Informatique et architecture : réalisation du prototype du système MARS (Modélisation Architecturale à Référence Spatiale)

Brigitte David, Didier Berger, Colette Bernanose

206.

Ministère de l'Urbanisme et du Logement
Direction de l'Architecture
Secrétariat de la Recherche Architecturale

Ecole d'Architecture
de Grenoble

Association Grenobloise
de Recherche Architecturale

10 galerie des Baladins
38100 Grenoble

Informatique et Architecture:

Réalisation du prototype du système M.A.R.S.
(Modélisation Architecturale à Référence Spatiale)

B. DAVID - O. BERGER - C. BERNANOSE

RAPPORT FINAL

Contrat N° 84 01435 00 223 75 01

Juillet 85.

1. INTRODUCTION:

En 1984, un projet de recherche intitulé "Organisation des informations par références spatiales en vue de la programmation architecturale", nous a conduit à définir les objectifs et les fonctionnalités d'un système d'informations à références spatiales nommé :

Système M.A.R.S.
Modélisation Architecturale à Référence Spatiale

le présent rapport décrit l'application du travail précédent, c'est à dire la réalisation d'un prototype du système M.A.R.S.

L'élaboration de ce prototype devrait conduire à la mise en place d'un système d'informations à référence spatiale s'appuyant sur des outils de conception assistée par ordinateur (C.A.O.).
De tels systèmes existent et un soucis pendant notre étude consistera à imaginer et à prévoir les liaisons possibles ultérieurement.

PLAN DU RAPPORT :

Dans ce rapport, nous exposerons la spécificité de la C.A.O. dans le domaine de la programmation architecturale (Chapitre 2), puis nous donnerons quelques définitions ainsi que les fonctions du prototype (Chapitre 3).

Nous détaillerons ensuite les différents niveaux de modélisation des données (Chapitre 4), l'implantation effective des fonctions du prototype (Chapitre 5), pour terminer sur les perspectives d'avenir du système M.A.R.S. (Chapitre 7).

En annexe, une application du prototype sur un exemple volontairement simple de programme architectural, récapitule les différentes fonctions et les outils implantés.

LA CONFIGURATION UTILISEE :

Le matériel :

Du point de vue matériel, nous avons utilisé un micro ordinateur **BFM 186**, 16 bits disposant de 512K de mémoire centrale. Le stockage peut être fait sur disquettes de 1,2Mo ou sur disque dur de 10 Mo.

Le BFM a pris sa place sur le marché de X.A.O. car il constitue un bon outil graphique (résolution écran 965x625).
Cette option graphique du matériel est indispensable pour le système M.A.R.S puisqu'il devrait amener à la mise en place d'un système graphique interactif permettant l'établissement d'une proposition architecturale.

Le logiciel :

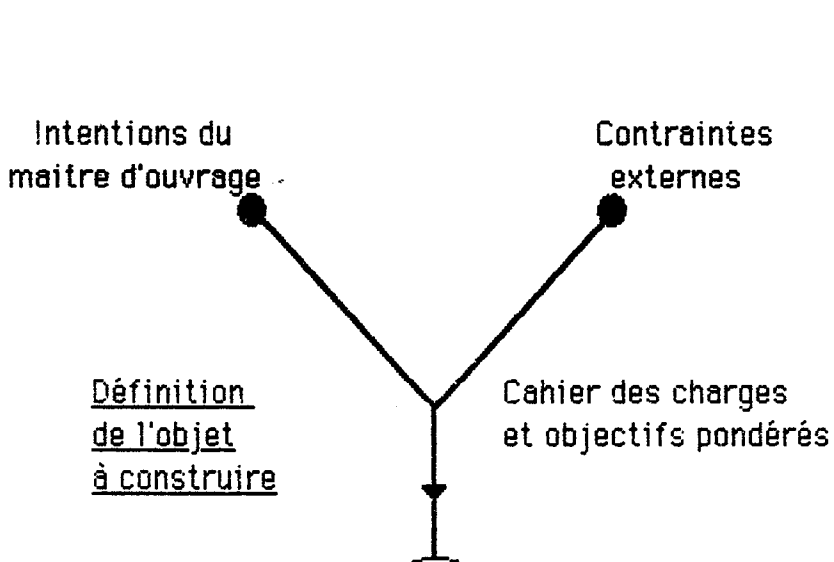
Le langage adopté est le **LOGO** car étant d'une approche assez facile, il peut être utilisé par des non-informaticiens.
En outre, c'est un langage très puissant puisqu'il permet la récursivité ainsi que des manipulations de listes.
Enfin, la sauvegarde sur disque permet à la fois celle des modules et des variables en mémoire, ce qui évite des lectures et écritures dans plusieurs fichiers (comme en Pascal par exemple).
Quand au nombre de primitives disponibles, il avoisine des 250.

2. LA CAO APPLIQUEE A LA PROGRAMMATION ARCHITECTURALE

2.1 Les données de base sont les formulations des attentes du maitre d'ouvrage.

exemple:

> "je souhaite une grande maison de plus de 140 m2, deux chambres, deux salles de bains, une cuisine avec cellier, un séjour proche de la salle à manger qui sera rustique, une terrasse devant le séjour .... la cuisine aura une fenêtre ouvrant sur l'entrée du terrain .... par sa volumétrie, ses matériaux, la maison aura un aspect rustique .... je ne peux pas investir plus de 800 KF"



Ce cahier des charges comprend à la fois des données impératives à respecter et des données auxiliaires à prendre en compte.

Sur la base de toutes ces informations est rédigé le programme de l'objet à construire que l'architecte utilisera.

Il pourra comporter des notions de :

- besoins
- contraintes physiques ou budgétaires
- critères de :
  - . proximité
  - . exclusion
  - . accès
  - . éclairage
  - . isolation
  - . matériaux
  - . ambiance
  - . volume .....

2.2 LA DEMARCHE C.A.O.

2.2.1 Le modèle concret :

celui ci s'élabore à partir d'un modèle abstrait accompagné de données relatives à l'objet à concevoir et qui sont:

- la liste des entités de l'objet
- la liste des attributs des entités

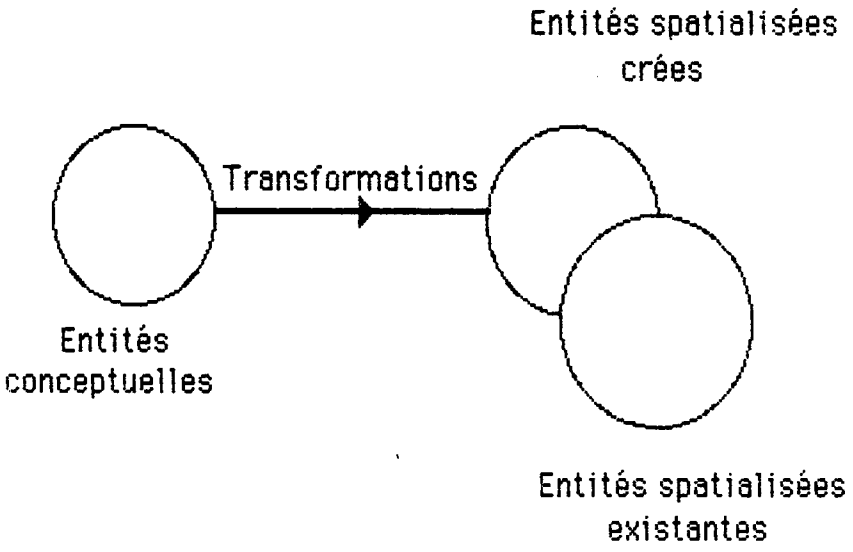
Ces listes sont fournies par l'analyse du cahier des charges.

2.2.2 Dans un deuxième temps, il faudra valuer les propriétés et relations affectées aux entités et les faire évoluer par modifications successives (suppressions, ajouts . . .)

2.2.3 Il s'agit ensuite par synthèse du modèle concret valué, d'aboutir à des premières propositions spatiales.
Si celle ci ne conviennent pas, il faut revoir soit le choix des entités, soit la valuation et faire à nouveau évoluer le projet à l'aide des fonctions du système.

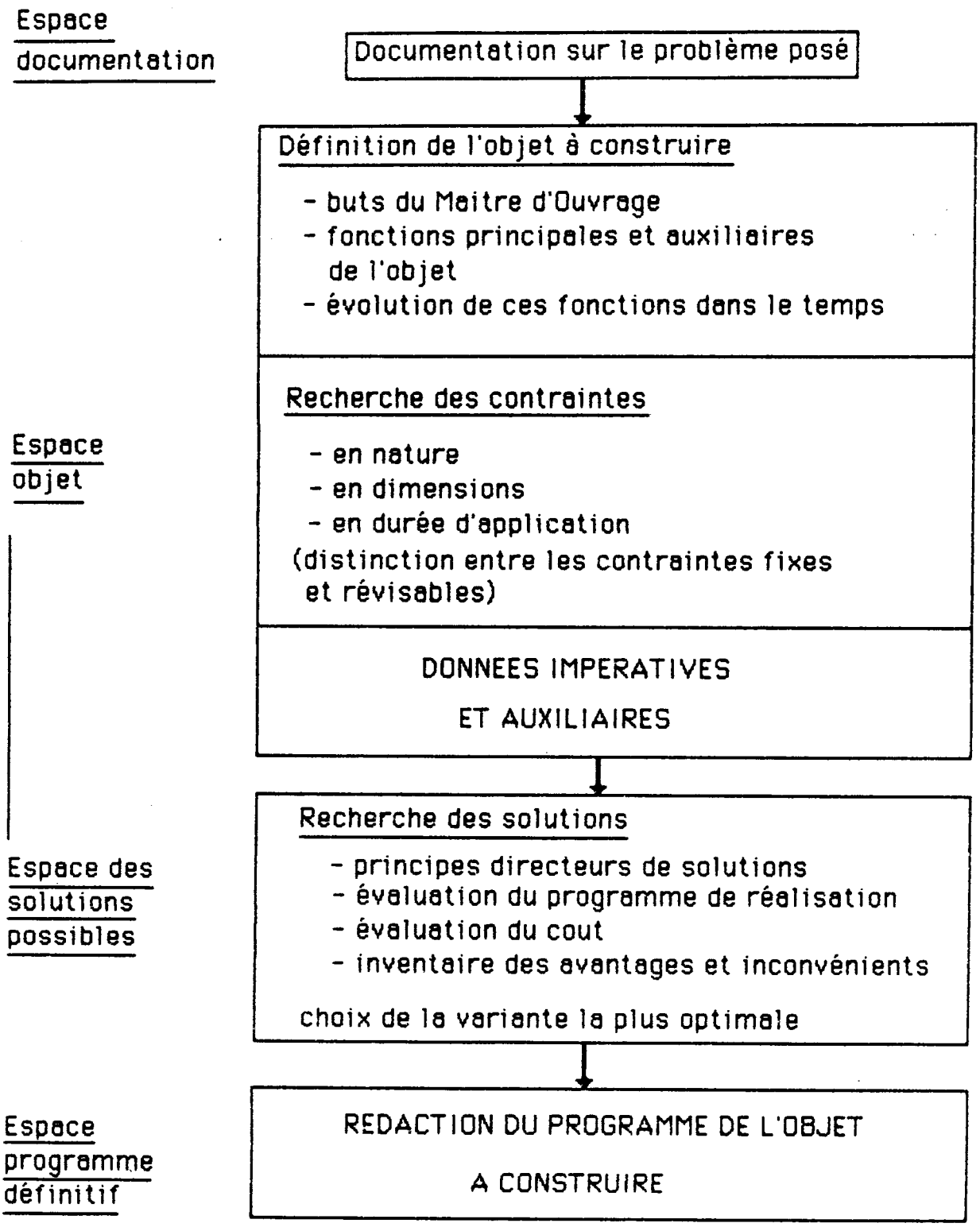
2.3. LES FONCTIONNALITES DU SYSTEME M.A.R.S.

M.A.R.S. doit traduire, les attentes du maitre d'ouvrage sous forme d'entités conceptuelles, et l'environnement dans lequel doit s'inscrire la construction sous forme d'entités spatialisées. Celles ci résultent du traitement des entités conceptuelles et viennent s'ajouter aux entités spatialisées déjà existantes.



Notre étude sera donc centrée sur les problèmes de modélisation de données pour traduire les premiers souhaits du maitre d'ouvrage ainsi que sur l'interactivité du système. On souhaite en effet, créer un système interactif facile d'utilisation et garantissant la meilleure fiabilité.
Cette fiabilité sera assurée par la détection de la plupart des erreurs de manipulation des fonctions ainsi que par la vérification de la cohérence des données et des résultats après traitement.

2.4. SCHEMA DES ETAPES DE LA PROGRAMMATION ARCHITECTURALE



3. DEFINITIONS

3.1 ENTITE :

une entité représente un atome d'information du projet. Elle peut avoir une nature concrète (salle à manger, chambre . . .) ou abstraite (ambiance, lumière, notion de prestige . . .).
On la caractérise par une liste d'attributs < a b c > et elle peut être liée à d'autres entités ou avoir des attributs liés aux attributs d'autres entités.

3.2 ENTITE CONCEPTUELLE :

elles appartiennent au domaine "conceptuel" et sont directement issues des objectifs du maitre d'ouvrage.

exemple: d'entités conceptuelles
- concept "dormir"
- ambiance chaleureuse
- espace calme

Ce sont des idées sous forme abstraites ou sous forme d'images (aspect extérieur "rustique"). On a souhaité introduire ces notions conceptuelles dans le système M.A.R.S. car elles permettent de conserver la démarche de réflexion de l'architecte qui, à partir de ces notions, entrevoit les orientations possibles du projet.

3.3. ENTITE OPERATIONNELLE :

elles appartiennent au domaine "opérationnel" et peuvent être soit directement issues des objectifs du maitre d'ouvrage soit déduites d'entités conceptuelles.

exemple: d'entités opérationnelles

- chambre
- salle de bain

déduites d'entités conceptuelles

entité conceptuelle: dormir
entité opérationnelle associée: chambre

entité conceptuelle: prestige
entité opérationnelle associée: revêtement sol marbre

Les entités opérationnelles sont donc des notions plus concrètes que les entités conceptuelles.

3.4 ENTITE SPATIALISEE :

elles appartiennent au domaine "de l'espace" et assurent le positionnement des entités par rapport à un référentiel (O,X,Y,Z).
Toutes les entités ne peuvent être spatialisée.
Une entité opérationnelle peut devenir spatialisée dès qu'on lui affecte un attribut de positionnement.

exemple: d'entités spatialisées
entité opérationnelle : chambre
entité spatialisée: chambre origine (Xo,Yo,Zo)
longueur a
largeur b

L'utilisateur peut soustraire de la suite du projet les entités spatialisées qu'il juge correctes ou les entités conceptuelles ou opérationnelles correctes mais qui ne peuvent pas être positionnées dans l'espace (on lui donnera alors une référence spatiale, c'est à dire un lien avec une entité spatialisée).

3.5. ATTRIBUTS D'UNE ENTITE :

ils décrivent l'entité et constituent une liste éventuellement vide attachée à l'entité.
Au sein même d'une entité, les attributs peuvent être liés. On établira donc les liens de dépendances horizontaux qui permettront de vérifier la cohérence des informations au sein d'une même entité - c'est à dire la cohérence de la valuation des attributs-.

3.6. LES ENTITES CONCEPTUELLES OPERATIONNELLES SPATIALISEES

quelques exemples :

CONCEPTUELLES	OPERATIONNELLES	SPATIALISEES
activité de dormir	chambre à coucher de x m2	localisée en x, y, z
caractère rustique d'un batiment		
espace calme	murs insonorisés	
espace de prestige	revêtement de sol en marbre	hall d'entrée x, y, z
confort	chauffage central	
ambiance de détente	espace 1 avec des coussins et une cheminée	
prix abordable	prix maxi: 600 KF	

4. FONCTIONS DU PROTOTYPE :

Il doit permettre :

- la création et la suppression d'entités
- la mise à jour d'attributs d'entités
- la mise à jour de la valeur d'attributs d'entités
- la transmutation d'entités (passage tri-directionnel d'une nature d'entité à une autre: conceptuelle; opérationnelle, spatialisée)
- la création d'une hiérarchie d'entités avec la construction d'un arbre repéré par sa racine. Cette hiérarchie crée des liens de dépendance verticale entre pères et fils de l'arbre. Ceci impose de vérifier la cohérence verticale des valeurs des attributs des noeuds de l'arbre lors de la valuation des entités ou lors de la mise à jour de cette valuation.
- Cette hiérarchie engendre de nouveaux traitements au cours de l'évolution du processus de conception. Ainsi, si l'on supprime ou si l'on ajoute des entités alors que la hiérarchie a déjà été crée, il faut revoir la cohérence verticale de l'arbre et l'affectation de la descendance (ou du sous arbre) des noeuds supprimés ou de l'ascendance des noeuds ajoutés.
  Nous reviendrons sur ces problèmes résolus par programme.

- l'affichage
- la sauvegarde sur disque ou disquette

Nous donnons plus de détails sur ces fonctions et le modèle de données dans les chapitres suivants.

Le système M.A.R.S. doit gérer les dialogues, les données et les traitements. On distingue trois grandes classes de dialogues:

- les fonctions de manipulations précédentes
- les programmes de traitement utilisant les fonctions de manipulation
- les fonctions d'extension permettant l'évolution du système notamment la prise en compte de nouveaux types de données et leur sémantique.

Le prototype dispose des trois classes énumérées précédemment.

5. LES DIFFERENTS NIVEAUX DE MODELISATION DES DONNEES

La modélisation des données conditionne toute la souplesse du système. Ne souhaitant pas créer une base de données trop figée, l'utilisateur dispose d'une typologie d'attributs qu'il pourra utiliser pour renseigner ses entités. On les appellera : type de propriétés

Ainsi lors de la définition des entités, lorsqu'on s'appuie sur un type déjà défini et connu par M.A.R.S., il n'est pas nécessaire d'introduire d'autres informations sur ce type. Par contre, si on souhaite utiliser un attribut non typé, il faut au préalable l'introduire dans la typologie des attributs.

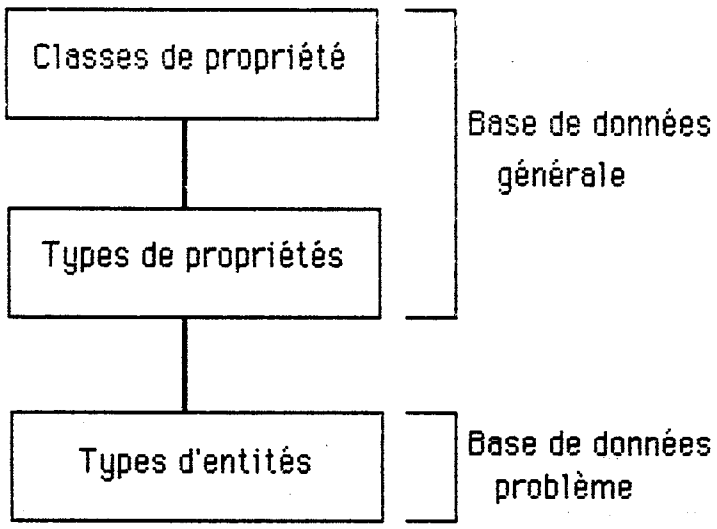
Comment l'introduire ?

pour faciliter l'introduction de types de propriétés, on a défini un niveau supplémentaire: celui des classes de propriétés. Ainsi chaque propriété aura une classe bien définie appartenant à une typologie de classes.

Nous disposons alors d'une modélisation des données sur trois niveaux:

- les classes de propriétés
- les types de propriétés
- les types d'entités

Les deux premiers constituent une base support du troisième et peuvent être utilisés dans le cadre de projets distincts. En ce sens, ils constituent une base de données générale alors que les types d'entités constituent la base de données d'un problème particulier ou la base de données problème.



Les trois niveaux de modélisation.

Nous avons recensé quatre grands types de classes. Nous les donnons à titre d'exemple mais l'utilisateur peut en créer autant qu'il le désire.

- classe "qualitative" : les propriétés de cette classe indiquent la nature qualitative de certains éléments auxquels on ne peut pas attribuer une valuation numérique.

exemple :

propriété	valuation
ambiance	<chaleureuse>
couleur	<bleu>

- classe quantitative : les propriétés de cette classe peuvent recevoir une valuation numérique.

exemple:

propriété	valuation
volume	<50 m3>
surface	< 40 m2>

- classe de positionnement: les propriétés de cette classe peuvent recevoir une valuation de localisation dans l'espace

exemple:

propriété	valuation
orientation	< nord 90°>
origine	< X Y Z >

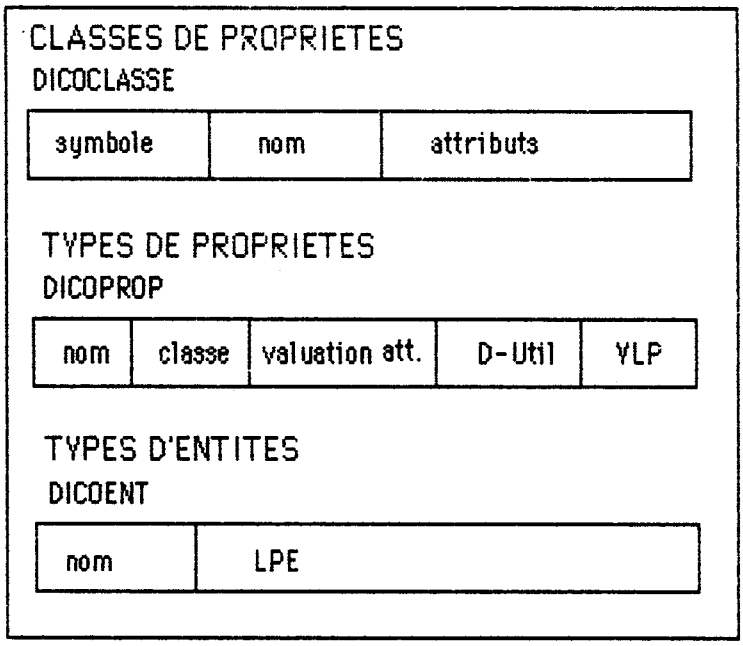
- classe relation entre entités: les propriétés de cette classe traduisent une relation de dépendance de l'entité.

exemple:

propriété	valuation
proximité	< cuisine >
contiguïté	< chambre>

On impose toujours à une propriété d'appartenir à une classe de propriété; par contre, on peut définir une entité non typée (c'est à dire qu'elle peut être sans attribut).

Nous avons crée une base de données relationnelle décrite ci après :



Signification des champs et variables:

- DICOCLASSE :
  - représente le dictionnaire des classes de propriétés.
  - Le symbole constitue l'index de la table "DICOCLASSE" et permet d'accéder aux chaps nom et attributs du symbole.
  - symbole : symbole de la classe
  - nom : nom de la classe en toutes lettres
  - attributs : attributs de la classe

exemple:

Symbole	Nom	Attributs
QUAN	quantitatif	<D-VAR unité>
QUAL	qualitatif	< D-VAR prix max unité>
PO	positionnement	<D-VAR X Y Z angle>
RE	relat. ent. entités	< D-VAR>

(D-VAR = domaine de variation)

- DICOPROP :

représente le dictionnaire des types de propriétés.
Le nom constitue l'index d'accès de la table "DICOPROP" et permet d'accéder aux champs classe, D-UTIL, VLP ainsi qu'à chacun des attributs de la classe de la propriété qu'il faut aller récupérer dans "DICOCLASSE"
- nom : nom de la propriété
- classe : classe de la propriété
- D-UTIL : domaine d'utilsation de la propriété. C'est à dire conceptuel, opérationnel ou spatialisé (C, O, S)
On peut en effet définir un domaine d'utilisation per propriété. Lors de l'affectation des entités, l'entité prend pour nature le "ou" des domaines d'utilisation de ses attributs.

exemple :

entité Séjour < a b >

la propriété a ayant pour D-UTIL <O>
la propriété b ayant pour D-UTiL <C>

L'entité Séjour pourrait prendre pour nature conceptuelle ou opérationnelle.

- VLP : vecteur lien entre propriétés
c'est une liste comprenant les propriétés de "DICOENT" avec lesquelles est liée une propriété donnée.

exemple :

NOM	CLASSE	VALUATION DES ATTRIBUTS	D-UTIL	LP
volume	QUAN	D-VAR=< 0 . . 500> unité =< m3 dm3 m3>	0	< surface longueur>
surface	QUAN	D-VAR= < 0 . . 400> unité = < ha a m2>	0	< longueur volume>
longueur	QUAN	D-VAR= < 0 . . 20 > unité = < km m >	0	< surface volume>
matériaux	QUAL	D-VAR= < bois béton > prix max= 500 F unité = m3 kg t	0	< >
origine	PO	D-VAR= < > X= <0 . . 50 > Y= < 0 . . 45 > Z= <0 . . 6 > Angle=< >	5	< >
orientation	PO	D-VAR=< ouest sud est> X=< > Y=< > Z=< > Angle=< 0 . . 360>	5	< >
proximité	RE	D-VAR=< quelconque >	0	< >

< > représente le liste vide.
On constate que le troisième champ de "DICOPROP" est variable. Lors de la valuation des attributs des entités, il suffira d'entrer des données comprises dans les domaines mentionnés dans ce troisième champ.
Nous spécifierons plus tard les manipulations possibles sur tous ces champs.
La propriété volume est liée aux propriétés surface et longueur; ainsi, si on modifie la valeur du volume dans une entité, il faudra veiller à la cohérence dite "horizontale" de ses attributs.

par exemple:
entité Séjour <volume, surface, longueur>

si on modifie la longueur, il faut réajuster les autres propriétés qui lui sont liée.

VLP: vecteur lien entre propriétés

- DICOENT: représente le dictionnaire des entités
Le nom de l'entité constitue l'index de la table "DICOENT" et permet d'accéder au champ "LPE".

LPE: liste des prorpiétés de l'entité

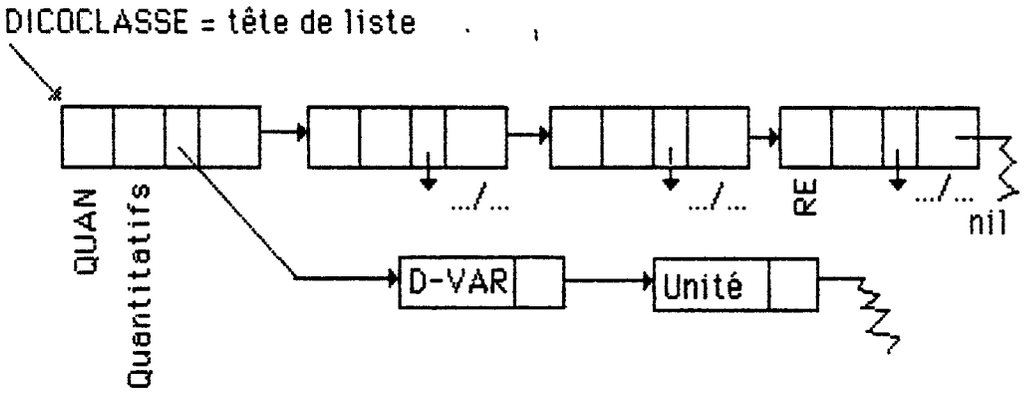
exemple:

Nom	LPE
Maison	< orientation, surface >
Etage 1	< orientation proximité >
Etage 2	< origine orientation volume >

Nous avons donc crée trois dictionnaires qui sont en logo, les trois listes suivantes:

DICOCLASSE ⟶	<QUAN QUAL PO RE>
DICOPROP ⟶	<volume surface longueur matériaux origine orientation proximité>
DICOENT ⟶	<maison étage1 apart1>

Dans une structure type 'PASCAL', on aurait les listes suivantes:



On définirait un type d'enregistrement:

type dicocla = record

| symbole : chaine de caractères
| nom : " " "
| attributs: | typeatt
| suivant : | dicocla
| end;

type typeatt = record

| at : chaine de caractères
| svt : | typeatt
| end;

La structure est nettement plus lourde !

D'autre part, pour accéder à un enregistrement, il faudrait parcourir séquentiellement toute la liste.

Par contre, en logo, la récupération des propriétés d'un élément d'une liste est très aisée.

Donnons un exemple des performances logo au niveau des listes:

```
dprop  " QUAN " nom " quantitatif
```

donne la propriété quantitatif au champ "nom de QUAN

```
rprop " QUAN " nom
```

retourne la propriété "nom de QUAN, soit: "quantitatif

rprop

Cout en instructions = 2

5. IMPLANTATION DES FONCTIONS DU SYSTEME M.A.R.S. SUR B.F.M. 186

5.1 OPERATIONS SUR LES ELEMENTS DES DICTIONNAIRES :

Les opérations disponibles sont :

- création
- suppression
- ajout
- affichage

Dans tout le programme, 'CREATION' signifie que le dictionnaire initial sera détruit (c'est à dire dicoclasse pour les classes, dicoprop pour les propriétés, dicoent pour les entités).

'AJOUT' signifie qu'on souhaite ajouter une liste d'éléments à un dictionnaire. La double définition est implantée : c'est à dire qu'on ne peut pas mettre le même élément dans un dictionnaire. Des tests d'erreurs sur le nombre d'arguments rentrés sont prévus.

'SUPPRESSION' signifie qu'on souhaite supprimer une liste d'éléments d'un dictionnaire. Des tests d'erreurs sur l'existence des éléments dans le dictionnaire ou sur le dictionnaire vide sont prévus.

'AFFICHAGE' signifie qu'on souhaite afficher les éléments du dictionnaire avec la liste des propriétés attachée à chacun d'eux.

Modules créés pour ces fonctions :

classes	Propriétés	Entités
creercla	creerprop	
creerent		
ajcla	ajprop	ajent
supcla	supprop	supent
affcla	affprop	affent

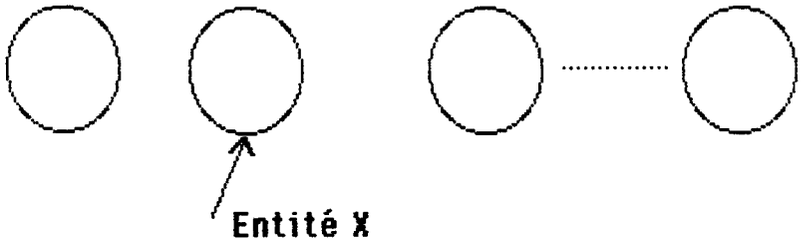
N.B. : La fin de la création, de l'ajout ou de la suppression est donnée par 'return'.

5.2 OPERATIONS SUR LES TYPES D'ENTITES :

En plus des fonctions décrites ci-dessus, on dispose de la fonction :
' mise à jour des entités ' : 'majent
qui permet de modifier le nom d'une entité, ou sa liste complète d'attributs ou simplement une partie de cette liste.

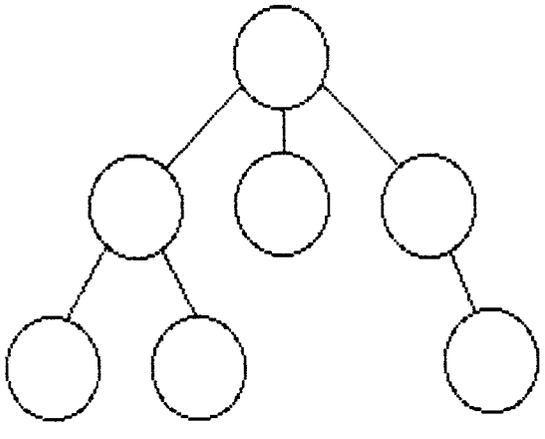
opérations d'évolution du processus de conception :

Lors du processus de conception, il n'est pas souhaitable de travailler sur un seul niveau :



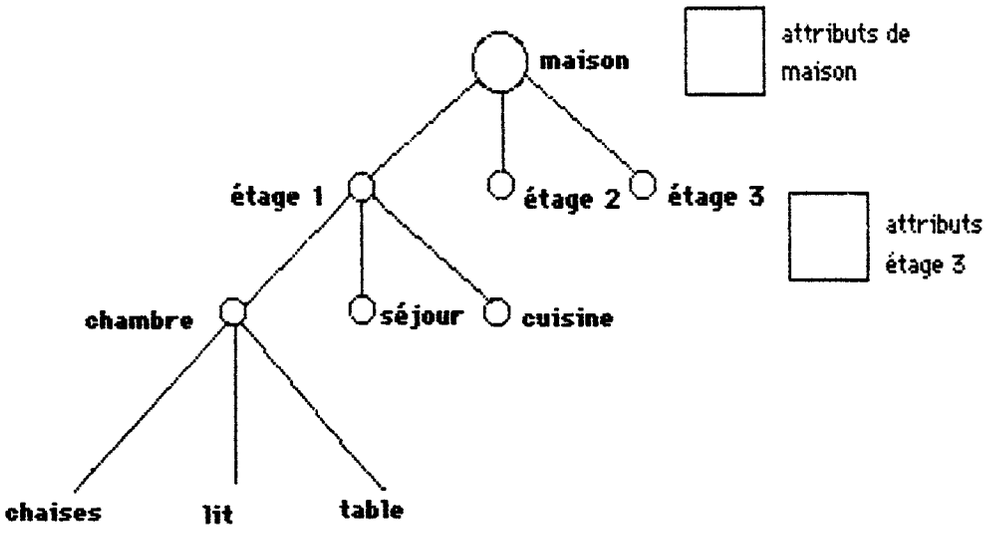
travail à plat sur les entités

Ainsi, avant de travailler sur la chambre de l'étage 1 de l'immeuble 1, on préfèrera travailler sur l'immeuble 1, puis à l'étage 1, enfin sur la chambre.
Ce qui justifie la création d'une hiérarchie des entités :



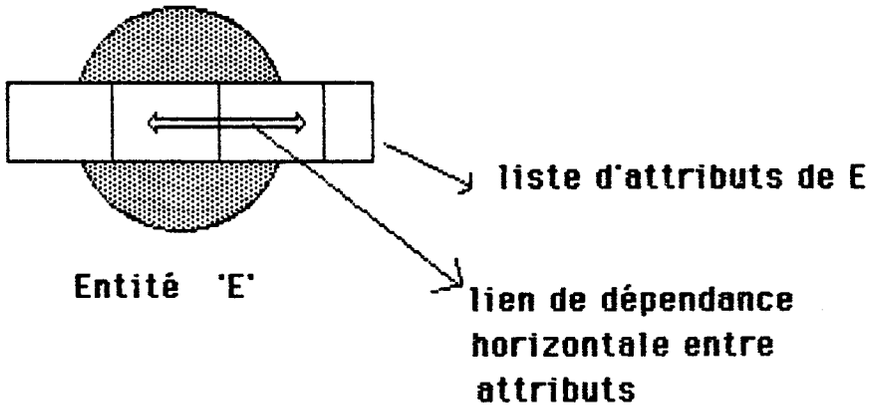
Hiérarchie d'entités

exemple :



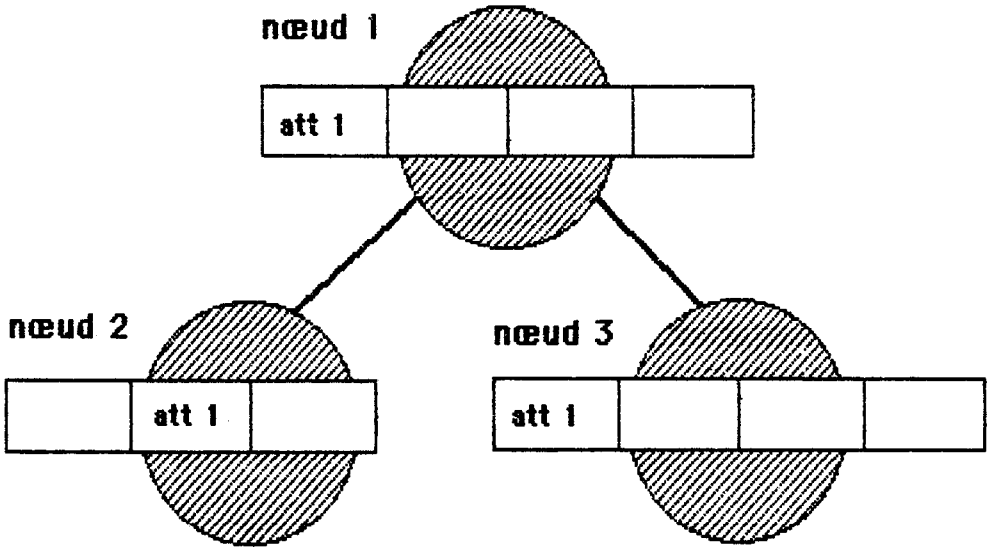
les problèmes de cohérence verticales ou horizontales :
Ces problèmes interviennent lors de la valuation de la hiérarchie.

cohérence horizontale :



Nous avons déjà mentionné ce problème. Il est résolu par le module "cohearbreh" qui demande pour un nœud donné si la cohérence horizontale est respectée. Si elle ne l'est pas, on peut modifier la valeur d'un attribut quelconque.

cohérence verticale :



si nœud 1, nœud 2 et nœud 3 ont un attribut commun, il faut vérifier qu'il y a cohérence des trois.
Par exemple :

si att1 = surface
nœud 1 = appart 1
nœud 2 = séjour
nœud 3 = appart 1 - séjour

on doit vérifier : surface (nœud 1) = Σ surface (nœud i)

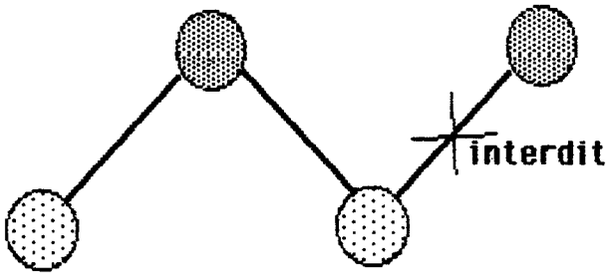
Cette cohérence est vérifiée par le module "cohearbrev" qui demande pour un nœud donné (le nœud père) si la cohérence verticale avec tous ses fils est respectée. Si elle ne l'est pas, on demande :

1°) la liste des nœuds à modifier,
2°) pour chaque nœud de cette liste, la liste des attributs dont on souhaite modifier la valuation.

Modélisation de la hiérarchie des entités :

On définit pour chaque entité deux attributs supplémentaires : père et fils (fils est une liste d'entités).

On interdit à un fils donné d'avoir plusieurs pères :

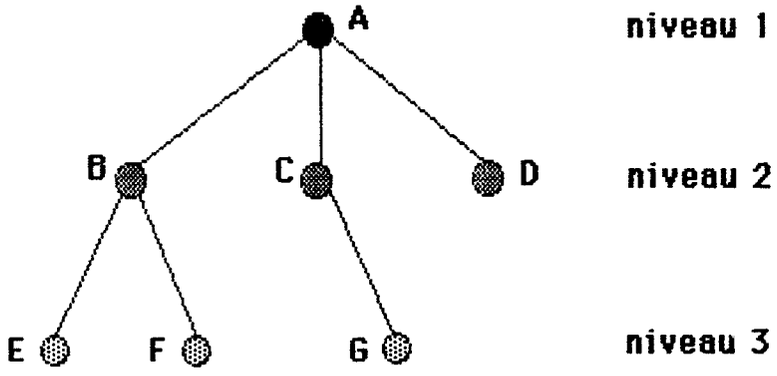


Pour cela, on propose une liste de fils possibles, si l'élément rentré n'est pas dans la liste, il y a message d'erreur et demande d'une entrée correcte.

L'arbre est repéré par sa racine dont le père est le mot "RACINE". La hiérarchie est stockée dans un dictionnaire appelé "vieux pères".
Les feuilles de l'arbre sont la liste vide.
Toutes les entités doivent être installées dans la hiérarchie.

Intérêt du dictionnaire "vieux pères" :

Les entités sont stockées dans le dictionnaire par niveaux, le premier élément étant la racine.



vieuxpères = < A B C D E F G >
= < racine, fils (racine), fils du premier fils de la racine, fils du 2° fils, ..., ... >

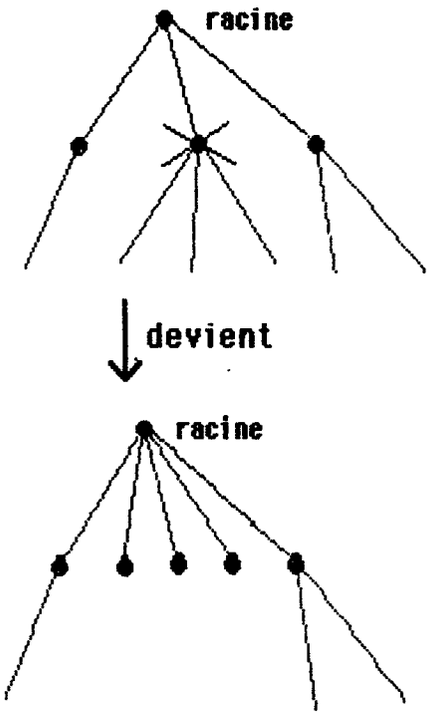
Remarque très importante : la valuation doit être réalisée après la hiérarchie, elle s'effectue niveau par niveau, en consultant vieuxpères car le module 'valuation' appelle le module 'cohearbrev' qui est lié à l'arbre.

Nota Bene : c'est le module 'classement' qui réalise le

stockage dans vieux pères.
Si lors du traîtement, on souhaite supprimer ou ajouter des entités dans dicoent, qu' advient-il de la hiérarchie ? Il faut la remettre à jour.

Pour cela, nous avons créé un module 'majhiera' mise à jour de la hiérarchie qui vérifie si "vieuxpères" et "dicoent" sont identiques. 'Vieuxpères' est le résultat de la dernière hiérarchie. Si 'dicoent' a été modifié, on appelle systématiquement un ou deux modules :

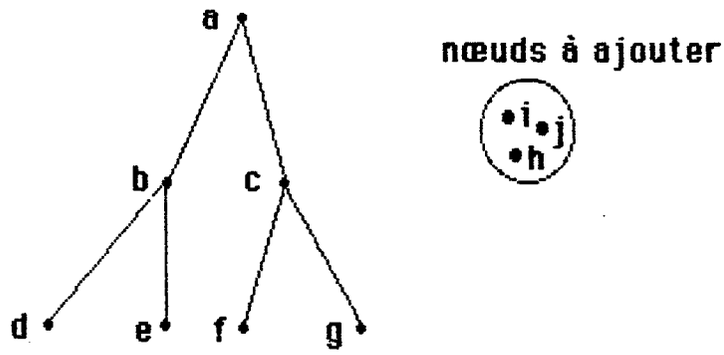
'suppnœud' : suppression d'un nœud dans la hiérarchie. On ajoute alors au père du nœud supprimé une liste de fils comprenant ses anciens fils, **moins** le nœud supprimé,
plus les fils du nœud ajouté.
On enlève ensuite le nœud supprimé de "vieuxpères".



suppression d'un nœud

On sait qu'il faut supprimer un nœud lorsque 'vieuxpères' contient des élémentsqui ne sont pas dans 'dicoent'.
On répète l'opération de suppression pour tous les nœuds de 'vieuxpères' qui ne sont pas dans 'dicoent'.
Si ensuite, 'dicoent' contient des éléments qui ne sont pas dans 'vieuxpères', alors on appelle le module :

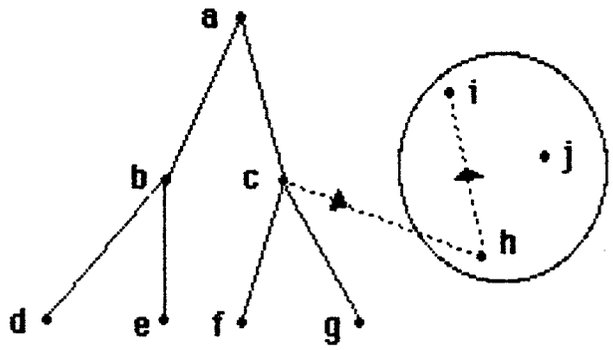
'ajnœud' qui ajoute un élément dans la hiérarchie.



hierarchie actuelle

'ajnœud' demande tous les nœuds à ajouter

- le père parmi tous les nœuds de 'dicoent' (sauf le nœud traîté)
- le fils parmi les nœuds de 'dicoent' qui ne sont pas dans 'vieuxpères' et qui n'ont pas encore de père.



exemple : père (h) = c à choisir parmi la liste :

< a b c d e f g i j >

fils (h) = j à choisir parmi la liste

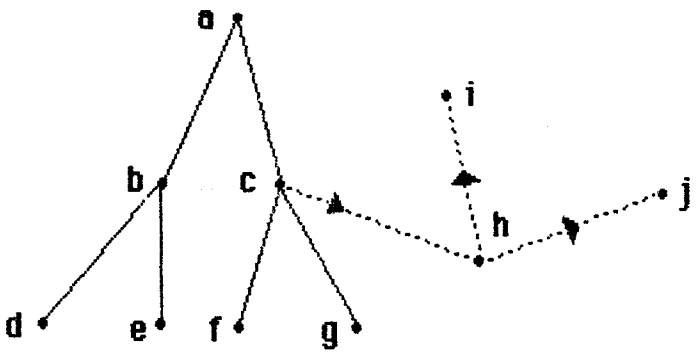
< i j > ---> père (i) = h

on ajoute ensuite h et i à 'vieuxpères'

et on continue le traîtement.

N.B. : on initialise fils (i), fils (j), fils (h) à la liste vide au départ.

hierarchie modifiée

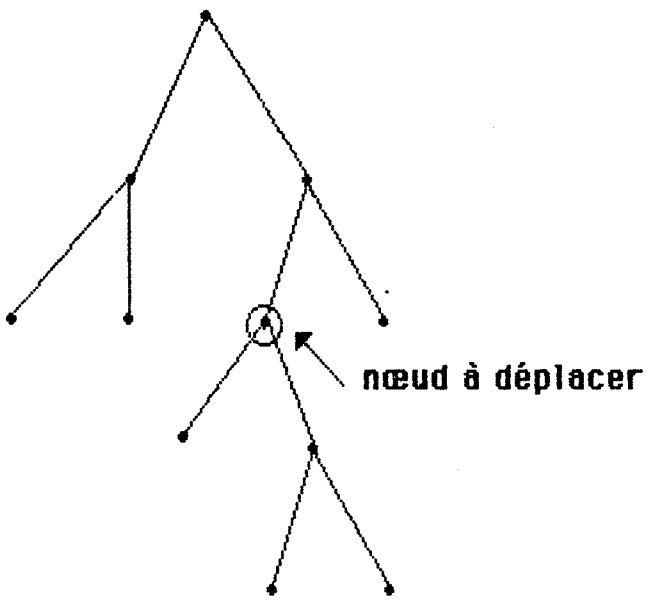


Cette méthode permet encore de ne pas donner le même père à un fils donné.

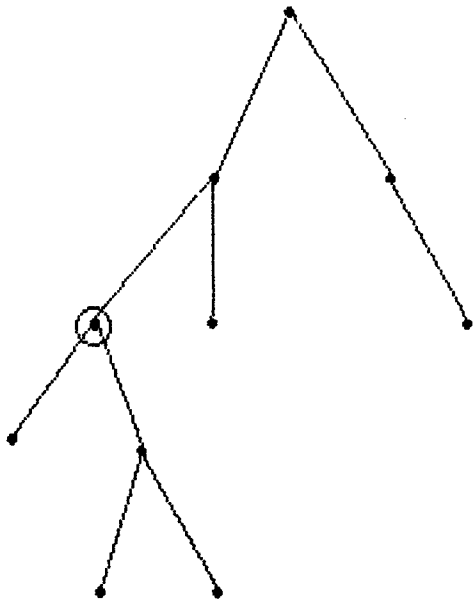
Autres opérations sur la hiérarchie :

Le module 'déplanœud' permet de déplacer un nœud et son sous-arbre dans la hiérarchie.
Il demande quel est le nouveau père du nœud déplacé.

avant déplacement



après déplacement



Le module 'affhiera' affiche pour toutes les entités de 'dicoent' le père et les fils.

Le module 'affval' affiche la valuation des entités, c'est à dire la valeur des attributs de chaque entité.

5.3 SAUVEGARDE DES FICHIERS :

Le module 'sauvegarde' sauve sur disque ou disquette (au choix de l'utilisateur) les données créées lors du processus de conception dans un fichier dont le nom est demandé à l'utilisateur.
On affiche la liste des fichiers existants pour éviter toute destruction.

5.4 LES UTILITAIRES CREES :

Lors de l'élaboration du programme, nous avons jugé intéressant de créer des utilitaires ou modules utilisables plusieurs fois.
Ce sont les modules suivants :

'impartie' : affiche une partie des procédures existant dans l'espace de travail.

En entrée : donner la liste des procédures à imprimer.
En sortie : impression.

'imprime' : affiche toutes les procédures de l'espace de travail.

'tantque' : c'est l'analogue du 'while' 'pascalien'
nécessite deux arguments :
- une condition
- une instruction.

'appartenance' : permet de savoir si une liste est incluse dans une autre.
Elle nécessite deux arguments qui sont deux listes.

'supprimer' : permet de supprimer un mot d'un dictionnaire.
Elle nécessite deux arguments, un mot et une liste.

'suplist' : permet de récupérer les éléments d'une liste 2 qui ne sont pas dans une liste 1.
deux arguments : liste 1 (en premier)
liste 2 (en second).

'tempo' : crée une temporisation utile pour visualiser l'affichage au cours du déroulement du programme.
Equivaut à 'wait 15'.

Remarque : Le nombre de modules en LOGO est très important car on ne peut pas insérer de return au sein d'une instruction.

Exemple : Si condition < instr 1 > <instr 2 > signifie si la condition est réalisée, exécuter l'instruction 1 sinon exécuter l'instruction 2.

Instr 1 doit être un nouveau module de même que instr 2 dans le cas où toute l'instruction 'si' ne peut pas être écrite sur moins de 256 caractères.

CONSULTER LE LISTING ET LE JEU D'ESSAI EN ANNEXES

POUR PLUS DE PRECISIONS SUR CE CHAPITRE V

5.5 RECAPITULATION DES FONCTIONS DU PROTOTYPE :

MENU

CLASSES DE PROPRIETES :

1. Créer classes
2. Ajouter classes
3. Supprimer classes
4. Afficher classes

TYPES DE PROPRIETES :

5. Créer types
6. Ajouter types
7. Supprimer types
8. Afficher types

TYPES D' ENTITES :

9. Créer entités
10. Ajouter entités
11. Supprimer entités
12. Afficher entités
13. Mettre à jour
14. Fusionner deux entités
    < à créer = 1 ajout + 2 suppressions >
15. Eclater une entité
    < à créer = 2 ajouts + 1 suppression >

HIERARCHIE DES ENTITES :

16. Créer la hiérarchie
17. Mettre à jour la hiérarchie
18. Déplacer un nœud
19. Afficher la hiérarchie

VALUATION : (à faire en dernier lieu)

20. Valuer les entités
21. Mettre à jour
22. Afficher la valuation
23. Sauvegarde sur disque

6. UNE APPLICATION

Nous avons testé notre programme sur les objectifs du maître d'ouvrage suivants :

- *une grande maison*
- *deux enfants + parents*
  2 chambres + salle bains chambre + salle bains
- *une cuisine de plus de 12m2 avec cellier*
- *un grand séjour (40 à 50 m2)*
  avec une grande cheminée
- *proximité du séjour et de la salle à manger*
  qui sera: prestigieuse
  rustique
- *un sous-sol complet avec*
  rangement
  garage pour deux voitures
  un supplément à aménager
- *une terrasse devant séjour*
- *un balcon devant la chambre des parents*
- *une cuisine ayant une fenêtre sur*
  l'entrée du terrain
- *un aspect extérieur "provençal"*

En annexe, nous avons identifié les entités et nous donnons le modèle des données valuées ainsi que le dialogue homme - machine assurant l'entrée de toutes les informations en mémoire.
Nous n'avons pas imprimé tous les tests et cas d'erreurs implantés, mais est vérifiable directement sur la machine.

7. CONCLUSIONS

PERSPECTIVES D'AVENIR DU SYSTEME M.A.R.S.

Nous avons essayé d'implanter les outils de base du prototype. Ils fonctionnent semble-t-il correctement et permettent d'envisager la poursuite de l'étude du système M.A.R.S. car il reste encore beaucoup à faire.

Nous donnons ci-après quelques idées pour greffer d'autres outils sur le système que nous avons implanté.

M.A.R.S. doit devenir un système à référence spatiale. Comment introduire cette référence ? Voici deux propositions :

1° idée : créer une nouvelle table ou un nouveau dictionnaire "DICOREFSPA" qui comprendrait deux champs :

- un champ "nature"
- un champ "lienent" (lien entité) qui ferait référence à une liste d'entités du dictionnaire des entités.

nature	lienent

dicorefspa

exemple :

nature	lienent
batiments du quartier	< ba 432 ba 510 >
immeuble de plus de 3 étages	< im 100 im 101 >
ambiance calme	< parc 'a' jardin 'j' >
prestige	< palais des sports >
zone constructible	< parcelle n° 510 >

avec :
ENTITES : = < ba 80 , ba 81, ba 82, ... ba 600, im 100, ...im 150, parc 'a', parc 'b', ...parc 'z', jardin 'a', ... jardin 'z',

palais des sports, parcelle n°500, ...parcelle n°510 >

On peut ainsi, avec un thème tel 'PRESTIGE' accéder aux entités qui vérifient ce thème (ou condition).

2° idée :

Créer de nouvelles entités qui seraient les thèmes ci-dessus et ajouter dans les propriétés affectées au dictionnaire des entités un nouveau champ

Lien Référence Spatiale = ' LRS '

qui serait une liste d'entités.

nom	L P E	L R S
batiments du quartier	/	< ba 432 ba 510 >
< ba 432	< a b >	/
< ba 510	< c d >	
ba 100 | ba 600	< . . . >	
im 100 | im 150	< . . . >	
prestige palais sports |	< e f >	palais des sports

Cette structure est moins pratique car elle place toutes les entités sur le même plan au niveau de l'évolution du processus de conception. Il est peut-être préférable d'envisager l'idée n° 1.

Au fur et à mesure de l'évolution du processus, il faudra veiller à mettre à jour le champ 'L R S', notamment lors de la création ou de la suppression d'entités.

Il serait également intéressant de sauvegarder l'historique du processus.

Il suffirait de sauvegarder en mémoire à chaque étape la structure arborescente des différentes variantes -c'est à dire ce que nous avons appelé ' vieuxpères ' - ainsi que la valuation de tous les nœuds et le dictionnaire des références spatiales. A cela, on pourrait ajouter la date d'enregistrement.

Enfin, quant à l'objectif final de l'architecte qui est la rédaction des plans, il faudra introduire les possibilités graphiques du micro-ordinateur. Là encore, donnons une idée au niveau du choix de placement des entités dans l'espace :

1. Regrouper dans un dictionnaire, toutes les entités ayant des attributs de positionnement ou de relation entre entités.

2. Affecter un poids aux entités qui ont le plus d'attributs de ce type.

Exemple :

entités	attributs de type P O ou R E	poids
E 1	< a b c d >	4
E 2	< e f >	2
E 3	< i >	1

1- proposer de placer sur l'écran l'entité E1
2- valider : oui / non
3- si réponse = non , proposer de placer l'entité E2
4- récupérer avec la tortue l'origine de l'entité.
donner une orientation par rapport à l'axe OY.
donner la forme géométrique de l'entité et les paramètres qui lui sont liés
5- passer à la prochaine entité après validation

Nous terminerons en remarquant que les outils implantés permettent

l'extensibilité du système et notamment la prise en compte de nouveaux concepts dans le cadre du prototype.

Il serait intéressant de créer d'autres variantes du prototype pour tester différentes solutions et opter pour pour un choix de la mise en œuvre effective de M.A.R.S.

==========