FACULTE DES SCIENCES

UFR Sciences Techniques Mathématiques Informatique Automatique
Ecole Doctorale IAE + M
DFD Automatique et Production Automatisée.

Thèse

présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy I

Spécialité Informatique Industrielle

par

Ikbal ARAB - MANSOUR

Conception et intégration des communications des systèmes automatisés distribués

Soutenue publiquement le 14 Décembre 1998 devant la Commission d'Examen composée de :

Membres du jury :

Président :	M. A. BERNARD	Professeur à l'Université Henri Poincaré-Nancy I
Rapporteurs :	M. J.P. BOUREY	Professeur à l'Ecole Centrale de Lille
	M. C.F. DUCATEAU	Professeur à l'Université Paris V
Examinateurs :	M. M. WACK	Maître de Conférences à l'Institut Polytechnique de Sévenans
	M. F. LEPAGE	Professeur à l'Université Henri Poincaré-Nancy I
	M. E. RONDEAU	Maître de Conférences à l'Université Henri Poincaré-Nancy I

Centre de Recherche en Automatique de Nancy
Faculté des Sciences - 54500 VANDOEUVRE-lès-NANCY

A
Mes parents,
Mon mari Anas,
Mes enfants Roujin, Siwar et Zavine,
Tous ceux que j'aime.

Remerciements

Le travail de recherche présenté dans ce mémoire a été réalisé au Centre de Recherche en Automatique de Nancy (C.R.A.N. – E.S.A. 7039) au sein de l'équipe Réseaux Locaux Industriels dirigée par le professeur Francis LEPAGE que je tiens à remercier pour m'avoir accueilli dans son laboratoire.

Je tiens à remercier l'Université d'Alep de m'avoir assuré le temps nécessaire et les moyens financiers pour la réalisation de ce travail.

Je tiens à adresser mes plus vifs remerciements aux personnes qui me font l'honneur de composer ce jury :

Monsieur le professeur Jean Pierre Bourey, de l'Ecole Centrale de Lille au Laboratoire d'Automatique et d'Informatique Industrielle, et à Monsieur Charles François DUCATEAU, Professeur à l'IUT de l'Université René Déscartes (Paris 5) et au laboratoire LIASI pour avoir accepté la charge d'évaluer ce travail en qualité de rapporteur.

Monsieur Alain BERNARD, Professeur à l'Université Henri Poincaré Nancy I, responsable de l'équipe Ingénierie de la Conception et de la Fabrication du Groupe Génie de la Production et Monsieur Maxime WACK, Maître de Conférences à l'Institut Polytechnique de Sévenans pour l'intérêt qu'ils ont porté à ce travail en tant qu'examinateurs.

Monsieur Francis LEPAGE, Professeur à l'Université Henri Poincaré Nancy I, Directeur du C.R.A.N, comme directeur de thèse pour m'avoir apporté les conditions nécessaires pour accomplir ces travaux, et à Monsieur Eric RONDEAU, Maître de Conférences à l'Université Henri Poincaré Nancy I, pour sa grande disponibilité, son encadrement et sa grande dévotion qu'il a mise dans ce travail.

Je tiens à remercier également Monsieur Thierry DIVOUX, Professeur à l'Université Henri Poincaré Nancy I et chef du Département de Génie de la Télécommunication et Réseaux à l'IUT Nancy-Brabois, pour ses précieux conseils et l'intérêt qu'il a porté à ce travail.

Je remercie mes parents et plus particulièrement mon père qui m'a encouragé à suivre mes études supérieures et pour qui ce travail représente un rêve réalisé.

Je remercie de tout coeur mon compagnon à vie, mon mari, grâce à qui ces années d'études sont bien passées, et mes enfants qui ont supporté ces années de galère et de travail intense.

Je remercie mes collègues du laboratoire et plus particulièrement Corinne, Lionel, Nicolas, Husseine, Olivier, Cédric, Mario, Raid, Eric, Evelyne, Jean Baptiste, Jean Christophe, Patrick, pour ces merveilleux moments passés ensemble, et l'ensemble du personnel pour l'humour et la gentillesse qu'ils ont approuvé.

Je remercie vivement mes amis Pascal, Hala, Lamia, Fairouze, Rim, Alaa.

Table des Matière

INTRODUCTION

CHAPITRE 1

MODELES D'INTEGRATION DES SYSTEMES AUTOMATISES DISTRIBUES

CHAPITRE 2

DU SYSTEME D'INFORMATION AU SYSTEME DE COMMUNICATION PAR UNE FORMALISATION UNIFIEE

CHAPITRE 3

INGENIERIE DE LA METHODE

CHAPITRE 4

APPLICATION DE TELE-PILOTAGE D'UN ROBOT INDUSTRIEL

ANNEXE

ELABORATION D'UNE NORME D'ACCOMPAGNEMENT (EXEMPLE ROBOT)

Introduction

1. Cadre de travail

L'intégration des diverses fonctions de l'entreprise au moyen des technologies de l'information est un facteur fondamental de maîtrise des systèmes de production (interopérabilité des fonctions administratives, commerciales, de gestion de production, et de production proprement dite). Par conséquent, le système d'information est considéré comme étant le facteur d'intégration des fonctions et services d'une entreprise intégrée.

Les échanges, effectués au sein du système de production, en terme de flux d'information sont supportés par le système de communication sous-jacent. Donc, l'automatisation est conjointe à l'intégration et est réalisée via les mécanismes offerts par le système de communication. D'où l'idée que la configuration des communications entre entités d'un système automatisé de production pourrait découler de la conception de son système d'information.

Ces constatations forment la base de notre travail. Ainsi notre objectif est la conception et l'intégration des communications industrielles des systèmes automatisés distribués. Se basant sur l'étude du système d'information et sans avoir recours à de nouveaux outils de modélisation, nous voulons générer la configuration des matériels vis-à-vis du système de communication tout en garantissant la cohérence entre ces deux systèmes. Par ce

fait, nous mettons à disposition du concepteur du système de production les moyens lui sont nécessaires pour la réalisation de son système. Alors, notre objectif est de donner au concepteur le prolongement nécessaire à l'implantation de son système.

2. Organisation du mémoire

Ce mémoire est organisé en quatre chapitres :

Dans le **chapitre 1**, nous effectuons un survol des modèles d'intégration des communications dans les systèmes automatisés distribués. Ceci en exposant les travaux réalisés dans le domaine de l'intégration et du concept CIM (Computer Integrated Manufacturing), en terme de modèles d'entreprise intégrée et de plates-formes d'intégration. Une présentation de la messagerie industrielle MMS (Manufacturing Message Specification) avec ses apports et ses avantages en terme d'intégration sera faite. Etant donné que nous nous basons dans notre démarche le système d'information, une présentation des méthodes de conception et leurs évolutions sera également faite. Alors, nous exposons notre problématique ainsi que notre contribution et la démarche que nous adoptons pour sa mise en œuvre.

Dans le **chapitre 2**, nous expliquons les étapes suivies pour la génération du système de communication (les VMDs et leurs structures internes en terme d'objets et services MMS) à partir de l'étude du système d'information de l'application étudiée. D'abord, nous procédons à une formalisation d'un ensemble des concepts génériques (classe d'objets, objet, relations, ...) d'OMT. Ensuite, nous formalisons également les concepts génériques (classe d'objets, objet, relations, ...) du modèle MMS.

Afin d'établir le passage entre ces deux représentations (d'OMT à MMS), nous définissons des règles appelées règles de transformation. Ces dernières mettent en correspondance les concepts OMT et MMS avec le minimum de pertes sémantiques.

Dans **le chapitre 3**, Dans une optique d'ingénierie de la méthode, nous nous préoccupons du contexte d'implémentation en prenant en compte les contraintes relatives aux classes de conformité des composants de l'application considérée. Pour déterminer la structure interne des VMDs, nous définissons des conventions avec lesquelles nous pouvons désigner les objets constituant sa structure.

Afin de déterminer la nature des services MMS générés, nous nous basons sur les aspects dynamiques dans les deux formalismes OMT et MMS et nous définissons ainsi des conventions appropriées. Quelques fonctionnalités des systèmes automatisés distribués, d'un point de vue communication, exprimées implicitement à travers certains objets et services MMS seront dégagées et seront représentées par des graphes d'états transitions.

Jusqu'ici nous considérions que les objets et services MMS générés seront tous supportés par les VMDs que nous avions définis. En prenant en considération les objets et services MMS réellement implantés par chaque composant de l'application, nous aurons recours à une adaptation des conventions déjà avancées.

Dans **le chapitre 4**, nous validons nos travaux et la démarche associée à l'aide d'une étude de cas d'une application de télé-pilotage d'un robot industriel avec retour vidéo effectué par deux caméras.

Nous terminons ce mémoire par une conclusion générale incluant les perspectives de recherches.

Modèles d'intégration des communications des systèmes automatisés distribués

1. L'intégration et le concept CIM

L'intégration de l'entreprise est stratégique autant que technologique. Elle doit conduire à une réactivité temps-réel de l'entreprise vis-à-vis des besoins de ses clients, aux sens fonctionnel (Fabbricino, 1994), qualité (Bajic, 1995), esthétique (Fabbricino, 1994), environnement, ...etc.

C'est cette perception de l'intégration qu'il faut entendre dans le fameux concept CIM (Computerized Integrated Manufacturing). Par ce fait, l'intégration d'un système de production revient à la conception d'un système global gérant les flux d'information, de contrôle et de la matière première. Par conséquent, Celui-ci a donné lieu ces dernières années à de nombreux travaux qui ont conduit à la définition d'une multitude de modèles d'entreprise intégrée que nous présentons maintenant en détail.

1.1. Modèles d'entreprise intégrée

L'intégration de l'entreprise ne relève pas de l'automatisation ou du développement d'applications informatiques, qui n'en sont que les moyens. L'intégration, c'est la collecte, l'échange, le traitement, le stockage, l'utilisation des données. Une entreprise intégrée coordonne sa stratégie, sa tactique, sa flexibilité, sa réactivité, grâce à un flux d'informations

efficace et une organisation qui gère celles-ci de manière optimale (Williams, 1994). Aussi parle-t-on d'intégration par le système d'information.

L'intégration des diverses fonctions de l'entreprise au moyen des technologies de l'information est un facteur fondamental de maîtrise des systèmes de production (interopérabilité des fonctions administratives, commerciales, de gestion de production, et de production proprement dite). Pour cela, il faut disposer de modèles généralisés et d'une architecture ouverte de système permettant d'identifier et de représenter les principaux composants, processus et activités, ainsi que les principales sources d'informations, de décisions et de contraintes nécessaires au fonctionnement de l'entreprise (Kosanke, 1990).

Au regard de la complexité de la tâche, il est nécessaire de se doter d'une méthodologie d'intégration. Elle définit l'ensemble des règles, techniques et procédures à suivre pour aboutir. Elle doit spécifier ou suggérer des outils pour concevoir, développer et implémenter. A chacun de ces trois niveaux, il faut pouvoir créer, développer et analyser des modèles. A titre d'exemple ou d'illustration, plusieurs modèles vont être cités ici. Ils ne seront néanmoins pas décrits. Le lecteur trouvera des renvois bibliographiques les présentant. CIM-OSA (Computerized Integrated Manufacturing - Open System Architecture) (Russel, 1991) sera une référence privilégiée; sans doute parce qu'elle est la plus formellement décrite, mais aussi parce qu'elle est pressentie comme la base d'une future normalisation.

Les projets d'intégration de toutes les entreprises, aussi différentes soient elles, présentent un certain nombre d'invariants. Aussi ont été développées des architectures de référence, normalisées ou en voie de l'être, permettant à tout projet d'intégration de démarrer sur des bases solides, éprouvées et cohérentes. Une architecture CIM peut être définie comme un ensemble structuré des modèles qui représentent les divers aspects de ce système dans son intégralité. Cette architecture peut être considérée comme une base pour la conception et l'implantation du système (Chen, 1990).

Un ensemble d'activités relevant d'une même préoccupation et réclamant un même ensemble d'information définit un domaine homogène dont la stabilité dans le temps est forte. Le modèle de référence de l'entreprise est constitué de ces domaines et de leurs liaisons. L'étude globale peut alors être morcelée, sans perte de la cohérence de l'ensemble (Foulard, 1994).

Il est alors envisageable de particulariser plus ou moins progressivement ces architectures, afin qu'elles s'appliquent et répondent plus largement aux besoins de familles d'entreprises d'un même secteur d'activité, jusqu'aux besoins précis d'une entreprise particulière. L'intérêt de cette démarche réside bien sûr dans les possibilités de réutilisabilité de solutions génériques. C'est typiquement l'objet de l'axe "instanciation" du modèle CIM-OSA (Gaches, 1990).

Simultanément, chaque architecture se décline selon plusieurs niveaux, successivement dérivés depuis l'expression des besoins, jusqu'à la réalisation. On retrouve généralement trois niveaux, décrits sur la figure 1.

Pour CIM-OSA, ils constituent l'axe dérivation et s'intitulent respectivement :

- niveau de modélisation de l'expression des besoins,

- niveau de modélisation des spécifications de conception,
- niveau de modélisation de la description de l'implantation.

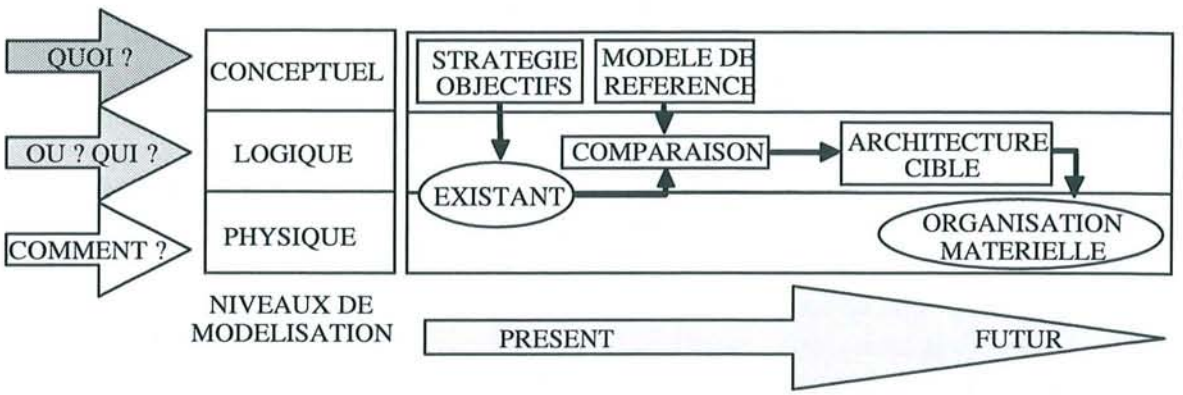

Figure 1. De l'expression des besoins à la réalisation (Foulard, 1994).

Enfin, chaque niveau peut être observé selon plusieurs vues. Plutôt que de voir le modèle dans sa globalité, et donc sa complexité, l'utilisateur peut selon ses intérêts spécifiques, le percevoir différemment. Pour CIM-OSA, l'axe génération est constitué des vues fonctionnelle, informationnelle, ressource, et organisation. Ces déclinaisons selon trois directions ont conduit à la représentation cubique de CIM-OSA illustrée par la figure 2 (Vernadat, 1990). Mais bien d'autres cadres de modélisation peuvent s'articuler selon ce repère tridimensionnel.

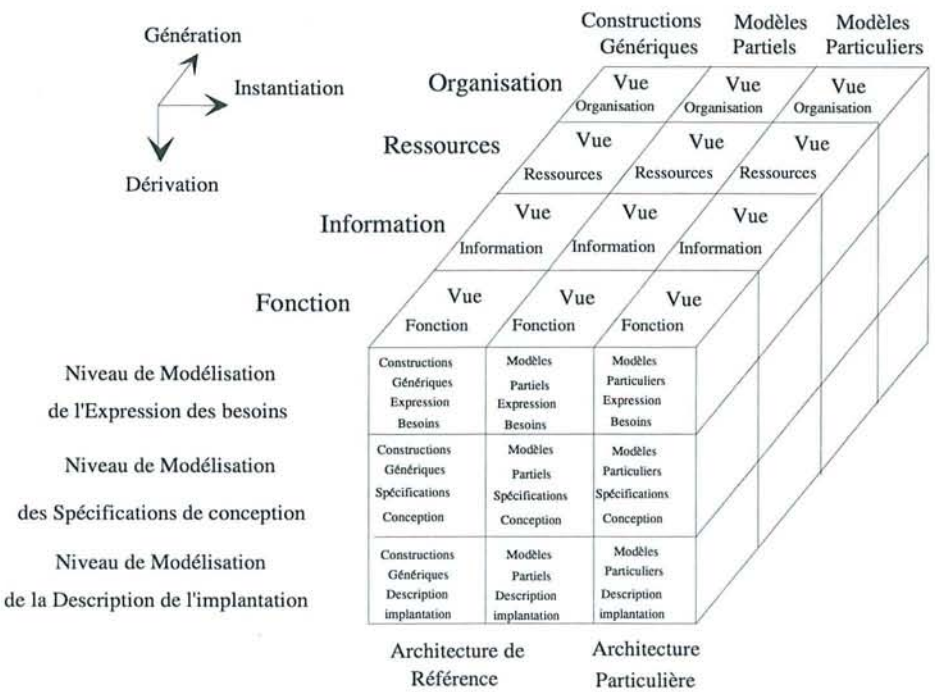

Figure 2 Cadre de modélisation de CIM-OSA.

1.2. Les principaux travaux de recherche dans le domaine

Les premiers travaux dans le domaine datent d'une vingtaine d'années. Ils ont été conduits de façon ponctuelle, mais le besoin évident de normalisation s'est rapidement fait sentir. Chaque entreprise doit concevoir son système intégré, sans risques ni coûts prohibitifs, en s'appuyant donc comme on l'a vu plus haut, sur des architectures de référence. Celles-ci fédèrent des compétences étendues et doivent suggérer les disponibilités technologiques du marché. La définition d'architectures normalisées et d'outils et méthodologies associés est donc devenue incontournable. De façon similaire, une volonté puissante de standardisation conduisit en 1977 à l'adoption par l'ISO (International System Organisation) du modèle OSI (Open System Interconnection) structurant les communications entre systèmes hétérogènes.

Des actions coordonnées à des échelons nationaux ou internationaux ont été entreprises. Aux Etats-Unis, sous l'impulsion d'organismes comme le NIST (National Institute of Standards and Technology) par exemple, mais aussi et surtout en Europe, dans le cadre des soutiens communautaires apportés par la Commission. Mais avant de présenter les résultats les plus significatifs obtenus par les Occidentaux, on peut s'interroger sur le retard apparent des japonais (Fukuhara, 1991).

A la fin des années soixante-dix, le Japon a préféré au concept CIM encore très abstrait, les systèmes flexibles de fabrication ou FMS (Flexible Manufacturing System), solutions intégrées pour petits îlots de production. L'industrie a rapidement évolué vers la notion de FA (Factory Automation), à l'échelle de toute l'entreprise, mais encore essentiellement focalisée sur la production. Parallèlement, les Japonais se sont surtout intéressés à la place de l'homme dans l'atelier, et ont considérablement étudié son poste de travail. Cette dimension sociale a pris toute son importance avec les cercles de qualité (Ishikawa, 1984). Ce n'est qu'au début des années quatre-vingt-dix que le Japon est brusquement entré dans l'ère CIM, avec semble-t-il, beaucoup d'enthousiasme. Le projet IMS (Intelligent Manufacturing Systems) proposant des moyens importants pour conduire cette orientation a eu néanmoins peu d'écho à l'Ouest, et aucun résultat normatif n'a encore été publié ou communiqué. On peut d'ailleurs remarquer que son nom ne transcrit pas la notion d'intégration, mais celle d'intelligence.

Aux Etats-Unis, l'US Air Force a lancé à la fin des années soixante-dix, un programme de développement de la productivité des industries aérospatiales appelé ICAM (Integrated Computer Aided Manufacturing) (Zgorzelski, 1994). Son but était de développer des méthodes d'analyse et de conception de systèmes manufacturiers, en mettant en avant les besoins en communication inter-métiers. On retiendra de ce programme les méthodes IDEF (ICAM DEFinition methodology), au nombre de seize, connues aussi sous le nom de Integration Definition. On peut citer :

- IDEF0 : modélisation fonctionnelle,
- IDEF1 : modélisation informationnelle,
- IDEF2 : modélisation dynamique de simulation (aujourd'hui abandonnée).

Ces méthodes ont été acceptées par les industriels, puis étendues et utilisées pour d'autres projets comme STEP (Standard for Exchange of Product model data), CALS

(Computer aided Acquisition on Logistic Support), ... En 1989, L'IDEF Users Group a été constitué. Associé au NIST, il a entamé des efforts de normalisation. Le NIST est l'auteur d'un modèle hiérarchique CIM à cinq niveaux. Il a étudié et proposé des interfaces standards, c'est-à-dire des liens informationnels entre ceux-ci. ICAM s'inscrivait donc parfaitement dans ses préoccupations.

En Europe, dans le cadre des programmes ESPRIT, de nombreux consortiums se sont constitués pour monter des projets ambitieux parmi lesquels on retrouve CIM-OSA déjà cité. Les résultats sont à la hauteur des moyens mis en œuvre, et la normalisation future s'appuiera sans aucun doute sur la contribution européenne. La très forte implication des industriels les plus puissants du vieux continent a donné à ces travaux une dimension concrète qui laisse augurer leur applicabilité.

Les occidentaux ont donc produit une multitude d'architectures et de modèles (Mertins, 1991) (Doumeingts, 1990). Devant l'importance du nombre des solutions avancées, L'IFAC (International Federation of Automatic Control) et l'IFIP (International Federation of Information Processing), et plus particulièrement leurs comités respectifs "Manufacturing Technology and Computer Committee", et "Technical Committee for Computer Applications in Technology" (TC5), ont uni leurs efforts en 1990 afin d'étudier toutes ces propositions. Une "Task Force" a été constituée, en relation avec l'ISO et l'IEEE (Institute of Electrical and Electronic Engineers) Control Society, ayant pour mission de les recenser, d'isoler la ou les meilleures, et à défaut, de définir les pré-requis d'une architecture générique idéale répondant aux besoins industriels.

Ses conclusions, publiées en 1993 (Williams, 1994), retiennent trois architectures. Ce sont les seules qui développent réellement l'aspect réalisation ou implémentation de l'intégration. Beaucoup s'arrêtent au "quoi" (formalisation des besoins), au "où", "qui" (conception), sans aborder le problème du "comment". L'important est en effet de disposer de cadres de modélisation couvrant l'ensemble du cycle de vie du projet d'intégration. Les architectures lauréates sont :

- CIM-OSA, déjà présentée, développée par le consortium ECIMA (European Computer Integrated Manufacturing Architecture) (ECIMA, 1992),

- GRAI-GIM (GRAI Integrated Methodology), développée à partir des travaux menés au Laboratoire français GRAI, dans le cadre du programme IMPACS cité ci-dessus (Doumeingts, 1992),

- PURDUE, développée à partir des travaux du Purdue Laboratory for Applied Industrial Control (Indiana USA), dans le cadre d'un consortium Université-Industrie (IUC, 1992).

Elles constituent les bases de GERAM (Generic Enterprise Reference Architecture and Methodology) en cours de définition (CSI). Ces trois architectures sont les plus avancées, diffèrent plus sur la forme que sur le fond, et proposent de vrais environnements méthodologiques d'implémentation, bien que la Task Force constate qu'aucune n'est réellement complète en terme d'utilité immédiate. Aussi parmi les recommandations formulées, figure le développement de "plates-formes d'intégration", véritables outils de génération des applications d'ordonnancement, de pilotage, de communication, ...

A titre d'exemple, on peut rapidement présenter l'Infrastructure Intégrante de CIM-OSA (Kosanke, 1992), ou IIS (Integrating Infra-Structure). Elle permet, selon le principe d'instanciation, l'exécution des modèles d'implantation particuliers, par l'intermédiaire d'un ensemble de services (information, traitement, présentation et communication) généralement utilisés dans les systèmes manufacturiers (Klittich, 1990) (Querenet, 1991) (Figure 3).

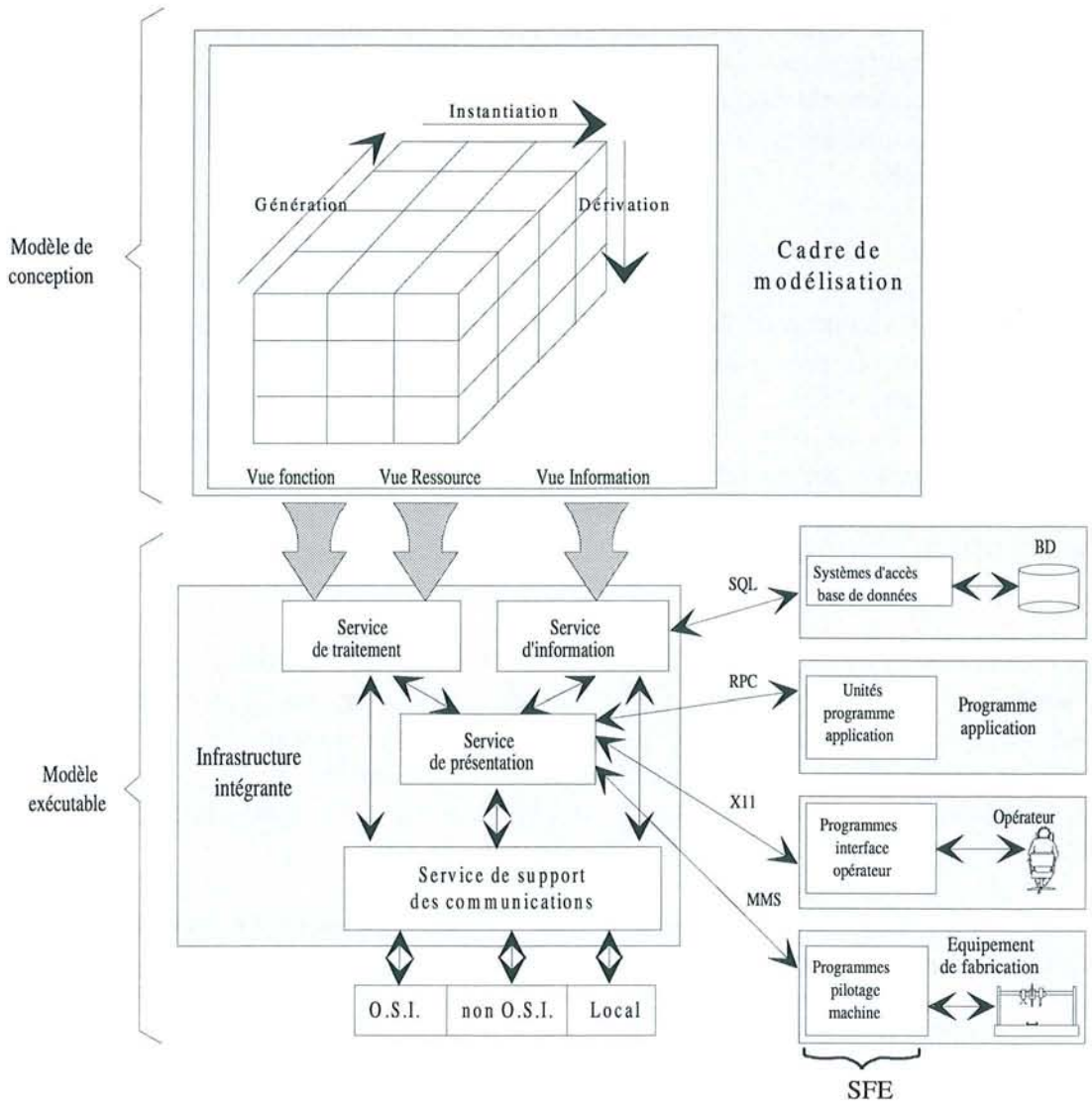

Figure 3. Exécution de modèles CIM-OSA par l'Infrastructure Intégrante.

En particulier, les services relatifs aux communications, qui nous intéressent plus particulièrement, harmonisent l'accès aux ressources physiques (Presentation Services) et sont directement associés aux services de base proposés par les grands standards OSI (Supporting Services) comme MMS (Manufacturing Message Specification) (VOICE, 1993). Des entités fonctionnelles spécifiques (SFE) représentent toutes les fonctions propres à une application (accès à une base de données, gestion de programmes application, dialogue opérateur, pilotage d'équipements de fabrication). Le service de présentation agit comme un agent entre des entités fonctionnelles spécifiques et leurs entités de services communs de l'infrastructure intégrante. De type homme, machine, application ou données, elles peuvent alors

communiquer entre elles, sans savoir comment, et sans avoir besoin de connaître leurs localisations. On réalise ainsi l'intégration des communications (Idelmerfaa, 1994).

Certains travaux suivent les recommandations de l'IFIP/IFAC Task Force présentées plus haut, en proposant de véritables plates-formes d'intégration. En fait, ils sont souvent antérieurs, mais les solutions proposées s'articulent relativement bien avec les architectures de référence que nous avons décrites.

La première instance d'une méthode organisée pour appliquer CIM-OSA est sans doute SEW-OSA (System Engineering Workbench) (Wilson, 1994), résultat des recherches menées depuis dix ans en Grande Bretagne à l'Université de Technologie de Loughborough (Weston, 1993). Il s'agit en fait d'un environnement logiciel intégré autorisant la cohérence et la communication d'outils couvrant l'ensemble du cycle de vie d'un système CIM (Weston, 1994).

D'autres résultats relèvent d'une approche différente. Les cadres de modélisation cités précédemment proposent une démarche descendante, depuis l'expression des besoins, jusqu'à l'implémentation, dont ils ne précisent que les grands principes. En caricaturant un peu, on peut penser que les plates-formes d'intégration existant aujourd'hui sont le résultat d'un cheminement inverse. Recensant les produits absolument indispensables, consensuels donc régis par des normes ou des standards, et disponibles sur le marché, les concepteurs de ces plates-formes les ont associés au sein d'environnements harmonieux, véritables ateliers logiciels pour l'intégration d'applications industrielles. Ceci relève donc d'une démarche ascendante qui rejoint l'aboutissement des phases de modélisation (Kapp, 1995).

Nous citerons ici plusieurs plates-formes d'intégration, développées par des groupes institutionnels :

- CCE-CNMA (CIME Computing Environment - Communication Network for Manufacturing Application), proposée par le consortium du même nom dans le cadre du projet européen ESPRIT 7096,

- DCE (Distributed Computing Environment), définie par l'OSF (Open Software Foundation),

- ANSAware, développée dans le cadre des projets ANSA (Advanced Networked Systems Architecture) et ESPRIT ISA,

Pour illustrer leurs architectures, on peut détailler sommairement CCE-CNMA. Elle est basée comme son nom l'indique, sur une infrastructure standardisée de communication (Communication Network for Manufacturing Application). Celle-ci associe les normes (Lederhofer, 1992) :

- MMS (Manufacturing Message Specification) pour homogénéiser les communications industrielles,

- RDA (Remote Database Access), indépendante des bases de données utilisées, pour y accéder en utilisant SQL,

- FTAM (File Transfert Access and Management), pour le transfert et la gestion des fichiers sur le réseau,

- X500 Directory Service, pour obtenir des informations sur les utilisateurs, applications ou objets, accessibles par le réseau.

L'infrastructure de communication CNMA est totalement indépendante des "couches basses", et donne la possibilité d'implémenter l'environnement CCE sur des architectures normalisées (OSI), standardisées (TCP/IP, Transmission Control Protocol/Internet Protocol), ou propriétaires. CCE-CNMA complète ce noyau avec quatre modules accessibles par l'utilisateur pour le développement de son application (Pleinevaux, 1994a), comme le montre la figure 4 :

- Application Independent Services : propose un ensemble d'objets dits universels et de services les manipulant, définis par CCE, et indépendants de l'application (variables, programmes, domaines, ...),

- Application Dependent Services, offre à l'utilisateur des services dédiés à la manipulation d'objets qu'il a lui même définis,

- Tools : ensemble d'outils pour configurer la plate-forme, associé à un préprocesseur SQL, et à un générateur de variables,

- Administration : ensemble d'outils de gestion, en exploitation, de la plate-forme d'intégration.

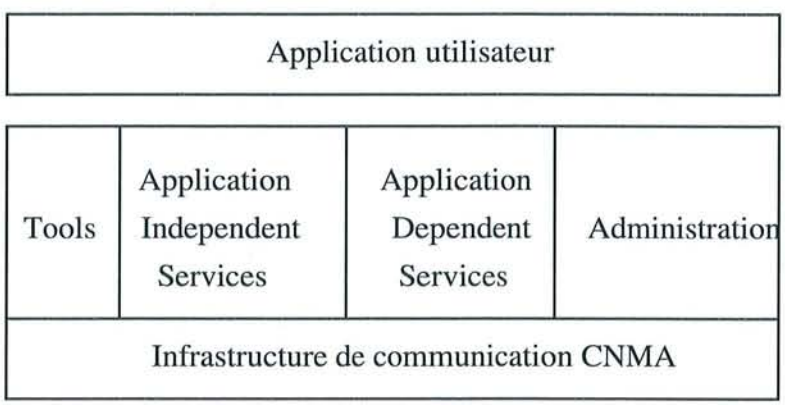

Figure 4. L'architecture CCE-CNMA (Pleinevaux, 1994a).

CCE satisfait la plupart des spécifications définies par CIM-OSA pour réaliser une plate-forme assurant le rôle de son infrastructure intégrante. Leurs services sont cohérents, en particulier vis-à-vis des communications. CCE a été testé dans un premier temps à l'Ecole Polytechnique Fédérale de Lausanne, puis utilisé sur des installations pilotes à l'Aérospatiale, Mercedes-Benz, ...

On peut dire que les recherches menées dans ce domaine doivent nécessairement s'articuler sur des architectures qui, même si elles ne sont pas encore parvenues à un état de maturité suffisant pour être normalisées, sont aujourd'hui incontournables et constituent des bases solides pour poursuivre. Les incitations sont multiples : organismes nationaux ou

internationaux, associations d'industriels, se penchent sur le problème et contribuent à sa résolution. Dans ce cadre, les efforts visent à maintenir un niveau d'abstraction suffisant, qu'il faut préserver devant la multitude de pré-solutions plus ou moins fermées que les grands constructeurs proposent aujourd'hui (Schweizer, 1995) (voir plus haut plates-formes d'intégration).

Mais rappelons que le système d'information de l'entreprise constitue un ensemble de données utilisées à long et court terme, ainsi qu'un ensemble d'informations produites, stockées, maintenues et traitées pour les besoins des utilisateurs et des applications. Dans le cadre de CIM-OSA, le système d'information, dont l'intégration des fonctions et des activités de l'entreprise est assurée, est représenté par la vue informationnelle (Barbier, 1992). Dans la section suivante, nous aborderons les méthodes d'analyse et de conception des systèmes d'information, des méthodes fonctionnelles à l'approche orientée objet.

2. La modélisation des systèmes d'information

Il est nécessaire de définir certains termes utilisés tout au long de ce paragraphe. Un modèle est une représentation du réel, qui doit posséder plusieurs propriétés. Il doit être lisible, compréhensible, communicable, porteur de sens et donc rédigé ou décrit dans un langage comportant des concepts organisés selon une syntaxe rigoureuse n'induisant qu'une seule interprétation possible. Une méthode est un assemblage de trois composantes : une démarche, un ensemble de concepts et un ou plusieurs outils (Larvet, 1994).

Nous présentons quelques-unes des méthodes utilisées dans la spécification des systèmes d'information. Deux catégories existent : les méthodes fonctions/données (non orientées objet) et les méthodes orientées objet (Jacobson, 1992). Les premières se concentrent sur des fonctions et/ou des données de manière plus ou moins séparée. Les secondes considèrent que fonctions et données sont étroitement liées.

2.1. Les méthodes non orientées objet

Les méthodes dites non orientées objet peuvent être classifiées en plusieurs types. Ces méthodes sont soit orientées fonctions, orientées données, orientées comportement ou combinées. Nous les aborderons brièvement ci-après.

2.1.1. Méthodes orientées fonctions

Ces méthodes se basent sur les fonctions des systèmes : Qu'elles sont les fonctionnalités attendues des systèmes ?. Ces méthodes se distinguent par la démarche suivie : soit une démarche basée sur les fonctions que doit réaliser le système et leur décomposition en sous-fonctions, soit basée sur le concept de processus et d'unités de stockage de données, ...

2.1.1.1. La décomposition fonctionnelle

Le système à développer est représenté par les fonctions qu'il doit réaliser, des sous-fonctions et des interfaces fonctionnelles (Coad, 1991a). La démarche suivie consiste à

sélectionner des étapes et sous-étapes de traitements anticipés pour un nouveau système. Ainsi le domaine du problème étudié est indirectement représenté et découle de la décomposition en fonctions et sous-fonctions. L'inconvénient majeur de cette méthode est que les décompositions en fonction/sous-fonctions sont difficiles à construire, à cause de la correspondance indirecte avec le domaine du problème étudié, et sont hautement volatiles, à cause du changement continuel des fonctionnalités.

2.1.1.2. La décomposition par les flots de données (l'analyse structurée, SA)

Le *processus* représente le concept central et transforme un flux de données entrant en un flux de données sortant, en utilisant des unités de stockage de données, des terminaisons et un dictionnaire de données (Coad, 1991a). Ceci est représenté par les diagrammes dits diagrammes de flots de données (DFDs). Au cours du développement, les traitements sont affinés, jusqu'à ce que tous les traitements aient été décrits en terme de flots de données.

Cette approche a été écartée par son incapacité à définir un schéma de base de données. Ainsi les données sont vues comme des paramètres passés aux processus et non pas comme un ensemble cohérent doté d'une structure. Le comportement (ou la dynamique) du système, n'est pas abordé. Une extension de la méthode en vue de modéliser les systèmes temps-réel a donné lieu à la méthode SA-RT (Structured Analysis Real-Time) (Ward, 1986).

2.1.2. Méthodes orientées données

Ces méthodes constituent incontournablement une avancée par rapport à la famille précédente dans la mesure où elles reconnaissent la nécessité d'expliciter des niveaux d'abstraction dans la représentation. On peut considérer qu'elles forment la base des méthodes de seconde génération dites méthodes conceptuelles. Le diagramme entité-association, développée par (Chen, 1976), représente l'outil principal de cette modélisation.

2.1.3. Méthodes orientées comportement

Bien qu'ils présentent l'avantage de pouvoir représenter explicitement les contraintes liées à la dynamique et l'évolution dans le temps, les modèles issus de ces méthodes (Remora, JSD) privilégient ensuite soit l'aspect données (Remora) (Rolland, 1988) soit l'aspect traitement (JSD, Jackson System Development) (Jackson, 1983).

2.1.4. Méthodes combinées

Ces méthodes (Merise, SADT) proposent de prendre en compte les deux aspects données et traitements mais de façon séparée. En voici quelques-unes :

- Merise (Tardieu, 1987) s'appuie sur un découpage en trois cycles pour la conception du système d'information : le cycle de vie permettant de faire évoluer le système d'information dans l'organisation, le cycle de décision permettant la hiérarchisation des décisions avec la présence indispensable des groupes de travail adaptés aux prises de décisions et le cycle d'abstraction consistant en la description du système d'information en trois niveaux : conceptuel, organisationnel ou logique et physique ou opérationnel.

- SADT, Structured Analysis and Design Technique (Marca, 1988) semble accorder une place secondaire aux datagrammes (les diagrammes de données), mais les actigrammes (Les diagrammes d'activités) inspirés des diagrammes de flots de données DFDs ont deux concepts supplémentaires : les données de contrôle et les moyens nécessaires à l'activité.

La figure suivante (Figure 5) présente les méthodes dites non orientées objet selon les aspects : fonctions, données et comportement. Aucune des méthodes présentées n'occupe la position idéale au sein de ce triangle. Dans les méthodes fonctions/données, l'accent est mis sur la spécification et la décomposition des fonctionnalités du système. Une telle approche semble être la plus efficace dans une optique d'implémentation. Mais un système s'appuyant sur une décomposition fonctionnelle nécessite une restructuration importante lorsque les besoins évoluent. Ainsi, un système développé par ces méthodes devient instable et souvent difficile à maintenir.

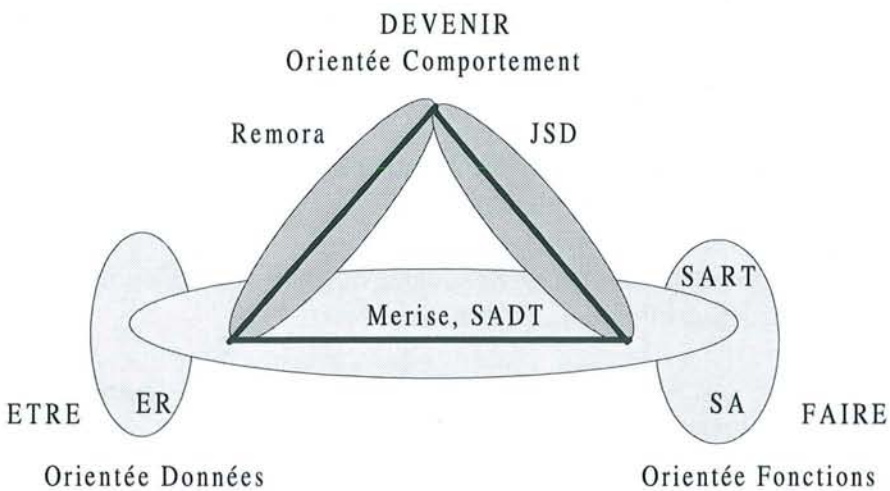

Figure 5. Classification des Méthodes.

L'approche Orientée Objet s'appuie sur une démarche de modélisation et de concepts intégrant les trois aspects : données, traitement et comportement. Cette approche est décrite dans la suite de ce document.

2.2. Les méthodes orientées objet

La méthode orientée objet est une technique de développement qui permet de modéliser le système sous forme d'objets interagissant les uns avec les autres. Grâce à cette approche la différence de sémantique entre réalité et système (une adaptation à la manière dont les gens perçoivent la réalité) est faible. Les modèles sont ainsi faciles à comprendre et représentent directement le domaine du problème étudié. De plus, les modifications apportées au modèle peuvent être bien localisées, car elles n'ont souvent d'impact que sur un seul élément représenté par un objet unique. En effet le système s'appuie sur la structure sous-jacente du domaine d'application étudié (Rumbaugh, 1991).

Quatre principes de base permettent de cerner complètement la notion d'objet en phase d'analyse : l'abstraction, l'encapsulation, l'héritage et la modularité. L'***abstraction*** est

une des voies fondamentales par laquelle l'esprit humain aborde la complexité. Elle consiste à se concentrer sur les aspects essentiels, inhérents d'une entité et à en ignorer les propriétés accidentelles. L'***encapsulation*** (dite aussi le masquage d'information) consiste à séparer les aspects externes d'un objet, accessibles par les autres objets, des détails de son implémentation interne, invisibles des autres objets. L'***héritage*** est le partage des attributs et des opérations entre des classes s'appuyant sur une relation hiérarchique. La ***modularité*** se définit par l'acte de partitionner un système en composants individuels dans le but de réduire sa complexité.

Peter Coad (Coad, 1991a) résume les avantages et les raisons de l'approche orientée objet par les points suivants :

- elle aborde mieux les questions du domaine du problème étudié et améliore l'interaction entre l'expert du domaine étudié et l'analyste,

- elle augmente la cohérence interne des résultats de l'analyse et réduit la séparation entre les différentes activités de l'analyse, en traitant les attributs et les services comme un tout intrinsèque,

- elle représente explicitement les éléments communs et utilise l'héritage pour identifier et capitaliser sur les éléments communs des attributs et des services,

- elle construit des spécifications résiliantes au changement, en intégrant la volatilité dans les constructions du domaine étudié,

- elle réutilise les résultats de l'analyse, en ajustant les compromis pratiques à l'intérieur du système,

Parmi les méthodologies, on peut citer :

- l'analyse orientée objet des systèmes OOSA (Object-Oriented System Analysis) de Shlaer et Mellor (Shlaer, 1988). C'est essentiellement de l'analyse d'information basée sur la modélisation de données. OOSA ne réussit pas à représenter le comportement et ne porte ni l'héritage ni la classification. Par contre elle se concentre sur les relations entre les objets.

- l'analyse orientée objet OOA (Object Oriented Analysis) de Caod et Yourdon (Caod, 1991a) est une méthodologie incrémentale pour l'élaboration des différents modèles du système. Une méthodologie de conception (OOD, Object Oriented Design) et de programmation (OOP, Object Oriented Programming) (Caod, 1991b) (Caod, 1993) ont suivi la méthodologie OOA.

- la conception orientée objet OOD (Object Oriented Design) de Booch (Booch, 1991, 1993) permet d'analyser et de concevoir les systèmes d'information en s'appuyant sur la notion d'objets coopérants. Elle permet de réaliser une application, et ce, quelque soit sa complexité grâce à une démarche itérative et incrémentale. Elle aide à gérer des applications de façon très modulaire, optimisant en parallèle l'organisation des équipes qui les conçoivent et les implémentent.

- la conception descendante orientée objet HOOD (Hierarchical Object Oriented Design) est une approche développée par ESA (European Space Agency). C'est une méthode de conception descendante orientée objet, utilisant une notation proche de Ada. HOOD est particulièrement adaptée pour les gros logiciels temps réel, scientifiques et techniques nécessitant un développement réparti (Le Bris, 1993).

- le génie logiciel orienté objet OOSE (Object-Oriented Software Engineering) de Jacobson (Jacobson, 1992) est un processus de développement prenant en compte les besoins du point de vue utilisateurs en se basant sur des cas d'utilisation.

- la technique de modélisation objets OMT (Object Modelling Technique) de Rumbaugh (Rumbaugh, 1991, 1995) est une méthode d'analyse et de conception orientée objets. Elle fournit un formalisme de description graphique du modèle objet assez complet, un modèle d'abstraction de l'aspect dynamique du système bien intégré et une perspective d'intégration du modèle fonctionnel. OMT est fondée sur des standards tels que entité-relation, diagrammes des flots de données et diagrammes d'états étendus ou statecharts (Harel, 1987, 1996) étendus aux objets. L'approche OMT couvre l'ensemble du cycle de vie du logiciel (à l'exception des parties «test» et «maintenance») à travers l'analyse, la conception du système, la conception des objets et l'implémentation. Elle est assez complète et se caractérise par une distribution selon trois axes principaux, chacun représenté par un modèle distinct :

  - le «modèle objet» (sur quoi agit le système ?) : description de la structure (partie statique) des objets et leurs relations dans le système,

  - le «modèle dynamique» (quand se font les actions ?) : description des interactions entre objets dans le système,

  - le «modèle fonctionnel» (en quoi consistent ces actions ?) : description des procédés de transformation.

La Figure 6 présente le processus de développement d'OMT pour un système.

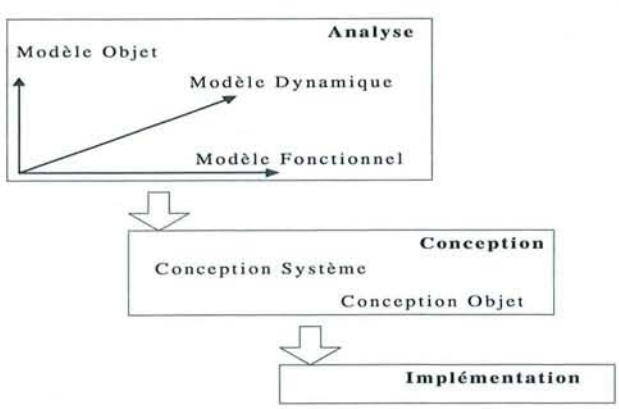

Figure 6. La démarche d'OMT (Laroque, 1995).

2.3. Synthèse

L'utilisation d'une méthode nécessite de bien maîtriser ses domaines d'application, ses possibilités, ses limites et ses difficultés de mise en œuvre. Ainsi, si l'on veut qu'une méthode apporte une meilleure productivité, il est nécessaire de la choisir en fonction de critères tels que :

- les finalités et les stratégies de l'entreprise,
- les acteurs concernés,
- le domaine d'application,
- les étapes du cycle de vie couvertes par la méthode,
- le niveau d'outillage de la méthode.

Nous avons retenu dans nos travaux la méthode OMT essentiellement pour sa capacité de décrire le système d'information et parce qu'elle satisfait bien les critères précédents. Il est à noter que cette méthode est largement utilisée et son apprentissage est assez rapide ce qui est primordial pour une méthodologie devant être utilisée par des personnes de disciplines différentes. OMT est supportée par un grand nombre d'outils informatiques (exemple : OMTool, LovOMT, Select OMT, PowerBuilder, Rose,...). Par conséquent, ces derniers autorisent une vérification de la cohérence du système analysé.

Le choix de la méthode OMT est également justifié par le fait que cette dernière est en voie de normalisation avec les méthodes Booch et Jacobson pour donner naissance à la méthode dite unifiée (UML; Unified Modelling Language). UML (Muller, 1997) représente la troisième génération de langage de modélisation orientée objet. Les deux méthodes majeures OMT (Rumbaugh, 1995) et Booch représentent 50% du marché des méthodes orientées objet et elles ne cessent de converger sur les points majeurs et les différences sont souvent superficielles (terminologie, notation, ...).

L'unification de ces méthodes a pour but de fournir le standard dont l'industrie a besoin, d'apporter au marché la stabilité nécessaire et d'ouvrir à la concurrence des outils fournissant les fonctionnalités nécessaires. Les outils qui implémentent les différentes méthodes objets et les outils multiméthodes vont progressivement intégrer la méthode unifiée. Le but est d'apporter des améliorations méthodologiques et de s'ouvrir à des domaines non encore traités tels que les systèmes distribués. Ainsi, les utilisateurs d'UML n'auront plus à se poser la question du choix de la méthode.

L'unification a débuté en octobre 1995 et un premier draft public (V0.8) a été rédigé. Puis, les documents ont été révisés en réponse aux commentaires faits début 1996 (V0.9). La méthode a été présentée à l'OMG (Object Management Group) dans la perspective de devenir un langage de modélisation normalisé pour le développement orienté objet. Ensuite, la version (V1.0) a été publiée pour standardisation mi-1996 et pour enfin standardiser les artefacts de la conception (modèles sémantiques, notation syntaxique, diagrammes et données à échanger). La version (1.1) a apporté plus de formalisme, l'unification de la sémantique des

relations, l'extension de la sémantique des cas d'utilisations (use case), ... Actuellement une version (V2.0) (Selic, 1998) a été publiée pour la conception des systèmes dits temps-réel.

D'autres recherches ont été réalisées dans une optique d'unification de différents modèles et méthodes. On peut citer le modèle générique MGCO2 (Modèle Générique de Conception Orientée Objet des systèmes d'information) développé par Gargouri et al (Gargouri, 1997). Cette proposition donne plus d'indépendance à l'utilisateur vis-à-vis d'une méthode spécifique et permet d'avoir les mêmes référence et langage pour toutes les méthodes. Ainsi, différents concepteurs peuvent utiliser différentes méthodes pour le développement des parties d'une même application. Ces travaux vont constituer la base de nos propositions. Nous les aborderons plus en détail dans le chapitre suivant.

2.4. L'utilisation des méthodes OO pour la conception des systèmes de communication

Un autre argument pour le choix d'une représentation objet est que l'approche objet n'est pas uniquement utilisée que pour l'analyse et la conception des systèmes d'information, mais aussi dans le domaine de la communication. En voici quelques exemples :

- Dans la gestion des réseaux, la MIB (Management Information Base) représente l'aspect informationnel du modèle de gestion de système. Elle est constituée par un ensemble d'objets gérés (objets peuvent être des vues de la gestion OSI (Open System Interconnection) des ressources du système ou des vues abstraites d'une ressource représentant ses propriétés de gestion) du système avec leurs attributs. Ainsi, la MIB est basée sur le concept d'objet, en terme d'attributs, d'opérations et de notifications. Elle définit également les notions de classes d'objets, d'arbres d'héritage et de contenance (Claudé, 1993).

- Dans le domaine de la messagerie industrielle, MMS (Manufactruring Message Specification) est un protocole de la couche application du modèle OSI. MMS spécifie les ressources des équipements industriels à piloter en terme d'objets, d'attributs et de services qui les manipulent. Ainsi, MMS utilise la technique de modélisation objets pour la spécification d'un service OSI (Pleinevaux, 1994b). Un concept fondamental est défini : le VMD (Virtual Manufacturing Device). Ce dernier est pour le pilotage de l'équipement ce que la MIB est pour la gestion de réseau.

Comme nous venons de le voir précédemment, l'infrastructure intégrante de CIM-OSA, et en particulier, les services relatifs aux communications qui nous intéressent plus particulièrement, harmonisent l'accès aux ressources physiques (Presentation Services) et sont directement associés aux services de base proposés par les grands standards OSI (Supporting Services). Ainsi, pour les communications industrielles, CIM-OSA recommande MMS (Manufacturing Message Specification). De même, la plate-forme CCE-CNMA présentée plus haut, s'appuie sur une architecture de communication qui implémente la messagerie MMS que nous présentons maintenant.

3. La messagerie industrielle MMS

3.1. Introduction

L'intégration d'un système de production passe nécessairement par une communication adaptée et fiable dans l'entreprise. Ainsi, l'évolution des systèmes de production vers les architectures intégrées nécessite le partage de différents types de ressources (matérielles et logicielles), et pour la commande et le contrôle de processus, la coordination des tâches réparties. L'échange d'informations entre les équipements de production et de gestion est primordiale. Les réseaux locaux industriels (RLI) sont des systèmes de communication répondant à ces besoins (Lepage, 1991).

Après avoir défini le modèle OSI (Open System Interconnection) décomposant le protocole général de communication entre systèmes hétérogènes, les organismes de normalisation ont précisé couche par couche les règles spécifiques régissant les échanges d'information. Considérant alors le réseau de communication comme un outil global offrant des services fiables au transfert de données, il a été nécessaire de les caractériser pour les différents types d'application.

Dans un milieu industriel, l'interconnexion de machines de production nécessite un langage de haut niveau approprié. La messagerie industrielle MMS (Manufacturing Message Specification) (ISO1, 1987) représente ce langage de commande, qui évite la personnalisation et la conversion des procédures de commandes pour interconnecter des équipements industriels. Avec MMS, l'intégration de machines revient à se concentrer sur la programmation des tâches orientées application et même à réutiliser des programmes existants. Respectant le modèle de dialogue client-serveur, MMS offre des services manipulant les objets qui représentent les systèmes réels.

3.2. Le VMD (Virtual Manufacturing Device)

Les fonctions offertes par MMS sont fondées sur une représentation abstraite d'un ensemble spécifique de ressources et de fonctionnalités associées à un dispositif réel, le VMD, dispositif de fabrication virtuelle (Figure 7). Celui-ci modélise le comportement de l'équipement tel qu'il est perçu de l'extérieur. Il est accessible par MMS au travers d'un nombre limité d'objets prédéfinis que nous décrivons par la suite. Ce modèle est exploité par un programme utilisateur pour réaliser les fonctions de pilotage, de contrôle et de transfert de données (Brill, 1991). De plus, le VMD contient une fonction dite "exécutive" qui réalise, à la réception d'une commande, les actions voulues sur l'équipement réel (lancement de tâches, mise à jour de sorties...).

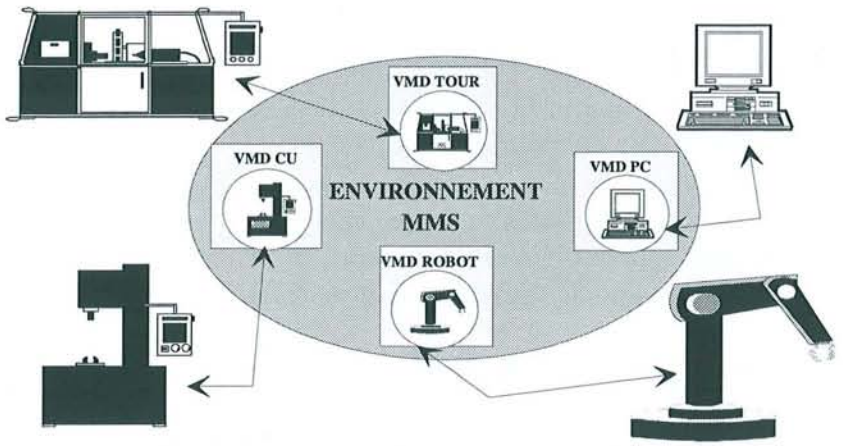

Figure 7. Principe de la communication par l'intermédiaire des VMD.

3.3. Les Objets MMS

La norme MMS définit un certain nombre de classes d'objets (variables, sémaphores, événements, programmes exécutables, journaux, ...) (Mackiewicz, 1994). Chaque objet est une instance d'une classe et constitue une entité abstraite réunissant certaines caractéristiques. Une classe possède un nom qui permet d'y faire référence. La classe est caractérisée par un ensemble d'attributs qui définissent les propriétés communes des objets de cette classe. Ces attributs possèdent tous un type particulier : entier, caractère, chaîne de caractères etc... Ces types sont ceux définis par la norme ISO 8824, plus les types complexes construits autour des types primitifs de la syntaxe abstraite ASN1 (Abstract Syntax Notation 1), auxquels sont donnés des noms particuliers.

Un objet doit être identifié sans ambiguïté. Pour cela, on distingue un objet d'un autre par une valeur unique de l'un de ses attributs ou une combinaison des valeurs de plusieurs de ses attributs. Chaque attribut appartenant à cette combinaison est appelé attribut clé. Enfin des objets contiennent des attributs conditionnels, au sens qu'ils ne caractérisent l'objet que si et seulement si certaines conditions sont remplies. Ces attributs sont appelés contraintes. Certains objets référencent, par l'intermédiaire d'attributs, d'autres objets. Ces attributs sont appelés attributs de référence. Si un objet référencé est instancié plusieurs fois, il l'est via un attribut de liste. Le Tableau 1 donne une représentation générale d'un objet MMS.

```
Objet : (nom de la classe)
        Attribut-clé : (nom de l'attribut (type,valeur))
        Attribut-clé : (nom de l'attribut (type,valeur))
        Attribut : (nom de l'attribut (type, valeur))
        .
        Attribut : (nom de l'attribut (type, valeur))
        Contrainte : (expression de la contrainte)
          Attribut : (nom de l'attribut (type, valeur))
          .
          Attribut : (nom de l'attribut (type, valeur))
```

Tableau 1. Représentation d'un objet MMS.

3.4. Services MMS

La norme MMS définit au total quatre vingt six services (Henault, 1995) agissant sur les objets précédemment cités. Ces services sont regroupés en dix ensembles distincts par leurs fonctionnalités (Figure 8). Par exemple, les services de gestion des variables permettent la lecture, l'écriture et la gestion d'alarmes sur des équipements distants (*read*, *write*, *informationreport*, ...); les services de gestion de programmes autorisent le contrôle d'applications déportées (*start*, *stop*, *kill*, *reset*, ...).

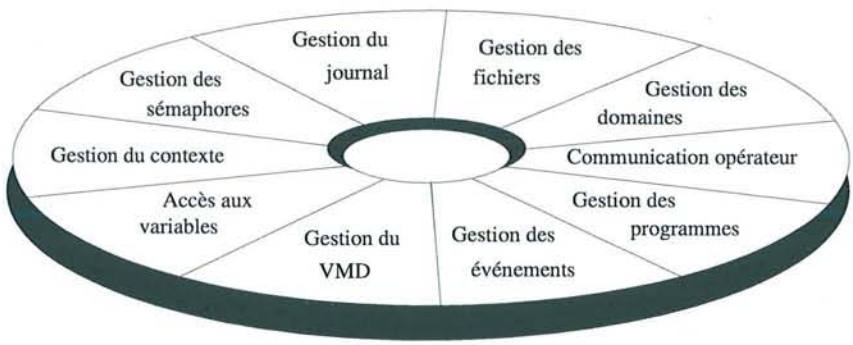

Figure 8. Les groupes de services MMS.

3.5. Normes d'accompagnement MMS

Les spécificités des communications industrielles ont conduit à définir un modèle général permettant aux équipements de dialoguer entre eux : c'est la norme MMS. Le processus de normalisation s'est poursuivi en instanciant la norme MMS en fonction des besoins en communication de chaque type d'équipements : ce sont les normes d'accompagnement *(companion standards)*. On définit en fait l'image et le comportement des grandes familles d'équipement de production : robot (ISO3, 1991), commande numérique (ISO4, 1989), automate (ISO5, 1989), contrôle de processus (ISO6, 1990) ...

Ainsi, les constructeurs peuvent sans ambiguïté, proposer des modèles de leurs produits au format normalisé des normes d'accompagnement. Puisque les normes citées

décrivent des types de matériel spécifique, elles sont appelées opérationnelles. Il existe une deuxième catégorie de normes d'accompagnement appelées fonctionnelles, puisqu'elles décrivent des types de fonctions (GPAO ; Gestion de Production Assistée par Ordinateur, le Multimédia) et elles déterminent une approche exhaustive d'expression des besoins des utilisateurs. Cette catégorie se caractérise par un contenu et une structure stables qui ne seront plus remis en cause par les évolutions technologiques (Patz 1990) (Rae, 1989), ce qui n'est pas le cas pour la première catégorie.

3.6. MMS et les modèles d'entreprise intégrée

La messagerie industrielle MMS offre tous les services nécessaires à l'établissement des communications dans un système automatisé de production. Elle présente des qualités remarquables qui autorisent l'élaboration d'applications portables, modulaires et réutilisables. Aussi, les différents projets comme CIM-OSA et IMPACS (Integrated Manufacturing Planning And Control System) (Chen, 1990) qui définissent des architectures CIM, retiennent cette norme pour la mise en œuvre des communications entre les équipements de production.

3.7. L'évaluation des performances de MMS

Mais outre les profits tirés de l'utilisation de MMS, cette dernière souffre d'une lacune importante qui est la non prise en compte des contraintes temporelles. Ainsi les performances temporelles des services MMS ne sont pas définies dans la norme. Ceci a été le noyau des travaux menés au CRAN par Afilal (Afilal, 1989) et Daoudi (Daoudi, 1994) qui avaient pour objectif l'étude de l'évaluation des performances des services MMS.

- Afilal a proposé un modèle de comportement des services MMS prévu pour être inséré entre le service d'application et le processus d'application et permettant de gérer correctement le service. Le modèle est décrit par un grafcet ayant des étapes représentant ses différents états et des transitions prenant en compte ses conditions d'évolution. L'étude a été expérimentée sur deux plates-formes de test sur du réseau MAP (Manufacturing Automation Protocol) et une plate-forme sur du réseau LAC. Différentes configurations ont été prises en compte durant l'expérimentation. Les résultats mettent en évidence l'influence des paramètres tels que l'augmentation des performances globales de l'équipement hôte, la vitesse d'horloge du processeur de l'équipement hôte, la vitesse de transmission sur le médium pour un réseau donné, la structure et la technologie du réseau, sur le comportement temporel de différents services.

- ils montrent que, les performances des services de la messagerie industrielle, en particulier leurs temps de réponse, varient d'une implémentation à l'autre. L'étude de l'évolution de ce temps permet à un concepteur de vérifier le comportement de son application et de sa fiabilité avant son implémentation. Daoudi (Daoudi, 1994) a étudié l'évolution du temps de réponse des services MMS sous différentes conditions de configuration et de charge. Ceci a été réalisé dans une démarche classique d'expérimentation (mesures) et de modélisation. Les mesures ont été réalisées sur une plate-forme MAP/MMS qui montrent que les performances de ces services se dégradent en fonction de l'augmentation du nombre de connexions et de la charge soumise par l'application.

Une autre constatation était que la variation que peut présenter un temps de réponse d'un même service est due principalement aux temps de transmission des PDU-MMS, unité de donnée protocolaire (Protocol Data Unit) par les couches inférieures constituant le réseau de communication. Les mesures de performances, basées sur l'observation, permettent d'obtenir des indices de performance réels sur le comportement du système opérationnel. Elles permettent le suivi et le contrôle en ligne de l'application. Une approche de modélisation des performances de services MMS prend en compte les caractéristiques d'une implémentation réelle de cette messagerie. Cette approche a permis de construire des modèles en utilisant le formalisme des réseaux de Pétri stochastiques colorés.

3.8. Extension temporelle de MMS

Comme nous venons de le souligner, MMS n'intègre pas le temps. Ainsi, des travaux de recherche ont été menés en vue d'adapter MMS aux besoins des applications contraintes par le temps. Une intégration des caractéristiques temporelles aux services MMS rend la modification de la norme actuelle inévitable.

- Rodd et al (Rodd, 1990) ont analysé les besoins temporels dans les systèmes de production automatisés distribués afin de garantir ces contraintes dans MMS. Ils ont proposé une modification du standard MMS et ont mis en évidence la nécessité de disposer de services de diffusion dans MMS. De nouveaux attributs ont été définis pour les objets MMS existants et également de nouveaux objets et services sont proposés. Un RTVMD (Real-Time Virtual Manufacturing Device), qui est le VMD classique de MMS avec une modification concernant l'ajout d'attributs temporels, a été développé. L'inconvénient de cette proposition est qu'elle est incompatible avec les systèmes MMS classiques.

- Une deuxième extension temps-réel a été proposée par Akazan et al (Akazan, 1995a). Sa proposition est fondée sur le principe d'intégration des spécifications temporelles dans les objets et les services MMS tout en gardant la norme actuelle avec sa structure d'objets et de services. De nouveaux attributs servant à la datation de ces objets et services ont été rajoutés. Les protocoles de communication ont alors été modifiés en regard de l'extension proposée.

- Castori (Castori, 1995a, 1995b) propose plusieurs niveaux de modifications de la norme MMS en prenant en compte les contraintes temporelles. Un certain niveau de garantie des contraintes temporelles peut être acquis en procédant à quelques changements de procédures de services MMS et à quelques modifications de leurs paramètres. L'intérêt de cette proposition réside dans le fait que le protocole MMS reste inchangé, mais des modifications sont à prendre en compte pour le comportement de l'application.

3.9. MMS et le multimédia

En ce qui concerne les travaux précédants, les couches basses du système de communication sont supposées capables de fournir des services temps-réel pour MMS. Aucune négociation des ressources de communication n'est prévue. Par ce fait, ces

propositions ne sont pas suffisantes pour intégrer des fonctionnalités comme le multimédia directement dans le contexte MMS.

- l'idée d'étendre la norme MMS pour le multimédia a été concrétisée par les travaux menés par Pokam (Pokam, 1995). Il a proposé un VMD distribué pour le multimédia en définissant de nouveaux objets et services. Ce VMD regroupe les équipements multimédias et le système de communication, pour fournir des services multimédias. Cette proposition présente plutôt un système dédié multimédia qu'une extension de MMS. Ceci est dû au fait que l'adaptation des objets et services MMS définis dans la norme actuelle s'avère impossible.

- Wu (Wu, 1997) a proposé un modèle du système de communication, le VCS (Virtual Communication Support), qui est au réseau ce que le VMD est pour l'équipement de production. Le but est de contrôler le système de communication par MMS comme c'est le cas pour les dispositifs de fabrication. Ce travail a été mené au CRAN et sera présenté par la suite.

3.10. D'autres messageries industrielles

Il faut mentionner que MMS n'est pas la seule messagerie industrielle dans une entreprise. Au plus bas niveau de la pyramide CIM les réseaux de terrain ou fieldbus assurent les échanges, entre capteurs et actionneurs répartis dans le processus à commander et les automatismes primaires ou réflexes. La finalité de ces réseaux est d'offrir un langage de communication pour les applications de pilotage ou de suivi en temps réel de procédés industriels. Pour ces réseaux, les contraintes temps-réel sont strictes, et l'information ne doit pas être entachée d'erreur. On peut citer FIP (Field Bus Protocol), Profibus, ...

- FIP (Elloy, 1989) est accompagné d'une norme au niveau de sa couche application : MPS (Manufacturing Periodic Services). MPS a été spécifiée comme un ensemble de services de gestion d'une base de données réparties et spécifie des objets qui permettent l'accès à des données physiques distribuées. Le rôle de MPS est de mettre à la disposition de l'utilisateur l'ensemble de ces variables réparties de l'application et d'en qualifier la validité. Les objets et services ont donc pour rôle essentiel d'assurer ces fonctions de base, et de permettre ainsi à un programme d'application de ne traiter que les variables valides. MPS spécifie la syntaxe abstraite et la syntaxe de transfert de toute variable, mais ces syntaxes ne font que décrire les différents formats «numériques» qui peuvent être adoptés pour décrire et coder des données transportées par FIP.

- Sur Profibus (Alfter, 1992), la couche application se scinde en deux composantes : FMS (Field Message Specification), spécification communication utilisateur et LLI (Low Layer Interface) interface couches inférieures. FMS, est une description des objets de communication et des services d'application ainsi qu'une description des modèles qui en découlent du point de vue de la station avec laquelle on communique. LLI représente une adaptation des fonctionnalités de l'application aux multiples caractéristiques de la couche 2. Un VFD (Virtual Field Device ou équipement de terrain virtuel) est défini et représente la partie accessible par la communication. Des objets sont définis tels que variables, programmes, plages de données ou domaines, événements, ... Ces objets sont recensés, pour une station

Profibus, dans son répertoire d'objets locaux (OD ; Object Directory). On distingue 8 groupes de services : gestion de contexte, de variables, de domaines, de programmes, des événements, OD et support VFD. Les services sont avec ou sans accusé de réception.

- les réseaux de multiplexage ou réseaux embarqués (ISO, 1994a) CAN (Controller Area Network) VAN (Vehicle Area Network) sont apparus dans le domaine de l'automobile. Ils se situent entre les réseaux de capteurs et les bus de terrain. Il n'existe à ce jour aucune normalisation relative à la couche application.

  - CAN (ISO, 1994b), initié par un équipementier automobile (Robert BOSCH Gmbh), supporte deux produits au niveau de la couche d'application tels SDS développé et commercialisé par Honeywell, DeviceNet d'Allen Bradley et CAL (Can Application Layer). Ce dernier a définit une couche application unique. CAL est divisée en quatre unités fonctionnelles qui ont pour nom CMS (CAN Message Specification), DBT (iDentifier Distributor), NMT (Network Management) et LMT (Layer ManagemenT).

  - VAN (ISO, 1994c), initié par les constructeurs français PSA et Renault, est associé à VDX (Vehicle Distributed eXecutive) qui est un système exécutif distribué pour véhicule et fournit des services de communication et des services temps réel. Les services de communication sont construits au-dessus de la couche LLC prenant en compte les besoins de communication synchrones, asynchrones et de transfert de données. Les services temps réel sont construits sur la base d'un exécutif temps-réel et permettent de supporter les besoins des applications multitâches, de l'horloge temps-réel, et de l'ordonnancement cyclique. Un projet de normalisation d'une couche supérieure est en cours et son nom est OSEK (Kiencke, 1998).

- Dans le domaine de la domotique, le protocole X-10 représente un protocole standardisé qui est largement supporté par la plupart des fabricants de composants domotique. Ce protocole permet la communication entre les composants d'automatisation domiciliaires et régit l'adressage de chacun de ces composants de même que les commandes qui doivent être exécutées par chacun de ceux-ci.

3.11. Conclusion sur les messageries industrielles

La messagerie a pour but essentiel d'offrir à l'utilisateur des services de haut niveau l'affranchissant des problèmes d'acheminement physique de l'information. Elles proposent également des formats de données standards, voire des représentations objets des équipements qu'elles servent.

Mais comment peuvent-elles être cohérentes avec le système d'information des applications pour lesquelles elles sont implémentées ? Nous étudions cette question dans la section suivante. Nous exposerons alors notre problématique ainsi que notre contribution.

Enfin, on peut citer des travaux de recherches menés au CEDRIC dont l'objectif était de séparer MMS de la pile OSI et de l'implémenter sur une pile de communication TCP/IP (Lefebvre, 1995), ou de l'adapter à l'environnement distribué CORBA. Il en résulte une

architecture appelée COOL-MMS proposée par Guyonnet (Guyonnet, 1997). MMS sur CORBA pourrait être la base d'une conception orientée objet des application manufacturières. De plus, elle peut simplifier l'intégration des services de messagerie industrielle dans les architectures de communication hétérogènes du modèle CIM.

4. Problématique

CIM-OSA détermine le contexte dans lequel on peut obtenir des solutions opérationnelles, à partir d'un modèle conceptuel et par dérivations successives. Ainsi, l'infrastructure intégrante associe SQL à ses services d'information, et MMS à ses services de présentation. Pour autant, l'implantation même du système de communication n'en découle pas. Ces services autorisent l'échange d'informations, mais celles-ci ne sont pas clairement identifiées.

Au sens informationnel, le problème semble être résolu, depuis l'analyse conceptuelle, en passant par l'analyse organisationnelle (ou logique), jusqu'à l'implantation. En effet, il existe aujourd'hui de véritables chaînes couvrant tout le cycle de conception d'un système d'information. Des outils basés sur des méthodes d'analyse normalisées ou standardisées (EXPRESS, MERISE, NIAM, ...) garantissent la cohérence des différentes phases d'instanciation jusqu'à la réalisation physique, par l'utilisation de "générateurs" ou "post-processeurs" à destination de Systèmes de Gestion de Bases de Données comme Oracle, Ingres, Informix, Progress, ...

En ce qui concerne les communications au sein d'une base de données distribuée, ces méthodes permettent d'identifier les flux d'information, de quantifier et de qualifier les messages qui les véhiculeront : périodicité, partenaires de l'échange, etc... Au stade de l'implantation, on dispose alors des requêtes (SQL par exemple) qui "animeront" ce système d'information.

Mais on ne peut plus s'inscrire dans ce cadre lorsqu'il s'agit de communications industrielles. En effet, les données manipulées en lecture ou en écriture sont alors des informations "vivantes", hétérogènes, à forte criticité, directement issues ou à destination de l'outil de production (machine, robot, ...). Elles n'ont pas la dimension "off line" qu'elles peuvent revêtir lorsqu'elles existent aussi dans la base de données.

De plus, pour MMS par exemple aucun outil ne permet la définition cohérente des VMD qui décrivent, pour la communication, les équipements constituant l'atelier. L'ingénieur doit créer les VMD, ex nihilo, relativement à l'idée qu'il se fait des besoins en communication de ces équipements. Il se situe exclusivement en phase de réalisation, privé des analyses conceptuelles et organisationnelles qui ont pu précéder son travail. Enfin, aucune méthode ne permet la génération des services qui doivent animer le système de communications industrielles en fonction de la dynamique du système de production.

En outre, l'adéquation avec le système de gestion de base de données n'est pas garantie : une même information peut être typée différemment au sein du VMD et au sein de la base. De graves problèmes de cohérence apparaissent alors quand la base de donnée rafraîchit ses tables en questionnant un VMD, ou lorsqu'un VMD est reconfiguré par lecture de la base.

Des travaux de recherches, menés au sein du CRAN, ont répondu à une partie de telles exigences et nous les présentons maintenant. Ceci sera suivi par une brève présentation des travaux menés au CRIN (Akazan, 1995b) qui traite le problème d'un autre point de vue.

4.1. Travaux antérieurs (Divoux, 1996)

C'est en partant des constats que nous venons de présenter que les travaux de (Rondeau, 1993) ont été initiés visant à implanter physiquement le système d'information et le système de communication à partir de la seule analyse informationnelle. Il est alors nécessaire d'établir des méthodes et outils qui permettent d'obtenir des objets MMS générés par dérivation du modèle logique de données (Figure 9).

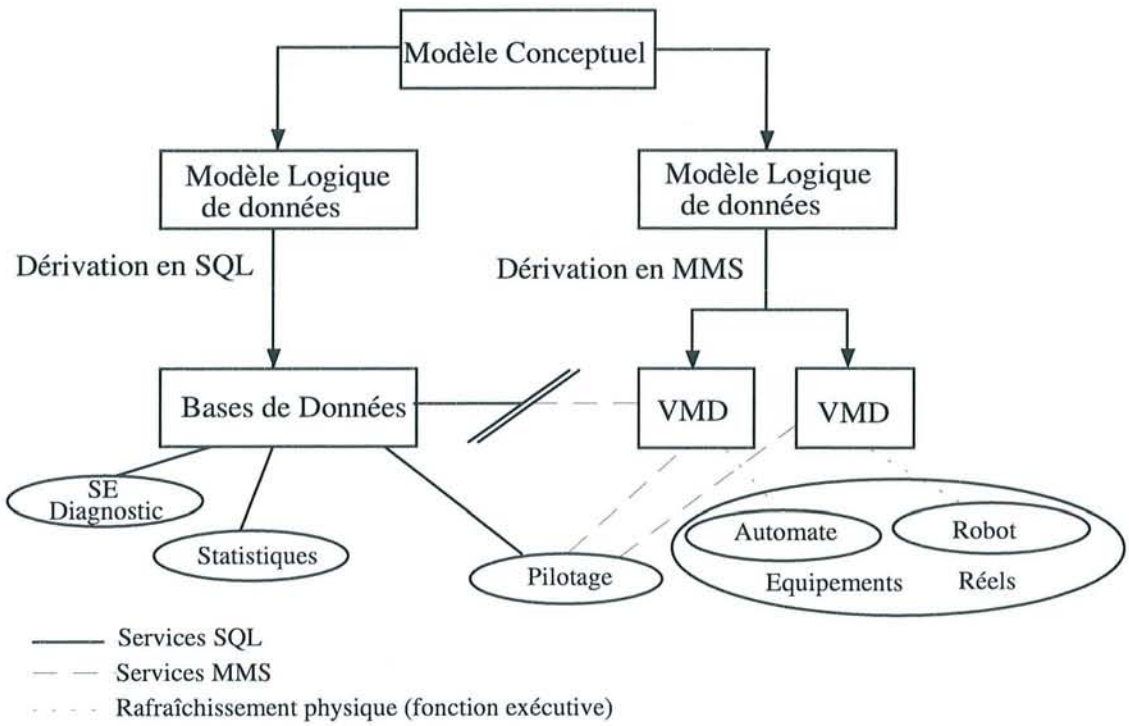

Figure 9. Dérivation SQL et MMS des modèles logiques de données.

Ces travaux s'inscrivent dans une démarche CIM-OSA en utilisant les formalismes recommandés dans le projet. L'étude s'est située donc au niveau de la modélisation de la description de l'implantation (en grisé sur la Figure 10). A ce niveau, la vue information spécifie les structures des bases de données, et la vue ressource la configuration des équipements industriels (adressage des informations, ...).

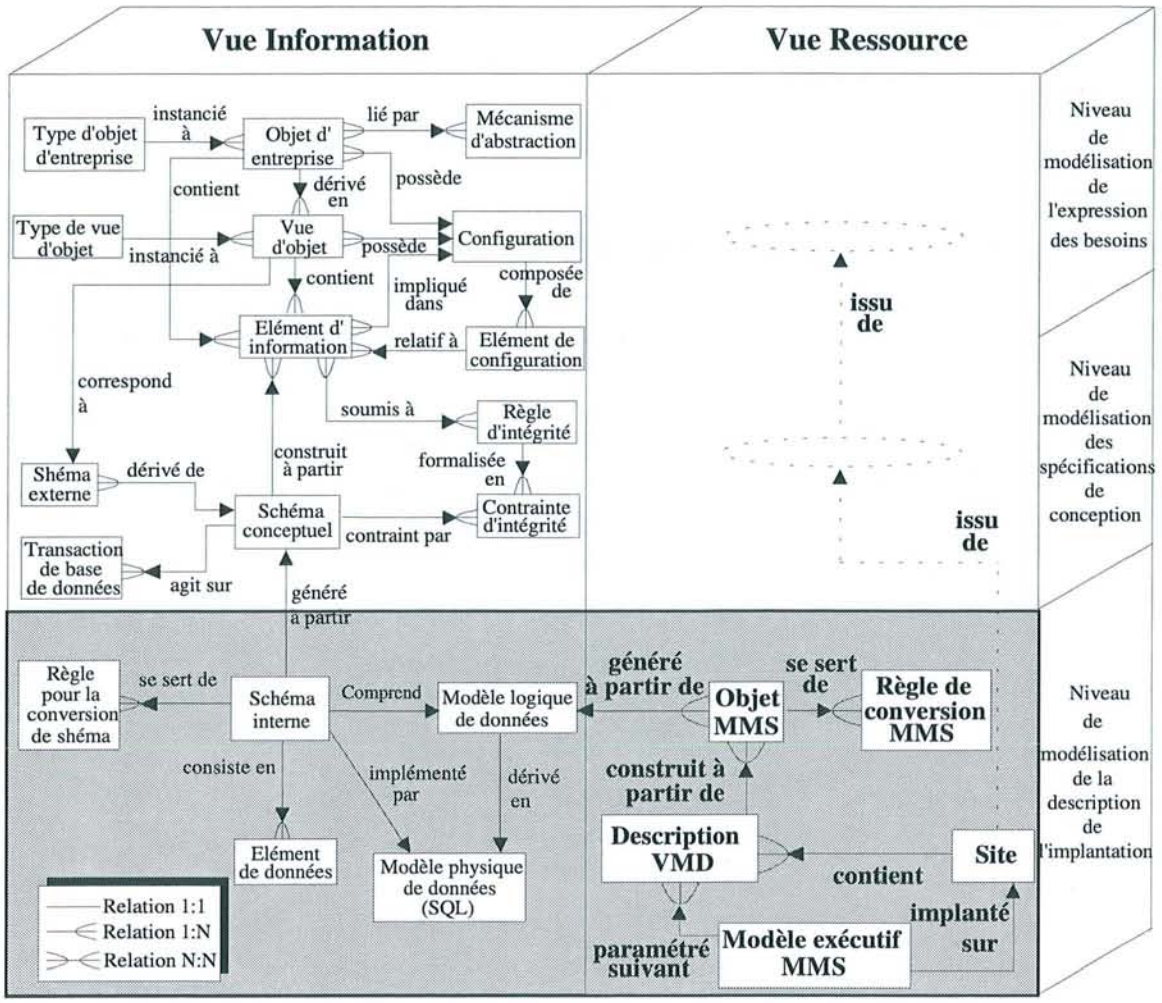

Figure 10. Relation entre la vue information et la vue ressource.

L'exploitation du modèle logique de données pour obtenir une représentation MMS du système de production a donc nécessité la définition d'un traducteur entre deux types de modèles, afin de pallier l'absence de relations entre les vues information et ressource qui sont pourtant logiquement liées.

Cette traduction soulève un certain nombre de problèmes issus des différences de pouvoir descriptif de chacun des modèles. Le passage d'un formalisme à l'autre risque en effet d'induire des pertes de sémantique : certaines représentations dans un modèle ne possèdent pas leur équivalent dans l'autre, et réciproquement. Aussi doit-on être conscient que tout n'est pas forcement a priori traduisible. Une proposition d'une méthodologie de conception d'un traducteur a été présentée et ensuite appliquée à la définition de deux traducteurs spécifiques :

- Entité-relation / MMS,
- EXPRESS / MMS.

Le premier est naturel dans le cadre de modélisation CIM-OSA dans lequel les travaux se situent. CIM-OSA recommande pour décrire la vue information l'utilisation de modèles entité-association et plus particulièrement la méthodologie M* (Vernadat, 1989). Au niveau de la description de l'implantation, le schéma conceptuel et ses schémas externes sont

exprimés uniquement en langage "à la SQL" (Standard Query Language) pour former le schéma interne du système d'information. En ce qui concerne la vue ressource, la messagerie industrielle MMS a été retenue pour la représentation et l'accès aux différents équipements de production communicants.

Ensuite le langage EXPRESS, normalisé et largement utilisé, a été choisi pour étudier le système d'information. Bien que relativement jeune, il semblait être plus consensuel et internationalement utilisé que l'était le formalisme entité-relation. Ce dernier, bien que recommandé dans un cadre CIM-OSA, n'est réellement connu qu'en Europe. Son avenir peut en souffrir ...

De plus, le choix de EXPRESS était basé sur le fait qu'il s'impose comme interface de référence entre de nombreux formalismes. De part ses propriétés de langage inter-métiers, il est cible de nombreux traducteurs ayant pour origine NIAM (Nijssen Information Analysis Method) (Habrias, 1988), IDEF1x, des méthodes objet, ou même des formalismes entité-relation classiques (Figure 11).

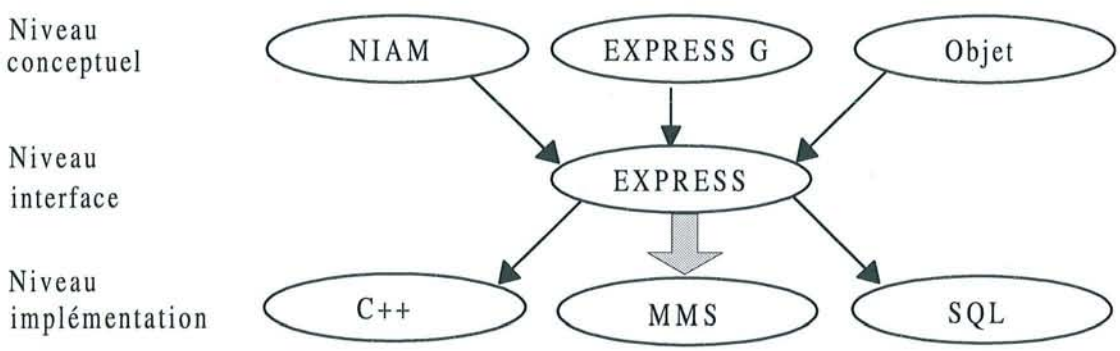

Figure 11. EXPRESS utilisé comme interface de référence.

Dans les deux cas (Entité-relation / MMS, EXPRESS / MMS), la configuration physique du système de communication, c.à.d la structure des VMD a été déterminée. Seule la dimension statique a été maîtrisée et ainsi uniquement un des deux volets de la norme MMS a été couvert.

Des intérêts considérables ont été soulignés par le fait de générer le système de communication à partir de l'étude du système d'information :

- harmonisation les données du système de communication et de la base de données. En effet, dans les deux cas, les données doivent avoir le même nom, le même type et naturellement la même signification. Pour cela, il est important qu'une même étude génère à la fois la base de données et les objets MMS, évitant ainsi tout conflit lorsque la base de données veut accéder à la valeur d'une donnée implantée dans un VMD et vice-versa. De plus, on garantit la cohérence permanente entre système d'information et système de communication. En effet, toute une modification du premier entraîne une évolution automatique du second,

- optimisation du système d'information. En effet, il n'est pas rare actuellement que la base de données supporte des informations volatiles juste pour les rendre plus facilement accessibles. Le modèle logique de données, suivant notre démarche,

sépare les informations à stocker au sein de la base de données (pour un système expert, un historique, des calculs statistiques, etc..), et les informations utilisées pour leurs valeurs courantes implantées uniquement dans des VMD,

- gain de temps en conception. L'étude du système d'information et celle du système de communication ne font plus qu'une. De plus, l'intégrateur n'utilise qu'une méthode sans connaître précisément la norme MMS.

Les travaux de Akazan ont visé la réalisation d'un outil baptisé MMS-ToolKit (Akazan, 1995b) ayant comme vocation la génération des objets et services MMS à partir d'une spécification fonctionnelle d'une application temps-réel et répartie. Le modèle fonctionnel, avec lequel l'application temps-réel est structurée, est basé sur les concepts suivants :

- tâches représentant l'élément de décomposition d'une application en unités fonctionnelles,

- règles de synchronisation entre les tâches fonctionnelles, basées sur les événements et sur les conditions portant sur l'état du système, montrant la manière dont les tâches interagissent pour accomplir l'objectif assigné à l'application,

- flux de données (ou variables d'application) qui permettent la communication entre les tâches.

La spécification fonctionnelle est exprimée dans un langage naturel (une grammaire) et est représentée en notation BNF (Backus-Naur Form). Une supposition avancée : la spécification de l'application en composants (tâches, règles de synchronisation et flux de données) est faite au préalable et on suppose qu'elle est correcte. Concernant la répartition de l'application sur les sites cibles, elle est décrite par le même langage que celui utilisé pour décrire la spécification fonctionnelle. Une étude des composants de la spécification fonctionnelle et MMS est faite en vue de déterminer les équivalences sémantiques entre eux. Ceci a pour but de faire le passage du modèle fonctionnel aux objets et services de communication nécessaire pour la réalisation de l'application étudiée.

Le passage du modèle fonctionnel à MMS est basé sur des règles de transformation portant sur chaque composant ou objet fonctionnel. Par exemple, une tâche fonctionnelle est transformée en un objet MMS invocation de programme (program invocation) résidant sur le même site que la tâche à partir de laquelle il est dérivé. Une validation en partie syntaxique a été également envisagée.

Concernant les services MMS générés, l'outil spécifie le nom du service , le site client où sera implanté l'appel au service, le serveur et l'objet MMS sur lequel se porte le service. Les paramètres réels d'appel à un service seront dépendants du produit MMS utilisé.

4.2. Objectif

Les travaux d'Akazan, ont visé le côté contrôle-commande du système automatisé de production et par conséquent le côté gestion de données techniques n'a pas été pris en

compte. C'est en partant de cette constatation et dans le souci de continuer les recherches déjà menées dans notre équipe (E. Rondeau) que nos travaux ont été initiés. Notre démarche vise une optique plus large et s'appuie sur un cadre méthodologique tout en s'intéressant aux aspects structurel (informationnel), comportemental et fonctionnel du système automatisé de production, en prenant en considération le deuxième volet de la norme MMS qui concerne les services (Figure 12). Notre objectif est d'identifier les services à utiliser, et les objets qu'ils manipuleront. Nous devons aussi définir leur séquencement, leur occurrence, qui relativement au système de pilotage assurent la dynamique du système de communication, donc celle du système de production.

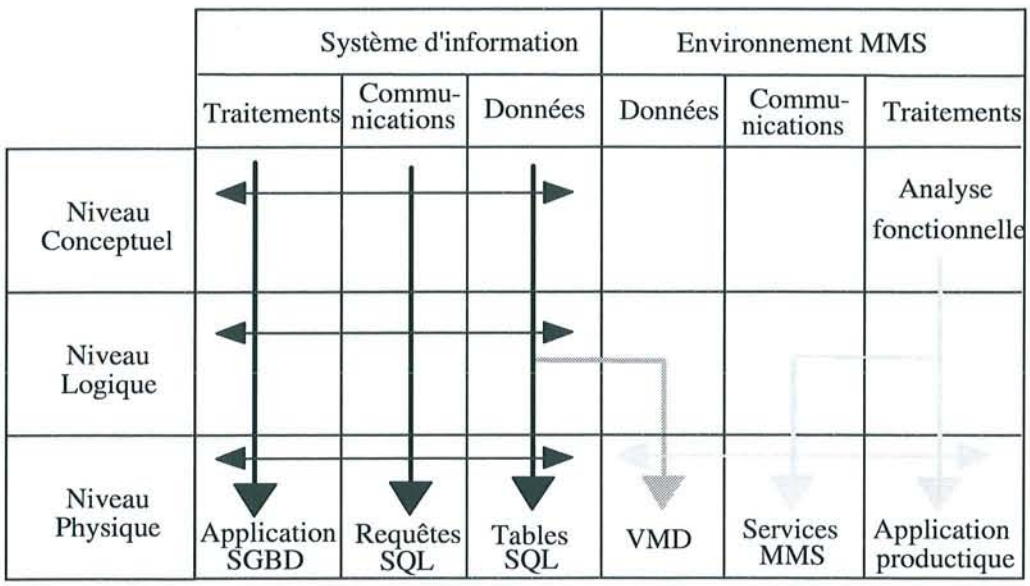

Figure 12. Intégration des communications industrielles MMS.

4.3. Démarche

Le point de départ pré-suppose un système de pilotage bien défini. Ainsi, notre démarche consiste à assister le concepteur ou l'intégrateur dans sa mission et ceci dès la phase d'analyse (éventuellement détaillée) de son système en sa totalité (structure, comportement, fonctions) sans qu'il se soucie de MMS. Il est à noter qu'une intervention de sa part est néanmoins nécessaire pour la répartition de son application (répartition des objets issus du modèle objet).

Pour dériver du modèle objet OMT du système à développer, le modèle objet MMS, une interface basée sur des règles de transformation doit être définie. Les règles de transformations seront présentées dans le chapitre suivant. Cette interface va définir le passage ou la mise en correspondance des concepts du modèle objet OMT en concepts du modèle MMS (classes, objets, attributs, associations, agrégations, opérations,...). Le résultat de ce passage et par conséquent la structure des VMDs sera contrainte par les classes de conformité (CC ; les services MMS réellement implémentés sur l'équipement) des équipements qui leur sont associés (Figure 13)

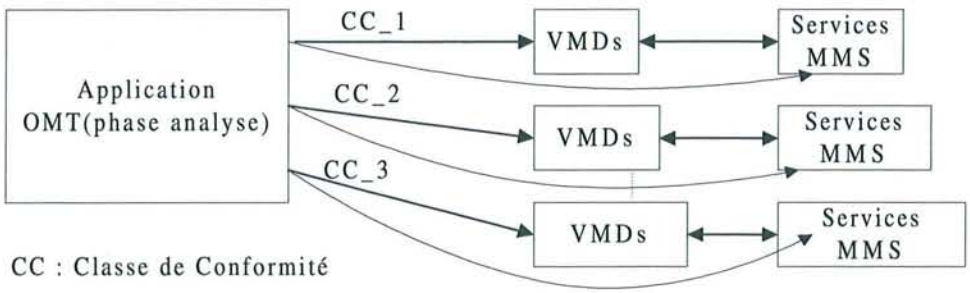

Figure 13. Processus suivi pour la génération du système de communication.

Dans un but de réutilisation des diagrammes d'états concernant le comportement, du point de vue de la communication, il faut trouver des sous-diagrammes (automates) réutilisables. Ceci consiste à identifier des sous-ensembles stables d'états. Un tel ensemble est constitué d'états et d'événements tels que, quel que soit l'événement de l'ensemble, son traitement fait rester dans un des états de l'ensemble. Afin que ce sous-automate soit utilisable, il faut le compléter par l'indication d'événements qui permettent d'en sortir.

Ces composants automates sont très utiles lors de la description des services dans les différents contrôleurs d'équipements ou ceux des utilisateurs puisque certains services peuvent être invoqués par plusieurs clients (application ou utilisateur). Donc on obtient une bibliothèque de composants automates. On construit fondamentalement ces composants dans le but d'être utilisés comme briques de base dans le développement. On peut procéder à ceci pour décrire le comportement du client MMS pour obtenir des scénarios possibles qui dépendent de la classe de conformité du client. Cette bibliothèque va être liée aux programmes sources durant la phase d'édition des liens et ces composants sont en relation avec les fonctions définies dans le produit implémentant MMS.

Nous définissons des automates réutilisables pour l'allocation d'une ressource, le contrôle des tâches et la supervision avec leur projection en services MMS (dépendant des classes de conformité). Ainsi, le client MMS va pouvoir choisir le scénario adéquat décrivant le mieux le séquencement des services (ou des demandes de services) à suivre pour satisfaire ses besoins et la classe de conformité des équipements. Ces aspects seront étudiés dans le chapitre 3.

Mais une constatation peut déjà être faite, après un approfondissement de la méthode OMT et du protocole MMS : Il n'y a pas de limites pour les opérations applicatives pouvant manipuler les classes d'objets OMT alors que les services MMS manipulant les objets MMS sont limités. Par exemple, pour un objet MMS de la classe Variable, il y a un ensemble de services définis pour le manipuler tels que : Read, Write et Information Report, ... Donc, il y a toujours le risque de pertes sémantiques à toute traduction. C'est le cas pour les objets applicatifs également, puisque les objets MMS sont bien connus et déterminés dans la norme (variable, domaine, programme, ...), alors que ceci ne l'est pas dans OMT.

Ces remarques concluaient les travaux antérieurs menés dans l'équipe. Deux questions avaient également été soulevées, pour lesquelles nos travaux apportent des solutions :

- Les objets générés par notre méthode seront-ils tous supportés par l'équipement considéré ?

- La partie des services disponibles sera-t-elle suffisante pour gérer les différents objets MMS issus de la traduction ?

Pour répondre à ces questions, en prenant en considération les classes de conformité de chaque classe d'objets traduite par un VMD, les objets constituant la structure d'un VMD seront supportés par l'équipement considéré et par conséquent les services disponibles seront suffisants pour les gérer.

Il est à noter que si une classe d'objet peut être traduisible par deux objets MMS de nature différente, donc manipulée par deux services différents (le cas où on peut imaginer deux éventualités possibles de la classe de conformité de l'équipement modélisé par cette classe d'objet), le choix entre un service ou un autre influencera nécessairement le comportement de l'application. Cette influence au niveau de la commande ou de la supervision est généralement sans conséquence au niveau du procédé.

4.4. Choix de la méthode et les outils associés

En ce qui concerne la vue fonctionnelle et pour les deux niveaux de modélisation (conceptuel, logique), il existe un ensemble d'outils de modélisation déjà éprouvés. Alors qu'au niveau physique (l'implémentation et la génération effective de code), de nombreuses solutions sont envisageables. L'outil retenu doit offrir une bonne ergonomie, une grande indépendance vis-à-vis des matériels supports de l'exécution, donc une bonne portabilité en permettant une programmation hiérarchique et parallèle.

Donc, nos efforts se sont focalisés sur la recherche d'une méthode et de l'éventuel outil associé, qui permettent réellement et d'une façon parfaitement cohérente, de décliner successivement les trois niveaux de modélisation jusqu'à l'implémentation.

Nous nous sommes intéressés à l'approche orientée objet, avec tous les avantages qu'elle présente et parce que son utilisation est de plus en plus admise dans le monde industriel :

- Menga et al (Menga, 1989) ont proposé le paradigme client-serveur-service, supporté par la programmation orientée objet, pour la conception et le prototypage des systèmes manufacturiers (MS ; Manufacturing System). L'analyse ascendante (bottom-up) des modèles de référence standards de tels systèmes a abouti à la définition d'une librairie de classes de base supportant la méthodologie de conception descendante (top-down). L'utilisation du concept d'héritage donne la force à cette méthodologie. La caractéristique de l'approche proposée est d'offrir une vue uniforme des différents niveaux de l'architecture de contrôle qui conduit à une bonne réutilisabilité du logiciel et à une implémentation simple des spécifications.

- Jochem et al (Jochem, 1989) de l'Unité de Recherches CIM de l'IPK (Institute for Production Systems and Design Technology; Berlin), ont contribué à un projet en collaboration avec Digital Equipment. Ce projet avait comme objectif de développer une méthodologie satisfaisant aux besoins spécifiques d'analyse et de conception des systèmes CIM en utilisant une approche orientée objet et également de l'implémenter dans un outil logiciel. La méthodologie développée, appelée MOOD (Modelling

Object Oriented Design), s'intéresse à la modélisation de la transaction de l'ordre ou de la commande à partir de son entrée via la planification de la production et le contrôle jusqu'aux systèmes manufacturiers et d'assemblages de l'atelier. En modélisant des interfaces entre ces différentes tâches, une dérivation de la spécification du logiciel est attendue. Ainsi, MOOD est perçue comme une combinaison d'outils et de techniques employés dans un cadre organisationnel qui peut être appliquée aux projets de développement successifs de systèmes logiciels.

- OOSIM (Object-Oriented Simulation in Manufacturing), est une approche orientée objet pour modéliser et simuler les systèmes manufacturiers. Elle a été développée à l'Institut Technologique de Georgia (Govindaraj, 1993) (Bonder, 1993) (Narayanan, 1992). Ces travaux ont mis en évidence les problèmes survenus lors de l'utilisation des méthodes analytiques de simulation. Le problème fondamental est la modélisation en utilisant les abstractions offertes par un langage de simulation pour décrire le système à analyser. Les langages de simulation traditionnels sont basés sur les abstractions des réseaux de files d'attente La difficulté dans la simulation d'une modélisation des systèmes manufacturiers est l'absence d'une telle description. OOSIM offre la possibilité de reconfigurer, dynamiquement l'usine pendant l'exécution et utilise un seul objet pour représenter des différents équipements en spécifiant différents comportements d'équipements dans l'objet. Sachant que le système de contrôle (i.e. contrôleur de cellule) est modélisé séparément des ressources matérielles (i.e. machines). De plus, OOSIM permet des simulations interactives et dites temps-réel et elle a été appliquée sur des systèmes manufacturiers complexes tels que les FMS (Flexible Manufacturing Systems) et la fabrication des circuits intégrés ICs (Integrated Circuits).

- Luttenbacher (Luttenbacher, 1995), a contribué à l'ébauche d'une méthodologie pour la formalisation d'un modèle de référence pour un capteur intelligent, basé sur une approche objet. Les cas d'utilisation ont été établis pour la formalisation des spécifications où les fonctionnalités du capteur intelligent sont définies du point de vue de l'utilisateur. Quant à la description des objets issus des cas d'utilisation, la méthode OMT a été employée pour la représentation des différentes vues : La vue informationnelle, la vue architecturale et la vue comportementale.

- Fofana (Fofana, 1996) (Fofana, 1997) a utilisé la méthode OMT pour l'analyse de la commande d'un poste d'usinage et pour la spécification et le prototypage d'un système de conduite technique d'atelier. Ce système permet aux sous-traitants de grandes entreprises de gérer rigoureusement les données techniques qu'ils manipulent au cours du cycle de production et il permet également une aide à la décision des différents opérateurs (méthodes, usinage, gestion d'outils, ...). Le système a été spécifié avec la méthode OMT et un ensemble de modules tels que le module de préparation des gammes de fabrication, le module d'usinage, le module de contrôle qualité et les modules de gestion d'outils et de bruts ont été définis par une décomposition du système en modules.

- Zaytoon et al (Zaytoon, 1995), ont proposé une extension de la méthode OMT au génie automatique. Cette extension est basée sur l'introduction d'une couche sémantique autour du modèle fonctionnel d'OMT et sur une méthodologie d'assistance et d'élaboration de ses trois modèles pour la phase de spécification d'un

système automatisé de production. Quelques considérations sur l'application d'OMT ainsi étendue aux ateliers flexibles, et plus particulièrement aux problèmes d'allocation des tâches, sont présentées.

- Enfin, Wu (Wu, 1995) a proposé HOOMA (Hierarchical and Object Oriented Manufacturing systems Analysis), une méthode d'analyse hiérarchique et orientée objet des systèmes manufacturiers, qui utilise les concepts des deux méthodes d'analyse orientées objet HOOD et OOA. Elle est développée spécifiquement pour supporter l'analyse et la définition des besoins des systèmes dans le contexte manufacturier. De plus, pour la spécification du processus manufacturier en terme d'objets, HOOMA permet la prise en compte de la structure intrinsèquement hiérarchique et des aspects dynamiques des opérations manufacturières.

Par conséquent, nous avons adopté la méthode orientée objet OMT (Object Modelling Technique), relativement aux avantages déjà cités et parce que MMS relève d'une représentation objet. Nous disposons de l'un de ses outils associés (Lov-OMT commercialisé par la société Verilog). Cet outil, sélectionné par différentes équipes du CRAN, consiste en un package de trois éditeurs (LOV, 1995) : éditeur objet (Object Editor), éditeur de comportement (Behavior Editor), éditeur de fonctions (Functional Editor). Un organisateur de projets (Project Organiser) assure leur cohérence mutuelle et permet à plusieurs utilisateurs de travailler sur un même projet. L'outil dispose également de deux générateurs de code SQL/C++.

5. Conclusion

Nous avons présenté un survol de quelques travaux menés pour la réalisation d'un outil d'assistance aux concepteurs des applications industrielles visant l'utilisation du protocole de communication MMS.

Nous avons remarqué les intérêts de suivre la ligne tracée par les travaux déjà réalisés dans notre équipe et de pouvoir les enrichir et les étendre par l'expérience menée par d'autres travaux. Ainsi notre problématique a été définie, qui est la génération automatique du système de communication en sa totalité : La configuration des sites MMS d'un système automatisé de production (la définition des VMDs) et la génération des services, à partir de la seule étude du système d'information (la prise en considération de la vue structurelle et fonctionnelle ou comportementale).

Le deuxième traducteur élaboré (EXPRESS/MMS) avait, à part les avantages énoncés plus haut, des inconvénients tels que le cumul des pertes sémantiques à travers deux conversions successives (Figure 11). Ce dernier constat nous a amené à rechercher le moyen efficace pour un passage en une seule fois entre une méthode de modélisation, de préférence une méthode objet telle que OMT, et le modèle MMS (Figure 14).

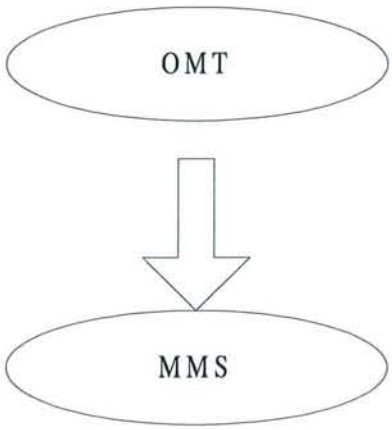

Figure 14. Passage d' OMT au modèle MMS.

La figure 15 montre la démarche globale que nous adaptons pour générer les services MMS à mettre en œuvre pour la réalisation d'une application donnée.

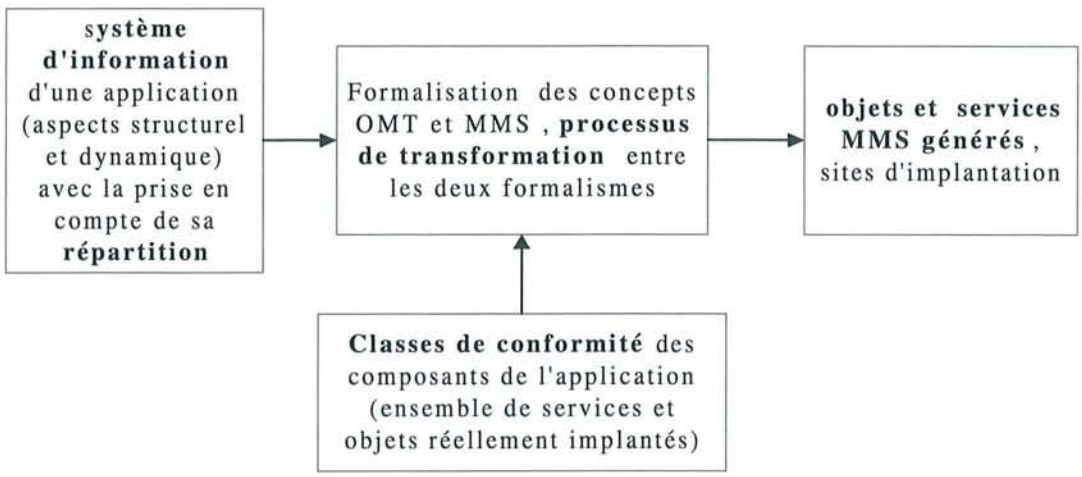

Figure 15. Etapes du Passage d'OMT au modèle MMS

Pour ce faire, nous nous sommes inspirés des travaux effectués par Gargouri et al qui consistaient à définir un modèle générique (MGCO2 ; Modèle Générique de Conception Orientée Objet) qui joue le rôle d'une interface pivot entre une modélisation conceptuelle orientée objet et les différents environnements de développement. Nous avons repris la même démarche et également leur formalisme pour assurer le passage du modèle OMT au modèle MMS. Nous le présentons dans le chapitre suivant.

Du système d'information au système de communication par une formalisation unifiée

1. Introduction

Le système d'information d'une organisation (au sens large : société, entreprise, institution, groupe structuré, ...) regroupe tout ce qui représente, stocke et traite des informations relatives à l'organisation concernée. Le système d'information est la mémoire de l'organisation. L'aspect statique de cette mémoire est d'enregistrer des faits, des règles et des contraintes. L'aspect dynamique de cette mémoire est la mise à jour des données, règles et contraintes.

Dans un contexte concurrentiel de plus en plus vif les entreprises recherchent un accroissement de flexibilité, d'adaptabilité et de réactivité. De nouveaux types d'organisations apparaissent (entreprise réseau, entreprise étendue, entreprise virtuelle) tous caractérisés par une distribution accrue basée sur des systèmes d'information répartis. Comme nous l'avons vu dans le chapitre précédent, l'approche objet s'adapte mieux à l'ingénierie ou à l'analyse et la conception de tels systèmes d'information, que les méthodes traditionnelles.

Nous présentons maintenant des travaux qui ont contribué à une normalisation des méthodes orientées objet pour aboutir à un modèle générique les unifiant.

2. Contribution à la standardisation des méthodes objets

Des travaux de recherche ont visé à la définition d'une interface spécifique avec des règles de transformation des représentations sources vers différents environnements d'implémentation. Ceci avait comme objectif de réduire le fossé sémantique existant, entre les niveaux conceptuel et physique, et induit par la différence de formalisme propre à chaque niveau. Parmi ces travaux, on peut en citer quelques-uns :

2.1. Interface Orientée Objet O2I (Object Oriented Interface)

Dans (Kraiem, 1993), Kraiem et al ont proposé une interface appelée O2I (Object Oriented Interface). Cette interface est supportée par un outil logiciel, basée sur un modèle pivot et un ensemble de règles de transformation. O2I comble le fossé existant entre les niveaux conceptuel et physique. Les méthodes de conception utilisées au niveau conceptuel sont O* (Rolland, 1993) et MCO (Pouillat, 1993). Le modèle pivot est un modèle de classe avec une structure générique composée de variables, contraintes et méthodes.

O2I s'appuie sur un langage pivot orienté objet (O2IL ; Object Oriented Interface Language). Ce dernier est considéré comme un ensemble de langages d'implémentation orientés objet. L'interface O2I contient une collection de règles de transformation propre à chaque transformation, i.e. de O* ou MCO en environnement de programmation orientée objet (ONTOS/C++ et Eiffel). Des règles R_c transforment la modélisation conceptuelle en spécification pivot.

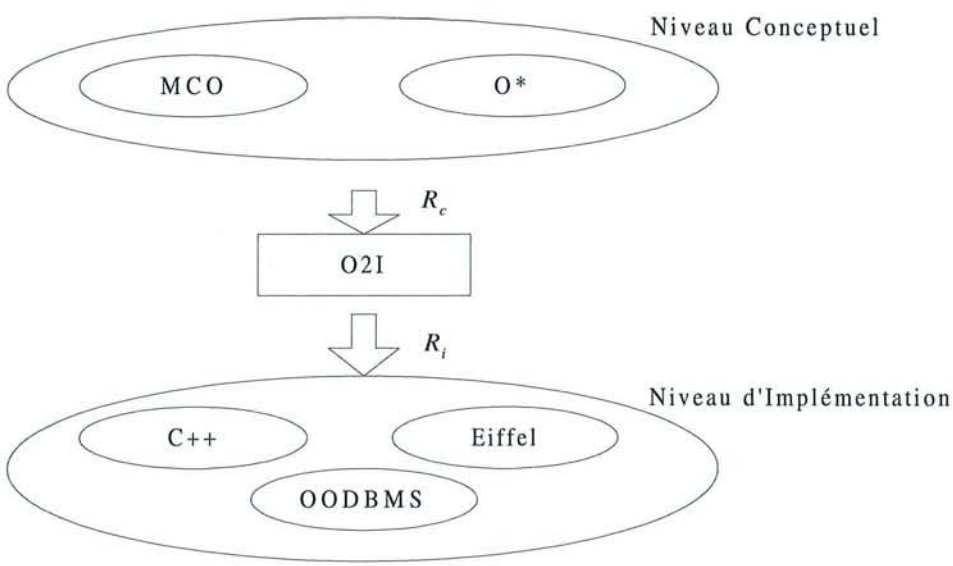

Figure 1. Présentation de l'interface O2I.

Des règles R_i transforment la spécification pivot en environnement d'implémentation (Figure 1). Deux alternatives sont possibles : soit le modèle conceptuel et le langage de destination font parties de l'interface O2I ; dans ce cas la transformation s'effectue automatiquement, soit un des deux ou les deux sont inconnus ; et dans ce cas les règles R_c et R_i doivent être définies.

2.2. De l'Objet au Relationnel OtoR (Object to Relational)

Afin d'allier les avantages des méthodes orientées objet au niveau conceptuel et les bases de données relationnelles au niveau physique, les travaux de Boufarès et al (Boufarès, 1994) ont été initiés. Ces travaux se distinguent, par rapport à d'autres travaux réalisés dans la même optique (Blaha, 1988), par le fait qu'ils génèrent les schémas relationnels optimaux puisqu'ils intègrent les aspects statiques et dynamiques d'un système d'information. Ainsi les schémas issus seront complets et n'auront pas besoin d'être à reconsidérer dans l'étape d'implémentation. O* est la méthode d'analyse et de conception orientée objet considérée au niveau conceptuel.

De O* à RDBMS (Relational Data Base Management System), une interface OtoR (Object to Relational) a été définie. Cette interface est basée sur un modèle pivot et un ensemble de règles de transformation (Figure 2). Le choix du schéma optimal d'une BD (Base de Données) se base sur des fonctions d'évaluation du coût de l'implémentation des relations structurelles (relations de composition, relations de références). Ces fonctions dépendent du type de l'opération utilisée, de la fréquence de l'opération et de la taille moyenne de la relation durant le cycle de vie de l'application.

Un mécanisme de transformation des représentations conceptuelles OMT vers une implémentation relationnelle a été présenté dans (Boufarès, 1996). Donc, grâce au couplage des deux approches : l'orientée objet et le relationnel, la conception devient plus intuitive et plus expressive et l'implémentation plus performante et plus cohérente.

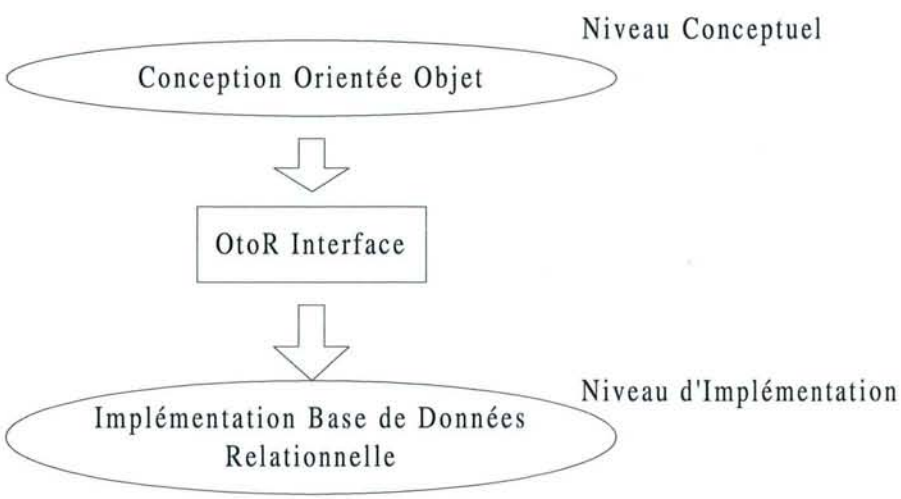

Figure 2. L'interface Object-to-Relationnel

2.3. Modèle Générique pour la Conception Orientée Objet des systèmes d'information (MGCO2)

Tous les travaux que nous venons de présenter ont été menés séparément. D'où l'idée d'unifier toutes les représentations ou méthodes d'analyse et de conception orientée objet en un modèle unique et générique. Les travaux de Gargouri et al (Gargouri, 1995b) (Gargouri, 1997), effectués au laboratoire LIASI, ont visé ce but et ont abouti à une représentation du

modèle générique appelé MGCO2 (Modèle Générique de la Conception Orientée Objet). Ce modèle unifie les concepts des méthodes d'analyse de conception et est indépendant de celles-ci (Figure 3).

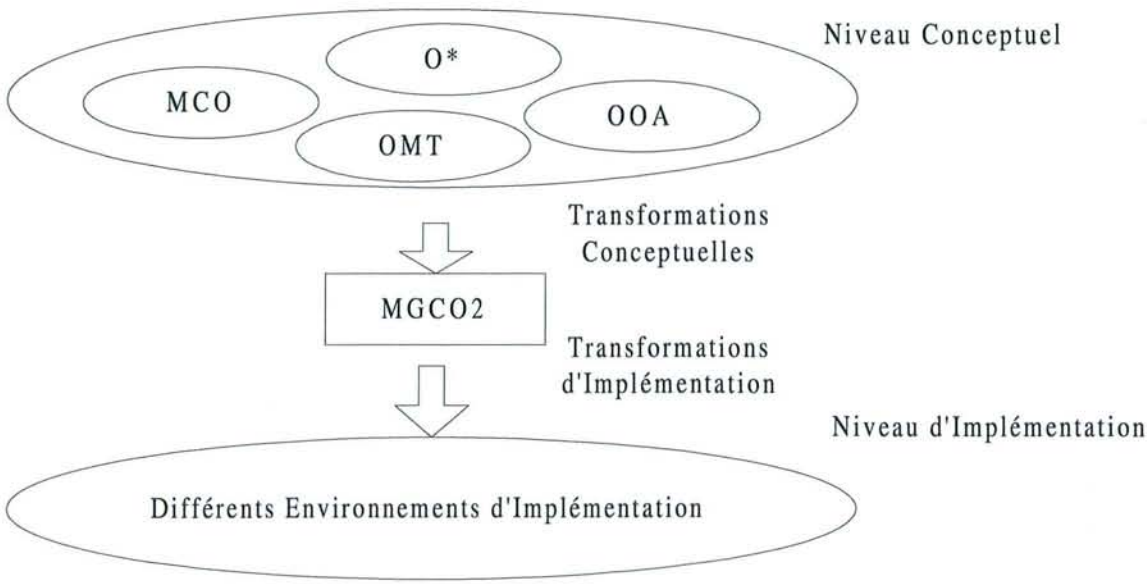

Figure 3. Le modèle générique MGCO2.

MGCO2 permet d'avoir une seule référence pour toutes les méthodes d'analyse et de conception orientée objet, et la transformation d'une application aux différents environnements d'implantation devient automatique. Ce modèle, comme dans toute méthode objet, définit deux modèles, statique et dynamique. Le premier est une représentation statique de l'univers du discours décrivant ses composants et les relations qui les lient. Le second représente les interactions entre les entités du modèle statique et leurs interactions avec l'environnement extérieur.

Objet, classe et propriétés sont les concepts de base du modèle statique de MGCO2. Ils sont équivalents aux concepts inhérents à l'approche objet.

Un objet a une identité unique, appartient à une classe et a un cycle de vie représenté par l'ensemble des événements, survenant depuis sa création et jusqu'à sa destruction.

Une classe regroupe un ensemble d'objets ayant une structure (propriétés) et un comportement communs. Une propriété est définie par une signature composée d'un nom et d'un domaine (type). Le domaine peut être simple (entier, booléen, réel, ...), structuré (énumération, agrégation d'autres propriétés) ou le nom d'une autre classe (représente les relations d'inclusion et d'utilisation existantes entre la classe et les autres classes du système d'information). La définition des relations entre les classes d'objets est basée sur la relation entre leurs cycles de vie. Trois types de relations (statiques) existent : inclusion, utilisation et héritage.

Nous venons de présenter rapidement les concepts du modèle statique du modèle générique MGCO2. Les concepts du modèle dynamiques sont : événements, actions et graphes d'états.

Un événement décrit un ensemble similaire d'occurrences. Une occurrence est une projection d'un fait du monde réel qui influence le système d'information. Une seule occurrence est possible à un instant donné. Un événement peut être externe, temporel ou interne. Il déclenche une ou plusieurs actions sur un ou plusieurs objets du système étudié. Une action est une opération déclenchée en réponse à l'arrivée d'un événement et elle décrit le comportement de l'objet. Un graphe d'états est un graphe ayant comme nœuds les états potentiels et comme arcs les actions associées aux événements du cycle de vie de l'objet considéré. A l'arrivée d'un événement, le nouvel état de l'objet dépend de son état actuel et de la nature de cet événement. Un objet peut changer, d'un état à un autre, durant son cycle de vie. Une procédure assure les changements d'états de l'objet, Change(), a comme paramètres :

- l'état initial (avant l'arrivée de l'événement),
- le nom de l'événement,
- le nouvel état de l'objet (après la réalisation de l'événement).

Des fonctions représentant la transformation des représentations conceptuelles en modèle MGCO2 ont été définies.

Dans (Gargouri, 1995a) Gargouri et al ont abordé, en utilisant le modèle MGCO2, la transformation des différentes représentations conceptuelles vers différents environnements d'implantation parmi lesquels l'environnement relationnel. Un ensemble de règles de transformation, dites des règles d'implantation, ont été définies (Figure 4).

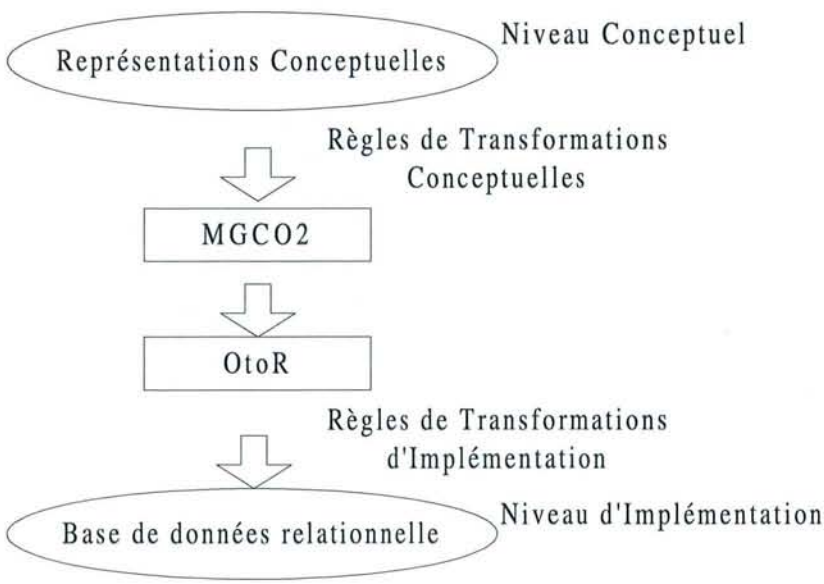

Figure 4. Le passage des représentations conceptuelles vers l'environnement relationnel via MGCO2.

Dans la suite nous présentons notre contribution pour une transformation des représentations conceptuelle OMT vers le modèle MMS.

3. D'OMT à MMS

Dans une première étape, les travaux menés dans notre laboratoire ont consisté à traduire directement une représentation dans le modèle Entité-Relation (adapté par la méthode Merise) en une représentation dans le modèle MMS (Rondeau, 1993). Cependant cette démarche, qui s'apparente à celle suivie par (Kraiem, 1993) (Boufarès, 1994, 1996), est spécifique à un unique modèle de représentation source (Chapitre 1, Section 4.1, Figure 12).

La deuxième étape de cette étude a alors été de s'appuyer sur un langage pivot interface de plusieurs modèles et plus particulièrement sur le langage EXPRESS (Divoux, 1997). En effet le langage de modélisation EXPRESS dans de nombreux travaux de recherche (Sanderson, 1993a, 1993b) (Morris, 1990) (Griffin, 1993) joue ce rôle.

L'inconvénient d'une telle approche est d'introduire une double traduction :

- du modèle de représentation source vers le langage pivot,
- du langage pivot vers la représentation destination MMS.

Cette double traduction peut cumuler les pertes sémantiques.

Puisque notre contribution est de définir le système de communication (configuration des sites MMS 'configuration des VMDs') à partir de la seule étude du système d'information, nous avons essayé de trouver un formalisme de modélisation ayant comme vocation la transition (ou la transformation) entre les deux niveaux de modélisation conceptuel et physique. Le système d'information est supposé être analysé et conçu par une approche objet. En occurrence c'est la méthode OMT.

Par cette démarche nous aurons une implémentation du système d'information, et en même temps une implémentation MMS. Mais rappelons que les formalismes utilisés aux deux niveaux d'abstraction sont différents. C'est pourquoi et à partir des constats cités plus haut, nous pouvons dire que nos travaux s'inscrivent bien dans la même voie que celle présentée dans la section précédente et s'apparentent plus particulièrement aux travaux menés par Gargouri et al. Nous utilisons leur formalisme défini pour le modèle générique MGCO2 dans la représentation des concepts d'OMT et ceux de MMS (Mansour, 1998a, 1998b, 1998c). Ensuite et par l'intermédiaire de règles de transformation, nous montrons le passage d'un formalisme à l'autre (Figure 5).

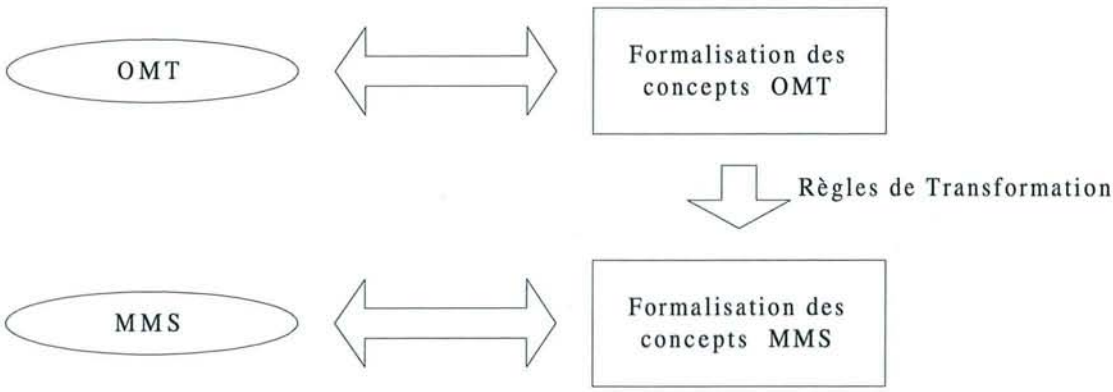

Figure 5. Le passage des représentations OMT vers des représentations MMS.

3.1. Formalisation d'OMT

OMT est une méthode de modélisation orientée objet organisée autour des concepts du monde réel. Elle modélise un système selon trois points de vue dépendants mais différents : modèle objet, modèle dynamique et modèle fonctionnel (Figure 6).

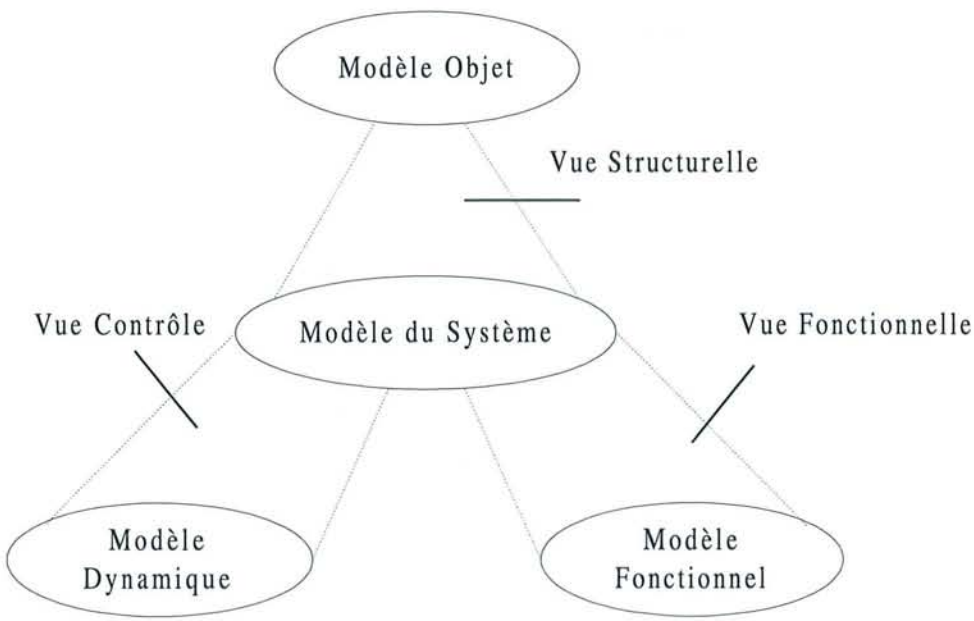

Figure 6. Les différents aspects et les modèles associés d'un système.

Le modèle objet donne une vision globale de la structure d'un système en terme d'objets et de relations qui les relient. Il fournit le cadre dans lequel les deux autres modèles s'insèrent. Il est représenté graphiquement par des diagrammes de classes.

Le modèle dynamique donne une vision globale du comportement du système et est représenté graphiquement par des diagrammes d'états étendus et des diagrammes de suivi d'événements (ou des scénarios).

Quant au modèle fonctionnel, il donne une vision globale des fonctions du système et est repésenté par les diagrammes de flots de données.

OMT couvre l'ensemble du cycle de vie du logiciel (exception faite des parties «test» et «maintenance») partant de l'analyse à l'implantation à travers la conception système et la conception objet. Durant la phase d'analyse, un modèle du domaine d'application est constitué, qui est une abstraction concise et précise du Qui et du Quand.

Les choix techniques d'implémentation du système d'objets se font durant l'étape de conception du système. Le modèle objet issu de l'étape d'analyse sera enrichi par les détails d'implémentation, en accord avec la stratégie définie dans l'étape précédente. A la phase d'implémentation le modèle objet sera transcrit dans un langage de programmation et/ou un SGBD (Système de Gestion de Base de Données).

3.1.1. Définitions préliminaires

Nous définissons les notations (Tableau 1) et nous allons focaliser notre travail uniquement sur les concepts essentiels de la méthode OMT. Nous en donnons une définition sous forme de fonctions mathématiques en reprenant les formalisations effectuées par Gargouri et al (Gargouri, 1995a, 1995b) (Gargouri, 1997).

Cl_{diag}	diagramme de classes
Cl_{omt}	une classe d'objets
O_{omt}	objet ou instance d'une classe
$KyAt_{omt}$	attribut clé
At_{omt}	attribut structurel
Op	une opération d'une classe
$Pr\,Op$	paramètre d'une opération
$Multp$	multiplicité ou cardinalité
Nme	nom d'une classe ou d'une propriété 'opération ou attribut'
Typ	type prédéfini ou type utilisateur

Tableau 1. Tableau de quelques notations définies.

3.1.2. L'objet

Un objet décrit un phénomène naturel du monde réel. Il est un concept, une abstraction ou quelque chose ayant des limites très claires et un sens précis dans le contexte du problème étudié. Chaque objet est identifié d'une façon unique dans le contexte de sa classe et représente une instance de cette dernière. Le terme «identité» signifie que les objets se distinguent grâce à leur existence inhérente et non pas grâce à la description des propriétés qu'ils peuvent avoir. L'objet peut être formellement défini comme suit :

$$\forall Cl_{omt} \in Cl_{diag} \ , \quad \forall O_{omt_1}, O_{omt_2} \in Cl_{omt}$$
$$\zeta(O_{omt_1}) \neq \zeta(O_{omt_2}) \Rightarrow O_{omt_1} \neq O_{omt_2}$$

Où $\zeta(O_{omt_i})$ représente l'identité de l'objet O_{omt_i}.

3.1.3. La classe

Une classe d'objet décrit un groupe d'objets ayant des propriétés similaires (attributs), un comportement commun (opérations), des relations communes avec les autres objets ainsi qu'une même sémantique. Ainsi, la classe est une abstraction qui décrit les propriétés pertinentes pour l'application et ignore les autres. Une classe est représentée par une boîte rectangulaire et est partagée en trois zones (Figure 7): nom de la classe, ses attributs (sa description statique), et ses méthodes (son comportement).

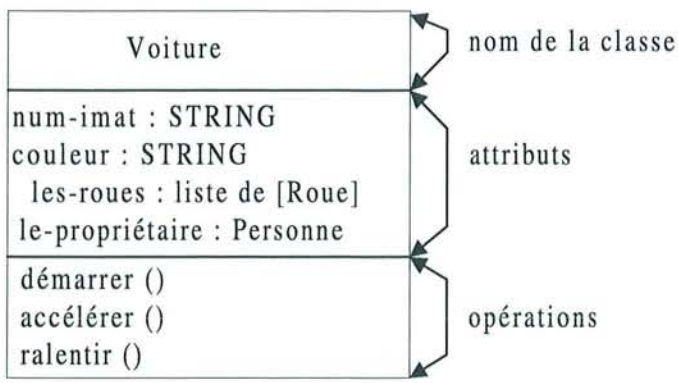

Figure 7. La notation OMT d'une classe d'objets.

Un attribut est une valeur de donnée détenue par les objets d'une classe et a une valeur quelconque pour chaque instance ou objet de sa classe. Un attribut a un type ou un domaine de définition (Ducateau, 1996). Le type peut être :

- prédéfini (booléen, caractère, entier, ...),
- composé à partir d'autres types, en utilisant les listes ou les ensembles.

Lors de la description d'un attribut, on peut restreindre le domaine de valeurs en supplément du type. Par exemple, on peut définir des domaines discrets (valeurs énumérées) ou un intervalle de valeurs (valeur minimale, valeur maximale). Tous les objets d'une classe partagent les mêmes opérations. L'opération a une signature constituée du nom de l'opération, des arguments ou paramètres ayant des noms et des types ou domaines de définition ainsi que du type de la valeur résultat (ou valeur de retour). Les opérations sont énumérées dans la troisième partie de la boîte de classe.

Les objets d'une classe ont les mêmes attributs et le même type de comportement. Formellement le concept de classe d'objets peut être défini comme suit :

$$\forall Cl_{omt} \in Cl_{diag} \ , \ \forall O_{omt_1}, O_{omt_2} \in Cl_{omt} \ ,$$
$$\gamma(O_{omt_1}) = \gamma(O_{omt_2}) \ \ et \ \ \beta(O_{omt_1}) = \beta(O_{omt_2})$$

Où $\gamma(O_{omt_i}) = \{a_{ij}\}$ $j = 1..n$ représente l'ensemble des attributs structurels de l'objet O_{omt_i},

$\beta(O_{omt_i})$ représente le comportement dynamique de l'objet O_{omt_i}.

3.1.4. Les relations

Il existe trois types de relations : association, agrégation et généralisation-spécialisation. Ces relations seront étudiées en détail.

3.1.4.1. Les associations

La notion d'association est utilisée dans les modèles entité-association et est adaptée dans les modèles objets tel que celui d'OMT. L'information qu'elle contient n'est pas subordonnée à une seule classe, mais dépend de deux classes ou plus.

Un lien est une connexion physique ou conceptuelle entre des instances de classes (ou objets). Mathématiquement, un lien est un tuple (une liste ordonnée d'instances). C'est une instance d'association. Une association a un nom et décrit un groupe de liens ayant une structure ainsi qu'une sémantique communes. Les associations sont fondamentalement bidirectionnelles. Une association est représentée par une ligne entre des classes. Un lien est représenté par une ligne entre des objets. Les associations peuvent être binaires, ternaires ou d'un autre ordre.

La multiplicité précise combien d'instances d'une classe peuvent être reliées à une seule instance d'une classe associée. Elle restreint le nombre d'objets reliés ensemble. En général, la multiplicité peut être précisée par un nombre, ou un ensemble d'intervalles, tels que «1» (exactement un), «1+» (un ou plus), «2-4» (deux à quatre, inclus) et «3, 5, 7» (trois, cinq ou sept).

Chaque association entre deux classes Cl_{omt_1}, Cl_{omt_2} peut être formellement définie par la fonction suivante :

$$\nu_{Cl_{omt_1},Cl_{omt_2}} : Cl_{omt_1} \rightarrow Cl_{omt_2}$$
$$O_{omt_1} \mapsto \nu_{Cl_{omt_1},Cl_{omt_2}}(O_{omt_1}) = \{O_{omt_2} \ / \ O_{omt_2} \in Cl_{omt_2}\}$$

Le type d'une association dépend de sa multiplicité (Figure 8). Deux types existent :

- si la valeur $Multp(\nu_{Cl_{omt_1},Cl_{omt_2}}(O_{omt_1}) = 1) \Leftrightarrow$ une association simple,
- si la valeur $Multp(\nu_{Cl_{omt_1},Cl_{omt_2}}(O_{omt_1}) = N) \Leftrightarrow$ une association multiple.

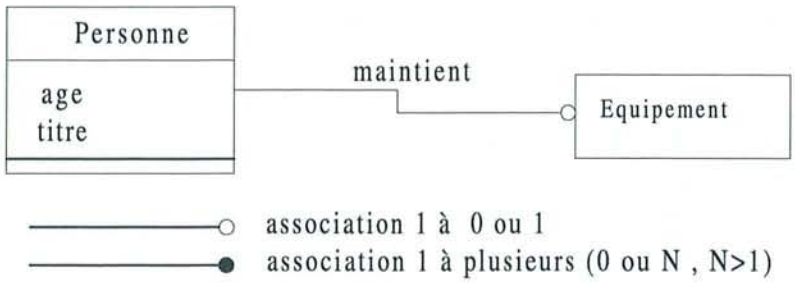

Figure 8. La notation OMT d'une relation d'association.

3.1.4.2. Les agrégations

Une agrégation est une forme spéciale de l'association car elle est couplée à une sémantique additionnelle qui est celle de «composé-composants» ou «partie de». Dans cette relation les objets représentant les composants d'une chose sont associés à un objet représentant l'assemblage entier (Figure 9). La propriété la plus significative d'une agrégation est la transitivité : si A est une partie de B et B est une partie de C, alors A est une partie de C. L'agrégation n'est pas symétrique : si A est une partie de B, B n'est pas partie de A.

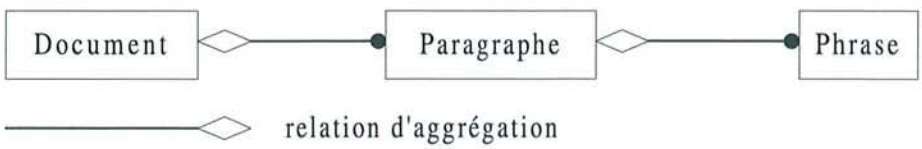

Figure 9. La notation OMT pour la relation d'agrégation.

Un couple (composé-composant) est une agrégation sur laquelle nous pouvons préciser la multiplicité du composant à l'intérieur du composé (ou l'assemblage). L'agrégation est représentée graphiquement comme une association, à l'exception du petit losange qui indique l'extrémité assemblage de la relation. L'existence d'un objet composant peut dépendre de l'existence d'un objet agrégat duquel il fait partie. Une agrégation peut avoir un nombre arbitraire de niveaux.

Une agrégation entre deux classes d'objets (Cl_{omt_1}, Cl_{omt_2}) peut être formellement définie par la fonction $\delta_{Cl_{omt_1}, Cl_{omt_2}}$ comme suit :

$$\delta_{Cl_{omt_1}, Cl_{omt_2}} : Cl_{omt_1} \rightarrow Cl_{omt_2}$$
$$O_{omt_1} \mapsto \delta_{Cl_{omt_1}, Cl_{omt_2}}(O_{omt_1}) = \{O_{omt_2} \ / \ O_{omt_2} \in Cl_{omt_2}\}$$

Le type d'une agrégation dépend de la valeur de multiplicité des composants faisant parties de l'assemblage :

- si $Multp(\delta_{Cl_{omt_1}, Cl_{omt_2}}(O_{omt_1})) = 1 \Leftrightarrow$ une agrégation simple,
- si $Multp(\delta_{Cl_{omt_1}, Cl_{omt_2}}(O_{omt_1})) = N \Leftrightarrow$ une agrégation multiple.

3.1.5. La relation généralisation/spécialisation

La généralisation est une construction utile pour le modèle conceptuel et pour l'implémentation. Elle facilite la modélisation en organisant les classes et en séparant ce qui est commun aux classes, de ce qui ne l'est pas. La généralisation est une relation entre une classe et une ou plusieurs versions affinées de la classe. La classe dont elle rassemble les propriétés communes (attributs et comportement) est appelée super-classe et chaque version affinée est appelée sous-classe. Les sous-classes héritent les propriétés (attributs et opérations) de leur super-classe. La généralisation est appelée la relation «est-un», puisque chaque instance d'une sous-classe est une instance de la super-classe. Le terme généralisation vient du fait qu'une super-classe généralise les sous-classes, alors que spécialisation fait référence au fait que les sous-classes affinent ou spécialisent la super-classe.

L'héritage est le mécanisme de partage des attributs et des opérations utilisant la relation de généralisation. Généralisation et héritage sont transitifs à travers un nombre arbitraire de niveaux. Une sous-classe peut redéfinir une caractéristique d'une super-classe en définissant une propriété portant le même nom. La généralisation est représentée par un triangle reliant une super-classe à ses sous-classes. La super-classe est reliée par une ligne au sommet du triangle (Figure 10).

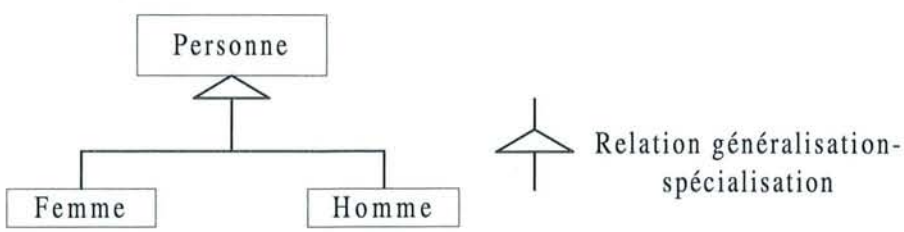

Figure 10. La notation OMT pour la relation généralisation/spécialisation.

Des contraintes de disjonction et de recouvrement sont définies dans la super-classe. Un triangle vide dans un héritage simple référence une contrainte de disjonction (chaque instance d'une sous-classe doit appartenir exactement et seulement à sa sous-classe), alors qu'un triangle plein référence une contrainte de recouvrement (une instance peut appartenir à plusieurs sous-classes, ou on dit que les sous-classes se chevauchent). L'héritage est le mécanisme qui implémente la relation de généralisation.

$$\eta_{Cl_{omt_g},Cl_{omt_s}} : Cl_{omt_g} \rightarrow Cl_{omt_s}$$
$$O_{omt_g} \mapsto \eta_{Cl_{omt_g},Cl_{omt_s}}\left(O_{omt_g}\right) = O_{omt_s} \quad / \quad O_{omt_s} \in Cl_{omt_s}$$

- $Cl_{omt_{si}} \cap Cl_{omt_{sj}} = \phi \quad , \forall i \neq j$ (contrainte de disjonction),

- $Cl_{omt_{si}} \cap Cl_{omt_{sj}} \neq \phi \quad , \forall i \neq j$ (contrainte de recouvrement).

3.2. Formalisation de MMS

La norme MMS (Manufacturing Message Specification) utilise une technique de modélisation par objets abstraits afin de décrire d'une manière aussi complète que possible le modèle de l'équipement industriel et les services qui le manipulent. Cette technique représente un outil qui aide à la compréhension de l'intention et des effets des services MMS plus que n'importe quelle sorte de description. Le concept de VMD (Virtual Manufacturing Device, Equipement Virtuel de Fabrication) représente un élément de base de la norme MMS. C'est un modèle représentant, pour la communication, le comportement d'un équipement réel visible de l'extérieur.

MMS est basée sur un mode de communication dit de type client-serveur Le VMD modélise le comportement du serveur. Le VMD est composé d'un ensemble d'éléments abstraits dits «objets MMS» qui représentent les ressources de l'équipement réel auquel le serveur est lié. Les services permettent à un client distant de contrôler l'équipement. Dans la suite nous présentons les concepts généraux définis dans la norme : objet, classe, attribut clé, attribut, ...

3.2.1. Définitions préliminaires

Nous définissons les notations (Tableau 2) utilisées dans la phase de formalisation des composants MMS.

Nr_{mms}	Modèle de la norme MMS
Cl_{mms}	Classe d'objets MMS
O_{mms}	Objet MMS ou instance d'une classe d'objets MMS
$KyAt_{mms}$	Attribut clé
At_{mms}	Attribut structurel
Srve	Opération ou un service d'une classe d'objets MMS
Pr Srve	Paramètre d'un service MMS
AtRef	Attribut de référence
AtLstRef	Attribut liste de références
AtCond	Attribut conditionnel
Multp	multiplicité ou cardinalité
X _Nme	Nom d'une classe ou d'une propriété 'service ou attribut'
X _Typ	Type de propriété
Const	Contrainte

Tableau 2. Tableau de quelques notations définies.

3.2.2. L'objet

La norme MMS définit un certain nombre d'objets abstraits résidant dans l'objet principal VMD. L'objet constitue l'interface d'accès à une ressource de l'équipement réel de

production (Pleinevaux, 1994). Un objet est une instance de sa classe. Un objet MMS a une identité unique dans le concept de sa classe et peut être défini comme suit :

$$\forall Cl_{mms} \in Nr_{mms} \quad , \quad \forall O_{mms_1}, O_{mms_2}$$
$$\zeta(O_{mms_1}) \neq \zeta(O_{mms_2}) \Rightarrow O_{mms_1} \neq O_{mms_2}$$

Où $\zeta(O_{mms_i})$ représente l'identité de l'objet O_{mms_i}.

3.2.3. La classe

La norme MMS définit un ensemble de classes génériques contenues dans le VMD. Une classe représente un groupe d'objets ayant une structure de données et un comportement communs. Chaque classe a un nom propre pour la référencer. Une classe d'objets MMS est caractérisée, par l'ensemble des attributs décrivant des caractéristiques visibles de tous les objets de cette classe, et par l'ensemble des opérations qui peuvent lui être appliquées en utilisant les services MMS définis dans la norme. Les objets d'une classe ont leurs propres valeurs d'attributs. Les valeurs sont définies par la norme ou peuvent être affectées par des services MMS. Une ou plusieurs valeurs d'attributs rendent un objet unique. Ce ou ces attributs sont appelés «Attributs Clé».

$$\forall Cl_{mms} \in Nr_{mms} \quad , \quad \forall O_{mms_1}, O_{mms_2}$$
$$\alpha(O_{mms_1}) = \alpha(O_{mms_2}) \ \ et \ \ \chi(O_{mms_1}) = \chi(O_{mms_2})$$

Où $\alpha(O_{mms_i})$ représente l'ensemble des attributs (attribut clé et attributs structurels) de l'objet O_{mms_i},

$\chi(O_{mms_i})$ représente l'ensemble des services MMS associés à l'objet O_{mms_i}.

3.2.4. Les relations

Dans la norme MMS et pour quelques objets, des attributs référençant d'autres objets existent. Ces attributs sont appelés «attributs de référence». Ils fournissent un mécanisme de création des relations entre les objets. Nous procédons à une abstraction de ces attributs en les modélisant sous forme de relations entre les objets.

Une relation entre deux classes d'objets MMS (Cl_{mms_1}, Cl_{mms_2}) sera définie formellement comme suit :

$$\vartheta_{Cl_{mms_1}, Cl_{mms_2}} : Cl_{mms_1} \rightarrow Cl_{mms_2}$$
$$O_{mms_1} \mapsto \vartheta_{Cl_{mms_1}, Cl_{mms_2}}(O_{mms_1}) = \{O_{mms_2} \ / \ O_{mms_2} \in Cl_{mms_2}\}$$

Le type de cette relation influence le type de l'attribut de référence qu'il présente. Ceci dépend du nombre d'objets O_{mms_2} de la classe Cl_{mms_2} :

- si $Multp\left(\vartheta_{Cl_{mms_1},Cl_{mms_2}}\left(O_{mms_1}\right)\right)=1 \Rightarrow$

$$\exists AtRef_1 \in O_{mms_1} \ / \ AtRef_1\_Nme = O_{mms_2}\_Nme,$$

- si $Multp\left(\vartheta_{Cl_{mms_1},Cl_{mms_2}}\left(O_{mms_1}\right)\right)=N \Rightarrow$

$$\exists AtLstRef_1 \in O_{mms_1} \ / \ AtLstRef_1\_Nme = \left\{O_{mms_2}\_Nme\right\}.$$

L'attribut de référence $AtRef_1$ correspond à une relation simple entre les deux objets, alors que l'attribut liste de référence $AtLstRef_1$ correspond à une relation multiple.

3.2.5. La relation d'héritage ou sous-typage

La relation d'héritage n'est pas définie explicitement dans la norme MMS. Nous allons utiliser les contraintes MMS pour exprimer cette relation. Les attributs appelés conditionnels le sont par le fait qu'ils font partie d'un objet si certaines conditions sont satisfaites. MMS exprime ce type d'attributs par le biais de contraintes spécifiant la condition à satisfaire. Les attributs sujets à la contrainte vont être considérés comme des attributs appartenant à l'objet, si la contrainte correspondante est satisfaite pour lui.

Considérons :

- $\alpha_1\left(O_{mms_1}\right)=\left\{At_{mms_i} \ / \ i=1,...,n\right\}$: l'ensemble des attributs structurels (non conditionnés) de l'objet O_{mms_1},

- $\alpha_{2(O_{mms_1},C_k)}\left(O_{mms_1}\right)=\left\{AtCond_{k_j} \ / \ j=1,...,m\right\}$: l'ensemble des attributs conditionnés par la contrainte C_k de l'objet O_{mms_1} et pour $k=1..p$.

On pose la fonction suivante ω :

$$\forall \alpha,\alpha_1,\alpha_2 \ / \ \alpha\left(O_{mms_1}\right)=\alpha_1\left(O_{mms_1}\right)\bigcup_{k=1}^{p}\alpha_{2(O_{mms_1},C_k)}\left(O_{mms_1}\right),$$

$$\omega_{Ommms_1,C_k}\left(\alpha\left(O_{mms_1}\right)\right)=\alpha_{2_{(Ommms_1,C_k)}}\left(O_{mms_1}\right).$$

La fonction ω sera explicitée ultérieurement, dans le paragraphe 4.8 du présent chapitre, dans l'exemple illustré par la figure 11.

Un nouvel attribut At_{mms_1} est aussi rajouté dans la structure de l'objet O_{mms_1} tel que :

$$At_{mms_1}\_Nme = Class\left(C_1\_Nme,...,C_p\_Nme\right)$$

Où p représente le nombre de classes ayant comme propres attributs structurels ceux considérés en tant qu'attributs conditionnés de l'objet O_{mms_1}.

Nous aurons autant de contraintes, $Const$, que de classes (citées dans le nouvel attribut At_{mms_1} de l'objet O_{mms_1}). Ces contraintes seront représentées comme suit :

$$Const_1\_Nme = [Class = C_1\_Nme\ ],$$

.

.

.

$$Const_p\_Nme = [Class = C_p\_Nme].$$

Il est à noter que les attributs conditionnels de l'objet O_{mms_1} (qui peuvent représenter les attributs structurels d'un autre objet MMS), seront cités après la contrainte qui lui est associée.

Dans la section suivante, nous abordons le processus de transformation ou de passage d'un modèle OMT vers un modèle MMS.

4. Processus de transformation d'OMT vers MMS

Afin d'établir le processus de transformation ou de traduction des modèles OMT en modèles MMS, nous définissons des règles appelées règles de transformation. Nous définissons tout d'abord une fonction de transformation $Trans$, d'OMT et de ses concepts vers les concepts MMS. Cette fonction est représentée comme suit :

$$Trans : OMT \times C \rightarrow C_{MMS}$$
$$(OMT, c) \mapsto c_{MMS} \ / \ Trans(OMT, c) = \{c_{MMS}\}$$

Tel que C, représente l'ensemble des concepts des formalismes OMT et MMS. La fonction $Trans$ transforme chaque concept source en celui (ou ceux) équivalent(s) MMS. Notons que chaque concept source pourrait être transformé en zéro, un ou plusieurs concepts MMS. Dans la suite, nous présentons les règles de transformation que nous avons définies.

4.1. Règle de transformation d'une classe d'objets

Une classe d'objet d'OMT (Cl_{omt}) sera transformée en une classe d'objets MMS avec le même nom.

$$R_{classe} :$$
$$\| Trans(OMT, Cl_{omt}) = Cl_{mms} \ / \ Cl_{omt}\_Nme = Cl_{mms}\_Nme.$$

4.2. Règle de transformation d'un attribut clé

Un attribut clé $KyAt_{omt}$ d'une classe d'objet OMT Cl_{omt} sera transformé en un attribut clé $KyAt_{mms}$ d'une classe d'objets MMS Cl_{mms} avec le même nom et le même type.

$$R_{KyAt_{omt}} :$$
$$\| Trans(OMT, KyAt_{omt}) = KyAt_{mms} \ / \ KyAt_{omt}\_Nme = KyAt_{mms}\_Nme,$$
$$KyAt_{omt}\_Typ = KyAt_{mms}\_Typ.$$

4.3. Règle de transformation d'un attribut structurel

Un attribut At_{omt} d'une classe d'objets OMT (Cl_{omt}) sera transformé en un attribut structurel At_{mms} de la classe d'objets MMS correspondante (Cl_{mms}), tout en gardant le même nom et le même type.

$$R_{At_{omt}} :$$
$$\| Trans(OMT, At_{omt}) = At_{mms} \ / \ At_{omt}\_Nme = At_{mms}\_Nme,$$
$$At_{omt}\_Typ = At_{mms}\_Typ.$$

4.4. Règle de transformation d'une opération

Comme nous venons de le présenter dans le chapitre précédent, une opération est une fonction ou une transformation qui peut être appliquée aux objets ou par les objets dans une classe. Ainsi une opération Op d'une classe d'objets OMT (Cl_{omt}) sera transformée en un service MMS ($Srve$) manipulant la classe d'objets MMS correspondante (Cl_{mms}).

$$R_{op} :$$
$$\| Trans(OMT, Op_{omt}) = Srve_{mms}$$

Le service correspondant aura la même signature que l'opération (le même nombre et le même type de paramètres et également le même type de valeur de retour, s'il en a).

4.5. Règles de transformation des relations

Les règles de transformation, propres aux relations statiques, que nous allons définir aborderons les trois types de relations structurelles.

4.6. Règle de transformation des relations d'association

Une relation d'association définie par la fonction ν reliant deux classes d'objets OMT (Cl_{omt_1}, Cl_{omt_2}) sera transformée en une relation ϑ, modélisant un attribut de référence ou un attribut liste de référence. Ceci dépend du type de la relation d'association au niveau conceptuel.

$$R_{assoc} :$$
$$\| Trans(OMT, \nu_{Cl_{omt_1}, Cl_{omt_2}}) = \vartheta_{Cl_{mms_1}, Cl_{mms_2}}$$

4.7. Règle de transformation des relations d'agrégation

Ce type de relation n'a pas été formellement et explicitement défini dans la norme MMS. Les relations existant entre les objets MMS sont concrétisées par des attributs de références (comme les pointeurs dans les langages de programmation à objets).

Malgré ceci, une relation d'agrégation ($\delta_{Cl_{omt_1},Cl_{omt_2}}$) reliant deux classes d'objets d'OMT (Cl_{omt_1}, Cl_{omt_2}) sera transformée en une relation ($\vartheta_{Cl_{mms_1},Cl_{mms_2}}$), modélisant un attribut de référence ou un attribut liste de référence. Ceci dépend du type de la relation d'agrégation au niveau conceptuel.

$$R_{agreg} :$$
$$\left\| Trans\left(OMT, \delta_{Cl_{omt_1},Cl_{omt_2}}\right) = \vartheta_{Cl_{mms_1},Cl_{mms_2}}\right.$$

4.8. Règle de transformation des relations de généralisation/spécialisation

Comme nous venons de le voir dans la section précédente, la relation de généralisation/spécialisation ou d'héritage sera transformée en une fonction ω, modélisant une contrainte MMS.

$$R_{gen-spec} :$$
$$\left\| Trans\left(OMT, \gamma\left(\eta_{Cl_{omt_1},Cl_{omt_2}}\right)\right) = \omega_{Cl_{mms_1},Cl_{mms_2}}\left(\alpha\left(O_{omt_1}\right)\right)\right.$$

Nous montrons par l'exemple suivant (Figure 11) l'application de la règle de transformation concernant la relation de généralisation-spécialisation.

Finalement, les règles que nous venons de présenter peuvent décrire les quelques concepts de la méthode OMT en concepts de la norme MMS.

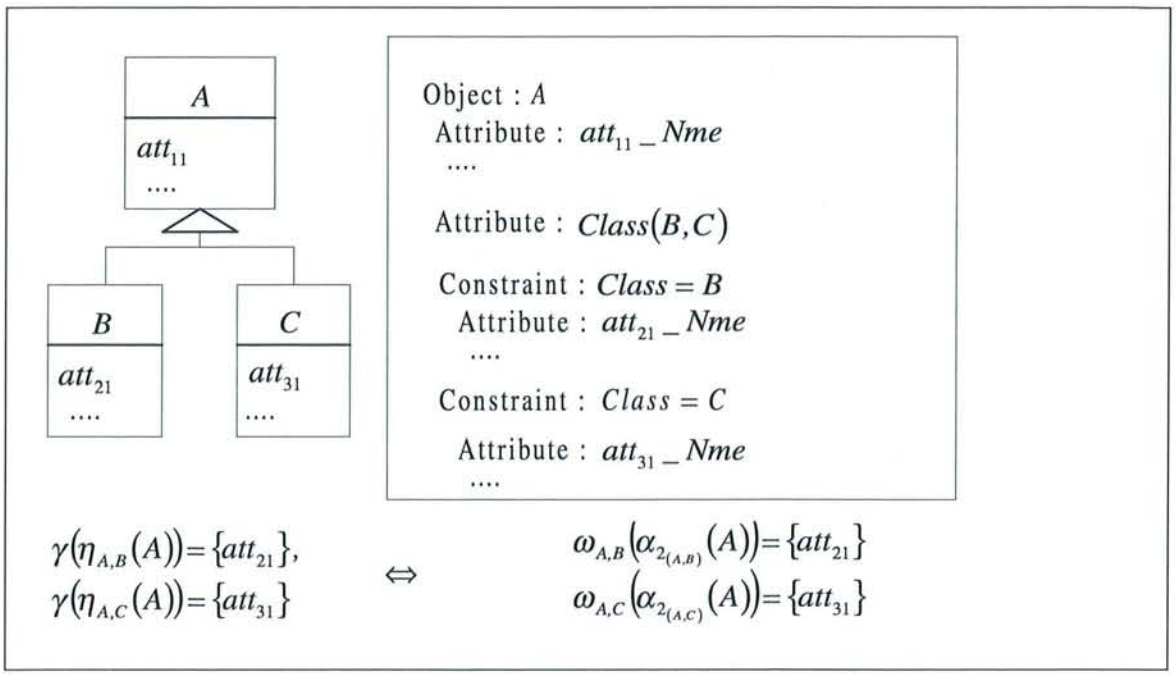

Figure 11. Exemple d'application de la règle sur une relation de généralisation-spécialisation.

5. Illustration

A titre d'exemple d'application de notre démarche, nous allons présenter la norme MMS elle-même en utilisant le modèle objet de la méthode OMT. A travers cette illustration, nous avons l'intention de valider les règles de transformation que nous venons de proposer. Pour ce faire, nous appliquons les règles sur l'objet Semaphore MMS dont on connaît a priori la structure. Nous retrouverons celle-ci après la transformation.

5.1. Modélisation

Un VMD est composé par différentes classes d'objets MMS définies dans la norme : variables nommées, type nommé, variable à accès dispersé, liste nommée, domaine, instance de programme, sémaphore, station opérateur, journal, variable anonyme, rubrique de sémaphore, condition événementielle, action événementielle et enregistrement d'événement.

La figure ci-dessous (Figure 12) présente le modèle OMT de l'objet MMS VMD de MMS, constitué de toutes les classes d'objets avec leurs attributs et leurs opérations qui peuvent leur être appliquées et qui se reposent sur les services MMS définis dans la norme.

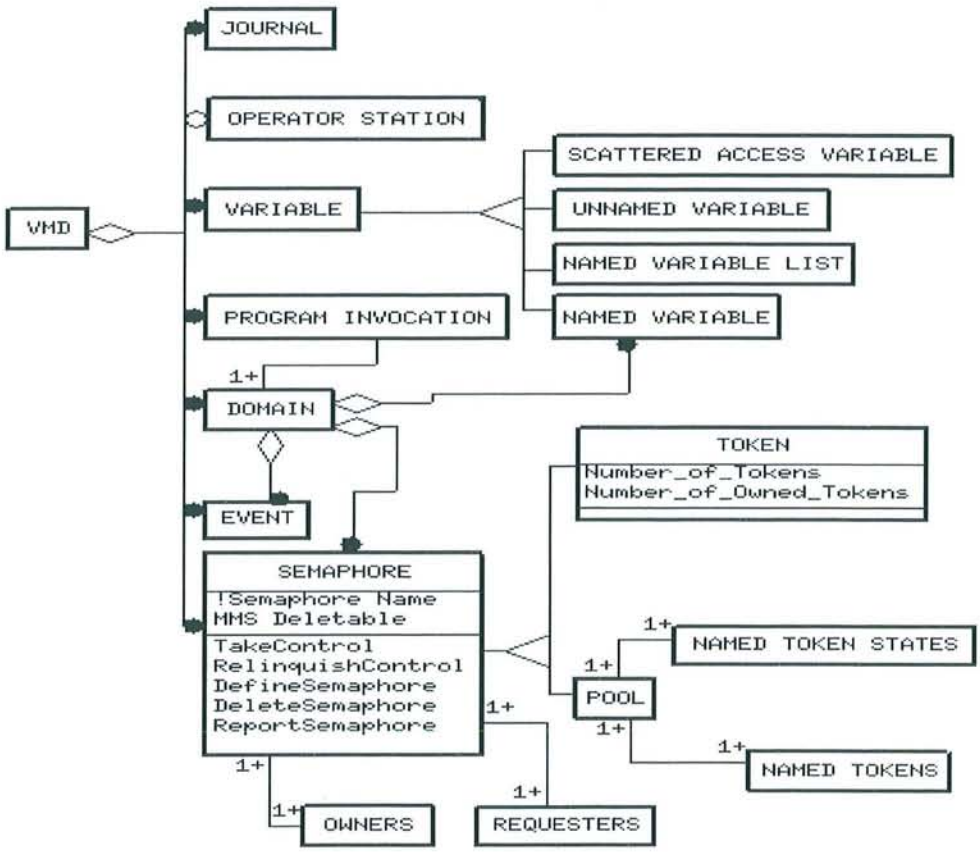

Figure 12. Modélisation OMT de l'objet MMS VMD.

Nous nous focalisons maintenant sur une partie du modèle objet MMS et détaillons la classe d'objets Sémaphore (Figure 13) et les classes d'objets qui lui sont associées. Le sémaphore MMS est porteur d'un mécanisme qui autorise l'accès contrôlé à des ressources partagées par différentes applications MMS. Il a des attributs tels que Semaphore Name, MMS Deletable Semaphore. Afin de manipuler l'objet sémaphore, MMS introduit un ensemble de services qui sont utilisés pour le créer, le détruire, le modifier ou le consulter. Le sémaphore MMS contient des jetons «tokens». Une synthèse complète et une étude approfondie de l'objet sémaphore se trouve dans (Castori, 1996).

MMS définit deux classes de sémaphore : le sémaphore banalisé «token semaphore» et le sémaphore étiqueté («pool semaphore»). Le sémaphore étiqueté contient des jetons nommés, alors que le sémaphore banalisé contient des jetons anonymes. Ces deux classes ne sont pas fondamentalement différentes.

La différence essentielle entre ces deux types de sémaphore réside dans la politique de l'allocation des jetons. Une application cliente peut effectuer une requête de prise de contrôle (Take Control) sur un sémaphore étiqueté en demandant l'acquisition d'un jeton nommé particulier. Ce dernier, s'il est libre, lui sera alloué. Sinon, l'application doit attendre même si d'autres jetons de ce sémaphore sont libres. Si l'application ne demande pas de jeton particulier alors n'importe quel jeton libre lui sera alloué. Dans les deux cas, le nom de jeton pris est retourné dans la réponse du Take Control. Ce nom doit être ensuite fourni à la requête Relinquish Control qui libère le jeton. Dans le cas d'un sémaphore banalisé, si l'application

demande l'acquisition d'un jeton, n'importe quel jeton libre lui est alloué. Généralement, le nombre d'applications pouvant prendre le contrôle d'un sémaphore est limité par le nombre initial de jetons de ce sémaphore. Si les jetons d'un sémaphore sont tous alloués, le sémaphore est dit pris. Si au moins un jeton n'est pas alloué, le sémaphore est dit libre.

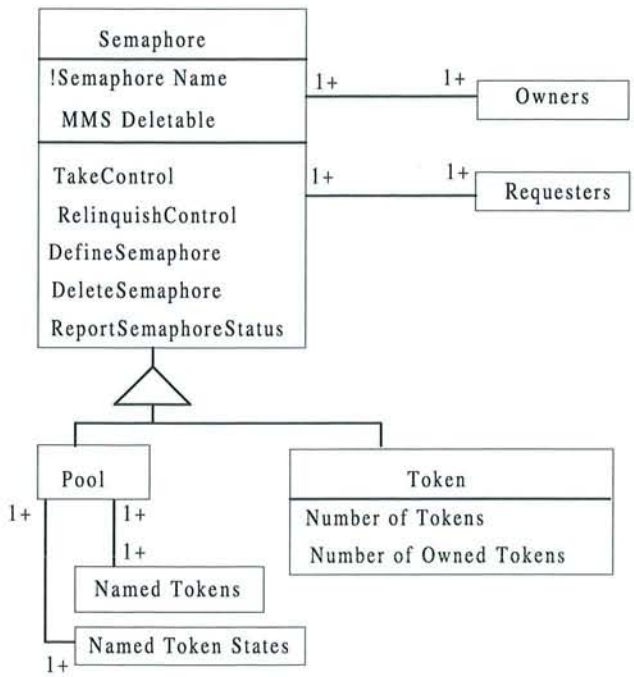

Figure 13. Modélisation OMT de l'objet MMS Semaphore.

Un sémaphore étiqueté est en quelque sorte lié à une ressource physique modélisée par le VMD. Par ce fait, le sémaphore étiqueté est toujours prédéfini et ne peut jamais être détruit par un service MMS DeleteSemaphore. Quant au sémaphore banalisé, il peut être détruit par le service précédent si la valeur de l'attribut MMS Deletable est «vrai». Ceci présente une autre différence entre les deux classes de sémaphores : étiqueté et banalisé.

5.2. Application des règles de transformation

R_{class}, $R_{KyAt_{omt}}$, $R_{At_{omt}}$, R_{assoc} , $R_{Gen\text{-}Spec}$, sont les règles de transformation utilisées dans la transcription de l'objet sémaphore modélisé en OMT vers la représentation normalisée de MMS. Chaque résultat obtenu après l'application d'une ou des règles de transformation, sera présenté en gras.

- la règle R_{class}, est utilisée comme suit :

$$R_{class} :$$
$$\| Trans(OMT, Semaphore_{omt}) = Semaphore_{mms}$$

Suivant la représentation MMS nous avons :

Object : Semaphore

- la règle sur l'attribut clé $R_{KyAt_{omt}}$, est utilisée comme suit :

$$R_{KyAt_{omt}} :$$
$$\| Trans(OMT, Semaphore\ Name) = Semaphore\ Name,$$

$$(Semaphore\ Name\_Typ)_{omt} = (Semaphore\ Name\_Typ)_{mms}$$

Suivant la représentation MMS nous avons :

Object : Semaphore
Key-Attribute : Semaphore Name : String

- la règle sur l'attribut structurel $R_{At_{omt}}$, est utilisée comme suit :

$$R_{At_{omt}} :$$
$$\| Trans(OMT, MMS\ Deletable) = MMS\ Deletable,$$

$$(MMS\ Deletable\_Typ)_{omt} = (MMS\ Deletable\_Typ)_{mms} = True - False$$

Suivant la représentation MMS nous avons :

Object : Semaphore
Key-Attribute : Semaphore Name : String
Attribute : MMS Deletable : Boolean

- la règle sur la relation d'une association R_{assoc}, est utilisée comme suit :

  - la relation entre les classes Semaphore et Owners :

$$R_{assoc} :$$
$$\| Trans(OMT, \nu_{Semaphore,Owner}) = \vartheta_{Semaphore,Owner}$$

Puisque la classe Semaphore est associée, par l'intermédiaire d'une relation d'association multiple $\nu_{Semaphore,Owners}$, avec la classe Owners, alors $\vartheta_{Semaphore,Owners}$ modélise un attribut liste de références appartenant à la classe Semaphore.

$$\exists AtLstRef_1 \in Semaphore \quad / \quad AtLstRef_1\_Nme = List\ of\ Owners$$

  - la relation entre les classes Semaphore et Requesters :

$$R_{assoc} :$$
$$\| Trans(OMT, \nu_{Semaphore,Requesters}) = \vartheta_{Semaphore,Requesters}$$

Puisque la classe Semaphore est associée, par l'intermédiaire d'une relation d'association multiple $\nu_{Semaphore,Requesters}$, avec la classe Requesters, alors $\vartheta_{Semaphore,Requesters}$ modélise un attribut liste de références appartenant à la classe Semaphore.

$$\exists AtLstRef_1 \in Semaphore \quad / \quad AtLstRef_1\_Nme = List\ of\ Requesters.$$

Ainsi, la représentation MMS est comme suit :

Object : Semaphore
Key-Attribute : Semaphore Name : String
Attribute : MMS Deletable : Boolean
Atribute : List of Owners
Attribute : List of Requesters

- la relation entre les classes Pool et Named Token :

$$R_{assoc} :$$
$$\| Trans(OMT, \nu_{Pool,Named\ Token}) = \vartheta_{Pool,Named\ Token}$$

Puisque la classe Semaphore est associée, par l'intermédiaire d'une relation d'association multiple $\nu_{Pool,Named\ Token}$, avec la classe Named Token, alors $\vartheta_{Pool,Named\ Token}$ modélise un attribut liste de références appartenant à la classe Pool.

$$\exists AtLstRef_1 \in Pool \quad / \quad AtLstRef_1\_Nme = List\ of\ Named\ Token$$

- La relation entre les classes Pool et Named Token States :

$$R_{assoc} :$$
$$\| Trans(OMT, \nu_{Pool,Named\ Token\ States}) = \vartheta_{Pool,Named\ Token\ States}$$

Puisque la classe Semaphore est associée, par l'intermédiaire d'une relation d'association multiple $\nu_{Pool,Named\ Token\ States}$, avec la classe Named Token, alors $\vartheta_{Pool,Named\ Token\ States}$ modélise un attribut liste de références appartenant à la classe Pool.

$$\exists AtLstRef_2 \in Pool \quad / \quad AtLstRef_2\_Nme = List\ of\ Named\ Token\ States.$$

- la règle sur la relation d'héritage $R_{gen-spec}$, est utilisée comme suit :

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Semaphore,Pool}(Semaphore))) = \omega_{Semaphore,Pool}(\alpha(Semaphore))$$

$$\gamma(\eta_{Semaphore,Pool}(Semaphore)) = \{List\ of\ Named\ tokens, List\ of\ Named\ tokens\ States\}$$
$$= \omega_{Semaphore,Pool}(\alpha(Semaphore)).$$

et

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Semaphore,Token}(Semaphore))) = \omega_{Semaphore,Token}(\alpha(Semaphore))$$

$$\gamma(\eta_{Semaphore,Token}(Semaphore)) = \{Number\ of\ Tokens, Number\ of\ Owned\ Tokens\}.$$
$$= \omega_{Semaphore,Pool}(\alpha(Semaphore)).$$

Un nouvel attribut est rajouté à l'objet Semaphore, ainsi que deux contraintes concernant respectivement les classes Pool et Token :

$$At_{mms}\_Nme = Class(Pool, Token),$$
$$Const_1\_Nme = [Class = Pool],$$
$$Const_1\_Nme = [Class = Token].$$

Suivant la représentation MMS nous avons :

Liste de règles de transformation :

$$R_{class}, R_{KyAt_{omt}}, R_{At_{omt}}, R_{assoc}, R_{Gen\text{-}Spec}$$

Object : Semaphore
Key-Attribute : Semaphore Name : String
Attribute : MMS Deletable : Boolean
Attribute : List of Owners
Attribute : List of Requesters
Attribute : Class (Pool, Token)
Constraint : Class = Pool
Attribute : List of Named Tokens
Attribute : List of Named Token States
Constraint : Class = Token
Attribute : Number of Tokens
Attribute : Number of Owned Tokens

- la règle sur l'opération R_{op}, est appliquée comme suit :

$$R_{op}:$$
$$\|Trans(OMT, Op_{Take\ Control}) = Srve_{Take\ Control}$$
$$R_{op}:$$
$$\|Trans(OMT, Op_{RelinquishControl}) = Srve_{RelinquishControl}$$

$$R_{op}:$$
$$\|Trans(OMT, Op_{DefineSemaphore}) = Srve_{DefineSemaphore}$$

$$R_{op}:$$
$$\|Trans(OMT, Op_{DeleteSemaphore}) = Srve_{DeleteSemaphore}$$

$$R_{op}:$$
$$\|Trans(OMT, Op_{ReportSemaphoreStatus}) = Srve_{ReportSemaphoreStatus}$$

5.3. Synthèse des résultats

En conclusion, l'application de toutes ces règles conduit bien au résultat attendu et connu dès le départ. Le tableau 3 montre la représentation MMS de l'objet Semaphore qui est en réelle concordance avec le résultat que nous venons d'avoir. Ceci représente une première validation de nos règles de transformation.

Object : Semaphore
Key-Attribute : Semaphore Name
Attribute : MMS Deletable
Attribute : Class(Token, Pool)
Constraint : Class = Token
Attribute : Number of Tokens
Attribute : Number of Owned Tokens
Constraint : Class = Pool
Attribute : List of Named Tokens
Attribute : List of Named Token States
Attribute : List of Owners
Attribute : List of Requesters.

Tableau 3. La représentation MMS de l'objet Sémaphore.

6. Conclusion

Nous avons présenté notre démarche qui a pour vocation la génération du système de communication à partir de la seule étude du système d'information et pour une application donnée. Pour ce faire, nous nous sommes inspirés des travaux menés par Gargouri et al qui ont défini un formalisme de modélisation ayant comme vocation l'aspect transformation entre différentes représentations conceptuelles orientées objet et les différents environnements de développement. Nous avons formalisé les concepts, que nous avons adoptés, pour les deux formalismes OMT et MMS. Nous avons défini également quelques règles de transformation afin d'assurer le passage, sans pertes sémantiques, entre les deux formalismes concernés.

Nous avons montré la validité de nos propositions en les applicant sur l'objet MMS Semaphore.

Rappelons que notre démonstration était basée sur le fait que nous connaissions bien la source (objet sémaphore modélisé par OMT) et la destination (la représentation MMS de l'objet sémaphore).

Nous avons appliqué, dans l'annexe de ce mémoire, notre démarche à la norme d'accompagnement du Robot (ISO3, 1991). Notre démarche semble représenter un moyen pour la définition de nouvelles normes d'accompagnement. Les travaux en cours ou déjà élaborés pourraient justifier une telle perspective :

- Le travail de recherche de C. Thomas (Thomas, 1997) concerne la problématique de l'intégration des activités de gestion de production d'atelier dans le système physique de fabrication. L'objectif est, en se basant sur l'étude des systèmes d'information de la gestion de production, de proposer une structure de communication permettant de supporter cette fonctionnalité dans l'atelier en intégrant au niveau de ses équipements des informations caractéristiques. Ce travail contribue à la définition d'une interface de communication entre la gestion d'atelier et les procédés industriels qui le composent, en utilisant les potentialités des réseaux locaux industriels.

- Le travail de recherche réalisé par Wu (Wu, 1997), a contribué à l'intégration des fonctionnalités multimédias dans le contexte MMS. Cette intégration a conduit à une modélisation de l'environnement multimédia distribué de telle sorte qu'on puisse obtenir l'interopérabilité et la coopération des composants (équipements réels, ressources,...) dans un milieu de communication normalisé. Un modèle appelé VCS (Virtual Communication Support), pour lequel nous avons contribuer à la spécification (Wu, 1996), a été proposé pour virtualiser le système de communication en le considérant comme un équipement spécifique caractérisé par les ressources de communication et les opérations les manipulant. Ainsi les ressources de communication seront manipulables de la même manière que l'équipement de production par des services MMS. Le VCS décrit le comportement extérieur du système de communication, vis-à-vis de l'application multimédia. Le concept VMD a été utilisé pour virtualiser le support de communication. Le VCS, est au support de communication ce que le VMD est à l'équipement de production.

- Le comité international IEC (International Electrotechnical Committee) est en train de définir une norme d'accompagnement MMS des entreprises d'électricité (IEC, 1996a, 1996b). Ainsi, un protocole d'application est spécifiquement adapté aux besoins des entreprises d'électricité pour permettre l'échange de données entre les centres de contrôle ICCP (Inter Control Center Protocol) et TASE.2 (Telecontrol Application Service Element) sont synonymes et correspondent au protocole de la couche d'application spécifique. Les services MMS utilisés par ICCP/TASE.2 sont, : les services de gestion de variables et liste de variables MMS, de gestion d'instances de programmes, de gestion de la station d'opérateur et gestion des événements. Il faut souligner que juste un sous-ensemble de chaque type de services a été implémenté.

Notre proposition prend en considération le modèle général MMS en sa totalité (objets et services). C'est à dire que tous les composants de MMS sont disponibles sur l'équipement considéré. Mais, malheureusement les constructeurs de VMDs n'implémentent pas toutes les possibilités offertes par MMS. Ainsi, ils implémentent des sous-ensembles de mécanismes MMS. L'ensemble des objets et services implémentés sur l'équipement réel est appelé la classe de conformité de cet équipement. Nous présentons dans le chapitre suivant, la prise en compte du contexte d'implémentation, c'est-à-dire des classes de conformité des équipements, et les effets qu'elles induisent sur l'application de nos propositions à travers l'ingénierie de notre méthode.

Ingénierie de la méthode

1. Répartition d'une application

Les systèmes automatisés actuels s'appuient sur des concepts de pilotage décentralisés qui peuvent être distribués sur un espace physique plus ou moins étendu (Darscht, 1995). Les possibilités technologiques pour distribuer les traitements et/ou les données sur un ensemble de machines doivent être organisées autour de moyen de communications (les réseaux locaux industriels, les réseaux de terrains) supportant les échanges et les traitements d'informations (Bayart, 1995).

Cependant, la distribution des éléments d'un système automatisé reste un problème majeur à résoudre. Elle doit prendre en compte les caractéristiques des éléments interconnectés et des éléments d'interconnexion ainsi que les contraintes liées à la sûreté de fonctionnement du système en terme de fiabilité, disponibilité, maintenabilité, ... A ces critères, il faut bien évidemment ajouter l'aspect économique qui représente le plus souvent un facteur déterminant du choix d'une solution par rapport à une autre.

La définition d'une architecture physique est le résultat d'activités de choix de matériels (machines, réseaux, protocoles), de dimensionnement de ces matériels et de définition des points de connexion. Afin de valider une telle architecture il est indispensable de connaître sa caractérisation (capacités, performances, coûts, ...). La validation sera faite sur la base du respect de contraintes budgétaires d'investissement en matériels, de

l'intéropérabilité des constituants, des contraintes d'ordre technologique, ... Le résultat de la projection de l'architecture fonctionnelle (l'abstraction formelle de la structure et du comportement des activités du système automatisé distribué) sur l'architecture matérielle définit l'architecture dite opérationnelle.

Nous pouvons conclure qu'actuellement le choix d'une architecture matérielle ne suit pas une méthode formalisée, ni une méthode déductive. Il repose plutôt sur le savoir-faire et l'expérience des concepteurs.

Nous avons souligné, dans le chapitre I, qu'une intervention de la part du concepteur ou de l'intégrateur est néanmoins nécessaire pour répartir les applications sur les différents sites reliés par un réseau local. Rappelons que nous considérons que le concepteur du système d'information ne connaît pas a priori la norme MMS.

Ainsi, la distribution de l'application visant à utiliser la messagerie industrielle sera déterminée par son concepteur. Nous proposons un moyen, afin de l'aider dans sa mission. Ainsi, nous définissons une classe OMT abstraite appelée «Référence» qui permet au concepteur d'identifier la répartition physique des objets de son système. Chaque sous-classe de la classe «Référence» indique alors un point de connexion à un réseau de communication. Par ce principe, la figure 1 modélise une application composée de deux supports d'information reliés par un système d'interconnexion.

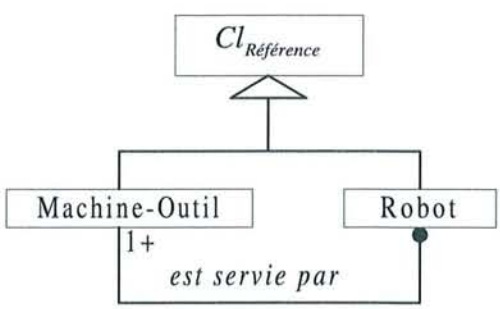

Figure 1. Répartition de l'application via la classe abstraite « $Cl_{Référence}$ ».

Ainsi, l'un des supports d'information contient les données spécifiques à la classe Machine-Outil, l'autre celles spécifiques à la classe Robot. Cependant, le modèle OMT mentionne un second lien «est servie par» entre ces deux classes. Si nous conservons la sémantique de cette relation dans la phase d'implémentation, alors le site «Machine-Outil» devrait contenir l'ensemble des informations supportées par la classe Robot et vice-versa (le site «Robot» devrait mémoriser les informations de la classe Machine-Outil). Le résultat induisant des informations redondantes sur les différents sites physiques de l'application est difficilement gérables (problèmes de cohérence spatio-temporelle des données).

Nous proposons alors de supprimer certaines relations du modèle lors de la phase d'implémentation de l'application en introduisant une notion de niveau entre les classes OMT. La classe «Référence» représente le niveau 0. Ensuite les classes en relation avec la classe «Référence» sont de niveau 1. Puis nous supprimons tous les liens entre les classes de niveau 1. Cette démarche est reconduite en identifiant les classes en relation avec les classes de niveau 1 pour obtenir les classes de niveau 2 et ainsi de suite. Nous présentons ci-dessous l'algorithme correspondant.

Nous posons la relation *R* suivante :

$$\forall Cl_i, Cl_j \in Cl_{diag} \;/\; Cl_j = \eta_{Cl_i,Cl_j}(Cl_i) \; ou \; Cl_j = \delta_{Cl_i,Cl_j}(Cl_i) ou \; Cl_j = \nu_{Cl_i,Cl_j}(Cl_i)$$

alors

$$R(Cl_i, Cl_j) = 1$$

sinon

$$R(Cl_i, Cl_j) = 0.$$

Début

Pour i de 1 à n faire /* Niveau des autres classes est indéterminé «Null»*/

Niveau_Cl_i ← Null /* n : nombre total des classes dans le modèle objet */

Fin Pour /* y compris la classe référence $Cl_{Référence}$ */

$Cl_1 \leftarrow Cl_{Référence}$

Niveau_$Cl_{Référence}$ ← 0 /* Niveau de la classe de référence $Cl_{Référence}$ */

Niveau ← 1

Répéter

fin-algo ← vrai

Calculer_Niveau (Niveau, fin-algo)

Supprimer_Lien (Niveau)

Niveau ← Niveau+1

Jusqu'à fin-algo

Fin

Fonction : Calculer_Niveau (Niveau, fin-algo)

Début

Pour i de 1 à (n-1) faire

pour j de 2 à n faire

Si [(Niveau_Cl_i = =Niveau-1) et (Niveau_Cl_j = = Null)]

Alors Si $R(Cl_i, Cl_j) = 1$

Alors Niveau_Cl_j ← Niveau

fin-algo ← faux

Fin Si

Fin Si

Fin Pour

Fin Pour

Fin

Fonction : Supprimer_Lien (Niveau) /* Pour des classes ayant le même niveau */

Début

Pour i de 1 à (n-1) faire

Pour j de 2 à n faire

Si (Niveau_Cl_i = Niveau_Cl_j = Niveau)

Alors $R(Cl_1, Cl_2) \leftarrow \emptyset$

Fin Si

Fin Pour

Fin Pour

Fin

Finalement, les relations restantes entre les classes OMT ne sont traduites que dans un seul sens qui est du niveau le plus bas vers le niveau le plus haut. En revenant à la figure 1 et puisque les classes «Machine-Outil» et «Robot» seront sur deux sites différents, les deux classes seront du niveau 1. Ainsi et d'après notre algorithme, la relation «est servie par» entre les deux classes précédentes sera supprimée puisqu'elle présente une relation entre des classes du même niveau. Mais la fonctionnalité n'est pas perdue car elle se retrouve au sens des opérations sur les objets.

2. Détermination de la nature des objets MMS générés

2.1. Introduction

Dans le paragraphe précédent nous avons introduit une classe abstraite appelée «Référence» permettant au concepteur de spécifier la distribution du modèle d'information à travers le système de communication. Nous nous attachons maintenant à définir la structure interne de chaque support physique à partir du modèle OMT, c'est à dire à identifier les différents objets MMS (VMD, variables, domaine, ...) qui virtualisent ce site. Contrairement au chapitre 2 où l'objectif était d'élaborer de nouvelles normes d'accompagnement MMS à partir d'OMT induisant la création et/ou l'évolution d'objets MMS, nous nous plaçons ici dans un contexte d'ingénierie. L'environnement OMT doit nous permettre de décrire une application industrielle puis de générer le code MMS basé sur les objets et services MMS existants. Une correspondance des classes OMT en objets MMS est donc nécessaire.

Force est de constater que la méthode OMT ne peut pas être porteuse de la sémantique permettant cette interprétation. Ainsi, afin d'automatiser le choix de la nature des objets générés structurant la future application, nous proposons un ensemble de conventions. Elles portent d'une part sur les objets (aspect statique) et d'autre part sur les services (aspect dynamique).

2.2. Conventions concernant l'aspect statique

Nous pouvons déterminer la nature de certains objets MMS, issus du processus de transformation, à partir du modèle objet OMT comme suit :

2.2.1. Convention concernant le VMD

Toute classe d'objets Cl_{omt}, d'un diagramme de classe de l'application étudiée, étant sous-type de la classe abstraite Référence $Cl_{Référence}$ ($\eta_{Cl_{Référence},Cl_{mms}}$), est supportée par un objet VMD_{mms}. Les informations propres à Cl_{omt} seront traduites en variables nommées MMS et supportées par le VMD ($VMD_{mms} - Cl_{omt}\_Nme$) qui lui est associé.

$$\forall Cl_{Référence}, Cl_{omt} \in Cl_{diag} \quad / \quad \eta_{Cl_{Référence},Cl_{omt}} \Rightarrow$$

$$\| Trans(OMT, Cl_{omt}) = VMD_{mms} \qquad (application\ de\ la\ règle\ R_{class})$$
$$avec\ VMD_{mms}\_Nme = Cl_{omt}\_Nme.$$

$$\forall At_{omt} \in \alpha(Cl_{omt}) \Rightarrow$$
$$\| Trans(OMT, At_{omt}) = variableNommée_{mms} \quad avec$$
$$variableNommée_{mms}\_Nme = At_{omt}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt}\_Typ .$$

Donc, toute classe OMT de niveau 1 est traduite en un objet VMD. Le nom de sa classe correspond au nom du VMD.

2.2.2. Convention concernant le domaine

Toute classe d'objet Cl_{omt_2}, d'un diagramme de classes, en relation d'*agrégation* ($\delta_{Cl_{omt_1}, Cl_{omt_2}}$) avec la classe Cl_{omt_1} sous-type de la *classe de référence* $Cl_{Référence}$ est supportée par une ou plusieurs instances de la classe $domaine_{mms}$.

$$\forall Cl_{Référence}, Cl_{omt_1}, Cl_{omt_2} \in Cl_{diag} \ / \ \eta_{Cl_{Référence}, Cl_{omt_1}} , \ \delta_{Cl_{omt_1}, Cl_{omt_2}} \Rightarrow$$
$$\| Trans(OMT, Cl_{omt_2}) = domaine_{mms} \quad (application\ de\ la\ règle\ R_{class})$$
$$avec \quad domaine_{mms}\_Nme = Cl_{omt_2}\_Nme .$$

Les informations propres à la classe Cl_{omt_2} sont traduites en variables nommées MMS et supportées par le ou les objets domaines ($domaine_{mms} - Cl_{omt_2}\_Nme$) qui lui sont associés.

$$\forall At_{omt} \in \alpha(Cl_{omt_2}) \Rightarrow$$
$$\| Trans(OMT, At_{omt}) = variableNommée_{mms} \quad avec$$
$$variableNommée_{mms}\_Nme = At_{omt}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt}\_Typ .$$

Donc, toute classe OMT de niveau 2 en relation d'agrégation avec une classe OMT de niveau 1 est traduite en objet domaine. Le nom de sa classe correspond au nom du domaine.

2.2.3. Convention concernant les variables structurées MMS

Toute classe OMT non traduite en objet VMD ou en objet de domaine MMS est traduite en variable structurée MMS. Le nombre d'instances de variables MMS générées dépend de la multiplicité de la relation entre les classes. Les variables sont attachées à un domaine si la classe OMT est associée directement ou indirectement à une classe OMT de niveau 2 traduite en objet de domaine.

$$\forall Cl_{omt} \in Cl_{diag} \ , \forall At_{Omt_i} \in \alpha(Cl_{omt}); i = 1..n \ /$$
$$\| Trans(OMT, Cl_{omt}) \neq VMD_{mms} \ et \ \| Trans(OMT, Cl_{omt}) \neq domaine_{mms}$$
$$\Rightarrow$$
$$\| Trans(OMT, Cl_{omt}) = VariableStructurée_{mms} \quad avec$$
$$VariableStructurée_{mms}\_Nme = Cl_{omt}\_Nme$$
$$= \{$$
$$At_{omt_1}\_Nme$$
$$At_{omt_2}\_Nme$$
$$.....$$
$$At_{omt_n}\_Nme$$
$$\}$$
$$VariableStructurée_{mms}\_Typ = \{$$
$$At_{omt_1}\_Typ$$
$$At_{omt_2}\_Typ$$
$$.....$$
$$At_{omt_n}\_Typ$$
$$\}$$

2.2.4. Convention concernant le sémaphore

Bigand et al (Bigand, 1998a, 1998b), ont proposé une extension à la méthode UML concernant la sémantique des relations entre les classes d'objets et plus particulièrement les contraintes dynamiques durant les opérations de création/destruction sur les instances (objets) reliées. Les contraintes dynamiques sont la contrainte d'exclusivité temporelle et d'existence.

Une classe Cl_{omt_1} est temporellement exclusive d'une classe Cl_{omt_2} quand chaque instance O_{omt_1} de la classe Cl_{omt_1} peut être associée avec plusieurs instances O_{omt_2} de la classe Cl_{omt_2} avec une unique instance à un instant donné (Figure 2). Cette contrainte est représentée par le symbole d'exclusivité *X* et par une flèche de la classe source vers la classe destination.

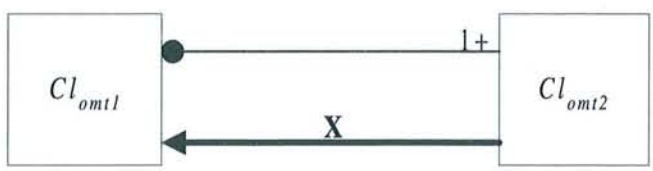

Figure 2. Contrainte d'exclusivité.

Nous pouvons formaliser cette association contrainte d'exclusivité temporelle comme suit par la fonction $\nu_{'X'omt_1,omt_{2\,t}}$:

$$A\ un\ t\ donné\quad \forall Cl_{omt_1}, Cl_{omt_2} \in Cl_{diag}\ , \forall O_{omt_1} \in Cl_{omt_1}\ , \exists!\, O_{omt_2} \in Cl_{omt_2}\ /$$

$$v_{'X'Cl\,omt_1, Cl_{omt_2}} : Cl_{omt_1} \rightarrow Cl_{omt_2}$$

$$O_{omt_1} \mapsto v_{'X'Cl_{omt1}, Cl_{omt2}}\left(O_{omt_1}\right) = O_{omt_2}$$

Et nous proposons la convention suivante :

- Toute relation avec une contrainte exclusive entre deux classes OMT réparties (Figure 1) est transformée en une instance de la classe sémaphore étiqueté (Pool Semaphore) MMS. Ceci est vrai si la classe OMT destinataire est traduite en objets MMS supportés respectivement par un VMD ou un domaine. (Figure 3).

$$\forall Cl_{omt_1}, Cl_{omt_2} \in Diag_{classe}\ /\ Trans\left(Cl_{omt_1}, Cl_{omt_2}\right) = VMD\ \ ou$$

$$Trans\left(Cl_{omt_1}, Cl_{omt_2}\right) = Domaine\ \ et\ v_{'X'Cl_{omt1}, Cl_{omt2}} \Rightarrow$$

$$\left\| Trans\left(OMT, v_{'X'Cl_{omt_1}, Cl_{omt_2}}\right) = Semaphore_{mms}\right.$$

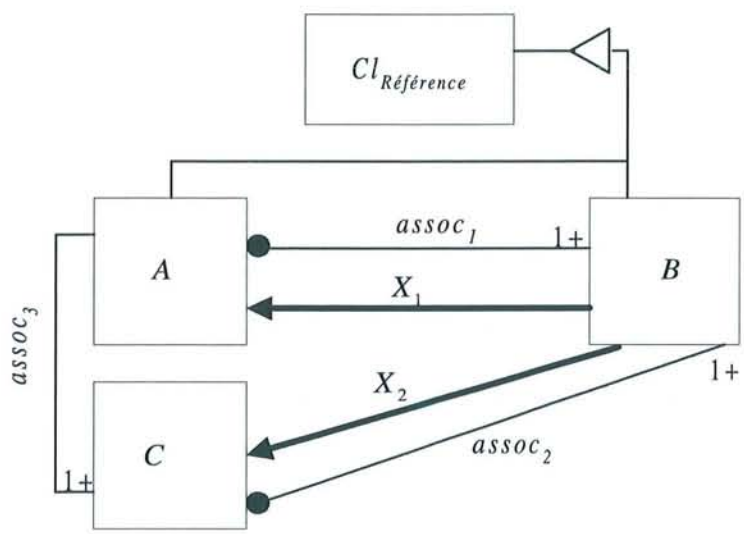

Figure 3. Relations avec contraintes d'exclusivité et l'influence sur le processus de transformation vers MMS.

Ainsi sur la Figure 3, les associations $assoc_1, assoc_2$ sont contraintes exclusives. Etant donné que les classes A, B sont sous-classes de la classe abstraite $Cl_{Référence}$, ces deux classes seront supportées par une classe VMD MMS. La classe C est associée à A par le biais de l'association multiple $assoc_3$, C est donc supportée par une instance de la classe domaine MMS. Les associations contraintes exclusives $assoc_1, assoc_2$ seront représentées par une instance de la classe sémaphore MMS et respectivement dans la classe VMD MMS et dans la classe domaine MMS.

Le Tableau 1 montre la représentation normalisée MMS, après la transcription du modèle objet OMT présenté par la figure 3 en MMS.

Objet : **VMD** Tous les attributs définis dans la norme Attribut Clé : *A* ..... Attribut : Sémaphore ..... Objet : Sémaphore Attribut Clé : X1 Attribut : MMS Deletable(false) Attribut : Class(Pool) Contrainte : Class=Pool	Objet : **Domaine** Tous les attributs définis dans la norme Attribut Clé : *C* ..... Attribut : Sémaphore ..... Objet : Sémaphore Attribut Clé : X2 Attribut : MMS Deletable(false) Attribut : Class(Pool) Contrainte : Class=Pool

Tableau 1. Représentation normalisée MMS des constituants de la figure 3.

3. Détermination de la nature des services MMS

Avant d'aborder les conventions que nous allons proposer pour la génération des services MMS, nous abordons les aspects dynamiques dans OMT et dans MMS.

3.1. Aspects dynamiques d'OMT

La dimension dynamique dans la méthode OMT est représentée par les concepts d'opération et d'événement. Rappelons que le concept d'opération permet de décrire, localement à la classe, l'ensemble des traitements qu'il est possible d'effectuer sur les objets. Une opération instantanée est une action, alors qu'une opération durable dans le temps est une activité. Le concept d'événement, quant à lui, est une abstraction d'un phénomène du monde réel opérant un changement dans le système d'information.

Un événement est une voie de transmission à sens unique d'un objet vers un autre. Il n'est pas comparable à un sous-programme appelé renvoyant une valeur. Mais dans le cas où l'objet attendrait une réponse de l'autre, cette réponse est considérée comme un événement distinct (Rumbaugh, 1995).

Les contraintes dynamiques permettent de spécifier les règles qui restreignent l'évolution des objets au cours du temps. De telles contraintes sont définies par le biais de graphes d'états-transitions (Figure 4) ayant comme nœuds les états et comme arcs les transitions.

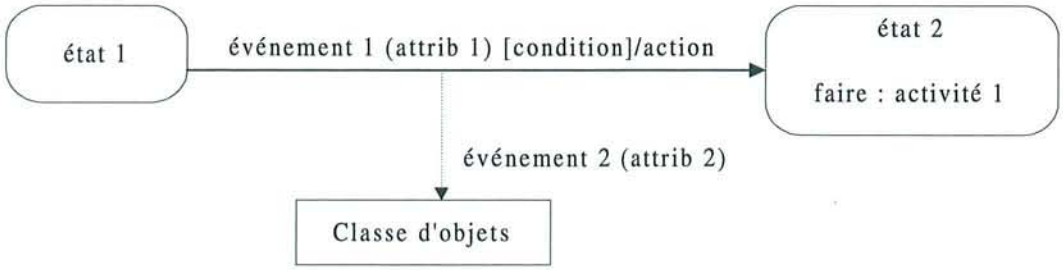

Figure 4. Notation étendue pour les diagrammes d'états.

Comme nous le montrons dans la figure 4, l'objet passe de l'*état 1* à l'*état 2* quand *événement1* se produit et si la condition est satisfaite. En changeant d'état, l'objet effectue une action. Un objet peut envoyer un événement vers un autre objet ; comme c'est le cas pour *événement 2* envoyé à classe d'objets. Un système d'objets interagit par l'échange d'événements. La ligne pointillée de la transition vers un objet indique qu'un événement est envoyé à l'objet quand la transition est franchie. Les attributs de *événement 2* appartiennent à la classe qui l'envoie.

Pour faciliter la transformation des modèles de comportement OMT en syntaxe MMS, nous définissons des fonctions représentant des actions élémentaires pour des applications en productique : lire une information, exécuter, arrêter, reprendre et réinitialiser une tâche, prendre une ressource, ...etc. En ce qui concerne les objets locaux le concepteur a la liberté complète pour choisir les opérations et les événements échangés avec les autres objets locaux. La figure 5 montre deux classes Cl_1 et Cl_2 supportées par deux entités physiques distantes. La figure 6 représente le graphe d'états de la classe Cl_1. La classe Cl_1 évoque une opération, sur la classe Cl_2, qui sera une opération de base : prendre ressource, manipuler une tâche, ...

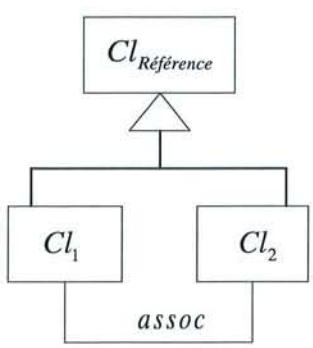

Figure 5. Exemple de deux classes sur deux entités physiques distantes.

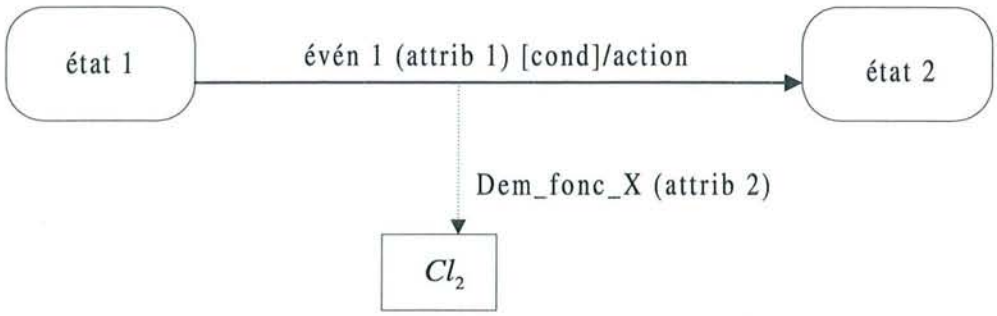

Figure 6. Modèle dynamique de la classe Cl_1.

Mais malheureusement nous manquons de sémantique explicite et bien définie sur la notion d'événement dans OMT. Nous avons étudié cet aspect aussi abordé dans le langage unifié UML et nous l'avons adopté. Puisque OMT constitue un des paliers de UML, nous pouvons justifier d'une telle utilisation. La communication entre objets passe via les messages. Un message est une sorte d'événement. Nous présentons la définition de la sémantique d'un événement au sein de UML : un événement est une occurrence de quelques chose remarquable. Il est une occurrence qui peut déclencher une transition d'état. Les événements peuvent être de types différents :

- **ChangeEvent** ; une condition (généralement décrite comme une expression booléenne) désignée devient vraie. La notation lui est associée est :

  When (expression booléenne)

  L'événement se produit à chaque fois que la valeur de l'expression change de faux à vrai. Notons que ceci est différent de la condition de garde que l'on trouve également sur une transition. La condition de garde est évaluée une fois que son événement s'est produit ; si elle est fausse alors la transition n'est pas franchie et l'événement est perdu.

- **SignalEvent** ; la réception d'un signal explicite (ou un stimulus asynchrone) d'un objet à un autre. La notation qui lui est associée est la signature de l'événement (nom d'événement et la liste de ses paramètres) comme déclencheur sur une transition.

- **CallEvent** ; la réception, par un objet, d'une demande d'une opération. La notation qui lui est associée est la signature de l'opération comme un déclencheur sur une transition.

  Un événement de type SignalEvent ou CallEvent peut être défini en utilisant le format suivant :

  event-name '(' comma-separated-parameter-list ')'

  le paramètre a le format : parameter-name ' :' type-expression

- **TimeEvent** ; le passage d'une période de temps après un événement désigné (souvent l'entrée à l'état courant) ou l'occurrence d'une date/temps donnée. La notation qui lui est associée est une expression temporelle comme déclencheur sur

une transition. La forme commune d'un tel événement est le temps passé depuis l'entrée à l'état courant :

After (période de temps)

Un événement n'est pas local à une seule classe. La figure 7, montre les types d'événement définis dans UML.

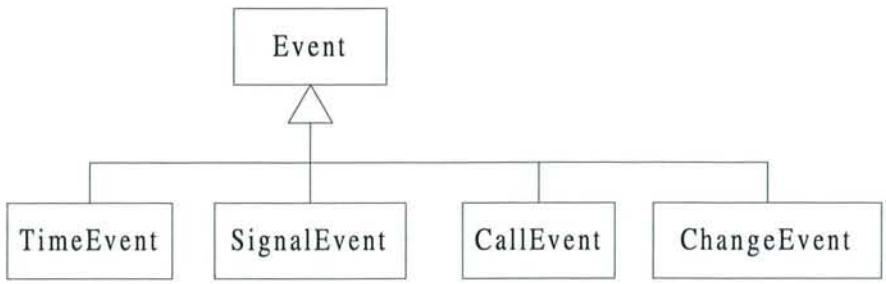

Figure 7. Types d'événements définis dans UML.

3.2. Aspects dynamiques de MMS

La gestion d'événements MMS fournit les mécanismes et services permettant d'informer les clients MMS de l'occurrence d'événements dans les serveurs. Un événement peut apparaître :

- à l'initiative d'un client (événement déclenché; network-triggered),

- à l'initiative de l'équipement physique modélisé par le VMD (événement scruté; event monitored).

Sur l'apparition de l'événement, les clients sont informés. Un événement peut impliquer l'exécution d'une ou plusieurs actions événementielles (des services MMS ou des actions locales automatiquement exécutées lors de l'apparition de l'événement considéré). L'événement MMS a deux fonctions essentielles : informer d'un changement d'état et synchroniser les applications MMS.

MMS représente l'information commune à un type d'événement par le biais de l'objet condition événementielle (EC ; Event Condition). Ce dernier spécifie la condition d'apparition d'un événement. L'apparition d'un événement peut survenir suite à un déclenchement effectué par le client et en utilisant le service Trigger Event (EC est de type Netwok Triggered) (Figure 8).

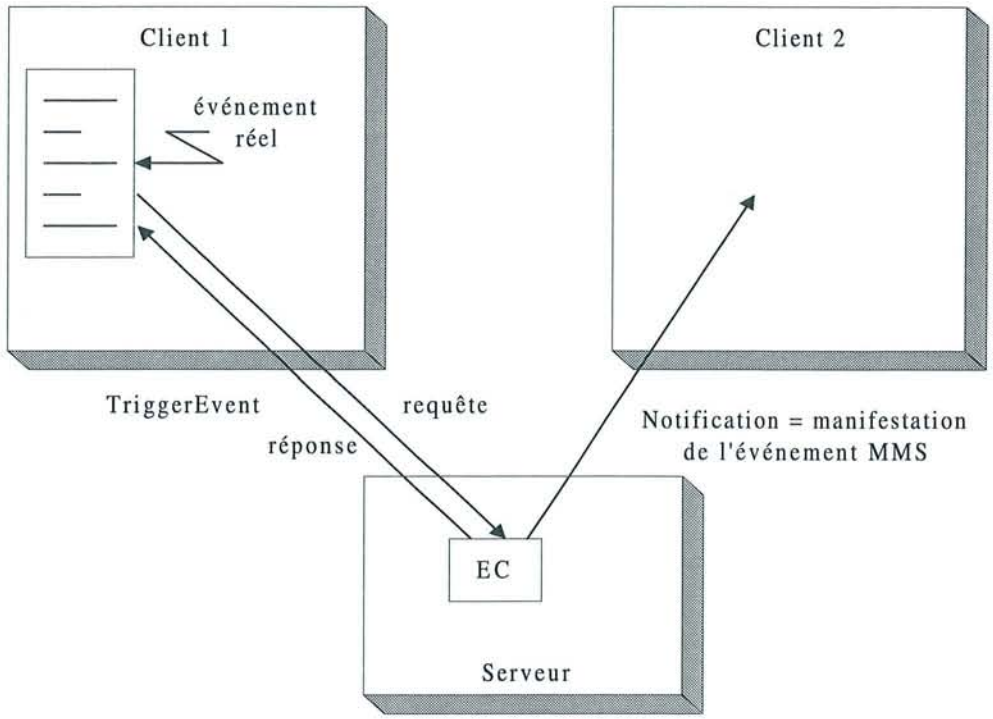

Figure 8. Evénement MMS déclenché (Castori, 1996).

L'apparition de l'événement peut également survenir suite à un changement de la valeur de l'attribut état (de Inactif à Actif ou de Actif à Inactif) de l'objet condition événementielle (spécifiant la condition de déclenchement de l'événement scruté) après une scrutation effectuée par le serveur de la variable booléenne qui lui est associée (Figure 9).

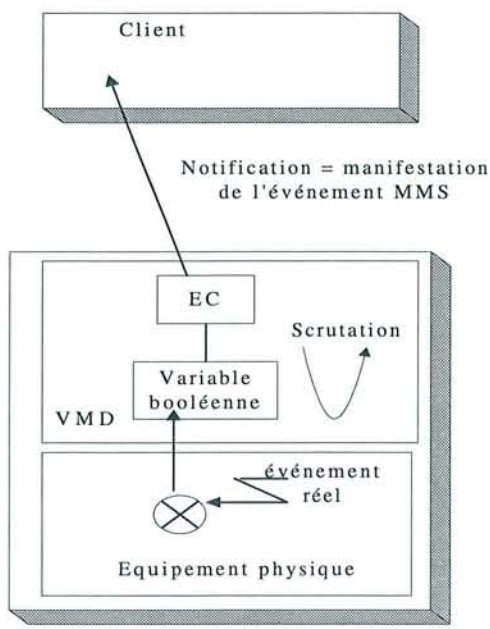

Figure 9. Evénement MMS scruté (Castori, 1996).

Il est à noter que dans le premier cas le déclenchement d'un événement peut s'effectuer localement (sans l'aide du client ; sans l'événement Trigger Event), mais la condition de son déclenchement n'est pas normalisée par MMS et c'est aux concepteurs d'en

décider. Dans le second cas, il est possible de créer l'objet condition événementielle malgré la non-existence de l'événement réel, mais c'est au client de déclencher l'événement et non pas le dispositif physique revenant ainsi au premier cas. L'objet condition événementielle scrutée est prédéfini, ce qui le différencie de celui qui est déclenché.

L'objet action événementielle représente l'action à effectuer sur le serveur lors de l'apparition de l'événement. Le lien entre l'événement et l'action événementielle est établi grâce à l'objet MMS enveloppe événementielle (event enrollment). Dans la norme MMS l'action événementielle est exécutée localement sur le serveur où se produit l'événement, mais une exécution distante de l'action sur un autre serveur est possible.

3.3. Conventions

Il est important de savoir si certains objets MMS peuvent être aussi générés à partir du modèle dynamique OMT. En se basant sur ce dernier le fait de passer à un nouvel état dépend de l'état courant, de l'occurrence de l'événement et si la condition est satisfaite. Comme résultat de ce changement, l'objet peut effectuer une action et peut également envoyer un événement à une autre classe d'objets. A titre d'exemple, nous pouvons présenter textuellement les conditions de franchissement d'une transition, d'un état à un autre, du modèle dynamique pour une classe «classe 1» comme suit :

Sur <événement 1 (attrib 1)> [si <condition>] appliquer <action> et/ou <envoi_événement 2 (attrib 2)> [classe 2]

Etant donné qu'il existe plusieurs façons au niveau de MMS pour définir l'équivalent du concept événement, il faut essayer de préciser plus en détail les choix possibles afin de pouvoir systématiser le passage du modèle dynamique OMT au modèle MMS. Ainsi, nous pouvons définir les règles de transformation suivantes des événements OMT, qui sont mieux définis dans UML, à ceux correspondant dans MMS :

3.3.1. Règle de transformation de l'événement $ChangeEvent_{omt}$

Cet événement va correspondre à l'événement MMS résultant suite d'un changement de la valeur (de faux à vrai) de la condition qui lui est associée de fausse à vraie. L'attribut portant sur l'expression booléenne au niveau OMT est transformé en un objet MMS condition événementielle scruté.

$$R_{ChangeEvent_{omt}} :$$
$$\| Trans(OMT, when\ (exp\ ression - booléenne)_{omt}) = CE_{mms_{scruté}}$$

3.3.2. Règle de transformation de l'événement $SignalEvent_{omt}$

Cet événement va correspondre, au niveau MMS et pour une classe destinataire comme étant Client, à une notification MMS avec la même signature.

$$R_{SignalEvent_{omt}} :$$
$$\| Trans(OMT, SignalEvent_{omt}) = Notification_{mms} \ /$$
$$signature\_Notification_{mms} = signature\_SignalEvent_{omt}$$

3.3.3. Règle de transformation de l'événement $CallEvent_{omt}$

Cet événement va correspondre, au niveau à une demande de service MMS avec la même signature.

$$R_{CallEvent_{omt}} :$$
$$\| Trans(OMT, CallEvent_{omt}) = Service_{mms} \ /$$
$$signature\_Service_{mms} = signature\_CallEvent_{omt}$$

L'interprétation de l'événement «CallEvent», qui sera assimilé à un service de communication, dépendra du choix du concepteur. Ainsi, c'est au concepteur de l'indiquer en précisant la sémantique (déterminer la fonction souhaitée) de «CallEvent» qui permet ensuite d'être transformé en service MMS (gestion de variables, gestion des invocations de programmes et gestion de sémaphores). Après une étude approfondie de la norme MMS, nous avons classé en 4 catégories les fonctions essentielles pour contrôler un système automatisé de production :

- catégorie A : lire, modifier (ou écrire) une information (ou une variable),

- catégorie B : exécuter, arrêter, reprendre ou réinitialiser une tâche,

- catégorie C : prendre ou libérer une ressource,

- catégorie D : superviser de manière passive.

La liste de fonctions que nous venons de présenter n'est pas exhaustive. Ces fonctions seront détaillées par la suite.

3.3.4. Règle de transformation de l'action associée à l'événement $action_{omt}$

Nous savons que suite à l'occurrence d'un événement, une action peut être exécutée. Une action du modèle dynamique correspond à un objet MMS Action Evénementielle.

$$R_{action_{omt}(\acute{e}v\acute{e}n)} :$$
$$\| Trans(OMT, action_{omt}) = AE_{mms} \ / \ AE_{mms}\_Nme = action_{omt}\_Nme$$

L'action événementielle est soit :

- l'exécution automatique d'un service MMS confirmé «se trouvant localement»,

- l'exécution d'une action locale définie préalablement,

- une action d'envoi d'une notification.

3.3.5. Règle de transformation concernant le lien événement-action

Nous considérons que le lien existant entre un événement et l'action lui est associé, à travers une transition entre deux états du modèle dynamique pour une classe d'objets donnée, pourrait correspondre à l'objet MMS EE (Enveloppe Evénementielle) qui établit le lien entre les objets MMS CE et AE (respectivement la condition événementielle et l'action événementielle).

$R_{évén_{omt}, action_{omt}(évén)}$:

$\| Trans(OMT, évén_{omt}, action_{omt}(évén)) = EE_{mms} \quad / \quad EE_{mms}\_Nme = Transit\_Nme$.

Notons que l'événement «TimeEvent» n'a pas de correspondance au niveau de MMS parce que MMS n'intègre pas les aspects temporels. Par conséquent, cet événement ne sera pas traité.

4. Fonctionnalités des systèmes automatisés distribués vues de la communication

Il existe au sein d'un système automatisé distribué quelques fonctionnalités classiques. La notion de réutilisation pourrait être prise en compte pour accroître la qualité du système final. La réutilisabilité passe par les deux étapes suivantes : l'étape d'acquisition (identification des parties réutilisable) et l'étape d'archivage (stockage des composants identifiés dans des bibliothèques afin de les réutiliser ultérieurement) des composants réutilisables. La première comporte trois parties :

- l'acquisition des composants d'analyse qui est constituée par la recherche des informations générales et leur regroupement dans un dossier de spécification réutilisable pour un domaine donné (composant domaine). Ici le niveau de réutilisabilité est le plus élevé par rapport au cycle de vie de développement.

- l'acquisition des composants de conception en isolant les constituants génériques parmi les descriptions de conception (composants services). Ainsi le modèle de la conception du programme (le code de la phase d'implémentation du système) pourrait être réutilisé,

- l'acquisition de composants logiciels et matériels concernant les conceptions détaillées. Ici, une identification de codes ayant un caractère de généralité et réutilisables est faite en vue d'implémenter le même service ou un autre. Ici la réutilisabilité prend les formes suivantes :

  - l'héritage : Il existe deux types d'héritage utilisés dans les langages orientés objet : héritage simple et héritage multiple. L'héritage est un mécanisme pour réutilisation par extension,

- liens binaires : Plutôt que d'insérer un appel direct à du code externe, il est plus sûr d'encapsuler son comportement dans une opération ou une classe. De cette manière, le code de la procédure ou du paquetage externe peut être modifié, et le code ne devra être modifié qu'à un seul endroit (Rumbaugh, 1995).

Nous venons de présenter d'une manière générale la notion de réutilisabilité (Coulange, 1996) et de son utilisation dans les différentes étapes de cycle de vie de développement du système. On peut rajouter à ce qui a été présenté ci-dessus les bibliothèques de composants orientés objet tout en confirmant qu'aucune méthode universelle pour faire des composants réutilisables existe. Il est à noter qu'il y a quand même des techniques permettant de fabriquer ces composants selon les différents types de formalismes utilisés : modèle objet et modèle dynamique.

Ceux qui ont contribué à la définition de la norme MMS ont défini certains services reflétant de manière non explicite des fonctionnalités des systèmes automatisés. Notons que de telles fonctionnalités sont perçues du point de vue du système de communication. Nous définissons, pour des raisons liées à la réutilisation, le comportement associé à ces fonctionnalités classiques (supervision avec les services MMS appropriés, contrôle de ressources et le groupe de services de gestion de sémaphore et gestion de tâches avec le groupe de services de gestion d'invocation de programme) des applications industrielles. Un tel comportement pourrait représenter un composant diagramme d'états de la même façon qu'un composant classe (Mansour, 1998c). Dans le diagramme d'états-transitions d'une classe donnée, le fait de trouver les sous-diagrammes réutilisables signifie l'identification des sous-ensembles stables d'états. Un tel ensemble est constitué d'états et d'événements tels que, quel que soit l'événement de l'ensemble, son traitement fait rester dans un des états de l'ensemble (Figure 10).

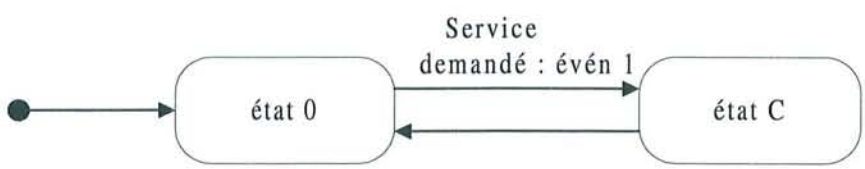

Figure 10. Diagramme d'états et les relations du composant diagramme d'états-transitions.

Afin que ce sous-diagramme soit utilisable, il faut le compléter par l'indication d'événements qui permettent d'en sortir. Bien qu'après avoir terminé le traitement de quelques événements (Figure 11), la classe peut rester dans un des états constituant le composant diagramme d'états-transitions.

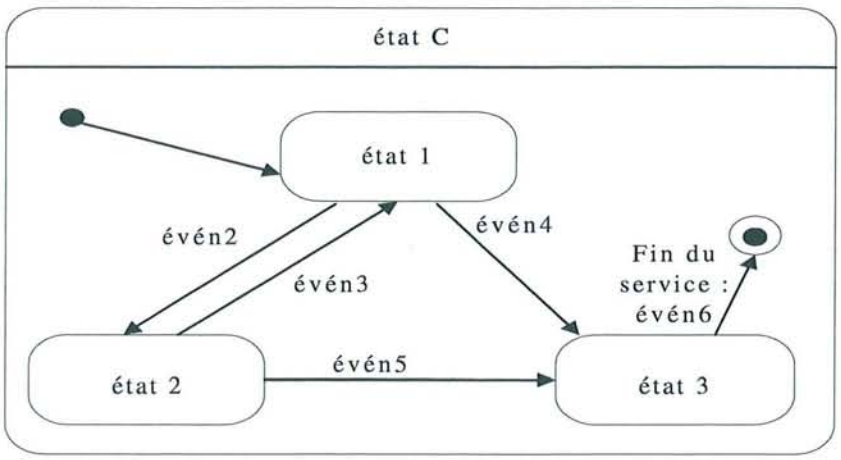

Figure 11. Vue détaillée du composant diagramme d'états «état C».

Puisque certains services peuvent être invoqués par plusieurs clients (applications ou utilisateurs), ces composants automates seront très utiles lors de la description des services dans les différents contrôleurs d'équipements et ceux des utilisateurs. Les composants automates seront stockés dans une bibliothèque et seront utilisés ultérieurement comme des briques de base dans le développement de l'application considérée. En ce qui nous concerne, ces composants modéliseront le comportement des différentes fonctions à réaliser par les applications industrielles, de même que leur transcription dans le modèle MMS en terme de services MMS.

4.1. Fonctionnalité : Gestion de ressources

4.1.1. Composant diagramme d'états-transitions

Un équipement muni d'une fonctionnalité de gestion de ressource, peut être modélisé par le diagramme d'états représenté sur la figure 12.

Le graphe d'états (Figure 12) est composé de deux états : Libre et Reserve, qui sont les états potentiels d'une ressource donnée.

Dans l'état 'Libre', si l'événement reçu est la demande de prise de ressource (*Demande_Prise__Ressource*), alors la ressource sera attribuée au demandeur et par conséquent sera sous son contrôle. Ainsi une mémorisation de l'identité du demandeur sera effectuée (P=demandeur) et un message de retour (*Acquittement_Prise__Ressource* (*OK*)) lui sera envoyé. La ressource devient alors réservée (état Reserve).

Dans l'état 'Reserve' si l'événement reçu est (*Demande_Prise__Ressource*) d'un autre demandeur, la demande sera alors refusée et un message (*Acquittement_Prise__Ressource* (*NACK*)) lui sera envoyé.

Si l'événement reçu est (*Demande_Liberation_Ressource*) mais provient d'un autre demandeur que celui qui contrôle la ressource (P != demandeur), alors la libération de ressource sera refusée et un message (*Acquittement_Liberation__Ressource* (*NACK*)) lui sera envoyé.

Si le demandeur est bien celui qui possède la ressource (P = = demandeur), alors un message (*Acquittement_Liberation__Ressource (OK)*) lui sera envoyé et la ressource sera libérée (revient à l'état 'Libre').

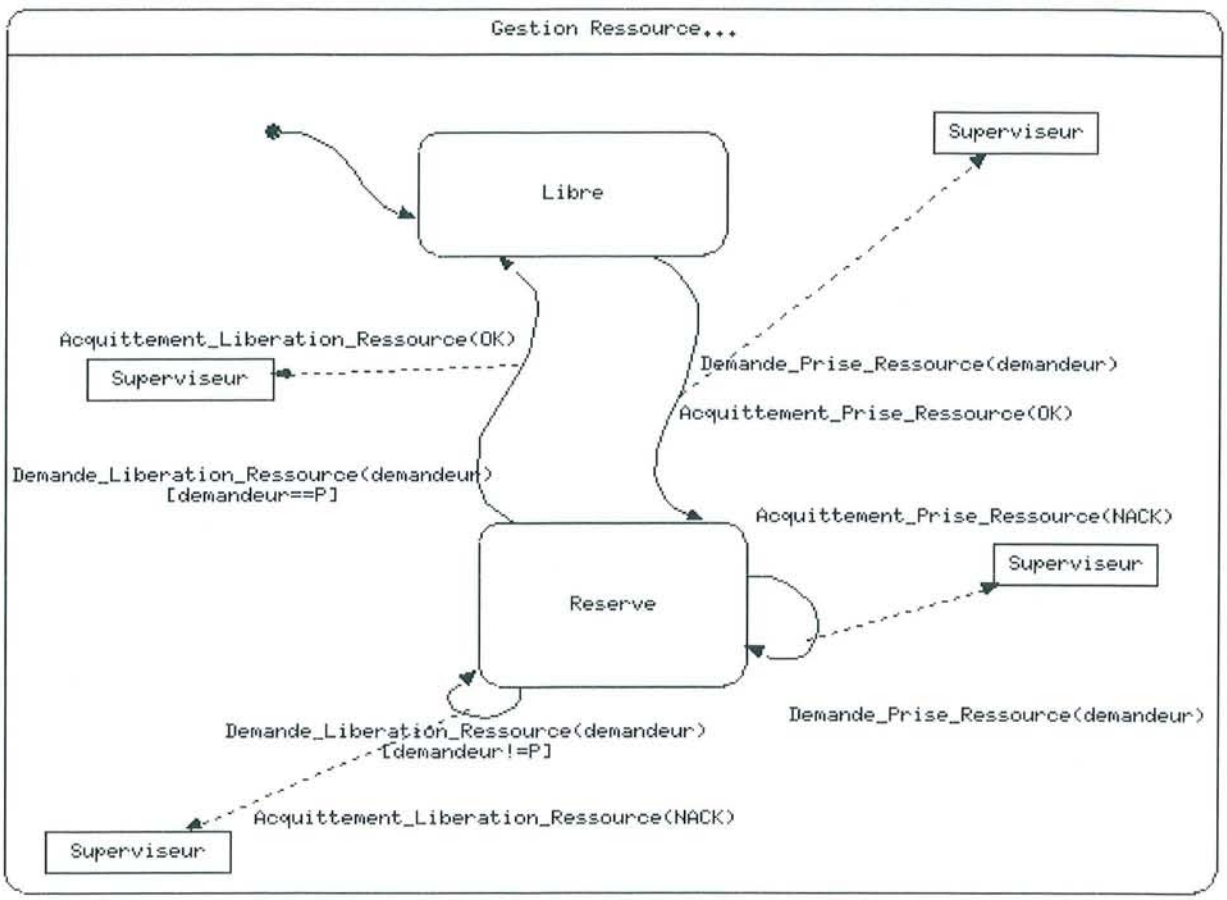

Figure 12. Le composant diagramme d'états «Gestion Ressource».

Dans un premier temps nous transcrirons le composant automate «Gestion Ressource» sur l'automate à états finis au niveau MMS. Ceci en prenant en considération que nous avons à notre disposition tous les services MMS (classe de conformité CC_3).

4.1.2. Transcription du composant sur les services MMS

La figure 13 présente le composant automate «Gestion Ressource» tout en reflétant la gestion des sémaphores. La demande *Demande_Prise_Ressource* est traduite par *TakeControl_REQ*. La réponse négative suite à l'indisponibilité de la ressource *Acquittement_Prise_Ressource (NACK)* sera *TakeControl_Ack (NACK)*. La réponse positive *Acquittement_Prise_Ressource (OK)* sera *TakeControl_Ack (OK)*. En ce qui concerne la demande de libération de la ressource *Demande_Liberation_Ressource* sera transcrite par *RelinquishControl_REQ*. Après la libération de cette ressource, une réponse positive *Acquittement_Liberation_Ressource (OK)* sera notifiée au demandeur. Cette réponse sera transcrite par *RelinquishControl_Ack (OK)*.

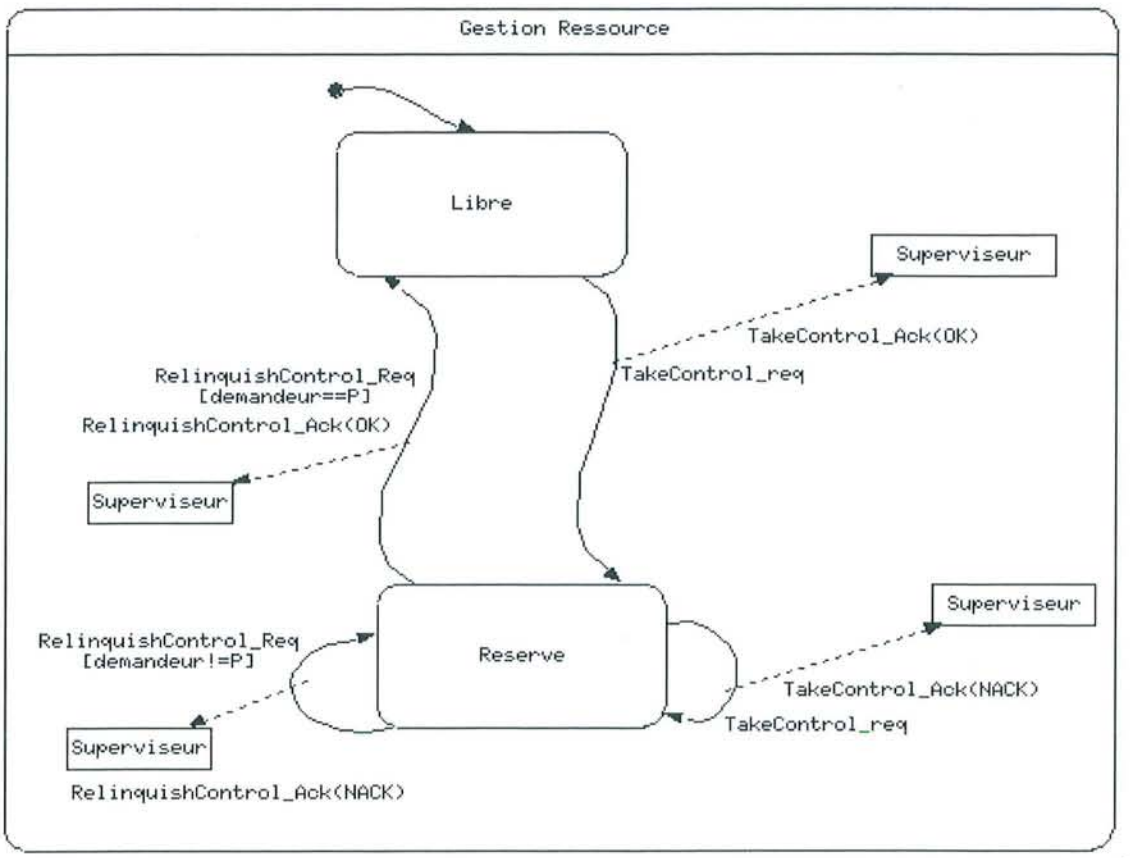

Figure 13. Transcription du composant «Gestion Ressource» sur les services MMS (cas tous les objets et services MMS disponibles).

Ainsi la transcription du composant sur les services MMS représente également une modélisation du comportement des objets MMS en terme de services MMS.

4.2. Fonctionnalité : Contrôle de tâche

4.2.1. Composant digramme d'états-transitions

Une tâche à réaliser par l'équipement est une activité séquentielle qui nécessite un certain temps d'exécution. Cette tâche peut être interrompue. Nous présentons le composant automate associé à la gestion des tâches. Les événements présents dans le composant contrôlent l'exécution de la tâche : exécuter, suspendre, reprendre, réinitialiser. Trois états principaux dans lesquel une tâche peut se trouver : Au Repos, En Execution et En Arret. Le nom de la tâche représente le paramètre commun aux demandes de manipulation de la tâche à contrôler. La figure 14 représente le composant automate «Gestion Tâche».

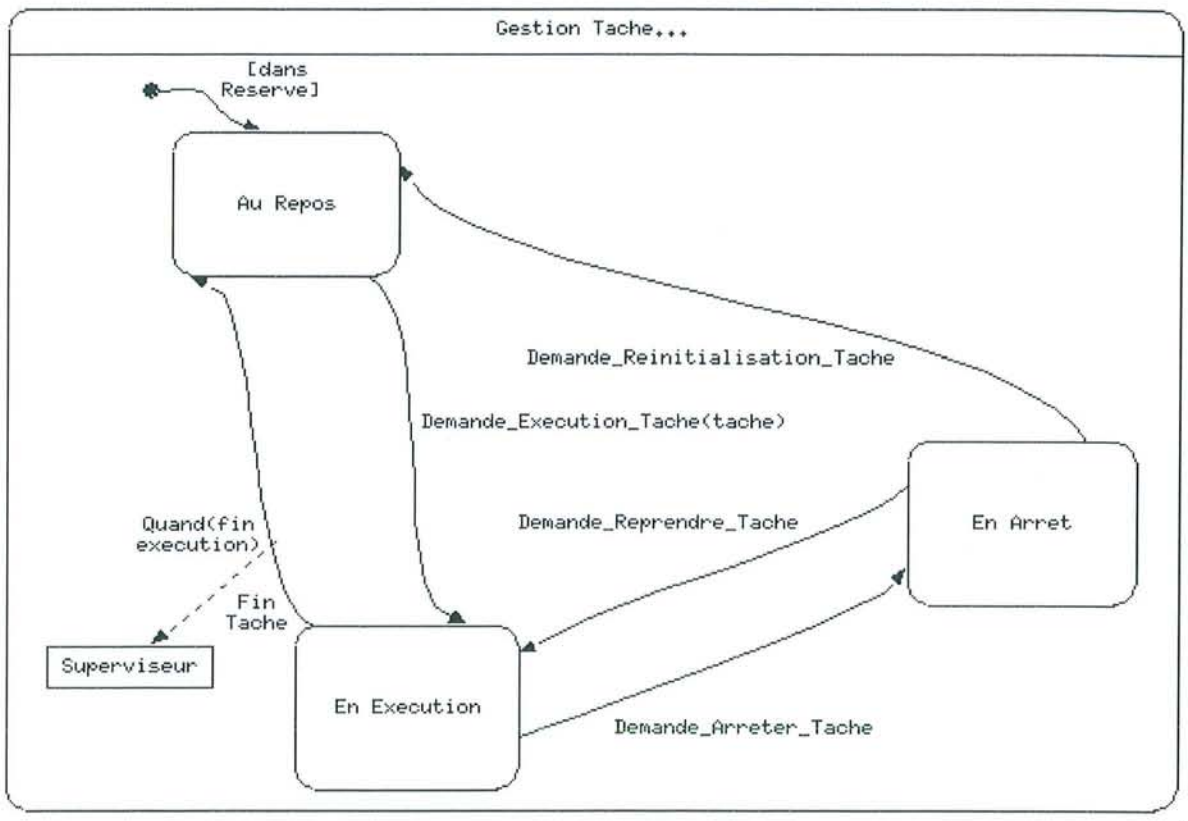

Figure 14. Le composant diagramme d'états «Gestion Tâche».

4.2.2. Transcription du composant sur les services MMS

Les demandes de contrôle d'exécution des tâches (Figure 15) ou des programmes seront transformées en demandes de services de gestion de Programmes Invocation MMS. Ainsi *Demande_Execution_Tâche* devient la demande *Start_REQ*. La demande *Demande_Arreter_Tâche* devient la demande *Stop_REQ*. *Demande_Reprendre_Tâche* deviendra *Resume_REQ*. Enfin, la demande *Demande_Reinitialiser_Tâche* sera transformée en une demande *Reset_REQ*.

Dans l'état En Execution et quand la valeur de la variable booléenne associée à la fin d'exécution d'une tâche passe à vrai, un changement passage vers l'état Au Repos se produit. Le message envoyé Fin Tâche se transforme en une notification MMS *EventNotification*

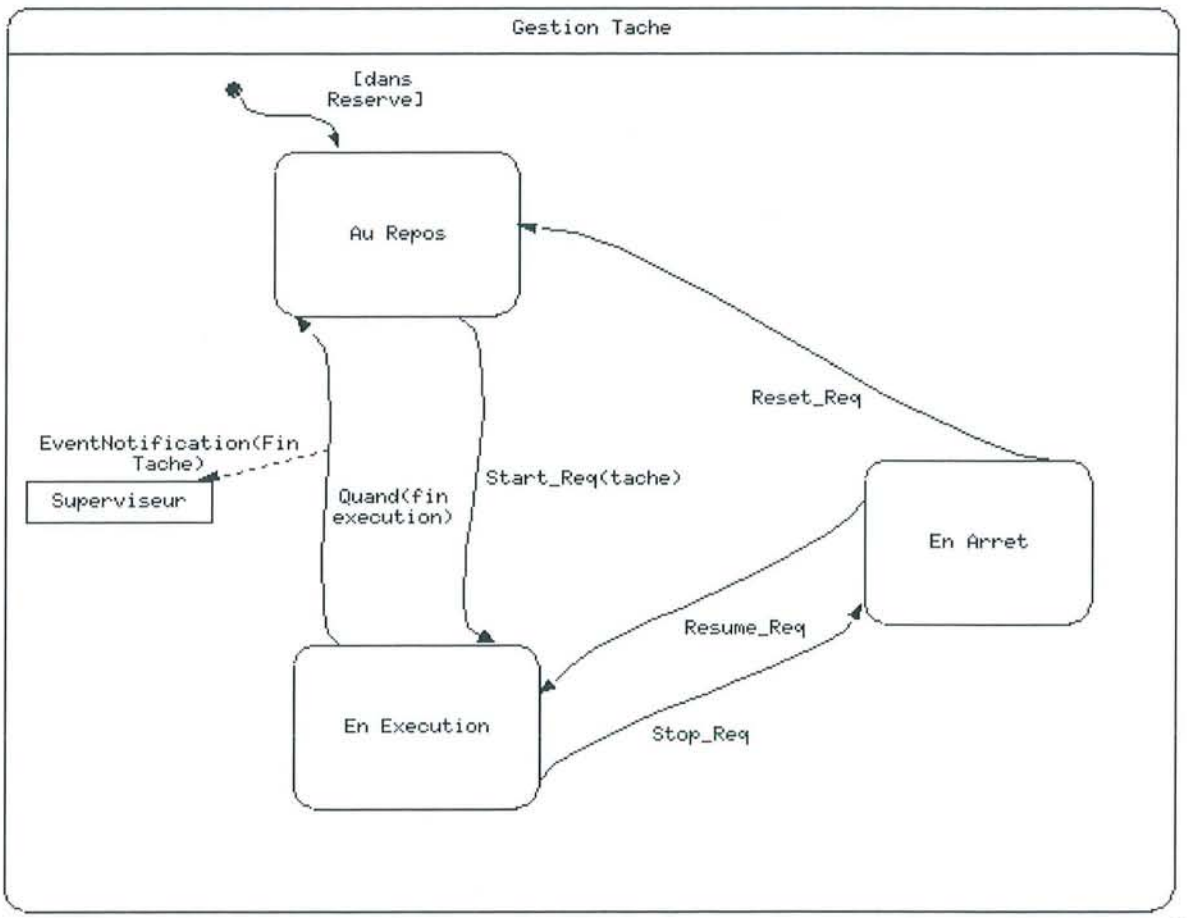

Figure 15. Transcription du composant «Gestion Tâche» sur les services MMS (tous les objets et services MMS sont disponibles).

Les fonctions que nous venons de présenter concernent plus particulièrement un équipement jouant le rôle d'un serveur MMS. Mais rappelons que les clients sont également concernés par ces fonctions et ce sont eux-mêmes qui exigent la réalisation, par le serveur, de telles fonctions. Il faut qu'ils soient capables d'invoquer les services qui sont associés au serveur. Ainsi par l'intermédiaire des services MMS les clients peuvent contrôler à distance les tâches associées au serveur et également d'accéder à ses ressources.

4.3. Fonctionnalité : Supervision passive

4.3.1. Composant digramme d'états-transitions

Cette fonctionnalité est associée aux clients et permet de superviser à distance l'équipement serveur modélisé par le VMD. Concernant le graphe d'états générique d'une telle fonctionnalité (Figure 16), des événements ou des demandes de services seront envoyés aux serveurs qui lui sont associés. Ces événements sont :

- *Demande_Info_Etat* afin de récupérer l'information sur l'état de l'autre classe lui est associée,

- *Demande_Info_Identité* afin de récupérer l'information sur l'identité de l'autre classe lui est associée,

- *Demande_Infos_Generales* afin de récupérer les informations générales concernant l'autre classe lui est associée.

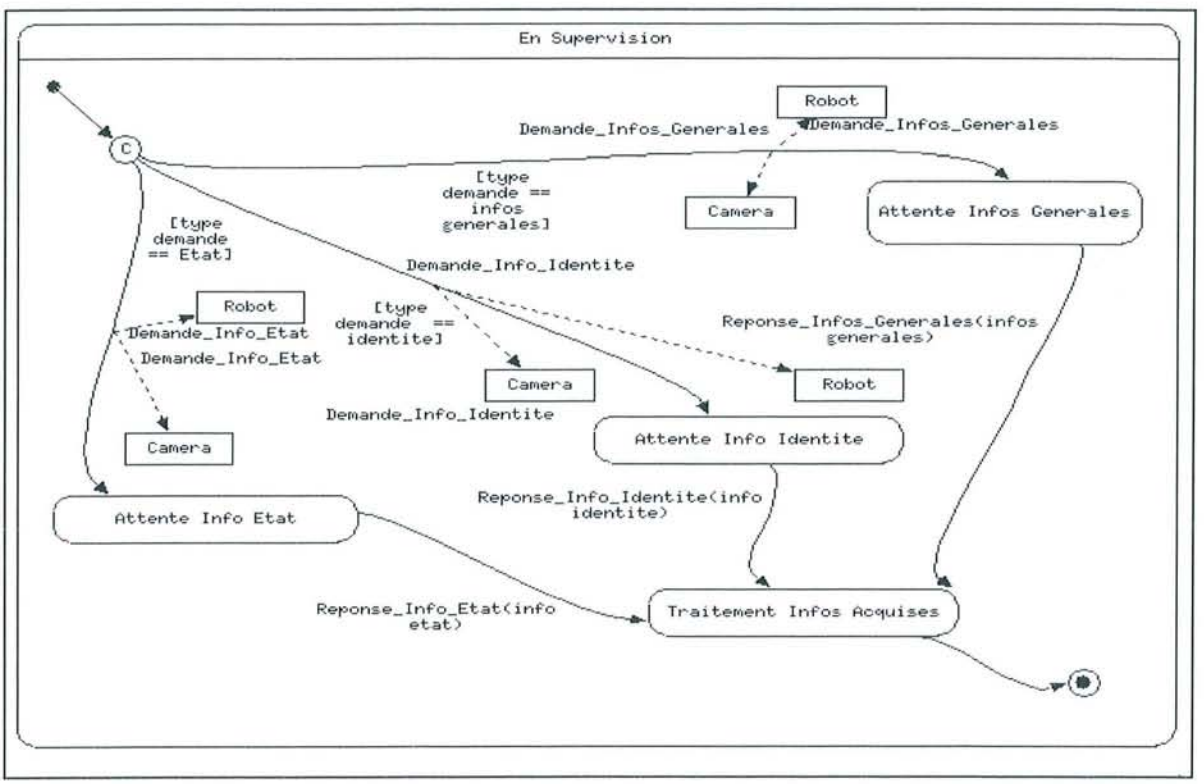

Figure 16. Le composant diagramme d'états «Supervision».

Dès que ces informations ont été récupérées, le client peut effectuer les traitements qui lui sont nécessaires (i.e. afficher les informations récupérées sur l'écran ou les archiver).

4.3.2. Transcription du composant sur les services MMS

En projetant le composant diagramme d'états «Supervision» en MMS et en prenant en considération que la classe de conformité de la classe reconnue en tant que client est CC_2 'complet, on obtient ce qui est représentée sur la figure ci-après (Figure 17). La demande *Demande_Info_Etat* devient *Status_REQ*. La réponse *Reponse_Info_Etat* envoyée par le serveur sera celle du *Status_RSP (info status)*. La demande *Demande_Info_Identite* devient la demande du service *Identify_REQ*. La réponse à la précédante demande sera *Identify_RSP (info identity)*. Enfin, la demande *Demande_Infos_Generales* devient *GetNameList (info list)*.

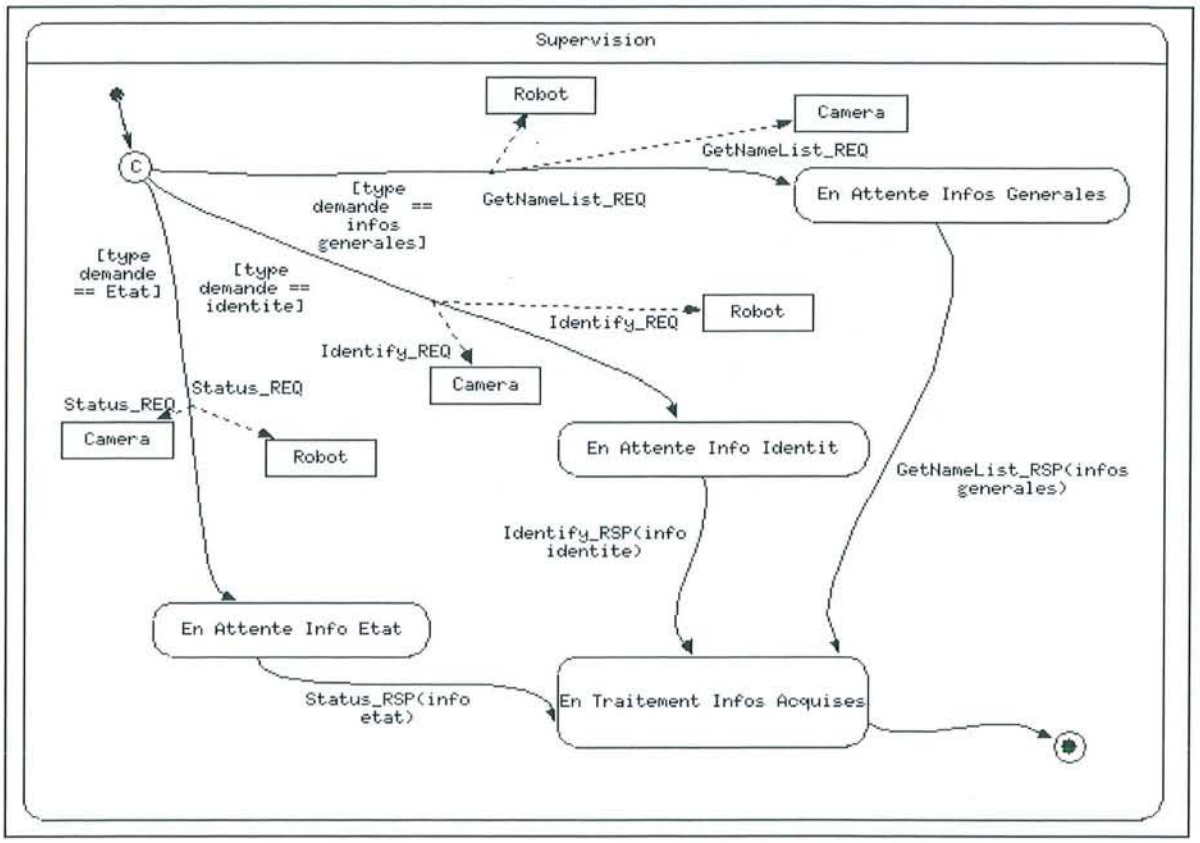

Figure 17. Transcription du composant «Supervision» sur les services MMS (cas tous les objets et services MMS disponibles).

5. Classes de conformité (Conformance Classes)

5.1. Offre constructeurs

La majorité de constructeurs n'implémentent, dans le serveur MMS, qu'une partie des objets et des services définis dans la norme. Ainsi l'offre concerne toujours un sous-ensemble d'objets et services normalisés. Ceci peut être dû à des problèmes de coût, mais aussi à des problèmes d'incompréhension de la norme MMS, ... Il existe un certain nombre d'offres constructeurs. On peut en citer quelques-unes :

- offre constructeur SIEMENS :

  - coupleurs supportant MMS : coupleur MAP, coupleur Ethernet et coupleur Profibus,

  - services MMS implémentés : services de gestion du contexte, gestion du VMD, gestion de variables, gestion de domaines et gestion d'instances de programmes,

  - services additionnels : transfert série assurant une compatibilité avec les automates Sinec AP1.0 et une interface HTB assurant une compatibilité avec des raccordements PC utilisant ancienne version SIMATIC,

- Offre constructeur Schneider :

  - Télémécanique

    - 2 coupleurs supportant MMS : coupleur Ethernet Client/Serveur MMS et Client/Serveur UNI-TE,

    - services MMS implémentés : services de gestion du contexte, gestion du VMD, gestion de variables, gestion de domaines (1 domaine = 1 UC) et gestion des instances de programmes,

    - au niveau terrain (FIP) : services MPS et sous-ensemble MMS.

  - APRIL : Ethernet April 5000 et 7000, 26 services MMS sont implantés,

  - NUM : Ethernet sur NUM 1060.

- D'autres offres constructeurs :

  - Allen Bradley : MAP (PLC-5/250, 42 services MMS sont implantés) et Ethernet,

  - AEG MODICON : MAP (32 services MMS), Ethernet sur PC (tous les services MMS, logiciel SISCO),

  - GMFANUC MAP : MAP (robot A600, S10, gestion de contexte + variables) et Ethernet,

- Des solutions Windows existent :

  - AEG MODICON (Ethernet) + AXS4-Windows (SISCO) : contexte, variables, fichiers,

  - SIEMENS (Ethernet) : tous les services implantés sur les automates,

  - Ethernet (3COM, ...) + AXS4-Windows (SISCO) : couches logicielles 3 à 7,

  - développement en cours d'une DLL (Dynamic-Link Library) pour panorama.

La liste que nous venons de présenter n'est pas exhaustive. Afin de satisfaire une application donnée, il s'avère qu'un sous-ensemble de fonctionnalités peut suffire. Par conséquent, il faut prendre en compte lors de la détermination des objets MMS structurant le VMD, la classe de conformité de l'équipement (Figure 18).

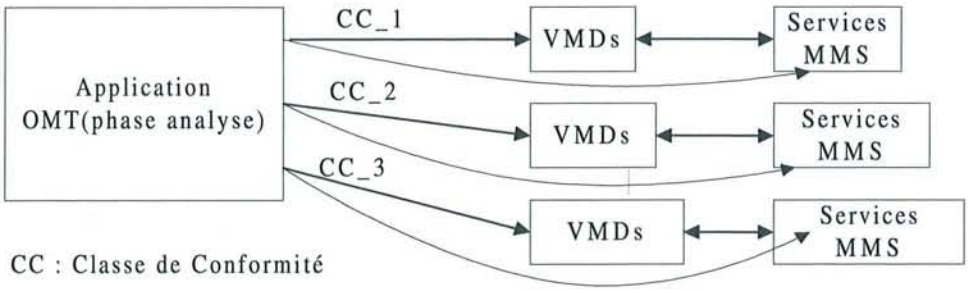

Figure 18. Processus suivi pour la génération du système de communication.

On peut définir la classe de conformité d'un dispositif comme l'ensemble des objets et services MMS disponibles sur l'équipement considéré. Un matériel avec une classe de conformité variable, limitera la représentation de ses ressources par des objets de la classe variable dans le VMD le modélisant. Ainsi les services disponibles sont ceux de la gestion de variables (Read, Write, InformationReport, ...). Cette situation aura une influence sur la projection du modèle OMT sur le modèle MMS. De plus, elle aura une influence sur les conventions proposées dans les sections précédentes.

5.2. Classes de conformité et les objets et services MMS générés

En survolant les offres constructeurs, présentée dans la section 3.1, et en essayant d'en tirer profit nous pouvons extraire 3 classes de conformité. Pour ce faire, nous procédons à un découpage ou une hiérarchisation artificielle (Figure 19) en définissant une classe abstraite appelée *Classe-Conformité* et 3 sous-classes *Classe-Conformité_1*, *Classe-Conformité_2* et *Classe-Conformité_3*.

Notons que les services de gestion de contexte de l'environnement MMS existent pour toutes les classes de conformité que nous avons définies et par conséquent nous les avons intégrées dans la classe abstraite *Classe-Conformité* en tant que propriétés communes à toutes ses sous-classes.

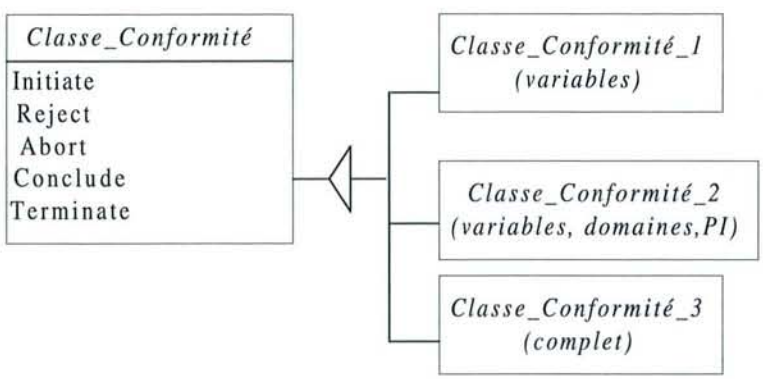

Figure 19. Hiérarchisation des classes de conformité.

On peut détailler un peu plus ces classes comme suit :

- *Classe-Conformité_1* (classe de conformité de base) : avec une telle classe, le VMD modélisant l'équipement supporte seulement les objets variables MMS. Par

conséquent seuls les services de gestion de variables seront disponibles sur l'équipement (Read, Write et InformationReport),

- *Classe-Conformité_2* (classe de conformité moyenne) : Dans ce cas les ressources et les fonctionnalités de l'équipement réel modélisé par un VMD seront représentées par des variables, domaines ou des instances de programmes (PI). Par conséquent les services disponibles seront ceux de la gestion de variables, gestion de domaines et gestion des instances de programmes,

- *Classe-Conformité_3* (classe de conformité évoluée) : le VMD modélisant l'équipement dispose de tous les objets MMS définis dans la norme. Ainsi tous les services (86 services) MMS définis dans la norme MMS seront également disponibles.

Nous avons recensé les classes de conformité des offres constructeurs déjà présentées et nous les avons classées dans le tableau suivant (Tableau 2). Dans ce tableau nous utilisons : *CC_1* (Classe Conformité «Variables»), *CC_2* (Classe Conformité «Variables, Domaines et Invocation Programmes») et *CC_3* (Classe Conformité Complet) pour référencer aux classes de conformité retenues.

Offre constructeur	*CC_1*	*CC_2*	*CC_3*
SIEMENS Coupleurs : MAP, Ethernet et Profibus.		X	
Télématique Coupleurs : Ethernet Client/Serveur MMS et Client/Serveur UNI-TE.		X	
APRIL Ethernet April 5000 et 7000		X	
Allen Bradley MAP (PLC-5/250) et Ethernet.		X	
AEG MODICON MAP, Ethernet sur PC			X
GMFANUC MAP	X		
NUM 10x0		X	
AEG MODICON +AXS4 (SISCO)	X		
SIEMENS (Ethernet)		X	
Ethernet (3COM, ...) + AXS4 (SISCO)		X(Serveur)	X(Client)

Tableau 2. Classification d'offres constructeurs par rapport à leur classes de conformité.

5.3. Les conventions proposées et l'influence des classes de conformité

La projection du modèle OMT sur le modèle MMS doit s'adapter (ou sera contrainte) à la classe de conformité de l'équipement modélisé. Ainsi, la structure interne du VMD de l'équipement considéré regroupe des objets MMS disponibles accompagnés de leurs services MMS respectifs. Ce constat évite une projection faite sans prendre en compte la possibilité de l'implémenter effectivement et ainsi d'être loin des capacités du dispositif réel.

Mais rappelons que quoiqu'il en soit il y a une différence de pouvoir sémantique des deux formalismes OMT et MMS. Aussi, il y aura des concepts OMT qui n'auront pas de correspondances directes dans MMS. Dépasser complètement le fossé sémantique et le faire disparaître sort de nos compétences.

Regardons de près les conventions que nous avons proposées afin de déterminer le choix de la nature des objets MMS issus de la transformation d'OMT en MMS. En ce qui concerne la classe de conformité Classe_Conformité_3, elle nous conduit aux conventions proposées. Nous étudierons les deux cas restants :

- si la classe de conformité de l'équipement est Classe_Conformité_1, alors cet équipement supporte les objets MMS variables et leurs services associés. Par conséquent toute information OMT sera transformée en objets MMS variables nommées.

- si la classe de conformité de l'équipement est Classe_Conformité_2, alors cet équipement supporte les objets MMS variables, domaine et instances de programme avec leurs services associés.

5.3.1. Convention concernant le domaine

Si la classe de conformité de l'équipement en considération est Classe_Conformité_1 (ou CC_1), alors toutes les classes d'objets modélisant la structure de son VMD seront transformées en classe variable structurée. Par conséquent la classe d'objets qui devrait être transformée en une classe domaine, va correspondre à une classe variable structurée MMS et la convention (dans la sous section 2.2.2. du présent chapitre) devient alors :

$$\forall Cl_{Référence}, Cl_{omt_1}, Cl_{omt_2} \in Cl_{diag}, \forall At_{Omt_i} \in \alpha\left(Cl_{omt_2}\right)\ i = 1..n\ /\ \eta_{Cl_{Référence}, Cl_{omt}}\ ,\ \delta_{Cl_{omt_1}, Cl_{omt_2}} \Rightarrow$$

$$\left\| Trans\left(OMT, Cl_{omt_2}\right) = VariableStructurée_{mms} \qquad (application\ de\ la\ règle\ R_{class})\ avec \right.$$

$$VariableStructurée_{mms}\_Nme = Cl_{omt}\_Nme$$

$$= \{$$

$$At_{omt_1}\_Nme$$

$$At_{omt_2}\_Nme$$

$$.....$$

$$At_{omt_n}\_Nme$$

$$\}$$

$$VariableStructurée_{mms}\_Typ = \{ At_{omt_1}\_Typ \; At_{omt_2}\_Typ \; ..... \; At_{omt_n}\_Typ \}$$

Donc nous étudions maintenant les classes de conformité d'un point de vue de services disponibles sur les matériels à notre disposition.

5.3.2. Transcription du composant automate «Gestion Ressource»

Avant de transcrire le composant automate «Gestion Ressource» sur les services MMS, Nous avons besoin de connaître les services MMS réellement implantés sur l'équipement considéré et par conséquent sa classe de conformité du point de vue services mis à disposition des clients potentiels. Mais rappelons qu'il est nécessaire de connaître également la classe de conformité des clients.

Ainsi, si la classe de conformité de l'équipement en question est CC_1 «variable» ou CC_2 «variables, domaines et invocation des programmes» alors les services de gestion de variables seulement seront disponibles (Figure 20). La requête *Demande_Prise_Ressource* correspondra à la demande de service MMS d'écriture de variable *Write_REQ (etat_ressource, 1)*. La réponse négative (quand la ressource a été déjà prise par un autre Client), *Reponse_Prise_Ressource (NACK)* envoyé au demandeur correspondra à une notification *Write_RESULT (NACK)* du résultat de telle sorte que cette variable n'a pas pu être modifiée et par conséquent l'écriture n'a pas eu lieu.

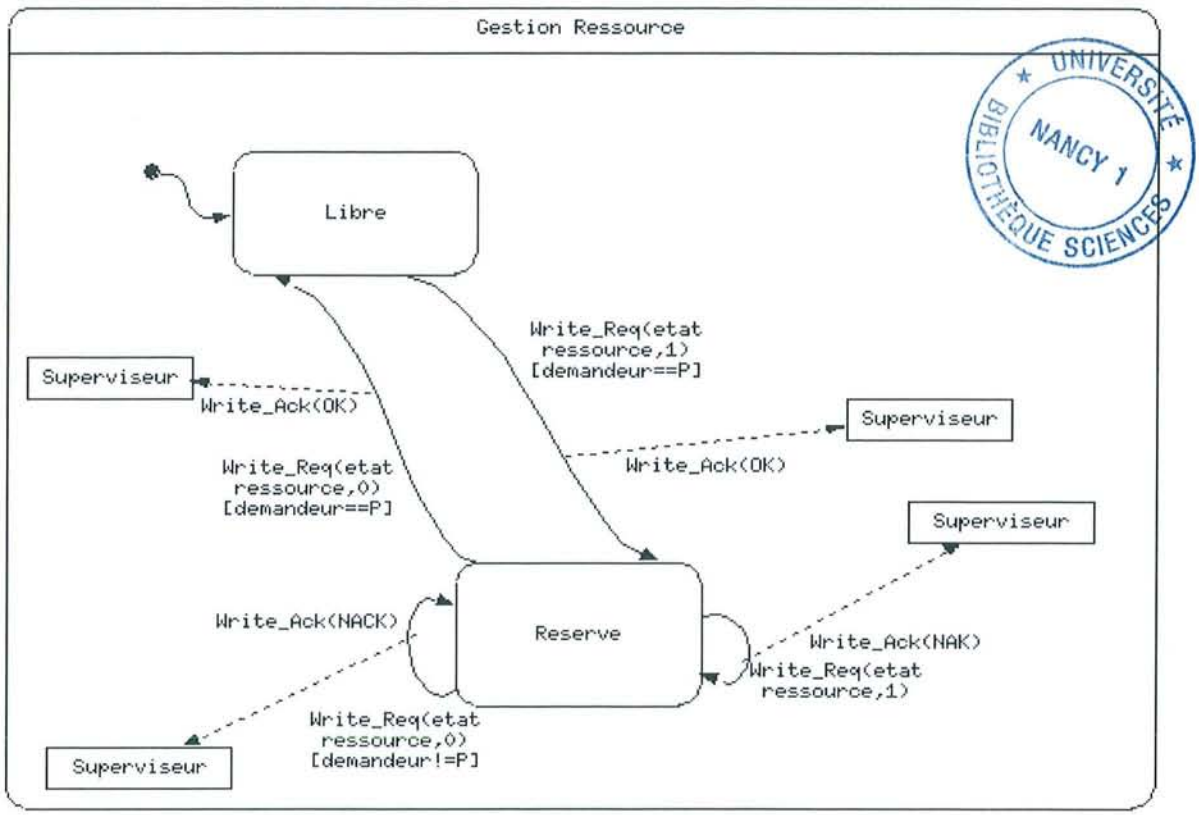

Figure 20. Transcription du composant «Gestion Ressource» sur les services MMS (CC_1, CC_2)».

Si la ressource était libre, une allocation de celle-ci est réalisée et la réponse positive *Reponse_Prise_Ressource (OK)* sera envoyée au demandeur, ce qui correspondra au service MMS d'écriture de variable *Write_RESULT (OK)* qui exprime le bon déroulement de l'opération d'écriture. La demande de libération de la ressource *Demande_Liberation_Ressource* sera traduite par le service MMS d'écriture de variable *Write_REQ (etat_ressource, 0)*. Ensuite, une réponse positive *Reponse_Liberation_Ressource* sera transcrite par *Write_Result (OK)* et envoyée au demandeur.

De plus une nouvelle variable est définie (appelée état_ressource) pour pouvoir y accéder via les services de gestion de variables.

5.3.3. Transcription du composant automate «Gestion Tâche»

Si la classe de conformité de l'équipement considéré est CC_1 «variables», ces demandes de services de contrôle des tâches seront représentées par les services de gestion des variables nommées MMS (Figure 21). Une variable sera associée à chaque tâche ou programme exécutable. Cette variable est une variable de type entier et prend par exemple les valeurs de 1 à 4. La demande *Demande_Execution_Tâche* correspondra à la demande Write_REQ ($Var\,Prog_i$, 1). La demande *Demande_Arreter_Tâche* correspondra à la demande Write_REQ ($Var\,Prog_i$, 2). *Demande_Reprendre_Tâche* sera remplacée par Write_REQ ($Var\,Prog_i$, 3), alors que *Demande_Reinitialiser_Tâche* sera transformée en

Write_REQ ($Var\,Prog_i$, 4). i représente le nombre de variables associées aux tâches que l'équipement peut exécuter.

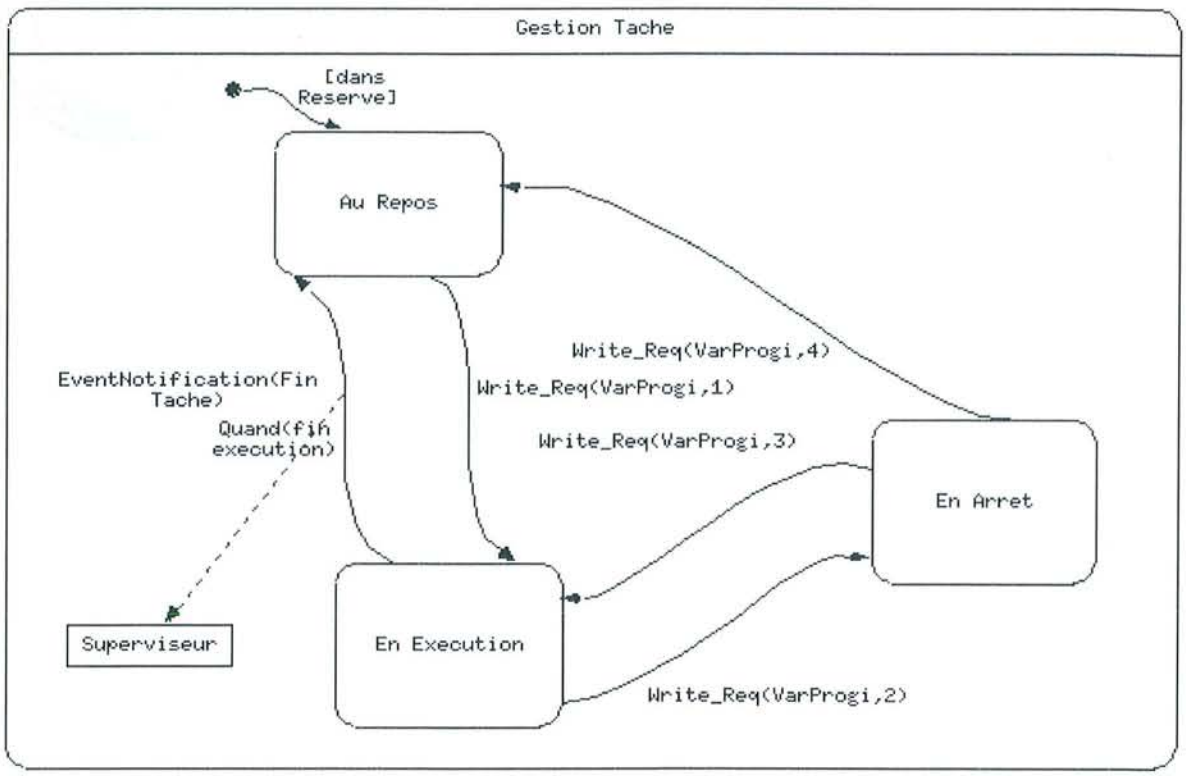

Figure 21. Transcription du composant «Gestion Tâche» sur les services MMS (CC_1).

De plus, nous aurons autant de variables nommées (appelées $Var\,Prog_i$) que d'invocations de programmes. L'exécution des invocations de programmes sera contrôlée via le service d'écriture MMS (Write).

5.3.4. Transcription du composant automate «Supervision»

La figure suivante (Figure 22) montre la projection du composant automate «Supervision» en MMS et dans le cas où la classe jouant le rôle client serait en mode dégradé et supporte la classe de conformité CC_1 'variables'. Les demandes d'informations de l'état du serveur ou de son identité ou bien de ses objets seront transformées en demande de lecture de variables *Read_REQ*. Les réponses correspondantes seront *Read_RSP*.

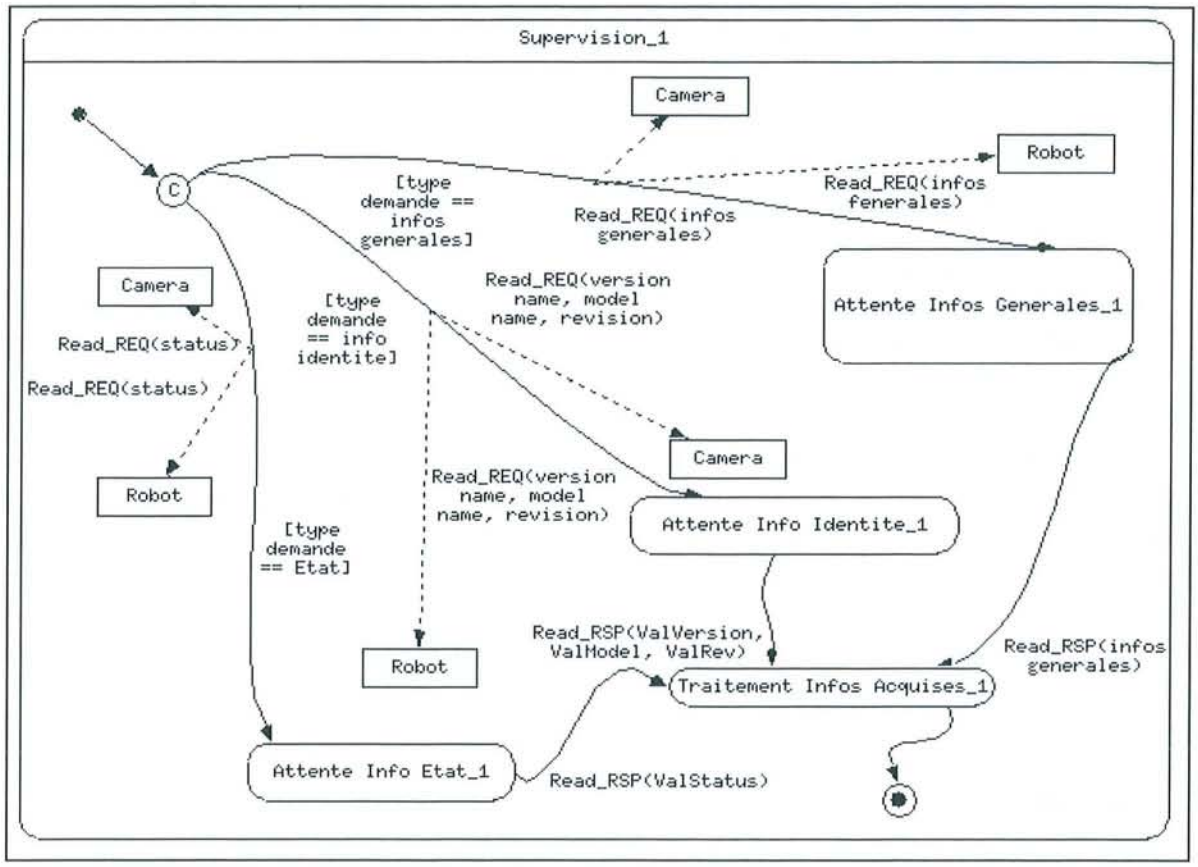

Figure 22. Transcription du composant «Supervision» sur les services MMS (CC_1)».

Dans la suite, nous nous basons sur les critères de choix et plus particulièrement sur les critères fonctionnels afin de définir l'ensemble des services MMS à prendre en compte pour la réalisation de l'application étudiée.

6. Critères de choix des objets MMS générés

Une fois que la structure du VMD est déterminée, les services manipulant les objets MMS le seront également. Mais dans le cas où plusieurs choix seraient possibles (dérivés des classes de conformité disponibles), lequel choisir et pourquoi ?

La réponse à cette question dépend du caractère critique accordé à un critère donné. On peut présenter quelques critères tels que :

- critère fonctionnel : la maîtrise des fonctionnalités attendues ou à réaliser par l'équipement pourrait influencer le choix de l'objet et du service générés,

- critère temporel : les performances temporelles peuvent être prises en compte pour le choix des objets et services MMS à implanter (cas de disponibilité de plusieurs alternatives). Dans (Daoudi, 1994), Daoudi a effectué des mesures de performance des services MMS sur une implémentation MMS-EASE de SISCO. Ceci dans le but d'analyser le comportement des services MMS, en terme de temps de réponse, en fonction de deux critères qui sont la configuration du système de communication (nombre de stations reliées au réseau et nombre d'associations établies entre les

entités MMS) et la charge de l'application. Selon le type de service choisi, le temps de réponse se variait différemment,

- critère structurel : optimiser l'espace mémoire disponible car l'utilisation de plusieurs services et objets dans une même application nécessite beaucoup de mémoire. Cela peut être aussi prendre le minimum commun de services MMS parmi l'ensemble de matériels disponibles,

- critère de complexité des services à mettre en œuvre : afin de choisir le service ou l'ensemble de services MMS les plus simples.

On peut simplement se justifier par la volonté de profiter au maximum de la richesse mise à notre disposition par la norme MMS. Il est évident que, parfois, le choix est arbitraire et dans ce cas c'est l'utilisateur qui prend le relais et choisit ce qui convient le mieux et ce qui satisfait ses besoins.

A noter que dans nos travaux seul l'aspect classe de conformité est étudié. Les différents critères définis ci-dessus ne sont pas pris en compte dans notre processus de transformation.

7. Conclusion

Dans ce chapitre, nous proposons une dérivation des modèles OMT de niveau conceptuel en code MMS implémentable. Pour cela nous passons par une phase intermédiaire permettant à l'aide de la classe Référence de définir la répartition des supports physiques de l'application.

A partir de ce modèle organisationnel, nous établissons des conventions pour transformer les objets OMT en objets MMS en nous appuyant sur les règles définies dans le chapitre 2.

Ce travail prend en considération à la fois l'aspect statique et dynamique de la méthode OMT. Finalement, nous prenons en compte dans le processus de transformation les classes de conformité des équipements MMS cibles pour aboutir quelque soit le niveau de sophistication des supports physiques MMS à la génération d'un code MMS exploitable.

Dans le tableau 3, nous montrons d'une part l'intérêt de notre étude par rapport aux travaux antérieurs réalisés dans ce domaine et d'une autre part que nous avons atteint les objectifs initiaux définis dans le cadre de cette thèse.

Dans le chapitre suivant, nous étudions sur une application concrète l'utilisation de notre démarche pour décrire les communications entre les équipements industriels à partir d'OMT.

	E. Rondeau	J. Akazan	I. Mansour
Analyse informationnelle	avec comme base, un cadre d'intégration méthodologique. Outil de modélisation Merise (modèle de données).	non	se base sur un cadre de modélisation avec comme outil de modélisation OMT (modèle objet) + formalisation des concepts OMT et MMS
Analyse fonctionnelle	non	décrite par un langage approprié (une grammaire) proche du langage naturel et basé sur les concepts de tâches, flux de données et règles de synchronisation.	modélisée par des graphes d'états-transitions et des diagrammes de suivi d'événements.
Objets MMS générés	oui, en utilisant des règles de transformation en langage naturel.	oui, en utilisant des règles de transformation en langage naturel.	oui, en utilisant des règles de transformation formalisées.
Services MMS générés	non	oui	oui
Prise en compte des Classes de conformité	non	non	oui
Outil supportant la méthode	oui	oui, en collaboration avec l'institut Eurocim.	en voie de réalisation

Tableau 3. Récapitulatif de nos travaux par rapport aux travaux antérieurs.

Chapitre 4

Application de télé-pilotage d'un robot industriel

1. Etude de cas

Nous validons dans ce chapitre nos propositions concernant les conventions (liées aux aspects statiques et aux aspects dynamiques) par l'intermédiaire d'une application de pilotage qui servira par la suite à l'application de notre démarche complète. Cette application vise à mettre en place le pilotage d'un robot industriel avec retour vidéo (Figure 1) :

- télécommande d'un robot (téléchargement, lancement de tâches, ...),
- télécommande de la prise de vue (choix de la caméra, déplacements, zooms, ...).

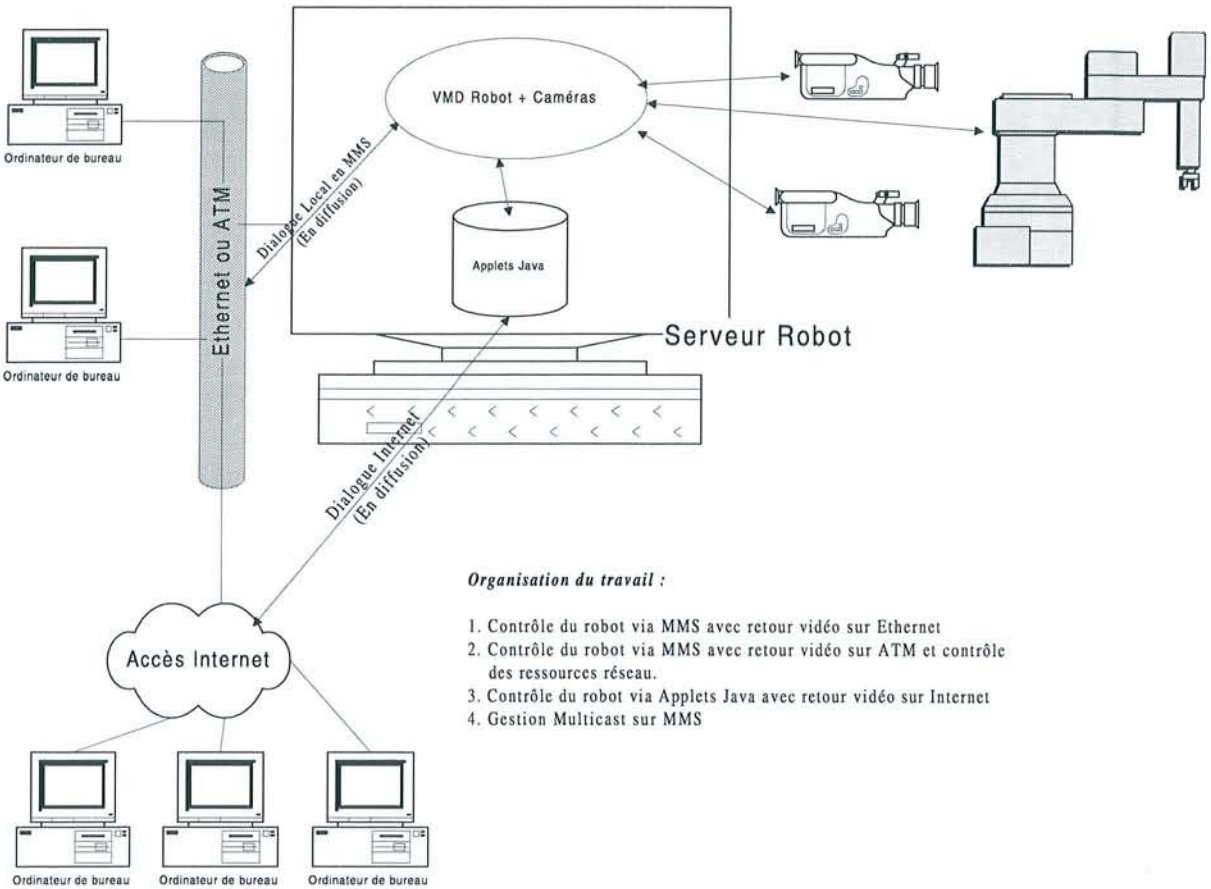

Figure 1. **L'application PLACOMAP de pilotage industriel.**

Nous modélisons l'application de pilotage appelée PLACOMAP (**PLA**teforme de **CO**mmunication **M**ultimédi**A** **P**roductique) du point de vue structurel et également du point de vue dynamique (comportemental). Le but de la maquette est de télécommander le robot et de télécommander la prise de vue au niveau local, par l'intermédiaire d'un poste de supervision connecté au réseau, avec retour vidéo via Ethernet et sans contrôle des ressources de communication. Le poste de supervision doit également pouvoir suivre le fonctionnement du robot et des caméras et les états opérationnels. Nous définissons les équipements virtuels multimédias (VMD caméras) et nous nous appuyons sur la norme d'accompagnement Robot pour représenter le robot.

Nous allons modéliser cette application selon deux points de vue : structurel et comportemental. La vue structurelle ou informationnelle sera représentée par le modèle objet, alors que la vue comportementale sera représentée par le modèle dynamique constitué des graphes d'états-transitions et des scénarios.

2. Aspect statique de l'application (modèle objet)

Nous présentons à l'aide de la figure ci-dessous (Figure 2) la structure de notre application qui est décrite à travers le modèle objet suivant.

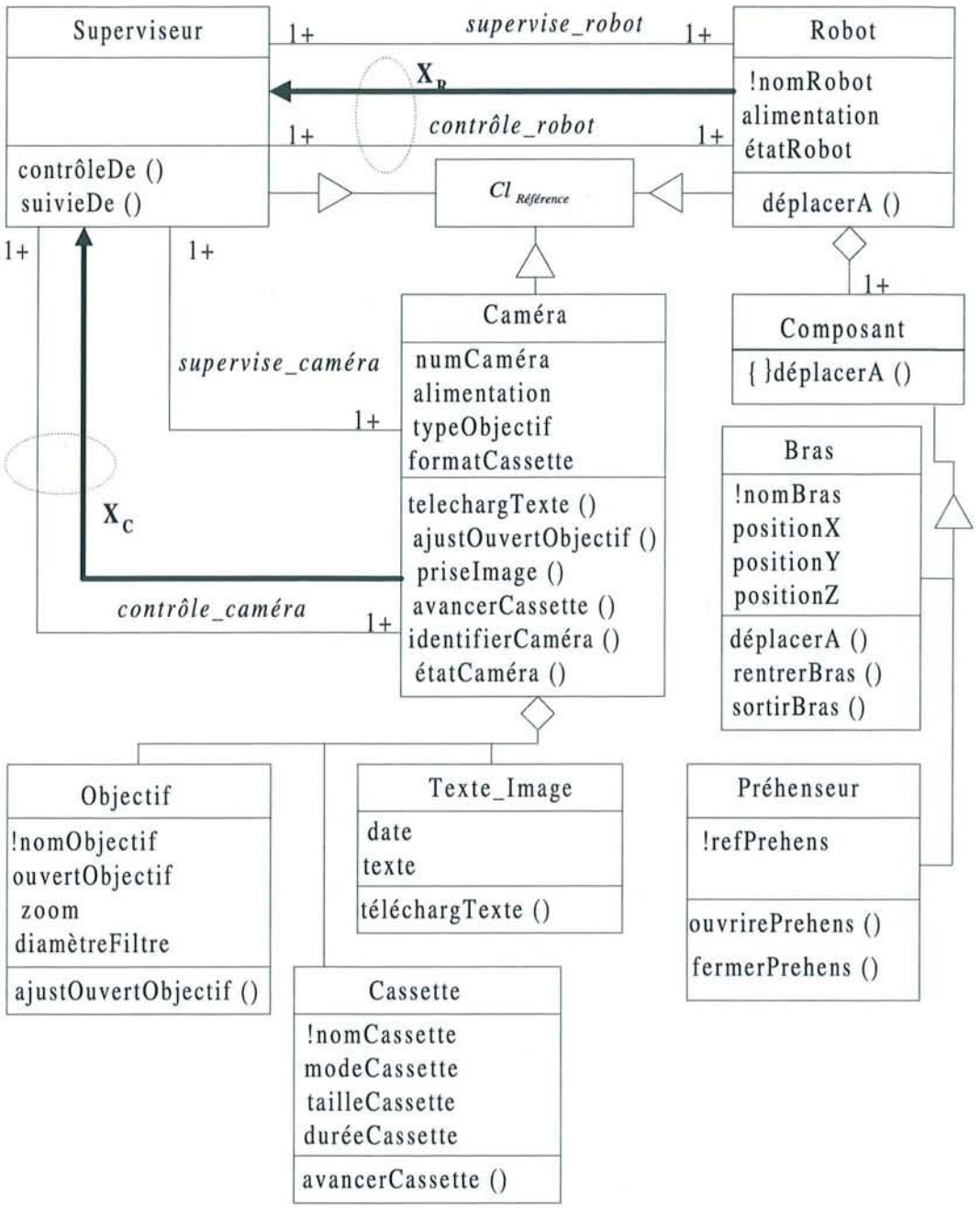

Figure 2. **Le modèle objet de l'application PLACOMAP.**

Les classes d'objets principales de ce modèle sont : Robot, Caméra, Superviseur. Afin de décrire la structure complète de chacune de ces classes, nous nous focalisons sur les caractéristiques (ou les propriétés) pertinentes et nécessaires pour la contrôler. De plus, nous déterminons les opérations qu'elles subissent.

2.1. Détails des classes constituant l'application

Nous présentons en détail les classes principales constituant notre application.

2.1.1. La classe Robot

La classe **Robot** possède des **attributs** tels que : *!nomRobot*, *étatRobot* et *alimentation*. Le premier attribut représente l'attribut clé par lequel le robot sera référencé. Le deuxième attribut représente l'état physique du robot, alors que le dernier représente l'alimentation du robot avec celle de ses composants. Une opération principale *déplacerA ()* représente une opération de déplacement dans la surface limitée par son volume de travail et elle représente une translation dans le plan *x-y* de son système de coordonnées. La classe ROBOT est composée, via la relation d'agrégation, d'un ou plusieurs composants. Un composant est un **Bras** ou un **Préhenseur** (relation de généralisation-spécialisation avec la contrainte de disjonction). Voici les détails de ces composants :

- le composant **Bras** est caractérisé par les attributs : *!nomBras*, *positionX*, *positionY*, *positionZ*. L'attribut *!nomBras* représente l'attribut clé du BRAS par lequel il sera référencé. Les autres attributs représentent les positions par rapport aux axes constituant le système de coordonnées (*X, Y, Z*). Les opérations qui lui sont appliquées sont : *déplacerA ()*, *rentrerBras ()* et *sortirBras ()*,

- le composant **Préhenseur** est caractérisé par l'attribut clé *!refPréhens* et par les opérations *ouvrirPréhens ()* et *fermerPréhens ()*.

2.1.2. La classe Caméra

La classe **Caméra** possède les attributs : *!numCaméra*, *alimentation*, *typeObjectif* et *formatCassette*. Ses opérations représentent des opérations de contrôle et de suivi qui seront mises à la disposition (ou au service) de la classe Superviseur, qui sont : *téléchargTexte ()*, *ajustOuvertObjectif ()*, *priseImage ()*, *avancerCassette ()*, *identifierCaméra ()* et *étatCaméra ()*. Grâce à l'opération *téléchargTexte ()* le Superviseur peut télécharger les informations de datation et de commentaire aux images qui seront prises. L'ajustement de l'ouverture de l'objectif sera réalisé par l'opération *ajustOuvertObjectif ()*. Le procédé de la prise d'images est représenté par l'opération *priseImage ()*. Une fois l'image prise, l'opération *avancerCassette ()* sera effectuée automatiquement et sous le contrôle de la classe Caméra.

Les deux dernières opérations représentent des opérations associées au suivi donnant lieu à une identification de la classe CAMERA et à la récupération des informations liées à son état opérationnel. La classe CAMERA est composée des sous-classes suivantes **Objectif**, **Cassette** et **Texte_Image** :

- la classe **Objectif** est un composant ayant comme attributs : *!nomObjectif*, *ouvertObjectif* (ouverture), *zoom* et *diamètreFiltre*. Comme opération, nous avons *ajustOuvertObjectif ()* qui est propagée à partir de la classe Caméra grâce à la propriété de propagation des opérations à travers d'une relation d'agrégation,

- la classe **Cassette** possède les attributs tels que : *!nomCassette*, *modeCassette*, *tailleCassette* et *duréeCassette*. *avancerCassette ()* représente l'opération qui avancera la cassette le long du processus de prise d'images,

- finalement, la classe Texte_Image est caractérisée par les attributs *date* et *texte*. L'opération *téléchargTexte ()* est propagée à la classe Caméra, via la relation d'agrégation, qui lui est appliquée.

2.1.3. La classe Superviseur

La classe Superviseur a pour objectif :

- de déplacer le Robot,

- et de contrôler, via la Caméra, le Robot.

Les classes que nous venons de citer entretiennent entre elles des relations de nature différente ; contrôle et suivi.

2.2. Les relations entretenues entre les classes d'application

Les trois classes d'objets principales : Superviseur, Robot et Caméra sont associées par des relations binaires avec une multiplicité multiple (de un ou plusieurs à un ou plusieurs). Nous détaillons maintenant ces relations.

2.2.1. Les relations entre les classes Superviseur et Robot

Les classes Superviseur et Robot sont associées via deux relations : ***contrôle_robot*** et ***supervise_robot***.

- la relation ***contrôle_robot*** est une relation un ou plusieurs à un ou plusieurs (multiple). Une instance de la classe Robot peut être contrôlée, durant sa vie, par plusieurs instances de la classe Superviseur mais à un temps *t à* une seule instance de la classe Superviseur. Ceci se traduit par l'ajout de la contrainte d'exclusivité temporelle représentée par la flèche et le symbole X_R. Cette contrainte se transformera en un objet MMS sémaphore (sous réserve de la classe de conformité),

- la relation ***supervise_robot*** est également une relation multiple de un ou plusieurs à un ou plusieurs. Cette relation n'est pas concernée par la contrainte d'exclusivité temporelle. Ainsi, le Robot pourrait être suivi par plusieurs Superviseur au même moment.

2.2.2. Les relations entre les classes Superviseur et Caméra

Les classes **Superviseur** et **Caméra** possèdent deux relations : ***contrôle_caméra*** et ***supervise_caméra***. Ces deux relations sont multiples (de un ou plusieurs à un ou plusieurs).

- la relation ***contrôle_caméra*** est contrainte par la contrainte d'exclusivité temporelle représentée, graphiquement, par la flèche et le symbole X_C de telle sorte que la Caméra soit contrôlée par un seul Superviseur à un instant donné,

- la relation ***supervise_caméra*** n'a pas de contraintes, de telle sorte que la Caméra peut être suivie par plusieurs Superviseurs à un instant donné.

Dans la suite (Section 4) nous appliquons notre démarche et nos propositions présentées lors des chapitres 2 et 3 pour aboutir finalement à sa validation à travers l'application PLACOMAP.

3. Aspect comportemental de l'application (modèle dynamique)

Cet aspect est représenté par deux outils : graphes d'états-transitions, et diagrammes de suivi d'événements (event trace).

3.1. Graphes d'états des classes : Robot et Caméra

Dans le chapitre précédent (Section 5), nous avons présenté quelques fonctionnalités fournies par les équipements manufacturiers dits Serveurs telles que : gestion de ressource, gestion de tâches avec leurs graphes d'états-transitions lui sont associés. Ces derniers feront partie du graphe d'états de leurs équipements respectifs. Notons que de tels graphes représentent le comportement vis-à-vis du système de communication. Le lecteur peut se reporter au chapitre 3 pour plus de détails.

3.2. Scénarios d'utilisation de l'application

Nous allons présenter tout d'abord un scénario classique (ou standard) entre un objet de la classe Superviseur et un objet de la classe Caméra en utilisant le diagramme de suivi d'événement associé. Puis, nous présentons un scénario plus complet entre un objet de la classe Superviseur, un objet de la classe Caméra et un objet de la classe Robot avec le diagramme de suivi d'événements associé.

3.2.1. Scénario 1

Pour l'aspect supervision, nous définissons une partie du scénario (les erreurs et les exceptions ne sont pas prise en considérations) en indiquant les messages échangés entre les objets des classes Superviseur et Caméra, voir figure 3.

- le superviseur lance une requête de prise de ressource caméra,
- au cas où la caméra est libre, le processus de prise de ressource est mis en route et un acquittement est renvoyé au superviseur. Par ce moyen, le superviseur est le seul autorisé à contrôler la caméra,
- le superviseur demande le lancement de la procédure d'acquisition vidéo et attend l'acquittement,
- suite au lancement de la procédure d'acquisition vidéo, un acquittement positif est envoyé par la caméra,
- à la réception de cet acquittement positif, le superviseur met en route l'affichage vidéo qui représente une opération locale du superviseur qui héberge la réception vidéo avec les ressources nécessaires,
- le superviseur lance une demande de libération de la ressource caméra et en attend une réponse à cette demande,
- la caméra envoie une réponse indiquant sa libération.

Figure 3. **Scénario d'utilisation de la Caméra par le Superviseur.**

Le diagramme de suivi d'événements associé à ce scénario (Figure 4) montre les objets Caméra_1 et Poste_Supervion en interaction avec les événements échangés par ces objets.

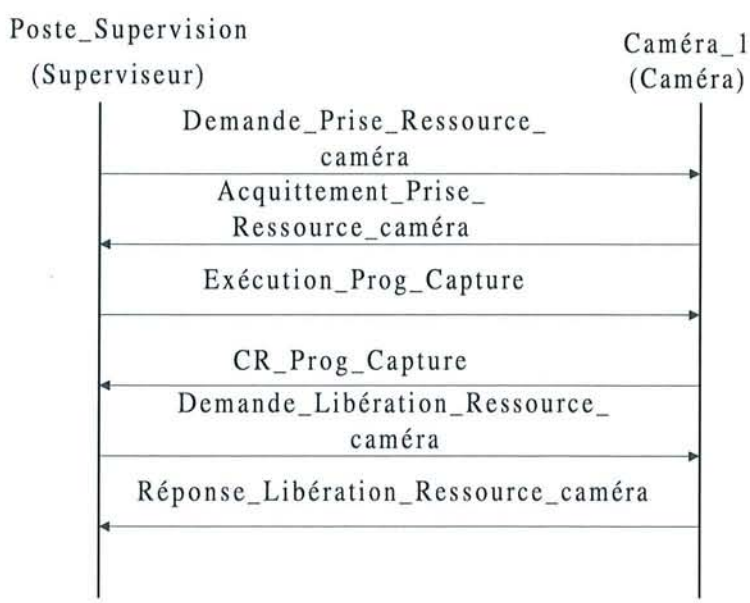

Figure 4. **Diagramme de suivi d'événement concernant Scénario 1.**

3.2.2. Scénario 2

Ce scénario est plus complet que le scénario précédent puisqu'il utilise les deux ressources de la maquette. La figure 5 montre le scénario d'utilisation approprié.

- le superviseur lance une requête de prise de ressource caméra,
- au cas où la caméra est libre, le processus de prise de ressource est mis en route et un acquittement est renvoyé au superviseur. Par ce moyen, le superviseur est le seul autorisé à contrôler la caméra,
- le superviseur demande le lancement de la procédure d'acquisition vidéo et attend l'acquittement,
- suite au lancement de la procédure d'acquisition de vidéo, un acquittement positif est envoyé par la caméra,
- à la réception de cet acquittement positif, si c'est la supervision visuelle qui est demandée nous retombons dans le scénario précédent. Si la commande est le contrôle du Robot, le superviseur demande la prise de ressource Robot et attend une réponse,
- à la réception de l'acquittement de la prise de ressource Robot, le superviseur lance le programme tâche du robot et attend l'acquittement du lancement,
- une fois l'acquittement reçu, le superviseur envoie une demande de libération de la ressource robot et attend la réponse à sa demande,
- à la réception de la réponse à la demande de libération du robot, le superviseur envoie la demande de libération de la ressource caméra et attend la réponse à cette demande,
- la caméra renvoie une réponse au superviseur indiquant sa libération et ainsi la possibilité qu'elle soit contrôlée par un autre superviseur.

Figure 5. **Scénario d'utilisation de la Caméra et du Robot par le superviseur.**

Le diagramme de suivi d'événements (Figure 6) associé à ce scénario montre les objets GMFanuc, Caméra_1 et Poste_Supervision en interaction avec les événements échangés par ces trois objets.

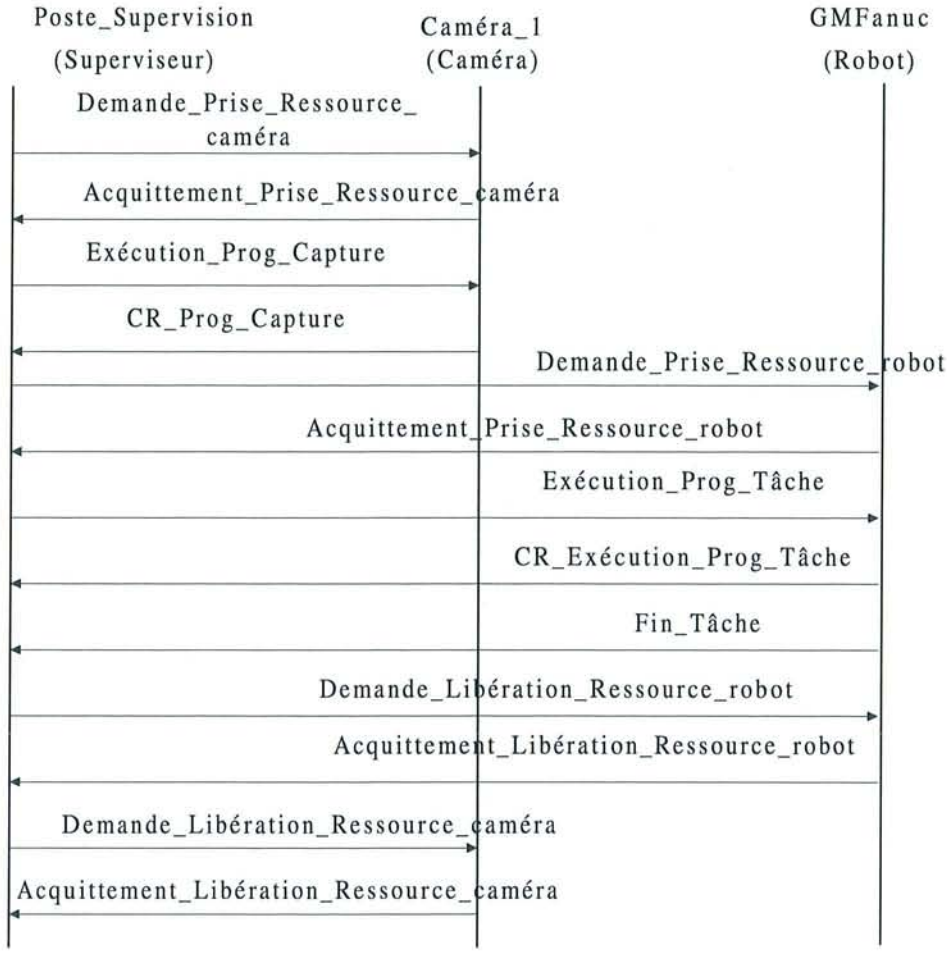

Figure 6. **Suivi d'événements concernant le Scénario 2.**

3.3. Diagrammes d'états de la classe Superviseur

Nous présentons une partie du graphe d'états de la classe Superviseur le scénario 1. La figure 7 montre le modèle de comportement du Scénario 1.

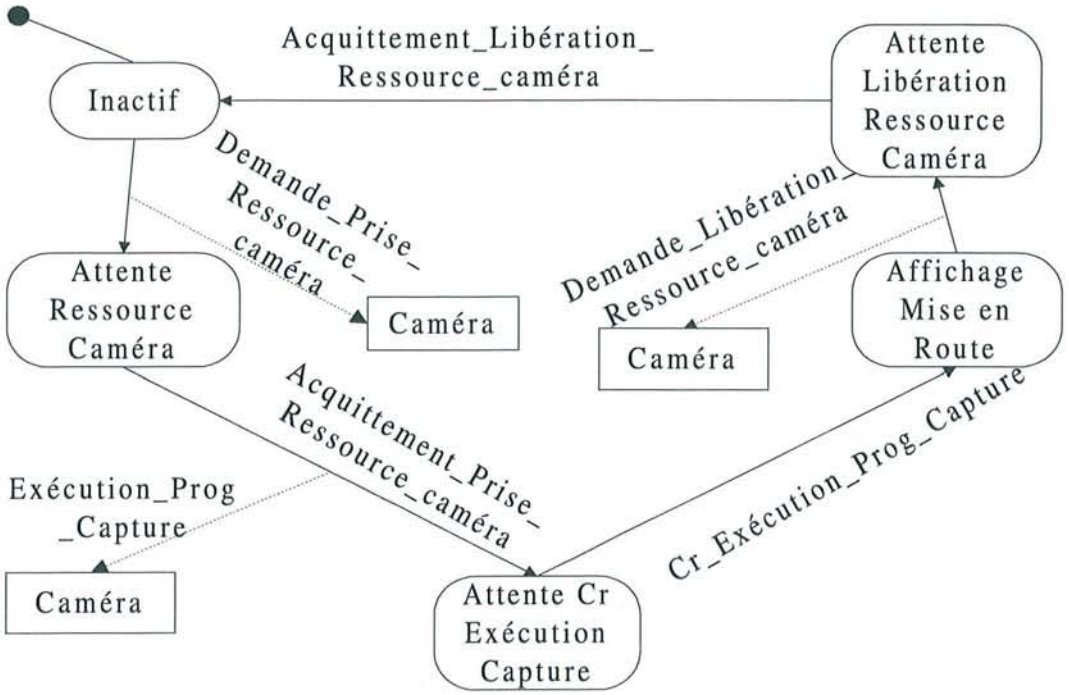

Figure 7. **Diagramme d'états de la classe Superviseur correspondant au Scénario 1.**

La figure 8 montre le modèle de comportement correspondant au scénario 2.

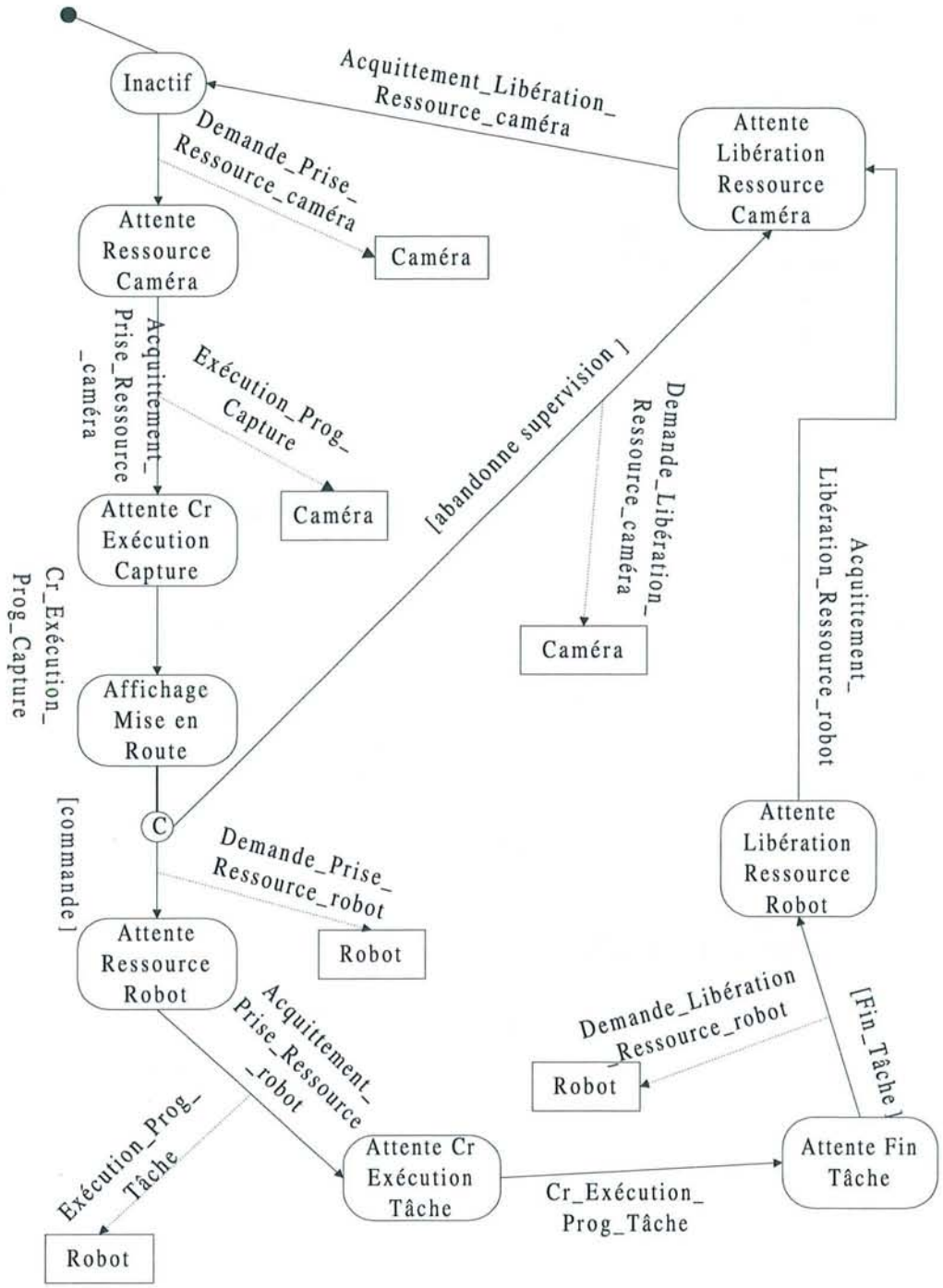

Figure 8. **Diagramme d'états de la classe Superviseur correspondant au Scénario 2.**

4. Application de nos propositions

Nous allons suivre les lignes directrices de notre approche afin de mettre en œuvre nos propositions. Voici les étapes de notre démarche.

4.1. La répartition de l'application

Afin de répartir l'application, le concepteur est sollicité pour spécialiser les classes réparties par rapport à la classe abstraite 'classe de référence ; $Cl_{Référence}$' via la relation de généralisation-spécialisation. Toute sous-classe de la classe $Cl_{Référence}$ se trouvera sur un site et supportera un VMD portant le nom de cette sous-classe. Dans notre cas, nous avons trois VMDs : **VMD_Superviseur**, **VMD_Caméra** et **VMD_Robot** (voir figure 2).

Chacune de ces classes principales réside sur un site différent avec bien sûr ses propres composants. Cela veut dire que le Robot avec ses composants sera sur un site, la Caméra avec ses composants sera sur un autre site. Donc, nous avons trois sites dans notre application.

4.2. Hiérarchisation et application de l'algorithme de suppression de liens

Nous hiérarchisons le modèle et nous déterminons les niveaux. Nous reprenons les suppositions présentées dans le chapitre 3 et ainsi la classe $Cl_{Référence}$ est de niveau 0 alors que les classes Superviseur, Caméra et Robot sont de niveau 1. Les classes Bras, Préhenseur, Objectif, Texte_Image et Cassette sont de niveau 2. En appliquant l'algorithme de suppression de liens entre les classes de même niveau, les relations *contrôle_robot*, *supervise_robot, contrôle_caméra* et *supervise_caméra* sont supprimées. Dans la suite nous allons déterminer les structures des VMDs (les objets MMS correspondant aux objets d'application).

4.3. Configuration des VMDs

Comme nous venons de le voir, les classes d'objets de niveau 1 sont supposées supporter des VMDs. Ainsi, nous avons trois VMDs : **VMD_Superviseur**, **VMD_Caméra** et **VMD_Robot**. Afin de déterminer la structure de chaque VMD, nous avons recours à nos règles de transformation présentées dans le chapitre 2 et nous prenons en considération les conventions faites lors du chapitre 3. Voici, en détail la structure de chaque VMD.

4.3.1. Configuration du VMD_Robot

Configurer un VMD c'est déterminer sa structure interne en terme d'objets MMS qui le composent. Par ce fait, le **VMD_Robot** est déterminé après avoir appliqué nos règles de transformation en prenant en compte les conventions proposées dans le chapitre précédent et la classe de conformité (l'ensemble d'objets et services MMS réellement implémentés sur le Robot). Sachant que la classe de conformité de notre robot est *CC_1* (classe de conformité de base ; supporte seulement les objets MMS variables et leurs services associés).

$$\forall Cl_{Référence},\ Robot_{omt} \in Cl_{diag},\ \eta_{Référence,ROBOT},$$

$$\| Trans(OMT, Robot_{omt}) = VMD_{mms} \qquad (application\ de\ la\ règle\ R_{class})$$
$$/\ VMD_{mms}\_Nme = Robot.$$

Il est à noter que tous les attributs déjà définis pour le modèle général d'un VMD sont supportés par le **VMD_Robot** avec en plus les attributs propres à la classe Robot.

Tout d'abord, nous désignons les **objets variables nommées MMS** générés à l'aide du processus de transformation appliqué aux attributs de la classe **Robot**. Les attributs de la classe Robot sont transformés en objets de la classe variable nommée MMS (en **objets variables nommées**) tout en préservant le même nom et le même type. Voici la liste des objets variables nommées :

$$\forall At_{omt} \in \alpha(Robot_{omt}) = \{nomRobot, alimentation, étatRobot\}\ /\ i = 1...3 \quad alors$$
$$\| Trans(OMT, At_{omt\,i}) = variableNommée_{mms} \quad avec$$
$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ,$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = nomRobot,$$
$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = string,$$

$$b - \| Trans(OMT, At_{omt_2}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = alimentation,$$
$$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = booléen,$$

$$c - \| Trans(OMT, At_{omt_3}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_3}\_Nme = étatRobot,$$
$$variableNommée_{mms}\_Typ = At_{omt_3}\_Typ$$
$$= enuméré.$$

Puisque la classe de conformité du Robot est *CC_1* nous ne pouvons pas appliquer la convention concernant l'objet MMS domaine en se basant sur la relation d'agrégation entre la classe **Bras** et la classe **Robot**. Ainsi la classe **Bras** est transformée en une classe variable nommée structurée MMS.

$$\forall Cl_{Référence}, Robot_{omt}, Bras_{omt} \in Cl_{diag}\ /\ \eta_{Cl_{Référence},Robot_{omt}} \quad et \quad \delta_{Robot_{omt},Bras_{omt}}\ alors :$$
$$\| Trans(OMT, Bras_{omt}) = VariableStructurée_{mms} \qquad (application\ de\ la\ règle\ R_{class})$$
$$avec \quad VariableStructurée_{mms} = Bras.$$

Le contenu de cette variable structurée est constitué des attributs de la classe **Bras** qui représentent à leur tour des variables nommées MMS portant les mêmes noms et les mêmes types.

$\forall At_{omt_i} \in \alpha(Bras_{omt}) = \{nomBras, positionX, positionY, positionZ\} \ / \ i = 1...4 \quad alors$

$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \quad avec$

$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$

$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ .$

$\Leftrightarrow$

$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$

$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = nomBras,$

$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = string,$

$b - \| Trans(OMT, At_{omt_2}) = variableNommée_{mms}$

$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = positionX,$

$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = pos\,(type\ défini\ par\ l'utilisateur),$

$c - \| Trans(OMT, At_{omt_3}) = variableNommée_{mms}$

$variableNommée_{mms}\_Nme = At_{omt_3}\_Nme = positionY,$

$variableNommée_{mms}\_Typ = At_{omt_3}\_Typ = pos\,(type\ défini\ par\ l'utilisateur),$

$d - \| Trans(OMT, At_{omt_4}) = variableNommée_{mms}$

$variableNommée_{mms}\_Nme = At_{omt_4}\_Nme = positionZ,$

$variableNommée_{mms}\_Typ = At_{omt_4}\_Typ = pos\,(type\ défini\ par\ l'utilisateur),$

Etant donné que la classe **Préhenseur** est un composant de la classe ROBOT, elle sera également supportée par un ou plusieurs objets de la classe variable nommée structurée MMS.

$\forall Cl_{Référence}, Robot_{omt}, Préhenseur_{omt} \in Cl_{diag} \ / \ \eta_{Cl_{Référence}, Robot_{omt}} : Cl_{Référence} \rightarrow Robot_{omt}$

et

$\delta_{Robot_{omt}, Préhenseur_{omt}} : Robot_{omt} \rightarrow Préhenseur_{omt} \quad alors :$

$\| Trans(OMT, Préhenseur_{omt}) = VariableStructurée_{mms} \quad (application\ de\ la\ règle\ R_{class})$

$avec \quad VariableStructurée_{mms} = Préhenseur .$

Quant aux attributs de la classe Préhenseur, ils appartiennent à un ou plusieurs objets de la classe variable structurée MMS. Ces attributs représentent des **objets variables nommées MMS** portant les mêmes noms et les mêmes types.

$$\forall At_{omt_i} \in \alpha(Pr\acute{e}henseur_{omt}) = \{refPrehens\} \quad / \quad i = 1 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \quad avec$$
$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ.$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = ref\ Prehens,$$
$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = entier,$$

La **contrainte d'exclusivité temporelle** X_R définie sur la relation *contrôle_Robot* est transformée en un objet MMS de la classe **sémaphore étiqueté** (Pool Semaphore) avec un nombre de jetons égal à un. Ceci limitera le contrôle à un seul superviseur.

$$à\ un\ t\ donné, \quad \forall Superviseur_{omt}, Robot_{omt} \in Cl_{diag} \quad / \quad v_{'X'Superviseur,Robot_t}$$
$$\Rightarrow$$
$$\| Trans(OMT, v_{'X'Superviseur,Robot_t}) = Sémaphore_{Pool_{mms}}, \quad avec$$
$$Sémaphore_{Pool_{mms}}\_Nme = X_R.$$

Nous notons que jusqu'ici nous avons défini la structure du VMD-Robot. Le comportement dynamique de ce VMD est décrit par les graphes d'états-transitions.

4.3.2. Configuration du VMD_Caméra

La structure interne du **VMD_Caméra** est déterminée en appliquant les règles de transformation que nous avions définies précédemment et par la prise en compte des conventions que nous avions proposées. Ceci en prenant également en considération la classe de conformité qui est dans ce cas *CC_3* (cas tous les objets et services MMS sont disponibles). Nous transformons d'abord la classe Caméra en un VMD portant le nom VMD_Caméra par l'application de la règle ci-dessous :

$$\forall Cl_{Référence}, \ Caméra_{omt} \in Cl_{diag}, \ \eta_{Référence,Caméra},$$

$$\| Trans(OMT, Caméra_{omt}) = VMD_{mms} \qquad (application\ de\ la\ règle\ R_{class})$$
$$/ \ VMD_{mms}\_Nme = Caméra.$$

Les attributs caractérisant le modèle général du VMD défini par la norme sont hérités par le VMD_Caméra. Les attributs de la classe Caméra sont rajoutés à ceux déjà hérités.

Les attributs (hérités et spécifiques) seront transformés par l'application des règles de transformation en des **objets** MMS de la **classe variable** nommée portant les mêmes noms et les mêmes types. La liste des objets variables nommées est constituée de :

$$\forall At_{omt_i} \in \alpha(Caméra_{omt}) = \{numCaméra, alimentation, typeObjectif, formatCassette\} \ / \ i = 1...4 \quad alors$$

$$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \quad avec$$

$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$

$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ,$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = numCaméra,$$

$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = entier,$$

$$b - \| Trans(OMT, At_{omt_2}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = alimentation,$$

$$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = booléen,$$

$$c - \| Trans(OMT, At_{omt_3}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_3}\_Nme = typeObjectif,$$

$$variableNommée_{mms}\_Typ = At_{omt_3}\_Typ = entier.$$

$$d - \| Trans(OMT, At_{omt_4}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_4}\_Nme = formatCassette,$$

$$variableNommée_{mms}\_Typ = At_{omt_4}\_Typ = entier.$$

Tout composant de la classe Caméra (une classe reliée avec la classe Caméra par une relation d'agrégation) est transformée en un objet MMS de la classe domaine. Ainsi nous avons trois domaines : **Domaine_Objectif**, **Domaine_Texte_Image** et **Domaine_Cassette**.

Par conséquent, la classe **Objectif** est supportée par un objet domaine appelé **Domaine_Objectif** et obtenue comme suit :

$$\forall Cl_{Référence}, Caméra_{omt}, Objectif_{omt} \in Cl_{diag} \ / \ \eta_{Cl_{Référence}, Caméra_{omt}} \quad et \quad \delta_{Caméra_{omt}, Objectif_{omt}} \Rightarrow$$

$$\| Trans(OMT, Objectif_{omt}) = domaine_{mms} \quad (application\ de\ la\ règle\ R_{class})$$

$$avec \quad domaine_{mms}\_Nme = Objectif\ .$$

Les **attributs** de la classe **Objectif** seront transformés en des objets de la classe MMS **variable nommée** en préservant les noms et les types.

$$\forall At_{omt_i} \in \alpha(Objectif_{omt}) = \{nomObjectif, ouvertObjectif, zoom, diamètreFiltre\} \ / \ i = 1...4 \quad alors$$

$$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \quad avec$$

$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$

$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ .$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = nomObjectif ,$$

$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = string ,$$

$$b - \| Trans(OMT, At_{omt_2}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = ouvertObjectif ,$$

$$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = entier,$$

$$c - \| Trans(OMT, At_{omt_3}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_3}\_Nme = zoom,$$

$$variableNommée_{mms}\_Typ = At_{omt_3}\_Typ = entier,$$

$$d - \| Trans(OMT, At_{omt_4}) = variableNommée_{mms}$$

$$variableNommée_{mms}\_Nme = At_{omt_4}\_Nme = diamètreFiltre,$$

$$variableNommée_{mms}\_Typ = At_{omt_4}\_Typ = entier.$$

Etant donné que la classe **Texte_Image** est un composant de la classe Caméra, elle sera également supportée par un objet de la classe domaine (**Domaine_Texte_Image**).

$$\forall Cl_{Référence}, Caméra_{omt}, Texte\_Image_{omt} \in Cl_{diag} \ / \ \eta_{Cl_{Référence}, Caméra_{omt}} \quad et \quad \delta_{Caméra_{omt}, Texte\_Image_{omt}} \quad alors :$$

$$\| Trans(OMT, Texte\_Image_{omt}) = domaine_{mms} \quad (application\ de\ la\ règle\ R_{class})$$

$$avec \quad domaine_{mms}\_Nme = Texte\_Image .$$

Les **attributs** de la classe **Texte_Image** appartiennent à un objet de la classe domaine de ce VMD (VMD_Caméra).

$$\forall At_{omt_i} \in \alpha(Texte\_Image_{omt}) = \{date, texte\} \quad / \quad i = 2 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \qquad avec$$
$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ .$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = date,$$
$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = date,$$

$$b - \| Trans(OMT, At_{omt_2}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = texte,$$
$$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = string,$$

La classe **Cassette** est supportée par un objet domaine appelé **Domaine_Cassette** et obtenue comme suit :

$$\forall Cl_{Référence}, Caméra_{omt}, Cassette_{omt} \in Cl_{diag} \quad /$$
$$\eta_{Cl_{Référence}, Caméra_{omt}} : Cl_{Référence} \rightarrow Caméra_{omt} \quad et$$
$$\delta_{Caméra_{omt}, Cassette_{omt}} : Caméra_{omt} \rightarrow Cassette_{omt} \quad alors :$$
$$\| Trans(OMT, Cassette_{omt}) = domaine_{mms} \qquad (application\ de\ la\ règle\ R_{class})$$
$$avec \quad domaine_{mms}\_Nme = Cassette .$$

Les **attributs** de la classe **Cassette** sont transformés en des objets de la classe MMS **variable nommée** en préservant les noms et les types.

$$\forall At_{omt_i} \in \alpha(Cassette_{omt}) = \{nomCassette, modeCassette, tailleCassette, duréeCassette\}$$
$$/ \quad i = 1...4 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = variableNommée_{mms} \qquad avec$$
$$variableNommée_{mms}\_Nme = At_{omt_i}\_Nme$$
$$variableNommée_{mms}\_Typ = At_{omt_i}\_Typ .$$

$$\Leftrightarrow$$

$$a - \| Trans(OMT, At_{omt_1}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_1}\_Nme = nomCassette,$$
$$variableNommée_{mms}\_Typ = At_{omt_1}\_Typ = string,$$

$$b - \|Trans(OMT, At_{omt_2}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_2}\_Nme = modeCassette,$$
$$variableNommée_{mms}\_Typ = At_{omt_2}\_Typ = énuméré,$$

$$c - \|Trans(OMT, At_{omt_3}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_3}\_Nme = tailleCassette,$$
$$variableNommée_{mms}\_Typ = At_{omt_3}\_Typ = entier,$$

$$d - \|Trans(OMT, At_{omt_4}) = variableNommée_{mms}$$
$$variableNommée_{mms}\_Nme = At_{omt_4}\_Nme = duréeCassette,$$
$$variableNommée_{mms}\_Typ = At_{omt_4}\_Typ = entier.$$

La **contrainte d'exclusivité temporelle** X_C associée à la relation *contrôle_Caméra* sera transformée en un objet MMS de la classe **sémaphore étiqueté** (Pool Semaphore) avec un nombre de jetons égal à un. Ceci limitera également le contrôle à un unique superviseur à un instant donné.

$$à\ un\ t\ donné,\ \forall Superviseur_{omt}, Caméra_{omt} \in Cl_{diag}\ /\ v_{'X'Superviseur,Caméra_t}$$
$$\Rightarrow$$
$$\|Trans(OMT, v_{'X'Superviseur,Caméra_t}) = Sémaphore_{Pool_{mms}},\ avec$$
$$Sémaphore_{Pool_{mms}}\_Nme = X_C.$$

Nous avons défini la structure des VMDs : VMD-Robot, VMD_Caméra. (sachant que une telle définition était en accord avec les classes de conformité des classes constituant les VMDs). Leur comportement dynamique est décrit par les graphes d'états-transitions appropriés (chapitre 3).

4.4. Comportement dynamique du Superviseur (en terme de services MMS)

Après une projection du modèle objet de l'application PLACOMAP sur le modèle objet MMS, nous effectuons une démarche similaire pour le modèle dynamique en le projetant sur les services MMS. Notons que les services MMS sont en total accord avec les objets qui les manipulent. Nous détaillons ci-après les suivis d'événements avec les graphes d'états associés exprimés via les services MMS appropriés.

4.4.1. Scénario 1 et graphe d'états associé

Par l'intermédiaire de la figure 9, nous montrons le suivi d'événements représentant le scénario 1 avec les échanges effectués entre les objets concernés (Poste_Supervision et Caméra_1) exprimés en services MMS. Les services de gestion de sémaphore offrent au client, qui est toujours le superviseur, le moyen d'avoir le droit de contrôler la Caméra.

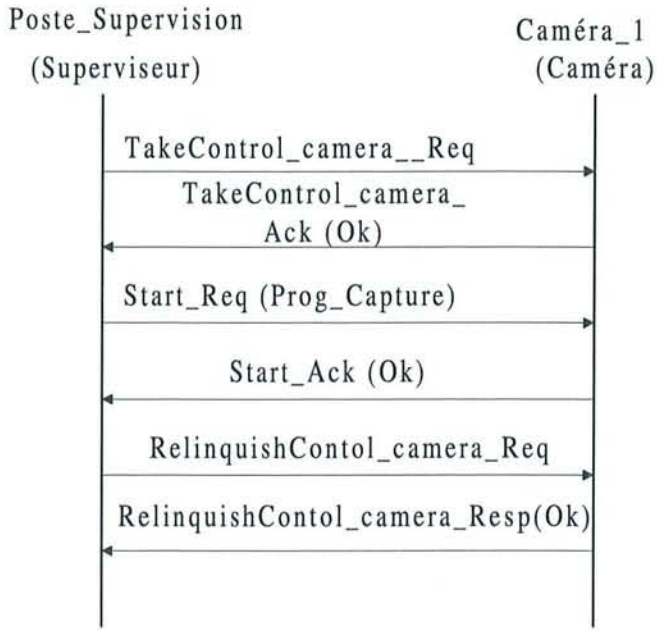

Figure 9. **Suivi d'événements représentant le Scénario 1, effectué par les services MMS.**

Les services *Take_Control* et *Relinquish_Control* permettent au Superviseur de réserver la ressource Caméra et de la libérer quand il a fini le contrôle. Les tâches associées à la Caméra seront manipulées par les services de gestion de programmes invocation MMS. Ainsi, le lancement d'une tâche sera effectué par le service Start. Nous voyons que pour chaque service nous avons deux messages (ou événements) un concerne la demande et l'autre l'acquittement. Figure 10 montre la partie du graphe d'états du Superviseur associé au scénario présenté dans la figure 9.

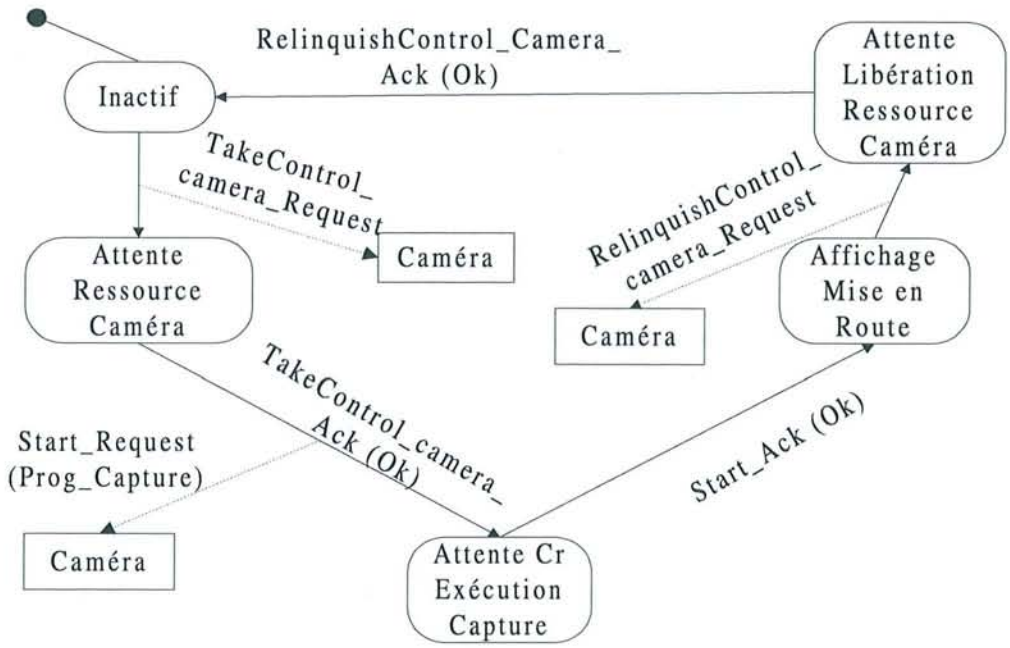

Figure 10. **Diagramme d'états associé au scénario 1.**

4.4.2. Scénario 2 et graphe d'états associé

La figure 11 montre le suivi d'événements représentant le scénario 2 avec les échanges effectués entre les objets concernés (Poste_Supervision, Caméra_1 et GMFanuc). Les échanges sont exprimés par les services MMS. Les services de gestion de sémaphore offrent au superviseur le moyen d'avoir le droit de contrôler la Caméra et le Robot. Nous avons déjà mentionné que la classe de conformité de la Caméra est *CC_3*. Concernant la classe Robot, elle appartient à la classe de conformité de base *CC_1* et dispose ainsi des services MMS de gestion de variables (Read, Write, InformationReport).

Par le biais du service de modification (d'écriture) de la variable, le Superviseur peut effectuer une demande de la ressource Robot. Le Robot possède une variable booléenne *ressource* par laquelle le Superviseur peut acquérir la ressource et la libérer. Pour acquérir la ressource, le Superviseur écrit la valeur 1 (équivalente à vrai) dans la variable *ressource*. Pour libérer la ressource Robot, le Superviseur écrit la valeur 0 (équivalente à faux) dans la variable *ressource*.

La manipulation des programmes (ou tâches) Robot sera effectuée via le service d'écriture d'une variable VarProg de type «entier» qui prend des valeurs de 1 à 4 pour lancer l'exécution, suspendre, reprendre et réinitialiser de tels programmes. Afin de lancer l'exécution d'un programme donné du Robot, le Superviseur écrit la valeur 1 dans la variable *VarProg* associée au programme.

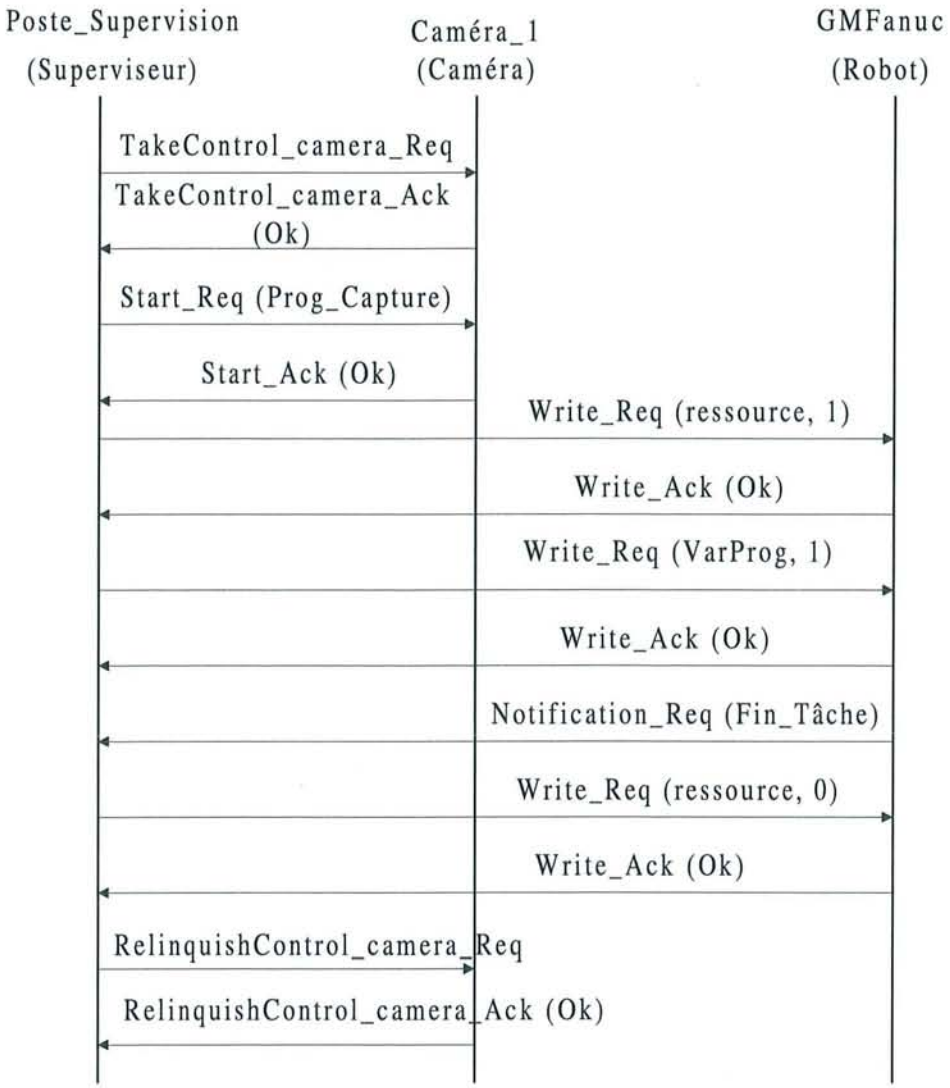

Figure 11. Suivi d'événements associé au Scénario 2.

La figure 12 montre une partie plus complète que celle décrite dans la figure 11 du graphe d 'états du Superviseur et qui est associée au scénario 2.

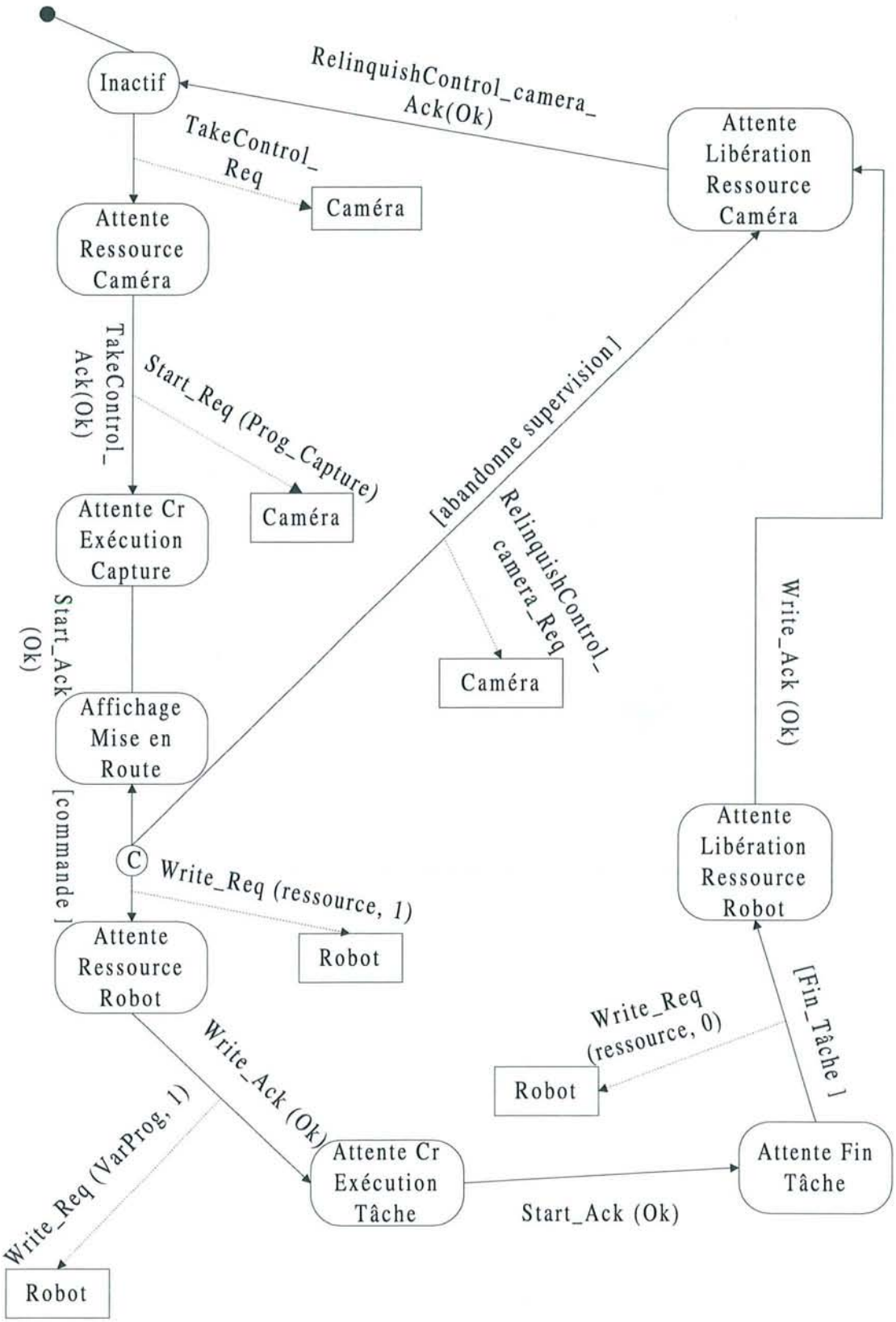

Figure 12. **Diagramme d'états représentant le suivi d'événements précédent.**

5. Configuration de l'application

La configuration de la structure de communication de l'application est montrée sur la figure 13. L'environnement se compose de trois PCs connectés sur un réseau Ethernet.

Le poste numéro 1 est un poste d'acquisition et traitement d'images équipé d'une carte d'acquisition vidéo, appelée VM422 construite par la société VETEC MULTIMEDIA, fonctionnant sous système d'exploitation Windows 95. Ce poste est connecté sur le réseau Ethernet.

Le poste numéro 2 est un poste simulant le Robot et qui est relié par une liaison série RS232 à l'armoire de commande du robot construit par la société GMFanucRobotics série A-600 (robot d'assemblage). Ce poste est lui aussi connecté sur le réseau Ethernet.

Le poste numéro 3 sert de poste de contrôle à partir duquel le superviseur peut piloter, via une interface uniforme, d'une part les équipements multimédias (les contrôleurs d'acquisition) et d'autre part le robot. Il sert aussi de poste de présentation sur lequel l'opérateur peut surveiller les sites distants. Le poste est équipé d'une carte réseau Ethernet et fonctionne sous Windows 95.

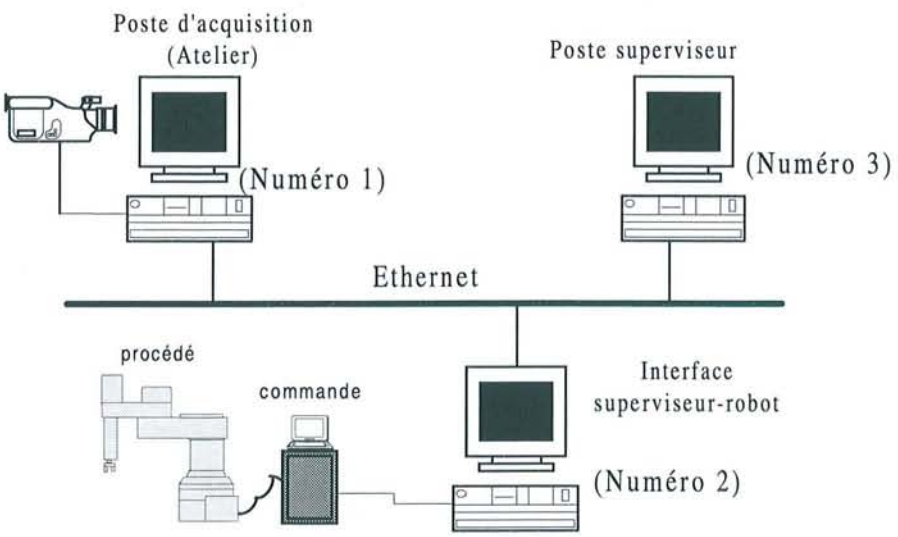

Figure 13. **La configuration de l'application de télépilotage.**

6. Mise en œuvre de l'application

Comme le montre la figure 13, l'environnement cible de l'application se caractérise par l'hétérogénéité des équipements à contrôler (multi-constructeur).

La figure 14 montre l'architecture technique (logicielle ou informatique) de notre application.

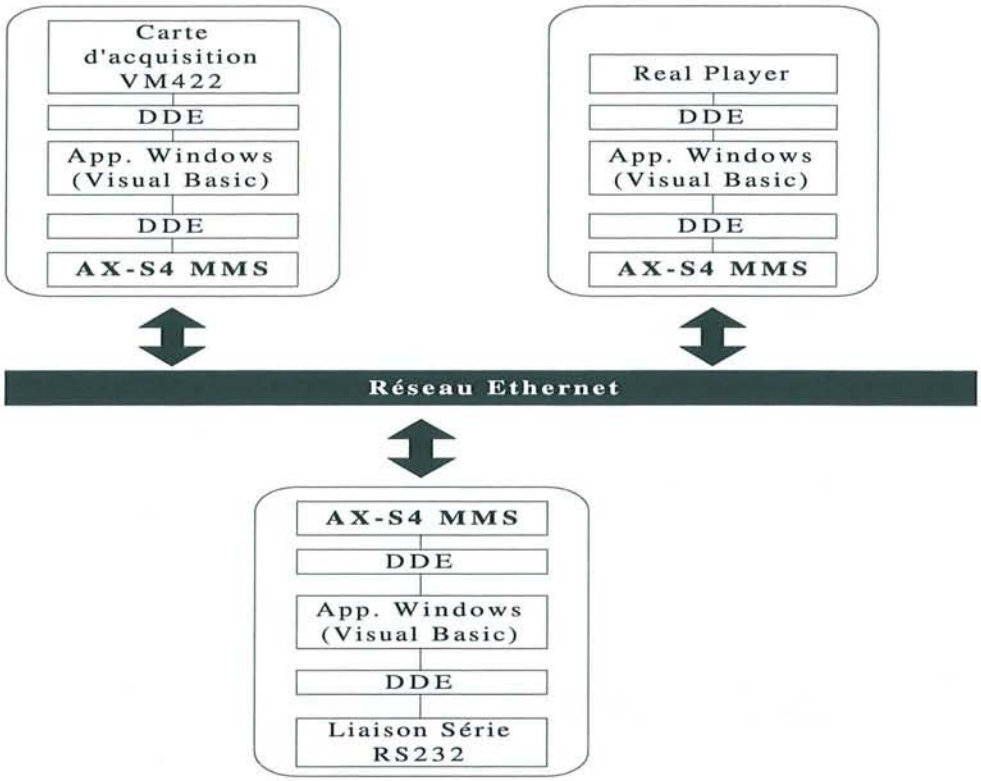

Figure 14. **Architecture logicielle de l'application.**

AX-S4 MMS est un produit de la société SISCO (Systems Integration Specialists Company , Inc.). AX-S4 MMS pour Windows est un serveur DDE (Dynamic Data Exchang) qui agit comme interface entre des applications clientes DDE et des équipements supportant le protocole de communication MMS. Avec AX-S4 MMS, tous logiciels Microsoft fonctionnant sous Windows et qui supportent DDE (Excel, Visual Basic, Visual C++, ...) peut utiliser le protocole MMS pour accéder aux données des équipements et applications interconnectés sur le réseau. Il permet à des applications développées sous Windows et à des équipements industriels (Automate programmable, Machine à Commande Numérique, Robot, ...) de communiquer sur un réseau Ethernet en utilisant MMS.

En utilisant NetDDE, les applications Windows sur les autres PCs peuvent communiquer avec les équipements et applications MMS via AX-S4 MMS.

Via l'interface du poste de supervision, l'opérateur établit une connexion avec le poste d'acquisition. Une fois la connexion établie, l'opérateur peut exercer une supervision dite passive grâce aux services d'identification de l'équipement connecté, et d'état (Identify et Status). L'opérateur peut également piloter le poste d'acquisition (démarrer, arrêter) à travers des services de modification de variables MMS (une variable est associée à une tâche ou programme) et enclenche, en local, l'application Real Player qui va visionner le Robot sur le poste. RealPlayer 5.0 de RealNetworks est un système unique de diffusion audio et vidéo en temps réel sur Internet.

Côté poste d'acquisition, une lecture locale via une liaison DDE, des valeurs des variables associées aux procédés d'acquisition. Une valeur un de la variable enclenche le procédé qui lui est associé. Ceci active l'application RealEncoder 5.1 qui permet de convertir les fichiers audio et vidéo en format RealVideo de RealNetworks. Les images capturées par la

carte d'acquisition seront sauvegardées dans fichier se trouvant sur le serveur NT supportant une carte d'acquisition de vidéo appelée METEOR et construite par la société MATROX. C'est à partir de ce poste serveur que les images récupérées seront diffusées par Internet et via le logiciel RealPlayer sur le poste superviseur. Par contre une valeur deux de la variable se traduit par l'arrêt du procédé d'acquisition qui lui est associée. La figure 15 montre la configuration MMS de la caméra, alors que la figure 16 montre la représentation MMS partielle du VMD-Caméra.

Le poste de supervision peut établir une supervision passive du poste émulant le robot (l'interface construite simule le comportement du VMD-Robot) à travers les services Status et lecture de la variable associée à l'identification du robot. L'opérateur peut également piloter le robot via les services de variables MMS (Write) en associant une variable MMS à un programme (ou tâche robot). Localement une lecture se fait d'une manière automatique grâce aux liaisons DDE. Ensuite la liaison à l'armoire de commande du robot se fait par la liaison série RS232 via le protocole DDCMP-NCP. Une valeur à un d'une variable se traduit par le démarrage du programme robot qui lui est associée. Alors que la valeur deux entraînera l'arrêt du programme qui vient d'être enclenché.

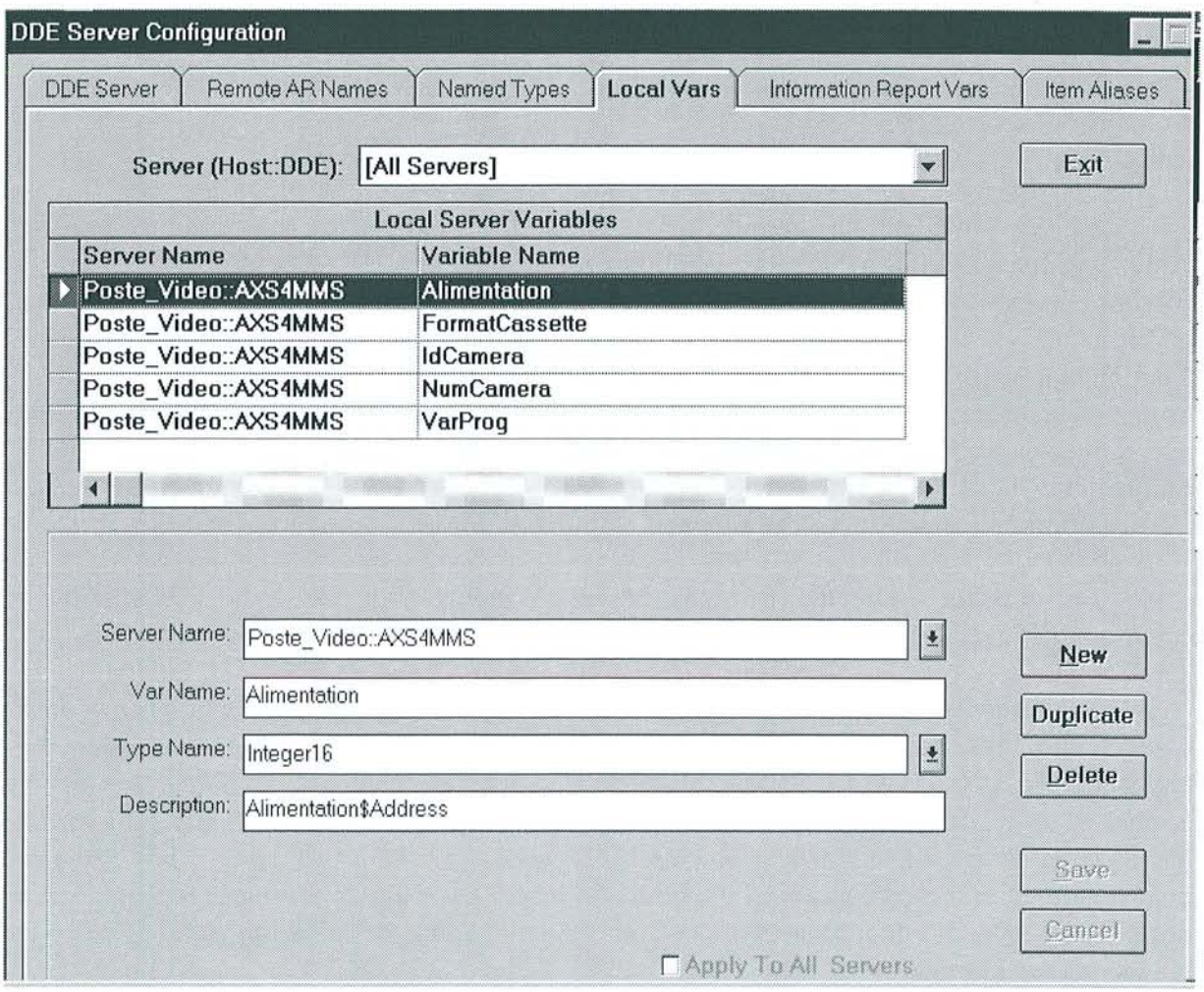

Figure 15. *Configuration MMS du poste connecté à la caméra.*

Objet : VMD_Caméra
Attribut-clé : Caméra_1
Attribut-clé : NumCaméra
Attribut : Alimentation
Attribut : TypeObjectif
Attribut : FormatCassette)
Attribut : Objectif
Attribut : Texte_Image
Attribut : Cassette

Objet : Domaine
Attribut-clé : Objectif
Attribut-clé : NomObjectif
Attribut : OuvertObjectif
Attribut : Zoom
Attribut : DiamètreFiltre

Figure 16. **La représentation MMS du VMD-Caméra et un de ses éléments.**

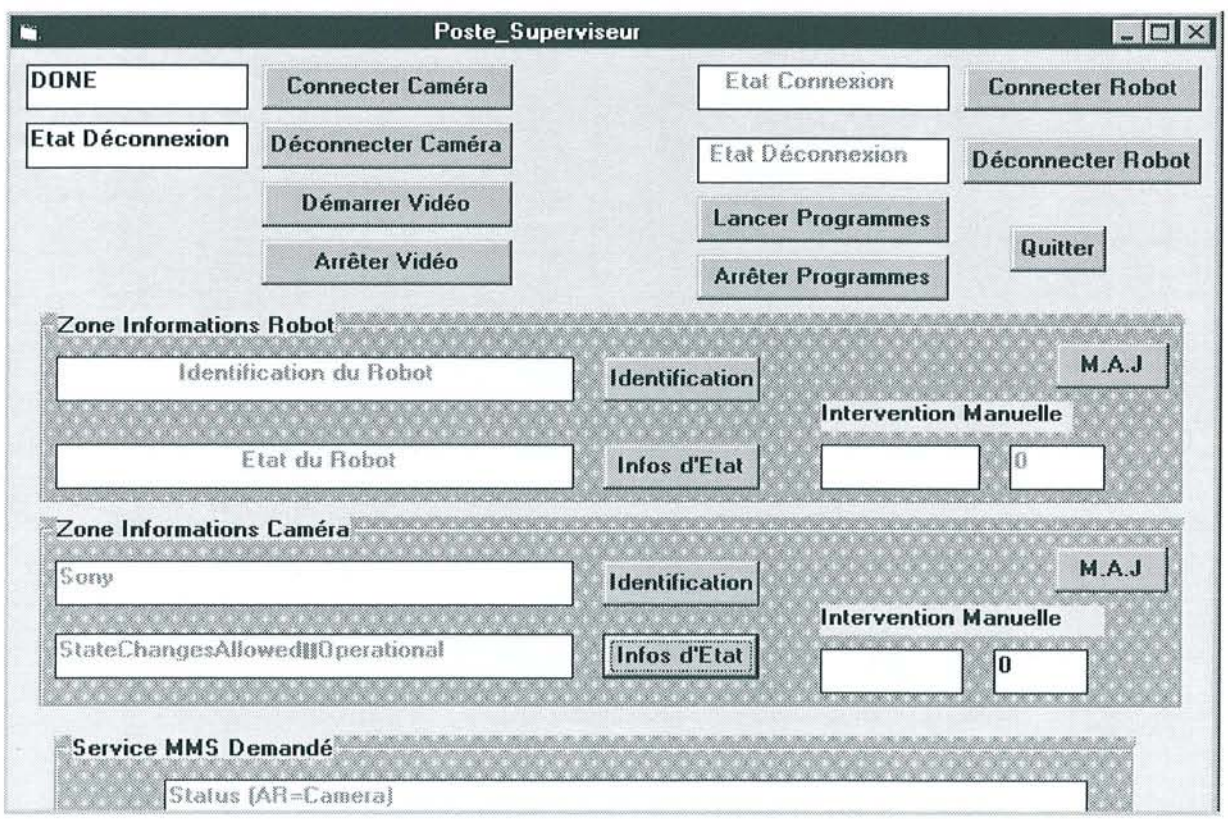

Figure 17. **Interface poste de supervision.**

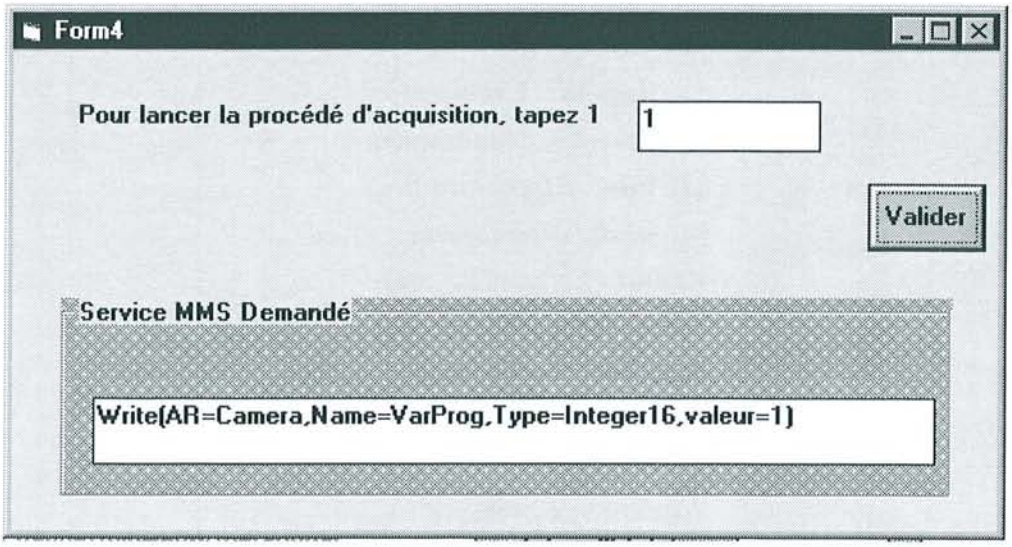

Figure 18. **Fenêtre de lancement du procédé d'acquisition de l'interface poste de supervision.**

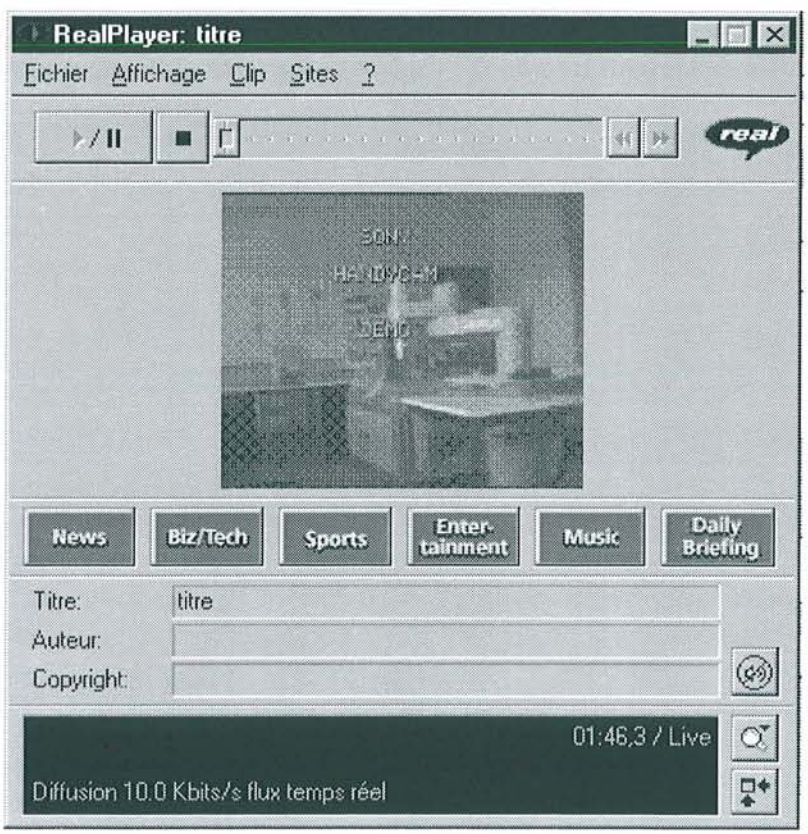

Figure 19. **Visualisation du robot via le logiciel Real Player.**

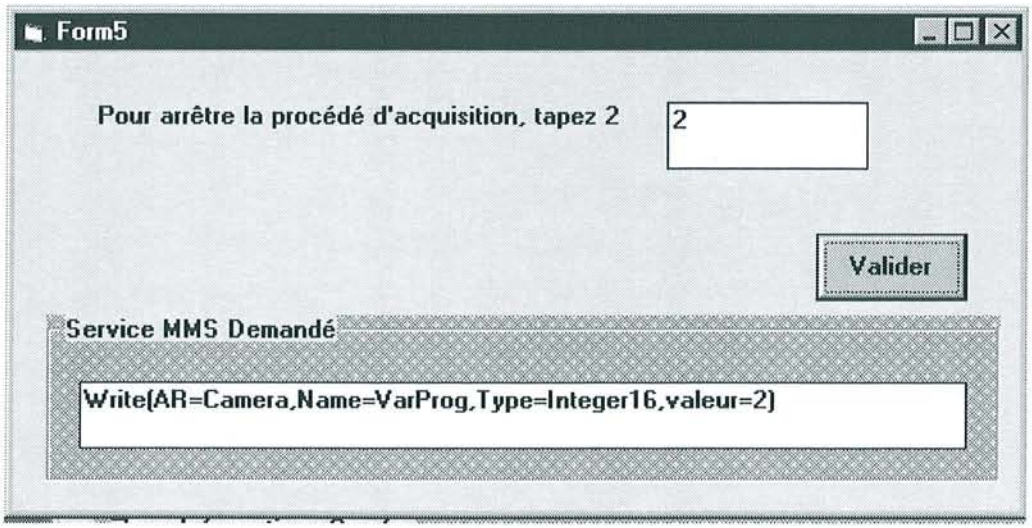

Figure 20. **Fenêtre de l'arrêt du procédé d'acquisition de l'interface poste de supervision**

Les figures 17, 18, 19, 20 concernent l'interface utilisateur définie pour le poste de supervision.

7. Conclusion

Dans ce chapitre nous avons d'abord présenté l'étude de cas d'une application de télé-pilotage d'un robot industriel (GMFanuc) avec retour vidéo par deux caméras visionnant le robot. L'objectif d'une telle étude est la validation de notre approche ainsi que la démarche et les propositions que nous avons avancées.

Pour ce faire, nous avons procédé d'abord à l'analyse de l'application en s'intéressant aux points de vue statique (modèle objet) et dynamique (diagrammes de suivi d'événements et les graphes d'états-transitions). Ensuite, nous avons appliqué notre démarche afin de déterminer les VMDs virtualisant les composants répartis de l'application du point de vue communication. La structure interne des VMDs a été définie en utilisant les conventions que nous avons proposées et qui sont basées sur les règles de transformation définies en préalable. Cela est accompagné par la transcription de la vue dynamique sur les services MMS.

A savoir que le contexte d'implémentation concernant les objets et services MMS disponibles sur les différents sites ou points de connexion (classe de conformité) a été aussi pris en compte lors de la détermination des objets et de la transcription de la dynamique en services MMS.

Ainsi, nous pouvons confirmer la faisabilité de notre approche qui peut être utilisée également pour en générer les objets et services MMS à partir d'une analyse complète du système d'information de l'application étudiée.

Conclusion et Perspectives

L'objectif primordial du concept CIM (Computer Integrated Manufacturing) est d'améliorer l'intégration des composants de l'entreprise afin d'accroître sa productivité et son efficacité. Ceci en partant sur la base d'une communication avancée, de coopération et de coordination des fonctions de l'entreprise. Dans un tel contexte, le système d'information représente un facteur important. En effet l'aspect communication prend ici toute son importance puisqu'il est le système de transfert des informations propres à l'entreprise. Il est également induit des interactions entre les différents composants répartis sur plusieurs sites pour accomplir l'objectif global visé.

Constatant la dualité information-communication, nous voulons contribuer à la définition d'un noyau d'un environnement de développement pour la conception et l'intégration des communications des systèmes automatisés distribués. La démarche adoptée possède comme point de départ l'étude du système d'information de l'application industrielle. Ainsi, nous voulons intégrer l'aspect communication dès les premières phases du cycle de développement de l'application.

Le premier chapitre a montré la dualité information-communication à travers l'intégration et le concept CIM ainsi que leur importance au sein des cadres méthodologiques d'intégration. Un bref état de l'art des méthodes d'analyse et de conception des systèmes d'information (des méthodes traditionnelles jusqu'à celles orientées objet) a été présenté. Le système de communication et le langage de commande normalisé MMS ont été explicités.

Dans cette optique nous avons présenté au chapitre 2, une formalisation unifiée des concepts OMT (classe, objet, association, ...) et du système de communication représentée par le modèle MMS (classe, objet, relation, ...). Afin de mettre en œuvre le passage du système d'information au système de communication, nous avons défini des règles de transformation. De telles règles aideront à générer les objets et services MMS à partir de l'étude structurelle. Une première validation de nos propositions est faite sur un exemple tiré de la norme MMS.

Dans un objectif d'ingénierie de la méthode, nous avons pris en compte dans le chapitre 3 les différentes contraintes imposées à la réalisation de l'application. Parmi ces contraintes apparaît la distribution des composants de l'application sur les différents sites interconnectés par le réseau local. A des fins d'automatisation du processus de transformation, nous avons eu recours à la définition de conventions liées aux aspects statique et dynamique.

Dans la pratique, les services MMS supportés par un équipement dépendent des spécificités de cet équipement. Aussi, nous avons regardé de près la notion de classe de conformité d'un équipement : l'ensemble des objets et services MMS réellement implantés sur cet équipement. Une telle notion aura une influence sur les conventions proposées et ainsi sur le choix des objets et services générés.

Ensuite dans le chapitre 4 nous avons présenté une application de télé-pilotage d'un robot industriel. L'objectif de cet exemple est la validation de nos propositions.

L'originalité de notre approche est la prise en compte des deux points de vues intégrés du système d'information afin de déterminer en sortie les objets et services MMS à mettre en œuvre pour la réalisation d'une application industrielle. L'aspect formalisation unifiée pour représenter les deux aspects des systèmes (système d'information et système de communication) renforce l'objectif de l'intégration. De plus, notre travail prend en considération les contraintes relatives aux classes de conformité des équipements industriels pour obtenir des solutions toujours opérationnelles. Cependant d'autres critères tels que fonctionnel, temporel, structurel pourraient être intégrés dans le processus de transformation afin d'affiner et optimiser le choix des objets et services MMS générés.

Ainsi, nos travaux simplifient la conception et la réalisation des applications contrôle-commande visant l'implantation MMS et fournissent une méthodologie pour le développement de celles-ci. Ils fournissent également des schémas de transformation entre une conception orientée objet et les plate-formes d'intégration existantes dans le domaine.

Notons que nos travaux rejoignent ceux visant la formalisation de MMS et de ses concepts.

Nous pouvons aussi remarquer que notre approche est applicable à toute messagerie adoptant une représentation comparable à celle de MMS. Ainsi les aspects temporels qui n'ont pas été abordés dans le cadre de la norme MMS et donc dans notre étude pourront être pris en compte dans des messageries temps-réel.

Finalement la réalisation complète de l'outil supportant notre démarche représente une des perspectives de ces travaux.

Bibliographie

(Afilal, 1989) F. Afilal, "Evaluation et Modélisation des Performances Temporelles de Messageries Industrielles", Thèse de Doctorat de l'Université Henri Poincaré, Nancy I, 6 juillet 1989.

(Akazan, 1995a) J. Akazan, Z. Mammeri, "Real Time Extensions to MMS", Proceedings IEEE International Workshop on Factory Communication Systems WFCS'95, Leysin, Switzerland, 1995, pages 193-200.

(Akazan, 1995b) J. Akazan, Z. Mammeri, "DEMMS : a Development Environment to Ease the Manufacturing Message Specification (MMS) Use", Proceedings IEEE International Symposium on Industrial Electronics, Athenes, Greece, july 1995.

(Alfter, 1992) Alfter, "Le standard PROFIBUS, le premier réseau de terrain standardisé répondant à niveau mondial aux exigences de la communication industrielle", Saint-Ouen, Paris, France, août 1992.

(Bajic, 1995) E. Bajic, "Gestion de la qualité en productique par la conduite des processus et le système d'information", Habilitation à Diriger des Recherches, Université Henri Poincaré - Nancy I, 17 mars 1995.

(Barbier, 1992) F. Barbier, P. Jaulent, "Techniques orientées objet et CIM", Editions Eyrolles, 1992.

(Bayart, 1995) M. Bayart, M. Staroswiecki, F. Simmonot-Lion, J.P. Thomesse, "La démarche de conception des systèmes automatisés distribués", Actes 3ème Colloque Bilan A2RP , Paris, novembre 1995.

(Bigand, 1998a) M. Bigand, D. Corbeel, J.P. Bourey, "Proposal of Unified Modelling Language extensions for the specification of constraints on linked objets", Multiconference Computational Engineering in System Applications CESA'98, Nabeul-Hammamet, Tunisie, 1-4 avril 1998.

(Bigand, 1998b) M. Bigand, D. Corbeel, J.P. Bourey, "Integration of view points by using an object oriented approach", IFAC, INCOM'98 9th Symposium on Information Control in Manufacturing, Advances in Industrial Engineering, Nancy-Metz, France, june 24-26, 1998, pages 523-528.

(Blaha, 1988) M.R. Blaha, W.J. Premerlani, J.E. Rumbaugh, "Relational Data Base Design Using an Object-Oriented Methodology", Communication of ACM, vol. 31, n° 4, april 1988.

(Bonder, 1993) D.A. Bonder et al, "Object Oriented Modelling and Simulation of Automated Control in Manufacturing", Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, Georgia, may 1993, vol. 3, pages 83-88.

(Booch, 1991) G. Booch, "Object-Oriented Design with Applications", Benjamin and Cummings, Redwood, City, CA, 1991.

(Booch, 1993) G. Booch, "Object-Oriented Analysis and Design with Applications", (Second Edition), Benjamin and Cummings, Redwood City, CA, 12993.

(Boufarès, 1994) F. Boufarès, F. Gargouri, N. Kraiem, "Using an Object-Oriented Methodology to Generate an Optimized Relational Scheme", 3rd Maghrebian Conference on Software Engineering and Artificial Intelligence, Marroco, april 11-14, 1994.

(Boufarès, 1996) F. Boufarès, F. Gargouri, "de la Conception Orientée Objet à l'Implantation Relationnelle de OMT à C/SQL2", Colloque National -Recherches Universitaires en Mathématique, Statistiques, Informatique et leurs applications : l'exemple des IUTs, Clermont-Ferrand, 14-16 fevrier, 1996.

(Brill, 1991) M. Brill, U. Gramm, "MMS : MAP application services for the manufacturing industry", Computer Networks and ISDN systems, 1991, North Holland, vol. 21, n° 21, pages 357-380.

(Caod, 1991a) P. Caod, E. Yourdon, "Object-Oriented Analysis", (Second Edition), Prentice Hall, Englewood Cliffs, NJ, 1991, 200 pages.

(Caod, 1991b) P. Caod, E. Yourdon, "Object-Oriented Design", Prentice Hall, Englewood Cliffs, NJ, 1991, 200 pages.

(Caod, 1993) P. Caod, J. Nicola, "Object-Oriented Programming", Prentice Hall, Englewood Cliffs, NJ, 1993.

(Castori, 1995a) P. Castori, P. Pleinevaux, "A Generic Architecture for MMS Servers", Proceedings of IEEE International Wokshop on Factory Communications Systems, Leysin, Switzerland, oct. 4-6, 1995, Pages 211-218.

(Castori, 1995b) P. castori, "On Schedeling of MMS Services", Proc. of IEEE Int. Workshop on Factory Communications Systems, Leysin, Switzerland, oct. 4-6, 1995, Pages 219-224.

(Castori, 1996) P. Castori, "Architecture de systèmes dans l'environnement MMS", Thèse de Doctorat en Sciences Techniques, Ecole Polytechnique Fédérale de Lausanne, 1996.

(Chen, 1976) P.P.S. Chen, "The entity-relationship model : toward a unified view of data", ACM transactions on database systems, vol. 1, n° 1, march 1976, pages 9-36.

(Chen, 1990) D. Chen, B. Vallespir, M. Zanettin, "Architecture globale CIM : Application dans le projet ESPRIT-IMPACS", CIM'90 International Conference, 12-14 juin 1990, Bordeaux, pages 243-250.

(Claudé, 1993) J.P et M. Claudé, "Gestion des Réseaux Informatiques", Collection réseaux et systèmes, Editions Eyrolles, ISBN 2-212-08786-1, 1993.

(Coulange, 1996) B. Coulange, "Réutilisation du logiciel", Editions Masson, 1996, 324 pages.

(CSI) CSIRO (Commonwealth and Industrial Research Organization), Rapport de recherche de la "Division of Manufacturing Technology", http://www.csiro.au.

(Daoudi, 1994) K. Daoudi, "Modélisation et Gestion de Performances de Services de la Messagerie Industrielle MMS (Manufacturing Message Specification)", Thèse de Doctorat de l'Université Henri Poincaré, Nancy I, 2 décembre 1994.

(Darscht, 1995) P. Darscht, A.H. Frigeri, C.E. Pereira, "Building up Object-Oriented Automation Systems : Experiences Interfacing Active Objects with Technical Plants", IEEE International Workshop on Factory Communication Systems WFCS'95, Leysin, Switzerland, 4-6 october, 1995.

(Divoux, 1996) T. Divoux, "Intégration des communications des systèmes automatisés distribués", Habilitation à Diriger des Recherches de l'Université Henri Poincaré, Nancy I, 20 décembre 1996.

(Divoux, 1997) T. Divoux, D. E. Rondeau, F. Lepage, " Using the EXPRESS language as a reference interface to define MMS communication", Journal of International Manufacturing, vol. 8, n°. 1, February 1997, pages 59-66.

(Doumeingts, 1990) G. Doumeingts, "Design and Specification methods for production systems", CIM'90 International Conference, 12-14 juin 1990, Bordeaux, pages 89-103.

(Doumeingts, 1992) G. Doumeingts, G. Vallespir, B. Zanettin, D. Chen, "GIM GRAI Integrated Methodology : a methodology for designing CIM systems", version 1.0, LAP/GRAI, Université de Bordeaux 1, France, mai 1992.

(Ducateau, 1996) C.F. Ducateau, "Fondements Formels de la Multi-Instanciation", Journées de synthèse-Points fondamentaux et expériences d'analyse et de conception par objet, 24-25-26 janvier, 1996, Paris.

(ECIMA, 1992) ECIMA Consortium, "CIM-OSA", ESPRIT project 5288, Mileston M-2, AD 2.0, Volume 2 : Architecture Description, Document R0443/1, Brussels, Belgium, 24 août 1992.

(Elloy, 1989) J.P. Elloy, T. Laine, "Besoin et rôle des normes d'accompagnement", FIP et les constituants «intelligents» d'automatismes : Capteurs, Actionneurs, Régulateurs et Automates, Journées d'étude, Paris, France, 29-30 novembre, 1989.

(Fabbricino, 1994) A. Fabbricino, "The customer-foccused entreprise", Actes du 10th International Conference on CAD/ CAM, Robotics and Factories of the Future, CARs and FOF, congrès ISPE/IFAC, pages 19-27, Ottawa (Canada), 21-24 août 1994.

(Fofana, 1996) M.F. Fofana et al, "Application de la méthode OMT à l'Analyse de la Commande d'un poste d'usinage", AFCET Modélisation des systèmes réactifs, Brest, France, 1996, pages 203-210.

(Fofana, 1997) M.F. Fofana, Spécification et Prototypage d'un Système de Conduite Technique d'Atelier", Thèse de Doctorat à l'Université Paris-Nord, 11 janvier 1997.

(Foulard, 1994) C. Foulard, "La modélisation en entreprise, CIM-OSA et ingénierie simultanée", Edition Hermès, Paris, 1994.

(Fukuhara, 1991) N. Fukuhara, "CIM state of the art in Japan", Computer Application in production and engineering : integration aspects, G. Doumeinggts, J. Browne and M. Tomljanovich Editors, Elsevier Science Publishers BV (North-Holland), 1991.

(Gaches, 1990) R. Gaches, B. Querenet, P. Viollet, F.B. Vernadat, "CIM-OSA : une architecture ouverte pour la productique", CIM'90 International Conference, 12-14 juin 1990, Bordeaux, pages 227-234.

(Gargouri, 1995a) F. Gargouri, F. Boufarès, C.F. Ducateau, "Relational Implementation of Object Oriented Information System Design Using a Generic Model", International Conference on Object Oriented Information Systems OOIS'94, London, UK, December, 1994, pages 114-129.

(Gargouri, 1995b) F. Gargouri, C.F. Ducateau, F. Boufarès, "Towards a Generic Model for Object-Oriented Information System Modelling", International Conference on Industrial Engineering and Production Managment, Marocco, April 4-7, 1995.

(Gargouri, 1997) F. Gargouri, C.F. Ducateau, F. Boufarès, "Towards a generic model for Object-Oriented Information System Modeling", Journal of Intelligent Manufacturing, Chapman&Hall Edition, Vol. 8, No. 1, Feb 1997.

(Govindaraj, 1993) T. Govindaraj et al, "OOSIM : A Tool for Simulation Modern Manufacturing Systems", Proceedings of the 1993 NSF Design and Manufacturing Systems Grantees Conference, Charlotte, North California, january 1993, pages 1055-1062.

(Griffin, 1993) C.M Griffin , J.M.P. Kendall, "A framework for an advanced NIAM to EXPRESS converter", Actes de la 3ème EXPRESS User's Group International Conference, EUG'93, Berlin, 2-3 octobre, 1993.

(Guyonnet, 1997) G. Guyonnet, E. Gressier-Doudan, J.Y. Simiand, F. Weis, "Une approche CORBA pour MMS : COOL-MMS", RTS'97, Paris, january 1997.

(Habrias, 1988) H. Habrias, "Le Modèle Relationnel Binaire, Méthode I.A. (NIAM) ", Editions Eyrolles, 1988.

(Harel, 1987) D. Harel, "Statecharts : A Visual formalism for complex systems", Science of Computer Programming, 1987, pages 231-274.

(Harel, 1996) D. Harel, "Executable object modelling with Statecharts", Proc. 18th Soft. Eng, Berlin, IEEE Press, March, 1996, pages 246-275.

(Henault, 1995) L. Henault, D. Le Ray, "La messagerie MMS", mars 1995. http://www-ensimag.imag.fr/profs/cours/Exposes.Reseaux/RLI/MMS/mms.html.

(Idelmerfaa, 1994) Z. Idelmerfaa, "Méthodologie de génération de la conduite d'un système intégré de production", Thèse de Doctorat de l'Université Henri Poincaré – Nancy I, 17 février 1994.

(IEC, 1996a) "TASE.2/ICCP : Services and Protocol", IEC 870-6-503, Version 6.0, Prepared by : Utility Communications Specification Working Group, april, 1996.

(IEC, 1996b) "TASE.2/ICCP : Object Models", IEC 870-6-802, Version 6.0, Prepared by : Utility Communications Specification Working Group, april, 1996.

(Ishikawa, 1984) K. Ishikawa, "Le TQC ou la qualité à la japonaise", Collection AFNOR, Editions Eyrolles, 1984.

(ISO, 1994a) "Road Vehicles – Low-Speed serial data communication – Part 1 : General and Definitions", ISO 11519-1, First edition, 15-06 1994.

(ISO, 1994b) "Road Vehicles – Low-Speed serial data communication – Part 2 : Low-Speed controller area network (CAN) ", ISO 11519-2, First edition, 15-06 1994, Amendment 1 : 01-04 1995.

(ISO, 1994c) "Road Vehicles – Low-Speed serial data communication – Part 3 : Vehicle area network (VAN) ", ISO 11519-3, First edition, 15-06 1994, Amendment 1 : 01-04 1995.

(ISO1, 1987) "Manufacturing Message Specification – Service definition", International Standard ISO 9506/1.

(ISO3, 1991) "Manufacturing Message Specification – Part 3 : Robot Companion Standard to MMS", ISO/DP 9506/3, ISO/TC 184/SC 5/WG 2 N178, 1991.

(ISO4, 1989) "Manufacturing Message Specification – Part 4 : Companion Standard for Numerical Control", ISO/DP 9506/4, ISO/TC 184/SC 5/WG 2 N177, 1989.

(ISO5, 1989) "Manufacturing Message Specification – Part 5 : « Companion Standard for Programmable Controllers", ISO/DP 9506/5, ISO/TC 184/SC 5/WG 2 N176, 1989.

(ISO6, 1990) "Manufacturing Message Specification – Part 6 : « Companion Standard for Process Control", ISO/DP 9506/6, 1990.

(IUC, 1992) Industry-University Consortium, Purdue Laboratory for Applied Industrial Control, "An implementation procedures manual for developping master plans for CIM", Report n° 155, West Lafayette IN, USA, juin 1992.

(Jackson, 1983) Jackson, "Jackson System Development", Prentice-Hall, 1983.

(Jacobson, 1992) I. Jacobson, M. Christerson, P. Jonsson and G. Overgaard, "Object-Oriented Software Engineering : a use case driven approach", Addison-Wesley, 1992.

(Jochem, 1989) R. Jochem, M. Rabe, W. Sussenguth, "An Object Oriented Analysis and Design for Computer Integrated Manufacturing Systems", TOOLS'89 Technology of Object Oriented Languages and Systems, CNIT Paris, La Défence, France, novembre 13-15, 1989.

(Kapp, 1995) K.H. Kapp, "System Platforms for extended client-servers architecture", Workshop CRAN-IMA/UKA –IAR on product and object oriented automation, Nancy, 4 juillet 1995.

(Kiencke, 1998) U. Kiencke, D. John, A. Maisch, "Standardisation of Open Systems and corresponding interfaces for automotive electronics OSEK/VDX", INCOM'1998, 9th Symposium on Information Control in Manufacturing, Advances in Industrial Engineering, june 24-26, Nancy_Metz, France, 1998.

(Klittich, 1990) M. Klittich, "CIM-OSA Part 3 : CIM-OSA integrating infrastructure-the operational basis for integrated manufacturing systems", International Journal of Manufacturing, 1990, vol. 3, n° 3 and 4, pages 168-180.

(Kosanke, 1990) K. Kosanke, "CIM-OSA : its role in manufacturing control", proceedings of the 11th triennal world congress of the International Federation of Automatic Control, Vol 5, pages 309-313, 1990.

(Kosanke, 1992) K. Kosanke, "Open System Architecture (CIM-OSA) ; an european prenormative development", CARs & FOF, 8th international conference on CAD/CAM, Robotics and Factories of the Future, Metz, France, 17-19 august 1992, pages 486-498.

(Kraiem, 1993) N. Kraiem, F. Gargouri, F. Boufarès, "From Object-Oriented Design Towards Object-Oriented Programming", 5th International Conference on Advanced Information System Engineering CAISE'93, Paris, 8-11 juin, 1993, pages 416-430.

(Laroque, 1995) P. Laroque, P. Perrin, "Présentation de la méthode Object Modelling Technique (OMT) ", AFCET Groupe de travail COOSI, 1995.

(Larvet, 1994) P. Larvet, "Analyse des systèmes : de l'approche fonctionnelle à l'approche objet", InterEditions, Pris, 1994.

(Le Bris, 1993) C. Le Bris, M. Augeraud, M.C. Lafaye, "Présentation de la méthode de conception orientée objet : HOOD", Actes des Journées de Synthèse : Méthodes d'Analyse et de Conception Orientées objet des Systèmes d'Information, 22-23 novembre 1993.

(Lederhofer, 1992) A. Lederhofer, "CNMA communications network for manufacturing applications : how CNMA technology is helping to resolve users problems", Actes de la 8th International Conference on CAD/CAM, Robotics and Factories of the Future, CARs & FOF, Metz, France, 17-19 aout 1992, pages 534-548.

(Lefebvre, 1995) M. Lefebvre, E. Gressier-Soudan, S. Natkin, "MMS sur TCP/IP : une nouvelle solution pour l'échange de données en informatique de production", RTS'95, Paris, january, 1995.

(Lepage, 1991) F. Lepage et al., "Les Réseaux Locaux Industriels", Hermès, 2e édition, Paris, 1991.

(LOV, 1995) "LOV/OMT", Manuels de Référence, VERILOG, 1995.

(Luttenbacher, 1995) D. Luttenbacher et al, "Intelligent Sensor : Object", Control Eng. Practice, vol. 3, n° 6, 1995, pages 805-812.

(Mackiewicz, 1994) R. Mackiewicz, "An overview to the Manufacturing Message Specification", SISCO, Inc, 1994.

(Mansour, 1998a) I. Mansour, E. Rondeau, T. Divoux, "MMS Companion Standard Specification with OMT Methodology", IX Workshop on Supervising and Diagnostics of Machining Symstems : Manufacturing Simulation for Industrial Use, Karpacz, Poland, march 22-27, 1998.

(Mansour, 1998b) I. Mansour, E. Rondeau , T. Divoux, "OMT Modelling to Specify MMS Companion Standard", Computational Engineering in Systems Applications CESA'98, Nabeul-Hammamet, Tunisia, april 1-4 1998.

(Mansour, 1998c) I. Mansour, "Modélisation de l'application pilote PLACOMAP", Séminaire, Journée thématique SCI (Systèmes de Communication Industriels) du 5 mars, Paris, 1998.

(Marca, 1988) D. Marca, C. McGowan, "SADT : Structured Analysis and Design Technique", McGraw-Hill, 1988.

(Menga, 1989) G. Menga, M. Morisio, G. Lo Russo, "A Framework for Object Oriented Design and Prototyping of Manufacturing Systems)", TOOLS'89 Technology of Object Oriented Languages and Systems, CNIT Paris, La Défence, France, novembre 13-15, 1989.

(Mertins, 1991) K. Mertins, W. Sussenguth, R. Jochem, "Integrated Information modelling for CIM : An object-oriented method for integrated enterprise modelling", Computer Application in production and Engineering : Integration aspects, G. Doumeingts, J. Browne and M. Tomljanovich Editors, Elsevier Science Publishers BV (North-Holland), 1991.

(Morris, 1990) K. Morris, "Translating EXPRESS to SQL : A User's Guide", National PDES Testbed Report Series, 8 may 1990.

(Muller, 1997) P.A. Muller, "Instant UML", 12 Decembre 1997, 342 pages.

(Narayanan, 1992) S. Narayanan et al, "Object Oriented Simulation to Support Modelling and Control of Automated Manufacturing Systems", Proceedings of the 1992 Western Multiconference, Society of Computer Simulation, Newport Beach, CA, january 1992, pages 59-63.

(Patz, 1990) M. Patz, "Manufacturing Message Specification (MMS) and its Companion Standards", MAP Workshop : MMS and Application,

SYSTEC'90, Munich, Allemagne, 24 octobre 1990.

(Pleinevaux, 1994a) P. Pleinevaux, « Integration of industrial applications : The CCE-CNMA appraoch », Actes de l'European Workshop on Integrated Manufacturing Systems Engineering IMSE'94, Grenoble, France, 12-14 octobre, pages 511-517.

(Pleinevaux, 1994b) P. Pleinevaux, « An Analysis of the MMS Object Model », IEEE Transactions on Industrial Electronics, vol. 41, n° 3, june 1994.

(Pokam, 1995) M.R. Pokam, "Extension multimédia de la messagerie MMS", Thèse de doctorat de l'Institut National Polytechnique de Grenoble, 1995.

(Pouillat, 1993) A. Pouillat, L. Rosin, M.C. Lafaye, « La méthode MCO de X. Castellani », Actes des journées de synthèse : Méthodes d'analyse et de conception orientées objet des systèmes d'information, 22-23 novembre 1993.

(Querenet, 1991) B. Querenet, « The CIM-OSA integrating infrastructure », Computing & Control Engineering Journal, may 1991, vol. 2, n) 3, pages 118-125.

(Rae, 1989) A.K. Rae, « An MMS companion standard for production management », University of Strathclyde, Draft PMCS specification, N175-89, 1989.

(Rodd, 1990) M.G. Rodd, G-F Zhao and I. Izigowitz, "RTMMS - An OSI-Based Real-Time Message System", The Journal of Real-Time Systems, 2, pages 213-234, 1990.

(Rolland, 1988) Rolland, Foucault, Benci, "Conception des systèmes d'information", Eyrolles, 1988.

(Rolland, 1993) C. Rolland, C. Cauvet, J. Brunet, "La méthode O*", Actes des journées de synthèse : Méthodes d'analyse et de conception orientées objet des systèmes d'information, 22-23 novembre 1993.

(Rondeau, 1993) E. Rondeau, "Intégration des communications MMS (Manufacturing Message Specification) par l'étude du système d'information", Thèse de Doctorat de l'Université Henri Poincaré, Nancy I, 9 fevrier, 1993.

(Rumbaugh, 1991) J. Rumbaugh et al., "Object Oriented Modelling and Design", Prentice-Hall, International Editions, 1991, ISBN 0-13-630054-5.

(Rumbaugh, 1995) J. Rumbaugh et al., "OMT : Modélisation et Conception Orientée objet", édition française revue et augmentée, Editions Masson et Prentice-Hall, 1995, ISBN 2-225-84684-7.

(Russel, 1991) P.J. Russel, "Modelling with CIM-OSA", Computing & Control Engineering Journal, 2, (3), 1991, pages 109-117.

(Sanderson, 1993a) D. Sanderson, D. Spooner, "Mapping between EXPRESS and traditional DBMS Models", EXPRESS User's Group 93, 263 october, 1993.

(Sanderson, 1993b) D. Sanderson, D. Spooner, "Mapping between EXPRESS and the Extended Entity Relationship model", Departement of Computer Science, Technical Report, february 1993.

(Schweizer, 1995) G. Schweizer, "Project managment and upcoming standards", Workshop CRAN - IMA/UKA - IAR on product and object oriented automation, Nancy, 4 juillet 1995.

(Selic, 1998) Bran Selic, "Using UML for Modeling Complex Real-Time Systems", ObjectTime Limited, Jim Rumbaugh, Rational Software Corporation, march 11, 1998.

(Shlaer, 1988) S. Shlaer, S.J. Mellor, "Object-Oriented Systems Analysis : Modelling the World in Data", Prentice-Hall International Editions, 1988.

(Tardieu, 1987) H. Tardieu, A. Rochfeld, R. Coletti, "La méthode MERISE", tome 1 et 2, Editions d'Organisation, 1987.

(Thomas, 1997) C. Thomas, J.Y. Bron, F. Lepage, "Contribution to the Definition of a Communication Structure Based On MMS in a Production Managment Context", IFAC Conference on Control of Industrial Systems, Belfort, France, 20-22 may 1997.

(Vernadat, 1989) F. Vernadat, A. Dileva, P. Giolito, "Organisation and information system design of manufacturing environment : the new M* approach", Computer Integrated Manufacturing Systems, vol. 2, n° 2, 1989, pages 69-81.

(Vernadat, 1990) F. Vernadat, "Modelling and analysis of enterprise information systems with CIM-OSA", Computer Integrated Manufacturing proceedings of the sixth CIM-Europe annual conference, Lisbon, Portugal, pages 16-27, 1990.

(Vernadat, 1993) F. Vernadat, "CIM-OSA : Enterprise modelling and integration using a process based appraoch", In H. Yoshikawa and J. Goosenaerts, editors, Preprints of JSPE-IFIP WG 5.3 Workshop on the Design of Information Infrastructure Systems for Manufacturing – DIISM'93, Tokyo, Japan, November 8-10, Japan society for Precision Engineering, pages 161-175.

(VOICE, 1993) VOICE Consortium, "VOICE Architecture of the Integrating Infrastrucure", VOICE Open Workshop, Berlin IPK, 14 septembre 1993.

(Ward, 1986) P. Ward, S. Mellor, "Structured Development of Real-Time Systems", Englewood Cliffs, New Jersey : Yourdon Press, 1986.

(Weston, 1993) R.H. Weston, "Step towards enterprise-wide integration : a definition of need and first generation open solutions", International Journal of Production Research, Ed Taylor and Francis, vol. 31, n 9, pages 2235-2254, septembre 1993.

(Weston, 1994) R.H. Weston, P. Clements, I.S. Mugatroyd, "Information modelling methods and tools for manufacturing systems", Actes du 27th ISATA Lean/agile manufacturing in automative industries, Aachen, Germany, 31 octobre-4 novembre1994, pages 227-234.

(Williams, 1994) T.J. Williams, P. Bernus, J. Brosvic, D. Chen, G. Doumeingts, L.

Nemes, J.L. Nevins, B. Vallespir, J. Vieltstra, D. Zoetekouw, "Architectures for integrating manufacturing activities and entreprises", IFIP Transactions B-17, Towards World Class Manufacturing, M.J. Wozny, G. Olling (Editeurs), Elsevier Science B.V. (North-Holland) ISSN 0926-5481, pages 181-194, 1994.

(Wilson, 1994) M. Wilson, C. Aguiar, I. Coutts, R.H. Weston, "Model Enactment as basis for rapid prototyping of manufacturing systems", Actes de l'European Workshop on Integrated Manufacturing Systems Engineering IMSE'94, Grenoble, France, 12-14 octobre 1994, pages 86-96.

(Wu, 1995) B. Wu, "Object-oriented system analysis and definition of manufacturing operations", International Journal of Production Research, vol. 33, n° 4, april 1995, pages 955-974.

(Wu, 1996) Q. Wu, I. Mansour, T. Divoux, "Network Resource Management For Multimedia Manufacturing Message Service", Proceedings of the 3rd International Workshop on Protocols for Multimedia systems (PROMS'96), October, 1996, Madrid, pp. 163-176.

(Wu, 1997) Q. Wu, "Contribution à l'intégration de communications multimédias dans un environnement MMS", Thèse de Doctorat de l'Université Henri Poincaré, Nancy-I, juin 1997, France.

(Zaytoon, 1995) J. Zaytoon, M. Niclet, V. Carre-Menetrier, G. Villermain-Lecolier, "Décomposition et Allocation des Tâches en Génie Automatique : Une Approche Objet", 2nd International Conference on Industrial Automation, 7-9 june, 1995, Nancy, France.

(Zgorzelski, 1994) M. Zgorzelski, "Enterprise reengineering : Structured and object-oriented techniques for enterprise modelling", Actes du 5th "GMI Engineering and Management Institute / Industry" Symposium, Flint, MI USA, 13 septembre, 1994.

Annexe

Elaboration d'une norme d'accompagnement (exemple robot)

1. Introduction

La norme MMS offre une description générique, à travers le modèle de comportement externe d'un équipement de production défini à l'aide de l'entité VMD, et elle n'est pas associée à une catégorie d'équipements particuliers. C'est la raison pour laquelle une spécialisation de la norme MMS a été entreprise afin de représenter des types d'applications spécifiques et répandues dans le milieu industriel. Ainsi des normes appelées normes d'accompagnement ont été élaborées. Ces dernières représentent une personnalisation du modèle générique pour différents types d'équipements.

Il est à noter qu'avant de développer une norme d'accompagnement, il faut justifier la coopération des entités distantes interconnectées par un réseau local industriel, dégager les fonctionnalités et les informations supportées par l'équipement, les modéliser et en proposer une nomenclature normalisée (Rondeau, 1993).

Rappelons également que les normes d'accompagnement évitent la création de VMD constructeurs ainsi qu'une mauvaise interprétation de la norme par des non-spécialistes (Patz, 1990). Toute norme d'accompagnement s'adresse à une classe particulière d'applications. Par classe d'application on peut alors comprendre aussi bien type d'équipement que type de fonctionnalité attendue d'un ensemble d'équipements.

Il existe deux types de normes d'accompagnement (Patz, 1990) : normes d'accompagnement opérationnelles qui décrivent un type de matériel spécifique (exemple les normes citées plus haut), des normes d'accompagnement fonctionnelles qui décrivent un type de fonction. Une norme d'accompagnement fonctionnelle se distingue de celle dite opérationnelle par le fait qu'elle détermine une approche exhaustive d'expression des besoins des utilisateurs et que son contenu et sa structure quelle que soit l'évolution technologique à venir, ne seront plus remis en cause. Par contre des modifications technologiques entraînant de nouvelles performances pour les équipements nécessiteront une reconsidération de la norme opérationnelle. La norme d'accompagnement propre à la gestion de production (Rae, 1989) s'inscrivait dans la catégorie normes d'accompagnement fonctionnelles. Mais le groupe de travail chargé de cette mission l'a abandonnée.

Il existe des normes d'accompagnement (dites opérationnelles) déjà définies pour quelques classes d'équipement tels que robot (ISO3, 1991), commande numérique (ISO4, 1989), le contrôle de processus (ISO6, 1990), les automates programmables (ISO5, 1989).

2. Processus d'élaboration d'une norme d'accompagnement

Les étapes du processus suivi pour élaborer une norme d'accompagnement sont les suivantes (ISO1) :

- tout d'abord, un modèle abstrait de l'équipement réel est construit (description de l'application),

- mise en correspondance des objets abstraits MMS avec les objets réels de l'application,

- les services MMS disponibles (classe de conformité de l'application),

- définition de nouveaux services spécifiques à l'application,

- définition d'objets spécifiques à l'application.

Nous allons décrire les étapes suivies qui ont permis d'écrire la norme d'accompagnement robot.

3. Modélisation du système robot

Le système d'information du système robot doit être établi ainsi que celui concernant la description de sa partie opérationnelle (ISO3, 1989). Comme le montre la figure 1, un robot est composé d'une interface de communication, d'un programme, d'un ou plusieurs bras et de zéro ou plusieurs équipements auxiliaires. Ces derniers peuvent être des Capteurs, des Préhenseurs, un Système de Vision, un Système de Peinture, un Système d'Urgence, ...

Le Bras du robot est composé des objets suivants : la Pince, le Servomécanisme et la Commande numérique de suivi de trajectoires. Cette dernière accepte les commandes du système de contrôle du Robot pour le mouvement du Bras et convertit ces requêtes en des séries temporelles des commandes de l'objet servomécanisme.

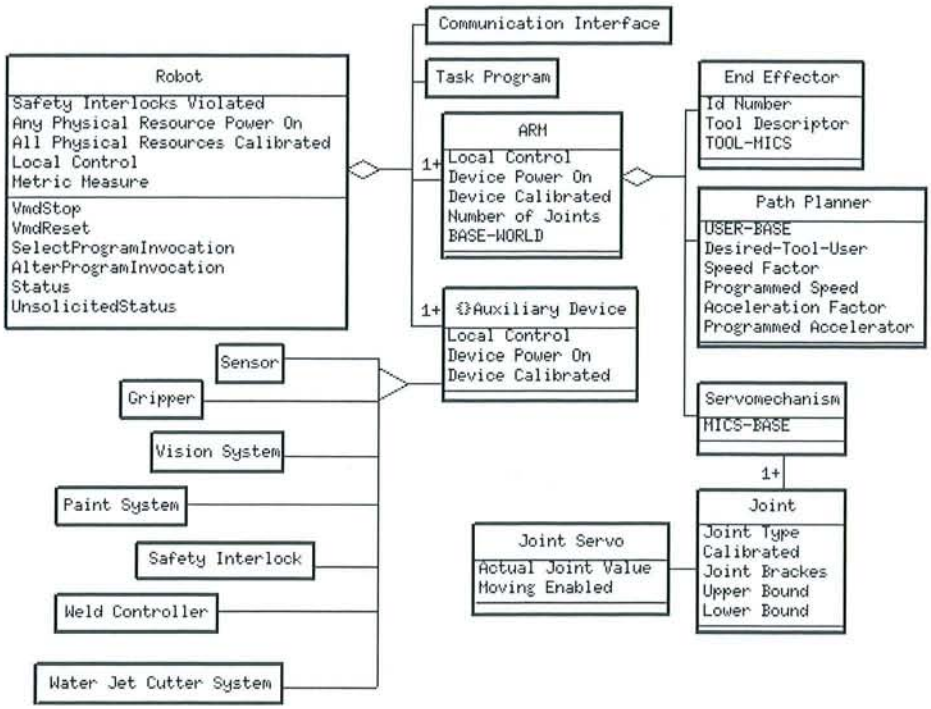

Figure 1. Le modèle objet du système robot.

La figure 2 montre le graphe d'états du robot (simplifié).

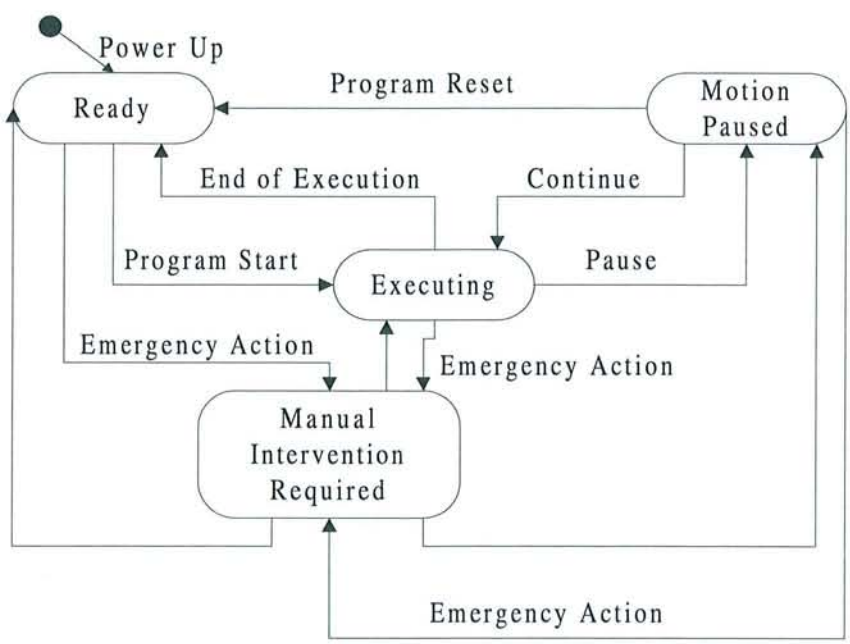

Figure 2. Le graphe d'états du système Robot.

Lors de l'alimentation du robot, il passe dans l'état Ready (son état initial). En recevant la demande 'Program Start', le robot passe à l'état Executing et le robot est alors en mouvement. Si l'exécution du programme est terminée, le robot revient à l'état Ready à moins qu'il reçoive la demande 'Pause', le robot passe à l'état Motion-Paused et le robot

s'arrête. Dans ce dernier état, deux issues sont possibles : soit le robot reçoit la demande de réinitialisation 'Program Reset' et dans ce cas il passe à l'état Ready, soit il reçoit la demande de reprise d'exécution de son programme 'Continue' ainsi le robot revient à l'état Executing et l'exécution du programme est reprise de nouveau. A la réception de la demande 'Emergency Action', à partir de n'importe quel état, le robot entre dans l'état Manual Intervention Required. Une action locale est exigée de la part de l'opérateur. Suite à l'intervention de l'opérateur, le Robot peut se trouver dans un des états.

4. Mapping du modèle objet du système robot sur les objets MMS

4.1. Introduction

Nous procédons à une projection du modèle objet du système robot sur le modèle objet MMS (Figure 3). Ainsi, le robot est identifié par l'intermédiaire d'un VMD qui lui est propre. Le VMD-Robot aura, en plus des attributs associés au VMD et conformes à la norme, quelques attributs supplémentaires qui décrivent le robot. La classe d'objets bras, aussi bien que les équipements auxiliaires hériteront de la classe domaine. Le programme robot quant à lui héritera de la classe d'objets instance de programme.

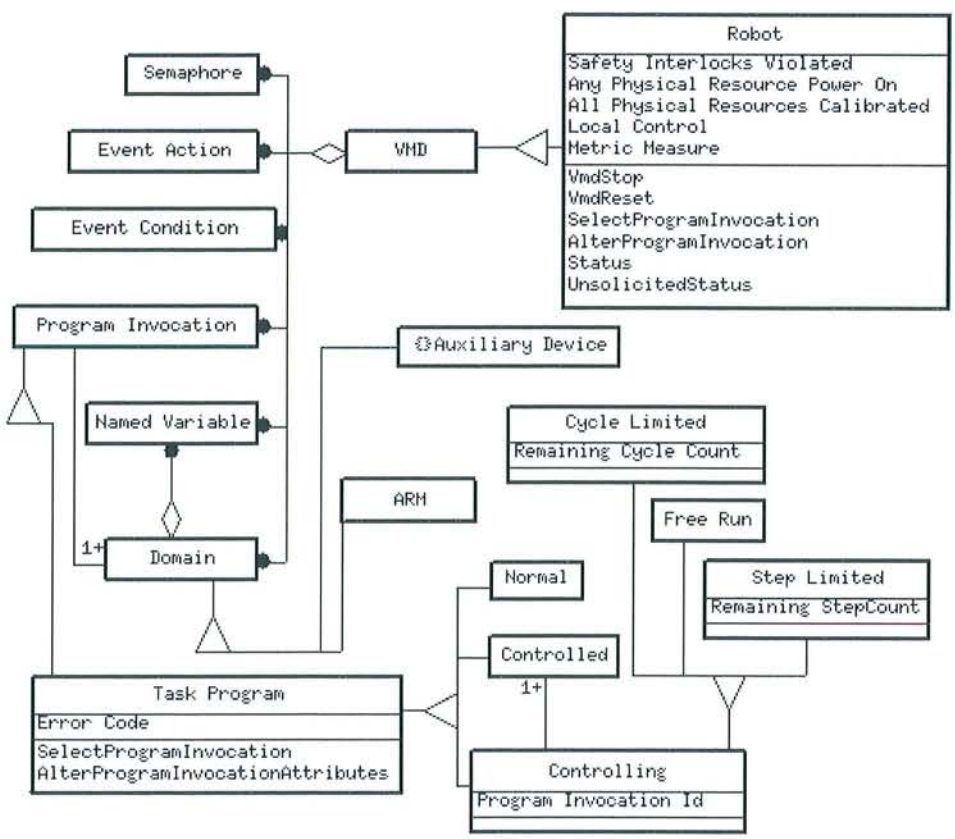

Figure 3. La projection du modèle objet du système robot sur le modèle objet MMS.

L'ensemble des services MMS permettant à un client d'accéder au robot est défini. Cet ensemble est constitué des services traditionnels tels que services de gestion de l'environnement, le service status, ... Quelques nouveaux services tels que : VMD-Stop, VMD-Reset, AlterProgramInvocation et Select sont ajoutés. Il est à noter que des

modifications ou ajouts de champs ont été effectués. Le service de création de l'objet instance de programme (program invocation) en est l'exemple. Ce dernier a un nouveau champ appelé control qui permet à l'instance de programme (programme contrôlant) de contrôler plusieurs autres instances de programmes (programmes contrôlés). Ainsi, la coopération entre deux instances de programme est autorisée au sein du VMD.

4.2. Application des règles de transformation

Nous allons utiliser les règles que nous avions définies dans le chapitre 2 tout en se basant sur le modèle OMT de l'objet VMD de MMS.

Ces règles seront appliquées à toutes les classes d'objets, avec leurs propriétés et leurs relations, constituant le modèle objet du système Robot.

4.2.1. Règles concernant la classe Robot

Tout d'abord, nous appliquons les règles concernant la classe Robot avec ses propriétés (attributs et opérations), ensuite nous appliquons la relation existante entre elle et la classe VMD.

$$R_{class} :$$
$$\| Trans(OMT, Robot_{omt}) = Robot_{mms}$$

$$\forall At_{omt_i} \in \gamma(Robot_{omt}) = \{ Safety\ Interlok\ Violated, Any\ Physical\ Resource\ Power\ On,$$
$$All\ Physical\ Resources\ Calibrated, Local\ Control, Metric\ Measure \}$$
$$/\ i = 1...5 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = At_{mms_i} \quad avec$$
$$At_{mms_i}\_Nme = At_{omt_i}\_Nme$$
$$At_{mms_i}\_Typ = At_{omt_i}\_Typ,$$

$$At_{mms_1}\_Nme = Safety\ Interlok\ Violated \quad , \quad At_{mms_1}\_Typ = boolean,$$
$$At_{mms_2}\_Nme = Any\ Physical\ Resource\ Power\ On \ , \ At_{mms_2}\_Typ = boolean,$$
$$At_{mms_3}\_Nme = All\ Physical\ Resources\ Calibrated \ , \ At_{mms_3}\_Typ = boolean,$$
$$At_{mms_4}\_Nme = Local\ Control, \quad At_{mms_4}\_Typ = boolean,$$
$$At_{mms_5}\_Nme = Metric\ Measure, \quad At_{mms_5}\_Typ = boolean.$$

$$\forall Op_{omt_j} \in \beta(Robot) = \{Pr\,ogram\ Start, Program\ Reset, Pause, continue, Emergency\ Action\}$$
$$/\ j = 1..5 \quad alors$$
$$\| Trans(OMT, Op_{omt_i}) = Srve_{mms_i} \quad avec$$
$$Srve_{mms_i}\_Nme = Op_{omt_i}\_Nme$$
$$Srve_{mms_i}\_Typ = Op_{omt_i}\_Typ,$$

$Srve_{mms_1}\_Nme = Start$

$Srve_{mms_1}\_Typ = Result,$

$Srve_{mms_2}\_Nme = Reset,$

$Srve_{mms_2}\_Typ = Result,$

$Srve_{mms_3}\_Nme = Stop,$

$Srve_{mms_3}\_Typ = Result,$

$Srve_{mms_4}\_Nme = Resume,$

$Srve_{mms_4}\_Typ = Result,$

Puisque la classe Robot est reliée à la classe VMD par la relation généralisation-spécialisation, cette dernière est transformée en une contrainte de type et sera représentée comme suit :

$R_{gen-spec}$:

$$\| Trans(OMT, \gamma(\eta_{VMD,Robot}(VMD))) = \omega_{VMD,Robot}(\alpha(VMD))$$

$$\gamma(\eta_{VMD,Robot}(VMD)) = \{Safety\ Interlok\ Violated, Any\ Physical\ Resource\ Power\ On,$$
$$All\ Physical\ Resources\ Calibrated, Local\ Control, Metric\ Measure\ \}$$
$$= \omega_{VMD,Robot}(\alpha(VMD)).$$

Un nouvel attribut est rajouté à la classe VMD, ainsi qu'une contrainte concernant la classe Robot :

$$At_{mms}\_Nme = Class(Robot),$$
$$Const_1\_Nme = [Class = Robot].$$

Suivant la représentation MMS nous avons :

Object: VMD
Tous les attributs définis dans la norme MMS

Attribute: Class(Robot)
Constraint: Class = Robot
Attribute: Safety Interlocks Violated(True, False)
Attribute: Robot VMD State(READY, EXECUTING, MOTION-

PAUSED, MANUAL-INTERVENTION_REQUIRED)
Attribute: Any Physical Resource Power On(True, False)
Attribute: All Physical Resources Calibrated(True, False)
Attribute: Local Control(True, False)
Attribute: Reference to Selected Controlling Program

De nouveaux services vont enrichir l'ensemble des services déjà définis dans la norme MMS. L'un VMD-Stop comporte comme le service d'arrêt d'urgence de telle sorte que l'utilisateur MMS distant peut arrêter complètement le robot en le plaçant dans l'état Manual Intervention Required. L'autre, VMD-Reset, place le robot dans son état initial pour des opérations d'auto-diagnostic.

4.2.2. Règle appliquée à la classe Task Program

$$R_{class} :$$
$$\| Trans(OMT, Task\ Program_{omt}) = Task\ Program_{mms}$$

$$At_{mms}\_Nme = Error\ code \quad , \qquad At_{mms}\_Typ = integer.$$

Des spécificités relatives au programme du robot telles que : deux types de programmes existent, un appelé programme contrôlant (Controlling Program) qui contrôle le fonctionnement du robot comme une entité globale et l'autre appelé programme contrôlé (Controlled program). Ce dernier est normalement associé aux composants du Robot tels que le Bras (ARM) et un équipement périphérique (Auxiliary Device).

Par ce fait, le VMD-Robot autorise l'existence de deux programmes reliés par un bus interne de telle sorte que ce lien est transparent vis-à-vis de l'utilisateur MMS distant. Ceci induit la définition de nouveaux services au sein du VMD-Robot : SelectProgramInvocation et AlterProgramInvocation. Ces deux nouveaux services rejoignent VMD-Stop et VMD-Reset.

Puisque la classe Task Program est reliée à la classe Program Invocation par la relation généralisation-spécialisation, cette dernière est transformée en une contrainte de type et sera représentée comme suit :

$$R_{gen-spec} :$$
$$\| Trans(OMT, \gamma(\eta_{Program\ Invocation, Task\ Program}(Program\ Invocation))) =$$
$$\omega_{Program\ Invocation, Task\ Proram}(\alpha(Program\ Invocation))$$

$$\gamma(\eta_{Program\ Invocation, Task\ Program}(Program\ Invocation)) = \{Error\ Code\}$$
$$= \omega_{Program\ Invocation, Task\ Program}(\alpha(Program\ Invocation)).$$

Un nouvel attribut est rajouté à la classe VMD, ainsi qu'une contrainte concernant la classe Robot :

$$At_{mms}\_Nme = Class(Task\ Program),$$
$$Const_1\_Nme = [Class = Task\ Program].$$

Suivant la représentation MMS nous avons :

Object: Program Invocation
Tous les attributs définis dans la norme MMS

Attribute: Class(Task Program)
Constraint: Class = Task Program
Attribute: Error Code-Integer.

La classe Task Program est une généralisation de trois sous-classes : Normal Program, Controlling Program et Controlled Program. Ainsi, en appliquant la règle sur la relation de généralisation-spécialisation nous aurons les résultats suivants :

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Task\ Program, Normal\ Program}(Task\ Program))) = \omega_{Task\ Proram,\ Normal\ Program}(\alpha(Task\ Program))$$

$$\gamma(\eta_{Task\ Program, Normal\ Program}(Task\ Program)) = \{\phi\} = \omega_{Task\ Proram,\ Normal\ Program}(\alpha(Task\ Program)).$$

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Task\ Program, Controlling\ Program}(Task\ Program))) = \omega_{Task\ Proram,\ Controlling\ Program}(\alpha(Task\ Program))$$

$$\gamma(\eta_{Task\ Program, Controlling\ Program}(Task\ Program)) = \{ProgramLocation\} = \omega_{Task\ Proram,\ Controlling\ Program}(\alpha(Task\ Program)).$$

La classe Controlling Program est spécialisée en trois sous-classes : Free Run Program, Step Limited Program et Cycle Limited Program. Ainsi l'application de la règle sur la relation de généralisation-spécialisation donne le résultat suivant :

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Controlling\ Program, FreeRun}(Controlling\ Program))) = \omega_{Controlling\ Program, Free\ Run}(\alpha(Controlling\ Program))$$

$$\gamma(\eta_{Controlling\ Program, FreeRun}(Controlling\ Program)) = \{\phi\} = \omega_{Controlling\ Program, Free\ Run}(\alpha(Controlling\ Program)).$$

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Controlling\ Program,StepLimited}(Controlling\ Program))) = \omega_{Controlling\ Program,Step\ Limited}(\alpha(Controlling\ Program))$$

$$\gamma(\eta_{Controlling\ Program,StepLimited}(Controlling\ Program)) = \{Remaining\ Step\ Count\} = \omega_{Controlling\ Program,Step\ Limited}(\alpha(Controlling\ Program)).$$

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Controlling\ Program,CycleLimited}(Controlling\ Program))) = \omega_{Controlling\ Program,Cycle\ Limited}(\alpha(Controlling\ Program))$$

$$\gamma(\eta_{Controlling\ Program,CycleLimited}(Controlling\ Program)) = \{Remaining\ Cycle\ Count\} = \omega_{Controlling\ Program,Cycle\ Limited}(\alpha(Controlling\ Program)).$$

La classe Controlling Program est en relation avec la classe Controlled Program (relation multiple) :

R_{assoc} :

$$\|Trans(OMT, \nu_{Controlling\ Program,Controlled\ Program}(Controlling\ Program)) = \vartheta_{Controlling\ Program,Controlled\ Program}$$

Puisque la classe Controlling Program est associée, par l'intermédiaire d'une relation d'association multiple $\nu_{Controlling\ Program,Controlled\ Programs}$, avec la classe Controlled Program, alors $\vartheta_{Controlling\ Program,Controlled\ Program}$ modélise un attribut liste de références appartenant à la classe Controlling Program.

$$\exists AtLstRef_2 \in Controlling\ Program \quad / \quad AtLstRef_2\_Nme = List\ of\ Controlled\ Program.$$

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Task\ Program,Controlled\ Program}(Task\ Program))) = \omega_{Task\ Proram,\ Controlled\ Program}(\alpha(Task\ Program))$$

$$\gamma(\eta_{Task\ Program,Controlled\ Program}(Task\ Program)) = \{\phi\} = \omega_{Task\ Proram,\ Controlled\ Program}(\alpha(Task\ Program)).$$

Suivant la représentation MMS nous avons :

Object: Program Invocation

Tous les attributs définis dans la norme MMS

Attribute: Class(Task Program)

Constraint: Class = Task Program
Attribute: Error Code-Integer
Attribute : Class(Normal Program, Controlling Program, Controlled Program)
Constraint : Class=Normal Program
Attribute : Normal Program
Constraint : Class=Controlling
Attribute : Program Location – String
Attribute : Liste of References to Controlled Program Invocations
Attribute : Class (Free Run, Step Limited, Cycle Limited)
Constarint : Class= Step Limited
Attribute : Remaining Step Count – Integer
Constraint : Class = Cycle Limited
Attribute : Remaining Cycle Count – Integer
Constarint : Class=Controlled Program
Attribute : Reference to Controlling Program Invocation

4.2.3. Règle appliquée à la classe ARM

$$R_{class} :$$
$$\| Trans(OMT, ARM_{omt}) = ARM_{mms}$$

$$\forall At_{omt_i} \in \gamma(ARM_{omt}) = \{ Local\ Control, Device\ Power\ On, Device\ Calibrated, Number\ of\ Joints, BASE\text{-}WORLD \}$$
$$/\ i = 1...5 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = At_{mms_i} \quad avec$$
$$At_{mms_i}\_Nme = At_{omt_i}\_Nme$$
$$At_{mms_i}\_Typ = At_{omt_i}\_Typ,$$

$At_{mms_1}\_Nme = LocalControl$, $At_{mms_1}\_Typ = Boolean,$
$At_{mms_2}\_Nme = DevicePowerOn$, $At_{mms_2}\_Typ = Boolean,$
$At_{mms_3}\_Nme = DeviceCalibrated$, $At_{mms_3}\_Typ = Boolean,$
$At_{mms_4}\_Nme = Number\ of\ Joints,$ $At_{mms_4}\_Typ = Integer,$
$At_{mms_5}\_Nme = BASE - WORLD,$ $At_{mms_5}\_Typ = pose(\ type\ défini\ par\ l'\ utilisateur\ ).$

La classe ARM est une sous-classe de la classe Domain de MMS. Ainsi, ARM sera représentée par une classe Domain.

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{Domain,ARM}(Domain))) = \omega_{Doamin,ARM}(\alpha(Domain))$$

$$\gamma(\eta_{Domain,ARM}(Domaine)) = \{ Local\ Control, Device\ Power\ On, Device\ Calibrated, Number\ of\ Joints, BASE\text{-}WORLD \} = \omega_{Domain,ARM}(\alpha(Domain)).$$

Ainsi la représentation MMS est :

Object: Domain

Attribute : Class (ARM)
Constraint : Class=ARM
Attribute: Local Control (True, False)
Attribute: Device Power On (True, False)
Attribute : Device Calibrated (True, False)
Attribute : Number of Joints-Integer
Attribute : BASE-WORLD-pose

Puisque la classe ARM est composée, par l'intermédiaire d'une relation d'agrégation $\delta_{ARM,EndEffector}$, $\delta_{ARM,PathPlanner}$, $\delta_{ARM,Servomechanism}$ des classes : End Effector, Path Planner et Servomechanism. Ces relations sont transformées en relations simples $\vartheta_{ARM,EndEffector}$, $\vartheta_{ARM,PathPlanner}$, $\vartheta_{ARM,Servomechanism}$.

R_{assoc} :

$$\|Trans(OMT, \delta_{ARM,EndEffector}(ARM)) = \vartheta_{ARM,EndEffector}$$

R_{assoc} :

$$\|Trans(OMT, \delta_{ARM,PathPlanner}(ARM)) = \vartheta_{ARM,PathPlanner}$$

R_{assoc} :

$$\|Trans(OMT, \delta_{ARM,Servomechanism}(ARM)) = \vartheta_{ARM,Servomechanism}$$

$\vartheta_{ARM,EndEffector}$ modélise des attributs de références appartenant à la classe ARM.

$$\exists AtRef_1 \in ARM \quad / \quad AtRef_1\_Nme = End\ Effector.$$

$\vartheta_{ARM,PathPlanner}$ modélise des attributs de références appartenant à la classe ARM.

$$\exists AtRef_2 \in ARM \quad / \quad AtRef_2\_Nme = PathPlanner.$$

$\vartheta_{ARM,Servomechanism}$ modélise des attributs de références appartenant à la classe ARM.

$$\exists AtRef_3 \in ARM \quad / \quad AtRef_3\_Nme = Servomechanism.$$

Suivant la représentation MMS nous avons :

Object: Domain

Attribute : Class (ARM)
Constraint : Class=ARM
Attribute: Local Control (True, False)
Attribute: Device Power On (True, False)
Attribute : Device Calibrated (True, False)
Attribute : Number of Joints-Integer
Attribute : BASE-WORLD-pose
Attribute : End Effector
Attribute : ID Number
Attribute : Tool Descriptor
Attribute : TOOL-MICS -pose
Attribute : Path Planner
Attribute : USER-BASE -pose
Attribute : Desired TOOL-USER -pose
Attribute : Programmed Speed -Floating_point
Attribute : Acceleration Factor - Floating_point
Attribute : Programmed Acceleration - Floating_point
Attribute : Servomechanism
Attribute : MICS-BASE -pose
Attribute : Ordered List Of Joint Description
Attribute : Joint Type (REVOLUTE, PRISMATIC)
Attribute : Calibrated(CALIBRATED, NOTCALIBRATED, CALIBRATING)
Attribute : Joint Brakes (True, False)
Constraint : Joint Brakes = On
Attribute : Brakes On (True, False)
Attribute : Upper Bound -Floating_point
Attribute : Lower Bound -Floating_point
Attribute : Joint Servo
Attribute : Actual Joint Value -Floating_point
Attribute : Moving Enabled (True, False)

4.2.4. Règles appliquées à la classe Auxiliary Device

$$R_{class} :$$
$$\| Trans(OMT, Auxiliary\ Device_{omt}) = Auxiliary\ Device_{mms}$$

$$\forall At_{omt_i} \in \gamma(AuxiliaryDevice_{omt}) = \{ Local\ Control, Device\ Power\ On, Device\ Calibrated \ \}$$
$$/\ i = 1...3 \quad alors$$
$$\| Trans(OMT, At_{omt_i}) = At_{mms_i} \quad avec$$
$$At_{mms_i}\_Nme = At_{omt_i}\_Nme$$
$$At_{mms_i}\_Typ = At_{omt_i}\_Typ,$$

$$At_{mms_1}\_Nme = LocalControl \ , \qquad At_{mms_1}\_Typ = Boolean,$$
$$At_{mms_2}\_Nme = DevicePowerOn \ , \qquad At_{mms_2}\_Typ = Boolean,$$
$$At_{mms_3}\_Nme = DeviceCalibrated \ , \qquad At_{mms_3}\_Typ = Boolean.$$

La classe Auxiliary Device est une sous-classe de la classe Domain de MMS. Ainsi, Auxiliary Device sera représentée par une classe Domain.

$$R_{gen-spec} :$$
$$\| Trans(OMT, \gamma(\eta_{Domain, Auxiliarydevice}(Domain))) = \omega_{Doa\,min, AuxiliaryDevice}(\alpha(Domain))$$

$$\gamma(\eta_{AuxiliaryDevice}(Domaine)) = \{Local\ Control, device\ Power\ On, Device\ Calibrated \ \}$$
$$= \omega_{Domain, Auxiliarydevice}(\alpha(Domain)).$$

Ainsi la représentation MMS est :

Object: Domain

Attribute : Class (Auxiliary Device)
Constraint : Class=Auxiliary Device
Attribute: Local Control (True, False)
Attribute: Device Power On (True, False)
Attribute : Device Calibrated (True, False)

La classe Auxiliary Device est spécialisée en sept sous-classes : Sensor, Gripper, Vision System, Paint System, Safety interlock, Weld Controller et Water Jet Cutter System. Ainsi l'application de la règle sur la relation de généralisation-spécialisation donne le résultat suivant :

$$R_{gen-spec} :$$
$$\| Trans(OMT, \gamma(\eta_{AuxiliaryDevice, Sensor}(AuxiliaryDevice))) =$$
$$\omega_{AuxiliaryDevice, Sensor}(\alpha(Auxiliary\ Device))$$

$$\gamma(\eta_{AuxiliaryDevice, Sensor}(AuxiliaryDevice)) = \{\phi\}$$
$$= \omega_{AuxiliaryDevice, Sensor}(\alpha(Auxiliary\ Device)).$$

$R_{gen-spec}$:

$$\left\| Trans\left(OMT, \gamma\left(\eta_{AuxiliaryDevice,Gripper}\left(AuxiliaryDevice\right)\right)\right)\right\| = \omega_{AuxiliaryDevice,Gripper}\left(\alpha\left(Auxiliary\ Device\right)\right)$$

$$\gamma\left(\eta_{AuxiliaryDevice,Gripper}\left(AuxiliaryDevice\right)\right) = \{\phi\} = \omega_{AuxiliaryDevice,Gripper}\left(\alpha\left(Auxiliary\ Device\right)\right).$$

$R_{gen-spec}$:

$$\left\| Trans\left(OMT, \gamma\left(\eta_{AuxiliaryDevice,Pa\,int\,System}\left(AuxiliaryDevice\right)\right)\right)\right\| = \omega_{AuxiliaryDevice,Pa\,int\,System}\left(\alpha\left(Auxiliary\ Device\right)\right)$$

$$\gamma\left(\eta_{AuxiliaryDevice,Pa\,int\,System}\left(AuxiliaryDevice\right)\right) = \{\phi\} = \omega_{AuxiliaryDevice,Pa\,int\,System}\left(\alpha\left(Auxiliary\ Device\right)\right).$$

$R_{gen-spec}$:

$$\left\| Trans\left(OMT, \gamma\left(\eta_{AuxiliaryDevice,VisionSystem}\left(AuxiliaryDevice\right)\right)\right)\right\| = \omega_{AuxiliaryDevice,VisionSystem}\left(\alpha\left(Auxiliary\ Device\right)\right)$$

$$\gamma\left(\eta_{AuxiliaryDevice,VisionSystem}\left(AuxiliaryDevice\right)\right) = \{\phi\} = \omega_{AuxiliaryDevice,VisionSystem}\left(\alpha\left(Auxiliary\ Device\right)\right).$$

$R_{gen-spec}$:

$$\left\| Trans\left(OMT, \gamma\left(\eta_{AuxiliaryDevice,safetyInterlok}\left(AuxiliaryDevice\right)\right)\right)\right\| = \omega_{AuxiliaryDevice,SafetyInterlok}\left(\alpha\left(Auxiliary\ Device\right)\right)$$

$$\gamma\left(\eta_{AuxiliaryDevice,SafetyInterlok}\left(AuxiliaryDevice\right)\right) = \{\phi\} = \omega_{AuxiliaryDevice,SafetyInterlok}\left(\alpha\left(Auxiliary\ Device\right)\right).$$

$R_{gen-spec}$:

$$\left\| Trans\left(OMT, \gamma\left(\eta_{AuxiliaryDevice,WeldController}\left(AuxiliaryDevice\right)\right)\right)\right\| = \omega_{AuxiliaryDevice,WeldController}\left(\alpha\left(Auxiliary\ Device\right)\right)$$

$$\gamma\left(\eta_{AuxiliaryDevice,WeldController}\left(AuxiliaryDevice\right)\right) = \{\phi\} = \omega_{AuxiliaryDevice,WeldController}\left(\alpha\left(Auxiliary\ Device\right)\right).$$

$R_{gen-spec}$:

$$\|Trans(OMT, \gamma(\eta_{AuxiliaryDevice,WaterJetCutterSystem}(AuxiliaryDevice))) = \omega_{AuxiliaryDevice,WaterJetCutterSystem}(\alpha(Auxiliary\ Device))$$

$$\gamma(\eta_{AuxiliaryDevice,WaterJetCutterSystem}(AuxiliaryDevice)) = \{\phi\} = \omega_{AuxiliaryDevice,WaterJetCutterSystem}(\alpha(Auxiliary\ Device)).$$

Ainsi la représentation MMS est :

Object: Domain

Attribute : Class (Auxiliary Device)
Constraint : Class=Auxiliary Device
Attribute: Local Control (True, False)
Attribute: Device Power On (True, False)
Attribute : Device Calibrated (True, False)
Attribute : Class (Sensor, Gripper, Paint System, Vision System, Safety Interlock, Weld Controller, Water Jet Cutter System)
Constraint : Classe = Sensor
Attribute : Sensor
Constraint : Class = Gripper
Attribute : Gripper
Constraint : Class =Paint System
Attribute : Paint System
Constraint : Class=Vision System
Attribute : Vision System
Constraint : Class = Weld Controller
Attribute : Weld Controller
Constraint : Class = Water Jet Cutter System
Attribute : Water Jet Cutter System

4.3. Graphe d'états du système Robot et le VMD

La figure 4 montre le 'mapping' des opérations induisant les changements des états en services MMS.

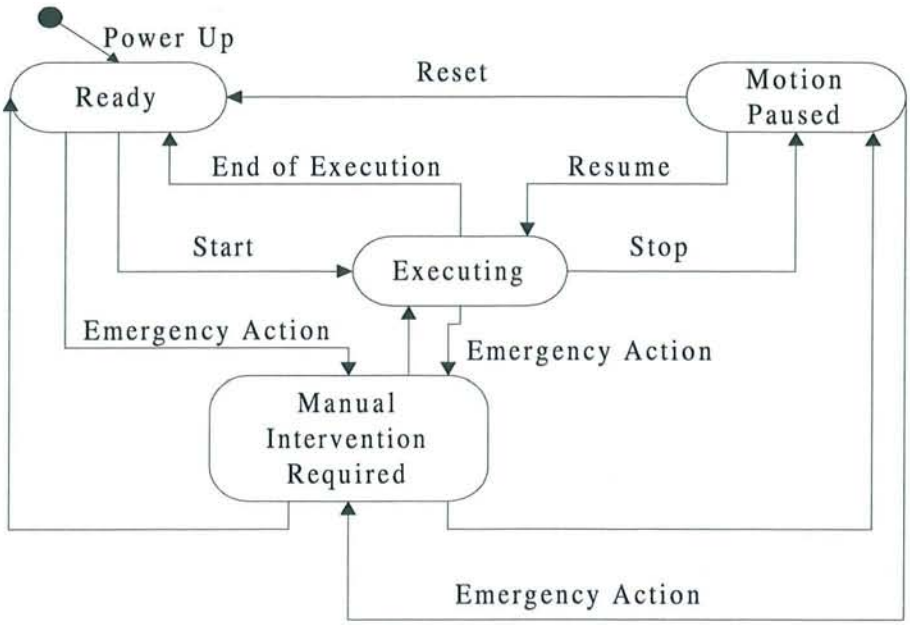

Figure 4. La Projection du graphe d'états du système Robot sur les services MMS.

4.4. Les objets normalisés spécifiques Robot

Des objets spécifiques au robot ont été définis : le nom normalisé attribué au domaine bras du robot est *R_ARM*. Si le robot est constitué de plusieurs bras, la syntaxe est alors *R_ARM_1*, *R_ARM_2*, ... Ainsi à chaque bras du robot est affecté un domaine. Ainsi que les équipements périphériques tels que Safety Interlock est représenté par un objet de la classe Domain et possède un nom normalisé *R_SAFE*. Un autre domaine existe *R_CAL*. Ce dernier est associé à la procédure de calibration des ressources physiques du Robot. Des objets normalisés représentent des objets MMS programmes invocation tels que : *R_ARM* associé au programme contrôlant le bras du robot (programme contrôlé), *R_CAL* représente la procédure de calibration d'une ressource physique du robot. Tous ces objets sont prédéfinis, c'est à dire ils ont l'attribut MMD Deletable égal à Faux.

Des objet types nommés (Named Type) seront normalisés. On peut en citer *R_PIS* (position dans l'espace), *R_OIS* (orientation dans l'espace), *R_EEF* (effecteur terminal), ...

Les attributs de la classe Robot et des autres classes définies en tant que domaines MMS, seront représentés par des variables nommées MMS avec des noms normalisés. La portée de ces variables est soit vmd-specific (pour les attributs de la classe Robot), soit domain-specific (pour les attributs des classes considérées comme des domaines MMS). Par exemple la variable *R_VWPR* est un objet MMS variable nommée représentant l'attribut Any Physical Resource Power On du VMD-Robot et aura une portée vmd-specific.

Ainsi, la représentation MMS du VMD-Robot est décrite dans le tableau 1.

Object: VMD
Tous les attributs définis dans la norme MMS

Attribute: List of Program Invocations(R_ARM, R_CAL, R_SAFE)
Attribute: List of Domains(R_ARM, R_CAL, R_SAFE)
Attribute: Class(Robot)
Constraint : Class = Robot
Attribute: Safety Interlock Violated(True, False)
Attribute: Robot VMD State(ROBOT-IDLE, ROBOT-LOADED, ROBOT-READY, ROBOT-EXECUTING, ROBOT-PAUSED, MANUAL-INTERVENTION_REQUIRED)
Attribute: Any Physical Resource Power On(True, False)
Attribute: All Physical Resources Calibrated(True, False)
Attribute: Local Control(True, False)
Attribute: Reference to Selected Controlling Program

Tableau 1. La représentation MMS du VMD-Robot.

Un objet Sémaphore est normalisé et appelé *R_CTRL*. Ce dernier est utilisé pour représenter un sémaphore qui maintient le contrôle des ressources du bras 'ARM'. Il existe dans une configuration Robot-Serveur et plusieurs Clients. Dans une telle configuration, le contrôle est exclusif à un seul client. C'est un objet Tocken Semaphore avec un seul jeton.

Des noms normalisés concernant les objets MMS conditions événementielles sont définis. *R_RVS* identifie une condition événementielle scrutée de l'attribut Robot VMD State en quittant l'état EXECUTING à un autre. *R_SIV* identifie une condition événementielle scrutée quand l'attribut Safety Interlocks Violated change sa valeur de False à True. *R_RLC* identifie une condition événementielle scrutée quand l'attribut Local Control change de valeur de False à True. Quant au *R_ARM*, il identifie une condition événementielle scrutée quand le programme *R_ARM* (programme contrôlé) quitte l'état 'Running'.

En ce qui concerne les actions événementielles exécutées à la suite des occurrences des événements, deux noms normalisés sont définis. *R_STC* et *R_ARM*. Le premier identifie l'action à réaliser quand un des événements précédemment nommés se produit et il générera une notification de l'événement (Event Notification) avec comme paramètres le résultat du service Status. L'autre identifie l'action à effectuer quand le programme du bras du robot quitte l'état 'Running' et générera une notification de l'événement accompagnée du résultat du service GetProgramInvocationAttribute.

L'ensemble des objets et services implémentés sur le robot constitue sa classe de conformité. Sachant que la configuration adoptée c'est : un Client , un Serveur (Figure 5), tel que le Robot joue le rôle d'un serveur. La norme d'accompagnement ne définit pas la configuration dans laquelle le robot joue le rôle d'un client. Ceci se justifie par l'attente de la normalisation des équipements avec lesquels il coopère. Pour une première configuration, la liste de services que l'on vient de présenter est complétée par des services de téléchargement de domaines.

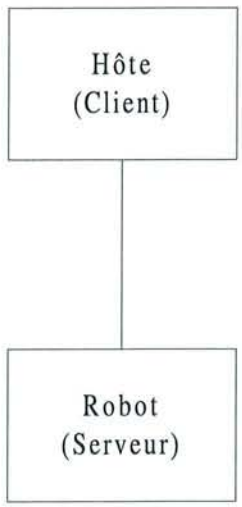

Figure 5. La configuration Robot Serveur – un seul Client.

Université HENRI POINCARÉ NANCY 1

FACULTÉ DES SCIENCES

Madame **ARAB – MANSOUR Ikbal**

DOCTORAT de l'UNIVERSITE HENRI POINCARE, NANCY-I

en INFORMATIQUE INDUSTRIELLE

VU, APPROUVÉ ET PERMIS D'IMPRIMER

Nancy, le 28 DÉC 1998 n° 168

Le Président de l'Université

UNIVERSITÉ HENRI POINCARÉ, NANCY 1 LE PRÉSIDENT

J.P. FINANCE

3C-F9

Résumé :

Cette thèse, effectuée au CRAN (Centre de Recherche en Automatique de Nancy) porte sur la conception et l'intégration des communications des systèmes automatisés distribués. La problématique est de concevoir un système de communication industrielle cohérent avec le système d'information de l'entreprise. On s'appuie sur les méthodes de conception orientées objet des systèmes d'information et on utilise la messagerie industrielle MMS (Manufacturing Message Specification). Des travaux antérieurs menés au CRAN ont permis d'intégrer la partie informationnelle, tandis que des travaux effectués dans d'autres laboratoires ont débouché sur l'intégration de la partie fonctionnelle. Afin d'aboutir à une intégration globale nous proposons une approche reposant sur la méthode OMT (Object Modeling Technique). Nous prenons comme base les travaux menés au laboratoire LIASI (Laboratoire) qui ont abouti à la définition d'un modèle générique pour la conception orientée objet des systèmes d'information (MGCO2). Nous utilisons leur formalisme pour représenter OMT et MMS. Ensuite, nous définissons des règles de transformation pour le passage d'une représentation à l'autre. La méthode ne peut pas être mise en œuvre sans définir des règles tenant en compte du contexte d'implémentation. Cette ingénierie de la méthode propose une dérivation des modèles OMT en objets et services de communication prédéfinis dans la norme MMS réellement disponibles sur les différents sites (classe de conformité). Nous validons notre approche et la faisabilité de la démarche proposée, sur une application de télé-pilotage d'un robot industriel avec retour vidéo pour laquelle sont générés les objets et services MMS nécessaires.

Mots clés : Système d'Information ,OMT (Object Modelling Technique), Objet, Dynamique, Systèmes Automatisés Distribués de Production, MMS (Manufacturing Message Specification).

Abstract :

This thesis realised at CRAN (Research Center for Automatic Control of Nancy) laboratory relates to the design and the integration of the communications of distributed automated systems. The issue is to conceive an industrial communication system consistent with the information system of the process. Our work is based on the object-oriented design methods of information systems and uses the industrial message protocol MMS (Manufacturing Message Specification). Prior works elaborated at CRAN made it possible to integrate the informational part, while the work carried out in other laboratories led to the integration of the functional part. In order to lead to a total integration we propose an approach based on OMT (Object Modelling Technique) method. We take as bases the work undertaken at LIASI (Laboratory) laboratory, which led to the definition of a generic model for the object-oriented design of the information systems (MGCO2). We use their formalism to represent OMT and MMS. Then, we define a set of transformation rules for the translation of one representation into the other. The method cannot be implemented without defining rules that take into account the implementation context. This engineering of the method proposes a derivation of OMT models into communication object and services predefined in MMS standard that are really available on the various sites (conformance classes). We validate our approach and the feasibility of the suggested step, by realising a remote-control application of an industrial robot with a video feedback for which the MMS objects and services are generated.

Keywords : Information System, OMT (Object Modelling Technique), Object, Dynamic, Distributed Automated Production Systems, MMS (Manufacturing Message Specification).