THÈSE
présentée à

l'UNIVERSITÉ de METZ

pour l'obtention du grade de
DOCTEUR de l'UNIVERSITÉ de METZ

Spécialité : INFORMATIQUE

Lotfi BELLALEM

Contribution à la conception et à la mise en oeuvre d'un système d'aide fondé sur la planification interactive dédié à la CAO

Présentée et soutenue le 3 Décembre 1998

Composition du Jury

Rapporteurs :
Ileana COSTEA — Professeur à l'Université de Californie
Marie-Christine HATON — Professeur à l'Université Henri Poincaré Nancy 1
Directeur de thèse :
Yvon GARDAN — Professeur à l'Université de Metz
Examinateurs :
Jean-Pierre JUNG — Professeur à l'Université de Metz
Ibrahima SAKHO — Professeur à l'Université de Metz
Abdou ZAKARI — Maître de Conférences à l'Université de Metz

LABORATOIRE DE RECHERCHE EN INFORMATIQUE DE METZ

A Nadia, Yanis et Inès

Remerciements

Ce travail a été réalisé au Laboratoire de Recherche en Informatique de Metz, sous la direction de Monsieur le Professeur YVON GARDAN. Je tiens à lui exprimer ma reconnaissance pour la confiance qu'il m'a témoignée, pour son aide, son dynamisme et la compétence dont il m'a fait profiter tout au long de cette thèse.

Qu'il me soit permis de remercier Mesdames MARIE-CHRISTINE HATON, Professeur à l'Université HENRI POINCARÉ NANCY 1 et ILEANA COSTEA Professeur à l'Université de Californie, pour avoir accepté de rapporter ce travail, pour les propositions judicieuses qu'elles m'ont suggérées et pour leurs participations à la commission d'examen.

Mes remerciements vont également à Messieurs JEAN-PIERRE JUNG et IBRAHIMA SAKHO, Professeurs à l'Université de Metz, pour avoir accepté de juger mon travail.

Je tiens à remercier tout particulièrement ABDOU ZAKARI qui a suivi ce travail, qui m'a conseillé, orienté, et très souvent motivé. Qu'il trouve ici l'expression de ma très profonde gratitude.

Je remercie très chaleureusement KAMEL SMAILI et DANIÈLE VAHIÉ pour leurs lectures consciences, leurs critiques et conseils constructifs qui m'ont aidé à améliorer la qualité de ce document mais aussi pour le soutien qu'il m'ont prodigué.

Je remercie également les membres du laboratoire pour l'ambiance tant scientifique qu'humaine dont j'ai bénéficié ainsi que toutes les personnes qui m'ont encouragé.

TABLE DES MATIERES

Introduction

La mondialisation actuelle des marchés raccourcit la durée de vie des produits. Elle impose donc la nécessité accrue d'un système qui permettrait l'amélioration du processus d'élaboration des produits industriels et, en particulier, de l'assemblage de pièces mécaniques qui y joue un rôle fondamental.

Les techniques issus de l'intelligence artificielle ont permis de faire évoluer les systèmes de C.A.O vers des systèmes plus performants. Des domaines comme la construction mécanique en général et les méthodes d'assemblage en particulier ont largement bénéficié de ces techniques. Pourtant, les systèmes proposés se basent le plus souvent sur une approche système expert dont l'objectif est de fournir des méthodes générales de construction ou d'assemblage, où l'utilisateur reste dans une position d'attente passive, ou alors, quand ce n'est pas le cas, sa participation se borne à agir sur certains paramètres de construction mais jamais sur l'activité d'assemblage elle-même. En outre, les systèmes interactifs dédiés à ce type d'applications sont souvent partiels et spécifiques de chaque application.

Dans ce travail, nous proposons une contribution à la mise en place d'un système d'aide fondé sur la planification interactive dédié à la C.A.O. Ce système permettant de générer des séquences d'assemblage de manière interactive, l'utilisateur agissant comme un partenaire du système et non plus comme un acteur passif. L'approche choisie est fondée sur la coopération entre les mécanismes de la planification interactive, la modélisation des objets C.A.O. sous forme prédicative à travers des diagrammes de relations, ainsi que la gestion du dialogue. Une interface, ayant des caractéristiques spécifiques, vient compléter l'architecture globale.

Ce travail s'organise autour de quatre chapitres :

Dans le premier chapitre nous passons en revue un certain nombre de mécanismes pour la construction de systèmes experts, précédés d'une présentation des fondements du couplage I.A-C.A.O, c'est-à-dire :

- Développer une nouvelle génération de systèmes de C.A.O en utilisant les techniques de l'I.A.
- Intégrer des fonctionnalités I.A dans les logiciels C.A.O.
- Coupler des systèmes déductifs avec des outils C.A.O traditionnels.

Par la suite, une première approche sur l'interaction Homme-Machine est abordée dans une perspective plus détaillée dans le chapitre 2. Enfin, la planification sous son aspect formel et sous forme d'exemples est largement développée. Cette première partie permet de définir les fondements et les outils sur lesquels sera défini le système d'aide qui est proposé.

Dans le deuxième chapitre, la problématique de l'interaction Homme-Machine est abordée dans son ensemble, c'est-à-dire à travers la modélisation formelle qui régit les systèmes interactifs d'une part, et des exemples aboutis de logiciels interactifs d'autre part. Ce chapitre se termine par la description du logiciel S.A.C.A.D.O (Système Adaptatif de Conception Assistée et de développement par ordinateur) qui est le système développé au LRIM (laboratoire de recherche en informatique et mécanique) et dont un des principaux buts est justement la modélisation du dialogue Homme-Machine.

Cette deuxième partie permet en outre de se rendre compte de la complexité dans la mise en place des interfaces, de leurs avantages ainsi que de leurs limites. A partir de là, sont dégagées les caractéristiques que doit posséder le système d'aide que nous proposons.

Dans le troisième chapitre, le système d'aide est décrit à travers les modules qui le composent. Auparavant l'application principale est précisée c'est-à-dire le processus de conception dans la construction et l'assemblage de pièces mécaniques.
La description du système d'aide est l'aboutissement des études des deux premiers chapitres, elle en est une synthèse mais pas une finalité. Elle présente comme principale originalité d'associer des outils très différents dans leurs principes, mais complémentaires dans le processus d'enchaînement des procédures d'action.

Dans le quatrième et dernier chapitre nous mettons en œuvre les orientations choisies à travers des exemples pratiques, comme l'assemblage d'un clapet de compresseur et celui d'un alternateur. Les particularités de notre système d'aide sont décrites plus précisément (multimodèles, type d'utilisateurs, modélisation des questions). Ce chapitre se termine par l'analyse des différentes caractéristiques de l'interface associée à notre système d'aide.

Enfin, une partie annexe décrit quelques généralités sur la construction mécanique ainsi qu'une partie de la réalisation informatique en langage Common Lisp.

Chapitre 1

L'état de l'art

1 Introduction

L'évolution de la C.A.O s'est faite ces vingt dernières années autour de cinq périodes principales.

La première période s'est caractérisée par l'aspect calculatoire des systèmes de C.A.O sous la forme, le plus souvent, d'outils autonomes construits autour d'algorithmes spécifiques.
La deuxième étape s'est caractérisée par la nécessité de proposer des interfaces Homme-Machine. Ainsi, les premières sorties graphiques ont vu le jour, suivies par l'utilisation d'outils interactifs et ce, grâce en partie au traitement en temps partagé.
La troisième période avait pour but l'intégration de divers outils au sein d'un même système de C.A.O. (traitement, mémorisation, interface) de sorte que le concepteur du produit dispose de l'ensemble de ces outils.
La quatrième période voit apparaître ce qu'on appelle les "ingénieries des systèmes" ou encore CIM (computer integrated manufacturing) ou (systèmes d'information intégrés), cela consiste à intégrer autour d'un même système tous les aspects de conception, d'étude, de développement, de fabrication et de gestion du processus industriel avec le souci d'une meilleure coopération possible entre la conception et la fabrication.
La cinquième période, qui nous intéresse plus particulièrement, est l'introduction de techniques d'intelligence artificielle à tous les niveaux du processus de conception. L'utilisation de ces techniques a pour but de faire évoluer les systèmes de C.A.O vers des systèmes dit "intelligents".

L'objectif de cette première partie est de mettre en évidence les différents aspects (système experts, interactivité Homme-Machine, planification) qui seront à la base du modèle que nous proposons dans le chapitre 3.

Ce chapitre se divise en trois parties. La première partie décrit les principales approches dans le couplage I.A-C.A.O ainsi que les principaux travaux sur les systèmes à base de règles de production. Par la suite, les concepts et outils des systèmes experts généraux ainsi que les tendances actuelles avec les SE2G (systèmes experts de deuxième génération) sont explicités. La deuxième partie décrit l'approche théorique des modèles associés à l'étude de l'interaction Homme-Machine. Plus précisément, le modèle du processeur Humain appelé G.O.M.S (Goal Opérateur Method Selection) ainsi que KEYSTROKE un dérivé de G.O.M.S. La troisième partie, décrit les concepts de la planification ainsi que le fonctionnement de certains systèmes de planification classiques. Cette partie se termine par une présentation des thèmes d'étude actuels dans les systèmes de planification.

Le but de cette étude est de mettre en évidence la diversité des approches dans la gestion des systèmes de C.A.O, et mettre en relief les choix possibles qui peuvent être entrepris dans la mise en place des systèmes d'aides.

2 Couplage entre l'I.A et la C.A.O

Le couplage de l'I.A et de la C.A.O permet d'apporter une réelle assistance au processus de conception et de préparation à la fabrication que la seule utilisation d'algorithmes fondés sur les modèles géométriques ne permet pas. Cependant, l'application des systèmes intelligents est encore limitée. Ceci est dû aux difficultés rencontrées lors du développement de ces systèmes. Nous allons décrire pour le moment les différentes approches, résultat de ce couplage.

2.1 Les approches

L'exploitation des techniques I.A en C.A.O peut être effectuée via les trois approches suivantes :

1 - Une première approche, dite dirigée par l'I.A, consiste à développer une nouvelle génération de systèmes de C.A.O à partir d'outils intégrants les techniques de l'I.A.
2 - Une seconde approche dite dirigée par la C.A.O consisterait à intégrer tout ou une partie des fonctionnalités I.A dans le développement de logiciels de C.A.O.
3 - La troisième approche, dite mixte, consiste à coupler outils de C.A.O classiques et outils intégrant les techniques d'I.A, c'est à dire déductifs.

Cette décomposition est bien sûr arbitraire. On peut en effet faire le choix d'une autre décomposition à partir des fonctionnalités offertes par les outils. Par exemple, séparer les systèmes orientés processus de conception des systèmes orientés modélisation géométrique. Néanmoins la première classification présente l'avantage de la simplicité.

2.2 Discussions

L'approche dirigée par l'I.A possède l'avantage d'une technologie qui offre un potentiel de développement énorme. Néanmoins, cette approche accuse un certain retard quant aux possibilités graphiques et aux performances d'affichage des logiciels, sans compter la modification profonde des habitudes de travail pour les utilisateurs finaux. En outre, l'intérêt de ce type d'outil au sein de l'entreprise reste à démontrer. En effet on peut se demander à qui s'adresse ce genre d'outils? Pas au bureau d'étude car ils sont trop difficiles d'accès pour les concepteurs. On assiste donc à un repositionnement de ce genre de système en amont des logiciels de C.A.O traditionnels assortis d'accords avec les éditeurs. On se rapproche ainsi de la troisième approche, c'est-à-dire l'approche mixte.

L'approche dirigée par la C.A.O est d'actualité, et les éditeurs de logiciel de C.A.O proposent des environnements C.A.O qui intègrent des fonctionnalités I.A. Néanmoins, il semble que ces fonctionnalités sont encore trop éloignées de celles nécessaires à la réalisation de véritables systèmes de conception. Par ailleurs, l'intégration de fonctionnalités puissantes dans un environnement C.A.O traditionnel impose pratiquement une réécriture du noyau. C'est pourquoi on se dirige vers une évolution progressive des environnements de C.A.O où l'I.A aura une place plus importante et les techniques associées seront plus puissantes encore.

L'expérience de XCON [Mac Dermott, 80], par exemple, constitue la première application des techniques d'I.A à la conception ou plutôt à la configuration, elle a montré qu'à partir d'une certaine complexité, la représentation des connaissances est inadaptée. Or, la réalisation d'un système expert de conception nécessite généralement une grande puissance d'expression.

Par ailleurs, l'intégration de fonctionnalités puissantes dans un environnement C.A.O traditionnel impose pratiquement une réécriture du noyau. La solution intermédiaire est d'évoluer progressivement vers des fonctionnalités de plus en plus puissantes. Par exemple, un système déductif à base de règles dans un premier temps, puis un système à objets, un système à base de contraintes, un système de solutions multiples, etc.
La troisième approche est celle en vigueur actuellement. Elle présente l'avantage de la maturité, maturité des logiciels de C.A.O et maturité des générateurs de systèmes experts.
Cette solution offre l'avantage de tirer partie de la puissance importante de chacun des deux thèmes de recherche : puissance de modélisation pour les logiciel d'I.A d'un côté et puissance de calcul et de représentation géométrique des logiciels de C.A.O de l'autre. C'est donc une bonne alternative aux deux autres approches à condition d'utiliser des logiciels suffisamment ouverts pour qu'un couplage puisse être rapidement développé. En effet, l'inconvénient essentiel de tout couplage réside dans la cohérence entre les données, il faut prévoir alors des interfaces de communication et des protocoles de maintien de la cohérence. La pérennité de cette approche réside dans la résolution de ces aspects.

En conclusion, si aucune de ces trois approches n'est parfaite, l'approche mixte semble être la plus utilisée actuellement. En effet, l'avènement des bases de données orientées objets et l'avancée des systèmes d'information intégrés poussent vers des solutions où les modèles de C.A.O se trouveront dans des bases de données externes. De ce fait, elles deviennent le support de communication entre l'I.A et la C.A.O et garantissent le maintien de la cohérence.

Avant de présenter des exemples de systèmes experts opérants, quelques notions fondamentales sur les systèmes experts sont indispensables.

2.3 Les systèmes experts

2.3.1 Définition

A partir de l'idée que les systèmes experts n'ont pas vocation à modéliser complètement le raisonnement humain, on peut définir un système expert comme étant un système informatique capable de simuler un expert dans son activité de résolution de problème dans un domaine de connaissances [Gardan, 91].

2.3.2 Les différentes parties d'un générateur de systèmes expert

L'hypothèse principale qui sous-tend la construction des systèmes experts est la reconnaissance du rôle majeur que jouent les connaissances dans la résolution d'un problème.
L'efficacité et la compétence de ces systèmes, y compris sur des problèmes non triviaux, proviennent davantage des connaissances spécifiques que ces systèmes possèdent sur ces problèmes particuliers, que de la puissance de techniques générales de résolution de problèmes. La construction d'un système expert nécessite donc l'acquisition de connaissances. Ces connaissances peuvent être acquises auprès d'experts du domaine, c'est-à-dire de personnes particulièrement reconnues au sein de leur communauté. Le système expert pourra prétendre exhiber un comportement de haut niveau manipulant des connaissances que seuls ces experts possèdent.

Historiquement, les premiers systèmes experts relèvent plutôt de cette catégorie. Plus souvent, les connaissances seront acquises auprès d'un groupe de spécialistes qui mettent en commun leur savoir faire. Plus rarement, ces connaissances seront acquises automatiquement, par apprentissage.

Le terme "système expert" est employé dans deux sens différents. Dans le premier cas, il s'agit d'un système informatique conçu suivant la définition précédente qui possède, en particulier [Haton, 91] :

- Une ou plusieurs bases de connaissances constituant la mémoire à long terme du système dans laquelle est codé, éventuellement dans plusieurs formalismes, l'ensemble des connaissances d'un domaine, à la fois des faits permanents du savoir-faire et l'expertise nécessaires pour résoudre un problème.
- Une base de faits contenant des faits ou des données propres à un problème à résoudre. Il s'agit donc d'une mémoire de travail qui s'enrichira de faits nouveaux découverts par le mécanisme de raisonnement jusqu'à ce que l'on parvienne à la solution du problème.
- Un moteur d'inférence, algorithme chargé d'exploiter les connaissances et les faits pour mener un raisonnement.

Cependant le même système informatique peut fort bien être livré vide, c'est-à-dire sans connaissances, à charge pour l'utilisateur d'y introduire les règles, et les faits souhaités.
Dans ce cas, nous dirons que le système est un « générateur de système expert » (GSE) ou un outil de développement. Comme tous les systèmes de traitement de l'information, un GSE possède une mémoire (sa base de connaissances), un processeur (son moteur d'inférence) et des procédures d'entrées-sorties sous la forme le plus souvent d'une interface Homme-Machine.
Pour introduire de nouvelles règles, modifier des faits et des règles, chaque GSE possède un éditeur. Cet éditeur est l'auxiliaire le plus précieux quand on développe le système voir figure 1.1. De sa qualité dépendra en particulier le temps consacré au développement.

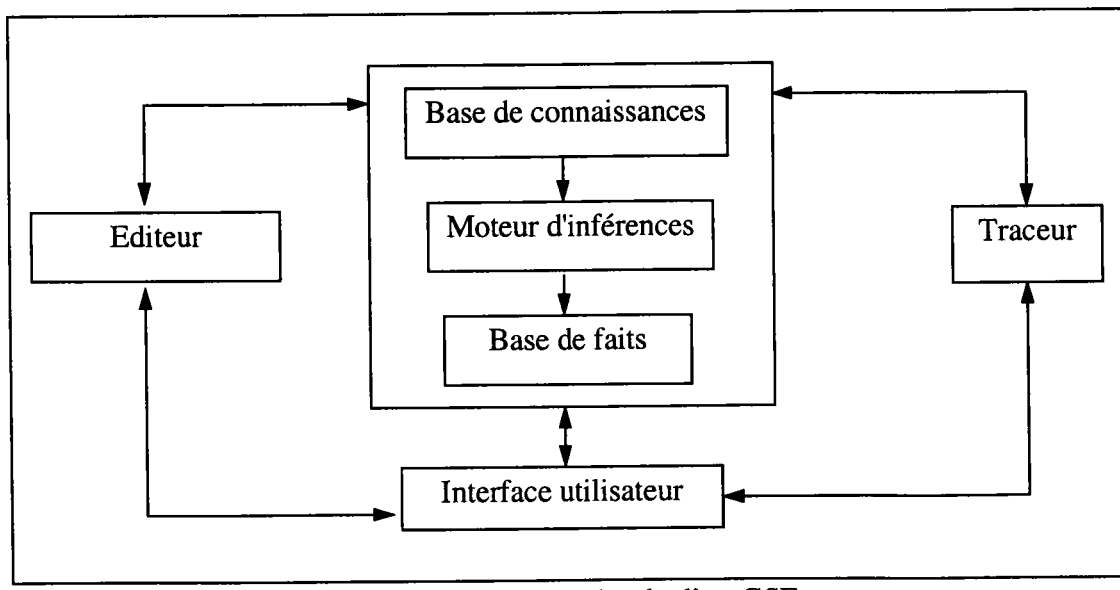


Figure1.1 Architecture simple d'un GSE

Avant d'aller plus loin il est important de définir plus précisément les mécanismes liés à la construction de systèmes experts

2.4 Les différents aspects liés à la construction de systèmes experts

Les difficultés rencontrées lors de la construction des systèmes experts concernent outre les problèmes liés à l'acquisition de connaissances, l'explication du raisonnement, le manque de robustesse et la portabilité. Il paraît intéressant de fournir quelques explications sur ces termes.

2.4.1 L'acquisition de connaissances

Les experts sont quelquefois peu disposés à expliquer rationnellement leur comportement devant tel ou tel problème et par conséquent ils ont des difficultés à fournir les associations heuristiques dont a besoin le système. De plus, rien ne permet d'aider l'expert ou le cogniticien à identifier les connaissances liées à une tâche.

2.4.2 L'explication du raisonnement

Les connaissances du domaine étant implicites dans la base de connaissances, elles restent inaccessibles au module d'explication. Une autre difficulté réside dans la description du processus de résolution du problème. En effet, ce processus est décrit à travers l'enchaînement des règles, ce qui paraît insuffisant pour décrire correctement le comportement d'un système expert.

2.4.3 La robustesse

Le problème des systèmes experts réside aussi dans leur difficulté à s'adapter à un nouveau problème. En théorie, il suffit d'ajouter ou de modifier une connaissance pour que le système évolue et réussisse à résoudre le nouveau problème. En pratique, les règles sont si dépendantes les unes des autres que la moindre modification peut entraîner des modifications non seulement sur la résolution des problèmes à venir mais aussi sur la capacité du système à pouvoir résoudre des problèmes qu'il savait résoudre auparavant.

2.4.4 La portablité

L'étroite imbrication entre tâche à réaliser, méthodes, et connaissances du domaine fait qu'il est impossible de réutiliser tel ou tel modèle ou méthode dans un autre domaine que celui pour lequel il a été construit. La plupart de ces problèmes sont issus de l'uniformité des connaissances utilisées et de leur faible abstraction, les chercheurs se sont donc dirigés vers deux voies de recherche pour tenter de les résoudre :
D'une part rendre les systèmes plus hétérogènes, c'est-à-dire intégrants différents types de connaissances avec différentes méthodes qui les exploitent. D'autre part, distinguer connaissances et représentation des connaissances, et c'est ainsi qu'est apparu le *knowledge level* qu'on détaillera plus loin.

3 Les différents systèmes associés au couplage I.A-C.A.O

3.1 Introduction

Les évolutions du domaine de l'intelligence artificielle appliquée à la C.A.O peuvent être décomposées suivant les étapes chronologiques suivantes.

- les premiers systèmes à base de règles de production,
- les systèmes mixtes intégrant les notions d'objets, règles, et méthodes,
- les systèmes I.A spécialisés conception,
- le couplage de ces systèmes avec les modeleurs C.A.O,
- les systèmes de C.A.O implémentés sur des techniques I.A,
- l'utilisation de la programmation par contrainte dans la résolution des sous-problèmes,
- l'intégration dans les systèmes de C.A.O traditionnels de fonctionnalités I.A.

Nous allons dans la suite de ce chapitre décrire des exemples de systèmes opérationnels fondés sur les trois approches citées en 2.1, en respectant autant que possible la chronologie des étapes décrites ci-dessus.

3.2 Les premiers systèmes à base de règles de production

Le système XCON
XCON [Mac Dermott, 80] constitue une des premières applications des techniques I.A à la conception et à la configuration. C'est un système développé par DEC pour assembler des composants informatiques (mémoire, unité de contrôle mémoire, interface de bus, unité périphérique, processeur, etc.) afin de configurer un VAX en réponse à un besoin précis. La combinatoire induite par la grande quantité de composants a incité DEC à réaliser un système expert capable de résoudre ce problème et dont les caractéristiques sont les suivantes :

- le système maintient en permanence la notion de contexte, c'est à dire qu'il sait à tout moment ce qu'il est en train de faire,
- le système utilise une base de données technique qui contient la description d'environ 400 composants,
- le système est écrit dans un langage de programmation propre appelé OPS5 et comporte des règles en grande quantité,
- le système possède une structure de raisonnement qui utilise le "pattern-matching d'OPS5".

Exemple de règles :
SI
Le contexte actif est l'affectation d'une unité de puissance et un adaptateur de type UNIBUS a été placé dans une carte et la position de cet adaptateur est connue et il reste de la place dans la carte pour l'unité de puissance nécessaire à la carte et une unité de puissance est disponible et il n'existe pas de régulateur de type H7101 disponible.
ALORS
Ajouter le régulateur de type H7101.

Le processus de conception est le suivant :
1 - Le processus vérifie que la commande client est globalement bonne.
2 - Il affecte ensuite les composants adéquats dans l'unité de traitement centrale et ses cartes d'extension.
3 - Le processus alloue des emplacements dans le bus d'extension et place les composants nécessaires à l'intérieur.
4 - Le processus place ensuite intelligemment les composants dans la configuration.
5 - Enfin, il met en place le câblage.

L'apport de ce système est d'avoir permis de montrer qu'il était possible de réaliser des systèmes intelligents avec des formalismes simples de type règle de production (SI-ALORS-SINON). En revanche, l'inconvénient reste le trop grand nombre de règles et la maintenance que cela implique.

Le système SMECI

SMECI (Système Multi-expert de Conception en Ingénierie) [Haren, 85b] est un projet INRIA dont l'objectif initial était de réaliser un environnement de développement de systèmes experts. Les spécifications de ce système ont donné lieu à la réalisation du premier générateur de système expert spécialisé dans la résolution de problème de conception. Il a ensuite été généralisé à d'autres classes de problèmes, comme la simulation, la planification, etc.

Les spécifications de ce système sont les suivantes :

- SMECI utilise le formalisme « objet » pour représenter le domaine, à base de catégories, de prototypes et d'objets réels. Les objets servent à modéliser l'univers de travail du système et les catégories définissent la structure des objets. Les prototypes imposent des contraintes et des comportements aux objets.
- SMECI offre la possibilité de décomposer la résolution d'un problème en sous-problèmes et fournit cette décomposition au moteur d'inférence. Le moteur de SMECI peut ainsi être vu comme un interpréteur de solveur de problèmes.
- Le formalisme à base de règles est regroupé en paquet. Le moteur n'examine à chaque instant qu'un ensemble restreint de règles.
- SMECI offre la possibilité de gérer de la multi-expertise, ce qui a été très peu utilisé par la suite.
- Ce système possède la possibilité de gérer plusieurs solutions au cours d'un même raisonnement.
- Possibilité de construire un arbre d'état représentant les différentes étapes et hypothèses du raisonnement.
- Possibilité de définir une fonction d'évaluation chargée de guider le raisonnement vers la solution la plus adéquate.
- Possibilité de demander des explications sur le résultat d'une conception.

Exemple de règle:

SMECI a été initialement conçu pour la conception de digues portuaires, d'où l'exemple de règle suivante :

De-Règle calcul-couronnement
Soit Co une Conception
Coupe-digue la coupe de Co
Rep une réponse
B un béton
Si le franchissement de Co = non et
le Couronnement de la Coupe-digue = (vide)
Alors créer C un Couronnement puis
C est un couronnement de la coupe -digue
la coupe-digue est la coupe de C
la côte-supérieure de la coupe-digue est la côte-supérieure de C
l'enfoncement de C = 0.
le prix-f/m3 de B est le coût-f/m3 de C
Fin-règle

La construction d'un système expert nécessite d'autre part comme nous l'avons décrit dans le paragraphe 2.3.2 l'acquisition de connaissances auprès d'un expert du domaine, ou mieux encore auprès d'un groupe de spécialistes du domaine. On note toutefois que ces mécanismes d'acquisition de connaissances peuvent être automatiques et on parle alors d'apprentissage automatique.

Nous allons dans un premier temps nous intéresser à l'évolution des systèmes experts des systèmes dits homogènes vers les systèmes hétérogènes

3.3 Systèmes hétérogènes

3.3.1 Introduction

La plupart des systèmes de première génération sont des systèmes homogènes en ce sens qu'ils manipulent un seul type de connaissances, représenté dans un formalisme unique. Le plus souvent, il s'agit d'associations heuristiques représentées sous forme de règles de productions.
L'identification de méthodes de raisonnement particulières associées à des modèles du domaine particuliers, a tout d'abord permis aux SBC (systèmes à base de connaissances) de se dégager de cette uniformité [Haton et al, 91]. Elle a également permis de concevoir des systèmes hétérogènes.
En effet, au contraire des systèmes experts de première génération, les SE2G (systèmes experts de deuxième génération) utilisent des connaissances de natures différentes [Simmons, 93]. Le but est de rendre ces systèmes plus compétents et plus performants.

3.3.2 Exemples de systèmes hétérogènes

Les systèmes hétérogènes datent du début des années 80, on peut citer par exemple les systèmes ABEL [Patil, 81] et HERSAYII [Erman, 80]. D'autres travaux plus récents ont mis en évidence la possibilité de résoudre un problème en utilisant d'autres types de connaissances que les associations d'heuristiques, c'est la notion de modèle du domaine et de méthodes de résolution associées.

L'exemple du diagnostic de pannes de voiture
Le problème considéré est le diagnostic et la réparation d'un moteur de voiture [David, 83a]. Trois types de connaissances sont mises en œuvre :

1 - Les connaissances pragmatiques de dépannage qui décrivent les réflexes, issues de l'expérience et représentées sous formes de règles.
Par exemple, si le démarreur tourne et le moteur ne démarre pas, alors regarder s'il y a de l'essence dans le réservoir. S'il n'y en a pas, alors remplir le réservoir, sinon regarder s'il y a une étincelle à la bougie. S'il n'y a pas d'étincelle, conclure que l'allumage est en panne, s'il y'a une étincelle alors...etc.

2 - Les connaissances causales décrivent des relations de cause à effet plus au moins complexes entre éléments du domaine. Par exemple : la carburation est trop riche => les bougies s'encrassent.

3 - Les connaissances structurelles décrivent la structure fonctionnelle du matériel. Par exemple le moteur est constitué du bloc moteur, de la carburation, de l'allumage, etc. Le bloc moteur est lui même constitué de pistons, etc. Ainsi, à chaque entité fonctionnelle sont attachés les phénomènes le concernant.

A chacun de ces trois types de connaissances correspond une méthode de diagnostic propre. Une maquette a été réalisée qui fait coopérer l'ensemble sous forme de raisonnement réflexe, causal et structurel.

Le principe est le suivant : si l'approche pragmatique ne suffit pas à résoudre complètement le problème, le système met en œuvre ses connaissances causales pour tenter de construire une explication des phénomènes. A tout moment, les informations acquises lors de ce raisonnement peuvent déclencher un nouveau comportement réflexe.

En dernier ressort, le système raisonnera sur la structure pour identifier la cause du problème. A nouveau, les informations acquises lors de ce raisonnement peuvent permettre de réactiver des connaissances pragmatiques ou causales.

L'exemple de l'interprétation et planification de la forme
Le système GTD (generate, test and debug) [Simmons, 87, 93a] a été développé par R.Simons du MIT, pour résoudre des problèmes d'interprétation ou de planification de la forme, c'est-à-dire construire une séquence d'événements qui conduisent d'un état initial à un état final donné. Le domaine d'utilisation est l'interprétation géologique; il s'agit en fait d'expliquer comment une région géologique est formée.

Les connaissances utilisées sont de deux types, des connaissances causales et des associations d'heuristiques, en effet selon R. Simmons, le raisonnement heuristique est à la fois efficace et fragile parce qu'il fait l'hypothèse que les résultats des associations peuvent être combinés de manière relativement indépendante, et qu'il ne cherche pas à vérifier ces interactions. En revanche, le raisonnement causal est robuste mais inefficace, parce qu'il cherche à vérifier et contrôler systématiquement les interactions potentielles en utilisant un modèle détaillé du domaine. Ce qui suggère d'utiliser ces deux types de raisonnement à des niveaux différents. GTD utilise d'abord le raisonnement heuristique pour générer une solution, ensuite le raisonnement causal est utilisé pour vérifier que cette solution est effectivement correcte, et dans le cas contraire, la corrige.

3.3.3 Discussions

La construction de systèmes hétérogènes permet sans doute de construire des systèmes plus efficaces. En effet, en décomposant les connaissances à acquérir en différents types distincts, et en permettant de choisir le formalisme de représentation le mieux adapté à un style de connaissance particulier (éventuellement plusieurs formalismes différents) les concepteurs facilitent la construction de ces systèmes. Cependant, se pose en premier lieu le problème de l'intégration de différentes méthodes ou représentations qui entraînent un effort de développement plus important. Ensuite, gérer différentes représentations simultanément nécessite d'assurer la cohérence entre les différents formalismes. Enfin, la propagation d'informations d'une représentation à l'autre peut être pénalisante pour l'efficacité globale du système.

3.4 Le niveau de connaissances (*knowledge level*)

Le but de l'approche *knowledge level* est de comprendre ce que le système fait et pourquoi il se comporte de telle ou telle manière. En effet, décrire un système en terme de règles et de chaînage avant/arrière masque la nature des tâches qui sont exécutées.

Selon A. Newel [Newel, 82], *le knowledge level* est un moyen de décrire un système ou un agent comme si cet agent possédait certaines connaissances, sans aucun *a priori* sur la représentation ou l'implémentation utilisée. Plus précisément, il s'agit d'un modèle construit par un observateur extérieur pour rendre compte de ses observations. La figure 1.2 décrit de manière générale le cycle de développement des systèmes experts selon cette approche. La modélisation à pour but de construire le modèle conceptuel du système. Une fois le modèle conceptuel défini, celui-ci va servir de cadre pour "l'élicitation" des connaissances. L'élicitation permet la construction effective de la base de connaissances, enfin l'acquisition de connaissances regroupe l'ensemble des activités : modélisation et élicitation.

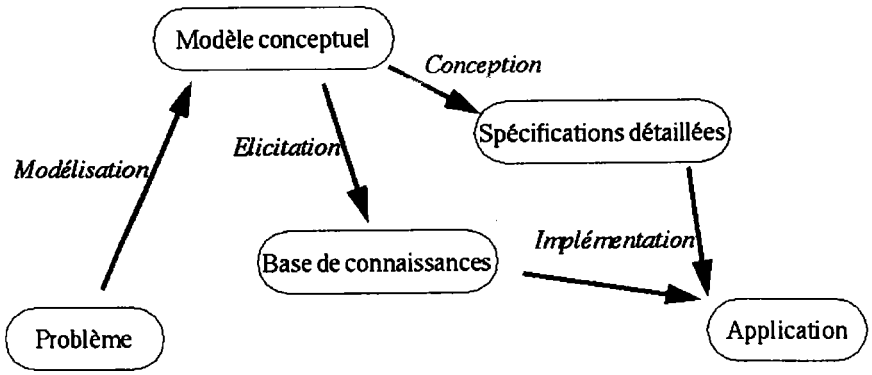


Figure 1.2 Conception d'un SBC selon une approche *knowledge level*

3.5 Exemple de méthode de conception d'un système à base de connaissances

Dans cette partie, nous allons décrire les systèmes à base de connaissances que nous considérons comme étant les plus représentatifs

Le système KADS

Le système KADS (knowledge acquisition and design structuring) est une méthodologie de conception de SBC (système à base de connaissances) qui a été développée à partir des travaux de l'université d'Amsterdam [Wielinga, 86, 92, 93] dans le cadre du projet ESPRIT. La conception d'un SBC selon la méthode KADS, se fait à travers la construction d'une succession de modèles, chacun de ces modèles s'intéresse à un aspect particulier du système à construire.

Le modèle de l'organisation décrit le contexte organisationnel dans lequel viendra s'insérer le système. Le modèle de la tâche décrit la tâche globale à réaliser, ainsi que la répartition des sous tâches aux différents agents. Les modèles des agents décrivent les compétences de chacun de ces agents. Le modèle de communication décrit les interactions entre les agents et en particulier les interactions entre le système et l'utilisateur. Le modèle de conception contient les spécifications détaillées du système. Enfin le modèle d'expertise décrit le processus de résolution de problèmes d'un agent selon trois niveaux :

1 - Le niveau domaine qui décrit les concepts du domaine que manipule le système et ses relations.
2 - Le niveau inférence décrit les inférences qui relient les métaclasses ou rôles.
3 - Le niveau tâche associe à un but un enchaînement d'inférence en imposant l'ordre dans lequel il doit être appliqué et la stratégie utilisée (dirigée par les données, par les buts, etc.).

Le système COMMET
Le système COMMET est le résultat des travaux de L. Steels à l'université de Bruxelles, en particulier dans le cadre du projet Esprit [Steels, 91, 92, 93]. COMMET permet de décrire une application selon trois perspectives : les tâches, les modèles du domaine et les méthodes. Plusieurs modèles organisent et structurent ces entités.

La structure de la tâche explicite la décomposition des tâches en sous tâches. Les relations entre modèles du domaine et tâches sont décrites dans un diagramme de dépendance entre modèles, et les diagrammes de contrôle permettent de représenter le contrôle qu'une méthode impose aux différentes tâches. La modélisation d'une application se fait par itérations successives selon les trois perspectives décrites auparavant.

A ce système est associé un outil le WORKBENCH dont la vocation est d'apporter une aide non seulement dans la phase de modélisation d'une application, mais aussi dans le cycle complet de réalisation. La génération d'une application exécutable à partir de sa description au *knowledge level* nécessite qu'à chaque objet du *knowledge level* (modèle, méthode, etc.) corresponde un objet au niveau symbolique.

Comme nous l'avons vu, les systèmes experts dits de la deuxième génération utilisent deux moyens pour dépasser les limites des systèmes experts classiques : faire coopérer plusieurs modèles ou méthodes de résolution de problème ou bien adopter une approche *knowledge level.* Ces deux moyens se rejoignent par l'importance qu'ils accordent à une meilleure modélisation du domaine, à la compréhension du rôle qu'un type de connaissances particulier peut jouer (et comment plusieurs types de connaissances différents peuvent se compléter) et enfin par l'importance qu'ils accordent à une meilleure modélisation du processus de résolution.

Pour compléter cette étude, nous allons décrire quelques systèmes experts hétérogènes plus spécifiquement dédiés à la C.A.O.

3.6 Les systèmes experts dédiés à la C.A.O

3.6.1 Exemples de systèmes de C.A.O dirigé par l'I.A

Le système ICAD [Monceyron, 90a] par exemple est issu de la même école, celle de la programmation objet. Il offre aux concepteurs la possibilité de définir un modèle explicite correspondant à l'objet en cours de conception. Ce modèle peut ensuite être manipulé pour générer des représentations externes diverses, comme par exemple du 3D, des données pour des codes de calcul, des données pour des modeleurs volumiques etc. Ils permettent de générer des liens inter-objets de la forme "est composé de", "contient", etc. Ils possèdent, par ailleurs, un ensemble de primitives dédiées à la C.A.O mécanique.

L'inconvénient de ce type de système est l'aspect rebutant du langage qui reste de bas niveau, néanmoins la nature extrêmement dynamique de la programmation objet convient parfaitement à la C.A.O.

3.6.2 Exemples de systèmes d'I.A dirigé par la C.A.O

Pro-Engineer de Parametric Technology et MTEL de Caroline informatique sont des exemples types de cette approche.
Pro-engineer permet de gérer des variantes de conception d'objets et de revenir sur sa conception. Il rappelle d'une certaine manière les fonctionnalités de SMECI de gestion des hypothèses et mémorisation des opérations des utilisateurs afin de revenir éventuellement sur les différentes solutions.

MTEL possède un module I.A qui est à base de PROLOG. Il permet à des utilisateurs de développer des applications qui utilisent les objets du modèle en cours de définition et les méthodes associées à ces objets.
Bien que limitées, ses possibilités de raisonnement doivent permettre de réaliser de petites applications correspondant à des problèmes simples.

3.6.3 Exemples de systèmes mixtes

Le premier système, ARCHIX [Delmas 90], est un exemple de couplage avec un logiciel de C.A.O traditionnel (EUCLID-IS). Le second, DIADEME [Samin 90], est intéressant par la grande quantité de connaissances qu'il comporte. Le troisième, EXPORT [Monceyron 90b], combine différents modes de connaissances et différents modes de raisonnement.

Le système ARCHIX
ARCHIX est un système expert de conception de véhicule en phase d'avant projet.
Il résout trois types de problèmes :

- gérer les contraintes 3D,
- gérer la combinatoire au niveau de la valeur de paramètres,
- maintenir la cohérence globale de la solution (résout les problèmes de conflit sur un paramètre).

Le premier point est résolu par l'utilisation d'EUCLID-IS. Lorsque le système désire vérifier si une contrainte est satisfaite, il appelle le module EUCLID-IS correspondant, en fournissant les bonnes valeurs en entrée. Pour cela, une modélisation des différents modules de calcul a dû être réalisée dans le système expert.

Le second point est résolu par l'emploi d'heuristiques codées sous la forme de règles de production. Ces règles sont organisées en "blocs", dont l'enchaînement est défini à l'avance (sorte d'algorithme de conception). Chaque bloc dispose de son environnement d'exécution (valeurs de paramètres d'entrées) ce qui évite au système de déclencher deux fois le même bloc avec les mêmes entrées.
Le dernier point repose sur la capacité du système de revenir sur ses décisions. Lorsqu'un bloc détermine la valeur d'un paramètre et que cette valeur contredit la précédente, un module de gestion de conflit détermine le bloc qui lui semble le plus suspect et lui demande de recommencer son calcul.

ARCHIX est un système d'aide à la conception qui intègre des modules de conception automatique. Il laisse néanmoins le concepteur libre d'accepter ou de refuser les recommandations du système.

Le système DIADEME

DIADEME est un système de configuration de machines électriques, il résout les problèmes de devis en les automatisant.

Il fonctionne dans une architecture distribuée : par exemple un utilisateur saisit le cahier des charges en vérifiant sa cohérence, et l'envoie, via le réseau, à une station de travail centrale. Celle-ci effectue la conception et renvoie le résultat à l'utilisateur qui calcule les coûts avant de fournir les résultats.

Le système EXPORT

EXPORT est un système multi-expert dédié à l'ingénierie portuaire. Il intègre les composants suivants :

- un système expert de conception de digue : EXDIG,
- un système expert d'aide à la configuration du plan masse du port : EXPLAN,
- un couplage avec un modeleur surfacique : MOSS.
- un système expert d'aide à l'utilisation d'un code de calcul aux éléments finis d'hydrodynamique et de transport de sédiments. L'ensemble doit coopérer autour d'une structure de données de type tableau noir.

EXPORT possède les caractéristiques suivantes :

- Le raisonnement de chaque système expert est décomposé hiérarchiquement et récursivement en sous-problèmes. Dans certains cas la résolution d'un sous problème peut nécessiter une expertise de nature différente de celle nécessaire à la résolution d'un autre sous-problème, on parle alors de système multi-expert récursif.
- Il permet de trouver plusieurs solutions à un même problème.
- L'interface graphique d'EXPORT est associée à l'état courant de la solution, si on change d'état ou de solution, le système maintient la cohérence graphique avec l'état courant.
- L'ensemble des informations descriptives est modélisé sous forme d'objets.
- Le modeleur surfacique MOSS est utilisé pour effectuer plusieurs tâches : récupérer le modèle numérique du terrain, mettre en place les surfaces selon les profils en travers fournis par EXPORT, faire le calcul des courbatures, et permettre la visualisation du modèle conçu.

3.7 Conclusion

Les systèmes de conception à base de connaissances se révèlent très bien adaptés dans le cadre d'applications expertes, comprenant une forte part de savoir-faire où le processus de conception est relativement stable et bien maîtrisé. Il permet aussi d'automatiser les phases répétitives de conception, l'utilisateur restant en charge des tâches plus créatives et des choix finaux de conception.

Les systèmes experts actuels appartiennent à la deuxième génération (SE2G système expert de deuxième génération) où différents modèles du domaine et différentes méthodes de résolution sont nécessaires selon le type de problème considéré. Les travaux sur les SE2G s'articulent autour de deux principales approches, qui mettent respectivement l'accent sur la coopération

entre plusieurs types de connaissances et plusieurs méthodes de résolution d'une part, et sur l'utilisation du *knowledge level* d'autre part.

Cependant, il reste à donner à ces systèmes l'interface qu'ils méritent c'est à dire intégrer des mécanismes de dialogue Homme/Machine qui soient aussi performants que les modèles conceptuels qui sont en train de voir le jour avec les SE2G. Pour cela, il faut répondre à deux problèmes essentiels :

- Prendre en compte les intentions du concepteur, et proposer des alternatives à chaque fois que cela est nécessaire ; ceci dans l'objectif d'améliorer la qualité de l'interaction.
- Fournir des fonctionnalités intelligentes, c'est-à-dire des mécanismes d'inférence pour que même un concepteur non spécialiste puisse concevoir son application convenablement. Cette deuxième fonction existe avec plus ou moins de succès dans certains systèmes dits "intelligents".

En tout état de cause, il nous semble que l'étude de l'interactivité Homme/Machine devrait être faite selon ces deux optiques.

4.L'interactivité Homme-Machine

4.1 Introduction

Le but de notre travail est l'intégration d'outils conviviaux dans le cadre de la coopération entre un utilisateur et un système de C.A.O, cela implique d'une part de prendre en compte les facteurs humains qui interviennent dans le processus de conception que l'on désigne par modèle du processeur humain, et d'autre part de mettre en adéquation ces mécanismes avec les spécificités de la C.A.O.

4.2 Le modèle du processeur humain

Le modèle du processeur humain présente le sujet humain comme un système de traitement de l'information. Ce système comprend trois sous-systèmes interdépendants (sensoriel, moteur, et cognitif) munis chacun d'un processeur et d'une mémoire dont les caractéristiques sont les cycles de base, la capacité et la persistance). Une référence par rapport au modèle du processeur humain est G.O.M.S (Goal Operator Method Selection) dont nous allons décrire les composants.

4.2.1 Le modèle G.O.M.S

G.O.M.S (Goal Operator Method Selection) [Newel, 82] est un modèle de description qui prend comme hypothèse le caractère adaptatif du sujet humain. G.O.M.S introduit quatre ensembles pour représenter l'activité cognitive d'un individu engagé dans la réalisation d'une tâche : les buts, les opérateurs, les méthodes et les règles de sélection.

1 - Un but est une structure symbolique qui définit un état recherché. Les buts sont organisés d'une manière hiérarchique. Ainsi, un but complexe est atteint lorsque plusieurs sous-buts sont satisfaits et ceci récursivement jusqu'à atteindre les buts élémentaires. Les buts élémentaires sont réalisés par une suite d'opérateurs.

2 - Un opérateur est une action élémentaire dont l'exécution provoque un changement d'état.

3 - Une méthode décrit le procédé qui permet d'atteindre un but. Elle s'exprime sous la forme d'une suite conditionnelle de buts et d'opérateurs. Les conditions font référence au contenu de la mémoire à court terme et à l'état de l'environnement.

4 - Une règle de sélection exprime le choix d'une méthode lorsqu'il y a conflit. C'est-à-dire lorsque plusieurs méthodes conduisent au même but. Par exemple, pour un éditeur pleine page qui permet à la fois les commandes à la "EMACS" et les commandes à la "WORD", il existe plusieurs façons de déplacer le curseur, soit avec le clavier, soit avec la souris soit une combinaison des deux. Le choix entre les méthodes peut s'exprimer en fonction de la distance entre la position actuelle du curseur et la localisation visée.

Parallèlement à ces notions de but, méthode, opérateur et règle de sélection, G.O.M.S peut-être considéré comme un générateur d'une famille de modèles. Un niveau de modélisation se définit essentiellement par le temps d'exécution des opérateurs. Quatre niveaux d'analyse interviennent : le niveau tâche, le niveau fonctionnel, le niveau argument et physique.

Le niveau tâche structure l'espace de travail en hiérarchie de sous-tâches elles-mêmes structurées en fonction. Chaque fonction se réalisant par une suite de commandes. Le niveau physique décrit en terme d'action physique la spécification des commandes.

Evaluation de G.O.M.S

Apport

1 - La modélisation d'une tâche peut-être, soit raffinée, soit élaborée à partir de constituants élémentaires.

2 - G.O.M.S fournit au concepteur un support d'évaluation prédictive des performances. En effet, connaissant le temps d'exécution de chaque opérateur, il est possible de prédire les temps nécessaires à la réalisation d'une tâche.

Limites

1 - Dans le cas d'une prédiction, cela ne concerne qu'un utilisateur qui ne commet pas d'erreur, or, l'erreur est inévitable et le traitement des erreurs est omniprésent y compris dans les cas simples. Dans le cas du sujet humain, le traitement d'une erreur peut se voir comme la réalisation d'une tâche particulière et par conséquent lui correspond un temps de résolution qui devrait se greffer sur le reste du plan.

2 - G.O.M.S reste réducteur et ne décrit qu'un aspect limité des mécanismes cognitifs : celui de la résolution de problèmes déjà résolus. Un modèle issu de G.O.M.S (présentant néanmoins des conditions restrictives) a été proposé. Il s'intitule KEYSTROKE.

4.2.2 Le modèle KEYSTROKE

Le problème que se propose de résoudre KEYSTROKE se formule ainsi :

Soit :
- une tâche (constituée éventuellement de plusieurs sous-tâches),
- le langage de commandes de système,
- les paramètres caractéristiques des capacités motrices de l'utilisateur,
- les paramètres des temps de réponse du système,
- la méthode de réalisation de la tâche.

But : Prédire le temps d'exécution de cette tâche par un utilisateur expert.

Deux restrictions par rapport à G.O.M.S :
1 - Pas de choix de méthode : la méthode est donnée.
2 - Pas de prédiction : KEYSTROKE évalue le temps d'exécution et non pas le temps total de l'accomplissement d'une tâche.

KEYSTROKE se situe au niveau des aspects lexicaux et syntaxiques de l'interaction. De plus la méthode étant imposée par le système ce qui implique que les notions de but et de règle sont absentes dans KEYSTROKE et n'utilise de ce fait que deux entités : les opérateurs et les méthodes.

Evaluation de KEYSTROKE

Avantages
KEYSTROKE permet de comparer ou de prédire les performances de l'utilisateur en fonction d'une syntaxe donnée. De plus la terminologie et la technique de modélisation de KEYSTROKE est compréhensible par un non spécialiste (cette compréhension reste cependant relative).

Limites
KEYSTROKE concentre son analyse sur des aspects purement lexicaux et syntaxiques sans tenir compte de la problématique générale de l'utilisateur. En effet, le comportement d'un utilisateur n'est pas seulement dirigé par des considérations d'efficacité lexicales locales, il l'est aussi par une logique globale.

Un autre point de vue dans le cadre du modèle du processeur humain est l'analyse psychologique qui conduit certains comportements de l'utilisateur de la machine. Cette théorie est désignée sous le nom de théorie de l'action.

4.3 Théorie de l'action

4.3.1 Introduction

Un acteur Humain présente deux particularités qui peuvent à la fois présenter des avantages et des inconvénients du point de vue de la réalisation d'une interface.

Tout d'abord, un utilisateur est adaptable; il est capable de comprendre et de réagir en conséquence. Ensuite il est créatif. Il est impossible de décrire de façon exhaustive ses comportements possibles, car il souhaite pouvoir créer, face à un problème à résoudre, sa propre logique d'utilisation. Lui permettre de la suivre tout en respectant les impératifs du logiciel constitue un des points les plus difficiles des interfaces Homme-Machine.
La théorie de l'action tente de trouver des réponses à cette problématique. Elle est caractérisée par deux aspects : l'aspect modèle conceptuel, et l'aspect structure de processus de manipulation de modèle.

4.3.2 Le modèle d'activité

Norman [Norman, 86], propose un modèle global de l'activité humaine, qui se base sur l'hypothèse selon laquelle l'individu élabore des modèles conceptuels qui constituent les données essentielles déterminant le comportement. Toute action y est analysée selon un cycle chronologique à trois phases : (1) objectifs à atteindre, (2) actions à effectuer pour atteindre l'objectif, (3) évaluation du résultat par comparaison entre l'état du monde (ou lorsque celui-ci est représenté par un système interactif, état du système) et l'objectif à atteindre. Ce cycle est illustré par la figure 1.3.

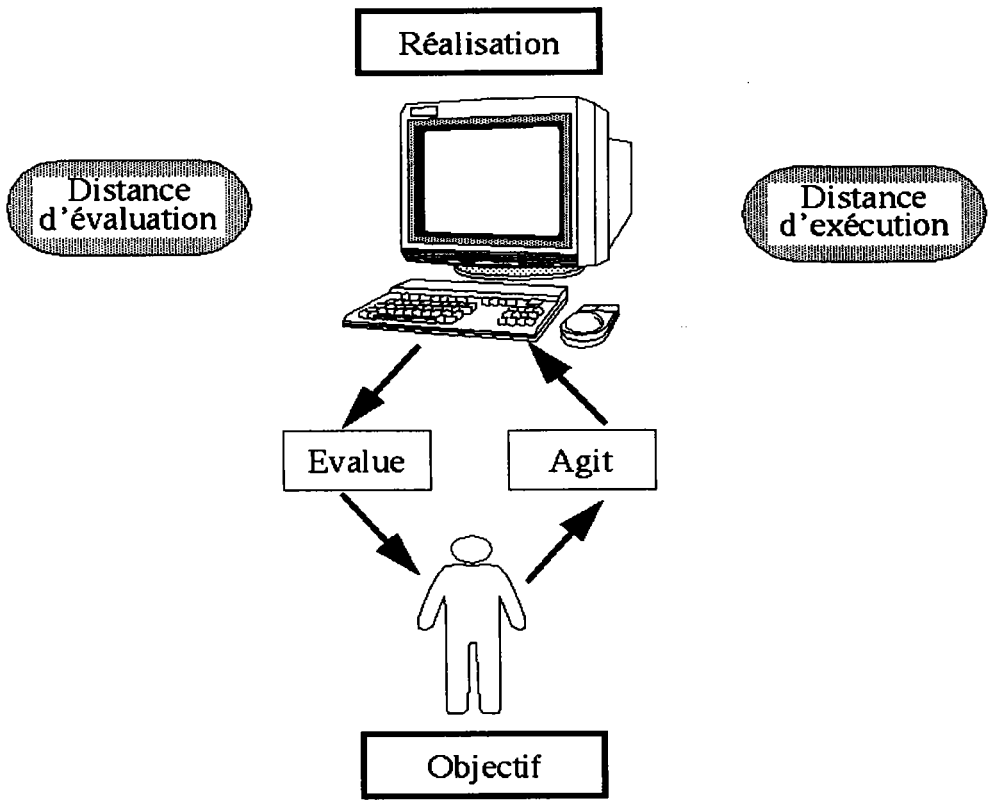


Figure1.3 Cycle d'activité

Nous allons dans la suite préciser les composants de ce modèle et tirer les conclusions sur la nature des tâches devant être supportées par le système. Les notions de distance d'exécution et distance d'évaluation seront décrites à part en 4.3.3.

4.3.3 Le modèle conceptuel

Un modèle conceptuel est la représentation mentale d'un individu, elle dépend de la connaissance déjà acquise et de la compréhension de la situation présente. Il est évolutif, incomplet, mais influe fortement sur son comportement. Lorsque l'on parle d'interface utilisateur on distingue deux formes de modèle : le modèle de conception, et le modèle de l'utilisateur.

a- Le modèle de conception
Ce modèle doit résulter d'une étude approfondie des besoins, des possibilités et des limitations d'un utilisateur type. Il dépend des caractéristiques propres du domaine à modéliser, mais également d'un utilisateur type.
Il s'exprime par des variables Vi qui permettent de représenter, par l'intermédiaire de l'interface, l'état du système, et par des commandes Ci qui permettent d'agir sur cet état.

b- Le modèle de l'utilisateur
C'est la représentation mentale que l'utilisateur a des outils mis à sa disposition. Le but est de permettre une adéquation aussi complète que possible entre la représentation mentale de l'utilisateur et le modèle de conception. Il résulte d'une interprétation du résultat des actions qu'il effectue au cours de ses différentes sessions et, le cas échéant de la documentation associée à l'outil. Il s'exprime par des variables Vy, qui constituent la représentation mentale, pour l'utilisateur, de l'état du système, et par des actions Cy, susceptibles d'être réalisées par l'intermédiaire de celui-ci.
L'image joue un rôle important dans la mise en correspondance de ces deux modèles. c'est elle qui permet à l'utilisateur d'élaborer son propre modèle. Le rôle de l'utilisateur est illustré par la figure 1.4.

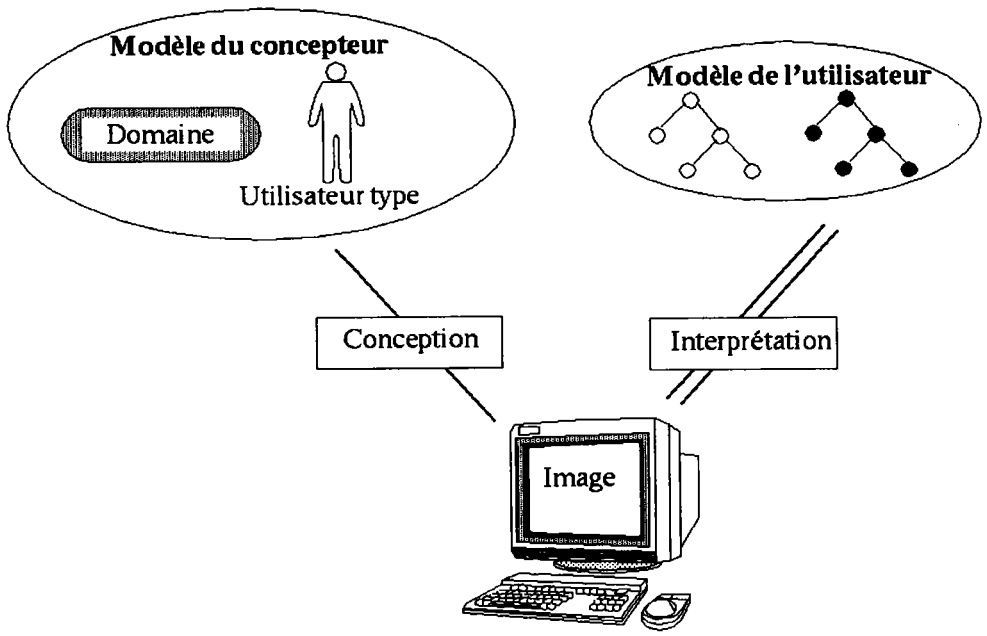


Figure 1.4 Rôle de l'image dans le modèle de conception

4.3.4 Structure du processus de manipulation de modèle

Ce processus se situe dans la réalisation d'une tâche et met en jeu sept activités [Norman, 86] :

- L'établissement du but : Un but correspond en fait à un état à atteindre, ce qui se traduit pour l'utilisateur par l'élaboration d'une représentation mentale de cet état.
- La formation d'une intention : L'utilisateur compare l'état du système avec sa représentation mentale du but à atteindre, pour dégager une intention d'action.
- La spécification d'une suite d'actions : L'intention est décomposée en suite d'actions, dont l'utilisateur imagine la séquence. Cette phase est la plus complexe, puisqu'elle nécessite, pour chaque sous-action, de connaître l'adéquation entre le modèle mental et réalité du système, tant au niveau de l'état (les données) que des actions (dispositifs physiques).
- L'exécution des actions : Elle s'effectue selon la séquence imaginée ci-dessus.
- La perception de l'état du système : Cette phase de perception dépend évidemment de la réaction du système : selon le comportement de celui-ci, la perception de l'état du système peut être globale et immédiate. Dans le dernier cas, le système répond à chaque action tout au long de son exécution, alors que dans le second cas, il répond globalement à la fin de l'action, ou de la séquence d'actions.
- L'interprétation de l'état du système : Cette phase correspond à la reconstruction du modèle de l'utilisateur à partir de l'état qu'il perçoit.
- L'évaluation de l'état du système par rapport au but : Au cours de cette phase, l'utilisateur cherche à comparer la représentation qu'il se fait du nouvel état du système par rapport à la représentation mentale qu'il avait imaginée. Selon la proximité ou non de ces deux représentations, de nouveaux buts et intentions peuvent être dégagés.

L'intérêt de cette classification est d'identifier les points critiques, notamment la distance entre les variables psychologiques et les variables logiques. En effet, cette distance complique quelquefois les opérations de traduction de l'utilisateur lorsqu'elle est importante.

4.3.5 Distance d'exécution/distance d'évaluation

L'analyse de la tâche selon ces activités a conduit Norman à introduire une notion de distance, qu'il a séparée en deux aspects : (1) la distance d'exécution et (2) la distance d'évaluation. Ces deux concepts ont été représentés dans la figure 1.3
La distance représente la mesure de la différence entre les modèles conceptuels du concepteur de l'outil et de l'utilisateur, c'est-à-dire entre (Vi,Ci) et (Vy, Cy). Plus cette distance est courte, mieux le système reflète les intentions de l'utilisateur, tant dans ses moyens que dans ses résultats.

La distance d'exécution correspond à l'effort requis par l'utilisateur pour passer des intentions (exprimées en termes de Cy) aux séquences d'action (devant se définir en termes de Cj).
Cette distance caractérise la difficulté de la phase3. La diminution de cette distance suppose que les commandes (Ci) disponibles au niveau du système soient les plus proches possibles des actions abstraites (Cy), en termes desquels l'utilisateur exprime naturellement son intention tant en nature qu'en ordre d'invocation.

La distance d'évaluation mesure la charge mentale requise par l'utilisateur pour passer de la perception de l'état du système en termes de variables d'interface Vi à son propre modèle mental exprimé en termes des Vy.

4.3.6 Analyse en buts et sous-buts

La mise en œuvre de la démarche de Norman n'est immédiate en l'état que si le but à atteindre est relativement facile à obtenir avec les actions disponibles. Si cela n'est pas le cas, une décomposition récursive est effectuée. C'est ce que l'on appelle l'analyse en buts et sous-buts. L'analyse en buts et sous-buts consiste donc en une décomposition des objectifs à atteindre en sous-objectifs, jusqu'à atteindre des buts suffisamment "modestes" pour être atteints par des actions disponibles dans le système. Cette décomposition est exprimée par la structure arborescente représentée par la figure 1.5.

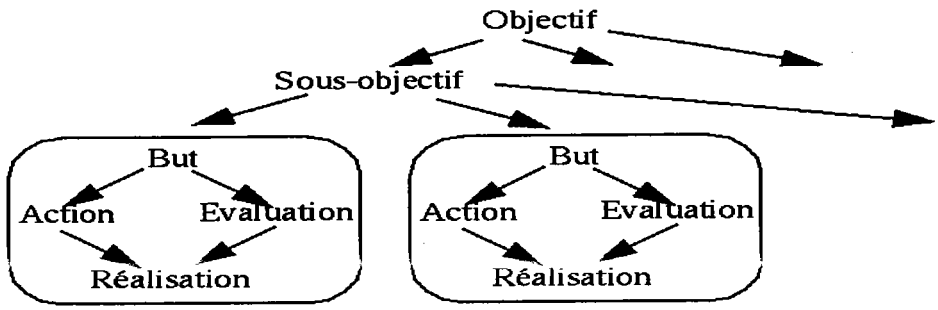


Figure 1.5 Structure arborescente des buts et sous-buts

4.3.7 Evaluation

La théorie de l'action complète les modules G.O.M.S et le module du processeur humain. Elle précise la notion d'état, elle prend en compte les erreurs, explique les difficultés de l'utilisateur et justifie l'utilité de la notion de modèle conceptuel. On explique plus finement ces points :

- La notion d'état est précisée dans le sens où l'on distingue l'état effectif de l'état perdu. Le premier est lié aux variables physiques caractéristiques du modèle, le second est lié aux variables dites psychologiques caractéristiques du modèle conceptuel de l'utilisateur.
- Une erreur est commise soit parce que les besoins de l'utilisateur sont mal satisfaits ou pas satisfaits du tout, soit parce que l'image n'explicite pas l'état du système, soit enfin que les dispositifs de contrôle des variables physiques ne sont pas adaptés à la tâche. La théorie de l'action essaie d'identifier les sources d'erreurs et d'expliquer les difficultés rencontrées.
- Le module conceptuel de l'utilisateur traduit l'aspect "système adaptatif" du système, à la fois, de l'utilisateur au système (accommodation) et du système à l'utilisateur (assimilation).

4.3.8 *Conclusion*

Cette étude a posé les bases des mécanismes adaptatifs de l'interaction Homme/Machine dans le processus de conception. La théorie de l'action est l'illustration formelle de ces mécanismes. A partir de là, et prenant en compte les objectifs définis au préalable, l'outil qui nous semble le plus intéressant pour arriver à ces objectifs et qui reprend les grands principes à la fois des modèles des systèmes experts et de la théorie de l'action est la planification de l'action en général et la planification interactive en particulier.

5 La planification

5.1 Introduction

Un générateur de plan d'action ou planificateur est communément défini comme étant la séquence d'opérations élémentaires permettant d'atteindre un objectif à partir d'une situation d'origine sans intervention extérieure. Ainsi l'intelligence artificielle aborde le problème de la planification en définissant :

- un modèle du monde,
- un ensemble d'actions, d'opérateurs modifiant l'état du monde,
- un état initial et un état final (ce dernier n'étant éventuellement que partiellement décrit).

L'activité de la planification consiste donc à décider d'une suite d'actions destinées à faire passer le monde de l'état initial à l'état final comme le montre la figure 1.6.

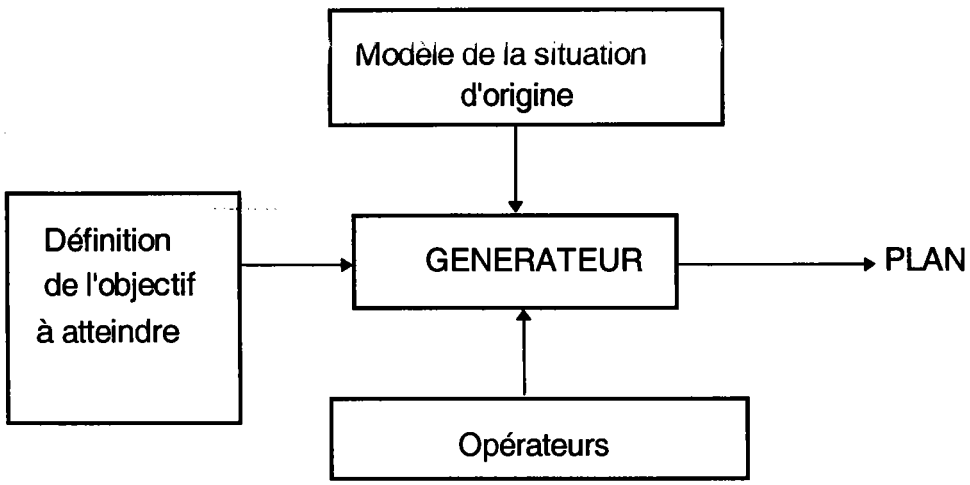


Figure 1.6 : Structure d'un générateur de plans

5.2 Caractéristiques du processus de planification

5.2.1 Le processus de planification

La planification est un processus en deux temps : la génération d'un plan et l'exécution/rectification du plan généré.

La génération de plan se modélise comme une recherche dans un espace de problème dont les éléments sont une situation initiale, un ensemble de situations finales, et des opérateurs permettant de passer d'une situation à une autre. Un espace de problème est représenté comme un arbre de recherche dont chaque nœud représente l'état d'un plan à un moment de son développement et la situation correspondante d'un modèle du monde. La racine représente un plan nul, auquel est associée la situation initiale.
Chaque chemin issu de la racine est l'évolution d'un plan susceptible d'être généré à partir de la situation initiale. Sur fond de ce paradigme, planifier consiste à effectuer une recherche dans l'espace des plans possibles afin de trouver un qui conduise à la situation souhaitée. On planifie d'autant plus intelligemment qu'on minimise cette recherche.

Avant d'aller plus loin il paraît nécessaire de clarifier un problème fondamental posé en planification connu sous le nom de "frame problem".

5.2.2 Le frame problem

Les premiers travaux concernant la planification utilisaient les techniques de démonstrations automatiques de théorèmes se fondant sur la logique du premier ordre. Ainsi, on essayait d'introduire la notion temporelle de changement. Dans un tel formalisme, la description du monde en évolution se faisait à l'aide d'un prédicat comportant une variable situationnelle.

L'inconvénient de ce formalisme est sa lourdeur. En effet, la manipulation de chaque situation ou état de l'univers est coûteuse, car elle nécessite de noter explicitement tout ce qui change ou ne change pas lors de l'application d'un opérateur, c'est le *frame problem.*

Une solution à cette difficulté est de donner une description complète de l'état initial puis seulement les modifications dues aux opérateurs qui sont appliqués (ou la succession des opérateurs eux-mêmes). On résout de cette façon le problème de place et celui de temps de recopiage. C'est une solution commode si l'on n'a jamais à supprimer l'action d'un opérateur, comme dans le cas d'un retour en arrière dans le parcours du graphe des états. Dans cette dernière situation, il est nécessaire de recourir à des méthodes particulières pour retrouver l'état précédent l'application d'un opérateur dont on veut annihiler l'effet.

5.3 L'apprentissage dans les systèmes de planification

Un des problèmes majeurs associés à la mise en service des systèmes de planification concerne la maintenance. D'une part, un planificateur, même à vocation générale, porte souvent la marque du domaine au sein duquel il a été conçu et développé. Son transfert à d'autres domaines peut conduire à des performances bien en-deça de celles observées dans le domaine d'origine. D'autre part, à l'intérieur même d'une application donnée, des changements tout à fait compréhensibles dans l'environnement de travail peuvent avoir des incidences sur l'efficacité du système. D'où la nécessité d'une surveillance et d'une adaptation constante du système pour assurer des performances correctes.

Une des solutions préconisées consiste à construire un planificateur capable de s'adapter aux spécificités des diverses applications et d'améliorer ses performances au gré de ses expériences, c'est la notion même de planificateur doté de capacités d'apprentissage. L'objectif de l'apprentissage est donc de remplacer progressivement la recherche aveugle par des connaissances accumulées à travers l'expérience en vue d'aboutir rapidement à un plan satisfaisant, voir optimal.

5.4 Classifications des modèles de planification

5.4.1 Introduction

Dans la suite de cette étude concernant la classification, et prenant en compte les différents aspects cités plus haut, nous avons volontairement choisi de classer les modèles de planification selon deux optiques :

1 - modèles classiques
2 - modèles avec apprentissage

5.4.2 Les modèles de planification classique

La classification des modèles de planification classique, c'est-à-dire sans apprentissage est globalement subdivisé en deux catégories [Haton & al, 91]:

- les approches générales, relativement indépendantes du domaine d'application grâce à l'utilisation de représentation de connaissances et de raisonnement,
- les approches spécifiques, fondées sur des heuristiques du domaine et peu transportables à un autre secteur d'activité.

Cette catégorie regroupe l'essentiel de la recherche dans le domaine de la planification, et des prototypes existent et sont opérationnel, notamment GPS, FORBIN, etc...

5.4.3 Les modèles de planification avec apprentissage

On peut tenter une taxonomie des planificateurs capables d'apprendre suivant le type des connaissances distillées de l'expérience pour améliorer leur performances.
Dans une première catégorie, les efforts se dirigent vers le stockage de plans éprouvés en vue de leur réutilisation future. Moyennant quelques variations sur ce thème, les connaissances ainsi générées sont diversement appelés plans généralisés [Carbonell 83], des macro-opérateurs [Tenenberg, 86], ou des épisodes [Kibler, 83].

Une deuxième approche consiste à mémoriser des erreurs de planification et leurs solutions correspondantes. La planification suit un cycle de génération et de rectification.

Une troisième approche vise à créer des heuristiques de contrôle, des règles permettant de focaliser directement sur le bon choix à chaque bifurcation dans l'arbre de recherche [Minton, 88], [Hilario, 89].

5.5 Le problème de la complexité

La complexité de la recherche du plan est abordée par rapport aux algorithmes de planification dont la seule contrainte est de choisir parmi tous les opérateurs applicables ceux dont la «somme» des postconditions ajoutée à l'état initial donne l'état final, sans heuristique de choix supplémentaire [Fikes et al., 72], [Chapman, 87], [Puget, 89].
Pour un algorithme de planification utilisant l'algorithme optimal de recherche itérative en profondeur d'abord (depth-first, iterative deepening) [Korf, 85], la complexité en nombre

d'opérations est en $O(N^{PS})$ où N est le nombre d'opérateurs de la solution la plus courte et où PS représente le nombre moyen d'opérateurs applicables à un état donné.

La planification hiérarchique avec le système ABSTRIPS [Sacerdoti, 74] est basée sur le principe d'une planification à travers un espace réduit d'état, à l'aide d'opérateurs «abstraits», mais l'abstraction d'un opérateur n'est pas mesurée par la globalité de son action, mais par le degré de contrainte de ses préconditions.

A chaque précondition d'un opérateur est associée une valeur qui indique son niveau de criticalité pour la planification. La planification se décompose en autant d'étapes qu'il existe de niveaux de criticalité. La première étape de planification consiste donc à construire une séquence d'opérateurs qui appliquée à l'état initial, donne l'état final, mais où seules les préconditions de criticalité maximales (de niveau 1) sont considérées. Toutes les préconditions de niveau de criticalité inférieur à 1 sont négligées. Les opérateurs sélectionnés ne sont pas réellement applicables dans leur contexte d'application puisque leurs préconditions non critiques ne sont pas nécessairement vérifiées. Au niveau 2, ABSTRIPS ajoute devant chaque opérateur sélectionné à l'étape précédente, la séquence d'opérateurs qui permettra à leurs préconditions de niveau 2 d'être vérifiées, tout en ne considérant pour les opérateurs des séquences ajoutées, que les préconditions de niveau 1 et 2 et ainsi de suite jusqu'à ce que toutes les préconditions de tous les opérateurs de la séquence soient satisfaites.

La planification consiste donc tout d'abord à trouver quelques opérateurs disséminés dans ce qui sera la séquence finale d'opérateurs, puis à chaque étape à compléter un peu plus la séquence en ajoutant de nouveaux opérateurs devant ceux qui sont sélectionnés déjà.

Exemple
La figure suivante illustre le processus. Les états sont représentés par des rectangles, les opérateurs par des ronds, un opérateur relié à un état Ei par une flèche sortante et à un autre état Ef par une flèche entrante signifie que le plan calculé par ABSTRIPS prévoit que l'opérateur soit appliqué dans l'état Ei et que l'état Ef en résultera. Supposons que le nombre de niveaux d'abstraction soit de trois.

Au premier niveau de planification ABSTRIPS sélectionne l'opérateur 1 comme applicable dans le contexte initial, c'est ce que représente la flèche en pointillé, en négligeant les préconditions de l'opérateur 1 qui ne sont pas de niveau de criticalité maximum.
Au deuxième niveau ABSTRIPS considère les préconditions de l'opérateur 1 de niveau de criticalité 2 qui ne sont pas vérifiées dans l'état initial et recherche la séquence d'opérateurs qui appliquée à l'état initial permettra de les vérifier. Il crée la séquence des opérateurs 01, et 03 pour lesquels les préconditions de niveau 1 et 2 sont vérifiées respectivement dans le contexte initial, et le contexte 0.2.

Au troisième niveau, il considère toutes les préconditions non encore satisfaites dans les opérateurs sélectionnés jusque là et insère dans le plan, devant chaque opérateur, une nouvelle séquence d'opérateurs qui assure que les préconditions de niveau 3 seront bien vérifiées avant que l'opérateur ne soit appliqué. Ainsi par exemple, la séquence des opérateurs 011 et 012 appliquée à l'état initial produit l'état pour lequel l'opérateur 01 est *réellement* applicable. Pour produire l'état 1 à partir de l'état initial et rendre l'opérateur 1 applicable, il a fallu 2 niveaux de séquences d'opérateurs. La hiérarchie d'abstraction de ABSTRIPS concerne donc les faits dans les états de l'espace.

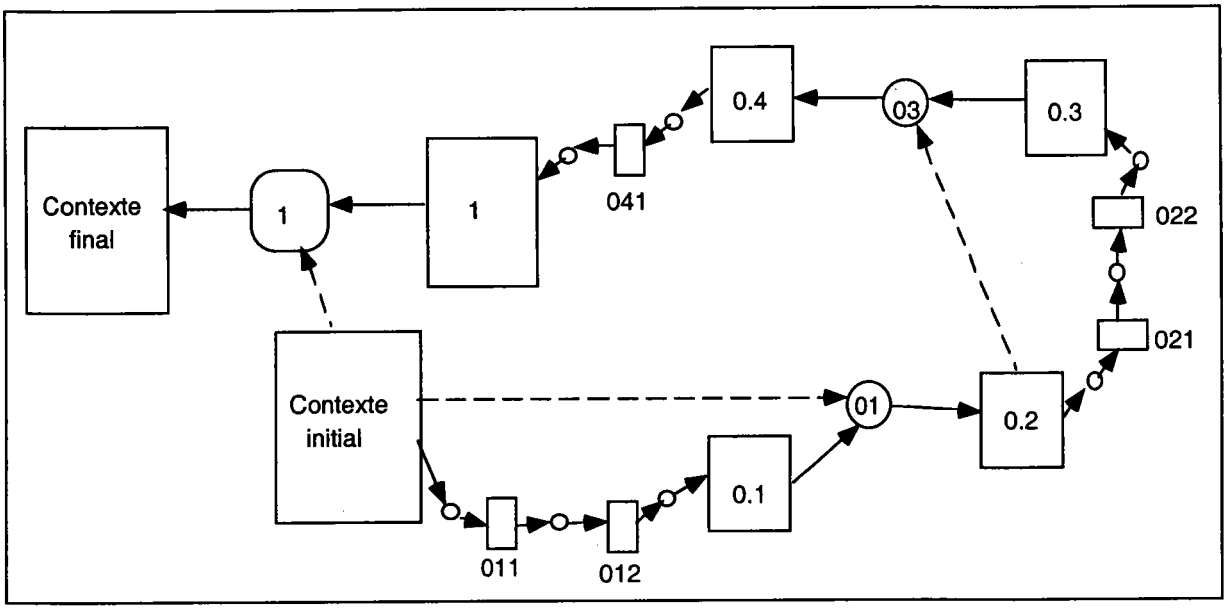


Figure 1.7 Hiérarchie d'états

5.6 Quelques thèmes de recherche en planification

5.6.1 *Le temps*

Tous les systèmes de planification décrivent un plan comme étant une succession d'actions. Cependant la durée des actions n'est pas représentée. On pourrait ainsi écrire que l'univers passe instantanément d'un état antérieur à un état postérieur. D'autre part, la simultanéité possible des actions n'est pas représentée, ainsi les planificateurs ne représentent pas ce parallélisme. NOAH ne fait pas exception à cette règle, car il ne permet pas une exécution parallèle des actions, mais plutôt une indépendance logique de certaines branches du plan.
Néanmoins des tentatives de représentation explicite du temps on été faites sur des systèmes de planification. Ces représentations (selon la nature des problèmes abordés) se dégagent en deux parties :
- un formalisme de planification et la manière dont il intègre les objets temporels,
- un formalisme de représentation du temps.

a - Le formalisme de planification

On peut distinguer deux familles d'approche :
- l'utilisation de planificateurs classiques,
- l'utilisation de planificateurs fondés sur la propagation de contrainte.

La première famille a une démarche similaire à celle de NOAH, la seule différence est qu'à chaque prédicat intervenant dans la description de l'action, lui est associé un intervalle de validité temporelle. Les dépendances entre les propositions sont traduites dans un formalisme temporel, par exemple deux intervalles associés au même prédicat sont égaux ou disjoints.

Pour la deuxième famille, c'est la tâche proprement dite à laquelle est associé un attribut temporel (instant ou intervalle). Le formalisme exact dépend du modèle de représentation du temps choisi. Quant à la fonction de propagation de contraintes, il s'agit pour elle d'assurer la cohérence du plan au fur et à mesure de sa construction. Cela inclut évidemment la cohérence temporelle. Les contraintes temporelles peuvent correspondre aux différents moments auxquels on commence une tâche, alors que les contraintes non temporelles peuvent correspondre à un ensemble d'objets auxquels on veut faire appliquer une tâche donnée.

b - Le formalisme de représentation du temps

L'objet temporel commun associé à tous les systèmes est l'intervalle, associés suivant le cas à un prédicat ou à une action. A partir de là, il existe plusieurs façons de représenter ces intervalles. Deux critères peuvent définir cette représentation [Ghallab et al, 89] :
- une représentation symbolique,
- une représentation numérique.

b1 - Les systèmes numériques
Ces systèmes utilisent des nombres dans la description des intervalles temporelles. La représentation temporelle peut utiliser d'autres symboles, qui dénotent en particulier une relation entre deux intervalles. Le maintien de la cohérence temporelle passe par la modification des nombres.

b2 - Les systèmes symboliques
Ces systèmes manipulent des symboles qui peuvent être des intervalles ou des instants. Le maintien de la cohérence temporelle est assuré en modifiant les relations symboliques entre intervalles ou points. Le modèle symbolique de représentation du temps par intervalle de Allen [Allen, 83], [Allen, 84] rentre dans le cadre d'un formalisme de graphe temporel.
En effet, les noeuds sont des symboles qui font référence à des intervalles, et les arcs des relations temporelles disjonctives (avant , pendant ou après). Il existe en fait treize primitives reliant deux intervalles. Ainsi les déductions aboutissent au recalcul de certaines disjonctions des relations, ce qui permet de les préciser. Enfin les relations d'Allen sont mutuellement exclusives et permettent de relier tout couple d'intervalles.

5.6.2 *L'apprentissage du plan*

Un planificateur confronté à la même situation fera deux fois le même travail. Une extension de STRIPS par exemple permet néanmoins au planificateur de stocker les plans déjà formés, mais l'inconvénient est qu'il faut rigoureusement les mêmes conditions pour pouvoir les appliquer ; ce qui est à la fois trop rare et trop long à réaliser. Les travaux associés à ce type de problème porte le nom de couplage résolution/apprentissage, comme nous allons le décrire dans ce qui suit.

Les systèmes d'apprentissage apprennent certes des connaissances destinées, sous une forme ou sous une autre, à résoudre des problèmes. Pourtant la majorité des modules d'apprentissage ne sont pas intégrés dans un système plus général comprenant aussi un résolveur de problème. Ce découplage pourrait présenter l'avantage de rendre l'apprentissage d'une base de connaissance indépendante de l'usage qui en sera fait. En fait les spécificités des

représentations des connaissances obligent de toute façon à «traduire» les résultats de l'apprentissage sous une forme utilisable par le résolveur de problème. Les résultats d'un module d'apprentissage intégré peuvent de la même façon être traduits pour un autre résolveur de problème que celui qui est intégré.

En fait, sont généralement intégrés les modules d'apprentissage qui apprennent des connaissances opérationnelles de résolution de problème sous forme d'opérateurs de planification, de règles de contrôle comme LEX, [Mitchell & al, 83], PRODIGY, [Minton & al, 87], [Hilario, 91], de structures de tâche [Thomas & al, 93], ou de macro-opérateurs comme STRIPS ou ABSTRIPS, en opposition avec les systèmes d'apprentissage qui apprennent des définitions de concept du domaine d'application. L'intérêt d'intégrer le module d'apprentissage présente trois avantages majeurs.

D'une part, seules les connaissances nécessaires au résolveur sont apprises et sous la forme attendue par celui-ci. D'autre part, les connaissances peuvent suivre un cycle complet au sein d'un même système, elles sont acquises automatiquement par apprentissage et transmises au résolveur qui les valide empiriquement et elles sont éventuellement révisées par le module d'apprentissage ou remplacées par de nouvelles connaissances quand leur utilisation ne donne pas toute satisfaction. Les différentes phases de ce processus, analyse des performances du résolveur, déclenchement d'une phase d'apprentissage et retour au résolveur sont plus ou moins automatisées selon les systèmes. Dans les systèmes apprentis, l'analyse des performances du résolveur est à la charge de «l'enseignant» qui décide *quand* il faut apprendre. Quand les modules sont indépendants, le contrôle de la boucle d'acquisition est complètement à la charge de l'utilisateur, que ce soit la décision de valider empiriquement les connaissances par le résolveur ou la décision de déclencher un nouvel apprentissage sur la base de l'interprétation et de l'évaluation des résultats du résolveur.

Enfin, le dernier argument pour justifier l'intérêt du couplage résolution / apprentissage concerne les systèmes qui acquièrent les exemples d'apprentissage auprès d'un utilisateur, et non pas en analysant leur environnement, comme le font par exemple certains robots. Les nouvelles étapes de résolution enseignées par l'utilisateur au système peuvent être considérés comme des exemples d'apprentissage.

Dans un système d'apprentissage isolé du résolveur, les exemples d'apprentissage sont fournis par l'utilisateur en dehors de tout contexte de résolution. Ce choix peut se justifier par les connaissances descriptives du domaine dans la mesure où elles doivent être relativement indépendantes de leur utilisation. Mais en ce qui concerne les connaissances opérationnelles l'acquisition en situation de problème est évidemment plus facile et plus pertinente. L'analyse de protocoles en particulier a prouvé son efficacité comme moyen de recueillir des connaissances de résolution par rapport à des méthodes d'interview classique : observer un expert résolvant un problème réel et analyser ensuite les étapes de résolution avec son aide permettant d'acquérir des connaissances d'expertise que l'expert n'aurait pas pu expliciter dans une interview hors du contexte de résolution.

Dans les systèmes apprentis, la situation est un peu différente de celle de l'analyse de protocole puisque l'utilisateur observe le résolveur en situation de résolution et intervient quand il observe ce qu'il considère comme une mauvaise réponse du système à un problème. Cela suppose que l'utilisateur comprenne le raisonnement suivi par le système et intervienne ponctuellement quand une erreur est commise.

Ce type de méthode n'est clairement pas adapté à l'acquisition de méthodes complexes de résolution, mais a prouvé son efficacité sur des problèmes décomposables en sous-problèmes de proche en proche dans les systèmes LEAP, [Mitchell et al, 89], DISCIPLE, [Tecuci & Kodratoff, 89], CAP, [Jourdan et al, 91], [Dent et al., 92] et APT, [Ferreira et al, 93]. APT et DISCIPLE ont également montré comment l'acquisition de connaissances en situation de résolution de problème pouvait ne pas se limiter à l'acquisition de l'exemple de résolution, mais être également étendue à différentes phases de l'apprentissage.

5.6.3 *Planifier la planification ou métaplanification*

On peut avoir à raisonner sur l'activité même de la planification, c'est le problème de la métaplanification. Il s'agit de manipuler explicitement la structure du plan, malgré quelques essais notamment avec le concept de BLACKBOARD, il n'en reste pas moins qu'aucun système ne manipule vraiment le plan comme un objet formel.

5.6.4 *Réactivité de la planification*

Il est fatal d'arriver à un moment ou un autre à élaborer une prévision de ce que sera l'état de l'univers au moment de l'application de chaque action, et de la façon dont elle se déroulera. Or cette prévision peut être fausse ou peu précise ce qui peut nécessiter ce qu'on appelle une replanification. Ce type de problème a donné lieu à des systèmes dits de planification réactive.

5.6.5 *Interactivité de la planification*

En général, un planificateur résout un problème de manière complètement autonome, il est donc utile, voire indispensable, de pouvoir contrôler la progression du système lors du processus de résolution [Bellalem, 91], [Bellalem, 97]. Ceci est encore plus vrai dans un système interactif, comme nous le verrons ultérieurement. Il existe aussi des systèmes d'apprentissage interactifs qui ne jouent pas le rôle de l'expert du domaine, mais celui d'un cogniticien intermédiaire entre le système et l'expert. Même si l'apprentissage simplifie la tâche de modélisation des connaissances, elle ne la supprime pas, et il semble irréaliste d'exiger d'un expert du domaine d'application qu'il sache représenter à la fois des exemples de concepts, une théorie du domaine, des paramètres éventuels de l'univers externe, tout cela sans posséder un minimum de compétences en apprentissage.

Pour terminer cette étude sur la planification, il nous reste à présenter un exemple de système d'aide à la planification.

5.7 Exemple de système d'aide à la planification

Planex [Michard, 87] est un outil destiné à faciliter l'utilisation des systèmes de commandes développé à l'INRIA par A. Michard. Les objectifs consistent à aider l'utilisateur à planifier ses actions, et cela consiste à lui suggérer une suite de transformations convergeant vers l'objectif recherché, avec un niveau de détail dépendant du type de tâche et du savoir disponible chez cet utilisateur, mais sans pour autant être un outil d'apprentissage pour débutant.

L'élément central est un générateur de plans indépendant du domaine qui utilise une base de connaissances relative à un domaine spécifique. La succession d'actions produites permet de passer d'un état courant à un autre état souhaité. L'interaction se déroule en langage naturel grâce à un analyseur, ESOPE, qui permet de traduire toute phrase en français formée à partir d'un vocabulaire fixé en une représentation interne arbitraire, compatible avec l'application interfacée : des explications sont générées par chaînage arrière à partir de l'objectif ainsi déterminé et en utilisant les prérequis de chaque commande.

Trois types de connaissances sont utilisés :

- La description fonctionnelle du système (ensemble des fonctions élémentaires qu'il peut réaliser) ;
- La description du ou des plans(s) permettant de réaliser la tâche ;
- Une représentation de l'utilisateur c'est-à-dire des indications sur l'ensemble des connaissances dont ce dernier dispose à un instant donné sur le fonctionnement du système.

Les commandes sont regroupées en opérateurs. Une suite d'opérateurs constitue une tâche et les explications produites par le système. Le plan d'action généré, se présentent sous la forme d'une structure hiérarchique assez proche de la représentation cognitive que l'utilisateur peut avoir de sa tâche. Cette tâche est une suite non structurée de commandes.
Ces plans se présentent sous la forme :

tâche T: opérateur T1
commande a
commande b
opérateur T2
commande c
opérateur T3 ....
tâche M: opérateur M1
commande a
commande b
commande c
opérateur M2
.....

La base de connaissances du système est formée exclusivement de règles de production qui peuvent décrire des états (comme conjonctions de faits), les effets des commandes en terme de changement d'états de décomposition d'opérateurs, ou des conditions de cohérence (possibilité ou non par exemple de considérer un opérateur dans certains états) :

Opérateur :
- **Si** (état I) **et** (opérateur T1) **alors** (état J)
Décomposition d'opérateurs :
- **Si** (état k) **et** (opérateur Ta) **et** (opérateur Tb)
alors (opérateur T1)
- **Si** (état K) **et** (commande X) **alors** (opérateur T2)
Etats:
- **Si** (fait A) et (fait B) ...**alors** (état I)

La vérification de la valeur d'un fait (vrai/faux) se fait par des fonctions accédant à une structure de données servant d'interface entre le système cible et PLANEX et reflétant à tout instant l'état réel du système cible.

Pour la réalisation des buts fixés, il a été prévu :

1 - un module de raisonnement,
2 - une base de connaissances, avec des structures plus au moins différentes, selon le système d'aide et son domaine d'application,
3 - un interpréteur de requêtes.

L'accent étant mis sur l'utilisation de techniques de génération de plans, un tel système sera d'autant mieux adapté que l'aide dont l'utilisateur débutant a besoin, porte sur la planification d'actions. Tel est le cas des logiciels de C.A.O, ou de manipulation de documents à condition de s'adresser à des personnes connaissant les principaux concepts.
Ces conditions ne sont pas vérifiées si les systèmes cibles sont des systèmes d'exploitation, car ces outils présentent des commandes assez complexes (arguments, options..) et posent souvent des problèmes conceptuels quant aux fonctions réalisées.

Nous verrons par la suite que l'on peut résoudre ce genre de problèmes grâce à des techniques appropriées que nous montrerons.

6 Conclusion

Dans ce premier chapitre nous avons essayé de dresser l'état de l'art (non exhaustif) de la recherche dans le domaine particulier des systèmes experts qui sont dédiés en particulier à la C.A.O en décrivant les principes de base, les nouvelles tendances, ainsi que certains systèmes opérationnels.

Le but de la plupart de ces modèles (en particulier ceux dédiés à la C.A.O) est de proposer à l'utilisateur un système d'aide à la conception plus souple que les systèmes classiques, capable de le guider dans les tâches à accomplir sans pour autant le contraindre dans sa démarche de conception.

Cependant, l'utilisateur rencontre des problèmes importants lorsqu'il s'agit de proposer des alternatives de conception variées, en particulier pour des utilisateurs ayant des profils différents. Pour cela, l'étude des techniques liées au comportement de l'utilisateur face à une tâche précise à accomplir peut contribuer à trouver les moyens de répondre à leurs attentes.
Avec l'étude consacrée au modèle du processeur humain, nous avons essayé de dégager les principes de base pour intégrer ces mécanismes dans notre système.

Enfin, l'un des prolongements naturels consiste à trouver un outil qui nous permette de pallier le problème de conception variée, pour cela l'étude de la planification générale et de la planification interactive nous a semblé appropriée.
En combinant des mécanismes liés aux systèmes experts, à la planification interactive et en prenant en compte les aspects du modèle du processeur humain, nous proposons par la suite un modèle qui présente les caractéristiques suivantes :

1 - Donner des alternatives variées pendant la conception en fonction du profil de l'utilisateur (expert, non expert) et ne pas imposer une alternative précise sans passer par une négociation rigoureuse.
2 - Offrir un environnement qui gère les différentes informations produites au fur et à mesure de la conception.
3 - Mettre en œuvre des mécanismes de raisonnement dits "intelligent" (systèmes d'inférence).
4 - Offrir des mécanismes d'apprentissage, pour permettre d'une part l'évolution globale du système et d'autre part l'initiation de l'utilisateur non expert aux difficultés de la conception.

Chapitre 2

Problématique de l'interactivité Homme-Machine appliquée à la C.A.O

1 Introduction

Après avoir défini dans le chapitre 1 les concepts des différents outils utilisés par le modèle que nous proposons : système expert, planification et outils d'interaction Homme-Machine. Il s'agit dans ce deuxième chapitre d'une part, de préciser le domaine de travail dans lequel évolue le système que nous proposons c'est-à-dire la C.F.A.O à travers les concepts qui y sont définis et d'autre part, de compléter l'étude de l'interaction Homme-Machine.

Ce chapitre comprend trois parties distinctes : la première partie donne une définition précise du domaine de la C.F.A.O constitué d'une partie modèle et d'une partie dialogue Homme-Machine. La deuxième partie donne un aperçu de ce qu'est un logiciel interactif à travers les différents modèles qui le constituent. La troisième partie décrit quelques exemples parmi les plus utilisés des modèles d'architecture pour les applications graphiques interactives.

2 Le domaine de la C.F.A.O

2.1 Définitions

La C.F.A.O (conception et fabrication assistée par ordinateur) est la combinaison de deux disciplines : la C.A.O (conception assistée par ordinateur) et la F.A.O (fabrication assistée par ordinateur).

- La C.A.O est l'ensemble des aides informatiques aux bureaux d'études et de méthodes, depuis l'élaboration du cahier de charges jusqu'à l'établissement des documents nécessaires à la fabrication [Gardan, 91]
- La F.A.O représente l'utilisation de l'informatique pour planifier, gérer et contrôler les opérations de fabrication.

Deux composantes essentielles constituent la plupart des systèmes de C.F.A.O : le modèle et le dialogue Homme-Machine.

- Le modèle est défini comme la représentation informatique de l'objet en cours de conception ou de fabrication.
- Le dialogue Homme-Machine contribue à rapprocher les utilisateurs du système et ses possibilités. Les différents utilisateurs qui interviennent dans un système de C.F.A.O sont : l'opérateur, le programmeur d'interfaces et le programmeur d'applications.

Nous allons décrire plus précisément chaque composant (modèle et dialogue) d'un système de C.F.A.O en insistant plus longuement sur l'aspect dialogue qui nous intéresse plus particulièrement. Le but est de pouvoir d'une part, de mieux cerner notre étude et d'autre part, d'avoir une vue d'ensemble d'un système de C.F.A.O.

2.2 Le modèle

Le modèle est défini à partir d'une multitude de modèles dits applicatifs ayant des caractéristiques propres. Les différents types de modèles sont :

1 - Le modèle géométrique : (qui est le coeur des systèmes de C.F.A.O) est l'expression employée pour désigner les propriétés géométriques des objets, leurs formes et leurs représentations. Le concepteur construit le modèle géométrique qui représente l'image virtuelle d'un objet. Comme exemple de modèle géométrique on peut citer :

- Le modèle B-Rep (modèle par les limites) : il comporte des informations géométriques et topologiques d'un objet. Les entités topologiques (faces, contours, arêtes, sommets) s'appuient sur les entités géométriques (surfaces, courbes, points) pour décrire un objet.
- Le modèle CSG (modèle par arbre de construction) : il comporte un arbre qui permet de conserver l'historique de construction, c'est à dire les différentes opérations (en général booléennes) mises en œuvre pour réaliser un objet. La structure de ce modèle est un arbre binaire dont les feuilles représentent des objets primitifs (parallélépipède, cylindre, sphère, etc...) ou des demi-espaces et les nœuds des opérations booléennes ou des transformations.

2 - Le modèle de visualisation : qui définit des données propres à la représentation visuelle de l'objet.

3 - Le modèle commande numérique : son rôle consiste à définir les données propres à la fabrication de l'objet.

L'ensemble de ces modèles applicatifs sont extraits à partir d'informations tirées d'un ensemble plus global dit modèle générique. Le passage du modèle générique vers un modèle applicatif est une localisation, le passage inverse est une globalisation.

2.3 Le dialogue Homme-Machine

La mise en place d'un dialogue Homme-Machine est nécessaire dans un système de C.F.A.O en raison de la grande complexité du processus de conception. De la même manière le dialogue est complexe en raison de problèmes dus entre autres aux éléments suivants [Gardan, 91] :

- Les entrées de l'opérateur ne sont plus seulement numériques ou alphanumériques, mais aussi grapho-numériques.
- Les dialogues sont nombreux et font intervenir un très grand nombre d'objets plus au moins complexes.
- Les contraintes liées à la construction entre les objets impliquent de prendre en compte la moindre modification qui intervient dans le dialogue.

Pour construire notre modèle, nous avons besoin de prévoir une structure modulaire de gestion du dialogue qui soit adaptable (nous expliciterons ce terme plus longuement dans le chapitre 4) à des types d'utilisateurs différents (expert, non expert). De plus cette interface doit pouvoir prendre en compte les mécanismes de la planification interactive (qui est l'outil de base de notre système) en étant indépendant de l'application en cours.

Pour ces raisons, nous ne pouvons faire l'économie d'une étude (même non exhaustive) qui nous permet d'avoir une vue générale des modèles d'interaction Homme-Machine à travers les modèles d'architecture pour les applications graphiques. Mais avant d'entreprendre cette étude, nous allons présenter quelques généralités sur la communication Homme-Machine.

3 Généralités sur la communication Homme-Machine

3.1 Introduction

Le système que l'on désire développer est un système où l'un des deux intervenants est humain et l'autre, une machine, de ce fait on parle de communication Homme-Machine.
Le modèle de la communication Homme-Machine qui a jeté les bases des études dans ce domaine est celui de Shanon [Horman, 72] qui se présente comme un processus de transmission d'une source (S) vers un récepteur (R) via un canal. La source et le récepteur pouvant accéder à un répertoire commun comme indiqué sur la figure 2.1. Le message est encodé dans un premier temps en passant par le canal, décodé dans un deuxième temps par le récepteur.

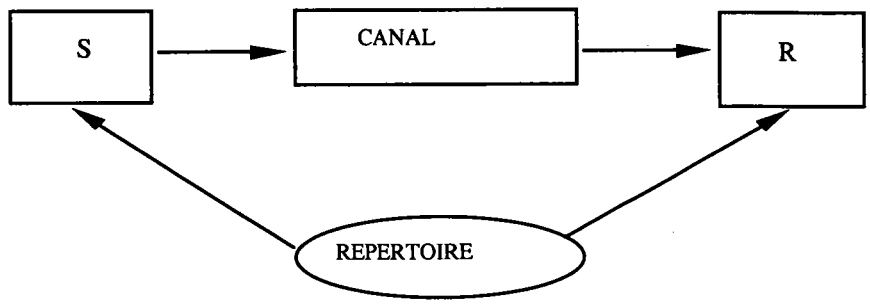


Figure 2.1 Modèle de la communication selon Shanon

3.2 Les différentes approches

Les nombreux travaux qui de près ou de loin appartiennent à la communication Homme-Machine se situent dans deux grandes catégories :(1) les travaux liés au génie logiciel, (2) les travaux liés à l'intelligence artificielle.

L'approche liée au génie logiciel est utilisée pour l'amélioration des interfaces en minimisant les contraintes d'utilisation d'un système informatique, l'évaluation des performances pour de tels systèmes est liée à la faculté qu'aura un usager à s'y adapter rapidement.
Malgré le succès qu'ont connu un certain nombre de systèmes, l'inconvénient majeur de cette approche reste pour une grande part un mode de communication artificielle, et un effort d'adaptation qui peut paraître parfois difficile à supporter.

L'approche liée à l'intelligence artificielle a pour objectif ambitieux de doter la machine de certaines facultés de communication possédées par l'être humain. Elle se retrouve aussi dans un ensemble de domaine d'application, dont on peut citer par exemple la démonstration automatique de théorème, le traitement automatique du langage naturel écrit, l'interprétation

d'image et la vision par ordinateur, ou bien encore, le traitement automatique de la parole. Nous allons dans la suite passer en revue certaines fonctionnalités du dialogue oral.

Le centre de recherche de Xerox à Palo Alto au Etats-Unis peut être considéré comme l'initiateur de la révolution informatique qui a marqué la communication Homme-Machine. On lui attribue en effet, l'apparition de la souris en 1964 et la naissance des notions de fenêtre et d'interaction graphique que l'on trouve dans Smaltalk. Steve Jobs avec Lisa et le Macintosh (toujours à Palo Alto) ont permis l'éclosion des notions d'icônes qui ont été le moteur permettant le développement des interfaces graphiques.

La recherche sur les modèles d'architecture pour les interfaces Homme-Machine a débuté sur ces bases. Elle a été lancée par la réunion de travail à Seattle en 1982. Cette réunion a principalement porté sur les techniques d'interaction graphique et a clairement établi la nécessité de séparer l'interface utilisateur de l'application (comme nous le verrons plus en détail plus loin). La réunion de Seeheim en Allemagne en 1983 a jeté les bases d'un des modèles les plus largement reconnus en informatique, concernant l'architecture des applications interactives.

Dans la suite de ce chapitre nous allons détailler les principes sur lesquels repose l'architecture des interfaces Homme-Machine et les principaux modèles qui en découlent.

4 Les logiciels interactifs

4.1 Introduction

Le développement de logiciels interactifs est depuis longtemps régi par le principe de séparation entre l'interface de l'application et son noyau fonctionnel :

Application interactive = interface + noyau fonctionnel.

Le noyau fonctionnel contient tous les traitements réalisés par l'application et qui ne sont pas liés à l'interaction. Par exemple, la simulation d'un processus, l'accès à une base de données font partie du noyau fonctionnel d'une application interactive. L'interface contient uniquement les modules nécessaires à la présentation des données et à l'entrée des commandes. De façon idéale, on doit pouvoir modifier ou bien changer l'interface sans toucher au noyau fonctionnel et inversement.

4.2 Principe de séparation

Le principe de séparation est souvent annoncé, mais plus rarement mis en œuvre pour les raisons suivantes :

1 - Le noyau fonctionnel est souvent conçu et spécifié sans souci de l'interface, et, lorsqu'on veut intégrer l'interface, le noyau fonctionnel ne se prête plus à une séparation propre.

2 - Certaines applications sont conçues sans avoir réellement identifié l'interface et le noyau fonctionnel. C'est particulièrement vrai pour les éditeurs (de texte, de dessin ou autres) pour lesquels il y a identité entre les objets manipulés par l'utilisateur et ceux du noyau fonctionnel.

3 - Les outils utilisés pour construire l'interface ne favorisent pas la séparation. Par exemple la plupart des systèmes imposent un modèle de programmation dirigé par les événements qui, si l'on n'y prend pas garde, conduit à un mélange des fonctions du noyau.

Pour rendre le principe de la séparation effectif, il est indispensable de concevoir l'application et le noyau fonctionnel en même temps. On peut également se donner des contraintes techniques qui forcent cette séparation. Par exemple, on peut utiliser une architecture distribuée dans laquelle deux processus communicants gèrent l'un, l'interface et l'autre, le noyau fonctionnel.

Cette approche offre d'ailleurs d'autres avantages comme la possibilité de parallélisme entre l'interface et le noyau fonctionnel et l'évolution vers un modèle client-serveur.
Les logiciels disponibles pour la construction d'applications interactives se situent généralement à un faible niveau d'abstraction. On trouve des bibliothèques graphiques qui permettent de gérer l'affichage en deux ou trois dimensions dans des fenêtres, des boîtes à outils et des générateurs d'interfaces. Parmi les bibliothèques graphiques, une standardisation de fait a eu lieu par la généralisation du système de fenêtrage X Windows Systems [Sheifler, 86] du MIT pour les stations de travail.

Pour ce qui concerne les boîtes à outils et les générateurs d'interfaces, ils font l'objet de nombreux travaux. Comme par exemple les travaux sur les UIMS (User Interface Management Systems) [Pfaff, 85], ainsi un UIMS est une collection d'outils qui permet de concevoir, d'implanter, de tester et de maintenir des applications interactives. En d'autres termes, il s'agit d'un atelier de génie logiciel appliqué aux interfaces Homme-Machine.

4.3 Le processus de conception

4.3.1 Introduction

La réalisation de logiciels interactifs est sujette à une forte inertie due, entre autres, à deux facteurs :

1 - Facteurs techniques : par exemple l'inexistence de périphériques adéquats ou encore de logiciels de base.

2 - Facteurs humains : un constructeur change rarement le style d'interaction entre plusieurs versions d'un même logiciel, ceci est dû au fait qu'un utilisateur préfère utiliser un système dépassé dont il a l'habitude, plutôt qu'un nouveau logiciel qu'il doit commencer par apprendre.

La plupart du temps, l'utilisateur ne cherche pas à utiliser un logiciel pour le plaisir, mais simplement pour effectuer une tâche donnée le plus efficacement possible, en fonction de ses connaissances et des moyens disponibles.

La notion de tâche de l'utilisateur nous semble être à la base de la conception de toute application interactive. Le processus de conception d'une application interactive devrait

comporter de façon idéale quatre modèles : le modèle cognitif, le modèle conceptuel, le modèle structurel, et le modèle perceptif, comme indiqué sur la figure 2.2 est issus des travaux sur les UIMS [pfaff, 85].

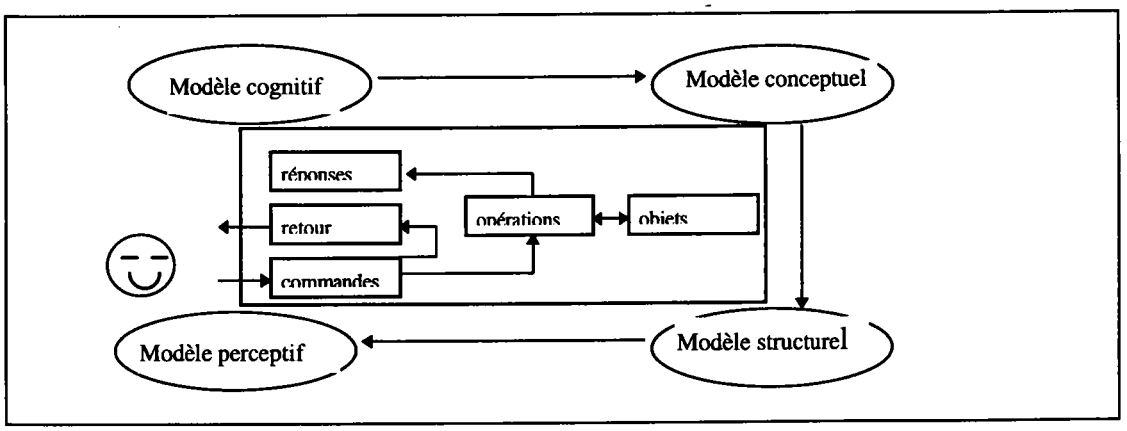


Figure 2.2 Le processus de conception

Avant de présenter les différents modules qui composent ce modèle, nous allons décrire le fonctionnement des fonctions, objets et opérations à travers l'exemple d'un logiciel de dessin comme indiqué sur la figue 2.3.

Fonctions	Objets	Opérations
Gestion du document	Fichier	Sauvegarde, chargement Impression
Gestion de la présentation	Fenêtre de visualisation	Ajustement de la fenêtre Vue sur le dessin
Edition globale	Formes géométriques	Agencement en profondeur Mise en page : couper, copier, coller, déplacer, transformer, regrouper, ... Attributs : couleur, motifs, styles, ...
Edition locale	Poignées (objets simples) Sommets (polygones)	Déformation Changement de taille Déplacement, suppression et addition de sommets

Figure 2.3 Modèle conceptuel d'un logiciel de dessin

4.3.2 Le modèle cognitif

Cela consiste en l'analyse des tâches de l'utilisateur et doit prendre en compte les facteurs humains généraux ainsi que ceux propres à l'application envisagée. Un logiciel de dessin qui s'adresse à des graphistes n'aura pas la même interface que celui s'adressant au grand public. Il s'agit donc de caractériser les habitudes de travail, les besoins des utilisateurs et les contraintes techniques.
Cette étape de la conception doit faire intervenir des cogniticiens, des ergonomes, des informaticiens et les utilisateurs potentiels. Comme la phase d'analyse des besoins dans la conception d'un logiciel classique, cette phase est déterminante pour la réussite de l'application.

4.3.3 Le modèle conceptuel

C'est la phase suivante qui consiste à rationaliser et à organiser les résultats du modèle cognitif pour décrire la façon dont un utilisateur du système final pourrait se représenter son fonctionnement. Le modèle générique d'interaction sous-jacent aux modèles conceptuels est présenté au centre de la figure 2.2. L'utilisateur entre des commandes qui donnent lieu à un retour immédiat. Ces commandes sont ensuite traduites en opérations sur des objets connus de l'utilisateur et le résultat de ces commandes est rendu visible à l'utilisateur.

Pour décrire le modèle conceptuel, on décompose l'application en plusieurs fonctions principales d'interaction. Par exemple, un éditeur de dessin se décomposera en fonction de la gestion de document entier (sauvegarde, chargement, etc.) en fonction de la présentation (création de fenêtres sur le document, défilement, zoom, etc.) et en fonction de l'édition proprement dite. On identifie ensuite les objets, leurs propriétés et les opérations qui leurs sont associées.

4.3.4 Le modèle structurel

Il correspond à l'implémentation de l'interface proprement dite. Il s'agit de déterminer comment mettre en œuvre le modèle conceptuel à partir des boîtes à outils et des autres outils existants en respectant les guides et styles qui leur sont associés. Des modèles tels que celui de Seeheim [Pfaff, 86] (voir figure 2.4) ou le modèle PAC [Coutaz, 90] aident également à structurer l'interface. On verra une étude plus détaillée de chacun de ces modèles dans la partie consacrée aux exemples de modèles de logiciels interactifs. Souvent, un certain nombre de compromis doivent être faits à ce stade, car les outils ou les styles d'interaction utilisés imposent des contraintes techniques importantes.

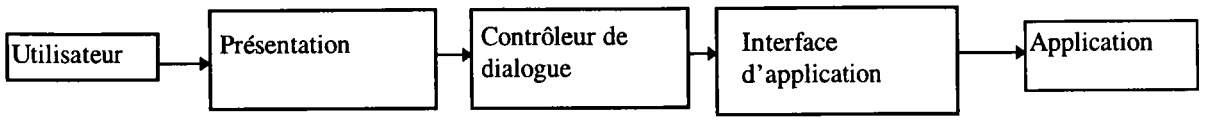


Figure 2.4 Une représentation du modèle de Seeheim

4.3.5 Le modèle perceptif

Il n'est pas construit dans le même sens que les modèles précédents, il représente la façon dont l'utilisateur perçoit le système final. Dans le cas idéal, le modèle perceptif correspond au modèle conceptuel et dans ce cas, l'utilisateur peut anticiper le fonctionnement du système et donc l'utiliser de manière plus efficace, tout en diminuant les erreurs. Si les deux modèles (perceptif et conceptuel) ne correspondent pas, l'utilisateur ne comprend pas les réactions du système et ne peut accomplir son travail de façon satisfaisante.

Le modèle perceptif n'est pas une propriété intrinsèque du système puisqu'il dépend des habitudes et des méthodes de travail de l'utilisateur. La principale manière d'évaluer le modèle perceptif est de réaliser des expérimentations auprès des utilisateurs, en leur donnant une tâche à accomplir. A l'issue de l'expérimentation, un questionnaire permet de recueillir les impressions des utilisateurs.
Le processus de conception est relativement indépendant d'une méthode de conception ou d'un modèle particulier de cycle de vie. On constate en général que l'utilisation de prototypes et l'intégration des utilisateurs au sein du processus de conception améliorent la qualité des applications finales. C'est le cas par exemple de la version 7 du système d'exploitation du Macintosh. Les tests ont été très longs, aussi bien auprès des utilisateurs avertis du Macintosh, qu'auprès des utilisateurs ne le connaissant pas, afin d'assurer une bonne adéquation entre modèle cognitif et modèle perceptif pour le plus grand nombre d'utilisateurs.

4.4 Les modèles d'interaction

4.4.1 Introduction

Même si chaque application a son propre modèle conceptuel, un certain nombre de styles d'interaction sont devenus suffisamment généraux pour s'adapter à de nombreuses applications. Nous allons décrire ces modèles d'interaction.

4.4.2 Le modèle conversationnel

Historiquement, le modèle d'interaction dit conversationnel, a été le premier utilisé et reste encore très répandu. Dans le modèle conversationnel, le système indique à l'utilisateur qu'il est prêt à saisir une commande. Lorsque celle-ci est entrée, le système l'exécute et fournit à l'utilisateur le résultat de la commande avant d'accepter la suivante. Pendant l'exécution de la commande, le système peut demander à l'utilisateur des informations complémentaires, toujours sur un mode questions/réponses.

La puissance des systèmes utilisants le mode conversationnel provient de la puissance du langage d'entrée. Ainsi sous UNIX par exemple l'interpréteur de commande "Shell" utilise le mode conversationnel. Le langage d'entrée est à la fois précis, concis et puissant, ce qui permet d'entrer des commandes complexes facilement.

L'inconvénient majeur des systèmes conversationnels réside dans le fait que l'utilisateur est au service de la machine et non l'inverse. En effet, c'est le système qui demande une information lorsqu'il en a besoin.

Si l'utilisateur n'a pas les moyens de fournir cette information, il est obligé d'annuler la commande pour éventuellement demander des informations à l'aide d'autres commandes.

4.4.3 Les modèles fondés sur la manipulation directe

Le terme de manipulation directe a été introduit par Shneiderman [Shneiderman, 83]. L'objectif de la manipulation directe est de présenter constamment les objets de l'interaction et d'utiliser des actions physiques sur ces objets, généralement à l'aide de la souris, pour effectuer des opérations. L'effet de ces opérations doit être visible sur les objets eux-mêmes et, si possible, facilement réversible. Un système à base de manipulation directe bien conçu permet à l'utilisateur d'apprendre le système en l'utilisant.

Par rapport au modèle conversationnel, la manipulation directe apporte deux avantages décisifs : d'une part, l'utilisateur contrôle le système, et non l'inverse, car il voit des objets sur lesquels il peut agir à tout moment sans avoir à attendre une sollicitation de la part du système, d'autre part, la manipulation directe permet d'interrompre une tâche pour en entamer une autre, de mener des tâches en parallèle, ou d'effectuer une sous-tâche en cours. Les systèmes à manipulation directe dépendent d'un certain nombre de paradigmes, c'est le cas entre autres du modèle de fenêtrage.

Nous passons à la tâche d'interaction entre l'utilisateur et les services du système. Cette tâche relève de la gestion du dialogue.

4.5 La gestion du dialogue

4.5.1 Introduction

Les points clés de la modélisation logicielle du dialogue se situent sur plusieurs niveaux :
- le niveau échange entre l'application et l'interface,
- le niveau localisation du contrôle de l'interaction au sein du système,
- le niveau modélisation du fonctionnement de ce contrôle.

4.5.2 Le niveau application et interface

Définissons d'abord les notions d'application et d'interface qui interviennent dans un système de dialogue :
a - Une application regroupe pour un domaine donné l'ensemble des concepts qui correspondent aux variables psychologiques de l'utilisateur.

b - L'interface assure la communication d'informations entre l'application et l'utilisateur.

Une fois le langage d'entrée fixé, en s'appuyant sur un système d'interaction lié aux notions précédentes, il reste à définir le choix de d'abstraction des échanges entre l'application et l'interface. Ainsi, lorsque le niveau est bas (par exemple en événement de type "clic souris") le traitement correspondant est noyé dans le code de l'application. En revanche, lorsque les unités d'échange sont de mêmes niveaux que celles de l'application alors l'application est indépendante des techniques de présentation.

4.5.3 Le niveau localisation du contrôle

Si l'on considère que le système est organisé en deux composants, le choix de la localisation de contrôle se situe entre l'application et l'interface. A partir de cela, on distingue trois formes de contrôle : interne, externe et mixte. Le contrôle est interne lorsqu'il réside dans l'application. Il est externe lorsqu'il est maintenu par l'interface. Il est mixte lorsqu'il est géré en alternance par l'application et l'interface.

4.5.4 La modélisation du contrôle du dialogue

Le contrôle du dialogue peut se modéliser selon deux techniques essentielles : les automates, et le traitant des événements.

La technique des automates présente l'avantage d'être simple à maîtriser. En revanche, son application doit tenir compte du caractère imprévisible du comportement de l'utilisateur. A cet effet le système doit être capable dans chaque état de répondre avec précision à toutes les actions de l'utilisateur.
Le traitant des événements organise le système en une multitude d'agents coopérants prêts à réagir aux événements pertinents, ainsi, comme pour un dialogue entre individus, l'utilisateur peut changer d'interlocuteur ou s'adresser à plusieurs partenaires simultanément. Le modèle multi-agents reflète cette conception de l'interface.

On assiste à une sorte de standardisation de la forme des interfaces et des fonctionnalités qu'elles offrent. Pourtant, certaines fonctionnalités résistent à cette généralisation, par les problèmes difficiles qu'elles posent. Ainsi, la fonctionnalité d'annulation ("UNDO") est bien souvent primitive, permettant seulement d'annuler la dernière commande. Or, nous avons vu qu'un des principes de la manipulation directe est d'offrir des actions réversibles pour encourager l'exploration du système.

La possibilité d'annulation coûte souvent chère, lorsque le noyau fonctionnel de l'application ne la rend pas impossible à mettre en œuvre. Dans le même ordre d'idée, les fonctions d'aide en ligne se bornent à donner un accès à un index de commandes et ne permettent pas de répondre aux questions que se posent l'utilisateur qui y fait appel : où suis-je ? comment suis-je arrivé là ? que puis-je faire maintenant ? En d'autre termes, l'aide est rarement contextuelle et encore plus rarement rapportée à la tâche de l'utilisateur.

4.6 La prise en compte des facteurs humains

4.6.1 Motivations

Ayant pris conscience des potentialités autant que des limites des machines, la plupart des concepteurs d'interfaces reconnaissent le rôle clé de l'homme dans la boucle de décision. L'opérateur humain est pris en compte avec son intelligence, son expertise, ses contraintes physiologiques ou cognitives.

A partir de ce constat, les principes des interfaces Homme-Machine s'orientent donc de plus en plus vers la notion de dialogue coopératif et intelligent où la machine ne se substitue pas à l'utilisateur mais doit faciliter son travail, en particulier dans les tâches difficiles comme les tâches de décision.

La spécification des interfaces Homme-Machine demande une démarche propre. Par exemple, l'aide en ligne, comme l'annulation, nécessite de la part du système une représentation précise de l'utilisateur, un domaine que l'on maîtrise mal. Pour cette même raison les interfaces sont parfois adaptables, mais rarement adaptatives.

Par adaptable, on entend que l'utilisateur peut configurer l'interface selon ses goûts. Par exemple, il peut choisir les commandes disponibles dans les menus, les valeurs par défaut des différents paramètres, etc. Adaptatif signifie que le système lui même détermine l'expérience de l'utilisateur et s'adapte à lui. Par exemple, les fonctions disponibles dans les menus évoluent au fur et à mesure de l'utilisation du système. Voir par exemple les travaux de B.Martin [Martin, 95].

4.6.2 L'activité de l'opérateur

L'activité d'un opérateur humain en situation de travail (comme cela a déjà été définis dans la théorie de l'action de Norman [Norman, 86] est organisée en différentes phases :

- appréhension de la situation,
- identification des buts,
- préparation d'un plan d'action,
- mise en œuvre de ce plan,
- suivi et contrôle de la réalisation,
- correction.

Dans une situation difficile, plusieurs cycles sont menés en parallèle, avec des déphasages possibles et des priorités variables, ce qui nécessite un contrôle global de toutes les activités engagées. Selon ce découpage, on peut organiser les éléments à prendre en compte de la façon suivante :

- Hiérarchiser selon leur niveau critique sur chaque phase. Ce classement est utilisé pour établir des règles concernant les sources, les supports et les formats des informations ainsi que leurs modes d'acquisition et de manipulation. En effet, l'adéquation entre les besoins en informations et les moyens d'y accéder contribue à une appréhension rapide des informations nécessaires à la tâche. Ces informations sont triées selon les phases de l'activité et des situations, et elles permettent des actions plus efficaces et à un suivi plus facile.
- La typologie des activités et l'organisation en tâches et sous-tâches. Des modèles de représentation des tâches apparaissent comme le modèle MAD [Pierret, 90], mais la mise en évidence des plans d'action chez les opérateurs est difficile, surtout quand cela dépasse les contraintes liées à la logique du système. La stratégie adoptée pour atteindre un but est en partie dans les règles opérationnelles communes, en partie dans l'expertise et le mode de travail de l'opérateur. La prise en compte, l'organisation des tâches de l'opérateur permet de ne plus penser à l'interface en fonction de la seule logique du système informatique, mais de l'adapter à l'idée que l'utilisateur a de ce système.

4.6.3 La réalisation informatique par maquettage

Dans la définition des interfaces d'un système informatique complexe, il est difficile d'atteindre d'emblée des spécifications valides ; la prise en compte des facteurs humains déterminant la qualité d'une interface est encore trop formalisée pour autoriser une évaluation *a priori* : seules des expérimentations avec les futurs utilisateurs peuvent valider ou remettre en cause les choix qui ont été faits dans les spécifications.
L'intérêt de procéder à un maquettage rapide (voir la figure 2.5) sur un environnement simulé est certain ; en particulier pour permettre la mise en place très tôt des évaluations des maquettes produites afin d'améliorer et d'affiner l'interface Homme-Machine.

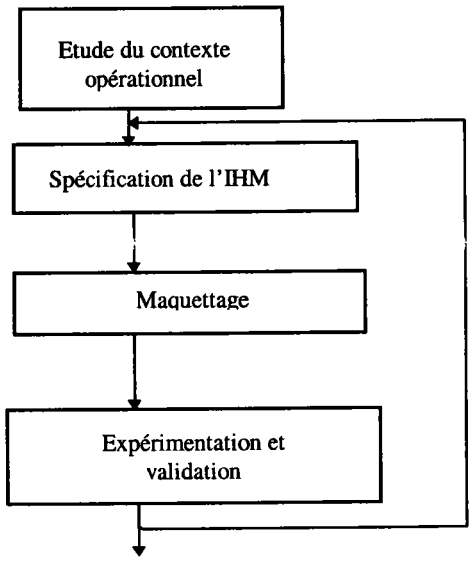


Figure 2.5 Maquettage informatique

4.6.4 Expérimentation et validation opérationnelle

Cette phase consiste à faire travailler l'utilisateur en situation la plus réaliste possible (l'opérateur a une tâche à accomplir), elle nécessite l'établissement d'un plan d'expérimentation qui définit complètement les objectifs, le scénario (durée, situation opérationnelle, tâche à accomplir) et les procédures de recueil des données. Parmi ces données on peut citer :

- L'évaluation subjective par les utilisateurs des moyens de commandes, de présentations de l'information, de l'utilité et de la disponibilité des commandes et des informations, de la charge de travail.
- Les temps d'exécution des différentes tâches du scénario et la performance globale.
- Les taux d'erreur dans l'utilisation des commandes.
- Les modes opératoires adoptés et l'enchaînement des actions selon la tâche et les contraintes temporelles (recueillies lors du suivi de l'expérimentation).

Les étapes précédentes (activité, spécification, réalisation, validation) aboutissent à la confrontation de deux démarches : la démarche conceptuelle qui, à partir de l'analyse de la tâche, déduit les principes de l'interface ; la démarche expérimentale qui produit des conclusions sur la performance globale du système homme-machine et sur la satisfaction de l'opérateur. L'interface optimale se situe à la convergence de ces deux approches.

5 Exemples de modèles de logiciels interactifs

5.1 Introduction

La plupart des applications informatiques disposent d'une interface utilisateur graphique pour permettre la mise en valeur de l'application « créer ». Il était important pour construire ces logiciels de disposer de modèle d'architecture logiciel pour séparer convenablement ce qui est « application » de ce qui est « dialogue » avec l'utilisateur.

On présente dans ce qui va suivre les principaux modèles d'architecture pour les applications graphiques interactives.

5.2 Le modèle Seeheim

Ce modèle doit son nom au congrès sur les systèmes de gestion d'interfaces utilisateur qui a lieu à Seeheim en RFA en 1983 [Pfaff, 85]. Ce modèle se propose d'isoler la gestion du dialogue de la sémantique de l'application. Pour cela le modèle de Seeheim divise l'interface utilisateur d'une application en trois composants logiques : présentation, contrôle du dialogue et interface avec l'application (voir figure 2.6).

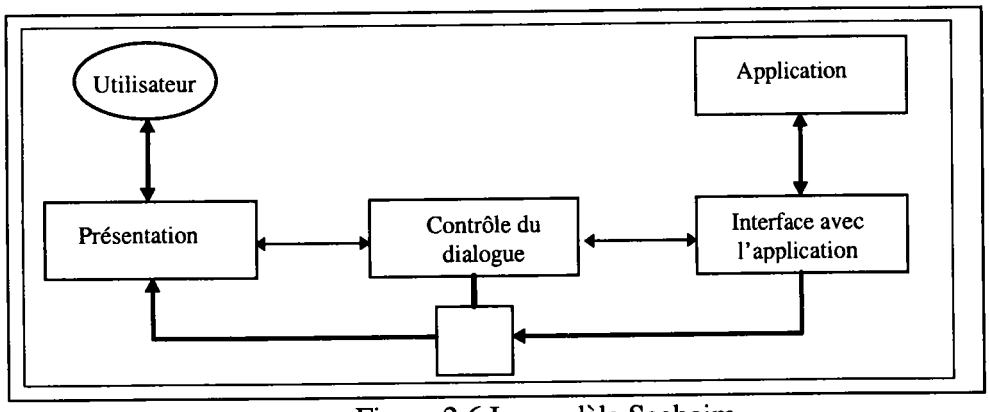


Figure 2.6 Le modèle Seeheim

- Présentation : C'est l'image du système interactif. Il est responsable de la gestion de l'écran, de l'affichage de l'information, des dispositifs d'entrée, des modes d'interaction et de la rétroaction immédiate. C'est à ce niveau que doivent être pris en compte un certain nombre de règles d'ergonomie. Par analogie avec l'analyse d'un programme source, on a pris l'habitude de comparer les différentes couches du modèle de Seeheim avec les phases successives de la théorie de la compilation. La couche de présentation est ainsi assimilée à la couche lexicale des langages de programmation.

- Contrôle du dialogue : c'est l'intermédiaire entre le composant présentation et le composant interface avec l'application. Les entrées issues du composant présentation sont converties en structure représentant les actions de l'utilisateur, qui à leur tour sont converties en une séquence de requêtes envoyées à l'interface avec l'application.

- Interface avec l'application : contient la description de toutes les structures de données de l'application et toutes les procédures accessibles à l'interface utilisateur. La description des procédures de l'application comprend le nom de la procédure, le type, les paramètres ainsi qu'éventuellement les pré et post conditions correspondantes, c'est à dire les conditions requises à l'exécution d'une procédure ainsi que les effets de son exécution. De ce fait, cette couche est assimilée à la phase sémantique de la compilation.

Ce modèle, qui a été l'un des premiers, reste aujourd'hui encore la référence pour les modèles d'architecture logicielle pour les applications interactives. La plupart du temps l'interprétation de ce modèle s'est faite par une décomposition stricte en trois composants sans passer par une décomposition répartie. Une autre confusion consiste à mélanger les composants « application » et « interface avec l'application ».

De nombreuse critiques lui ont été adressées. Certaines résultent du manque de précision dont il fait preuve à certains endroits, les autres proviennent de l'évolution des concepts informatiques, comme par exemple celui de l'orienté-objet, qui ont rendu la structure monolithique du modèle de Seeheim criticable. Ainsi, certaines améliorations ont été proposées, comme le modèle "Seeheim modifié" [Alty, 89] qui ajoute une connexion directe entre la présentation et l'application, ou "Seeheim étendu" [karsenty, 91] qui divise la présentation en trois modules plus spécifiques, correspondant aux abstractions de l'interface, de l'application et au contrôleur de l'application.

5.3 Le modèle PAC

Ce modèle a été proposé par J. Coutaz [Coutaz, 90]. Il est fondé sur l'interprétation répartie du modèle Seeheim, c'est-à-dire que l'application et son interface utilisateur peuvent être décomposés en une hiérarchie d'agents interactifs. Il est formé de trois composants : présentation, abstraction et contrôle (voir figure 2.7).

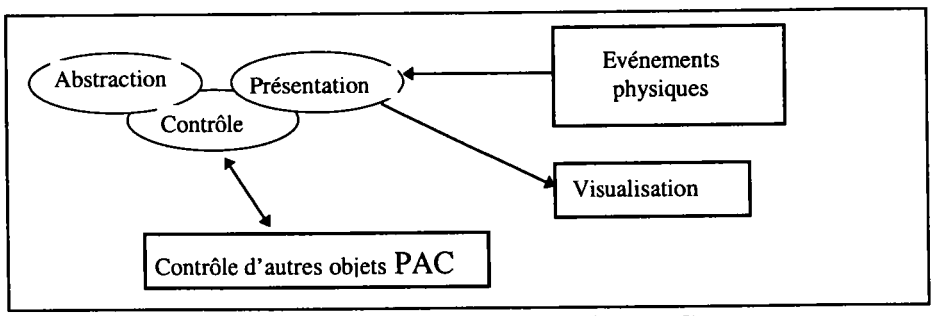


Figure 2.7 Le modèle PAC

- Présentation : définit le comportement de l'application vis à vis de l'utilisateur, c'est-à-dire gère les entrées/sorties de l'objet PAC et facilite donc la réalisation d'écho ou la décentralisation dans le serveur graphique.
- Abstraction : décrit les fonctionnalités de l'application, à la différence de Seeheim, il ne s'agit pas d'une interface vers l'application, mais d'une partie même de l'application.

- Contrôle : il maintient la cohérence entre le composant présentation et le composant abstraction. Ce niveau joue le rôle d'arbitre pour résoudre les conflits, les synchronisations ou les rafraîchissements de représentation. Il gère les relations avec les autres agents PAC.

La différence avec le modèle Seeheim se situe au niveau de l'organisation répartie qui gère la composition et l'interaction d'une multitude d'objets, appelés objets interactifs. Chaque objet est caractérisé par un comportement syntaxique vis-à-vis de l'utilisateur (présentation), des fonctions et des attributs fonctionnels (abstraction), et la gestion des liens entre côtés abstraits et présentation (contrôle).

5.4 Le modèle PAC-Seeheim

Ce modèle a été proposé par J. Coutaz et L. Nigay [Coutaz, 91]. C'est un modèle qui combine le modèle Seeheim et l'approche multi-agent du modèle PAC. Il se décompose en quatre composants : noyau fonctionnel, adaptateur de noyau fonctionnel, contrôleur de dialogue et présentation (voir figure 2.8).

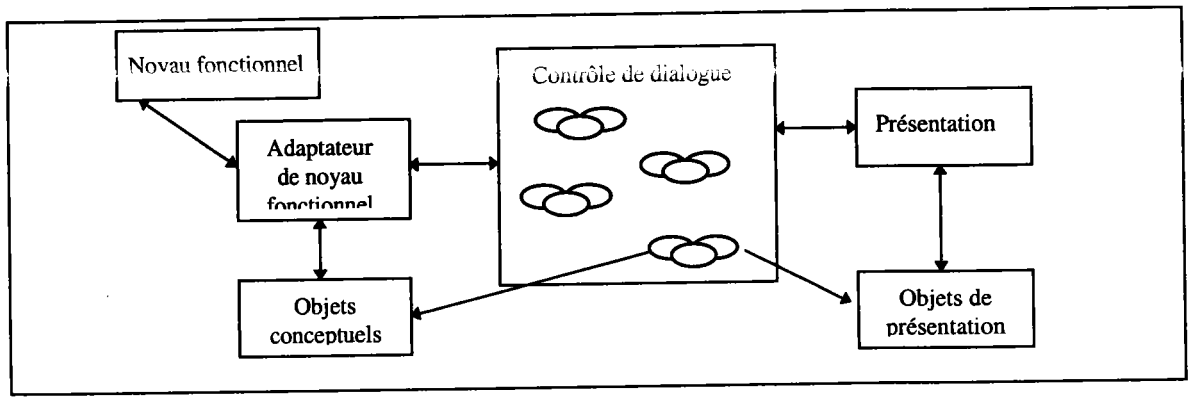


Figure 2.8 Le modèle PAC-Seeheim

- Le noyau fonctionnel : regroupe les concepts propres au domaine de l'application indépendamment de toute interface utilisateur.

- L'adaptateur du noyau fonctionnel : un protocole entre le « noyau fonctionnel » et le contrôleur de dialogue » est implémenté dans ce composant avec la définition des objets conceptuels.

- Le contrôleur de dialogue : il permet l'enchaînement des tâches, des transformations de formalisme, et la mise en place du mécanisme de correspondance entre les objets de la présentation.

- La présentation : regroupe les objets de la présentation puisé dans une boîte à outils, ce composant est décomposé en trois couches qui sont une boîte à outils, une boîte à outils virtuelle et une extension personnalisée de cette boite à outils virtuelle.

Ce modèle est plus précis que les autres modèles, néanmoins, il est trop flou en ce qui concerne sa mise en œuvre.

5.5 Le modèle Arche

Ce modèle est une extension du modèle Seeheim, il a été présenté lors de séminaires regroupant des développeurs de gestionnaires d'interfaces utilisateurs [Bass, 91]. Il est constitué de cinq composants : boites à outils, présentation, dialogue, adaptateur de domaine, et domaine (voir figure 2.9).

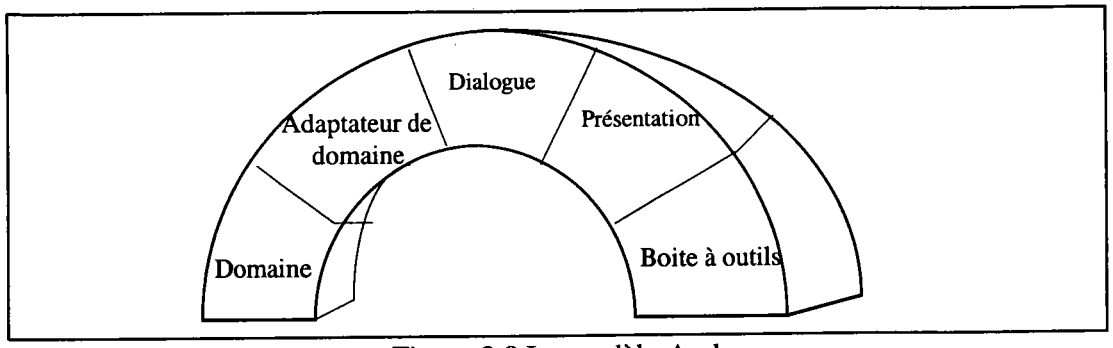


Figure 2.9 Le modèle Arche

- Le domaine : c'est l'application, il contient les fonctionnalités qui sont indépendantes de toute interface utilisateur.

- L'adaptateur du domaine : c'est l'intermédiaire entre le composant « domaine » et le composant « dialogue », il implante les tâches relatives au domaine nécessaire à une utilisation par un opérateur. Ce composant déclenche aussi des tâches de dialogue et réorganise les données du composant « domaine ».

- Le dialogue : il permet le séquencement des tâches de l'opérateur et de la partie du séquencement du « domaine » qui dépend de cet opérateur. Cette composante fait la correspondance entre le formalisme du domaine et l'interface utilisateur, ainsi que le maintien de la cohérence entre plusieurs vues d'un même concept du domaine.

- La boîte à outil : constituée d'un ensemble de classes d'objets d'interaction, qui utilisent un média particulier avec certaines possibilités de gestion des événements à travers Motif par exemple.

Ce modèle ne propose pas d'approche répartie pour la conception d'applications, mais néanmoins il est plus précis par rapport au modèle Seeheim. Par son architecture plus découpée le modèle ARCHE se montre plus à même que le modèle de Seeheim de supporter les changements, qu'ils se produisent au niveau du domaine, du contrôle ou des outils d'interaction. Cependant, il ne rentre pas dans la structure interne des composants (programmation fonctionnelle, modulaire ou objet par exemple) et ne définit pas de façon précise l'interface informatique entre les différents composants. De multiples interprétations peuvent donc en être élaborées.

5.6 Le modèle SPA (Seeheim-PAC-Arche)

Ce modèle divise un système interactif en une hiérarchie d'objets SPA [Duval, 92a,92b,92c]. Chaque objet est lui même divisé en six composants : présentation, interface présentation, contrôle, interface contrôle, application et interface application (voir figure 2.10).

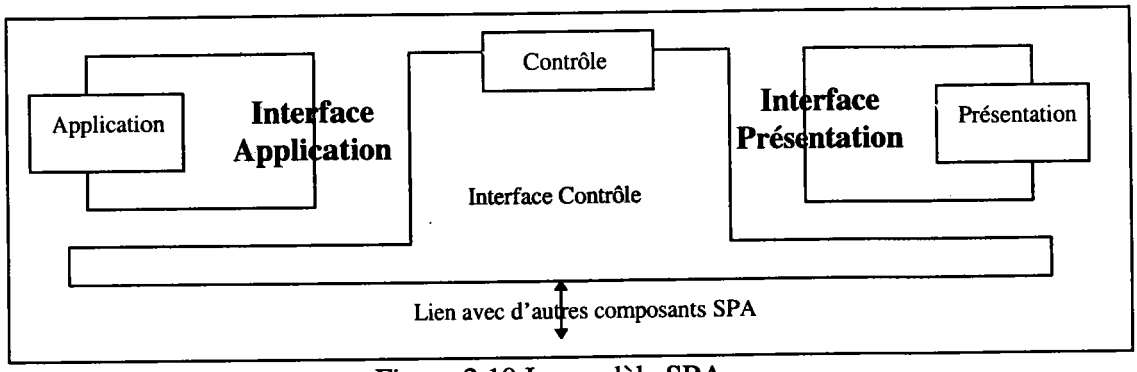


Figure 2.10 Le modèle SPA

- La présentation : cette partie implémente l'interaction physique avec l'utilisateur.
- L'interface présentation : définit les attributs et fonctionnalités du composant présentation que ce dernier exporte au niveau du composant contrôle.
- Le contrôle : il est identique au composant « contrôle du dialogue » du modèle Seeheim.
- L'interface contrôle : c'est la spécification du composant contrôle, c'est-à-dire qu'il définit les primitives du composant contrôle, et ensuite les envoie vers les autres objets du système.
- L'application : il contient l'implémentation des fonctionnalités de l'application indépendamment de toute interface utilisateur.
- L'interface avec l'application : intermédiaire entre le composant « application » et le composant « contrôle » ; le fonctionnement de ce composant a le même principe que celui du modèle Seeheim.

Ce modèle peut être considéré comme une synthèse des modèles Seeheim, PAC et Arche. L'interprétation qui est faite de ce modèle est une amélioration plus fine du modèle PAC (les échanges sont mis en évidence entre les composants : application, présentation, contrôle). Une autre interprétation est l'interprétation répartie du modèle Arche en particulier par rapport au composant « application ».

5.7 Conclusion

L'objectif des modèles présentés ci-dessus est de définir les composants susceptibles d'être développés avec une relative indépendance. De ce fait un modèle comme Seeheim est un modèle d'interface et non d'applications interactives. Les accès à l'application dans ce modèle s'expriment par des appels provenant de l'interface, ce qui se traduit par des demandes de services. L'application rend des résultats par affichage. Or, toutes les applications interactives ne sont pas modélisables de cette manière.

La plupart des modèles à objets se développent à partir de la décomposition faite par le modèle de Seeheim. De ce fait, on s'est occupé à la modélisation en objets de la partie interface de l'application interactive. Des modèles tel que PAC essayent de remédier à cela en proposant un modèle à objets homogènes c'est-à-dire l'objet contient en même temps sa présentation, son contrôle et son abstraction qui correspond à sa partie traitement.

Nous terminons cette étude sur la problématique de l'interactivité Homme-Machine par la présentation du système de dialogue S.A.C.A.D.O (Système Adaptatif de Conception Assistée et de Développement par Ordinateur).

6 Le système de dialogue S.A.C.A.D.O

6.1 Introduction

Le système de dialogue S.A.C.A.D.O est développé au laboratoire de recherche en informatique de Metz, il est le système fédérateur du laboratoire. Un des principaux buts du système SACADO est la modélisation du dialogue Homme-Machine. La première présentation du système SACADO est faite dans [Gardan, 88], ensuite un formalisme à été développé pour modéliser le dialogue [Gardan, 93].

6.2 Les bases du système S.A.C.A.D.O

Les bases du dialogue dans S.A.C.A.D.O sont situées à deux niveaux :

- un dialogue séquentiel, c'est-à-dire que l'utilisateur se déplace d'un état à un autre en n'exécutant qu'une seule tâche à un instant donné,
- un dialogue simultané, c'est-à-dire que l'utilisateur peut exécuter plusieurs tâches simultanément.

Une notion importante a été définie dans S.A.C.A.D.O c'est celle de générateur permettant la construction de systèmes. Il existe deux sortes de générateurs :

- des générateurs primaires : ils permettent de compléter les actions déjà définies dans l'implantation S.A.C.A.D.O,
- des générateurs applicatifs : ils permettent de créer une application qui utilise une implantation S.A.C.A.D.O.

En outre trois familles d'utilisateurs peuvent intervenir :

- l'opérateur (ou utilisateur final),
- le programmeur d'application : il se charge d'écrire le code pour les fonctions utilisées par l'opérateur,
- le concepteur d'interface : il se charge de construire l'interface du système permettant à l'opérateur de dialoguer avec le système.

6.3 Caractéristiques du système S.A.C.A.D.O

Ce modèle permet de représenter l'interface d'une application avec des mécanismes de compatibilités (inaccessibles, immédiats, locaux et différés). L'affichage de ces compatibilités par l'intermédiaire d'un code de couleurs permet à un utilisateur de savoir quels sont les menus locaux qui peuvent permettre de terminer la construction en cours. L'utilisateur peut aussi voir immédiatement les menus immédiats qui lui permettent de faire autre chose.

Un autre point important est que le modèle de dialogue de SACADO permet aussi de décrire les phases de dialogues associées à chaque menu.

Une phase de dialogue est une action globale qui peut contenir deux types d'actions : les interactions et les actions non interactives. Les interactions permettent de décrire les actions attendues de la part de l'utilisateur. Les actions non interactives permettent, par exemple, de consulter le modèle de données ou d'effectuer des calculs.

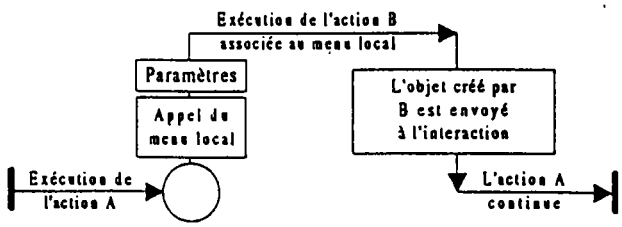


Fig 2.11 Menu local

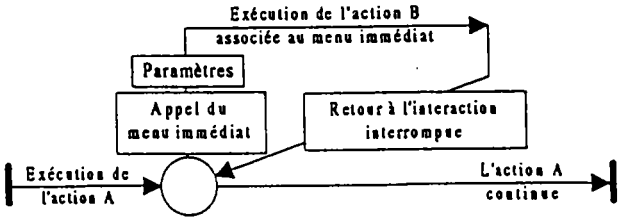


Fig 2.12 Menu immédiat

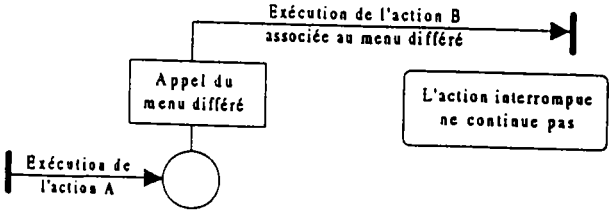


Fig 2.13 Menu différé

6.4 L'architecture générale

La figure suivante (figure 2.14) représente l'architecture de l'interface de S.A.C.A.D.O. Le moniteur de dialogue joue un rôle central dans l'ensemble du système de l'interface. Il contrôle en effet les actions physiques de l'utilisateur, ensuite il recherche le menu activé dans la base des menus, il va également rechercher, dans la base des programmes appelée NADRAG (qui représente le langage d'action) le programme à lancer et, enfin, appelle la fonction indiquée dans ce programme dans la bibliothèque des fonctions geométriques.

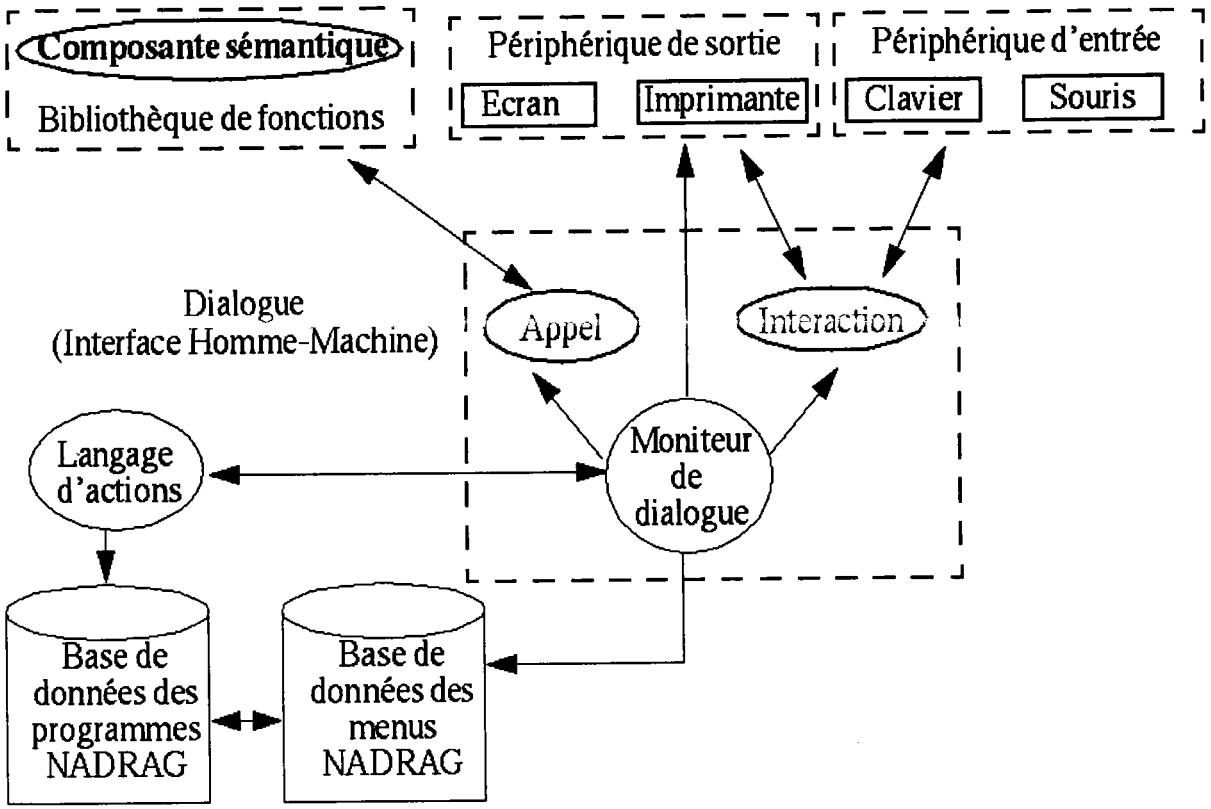


Figure 2.14 Architecture générale de l'interface de S.A.C.A.D.O

7 Conclusion

A partir de l'étude qui a été faite dans ces deux premiers chapitres nous disposons d'une base de travail suffisamment riche pour proposer notre modèle. En effet, l'étude sur les systèmes experts et la planification nous permettra de constituer les modules associés aux processus de conception. Pour ce qui est de l'interface dialogue, l'étude complète des systèmes de logiciels interactifs nous permettra de proposer des mécanismes interactifs adéquats suivant le profil de l'utilisateur (expert, non expert).

La base de ce travail reposant sur l'intégration de l'outil « planification interactive » qui joue le rôle de pivot dans un ensemble, que l'on appellera « système d'assistance ». C'est-à-dire qu'il est le lien entre une base de connaissances produite (à travers les données) et une base fonctionnelle (les modes de conception choisis par l'utilisateur). Il permet aussi sur la base d'une interactivité planifiée de pouvoir gérer le dialogue système-utilisateur à travers la gestion de la procédure du dialogue, de l'historique du dialogue et des alternatives.

L'architecture générale du système a été définie pour s'adapter au profil de l'utilisateur (expert ou non expert) qui intervient pendant une session, elle se compose de trois parties principales.

1- La base de connaissances :
Elle contient les données et les relations les liant ainsi que la représentation des règles et des contraintes.

2-La base fonctionnelle :
La base fonctionnelle a un double rôle :

- elle analyse et valide éventuellement le produit candidat grâce à des procédures spécifiques qui sont adaptées à des profils d'utilisateurs différents,
- elle stocke les différentes alternatives validées pendant une session de conception et ceci, à travers des versions plus ou moins réussies en fonction de la complexité de l'assemblage ou de la construction produite.

Le niveau fonctionnel permet donc de contribuer à l'évolution du système ainsi qu'à la définition précise du modèle de conception d'assemblage choisi. L'idée générale est de permettre à l'utilisateur non seulement d'apprendre, mais aussi d'enrichir le système par de nouvelles connaissances produites au cours d'une session.

3-La planification de la conception et du dialogue :
Le planificateur est constitué de deux parties, le générateur de plan et un moteur d'inférence. Le générateur représente la partie algorithmique, il se charge de l'identification des opérateurs pouvant atteindre un but donné, il représente les connaissances minimales nécessaires pour construire un plan. Le moteur d'inférence représente la partie qui permet de réduire l'espace de recherche pour choisir le but à poursuivre, l'opérateur à appliquer ou les objets à lier aux variables d'un opérateur. A chaque fois qu'il y a un choix à faire on fait intervenir le moteur d'inférence.

Au fur et à mesure de l'acquisition de connaissances, le moteur d'inférence s'enrichit de nouvelles règles, améliorant à la fois l'efficacité du processus de planification et la qualité des plans générés. L'architecture proposée s'appuie sur l'expertise du moteur d'inférence d'ordre 1 qui affine et augmente cette expertise en vue d'optimiser le comportement global du système.

Chapitre 3

Une proposition de système d'aide à la conception et à l'assemblage fondée sur la planification interactive

1 Introduction

La problématique posée par l'interactivité dans les systèmes de C.A.O est résolue de différentes manières selon les applications qui sont prises en compte.

Le but de ce travail est de monter que la planification interactive peut contribuer à l'amélioration de l'interactivité dans les systèmes de C.A.O pendant la conception et ceci, indépendamment de l'application qui s'y déroule. Pour cela, on associe aux mécanismes de la planification interactive un environnement approprié qui regroupe les fonctions suivantes :

1- un mode de conception adapté au profil de l'utilisateur (expert ou non expert),
2- la gestion du dialogue,
3- l'adaptabilité à la base de connaissances,
4- une architecture modulaire et multi-niveau.

A partir d'un objectif spécifié par l'utilisateur on génère automatiquement une chaîne d'opérateurs répondant à cette demande. L'intérêt d'un tel système vis-à-vis de l'aide à la conception est évident. En effet, pour réaliser ses objectifs l'utilisateur n'a plus besoin de maîtriser certaines connaissances sur les traitements liés à la C.A.O. Le système ainsi défini est couplé à un modeleur géométrique C.A.O de type B-Rep ou CSG.

Les caractéristiques opératoires du modeleur ne sont pris en compte qu'au moment où le système produit une séquence de commandes (sous la forme de prédicats) qui sont ensuite traduites dans les primitives du modeleur (voir figure 3.1).
Le couplage système d'aide/modeleur est défini à travers la notion de multi-modèles [Gardan & Zakari, 91], c'est à dire que le système est conçu indépendamment du modeleur et, par conséquent, peut s'y adapter. Cette partie sera développée dans le chapitre 4.

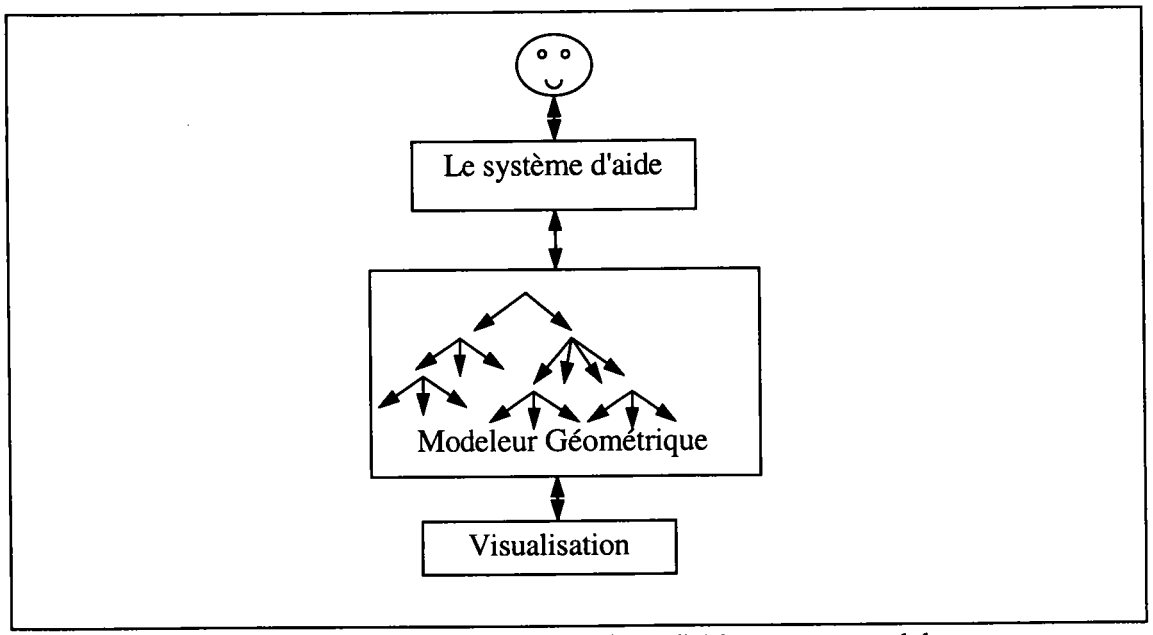


Figure 3.1 Le couplage du système d'aide avec un modeleur

Le système d'aide que nous proposons doit être vu comme un outil en complément à un système de C.A.O classique. Il utilise les techniques d'intelligence artificielle comme la planification. C'est pour cette raison que nous proposons dans la première partie de ce chapitre une description succincte des concepts attachés aux systèmes dits "intelligents" d'assistance à la C.A.O, c'est à dire possédant des mécanismes d'inférence. D'autre part, nous avons choisi comme environnement de travail des aspects liés à la construction mécanique, en particulier ceux propres à l'assemblage de pièces mécaniques. Sans rentrer dans les détails fonctionnels de la construction mécanique nous décrivons les méthodes de représentation d'objets mécaniques dans le cas de l'assemblage à travers les modèles existants. Cela nous permet de structurer les objets construits en cours de la conception. Dans la dernière partie de ce chapitre nous proposons une architecture globale de notre système d'aide en décrivant chaque module qui le compose.

2 Notion de système intelligent en C.A.O

2.1 Définitions et concepts

Le processus de conception en C.A.O ou en C.F.A.O est un processus difficile à automatiser. En effet, pour obtenir cette automatisation il faut pouvoir remplacer le concepteur par un système que l'on peut considérer comme « partiellement intelligent de C.A.O », ainsi, plus le système est capable de remplacer le concepteur, plus il est dit « intelligent ». La difficulté réside dans la complexité de ce processus, pour cela la mise en place doit se faire de manière graduelle.

Un système dit « intelligent » doit prendre en compte des traitements non algorithmiques, gérer des connaissances avec leurs données sémantiques, fournir des règles de déduction et de cohérence, définir une interface utilisateur adaptable et évolutive pour prendre en compte la diversité des utilisateurs et des situations. En effet, une vision plus globale des problèmes de conception doit être étudiée car on ne peut pas se contenter des traitements qui ont déjà été analysés, c'est-à-dire des traitements purement algorithmiques, sans mécanisme de déduction et d'apprentissage.

A partir de ce constat, on peut définir les fonctionnalités globales que devrait posséder un système dit "intelligent" :

- Fournir des fonctionnalités intelligentes, c'est à dire des mécanismes d'inférence pour que même un concepteur non spécialiste puisse utiliser son application correctement.
- Prendre en compte les intentions du concepteur, c'est-à-dire détecter les erreurs le plus tôt possible et proposer des alternatives à chaque fois que cela est nécessaire.

Des efforts dans le cadre d'une meilleure intégration d'outils d'intelligence artificielle en particulier pour les systèmes experts, ont vu le jour. C'est ce que nous décrivons dans la suite.

2.2 L'approche système expert

L'approche système expert comme nous l'avons vu permet la constitution d'une base de connaissances, la gestion de la cohérence, de la déduction, et en général la manipulation de la sémantique des informations. Néanmoins, il s'avère que le problème de leur utilisation réside dans leur limitation à des domaines très ciblés, or en C.A.O et pour des produits complexes, on se retrouve dans un contexte de travail où le domaine de compétences doit être vaste, et nécessite la participation de plusieurs concepteurs, chacun compétent dans son domaine. A partir de là deux solutions sont possibles :

- Utilisation d'un "super système expert" c'est à dire un système capable de prendre en compte tout les aspects de la conception. Il est évident que cela paraît illusoire, en effet cela implique de renfermer dans une seule base de connaissance toutes les informations de la conception, et de les prendre en compte en même temps. La nature très différente des informations à modéliser rend cette méthode difficile à réaliser, d'autant plus qu'aucun système à l'heure actuelle n'est opérationnel avec cette approche.

- Utilisation d'un "système multi-expert" (on parle aussi de société d'experts) c'est-à-dire de plusieurs petits systèmes experts, les informations sont organisées suivant la nature de chaque système expert. Le problème réside dans la coopération entre les différents systèmes et de leurs supports. Cette méthode est actuellement la plus utilisée.

Lorsqu'une approche système expert est choisie, elle doit être complétée par une interface adéquate. Comme nous l'avons vu dans le chapitre 2, l'aspect interface doit compléter efficacement l'application produite à travers un modèle. Cela permet de donner au système de C.A.O. toute sa valeur.

2.3 L'interface Homme-Machine

Les résultats de l'étude faite dans le chapitre 2 montre que l'interface dédiée aux systèmes de C.A.O peut bénéficier au niveau plus particulier de l'assistance à l'utilisateur. Cette assistance peut être de plusieurs natures :

- Syntaxique, c'est à dire fournir des informations sur la mise en œuvre d'une commande.
- Conseil d'utilisation pour répondre à des questions relatives à la manière d'utiliser le logiciel de C.A.O. proposé.
- Faire un diagnostic d'erreur efficace.

A côté de ces trois types d'assistance souhaitables, dans tout logiciel de C.A.O, il est important d'intégrer des outils d'aide plus spécifiques et qui tiennent compte du profil de chaque utilisateur : utilisateur néophyte, occasionnel, expérimenté.
Ces outils adaptables pour chaque utilisateur doivent, d'une part guider l'utilisateur à naviguer dans le système c'est à dire :

- faire connaître les commandes et leurs organisations en environnement,
- fournir des explications des erreurs,
- fournir des aides en relation avec la compétence de l'utilisateur, avec l'état de l'objet conçu dans l'espace de développement.

D'autre part, ces outils doivent aider l'utilisateur à se servir efficacement du système c'est-à-dire :

- prendre en compte la position dans l'espace de développement de l'objet à concevoir et de l'objet conçu pour suggérer à l'utilisateur des stratégies de travail,
- aider l'utilisateur dans sa réflexion sur le "comment s'y prendre".

Avec la définition d'un système intelligent, nous devons montrer quel type de processus de conception il doit prendre en compte, et jusqu'à quel niveau. Pour cela, nous allons d'abord définir la classification du processus de conception général et ensuite préciser l'environnement dans lequel on évolue.

3 Classification du processus de conception

3.1 Le déroulement de la conception

Les travaux des différents spécialistes du domaine ont abouti à l'élaboration d'un schéma détaillé des étapes qu'il faut suivre lors de la conception d'un nouveau produit [Kimura, 89] [Yoshikawa & Arbab, 89]

1- La conception créative : c'est la première étape lorsque la conception est nouvelle, elle est rare en mécanique, mais fréquente par exemple dans le bâtiment et les industries chimiques.
2- Le "conceptual design" c'est à dire le choix de la méthode et de la structure de base pour le processus de conception.
3 - La conception de base : première version de la conception appelée aussi découpage global en mécanique ou bien avant projet sommaire en architecture.
4- La conception détaillée : toutes les composantes de la conception sont détaillées, dimensions et positions.
5- La production : on génère les données qui seront utilisées pour la fabrication d'un prototype.
6- Test : des essais de fabrication sont réalisés et des tests sont effectués.
7- Re-conception : les premiers tests sont effectués sur le prototype montrent le besoin de revenir en arrière pour modifier les paramètres.
8- Contrôle et diagnostic.
9- Maintenance et formation.

Pour mettre en relief les étapes abordées ci-dessous et fixer les idées, nous allons aborder le cas de la construction mécanique, car c'est un domaine où le travail de conception est important et où les travaux sur la mise en place d'outils d'aide à la conception ne sont pas très répandus.

3.2 La construction mécanique

3.2.1 Chronologie de la conception

La construction mécanique est un domaine qui regroupe des thèmes très différents par leurs approches et leurs finalités. On peut distinguer dans l'ordre :

1 - L'analyse des besoins
Elle fait intervenir l'établissement d'un cahier des charges qui lui-même fait appel à l'analyse du marché (informations sur le fournisseur, les concurrents, la sous-traitance...) en évaluant les besoins du demandeur (questionnaire suivi d'un entretien).

2 - L'analyse fonctionnelle
Il s'agit de définir les fonctions et les coûts associés à la conception d'un produit.
Pour ce qui est des fonctions, il existe plusieurs types de fonctions interdépendantes, d'une part les fonctions globales, c'est à dire les fonctions dont le rôle est joué par un moteur ou un ensemble destiné à remplir une fonction de service (voir figure 3.2), d'autre part, les fonctions composantes qui sont des fonctions hiérarchisées et qui peuvent être indispensables à la réalisation de la fonction globale. A ces fonctions on peut rajouter les fonctions spécifiques que sont les fonctions de guidage ainsi que les fonctions de contrainte.
Pour ce qui est des coûts, trois principales évaluations interviennent, celles qui concernent le produit lui-même, celles plus liées aux fonctions, et enfin, le coût des composants technologiques.

3 - Etude d'avant projet
Il s'agit de justifier le choix d'un projet parmi plusieurs qui se présentent, on appelle aussi cette partie pré-développement.

4 - La conception définitive
C'est la spécification technique et économique du produit associé à une organisation industrielle qui doit permettre la définition du produit (dessin, étude du procédé de fabrication).

5 - Industrialisation
Cela correspond à la mise en place de l'outil de production (gamme, temps, outillage, contrôle...).

On peut représenter l'ensemble de ce processus par les deux schémas ci-dessous.
Le premier représente les étapes de l'élaboration d'un produit et le second les fonctions globales d'un moteur en mécanique.

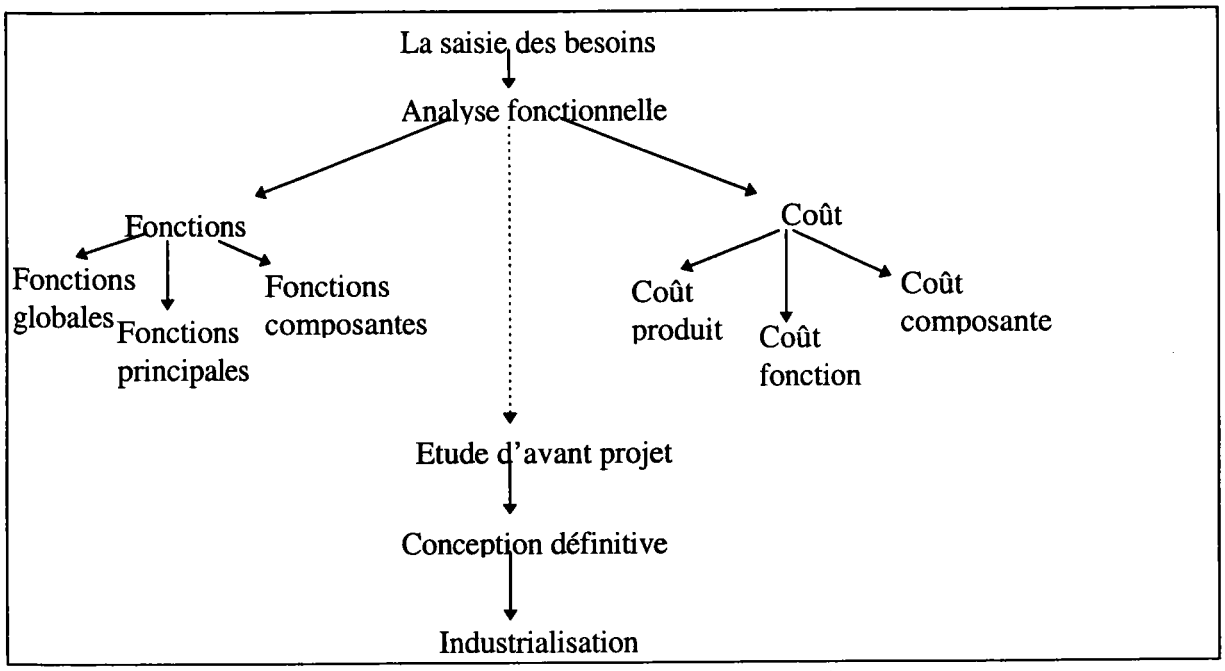


Figure 3.2 Schéma d'élaboration d'un produit

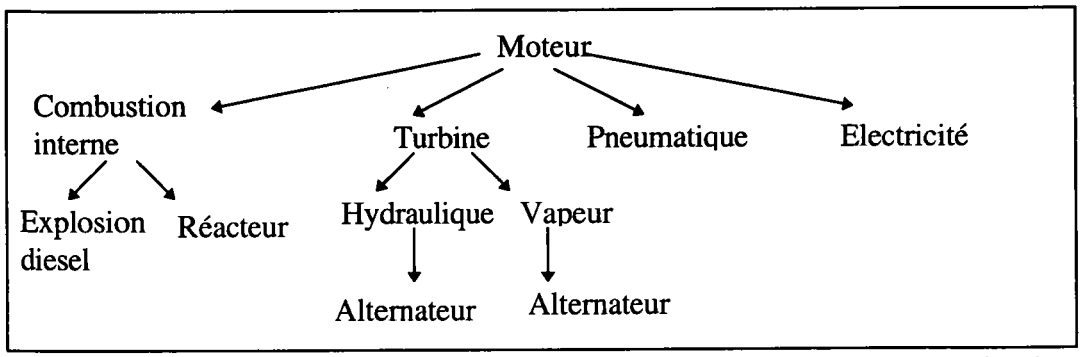


Figure 3.3 Schéma des fonctions globales possibles en construction mécanique

3.2.2 L'assemblage en construction mécanique

Dans le domaine de la construction mécanique tout produit apparaît comme l'union de composants indivisibles dont le résultat à l'assemblage fournit le produit lui-même. De même pour la robotique par exemple, l'assemblage automatisé d'un produit implique deux activités essentielles : la génération de la séquence d'assemblage adéquate et la programmation du robot.

La génération de la séquence d'assemblage consiste en la détermination de l'ordre précis d'assemblage à partir de la combinaison des composants du produit. Cet ordre d'assemblage dépend souvent de la géométrie des composants. En effet, les caractéristiques géométriques de quelques composants suffisent à réduire considérablement l'ordre d'assemblage.

Les travaux dans ce domaine sont nombreux, mais utilisent pour la plupart le même principe. Par exemple, les travaux de Bourjault et de Whitney [Boujault 84] [Whitney et al 88] proposent de modéliser l'assemblage dans un graphe de liaison qui est, par définition, un simple graphe où les arêtes correspondent aux composants et les arcs correspondent aux contacts fonctionnelles entre les composants à assembler. L'utilisateur fournit une liste de relations de précédence qui contraint l'ordre d'établissement des liaisons pour construire le produit.

Ces relations de précédence sont définies par les réponses à une série de questions proposées par le système qui génère toutes les séquences d'assemblage mécanique qui satisfont les relations de précédence définies par l'utilisateur. La différence entre les travaux de Bourjault et Whitney tient au nombre de questions posées.

D'autres travaux comme ceux de De Mello et Sanderson [De Mello 89] se basent sur une approche par décomposition c'est-à-dire que le produit est transformé en une série de séquences de désassemblage. Cette approche commence par la construction du modèle relationnelle de l'assemblage dans un graphe où les arêtes sont soit des contacts soit des fixations et où les arcs représentent les différentes relations entre les paires d'arêtes. Une autre, approche est celle proposée dans les travaux de Laperriere et ElMaraghy [Laperriere 91] où les relations entre les composants d'assemblage sont modélisées interactivement à travers un diagramme de relations

En conclusion, la mise en place d'un système qui puisse prendre en charge l'ensemble de ces fonctions n'est pas disponible à l'heure actuelle. Néanmoins des systèmes (où une modélisation partielle des éléments cités auparavant) deviennent de plus en plus nombreux. On peut citer CII (Conception Intégré Intelligente) [Hernot & al 95], DEKLARE [Vargas & al, 95].
Le premier est un outil de conception alliant les avantages de la C.A.O et l'IA. Le second est une démarche expérimentale visant à la mise en œuvre de développement d'applications d'aide à la conception d'organe mécanique. Une des techniques de modélisation parmi la plus utilisée repose sur les mécanismes des systèmes experts comme nous l'avons détaillée dans le chapitre 1.

3.3 La problématique de l'interaction dans le cas de la C.F.A.O

Le problème posé est celui de l'assistance apportée à un utilisateur et qui soit la plus ouverte possible. C'est-à-dire ouvert sur les applications proposées et ouvert sur les différents profils d'utilisateurs. Les utilisateurs des systèmes de C.A.O aboutissent presque tous à la même conclusion que le systèmes de C.F.A.O sont très complexes. En effet, il est toujours possible d'appliquer une des commandes disponibles, en revanche il est moins évident de choisir la bonne commande (ou la succession de commandes) en respectant la logique de l'outil utilisé.

De plus la richesse sans cesse croissante des logiciels de C.F.A.O rend de plus en plus difficile l'apprentissage de ces logiciels. Quelquefois même un utilisateur expérimenté ne connaît pas toutes les commandes (ou les enchaînements de commandes) qu'il veut obtenir par rapport à ces spécifications.

On constate aussi que le processus suivi par deux utilisateurs expérimentés pour traiter le même problème (bien que souvent ayant la même solution) est atteint par des cheminements différents. En outre, l'absence de méthodologies d'utilisation du logiciel est un handicap à la bonne maîtrise des résolutions des problèmes identifiés pour créer des objets type.

Tous ces problèmes peuvent être partiellement résolus par l'utilisation d'une documentation mais les études ont montré que les utilisateurs n'appréciaient que très faiblement la consultation de manuels surtout si cette documentation est très riche ou complexe.

3.4 L'intérêt d'une assistance intelligente en C.F.A.O

L'intégration d'un système d'aide intelligent à l'utilisation d'un système de C.F.A.O entraînerait une productivité plus grande des concepteurs. Cependant, si on cherche à fournir ce genre d'assistance, il faut pouvoir prendre en compte l'intégralité des connaissances à manipuler, à savoir les connaissances relatives au fonctionnement général du système interactif et celles relatives au contexte d'utilisation. D'autre part, l'intégration du savoir-faire des experts permettrait au système d'aide de suggérer à l'utilisateur des méthodes pour la conception de produits connus avec, en plus, la prise en compte de critères technologiques spécifiques à l'application.

4 L'activité de conception en C.A.O : vers une solution

La première partie de notre étude était axée sur la notion de système expert, d'éléments d'interface, et de planification (la partie interface à été développée plus longuement dans le chapitre 2). L'idée était d'avoir d'une part, une vue générale des systèmes experts, en particulier ceux dédiés à la C.A.O et, d'autre part, de voir quels sont les outils susceptibles de contribuer à la création de l'environnement qui nous permet de réaliser les objectifs cités auparavant.

De ce fait, le processus de conception est analysé en terme de tâches ou de buts à accomplir par l'utilisateur, ce qui nous permet de mettre en évidence les sous-tâches ou sous-buts qui le composent, leurs entrées-sorties et les conditions de leurs déroulements. On introduit ainsi une vue globale de la tâche de conception que nous nuançons ensuite progressivement en la détaillant. Le schéma global d'une solution est ensuite proposé, puis affiné. Cette analyse a pour but d'identifier clairement les objets et processus que le système d'aide fait intervenir. Il permet aussi de justifier l'architecture pour l'environnement de conception du système d'aide que proposons dans la section 5 de ce chapitre.

On se propose de décrire ce processus dans ce qui suit.

4.1 L'application/La tâche

L'application peut se définir comme un ensemble de compétences logicielles permettant de réaliser des tâches. La tâche est caractérisée par un but à atteindre. La démarche de l'utilisateur va consister à faire évoluer la tâche d'un état initial vers un état final, qui constitue son but par le biais d'actions successives sur l'état courant.

C'est au niveau du modèle de la tâche que sont définies les conditions d'application des actions sur les objets de la tâche ainsi que leur effets. Il est possible de voir ce modèle comme la description des objets de l'application et des tâches ou des actions élémentaires réalisables dans le cadre de l'application.

Le plus souvent, l'utilisateur agit sur la tâche via une visualisation sur l'écran de l'ordinateur. Cette visualisation est une représentation physique virtuelle d'un état de la tâche comprenant ses objets, ses liens entre objets et un comportement associé à chacun d'entre eux.

4.2 Modélisation en termes de buts ou de tâches

Notre problématique est centrée sur de la représentation d'un processus de conception (l'assemblage en construction mécanique) qui possède certaines spécifications (décrites précédemment) et qui requièrent une modélisation. Pour expliciter ce modèle, précisons que les utilisateurs potentiels visés sont :

- soit des utilisateurs confirmés, on parle plus souvent d'utilisateurs experts dans le domaine de la tâche,
- soit des utilisateurs épisodiques, non spécialistes du domaine de la tâche.

Nous supposons ensuite que notre système est associé à un modeleur de type B-Rep ou CSG. Nous avons en outre choisi de considérer l'activité de conception comme une tâche à part entière, c'est-à-dire comme la réalisation d'un but donné dans des conditions déterminées [Nanard, 90]

4.3 Analyse de la tâche de conception

Dans le domaine de la conception en C.A.O et en particulier dans le cas de l'assemblage il faut prendre en compte l'organisation globale de la tâche à accomplir par l'utilisateur. Deux éléments d'organisation interviennent : la décomposition hiérarchique en sous-tâches et l'organisation des données autour de ces tâches.

Dans le premier cas, l'utilisation de la planification interactive comme un outil qui peut servir à la fois à créer cette décomposition et à mettre en place un dialogue conséquent.

Dans le deuxième cas, il faut créer un environnement qui puisse prendre en compte les données générées par le système de planification. Une base de connaissances (avec ses règles, ses faits et ses contraintes) est appropriée aux niveaux des données. Une base fonctionnelle est tout aussi appropriée pour gérer les profils des différents utilisateurs.

Enfin, toujours dans le cas de l'organisation globale de la tâche, sans une interface adéquate l'ensemble du système ne peut répondre de façon précise aux sollicitations des utilisateurs potentiels.

L'ensemble de ces modules doit fonctionner autour d'une architecture choisie pour permettre le bon déroulement des enchaînements des tâches à accomplir et garantissant une vision globale de l'organisation de la conception. C'est ce que nous décrivons dans la partie suivante.

5 Le modèle proposé et son architecture

5.1 Introduction

La base de ce travail repose sur l'intégration de l'outil " planification interactive " qui joue le rôle de pivot dans l'ensemble du système d'assistance. C'est à dire qu'il est le lien entre la base de connaissances produite (à travers les données) et la base fonctionnelle (les modes de conception choisis par l'utilisateur). Il permet aussi (toujours sur la base d'une interactivité planifiée en forme de buts/tâches) de pouvoir gérer le dialogue système-utilisateur à travers la gestion de la procédure du dialogue, de l'historique du dialogue, et des alternatives.

Le modèle défini, comme indiqué sur la figure 1.3 comporte plusieurs modules, qui agissent de manière autonome mais complémentaire. L'architecture a été définie pour s'adapter au profil de l'utilisateur (expert , non expert). Nous allons dans ce qui suit décrire chaque module qui compose le système (sauf la partie interface qui sera décrite plus tard) d'aide à l'exception du module de générateur de réponses qui permet la mise en place de réponses générées par le système sous une forme compréhensible par l'utilisateur.

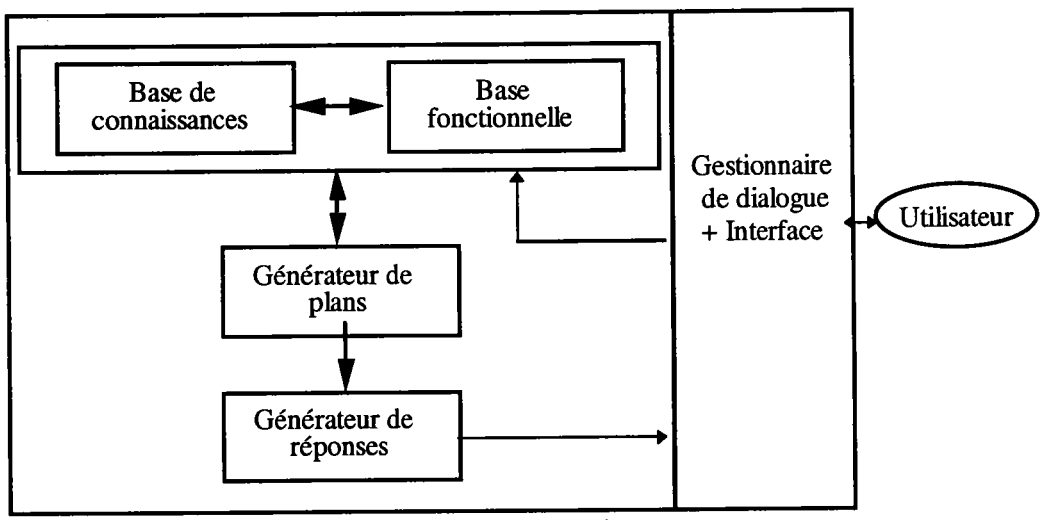


Figure 3.4 Architecture globale du système

5.2 L'algorithme principal du système d'aide

Le fonctionnement général est le suivant :

- L'utilisateur fournit les grandes lignes de l'objet qu'il souhaite élaborer sous forme d'actions interprétables par le système.
- L'utilisateur choisit le mode de conception (consultatif ou créatif)
- Le système charge la base de connaissances pour la session courante de conception.

Deux cas possibles :

1 - En mode consultatif :
Le système cherche dans sa base (base des méthodes) des méthodes applicables, sinon il consulte les versions déjà produites dans les sessions précédentes.

- Soit il trouve, d'où proposition à l'utilisateur de la séquence de commandes correspondant aux critères de choix souhaités.
- Sinon il demande à l'utilisateur de décomposer sa demande en sous-buts plus accessibles et recommence le processus.

2 - En mode créatif :

- Le système propose à l'utilisateur des alternatives selon son profil (expert, non expert) jusqu'à satisfaction.
- Le système enregistre le processus de création pour bâtir de nouvelles méthodes.

Une représentation des différentes étapes parcourues par l'utilisateur est décrite par la figure3.5.

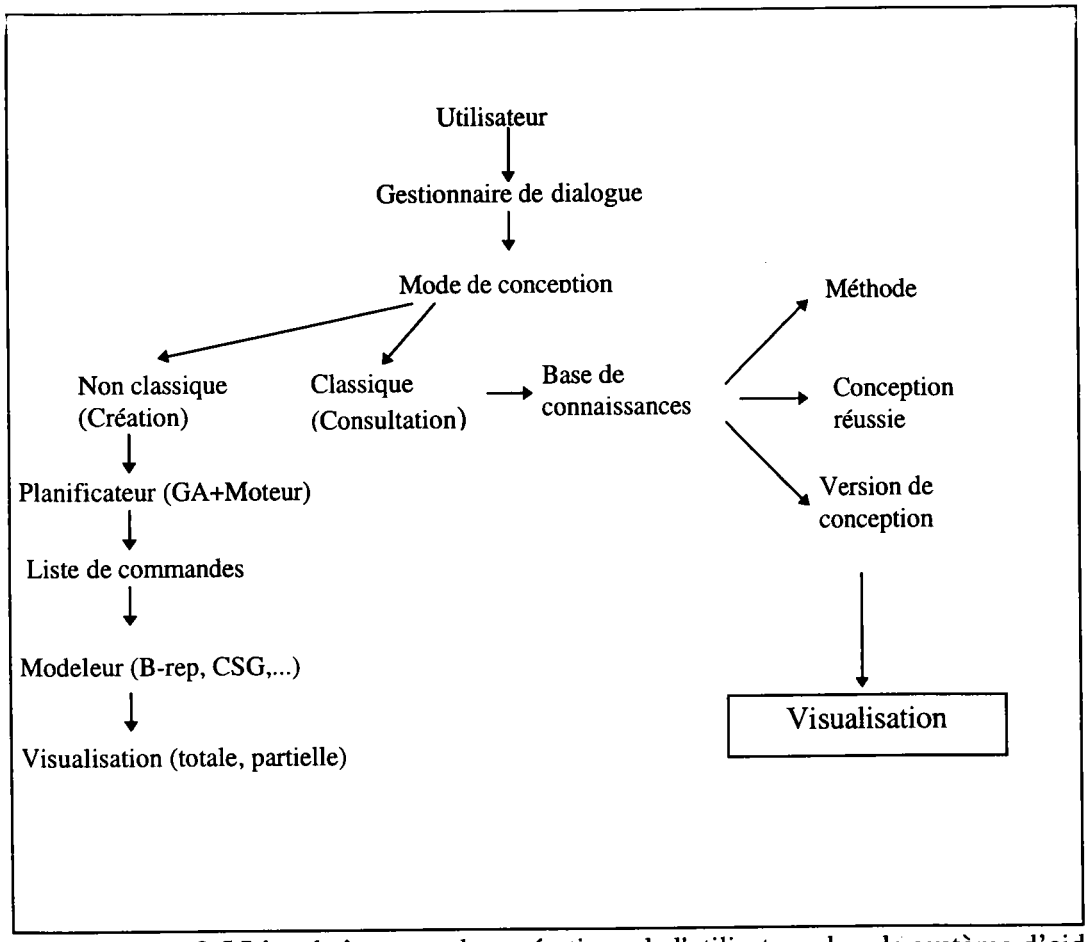


Figure 3.5 L'enchaînement des opérations de l'utilisateur dans le système d'aide

6 Description du système proposé

Nous décrivons dans cette partie les différents modules du système d'aide proposé, ainsi que les interactions qui en découlent.

6.1 La base de connaissances

Elle contient les données et les relations les liant ainsi que la représentation des règles et contraintes.

6.1.1 Catégories des données

La base de connaissances est constituée, dans sa partie objet, de trois catégories : le produit, les composants et les caractéristiques géométriques.

1 - Le produit est un objet composé construit à partir de l'assemblage de ses composants.

2 - Les composants sont de deux types :

- Les composants standards comme les vis, les écrous, les surfaces d'appui etc., et qui sont créés de manière paramétrique à partir d'une liste de valeurs où sont spécifiées les formes et dimensions possibles de chaque composant.
- Les composants non standards sont des composants qui n'ont pas une forme standard et dont la modélisation paramétrique n'est pas évidente. Cependant, on peut plus facilement définir des portions de ces composants en paramétrant chacune d'elles. Ainsi, la paramétrisation se fera à l'union ou l'intersection des paramètres de ces portions grâce à des opérations booléennes.

3 - Les caractéristiques géométriques sont définies à partir de trois familles de base : cylindre, trou, bloc.

Une liste de paramètres est associée à chaque famille ainsi que ses caractéristiques concernant la forme et la dimension. Chaque objet est représenté sous forme d'une liste de prédicats chaque liste stocke une partie individuelle d'information concernant un objet. Par exemple, une goupille (composant non standard) comme sur la figure 3.6 est constituée d'un bloc (composant standard) et d'un cylindre (composant standard).

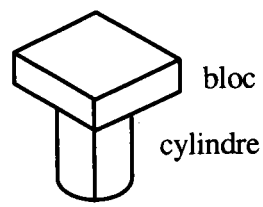


Figure 3.6 Schéma d'une goupille

6.1.2 Les relations entre les données à travers le diagramme des composants

Les relations entre les données sont définies à travers un diagramme de relation, lui-même construit autour de deux relations types : les connexions physiques et les contraintes spatiales. Ces relations existent entre les composants (standards ou non standards).

La connexion physique entre deux composants impose un contact entre ces composants, pour cela il y a trois types de connexions physiques : *visser, contre* et *ajuster.*
Les contraintes spatiales permettent de définir des relations de précédence entre les différents composants pour savoir dans quel ordre les composants vont être assemblés. Pour cela, on dispose de trois types de contrainte spatiale : *est à l'intérieur, est recouvert, est recouvert* et *est à l'intérieur.*

Le diagramme des relations est structuré en utilisant les opérateurs de connexion physique et les opérateurs de contrainte spatiale. Ce diagramme représente la structure interne des états initiaux et finaux de chaque composant.
Par exemple, dans le cas de l'assemblage d'une goupille, d'une plaque trouée et d'une base plate comme sur la figure 3.7.

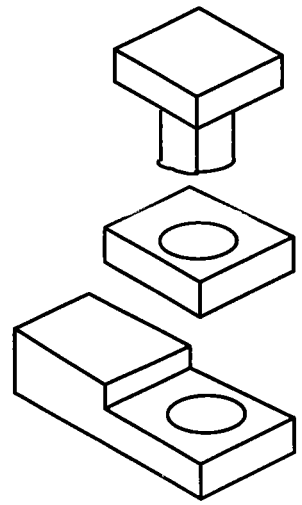


Figure 3.7 Les blocs à assembler

Le diagramme des relations est défini à travers les opérateurs physiques et de connexion. Pour plus de facilité d'écriture les opérateurs ont été abrégés (voir figure 3.8) :
v : visser, c : contre, a : ajuster, i : est à l'intérieur, r : est recouvert, ri : est recouvert et à l'intérieur.

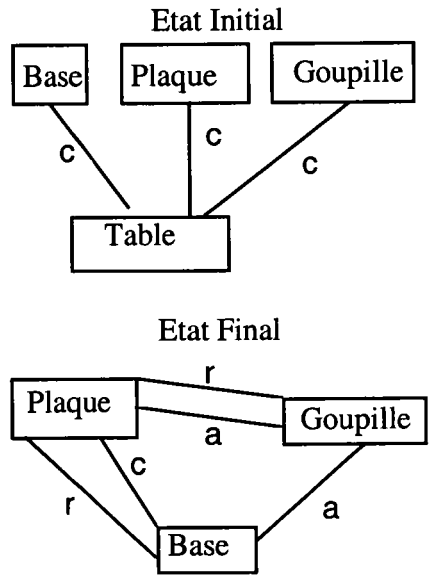


Figure 3.8 Représentation des états

6.1.3 Les règles de compatibilité physique

Pour vérifier la validité de la connexion physique entre deux composants, les caractéristiques géométriques de ces composants sont récupérées et ensuite analysées à travers une liste de règles.
Ainsi, pour valider la connexion physique entre la goupille et la plaque, il faut une compatibilité à la fois géométrique et dimensionnelle. Afin de réaliser cela, il faut appliquer la règle suivante :

si (une caractéristique est un trou plein)
et (l'autre caractéristique est cylindre plein)
et (les diamètres nominaux du trou et du cylindre sont compatibles)
alors (proposer que la connexion physique est valide)
sinon (saisir une autre paire de caractéristiques).

6.2 La base fonctionnelle

La base fonctionnelle à un double rôle :

1 - analyse et valide éventuellement le produit candidat grâce à des procédures spécifiques qui sont adaptées à des profils d'utilisateurs différents ;
2 - stocke les différentes alternatives validées pendant une session de conception d'assemblage et ceci à travers des versions plus ou moins réussies en fonction de la complexité de l'assemblage produit.

Le niveau fonctionnel (voir figure 3.9) permet donc de contribuer à l'évolution du système ainsi qu'à la définition précise du modèle de conception d'assemblage choisi. L'idée générale est de permettre à l'utilisateur non seulement d'apprendre mais aussi d'enrichir le système par de nouvelles connaissances produites au cours d'une session. La figure suivante décrit les différents modules qui interviennent pour mettre en place les mécanismes décrits précédemment.

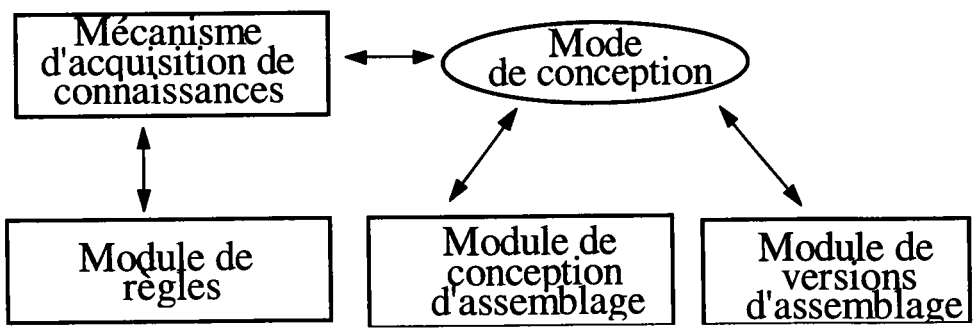


Figure 3.9 La base fonctionnelle

6.2.1 Le module de conception

C'est une forme de bibliothèque où on mémorise et classifie tous les modèles conçus avec succès. L'ensemble peut être consulté ou utilisé à la demande, soit pour une conception ultérieure, soit pour faire l'objet de l'établissement d'un catalogue de collection des objets.

6.2.2 Le module de version

Ce module gère les nombreuses versions produites au cours d'une session de conception en vue de faciliter le retour arrière. En effet, la conception amène à choisir quelquefois différentes alternatives sans savoir *a priori* les résultats que ces choix impliquent.

6.2.3 Le module de règles

Ce module gère les méthodes utilisées par le spécialiste du domaine pour résoudre un problème ainsi que les différentes catégories de contraintes et de règles.

1 - Les méthodes : Elles impliquent des processus de raisonnement ainsi que des méthodes spécifiques de traitement du domaine. Il s'agit en général de calcul de paramètres pour la création ou la modification d'une pièce, il peut s'agir aussi d'exprimer un avis sur un résultat obtenu ou bien de spécifier un besoin qui n'a pas été pris en compte.

2 - Les règles ou contraintes : Ce sont des règles ou des contraintes qui peuvent être d'ordre géométriques, chronologiques, de fabrication, etc., elles peuvent être aussi explicites ou implicites.

6.2.4 Le mode de conception

Les besoins exprimés par l'utilisateur se matérialisent en deux types de conception, cette distinction est introduite dans le but de fournir un mécanisme spécifique et approprié afin de s'adapter à l'utilisateur du système et cela, en fonction de son profil (expert, non expert) ou de la difficulté de conception à faire. On distingue donc deux modes de conception :

- Le mode de conception consultatif :

Ce mode est plus approprié à l'utilisateur non expert, en effet, on privilégie à ce niveau l'apprentissage de l'utilisateur.

- Le mode de conception créatif :

Ce mode prend en compte la capacité de l'expert à pouvoir créer des objets.

Les algorithmes principaux de la base fonctionnelle

- Après chargement de la base de connaissances
- L'utilisateur choisit son mode de conception (Consultatif ou Créatif)

*Consultatif: Appel de la procédure mode de conception consultatif.
- Tant que l'utilisateur ne demande pas l'arrêt
Faire
- Demander à l'utilisateur les paramètres de recherche ;
- Guider l'utilisateur pour préciser ces valeurs ;
- Vérifier la non violation des contraintes ;
-Si violation de contrainte
Alors Retour pour préciser ses contraintes ;
Sinon Proposer des objets candidats ;
Visualiser l'objet choisi ;
Fsi
FinTantque

*Créatif : Appel de la procédure mode de conception créatif.

-Tant que l'utilisateur ne demande pas l'arrêt

Faire

- Demander à l'utilisateur les paramètres de création de l'objet ;
- Guider l'utilisateur pour préciser les valeurs de création à travers le gestionnaire de dialogue ;
- Vérifier la non violation de contrainte ;
- Si violation de contrainte

Alors Retour pour préciser les contraintes ;

Sinon Proposer des objets candidats ;

Visualiser l'objet choisi ;

Fsi

FinTantque

- Stockage des différentes alternatives validées selon la procédure suivante :

Si la conception est complètement réussie

Alors Stockage dans un module de conception réussie

Sinon Stockage dans un module de versions de conception

Fsi

6.3 Le planificateur

Le planificateur (comme nous l'avons déjà souligné) joue le rôle de pivot dans l'ensemble du système d'assistance. En effet, il est le lien entre la base de connaissances et la base fonctionnelle ainsi que le gestionnaire de dialogue. Son objectif est de proposer un ou plusieurs plans d'action lorsqu'un utilisateur fait une requête.

6.3.1 L'architecture du planificateur

Le planificateur est constitué de deux parties : un générateur de plans et un moteur d'inférence. Le générateur représente la partie algorithmique, il se charge de l'identification des opérateurs pouvant atteindre un but donné, il représente les connaissances minimales nécessaires pour construire un plan. Le moteur d'inférence représente la partie qui permet de réduire l'espace de recherche pour choisir le but à poursuivre, l'opérateur à appliquer ou les objets à lier aux variables d'un opérateur. A chaque fois qu'il y a un choix à faire on fait intervenir le moteur d'inférence.

Au départ, le moteur d'inférence ne manipule que quelques règles par défaut qui correspondent simplement à des alternatives disponibles. Les plans ainsi crées sont loin d'être efficaces. Il incombe au moteur d'inférence de modifier les règles d'expertise suivant les leçons tirées des expertises du système. Ainsi au fur et à mesure de l'apprentissage, des règles de contrôle plus informées viennent remplacer les règles par défaut, améliorant à la fois l'efficacité du processus de planification et la qualité des plans générés.

L'architecture proposée s'appuie donc sur une expertise pilotée par un moteur d'inférence d'ordre 1, qui affine et augmente cette expertise en vue d'optimiser le comportement global du système (voir figure 3.10). D'autre part la stratégie de développement de l'arbre de planification est décrite dans la partie B du 6.3.2.

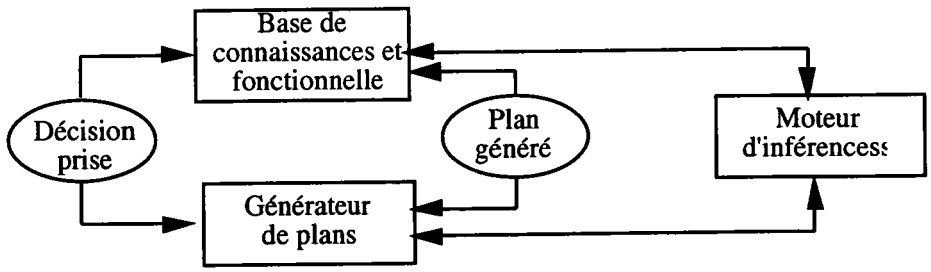


Figure 3.10 Architecture du planificateur

La planification se concrétise par la construction d'un arbre de recherche, chaque nœud représente l'état d'un plan à une étape de son évolution.

6.3.2 L'algorithme général du planificateur

Le fonctionnement du système est fondé sur deux composantes distinctes qui forment son noyau principal.
La première composante est un moteur d'inférence en chaînage arrière d'ordre 1, il représente un ensemble de règles minimales qui permet de réduire l'espace de travail pour choisir le but à poursuivre, l'opérateur à appliquer et les objets à lier aux variables d'un opérateur.

La deuxième composante est un générateur de plan interactif en chaînage arrière d'ordre 1 qui est la partie algorithmique, il se charge de l'identification des opérateurs pouvant atteindre un but, et contient les connaissances minimales pour construire un plan.

La modélisation des actions porte sur les constituants élémentaires, descriptions et actions.
Chaque état (décrit à partir de l'univers de travail) est présenté sous la forme d'une liste de prédicats qui en constitue la base courante sur laquelle les actions vont s'opérer.

Chaque opérateur est défini à partir de :

- l'action à entreprendre,
- la liste de pré-conditions à vérifier avant application de l'opérateur,
- la liste d'ajouts à rajouter dans la base courante,
- la liste de retraits à retirer de la base courante.

Nous allons définir le fonctionnement de chaque composante. Les procédures qui interviennent dans chaque cas pourront être consultées en annexe.

A - Le moteur d'inférence
Il sert en premier lieu à établir les faits qui sont à vérifier à partir de la liste de préconditions qu'on lui fournit.

Deux cas peuvent se présenter dans le processus d'échange entre l'utilisateur et le système : soit la demande est simple, soit la demande est complexe.

a - Cas simple

Le moteur se contente de faire une unification simple des faits proposés avec la base de l'état courant sans inférence sur cette dernière.
C'est la procédure d'unification qui se charge de ce travail.

b - Cas complexe

Lorsque la liste des préconditions comporte des éléments appartenant au module de description de contraintes, le moteur va inférer sur ces règles on considérant les prédicats courants comme la base de faits. On crée donc une arborescence locale des solutions possibles sur la base de l'état courant et le moteur fournit, en retour, la liste des objets qui vérifie l'ensemble des contraintes.

A chaque fois qu'il y a ambiguïté sur l'opérateur à appliquer, le générateur active le moteur d'inférence. De plus, il est important de définir le module de contraintes associé à chaque application.

- à l'état initial le moteur possède une seule règle : choisir la première alternative,
- aux états suivants le moteur stocke, à chaque fois, le choix qui n'a pas engendré un échec ou un cycle infini,
- à un état courant, le moteur charge les règles associées à l'opération en cours,
- après inférence, il fait le choix de la meilleure ou des meilleures alternatives (procédure d'unification),
- au fur et à mesure qu'on charge de nouvelles règles, on crée des mécanismes d'apprentissage du planificateur qui réduisent l'espace de travail engendré.

Exemple de l'algorithme

La gestion se fait à travers une pile de buts à atteindre :

Début
Tant que PILEBUTS <> nulle
  Faire **Tant que** ARBRE <> nulle
    Faire Récupérer dans PILEBUTS une REGLE à partir de ARBRE ;
      Appliquer la Règle choisie par la procédure Règle ;
      Procédure Règle ;
      Passer à la règle suivante ;
      Mettre à jour l'arbre de recherche ARBRE ;
      Mettre à jour la pile de buts PILEBUTS ;
    Fait
  Fait
Fin

B - Le générateur d'action

Deux aspects interviennent : un niveau formel, c'est-à-dire un travail de planification classique tel qu'il est décrit dans le paragraphe 5 du premier chapitre 1 et, à un niveau interactif de coopération Homme-Machine qui permet de diriger l'utilisateur et de lui proposer des solutions intermédiaires.

La stratégie du planificateur est en largeur d'abord. Il choisit donc toujours le nœud le plus à gauche à développer, chaque nœud représentant l'état du plan à une étape de son évolution.
La stratégie de développement en largeur d'abord, consiste donc à décomposer les fils de tous les successeurs d'un problème avant de décomposer ses «petits-fils» (figure 3.11), contrairement à la stratégie de développement en profondeur d'abord, qui consiste à décomposer les «petits-fils» d'un problème par appel à la procédure de planification avant de décomposer les fils de son successeur (figure 3.12). Le choix de cette stratégie est dicté par un double objectif.

Le premier objectif est de permettre d'arriver à proposer un chemin dans tous les cas possibles. Le deuxième objectif est une conséquence du premier. En effet, le nombre de possibilités de constructions et d'assemblages des pièces mécaniques n'est pas illimité, et dépend en grande partie de l'expertise de l'expert. Il reste donc tout à fait raisonnable au niveau du développement de l'arbre de recherche.

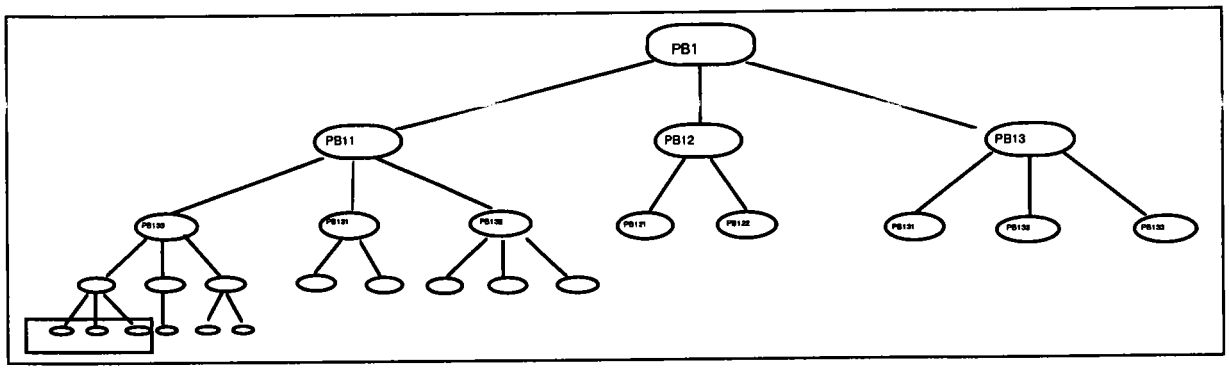


Figure 3.11 Algorithme de résolution en profondeur d'abord.

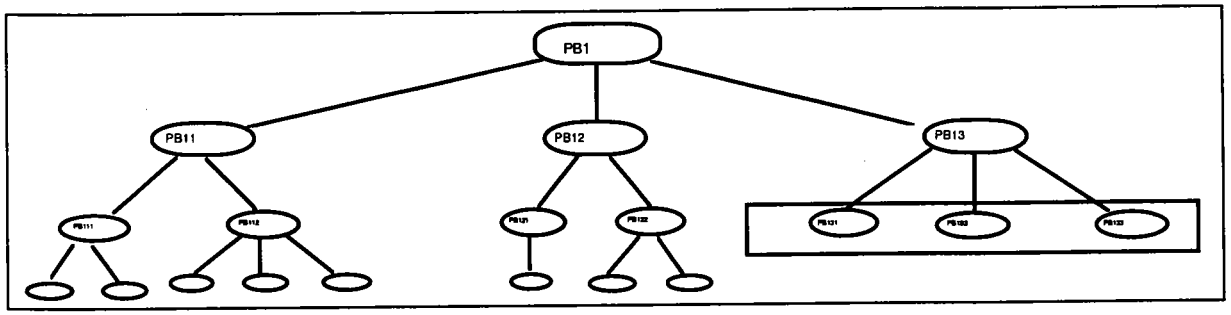


Figure 3.12 Algorithme de résolution en largeur d'abord

Les étapes générales du générateur d'actions et une partie de son algorithme sont décrites dans ce qui suit :

- Au démarrage, le générateur initialise l'arbre de recherche. L'arbre d'action développé contient trois types de nœuds, un nœud initial, un nœud but, un nœud action. Le nœud but est formé d'une conjonction de prédicats définissant le but à atteindre par le générateur. Le générateur crée la racine qui devient, d'office, le nœud courant, à ce stade la pile de but et d'actions est réduite à un seul élément. C'est la procédure d'initialisation.

- Pour développer un nœud, le générateur examine le sommet de la pile. S'il s'agit d'un but, il repère les opérateurs susceptibles de l'atteindre. Pour cela il teste pour chaque opérateur si la liste de préconditions est satisfaite. Cette liste de préconditions est renvoyée sous forme

de pile de buts au moteur d'inférence qui renvoie à son tour selon le cas (simple ou complexe), une liste de prédicats unifiés. Ce sont dans l'ordre les procédures suivantes :
- Procédure choix opérateur,
- Procédure d'unification,
- Procédure d'inférence sur les contraintes.

- Pour éviter le *frame problem* (conservation des états intermédiaires) le générateur ne garde en mémoire que la liste d'ajout et de retrait du nœud à développer. C'est la procédure conservation des états intermédiaires ;

- Dans le cas d'une action, ces préconditions sont vérifiées, s'il existe une précondition non satisfaite dans la situation courante elle est empilée comme un but à atteindre, sinon l'action est dépilée et intégrée dans le plan. Ce sont les procédures suivantes :
  - Procédure vérifie,
  - Procédure ajout.

- Avant d'engendrer un successeur éventuel au nœud choisi, le générateur commence par tester si ce nœud correspond à l'état final souhaité, auquel cas, il fait le chemin inverse de celui parcouru en retrouvant les opérateurs qui ont abouti à cet état. Dans le cas contraire, on développe ce nœud de façon récursive et selon le même principe que pour l'expansion d'un nœud. Ce sont les procédures suivantes :
  - Procédure compare,
  - Procédure chemin.

- A chaque fois qu'un opérateur est applicable, le système vérifie si cet opérateur n'est pas le dual du précédent pour éviter de s'engager d'un cycle.
  - Procédure dual.

Exemple d'une partie de l'algorithme :

L'algorithme
Debut
Tant-que Base <> nulle
Début
Faire
Appliquer une fonction qui étant donné un état de la base donne l'ensemble des opérateurs applicables et les met dans LISTE-OP (listes des opérateurs).
Procédure Ens-opérateurs;
Appel du Moteur d'inférence;
Tant-que LISTE-OP <> nulle ou PILEBUTS <> nulle
Faire
Appliquer une fonction qui permet de choisir un opérateur OP en vue de l'appliquer et le supprime de la liste des opérateurs applicables.
Procédure Choix-opérateurs;
Appliquer l'opérateur choisi ;
Procédure Applic-op;
Ajouter un nœud (pere etat nil) dans l'arbre de résolution et mettre à jour les liens
Procédure Ajouter-nœud;

Comparer l'état final et l'état de la base courante.
Procédure Compare;
Si Succès
Alors Retrouver le chemin de la liste des opérateurs qui ont produit l'état final.
Procédure Chemin;
Afficher le résultat;
Sinon Recommencer le processus;
Fsi Fait Fait
Fin

6.3.3 L'acquisition de connaissances

L'acquisition de connaissances se situe sur deux niveaux :

- Au niveau de la base fonctionnelle : il s'agit d'adjoindre plusieurs méthodes techniques d'acquisition de connaissances. Par exemple dans le cas ou le travail de conception dépend de l'intuition d'un spécialiste, on associe des techniques d'apprentissage à partir d'exemples techniques divers qui permettent de mettre à jour la base fonctionnelle en vue de s'adapter aux besoins nouveaux.

- Au niveau du planificateur : dans le cycle planification/apprentissage, les mécanismes d'apprentissage interviennent *a posteriori* par rapport à un point de choix donné, c'est-à-dire que toutes les alternatives ont été essayées, et l'on sait à quoi a abouti chacune d'elles, soit à un échec, soit à l'achèvement d'un plan. On sait donc *a posteriori* quel est le choix optimal, et la tâche des mécanismes d'apprentissage consiste à créer des règles pour imposer celui-ci dans des situations analogues.

6.4 Le gestionnaire de dialogue

Le gestionnaire de dialogue contribue à rapprocher les besoins de l'utilisateur avec les possibilités du système en lui permettant de mieux formuler sa demande et ceci, grâce à des mécanismes de négociation adaptés à chaque profil d'utilisateur. Il représente l'interface entre l'application et l'utilisateur. C'est donc l'intermédiaire incontournable entre le système et l'utilisateur.

Le gestionnaire doit assurer les fonctions suivantes :
- aider l'utilisateur à établir sa demande,
- compléter les informations manquantes pour préciser le but,
- relancer la procédure dans le cas d'insatisfaction de l'utilisateur.

6.4.1 La procédure de dialogue

On peut résumer la procédure comme suit :

- l'utilisateur établit sa demande,
- le gestionnaire fait préciser le but à atteindre lorsqu'il y a ambiguïté,
- le générateur activé devra selon les cas avoir un complément d'informations sous peine de rejet de la demande,
- le gestionnaire soumet une première solution à l'utilisateur, deux cas se présentent :

a - L'utilisateur approuve la solution, auquel cas le système garde en mémoire la trace du chemin parcouru et propose à l'utilisateur de continuer, avec pour nouvel état initial, la solution approuvée.

b - L'utilisateur désapprouve la solution, auquel cas le système continue à développer l'arbre d'actions, jusqu'à trouver une autre solution.

Il se peut, par ailleurs, que l'utilisateur désapprouve toutes les solutions qui lui sont proposées, pour cette raison il est nécessaire de garder trace de toutes les solutions non approuvées.
Le système les propose une seconde fois à l'utilisateur, lui offrant ainsi une deuxième occasion de faire un choix, on peut parler de négociation à ce moment là (voir figure 3.13).

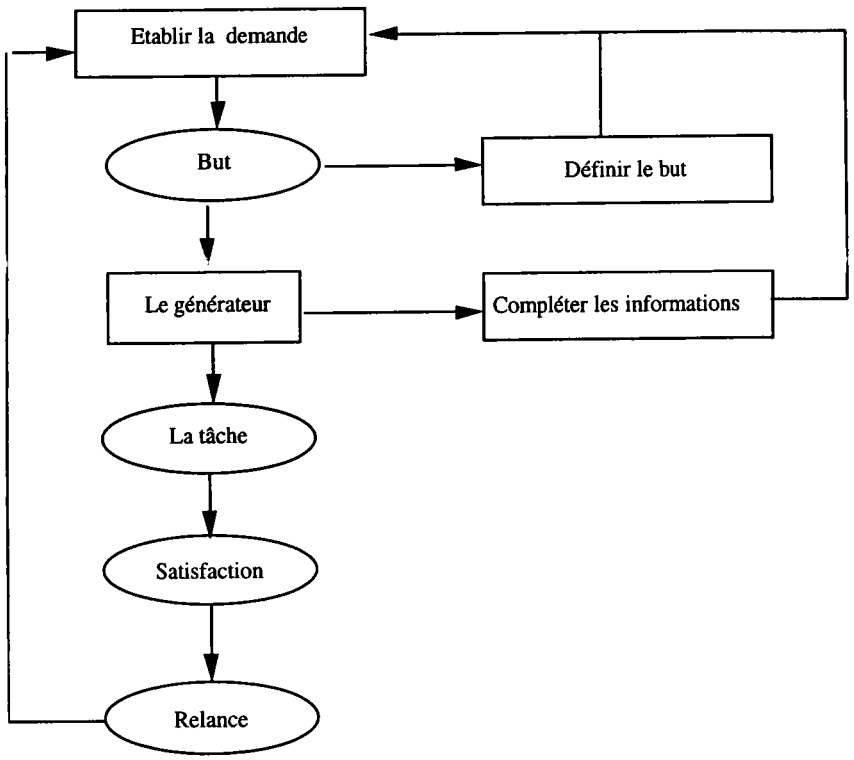


Figure 3.13 Les enchaînements dans la procédure de dialogue

6.4.2 L'historique du dialogue

En gardant à chaque fois la trace du chemin choisi par l'utilisateur, le gestionnaire de dialogue constitue, de fait, un historique du dialogue comme indiqué sur la figure 3.14 .

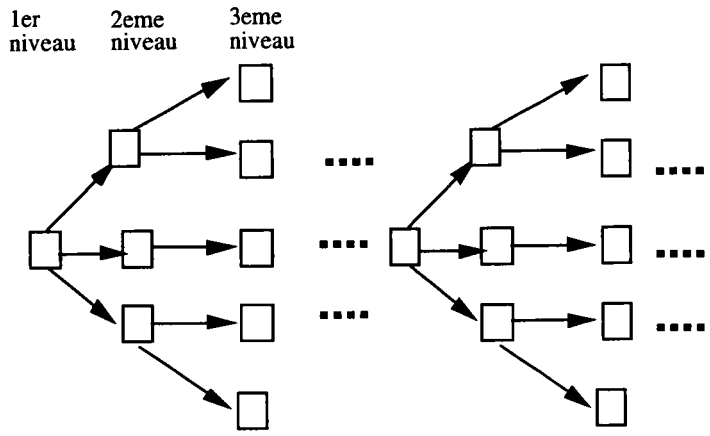


Figure 3.14 La représentation de l'historique du dialogue

Ceci permet, dans un premier objectif, de gérer cet historique en fournissant à l'utilisateur :

- dans le cas d'une répétition, le chemin tel qu'il lui a été donné au préalable,
- dans le cas d'une contestation (l'utilisateur pouvant contester toutes les solutions qui lui sont proposées et ne pas accepter de négocier). Le système accède à l'historique du dialogue dans un niveau supérieur équivalent à une unité pour chaque retour arrière,
- dans le cas de non satisfaction, une procédure de relance est activée.

7 Conclusion

Dans ce chapitre nous avons essayé d'une part, de montrer qu'un modèle d'assistance utilisant la planification interactive comme support, à la fois, à l'assemblage d'une pièce mécanique et au dialogue avec l'utilisateur, est possible. D'autre part, la nécessité d'un environnement adéquat à travers des modules liés à la base de connaissance et la base fonctionnelle est indispensable pour pouvoir utiliser efficacement les résultats fournis par le planificateur. Le but de l'architecture choisie s'articule autour de trois idées.

- *L'adaptation au profil de l'utilisateur*

1 - Pour un utilisateur non expert, le but est de pouvoir lui proposer des mécanismes de reprise en nombre important, pour cela, d'une part, des mécanismes de négociation à travers la gestion de l'historique du dialogue sont mis à contribution pour le satisfaire, d'autre part, la gestion des versions déjà produites lui permet d'avoir un outil de référence pour la construction et l'assemblage.
2 - Pour un utilisateur expert, le but est de pouvoir lui proposer des alternatives plus riches et plus nombreuses, pour cela on dispose de trois outils :

- L'outil "planification interactive" qui génère des séquences opératoires plus nombreuses si l'utilisateur le demande ;
- L'outil "gestion de dialogue" qui diminue le nombre de reprise ;
- L'outil "base fonctionnelle" qui, à travers la gestion des versions produites, fournit une procédure de consultation rapide, en particulier pour les produits complexes.

- *La modularité*

Un autre aspect important du choix de cette architecture est la possibilité de dissocier le niveau algorithmique, c'est-à-dire, la partie planification, du niveau connaissances et données (base fonctionnelle + base de connaissances). En effet, cela permet à l'utilisateur de mieux maîtriser sa tâche. De plus, la modularité du système d'aide rend possible la modification d'un composant sans en altérer les autres. En effet, lorsque l'utilisateur intervient sur un objet et selon le choix qu'il impose (consultation ou création) le système prend en charge la mise à jour de l'opération en utilisant les modules adéquats.

- *L'interactivité*

L'activité de l'utilisateur est implicitement guidée lors du processus de conception en organisant la production du système d'aide en étapes significatives, clairement identifiables. En effet, l'utilisation de la planification interactive permet de construire un historique du dialogue et, par conséquent, d'identifier les choix probables lors de la conception.
L'enchaînement logique des étapes constituées permet à l'utilisateur de se représenter son activité. La ponctuation de chaque étape par un ensemble d'objets qui en résulte jalonne son chemin d'objectifs partiels atteints et à atteindre et, contribue à donner une vision précise des objets et leurs possibles réutilisations.

Chapitre 4

Mise en œuvre du système d'aide

1 Introduction

Nous avons montré dans le chapitre 3, que l'intérêt que présente un système d'aide (dédié à la C.A.O) est d'avoir des fonctionnalités d'assistance à l'utilisateur permettant à ce dernier de choisir la meilleure alternative de conception. Dans ce même chapitre, nous avons décrit l'architecture générale du système d'aide et ses différentes fonctions. Il reste à valider les orientations choisies à travers des exemples concrets.

Dans la première partie de ce chapitre nous introduisons et décrivons deux notions déjà esquissée dans le début du chapitre 3, celle de multimodèle et du couplage système d'aide-modèles géométriques. Il s'agit d'une part, de définir plus précisément ces deux notions et leurs contextes dans notre étude et, d'autre part, de montrer (à travers des exemples) l'intérêt de proposer ces notions dans notre système d'aide.

Dans la deuxième partie de ce chapitre, nous montrons à travers le cas de la construction d'un clapet et d'un alternateur les différentes étapes produites lors de cette construction et qui passent par la gestion du dialogue, le choix du mode de conception et l'utilisation du planificateur comme outil de conception.

Dans la troisième partie nous proposons un modèle d'interface Homme-Machine, compatible avec les orientations choisies dans l'architecture globale du système d'aide.

2 Notion de mutimodèles

2.1 Multiplicité des modèles

D'après la définition de Minsky [Minsky 86], on appelle modèle d'un objet de représentation, un modèle qui permet à un observateur de répondre aux questions qu'il se pose sur cet objet.
Il résulte de cette définition qu'un dessinateur cherchant à concevoir une pièce mécanique sera sensible au problème décrivant des éléments du mouvement, alors que le thermicien étudiant la même pièce ignorera ces données. De ce fait, un objet technique modélisé comporte une multitude de vues.

2.2 Les modèles géométriques et leurs caractéristiques

Le modèle géométrique est constitué d'identités géométriques et d'attributs. Les premières vont du simple point de vue géométrique à la notion topologique. Ces entités peuvent être aussi regroupées (voir chapitre 3).

La première caractéristique des entités constituant un modèle, par exemple de type B-Rep, est la structuration hiérarchique des entités qui constituent le modèle, ainsi, quelque soit le niveau de détail modélisé, le grand handicap des modèles géométriques est la faible autonomie de leurs constituants.

2.2.1 Le cas du modèle B-Rep

- Au niveau des actions

Un nœud ne peut être détruit, ni être déplacé, sans vérifier que d'une part, le maillage reste consistant et, d'autre part, qu'il ne soit pas de la même matière que l'objet (liaison entre modèles d'éléments finis et modèles géométriques).

- Au niveau des données de représentation sémantique

Leur représentation appartient à une structure très fortement relationnelle qui définit la topologie d'une coque fermée.

- Au niveau des données de représentation graphique

Le fait qu'il soit vu ou non dépend de sa position respective par rapport à d'autres parties de l'objet, s'il est associé à une côte, la côte doit suivre son déplacement.

2.2.2 Le cas du modèle CSG

- Au niveau des actions

Différentes opérations booléennes sont mises en œuvre pour réaliser un objet.

- Au niveau des données de représentation sémantique :

Le modèle CSG possède un arbre de construction qui lui permet de conserver l'historique de construction d'un objet.

- Au niveau des données de représentation graphique :

La structure de ce modèle est un arbre binaire dont les feuilles représentent des objets primitifs (cylindre, sphère, parallélépipède).

2.3 Conséquences sur l'architecture de l'application

L'existence de la multiplicité des vues du modèle n'est pas sans conséquences sur l'architecture du système d'aide. En effet, l'existence de données communes manipulées par chacun des points de vue sur l'objet (crée, modifié) rend nécessaire la définition d'un module commun regroupant et assumant leur gestion et leur cohérence. En revanche, les algorithmes qui permettent de les exploiter ou de les créer sont définis de manière externe sous forme d'entités (voir figure 4.1)

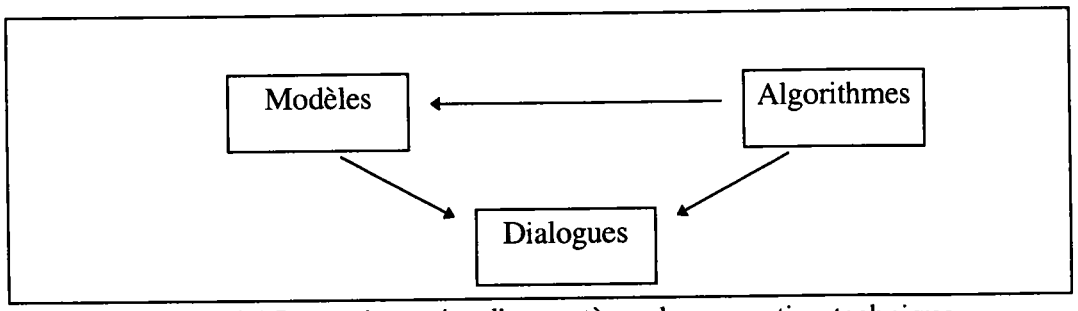


Figure 4.1 Les trois parties d'un système de conception technique

De ce fait, la nécessité de définir une image à vue multiple à travers notre système se trouve confortée. En effet, l'application étant indépendante, l'idée esquissée dans le chapitre3-figure1, est de proposer sur différents modèle type (B-Rep, CSG) les possibilités de conception d'un objet.

3 Les primitives associées au modèle géométrique

3.1 Exemple du cube troué

Le schéma de la représentation d'un produit fait l'objet de travaux actifs. Pour notre part, notre choix s'est porté sur une représentation fondée sur la logique des prédicats du premier ordre à cause de son universalité et de sa flexibilité. Un autre élément qui conditionne ce choix, est le fait que la plupart des informations sur les données, peuvent être traitées comme des contraintes imposées sur les éléments du modèle B-Rep (arêtes, arcs, faces) : par exemple les contraintes de dimension, de distance, d'angle entre deux faces.

Les figures sont décrites sous formes de prédicats où chaque nom spécifie le type de dimension (Distance ou Diamètre ou Surface), avec ci-dessous (figure 4.2 et figure 4.3) un aperçu de cette représentation dans le cas d'un cube troué.
Fi représente les faces du cube contenant et fi représente les faces du cube contenu.

Distance(F1, f1, 20) : représente la distance entre la surface F1 et f1 dans la figure 4.2.
Distance(F2, f2, 10) : représente la distance entre la surface F2 et f2.
Distance(F3, f3, 10) : représente la distance entre la surface F3 et f3.
Distance(F4, F4, 10): représente la distance entre la surface F4 et f4.
Surface(f1, 10) : représente la surface de la face f1, etc..

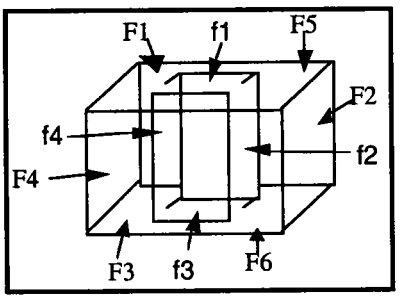


Figure 4.2 Le cube troué

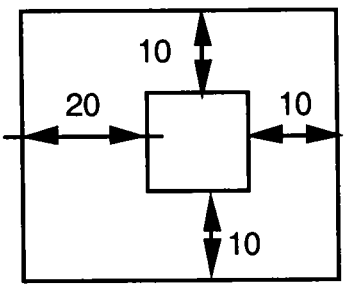


Figure 4.3 Le cube et sa vue de dessus

Il s'agit dans cette perspective de réaliser une traduction d'une formule en tenant compte des opérations (+) et (-) respectivement opérations d'ajout et de retrait dans les modèles B-Rep et CSG à partir des données décrites ci-dessus. La flèche ---> représente le résultat des actions précédentes.

3.2 Cas du modèle B-Rep

Début
1 - création du cube1
arête1, arête2, arête3, arête4---> création-contour1-->création-face1.
arête5, arête6, arête7, arête8---> création-contour2-->création-face2.
arête9, arête10, arête11, arête12---> création-contour3-->création-face3.
arête13, arête14, arête15, arête16---> création-contour4-->création-face4
....
--->création-volume1.

2 - création cube2
arête1, arête2, arête3, arête4---> création-contour1-->création-face1.
arête5, arête6, arête7, arête8---> création-contour2-->création-face2.
arête9, arête10, arête11, arête12---> création-contour3-->création-face3.
arête13, arête14, arête15, arête16---> création-contour4-->création-face4
...
--->création-volume2.

3 - opération d'ajout ou de retrait
création-face-dessus
création-face-dessous

4 - opérations de commandes
visualisations-fines : arêtes, contour, faces
visualisations-larges : cube1, cube12, retrait ou ajout
choix des opérations + ou - choix de la visualisation fine ou large retour.
Fin.

3.3 Cas du modèle CSG

Début
1 - création1 ---> volume1
2 - création1---> volume2
3 - opération d'ajout ou de retrait.
Fusion volume1, volume2.
4 - opération de commande :
visualisations-fines : facei, facej.
Visualisations-larges : volume1, volume2.
Fin.

4 L'assistance dans le cas de la construction et l'assemblage

4.1 Introduction

Comme nous l'avons déjà vu dans le chapitre 3, le couplage du système d'aide avec le modeleur doit permettre l'adaptabilité du système par rapport à l'application selon les critères suivants :
1 - Le système de C.A.O possède sa propre interface utilisateur et le système d'aide doit le respecter, il s'agit alors d'une greffe.
2 - Chaque partie (application et système d'aide) utilise sa propre interface utilisateur, il s'agit alors d'une cohabitation amicale.
3 - L'interface du système d'aide est utilisée par l'application pour toutes les interactions.

Dans les trois cas, la structure de représentation de l'adaptabilité du système d'aide est la suivante :

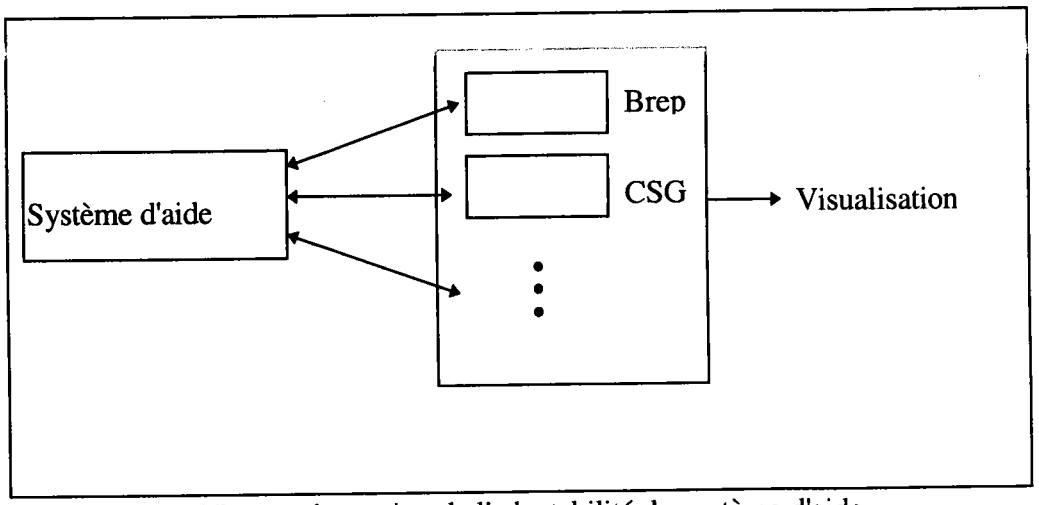


Figure 4.4 La représentation de l'adaptabilité du système d'aide

L'architecture proposée s'adapte à des exemples de conception d'objets dans le cadre de système d'assemblage. Nous proposons dans cette partie de montrer sur des exemples concrets la validation de notre approche.

4.2 Etudes à travers des exemples de conception de pièces mécaniques

4.2.1 Introduction

Le système d'aide comme nous l'avons déjà décrit a pour but de proposer des alternatives de conception mais aussi de fournir éventuellement des informations relatives à la conception d'une pièce mécanique. Dans la plupart des cas, l'aide sémantique qui accompagne les systèmes d'aide (appelée aussi conseil stratégique) a pour rôle de fournir des informations relatives à la sémantique des commandes.

Il s'agit pour l'essentiel de suggérer à l'utilisateur une ou plusieurs démarches de conception correspondant à des plans d'actions issus de l'expertise individuelle. En définitive, le conseil dit « intelligent » a pour but de contribuer à rendre cette expertise plus collective.

En C.A.O, il s'agit d'aider l'utilisateur à concevoir son objet en utilisant « au mieux » le modeleur, c'est à dire en respectant sa philosophie. De façon plus précise, il s'agit de suggérer les enchaînements de commandes les plus appropriées des différents points de vue.

- Manipulations ;
- Travaux futurs (fabrication) ;
- Mise à jour, c'est-à-dire de la maintenance.

A titre d'exemple, nous allons montrer de deux manières différentes comment procéder à la conception de la pièce suivante : un clapet de compresseur (figure 4.5) et un des modèles d'assemblage possibles (figure 4.6).

4.2.2 Le cas du clapet

La nomenclature des composants constituant le clapet du compresseur est la suivante :

1 - Vis,
2 - Corps de valve,
3 - Joint,
4 - Rivet,
5 - Limiteur de course,
6 - Membrane de course,
7 -Support de course,
8 - Membrane d'aspiration.

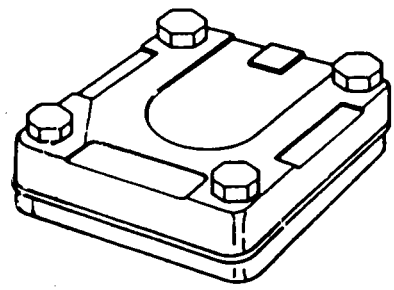


Figure 4.5 Le clapet à construire de deux manières

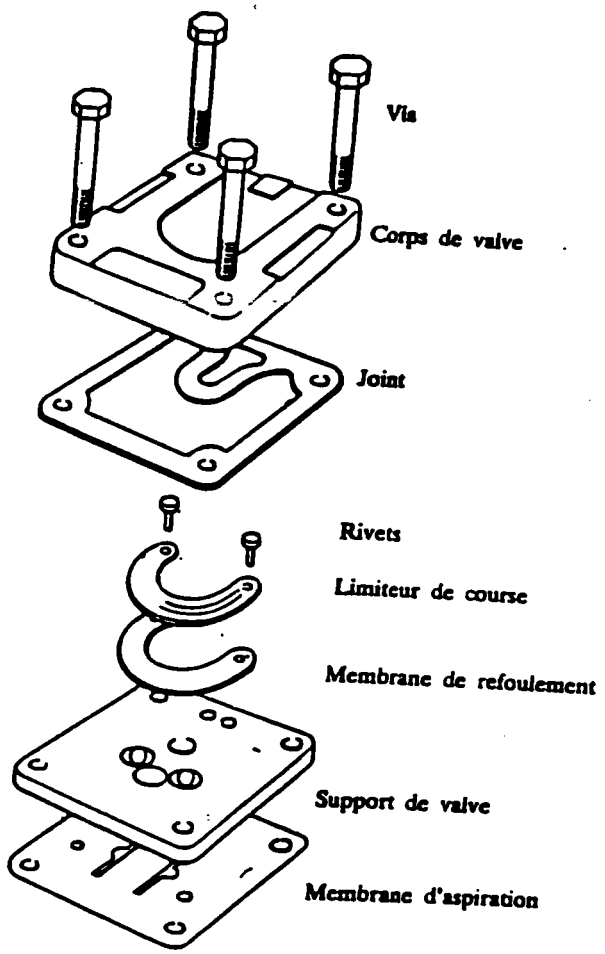


Figure 4.6 Un modèle d'assemblage possible

Première méthode d'assemblage du clapet

Le principe général est le suivant. On commence par mettre en place les disques sur les contours pour réaliser une surface trouée. On obtient le volume V1 par la généralisation d'un volume V à partir de la surface trouée. La deuxième partie de cette méthode consiste à générer un autre volume V2 de la même façon que le premier et ainsi de suite pour les volumes Vi.

On obtient la pièce désirée par fusion des différents volumes générés. Enfin, l'assemblage des pièces se fait au fur et à mesure de la génération des volumes.

Les différents enchaînements de cette méthode sont résumés par la figure 4.7 par l'arbre suivant :

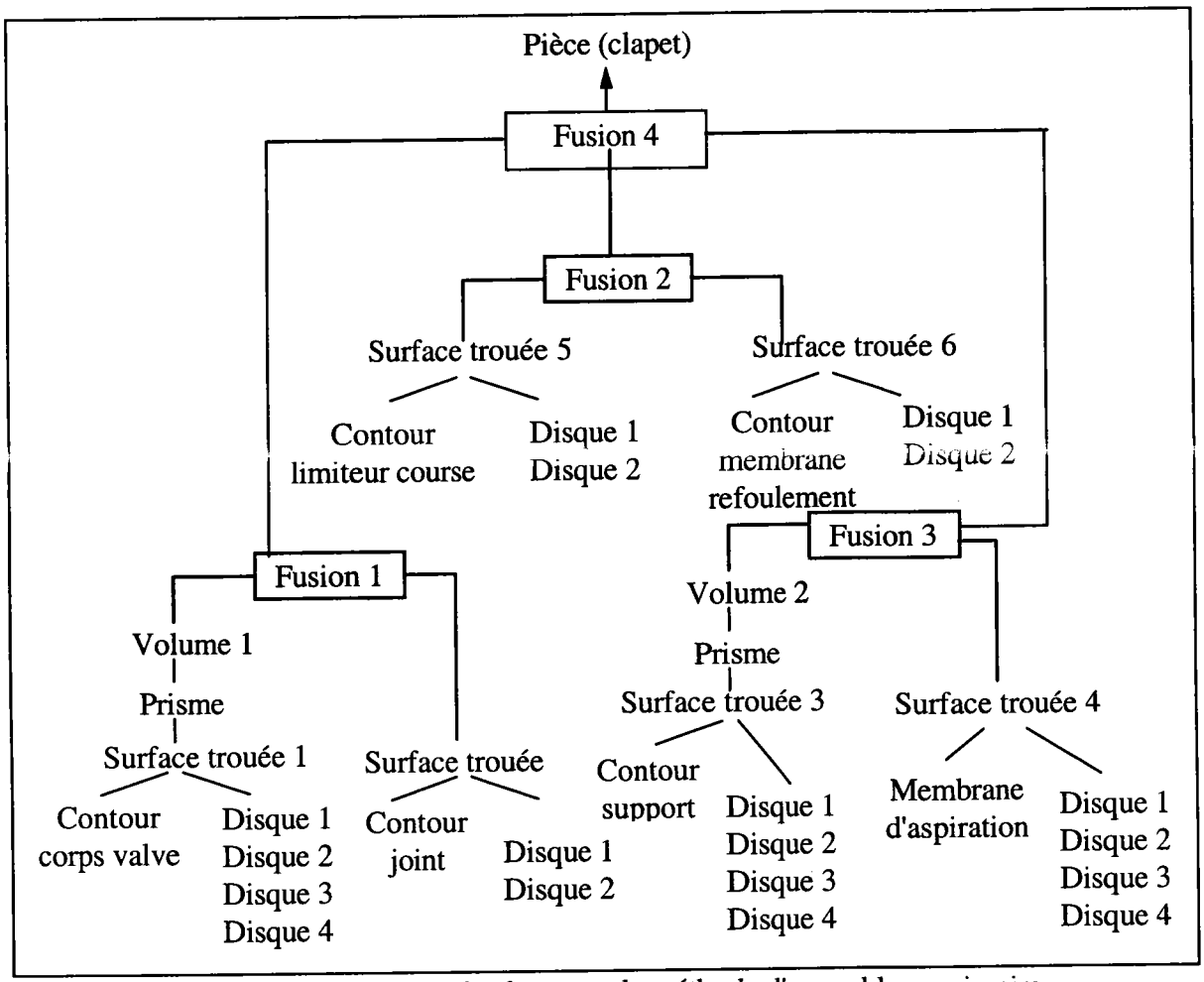


Figure 4.7 Arbre de conception du clapet par la méthode d'assemblage primaire

Deuxième méthode d'assemblage du clapet

Le principe général est le suivant : Il s'agit de générer les volumes à partir du contour pour réaliser les volumes, puis de mettre en place des entités outils sur ces volumes pour y effectuer des coupures. On obtient un nouveau volume. D'autre part, on réalise d'autres volumes à l'aide des cylindres outils. Parvenu à ce stade, il suffit de fusionner les volumes pour obtenir la pièce avant assemblage.

Les différents enchaînements de cette deuxième méthode sont résumés par la figure 4.8 par l'arborescence suivante.

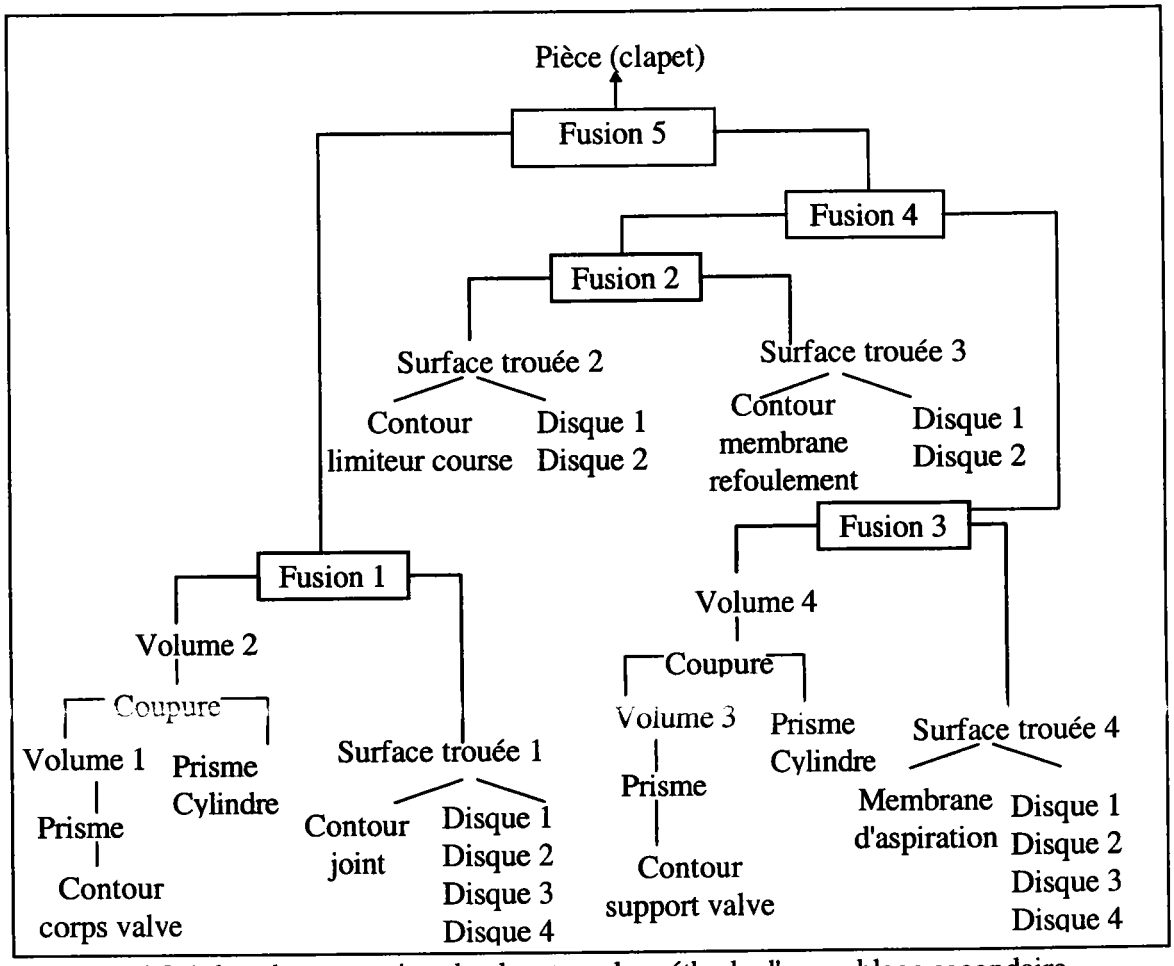


Figure 4.8 Arbre de conception du clapet par la méthode d'assemblage secondaire

Pour arriver à suggérer les enchaînements des commandes les plus appropriées, le système doit intervenir avant le début de la conception de l'objet par l'utilisateur. En effet, pour suggérer à l'utilisateur une bonne démarche de conception, c'est-à-dire une bonne séquence de commandes, il faut le faire avant qu'il ne commence à bâtir l'objet. Il s'agit donc de déterminer les intentions de l'utilisateur et de lui proposer une démarche de construction en fonction de ses intentions.

Dans cette perspective une démarche à deux niveaux nous semble pertinente :
1 - Une classification des intentions des utilisateurs suivant leurs profils.
2 - Une expertise propre à chaque domaine d'application, dans ce dernier cas on récupère l'expertise au fur et à mesure de sa constitution par des utilisateurs du domaine.
Une fois l'expertise réalisée, on associe à chaque élément (objet ou partie d'un objet) une ou plusieurs séquences.

En extension, l'exploitation du système d'aide consiste non seulement en l'utilisation de méthodes prédéfinies mais aussi, en la création de nouvelles méthodes, ce qui correspond à la récupération de l'expertise propre au domaine d'utilisation.

4.2.3 Le cas de l'alternateur

Le deuxième exemple se présente sous la forme de la construction d'un alternateur avec transfert d'un sous-assemblage constitué d'une sortie d'alternateur. La nomenclature des composants constituant l'alternateur est la suivante :

- Vis,
- Ecrou,
- Rondelle grower,
- poulie,
- Ventilateur,
- Entretoise,
- Sous-assemblage:
    - Vis,
    - Cage de roulement,
    - Roulement,
    - Capot de sortie d'alternateur.
- Entretoise,
- Rotor,
- Capot de l'alternateur de roulement.

La figure suivante décrit cette configuration.

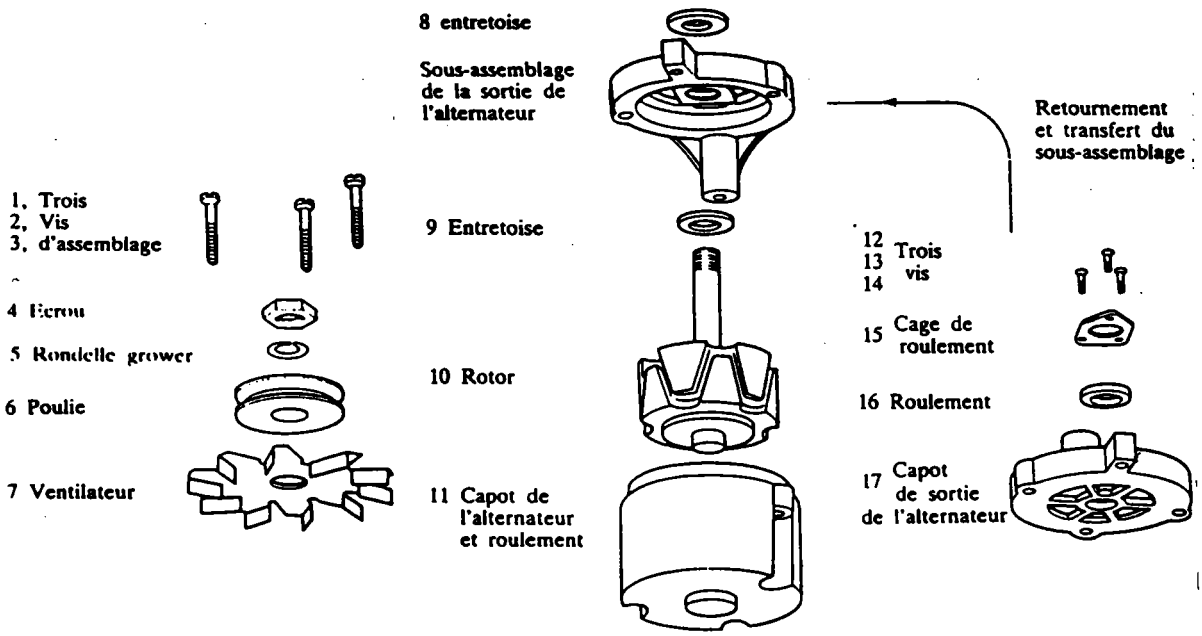


Figure 4.9 : L'alternateur

Les deux méthodes de construction possibles sont bien sûr un peu plus complexes que dans le cas du clapet, mais la méthode globale de construction est la même que dans le 4.2.1.

Première méthode d'assemblage de l'alternateur

Les différents enchaînements de cette première méthode sont résumés par la figure (sous forme d'arborescence) suivante.

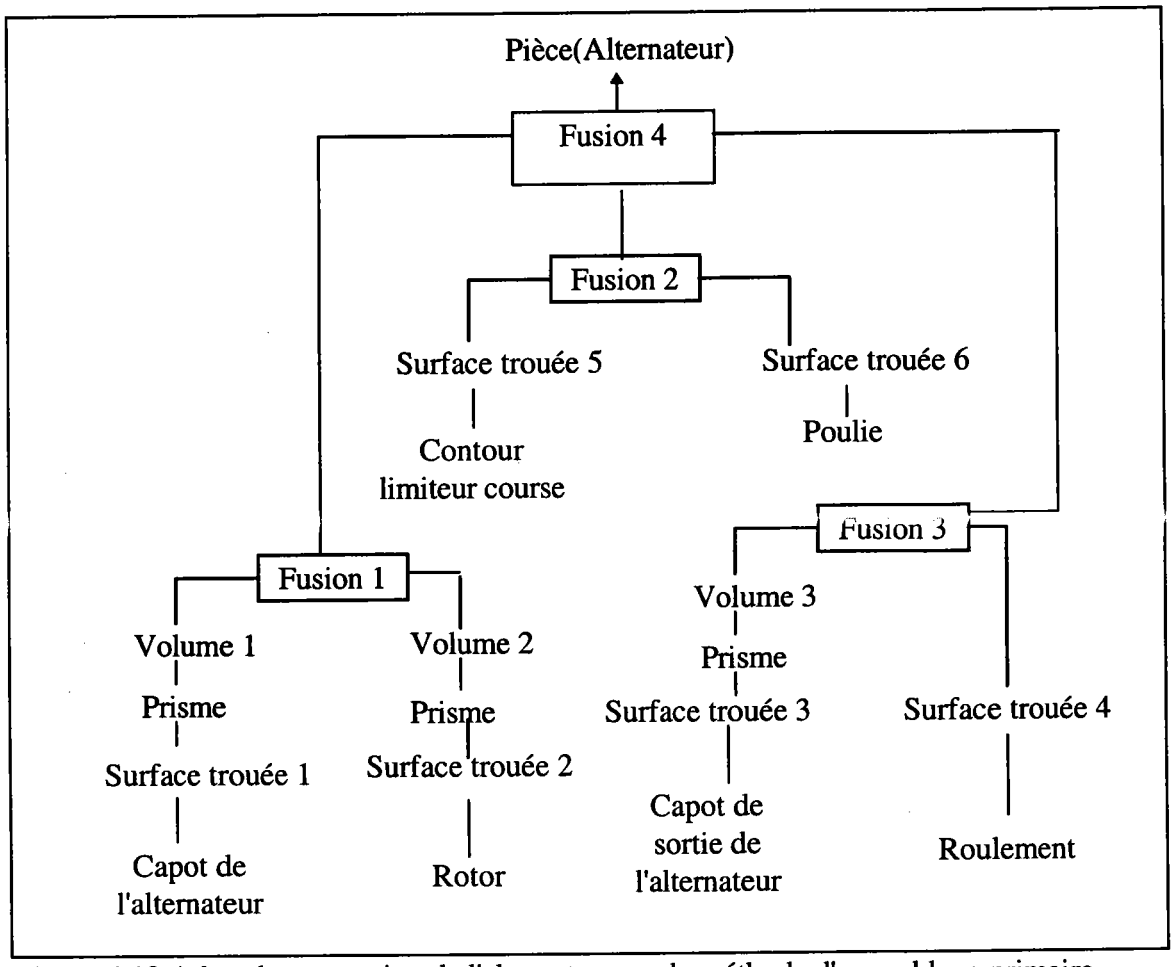


Figure 4.10 Arbre de conception de l'alternateur par la méthode d'assemblage primaire

Deuxième méthode d'assemblage de l'alternateur

Les différents enchaînements de cette méthode sont résumés par la figure 4.11 par l'arborescence suivante.

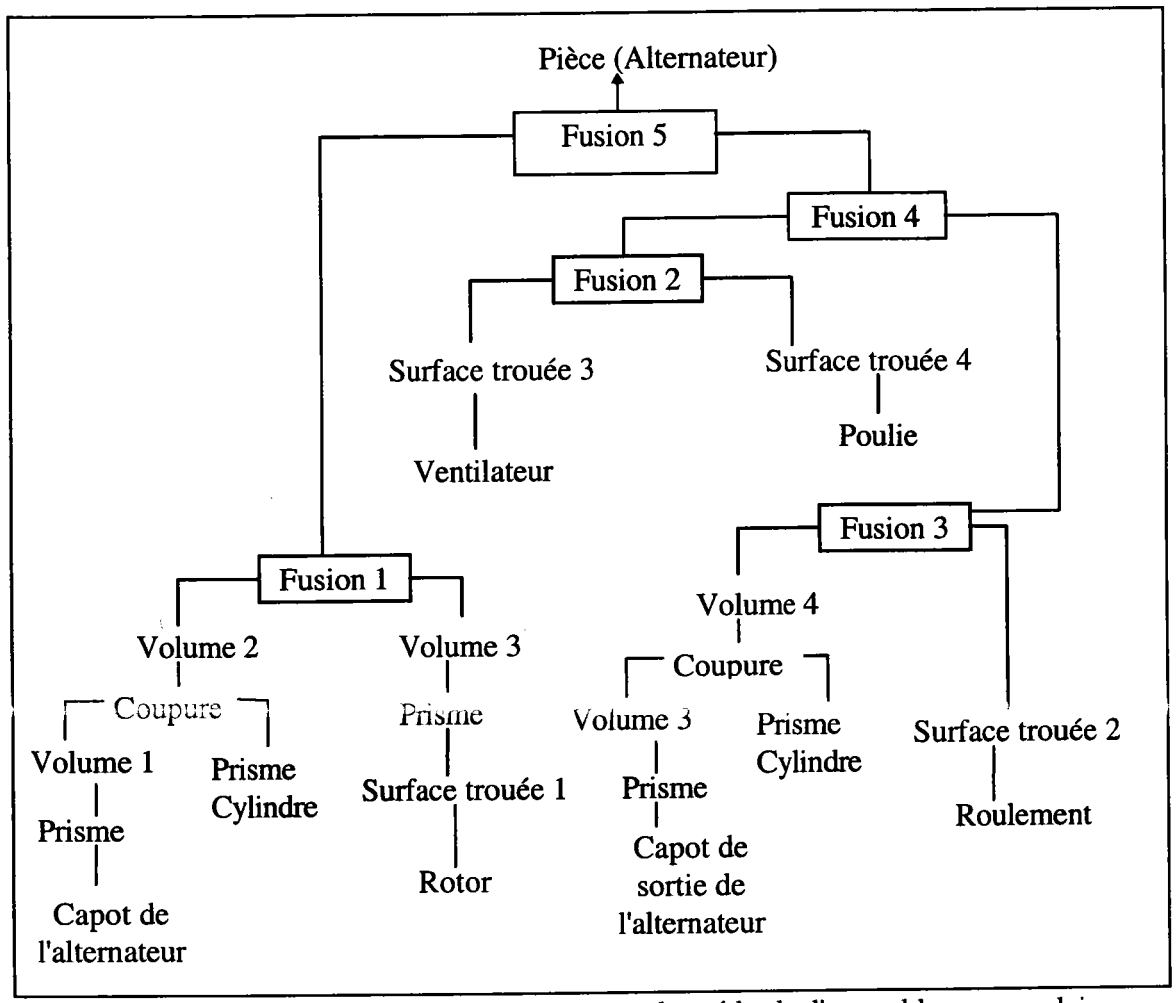


Figure 4.11 Arbre de conception de l'alternateur par la méthode d'assemblage secondaire

Nous nous trouvons ici devant un cas où la compétence de l'utilisateur de l'application est importante en effet, la possibilité de construire une sous-pièce qui devrait s'intégrer dans un ensemble plus complet pose le problème d'une hiérarchie d'assemblage précise et incontournable, ce qui implique un schéma de solution unique dans cette perspective.

Néanmoins, la distinction pourrait se faire au niveau de la construction de chaque pièce, ainsi qu'au niveau de la volonté de l'utilisateur(expert, non expert) d'aller plus loin lorsque le sous-assemblage primaire est construit.

Comme apport supplémentaire dans le système d'aide on peut intégrer des niveaux complémentaires qui sont décrits dans la suite de ce chapitre.

4.3 Les niveaux dans le système d'aide

On peut distinguer deux niveaux complémentaires dans le fonctionnement d'un système d'aide : un niveau fonctionnel et un niveau sémantique.

4.3.1 Le niveau fonctionnel des commandes

Le niveau fonctionnel correspond aux diverses fonctions du système qui permettent de guider un utilisateur (en particulier néophyte) lors de sa navigation dans le système. L'avantage de ce guide est de permettre à l'utilisateur d'éviter de trop nombreux allers-retours entre les différentes fonctions du système.
Il peut fournir à la demande de l'utilisateur la structure d'une commande, sa place dans l'organisation des environnements, les différents arguments nécessaires à l'exécution (leur type et leur nombre), ainsi que le résultat de celle-ci. Pour cela on doit prendre en compte trois éléments :

- les commandes,
- les environnements dans lesquels on regroupe les commandes,
- les données sur lesquelles les commandes travaillent.

En effet, dans un système de C.A.O il existe un nombre important de commandes. Pour permettre à l'utilisateur de mieux les maîtriser, on les organise de façon arborescente (environnement hiérarchisé de commandes). De plus, les systèmes de C.A.O manipulent des données de différentes natures (point, segment, ligne, solide...). Le passage d'un objet à un autre doit suivre un chemin déterminé. Par exemple sur EUCLID, la création d'un objet de type "surface de Bezier" ne peut conduire qu'à un objet de type surface de Bezier mais jamais directement à un objet de type "solide", c'est la structure de données du système qui l'impose.

Ceci constitue un graphe correspondant aux types d'objets dont les nœuds représentent la description formelle de l'objet manipulé, les arcs, les passages autorisés. Ces derniers sont uni-directionnels ou bi-directionnels. Lors d'une session de travail le système manipule uniquement des réalisations concrètes de ce type.

En conclusion, l'utilisateur manipule les objets à l'aide des commandes. Ces commandes ont des paramètres définis d'abord de façon formelle. Lors de l'application d'une commande, l'utilisateur doit fournir des paramètres effectifs correspondant aux types autorisés et les commandes doivent respecter le graphe de dépendances des données.

4.3.2 Le niveau sémantique des commandes

Le niveau sémantique, c'est à dire le niveau où l'on suggère à l'utilisateur, une ou plusieurs démarches avec à chaque fois les raisons de ces divers choix. Il va de soi que les propositions décrites sont dans un cas global, c'est à dire indépendamment des différentes alternatives qui peuvent être proposées ponctuellement.

On peut constater que dans les systèmes complexes, comme les modeleurs 2D ou 3D, il est toujours possible d'appliquer une des commandes disponibles, toutefois, il est beaucoup moins évident de choisir une action utile, en respectant la logique de l'outil ou, de façon plus générale, les principes du modeleur.

Cela veut dire qu'il est possible que deux utilisateurs traitant le même problème puissent aboutir à la même solution en empruntant des cheminements différents. Ainsi, un utilisateur débutant d'un modeleur 2D ou 3D rencontre des difficultés importantes pour maîtriser correctement cette démarche incrémentale. Il lui faudra longtemps pour pouvoir distinguer et

prendre, chaque fois qu'il sera nécessaire, de bonnes décisions concernant la stratégie de développement (voir figure 4.12).

Le rôle de ce deuxième niveau est essentiel dans le système d'aide, en effet, son objectif est double, ainsi il permet :

- d'aider l'utilisateur dans sa réflexion sur le "comment s'y prendre" pour traiter avec un modeleur 2D ou 3D, un problème donné,
- de contribuer à rendre collective une expertise individuelle.

L'utilisateur doit formuler de façon non géométrique les objectifs à concevoir. Le système peut alors consulter sa base de connaissances et proposer à l'utilisateur un certain type de construction. La qualité de l'aide apportée dépend de la capacité de l'utilisateur à formuler la demande et de l'expertise emmagasinée. De ce fait, l'étude suivante nous semble pertinente.

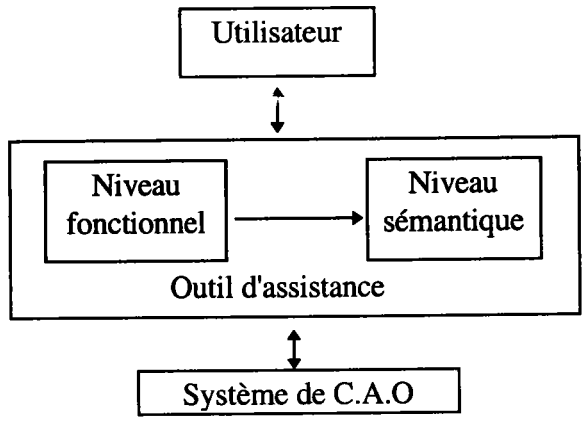


Figure 4.12 L'outil d'assistance

5 Modélisation des questions

Il s'agit de pouvoir répondre aux questions des utilisateurs selon les différents profils qu'ils représentent. En prenant le cas de la construction d'une pièce P avec un modèle B-Rep, plusieurs types de questions, selon leurs catégories, peuvent être classés.

5.1 Différents types de questions

Questions de type analytique
Quels sont les paramètres de la commande "créer contour" ?

Questions de type interrogative
Pourquoi la commande "créer contour" n'a pas été exécutée ?

Questions de synthèse
Comment créer un contour ?
Ai-je le droit de faire une coupure ?
Comment faire pour pouvoir faire une fusion ?

Questions de type alternative
Quelles sont les alternatives à la construction de la pièce P ? (alternative globale).
Quelles sont les alternatives à la suite des commandes proposées ? (alternative directe).
Quelle est l'alternative à la commande fusion1 dans la liste des commandes proposées? (alternative indirecte).
Question de type navigationnel :
Où se trouve la commande de création de contour ?

Toutes ces questions sont exprimées sous la forme de prédicats dont les arguments, si c'est nécessaire représentent les instances obligatoires à la définition des objets sur lesquels on doit agir.

5.2 Exemple de création d'un contour

Nous prenons comme exemple simple la création d'un contour :
Comment faire pour créer un contour?
Représentation : Comment (créer, contour, liste_commande)

La syntaxe des prédicats est la suite :
type_question(type_opération, liste_instanciée, liste de commandes)
type_question(type_opération, liste_non_instanciée, liste_de_commandes)
type_question(type_opération, liste_mixte, liste_de_commandes)

Définitions des opérateurs :
Liste_de_commande : l'enchaînement des commandes réalisant le but.
Liste_instanciée : liste des objets utilisés par l'opération
Liste_non_instanciés : liste des types objets utilisés par l'opération, avec leurs modes de construction.

Liste_mixte :
Liste de types et des objets utilisés par l'opération, avec leurs modes de construction.

5.3 Conclusion

Connaissant le type de question, on peut faire intervenir le mécanisme d'unification pour déterminer :

- si l'on connaît le type de l'objet, on déterminera l'ensemble des opérations,
- si l'on connaît le type de l'opération, on déterminera l'ensemble des objets sur lesquels on peut agir,
- si l'on connaît le type de l'opération et le type de l'objet, on vérifiera la validité de l'opération sur cet objet.

6 L'interaction à travers plusieurs types d'utilisateurs

Cette interaction a pour but d'augmenter la productivité du couple utilisateur-modeleur selon plusieurs critères que nous allons détailler.

6.1 Objectifs

Les objectifs de cette interaction sont les suivantes :

- Permettre à l'utilisateur d'aboutir plus facilement dans son environnement de travail et lui éviter ainsi des recherches inutiles.
- Pousser l'utilisateur à utiliser au moment opportun la ou les fonctionnalités, qui personnalisent le système et le rendent capable de résoudre plus au moins les problèmes posés
- Proposer des fonctionnalités applicables au moment de l'appel.

6.2 L'interaction vue par un utilisateur non expert

Le rôle visé par cette interaction pour ce type d'utilisateur est de faciliter l'utilisation d'un logiciel interactif dédié à la C.A.O en lui procurant les services cités dans le 6.1. Ainsi l'utilisateur se voit libéré du recours permanent à la documentation écrite ou à un expert du système. Le principal outil utilisé est le gestionnaire de dialogue qui sert d'intermédiaire dans les mécanismes d'interaction planificateur-modeleur.
Plus globalement, le système détermine à un instant donné l'état des requêtes formulées par l'utilisateur, c'est-à-dire l'ensemble des commandes faisables et applicables dont nous allons expliquer la signification ci-dessous.

6.2.1 La faisabilité d'une commande

Une commande est faisable si, pour chaque argument de la commande, il existe au moins une instanciation correspondant au type de l'argument. Ce qui permet de minimiser le nombre de commandes que l'utilisateur doit prendre en considération à un instant donné.

Exemple:

Règles	**Préconditions**	**Effets**
Commande Axe	Point, Point	Axe
Commande Cylindre	Axe, Rayon, Hauteur	Cylindre
Commande Tuyau	Trajet, Cylindre	Tuyau

Base de faits
Trajet, Points, Rayon, Hauteur.

Si la faisabilité de la "Commande Tuyau" est à tester, alors il convient de vérifier le but "Tuyau" produit par la règle "Commande Tuyau". Or la base de faits ne contient pas "Commande Cylindre", mais celui-ci est produit par "Commande Cylindre" dont l'un des arguments c'est à dire "Axe" n'existe pas dans la base de fait, donc il sera produit par "Commande Axe" dont tous les arguments existent dans la base de faits, et par chaînage on peut produire "Tuyau".

6.2.2 Applicabilité d'une commande

Une commande est applicable si les arguments fournis par l'utilisateur correspondent aux arguments attendus par la commande. Ce qui permet de contrôler la validité d'une commande. On peut remarquer à travers ces deux définitions qu'une commande est applicable si, et seulement si, elle est faisable.

6.2.3 Acquisition de paramètres

Cette fonction permet de guider l'utilisateur lors de l'acquisition des paramètres d'une commande. Le gestionnaire de dialogue permet d'éviter les erreurs lors de la saisie des paramètres en indiquant leurs types et leurs nombres. De plus une procédure de tri de commandes permet de filtrer les commandes inutiles voir parasites, en effet prenant l'exemple d'un modeleur qui contient plusieurs centaines de commandes, il est tout à fait justifié de chercher à réduire ce nombre pour ne garder à tout moment qu'un ensemble de commandes significatifs par rapport au traitement suivi.

6.3 L'interaction vue par un utilisateur expert

Le rôle de cette interaction par rapport à ce type d'utilisateur (en plus des différentes propositions décrites précédemment) est de mettre à la disposition de ce type d'utilisateur des outils qui vont l'aider à construire, modifier, consulter à travers le système d'aide.

6.3.1 Outils de mise en œuvre

1 - Visualisation de l'arborescence
Lors de la manipulation du système d'aide et en cours d'une session d'activité, il est souhaitable de pouvoir visualiser à tout moment le squelette de la hiérarchie du système.

2 - Manipulation des commandes
Il est aussi souhaitable d'appeler une commande par le nom le plus significatif. Pour cela, le système peut collectionner les différentes désignations dans une classe de synonymes.

6.3.2 Le graphe des commandes

Les différentes fonctions associées aux graphes de commandes sont :

1 - L'insertion
L'insertion d'un élément dans le squelette du graphe de commandes permet son extension.

2 - La modification
La modification permet d'apporter des corrections au graphe. Cette modification (totale ou partielle) peut porter sur tous les éléments introduits par la fonction d'insertion.

3 - La destruction
Comme on modélise les dialogues à l'aide d'un graphe hiérarchisé, alors la destruction d'un nœud en dépend.

Nous terminons cette étude par la partie interface du système d'aide.

7 Vers une interface adaptée au système d'assistance

7.1 Introduction

Le but de cette interface est de pouvoir intégrer l'ensemble des possibilités du système d'assistance (à travers ces fonctionnalités) qui deviennent partie intégrante de l'interface. Un autre objectif est de fournir une présentation des éléments de l'application et de dialogue, adaptée au type de l'utilisateur et une vue sur l'application qui correspond le mieux aux types et aux besoins de ce dernier.
Pour mettre en évidence ces aspects, nous allons détailler les fonctionnalités souhaitées pour une telle interface. Dans une optique de synthèse et pour dégager son architecture ainsi que son fonctionnement, nous prenons comme base de travail les interfaces dédiée à la C.A.O.

7.2 Présentation de l'interface

La présentation des fonctionnalités d'un système par une interface classique se fait généralement par un système de menus ou un ensemble d'icônes comme par exemple le système S.A.C.A.D.O. En outre, cette présentation est figée dans le sens où ce sont toujours les mêmes éléments qui reviennent. Ce genre d'interface pose des problèmes d'ergonomie dans la mesure où les systèmes interactifs évoluent dans le temps et changent fréquemment de type d'utilisateurs. De ce fait, une présentation qui était valable en un moment du fonctionnement du système et pour un utilisateur donné, risque de ne plus être adéquate à un autre moment et pour un autre utilisateur. De ce fait, l'adaptabilité de l'interface nous semble primordiale.

7.3 L'adaptabilité de l'interface

L'adaptation se fait par le composant contrôle qui puise des informations émanant principalement de trois composants.

- Le modèle de l'utilisateur : il constitue une modélisation de l'utilisateur.
- Le modèle de fonctionnement : il définit les règles de fonctionnement de l'application avec toutes les dépendances entre les éléments qui le constituent.
- le modèle d'utilisation : il décrit le contexte d'utilisation du système avec une gestion de l'historique de l'interaction.

Détaillons chaque composant.

7.3.1 Le modèle de l'utilisateur

Le modèle de l'utilisateur est défini par deux caractéristiques : le type de l'utilisateur et le rôle de l'utilisateur :

- Le type de l'utilisateur : les utilisateurs sont classés selon leur niveau d'expertise dans le domaine d'application. Actuellement, nous distinguons trois classes d'utilisateurs : les débutants, les occasionnels et les experts.
- Le rôle de l'utilisateur : le rôle de l'utilisateur détermine le ou les domaines d'utilisation, comme par exemple, dans un système de C.A.O, le dessin 3D, le dessin 2D, etc.; la connaissance du domaine d'application permet de restreindre le système de méthodes aux seules méthodes applicables sur le domaine.

Le rôle de l'utilisateur peut être déterminé de deux manières :

- Statique : c'est le chef de projet qui affecte un utilisateur à une tâche précise et donc à un domaine d'utilisation du système de C.A.O.
- Dynamique : à partir des méthodes utilisées, on détermine le domaine d'utilisation. Dans ce cas, cette détermination du domaine n'est pas ferme dans le sens où l'utilisateur a la possibilité d'accéder à un autre domaine par la seule évocation d'une méthode de ce dernier.

7.3.2 Le modèle de fonctionnement

Pour pouvoir répondre aux demandes de l'utilisateur, l'interface doit connaître les potentialités de l'application. Ces connaissances et capacités sont regroupées en ce que nous appelons le modèle de fonctionnement.
Dans le cas d'un système de C.A.O, le modèle de fonctionnement est essentiellement constitué des trois éléments suivants :

- Un arbre fonctionnel des méthodes : représentant la hiérarchie des méthodes constituant le système de C.A.O.
- Un graphe de dépendance : définissant les relations entre les différents types d'entités manipulées par le système.
- Un arbre des méthodes : représentant des domaines d'utilisation du système de C.A.O.

7.3.3 Le modèle d'utilisation

Le modèle d'utilisation regroupe les connaissances nécessaires pour déduire la faisabilité et l'applicabilité des méthodes.
Ces connaissances se présentent sous la forme d'un graphe de dépendances renseigné, d'une part, par l'existence ou la non existence d'instances correspondant aux différents types d'entités et, d'autre part, par l'ensemble d'instance de ces types. Ces informations seront essentiellement utilisées par le contrôle dont nous allons présenter le rôle.

7.4 Le contrôle d'interface

Dans le chapitre 3 nous avons vu le rôle du contrôle de manière générale, nous allons maintenant présenter son rôle en ce qui concerne les connaissances propres à l'application et qui sont exprimées au niveau des trois modèles à savoir : le modèle de l'utilisateur, le modèle de fonctionnement, et le modèle d'utilisation.

L'interface doit permettre de rendre l'application adaptable aux caractéristiques de l'utilisateur et ne présente que ce qui est faisable au moment de l'interaction.

7.4.1 Caractéristiques de l'utilisateur et adaptabilité

L'adaptabilité aux caractéristiques de l'utilisateur consiste en l'exploitation des informations sur l'utilisateur.
Concrètement, le contrôle reçoit deux types d'informations sur l'utilisateur, fournis par le modèle de l'utilisateur. Ces informations sont le type d'utilisateur et le domaine d'application. D'un autre côté, le contrôle reçoit l'arbre fonctionnel contenant toutes les méthodes étiquetées par leur niveau de difficulté.

Ayant reçu ces informations, le contrôle réalise des filtres. Le premier filtre consiste à ne garder que les méthodes qui sont d'un niveau correspondant au type de l'utilisateur. Le deuxième filtre porte sur le domaine d'application qui produit, à partir du sous-arbre déjà produit un sous arbre de méthodes par le premier filtre, pour ne garder que les méthodes du niveau de l'utilisateur et qui portent sur le domaine d'application désiré (voir figure 4.13).

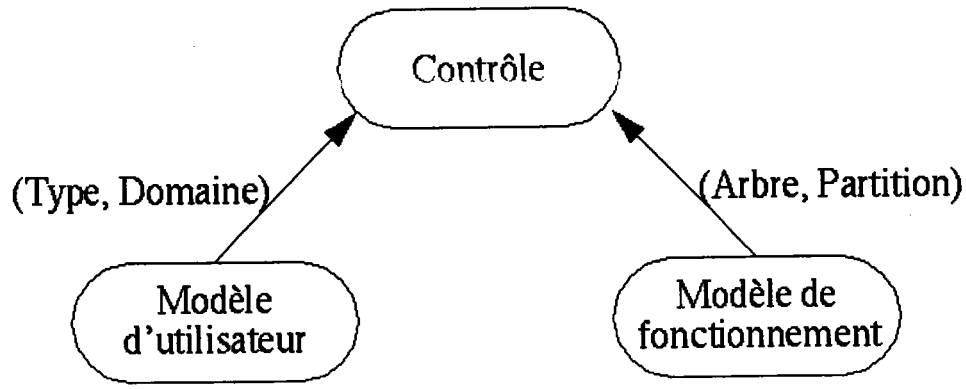


Figure 4.13 Le sous-arbre des méthodes

7.4.2 Les méthodes faisables

Pendant l'utilisation d'un système interactif, l'utilisateur crée ou détruit des objets. Ces modifications entraînent des changements d'états au niveau du contexte utilisateur.

A chaque manipulation d'objet, le contrôle reçoit la faisabilité des méthodes concernées par le type de cet objet et met à jour le sous-arbre représentant l'ensemble des méthodes faisables (voir la figure 4.14).

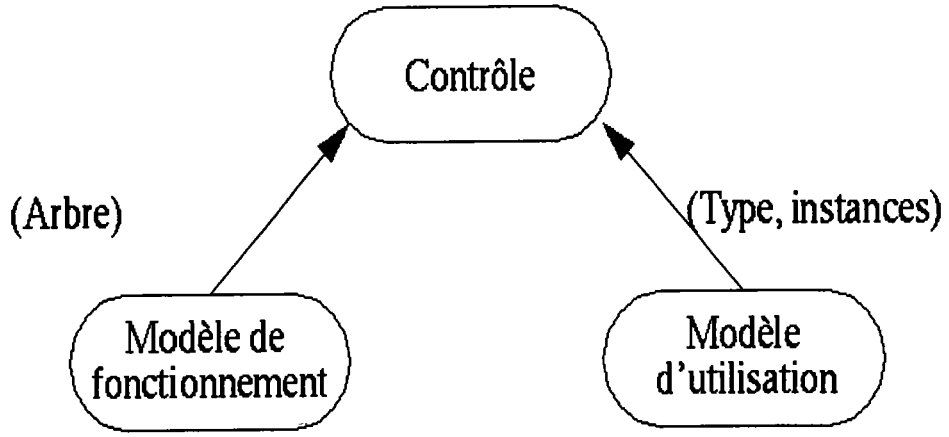


Figure 4.14 Le sous-arbre des méthodes faisables

7.4.3 Les méthodes applicables

A tout moment l'utilisateur se retrouve devant un ensemble de méthodes faisables qui ne sont pas toujours les mêmes, mais l'applicabilité de ces méthodes dépend des arguments présentés.

Pour vérifier l'applicabilité d'une méthode, le contrôle exploite les instances présentées comme arguments de la méthode. Dans le cas où la méthode n'est pas directement applicable, le contrôle déclenche un mécanisme de déduction pour vérifier l'applicabilité directe ou indirecte de la méthode. Si c'est le cas, une liste de méthodes est suggérée à l'utilisateur qui se charge de l'instanciation (voir figure 4.15).

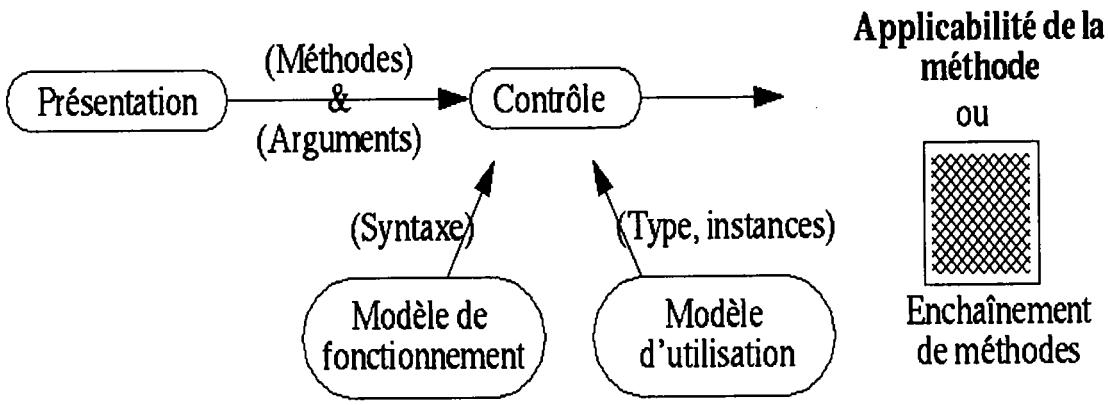


Figure 4.15 L'applicabilité d'une méthode

8 Conclusion

Le but de ce chapitre était de montrer la validité de notre approche "théorique" sur des cas concrets de réalisation (construction et assemblage) de pièces mécaniques. En particulier, le but était de situer notre approche dans un contexte multi-utilisateurs (expert, non expert) et multi-modèles (B-Rep, CSG).

Dans le premier cas, la gestion des différents utilisateurs se situe au niveau des outils de mise en œuvre à la disposition des utilisateurs selon leurs profils :

- simple et efficace pour un utilisateur non expert,
- plus au moins complexe pour un utilisateur expert.

Dans le deuxième cas, la notion de multi-modèles est présentée comme une ouverture du système vers des modeleurs de conception différentes (B-Rep, CSG, ...).

Les différentes interactions entre les modules nécessitent une cohérence parfaite dans la stratégie globale de mise en œuvre d'une conception. En effet, l'alternative entre les deux méthodes présentées dans ce chapitre (cas du clapet par exemple) ne peut se faire que si l'on maîtrise plusieurs facteurs qui sont : les intentions de l'utilisateur, son profil et la complexité de la conception.

Conclusion

Ce travail a débuté par une étude sur les systèmes qui associent C.A.O et I.A, les différentes approches ont été décrites et discutées et des exemples de systèmes existants ont été passés en revue et commentés.
Compte-tenu de la multiplicité des modèles proposés et dans une perspective d'adéquation de nos critères plus spécifiques à la construction et à l'assemblage, nous nous sommes efforcés de définir une structuration qui puisse prendre en compte tous les aspects du couplage entre la C.A.O et l'I.A.

L'objectif de cette première partie est donc de dégager les grands principes qui contribuent à l'élaboration d'un système d'aide dédié à la C.A.O et de montrer que la planification interactive peut contribuer à l'amélioration de l'interactivité Homme-Machine indépendamment de l'application produite.
Parallèlement à ce premier travail nous avons présenté les fondements formels de l'étude de l'interactivité Homme-machine d'une part, et de la planification d'autre part. Le but de ces deux études est de définir les critères que peut supporter notre système, pour une meilleure coopération entre la tâche à accomplir par l'utilisateur et les possibilités de notre modèle.

Dans la deuxième partie de ce travail, outre le domaine d'application qu'est la construction mécanique et l'assemblage en C.F.A.O, nous avons élargi notre étude à un contexte plus général dans l'interactivité Homme-machine. Par ce fait, nous nous sommes rendu compte de la complexité à mettre en place des interfaces Homme-machine performantes.
Les concepts généraux d'élaboration des interfaces Homme-Machine y ont été décrits et commentés et des exemples de modèles les plus significatifs comme Seeheim, PAC et PAC-Seeheim ont été étudiés et discutés. Ainsi, le découpage d'une application interactive en trois blocs (modèle de Seeheim) conduit généralement à des architectures hétérogènes.

Notre objectif de départ est de proposer à l'utilisateur un système d'aide à la construction et l'assemblage de pièces mécaniques plus souple que les systèmes traditionnels, capable de le guider dans les tâches à accomplir sans pour autant le contraindre dans sa démarche de conception.
A partir de ce fait et grâce à ces deux études, nous avons mis en évidence les différentes facettes d'un système d'assistance interactif dédiés à la C.A.O et dégagé l'organisation de ce dernier.

Parmi les problèmes que nous avons traités, celui qui est lié à la notion de buts à long terme nous apparaît comme essentiel. En effet, dans le cadre d'un système de C.A.O, il est important de souligner que dans la plupart des cas il n'y a pas de buts à long terme, mais des tâches élémentaires à exécuter. La prédiction des intentions de l'utilisateur est impossible dans ce cas et, lorsque il y a violation de contraintes, par exemple, le guide accompagnant le logiciel se contente de faire des suggestions sans plus.
La situation de buts à long terme a été intégrée dans notre étude et par conséquent, prise en compte dans le modèle d'architecture que nous avons proposé, tout en gardant la possibilité d'exécuter des tâches élémentaires.

Pour atteindre l'objectif fixé grâce au modèle, nous avons exploré les différentes phases d'un système d'assemblage et, pour chaque phase, nous avons déterminé le type de caractéristiques que le système doit prendre en compte pour aider l'utilisateur.

En ce qui concerne la modélisation des composants nous avons proposé un modèle issu de la théorie des graphes et qui prend en compte les caractéristiques non géométriques du produit.

Pour ce qui est de la génération des séquences d'assemblage, outre les contraintes géométriques et non géométriques à prendre en compte, il s'agit de faire varier cette liste de séquences autant que possible.
Dans les propositions que nous avons présentées dans la troisième partie de notre étude, nous avons axé notre description sur les points clés suivants :

- Le système permet à un utilisateur selon son profil (expert ou non expert), de situer la tâche qu'il veut accomplir et les possibilités qui lui sont offertes, ce qui d'une part, améliore l'apprentissage nécessaire à tout utilisateur pour s'adapter au système et, d'autre part, permet d'arriver (selon les cas) plus rapidement à la tâche fixée.
- Le système permet d'orienter l'utilisateur dans le cas d'inadéquation entre les demandes de l'utilisateur et les possibilités du système.
- L'intégration des mécanismes de la planification interactive dans l'architecture globale du système d'aide permet, à partir d'une situation particulière et, d'autre part, en fonction d'un but, de sonder tous les chemins possibles, selon un schéma en largeur d'abord, l'interactivité intervient comme un instrument de filtrage.
- L'utilisateur peut éventuellement solliciter une tâche impossible à satisfaire par le système au cours des échanges, auquel cas, il faudra déterminer avec précision à partir d'où il faut reprendre le processus d'échange grâce à l'élaboration d'un historique du dialogue.

La dernière partie de notre travail est applicative, à travers l'étude de la construction d'un clapet et d'un alternateur. En effet, l'idée maîtresse est de montrer que l'originalité de l'approche choisie (dans la construction de notre système d'aide) permet des applications différentes tant aux niveaux des pièces à construire que des utilisateurs qui interviennent.
Ainsi, nous montrons que la complexité de la construction ne dépend plus des capacités de l'utilisateur à maîtriser un outil, aussi complexe soit-il. En effet, l'assistance à la construction d'un produit est issue d'un processus itératif. Le concepteur répète en affinant plusieurs fois son produit jusqu'à ce qu'il soit complètement satisfait.
Pour assister le concepteur dans ce processus, le système d'assistance à l'assemblage de pièces mécaniques prépare des fonctions variées, par exemple la fonction *undo* Cette fonction est appelée fonction de retour car elle est censée permettre le retour à une étape antérieure dans le processus de conception. Le problème est que la plupart des systèmes d'aide en C.A.O ne la propose pas. Pour notre part ce type de fonction à été intégré dans l'architecture globale du système d'aide.

De plus, le couplage système d'aide-modeleurs que nous avons proposé permet de faire appel à différents modeleurs, ce qui a pour conséquence une meilleure flexibilité du système.

Pour compléter notre étude, nous avons proposé une interface adaptée au système d'aide et adaptable aux utilisateurs, dont le but est, d'une part, intégrer l'ensemble des possibilités du système d'aide et, d'autre part, de prendre en compte les profils des utilisateurs et leurs démarches de conception. Cette partie, quoique décrite, reste à réaliser dans la pratique.

Dans la suite de nos travaux nous envisageons de poursuivre la mise en œuvre de l'ensemble des mécanismes décrits dans cette thèse, d'étendre le couplage système d'aide-modeleurs à d'autres modeleurs géométriques autre que les modèles B-Rep et CSG, comme par exemple, le modèle par extrusion.

Enfin, il serait intéressant de comparer l'efficacité de l'utilisation de notre système d'aide par des expériences appropriées sur des utilisateurs experts et non experts et d'évaluer les améliorations éventuelles à apporter au système pour une plus grande efficacité.

Annexe A

Quelques généralités sur la construction mécanique

Dans l'annexe suivante, nous allons préciser le domaine de travail dans lequel s'est effectué ce travail, c'est-à-dire la construction mécanique, ainsi que les différentes fonctions principales programmées.

1 La construction mécanique

La construction mécanique est un domaine qui regroupe des thèmes très différents par leurs approches et leurs finalités. On peut distinguer dans l'ordre :

1 - La saisie des besoins

Elle fait intervenir l'établissement d'un cahier des charges qui lui même fait appel à l'analyse du marché (information sur le fournisseur, les concurrents, la sous-traitance ...) en évaluant les besoins du demandeur (questionnaire suivi d'un entretien).

2 - L'analyse fonctionnelle

Il s'agit de définir les fonctions et les coûts associés à la conception d'un produit.
Pour ce qui est des fonctions, il existe plusieurs types de fonctions dépendantes les unes des autres, d'une part les fonctions globales c'est-à-dire les fonctions dont le rôle est joué par un moteur ou un ensemble destiné à remplir une fonction de service (voir figure 2), d'autre part les fonctions composantes qui sont des fonctions hiérarchisées qui peuvent être indispensables à la réalisation de la fonction globale.
Enfin, il y a les fonctions principales qui se composent essentiellement des fonctions de résistance et de liaison.

A ces fonctions on peut rajouter les fonctions spécifiques que sont les fonctions de guidage ainsi que les fonctions de contraintes.
Pour ce qui est des coûts, trois principales évaluations interviennent, celles qui concernent le produit lui même, celles plus liées aux fonctions, et enfin les composants technologiques (à acheter ou à fabriquer).

3 - Etude d'avant projet

Il s'agit de justifier le choix d'un projet parmi plusieurs qui se présentent.
On appelle aussi cette partie pré-développement.

4 - La conception définitive

C'est la spécification technique et économique du produit associée à une organisation industrielle qui doit permettre la définition du produit (dessin, étude du procédé de fabrication).

5 - Industrialisation
Mise en place de l'outil de production (gamme, temps, outillage, contrôle...).

Pour finir on peut schématiser l'ensemble de ce processus par les figures suivantes :

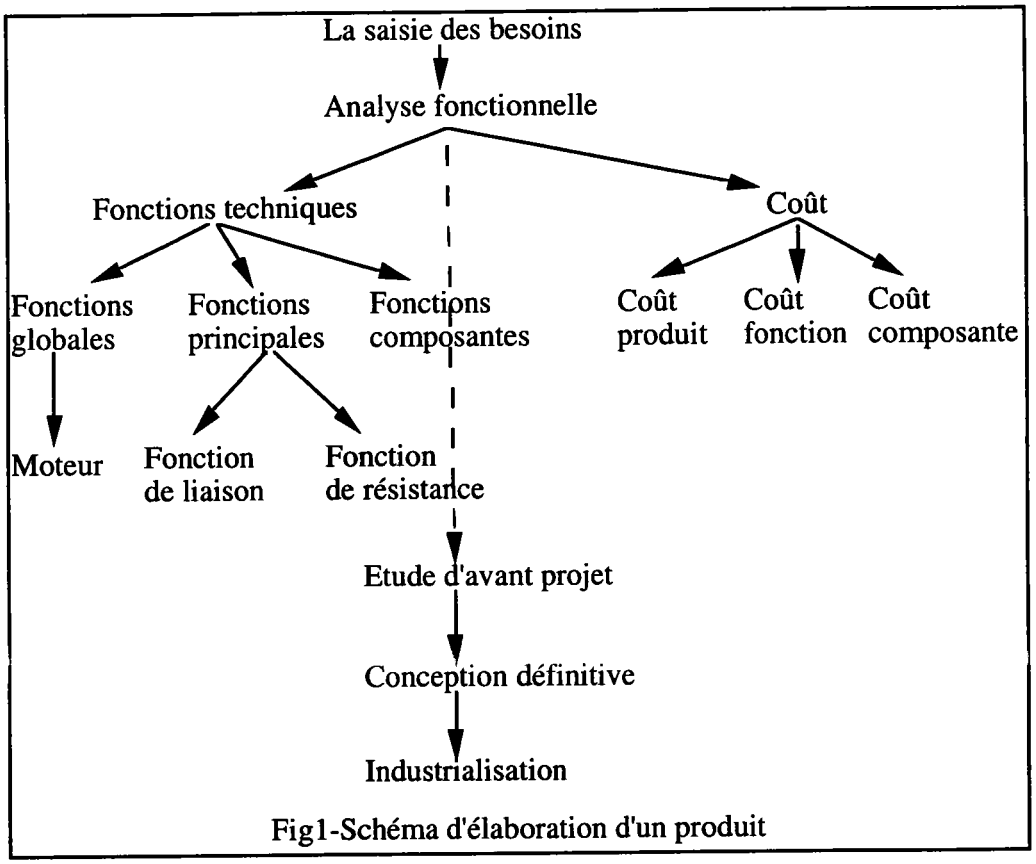


Fig1-Schéma d'élaboration d'un produit

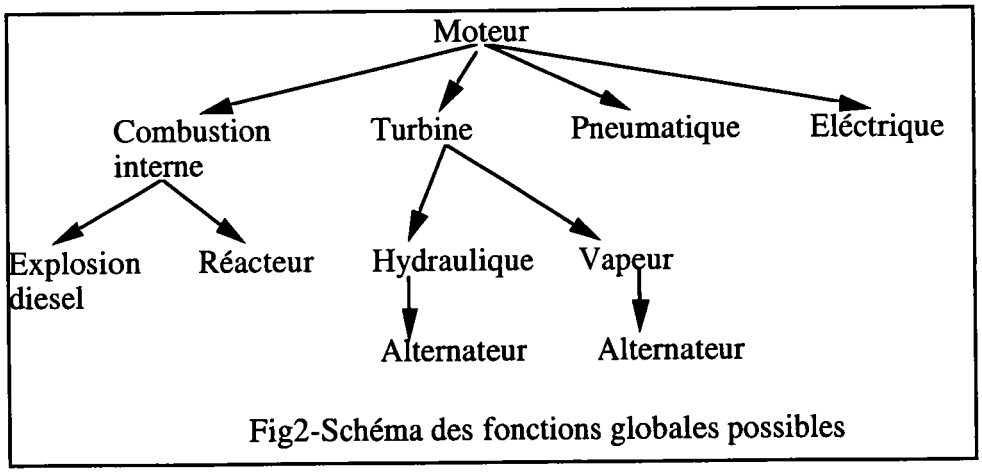


Fig2-Schéma des fonctions globales possibles

2 Le domaine de travail

Le schéma conceptuel intègre en premier lieu la composante associée au domaine de travail, cette composante appelée aussi niveau métier, concerne les actions des experts et des utilisateurs.

Il s'agit de définir un vocabulaire commun à ces deux profils.

L'analyse des besoins permet d'affiner cette composante sur laquelle va se baser le système dans l'assistance à l'élaboration du produit.

2.1 Le vocabulaire utilisé

L'expertise dans le domaine de la construction mécanique se base à la fois sur un support technique et une évaluation des besoins du concepteur.
Cette expertise se fait en deux phases.

Première phase
C'est la partie fonctionnalité de la pièce, elle permet de définir entièrement la pièce avant, soit une intervention directe (c'est à dire usinage de la pièce), soit avant mise sous assemblage.
Les caractéristiques techniques de la pièce peuvent être très complexes selon les niveaux de difficultés de conception du produit.

Elles sont de deux natures :

1- Des pièces accessoires filetées par exemple (écrous, goujons d'assemblage, rondelles) leurs caractéristiques concernent :
a - leurs formes : tête, corps, écrous, extrémités,
b - leurs dimensions,
c - leur résistance,
d - leur désignation.

2- Des pièces uniformes, leurs caractéristiques concernent :
a - leurs formes : carré, parallélépipède, rectangle, etc..
b - leurs dimensions,
c - leur résistance,
d - leur désignation.

Deuxième phase :
C'est la partie assemblage, elle permet de lier les pièces précédemment définies entre elles suivant les possibilités de liaisons qu'elles impliquent.

2.2 Concepts de base

On reprend les concepts de base de la construction mécanique ensuite, chaque étape est étudiée plus précisément :

Donc, la chronologie à partir de laquelle un produit est conçu en construction mécanique est :

1 - La saisie des besoins :
Entretiens et questionnaires.

2 - L'analyse fonctionnelle :
Définition des fonctions et évaluation des coûts.

3 - L'étude d'avant projet :
Justifier le choix d'un projet parmi plusieurs qui se présentent.

4 - La conception définitive

5 - L'industrialisation :
Mise en place de l'outil de production.
6- La qualification, homologation.

2.3 Spécifications techniques et économiques d'un produit

Nous nous contentons dans un premier temps de mettre en évidence les caractéristiques des deux premiers thèmes c'est-à-dire la saisie des besoins et l'analyse fonctionnelle qui représentent l'ossature essentielle lors de la conception d'un produit.

2.3.1 La saisie des besoins

Elle fait intervenir l'établissement d'un cahier de charge qui fait appel à l'analyse du marché en évaluant les besoins du demandeur.
Les différentes phases d'un cahier de charges sont :

1 - Formulation des besoins à l'aide d'un questionnaire et d'entretiens.
2 - Etude de marché à travers l'information sur les fournisseurs, les concurrents, la sous-traitance, les normes et les règlements.

2.3.2 Analyse fonctionnelle

Il s'agit de définir les fonctions et les coûts associés à la conception d'un produit, pour cela on doit dans l'ordre :
1 - recenser le produit,
2 - caractériser le produit,
3 - ordonner le produit,
4 - hiérarchiser le produit,
5 - valoriser le produit.

Nous allons dans la suite d'une part, définir comment évaluer les coûts associés à la fabrication d'un produit et d'autre part, définir les fonctions techniques associées à un produit.

2.3.3 Evaluation des coûts

Trois principales évaluations interviennent à ce niveau, celles qui concernent le produit lui même, celles plus liées aux fonctions, et enfin, les composants technologiques.

1 - Le produit :
Deux types de coût :
a - coût complet : évaluer avec distribution incluse,
b - coût objectif : c'est le coût plafond exprimé par le demandeur.

A partir de ces deux coûts on définit soit une conception pour un coût objectif (coo), soit une conception pour un coût global (ccgo).

2 - Les fonctions :
Il s'agit d'abord d'analyser les coûts selon une norme particulière suivant un ordre hiérarchique des fonctions et ensuite d'optimiser la relation coût/performance, c'est-à-dire de proposer éventuellement des modifications de l'analyse de la valeur sur la base de nouveaux critères d'appréciation.

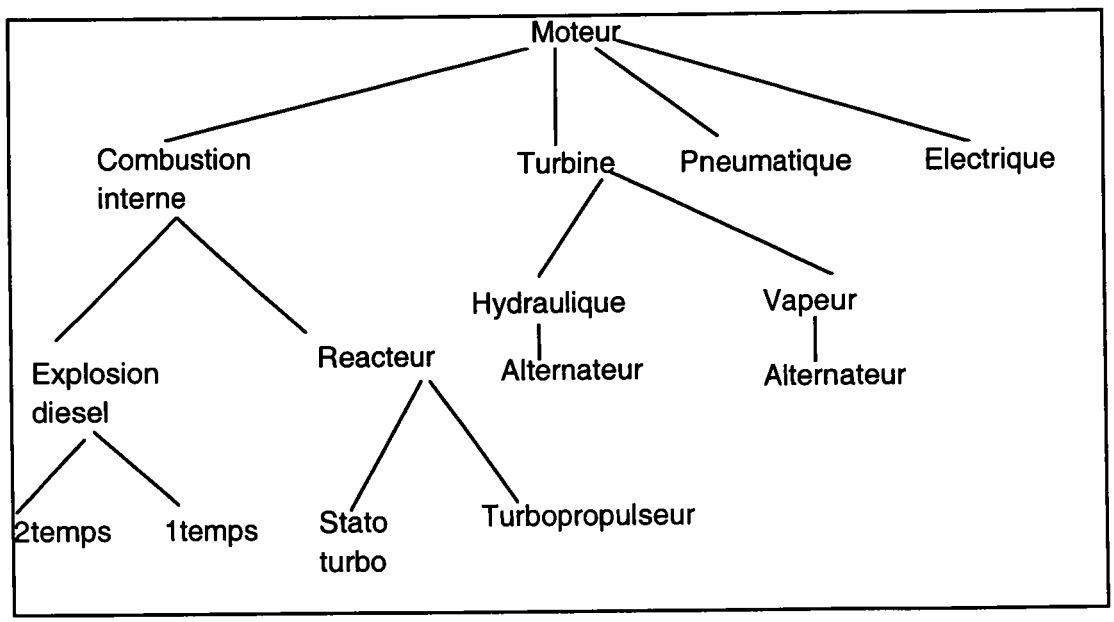
3 - Les composants technologiques :
Il s'agit soit de les fabriquer spécialement soit de les acheter et, dans les deux cas, il faut faire un choix sur les matériaux selon la classification suivante :
1 - métaux : origine, semi-produit, obtention des bruts, état des surfaces bruts, etc..
2 - matière plastique : joints, coussinets, poignées, roulettes,
3 - matériaux composites : spécialiste.
Il s'agit ensuite de mettre le produit sous une forme finale (moulage en sable ou bien moulage en coquille, usinage), et de définir quelle qualité de surface fonctionnelle il permet d'obtenir.

3 Fonctions techniques en construction mécanique

On distingue à ce niveau :

3.1 Fonctions globales

C'est le rôle joué par un moteur, une pompe, un ensemble destiné à remplir une fonction de service. Le rôle du moteur peut être défini suivant la hiérarchie suivante :



Ainsi, si le choix se porte sur un moteur électrique (le plus courant) il faudra rajouter les conditions d'installation, le type de courant et d'alimentation disponibles c'est à dire moteur à courant continu ou moteur à courant alternateur.

Au préalable il faut faire le choix entre un moteur seul et un moteur avec un mécanisme adjoint.

3.2 Fonctions composantes

Elles sont définies par l'analyse des valeurs et elles sont hiérarchisées selon qu'elles sont plus ou moins indispensables à la réalisation de la fonction globale.
L'analyse de la valeur est une méthode de compétitivité organisée et créative visant la satisfaction du besoin de l'utilisateur par une démarche spécifique de conception qui soit à la fois fonctionnelle, économique, et pluridisciplinaire.

La démarche est la suivante :

Quand :
Répartition de l'origine des coûts, conception, reconception.

Qui :
Le demandeur.
Le décideur : fait connaître ou bien, fait appliquer.
L'animateur : organise, gère le suivi.
Le groupe interdisciplinaire examine , trouve un consensus.

Comment :
Orientation de l'action.
Recherche de l'information.
Analyse des fonctions et des coûts .
Recherche d'idées, de voies de solutions.
Etude et évaluation des solutions.
Bilan prévisionnel et proposition de choix.
Suivi et réalisation.

3.3 Fonctions principales

Elles se composent essentiellement des fonctions de résistance et de liaison.

3.3.1 Les fonctions de résistance

Il s'agit pour l'essentiel de définir les différents matériaux (homogènes, pièce simple, déformation) et les sollicitations, c'est-à-dire sélectionner la pièce en deux tronçons et réduire le torseur. Ensuite, il s'agit de calculer la charge théorique de la pièce, elle se fait selon la procédure suivante :

1 - calculer la charge à l'aide d'une étude statistique,
2 - majorée de 10 % à 100 % selon la régularité de transmission et autres incertitudes,
3 - considérer à l'aide de ses 2 composantes : radiale et axiale.
3.3.2 Les fonctions de liaisons

A ce niveau nous avons deux types de liaisons :

1- Liaisons avec les pièces voisines :

- prévoir deux pièces : résistance, coût, montage, entretien,
- liberté de mouvement de la pièce A/B : fait appel à la cinématique,
- forme des surfaces de liaison précision et rugosité : qualité/coût, montage, réglage, démontage,
- quel type d'action s'exerce entre A et B : isolement, étude de l'équilibre,
- quelle est l'intensité de cette action : calcul de la résistance de A et B,
- quelle grandeur doit avoir la surface de liaison : pression, etc.

2 - Liaison rigide :

Choix des diamètres du cylindre, forme du cylindre, etc.

3 - Guidage des pièces :

- Rotation autour d'un axe : guidage direct, interposer des pièces accessoires, contacts coniques, contacts sphériques.
- Translation rectiligne : modification du cylindre, prismes divers.

Annexe B

Principales fonctions du planificateur en langage Common Lisp

Le langage de programmation utilisé est le "common lisp"

```
,,,La fonction menus
  ***************
(defun menu ()
(format t " ~% " )
(format t "                    MENU    ~%" )
(format t "                   ********  ~%~%~%~%~%" )
(format t "                 1- NON EXPERT  ~%" )
(format t "                 2- EXPERT      ~%" )
(format t " ~% " )
(format t "                    VOTRE CHOIX   " )
 (let* ( (rep (read))
      )
      (cond
              ( (= rep '1)
               (menu1)
           )
              ( (= rep '2)
            ;(menu2)
              )
           ( (format t "EREUR  " )
            (menu)
           )

      )
  )
)


,,,La fonction lecture de commandes
;;;**************************
(defun Lire_Commande ()
        (let* (
          (liste (read))
            (commande (first liste))
          )
          (cond
                ( (eq commande 'arete)
                        (let
                 (
                 (n (second liste))
                 (liste (lire_arete))
                      )
                 (setq liste  (nth (1- n)  liste))
                 )
            )
                ( (eq commande 'face)
                        (let
```

```
                (
                (n (second liste))
                (liste (lire_face))
                     )
                (setq liste  (nth (1- n)  liste))
                )
               )
          ( (eq  commande 'contour)
                    ;( fonction a appeler)
          )
               ( (eq  commande 'volume)
                    ;( fonction a appeler)
               )


         )
     )

)
;;; Fonction de lecture des différents opérateurs dans les fichiers
;;;**********************************
(defun lecture1 (fich)

 (let ((res(read fich nil nil)))
  (if  (null res)
     ()
    (cons res (lecture fich))
    )
    )
 )
(defun lire1 ()

      (let* ( (res (open "resultat" :direction :input))
            (liste (lecture res))
            )
      (close res)
      (imp-fich1 liste)
      liste
      )
)
(defun lire_operate ()

      (let* ( (res (open "operate" :direction :input))
            (liste (lecture res))
            )
      (close res)
      liste
      )
)
```

```
(defun lire_initial ()
      (let* ( (res (open "initial" :direction :input))
            (liste (lecture res))
            )
      (close res)
      (imp-fich1 liste)
      liste
      )
)
(defun lire_final ()
      (let* ( (res (open "final" :direction :input))
            (liste (lecture res))
            )
      (close res)
      (imp-fich1  liste)
      liste
      )
)
(defun lire_arete ()
      (let* ( (res (open "arete.lsp" :direction :input))
            (liste (lecture res))
            )
      (close res)
      liste
      )
)
(defun lire_face ()

      (let* ( (res (open "face.lsp" :direction :input))
            (liste (lecture res))
            )
      (close res)
      liste
      )
)
(defun imp-fich1 (liste)
  (let ((res (open "result" :direction :output)))
    (impp-fich1 liste res '0)
    (close res)
    )
  )
(defun impp-fich1 (liste res n)
  (unless (null liste)
        (format res "~S~%~%" (car liste))
        (format t "Etat numero ~S~%" (1+ n))
        ;(format t "~S~%~%" (car liste))
        (affiche-element (car liste))
        (impp-fich1 (cdr liste) res (1+ n))
        )
```

```
 )
(defun affiche-element (liste)
 (if (null liste)
      ()
      (progn
      (format t "~S~%" (car liste))
      (affiche-element (cdr liste))
      )
      )
      )

;;;*    UNE PARTIE DU GENERATEUR DE PLAN EN CHAINAGE AVANT
;;;*************************************************************************
#-:mote.lsp(load"mote.lsp")
#-:nmot.lsp(load "nmot.lsp")
#-:loo.lsp(load "loo.lsp")
#-:initial.lsp(load "initial.lsp")
#-:lecture.lsp(load "lecture.lsp")
#-:commande.lsp(load "commande.lsp")
#-:menu.lsp(load "menu.lsp")
#-:menu1.lsp(load "menu1.lsp")

;;;fonction pour le plan global
;;;************************************
(defun plan ()
(setq OPERATEURS (lire_operate))
(setq  base-c f-init)
(setq $ARB (cons (list '-1 f-init) ()))
(setq ch (choix-nœud))
(format t "~%")
(menu)
(if (null base)
   (setq base '(rien))
   ()
)
(loop
  (if (null   base )
    (return)
    (progn
          (ens-operateur base-c)
    (loop
      (if (or (null LISTE-OP)(null pilebuts))
          (return)
          (let (( op (choix-operateur)))
            (loop
              (if (or (equal  (car $SOLUT)  'fsol)(null pilebuts))
                    (return)
                       (let ((sol  (choix-solution)))
                       (loop
```

```
                (if (null sol)
                    (return)
                    (progn
                    (setq sol1 (car sol))
                    (applic-op op sol1 base-c)
                    (ajouter-nœud $N-ARB op sol1)
                    (setq sol (cdr sol))
                (if (compare BASE  F-FIN)
                     (progn
                     (chemin $N-ARB (1-  (length $ARB)))
                     (resultat-ch)
(format t "~%")
(format t "LES OPERATEURS A APPLIQUE DANS LE MODELEUR  SONT :  ~%" )
(format t "************************************************** ~%~%")
(imp-resultats $RESULTATS)
(setq $global  (list  $resultats  $global))
(format t "~%")
(setq $choix $resultats)
(format t "VOULEZ-VOUS UNE AUTRE SOLUTION O/N  ~%")
                        (setq rep (read))
                        (if (eq rep  'o)
                          T
                          (progn
 (format t "L'ETAT DE LA BASE EST MAINTENANT   ~%~%")
 (format t "*********************************  ~%~%")
 (imp-resultats f-fin)
 (format t "~%")
 (format t "VOULEZ-VOUS CONTINUER O/N  ~%")
                           (setq rep (read))
                           (if (eq rep  'o)
                                (progn
                                (setq $ARB (cons (list '-1 f-fin) ()))
                                (setq f-init f-fin)
                       (menu1)

                                (initialise)
                                (setq ch (choix-nœud))
                                (setq base-c f-init)
                                (return)
                                )
                           (progn
                           (setq  LISTE-OP  '())
                           (setq  $SOLUT  ( cons 'fsol $SOLUT))
                           (setq  BASE '())
                           (return)
                           )
                          )
                         )
                        )
```

```
                    )
                    ()
                   )
                  )
                )
               )
               )
          )
         )
         (setq $SOLUT (cdr $SOLUT))
        )
       )
    )
   )
)
(if ( null base)
   (return)
   (if (null pilebuts)
        ()
(progn
(setq $SOLUT '())
(setq $SOL-2 '())
(setq $SOL-3 '())
(setq $SOL-4 '())
(setq $LISTE-CH '())
(setq $RESULTATS '())
(setq ch (choix-nœud))
(if (equal ch NIL)
   (progn
(format t "~%")
(format t "DESOLE IL N' Y A PLUS DE SOLUTIONS!!! ~% " )
(format t "VOULEZ FAIRE UN CHOIX PARMIS LES SOLUTIONS DEJA PROPOSE?
O/N~%~% ")
    (setq rep (read))
    (if (eq rep 'o)
        (progn
        (imp-resultats $choix)
        (return)
        )
        (return)
    )
    )
    (progn
 (chemin (first ch) (second ch))
 (setq base f-init)
 (setq base-c (reconstit $liste-ch))
 (setq $RESULTATS '())
 (setq $LISTE-CH '())
    )
```

```
 )
 )
 )
)
)
 (if (null $LISTE-CH)
    NIL
    T
 )
)
;;;fonction qui choisit une solution parmi la liste de solution
;;;*********************************************************
(defun choix-solution()

(let (( sol (car $SOLUT )))
   (setq $SOLUT (cdr $SOLUT))
   sol
)
)

;;;fonction qui avance dans la liste de chemin des operateurs
;;;*************************************************
(defun avance()
 (if (ATOM (car $LISTE-CH))
    $LISTE-CH
    (progn
    (setq $LISTE-CH  (cdr $LISTE-CH))
    (avance)
    )
 )
)
;;;fonction qui delivre les resultats du chemin sous forme d'operateurs
;;;*********************************************************
(defun resultat-ch ()
(loop
(let* (
      ( p (car $LISTE-CH))
      ( l1 (second(second $LISTE-CH)))
      ( l2 (second(third  $LISTE-CH)))
      ( l3 (nth p OPERATEURS))
      (xx  'x)
      (yy  'y)
      (ii  'i)
      )
 (COND
 (  (= p  '0)  (progn
              (setq $RESULTATS (append $RESULTATS (subst l1 'x (second l3))))
              (setq $LISTE-CH (cdr $LISTE-CH))
              (avance)
```

```
            ) )
( (= p '1) (progn
            (setq $RESULTATS (append
                                $RESULTATS
                                (car ( remplace
                                    (list (list xx l1) (list yy l2))
                                    (second l3)
                                    ))))
            (setq $LISTE-CH (cdr $LISTE-CH))
            (avance)
            ) )

( (= p '2) (progn
               (setq $RESULTATS
                   (append
                   $RESULTATS
                   (car(remplace
                         (list (list xx l1) (list ii l2))
                         (second l3)))))
            (setq $LISTE-CH (cdr $LISTE-CH))
            (avance)
            ) )
( (= p '3) (progn
            (setq $RESULTATS (append
                                $RESULTATS
                                (car ( remplace
                                    (list (list xx l1) (list yy l2))
                                    (second l3)
                                    ))))
            (setq $LISTE-CH (cdr $LISTE-CH))
            (avance)
            ) )
)
 (if (null $LISTE-CH )
    (return)
    ()
 )
)
)
)

 ;;;fonction de reconstitution
 ;;;*********************
 (defun reconstit (liste)

 (if (null liste)
    base
    (let* (( op1 (car liste))
           (sol1 (list (second liste) (third liste))))
```

```
    (applic-op op1 sol1 base)
    (reconstit (cdr(cdr(cdr liste))))
    )
)
)
```

Bibliographie

[Allen, 83] James F. Allen "Maintaining knowlegde about temporal intervals", *Communications of the ACM*, VOL.26 n°11, 1983 PP.832-843.

[Allen, 84] James F.Allen "Towards a general theory of action and time"*AI journal*, PP 123-154.

[Alty, 89] J.L Alty, J Mullin, "Dialogue Specification in the Gradient-Dialogue System", People and Computers. Vol.5, No8 -- pp.151-169 - Septembre 1989.

[Bass, 91] L. Bass, R. Little, R. Pelligrino, S. Reed, R. Seacord, S. Shepard, M. Szezur, "The Arch Model : Seeheim Revesited" User interface Developers, Workshop, Avril 1991.

[Bellalem, 91] L. Bellalem "Planification interactive et représentation temporelle, application à la conduite d'un système de dialogue Homme-machine". Rapport de DEA, CRIN, 1991.

[Bellalem, 97] L. Bellalem, Y. Gardan, A. Zakari " An Assistant Model in C..A..D. Systems Based on Interactive Planning" International Conference On Manufacturing Automation. ICMA'97, Hong Kong, 28-30 Avril 1997.

[Bessière, 83], P. Bessière, "Etude de la génération de plans en univers multi-agent", Thèse de Docteur-Ingénieur de l'INPG, Grenoble, 1983.

[Bourjault, 84] A. Bourjault, "Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires", Thèse de doctorat, Université de Franche-Comté, France, 1984.

[Carbonell, 83] J.G Carbonell "formulating and Generalizing Plans from Past Experience" In Michalski R.S., Carbonell J.G & Mitchell T.M. Machine Learning. An Artificial Intelligence Approach, Springer-Verlag.

[Coutaz, 90] J. Coutaz, "Interface Homme-Ordinateur, conception et réalisation", Dunod informatique, 1990.

[Coutaz, 91] J. Coutaz, L. Nigay, "Seeheim et architecture multi-agent" journées IHM'91, Dourdan, 1991.

[De Mello, 89] L. S. Homem De Mello, A. C. Sanderson "Automatic Generation of Mechanical Assembly Sequences", Proceedings of IEEE Internationnal Conference of Robotic and automation, Scottsdale, Arizona, pp 56-61, 1989.

[Delmas, 90], J. P. Delmas, C. Houellebcq, P. Mazas, D. Ouvry, P. Thoraval, "Système expert et C.A.O : ARCHIX, un système d'aide à la conception d'un véhicule en phase d'avant projet", MICAD 1990, France.

[Descotte, 87] Y. Descotte, H. Delasale, "Une architecture de système expert pour la planification d'activité", Proceedings of sixth International Workshop an expert system and their applications, Avignon, pp 903-916, Avril 1987.

[Duval, 92a] T. Duval, "Etude de quelques modèles d'architecture pour les applications interactives", Rapport de recherche IRIN-1992.

[Duval, 92b] T. Duval, "Le modèle d'architecture logicielle SPA", Rapport de recherche IRIN-1992.

[Duval, 92c] T. Duval, J. Hameon "Le modèle d'architecture SPA et un mécanisme permettant son évaluation", Communication aux journées IHM'92, 1992.

[Fikes et al, 72] R.E. Fikes, P.E Hart, N.J Nilsson, "Learning and Executing Generalized Robot Plans", Artificial intelligence, vol 3.1972.

[Gardan & Zakari, 91] Y. Gardan, A. Zakari, "L'intégration dans les systèmes de C.F.A.O : à travers le modèle ou à travers les outils ?", Acte de la convention IA. 1991.

[Gardan, 88] Y. Gardan, J. P. Jung, J.N.Kolopp, C. .Minich, W. Totino, "Une approche nouvelle de la convivialité dans un système de C.A.O : les principes du dialogue dans S.A.C.A.D.O", Actes de MICAD 88, Hermès, pp281-196.

[Gardan, 91] Y. Gardan, "La C.F.A.O introduction, techniques, et mise en œuvre", Hermès, 3eme édition, 1991.

[Gardan, 93] Y. Gardan, J. P Jung, B.Martin "An End-User Oriented Approach to Design Man Machine Interface for CAD/CAM", in proceeding of IEEE International Conference on Systems, Man and Cybernetics, Le Touquet, FRANCE pp525-530, 1993.

Ghallab & al, 89] M.Ghallab, A.M.Allaoui, "Relations temporelles symboliques : représentations et algorithmes", Revue d'intelligence artificielle, vol.3, no3/1989, pp67 à 115.

[Haren, 85], P. Haren, B. Neveu, O. Corby, M. Montalban, "MEPAR un moteur d'inférence pour la conception en ingénieurie", 5ème Congrès de RFIA 1985.

[Haton, 91] J.P Haton et al "Le raisonnement en intelligence artificielle", InterEdition, Paris 1991

[Hilario, 89] M. Hilario "L'apprentissage d'heurristiques de séquencement de buts" Colloque Intelligence Artificielle, Université du Maine, Le Mans, septembre 1989.

[Horman, 72] H. Horman "introduction à la psycholinguistique et langage ", Larousse, 1972

[Karsenty, 91] S Karsenty., "La construction d'interfaces utilisateur, Génie logiciel et Systèmes experts", 1991.

[Kibler, 83] D. Kibler D, B.Porter "Episodic Learning". Proceeding of the National Conference on Artificial Intelligence. AAAI-83.

[Kimura, 89] F. Kimura, Y. Yamaguchi, K. Kobayashi, "News features Geometric Modelling for Product Description", International Symposium on Advanced Geometric Modelling for Engineering Application, Berlin, 1989.

[Laperriere, 91] L. Laperriere, H. ElMaraghy, "Automatic Generation of Robotics Assembly Sequences", International Journal of Advanced Manufacturing Technology, 1991.

[Martin, 95] B. Martin, "Contribution pour une nouvelle approche du dialogue Homme-machine en C.F.A.O", Doctorat de l'université de Metz, 1995.

[McDermott, 80] J. .McDermott. "A Rule Based Configure of Computer Systems", Technical report 80-119, CMU, april 1980.

[Michard, 87] A. .Michard, A. Giboin, C. Mais, P. Verdret "PLANEX : Système d'aide à la planification. Son application aux activités bureautiques ". INRIA Sophia-Antipolis, 1987.

[Minton, 88] S.Minton "Learning Effective Search Control Knowledge : An Explanation Based Approach" Phd Thesis, Canargie-Mellon, 1988.

[Monceyren, 90a] E. Monceyren, "Developping large industrial ICAD systems: un example for harbor design", Fourth Eurographics Workshop on intelligent CAD systems, Avril 1990.

[Monceyren, 90b] E. Monceyren, "EXPORT, un poste de travail d'ingénierie portuaire", Rapport interne, Novembre 1990

[Newell, 82] A. Newell, "The Knowledge Level", Artificial Intelligence 1982.

[Nilsson, 80] N. Nilsson, "Advanced Plan Generating System", Chapter 8 of Principles of Artificial Intelligence", TIOGA Publishing Company, 1980.

[Norman, 86] D. Norman, S. Draper, "User centred design", Lawrence Erlbaum Associates Publishers, 1986.

[Patil, 81] R. Patil & al., "Causal understanding, of patience illness in medical diagnosis", IJCAI 1981.

[Pfaff, 85] G. E. Pfaff, "User interface management systems", Springer-Verlag, Eurographics seminars serie, 1985.

[Pierret, 90] C. Pierret, "Un outil d'acquisition et de représentation des tâches orienté objet", INRIA, Rapport de recherche technique no 1063, 1990.

[Sacerdotti, 80] E. Sacerdotti, "Plan generation and execution for robotics", Technical notes 209 SRI, April 1980.

[Samin, 90] G. Samin, J Vandevelde, "DIADEME : devis informatisé et aide au dimensionnement et à l'étude des machines électriques", Convention I.A 1990.

[Scheifler, 86] R. W. Scheifler, J. Gettys, "The XWindow Systems", ACM transaction on graphics, Volume 5, no 2, avril 1986, pp57-96.

[Shneiderman, 83] B. Shneiderman, "Direct manipulation : a step beyond programming languages", IEEE Computer, pp 57-96, 1983.

[Simmon, 93] R.Simmons, R.Davis " The Roles of Knowledge and Representation in Problem Solving", in SGES 1993

[Stefik, 85a] M. Stefik, "Planning with Constraints (MOLGEN : part I)", Artificial Intelligence, Vol.16 111-140, 1985.

[Stefik, 85b] M. Stefik, "Planning and Metaplanning (MOLGEN : part II)", Artificial Intelligence, Vol.16 141-170, 1985.

[Tenenberg, 86] J.Tenenberg "Planning with Abstraction" Proceeding of the 5th National Conference on artificial Intelligence, AAAI-86.

[Whitney, 88] D. E. Whitney, T. L. Defazio, R. E. Gustavon, S. C. Graves, K. Cooprider, C.A. Holmes, C. J. Klein, M. Lui, S. Pappu, "Computer Aided Design of Flexible Assembly Systems", report no. CSDL R-2033, C.S. Draper Laboratory Inc, Cambridge, Mass, 1988.

Résumé :

L'introduction des techniques d'intelligence artificielle a permis de faire évoluer les systèmes de C.A.O vers des systèmes plus performants. Des domaines comme la construction mécanique en général et les méthodes d'assemblage en particulier ont largement bénéficié de ces techniques. Pourtant, les systèmes proposés se basent le plus souvent sur une approche système expert dont l'objectif est de fournir des méthodes générales de construction ou d'assemblage, où l'utilisateur reste dans une position d'attente passive, ou alors, quand ce n'est pas le cas, sa participation se borne à agir sur certains paramètres de construction mais jamais sur l'activité d'assemblage elle même. En outre les systèmes interactifs dédiés à ce type d'applications sont souvent partiels et spécifiques à chaque application.
Dans ce travail, nous proposons une contribution à la mise en place d'un système d'aide fondé sur la planification interactive dédié à la C.A.O. Ce système permet de générer des séquences d'assemblage de manière interactive, l'utilisateur agissant comme un partenaire du système. L'approche choisie se base sur la coopération entre les mécanismes de la planification interactive, la modélisation des objets C.A.O. sous forme prédicative à travers des diagrammes de relations ainsi que la gestion du dialogue. Une interface Homme-Machine vient compléter l'architecture globale avec des caractéristiques spécifiques au système d'aide.
Le couplage système d'aide-modeleurs que nous proposons permet de faire appel à différents modeleurs, ce qui a pour conséquence une meilleure flexibilité du système.

Mots clés :

CAO, Assistance, Planification, IA, Système Expert, Interactivité, Gestionnaire de Dialogue, Interface Homme-Machine.

Résumé :

L'introduction des techniques d'intelligence artificielle a permis de faire évoluer les systèmes de C.A.O vers des systèmes plus performants. Des domaines comme la construction mécanique en général et les méthodes d'assemblage en particulier ont largement bénéficié de ces techniques. Pourtant, les systèmes proposés se basent le plus souvent sur une approche système expert dont l'objectif est de fournir des méthodes générales de construction ou d'assemblage, où l'utilisateur reste dans une position d'attente passive, ou alors, quand ce n'est pas le cas, sa participation se borne à agir sur certains paramètres de construction mais jamais sur l'activité d'assemblage elle même. En outre les systèmes interactifs dédiés à ce type d'applications sont souvent partiels et spécifiques à chaque application.
Dans ce travail, nous proposons une contribution à la mise en place d'un système d'aide fondé sur la planification interactive dédié à la C.A.O. Ce système permet de générer des séquences d'assemblage de manière interactive, l'utilisateur agissant comme un partenaire du système. L'approche choisie se base sur la coopération entre les mécanismes de la planification interactive, la modélisation des objets C.A.O. sous forme prédicative à travers des diagrammes de relations ainsi que la gestion du dialogue. Une interface Homme-Machine vient compléter l'architecture globale avec des caractéristiques spécifiques au système d'aide.
Le couplage système d'aide-modeleurs que nous proposons permet de faire appel à différents modeleurs, ce qui a pour conséquence une meilleure flexibilité du système.

Mots clés :

CAO, Assistance, Planification, IA, Système Expert, Interactivité, Gestionnaire de Dialogue, Interface Homme-Machine.