant les détecteurs chargés de surveiller les entrées-sorties. Celles-ci indiquent notamment si l'on détecte une présence.
- Type : **DETECTEUR**
- Lien : • **Surveillance** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.8. Contenus_possibles_E/S

- Description : Contient une description des types d'objets que les E/S acceptent.
- Type : **T_OBJET**
- Lien : • **Acceptation** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

8.2.9. Contenu_effectif_E/S

- Description : Contient les informations sur l'objet figurant sur une E/S.
- Type : **OBJET**
- Lien : • **Accueil** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.10. Prog_NC_chargés

- Description : Contient une série d'informations sur les programmes numériques chargés dans la mémoire de la fraiseuse (adresse de la première instruction du programme et identifiant du programme).
- Type : **PGM_NC_CHARG**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.11. Prog_NC_en_cours

- Description : Contient une série d'informations sur le programme numérique en cours d'exécution: identifiant et moment où le programme a été lancé.
- Type : **PGM_NC_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.12. Outil_utilisé

- Description : Contient les informations sur l'outil effectivement utilisé par la fraiseuse.
- Type : **OUTIL**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

8.2.13. Fin_programme

- Description : Signale la fin de l'exécution d'un programme NC
- Type : **FIN_PROG**
- Destination : *PPC_ATELIER*
- Structure : ☞ Annexe C

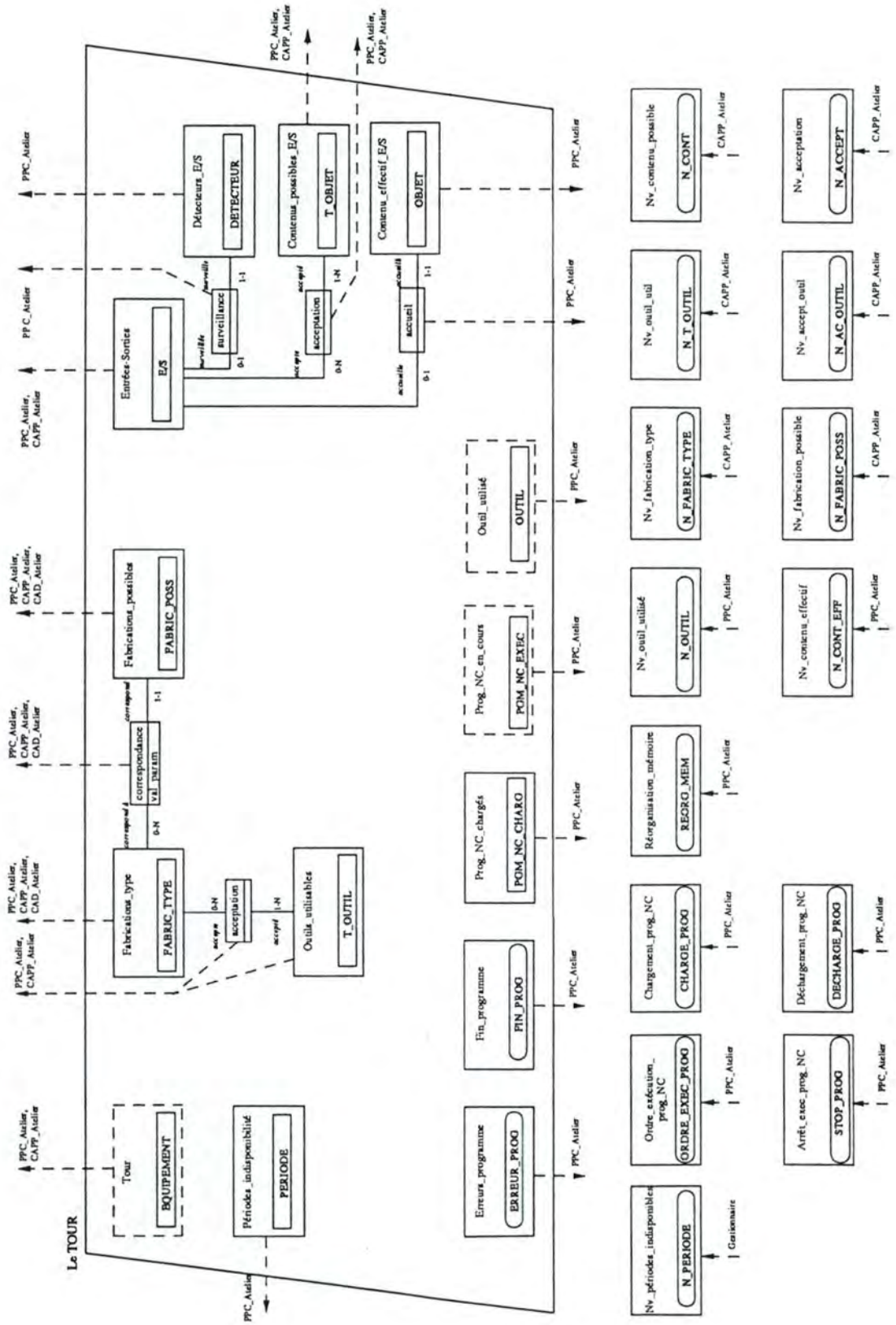
8.2.14. Erreurs_programme

- Description Signale une erreur lors de l'exécution d'un programme
- Type **ERREUR_PROG**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

9. Le tour

9.1. Structure d'état du tour

On trouvera à la figure suivante la structure d'état de l'agent Tour.


Le TOUR
Tour
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Fabrications_type
FABRIC_TYPE
correspondance
val_param
Fabrications_possibles
FABRIC_POSS
acceptation
Outils_utilisables
T_OUTIL
Entrées-Sorties
E/S
surveillance
Détecteurs_E/S
DETECTEUR
acceptation
Contenus_possibles_E/S
T_OBJET
accueil
Contenus_effectif_E/S
OBJET
Prog_NC_chargés
PGM_NC_CHARG
Prog_NC_en_cours
PGM_NC_EXEC
Outil_utilisé
OUTIL
Fin_programme
FIN_PROG
Erreurs_programme
ERREUR_PROG
Chargement_prog_NC
CHARGE_PROG
Déchargement_prog_NC
DECHARGE_PROG
Réorganisation_mémoire
REORG_MEM
Ordre_exécution_prog_NC
ORDRE_EXEC_PROG
Arrêt_exec_prog_NC
STOP_PROG
Nv_périodes_indisponibles
N_PERIODE
Nv_fabrication_type
N_FABRIC_TYPE
Nv_fabrication_possible
N_FABRIC_POSS
Nv_outil_utilisé
N_OUTIL
Nv_outil_util
N_T_OUTIL
Nv_accept_outil
N_AC_OUTIL
Nv_contenu_effectif
N_CONT_EFF
Nv_contenu_possible
N_CONT
Nv_acceptation
N_ACCEPT
PPC_Atelier
CAPP_Atelier
CAD_Atelier
Gestionnaire

9.2. Description du schéma

9.2.1. Tour

- Description : Contient les informations décrivant le tour.
- Type : **EQUIPEMENT**
- Lien : ∅
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

9.2.2. Périodes_indisponibilité

- Description : Contient les informations sur les périodes d'indisponibilité du tour ainsi que leur cause.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.3. Fabrications_type

- Description : Contient les informations concernant les fabrications types que le tour peut réaliser.
- Type : **FABRIC_TYPE**
- Liens :
  - **Correspondance** : Effectue le lien entre chaque Fabrications-type et les Fabrications-possibles qui lui correspondent (une fabrication possible étant une fabrication-type pour des valeurs de paramètres bien déterminées).
  - **Acceptation :** Détermine les outils que la machine accepte pour chaque opération type.
- Visible par : *PPC_ATELIER, CAPP_ATELIER, CAD_ATELIER*
- Structure : ☞ Annexe C

9.2.4. Fabrications_possibles

- Description : Contient les informations concernant les fabrications possibles (càd des opérations type pour lesquelles les valeurs de paramètres sont définies) que le tour est en mesure d'exécuter.
- Type : **FABRIC_POSS**
- Lien :
  - **Correspondance** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER, CAD_ATELIER*
- Structure : ☞ Annexe C

9.2.5. Outils_utilisables

- Description : Contient la description des types d'outils qu'accepte le tour.
- Type : **T_OUTIL**
- Lien : • **Acceptation** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

9.2.6. Entrées-Sorties

- Description : Contient les informations sur les entrées-sorties
- Type : **E/S**
- Lien : • **Surveillance** : Relie l'Entrée-Sortie au Détecteur chargé de la surveiller.
  • **Acceptation** : Indique les types d'objets que l'E/S peut accepter.
  • **Accueil** : Permet de désigner l'objet qui se trouve réellement sur une E/S
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

9.2.7. Détecteurs_E/S

- Description : Contient les informations décrivant les détecteurs chargés de surveiller les entrées-sorties. Celles-ci indiquent notamment si l'on détecte une présence.
- Type : **DETECTEUR**
- Lien : • **Surveillance** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.8. Contenus_possibles_E/S

- Description : Contient une description des types d'objets que les E/S acceptent.
- Type : **T_OBJET**
- Lien : • **Acceptation** : Voir ci-dessus.
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

9.2.9. Contenu_effectif_E/S

- Description : Contient les informations sur l'objet figurant sur une E/S.
- Type : **OBJET**
- Lien : • **Accueil** : Voir ci-dessus.
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.10. Prog_NC_chargés

- Description : Contient une série d'informations sur les programmes numériques chargés dans la mémoire du tour (adresse de la première instruction du programme et identifiant du programme).
- Type : **PGM_NC_CHARG**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.11. Prog_NC_en_cours

- Description : Contient une série d'informations sur le programme numérique en cours d'exécution: identifiant et moment où le programme a été lancé.
- Type : **PGM_NC_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.12. Outil_utilisé

- Description : Contient les informations sur l'outil effectivement utilisé par le tour.
- Type : **OUTIL**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

9.2.13. Fin_programme

- Description : Signale la fin de l'exécution d'un programme NC
- Type : **FIN_PROG**
- Destination : *PPC_ATELIER*
- Structure : ☞ Annexe C

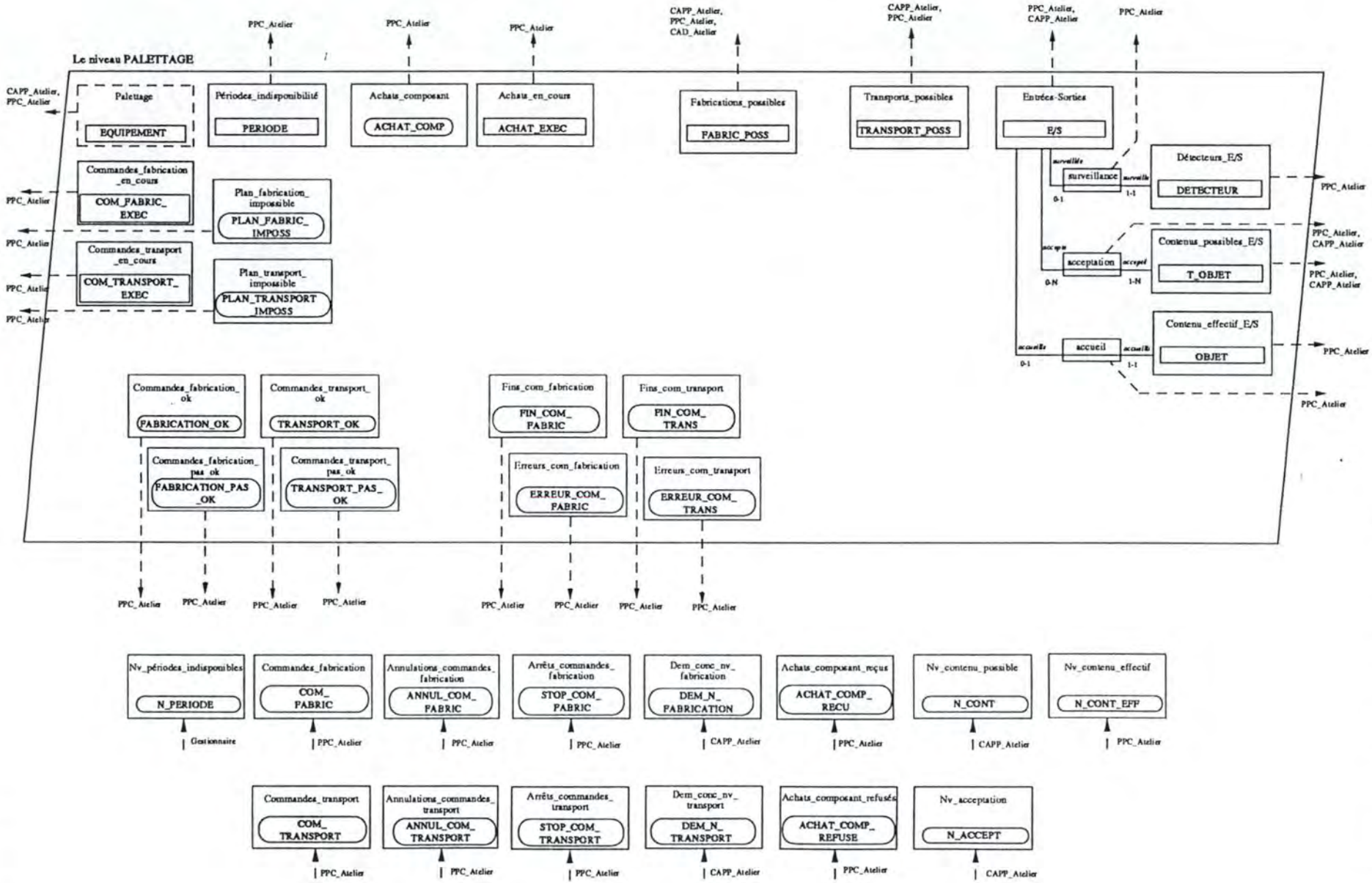
9.2.14. Erreurs_programme

- Description Signale une erreur lors de l'exécution d'un programme
- Type **ERREUR_PROG**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10. Le niveau Palettage

10.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent Niveau_Palettage.


Le niveau PALETTAGE
Palettage
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Achats_composant
ACHAT_COMP
Achats_en_cours
ACHAT_EXEC
Fabrications_possibles
FABRIC_POSS
Transports_possibles
TRANSPORT_POSS
Entrées-Sorties
E/S
surveillance
Détecteurs_E/S
DETECTEUR
acceptation
Contenus_possibles_E/S
T_OBJET
accueil
Contenu_effectif_E/S
OBJET
Commandes_fabrication_en_cours
COM_FABRIC_EXEC
Plan_fabrication_impossible
PLAN_FABRIC_IMPOSS
Commandes_transport_en_cours
COM_TRANSPORT_EXEC
Plan_transport_impossible
PLAN_TRANSPORT_IMPOSS
Commandes_fabrication_ok
FABRICATION_OK
Commandes_transport_ok
TRANSPORT_OK
Commandes_fabrication_pas_ok
FABRICATION_PAS_OK
Commandes_transport_pas_ok
TRANSPORT_PAS_OK
Fins_com_fabrication
FIN_COM_FABRIC
Fins_com_transport
FIN_COM_TRANS
Erreurs_com_fabrication
ERREUR_COM_FABRIC
Erreurs_com_transport
ERREUR_COM_TRANS
Nv_périodes_indisponibles
N_PERIODE
Commandes_fabrication
COM_FABRIC
Annulations_commandes_fabrication
ANNUL_COM_FABRIC
Arrêts_commandes_fabrication
STOP_COM_FABRIC
Dem_conc_nv_fabrication
DEM_N_FABRICATION
Achats_composant_reçus
ACHAT_COMP_RECU
Nv_contenu_possible
N_CONT
Nv_contenu_effectif
N_CONT_EFF
Commandes_transport
COM_TRANSPORT
Annulations_commandes_transport
ANNUL_COM_TRANSPORT
Arrêts_commandes_transport
STOP_COM_TRANSPORT
Dem_conc_nv_transport
DEM_N_TRANSPORT
Achats_composant_refusés
ACHAT_COMP_REFUSE
Nv_acceptation
N_ACCEPT
PPC_Atelier
CAPP_Atelier, PPC_Atelier
CAPP_Atelier, PPC_Atelier, CAD_Atelier
PPC_Atelier, CAPP_Atelier
Gestionnaire

10.2. Description du schéma

10.2.1. Palettage

- Description: Contient les informations générales concernant le Palettage: localisation, description informelle, statut (marche, panne ou indisponible),...
- Type: **EQUIPEMENT**
- Lien: ∅
- Visible par: *CAPP_ATELIER ,PPC_ATELIER*
- Structure: ☞ Annexe C

10.2.2. Périodes_indisponibilité

- Description: Contient les informations concernant les périodes durant lesquelles le niveau Palettage est indisponible.
- Type: **PERIODE**
- Lien: ∅
- Visible par: *PPC_ATELIER*
- Structure: ☞ Annexe C

10.2.3. Achats_en_cours

- Description: Contient une description des achats de composants en cours
- Type: **ACHAT_EXEC**
- Lien: ∅
- Visible par: *PPC_ATELIER*
- Structure: ☞ Annexe C

10.2.4. Fabrications_possibles

- Description: Reprend les informations (type d'objet en input, en output, durée théorique d'exécution,...) au sujet des fabrications que le niveau Palettage est en mesure de réaliser.
- Type: **FABRIC_POSS**
- Lien: ∅
- Visible par: *CAPP_ATELIER, PPC_ATELIER, CAD_ATELIER*
- Structure: ☞ Annexe C

10.2.5. Transports_possibles

- Description : Décrit les transports que le niveau palettage peut réaliser.
- Type : **TRANSPORT_POSS**
- Lien : ∅
- Visible par : *CAPP_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.6. Entrées-Sorties

- Description : Reprend une série d'informations sur les entrées-sorties du niveau palettage.
- Type : **E/S**
- Liens :
  - **Surveillance** : Relie les entrées-sorties aux détecteurs chargés de les surveiller.
  - **Acceptation** : Relie les entrées-sorties aux types d'objets qu'elles peuvent accepter.
  - **Accueil** : Donne le contenu effectif d'une entrée-sortie.
- Visible par : *CAPP_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.7. Détecteurs_E/S

- Description : Descriptif des détecteurs du niveau palettage indiquant entre autres si une présence est détectée.
- Type : **DETECTEUR**
- Lien : • Surveillance : Voir ci-dessus
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.8. Contenus_possibles_E/S

- Description : Reprend les informations décrivant les types d'objets acceptés par les entrées-sorties du niveau.
- Type : **T_OBJET**
- Lien : • **Acceptation** : Voir ci-dessus
- Visible par : *PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

10.2.9. Contenu_effectif_E/S

- Description : Donne un descriptif des objets réellement situés aux entrées-sorties du niveau palettage.
- Type : **OBJET**
- Lien : • **Accueil** : Voir ci-dessus
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.10. Commandes_fabrication_en_cours

- Description : Contient les informations concernant les commandes de fabrication en cours d'exécution dans le niveau palettage: type d'objet à fabriquer, quantité à fabriquer, échéance, moment où la réalisation de la commande a commencé, taux d'avancement,...
- Type : **COM_FABRIC_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.11. Commandes_transport_en_cours

- Description : Contient les informations concernant les commandes de transport en cours d'exécution: entrée-sortie origine, entrée-sortie destination, objet transporté, moment où l'on a lancé le transport, échéance,...
- Type : **COM_TRANSPORT_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

10.2.12. Achats_composant

- Description : Demande l'achat de composants au PPC de l'atelier.
- Type : **ACHAT_COMP**
- Destination : *PPC_ATELIER*
- Structure : ☞Annexe C

10.2.13. Plan_fabrication_impossible

- Description : Indique au PPC_ATELIER l'impossibilité de respecter le plan de fabrication initialement prévu.
- Type : **PLAN_FABRIC_IMPOSS**
- Destination : *PPC_ATELIER*
- Structure : ☞Annexe C

10.2.14. Plan_transport_impossible

- Description Signale qu'il est impossible de suivre le plan de transport
- Type **PLAN_TRANSPORT_IMPOSS**
- Destination *PPC_ATELIER*
- Structure ☞Annexe C

10.2.15. Commandes_fabrication_pas_ok

- Description Permet de signaler que la commande de fabrication indiquée est refusée.
- Type **FABRICATION_PAS_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.16. Commandes_fabrication_ok

- Description Signale que le niveau palettage accepte de réaliser la commande précisée.
- Type **FABRICATION_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.17. Commandes_transport_pas_ok

- Description Indique au PPC de l'atelier qu'on refuse de réaliser la commande de transport désignée.
- Type **TRANSPORT_PAS_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.18. Commandes_transport_ok

- Description Permet de désigner les commandes de transports acceptées par le niveau.
- Type **TRANSPORT_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.19. Fins_com_fabrication

- Description Permet de signaler que la commande de fabrication indiquée est terminée.
- Type **FIN_COM_FABRIC**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.20. Fins_com_transport

- Description Indique la fin d'exécution d'une certaine commande de transport.
- Type **FIN_COM_TRANS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

10.2.21. Erreurs_com_fabrication

- Description Fait savoir au PPC_ATELIER qu'une erreur est apparue et a empêché l'exécution de la commande indiquée.
- Type **ERREUR_COM_FABRIC**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

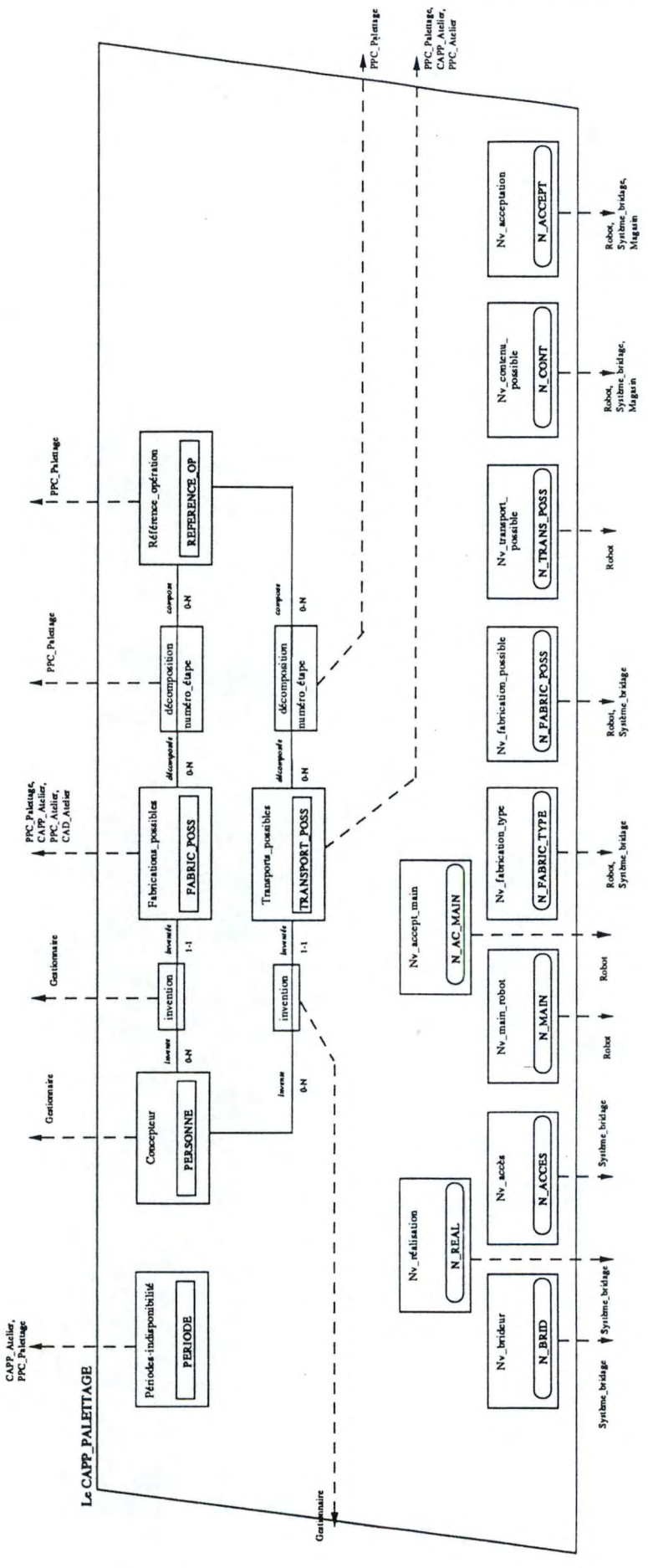
10.2.22. Erreurs_com_transport

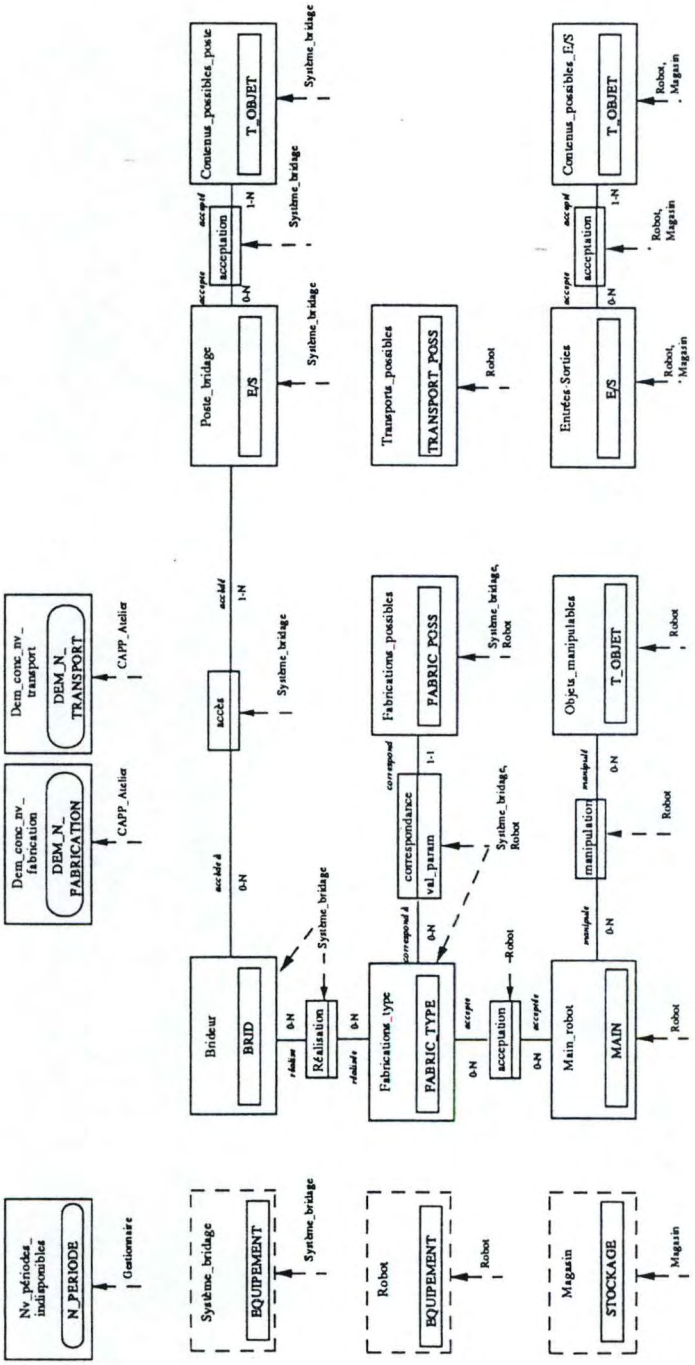
- Description Signale qu'une erreur a eu lieu lors de la réalisation de la commande spécifiée.
- Type **ERREUR_COM_TRANS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

11. Le CAPP_Palettage

11.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent CAPP_Palettage.


Le CAPP_PALETTAGE
CAPP_Atelier, PPC_Palettage
Gestionnaire
PPC_Palettage
PPC_Palettage, CAPP_Atelier, PPC_Atelier
Périodes indisponibilité
PERIODE
Concepteur
PERSONNE
invention
Fabrications_possibles
FABRIC_POSS
décomposition
numéro_étape
Référence_opération
REFERENCE_OP
Transports_possibles
TRANSPORT_POSS
Nv_réalisation
N_REAL
Nv_brideur
N_BRID
Nv_accès
N_ACCES
Nv_main_robot
N_MAIN
Nv_accepl_main
N_AC_MAIN
Nv_fabrication_type
N_FABRIC_TYPE
Nv_fabrication_possible
N_FABRIC_POSS
Nv_transport_possible
N_TRANS_POSS
Nv_contenu_possible
N_CONT
Nv_acceptation
N_ACCEPT
Système_bridage
Robot
Robot, Système_bridage, Magasin


Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
Dem_conc_nv_fabrication
DEM_N_FABRICATION
CAPP_Atelier
Dem_conc_nv_transport
DEM_N_TRANSPORT
CAPP_Atelier
Système_bridge
EQUIPEMENT
Système_bridge
Robot
EQUIPEMENT
Robot
Magasin
STOCKAGE
Magasin
Brideur
BRID
accède
0-N
accès
accédé
1-N
Système_bridge
Réalisation
réalise
réalisé
Système_bridge
Fabrications_type
FABRIC_TYPE
accepte
0-N
acceptation
accepté
Robot
Main_robot
MAIN
Robot
manipule
manipulé
0-N
Fabrications_possibles
FABRIC_POSS
Système_bridge, Robot
correspond
correspondance
val_param
1-1
Système_bridge, Robot
Objets_manipulables
T_OBJET
Robot
manipulation
0-N
Robot
Poste_bridge
E/S
Système_bridge
accepte
acceptation
accepté
0-N
Système_bridge
Contenus_possibles_poste
T_OBJET
1-N
Système_bridge
Transports_possibles
TRANSPORT_POSS
Robot
Entrées Sorties
E/S
Robot, Magasin
accepte
acceptation
accepté
0-N
Robot, Magasin
Contenus_possibles_E/S
T_OBJET
1-N
Robot, Magasin

11.2. Description de l'agent CAPP_Palettage

11.2.1. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes durant lesquelles le CAPP_Palettage sera indisponible.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_PALETTAGE,CAPP_ATELIER*
- Structure : ☞ Annexe C

11.2.2. Concepteur

- Description : Contient les informations concernant le concepteur des recettes du niveau Palettage
- Type : **PERSONNE**
- Liens :
  - **Invention** : Relie le Concepteur aux fabrications ou aux transports qu'il a inventés.
- Visible par : *GESTIONNAIRE_PALETTAGE*
- Structure : ☞ Annexe C

11.2.3. Fabrications_possibles

- Description : Contient les informations concernant les fabrications que le niveau Palettage peut exécuter.
- Type : **FABRIC_POSS**
- Liens :
  - **Décomposition** : Décompose la fabrication possible en références à des opérations, exécutables par le robot ou le système de bridage, dont le séquencement réalisera la fabrication concernée.
  - **Invention** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, PPC_ATELIER, CAPP_ATELIER, CAD_ATELIER*
- Structure : ☞ Annexe C

11.2.4. Transports_possibles

- Description : Contient les informations concernant les transports qu'il est possible de réaliser au sein du niveau Palettage
- Type : **TRANSPORT_POSS**
- Liens :
  - **Décomposition** : Permet de désigner les opérations à exécuter les unes après les autres afin de réaliser le transport en question.
  - **Invention** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, CAPP_ATELIER, PPC_ATELIER*
- Structure : ☞ Annexe C

11.2.5. Référence_opération

- Description : Références aux fabrications possibles réalisables par le système de bridage, ainsi qu'aux fabrications possibles et transports possibles réalisables par le robot.
- Type : **REFERENCE_OP**
- Lien : • **Décomposition** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

11.2.6. Nv_brideur

- Description : Communique, au système de bridage, la description d'un nouveau brideur.
- Type : **N_BRID**
- Destination : *SYSTEME_BRIDAGE*
- Structure : ☞ Annexe C

11.2.7. Nv_réalisation

- Description : Informe le système de bridage que le brideur désigné peut réaliser la fabrication type indiquée.
- Type : **N_REAL**
- Destination : *SYSTEME_BRIDAGE*
- Structure : ☞ Annexe C

11.2.8. Nv_accès

- Description : Indique que tel brideur peut accéder à tel poste de bridage.
- Type : **N_ACCES**
- Destination : *SYSTEME_BRIDAGE*
- Structure : ☞ Annexe C

11.2.9. Nv_main_robot

- Description : Signale au robot qu'il a une nouvelle main à sa disposition.
- Type : **N_MAIN**
- Destination : *ROBOT*
- Structure : ☞ Annexe C

11.2.10. Nv_accept_main

- Description : Fait savoir au robot qu'une certaine fabrication peut utiliser une certaine main.
- Type : **N_AC_MAIN**

- Destination *ROBOT*
- Structure ☞ Annexe C

11.2.11. Nv_fabrication_type

- Description Permet de signaler au Robot et au Système de bridage qu'ils doivent être en mesure de réaliser une nouvelle fabrication type dont on transmet les caractéristiques.
- Type **N_FABRIC_TYPE**
- Destination *ROBOT, SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

11.2.12. Nv_fabrication_possible

- Description Permet de signaler au robot ou au système de bridage qu'ilest capable d'exécuter une nouvelle fabrication-possible.
- Type **N_FABRIC_POSS**
- Destination *ROBOT, SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

11.2.13. Nv_transport_possible

- Description Permet de signaler au robot qu'il est désormais en mesure d'effectuer un nouveau transport possible.
- Type **N_TRANS_POSS**
- Destination *ROBOT*
- Structure ☞ Annexe C

11.2.14. Nv_contenu_possible

- Description Permet de signaler au robot ou au système de bridage qu'ils peuvent accepter un nouveau type d'objet à leurs entrées-sorties
- Type **N_CONT**
- Destination *ROBOT, SYSTEME_BRIDAGE, MAGASIN*
- Structure ☞ Annexe C

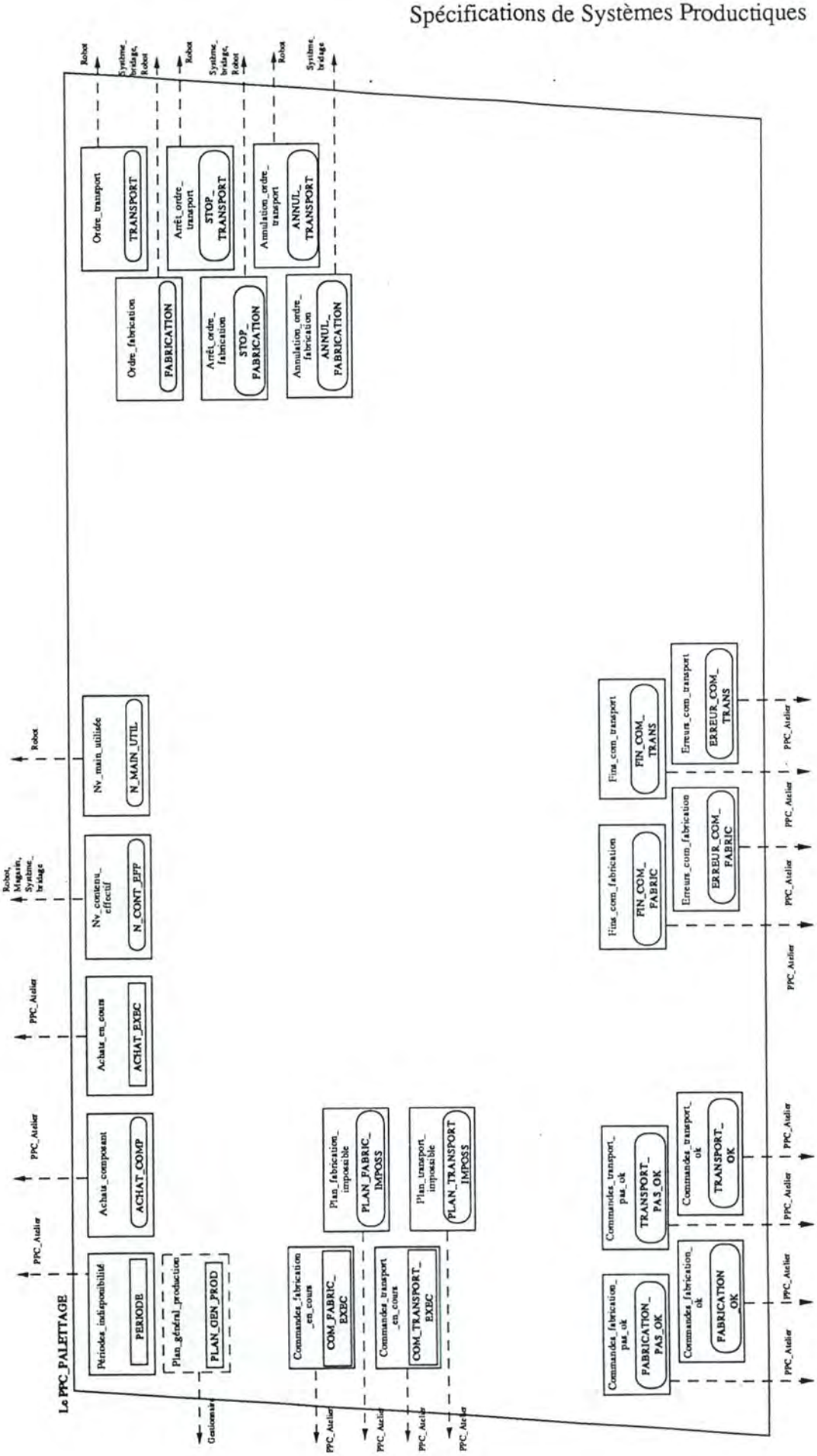
11.2.15. Nv_acceptation

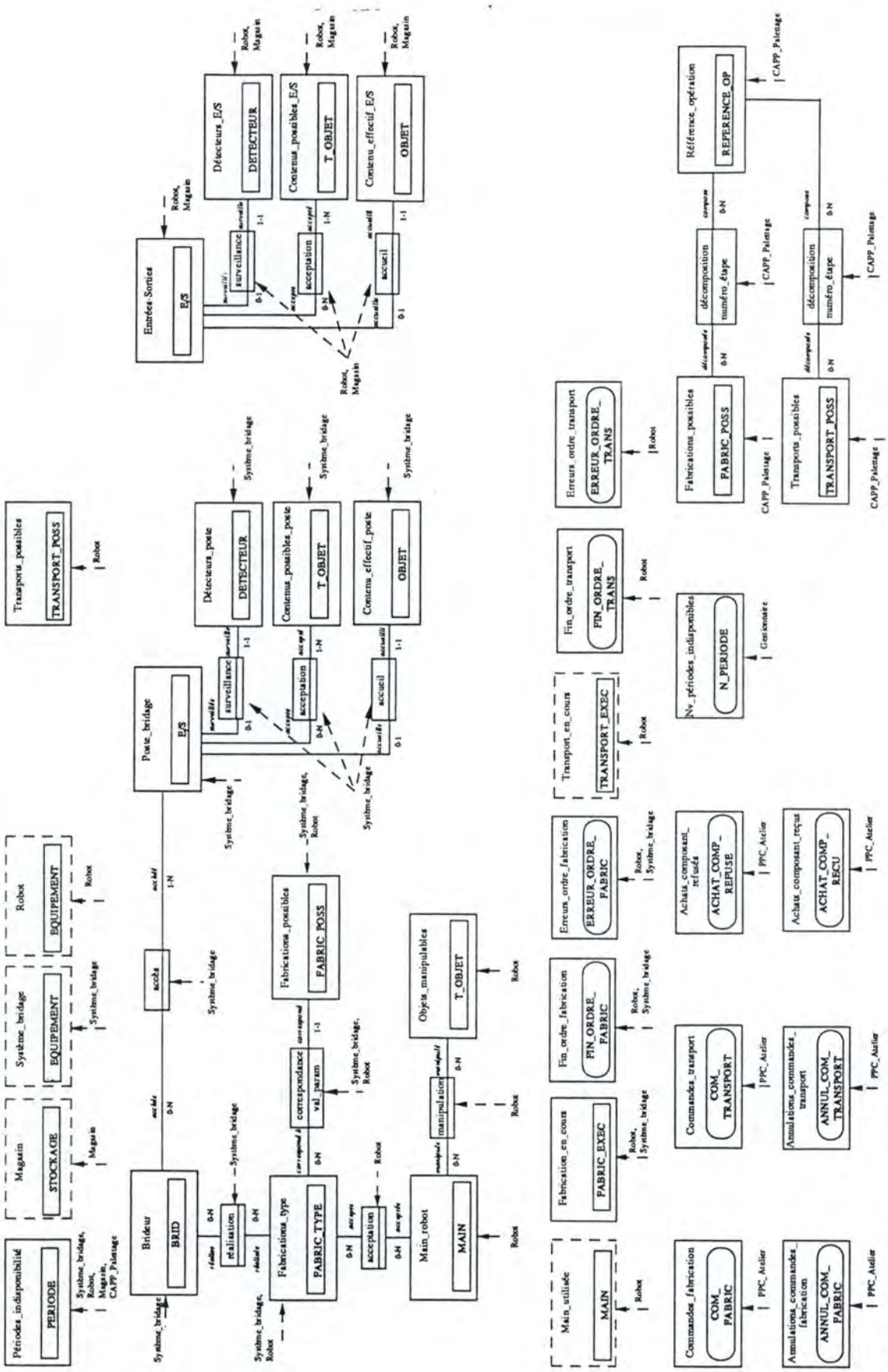
- Description Indique que telle entrée-sortie peut accueillir tel objet.
- Type **N_ACCEPT**
- Destination *ROBOT, SYSTEME_BRIDAGE, MAGASIN*
- Structure ☞ Annexe C

12. Le PPC_Palettage

12.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent PPC_Palettage.


Le PPC_PALETTAGE
Période_indisponibilité
PERIODE
Plan_général_production
PLAN_GEN_PROD
Achats_composant
ACHAT_COMP
Achats_en_cours
ACHAT_EXEC
Nv_contenu_effectif
N_CONT_EFP
Nv_main_utilisée
N_MAIN_UTIL
Commandes_fabrication_en_cours
COM_FABRIC_EXEC
Commandes_transport_en_cours
COM_TRANSPORT_EXEC
Plan_fabrication_impossible
PLAN_FABRIC_IMPOSS
Plan_transport_impossible
PLAN_TRANSPORT_IMPOSS
Commandes_fabrication_pas_ok
FABRICATION_PAS_OK
Commandes_fabrication_ok
FABRICATION_OK
Commandes_transport_pas_ok
TRANSPORT_PAS_OK
Commandes_transport_ok
TRANSPORT_OK
Fins_com_fabrication
FIN_COM_FABRIC
Erreurs_com_fabrication
ERREUR_COM_FABRIC
Fins_com_transport
FIN_COM_TRANS
Erreurs_com_transport
ERREUR_COM_TRANS
Ordre_fabrication
FABRICATION
Arrêt_ordre_fabrication
STOP_FABRICATION
Annulation_ordre_fabrication
ANNUL_FABRICATION
Ordre_transport
TRANSPORT
Arrêt_ordre_transport
STOP_TRANSPORT
Annulation_ordre_transport
ANNUL_TRANSPORT
Gestionnaire
Robot
Robot, Magasin, Système_bridage
PPC_Atelier
Système_bridage, Robot


Périodes_indisponibilité
PERIODE
Magasin
STOCKAGE
Système_bridge
EQUIPEMENT
EQUIPEMENT
Robot
Transports_possibles
TRANSPORT_POSS
Détecteurs_E/S
DETECTEUR
Contenus_possibles_E/S
T_OBJET
Contenu_effectif_E/S
OBJET
Entrées-Sorties
E/S
surveillance
acceptation
accueil
Détecteurs_poste
Contenus_possibles_poste
Contenu_effectif_poste
Poste_bridge
Bridage
BRID
accès
réalisation
Fabrications_type
FABRIC_TYPE
correspondance
val_param
Fabrications_possibles
FABRIC_POSS
Objets_manipulables
T_OBJET
Main_robot
MAIN
manipulation
Main_utilisée
Fabrication_en_cours
FABRIC_EXEC
Fin_ordre_fabrication
FIN_ORDRE_FABRIC
Erreurs_ordre_fabrication
ERREUR_ORDRE_FABRIC
Achats_composant_refusés
ACHAT_COMP_REFUSE
Achats_composant_reçus
ACHAT_COMP_RECU
Commandes_transport
COM_TRANSPORT
Annulations_commandes_transport
ANNUL_COM_TRANSPORT
Arrêts_commandes_transport
STOP_COM_TRANSPORT
Transport_en_cours
TRANSPORT_EXEC
Fin_ordre_transport
FIN_ORDRE_TRANS
Erreurs_ordre_transport
ERREUR_ORDRE_TRANS
Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
Fabrications_possibles
FABRIC_POSS
Transports_possibles
TRANSPORT_POSS
décomposition
numéro_étape
Référence_opération
REFERENCE_OP
Commandes_fabrication
COM_FABRIC
Annulations_commandes_fabrication
ANNUL_COM_FABRIC
Arrêts_commandes_fabrication
STOP_COM_FABRIC
PPC_Atelier
CAPP_Palletage
Robot
Magasin
Système_bridge
Gestionnaire

12.2. Description du schéma

12.2.1. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes durant lesquelles le PPC_Palettage est indisponible.
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

12.2.2. Achats_en_cours

- Description : Contient les informations concernant les achats de composants en cours
- Type : **ACHAT_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

12.2.3. Plan_général_production

- Description : Contient le séquencemenent des instructions afin de répondre aux commandes acceptées.
- Type : **PLAN_GEN_PROD**
- Lien : ∅
- Visible par : *GESTIONNAIRE_PALETTAGE*
- Structure : ☞ Annexe C

12.2.4. Commandes_fabrication_en_cours

- Description : Contient les informations au sujet des commandes de fabrication en cours d'exécution, dans le niveau palettage.
- Type : **COM_FABRIC_EXEC**
- Lien : ∅
- Visible par : *PPC_ATELIER*
- Structure : ☞ Annexe C

12.2.5. Commandes_transport_en_cours

- Description Contient les informations concernant les commandes de transport en cours d'exécution, dans le niveau palettage.
- Type **COM_TRANSPORT_EXEC**
- Lien ∅
- Visible par *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.6. Achats_composant

- Description Demande l'achat du composant indiqué au PPC de l'atelier.
- Type **ACHAT_COMP**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.7. Nv_contenu_effectif

- Description Permet de signaler que le nouveau contenu d'une entrée-sortie est l'objet indiqué
- Type **N_CONT_EFF**
- Destination *ROBOT, MAGASIN, SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

12.2.8. Nv_main_utilisée

- Description Permet de signaler au Robot qu' il utilise la main indiquée.
- Type **N_MAIN_UTIL**
- Destination *ROBOT*
- Structure ☞ Annexe C

12.2.9. Ordre_transport

- Description Permet de demander au Robot d'effectuer un transport.
- Type **TRANSPORT**
- Destination *ROBOT*
- Structure ☞ Annexe C

12.2.10. Ordre_fabrication

- Description Demande au Robot ou au Système de bridage d'exécuter l'ordre de fabrication indiqué.
- Type **FABRICATION**
- Destination *ROBOT, SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

12.2.11. Arrêt_ordre_transport

- Description Permet de signaler au Robot qu'il doit stopper le transport en cours.
- Type **STOP_TRANSPORT**
- Destination *ROBOT*
- Structure ☞ Annexe C

12.2.12. Arrêt_ordre_fabrication

- Description Permet d'interrompre la fabrication en cours
- Type **STOP_FABRICATION**
- Destination *ROBOT, SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

12.2.13. Annulation_ordre_transport

- Description Permet de signaler au Robot qu'il doit annuler l'ordre de transport qu'il est en train d'exécuter. Il doit donc revenir à la position à laquelle il se trouvait le robot avant que ne débute le transport.
- Type **ANNUL_TRANSPORT**
- Destination *ROBOT*
- Structure ☞ Annexe C

12.2.14. Annulation_ordre_fabrication

- Description Signale au Système de bridage qu'il doit annuler l'ordre de fabrication qu'il est en train d'exécuter. Il fautdonc revenir à l'état précédant le moment où le système de bridage a commencé la fabrication.
- Type **ANNUL_FABRICATION**
- Destination *SYSTEME_BRIDAGE*
- Structure ☞ Annexe C

12.2.15. Plan_fabrication_impossible

- Description Signale au niveau supérieur que le plan de fabrication initialement prévu est impossible a réaliser.
- Type **PLAN_FABRIC_IMPOSS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.16. Plan_transport_impossible

- Description Permet de signaler que le niveau palettage se trouve dans l'incapacité de réaliser le plan de transport initialement prévu.
- Type **PLAN_TRANSPORT_IMPOSS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.17. Commandes_fabrication_pas_ok

- Description Indique que la commande de fabrication indiquée n'a pas pu être réalisée
- Type **FABRICATION_PAS_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.18. Commandes_fabrication_ok

- Description Signale au PPC de l'atelier que le niveau palettage accepte la commande précisée.
- Type **FABRICATION_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.19. Commandes_transport_pas_ok

- Description Signale qu'on n'accepte pas la commande spécifiée.
- Type **TRANSPORT_PAS_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.20. Commandes_transport_ok

- Description Indique que la commande de transport est acceptée.
- Type **TRANSPORT_OK**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.21. Fins_com_fabrication

- Description Permet de signaler que la commande de fabrication indiquée est terminée.
- Type **FIN_COM_FABRIC**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.22. Fins_com_transport

- Description Informe de la fin de la réalisation d'une commande.
- Type **FIN_COM_TRANS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

12.2.23. Erreurs_com_fabrication

- Description Communique les erreurs qui ont empêché la réalisation de la commande précisée.
- Type **ERREUR_COM_FABRIC**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

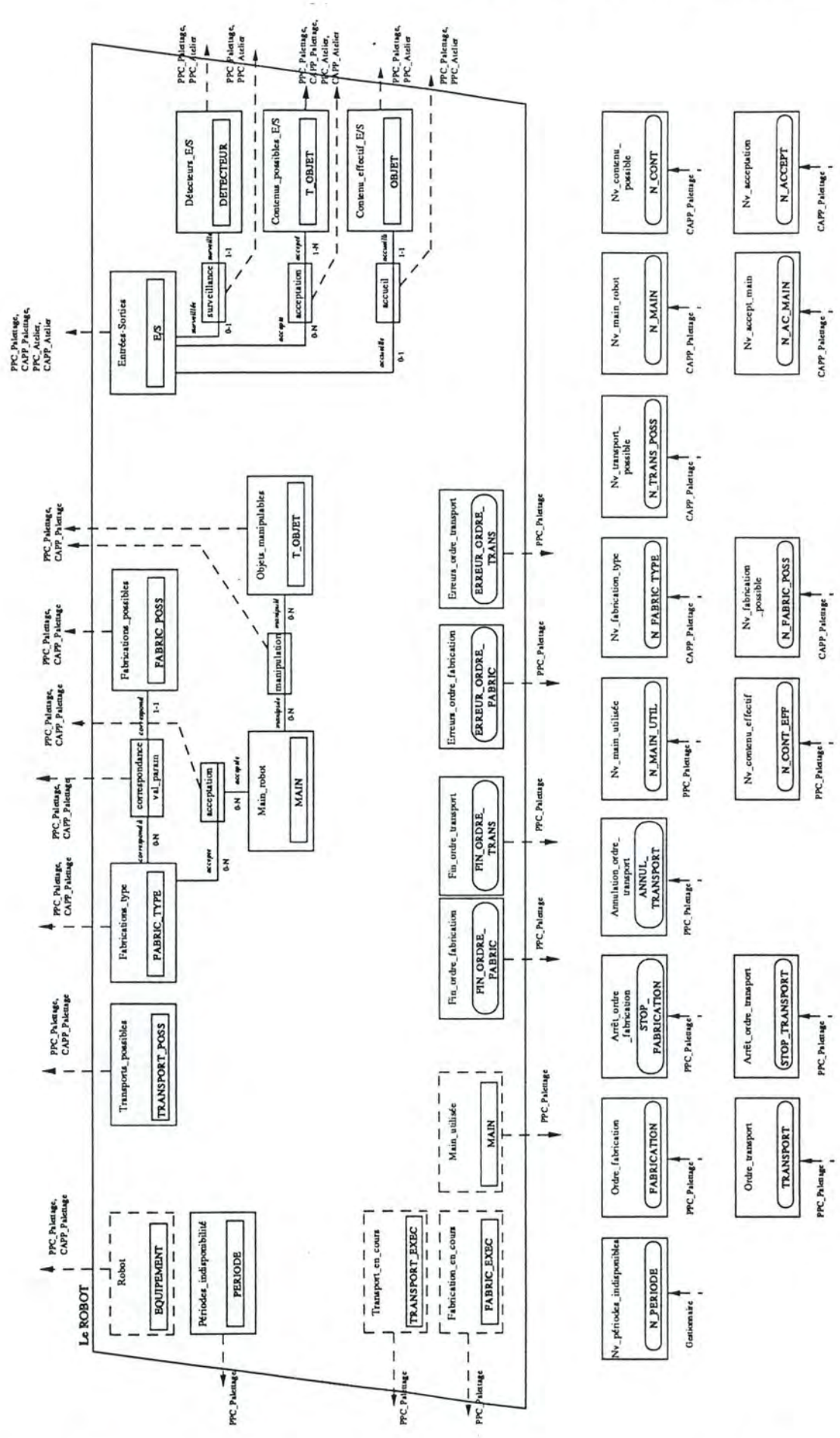
12.2.24. Erreurs_com_transport

- Description Signale qu'une erreur a arrêté l'exécution d'une commande.
- Type **ERREUR_COM_TRANS**
- Destination *PPC_ATELIER*
- Structure ☞ Annexe C

13. Le robot

13.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent Robot.


Le ROBOT
Robot
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Transport_en_cours
TRANSPORT_EXEC
Fabrication_en_cours
FABRIC_EXEC
Transports_possibles
TRANSPORT_POSS
Fabrications_type
FABRIC_TYPE
Main_utilisée
MAIN
Ordre_fabrication
FABRICATION
Ordre_transport
TRANSPORT
Arrêt_ordre_fabrication
STOP_FABRICATION
Arrêt_ordre_transport
STOP_TRANSPORT
Fin_ordre_fabrication
FIN_ORDRE_FABRIC
Fin_ordre_transport
FIN_ORDRE_TRANS
Annulation_ordre_transport
ANNUL_TRANSPORT
Main_robot
MAIN
correspondance
val_param
acceptation
Fabrications_possibles
FABRIC_POSS
Objets_manipulables
T_OBJET
manipulation
Entrées-Sorties
E/S
surveillance
acceptation
accueil
Détecteurs_E/S
DETECTEUR
Contenus_possibles_E/S
T_OBJET
Contenu_effectif_E/S
OBJET
Erreurs_ordre_fabrication
ERREUR_ORDRE_FABRIC
Erreurs_ordre_transport
ERREUR_ORDRE_TRANS
Nv_main_utilisée
N_MAIN_UTIL
Nv_fabrication_type
N_FABRIC_TYPE
Nv_transport_possible
N_TRANS_POSS
Nv_main_robot
N_MAIN
Nv_contenu_possible
N_CONT
Nv_contenu_effectif
N_CONT_EFF
Nv_fabrication_possible
N_FABRIC_POSS
Nv_accept_main
N_AC_MAIN
Nv_acceptation
N_ACCEPT
Nv_périodes_indisponibles
N_PERIODE
Gestionnaire
PPC_Palettage
CAPP_Palettage
PPC_Atelier
CAPP_Atelier

13.2. Description du schéma

13.2.1. Robot

- Description : Contient les informations concernant le robot
- Type : **EQUIPEMENT**
- Lien : ∅
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.2. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes d'indisponibilité de l'équipement Robot
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

13.2.3. Transports_possibles

- Description : Contient les informations concernant les transports que le Robot est en mesure de réaliser.
- Type : **TRANSPORT_POSS**
- Lien : ∅
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.4. Fabrications_type

- Description : Contient les informations concernant les fabrications types que le Robot peut reconnaître.
- Type : **FABRIC_TYPE**
- Liens :
  - **Correspondance** : Relie les Fabrications_type aux Fabrications_possibles correspondantes.
  - **Acceptation** : Relie les Fabrications_type aux mains connues du robot afin d'indiquer les mains qu'il est possible d'utiliser pour une fabrication type précise.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.5. Fabrications_possibles

- Description : Contient les informations concernant les fabrications possibles directement réalisables par le Robot. Rappel: une fabrication possible est une fabrication type pour laquelle on a définit les valeurs que prennent les paramètres.
- Type : **FABRIC_POSS**
- Lien : • **Correspondance** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.6. Main_robot

- Description : Contient les informations concernant les mains que le robot accepte.
- Type : **MAIN**
- Liens : • **Acceptation** : Voir ci-dessus.
  • **Manipulation** : Relie la Main_robot aux Objets qu'elle est en mesure de manipuler
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.7. Objets_manipulables

- Description : Contient les informations concernant les types d'objet que les mains du robot peuvent manipuler.
- Type : **T_OBJET**
- Lien : • **Manipulation** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

13.2.8. Entrées-Sorties

- Description : Contient les informations concernant les entrées-sorties du Robot
- Type : **E/S**
- Liens : • **Surveillance** : Relie l'Entrée-Sortie au détecteur chargé de la surveiller.
  • **Acceptation** : Relie l'Entrée-Sortie aux objets qu'elle peut accepter.
  • **Accueil** : Relie l'Entrée-Sortie à son contenu effectif.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE, PPC_ATELIER, CAPP_ATELIER*
- Structure : ☞ Annexe C

13.2.9. Détecteurs_E/S

- ■ Description : Contient les informations concernant les détecteurs chargés de surveiller les entrées-sorties du robot
- ■ Type : **DETECTEUR**
- ■ Lien : • **Surveillance** : Voir ci-dessus.
- ■ Visible par : *PPC_PALETTAGE, PPC_ATELIER*
- ■ Structure : ☞ Annexe C

13.2.10. Contenus_possibles_E/S

- ■ Description : Contient les informations concernant les contenus possibles des différentes entrées-sorties du Robot
- ■ Type : **T_OBJET**
- ■ Lien : • **Acceptation** : Voir ci-dessus
- ■ Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE, CAPP_ATELIER, PPC_ATELIER*
- ■ Structure : ☞ Annexe C

13.2.11. Contenus_effectif_E/S

- ■ Description : Contient les informations concernant les objets présents aux entrées-sorties du Robot
- ■ Type : **OBJET**
- ■ Lien : • **Accueil** : Voir ci-dessus.
- ■ Visible par : *PPC_PALETTAGE, PPC_ATELIER*
- ■ Structure : ☞ Annexe C

13.2.12. Transport_en_cours

- ■ Description : Contient les informations concernant le transport en cours d'exécution: Référence au transport possible exécuté, taux d'avancement et moment où le transport a débuté.
- ■ Type : **TRANSPORT_EXEC**
- ■ Lien : ∅
- ■ Visible par : *PPC_PALETTAGE*
- ■ Structure : ☞ Annexe C

13.2.13. Fabrication_en_cours

- ■ Description : Contient les informations concernant la fabrication en cours d'exécution
- ■ Type : **FABRIC_EXEC**
- ■ Lien : ∅
- ■ Visible par : *PPC_PALETTAGE*
- ■ Structure : ☞ Annexe C

13.2.14. Main_utilisée

- Description : Contient les informations concernant la main effectivement utilisée par le Robot
- Type : **MAIN**
- Lien : ∅
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

13.2.15. Fin_ordre_fabrication

- Description : Permet de signaler la fin de l'exécution de l'ordre de fabrication indiqué.
- Type : **FIN_ORDRE_FABRIC**
- Destination : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

13.2.16. Fin_ordre_transport

- Description : Permet de signaler la fin de l'exécution de l'ordre de transport indiqué.
- Type : **FIN_ORDRE_TRANS**
- Destination : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

13.2.17. Erreurs_ordre_fabrication

- Description : Signale qu'une erreur est apparue lors de l'exécution de l'ordre de fabrication indiqué.
- Type : **ERREUR_ORDRE_FABRIC**
- Destination : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

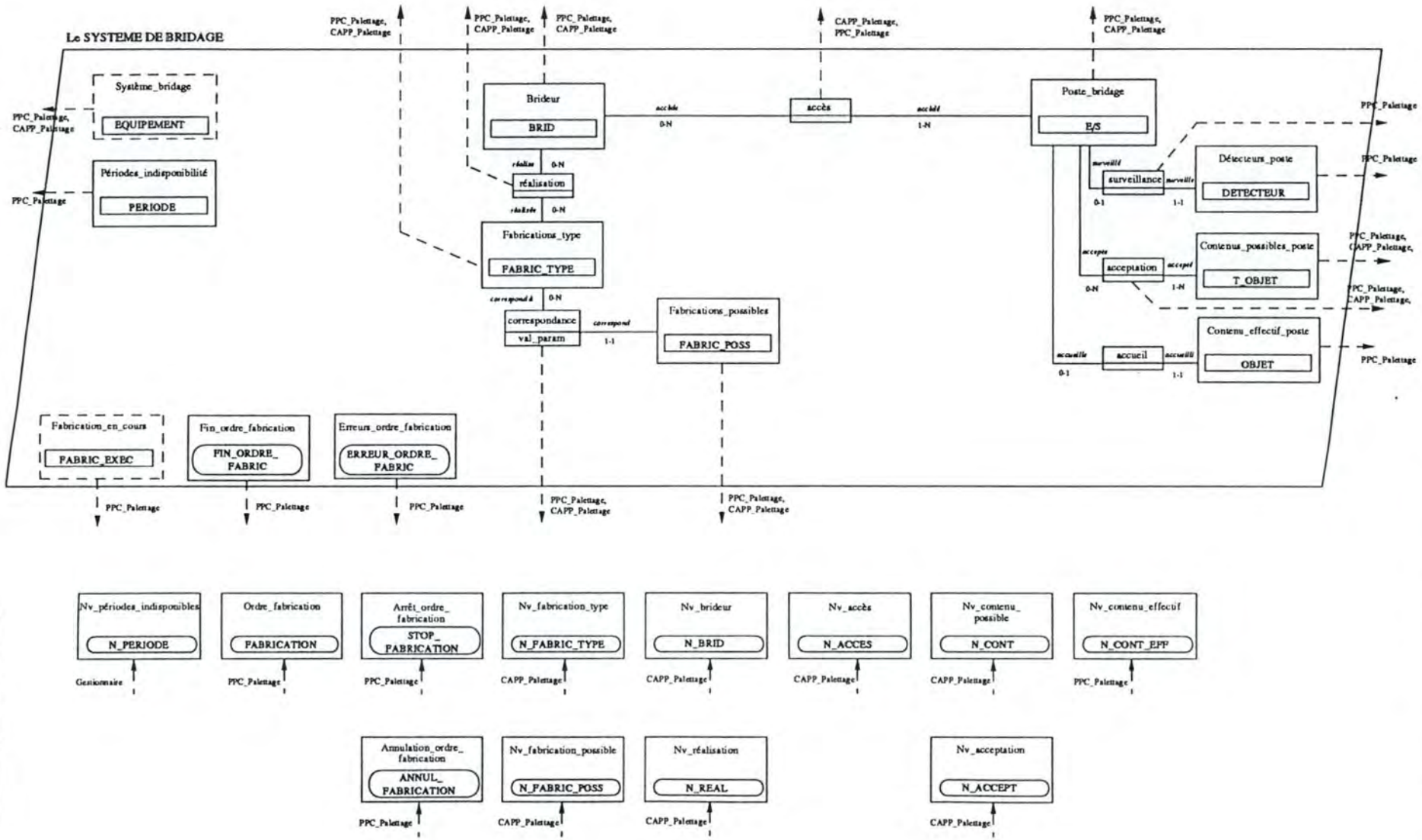
13.2.18. Erreurs_ordre_transport

- Description : Fait savoir qu'une erreur a empêché l'exécution du transport indiqué.
- Type : **ERREUR_ORDRE_TRANS**
- Destination : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

14. Système de bridage

14.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent Système de Bridage.


Le SYSTEME DE BRIDAGE
Système_bridage
EQUIPEMENT
Périodes_indisponibilité
PERIODE
Brideur
BRID
accès
Poste_bridage
E/S
réalisation
Fabrications_type
FABRIC_TYPE
correspondance
val_param
Fabrications_possibles
FABRIC_POSS
surveillance
Détecteurs_poste
DETECTEUR
acceptation
Contenus_possibles_poste
T_OBJET
accueil
Contenu_effectif_poste
OBJET
Fabrication_en_cours
FABRIC_EXEC
Fin_ordre_fabrication
FIN_ORDRE_FABRIC
Erreurs_ordre_fabrication
ERREUR_ORDRE_FABRIC
Nv_périodes_indisponibles
N_PERIODE
Ordre_fabrication
FABRICATION
Arrêt_ordre_fabrication
STOP_FABRICATION
Nv_fabrication_type
N_FABRIC_TYPE
Nv_brideur
N_BRID
Nv_accès
N_ACCES
Nv_contenu_possible
N_CONT
Nv_contenu_effectif
N_CONT_EFF
Annulation_ordre_fabrication
ANNUL_FABRICATION
Nv_fabrication_possible
N_FABRIC_POSS
Nv_réalisation
N_REAL
Nv_acceptation
N_ACCEPT

14.2. Commentaires concernant le schéma de l'agent

14.2.1. Système_bridage

- Description : Contient les informations concernant le système de bridage
- Type : **EQUIPEMENT**
- Lien : ∅
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

14.2.2. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes d'indisponibilité de l'équipement Système de bridage
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

14.2.3. Brideur

- Description : Contient les informations concernant le(s) brideur(s). Ces informations indiquent entre autres s'il est en état de fonctionner et s'il ne réalise que des bridages verticaux ou horizontaux.
- Type : **BRID**
- Liens :
  - **Accès** : Relie le brideur aux Entrées-Sorties auquel il a accès.
  - **Réalisation** : Relie le brideur aux Fabrications_type qu'il est capable de réaliser
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

14.2.4. Fabrications_type

- Description : Contient les informations concernant les différentes fabrications que le système de bridage est en mesure d'effectuer. On peut comparer ces informations à un catalogue des types de fabrication, que l'on peut demander au bridage, telles que : "Brider", "Débrider",...
- Type : **FABRIC_TYPE**
- Lien :
  - **Correspondance** : Relie les Fabrications_type aux Fabrications_possibles correpondantes.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

14.2.5. Fabrications_possibles

- Description : Contient les informations (type d'objet en input, en output, durée,...) concernant les différentes fabrications que les système de bridage est en mesure d'effectuer telles quelles. Ici la description des fabrications est plus précise que celles contenues dans les fabrications types. En effet, on connait les valeurs des paramètres. Ces fabrications possibles sont du genre : "Brider la palette se trouvant au poste 2"
- Type : **FABRIC_POSS**
- Lien : **Correspondance** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

14.2.6. Poste_bridage

- Description : Contient les informations concernant les postes de bridage du système de bridage
- Type : **E/S**
- Liens :
  - **Accès** : Voir ci-dessus.
  - **Surveillance** : Relie le Détecteur à l'Entrée-Sortie qu'il surveille
  - **Acceptation** : Relie les objets aux Entrées-Sorties qui les acceptent.
  - **Acceuil** : Relie le Contenu effectif au poste où il se trouve.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

14.2.7. Détecteurs_poste

- Description : Contient les informations concernant les détecteurs chargés de surveiller les entrées-sorties
- Type : **DETECTEUR**
- Lien :
  - **Surveillance** : Voir ci-dessus.
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

14.2.8. Contenus_possibles_poste

- Description : Contient les informations concernant les types d'objet que l'on peut placer sur les entrées-sorties du système de bridage
- Type : **T_OBJET**
- Lien :
  - **Acceptation** :Voir ci-dessus.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*

■ Structure ☞ Annexe C

14.2.9. Contenu_effectif_poste

■ Description Contient les informations concernant l'objet se trouvant réellement sur une entrée-sortie du système de bridage
■ Type **OBJET**
■ Lien • **Acceuil** : Voir ci-dessus.
■ Visible par *PPC_PALETTAGE, PPC_ATELIER*
■ Structure ☞ Annexe C

14.2.10. Fabrication_en_cours

■ Description Contient les informations concernant la fabrication en cours d'exécution: référence à l'opération type, valeur des paramètres, taux d'avancement de l'exécution et moment où l'on a commencé cette fabrication..
■ Type **FABRIC_EXEC**
■ Lien ∅
■ Visible par *PPC_PALETTAGE*
■ Structure ☞ Annexe C

14.2.11. Fin_ordre_fabrication

■ Description Permet de signaler que l'ordre de fabrication demandé est terminé.
■ Type **FIN_ORDRE_FABRIC**
■ Destination *PPC_PALETTAGE*
■ Structure ☞ Annexe C

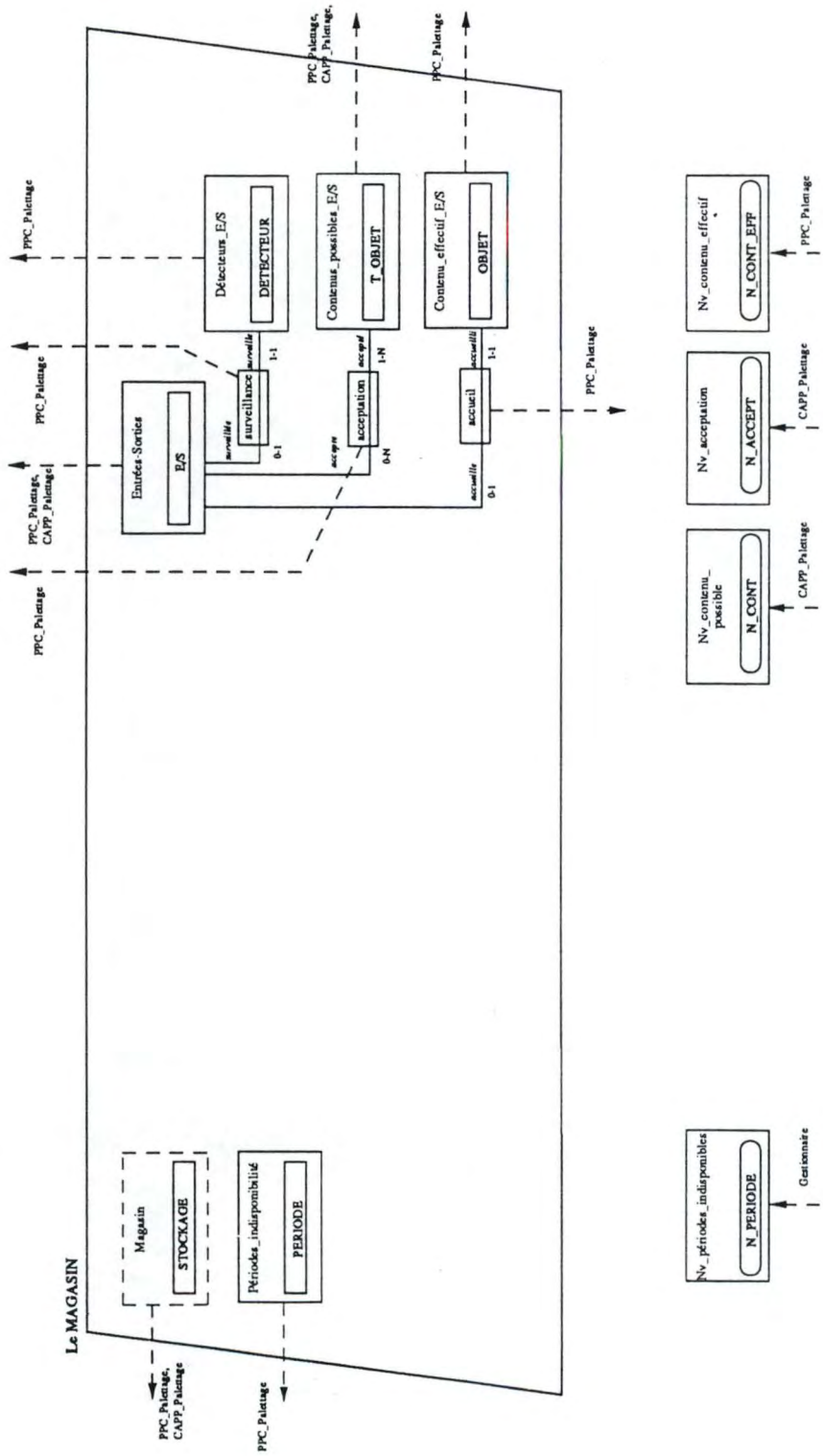
14.2.12. Erreurs_ordre_fabrication

■ Description Permet de signaler qu'une erreur s'est produiteet a empêché d'exécuter l'ordre de fabrication spécifié.
■ Type **ERREUR_ORDRE_FABRIC**
■ Destination *PPC_PALETTAGE*
■ Structure ☞ Annexe C

15. Le magasin

15.1. Structure d'état de l'agent

On trouvera à la figure suivante la structure d'état de l'agent Magasin.


Le MAGASIN
Magasin
STOCKAGE
Périodes_indisponibilité
PERIODE
Entrées-Sorties
E/S
Détecteur_E/S
DETECTEUR
surveillance
Contenus_possibles_E/S
T_OBJET
acceptation
Contenu_effectif_E/S
OBJET
accueil
PPC_Palotage
PPC_Palotage, CAPP_Palotage
surveille
accepte
accueille
1-1
0-1
1-N
0-N
Nv_contenu_ effectif
N_CONT_EFF
Nv_acceptation
N_ACCEPT
Nv_contenu_ possible
N_CONT
Nv_périodes_indisponibles
N_PERIODE
CAPP_Palotage
Gestionnaire

15.2. Description du schéma de l'agent

15.2.1. Magasin

- Description : Contient les informations concernant le magasin
- Type : **STOCKAGE**
- Lien : ∅
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

15.2.2. Périodes_indisponibilité

- Description : Contient les informations concernant les périodes durant lesquelles le magasin est indisponible
- Type : **PERIODE**
- Lien : ∅
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

15.2.3. Entrées-Sorties

- Description : Contient les informations concernant les Entrées-Sorties constituant le magasin
- Type : **E/S**
- Liens :
  - **Surveillance** : Relie l'Entrée-Sortie au Détecteur chargé de la surveiller.
  - **Acceptation** : Relie l'Entrée-Sortie au contenu qu'elle peut accepter.
  - **Accueil** : Relie les Entrées-Sorties aux objets qu'elles accueillent réellement.
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

15.2.4. Détecteurs_E/S

- Description : Contient les informations concernant les éétecteurs chargés de surveiller les Entrées-Sorties du magasin
- Type : **DETECTEUR**
- Lien : • **Surveillance** : Voir ci-dessus
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

15.2.5. Contenus_possibles_E/S

- Description : Contient les informations concernant les types d'objets que les Entrées-Sorties du magasin peuvent accepter
- Type : **T_OBJET**
- Lien : • **Acceptation** : Voir ci-dessus
- Visible par : *PPC_PALETTAGE, CAPP_PALETTAGE*
- Structure : ☞ Annexe C

15.2.6. Contenu_effectif_E/S

- Description : Contient les informations au sujet des objets réellement situés aux entrées-sorties du magasin
- Type : **OBJET**
- Lien : • **Accueil** : Voir ci-dessus
- Visible par : *PPC_PALETTAGE*
- Structure : ☞ Annexe C

Table des matières de l'annexe A

Annexe B

Spécification complète des agents constituant le système AGV

1. Introduction

Nous présentons, dans cette annexe, les spécifications complètes des composants qui constituent le système AGV, à savoir: le serveur, le gestionnaire et l'AGV.

2. Spécification du serveur

2.1. Structure des états

Le schéma décrivant la structure du serveur est donné à la figure B.1.

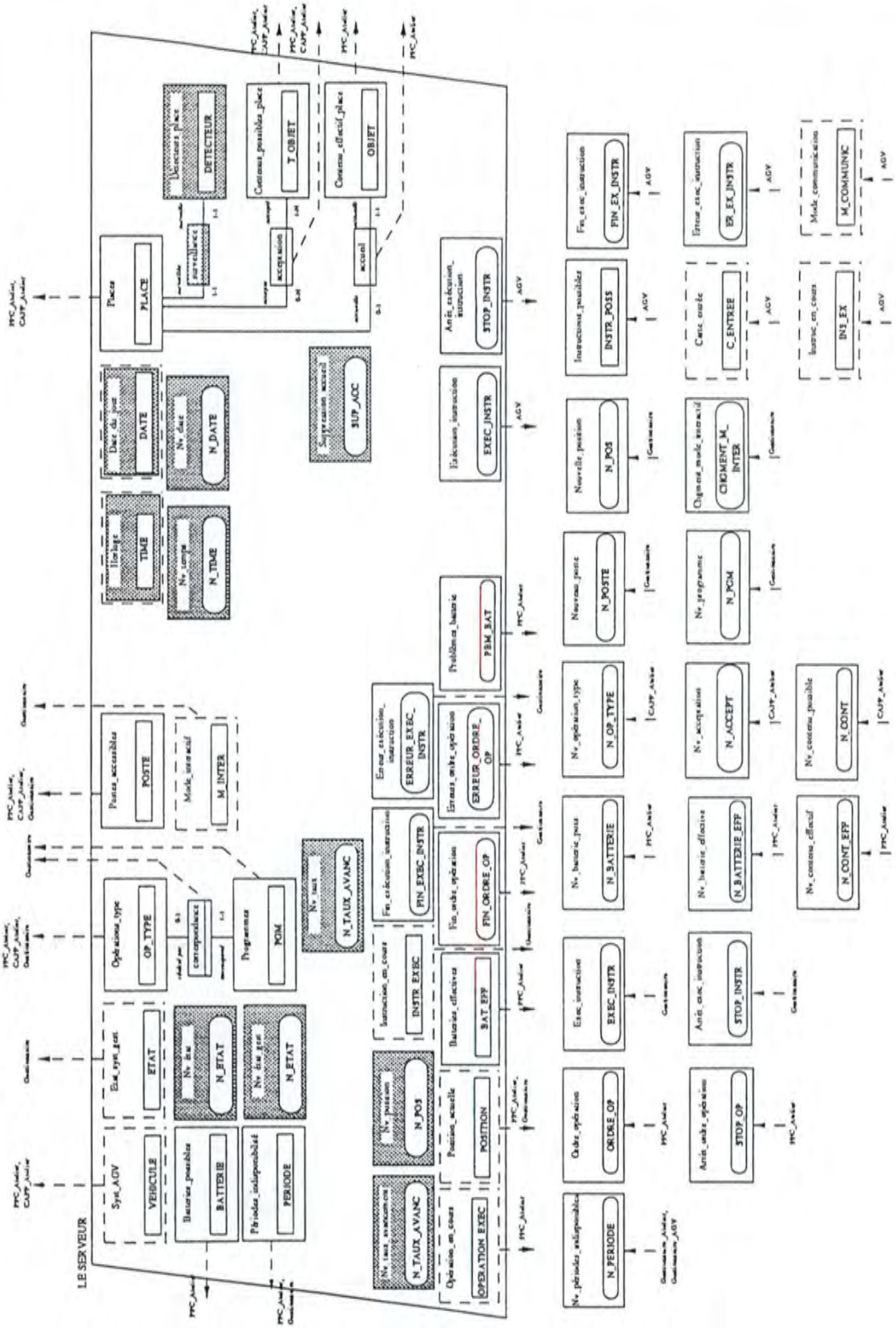


Figure -**B.1.**- Structure des états du serveur

2.2. Commentaires

Le lecteur peut constater que la structure d'état du serveur et celle du système AGV sont très proches. Afin de réduire les redondances inutiles, nous ne détaillerons plus les éléments pour lesquels les commentaires sont identiques et renvoyons le lecteur au chapitre 5 de ce mémoire pour les points non abordés.

Informations gérées par l'agent

- Périodes_indisponibilité

  - Description : Périodes durant lesquelles le système AGV est indisponible.
  - Type : **PERIODE**
  - Lien : ∅
  - Visible par : *PPC_ATELIER, GESTIONNAIRE*
  - Structure : ☞ Annexe C

- Etat_syst_gest

  - Description : Donne l'état du système pour le gestionnaire.
  - Type : **ETAT**
  - Lien : ∅
  - Visible par : *GESTIONNAIRE*
  - Structure : ☞ Annexe C

- Opérations_type

  - Description : Opérations de haut niveau que l'agent peut exécuter.
  - Type : **OP_TYPE**
  - Lien : • **Correspondance**: désigne le programme qui permet de réaliser une opération.
  - Visible par : *PPC_ATELIER, CAPP_ATELIER, GESTIONNAIRE*
  - Structure : ☞ Annexe C

• Programmes

- ■ Description: Ensemble des programmes réalisant des opérations de haut niveau. Il s'agit en fait de séquences d'instructions (exécutables par l'AGV) pour lesquelles les valeurs de paramètres sont définies.
- ■ Type: **PGM**
- ■ Lien: • **Correspondance**: Voir ci-dessus
- ■ Visible par: *GESTIONNAIRE*
- ■ Structure: ☞ Annexe C

• Postes_accessibles

- ■ Description: Ensemble des postes accessibles par le système.
- ■ Type: **POSTE**
- ■ Lien: ∅
- ■ Visible par: *PPC_ATELIER, CAPP_ATELIER, GESTIONNAIRE*
- ■ Structure: ☞ Annexe C

• Position_actuelle

- ■ Description: Indique, au PPC et au gestionnaire de l'AGV,.le poste qu'il occupe (indéterminé si le moyen de transport se déplace)
- ■ Type: **POSITION**
- ■ Lien: ∅
- ■ Visible par: *PPC_ATELIER, GESTIONNAIRE*
- ■ Structure: ☞ Annexe C

• Mode_interactif

- ■ Description: Détermine si le système se trouve dans le mode interactif ou non. Dans le mode interactif les ordres proviennent du gestionnaire de l'AGV tandis que, dans le cas contraire, ils proviennent du PPC de l'atelier.
- ■ Type: **M_INTER**
- ■ Lien: ∅
- ■ Visible par: *GESTIONNAIRE*
- ■ Structure: ☞ Annexe C

• Instruction_en_cours

- Description : Informe le gestionnaire de l'instruction en cours, du moment de lancement de l'opération et du taux d'avancement de son exécution
- Type : **INSTR_EXEC**
- Lien : ∅
- Visible par : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Fin_exécution_instruction

- Description : Renseigne le gestionnaire de la fin d'une opération, du moment où on lui a demandé de l'exécuter, de la durée réelle d'exécution.
- Type : **FIN_EXEC_INSTR**
- Visible par : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Erreur_exécution_instruction

- Description : Informe le gestionnaire des erreurs qui ont interrompu l'exécution de l'opération demandée, de l'heure de prise en charge de l'opération et du taux d'avancement de l'exécution.
- Type : **ERREUR_EXEC_INSTR**
- Visible par : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Exécution_instruction

- Description : Le serveur demande à l'AGV d'exécuter une instruction pour les valeurs de paramètres définies.
- Type : **EXEC_INSTR**
- Visible par : *AGV*
- Structure : ☞ Annexe C

• Arrêt_exécution_instruction

- Description : Demande à l'AGV d'interrompre l'exécution d'une instruction
- Type : **STOP_INSTR**
- Visible par : *AGV*
- Structure : ☞ Annexe C

• Nv_état_gest

- Description : Action qui modifie l'état du système pour le gestionnaire
- Type : **N_ETAT**
- Visible par : ∅
- Structure : ☞ Annexe C

• Nv_taux

- Description : Met à jour le taux d'avancement pour l'exécution d'instruction en cours
- Type : **N_TAUX_AVANC**
- Visible par : ∅
- Structure : ☞ Annexe C

Informations reçues de l'extérieur

• Instructions_possibles

- Description : Donne un descriptif des instructions connues de l'AGV.
- Type : **INSTR_POSS**
- Lien : ∅
- Provient de : *AGV*
- Structure : ☞ Annexe C

• Carte_entrée

- Description : Indique si les places sont occupées, s'il y a perte de piste, si le signal d'urgence est activé,...
- Type : **C_ENTREE**
- Lien : ∅
- Provient de : *AGV*
- Structure : ☞ Annexe C

• Instruc_en_cours

- Description : Renseigne le serveur de l'instruction qu'exécute l'AGV.
- Type : **INS_EX**
- Lien : ∅
- Provient de : *AGV*
- Structure : ☞ Annexe C

• Mode_communication

- Description : Détermine si l'AGV reçoit ses ordres de la télécommande ou du serveur.
- Type : **M_COMMUNIC**
- Lien : ∅
- Provient de : *AGV*
- Structure : ☞ Annexe C

• Nv_périodes_indisponibles

- Description : PPC de l'atelier et gestionnaire AGV indiquent les moments au cours desquels le système est hors d'usage.
- Type : **N_PERIODE**
- Origine : *GESTIONNAIRE_ATELIER, GESTIONNAIRE_AGV*
- Structure : ☞ Annexe C

• Exec_instruction

- Description : Le gestionnaire demande d'exécuter une opération pour une certaine valeur de paramètres.
- Type : **EXEC_INSTR**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Arrêt_exec_instruction

- Description : Demande l'arrêt de l'instruction que le serveur exécute.
- Type : **STOP_INSTR**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Nouveau_poste

- Description : Permet de mettre à jour l'ensemble des postes accessibles.
- Type : **N_POSTE**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

- Nv_programme

  - Description : Par cette action, le gestionnaire fait connaître au serveur le programme qui réalisera l'opération mentionnée.
  - Type : **N_PGM**
  - Origine : *GESTIONNAIRE*
  - Structure : ☞ Annexe C

- Nouvelle_position

  - Description : Le gestionnaire indique la nouvelle position occupée par l'AGV.
  - Type : **N_POS**
  - Origine : *GESTIONNAIRE*
  - Structure : ☞ Annexe C

- Chgment_mode_interactif

  - Description : Permet d'utiliser le serveur dans le mode interactif ou dans le mode automatique.
  - Type : **CHGMENT_M_INTER**
  - Origine : *GESTIONNAIRE*
  - Structure : ☞ Annexe C

- Fin_exécution_instruction

  - Description : L'AGV signale au serveur qu'il a terminé une opération
  - Type : **FIN_EX_INSTR**
  - Origine : *AGV*
  - Structure : ☞ Annexe C

- Erreur_exécution_instruction

  - Description : Indique au serveur que l'AGV n'a pu exécuter l'instruction spécifiée.
  - Type : **ERREUR_EX_INSTR**
  - Origine : *AGV*
  - Structure : ☞ Annexe C

2.3. Les contraintes

Comme pour les commentaires, nous ne reprendrons pas, dans cette partie, les contraintes auxquelles nous n'avons apporté aucune modification. Le lecteur peut consulter le chapitre 5. pour les points non abordés.

Contraintes sur l'état

- *Le système AGV ne peut pas être indéfiniment hors d'usage.*

Etat(Syst_AGV)="*Panne*" **or** Etat(Syst_AGV)="*Indisponible*"
⇒ ◊ Etat (Syst_AGV)="*Marche*"

- *Le système doit à un moment ou à un autre terminer l'exécution d'une opération.*

Concerne (Opération_en_cours)
⇒ ◊ Concerne (Opération_en_cours)=*Indéterminé*

- *Les taux de charges des batteries indiqués au niveau du serveur doivent être égaux à ceux indiqués au niveau de l'AGV (dans la carte d'entrée).*

{Taux_chargement (be1)=Charge_b1(Carte_entrée) **and**
Emplacement (be1)="1" **and**
In_Batteries_effectives (Réf_batterie(be1))} **and**
{Taux_chargement (be2)=Charge_b2(Carte_entrée) **and**
Emplacement (be2)="2" **and**
In_Batteries_effectives (Réf_batterie(be2))}

- *Les valeurs des champs "Présence" des détecteurs doivent correspondre à celles des champs "P1_occupée" et "P2_occupée".*

{Présence (d1)=P1_occupée(Carte_entrée) **and**
In_Détecteurs_place (N°_id(d1)) **and**
In_Surveillance (p1,N°_id(d1))} **and**
{Présence (d2)=P2_occupée(Carte_entrée) **and**
In_Détecteurs_place (N°_id(d2)) **and**
In_Surveillance (p2,N°_id(d2))}

Effets des actions

- *L'action Nouveau_poste permet d'ajouter un élément dans l'ensemble des postes accessibles.*

Gest.Nouveau_poste (p):
Add (Postes_accessibles,p)

- *Cette action donne la nouvelle position qu'occupe le système AGV.*

Gest.Nouvelle_position (npos):
Position_actuelle=npos

• *L'action Chgment_mode_interactif permet de donner une autre valeur au mode interpréteur.*

Gest.Chgment_mode_interactif (change):
Mode_interactif=change

• *L'action Nv_programme associe un programme à une opération type.*

Gest.Nv_programme (np):
Add (Programmes,Est (np)) **and**
Add (Correspondance, <Concerne (np), Nom_programme(Est(np))>)

• *Suite à une demande d'exécution d'une instruction les différents champs de l'instruction en cours sont mis à jour.*

Gest.Exec_instruction (instr):
Concerne (Instruction_en_cours)=Concerne (instr) **and**
Valeurs_paramètres (Instruction_en_cours)= Valeurs_paramètres (instr) **and**
Rens_temp (Instruction_en_cours)=Horloge **and**
Taux_avancement (Instruction_en_cours)=*0*

• *Quand l'exécution d'une opération est terminée, l'instruction en cours est indéterminée.*

Fin_exécution_instr (fin_instr).*Gest*:
Concerne (Instruction_en_cours)=*Indéterminé*

• *S'il y a une erreur dans l'exécution d'une opération, l'instruction en cours est indéterminée.*

Erreur_exécution_instruction (erreur_instr).*Gest*:
Concerne (Instruction_en_cours)=*Indéterminé*

• *L'action Nv_taux indique le taux d'avancement de l'exécution en cours (pour une instruction).*

Nv_taux (nouv_taux):
Taux_avancement (Instruction_en_cours)=nouv_taux

• *L'état du système aux yeux du gestionnaire est mis à jour par l'action Nv_état_gest.*

Nv_état_gest (nouv_état_gest):
Etat_syst_gest = nouv_état_gest

Engagements

- *Suite à une demande d'exécution d'une opération (autre que l'opération d'attente), l'agent répond directement par une erreur ou, alors, il commande l'AGV. Dans ce dernier cas, il génère par la suite une action Erreurs_ordre_opération ou une action Fin_ordre_opération.*

```
PPC.Ordre_opération (ordre)
  With Concerne (ordre)≠"Wait"
  → ◊ (≤10min)
      [Erreurs_ordre_opération (erreur).PPC
       With Concerne (erreur)=Concerne (ordre) and
            Valeurs_paramètres (erreur)=Valeurs_paramètres (ordre) and
            Rens_temp (erreur)=Rens_temp (Opération_en_cours) and
            Taux_avancement (erreur)
              =Taux_avancement (Opération_en_cours)] ⊕
      [Exécution_instruction (ins).AGV;
       {Fin_ordre_opération (fin).PPC
        With Concerne (fin)=Concerne (ordre) and
             Valeurs_paramètres (fin)
                  =Valeurs_paramètres (ordre) and
             Rens_temp (fin)=Rens_temp (Opération_en_cours) and
             Durée_réelle_exec (fin)
                  =Dif_temps (Horloge,Rens_temp(Opération_en_cours))} ⊕
       {Erreurs_ordre_opération (erreur).PPC
        With Concerne (erreur)=Concerne (ordre) and
             Valeurs_paramètres (erreur)=Valeurs_paramètres (ordre) and
             Rens_temp (erreur)=Rens_temp (Opération_en_cours) and
             Taux_avancement (erreur)
                  =Taux_avancement (Opération_en_cours)}]
```

- *Quand le PPC demande au système AGV de s'immobiliser pendant un certain temps, le serveur s'engage à générer une action Erreurs _ordre_opération (s'il rencontre un problème) ou Fin_ordre_opération.*

```
PPC.Ordre_opération (ordre)
  With Concerne (ordre)="Wait"
  → ◊ (≤10min)
      [Erreurs_ordre_opération (erreur).PPC
       With Concerne (erreur)=Concerne (ordre) and
            Valeurs_paramètres (erreur)=Valeurs_paramètres (ordre) and
            Rens_temp (erreur)=Rens_temp (Opération_en_cours) and
            Taux_avancement (erreur)
                  =Taux_avancement (Opération_en_cours)] ⊕
      [Fin_ordre_opération (fin).PPC
       With Concerne (fin)=Concerne (ordre) and
            Valeurs_paramètres (fin)=Valeurs_paramètres (ordre) and
            Rens_temp (fin)=Rens_temp (Opération_en_cours) and
            Durée_réelle_exec (fin)
                  =Dif_temps (Horloge,Rens_temp (Opération_en_cours))]
```

• *Quand le PPC demande d'arrêter l'exécution d'une opération, l'agent s'engage à stopper l'AGV et à générer ensuite une action Erreurs_ordre_opération.*

```
PPC.Arrêt_ordre_opération (ar_ordre)
   → ◊ (≤1min)
      {Arrêt_exécution_instruction (arrêt_ins).AGV
         With Concerne (arrêt_ins)=AGV.Concerne (Instruction_en_cours) and
            Valeurs_paramètres(arrêt_ins)=
                  AGV.Valeurs_paramètres (Instruction_en_cours)};
      {Erreurs_ordre_opération (erreur).PPC
         With Concerne (erreur)=Concerne (ar_ordre) and
            Valeurs_paramètres (erreur)=Valeurs_paramètres (ar_ordre) and
            Rens_temp (erreur)=Rens_temp (Opération_en_cours) and
            Taux_avancement (erreur)
                  =Taux_avancement(Opération_en_cours) and
            Cause(erreur)="Arrêt"}
```

• *Si le gestionnaire demande d'exécuter une instruction (autre que l'instruction "Exécuter"), le serveur s'engage à la faire exécuter par le moyen de transport. Il répond ensuite par une action Fin_exécution_instruction (si tout s'est produit normalement) ou par une action Erreur_exécution_instruction (dans le cas contraire).*

```
Gest.Exec_instruction (instr)
  With Concerne (instr)≠"Exécuter"
   →◊ (≤5min)
      Exécution_instruction (instr).AGV;
        [Fin_exécution_instruction (fin_instr).Gest
          With Concerne (fin_instr)=Concerne (instr) and
             Valeurs_paramètres (fin_instr)=Valeurs_paramètres (instr) and
             Rens_temp (fin_instr)=Rens_temp (Instruction_en_cours) and
             Durée_réelle_exec (fin_instr)
                =Dif_temps (Horloge,Rens_temp(Instruction_en_cours))] ⊕
        [Erreur_exécution_instruction (er_instr).Gest
          With Concerne (er_instr)=Concerne (instr) and
             Valeurs_paramètres (er_instr)=Valeurs_paramètres (instr) and
             Rens_temp (er_instr)=Rens_temp (Instruction_en_cours) and
             Taux_avancement (er_instr)
                   =Taux_avancement (Instruction_en_cours)]
```

• *Suite à une action d'arrêt provenant du gestionnaire, le serveur s'engage à interrompre l'activité de l'AGV et à répondre par une action Erreur_exécution_instruction.*

```
Gest.Arrêt_exec_instruction (ar_instr)
  With Concerne (instr)≠"Exécuter"
  → ◊ (≤1min)
      Arrêt_exécution_instruction (ar_instr).AGV;
      Erreur_exécution_instruction (er_instr)
        With Concerne (er_instr)=Concerne (ar_instr) and
             Valeurs_paramètres (er_instr)=Valeurs_paramètres (ar_instr) and
             Rens_temp (er_instr)=Rens_temp (Instruction_en_cours) and
             Taux_avancement (er_instr)
                  =Taux_avancement (Instruction_en_cours) and
             Cause (er_instr)="Arrêt"
```

• *Si le gestionnaire demande d'exécuter un programme (autre que celui associé à l'opération d'attente), le serveur répond par un message d'erreur ou, alors, il commande l'AGV. Dans ce dernier cas, il s'engage à générer par la suite l'action Fin_exécution_instruction ou Erreur_exécution_instruction.*

```
Gest.Exec_instruction(instr)
  With  Concerne (instr)="Exécuter"
        Valeur(paramètre)≠"Wait"
  → ◊ (≤10min)
      [Erreur_exécution_instruction(er_instr).Gest
        With Concerne (er_instr)=Concerne (instr) and
             Valeurs_paramètres(er_instr)=Valeurs_paramètres(instr) and
             Rens_temp(er_instr)=Rens_temp(Instruction_en_cours) and
             Taux_avancement(er_instr)
                  =Taux_avancement(Instruction_en_cours)] ⊕
      [Exécution_instruction(ins).AGV;
        {Fin_exécution_instruction(fin_instr).Gest
          With Concerne (fin_instr)=Concerne (instr) and
               Valeurs_paramètres(fin_instr)=Valeurs_paramètres(instr) and
               Rens_temp(fin_instr)=Rens_temp(Instruction_en_cours) and
               Durée_réelle_exec(fin_instr)
                 =Dif_temps(Horloge,Rens_temp(Instruction_en_cours))} ⊕
        {Erreur_exécution_instruction(er_instr).Gest
          With Concerne (er_instr)=Concerne (instr) and
               Valeurs_paramètres (er_instr)=Valeurs_paramètres (instr) and
               Rens_temp(er_instr)=Rens_temp(Instruction_en_cours) and
               Taux_avancement(er_instr)
                  =Taux_avancement(Instruction_en_cours)}]

{Remarque:      paramètre ∈ Valeurs_paramètres (instr) and
                Nom(paramètre)="Nom du programme"}
```

• *Une minute après avoir reçu l'ordre d'interrompre l'exécution d'un programme, le serveur doit avoir arrêté l'AGV et répondu par un message d'erreur.*

```
Gest.Arrêt_exec_instruction(ar_instr)
  With Concerne (ar_instr)="Exécuter"
  →◊ (≤1min)
      Arrêt_exécution_instruction(ar_instr).AGV;
      Erreur_exécution_instruction(er_instr).Gest
        With Concerne (er_instr)=Concerne (ar_instr) and
             Valeurs_paramètres (er_instr)=Valeurs_paramètres (ar_instr) and
             Rens_temp(er_instr)=Rens_temp(Instruction_en_cours) and
             Taux_avancement(er_instr)
                  =Taux_avancement(Instruction_en_cours) and
             Cause(erreur)="Arrêt"
```

• *Si le gestionnaire demande d'exécuter le programme correspondant à l'opération d'attente, le serveur lui répond par une action d'erreur ou de fin.*

```
Gest.Exec_instruction(instr)
  With  Concerne (instr)="Exécuter" and
        Valeur (paramètre)="Wait"
  → ◊ (≤10min)
      [Erreur_exécution_instruction(er_instr)
        With  Concerne (er_instr)=Concerne (ar_instr) and
              Valeurs_paramètres (er_instr)=Valeurs_paramètres (ar_instr) and
              Rens_temp(er_instr)=Rens_temp(Instruction_en_cours) and
              Taux_avancement(er_instr)
                        =Taux_avancement(Instruction_en_cours)] ⊕
      [Fin_exécution_opération(fin_instr)
        With  Concerne (fin_instr)=Concerne (instr) and
              Valeurs_paramètres(fin_instr)=Valeurs_paramètres(instr)
              Rens_temp(fin_instr)=Rens_temp(Instruction_en_cours) and
              Durée_réelle_exec(fin_instr)
                   =Dif_temps(Horloge,Rens_temp(Instruction_en_cours))]

{Remarque:    paramètre ∈ Valeurs_paramètres(instr) and
              Nom(paramètre)="Nom du programme" }
```

Responsabilités

• *L'action Nouvelle_position provenant du gestionnaire n'est pas prise en considération quand elle ne fait pas référence à un poste accessible.*

```
F[Gest.Nouvelle_position(npos)/
        NOT In_Postes_accessibles(npos)]
```

• *L'action Chgment_mode_interactif n'a aucune conséquence si une exécution est en cours ou si le mode de communication courant est le mode télécommande.*

F[*Gest*.Chgment_mode_interactif(change)/
Concerne (Instruction_en_cours)≠*Indéterminé* **or**
Concerne (Opération_en_cours)≠*Indéterminé* **or**
AGV.Mode_communication="*Télécom*"]

• *On ne tient pas compte d'un nouveau programme s'il ne correspond pas à une opération type existante ou s'il comprend des instructions que l'AGV ne peut réaliser.*

F[*Gest*.Nv_programme(np)/
NOT In_Opérations_type(Concerne(np)) **or**
{∃ elem ∈ Corps_programme(Est(np)):
NOT In_Instructions_possibles(Instruction(elem))}]

• *Le serveur n'exécute pas l'instruction reçue du gestionnaire s'il est indisponible ou en panne pour le gestionnaire, ou s'il exécute déjà une autre instruction.*

F[*Gest*.Exec_instruction(instr)/
Etat_syst_gest="*Panne*" **or**
Etat_syst_gest="*Indisponible*" **or**
Concerne (Instruction_en_cours)≠*Indéterminé*]

• *Le serveur ne prend pas en considération la demande d'arrêt si les paramètres communiqués ne correspondent pas à ceux de l'instruction en cours ou s'il ne fait rien.*

F[*Gest*.Arrêt_exec_instruction(ar_instr)/
Concerne (Instruction_en_cours)=Indéterminé **or**
Concerne (Instruction_en_cours)≠Concerne (ar_instr) **or**
Valeurs_paramètres (Instruction_en_cours)
≠Valeurs_paramètres(ar_instr)]

• *Le serveur doit interrompre l'exécution d'une opération si celle-ci dépasse les 15 minutes.*

O[Arrêt_exécution_instruction(arrêt_ins).*AGV*/
{Concerne(Instruction_en_cours)≠*Indéterminé* **and**
Dif_temps(Horloge,Rens_temp(Instruction_en_cours))≥*15min}* ***or***
{Concerne(Opération_en_cours)≠*Indéterminé* **and**
Dif_temps(Horloge,Rens_temp(Opération_en_cours))≥*15min}*]

• *Le serveur doit signaler la fin d'exécution d'une opération quand il s'agit de l'opération d'attente et que le temps précisé s'est écoulé.*

O[Fin_ordre_opération (fin).*PPC*
With Concerne (fin)="*Wait*" **and**
Valeurs_paramètres (fin)=Valeurs_paramètres (Opération_en_cours) **and**
Rens_temp (fin)=Rens_temp (Opération_en_cours) **and**
Durée_réelle_exec (fin)
=Dif_temps (Horloge,Rens_temp (Opération_en_cours)) /
Concerne(Opération_en_cours)= "Wait" **and**
Dif_temps(Horloge,Rens_temp(Opération_en_cours))
≥Valeur(Valeurs_paramètres(Opération_en_cours))]

• *Le serveur doit signaler la fin d'exécution d'une instruction quand on exécute le programme associé à l'opération d'attente et que le temps spécifié s'est écoulé.*

O[Fin_exécution_instruction (fin_instr).*Gest*
With Concerne (fin_instr)="*Exécutert*" **and**
Valeurs_paramètres (fin_instr)
=Valeurs_paramètres (Instruction_en_cours) **and**
Rens_temp (fin_instr)=Rens_temp (Instruction_en_cours) **and**
Durée_réelle_exec (fin_instr)
=Dif_temps (Horloge,Rens_temp (Instruction_en_cours)) /
Concerne(Instruction_en_cours)= "*Exécuter*" **and**
{∃ param1, param2 ∈ Valeurs_paramètres (Instruction_en_cours):
Nom(param1)="Nom du programme" **and**
Valeur(param1)="Wait" **and**
Nom(param2)="Temps d'attente" **and**
Dif_temps(Horloge,Rens_temp(Opération_en_cours))
≥Valeur(param2)]

• *Le système est en état de marche (aux yeux du PPC de l'atelier) si les cinq conditions suivantes sont vérifiées:*
- *Le signal d'urgence n'est pas activé.*
- *Il n'y a pas perte de piste.*
- *On ne se trouve pas dans une période d'indisponibilité.*
- *Le système ne se trouve pas dans le mode interactif.*
- *L'AGV reçoit ses ordres via la liaison sérielle.*

O[Nv_état("*Marche*")/
AGV.Urgence_activée(Carte_entrée)="*N*" **and**
AGV.Perte_piste(Carte_entrée)="*N*" **and**
{**NOT** ∃ d1,d2: In_Périodes_indisponibilité(d1,d2) **and**
Supérieur(Date_du_jour,d1) **and**
Supérieur(d2,Date_du_jour)} **and**
Mode_interactif="*N*" **and**
AGV.Mode_communication="*Liaison*"]

• *L'état du système (pour le PPC de l'atelier) prend la valeur "Indisponible" s'il se trouvait préalablement dans l'état "Marche" et si on est au cours d'une période d'indisponibilité, ou bien si le mode interactif est le mode courant, ou enfin si le moyen de transport ne peut recevoir d'ordres que de la télécommande.*

```
O[Nv_état("Indisponible")/
        Etat(Syst_AGV)="Marche" and
        {∃ d1,d2: In_Périodes_indisponibilité(d1,d2) and
                Supérieur(Date_du_jour,d1) and
                Supérieur(d2,Date_du_jour)} or
        Mode_interactif="O" or
        AGV.Mode_communication="Télécom"}]
```

• *L'agent est en panne quand le signal d'urgence est activé ou quand l'AGV détecte une perte de piste.*

```
O[Nv_état("Panne")/
        AGV.Urgence_activée(Carte_entrée)="O" or
        AGV.Perte_piste(Carte_entrée)="O"]
```

• *Le gestionnaire peut uniquement utiliser le serveur si les cinq conditions suivantes sont vérifiées:*

- *Le signal d'urgence n'est pas activé.*
- *Il n'y a pas perte de piste.*
- *On ne se trouve pas dans une période d'indisponibilité.*
- *Le système se trouve dans le mode interactif.*
- *L'AGV reçoit ses ordres via la liaison sérielle.*

```
O[Nv_état_gest("Marche")/
        AGV.Urgence_activée(Carte_entrée)="N" and
        AGV.Perte_piste(Carte_entrée)="N" and
        {NOT ∃ d1,d2:  In_Périodes_indisponibilité(d1,d2) and
                       Supérieur(Date_du_jour,d1) and
                       Supérieur(d2,Date_du_jour)} and
        Mode_interactif="O" and
        AGV.Mode_communication="Liaison"]
```

• *L'état (aux yeux du gestionnaire) prend la valeur Indisponible s'il se trouvait préalablement dans l'état "Marche" et si on est au cours d'une période d'indisponibilité, ou bien si le mode interactif n'est pas le mode courant, ou enfin si le moyen de transport ne peut recevoir d'ordres que de la télécommande.*

```
O[Nv_état_gest("Indisponible")/
        Etat_Syst_gest="Marche" and
        {∃ d1,d2:In_Périodes_indisponibilité(d1,d2) and
                Supérieur(Date_du_jour,d1) and
                Supérieur(d2,Date_du_jour)} or
        Mode_interactif="N" or
        AGV.Mode_communication="Télécom"}]
```

• *L'agent est en panne quand le signal d'urgence est activé ou quand l'AGV détecte une perte de piste.*

O[Nv_état_gest("*Panne*")/
AGV.Urgence_activée(Carte_entrée)="*O*" **or**
AGV.Perte_piste(Carte_entrée)="*O*"]

• *Le système ne considère pas l'action Nv_taux si aucune instruction n'est en cours d'exécution ou si le nouveau taux est inférieur au taux d'avancement de l'exécution en cours.*

F[Nv_taux(nouv_taux)/
Concerne (Instruction_en_cours)=*Indéterminé* **and**
(Concerne (Instruction_en_cours)≠*Indéterminé* **and**
Taux_avancement(Instruction_en_cours)>nouv_taux)]

3. Spécification du gestionnaire

3.1. Structure des états

La figure ci-dessous donne la structure d'états du gestionnaire de l'AGV.

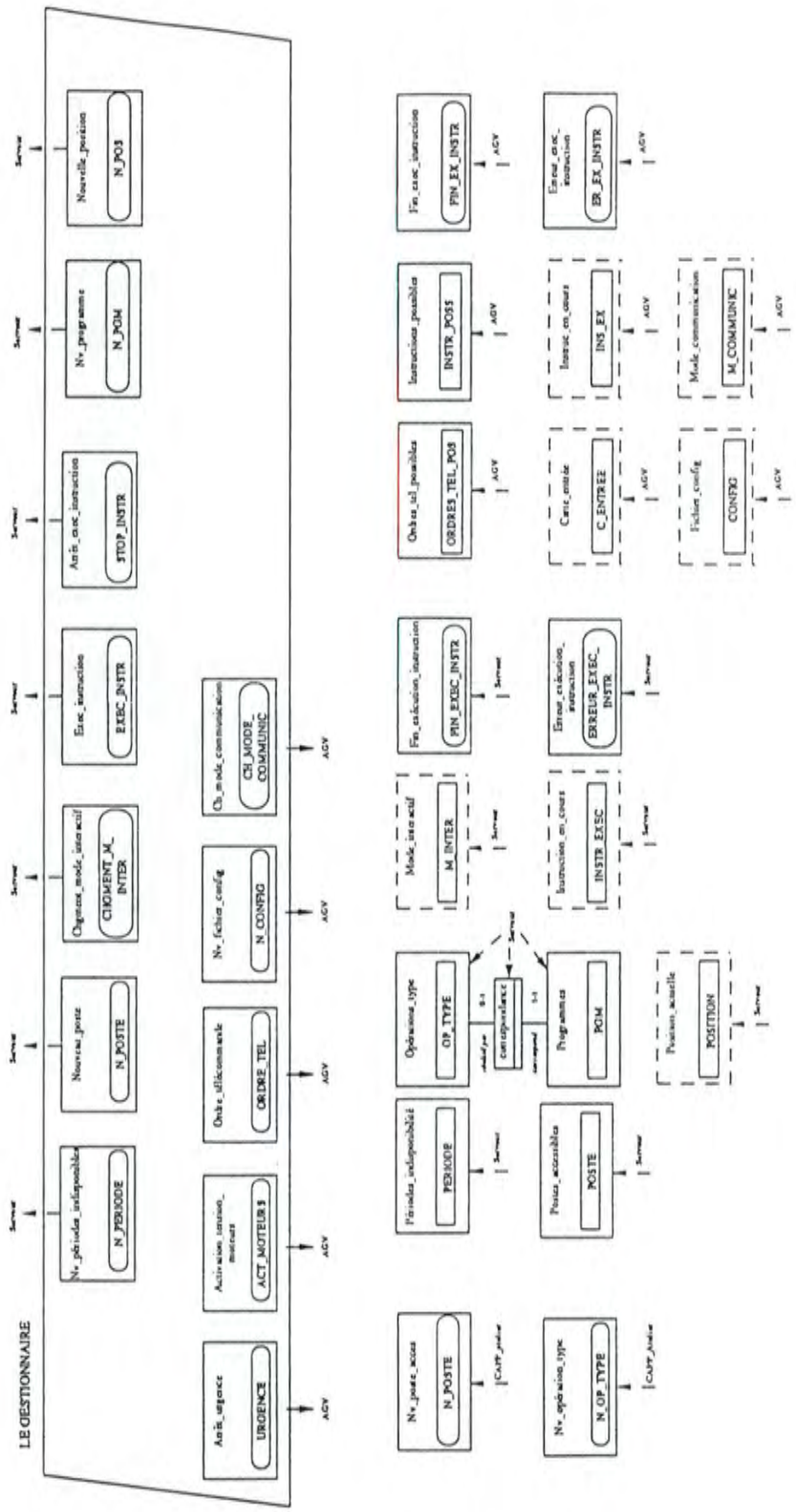


Figure -B.2- Structure des états du gestionnaire

3.2. Commentaires

Informations gérées par l'agent

- Nv_périodes_indisponibles

  - Description : Définit les nouvelles périodes au cours desquelles l'AGV est inutilisable.
  - Type : **N_PERIODE**
  - Destination : *SERVEUR*
  - Structure : ☞ Annexe C

- Nouveau_poste

  - Description : Permet au gestionnaire de metter à jour l'ensemble des postes auxquels l'AGV a accès
  - Type : **N_POSTE**
  - Destination : *SERVEUR*
  - Structure : ☞ Annexe C

- Chgment_mode_interactif

  - Description : Permet de changer le mode d'utilisation du serveur (interactif ou automatique).
  - Type : **CHGMENT_M_INTER**
  - Destination : *SERVEUR*
  - Structure : ☞ Annexe C

- Exec_instruction

  - Description : Le gestionnaire demande au serveur d'exécuter une opération.
  - Type : **EXEC_INSTR**
  - Destination : *SERVEUR*
  - Structure : ☞ Annexe C

- Arrêt_exec_instruction

  - Description : Demande d'interruption de l'instruction en cours.
  - Type : **STOP_INSTR**
  - Destination : *SERVEUR*
  - Structure : ☞ Annexe C

• Nv_programme

- Description : Transmet au serveur le programme réalisant une opération.
- Type : **N_PGM**
- Destination : *SERVEUR*
- Structure : ☞ Annexe C

• Nouvelle_position

- Description : Donne la référence du poste qu'occupe réellement le système AGV.
- Type : **N_POS**
- Destination : *SERVEUR*
- Structure : ☞ Annexe C

• Arrêt_urgence

- Description : Arrête d'urgence toute activité de l'AGV
- Type : **URGENCE**
- Destination : *AGV*
- Structure : ☞ Annexe C

• Activation_tension_moteurs

- Description : Permet d'activer la tension des moteurs suite à un arrêt d'urgence.
- Type : **ACT_MOTEURS**
- Destination : *AGV*
- Structure : ☞ Annexe C

• Ordre_télécommande

- Description : Envoie un ordre par la télécommande.
- Type : **ORDER_TEL**
- Destination : *AGV*
- Structure : ☞ Annexe C

• Nv_fichier_config

- Description : Remplace le fichier de configuration situé sur l'AGV.
- Type : **N_CONFIG**
- Destination : *AGV*
- Structure : ☞ Annexe C

• Ch_mode_communication

- Description : Détermine si l'AGV doit être commandé par la télécommande ou par la liaison sérielle
- Type : **CH_MODE_COMMUNIC**
- Destination : *AGV*
- Structure : ☞ Annexe C

Informations reçues de l'extérieur

• Périodes_indisponibilité

- Description : Reprend les périodes d'indisponibilité de l'AGV et leur cause.
- Type : **PERIODE**
- Lien : ∅
- Provient de : *SERVEUR*
- Structure : ☞ Annexe C

• Postes_accessibles

- Description : Ensemble des informations sur les postes, auxquels le moyen de transport a accès.
- Type : **POSTE**
- Lien : ∅
- Provient de : *SERVEUR*
- Structure : ☞ Annexe C

• Opérations_type

- Description : Opérations de haut niveau que le serveur peut exécuter.
- Type : **OP_TYPE**
- Lien : • **Correspondance**: Associe chaque programme à l'opération qu'il réalise.
- Provient de : *SERVEUR*
- Structure : ☞ Annexe C

• Programmes

- Description : Programmes réalisant les opérations de haut niveau.
- Type : **PGM**
- Lien : • **Correspondance**: Voir ci-dessus.
- Provient de : *SERVEUR*
- Structure : ☞ Annexe C

• Position_actuelle

- ■ Description : Le gestionnaire doit pouvoir connaître le poste que le véhicule occupe.
- ■ Type : **POSITION**
- ■ Lien : ∅
- ■ Provient de : *SERVEUR*
- ■ Structure : ☞ Annexe C

• Mode_interactif

- ■ Description : Indique, à tout moment, si le serveur reçoit ses ordres du PPC (mode automatique) ou du gestionnaire (mode interactif).
- ■ Type : **M_INTER**
- ■ Lien : ∅
- ■ Provient de : *SERVEUR*
- ■ Structure : ☞ Annexe C

• Instruction_en_cours

- ■ Description : Instruction que le serveur est en train d'exécuter.
- ■ Type : **INSTR_EXEC**
- ■ Lien : ∅
- ■ Provient de : *SERVEUR*
- ■ Structure : ☞ Annexe C

• Ordres_tel_possibles

- ■ Description : Ensemble des possibilités qu'offre la télécommande.
- ■ Type : **ORDRES_TEL_POS**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Carte_entrée

- ■ Description : Le gestionnaire a accès au contenu de la carte d'entrée de l'AGV.
- ■ Type : **C_ENTREE**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Fichier_config

- ■ Description : Reprend les informations qui constituent le fichier de configuration de l'AGV.
- ■ Type : **CONFIG**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Instructions_possibles

- ■ Description : Ensemble des instructions que l'AGV sait exécuter.
- ■ Type : **INSTR_POSS**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Instruc_en_cours

- ■ Description : Le gestionnaire peut connaître l'instruction exécutée par l'AGV.
- ■ Type : **INS_EX**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Mode_communication

- ■ Description : Détermine, à tout moment, si les ordres proviennent de la télécommande ou du serveur.
- ■ Type : **M_COMMUNIC**
- ■ Lien : ∅
- ■ Provient de : *AGV*
- ■ Structure : ☞ Annexe C

• Nv_poste_acces

- ■ Description : Le CAPP donne le descriptif d'un nouveau poste auquel l'AGV peut accéder.
- ■ Type : **N_POSTE**
- ■ Origine : *CAPP_ATELIER*
- ■ Structure : ☞ Annexe C

• Nv_opération_type

- Description : Le CAPP demande parfois au gestionnaire de concevoir les programmes qui réalisent de nouvelles opérations dont il communique les caractéristiques
- Type : **N_OP_TYPE**
- Origine : *CAPP_ATELIER*
- Structure : ☞ Annexe C

• Fin_exécution_instruction

- Description : Informe de la fin de l'instruction demandée, de l'heure à laquelle l'exécution a commencé et de la durée réelle de celle-ci.
- Type : **FIN_EXEC_INSTR**
- Origine : *SERVEUR*
- Structure : ☞ Annexe C

• Erreur_exécution_instruction

- Description : Donne les problèmes rencontrés lors de l'exécution d'une instruction ainsi que le moment où l'exécution a débuté et son taux d'avancement quand elle a été interrompue.
- Type : **ERREUR_EXEC_INSTR**
- Origine : *SERVEUR*
- Structure : ☞ Annexe C

• Fin_exec_instruction

- Description : L'AGV signale au gestionnaire qu'il a terminé une instruction (par l'affichage LCD).
- Type : **FIN_EX_INSTR**
- Origine : *AGV*
- Structure : ☞ Annexe C

• Erreur_exec_instruction

- Description : Communique les problèmes survenus lors de l'exécution d'une opération.
- Type : **ERREUR_EX_INSTR**
- Origine : *AGV*
- Structure : ☞ Annexe C

3.3. Les contraintes

Engagements

- *Si le gestionnaire choisit, à un moment donné, de commander le moyen de transport au moyen de la télécommande, il devra, ensuite, faire en sorte que l'AGV puisse recevoir des ordres du serveur.*

```
Ch_mode_communication(ch).AGV With ch="Télécom"
        → ◊ Ch_mode_communication(ch).AGV
                With ch="Liaison"
```

- *Le gestionnaire ne peut pas laisser indéfiniment le système dans le mode interactif.*

```
Chgment_mode_interactif(change).Serveur
    With change="O"
        → ◊ Chgment_mode_interactif(change).Serveur
                With change="N"
```

- *Si le signal d'urgence a été enclenché, la tension des moteurs doit par la suite être activée.*

```
Arrêt_urgence(urg).AGV
        →◊ Activation_tension_moteurs(act).AGV
```

- *Quand le CAPP signale qu'un nouveau poste est accessible, le gestionnaire doit mettre à jour le fichier de configuration et la liste des postes accessibles au niveau du serveur.*

```
CAPP.Nv_poste_acces(p)
            → ◊ Nouveau_poste(p).Serveur and
                Nv_fichier_config(fichier).AGV
```

- *Si le CAPP exige du système de pouvoir exécuter une nouvelle opération, il faudra que le gestionnaire réalise le programme correspondant.*

```
CAPP.Nv_opération_type(op)
        → ◊ Nv_programme(np)
                With Concerne (np)=N°_id(op)
```

4. Spécification de l'AGV

4.1. Structure des états

Le schéma de la structure d'états se trouve à la figure suivante.

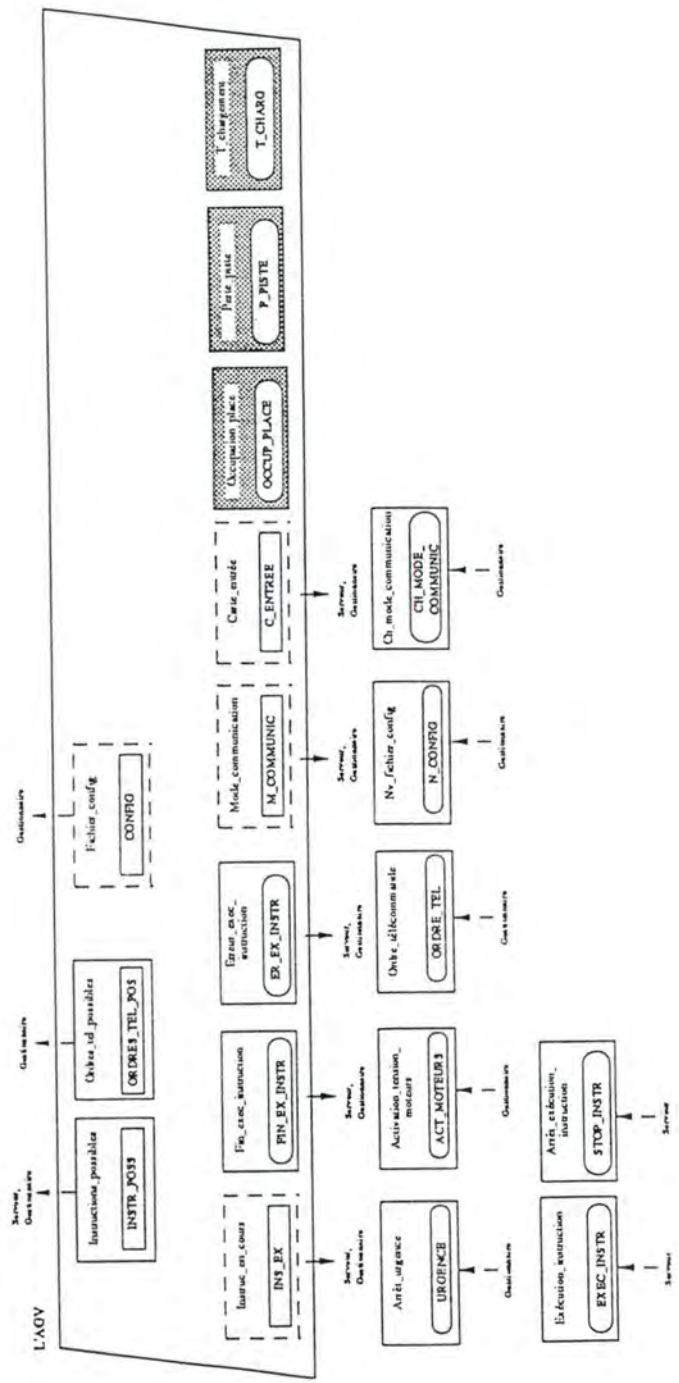


Figure -**B.3.**- Structure des états de l'AGV

4.2. Commentaires

Informations gérées par l'agent

- Instructions_possibles

  ■ Description Informe des caractéristiques (identifiant, description informelle, description des paramètres) de toutes les instructions connues de l'AGV.
  ■ Type **INSTR_POSS**
  ■ Lien ∅
  ■ Visible par *SERVEUR, GESTIONNAIRE*
  ■ Structure ☞ Annexe C

- Ordres_tel_possibles

  ■ Description Informe le gestionnaire des ordres qu'il peut envoyer à partir de la télécommande.
  ■ Type **ORDRE_TEL_POSS**
  ■ Lien ∅
  ■ Visible par *GESTIONNAIRE*
  ■ Structure ☞ Annexe C

- Fichier_config

  ■ Description Ensemble des informations reprises dans le fichier de configuration de l'atelier. Ce fichier décrit la piste sur laquelle l'AGV se déplace (longueur, nombre total de codes détectés au sol, nombre de postes réels) et contient une série d'informations sur les postes figurant sur cette piste (numéro de code à partir de l'origine, distance entre l'origine et le poste, hauteur du palettiseur pour ce poste,...).
  ■ Type **CONFIG**
  ■ Lien ∅
  ■ Visible par *GESTIONNAIRE*
  ■ Structure ☞ Annexe C

• Instruc_en_cours

- Description : Décrit l'instruction en cours d'exécution et la valeur de ses paramètres.
- Type : **INS_EX**
- Lien : ∅
- Visible : *SERVEUR, GESTIONNAIRE*
- Structure : ☞ Annexe C

• Fin_exécution_instruction

- Description : Signale que l'AGV vient de terminer l'exécution de telle instruction pour telle valeur de paramètres.
- Type : **FIN_EX_INSTR**
- Destination : *SERVEUR, GESTIONNAIRE*
- Structure : ☞ Annexe C

• Erreur_exécution_instruction

- Description : Indique que l'instruction demandée n'a pu être exécutée.
- Type : **ERREUR_EX_INSTR**
- Destination : *SERVEUR, GESTIONNAIRE*
- Structure : ☞ Annexe C

• Mode_communication

- Description : Indique à tout moment si l'AGV est commandé par télécommande ou par liaison sérielle.
- Type : **M_COMMUNIC**
- Lien : ∅
- Visible par : *SERVEUR, GESTIONNAIRE*
- Structure : ☞ Annexe C

• Carte_entrée

- Description : Renseigne le serveur et le gestionnaire du contenu de la carte d'entrée. Celle-ci indique si les emplacements du palettiseur sont occupés, si on a déclenché le signal d'urgence, s'il y a perte de piste. Elle donne également le code de télécommande reçu (peut être indéterminé) et le taux de charge effectif des deux batteries.
- Type : **C_ENTREE**
- Lien : ∅
- Visible par : *SERVEUR, GESTIONNAIRE*
- Structure : ☞ Annexe C

• Occupation_place

- Description : Indique si oui ou non on détecte une présence sur la place désignée.
- Type : **OCCUP_PLACE**
- Origine : ∅
- Structure : ☞ Annexe C

• Perte_piste

- Description : Signale toute perte de piste.
- Type : **P_PISTE**
- Origine : ∅
- Structure : ☞ Annexe C

• T_chargement

- Description : Communique le taux de chargement réel des batteries branchées..
- Type : **T_CHARG**
- Origine : ∅
- Structure : ☞ Annexe C

Informations reçues de l'extérieur

• Arrêt_urgence

- Description : Arrête exceptionnellement toute activité de l'AGV.
- Type : **URGENCE**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Activation_tension_moteurs

- Description : Activation la tension des moteurs après enclenchement du signal d'urgence.
- Type : **ACT_MOTEURS**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Ordre_télécommande

- Description : Ordre envoyé à l'AGV par télécommande.
- Type : **ORDRE_TEL**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Nv_fichier_config

- Description : Transmet le nouveau fichier de configuration de l'atelier.
- Type : **N_CONFIG**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Ch_mode_communication

- Description : Permet de changer le mode de communication courant (communication par liaison sérielle ou par télécommande).
- Type : **CH_MODE_COMMUNIC**
- Origine : *GESTIONNAIRE*
- Structure : ☞ Annexe C

• Exécution_instruction

- Description : Informe l'AGV de l'instruction à exécuter et de la valeur de ses paramètres.
- Type : **EXEC_INSTR**
- Origine : *SERVEUR*
- Structure : ☞ Annexe C

• Arrêt_exécution_instruction

- Description : Permet d'interrompre l'exécution d'une instruction.
- Type : **STOP_INSTR**
- Origine : *SERVEUR*
- Structure : ☞ Annexe C

4.3. Les contraintes

Effets des actions:

- *L'action Exécution_instruction met à jour les informations concernant l'instruction en cours d'exécution.*

Serveur.Exécution_instruction(ins):
Concerne (Instruction_en_cours)=Concerne (ins) **and**
Valeurs_paramètres(Instruction_en_cours)=Valeurs_paramètres(ins)

- *Si le signal d'urgence a été activé, le champs le concernant dans la carte d'entrée doit le signaler.*

Gest.Arrêt_urgence(urg):
Urgence_activée(Carte_entrée)="*O*"

- *Une fois les moteurs activés, le signal d'urgence est désactivé.*

Gest.Activation_tension_moteurs(act):
Urgence_activée(Carte_entrée)="*N*"

- *Si le gestionnaire commande l'AGV grâce à la télécommande, la carte d'entrée doit indiquer l'ordre reçu.*

Gest.Ordre_télécommande(com):
Code_tel(Carte_entrée)=com

- *L'action Nv_fichier_config a pour effet de mettre à jour le fichier de configuration.*

Gest.Nv_fichier_config(fichier):
Fichier_config=fichier

- *L'action Ch_mode_communication permet de changer le mode de communication (télécommande ou liaison sérielle).*

Gest.Ch_mode_communication(ch):
Mode_communication=ch

- *Une fois l'exécution terminée, la référence à l'instruction de la variable Instruction_en_cours est indéfinie.*

Fin_exec_instruction(fin_ins):
Concerne (Instruction_en_cours)=*Indéterminé*

• *Suite à l'action Erreur_exec_instruction, le champs "Concerne" de l'instruction en cours est indéterminé.*

Erreur_exec_instruction(erreur_instr):
Concerne(Instruction_en_cours)=*Indéterminé*

• *L'action Occupation_place indique si oui ou non la place désignée est occupée.*

Occupation-place(o_pl):
Place_concernée(o_pl)="1"
=>P1_occupée(Carte_entrée)=Valeur(o_pl) **and**
Place_concernée(o_pl)="2"
=>P2_occupée(Carte_entrée)=Valeur(o_pl)

• *L'action P_piste indique si l'on détecte une perte piste.*

P_piste(pp):
Perte_piste(Carte_entrée) = "O"

• *L'action T_chargement donne le taux de chargement réel de la batterie située à l'emplacement désigné.*

Etat_charge_batterie(e_bat):
De(e_bat) = "1"
⇒ Charge_b1(Carte_entrée) = Valeur(e_bat) **and**
De(e_bat)="2"
⇒ Charge_b2(Carte_entrée) = Valeur(e_bat)

Engagements

• *Quand l'agent reçoit un ordre, il s'engage à répondre dans les cinq minutes par un signal de fin ou d'erreur.*

Serveur.Exécution_instruction(ins)
→ ◊ (≤*5min*)
{Erreur_exécution_instruction(erreur_ins)
With Concerne(erreur_ins)=Concerne(ins) **and**
Valeurs_paramètres(erreur_ins)=Valeurs_paramètres(ins)} ⊕
{Fin_exécution_instruction(fin_ins)
With Concerne (fin_ins)=Concerne (ins) **and**
Valeurs_paramètres(fin_ins)=Valeurs_paramètres(ins)}

• *Après avoir reçu une demande d'arrêt d'activité, l'AGV répond endéans les 30 secondes par une action Erreur_exécution_instruction.*

Serveur.Arrêt_exécution_instruction(arrêt_ins)
→ ◊ (≤*30sec*)
Erreur_exécution_instruction(er_instr)
With Concerne (erreur_ins)=Concerne (arrêt_ins) **and**
Valeurs_paramètres (erreur_ins)
=Valeurs_paramètres (arrêt_ins)

Responsabilités

• *L'agent n'exécute pas l'instruction reçue du serveur, si le mode de communication courant est le mode télécommande ou s'il exécute déjà une autre instruction.*

F[*Serveur*.Exécution_instruction(ins)/
Mode_communication="*Télécom*" **or**
Concerne (Instruction_en_cours)≠*Indéterminé*]

• *L'AGV ne tient pas compte de la demande d'arrêt, si la valeur paramètres de l'action ne correspond pas aux informations concernant l'instruction en cours.*

F[*Serveur*.Arrêt_exécution_instruction(arrêt_ins)/
Concerne (arrêt_ins)≠Concerne (Instruction_en_cours) **or**
Valeurs_paramètres (arrêt_ins)
≠Valeurs_paramètres(Instruction_en_cours]

• *L'agent ne tient pas compte des ordres de la télécommande lorsqu'il ne peut être commandé par la télécommande ou qu'il exécute déjà un autre ordre.*

F[*Gest*.Ordre_télécommande(com)/
Mode_communication="*Liaison*" **or**
Code_tel(Carte_entrée)≠*Indéterminé*]

• *L'action Activation_tension_moteurs est omise si le signal d'urgence n'a pas été activé.*

F[*Gest*.Activation_tension_moteurs(act)/
Urgence_activée(Carte_entrée)="*N*"]

• *On ne considère pas l'action Arrêt_urgence si le signal d'urgence est déjà enclenché.*

F[*Gest*.Arrêt_urgence(urg)/
Urgence_activée(Carte_entrée)="*O*"]

• *L'agent ne prend pas en considération le changement du mode de communication s'il est en train d'exécuter un ordre provenant du serveur ou de la télécommande.*

F[*Gest*.Ch_mode_communication(ch)/
Concerne (Instruction_en_cours)≠*Indéterminé* **or**
Code_tel(Carte_entrée)≠*Indéterminé*]

• *Si le signal d'urgence est activé ou qu'il y a perte de piste lors de l'exécution d'une instruction, on signale l'erreur au serveur.*

O[Erreur_exécution_instruction(erreur_ins).*Serveur*
With Concerne (erreur_ins) =Concerne (Instruction_en_cours) **and**
Valeurs_paramètres(erreur_ins)
= Valeurs_paramètres(Instruction_en_cours)/
Concerne (Instruction_en_cours)≠*Indéterminé* **and**
(Urgence_activée(Carte_entrée)="*O*"**or**
Perte_piste(Carte_entrée)="*O*")]

Table des matières de l'annexe B

Annexe C

Définition des structures de données

1. Introduction

Dans cette annexe, nous donnons les structures de données que nous avons utilisées dans les chapitres précédents.

Afin de faciliter, autant que possible, la lecture de cette annexe, nous avons opté pour un classement alphabétique des types, ainsi que pour les conventions typographiques suivantes :

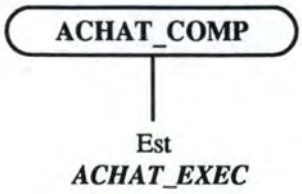
En ***MAJUSCULE-GRAS-ITALIQUE*** , on trouve les types que nous avons définis lors de la modélisation.

En **MAJUSCULE-GRAS,** on trouve les types de base définis dans le langage ALBERT.

2. Définition des structures de données

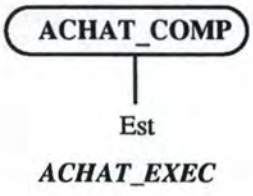
2.1. ACHAT_COMP

CP [Est : ***ACHAT_EXEC***]



2.2. ACHAT_COMP

CP [Est: ***ACHAT_EXEC***]

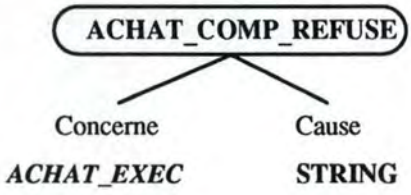


2.3. ACHAT_COMP_RECU

CP [Est: ***ACHAT_EXEC***]

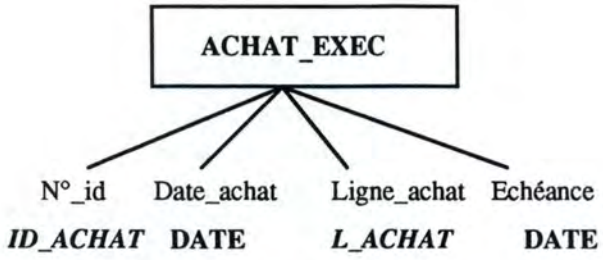
2.4. ACHAT_COMP_REFUSE

CP [Concerne: ***ACHAT_EXEC***
Cause: **STRING**]



2.5. ACHAT_EXEC

CP [N°_id : ***ID_ACHAT***, {à déterminer}
Date_achat : **DATE**,
Ligne_achat : ***L_ACHAT***,
Echéance: **DATE**]



2.6. ACT_MOTEURS

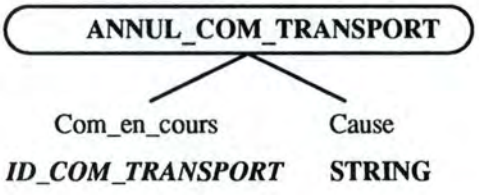
{à déterminer}

2.7. ANNUL_COM_FABRIC

CP [Com_en_cours: ***ID_COM_FABRIC***,
Cause: **STRING**]

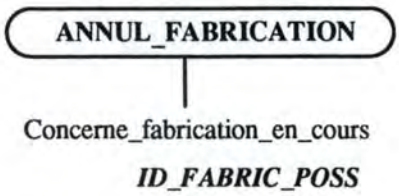
2.8. ANNUL_COM_TRANSPORT

CP [Com_en_cours: ***ID_COM_TRANSPORT***,
Cause: **STRING**]



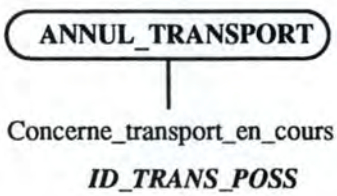
2.9. ANNUL_FABRICATION

CP [Concerne_fabrication_en_cours: ***ID_FABRIC_POSS***]



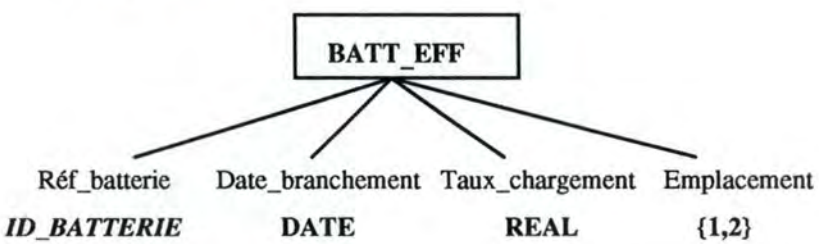
2.10. ANNUL_TRANSPORT

CP [Concerne_transport_en_cours: ***ID_TRANS_POSS***]



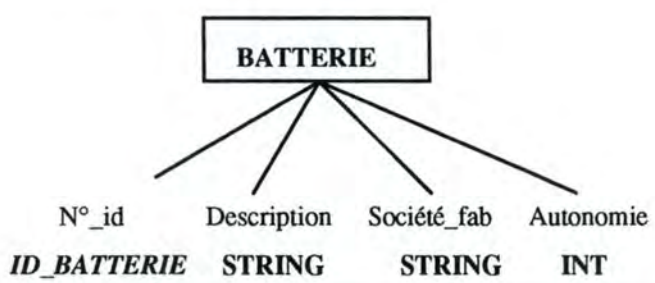
2.11. BATT_EFF

CP [Réf_batterie: ***ID_BATTERIE***, {à déterminer}
Date_branchement: **DATE**,
Taux_chargement: **REAL**,
Emplacement: **{1,2}**]



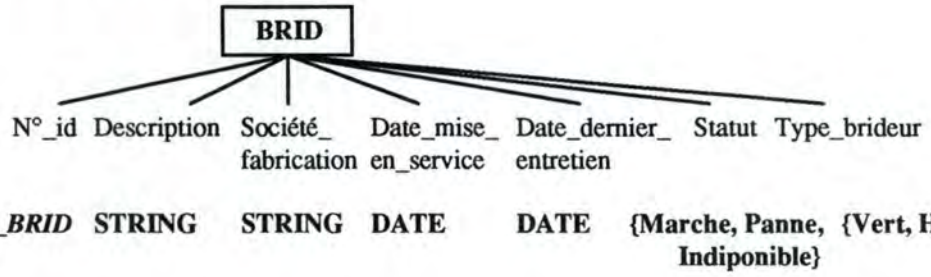
2.12. BATTERIE

CP [N°_id : ***ID_BATTERIE***, {à déterminer}
Description : **STRING**,
Société_fab : **STRING**,
Autonomie: **INT**]



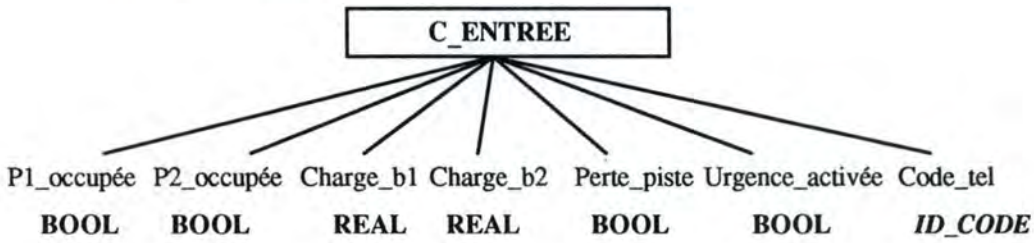
2.13. BRID

CP [N°_id : ***ID_BRID***, {à déterminer}
Description: **STRING**,
Société_fabrication: **STRING**,
Date_mise_en_service: **DATE**,
Date_dernier_entretien: **DATE**,
Statut: **{Marche, Panne, Indisponible}**,
Type_brideur: **{Vert, Hor}**]



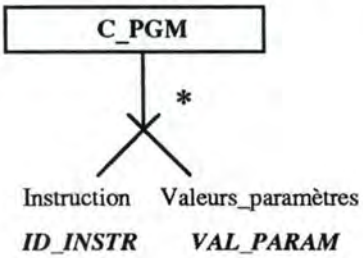
2.14. C_ENTREE

CP [P1_occupée: **BOOL**,
P2_occupée: **BOOL**,
Charge_b1: **REAL**,
Charge_b2: **REAL**,
Perte_piste: **BOOL**,
Urgence_activée: **BOOL**,
Code_tel: ***ID_CODE***]



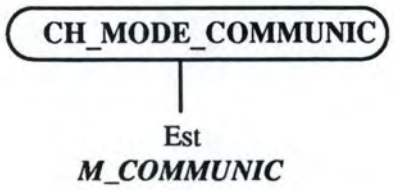
2.15. C_PGM

SEQ[CP [Instruction: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***]]



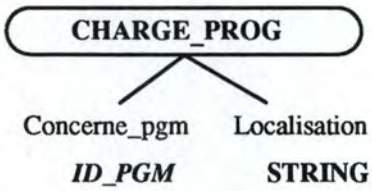
2.16. CH_MODE_COMMUNIC

CP [Est: ***M_COMMUNIC***]



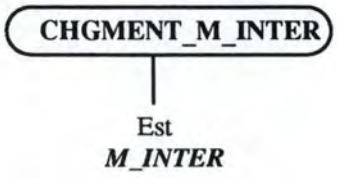
2.17. CHARGE_PROG

CP [Concerne_pgm : ***ID_PGM***,
Localisation: **STRING**]



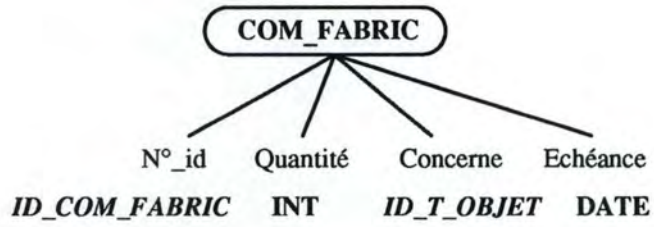
2.18. CHGMENT_M_INTER

CP [Est: ***M_INTER***]



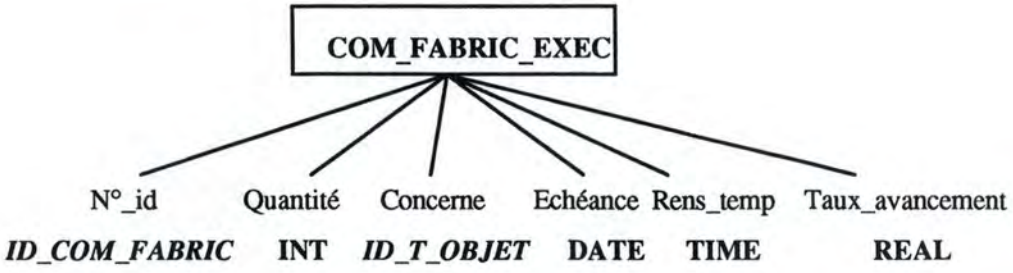
2.19. COM_FABRIC

CP [N°_id : ***ID_COM_FABRIC***, {à déterminer}
Quantité : **INT**,
Concerne : ***ID_T_OBJET***,
Echéance : **DATE**]



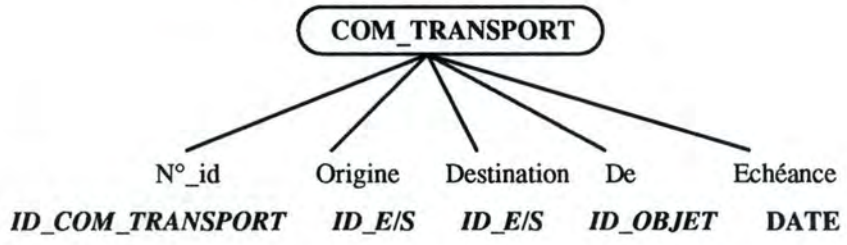
2.20. COM_FABRIC_EXEC

CP [N°_id: ***ID_COM_FABRIC***, {à déterminer}
Quantité: **INT**,
Concerne: ***ID_T_OBJET***,
Echéance: **DATE**,
Rens_Temp: **TIME**,
Taux_avancement: **REAL**]



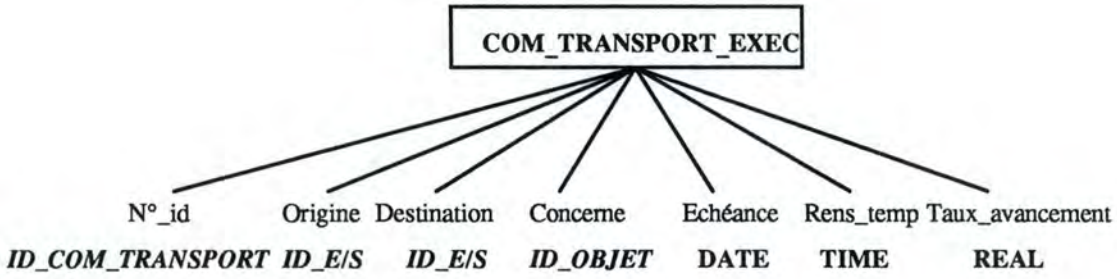
2.21. COM_TRANSPORT

CP [N°_id : ***ID_COM_TRANSPORT***, {à déterminer}
Origine : ***ID_E/S***
Destination : ***ID_E/S***,
Concerne : ***ID_OBJET***,
Echéance : **DATE**]



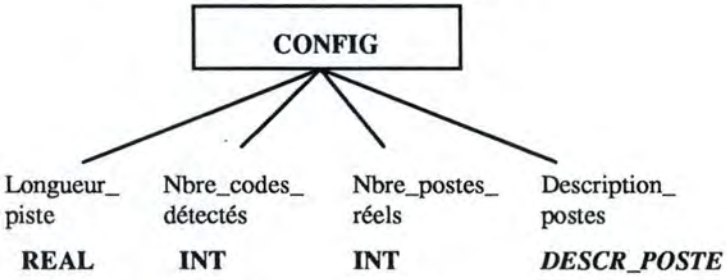
2.22. COM_TRANSPORT_EXEC

CP [N°_id: ***ID_COM_TRANSPORT***, {à déterminer}
Origine: ***ID_E/S***
Destination: ***ID_E/S***,
Concerne: ***ID_OBJET***,
Echéance: **DATE**,
Rens_Temp: **TIME**,
Taux_avancement: **REAL**]



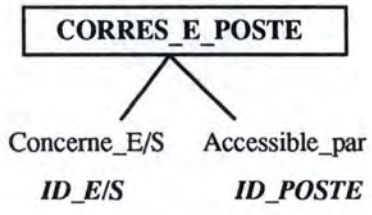
2.23. CONFIG

CP [Longueur_piste: **REAL**,
Nbre_codes_détectés: **INT**,
Nbre_postes_réels: **INT**,
Description_postes: ***DESCR_POSTE***]



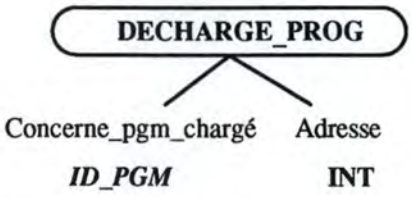
2.24. CORRES_E_POSTE

CP [Concerne_E/S: ***ID_E/S***,
Accessible_par: ***ID_POSTE***]



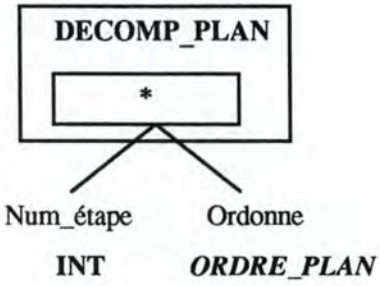
2.25. DECHARGE_PROG

CP [Concerne_pgm_chargé : ***ID_PGM***,
Adresse: **INT**]



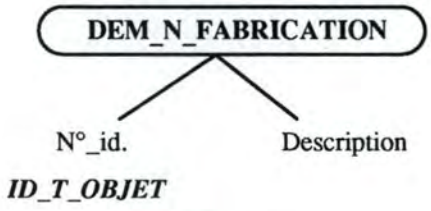
2.26. DECOMP_PLAN

SET[CP [Num_étape: **INT**,
Ordonne: ***ORDRE_PLAN***]]



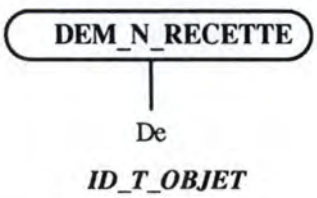
2.27. DEM_N_FABRICATION

CP [N°_id: ***ID_T_OBJET***, {à déterminer}
Description: {à déterminer}]



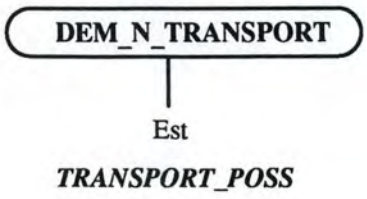
2.28. DEM_N_RECETTE

CP [De: ***ID_T_OBJET***]



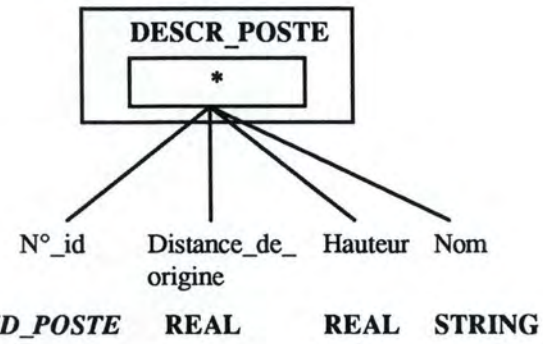
2.29. DEM_N_TRANSPORT

CP [Est: ***TRANSPORT_POSS***]



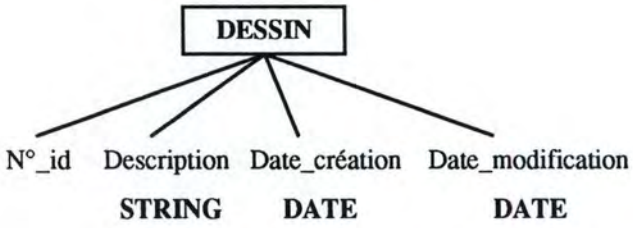
2.30. DESCR_POSTE

SET[CP [N°_id: ***ID_POSTE***,
Distance_de_origine: **REAL**,
Hauteur: **REAL**,
Nom: **STRING**]]



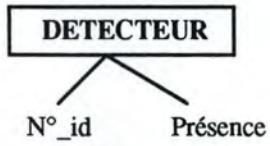
2.31. DESSIN

CP [N°_id : , {à déterminer}
Description : **STRING**,
Date_création : **DATE**,
Date_modification : **DATE**,]



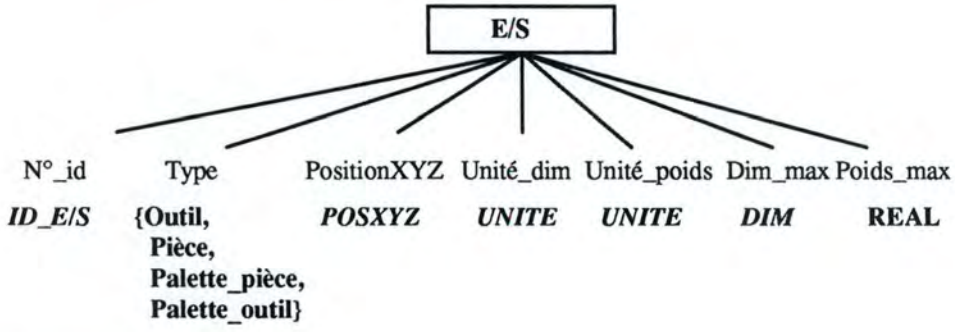
2.32. DETECTEUR

CP [N°_id: ***ID_DETECTEUR***, {à déterminer}
Présence: **BOOL**]



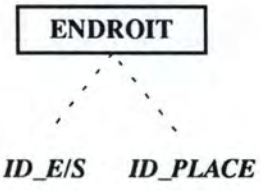
2.33. E/S

CP [N°_id : ***ID_E/S***, {à déterminer}
Type : {**Outil, Pièce, Palette_pièce, Palette_outil**},
PositionXYZ : ***POSXYZ***, {à déterminer}
Unité_dim: ***UNITE***, {à déterminer}
Unité_poids: ***UNITE***, {à déterminer}
Dim_max : ***DIM***, {à déterminer}
Poids_max : **REAL**]



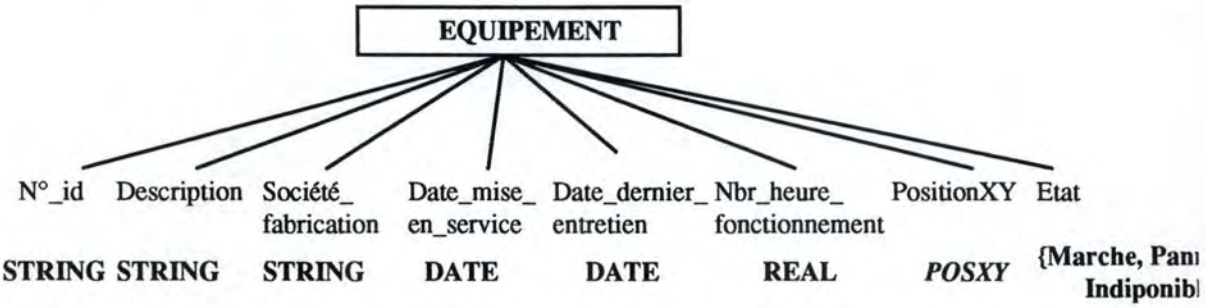
2.34. ENDROIT

Union [***ID_E/S***, ***ID_PLACE***]



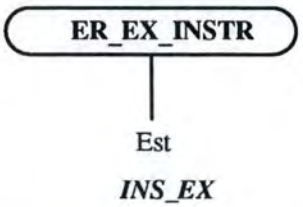
2.35. EQUIPEMENT

CP [N°_id: **STRING**,
Description: **STRING**,
Société_fabrication: **STRING**,
Date_mise_en_service: **DATE**,
Date_dernier_entretien: **DATE**,
Nbr_heure_fonctionnement: **REAL**,
PositionXY: ***POSXY***,
Etat: **{Marche, Panne, Indisponible}**]



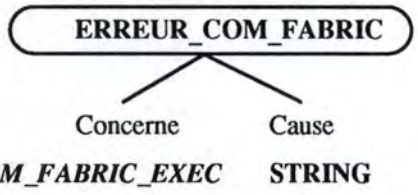
2.36. ER_EX_INSTR

CP [Est: ***INS_EX***]



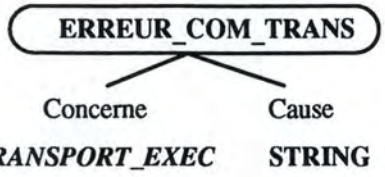
2.37. ERREUR_COM_FABRIC

CP [Concerne: ***COM_FABRIC_EXEC***,
Cause: **STRING**]



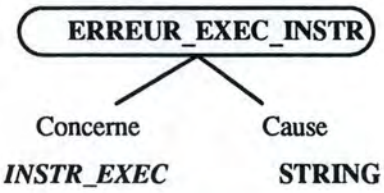
2.38. ERREUR_COM_TRANS

CP [Concerne : ***COM_TRANSPORT_EXEC***,
Cause : **STRING**]



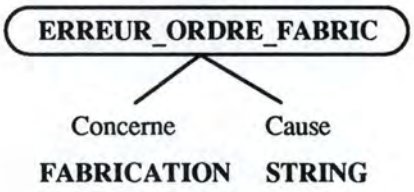
2.39. ERREUR_EXEC_INSTR

CP [Concerne: ***INSTR_EXEC***,
Cause: **STRING**]



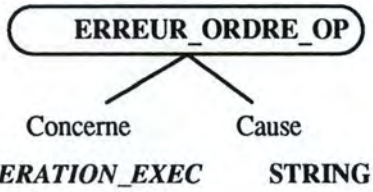
2.40. ERREUR_ORDRE_FABRIC

CP [Concerne: ***FABRIC_EXEC***,
Cause: **STRING**]



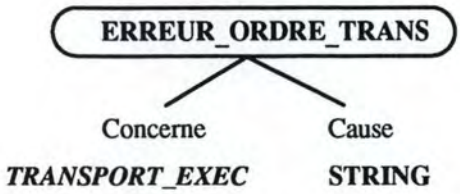
2.41. ERREUR_ORDRE_OP

CP [Concerne : ***OPERATION_EXEC***,
Cause : **STRING**]



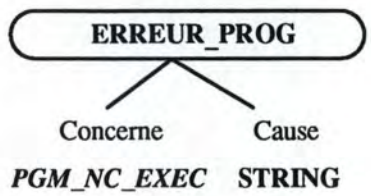
2.42. ERREUR_ORDRE_TRANS

CP [Concerne: ***TRANSPORT_EXEC***,
Cause: **STRING**]



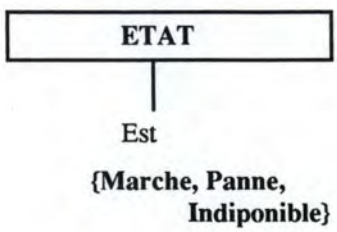
2.43. ERREUR_PROG

CP [Concerne : ***PGM_NC_EXEC***,
Cause: **STRING**]



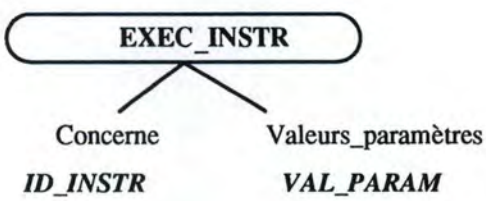
2.44. ETAT

CP [Est: **{Marche, Panne, Indisponible}**]



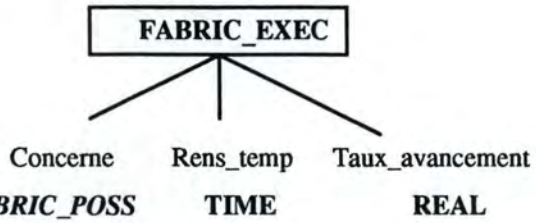
2.45. EXEC_INSTR

CP [Concerne: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***]



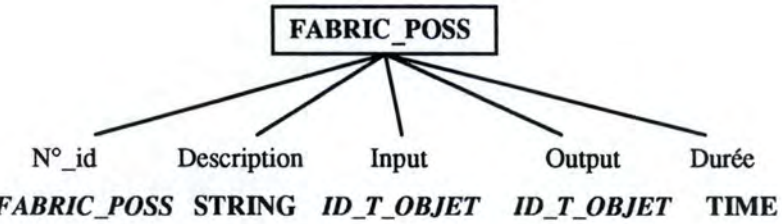
2.46. FABRIC_EXEC

CP [Concerne: ***ID_FABRIC_POSS***,
Rens_Temp: **TIME**,
Taux_avancement: **REAL**]



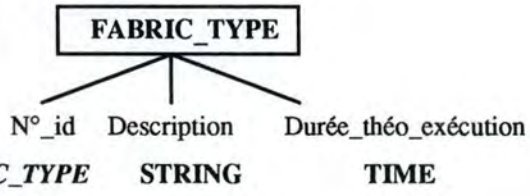
2.47. FABRIC_POSS

CP [N°_id: ***ID_FABRIC_POSS***, {à déterminer}
Description: **STRING**,
Input: ***ID_T_OBJET***,
Output: ***ID_T_OBJET***,
Durée: **TIME**]



2.48. FABRIC_TYPE

CP [N°_id: ***ID_FABRIC_TYPE***, {à déterminer}
Description: **STRING**,
Durée_théo_exécution: **TIME**]



2.49. FABRICATION

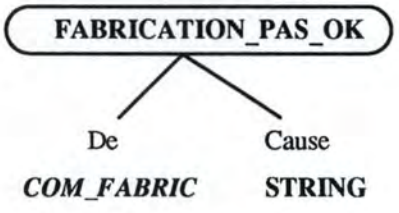
CP [Concerne: ***ID_FABRIC_POSS***]

2.50. FABRICATION_OK

CP [Est: ***COM_FABRIC***]

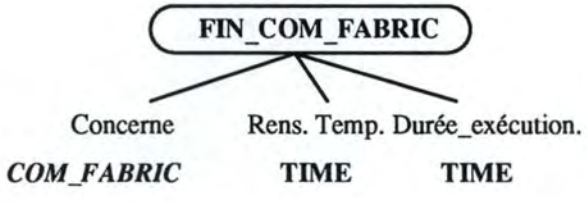
2.51. FABRICATION_PAS_OK

CP [De: ***COM_FABRIC***,
Cause: **STRING**]



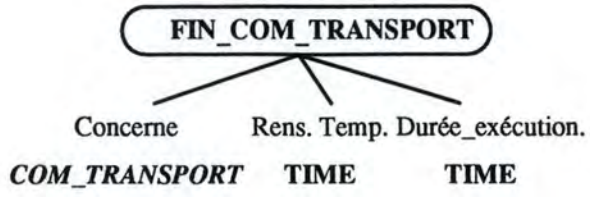
2.52. FIN_COM_FABRIC

CP [Concerne: ***COM_FABRIC***,
Rens_Temp: **TIME**,
Durée_exécution: **TIME**]



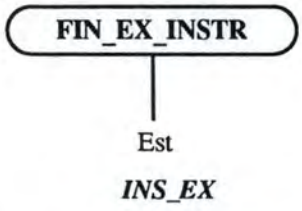
2.53. FIN_COM_TRANSPORT

CP [Concerne: ***COM_TRANSPORT***,
Rens_Temp: **TIME**,
Durée_exécution: **TIME**]



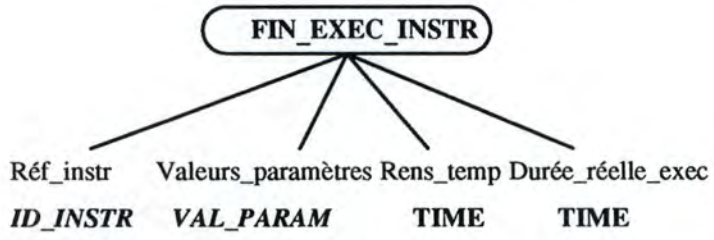
2.54. FIN_EX_INSTR

CP [Est: ***INS_EX***]



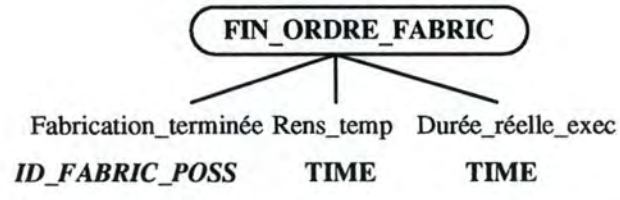
2.55. FIN_EXEC_INSTR

CP [Concerne: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***,
Rens_temp: **TIME**,
Durée_réelle_exec: **TIME**]



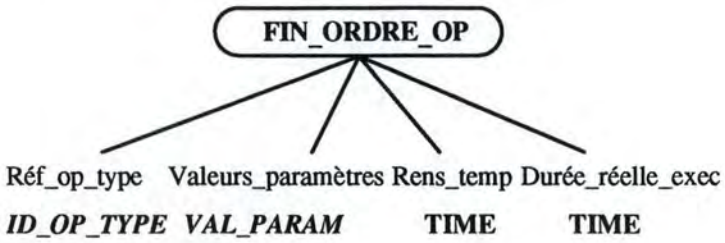
2.56. FIN_ORDRE_FABRIC

CP [Fabrication_terminée: ***ID_FABRIC_POSS***,
Rens_temp: **TIME**,
Durée_réelle_exec: **TIME**]



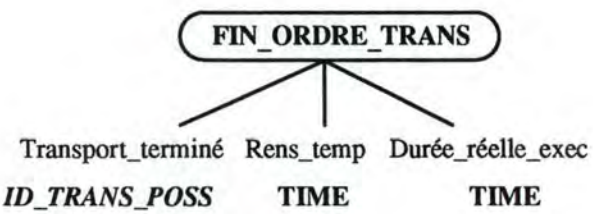
2.57. FIN_ORDRE_OP

CP [Concerne: ***ID_OP_TYPE***,
Valeurs_paramètres: ***VAL_PARAM***,
Rens_temp: **TIME**,
Durée_réelle_exec: **TIME**]



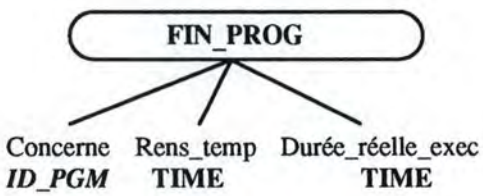
2.58. FIN_ORDRE_TRANS

CP [Transport_terminé: ***ID_TRANS_POSS***,
Rens_temp: **TIME**,
Durée_réelle_exec: **TIME**]



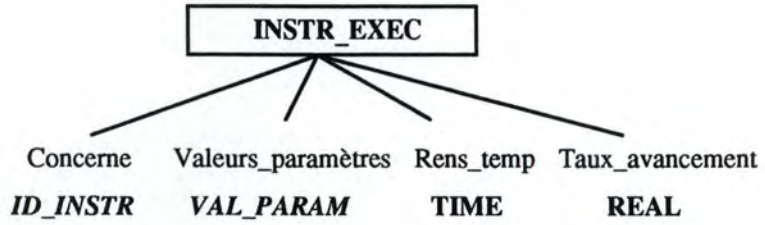
2.59. FIN_PROG

CP [Concerne : ***ID_PGM***,
Rens_temp: **TIME**,
Durée_réelle_exec: **TIME**]



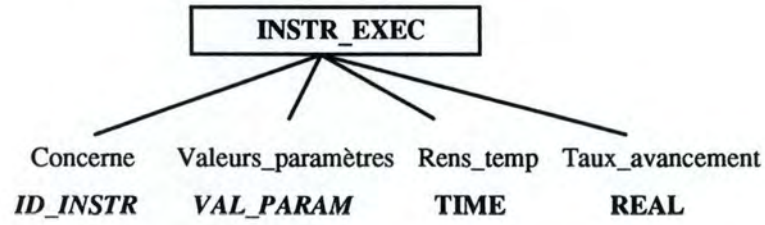
2.60. INS_EX

CP [Concerne: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***]



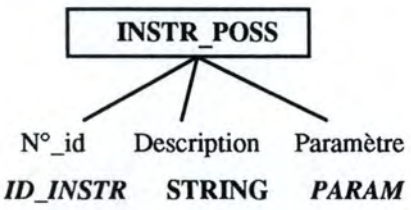
2.61. INSTR_EXEC

CP [Concerne: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***,
Rens_temp: **TIME**,
Taux_avancement: **REAL**]



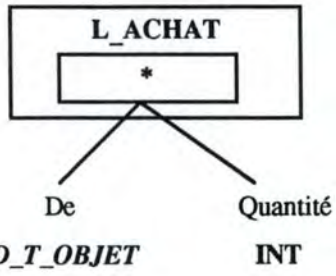
2.62. INSTR_POSS

CP [N°_id: ***ID_INSTR***, {à déterminer}
Description: **STRING**,
Paramètre: ***PARAM***]



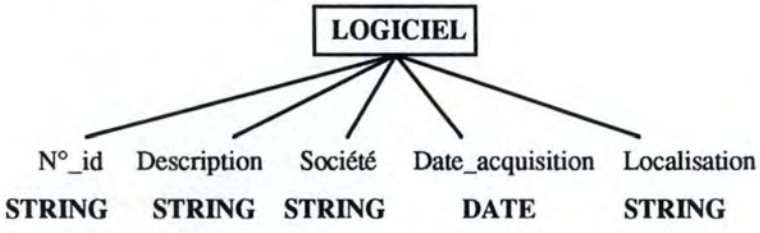
2.63. L_ACHAT

SET [CP[De : ***ID_T_OBJET***
Quantité : **INT**]]



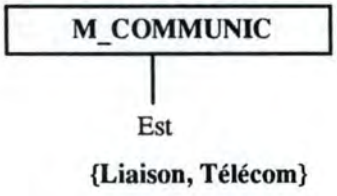
2.64. LOGICIEL

CP [N°_id: **STRING**,
Description: **STRING**,
Société: **STRING**,
Date_acquisition: **DATE**,
Localisation: **STRING**]



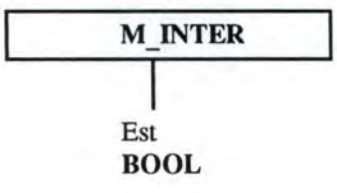
2.65. M_COMMUNIC

CP [Est: **{Liaison, Télécom}**]



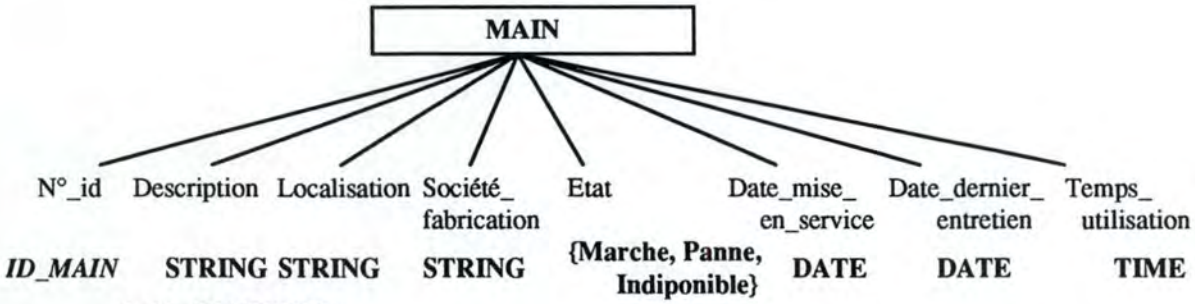
2.66. M_INTER

CP [Est: **BOOL**]



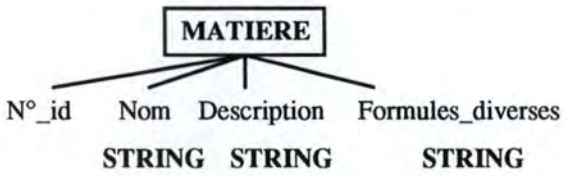
2.67. MAIN

CP [N°_id: ***ID_MAIN***, {à déterminer}
Description: **STRING**,
Localisation: **STRING**,
Société_fabrication: **STRING**,
Statut: **{Panne, Marche, Indisponible}**,
Date_mise_en_service: **DATE**,
Date_dernier_entretien: **DATE**,
Temps_utilisation: **TIME**]



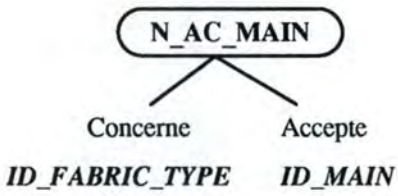
2.68. MATIERE

CP [N°_id : , {à déterminer}
Nom : **STRING**,
Description : **STRING**,
Formules_diverses : **STRING**]



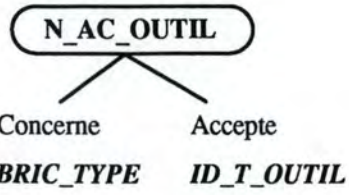
2.69. N_AC_MAIN

CP [Concerne: ***ID_FABRIC_TYPE***,
Accepte: ***ID_MAIN***]



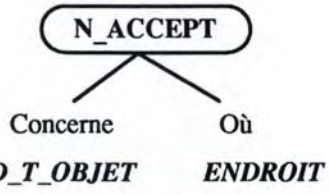
2.70. N_AC_OUTIL

CP [Concerne: ***ID_FABRIC_TYPE***,
Accepte: ***ID_T_OUTIL***]



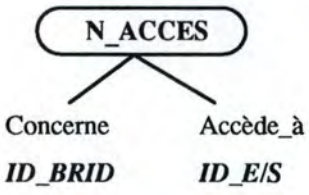
2.71. N_ACCEPT

CP [Concerne: ***ID_T_OBJET***,
Où: ***ENDROIT***]



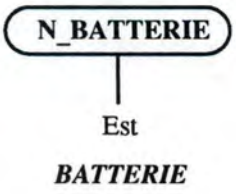
2.72. N_ACCES

CP [Concerne: ***ID_BRID***,
Accède_à: ***ID_E/S***]



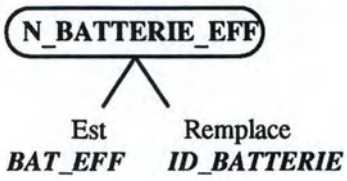
2.73. N_BATTERIE

CP [Est: ***BATTERIE***]



2.74. N_BATTERIE_EFF

CP [Est: ***BAT_EFF***,
Remplace: ***ID_BATTERIE***]

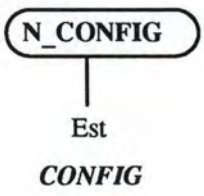


2.75. N_BRID

CP [Est: ***BRID***]

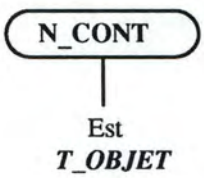
2.76. N_CONFIG

CP [Est: ***CONFIG***]



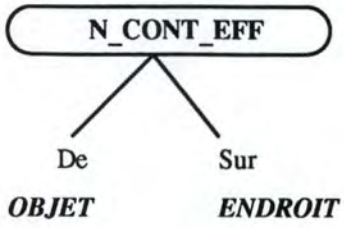
2.77. N_CONT

CP [Est: ***T_OBJET***]



2.78. N_CONT_EFF

CP [De : ***OBJET***,
Sur : ***ENDROIT***]

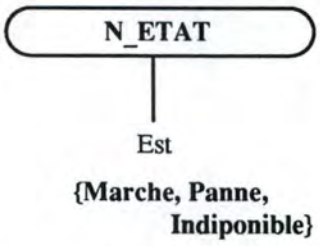


2.79. N_DATE

CP [Est: **DATE**]

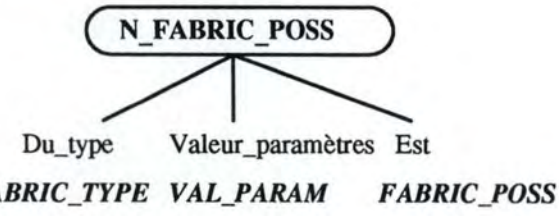
2.80. N_ETAT

CP [Est: **{Marche, Panne, Indisponible}**]



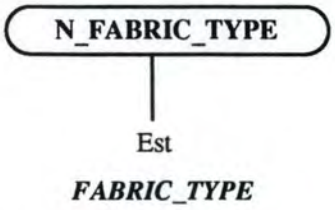
2.81. N_FABRIC_POSS

CP [Concerne : ***ID_FABRIC_TYPE***,
Valeur_paramètres : ***VAL_PARAM*** ,
Est : ***FABRIC_POSS***]



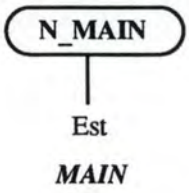
2.82. N_FABRIC_TYPE

CP [Est : ***FABRIC_TYPE***]



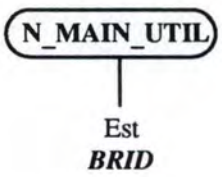
2.83. N_MAIN

CP [Est: ***MAIN***]



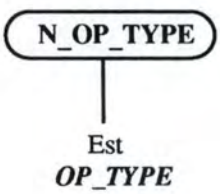
2.84. N_MAIN_UTIL

CP [Est: ***MAIN***]



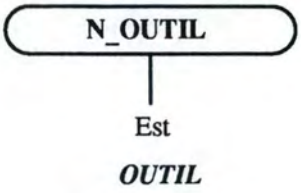
2.85. N_OP_TYPE

CP [Est : ***OP_TYPE***]



2.86. N_OUTIL

CP [Est: ***OUTIL***]

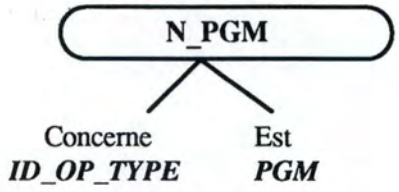


2.87. N_PERIODE

CP [Est: ***PERIODE***]

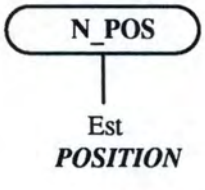
2.88. N_PGM

CP [Concerne : ***ID_OP_TYPE***,
Est : ***PGM***]



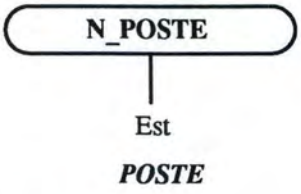
2.89. N_POS

CP [Est: ***POSITION***]



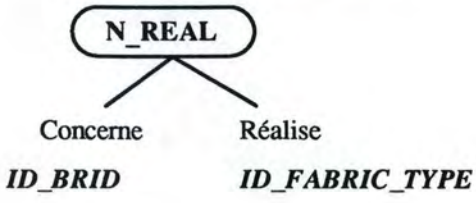
2.90. N_POSTE

CP [Est: ***POSTE***]



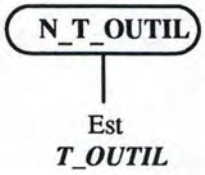
2.91. N_REAL

CP [Concerne: ***ID_BRID***,
Réalise: ***ID_FABRIC_TYPE***]



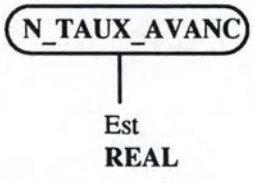
2.92. N_T_OUTIL

CP [Est: ***T_OUTIL***]



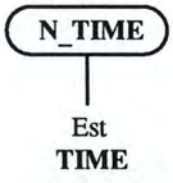
2.93. N_TAUX_AVANC

CP [Est: **REAL**]



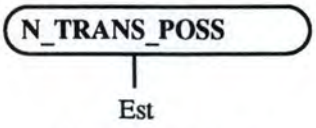
2.94. N_TIME

CP [Est: **TIME**]



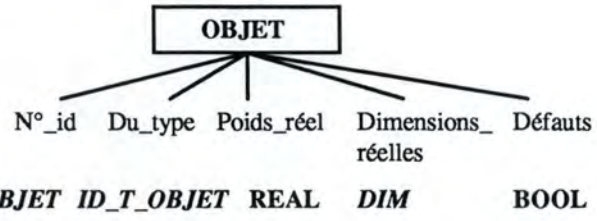
2.95. N_TRANS_POSS

CP [Est : ***TRANSPORT_POSS***]



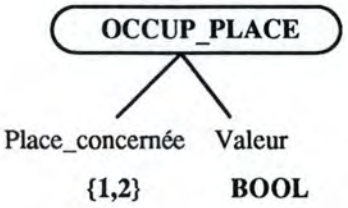
2.96. OBJET

CP [N°_id: ***ID_OBJET***, {à déterminer}
Du_type : ***ID_T_OBJET***,
Poids_réel : **REAL**,
Dimensions_réelles : ***DIM***, {à déterminer}
Défauts : **BOOL**]



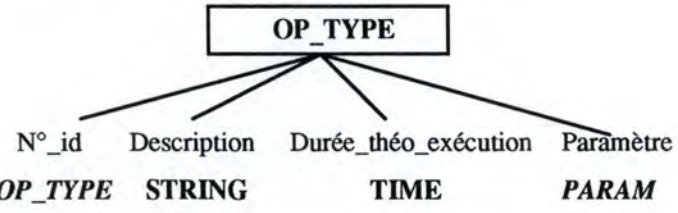
2.97. OCCUP_PLACE

CP [Place_concernée: **{1,2}**,
Valeur: **BOOL**]



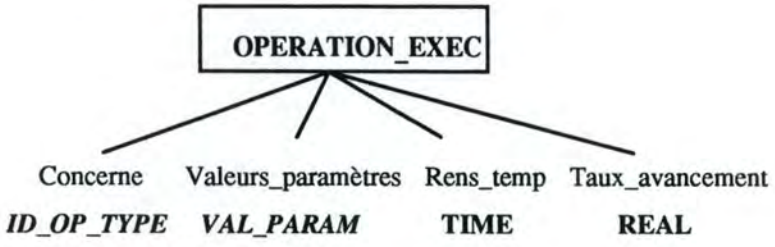
2.98. OP_TYPE

CP [N°_id: ***ID_OP_TYPE***, {à déterminer}
Description: **STRING**,
Durée_théo_exécution: **TIME**,
Paramètre: ***PARAM***]



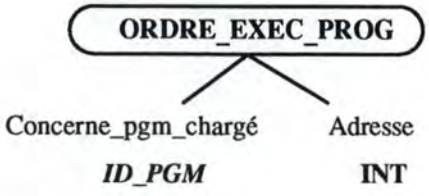
2.99. OPERATION_EXEC

CP [Concerne: ***ID_OP_TYPE***,
Valeurs_paramètres: ***VAL_PARAM***,
Rens_temp: **TIME**,
Taux_avancement: **REAL**]



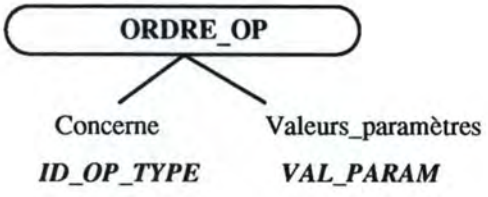
2.100. ORDRE_EXEC_PROG

CP [Concerne_pgm_chargé : ***ID_PGM***,
Adresse: **INT**]



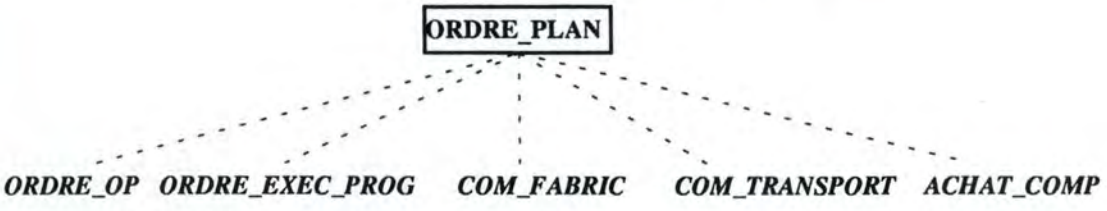
2.101. ORDRE_OP

CP [Concerne: ***ID_OP_TYPE***,
Valeurs_paramètres: ***VAL_PARAM***]



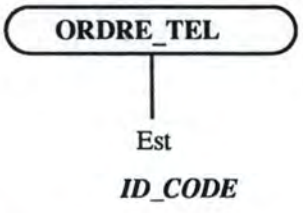
2.102. ORDRE_PLAN

Union [*ORDRE_OP*, *ORDRE_EXEC_PROG*, *COM_FABRIC*,
COM_TRANSPORT, *ACHAT_COMP*]



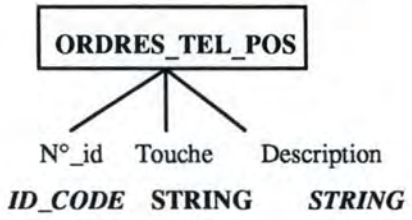
2.103. ORDRE_TEL

CP [Est: *ID_CODE*]



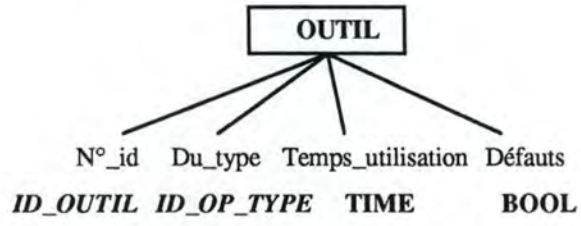
2.104. ORDRES_TEL_POS

CP [N°_id: *ID_CODE*, {à déterminer}
Touche: **STRING**,
Description: **STRING**]



2.105. OUTIL

CP [N°_id: *ID_OUTIL*, {à déterminer}
Du_type: *ID_OP_TYPE*,
Temps_utilisation: **TIME**,
Défauts: **BOOL**]

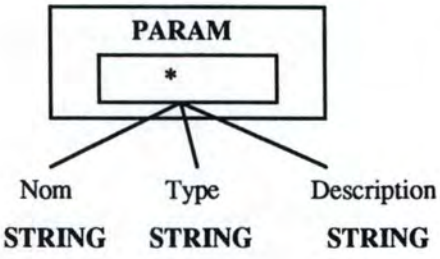


2.106. P_PISTE

{à déterminer}

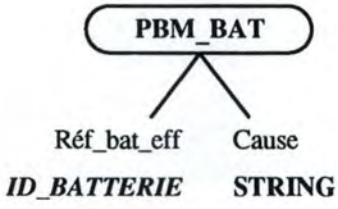
2.107. PARAM

SET[CP [Nom: **STRING**,
Type: **STRING**,
Description: **STRING**]]



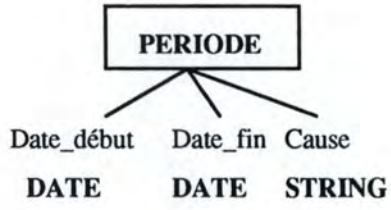
2.108. PBM_BAT

CP [Réf_bat_eff: ***ID_BATTERIE***,
Cause: **STRING**]



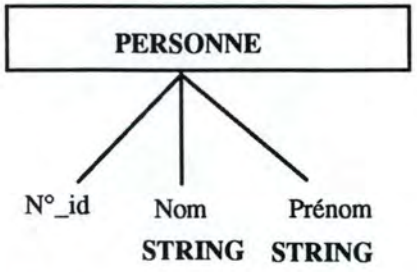
2.109. PERIODE

CP [Date_début : **DATE**,
Date_fin : **DATE**,
Cause : **STRING**]



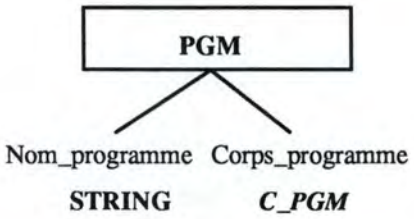
2.110. PERSONNE

CP [N°_id : , {à déterminer}
Nom : **STRING**,
Prénom : **STRING**]



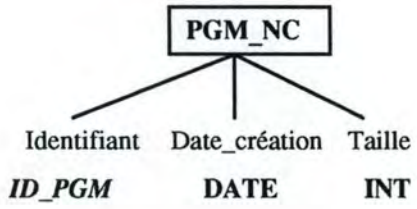
2.111. PGM

CP [Nom_programme: **STRING**,
Corps_programme: ***C_PGM***]



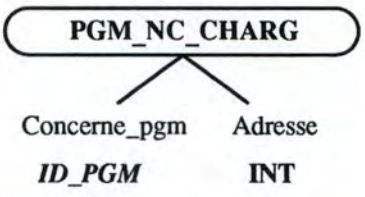
2.112. PGM_NC

CP [Identifiant : ***ID_PGM***, {à déterminer}
Date_création : **DATE**,
Taille: **INT**]



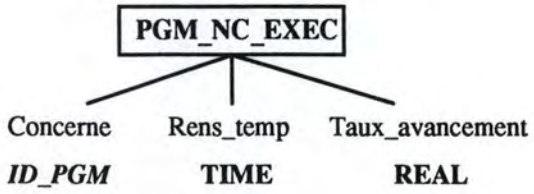
2.113. PGM_NC_CHARG

CP [Concerne_pgm: ***ID_PGM***,
Adresse: **INT**]



2.114. PGM_NC_EXEC

CP [Concerne: ***ID_PGM***,
Rens_temp: **TIME**,
Taux_avancement: **REAL**]

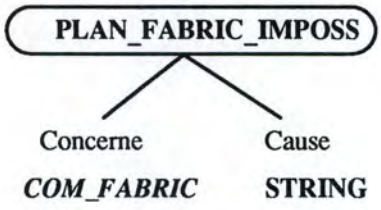


2.115. PLACE

CP [N°_id : ***ID_PLACE***, {à déterminer}
Dim_max : ***DIM***, {à déterminer}
Unité_dim : ***UNITE***, {à déterminer}
Poids_max : **REAL**,
Unité_poids: ***UNITE*** {à déterminer}]

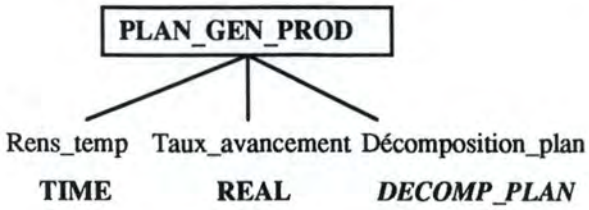
2.116. PLAN_FABRIC_IMPOSS

CP [Concerne: ***COM_FABRIC***,
Cause: **STRING**]



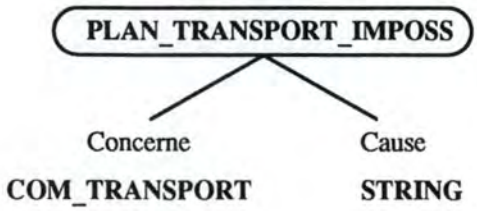
2.117. PLAN_GEN_PROD

CP [Rens_temp: **TIME**,
Taux_avancement: **REAL**,
Décomposition_plan: ***DECOMP_PLAN***]



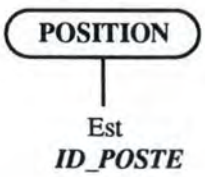
2.118. PLAN_TRANSPORT_IMPOSS

CP [Concerne: ***COM_TRANSPORT***,
Cause: **STRING**]



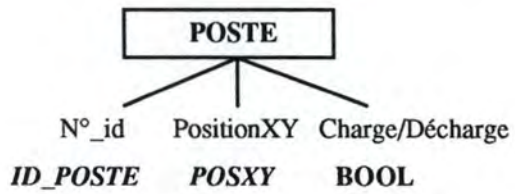
2.119. POSITION

CP [Est: ***ID_POSTE***]



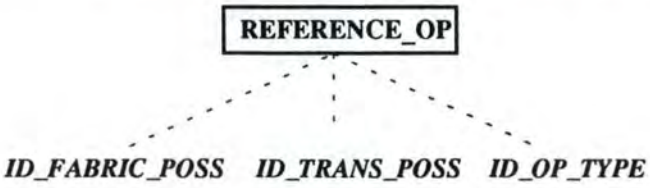
2.120. POSTE

CP [N°_id : ***ID_POSTE***, {à déterminer}
PositionXY: ***POSXY***,
Charge/Décharge: **BOOL**]



2.121. REFERENCE_OP

Union [***ID_FABRIC_POSS***, ***ID_TRANS_POSS***, ***ID_OP_TYPE***]

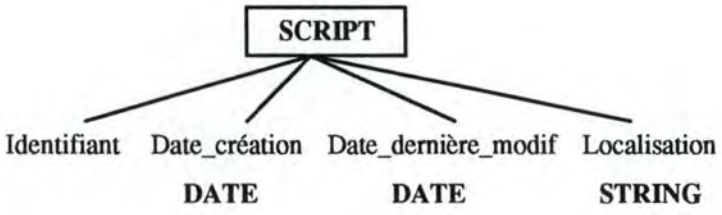


2.122. REORG_MEM

{à déterminer}

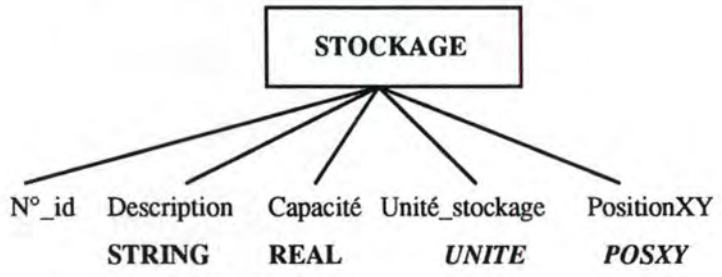
2.123. SCRIPT

CP [Identifiant : , {à déterminer}
Date_création : **DATE**
Date_dernière_modif: **DATE**
Localisation: **STRING**]



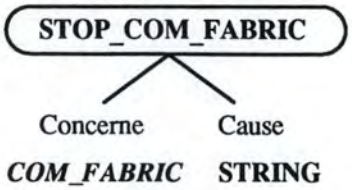
2.124. STOCKAGE

CP [N°_id: , {à déterminer}
Description: **STRING**,
Capacité: **REAL**,
Unité_stockage : ***UNITE***, {à déterminer}
PositionXY : ***POSXY*** {à déterminer}]



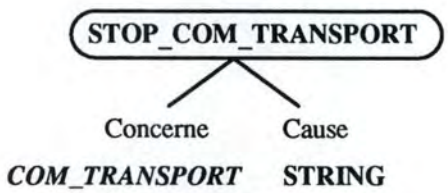
2.125. STOP_COM_FABRIC

CP [Concerne: ***COM_FABRIC***,
Cause: **STRING**]



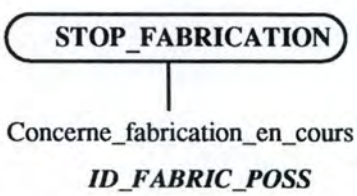
2.126. STOP_COM_TRANSPORT

CP [Concerne: ***COM_TRANSPORT***,
Cause : **STRING**]



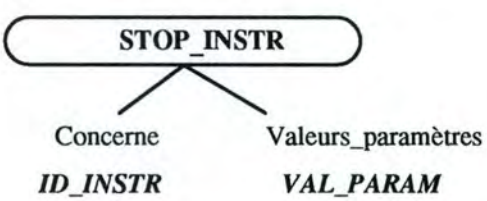
2.127. STOP_FABRICATION

CP [Concerne_fabrication_en_cours: ***ID_FABRIC_POSS***]



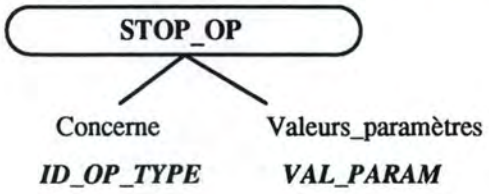
2.128. STOP_INSTR

CP [Concerne: ***ID_INSTR***,
Valeurs_paramètres: ***VAL_PARAM***]



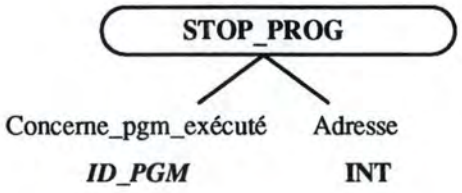
2.129. STOP_OP

CP [Concerne: ***ID_OP_TYPE***,
Valeurs_paramètres: ***VAL_PARAM***]



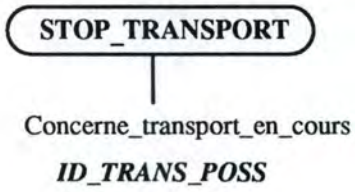
2.130. STOP_PROG

CP [Concerne_pgm_exécuté : ***ID_PGM***,
Adresse: **INT**]



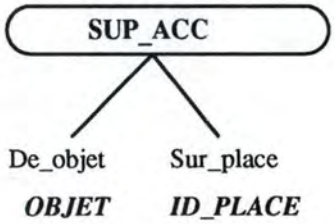
2.131. STOP_TRANSPORT

CP [Concerne_transport_en_cours: ***ID_TRANS_POSS***]



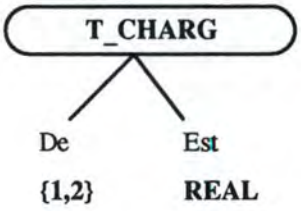
2.132. SUP_ACC

CP [De_objet: ***OBJET***,
Sur_place: ***ID_PLACE***]



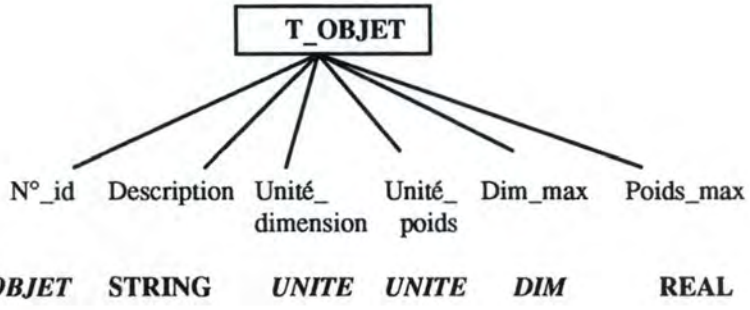
2.133. T_CHARG

CP [De: **{1,2}**,
est: **REAL**]



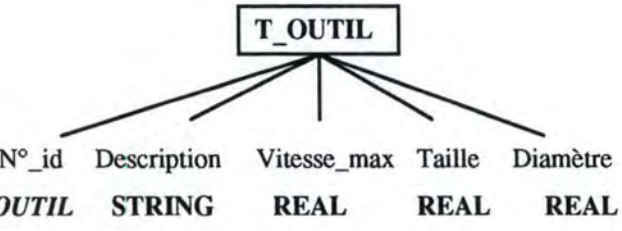
2.134. T_OBJET

CP [N°_id : ***ID_T_OBJET***, {à déterminer}
Description : **STRING**,
Unité_dim: ***UNITE***, {à déterminer}
Unité_poids : ***UNITE***, {à déterminer}
Dim_max : ***DIM***, {à déterminer}
Poids_max : **REAL**]



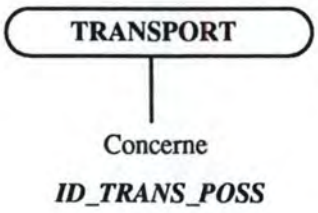
2.135. T_OUTIL

CP [N°_id: ***ID_T_OUTIL***, {à déterminer}
Description : **STRING**,
Vitesse_max : **REAL**,
Taille: **INT**,
Diamètre: **INT**]



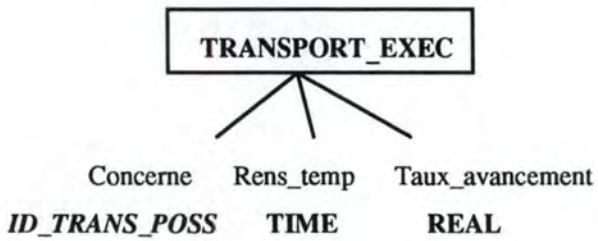
2.136. TRANSPORT

CP [Concerne: ***ID_TRANS_POSS***]



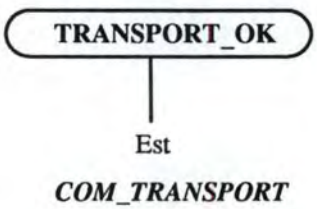
2.137. TRANSPORT_EXEC

CP [Concerne: ***ID_TRANS_POSS***,
Rens_Temp: **TIME**,
Taux_avancement: **REAL**]



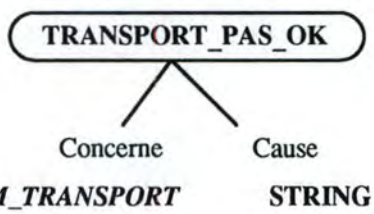
2.138. TRANSPORT_OK

CP [Est: ***COM_TRANSPORT***]



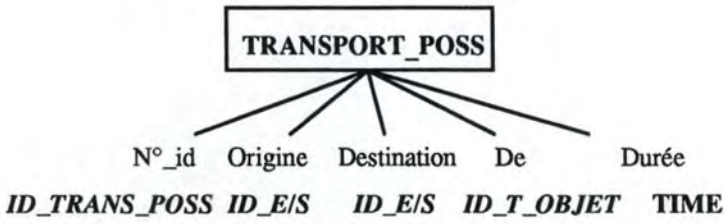
2.139. TRANSPORT_PAS_OK

CP [Concerne: ***COM_TRANSPORT***,
Cause: **STRING**]



2.140. TRANSPORT_POSS

CP [N°_id : ***ID_TRANS_POSS***, {à déterminer}
Origine : ***ID_E/S***,
Destination : ***ID_E/S***,
De : ***ID_T_OBJET***,
Durée : **TIME**]

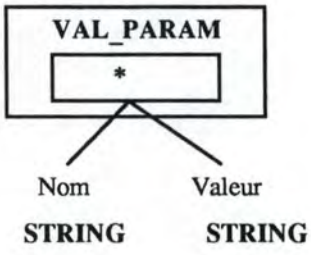


2.141. URGENCE

{à déterminer}

2.142. VAL_PARAM

SET[CP [Nom: **STRING**,
Valeur: **STRING**]]



2.143. VEHICULE

CP [N°_id: **STRING**,
Description: **STRING**,
Société_fabrication: **STRING**,
Date_mise_en_service: **DATE**,
Date_dernier_entretien: **DATE**,
Nbr_heure_fonctionnement: **REAL**,
Etat: **{Marche, Panne, Indisponible}**]

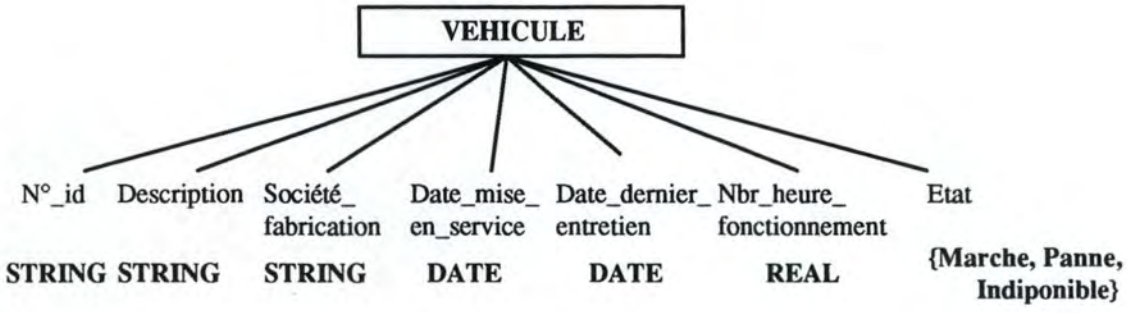


Table des matières de l'annexe C