Titre: Title:	Etude des systèmes multi-agents pour la recherche de solutions d'assemblage
Auteur: Author:	Bernard DeGuire
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	DeGuire, B. (1997). Etude des systèmes multi-agents pour la recherche de solutions d'assemblage [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6711/

UNIVERSITÉ DE MONTRÉAL

ÉTUDE DES SYSTÈMES MULTI-AGENTS POUR LA RECHERCHE DE SOLUTIONS D'ASSEMBLAGE

BERNARD DeGUIRE
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE MÉCANIQUE)
AOÛT 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

UTILISATION D'UN SYSTÈME MULTI-AGENTS
POUR LA RECHERCHE DE SOLUTIONS D'ASSEMBLAGE

présenté par : DeGUIRE Bernard

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. CLOUTIER Guy, doctorat, président-rapporteur

M. MASCLE Christian, doctorat, membre et directeur de recherche

M. FORTIN Clément, Ph. D., membre et codirecteur de recherche

M. HOUMMADY Abdellah, Ph. D., membre

À ma fiancée Mélanie

et à toute ma famille

pour leurs encouragements continuels.

REMERCIEMENTS

J'aimerais premièrement remercier M. Abdellah Hoummady ainsi qu'Alliance Commerciale Technologique (ACT), qui ont su me donner un projet à la fine pointe de la technologie. La confiance qu'ils ont mise en moi m'a permis d'atteindre de hauts sommets.

J'aimerais également remercier MM. Christian Mascle et Clément Fortin, qui ont su me guider, même dans des chemins très difficiles.

J'aimerais aussi remercier le CRSNG pour leur support financier. Sans eux, cette maîtrise n'aurait pu avoir lieu. J'aimerais également féliciter leur initiative d'encourager la recherche dans les milieux industriels, grâce à leur programme à Incidence industrielle.

Merci aussi à toute ma famille, mes parents et ma sœur, qui m'ont encouragé tout au long de ce travail.

J'aimerais finalement remercier ma très chère Mélanie, qui, au travers de mes moments difficiles, a su être là pour moi. Je te dédie ce travail.

RÉSUMÉ

Le but de ce travail est d'étudier dans un premier temps le catalogue d'assemblage exposé par les Allemands Warnecke, Lörh et Kiener et encore inconnu en Amérique du Nord. Dans un deuxième temps, il consiste à étudier la viabilité de l'utilisation d'un système multi-agents pour la recherche de solutions dans un catalogue informatisé. Enfin, il discute des moyens d'intégrer ce système multi-agents à un système CFAO. Ce catalogue s'adresse aux gammistes lors de leur étape de sélection de la méthode d'assemblage de composants, et permet de conserver le savoir-faire de l'entreprise.

Pour permettre une recherche plus efficace des solutions d'assemblage, un nouveau concept de signature d'assemblage est présenté. Ce concept, que nous proposons dans ce présent mémoire, s'adapte surtout aux insertions et permet de filtrer les solutions choisies. Or, au stade actuel, ce concept ne s'adapte qu'aux contacts surfaciques et ne permet pas de reconnaître les degrés de libertés bloqués entre les éléments en contact.

L'étude propose également une manière efficace de structurer l'information dans le catalogue ainsi qu'une approche permettant au gammiste d'effectuer ses requêtes.

La structure multi-agents proposée est composée d'agents mobiles et asynchrones. Ce sont de petits programmes indépendants qui communiquent entre eux et ils permettent une recherche efficace de solutions tout en offrant les avantages d'une programmation modulaire. Cette programmation est en langage Java, puisqu'il s'agit d'une programmation par objets et qu'il peut s'exécuter sur une grande variété de plates-formes. Cependant, sans algorithme d'intelligence artificielle, le système multi-agents ne peut se vanter de posséder de « l'intelligence ».

ABSTRACT

The purpose of this work is threefold: firstly to study the concept of an assembly solution catalog, developed in Germany by Warnecke, Lörh and Keiner and still unknown in North America. Secondly, this work studies the viability of a multi-agent system incorporated within a computerized version of this catalog. Finally, ways of integrating this technology within a CAD/CAM product are discussed. This kind of catalog is designed to be used by assembly personnel and has a purpose of conserving the know-how of a company.

In order to obtain a more efficient result, a concept of assembly signature is added to this catalog. This innovative idea is mainly targeted towards insertions and eases the filtering of probable solutions. Unfortunately, this method only works for surface contacts and it is unable to determine the degrees of freedom between the assembled elements.

This study also proposes an effective way to structure the information within the catalog, and also studies the different steps the assembly personnel must take in order to select possible solutions.

The multi-agent structure proposed is designed with mobile and asynchronous agents, which are small independent programs able to between each other. They offer a powerful way to search for information while having all of the advantages of a modular approach. The Java programming language is proposed for this task since it is object-oriented and it can be executed on a wide range of platforms. Unfortunately, without any artificial intelligence algorithms, this multi-agent system does not really possess "intelligent" agents.

TABLE DES MATIÈRES

LISTE DES ANNEXES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

BD	Base de données
BDOO	Base de données orientée objet
BDR	Base de données relationnelle
c_n	Vérité atomique d'un agent n
C_n	Espace de connaissances d'un agent n
CATIA	Computer Aided Three dimentionnal Interactive Application
CDM	CATIA Data Management
CFAO	Conception et fabrication assistée par ordinateur
E	Environnement des agents
f	Face particulière pour le calcul de la signature d'assemblage
GBD	Gestionnaire de base de données
i_n	Intention atomique d'un agent n
I_n	Intention globale de l'agent n
IA	Intelligence artificielle
IAD	Intelligence artificielle distribuée
IS	Ingénierie simultanée
JDBC	Java Database Connectivity
n	Dimension de la signature d'assemblage
p	Pièce particulière pour le calcul de la signature d'assemblage

s	Signature d'assemblage
T_c	Temps de cycle contrat
T_t	Temps de cycle technique
v	Variable arbitraire pour le calcul de la signature d'assemblage

INTRODUCTION

Nul ne pouvait prédire un essor aussi important de l'informatique. Un ancien dirigeant d'IBM, ce géant américain de l'informatique, prévoyait aux environs des années '50 que le monde entier aurait au maximum une dizaine d'ordinateurs dans le futur. Jamais n'aurait-il pu être si loin de la réalité...

L'ordinateur est devenu pour l'ingénieur aussi indispensable que le scalpel pour le chirurgien : simulation de comportement d'un pont lors de tremblements de terre, validation de résultats expérimentaux variés, prévision de l'injection d'un plastique dans un moule, analyse de la résistance d'une structure aéronautique, programmation d'une machine à commandes numériques, etc.

À ses débuts, l'ordinateur facilitait la manipulation d'une grande quantité de nombres. La vaste majorité des opérations arithmétiques et trigonométriques étaient implantées correctement, ce qui accordait un gain de temps appréciable pour certains calculs simples et répétitifs. Avec le temps, l'interface graphique acquérait une impressionnante flexibilité et permettait l'apparition d'une vaste gamme d'applications graphiques, dont plusieurs pour résoudre des problèmes d'ingénierie. Parmi ces applications se trouvent les mailleurs et solveurs d'éléments finis, et les logiciels de conception et de fabrication assistées par ordinateur (CFAO).

Cependant, certaines activités liées à la conception sont difficilement résolues par ces logiciels. Nous pouvons noter toutes les étapes retrouvées au début du processus de conception, telles les idées et les esquisses. Également, d'autres activités telles la

recherche de solutions d'un problème d'assemblage nécessitant une certaine prise de décision n'ont pas encore été implantées efficacement à la suite de plusieurs difficultés. Par exemple, les solutions à deux problèmes d'assemblage similaires peuvent varier énormément dues à de légères différences, telles les matériaux utilisés ou une géométrie empêchant une insertion sans collision.

La solution des problèmes d'assemblage pose effectivement une grande difficulté. Le succès d'une entreprise repose sur l'expertise de quelques employés possédant beaucoup d'informations. Ainsi, lors du départ d'un de ces employés, l'expérience perdue entraîne des difficultés d'adaptation pour les remplaçants, ralentissant le processus de développement d'un produit. Il est évident que l'existence d'un logiciel qui permettrait de sauvegarder ce savoir, lequel pourrait inclure les solutions d'assemblage, serait fort utile dans ce domaine.

Lors de la révolution informatique expliquée précédemment, plusieurs approches de programmation ont vu le jour. La première, la programmation procédurale, consiste à programmer d'une manière relativement séquentielle. La programmation par objets a évolué par la suite, en facilitant la réutilisation de code programmé tout en permettant une hiérarchisation des informations. La plus récente programmation consiste en l'utilisation d'entités individuelles appelées « agents ». Cette technologie semble avoir un grand potentiel en ce qui concerne la conservation du savoir.

Ce projet consiste donc en l'étude de la technologie multi-agents pour valider son utilisation comme fondation dans un système offrant des solutions à des problèmes d'assemblage, à travers un catalogue de solutions d'assemblage. Ce catalogue favorisera

l'utilisation de solutions homogènes à des problèmes d'assemblage similaires en proposant diverses solutions aptes à résoudre le problème posé.

Ce document couvre quatre thèmes majeurs. Le **Chapitre 1** expose plus précisément la problématique. Il explique le processus actuel du choix d'une solution d'assemblage puis le problème relié à la conservation du savoir-faire. Il montre également la méthodologie utilisée ici pour étudier le problème de recherche de solutions d'assemblage.

Le **Chapitre 2** montre l'état de l'art dans les domaines utiles à cette étude, dont la conception de produits et d'assemblages, les systèmes de CFAO actuels, les bases de données et l'intelligence artificielle.

Par la suite, le **Chapitre 3** explique les méthodes proposées pour résoudre ce problème : une nouvelle procédure à suivre lors de la recherche d'une solution d'assemblage est montrée, les différentes caractéristiques nécessaires pour bien identifier les solutions d'assemblage sont expliquées, l'utilisation des systèmes multi-agents est décrite en détail, la structure de cette information au travers des différents agents est exposée et, finalement, nous effectuons une analyse des résultats et des limites du système proposé.

Enfin, la conclusion termine le document en mettant en évidence les points importants retenus.

CHAPITRE 1. PROBLÉMATIQUE

1.1 INTRODUCTION

Ce chapitre consiste à analyser la problématique actuelle. En effet, les raisons qui poussent à aider la recherche de solutions d'assemblage sont nombreuses. Ainsi, ce chapitre est divisé de la manière suivante : premièrement, les problèmes entourant la conservation du savoir-faire dans les compagnies sont discutés ; deuxièmement, le processus actuel pour choisir une solution d'assemblage est analysé ; finalement, la méthode utilisée pour résoudre ces problèmes est énoncée.

1.2 CONSERVATION DU SAVOIR-FAIRE

Il n'y a pas si longtemps, une personne qui s'intégrait dans une compagnie devenait membre d'une petite communauté et son expertise s'accumulait à la suite d'années d'expérience dans un même domaine. Or, il n'est pas rare de voir de nos jours des gens changer d'emploi à tous les cinq ans. Dans le domaine de l'ingénierie, les ingénieurs changent d'employeur à tous les 8 ans en moyenne (selon une enquête de 1997 de l'Ordre des ingénieurs du Québec), mais peuvent sans doute, pour un même employeur, gravir des échelons internes. Ce phénomène social provient entre autre du fait que les gens tentent d'atteindre les sommets en sautant d'un emploi à l'autre. Or, l'expertise demandée diffère d'un emploi à l'autre et un départ signifie habituellement l'embauche d'une ressource humaine supplémentaire ou le déplacement d'une ancienne ressource d'une tâche à une

autre. Il est rare de voir l'expérience du remplaçant épouser parfaitement les critères des nouveaux défis à surmonter.

Quelle que soit la solution préconisée pour pallier le départ d'une ressource, le temps d'adaptation de la nouvelle personne à sa récente tâche peut être long. Comprendre les objectifs de la tâche, connaître les intervenants, déterminer l'emplacement des différentes informations utiles à la tâche – bref, acquérir l'expérience nécessaire à la tâche – peuvent nécessiter beaucoup de temps.

Cependant, même en conservant son poste, les connaissances d'une personne peuvent ne plus être à jour. Le domaine de l'informatique présente un exemple où le savoir-faire peut vite devenir désuet. Du simple traitement de texte au plus complexe logiciel de modélisation tridimensionnelle, toute compagnie voulant demeurer compétitive doit s'adapter à la vague d'informatisation. Mais puisque tout nouveau produit entraîne une certaine période d'adaptation et d'apprentissage, la sortie continuelle de ces produits induit de fréquentes mises à jour.

L'informatique pose un autre problème relié à la quantité d'information qui augmente à un rythme effarant. L'explosion de l'Internet en est l'exemple le plus frappant. Les usagers de ce réseau peuvent accéder à des téraoctets d'information et à des milliers de nouveaux produits annuellement – et par conséquent, à de nouveaux standards – mais la lecture et la compréhension de toute cette information n'est pas humainement possible. Ainsi, bien que la personne maîtrise adéquatement les logiciels reliés à sa tâche, son savoir-faire s'altère puisqu'elle ne peut se tenir à jour face à cette information croissante.

Il existe des problèmes importants pour gérer l'information, en particulier dans le domaine de l'assemblage, puisqu'il n'existe aucun standard et que les solutions sont principalement basées sur la subjectivité et l'expérience du gammiste, la personne déterminant les gammes d'assemblage[1] d'un produit. Ainsi, s'il y a départ d'une ressource d'expérience, des délais importants peuvent survenir, impliquant des coûts plus élevés que prévus.

Plusieurs recherches informatiques tentent de pallier à ce problème de manque d'expérience. Entre autres, le concept de bases de connaissances a fait son apparition. Ce sont des systèmes de structure simple fournissant des solutions dès l'apparition de certaines conditions grâce à une logique « Si... Alors... Sinon... ». Également, les méthodes de logique floue et de systèmes neuronaux sont apparues. Elles permettent de résoudre certains types spécifiques de problèmes, mais leur utilisation est surtout à un niveau expérimental. Seul un nombre restreint d'applications industrielles particulières existent réellement, surtout en médecine et en analyse d'images.

Plus récemment, le concept de systèmes multi-agents tend à être la voie du futur de la conservation de l'expertise (Lashkari, Metral et Maes, 1993 ; Hale *et al.* , N/D ; Olsen *et al.* , 1994 ; Gómez-Pérez, 1994). Il s'agit principalement de plusieurs petits programmes indépendants, chacun possédant ses capacités et son expertise, pouvant interagir dynamiquement entre eux, et permettant ainsi de résoudre des problèmes complexes. Grâce à cette technique, l'expertise sur un sujet peut être transmise aisément. Certains

---

[1] Une gamme d'assemblage est une suite chronologique d'opérations d'assemblage nécessaires à l'obtention d'un produit fini assemblé, incluant les directions d'assemblage et d'autres informations technologiques.

logiciels « intelligents » ont déjà vu le jour (Firefly, 1996). Ils permettent d'automatiser des tâches en consultant les habitudes de l'utilisateur ou de ses confrères. Rendu à un certain seuil d'automatisation, une décision sera prise soit par confirmation de l'utilisateur, soit sans confirmation.

Une des difficultés de la conservation du savoir-faire dans le domaine de l'assemblage est qu'il n'existe pas de méthode pour isoler une solution géométrique, i.e. une solution pouvant être déterminée grâce à l'agencement physique entre les composants. Nous tenterons dans ce travail de montrer une nouvelle approche – appelée *signature d'assemblage* – qui permettra, grâce à un modèle géométrique de l'assemblage, de retrouver les solutions pouvant résoudre un problème.

Quelle que soit la technologie utilisée pour conserver le savoir-faire, les compagnies ne possèdent pas d'outil efficace pour le faire, surtout dans le domaine des solutions l'assemblage. Ce travail tentera de montrer une approche de conservation du savoir-faire dans ce domaine en utilisant un système multi-agents.

1.3 PROCESSUS ACTUEL DE CHOIX D'UNE SOLUTION D'ASSEMBLAGE

Dans le processus de conception d'un produit, l'étape d'assemblage a longtemps été vue comme non prioritaire. Normalement, la conception était faite indépendamment de l'assemblage et les gammistes devaient tenter de trouver une solution d'assemblage pour un produit quelquefois mal conçu. Ainsi, ne possédant pas la technologie pour assembler efficacement le produit, le bureau de conception devait concevoir à nouveau certains de ses aspects.

Or, plusieurs études démontrent que le coût d'une modification, ayant lieu en cours de développement d'un produit, augmente énormément si le cycle de conception est avancé. Ainsi, l'étape d'assemblage étant l'une des dernières du processus de conception d'un produit, le coût associé à une modification tardive du design est très élevé.

La recherche d'une solution d'assemblage est pour sa part très complexe. En effet, le domaine de l'assemblage est un très vaste domaine car il s'agit à la base d'un concept simple mais permettant un très large choix de possibilités. Rappelons qu'assembler consiste à unir physiquement deux ou plusieurs composants. Or, selon les matériaux en présence, les coûts permis et la qualité requise, la solution peut varier entre le soudage, le serrage, le vissage, le sertissage, etc. De plus, à l'intérieur d'une même famille de solutions, énormément de possibilités sont offertes. Aussi faisons-nous face à une croissance exponentielle des solutions possibles dû à ce grand nombre de possibilités.

Il est évident que l'efficacité de la recherche d'une solution d'assemblage pour un gammiste varie en fonction de son expérience. Ainsi, dans certains cas, l'étape de remise en cause de la conception du produit aurait pu être évitée s'il existait une méthode efficace de choisir les solutions d'assemblage. Pour résoudre ce problème, certaines recherches ont tenté de réduire les retours à la conception en offrant aux concepteurs un outil permettant de garder en perspective l'assemblage (Boothroyd et Dewhurst, 1983).

Les gammistes possèdent certains outils particuliers pour les aider dans leurs tâches. Quelques recherches ont permis de déterminer l'ordre dans lequel doivent s'effectuer certaines opérations d'assemblage. Une de ces méthodes présentement utilisées pour déterminer la gamme d'assemblage consiste à procéder par désassemblage pour partir

d'un produit fini pour ensuite en inverser les étapes afin de trouver la gamme. Une fois cette gamme trouvée, la manière utilisée pour assembler les composants est établie normalement par expérience.

Or, aucun outil ne peut véritablement servir à trouver une manière d'effectuer l'opération d'assemblage proprement dite. En effet, diverses options s'offrent au gammiste lors de sa prise de décision : l'assemblage peut être automatique, semi-automatique ou manuel; la fixation peut s'effectuer par collage ou par soudage; l'insertion peut s'accomplir horizontalement ou verticalement ; etc. Normalement, les gammistes utilisent leur expérience pour déterminer la machinerie utilisée et les conditions particulières à respecter. Cependant, cette méthode manque de rigueur et les résultats obtenus peuvent varier selon le gammiste ou le moment auquel la décision a été prise. Ainsi, dans le cas où le gammiste a peu d'expérience, plusieurs conséquences peuvent survenir : le taux de retour du produit à une nouvelle conception peut être plus élevé, l'assemblage peut être de moins bonne qualité (i.e. moins résistant), moins efficace ou plus coûteux. Par exemple, un gammiste sans expérience pourrait choisir une solution d'assemblage mal adaptée à un problème tel l'union permanente de deux pièces en plastique. Doit-il choisir la soudure par ultrasons ou la colle?

Dans le but de pallier à ce problème, un groupe de recherche, localisé en Allemagne, a travaillé sur un catalogue de solutions d'assemblage (Warnecke, Lörh et Kiener, 1980). Il s'agit d'un document proposant plusieurs solutions techniques concernant l'assemblage de composants. Le gammiste connaît donc le principe de la solution, certaines contraintes à respecter, ainsi que certaines limites techniques. Il peut alors prendre les composants (ou

sous-assemblages) deux par deux et assembler le produit. Il peut également visualiser l'assemblage grâce à une esquisse lui permettant d'évaluer sa faisabilité. Connaissant la méthode d'assemblage préconisée, il peut vérifier la disponibilité des différentes machines nécessaires pour sa réalisation. Ce document est malheureusement présenté uniquement sous format papier, ce qui rend sa consultation pénible, surtout si le nombre de solutions d'assemblage est élevé.

Ainsi, comment faire pour aider les gammistes à connaître rapidement les différents procédés connus et maîtrisés par la compagnie ? Comment savoir si l'outillage est présentement disponible ou non, et par conséquent quels sont les outils alternatifs ? Comment connaître les limites de l'outillage ? Comment déterminer les étapes préalables à effectuer pour une solution d'assemblage donnée ? Quels sont les avantages et les limites d'un certain type d'assemblage ? Comment aider un gammiste à s'y retrouver dans cette montagne d'information ?

Même si quelques résultats de recherches dans le domaine de l'assemblage sont intéressants, il en ressort que la décision est encore prise avec l'aide de l'expérience de la personne et que les outils pertinents d'aide à la décision pour les gammistes sont rares. Nous tenterons dans ce travail de fournir aux gammistes un outil efficace et rapide pour le choix de solutions d'assemblage.

1.4 MÉTHODE UTILISÉE

Cette section présente la méthode utilisée pour résoudre un problème d'assemblage. Puisque ce travail consiste en une étude de plusieurs éléments distincts et de leur intégration dans la recherche de solutions d'assemblage, aucun prototype n'est présenté.

À date, la seule manière d'intégrer des solutions d'assemblage au travail des gammistes consiste à utiliser des catalogues de solutions d'assemblage. Le catalogue de solutions d'assemblage présenté par Warnecke (Warnecke, Lörh et Kiener, 1980) est utilisé comme fondation du catalogue présenté dans ce projet. Cette manière de résoudre un problème d'assemblage – par détermination de la meilleure solution pour chacune des sous-étapes d'une solution d'assemblage – est très peu utilisée en Amérique du Nord. Il s'agit d'une approche d'origine principalement allemande. Seuls des outils de rédaction de documentation d'assemblage sont activement utilisés par nos industries nord-américaines. Contrairement au catalogue d'assemblage, cette dernière approche n'offre aucune manière de conservation des solutions d'assemblage propres à une compagnie. Actuellement, lorsqu'un assemblage nécessite de l'outillage spécialisé, le gammiste doit se fier à sa mémoire et à son expérience pour déterminer si un cas semblable a déjà été rencontré. Il peut donc arriver qu'un outillage pouvant solutionner deux cas très similaires espacés dans le temps soit oublié, induisant des coûts et des délais inutiles de fabrication d'un autre outil.

En complément à l'étude du catalogue, une analyse des logiciels de CFAO est nécessaire. Étant donné son utilisation de plus en plus présente dans l'industrie, une étude d'intégration d'un catalogue de solutions d'assemblage est intéressante. En effet,

plusieurs logiciels permettent maintenant de concevoir les produits en trois dimensions et de leur attribuer des caractéristiques en vue de leur assemblage. Ainsi, l'extraction d'information de la CFAO est essentielle afin d'éviter la répétition du travail nécessaire. Il serait donc possible de conserver un certain savoir-faire de la compagnie en matière d'assemblage et de permettre son accès aux gammistes et aux concepteurs grâce à un logiciel commun.

L'étude des systèmes multi-agents suit cette partie. Les systèmes multi-agents proviennent d'un nouveau paradigme de programmation consistant à utiliser une société d'agents pour résoudre des problèmes complexes, chaque agent possédant sa propre expertise. L'état de l'art sur cette nouvelle technologie est analysé. Le potentiel de ces agents est examiné afin de trouver leurs caractéristiques intéressantes, ainsi que leurs forces et leurs faiblesses. La possibilité d'intégration de certains algorithmes d'intelligence artificielle est également analysée. Ensuite, leur structure en société est décrite, et les différents langages de communication sont évalués.

Finalement, l'étude de la résolution d'un problème d'assemblage grâce aux systèmes multi-agents est effectuée. Une méthode géométrique de filtrage de solutions est analysée, et les résultats sont validés grâce à certains exemples dans un domaine précis de l'assemblage. En dernier lieu, la conception d'un prototype est discutée, en incluant le langage de programmation suggéré ainsi que les différentes interactions nécessaires.

Le système analysé devra permettre aux gammistes de :

1. choisir la meilleure solution d'assemblage lorsque plusieurs solutions s'offrent à lui ;

2. choisir adéquatement tous les outils nécessaires pour le type d'assemblage traité ;

3. trouver toutes les opérations préalables nécessaires à réaliser afin de permettre l'assemblage choisi ;

4. fournir, si elles existent, les opérations élémentaires que doivent effectuer les assembleurs sur le plancher.

CHAPITRE 2. ÉTAT DE L'ART

2.1 INTRODUCTION

Ce chapitre consiste à analyser l'état de l'art dans les domaines de l'assemblage, des systèmes CFAO, des bases de données et de l'intelligence artificielle. Dans certains cas, une appréciation du concept étudié est donnée en vue de sa possibilité d'intégration dans le projet actuel.

2.2 CONCEPTION DE PRODUITS ET D'ASSEMBLAGES

La conception de produits et d'assemblages est un vaste domaine. Pour cette raison, ce sous-chapitre se décompose en plusieurs sections. Premièrement, le cycle de développement d'un produit est analysé afin de déterminer les moments durant lesquels le gammiste doit prendre des décisions concernant le problème d'assemblage posé. Par la suite, le système d'assemblage est décomposé dans ses principaux composants afin de mieux le comprendre et l'apprécier. Ensuite, le principe de l'ingénierie simultanée est énoncé pour voir comment doivent se comporter les différents intervenants d'une compagnie. Puis, différentes méthodes développées en vue d'améliorer l'assemblage sont énoncées. Après, le concept de catalogue de solutions d'assemblage est montré. Suivent les méthodes utilisées pour déterminer la gamme d'assemblage. L'analyse du concept des caractéristiques de formes clôture cette section.

2.2.1 Cycle de développement d'un produit

Le cycle de développement d'un produit se présente principalement comme regroupant les étapes de la production du produit, de sa négociation et de son utilisation (Grenier, 1989). La production peut elle-même se décomposer en plusieurs sous-sections qui sont habituellement exécutées dans un ordre général : l'idée, les concepts, les spécifications, les dessins, les prototypes, et les modèles de présérie et de série (figure 2.1).

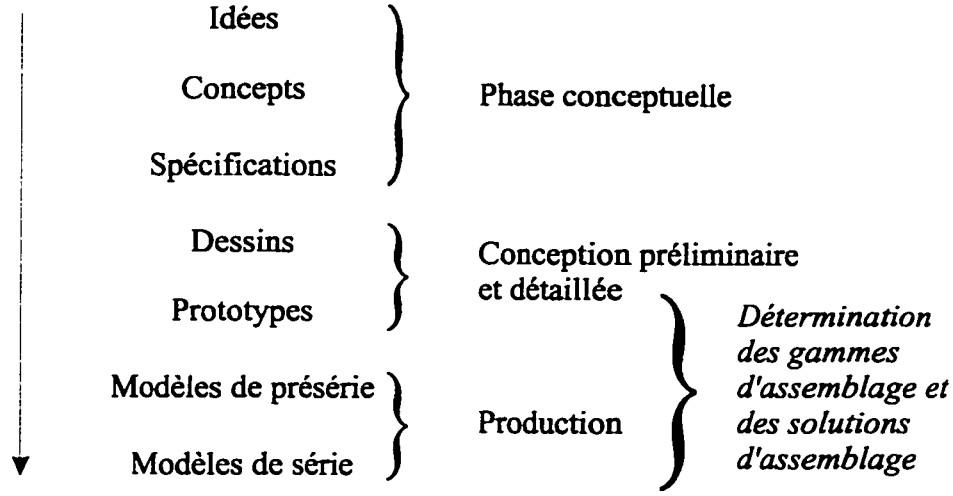


Figure 2.1 : Phases de production d'un produit

Dans cette séquence, l'assemblage se situe environ au niveau des prototypes et des modèles de présérie et de série, et donc après la conception. Cette approche linéaire ne favorise cependant pas la détection des erreurs potentielles tôt dans le processus de conception. En effet, une erreur détectée tard dans le cycle risque d'être très coûteuse car les coûts commis sont élevés (figure 2.2).

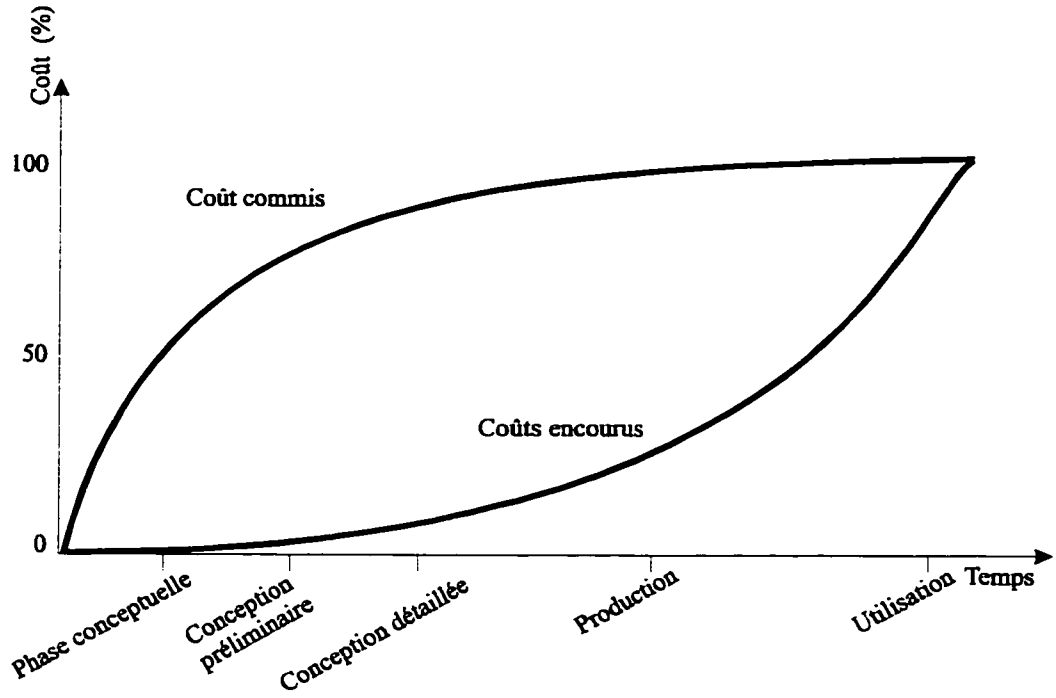


Figure 2.2 : Coûts encourus et coûts commis en fonction de l'avancement d'un projet (Cambron, 1996)

Sur cette figure, deux coûts distincts sont à définir. Premièrement, les **coûts commis** représentent les coûts engagés en fabrication, i.e. le coût total du produit défini à un instant *t*. Les **coûts encourus** représentent les argents réellement dépensés. Ainsi, lorsque environ 20% de la phase de conception est complétée, 80% des coûts de fabrication sont engagés (Cambron, 1996).

Nous voyons ici l'importance d'éviter les erreurs de conception au tout début du projet pour empêcher des coûts de modification trop élevés. Le gammiste doit éviter de retourner le produit dans le département de conception et ce projet tente justement de résoudre ce problème.

2.2.2 Systèmes d'assemblage

La méthode de décomposition suivante des systèmes d'assemblage a pour but de situer chaque sous-système dans le système d'assemblage global (Perrard, 1992). Nous verrons, à la fin de cette description, le sous-système influencé par la solution proposée dans ce travail.

Le **centre de production** regroupe tous les moyens ayant un contact direct avec les produits matériels. Le département de fabrication a pour but de mettre en œuvre le produit grâce à diverses technologies. L'**atelier d'assemblage** est un des ateliers ajoutant le plus de valeur au produit en liant physiquement les différents sous-produits créés.

Un **îlot d'assemblage** est défini comme un sous-ensemble de l'atelier d'assemblage assurant l'assemblage complet d'un produit. La **cellule** quant à elle représente un sous-système à l'intérieur de l'îlot, possédant son propre stock tampon. Elle peut donc travailler indépendamment et réduire les risques d'engorgements sur les lignes de montage suite à de pannes causées par une faible efficacité des machines. À l'intérieur d'une cellule, un **poste** permet à un composant de recevoir un certain nombre de transformations. Une fois ces transformations terminée, un nouveau composant le remplace. C'est finalement à l'intérieur d'un poste que se produisent une ou plusieurs **opérations**, simultanément ou non. Les solutions d'assemblage sont justement implantées à cet endroit.

2.2.3 Ingénierie simultanée

De plus en plus de compagnies se tournent vers le concept de l'ingénierie simultanée (IS) (Ettlie et Stoll, 1990) pour accroître la qualité du produit, tout en diminuant le coût et les délais nécessaire, en évitant l'approche séquentielle et en favorisant une approche plus parallèle. Ainsi, les problèmes potentiels de l'assemblage, par exemple, peuvent être abordés dès l'étape de conception puisque des employés de chaque département peuvent discuter et éviter plusieurs problèmes potentiels.

L'objectif visé dans ce travail se situe dans l'optique de l'IS. En effet, un accord devra avoir lieu entre différents intervenants pour que les solutions présentées aux gammistes satisfassent les intérêts de tous.

2.2.4 Conception en vue de l'assemblage

Tel que mentionné, suite à la conception du produit, de nombreux problèmes apparaissent lorsque vient le temps d'assembler les différents composants. Entre autres, certains composants ne possédant aucune fonctionnalité sont responsables d'une grande partie du coût total de l'assemblage d'un produit. Boothroyd et Dewhurst (1983) ont été les premiers à être reconnus pour leurs travaux effectués dans le domaine de la conception en vue de l'assemblage, ceci grâce à leur méthode *Design For Assembly* (DFA). Cette méthode permet de vérifier la fonctionnalité de chaque composant, afin d'en réduire le nombre et, par la suite, de s'assurer que les composants restants sont faciles à assembler. L'utilisation de nombreux tableaux et graphiques est essentielle pour analyser chaque composant. Ces chercheurs ont commercialisé un logiciel permettant d'évaluer

l'assemblage en répondant à des questions. Leur concept d'attributs associés aux composants sera retenu au cours de ce travail.

Initialement, la conception des produits était faite pour ne répondre qu'aux besoins fonctionnels. Le gammiste, de son côté, devait trouver la meilleure manière d'assembler ce produit. Maintenant, grâce à la méthode DFA, les problèmes d'assemblage sont accessibles à travers la compagnie et des solutions répondant aux objectifs de chacun sont déterminés.

Or, Boothroyd et Dewhurst (1983) n'avaient pas réussi à évaluer automatiquement certains critères, telle la symétrie. Chan et Mo (1993) ont amélioré cette méthode en extrayant l'information contenue dans un logiciel de CFAO pour trouver automatiquement les informations manquantes, par exemple si une pièce est axi-symétrique ou non. Une telle technologie peut avoir du potentiel dans une application voulant déterminer automatiquement des solutions d'assemblage si ces critères géométriques sont intégrés dans le processus de recherche d'une solution d'assemblage.

La vague créée par la méthodologie de Boothroyd et Dewhurst (1983) a été énorme. Beaucoup de gens ont apporté des variantes à leur méthode pour l'adapter à des sujets plus précis. Gabriele (N/D), Wong et Sturges (1992) ont apporté de nouveaux principes sur les structures aérospatiales et les composants lourds, respectivement. D'autres ont dénoncé l'utilisation aveugle de la méthode DFA. En effet, certains reprochaient à cette méthode d'être trop rigide et de concevoir en vue de réduire le nombre d'opérations au minimum, au détriment du temps d'assemblage. Atiyeh (1992) présente un cas où le temps d'assemblage diminue de 169 à 86 secondes, en augmentant le nombre d'opérations de 21

à 25. Fernandez (1993) présente également un cas semblable et explique le pour et le contre d'une réduction de pièces selon plusieurs points, telles la conception des pièces et la qualité de l'assemblage.

Boothroyd et Dewhurst (1983) ne sont pas les seuls à avoir attaqué ce problème d'assemblage. Deux autres méthodes plus ou moins connues ont également vu le jour, soient celles de Hitachi et de Lucas et elles sont analysées par Leaney et Wittenberg (1992). Contrairement à Boothroyd et Dewhurst (1983), ceux-ci perçoivent le problème en considérant la séquence d'assemblage, ce qu'appuie également Kobe (1994). Bien que ces méthodes permettent de trouver des manières d'assembler, elles ne permettent que de déterminer la catégorie d'assemblage – manuelle ou automatisée – et d'optimiser la conception du produit, mais elles n'aident pas à trouver la manière d'effectuer l'assemblage – les manipulateurs à utiliser, la trajectoire d'approche à effectuer, etc.

Dans un cas comme le nôtre, les méthodes de DFA décrites ci-dessus se concentrent surtout sur l'aide à fournir au concepteur. Bien que leur aide ne se destine pas aux gammistes, nous pouvons quand même retenir certains aspects de ces méthodes pour leurs caractéristiques extraites des formes géométriques. De plus, l'idée de fournir des commentaires directement au concepteur – et donc de diminuer le risque d'erreurs plus tard dans le cycle de conception – est très intéressante. Il serait peut-être possible d'intégrer le système étudié ici à l'étape de la conception.

2.2.5 Solutions d'assemblage

L'approche privilégiée dans ce travail est très semblable à celle montrée par Warnecke, Lörh et Kiener (1980). Ainsi, au lieu d'analyser les pièces pour trouver le meilleur assemblage et de l'optimiser par la suite, comme le fait la méthode DFA, ils présentent un catalogue de solutions d'assemblage permettant de choisir une solution – la machinerie à utiliser et les contraintes à respecter – pour une dite opération d'assemblage, allant du stockage des composants manipulés à l'assemblage physique proprement dit. Pour bien visualiser l'emplacement de notre étude dans la subdivision des différents sous-systèmes présentés ci-haut (voir §2.2.2), les solutions trouvées par cette méthode permettent de déterminer les meilleures solutions au niveau des opérations d'assemblage. Warnacke utilise ensuite une analyse de la valeur pour choisir parmi les solutions retenues à chacune des sous-opérations.

Ainsi, chaque opération d'une gamme d'assemblage peut se décomposer en plusieurs sous-opérations, appelées fonctions partielles. La figure 2.3 montre une telle décomposition proposée par Warnecke, Lörh et Kiener (1980).

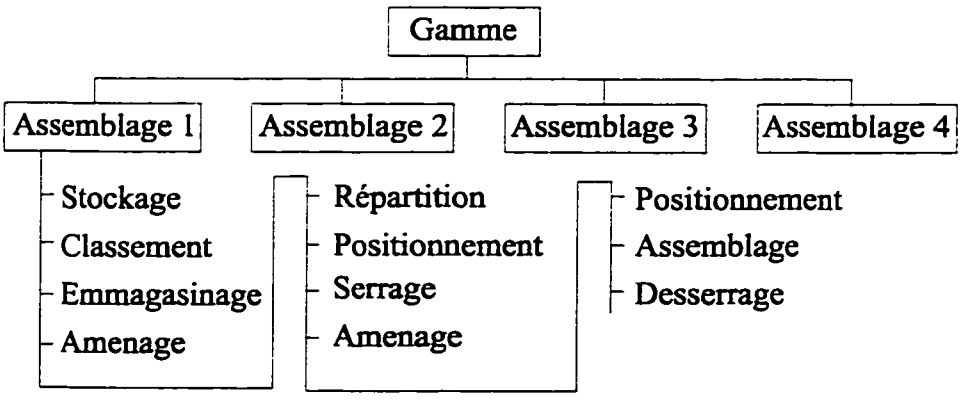


Figure 2.3 : Décomposition d'une opération de la gamme d'assemblage en fonctions partielles

Analysons chacune de ces fonctions.

Stockage : Sous-opération consistant à conserver une quantité suffisante d'un composant en vue d'une utilisation subséquente, sans qu'il ne possède nécessairement un ordre précis.

Classement : Sous-opération servant à mettre les composants dans un ordre précis, selon une certaine orientation.

Emmagasinage : Sous-opération consistant à accumuler une quantité de composants un à la suite de l'autre, avec une orientation précise.

Amenage : Sous-opération consistant à apporter un composant d'un lieu à un autre.

Répartition : Sous-opération servant à distribuer également un composant.

Positionnement : Sous-opération consistant à placer un composant avec exactitude.

Serrage : Sous-opération consistant à maintenir fermement un composant en vue d'une opération subséquente.

Assemblage : Sous-opération consistant à réunir par une liaison physique deux composants.

Desserrage : Sous-opération consistant à relâcher un composant qui était serré.

La partie concernée dans ce travail porte sur certains points de cette décomposition, soient le serrage, l'amenage, le positionnement et l'assemblage.

L'approche par catalogue de solutions d'assemblage est différentes des méthodes nord-américaines habituelles, où les solutions d'assemblage sont souvent résolues cas par cas,

sans nécessairement respecter des méthodologies établies. En effet, les recherches mettent beaucoup d'emphase sur la détermination automatique et l'optimisation des gammes d'assemblage, mais peu d'effort aux choix faits à chaque étape de la gamme.

Cette méthode présente plusieurs inconvénients : le catalogue doit être consulté manuellement, la mise à jour du catalogue comporte certains délais et coûts plus ou moins importants dus à l'impression du nouveau document, et il n'est pas intégré aux systèmes déjà utilisés par l'industrie, comme les logiciels de CFAO, par exemple. Un autre inconvénient de cette approche est qu'elle ne permet pas de percevoir l'assemblage comme une séquence d'opérations, mais limite le gammiste à une vision binaire, entre le récepteur et le composant inséré. De plus, les informations contenues dans ce catalogue sont subjectives – elles sont basées sur l'expérience et non sur des bases très scientifiques – et ne permettent pas de prendre de décisions objectives face à une situation donnée.

En plus de ces inconvénients, nous remarquons que l'information contenue dans ce catalogue se prête mal à une adaptation informatique. En effet, la majorité de l'information inscrite est textuelle et il est ainsi difficile d'effectuer des recherches efficaces et utiles à ce niveau. Par exemple, si une solution du catalogue d'assemblage spécifie que cette solution ne s'applique qu'aux « petites pièces » ou aux « grands débits », il est difficile de caractériser adéquatement ces solutions, et il faudrait utiliser des techniques d'intelligence artificielle, telle la logique floue (§2.5). Or, comme nous le verrons plus tard, il sera possible de contourner ce problème en changeant ces caractéristiques textuelles par des caractéristiques plus appropriées.

Cependant, plusieurs avantages émergent face à l'utilisation d'un tel catalogue. Premièrement, les solutions représentent le savoir-faire d'une compagnie en matière d'assemblage. Ainsi, en facilitant et en optimisant son utilisation, il est possible de réduire les inconvénients liés à l'expertise d'un employé, tel que mentionné précédemment. Deuxièmement, il s'agit d'un outil encore peu utilisé en Amérique du Nord. L'implémentation d'un tel système peut donc plaire à plusieurs compagnies en offrant une méthode innovatrice. Troisièmement, cette méthode s'adapte aisément aux concept de bases de données (§2.4) ; il est donc possible d'informatiser ce catalogue pour permettre sa consultation au travers d'une interface graphique, en lui permettant d'interagir avec tout autre logiciel, spécialement ceux de CFAO.

L'utilisateur premier d'un tel catalogue de solutions d'assemblage sera sans doute le gammiste. Mais connaissant les solutions répertoriées, un concepteur peut également les utiliser à son avantage et concevoir des pièces en fonction de la méthode d'assemblage préconisée, comme dans la méthode DFA. Dans le cadre de l'IS, il est important que de telles notions puissent être partagées entre les autres membres de l'équipe multidisciplinaire.

2.2.6 Sélection de la gamme d'assemblage

La planification de l'assemblage a été analysée en détail par différentes personnes. Elles ont réussi à extraire la grande majorité des contraintes – ou des règles – à considérer durant la planification automatique de l'assemblage. Parmi celles-ci se retrouvent Jones et Wilson (1996) qui ont énuméré de nombreuses contraintes à considérer lors d'assemblages automatisés. Ces contraintes sont des directives de l'usager le guidant vers

un choix final de planification. Par exemple, ces directives comprennent la maximisation de la stabilité et la minimisation du nombre de réorientations des sous-assemblages. L'intérêt de ce travail réside dans l'énumération que l'auteur fait des contraintes faciles à utiliser dans des systèmes informatiques, contrairement aux informations contenues dans le catalogue de solutions de Warnecke, Lörh et Kiener. Certaines de ces contraintes peuvent s'appliquer aux assemblages binaires dans le cadre du choix d'une solution d'assemblage – telle que choisir la solution la moins coûteuse – mais la majorité concerne la séquence d'opérations d'assemblage. Étant donné que nous ne considérons que l'assemblage et non la gamme, cette recherche s'applique très peu dans notre cas.

Schraft (N/D) a également énuméré des règles à suivre en vue d'un assemblage automatique réussi. Il énumère ses dix règles importantes, mais contrairement à Jones et Wilson, Schraft mentionne des concepts généraux, donc complexes à programmer. Un exemple de ces concepts est que les pièces difficiles à manipuler doivent être distribuées par magasins. Les contraintes peuvent ainsi être utilisées pour justifier certaines solutions choisies et s'intégrer au catalogue de solutions pour valider quelques points difficilement vérifiables numériquement.

Afin de déterminer la manière de concevoir un produit en fonction de la séquence d'assemblage, plusieurs recherches ont été effectuées sur le désassemblage (*Design For Disassembling* ou DFD). Brooke (1991) discute les différents secteurs où le désassemblage peut aider l'industrie automobile, surtout en ce qui concerne le recyclage des produits. Cependant, Mascle (1993) a effectué des recherches beaucoup plus poussées où il se sert du désassemblage et de l'ingénierie inverse (*reverse engineering*) afin

d'effectuer une conception de produit et une séquence d'assemblage plus efficaces. Son utilisation des demi-degrés de liberté permet de trouver les directions libres de désassemblage – et donc d'assemblage – et d'ainsi détecter des erreurs potentielles de conception. Bien que le cas présenté dans ce travail se penche sur les opérations effectuées par le gammiste et ne tente pas de proposer des solutions aux concepteurs, les idées provenant du désassemblage permettent une meilleure classification et caractérisation des solutions d'assemblage retenues pour le catalogue de solutions. Par exemple, les degrés de liberté à obtenir suite à l'assemblage de deux composants impliquent des méthodes d'assemblage particulières et des conditions de serrage à respecter.

Dans le cadre de ce travail, l'automatisation de certaines tâches nécessaires dans l'établissement du choix d'une solution d'assemblage répond à un des objectifs de ce travail. En effet, la prise de décision du type d'assemblage comporte beaucoup de particularités. Si le gammiste prend une décision de manière manuelle – i.e. basée sur son expérience personnelle –, elle peut être différente s'il doit résoudre le même problème à des moments différents (Bedworth, Henderson et Wolfe, 1991). Ainsi, la manière d'utiliser son expérience diffère en fonction du cas et du contexte rencontré, entraînant peut-être des solutions différentes pour des problèmes techniquement identiques. Ettlie et Stoll (1990) sont du même avis. Selon eux, trois observations peuvent être faites face à une prise de décision dans une compagnie:

1. En général, différentes solutions existent face à un problème de conception ;
2. La solution choisie peut l'être pour une combinaison de bonnes et de mauvaises raisons ;

3. Contrairement à une méthode scientifique, le processus de conception n'offre pas de garantie intrinsèque que la solution choisie est en effet la meilleure solution ou même la bonne solution.

Cependant, certaines spécifications du produit peuvent aider à prendre des décisions plus automatiques, et donc plus uniformes : la forme, les dimensions et les tolérances pour ne nommer que celles-ci. Une automatisation du choix d'une solution d'assemblage aidera donc à faire respecter les méthodologies internes à une compagnie ; dans le cas où elles seraient inexistantes, la venue d'une tel changement entraînera des remises en question au sein de l'entreprise, et donc des méthodes plus cohérentes.

2.2.7 Les caractéristiques de formes

L'utilisation d'attributs du produit pour caractériser un assemblage fait l'objet de plusieurs recherches. Énormément de définitions existent au sujet de ces caractéristiques de formes – ou *features* – (Salomons, Van Houten et Kals, N/D), mais celle qui reflète le plus le contexte recherché est la suivante :

> *Entité possédant un trait caractéristique, soit sa forme, sa fonction particulière et son montage, et qui peut s'unir à d'autres entités.*

Ainsi, si un utilisateur de logiciel définit un trou et un boulon comme deux caractéristiques de formes distinctes, il pourrait définir un assemblage qui unit les deux.

L'utilisation de caractéristiques de formes n'apporte malheureusement pas les avantages voulus dans ce travail, puisqu'elles ne représentent qu'un endroit local de l'assemblage total et ne permettraient pas une détermination automatique de solutions d'assemblage.

En effet, il n'existe pas actuellement de recherche sur la détermination des liens qui unissent plusieurs caractéristiques de formes. Cet aspect, au moment de la rédaction de ce rapport, est recherché par une équipe dirigée par Jean-Michel Henrioux de l'École Nationale Supérieure d'Ingénieurs de Mécanique et des Microtechniques (ENSMM), à Besançon, France. Un simple exemple qui explique l'échec de cette technique ici est représenté à la figure 2.4. En sachant uniquement qu'une entité **trou** doit s'unir avec une entité **boulon**, la présence d'une forme entraînant de grandes différences sur la méthode d'assemblage choisie est cachée. Cependant, l'utilisation de tels concepts pourrait éventuellement être utile pour définir certains agents de la société créée (§3.4). À la suite de cet échec d'automatisation de la reconnaissance de formes dans le domaine de l'assemblage, le système actuel ne fera que proposer une liste de solutions possibles et le gammiste validera le choix.

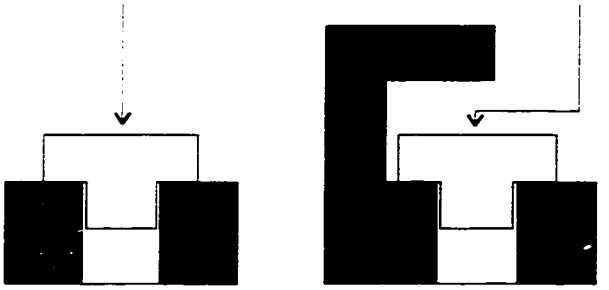


Figure 2.4 : Représentation locale de l'assemblage

De Fazio (1991) utilise justement le concept de caractéristiques de formes pour la planification de séquences d'assemblage, la disponibilité des équipements de bridage et les coûts des opérations d'assemblage. Cependant, bien que cette approche semble donner des résultats intéressants, la présence de caractéristiques de formes est essentielle et alourdit le processus de conception. En effet, leur présence nécessite l'ajout d'une grande

quantité d'information en plus des informations géométriques. Également, son projet ne semble pas présenter une liste de solutions d'assemblage possibles face à un problème posé ; il semble plutôt en imposer une. De plus, dû aux problèmes mentionnés ci-dessus, la recherche de relations entre les caractéristiques de formes est encore à un stade préliminaire et donc peu utile.

2.3 Systèmes de CFAO actuels

L'histoire de la conception et de la fabrication assistées par ordinateur (CFAO) remonte aux années '50 lors de l'apparition des machines à contrôle numérique (NC) (Bedworth, Henderson et Wolfe, 1991). À ce moment-là, des cartes perforées permettaient de programmer les machines afin d'obtenir les mouvements précis voulus. Au cours des décennies suivantes et à la suite de plusieurs changements majeurs est apparu le système de CFAO tel que connu aujourd'hui, avec son écran cathodique et sa souris.

Les logiciels actuels permettent de créer des entités géométriques générées par de puissants algorithmes mathématiques (McMahon et Browne, 1993). À ce jour, trois grands types d'entités géométriques existent : filaire, surfacique et solide. La classe des filaires inclut tous les points, les lignes et les différentes familles de courbes (monoparamétrique). La classe des surfaces inclut les éléments ne possédant aucune épaisseur (biparamétrique). La classe des solides inclut tous les éléments ayant un volume interne (triparamétrique). Cette dernière classe peut elle-même se subdiviser en deux sous-classes : les solides B-rep (*boundary representation*) et les solides CSG (*constructive solid geometry*).

Dû à la précision des entités créées, de tels logiciels s'adaptent mal aux étapes floues du développement d'un produit, telles les étapes des idées, des concepts et des spécifications du produit (§2.2.1). Ceci est compensé par la maîtrise quasi-totale de ces logiciels aux étapes ultérieures du développement du produit, soit en représentation d'un modèle géométriquement précis. D'autres fonctions sont également effectuées par ces logiciels, comme la programmation des machines à commande numérique, l'analyse cinématique des différents composants d'un produit et l'analyse de contraintes par éléments finis.

En effet, il est possible de représenter un composant d'un produit selon trois aspects:

- sa forme ;
- sa tolérance ;
- sa fonction[2].

Ainsi, bien que les logiciels avancés permettent de reproduire avec une très grande précision presque toute forme possible, ils présentent encore beaucoup de faiblesse à conserver les connaissances des utilisateurs, i.e. la fonction. À la suite de cette grande faiblesse sont apparues les caractéristiques de formes (§2.2.7) ainsi que les attributs[3]. Certains logiciels utilisent uniquement ce terme pour signifier des outils pouvant créer automatiquement un certain type de géométrie (par exemple, un trou avec chanfrein), tandis que d'autres logiciels permettent de conserver des connaissances plus précises sur

[2] Mieux connu sous Form, Fit, Function.

[3] Caractéristiques textuelles associées à une entité géométrique dans certains logiciels de CFAO.

l'entité dont, par exemple, la machinerie nécessaire à la fabrication (Gardan et Minich, N/D).

Depuis quelque temps, beaucoup de recherches s'effectuent dans le domaine de l'assemblage, permettant ainsi d'assembler automatiquement des entités géométriques en spécifiant les surfaces en contact et les degrés de liberté permis (ASDESIGN, 1996). L'avantage de ce type de produit, dans la présente recherche, est que les surfaces de contact et les degrés de liberté sont maintenant connus dès la conception du produit. Ainsi, avec un outil approprié, il serait possible d'exploiter ces informations et de faciliter le choix d'une solution d'assemblage pour le gammiste. Cependant, aucun produit semblable n'est actuellement commercialisé. Ainsi, les produits existant dans les logiciels de CFAO se limitent surtout à la représentation informatique de l'assemblage – afin de trouver des interférences – et n'entre pas dans les détails du travail même de l'assembleur ou du gammiste.

Le projet SCOPES (Fan, N/D) est l'un des projets les plus intéressants des dernières années en ce qui concerne toute la planification de la fabrication et de l'assemblage. Son but est d'aider au développement d'un produit durant les étapes suivantes :

- Conception, en utilisant les règles et les recommandations du DFA ;
- Planification de l'assemblage, en utilisant les informations géométriques et topologiques du produit ;
- Création des gammes d'assemblage, en utilisant une base de données des équipements existants ;
- Validation de la gamme par simulation ;

- Planification des différents aspects sur le plancher grâce aux résultats de la simulation.

Ce produit apporte plusieurs bonnes idées, dont la création d'une base de données sur l'outillage de la compagnie.

Quelques années suivant la sortie de SCOPES a débuté le développement de LOGAM'97, un logiciel développé à l'École Polytechnique de Montréal ayant pour but d'aider à la génération des gammes de fabrication, d'inspection et d'assemblage. Il apporte une nouveauté dans la discrétisation de l'assemblage en subdivisant l'assemblage en trois bases de données (§2.4) distinctes :

- les ressources ;
- les méthodes ;
- les gammes.

Les ressources représentent ici le monde physique, dont toute la machinerie existante dans la compagnie. Les méthodes englobent tout le savoir-faire de la compagnie et les gammes contiennent l'information concernant les gammes d'assemblage. Il s'agit d'une subdivision de l'information très intéressante et c'est une représentation similaire qui sera utilisée au cours de ce projet.

2.4 BASES DE DONNÉES

Tel que déjà mentionné, il y a eu un accroissement exponentiel de l'information électronique depuis quelques années. Pour permettre de s'y retrouver, une méthode adéquate de stockage et de manipulation de l'information est nécessaire. Une des

méthodes préconisées pour résoudre ce problème consiste à emmagasiner l'information semblable dans une base de données (BD) (Elmasri et Navathe, 1989).

Une BD peut se visualiser comme un tableau, où les colonnes représentent des informations communes à plusieurs instances (champs), et les lignes représentent différents enregistrements (instances). Par exemple, une BD contenant des informations sur un sujet, tels les amis, aurait comme entêtes de colonne le nom, le prénom, l'adresse, etc. Chaque ligne représenterait un ami précis. Ainsi, le nombre de lignes est variable tandis que le nombre de colonnes est relativement stable. Cette structure, quoique bien simple, permet de résoudre un grand nombre de problèmes, et facilite l'accès et l'utilisation à une grande quantité d'information.

Une révolution s'est produite lors de l'apparition de la base de données relationnelle (BDR) (Atzeni et De Antonellis, 1993). Il s'agit d'un ensemble de tables, reliées entre elles par des liens unissant des attributs (colonnes). Ce système permet entre autre d'alléger la quantité d'informations et d'enlever certaines limites imposées par les BD traditionnelles. La puissance et la flexibilité des bases de données relationnelles proviennent surtout de la stabilité du modèle mathématique sur lequel elles s'appuient. Ce modèle utilise un ensemble minimal complet composé des 6 opérations élémentaires suivantes : renommage, projection, union, différence, produit cartésien et sélection. Toutes les manipulations peuvent s'effectuer à l'aide de ces opérations élémentaires.

Actuellement, la méthode relationnelle est utilisée par les grands gestionnaires de bases de données (GBD) : Oracle de Oracle Corporation et DB2/6000 de IBM. Certains autres logiciels plus petits utilisent également cette technologie, soient Microsoft Access,

Microsoft Foxpro et Borland Paradox. Dans le cadre de ce projet, ce type de gestion de bases de données sera adopté car il s'agit d'une méthode relativement simple, bien répandue et bien éprouvée.

Les systèmes CFAO utilisent actuellement des BDR pour emmagasiner leurs données. Il s'agit de la méthode la plus facile pour manipuler l'information générée par ces logiciels, tels la géométrie, les attributs, etc. Une limite existe cependant dans les BDR car les enregistrements sont limités en nombre de caractères maximaux pour un attribut, tandis qu'un modèle géométrique de logiciel ne possède aucune dimension fixe, pouvant grandir et rapetisser au fur et à mesure que la géométrie change. Les logiciels haut de gamme pallient à ce problème en fractionnant l'information emmagasinée en plusieurs chaînes de caractères de longueur finie et en les unissant grâce à des relations. Le logiciel CATIA, développé par Dassault Systèmes, utilise le CATIA Data Management (CDM) pour manipuler les différents modèles utilisateurs. Ceci permet entre autres, grâce au système relationnel, de créer et de gérer efficacement des assemblages.

Il existe cependant une nouvelle technologie de bases de données qui devrait bientôt prendre d'assaut le marché de l'informatique. Il s'agit des bases de données orientées objet (BDOO) (Kim, 1990). Ce sont des bases de données où chaque instance sauvegardée prend l'allure d'un objet informatique. Ceci évite de séparer l'information en plusieurs relations, permettant ainsi une représentation plus facile d'objets réels. Bien que les BDOO facilitent la manipulation des données, leur technologie est encore trop jeune pour être utilisée. Cependant, leur entrée sur le marché ne devrait pas tarder et, à ce moment, elles seront à considérer.

Lors de la sortie officielle de BDOO stables, cette technologie devrait apparaître dans la majorité des marchés, en particulier celui de la CFAO. En effet, il est naturel de représenter de la géométrie par des objets car toute l'information est retrouvée dans un seul élément informatique, l'objet. Également, la représentation d'assemblages est facilitée par l'absence de relations entre les éléments ; les objets sont intimement liés par leur adresse informatique.

Les BD sont fréquemment utilisées par des logiciels d'intelligence artificielle (IA). De cette manière, les règles d'un système expert (§2.5) sont écrites dans une BD, et la communication entre le logiciel et la BD se fait entièrement via l'interface du GBD. Ainsi, tous les aspects de sécurité et d'intégrité de l'information sont indépendants du logiciel. Également, les créateurs de Java (§2.5) encouragent l'utilisation des bases de données grâce à leur interface JDBC (Java Database Connectivity).

Dans le cas d'un système multi-agents (§2.5), puisque chaque agent est un logiciel indépendant possédant sa propre expertise, il est nécessaire que chacun possède une BD indépendante. Ceci permet de bonnes performances puisque la quantité d'information dans laquelle la recherche s'effectue est réduite. Également, cette indépendance permet la mobilité de l'agent puisque son information lui est propre. Les BDOOs dans ce domaine seront bien reçues car elles assurent implicitement la sécurité de l'information et facilitent l'accès à l'information par l'entremise de méthodes.

Ainsi, une vaste majorité de groupes de recherche appuient l'utilisation de bases de données en les intégrant dans leurs applications multi-agents, dont Pearson *et al.* (Pearson, 1996-1) qui s'en servent afin de conserver les croyances de chaque agent dans un système

de support technique informatisé, où chaque personne possède son expertise et sa compétence.

En résumé, la technologie relationnelle offre la flexibilité, la puissance et la compatibilité recherchées pour conserver adéquatement le savoir-faire de chaque agent dans un système multi-agents. Nous retiendrons donc ce choix dans le cadre de ce travail.

2.5 INTELLIGENCE ARTICIFIELLE

Le concept de l'intelligence artificielle (IA) est né vers les années '50 à la suite de questionnements sur la possibilité de rendre l'ordinateur « intelligent » – c'est-à-dire de permettre aux ordinateurs d'imiter le comportement humain. Ainsi, le but ultime de l'IA n'est pas uniquement d'aider aux prises de décision, mais bien de développer une machine intelligente qui sera capable de prendre elle-même des décisions (Ignizio, 1991).

Weizenbaum (1976) définit l'intelligence artificielle comme faisant partie du domaine particulier des sciences cognitives, soient les sciences essayant de comprendre et de reproduire le raisonnement humain. Parmi les autres domaines des sciences cognitives nous rencontrons la vision robotique, le langage naturel, la reconnaissance de la voix et l'amélioration des interfaces humaines dans les logiciels (Carrico, Girard et Jones, 1989 ; Harmon, Maus et Morrissey, 1988).

Une des recherches intéressantes apportées dans ce domaine concerne la logique floue, développée par Lotfi A. Zadeh. Il s'agit d'une technique permettant de gérer l'imprécision, méthode intéressante à utiliser avec un outil aussi précis qu'est l'ordinateur. Une application a déjà été développée par Balazinski (1994) dans le cadre de la

détermination de paramètres d'usinage pour la fabrication. Dans le cadre de l'assemblage, cet aspect flou apporte peu d'avantages. En effet, afin que cette méthode donne un résultat adéquat, il est nécessaire de déterminer des paramètres auxquels il est possible d'associer des valeurs floues (petit, grand, étroit, large, etc.). Cependant, la méthode de la logique floue nécessite premièrement la création d'une série de graphes d'appartenance ; puis, grâce à des liens entre certains paramètres, une décision peut être prise sur un ou des graphes d'appartenance de résultats. Certes, ceci s'applique adéquatement dans certains cas de la fabrication, où il est utile d'obtenir une approximation numérique d'une vitesse d'avance d'un outil, par exemple, car tous les paramètres initiaux sont reliés entre eux (dureté, contenu en carbone, etc.). Or, dans le cas du choix d'une solution d'assemblage, il est extrêmement difficile de trouver des paramètres reliés. De plus, une légère différence des paramètres initiaux (différence de forme, différence de tolérances, etc.) modifiera entièrement la machinerie utilisée, car il n'y a pas de manière de caractériser les outils graduellement, contrairement à une caractéristique telle la vitesse. Donc, la logique floue ne pourra être retenue pour trouver une solution d'assemblage.

D'autre part, l'idée des systèmes neuronaux a pris beaucoup d'ampleur. Ce sont des algorithmes simulant des neurones, tels que ceux retrouvés dans un cerveau (WhatIs, 1997). À la suite de l'entrée d'un certain signal, les poids accordés aux différents liens unissant les nombreux neurones permettent l'obtention d'une certaine réponse. Ce système offre l'avantage de pouvoir apprendre en réajustant certains poids en vue de l'obtention d'une réponse précise. Cependant, ces systèmes nécessitent normalement un nombre d'entrées fixe. Dans le contexte de l'assemblage, puisque chaque cas comporte ses propres caractéristiques, il n'est pas possible d'avoir un nombre d'entrées fixe.

Les bases de connaissances (Schach, 1993) sont également des avenues intéressantes. Ces systèmes permettent l'inscription des connaissances des experts à l'aide de règles « Si... Alors... Sinon... ». Bien que cette méthode soit simple, elle est particulièrement efficace dans bien des cas, mais pas dans le nôtre. Par exemple, en médecine, une série de symptômes pourra révéler la maladie d'un patient ; en CFAO, une série de règles de conception peut être écrite dans la base de connaissances, afin d'aider le concepteur lors du design (CKE, N/D). Cependant, les règles provenant des experts sont ardues à créer, et les systèmes nécessitent également beaucoup d'entretien étant donné que les règles ne sont pas automatiquement mises à jour.

Or, au cours des dernières années, un nouveau paradigme est apparu dans le domaine de l'intelligence artificielle distribuée (IAD) : les systèmes multi-agents – aussi connus sous le nom d'agents intelligents (Nwana, 1996). Les agents sont de petits programmes indépendants, possédant habituellement leurs propres connaissances restreintes, pouvant communiquer entre eux et pouvant résoudre des problèmes plus complexes dû à leur multiple expertise, d'où l'illusion d'intelligence. Nwana caractérise ces entités informatiques selon trois critères : leur mobilité, leur réactivité et leurs attributs clés possibles – l'autonomie, l'apprentissage et la coopération. Ceci lui permet de définir une série d'agents différents, tel que montré à la figure 2.5.

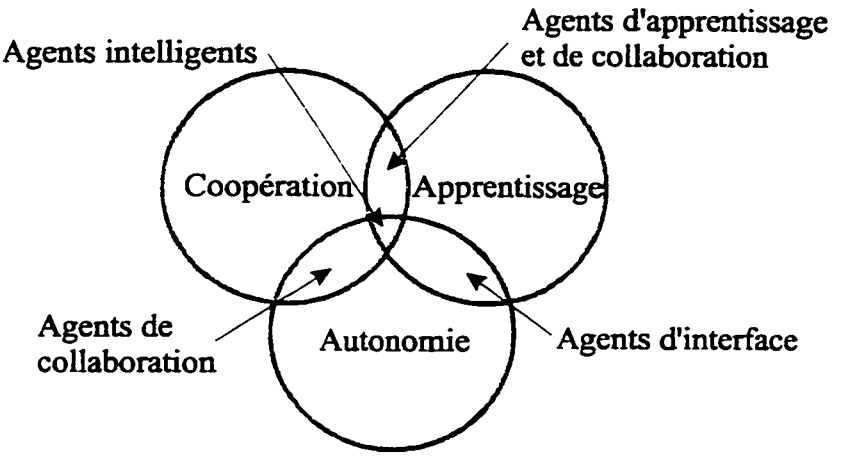


Figure 2.5 : Une vue partielle de la topologie d'un agent (Nwana, 1996)

Belgrave (1995) perçoit pour sa part les agents d'une autre manière. Selon lui, les aspects fondamentaux des agents sont : la persistance, l'autonomie, la réactivité et la communication. Il propose également d'autres caractéristiques qu'ils pourraient posséder : l'initiative, la mobilité, le raisonnement, la planification, l'apprentissage et l'adaptation. Comme il est possible de le remarquer, il n'y a pas d'entente entre les différents groupes de recherche. Dans le cas qui nous concerne, l'agent aura des caractéristiques de mobilité et de communication.

Il existe probablement autant de définitions d'agents qu'il y a de chercheurs sur ce sujet. Cette vaste plage de définitions provient en partie de la grande diversité des applications possibles en utilisant ces agents. Elle vient également du fait que le terme « agent » n'appartienne à personne, contrairement au terme « logique floue » appartenant à L. A. Zadeh, puisque le mot « agent » est utilisé fréquemment dans le langage courant – agent immobilier, agent de voyage, etc. Dans le cadre de ce projet, un agent sera défini d'une manière semblable à Wayner (1995) :

> ***Logiciel pouvant se mouvoir sur un réseau informatique, pouvant communiquer avec d'autre agents, et ayant la possibilité d'acquérir de l'information pour la ramener à son maître ou pour prendre des décisions afin de poursuivre la recherche ailleurs.***

Leur mobilité est un des aspects des plus intéressants pour l'utilisation sur les réseaux informatiques. La compagnie General Magic a développé la technologie Telescript afin de se lancer dans le marché des appareils portables communiquant à un réseau (White, 1995). Cet appareil permet de générer un agent mobile qui peut se mouvoir sur un réseau et résoudre les requêtes de son maître. Cependant, cette technologie est restrictive et ne permet pas de programmer leurs agents pour d'autres tâches sur des réseaux non conçus à cette fin. Cet aspect de mobilité sera tout de même retenu dans notre cas, car il permet la coopération de multiples intervenants localisés à divers endroits (fournisseurs, ateliers, etc.) dans un contexte d'ingénierie simultanée.

La mobilité d'un agent est nécessaire puisqu'il est reconnu qu'un échange est plus efficace pour le logiciel extrayant l'information s'il se situe au lieu même de cette information (Harrison, Chess et Kershenbaum, 1995). Ceci est dû à la grande quantité d'information inutile qui sera transmise. L'agent peut donc faire son propre index de l'information transmise et éliminer tout ce qui est inutile. Également, la mobilité de l'agent évite une grande quantité d'échange sur les réseaux informatiques et permet ainsi de réduire la surcharge des réseaux.

Dû à cette caractéristique de mobilité, un agent doit avoir la possibilité de s'exécuter sur n'importe quelle machine du réseau, quelle que soit la technologie de cette machine – PC, MacIntosh, UNIX, OS/2, etc. Beaucoup de travail a été effectué sur la génération de code

portable, avec un interpréteur présent sur chaque machine. La compagnie Sun Microsystems a développé un langage nommé Java pouvant s'exécuter sur chaque machine (Java, 1996). Le code du programme n'est donc écrit qu'une seule fois, compilé, et ensuite exécuté partout, quelle que soit la plate-forme. Il s'agit donc d'un langage de programmation idéal pour Internet, par exemple, puisque les machines sont diverses. Il est également idéal pour toute autre programmation, car le concept même de ce langage de programmation se situe au niveau des objets. Le code généré est donc plus facilement réutilisable, diminuant ainsi le temps de développement.

Cependant, le code produit après compilation du programme en Java – appelé *bytecode* – a l'inconvénient de s'exécuter lentement sur les machines, car il est interprété. Il s'agit cependant d'un inconvénient mineur, car avec l'évolution de la rapidité des ordinateurs, cet obstacle n'en sera plus un.

Bien qu'il n'existe pas encore – au moment de la rédaction de ce rapport – de moyens directs de programmer des agents avec la technologie Java, des programmes autonomes ont la possibilité d'être créés (grâce à des *threads* ou des démons). Une fois programmées adéquatement, ces entités informatiques peuvent donner des résultats très similaires aux agents intelligents puisqu'ils s'exécutent de manière autonome et peuvent même se déplacer de machine en machine afin de continuer leur exécution ailleurs.

Chess *et al.* (N/D) ont déterminé deux catégories d'applications pour l'utilisation des agents mobiles : 1) utilisation pour les ordinateurs mobiles ou les appareils légers, et 2) utilisation dans les nombreux réseaux nécessitant des méthodes asynchrones pour transférer l'information ou pour effectuer les recherches. Notre travail se concentre sur ce

deuxième aspect qu'est le transfert asynchrone d'information. La puissance de cette méthode provient du fait que, dans le cadre d'un travail distribué (en utilisant des agents mobiles), les requêtes peuvent s'effectuer sur de grandes distances et/ou sur des réseaux surchargés. Ces méthodes asynchrones permettent de continuer l'exécution du programme sans attendre immédiatement des résultats.

La mobilité d'un agent nécessite la présence d'un agent important qu'est le facilitateur (Pearson, 1996-1 et 1996-2 ; Nwana, 1996 ; Williamson, Decker et Sycara, 1996). En effet, un agent ne peut se promener à sa guise sur un réseau et doit donc recevoir les approbations nécessaires avant de pouvoir se copier sur une autre machine, afin d'assurer un minimum de sécurité. De plus, une fois rendu dans une nouvelle société, l'agent doit avoir une manière de connaître les compétences des agents présents. Toutes ces étapes sont effectuées par l'intermédiaire de ces agents facilitateurs. Cette structure est aussi simple que puissante, puisque tous les messages échangés se font via une entité unique qui redistribue ensuite le message au récepteur et tous les changements à l'environnement sont centralisés, évitant les dédoublements. Elle sera donc retenue pour ce travail.

La répartition des réseaux aux quatre coins du monde implique des problèmes de communication. Pour cette raison, en autre, est apparu le KQML – *Knowledge Query and Manipulation Language* (Finin, Fritzson, McKay et McEntire, 1994). Il s'agit d'un langage standard supportant l'identification, la connexion et l'échange d'information dans un contexte client/serveur – donc, entre deux agents. Il permet entre autre de poser la requête dans un langage relativement simple, d'identifier le contexte de la question, tout en permettant de spécifier des informations complémentaires, telles le nom du demandeur

et le numéro d'identification de la requête. Dans le cadre de ce projet, il s'agit d'un langage intéressant, surtout dans un contexte de fournisseurs multiples et donc d'expertise distribuée. Cependant, notons que le KQML est encore à une étape de spécifications, donc difficilement utilisable.

La conservation du savoir-faire est également un sujet traité dans le domaine de l'intelligence artificielle. Afin d'effectuer un parallèle avec le KQML, le langage KIF – *Knowledge Information Format* – a été établi (Genesereth et Fikes, 1992). Ce langage permet la conservation des connaissances à l'intérieur d'une base de connaissances (Knowledge Base System) de chaque agent. Bien que l'idée soit très intéressante, il n'existe pas à l'heure actuelle de logiciel connu utilisant cette technologie, sûrement dû à sa difficulté d'utilisation. En effet, toute connaissance doit être représentée en fonction d'une ontologie[4] commune (Gómez-Pérez, 1994). Une fois tous les agents communiquant avec une même ontologie, ils pourront comprendre le sens exact des requêtes posées et posséderont une manière pour stocker toutes connaissances éventuelles. Il s'agit ici d'une lourde tâche dont il n'existe pas d'implémentation existante dans le domaine de l'assemblage.

La combinaison d'un système multi-agents avec une technologie d'IA suscite beaucoup d'intérêts. Dans un rapport détaillé, Arcand (N/D) propose l'utilisation d'un réseau de neurones artificiel ainsi qu'une chaîne de Markov pour permettre aux agents d'aider les utilisateurs de l'ordinateur.

---

[4] Dans le contexte actuel, spécification explicite de conceptualisation.

L'utilisation de systèmes multi-agents est très répandue afin de réduire la surcharge de travail des utilisateurs. Maes (Firefly, 1996) propose un système intéressant pouvant apprendre les habitudes d'un usager et pouvant prendre des décisions basées sur son apprentissage. Par la suite, l'agent tente de prévoir la prochaine action de l'usager. Selon le niveau de confiance du résultat, l'agent proposera une solution ou, dans le cas d'une grande certitude, il prendra l'initiative sans poser de question. Ce système peut être encore plus évolué, en consultant les habitudes d'autres usagers, en vérifiant la similitude de leurs habitudes avec l'usager présent et en prenant des décisions sur les habitudes d'usagers semblables.

Dans le cadre de ce projet, ces deux derniers systèmes apportent quelques inquiétudes. Il s'agit en effet de systèmes très intéressants et qui prendront probablement beaucoup de place dans le marché de l'informatique dans l'avenir. Mais dans le cas du choix d'une solution d'assemblage, les gammistes doivent demeurer proche des standards de la compagnie et non s'éloigner, en se faisant proposer des habitudes quelquefois subjectives de certains autres employés (Warnecke, Lörh et Kiener, 1980).

CHAPITRE 3. RECHERCHE DE SOLUTIONS

3.1 INTRODUCTION

Le présent chapitre consiste à résoudre le problème du choix d'une solution d'assemblage proprement dit. Il débute en proposant une procédure pour trouver une solution d'assemblage dans notre catalogue de solutions. Puis, les caractéristiques conservées dans notre nouveau catalogue d'assemblage sont décrites en détail, dont la méthode de signature d'assemblage. Par la suite, l'utilisation proposée des systèmes multi-agents est décrite, dont les différents agents nécessaires dans notre société ainsi que la communication qui se fait entre eux. Ensuite, la structure de l'information est analysée afin de voir son implantation grâce à une base de données. Nous concluons le chapitre par une analyse approfondie du système présenté.

3.2 PROCÉDURES POUR TROUVER UNE SOLUTION D'ASSEMBLAGE

Le but de cette section est de voir un scénario typique de recherche de solutions dans un catalogue de solutions d'assemblage. Il faut noter que la mentalité des gammistes doit être modifiée car la présente méthode est différente de celle qu'ils utilisent habituellement.

La première partie porte sur l'analyse du problème d'assemblage telle qu'effectuée aujourd'hui. Il s'agit donc de prendre un composant, de procéder par désassemblage (Mascle, 1993) et de trouver l'ordre avec lequel il faudra assembler le produit. Cette méthode est à la fois simple et pratique car elle permet de détecter des séquences d'assemblage provoquant des interférences et de les retirer des possibilités d'assemblage.

Comme suite à la découverte de l'ordre d'assemblage, le catalogue informatisé proposé ici peut aider à trouver la méthode d'assemblage pour chaque assemblage binaire.

La deuxième partie de la recherche d'une solution d'assemblage consiste à utiliser les informations venant d'un logiciel de CFAO. De plus en plus de logiciels de CFAO permettent de concevoir des pièces tridimensionnelles et d'en placer les modèles les uns par rapports aux autres selon les relations géométriques qui seront celles des pièces elles-mêmes prises deux à deux dans l'état final de chaque relation. En connaissant les composants devant être assemblés, nous pouvons tirer de l'assemblage sa signature (§3.3.1), méthode permettant d'identifier l'assemblage entre deux pièces en fonction de leurs contacts géométriques. Ceci aidera à sélectionner la solution, car un vecteur représentant l'assemblage permettra de filtrer les différentes solutions emmagasinées dans le catalogue, et donc d'améliorer les performances de la recherche.

Par la suite, le gammiste doit déterminer les dimensions du composant manipulé. Ceci a pour effet de filtrer les solutions selon la machinerie pouvant manipuler un composant de cette dimension. Ainsi, puisque à chaque solution est associé un certain nombre de machines, seules les solutions utilisant des machines non refusées seront conservées. Nécessairement, la machinerie devra posséder cet attribut de dimension admissible. Les différents attributs de la machinerie sont exposés à la sous-section 4.3.2. Dans le cas d'une insertion manuelle où les dimensions sont à une échelle « humaine », cet aspect n'est pas envisagé, car il est estimé que l'employé aura la capacité de s'adapter aux dimensions des composants.

Ensuite, il se peut qu'il y ait des restrictions d'ordre temporel à effectuer les assemblages en question. Pour cette raison, une valeur moyenne du temps de cycle contrat est considérée. Il y aura ainsi comparaison entre le temps de cycle contrat T_c et le temps de cycle technique T_t (Mascle, 1993). Le temps de cycle contrat représente le temps à respecter afin de rencontrer les exigences du contrat. Cette valeur tient normalement compte de la quantité horaire de produits nécessaires et de l'efficacité des machines. Le temps de cycle technique, pour sa part, tient compte de la rapidité moyenne des machines aptes à assembler les composants. Ce sera un temps associé avec chaque machine. Ainsi, si $T_c << T_t$, alors nous pouvons assembler en un seul poste, peut-être robotisé ; cependant, il faudra faire attention à l'accessibilité. Si $T_c \sim T_t$, alors une machine transfert peut être employée. Finalement, si $T_c < T_t$, alors un système hybride peut être utilisé, avec quelques opérations effectuées sur un même poste. C'est en comparant la technique d'assemblage nécessaire et celle qui est offerte dans le catalogue que nous pouvons savoir si la solution est réalisable. Bien sûr, si $T_c > T_t$, la solution d'assemblage est soit refusée car elle est trop lente, soit utilisée x fois en parallèle pour compenser son manque de performance.

Par exemple, en comparant le temps de cycle contrat au temps de cycle technique, nous pourrions obtenir $T_c \sim T_t$, ce qui nous permet de conclure que l'utilisation d'une solution transfert serait appropriée. Cependant, si la solution d'assemblage ne respecte pas ces conditions, alors elle sera retirée des solutions proposées.

Ensuite, le gammiste peut prendre une décision sur le type d'assemblage désiré. Il s'agit ici d'un filtre simple, rapide et grossier. Il n'a donc qu'à choisir le type d'assemblage, par exemple une fixation, un positionnement ou un retournement.

Une fois que les différents critères de recherche nécessaires sont connus, soit à l'aide d'un logiciel de CFAO, soit par calcul manuel, un catalogue de solutions d'assemblage informatisé est démarré. Il s'agit d'un logiciel pouvant être consulté tel un catalogue sur papier (Warnecke, Lörh et Kiener, 1980), contenant toutes les solutions retenues par la compagnie. L'ajout d'une solution dans ce catalogue s'effectue par le bureau des méthodes, après consultation des spécialistes en assemblage, pour s'assurer que les solutions soient adéquates et les informations, complètes. Il y a donc création d'une équipe multidisciplinaire, tel que requis dans l'optique d'IS. L'option de recherche permet de sélectionner les meilleures solutions d'assemblage répondant à un certain nombre de critères. La structure multi-agents proposée ci-après (voir §3.5) montre comment l'information sera obtenue.

Voici une liste des différents critères de recherche mentionnés ci-haut :

- Signature d'assemblage ;
- Dimensions de la pièce manipulée ;
- Temps de cycle contrat ;
- Type d'assemblage;
- Force ou moment nécessaire;
- Précision de la machinerie.

Les solutions plausibles sont affichées au gammiste en ordre de pertinence (voir §3.4.5). Il peut maintenant obtenir une description complète de chaque solution, incluant une image du principe utilisé ainsi que certains aspects textuels, tels la sécurité de fonctionnement et les conditions à respecter.

Une fois la solution finale trouvée, le gammiste effectue une dernière vérification en s'assurant que la machinerie pourra assembler les composants sans collision. Cette vérification peut être manuelle ou informatique en utilisant le module KINEMATICS (cinématique) de CATIA par exemple. Il peut également vérifier la disponibilité des machines en consultant l'horaire des assemblages grâce à l'agent d'assemblage (voir §3.5).

3.3 CARACTÉRISATION D'UNE SOLUTION D'ASSEMBLAGE

L'efficacité d'une recherche repose sur les paramètres utilisés pour conserver les solutions d'assemblage. Le but de ce sous-chapitre est de montrer ces paramètres, en particulier une méthode innovatrice proposée, la signature d'assemblage. Ces paramètres serviront par la suite de filtres pour les solutions d'assemblage contenues dans le catalogue de solutions.

3.3.1 Signature d'assemblages

Cette méthode consiste à utiliser la géométrie solide existante d'un logiciel de CFAO. Dans tous les logiciels de CFAO actuels, un solide est défini comme le volume interne fermé d'un groupement de faces. En fait, une entité solide comprend les entités suivantes:

- les faces
- les arêtes
- les sommets

Les faces sont les entités surfaciques définissant les « murs » du solide. Les arêtes (*edges* en anglais) sont les courbes d'intersection de 2 surfaces. Et finalement, les sommets sont

les points d'intersection de 3 arêtes ou plus (voir figure 3.1). Ainsi, à la figure suivante, l'entité contient 6 faces, 12 arêtes et 8 sommets.

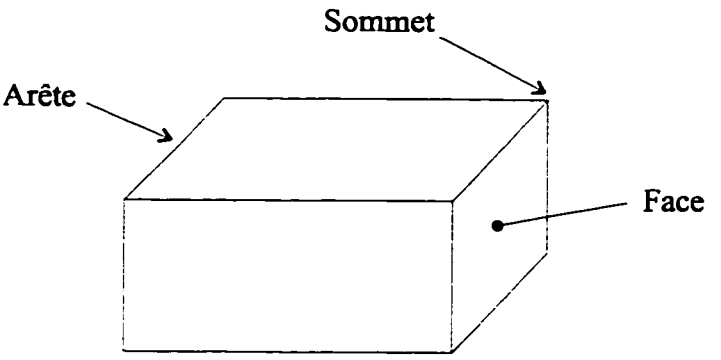


Figure 3.1 : Décomposition d'une entité CFAO en ses composants élémentaires

Certes, le fait d'utiliser dans notre méthode l'élément surfacique, déjà présent dans les logiciels de CFAO, facilite grandement la création des algorithmes mathématiques présentés ci-après.

En effet, le produit ASDESIGN de CATIA (ASDESIGN, 1996) permet d'assembler automatiquement des composants selon différentes contraintes : contact cylindre-cylindre, contact plan-plan et contact sphère-sphère. Il permet également d'imposer des contraintes de distances à respecter entre certains éléments, mais tous ces aspects sophistiqués ne nous préoccupent pas. Il est donc possible d'extraire ces informations du logiciel pour faciliter la reconnaissance des contacts voulus.

Notons que l'assemblage peut être caractérisé par deux étapes distinctes : l'approche et le résultat final. Par approche, nous entendons le parcours nécessaire pour amener la pièce dans son état final, sans collision. Dans le cas d'un logiciel de CFAO, cette information est souvent manquante. Seul un outil de simulation cinématique peut tenir compte de la

dimension temporelle de l'insertion et donc de son parcours. L'information qui nous intéresse surtout dans notre cas est le résultat statique final, avec les différentes faces qui sont en contact, car cette information est bien représentée dans plusieurs logiciels de CFAO.

3.3.1.1 Représentation individuelle

En analysant un solide unique, il est possible de ressortir un graphe montrant l'union entre chaque face. Par exemple, à la figure 3.2, la représentation « graphique » de trois entités élémentaires est effectuée. Le graphe est simplement formé en numérotant chaque face du solide, puis en construisant un schéma où apparaissent toutes les faces. Puis, les faces en contact sont reliées entre elles par une ligne simple.

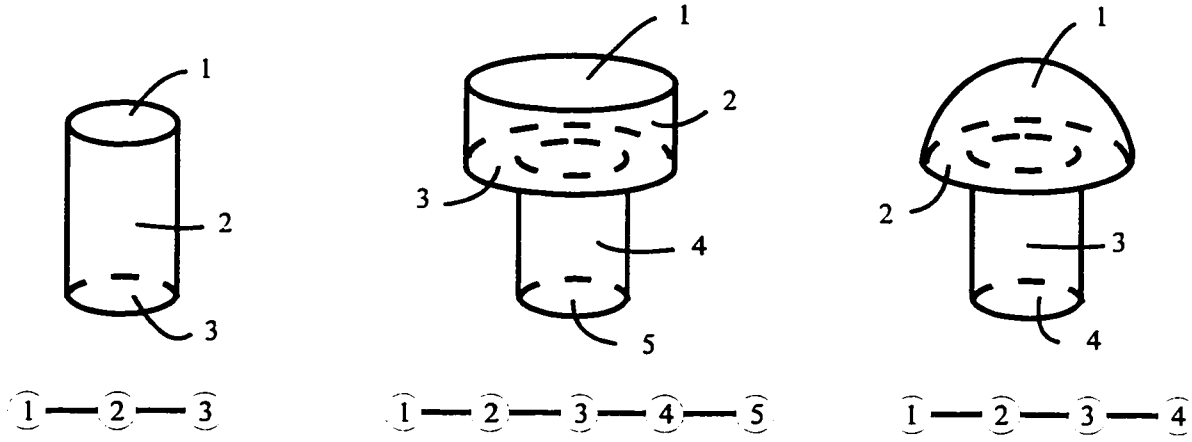


Figure 3.2 : Représentation graphique de certains composants

De la même manière, nous pouvons représenter des composants plus complexes (figure 3.3).

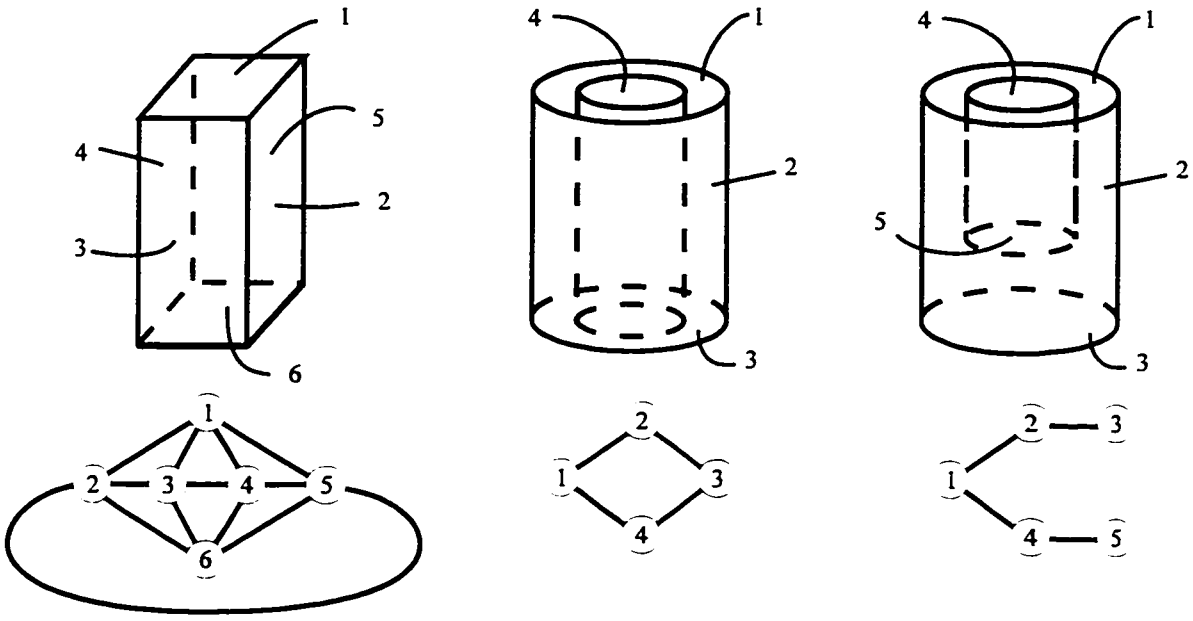


Figure 3.3 : Représentation graphique de composants plus complexes

3.3.1.2 Représentation binaire

Une fois cette représentation unique générée, il est possible de procéder à la représentation binaire de l'assemblage. La figure 3.4 montre un exemple simple d'assemblage binaire.

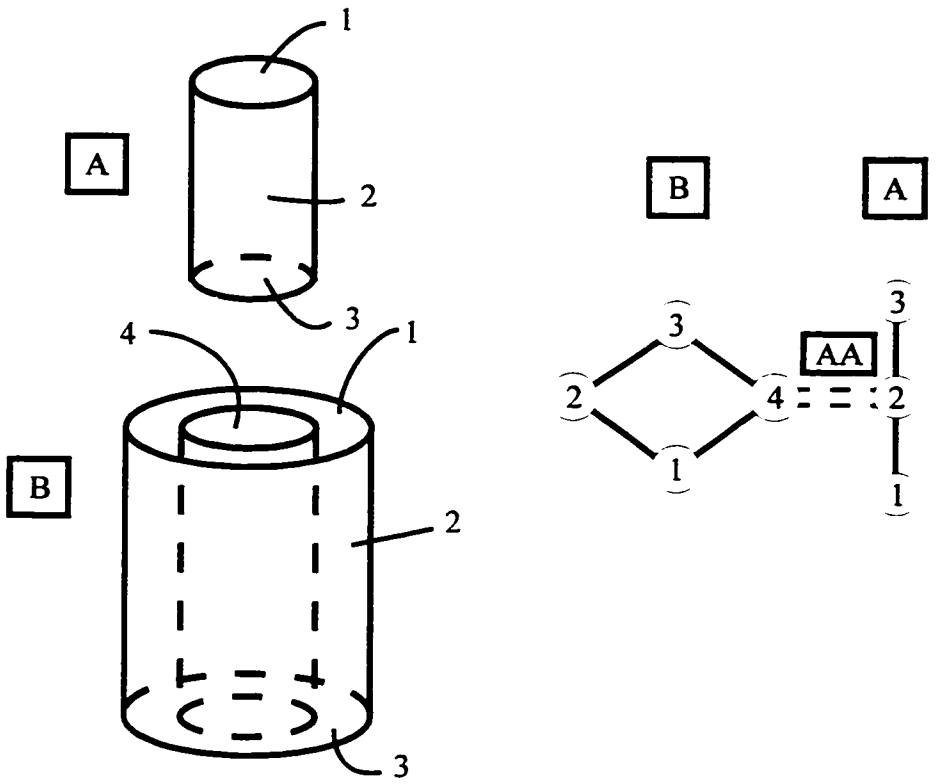


Figure 3.4 : Exemple 1 - assemblage binaire

Ainsi, chaque pièce montrée ci-dessus peut être décomposée dans les tableaux suivants (tableaux 3.1 et 3.2). Ils sont générés en prenant chaque face une à la fois, puis en indiquant les autres faces auxquelles elles sont liées. Notons que la redondance dans le tableau sera retirée plus tard.

Tableau 3.1 : Décomposition de la pièce A (exemple 1)

Pièce	Face	Lié à...
A	1	2
A	2	1
A	2	3
A	3	2

De la même manière :

Tableau 3.2 : Décomposition de la pièce B (exemple 1)

Pièce	Face	Lié à...
B	1	2
B	1	4
B	2	1
B	2	3
B	3	2
B	3	4
B	4	3
B	4	1

Ainsi, on voit apparaître sur la figure 3.4 un double lien *AA*, indiquant les faces de chaque entité qui sont en contact. Ici, la face 2 de la pièce A est en contact avec la face 4 de la pièce B. Écrivons maintenant la matrice complète de ces deux pièces, dans une unique matrice (tableau 3.3).

Tableau 3.3 : Union de la pièce A et B dans un même tableau (exemple 1)

Pièce	Face	Lié à...
A	1	2
A	2	3
A	2	1
A	3	2
B	1	2
B	1	4
B	2	1
B	2	3
B	3	2
B	3	4
B	4	3
B	4	1

En enlevant toutes les lignes redondantes, i.e. tous les liens définis plus d'une fois, nous obtenons le tableau simplifié suivant (tableau 3.4).

Tableau 3.4 : Union des pièces A et B dans un tableau simplifié (exemple 1)

Pièce	Face	Lié à...
A	1	2
A	2	3
B	1	2
B	1	4
B	2	3
B	3	4

La liaison *AA*, pour sa part, peut être simplement définie dans le tableau 3.5.

Tableau 3.5 : Liens unissant les pièces A et B (exemple 1)

Lien	Pièce 1	Face de la pièce 1	Pièce 2	Face de la pièce 2
AA	A	2	B	4

En utilisant les deux tableaux précédents, soit le tableau simplifié des liens entre les faces et le tableau de la définition de la liaison, nous obtenons un tableau global 3.6. Cette union est effectuée en prenant chaque ligne du premier tableau concernant la pièce 1, dont une face fait partie du lien considéré, et en trouvant pour la pièce 2 toutes les lignes dont la face est également dans ce lien.

Tableau 3.6 : Combinaison des pièces et des liens (exemple 1)

Lien	Pièce 1	Face 1	F.1 Lié à...	Pièce 2	Face 2	F.2 Lié à...
AA	A	2	3	B	4	1
AA	A	2	3	B	4	3
AA	A	2	1	B	4	1
AA	A	2	1	B	4	3

Suite à la création de ce tableau, on peut créer une variable v pour chaque paire (p,f), où p est le numéro (nom) de la pièce, et f est le numéro de la face. Donc, définissons dans le tableau 3.7 certaines variables, dites d'utilisation (notées par des lettres minuscules), pour chaque face retrouvée dans le tableau 3.6 :

Tableau 3.7 : Détermination des variables d'utilisation (exemple 1)

Variable	Paire (p,f)
a	(A,1)
b	(A,2)
c	(A,3)
d	(B,1)
e	(B,3)
f	(B,4)

Maintenant, reprenons le tableau 3.6 en créant, pour chaque ligne, un tableau pouvant posséder deux valeurs : 0 si le lien n'est pas présent, 1 si le lien est présent. Nous obtenons donc le tableau 3.8, avec la somme de l'utilisation de chaque variable au bas :

Tableau 3.8 : Somme des variables d'utilisation (exemple 1)

	(A,1)	(A,2)	(A,3)	(B,1)	(B,3)	(B,4)
	a	**b**	**c**	**d**	**e**	**f**
	0	1	1	1	0	1
	0	1	1	0	1	1
	1	1	0	1	0	1
	1	1	0	0	1	1
$\Sigma =$	**2**	**4**	**2**	**2**	**2**	**4**

Nous obtenons finalement le vecteur s suivant, appelé *signature de l'assemblage*, en plaçant les valeurs des sommes du tableau 3.8 en ordre croissant:

$$s = [2,2,2,2,4,4] \;\; ; \;\; n = 6$$

où n représente la dimension (le nombre d'éléments) du vecteur s.

3.3.1.3 Autre exemple

Reprenons ces étapes pour trouver la signature de l'assemblage suivant (figure 3.5):

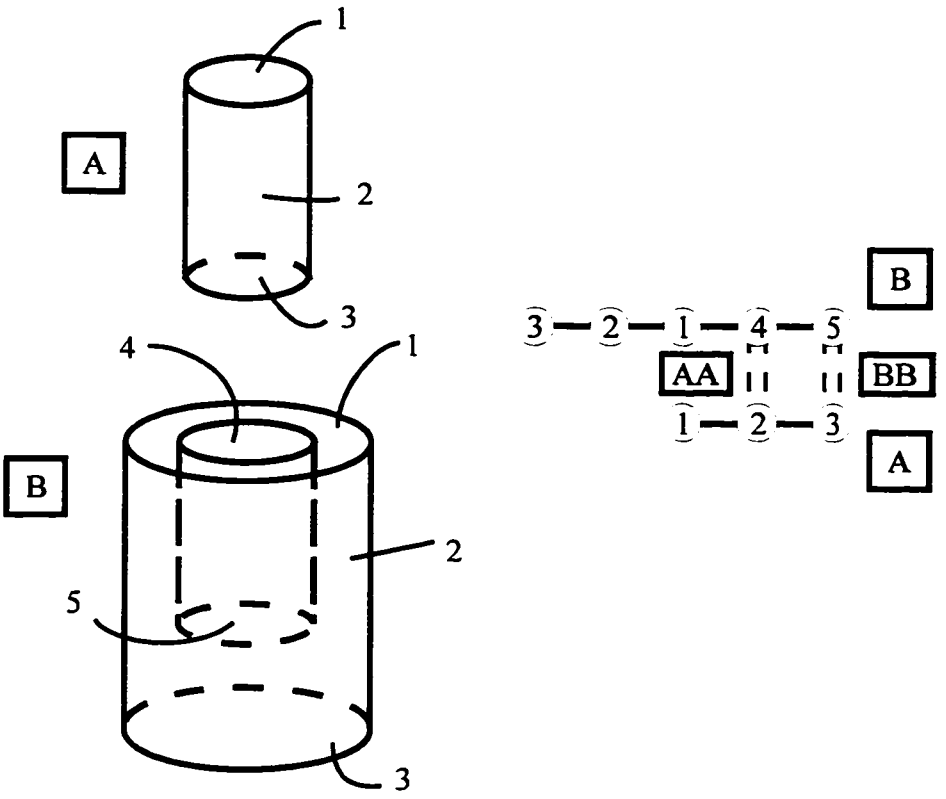


Figure 3.5 : Exemple 2 - assemblage binaire

Commençons par définir les faces des 2 pièces dans un seul tableau unique (tableau 3.9).

Tableau 3.9 : Union des pièces A et B dans un même tableau (exemple 2)

Pièce	Face	Lié à...
A	1	2
A	2	1
A	2	3
A	3	2
B	1	2
B	1	4
B	2	1
B	2	3
B	3	2
B	4	1
B	4	5
B	5	4

Une fois le tableau 3.9 simplifié, nous obtenons le tableau 3.10 :

Tableau 3.10 : Union des pièces A et B dans un tableau simplifié (exemple 2)

Pièce	Face	Lié à...
A	1	2
A	2	3
B	1	2
B	1	4
B	2	3
B	4	5

Maintenant, écrivons le tableau de définition des liens dans le tableau 3.11 :

Tableau 3.11 : Liens unissant les pièces A et B (exemple 2)

Lien	Pièce 1	Face de la pièce 1	Pièce 2	Face de la pièce 2
AA	A	2	B	4
BB	A	3	B	5

Unissons ces deux derniers tableaux pour obtenir le tableau 3.12 :

Tableau 3.12 : Combinaison des pièces et des liens (exemple 2)

Lien	Pièce 1	Face 1	F.1 Lié à...	Pièce 2	Face 2	F.2 Lié à...
AA	A	2	3	B	4	1
AA	A	2	3	B	4	5
AA	A	2	1	B	4	1
AA	A	2	1	B	4	5
BB	A	3	2	B	5	4

Définissons maintenant les variables créées à partir des paires (*p,f*) (tableau 3.13).

Tableau 3.13 : Détermination des variables d'utilisation (exemple 2)

Variable	Paire (p,f)
a	(A,1)
b	(A,2)
c	(A,3)
d	(B,1)
e	(B,4)
f	(B,5)

Grâce au tableau généré par l'union des tableaux de liens et de faces (tableau 3.12), nous obtenons la matrice suivante (tableau 3.14) :

Tableau 3.14 : Somme des variables d'utilisation (exemple 2)

	a	**b**	**c**	**d**	**e**	**f**
	0	1	1	1	0	1
	0	1	1	0	1	1
	1	1	0	1	0	1
	1	1	0	0	1	1
	0	1	1	0	1	1
$\Sigma =$	**2**	**5**	**3**	**2**	**3**	**5**

Nous trouvons finalement la *signature de l'assemblage* suivante:

$$s = [2,2,3,3,5,5] \;\; ; \;\; n = 6$$

3.3.1.4 Remarques

Les cas analysés ici-haut ne sont qu'un échantillon des configurations examinées. Un plus grand nombre de situations géométriques est présenté à l'annexe I. Les cas analysés comportent les caractéristiques suivantes : chanfreins, appuis dans le fond du trou et à la tête du trou et formes prismatiques. Ces résultats démontrent que certains aspects géométriques, tels les chanfreins, influencent la signature si et seulement si cette face est localisée entre des surfaces en contact (voir cas 4 et 5 à l'annexe I). Dans les cas contraires, le chanfrein est ignorée par le calcul de signature d'assemblage et la signature est identique (voir cas 1 et 2 à l'annexe I).

Cette méthode s'applique à tout type d'assemblage, aussi longtemps que les surfaces en contact soient surfaciques. En effet, dans tout logiciel de CFAO, les éléments sont décrits de la manière utilisée ici, i.e. avec des faces, des arêtes et des sommets, aussi connu sous le nom de représentation par frontières (voir §2.3).

La précision mentionnée au dernier paragraphe est importante car nous ne pouvons, avec notre méthode, résoudre des problèmes où les contacts sont linéaires (voir figure 3.6). En effet, l'assemblage montré à cette figure permet de bloquer toutes les translations et rotations, sans créer d'état hyperstatique. Or, pour ce faire, des contacts linéaires tels que montrés sur la figure sont nécessaires et de tels contacts ne sont pas supportés par les algorithmes actuels.

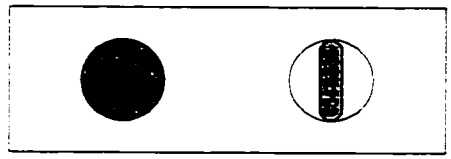


Figure 3.6 : Exemple de contact linéaire entre deux surfaces

La méthode de signature d'assemblage ne permet pas non plus d'identifier le type de contrainte (degrés de liberté bloqués ou libres) entre les pièces assemblées. Ainsi, bien que des assemblages d'une tige rectangulaire et celui d'une tige pentagonale dans leurs trous de forme respective donnent des assemblages mécaniques où la rotation de la tige est bloquée, leur signature n'est pas la même. Cependant, il y a peu de risque d'erreur en concluant que la valeur de s dans le premier cas sera plus petite que dans le deuxième cas, car le nombre de surfaces en contact est plus élevé.

La signature d'assemblage ne peut alors pas identifier les degrés de liberté existant entre deux éléments assemblés. En effet, le calcul de la signature se base sur les lieux de rencontre entre les différentes surfaces composant chacune des pièces. Cependant, l'apparition de surfaces supplémentaires (chanfreins, rainures, etc.) influence ce calcul et peut mener à des réponses ambiguës avec d'autres cas rencontrés. Il est donc impossible de ressortir une signature propre à une configuration de degrés de liberté.

À ce dernier point s'ajoutent les directions de blocage causées par les faces en contact. En effet, certaines configurations de contact peuvent donner des signatures d'assemblage comparables sans toutefois nécessiter la même machinerie ou présenter une fonctionnalité identique. Cette particularité a l'avantage de n'offrir qu'une vision locale de l'assemblage, i.e. toutes les surfaces qui ne sont pas impliquées directement dans

l'assemblage sont ignorées du calcul. En effet, si le cylindre troué de l'exemple 1 était remplacé par un cube troué, la signature serait identique. La signature d'assemblage n'est donc qu'un outil d'aide à la décision et non un outil d'automatisation ; l'intervention humaine est quand même nécessaire. De plus, **quelles que soient les formes des pièces assemblées, la signature sera identique si la configuration des surfaces en contact l'est aussi.**

Ce dernier aspect apporte plusieurs avantages car cela permet, entre autre, de « filtrer » la pièce (enlever les complexités des pièces) et de ne regarder que le contact final résultant de l'assemblage (voir les exemples avec chanfrein en annexe I). Dans le cas d'une recherche de solutions d'assemblage, cette signature offre un intérêt dans la recherche d'une solution, car il est possible de classer les solutions d'assemblage en fonction des signatures qu'elles permettent d'obtenir. Bien qu'il puisse arriver des occasions où il est intéressant d'avoir un aperçu plus global de l'assemblage, par exemple l'impossibilité d'une approche rectiligne due à une forme complexe, cette méthode apporte beaucoup de possibilités face à une éventuelle détection automatique des assemblages.

Mentionnons également que pour qu'une méthode soit efficace et universelle, le résultat obtenu doit être indépendant de la numérotation effectuée. C'est exactement ce que nous apporte cette méthode de signature. Ainsi, quelle que soit la numérotation initiale des faces, le résultat est identique, ce qui est tout à fait logique.

L'analyse des graphes créés pour les signatures d'assemblage nous permet d'utiliser la théorie des graphes. Or, une caractéristique de cette théorie nous permet de déterminer concerne la notion d'unicité, i.e. il n'existe pas deux assemblages différents donnant la

même signature. Notons que la signature d'assemblage ne présente pas cette caractéristique d'unicité. En effet, l'insertion d'une pièce à tête conique à l'intérieur d'une pièce avec un chanfrein permettant d'accueillir cette tête donne la même signature que l'insertion d'un boulon dans un simple trou (voir cas 4 à annexe I). Or, ces deux méthodes nécessitent des outils d'assemblage différent car la tête de la pièce insérée du premier cas devient coplanaire avec la surface supérieure, empêchant ainsi tout outillage avec préhension par la tête. Néanmoins, la recherche des solutions admissibles en fonction d'une signature donnée sera moins laborieuse.

Il va falloir dans des recherches ultérieures analyser la signature d'assemblage pour voir :

- si nous pouvons détecter des assemblages hyperstatiques (par exemple, insertion de deux tiges séparées d'une distance *d* provenant d'une même pièce dans les deux trous d'une autre pièce) ;
- si nous pouvons détecter des pièces bloquées, i.e. impossibles à désassembler (très utile pour la détermination automatique de gammes d'assemblage) ;
- si nous pouvons améliorer la méthode pour considérer l'approche et non seulement le résultat final.

L'annexe II représente cet algorithme de manière schématique, tandis qu'un code fonctionnel rédigé en C est situé à l'annexe III.

3.3.2 Autres caractéristiques de recherche d'assemblage

Outre la signature d'assemblage, quelques autres caractéristiques sont retenues pour effectuer la recherche des solutions d'assemblage possibles dans le catalogue. Nous

retrouvons le type d'assemblage, les dimensions de la pièce manipulée et le temps de cycle contrat.

Le type d'assemblage représente la famille globale dans laquelle se situe l'assemblage en question. Parmi les types d'assemblages retrouvés, nous rencontrons le positionnement, la fixation et le classement. Tous ces types ont été extraits du catalogue de solutions d'assemblage de Warnecke (Warnecke, Lörh et Kiener, 1980). Il est ainsi possible d'emmagasiner plusieurs types de solutions d'assemblage à l'intérieur du catalogue de solutions et de ne ressortir que les solutions pertinentes à l'assemblage rencontré. Il s'agit donc d'un filtre grossier et efficace. Bien sûr, à l'intérieur d'un type d'assemblage peut se retrouver un second classement, nommé sous-type d'assemblage. Nous en tiendrons compte dans notre analyse des bases de données nécessaires.

Les dimensions de la pièce manipulée sont importantes afin de s'assurer que les manipulateurs sont aptes à manier la pièce à assembler. Chaque machine – préhenseur, visseuse, etc. – possède ainsi des caractéristiques clés de dimensions des pièces pouvant être manipulées. Seules les solutions élaborées à partir de la machinerie valide seront retenues. Il s'agit ici d'un filtre de solutions basé uniquement sur le volume général d'une pièce, i.e. le plus petit prisme rectangulaire englobant la pièce. Il faut cependant que ce prisme soit construit adéquatement pour ne pas confondre les largeurs et la hauteur, car la préhension s'effectue normalement sur ses largeurs. Pour cette raison, il est nécessaire de créer une référence locale, où un certain axe (l'axe des z) représente la direction d'insertion. Ainsi, les dimensions de préhension seraient prises à partir des valeurs en x et en y de cet axe.

Le temps de cycle contrat T_c (Mascle, 1993), pour sa part, représente le temps moyen de création d'une nouvelle pièce à obtenir pour respecter les exigences du client. Il est possible d'avoir une appréciation de l'assemblage requis en comparant cette valeur au temps de cycle technique T_t, représentant le temps moyen requis pour qu'une machine particulière effectue son cycle.

La comparaison de ces deux temps de cycle permet de valider le type de solution présenté et de filtrer uniquement les solutions adéquates. En effet, si $T_t \sim T_c$, une solution par transfert est à proposer. Si la solution d'assemblage ne peut s'harmoniser à ce type de solution, alors la solution est refusée.

3.3.3 Caractéristiques générales d'une solution d'assemblage

La solution d'assemblage doit fournir au gammiste les informations essentielles pour résoudre le problème posé. Parmi les éléments expliqués ci-dessous, certains se retrouvent dans le catalogue de solutions d'assemblage de Warnecke (Warnecke, Lörh et Kiener, 1980).

3.3.3.1 Numéro d'identification de la solution

Il s'agit d'un simple numéro d'identification. Il est utile dans le contexte d'une base de données relationnelle, où l'utilisation d'une clé est essentielle.

3.3.3.2 Titre de la solution

C'est le titre de la solution, en quelques mots seulement.

3.3.3.3 Type d'assemblage principal

Il s'agit du type d'assemblage, par exemple fixation, répartition et positionnement.

3.3.3.4 Sous-type d'assemblage

Chacun des assemblages principaux peut se décomposer en sous-types d'assemblage plus précis, chacun faisant partie d'une famille particulière. Par exemple, parmi les types de fixation connus, nous retrouvons l'insertion, le rivetage, le sertissage et le vissage.

3.3.3.5 Principe de la solution

Il s'agit d'un bref texte expliquant le principe de la solution, en utilisant normalement les mots du type et du sous-type d'assemblage dans son texte. Ce n'est qu'une simple description expliquant la règle générale et qui peut montrer les types de composants aptes à être utilisés pour cette solution.

3.3.3.6 Sécurité de fonctionnement

Cette caractéristique donne une idée sur les aspects particuliers à considérer lors de la mise en œuvre de cette solution. Nous parlons fréquemment ici de vibration et d'instabilité possible. Également, tout aspect particulier à considérer pour la sécurité des travailleurs est expliqué ici.

3.3.3.7 Assemblage automatique ou manuel

Cette caractéristique démontre s'il s'agit d'une solution à principe général automatique ou manuel. Nous pouvons ainsi conclure si du personnel supplémentaire est nécessaire ou si

nous devons prévoir l'utilisation de machinerie automatique. Nous nous attendons normalement, dans le cas d'assemblage manuel, à un temps de cycle technique plus long, mais à une possibilité de mouvements plus caractérisés par une grande adaptabilité.

3.3.3.8 Critères d'utilisation

Ce texte permet de voir les cas dans lesquels cette solution peut être utilisée. Nous retrouvons ici certains aspects de géométrie (par exemple : L/D < 1,5) ou d'autres aspects dus, par exemple, aux matériaux avec lesquels les pièces doivent être fabriquées pour permettre l'utilisation de cette solution.

3.3.3.9 Image de la solution

Cette image permet une appréciation visuelle de la solution en question. Il est ainsi possible de voir l'approche des composants et certains mouvements relatifs. Ainsi, tous les aspects de collisions possibles doivent être analysés ici en fonction de la géométrie réelle des composants à assembler.

3.3.3.10 Machinerie/outillage nécessaire

Cette caractéristique énumère tous les outils, dans un sens général, essentiels à l'utilisation de cette solution. Ainsi, dans le cas d'une solution automatisée avec préhension, tous les préhenseurs possibles seraient énumérés ici. Également, tout autre outillage, tel des graisses et des gants particuliers à utiliser, serait présent.

3.3.3.11 Appréciation de l'assemblage

Dans le contexte où une certaine solution doit être choisie, un gammiste sans expérience ne peut apprécier totalement une solution uniquement avec les informations énumérées ici. Pour cette raison, une caractéristique numérique d'appréciation générale de l'assemblage de 1 à 10 permet au gammiste d'écarter certaines solutions. Cette valeur peut tenir compte d'aspects de difficulté non qualifiables ou d'efficacité générale du principe de la solution. Il tient donc compte de l'expérience passée d'utilisation de ce principe de solution.

3.3.3.12 Étapes préalablement nécessaires

Il est possible que certaines solutions d'assemblage nécessitent certaines étapes de préparation, soit du graissage, du préchauffage ou du nettoyage. Ainsi, cette caractéristique permet d'énumérer toutes les étapes afin d'assurer un assemblage de qualité, tout en respectant les normes de la compagnie.

3.3.3.13 Temps de cycle technique

Il s'agit du temps de cycle technique de la solution d'assemblage en fonction des machines utilisées. Étant donné qu'une même solution d'assemblage peut s'effectuer grâce à différentes machines (selon les dimensions du produit à assembler, la précision requise, etc.), le temps de cycle varie en fonction de la machinerie choisie. Ainsi, le temps de cycle moyen sera déterminé en fonction de l'information obtenue sur la machinerie (agent outillage, §3.4.4).

3.3.3.14 Coût de la solution

Dans le contexte économique actuel, il est essentiel dans certains cas d'utiliser la solution la moins coûteuse pour permettre un assemblage rentable. Pour cette raison, la caractéristique de coût de la solution est ajoutée, afin que le gammiste ait la possibilité d'évaluer cet aspect de la solution et qu'il puisse faire un choix éclairé. Notons cependant que nous choisissons normalement la solution offrant la meilleure valeur, i.e. celle qui répond le mieux aux objectifs fixés, et non uniquement la moins chère.

3.3.3.15 Image des sous-opérations

Certaines solutions conserveront quelques images de sous-opérations de l'assemblage. Ceci fournit une explication plus précise des différentes étapes possibles, comme des endroits de graissage. Il s'agit donc d'une information intéressante pour les gens du plancher d'assemblage. Ainsi, dans le cas où certains documents d'assemblage devraient être générés, cette information serait incluse.

3.3.3.16 Signature de l'assemblage

Cette caractéristique permet de conserver, pour certains types d'assemblages, une ou des signatures d'assemblage normalement retrouvées. Ainsi, lors de la recherche d'une solution respectant une certaine géométrie particulière, cette signature permet un filtrage intéressant des solutions d'assemblage, en n'identifiant que les solutions permettant d'assembler une géométrie particulière.

Voici l'apparence d'une page du catalogue de solutions d'assemblage (figure 3.7).

Problème d'assemblage :

Fixation d'un composant par

Introduction de la pièce 1 dans la pièce 2.

Titre : Appareil de préhension oscillant et doigt de centrage	**Identification :** F1113

Principe :

- Répartition et positionnement des bagues sur la douille par un doigt oscillant avec éjecteur.
- Un doigt de centrage, traversant la douille, centre la chute libre de la bague dans la douille.

Sécurité de fonctionnement :	**Critères d'utilisation :**
• Des perturbations peuvent se produire lors de la chute libre de la bague sur le doigt (bagues présentant des arêtes vives).	• Assemblage de pièces annulaires en position centrée. • Uniquement pour des composants présentant un trou central.

Étapes préalables nécessaires :

- Aucune.

Temps de cycle technique : 2,3 sec.	**Coût de l'assemblage :** 0,04$
Signature de l'assemblage : 2 2 2 2 4 4	**Appréciation :** 6

Outillage : DC1002 (doigt de centrage), EJ4883 (éjecteur avec pince de serrage)

Image de la solution :

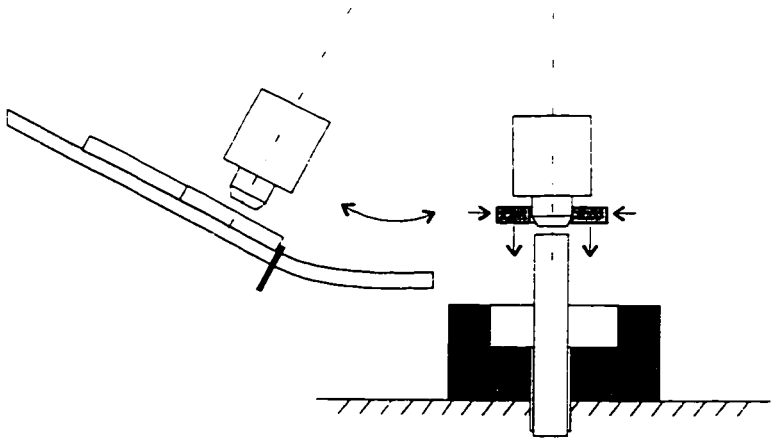


Figure 3.7 : Exemple de page du catalogue de solutions d'assemblage

Nous croyons que ces éléments de caractérisation combinés au jugement du gammiste lui permettront de trouver aisément une solution d'assemblage pouvant résoudre son problème posé.

3.4 UTILISATION DES SYSTÈMES MULTI-AGENTS

La structure d'agents utilisée dans le projet repose grandement sur la structure d'une société, d'où l'appellation fréquemment rencontrée de société d'agents. En effet, chaque membre de la société, soit les agents, possède ses connaissances et son expertise, et peut ainsi partager son savoir avec les autres membres. Ainsi, bien que chaque agent possède une expertise pointue, la société réussie à accomplir des tâches complexes.

L'approche par système multi-agents apporte plusieurs avantages. Premièrement, la programmation est modulaire, permettant ainsi une programmation robuste et plus simple. Cette mentalité repose sur le proverbe anglais « *Divide and conquer* », signifiant qu'il est plus facile de résoudre un gros problème en le subdivisant en plusieurs petits problèmes.

Deuxièmement, une société peut évoluer et chaque membre peut s'insérer et se retirer sans que cela n'influence le fonctionnement de cette société. Une des principales raisons de cette qualité provient du langage de requête, le KQML (Finin, Fritzson, McKay et McEntire, 1994). Il permet en effet de formuler des requêtes en demandant à l'agent contacté s'il possède les connaissances pour répondre à ces questions. Dans la négative, il est possible de lui demander s'il connaît l'emplacement d'un agent qui pourrait l'aider. C'est donc un langage très flexible, approprié dans ce contexte de société en constante évolution.

Troisièmement, les traitements effectués sont asynchrones (Chess, N/D), signifiant qu'aucun arrêt de travail de la part des agents n'a lieu lorsqu'ils sont en attente d'une réponse. Dans le cas d'un travail distribué, où le trafic des réseaux peut provoquer des retards de transmission de l'information, il est avantageux de conserver un agent en état d'utilisation.

Quatrièmement, les agents sont idéaux pour conserver l'expertise d'une seule personne. Dans le cas d'expertise distribuée ou de conservation du savoir-faire, chaque agent pourrait représenter un employé particulier et connaître ses forces et ses faiblesses. Ainsi, lorsqu'un employé fait face à une tâche difficile, son agent peut immédiatement consulter des agents experts dans ce domaine ; à l'inverse, s'il possède lui-même une expertise dans un domaine utile pour les autres agents, il la partage avec les autres. Cependant, tel que mentionné par Warnecke, Lörh et Kiener (1980), le savoir-faire ne sera pas conservé pour chaque individu, mais bien pour un groupe en entier, car l'expertise partagée doit reposer sur des méthodes acceptées par l'ensemble des intervenants de la compagnie et non sur les goûts d'un utilisateur en particulier.

Plusieurs difficultés font malheureusement surface lors de l'implantation de systèmes multi-agents. Premièrement, il doit y avoir accord sur les règles de grammaire utilisées et le langage, soit l'ontologie. En effet, dans chaque spécialité ou dans chaque société, les termes de communication ne doivent pas être ambigus. Ainsi, lorsqu'un nouvel agent est inséré dans la société, il doit pouvoir communiquer efficacement avec les autres agents (Gómez-Pérez, 1994). Il s'agit donc d'un point vaste où la complétude est essentielle, d'où la certaine complexité d'implantation et de spécification.

Deuxièmement, l'expertise nécessite un encodage spécial à l'intérieur même de l'agent, spécifié par la norme KIF (Genesereth et Fikes, 1992). Cette norme se joint partiellement au premier problème soulevé, soit l'ontologie, puisque les termes et les règles tirés de l'ontologie devront être conservés de cette manière. Ce type de codification peut non seulement servir à emmagasiner l'information à l'intérieur d'un agent, mais peut également permettre la diffusion d'information d'un agent à un autre. Dans le cadre de ce travail, l'exploration d'un tel standard est très approprié, mais dû aux difficultés de son implantation et à l'aspect « étude » du projet, il n'est pas essentiel d'analyser ce langage en profondeur.

La structure multi-agents proposée dans ce travail est la suivante (figure 3.8).

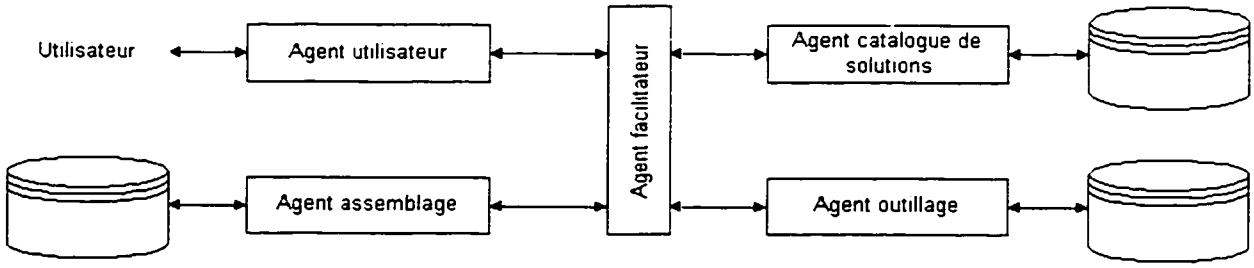


Figure 3.8 : Structure du système multi-agents pour le catalogue de solution d'assemblage

Analysons plus en détail l'organisation des connaissances des agents. Nous pouvons considérer chaque agent comme ayant un espace de connaissances $C_n(c_{n1},\ldots,c_{nk})$ – i.e. ce qu'il considère des vérités – où n représente chacun des agents et c_{nx} représente une vérité atomique pour cet agent. Or, à la suite de la réception d'ordres provenant de son « maître », l'agent possède des intentions atomiques i_{nx} à résoudre afin de solutionner une intention globale I_n. Il tentera alors de résoudre chacune des intentions atomiques grâce

aux informations se trouvant dans l'environnement *E* des agents. Ainsi, pour que la société puisse fournir l'expertise voulue afin de résoudre un problème donné, il faut que les connaissances de tous les agents *C* et que l'environnement *E* contiennent les vérités nécessaires.

Nous ne ferons pas référence à ce principe durant le restant de notre document puisque nous imposons un parcours prédéfini à notre agent afin de trouver une réponse et nous connaissons très bien la structure des intentions à résoudre. Cependant, dans une société évoluée, nous ne pouvons prédire qui aura l'expertise voulue, ni comment elle sera fournie. Une gestion efficace de l'ensemble des connaissances sera essentielle, d'où l'importance de l'utilisation d'une ontologie commune.

Notons un aspect particulier de cette théorie. Nous estimons dans le cadre de notre travail que chaque agent possède uniquement des vérités. Or, il pourrait certainement arriver des cas où certains agents croient posséder la vérité sans pour autant l'avoir (par exemple, une erreur se glissant dans les données d'un agent). Il faudra donc considérer l'aspect de vérité hypothétique – et même de confiance à un agent – lorsqu'il sera impossible aux agents de certifier les connaissances des autres agents de la société.

Les sections suivantes décrivent en détail les différentes caractéristiques de chaque agent et montrent ensuite le diagramme logique que chacun possède.

3.4.1 Agent facilitateur

L'agent facilitateur est l'agent central de cette structure. Cet agent possède les caractéristiques suivantes :

Présence : il est présent sur chaque ordinateur ;

Sécurité : il est en charge de la sécurité de l'ordinateur. Il peut donc refuser l'entrée d'un agent sur le système si ce dernier ne possède pas l'autorisation nécessaire ;

Persistance : il est toujours éveillé ;

Connaissances : il connaît l'expertise de chaque agent sur sa machine. Il connaît également, via l'agent facilitateur maître, l'emplacement de tous les autres agents gestionnaires de sa société ;

Mobilité : il est en charge du transport des agents d'une machine à l'autre. Il transfère le code de l'agent sur l'autre machine du réseau en communiquant des informations d'entrée au facilitateur hôte ;

Messages : il connaît le port de communication de chaque agent sur sa machine. Ainsi, lorsqu'un agent désire envoyer un message à son confrère, il lui fournit l'information nécessaire pour communiquer avec lui.

La figure 3.9 montre le diagramme logique de l'agent facilitateur. Il s'agit d'un raisonnement basé sur celui présenté par Wayner (1995).

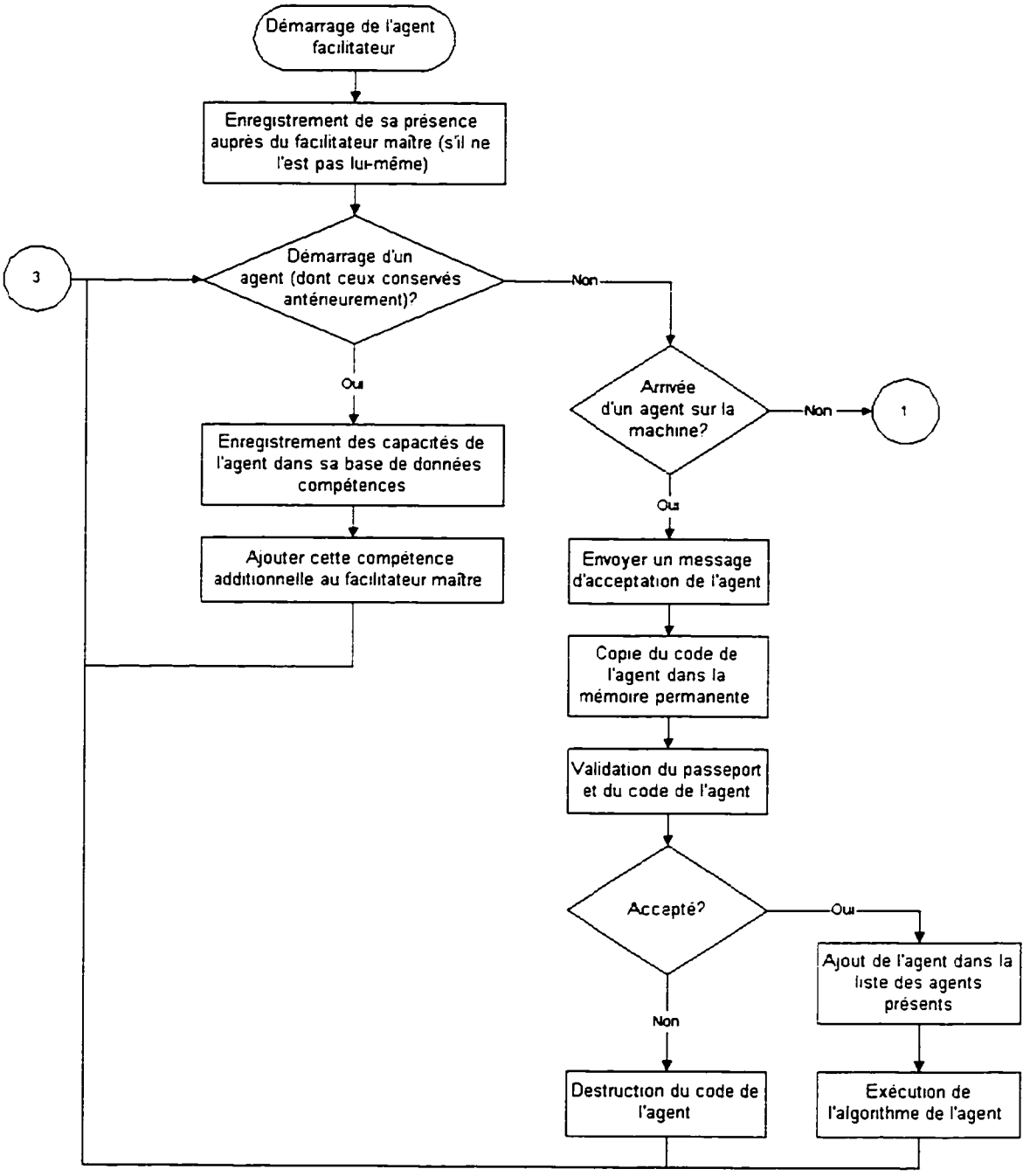


Figure 3.9 : Diagramme logique de l'agent facilitateur (partie 1)

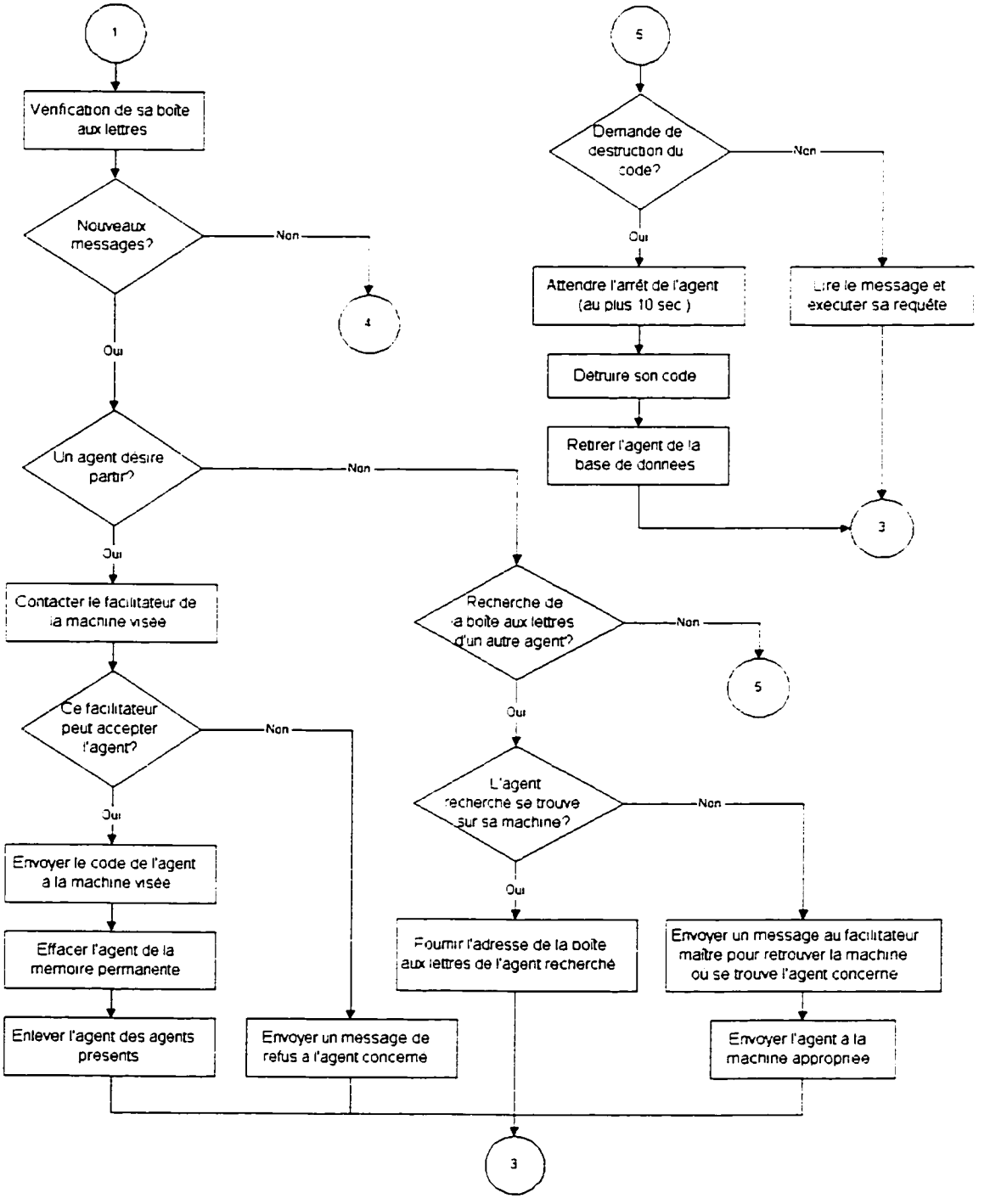


Figure 3.10 : Diagramme logique de l'agent facilitateur (partie 2)

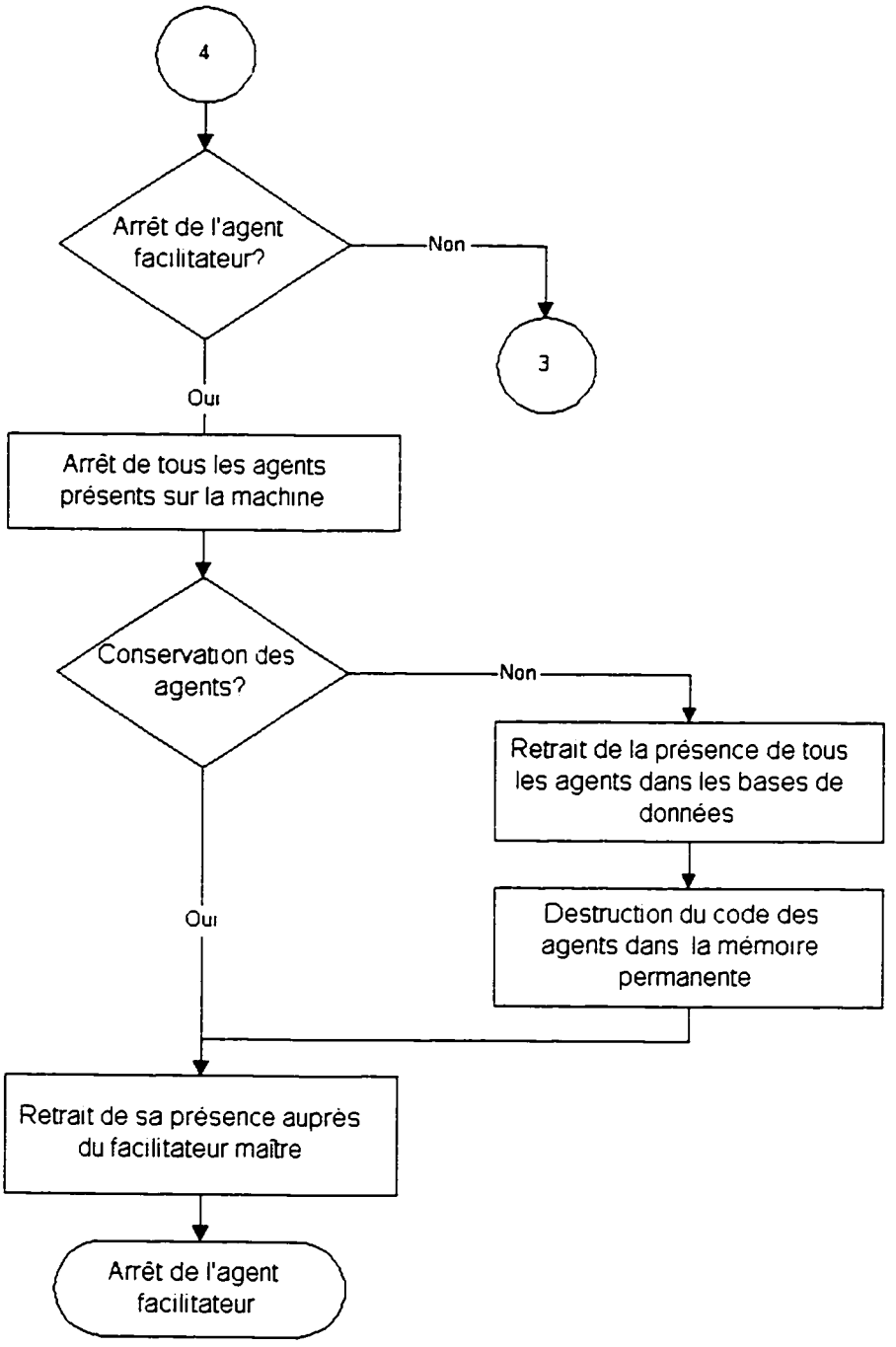


Figure 3.11 : Diagramme logique de l'agent facilitateur (partie 3)

Notons qu'un agent localisé sur une machine peut être endormi, i.e. pas en mode d'exécution. Ainsi, lorsque le facilitateur détecte que ce dernier a reçu de nouveaux messages, il l'exécute de nouveau à partir de l'item 3 de la figure 3.9.

Explicitons plus en détail les différentes actions qu'effectue l'agent facilitateur.

Enregistrement des agents : Les capacités des agents particuliers sont enregistrées dans une base de connaissance et partagées avec l'agent facilitateur maître, permettant ainsi de localiser l'emplacement exact d'un agent sur le réseau. Ceci permet de déplacer l'agent demandeur sur la même machine que l'agent possédant l'expertise. Dans le cas où les échanges afin d'obtenir une réponse sont nombreux, cet méthode diminue habituellement le trafic sur les réseaux.

Validation des agents reçus : Afin d'augmenter la sécurité, chaque agent doit posséder un passeport et un code valable avant d'être exécuté. Ainsi, une des tâches de l'agent facilitateur est de participer à cette validation.

Activation des agents : En réponse à une validation de l'agent, ce dernier est exécuté par l'agent facilitateur. Un agent peut également être exécuté s'il s'est « endormi » et reçoit de nouveaux messages. Dû à l'environnement multitâche dans lequel s'exécutent les agents, la seule intervention nécessaire de la part de l'agent facilitateur sur l'exécution d'un agent, suite à son activation, est sa destruction.

Envoi et réception des agents : L'agent facilitateur participe au transfert des agents de sa machine à une autre. Ainsi, il communique avec les agents facilitateurs

localisés sur les autres machines et copie le code des agents. La destination de l'agent peut être précisée ou non. Dans le cas où l'agent désirerait rencontrer un agent particulier, tel l'agent catalogue de solutions d'assemblage, l'agent facilitateur communique avec l'agent facilitateur maître pour trouver la machine où il se trouve puis transfère l'agent en question.

Arrêt des agents : Selon la méthode d'arrêt voulue, les agents peuvent être conservés ou non en mémoire. Ainsi, dans les circonstances d'une panne, les agents seront conservés pour ensuite être exécutés de nouveau une fois la panne réparée. Dans le cas d'un arrêt normal, tous les agents sont détruits s'ils n'ont pas d'autres tâches à effectuer ailleurs.

3.4.2 Agent utilisateur

L'agent utilisateur est l'interface entre l'utilisateur et la société d'agents. Lorsqu'un utilisateur désire obtenir une réponse à une solution du catalogue d'assemblage, il transige avec cet agent. Ainsi, toute interaction avec l'usager s'effectue par l'intermédiaire de cet agent. L'agent utilisateur n'est pas nécessairement unique à chaque employé, car dans le système que nous concevons, il n'y a pas de conservation d'expertise personnelle de la part de cet agent.

Il est donc en charge de :

- recueillir l'information de l'utilisateur ;
- formuler des requêtes aux autres agents et recueillir les résultats ;
- présenter à l'utilisateur les résultats.

Les figures 3.12 et 3.13 suivantes montrent le diagramme logique de l'agent utilisateur.

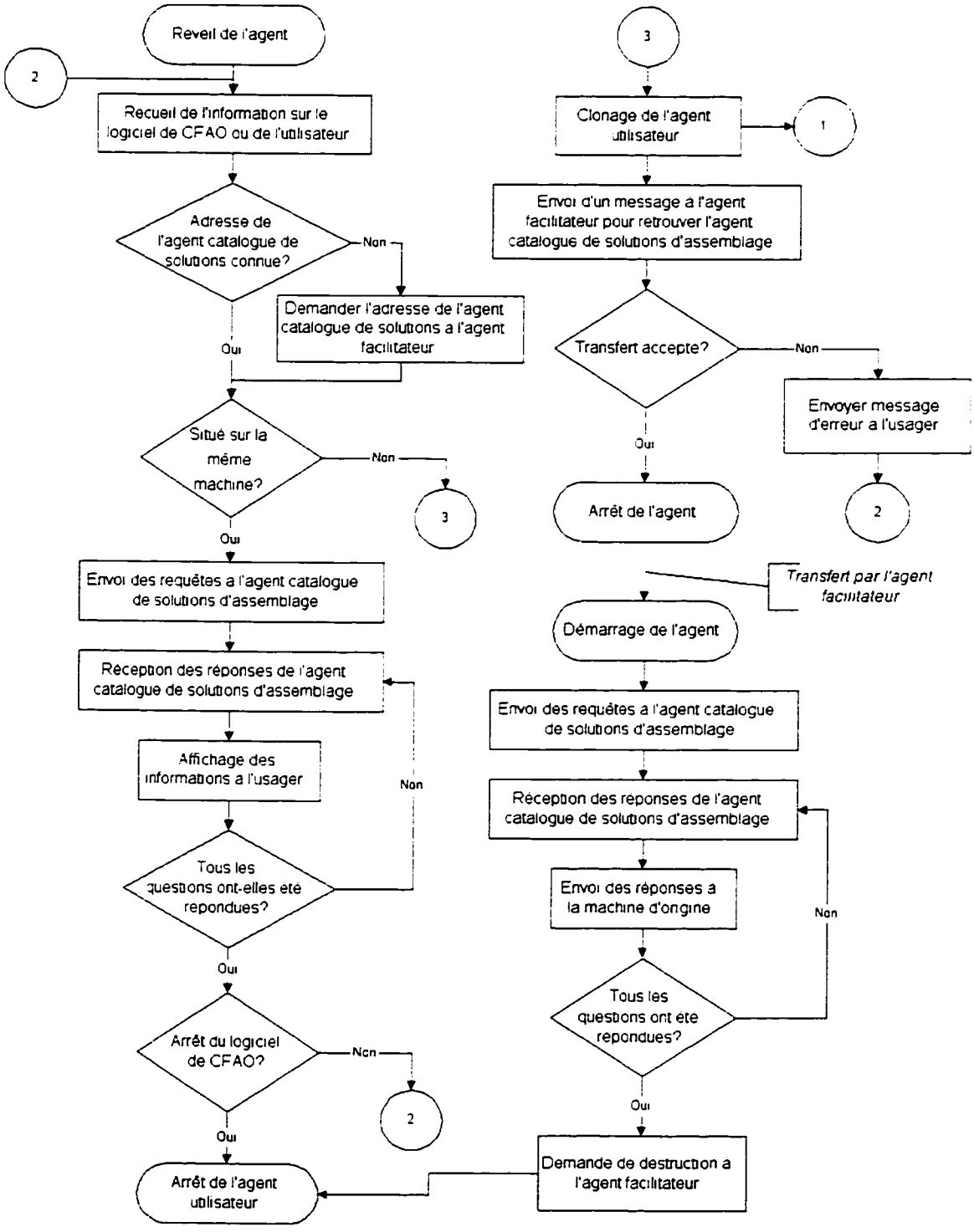


Figure 3.12 : Diagramme logique de l'agent utilisateur (partie 1)

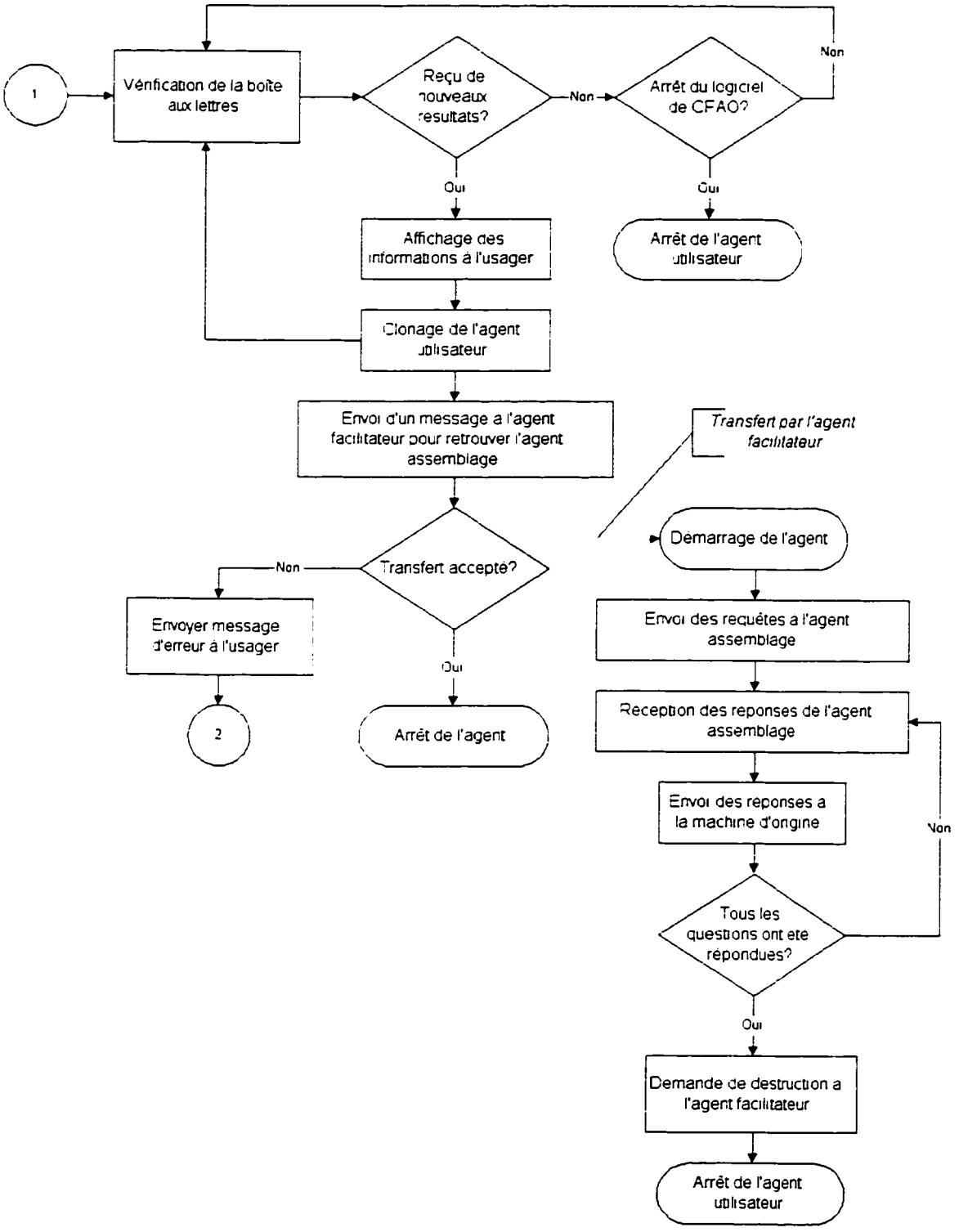


Figure 3.13 : Diagramme logique de l'agent utilisateur (partie 2)

Décrivons en plus de détails les différentes actions qui doivent être effectuées par l'agent utilisateur :

Recueil de l'information : L'agent utilisateur questionne le logiciel de CFAO. Toutes les informations additionnelles nécessaires sont obtenues directement par une interface usager.

Recherche de l'agent catalogue de solutions d'assemblage : L'agent utilisateur (AU) doit maintenant interagir avec l'agent catalogue de solutions d'assemblage (ACS). Pour faire ceci, il doit connaître son emplacement physique sur le réseau. Suite à une requête avec l'agent facilitateur, si l'AU détermine que l'ACS n'est pas sur la même machine, il se clone et envoie sa nouvelle copie rencontrer l'ACS. Sinon, il transige directement avec lui sur la machine actuelle.

Envoi de messages : Suite à la réception de chaque réponse de la part de l'agent catalogue de solutions d'assemblage, l'agent utilisateur envoie un message à son clone situé sur la machine mère. Ainsi, les réponses sont affichées de manière asynchrone à l'usager.

Ordonnancement des solutions : Chaque nouveau message reçu par l'agent utilisateur peut influencer les résultats affichés à l'écran. Ainsi, suite à une requête avec d'autres agents, l'agent utilisateur peut changer l'ordre d'affichage des solutions puisque certaines sont devenues prioritaires. Ce changement

survient habituellement à la suite des réponses obtenues par les agents machinerie et assemblage.

3.4.3 Agent catalogue de solutions d'assemblage

Une fois l'information recueillie de l'agent utilisateur, une requête est envoyée à l'agent catalogue de solutions. Ce dernier est en charge de la gestion de cette information et de toute requête demandée. Il est la seule manière d'accéder à l'information contenue dans la base de données du catalogue. De plus, il gère la sécurité en autorisant ou non la modification ou l'accès à l'information. Une fois une requête admissible reçue, il filtre au maximum les solutions admissibles, puis il communique avec l'agent outillage pour obtenir les caractéristiques concernant l'outillage (dimensions, précision, etc.).

La figure 3.14 montre le diagramme logique de l'agent catalogue de solutions d'assemblage. Ce diagramme d'allure simple permet d'obtenir un résultat robuste. Précisons que l'agent catalogue de solutions d'assemblage gère tous les accès à ses données.

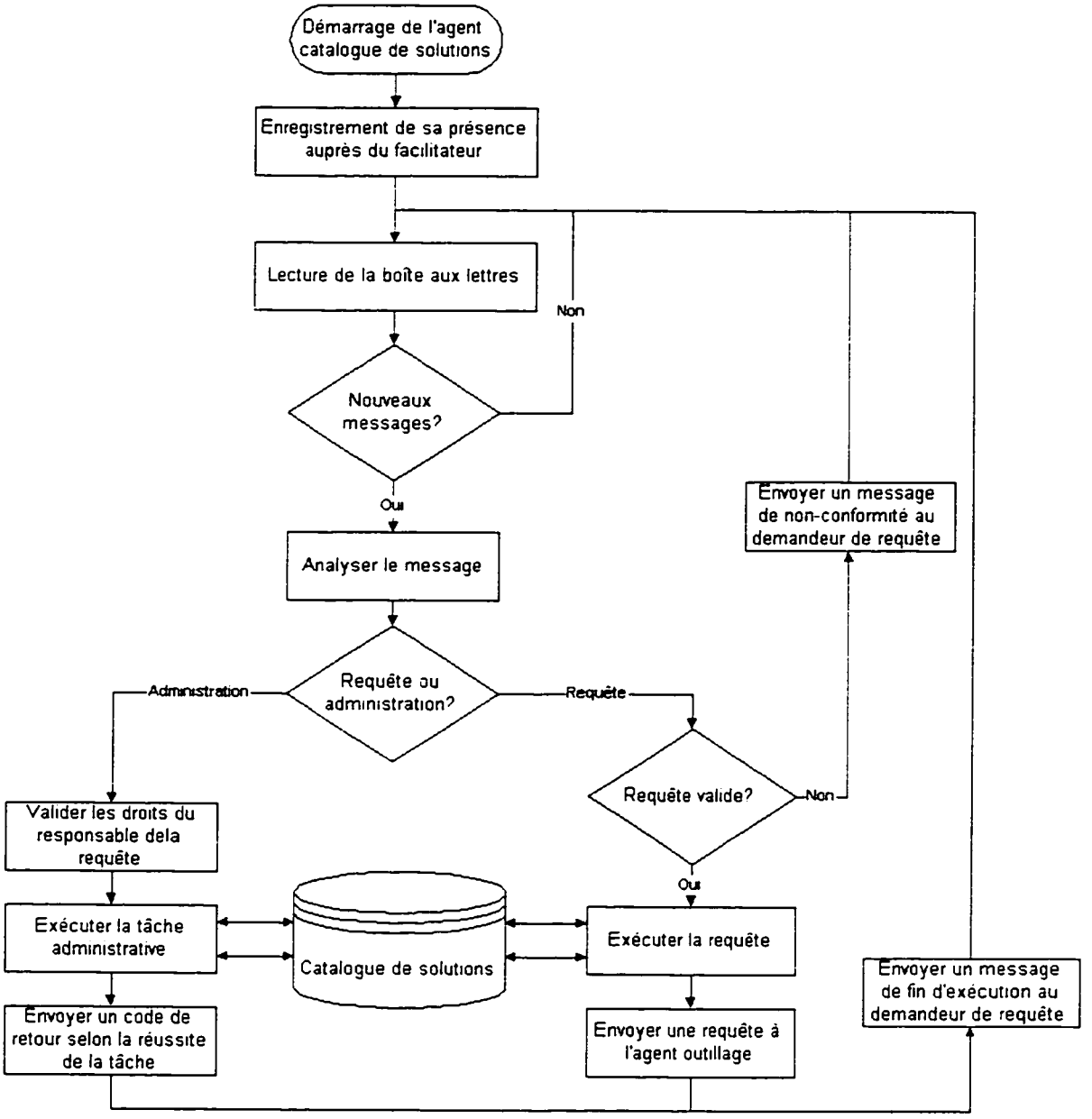


Figure 3.14 : Diagramme logique de l'agent catalogue de solutions d'assemblage

Notons que cet agent effectue la requête à la place de l'agent utilisateur. Ceci apporte plusieurs avantages. D'abord, ceci évite à l'agent utilisateur de recueillir de l'information supplémentaire et d'engorger sa mémoire. Ensuite, ceci simplifie les tâches que doit

effectuer l'agent utilisateur dû à la création d'un « partenariat » entre les agents catalogue de solutions d'assemblage et outillage, ce dernier agissant un peu comme « sous-traitant ». Nous voyons ici apparaître certaines caractéristiques d'une société évoluée.

3.4.4 Agent outillage

L'agent outillage, pour sa part, gère l'information concernant tous les outils. L'outillage est séparé de la solution d'assemblage pour plusieurs raisons :

- La séparation des ressources des gammes et des méthodes permet une meilleure gestion de l'information (tel que présenté dans LOGAM'97) ;
- Au point de vue de la flexibilité, il est préférable de les séparer, car, éventuellement, il serait utile d'utiliser l'information de l'outillage pour d'autres applications et cet expert (l'agent) sera nécessaire. On peut penser aux machines à commandes numériques, aux futurs agents pouvant gérer un poste d'assemblage, aux personnes en charge des achats, etc. ;
- La séparation du matériel et des méthodes est nécessaire. Si, du jour au lendemain, le catalogue de solutions d'assemblage disparaît ou est remplacé, l'agent outillage doit toujours exister pour fournir son expertise dans la société croissante. N'oublions pas que les agents forment une société et que l'expertise fournie par chacun est nécessaire au bon fonctionnement.

La figure 3.15 montre le diagramme logique de l'agent outillage. Comme nous pouvons le constater, cette structure est très similaire à celle montrée à la figure 3.14. Ceci démontre un gros avantage des systèmes multi-agents, soit la programmation modulaire, permettant de réutiliser du code existant. Ainsi, nous pouvons supposer qu'une partie importante du code de l'agent catalogue de solutions d'assemblage sera très similaire à celui de l'agent outillage.

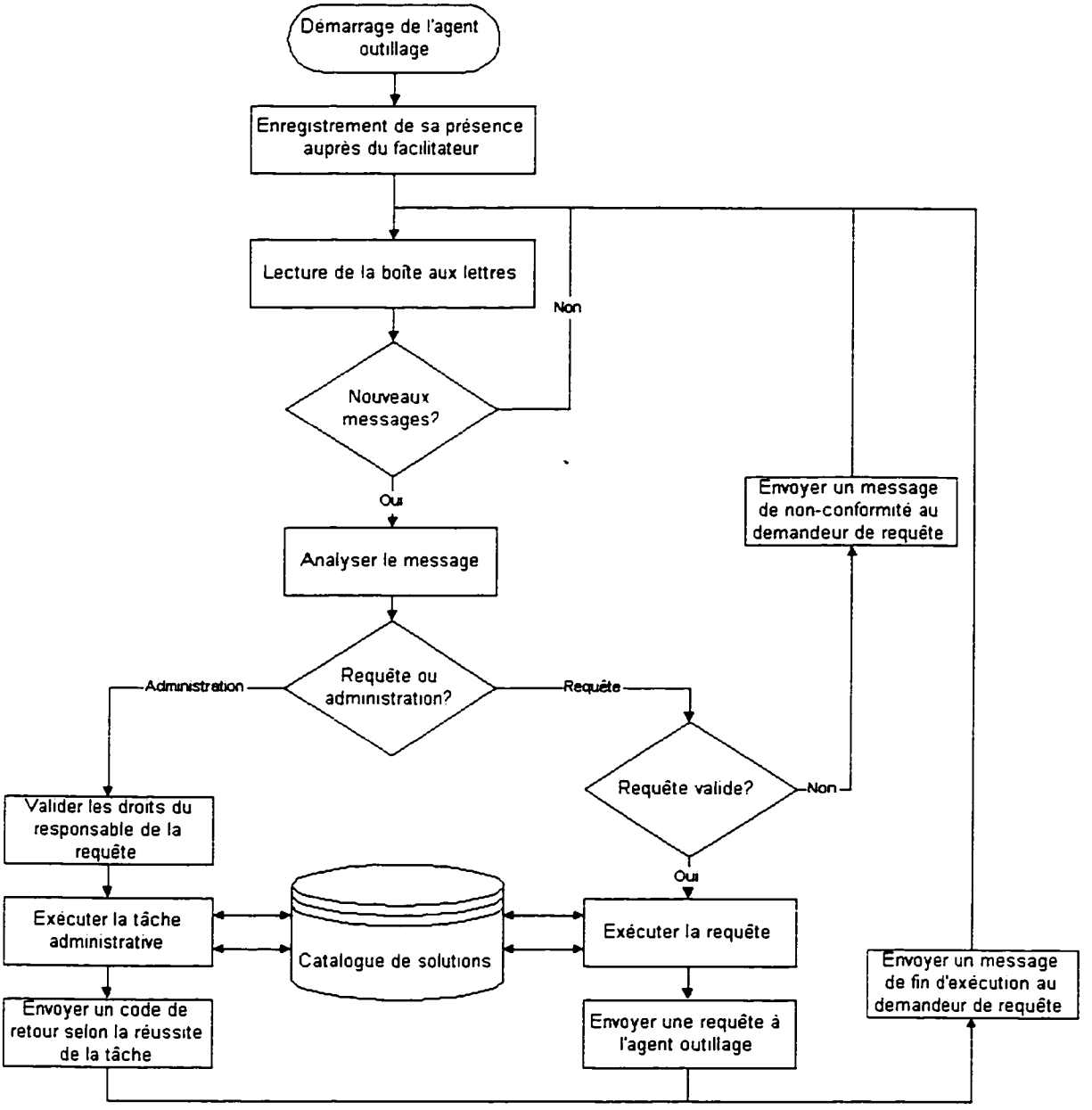


Figure 3.15 : Diagramme logique de l'agent outillage

3.4.5 Agent assemblage

L'agent assemblage possède toutes les connaissances portant sur l'utilisation de la machinerie d'assemblage sur le plancher. Il connaît ainsi les échéanciers de tous les outils

en cours d'utilisation. Ceci permet donc de prévoir un manque de matériel potentiel, et de choisir par conséquent une alternative.

La figure 3.16 montre le diagramme logique d'un tel agent. Ce dernier possède également une grande ressemblance aux agents montrés aux figures 3.14 et 3.15 et sera ainsi relativement simple à créer et à implanter.

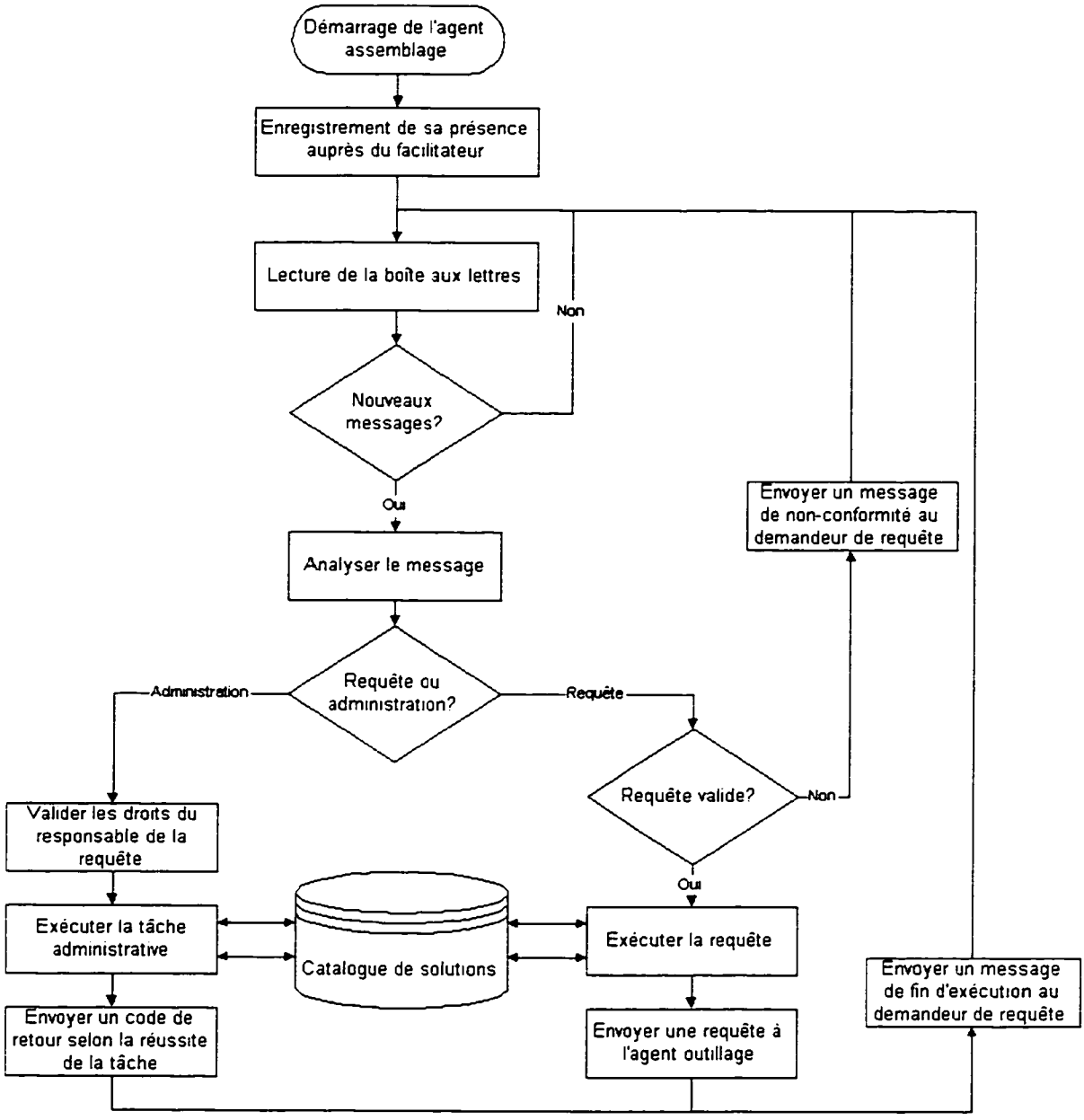


Figure 3.16 : Diagramme logique de l'agent assemblage

3.4.6 Mise en situation

Analysons maintenant un scénario typique de recherche dans le catalogue de solutions d'assemblage proposé ici.

1. L'agent utilisateur acquiert de l'information concernant l'assemblage à résoudre, soit le type d'assemblage, la signature d'assemblage, les dimensions principales du composant inséré et les temps de cycle à respecter.

2. Comme suite à l'acquisition de cette information, l'agent utilisateur questionne l'agent facilitateur afin de connaître l'adresse de l'agent catalogue de solutions d'assemblage.

3. L'agent utilisateur envoie un message à l'agent catalogue de solutions d'assemblage afin que ce dernier trouve des solutions d'assemblage pouvant résoudre l'assemblage rencontré.

4. L'agent catalogue de solutions d'assemblage interroge l'agent facilitateur afin qu'il lui fournisse l'adresse de l'agent outillage.

5. L'agent catalogue de solutions d'assemblage envoie un message à l'agent outillage pour déterminer si, en fonction des solutions retenues, l'outillage utilisé a les capacités de manipuler le composant assemblé.

6. Pendant ce temps, l'agent catalogue de solutions d'assemblage envoie une réponse de requête à l'agent utilisateur. Cependant, il est à noter que cette réponse peut être incomplète si l'agent outillage n'a pas eu le temps de recueillir l'information nécessaire sur les outils recherchés. L'agent outillage enverra son information dès qu'il le pourra. L'agent utilisateur a maintenant la tâche de présenter à l'utilisateur les différentes solutions de manière claire et ordonnée.

7. L'agent utilisateur peut maintenant questionner l'agent assemblage pour connaître la disponibilité de l'outillage. Cette disponibilité peut s'associer à des critères de date de livraison à laquelle les composants sont requis, afin de connaître les assemblages non réalisables dus à de l'outillage non disponible.

La figure 3.17 montre, en résumé, la communication ayant lieu entre les agents. Les nombres indiqués représentent le numéro de la description ci-dessus.

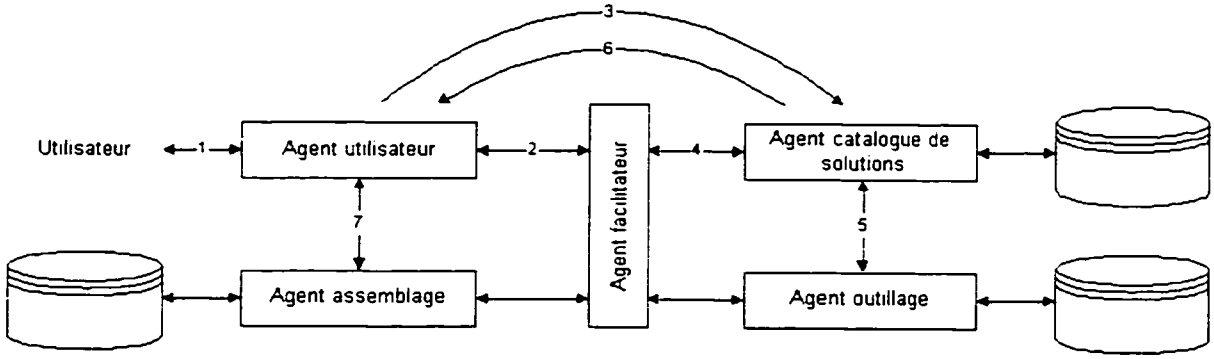


Figure 3.17 : Communication entre les agents

Une fois toutes les solutions obtenues, elles sont classées en ordre de priorité selon :

- le coût ;
- le temps de cycle de la solution ;
- l'appréciation de la solution.

Il s'agit donc d'offrir au gammiste la solution offrant la meilleure valeur, en lui évitant de prendre des décisions sur une expertise qu'il ne possède pas nécessairement.

3.5 STRUCTURE DE L'INFORMATION D'ASSEMBLAGE DANS LES BASES DE DONNÉES

La structure de l'information dans le catalogue d'assemblage est un aspect important dans la recherche des solutions et de la technologie utilisée. Nous allons décrire dans cette section la structure de l'information dans toutes les bases de données de notre système.

Cette structure fait référence au sous-chapitre 3.4, car chaque agent possède sa propre information.

Tel que décrit à la section 3.4, notre système possède différents agents, dont certains possèdent des bases de données particulières. Les agents de catalogue de solutions d'assemblage et de machinerie forment, à eux deux, le catalogue tel que nous l'avons décrit tout au long de ce document. L'information a été séparée afin d'améliorer la recherche de l'information et de permettre une expertise distribuée dans la société d'agents proposée. En effet, en permettant une recherche asynchrone de l'information, le temps de recherche sera amélioré (Chess, N/D), surtout si le système est implanté sur un grand réseau, où le trafic peut ralentir le transfert d'information. Ainsi, les agents demandant de l'information n'auront qu'à transmettre une unique requête, grâce au langage KQML (Finin, Fritzson, McKay et McEntire, 1994) ; ils pourront toujours continuer à fonctionner. En contrepartie, ceux offrant de l'information transmettront cette information dès que possible.

À un certain égard, la structure proposée s'apparente à la structure de LOGAM'97, en subdivisant l'information en trois catégories distinctes : les ressources, les méthodes et les gammes. Dans le contexte de ce travail, les ressources représentent les machines accessibles et tous les outillages. La fonction des méthodes est, pour sa part, remplie par le catalogue de solutions d'assemblage. C'est à cet endroit qu'est conservée l'expertise proprement dite de la compagnie. Or, contrairement à LOGAM, l'information concernant les gammes n'est pas conservée dans notre structure. Il s'agit d'information que bâtit lui-même le gammiste.

3.5.1 Agent catalogue de solutions d'assemblage

L'agent du catalogue de solutions d'assemblage doit conserver l'information des différentes solutions d'assemblage retenues par la compagnie. Il conserve également, pour chaque solution, la différente machinerie et les accessoires pouvant être utilisés.

Notons que dans cette structure, l'aspect de sécurité n'a pas été implanté. Cependant, il ne nécessitera que l'ajout d'une table avec une liste d'employés et leurs droits d'accès.

Voici la structure utilisée dans notre système (figure 3.18).

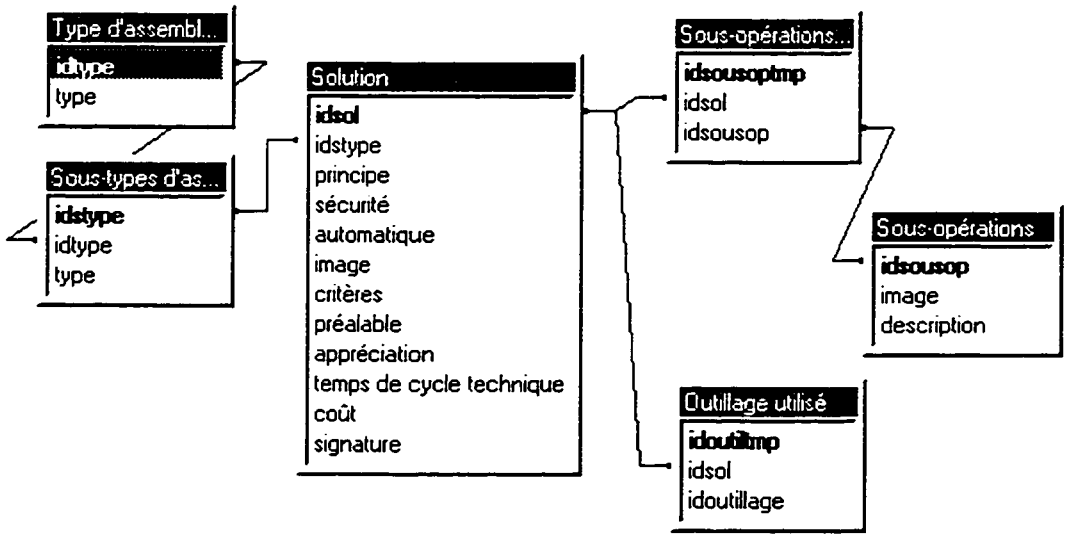


Figure 3.18 : Structure relationnelle de l'information contenue par l'agent catalogue de solutions d'assemblage

Les différentes tables ont pour but d'utiliser à leur pleine capacité les propriétés des BDR. En effet, en créant une table indépendante des types d'assemblages, nous réussissons à diminuer le nombre de répétitions inutiles, ainsi qu'à accélérer la recherche d'un certain type d'assemblage en utilisant la propriété de jointure de la théorie relationnelle.

3.5.2 Agent outillage

L'agent outillage, pour sa part, gère l'information concernant tous les outillages de la compagnie. Il connaît ainsi toutes les caractéristiques de chaque outil et peut aider les autres agents à déterminer les outils adéquats pour la résolution d'un problème d'assemblage. Il contient également de l'information supplémentaire sur l'outillage, tel que le manufacturier avec ses coordonnées.

Voici la structure d'information utilisée dans le cas de notre système (figure 3.19).

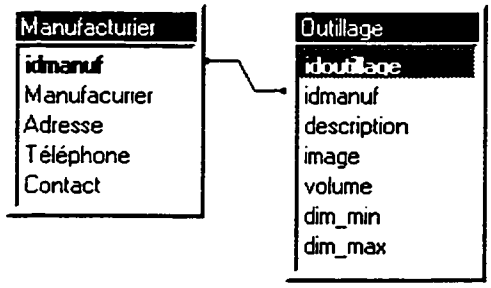


Figure 3.19 : Structure relationnelle de l'information contenue par l'agent outillage

3.5.3 Agent assemblage

L'agent assemblage permet de fournir de l'information sur l'utilisation de la machinerie. Donc, nous pouvons déterminer si un outil est disponible, de même que la date où il le deviendra. En connaissant la date de livraison du produit assemblé, il est possible de trouver le meilleur outillage à utiliser.

Une seule table compose cet agent. Elle est présentée ci-dessous (figure 3.20).

Figure 3.20 : Information contenue par l'agent assemblage

Afin de bien lier l'information entre les agents, une information unique permet d'identifier les éléments importants. Dans le cas de l'agent catalogue de solutions d'assemblage, il s'agit de **idoutillage**. Cette identification représente d'une manière unique un outil, quel qu'il soit. Lors de la communication entre les agents, cette identification est fournie pour effectuer la requête voulue.

Rappelons que, quelque soit le GBD utilisé pour gérer les informations, le JDBC permet un accès relativement aisé aux données. Il s'agit donc d'une raison de plus pour laquelle le système proposé utilise le langage Java (1996).

3.6 ANALYSE

Nous avons analysé au cours de ce rapport une manière permettant de conserver les solutions d'assemblage et de les rendre accessibles aux gammistes. Le but premier est de conserver le savoir-faire d'une entreprise afin d'éviter le cas où l'expertise disparaît avec le départ d'une ressource humaine. Bien sûr, cette méthode nécessite un changement de la méthode de travail et de la mentalité des gens dans l'entreprise, tâche plutôt difficile à accomplir.

Le premier aspect à remarquer est que le choix dans la conservation du savoir-faire s'effectue à un haut niveau départemental et non au niveau des employés. Ceci veut dire que les solutions rapportées dans le catalogue de solutions doivent faire suite à des consultations entre divers départements de la compagnie, soit les achats, la fabrication, l'ingénierie et bien sûr l'assemblage, tel qu'incité par le concept d'ingénierie simultanée. Tous ces départements doivent fournir un effort pour s'assurer que les solutions répondent à toutes leurs exigences. En effet, il a été prouvé que la quantité d'erreurs de conception peut être réduite à un très bas niveau si tous les intervenants communiquent entre eux (Cambron, 1996). Puisque les solutions retenues dans le catalogue proviennent de consultations entre divers intervenants – l'équipe multidisciplinaire –, les solutions ont plus de chances de répondre aux exigences de tous.

Il aurait été possible, grâce à la technologie multi-agents, de créer un système où l'expertise de chaque employé est emmagasinée. Plusieurs prototypes dans divers domaines de recherche ont été dirigés dans ce sens, où certains agents experts de la société d'agents sont fréquemment consultés. Cependant, il est dangereux dans le domaine de l'assemblage de ne faire appel qu'aux méthodes intuitives des experts dans les prises de décisions. En effet, les décisions prises par les experts ne reposent pas nécessairement sur des fondements solides et le système aurait favorisé ces solutions, alors qu'il ne s'agit pas du savoir-faire que les compagnies désirent conserver. Il faut s'assurer que les solutions répondent aux conditions techniques et économiques imposées (Warnecke, Lörh et Kiener, 1980) et c'est justement l'équipe multidisciplinaire mentionnée ci-dessus qui répond à ce besoin.

Nous pouvons remarquer que le catalogue de solutions d'assemblage peut s'adapter à n'importe quel type de problème d'assemblage, quel que soit le type d'assemblage rencontré. L'importance réside entre autres dans la décomposition possible d'un problème d'assemblage. Dans le contexte que nous avons analysé ici, seule la solution de l'assemblage proprement dite a été intégrée, mais il ne faut pas oublier que d'autres fonctions partielles devront également faire partie du catalogue de solutions. Entre autres, nous retrouvons tous les aspects entourant le classement des composants, dont l'utilisation des bols vibrants associés aux sélections appropriées. D'autres algorithmes devront être développés dans ce domaine pour faciliter leur choix.

Bien que les gammistes utiliseront ce système, il faut toutefois encourager son utilisation par les concepteurs (Boothroyd et DeWhurst, 1983). De cette manière, ces derniers pourraient favoriser des formes et des contacts spécifiques en vue d'assembler les composants en y voyant un avantage énorme sur la facilité d'assemblage.

La méthode de signature d'assemblage développée limite cependant la portée de l'application du catalogue en le limitant principalement à l'insertion. Bien que le champ de recherche soit restreint – en se limitant à l'insertion –, ce domaine représente une vaste proportion des assemblages rencontrés, d'où sa pertinence comme sujet traité. Il faudrait cependant, dans le cadre d'un travail ultérieur, tenter de répertorier tous les types d'assemblages pour lesquels cette méthode peut s'appliquer.

Le concept de signature d'assemblage possède une seconde limite importante. En effet, cette méthode ne considère que le résultat statique final de l'assemblage. Afin de pouvoir apprécier l'assemblage dans son aspect global, il faudrait également prendre en compte

l'approche des composants. Il faudrait ainsi connaître la trajectoire des composants, les collisions possibles, les volumes nécessaires pour l'insertion, etc. La méthode nécessiterait donc des ajustements majeurs mais ce travail pourra être facilité grâce aux outils de simulation cinématique retrouvés dans plusieurs logiciels de CFAO, dont CATIA. Cependant, en ce qui concerne l'aspect final de l'assemblage, notre méthode semble remplir adéquatement son rôle. En effet, nous voyons qu'il est maintenant possible de filtrer des solutions basées sur leur géométrie, ce qui n'a jamais été effectué auparavant.

Or, la méthode de signature d'assemblage présente encore plusieurs inconnues. Il serait intéressant de faire ressortir les caractéristiques des assemblages en analysant uniquement la signature. Par exemple, y a-t-il une signature particulière pour les assemblages hyperstatiques ou les assemblages facilités par des chanfreins ? Serions-nous capable de déterminer automatiquement les pièces d'un assemblage complexe qui peuvent être désassemblées car elles possèdent une direction de retrait libre de toute collision ? Ou encore, existerait-il une méthode de signature d'assemblage pouvant considérer un assemblage d'une manière globale plutôt que locale ? Il est donc évident que cette méthode nécessite encore beaucoup de recherche afin de perfectionner son utilisation.

Outre la signature d'assemblage, nous remarquons que la résolution d'un problème d'assemblage repose énormément sur du cas par cas, dû au si grand nombre d'inconnues et de variables. Il est donc difficile de faire des choix éclairés et de proposer une solution unique. Pour ces raisons, ce projet remet une partie de la prise de décision au gammiste. En effet, tel que démontré tout au long de ce document, les systèmes d'intelligence

artificielle qui auraient pu aider s'appliquent mal et possèdent des limites importantes d'implantation dans ce domaine. Cependant, dans un avenir prochain, il sera probablement possible d'utiliser l'IA comme outil de reconnaissance de forme ou de conservation d'expérience, afin de trouver toutes les solutions possibles.

En ce qui concerne l'implantation d'un tel système dans un logiciel de CFAO, nous préférons le garder indépendant pour plusieurs raisons. Premièrement, la programmation du système en Java (Java, N/D) lui permet une transition aisée entre les différentes plates-formes. Il est donc insensé de se limiter à un seul logiciel sur une seule plate-forme. L'approche proposée consiste à programmer pour chaque logiciel de CFAO distinct un « pont » permettant son intégration avec le reste du système. Il est ainsi possible pour une compagnie de posséder plusieurs logiciels différents et de n'utiliser que des « ponts » différents pour se relier au système conçu. Ainsi, chaque « pont » sera programmé en Java, si nécessaire, en utilisant le langage de programmation propre à chaque logiciel de CFAO.

Concernant les systèmes multi-agents, nous remarquons que, dans le contexte actuel, ce système n'utilise pas nécessairement des agents proprement dits. En effet, la méthode présentement utilisée se compare surtout à de la programmation distribuée, i.e. une programmation mobile sur un réseau. Pour que l'agent soit un agent tel que défini précédemment (§2.5), il doit être apte à prendre lui-même des décisions. Or, pour que de telles décisions soient prises, deux différentes options peuvent être envisagées :

1. Programmer un certain algorithme génétique : systèmes experts, réseaux neuronnaux, etc. ;

2. Utiliser un langage et un mode de pensée évolués dans lesquels les caractéristiques telles la croyance, la connaissance, le désir, l'intention et l'obligation sont efficacement implantés (Wooldridge, 1994).

Mais tel que discuté dans ce travail, il y a encore plusieurs embûches à surmonter avant de pouvoir les implanter efficacement dans le domaine de l'informatique mais surtout dans celui de l'assemblage. Ainsi, le jour où des agents seront effectivement utilisés à leur plein potentiel, il devra y avoir une grande découverte dans le domaine de l'assemblage qui permettra d'y incorporer tous les facteurs décisionnels ainsi que toutes les contraintes géométriques possibles.

Cependant, le système mobile tel que montré présente quand même plusieurs avantages, même s'il ne possède pas d'intelligence proprement dite. En effet, en possédant une communication réseau avec ses fournisseurs et sous-traitants – Internet –, il est possible, dans un esprit de partenariat, de consulter leur équipement afin de trouver peut-être de meilleures solutions d'assemblages chez eux, soit sur le plan technique ou financier. Il s'agit tout simplement d'installer un agent facilitateur sur leur système. Une fois ce facilitateur identifié par les autres, la communication s'amorce sans difficulté et sans perturber l'état actuel du système.

Également, l'asychronicité de l'agent lui fournit des propriétés intéressantes. Il peut ainsi continuer à travailler même s'il est en attente d'une réponse. Dans un contexte où il est important de posséder des systèmes informatiques rapides et performants, la possibilité d'utiliser l'agent en attente de réponse peut permettre des économies de temps et d'argent. Cependant, une autre manière d'améliorer les performances de l'agent serait de le cloner.

En effet, le clonage permet d'obtenir une copie de l'agent et ainsi de lui permettre d'effectuer d'autres tâches. Une fois ces tâches terminées, l'agent est détruit.

Nous pouvons conclure que la viabilité d'un système multi-agents dans le domaine de l'assemblage ne peut être utilisée, pour l'instant, que pour sa mobilité et son asynchronicité, car les recherches dans le domaine de l'assemblage n'ont pu fournir des paramètres clairs et formels sur lesquels s'appuyer pour permettre à des systèmes informatisés de prendre des décisions autonomes.

CONCLUSIONS

Nous avons étudié, au cours de ce mémoire, l'intégration d'un système multi-agents à un catalogue de solutions d'assemblage. Après avoir étudié le processus actuel de recherche d'une solution d'assemblage, les systèmes CFAO modernes ainsi que l'intelligence artificielle, nous avons tenté d'intégrer ces systèmes afin de tirer profit de chacun de leurs avantages. Nous avons déterminé que l'utilisation d'un système multi-agents pour effectuer cette tâche aide le gammiste en lui offrant une gamme de solutions aptes à résoudre son problème d'assemblage, lui demandant ainsi moins de savoir-faire.

Nous avons également évalué les manières d'implanter un tel système. Avec le langage orienté objet Java, la programmation est relativement aisée, la conception est modulaire et le code généré peut s'exécuter sur une vaste gamme de plates-formes informatiques. Les systèmes multi-agents, dotés d'agents mobiles, permettent d'améliorer les performances des systèmes informatiques disposés en réseaux, car leur mobilité leur permet de s'exécuter directement à l'endroit où a lieu la requête.

Nous pouvons conclure que l'utilisation d'algorithmes d'intelligence artificielle est mal adaptée pour résoudre les problèmes d'assemblage. La raison principale provient du fait que l'assemblage est un domaine où une légère différence entre deux assemblages peut faire varier énormément le type de solution retenue. Puisque le nombre de variables dont il faut tenir en compte est très élevé, nous ne pouvons appliquer un algorithme, où il peut exister différentes solutions à un problème et où le choix de la meilleure solution est difficile, voire impossible à déterminer.

Cependant, le potentiel de cette technologie multi-agents est énorme, dû en grande partie à sa mobilité. Il y aura probablement beaucoup d'applications d'ici peu de temps, surtout avec l'explosion d'Internet. Également, avec la vaste gamme de logiciels disponibles, les multi-agents permettront sûrement d'enlever le fardeau d'apprentissage en apprenant les habitudes de l'utilisateur et en les exploitant en conséquence.

Les applications futures des technologies multi-agents sont bien plus nombreuses. Nous pouvons aisément trouver plusieurs autres applications dans le domaine des technologies distribuées. Par exemple, imaginez son potentiel sur un groupe de programmeurs désirant partager les versions les plus récentes de fragments de code ou voulant spécifier un routage particulier pour la vérification, mais ces derniers étant situés à des kilomètres de distance. Au sein d'une entreprise, il y a également une explosion de possibilités. En réponse au problème de quantité d'information sans cesse croissante, nous pouvons trouver un système multi-agents effectuant des recherches dans plusieurs domaines pour ensuite compiler ses résultats sous forme d'un rapport concis. Il s'agit d'une application idéale pour, par exemple, un dirigeant d'entreprise désirant obtenir automatiquement toute l'information nécessaire pour réaliser une réunion efficace (graphiques, prix à la bourse de certaines actions, etc.) et cela, quelques minutes avant cette réunion.

Notons que le travail d'un gammiste ne s'arrête nécessairement pas au choix de la solution d'assemblage. En effet, dans le cas de produits automatisés, il faut également prévoir la structure de l'atelier d'assemblage, i.e. l'emplacement et l'ordre de chaque machine. Il reste donc encore énormément de défis à surmonter dans ce domaine avant que nous

puissions affirmer que le domaine de l'assemblage est aussi avancé techniquement que le domaine de la conception.

En ce qui concerne l'utilisation des signatures d'assemblage, cette nouvelle approche montre du potentiel en ce qui concerne la reconnaissance des assemblages. Or, sans la caractéristique d'unicité, la signature d'assemblage ne servira que de filtre amélioré de solutions d'assemblage et ne sera valable que pour les contacts surfaciques entre les éléments assemblés.

L'automatisation des tâches prendra sûrement de plus en plus de place dans la vie d'un ingénieur, aussi bien au niveau de la conception d'un produit qu'au niveau de l'assemblage. Nous pouvons aisément voir l'utilité d'une application permettant de naviguer au travers des conceptions antérieures de l'entreprise afin de trouver une conception similaire à celle présentement créée. Il est donc possible d'éviter les pertes de temps dues à un dédoublement de conception inutile et laisser ainsi chacun des coéquipiers – humain et ordinateur – exécuter la tâche qu'il sait le mieux faire, soit créer et calculer respectivement.

BIBLIOGRAPHIE

ARCAND, J-F. (N/D). Un agent intelligent pour la supervision de tâches dans un interface humain-ordinateur, Centre d'innovation en technologies de l'information (CITI), Québec, Canada.

ASDESIGN (1996). Assembly Design, CATIA Reference Manual, V.4.1.6.

ATIYEH, P. G. (1992). Design For Assembly : Sometimes More Is Less, Assembly Automation, Vol. 12, No. 2, 1992, 26-30.

ATZENI, P., DE ANTONELLIS, V. (1993) Relational Database Theory, The Benjamin/Cummings Publishing Company, Californie, 387 p.

BALAZINSKI, M. (1994). Logique floue en fabrication, École Polytechnique de Montréal, Département de fabrication.

BEDWORTH, D. D., HENDERSON, M. R., WOLFE, P. M. (1991). Computer-integrated Design and Manufacturing, McGraw-Hill, New York, 238.

BELGRAVE, M. (1995). The Unified Agent Architecture: A White Paper, Université McGill. (http://www.ee.mcgill.ca/~belmarc/uaa_paper.html)

BOOTHROYD, G., DEWHURST, P. (1983). Design for Assembly - A Designer's Handbook, University of Massachusetts, Department of Mechanical Engineering, Amherst, MA.

BROOKE, L. (1991 Sept.). Think DFD !, Automotive Industries, 71-73.

CAMBRON, M. (1996 Hiver). Ingénierie simultanée, cours MEIN601 de cycle supérieurs, École Polytechnique de Montréal.

CARRICO, M. A., GIRARD, J. E., JONES, J. P. (1989). Building Knowledge Systems, Intertext Publications, McGraw-Hill Book Company, New York, 335 p.

CHAN, D. S. K., MO, J. P. T. (1993). Transformation of Product Design Features for Assembly Analysis, Royal Melbourne Institute of Technology, 1993 International Conference on Assembly, 22-24 November, Adelaide, Australie, 55-60.

CHESS, D., *et al.* (N/D). Itinerant Agents for Mobile Computing, IBM T.J. Watson Research Center, Yorktown Heights, New York.

CKE (N/D). CATIA Knowledge Engineering, doc. #USMCKE18, Dassault Systèmes, France.

DE FAZIO, T.L., *et al.* (1991). A Prototype of Feature-based Design For Assembly, The Charles Stark Draper Laboratory, Inc., Cambridge, MA.

ELMASRI, R., NAVATHE, S. B. (1989). Fundamentals of Database Systems, The Benjamin/Cummings Publishing Company, Californie, 802 p.

ETTLIE, J.E., STOLL, H.W. (1990). Managing the Design-manufacturing Process, McGraw-Hill Engineering and Technology Management Series, New York, 79-87.

FAN, Ip-Shing. (N/D). Design for Manufacture and Assembly in Concurrent Engineering, The CIM Institute, Cranfield University, United Kingdom.

FERNANDEZ, L. (1993). When Part Count Reduction Goes Too Far, Machine Design, December 10, 92-93.

FININ, T., FRITZSON, R., MCKAY, D., MCENTIRE, R.. (1994). KQML - A Language and Protocol for Knowledge and Information Exchange, Computer Science Department of the University of Maryland and Valley Forge Engineering Center of the Unisys Corporation.

FIREFLY (1996). http://www.firefly.com : Logiciel multi-agent pour déterminer les goûts musicaux et cinématographiques de personnes.

GABRIELE, G. A. (N/D). The Application of Design for Assembly Principles to the Design of Aerospace Structures, Rensselaer Polytechnic Institute, Department of Mechanical Engineering, Aeronautical Engineering and Mechanics, Troy, New York.

GARDAN, Y., MINICH, C. (N/D). La modélisation géométrique et l'extraction de caractéristiques de forme, Laboratoire de recherche en informatique de Metz, UFR MIM, Ile du Saulcy, Metz.

GENESERETH, M.R., FIKES, R.E. (1992 Juin). Knowledge Interchange Format, Version 3.0, Reference Manual, Computer Science Department, Stanford University, Californie.

GÓMEZ-PÉREZ, A. (1994). From Knowledge Based Systems to Knowledge Sharing Technology : Evaluation and Assessment, Knowledge Sharing Laboratory, Stanford University, Californie.

GRENIER, B. (1989). Méthodologie du développement industriel d'un produit, Techniques de l'ingénieur, Cahier T100.

HALE, M. A., CRAIG, J. I., MISTREE, F., SCHRAGE, D. P. (N/D). On the Development of a Computing Infrastructure that Facilitates IPPD from a Decision-Based Perspective, Georgia Institute of Technology, Atlanta, Géorgie.

HARMON, P., MAUS, R., MORRISSEY, W. (1988). Expert Systems : Tools and Applications, John Wiley & Sons Inc., New York, 289 p.

HARRISON, C.G., CHESS, D.M., KERSHENBAUM, A. (1995 Mars). Mobile Agents : Are they a good idea ?, IBM T.J. Watson Research Center, New York.

IGNIZIO, J. P. (1991). Introduction to Expert Systems, McGraw-Hill Inc., New York, 1991, 402 p.

JAVA (1996). http://java.sun.com

JONES, R. E., WILSON, R. H. (1996). A Survey of Constraints in Automated Assembly Planning, Proc. 1996 IEEE Intl. Conf. On Robotics and Automation, 1525-1532.

KIM, W. (1990). Introduction to Object-Oriented Databases, The MIT Press, Cambridge, Massachusetts.

KOBE, G. (1994 Mai). Beyond DFA, Automotive Industries, 64.

LACHKARI, Y., METRAL, M., MAES, P. (1993). Collaborative Interface Agents, Massachusetts Institute of Technology, Cambridge, MA.

LEANEY, P. G., WITTENBERG, G. (1992). Design For Assembling - The Evaluation Methods Of Hitachi, Boothroyd And Lucas, Assembly Automation, Vol. 12, No. 2, 8-17.

MASCLE, C. (1993 Juillet). Conception d'assemblage, Cours donné aux études supérieures, Département de génie mécanique, École Polytechnique de Montréal, Montréal.

MCMAHON, C., BROWNE, J. (1993). CADCAM From Principles to Practice, Addison-Wesley, Wokingham, England, 508 p.

NWANA, H. S. (1996 Sept.). Software Agents : An Overview, Knowledge Engineering Review, Vol. 11, No. 3, 1-40.

OLSEN, G. R., CUTKOSKY, M., TENENBAUM, J. M., GRUBER, T. R. (1994). Collaborative Engineering based on Knowledge Sharing Agreements, 1994 ASME Database Symposium.

PEARSON, S. *et al.* (1996-1). An Agent-Based Approach to Task Allocation in a Computer Support Team, Intelligent Networked Computing Laboratory, Hewlett-Packard Laboratories Bristol, Bristol, UK.

PEARSON, S. *et al.* (1996-2). An Agent Communication Platform Developed in Object Oriented Prolog, Intelligent Networked Computing Laboratory, Hewlett-Packard Laboratories Bristol, Bristol, UK.

PERRARD, C. (1992). Contribution à une méthodologie d'aide à l'implantation et à la spécification des équipements des systèmes flexibles d'assemblage, Thèse de doctorat, Faculté des sciences et des techniques de l'Université de Franche-Comté, 7-12.

SALOMONS, O. W., VAN HOUTEN, F. J. M. A., KALS, H. J. J. (N/D). Review of Research in Feature-Based Design, Journal of Manufacturing Systems, Vol. 12, No. 2.

SCHACH, S. R. (1993). Software Engineering, Second Edition, Aksen Associates, Homewood, IL, 516-518.

SCHRAFT, R. D. (N/D) Assembly-Oriented Design - Condition for Successful Automation (Montagegerechte Konstruktion - Die Voraussetzung für eine Erfolgreiche Automatisierung), Fraunhofer Institut für Produktionstechnik und Automatisierung (IPA), Stuttgart, West Germany.

TELESCRIPT (1995). http://www.genmagic.com

WARNECKE, LÖRH ET KIENER. (1980). Techniques de montage et d'assemblage, Edirep, Paris.

WAYNER, P. (1995). Agents Unleashed, AP Professional, Boston, 358 p.

WEIZENBAUM, J. (1976). Computer Power and Human Reasoning, W.H. Freeman and Co.

WHATIS (1997). http://whatis.com/neuralne.htm

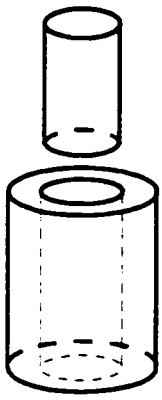
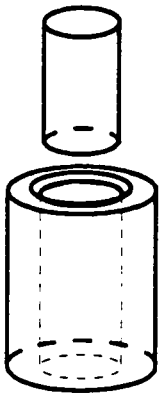
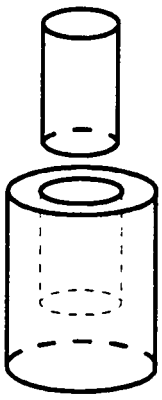
WHITE, J.E. (1995 Oct.). Mobile Agents, General Magic Inc., http://www.genmagic.com.

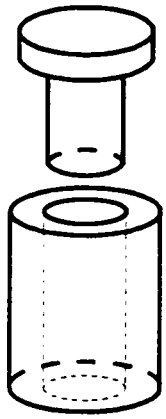
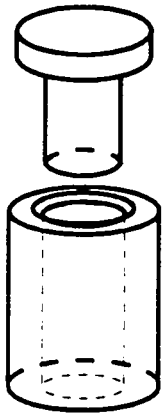
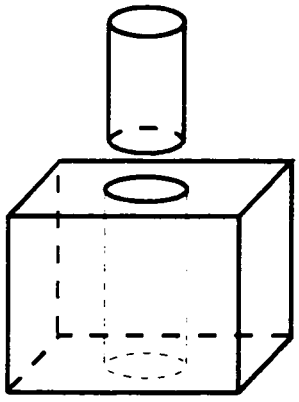
WILLIAMSON, M., DECKER, K., SYCARA, K. (1996 Avril) Executing Decision-theoretic Plans in Multi-agent Environments, The Robotics Institute, Carnegie-Mellon University.

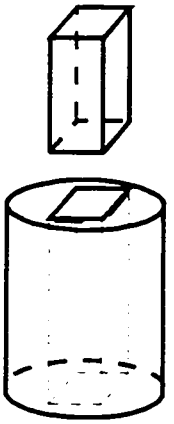
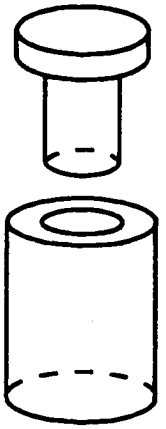
WOOLDRIDGE, M. J., JENNINGS, N. R. (1994). Agent Theories, Architectures, and Languages: A Survey, Department of Computing, Manchester Metropolitan University, United Kingdom.

WONG, J. W., STURGES, R. H. (1992). An Extension of Design for Assembly Methods for Large and Heavy Parts, Carnegie Mellon University, Department of Mechanical Engineering, Pittsburgh, Pennsylvania. Concurrent Engineering, PED-Vol. 59.

ANNEXE I - SIGNATURE D'ASSEMBLAGES

Cas	Image	Signature d'assemblage	Remarques
1		s = [2,2,2,2,4,4] n = 6	Trou au travers.
2		s = [2,2,2,2,4,4] n = 6	Avec chanfrein. Donne le même résultat que sans chanfrein (donc simplification de certains détails de l'assemblage)
3		s = [2,2,3,3,5,5] n = 6	Accotement de la pièce insérée dans le fond du trou

4		s = [2,2,2,2,6,6,6,6] n = 8	Contact de deux surfaces.
5		s = [2,2,2,2,4,4,4,6,6] n = 9	Ce chanfrein est situé entre deux surfaces mutuellement en contact. La signature est différente dans ce cas-ci. Il est donc détectable.
6		s = [2,2,2,2,4,4] n = 6	Même résultat que le premier cas. La signature de l'assemblage donne donc une appréciation locale de l'assemblage.

7		s = [16,16,16,16,24,24, 24,24,24,24,24,24] n = 12	Insertion prismatique.
8		s = [2,2,3,3,6,6,7,7] n =8	État hyperstatique.

ANNEXE II - ALGORITHME DE RECHERCHE D'UNE SIGNATURE D'ASSEMBLAGE

Faire un tableau de données avec toutes les faces connexes d'un composant, dont le résultat de l'intersection est non nul, en éliminant ensuite toutes les redondances.

Tant que toutes les faces du composant 1 n'ont pas été analysées

    Tant que toutes les faces du composant 2 n'ont pas été analysées

        Si l'union des deux faces donne un résultat non nul

            Conserver ce contact dans le tableau des liaisons

Pour chaque ligne du tableau de liaison

    Tant que, pour la première pièce, il reste des faces qui intersectent la face de la première pièce contenue dans la liaison.

        Tant que, pour la deuxième pièce, il reste des faces qui intersectent la face de la deuxième pièce contenue dans la liaison.

            Inscrire les couples de faces qui s'intersectent dans un tableau

Totaliser le nombre d'apparitions pour chaque face.

Ordonner ces valeurs pour trouver le vecteur de signature de l'assemblage.

ANNEXE III - CODE EN C POUR LA DÉTERMINATION DE LA SIGNATURE D'ASSEMBLAGE

```
/*************************************************************************
 * Programme: SIGNATURE D'ASSEMBLAGES
 *
 * Projet:    Maitrise de Bernard DeGuire, en collaboration avec
 *            MM. C. Mascle et C. Fortin de l'Ecole Polytechnique de
 *            Montreal, et Alliance Commerciale Technologique.
 *
 * Auteur:    Bernard DeGuire
 * Date:      27 avril 1997
 *
 * Objectif:  Le but de ce programme est de calculer automatiquement
 *            la "signature" d'un assemblage.
 *
 *                    Il s'agit d'un vecteur de nombre provenant de
l'utilisation
 *            de faces de deux pieces.  Ce vecteur est different selon
 *            le cas d'assemblage rencontre.
 *
 *            Afin d'utiliser ce programme, il faut:
 *             - avoir uniquement un assemblage de 2 pieces;
 *             - decomposer chaque pieces en faces et connaitre
 *               les liens les unissant;
 *             - connaitre tous les liens entre les differentes pieces.
 *
 * Mises a jour:
 *************************************************************************/

#include <stdio.h>
#define FACESMAX 20
#define VECTEURMAX 30
struct utilisation {
  int piece;
  int face;
  int nbfois;
};

void imprimerSignature(struct utilisation * s);
void ordonnerSignature(struct utilisation * s);
void incrementerFace(int face, int piece, struct utilisation * s);
void afficherPieces(int p1[][FACESMAX], int nb1, int p2[][FACESMAX], int
nb2);
void main(void)
{
 int tmp;
 int piece1[2][FACESMAX], piece2[2][FACESMAX];
 int liaison[2][10];
 struct utilisation facesUtil[VECTEURMAX];
 int i, j;
```

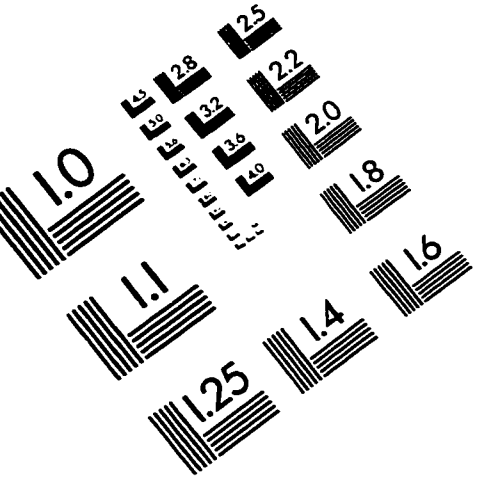
```
 int nopiece, nolien, nbliens;
 int nbliens1, nbliens2;
 int tmplien1, tmplien2;
 int noliaison, nbliaisons;

/*************************************************************************
 * Initialisation
 *************************************************************************/
 for(tmp=0; tmp<VECTEURMAX; tmp++)
 {
  facesUtil[tmp].piece = facesUtil[tmp].face = facesUtil[tmp].nbfois =
0;
 }
 printf("SIGNATURES D'ASSEMBLAGE -- Version 1.0 -- 27-04-97");
 printf("\n     Detection d'identite pour les assemblages possedant 2
composants");
 printf("\n   par Bernard DeGuire");
 printf("\n----------------------\n");

/*************************************************************************
 * Lecture de la morphologie des pieces
 *************************************************************************/
 for(nopiece=1; nopiece<=2; nopiece++)
 {
  printf("\nCombien de liens la piece %d possede-t-elle? ", nopiece);
  scanf("%d", &nbliens);
  (nopiece==1)? (nbliens1=nbliens): (nbliens2=nbliens);
  for(nolien=1; nolien<=nbliens; nolien++)
  {
   printf("Veuillez inscrire des noeud \"x y\":  ");
   scanf("%d %d", &i, &j);
   if(nopiece==1)
   {
    piece1[0][nolien-1] = i;
    piece1[1][nolien-1] = j;
   } else
   {
    piece2[0][nolien-1] = i;
    piece2[1][nolien-1] = j;
   }
  } //for
 } // for
 afficherPieces(piece1, nbliens1, piece2, nbliens2);

/*************************************************************************
 * Lecture des liaisons entre les pieces
 *************************************************************************/
 printf("\n\nCombien de liaisons l'assemblage possede-t-il? ");
 scanf("%d", &nbliaisons);
 for(noliaison=1; noliaison<=nbliaisons; noliaison++)
 {
  printf("Veuillez inscrire la liaison \"face_piece_1 face_piece_2\":
");
  scanf("%d %d", &i, &j);
```

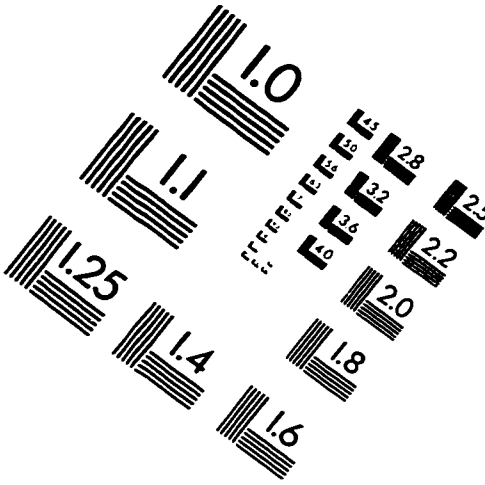
```
   liaison[0][noliaison-1] = i;
   liaison[1][noliaison-1] = j;
  }

/*************************************************************************
 * Determination de la signature
 *************************************************************************/
 // 1. Regarder chaque lien un apres l'autre
 for(noliaison=1; noliaison<=nbliaisons; noliaison++)
 {
 // 2. Regarder sur la piece 1 les lignes du tableau les unes apres les
autres
   for(tmplien1=1; tmplien1<=nbliens1 ; tmplien1++)
   {
    if(piece1[0][tmplien1-1]==liaison[0][noliaison-1] ||
       piece1[1][tmplien1-1]==liaison[0][noliaison-1])
 // 3. Regarder sur la piece 2 les lignes du tableau les unes apres les
autres
    for(tmplien2=1; tmplien2<=nbliens2 ; tmplien2++)
    {
     if(piece2[0][tmplien2-1]==liaison[1][noliaison-1] ||
        piece2[1][tmplien2-1]==liaison[1][noliaison-1])
     {
 // 4. Incrementer l'utilisation des 4 faces
      incrementerFace(liaison[0][noliaison-1],1,facesUtil);
      incrementerFace(liaison[1][noliaison-1],2,facesUtil);
      if(piece1[0][tmplien1-1]==liaison[0][noliaison-1])
      {
       incrementerFace(piece1[1][tmplien1-1],1,facesUtil);
      } else
      {
       incrementerFace(piece1[0][tmplien1-1],1,facesUtil);
      }
      if(piece2[0][tmplien2-1]==liaison[1][noliaison-1])
      {
       incrementerFace(piece2[1][tmplien2-1],2,facesUtil);
      } else
      {
       incrementerFace(piece2[0][tmplien2-1],2,facesUtil);
      }
     }
    }
   }
  }
 }
 ordonnerSignature(facesUtil);
 imprimerSignature(facesUtil);
 printf("\nFini.\n");
}

/*************************************************************************
 * Impression de la signature a l'ecran
 *************************************************************************/
void imprimerSignature(struct utilisation * s)
{
```

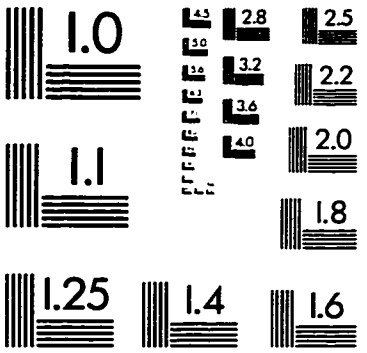
```
 int tmp=0;

 printf("\nSignature: ");
 while(s[tmp].nbfois!=0)
 {
  printf(" %d",s[tmp].nbfois);
  tmp++;
 }
 return;
}

/*****************************************************************************
 * Ordonnancement de la signature en ordre chronologique
 *****************************************************************************/
void ordonnerSignature(struct utilisation * s)
{
 int tmp=0;
 int plusbas=1;
 struct utilisation stmp;

 while(s[tmp].nbfois!=0)
 {
  while(s[plusbas].nbfois!=0)
  {
   if(s[plusbas].nbfois<s[tmp].nbfois)
   {
    stmp = s[plusbas];
    s[plusbas] = s[tmp];
    s[tmp] = stmp;
   }
   plusbas++;
  }
  tmp++;
  plusbas = tmp + 1;
 }
 return;
}

/*****************************************************************************
 * Incrementation de l'utilisation d'une face
 *****************************************************************************/
void incrementerFace(int face, int piece, struct utilisation * s)
{
 int tmp;

 // Voir si la face est deja la, sinon l'ajouter
 for(tmp=0; tmp<VECTEURMAX; tmp++)
 {
  if(s[tmp].face==face && s[tmp].piece==piece)
  {
   s[tmp].nbfois++;
   return;
  }
 } //for
```

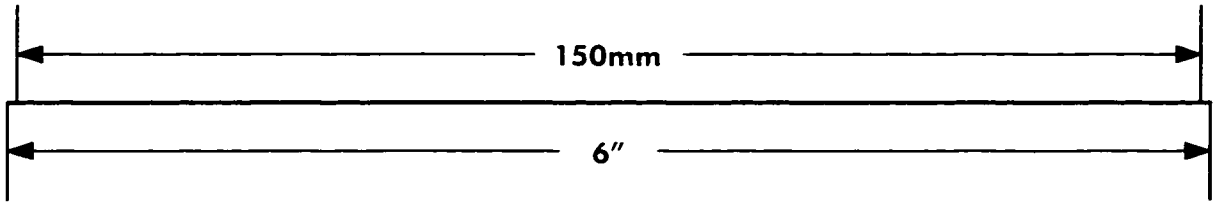
```
 // Trouver la premiere occurence de nbfois egale a 0
 for(tmp=0; tmp<VECTEURMAX; tmp++)
 {
  if(s[tmp].nbfois==0)
  {
   s[tmp].face = face;
   s[tmp].piece = piece;
   s[tmp].nbfois++;
   return;
  }
 }
}

/*************************************************************************
 * Affichage de la definition des pieces
 *************************************************************************/
void afficherPieces(int p1[][FACESMAX], int nb1, int p2[][FACESMAX], int
nb2)
{
 int i;
 printf("\n+-------+-----------+-----------+");
 printf("\n| Piece | Lien entre |     et     |");
 printf("\n|-------+-----------+-----------|");
 printf("\n|   1   |           |           |");
 for(i=0; i<nb1; i++)
 {
  printf("\n|       |    %4d    |    %4d    |",p1[0][i],p1[1][i]);
 }
 printf("\n|-------+-----------+-----------|");
 printf("\n|   2   |           |           |");
 for(i=0; i<nb2; i++)
 {
  printf("\n|       |    %4d    |    %4d    |",p2[0][i],p2[1][i]);
 }
 printf("\n+-------+-----------+-----------+");
 return;
}
```

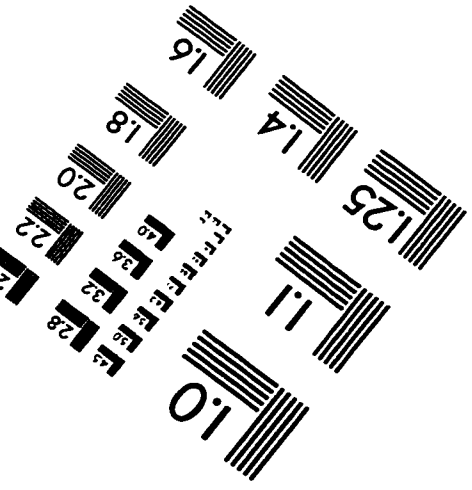
IMAGE EVALUATION
TEST TARGET (QA-3)

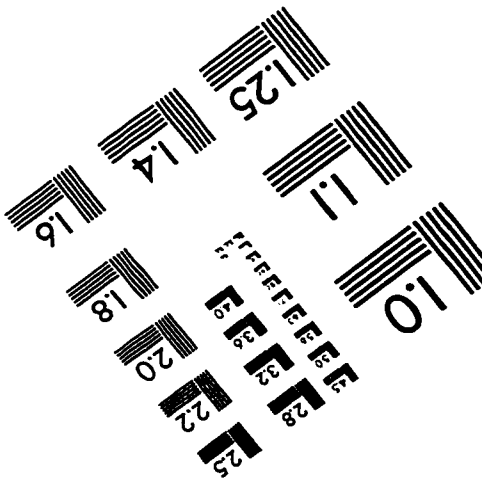












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989