Contribution à la conception et réalisation d'un système de gestion de bases de données pour la conception assistée par ordinateur

Gabriel Michel

UNIVERSITE DE METZ
UER SCIENCES EXACTES ET NATURELLES

THESE

présentée par

Gabriel MICHEL

pour obtenir le titre de

Docteur de 3ème Cycle en Informatique

Sujet

Contribution à la conception et à la réalisation d'un système de gestion de bases de données pour la conception assistée par ordinateur

Thèse soutenue le 9 février 1988 devant la commission d'examen.

M. CREHANGE
Y. GARDAN
G. GOVAERT
B. HEULLUY
J.P. JUNG

Thèse préparée au sein du Laboratoire de Recherche en Informatique de l'université de METZ.

Je tiens à remercier:

- Yvon GARDAN qui m'a accueilli dans son équipe, m'a encouragé et orienté chaque fois que c'était nécessaire.

- Francis MICHEL de sa patience durant nos longues discussions concernant les exemples de conception mécanique.

- Marion CREHANGE pour sa lecture critique de cette thèse.

- Jean-Pierre JUNG de sa disponibilité et de l'aide qu'il m'a apportée.

- Gérard GOVAERT et Bernard HEULLUY pour leurs conseils.

- Tous les membres du Département Informatique de l'IUT ainsi que du LRIM pour leur soutien amical.

Isabelle, Marilyne et Rachel pour le soin et la gentillesse avec lesquels elles ont assuré la frappe de ce document.

PLAN

INTRODUCTION :

PARTIE 1 : BASES DE DONNEES ET CAO

I . Les différentes informations utilisées dans un processus de CFAO

I.1 Caractéristiques des informations en CFAO

I.2 Les informations utilisées en CFAO

I.3 Découpage des données en CFAO

I.3.1 Base de Données Projet et Base de Données de Connaissances
I.3.2 Exemples de propositions de découpages
I.3.3 Conclusion concernant ces découpages

I.4 Données gerées dans les systèmes BD-CAO actuels

I.5 Propriétés de la Base de Données Projet

II. Apport des modèles de données et des SGBD actuels à la construction d'un système SGBD-CAO

II.1 Rappels sur les notions de modèles de données et SGBD

II.2 Qualités requises aux modèles de données et SGBD

II.3 Systèmes BD-CAO existants

II.4 Insuffisances des SGBD classiques

III. Axes de recherches pour un SGBD en CAO

III.1 Nouveaux modèles de données et nouvelles fonctionnalités

III.1.1 Modèles de données issus de l'Intelligence Artificielle

III.1.1.1 Les réseaux sémantiques
III.1.1.2 Les schémas
III.1.1.3 Les représentations orientées objet et les langages orientés objet

III.1.2 Bases de Données Déductives

III.2 Quelques études actuelles

III.2.1 Extension du modèle relationnel
III.2.2 Elaboration de modèles sémantiques
III.2.3 Exemples de modèles de données utilisés dans des systèmes BD-CAO

IV. Le système SACADO

IV.1 Les Données

IV.2 Architecture générale de SACADO

IV.2.1 Architecture pour un projet donné
IV.2.2 Relations inter-projets

IV.3 Modèle générique

IV.3.1 Les Modèles
IV.3.2 Gestion des versions et Bases de Données

IV.4 Modèles de dialogue

PARTIE 2 : LE MODELE

I. Le Modèle :

I.1 Présentation des fonctionnalités du Modèle

I.1.1 Définition et manipulation simultanées des données
I.1.2 Conception ascendante et descendante
I.1.3 Création imbriquée
I.1.4 Héritage et déduction
I.1.5 Objets incomplets et objets incohérents

I.2 Le modèle de données

I.2.1 Les prototypes
I.2.2 Les attributs et facettes

I.2.1.1 Définitions
I.2.2.2 Attributs descriptifs
I.2.2.3 Attributs structurels et fonctionnels
I.2.2.4 Autres attributs

I.2.3 Les procédures
I.2.4 Les contraintes d'intégrité
I.2.5 Les instances

I.2.5.1 Définitions
I.2.5.2 Version d'un projet

I.3 Les traitements

I.3.1 Définition et manipulation des données de conception

I.3.1.1 Opérations des données de conception
I.3.1.2 Opérations sur les ensembles
I.3.1.3 Opérations sur les procédures
I.3.1.4 Opérations sur les instances
I.3.1.5 Quelques fonctionnalités spécifiques au Modèle

INTRODUCTION

Les premiers systèmes de conception assistée par ordinateur (C.A.O.) ont résolu des problêmes ponctuels ou encore certaines étapes clés d'un processus de conception. Ils étaient constitués d'une série de programmes correspondant à des tâches bien précises: le besoin d'intégrer ces programmes dans des chaines de programmes s'est fait sentir.

Mais comme l'enchainement des programmes est apparu comme une solution trop lourde et trop contraignante, la tendance actuelle est d'organiser le processus de conception autour des données (donc de la base de données). Ces données constituent le noyau des systèmes de CAO et l'on parle maintenant de systèmes BD-CAO.

Dans un premier temps les tentatives de construction de tels systèmes ont été faites à partir des systèmes de gestion de bases de données (SGBD) classiques: cet axe de recherche a été abandonné du fait des limitations de ces SGBD (modèles sémantiquement trop pauvres, fonctionnalités trop restreintes). Les tendances actuelles vont dans le sens d'une extension du modèle relationnel ou encore de l'élaboration de modèles sémantiques.

Le but de notre étude est de concevoir un système BD-CAO capable de gérer les informations non géométriques circulant dans un processus de conception (tout en restant en liaison avec le modèle géométrique), de réaliser certains modules de ce système et de le valider par le biais d'exemples de conception. Cette étude fait partie d'un projet plus important de création d'un système de CAO appelé SACADO.

Dans la première partie nous mettons d'abord en évidence les différentes informations utilisées dans un processus de conception, puis nous présentons les axes de recherche de systèmes BD-CAO ainsi que les outils de l'Intelligence Artificielle pouvant être utilisés pour la conception de tels systèmes. Ensuite nous décrivons l'aspect gestion des données du projet SACADO.

Dans la deuxième partie nous présentons le modèle de données choisi ainsi que les opérations qui s'y appliquent. Ensuite nous décrivons la structure de données (et ses liens avec le modèle de données) ainsi que les mécanismes d'attachement et de déclenchement procéduraux.

Trois exemples de conception mécanique sont présentés dans la troisième partie: ces exemples permettent de fixer le cadre d'utilisation du système BD-CAO et de la maquette associée.

La quatrième partie est consacrée à la description de deux propositions d'interfaces: leur objectif étant d'offrir à tout opérateur un dialogue aussi convivial que possible.

PARTIE 1 : BASES DE DONNEES ET CAO

Le but de cette première partie est de mettre en évidence le cadre d'utilisation d'outils informatiques actuels (modèles de données, SGBD, intelligence artificielle) pour la gestion d'informations de CAO.

En premier lieu nous décrivons au paragraphe 1 les différentes informations circulant dans un processus de conception et fabrication assistées par ordinateur (CFAO).

Ensuite dans le paragraphe 2 est étudié l'apport des modèles de données classiques et des SGBD associés (c'est à dire destinés aux informations de gestion) à la construction d'un système BD-CAO.

Au paragraphe 3 sont présentés différents outils (modèles de données, fonctionnalités) pouvant permettre de construire un SGBD pour la CAO: quelques maquettes actuelles allant dans ce sens y sont décrites.

Enfin dans le dernier paragraphe (c'est à dire le paragraphe 4) nous présentons SACADO, un projet plus global de conception d'un système de CAO auquel appartient notre étude.

I) Les différentes informations utilisées en CFAO :

[ADI-83], [CHO-83a], [DAR-83], [DAV-81], [GAR-83], [GAR-85], [GAR -86a], [MIC-86], [RIE-85]

Dans ce premier paragraphe nous allons dans un premier temps décrire les informations intervenant dans un processus de conception assistée par ordinateur (paragraphes I.1 et I.2) puis nous étudions quelques propositions de répartition de ces informations en différentes Bases de Données (paragaphe I.3).
Les informations et les Bases de Données gérées dans le cadre des sytèmes et maquettes actuelles sont analysées au paragraphe I.4.
Enfin au paragraphe I.5 nous étudions les propriétés de la Base de Données Projet qui est la Base de Données associée à un processus de conception donné.

I.1) Caractéristiques des informations en CFAO :

Les données utilisées en gestion sont toutes du type alphanumérique. De plus, en gestion, il y a peu de "types d'entités" mais beaucoup d'occurrences par type d'entités alors qu'en CFAO il y a beaucoup de "types d'entités" mais peu d'occurrences de "type d'entités".
Les données utilisées en CFAO sont complexes, variées et nombreuses:

- complexes: les objets à concevoir ont des structures complexes, avec des contraintes importantes sur les liens.

- variées: dessins, textes, connaissances.

- nombreuses: la réalisation du modèle demande la conception ou l'utilisation de nombreux objets dont la structure est complexe (éléments du modèle, standards, anciennes versions, ...). Par exemple plusieurs millions de pièces pour la conception d'un avion ou plusieurs milliers de portes pour un VLSI.

I.2) Les informations utilisées en CFAO :

. modèle du projet en cours
. versions antérieures de ce modèle
. produits conçus (plans, étapes de la conception, résultats, normes...)
. catalogues de pièces, pièces standards
. connaissances expertes (manière de concevoir le produit, savoir-faire de l'entreprise, règles)
. données FAO : parc des machines, capacités des machines, techniques de fabrication, temps, coût ...
. données de gestion (càd non techniques): budget, fournisseurs, personnel, ...

I.3) Découpages des données de CFAO :

I.3.1) Base de Données Projet et Base de Données de Connaissances :

L'étude d'un certain nombre de travaux montre que le découpage des données n'est pas le même selon les différents auteurs. Il en ressort malgré tout 2 types de données CAO, regroupées en différentes bases de données. Il s'agit de la Base de Données de Projet (BDP) contenant les données du projet en cours et de la Base de Données de Connaissances (BDC). (Ces notions ont d'ailleurs été introduites en Intelligence Artificielle dans le domaine des systèmes experts).

Il existe des variantes, selon les auteurs, du contenu de la BDC et de la BDP. Toutefois, il en ressort toujours les différences essentielles suivantes au niveau:

- de la dynamicité: la BDP est en constante évolution (au niveau de sa structure et de ses données) alors que la BDC est plus statique (la quantité de modifications faites sur cette base, par exemple ajout d'une nouvelle règle, est beaucoup plus faible).

- de l'accessibilité: la BDP est propre à un concepteur (ou à un groupe de concepteurs) alors que la BDC est commune à tous les concepteurs.

Nous allons donner maintenant quelques exemples de répartition des différents types d'informations CFAO dans ces Bases de Données.

I.3.2) Exemples de propositions de découpages :

Selon les différentes propositions de découpages, on peut distinguer différentes catégories d'organisation de données : certaines tendent vers une Base de Données unique où on ne ferait pas de distinction entre données "existant" et données "conception", mais cette tendance est marginale.

- dans [FOI-82] on distingue:

a) La Base de Données de Connaissances (BDC): constituée de données figées et stabilisées constituant l'acquis de l'entreprise.

b) La Base de Données Projet (BDP): qui regroupe les informations liées à un projet donné; elle reflète à tout moment l'état d'avancement du projet. La Base de Projet est elle-même divisée en deux bases logiques:

. La Base de Données de Version (BDV): constituée des différentes versions et représentations du modèle. La BDV permet de mémoriser l'évolution du processus de conception.

. La Base de Données de Travail (BDT): contient les informations liées à une ou plusieurs versions en cours de manipulation

- dans [DGM-80] on définit:

a) Une base de Données Générales (BDG) contenant des informations "générales", c'est à dire de gestion: personnel, budget, informations financières, ...

b) Une Base de Données Techniques (BDT) constituée d'informations techniques (matériel, fournisseurs, délais de livraison, ...) et de Connaissances (comment concevoir, méthodes de conception, projets déjà résolus)

c) Une Base de Données Projet (BDP) qui intégrera toutes les informations concernant le projet en cours.

- dans [DAR-83] on définit:

a) La BDC qui est constituée par:

- les produits conçus.
- les catalogues.
- les connaissances expertes.
- les données de gestion.

b) La BDP:

- à un projet seront rattachées à tout moment plusieurs BDP (plusieurs personnes peuvent travailler dans des domaines différents à la conception d'un même produit et les informations contenues dans ces BDP seront différentes: prise en compte du problème de vues multiples en CAO).

I.3.3) Conclusion concernant ces découpages:

On peut constater que dans ces découpages on distingue deux Bases de Données: la BDP et la BDC avec différents niveaux de richesses sur le contenu de ces bases (par exemple intrégration ou non des données concernant la FAO, le cahier des Charges,...), ces bases étant souvent elles-mêmes éclatées en d'autres Bases de Données.

I.4) Données gérées dans les systèmes BD-CAO actuels :

. La Base de Données Générales définie dans [DAV-81] est une Base de Données classique (qui est souvent intégrée dans la Base de Données de Connaissances pour les autres auteurs) contenant des données de gestion : elle n'est pas spécifique à la CFAO.

. Les données concernant la fabrication (machines, temps, coûts, ...) quant à elles sont rarement prises en compte lors des projets de systèmes BD-CAO.

. Les connaissances expertes sont toujours classées dans la BDC: quand elles sont prises en compte, il s'agira généralement d'informations sur la manière de concevoir les produits (par exemple règles d'assemblages). On utilisera ici des notions de faits et de règles. Les connaissances expertes sont des types de données qui ne sont pas spécifiques à la CAO et qui relèveraient plutôt de l'Intelligence Artificielle.

Dans les études actuelles, a priori seules les données concernant le projet en cours, les produits conçus, les catalogues sont prises en compte. Il arrive cependant que le cahier des charges ou certaines connaissances expertes soient intégrées dans les maquettes de systèmes BD-CAO.

I.5) Propriétés de la Base de Données Projet :

Voici une liste non exhaustive de propriétés de la Base de Données Projet (donc du SGBD qui permettrait de la prendre en compte).

- Volume d'information: en constante évolution. Alors que dans le cas d'une BD classique, on travaille surtout en mises à jour, dans la BDP, on est constamment en création/modification.

- Durée de vie: elle est relativement courte, allant de quelques jours à un an (c.à.d. la durée de la conception du modèle) ce qui est peu par rapport aux Bases de Données de gestion. Donc les problèmes d'optimisation de place mémoire sont moins importants.

- Complexité des objets: les informations manipulées sont souvent complexes, par exemple une pièce mécanique peut être composée de plusieurs centaines d'objets différents.

- Dynamicité du schéma: En gestion, le schéma de la Base de Données est créé une fois pour toutes: les utilisateurs remplissent la Base de Données avec des valeurs correspondant à ce schéma. Il sera difficile et même souvent impossible de modifier ce schéma par la suite: lorsqu'on parle de langage de manipulation de données (LMD), il s'agit de données = valeurs. Une des particularités d'un système CAO est que les informations manipulées par l'utilisateur sont aussi bien des structures d'objets que des valeurs associées à ces structures. D'où il faudrait un LMD où données = valeurs + structures.
A ce niveau on parlera d'extensibilité et d'évolutivité.

. l'extensibilité (ou redéfinition): c'est la possibilité de prendre en compte de nouvelles caractéristiques (attributs, relations) d'où modification de liens entre attributs, des structures d'entités, des valeurs.

. l'évolutivité (ou restructuration): c'est la possibilité de modifier la structure de l'objet; il n'y aura modification que des liens entre entités.

- représentations multiples: (correspond au problème de vue en BD classique). Le même objet doit pouvoir avoir des représentations différentes (par exemple un circuit électronique doit pouvoir être représenté d'une manière physique, logique, fonctionnelle ou électrique).

- cohérence sémantique: très importante car l'univers de conception est très mouvant donc susceptible de posséder de nombreuses contradictions.

- cohabitation de données "texte", "dessin" et des "connaissances".

- le cahier des charges: considéré comme un ensemble de contraintes éventuellement violées tout au long de la conception et satisfaites uniquement à la fin du processus.

- Accès à la BDP: de façon transparente à l'utilisateur. Il n'y a pas de notion d'administrateur de Bases de Données, de même un langage d'interrogation ne suffit pas (les accès à la base seront faits par les utilisateurs finals: manipulation de la structure et des données).

- Possibilité de définir des fonctions: qui, soit permettront d'apprécier la conception (progiciels de calculs des bureaux d'études), soit de calculer et déduire de nouvelles informations.

Cette liste de propriétés n'est pas exhaustive, mais doit permettre de mieux définir le modèle de données ainsi que le SGBD qui permettrait de prendre en compte la Base de Données Projet.

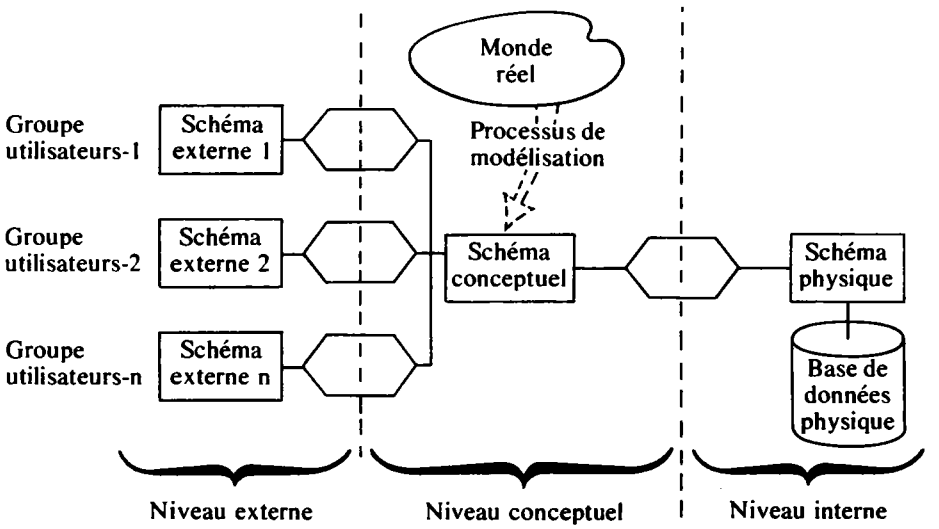
II) Apport des modèles de données et des SGBD actuels à la construction d'un système SGBD-CAO :

[ADI-82], [DAT-81], [CHO-83a], [DAR-83], [MIC-86], [RIE-85]

Dans ce paragraphe, nous allons étudier l'adéquation des systèmes de gestion de base de données actuels (et des modèles de données associés) par rapport à la gestion des informations de CAO. Au paragraphe II.1 sont rappelées les notions de modèles de données et de systèmes de gestion de bases de données (SGBD), puis sont définies les qualités requises à la CAO (paragraphe II.2). Puis nous donnons quelques exemples de systèmes BD-CAO construits à partir de SGBD classiques et nous montrons leurs limites par rapport à la gestion des informations circulant dans un processus de CAO (paragraphe II.4).

II.1) Rappels sur les notions de modèles de données et de SGBD : [ADI-82]

Un modèle de données permet de décrire en termes abstraits mais fidèles une certaine réalité d'une organisation: les objets du monde réel sont classés en catégories et désignés par des noms. Avant de soumettre les données au SGBD, celles-ci doivent être mises sous une forme appelée modèle de données, adaptée à leur traitement par un tel système. Différents modèles de données existent et ont donné lieu à différents types de SGBD (hiérarchique, réseau, relationnel). Le processus de modélisation par le biais d'un modèle de données aboutit au schéma conceptuel. Le SGBD fournit un langage de définition de données qui permet de spécifier le schéma conceptuel.
Cette démarche de modélisation est représentée par la figure ci-dessous.



Les différents niveaux de représentation d'une base de données: (schéma ANSI-SPARC)

II.2) Qualités requises aux modèles de données et SGBD : [ENC-82], [GRA-82], [RIE-86]

Les applications de CAO et particulièrement la mise en oeuvre de la BDP nécessitent l'utilisation d'un modèle de données adapté, permettant de représenter des propriétés particulières que les modèles de données classiques (hiérarchiques, réseau, relationnel) ne peuvent mettre en évidence. Un modèle de données pour la CAO (et le SGBD destiné à le gérer) devra donc permettre:

- la modélisation de structures dynamiques (le SGBD devra permettre une description et une manipulation intégrées des données, ce qui est en général nettement séparé dans les SGBD classiques).

- l'expression d'opérations sur ces structures (par exemple créer une propriété commune à un ensemble d'objets).

- l'expression d'une bonne sémantique (pour exprimer non seulement de nombreuses contraintes sur un objet, mais aussi entre objets différents).

- l'expression et le maintien de la cohérence (tout en permettant des descriptions incomplètes ou des incohérences momentanées durant la conception d'un objet).

- la gestion d'objets indéfinis ou non-complètement définis: il doit être possible pendant toute la durée de vie d'un objet de créer, modifier ou supprimer ses propriétés et celà à différents moments de la conception. De plus pour un objet donné on pourra avoir certaines de ses propriétés qui ne seront pas instanciées.

- la possibilité de création d'objets imbriqués, de création ascendante et descendante.

- la possibilité de déduction (de nouvelles propriétés ou même de nouveaux objets).

De plus un SGBD pour la CAO devra posséder des outils de gestion de versions (validation, restauration, cohérence entre versions). En plus de l'évolution d'un objet dans le temps celui-ci doit pouvoir être vu de différentes manières selon le type d'utilisateur: par exemple pour un bâtiment, l'architecte, l'électricien et le plombier auront une perception différente ou encore pour un circuit intégré il existera une vue logique, une vue fonctionnelle ou encore une vue électrique.

II.3) Systèmes BD-CAO existants : [BIL-83]

Dans la plupart des systèmes de CAO existants, ce qui est appelé "Bases de Données" est en fait un ensemble de fichiers gérés par un système de gestion de fichiers classique. Cet aspect "Base de Données" a d'ailleurs été étudié dans [GAR-86].

L'importance de la gestion des données dans un système de CAO a conduit à étudier son adéquation à des SGBD classiques (hiérarchiques, réseaux ou relationnels).

Plusieurs systèmes (le plus souvent opérationnels) ont été bâtis autour de ces types de SGBD. On peut citer:

- TORNADO ([ULF-80]) et PHIDAS ([FIS-79]) construits autour de SGBD de type réseau. D'autres systèmes de ce type sont définis dans [HUB-79] et [ZIN-81].

- VERDI ([BOU-82]) construit autour d'une SGBD de type hiérarchique.

D'autres exemples de systèmes peuvent être trouvés dans [HEU-86].

Par contre de nombreux travaux récents présentent des prototypes construits autour de SGBD de type relationnel. Ceci est dû à la simplicité de ce modèle et aussi au fait qu'il existe de nombreuses versions de SGBD relationnels sur micros. A notre connaissance il n'existe pas encore à ce jour de systèmes construits autour d'un SGBD relationnel. Cependant de nombreuses maquettes basées autour de ce modèle, ou d'une extension de ce modèle, ont été conçues à ce jour (se reporter pour celà au paragraphe III.2).

II.4) Insuffisances des SGBD classiques :

Un des inconvénients de ces SGBD provient de la difficulté à modifier le schéma de la BdD (sauvegarde de l'ancienne BdD, recompilation du schéma, etc...). On ne peut pas faire n'importe quelle modification du schéma, et dans le cas où une modification est possible elle est très coûteuse en temps.(1)

La prise en compte des problèmes de cohérence de l'information est limitée dans les SGBD classiques. En particulier l'insuffisance des modèles de données associés pour exprimer les contraintes d'intégrité a été montrée dans [ADI-83].

D'autre part, dès que les volumes d'informations à gérer augmentent, les temps de réponse deviennent trop grands pour les SGBD relationnels ainsi que l'a montré [SHE-83]. Ce type de problème risque d'être résolu grâce à l'évolution des performances du matériel et plus particulièrement aux recherches concernant les machines bases de données (se reporter à [KOR-83]).

On peut donc voir que les SGBD disponibles actuellement sont trop limités pour résoudre les problèmes de CAO. D'autre part, les modèles de données qui leur sont associés s'intéressent plus à la "représentation des choses" qu'à la "signification des choses". Nous allons maintenant étudier les recherches poursuivies pour pallier ces inconvénients.

(1) On trouve toutefois depuis peu de temps des SGBD relationnels offrant une gestion dynamique du schéma de la base (se référer à [CXP-86]).

Nous allons étudier maintenant un certain nombre d'outils (modèles de données et fonctionnalités) semblant mieux adaptés à la CAO (paragraphe III.1). La plupart de ces outils sont intégrés dans les projets de recherche de systèmes BD-CAO que nous présentons au paragraphe III.2.

III) Axes de recherches pour un SGBD en CAO : [CHO-83b], [DAR-83], [RIE-85]

III.1) Nouveaux modèles de données et nouvelles fonctionnalités :

Nous avons vu que les modèles de données existants sont trop pauvres pour pouvoir réaliser une bonne modélisation de la Base de Données Projet et que les fonctionnalités des SGBD classiques ne sont pas adaptées aux systèmes CAO.
Ces problèmes ne sont pas spécifiques à la CAO ; ils concernent d'autres domaines domme la synthèse de la parole, la bureautique, les applications médicales, la robotique, la cartographie, le génie logiciel, ...

A partir de ces constatations, des recherches ont été poursuivies pour une meilleure prise en compte des données de ces nouvelles applications: la CAO est un domaine privilégié de ces études.

Deux axes principaux de recherches ont été suivis:

- extension du modèle relationnel
- élaboration de nouveaux modèles dits sémantiques

Avant d'aborder quelques recherches correspondant à ces deux catégories, nous allons étudier des concepts communs que l'on peut y retrouver: il s'agit des modèles de données issus de l'Intelligence Artificielle et de la fonction de déduction étudiée en particulier dans le domaine des Bases de Données Déductives.

III.1.1) Modèles de Données issus de l'Intelligence Artificielle : [BD3-83], [BON-84], [CHO-81], [CHA-86], [DEM-86], [LAU-82], [MBD-86], [MIR-86], [NIL-82], [YAZ-85]

III.1.1.1) Les réseaux sémantiques :

a) Définition :

Un réseau sémantique est un graphe dont les noeuds sont des concepts et les arcs étiquetés des liens entre les concepts.

De plus, à chaque noeud est attaché un ensemble de propriétés qui permettent de définir le concept correspondant.

Il existe des noeuds qui ne sont pas simples et qui sont en fait des sous-réseaux sémantiques.

b) Différents types de liens utilisés :

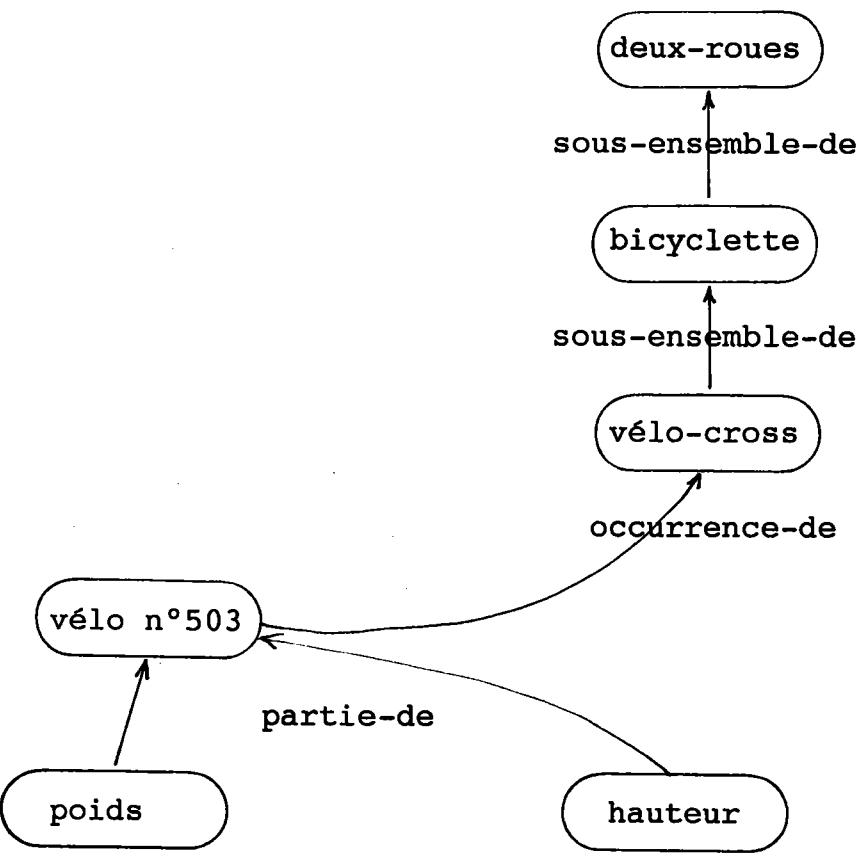
Un réseau sémantique possède en général au moins les 3 liens suivants:

- sous-ensemble de (généralisation) (cette relation est souvent notée AKO*, ou sorte-de)

- partie-de (agrégation)

- occurrence-de (classification) (cette relation est souvent notée ISA**, ou est-un)

Exemple: (cas de la bicyclette)



Il faudra distinguer les liens "occurrence-de" des liens "sous-ensemble-de" qui correspondent respectivement dans la théorie des ensembles aux opérateurs $\in$ et $\subset$ (ces liens permettent une représentation hiérarchique).

* de l'anglais a "kind of"
** de l'anglais "is a"

On pourra remarquer que l'enrichissement sémantique du modèle individu-association pourrait permettre d'obtenir un réseau sémantique.

c) Fonctionnalités des réseaux sémantiques:

Initialement les réseaux sémantiques ont été conçus par des psychologues en vue de représenter le sens des mots (chaque mot était un concept relié aux autres mots par différents types de liens). Aujourd'hui, les types de réseaux sémantiques varient selon l'objectif (le domaine à représenter) pour lequel il sont utilisés.

On peut citer un certain nombre de fonctionnalités communes aux différents réseaux sémantiques : héritage, transitivité, déduction, matching, etc...

La notion d'héritage permet de transmettre un certain nombre de propriétés d'un noeud à un autre noeud en fonction de la sémantique des liens reliant ces noeuds.

Ainsi si "deux-roues" a comme propriété de coûter moins de 3200 frs, le fait que "bicyclette" soit relié à "deux-roues" par la liaison "sous-ensemble-de" entraînera automatiquement que "bicyclette" coûte moins de 3200 frs, et par transitivité, il en sera de même pour "vélo-cross".
Dans ce cas, les caractéristiques des classes supérieures sont héritées par les classes inférieures.

Les réseaux sémantiques sont bien adaptés à la représentation des taxinomies (structure d'arbres dont les noeuds sont des concepts du domaine et dont les arcs représentent les relations entre noeuds du type "sous-ensemble-de" et "élément-de").

D'autre part les réseaux sémantiques permettent de créer des partitions c'est-à-dire de grouper des ensembles de noeuds et d'arcs dans des espaces spécifiant la portée des différentes relations.

d) Conclusion:

Les travaux sur les réseaux sémantiques ont d'avantage porté sur la finesse et la cohérence de représentation des concepts que visé des applications pratiques.

Leurs avantages sont la capacité de structuration des données et de représentation des liens procéduraux.

Les inconvénients essentiels des réseaux sémantiques sont :

- l'absence d'une terminologie standardisée (manque de support théorique).

- le nombre de noeuds et d'arcs augmente très vite d'où problème de clarté du modèle.

- ambiguïté dans la définition des noeuds et des concepts qui y sont attachés, de même pour la nature des liens (entre concepts, ou concepts-propriétés). Par exemple les noeuds-concepts permettent d'abord de représenter une classe, mais peuvent également devenir membre de la classe (c'est-à-dire le concept lui-même). La manipulation de ces éléments peut alors devenir ambigüe.

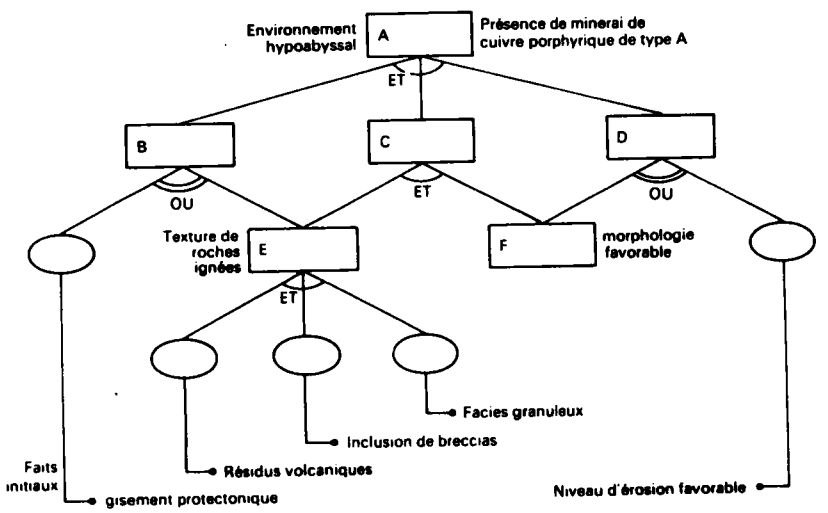
- capacité de raisonnement et de déduction limitées par rapport à d'autres modèles de représentation de l'Intelligence Artificielle (objets structurés ou schémas).

- les réseaux sémantiques sont devenus des représentations hybrides entre règles de production et objets.

Actuellement les réseaux sémantiques sont essentiellement utilisés dans l'analyse du langage naturel. Ils servent de support dans les systèmes de classification utilisés dans les systèmes experts. Quelques rares applications pratiques ont néanmoins été développées telle que PROSPECTOR (voir schéma qui suit).

Il y a actuellement relativement peu de travaux portant sur les réseaux sémantiques peut-être parce que les notions de réseaux sémantiques se retrouvent dans d'autres modèles de l'Intelligence Artificielle (schémas) ou du Génie Logiciel (modèles orientés objets).

Exemple de réseau sémantique : PROSPECTOR



PROSPECTOR est un système d'aide à la recherche minière. Le moteur d'inférences construit ici un graphe dont les sommets sont des propositions élémentaires. Une règle est un sous-ensemble d'arcs reliant des sommets: un sous-ensemble est marqué "ET" pour les prémisses d'une même règle; il est marqué "OU" pour des conclusions identiques données par plusieurs règles (voir figure précédente). Les déductions successives à partir des faits initiaux sont obtenues par propagation des pointeurs dans ce graphe.

III.1.1.2 Schémas

Sous le nom de schémas on regroupe les notions de prototypes, de frames et de scripts (schémas particuliers).

"Un schéma est une structure de données permettant de décrire un objet complexe généralisant la notion de réseau sémantique, prototype d'une classe d'objets; les objets sont structurés, ont des propriétés qui peuvent être des procédures, la structure est souvent de type réseau" ([BD3-83]).

Un schéma est une structure permettant de représenter aussi bien une situation qu'un objet physique. Nous allons ici nous intéresser plus particulièrement à la représentation des objets physiques : en fait une représentation d'un objet à l'aide d'un schéma n'en sera qu'une description partielle (d'où encore le nom de prototype).

Un objet est défini par un ensemble d'attributs, chaque attribut peut avoir lui-même plusieurs facettes. Une facette pourrait par exemple être un domaine (l'ensemble des valeurs permises) ou défaut (contient la valeur par défaut de l'attribut) ou si-besoin, si-ajout (voir exemple qui suit).

Les facettes décrivent la nature de l'attribut, mais peuvent aussi préciser comment il doit être utilisé ou calculé: les facettes d'un attribut permettent dans ce dernier cas de déclencher l'exécution de procédures, on les appellera attachements procéduraux (facettes si-besoin et si-ajout par exemple).
Les schémas sont en général organisés en réseaux (un attribut d'un prototype sera une référence à un autre prototype). On utilisera pour ces constructions des objets les notions de classification, généralisation et agrégation déjà rencontrées dans le cas des réseaux sémantiques.
On pourra ainsi passer de la classe d'objets la plus générale à la plus spécifique, les fils héritant des propriétés du ou des pères.

Dans un modèle basé autour de la notion de schéma, on distingue en général deux types d'objets: les prototypes qui représentent une classe d'objets et les instances qui sont les occurrences des objets d'une même classe.

Exemple de prototype : (tiré de [MBD-86])

```
<prototype>voiture
est-un        : véhicule
marque        : une marque
                (domaine = Renault, Peugeot, BMW, Ford)
                (défaut = Renault)
                (si-besoin = questionner "quelle est la marque?")
modèle        : un modèle
couleur       : une couleur
propriétaire  : une personne
                (si-ajout =  placer dans (ce propriétaire) voiture-
                self)

<instance>voiture # 2
est-un        : voiture
marque        : Peugeot
modèle        : 305
couleur       : beige
propriétaire  : Jean Dupont
```

Avant d'utiliser un schéma, il faut l'apparier (matching en anglais) à la situation courante pour laquelle on veut créer une instance : certains de ses attributs ont une valeur (dans ce cas il y aura vérification de cohérence) d'autres n'en ont pas (elles seront calculées par des procédures spécialisées).

En conclusion on peut constater que les schémas sont des modèles de données qui sont des prolongement des réseaux sémantiques et qui en ont les avantages et inconvénients.

Tout comme les réseaux sémantiques, les schémas ne sont qu'un outil général de représentation et apparemment, cette méthode a été peu utilisée. Par contre de nombreux concepts des schémas ont été repris, et ont abouti aux langages orientés objets et aux représentations orientées objets.

III.1.1.3) Représentations orientées objet et langages orientés objet :

a) Représentations orientées objet:
(Exemples tirés de "langages orientés-objet" [CRI-86])

La notion de représentations orientées objet (notée ROO) est issue des études concernant les réseaux sémantiques et les schémas. Selon les auteurs la représentation orientée objet s'appelle encore représentation centrée objet ou représentation en objets structurés.

Dans une ROO la notion fondamentale est l'objet. De même que dans les prototypes ou les réseaux sémantiques un objet est défini par un ensemble d'attributs (composés de facettes) qui peuvent eux-même être des références à d'autres objets. Une facette est définie de la même façon que dans les schémas: elle contiendra aussi bien des données que des procédures.

Un objet peut être schématisé comme une structure à 3 niveaux représentée en Lisp par:

```
(OBJET
     (Attribut-1
                (facette-1-1)
                (facette-1-2)
                 ............
                (facette-1-n))
     (Attribut-2)
     ............
     (Attribut-n))
```

Les différents types de facettes rencontrées en ROO sont en général les mêmes que celles rencontrées dans les schémas. En voici quelques unes:

* domaine et défaut (déjà vus précédemment)
  ----

* valeur (contient la valeur de l'attribut)
  ------

ex: (Pâte
        (couleur (valeur (jaune)))
        (forme (valeur (longue)))

* possibilité (exemple de valeurs possibles quelconques:
  ----
n'appartenant pas par exemple à un domaine ordonné)

ex: (Pâte
          ....
            (forme
                (possibilité (longue courte courbe))))

* intervalle (référence à un domaine ordonné, en général
  ----
numérique)

ex: (Pâte
          ......
               (temps-de-cuisson
                          (intervalle (10 20))))

Procédure (mêmes notions d'attachement procédural déjà
---------   vues pour les schémas).

La procédure calcule la valeur de l'attribut.
Le déclenchement de la procédure peut se faire dans les cas suivants:

- lecture de l'attribut si celui-ci n'a pas de valeur (procédure "si-besoin" déjà vue précédemment ou "à-établir").

- ajout ou modification de l'attribut auquel la procédure est associée: la procédure s'appellera un démon. Les procédures de ce type les plus courantes sont "si-ajout" (voir exemple déjà vu sur les schémas) et "si-modif".

ex:
```
(Pâte
    (couleur (valeur (jaune)))
    (matière (valeur (farine oeufs sel))
             (si-ajout (démon-couleur))))

(définir démon-couleur (c)
   (cond (( = c "épinard") (--> couleur valeur vert))
         (( = c "tomate")  (--> couleur valeur rouge))))
```

Dans cet exemple, le "démon-couleur" modifiera la valeur de l'attribut "couleur" de l'objet "pâte" selon la nouvelle "matière" entrée dans la composition: par exemple si on ajoute de la "tomate" dans "matière", alors la "couleur" deviendra "rouge".

Enfin la propriété d'héritage dans les ROO a les mêmes caractéristiques que pour les schémas entre autres un objet peut hériter simultanément de plusieurs objets génériques : cas de l'héritage multiple. L'héritage entre objets est obtenu grâce à des facettes qui sont des liens (notions de généralisation, classification déjà vues précédemment).

Aujourd'hui il existe un certain nombre de systèmes dits hybrides, où l'on combine la ROO et une représentation orientée règles (à base de règles de productions).
Cette représentation est très utilisée dans les systèmes experts déstinés aux diagnostic.

En conclusion, on pourra dire que les principes de la ROO sont utilisés dans de nombreux domaines: langages orientés objet, Bases de Données (maquettes de SGBD basés autour de la ROO comme par exemple Sembase [SEM-86]), représentation de la connaissance, Génie Logiciel (liens avec les types abstraits), CAO (tendance à utiliser une ROO pour la Base de Données Projet).

b) Langages orientés objets:

Les langages orientés objet sont des dérivés directs des systèmes de schémas, leur caractéristique principale est d'associer aux classes d'objets un ensemble de procédures appelées méthodes, dont bénéficieront tous les représentants de cette classe.

Un objet est décrit par une classe: cette classe permet de définir un moule pour créer les représentants physiques de l'objet appelés instances.

A la différence des schémas ou des ROO, les méthodes ne sont pas des facettes d'un attribut, mais sont rattachées à l'objet lui-même.

D'autre part, l'envoi de messages entre objets active les méthodes: l'activation d'une méthode équivaut à un appel de procédure.

exemple de représentation objet: (tiré de [MBD-86])

```
<classe> Compte

 champs      : nom
               numéro
               débit
               crédit
               solde

 méthodes    : dépots (V)
                        crédit  = crédit  + V
                          solde = solde + V

               retrait (V)
                          débit = débit + V
                          solde = solde - V


<instance> Compte # 1
 nom       : Dupont
 numéro    : 12345
 débit     : 2000
 crédit    : 4000
 solde     : 2000
```

Tout envoi de message mentionne l'objet auquel il est destiné, ainsi que le nom et les arguments de la méthode qui sera exécutée.

Par exemple, le message dépots (1000) adressé à l'instance Compte #1 permet de mettre à jour ce compte suite à un dépôt de 1000 F.

Cette méthode pourra s'écrire:

crédit, solde <----- envoyer dépots (1000) à compte #1

Il existe différents modes de transmission de messages : par exemple avec retour ou continuation.

S'il s'agit dans cet exemple d'une méthode de transmission avec retour, dès que la méthode est éxécutée, le résultat est retourné à l'expéditeur du message : dans ce cas le résultat est constitué des nouveaux crédits et soldes. Dans le cas d'une transmission avec continuation, le résultat est envoyé à un autre objet (qui est d'ailleurs appelé continuation).

On voit sur l'exemple précédent qu'un programme ne sera plus une suite d'instructions à exécuter, mais un ensemble d'objets dynamiques, autonomes et communicants.

Dans certains langages orientés objet, les structures de contrôle sont représentées sous forme d'objets et, par le biais des messages, les méthodes associées sont exécutées. Le résultat sera envoyé à l'utilisateur qui est lui-même considéré comme un objet.

Il existe à ce jour beaucoup de langages orientés objet. Le premier a été SMALLTALK (héritier de SIMULA). On peut noter dans les langages orientés objet une similitude avec les langages à types abstraits de données (remise en cause de la séparation programmes et données).

La version actuelle de SMALLTALK garantit un environnement de programmation très riche dont il offre une panoplie complète: écran bitmap, souris, multifenêtrage, générateur de menus, etc...

La contribution de LISP et puis ensuite de PROLOG a été importante à l'évolution de la programmation objet. En fait les recherches en Intelligence Artificielle ont joué un rôle moteur: leur tendance a été de développer des langages adaptés à chaque type de besoin plutôt que d'utiliser un langage existant.

Actuellement parmi les nombreux langages orientés (ou assimilés comme tel) on pourra aussi citer SMALLTALK ,LOOPS , KOOL, LRO2, LOGO, etc...

c) Représentations orientées objet et langages orientés objet

Les représentations orientées objet (ROO) et les langages orientés objet (LOO) ont beaucoup de points en commun : mécanismes d'héritage et d'instanciation qui impliquent une représentation sous forme de réseau.
Alors que dans les ROO l'attachement procédural se fait au niveau d'un attribut d'un objet, dans les LOO celui se fait pour l'objet complet (la méthode s'applique donc à tous ses attributs).
Certains langages orientés objet ont été construits à partir de représentations centrées objet.

III.1.2) Bases de données déductives :

L'apport de l'Intelligence Artificielle est importante dans le domaine des Bases de Données (mise en oeuvre d'interfaces pour utilisateurs non spécialisés, interrogation de BD en langue naturelle, conception de schémas de Bases de Données). Nous allons nous intéresser ici à la déduction dans les Bases de Données.

On peut dire qu'il existe actuellement 2 courants de recherche dans les Bases de Données Déductives.

Il s'agit de:

- partir des notions de Bases de Données en y intégrant les notions d'Intelligence Artificielle.

- partir des notions d'IA et utiliser les techniques Base de Données pour leurs qualités de stockage et de rapidité d'accès.

C'est la première approche que nous allons étudier ici.

a) Définitions:

Un SGBD déductif est un système qui permet la dérivation de nouvelles données à partir de celles physiquement stockées dans la base par l'intermédiaire d'axiomes logiques définissant des règles de déduction applicables.

En intelligence artificielle, les informations du monde réel peuvent être découpées en 2 catégories :

- les faits élémentaires

- les lois générales ou règles (formes logiques type hypothèses-conclusion)

b) Exemple :

Soit une Base de Données relationnelle avec une relation PERE (nom du père, nom du fils) qui permet de décrire le fait élémentaire "Bernard est le père de Jean-Pierre" et d'ajouter à cette base 1 règle logique exprimant que grand-père est père du père.

On aura donc :

- la relation : PERE (nom du père, nom du fils)

- la règle R : PERE (X,Y) $\wedge$ PERE (Y,Z) ---> GPERE (X,Z)

L'adjonction à la relation PERE de la règle GPERE permet d'étendre les possibilités d'interrogation de la base à des données déduites des données d'origine (repertoriées ici sous la forme de 2-uples) à l'aide des règles de déduction.

c) Différents types de règles :

On suppose qu'à un moment donné les définitions en extension des relations PERE et GPERE soient les suivantes :

PERE

nom du père	nom du fils
Bernard	Jean-Pierre
Jean-Pierre	Nicolas
Jean-Pierre	Michel

GPERE

nom du grand=père	nom du petit=fils
Bernard	Nicolas
Yvon	Walter

Si la règle R que nous avons définie ci-dessus est considérée comme une règle de cohérence, alors cet état de la Base de Données doit être rejeté comme non valide puisqu'il falsifie la règle.

En effet GPERE ne contient pas le couple <Bernard, Michel>. Dans ce cas on aura fait l'hypothèse que les n-uples qui satisfont à un moment donné à la relation GPERE sont exactement ceux apparaîssant dans son extension.

En levant cette hypothèse, on utilisera cette règle comme une règle de déduction.
Un des problèmes des Bases de Données Déductives est de choisir pour une règle générale, son utilisation comme règle de déduction ou comme contrainte d'intégrité.

La notion de règle de déduction peut-être affiné en :

- règle de dérivation.
- règle de génération.

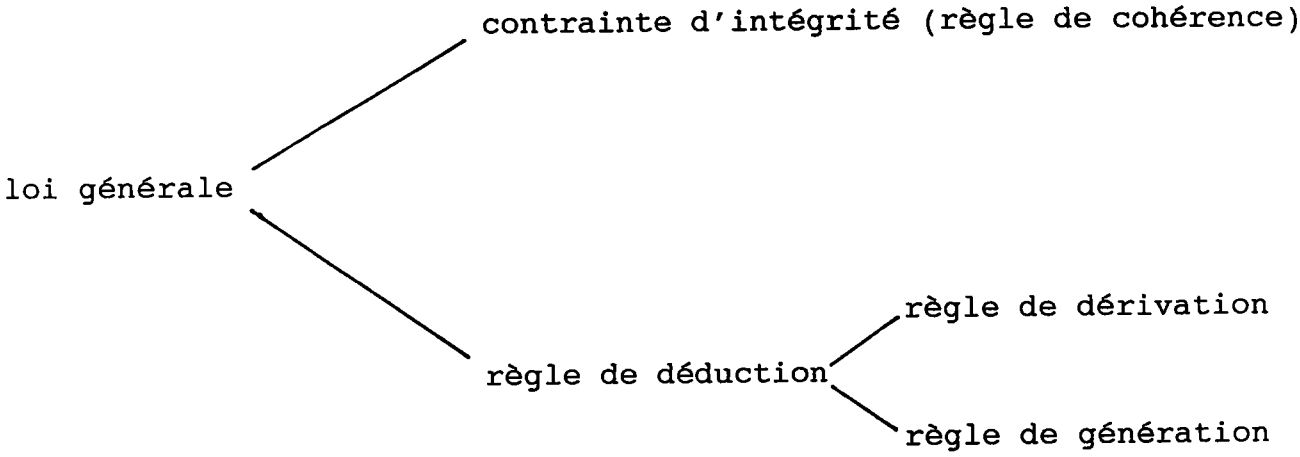
Afin d'illustrer la différence entre ces 2 types de règles, revenons à l'exemple précédent.

Si la règle R est exploitée en <u>règle de dérivation</u>, l'ajout d'une nouvelle donnée concernant la relation PERE (ex: insertion du couple <Jean-Pierre, Claude>) n'aura aucun effet direct sur l'extension de la relation GPERE. Par contre toute question faisant intervenir la relation GPERE fera appel à un mécanisme déductif permettant d'accéder aux informations déductibles concernant cette relation (dans l'exemple, il s'agit du couple <Bernard, Claude>.

Si la règle R est exploitée en <u>règle de génération</u>, l'insertion du couple <Jean-Pierre, Claude>, impliquera la génération automatique du couple <Bernard, Claude> dans l'extension de la relation GPERE. Dans ce cas toute évaluation de questions à la Base de Données se fera par rapport à l'extension des relations concernées (comme dans SGBD classique).

On pourrait ainsi voir les règles de génération comme un moyen de restaurer la cohérence.

En résumé, on peut représenter les différents types de lois générales de la façon suivante :



d) Prise en compte des règles par les SGBD actuels :

On pourra remarquer que les fonctionnalités offertes par un système déductif sont assez proches de celles offertes par les SGBD relationnels possédant un mécanisme de vues élaboré (SYSTEM-R par exemple).

Ces SGBD permettent de déduire des faits nouveaux (quand il s'agit d'une déduction triviale), mais ne peuvent déduire des faits liés à des définitions récursives de prédicats (dans l'exemple précédent on pourraît définir un prédicat récursif "ANCETRE").

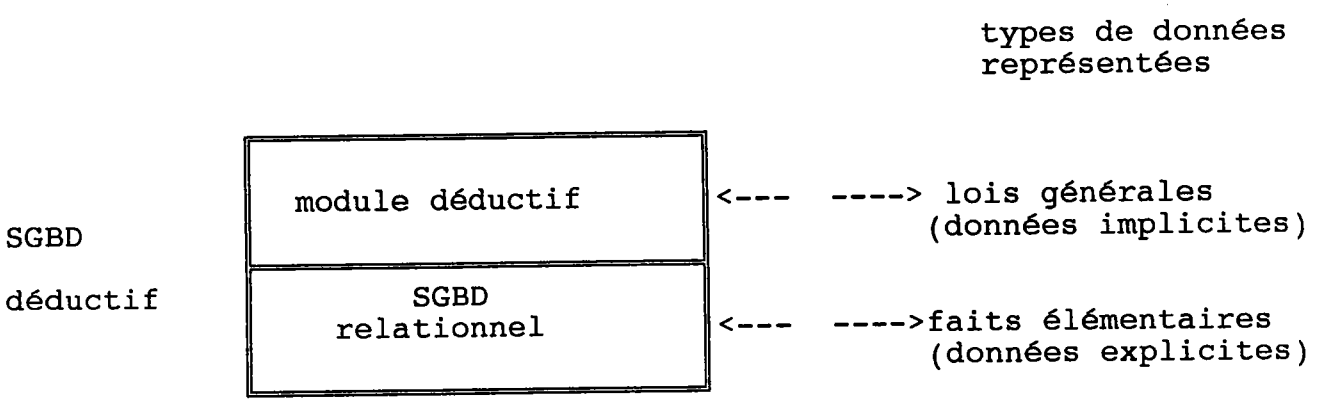
D'autre part dès que le problème à résoudre n'est plus élémentaire, les critères de sélection à fournir par l'administrateur de Base de Données deviennent rapidement complexes.

Les SGBD actuels ne permettent de prendre en compte que certains problèmes déductifs : il leur manque la possibilité d'exprimer un problème sous forme logique (plus proche de notre manière de penser) et donc de se libérer ainsi des contraintes d'implémentation. Ce sont les outils de l'Intelligence Artificielle, utilisés dans les systèmes déductifs qui permettent d'obtenir ce niveau d'abstraction supplémentaire.

e) Systèmes déductifs actuels :

On sait gérer les faits élémentaires par les SGBD classiques (en l'occurence relationnels) et de même les lois sur ces faits par des langages logiques (ex : PROLOG, LISP, PROLISP, etc...).

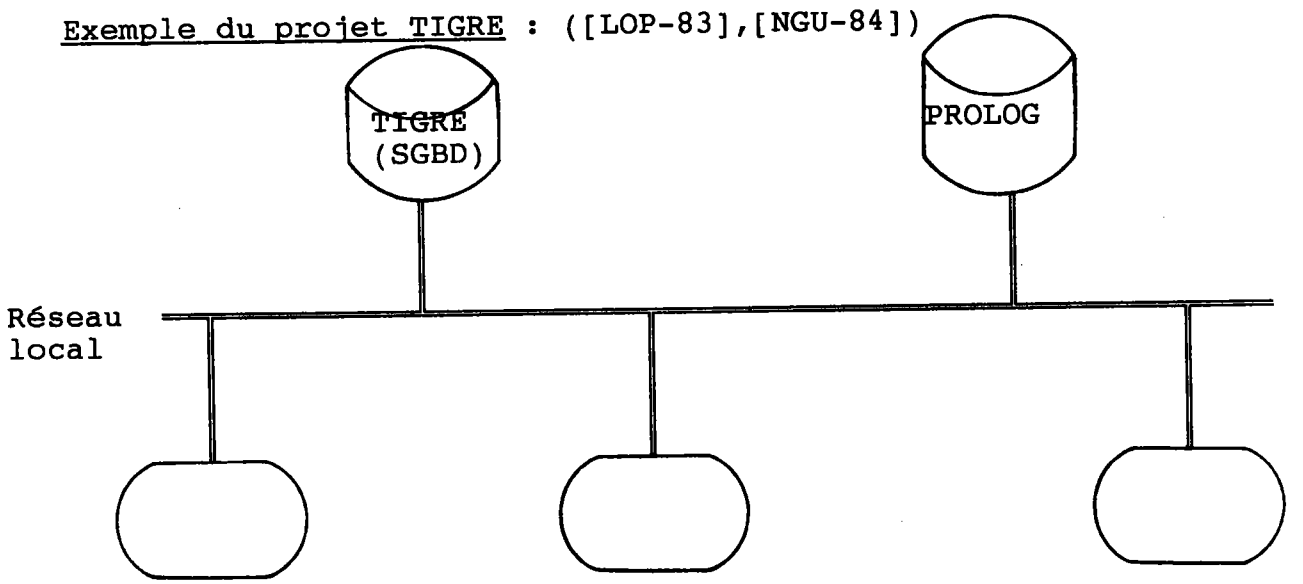
On peut en déduire l'architecture générale suivante :



Des propositions de différentes architectures d'implémentation des Bases de Données Déductives sont faites dans [DEM-86].

De nombreuses recherches sont actuellement menées sur la coopération entre Bases de Données relationnelles et le langage PROLOG (par exemple le projet TIGRE).

Les langages de programmation logique d'une part apportent leur puissance d'expression et leur souplesse, les SGBD d'autres part offrent des moyens sophistiqués de consultation de grandes masses d'informations tout en assurant leur cohérence.



TIGRE offre des outils de stockage, d'accés et de manipulation d'objets volumineux et/ou complexes (par exemple des documents de description de Circuits Intégrés).

PROLOG offre des outils de définition, de contrôle, de manipulation aussi bien des objets que des traitements qu'ils subissent et des contraintes dynamiques qu'ils doivent respecter.

Remarques :

- d'un point de vue Base de Données, PROLOG peut apparaître comme un langage hôte.

- les langages utilisés dans les 2 modules d'un système déductif ont la même structure (langage prédicatif pour le SGBD relationnel, et langage orienté "calcul de prédicats" pour le système déductif) celà permettra une meilleure osmose entre eux. Dans ce cas ce SGBD apparaîtra dans le système déductif comme un module de l'interpréteur de logique, spécialisé dans la manipulation des faits élémentaires.

- un certain nombre d'études ont été menées dans le sens de la coopération entre un système CAO et un système expert: on peut citer par exemple [REH-85].

f) Quelques problèmes posés par les Bases de Données Déductives :

Nous nous contenterons ici de citer les principaux problèmes existant actuellement dans ce domaine qui sont :

- Le problème de l'arrêt :
Ce problème est le suivant: étant donné un ensemble de règles, existe-t-il une procédure qui se termine en un temps fini, qui permette de déduire n'importe quel fait déductible à l'aide de ces règles. Ce problème a été résolu d'un point de vue théorique, mais ne fonctionne pas toujours en pratique.

- Les problème de performances :
Une des stratégies utilisées est de considérer que le temps de déduction sera d'autant plus court que l'on aura évité de déduire des faits inutiles: par exemple pour déduire les ancêtres de x, si on commence par déduire tous les ancêtres de toute le monde, puis qu'on sélectionne ceux de x alors on aura déduit beaucoup de faits inutiles.

- Le problème de complétude :
Il faut démontrer pour chaque stratégie utilisée, qu'elle permet d'obtenir tous les faits déductibles, même si elle s'arrête en un temps fini.

- Le problème de cohérence :
Comment déterminer si une règle donnée doit être utilisée en cohérence ou en déduction. Une solution a été proposée dans [YAZ-85].

- Le problème d'optimisation des réponses :
Dans le cas où il y a un grand nombre de réponses possibles, dont certaines sont plus ou moins intéressantes, il faut pouvoir définir des critères pour déterminer lesquelles sont intéressantes.

- Le traitement d'informations incomplètes :
Dans l'exemple précédent supposons que l'on veuille insérer le n-uple <François,---> dans la relation PERE, est-ce que cela signifie que François n'a pas de fils, ou bien qu'il a un fils et qu'on ne le connaît pas. On tombera sur un paradoxe dans le cas où l'on demandera les n-uples pour lesquels le fils est, ou n'est pas Pierre car dans ce cas on devrait obtenir tous les n-uples.

g) Conclusion :

Par rapport aux SGBD traditionnels, les méthodes déductives permettent en plus de manipuler des règles ou encore des faits incomplets. Ceci pose bien-sûr de nombreux problèmes théoriques, d'architecture, de performances et à ce jour il n'existe pas à notre connaissance de systèmes déductifs opérationnels.

En conclusion, si l'on veut concevoir aujourd'hui un système CAO qui fonctionne on ne pourra pas utiliser directement les méthodes déductives, mais par contre comme ce domaine progresse rapidement on risque d'avoir des retombées très enrichissantes dans les systèmes de CAO.

III.2) Quelques études actuelles :

Nous avons vu au paragraphe III.1 qu'il existait aujourd'hui deux grands domaines de recherches pour la conception de systèmes BD-CAO.

Il s'agit de :

- l'extension du modèle relationnel.
- l'élaboration de modèles sémantiques.

III.2.1) Extension du modèle relationnel :
[AUT-85], [DAR-83], [DAT-86], [EDM-83], [GRA-78], [GUT-82], [GUT-84], [HAR-84], [HAS-81]

* Une première approche est celle par les types abstraits qui permettent de prendre en compte de nouveaux types de données. L'utilisateur peut définir ses propres types de données ainsi que de nouveaux opérateurs ([STO-83]). Cette proposition a été implémentée sur INGRES.

* Le modèle relationnel rend difficile la modèlisation d'objets complexes du fait de l'éclatement de l'information en nombreuses relations. Une autre approche a consisté à prendre en compte un objet complexe comme un ensemble de tuples formant un tout: c'est à dire que plusieurs tuples de relations différentes peuvent être vus comme une entité unique ([LOR-83]). Cette proposition a été implémentée sur SYSTEM R.

III.2.2) Elaboration de modèles sémantiques :

Il y a actuellement 2 tendances de recherches de nouveaux modèles de données pouvant être appliquées en CAO.

- Les modèles généralisés (appelés encore multi-médias)

- Les modèles spécifiques (à la CAO, ou même, d'une façon plus précise, à l'architecture, aux circuits VLSI, ...)

* En ce qui concerne les modèles généralisés, ils sont conçus pour pouvoir prendre en compte de nouveaux types d'informations afin de pouvoir étendre les bases de données à d'autres domaines que la gestion. Voici quelques exemples de domaines concernés:

. La CAO bien-sûr où les informations sont des textes, des dessins etc....

. La bureautique où l'on manipule des documents.

. Le génie-logiciel où l'on manipule des programmes.

. La synthèse de la parole où l'on manipule des voix digitalisées.

On pourra signaler aussi l'économie, la cartographie, la recherche documentaire, le traitement d'images, etc....
L'objectif de la recherche de modèles généralisés est d'aboutir à une rigueur de formalisation proche de celle mise en oeuvre autour du modèle relationnel.

* Un raisonnement opposé considère que les nouveaux domaines spécifiés précédemment ont des besoins en gestion des données trop différents pour être pris en compte par un modèle de données unique. Cette tendance est donc orientée "application": alors que la tendance modèles généralisés est poursuivie par les chercheurs, les modèles orientés application ont leurs adeptes parmi les informaticiens d'un domaine d'application donné.

Les modèles sémantiques ont une approche modèle entité-association ou modèle orienté objet. Dans les deux cas on utilisera des mécanismes d'agrégation, de spécialisation (ou de généralisation), d'association, que nous avons vu au III.1.1.

On peut donner quelques exemples de modèles sémantiques développés pour des applications CAO:

TIGRE : modèle généralisé ([NGU-84])

FLOREAL : modèle orienté CAO ([CHO-83a])

ARLANG : modèle orienté ARCHITECTURE en CAO ([AUT-82])

III.2.3) Quelques axes de recherche :

Comme nous l'avons déjà signalé précédemment, la majorité des recherches actuelles sont basées autour de la Base de Données Projet. On peut classer ces recherches par rapport au modèle de données utilisé, mais on peut définir d'autres critères de classification comme le domaine de conception, la nature des informations modélisées ou encore les fonctionnalités des systèmes. En voici quelques exemples:

- Parmi les domaines de conception les plus fréquemment abordés on peut citer les VLSI ([BAT-85], [CHU-83], [LEC-84], [RIE-86]) et l'architecture ([AUT-85], [PHI-79]).

- Dans de nombreuses études, on utilise les modèles de l'Intelligence Artificielle ([WON-77]): on s'intéresse aussi de plus en plus à la modélisation de la Base de Connaissances ([BIL-87], [CAD-85], [FAU-87]).

- Parmi les fonctionnalités des systèmes BD-CAO intégrées dans les systèmes BD-CAO, l'une de celles que l'on retrouve le plus souvent est la gestion des versions et alternatives ([BAT-85], [KAT-83], [NGU-86]).

III.2.4) Exemples de modèles de données utilisés dans des systèmes BD-CAO:

Nous allons étudier rapidement trois modèles de données . Il s'agit de modèles de données utilisés dans des projets que l'on appellera par les noms suivants :

- **FLOREAL**
- **DARONNAT**
- **RIEU**

a) FLOREAL : [CHO-83a]

Ce modèle de données est de type Entité-Relation étendu. On y définit les concepts suivants:

- la classe : collection de valeurs possédant une structure identique et satisfaisant au même ensemble de contraintes (on différenciera pour une classe donnée les propriétés structurelles des propriétés non structurelles appelées contraintes). On distingue 2 types de classes : les classes statiques et les classes dynamiques.

- les relations : permettent de lier ou de contraindre des éléments d'une même classe ou de classes différentes (notion équivalente à celle de relation dans le modèle E-R).

Dans ce modèle on fait une distinction nette entre classes et occurences de classes, ainsi qu'entre relations et occurences de relations.
Un objet est une classe d'éléments en cours d'élaboration, et une occurence d'objet est un représentant de cette classe.
D'autre part ce modèle de données intègre les concepts suivants:

- l'agrégation, la généralisation, l'association.

- l'utilisation de la notion de fonctions pour prendre en compte les contraintes sur les objets (contraintes fonctionnelles et relationnelles), entre objets (testeurs, générateurs) ou sur un ensemble d'objets (hypothèses). Ces concepts s'expriment sous la forme de règles logiques.

Le modèle de données est un modèle dynamique (grâce aux concepts d'agrégation et de généralisation), il a été crée pour modéliser la Base de Données Projet. Toutefois il permet de gérer certaines connaissances telles que les hypothèses qui représentent certaines contraintes du cahier des charges.

L'accent a été mis sur la notion de vue qui permet de donner des définitions différentes d'un même objet.

D'autre part une certaine forme de déduction est rendue possible par le biais des propriétés calculées et de dérivation de classes.
Une implantation du modèle a été réalisée en PROLISP.

b) DARONNAT : [DAR-83]

Le modèle de données étudié par DARONNAT est un modèle de type relationnel enrichi.
Les différentes données d'un processus de conception peuvent être découpées en types pouvant être regroupés en hiérarchies fonctionnelles et en hiérarchie structurelles.

Une hiérarchie structurelle regroupe les types structurés suivants :

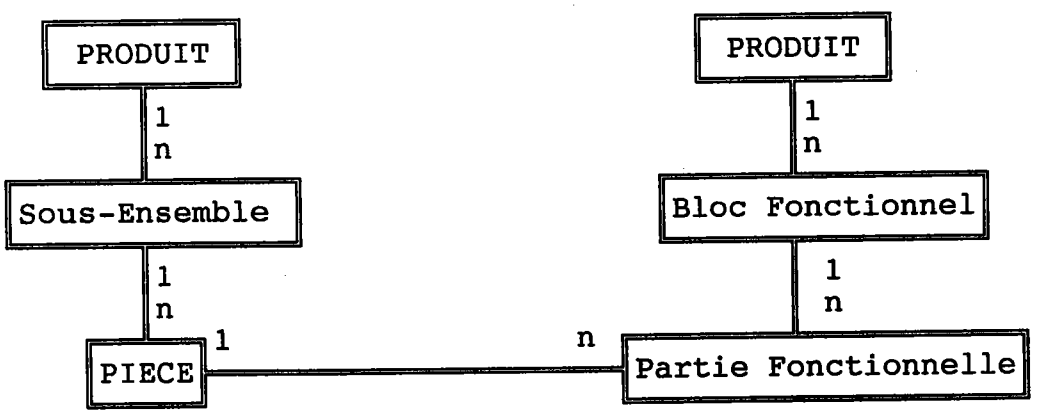
- le produit (à concevoir).
- les sous-ensembles structurels du produit.
- les pièces (objets structurels élémentaires).

Une hiérarchie fonctionnelle regroupe les types fonctionnels de données suivants :

- les blocs fonctionnels qui regroupent chacun les parties fonctionnelles participant à la même fonction principale.
- les parties fonctionnelles qui ont une fonction élémentaire précise.

La hiérarchie structurelle correspond à la structure physique du produit, elle est unique. Par contre on peut avoir plusieurs hiérarchies fonctionnelles pour un même produit permettant de prendre en compte plusieurs vues de l'objet (la cohérence entre ces hiérarchies fonctionnelles étant obtenue grâce à une hiérarchie structurelle unique).

On pourra représenter ces notions par la figure suivante :



Hiérarchie Structurelle Hiérarchie Fonctionnelle

Pour la modélisation des données les concepts suivants sont présentés :

- attribut simple : on y définit les notions d'entités, d'attributs simples et d'attributs clés (ce sont les mêmes notions que dans les modèles E-R).

- attribut composé : réunion de 2 ou plusieurs attributs simples ou composés. Par exemple "dimension" est un attribut composé comprenant les attributs "longueur", "largeur" et "hauteur".

- attribut complexe : attribut composé possédant une clé primaire.

- entité caractéristique : permet de définir des liens entre entités.

Ces différentes notions sont déjà existantes dans le modèle relationnel, on leur rajoute deux concepts non représentables ou difficilement représentables dans le modèle relationnel. Ces concepts sont les suivants:

- attribut caractéristique : possède les propriétés d'un attribut complexe et peut en outre participer à des liens avec d'autres attributs caractéristiques.

- généralisation.

Le modèle de données DARONNAT a été conçu pour prendre en compte une Base de Données Projet en électro-mécanique avec en plus certains éléments de connaissance comme les règles d'assemblage. La déduction est rendue possible grâce à des attributs déduits.

Une implantation a été réalisée en PROLOG.

c) RIEU : [RIE-85]

Le modèle étudié par RIEU est un modèle orienté objet : le domaine de conception étant l'électronique en particulier.
Les concepts de ce modèle sont les suivants :

- les ensembles : définis par un nom, une suite de caractéristiques, des contraintes et des liens. Il existe comme dans FLOREAL des ensembles structurés et des ensembles non struturés.

Par exemple :

```
DEF-E    Point
  ‖ X : Reel
  ‖ Y : Reel
FDEF-E
```

```
DEF-E Segment
    ‖   org : Point
    ‖   ext : Point
FDEF-E
```

définit l'ensemble des points et des segments (qui sont des ensembles structurés): X ,Y, org, ext sont des caractéristiques et Reel un ensemble de valeurs (Reel est un ensemble non structuré).

Lors de la création d'une occurence d'ensemble toute caractéristique devra être instanciée.

- les fonctions : utilisées dans la définition des contraintes et des liens sur les caractéristiques des ensembles. Il existe 2 types de fonctions; il s'agit des fonctions de base et des fonctions génériques (calculs particuliers du concepteur ou du domaine de conception). Les fonctions ne sont jamais activées par l'utilisateur.

- Les contraintes et les liens : permettent la gestion de la cohérence des structures et des liens entre objets. Une contrainte définit une relation entre les caractéristiques d'un ensemble. Par exemple "notconf(org,ext)" exprime par le biais de la fonction booléenne notconf que l'origine et l'extrémité d'un segment ne peuvent être confondus.
Un lien permet d'imposer une valeur à une caractéristique.

En plus on utilise des concepts comme l'agrégation et la spécialisation. Ce modèle de données est prévu pour représenter aussi bien la BDP que la BDC, ce qui doit favoriser les échanges entre ces bases.

Au niveau de la réalisation, un prototype de système BD-CAO a été réalisé en PROLOG: à partir de là, la mise en oeuvre d'une coopération entre un SGBD et un système basé sur la programmation logique a été prévue.

Nous présentons maintenant le projet SACADO auquel appartient notre étude. Dans un premier temps, nous décrirons ce projet puis nous verrons en particulier son lien avec un sytème BD-CAO.

IV) Le système SACADO : [GAR-86b]

Ce système est en cours de développement au LRIM (Laboratoire de Recherche en Informatique de Metz). SACADO signifie Système Adaptatif de Conception Assistée et de Développement par Ordinateur.

Les objectifs de SACADO sont les suivants :

- convivialité : obtenir un dialogue aussi souple que possible avec un minimum de contraintes pour l'utilisateur final (menus dynamiques, multi-fenêtrage,...).

- ouverture : le système doit s'adapter aux problèmes des entreprises et à leur façon de les résoudre. En particulier utilisation d'un langage de haut niveau comme PASCAL.

- mise en oeuvre de solutions techniques nouvelles : prise en compte des évolutions logicielles (arbres octaux, algorithmes,...) et matérielles (implantation sur micro-ordinateurs, adaptabilité à différents postes de travail).

IV.1) Les données :

Tout processus de CFAO peut être considéré comme étant un processus informationnel. Les différents types d'informations gérées par le système sont :

* informations liées au projet :

- cahier des charges des projets (règles et objectifs de chaque projet)

- modèles des projets (représentation informatique des projets en cours ou antérieurs).

- connaissances (historiques de projets, connaissances expertes ou même systèmes experts).

* informations communes :

- catalogues de pièces ou d'ensembles

- connaisances de l'entreprise : savoir-faire, éléments techniques (capacité des machines, coûts,...).

IV.2) Architecture générale de SACADO :

Un système de CFAO doit pouvoir être utilisé aussi bien par des personnes individuellement que par des groupes de personnes. D'autre part le système doit permettre la gestion simultanée de plusieurs projets.

IV.2.1. Architecture pour un projet donné :

Pour un projet donné, on considérera 2 sous-structures:

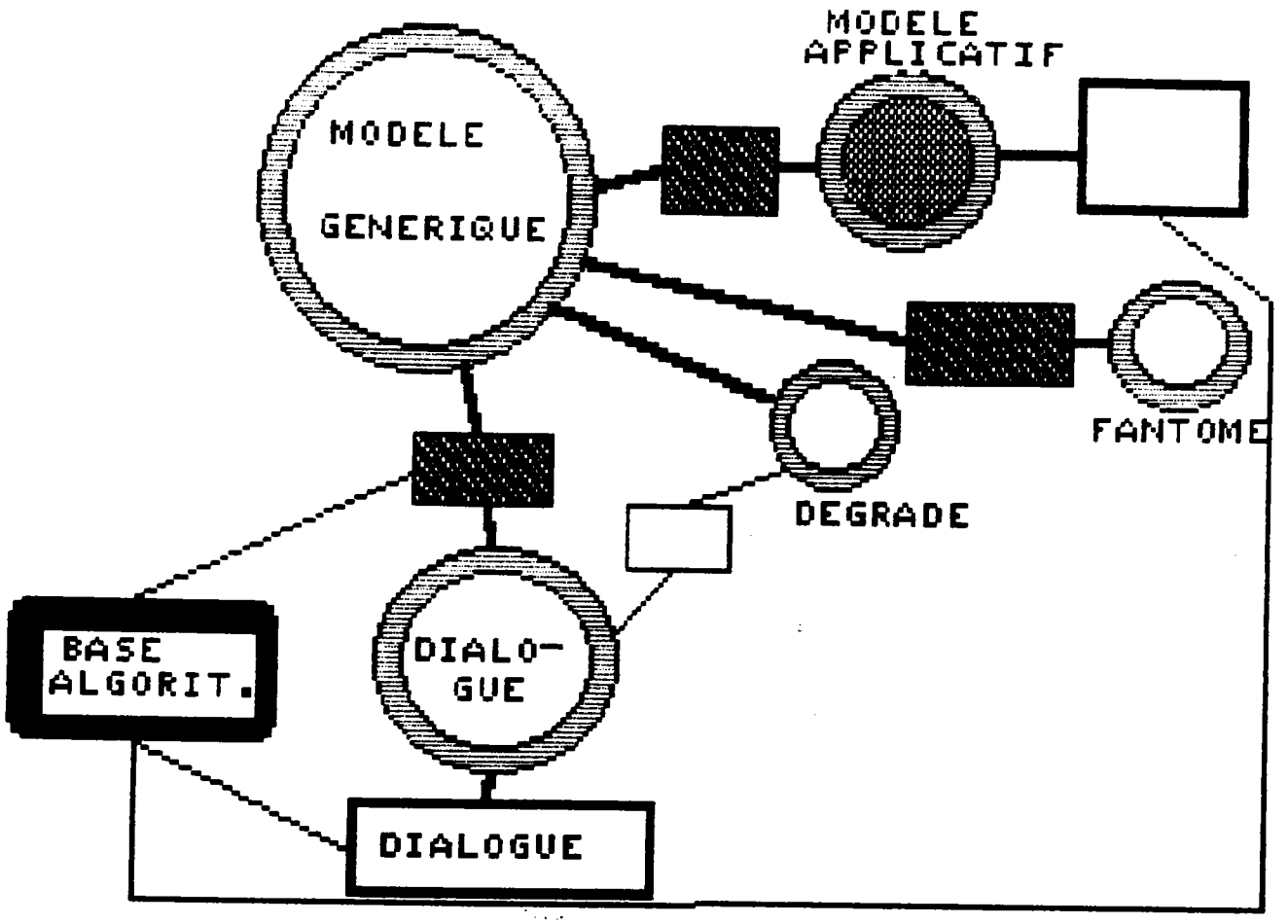
- les informations et les processus de gestion du projet: (contraintes générales du projet), elles sont initialisées par un chef de projet. Elles sont constituées des renseignements généraux, des contraintes du projet (fonction à réaliser, règles de gestion des versions, règles temporelles,...) et d'informations permettant de gérer les droits et rôles des personnes individuelles et des groupes.

- les informations et processus du projet (spécifiques à un projet donné).

La première sous-structure devra exister lors de la conception du premier objet, tandis que la deuxième sous-structure sera inexistante en début de conception mais elle sera alors créé et évoluera pendant toute la durée de cette phase.

Pour un projet donné, l'architecture logicielle du système SACADO, sera composée à partir des éléments suivants :
(voir aussi schéma)

- Le modèle générique : ce modèle est le coeur du système, il contient toutes les "informations" concernant un projet (données, algorithmes, connaissances).
On désignera par MODELE dans SACADO non seulement la représentation des informations, mais aussi les opérateurs permettant l'accès à ces informations.



LEGENDE:

MODELE ET GESTIONNAIRE

PROCESSUS DE LOCALISATION-GLOBALISATION

PROCESSUS APPLICATIF

SACADO : Architecture générale des modèles

- les modèles applicatifs :
Ce sont des vues externes du modèle générique. Pour chaque application, il faudra créer un modèle applicatif. Par exemple on peut citer le modèle de dialogue.

- les processus de localisation et de globalisation :
Permettent le passage du modèle générique à un modèle applicatif et inversement.

- les bases d'algorithmes :
Elles sont constituées de programmes indépendants des modèles, donc accessibles par tout processus.

- les processus applicatifs :
Liés au modèles applicatifs, par exemple on pourra parler du processus de dialogue.

- les modèles dégradés :
Ils sont déduits d'un processus de localisation, mais ne dépendront plus du modèle générique: ainsi aucune modification du modèle générique n'aura de répercussion sur les modèles dégradés.

IV.2.2) Relations inter-projet :

On distinguera 2 types de relations inter-projets : les relations ponctuelles et les relations permanentes. Les différences entre ces 2 catégories de relations vont être mises en évidence par l'exemple qui suit.
Supposons que pour 2 projets A et B, des éléments du projet A soient utilisés dans B. On dira que la relation entre A et B est ponctuelle si aucune modification de A n'affectera B, et cette relation sera permanente dans le cas contraire.

IV.3) Modèle générique :

Les informations contenues par ce modèle sont les suivantes:

- les données : le modèle géométrique en particulier (tridimensionnel solide).

- les connaissances : elles peuvent avoir été explicitement définies par l'utilisateur ou déduites par le système.

- les algorithmes : permettent de gérer la connaissance (moteurs,...) ou de représenter une information (sous forme paramétrée par exemple).

IV.3.1. Les modèles :

Un objet donné peut avoir plusieurs structures possibles (géométrique, assemblage, de fabrication). Ainsi le modèle générique devra pouvoir posséder plusieurs modèles structurels : le modèle fonctionnel en est un cas particulier.

A chaque modèle on associera un ensemble de règles de cohérence. Ces règles de cohérence pourront être :

- fortes : c'est à dire obligatoirement vérifiées

- faibles : leur violation impliquerait l'envoi d'un message.

- immédiates : dès qu'un élément sera crée, supprimé ou modifié les règles devront être vérifiées.

- différées :vérification lors de la constitution d'une version.

L'application de ces règles peut être ponctuelle (on ne conserve que le résultat) ou permanente (règles réappliquées chaque fois que c'est utile).

IV.3.2) Gestion des versions et Bases de Données :

Afin de pouvoir étudier des solutions en parallèle, et de pouvoir revenir en arrière aussi, il est indispensable de posséder un outil de gestion de versions.

Une solution pourra être sauvegardée à tout moment sous forme de version même si les ensembles qui la composent ne sont pas complets : on parle dans ce cas de version de travail. Dans le cas où les sous ensembles composant une version sont complets et où leurs contraintes fortes sont vérifiées, on pourra dire que la version est valide.

Toute version qu'elle soit valide ou de travail sera mémorisée dans un Base de Donnée. On distingue deux Bases de Données :

- la BD Projets : structurée en projets, elle contient tous les renseignements concernant un projet donné (règles, contraintes générales...) ainsi que les différentes versions le concernant (versions de travail et valides).

- la BD Connaissances : cette base contient l'ensemble des connaissances de l'entreprise. Elle contient différents types de connaissances (catalogues, savoir-faire, données techniques). Sa structure ne peut donc être décrite de manière unique.

IV.4) Modèles de dialogue :

Parmi les modèles applicatifs nous décrirons ici uniquement le modèle applicatif lié à l'application dialogue.
On peut décomposer les actions de base du dialogue en actions primaires telles que l'affichage, l'effacement, la désignation, la valuation, etc... La plupart de ces primitives de dialogue ont besoin de disposer d'informations contenues dans le modèle générique, mais ces informations sont sous forme non adaptée au dialogue.

On fera donc appel à un processus de localisation pour créer une vue particulière du modèle générique, adaptée au dialogue (il s'agit ici d'une vue au sens Bases de Données).

Pour un modèle générique donné il peut exister simultanément plusieurs modèles de dialogue (par exemple un modèle 2D et un modèle s'appuyant sur des facettes planes).

Un processus de localisation a pour rôle d'épurer, de restructurer les informations du modèle générique, mais parfois aussi de recalculer de nouvelles informations.

A partir du moment où un modèle de dialogue est bien défini, le processus de localisation est facile à décrire, par contre le processus de globalisation pose des problèmes suite à certaines opérations sur le modèle de dialogue. D'autre part il faut définir aussi à quel moment lancer le processus de globalisation : une solution raisonnable serait de faire la mise à jour du modèle générique périodiquement avec utilisation d'un journal.

PARTIE 2 : LE MODELE

Dans cette partie, nous allons définir le MODELE choisi pour réaliser notre système BD-CAO. Rappelons que comme nous l'avons déjà décrit dans SACADO (partie 1, paragraphe IV), qu'un MODELE est composé d'un modèle de données, et de traitements qui s'y appliquent.

Ce MODELE est présenté dans le paragraphe I : nous décrivons d'abord ses fonctionnalités (paragraphe I.1) puis le modèle de données (paragraphe I.2) issu des frames et enfin les traitements (paragraphe I.3).

Ensuite dans le paragraphe II, nous allons décrire la structure de données associée à ce modèle de données ainsi que les liens existants entre modèle et structure.

Les notions d'attachement et de déclenchement procéduraux qui sont des concepts essentiels dans les frames sont présentés au paragraphe III.

Enfin le MODELE sera situé par rapport au projet SACADO dans le paragraphe IV.

I) Description du Modèle

I.1) Présentation des fonctionnalités du Modèle :

Avant de faire une description des principales fonctionnalités de notre Modèle, nous allons définir les notions de modèle du projet et d'objets.

Un modèle du projet correspond à une image du projet à un instant donné : il sera constitué d'un ensemble d'objets que le ou les concepteurs auront fait évoluer depuis le début de la phase de conception. Chaque objet peut être lui-même découpé en sous-objets (le modèle du projet est lui-même un objet). Un objet regroupe à la fois des informations structurelles des contraintes et des valeurs (occurences) correspondantes.

I.1.1) Définition et manipulation simultanées des données :

Dans une même session de travail il devra être possible de créer, modifier, supprimer aussi bien les structures que les données.

I.1.2) Conception ascendante et descendante :

Le Modèle doit pouvoir permettre de créer un objet à partir d'objets déjà existants: c'est le cas de la conception ascendante.

De même pour un objet donné, pouvoir le décomposer en "sous -objets" qui eux-mêmes pourront être décomposés en "sous-objets"... : c'est le cas de la conception descendante. Les notions de conception ascendantes et descendantes permettent une conception moins rigide.

Il est possible bien-sûr de créer aussi des objets isolés c'est-à-dire non liés à d'autres objets.

I.1.3) Création imbriquée :

Si pendant la phase de création d'un objet A, on a besoin de faire référence à une entité B jusque là inexistante, il faut avoir la possibilité de créer alors B et de pouvoir ensuite terminer la phase de création de A.

I.1.4) Héritage et déduction :

On peut considérer la notion de déduction à deux niveaux : entre objets et pour les propriétés d'un objet.
Dans le premier cas, il s'agit du concept de déduction par héritage: un objet héritera des propriétés d'autres objets auquel il est lié par le biais de liens d'héritages. Toute propriété héritée d'un objet ne sera pas explicitée en tant que propriété lors de la création de cet objet, mais aura une existence issue d'un lien d'héritage.

Dans le deuxième cas, une propriété d'un objet pourra être définie par rapport à d'autres propriétés de cet objet ou d'autres objets: ses occurences seront déduites des occurences des autres propriétés.

I.1.5) Objets incomplets et objets incohérents :

Tout objet CAO en cours de conception va évoluer aussi bien en structure qu'en valeurs par affinages successifs.
Le Modèle doit donc permettre de modifier la structure de tout objet (celà reviendra à créer, supprimer ou encore modifier des propriétés de cet objet).

Par ailleurs pour un objet donné il doit être possible de ne pas instancier toutes ses propriétés.
Donc le Modèle doit pouvoir gérer des objets incomplets aussi bien en structure qu'en valeurs. On peut noter que l'existence d'un objet incomplet impliquera l'incomplétude d'autres objets qui lui seront liés : en particulier l'objet modèle du projet sera incomplet pendant toute la phase de conception. Pour créer un objet il suffira de le nommer sans avoir à définir d'autres propriétés.

La définition d'un objet comprend un certain nombre de contraintes sur cet objet. Etant donné le caractère essai-erreur de toute phase de conception, il est évident que tout objet est incohérent si l'ensemble des valeurs associées à cet objet viole au moins une de ses contraintes.

Certaines contraintes se comportent en règles de cohérence fortes (par exemple un poids d'un objet doit être positif) c'est-à-dire qui devront être vérifiées instantanément. Les autres contraintes correspondront à des règles de cohérence faibles : leur non-vérification n'impliquera que l'émission d'un message et la vérification de ces contraintes se fera à des moments bien précis (à la demande explicite du concepteur, au moment de la validation du modèle du projet,...).

Tout objet CAO pourra donc se trouver dans l'un de ces 4 états :

- complet et cohérent
- complet et incohérent
- incomplet et cohérent
- incomplet et incohérent

On considérera qu'un objet incomplet ne sera pas nécessairement incohérent car si une de ses propriétés n'est pas instanciée celà n'impliquera pas la violation de la contrainte associée. Parmi des différents états d'un objet, l'état complet-incohérent devra être évité, alors que l'état auquel on veut arriver en fin de conception est complet-cohérent.

I.2) Le Modèle de Données :

Rappels et présentation des composantes du modèle de données :

Dans la première partie, nous avons vu que les qualités essentielles d'un modèle de données pour la CAO doivent être :

* la dynamicité (aussi bien de la structure d'un objet que des liens)

* la possibilité de modéliser des structures volumineuses et complexes.

* la richesse sémantique (le modèle de données doit offrir des mécanismes comme l'agrégation ou la spécialisation) et en particulier la possibilité d'introduire des contraintes dans la définition du modèle.

En CAO, pour une classe d'objets il existe en général peu d'instances. La conception de l'objet se fait par affinages successifs.

Par rapport au domaine des bases de données, le mécanisme de conception assistée par ordinateur ressemble à la conception d'un schéma conceptuel de données dynamique qui serait au fur et à mesure instancié pour tendre vers l'objet final.

On peut donc parler de moule de l'objet (que l'on appelle prototype dans notre Modèle), moule qui sera pendant toute la durée de la conception "essayé" (instancié) puis affiné.

Le modèle de données qui a été choisi est un modèle de données issu des frames (en particulier, on ne fera pas de distinction entre les éléments et les liens entre les éléments) avec une représentation non graphique (à cause de la complexité de la structure des objets).

Dans ce modèle de données on distingue en général 2 types d'objets : les prototypes (qui représentent une classe d'objets) et les instances qui sont les occurences d'une classe donnée.

Un prototype est défini par des attributs décomposés en facettes que nous étudions aux paragraphes I.2.1 (prototype) et aux paragraphes I.2.3 (attributs et facettes).

L'ensemble des attributs qui expriment des propriétés non structurelles du prototype sont exprimées par des procédures que nous étudions dans le paragraphe I.2.3.

Les procédures permettent la prise en compte des contraintes d'intégrité sur les objets que nous verrons au paragraphe I.2.4 .

Tout prototype d'objet pourra être "valué" : la valuation d'un prototype sera une instance que nous étudions au paragraphe I.2.5. Afin d'éviter toute confusion nous allons rappeler les définitions suivantes :

* On appellera M le modèle du projet, M sera constitué d'un ensemble d'objets en cours de conception.
* Un objet en cours de conception, appelé encore objet, est caractérisé par un prototype et par une instance de ce prototype.

I.2.1 Les prototypes :

Avant de définir un prototype, on peut rappeler la notion de classe d'objets. Une classe d'objets est un ensemble d'objets possédant une structure identique et satisfaisant au même ensemble de contraintes.

Un prototype sera le moule d'une classe d'objets, ce moule étant évolutif.

Il existe 2 types de prototypes :

- les prototypes d'objets.
- les prototypes de parties fonctionnelles .

a) notion de partie fonctionnelle :

Une partie fonctionnelle permet de regrouper un ensemble de prototypes d'objets réalisant la même fonction (par exemple dans le cas d'un tournevis l'objet poignée appartient à la partie fonctionnelle "tenir"). Par ailleurs, une partie fonctionnelle permet d'exprimer certains éléments du cahier des charges, qui seront des contraintes que devront vérifier tous les objets appartenant à cette partie fonctionnelle.

b) attributs et facettes :

Un prototype est référencé par un nom : ce nom est unique.

Il est défini par un ensemble d'attributs décomposés en facettes.
Certains attributs permettent de décrire la structure du prototype (attribut-structure et attribut-partie fonctionnelle), d'autres des propriétés non-structurelles (attribut-contrainte intra, attribut-lien-binaire).

Tout attribut est formé de une ou plusieurs facettes : les facettes permettent une description de l'attribut auquel elles appartiennent.

Une facette sera une référence à un prototype ou à un type de prototype (type objet, type partie-fonctionnelle) ou une procédure ou encore à un ensemble.

c) ensembles

Un ensemble correspond à la notion de domaine utilisé dans la définition en intention d'une relation (dans le modèle relationnel).
Il permet de regrouper des objets de structures ou de types parfois différents : leur seule propriété commune sera leur appartenance au même ensemble.

On suppose connus du système les ensembles de base suivants : Entier, Réel, Chaine (taillemax).

Ces ensembles de base sont entièrement instanciés.
Par contre les autres ensembles peuvent être spécifiques à un utilisateur, à un groupe d'utilisateurs ou à même un domaine de conception (ensemble des diamètres de vis d'un catalogue, ensemble de matériaux possibles pour une pièce donnée,...).

d) Un exemple de prototype :

Avant de détailler les notions d'attributs, de facettes, de contraintes, de liens et de procédures nous allons donner un exemple de prototype : il s'agit du prototype de l'objet poignée de tournevis.

```
<prototype> poignée
 type : objet
 père : tournevis
 partie-fonctionnelle : tenir
 longueur : infreel (7.1)
 poids : Réel
```

Les informations prises en compte sont les suivantes :

- le prototype de l'objet poignée appartenant à l'objet tournevis.

- sa fonction est tenir

- sa longueur est une valeur réelle inférieure à 7,1.

- son poids est une valeur réelle.

On peut faire tout de suite quelques remarques au vu de cet exemple :

* certains attributs correspondent à des propriétés structurelles ou fonctionnelles (il s'agit des attributs père et partie fonctionnelle), les autres prennent en compte des propriétés non structurelles (longueur, poids).

* les références faites dans les attributs (par le biais de leurs facettes) à d'autres prototypes (tournevis, tenir) ou à des procédures (infréel) supposent que ceux-ci ont déjà été crées.

* l'attribut longueur possède une facette : infréel (7.1).

* l'attribut poids possède une facette : Réel.

* Réel est un ensemble de base : celui des nombres réels.

I.2.2) Les attributs et les facettes :

I.2.2.1) Définitions :

Comme nous l'avons déjà défini, un prototype est caractérisé par un ensemble d'attributs.

Il existera différents types d'attributs. Un attribut appartiendra à l'un des 5 types suivants :

* descriptif
* structurel
* fonctionnel
* contrainte-intra
* lien-binaire

Tout attribut aura un nom: ce nom sera toujours le même quel que soit le prototype si la caractéristique de cet attribut est de désigner l'objet ou une de ses propriétés structurelles ou fonctionnelles (il s'agira des attributs nom, type, père, fils, partie-fonctionnelle).
Par contre le nom de l'attribut aura une signification sémantique (pour l'exemple précédent longueur et poids) dans le cas où il s'agit d'un attribut de type contrainte-intra ou lien-binaire.

type d'attribut	nombre d'attributs appartenant à ce type	noms des attributs	nombre de facettes
descriptif	2	- nom - type	1 1
structurel	0 à 2	- père - fils	1 1 à n
fonctionnel	0 ou 1	partie fonctionnelle	1 à n
contrainte - intra	0 à n	selon le prototype	1 à n
lien-binaire	0 à n	selon le prototype	1 à n

L'attribut descriptif est le seul attribut obligatoire, c'est à dire qu'il faut le prendre en compte dès la création du prototype, les autres attributs seront crées, modifiés pendant le processus de conception du prototype.

Une facette est spécifique à l'attribut et au prototype auquels elle appartient et son existence n'a de sens que par rapport à l'existence du prototype. Un facette permet de désigner aussi bien le domaine des valeurs de l'attribut (utilisation de procédures, d'ensembles, de prototype) que la façon de l'instancier (explicitement ou automatiquement).

Pendant toute la durée de la conception du prototype d'un objet, la définition de ses différents attributs sera affinée par le biais des créations et modifications de leurs facettes respectives.

Nous allons maintenant décrire plus précisemment les différents types d'attributs que peut posséder un prototype.

I.2.2.2) Attributs descriptifs :

Ces attributs permettent de désigner chaque prototype par le biais de son type et de son nom (qui correspondent chacun à un attribut).
Comme nous l'avons déjà vu précédemment, le nom d'un prototype est unique, tandis que son type appartiendra forcément à l'ensemble {prototype, partie-fonctionnelle}.

Exemple :

<prototype>
 nom : tournevis
 type : objet

<prototype>
 nom : tourner
 type : partie-fonctionnelle

qui peut encore se noter

<prototype> tournevis
 type : objet

<prototype> tourner
 type : partie-fonctionnelle

Les attributs descriptifs d'un prototype sont instanciés dès sa création et suffisent pour marquer son existence. (voir l'exemple ci-dessus)

I.2.2.3) Attributs structurels et fonctionnels :

* attributs structurels :

Les attributs structurels permettent de prendre en compte la structure hiérarchique des objets : un prototype peut appartenir à un prototype père et être composé d'un ou plusieurs prototypes fils.

Tout prototype pourra posséder 2 attributs structurels appelés respectivement attribut père et attribut fils dont les facettes sont des prototypes.

Nous ne tiendrons compte dans le Modèle que de la hiérarchie structurelle entre prototypes appartenant au type objet (on ne parlera pas de liens structurels entre prototypes de parties-fonctionnelles).

Le cas particulier des liens entre prototypes d'objets et de parties-fonctionnelles sera étudié dans le prochain paragraphe concernant les attributs fonctionnels.

Exemple :

<prototype> pr1
type : objet
père : pr5
fils : pr2, pr3, pr4

Cet exemple exprime les liens structurels suivants :

- le prototype pr1 appartient à pr5
- le prototype pr1 est lui-même composé des prototypes pr2, pr3 et pr4.

En particulier l'attribut fils permet de créer un prototype à partir d'autres prototypes existants (elle correspond au concept d'agrégation défini dans les modèles sémantiques).

Le nombre de niveaux de la hiérarchie structurelle de tout modèle à concevoir dépendra de sa complexité : par exemple un circuit intégré est un objet dont la structure hiérarchique sera constituée de nombreux niveaux.

Notes :

- La définition pour un objet a, d'un lien structurel avec un objet b impliquera automatiquement la définition du lien structurel inverse dans le prototype de l'objet b.

- La prise en compte du lien composé-de (c'est à dire de la facette fils) s'apparente au constructeur de structure LIST que l'on peut rencontrer dans de nombreux ouvrages.

- Les liens structurels qui ont été définis dans ce modèle de données favoriseront un certain nombre de fonctionnalités pour le SGBD-CAO associé :
conception ascendante, descendante, imbriquée, héritage.
(voir paragraphe I.3 sur les fonctionnalités).

* attribut fonctionnel :

L'attribut fonctionnel permet d'intégrer le prototype d'un objet dans une ou plusieurs parties fonctionnelles.

Cet attribut ne sera défini dans notre modèle que pour les prototypes d'objets : celà implique que nous n'utiliserons pas la notion d'hiérarchie fonctionnelle qui était rendue possible par la définition du modèle de données.

En effet, contrairement aux objets CAO dont la structure hiérarchique est évidente, il est plus rare de rencontrer la notion d'hiérarchie fonctionnelle dans les exemples de conception.

Comme nous l'avons signalé précédemment, les parties fonctionnelles permettent en général de collecter les informations du cahier des charges et seront donc pour la plupart définies en début de conception.

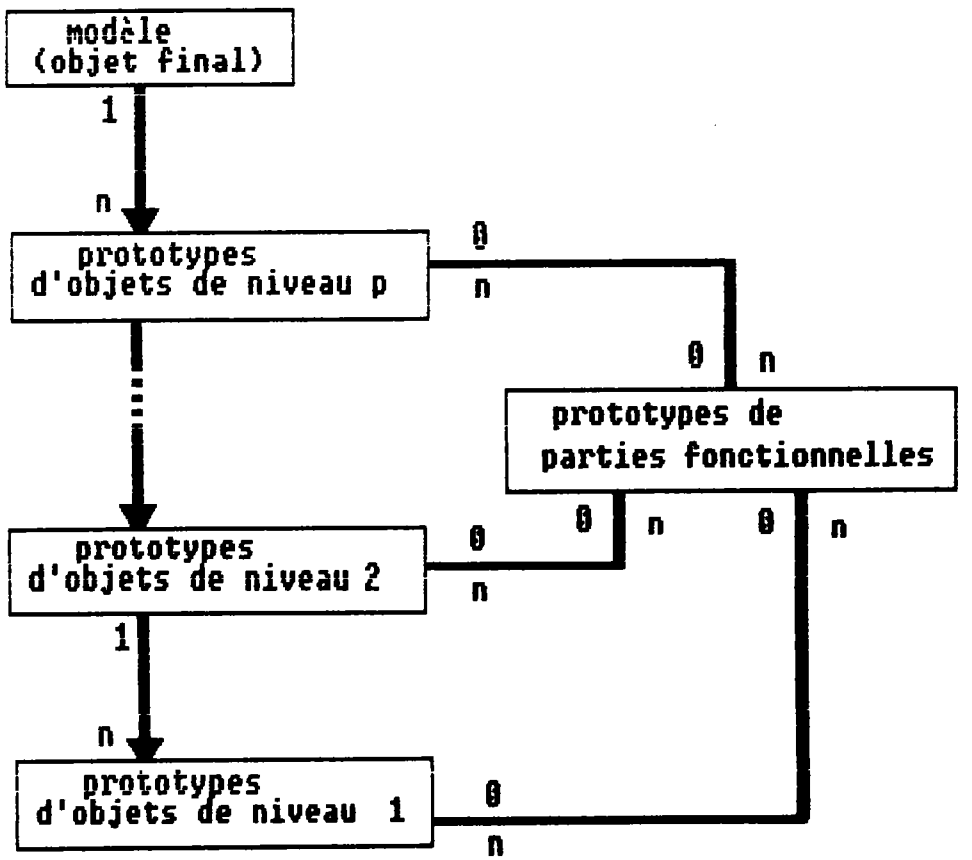
L'attribut fonctionnel d'un prototype est défini par son nom qui est "partie-fonctionnelle" et ses facettes qui seront les différentes parties-fonctionnelles auquelles appartient le prototype de l'objet. Inversement l'attribut fils d'un prototype de partie-fonctionnelle permet de connaitre la liste des prototypes d'objets appartenant à cette partie-fonctionnelle (voir schéma et tableau qui suivent).

Nous avons distingué 2 découpages possibles des prototypes d'objets :

- une hiérarchie structurelle décrite par les liens structurels (figure ci-dessous)

- un découpage fonctionnel décrit par les liens fonctionnels et structurels.

Hiérarchie structurelle



Les apports du découpage fonctionnel par rapport à la hiérarchie structurelle sont les suivants :

- la mise en évidence de nouveaux liens entre objets : ces liens sont en particulier proches de la démarche de conception et permettent une prise en compte des contraintes du cahier des charges.

- la possibilité d'intégrer la notion de vue : ces liens permettent en particulier le découpage du projet de conception en sous projets.

- l'héritage de propriétés devient possible sur un ensemble d'objets qui ne sont pas liés hiérarchiquement.

Nous avons décrit ces différents éléments lors de la présentation des fonctionnalités du système BD-CAO au paragraphe I.1.

I.2.2.4) Autres attributs :

Les autres attributs permettent de définir les propriétés non structurelles d'un prototype : ces propriétés porteront sur toutes ses occurences et seront donc un moyen d'accepter ou non un objet comme occurence de ce prototype.

Ces attributs intègrent simultanément la notion de propriété et de contrainte sur cette propriété. Leurs facettes peuvent faire référence aux éléments suivants :

- d'autres facettes
- d'autres prototypes
- des ensembles
- des procédures

Parmi ces attributs nous distinguons deux catégories :

- les attributs contraintes-intra : permettent de prendre en compte une propriété spécifique à un prototype donné (par exemple son poids) même si cette propriété dépend de propriétés d'autres prototypes (le poids d'un prototype est le double de celui d'un second prototype et doit être inférieur à celui d'un troisième prototype).

- les attributs liens-binaires : représentent une propriété commune à deux prototypes, cette propriété n'aura de prototypes que par rapport à des attributs de ces deux sens (par exemple le poids total de deux objets est inférieur à 10kgs, ce poids sera donc déterminé par rapport aux poids de ces deux objets).

Nous allons maintenant décrire plus en détails ces deux types d'attributs.

a) Attribut contraintes-intra :

Un attribut contraintes-intra n'aura de signification que par rapport aux propriétés concernant le prototype dans lequel il a été défini.

Comme tous les autres attributs, il est composé de facettes qui pourront faire référence à une procédure, celle-ci pouvant avoir comme paramètres d'autres attributs du prototype ou d'un autre prototype.

* Exemple 1

```
<prototype> lame
 type : objet
 père : tournevis
 longueur : supréel (4.0)
          : infréel (7.1)
 densité : Réel
 volume  : Réel
 poids   : = mult ( volume, densité)
```

----> attributs contraintes-intra

Dans cet exemple, le prototype de lame possède 4 attributs contraintes-intra objet : longueur, densité, volume et poids. Ces attributs se réfèrent à 3 procédures supréel() (réel supérieur strictement à), infréel() (réel inférieur strictement à) et mult() (multiplication).

Pour toutes les instances du prototype lame, les valeurs de leurs attributs longueur devront être des réels supérieurs à 4,0 et inférieurs à 7,1 tandis que les valeurs de leurs attributs densité et volume doivent être des réels.

L'attribut poids est un attribut calculé : les occurences de cet attribut sont directement liées à celle des attributs volume et densité et appartiennent implicitement à l'ensemble des réels positifs (ensemble des résultats possibles défini lors de la définition de la procédure mult).

On pourra remarquer sur cet exemple qu'on peut distinguer 2 catégories d'attributs contraintes-intra par rapport à la façon de les instancier : les attributs instanciés explicitement et ceux instanciés implicitement appelés attributs calculés ou encore dérivés . Tout attribut calculé possédera obligatoirement une facette qui définira la fonction de calcul : on appelle cette facette la facette calculée (appelée encore lien dans de nombreuses études).

* Exemple 2

Dans l'exemple ci-dessus nous n'avons pas considéré le cas particulier de contraintes-intra pour lesquelles certaines facettes (calculées ou non) font référence à des attributs contraintes-intra d'autres prototypes. Nous allons décrire maintenant un exemple d'attribut contrainte-intra pour lequel ses deux facettes font référence à d'autres prototypes :

```
              <prototype> tournevis
               type  : objet
facette --->   poids : add((lame, poids), (poignée, poids))
calculée
facette --->           infreel(1.3)
non calculée
```

```
              <prototype> poignée
               type  : objet
facette --->   poids : supréel((lame, poids))
non calculée
```

Les attributs contraintes-intra poids des objets tournevis et poignée expriment par le biais de leurs facettes les règles suivantes :

- le poids du tournevis est égal à la somme des poids de la poignée et de la lame (utilisation de la fonction de calcul add() correspondant à l'opérateur addition) : ce poids doit être inférieur à 1,3.
- le poids de la poignée est supérieur à celui de la lame.

Dans cet exemple la facette "supreel((lame, poids))" fait référence à deux procédures : supréel() et égal(). Cette dernière procédure qui correspond à l'opérateur égalité est implicite et donc n'est pas présente dans la facette.
Si on voulait fixer la règle suivante "le poids de la poignée est supérieur à deux fois celui de la lame", la contrainte-intra du prototype de poignée correspondant à cette règle serait définie par une facette et aurait la forme suivante :

```
poids : supréel(mult(2,(lame,poids))
```

L'attribut poids dans ce cas est composé d'une seule facette qui est une facette non calculée bien qu'elle prenne en compte la fonction de calcul mult().

* Définition des facettes :
Suite à ces exemples on peut donner la définition suivante permettant de distinguer les facettes calculées des facettes non calculées : une facette calculée est une facette qui instancie automatiquement l'attribut (calculé) auquel elle appartient, les autres facettes ont pour rôle de déterminer des ensembles de valeurs admissibles.

type de prototype / attributs	OBJET		PARTIE-FONCTIONNELLE	
	nombre de facettes possibles	type des facettes	nombre de facettes possibles	type des facettes
* nom	1	pr. ob	1	pr. pf
* type	1	type	1	type
père	1	pr. ob	0	-
fils	1 à n	pr. ob	1 à n	pr. ob.
partie-fonctionnelle	1 à n	pr. pf	0	-
contrainte-intra 1	1 à n	proc1, ens.	1 à n	proc1, ens.
. . . .				
contrainte-intra p	1 à n	proc1, ens.	1 à n	proc1, ens.
lien-binaire 1	1 à n	proc2, ens.	1 à n	proc2, ens.
. . . .				
lien-binaire q	1 à n	proc2, ens.	1 à n	proc2, ens.

* : attribut obligatoire

proc1 = procédures dont les paramètres peuvent être des facettes du même prototype.
proc2 = procédures dont les paramètres peuvent être des facettes du prototype ou d'un autre prototype.
pr.ob. = prototype d'objet
pr.pf. = prototype de partie fonctionnelle
ens. = ensemble

Attributs et Facettes

* Consistance

L'ensemble des contraintes-intra défini sur un prototype doit toujours vérifier la propriété de consistance : celà veut dire que l'ensemble des objets solutions réalisables de tout prototype soit non vide. Par exemple si les facettes de l'attribut longueur du prototype lame vu précédemment, avaient été :

```
longueur : supreel (4.0)
           infreel (3.0)
```

La propriété de consistance n'aurait pas été vérifiée.

b) Attribut lien-binaire :

Un attribut lien-binaire sera défini par rapport à 2 attributs contraintes-intra appartenant à 2 prototypes différents.

Reprenons dans l'exemple du tournevis les prototypes des objets lame et poignée.

```
<prototype> lame
 type : objet
        ‖

 longueur :supreel(4.0)
           infreel(7.1)
        ‖

longueur-totale := somme (longueur,(poignée,longueur))
                          infreel(10.5)
<prototype> poignée
  type : objet
        ‖

longueur : Reel
```

L'attribut lien-binaire longueur-totale ci-dessus permet d'exprimer la contrainte suivante :"la somme des longueurs de la lame et de la poignée est un réel strictement inférieur à 10,5."

De même que dans le cas des contraintes-intra, tout attribut lien-binaire devra être consistant.

L'instanciation des attributs liens-binaires se fera toujours automatiquement, c'est-à-dire, qu'on ne prendra en compte que des attributs liens-binaires calculés.

I.2.3 Procédures :

Nous avons vu que dans la définition de certains attributs d'un prototype (contrainte-intra et lien-binaire), on faisait référence à des procédures : celles-ci permettent d'exprimer des contraintes portant sur un ou deux attributs, ou encore d'instancier automatiquement un attribut.

Ainsi toute facette d'un attribut contrainte-intra ou lien-binaire utilisera une procédure, même si celle-ci n'est pas exprimée explicitemenet dans le prototype : par exemple, la facette "volume : Reel" du prototype de l'objet lame implique l'utilisation d'une fonction booléenne testant l'appartenance à l'ensemble des réels.

Nous distinguerons 2 types de procédures : (1)

- les procédures générales.
- les procédures spécifiques.

En voici quelques exemples :

Procédures générales :
---

* intervalle : permettent de vérifier si une occurrence d'attribut appartient à un intervalle donné. Cet intervalle sera soit entier ou réel.

* supérieur-à (respectivement inférieur-à) : cas particulier de la procédure intervalle.

* appartient : permettent de vérifier si une occurrence d'attribut appartient à un ensemble donné.

* valeur : procédure constante dont le résultat sera une ou plusieurs valeurs numériques réelles.

* opérations usuelles : ces procédures correspondent aux opérateurs arithmétiques comme l'addition, la soustraction, la multiplication sur les ensembles de base tels que les entiers ou les réels.

(1) On peut définir une autre classification des procédures : les procédures contrôlant l'appartenance à un ensemble donné et les procédures de calcul.

Procédures spécifiques :

Ce sont des procédures qui sont spécifiques au domaine de conception (mécanique, électronique, architecture, ...)
De plus parmi ces procédures, on pourra distinguer 2 catégories:

- procédures du domaine

- procédures du concepteur (procédures privées)

Par exemple en mécanique, il existe des programmes de calculs d'engrenages qui dépendront de paramètres comme la durée de vie prévue, la puissance du moteur, les matériaux utilisés ou les conditions de fonctionnement.

Ces procédures spécifiques doivent être intégrées facilement par l'utilisateur dans la base de données : le logiciel BD-CAO devra donc posséder un outil de haut niveau permettant l'attachement procédural.

D'autre-part toute procédure, qu'elle intervienne dans une contrainte-intra ou un lien-binaire doit pouvoir être activée automatiquement : nous reverrons ces problèmes dans le paragraphe III dans le cadre des notions d'attachement et de déclenchement procéduraux.

I.2.4. Contraintes d'intégrité :

L'objectif d'un modèle de données est de fournir une image aussi fidèle que possible de la réalité, tout en fournissant un maximum de sémantique : cette sémantique est exprimée à l'aide de la structure de données et des contraintes d'intégrité.

En Base de Données on considère que les contraintes d'intégrité représentent tout ce que l'on veut intégrer dans le modèle et qui ne peut pas être pris en compte par la structure de données: donc elles constituent un ajout par rapport à la structure de données. Il s'en suit que plus une structure de données sera riche, plus le nombre de contraintes d'intégrité sera faible.

Les contraintes d'intégrité au sens Bases de Données peuvent être classées selon différents critères :

- contraintes statiques et contraintes dynamiques
- contraintes simples, ensemblistes et référentielles.

Nous distinguerons 2 types de contraintes dans notre modèle:

- celles qui permettent de représenter l'ensemble des valeurs admissibles pour un attribut donné (qu'il soit instancié implicitement ou explicitement), par exemple supreel(4.) ou Entier.

- celles qui permettent l'instanciation automatique d'un attribut, par exemple mult(volume,densité): ces contraintes correspondent aux attributs calculés. Ce type de contraintes permet de prendre en compte la façon d'instancier alors que les autres contraintes contrôleront l'instance.

Tout attribut contrainte-intra ou lien-binaire pourra posséder simultanément des facettes correspondant au premier type de contrainte et une facette calculée. Par exemple l'attribut lien-binaire "longueur-totale" du prototype de l'objet vu précédemment possède une facette calculée (somme (longueur, (poignée, longueur)) et une facette permettant le contrôle du résultat de la facette calculée (infreel(10.5)). Ainsi à chaque contrainte correspondra une procédure.

Par rapport à la classification des contraintes en Bases de Données, on pourra dire que :

- les contraintes de notre modèle de données sont toutes statiques.

- Ces contraintes seront soit simples ou référentielles (dans ce cas elles feront référence à 2 facettes appartenant soit au même prototype ou à 2 prototypes différents). Par contre étant donné que chaque occurrence d'un prototype correspond à autant d'essais de conception, la notion de contrainte ensembliste au sens Bases de Données n'a pas d'intérêt (nous reverrons cette notion dans le paragraphe sur l'instanciation). On pourra toutefois noter qu'un héritage de contraintes par le biais des liens structurels ou fonctionnels est rendu possible par le modèle de données.

I.2.5. Instances

I.2.5.1) Définitions :

Une instance d'un prototype P est l'association d'un nom d'instance (c'est le nom du prototype et le numéro d'instance) et d'une suite de valeurs qui appartiennent aux ensembles de valeurs acceptables définis dans le prototype P.

A un prototype on pourra associer à tout moment un certain nombre d'instances qui seront différentes "solutions" correspondant à ce prototype. La création d'une instance d'un prototype reviendra à la valuation de ses attributs liens-binaires ou contraintes-intra. On pourra rappeler qu'un objet sera constitué d'un prototype et d'une instance de ce prototype.

Afin d'illustrer ces concepts, reprenons l'exemple du prototype de lame vu &I.2.2.d

```
<prototype>lame
type : objet
père : tournevis
longueur : supréel(4.0)
         : infreel(7.1)
densité : Reel
volume : Reel
poids : =mult(volume, densité)

<instance 1>lame
longueur : 4.5
densité : 3
volume : 0.1
poids : 0.3

<instance 2>lame
longueur : 5
densité : 4
volume : 0.2
poids : 0.8
```

Sur cet exemple on peut remarquer les 2 types d'attributs correspondant à la façon dont on les instanciera :

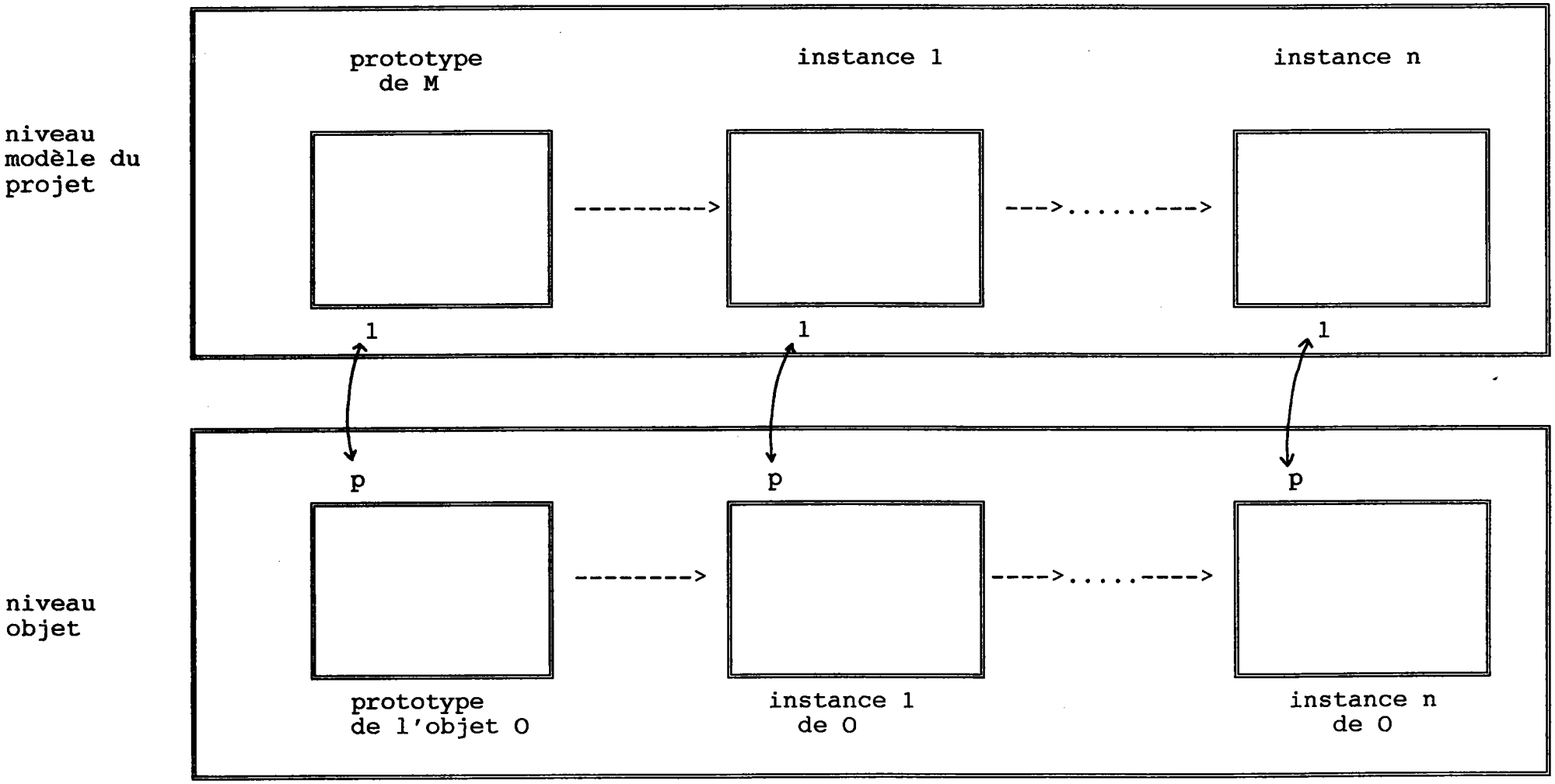
- les attributs calculés dont l'intanciation se fera automatiquement : chaque attribut de ce type devra donc posséder une facette calculée (ainsi l'attribut "poids" possède une facette calculée "mult(volume, densité)".

- les attributs explicites dont l'instanciation se fera directement par l'utilisateur: sur l'exemple c'est le cas des attributs "longueur", "densité" et "volume".

On peut signaler le problème posé par une instanciation incomplète d'un prototype : nous l'étudierons dans le paragraphe I.3.3.

I.2.5.2) Notion de version d'un projet :

Un processus de conception possède un caractère essai-erreur ce qui implique qu'un système de CAO doit offrir la possibilité de revenir en arrière et d'autre part de mener plusieurs conceptions en parallèle.
Si M désigne le modèle du projet, alors M sera constitué d'un ensemble d'objets qui seront une image du projet à un instant t. Tout au long de l'exsitence d'un projet de CAO, cette image va évoluer et aussi à tout moment il pourra exister simultanément plusieurs images c'est à dire plusieurs modèle du projet.



Lien entre modèle d'un projet et objet .

On appelera version l'ensemble des modèles du projet existant à un moment donné : ces modèles du projet seront tous constitués des mêmes prototypes d'objets mais auront pour ces prototypes des instances différentes. Dans une même version, le nombre d'instances pour tout prototype sera égal au nombre de modèles du projet appartenant à cette version (voir schéma qui suit).

Ce nombre est variable d'une version à l'autre.

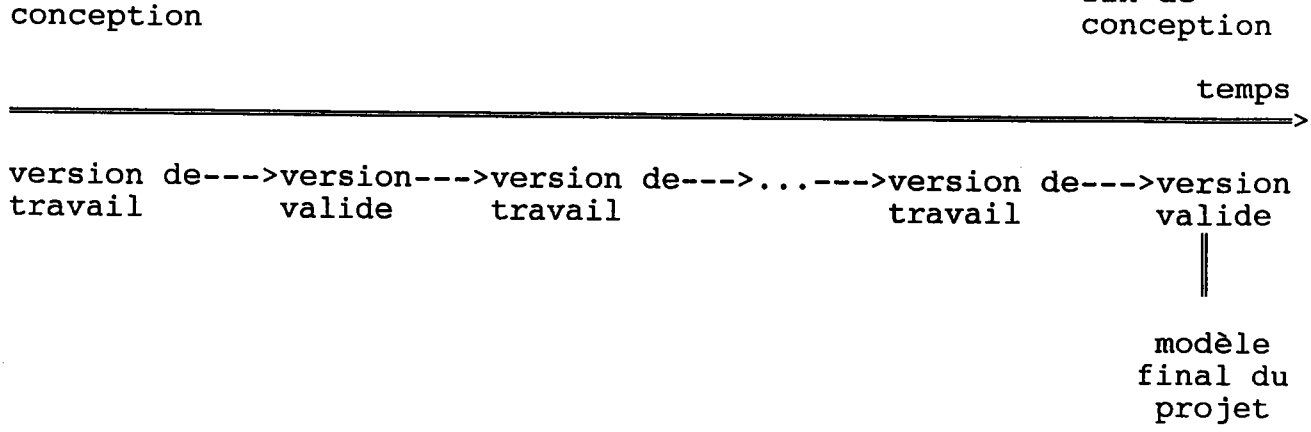
Remarques concernant le schéma :
Le schéma représente une version composée de n modèles du projet : tous ces modèles du projet auront en commun le même prototype.
Tout modèle du projet contient p objets, et pour tout prototype il existera n instances.

I.2.5.3) Version de travail et version valide :

Une version de M correspondra donc à l'ensemble de solutions d'un projet ou encore à un état de ce projet à un moment donné. On pourra d'ailleurs parler de version de travail et de version valide. La différence entre les deux est qu'une version de travail peut contenir un certain nombre d'incohérences (des instances d'objets vis-à-vis des prototypes associés ou encore des objets incomplets) alors qu'une version valide ne comportera plus d'incohérences au niveau de l'ensemble des objets qui la composent.

Au cours d'une phase conception, toute version de travail évoluera afin de tendre vers une version valide. Une version valide tant qu'elle ne correspondra pas exactement au cahier des charges sera à nouveau affinée et transformée en version de travail.

L'ensemble des versions valides pour un projet de conception donné permet d'en représenter les différentes étapes (ce qui donne la possibilité de revenir en arrière lorsque les choix ont été mauvais à un moment donné).
Donc un processus de conception en le regardant par rapport aux versions pourra se schématiser de la façon suivante.



Pour un projet donné, l'ensemble de ses versions valides et certaines versions de travail vont être stockés dans Base de Données Projet (BDP).

I.3) Les traitements

Le modèle que nous utilisons doit pouvoir gérer différents types d'informations : des prototypes et des instances mais aussi des ensembles et des procédures (c'est-à-dire une certaine forme de connaissance);

On peut considérer que le Modèle doit assurer deux fonctions :

- permettre à un utilisateur de définir et manipuler ses données de conception conformément au modèle de données que nous avons défini dans le paragraphe I.2.
- préserver les données de dégradations qui seront dues à l'utilisateur lui-même (nous ne prenons pas en compte ici les problèmes d'accès multi-utilisateurs ou de pannes).

I.3.1) Définition et manipulation des données de conception

Dans une même session de travail la Modèle permet de manipuler aussi bien les structures que les données associées.Pour un utilisateur ces manipulations reviennent à faire des opérations (création, modification, suppression et d'affichage) sur les constituants suivants du modèle de Données :

- **les prototypes (étudiés au paragraphe I.3.1.1).**
- **les ensembles (étudiés au paragraphe I.3.1.2).**
- **les procédures (étudiées au paragraphe I.3.1.3).**
- **les instances (étudiées au paragraphe I.3.1.4).**

Enfin dans le paragraphe I.3.1.5 nous verrons comment le modèle prend en compte les fonctionnalités définies au paragraphe I.1.

I.3.1.1) Opérations sur les prototypes :

Dans ce paragraphe nous examinons les différentes actions applicables sur les prototypes dans le cadre d'un processus de conception. On peut rappeler qu'il existe deux types de prototypes: les prototypes d'objets et les prototypes de parties -fonctionnelles.

Les opérations qu'on appliquera sur un prototype sont classiques. Elles sont les suivantes :

- création
- modification
- suppression
- affichage

Nous allons séparer ces opérations en deux catégories: les mises à jour (création, modification et suppression) et les affichages.

a) Mise à jour d'un prototype :

* création et modification :

Nous avons vu que la création d'un prototype revient à l'instanciation de son attribut descriptif.

Un prototype comprend 2 types d'attributs correspondant à ses propriétés structurelles et ses propriétés non structurelles. Il s'en suit que l'évolution d'un prototype est issue de la modification d'un de ces types d'attributs.

La modification d'un attribut correspond à la création, la modification ou la suppression d'une de ses facettes.

* suppression :

La suppression d'un prototype devra impliquer la suppression de toutes les références à ce prototype et à ses attributs dans les autres objets.

Il existe un deuxième niveau de suppression, c'est celui d'un attribut ou encore d'une facette. Un attribut ou une facette sont spécifiques au prototypes auquels ils appartiennent. Dans le cas où dans un attribut il est fait référence à un autre prototype ou à un attribut d'un autre prototype (cas des facettes calculées), la suppression de cet attribut impliquera une mise à jour des prototypes associés.

Le problème des mises à jour en cascade est illustré dans l'exemple qui suit :

exemple :

On suppose que dans le cadre d'une conception on ait crée à un moment donné 3 prototypes : un prototype d'objet désigné par p01 et deux prototypes de parties fonctionnelles appelés pf1 et pf2. On supposera que ces 3 prototypes ont comme seule propriété leur attribut descriptif et s'exprimeront dans le modèle de données de la façon suivante :

<prototype> p01
type : objet

<prototype> pf1
type : partie-fonctionnelle

<prototype> pf2
type : partie-fonctionnelle

Les 2 opérations suivantes auront le même effet sur la constitution de ces prototypes :

- dans le cadre de l'affinage du prototype de p01, définition de pf1 et pf2 comme occurences de son attribut fonctionnel.

- lors de la création des attributs structurels des prototypes pf1 et pf2, ajout du prototype de l'objet p01 en tant qu'occurence de leurs facettes Fils.

La définition de ces 3 prototypes après l'une de ces 2 opérations sera la suivante :

<prototype> p01
type : objet
partie-fonctionnelle : pf1, pf2

<prototype> pf1
type : partie-fonctionnelle
fils : p01

<prototype> pf2
type : partie-fonctionnelle
fils : p01

remarque :

On peut remarquer suite à cet exemple, que si dans la conception d'un prototype, on définit un lien avec un autre prototype, le lien inverse devra être généré automatiquement dans le deuxième prototype (il en sera de même dans le cas d'une modification ou une suppression de liens: le lien inverse sera automatiquement modifié ou suppprimé).

Ces liens apppartiendront à l'un des types suivants :

- structurel
- fonctionnel
- contrainte-intra (dans certains cas)
- binaire

Cette génération automatique du lien "inverse" provient de la définition du modèle de données. En effet dans ce modèle on ne fait pas de distinction entre objets et relations entre objets : une relation (ou un lien) entre objets ne sera pas comme entité à part, mais sera simplement considérée comme une propriété de ces objets.

b) <u>affichages</u> :

Il doit être possible à tout moment d'afficher un prototype ou un groupe de prototypes (par exemple les objets appartenant à une partie fonctionnelle donnée). L'affichage d'un prototype impliquera celui de ses propriétés structurelles et non structurelles: bien-sûr on pourra n'afficher qu'une partie de ses attributs (par exemple les attributs liens-binaires).

I.3.1.2) <u>Opérations sur les ensembles</u> :

L'utilisation d'ensembles lors de la définition d'un prototype a été introduite dans le paragraphe I.2.1.

On peut considérer qu'il existe 2 types d'ensembles:

- les ensembles de base.
- les ensembles spécifiques.

Les ensembles de base tel que Réel ou Chaine (taillemax) existent implicitement dans le Modèle.
Par contre les ensembles explicites (comme par exemple l'ensemble composé des différents diamètres de vis d'un catalogue) devront être crées par l'utilisateur du système.
On pourra créer, modifier ou supprimer les ensembles spécifiques et ceci à n'importe quel moment de la conception: par exemple lors de la session de création du prototype d'une vis, on peut définir l'ensemble des diamètres possibles (il s'agit ici d'un cas de conception imbriquée).

I.3.1.3) <u>Opérations sur les procédures</u> :

Les procédures sont utilisées dans la définition des contraintes-intra et des liens-binaires (voir paragraphe I.2.3). Les <u>procédures générales</u> seront connues du MODELE, donc non accessibles en modification ou en suppression par un utilisateur.

Les procédures spécifiques (à l'utilisateur ou au domaine) sont accessibles par le biais de leurs algorithmes.
Les opérations possibles sur les procédures spécifiques sont:

- création.
- modification.
- suppression.

* création :

Un utilisateur doit pouvoir créer un programme de calcul par l'intermédiaire d'un système de développement classique, et ensuite de l'intégrer dans son système CAO sous la forme d'une procédure (voir attachement procédural au paragraphe III).

* modification :

La seule modification possible pour une procédure spécifique concerne son algorithme. Suite à la modificaiton d'une procédure, toutes les facettes calculées utilisant cette procédure seront aussitôt reexécutées.

* suppression :

La suppression d'une procédure provoque la suppression de toutes les facettes calculées définies à partir de cette procédure. Pour supprimer une procédure il suffira de donner son nom.

I.3.1.4) Les opérations sur les instances :

Rappelons qu'un objet CAO est défini par son prototype et une instance de ce prototype. On accèdera à une instance par le nom du prototype et son numéro d'instance (voir schéma dans le paragraphe I.2.5).

Une instance correspondra à un ensemble d'occurences des attributs contraintes-intra et liens-binaires d'un objet.

Nous avons vu qu'il existe 2 manières d'instancier un attri
-but :

- explicitement (cas d'un attribut non calculé)

- implicitement (cas d'un attribut calculé).

Cela voudra dire que l'utilisateur ne pourra modifier directement que les attributs non calculés c'est-à-dire uniquement les attributs contraintes-intra non calculés(étant donné que les attributs liens-binaires sont tous calculés).

Nous allons étudier les différentes opérations possibles sur une instance, puis mettre en évidence les problèmes de complétude et de cohérence que celà pose.

* Opérations sur les instances :

Les opérations possibles sur les instances sont classiques. Elles sont les suivantes :

- création
- modification
- suppression

Pour créer une instance d'un prototype il suffit de donner son numéro d'instance.

Ensuite le concepteur pourra affecter des valeurs aux différents attributs non calculés.

La modification d'une instance d'un objet est effectuée dans l'un des cas suivants :

- instanciation de l'un de ses attributs explicites (que cet attribut soit déjà instancié ou non)
- instanciation implicite de l'un de ses attributs calculés (par instanciation d'un des objets paramètres de la facette calculée correspondant à l'attribut calculé).

La suppression pour l'instance i d'un objet d'une occurence x de l'un de ses attributs (soit a cet attribut) peut avoir les conséquences suivantes :

- suppression logique de l'instance i dans le cas où a était le seul attribut instancié.
- dans le cas où a est paramètre d'une fonction de calcul facette d'un autre attribut a' (a' appartient ou non au même objet que a)
  . si cette fonction de calcul permet d'instancier automatiquement a', la suppression de x impliquera celle de l'instance de a' (si celle-ci existe).
  . si cette fonction de calcul sert à déterminer un ensemble de valeurs admissibles, la suppression de x entraînera celle de cet ensemble (si celui-ci existe).

Tout objet, suite à une opération qui sera faite sur l'instance de son prototype (création, modification ou suppression) pourra alors se trouver dans un état incomplet ou incohérent: ce problème est étudié au paragraphe I.3.2.

Enfin il doit être possible d'afficher l'instance (au partie de l'instance) d'un prototype ou encore un ensemble d'instances (par exemple tous les fils d'un objet donné).

I.3.1.5) Quelques fonctionnalités spécifiques au Modèle :

* Conception ascendante et descendante :

Le Modèle permet de créer un prototype d'objet (père) ou d'une partie fonctionnelle à partir de prototype d'objets déjà existants: c'est le cas de la conception ascendante. De même pour un prototype existant d'objet (père) ou de partie fonctionnelle pouvoir rajouter un prototype d'objet (fils ou appartenant à la partie fonctionnelle) : c'est le cas de la conception descendante.

* Création imbriquée :

Au cours de la phase de la création d'un prototype A, on doit pouvoir faire référence à un prototype ou à un ensemble jusque là inconnus: le Modèle doit permettre de créer alors ce prototype ou cet ensemble et de poursuivre ensuite la phase de création du prototype A. Le cas particulier de la référence à une procédure inexistante sera étudié au paragraphe III.

* héritage et déduction :

L'héritage se fera par le biais des liens fonctionnels : ainsi un prototype d'objet appartenant à une partie-fonctionnelle possédera en plus de ses caractéristiques propres celles issues de la partie-fonctionnelle. Donc toutes les containtes-intra d'une partie-fonctionnelle seront des propriétés supplémentaires pour tous les prototypes appartenant à cette partie fonctionnelle. Un deuxième niveau d'héritage peut-être obtenu par le biais des liens structurels : par exemple, si un objet est en fonte, il en sera de même des objets fils.

Une autre forme de déduction sera obtenue dans le cadre des attributs calculés (il s'agit ici d'une dérivation). Le choix de la dérivation comme forme de déduction a été fait car étant donné les temps de réponses relativement longs en CAO, une déduction par génération n'aurait fait que diminuer les performances (par exemple on peut imaginer la facette calculée d'un lien-binaire qui serait une référence à des attributs calculés).

* Objets incomplets et incohérents :

Pour le prototype d'un objet (donc pour l'objet) on peut désigner deux niveaux d'incomplétude:

- niveau attribut: étant donné les affinages successifs on ne peut connaître à l'avance tous les attributs d'un prototype. Dans certains cas, au moment de la création d'un prototype celui-ci n'aura comme seule propriété que son attribut descriptif.

- niveau facette: pour un attribut d'un prototype l'ensemble des facettes le composant peut évoluer au cours de la conception (ajout d'une nouvelle facette, modification de la facette calculée,...)
La gestion par le Modèle des objets incohérents est étudiée au paragraphe I.3.2.

I.3.2) Contrôles d'intégrité :

Le Modèle offre la possibilité de définir pour chaque projet et même pour chaque objet un ensemble de règles qui devront être respectées par toutes donnée de conception. Le respect de ces règles se fera par le biais de controles d'intégrité.

On pourra distinguer deux types de règles:

- celles qui sont spécifiques au modèle de données (règles implicites quelque soit le projet).

- celles qui sont spécifiques à un projet: elles devront donc être exprimées par le concepteur.

I.3.2.1) Règles spécifiques au modèle de données :

On donnera ici un certain nombre de contrôles faits par le Modèle pour la vérification de ces règles:

- contrôle d'unicité et d'existence: lors de la création d'un prototype, d'une instance, d'une procédure ou d'un ensemble, le Modèle devra vérifier l'unicité de leur nom. De même si pendant la conception il est fait référence à un prototype un instance, une procédure ou un ensemble le Modèle devra vérifier leur existence (dans le cas de la non existence, un message devra être envoyé au concepteur avec la possibilité de création de l'entité manquante).

- contrôle d'hiérarchies structurelles.

- autres contrôles: un objet ne pourra avoir au plus qu'un attribut fonctionnel, qu'une seule facette dans son attribut père, qu'une seule facette calculée dans un attribut lien-binaire, etc...

I.3.2.2) Règles spécifiques à un projet :

Ces règles correspondent dans le vocabulaire Base de Données aux notions de contraintes d'intégrité c'est-à-dire qu'elles n'ont pas pu être exprimées par le modèle de données.

Nous allons donner ici deux exemples, le premier concerne la phase de conception d'un prototype, le deuxième celle de son instanciation. Pour illustrer les contrôles effectués par le Modèle, nous allons reprendre l'exemple de la lame de tournevis.

```
<prototype> lame
type: objet
père: tournevis
longueur: supreel (4.0)
        : infreel (7.1)
densité : Reel
volume  : Reel
poids   : = mult (volume, densité)
          infreel (4.5)
```

* L'ensemble des contraintes-intra défini sur un prototype doit toujours vérifier la propriété de consistance: celà veut dire que l'ensemble des objets solutions réalisables de tout prototype soit non vide. Par exemple si les facettes de l'attribut longueur du prototype lame vu précédemment, avaient été :

```
longueur: supreel (4.0)
          infreel (3.0)
```

La propriété de consistance n'aurait pas été vérifiée.
Donc tout ajout d'une nouvelle facette non calculée devra être automatiquement suivi de la vérification de sa consistance.

* D'autre part supposons que suite à la création d'instances de ce prototype, on obtienne les instances suivantes:

```
<instance 1> lame
longueur: 2.0
densite : 1.0
volume  : 3.0
poids   : 3.0

<instance 2> lame
longueur: 5.0
densité : 1.0
volume  : 6.0
poids   : 6.0
```

Pour la première instance il s'agira d'une violation d'une règle de cohérence forte c'est-à-dire que la valeur "2.0" pour l'attribut "longueur" sera rejetée au moment de l'instanciation de cet attribut.

La deuxième instance correspond à la violation d'une règle de cohérence faible attachée à la valeur de l'attribut "poids". Dans ce cas le Modèle le signalera à l'utilisateur par le biais d'un message: le contrôle des règles de cohérence faibles comme nous l'avons déjà vu dans le paragraphe I.1 peut se faire suite à chaque instanciation, à la demande explicite du concepteur ou encore en phase de validation d'un version.

I.4) Synthèse du Modèle :

Dans ce Modèle tout objet CAO est défini par le biais d'un prototype et d'une instance de ce prototype (même notion que les frames). Un prototype est constitué d'un ensemble d'attributs qui correspondront à ses propriétés structurelles, fonctionnelles, à des contraintes sur l'objet et entre objets. Tout attribut est exprimé par l'intermédiaire de références à des prototypes, des ensembles et par des procédures.

Les fonctionnalités de ce Modèle sont les suivantes :

- définition et manipulation simultanées des données.

- conception ascendante et descendante grâce aux attributs structurels et fonctionnels.

- création imbriquée : dans une phase de création d'un prototype il est possible de créer un ensemble ou un autre prototype.

- héritage de propriétés grâce aux attributs fonctionnels : cet héritage est limité du fait de la non-existence d'une hiérarchie fonctionnelle.

- déduction d'une propriété d'un objet par l'intermédiaire d'autres propriétés de cet objet ou d'un second objet.

- gestion d'objets incomplets et/ou incohérents.

Le Modèle que nous avons défini doit permettre à un utilisateur de définir et de manipuler ses données de conception. De plus un ensemble de régles gérées par le Modèle permet d'assurer une certaine intégrité de ces données : la gestion des pannes et les accès multi-utilisateurs ne sont pas intégrés dans le Modèle, par contre il doit posséder des outils de gestion des versions (validation, sauvegarde, reprise, etc...).

II) Lien entre Modèle et structure de données :

Le Modèle que nous avons décrit précédemment (paragraphe I) est composé d'un modèle de données et d'un ensemble de traitements applicables à ce modèle de données.

Nous allons maintenant définir une structure de données et montrer son adéquation avec le modèle de données par le biais de quelques exemples. Cette structure de données a été le point de départ d'une maquette de SGBD-CAO réalisée en TURBO-PASCAL: les fonctionnalités de ce système correspondent à celles définies pour le Modèle (paragraphe 1.1). La maquette du SGBD-CAO est décrite en ANNEXE 1.

II.1) La structure de données :

II.1.1) Présentation générale :

La structure de données que nous allons définir est dynamique: elle est construite à partir du concept de liste linéaire. Les objets c'est-à-dire les prototypes aussi bien que les instances sont représentés par des listes linéaires. Il en sera de même des attributs de ces objets. Dans cette structure de donnée, les procédures sont identifiées par un numéro de procédure; certaines procédures sont des fonctions booléennes testant l'appartenance à un ensemble qui peut être spécifique à l'utilisateur.

Tout modèle du projet sera stocké dans une Base de Données Projet. Chaque objet (prototype + instance) composant ce modèle du projet sera représenté par la structure de données. L'ensemble des prototypes d'un modèle du projet sera exprimé par la structure de données et en particulier par un dictionnaire de données.

Nous allons d'abord décrire rapidement ce dictionnaire de données (paragraphe II.1.2) puis nous donnerons la représentation de chacun des concepts du modèle de données dans la structure de données (prototype, attribut, facette, instance) au paragraphe II.2.

II.1.2) Dictionnaire des données :

Afin de mémoriser et de gérer la structure de données ainsi qu'un certain nombre d'éléments qui n'y figurent pas explicitement (noms d'attributs et de procédures, programmes correspondant aux procédures,...) et aussi afin de permettre l'accès aux données, le Modèle utilisera un dictionnaire des données qui comprendra les éléments suivants:

- une table des protoytpes.
- une table des attributs contraintes-intra.
- une table des attributs liens-binaires.
- une table des facettes.

- une Base de procédure et une Base des ensembles spécifiques.

Un schéma de ces tables ainsi qu'une illustration de leur utilisation est donnée au paragraphe II.3.

II.1.2.1) Table des prototypes :

Pour chaque prototype, la table des prototypes comprend les informations suivantes:

- le nom du prototype ainsi que son type (objet ou p.f.) fournis par le concepteur.

- le numéro du prototype, établi par le Modèle. Ce numéro est interne au Modèle et lui sert à repérer le prototype.

- des pointeurs, établis par le système permettant de représenter les attributs non descriptifs d'un prototype: pour les attributs contraintes-intra et liens-binaires il s'agira de pointeurs sur une table d'attributs, pour les attributs fils et partie-fonctionnelle le pointeur associé permettra d'accéder à une table de facettes enfin dans le cas de l'attribut père le pointeur donnera l'adresse du prototype père.

- des informations permettant la gestion de la table des prototypes.

II.1.2.2) Table des attributs contraintes-intra et liens-binaires :

Le descriptif d'un attribut dans la table des attributs comprendra:

- son nom, fourni par le concepteur.

- un pointeur géré par le système qui permettra d'accéder au facettes de cet attribut dans la table des facettes.

- des informations de gestion de la table des attributs (permettant en particulier de retrouver l'attribut suivant du même prototype).

- le numéro de l'autre prototype concerné par un lien-binaire (cette information ne figurera que dans la table des attributs liens-binaires).

II.1.2.3) Table des facettes :

Pour une facette d'un attribut cette table contient:

- le numéro de la procédure ou du prototype associés à cette facette.

- des informations de gestion de cette table des facettes (prochaine facette du même attribut, occupation de la table,...)

II.1.2.4) Base de procédures et Base d'ensembles :

- La Base de procédures est composée d'un ensemble de programmes compilés: elle est accessible par le biais d'une table qui contiendra le numéro de la procédure (ce numéro est unique et géré par le système), son nom de procédure, ainsi qu'un pointeur sur le programme correspondant dans la Base de procédures: la table joue le rôle d'un catalogue.
On pourra codifier les numéros de procédures afin de distinguer les procédures calculées des autres procédures.

- La Base d'ensembles contient tous les ensembles qui sont spécifiques: elle est équivalente aux tables des domaines utilisées dans un SGBD relationnel.
Pour chaque ensemble spécifique elle contient son nom, son numéro d'ensemble, le numéro de la procédure associée et la liste des éléments de l'ensemble.

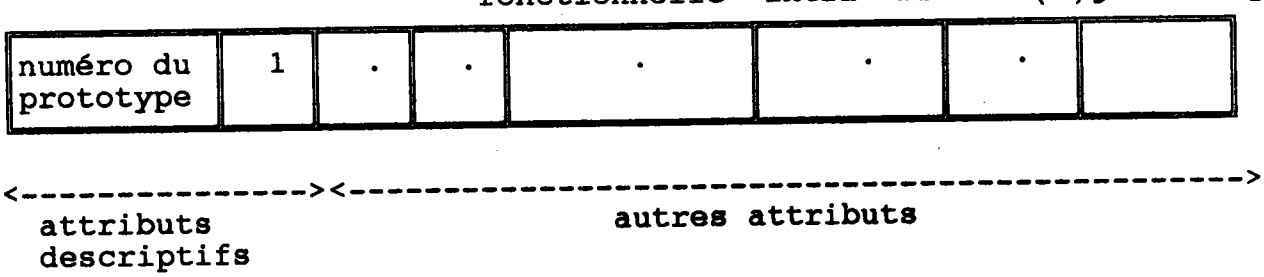
II.2) Représentation des différents constituants du modèle de données :

Nous allons décrire la représentation par la structure de données des différents éléments du modèle de données. Cette description est complémentaire, de celle du dictionnaire des données.

II.2.1) Représentation d'un prototype :

Rappelons qu'il existe 2 types de prototypes: les prototypes d'objets et les prototypes des parties-fonctionnelles.

* prototype d'objet :



* prototype de partie-fonctionnelle :

numéro du prototype	type père	fils	contrainte(s) intra	lien(s) binaire(s)
	2	.	.	.	.

<----------------><----------------------------------->
attributs descriptifs — autres attributs

On pourra noter que la valeur de l'attribut type (attribut descriptif) est "1" s'il s'agit d'un prototype d'objet et "2" dans le cas d'un prototype de partie-fonctionnelle.

Tous les champs correspondant aux attributs non descriptifs sont des pointeurs :

- la zone objet géométrique est un pointeur sur la représentation de cet objet dans le modèle géométrique : il sera un lien entre modèles dans le modèle générique.

- chacun des autres pointeur permet d'accéder à un certain nombre de listes qui représenteront l'attribut ou les attributs d'un type donné ainsi que les facettes associées.

II.2.2) Représentation des attributs :

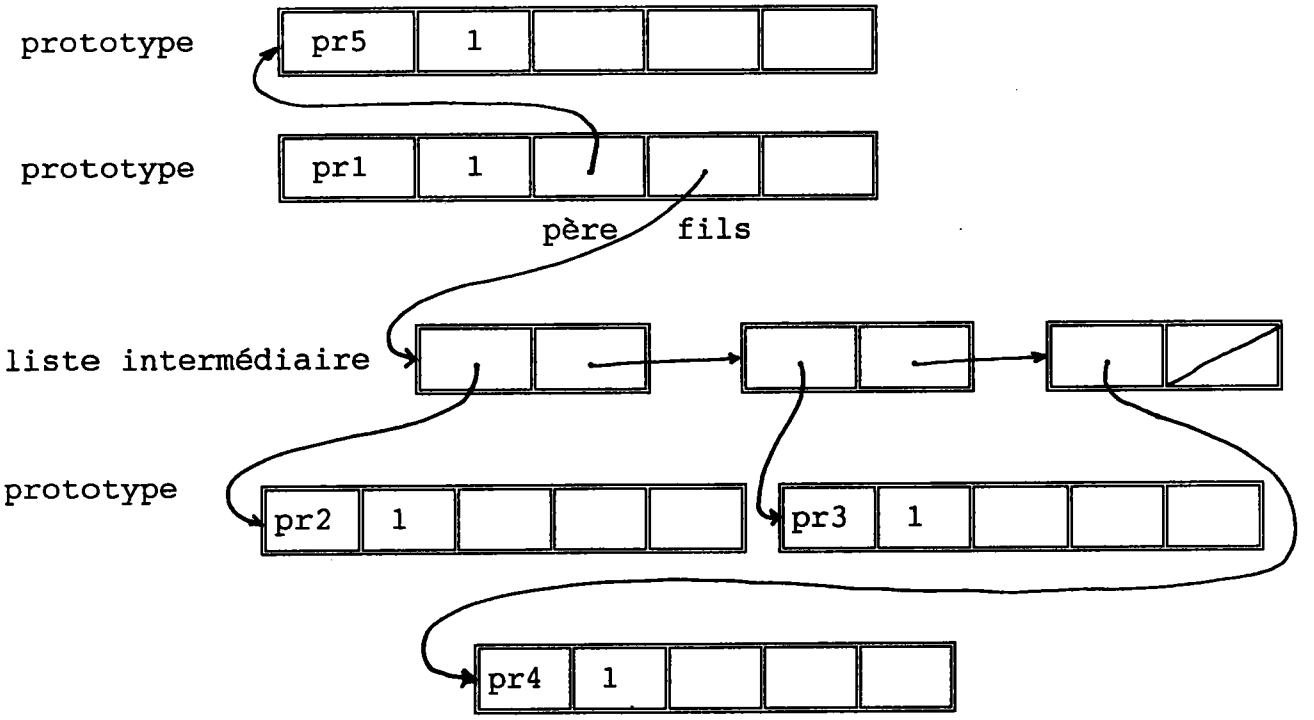
Chaque attribut descriptif est représenté par un champ dans la structure de données (voir paragraphe II.2.1). Tous les autres attributs sont décrits par le biais de structures dynamiques que nous allons présenter maintenant.

II.2.2.1) Attributs structurels :

Soit l'exemple suivant de prototype d'objet (le cas des prototypes de parties-fonctionnelles sera étudié au I.2.2.2).

<prototype> pr1
type : objet
père : pr5
fils : pr2, pr3, pr4

La structure de données associée est la suivante (pour des raisons de clarté sur le schéma sont décrits uniquements les liens structurels entre pr1 vers pr2, pr3, pr4).



Notons qu'à chacune des 3 facettes pr2, pr3, pr4 de l'attribut fils correspond un élément de la liste intermédiaire.

II.2.2.2) Attributs fonctionnels :

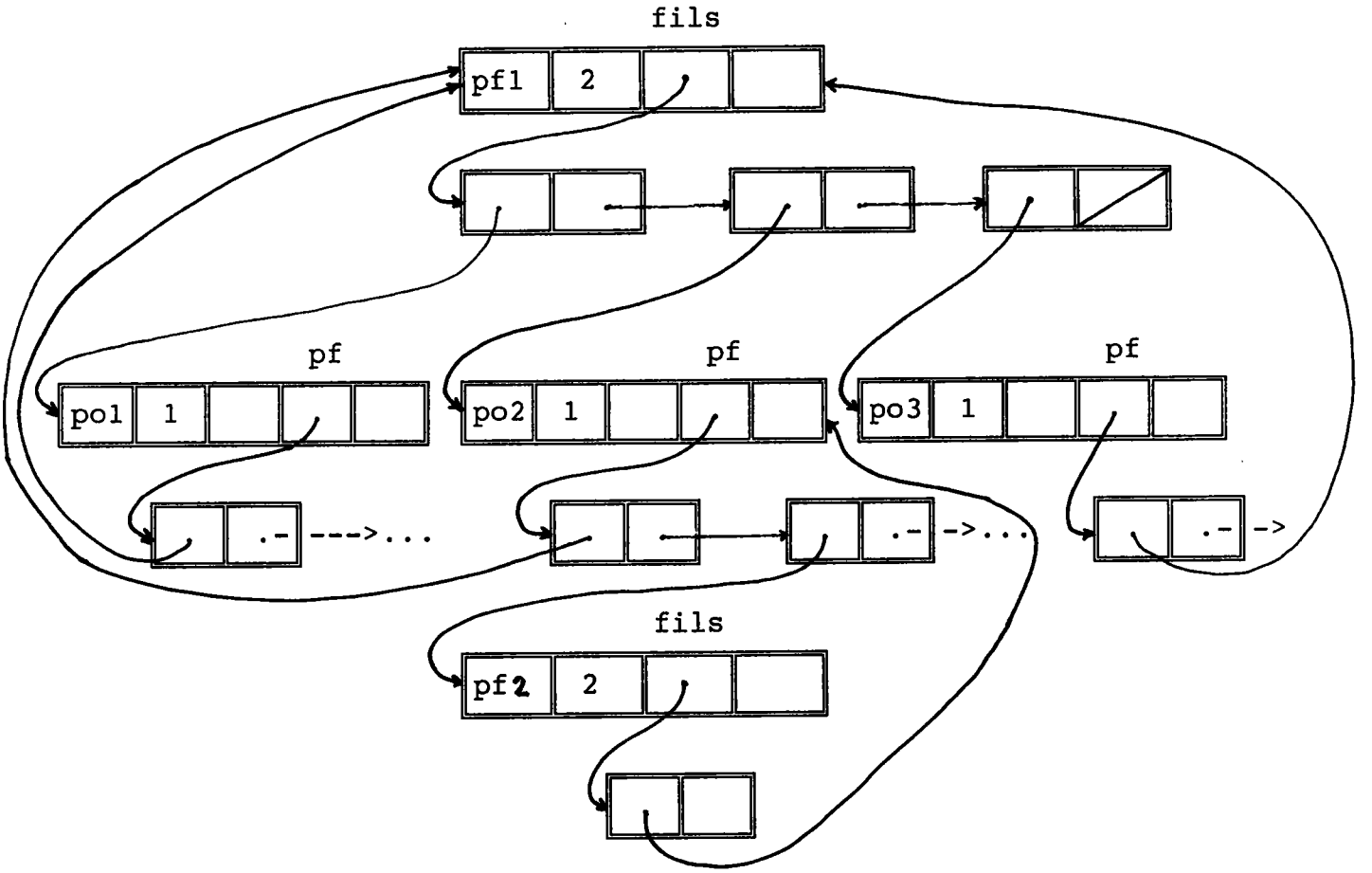
On peut rappeler que l'on ne définit pas de hiérarchies fonctionnelle dans le Modèle.

Soient les prototypes de parties-fonctionnelles suivantes:

<prototype> pf1
 type : partie-fonctionnelle
 fils : po1, po2, po3

<prototype> pf2
 type : partie-fonctionnelle
 fils : po2

La structure de données associée sera la suivante:



On peut faire la même remarque qu'au paragraphe II.2.2.1: dans le graphe ci-dessus, il existera différents types de noeuds; ceux qui correspondent au prototypes, et ceux qui correspondent aux facettes.

II.2.2.3) Attributs contraintes-intra :

Avant de donner la représentation des attributs contraintes-intra d'un prototype donné par le biais d'une structure de données dynamique nous allons d'abord définir par niveaux les différents types d'éléments que l'on trouvera dans cette structure de données.

Rappelons qu'un prototype contient de 0 à n attributs contraintes-intra décomposées en facettes; une facette correspond à une procédure munie de paramètres.
Nous ne représenterons pas ici les facettes d'attributs contraintes-intra faisant référence à un attribut d'un autre prototype.

a) niveau attribut contrainte-intra :

numéro attribut	type	facette	instance	attcalc	asuivant

numéro : entier représentant le numéro d'attribut attribut contrainte-intra pour un prototype donné.

type: booléen permettant de distinguer le type d'attribut (0 si non calculé, 1 si calculé).

facette: pointeur sur la première facette de l'attribut.

instance: pointeur sur la première instance de l'attribut.

attcalc : pointeur sur une liste d'identificateurs d'attributs calculés pour lesquels l'attribut courant est un paramètre.

asuivant: pointeur sur l'attribut contrainte-intra suivant.

* niveau facette :

procens	type	paramètres	fsuivante

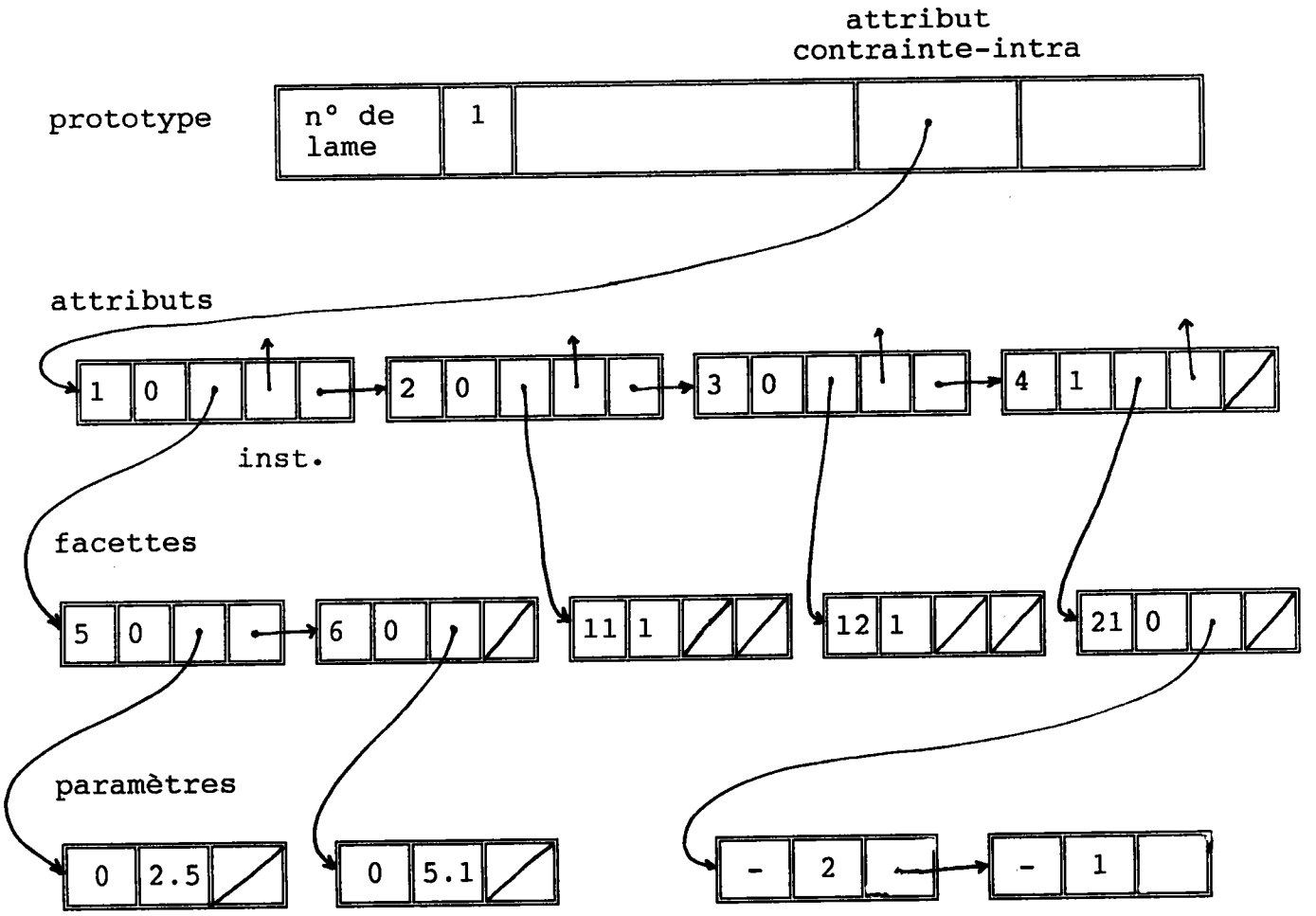
procens : correspond au numéro de la procédure ou de l'ensemble associé.
Dans le cas d'une procédure, ce numéro permet de distinguer une facette non calculée d'une facette calculée.

type : booléen (0 si procédure, 1 si ensemble)

paramètres: pointeur vers la liste des paramètres associés à la procédure.

fsuivante: pointeur sur la facette suivante correspond au même attribut.

Ce prototype contient 4 attributs contraintes-intra dont un calculé. La représentation de ce prototype dans la structure de données sera la suivante :



On pourra remarquer sur cet exemple que les numéros des procédures sont respectivement 5, 6, 21 et les numéros d'ensembles 11 et 12.

5 et 6 correspondent respectivement aux procédures supréel() et infréel().

11 et 12 désignent les ensembles Réel et Matière.

21 est le numéro de la fonction de calcul mult(): ses paramètres sont représentés par les numéros d'attributs contraintes-intra associés à volume et densité (c'est-à-dire 2 pour volume et 1 pour densité : ce sont les valeurs des champs "numéros attribut" du niveau attribut contrainte-intra).

d) Eléments du dictionnaire des données utilisées pour représenter le prototype lame :

Table des objets

nom du prototype	Numéro du prototype	Premier attribut cont-intra	Autres * Pointeurs	Attribut suivant
lame	1			
tournevis	2			

Table des attributs contraintes-intra

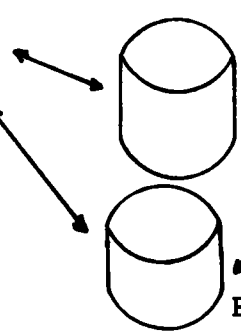
Nom de l'attribut	Première facette	Attribut suivant
densité		
volume		
matière		
poids		NIL

Table des facettes

Numéro de la procédure	Premier paramètre	Facette suivante
5		
6		NIL
11		NIL

Table des paramètres

Valeur	Suivant
2.5	NIL



* Il s'agit des pointeurs correspondant aux attributs (liens -binaires, partie-fonctionnelles, père et fils).

II.2.2.3) Attributs liens-binaires :

Tous les attributs liens-binaires sont des attributs calculés celà veut dire que chacun d'entre eux contiendra une facette calculée: cette facette calculée correspond à une fonction de calcul dont les deux paramètres sont respectivement un attribut contrainte-intra du prototype et un attribut contrainte-intra d'un autre prototype.

La représentation d'un attribut lien-binaire par la structure de données sera similaire à celle d'un attribut contrainte-intra. Les seules différences dans le cas d'un attribut lien-binaire seront :

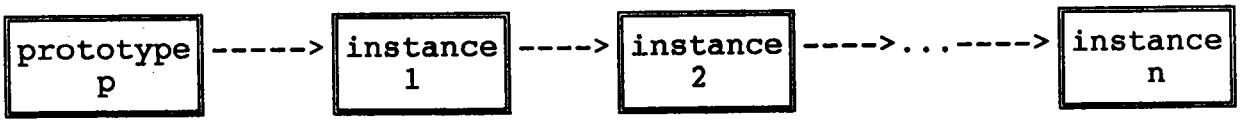
- rajouter au niveau attribut un champ contenant le numéro du second attribut auquel il est fait référence dans le lien.

- au niveau paramètres associés à la facette calculée, fixer par convention que le premier numéro correspondra à l'attribut contrainte-intra du prototype et le deuxième à l'attribut contrainte-intra de l'autre prototype participant au lien : à la différence des contraintes-intra calculées pour lesquelles le nombre de paramètres est variable, dans le cas des liens-binaires on aura exactement deux paramètres.

Pour plus de détails sur cette représentation des attributs liens-binaires se reporter à l'Annexe 1.

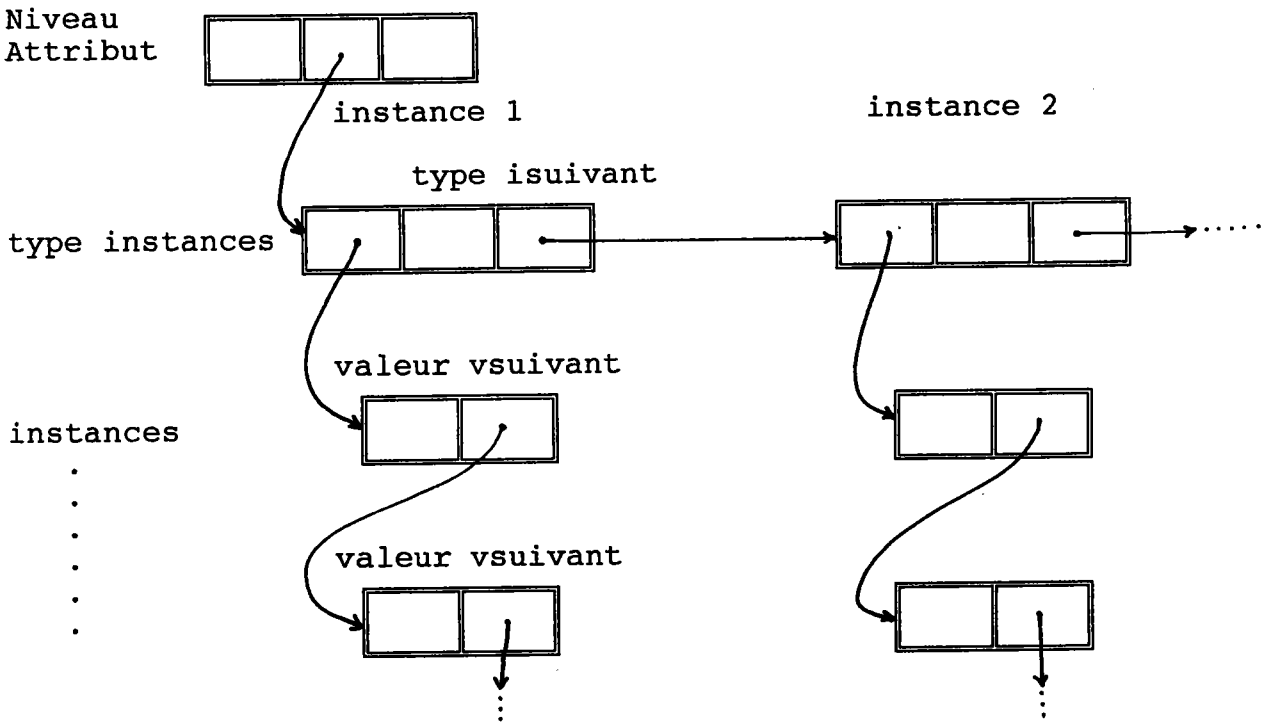
II.2.3) Instances :

A un instant donné d'un projet de CAO, il peut exister en parallèle différents modèles du projet correspondant à différents essais de conception. Ces modèles du projet auront en commun des prototypes des objets qui les composent identiques (se reporter au I.2.5). Donc au niveau des objets, pour un prototype donné il pourra exister plusieurs instances différentes. On peut représenter celà par le schéma qui suit où pour un prototype p on aura n instances: n correspond au nombre de modèles du projet.



Une instance d'un prototype est constitué de l'ensemble des occurences de ses attributs contraintes-intra et liens-binaires. Une occurence d'attribut pourra être une liste d'entiers ou réels (liste souvent réduite à un élément) ou encore d'éléments d'un ensemble spécifique.

Représentation générale des instances d'un attribut



pvaleur : pointeur sur la première correspondant à l'instance.

type : désigne le type de l'instance
type = 0 si entier
type = 1 si réel
type = 2 si élément d'un ensemble spécifique.
(cet ensemble a été défini par une facette au niveau attribut).

valeur : élément de l'ensemble des valeurs admissibles (réel, entier, ensemble spécifique).
Dans le cas d'un ensemble spécifique (type = 2), valeur contiendra le numéro d'ordre de l'élément dans la liste des constituants de l'ensemble.

isuivant : pointeur sur l'instance suivante de l'attribut.

vsuivant : pointeur sur la valeur suivante de l'instance.

Exemple des instances d'attributs :

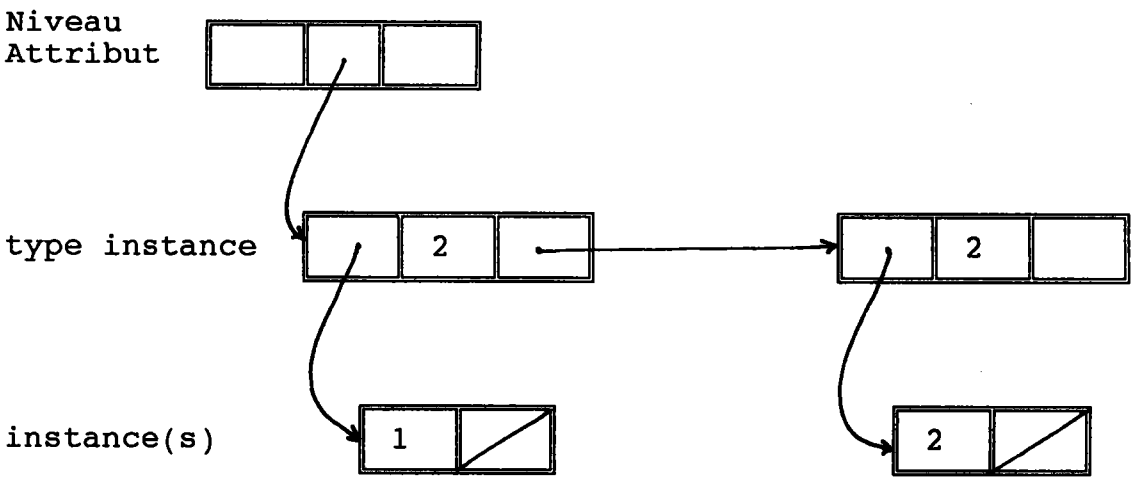
Nous allons donner la représentation d'attributs de 2 objets lame de tournevis dont la description des instances est la suivante :

représentation dans le modèle de données

<instance 1> lame
densité : 2.7
volume : 0.1
matière : aluminium
poids : 0.27

<instance 2> lame
densité : 5
volume : 0.2
matière : acier
poids : 1.

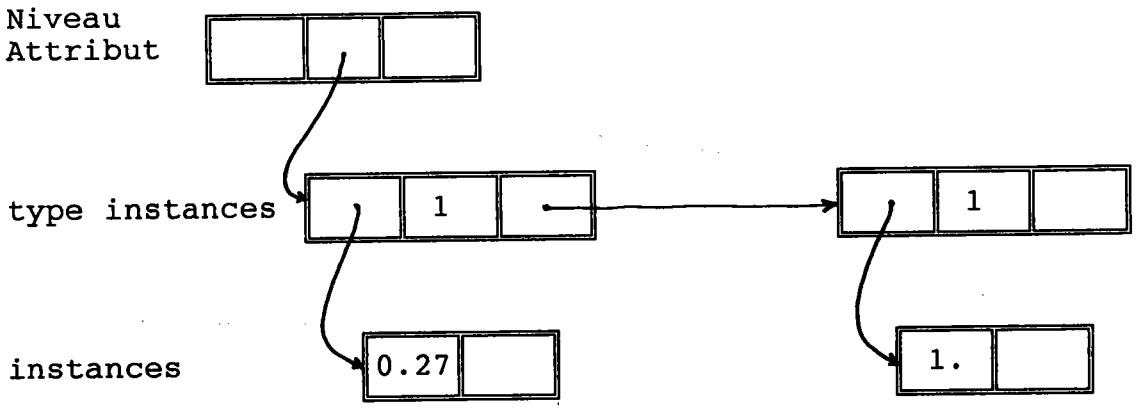
représentation des instances de l'attribut matière :



Rappelons que l'ensemble des occurences possibles de l'attribut matière est Matière défini par :
Matière = { aluminium, acier, bakélite }.

Matière est un ensemble spécifique dont chaque élément correspond à un numéro d'ordre dans la Base d'ensembles: ainsi aluminium, acier et bakélite sont associés respectivement aux entiers 1, 2, et 3.

Représentation des instances de l'attribut poids :



III. Attachement et déclenchement procéduraux : (1)

III.1) Le problème d'attachement et de déclenchement :

Nous avons vu lors de la définition d'un prototype que chaque facette de ses attributs contraintes-intra ou liens-binaires pouvait s'exprimer par le biais de procédures (paragraphe I.2.2.4 et paragraphe I.2.3). Ces procédures sont de deux types : les procédures contrôlant l'appartenance à un ensemble donné (facettes non calculées) et les procédures de calcul (facettes calculées).

Le premier objectif de cette partie est d'étudier commment associer lors de la création ou de la modification d'une facette une procédure à cette facette et plus particulièrement quelle en sera la répercution sur la structure de données : le mécanisme utilisé pour répondre à cet objectif est l'attachement procédural.

Le deuxième objectif de cette partie est de définir l'utilisation de ces procédures lors d'une instanciation d'un attribut contrainte-intra ou lien binaire : il s'agit là du déclenchement procédural.

Nous allons d'abord décrire les outils utilisés (paragraphe III.2) puis le fonctionnement des mécanismes d'attachement et de déclenchement procéduraux (paragraphe III.3), et enfin l'architecture des procédures (paragraphe III.4).

(1) Dans ce paragraphe, les seuls attributs dont on parlera seront soit des attributs contraintes-intra ou liens-binaires.

III.2) Les outils :

III.2.1) Lien entre Base de procédures et Base d'ensembles :

Toute facette d'un prototype s'exprime par le biais d'une procédure dans la structure de données . Dans ce modèle de données, on associe à chaque facette un procédure soit de façon explicite (cas des facettes caculées ou encore infréel(), supréel(),...) ou de façon implicite par le biais d'un ensemble (Réel, Entier, Matière,...). Dans le cas où dans le modèle de données, une facette d'un attribut est définie par le biais d'un ensemble, la procédure associée à cet ensemble sera une fonction booléenne vérifiant l'appartenance d'une occurence de l'attribut à cet ensemble. Dans la structure de données cette facette sera mémorisée par le numéro de l'ensemble et les instances de l'attribut par des éléments de l'ensemble ou des numéros d'ordre de ces éléments (voir exemple paragraphe II.2.2.3.d).

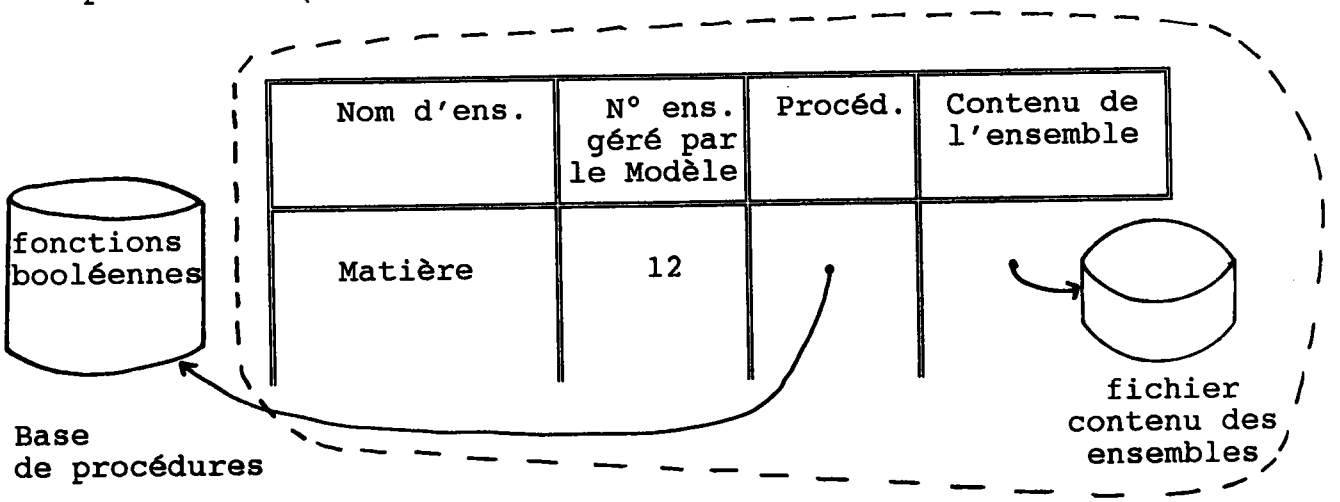
Parmi les procédures testant l'appartenance à un ensemble, on peut distinguer deux catégories :

- celles liées à des ensembles de base comme Réel, Entier, Chaine (taillemax).

- celles liées à des ensembles spécifiques comme Matière.

Les ensembles de base et donc les procédures associées sont connues du Modèle et n'impliquent aucune gestion particulière.

La Base d'ensembles contiendra tous les ensembles spécifiques: chacun de ces ensembles sera constitué d'une liste finie d'éléments (stockée dans le fichier contenu des ensembles) et sera asocié à une seule procédure par le biais de son numéro de procédure. (voir schéma ci-dessous)



III.2.2) Compléments sur la structure de données :

Nous allons nous intéresser ici uniquement aux éléments de la structure de données permettant la déduction par l'intermédiaire des attributs calculés.

Pour tout attribut il faudra donc pouvoir représenter :

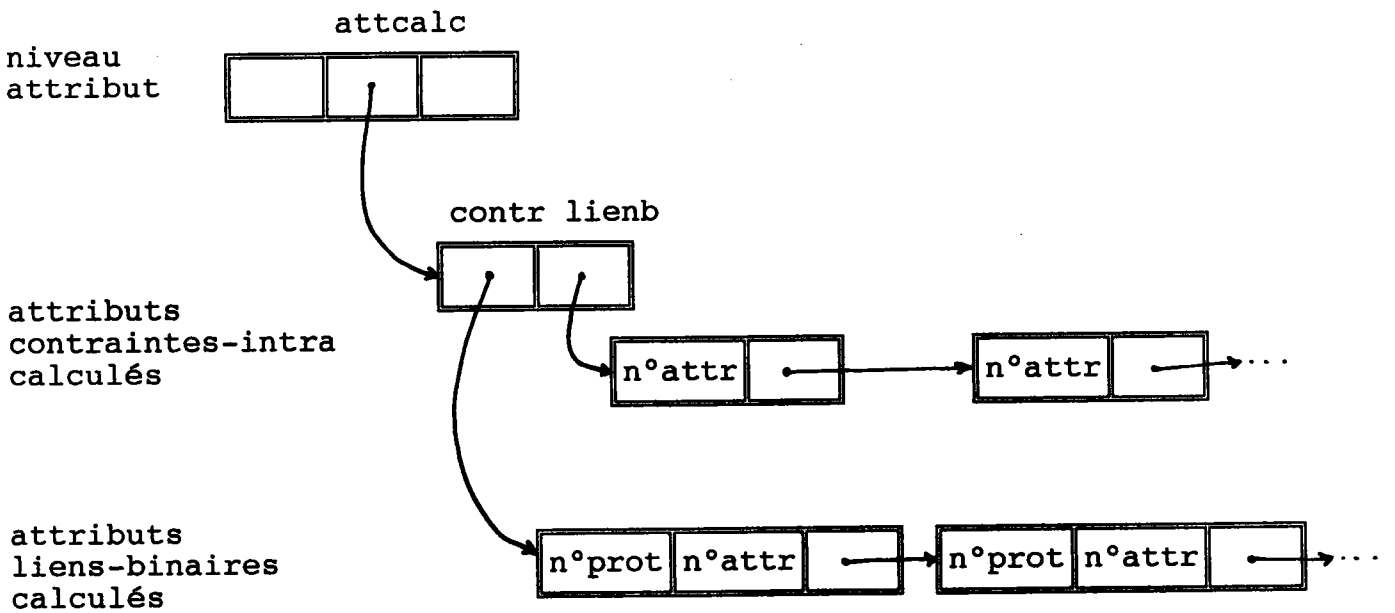
- dans le cas où cet attribut est calculé, la liste des attributs qui seront paramétres de sa facette calculée.

- la liste des attributs calculés pour lesquels l'attribut est paramètre de leurs fonctions de calcul.

On pourra rappeler que seuls les attributs contraintes-intra peuvent être paramètres d'un attribut calculé.

III.2.2.1) Représentation de la liste des attributs calculés pour lesquels les instances de l'attribut sont paramètres :

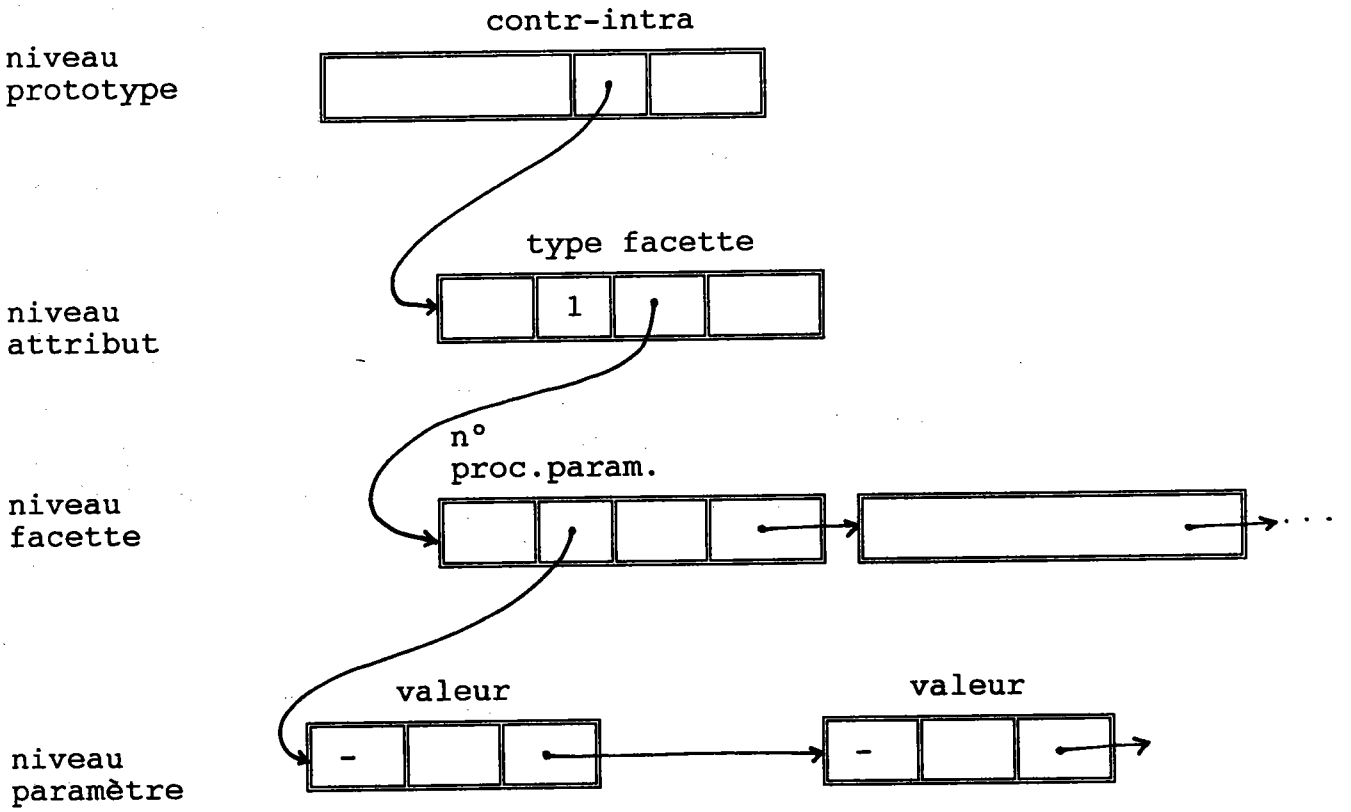
On reprend pour celà la représentation d'un attribut (se référer au paragraphe II.2.2.3.a) et on s'intéresse plus particulièrement au champ attcalc.



Cette représentation permet de connaître à tout moment les attributs contraintes-intra du prototype (pointeur contr) et les attributs liens binaires (pointeur lienb) concernés.

III.2.2.2) Représentation d'un attribut calculé :

Nous allons représenter un attribut contrainte-intra calculé. Cet attribut possédera donc une facette calculée et les paramètres de la fonction de calcul seront d'autres attributs contraintes-intra du même prototype. La prise en compte d'un attribut calculé pour un prototype donné se représentera de la façon suivante :



Interprétation :

Le champ type au niveau attribut indique qu'il s'agit d'un attribut calculé. Au niveau facette, la première facette représente le numéro de la fonction de calcul (champ numéro procédure). Enfin au niveau paramètres les champs valeur contiendront dans l'ordre les numéros d'attributs contraintes-intra paramètres de la fonction de calcul.

Note :

La représentation d'un attribut lien-binaire sera semblable à celle des attributs contraintes-intra calculés avec quelques petites variantes (voir paragraphe II.2.3). C'est la raison pour laquelle nous ne la donnons pas ici.

III.3) Mécanismes de l'attachement et du déclenchement procéduraux :

III.3.1) Les procédures concernées :

L'ensemble de procédures utilisables dans une phase de conception est mémorisé dans la Base de procédures : chaque procédure est identifiée par son numéro de procédure qui est unique, elle sera définie par un nom de procédure et une liste de paramètres avec leurs types respectifs.

Les procédures de base sont intégrées dans le Modèle alors que les procédures spécifiques sont créées en dehors du processus de conception d'un modèle du projet par l'intermédiaire d'un système de développement classique (nous étudierons l'architecture de ces procédures au paragraphe III.4). La création d'une procédure dans le cadre d'une conception imbriquée nécessiterait d'intégrer dans le Modèle un analyseur syntaxique.

Parmi les procédures définissant les facettes des attributs, certaines sont exprimées explicitement dans la structure de données par le biais de leur numéro de procédure, les autres seront représentées indirectement par le numéro de l'ensemble pour lesquelles elles seront la fonction booléenne d'appartenance (se reporter au paragraphe III.2.1)

On pourra aussi distinguer les procédures pour lesquelles aucun paramètre ne sera représenté dans la structure de données (fonctions booléennes d'appartenance aux ensembles de base : Entiers, Réels, chaîne (taillemax),...) des procédures avec paramètres dans la structure de données (appartenance à un ensemble, un intervalle, fonctions de calculs). Malgré cette distinction les mécanismes d'attachement et de déclenchement seront semblables. Nous allons étudier maintenant ces mécanismes dans le cas d'une fonction de calcul pour un attribut contrainte-intra. Cette fonction de calcul pourra se découper en 3 parties: les parties 1 et 2 destinées aux mécanismes d'attachement et de déclenchement et la partie calcul (voir schéma III.4).

III.3.2) Attachement procédural :

Nous allons décrire un protocole d'éxecution tour à tour de deux programmes : le programme du Modèle et la procédure de calcul concernée par l'attachement dans le cas de la création d'une facette.

MODELE	PROCEDURE DE CALCUL
cas de création d'une facette calculée. début module - consultation de la liste des procédures de calcul (dans la Base de procédures) et choix de la procédure cherchée (mémorisation de son numéro de procédure). si aucune procédure ne convient fin du module. - écriture dans une variable du numéro du prototype et du numéro d'attribut. ---> exécution de la procédure choisie. -sauvegarde des informations concernant l'attachement procédural dans la structure de données : (1) *mise à jour de la zone type de l'attribut au niveau attribut. *sauvegarde du numéro de procédure au niveau facette. *sauvegarde des différents attributs paramètres de la procédure au niveau paramètres. *mise à jour d'un drapeau pour chaque attribut du prototype paramètre de la procédure. fin module	PARTIE 1 -lecture du numéro de prototype et du numéro d'attribut. -saisie des différents noms d'attributs qui seront paramètres de la procédure. Les controles suivants seront exécutés pendant la saisie des attributs : * existence de chacun des attributs par le biais du dictionnaire des données : si un des attributs n'existe pas, fin du module. * compatibilité des domaines des valeurs admissibles des attributs avec les types des paramètres correspondant de la proced. -écriture dans une variable des numéros d'attributs paramètres de la procédure. <---- retour Modèle.

(1) pour illustrer cette mise à jour se reporter à la représentation de l'attribut calculé poids du II.2.2.3.d.

(2) se reporter au II.2.2.1.

III.3.3) Déclenchement procédural :

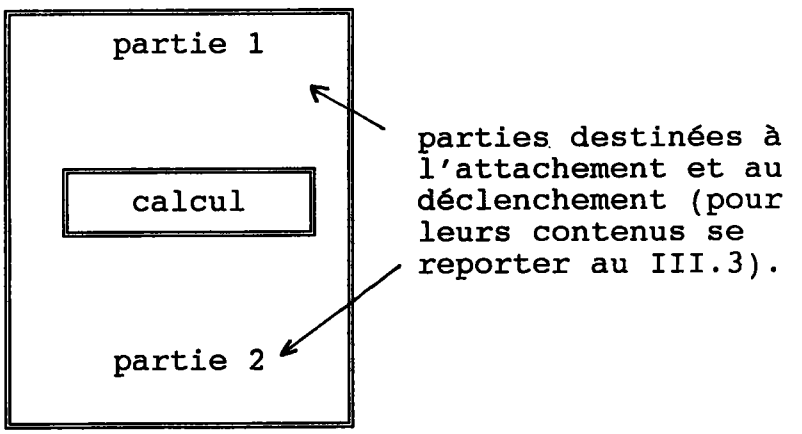
Nous allons décrire le déclenchement procédural impliquant l'instanciation d'un attribut calculé. Ce module comprend les exécutions tour à tour de deux programmes : Modèle et la procédure de calcul concernée par le déclenchement procédural dans le cas de l'instanciation explicite d'un attribut.

MODELE	PROCEDURE DE CALCUL
cas de l'instanciation (ou de la modification) explicite d'un attribut. Cet attribut est il paramètre d'une procédure calculée? (contrôle dans la structure du champ attcalc au niveau attribut). si oui début module pour chaque attribut calculé lié à cette instanciation faire. Si tous les autres attributs paramètres de la procédure de calcul sont instanciés alors Stockage des valeurs des paramètres dans une variable. ----> exécution de la procédure. écriture du résultat du calcul dans la partie instance de l'attribut calculé. fsi fpour fin module	PARTIE 1 lecture de la liste des paramètres exécution de la partie calcul de la procédure écriture des résultats dans une variable PARTIE 2 <---- retour Modèle

III.4) Architecture d'une procédure de calcul :

Cette procédure est écrite dans un langage évolué (en l'occurence TURBO-PASCAL), elle se trouvera dans une Base de procédures ou éventuellement pour une procédure spécifique à l'utilisateur sous son directory.

Une procédure de calcul est découpée logiquement en deux parties: la partie calcul et la partie de gestion de l'attachement et du déclenchement procéduraux. On pourra la représenter de la façon suivante :



Les problèmes essentiels de l'attachement et du déclenchement sont les communications de données entre programmes et retour au Modèle dans le contexte de l'appel à la fin de la procédure de calcul.

III.4.1) Communications de données :

L'ensemble des données à transmettre entre le Modèle et la procédure de calcul est décrit par le tableau suivant :

sens / module	Modèle-->Proc. de calcul	Proc. de calcul-->Modèle
attachement	numéro de prototype et numéro d'attribut	numéros du prototype et de ses attributs paramètres de la procédure.
déclenchement	liste des données paramètres de la procédure	liste des données résultat de la procédure de calcul.

Afin de permettre cette communication de données, une solution envisageable serait de créer pour chaque procédure appelée une zone déclaration de données identique à celle du Modèle suivie d'une zone déclaration de données spécifiques à la procédure. Mais comme la plupart des variables du Modèle sont des variables dynamiques l'espace mémoire qui leur sera alloué est réutilisé lors de l'exécution de la procédure.

Parmi les solutions qui permettent la transmission des données entre programmes on peut citer les suivnates :

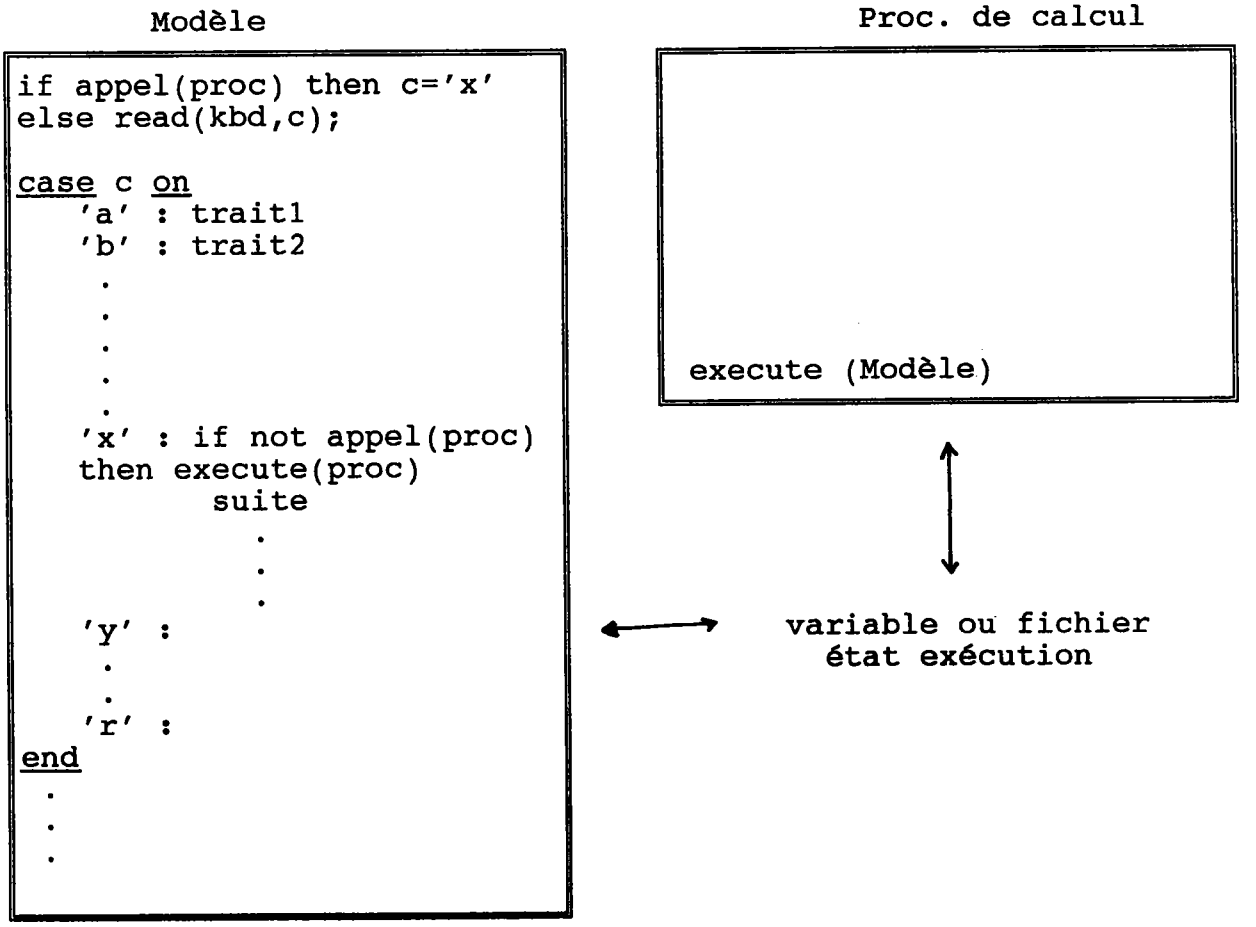
- déclarer la structure de donnée dynamique du Module à une adresse fixe en mémoire centrale.

- sauvegarder cette structure dynamique dans des tableaux ou encore des fichiers lors de chaque exécution d'un programme de calcul.

Cela implique que les variables auxquelles on a fait référence pour la communication des données au III.3 peuvent aussi être des enregistrements d'un fichier.

III.4.2) Retour au Modèle en fin d'exécution de la procédure :

Une solution de retour au Modèle dans le même contexte peut se représenter par le schéma qui suit :



Cette solution permet grâce aux informations contenues dans "état exécution" de savoir lors de toute exécution de Modèle s'il s'agit ou non d'une exécution efféctuée par une procédure de calcul. Dans le cas d'une exécution faite par une procédure de calcul il faut de plus connaître le cadre de cette exécution. Selon qu'il s'agit d'un attachement ou d'un déclenchement procédural le retour dans Modèle ne se fera pas au même endroit.

Dans le schéma ci=dessus le retour se fait au niveau de suite.

Remarques:

- La solution envisagée est alourdie très rapidement en fonction du nombre d'imbrications du programme Modèle.

- Un problème identique à celui évoqué précédemment se pose lors de l'exécution de la procédure de calcul : il faut savoir s'il s'agit d'une exécution effectuée dans le cadre d'un attachement ou dans le cadre d'un déclenchement.

III.4.3) Conclusions sur les problèmes d'attachement et de déclenchement :

La réalisation d'un système intégrant les mécanismes d'attachement et de déclenchement procéduraux en Turbo-Pascal ne pose pas de problèmes mais impliquera toutefois un certain nombre de contraintes qui risquent d'alourdir ce système.

Dans la maquette qui a été réalisée en TURBO-PASCAL il existe un autre problème que nous n'avons pas signalé jusque là qui est celui de la limitation de la taille mémoire en code à 54KO ce qui est très gênant lorsqu'il s'agit de développer des systèmes CAO gourmands en place mémoire.
Toutes ces raisons tendent vers un choix du MS-Pascal qui permet la compilation séparée (d'où pas de problèmede communication de données, de retour au Modèle par chainages de programmes) et qui autorise en mémoire 450 Ko en exécutable.

IV. Liens entre le Modèle et SACADO :

Nous allons maintenant étudier la façon dont le Modèle s'intègre dans SACADO aussi bien au niveau des informations qu'il gère que de son architecture.

IV.1) Les informations :

Les différents types d'informations gérées par le Modèle sont des objets (définis par des propriétés non géométriques) et des procédures (certaines d'entre elles sont liées à des ensembles). Nous allons faire la classification suivante des informations par rapport à la notion de projet :

* les informations liées à un projet donné :

  - le cahier des charges : certains éléments sont pris en compte par les parties fonctionnelles qui seront créées avant la conception du premier objet.

  - les modèles des projets : mémorisés sous formes de versions.

  - les connaissances : par le biais de procédures ou d'ensembles spécifiques au domaine de conception ou même au concepteur.

* les informations communes à plusieurs projet :

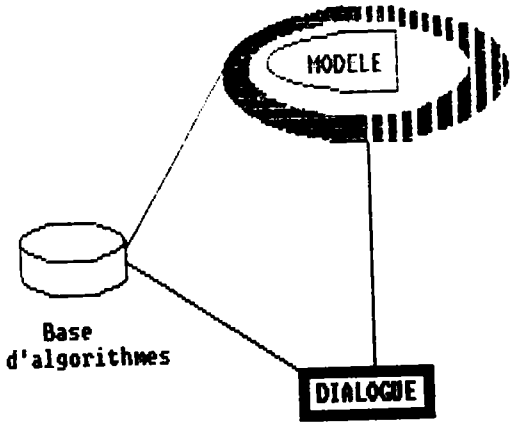
  - les ensembles et les procédures de bases.
    En ce qui concerne les catalogues de pièces, ceux-ci ne sont pas gérés par le Modèle, mais comme ils ont une structure identique aux objets du Modèle leur prise en compte est envisageable.

IV.2) Architecture du Modèle par rapport à SACADO :

Le modèle générique possède différents modèles structurels : modèle de données défini dans le cadre du Modèle est l'un d'entre-eux. A chaque objet crée par le Modèle correspondra un objet dans le modèle géométrique grâce à l'attribut objet géométrique (se reporter au II.2.1).

L'architecture du Modèle par rapport à celle de SACADO est représentée par le schéma qui suit :

MODELE GENERIQUE



IV.3) Le Modèle et le modèle générique :

Un objet possède une structure liée à sa structure géométrique. Une autre structuration est possible grâce aux parties-fonctionnelles. Dans ce Modèle tout objet devra vérifier un ensemble de règles. La plupart de ces règles appartiennent à l'une des catégories suivantes :

- fortes : vérification de la hiérarchie structurelle, appartenance à un domaine pour une contrainte-intra non calculée, consistance,...

- faibles : instanciation d'un lien-binaire, héritage,...

Certaines sont immédiates (la plupart des contraintes fortes) d'autres sont différées.
Le Modèle ne gère que des éléments de la Base de Données Projet; les seuls éléments de connaissances sont pris en compte par le biais de procédures appartenant à la Base d'algorithmes. La Base de données Projet est constituée pour un projet donné de versions (valides ou de travail) de ce projet.

PARTIE3 :

EXEMPLES D'UTILISATION DU MODELE

Dans cette partie sont présentés trois exemples de conception dans le domaine de la mécanique.

Tout d'abord dans le premier paragraphe est décrite la démarche générale utilisée dans un processus de conception mécanique ainsi que les objectifs de la présentation des trois exemples.

Les exemples de conception sont décrits dans le deuxième paragraphe et pour chacun d'entre eux y sont définis les éléments pouvant être gérés par le Modèle.

Le dernier paragraphe décrit le cadre d'utilisation du Modèle et aussi de SACADO dans un processus de conception.

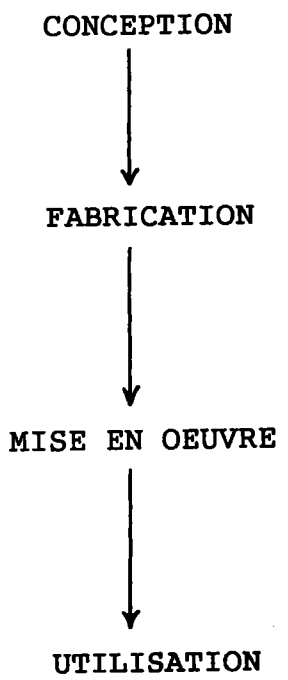
I) La démarche de conception en mécanique : [GAR-83], [GAR-86a]

I.1) Description de la phase de conception :

I.1.1) Notion de vie d'un objet industriel :

Un objet industriel, quel qu'il soit (voiture, meuble, immeuble, rail, roulement, disjoncteur) a une vie à l'intérieur d'une entreprise (ou d'un ensemble d'entreprises si l'on considère les sous-traitants par exemple) qui vade sa conception a sa fabrication et une vie à l'extérieur de l'entreprise qui va de la livraison et la mise en oeuvre à l'utilisation effective de l'objet dans un certain contexte ([GAR-83]).

On peut résumer de façon macroscopique la vie d'un objet industriel par le schéma suivant :



Dans la conception de tout objet industriel il faudra tenir compte des contraintes de fabrication (machines, stocks,...), de sa mise en oeuvre (assemblages,...) et de son utilisation (conditions de fonctionnement,...).

I.1.2) La phase de conception :

Le déroulement d'une phase de conception varie selon le domaine de conception (mécanique, électronique,...) et selon le contexte de conception (produit nouveau, reprise d'une conception antérieure, utilisation de catalogues,...).

Nous allons nous intéresser à la conception d'objets industriels en mécanique. Une phase de conception d'un tel objet peut se décomposer en différentes tâches successives qui sont les suivantes :

- établissement du cahier des charges (revient à définir une liste de contraintes que devra vérifier l'objet industriel).

- étude cinématique (du fonctionnement, des liaisons,...).

- étude statique (dimensionnement).

De même que dans toute démarche de conception (au sens général du terme), il arrive souvent que l'on soit obligé de revenir à la tâche précédente : par exemple si suite au dimensionnement de pièces on se rend compte qu'il n'existe pas les pièces correspondantes dans les catalogues on sera obligé de revenir à l'étude cinématique ou même au cahier des charges.

I.2) Objectifs de la description des exemples :

La description dans le prochain paragraphe de trois exemples a les objectifs suivants :

- présenter la démarche suivie dans une phase de conception et les problèmes rencontrés.

- pour chaque exemple de conception étudier les éléments de conception pouvant être gérés par le Modèle : celà revient à recenser les informations pouvant être représentées par le modèle de données et à confronter les fonctionnelités du Modèle avec le processus de conception.

II) Description des exemples :

II.1) Premier exemple : la bicyclette.

II.1.1) Présentation de l'exemple :

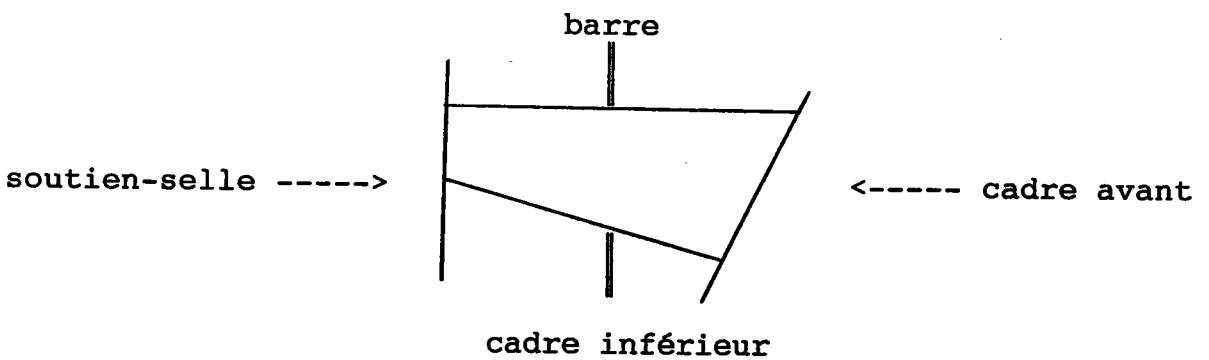
Les fonctions qu'aura à réaliser une bicyclette sont les suivantes :

- transporter
- déplacer
- guider
- transmettre
- maintenien de l'ensemble

On définit successivement les objets qui correspondent à ces différentes fonctions (chaque fonction sera représentée par une partie-fonctionnelle dans le Modèle).

* Fonction "maintien de l'ensemble" :

Cette fonction représente la création du cadre.



Ces 4 objets appartiennent à la même partie-fonctionnelle "maintien de l'ensemble".

Chaque objet aura ses propres propriétés : par exemple le poids et la longueur de la barre appartiendront à un certain intervalle tandis que leurs angles avec l'horizontale auront une valeur fixe (notion de contrainte intra-objet dans le Modèle).

Ces objets pourront aussi être liés entre-eux : ainsi l'angle existant entre l'objet "soutien-selle" et l'objet "barre" devra être inférieur à 80 degrés. (notion de lien binaire).

Enfin l'ensemble constitué par ces 4 objets peut être considéré comme un objet d'ordre supérieur appelé "cadre" (dans le modèle

cette création est possible par le biais des liens structurels).

L'objet "cadre" pourra en dehors de ses liens structurels avoir d'autres propriétés comme par exemple que son poids soit inférieur à 3 kg.

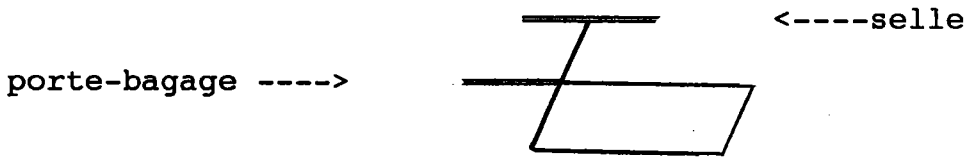
Après la création du cadre il est toujours possible de rajouter, modifier ou supprimer ses constituants : ainsi on pourrait par exemple supprimer l'objet "barre" ou modifier la contrainte concernant la longueur de l'objet "cadre inférieur".

* Fonction "transporter" :

La prise en compte de cette fonction revient à la création de la partie fonctionnelle "transporter".

L'objet "barre" déjà créé précédemment appartiendra à cette nouvelle partie fonctionnelle.

Il y aura de plus création de 2 nouveaux objets appelés "porte-bagage" et "selle".



Supposons que les contraintes que devront vérifier ces 2 objets soient les suivantes :

Pour "selle" : épaisseur $\in$ [1cm, 2cm]
longueur > 20cm
poids < 1kg

Pour "porte-bagage" : longueur $\in$ [18cm, 22cm]
largeur $\in$ [10cm, 14cm]

(On remarquera que ces propriétés correspondent à des contraintes intra-objet dans notre modèle de données).

Pour "porte bagage" et "selle" : angle entre ces 2 objets nul.

Pour "selle" et "barre" : angle entre ces 2 objets nul.

(Ces 2 propriétés correspondent à des liens binaires dans notre modèle).

* Fonction "déplacer" :

La réalisation de cette fonction se fait par le biais de la création des 2 roues : "roue arrière" et "roue avant".

Ces 2 objets ont apparemment les mêmes propriétés (poids, rayon, etc...) mais la définition de contraintes par rapport à d'autres objets permet de mettre en évidence le fait que ces 2 objets sont différents : par exemple la distance entre la "roue arrière" et le "porte-bagage" doit être supérieure à 5cm ou encore les valeurs de leurs torseurs cinématiques sont différentes (nous reviendrons sur cette notion de torseur cinématique dans les exemples qui suivent).

* Fonction "transmettre" :

Le processus de création des objets qui réaliseront cette fonction est identique à ce que nous avons vu pour les 3 fonctions précédents et nous ne feront ici que citer les noms des objets.

Il s'agit de : -"pédalier".

-"pignons".

* Fonction "guider" :

Même remarque que précédemment. Les objets crées seront ici :

- "guidon"

- "fourche"

De plus l'objet "roue avant" déjà créé contribuera à cette fonction. On pourra par contre lui ajouter de nouvelles propriétés comme par exemple la valeur de son torseur cinématique par rapport à l'objet "fourche".

D'autre part les 3 objets réalisant la fonction guider permettent de créer un nouvel objet d'ordre supérieur appelé "fourche" par le biais de liens structurels (se reporter au schéma qui suit).

II.1.2) Evolution du modèle du projet pendant la conception :

II.1.2.1) Création des parties fonctionnelles :

Les cinq fonctions à réaliser par la bicyclette se traduisent par la création des prototypes de parties-fonctionnelles suivants :

```
<prototype> transporter
 type : partie-fonctionnelle

<prototype> déplacer
 type : partie-fonctionnelle

<prototype> guider
 type : partie-fonctionnelle

<prototype> transmettre
 type : partie-fonctionnelle

<prototype> maintien de l'ensemble
 type : partie-fonctionnelle
```

II.1.2.2) Création des éléments de la partie-fonctionnelle "maintien de l'ensemble" :

a) Création des 4 prototypes :

La conception des objets correspondant à la fonction "maintien de l'ensemble" revient à la création des prototypes suivants (nous ne donnerons ici que les descriptions des prototypes "barre" et "soutien-selle", on suppose que les 2 autres prototypes n'ont pour le moment que leurs attributs descriptifs et parties-fonctionnelles comme seuls attributs).

```
<prototype> soutien-selle
 type : objet
 partie-fonctionnelle : maintien de l'ensemble
 poids : intervreel(0.25, 0.30)
 longueur : infreel(0.5)
 angle : val(65)
 diff-angle : =différence(angle,(barre,angle))
              infreel(80)

<prototype> barre
 type : objet
 partie-fonctionnelle : maintien de l'ensemble
 poids : intervreel(0.2,0.25)
 longueur : infreel(0.5)
 angle : val(0)
 diff-angle : =différence((soutien-selle,angle),angle)
              infreel(80)
```

Notes :
Les procédures utilisées comme facettes d'attributs doivent exister dans le Modèle ou la Base de procédures pour que l'on puisse faire référence à elles : val() est la procédure valeur qui oblige tous les attributs angle à une occurence donnée, intervréel() est une fonction booléenne d'appartenance à un intervalle réel donné, différence() est un fonction de calcul correspondant à l'opérateur arithmétique différence.

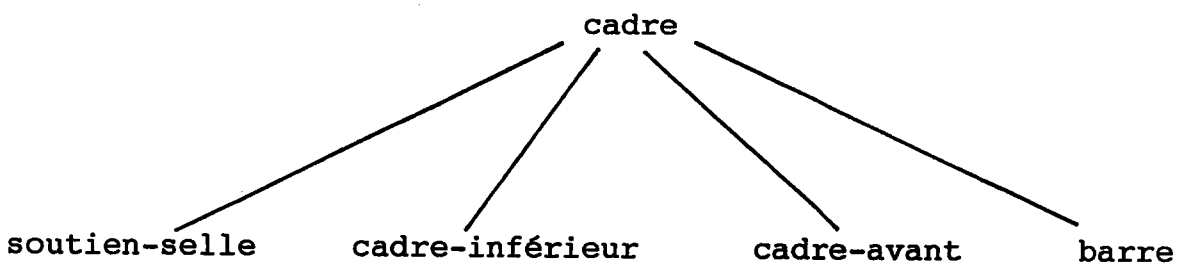
b) Création de cadre :

Après la conception des prototypes des 4 objets de la partie-fonctionnelle "maintien de l'ensemble", on crée un prototype appelé "cadre" qui sera composé de ces 4 objets.

<prototype> cadre
type : objet
fils : soutien-selle, cadre-inférieur, cadre-avant, barre
poids : infreel(3.)

La création du prototype de "cadre" entraîne l'ajout automatique pour les 4 objets composants, de la référence à "cadre" dans leurs facettes père.

On obtient le schéma structurel suivant :



c) Suppression de barre :

La suppression de l'objet "barre" implique évidemment celle du prototype associé, mais aussi la modification des liens et procédures qui y font référence :

- suppression du lien binaire angle avec l'objet "barre" dans le prototype de "soutien-selle".

- suppression de la référence à l'objet "barre" dans le lien structurel fils du prototype de "cadre".

A la fin de la conception des prototypes participant à la fonction "maintien de l'ensemble", la description des prototypes d'objets crées dans le cadre du Modèle est la suivante :

<prototype> cadre
type : objet
fils : soutien-selle, cadre-inférieur, cadre-avant.
poids : infréel(3.)

<prototype> soutien-selle
type : objet
père : cadre
partie-fonctionnelle : maintien de l'ensemble
poids : intervreel(0.25, 0.30)
longueur : infreel (0.5)
angle : val(65)

<prototype> cadre inférieur
type : objet
père : cadre
partie-fonctionnelle : maintien de l'ensemble

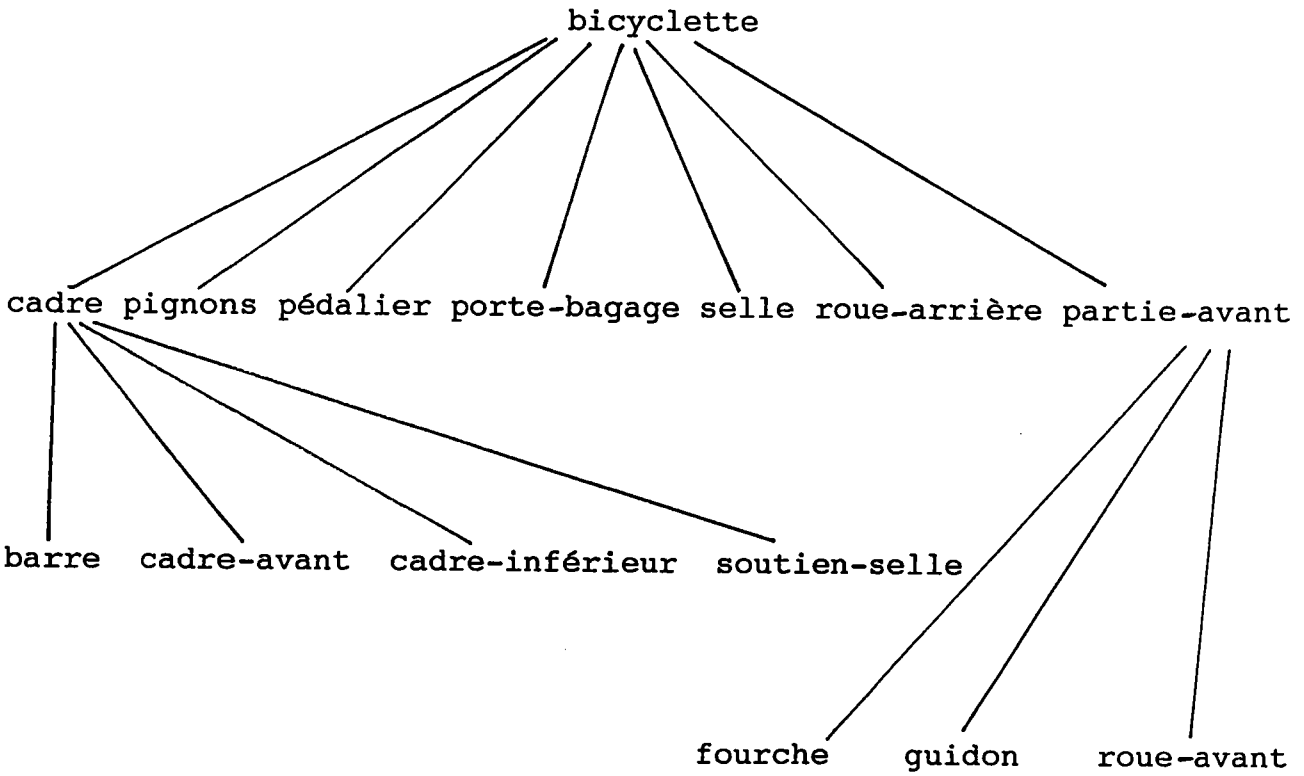
<prototype> cadre avant
type : objet
père : cadre
partie-fonctionnelle : maintien de l'ensemble

Remarque :

Etant donné que le poids de l'objet "cadre" doit être inférieur à 3kgs, celui de chacun de ses fils doit vérifier cette contrainte. Un deuxième niveau d'héritage serait de considérer que le cumul du poids de ses fils soit inférieur à 3kgs.

II.1.2.3) Création des prototypes des autres parties fonctionnelles :

Les processus de prise en compte par le Modèle des informations concernant les autres fonctions à réaliser est semblable : c'est la raison pour laquelle nous ne le développons pas ici. L'ensemble des prototypes créés à la fin de cette phase de conception pourra être représenté par le schéma suivant :



Exemple de la bicyclette : hiérarchie structurelle (à un moment donné de la conception).

Remarques :

Certains objets n'ont pas encore été définis c'est l'exemple de l'objet "chaîne". D'autres déjà créés, sont incomplets ainsi pour "roue avant" on ne possède que certaines propriétés alors quelle est composée d'un pneu, d'une jante, de rayons, etc... qui sont des objets qui n'ont pas encore été conçus.

La hiérarchie structurelle définie sur le schéma ci-dessus est différente du découpage fonctionnel qui a guidé le processus

de conception.

II.1.3) Conclusions de l'exemple :

Ce premier exemple bien qu'étant assez général a permis de mettre en évidence le cadre d'utilisation du Modèle dans un processus de conception aussi bien au niveau des informations gérées que de la manière de concevoir.

II.1.3.1) Informations gérées par le Modèle :

Tout d'abord le Modèle a permis de prendre en compte par le biais des parties-fonctionnelles les éléments du cahier des charges. Ensuite tous les objets réalisant une fonction donnée ainsi que les liens entre ces objets ont été modélisés par l'intermédiaire des prototypes.

II.3.1.2) Méthode de conception :

L'exemple de conception de la bicyclette a mis en évidence un certain nombre de fonctionnalités du Modèle :

- dynamicité de la structure : ajout de nouveaux prototypes et de nouveaux attributs, suppression de prototypes (exemple de "barre"),...

- conception ascendante (création de "cadre") et descendante (création des prototypes de chaque partie-fonctionnelle).

- héritage à partir de l'objet "cadre".

- gestion d'objets incomplets : par exemple à la fin de la phase de conception que nous venons de décrire, la plupart des objets ayant été définis sont incomplets.

II.2) Deuxième exemple : la lampe de bureau

Cette exemple décrit la démarche faite dans un bureau d'études dans le but dela création d'une lampe de bureau : cette démarche de conception est constituée d'un certain nombre de phases décrites dans [GAR-83] : il s'agit d'abord de la définition du cahier des charges, puis de l'étude cinématique et enfin de l'étude statique.

II.2.1) Cahier des charges :

II.2.1.1) Description :

On veut pouvoir concevoir 2 types de lampes de bureau.

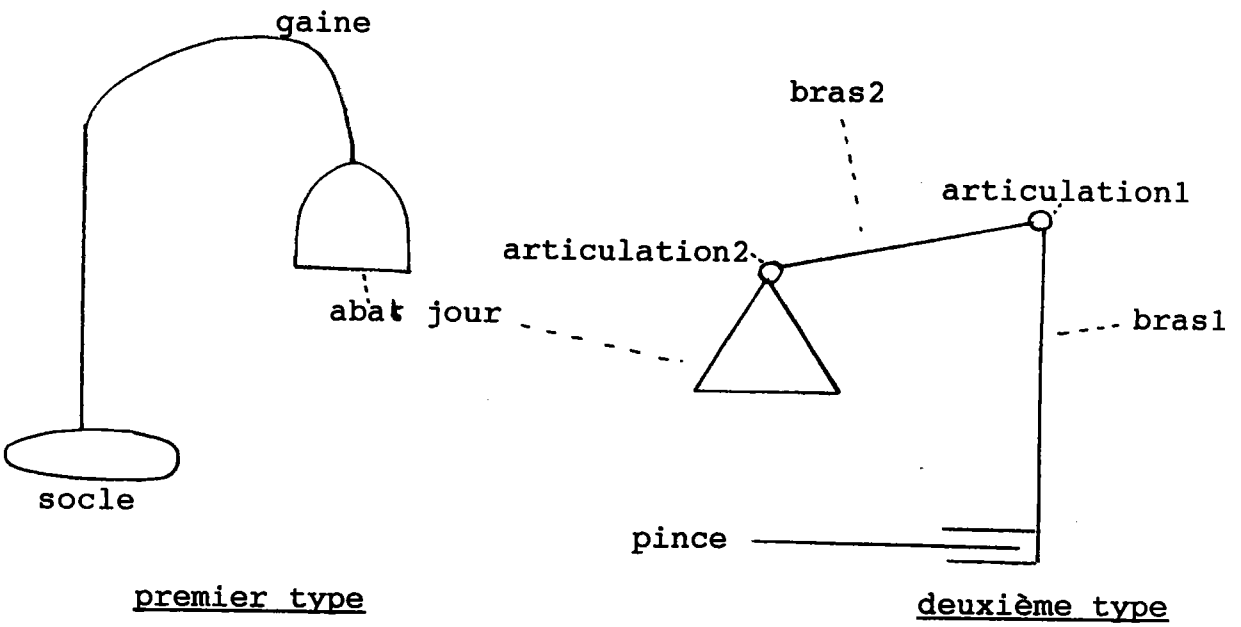


schéma global

Leur fonctions essentielles sont : éclairage et mobilité.

Evidemment des critères comme l'esthétique et le coût sont à prendre en compte car ils sont spécifiques à ce domaine (de même qu'en aéronautique la légèreté ou le coefficient de sécurité sont implicites).

II.2.1.2) Gestion du cahier des charges par le Modèle :

Voici les éléments du cahier des charges pris en compte par le Modèle grâce à la création des prototypes des objets lampe 1 et lampe 2 (qui implique celle de abat-jour, gaine,...).

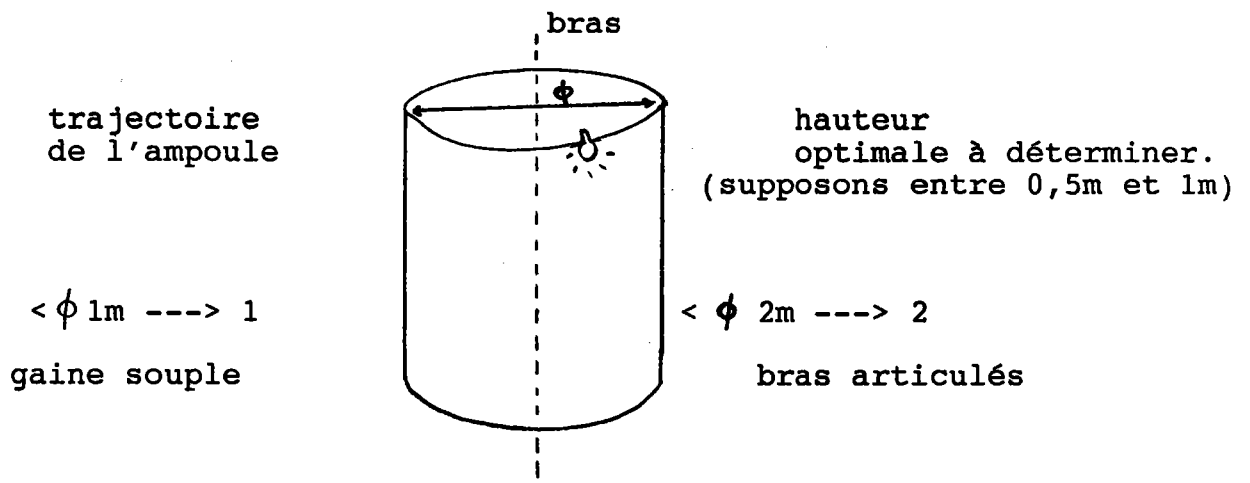
```
<prototype> lampe1
 type : objet
 fils : abat-jour, gaine, socle
 prise : infréel(15.)
```

```
<prototype> lampe2
 type : objet
 fils :  abat-jour,  bras1,  bras2,  pince,  articulation1,
        articulation2
 prise : infréel(120)
```

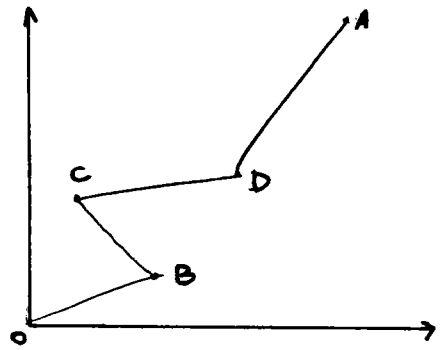
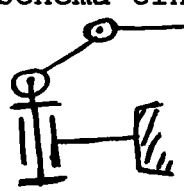
II.2.2) Etude cinématique :

II.2.2.1) Description :

Les propriétés essentielles de ces lampes seront de la mobilité et le champ d'éclairage (voir schéma ci-dessous)



Fonction à étudier : "mise en position de la lampe par rapport au socle"

cas gaine souple	cas bras articulés
calcul de la longueur de la gaine en fonction de la hauteur. $l^2 = h^2 + (\frac{d}{2})^2$ l = longueur minimale h = hauteur de l'abat jour d = diamètre avec l maxi = l + 10 %	 On cherche n (nombre de bras minimal) tel que OA appartienne au cylindre (schéma ci dessus ϕ = 2m, hauteur=h) Mathématiquement on trouve n=2 (on a d'abord montré que pour n=1 --->ne marche pas). De plus la longueur des bras doit être identique ----> $2l = \sqrt{h^2 + (\frac{d}{2})^2}$ D'où le schéma cinématique Liaison pivot   pince liaison avec le sol (ici le bureau)
(critère que l'on s'est fixé en plus)	

II.2.2.2) Utilisation du Modèle :

a) cas de la gaine souple :

L'étude cinématique revient à définir l'intervalle des longueurs acceptables de la gaine. Les calculs correspondant à cette étude se modélisent par la définition suivante du prototype de l'objet gaine :

```
<prototype> gaine
 type : objet
 père : lampe1
 hauteur : intervreel(0.5, 1)
 d : infreel(1.)
 l : intervreel(f1(h,d),mult(f1(h,d),1.1))
```

la création de l'attribut "l" (longueur) du prototype "gaine" nécessite l'existence de la fonction de calcul f1() définie par $f1(x,y)= \sqrt{x^2 + (y/2)^2}$ et de la fonction mult() qui correspond à l'opérateur de multiplication.

f1() est une procédure spécifique au domaine de conception ou même au concepteur alors que mult() est une procédure de base.

L'attribut "l" a la sémantique suivante : la longueur de la gaine est un réel compris entre lmin = $\sqrt{h^2 + (d/2)^2}$ et lmax=lmin * 1.10.

Le calcul des bornes de l'intervalle des valeurs admissibles pour "l" se fait par le biais de deux procédures (les bornes de cet intervalle vont varier selon les valeurs des attributs "h" et "d"). Il s'agit du cas d'une facette non calculée (appartenant à un attribut non calculé) faisant référence à une fonction de calcul.

b) cas des bras articulés :

La recherche du nombre optimal de bras est faite manuellement, tandis que le calcul de la longueur des bras sera pris en compte par le Modèle. Pour celà on adopte une démarche identique à celle suivie pour la définition de l'attribut "l" dans le cas de la gaine souple. L'attribut "l" de chacun des bras articulés sera défini de la façon suivante :

l : f2(h,d)

h et d sont respectivement la hauteur et le diamètre du cylindre. f2() est une procédure spécifique définie par $f2(x,y) = 1/2\sqrt{h^2 + (d/2)^2}$.

II.2.3) Etude statique :

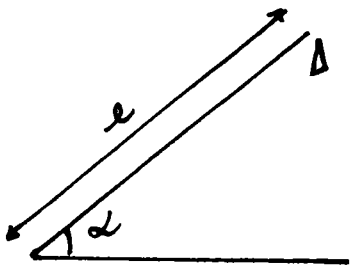
II.2.3.1) Description :

Cette étude doit permettre de dimensionner les pièces et les liaisons et aussi de choisir les matériaux (ces trois notions sont dépendantes les unes des autres).

étude statique <----> étude des efforts.

Les efforts dans les 2 cas sont dûs au poids de l'abat-jour (la lampe ne doit pas se déformer sous son poids).

Le choix de l'abat-jour implique la connaissance de son poids (ce choix se fait en fonction de critères esthétiques, de stock, de coût,...)

cas gaine souple	cas bras articulé
poids abat-jour--> efforts dans la gaine La connaissance des efforts dans la gaine permet son choix dans un catalogue constructeur. (ici encore interviennent les critères de coût, d'esthétique,de disponibilité...). Rque: Dans le catalogue figurent par exemple le poids limite avant déforma- pour une 1 longueur donnée. Il reste ici à déterminer le socle et les liaisons binaires Recherche du socle ---------- choix du socle dans le catalogue en fonction (en plus des critères classiques) de la position limite que l'on veut atteindre (on parlera aussi d'angle limite).  	On va essayer de définir les efforts maximaux (cas de l'ensemble à l'horizontale) Ces calculs permettent de connaître la valeur des torseurs aux différentes liaisons et en chaque point des bras. (torseur <---> liaison binaire) On va se retrouver dans 2 cas de figures : - série limitée - très grande série a) série limitée : On consulte le catalogue . En fonction du profilé du matériau et du coût on choisira les bras. ex. de profilés:  De plus le cable electrique doit pouvoir passer à l'intérieur des bras. Cette étude aboutit au choix du bras.

Remarque : Pour la gaine on avait consi déré le cas le plus défavo-rable, mais il se peut que cette position ne soit plus souhaitée. Dans ce cas on peut retourner au cahier des charges et choisir une nou-velle gaine. Détermination des liaisons --- binaires. --- (Socle-gaine et gaine lampe) Etant donné que le socle, l'abat-jour et la gaine sont normalisés et en fonction des efforts le choix sera facile.	b) Très grande série (On fabriquera le profilé) On ne se trouve pas limité ici aux propositions de profilés du catalogue. Recherche de la pince. --- - Dans le cas de la série limitée choix de la pince dans le catalogue. - dans le cas de la grande série on fabriquera une pince adaptée au profilé (on a déjà calculé les efforts) le prin-cipe est le même que pour la fabrication du profilé. Recherche des articulations --- Même démarche que précé-demment.

II.2.3.2) Utilisation du Modèle :

a) cas de la gaine souple :

* choix de l'abat-jour et de la gaine :

Nous allons choisir un "abat-jour", puis une "gaine" compatible dans les catalogues. Celà revient à considérer la contrainte-intra "poids" des objets "abat-jour" dans la recherche des objets "gaine". On déduit de cette règle les prototypes suivants de ces objets :

```
<prototype> abat-jour
 type : objet
 père : lampe1
 poids : Réel

 <prototype> gaine
  type : objet
  père : lampe1
  h : intervreel(0.5,1)
  d : infréel(1.)
  l : intervreel (fl(h,d),1.1)
  poids : supréel((abat-jour,poids))
```

"poids limite : supréel((abat-jour,poids))" contraint tout objet "gaine" à supporter sans déformation "l'abat-jour" associé.

Le choix d'un abat-jour dans un catalogue va impliquer la connaissance de son poids. La recherche de la gaine appropriée se fait aussi dans un catalogue : la longueur et le poids limite (qui est le poids de l'abat-jour) sont les paramètres qui vont permettre de déterminer la ou les gaines recherchées (d'autres critères comme le prix peuvent aussi être pris en compte).

Soit l'exemple suivant d'un objet abat-jour pour lequel on ne décrira que la propriété poids.

<instance 1> abat-jour
 poids : 0.25

L'objet gaine suivant correspondant à un élément du catalogue est compatible avec abat-jour :

<instance 1> gaine
 h : 0.6
 d : 0.8
 l : 0.72
 poids limite : 0.30

En effet le poids limite correspondant à l'objet gaine est supérieur au poids de l'abat-jour.

L'instanciation des attributs poids et poids-limite a été faite explicitement par le concepteur suite à la lecture des catalogues.

Dans le cas où l'on voudrait modifier la position limite de la gaine, celà reviendrait à retourner au cahier des charges et à déterminer une nouvelle fonction de calcul de la longueur de la gaine.

* Choix du socle et des liaisons :

Cette étape se fera exactement de la même façon que la précédente c'est-à-dire par une recherche dans des catalogues d'éléments compatibles avec la gaine et l'abat-jour (par exemple tout fabricant de gaines va proposer pour chaque gaine, un ensemble d'extrémités compatibles).

b) cas des bras articulés :

A chaque torseur au niveau d'une liaison correspond une contrainte-intra dans les prototypes des deux objets concernés par la liaison (nous reviendrons sur les torseurs plus en détail dans le prochain exemple).

- S'il s'agit d'une petite série, la conception d'une lampe se fera par l'intermédiaire de recherches dans des catalogues. La démarche suivie est identique à celle étudiée précédemment.

- Dans le cas d'une grande série, il va falloir dessiner les éléments de la lampe, les dimensionner, choisir aussi les matériaux utilisés pour chaque pièce : à chacune de ces étapes les objets crées doivent vérifier les contraintes définies dans leurs prototypes (au sens modèle de données). Par exemple le dessin de la lampe découle d'un habillage de son schéma cinématique, le dimensionnement est dépendant des contraintes du cahier des charges et le choix du matériau d'un objet dépend des efforts (torseurs) appliqués sur cet objet.

II.2.4) Conclusions de l'exemple :

Dans l'exemple de conception que nous venons de décrire, l'importance des recherches d'objets dans les catalogues constructeurs est primordiale : c'est un cas de figure très fréquent en conception mécanique. Une part importante du travail de conception consiste à rechercher dans un catalogue, un ou plusieurs objets répondant à un certain nombre de contraintes. En utilisant le Modèle la confrontation d'un objet du catalogue avec le prototype auquel il doit être associé se fait manuellement : il faut donc saisir les caractéristiques de cet objet. On se rend compte dans cet exemple de l'importance d'une communication entre le Modèle et une Base de catalogue : cette communication serait d'autant plus facile que les objets des catalogues ont la même structure que ceux gérés par le Modèle. Il serait alors possible non seulement de faire des recherches automatiques dans un catalogue mais aussi de mettre ces catalogues à jour par exemple en y rajoutant des objets de conception.

En dehors des consultations de catalogues, cet exemple a permis de mettre en évidence des fonctionnalités du Modèle qui n'étaient pas apparues dans le premier exemple.

Ces fonctionnalités sont les suivantes :

- conception imbriquée de prototypes jusque là inexistants pendant la phase de création d'un prototype (exemples de lampe1 et lampe2).

- déduction par le biais des ensembles de valeurs admissibles (attribut longueur de l'objet gaine). La déduction est d'autant plus intéressante du fait du caractère essai-erreur et les retours en arrière des processus de conception.

II.3) Troisième exemple : la pompe doseuse
(Exemple tiré de [LTC-87])

Les exemples développés jusque là étaient relativement simples, aussi bien en ce qui concerne le modèle du projet à concevoir que les détails de la conception des sous-objets. Par contre ils ont permis de mettre en évidence l'adéquation du Modèle et de la structure de données aux informations manipulées dans le cadre d'une conception mécanique.

Ce troisième exemple est déjà plus complexe pour un non mécanicien et pour cette raison on fera d'abord une description générale du problème et on reviendra plus en détails sur une partie précise de la phase de conception (il s'agit ici du calcul de l'engrènement et de l'arbre de vis). Enfin on étudiera la compatibilité du Modèle avec les données manipulées dans cette étape.

II.3.1) Présentation générale de l'exemple :

II.3.1.1) Description :

Une pompe doseuse est une pompe à mouvement alternatif, dont le déplacement du piston est commandé de façon positive (la transmission se fait par obstacles, sans glissement). Elle est étudiée pour mesurer des liquides avec précision, sous des pressions différentielles positives (la pression de refoulement doit être plus grande que la pression d'aspiration). La pompe exécute cette fonction de dosage avec une fidélité de +-1%.

Dans le principe, la pompe est constituée par un ensemble mécanique, un piston plongeur et un doseur dans lequel le piston exerce son mouvement. La pompe délivre un volume de liquide parfaitement défini à chaque course de refoulement du piston plongeur.

Le débit de la pompe est réglabe en changeant, soit la course du piston, soit la cadence de pompage.

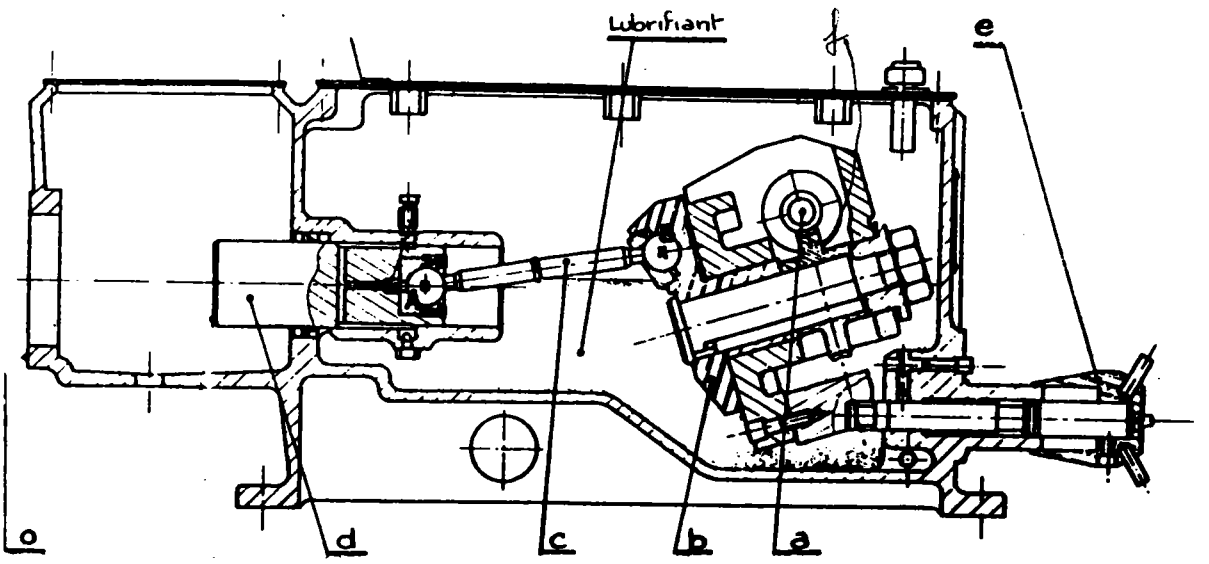


Figure 1

Fonctionnement :

Le système d'entrainement de la mécanique est basé sur le principe à bielle tournante (fig.1). Le moteur de la pompe transmet son mouvement à l'arbre d'entrainement à vis sans fin (a), la manivelle (b) est alors entrainée en rotation.
La bielle (c), solidaire de la manivelle (b), communique son mouvement au coulisseau (d) et par conséquent au piston.
Le réglage de course est assuré par une vis micrométrique (e).

Course 0 :

La roue dentée (f) se trouve dans un plan vertical, la manivelle tourne dans le même plan. Aucun mouvement n'est communiqué au coulisseau ainsi qu'au piston, et par conséquent, pas de débit.

Course 100% :

La manivelle est inclinée par rapport à la verticale, la bielle tourne dans le même plan et le piston est alors alternativement poussé en avant puis en arrière.
Il en résulte donc une course maximum du piston ainsi qu'un débit maximum.

II.3.1.2) Cahier des charges :

a) Description :

Il s'agit ici de concevoir une pompe doseuse dont la représentation graphique finale se trouve sur la figure 1 (pour une meilleure compréhension du problème à résoudre) et dont les principes de fonctionnement ont été décrits ci-dessus.

Les seules informations dont nous disposons pour démarrer cette étude sont les caractéristiques du moteur.

Vitesse de rotation du moteur : 1400 tr/mn (vitesse de l'arbre entrée)
Puissance : 7,5 kW
Rapport de réduction de vitesse entre vis et roue : 11

b) utilisation du Modèle :

La prise en compte par le Modèle du cahier des charges revient à créer des prototypes et instances suivants :

```
<prototype> moteur
 type : objet
 vitesse : Réel
 puissance : Réel

<instance> moteur
  vitesse : 1400
  puissance : 7.5
```

```
<prototype> pompe-doseuse
 type : objet
 fils : vis, roue

<prototype> vis
 type : objet
 père : pompe-doseuse
 vitesse : val((moteur, vitesse))

<prototype> roue
 type : objet
 père : pompe-doseuse
 rapport-reduc : val (0.0909)
 vitesse : mult(rapport-reduc,(vis,vitesse))
```

Remarques :

- les caractéristiques du moteur sont connues à l'avance, celà veut dire que l'on peut dès le début de la conception créer l'objet moteur qui est un objet complet : l'instance ou les instances du moteur seront donc connues dès le départ.
Il n'en est pas de même des autres objets pour lesquels on ne connaît que certains de leurs attributs.

- la procédure val() correspond à l'opérateur d'égalité.

- les attributs vitesse des trois objets sont des attributs contraintes-intra. Ils expriment la règle suivante : la vitesse de la vis est égale à celle du moteur et la vitesse de la roue est égale à celle de la vis multipliée par le rapport de réduction.

II.3.1.3) <u>étude cinématique</u> :

a) <u>Description</u> :

La première étape sera l'étude cinématique (étude des liaisons) qui permettra la transformation de mouvements attendue : rotation ---> rotation ---> translation.
Cette étude aboutit au schéma cinématique ci-dessous.

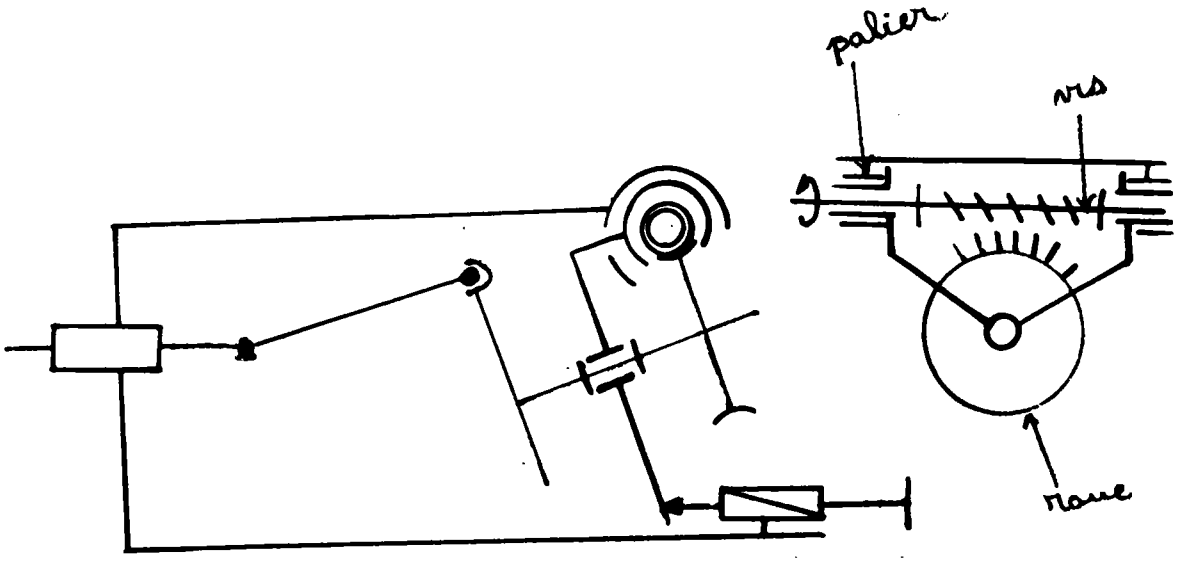


schéma cinématique : (1 = perspective, 2 = planes)

Remarque : Le schéma cinématique, est aussi un schéma "fonctionnel", c'est-à-dire qu'il met en évidence le fonctionnement et non le choix des matériaux.

b) <u>Le schéma cinématique</u> :

Le schéma cinématique explique le fonctionnement du système recherché. Il permet de déterminer les pièces (ou les sous-ensembles de pièces fixes les unes par rapport aux autres : par exemple la bielle peut être composée de plusieurs pièces), ainsi que les liens entre ces pièces. Chacun de ces liens appartient à un type de lien auquel correspondra un symbole graphique (cf Annexe2 : types de liaisons mécaniques). Pour un objet donné, chacune de ses liaisons implique un torseur cinématique et un torseur statique qui sont tous deux représentés par une matrice 3*2 : à chaque torseur cinématique correspond un torseur statique unique. Il suffit de connaître le torseur cinématique d'un objet en un point pour déduire automatiquement tous les autres torseurs cinématiques.

Pour les objets qui nous intéressent ici (vis, roue et palier) l'étude cinématique permet de leur associer leurs torseurs respectifs.

Pour la vis on aura pour chacun de ses liens avec les autres objets (roue, moteur, palier) un torseur cinématique et un torseur statique. Pour ses torseurs cinématiques on connaitra leurs valeurs car celles-ci dépendent de la vitesse de la vis qui a été fixée lors du cahier des charges. De plus on sait que la vis aura d'autres propriétés comme son diamètre et le nombre de filets.

c) Utilisation du Modèle :

On déduit de l'étude cinématique les prototypes suivants des objets vis et roue.

* objet vis :

```
<prototype> vis
 type : objet
 père : pompe-doseuse
 vitesse : val((moteur,vitesse))
 diamètre : Reel
 nb-filets : intervent(1,5)
 cin vis-roue :tc5(vitesse)
```

Les attributs vitesse, diamètre, nb-filets et cin vis-roue sont tous des attributs contraintes-intra. Pour une vis le nombre de filets est toujours compris entre un et cinq ce qui est exprimé par la procédure intervent(1,5) (intervalle entier de un à cinq).

Le torseur cinématique de la vis par rapport à la roue est une procédure de calcul tc5() qui est directement associée au type de lien (entre la roue et la vis) du schéma cinématique. On fixe par convention qu'au type de lien n°i on associe tci(). Dans un souci de clarté nous n'avons pas exprimé dans le prototype de vis ses deux autres torseurs cinématiques ainsi que ses trois torseurs statiques : ces cinq torseurs sont représentés par des attributs contraintes-intra calculés dont les occurences sont directement déduites de "cin vis-roue".

* Objet roue :

```
<prototype> roue
 type : objet
 père : pompe-doseuse
 rapport-réduc : val(0.0909)
 vitesse :=mult(rapport-reduc,(vis,vitesse))
 diamètre : Réel
 nb-dents :=mult(rapport-reduc,(vis,nb-filets))
 angle : intervreel(0.90)
 cin roue-vis :=transftc(rapport-reduc,(vis,cin vis-roue))
```

Les informations supplémentaires intégrées dans le prototype de roue lors de l'étude cinématique sont les suivantes : son diamètre est un réel, le nombre de dents est déduit du nombre de filets, son angle varie entre 0° et 90° et son torseur cinématique par rapport à l'objet vis est déduit de celui de l'objet vis et du rapport de réduction.

transftc() est une procédure qui transforme un torseur cinématique (matrice 3*2) dans le torseur cinématique inverse (matrice 3*2) en utilisant le rapport de réduction. On peut représenter cette transformation par l'exemple ci-dessous :

$$\begin{bmatrix} va & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \xrightarrow{\text{transftc()}} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ va' & 0 \end{bmatrix}$$

avec va/va'= rapport de réduction des vitesses de rotation.

Remarques :

- cette matrice est représentée dans le Modèle par une liste de 6 éléments.

- quand on connaît la valeur d'un torseur cinématique d'un objet en un point donné, les autres torseurs cinématiques de cet objet en seront déduits directement : ainsi dans le Modèle ces autres torseurs cinématiques sont représentés par une contrainte-intra calculée (le paramètre de la fonction de calcul associée est le premier torseur cinématique). Il en sera de même pour les torseurs statiques de cet objet.

- à la fin de l'étude cinématique on connaît les valeurs (instances dans le Modèle) des torseurs cinématiques (c'est-à-dire les instances des contraintes-intra associées). Il n'en est pas de même des torseurs statiques pour lesquels on ne connaît que l'attribut contrainte-intra associé : cet attribut sera instancié lors de l'étude statique.

* Autres objets :

L'étude cinématique permet de créer les autres objets (ou groupes d'objets) qui ont été définis sur le schéma cinématique : manivelle, bielle, coulisseau, etc... Les prototypes de ces objets auront comme seules propriétés leur appartenance à la pompe-doseuse et leurs torseurs cinématiques.

II.3.1.4) Etude statique :

Son but est le dimensionnement des pièces et des liaisons constituant le mécanisme, ainsi que le choix des matériaux.
Cette étude comporte différentes étapes :

1. calcul de l'engrènement (du module de la roue et de la vis)
2. calcul de l'arbre vis et celui des paliers (c'est-à-dire des roulements).
3. calcul des dimensionnements de toutes les autres pièces. Ainsi le dimensionnement du palier (vu précédemment) impliquera celui des pièces directement dépendantes non encore calculées.

On pourra remarquer que les calculs faits dans l'étude statique peuvent impliquer un retour en arrière pour des raisons comme l'encombrement, de calculs d'inerties (1), etc...
Ce cas de figure se rencontre souvent dans les problèmes de conception en générale et particulièrement en CAO mécanique (méthode itérative).

II.3.1.5) Etapes suivantes :

L'étude statique permet de définir le dessin final avec les caractéristiques de ses différents éléments (dimensions, matière, etc...).
Les étapes suivantes seront le passage au bureau des méthodes qui établira un plan de fabrication, puis la fabrication.

II.3.2) Calcul de l'engrenement et de l'arbre vis :

II.3.2.1) Calcul de l'engrenement :

a) Présentation :

Ce calcul se fait à partir de 2 objets : la roue et la vis. Les propriétés caractéristiques utilisées pour ce calcul sont :

- pour le moteur sa puissance qui est de 7,5 kW, sa vitesse de rotation qui vaut 1400 tr/mn (qui sera donc aussi celle de la vis).
Les caractéristiques citées ci-dessus sont en fait des standard-moteur.

- le rapport de réduction entre la vis et la roue dont la valeur est de 11.

A partir de ces données on définit l'engrenement (c'est-à-dire les efforts entre roue et vis) ainsi que les diamètres de la roue et de la vis.

(1) le dimensionnement des pièces est suivi des caculs d'inerties : ces calculs peuvent mettre en valeur des forces que l'on ne pouvait prévoir auparavant. L'existence de ces forces peut impliquer un nouveau dimensionnement ou même un retour à l'étude cinématique.

Pour réaliser ce passage on aurait besoin d'outils de type système-expert : en fonction de la nature des matériaux choisis et le rendement du système mécanique (par exemple selon les différentes valeurs de l'angle), l'utilisateur aura à choisir entre différentes solutions. Il fera ce choix par rapport à des critères de fonctionnement (souplesse, nombre d'heures de fonctionnement par jour, athmosphère corrosive ou non).

Les caractéristiques géométriques de la roue et de la vis, seront les résultats de cette étape.

vis : - diamètre = 37 mm
- nombre de filets = 4

roue : - diamètre = 139,5 mm
- nombre de dents = 44
- angle = 20 degrés

La phase suivante sera de déduire à partir de ces coordonnées, ainsi que du couple moteur, l'action de la roue sur la vis. Ce calcul est totalement automatisable (utilisation d'un algorithme simple connu des mécaniciens).

Son résultat est le suivant :

Composantes de F : (action de la roue sur la vis)

T = 2765 N (tangentielle)
W = 8067 N (axiale)
V = 3104 N (radiale)

b) Utilisation d'un système expert :

L'instanciation pour les prototypes roue et vis de leurs attributs diamètre, nombre de filets (et de dents) et angle se fait manuellement. Pour calculer les caractéristiques de ces objets l'utilisateur aura recours à un système expert extérieur au Modèle actuel (mais qui pourrait être intégré dans le modèle générique). Les paramètres en entrée de ce système expert sont des occurences d'attributs des objets roue et vis, et des informations plus générales dont certaines peuvent être considérées comme des attributs de l'objet pompe-doseuse (nombre d'heures de fonctionnement par jour). La souplesse et la corrosivité de l'atmosphère sont des facteurs de service dépendant d'autres paramètres et difficilement mesurables mais ils sont pris en compte par le système expert. Ces trois paramètres (durée de vie, souplesse, corrosivité) peuvent être considérés comme des éléments supplémentaires du cahier des charges.

Le système expert permet de déduire les caractéristiques géométriques de la roue et de la vis (se reporter annexe 2 : liste de règles du système expert).
L'instanciation des attributs correspondant se fait manuellement.

Le résultat de cet étape implique la création des instances suivantes :

<instance 1> vis
 vitesse : 1400
 diamètre : 37
 nb-filets : 4
 cin vis-roue : 1400,0,0,0,0,0

<instance 2> roue
 rapport-reduc : 0.0909
 vitesse : 127.26
 diamètre : 139.5
 nb-dents : 44
 angle : 20
 cin roue-vis : 0,0,139.5,0,0,0

II.3.2.2) Détermination de l'action de la roue sur la vis :

Rappelons d'abord qu'à chaque type de liaison est associée un torseur cinématique et un torseur statique (cf. Annexe 2 : différents types de liaisons).

Donc dès l'étude cinématique on connait la forme générale des torseurs statiques mais leur calcul (déduit de celui du dimensionnement) s'effectue dans l'étude statique.

Nous allons maintenant déterminer les torseurs statiques entre la roue et la vis : ils correspondent à des matrices 3*2 obtenues grâce à une fonction de calcul dont les paramètres sont les suivants : les caractéristiques géométriques calculées précédemment ainsi que la donnée de la puissance du moteur et de la vitesse de rotation de la roue. Il existe un lien entre torseur statique roue-vis et torseur statique vis-roue : les matrices 3*2 associées sont opposées.

La prise en compte par le Modèle des actions entre roue et vis revient à la création d'attributs contraintes-intra calculés pour ces deux objets.

Ces attributs sont définis de la façon suivante :

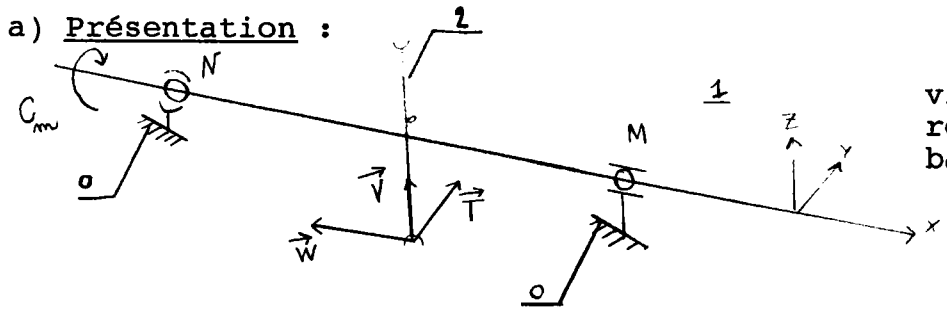
```
                    <prototype> vis
                     type : objet
                     père : pompe-doseuse
                     vitesse : val ((moteur,vitessse))
                     diamètre : Reel
                     nb-filets : intervent (1,5)
                     cin vis-roue : =tc5(vitesse)
création
attribut --------->  stat  vis-roue : =ts5(diamètre,nb-filets, (roue,
                                       diamètre), (roue,nb-dents),
                                       (roue,angle), (roue,vitesse)
                                       ,(moteur,puissance))


                    <prototype> roue
                     type : objet
                     père : pompe-doseuse
                     rapport-reduc : val(0.0909)
                     vitesse : =mult(rapport-reduc,(vis,vitesse))
                     diamètre : Reel
                     nb-dents : =mult(rapport-reduc,(vis,nb-filets)
                     cin roue-vis  : =transftc (rapport-reduc,(vis,cin
                                      vis-roue))
création
attribut ---------> stat roue-vis :=transfts ((vis,stat vis-roue))
```

ts5() est une fonction de calcul associée au type de liaison. Cette fonction fournit comme résultat une liste de 6 réels correspondant au torseur statique.

transfts() transforme une liste de 6 réels en une liste de leurs opposés.

II.3.2.3) Calcul de l'arbre vis (et de ses paliers) :

a) Présentation :



vis : 1
roue : 2
balancier : 0

Pour mettre en évidence les actions qui s'exercent sur la pièce, on isole celle-ci et on fait le bilan des actions extérieures :

- le poids (à négliger ici par rapport aux actions en jeu).
- les torseurs (matrices 3 * 2) dont les paramètres ont été calculés aux étapes précédentes. Il s'agit de :
    * torseurs dûs à l'action du moteur.
    * torseur de la roue sur la vis.
- les torseurs représentant les actions aux paliers (que l'on cherche ici).

Dans un premier temps on va définir les valeurs des torseurs aux paliers: ce calcul est automatique et revient à la résolution d'un système de n équations à n inconnues.

Ensuite la connaissance des efforts sur les paliers (axiaux et radiaux) permet le calcul des paliers (c'est-à-dire leur dimensionnement par rapport à un catalogue). Le calcul des paliers dépend aussi d'autres paramètres comme la durée de vie, le type de matériau. Tous ces paramètres permettent de trouver dans le catalogue constructeur un certain nombre de solutions. On déduit ensuite l'arbre vis de la même façon que les paliers avec comme condition supplémentaire que l'arbre vis doit être compatible avec les paliers.

b) Utilisation du Modèle :

Le calcul des torseurs (aux paliers) s'effectue de la même façon que précédemment : ce calcul revient à instancier des attributs contraintes-intra calculés dans les prototypes des objets concernés par les efforts sur les paliers. Les facettes calculées associées correspondent à des procédures de calculs utilisées couramment par les mécaniciens. Cette étape de calcul des torseurs statiques est automatique.

L'étape suivante est la recherche dans un catalogue, d'objets compatibles (paliers puis arbre vis) avec les efforts à supporter (torseurs statiques) et les dimensions des autres pièces.

Cette étape revient à confronter un prototype avec des pièces d'un catalogue : c'est un cas de figure que nous avons déja rencontré dans l'exemple 2.

II.3.4) Conclusions concernant cet exemple :

L'exemple de conception qui vient d'être décrit est moins général et plus complet (liens avec la fabrication) que les précédents : la description détaillée de certaines parties du processus de conception de la pompe-doseuse a permis de déterminer avec précision le cadre d'utilisation du Modèle.

Un des intérêts de cet exemple concerne la gestion de l'étude cinématique dont le but est la création du schéma cinématique. Un schéma cinématique est composé d'un ensemble d'objets reliés entre eux par des liaisons; chaque objet aura un nom et chaque liaison sera représentée par un symbole graphique normalisé correspondant au type de la liaison (il existe 11 types de liaisons donc 11 symboles : se référer à l'annexe 2). A chaque type de liaison est associé un torseur cinématique unique. A partir d'un schéma cinématique on peut donc déduire un ensemble de prototypes : leurs noms sont ceux des objets du schéma et chaque liaison pour un objet donné impliquera la création d'un attribut contrainte-intra (représentant le torseur cinématique associé) dans le prototype de cet objet.

La "traduction" du schéma cinématique dans le modèle de données dans le cadre de notre étude s'est fait manuellement, mais elle est facilement automatisable. Il faudrait pour celà rajouter au Modèle un module d'interprétation de schémas cinématiques normalisés permettant de générer automatiquement des prototypes et des attributs de prototypes à partir d'un schéma cinématique. Ce module prendrait en compte les associations suivantes :

schéma cinématique	modèle de données
objet ou groupe d'objets liaison	prototype attribut contrainte-intra

Un autre intérêt de cet exemple a été de mettre en évidence les besoins en systèmes experts : la communication entre le Modèle et ces systèmes experts est manuelle. Il en est de même pour la communication entre Modèle et les catalogues qui une fois de plus sont nécessaires dans une conception.

III. Conclusions de l'utilisation du Modèle dans des exemples de conception :

III.1) La démarche de conception :

Les trois exemples que nous venons d'étudier nous ont permis de mieux appréhender le fonctionnement d'un processus de conception mécanique : étapes successives, liens entre ces étapes, schémas, retours en arrière, etc...(se reporter au schéma qui suit). En théorie dans une conception il faudrait partir de rien, mais en général toute conception est biaisée car on aura dès le départ une idée de la solution. Soit l'énoncé général suivant : "Il faut concevoir un moyen de locomotion permettant de passer d'un point x à un point y dans des conditions particulières (terrain, climat, vitesse, poids, autonomie)". Dans l'idéal le système de conception qui permettrait de résoudre ce problème serait construit autour d'un système expert capable de spécifier un moyen de locomotion d'un type différent de tous ceux existant (ni voiture, ni bateau, ni avion, etc...). Mais en conception mécanique on est loin de cet idéal et comme nous l'avons vu sur les exemples, la part des consultations de catalogues est très importante par rapport à la conception elle même.

En plus des catalogues ces trois exemples ont mis en évidence pour une conception mécanique de besoins importants en systèmes experts et en programmes de calculs spécifiques.

III.2) Adéquation du Modèle par rapport à un processus de CAO :

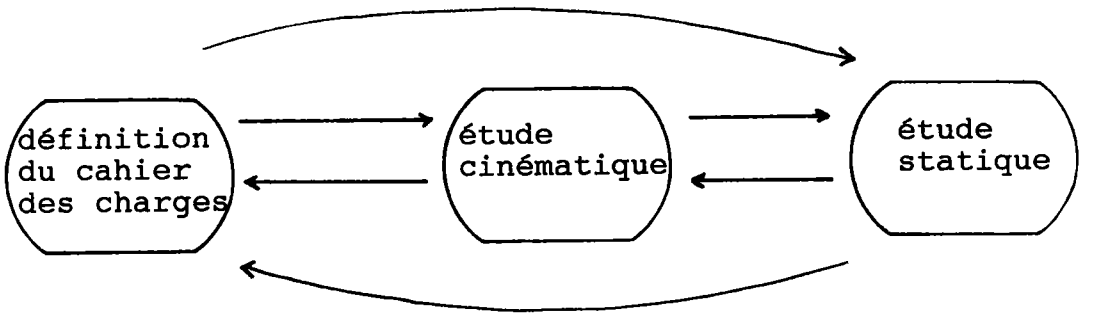
III.2.1) Cadre d'utilisation du Modèle :

Ce chapitre a permis de définir avec une certaine précision le cadre d'utilisation du Modèle, et en particulier ses limites, dans un processus de conception mécanique. Par rapport à une conception "à la main" où certains calculs seraient éventuellement automatisés sous formes de programmes de calculs "isolés", le Modèle constitue une certaine aide à la conception. Il intègre des informations non seulement factuelles, mais aussi des procédures. La notion de prototype permet de mémoriser une quantité importante d'informations et aussi de contraintes correspondant à un instant donné de la conception : tous les objets créés par la suite devront vérifier les contraintes fixées à cet instant. Par exemple les objets définis lors du dimensionnement devront être compatibles avec les prototypes auquels ils sont associés.

Au cours d'un processus de conception mécanique les objets voient leurs structures devenir de plus en plus complexes et la quantité d'information croît très rapidement. Le Modèle offre la possibilité de gérer des structures complexes et aussi une grande partie de l'information manipulée dans le cadre d'un processus de conception. De plus il permet d'automatiser certains calculs par l'intermédiaire de procédures spécifiques. Pour ces raisons il peut être considéré comme un outil d'aide à la conception complémentaire à un logiciel de DAO qui ne gérerait les propriétés géométriques des objets.

II.2.2) Le Modèle et les étapes de conception :

Une démarche de conception peut-être représentée par le schéma qui suit :



Par rapport à ces différentes étapes, les types d'actions (création, modification, suppression) faites sur les objets du modèle du projet peuvent être résumés sur le tableau suivant :

étapes éléments du m.d.d.(1)	cahier des charges	étude cinématique	étude statique
prototypes	*	*	
a. structurel		*	
a. fonctionnel	*		
a. contr-intra	*	*	
a. lien binaire	*	*	
instances			*

Ce tableau met en évidence la tendance suivante : le cahier des charges et l'étude cinématique correspondent aux manipulations des prototypes, tandis que l'étude statique est associée aux manipulations d'instances.

(1) m.d.d. = modèle de données.

Il faut toutefois remarquer que ce tableau ne fait qu'une classification générale des différents types d'actions effectuées par le Modèle selon l'étape de conception : pour cette raison, il ne constitue qu'une indication sur son utilisation dans une démarche de conception. La prise en compte des éléments de conception dans une étape ou dans une autre peut dépendre des "habitudes" de conception : ainsi dans l'exemple 3 la création de torseurs statiques (attributs contraintes-intra) pouvait se faire aussi bien dans l'étude cinématique que statique.

III.3) Extension du Modèle : le Modèle et SACADO

Nous avons vu grâce à ces exemples que le Modèle permettait de gérer la plupart des informations factuelles concernant les objets de conception : à chaque objet du Modèle correspond sa représentation géométrique dans le modèle géométrique. Certaines recherches de solutions nécessitent l'utilisation de systèmes experts et de catalogues contructeurs qui sont des informations gérées par SACADO. Afin d'augmenter ses fonctionnalités il faudrait ajouter au Modèle les composantes suivantes :

- des outils permettant d'accéder et d'utiliser les systèmes experts et les catalogues.

- des outils de gestion de versions plus complets permettant en particulier d'intégrer un objet conçu dans un catalogue.

- des outils plus spécialisés dépendant du domaine de conception comme un générateur et un interpréteur de schéma cinématique.

Enfin, pour aboutir à un système d'aide à la conception aussi convivial que possible, il est indispensable d'associer au Modèle une interface aussi évoluée que possible : la définition de cette interface est l'objet de la Partie 4.

PARTIE 4 : INTERFACE

Dans la deuxième partie nous avons fait une description du Modèle (modèle de données et fonctionnalités) puis de la structure de données logique associée à ce Modèle. Une maquette présentée en Annexe 1 a été réalisée suite à cette première étape. Dans cette maquette, le dialogue avec l'utilisateur est assez proche de la représentation des données manipulées. La troisième partie a permis de fixer le champ d'utilisation du Modèle dans un processus de conception mécanique. En particulier nous avons pu déterminer les éléments de conception pouvant être gérés par le Modèle ainsi que ses conditions d'utilisation (étape, type de calcul, etc...).

Pour permettre une utilisation aisée du Modèle en conception mécanique, il est indispensable de lui associer une interface aussi conviviale que possible c'est-à-dire utilisant le même vocabulaire et aussi naturelle que possible (c'est-à-dire proche de la méthodologie de conception).

Dans un premier temps nous allons faire au paragraphe 1, un tour d'horizon des interfaces de haut-niveau actuelles utilisées dans le cadre des langages de 4ème génération.

Puis dans le deuxième paragraphe nous proposons deux types d'interfaces pour le Modèle.

Enfin dans le troisième paragraphe nous étudions les liens existant entre ces interfaces et les modèles de dialogue du projet SACADO.

I) Interfaces de haut niveau actuelles :
[CHR-85],[GAL-83b],[MAR-86],[MBD-86],[MIR-86]

Dans de nombreux domaines de l'informatique où l'interactivité tient une part essentielle (bureautique, Bases de Données, EAO, etc...) on peut remarquer actuellement de gros efforts dans la recherche d'interfaces de haut niveau devant ouvrir ces domaines à des non-informaticiens. Ces interfaces s'intègrent dans le cadre des langages de 4ème génération : l'apparition de ces langages a été rendue possible par l'augmentation de la capacité de stockage et de traitement des machines et par la diminution des coûts des terminaux graphiques (qui permettent une présentation agréable des informations). Nous allons décrire certaines de ces interfaces dans les domaines suivants :

-Bases de Données
-Bureautique

I.1) Interfaces pour Bases de Données :

I.1.1) Le problème :

L'accès à une Base de Donnée demande la maîtrise d'un langage formel LMD et/ou LDD (1). Il en existe une grande variété correspondant chacun à un type de langage (ensembliste, prédicatif, orienté t-uples, orienté domaine, etc...). La connaissance d'un de ces langages nécessite un effort de formation assez importants bien qu'il s'agisse là de langages non procéduraux.

Cette complexité et cette diversité limitent l'utilisation des Bases de Données par des utilisateurs non-informaticiens. L'utilisateur se place dans l'univers réel et la Base de Données en est une représentation : l'interface devra donc atténuer les différences de "langages" et de référentiels sémantiques. Cette interface doit être conviviale et intelligente permettant de définir et de manipuler une Base de Données en masquant le schéma relationnel.

L'interrogation des Bases de Données a été l'une des premières situations d'interaction homme-machine étudiée par les ergonomes . La recherche d'interfaces pour un non - informaticiens a d'abord débouché sur des systèmes à menus qui se sont révélés fastidieux et contraignants du point de vue de l'expression des requêtes. La recherche actuelle s'oriente vers des interfaces intelligentes et conviviales utilisant les moyens de l'intelligence artificielle (déduction, langage naturel, reconnaissance et synthèse de la parole, etc...) et des techniques graphiques. Les produits

(1) LMD: Langage de Manipulation de Données, LDD: Langage de Description de Données.

les plus complets issus de ces recherches sont actuellement tous des interfaces purement graphique: il n'existe pas encore à notre connaissance d'interfaces intelligentes remplissant la totalité des moyens de l'Intelligence Artificielle cependant des produits et des recherches actuelles en abordent différents sous-ensembles (il s'agit essentiellement de l'interrogation en langage naturel).

Nous allons en décrire quelques représentants qui sont :

- QBE
- LAGRIF
- autres interfaces

I.1.2) QBE :

Le langage QBE (Query By Example) permet l'interrogation d'une Base de Données relationnelle directement sous forme de tableaux. Nous allons étudier QBE sur un exemple de gestion des vols d'une compagnie aérienne. Soient les relations suivantes :
VOL (NO VOL, NO AVION, V-DEPART, V-ARRIVEE, H-DEPART, H-ARRIVEE)

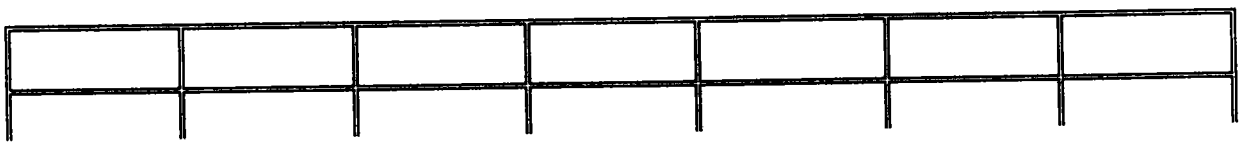
AVION (NO AVION, TYPE-AVION, CAPACITE)

Ces relations expriment les renseignements concernant les vols (numéros de vols et d'avions, heures et villes de départ et d'arrivée) et les avions (numéro d'avion, type d'avion et capacité).

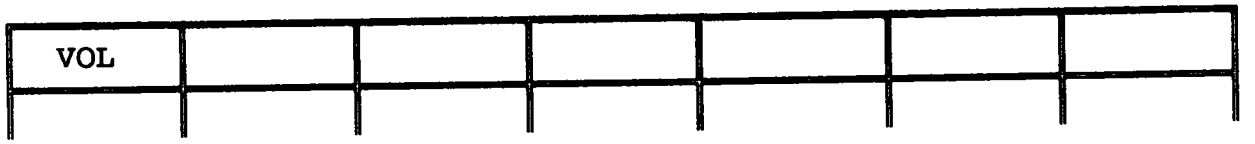
Nous allons étudier QBE par le biais de deux requêtes :

* Première requête :

Considérons la requête suivante : "Quels sont les numéros de vols au départ de Paris ?"
Le système présente initialement une question à blanc ("un cadre") à l'utilisateur sur l'écran.



L'utilisateur indique ensuite le nom de la relation concernée par la requête (c'est-à-dire VOL) dans la partie gauche du cadre.



Le système répond en donnant la liste des attributs composant la relation cible.

VOL	NO VOL	NO AVION	V-DEPART	V-ARRIVEE	H-DEPART	H-ARRIVEE

L'utilisateur remplit alors le tableau de la façon suivante :

VOL	NO VOL	NO AVION	V-DEPART	V-ARRIVEE	H-DEPART	H-ARRIVEE
	P.		Paris			

Ce tableau fournira la liste des numéros de vols au départ de Paris (P. comme print).

* Deuxième requête :

Supposons que l'on veuille maintenant poser la requête suivante :
"Quelles sont les heures de départ des vols assurés pour un Airbus ?".
L'utilisateur va pour celà indiquer au système le nom des deux relations concernées par la requête : il s'agit de VOL et AVION. Tous les attributs de ces relations seront alors affichés à l'écran. Puis il suffit de remplir les tableaux ainsi :

VOL	NO VOL	NO AVION	V-DEPART	V-ARRIVEE	H-DEPART	H-ARRIVEE
		AV			P.	

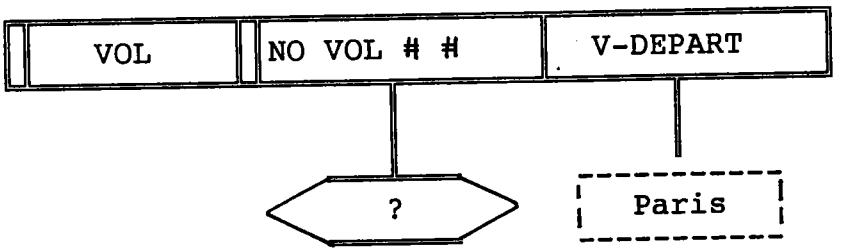
AVION	NO AVION	TYPE=AVION	CAPACITE
	AV	Airbus	

La variable soulignée (AV) permet de faire la jonction entre les deux relations.

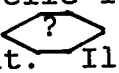
I.1.3) LAGRIF :

LAGRIF signifie LAngage GRaphique pour non InFormaticiens. L'utilisateur construit sa requête graphiquement à partir de symboles prédéfinis et de commandes simples. Le clavier n'est utilisé que pour saisir des constantes.
Nous allons donner les images écrans correspondant aux deux requêtes étudiées pour QBE.

* Première requête :
"Quels sont les numéros de vols au départ de Paris ?"



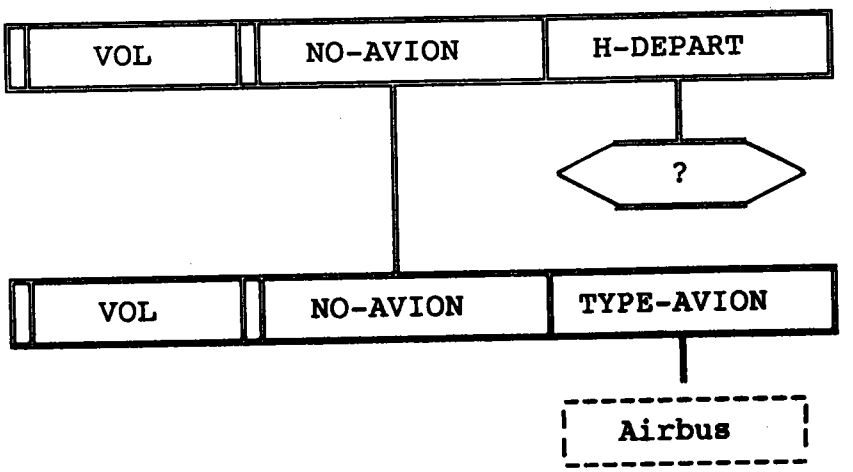
L'utilisateur crée le schéma utile (sans rien taper sur le clavier). Ce schéma est constitué de la relation VOL avec uniquement ses attributs participant à la requête c'est-à-dire NO VOL (# # indique qu'il s'agit d'un attribut clé) et V-DEPART.

Ensuite l'utilisateur relie l'attribut (NO-VOL ERREUR) au symbole de recherche  qui indique que cet attribut fait partie du résultat. Il connecte ensuite l'attribut V-DEPART à la constante "Paris" de la façon suivante :

- sélection du symbole graphique 
- saisie de la constante "Paris"
- connection de la constante à l'attribut.

* Deuxième requête :

"Quelles sont les heures de départ des vols assurés par un Airbus ?"
Cette requête sera exprimé par l'écran suivant :



La liaison entre les deux relations est exprimée par une connexion entre les attributs communs.

I.1.4) Autres interfaces :

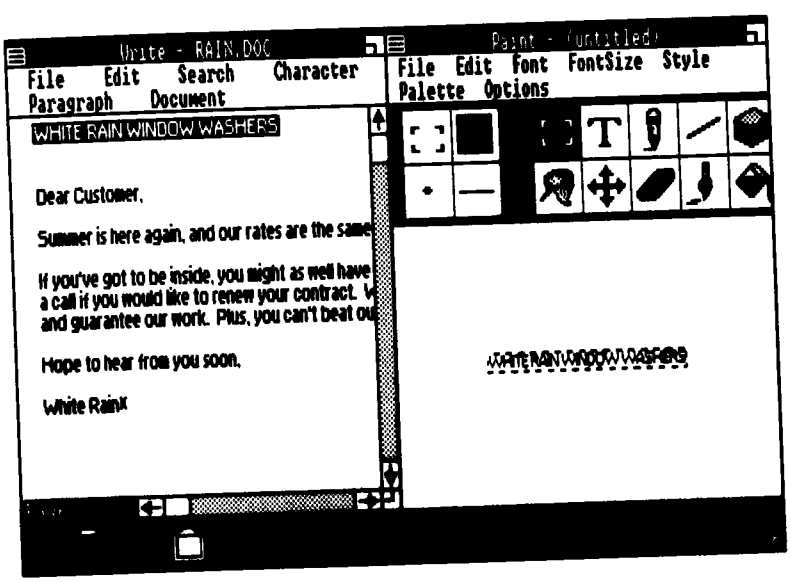
Dans la plupart de ces interfaces, l'interaction avec l'utilisateur tend à être de type Macintosh c'est-à-dire s'appuyant sur :

- l'utilisation de la souris
- les menus déroulants
- le multifenêtrage
- les zones de défilement (ascenseurs, etc...)
- les possibilités graphiques
- etc...

Nous reverrons plus en détails ces propriétés dans un prochain paragraphe.
Parmi les SGBD pour micros possédant ces interfaces on peut signaler DBASE III, INFORMIX, ORACLE et 4ème DIMENSION.
Toutes ces interfaces permettent à l'utilisateur non - informaticien de créer puis de manipuler sa propre base après seulement quelques heures d'apprentissage.

I.2) Interfaces pour la bureautique :

L'automatisation des tâches administratives est un domaine où il existe de nombreux outils informatiques destinés à des utilisateurs finaux. Ces outils vont du traitement de texte au tableur en passant par les gestionnaires de documents ou les logiciels graphiques : les systèmes de gestion de bases de données que nous venons d'étudier en font partie.
Ces progiciels de bureautique sont pourvus d'interfaces dont les caractéristiques sont identiques à celles de l'environnement Macintosh décrites au paragraphe I.1.3 (utilisation de la souris, menus déroulants, etc...)
Le schéma est un écran utilisé dans le cadre du gestionnaire de document "Windows Write" disponible sous MS-DOS.



Les caractéristiques de cet écran sont les suivantes:

- multifenêtrage (fenêtres alphanumériques, graphique, de commande, d'état, etc...)
- utilisation de la souris
- les symboles graphiques représentant les différents types d'actions possibles (gomme, déplacement, dessiner, écrire, création de caractères, etc...).

I.3) Conclusion :

Les interfaces que nous avons étudiées précédemment sont la face visible des langages de 4ème génération. Nous avons décrit jusque là essentiellement les propriétés de représentation des informations de ces interfaces. En dehors de la convivialité tout langage de quatrième génération doit se caractériser par :

- sa facilité d'apprentissage.
- une syntaxe non procédurale.
- des facilités d'aides.

Ces fonctionnalités se retrouvent dans le dialogue offert par ces langage de 4ème génération.

Nous allons maintenant définir le dialogue (et donc l'interface) qui sera associé au Modèle. Ce dialogue reprendra un certain nombre de concepts des dialogues pour langages de 4ème génération que nous venons de décrire.

II) Recherche d'une interface :

II.1) Objectifs de cette interface :

Le Modèle que nous avons défini peut être considéré comme un outil d'aide à la conception complémentaire d'un outil de DAO. Afin d'en faciliter l'usage dans le cadre d'une conception mécanique il est indispensable d'associer au Modèle une interface de haut niveau. Pour définir cette interface nous utiliserons un certain nombre de caractéristiques propres à celles des interfaces de haut niveau actuelles que nous avons étudiées dans le paragraphe précédent.

Nous allons d'abord établir le cahier des charges d'une interface adaptée à notre Modèle. Ce cahier des charges est constitué des éléments suivants :

- les principes du dialogue
- les outils gérés
- la liste des données et des traitements pris en compte.

II.1.1) Les principes du dialogue :

Dans le but d'avoir un dialogue aussi proche que possible du travail de conception classique des bureaux d'études (dont les outils essentiels sont un crayon, une gomme et un ensemble de catalogues), nous allons donner une liste de concepts à partir desquels sera bâti ce dialogue.

- la convivialité : le dialogue sera basé autour du multifenêtrage avec pour chaque écran une fenêtre graphique. Dans le but d'une simplification de ce dialogue on minimisera le nombre de menus (ce qui implique un regroupement des traitements) et les contraintes de passage d'un menu à l'autre afin de tendre vers la possibilité d'effectuer n'importe quel traitement quelque soit le contexte.

- manipulation simultanées d'informations alphanumériques et graphiques.

- utilisation du zoom : afin de passer de la description générale d'un objet au détail de ses propriétés.

- les aides : permettant par exemple de décrire une procédure (quel est son but, quels sont ses paramètres,...) ou un ensemble.

- possibilité d'utiliser la souris.

- possibilité de créer ses propres primitives graphiques.

II.1.2) Les outils gérés :

Afin de représenter les objets de conception on utilisera un certain nombre de primitives graphiques : celles-ci sont mémorisées dans une Base de Dessins. Un dessin peut représenter une figure géométrique de base (cercle, carré, triangle, etc...) ou encore une figure géométrique spécifique au concepteur. Il en résulte que l'interface associée au Modèle doit être capable non seulement de manipuler des primitives graphiques (affichage, déplacement, suppression,...) mais aussi de les créer : elles seront alors intégrées dans la Base de Dessins.
En plus de la Base de Dessins, l'interface aura aussi à prendre en compte la Base de Procédures dans les cas d'attachement et de déclenchement procéduraux.

II.1.3) Les données et les traitements pris en compte :

Les types de données qui apparaîtront au niveau du dialogue sont :

- des objets (prototypes ou instances de ces prototypes).
- des ensembles.
- des procédures.

Dans le cadre du dialogue on doit prendre en compte les opérations suivantes :

- création.
- suppression.
- modification.
- affichage.
- recherche.

L'ensemble des opérations possibles selon les types de données sur lesquels sont appliqués ces opérations peut être représenté par le tableau qui suit: dans ce tableau pour chaque opération effectuée, selon la donnée concernée (prototype, instance, ensemble, procédure) on décrit le composant de cette donnée affecté par l'opération (attribut, facette, élément d'ensemble,..).

	création	suppression	modification	recherche	affichage
prototype	attribut descriptif	oui	une facette d'un attribut s., p.f l.b. ou c.i. (1) (directe ou indirecte)	par son nom, les prototypes d'une p.f., les prototypes référencés dans un attribut s., p.f., l.b. ou c.i. (1)	entier ou partiel (certains attributs)
instance	instanciation d'un attribut l.b. ou c.i. (1)	oui	une instance d'un attribut s., p.f. l.b. ou c.i. (1) (directe ou indirecte)	par le prototype associé et le numéro d'instance	entier ou partiel (les instances de certains attributs)
ensemble	oui	oui	création ou suppression d'un élément de l'ensemble.	par son nom	entier
procédure	impossible	impossible	impossible	par son nom	impossible (certaines informations peuvent être obtenues lors de l'attachement procédural).

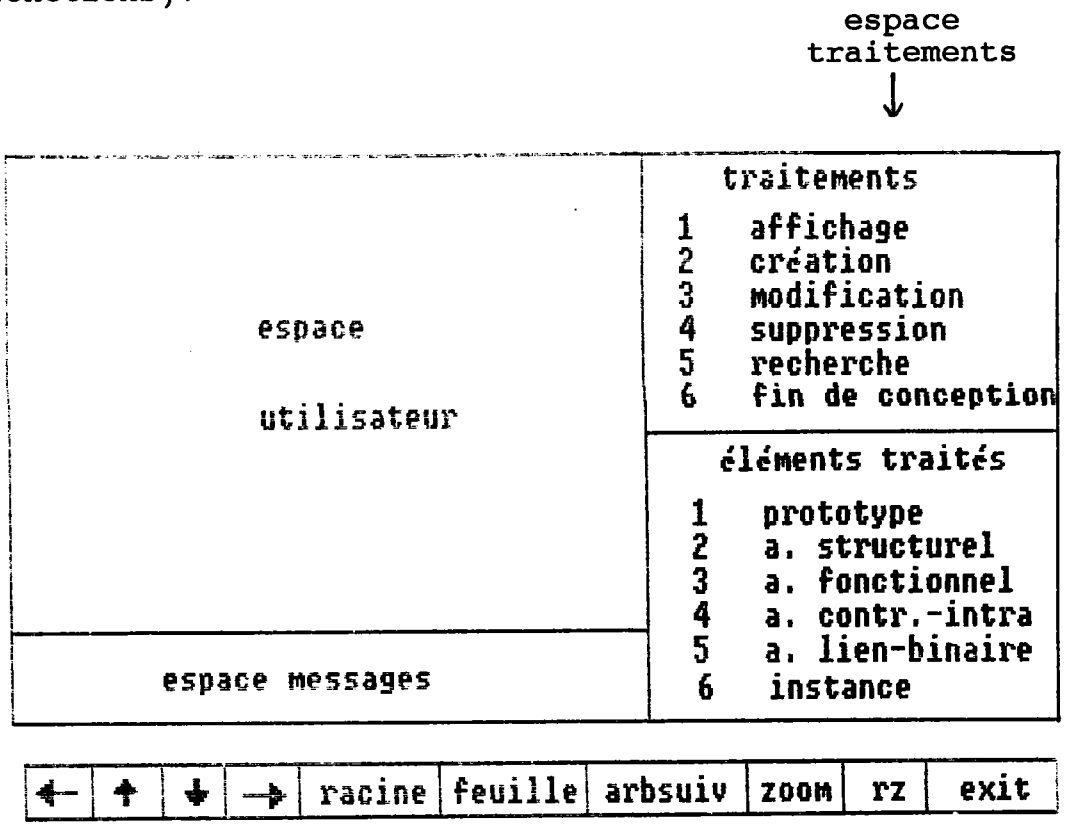
Ensembles des opérations possibles dans le cadre du dialogue

(1) s.=structurel, p.f.=partie-fonctionnelle, l.b.=lien-binaire, c.i.=contrainte-intra

I.2) Première interface:

II.2.1) Présentation générale:

Cette interface permet de créer et de manipuler simultanément les données. Elle est construite a partir de l'écran principal suivant (avec description des touches de fonctions).



Cet écran est découpé en trois fenêtres principales qui sont:

- l'espace utilisateur.
- l'espace messages.
- l'espace traitements.

Les touches de fonctions sont utilisées dans le cadre de la recherche et de la visualisation d'objets.

L'espace utilisateur permet les affichages des différents éléments de conception : objet, attributs et ensembles. On définit dans cet espace la notion de prototype courant.

L'espace traitements permet à l'opérateur de choisir un traitement donné: par exemple création d'un prototype (qui deviendra alors le prototype courant) ou suppression d'une facette d'attribut contrainte-intra d'un prototype donné. Dans chacun de ces cas de figure, les éléments de l'espace traitements associés au traitement actif seront en reverse vidéo.

II.2.2) Exemples de traitements :

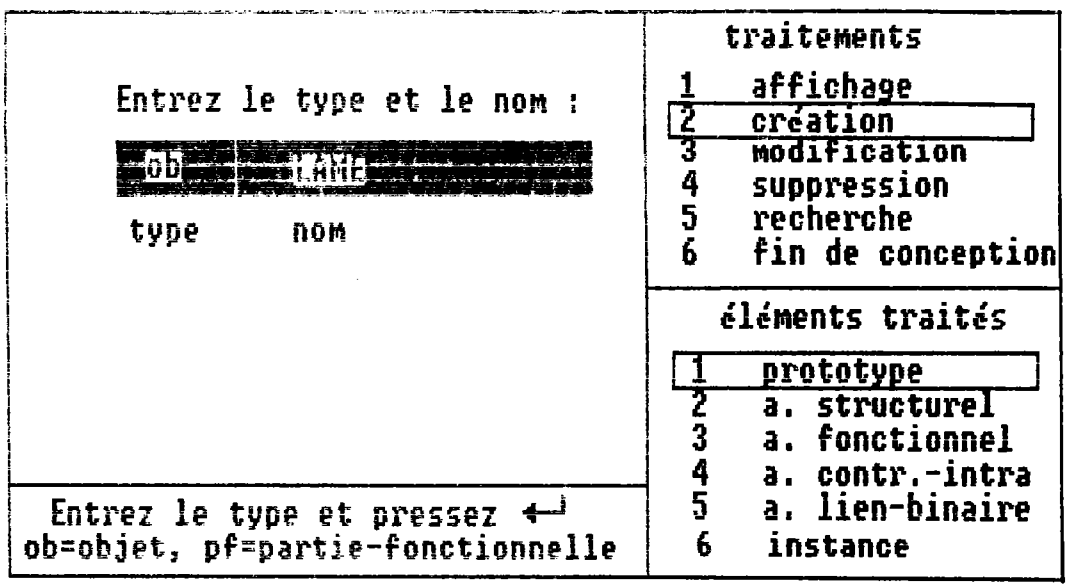
Nous allons maintenant décrire le dialogue proposé par cette interface par l'intermédiaire de quelques exemples de conception qui sont:

- la création d'un objet.
- la recherche d'un objet.
- l'affichage d'un objet.
- la création d'une instance.

L'objet sur lequel sont effectués ces traitements est la lame d'un tournevis.

II.2.1) Création d'un objet :

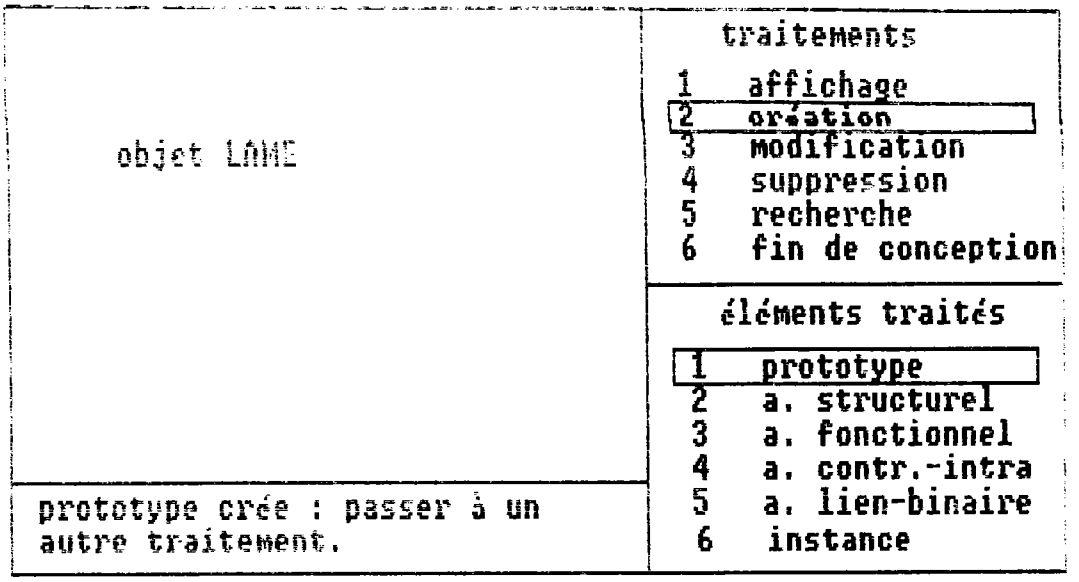
Supposons que l'on veuille créer un objet "lame": cette création revient à celle du prototype associé. Le processus de création correspond aux images écrans suivantes:



Les choix de "création" et de "prototype" sont faits par le biais de la souris et impliquent l'affichage automatique dans l'espace utilisateur d'une grille de création d'un prototype . Le curseur est positionné dans la zone "type et nom" : cette zone est destinée à recevoir le type du prototype puis son nom.

L'espace message donne des indications sur la façon de saisir les informations : en cas de problème (type inexistant ou prototype déjà créé), cet espace contiendra un message décrivant la nature du problème et la démarche à suivre.

Aprés avoir frappé "ob" puis "LAME", l'écran suivant va apparaître:



L'objet courant devient l'objet "LAME" et à partir de là, on peut passer à la création de ses attributs puis de ses instances : pour celà, il faut choisir les options correspondantes dans la fenêtre "éléments traités".

II.2.2.2) Recherche d'un objet:

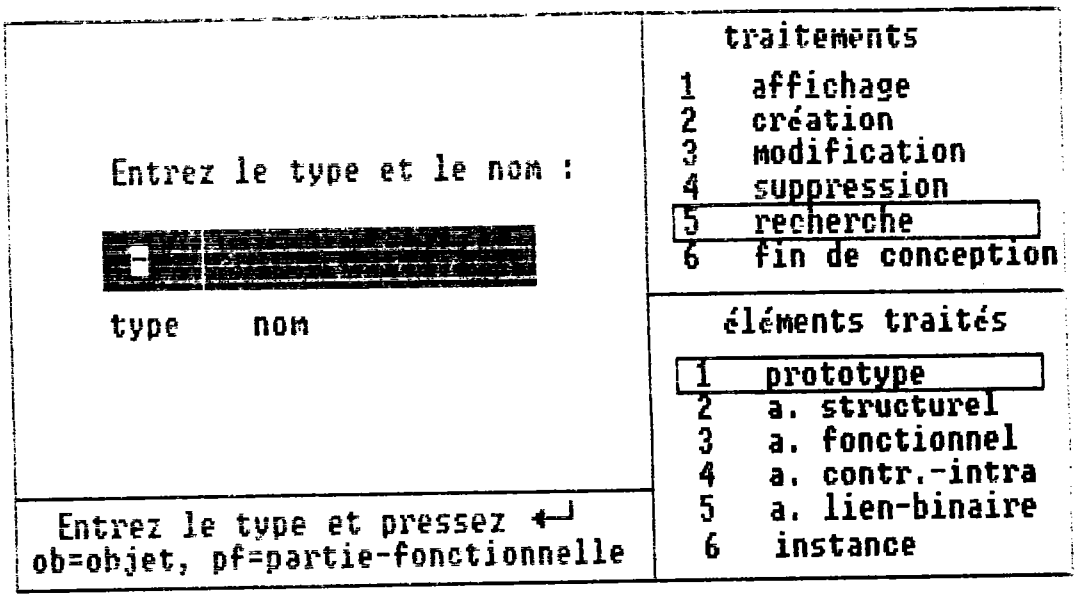
On peut retrouver les informations concernant un objet de deux manières différentes :

- soit directement en nommant cet objet.
- soit indirectemenet par un autre objet qui lui est lié hiérarchiquement.

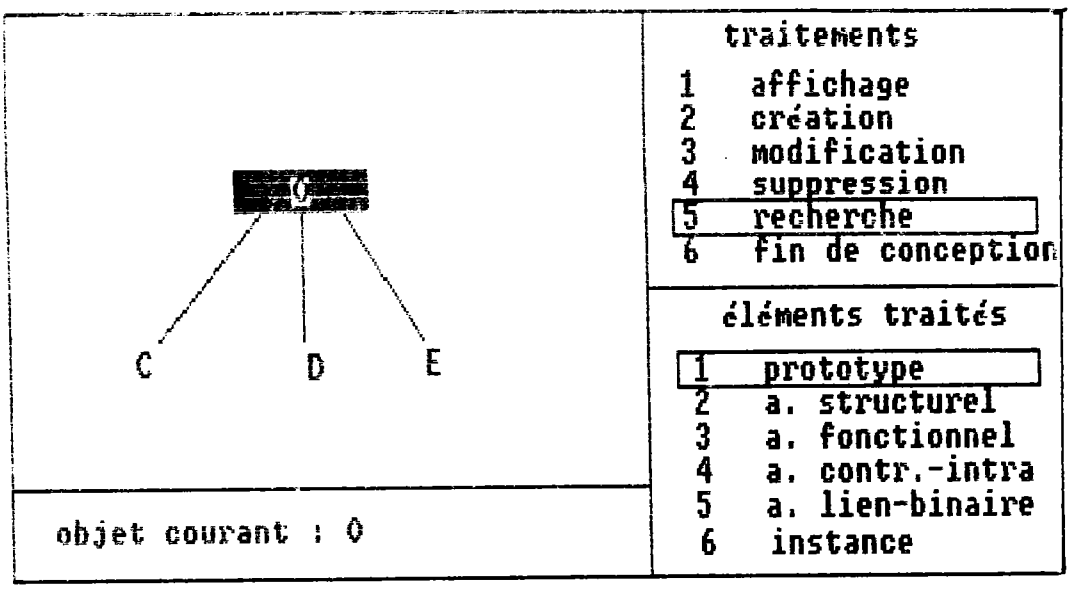
Nous allons étudier un exemple dans lequel sont conjugués ces deux modes de recherche.

Supposons que l'on veuille avoir des informations sur un objet "O" (on admettra que cet objet appartient à un autre objet "A" et qu'il contient trois objets "C","D" et "E").

Pour rechercher l'objet "O" dans la Base, la démarche à suivre est la suivante : valider les options "recherche" et "prototype" ce qui entraîne l'apparition de l'écran suivant:



L'opérateur saisit ensuite "OB" puis "O" (même processus que pour la création d'un objet). L'écran suivant apparaîtra:



L'arborescence affichée dans la zone utilisateur est à deux niveaux. Il est possible de parcourir cette arborescence grâce aux touches de fonctions. Ce parcours se fait à partir de l'objet courant (celui-ci est toujours visualisé en reverse vidéo).

a) <u>Description des touches de fonctions</u>:

* Parcours de l'arborescence:

- La touche "↑" permet d'obtenir comme objet courant le père de l'ancien objet courant : l'appui sur cette touche implique un affichage dans la zone utilisateur de l'objet "A" et de ses fils (dont "O").

- La touche " ↓ " permet d'obtenir comme nouvel objet courant le premier des fils de l'ancien objet courant: dans le cas présent l'appui sur cette touche valide "C" comme nouvel objet courant (un deuxième appui de cette touche implique un affichage dans la zone utilisateur de "C" et de ses fils).

- Les touches "<--" et "-->" permettent un défilement à l'écran des différents fils de l'objet courant dans le cas où l'espace utilisateur est trop petit pour les afficher tous simultanément (1).

- La touche "racine" est utilisée pour obtenir l'objet racine de l'arbre courant.

- La touche "feuille" permet d'obtenir directement à l'écran, la première feuille du sous-arbre dont l'objet courant est racine.

- La touche "arbsuiv" implique un passage à un autre arbre: l'objet racine de cet arbre deviendra objet courant.

On peut signaler aussi d'autres touches de fonctions qui ne correspondent pas à des déplacement dans l'arborescence. Voici la description de quelques unes d'entre elles :

* Autres touches de fonctions:

- "zoom" implique un affichage dans l'espace utilisateur des attributs de l'objet courant.

- "rz" est la fonction inverse "zoom".

- "exit" est une touche de fonction permettant d'interrompre la transaction en cours et de retourner à la transaction précédente : par exemple si dans un session de création d'un attribut on conçoit un autre prototype (cas de la création imbriquée) il est possible d'interrompre la création de ce prototype et de revenir automatiquement à la création de l'attribut.

II.2.2.3) Affichage d'un prototype:

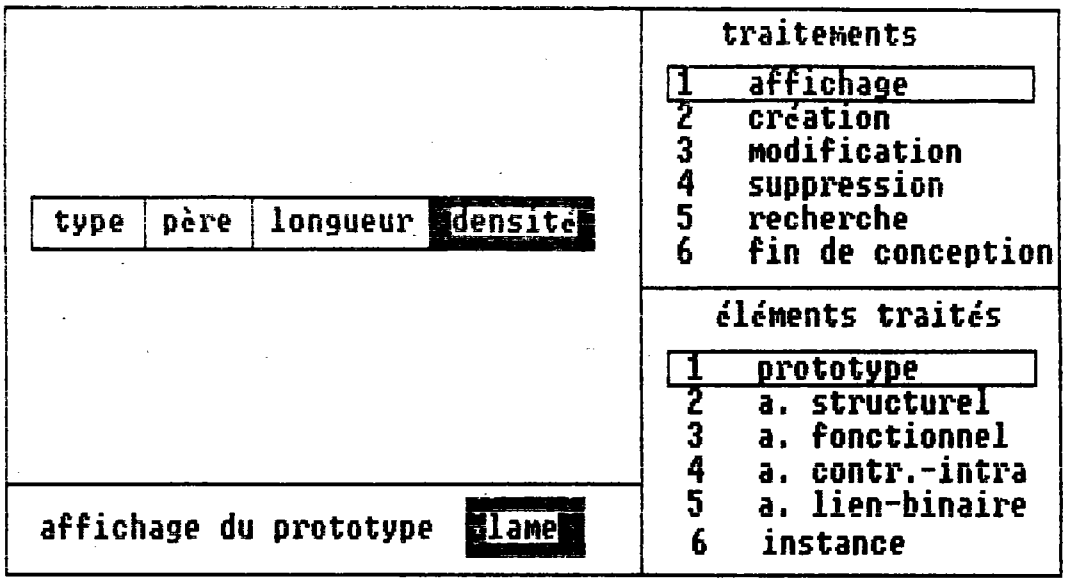
L'affichage d'un prototype donné correspond à l'affichage de tous ses attributs. Cet affichage peut être obtenu dans les cas de figures suivants :

- directement à partir des options de l'espace traitement en validant les zones "affichage" et "prototype" : il suffit alors de nommer le prototype.

(1) Ces quatre touches de fonctions sont aussi utilisées dans le cadre de l'affichage des propriétés d'un objet.

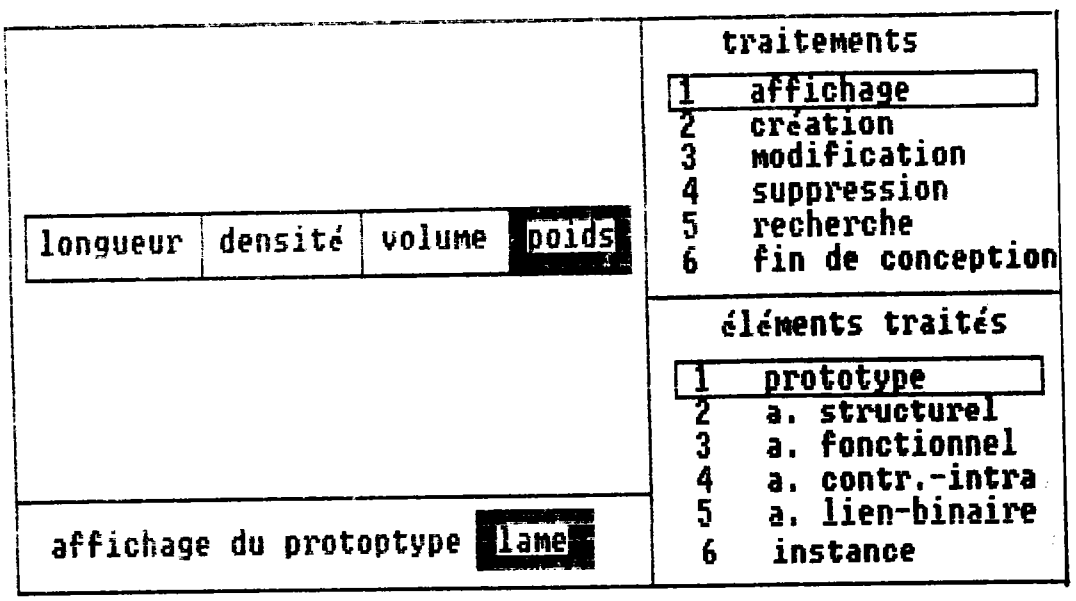
- suite à une recherche de ce prototype: directement en utilisant les options correspondantes de l'espace traitement ou indirectement par parcours de son arborescence. L'utilisation de la fonction "zoom" permet ensuite d'afficher l'ensemble de ses attributs.

Supposons que l'objet courant soit "LAME" et que l'on ait demandé un affichage de ses attributs. L'écran correspondant est décrit ci-dessous:

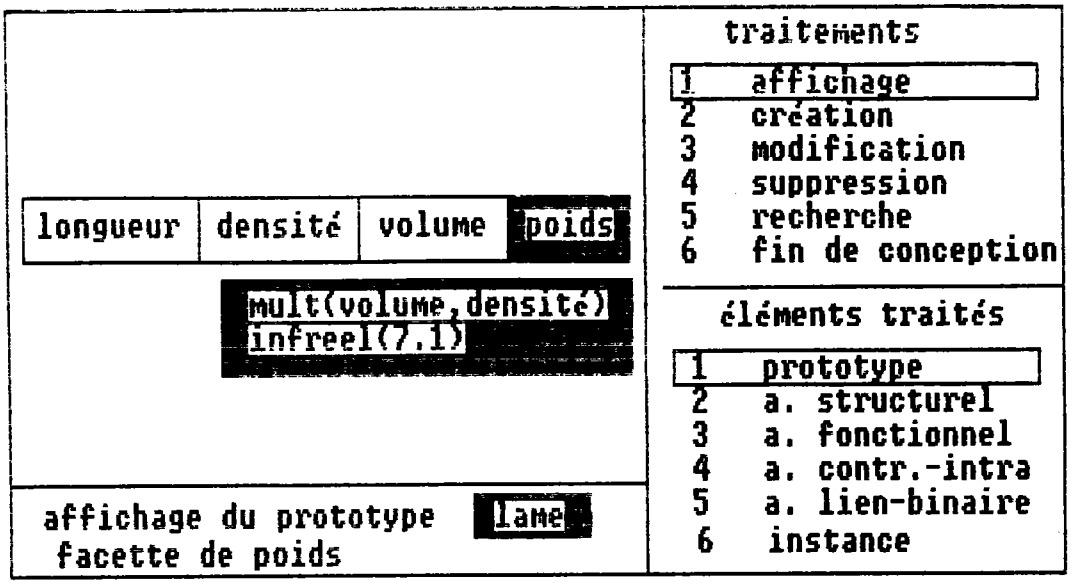


Dans l'espace utilisateur sont affichés les noms des attributs de l'objet. L'attribut courant est le dernier attribut visualisé: dans le cas présent il s'agit de l'attribut "densité".

L'appui sur la touche "-->" implique l'écran suivant:



Les attributs suivants sont alors affichés : l'attribut courant devient "poids". L'appui sur la touche de fonction "zoom", impliquera l'écran ci-dessous:



Les différentes facettes de l'attribut "poids" sont alors visualisées.

Remarque: Les touches de fonctions "-->" et "<--" sont utilisées pour faire défiler les différents attributs du prototype, tandis que les touches de fonctions " ↑ " et " ↓ " permettent le défilement des facettes de l'attribut courant.

II.2.2.4) Création d'une instance:

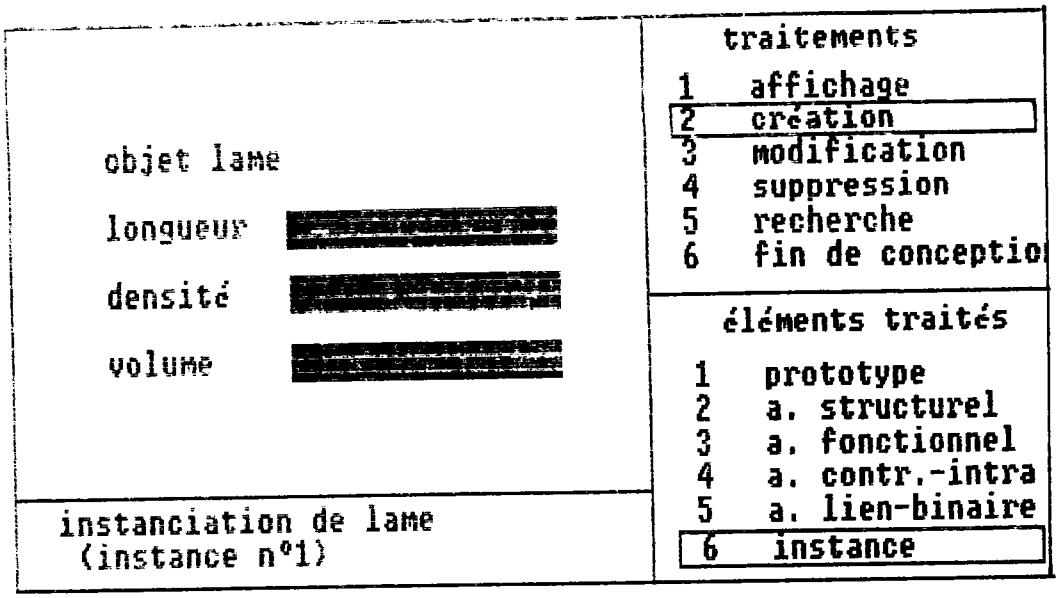
Supposons que l'on veuille créer la première instance de l'objet lame définie par :

```
<instance> lame
 longueur : 2.
 densité  : 1.
 volume   : 3.
```

L'attribut poids étant calculé, son instanciation est automatique: cet attribut ne sera donc pas directement accessible à l'opérateur.

Pour effectuer l'opération d'instanciation, il faut en premier lieu rendre l'objet "lame" objet courant (se reporter aux paragraphes précédents), ensuite l'opérateur doit sélectionner dans l'espace traitement les options "création " et "instance".

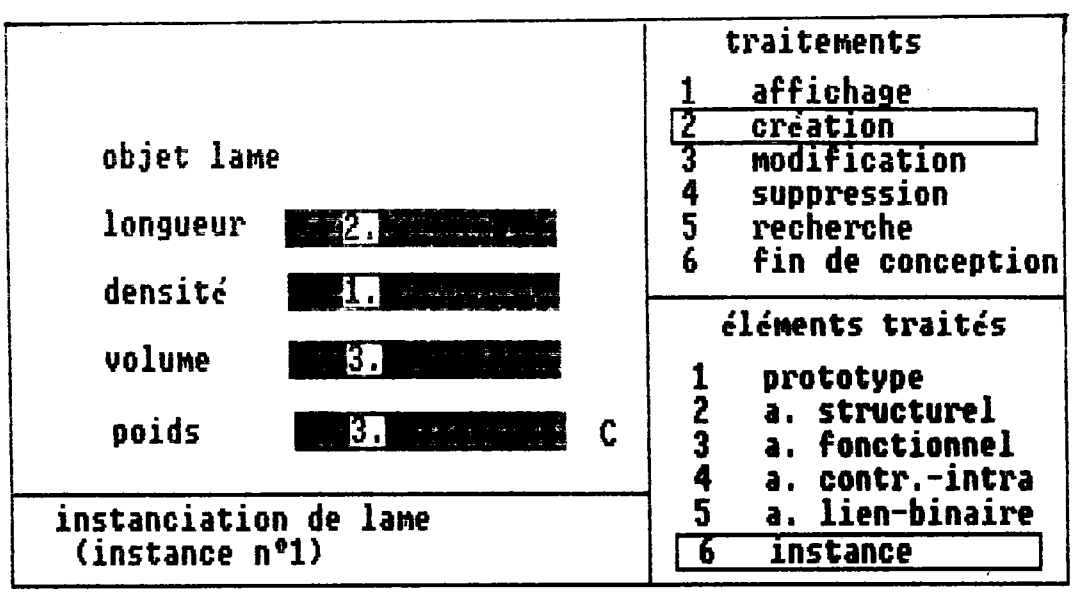
L'écran qui apparaîtra alors demandera le numéro d'instance à créer : il faut dans ce cas répondre par "1". Suite à cette réponse, le nouvel écran sera:



L'instanciation se fait par saisies successives des occurences de ces trois attributs comme dans le cas d'une grille de saisie. Il est possible de ne pas instancier tous les attributs.

Dans le cas d'une incohérence d'une instance d'attribut avec l'ensemble des valeurs admissibles pour cet attribut (contrainte forte), un message d'erreur sera émis dans l'espace messages.

Dès que les instances des attributs "densité" et "volume" auront été créés, l'instance déduite de l'attribut "poids" sera automatiquement affichée. Dans le cas où les trois attributs ("longueur","densité" et "volume") ont été instanciés, l'écran associé sera le suivant :



Le poids déduit est égal à 3 (=volume * densité).
Le caractère 'c' affiché à la suite de l'instance de poids indique qu'il s'agit d'un attribut calculé.

II.2.3) Analyse de cette interface:

Cette première proposition d'interface dont nous n'avons exposé que certains concepts, ressemble par la démarche de conception qu'elle impose à celles des systèmes de gestion de bases de données (relationnels) pour micros. Le dialogue induit par cette interface est proche de ce que l'on peut trouver actuellement dans beaucoup de logiciels pour la bureautique. On peut d'ailleurs l'améliorer en intégrant par exemple les fonctions (représentées par les touches de fonctions) dans l'écran par l'intermédiaire de cases sélectionnables par la souris.

Dans le dialogue correspondant à cette interface, la notion d'objet à concevoir reste trop abstraite: on peut conclure qu'il n'est pas assez proche des habitudes de conception du type crayon-papier-gomme.

Nous allons maintenant définir une deuxième proposition d'interface, qui tout en s'appuyant sur certains concepts développés dans la première interface sera plus proche des habitudes de conception en mécanique.

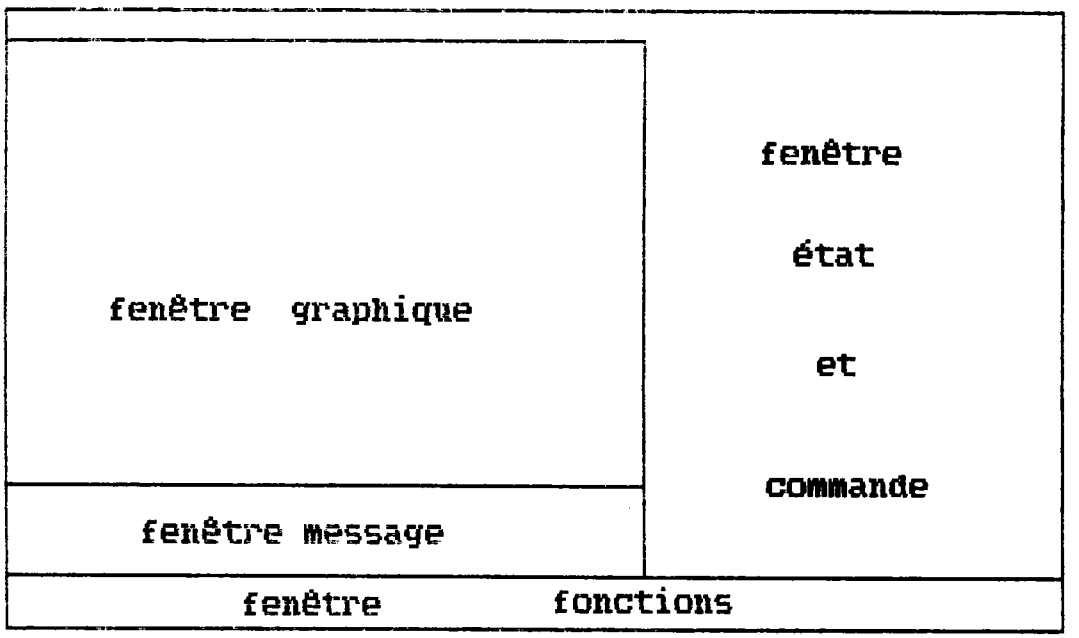
II.3) Deuxième interface :

II.3.1) Principes :

La nouvelle proposition d'interface que nous allons décrire maintenant se veut plus proche des concepteurs (utilisation de primitives graphiques, vocabulaire du dialogue moins théorique, etc ...) tout en ayant un certains nombres de points communs avec la première interface qui se situent :

- au niveau des outils (multifenêtrage, menus déroulants, utilisation de la souris).
- au niveau des traitements proposés à l'utilisateur : les deux interfaces offrent à l'utilisateur avec une forme différente les mêmes possibilités de traitements.

L'ensemble des écrans proposés par cette interface sont découpés en quatre fenêtres de la façon suivante :



- La fenêtre graphique permet de visualiser les objets sous forme de primitives graphiques, de créer ces primitives graphiques, mais aussi d'afficher des informations alphanumériques (attributs, instances, etc ...).
- La fenêtre état et commande est utilisée pour demander au système d'effectuer un traitement donné (par exemple création d'un prototype) mais elle permet aussi à l'utilisateur de savoir à tout moment le type de traitement en cours, le nom de l'objet courant etc ...
- La fenêtre message permet au système d'envoyer à l'utilisateur des messages d'erreurs (incomplétude, incohérence) ou des messages systèmes.
- La fenêtre fonctions est l'équivalent des touches de fonctions que nous avons définies dans la première proposition d'interface.

II.3.2) Eléments de base de l'interface :

II.3.2.1) Vocabulaire du dialogue :

Afin d'adapter le dialogue au vocabulaire de conception mécanique, nous allons utiliser dans cette deuxième interface un ensemble de termes familiers aux mécaniciens. La correspondance entre ces termes et ceux du modèle peut se résumer par le tableau qui suit :

CONCEPTION MECANIQUE	MODELE
pièce ou objet	prototype
dimensionnement	instance
fonction	partie-fonctionnelle
propriété	attribut ou facette
est composé de	fils
appartient à	père
caractéristique d'un objet donné	contrainte-intra
caractéristique liée à deux objets	lien-binaire
liaison	contrainte-intra

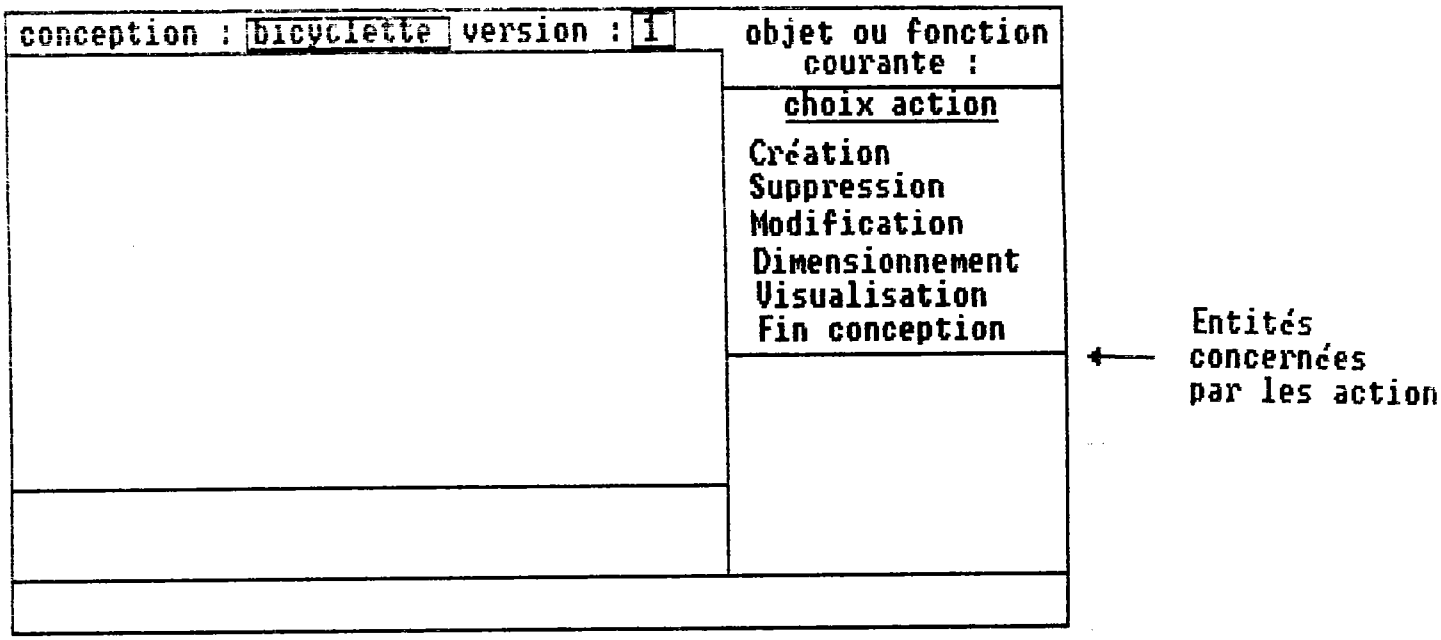
On pourra rappeler aussi que la gestion des informations du cahier des charges et de l'étude cinématique est réalisée par le biais des prototypes tandis que les informations de l'étude statique seront intégrées dans les instances.

L'interface que nous allons définir peut être une interface générale pour la conception mécanique c'est à dire qu'elle simulera la démarche classique de conception en demandant chaque fois à l'utilisateur dans quelle étape il se trouve (cahier des charges, étude cinématique, étude statique).

Une autre solution serait de faire abstraction des ces différentes étapes dans le dialogue et de proposer à l'utilisateur de choisir entre les différentes opérations (création d'un objet, d'une liaison, dimensionnement, etc ...) que l'on peut rencontrer dans un domaine de conception mécanique donné : c'est cette dernière solution que nous avons choisie ici.

II.3.2.2) Description générale :

Au début d'une phase de conception, l'utilisateur indique au système le nom de la conception qu'il envisage d'effectuer ainsi que le nom ou le numéro de la version. Pour une meilleure compréhension, on va supposer que l'on se trouve dans le cas de la conception de la bicyclette (se reporter dans la partie 3, paragraphe II.3) et qu'il s'agit d'en créer la première version. Dès que le nom de la conception (dans notre cas bicyclette) et son identificateur de version (c'est à dire le numéro 1) auront été fournis par l'utilisateur, l'écran suivant va apparaître :



A tout moment d'une conception, le système affiche :
- le nom de la conception
- son numéro de version

La notion d'objet ou de fonction courante est la même que dans la première proposition d'interface : il s'agit de l'objet (ou de la fonction) en cours de conception ou encore l'objet (ou de la fonction) pointée par la souris. En début de conception, l'objet courant pourra être le modèle du projet à concevoir (dans notre cas le premier objet courant serait alors l'objet bicyclette).

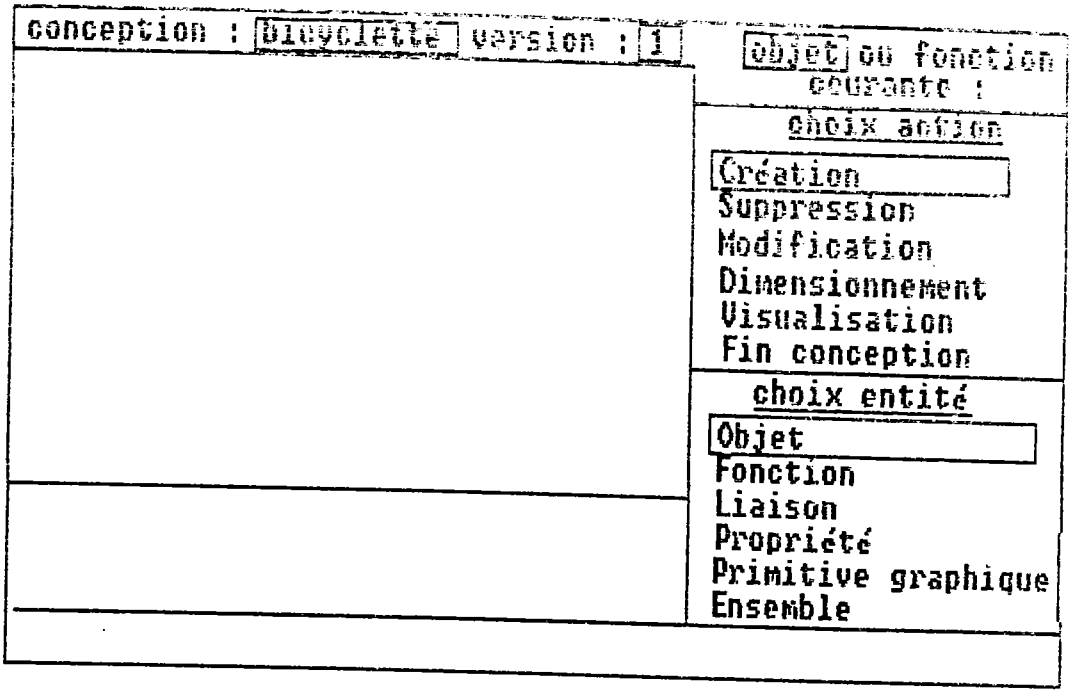
Nous allons décrire maintement quelques concepts du dialogue liés à cette interface dans le cadre de trois exemples qui sont:

- la conception d'un objet.
- la conception d'une fonction.
- la création de propriétés.

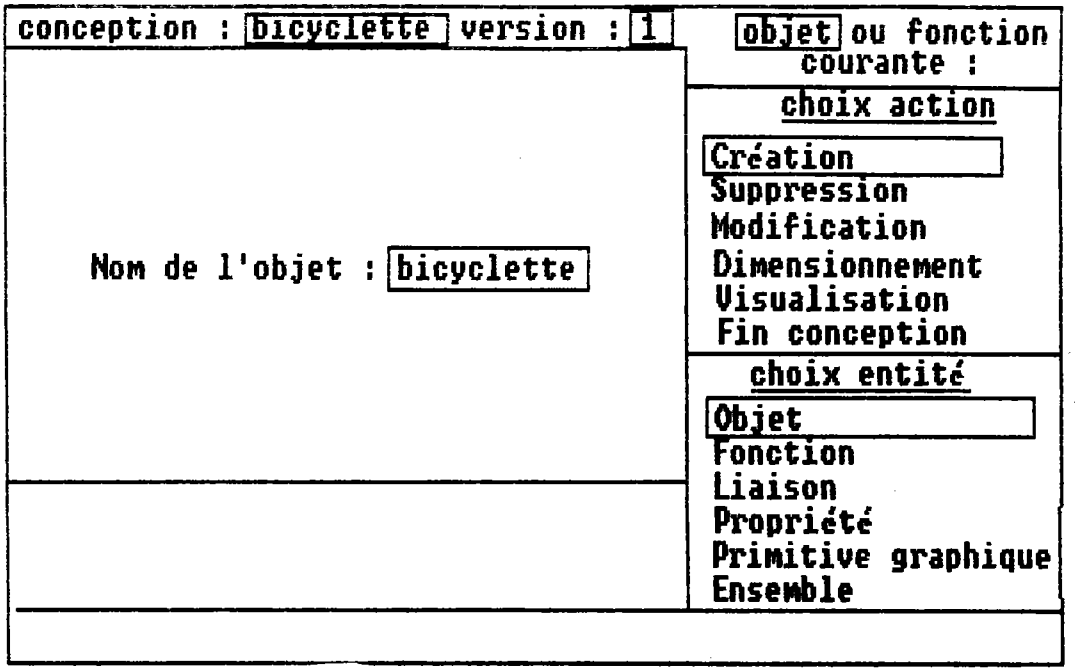
II.3.3) Exemples :

II.3.3.1) Conception d'un objet :

Pour créer un objet, il faut sélectionner création "Création" dans la sous-fenêtre "choix-action". Le système proposera alors à l'opérateur, l'écran suivant:



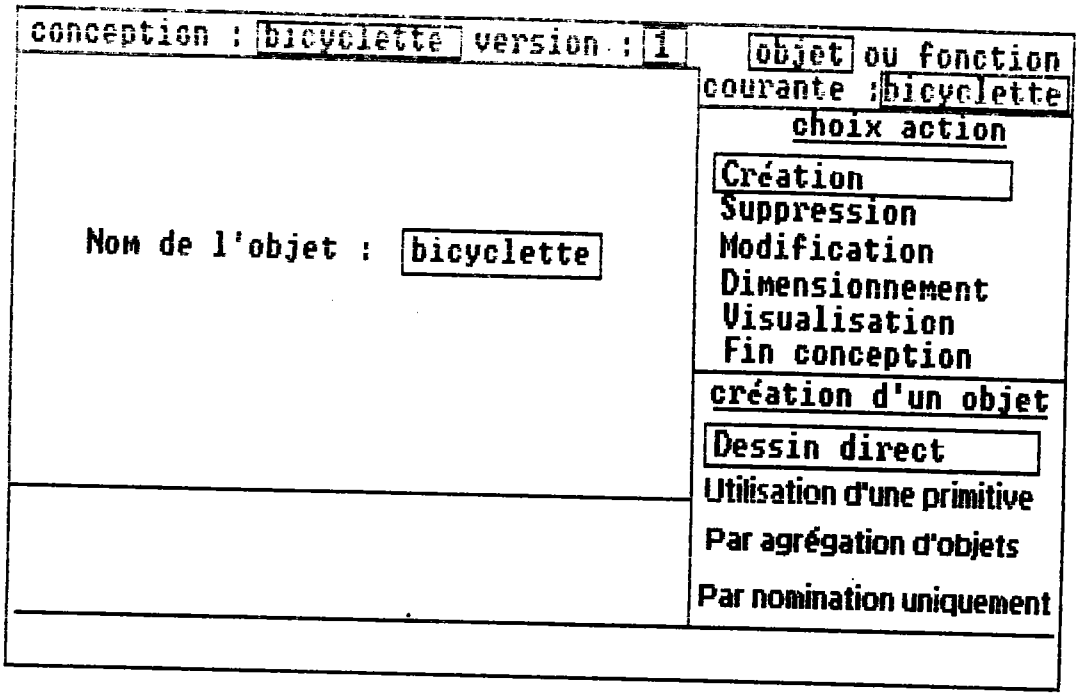
Suite au choix des options création et objet, le système va demander le nom de l'objet à l'opérateur (se reporter à l'écran ci-dessous).



Aprés avoir controlé si cet objet n'existe pas déjà, le système va le prendre en tant qu'objet courant et demander à opérateur de quelle façon il veut le créer. L'utilisateur a le choix entre les options suivantes :

- dessin direct.
- utilisation d'une primitive graphique existante.
- à partir d'autres objets (agrégation d'objets, utilisation des liens structurels).
- par nomination uniquement.

Ces différentes options sont présentées sur l'écran qui suit. On peut remarquer que si on se restreignait aux deux dernières options, le dialogue associé à la création d'un objet serait semblable à celui de la première proposition d'interface.

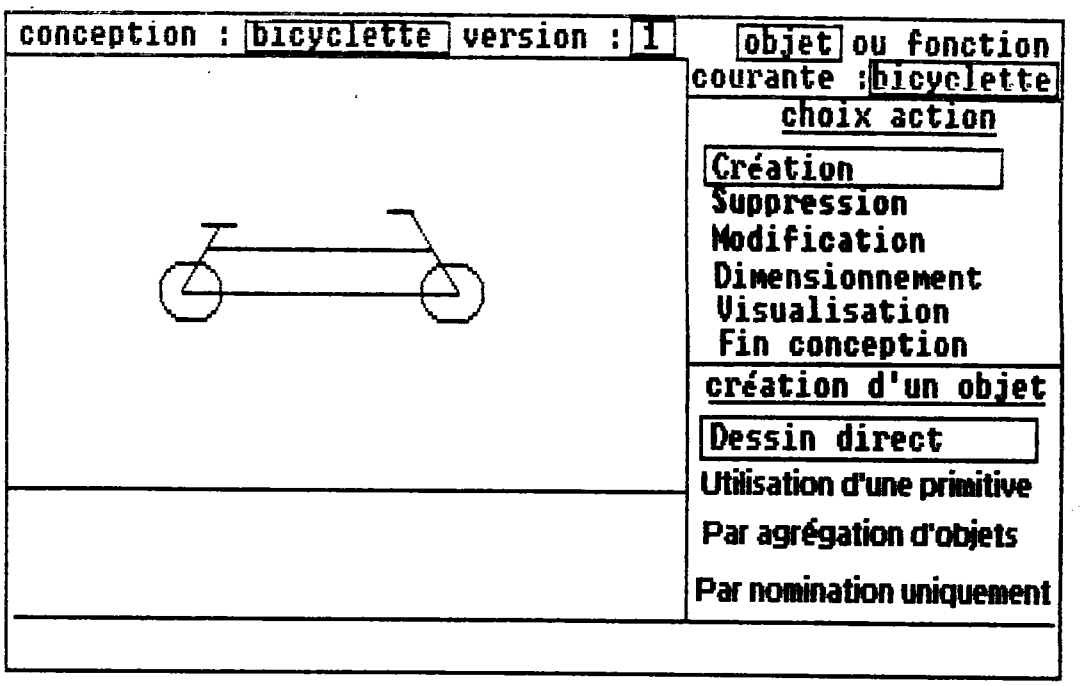


Nous allons montrer maintenant comment associer une primitive graphique à l'objet bicyclette. Pour ce faire, il suffit de choisir un des deux modes de création suivants :

- dessin direct
- utilisation d'une primitive

a) <u>dessin direct</u> :

Supposons que l'on veuille créer l'objet bicyclette en mode dessin direct : dans ce cas la souris va permettre à l'opérateur de dessiner l'objet dans la fenêtre graphique.



L'opérateur peut effectuer son dessin avec la souris dans la fenêtre graphique, en utilisant des outils classiques tels que :

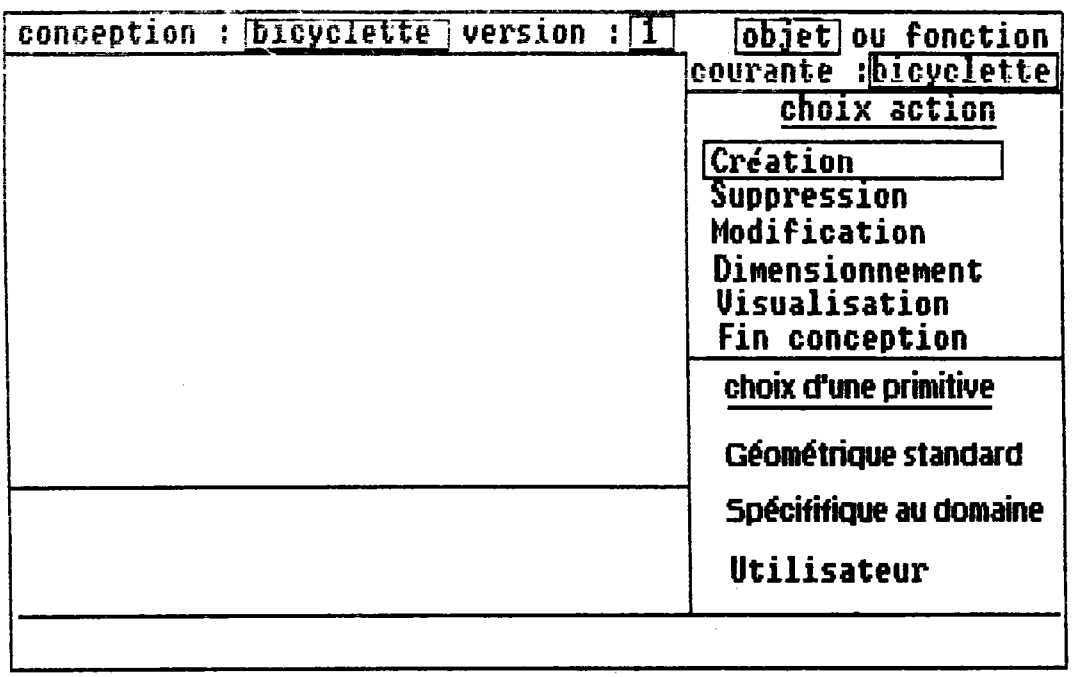
- la gomme
- le choix de l'épaisseur des traits
- les déplacements du dessin et en particulier des rotations
- le zoom (agrandissement, réduction)
- l'effacement de zones d'écran

Chacun de ces outils correspond à une plusieurs fonctions que l'on peut exécuter en validant avec la souris la zone correspondante de la fenêtre fonctions.

Quand l'opérateur a terminé son dessin, il lui suffit de cliquer la fonction "dessin OK" dans la fenêtre fonction.

b) Utilisation d'une primitive :

Le choix de création d'un objet en utilisant une primitive graphique s'effectue en validant les champs "Création" et "Utilisation d'une primitive" de le fenêtre état et commande. L'écran qui suit va correspondre à ce choix :



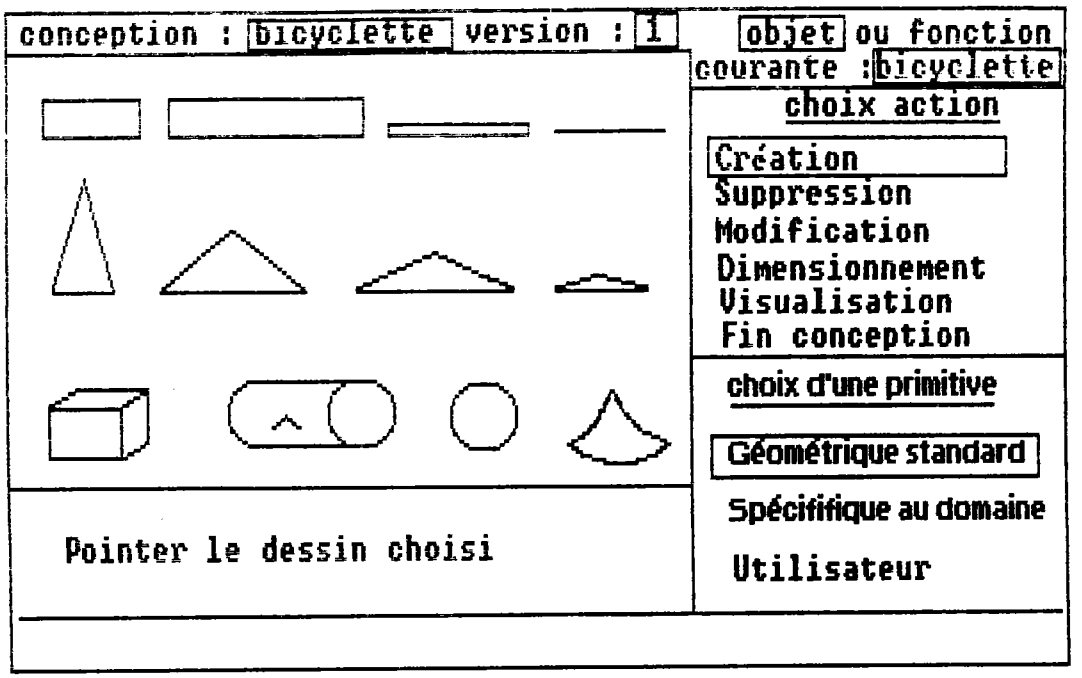
Il est possible d'utiliser différents types de primitives graphiques qui sont les suivants :

- géométriques standards: ces primitives correspondent aux formes de base (carré, triangle, losange, cercle, cube, cylindre, etc...).

- spécifiques au domaine : ce sont des formes géométriques couramment utilisées (parfois normalisées) dans un domaine donné. Par exemple il peut s'agir des symbôles des différents types de torseurs (se reporter annexe 2) que l'on utilise pour la création des liaisons.

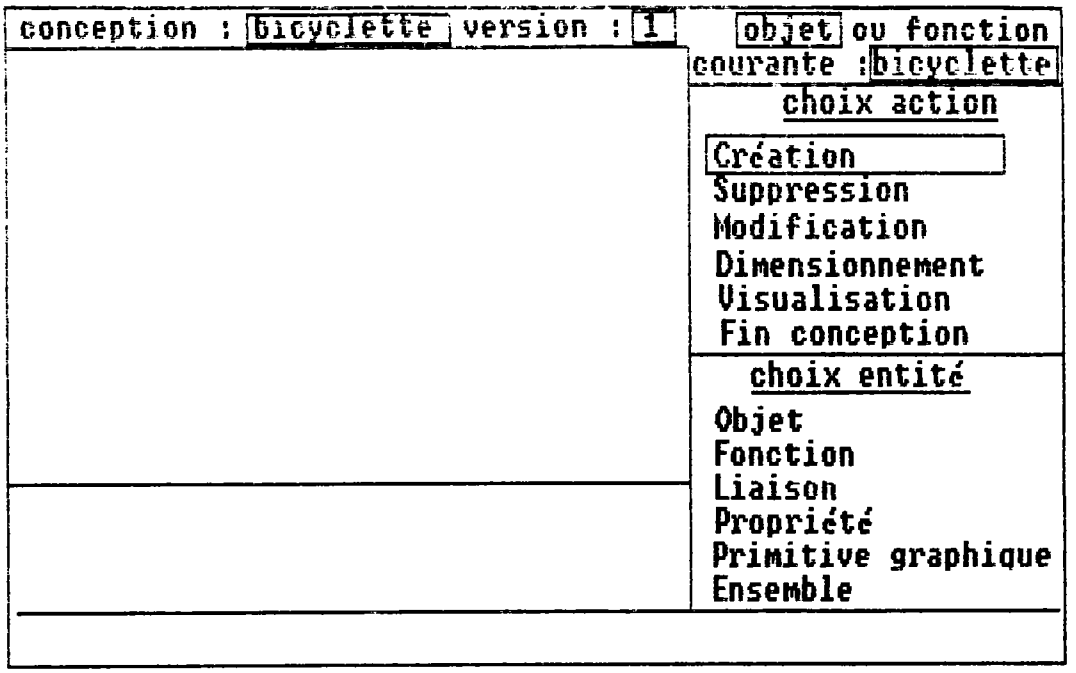
- utilisateur : il s'agit d'un ensemble de dessins spécifiques à l'opérateur; celui-ci peut créer sa propre Base de Dessins (il suffit pour celà de choisir dans la fenêtre état et commande les options"Création" et "Primitives graphiques").

Supposons que l'opérateur veuille choisir une forme géométrique standard : il lui suffit pour celà de valider la zone "Géométrique standard" dans la sous-fenêtre "choix d'une primitive". Le système va lui proposer alors un ensemble de formes de base dans la fenêtre graphique (voir écran ci-dessous)



L'opérateur pointe alors la forme qui l'intéresse (par exemple un cylindre) et il validera son choix par le biais de la fonction validation (fenêtre fonction).

Suite à ces opérations, l'écran suivant apparaîtra :



Supposons que l'opérateur veuille connaître toutes les informations concernant l'objet bicyclette. Il lui suffit pour celà d'effectuer les opérations suivantes:

- choix des zones "Visualisation" et "Objet" qui implique l'affichage d'un cylindre à l'écran.

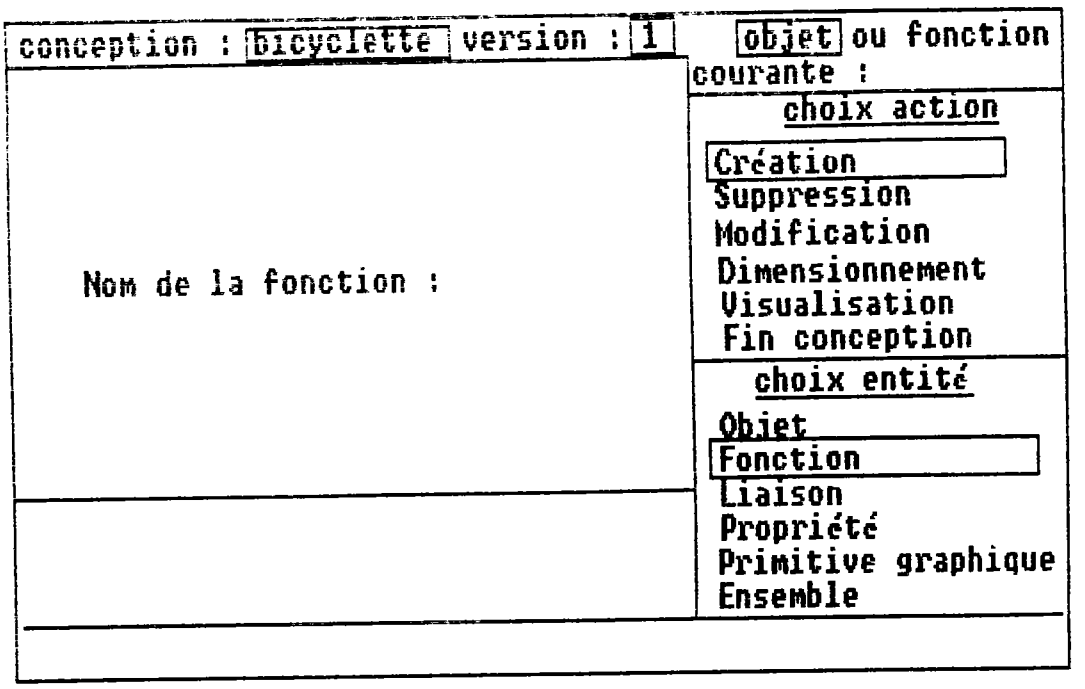
- choix des zones "Visualisation" et "Propriété" qui provoque l'émission du message "AUCUNE PROPRIETE" dans la fenêtre message.

II.3.3.2) Conception d'une fonction :

Dans l'exemple de conception de la bicyclette,le cahier des charges associé comportait 5 fonctions à réaliser ("transporter", "déplacer", "guider", "transmettre" et "maintien de l'ensemble"). Suite à la conception du premier modèle du projet (qui correspond à la création de l'objet "bicyclette"), le concepteur crée ensuite ces 5 fonctions.

Nous allons présenter ici la création de la fonction "maintien de l'ensemble" dans le cadre d'un dialogue décrit par l'intermédiaire des écrans qui suivent.

En premier lieu l'opérateur sélectionne les champs "Création" et "Fonction". Suite à ce choix l'écran ci-dessous apparaîtra:



L'opérateur peut alors saisir dans la fenêtre graphique le nom de la fonction c'est-à-dire "maintien de l'ensemble". La création des autres fonctions s'effectue par le même processus. L'étape suivante sera la conception des différents objets réalisant les fonctions qui viennent d'être définies.

Supposons que pour la fonction "maintien de l'ensemble" on ait déjà conçu les objets "soutien-selle", "barre", "cadre inférieur" et "cadre avant" (1) et que l'opérateur veuille créer les différentes propriétés de l'objet "soutien-selle": nous nous intéresserons plus particulièrement à la création de 2 propriétés correspondant à des contraintes-intra dans le modèle de données. Ces propriétés sont représentées dans le modèle de données par :

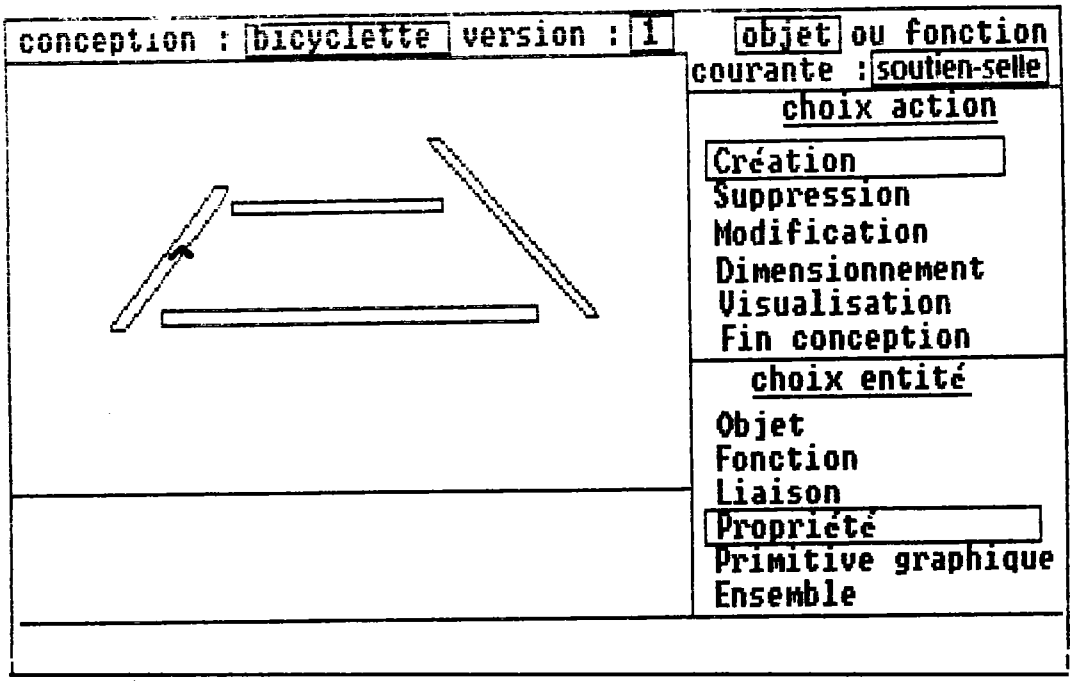
- poids : intervréel (0.25, 0.30)
- diff-angle := différence (angle, (barre, angle))

II.3.3.3) Création de propriétés :

a) Propriété "poids":

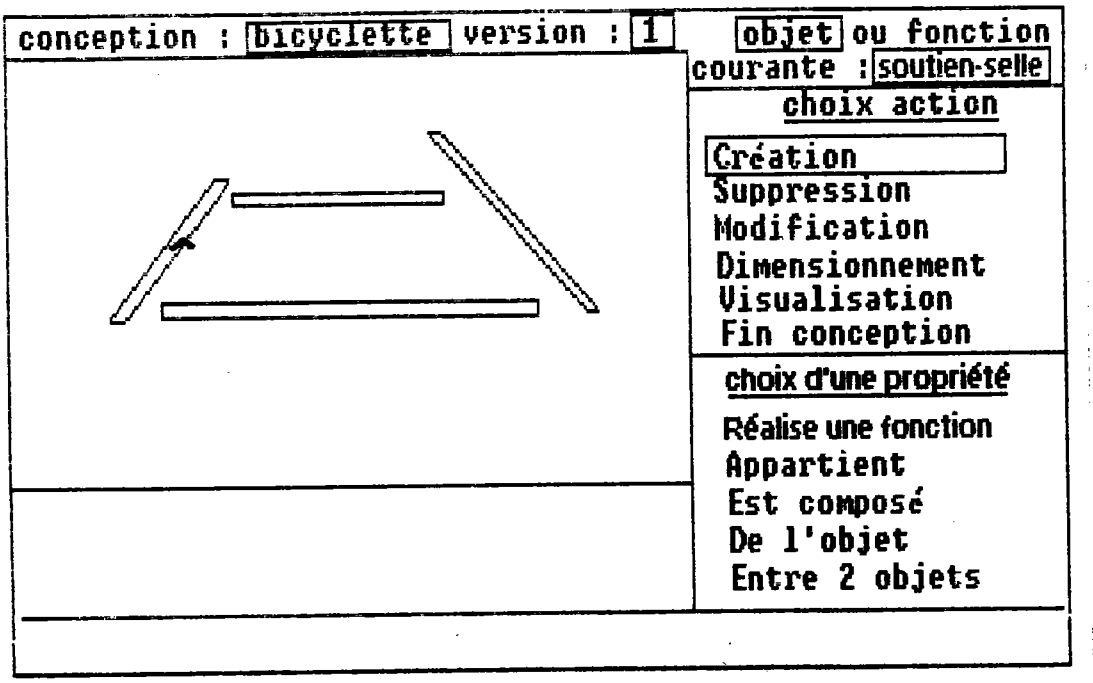
Les propriétés que l'on veut créer concernent toujours l'objet courant. Ainsi si pour l'objet "soutien-selle" l'opérateur veuille lui adjoindre la propriété suivante : "sa longueur est comprise entre 0.25 et 0.30", le dialogue correspondant peut se décrire par les écrans suivants.:

(1) On supposera que tous ces objets ont été créés par le biais des primitives graphiques.



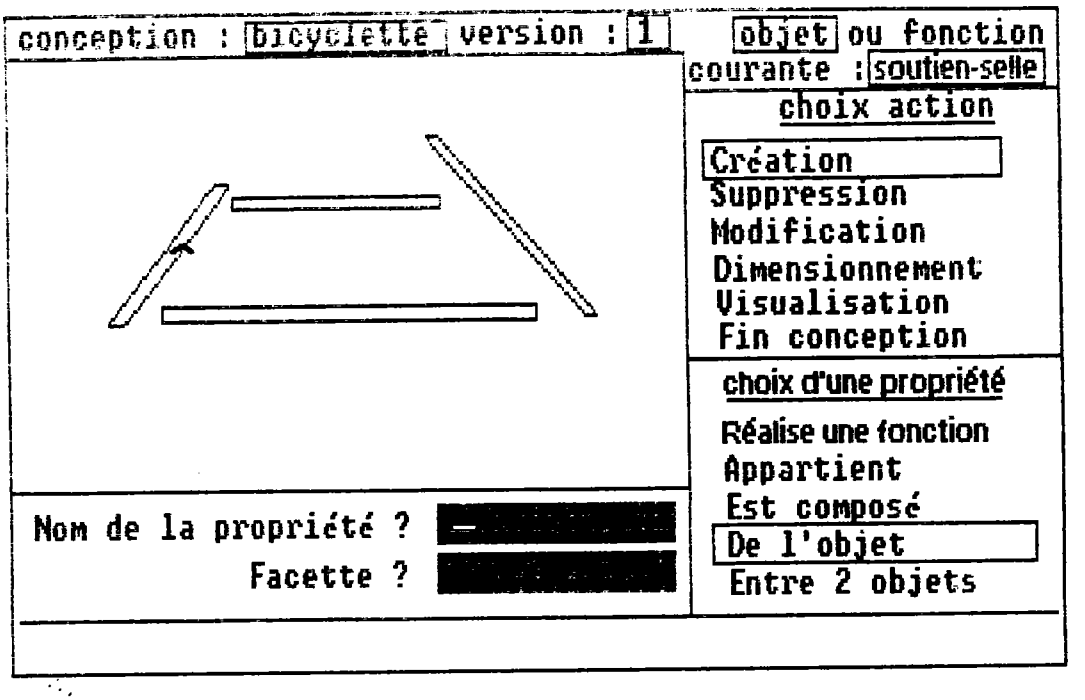
Suite au choix des champs "Création" et "Propriété" l'écran suivant apparaîtra:

E1--->



Les différents types de propriétés apparaissant dans la sous-fenêtre "choix d'une propriété" correspondent aux différents types d'attributs définis dans le modèle de données.
Le choix de la zone "De l'objet" implique l'apparition de l'écran suivant :

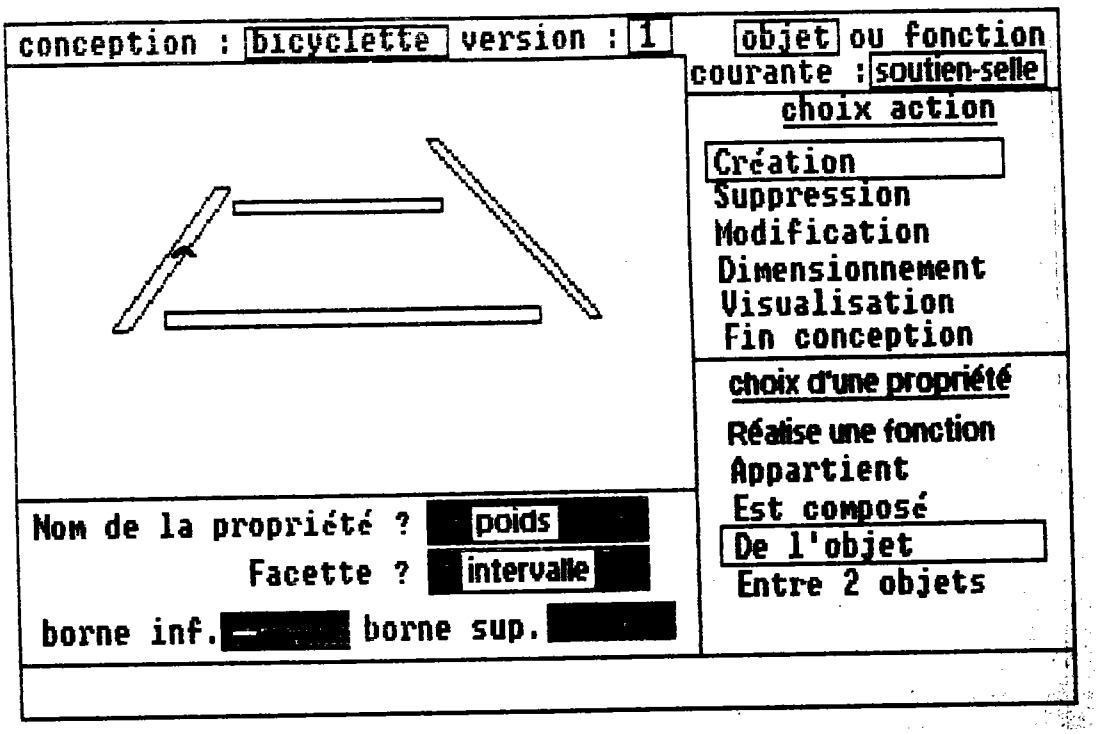
E2----->



Tout d'abord le système demandera le nom de la propriété : l'opérateur répondra en tapant "poids". Puis le système demandera de définir cette propriété (l'utilisateur a la possibilité grâce à une fonction "Aide" de consulter la liste des facettes utilisables).

L'opérateur répondra en donnant le nom de la facette qui est "intervalle".

Suite à ce dialogue l'écran suivant apparaîtra:

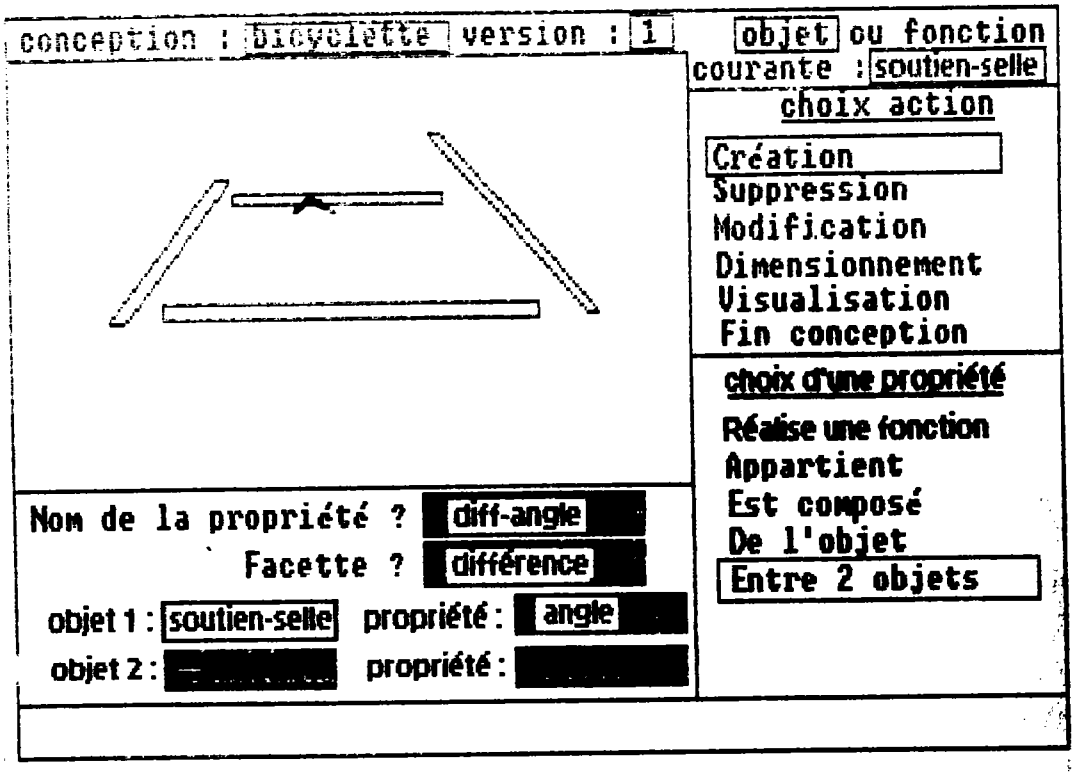


Cet écran permet de définir les bornes de l'intervalle. Grâce à la fonction "validation" (fenêtre fonction) l'opérateur confirmera la création de la facette. Le système proposera alors la création d'une autre facette ou d'une autre propriété.

b) Propriété "diff-angle" :

L'opérateur veut créer une propriété calculée pour les objets barre et soutien-selle qui calculera automatiquement la différences entre les angles entre ces deux objets: on supposera que les propriétés "angle" de ces 2 objets on déjà été définies. On supposera aussi que l'objet "soutien-selle" est l'objet courant. Pour créer cette propriété le processus est identique à celui de la création de la propriété précedente à la différence qu'au niveau des écrans il faut valider "Entre 2 objets" pour le type de propriété (se reporter à E1). Ce choix implique l'affichage d'un écran identique à celui dans lequel il est demandé le nom de la propriété et celui de sa première facette (E2). L'opérateur répondra en saisissant respectivement "diff-angle" et "différence".

Le système demandera ensuite les noms des deux objets et de leurs propriétés respectives paramètres de la facette différence : l'opérateur répondra d'abord en validant l'objet pointé par la souris (c'est-à-dire "soutien-selle") puis en donnant le nom de sa propriété concernée c'est-à-dire "angle". Puis il va pointer avec la souris l'objet "barre". L'écran sera à ce moment-là le suivant:



Il suffira ensuite à l'opérateur de valider l'objet pointé (c'est-à-dire "barre") et de définir la propriété associée (c'est-à-dire "angle"). Le choix de la propriété peut se faire directement en saisissant son nom ou en demandant un affichage de l'ensemble des propriétés de l'objet concerné et en validant la propriété recherchée grâce à la souris.

II.3.4) Analyse de cette interface :

Dans cette deuxième proposition d'interface l'accent a été mis sur la convivialité. Sa description a été partielle (seul le dialogue correspondant à certains traitements a été présenté), toutefois son objectif était d'être aussi proche que possible de la conception mécanique classique.

Nous allons énumérer certains éléments intervenant dans le dialogue qui n'ont pas été présentés içi. On pourra par exemple citer:

- l'utilisation d'aides (en particulier dans la description des procédures de calcul).

- l'auto-apprentissage.

II.3.4) Analyse de cette interface :

Dans cette deuxième proposition d'interface l'accent a été mis sur la convivialité. Sa description a été partielle (seul le dialogue correspondant à certains traitements a été présenté), toutefois son objectif était d'être aussi proche que possible de la conception mécanique classique.

Nous allons énumérer certains éléments intervenant dans le dialogue qui n'ont pas été présentés içi. On pourra par exemple citer:

- l'utilisation d'aides (en particulier dans la description des procédures de calcul).

- l'auto-apprentissage.

III) Les Interfaces et SACADO :

III.1) Synthèse de ces interfaces :

Les deux propositions d'interfaces qui viennent d'être présentées bien qu'étant incomplètes (par rapport à l'ensemble des traitements à prendre en compte), impliquent chacune un type d'application dialogue :

- Pour la première interface, le dialogue est proche de celui associé aux SGBD relationnels sur micros (par exemple DBASE III+ ou 4ème DIMENSION) destinés à des utilisateurs non-informaticiens.

- Dans le cas de la deuxième interface, le dialogue simule une conception en mécanique classique (c'est-à-dire avec comme seuls outils un crayon, une gomme et une calculette et aussi parfois des programmes isolés sur micros pour certains calculs spécifiques).

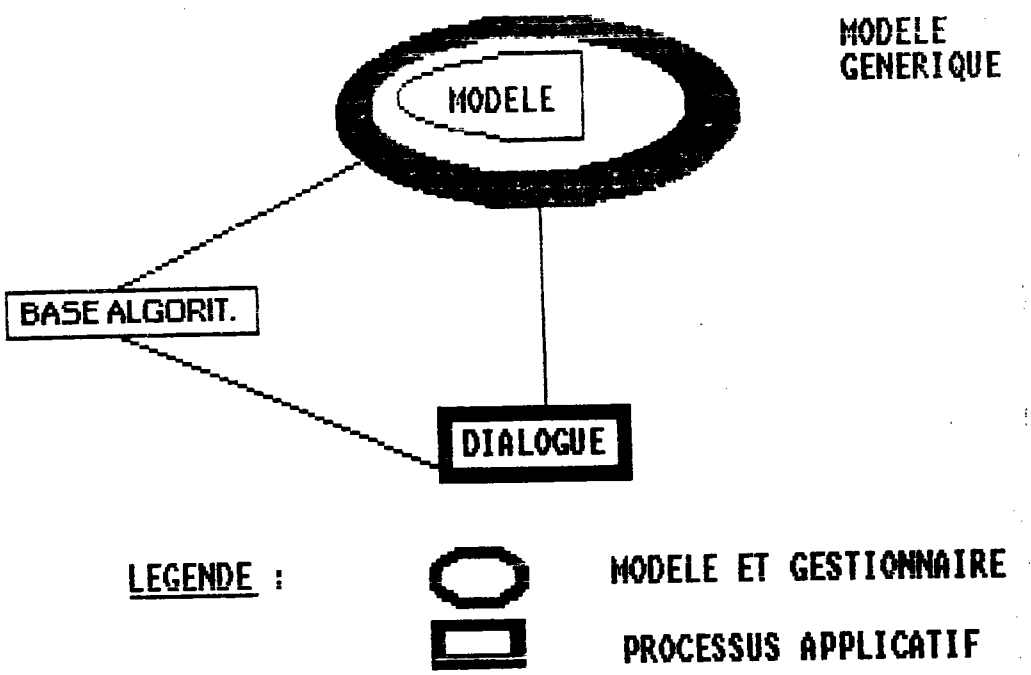
Ces interfaces ont été conçues à partir d'un certain nombre de principes du dialogue définis au paragraphe II.1.1 (convivialité, manipulation simultanées d'informations alphanumériques et graphiques, zooms, aides, souris, possibilité de créer ses propres primitives graphiques).

En conclusion ces principes du dialogue décrits dans le cadre de ces interfaces sont communs à ceux formulés projet SACADO. Parmi le deux interfaces que nous venons de décrire, la deuxième semble la plus proche des concepts du dialogue bien que certains d'entre eux ne soient que partiellement vérifiés : par exemple les outils d'aide devant permettre la description d'une procédure, sont trés limités.

III.2) Les deux propositions d'interfaces par rapport à SACADO :

III.2.1) Première interface :

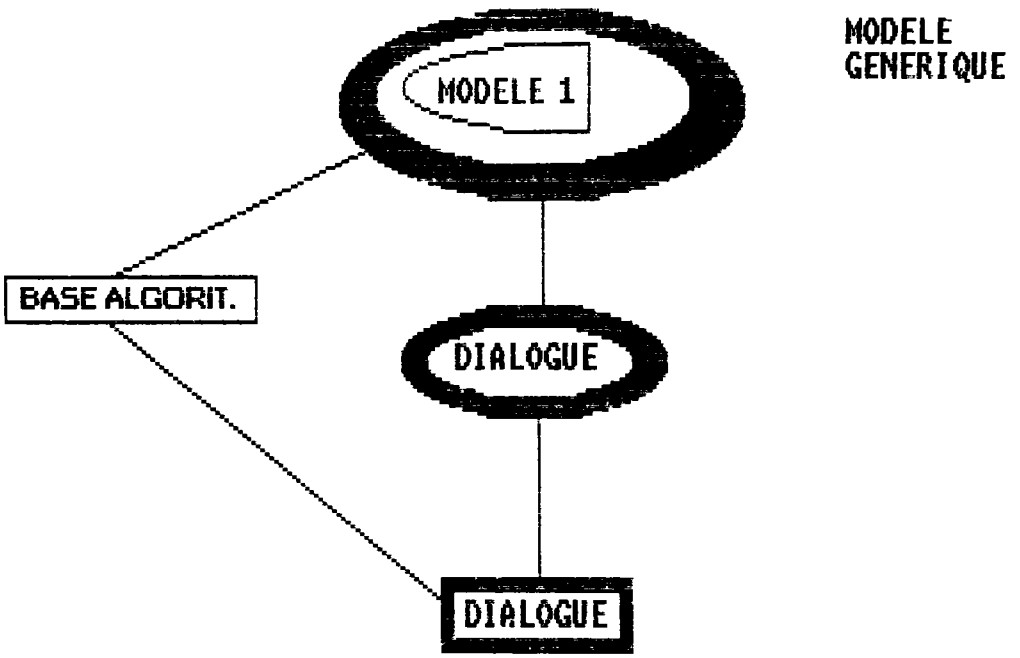
L'application dialogue déduite de la première interface peut être décrite par rapport à celle de SACADO dans le schéma qui suit :



Pour cette application dialogue, il est inutile d'utiliser un processus de localisation-globalisation : les informations du MODELE sont directement extraites sans nécessiter de mise en forme particulière.

III.2.2) Deuxième interface :

Le dialogue déduit de cette deuxième proposition d'interface peut être représenté de la façon suivante :



MODELE 1 par rapport à MODELE n'a qu'une seule différence: dans son modèle de données par rapport à celui de MODELE on a rajouté un champ dans la structure de données associée à tout objet, permettant d'associer à tout objet sa primitive graphique.

Le modèle de dialogue est constitué par:

- le modèle de données identique à celui de MODELE.
- des algorithmes correspondant à différents types d'opérations (affichage, effacement, zoom, désignation, déplacement, etc ...) ou permettant de représenter des primitives graphiques.

Il est inutile d'utiliser ici un processus de localisation/globalisation car les modèles de données sont identiques dans MODELE 1 et dans DIALOGUE.

Pour chaque objet de conception il est possible par le biais d'un autre modèle de dialogue d'avoir sa représentation géométrique "exacte" (obtenu par l'outil de DAO). A un même objet seront donc associeés simultanément deux repreprésentations géométriques complémentaires : on peut considérer que la primitive graphique est le "brouillon" ou encore une vue générale de la représentation "exacte".

Ces remarques confirment le fait que le dialogue (issu de la deuxième proposition d'interface) et le Modèle que nous avons définis dans le cadre d'une réalisation complète pourraient être des outils utilisables dans les bureaux d'études en mécanique. Etant donné qu'ils permettent d'intégrer et de gèrer de les informations non gèométriques d'un processus de conception ils semblent complémentaires des programmes de DAO.

CONCLUSION

L'objectif de notre étude était de montrer comment à partir d'un modèle de données ne possédant qu'un nombre limité de concepts on peut construire un système de gestion de données pour la CAO.

Le modèle de données que nous avons défini est un modèle orienté objet permettant de gèrer les informations suivantes: des objets, des ensembles, des contraintes et des fonctions de calcul. L'aspect modélisation géométrique étant à ce jour en partie résolu, n'a pas été pris en compte. Le système BD-CAO construit autour de ce modèle de données offre les fonctionnalités qui suivent:

- définition et manipulation simultanées des données.
- conception ascendante, descendante, imbriquée.
- héritage et déduction.
- gestion d'objets incomplets et incohérents.

Les informations modélisées appartiennent toutes à la Base de Données Projet: toutefois le modèle de données semble adapté à la prise en compte des catalogues ainsi que des projets antérieurs.

Au niveau de la réalisation, nous disposons actuellement d'une maquette de ce système BD-CAO. Seule la gestion des prototypes est actuellement implantée; de plus les mécanismes d'attachement et de déclenchement procéduraux ainsi que le dialogue (décrit dans la Partie 4) n'y ont pas encore été intégrés.

Nous avons aussi montré l'adéquation du Modèle au processus de conception par le biais d'exemples de conception en mécanique: ainsi le Modèle, dans le cadre d'une réalisation plus complète pourrait être considéré comme un outil de conception complémentaire d'un système de CAO. Afin d'automatiser un maximum de tâches (jusque là manuelles) du processus de conception, il faudrait enrichir le système BD-CAO des composantes suivantes:

- des interfaces avec des systèmes experts, les catalogues et les projets antérieurs.
- des outils de gestion de versions (permettant en particulier d'intégrer un objet conçu dans un catalogue).
- des outils liés au domaine de conception: par exemple un générateur et un interpréteur de schémas cinématiques.

Actuellement les deux derniers points semblent assez faciles à résoudre. En ce qui concerne le premier point, la recherche de pièces dans un catalogue ou dans une base des projets antérieurs est un problème proche des fonctionnalités offertes par les langages de requêtes des SGBD. Par contre la conception d'une interface avec un système expert est un problème à notre connaissance non encore résolu à ce jour.

ANNEXE 1 : LA MAQUETTE

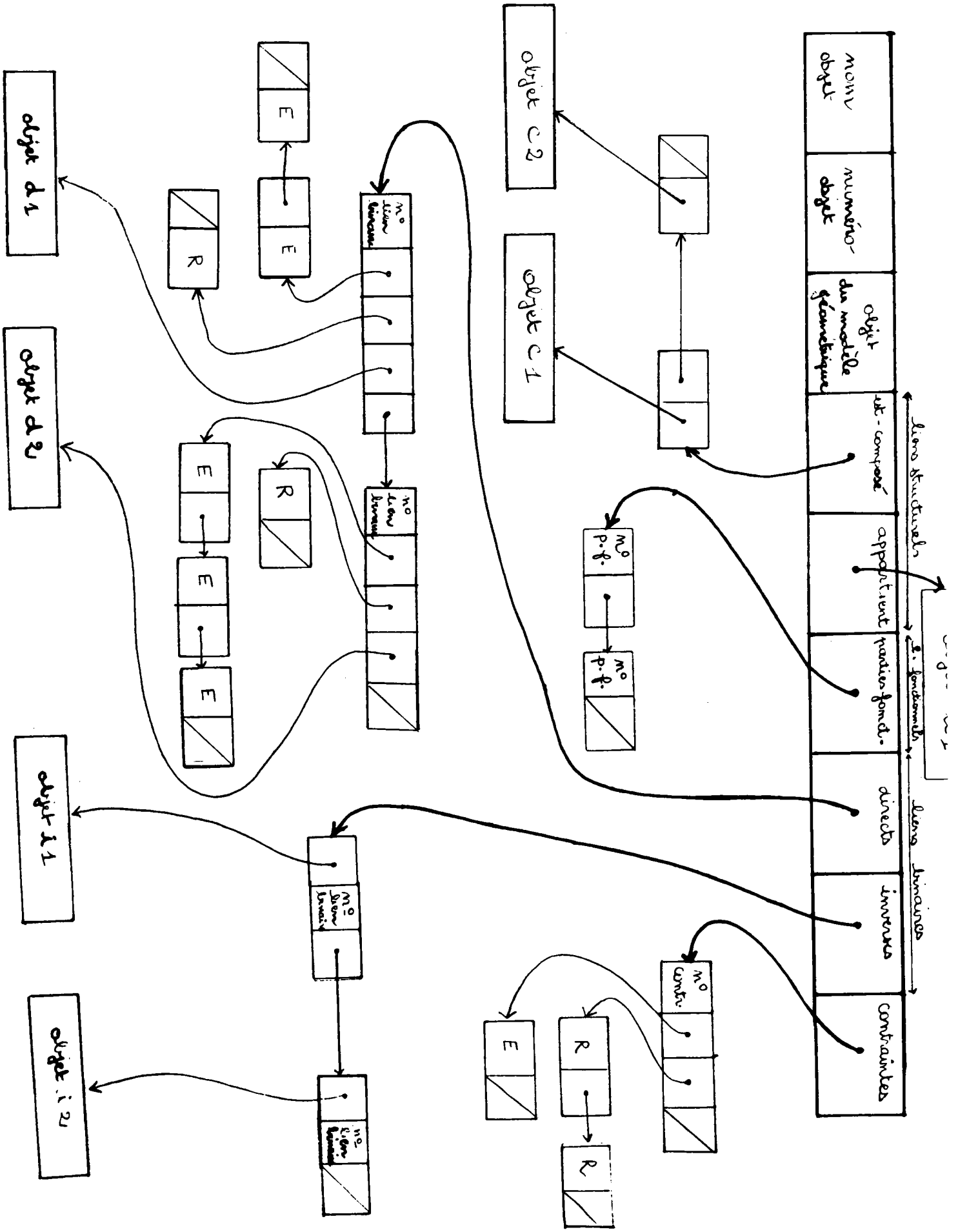
I) Objectifs de cette maquette : [MIC-87]

L'objectif de cette maquette est de créer un gestionnaire d'une structure de données dynamique utilisée en CAO. La modèle (ou objet final) au sens CAO est défini par un arbre d'objets : ces objets sont reliés entre eux par des liens du type père-fils qui permettent de définir une structure hiérarchique. D'autre part tout objet est défini par ses propriétés:
son nom, son numéro et des contraintes dites intra-objet (chaque contrainte intra-objet est définie par un numéro de procédure et un certain nombre de paramètres entiers et réels).

On supposera aussi que tout objet peut appartenir à une ou plusieurs parties fonctionnelles (une partie fonctionnelle est définie par un numéro de partie fonctionnelle).

Enfin il peut exister entre 2 objets des contraintes inter-objets appelées encore liens-linaires. Un lien-linaire est défini par un numéro de procédure et un certain nombre de paramètres entiers et réels.
Les traitements qui seront effectués sur cette structure de données sont classiques c'est à dire création, modification, suppression d'objets.

II) Description de la structure de donnée utilisée :
(pour les déclarations se reporter IV.3)



Annexe 1: 2

III) Description des traitements :
(se reporter aussi au paragraphe IV.4)

Les différents traitements sont les suivants:

-création des parties fonctionnelles et des procédures.
-création d'un objet.
-modification (ou suppression) d'un objet ou d'un lien.
-affichage.

III.1) Création des parties fonctionnelles et procédures :

Tout objet de la conception peut appartenir à une ou plusieurs parties fonctionnelles. De même tout objet a des propriétés qui lui sont propres (contraintes intra-objet) ou peut être lié à un autre objet (lien binaire). Avant de pouvoir intégrer un objet dans une partie fonctionnelle ou de lui associer une procédure (contrainte intra-objet ou lien linéaire) il faut que celles-ci aient été définies par le concepteur. C'est la raison pour laquelle on crée 3 tables :

. la première contiendra l'ensemble des numéros de parties fonctionnelles : elle s'appelle "tabpf".

. la deuxième l'ensemble des numéros de procédures correspondant à des liens binaires, ainsi que le nombre de leurs paramètres entiers et réels : cette table s'appelle "tablienb".

. la troisième à la même structure que la deuxième; elle correspond à l'ensemble des contraintes intra-objets et s'appelle "tabcontrintra".

III.2) Création d'un objet :

La phase de création d'un objet doit permettre de rajouter un nouvel objet à l'ensemble des objets déjà crées. Ce nouvel objet sera défini par :

. son nom.
. son numéro (qui est unique et sera calculé automatiquement).
. un accès à l'objet du modèle géométrique associé.
. ses liens structurels ("appartient" et "est-composé") qui permettent de créer une hiérarchie structurelle entre objets.
. les parties fonctionnelles auquel il appartient.
. ses liens binaires.
. ses contraintes intra-objet.

Il doit être possible de pouvoir créer des objets incomplets (par exemple on pourra ne définir que le nom de l'objet et une contrainte intra-objet), des propriétés pourront être ajoutées ou supprimées lors de la phase de modification.

La phase de création d'un objet doit permettre de rajouter un nouvel objet à l'ensemble des objets déjà crées. Ce nouvel objet sera défini par :

III.3) Modification ou suppression d'un objet ou d'un lien :

* Suppression d'un objet :

Ce traitement revient à supprimer un objet du modèle (son numéro sera realloué) ainsi que tous les liens qu'il avait avec d'autres objets (liens structurels et liens binaires).

* Modification d'un objet :

Par rapport à un objet donné, ce traitement se fera par le biais de :
. l'ajout ou la suppression d'un lien structurel.
. l'ajout ou la suppression de cet objet par rapport à une partie fonctionnelle.
. l'ajout, la suppression ou la modification d'un lien binaire (la phase de modification d'un lien binaire revient à modifier les paramètres de ce lien).
. l'ajout, la suppression ou la modification d'une contrainte intra-objet (même remarque que précédemment pour la phase de modification).

III.4) Affichage :

Ce module permet d'afficher :

. la table initiale des objets créés "tabinit ".
. tout objet avec l'ensemble de ses propriétés.
. l'ensemble des parties fonctionnelles et procédures ayant été crées jusque là.

IV) Procédures essentielles :

IV.1) Procédures générales :

Ces procédures concernent les traitements essentiels que l'on effectue en général sur les listes .
Il s'agit de :

- création.
- lecture.
- recherche d'un élément.
- insertion d'un élément .
- suppression d'un élément.

Un deuxième type de procédures générales permet de faciliter l'utilisation des nombreux menus et grilles de saisies utilisés dans cette maquette.

IV.2) Procédures de contrôle :

IV.2.1) Contrôle d'hiérarchie :

Ce contrôle d'hiérarchie se fait dans les 2 cas suivants :

- lors de la création d'un objet, quand on veut désigner pour cet objet, un objet père et des objets fils .
- lors de la création d'un lien structurel entre 2 objets.

Donc quand on veut créer un lien structurel entre 2 objets , il faut pouvoir déterminer si ce lien est possible ou non. Pour celà, on utilisera les principes suivants:

- s'ils font partie du même sous-arbre alors le lien sera impossible.
- s'ils appartiennent à 2 sous-arbres différents, alors il faudra que l'objet fils soit racine de son sous-arbre.

Pour le contrôle de la hiérarchie, on utilisera 2 fonctions :

. "numéroracine" : qui pour 1 objet quelconque, (appartenant donc à un sous-arbre) retournera le numéro de l'objet racine du sous-arbre .

. "hiérarchie" : dont les paramètres sont les adresses d'un objet père et d'un objet fils que l'on veut relier par un lien structurel. Cette fonction retournera un booléen.

IV.2.2) Contrôle d'existence :

."existtabpf" : fonction qui permet de savoir si un numéro de partie fonctionnelle a déjà été crée dans la table des parties fonctionnelles "tabpf".

."existtabproc" : fonction dont le rôle est identique à celui de la précédente. Ici il s'agit de procédures (liens binaires ou contraintes intra-objet) et les tables concernées sont : "tablien" et "tabcontintra".

."existobj" : procédure d'existence d'un objet identifié par son nom. L'ensemble des objets crées ainsi que leurs adresses se trouve dans "tabobjet". Si cet objet existe déjà , cette procédure permettra de passer son adresse par un des paramètres.

."existlienstruct" : fonction de test d'existence d'un lien structurel entre un objet père et un objet fils. Cette fonction retournera un booléen.

."existlienbindir" : procédure effectuant le test d'existence d'un lien binaire direct donné (identifié par son numéro de lien) entre 2 objets : on parlera de liens binaires directs entre 2 objets a et b, si le numéro de l'objet a est supérieur à celui de b, dans le cas contraire, on parlera de liens binaires inverses.
S'il existe un lien binaire entre 2 objets, la procédure passera l'adresse du maillon correspondant à ce lien binaire (se reporter au IV.2) par le biais de l'un de ses paramètres.

."existcontrintra" : procédure faisant le contrôle d'existence
d'une contrainte intra-objet donnée pour un objet. De même que la procédure précédente, si cette contrainte existe l'adresse du maillon correspondant (se reporter au schéma du paragraphe II.) sera fournie dans un des paramètres de la procédure.

V.3) Procédures de création :

."creatabpf" : procédure d'ajout d'une partie fonctionnelle dans la table des parties fonctionnelles "tabpf" avec test de table saturée.

."creattabproc" : procédure permettant l'ajout d'une procédure (c'est à dire un lien binaire ou une contrainte intra-objet) ainsi que le mombre de ses paramètres entiers et réels dans la table des procédures (c'est à dire "tablienb" ou " tabcontrintra"). Ici encore il y aura un test de table saturée.

."creatnomstruct" : procédure de création d'un objet
s'occupant d'abord de le créer dans la table des objets crées jusque là "tabobjet" (il sera défini dans cette table par un nom, un numéro qui lui sera automatiquement associé et une adresse). Ensuite l'objet est crée dans la structure dynamique à une certaine adresse qui sera mémorisée dans la table : les seules propriétés qu'on lui associera seront son nom et son numéro, les autres correspondant à des pointeurs (voir schéma du paragraphe II.) seront mis à nil.

."creatlienstruct" :(se reporter au schéma du paragraphe V.1) procédure de création d'un lien structurel entre un objet père et un objet fils.(dans la structure dynamique): cela revient en fait à créer 2 liens inverses (chaînage entre les 2 objets).

."creatlienfonct" : rajout d'un objet donné dans une partie fonctionnelle définie par son numéro (revient à rajouter un élément dans la liste des parties fonctionnelles auquel appartient cet objet).

."creatlienbin" : (se reporter au schéma du paragraphe V.2) procédure de création de liens binaires entre 2 objets. Un lien binaire est identifié par un numéro de procédure ainsi

que par une liste de paramètres entiers et une liste de paramètres réels. "creatlienbin" effectue la création dans la struture dynamique du lien binaire direct ainsi que du lien binaire inverse (ici encore les 2 objets seront chaînés).

."creatcontrintra" : Cette procédure ajoute une nouvelle contrainte à un objet donné : cette contrainte intra-objet est définie par un numéro de procédure et par une liste de paramètres entiers et une liste de paramètres réels.

IV.4) Procédures de suppression :

Leur principe est exactement l'inverse de celui des procédures de création, c'est la raison pour laquelles nous ne feront que les citer.

"supprnomtable" : suppression d'un objet dans la table "tabobjet " des objets de la conception.

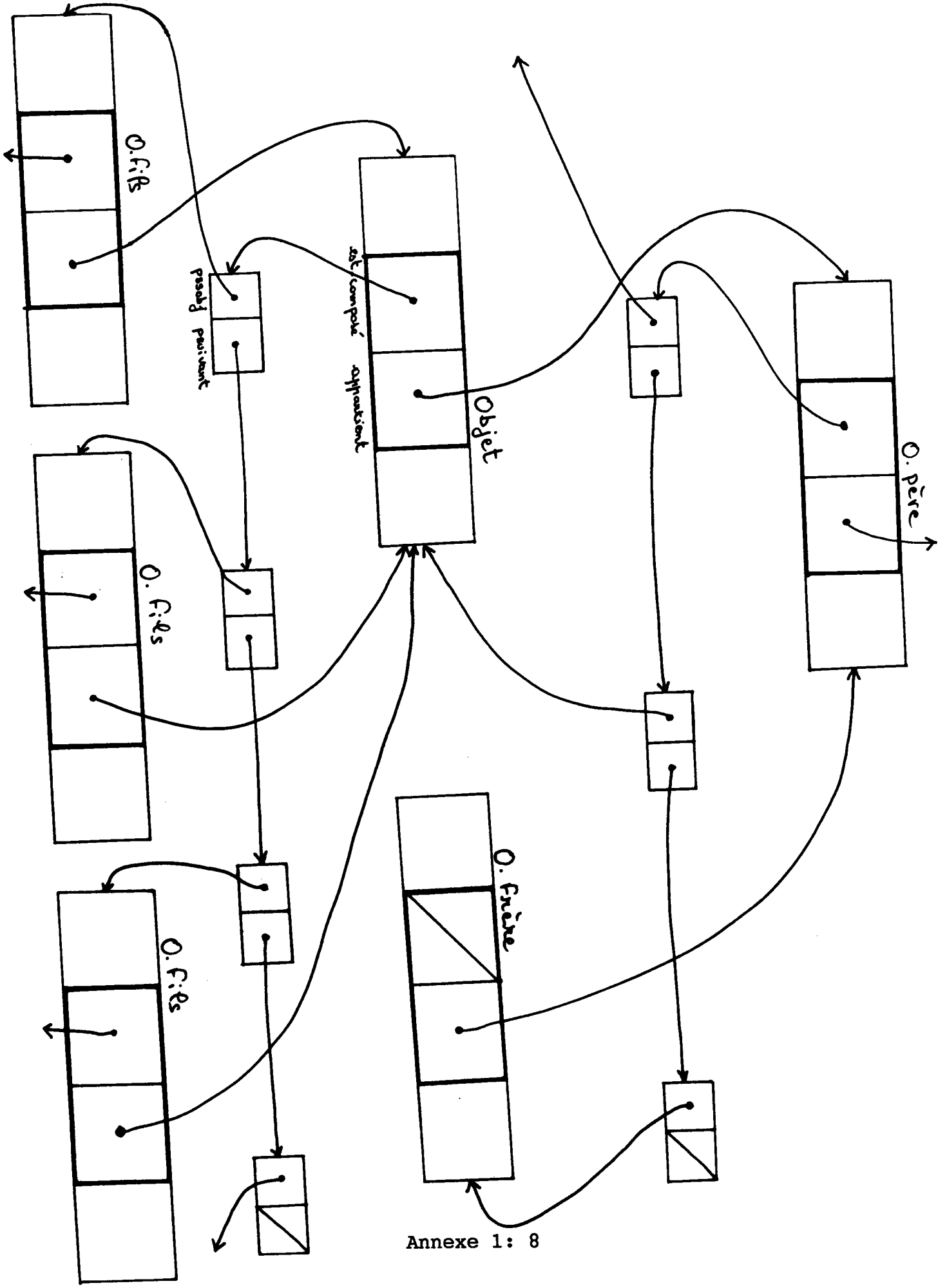
"slienstruct" : suppression des liens structurels entre 2 objets (liens père-fils et fils-père).

"supprlienbindir" (respectivement "supprlienbininv"): suppression d'un lien binaire direct (resp. inverse) défini par un numéro et les 2 objets concernés par ce lien.

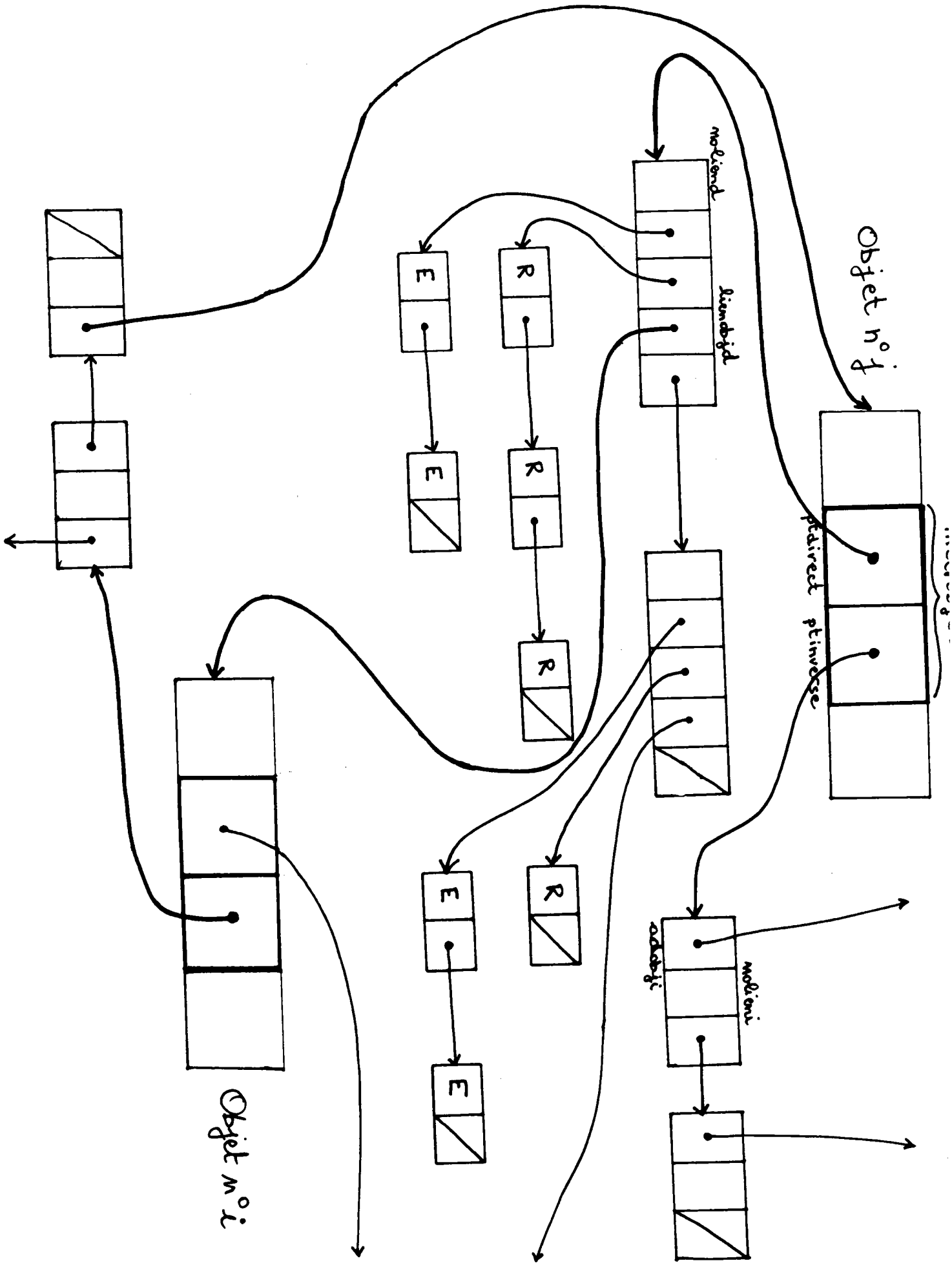
"scontrintra" : suppression d'une contrainte intra-objet (définie par son numéro) concernant un objet donné.

V) Eléments de la maquette :

V.1) Liens structurels :



V.2) Liens binaires :



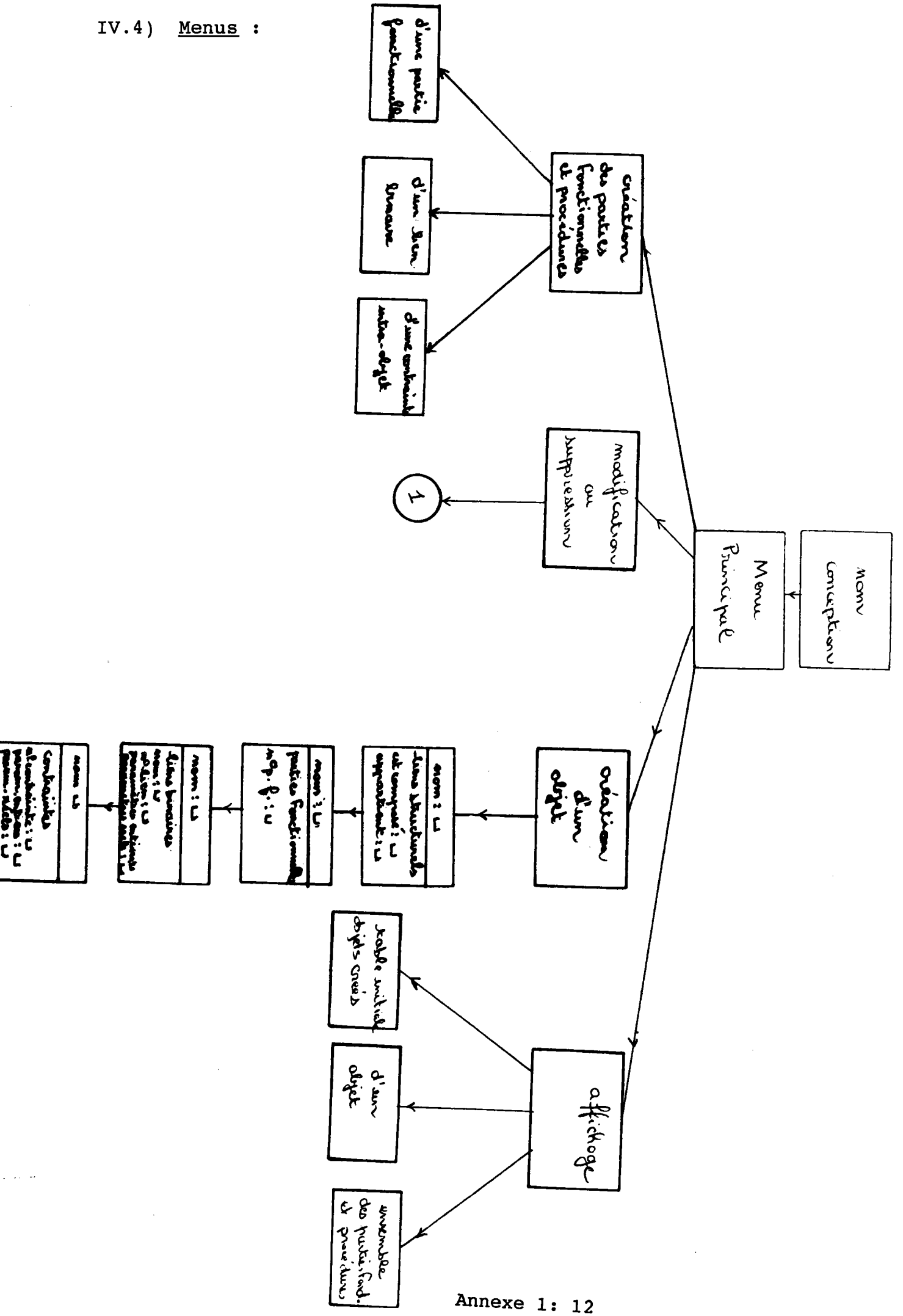
V.3) Déclarations :

```
program maquette1;
const
 nbobjet=10;
 longcrainte=10;
 longnomconc=4;
 nbc=3;
 nbproc=3;

type
 crainte=string[longcrainte];
 nomconc=string[longnomconc];
 occupation=(libre,occupe);
 etattable=(nolibre,nolibre);
 pt_parment=^type_listent;
 pt_parmreel=^type_listreel;
 pt_objet=^type_objet;
 pt_lienbind=^type_lienbind;
 pt_lienbini=^type_lienbini;
 pt_contr=^type_contrainte;
 pt_sousobj=^type_sousobj;

type_listent=record
             parment:integer;
             pte:pt_parment;
             end;

type_listreel=record
              parmreel:real;
              ptr:pt_parmreel;
              end;

type_lienbind=record
              noliend:integer;
              parliene:pt_parment;
              parlienr:pt_parmreel;
              lienobjd:pt_objet;
              liensuivantd:pt_lienbind;
              end;

type_lienbini=record
              nolieni:integer;
              adobji:pt_objet;
              liensuivanti:pt_lienbini;
              end;

type_lienbin=record
             ptdirect:pt_lienbind;
             ptinverse:pt_lienbini;
             end;

type_contrainte=record
                noproc:integer;
                parconte:pt_parment;
                parcontr:pt_parmreel;
                contrsuiv:pt_contr;
                end;
```

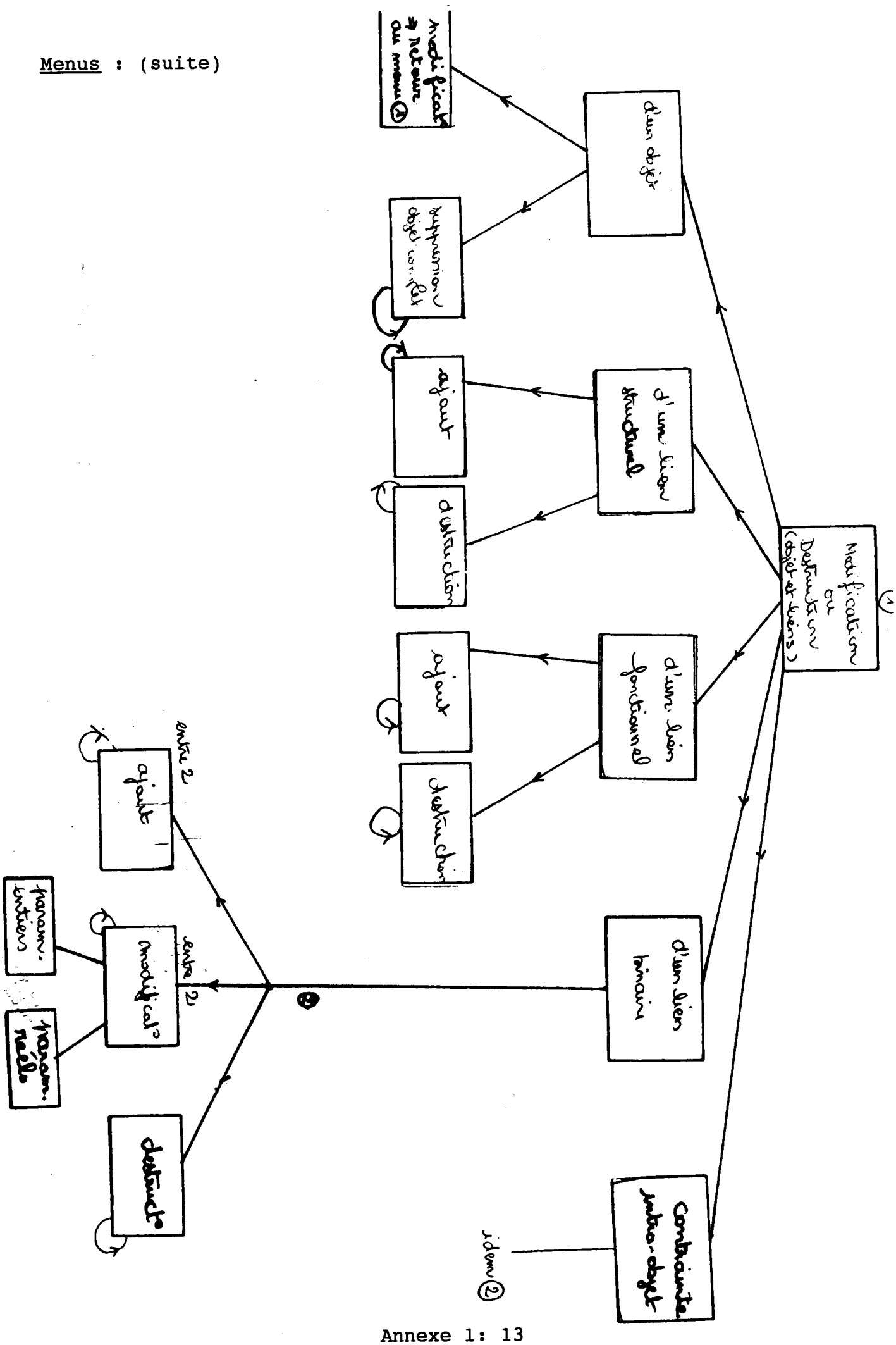
Déclarations : (suite)

```
type_sousobj=record
        (   pssobj:pt_objet;
            psuivant:pt_sousobj;
            end;

type_objet=record
           nom:chaine;
           numero:integer;
           geom:integer;
           estcompose:pt_sousobj;
           appartient:pt_objet;
           partiefonct:pt_parment;
           interobjet:type_lienbin;
           intraobjet:pt_contr;
           end;

type_table=record
           nomobj:chaine;
           adresseac:pt_objet;
           numero:1..nbobjet;
           adressegeom:integer;
           nopere:integer;
           fils:integer;
           nopf:integer;
           lienbin:integer;
           contrainte:integer;
           suivant:integer;
           end;

type_proced=record
            nop:integer;
            nbpe:integer;
            nbpr:integer;
            end;

type_tabproced=array[1..nbproc] of type_proced;


 var
  tabobjet:array[1..nbobjet] of type_table;
  tabpf:array[1..nbpf] of integer;
  tablienb,tabcontrintra:type_tabproced;
  c,p:char;
  i:integer;
  valid:boolean;
  nobj:chaine;
  pto,ptp,ptf:pt_objet;
  tabinit:array[occupation] of integer;
```

IV.4) Menus :


Menu Principal
Menu conception
création des parties fonctionnelles et procédures
d'une partie fonctionnelle
d'un bien
d'une entité inter-objet
modification ou suppression
1
création d'un objet
affichage
d'un objet
ensemble des parties fonct. et procédures

Menus : (suite)


Modification ou Destruction (objet et liens)
(1)
d'un objet
Modification ou Retour au menu
Suppression objet courant
d'un lien fonctionnel
ajout
destruction
d'un lien fonctionnel
ajout
destruction
d'un lien binaire
Contrainte inter-objet
idem
(2)
entre 2
ajoute
entre 2
modifie
param. entier
param. réel
destructe

V.5) Chargement et sauvegarde de la base :

L'option qui a été prise ici est de faire le chargement de la base (s'il existe déjà une version du modèle) en début de phase de conception et de même la sauvegarde se fera à la fin de cette conception. Celà veut dire que au cours d'une session de travail, les données manipulées seront celles de la structure dynamique. Les principaux fichiers utilisés pour sauvegarder la base, ainsi que les liens existant entre-eux sont décrits dans le schéma qui suit. Pour une meilleure compréhension de l'organisation des fichiers, on a simulé dans ce schéma chaque fichier par un tableau ayant la même organisation. (pour des raisons de clarté du schéma, certain tableaux ont toutefois été découpés).

Exemple d'utilisation de ces tables :

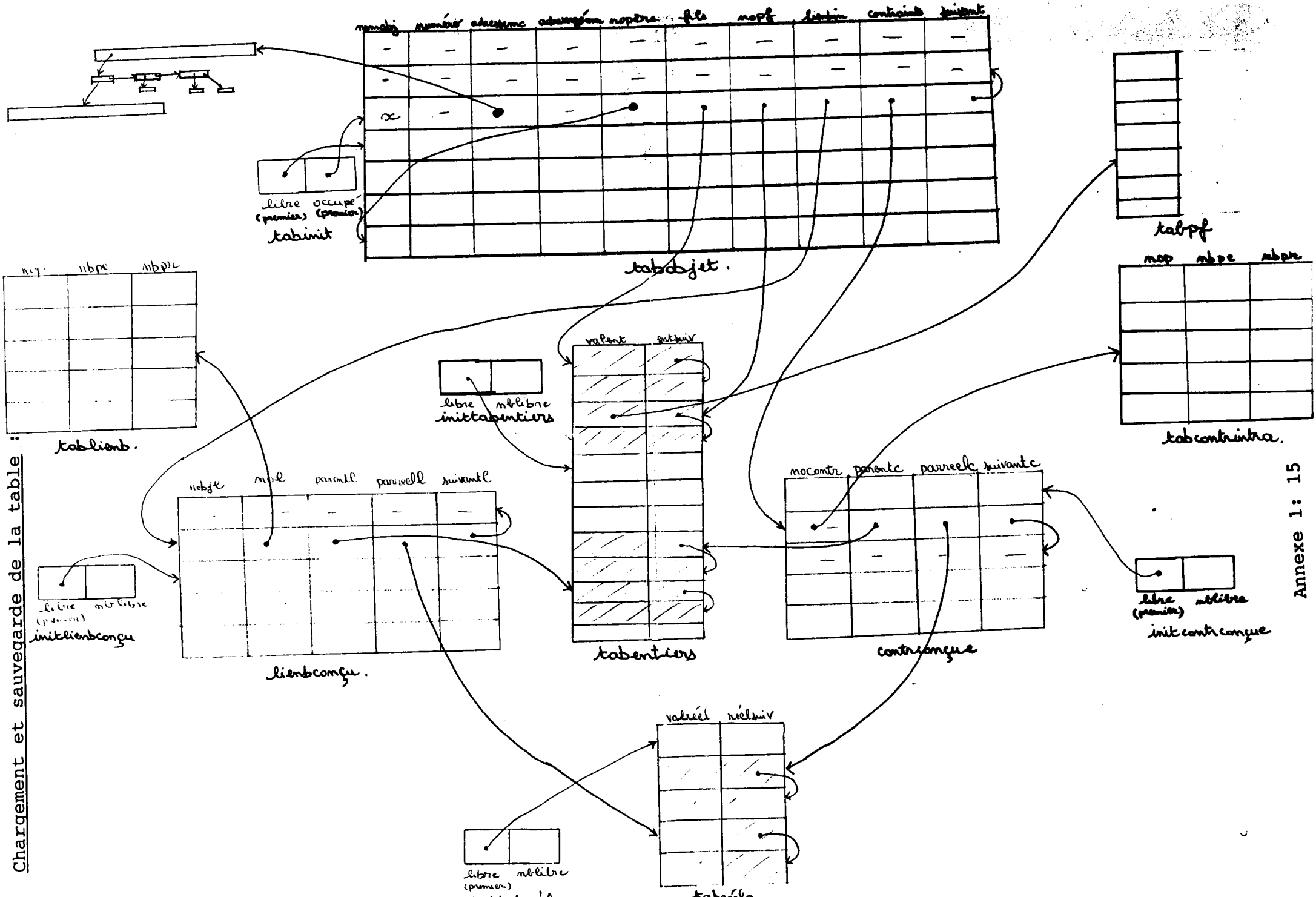
1) Quel est le père de l'objet x?

Pour cela on cherche d'abord dans "tabobjet" l'élément correspondant à cet objet (pour cette recherche on utilisera "tabinit" qui permet une allocation dynamique de "tabobjet"), puis la zone nopere nous donnera le numéro de l'objet père (nopere = 0 si pas de père).

2) Quels sont les liens binaires concernant x?

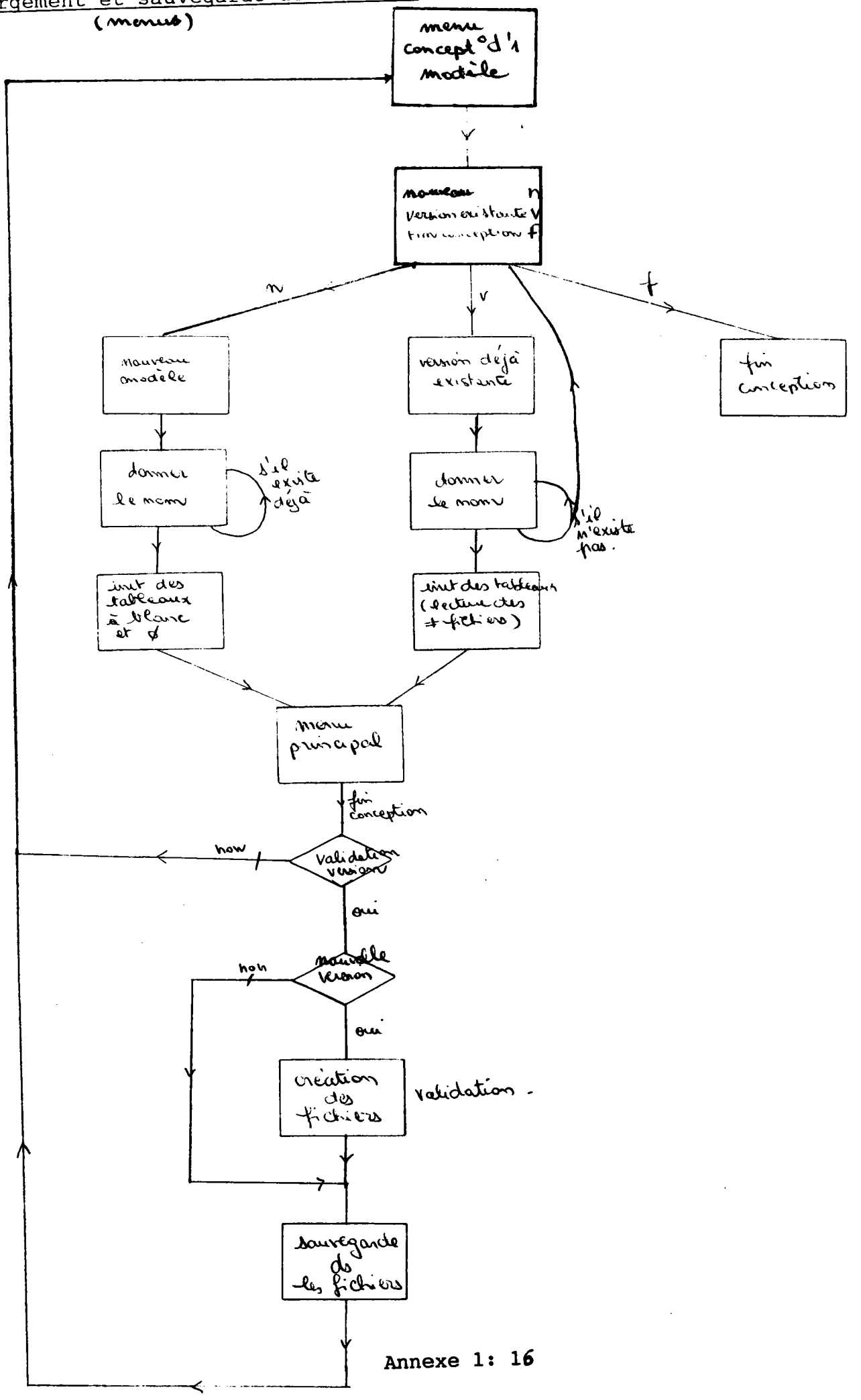
De même que dans la question précédente, on cherche d'abord l'élément de "tabobjet" contenant les informations de base concernant l'objet x. La zone lienbin permet l'accès au premier des liens binaires de x dans "lienbconcu" : nobjl est le numéro de l'autre objet concerné par ce lien, nol est le numéro du lien, parentl (resp. parreell) pointent sur la liste des paramètres entiers (resp. réels) contenus dans "tabentiers" (resp. "tabréels"). Enfin la zone suivante de "lienconcu" permet l'accès au lien binaire suivant concernant l'objet x.

Chargement et sauvegarde de la table :



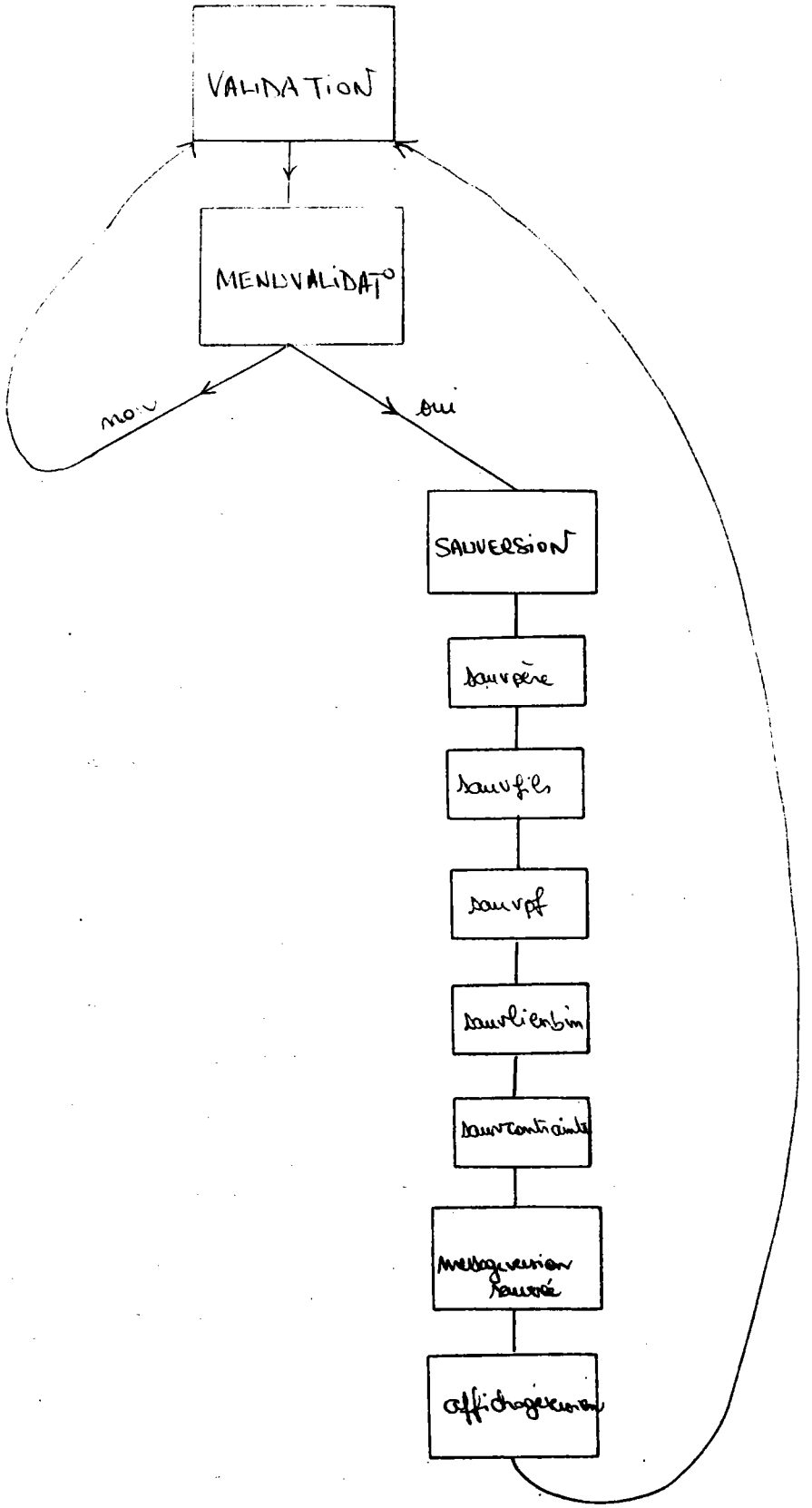
Annexe 1: 15

Chargement et sauvegarde de la table :



Chargement et sauvegarde de la table :

(menus : suite)

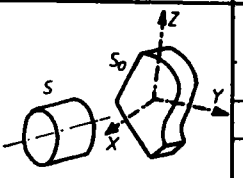
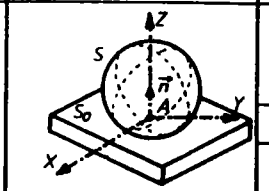
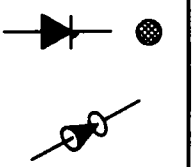
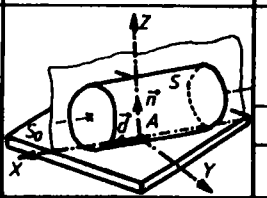
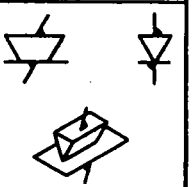
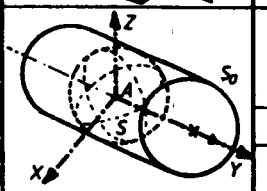
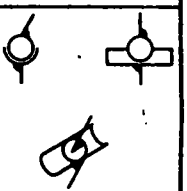
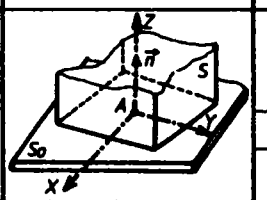
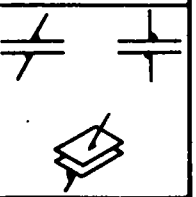
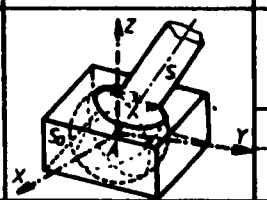
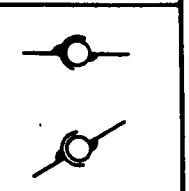
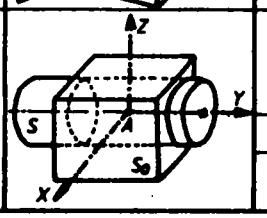
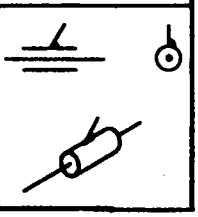


ANNEXE 2 : ELEMENTS DE CONCEPTION MECANIQUE

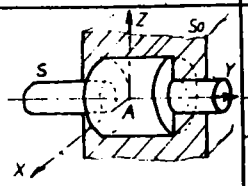
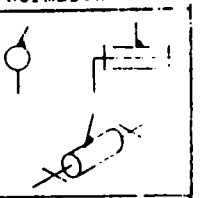
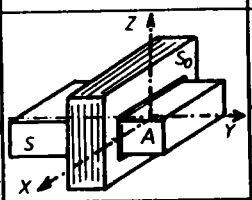
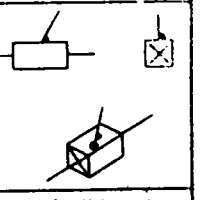
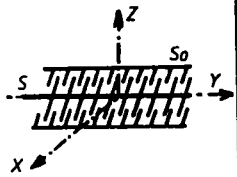
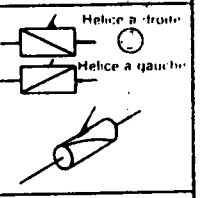
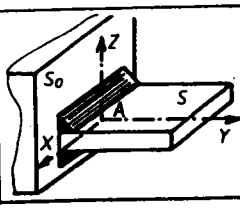
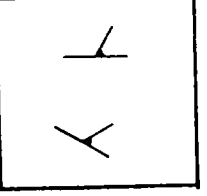
Les éléments de cette annexe ont été tirés de [LTC-87]

I) Liaisons usuelles entres deux solides :

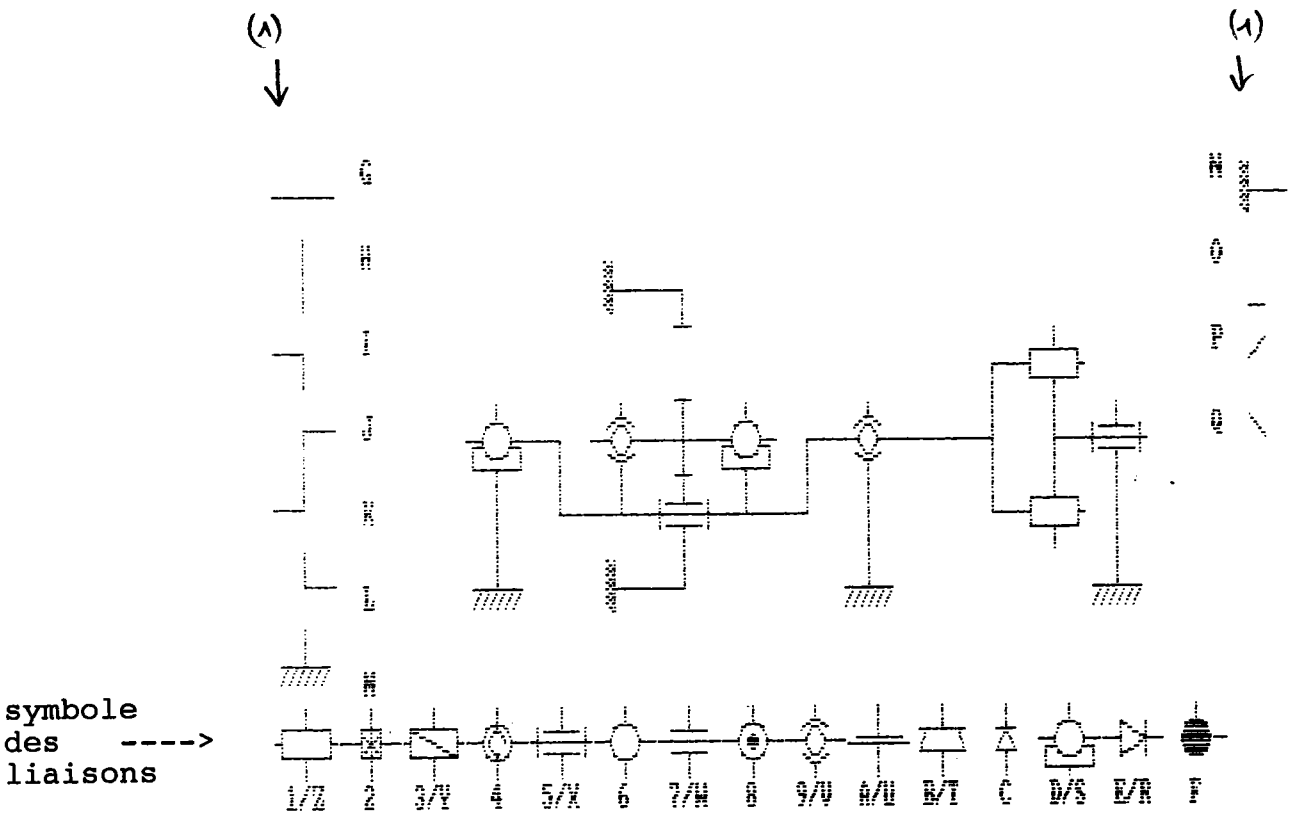
L'étude des assemblages simples entre deux pièces à permis de dénombrer 11 types de liaisons. Ces liaisons et les torseurs associés sont représentés sur les tableau suivant :

Nom de la Liaison		Torseur Transmissible assoc.	Torseur Cinématique associé	Schématisation normalisée
PAS DE LIAISON		$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} \omega_x & V_x \\ \omega_y & V_y \\ \omega_z & V_z \end{Bmatrix}_A$	Représentation plane Représentation en perspective
		D° liaison: 0	d° mobilité: 6	
		Aucun contact entre les pièces		
PONCTUELLE		$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ Z & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} \omega_x & V_x \\ \omega_y & V_y \\ \omega_z & 0 \end{Bmatrix}_A$	
		d° liaison: 1	d° mobilité: 5	
		mêmes particularités en tout point $\in \vec{n}$.		
LINEAIRE RECTILIGNE		$\begin{Bmatrix} 0 & 0 \\ 0 & M \\ Z & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} \omega_x & V_x \\ 0 & V_y \\ \omega_z & 0 \end{Bmatrix}_A$	
		d° liaison: 2	d° mobilité: 4	
		mêmes particularités en tout point du plan normal $(\vec{n}, \vec{d})$		
LINEAIRE ANNULAIRE		$\begin{Bmatrix} X & 0 \\ 0 & 0 \\ Z & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} \omega_x & 0 \\ \omega_y & V_y \\ \omega_z & 0 \end{Bmatrix}_A$	
		d° liaison: 2	d° mobilité: 4	
		particularités vraies qu'au seul point A		
APPUI PLAN		$\begin{Bmatrix} 0 & L \\ 0 & M \\ Z & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} 0 & V_x \\ 0 & V_y \\ \omega_z & 0 \end{Bmatrix}_A$	
		d° liaison: 3	d° mobilité: 3	
		mêmes particularités en tout point $\in \vec{n}$		
ROTULE		$\begin{Bmatrix} X & 0 \\ Y & 0 \\ Z & 0 \end{Bmatrix}_A$	$\begin{Bmatrix} \omega_x & 0 \\ \omega_y & 0 \\ \omega_z & 0 \end{Bmatrix}_A$	
		d° liaison: 3	d° mobilité: 3	
		particularités vraies qu'au seul point A		
PIVOT GLISSANT (verrou)		$\begin{Bmatrix} X & L \\ 0 & 0 \\ Z & N \end{Bmatrix}_A$	$\begin{Bmatrix} 0 & 0 \\ \omega_y & V_y \\ 0 & 0 \end{Bmatrix}_A$	
		d° liaison: 4	d° mobilité: 2	
		même particularités en tout point $\in \vec{Y}$		

Liaisons usuelles entre deux solides : (suite)

Nom de la Liaison		Torseur Transmissible associé	Torseur cinématique associé	Schéma Normalisé
PIVOT (rotoïde)		$\begin{Bmatrix} X & L \\ Y & O \\ Z & N \end{Bmatrix}_A$	$\begin{Bmatrix} O & 0 \\ \omega y & 0 \\ O & 0 \end{Bmatrix}_A$	
		d° liaison: 5	d° mobilité: 1	
		Mêmes particularités en tout point $\in$ à Y		
GLISSIERE (prismatique)		$\begin{Bmatrix} X & L \\ O & M \\ Z & N \end{Bmatrix}_A$	$\begin{Bmatrix} O & 0 \\ O & Vy \\ O & 0 \end{Bmatrix}_A$	
		d° liaison: 5	d° mobilité: 1	
		Mêmes particularités en tout point $\in$ à Y		
GLISSIERE HELICOIDALE		$\begin{Bmatrix} X & L \\ Y & M \\ Z & N \end{Bmatrix}_A$	$\begin{Bmatrix} O & 0 \\ \omega y & Vy \\ O & 0 \end{Bmatrix}_A$	
		d° liaison: 5	d° mobilité: 1	
		M = k.Y	Vy = K.ωy	
ENCASTREMENT (liaison complète)		$\begin{Bmatrix} X & L \\ Y & M \\ Z & N \end{Bmatrix}_A$	$\begin{Bmatrix} O & 0 \\ O & 0 \\ O & 0 \end{Bmatrix}_A$	
		d° liaison: 6	d° mobilité: 0	

II) Exemple de schéma cinématique :



(1) : symboles des objets.

III) Liste de règles pour la détermination d'un réducteur d'engrenage[5]:

Ces règles s'appliquent à l'exemple de la pompe doseuse (Partie) et permettent de déterminer l'engrenement (se référer au paragraphe II.3.2.1).

Voici quatre exemples de ces règles :

REGLE N° 1 :

SI EST CONNU LE RAPPORT DE REDUCTION I
ET SI I EST INFERIEUR A 7
ET SI N'EST PAS CONNUE LA POSITION DE L'AXE D'ENTREE PAR RAPPORT A L'AXE DE SORTIE
ET SI LE CRITERE DE CHOIX EST LE COUT

ALORS LA REDUCTION SE FERA AVEC UN ENGRENAGE CYLINDRIQUE A AXES PARALLELES A DENTURE EXTERIEURE

REGLE N° 2 :

SI EST CONNU LE RAPPORT DE REDUCTION I
ET SI I EST INFERIEUR A 7
ET SI L'AXE D'ENTREE DOIT ETRE PARALLELE A L'AXE DE SORTIE
ET SI N'EST PAS CONNUE LA DISTANCE ENTRE L'AXE D'ENTREE ET L'AXE DE SORTIE
ET SI LE CRITERE DE CHOIX EST LE COUT

ALORS LA REDUCTION SE FERA AVEC UN ENGRENAGE CYLINDRIQUE A AXES PARALLELES A DENTURE EXTERIEURE

REGLE N° 3 :

SI EST CONNU LE RAPPORT DE REDUCTION I
ET SI I N'EST PAS INFERIEUR A 7
ET SI IL EST INFERIEUR A 40
ET SI N'EST PAS CONNUE LA DISTANCE ENTRE L'AXE D'ENTREE ET L'AXE DE SORTIE
ET SI LE CRITERE DE CHOIX EST LE COUT

ALORS LA REDUCTION SE FERA AVEC DEUX ENGRENAGES CYLINDRIQUES A AXES PARALLELES A DENTURE EXTERIEURE

REGLE N° 4 :

SI EST CONNU LE RAPPORT DE REDUCTION I
ET SI I N'EST PAS INFERIEUR A 7
ET SI IL EST INFERIEUR A 40
ET SI L'AXE D'ENTREE DOIT ETRE PARALLELE A L'AXE DE SORTIE
ET SI N'EST PAS CONNUE LA DISTANCE ENTRE L'AXE D'ENTREE ET L'AXE DE SORTIE
ET SI LE CRITERE DE CHOIX EST LE COUT

ALORS LA REDUCTION SE FERA AVEC DEUX ENGRENAGES CYLINDRIQUES A AXES PARALLELES A DENTURE EXTERIEURE

BIBLIOGRAPHIE

ADI-82 ADIBA M., DELOBEL M.
"Bases de Données et sytèmes relationnels".
DUNOD.

ADI-83 ADIBA M.
"Bases de Données et CAO".
Rapport BD3 Bases de Données. Nouvelles perspectives.
Janvier 1983.

AUT-82 AUTRAN J., FLORENZANO M.
"Etude exploratoire en vue de la conception et de la réalisation d'un SGBD en CAO en architecture".
GAMSAU. Marseille-Luminy. Mai 1982.

AUT-85 AUTRAN J., FLORENZANO M.
"L'utilisation des SGBD dans le domaine du bâtiment".
Modèles et Bases de Données. N°1. Octobre 1985.

BAT-85 BATORY D.S., KIM W.
"Modeling concepts for VLSI CAD Objects".
ACM-TODS Vol. 10. N°3. Septembre 1985.

BD3-83 "Bases de données et nouvelles perspectives".
Rapport INRIA-ADI. Janvier 1983.

BIL-83 BILLET A.
"Etude de quelques systèmes de gestion de données pour la conception assistée par ordinateur".
BIGRE N°36. Octobre 1983.

BIL-87 BILLET A., CHELGOUM K.
"Un modèle de représentation de connaissances pour la conception assistée par ordinateur".
MICAD 87.

BON-84 BONNET A.
"I.A. : promesses et réalité".
INTER-EDITIONS.

BOU-82 BOUYAT M., VIGNAT J.C.
"Développement d'un système de CAO en Génie Civil appliqué aux voiries et réseaux divers".
Rapport final. Convention ADI-INSA Lyon-CERT.
Septembre 1982.

CAD-85 "Expert Systems".
CAD. Novembre 1985.

CHA-86 CHAUVIN A.M., PUCHERAL P., ZENG Y., NARAT V.
"Les Bases de Données Déductives".
Laboratoires MASI. Janvier 1986.

CHO-81 CHOURAQUI E.
"Contribution à l'étude théorique de la représentation des connaissances : le système expert ARCHES".
Thèse d'état. INPL-1981.

CHO-83a CHOLVY L.
"Structuration et intégrité des données dans les bases de données CAO : définition d'un modèle de données et réalisation d'une maquette".
Thèse de Docteur-ingénieur. ENSAE. Décembre 1983.

CHO-83b CHOLVY L., FOISSEAU J.
"ROSALIE : an object-oriented and rule-based CAD system".
IFIP 83. PARIS. Septembre 1983.

CHR-85 CHRISTMENT C., CRAMPES J.B., ZURFLUH B.
"Bases d'informations généralisées".
Dunod Informatique. 1985.

CHU-83 CHU.
"VDD : a VLSI Design Database System".
Bell Laboratories New Jersey.
ACM Database week. San José. 1983.

CRI-86 COLNET D., MASINI D., NAPOLI A., NOIRET Y., TOMBRE K.
"Les Langages Orientés Objets".
CRIN-1986.

CXP-86 "SGBD relationnels pour moyens et grands sytèmes : analyse et comparaison".
Etude CXP. N° 119. Juillet 1986.

DAR-83 DARONNAT Y.
"Programmation logique d'un modèle de données de conception".
Thèse 3° cycle. Université de Lyon I. Novembre 1983.

DAT-81 DATE J.C.
"An introduction to Database Systems".
(3rd edition). Addison- Wesley. 1981.

DAT-83 DATE J.C.
"An introduction to Database Systems".
Volume II. Addison- Wesley. 1983.

DAT-86 DATE J.C.
"An introduction to Database Systems".
Volume I. Fourth edition. 1986.

DAV-81 DAVID B.
"Methodologie pour la construction de système CAO : SIGMA-CAO".
Thèse d'état. Univ. de Grenoble. Septembre 1981.

DEM-83 DEMOLOMBE R.
"Ergonomie de l'utilisation des SGBD".
Dans BD3-83 .

DEM-86 DEMOLOMBE R.
"Problèmatique des Bases de Données Déductives".
MBD N° 3. Mai 1986

DGM-80 DAVID, GARDAN, MERMET.
"CAD in small and medium sized industries".
IFIP. 1980. Tokyo.

EDM-83 EDMOND, MARECHAL.
"Experience in building ARCADE : a computer aided design system bases on a relationnal DBMS".
ACM database week. San José. 1983.

ENC-82 ENCARNACAO J., KRAUSE F.
"File structures and databases for CAD".
IFIP. North-Holland. 1982.

FAU-87 FAUVET M.C., RIEU D.
"CADB : un système de gestion de bases de données et de connaissances pour la CAO".
MICAD 87.

FIS-79 FISCHER W.E.
"PHIDAS : a database management system for CAD/CAM applications".
CAD. Vol. 11. N°3. 1979.

FOI-82 FOISSEAU J., CHOLVY L.
"Elaboration d'un modèle de données pour la manipulation d'objets CAO".
Séminaire CAO en architecture et bâtiment.
14-15 Octobre 1982.

GAL-83a GALLAIRE H.
"Interrogation en langage naturel".
Dans DB3-83 .

GAL-83b GALLAIRE H.
"Interfaces graphiques pour les Bases de Données".
Dans DB3-83 .

GAR-82 GARDAN Y.
"Eléments méthodologiques pour la réalisation de systèmes de CFAO et leur introduction dans les entreprises".
Thèse doctorat d'Etat. Grenoble. décembre 1982.

GAR-83 GARDAN Y.
"Systèmes de CFAO".

GAR-85 GARDARIN G., VALDURIEZ P.
"Bases de Données relationnelles : analyse et comparaison des systèmes".
EYROLLES.

GAR-86a GARDAN Y.
"La CFAO".
Editions HERMES. 1986.

GAR-86b GARDAN Y.
"SACADO : système Adaptif de Conception Assistée et de Développement par Ordinateur".
Rapport LRIM. Metz. Août 1986.

GRA-78 GRABOWSKI H., EIGNER M.
"Employing a relational data structure in a CAO system".
Conf. Interactive Techniques in CAD. Bologne. 1978.

GRA-82 GRABOWSKI H., EIGNER M.
"A data model for a data base design".
in ENC-82.

GUT-82 GUTTMAN A., STONEBRAKER M.
"Using a relational system for computer aided design data".
Bulletin of IEEE in Database Engineering. Vol. 5. N°2.
Janvier 1982.

GUT-84 GUTTMAN A.
"New features in a relational database system to support computer-aided design".
Ph. D. dissertation. Electonics Research Lab.
Univ. of California. Berkeley. 1984.

HAR-84 HARDWICK M.
"Extending the relationnal database datamodel for design applications".
IEEE. Juin 1984

HAS-81 HASTIN R.L., LORIE R.A.
"On extending the functions of a relationnal database system".
IBM Research Report RJ3182. San José. 1981.

HEU-81 HEULLUY B.
"Propositions pour un processeur de gestion de la dynamique d'un système de données réparties".
Thèse Doctorat 3ème cycle.
CRIN. Juillet 1981.

HEU-86 HEULLUY B., MICHEL G.
"Le problème des bases de données en CFAO : Etude bibliographique".
Revue Iternationale de CFAO et d'Infographie. N°1.
Février 1986.

HOU-87 HOUBARD G.
"Bases de données et CAO : um mariage de raison".
MICRO-SYSTEMES. N° 79. Octobre 1987.

HUB-79 HUBBARD G.U.
"Computer-assisted logical database design".
CAD 11-3. Mai 1979.

KAT-83 KATZ R., LEHMAN T.
"Storage Structures For Versions and Alternatives ".
T.R. 479. Computer Sciences Departement.
University of Wisconsin. Madison. 1983.

KOR-83 KORIBA M.
"Data base systems : their applications to CAD software design".
CAD. Vol. 15-5. Septembre 1983.

LAU-82 LAURIERE J.L.
"Représentation et utilisation des connaissances".
T.S.I. Vol 1 N° 2. 1982.

LEC-84 LECOURVOISIER
"CASSIOPEE : un systéme intégré pour la CAO de VLSI".
Echo des recherches (118). Novembre 1984.

LCT-87 Licence de technologie de construction.
Sujets de projets. Université de Metz.

LOP-83 LOPEZ, PALLAZZO, VELEZ.
"The Tigre data base model".
Rapport de recherche Tigre (2) IMAG. Novembre 1983.

LOR-83 LORIE R.A., PLOUFFE W.
"Complex objects and their use in design transactions".
ACM Database Week. Mai 1983. 115-121.

MAR-86 MARTIN P.
"Logiciels de développement d'applications et d'interfaces homme-machine".
Techniques de l'Ingénieur.

MBD-86 Groupe de travail AFCET
"Convergence des Bases de Données et des Systèmes Experts".
MBD N° 5. Décembre 1986.

MIC-86 MICHEL G.
"Les Bases de Données en CAO : état de l'art".
Rapport de recherche.
LRIM. Metz. Janvier 1986.

MIC-87 MICHEL G.
"Maquette d'un SGBD pour la CAO".
Rapport Interne.
LRIM. Metz. Janvier 1987.

MIR-86 MIRANDA S., BUSTA J.M.
"L'art des Bases de Données 2".
EYROLLES.

NAN-84 NANARD J., NANARD M.
"Manipulation interactive de documents".
T.S.I. Vol. 3. N° 6. Novembre-décembre 1984.

NGU-84 NGUYEN, OLIVARES, WINNINGER.
"Coopération de Prolog et d'un SGBD Généralisé. Principes et applications".
Rapport de recherche TIGRE (15). IMAG. Avril 1984.

NGU-86 NGUYEN G.T., RIEU D.
"Modélisation d'objets en CAO : versions, représentations et implantations".
MBD N°5. Décembre 1986.

NIL-82 NILSON J.
"Artificielle Intelligence".
Springer Verlag.
Berlin - Heidelberg - New-York. 1982.

PEP-85 BOUCHET P., CHESNAIS A., FEUVRE J.M., JOMIER G., KURINCK A.
"Introduction aux sytèmes de Gestion de Bases de Données".
EYROLLES.

PHI-79 PHILIPS R.J., BEAUMONT M.J., RICHARDSON D.
"AESOP : an architectural relational database".
CAD. Vol. 11. N°4. Juillet 1979. pp 217-226.

REH-85 REHAK D.R., HOWARD H.C.
"Interfacing expert systems with design databases in integrated CAD systems".
CAD. Vol. 17. N°9. 1985.

RIE-85 RIEU D.
"Modèle et fonctionnalités d'un SGBD pour les applications CAO".
Thèse de Doctorat. Grenoble. Juillet 1985.

RIE-86 RIEU D.
"Nature, état, dynamicité de l'objet en CAO".
Journées Bases de Données Avancées. Gien. Avril 1986.

ROU-79 ROUSSOPOULOS N.
"Tools for designing conceptual shemata of a CAD data bases".
CAD. Vol. 11. N°3. Mai 1979.

SEM-86 KING R.
"A DBMS based on an Object-Oriented Model".
Expert Data Base Systems.
University of Colrado. 1986.

SHE-83 SHENOY R.S., PATNAIK L.M.
"Data definition and manipulation languages for a CAD data base".
CAD. Vol. 15. N°3. Mai 1983

STO-83 STONEBRAKER, RUBENSTEIN, GUTTMAN.
"Application of abstract data types and abstracts indices to CAD data bases".
ACM SIGMOD. San José. Mai 1983.

ULF-80 ULFSBY S.
"TORNADO user's guide".
Central Institute for Indistrial Research, PB 350.
Blindern. Oslo 3. Norway. Octobre 1980.

WON-77 WONG H.K.T., MYLOPOULOS J.
"Two views of data models in artificial intelligence and data base management".
INFOR. Vol. 15. N°3. Octobre 1977.

YAZ-85 YAZDANIAN K.
"Déduction dans les Bases de Données Relationnelles : fondements logiques et mise en oeuvre".
8ème Journées Informatiques de Nice. Mai 1985.

ZIN-81 ZINTL G.
"A codasyl CAD data base system".
Proc. 18 th. Design Automation Conference. Nashville. Juillet 1981.

UNIVERSITE DE METZ
Ile du Saulcy - 57000 Metz - B.P. 794 - 57012 Metz-cedex 1 - Tél. 87 30 26 63
Télex : UNIMETZ 930 462

Metz, le

Affaire suivie par :

LE PRESIDENT DE L'UNIVERSITE DE METZ

VU l'arrêté du 16 avril 1974 relatif au Doctorat de 3ème Cycle

VU la proposition du Directeur de l'UER SCIENCES EXACTES ET NATURELLES

A R R E T E

ARTICLE 1 : La composition du jury en vue de la soutenance de la thèse de Doctorat de 3° cycle de Monsieur **Gabriel MICHEL** intitulée

"Contribution à la conception et à la réalisation d'un système de gestion de bases de données pour la conception assistée par ordinateur"

est fixée comme suit :

Mme CREHANGE, Professeur (CRIN)
M. GARDAN, Professeur à l'Université de METZ
M. GOVAERT, Professeur à l'Université de METZ
M. HEULLUY, Maître-Assistant à l'Université de METZ
M. JUNG, Maître-Assistant à l'Université de METZ

ARTICLE 2: La soutenance aura lieu le MARDI 9 FEVRIER 1988 à 16H00 AMPHI A de l'I.U.T.

FAIT A METZ, le 25 JAN. 1988

Le Président de l'Université de METZ

UNIVERSITE DE METZ
LE PRÉSIDENT
57

J. DAVID

RESUME

Nous nous proposons d'introduire un outil de gestion de Bases de Données pour la CAO.
Les informations circulant dans un processus de conception peuvent être classées en deux Bases de Données: la Base de Données Projet et la Base de Données de Connaissances.
Les informations modélisées concernent essentiellement la Base de Données Projet: dans ce but est défini un système BD-CAO, appelé encore Modèle, composé d'un modèle de données orienté-objet et d'un ensemble d'opérations s'appliquant sur ce modèle de données.
Ce système BD-CAO permet d'exprimer pour les objets de conception, une hiérarchie structurelle et une hiérarchie fonctionnelle, des contraintes sur ces objets et des liens entre eux. Il possède des fonctionnalités adaptées à la CAO: conception ascendante, descendante et imbriquée, déduction, dynamicité de la structure, propriétés calculées, gestion d'objets incohérents ou incomplets.
Nous étudions le cadre d'utilisation du Modèle par l'intermédiaire d'exemples de conception dans le domaine de la mécanique puis nous définissons le dialogue par l'intermédiaire de deux propositions d'interfaces. Une maquette de ce système BD-CAO est en cours de réalisation.

Mots Clés:
attachement procédural, Base de Données, Conception Assistée par Ordinateur, déclenchement procédural, conception mécanique, hiérarchie fonctionnelle, hiérarchie structurelle, interface homme-machine, modèle de données, modèle orienté-objet.

RESUME

Nous nous proposons d'introduire un outil de gestion de Bases de Données pour la CAO.
Les informations circulant dans un processus de conception peuvent être classées en deux Bases de Données: la Base de Données Projet et la Base de Données de Connaissances.
Les informations modélisées concernent essentiellement la Base de Données Projet: dans ce but est défini un système BD-CAO, appelé encore Modèle, composé d'un modèle de données orienté-objet et d'un ensemble d'opérations s'appliquant sur ce modèle de données.
Ce système BD-CAO permet d'exprimer pour les objets de conception, une hiérarchie structurelle et une hiérarchie fonctionnelle, des contraintes sur ces objets et des liens entre eux. Il possède des fonctionnalités adaptées à la CAO: conception ascendante, descendante et imbriquée, déduction, dynamicité de la structure, propriétés calculées, gestion d'objets incohérents ou incomplets.
Nous étudions le cadre d'utilisation du Modèle par l'intermédiaire d'exemples de conception dans le domaine de la mécanique puis nous définissons le dialogue par l'intermédiaire de deux propositions d'interfaces. Une maquette de ce système BD-CAO est en cours de réalisation.

Mots Clés:
attachement procédural, Base de Données, Conception Assistée par Ordinateur, déclenchement procédural, conception mécanique, hiérarchie fonctionnelle, hiérarchie structurelle, interface homme-machine, modèle de données, modèle orienté-objet.